UCLA
UCLA Electronic Theses and Dissertations

Title
Scalable and Analog Neuromorphic Computing Systems

Permalink
https://escholarship.org/uc/item/95p0x7ws

Author
Wan, Zhe

Publication Date
2020

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA

Los Angeles

Scalable and Analog Neuromorphic Computing Systems

A dissertation submitted in partial satisfaction of the

requirements of the degree Doctor of Philosophy

in Electrical and Computer Engineering

by

Zhe Wan

2020

© Copyright by

Zhe Wan

2020

ABSTRACT OF THE DISSERTATION

Scalable and Analog Neuromorphic Computing Systems

by

Zhe Wan

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Los Angeles, 2020

Professor Subramanian Srikanteswara Iyer, Chair

Recent developments in artificial intelligence (AI) have been possible due to the increased computing power of the hardware. However, the systems are mainly digital and are optimized for fast, accurate, and versatile computing. Analog computing systems are attractive for their energy-efficiency and throughput in AI applications. In this dissertation, we explore and optimize a conventional CMOS transistor, the charge-trap transistor (CTT), as an analog in-memory computing unit for neural networks. In addition, to adapt to the finite variation of the analog devices and circuits, we develop novel methods to characterize and improve the resiliency of neural networks deployed on analog computing systems. Furthermore, as the scaling of the network plays a crucial role in enhancing its capability, this dissertation evaluates advanced system scaling technologies to scale out the analog computing hardware in a scalable non-von Neumann architecture. Finally, our findings are brought together and realized by the hardware demonstration of an analog neuromorphic system. We conclude with the characterization result of the system and discuss several future directions for scalable and analog neuromorphic systems.

The dissertation of Zhe Wan is approved.

Jingsheng Jason Cong

Vwani P. Roychowdhury

Sudhakar Pamarti

Subramanian Srikanteswara Iyer, Committee Chair

University of California, Los Angeles

2020

To my parents

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to first thank my advisor, Prof. Subramanian (Subu) Iyer, for his support and guidance over the years. This work would not be possible without his vision. It was quite a journey for me to explore and a very wide range of research topics with Subu. These years have fulfilled my rather childish dream when I picked electrical engineering as my college major: to obtain the ability to appreciate and develop computing hardware, which entangled my life with video games.

I am also very grateful for the help and encouragement I received from my committee members: Prof. Sudhakar Pamarti, Prof. Vwani Roychowdhury, and Prof. Jason Cong. In particular, I learnt a lot from the many discussions I had with Prof. Pamarti during the tapeouts.

I have been lucky to have collaborators from the industry, in particular, Arvind Kumar, Kevin Winstel from IBM, John Barth from Invecas, and Toshiaki Kirihata from GlobalFoundries.

I enjoyed my graduate school years with many lab mates in CHIPS. I am very lucky to work as a member of the NeuroCTT team with Steven Moran, Premsagar Kittur, Xuefeng Gu, and Jonathan Cox. I also thank my other CHIPS colleagues: Siva Jangam, Krutikesh Sahoo, Yandong Luo, Meng-Hsiang (Andy) Liu, Yongxi Li, Pranshu Garg, Yu-tao Yang, Guangqi Ouyang, Saptadeep Pal, Dr. Adeel Bajwa, Prof. Takafumi Fukushima, 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 also like to acknowledge collaborators in academia, including Tianyi Wang, Yiming Zhou, Qiujing Lu from UCLA, and Prof. Janakiraman Viraraghavan from IIT Madras.

In the end, I would like to thank my parents, Dan and Minli. They make me who I was and support who I have become.

Vita

Sep. 2011 – Dec. 2013 Bachelor of Science, Electrical Engineering, UCLA

Sep. 2014 – Dec. 2017 Master of Science, Electrical and Computer Engineering, UCLA

Jun. 2015 – Dec. 2015 Coop Engineer, 3Di team @ Albany Nanotech, IBM

Dec. 2017 – Ph. D. Candidate, Electrical and Computer Electrical Engineering, UCLA

Publications

1. S. Moran, J. Cox, **Z. Wan**, R. Brewer, E. X. Zhang, B. Sierawski, J. Woo, and S. S. Iyer, "Impacts of Perturbation on a Charge Trap Transistor Analog Neural Network", GOMACTech-20, Microelectronics for a New Decade: Global Competition and Near-Peer Challenges, March 16-19, 2020, San Diego, CA. (Accepted)
2. X. Gu, **Z. Wan** and S. S. Iyer, Charge-Trap Transistors for CMOS-Only Analog Memory, IEEE Transactions on Electron Devices (2019).
3. M. Liu, A. Hanna, Y. Luo, **Z. Wan** and S. S. Iyer, Process Development of Power Delivery Through Wafer Vias for Silicon Interconnect Fabric, IEEE 69th Electronic Components and Technology Conference (ECTC), May 28-31, 2019, Las Vegas, NV.
4. **Z. Wan**, S. Moran, J. Cox, X. Gu, and S. Iyer, Characterization Approaches to Test the Robustness of Neuromorphic Systems, 2019 Government Microcircuit Applications and Critical Technology Conference, Albuquerque, NM, 2019.
5. **Z. Wan**, K. Winstel, A. Kumar, and S. Iyer, Low-Temperature Wafer Bonding for Three-Dimensional Wafer-Scale Integration, 2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S), Burlingame, CA, 2018.
6. **Z. Wan** and S. Iyer, Fine-Pitch Integration Technology for Cognitive System Scaling, 2018 Semiconductor Research Corporation Technical Conference (SRC TechCon) TechCon, Austin, TX, 2018.
7. **Z. Wan** and 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.
8. T. Takafumi, A. Alam, S. Pal, **Z. Wan**, S. C. Jangam, G. Ezhilarasu, A. Bajwa, and S. S. Iyer, "FlexTrate®" - Scaled Heterogeneous Integration on Flexible Biocompatible Substrates, Proc. of IEEE Electronic Components and Technology Conference (ECTC), Orlando, FL, USA, 2017, pp. 649-654. doi: 10.1109/ECTC.2017.226
9. A. Kumar, **Z. Wan**, W. Wilcke and S. Iyer, Towards Human-Scale Brain Computing Using 3D Wafer Scale Integration, ACM Journal of Emerging Technologies in Computing Systems (JETC) 13 (3), 45
10. Q. Wei, H. Qi, W. Luo, D. Tseng, S. Jung Ki, **Z. Wan**, Z. Göröcs, L.A. Bentolila, T. Wu, Ren Sun, and A. Ozcan, Fluorescent Imaging of Single Nanoparticles and Viruses on a Smart-Phone, ACS Nano DOI:10.1021/nn4037706 (2013)
11. Q. Wei, E. McLeod, H. Qi, **Z. Wan**, R. Sun, and A. Ozcan, On-Chip Cytometry using Plasmonic Nanoparticle Enhanced Lensfree Holography, Scientific Reports (Nature Publishing Group) DOI:10.1038/srep01699 (2013)
12. Q. Wei, E. McLeod, H. Qi, **Z. Wan**, R. Sun and A. Ozcan, Lensfree holographic cytometry using plasmonic nanoparticles, 2013 IEEE Photonics Conference (2013)

Chapter 1: Introduction

Despite the significant development of Moore's Law in the last few decades, biological systems, the brains in particular, still have superior *cognitive* capability and energy efficiency compared with the existing computers. The power consumption of the human brain is only about 20 Watts [Drubach 00], a fraction of the power requirement of modern computing systems. In contrast, the brain can work on many cognitive tasks unsolvable by computers. As a result, neuromorphic computing systems [Mead 89] have been inspired by neuroscience to improve the cognitive performance of the computing system and its energy efficiency. Among many of the neuromorphic algorithms, artificial neural networks (NNs) are inspired by the brain structure and proposed by McCulloch in 1943 to develop an understanding of neurophysiology and provide a new method for calculation [McCulloch 43]. In 1958 Rosenblatt proposed the perceptron model for pattern recognition while mentioning that "higher order functions" (e.g., speech, communication, thinking) might be achieved by more advanced models [Rosenblatt 58]. Unfortunately, there are no practical computing engines at that time, limiting the research to mathematical analyses.

The spotlight turned to the artificial NNs until the famous Alexnet [Krizhevsky 12] appeared and achieved much better cognitive performance (i.e., classification accuracy) than any other contemporary machine learning solutions on the ImageNet classification challenge (1000-class image recognition) [Deng 09]. The Alexnet was deployed on a commercial GPU, and was able to be trained on the ImageNet dataset in several days. Since then numerous applications and systems based on neural networks have been developed to advance artificial intelligence (AI) and

surpassed human performance in many challenges [Russakovsky 15, He 16, Shafiee 16, Real 17, Guo 17, Silver 17, Jouppi 17, Desoli 17, Chen 17, Zheng 18, LeCun 19, Cai 19].

Artificial NNs, including multi-layer perceptrons (MLP), convolutional neural networks (CNN), and deep neural networks (DNN), are intensively used by the industry for commercial applications [LeCun 15, Jouppi 17]. It is observed that artificial NNs tend to perform better as they expand in size. As shown in **Fig. 1. 1** (adapted from [He 16]), for a given image classification challenge, a larger network can achieve a lower error rate. Furthermore, as the challenge becomes more difficult (from 1000-class ImageNet to 10-class CIFAR-10 dataset [Krizhevsky 09]), a larger network is required to reduce the error rate to potentially match it to a simpler challenge. Therefore, as people try to use artificial NNs for more advanced problems, the size (size of layers and number of layers) of the artificial NNs is expected to grow. As a result, there is also an increasing demand for advanced hardware to reduce the energy efficiency gap between the brain the computing machines to calculate artificial NNs more efficiency in both time and energy.

The advanced hardware needs to be suitable for neuromorphic algorithms such as artificial NNs. And thus its design should consider for several properties:

(a) Distributed and parallel operations. Most computations of the artificial NNs can be distributed and parallelized (like the brain), especially for DNNs that have many computing operations independent of each other. These can be executed in parallel. Thus, the graphics processing units (GPUs) [Nvidia 15] become more favored as the primary workforce for artificial NNs computations than the central processing units (CPUs) because their architecture supports more cores per chip for distributed computing in parallel threads.

(b) Reduced complexity of instructions and dataflow. Artificial NN computation workload is mainly multiplication and addition with few other arithmetic operations (e.g., taking

maximum). Also, the data dependencies between operations are very predictable (almost no conditional decisions or loop structures). Instead of the x86 instruction set, many of the recent digital artificial NN accelerator designs adapt the RISC architecture [Desoli 17]. In flexible platforms such as field-programmable gate arrays (FPGAs), dataflow and resource utilization can be improved by hardware-software co-optimization, making the systems more efficient than general-purpose computing engines [Zhang 18]. Furthermore, application-specific integrated circuits (ASICs) provide the highest degree of freedom for optimized architecture and instruction set to maximize computing efficiency [Chen 17, Jouppi 17].

(c) Data-intensive computing: despite the simplicity in the instructions, artificial NN computations are very data-intensive. As a result, the von Neumann bottleneck [Backus 78] becomes more critical as the systems scale up, demanding more data to be moved across the memory hierarchy to the processing elements. Therefore, the gating factor of the speed and even energy efficiency of such computation becomes the utilization of memory bandwidth, instead of the arithmetic operations themselves [Zhang 15], which can be visualized using the roof-line model [Williams 09]. This also limits the marginal gain of "More Moore" for related systems and calls for innovation in the compute-memory interaction. To summarize, **Fig. 1. 2** shows the comparison of artificial NN computation performed in different platforms in terms of time efficiency (throughput) and energy efficiency.

Another field of growing interest is to use analog in-memory computing for artificial NN operations. Specifically, use Ohm's Law ($I = G * V$) for multiplication and conservation of charge for summation to compute the dot-product and vector-matrix multiplication (VMM). It also happens that computations of artificial NNs do not require high precision for cognitive applications [Rastegari 16], and thus costly high-precision analog circuits may not be required. In addition, due

to the fixed data flow and its simplicity in the organization, analog devices can be used to store the weight matrices, while the VMM can be done in the analog memory itself. It is estimated that the analog non-volatile memory (NVM) can be used to significantly increase the throughput and efficiency with respect to digital implementations (**Fig. 1. 2**). Many emerging analog NVM devices, such as phase-change memory (PCM) [Burr 16, Burr 14], conductive bridging RAM (CBRAM) [Valov 11], resistive RAM (RRAM) [Wong 12, Seo 11] and spin-transfer-torque RAM (STT-RAM) [Sengupta 16, Vincent 15] can be used in a crossbar array to implement analog in-memory computing of artificial NNs.

As previously mentioned, the development of artificial NNs depends on scaling, and therefore requires the hardware technology to be scalable, both in terms of scaling up (increasing computing density) and scaling out (increasing number of chips in a system). It is also important to evaluate how the scaling affects the performance of artificial NNs in terms of accuracy, as they might not require as much investment in the hardware for high precision as in general-purpose computations.

To indicate a viable path for advanced and efficient computing hardware for the artificial NNs, this dissertation proposes a design of scalable and analog neuromorphic computing system, leveraging the highly scalable (scaling-up) commercial CMOS technology, featuring the charge-trap transistor (CTT) [Gu 19] as the analog device. The system is also scalable for scaling out by leveraging novel system integration technologies. A neuromorphic engine for neural network inference, featuring the CTT, is designed and fabricated to evaluate the feasibility and performance of the CTT as an analog computing element.

The dissertation is organized as follows: Chapter 1 provides the motivation for scalable and analog neuromorphic computing systems.

Chapter 2 introduces a CMOS-compatible and commercially available device, the charge-trapping transistor (CTT), and demonstrates how it can be used and optimized as analog memory. CTTs are also benchmarked with other state-of-the-art analog memory devices for neural network computing engines.

Chapter 3 introduces a design for a CMOS-compatible analog in-memory computing engine for different types of neural networks, including multi-layer perceptron and convolutional neural networks, and how the system can be scaled out for very large neural networks.

Chapter 4 investigates the resiliency of neural networks due to the imprecise nature of the analog neuromorphic systems. As it identifies the degradation of network and network resiliency for larger networks, it also proposes a modified training method to enhance resiliency without any extra cost in the inference engine to clear the obstacle for scaled analog computing systems.

Chapter 5 discusses the challenges for the scaled-out systems for deep neural networks and evaluates novel fine-pitch integration technologies, such as the Si-interconnect fabric, for system scaling.

Chapter 6 shows the design of a CTT-based mixed-signal in-memory computing engine, the NeuroCTT, and the testing results from the fabricated chips.

Finally, this dissertation concludes with a summary of the presented findings and an outlook on the future of scalable and analog neuromorphic computing systems.

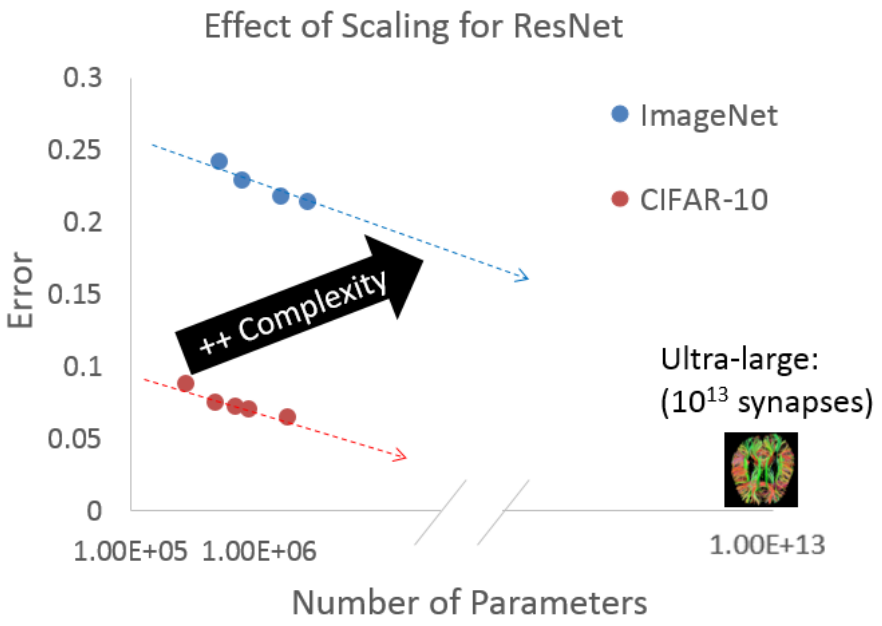


Fig. 1. 1 The effect of scaling of the ResNet [He 16] (a state-of-the-art neural network structure). The ImageNet challenge is an image classification problem for 1000 classes while CIFAR-10 is a simpler problem (10 classes only). For each of the problem, the error rate decreases as the network scales up.

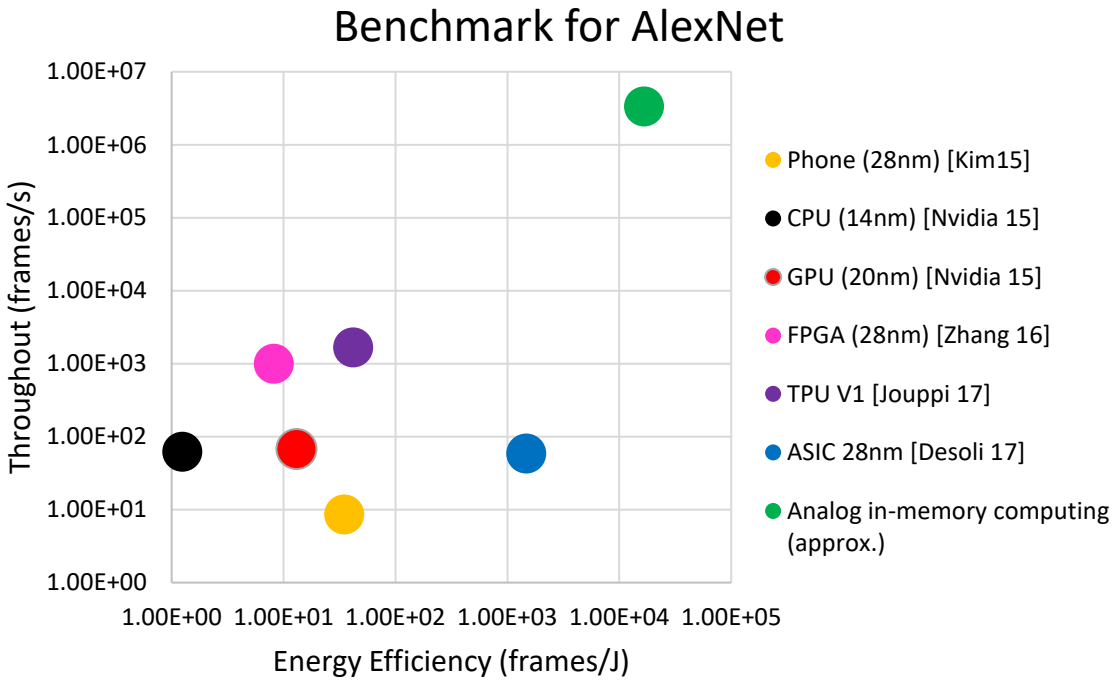


Fig. 1. 2 The throughput on image recognition application and the efficiency (defined as throughput per Watt) of some typical and novel architectures.

Chapter 2: Charge-Trap Transistor as CMOS Compatible Analog Memory

2.1 Introduction

Analog memories store a continuum of states which can be grouped into more than two "significantly separated" distributions (2 states will be a digital memory). In the electrical domain, the states usually are defined by current values at a given voltage bias, or threshold voltage values at a given threshold current. Although every real device may be treated like analog memory, observing the memory state inevitably digitizes the states under the limitation of the sensing circuit. As a result, digital memories are the ones whose sensing circuit gives two possible outputs for each cell, normally as "0" and "1". In the meantime, the sensing circuit of the analog memories should be able to distinguish more than two states, while the cost of the sensing circuit is balanced with the number of states for detection. In general, a device can be a good candidate of analog memory with the following characteristics:

(a) A wide memory window for a large number of states. Any programmable physical memory can be characterized for a probability density function (PDF) of its possible states. The information stored in this memory can be obtained by collecting the well-separated groups in the probability density function, which should be more in a wider memory window (i.e., the range of the states), as shown in **Fig. 2. 1**.

(b) Distribution of the states. When the state of the analog memory is interpreted by converting into digital values (more than 2), the relative positions and distribution of the states are irrelevant if the sensing circuit can distinguish the states. However, in the domain of analog in-memory computing, the stored values (e.g., current under a given bias)

themselves might be used directly as an operand for the computation based on natural laws (e.g., $I = I_1 + I_2$) instead of arithmetic laws. Therefore, the relative positions of the states should be compatible with the intended computation. In the case of linear operations, such as addition, uniformly distributed states across the memory window are preferred to reduce the potential error. The distribution of the states can then be modeled for evaluations at a higher level.

(c) Stable states (i.e., good data retention). The distribution of states needs to be evaluated for its retention to meet the requirement of some applications. It is possible to strengthen data retention by refreshing, as implemented for dynamic random-access memory (DRAM) and spin-transfer torque magnetoresistive RAM (STT-MRAM) [Shihab

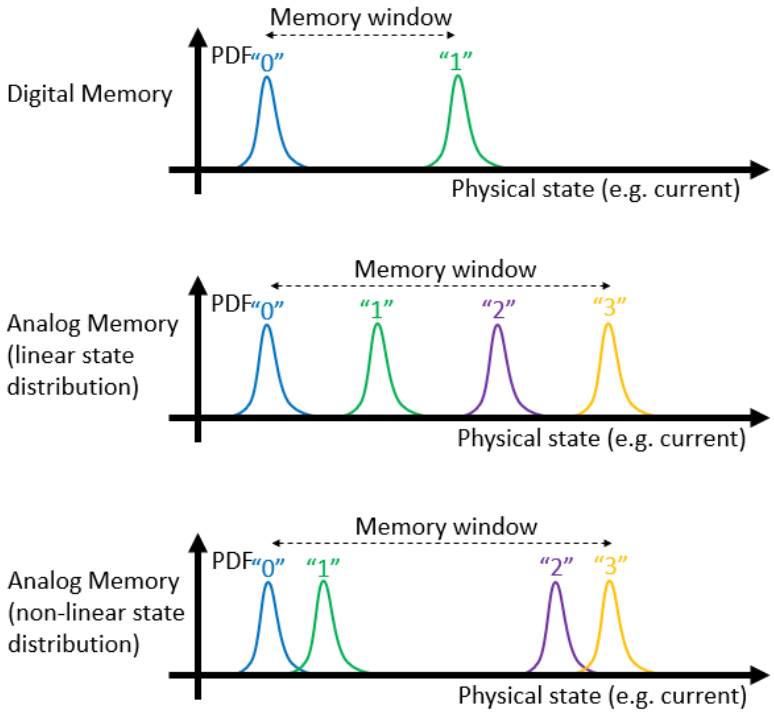


Fig. 2. 1 Top: memory window and state distribution of a digital memory. "0" state and "1" are achieved by writing "0" and "1" to the memory cell. Middle: memory window and state distribution of an analog memory with 4 states distributed linearly (roughly equally spaced). Bottom: memory window and state distribution of an analog memory with 4 states distributed non-linearly.

16]. However, the number of analog states increases the cost of the detection and refreshing circuit, which might significantly undermine the advantage of the system and should be avoided if possible.

(d) Analog memory used for frequently writing and re-writing needs to be qualified for its endurance and the write speed. For example, the neural network training operation requires sufficient symmetry of the incrementing and decrementing the memory value [Haensch 18]. However, these requirements are loosened if the application, such as neural network inference, requires only frequent reading but infrequent writing and re-writing.

While lots of materials and devices can exhibit properties of good analog memory described above, they need to be benchmarked against the digital counterpart at the system level, in terms of latency, energy efficiency and area cost. Therefore, it is, in general, preferred that the memory candidates leverage the core of the Moore's law - the CMOS technology. Many of the emerging analog memories today focus the R&D effort in the integration with the CMOS processes [Burr 16], which also makes it easier for the integration with existing silicon intellectual properties (IP) for optimized computing performance.

In this chapter, a CMOS-compatible analog memory, the charge-trap transistor (CTT), is characterized and optimized for a competitive analog non-volatile memory.

2.2 Characterization of CTTs for Analog Memory

2.2.1 Basic Operations

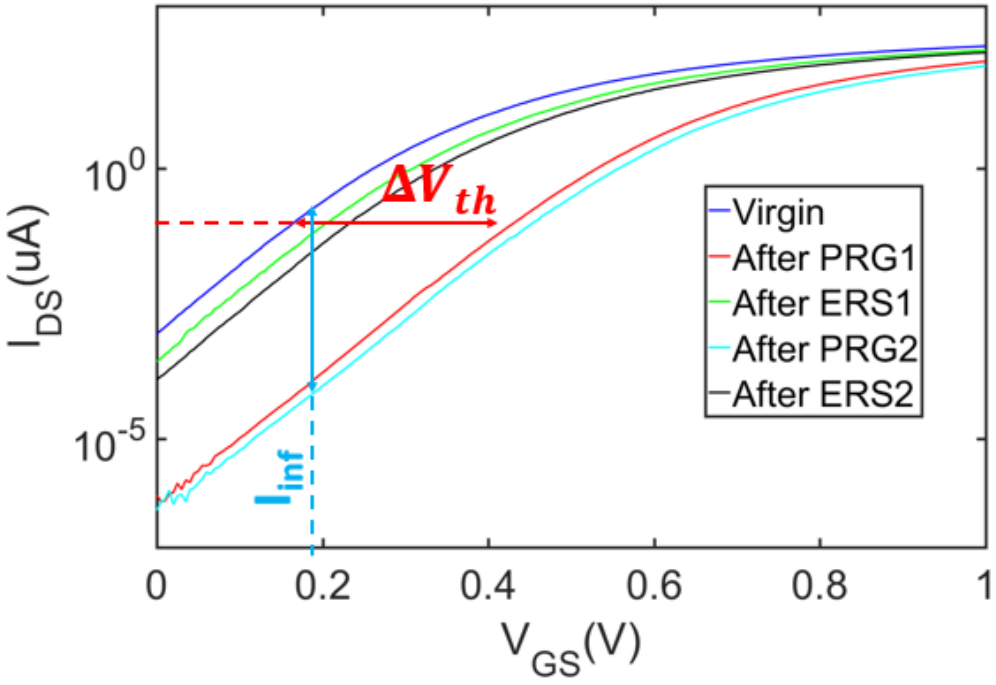


Fig. 2. 2 The charge trapping and de-trapping of the CTT devices is manifested by the shift in their threshold voltage after programming (PRG) and erasing (ERS) events. The shift in threshold voltage provides a significant dynamic range in the sub-threshold drain current of the device. The subthreshold current used for most characterizations is the inference current I_{inf}, measured at V_G = 0.2V, V_D = 0.05V, V_S = V_{Sub} = 0V. The figure shows the I_{DS}-V_{GS} characteristics after the event in the order: PRG1, ERS1, PRG2, ERS2

The charge-trap transistor (CTT) is a n-type metal-oxide-semiconductor field-effect transistor (nMOSFET) that uses high dielectric-constant (high-k) materials as its gate dielectric. Because such dielectric material can replace silicon oxide as a more reliable gate dielectric and is supported by the high-k metal gate (HKMG) process, it is ubiquitous in all logic transistors since the 45nm technology node [Ghani 00, Auth 08]. During HKMG fabrication, oxygen-induced traps are formed in the gate dielectric. These traps can capture electrons in the channel and release the captured electrons to the channel through tunneling [Cartier 06]. The trapped charge does not

require energy to remain trapped, and therefore the CTT device as analog memory is non-volatile. A high gate-to-source bias (V_{GS}) with high drain current (I_D) helps to support significant and stable charge trapping for programming (PRG), while a high negative gate-to-source bias can de-trap the charges for erasing (ERS). As the trapped charges generate an electric field in the channel, a shift in the MOSFET's threshold voltage can be observed (**Fig. 2. 2**). Such phenomenon has been shown in 32nm, 22nm bulk and silicon-on-insulator (SOI) structures, 14nm FinFET [Khan 16], and 7nm FinFET [Khan 19]. For the application of digital multi-time programmable memory (MTPM), the retention of 10 years at 105°C has been shown in 32nm and 22nm technology nodes [Viraraghavan 16]. In this chapter, all device characterization data are based on standard logic devices from Globalfoundries 22nm fully-depleted SOI (22FDX) technology [Carter 16] with device width of 428nm unless otherwise specified.

By using the sub-threshold drain current as the analog information stored in the CTTs, CTTs become analog memory with a substantial (up to 1,000) dynamic range [Gu 19]. For the characterization of the CTTs, we define its threshold voltage V_{th} as the gate voltage when V_D = 50mV, V_S = 0V and I_D = 200nA, and define the inference current (I_{inf}) as the sub-threshold drain current at V_{GS} = 200mV and V_{DS} = 50mV (**Fig. 2. 2**). In all experiment results shown, I_{inf} is measured with the high-resolution single measurement unit (HRSMU) in the Keysight B1500A parameter analyzer (1fA to 100fA resolution) [Keysight 18]. The V_{th} is extrapolated with the subthreshold slope measured in the I_D-V_G sweep at V_{DS} = 50mV, V_{GS} = 0V, 50mV, 100mV,…,300mV (or higher, depends on the experiment) when the device is in the subthreshold region (i.e., the subthreshold slope $\partial \log(I_D)/\partial V_G$ is almost a constant). As logic devices, CTTs are designed for a nominal voltage of 0.8V, with a +/- 10% tolerance from the foundry's specification. In **Fig. 2. 2**, the device PRG is done by using a 0.5ms of 2.7V V_{GS} pulse with a 1.5ms of 1.2V V_{DS}

pulse, which starts 0.5ms earlier and ends 0.5ms later than the V_{GS} pulse. Also, the ERS is done by using a 0.5ms of -2.7V V_{GS} pulse while V_{DS} is grounded.

2.2.2 Charge Trapping in Bulk-Oxide and Interfacial Traps

The location of the traps for the charge trapping in the device depends on the relative bias points of the device. In this section, we discuss the charge trapping in the bulk-oxide traps and interfacial traps. A commonly studied case on such logic devices is the negative or positive-bias temperature instability (NBTI or PBTI) [Jeppson 77, Khan 15], where the V_{th} of the device can shift due to a negative or positive gate bias. For nMOSFETs, PBTI is more significant due to the positive V_{GS} used during most operations. PBTI induces the V_{th} shift by trapping charges to the bulk-oxide traps. While PBTI can also induce the V_{th} shift, it is not as significant as the proposed PRG method, which also mainly traps charge in the bulk-oxide but enhanced by the resistive self-heating from the large drain current [Khan 15]. **Fig. 2. 3** shows the V_{th} shift induced by both PBTI (high V_G, with $V_S = V_D = 0V$ for 20ms) and the CTT PRG method (high V_G, with $V_D = 1.2V$, $V_S = 0V$ for 5ms). The amount of charge-trapping by PRG is almost three times more efficient and has good retention. It has been reported that the self-heating enhanced CTT PRG method can retain 70% of the trapped charge at 105 °C for ten years [Viraraghavan 16].

Another method to leverage charge trapping for device V_{th} shift is to use hot-carrier injection (HCI), which requires very high V_{DS} to generate interfacial traps between the oxide and the drain side channel [Hu 85]. It has been shown that HCI can be used to shift V_{th} significantly with very good retention (< 10% charge loss in 10 years at 125 °C) [Ma 19]. But it can only be de-trapped by high temperature and long time (> seconds) annealing [Pobegen 13]. The bulk-oxide trapping, on the other hand, can be reversed without heating (resistive self-heating during negative

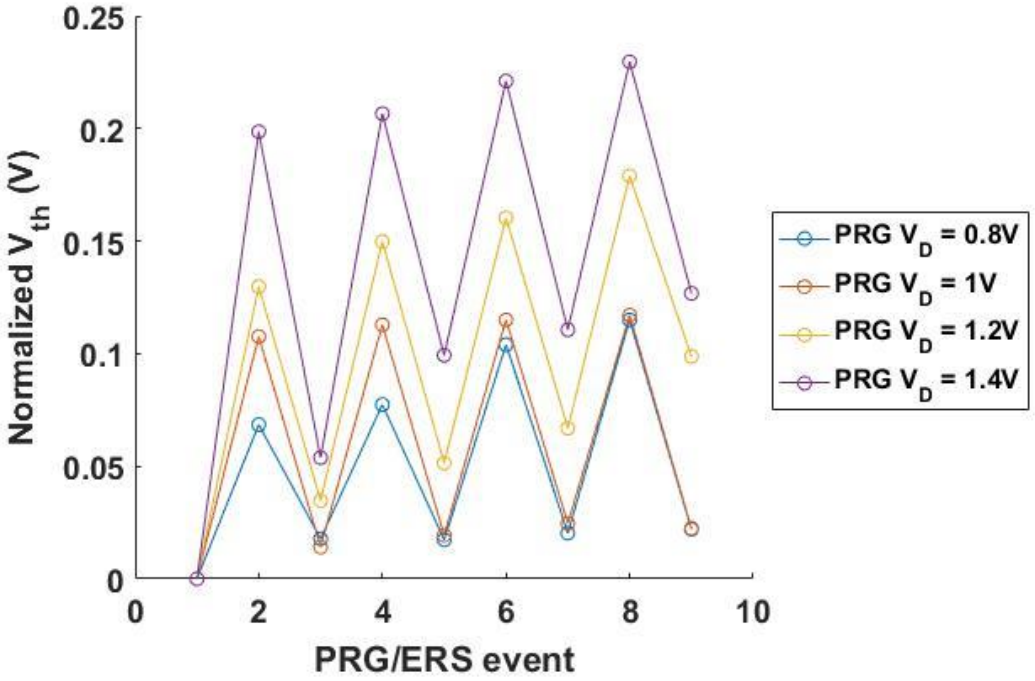


Fig. 2. 4 The V_{th} shift of CTTs during alternating PRG and ERS events under the same series of V_G pulses but with different V_D.

V_{GS} is impossible due to the off-state of the device) by applying high V_{GS}, with the drain floated or shorted to the source.

Fig. 2. 4 shows the CTT's V_{th} shift during alternating PRG and ERS events, using the same set of V_G pulses but different V_D magnitude. It shows that when higher V_D is used (i.e., more interfacial traps generated by larger HCI), the V_{th} shift is larger, and the remaining V_{th} change (i.e., trapped charge) after ERS is more significant. **Table 1** summarizes the three different methods to induce charge-trapping in the n-type MOSFETs.

Therefore, to use CTTs as an analog re-writable non-volatile memory, the self-heating enhanced charge-trapping in the bulk-oxide traps is the best mechanism. To avoid the effect from the non-reversible (at room temperature) interfacial traps, the CTTs should first be initialized by a

PRG/ERS cycle. Then the memory window of the CTT becomes more consistent for a multi-time programmable analog memory.

Table 1 Summary of the PBTI, self-enhanced CTT PRG and HCI induced charge trapping

	V_{GS}	V_{DS}	Major traps used/generation	Write time	Retention	Re-writability
PBTI	High	Zero	Bulk-oxide traps	Moderate (~sub-ms)	Poor	Very good
Self-heating enhanced CTT PRG	High	Moderate	Bulk-oxide traps and interfacial traps	Short (~us)	Good	Good
HCI	Moderate	High	Interfacial traps	Long (> ms)	Very good	N/A without annealing

2.2.3 Gate Leakage

Gate leakage of the CTTs changes as the number of total traps and occupied traps changes due to the PRG and ERS operations. **Fig. 2. 5** shows the device I_D-V_G, I_G-V_G characteristics at the virgin state, and after a series of PRG pulses. It shows that as the amount of the trapped charge increases, the gate leakage increases due to the increasing trap-assisted tunneling and barrier distortion [DiMaria 95]. This manifests as the measured gate leakage of the CTT increases after PRG and decreases after ERS. In addition, for the same amount of trapped charge (i.e., V_{th}), the gate leakage increases if the same V_{th} is obtained after more PRG/ERS events. This suggests that PBTI-like stress could generate traps in the bulk-oxide, which increases the gate leakage as the stress-induced leakage current (SILC) [Cartier 09]. **Fig. 2. 6** shows the gate leakage (V_{GS} = 0.8V, V_{DS} = 0.05V) of a CTT as it experiences alternating PRG and ERS events.

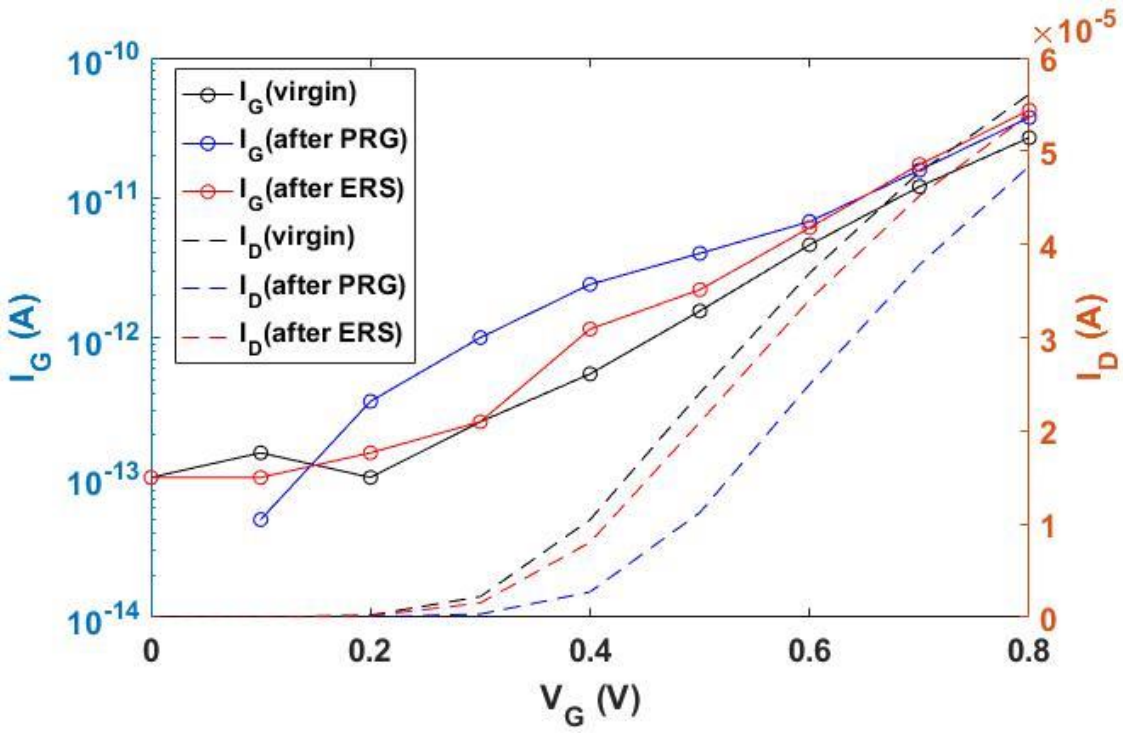


Fig. 2. 5 The change of the I_D-V_G and I_G-V_G characteristics before and after CTT PRG and CTT ERS which is after PRG. The gate leakage increases after PRG, and decreases after ERS.

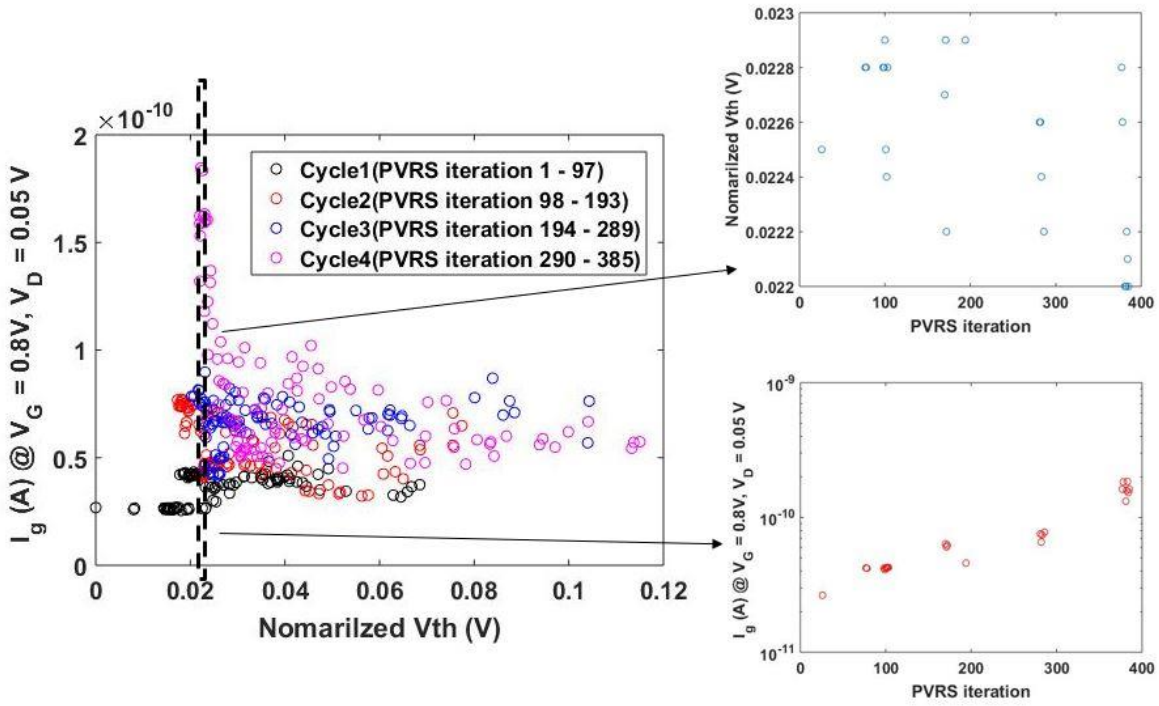


Fig. 2. 6 Left: the gate leakage (V_{GS} = 0.8V, V_{DS} = 0.05V) of the CTT with respect its V_{th} at its various analog states during 4 PRG/ERS cycles from 385 PVRS events. **Right:** Although the amount of trapped charges can be controlled to the same amount by repeated PRG and ERS cycles, the gate leakage gradually increases due to the traps generated during PRG. For example, the data points between $0.022V < \Delta V_{th} < 0.023V$ at different PVRS iterations are extracted (upper right). Although they have similar V_{th}, the gate leakage gradually increases (lower right).

2.2.4 Closed-Loop Verification Requirement

The device variation due to the manufacturing process generates a variation on the memory window of the CTTs. For example, the virgin devices have a variation of the $I_{inf,}$ as shown in **Fig. 2. 7** for 5 CTT arrays in different dies. While the device variation can be reduced by increasing the size of the device to reduce the effect of dopant variation, it increases cost in area. In addition, for different devices, the change of I_{inf} due to the same PRG operations can have a spread distribution, as shown in **Fig. 2. 8,** similar for ERS, as shown in **Fig. 2. 9**. Therefore, to write the CTT to accurate analog states closed-loop in a read-write-read process (**Fig. 2. 10**) is required.

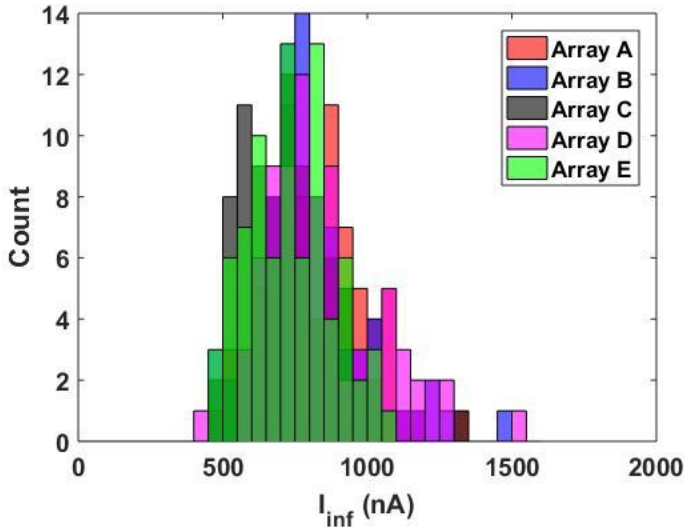


Fig. 2. 7 The distribution of the I_{inf} values of virgin devices in 5 different CTT arrays. All arrays are on different dies.

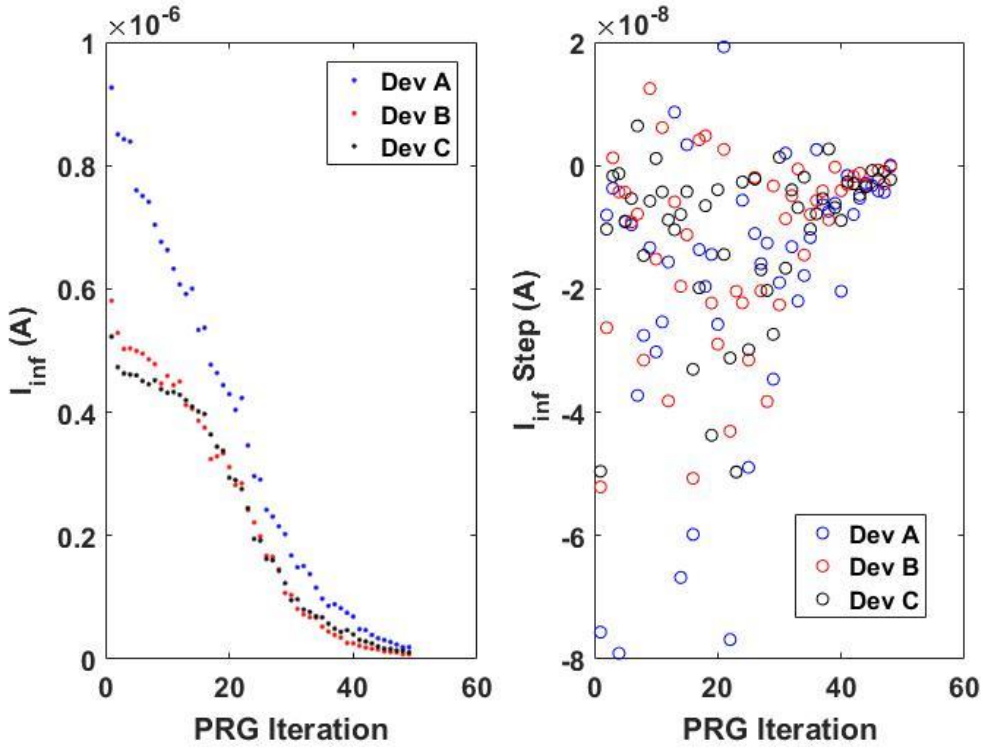


Fig. 2. 8 I_{inf} evolution of 3 different CTTs (same design parameters) during the same series of PRG events shows large variation of the CTT PRG response from both the variation of the virgin state (left) and the change of I_{inf} in terms of step size (right). The positive step in the right figure is due to the device relaxation between the PRG event and the measurement after it, which will be discussed in a later section.

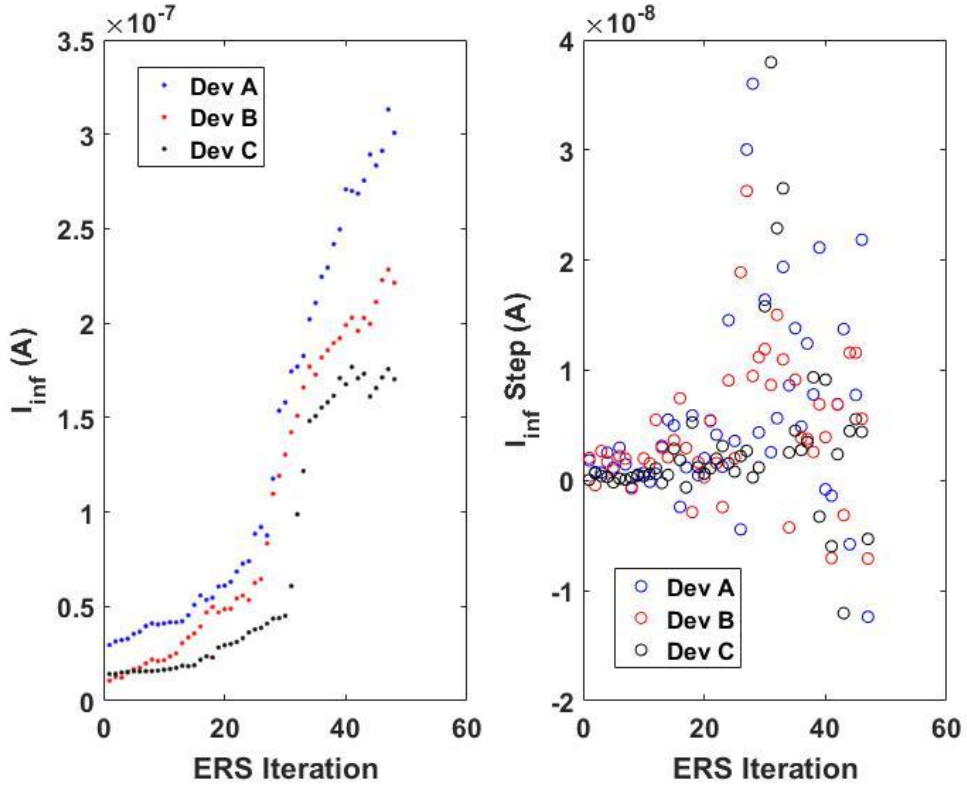


Fig. 2. 9 I_{inf} evolution of 3 different CTTs (same design parameters) during the same series of ERS events shows large variation of the CTT ERS response from both the variation of the virgin state (left) and the change of I_{inf} in terms of step size (right). The negative step in the right figure is due to the device relaxation between the ERS event and the measurement after it.

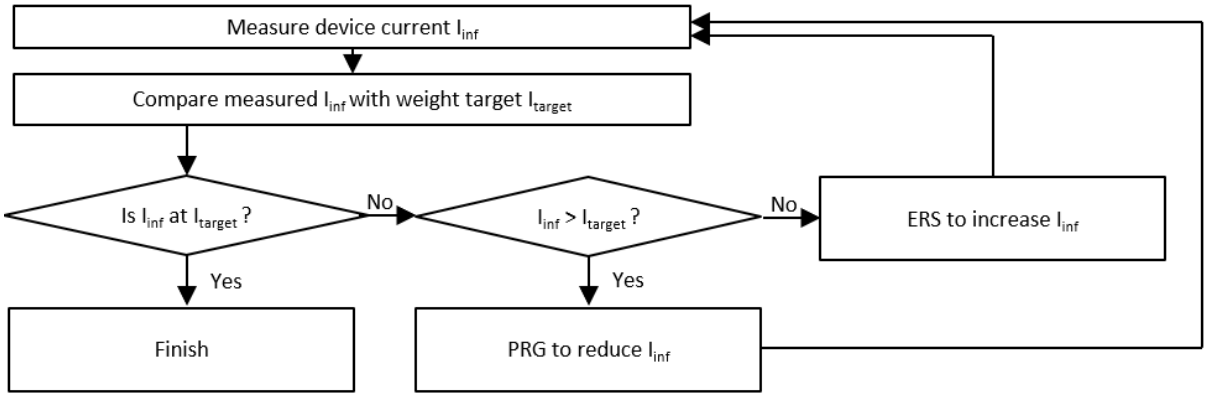


Fig. 2. 10 the closed-loop read-write-read flow to write CTT I_{inf} to target value (i.e.,I_{target}) using PRG to reduce current I_{inf} and ERS to increase current ERS.

2.3 Optimization of CTTs for Analog Memory

2.3.1 Distribution of Analog States

I_{inf} was the focus of characterization from the section. Since I_{inf} is a real physical quantity, it can be used for addition and multiplication (i.e., multiple additions) in analog computation. To use the CTT as an analog computing element, continuously distributed values of I_{inf} are preferred as stored analog information for direct linear computations. **Fig. 2. 11** shows the change of I_{inf} when constant PRG and ERS pulses are applied successively to the CTT device. The change of the I_{inf} is larger in the first few PRG/ERS events than the later ones until the magnitude of V_G is increased. This can be explained by the saturation of the existing traps in the high-k gate dielectric at the energy levels accessible by the electrons at a fixed V_G. Although the stress can generate new traps [Cartier 09], it is not significant at a relatively low bias (e.g., V_G = 1.5V, 2.1V) for short pulses (< 1ms). If the PRG pulse is increased for more trap generation, the write speed will be reduced.

Therefore, to distribute I_{inf} more uniformly, the method of pulsed gate voltage ramp sweeps (PVRS) [Kerber 09] is used. For PRG, the magnitude of the gate bias gradually increases, and the drain voltage is held constant to induce self-heating by I_D and irreversible HCI-induced charge trapping (when V_{DS} is too high), as previously discussed in Section 2.2.2. Similarly, negative V_{GS} pulses are used for fine-step ERS operations while V_{DS} = 0V. This is to gradually access traps at higher energy levels for charge trapping/de-trapping to get a smoother increment/decrement of the trapped charge. PRG and ERS V_{GS} pulses of a width of 0.5ms are used, ranging from 1.5V to 2.7V for PRG and -1.5V to -2.7V for ERS, with 50mV or 100mV increment/decrement as shown in**Fig. 2. 11**. V_{DS} pulses have a constant magnitude of 1.2V for PRG and 0V for ERS, with a1.5ms pulse width, rising 5ms before and falling 5ms after the V_{GS} pulse. By using the PVRS method instead

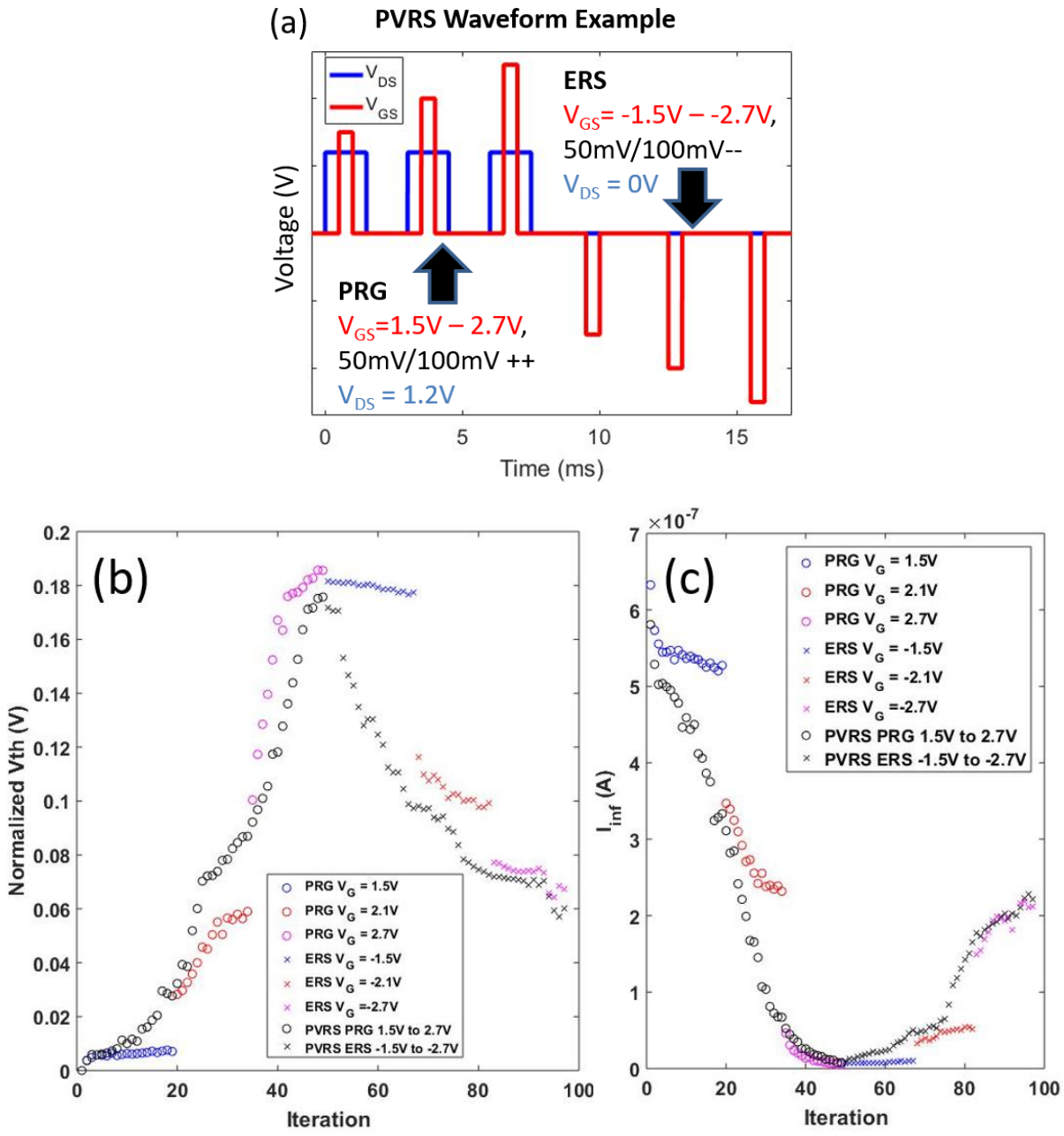


Fig. 2. 11 (a) the waveform shapes and the magnitude used for the PVRS experiment. Each PRG event uses 1.5ms pulsed V_{DS} = 1.2V, which starts before and ends after the 0.5ms pulsed V_{GS} whose magnitude ramps from 1.5V to 2.7V, and with V_{DS} = 0V, from -1.5V to -2.7V for ERS. (b) normalized V_{th} (i.e.,ΔV_{th}) of a CTT cycled by using multiple PRG/ERS pulses with the constant V_G magnitude in each cycles, changed for +/-1.5V (blue circles and crosses), +/-2.1V (red circles and crosses), +/-2.7V (magenta circles and crosses) in one cycle, and the ΔV_{th} of a CTT cycled by using pulsed voltage ramped sweep (PVRS) which has multiple PRG/ERS pulses with increasing V_G magnitudes, from 1.5V to 2.7V in each cycles (black circles and crosses). (c) the I_{inf} from the same experiment in the bottom left figure, showing a much finer I_{inf} steps using PVRS method.

of constant pulse trains, I_{inf} states with fine granularity can be achieved and can be finer using finer steps of the V_{GS}. **Fig. 2. 12** shows the achieved I_{inf} states for random targets in the device memory window by using PVRS PRG with 20mV and 40mV increment of V_G per step from 1.5V to 2.7V. Because the stop condition is to be equal or less than the target, all achieved values are below the target. The mean error is $-4.31nA$ for 20mV steps and $-8.64nA$ for 40mV steps. **Fig. 2. 13** shows the histogram of the step sizes in both experiments, showing that smaller steps in V_G produce smaller steps in I_{inf}. However, although the steps are fixed for V_G, the step size can still vary significantly for I_{inf}, providing another reason for using the closed-loop verification, as discussed in Section 2.2.4.

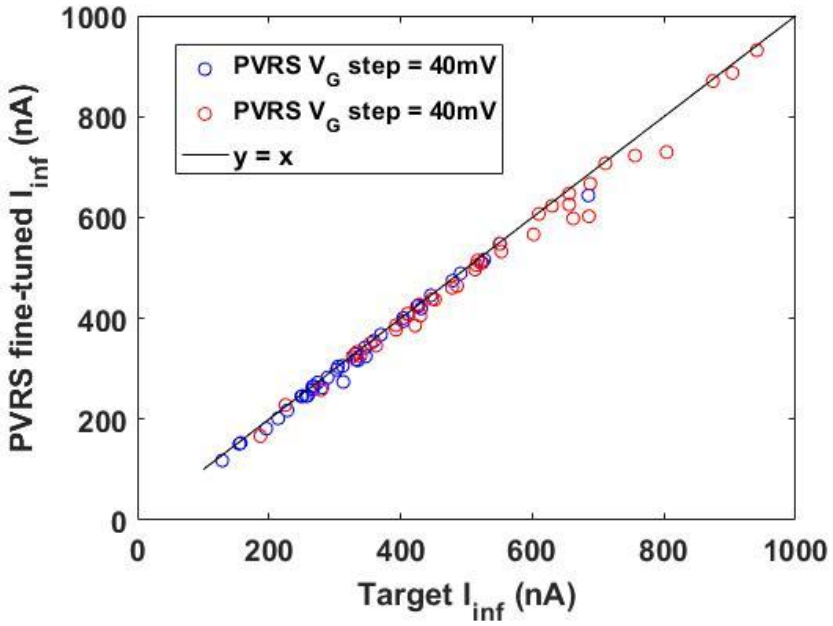


Fig. 2. 12 the target I_{inf} values (generated randomly in the memory window) with respect to the achieved I_{inf} values by using PVRS PRG, with V_D = 1.2V, V_G = 1.5V to 2.7V and t = 0.5ms for each PRG pulse. PVRS stops only when the measured I_{inf} (right after last PRG event) is lower than the target (i.e.,all achieved points are below the reference line of y = x). Two cases of 20mV and 40mV increment of V_G per step are shown. The mean error is -4.31nA for 20mV step and is -8.64nA for 40mV across 20 CTT devices.

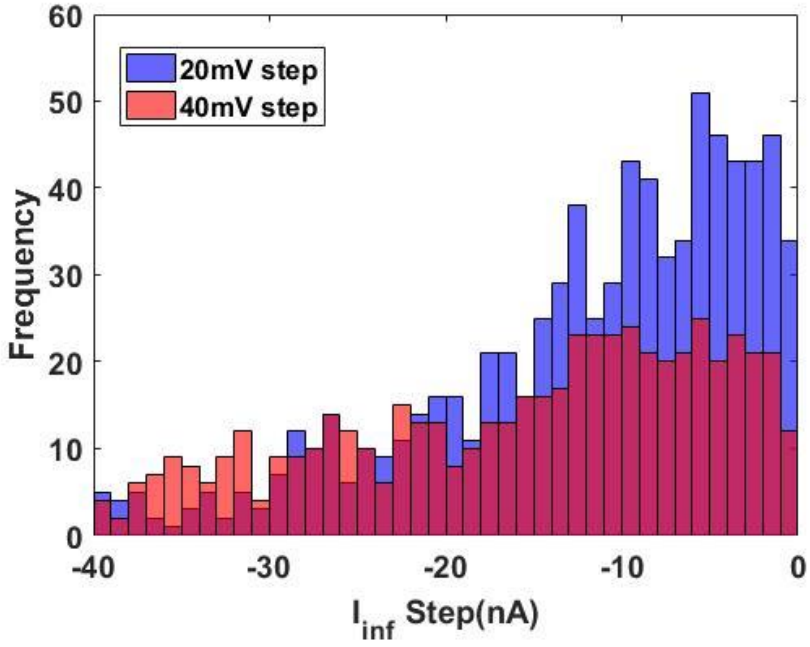


Fig. 2. 13 the step of I_{inf} achieved by the PVRS sequences with 20mV and 40mV V_G steps in the experiment shown in **Fig. 2. 12**.

2.3.2 Analog Data Retention

Retention is a significant challenge for the CTTs. Because the trapped charges can be emitted spontaneously from the oxide, threshold voltage, and the read current of a CTT with trapped charge can change significantly. Since the physical quantity of interest, I_{inf}, is at the subthreshold, it is very sensitive to the loss of trapped charge, roughly in an exponential relationship

$$I_{inf} \propto \exp\left(\frac{qV_{G,inf} - qV_{th}(Q_{trapped})}{k_b T}\right)$$

where $V_{G,inf} = 0.2V$ is the gate voltage for I_{inf} measurement, $V_{th}(Q_{trapped})$ is the threshold voltage as a function of the trapped charge $Q_{trapped}$, k_b is the Boltzmann constant, and

T is the temperature. The amount of charge loss, both for the bulk-oxide traps and the interfacial traps, as a function of time t, can be modeled as

$$V_{th}(t) \propto \exp\left(-\frac{t/\tau_{trap}}{T}\right)$$

where τ_{trap} is the time constant for the emission of trapped charge. τ can be different for traps of different types and at different energy levels. The charge loss is accelerated by elevated temperature. **Fig. 2. 14** shows the percentage of trapped charge loss in programmed CTT devices at room temperature and at elevated temperature (85C) for 1, 10, and 100 hours. This shows that CTTs are most suitable for low-temperature environments.

To improve charge retention, **Fig. 2. 15** shows that the amount of charge loss at a constant temperature (e.g., room temperature) can be empirically predicted and compensated by over-PRG of the device, where the magnitude of the relaxation (i.e., compensation) ΔI_{inf} is a function of its final I_{inf} state [Gu 19]. In addition, resistive heating during read operations should be minimized to improve the retention of the stored data.

To apply CTTs in high-performance systems, the target resolution of the CTTs needs to be reduced according to the retention requirement. The amount of required resolution reduction can be estimated by modeling the relaxation over time. **Fig. 2. 16** shows the relaxation of the CTT devices at both room temperature and elevated temperature (85C). The discrepancy between the linearly fit model for the room-temperature and baked environment is the larger intercept at the baked environment due to the systematic acceleration of the relaxation. Therefore, the relaxation should be modeled not only with respect to the analog state (I_{inf}) of the device, but also to the time

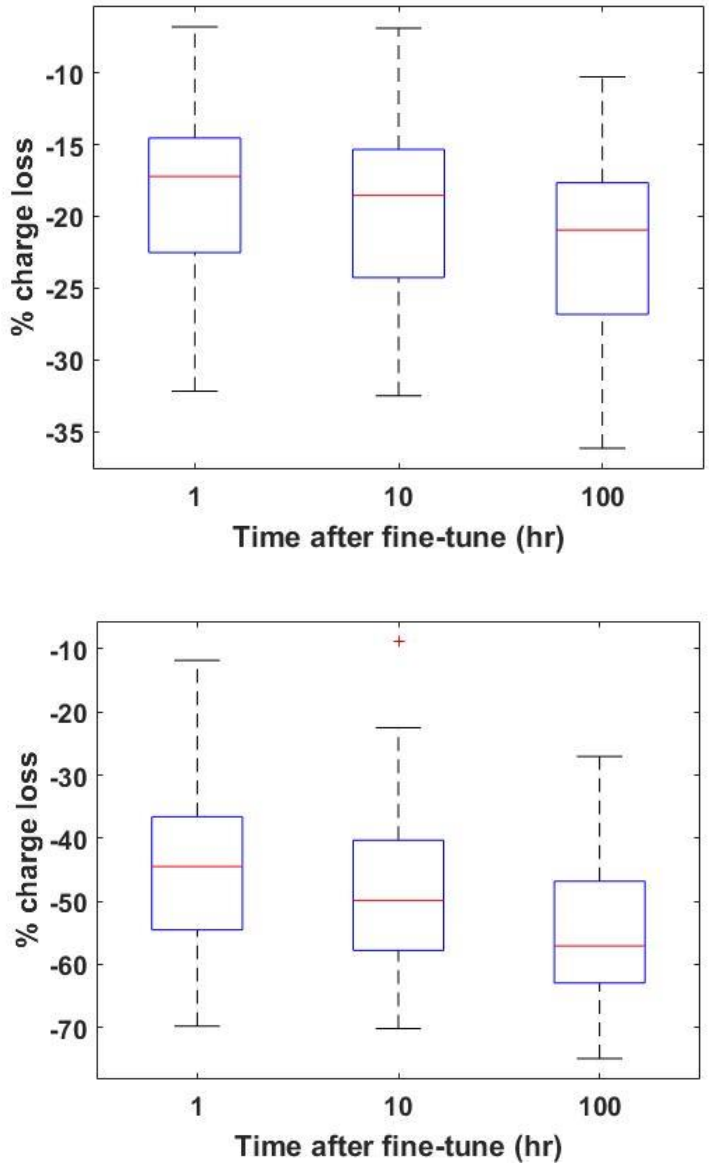


Fig. 2. 14 Box and whisker plot of 40 devices programmed to random V_{th} values and the retention of the trapped charge over time at room temperature (top) and at 85 degrees Celsius (bottom). The de-trapping of the trapped charge is accelerated greatly at the elevated temperature.

for which is retention is measured. The new model is modified based on the previous model [Gu 18], so that:

$$\Delta I_{inf} = -0.075 * I_{inf,RA} + B(t,T)$$

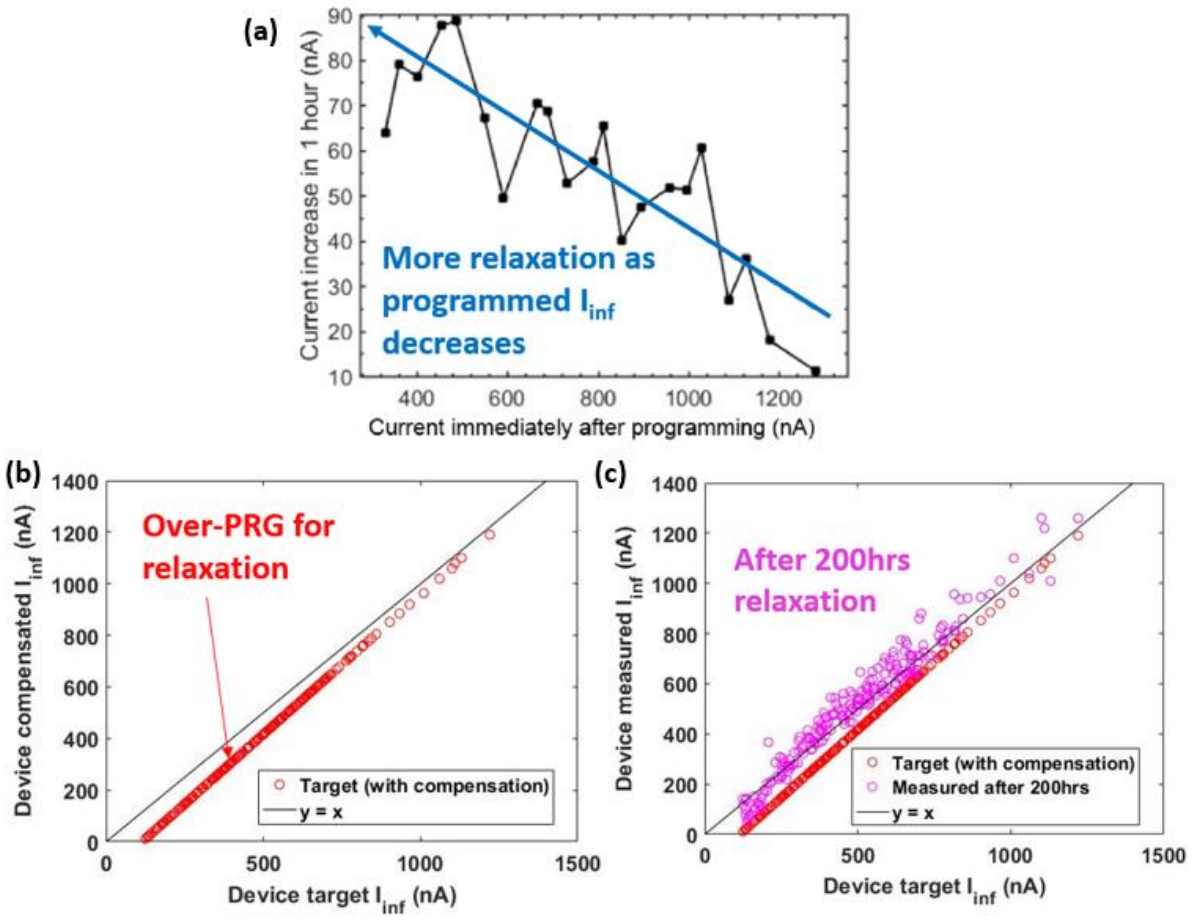


Fig. 2. 15 (a) Relaxation of the I_{inf} current after PRG is measured and can be modelled using linear regression as a function of the I_{inf} measured right after the last PRG event (adapted from [Gu 19]). (b) The relaxation model can be used to offset the PRG targets so it over-PRG the device to compensate for relaxation. (c) The device measured after 200 hours of relaxation (at room temperature) approaches to the target line (measurement = target). While the model in (a) is extracted based on 1 hour relaxation, it can be applied for compensating longer relaxation time.

where $I_{inf,RA}$ is measured right after the last PRG event of the CTT and ΔI_{inf} is the relaxation measured with respect to $I_{inf,RA}$. $B(t,T)$ represents the average magnitude of the relaxation as a function of time (t) and temperature (T). In the previous model [Gu 18], $B(t,T) = 114.5nA$ is constant. The linear fit of the function $B(t,T)$ with respect to log(t) is shown in **Fig. 2. 17** which is a stronger function of time (i.e., larger $\frac{\partial B}{\partial t}$) when temperature is high. In this semilog plot, slope and intercept of the linear model at the room temperature is $k_{25C} = 2.19$ (nA/$\log_{10}$(hours)) and

$b_{25C} = 47$ (nA). At the baked temperature (85 degrees Celsius), $k_{85C} = 10.4$ (nA/log$_{10}$(hours)) and $b_{25C} = 157$ (nA).

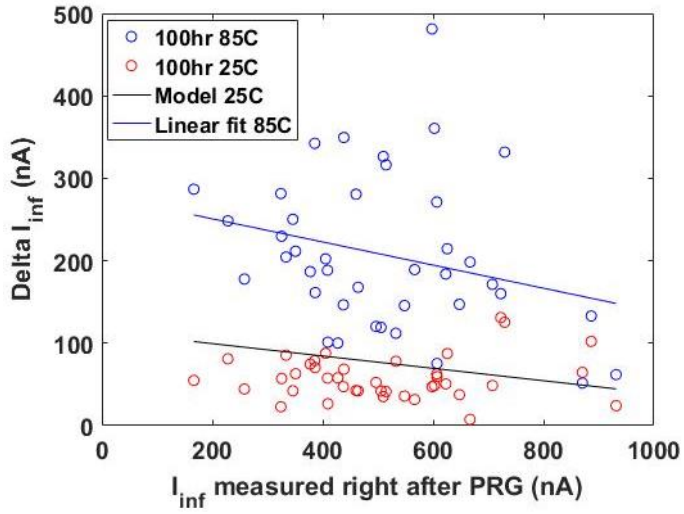


Fig. 2. 16 Amount of I_{inf} change (Delta I_{inf}) of the CTT devices at room temperature and at elevated temperature (85C). The room temperature retention is well predicted by the previous model [Gu 18]. The major difference between the linear models is their intercept values, which need to be adjusted for the systematic acceleration of charge loss due to baking.

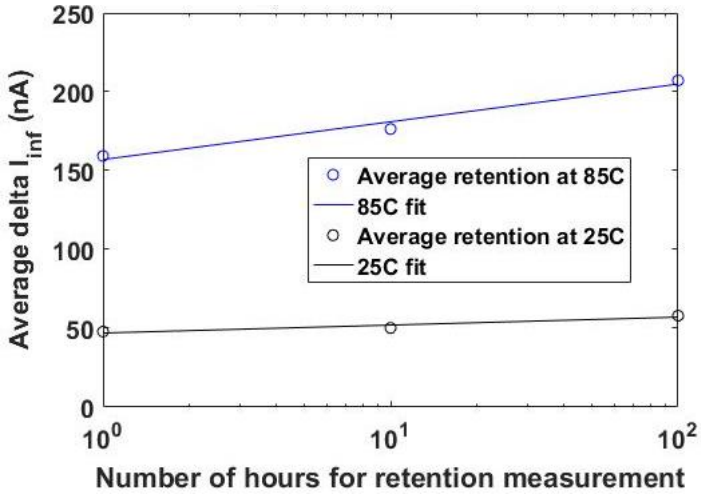


Fig. 2. 17 Linear fit of the intercept of the delta I_{inf} model, $B(t,T)$as a function of time (log) at different temperature. $\frac{\partial B}{\partial t}$ is larger at higher temperatures.

2.3.3 In-Array Write/Erase Parameters

Based on the previous observation in Section 2.2.2, the accessibility of the bulk-oxide traps depends on the magnitude of V_G. In contrast, the accessibility of the interfacial traps depends on both V_G and V_D. Since the bulk-oxide traps are the preferred traps for reversible trapping and de-trapping of the charge, it is ideal to have a low V_D (that is still sufficient for self-heating) and high V_G (without breaking the gate dielectric) for a single CTT device to work as an

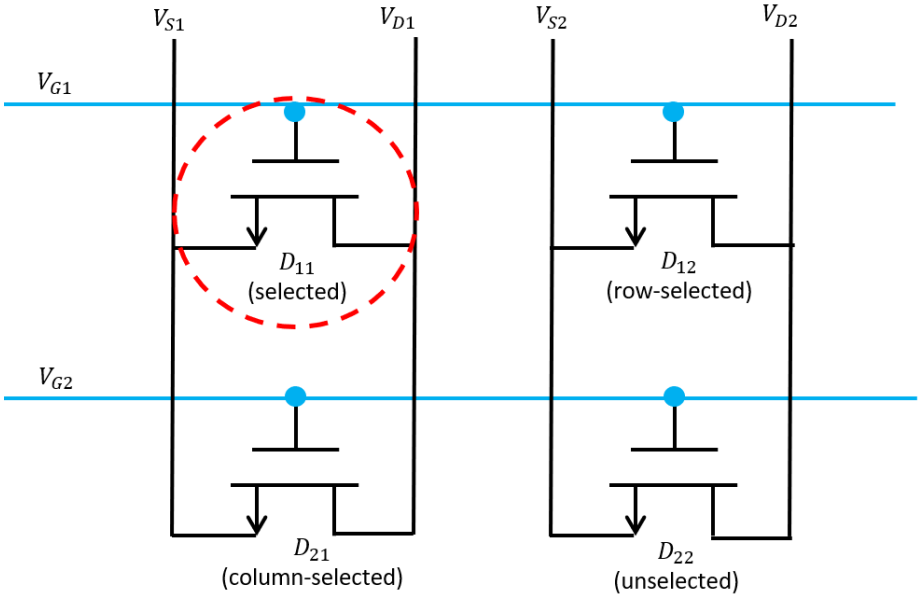


Fig. 2. 18 Schematic of a CTT array. When one device (e.g., D_{11}) is selected for some operation (e.g., PRG), the devices in the same row (D_{12}) and column (D_{21}) are half-selected.

analog memory. However, a high V_{GD} and V_{GS} can become problematic when PRG operation is done in an array. Suppose the array structure is designed, as shown in **Fig. 2. 18**, where gates of the CTTs in a row are shorted, and the drain/source terminal of the CTTs in a column are shorted, respectively. When device D_{11} is selected for PRG, the devices sharing the gate or source/drain are half-selected. In addition, the voltages to apply for all terminals should not only provide good programmability to the selected device, but also ensure the half-selected and unselected devices are not affected by this PRG operation.

For the target device (i.e.,D_{11}), assign V_{S1} = 0V as a global reference, high V_{G1} and high V_{D1} is required for the PRG operation. As a result, V_{DS} of the column-selected device (i.e.,D_{21}) will also be high. To avoid HCI-type charge trapping in D_{21}, the gate of the other row must be low (i.e.,$V_{G2} < V_{th2}$), but not too low to induce ERS behavior (i.e.,$V_{G2} - V_{D2} > -1.5V$). Based on these restrictions, $V_{G2} = 0V$ is an option for the unselected rows' gates. For the row-selected devices (i.e.,D_{12}), low device current is required to avoid PRG behavior, and therefore $V_{S2} = V_{D2}$. Since V_{G1} is high and $V_{G2} = 0V$. V_{S2} should be about the middle of these two levels to avoid both PBTI at the row-selected devices and ERS behavior at the unselected devices (i.e.,D_{22}).

Based on these restrictions, we select $1.5V < V_{G1} < 2.7V$ during the PVRS-PRG scheme, $V_{D1} = 1.2V, V_{G2} = 0V, V_{S2} = V_{D2} = 1.2V$. The drain voltages are always first raised before the V_{G1} rises to avoid PBTI on the row-selected devices during the transient time. A similar analysis can be done for PVRS-ERS, and we select $0V < V_{G1} < 1.2V$, $V_{D1} = V_{S1} = 2.7V, V_{G2} = 1.5V, V_{S2} = V_{D2} = 1.5V$. For in-array ERS operations, the gate voltages are always first raised before V_{D1}, V_{S1} rises to avoid NBTI on the column-selected devices.

To verify the PVRS-PRG method in the array. An experiment is done in the CTT array where every two columns' drain lines are shorted. Over-PRG is also used to compensate for relaxation. **Fig. 2. 19** shows the in-array CTT devices fine-tuned to random targets using PVRS-PRG. For the half-selected devices, including row-selected and column-selected, the error of the device before and after their twin/neighbor devices are programmed, is shown in **Fig. 2. 20**.

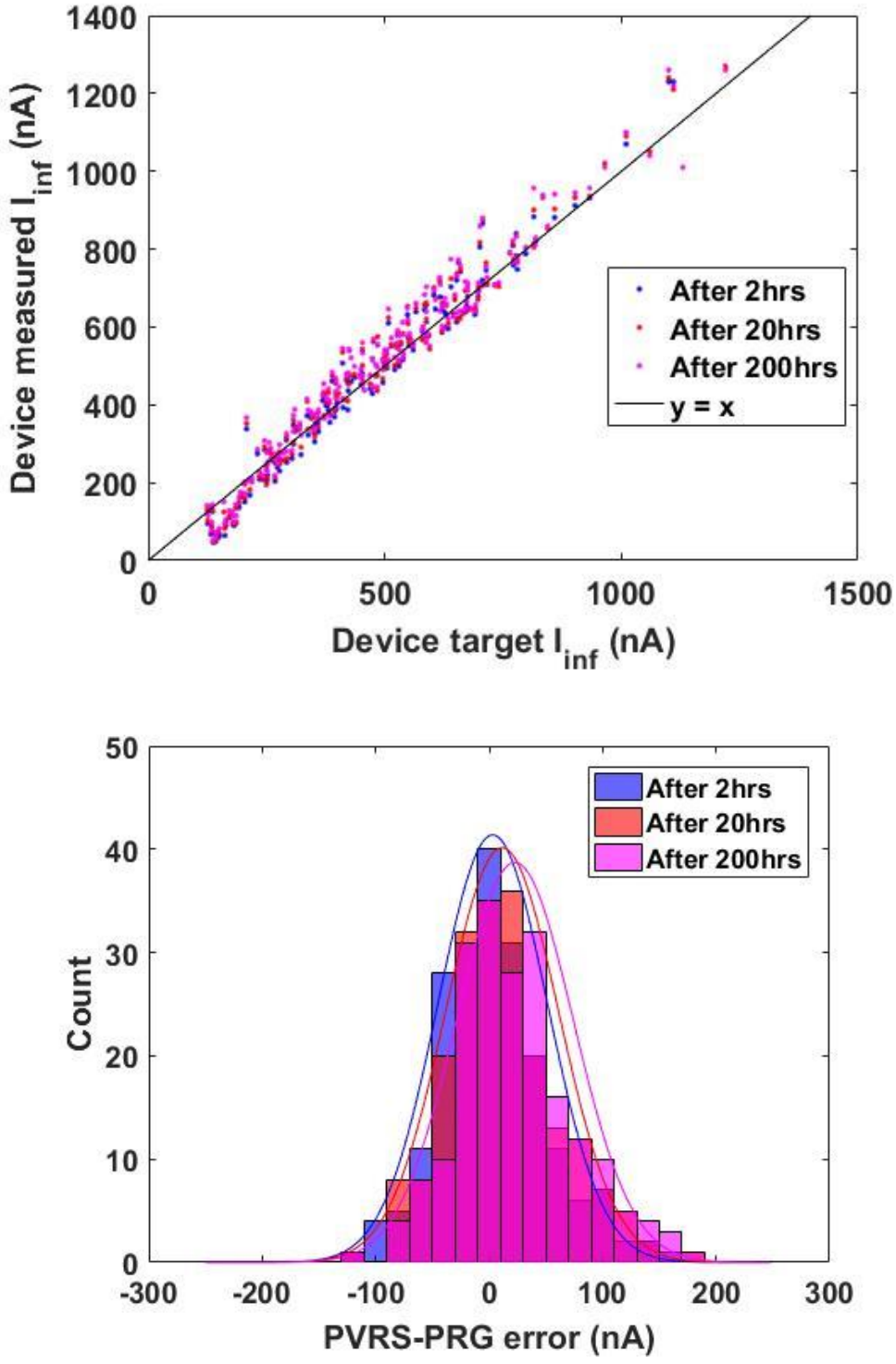


Fig. 2. 19 The target and the device I_{inf} measured after 2, 20 and 200 hours relaxation at room temperature for the selected devices (top). The error can be modeled as a Gaussian distribution centered around 0nA for the selected devices (bottom).

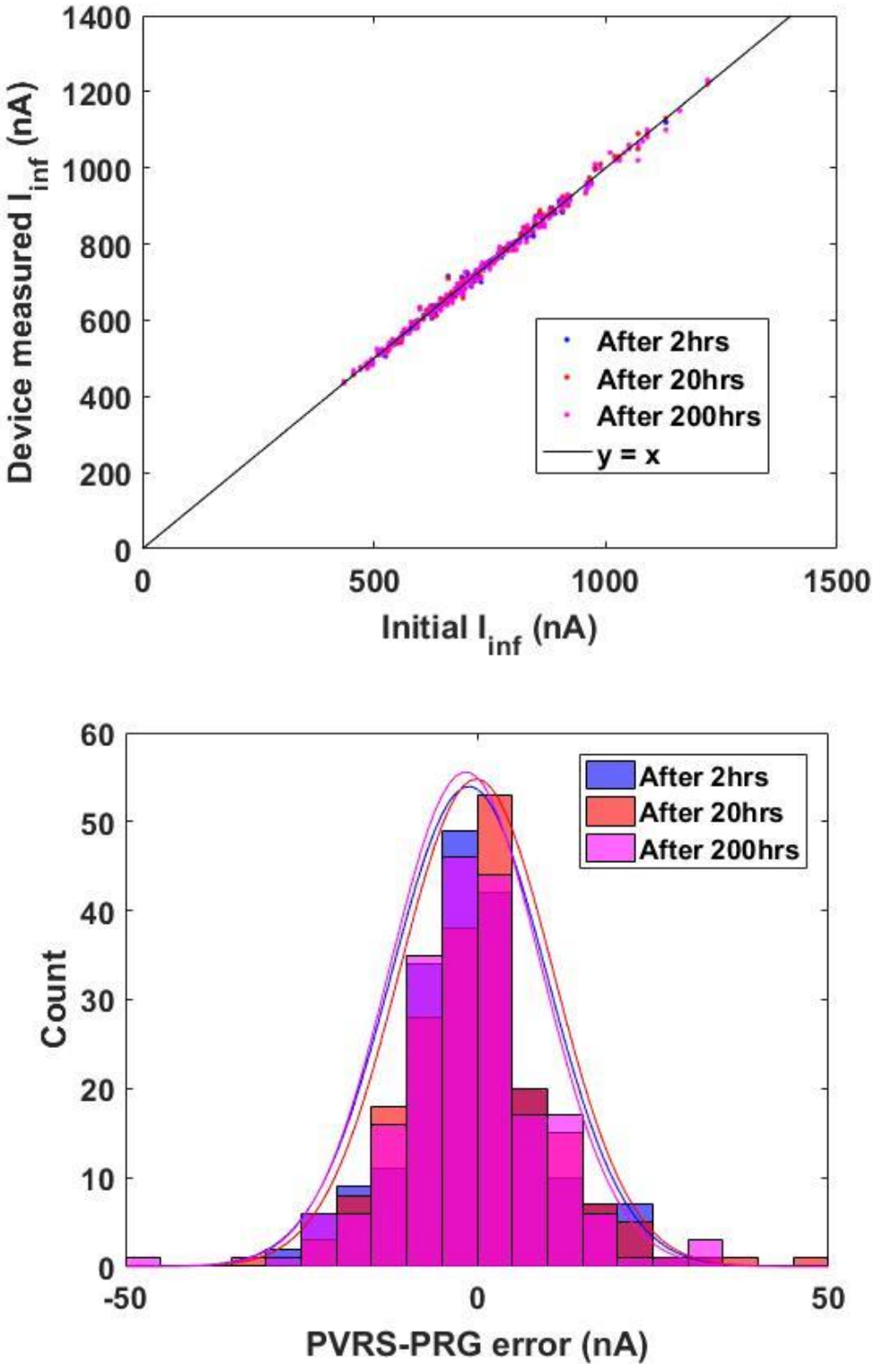


Fig. 2. 20 The target and the device I_{inf} measured after 2, 20 and 200 hours relaxation at room temperature for the half-selected devices (top). The error can be modeled as a Gaussian distribution centered around 0nA for the selected devices (bottom).

2.3.4 CTT Device and Array Reset

A critical advantage of the CTTs' charge-trapping through bulk-oxide traps is to enable room temperature ERS of the device to reset. However, since the room-temperature ERS cannot de-trap or anneal interfacial traps [Ma 19], the memory window of the CTT will be reduced for CTTs subject to reset. **Fig. 2. 21** shows the evolution of I_{inf} during four repeated cycles using the PVRS-PRG (V_G from 1.5V to 2.7V) and PVRS-ERS (V_G from -1.5V to -2.7V) methods (**Fig. 2. 11**) at different PRG drain bias points (0.8V, 1V, 1.2V, 1.4V). After the first PRG/ERS cycle, the usable memory reduces (assuming the conditions for PVRS-PRG/ERS do not change) depending on the PRG V_D (i.e., interfacial traps).

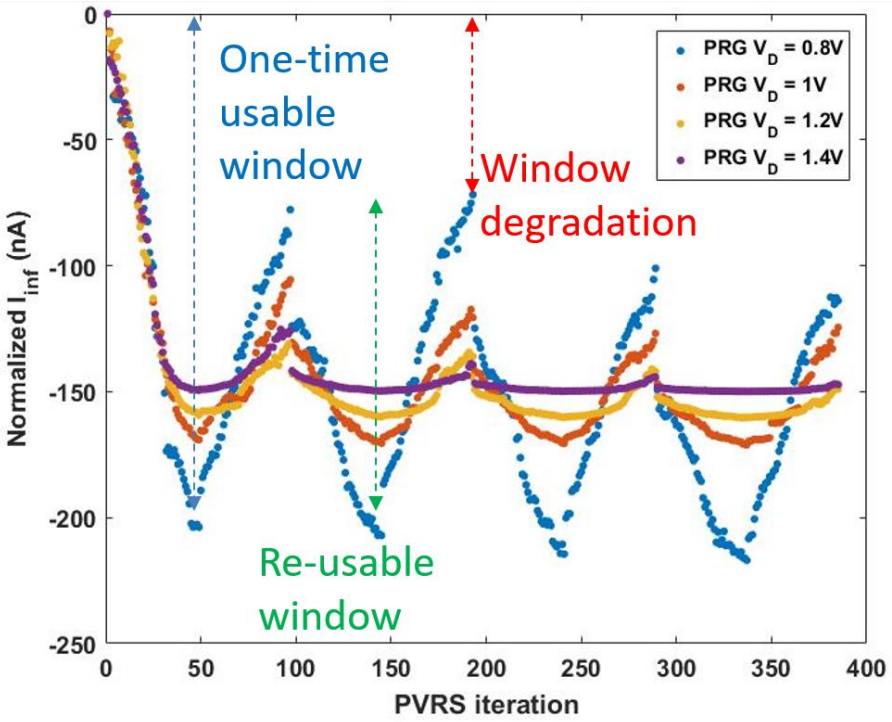


Fig. 2. 21 the evolution of I_{inf} of CTTs under 4 cycles of PVRS-PRG/ERS, each cycle consists of 48 PVRS-PRG followed by 48 PVRS-ERS. Four different V_D values used during PVRS-PRG are compared. Because the proposed ERS method is at room temperature and cannot de-trap interfacial traps, the memory window degrades. Higher V_D during PVRS-PRG traps more charge to the interfacial traps and therefore suffer more from memory window degradation. The one-time usable memory window, re-usable memory window and window degradation are roughly marked for the case of PRG V_D = 0.8V.

To remove the contribution of the interfacial traps from the memory window, an initialization cycle can be deployed. Here a PRG/ERS cycle is applied to the CTT before the actual writing to its target. The initialization will ensure that the remaining memory window is mainly from bulk-oxide traps and therefore, reusable. **Fig. 2. 22** shows the distribution of an array of CTTs' I_{inf} at the virgin state, and after two initialization cycles. The memory window of the devices in the array is roughly the minimum I_{inf} of that array since I_{inf} decreases exponentially with increasing V_{th} and can be very close to zero.

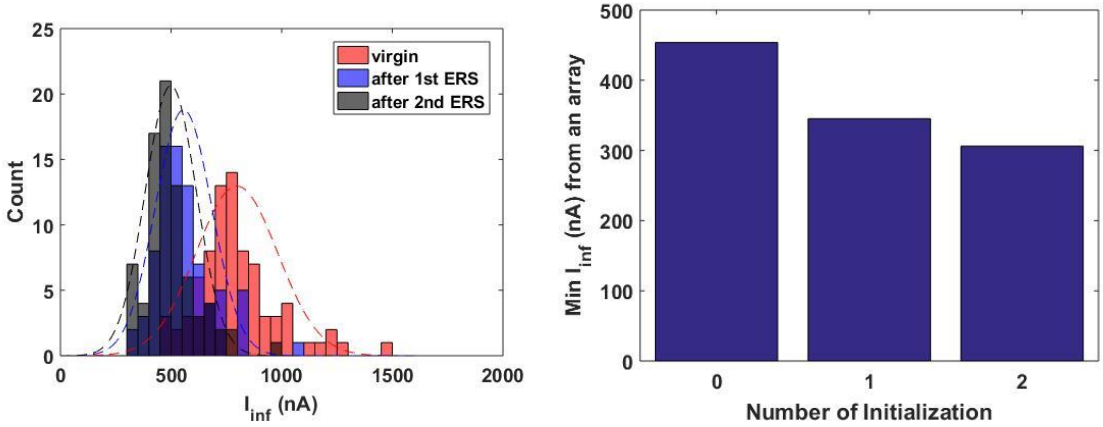


Fig. 2. 22 Left: distribution of the I_{inf} of an array of CTTs at virgin state, after the first and second initialization cycle. **Right:** the minimum value of the I_{inf} in the array, which is roughly the size of the memory window.

The CTTs can be programmed using the same PVRS-PRG method proposed in the last section. **Fig. 2. 23** shows the target and the reused device (after reset) I_{inf} measured after 1, 10, and 100 hours at room temperature. The error of the PRG result can also be modeled as a Gaussian distribution. **Table 2** shows the mean and standard deviation of the in-array programming error of CTTs.

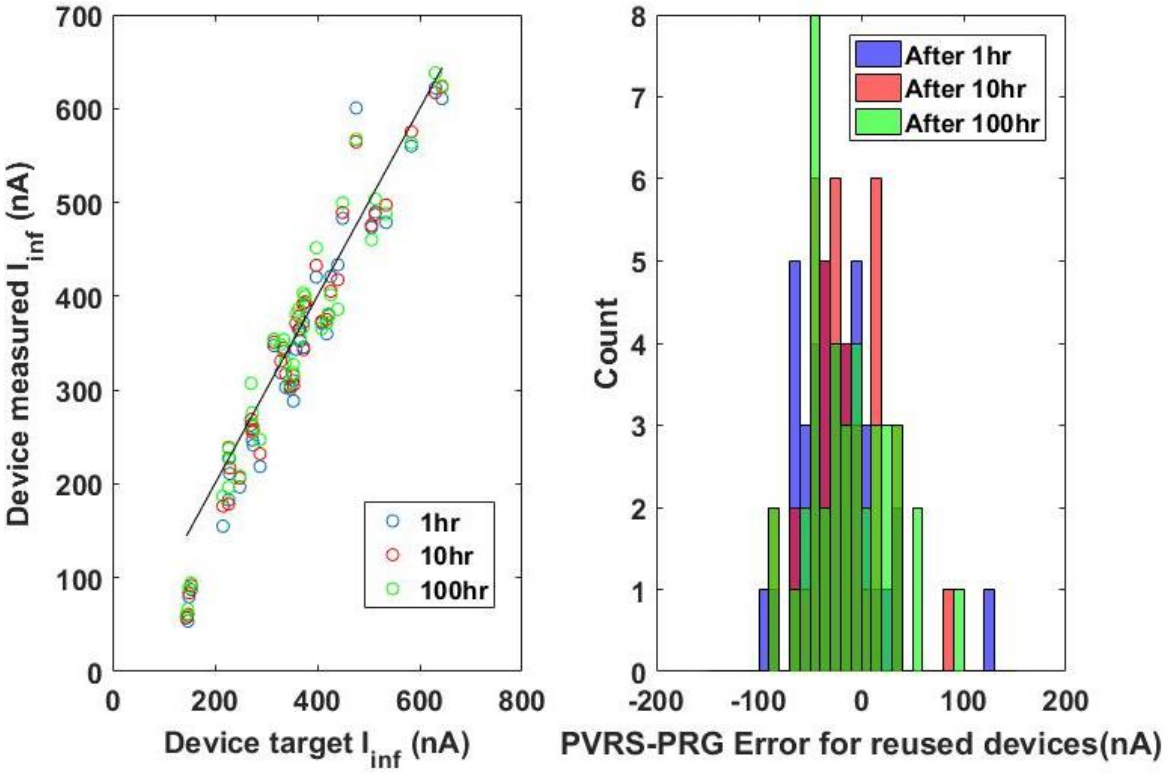


Fig. 2. 23 The target and the device I_{inf} measured after 1, 10 and 100 hours at 300K for reused devices (left). The errors in all cases can also be modeled as a Gaussian distribution for the reused devices (right).

Table 2 Summary of the statistics of the errors from CTT programming under different conditions (program target is randomly selected in the target range)

	Error mean (μ_{error})	Error standard deviation (σ_{error})	Target range	$\frac{\mu_{error}}{range}$	$\frac{\sigma_{error}}{range}$
One-time 2hr	3.15nA	48.2nA	1200nA	0.26%	4.02%
One-time 20hr	11.4nA	49.7nA	1200nA	0.95%	4.14%
One-time 200hr	22.7nA	51.5nA	1200nA	1.89%	4.29%
Reuse 1hr	-24.4nA	39.0nA	500nA	-4.88%	7.80%
Reuse 10hr	-18.2nA	35.4nA	500nA	-3.64%	7.08%
Reuse 100hr	-12.2nA	38.6nA	500nA	-2.44%	7.72%

2.4 Summary

In this chapter, we have demonstrated that the charge-trapping phenomenon in the high-k dielectric of standard CMOS logic devices can be leveraged for analog data storage with good data retention at room temperature. The optimal application scenario of CTTs as analog memory is for one-time use in low-power systems. For multi-time programmability, the memory window will be reduced while the absolute programming accuracy is similar, and therefore reducing the resolution of the memory. The reduction of memory window can be alleviated by providing a heated environment during ERS to de-trap the interfacial traps but at the cost of the extra supporting circuitries.

Comparing with the other emerging analog devices CTTs have several advantages

1. CTT is CMOS-compatible at advanced nodes (demonstrated up to 7nm [Khan 19]). Unlike most of the other analog devices, no extra materials or processes are introduced to fabricate the CTT. As a result, variations and reliability of CTTs are relatively well studied and controlled, making it a competitive candidate as analog memory, which can be easily integrated with other existing VLSI designs.
2. In contrast to some two-terminal analog devices such as phase-change memory (PCM) resistive RAM (RRAM) and memristors, CTT is, by nature, a three-terminal device with a non-linear selector (i.e., gate) to avoid leakage as the array scales up in size. For two-terminal devices organized in a crossbar architecture, the selector needs to be added (**Fig. 2. 24**) [Zhou 14].
3. The mature CMOS technology also ensures a very high yield compared with the emerging analog devices whose yield can still be challenging. In addition, the proposed methods to use CTT, including I_{inf} reading, PRG and ERS, are very repeatable and safe on top of the

well-controlled device variation. In contrast, “dead” cells can become a problem for RRAM due to the device variation and the repeating operations [Zheng 18].

4. The CTTs can be arranged densely in an array because the footprint of the CTTs is very small due to the advanced node. The charge-trapping mechanism does not have a strict requirement on the channel length and width of the CTT device.
5. Because the CTTs operate in the subthreshold region, and the relatively small size, the on-resistance of the CTTs is high compared with other analog devices, making it energy efficient during read operations and suitable for applications such as neural network inference.

To benchmark the CTTs with other analog memories for neural network inference, CTT device parameters are extracted from the experiment for hardware simulation using the NeuroSim simulator [Chen 18]. The CTTs are simulated with peripheral circuits for an 8-layer VGG-like convolutional neural network with 8-bit weight and 8-bit activation. The benchmark result is shown in**Fig. 2. 25.** It shows that the analog inference engine based on CTTs is faster and more energy-efficient than the compared ones based on phase-change memory [Burr 15],

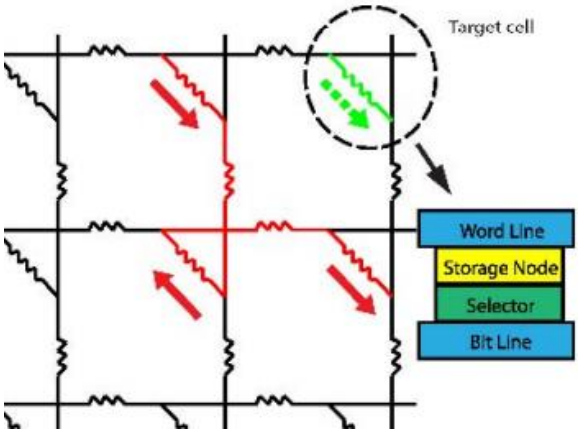


Fig. 2. 24 Cross-bar array structure of the two-terminal analog memory (e.g., RRAM). Green dashed arrow shows the current through the target cell. Red arrow shows the leakage current. The half-selected devices requires a non-linear selector (right) to reduce the leakage (Adapted from [Zhou 14]).

resistive RAM (RRAM) [Jain 19] and spin-transfer torque magnetoresistive RAM (STT-MRAM) [Kim 11].

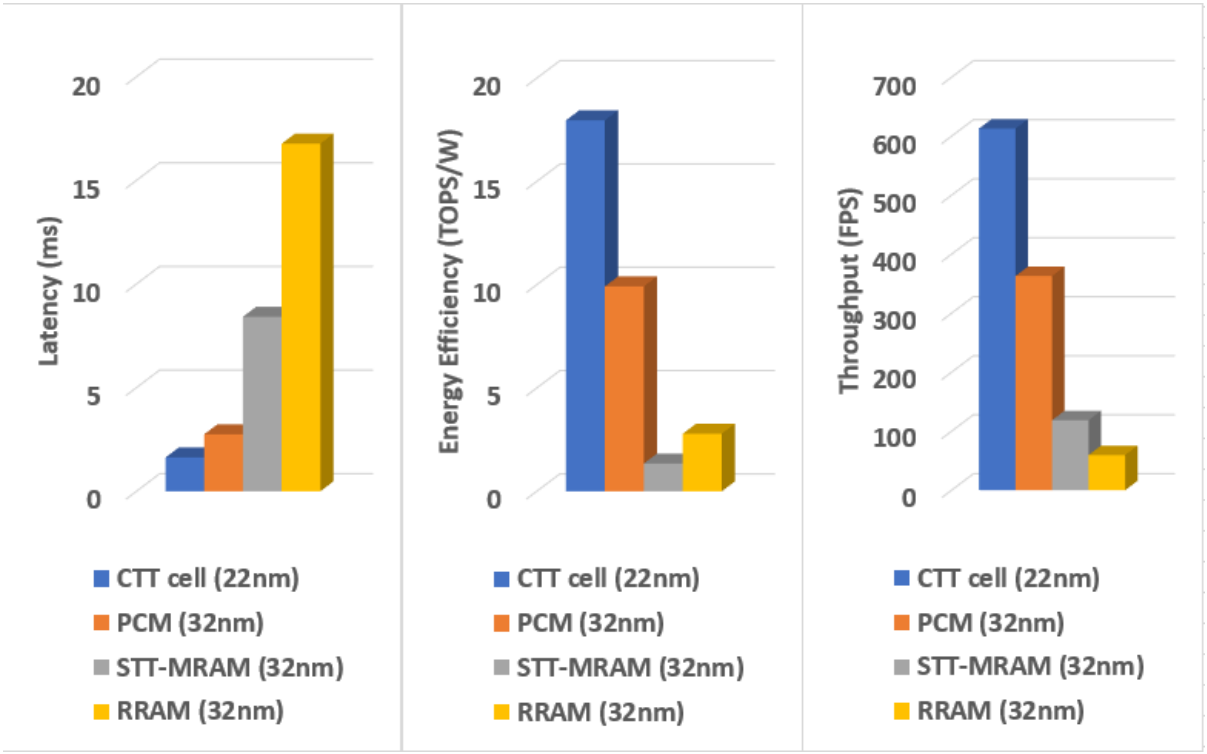


Fig. 2. 25 Benchmarking CTT for neural networks against other analog devices using NeuroSim simulator [Courtesy of Yandong Luo and Prof. Shimeng Yu].

3. CMOS-Compatible Analog Neuromorphic Computing Engine

3.1 Introduction

Neuromorphic electronic systems include a wide range of applications at different levels. Silicon-based electronic systems can be used to simulate large-scale neural models in real-time [Silver 07, Schemmel 10, Indiveri 11, Benjamin 14], or to implement a specific function based on biological structure such as vision [Koch 96] and echolocation [Wen 18]. For machine learning, various neuromorphic systems are applied for inference of supervised learning, including spiking neural networks [Merolla 14], multi-layer perceptron [Guo 17], and convolutional neural networks [Shafiee 16]. Training is also demonstrated in neuromorphic hardware for supervised learning [Davies 18] and unsupervised learning [Cai 19]. The CTT devices discussed in the last chapter have also been demonstrated for unsupervised machine learning applications [Gu 18(2)].

In this chapter, we discuss the use of CTTs in a CMOS-compatible analog neuromorphic computing, with a focus on the inference engines for multi-layer perceptron and convolutional neural networks. Natural laws, such as the charge conservation, is used to perform analog charge-based computing for the vector-matrix multiplication (VMM), the major computing workload in the networks. To maximize the energy-efficiency, the computation is performed in the memory (i.e., CTT array) to eliminate the requirement for off-chip memory access. In addition, minimum data conversion from and to the digital domain is used since it does not contribute to real computations.

3.2 Compute VMM in CTT array

Analog memory devices such as CTTs can be arranged into an array to build an analog vector-matrix multiplication (VMM) engine for applications such as neural networks [Gu 18]. A basic VMM is defined as $\boldsymbol{y} = \boldsymbol{xW}$ where $\boldsymbol{x}$ is an input vector of 1 * M, $\boldsymbol{W}$ is a weight matrix of M * N. A schematic of the VMM engine based on the CTTs is shown in **Fig. 3. 1**. In this architecture, CTTs are arranged in an M * N matrix with M rows and N columns. Each CTT in the array represents an element in the matrix using its conductance G_{mn} at a given bias condition (i.e.,Iinf/VDS from the last chapter, where VDS = 50mV and VGS = 200mV). The gate terminals of each row of the array are connected by a word-line (WL). The source/drain terminals of each column of the array are connected by a source-line (SL) and bit-line (BL), respectively. The gates of the CTTs are used as the terminal to receive the input "on" of VG = 200mV. At the meantime, the BLs are connected to VD = 50mV, and SLs are connected to VS = 0V, so that the CTT device which receives an "on" input of VG = 200mV will draw the current of the pre-programmed Iinf value from BL to SL. When the input is "off", the WL can be reduced below ground level (e.g., VG = -300mV) to reduce the leakage due to VDS > 0.

Following this convention, each WL represents an element of the input vector. As the CTTs operate in the subthreshold region, the relationship between the V_{GS} and the I_{DS} is exponential, making it difficult to use the magnitude of V_G at each WL for linear calculation. Therefore, the input values are encoded as pulse-width modulated (PWM) signals $V_m(t) = V_0[u(t) - u(t - x_m\Delta_t)]$ at the WLs (e.g., the **m**th row), where $u(t)$ is the step function, V_0 is the "on" voltage (200mV), x_m is the **m**th entry of the input vector and Δ_t is the unit pulse width when $x_m = 1$. For

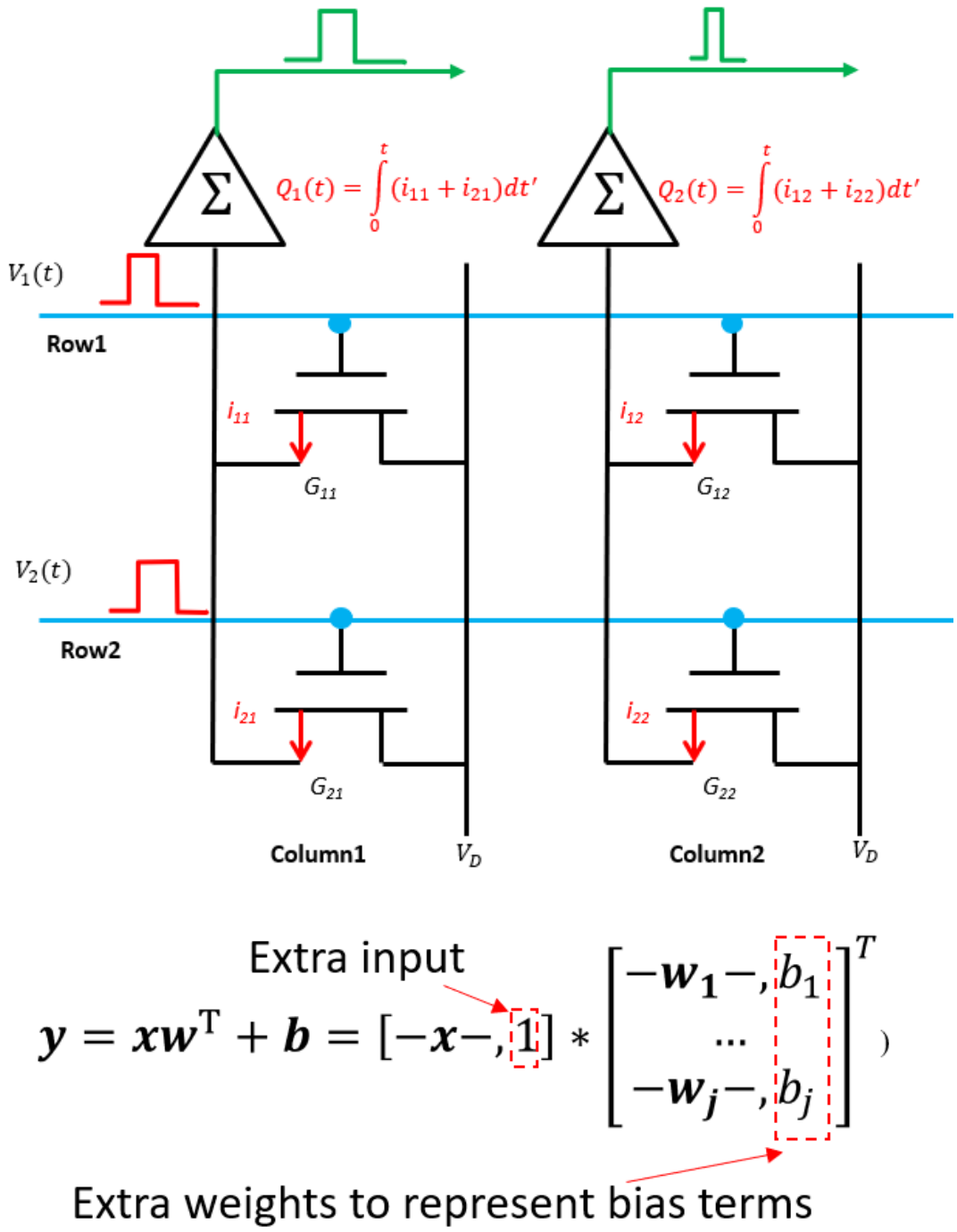


Fig. 3. 1 Top: array of CTTs can be used for vector-matrix multiplication (VMM). Current (e.g., I_{inf}) of CTT devices can be programmed to represent the values in the matrix. Input vector can be represented by pulse-width modulated (PWM) signal sent to the word-line (WL) of the array, which connects the gates of the CTTs in a row. Multiplication is thus computed by integrating I_{inf} for the time period specified by the PWM signal. Summation is computed by collecting the drain current of all the devices in a bit-line which connects the sources and drains of the CTTs in a column, respectively. **Bottom:** the VMM engine can be directly used for VMM with a bias term by adding an extra input and an extra row of devices in the array.

the time when the input is "high" (200mV), the **m**th device in the **n**th column will draw the amount of charge $Q_{mn} = I_{inf,mn} x_m \Delta_t$ from BL to SL which is also the multiplication result between the input value x_m and the stored matrix value $I_{inf,mn}$. By charge conservation, the total charge moved to the SL of each column is the dot-product of the input and the CTTs at that column:

$$Q_n = \left(\sum_m I_{inf,mn} x_m \right) \Delta_t + Q_{leakage}$$

To obtain accurate dot-product, the charge collection time T_{int} must be longer than the longest PWM input signal presented in all WLs, and some charge from leakage $Q_{leakage}$ will be included when some of the WLs are "off" during the charge collection by

$$Q_{leakage}(T_{int}) = \sum_m I_{leakage,mn}(T_{int} - x_m \Delta_t)$$

As each column generates the output of a dot-product, the entire array will generate an output vector that represents the result of the vector-matrix multiplication. In addition, bias terms $\boldsymbol{b}$ in $\boldsymbol{y} = \boldsymbol{xW} + \boldsymbol{b}$ can also be calculated in the VMM engine by programming an extra row of the CTTs to the bias values and adding an extra constant in the input (**Fig. 3. 1)**. A similar structure can also be built from any two-terminal analog devices, as previously reported [Shafiee 16, Cai 19]. One disadvantage of the PWM scheme is that it cannot directly support negative values, but negative inputs to the VMM engines are not required in many neural network applications.

While negative inputs are not required in the neural networks, negative values in the matrices are still required. To represent negative entries in the matrices, the CTT memory cells use a twin-CTT structure to enable bipolar conductance, where the current from the true device (I_{inf}^{+}) and complement device (I_{inf}^{-}) of the cell is processed as a differential current

$$I_{inf} = I_{inf}^{+} - I_{inf}^{-}$$

To reduce the footprint of the cell, either the sources or the drains of both devices can be shorted [Gu 18]. **Fig. 3. 2** shows the closed-loop read-write-read process to adjust the twin-CTT cell values using *only* PRG operation. Virgin twin-CTT cell weights are first measured. Then if the target weight is below the virgin weight, the "true" device is chosen to be programmed (I_{inf}^{+} is reduced) to the desired target with over-PRG compensation. Otherwise, the "complement" device is chosen (I_{inf}^{-} is reduced). The programming in the array uses the optimized PRG conditions from the last chapter. The twin-CTT cell structure does not affect the program accuracy of each of the devices. Therefore, the programming accuracy of the twin-CTT cell depends on whether both devices need to be programmed. If both devices are programmed, the standard deviation of the cell's error becomes $\sqrt{2}\sigma_0$, where σ_0 is the standard deviation of the error of single CTT. In the flow illustrated by **Fig. 3. 2**, only one device is programmed for one-time use, and therefore the standard deviation of the cell's error is σ_0 for one-time use. **Fig. 3. 3** shows the accurate programming of the twin-CTT cells, applying the PVRS-PRG and the over-PRG compensation models proposed in the last chapter. The mean (μ) and the standard deviation (σ) of the error for different conditions are summarized in **Table 3**.

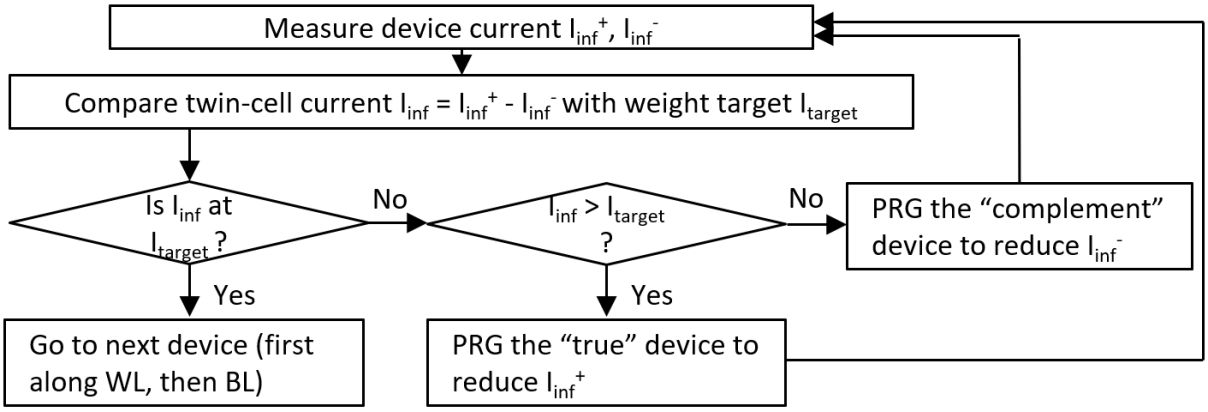


Fig. 3. 2 Close-loop read-write-verify steps for array fine-tuning. Based on the difference between the initial weight of the twin-CTT cell and the target weight value, either the "true" (+) device or the "complement" (-) device is chosen for PRG. The device is read after each programming step to verify its analog state.

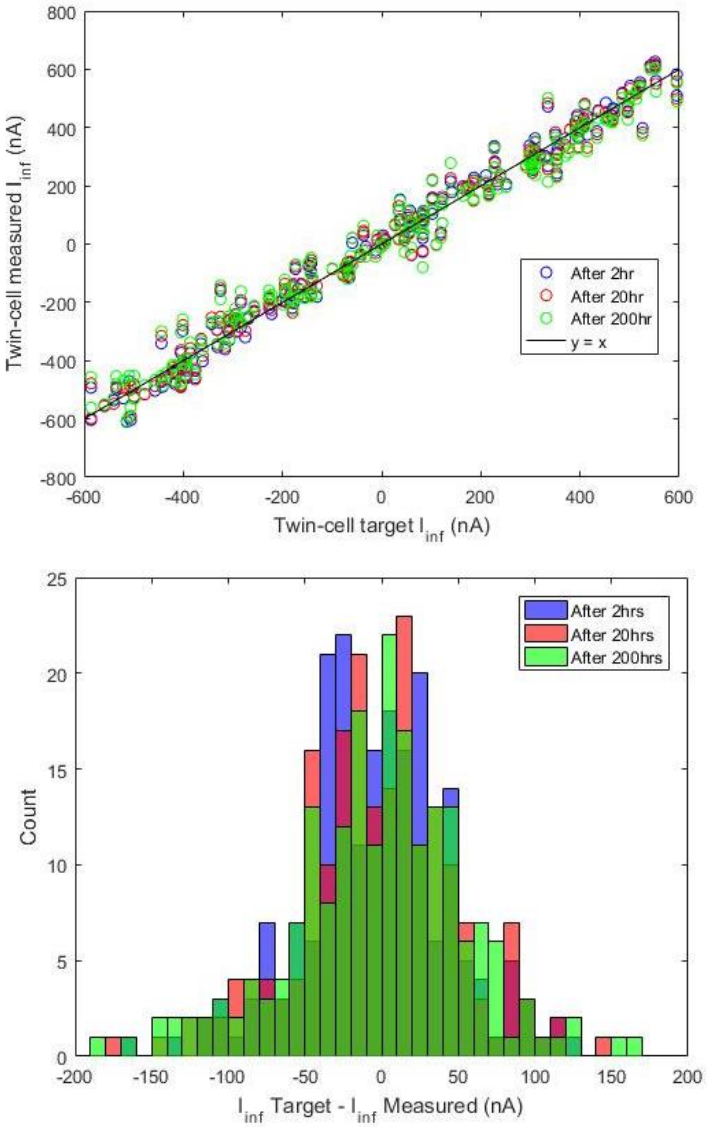


Fig. 3. 3 Top: The target (randomly generated in the [-600nA,600nA] window) and the twin-CTT cell I_{inf} measured after 2, 20 and 200 hours at 300K. **Bottom:** the errors in all cases can also be modeled as a Gaussian distribution centered around 0nA for the Twin-CTT cell.

Table 3 Twin-CTT cell programming error, mean (μ) and standard deviation (σ) statistics

	μ_{error}	σ_{error}	Target range	μ_{error}/range	σ_{error}/range
25C 2hr	-3.29nA	48.5nA	1200nA	-0.27%	4.04%
25C 20hr	-3.61nA	51.1nA	1200nA	-0.30%	4.26%
25C 200hr	-3.07nA	56.8nA	1200nA	-0.26%	4.73%

The output vector from the VMM, stored as a vector of analog charge, can be discharged in the form of PWM signals by a constant circuit source. The pulse width from discharging will

be linearly proportional to the stored value and therefore, can be directly used as the input of another array with a similar structure. This makes it possible to cascade the CTT arrays for neural networks with multiple layers. However, the storage of an analog charge value, normally on a leaky capacitor, will have very weak data retention. Refreshing this analog value can demand costly hardware when a high-resolution sensing circuit is required. Therefore, the computed results, in the form of analog values, should either be digitized for good data retention or transmitted quickly, by the next array or to the output channels.

The fabricated CTT array is also verified for the vector-matrix multiplication operations. Random DC input vectors (200mV for "on", -300mV for "off") are fed to the WLs, and the current at each BL is measured. This captures different moments of the system when the inputs are random PWM signals, and helps to circumvent the distortion of AC signals by the large capacitance of the off-chip measurement equipment. The experiment shows that the mean and standard deviation of the VMM error $(\Sigma I)_{\text{measured}} - (\Sigma I)_{\text{target}}$ is less than 0.5% and 4% of the possible range, respectively, after up to 200 hours of room temperature relaxation (**Fig. 3. 4**). The additional error from PWM inputs is the error of transition between the "on" and "off" of the WL, whose effect

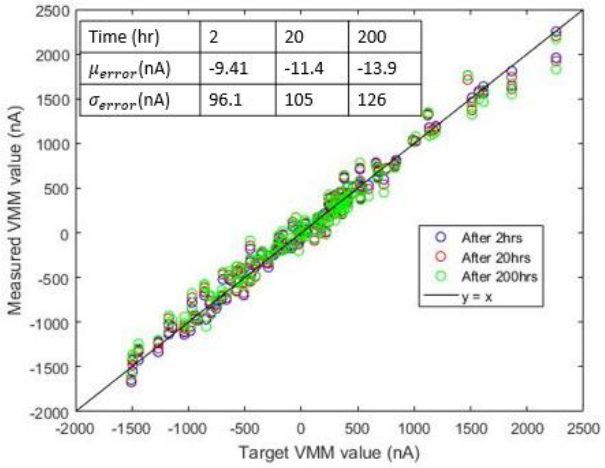


Fig. 3. 4 Vector-matrix multiplication error in the CTT array: 32 random input binary vectors (10x1) are multiplied with the 5 fine-tuned twin-CTT cell arrays (4x10 each) with V_{on} = 200mV and V_{off} = -300mV.

depends on the drive strength of the PWM signal generator at the WL and the capacitive load of the WLs. Its impact on a neural network application is evaluated in the next chapter.

3.3 CTT-Based Analog Inference Engine for Perceptron

Based on the VMM engine introduced in the last section, the analog inference engine can be readily built for a perceptron A perceptron is a type of neural network featuring full connectivity between layers of neurons (**Fig. 3. 5**). Multi-layer perceptrons (MLP) are widely used by the industry nowadays for machine learning tasks [Jouppi 17]. Due to the CMOS-compatibility of the CTTs, the peripheral circuits for the analog inference engine can also be designed in the same CMOS technology that fabricated the CTTs.

A block diagram for the MLP architecture with some key peripheral circuits is shown in **Fig. 3. 6**. The network is trained at another training engine and then programmed into the CTT. During inference operation, CTTs operate at subthreshold bias points. For a complete VMM engine, the differential current/charge integrator at the end of the BLs can be implemented by op-amp based CMOS circuits [Razavi 02]. If the input is conventional digital data, it needs to be converted to PWM inputs to the first array. Similarly, the output of the last array can be converted to digital signals, but this is not required if the output is fed into another array.

Other than the VMM operations, the non-linear activation function is also required by the perceptron. One of the most used activation functions is the rectifying linear unit (ReLU), which suppresses negative inputs to zero and passes the non-zero inputs (i.e.,$ReLU(x) = \max(0, x)$). The ReLU function can be implemented by a comparator circuit (**Fig. 3. 7**) [Gu 18] which produces a PWM signal as the ReLU's output while VMM result was discharged by the current source. Other activation functions can also be implemented using CMOS circuits [Huijsing 13]. The output of the comparator can be fed to the WL of another CTT array to build a multi-layer perceptron (MLP).

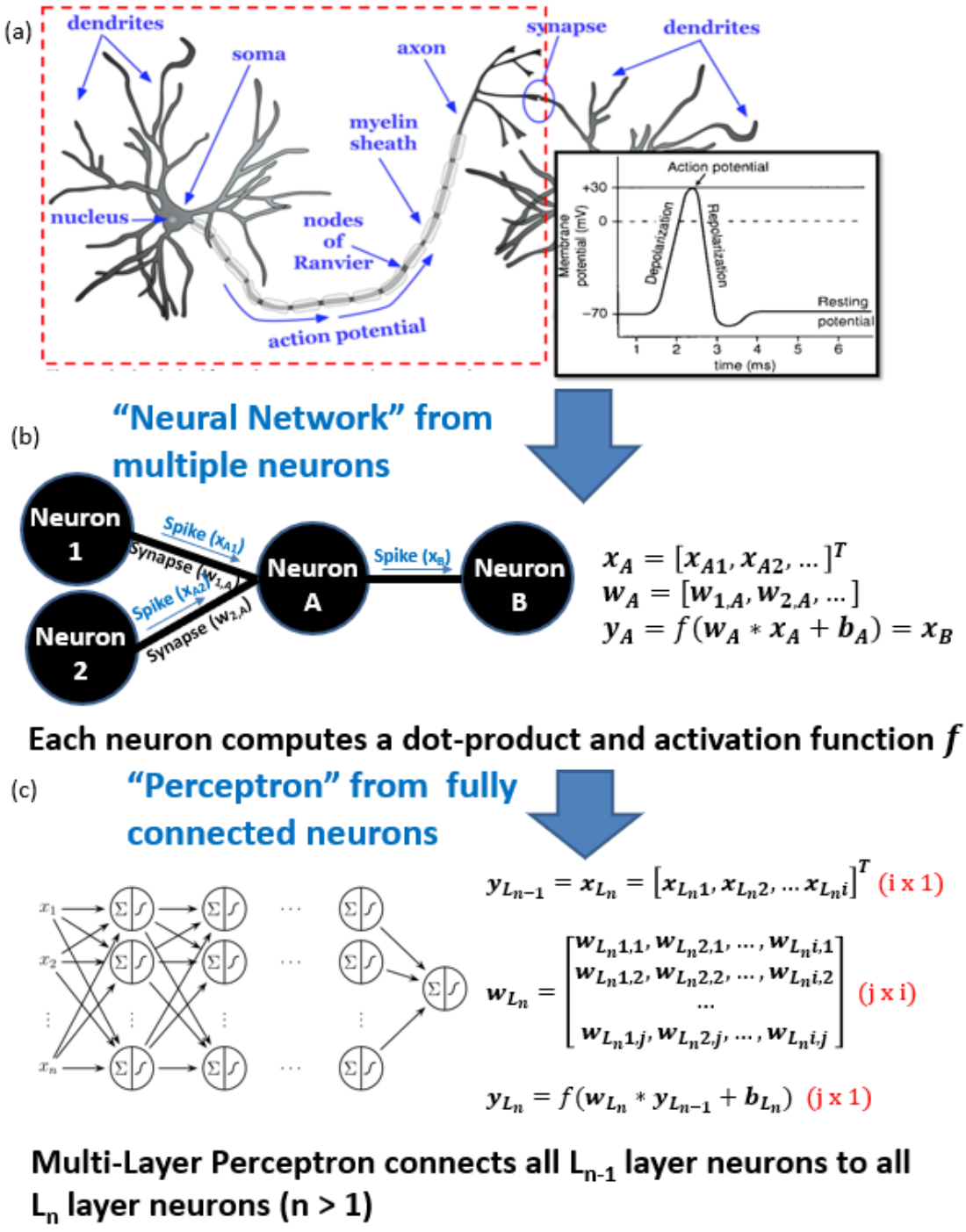


Fig. 3. 5 Mathematical formation of the multi-layer perceptron (MLP) from the biological neurons. (a) Neurons connect to each other through synapses. (b) The receiving neuron computes a dot-product between the input (output of previous neuron) and the "synaptic weights" assigned the associated synapses, followed by a non-linear activation function. Neurons can be grouped into layers. (c) Single-layer perceptron and MLP have 2 or more layers of neurons, respectively. They feature fully connectivity that each neuron is connected to all neurons in the last layer.

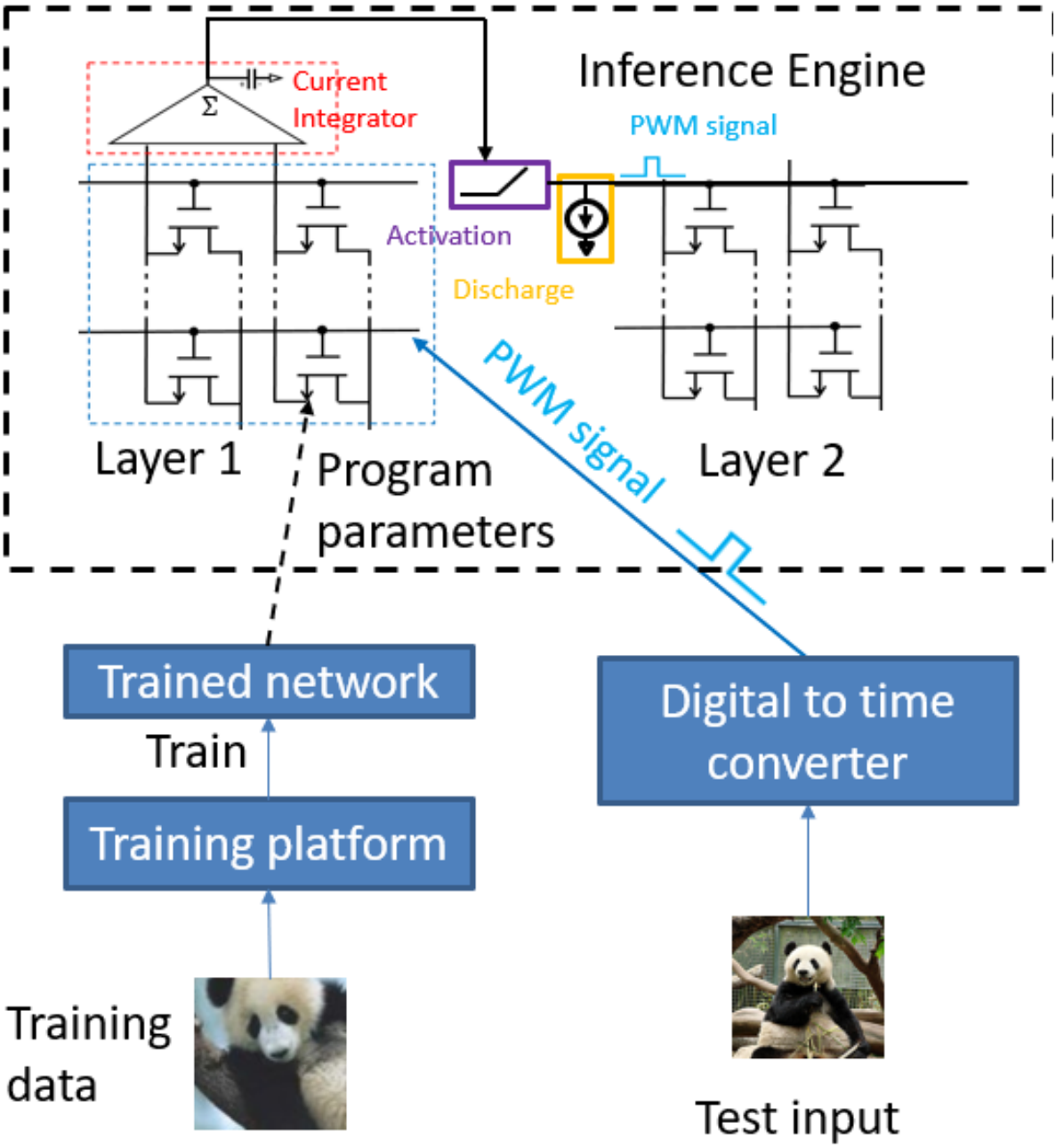


Fig. 3. 6 The system block diagram of a CMOS analog neural network inference engine. It shows the interface between the system and the digital input, and the interface between two network layers. Digital-to-time converters (DTCs) are used to convert the digital input to a PWM signal for the array for computation. The output of the current integrator at the end of the CTT array is further processed by a comparator-based activation circuit and discharged by a constant current source to generate another PWM signal. The pulse-width of the output PWM signal is proportional to multiplication result, and therefore can be applied as the input to the next layer. The neural network is trained using training data at other training platforms, before deployed into the inference engine.

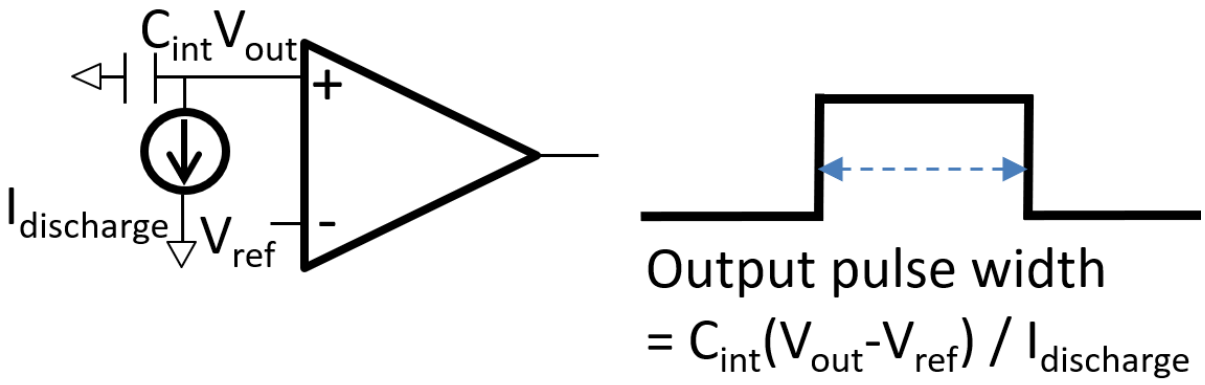


Fig. 3. 7 Comparator based circuit to implement rectifying linear unit (ReLU) activation function. The comparator compares the reference voltage V_{ref} and the voltage stored at the array's output capacitor (output of a dot-product) to produce a voltage output (otherwise 0V) when the reference voltage is lower. As a result, the output is rectified by V_{ref} and the output is also a PWM signal proportional to the value of $C_{int}(V_{out}-V_{ref})/I_{discharge}$ only if $V_{out} > V_{ref}$.

The advantage of the CTT-based analog inference engine for MLP is the low latency and high throughput of the system due to the in-memory computing architecture. For all values used and generated in the computations, only the input of the MLP and the output of the MLP need to be communicated to other blocks of the system (e.g., data I/O). Therefore, the time to wait to fetch data from memory, as required for most digital systems, is significantly reduced. For CTTs, the single-device response time can be in the order of 10ps [Carter 16], but the read-speed at the array-level is limited by the capacitive load of the device and the passive elements associated with the WLs, which can be in the order of 10^{-9} second/operation depending on the physical design.

The latency of each layer can be divided into two parts, (1) duration of input: when the array is taking the PWM inputs and simultaneously integrating the dot-product as a charge, and (2) duration of output: when the charge is discharged as either another layer's input or the final result of the MLP. Since the PWM inputs are taken directly from the last layer's output except for the first layer, the system's pipeline is naturally balanced when the duration of the output of each layer is the same. **Fig. 3. 8** shows the pipeline of the proposed MLP inference engine during inference.

The output duration must be at least as long as the longest output of the layer, which is determined by the ratio between the maximum charge held at the output capacitor and the discharging current.

However, write-speed of the CTTs is less than 10^{-2} second/operation due to the closed-loop read-write-read process, based on our experiment that the average number of iterations for random targets is about 20 (each is about 0.5ms). Therefore, the required number of CTT memory cells in the system is the number of weights the network has (assume one cell per weight), and all weights are required to be programmed to the on-chip CTT memory without leveraging the memory hierarchy as done in typical digital systems.

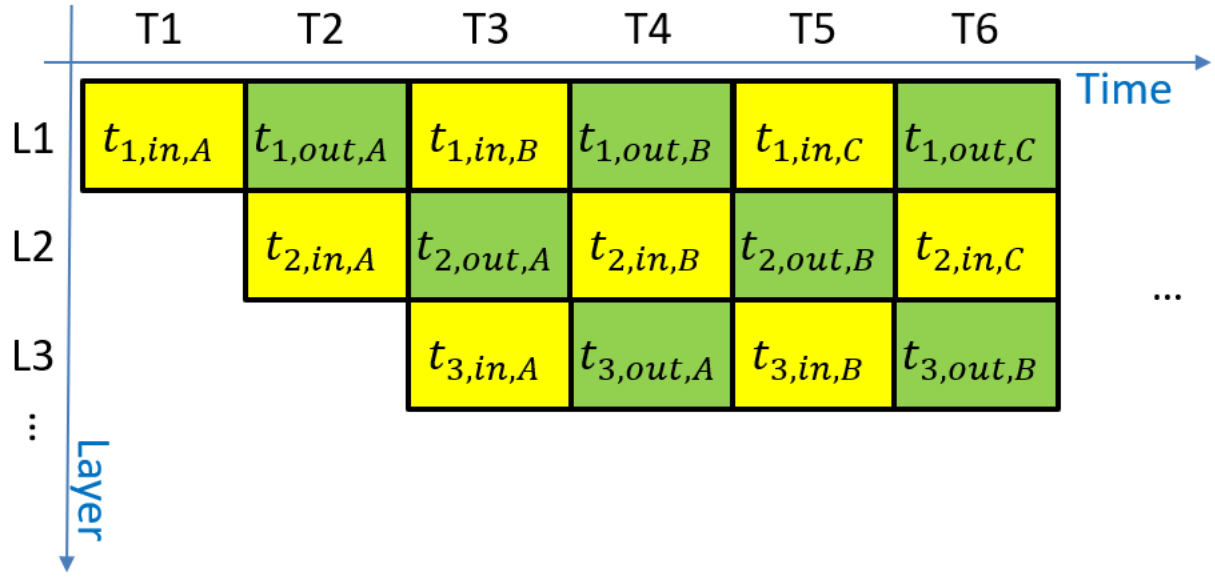


Fig. 3. 8 The pipelining of the MLP inference engine during inference. For each layer (L1, L2, L3), the time used for computation is divided into input duration (t_{in}) when it is taking the PWM inputs and integrating the current, and output duration (t_{out}) when the integrated charge is being discharged. $t_{1,in,A}$ represents the input duration of L1 for input A. Since the output duration of layer L-1 is always the same as the input duration of the layer L, when output duration is the same for all layers, the system pipeline can be balanced well.

3.4 CTT-Based Analog Inference Engine for Convolutional Neural Networks

Another frequently used type of neural network is the convolutional neural network (CNN) [LeCun 89]. The CNNs usually have two types of layers (the ensemble of the synaptic weights and the interaction between layers): the convolutional (CONV) layer and the fully-connected (FC) layer. The FC layers are the same as the MLP layers and often located at the end of the CNNs after all the CONV layers. The CONV layers are very similar to the FC layers with two modifications. First, in the CONV layer, each neuron takes *a part of* the values from the previous layer as its input, instead of *all* values as in FC layers. Second, the neurons' synaptic weights are *repeatedly* used across the input by convolution. **Fig. 3. 9** (adapted from [LeCun 15]) shows how the CONV layers,

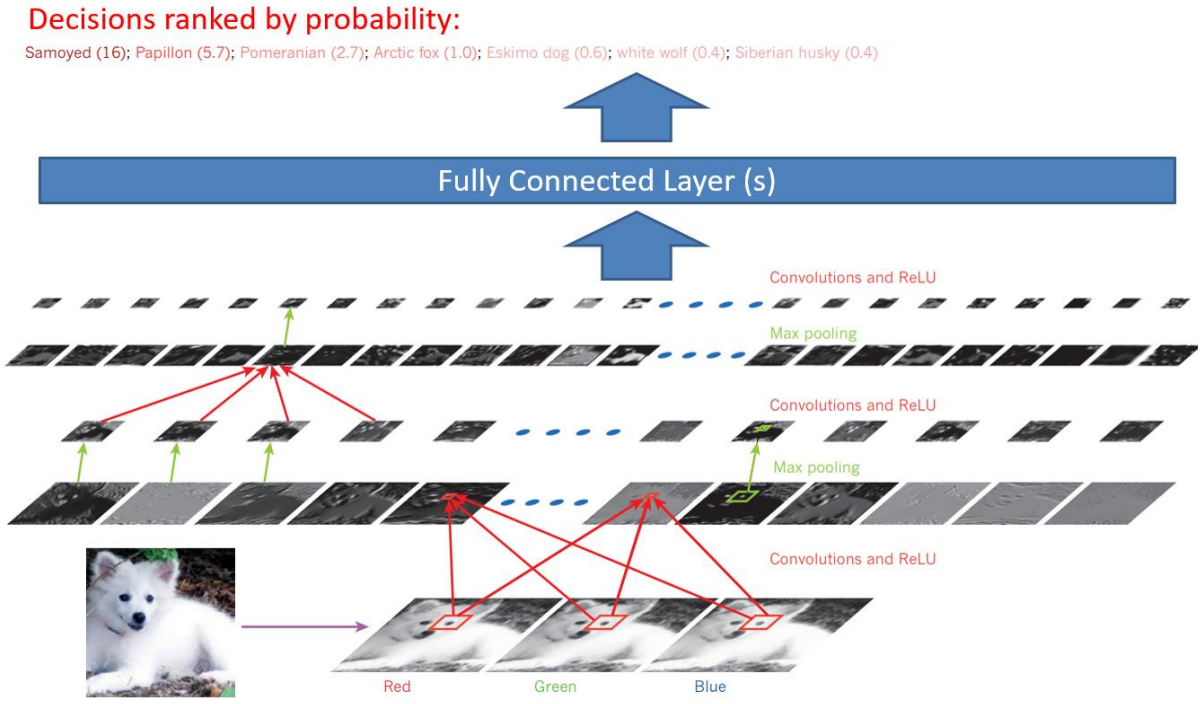


Fig. 3. 9 In the CNN, CONV layers compute a convolution between the weights and the input whose output is processed by activation functions such as the ReLU. Pooling is another feature including maximum pooling and average pooling. Fully-connected (FC) layers are normally at the end of the CNN, after all the CONV layers, to calculate for the final decision. FC layers operate the same way as MLP layers.

with some auxiliary functions, are typically organized in a CNN for applications (e.g., image classification).

In addition to the CONV layers, pooling layers are also added to the CNN. Pooling is more common in the CNNs than in the MLPs with two major types: average pooling and maximum pooling. Both can be implemented with minor modifications to the MLP hardware. Average pooling calculates the average of the inputs and can be treated as a dot-product where the weight vector has the value of $\frac{1}{n}$, where n is the length of the input vector. Maximum pooling calculates the maximum value of the inputs, which can be implemented by a logic OR gate. If the outputs of a layer are discharged at the same time, then the output of the OR gate is the longest pulse (i.e., the maximum input).

The adaption of the CONV layers makes it possible to train CNNs that are very deep and perform well in advanced machine learning challenges such as the ImageNet challenge [Deng 09]. CNNs become more powerful by scaling up in the number of layers, weights, and computations. **Fig. 3. 10** shows the number of weights and calculations required by different CNNs with respect to the network accuracy for the ImageNet challenge (adapted from [Canziani 16]).

As a result, one significant difference between the MLP and CNN is the data flow. In the MLP architecture discussed in the last section, each layer in the MLP will produce all values required for the next layer simultaneously in one computing cycle of a layer, including the time to discharge for the PWM outputs. For the CONV layers in the CNNs, the convolution is computed using the same weight matrices multiple times, which is not compatible with the MLP architecture. Since CONV layers and FC layers do not have a fundamental difference in the computations (both

are mainly vector-matrix multiplications), there are ways to convert CONV layers to FC layers and vice versa for the trade-off between memory capacity and memory bandwidth [Zhang 18].

For the proposed CTT analog memory, the memory bandwidth for reading is high, but the memory bandwidth for writing is low due to the low device programming speed (see Section 2.2). As a result, the CTT-based analog computing for vector-matrix multiplication requires the weights to be pre-programmed. To address this issue, two options for CNN computation using the CTT-base architecture are discussed: (1) store digitized outputs for convolution, and (2) unroll CONV layers to equivalent MLP layer structures to use the proposed MLP infrastructure.

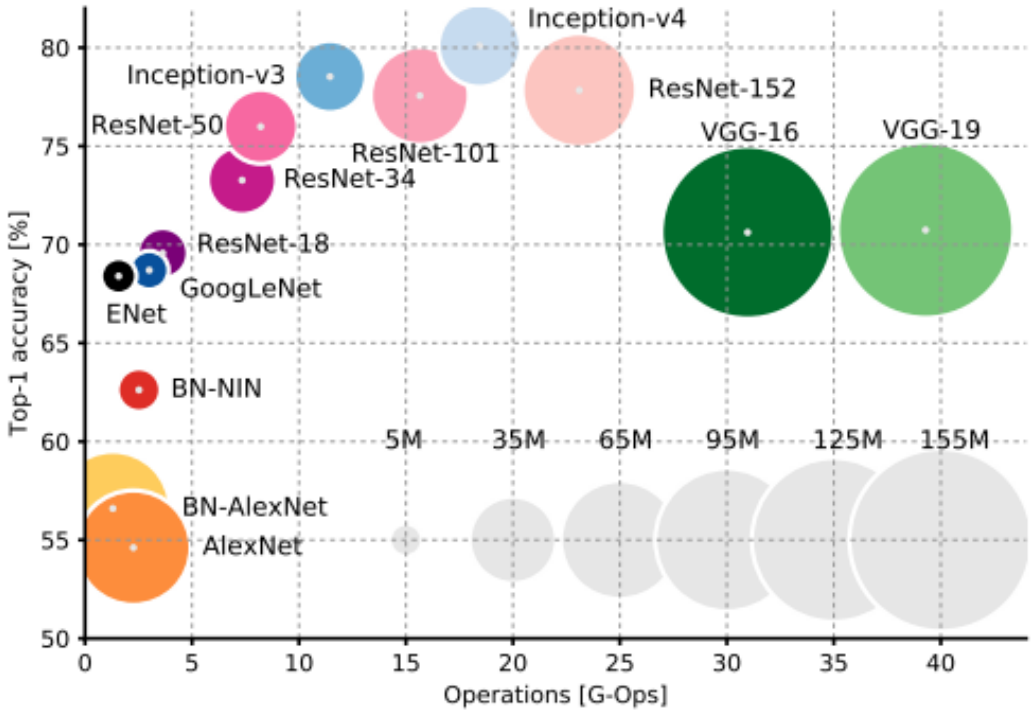


Fig. 3. 10 The accuracy and the scale of the state-of-the-art convolutional neural networks for the ImageNet image recognition challenge. The size of the dot represents the number of weights of that network. In general, higher number of weights and operations are required to achieve higher accuracy for the challenge.

3.4.1 Digitized Storage for Convolution

Since the convolution uses an input multiple times, any computed output from layer L-1 should support repeated use of the computations in layer L. However, in the MLP inference engine, the outputs are consumed by discharging current, and therefore do not support reset at different times. For the re-usability of the outputs, they must be digitized and stored in some fast memory (e.g., flip-flops). This requires the addition of time-to-digital converters at the end of the previous layer, and the addition of digital to time converters at the beginning of the next layer (**Fig. 3. 11**).

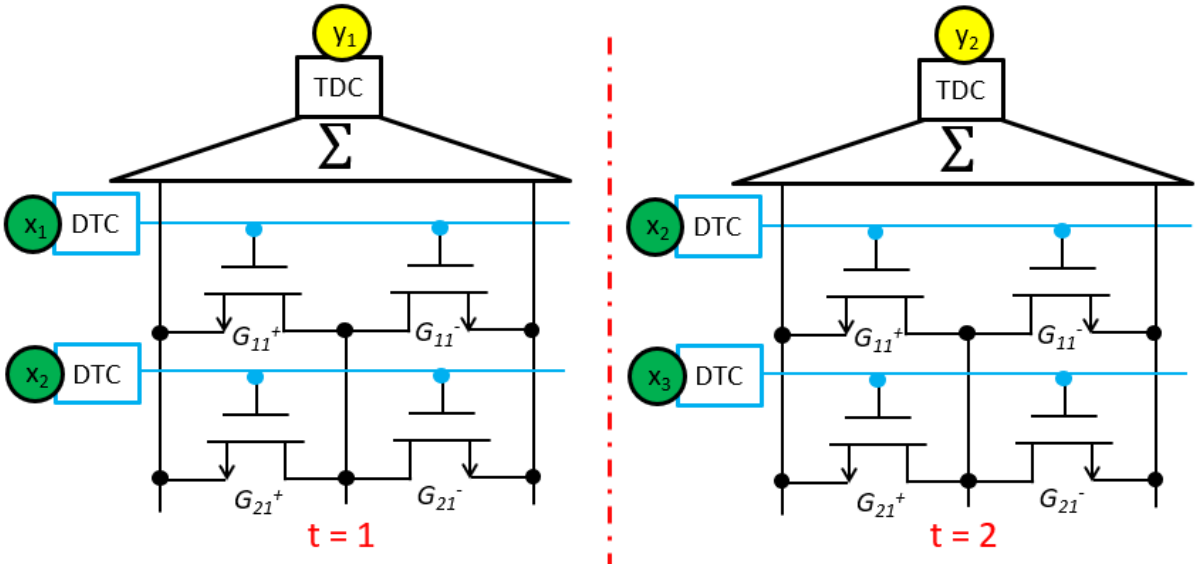


Fig. 3. 11 Compute the CONV layer using the CTT-based MLP engine with the input vector $[x_1,x_2,x_3]$ and output vector $[y_1,y_2]$. The convolution is computed at different times as the inputs switch from the sub-vector $[x_1,x_2]$ to $[x_2,x_3]$. Extra time-to-digital converters (TDC) are required after the outputs from array to digitize the outputs for repeated usage required by the CONV layers. To make it compatible for concatenation, extra digital-to-time converters (DTC) are also required before the inputs to the array to interface with the digitized outputs from the last layer.

This architecture treats the analog device array as a processing block for the VMM computations while the interface to the array block is digital, similar to the existing architectures [Shafiee 16, Cai 19]. This architecture operates similarly to a digital architecture where computed outputs are stored and repeatedly used for the convolution. However, one major difference is that

the weights in the CTT matrix (or based on any other analog device) behave as a read-only memory during the inference due to the low write-speed of the analog memory compared to its read-speed, as discussed in the previous section.

Since the data-conversion is an overhead that does not contribute to the actual computation, the energy-efficiency is not optimal due to the extra energy spent on time-digital conversions. A larger array is more energy-efficient in such architecture due to the sharing of the WLs (sharing the DTC and the register storing the value for repeated use) and BLs (sharing the TDC). Based on synthesized logic in GlobalFoundries 22FDX technology, each 8-bit register with the DTC consumes about P_{DTC} = 0.05 mW and each 8-bit TDC consumes about P_{TDC} = 0.15 mW, both at 1GHz [Moran 20]. Both P_{DTC} and P_{TDC} can be further reduced with state-of-the-art designs. The power of a charge integrator with the ReLU circuit, designed in the same technology consumes about $P_{int} + P_{ReLU} = 0.5$ mW, also at 1GHz [Kittur 20]. Therefore, the power used for computation and digitization of each array is approximately

$$P_{digitize} = \frac{1}{2}(R * P_{DTC} + C * P_{TDC})$$

$$P_{compute} = \frac{1}{2}C * (P_{int} + P_{ReLU}) + \alpha_{CTT} * R * C * I_{D,CTT} * V_{D,CTT} + R * \frac{C_{WL}V_G^2}{2t_{in,max}}$$

$$P_{total} = P_{compute} + P_{digitize}$$

where the $\frac{1}{2}$ factor in the equations reflects the balanced pipeline structure shown in **Fig. 3. 8**. α_{CTT} is the average activity factor of the PWM input signals based on the average CTT weights (i.e.,$I_{D,CTT}$ of the twin-CTT cell). $\frac{C_{WL}V_G^2}{2t_{in,max}}$ is the average power to charge up each word-line of the array with capacitance C_{WL} approximated from physical design (200fF / 128 columns [Moran 20]),

in the input duration of $t_{in,max}$ as discussed in **Fig. 3. 8**. Using $\alpha_{CTT} = 0.25$, the amount of power used for computing and the amount of power used for digitizing for each CONV layer of AlexNet is shown in **Fig. 3. 12**. A significant portion of the energy used in the system is thus spent on digitization instead of the actual computing from CTT devices and charge integrator & activation circuits.

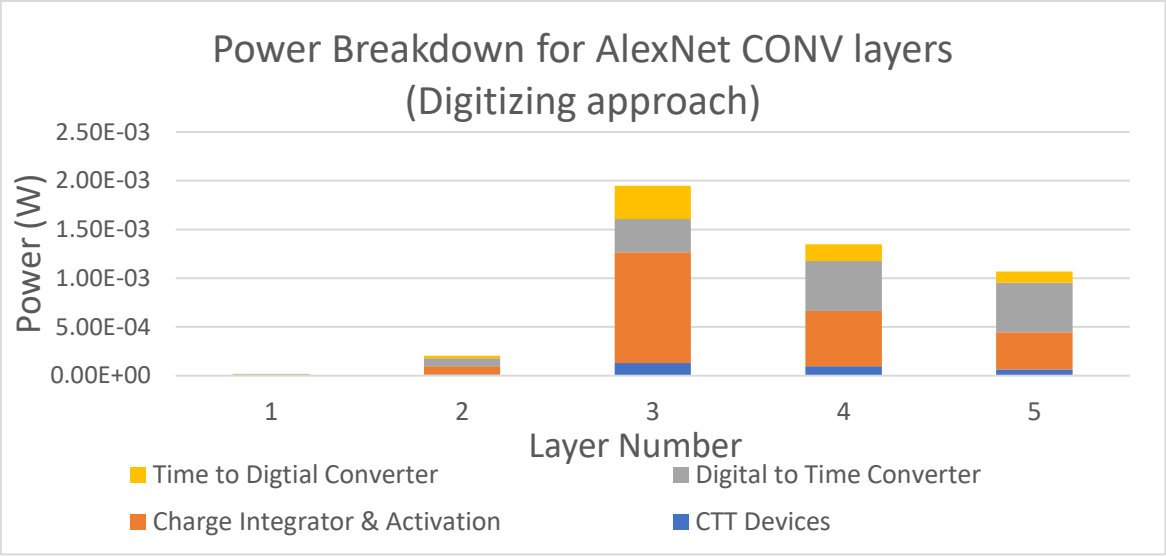


Fig. 3. 12 The power used for analog computing and digitizing CONV layers in AlexNet. A significant amount of power is used for analog-digital conversion, limiting the energy efficiency of the system.

3.4.2 Unroll CONV Layers by Weight Duplication

To maximize the energy-efficiency of the computations, unnecessary data-conversion steps and repeated computations should be avoided. The MLP structure discussed in Section 3.3 is a good candidate for this purpose. Therefore, CNN should be converted to an equivalent MLP structure to leverage the MLP computing architecture. The FC layers are already the same as the MLP layers, and therefore the remaining part is the CONV layers. The CONV layer can be unrolled to the MLP layer with multiple copies of the duplicating weights. As shown in **Fig. 3. 13**, all synaptic weights in the equivalent MLP layer are either duplicate of the CONV layer's weights or suppressed to zero. In addition, the direct mapping from a CONV layer to one gigantic array yields very low array-efficiency since only CTTs around the diagonal positions are used. For better area-

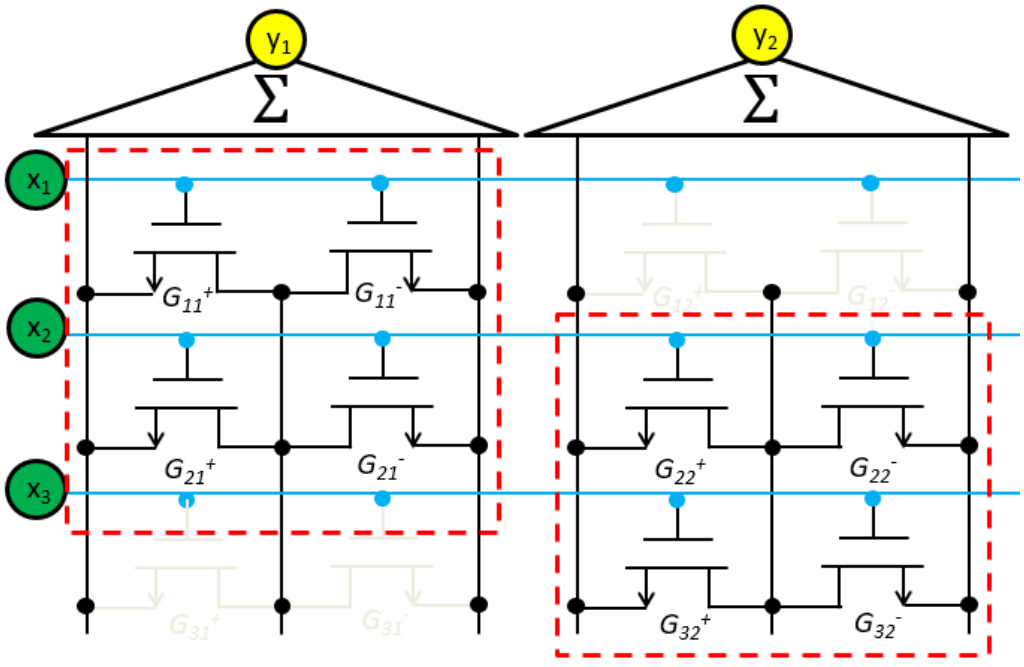
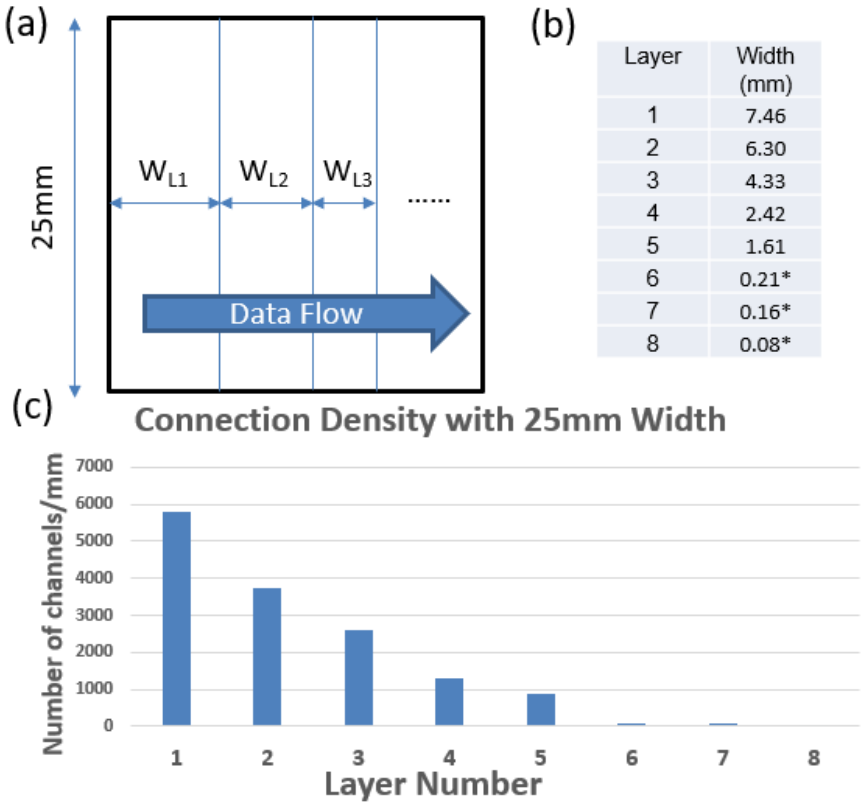


Fig. 3. 13 Compute the CONV layer by expanding it to a FC layer (a layer of MLP). The CTT-based CONV layer computation with input vector [x_1,x_2,x_3] and output vector [y_1,y_2]. All inputs are connected to the expanded array to compute all the outputs at the same time with the duplicated weights (i.e.,$G_{22} = G_{11}$, $G_{32} = G_{21}$). Unused weights (i.e.,G_{12}, G_{31} in grey) need to be programmed to zero. The expanded array can be effectively partitioned to separated arrays (each has one charge integrator) to reduce the amount of unused CTT cells. But the capacitive load needs to be driven by the input PWM generator becomes large regardless of the partition.

efficiency, this array can be partitioned into smaller arrays, as shown in **Fig. 3. 13**. While the unrolling architecture requires little addition to overhead hardware, it increases the number of CTTs and CTT arrays needed for each layer. The estimated total area is 565mm^2 with all peripheral circuits [Kittur 20, Moran 20]. Assuming that this area can be roughly fit into a 25mm * 25mm silicon area, the layers can be mapped onto different slices on the silicon, as shown in **Fig. 3. 14(a)**. The required width of the silicon area for each network layer is shown in **Fig. 3. 14(b)**. The required density of the connections between every two layers is summarized in **Fig. 3. 14(c)**, which is higher for the initial layers. For example, the density of channels between the first two layers is



Layer	Width (mm)
1	7.46
2	6.30
3	4.33
4	2.42
5	1.61
6	0.21*
7	0.16*
8	0.08*

Fig. 3. 14 (a) Mapping the unrolled AlexNet on a 25mm * 25mm silicon area. All layers are arranged consecutively so the data flows from the left to the right of the system. **(b)** The approximated width of each slice. The numbers for the last few layers (layer 6, 7, 8) might be larger due to the limitation of physical design, but the increase will be insignificant. **(c)** Required connection density for each layer's output.

Table 4 Area and power estimation of an unrolled AlexNet implementation based on the CTTs

	Weights		Unrolled Layer Area (mm^2)			Unrolled Layer Power (W)		
Alext Net layer	Original weights	Unrolled weights	CTT	Peripheral circuits	Total area	CTT (DC + AC)	Peripheral circuits	Total power
1	3.5E+04	1.1E+08	5.3E-01	1.9E+02	1.9E+02	2.6E+00	7.3E+01	7.5E+01
2	3.1E+05	2.2E+08	1.1E+00	1.6E+02	1.6E+02	5.6E+00	4.7E+01	5.2E+01
3	8.8E+05	1.5E+08	7.5E-01	1.1E+02	1.1E+02	3.7E+00	3.2E+01	3.6E+01
4	6.6E+05	1.1E+08	5.6E-01	6.0E+01	6.1E+01	2.8E+00	1.6E+01	1.9E+01
5	4.4E+05	7.5E+07	3.7E-01	4.0E+01	4.0E+01	1.9E+00	1.1E+01	1.3E+01
6	3.8E+07	3.8E+07	9.4E-02	5.2E+00	5.2E+00	9.4E-01	1.0E+00	2.0E+00
7	1.7E+07	1.7E+07	4.2E-02	4.0E+00	4.0E+00	4.2E-01	1.0E+00	1.4E+00
8	4.1E+06	4.1E+06	2.0E-02	1.9E+00	2.0E+00	1.0E-01	5.0E-01	6.0E-01
Total	6.1E+07	7.2E+08	3.5E+00	5.6E+02	5.6E+02	1.8E+01	1.8E+02	2.0E+02

about 6000/mm, which can be implemented by using six layers of metals with 1 μm pitch, or fewer layers if the metal pitch can be made finer. **Table 4** shows the number of weights (i.e., a twin-CTT cell) required for the entire network before and after the unrolling needed for each layer of the AlexNet [Krizhevsky 02]. **Table 4** also shows the area and power estimation of the CTTs and peripheral circuits for the unrolled AlexNet. Communication power consumed in the communication channels can be estimated by CV^2f, assuming the main capacitance is the wire self-capacitance of 0.2fF/ μm. Using the layout from **Fig. 3. 14(a)**, with V = 0.8V, f = 5MHz, we approximate that the communication power is less than 10% of the system power estimated in **Table 4**, depending on the actual routing.

The advantages of the unrolling approach include the following:

1. The simplicity of the architecture: by enrolling the CONV layers to the form of MLP layers, data flow in the system never changes, and no explicit instruction is required by the architecture.
2. Low latency: the total latency of the system roughly scales linearly to the number of layers.

3. High throughput: the latency of each stage (i.e., layer) is easy to balance due to the similarity of the structure and operation of each stage. Therefore, the system, when pipelined, will have no idle arrays and therefore produce a very high throughput.
4. High energy-efficiency: as stated at the beginning of this subsection, unnecessary data-conversion is avoided so that the energy-efficiency of this architecture will be higher than the digitization approach discussed in Section 3.4.1. Due to the high utilization of the components in the time domain (i.e., the reason for high throughput), and the simple data-flow, energy-saving techniques such as power gating can be implemented more easily.

The unrolled architecture effectively trades the silicon area for throughput and efficiency. Since the weight duplication implies the duplication of the peripheral circuits such as the charge integrator, and more arrays required by the system. The area of the charge integrator can take a significant portion of the area of the chip, depending on the geometry of the array. Since the charge integrator is an analog circuit and requires a capacitor for charge storage, it would not scale following the Moore's Law. Therefore, the area requirement of the system increases dramatically as the number of layers in the CNN increases. Scaling out to multiple chips for a system will be required for large and deep CNNs when the CONV layers are to be unrolled.

3.5 Summary

This chapter described the architecture of the CTT-based analog inference engine. By using the natural laws for computing, CTT arrays can be built for vector-matrix multiplication and with some peripheral circuits, for multi-layer perceptron inference engines. However, for convolutional neural networks, the use of a computed analog output (e.g., charge on a capacitor) limits their compatibility with the MLP inference engine.

Two solutions are proposed for CNN inference engine architecture. Either the analog outputs can be digitized for convolution, or the CONV layers can be unrolled to equivalent MLP layers by weight duplication. While the first architecture requires the minimum amount of on-chip CTT memory, its energy-efficiency is limited by the overhead used for data-conversion between the analog and digital domain, which does not contribute to the computing process directly. On the other hand, the unrolling architecture eliminates the data-conversion overhead by adding computing hardware, the duplicate of CTT arrays (with associated peripheral circuits), for very fast convolution so that the analog outputs can be reused simultaneously by the arrays from the next layer. While the weights are duplicated, the computation performed by those added arrays are not redundant, and all contribute to the computing throughput of the system.

The main tradeoff between the two approaches is between the energy-efficiency and throughput, with the scale of the system. **Fig. 3. 15** shows the amount of CTTs needed for both methods with respect to the throughput, measured in Tera-operation per second (TOPS), and energy efficiency, measured in TOPS/Watt. There is a large design space between these two cases where digitization can help to reduce the size of the system, while still granting a good energy efficiency and throughput, which can be realized by digitizing every several layers instead of every layer. **Table 5** shows the system comparison between the proposed unrolling structure with some

other state-of-the-art implementations for low-power applications. The proposed CTT-based CNN analog inference engine shows promising energy efficiency and area efficiency for computation.

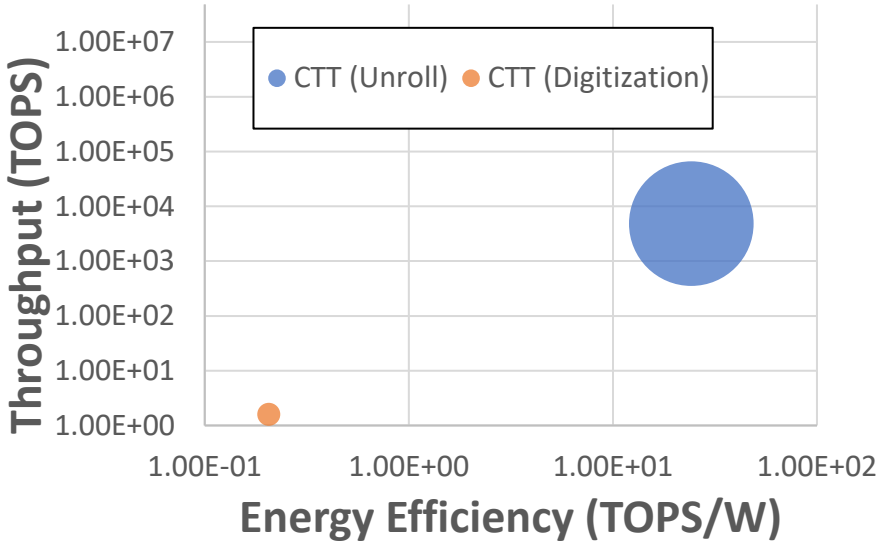


Fig. 3. 15 The throughput and energy efficiency of the CTT systems for AlexNet based on unrolling and digitization. The size of the dot represents the approximated chip size of the system, 565mm^2 (unroll) and 19mm^2 (digitization).

Table 5 Low-power CNN system comparison

	Tech	**Precision**	**Energy Efficiency (TOPS/W)**	**Area Efficiency (TOPS/mm^2)**
Analog CTT, with CNN unrolled	22nm	Analog	24.2	85.5
ISSCC20 [Yuan 20]	65nm	8 bit	6.71	0.0124
ISSCC19 (1) [Song 19]	8nm	8 bit	11.5	0.347
ISSCC19 (2) [Yue 19]	65nm	5 bit	3.76	0.00663
Eyeriss [Chen 17]	65nm	16 bit	0.275	0.00343

Chapter 4: Scalability of Large-Scale Analog Neural Networks

4.1 Introduction

Analog neural networks (ANN) are the neural networks (NN) deployed in analog computing systems, such as the computing engine discussed in the last chapter. Since analog computing is based on physical quantity (i.e., charge) and physical laws (i.e., charge integration and charge conservation) instead of symbolic representation (i.e., numbers) and arithmetic laws (addition and multiplication), the precision of the analog computing is potentially infinite by nature. In contrast, digital computing can never achieve infinite precision due to the discrete nature of the symbols used (e.g., 8-bit per digital value).

However, the precision of the analog system is limited by the non-ideality of the hardware manifested by the intrinsic error of the analog device and circuit. **Fig. 4. 1** shows some sources of errors in the analog system. For example, the PWM signal sees the physical wires of the WLs as capacitive load and, therefore, will be distorted during "on"/ "off" transistor. However, the major source of error is from the analog information, such as the "on" current (e.g., I_{inf}), stored in the analog devices, which can be up to several percentages of its range [Guo 17, Burr 16, Zheng 18]. Since this error is a stochastic property of the device itself, it cannot be corrected by other circuits and will inevitably propagate to all calculations. Therefore, it is crucial to evaluate whether a network can still be useful with the error from analog computations, especially from analog devices to indicate whether an analog device technology (e.g., CTT) is feasible to build a real system with the proposed analog architecture.

In this chapter, we propose a framework to evaluate the degrading effect on the performance of scaled (wide and deep) NNs due to various errors, with a focus on the device errors, and a solution to make ANNs more resilient to such error, especially for large-scale systems.

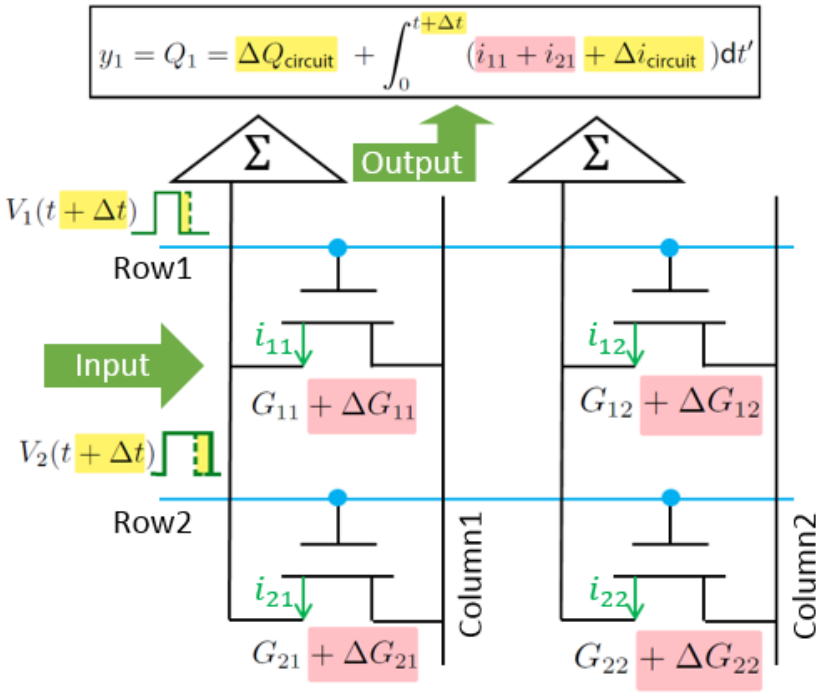


Fig. 4. 1 Analog errors in the proposed analog computing architecture; the device array for analog computing from **Fig. 3. 1Fig. 3. 13** with some errors marked. Device related error terms are highlighted in red ($G_{inf} = \frac{I_{inf}}{V_D}$), and the circuit related error terms are highlighted in yellow.

4.2 Hardware-Based Simulation Framework for Analog Neural Networks

First, we establish a simulation framework for analog NNs for both MLPs and CNNs, based on the CTT characterization results and models in Chapter 2. The continuous nature of the analog devices error is fundamentally different from the discrete error in digital memories, which have been studied for neural networks [Reagen 18]. While there is existing neural network simulator for analog in-memory NN accelerators [Chen 18], it digitizes the output of the array (i.e., the result of the vector-matrix multiplication) and therefore does not match the pure analog architecture (e.g., MLP and unrolled CNN) proposed in the previous chapter.

In the proposed analog architecture, synaptic weights of the neural network are represented by the read current of the device (e.g., I_{inf} for CTT). Therefore, the synaptic weights are random variables derived from the Gaussian error statistics of the CTT devices (e.g., **Fig. 3. 3)** to match the physical (I_{inf}) and numerical domains. We define the device noise as the ratio between the standard deviation of the I_{inf} error and the entire range of device I_{inf}, which is around 4% - 7% for twin-CTT cells (**Table III**), and can be similar or more for the other analog devices [Burr 16, Guo 17, Zheng 18].

To map this device noise onto the numerical domain for the synaptic weights, the maximum and minimum synaptic of each filter (i.e., an array) of the neural network is calculated. Since the maximum is always positive and the minimum is always negative, the one with larger magnitude (i.e.,$w_{absmax} = \max(w_{max}, |w_{min}|)$) is used, and this value is mapped to the maximum of the window, which is always symmetric to zero for the twin-CTT cells. As a result, the mapping coefficient between the device conductance and the weight is $\beta = \frac{G_{max}}{w_{absmax}} > 0$, which is constant

in one filter, but can be different in different filters due to different w_{absmax} values. This is to make sure that the conductance window is fully used during the mapping.

Suppose filter A and B have different $w_{absmax,A} \neq w_{absmax,B}$ then $\beta_A = \frac{G_{max}}{w_{absmax,A}}$ and $\beta_B = \frac{G_{max}}{w_{absmax,B}}$, $\beta_A \neq \beta_B$ since the G_{max} is the same across the system (using the same design of twin-CTT cells). The different ratio values do not affect the result of the computation in the numerical domain because it can be compensated in hardware by adjusting the magnitude of the discharging current during the output PWM signal generation. Without losing generality, we prove this by showing that different conductance mapping coefficient β can produce the same multiplication result in the numerical domain by using a different discharging current $I_{discharge}$ when ReLU is used as the activation function.

Suppose a synaptic weight w_A in filter A has the same numerical value with a synaptic weight in filter B, i.e.,$w_B = w_A$. Then for the identical input of x, the multiplication output y must be equal:

$$y_A = xw_A = xw_B = y_B$$

We write the input as t_x since it is PWM, then the charge accumulated by this computation is

$$Q_A = V_D G_A t_x$$

$$Q_B = V_D G_B t_x$$

where V_D is the constant bias in the system (e.g., $V_D = 50mV$ for the I_{inf} characterized in previous chapters). The mapping is $G_A = \beta_A * w_A$ and $G_B = \beta_B * w_B$, assuming $\beta_A \neq \beta_B$ and thus $G_A \neq G_B$.

If $w_A = w_B \leq 0$, since $\beta > 0$, $G_A < 0, Q_A < 0$ and $G_B < 0, Q_B < 0$. No output will be produced in either case and therefore

$$t_{y_A} = t_{y_B} = 0$$

If $w_A = w_B > 0$, then the ReLU function simply passes the pulse width from discharging

$$t_{y_A} = ReLU\left(\frac{Q_A}{I_{discharge,A}}\right) = \frac{Q_A}{I_{discharge,A}}$$

$$t_{y_B} = ReLU\left(\frac{Q_B}{I_{discharge,B}}\right) = \frac{Q_B}{I_{discharge,B}}$$

The computation in the numerical domain, $y_A = w_A * x = w_B * x = y_B$, requires that $t_{y_A} = t_{y_B}$ in the physical domain. Therefore

$$t_{y_A} = \frac{Q_A}{I_{discharge,A}} = \frac{V_D G_A t_x}{I_{discharge,A}} = \frac{V_D \beta_A * w_A t_x}{I_{discharge,A}} = t_{y_B} = \frac{Q_B}{I_{discharge,B}} = \frac{V_D \beta_B * w_B t_x}{I_{discharge,B}}$$

which will always hold if $\frac{\beta_A}{I_{discharge,A}} = \frac{\beta_B}{I_{discharge,B}}$. This indicates that the mapping coefficient is controlled by the discharge current. In the extreme case, this mapping coefficient can be different for each charge integrator, if the discharging current can be set differently, but this implies extra storage for each charge integrator.

From this mapping method, the Gaussian error model of the analog device conductance (i.e.,I_{inf}/V_D) is mapped to the Gaussian error model of the synaptic weights by $\Delta w = \Delta G/\beta$ for simulation. The calculation of a filter including errors in the numerical domain is thus

$$y = ReLU((W + \Delta W)(x + \Delta x) + \Delta h) + \Delta y$$

where W is the synaptic weight matrix, ΔW is the error of the weight matrix in which all entries are sampled from the independent and identically distributed random variable Δw. x represents the input PWM signal, and Δx is the error of the input. Δh is the error of the activation circuit. Δy is the error from the transmission of the PWM output to the next layer.

To enable the simulation for state-of-the-art deep neural networks such as ResNet [He 16], novel structures and operations, such as batch normalization [Ioffe 15], shortcut layers and residual block are implemented in the simulation framework based on the proposed analog architecture.

Batch Normalization layers is essentially a linear operation, normalizing its input for each channel individually with learned parameters (mean μ_c, variance σ_c^2, the learnable scaling factor γ_c and learnable bias β_c, for each channel c):

$$y_{c,i} = w_{eff,c}\, x_{c,i} + b_{eff,c}$$

where $w_{eff,c} = \frac{\gamma_c}{\sqrt{\sigma_c^2+\epsilon}}$, $b_{eff,c} = \beta_c - \frac{\gamma_c \mu_c}{\sqrt{\sigma_c^2+\epsilon}}$. Therefore, it can be implemented through convolutional layers with unit-size, unit stride convolutional kernel, which has the weight $w_{eff,c}$ and bias $b_{eff,c}$.

Shortcut layers were introduced by ResNet [He 16] to address the gradient vanishing problem, and have become an indispensable component in deep neural networks. In our noise-considering implementation of shortcut connections we assumed that each positive entry of the identity matrix suffers from a Gaussian noise U with zero mean and variance δ, where δ is the device noise, and $I_{shortcut} = U + I$. This is to reflect the noise from possible circuit implementation of the shortcut (e.g., current mirror).

Residual blocks used in ResNet are implemented based on the CNN layers and shortcut layers. The parameters for the behavior of analog devices (e.g., mean and variance in the case of Gaussian noise) can be individually specified for each layer.

After a network is trained, the simulator is used to evaluate the performance of the network using analog devices. During the forward-propagation, perturbations for all weights from all layers are sampled from the specified random variable before one test run. The same set of sampled weights is used for the entire test set to generate the accuracy scores. Since the weights are now stochastic, 50 testing runs are performed (unless specified otherwise) to obtain the statistics of the accuracy scores for a given analog deep neural network.

4.3 Resiliency of Analog Neural Networks

An MLP with 784 input nodes, 99 nodes in its only hidden layer, and ten nodes as output, with ReLU after each layer, is trained on conventional 64-bit CPU using backpropagation [LeCun 89] and stochastic gradient descent, for the MNIST hand-written digit classification problem [LeCun 98]. Then the trained weights and biases are loaded into the simulation framework. Three types of errors are first evaluated in the simulation:

1. Device errors are defined as a Gaussian random variable with zero mean and a standard deviation of Δw, where $\Delta w = \Delta G/\beta$ as explained in the last section (**Fig. 4. 2**).

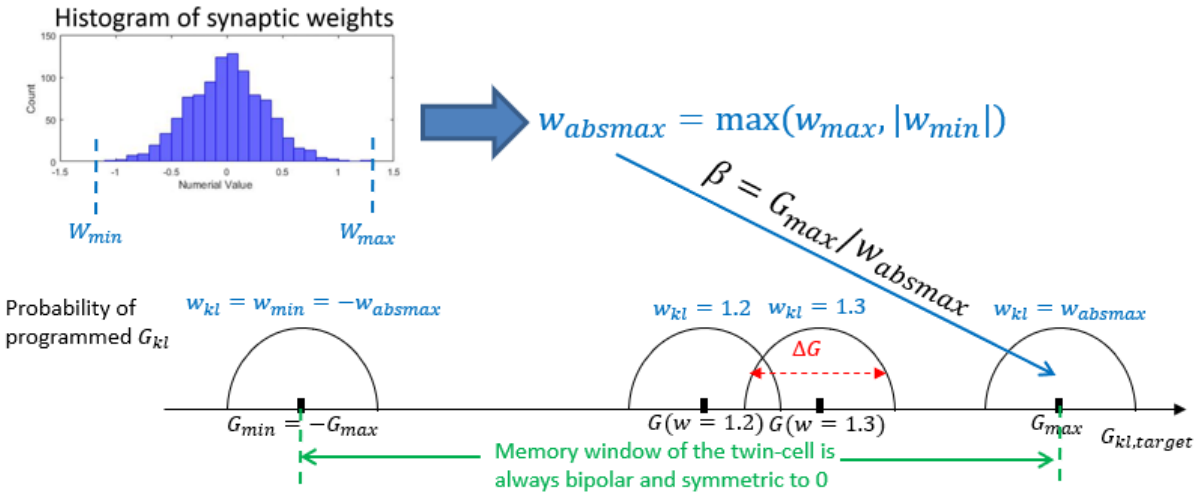


Fig. 4. 2 Mapping numerical values of the synaptic weights (w_{kl}) to the conductance values of the analog device (G_{kl}) by the mapping coefficient β.

2. The error of the input PWM signal, which always has a finite rise and fall time seen by the gates of the CTTs due to the capacitance on the path (**Fig. 4. 3**). Because of the exponential behavior of I_D-V_G at the subthreshold region, the net of current drawn from the device is always smaller than the ideal case. Since the rise times and fall times are not a function of the ideal pulse width itself, this error is more salient when the pulse width is short (i.e.,

when this input value is small). To model this error, we assume that for a fixed duration of Δt_{PWM}, the average current of any device is reduced to $k_{PWM} I_{inf}$ where $k_{PWM} < 1$.

3. Noise current, mainly the white noise integrated as the charge integrator. This is a property of the charge integrator and therefore is only a function of the duration of the integration. This duration is fixed for all arrays, and the error is modeled at each column as an addition of charge ΔQ_{int} (before the ReLU function), which is an independent Gaussian random variable with zero mean and a constant standard deviation.

The input to the first layer is converted to 8-bit PWM signals. The output of the first layer is not digitized (but limited to the 16-bit floating-point precision of the double-precision data type in the simulator) before converted to PWM signals as the input of the hidden layer. ReLU is applied as the activation function. The output node with the longest PWM represents the classification result.

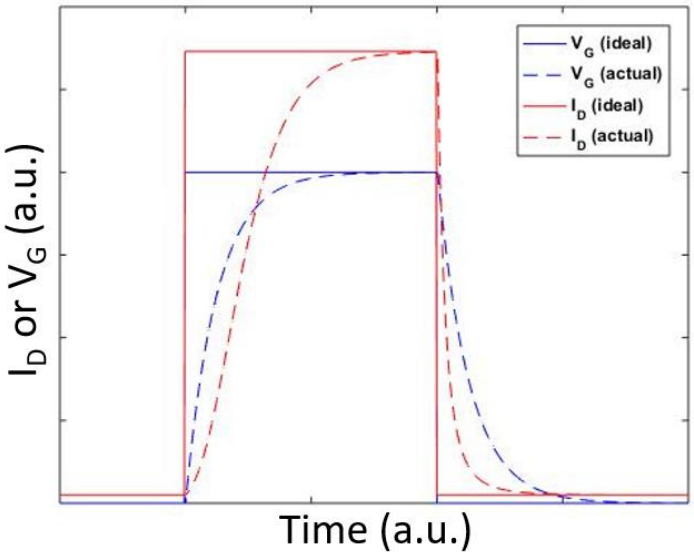


Fig. 4. 3 The input PWM signal at the WL always has a finite rise and fall time due to RC delay, so that the gate voltage of the device at the PWM period receives a smeared pulse, making the integrated current of the device always smaller than the ideal case due to the exponential relationship between I_D and V_G at the subthreshold region.

Fig. 4. 4 shows the accuracy of the MLP for the test set of the MNIST dataset from 100 Monte Carlo simulations across a range of device noise. The parameters used are $\frac{\Delta w}{W_{range}} = 5\%$, $\Delta t_{PWM} = 10$ cycles, $k_{PWM} = 0.8$, $\Delta Q_{int} \sim N(0,0.255) * 1pC$. The circuit related noises are all exaggerated from designs based on unpublished analog circuit design using 22FDX technology [Moran 20, Kittur 20]. The classification accuracy degrades slightly from the 16-bit digital baseline in most of the cases, due to all the injected errors, and the device error is the main contributor to this degradation after 10%, which is the device noise level for many analog devices.

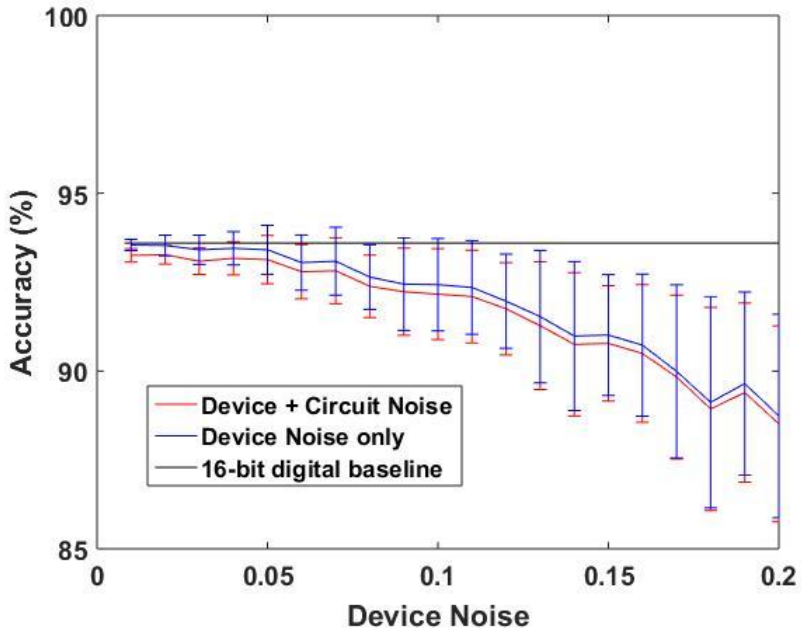


Fig. 4. 4 Accuracy of the analog MLP simulated with the errors from analog device and the peripheral circuits from 100 Monte Carlo simulations at different device noise level. Errors from the peripheral circuits are exaggerated to reveal its effect on the network. Baselines of 16-bit digital network and device error only networks are also plotted, showing that the main contribution to the network degradation is from the device error when device noise is larger than 10%.

To accelerate the Monte Carlo simulations for very large and deep neural networks, only device noise is considered as non-zero when we simulate the deep convolutional neural networks (e.g., WideResNet [Zagoruyko 16]). This focuses on qualifying the analog device technology with respect to the scaling of the neural networks.

The simulation framework proposed in the last section is used to simulate DNNs based on Wide-ResNet models. It includes many of the advanced operations of state-of-the-art neural networks (e.g., batch normalization, residue layers), which are affected by the device noise. The DNN model is trained on commercial GPUs using 32-bit floating-point precision and the ImageNet training set. During testing of the trained network, device models are included to emulate the behavior of analog devices. Each pre-trained digital network is instantiated 50 times by independent sampling from the device noise statistics (i.e.,$w = w_0 + \Delta w$). Then the system is evaluated by all testing patterns on all instantiated networks to obtain the statistics for network accuracy.

Two different structures of Wide-ResNet with depth level 16 (17.1 million weights) and 28 (36.5 million weights) are trained and tested on the CIFAR-100 dataset (same for all network simulation results shown unless otherwise specified). In **Fig. 4. 5**, all networks presented show significant degradation of the network accuracy as the device noise increases, while the larger network is more resilient. When device noise is 6%, the top5 accuracy is degraded from 94.28% to 44.96% for the 16-layer network and from 94.39% to 70.99% for the 28-layer network. Both networks start to fail completely at a device noise of 14%. The deeper network (also with more weights) is less sensitive to the increase of device noise, and therefore is more resilient to device noise. The resiliency of the network also depends on the application. **Fig. 4. 6** shows the network accuracy of the 16-layer WideResNet on the CIFAR-100 problem, compared with the MNIST

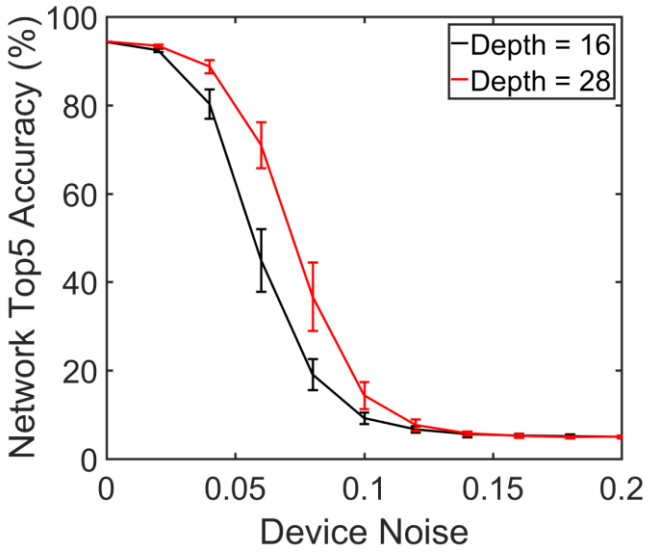


Fig. 4. 5 The degradation of network due to analog device noise: two Wide-ResNet models of depth level 16 and 28 are trained and tested on the CIFAR-100 dataset.

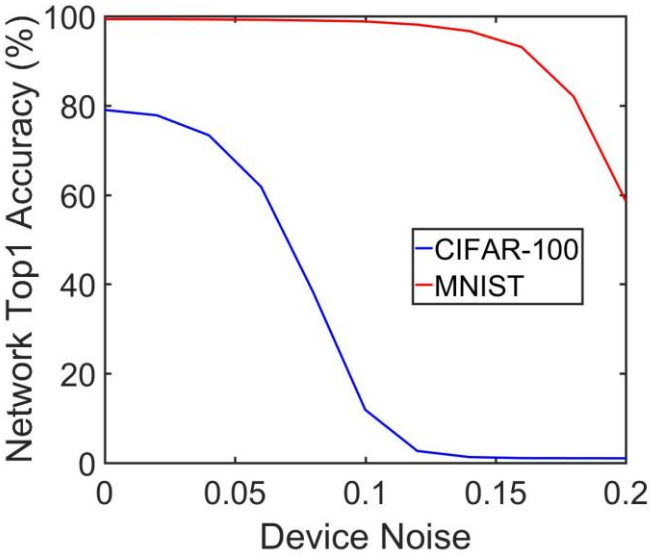


Fig. 4. 6 The degradation of network due to analog device noise for different applications: Wide-ResNet model with depth level 16 for both CIFAR-100 and MNIST are trained and tested, showing that the network is more resilient to device noise when the application is simpler.

recognition problem, which is significantly easier than the CIFAR-100 problem. The resiliency of the neural network of a similar scale is better for easier applications.

The network is also simulated to incorporate the CTT characterization data (one-time use with 1200nA memory window). Since the twin-CTT cell fine-tuning includes over-programming, the top5 accuracy of the CTT-based network increases from 36.40% (right after over-programming) to 83.98% (after 200 hours), which is 10.41% lower than the 32-bit digital baseline (**Fig. 4. 7**).

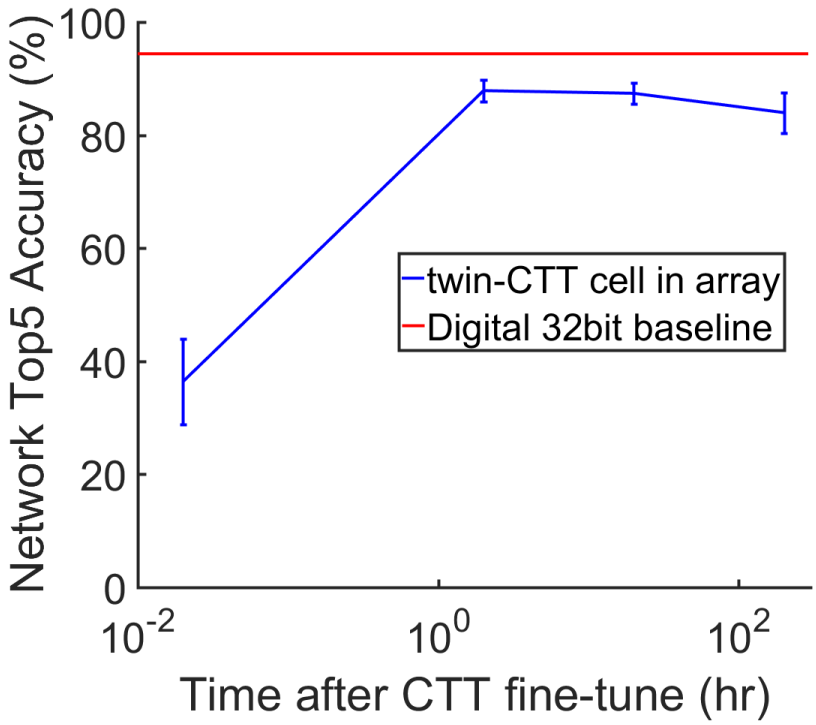


Fig. 4. 7 Simulate network with CTT characterization data: accuracy of the 28-layer network from **Fig. 3. 3** is simulated using CTT characterization data, with the baseline from digital machine of 32-bit data precision.

4.4 Improving Resiliency of Analog Neural Networks

The last section showed the resiliency of analog neural networks, which can be improved by scaling up the network for the same application. In addition, by duplicating *n* analog devices and taking the average of the programmed weight, or running *n* filters in parallel and averaging the output, one can statistically reduce the standard deviation of the device error (i.e., the device noise) by $\frac{1}{\sqrt{n}}$ [Ma 18]. However, they all need significant investment in the hardware of the inference engine. In this section, three methods are shown to improve the resiliency with little or no extra cost in the inference engine's hardware:

1. Use generalization methods such as L2-regularization [Schölkopf 02] and dropout [Srivastava 14]
2. Rescale the bias terms in the network filters
3. Hessian-aware stochastic gradient descent

4.4.1 L2-Regularization and Dropout

Some improvements in both the network accuracy and resiliency can be achieved by standard generalization methods such as L2-regularization and dropout. For example, at 4% device noise, the top5 accuracy can be improved from 68.25% to 90.72%, as shown in **Fig. 4. 8**. However, it is not necessarily true that a network with the best L2-regularization and dropout parameters at zero device noise will also make the best network at a higher device noise (curves in **Fig. 4. 8** cross each other). Therefore, the parameter optimization of the network depends on the target device noise.

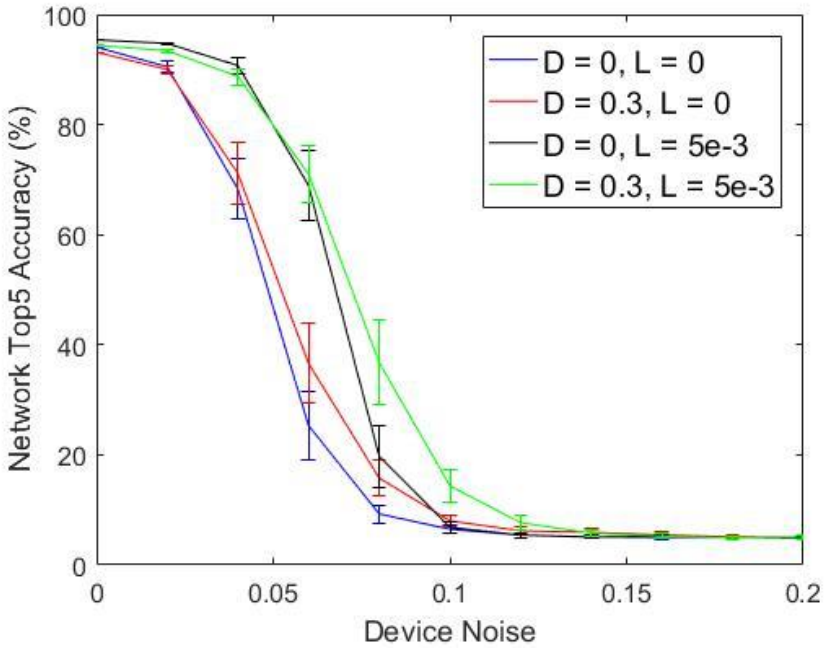


Fig. 4. 8 Effect of conventional generailization methods: Accuracy of Wide-ResNet (depth level 28) is trained with different L2-regularization (L) and dropout factors (D). Some improvement in the analog resiliency can be achieved.

4.4.2 Rescaling Bias Terms

Another improvement in network resiliency is to rescale the bias terms in the filters. As introduced previously in Section 3.1, the bias terms of the layers are combined with the weights and in the VMM engine for the ease of hardware implementation (**Fig. 3. 1**). However, the range of the bias parameters is often higher than that of the weight parameters. As a result, naïvely combining the bias and weight together will increase the parameter range significantly. Since all parameters share the mapping coefficient, a high numerical range of the bias leads to a high noise level for the weights. To address this problem, the bias terms can be scaled, as shown in **Fig. 4. 9**. Instead of using "1" as the extra input for the bias term, the extra input can be other value to correspondingly scale bias parameters to match the range of the programmed biases with the range of weights. This ensures that both weights and biases use the full dynamic range of the device to minimize the effect of the device noise. **Fig. 4. 10** shows that the range matching between the bias

terms and the weight can improve the resiliency of the same trained network. The cost of such implementation is the storage of the scaling factor *s*, which can be different in different arrays.

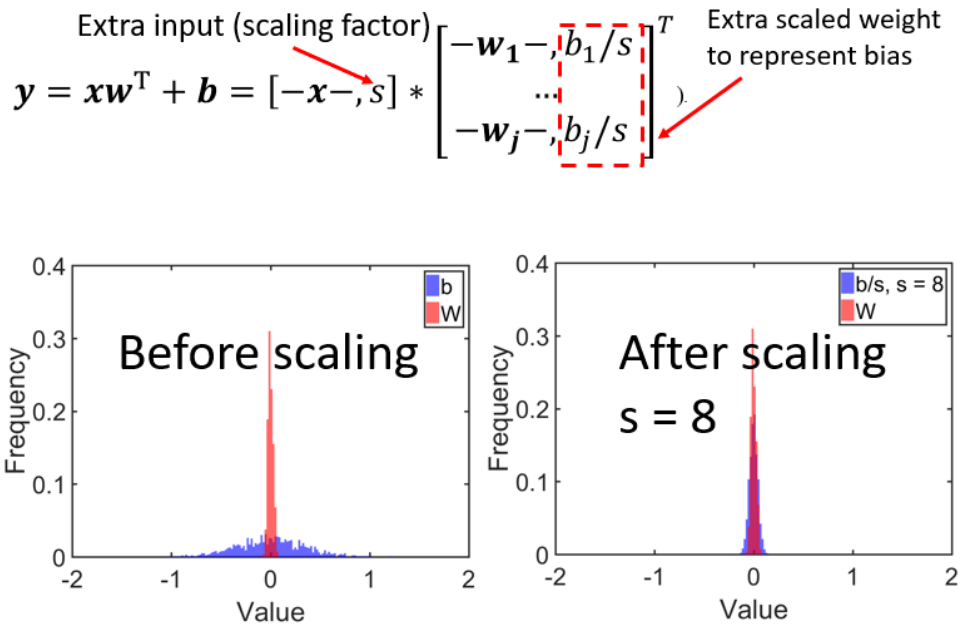


Fig. 4. 9 Top: scaling factor as an extra input per layer to include bias term *b* in the vector-matrix multiplication. **Bottom:** normalized histograms showing the distribution of the weights *w* and bias *b* of a filter before and after the use of scaling factor *s*.

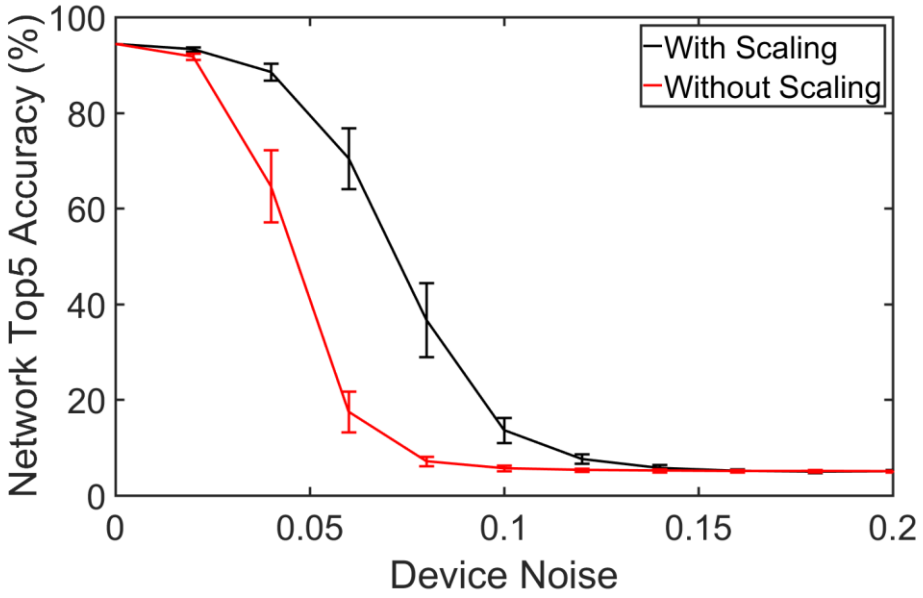


Fig. 4. 10 Network (Wide-ResNet-28) top5 accuracy on CIFAR-100 with and without the scaling for *w* and *b*.

4.4.3 Hessian-Aware Stochastic Gradient Descent

Inspired by the adversarial training [Goodfellow 14], which generates and uses difficult inputs for the network during the training phase to enhance its robustness against adversarial attacks, we propose the Hessian-aware stochastic gradient descent (HA-SGD) method to train neural networks with improved resiliency against analog device noise. The HA-SGD ensures that at convergence, the local minimum of the cost function during training will not be at a point with high-norm Hessian.

As shown in **Fig. 4. 11**, at any given weight W_0, the HA-SGD computes the average of sample gradients, computed at points obtained by random perturbations of W_0, as an estimation of

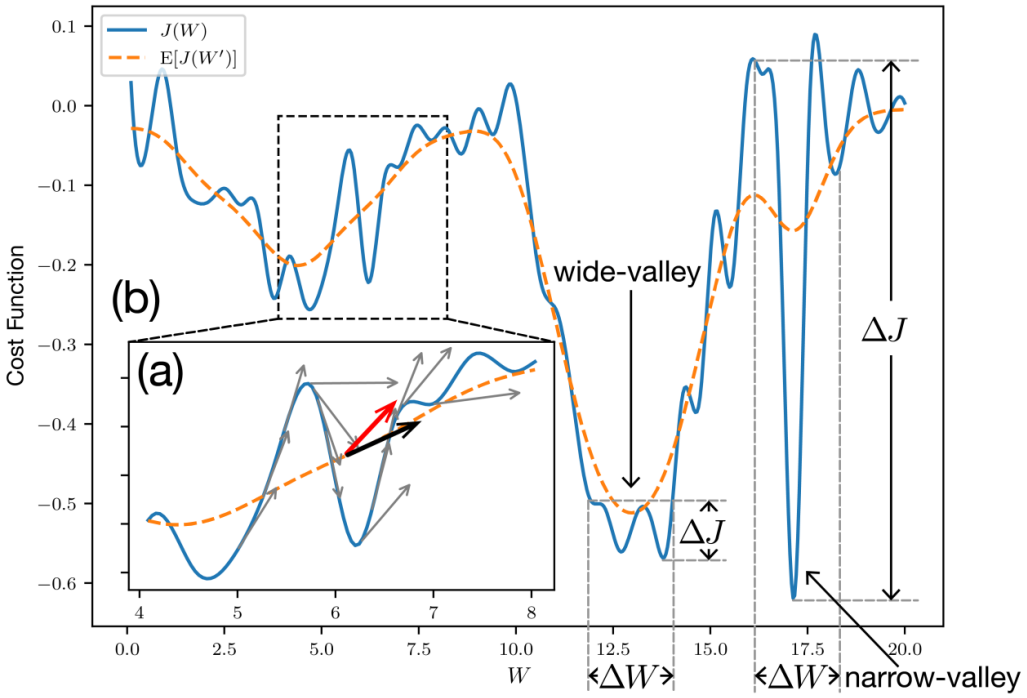


Fig. 4. 11 Proposed novel training method – Hessian-Aware Stochastic Gradient Descent (HA-SGD): The HA-SGD algorithm is used to estimate the gradient of the smoothed cost function (inset) so that the algorithm is more likely to converge at a wider valley of the original cost function (see methods), which will be more resilient to analog device noise. The gradient of the smoothed cost function is computed by averaging gradients computed at neighboring points obtained by random perturbations of the current weights. The variance of the random perturbations of current weights is referred to as the level or intensity of the training noise in HA-SGD in parallel with the device noise previously defined.

the gradient of the smoothed cost function ($\tilde{J}(W) = E_{W'}[J(W')]$). The variance of the random perturbations of current weights is referred to as the level of intensity of the training noise in HA-SGD. Thus, this training algorithm samples the local neighborhood (around W_0) of the landscape.

If the Hessian has a high norm, then the gradient will deviate significantly from the gradient at W_0. In other words, the optimizer seeks local minima in the smoothed cost function, which is more likely to correspond to a wide valley in the original landscape. As a result, in the wide valleys, the neural network perturbed by finite device noise would maintain performance comparable to the case having no noise, with small variance and bias. The HA-SGD can improve the network resiliency significantly. **Fig. 4. 12** shows the enhancement of resiliency when HA-SGD is used for training. At a 6% device noise, HA-SGD improves performance from 70.99% to 88.47%. HA-SGD has an even more significant effect when the device noise is higher. At 10% device noise, a

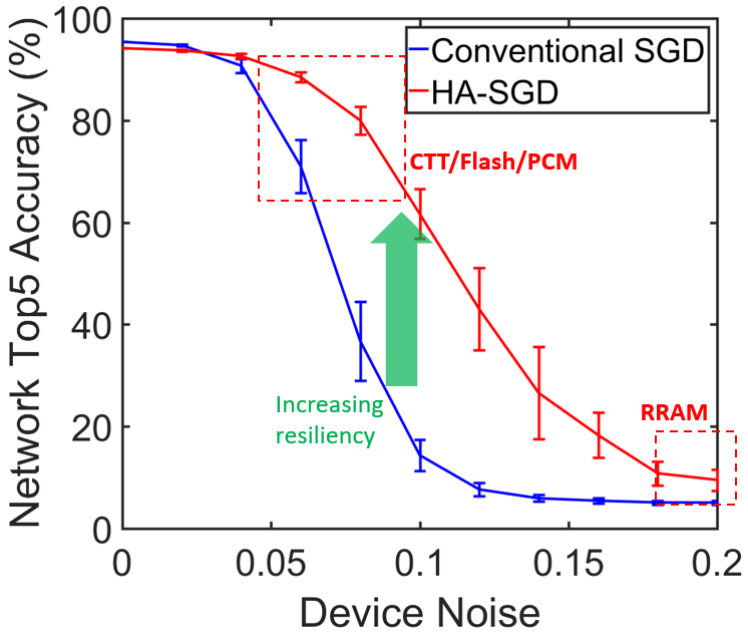


Fig. 4. 12 Improved network resiliency with HA-SGD: the performance of networks trained with the HA-SGD algorithm are compared with those trained by highly optimized conventional training algorithms, showing improved resiliency from the HA-SGD method. Parameters such as L2-regularization factor, dropout factor and the training noise level are all optimized for both cases. The device noise levels of some analog devices, such as CTT, Flash [Guo 17], phase change memory (PCM) [Burr 16], and resistive RAM (RRAM) [Zheng 18] are indicated.

network trained with injected noise of 10% achieves a top5 accuracy of 61.67%, which is more than four times the top5 accuracy of the networks trained by standard SGD (14.37%). The optimal level of the training noise depends on the device noise of the target device (e.g., CTT). **Fig. 4. 13** shows the effect of increasing training noise on different device noise during inference. In general, a higher training noise performs better for higher device noise because the gradient estimation during training is more accurate. Notably, the network tested with zero device noise would not benefit from the HA-SGD algorithm, and therefore HA-SGD should only be applied when some finite device noise is expected in the analog inference engine. **Fig. 4. 14** shows the network top1 accuracy for the WideResNet-28 tested for CIFAR-100. Three groups of networks are trained with different training noise (device noise injection in the forward propagation during training). The

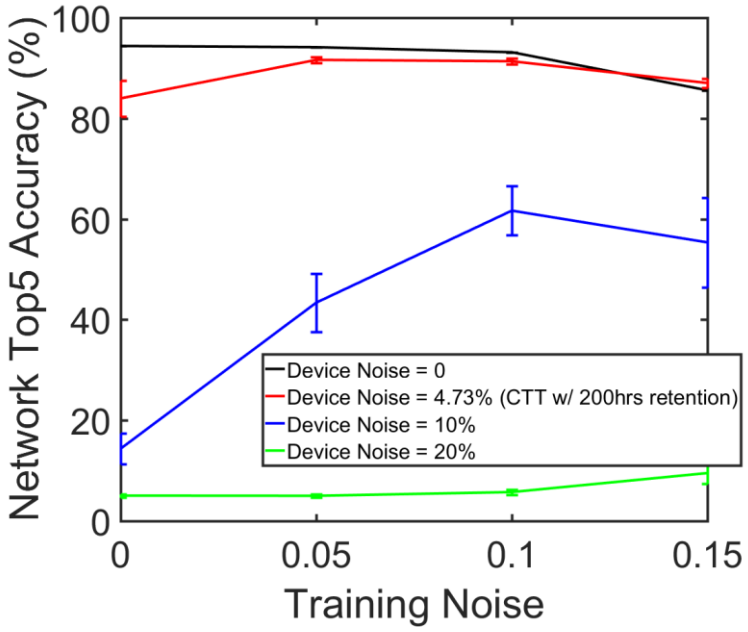


Fig. 4. 13 Effect of noise-level during training: Increasing the level of training noise in HA-SGD has different effect on the resiliency of the network, depending on the device noise of the target device.

optimal accuracy can be achieved when the training noise is close to the expected device noise during testing. Also, in **Fig. 4. 14** some degradation of the network at zero device noise is observed. This implies that the local minimum of the cost function found by HA-SGD is not necessarily deeper, despite smoother. However, this degradation of the network due to HA-SGD can be hidden if the network is trained for a simpler challenge. **Fig. 4. 14** shows the same network (with slight modification at the last layer) trained for MNIST challenge using different training noise, whose accuracy at zero device noise during testing does not change.

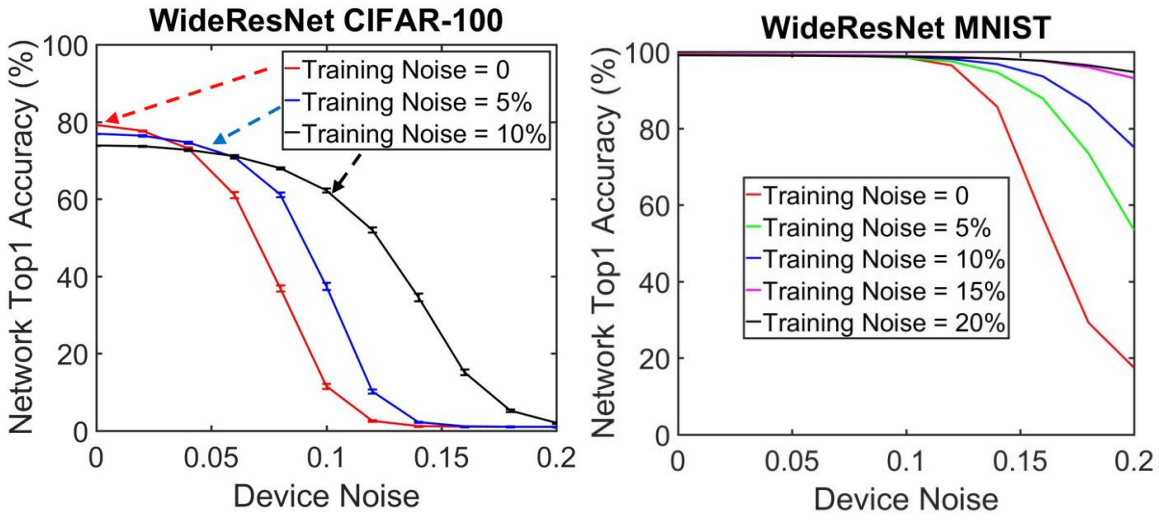


Fig. 4. 14 Left: network top1 accuracy for the WideResNet-28 tested for CIFAR-100. The training noise is preferred to be close to the target device noise during inference for the best network accuracy. At the meantime, the zero-noise inference accuracy might decrease as the training noise increases. **Right:** the zero-noise inference accuracy degradation can be hidden when the network is trained for a simpler problem, such as MNIST.

4.5 Summary

This chapter explored the feasibility of using low-precision analog memory cells as the weights for very large and deep (> 30 million weights) neural networks. A simulation framework is established to include the associated calculation errors for the proposed analog inference engine architecture. The simulation shows that the networks degrade due to analog errors from both the analog device and the circuit, while the contribution from the device is more significant for many analog device technologies. Several methods to improve the analog resiliency without adding much cost to the analog inference engine are discussed. A combination of using generalization methods (L2-regularization, dropout), rescaling the bias terms, and the Hessian-aware stochastic gradient descent can be used to enable resilient analog deep neural networks.

In addition, this chapter showed that neural network training can still converge with noise injection (at proper levels) to the synaptic weights. This indicates the potential of analog training engines, which use analog devices for training analog and digital neural networks. Digital neural networks can be treated as a subset of analog neural networks when the uncertainty of the memory is zero.

Chapter 5: Scaling Analog Neuromorphic Systems

5.1 Introduction

Scaling of the neuromorphic system can, in general, provide benefit to the cognitive capability of the system [LeCun 19] in mainly two aspects:

1. For the same cognitive challenge and the same hardware architecture (e.g., fixed digital precision or fixed analog device error), a larger neural network tends to provide better accuracy (**Fig. 5. 1**). As a result, to reach the same level of network accuracy, a larger network is required for a more difficult cognitive challenge.
2. In the analog system, a larger neural network enhances the resiliency of the system against analog errors for the same cognitive challenge (**Fig. 4. 5**). This holds with or without the resiliency enhancing techniques proposed in Section 4.4.

There are two major challenges in scaling the neural networks:

1. Hardware constraint on the memory. For the digital systems, the extension of memory is through the memory hierarchy to bring in denser but slower memory (e.g., DRAM, Flash) while the on-chip memory (e.g., SRAM, eDRAM) is rewritten when different filters/layers are computed. For analog systems, the on-chip memory is the analog non-volatile memory that cannot be rewritten quickly. Therefore, scaling out the analog system is required by adding multiple chips into the system at the same level of the memory hierarchy, which requires a scalable architecture for the multi-chip system.

2. Overfitting of the data [Hawkins 04] is a general machine learning problem for supervised learning when the network has fit very well on the training data, but cannot generalize the knowledge and work well with the testing/validation data that are not used during training.

This chapter will focus on the scaling of the analog neuromorphic system hardware based on the analog in-memory computing architecture proposed in Chapter 3.

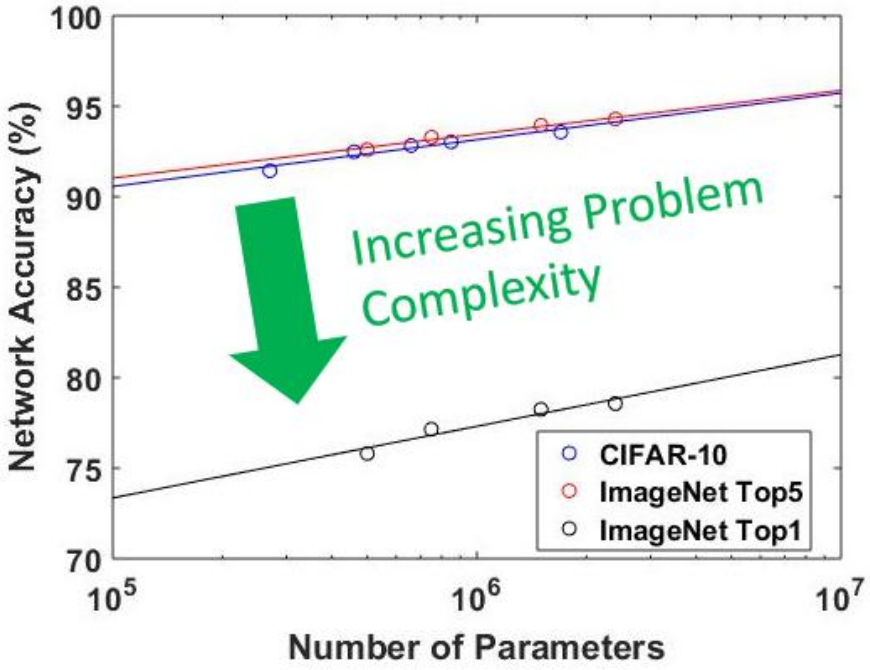


Fig. 5. 1 Improve neural network performance by scaling: the change of the accuracy of the advanced neural network, ResNet [He 16], as a function of network size for different cognitive challenges. Before overfitting becomes an issue, accuracy increases as network size increases, usually due to the increase in depth. In addition, accuracy decreases as the task becomes more difficult (e.g., from CIFAR-10 top1 score to ImageNet top1 score, and from ImageNet top5 score to top1 score). Therefore, network scaling is crucial to reduce error for complicated challenges.

5.2 Scaling out Non-von Neumann architecture

The multi-layer structure of the proposed analog in-memory computing engine is a non-von Neuman architecture suitable for scaling. As shown in **Fig. 5. 2**, a multi-layer network can be partitioned at the interface between arrays and after the generation of the output PWM signals. This interface can be implemented as either an analog interface or a digital interface.

If an analog interface is used, the communication is to pass the timing information of the PWM signals between chips, replacing the on-chip wiring if both arrays are on the same chip. Depending on the quality of the channel, buffers can be added at both ends of the channel.

If a digital interface is used, the communication is to pass the binary data that carries the timing information of the PWM. This data needs to be generated by a time-to-digital converter at the sender's end, and recovered by a digital-to-time converter at the receiver's end. Therefore, this requires a higher cost of extra hardware than the analog interface. However, the extra cost will be reduced if the communication between layers is already digitized for CNNs, as described in Section 3.4.2.

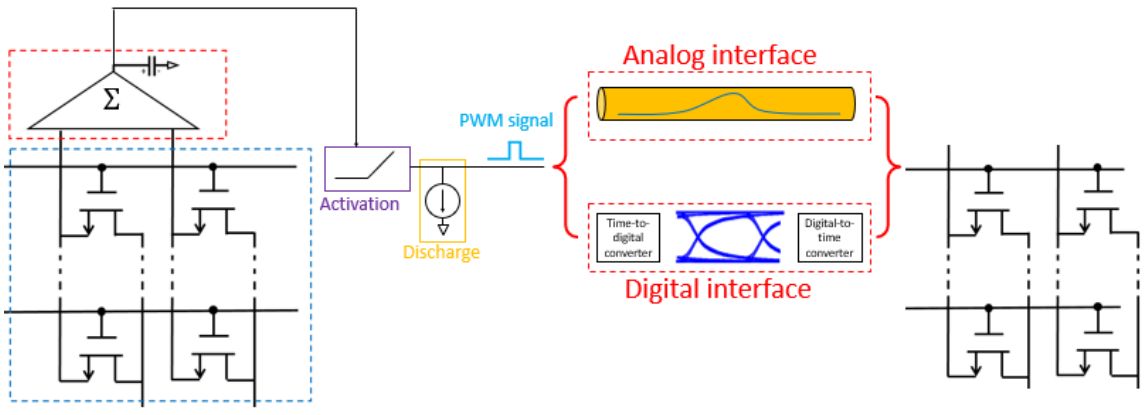


Fig. 5. 2 Interface between layers of the neural network can be implemented as either analog or digital interface between 2 chips for a multi-chip system.

Since the analog interface transmits the pulse width as the information, its speed is fundamentally limited by the maximum possible pulse width generated from the previous layer. The dynamic range of the pulse width determines the range of the dot-product outputs and therefore has a lower limit. As a result, multiplexing the analog communication in time by $n_{multiplex}$ times increases the communication latency linearly by the equation $\tau_{analog} = T_{PWM,max} * n_{multiplex}$, where $n_{multiplex} = ceil(\frac{BW_{analog}}{N_{channel}})$ is determined by the required bandwidth of the analog communication and the number of channels for both inputs and outputs of a layer.

For $n_{multiplex} = 1$, the number of outputs of each layer equals the number of channels required if the outputs go to another chip. The unrolled CNN can be partitioned into different dies/chips across the boundaries of the array (**Fig. 3. 13**). **Fig. 5. 3** shows the total number of inputs and outputs of each array for unrolled Alexnet.

As a result, a very high density of interconnect channels between chips is required to scale out the analog neuromorphic system with an analog interface. The dense physical channels can be combined with time-multiplexing (if required, but not preferred) to enable the analog interface for optimized energy efficiency of the system. When the time-multiplexing factor is properly balanced between layers, it can be hidden in the pipeline so that it does not affect the overall throughput of the system.

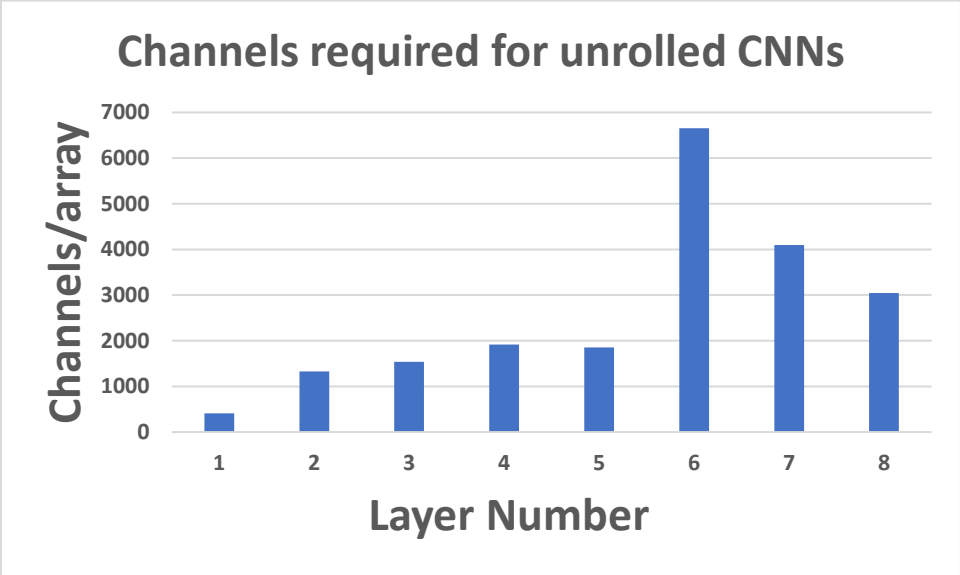


Fig. 5. 3 Number of channels (inputs and outputs) of the arrays in each layer of the Alexnet.

5.3 Fine-Pitch Integration Technologies for Analog Neuromorphic Systems

Conventional system scaling and integration technologies based on printed circuit boards (PCB) are not suitable for high-bandwidth analog transmission due to the coarse pitch at the chip/board interface. The pitch between pad and traces on the PCBs is normally more than 0.2mm and 0.1mm. To increase the aggregate bandwidth of analog transmission, fine-pitch integration technology, such as silicon interconnect fabric (Si-IF) [Bajwa 18], can be used to provide more physical channels between chips. As shown in **Fig. 5. 4**, known-good-dies can be placed in high proximity (< 100um) on a Si wafer substrate. In addition, the interconnect channels are fabricated using a conventional back-end-of-line (BEOL) processes to obtain interconnect pitch comparable to the fat-wire level (2 – 10 um) of the die itself. A schematic of two chips integrated within a single system using Si-IF is shown in **Fig. 5. 5**. Si-IF can be used to seamlessly integrate the different layers of the neural network by supporting analog communication with low insertion loss and crosstalk [SivaChandra 18] at < 10GHz. Comparing with the PCB, Si-IF can provide many

more channels for high-quality analog communication between chips. **Table 6** shows the number of channels supported by both PCB and Si-IF technology for different sizes of dies/chips. To implement the Alexnet (unrolled) system shown in Section 3.4.2. The requirement of the Alexnet (**Fig. 5. 3**) can be met by Si-IF but not PCB technology.

To further scale out the system, CTT-based inference circuits can be used as the basic building blocks for three-dimensional integrated circuits (3D-IC) [Xie 10]. For example, the limitation of Si-IF technology is the size of the wafer used for Si-IF fabrication. It is possible to add through-wafer vias (TWV) [Liu 19] to utilize both sides of the wafer, but the TWV pitch is limited by the aspect ratio of electroplating and therefore is about 0.05mm for un-thinned wafers whose thickness is more than 0.5mm. Three-dimensional integration technologies such as wafer-scale integration (3D-WSI) [Batra 14, Kumar 17] can be used, which can further improve the density of links by using through-silicon vias (TSVs) in the vertical direction, and energy-efficiency of communication by at least 100 times [Wan 17].

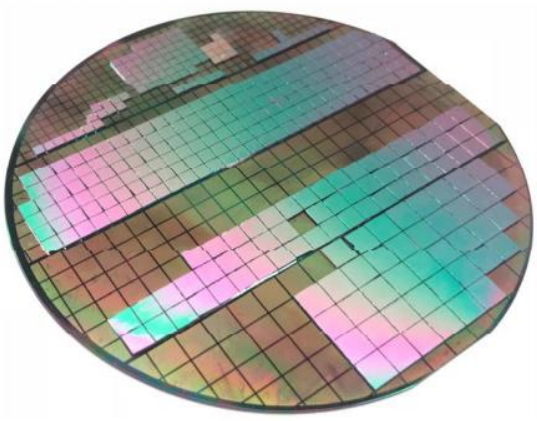


Fig. 5. 4 Scaling-out silicon on silicon: the silicon interconnect fabric (Si-IF) that can be used to scale-out analog neuromorphic systems with fine-pitch interconnect channels and close distance between the dies (adapted from [Bajwa 18]).

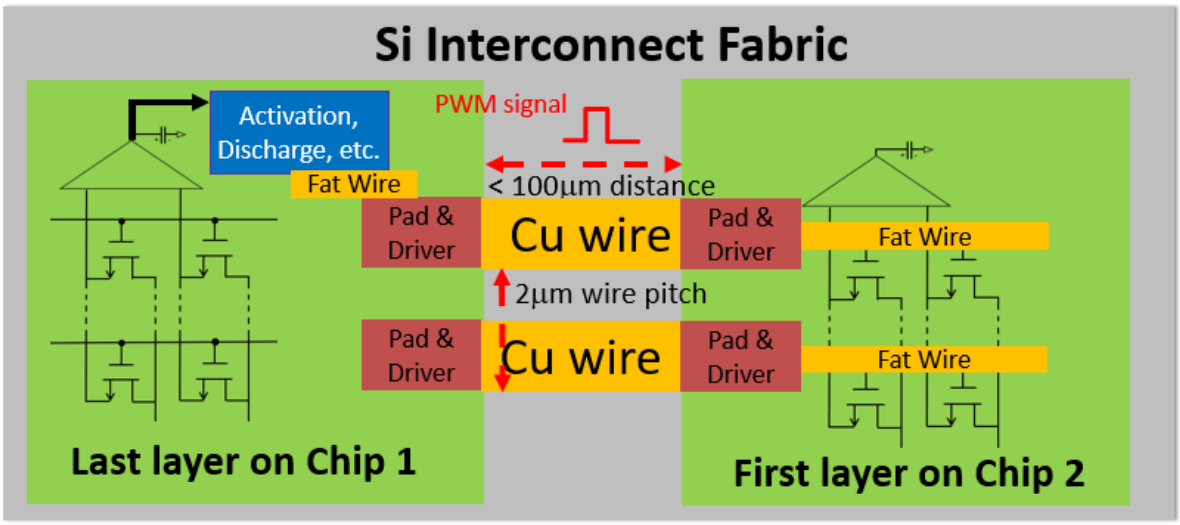


Fig. 5. 5 A schematic of the analog neuromorphic system scaled-out using the Si-IF for multi-layer neural networks.

Table 6 Number of links supported by PCB and Si-IF technology

	Number of links supported at periphery	
Die size	PCB (0.1mm pitch) 10 layers' periphery	Si-IF (0.002mm pitch) 4 layers' periphery
1mm * 1mm	400	8,000
3mm * 3mm	1200	24,000

Chapter 6: System Design and Hardware Demonstration

NeuroCTT Version 1 (V1) is designed as a mixed-signal system featuring digital controls on the twin-CTT cell array using GlobalFoundries 22FDX technology. The block diagram of the system is shown in **Fig. 6. 1**. A CTT-array with 1024 word-line (WL) and ten bit-line (BL) is the core of the system. The WL is driven by the WL driver (WLD), which receives digital control signals from a customized on-chip digital controller to apply a selected analog voltage to the WL for an adjustable amount of time, timed by the clock of the controller's signals. The digital controller is designed at the HDL level, then synthesized, placed, and routed (PnR) using CAD software (e.g., ModelSim, Innovus). The neuron is also controlled by the controller and can be used to integrate differential current from the BL pair (i.e., BLt and BLc) with the ReLU function.

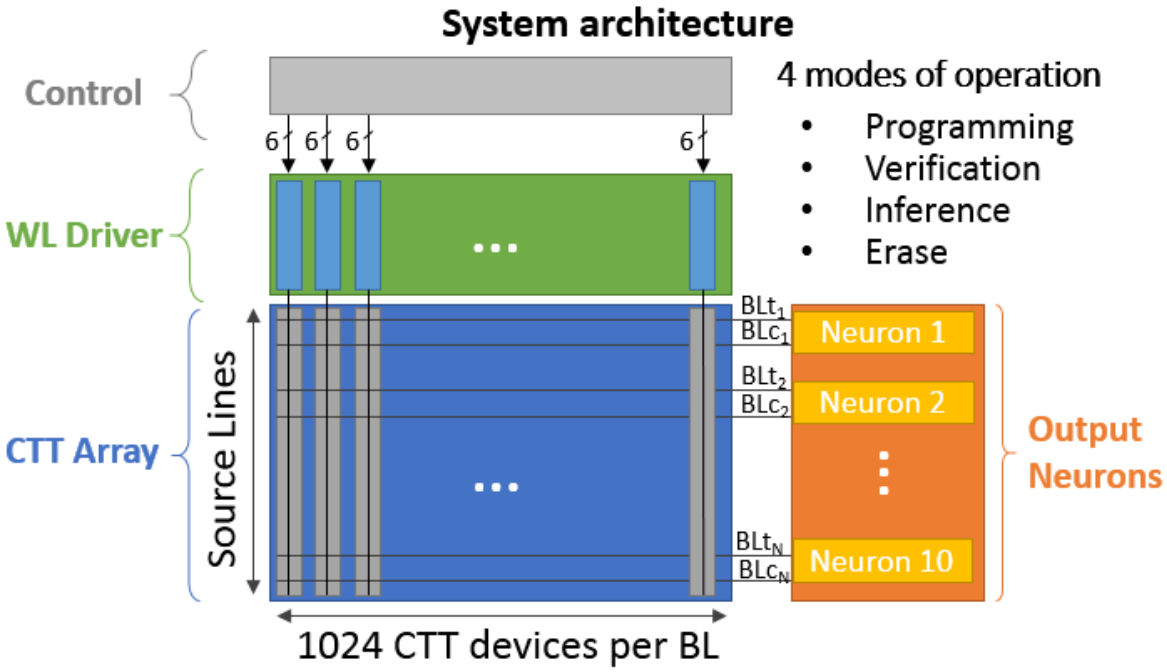


Fig. 6. 1 The system architecture of NeuroCTT V1. The system is composed of a CTT array with 1024 WLs and 10 BLs, 1024 WL drivers for each WL, neurons to integrate differential current from BLt and BLc, and a digital controller that provides all auxiliary functions.

The flow of instruction and data in the system is shown in **Fig. 6. 2**. In addition, the digital controller has a scan-chain to check the instruction received by the chip. The scan-chain is verified at 40MHz for array operation modes using the FPGA, as shown in **Fig. 6. 2**.

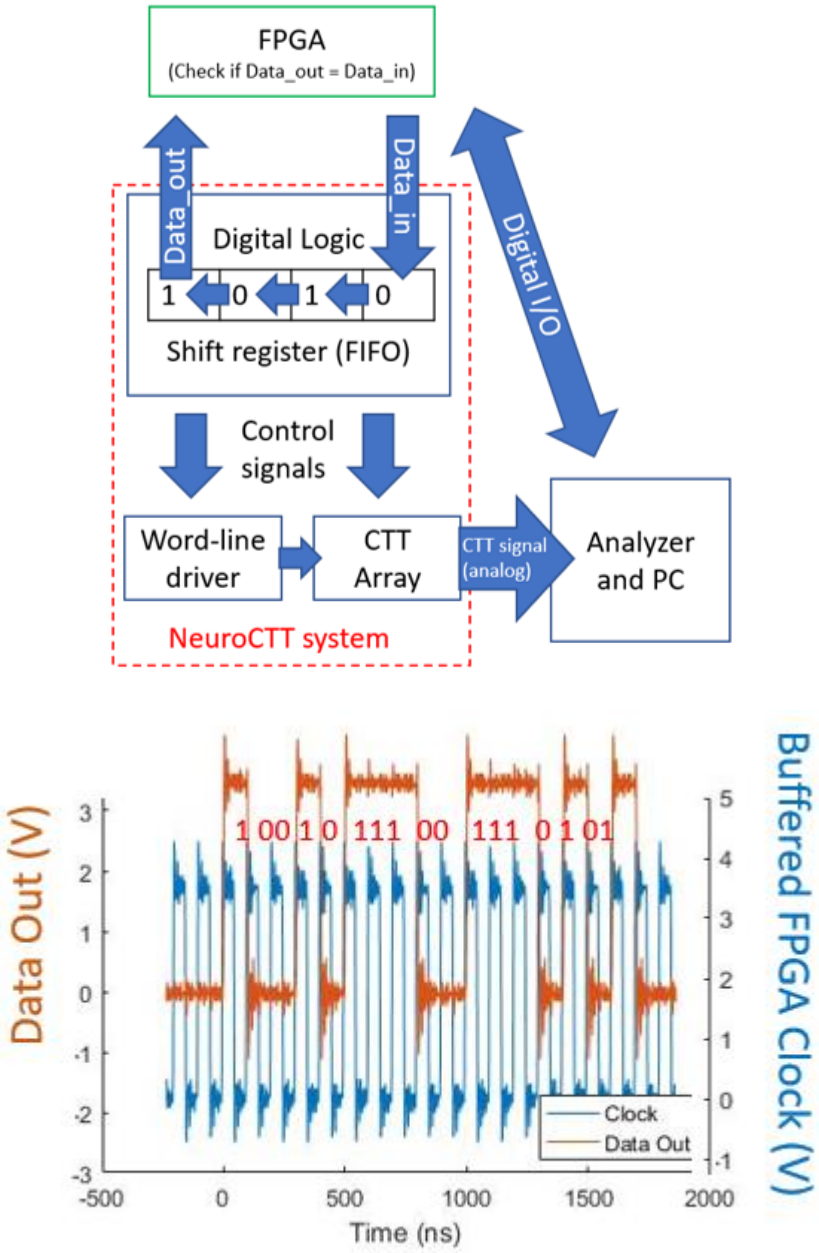


Fig. 6. 2 Top: the data flow in the V1 system and including using the FPGA to check the scan-chain of the V1 system. **Bottom:** measured waveforms with 10MHz clock I/O. The scan chain is verified up to 40MHz.

The structure of the array is shown in **Fig. 6. 3**. Each bit-line is composed of the bit-line true (BLt) and bit-line complement (BLc), connecting to the drains of the true device and the complement device of each twin-CTT cell in the chip. Other than the twin-CTT cells, the array

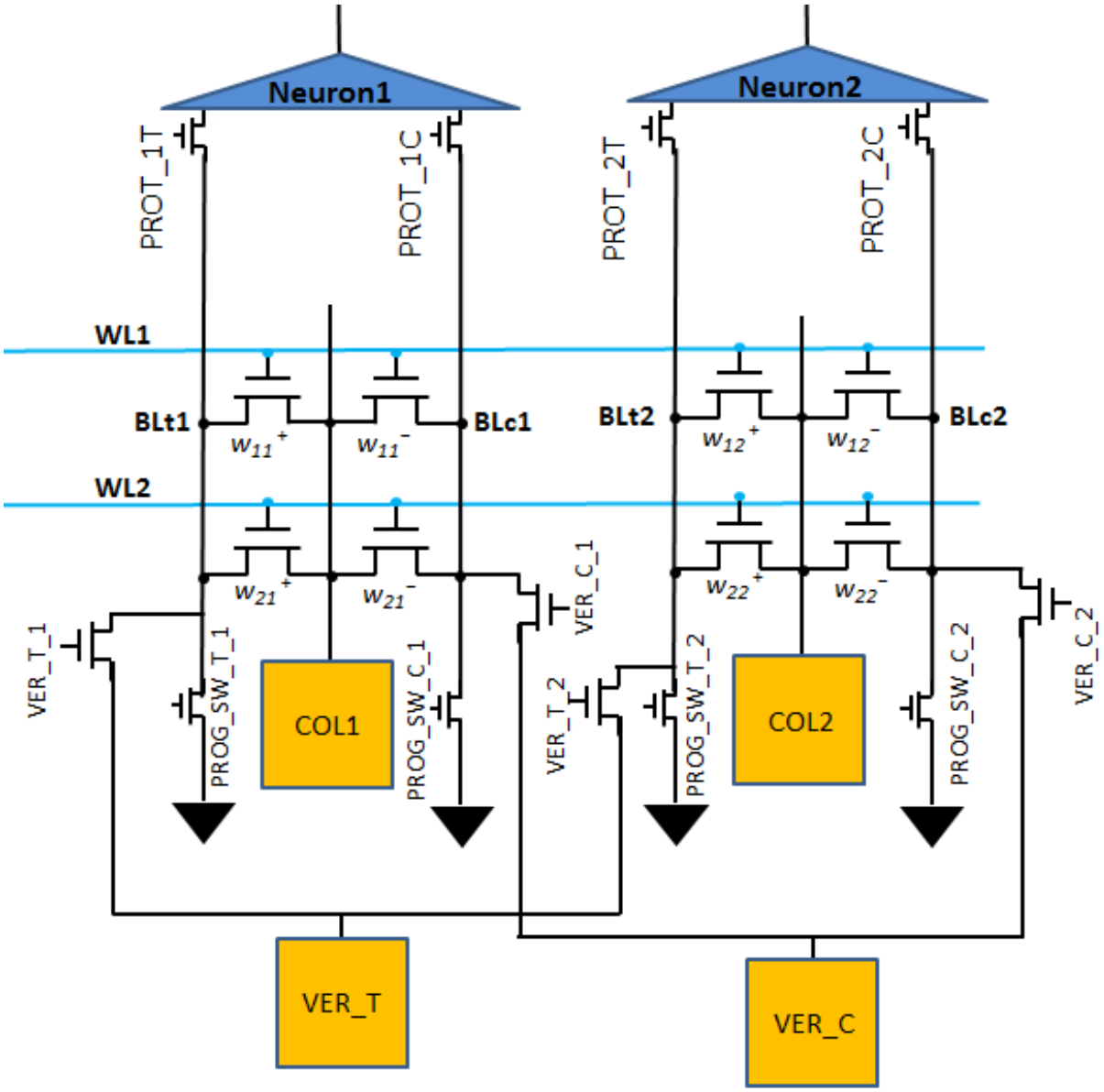


Fig. 6. 3 The array structure in V1. The array is 1024-by-10 but a 2-by-2 sub array is shown for simplicity. CTTs are arranged as twin-CTT cells (e.g., cell w_{11} has device w_{11}^+ and device w_{11}^-). The auxiliary array switches, PROT_xC, PROT_xT, VER_T_x, VER_C_x, PROG_SW_T_x, PROG_SW_C_x (x = 1,2,…,10). Are controlled by the digital controller to configure the array for different modes. Each source-line has a pad (COLx) and there are two more pads (VER_T, VER_C) for reading the CTT device current in the array.

also has auxiliary array switches that are directly controlled by the digital controller. The operation of a selected device being read (VER) is shown in **Fig. 6. 4**. Since the CTTs are used as analog

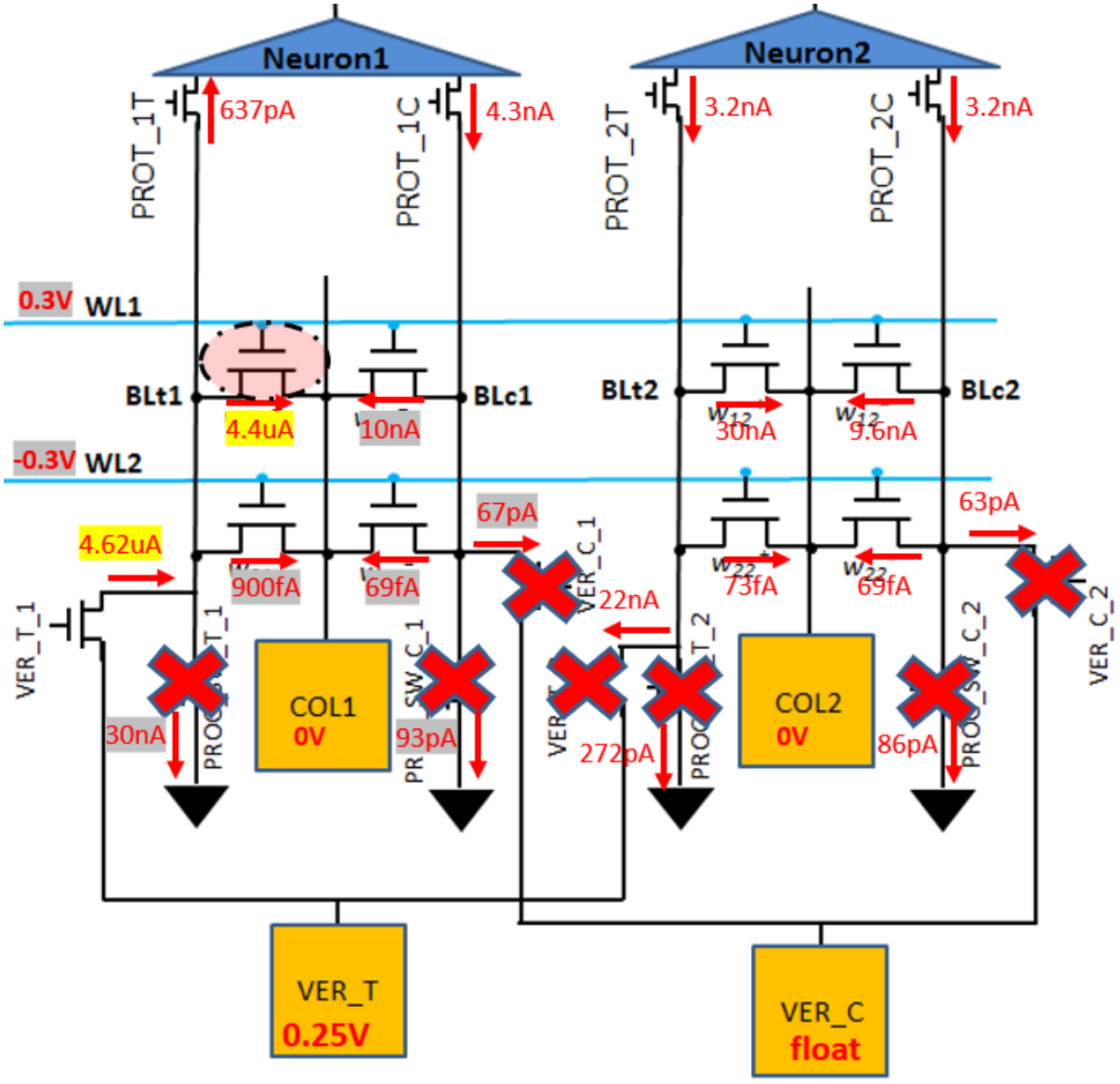


Fig. 6. 4 The read (VER) operation of the V1 chip. As the digital logic of the chip receives the instruction to read a certain device (e.g., circled CTT W_{11}^{+}), it configures the array auxiliary switches (the red cross represents that the array switch is turned off by the control logic for this VER operation) on the peripheral of the array and the word-line drivers to supply a read voltage (300mV) to the target word-line, and a negative bias (-0.3V) to all other word-lines. An analyzer is then connected to the pad of the target column and the VER (VER_T or VER_C) pad to supply a read V_{DS} and measure the current I_{VER}. The current numbers on the devices are the simulated result using spectre models.

memory, the precision of the read operation is important, the read current, when one device is turned on, can be written as

$$I_{READ} = I_{on,target} + 1023\, I_{off} + I_{leakage}$$

where I_{off} is the current from other CTTs in the same column, whose gate voltage is negative (i.e., at off state). $I_{leakage}$ is the current through other leaking paths. Although the off current is much smaller in magnitude compared with the on current ($I_{off} < 10^{-4} I_{on}$), the on current will become smaller when the device is programmed (i.e.,I_{on} decreases), decreasing the distinction of I_{on} from I_{READ}. Therefore, baseline subtraction is needed for accurate reading of the device where

$$I_{baseline} = 1024\, I_{off} + I_{leakage}$$

$$I_{READ} - I_{baseline} = I_{on,target} - I_{off,target}$$

Fig. 6. 5 shows the read current statistics of all CTT devices. The limitation of the reading is the throughput since V1 only supports the reading of one device at a time. The analyzer requires

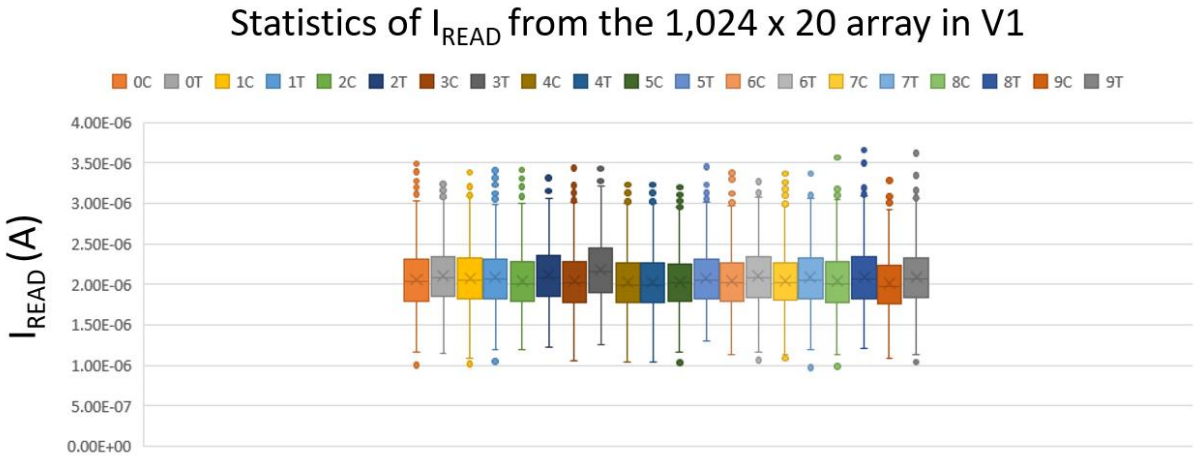


Fig. 6. 5 The box and whisker plot of I_{READ} read from a V1 chip for the 1,024 x 20 array. Due to the use of twin-cell CTT, each column (e.g., Column0 to Column9) is consist of the "true" line (e.g., 0T) and the complement line (e.g., 0C), which is measured separately during the read operation.

about 0.3s for an accurate reading of the current. Therefore, on-chip reading circuit and parallel reading of the CTT current are required to enable large-scale CTT-based system.

The operation of a selected device being programmed (PRG) is shown in **Fig. 6. 6**. The result of on-chip CTT PRG is shown in **Fig. 6. 7**, where two measurements are taken before and after PRG events to show the shift in the V_{th} of the CTTs. Different PRG conditions are tested as shown in **Table 7** and have achieved a maximum ΔV_{th} of 60mV. Half-selected devices are slightly

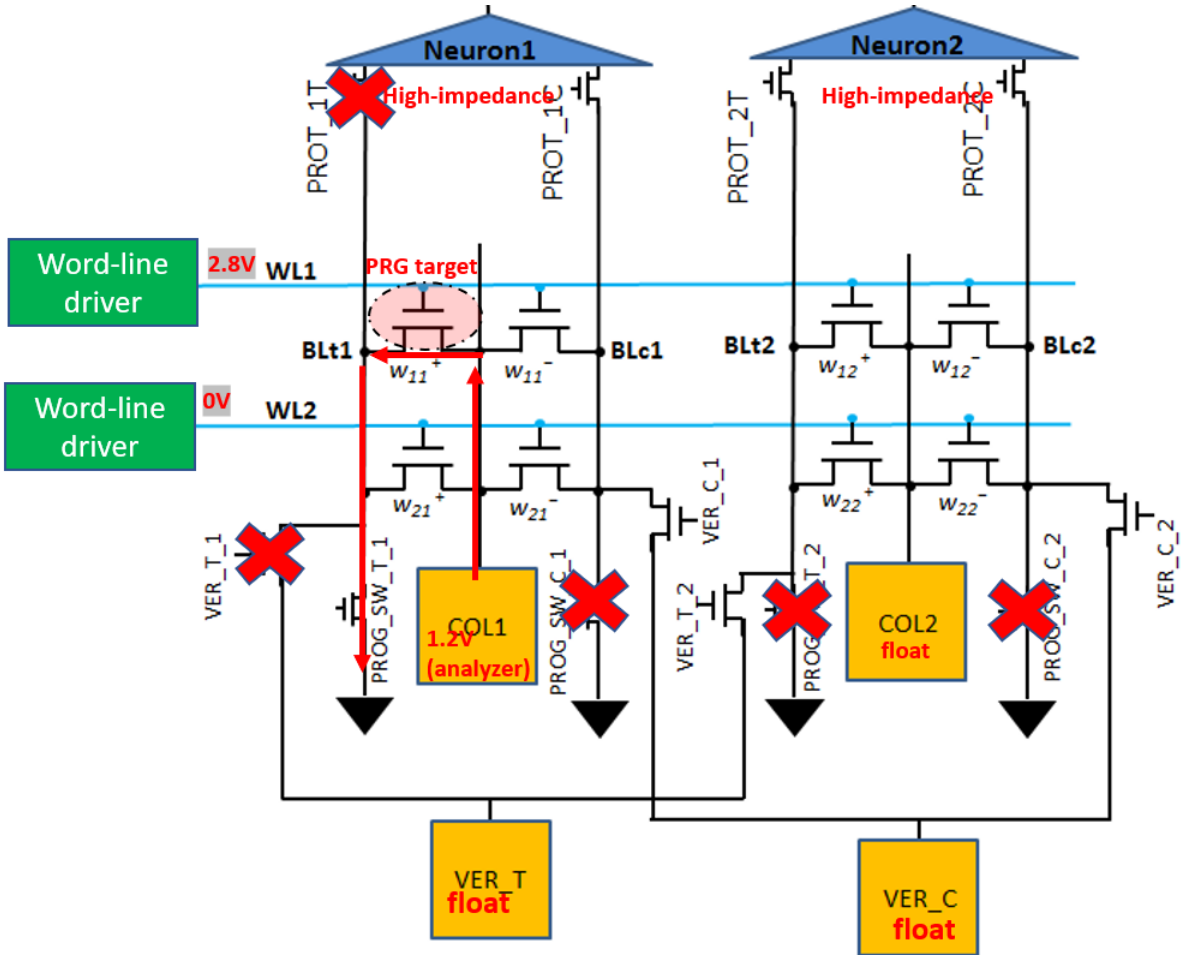


Fig. 6. 6 The PRG operation of the V1 chip. Comparing with the VER operation shown in **Fig. 6. 4**, the PRG operation of same target device has a different auxiliary array switch configuration, where the source-line will take a programming pulse (e.g., 1.2V), synchronized with a gate pulse from the word-line driver of 2.8V on the target device. The bit-line of the target device is brought down to ground by the programming switch (*e.g.*, PRG_SW_T_1).

affected in the same WLs, but not in the same BL. Raising the SL voltage before the WL is important to suppress the half-selection effect on the WL-selected devices.

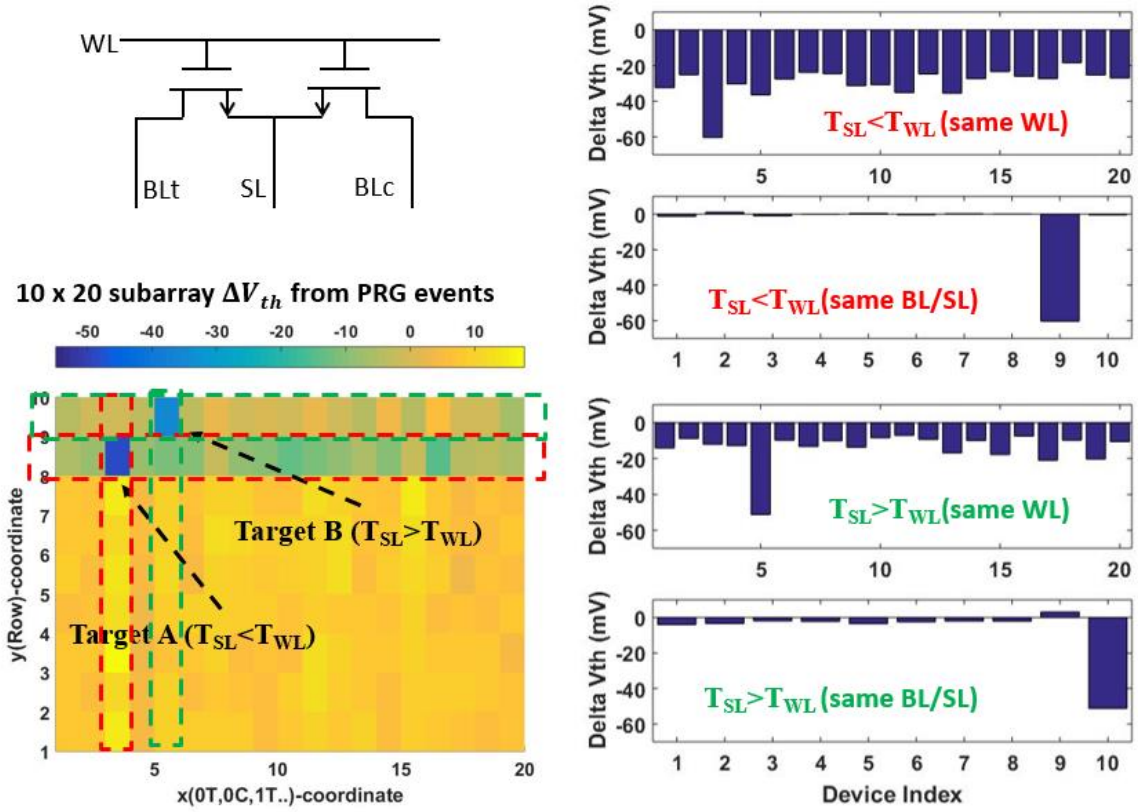


Fig. 6. 7 A sub-array of 10-by-20 is measured before and after PRG events to see the change of V_{th} of the devices. A maximum of 60mV V_{th} shift is achieved. Two conditions of PRG are shown, one (Target A) with the WL pulse slightly wider than the SL pulse (T_{SL}<T_{WL}), the other one (Target B) with the SL pulse slightly wider than the WL pulse (T_{SL}>T_{WL}). Half-select along column (BL/SL) is not significant in both cases. However, PRG using wider SL alleviates half-selection along WL.

Table 7 Different PRG conditions tested on the V1 chip, the effect of the PRG events are shown in **Fig. 6. 7**.

PRG Conditions	**PRG A**	**PRG B**
WL pulse (10ms)	2.8V	2.8V
SL voltage (10ms)	1.4V	1.4V
Pulse sequence	WL pulse rises first and falls later	SL pulse rises first and falls later

Chapter 7: Conclusions and Outlooks

7.1 Conclusions

This dissertation provides a comprehensive study on the use of an analog non-volatile memory device, the charge-trap transistor (CTT), for analog in-memory computing, specifically for neuromorphic computing applications. The investigation involves a wide range of topics, including the analog memory device, the analog in-memory computing architecture for the neural network algorithm, and the feasibility of using the proposed architecture for scaled neural network applications (36 million synaptic weights).

In Chapter 2, the CTTs are characterized as analog memories using their subthreshold current I_{inf} as stored data, a real physical quantity that can be used later for direct computation in the analog domain. Different mechanisms of charge-trapping are discussed and compared. For bulk-oxide traps, the relaxation of the device after charge-trapping is spontaneous de-trapping, which can be detected as an increase in I_{inf}. This relaxation at room temperature can be compensated by offsetting the programming target. Different models should be used for relaxation at elevated temperature as it accelerates the charge loss significantly. Fine step charge trapping and de-trapping of CTTs can be achieved by using finely increased voltage magnitude for V_{GS} to access traps at different energy levels. Combined with the relaxation compensation, the CTTs can be programmed accurately with a Gaussian model for the error.

In Chapter 3, the analog in-memory computing architecture for vector-matrix multiplication using natural laws is investigated. Using CMOS peripheral circuits, multi-layer perceptron (MLP) and convolutional neural networks (CNN) can be implemented. To build the

system for deep neural networks such as AlexNet, scaling out the hardware is required for unrolled AlexNet.

Chapter 4 and Chapter 5 evaluate the feasibility of the neural network algorithm and the system integration technology for scaling out the analog system. The resiliency of the neural networks against analog computing errors is assessed and improved by the proposed Hessian-aware training algorithm for very large and deep neural networks. Novel hardware integration technology, such as the Si-interconnect fabric, can support the massive analog I/O channels required by the system.

Finally, hardware design for the CTT-based inference engine is presented and discussed in Chapter 6. The use of CTTs as a tunable analog memory is demonstrated in a mixed-signal system-on-chip (SoC) with 20 thousand CTTs in GlobalFoundries 22FDX technology.

In conclusion, this dissertation integrated the studies on (1) the CTT as an analog device, (2) the analog in-memory computing architecture, (3) the neural network optimization for analog computing, and (4) the approaches to scaling out analog computing system. The results show the feasibility of scalable and analog neuromorphic computing systems from the device level all the way to the application level.

7.2 Outlook

In order to realize and optimize the proposed system. Four major directions for the future are suggested below:

1. Physical modeling of CTT devices: all CTT results shown in the dissertation are from experiments directly. While some curve fitting can be used to optimize the use of CTT as analog memory. Physical modeling can be more useful to provide insights to device design

and optimization principles. Although it has been explored for self-heating enhanced charge-trapping [Khan 15], it also needs to be explored for the associated data retention. For example, the expected charge loss with respect to temperature and time when different programming conditions are used (e.g., the PVRS method), so that traps at different energy levels are involved. This helps to improve the modeling of the device for different application scenarios.

2. The figure of merit for peripheral circuits in the analog computing engine: the figure of merit for the peripheral circuits, such as the current integrator, needs to be studied for the analog neural network inference engine. For example, it has been shown that if the integrated values are quantized non-linearly by the circuit, it will perform better than linear quantization for the analog neural networks. Therefore, research on a proper figure of merits can guide the design of more suitable peripheral circuits.
3. Efficient programming for large-scale inference engine: for analog memory with low write speed such as the CTTs, efficient programming in both energy and time will be crucial to pave the way for the low-cost deployment of the inference engine. For example, an algorithm and supporting architecture to quickly converge to the target values for the devices in batch mode, which is not discussed in this dissertation, can be developed.
4. System reconfigurability: while the proposed system (with unrolling for CNN) does not require explicit instruction during inference. It also limits the reconfigurability of the system. Therefore, a tiled design should be considered to provide flexibility at the chip-package level, or explored in the SoC if it does not require significant overhead.

References

[Auth 08] Auth, Chris, et al. "45nm high-k+ metal gate strain-enhanced transistors." 2008 Symposium on VLSI Technology. IEEE, 2008.

[Backus 78] Backus, John. "Can programming be liberated from the von Neumann style? A functional style and its algebra of programs." Communications of the ACM 21.8 (1978): 613-641.

[Bajwa 18] Bajwa, Adeel Ahmad, et al. "Demonstration of a Heterogeneously Integrated System-on-Wafer (SoW) Assembly." 2018 IEEE 68th Electronic Components and Technology Conference (ECTC). IEEE, 2018.

[Batra 14] Batra, Pooja, et al. "Three-dimensional wafer stacking using Cu TSV integrated with 45 nm high performance SOI-CMOS embedded DRAM technology." Journal of Low Power Electronics and Applications 4.2 (2014): 77-89.

[Benjamin 14] Benjamin, Ben Varkey, et al. "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations." Proceedings of the IEEE 102.5 (2014): 699-716.

[Burr 14] Burr, Geoffrey W., et al. "Phase change memory technology." Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena 28.2 (2010): 223-262.

[Burr 15] Burr, Geoffrey W., et al. "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element." IEEE Transactions on Electron Devices 62.11 (2015): 3498-3507.

[Burr 16] Burr, Geoffrey W., et al. "Recent progress in phase-change memory technology." IEEE Journal on Emerging and Selected Topics in Circuits and Systems 6.2 (2016): 146-162.

[Cai 19] Cai, Fuxi, et al. "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations." Nature Electronics (2019):

[Canziani 16] Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." arXiv preprint arXiv:1605.07678 (2016).

[Carter 16] Carter, R., et al. "22nm FDSOI technology for emerging mobile, Internet-of-Things, and RF applications." 2016 IEEE International Electron Devices Meeting (IEDM). IEEE, 2016.

[Cartier 06] Cartier, E., et al. "Fundamental understanding and optimization of PBTI in nFETs with SiO2/HfO2 gate stack." 2006 International Electron Devices Meeting. IEEE, 2006.

[Cartier 09] Cartier, Eduard, and Andreas Kerber. "Stress-induced leakage current and defect generation in nFETs with HfO 2/TiN gate stacks during positive-bias temperature stress." 2009 IEEE International Reliability Physics Symposium. IEEE, 2009.

[Chen 17] Chen, Yu-Hsin, et al. "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks." IEEE Journal of Solid-State Circuits 52.1 (2017): 127-138.

[Chen 18] Chen, Pai-Yu, Xiaochen Peng, and Shimeng Yu. "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 37.12 (2018): 3067-3080.

[Davies 18] Davies, Mike, et al. "Loihi: A neuromorphic manycore processor with on-chip learning." IEEE Micro 38.1 (2018): 82-99.

[Deng 09] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009.

[Desoli 17] Desoli, Giuseppe, et al. "14.1 a 2.9 tops/w deep convolutional neural network soc in fd-soi 28nm for intelligent embedded systems." Solid-State Circuits Conference (ISSCC), 2017 IEEE International. IEEE, 2017.

[DiMaria 95] DiMaria, D. J., and E. Cartier. "Mechanism for stress-induced leakage currents in thin silicon dioxide films." Journal of Applied physics 78.6 (1995): 3883-3894.

[Drubach 00] Drubach, Daniel. The Brain Explained. New Jersey: Prentice-Hall, 2000.

[Ghani 00] Ghani, T., et al. "Scaling challenges and device design requirements for high performance sub-50 nm gate length planar CMOS transistors." 2000 Symposium on VLSI Technology. Digest of Technical Papers (Cat. No. 00CH37104). IEEE, 2000.

[Goodfellow 14] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." arXiv preprint arXiv:1412.6572 (2014).

[Gu 18] Gu, Xuefeng. Charge-trap transistors for neurmorphic computing. Diss. UCLA, 2018.

[Gu 19] Gu, Xuefeng, Zhe Wan, and Subramanian S. Iyer. "Charge-Trap Transistors for CMOS-Only Analog Memory." IEEE Transactions on Electron Devices 66.10 (2019): 4183-4187.

[Guo 17] Guo, Xinjie, 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). IEEE, 2017.

[Hawkins 04] Hawkins, Douglas M. "The problem of overfitting." Journal of chemical information and computer sciences 44.1 (2004): 1-12.

[He 16] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[Hu 85] Hu, Chenming, et al. "Hot-electron-induced MOSFET degradation-model, monitor, and improvement." IEEE Journal of Solid-State Circuits 20.1 (1985): 295-305.

[Huijsing 13] Huijsing, Johan, Rudy J. van de Plassche, and Willy MC Sansen, eds. Analog Circuit Design: Volt Electronics; Mixed-Mode Systems; Low-Noise and RF Power Amplifiers for Telecommunication. Springer Science & Business Media, 2013.

[Indiveri 11] Indiveri, Giacomo, et al. "Neuromorphic silicon neuron circuits." Frontiers in neuroscience 5 (2011): 73.

[Ioffe 15] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[Jain 19] Jain, Pulkit, et al. "13.2 A 3.6 Mb 10.1 Mb/mm 2 Embedded Non-Volatile ReRAM Macro in 22nm FinFET Technology with Adaptive Forming/Set/Reset Schemes Yielding Down to 0.5 V with Sensing Time of 5ns at 0.7 V." 2019 IEEE International Solid-State Circuits Conference-(ISSCC). IEEE, 2019.

[Jeppson 77] Jeppson, Kjell O., and Christer M. Svensson. "Negative bias stress of MOS devices at high electric fields and degradation of MNOS devices." Journal of Applied Physics 48.5 (1977): 2004-2014.

[Jouppi 17] Jouppi, Norman P., et al. "In-datacenter performance analysis of a tensor processing unit." Proceedings of the 44th Annual International Symposium on Computer Architecture. 2017.

[Kerber 09] Kerber, Andreas, Siddarth A. Krishnan, and Eduard Albert Cartier. "Voltage Ramp Stress for Bias Temperature Instability Testing of Metal-Gate/High-k Stacks." IEEE Electron Device Letters 30.12 (2009): 1347-1349.

[Keysight 18]B1500A Semiconductor Device Analyzer Data Sheet. Keysight. Dec. 2018.

[Khan 15] Khan, Faraz, et al. "The impact of self-heating on charge trapping in high-k-metal-gate nFETs." IEEE Electron Device Letters 37.1 (2015): 88-91.

[Khan 16] Khan, Faraz, et al. "Charge Trap Transistor (CTT): An Embedded Fully Logic-Compatible Multiple-Time Programmable Non-Volatile Memory Element for High-k-Metal-Gate CMOS Technologies." IEEE Electron Device Letters 38.1 (2016): 44-47.

[Khan 19] Khan, Faraz, et al. "Turning Logic Transistors into Secure, Multi-Time Programmable, Embedded Non-Volatile Memory Elements for 14 nm FINFET Technologies and Beyond," 2019 IEEE Symposium on VLSI Technology (VLSI-Technology). IEEE, 2019.

[Kim 11] Kim, Y., et al. "Integration of 28nm MJT for 8~ 16Gb level MRAM with full investigation of thermal stability." 2011 Symposium on VLSI Technology-Digest of Technical Papers. IEEE, 2011.

[Kim15] Kim, Yong-Deok, et al. "Compression of deep convolutional neural networks for fast and low power mobile applications." arXiv preprint arXiv:1511.06530 (2015).

[Kittur 20] Kittur, Premsagar. Private communications

[Koch 96] Koch, Christof, and Bimal Mathur. "Neuromorphic vision chips." Ieee Spectrum 33.5 (1996): 38-46.

[Krizhevsky 02] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[Krizhevsky 09] Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009): 7.

[Kumar 17] Kumar, Arvind, et al. "Toward human-scale brain computing using 3D wafer scale integration." ACM Journal on Emerging Technologies in Computing Systems (JETC) 13.3 (2017): 45.

[LeCun 15] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521.7553 (2015): 436-444.

[LeCun 19] LeCun, Yann. "Deep Learning Hardware: Past, Present, and Future." 2019 IEEE International Solid-State Circuits Conference-(ISSCC). IEEE, 2019.

[LeCun 89] LeCun, Yann, et al. "Backpropagation applied to handwritten zip code recognition." Neural computation 1.4 (1989): 541-551.

[LeCun 98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.

[Liu 19] Liu, Meng-Hsiang, et al. "Process development of power delivery through wafer vias for silicon interconnect fabric." 2019 IEEE 69th Electronic Components and Technology Conference (ECTC). IEEE, 2019.

[Ma 18] Ma, Wen, et al. "Improving Noise Tolerance of Hardware Accelerated Artificial Neural Networks." 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2018.

[Ma 19] Ma, Siming, et al. "Fully-CMOS Multi-Level Embedded Non-Volatile Memory Devices With Reliable Long-Term Retention for Efficient Storage of Neural Network Weights." IEEE Electron Device Letters 40.9 (2019): 1403-1406.

[McCulloch 43] McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." The bulletin of mathematical biophysics 5.4 (1943): 115-133.

[Mead 89] Mead, Carver. "Analog VLSI and neutral systems." NASA STI/Recon Technical Report A 90 (1989).

[Merolla 14] Merolla, Paul A., et al. "A million spiking-neuron integrated circuit with a scalable communication network and interface." Science 345.6197 (2014): 668-673.

[Moran 20] Moran, Steven. Private communications

[Nvidia 15] Inference, GPU-Based Deep Learning. "A Performance and Power Analysis." Nvidia Whitepaper, November (2015).

[Pobegen 13] Pobegen, Gregor, et al. "Observation of normally distributed energies for interface trap recovery after hot-carrier degradation." IEEE electron device letters 34.8 (2013): 939-941.

[Rastegari 16] Rastegari, Mohammad, et al. "Xnor-net: Imagenet classification using binary convolutional neural networks." European conference on computer vision. Springer, Cham, 2016.

[Razavi 02] Razavi, Behzad. Design of analog CMOS integrated circuits. Tata McGraw-Hill, 2002

[Reagen 18] 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.

[Real 17] Real, Esteban, et al. "Large-scale evolution of image classifiers." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.

[Rosenblatt 58] Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." Psychological review 65.6 (1958): 386.

[Russakovsky 15] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." International journal of computer vision 115.3 (2015): 211-252.

[Schemmel 10] Schemmel, Johannes, et al. "A wafer-scale neuromorphic hardware system for large-scale neural modeling." Proceedings of 2010 IEEE International Symposium on Circuits and Systems. IEEE, 2010.

[Schölkopf 02] Schölkopf, Bernhard, Alexander J. Smola, and Francis Bach. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2002.

[Sengupta 16] Sengupta, Abhronil, et al. "Probabilistic deep spiking neural systems enabled by magnetic tunnel junction." IEEE Transactions on Electron Devices 63.7 (2016): 2963-2970.

[Seo 11] Seo, Kyungah, et al. "Analog memory and spike-timing-dependent plasticity characteristics of a nanoscale titanium oxide bilayer resistive switching device." Nanotechnology 22.25 (2011): 254023.

[Shafiee 16] Shafiee, Ali, et al. "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars." ACM SIGARCH Computer Architecture News 44.3 (2016): 14-26.

[Shihab 16] Shihab, Mustafa, et al. "Couture: Tailoring stt-mram for persistent main memory." 4th Workshop on Interactions of NVM/Flash with Operating Systems and Workloads ({INFLOW} 16). 2016.

[Silver 07] Silver, Rae, et al. "Neurotech for neuroscience: unifying concepts, organizing principles, and emerging tools." Journal of Neuroscience 27.44 (2007): 11807-11819.

[Silver 17] Silver, David, et al. "Mastering the game of go without human knowledge." Nature 550.7676 (2017): 354-359.

[SivaChandra 18] SivaChandra, et al. "Electrical Characterization of High Performance Fine Pitch Interconnects in Silicon-Interconnect Fabric." 2018 IEEE 68th Electronic Components and Technology Conference (ECTC). IEEE, 2018.

[Song 19] Song, Jinook, et al. "7.1 An 11.5 TOPS/W 1024-MAC butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile SoC." 2019 IEEE International Solid-State Circuits Conference-(ISSCC). IEEE, 2019.

[Srivastava 14] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.

[Sun 18] Sun, Xiaoyu, et al. "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks." 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2018.

[Valov 11] Valov, Ilia, et al. "Electrochemical metallization memories—fundamentals, applications, prospects." Nanotechnology 22.25 (2011): 254003.

[Vincent 15] Vincent, Adrien F., et al. "Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems." IEEE transactions on biomedical circuits and systems 9.2 (2015): 166-174.

[Viraraghavan 16] Viraraghavan, Janakiraman, 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). IEEE, 2016.

[Wan 17] Wan, Zhe, and Subramanian S. Iyer. "Three-dimensional wafer scale integration for ultra-large-scale cognitive systems." 2017 IEEE SOI-3D-Sub-threshold Microelectronics Technology Unified Conference (S3S). IEEE, 2017.

[Wen 18] Wen, Chenxi, and Timothy K. Horiuchi. "Power-Law Compression Expands the Dynamic Range of a Neuromorphic Echolocation System." 2018 IEEE Biomedical Circuits and Systems Conference (BioCAS). IEEE, 2018.

[Williams 09] Williams, Samuel. "Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore.", ACM Communications (2009).

[Wong 12] Wong, H-S. Philip, et al. "Metal–oxide RRAM." Proceedings of the IEEE 100.6 (2012): 1951-1970.

[Xie 10] Xie, Yuan, Jason Cong, and Sachin Sapatnekar. "Three-dimensional integrated circuit design." EDA, Design and Microarchitectures, New York: Springer 20 (2010): 194-196.

[Yuan 20] Yuan, Zhe et al. "A 65nm 24.7μJ/Frame 12.3mW Activation-Similarity-Aware Convolutional Neural Network Video Processor Using Hybrid Precision, Inter-Frame Data Reuse and Mixed-Bit-Width Difference-Frame Data Codec." 2020 IEEE International Solid-State Circuits Conference-(ISSCC). IEEE, 2020.

[Yue 19] Yue, Jinshan, et al. "7.5 A 65nm 0.39-to-140.3 TOPS/W 1-to-12b Unified Neural Network Processor Using Block-Circulant-Enabled Transpose-Domain Acceleration with 8.1× Higher TOPS/mm 2 and 6T HBST-TRAM-Based 2D Data-Reuse Architecture." 2019 IEEE International Solid-State Circuits Conference-(ISSCC). IEEE, 2019.

[Zagoruyko 16] Zagoruyko, Sergey, and Nikos Komodakis. "Wide residual networks." arXiv preprint arXiv:1605.07146 (2016).

[Zhang 15] Zhang, Chen, et al. "Optimizing fpga-based accelerator design for deep convolutional neural networks." Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2015.

[Zhang 16] Zhang, Chen, et al. "Energy-efficient CNN implementation on a deeply pipelined FPGA cluster." Proceedings of the 2016 International Symposium on Low Power Electronics and Design. 2016.

[Zhang 18] Zhang, Chen, et al. "Caffeine: Toward uniformed representation and acceleration for deep convolutional neural networks." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 38.11 (2018): 2072-2085.

[Zheng 18] Zheng, Xin, et al. "Error-Resilient Analog Image Storage and Compression with Analog-Valued RRAM Arrays: An Adaptive Joint Source-Channel Coding Approach." 2018 IEEE International Electron Devices Meeting (IEDM). IEEE, 2018.

[Zhou 14] Zhou, Jiantao, Kuk-Hwan Kim, and Wei Lu. "Crossbar RRAM arrays: Selector device requirements during read operation." IEEE Transactions on Electron Devices 61.5 (2014): 1369-1376.