UNIVERSITY OF CALIFORNIA

Los Angeles

Analog In-Memory Multiply-and-Accumulate Engine Fabricated in 22nm FDSOI Technology

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical & Computer Engineering

by

Steven Moran

2022

ABSTRACT OF THE DISSERTATION

Analog In-Memory Multiply-and-Accumulate Engine
Fabricated in 22nm FDSOI Technology

by

Steven Moran
Doctor of Philosophy in Electrical & Computer Engineering
University of California, Los Angeles, 2022
Professor Subramanian Srikanteswara Iyer, Chair

This dissertation presents the first on-chip demonstration of a Multiply-and-Accumulate (MAC) function in 22nm CMOS on SOI with the Charge-Trap Transistor (CTT).

Recent developments in machine learning and AI focus on digital-based von Neumann architectures to accelerate computation using massively parallel processing platforms including Graphics Processing Units (GPUs), Tensor Processing Units (TPUs), and Application-Specific Integrated Circuits (ASICs), to name a few. While these platforms have dramatically improved system performance, they are inherently limited by the von Neumann memory bottleneck. A resurgence of digital and analog in-memory & near-memory computing (iMC) techniques have been proposed to perform computation directly where the memory is stored, eliminating unnecessary memory accesses and minimizing memory access energy.

A hybrid approach to designing high-performance AI computation platforms is composed of learning on the cloud and energy-efficient inference at the edge. In this dissertation, we explore the latter through the use of the Charge-Trap Transistor (CTT)—commercial high-κ logic nFET device on SOI—as an ideal candidate nonvolatile memory device for analog-based

in-memory computing. Past results show that the CTT can be accurately programmed with excellent resolution, device programming variance, and retention characteristics. We propose a NeuroCTT inference architecture and present experimental results based on two test chips taped-out utilizing GlobalFoundries 22FDX technology. A first-time demonstration of an *on-chip* analog MAC Engine using the CTT in a commercial CMOS technology is provided. Accurate on-chip weight programming with sufficient retention are also demonstrated in hardware. In addition, we introduce a CTT-Hardware-based Inference Realistic Circuit Universal Simulator (CIRCUS) Platform for studying the effects of circuit-induced errors and device non-idealities on system performance and accuracy.

We conclude by evaluating the resiliency of general-purpose neural network applications by evaluating the effect of weight programming variance on analog-based in-memory computing and bit errors on digital-based architectures. As a baseline for digital-based & energy-efficient ASICs, an IBM TrueNorth Neurosynaptic System is exposed to 4MeV protons corrupting the on-chip model file for a trained 12-layer Convolutional Neural Network (CNN). The IBM TrueNorth continues to perform classification with negligible degradation to accuracy. For larger-scale networks and memory-intensive applications, reliability studies were also performed on 3D-stacked (3DS) DRAM to study the effect of radiation on more advanced 3D-stacked architectures.

The dissertation of Steven Moran is approved.

Achuta Kadambi

C. K. Ken Yang

Sudhakar Pamarti

Subramanian Srikanteswara Iyer, Committee Chair

University of California, Los Angeles

2022

To my partner Charlene, my parents Diane & Tim, and my siblings Chelsea & Michelle ...

thank you for your endless love, encouragement, guidance,

and unconditional support during my graduate studies.

This degree is as much yours as it is mine.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to first acknowledge my advisor, Prof. Subramanian (Subu) Iyer, for his support and guidance. I never considered pursuing a PhD until I took Subu's ECE 121B (Semiconductor Physics Course) during my undergrad. I met Subu during an office hour the day the UCLA PhD application was due, and he convinced me to join his lab & pursue a PhD. Despite all of the hardships along the way, I've learned quite a lot from Subu and I'm incredibly grateful for all of the opportunities he has provided to me.

Thank you to my committee members Prof. Sudhakar Pamarti, Prof. C. K. Ken Yang, and Prof. Achuta Kadambi for all of your questions, insights, and contributions to my work. I'm especially grateful to my co-advisor Prof. Sudhakar Pamarti for all of his chip design & testing guidance as well as being available for countless design reviews.

I would like to acknowledge many of peers in the CHIPS Lab. Thank you to Kyle Jung who during his tenure in the CHIPS Lab, was single-handily responsible for keeping the lab running efficiently. Thank you not only for all of your prompt help but also your friendship. Thank to Athena Sung-Miller who took on the CHIPS Lab Manager role, and handled all of my orders & special requests expeditiously—crucial to completing my PhD.

Thank you to my colleagues Zhe Wan, Xuefeng Gu, Johnathan Cox, Siyun Qiao, and Faraz Khan for all of the device physics discussions and late-night problem solving on the NeuroCTT project. Thank you to SivaChandra Jangam, Krutikesh Sahoo, Guangqi Ouyang, Randall Irwin, and Haoxiang Ren, among many others, for your help in the lab on various projects. Thank you to undergraduate students William Whitehead and Fred Chu for your contributions to our IBM TrueNorth and 3DS-Memory reliability studies. Thank you to Dr. Adeel Bajwa for your mentorship, friendship, and everything you have taught me over the years; I consider you a life-long friend. Thank you to Dr. Boris Vaisband for your mentorship and general interest in my well-being, I will never forget the happy-hour venting sessions at Barney's Beanery in Westwood. Thank you to Dr. Amir Hanna for all of your help in CNSI

and mentorship as well.

Thank you to my UCLA collaborators Prof. Vwani Roychowdhury and Tianyi Wang who were instrumental in developing new methods for training analog-based in-memory computing networks. Additionally Thank you to Dr. Bilwaj Gaonkar and Dr. Luke Macyszyn for all of our medical collaborations with the UCLA Neurosurgery Department.

Thank you to my PhD peers including Saptadeep Pal, Irina Alam, Sumeet Singh, Wojciech Romaszkan, and Uneeb Rathore for all of the fruitful whiteboard discussions. And, thank you to many of the friends who have supported me along my academic journey including Arpita Iddya, Letty Treviño, Felicia Hsu, Chen Xie, Brian Zutter, Erin Askounis, Trevor Black, Jonny Wong, Sharon Wong, Matteo Vesprini-Heidrich, & Jordan Robertson, among many others.

I would also like to acknowledge my academic and industry collaborators Toshiaki Kirihata (Global Foundries), John Barth (Synopsys), Bill Cowell (On Semiconductor), Robert Brennan (On Semiconductor), Jeffrey Dods (On Semiconductor), Kevin Mcilvain (IBM), Gerassimos Giannoulis (IBM), Jung Yoon (IBM), Prof. Janakiraman Viraraghavan (IIT Madras), Prof. Michael Alles (Vanderbilt University), Prof. Brian Sierawski (Vanderbilt University), Prof. Robert Reed (Vanderbilt University), Prof. Enxia Zhang (Vanderbilt University), Prof. Andrew Sternberg (Vanderbilt University), Michael McCurdy (Vanderbilt University), and Dr. Rachel Brewer (Vanderbilt University).

Thank you to GlobalFoundries for all of their 22FDX MPW and device support. Thank you to On Semiconductor for their extensive support of the CTT project over the years. Thank you to IBM Corporation for providing access to the IBM TrueNorth Neurosynaptic System which allowed us to conduct extensive studies regarding the effect of radiation on brain-inspired computing. Thank you to the Defense Threat Reduction Agency (DTRA) for your funding support allowing us to explore radiation effects on both brain-inspired computing and 3D Architectures. Finally, thank you to the entire CHIPS Consortium for their funding, mentorship, and overall guidance.

Thank you to Luna and Moon, who forever will be kittens at heart, for your positive impact on my mental health and wellness and a great addition to the new family I'm starting with my amazing life partner Charlene.

Thank you to my family including Michelle, Chelsea, Tyler, Tom, & Maria for celebrating with me every time I came home and for all of your support over the years.

And in the end, thank you most to my parents, Diane and Tim. To my mom, thank you for being an incredible support force throughout the challenges of my program. I'll never forget the time you flew down to LA just to drive me home after one of my circuit tapeouts. To my dad, thank you for your friendship, mentorship, and free teaching. And most of all, thank you for appreciating what I do and for recognizing the value of what I've accomplished in my program.

VITA

2014	IC Test Engineering Intern, Finisar Corporation.
2015	Reliability Engineering Intern, Space Systems Loral (SSL).
2016	B.S. (Electrical Engineering, Integrated Circuits), UCLA.
2018	M.S. (Electrical & Computer Engineering, Circuits), UCLA.
2019	Graduate Electrical Engineering Intern, Global Foundries.
2016–2022	Graduate Student Researcher, ECE Department, UCLA.
2021–2022	Teaching Assistant, ECE Department, UCLA..
2020–2022	Board Member, Associated Students UCLA (ASUCLA) Board.

PUBLICATIONS

Siyun Qiao, **S. Moran**, D. Srinivas, S. Pamarti, and S. S. Iyer, "Demonstration of Analog Compute-In-Memory Using the Charge-Trap Transistor in 22 FDX Technology," International Electron Devices Meeting (IEDM), 2022. (submitted)

Steven Moran, S. S. Iyer, Z. Wan, S. Pamarti, "NEURAL NETWORK SYSTEM WITH NEURONS INCLUDING CHARGE-TRAP TRANSISTORS AND NEURAL INTEGRATORS AND METHODS THEREFORPCT/US2021/053422, 2020. (published)

Rachel Brewer, J. Cox, D. R. Ball, **S. Moran**, B. D. Sierawski, P. F. Wang, E. X. Zhang, D. M. Fleetwood, R. D. Schrimpf, S. S. Iyer, and M. L. Alles, "Total Ionizing Dose Responses of 22nm FDSOI and 14 nm Bulk FinFET Charge-Trap Transistors," IEEE TNS, 2021.

Steven 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, 2020.

Rachel Brewer, **S. Moran**, J. Cox, B. Sierawski, M. McCurdy, E. X. Zhang, S. S. Iyer, R. D. Schrimpf, M. Alles, and R. Reed, "The impact of proton-induced single events on image classification in a neuromorphic architecture," IEEE TNS, 2019.

Steven Moran, J. Cox, R. Brewer, B. Sierawski, and S. S. Iyer, "Radiation Effects on Brain-Inspired Computing," GOMACTech, 2019.

Zhe Wan, **S. Moran**, X. Gu, J. Cox, and S. S. Iyer, "Characterization Approaches to Test the Robustness of Neuromorphic Systems," GOMACTech, 2019.

Rachel Brewer, **S. Moran**, J. Cox, M. McCurdy, R. Erbrick, M. Alles, R. Reed, S. S. Iyer, and B. Sierawski, "Proton-Induced Classification Changes in a Neuromorphic Computing System," Single Event Effects (SEE) Symposium, 2018.

Steven Moran, B. Gaonkar, W. Whitehead, A. Wolk, L. Macyszyn, and S. S. Iyer, "Deep Learning for Medical Image Segmentation – using the IBM TrueNorth Neurosynaptic System," SPIE Medical Imaging, Feb. 2018.

CHAPTER 1

Introduction

1.1 Motivation

Deep learning and neuromorphic computing are heavily inspired by the incredible energy-efficiency of the human brain. The human brain operates on a massive scale compared to today's largest semiconductor chips at a minuscule fraction of the overall power budget:

> Human Brain: $\sim 100B$ neurons, $\sim 1,000T$ synapses, $\sim 50W$, $\sim 1L$ [Zha19, Her09]
> Intel Loihi: $\sim 131K$ neurons, $\sim 130M$ synapses (Intel 14nm, $\sim 60mm^2$) [Dav18]
> IBM TrueNorth: $\sim 1M$ neurons, $\sim 256M$ synapses, $\sim 70mW$ (Samsung 28nm, $\sim 400mm^2$) [Ami13, Mer14, Saw16, Ess16]
> SpiNNaker 106 System: $\sim 1B$ neurons, 100kW (130nm technology, with an overall form factor of $10\times$ 19-inch racks) [Fur14, Yan19]

Neuromorphic computing aims to to leverage a semi-analogous architecture to that of the human brain to tackle problems spanning multiple domains including object detection, segmentation, and language recognition, among many others. The success of today's 'deep learning' efforts has led to larger and larger network designs, stressing computational requirements for modern computing architectures. As an example, the Generative Pre-trained Transformer 3 (GPT-3) model for generating human-like text has over 175 billion network parameters [BMR20]. Alternative avenues beyond device & system scaling must be explored to reduce energy costs considering the expected trends in workload growth.

1.2 Limitations to Reaching 'Brain-scale' Computing

Multiple challenges exist to designing 'brain-scale' computing systems, including (1) physical silicon die size limitations, (2) interconnect density, (3) designing adaptable & reconfigurable hardware, (4) memory bandwidth, (5) memory size limitations, (6) complex routers for connecting large multi-chip systems, (7) power & energy budgets, and (8) thermal dissipating & cooling, among many others.

From a scaling perspective, semiconductor chips are typically confined by the maximum EUV Lithography-defined reticle size of $104 \times 33mm^2$ (max. die size: $26 \times 33mm = 858mm^2$ for reduction ratio of $4\times$) [Neu13]. As an example, the largest *consumer* microprocessor to-date is the Apple M1 Pro CPU with $\sim$57 billion transistors and spans $432mm^2$ utilizing TSMC's 5nm technology [Ana21], however, IBM has been designing larger mainframe ICs such as the z14 ($696mm^2$, 14nm) since 2017. While current EUV technology supports chip sizes up to $\sim 858mm^2$, most chips stray away from this upper limit due to yield considerations. Recent spiking-neural network (SNN)-based architectures such as Intel Loihi and the IBM TrueNorth Neurosynaptic System both require scaling up by $\sim 10^4\times$ in order to reach 'brain-scale'. Additionally, as we scale these systems into large multi-chip–and multi-board–systems, power budgets go through the roof such as in the case of the SpiNNaker 106 System with a power budget of $\sim 100kW$ while only able to emulate $\sim$1% of the human brain's neurons. While monolithic wafer-scale processes enable comparatively massive systems such as the Cerebras' WSE-2 $46,225mm^2$ system ($\sim 2.6T$ transistors, $850K$ cores, & $40GB$ on-chip memory) [Lie21, Roc20], reducing expensive & repeated memory accesses is left somewhat unaddressed.

In-Memory Computing (IMC) offers the potential to improve system energy-efficiency and throughput especially for ML workloads by performing computation directly within the memory array itself—eliminating costly memory accesses.

1.3 In-Memory Computing (IMC)

In-Memory Computing (IMC) offers a promising solution to the von Neumann Memory Bottleneck by performing vast Vector Matrix Multiplication (VMM) locally in memory where trained model weights are stored which (1) eliminates unnecessary memory accesses, (2) minimizes energy spent due to memory accesses, and (3) naturally allows for simple, parallel computation.

In-Memory or Near-Memory computing enables computation directly within, or nearby, the memory itself. Typically, In-memory Computing consists of a cross-bar architecture with volatile or nonvolatile weights. A non-exhaustive list of proposed memory elements for weight-storage is provided below:

- SRAM [JXH21, Don20, CLF21, SCL21]
- DRAM [YKC19, XNS21]
- Dynamic-Analog RAM (DARAM)[CCG21]
- Flash [BV20, LL20]
- Phase-Change Memory (PCM) [Jos20, Nar21]
- Resistive RAM (RRAM) [Moc18, Che18, Xue19, Cor20, Liu20, Li21]
- Spin-Transfer Torque Magnetic RAM (STT-MRAM) [Kim11]
- Ferroelectric hafnium oxide-based transistors (FeFET) [BMM21]
- Monolayer MoS_2 Transistor (2T-1C) cells [WTX21]
- **Charge-Trap Transistor (CTT) [Kha15, Kot15, Vir16, Kha17, Hun19, Kha20, GI17, Gu18, GWI19, Wan20b, Wan20a]**
- Silicon-Oxide-Nitride-Oxide-Silicon (SONOS) [KRP18, APR20, Xia22].

The Charge-Trap Transistor (CTT) is identified as an excellent candidate nonvolatile memory device for IMC applications. The CTT is highly competitive against other proposed analog nonvolatile memory devices such as RRAM, PCM, MRAM, etc. given that it is completely CMOS-compatible, a 3-terminal device requiring no select transistor or low-leakage selector and scalable to smaller CMOS technology nodes. Additionally, the CTT can be accurately fine-tuned to a specific weight with sufficient retention and manageable device variance.

Through a variety of techniques, cross-bar architectures can be sufficiently optimized to improve energy-efficiency by applying voltage or pulse-width-modulated (PWM) inputs and computing weighted sums by measuring the column current in the analog domain. A variety of proposed input and ADC architectures are discussed in more detail in Chapter 3.

1.4 Dissertation Outline

A majority of this dissertation focuses on designing efficient deep learning hardware for edge applications. Relevant discussion based on experiments on the reliability of compute for both small-scale and large-scale neuromorphic systems is also included in the appendices. This dissertation is organized as follows:

Chapter 2 provides a deep dive on the CTT device and benchmarks it against competing nonvolatile memory devices. Sections 2.2 & 2.3 demonstrates excellent weight programming results obtained using the Pulsed-Gate Voltage Sweep (PVRS) *fine-tune* programming algorithm. Section 2.4 demonstrates sufficient cell retention at $85°C$ and Section 2.5 provides a thorough comparison of the CTT against other candidate analog nonvolatile memories.

Chapter 3 provides a detailed overview of the entire NeuroCTT Architecture for accelerating dot-product computation for fully-connected neural network layers. Sections 3.2-3.4 detail past chip design efforts, lessons learned, testing infrastructure design, and the overall testing user interface.

Chapter 4 details all hardware results obtained from past chip design efforts. Sections 4.1-4.3 detail hardware results demonstrate on-chip weight programming, weight verification, and the demonstration of a CTT-based MAC Engine with programmed weights. Section 4.4 provides a circuit-level simulation platform for evaluating the effect of both circuit-induced errors and device non-idealities on the performance and accuracy of the NeuroCTT architecture.

The accuracy and radiation tolerance of analog and digital-based Neural Network hardware is discussed in this dissertation in Appendix A. Digital-based neural networks are highly resilient to bit errors in network weights due to their inherent redundancy, demonstrated in Appendix A.1 with the IBM TrueNorth Neurosynaptic System [Ami13, Mer14, Saw16, Ess16]. In comparison, Analog-based neural networks are tolerant to bit errors and single-event upsets (SEUs), but are susceptible to weight programming variance ($w_{trained} \neq$

$w_{deployed}$), retention, temperature-dependent effects, and systematic threshold voltage shift (ΔV_{TH}) due to Total-Ionizing dose (TID). Appendix A.2 discusses the effect of these device variances on the accuracy of analog-based in-memory computing. The CTT Device & NeuroCTT Architecture are utilized as a baseline for evaluating the error tolerance of analog NVM-based IMC systems. Appendix A.3 concludes this section by examining the effects of Total Ionizing Dose (TID) on programmed CTT weights and provides potential remedies for lowering the device sensitivity to TID effects.

Appendix B concludes this dissertation by studying the effects of radiation on 3D-stacked memories and architectures. For large-scale neuromporphic system, system reliability is increasingly becoming more important. Many larger-scale systems today leverage 3D memories including High-Bandwidth Memory (HBM) and 3D-Stacked (3DS) DRAM in order to store large multi-layer networks. As an example, the Generative Pre-trained Transformer 3 (GPT-3) model for generating human-like text has over 175 billion network parameters trained on $\sim 45TB$ of training data across multiple datasets [BMR20].

Appendices C and D are provided as supplementary information for the CTT-Hardware-based Inference Realistic Circuit Universal Simulator (CIRCUS) and NeuroCTT chip testing.

Original contributions of this dissertation include (1) first demonstration of an *on-chip* analog MAC Engine using the CTT fabricated in a commercial CMOS technology, (2) PVTS methodology for *fine-tune* programming devices with programmable pulse widths without requiring ramping voltage supplies, (3) analysis of all the variances present in analog in-memory compute (IMC), (4) CIRCUS circuit-level simulation framework for evaluating the accuracy of analog IMC engines, (5) experimental radiation tolerance of analog & digital neuromorphic hardware, and (6) experimental radiation tolerance of 3D-stacked architectures using 3DS DRAM memory to detect the propagation of bit errors across multiple dies within each stack.

CHAPTER 2

Charge-Trap Transistor (CTT) as an Analog Nonvolatile Memory Device

The 'Charge-Trap' transistor (CTT)—any high-κ $HfSiO_x$ logic transistor—has been demonstrated as a re-writable nonvolatile memory device for digital [Kha15, Kot15, Vir16, Kha17, Hun19, Kha20] and analog [GI17, Gu18, GWI19, Wan20b, Wan20a] applications. The CTT is a 3-terminal device and completely CMOS-compatible, providing several performance & cost advantages over other proposed analog nonvolatile memory (aNVM) devices such as RRAM, PCM, MRAM, and SONOS. While GlobalFoundries 22nm Fully-Depleted Silicon-on-Insulator (FDSOI) technology is utilized for this project, the CTT has also been demonstrated using GlobalFoundries 12LP/14LP FinFET technology and all nodes utilizing high-κ gates as well. Research has primarily focused upon SOI technologies as the charge-trapping effect is greatly aided by the *enhanced* self-heating assisted trap tunneling afforded by SOI technologies.

The NeuroCTT system detailed thoroughly in Section 3.1 is a neuromorphic classifier featuring twin-cell CTT-based *analog synapses*. These *analog synapses* are biased in sub-threshold regime ($V_G = \sim 200mV$, $V_D = \sim 50 - 200mV$) during inference, where the weights are considered as the device on-state channel conductance or equivalently the threshold voltage ($G_{ch} \sim V_{TH}$). Twin-cell weights are utilized to support positive, negative, and zero-valued weights by programming each twin-cell weight to achieve the target differential weight (ΔG_{ij})

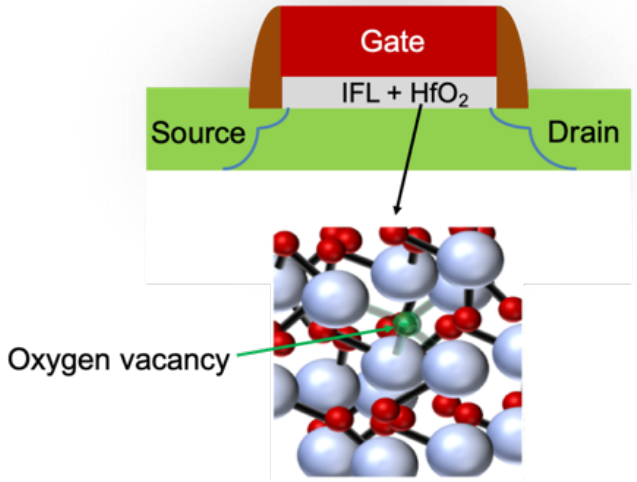
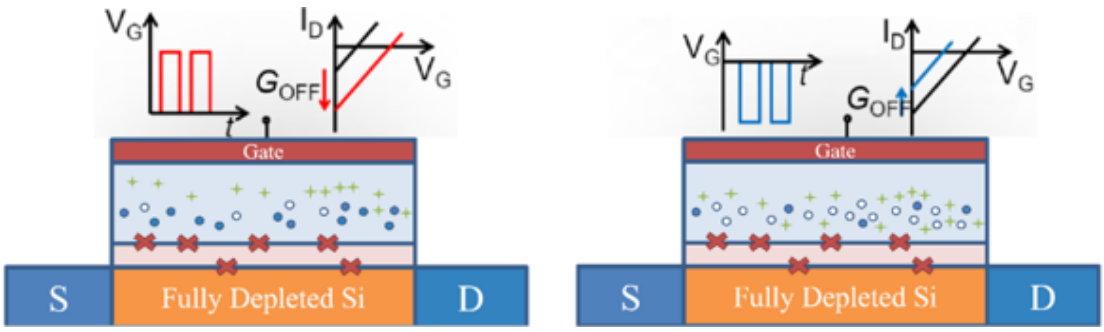


Figure 2.1: **Oxygen Vacancies in $HfSiO_x$ logic transistors.** High-κ gates are inherently "trappy" due to oxygen vacancies that can trap electrons.

2.1 CTT Device Overview

The "Charge-Trap Transistor" (CTT), or any high-κ gate-first logic transistor, relies on electron trapping in the gate oxide to modulate the threshold voltage of the device in a nonvolatile manner. The high-κ gate is inherently "trappy" due to oxygen vacancies in the gate dielectric (Fig. 2.1) that can easily trap electrons. The basic process is believe to be trap-assisted tunneling that is enhanced by self-heating. Silicon-on-insulator (SOI) technologies enhance this temperature-assisted trapping as shown with GlobalFoundries 22FDSOI technology in [Kha15]. Self-heating-assisted trapping may also generate additional traps. Charge-trapping has been shown to be more pronounced in "gate-first" processes.

Devices are programmed by applying large V_{GS} $(1.8-2.7V)$, V_{DS} $(1.6-2.2V)$ conditions to the target cell for short (e.g. $50-500\mu s$) pulses, shown in Fig. 2.2a. Detrapping ('erasing') is also possible by reversing the gate voltage polarity as shown in Fig. 2.2b; however, a small random amount of trapping persists without a temperature-induced reset.



(a) Programming (PRG) (b) Erasing (ERS)

Figure 2.2: **CTT Device Programming & Erasing Operations in 22FDX.** (a) CTT devices are programmed ($\uparrow V_{TH}, \downarrow \sigma_{ch}$) by applying large V_{GS}, V_{DS} voltage (red) pulses temporarily to the device. (b) CTT devices are erased ($\downarrow V_{TH}, \uparrow \sigma_{ch}$) by applying large reversed gate-voltage (blue) pulses to the device.

Programming and Erasing corresponds to effectively increasing ($\uparrow V_{TH}$) or decreasing ($\downarrow V_{TH}$) the threshold voltage of the device, respectively. This can be easily observed in Fig. 2.3 where full IV curves have been taken for an *as-fabricated* device (**Pre-PRG**) after programming (Post-PRG) and after erasing (Post-ERS). The *as-fabricated* device is subjected to a series of Programming pulses leading to an increased threshold voltage ($\Delta V_{TH} \sim 250mV$), shown by the Post-PRG curve. After programming, the device is then subjected to a series of Erase pulses, shown by the Post-ERS. The device can be erased and brought back close to the **Pre-PRG** condition but the device cannot be fully reversible, unless the device is "annealed".

Long-Term Depression (LTD) and Long-Term Potentiation (LTP) characteristics, necessary in Spike-Timing Dependent Plasticity (STDP)-based learning [BP01, SRR08], can be demonstrated by applying a series of programming (LTD) or erase (LTP) pulses to a CTT device, as shown in Fig. 2.4.

In order to accurately program each of the devices to a specific target-state, an iterative write-then-verify strategy is utilized and described in more detail in the next section.

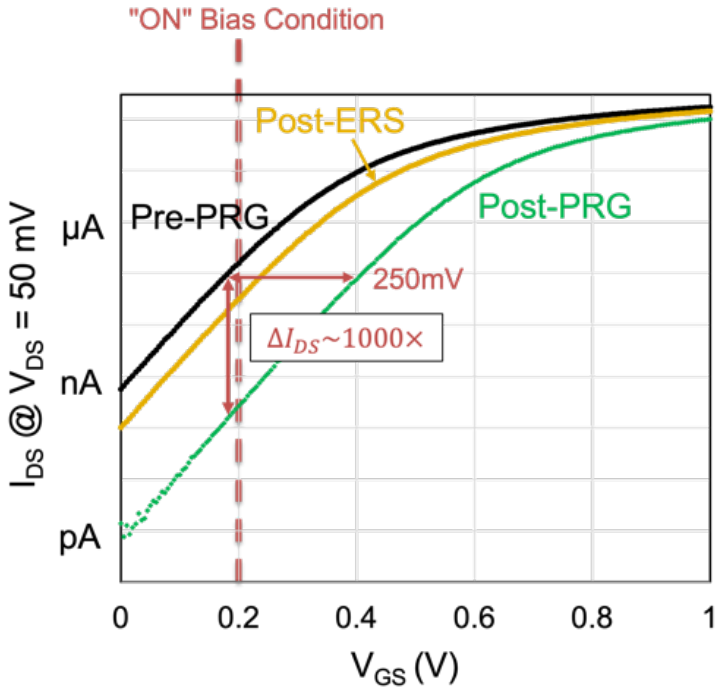


Figure 2.3: **CTT Device PRG/ERS IV Characteristics.**

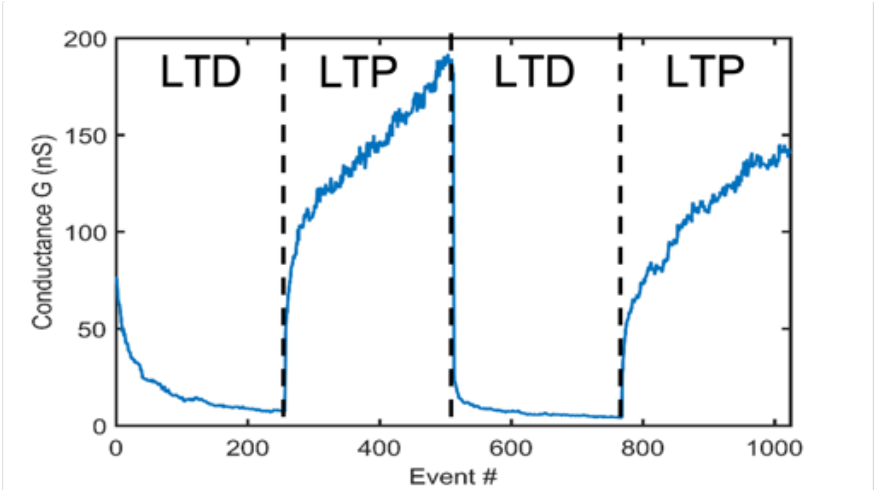
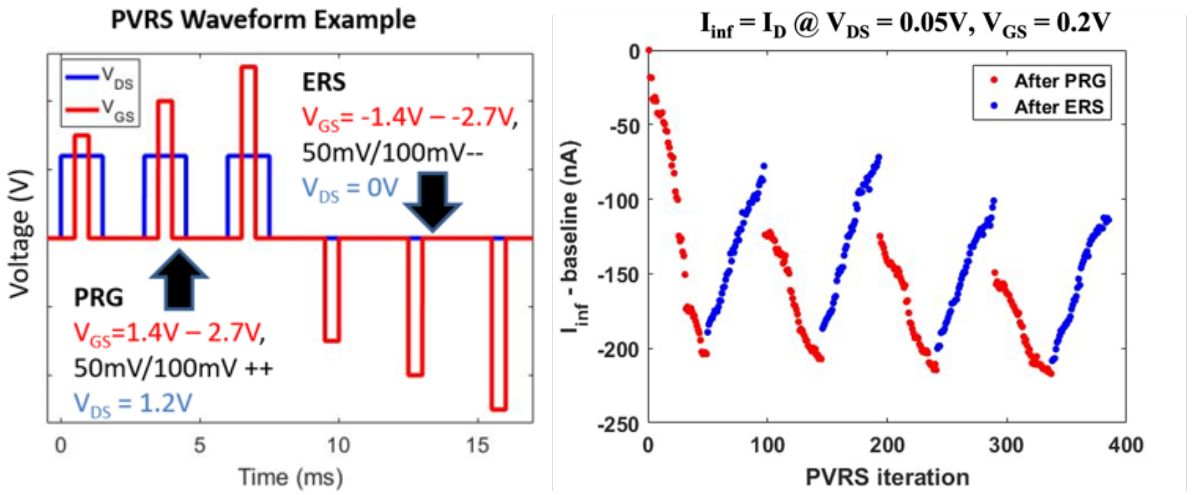


Figure 2.4: **CTT Long-term Potentiation (LTP) and Depression (LTD) Characteristics.** Reversible and reproducible device conductance changes are demonstrated by applying a series of "256" programming pulses followed by a series of 256 "erase" pulses, repeated for a total of 4 cycles. Previously reported in [GI17].



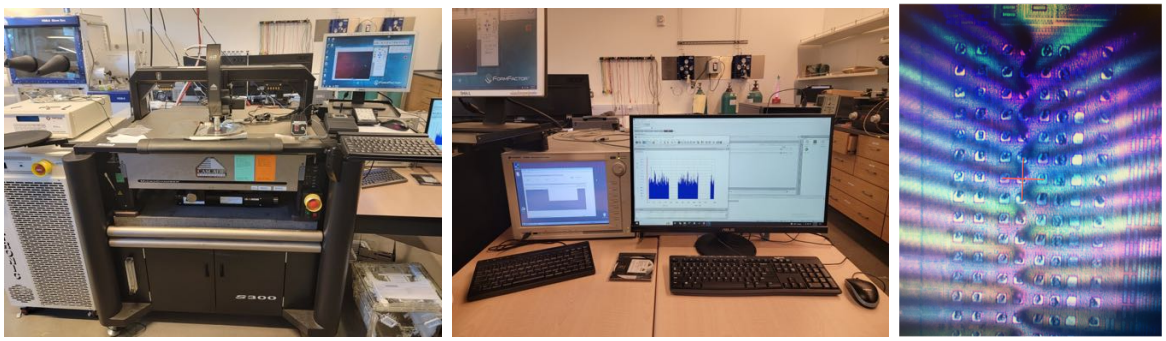
(a) PVRS Technique (b) Adjusting (±) CTT Device Weight using PVRS

Figure 2.5: **Pulsed-gate Voltage Ramp Sweep (PVRS) Programming.** PVRS Technique can be utilized to make fine-tune adjustments to CTT device weights (σ_{ch}, V_{TH}) as shown in [Wan20a].

2.2 Programming Methodology

A write, verify, and re-write (if necessary) strategy combined with the Pulsed-gate Voltage Ramp Sweep (PVRS) technique shown in Fig. 2.5 is used to accurately program and erase the CTT device to a specific target state. PVRS allows devices that have not yet reached their target state to be efficiently programmed by ramping up the gate-voltage condition after each subsequent pulse until the device reaches its target state. This mode of operation is also known as *fine-tune* programming.

Accurately programming and erasing CTT devices within an array also requires special considerations in terms of half-select issues. These array-level issues are explored more in Sections 3.1.4 & 4.1 which discuss the CTT Array design and past array programming results obtained on the NeuroCTT 0.1 and NeuroCTT 0.3 chip designs.



(a) Cascade Probe Station (b) Analyzer & Switch Matrix (c) Probed Die

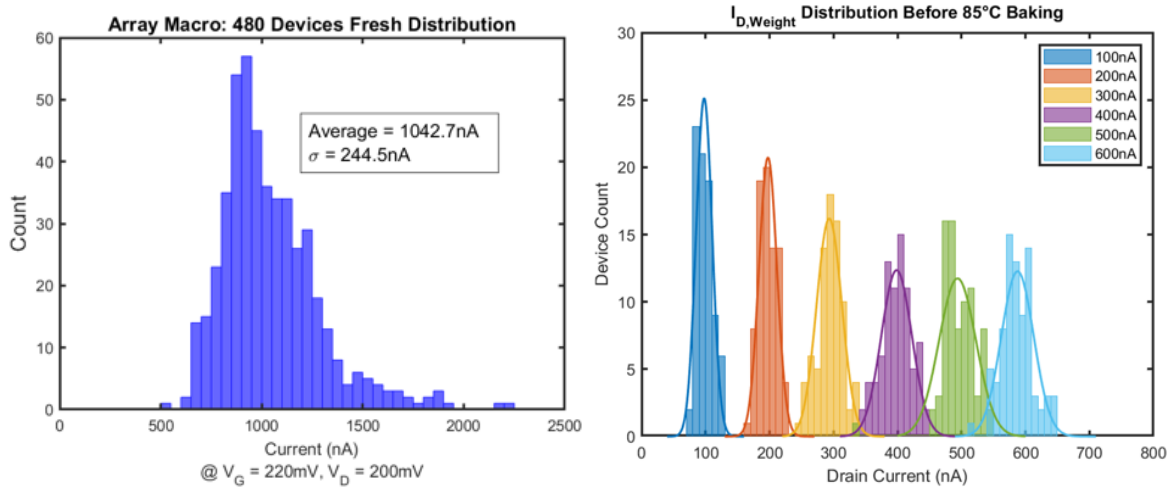
Figure 2.6: **CTT Device Testing using Cascade Probe Station.** Setup includes (a) Cascade 300mm wafer prober as well as (b) Keysight B1500A Analyzer, Switch Matrix, and Gateway Computer. Example probed die shown in (c).

2.3 Programming Variance

CTT device results were obtained using 1×25-pad ($72\mu m$ pitch) scribe line monitor (SLM)-based macros on NeuroCTT 0.1 and 0.3 designs as well as wafer-level device macros. Testing was performed using an in-house Cascade Probe Station with Keysight B1500A Analyzer and $1 \times 25\,(72\mu m\,pitch)$ Celadon Probe card, shown in Fig. 2.6.

Past results have shown that the CTT device (RVT: $W = 428nm, L = 20nm$) measured at $V_{GS} = 200mV, V_{DS} = 200mV$ can be accurately programmed to a specific target current within the range of $100{-}700\,nA$ with a device programming variance of $\sigma_{PRG} = {\sim}48.2\,nA$ or a normalized programming variance of $\sigma'_{PRG} = 48.2nA/(700{-}100\,nA) = {\sim}8\%$ [Wan20a].

Figure 2.7 displays more recent results from the CTT array macros on the NeuroCTT 0.3 chip design. Array macros consisting of 10 row/wordlines and 8 columns of twin-cell CTT devices (160 devices total) contained an array layout identical to that used within the NeuroCTT system for external testing and device programming validation. A group of 80 devices each was programmed to one of 6 target currents ($100, 200, 300, 400, 500,$ & $600nA$) using the testing setup shown in Fig. 2.6 and array macros across 3 chips. The programming



(a) *As-Fabricated* Device Distribution (b) Devices After Programming

Figure 2.7: **CTT Device Programming to 6 Target States.** (a) Distribution of 480 *as-fabricated* CTT (RVT, W=428nm, L=20nm) devices. (b) A distribution of 80 devices were programmed to each of the target currents, centered respectively around 100, 200, 300, 400, 500, & 600nA, and measured at $V_{GS} = 220mV$ and $V_{DS} = 200mV$.

variance (σ_{PRG}) is plotted as a function of the target current in Fig. 2.8. Figure 2.8 reports that the programming variance is a function of the target or programmed current, with a worst-case normalized programming variance of $\sigma'_{PRG} = \sigma_{PRG}/Range = \sim 6\%$, with respect to the $500nA$ target immediately after programming and $\sim$6.5% after 50hr bake at 85°C.

Recent unpublished results have shown that this result can be further improved upon by utilizing *fine-tune* erase. While fine-tune erasing has been shown to offer promising device programming accuracy and retention results, these results are omitted from this dissertation as the NeuroCTT 0.1, 0.2, & 0.3 chip designs were primarily architected to support exclusively *fine-tune* programming and *block* erase.

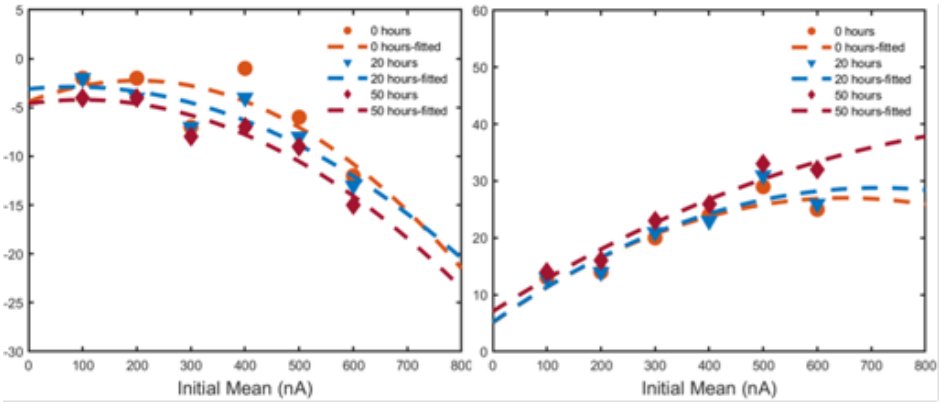
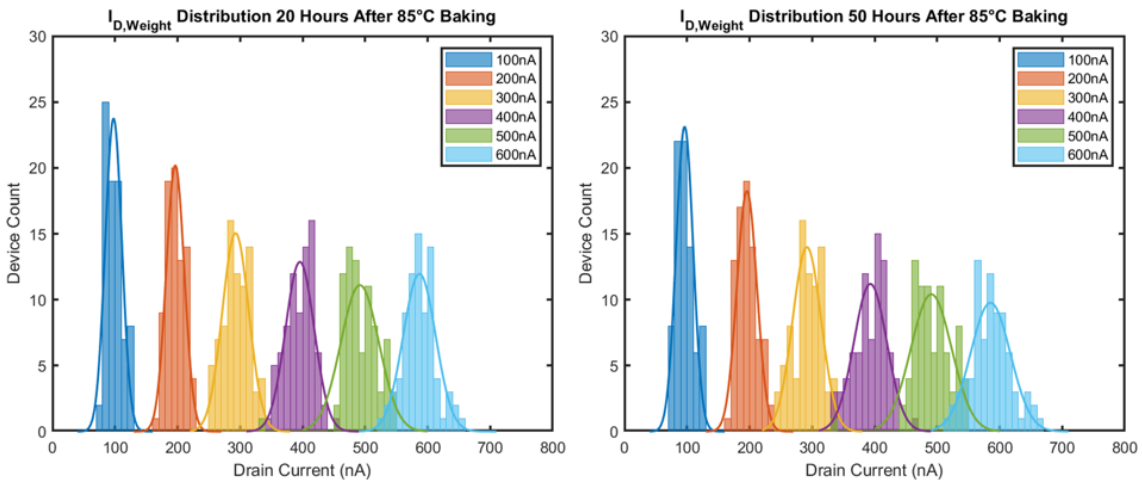


Figure 2.8: **CTT Device Programming Variance and Mean Drift.** (*left*) Measured and modeled mean drifts immediately after programming and after 20 & 50 hours at 85°C. (*right*) Measured and modeled standard deviation immediately after programming, after 20 & 50 hours at 85°C. Devices were not powered on during baking.

2.4 Retention Characteristics

A group of 480 *as-fabricated* devices were programmed to one of 6 target states (100, 200, 300, 400, 500, & 600nA), as shown in Fig. 2.7b. The devices were then baked for 50 hours at 85°C. CTT device programming variance and mean drift before and after baking are provided in Fig. 2.8. Each of the 6 target weight distributions demonstrated excellent retention after 50 hours of cumulative baking, as demonstrated by the plots shown in Fig. 2.9.

A cell retention comparison is performed between the CTT and recently published work with SONOS devices [Xia22] in Tables 2.1 & 2.2. It is observed that the SONOS devices from [Xia22] experience significant degradation at room temperature, where programming variance ($\Delta\sigma_{SONOS}$) and mean ($\Delta\mu_{SONOS}$) significantly drift after sitting for 120 hours at room temperature. It is assumed that these retention issues at room temperature will be exacerbated at higher temperature operating conditions and after longer periods of time. Retention for other Flash devices may vary. Negligible changes in programming variance ($\Delta\sigma_{CTT}$) and mean ($\Delta\mu_{CTT}$) are observed for CTT (428nm) devices after 50 hours at 85°C.



(a) 20hr (b) 50hr

Figure 2.9: **Device Retention over 50hr baking at** $85°C$**.** Devices were first programmed to one of 6 target currents centered around $100, 200, 300, 400, 500, \&600nA$ for $V_{GS} = 220mV$ and $V_{DS} = 200mV$, shown in Fig. 2.7 ($t = 0$). Devices were then baked for (a) 20 hours and (b) 50 hours at $85°C$. Devices were not powered on during baking.

2.5 Analog NVM Device Comparison

Tables 2.1 & 2.2 provide a comparison between state-of-the-art programming and retention characteristics for the CTT and Silicon–Oxide–Nitride–Oxide–Silicon (SONOS) devices, respectively. Significant changes in programming variance and mean are observed for the SONOS device at room temperature after only 120 hours. It is assumed that these retention issues at room temperature will be exacerbated at higher temperature operating conditions and after longer periods of time.

Using exclusively a *fine-tine* programming approach, excellent retention results were demonstrated for the CTT at $85°C$ over 50 hours, with negligible changes in programming variance and mean across all target states. Further improvements are also possible using a *fine-tune erasing* approach.

Target State	$\sigma_{PRG}(0hr)$	$\sigma_{PRG}(50hr)$	$\sigma'_{PRG}(50hr)$	$+\Delta\sigma_{CTT}$	$+\Delta\mu_{CTT}$
100 nA	13.3	13.8	2.8%	0.5	$-2nA$
200 nA	14.9	16.5	3.3%	1.6	$-2nA$
300 nA	20.3	22.8	4.6%	2.5	$-1nA$
400 nA	23.2	25.8	5.2%	2.6	$-6nA$
500 nA	29.5	33.7	6.7%	4.2	$-3nA$
600 nA	27.6	30.9	6.2%	3.3	$-3nA$

Table 2.1: **CTT Retention Over 50 hours at** $85°C$**.** σ'_{PRG} reflects programming variance normalized by the range ($\sigma_{PRG}/(I_{MAX} - I_{MIN})$). Change in Programming Variance ($+\Delta\sigma_{CTT}$) and Mean Drift ($+\Delta\mu_{CTT}$) are calculated based on data obtained after 50hr bake at $85°C$. All currents reported in nA.

Target State	$\sigma_{PRG}(0hr)$	$\sigma_{PRG}(120hr)$	$\sigma'_{PRG}(120hr)$	$+\Delta\sigma_{SONOS}$	$+\Delta\mu_{SONOS}$
100 nA	7	10	2.0%	3	$+10nA$
200 nA	12	18	3.6%	6	$+12nA$
300 nA	17	24	4.8%	7	$+14nA$
400 nA	21	31	6.2%	10	$+12nA$
500 nA	26	38	7.6%	12	$+12nA$
600 nA	27	40	8.0%	13	$+0nA$

Table 2.2: **Retention Comparison with 40nm SONOS device.** SONOS device data adapted from [Xia22]. Change in Programming Variance ($+\Delta\sigma_{SONOS}$) and Mean Drift ($+\Delta\mu_{SONOS}$) are calculated based on data obtained after 120hr bake at $27°C$. SONOS data reflects significant degradation at room temperature ($27°C$) over 120hr period. All numbers reported in nA.

	RRAM [Li21]	**CTT [Kha20]**
Technology	TSMC 40nm	GF 22nm/14nm/12nm
Reported Retention Data	Up to 30 hours	1000 hours
Relative Fluctuation	$\sim$200%	$< 5\%$
Long-Term Retention	—	10yr. charge loss $< 25\%$ at $125°C$
R_{ON}/R_{OFF} Ratio	$\sim$20$\times$	at subthreshold: $\sim$1000$\times$

Table 2.3: **CTT and RRAM Comparison for Digital Applications.** Weight instability, or large fluctuations over time, seen for RRAM devices in the High-Resistance State (R_{HS}), reported in [Li21]. Extensive retention data for CTT for 1000 hours at various baking temperatures ($25, 85, 125, 180, \&240°C$) reported in [JLV18].

While a majority of this dissertation is focused on analog in-memory computing applications for embedded nonvolatile memory devices, it's valuable to benchmark the them for digital memory applications as well. In this context, it is worthwhile comparing the CTT with RRAM devices. Table 2.3 compares recently published RRAM work [Li21] with the CTT device [Kha20, JLV18] for digital applications. Past work has shown that the CTT device as a digital NVM experiences a projected charge loss over 10 years at $125°C$ of <25%, with extensive retention data available over 1000 years at temperatures between $25 - 240°C$. Additionally, the CTT device is currently available by GlobalFoundries as a One-Time Programmable Memory (OTPM) product rated for 10 years at or above $85°C$, with a multiple-time programmable version currently in development. In contrast, RRAM devices utilizing a qualified 40nm technology node suffer poor long-term retention with limited published data (<30 hours) above room temperature. Additionally as reported in [Li21], large weight instabilities or fluctuations ($\sim$200%) over time exist for devices in the High-Resistance State (HRS). Generally, most varieties of RRAM devices provide a small R_{ON}/R_{OFF} ratio, requiring an additional access transistor or non-linear selector (e.g. 1T1R cells).

	Properties	Size	$R_{ON_{MIN}}$	$\frac{G_{MAX}}{G_{MIN}}$	σ'_{PRG}
CTT	22FDSOI	$22nm \times 428nm$	$0.25M\Omega$	~ 7[1]	6%
PCM	$Ge_2Sb_2Te_5$	$90nm$ CMOS ($r_{TOP} = 100nm$)[2]	$\sim 40k\Omega$	~ 25	3.8% ($\sigma = 0.94\mu S$)
STT	MRAM	$28 \times 70nm$	$14.8k\Omega$	2	$\sim 6\%$
		$80nm \times 180nm$	$2k\Omega$	2	$\sim 8\%$
RRAM	$Ag : a-Si$	$100 \times 100nm$	$2.5M\Omega$	~ 10	—
	$AlO_x/TiN/PCMO$	$150 \times 150nm$	$6.9M\Omega$	~ 7	—
	AlO_x/HfO_2	$400 \times 400nm$	$16.9k\Omega$	~ 3	—
	Winbond-90nm	$2\times$1T1R ($\sim 62F^2$)	—	—	$\sim 5\%$
	TSMC-40nm	2T2R	$\sim 1M\Omega$	~ 20	—
SONOS	Cypress-40nm	1T1S[3]	$\sim 2M\Omega$	~ 32	$\sim 3.5\%$

Table 2.4: **Analog NVM Device Comparison.** Information adapted from [GWI19, BSN14, Jos20, Kim11, JCE10, PSK13, WMS16, He20, Li21, KRP18, APR20, Xia22]. Last column refers to normalized programming variance, $\sigma'_{PRG} = \sigma_{PRG}/Range$. Limited Programming Accuracy & Retention information is available for RRAM devices. [ZZP18] suggests that the normalized programming variance can be as unacceptable as $\sim$20% for some RRAM devices, especially severe in the High-Resistance-State (HRS). [1] For current design target. Experiments have shown that this can be further improved to $>100\times$. [2]r_{TOP} refer to the radius of the top of the electrode. [3]1T1S denotes that cell consists of 1 select transistor and 1 SONOS device.

	FLASH [BGK16]	**RRAM** [LHY20]	**PCM** [Jos20]	**CTT**
Material Prop.	Floating Gate	Winbond-90nm	$Ge_2Sb_2Te_5$	22FDSOI
Cell Config.	4T	2T2R	2T2R	2T
Bits/Cell	Signed $8\,b$	$1-3\,b$	$1-4\,b$	Signed $5\,b$
Max. R_{ON}/R_{OFF}	$>10^5$	$<10^2$	$<10^2$	$>10^5$
PRG Energy/bit	$10-100nJ$	$<1nJ$	$<10nJ$	$\sim 100nJ$
PRG Voltage	$\sim 10V$	$\sim 2V$	$2-4V$	$\sim 1.5-2V$
INF Voltage	$1-4V$	$\sim 1V$	$\sim 0.3V$	$\sim 200mV$
Add. Fab Cost	yes	yes	yes	no

Table 2.5: **Additional Comparison Including FLASH.**

In summary, the CTT is (1) completely CMOS-compatible and already exists as a qualified manufacturing process, (2) includes a 'free' non-linear, low-leakage selector (e.g. gate) with $\frac{I_{on}}{I_{off}} \sim 10^5$, (3) is scalable to smaller technology nodes such as 12FDX, and (4) offers a high (e.g. $R_{ON} \sim 250k\Omega - 3M\Omega$), (5) excellent device controllability, (6) manageable device variance $\sigma'_{PRG} = \sigma_{PRG}/Range \sim 6\%$, (7) large number of analog states (twin-cell $\sim 5\,b$), & excellent retention characteristics compared to other candidate analog embedded nonvolatile memory devices.

The CTT device, along with many other candidate analog nonvolatile memory devices, suffers from low endurance, long write times, large cell area limited by low-resistance routing, and requires iterative fine-tune programming with weight verification to accurately program each weight. These limitations likely indicate that the CTT device cannot be used for highly iterative learning due to limited endurance, however, the CTT is an excellent candidate for *Inference* or in-memory computing with nonvolatile weight storage and infrequent weight tuning.

CHAPTER 3

NeuroCTT Architecture

3.1 Architecture

Figure 3.1 describes our proposed inference engine with pulse-width modulated inputs applied to the CTT gates, a differential integrator which integrates the differential current on each twin-cell column and converts the accumulated charge to a pulse-width modulated output signal that can be conveniently applied as an input to subsequent layer(s) without requiring any digital interface in between layers.

Additionally, the proposed inference engine can be efficiently expanded into a multi-layer network without any digital interfaces between layers, as the pulse-width-modulated (PWM) output of the n^{th} layer can be directly applied as inputs to the $(n+1)^{th}$ layer.

3.1.1 Overview

The proposed architecture in Fig. 3.1 consists of 3 majors blocks: (1) WL Drivers, (2) CTT Array, and (3) Output Neurons—conceptualized by Fig. 3.3. The WL Drivers convert differential digital (e.g. $0/0.8V$) pulse-width modulated inputs to subthreshold range (e.g. $V_G = 50 - 200mV$) PWM inputs. The CTT Array stores the synaptic weights and produces a differential current output waveform. Each Neuron consists of a (i) differential integrator which integrates this differential current waveform and (ii) a comparator for computing the output pulse width. The output pulse width is generated by discharging the accumulated charge during inference, Q_{INF}, using a *dc* discharge current source and computing the time

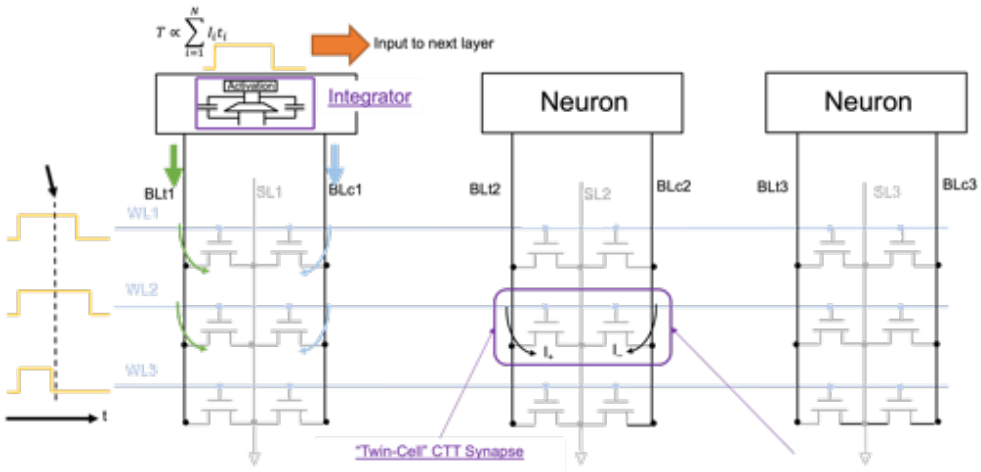


Figure 3.1: **Proposed NeuroCTT Inference Engine.** Example demonstrates a single-layer network with 3 inputs (WLs) and 3 outputs (columns/neurons).

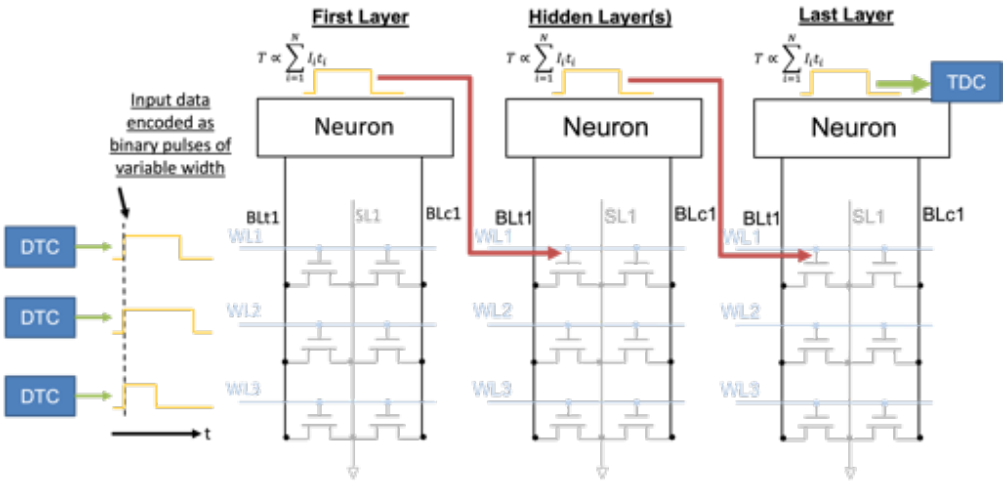


Figure 3.2: **Multi-Layer Inference Engine.** Example demonstrates a 3-layer network where the outputs of the 1^{st} and 2^{nd} layers are directly applied as inputs to subsequent layers. The output of the last layer is optionally converted back to digital using an ADC or Time-to-Digital converter (TDC). Inputs to the first layer are assumed to be digital and converted to pulse-width modulated (PWM) inputs using a digital-to-time (DTC) converter.

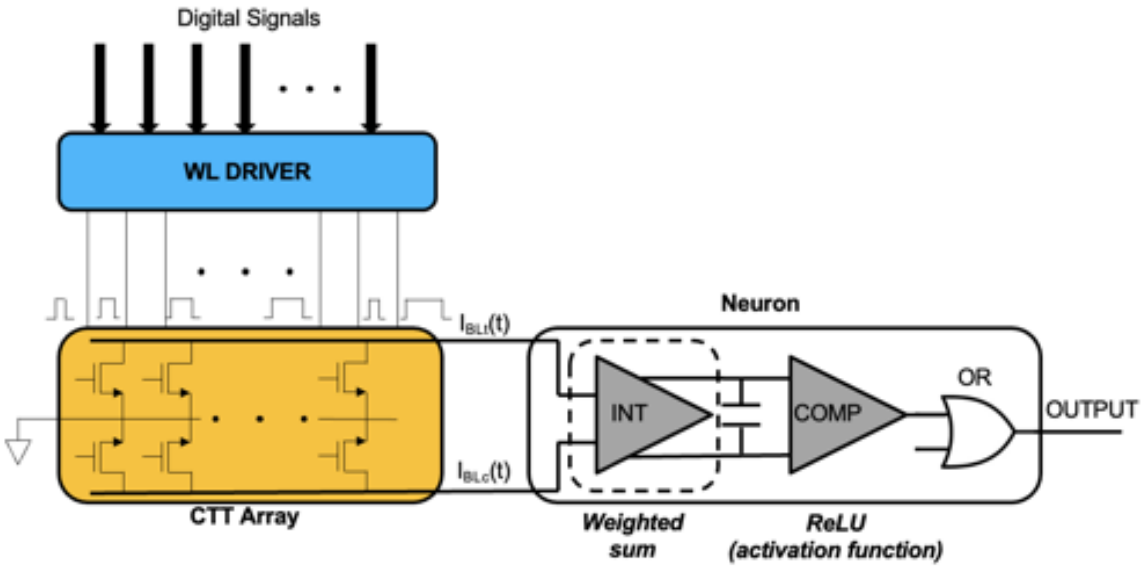


Figure 3.3: **NeuroCTT Architecture Overview.**

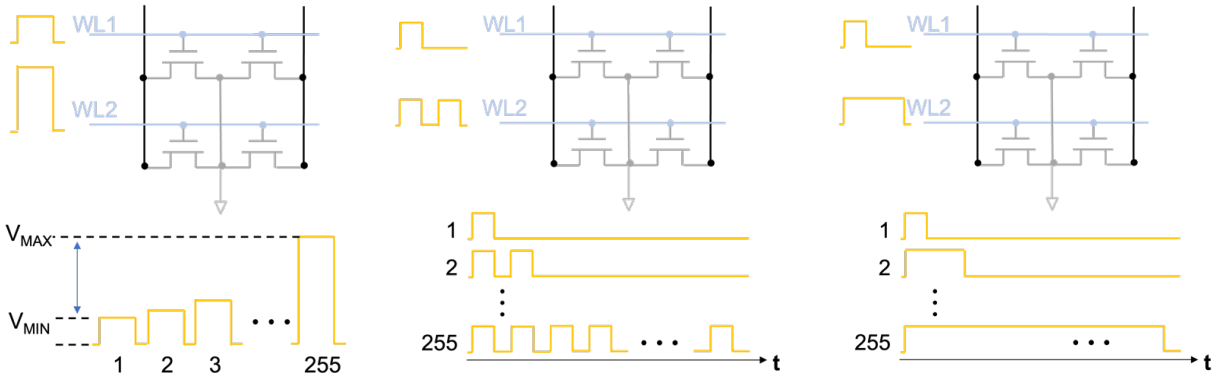
it takes to fully discharge the respective output capacitance using the comparator circuit:

$$t_{PWM,OUT} \propto \frac{Q_{INF}}{I_{DISCHARGE}} \tag{3.1}$$

This function effectively implements the ReLU activation function by producing an output pulse proportional to the accumulated output, Q_{INF}, for only positive-valued outputs by using a single polarity discharge source. Nonlinear activation functions, such as the ReLU function, are required by multi-layer neural networks.

3.1.2 Input Architecture

Various input architectures have been proposed for analog-IMC applications, shown in Fig. 3.4. When biased in subthreshold, the CTT is a strong function of V_{GS} and a weak function of V_{DS}, described by Equation 3.2. This would render the amplitude-based input configuration in Fig. 3.4a useless as the current is exponentially dependent on V_{GS}, making it difficult to control the device weights. In other words, the 'weight' would not be constant for all inputs; rather it would be a function of the input voltage, V_{GS}.



(a) Amplitude-based (b) Pulse-based (c) Pulse-Width Modulated

Figure 3.4: **Example Input Architectures.** Example designed for 8-bit inputs.

$$I_D = I_o \times e^{(V_{GS}/nv_T)}(1 - e^{-V_{DS}/v_T}), v_T = kT/q \tag{3.2}$$

Pulse-based (Fig. 3.4b) and Pulse-Width Modulated (Fig. 3.4c) were considered in this work. A pulse-based approach has been used in many recent works such as [Liu20]. While both methods are similar, both maintain the input at some constant voltage level for an equivalent total period of time, multiple trade-offs exist. Pulse-based methods require computing and accumulating the output after every individual pulse limited overall throughput of the system, while the Pulse-Width modulated scheme is difficult to implement because it requires a substantially larger integrating capacitance, C_{INF}, to integrate the differential current waveform.

Assuming similar I_{CM} and I_{DM} requirements for a Pulse-based and Pulse-Width-Modulated integrator design, if the total integrating capacitance for a 1-bit pulse-based input is C_{INF_o} and 8-bit inputs are used ($N_{input} = 8$), then:

$$C_{INF_{PWM}} = 2^{N_{input}} \times C_{INF_o} \tag{3.3}$$

While PWM designs require a substantially larger integrating capacitance which incurs a large area penalty, they in general provide higher overall throughput and efficiency. The

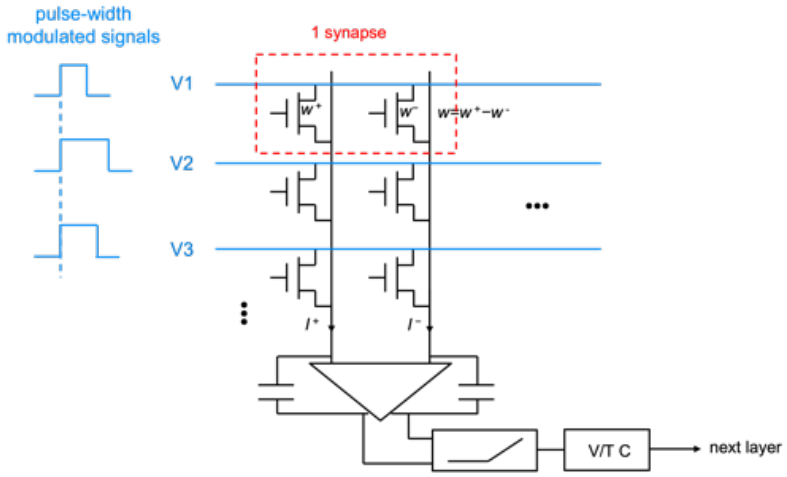
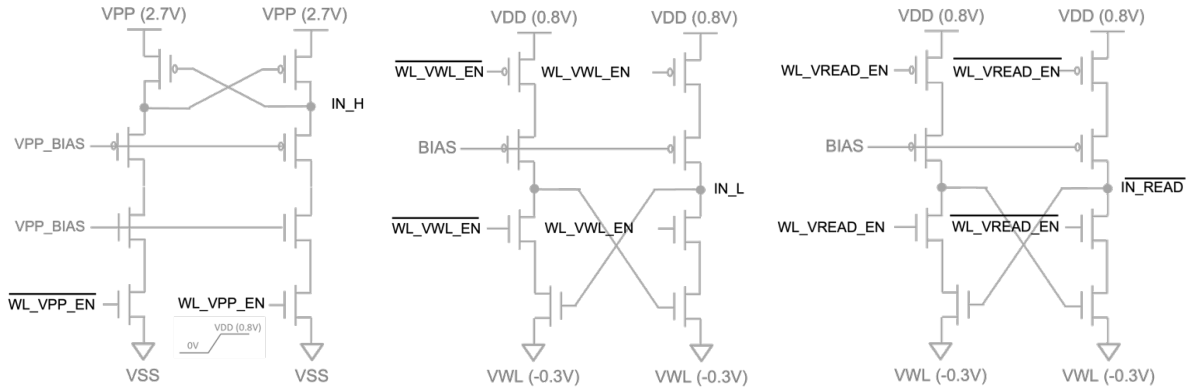


Figure 3.5: **CTT Array with Drain Inputs.** An alternative is to apply the inputs to the drains instead of the gates, where the gate would be set to some constant bias voltage to maintain the device in the subthreshold regime. A worse I_{on}/I_{off} ratio is observed when the gate is connected to some $V_{bias} > 0V$, causing substantial leakage for "off" devices and degrading the overall computational accuracy.

PWM design can also be more easily power-gated when not in use.

Other configurations were also considered such as applying PWM inputs to the CTT drain instead of the gate terminal—shown in Fig. 3.5. This configuration is not ideal as it provides a poor I_{on}/I_{off} ratio as the current is exponentially dependent on V_{GS}. Figure 3.1 represents the selected design where PWM inputs are applied to the WLs or CTT gates. A $V_{GS} = 200mV$ is applied to selected WLs or non-zero inputs and all non-active WLs or zero inputs are set to $V_{GS} = -300mV$.

The ideal programmed weight range for an RVT device with W=428nm, L=20nm, $V_{GS} = 200mV, V_{DS} = 200mV$ is $100-700nA$ while the leakage current for *unselected* devices (e.g. $V_{GS} = -300mV$, $V_{DS} = 200mV$) has been measured to be $<10pA$ assuming no subthreshold degradation.)



(a) VPP (IN_H) (b) VWL (IN_L) (c) VREAD (IN_READ_B)

Figure 3.6: **WL Driver Level Shifters.** Three level-shifter circuits are utilized to level-shift differential logic-defined ($0/0.8V$) signals to generate IN_H, IN_L, and IN_READ_B signals for the WL Driver Output Stage without affecting its reliability.

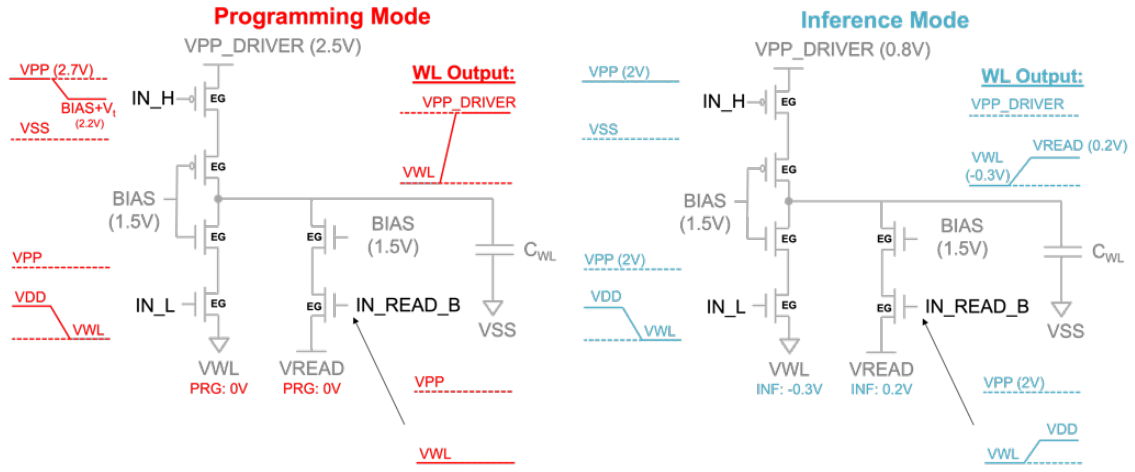
3.1.3 WL Driver Design

The WL Driver is designed to support several modes of chip operation including programming (e.g. $1.5{-}2.7V$), block erase ($-1V$), and inference ($50{-}200mV$) conditions. The design consists of 24 standard-gate (SG) devices and 6 thick-oxide (EG) devices in order to support maximal programming voltages of up to $2.7V$ without subjecting any of the SG devices to reliability issues (e.g. voltages beyond $1.2V$) using techniques similar to those discussed in [RBK12]. Figure 3.6 describes each of the three 8T level-shifter circuits designed to control the main output driver. The VPP level shifter converts $0V{\rightarrow}VPP$ (e.g. $0V{\rightarrow}2.7V$) and $0.8V \rightarrow VPP\_BIAS{+}V_{TH_P}$ (e.g. $0.8V{\rightarrow}2.2V$). The VWL level-shifter converts $0V{\rightarrow}VDD$ (e.g. $0.8V$) and $0.8V{\rightarrow}VWL$ (e.g.$-0.3V$), while the VREAD level-shifter converts $0V{\rightarrow}VWL$ (e.g. $-0.3V$) and $0.8V{\rightarrow}VDD$ (e.g. $0.8V$).

Figure 3.7 describes the 6T Output Stage with example operating conditions for programming and inference modes. Table 3.1 provide example logic inputs for each of the three level-shifters to provide the specified output (last-column).

Mode	WL status	VPP_EN⟨∗⟩	VWL_EN⟨∗⟩	VREAD_EN⟨∗⟩	Output
PRG	*selected*	1	1	0	VPP ($2.7V$)
	unselected	0	0	0	VWL ($0V$)
VER	*selected*	0	1	1	VREAD ($0.2V$)
INF	*unselected*	0	0	0	VWL ($-0.3V$)
ERS	*selected*	0	0	0	VWL ($-1V$)
	unselected	0	1	1	VREAD ($0V$)

Table 3.1: **WL Driver Control Logic.** Example Output Driver voltages for each condition are provided in the right-most column.



(a) Programming (PRG) (b) Inference (INF)

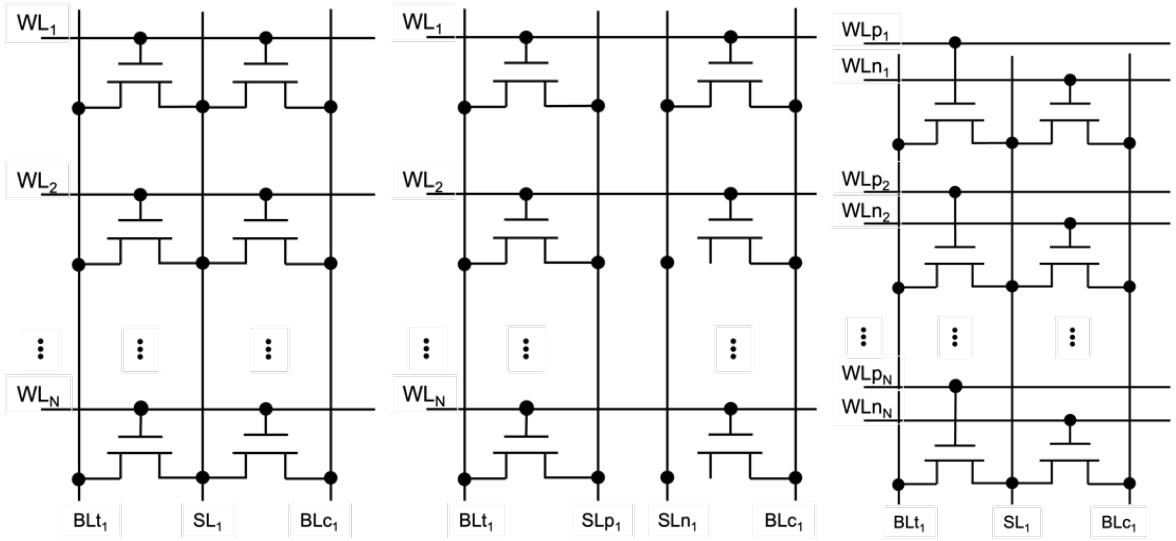
Figure 3.7: **WL Driver Output Driver Stage during Programming (PRG) & Inference (INF).** Three level-shifter circuits, shown in Fig. 3.6, are utilized to level-shift differential logic-defined ($0/0.8V$) signals to IN_H, IN_L, and IN_READ_B signals. Output Driver consists of thick-oxide (EG) devices to support programming voltages up to $2.7V$.

3.1.4 CTT Array Design

Cell area, programming efficiency, retention, device variability, half-select issues during programming, inference accuracy, and several other factors were taken into consideration when designing the twin-cell CTT array.

In this particular design, the cell area was not dominated by the twin-cell CTT device itself. Rather, it was limited by (1) the minimum pitch of peripheral circuitry and (2) the BL/SL routing which was designed to support $\sim mA$ programming currents during PRG modes of operation. The cell size was set to $0.52\mu m \times 3.6\mu m$ where $0.52\mu m$ is the minimum pitch of the EG-based WL Driver and $3.6\mu m$ is the designed height of each analog neuron circuit, laid out similarly to that shown in the conceptual diagram in Fig. 3.3. Cell size can be further reduced by a factor of $2\times$ by driving WLs from both (top and bottom) sides, reducing the minimum cell width to $260nm$. The cell height is mostly limited by the BL/SL routing. In order to ensure that CTT devices can be programmed efficiently (support $\sim mA$'s programming current), the BLt, SL, and BLc must have sufficiently low resistance. Using metal 8, these lines have been designed to have a resistance of $6.5-8.2\Omega$, neglecting tungsten device contact resistance. This limits the cell minimum height to $2.7\mu m$ ($W = 450nm, Pitch = 900nm$).

Leveraging empty space in between twin-cell CTT device columns, *as-fabricated* device variability was minimized by applying specific considerations during layout phase to include dummy devices to reduce lithography-related variability effects. The target cell, Regular V_{TH} (RVT) nfet device with W=428nm & L=20nm, was selected based on extensive device studies with a various threshold voltage (e.g. RVT, SLVT, etc.) and channel width device flavors. While the 428nm device was selected for the most recent NeuroCTT 0.3 tapeout, a 10×8 twin-cell array macros with 170nm devices was also included to evaluate the feasibility of using smaller channel width devices. Smaller devices reduce the subthreshold current range and hence improve the energy-efficiency of peripheral ADC circuitry such as the neu-



(a) Shared SL (b) Split SL (c) Split WL

Figure 3.8: **TWIN Cell Array Design.** Multiple array designs were considered providing area, half-select, and inference accuracy tradeoffs.

ron in this specific architecture, but at the potential cost of higher variability, worsened programming variance, and diminished charge retention.

While a Shared-SL (Fig. 3.8a) approach was utilized when designing the most recent array, Split-SL (Fig. 3.8b) and Split-WL (Fig. 3.8c) implementations were also considered. Split-SL offers an additional advantage of improved half-select during the erase mode at the expense of increased cell area and the design may also degrade inference accuracy as the the SLp and SLn are no longer set to an identical ground potential.

Programming and Erase mode half-select issues are reviewed by Figures 3.9 & 3.10, respectively, for the Shared-SL array configuration. Split-SL array design can eliminate half-select issues shown in Fig. 3.10 by using two separate SLs to apply separate bias conditions to both twin-cell devices.

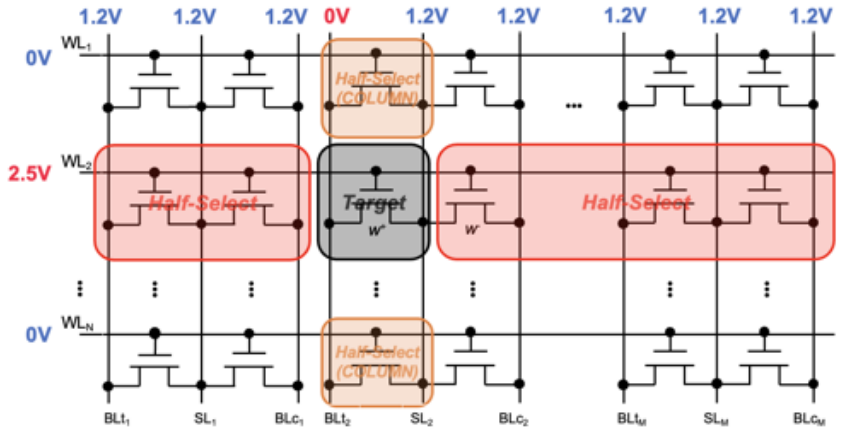


Figure 3.9: **TWIN-Cell Array: Programming (PRG) Half-Select.** Regular V_{TH} nfet devices are nominally rated for maximum $V_{GS}, V_{DS} = 0.96V$. While programming the target device ($\uparrow V_{TH}$), however, we apply higher voltage conditions to the device for short intervals of time (pulses$\sim 50-500\mu s$). These conditions can affect the threshold voltage of non-target cells on the same row (red) or column (orange).

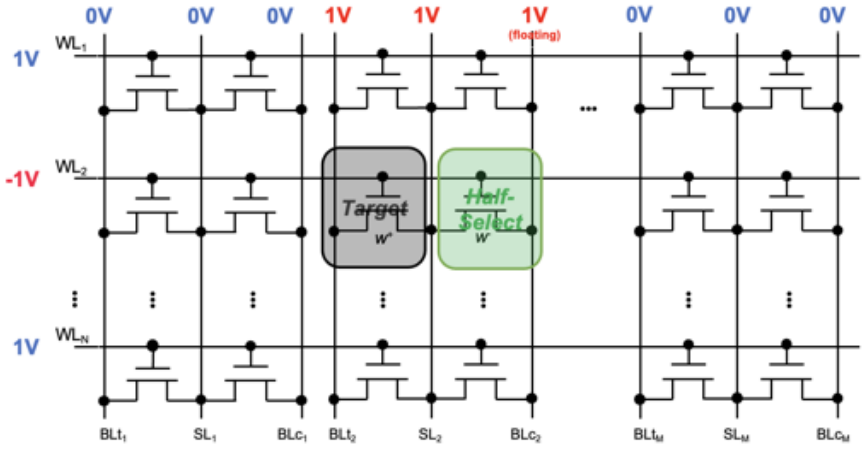


Figure 3.10: **TWIN-Cell Array: Erase (ERS) Half-Select.** Regular V_{TH} nfet devices are nominally rated for maximum $V_{GS}, V_{DS} = 0.96V$. Higher voltage conditions are applied to the device for short intervals of time (pulses$\sim 50-500\mu s$) while erasing the device ($\downarrow V_{TH}$). These conditions can affect the threshold voltage of the complement cell (green) as it is subject to similar voltage conditions and erasing does not require a large I_{DS} current.

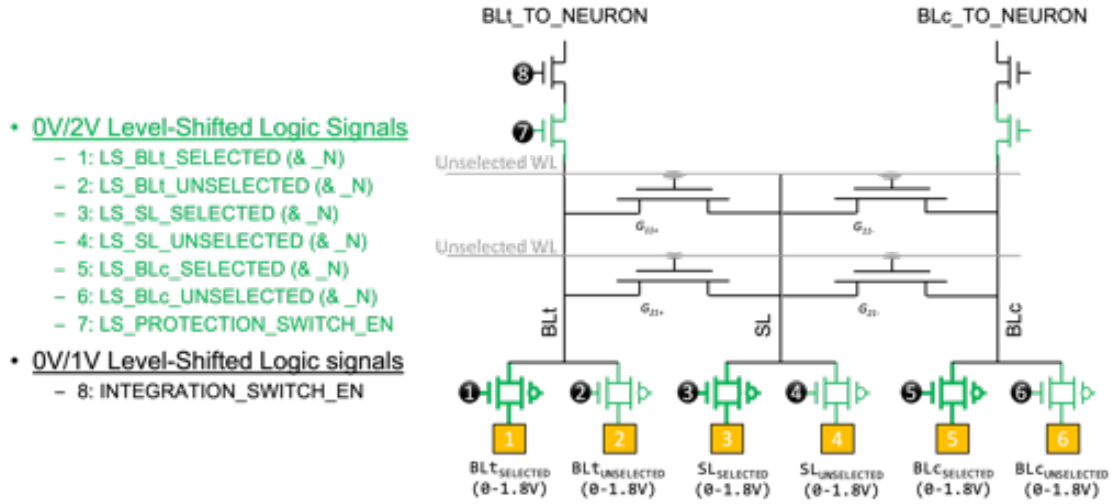


Figure 3.11: **Column Array Mux Design.**

3.1.5 CTT Array Mux & Level Shifter Design

The CTT Array is co-designed with an array mux which supports applying a variety of inference, verification, programming, and erase bias conditions to CTT devices within the array. During programming and erase modes, these mux devices can be enabled/disabled based on specified timing constraints loaded onto the chip, which allows for precisely tuned on-chip BL/SL voltage pulses to be applied to target devices while minimizing overall exposure time to higher voltage conditions to prevent reliability issues.

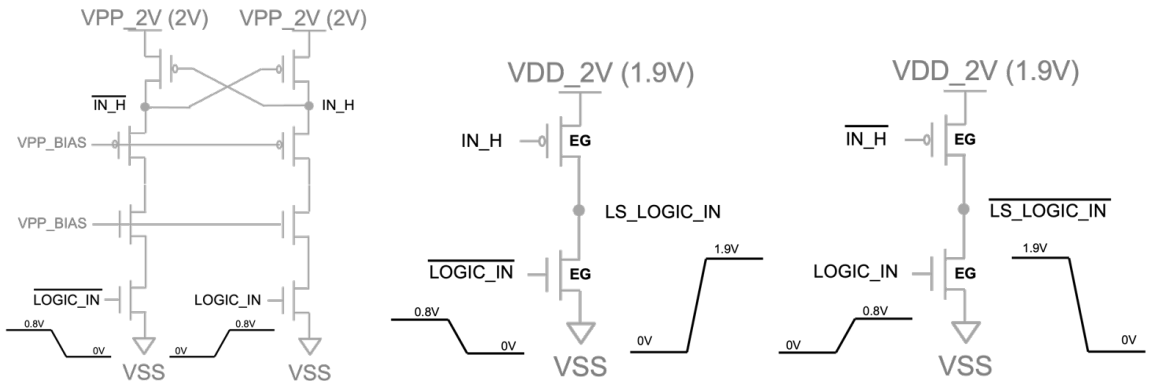
The Column Array Mux is shown in Fig. 3.11. The BLt, SL, and BLc are each connected to two external pads ($PAD\_BLt\_SELECTED, PAD\_BLt\_UNSELECTED$, etc.) via two separate mux's. These pads allow for two separate $0-1.8V$ voltages to be applied to the selected and unselected columns, respectively, depending on the mode of operation. In order to prevent stress on the array mux devices during PRG and ERS modes, thick-oxide (EG) devices are utilized. Programming mode (PRG) requires the *selected* BL/SL path to be fairly low-resistive as mentioned in the previous section, so the EG devices on the BL/SL selected paths are sized fairly large (e.g. $W{\sim}100\mu m$) and require level-shifted logic to properly turn on each of the EG mux devices.

Logic Signals ($0/0.8V$)	**Level-Shifted Logic** ($0/1.9V$)
BLt_SELECTED	LS_BLt_SELECTED
BLt_SELECTED_N	LS_BLt_SELECTED_N
BLt_UNSELECTED	LS_BLt_UNSELECTED
BLt_UNSELECTED_N	LS_BLt_UNSELECTED_N
SL_SELECTED	LS_SL_SELECTED
SL_SELECTED_N	LS_SL_SELECTED_N
SL_UNSELECTED	LS_SL_UNSELECTED
SL_UNSELECTED_N	LS_SL_UNSELECTED_N
BLc_SELECTED	LS_BLc_SELECTED
BLc_SELECTED_N	LS_BLc_SELECTED_N
BLc_UNSELECTED	LS_BLc_UNSELECTED
BLc_UNSELECTED_N	LS_BLc_UNSELECTED_N
PROTECTION_SWITCH_EN	LS_PROTECTION_SWITCH_EN

Table 3.2: **Column Array Mux Level-Shifted Logic Controls.**

On-chip logic generates necessary control logic signals to enable/disable each of the mux paths shown in Fig. 3.11. These control logic signals are level-shifted from $0/0.8V \rightarrow 0/1.9V$ using the level-shifter circuit shown in Fig. 3.12. The level-shifter shown in Fig. 3.12a first generates the IN_H and IN_H_N signals by converting the input signal from $0.8V \rightarrow VPP\_BIAS + V_{THP}$ (e.g. $1.2V$) and $0V \rightarrow 2V$. IN_H and IN_H_N signals are then used to enable/disable the output signal drivers shown in Figures 3.12b & Fig. 3.12c which generate level-shifted logic for the input signal and its complement, respectively.

While the chip is in the **IDLE** condition, the BLt and BLc mux's are deselected and the SL is set to $0V$ via the SL_UNSELECTED path, shown in Fig. 3.13. Additionally, the device is disconnected from the neuron circuit by turning off the the PROTECTION_SWITCH (⑦).



(a) VPP Level-Shifter (b) LS_LOGIC_IN (c) LS_LOGIC_IN_N

Figure 3.12: **Column Array Mux: Level-Shifted Logic Control Signals.**

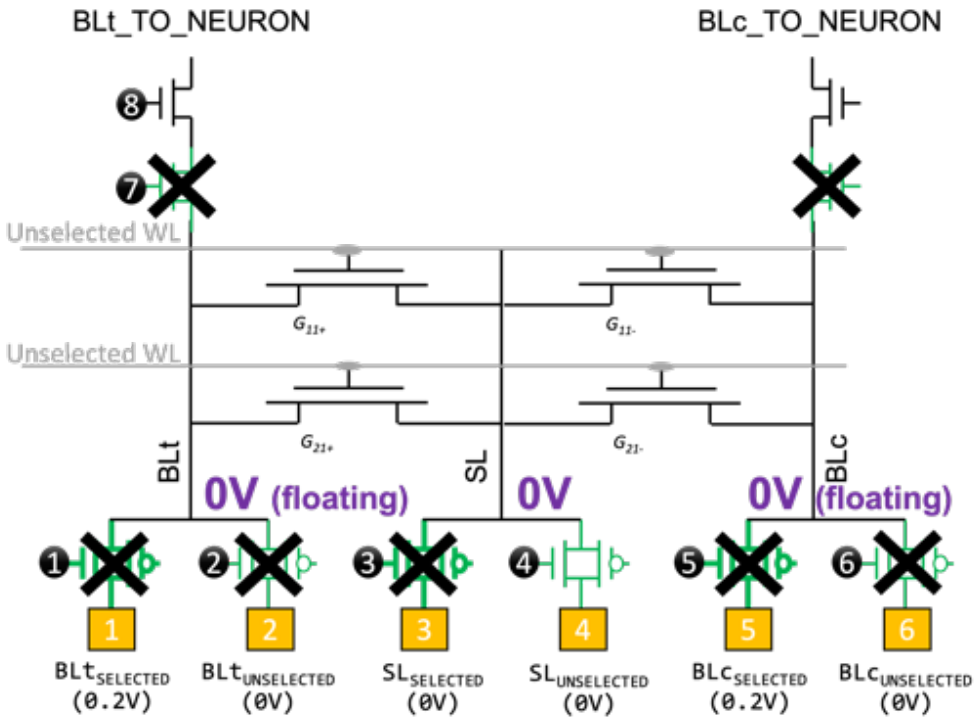


Figure 3.13: **Column Array Mux Design: IDLE Mode.**

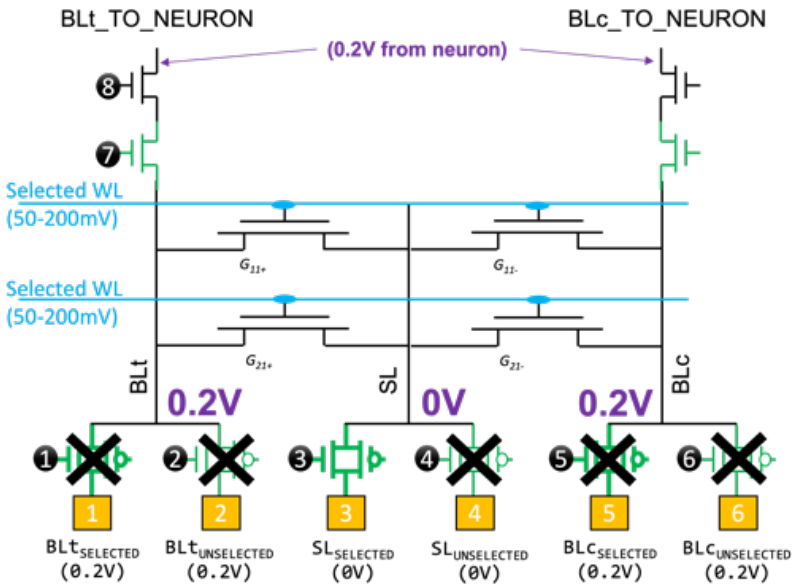


Figure 3.14: **Column Array Mux Design: INFERENCE Mode.**

During **INFERENCE** (Fig. 3.14), the switch mux stays in a configuration identical to that in **IDLE** mode, except the PROTECTION_SWITCH (⑦) is enabled. The neuron integrator provides a virtual supply ($V_D = \sim 200mV$) to the twin-cell CTT drains (BLt, BLc) and integrates the column differential current ($I_{BLt}(t) - I_{BLc}(t)$).

During normal **PROGRAMMING** mode of operation, pulse order and timing is taken into special consideration to reduce possible half-select issues. Figure 3.15 demonstrates programming operation on the True (T) device. First, the SL is raised high on all columns and the respective BLt/BLc for the target cell on the selected columns is brought to $0V$, providing a BL-SL pulse to the selected devices. Next, the selected WL(s) are raised to a large programming voltage (e.g. $V_{WL} = 1.5{-}2.5V$) for a specified duration of time.

Table 3.3 provides information on maximum pulse widths for the BL, SL, and WL pulses, specified at $f_{CLK} = 20MHz$. Actual logic implementations are chip programming configuration parameters are provided in Fig. 3.16, where all pulse timing information is derived from the SL_START (4*b*), BL_PULSE_WIDTH_X (16*b*), WL_PULSE_WIDTH (16*b*), BL_PULSE_WIDTH_Y (16*b*), & SL_END (4*b*) programming parameters.

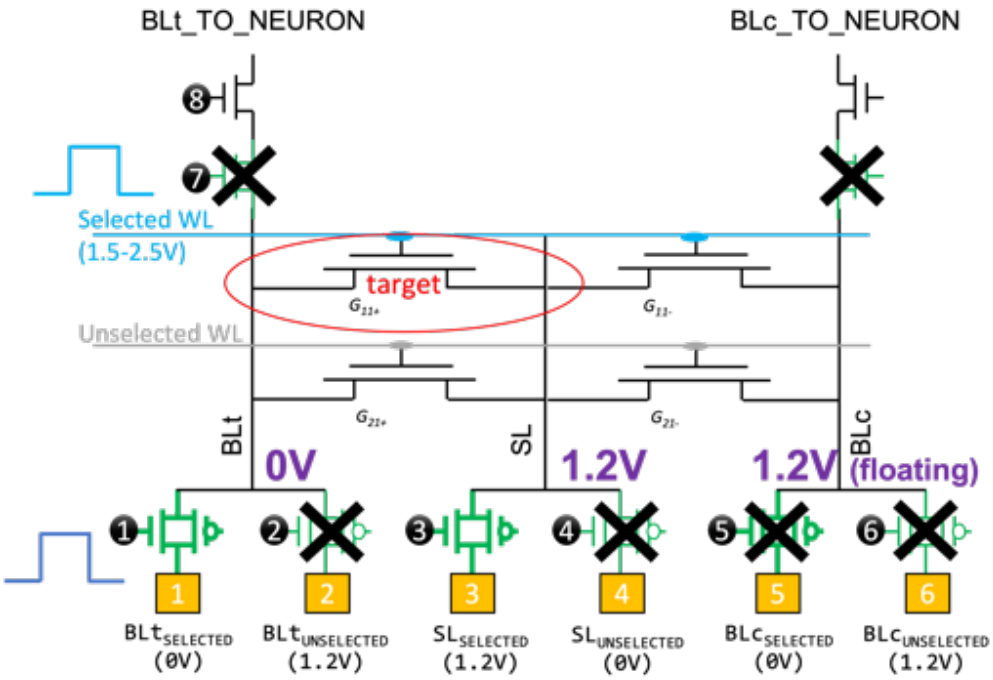


Figure 3.15: **Column Array Mux Design: PRG_T Mode.** SL_SELECTED enabled for all columns in order to reduce half-selected devices in other columns that are located on the SELECTED_WL by lowering the V_{GS} (e.g. $V_{SL} = 1.2V, V_{WL} = 2V \rightarrow V_{GS,unselected} = 0.8V$).

Parameter	MAX Pulse Width
SL_PULSE_WIDTH	$9.83175ms$
BL_PULSE_WIDTH	$9.83025ms$
WL_PULSE_WIDTH	$3.27680ms$

Table 3.3: **Programming (PRG) Pulse Timing Parameters.** Maximum pulse widths defined at $f_{CLK} = 20MHz$. Longer pulses can be applied by lowering system CLK frequency. Pulse widths derived from timing parameters shown in Fig. 3.16.

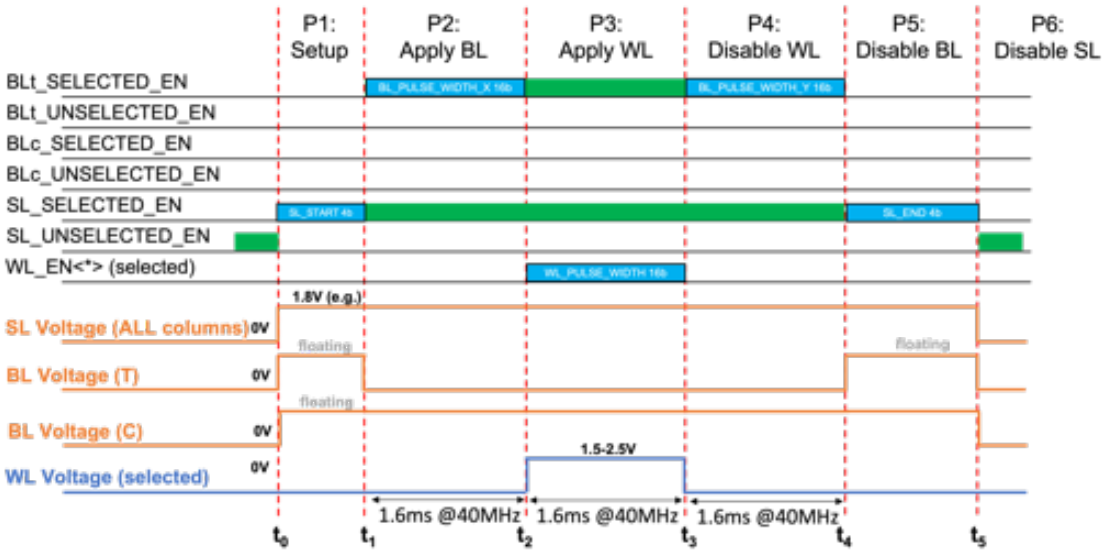
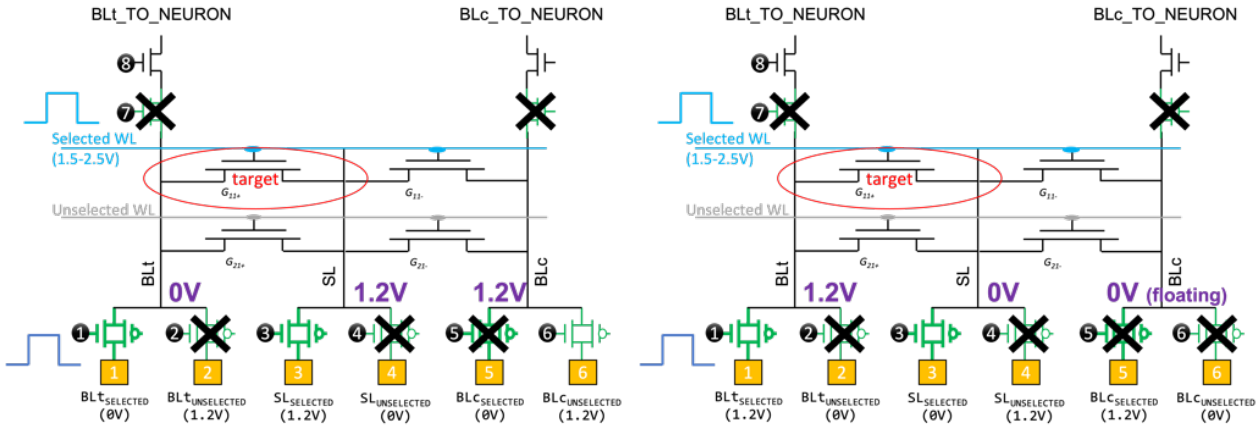


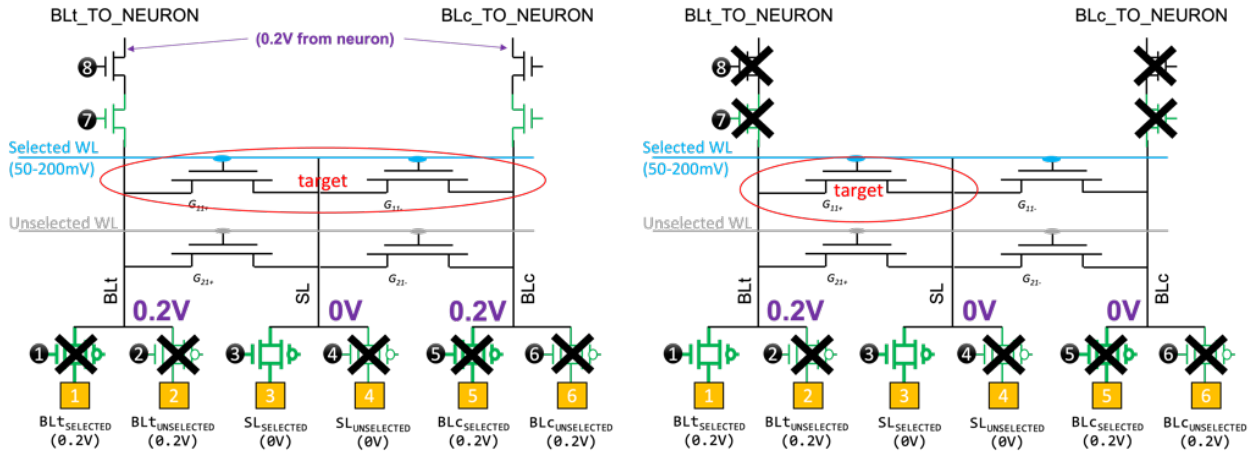
Figure 3.16: **Column Array Mux Design: PRG_T Timing Diagram.**



(a) PRG_T (No Float)

(b) PRG_T (Reverse)

Figure 3.17: **Column Array Mux Design: *Alternative* PRG_T Modes.** (a) The *No Float* configuration explicitly applies a voltage to unselected BLs (e.g. $1.2V$) rather than having them float, ideally to the SL voltage. (b) The *Reverse* configuration flips the BL and SL voltage polarities during programming, at the anticipated expense of worsened half-select issues for unselected devices in other columns connected to the SELECTED_WL.



(a) *On-Chip* Verification (Twin-Cell)

(b) *Off-chip* Verification (Single-Cell)

Figure 3.18: **Column Array Mux Design: Verification Modes.** (a) The *On-chip* verification mode utilizes the neuron circuit to measure a single twin-cell CTT differential current for a fixed duration in time. (b) The *Off-chip* verification mode allows the user to connect an Analyzer (Fig. 2.6b) to measure individual device currents accurately.

On-chip programming configuration parameters are further elaborated upon in Section 3.4.4, where the chip testing user interface is discussed. *On-Chip* and *Off-chip* verification schemes are also included in the design (Fig. 3.18. ***On-Chip* Verification** is similar to **INFERENCE** which utilizes the neuron circuit except typically only 1 twin-cell device is enabled or measured at a time for a fixed input duration. ***Off-Chip*** Verification, on the other hand, enables the BL_SELECTED and SL_SELECTED paths on selected column(s), which allow an external Analyzer to bias the device $V_{DS}{\sim}200mV$ and measure the subthreshold I_{DS}. A similar mode is included for ***Off-chip* INFERENCE** which allows multiple WLs to be selected and the column $(I_{BLt}(t) - I_{BLc}(t))$ to be measured externally.

Additionally, a variety of additional programming, inference, verification, and debug modes were also included. Figure 3.17 describes two additional programming modes (*No Float* and *Reverse*) that can be used for evaluating cell programming efficiency. Debug modes are included which allow all switch settings to be reconfigured after tapeout.

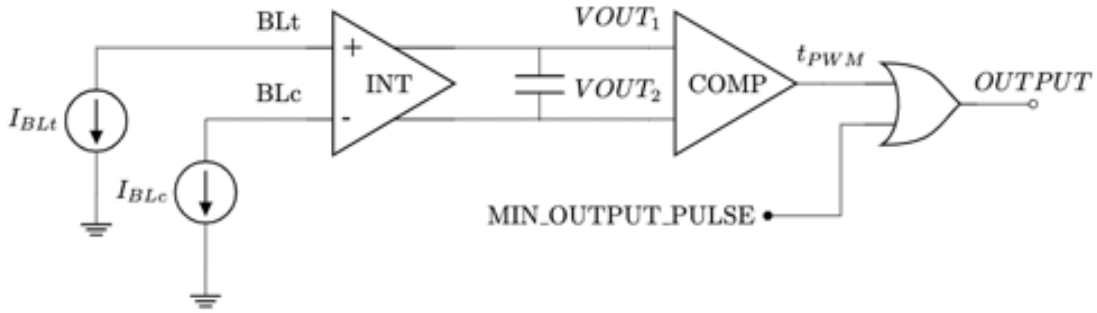


Figure 3.19: **Neuron Design Overview.** Design consists of a (1) differential current integrator, (2) comparator, and (3) logic or-gate.

3.1.6 Neuron Design

The neuron consists of three main components: an (1) *Integrator* tasked with computing the weighted sum, a (2) *Comparator* circuit which is used to convert the weighted sum (Q_{INF}) result into a pulse-width-modulated (t_{PWM}) output signal, and an (3) Or-gate—shown in Fig. 3.19. The pulse width-modulated output signal (t_{PWM}) is proportional to the weighted sum or accumulated charge on the C_{INF} capacitor during inference. A constant discharge current ($I_{DISCHARGE}$) is used to linearly discharge the output capacitor. The Comparator circuit calculates the amount of time it takes to fully discharge the capacitor (Eq. 3.1). An or-gate is included at the output to allow a logic-defined minimum output pulse for debugging purposes, specifically for when $t_{PWM} = 0$.

The Neuron Integrator (Fig. 3.20) integrates the differential current ($I_{BLt}(t) - I_{BLc}(t)$) only, requiring a 4× smaller capacitor than a traditional differential op-amp-based integrator configuration with two separate capacitors. It consists of a center gain-boosting stage which can be disabled. If the gain-boosting stage is disabled, an external bias voltage can be supplied to the CG amplifier. The circuit also includes a common-mode feedback loop designed to bias the BLt and BLc to $\sim 200mV$ and act as a virtual supply for the CTT array. The resultant differential current is integrated and stored on the C_{INF} until the integration

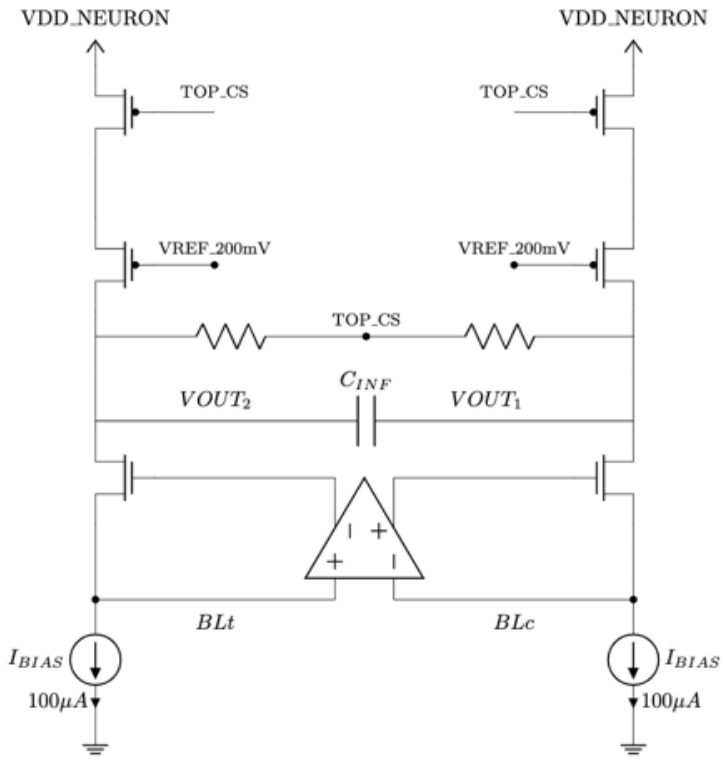
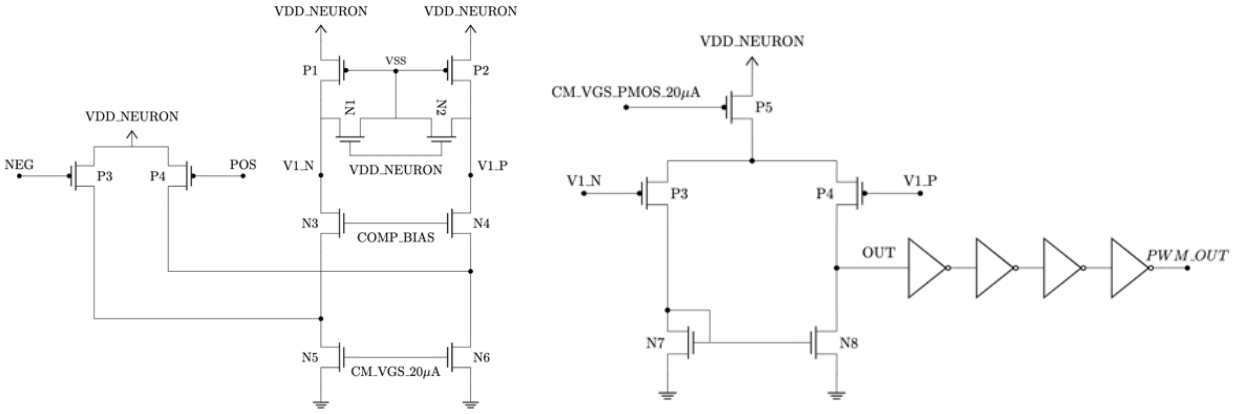


Figure 3.20: **Neuron Differential Integrator Design.** Design includes G_m gain-boosting stage (center op-amp) and common-mode feedback which provides a $200mV$ virtual supply for V_{BLt} & V_{BLc}. Gain-boosting stage can be disabled by providing a CG bias voltage (NO_INT_CG_VREF_EXT) and setting $USE\_BOOSTED\_INTEGRATOR = 1'b0$.



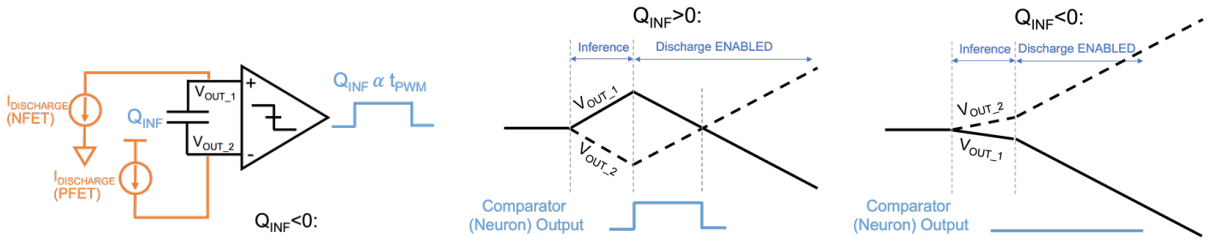
(a) 1^{st} Stage (b) 2^{nd} Stage

Figure 3.21: **Neuron Comparator Design.**

period has completed. After the integration (*INFERENCE*) period has completed, the *DISCHARGE* phase begins and the comparator circuit is activated.

The Comparator circuit consists of two stages followed by a chain of inverting buffers. The first stage consists of a PMOS-Input Folded Cascode Amplifier and the second stage consists of differential pair with active current source load and single-ended output, shown in Fig. 3.21. The output is then sent through a chain of inverters. Afterwards, The neuron's PWM_OUTPUT signal is sent to the Time-to-Digital Converter (TDC) block for digitization using a simple digital up-counter block. Special precautions were taken to prevent hold or setup violations since no timing constraints exist for the falling edge of the analog-valued pulse-width-modulated output signal.

During the *DISCHARGE* phase, constant current sources are connected to the two terminals ($VOUT_1, VOUT_2$) of the integrator's output capacitor, C_{INF}, and linearly discharge the capacitor, shown in Fig. 3.22a. $VOUT_1$ and $VOUT_2$ are connected to the comparator's input, and the comparator is tasked with accurately calculating the time it takes to fully discharge the capacitor. Fig. 3.22b and Fig. 3.22c demonstrate the comparator's output for positive and negative weighted sum results, respectively.



(a) Comparator (b) Positive Sums (c) Negative Sums

Figure 3.22: **Neuron Comparator Circuit Evaluating PWM Output.**

The Comparator and Discharge current source design realizes the Rectified Linear Unit (ReLU) Activation function where the neuron's pulse-width modulated output is zero for all negative weighted sum results, and proportional to the weighted sum for positive weighted sum results. The ReLU Activation function has been demonstrated to be an excellent candidate for classification or activation functions in deep neural networks [Aga18]. While the ReLU activation is implemented in this design, it is feasible to consider other activation functions in future designs. The implemented ReLU activation is summarized in Fig. 3.23.

Neuron Logic Control is divided into six phases: (1) Turn-on, (2) Setup, (3) Inference, (4) Discharge, (5) Reset, and (6) Turn-off, as shown in Fig. 3.24. All neuron logic control phases are derived from the chip's `INTEGRATOR_EN_X` (9b), `INTEGRATOR_SETUP` (9b), `INFERENCE_DURATION` (10b), `DISCHARGE` (10b), `RESET_Y` (8b), and `INT_EN_Y` (4b) configuration parameters. Additionally timing parameters are available to tune various other neuron logic signals externally. The neuron performs integration during the 3^{rd} phase ('INTEGRATION') on the differential input current to the integrator, $I_{BLt}(t) - I_{BLc}(t)$, presented by CTT array. The neuron's pulse-width modulated output is produced during the 4^{th} phase ('DISCHARGE') where the comparator is utilized to evaluate the time it takes to fully discharge the integrator's output capacitance, C_{INF}.

Given the typical sparsity of neural networks, a majority of weights tend to be zero or close to zero. A zero weight can be implemented using a twin-cell CTT device such that

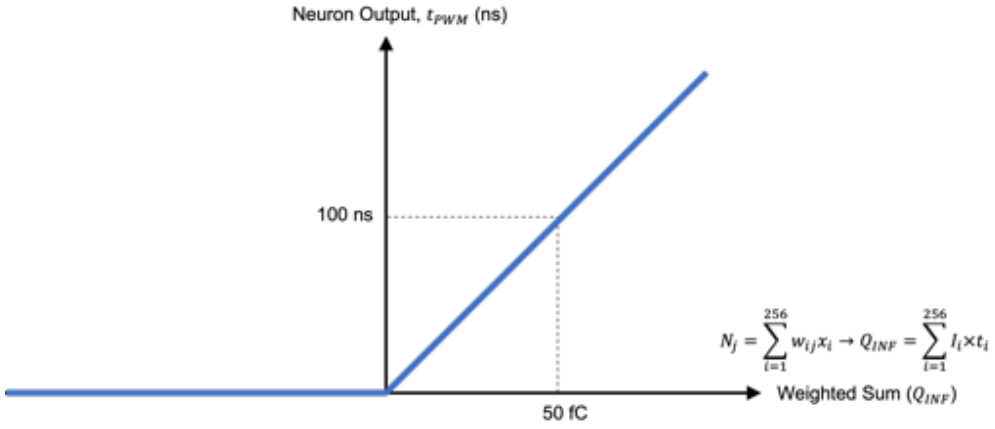


Figure 3.23: **Neuron Comparator Realized (ReLU) Activation Function.** The Rectified Linear Unit (ReLU) Activation function is implemented using the comparator to evaluate the time it takes for the neuron to fully discharge the C_{INF} capacitor, where positive sums result in a non-zero PWM output (Fig. 3.22b) and all negative sums result in a zero output (Fig. 3.22c). $I_{DISCHARGE} = 500nA$ is assumed.

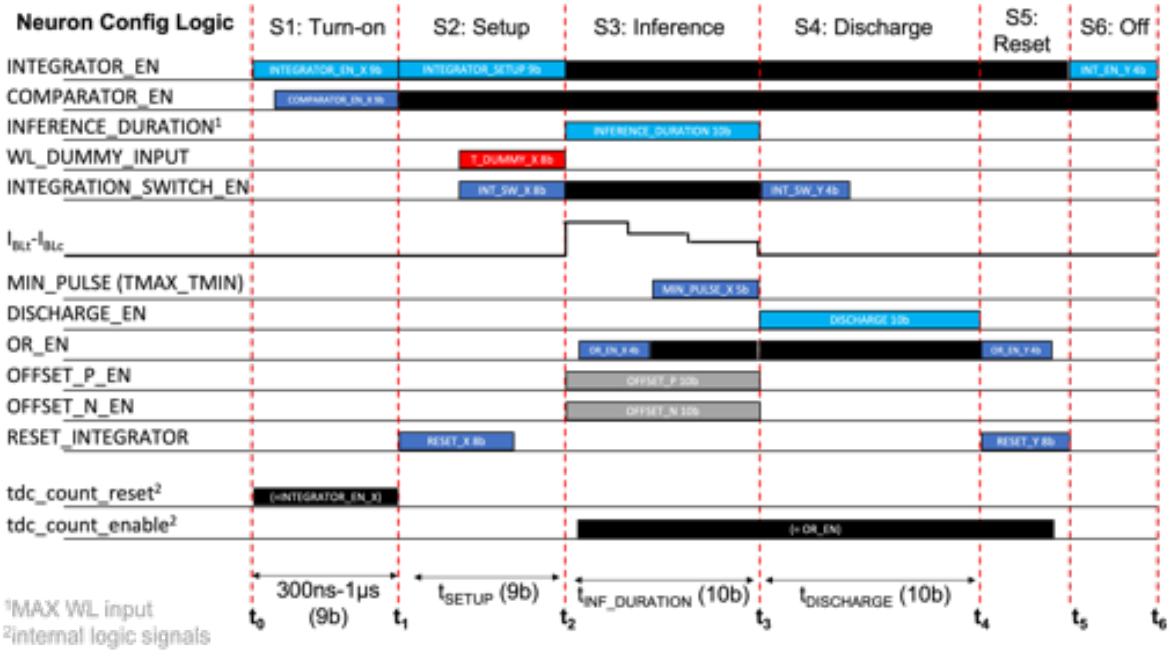


Figure 3.24: **Neuron Configuration Logic & Timing Diagram.** Compute consists of 6 stages. The duration of each stage is derived from the neuron configuration parameters.

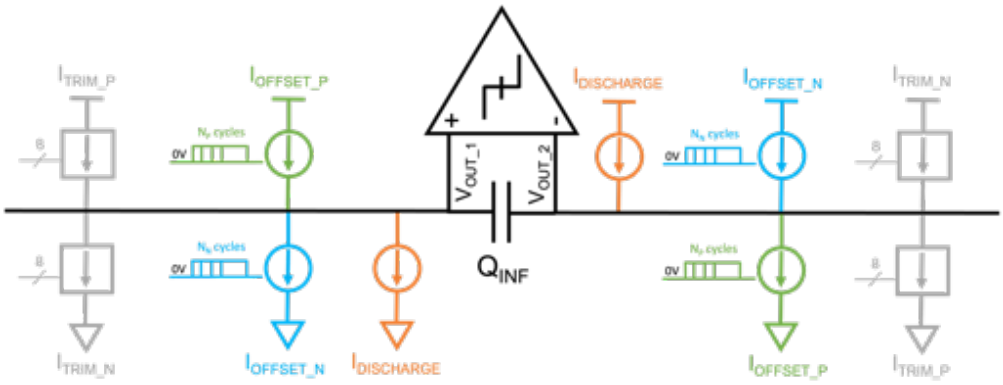


Figure 3.25: **Neuron Integrator Offset Cancellation Schemes.** The integrator design includes several positive and negative offset current sources and 8-bit trimming circuits to negate any mismatch-induced offsets. I_{OFFSET_P} & I_{OFFSET_N} are optional dc sources applied for N_P & N_N cycles, respectively. Trimming circuits are included for providing a positive or negative dc offset current with $8\,b$ resolution.

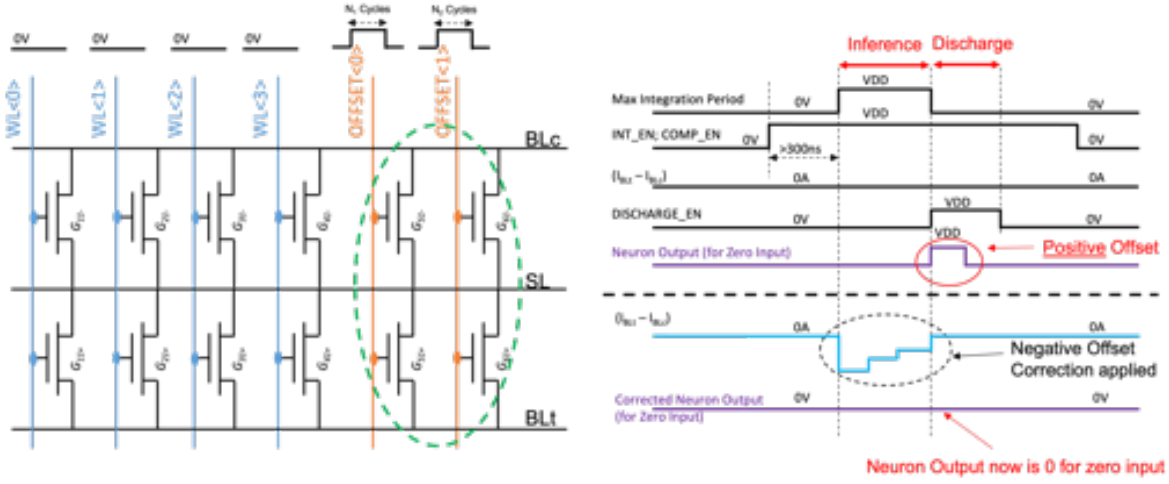


Figure 3.26: **Additional Offset Cancellation Using Extra CTT Devices.** Extra rows of CTT devices can be programmed to provide a specific weight and each WL can be enabled for a certain number of clock cycles. Applied offset of $\Delta Q = (\frac{N_1}{f}) \times G_{51} \times V_{BL} + (\frac{N_2}{f}) \times G_{61} \times V_{BL}$ is shown in provided example. It is, however, important to mention that this approach would not solve any input-dependent offsets.

the differential on-current or weight is ∼0; however, the common-mode current might be reasonably large. This presents an interesting problem to the differential integrator as it must be able to handle large common-mode currents, I_{CM}. In fact, in some instances the common-mode to differential mode (I_{CM}/I_{DM}) ratio can be as large as 1000×. The integrator has been designed to handle these scenarios; however, offset compensation schemes or post-fabrication calibration are required due to process-induced mismatches in the differential integrator mainly driven by the design of the bias current sources. In other words, the integrator might integrate mismatch-induced offsets or currents within the design which may lead to the neuron output saturating even in the zero-input case. These mismatch-induced offsets can be counteracted using a variety of calibration schemes which are reviewed at a high-level in Fig. 3.25.

The main offset correction scheme consists of 8-bit trimming current sources (I_{TRIM_P} & I_{TRIM_N}) which allow a positive or negative dc current to be supplied to counteract any dc-offsets at the output of the integrator. These sources have been designed to apply a dc offset of up to $\pm 1.6\mu A$ (nominally) with 8-bit precision. The second option involves a set current sources (I_{OFFSET_P} & I_{OFFSET_N}) which can be enabled for a specified number of cycles (N_P & N_N, respectively), to apply a fixed offset charge at the output.

$$\Delta Q_{applied} = I_{OFFSET\_P} \times N_P - I_{OFFSET\_N} \times N_N \tag{3.4}$$

Equation 3.4 describes this fixed offset charge applied to the output. It is important to note that this method does not correct for input-dependent offsets. Additionally, extra WLs within the CTT array itself can also be used to applied a $\pm\Delta Q$ offset, described in Fig. 3.26.

Finally, the design also included a 'swapping mode' which allowed the integrator's bias sources to be swapped from side-to-side at regular time intervals (specified by the `T_SWAPPING` parameter) In theory, this allows for any mismatch-induced offsets to be averaged out, neglecting channel length modulation. For the sake of brevity, these options have been omitted from this dissertation.

Target ΔG_{ij}	**True ($G_{T_{ij}}$)**	**Comp. ($G_{C_{ij}}$)**
$+600nA$	$700nA$	$100nA$
$+500nA$	$600nA$	$100nA$
$+400nA$	$500nA$	$100nA$
$+300nA$	$400nA$	$100nA$
$+200nA$	$300nA$	$100nA$
$+100nA$	$200nA$	$100nA$
$+0nA$	$100nA$	$100nA$
$-100nA$	$100nA$	$200nA$
$-200nA$	$100nA$	$300nA$
$-300nA$	$100nA$	$400nA$
$-400nA$	$100nA$	$500nA$
$-500nA$	$100nA$	$600nA$
$-600nA$	$100nA$	$700nA$

Table 3.4: **Example CTT Weight Mapping with 13 Target Differential Currents.**

3.1.7 Trained Network Layer Mapping to CTT Array

Efficiently mapping digitally trained networks weights to CTT devices ($\Delta w_{ij} \rightarrow \Delta G_{ij}$) is important as it has several accuracy, performance, and retention implications. Recent work as well as past work shown in [Wan20a, GWI19, Gu18] demonstrate that a differential twin-cell CTT (nfet, RVT, W=428nm) biased at $V_{GS} = \sim 200mV$ and $V_{DS} = \sim 200mV$ can realize a usable range after programming of $[-600nA, 600nA]$. As an example, 13 differential weight states ($\sim 5b$) can be created with 100nA spacing, as shown in Table 3.4. Additionally, rather than individually programming CTT devices, twin-cell weights can also be programmed such that it reaches the target differential current. The actual $G_{T_{ij}}$ & $G_{C_{ij}}$ weight values can be arbitrary (e.g. $G_{T_{ij}} = 347nA, G_{C_{ij}} = 148nA \rightarrow \Delta G_{ij} = 199nA$).

Analog Bi-Scale (ABS) Weight Representation:

An Analog Bi-Scale (ABS) weight representation, first reported in [WWZ22], is utilized to map weights of each layer to maximum CTT device conductance range. Mapping is first performed on a layer by layer basis, where each a scaling coefficient, β, is used to map the trained network weights to the usable range, $[-600nA, 600nA]$. The absolute maximum weight ($w_{absmax} = \max_{i,j} |w_{ij}|$) is used to determine the appropriate β scaling coefficient, where $\beta = \frac{G_{max}}{w_{absmax}}$. For any two arbitrary network layers a & b, $\beta_a \neq \beta_b$.

Since $\beta_a \neq \beta_b$, a secondary scaling can be performed at the output such that the two network layers are equivalently scaled. First, let's consider the pulse-width modulated output of each layer as proportional to the accumulated charge during integration, Q_{INF}.

$$Q_{INF} \propto t_{PWM} \tag{3.5}$$

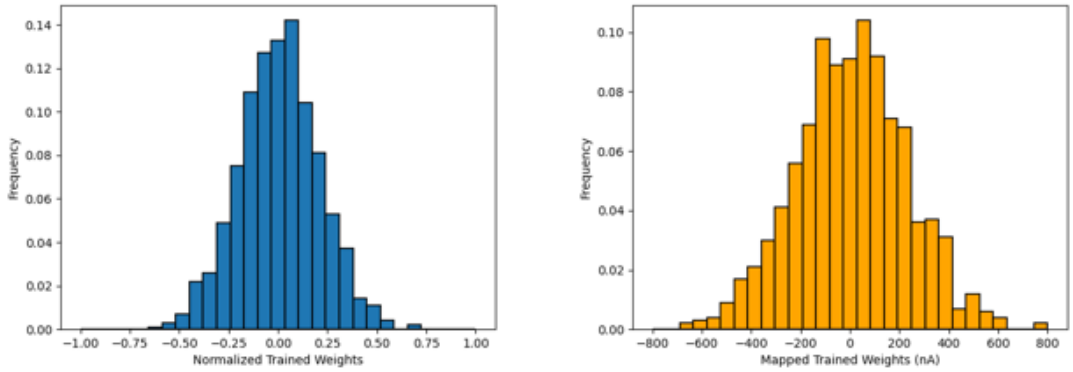
A DC discharge current source is utilized to convert the accumulated charge, $Q_{INF} \rightarrow t_{PWM}$, where t_{PWM} is the time it takes to fully discharge the integrator's output capacitance using the DC source:

$$t_{PWM} = \frac{Q_{INF}}{I_{DISCHARGE}} \tag{3.6}$$

For two layers a & b with the same weights and pulse-width modulated inputs, the outputs in terms of charge for each layer would be $Q_a = V_{ON} G_a t_{in}$ & $Q_b = V_{ON} G_b t_{in}$, respectively. In order two scale these two such that $t_{PWM_a} = t_{PWM_b}$, the discharge currents for each layer must be selected appropriately:

$$\frac{\beta_a}{I_{DISCHARGE_a}} = \frac{\beta_b}{I_{DISCHARGE_b}} \tag{3.7}$$

This implies that each network layer may be able to utilize a different discharge current, but actual implementations might limit the flexibility of this nob due to design limitations. A simplified design might restrict each network's scaling coefficients such that $\beta_a = \beta_b = \beta$ and all networks can utilize the same discharge current reference for compute.



(a) Normalized Weights (w_{ij}) (b) Mapped Weights (ΔG_{ij})

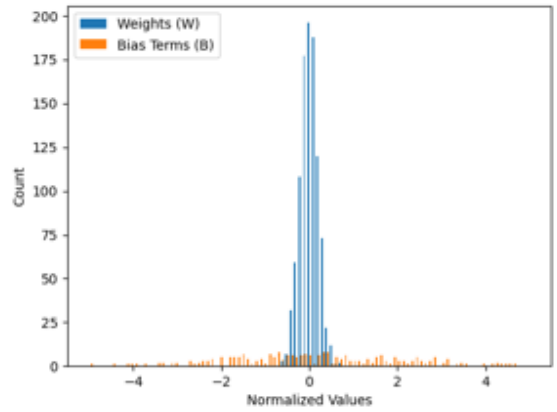
Figure 3.27: **Example Weight Mapping for Normally Distributed Weights.**

Bias Term (b_j) Implementation for PWM Architectures:

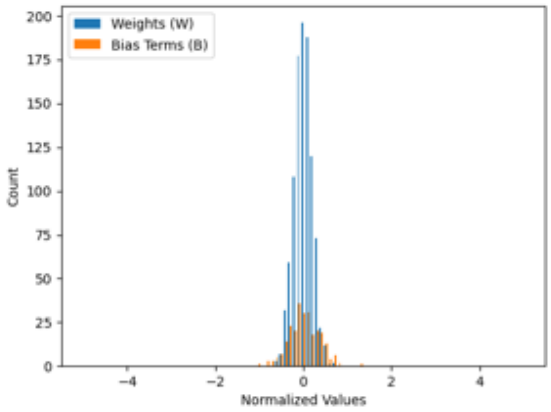
The layer bias terms can be combined with the weights, but a scaling typically must first be applied, because the bias term (b_j) is typically much larger than the range of the trained weights. By mapping the bias terms to extra row(s) of CTT device(s), a scaling factor ($S \geq 1$) can be mapped to a constant input applied to the *bias* rows for every input frame. In other words, the bias-term is remapped into a fixed amount of charge after integration, $b_j \rightarrow Q_{bias_j}$. This input-independent offset charge can be generated by programming a CTT cell to a specific differential weight value (ΔG_{bias_j}) and by setting the input to this row to some constant value for all input frames, (e.g. $Q_{bias_j} = V_{ON} \Delta G_{bias_j} t_S$, where $t_s \propto S$). Example weight and bias term mapping to a single weight array is shown in Eq. 3.8.

$$y = xw^T + b = \begin{bmatrix} x_1 & x_2 & \dots & x_N & S \end{bmatrix} \cdot \begin{bmatrix} w_{11} & \cdots & w_{N1} & \frac{b_1}{S} \\ \vdots & \ddots & \vdots & \vdots \\ w_{1j} & \cdots & w_{Nj} & \frac{b_j}{S} \end{bmatrix}^T \tag{3.8}$$

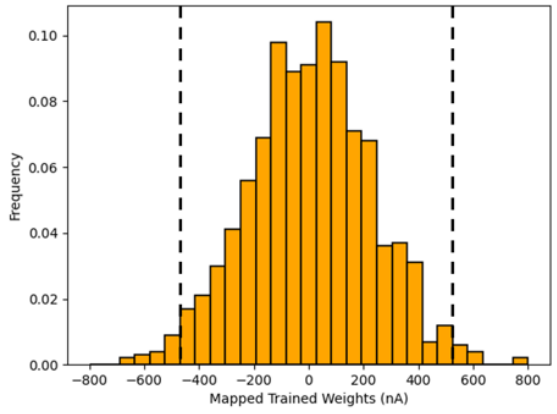
Concepts including *weight truncation* may also be worth exploring as it could expand the overall weight utilization of the entire range at the loss of truncating weights that likely have the largest affect on the network's output—example shown in Fig. 3.28d.



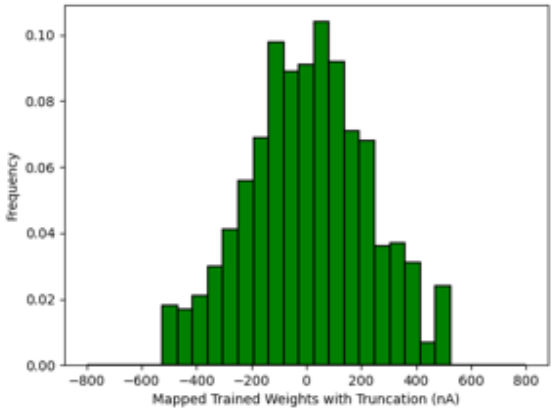
(a) Weights (w_{ij}) & Biases (b_j)



(b) Weights (w_{ij}) and Scaled Biases (b_j/S)



(c) Original Weights (G_{ij})



(d) With Truncation (G'_{ij})

Figure 3.28: **Example Bias Term Scaling & Weight Truncation.** (a) & (b) show example Bias Term Scaling with S=6. (c) & (d) show a separate example of weight truncation as a possible approach for mapping trained weights to devices (e.g. $w_{ij} \rightarrow G_{ij}$).

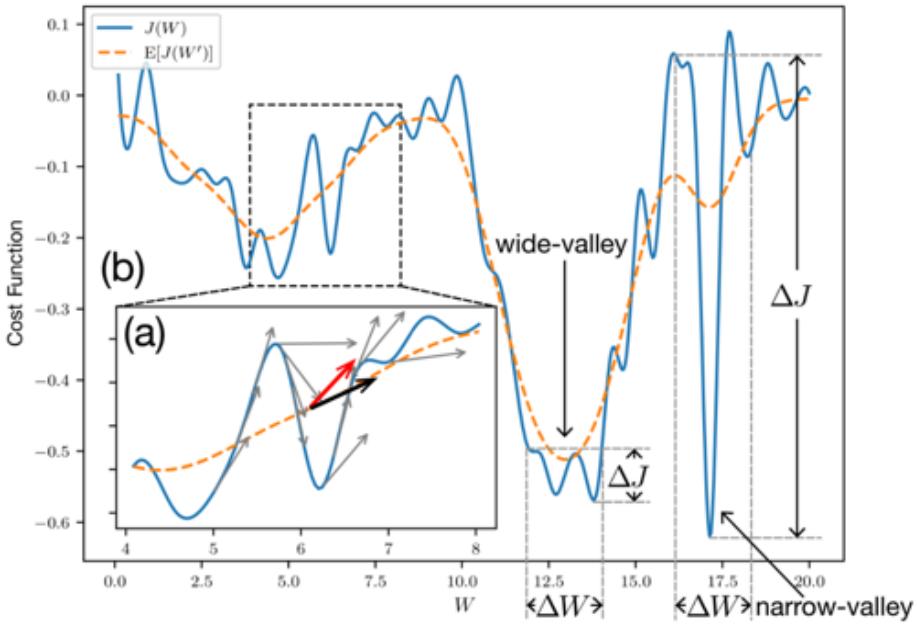


Figure 3.29: **Hessian-Aware Stochastic Gradient Descent (HA-SGD).** Given that analog networks deploy weights that are similar but not identical to the actual digitally trained weights, it is important to consider this programming variance (σ_{PRG}) or weight error (ΔW) during the training itself. HA-SGD training algorithm smoothens the gradient to prevent small weight errors during deployment from leading to substantial accuracy degradation. [WWZ22]

3.1.8 Training Networks Considering Analog IMC Implementations

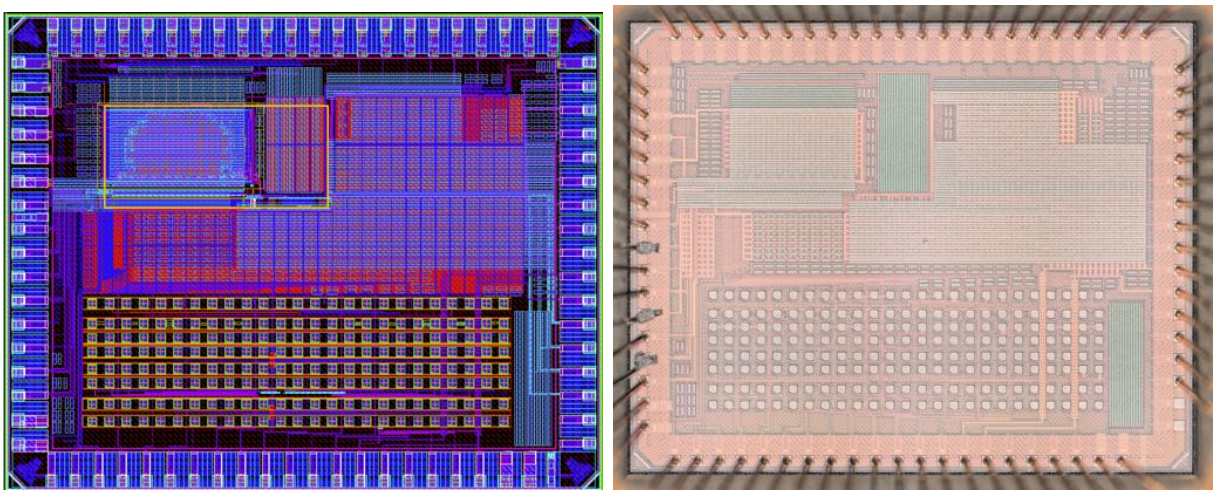
A Hessian-Aware Stochastic Gradient Descent (HA-SGD) algorithm is utilized to train neural networks at the software-level to consider inherent variabilities introduced when deploying trained networks to error-prone analog NVM devices. By considering the deployed weight error or programming variance (σ_{PRG}) as previously defined, the analog-resiliency of the network can be dramatically improved. Details of the HA-SGD approach are provided in [WWZ22].

3.2 Chip Design Efforts

Three designs—NeuroCTT_0.1 (*ZION*), NeuroCTT_0.2 (*GLACIER*), & NeuroCTT_0.3 (*DENALI*)—were taped out using Global Foundries 22FDX technology (MPWs 2219, 2229, & 2242, respectively), shown in Figures 3.30, 3.31, and 3.33. NeuroCTT_0.1 and NeuroCTT_0.3 were designed with $100\mu m$ pitch wirebond pads. NeuroCTT_0.2 was designed as a more complex chip with $100\mu m$ pitch ($50\mu m$ diameter) Cu-pillar flip-chip technology. Two versions of NeuroCTT_0.2 chip were fabricated: (1) *FLIP-CHIP* & (2) *Si-IF* versions. The *FLIP-CHIP* version of the die was fully-processed including FBEOL (e.g. Cu-pillar deposition) for connectivity to the main NeuroCTT system. The *Si-IF* version of the die was wafer-pulled after metal 9 (M9, last Cu metal layer) and before Al deposition. This version of the chip includes $10\mu m$ pitch Cu pads for compatibility with the CHIPS Lab Si-IF process [BJP17, BJP18, JRN20]. Wafer-pulled dies were then bonded to a Si-IF using thermal compression bonding to connect to device, ring oscillator [NI20], and Si-IF test [JRN20] macros. Si-IF test macro validating Si-IF bonding yield and SuperCHIPS communication protocol and IO cells reported in [JRN20].

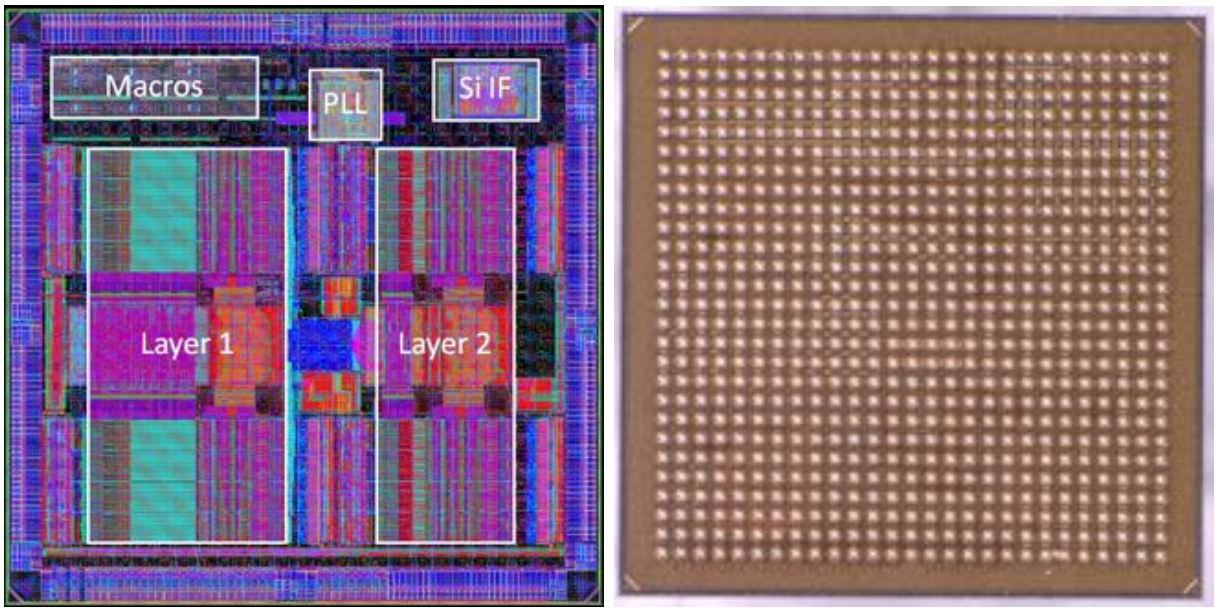
3.2.1 NeuroCTT 0.1 (*ZION*) Design

The first version design consists of 1024×10 CTT array with low-frequency (UHVT) logic. Design also include 5 standalone macros. Two of the macros were 1×25 Scribe Line Monitor (SLM) pad ($72\mu m$ pitch) discrete device macros while the remaining three macros were various CTT array macros. Each of the three array macros consist of two sets of 1×25 SLM pads designed in such a way such that half of the 10×8 twin-cell CTT array was accessible by probing either set of pads.



(a) Rendered Image (b) Actual Image

Figure 3.30: **NeuroCTT 0.1 (*Wirebond*) Die Images.** Designed for 1-layer (1024×10) fully-connected inference with up to $8b$ inputs at $200\,MHz$. Chip ($2\,mm \times 2.5\,mm$) was taped out in GlobalFoundries 22FDX technology with 77 wirebond pads at $100\mu m$ pitch. The chip also consists of 8 sets of 1×25 scribe line monitor pad (SLM) sets for device testing macros [Gu18, GWI19, Wan20b, Wan20a]. Highlighted area (gold) indicates active area of chip (e.g. logic, CTT array, neurons, etc.).

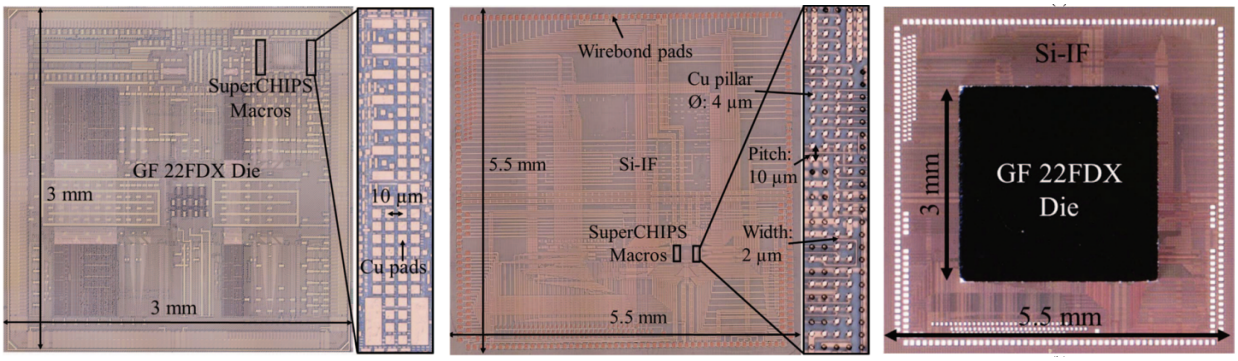


(a) Rendered Image (b) Actual Image

Figure 3.31: **NeuroCTT 0.2 (*Flip-Chip*) Die Images.** Designed for 2-layer fully-connected network inference (L1: 1024×256, L2: 256×256) with $8b$ inputs at $800\,MHz$. Chip ($3\,mm \times 3\,mm$) was taped out in GlobalFoundries 22FDX technology with 729 (27×27) $50\mu m$ Cu-pillar flip-chip pads at $100\mu m$ pitch. The chip also includes a set of fine-pitch *Si-IF enabled* device, ring-oscillator [NI20], and Si-IF test macros [JRN20].

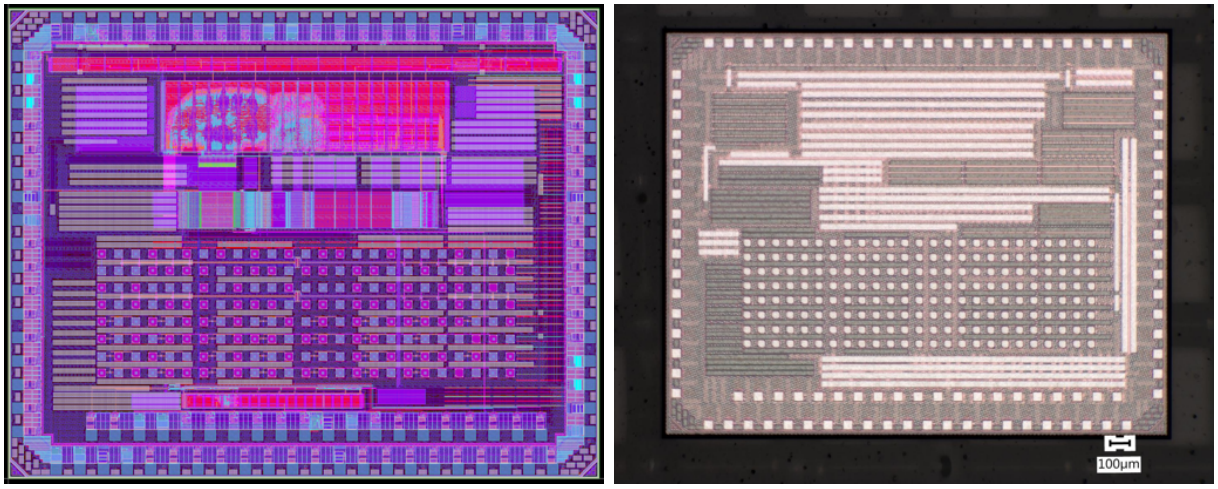
3.2.2 NeuroCTT 0.2 (*GLACIER*) Design

The second version design implemented a substantially more sophisticated network with 2 layers (L1: 1024×256, L2: 256×256). The outputs of the first layer were directly connected as inputs to the second layer, requiring no digital interface between layers. Additionally, a set of *Si-IF*-enabled macros were also included on the die. A subset of the dies were pulled after metal 9 (last Cu layer) in order to be diced and thermocompression-bonded to a Si-IF wafer, as shown in Fig. 3.32.



(a) Wafer-pulled Die Image (b) Si-IF Wafer Design (c) Si-IF Bonded Sample

Figure 3.32: **NeuroCTT 0.2 (*Si-IF*) Die Images.** NeuroCTT 0.2 Chip Design also included *Si-IF*-enabled macros, separate from the main flip-chip system. (a) A batch of dies were pulled wafer-pulled after the last Cu-metal layer (M9). The Si-IF wafer design in (b) serves as a *fanout* board for the IO Test, device, and ring oscillator macros with external wirebond and manual probe pads shown in both (b) & (c). Si-IF bonding results were previously reported in [JRN20] and shown in (c). Si-IF Test Macro designed to verify Si-IF bonding and demonstrate SuperCHIPS ([Jan17, NR22]) communication protocol & IO cells.



(a) Rendered Image (b) Actual Image

Figure 3.33: **NeuroCTT 0.3 (*Wirebond*) Die Images.** Chip ($2\,mm \times 2.5\,mm$) was taped out in GlobalFoundries 22FDX technology with 78 wirebond pads at $100\mu m$ pitch. The chip also consists of 8 sets of 1×25 scribe line monitor pad (SLM) sets for device testing and a 20-pad standalone neuron test macro.

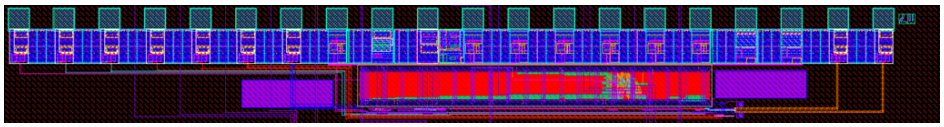


Figure 3.34: **NeuroCTT 0.3 Functional Neuron Macro.**

3.2.3 NeuroCTT 0.3 (*DENALI*) Design

The most recent design included a 256×32 system with Time-to-Digital Converted (TDC) neuron outputs. Six updated device macros were also included on the die:

1. MACRO_1_ARRAY_10x8_RVT_170nm
2. MACRO_2_ARRAY_10x8_RVT_428nm
3. MACRO_3_DISCRETE_RVT_NFET_L20nm
4. MACRO_4_DISCRETE_RVT_PFET_L20nm
5. MACRO_5_DISCRETE_EG_LVT_NFET_L150nm
6. MACRO_6_DISCRETE_EG_SLVT_NFET_L150nm

The first two macros are 10×8 *twin-cell* (160 total devices) CTT array macros can be probed using the in-lab Cascade probe station, Keysight B1500A Analyzer, and switch matrix. The array macros are accessed using two sets of 25×1 Scribe Line Monitor (SLM) pads at $80\mu m$ pitch.

Single-device macros were also included. Macros 3 & 4 each include 8 RVT nfet and pfet devices, respectively, with widths ranging $80nm - 5\mu m$ and $L = 20nm$. Macros 5 & 6 were included for thick oxide (EG) slvt and lvt device studies with widths ranging $160nm - 4.8\mu m$ and $L = 150nm$.

Additionally, a 20-pad functional neuron macro was included for directly debugging the neuron (differential integrator & comparator) design, shown in Fig. 3.34. This functional

neuron includes a standalone neuron and simplified logic control interface. Off-chip inputs can be directly applied to the integrator circuit and several probe points are setup for possible circuit biasing and debugging.

3.3 Testing Infrastructure

3.3.1 NeuroCTT 0.1 Infrastructure

The first version chip was bonded directly to a 84-pin C-QFN (Kyocera PB-C87729) with 77 wirebonds at $100\mu m$ pitch. A 12" $\times$ 12" testing PCB was designed with reusable Loranger C-QFN socket. Board design consisted of (1) a few manually-tuned LDOs to generate VDDIO ($3.3V$), VDDC ($0.8V$), and VDD_NEURON ($0.9V$) voltage domains, (2) logic buffer ICs between FPGA and packaged die, (3) various test points. Coaxial connections allowed the board to be hooked up directly to the Keysight B1500A Semiconductor Device Analyzer for accurate device current measurements before and after device programming. Full test setup is shown in Fig. 3.36 and consists of packaged die, PCB test board, Xilinx Artix 7 (AC701) FPGA Board, external power supplies, and PC computer. PC computer utilizes UART protocol to send/receive requests from FPGA controller. Additional Keysight B1500A and Analyzer PC were accessible for performing accurate off-chip current monitoring and device measurements pre- & post-programming.

The board as designed, however, suffered from several testing limitations. First of all, most voltages were externally & manually configured, requiring several DC power supply units to be frequently manually tuned. Secondly, the board was excessively large at 12" $\times$ 12" causing unnecessary trace parasitics on digital I/O paths. A combination of trace, socket, and packaging parasitics limited the data communication to/from the chip to $40\,MHz$. Additionally, the design lacked proper decap design (e.g. decap ranging $\sim 300pF - 100\mu F$). All of these insights were taken into consideration when designing the Mainboard for NeuroCTT 0.3, further discussed in Section 3.3.3.

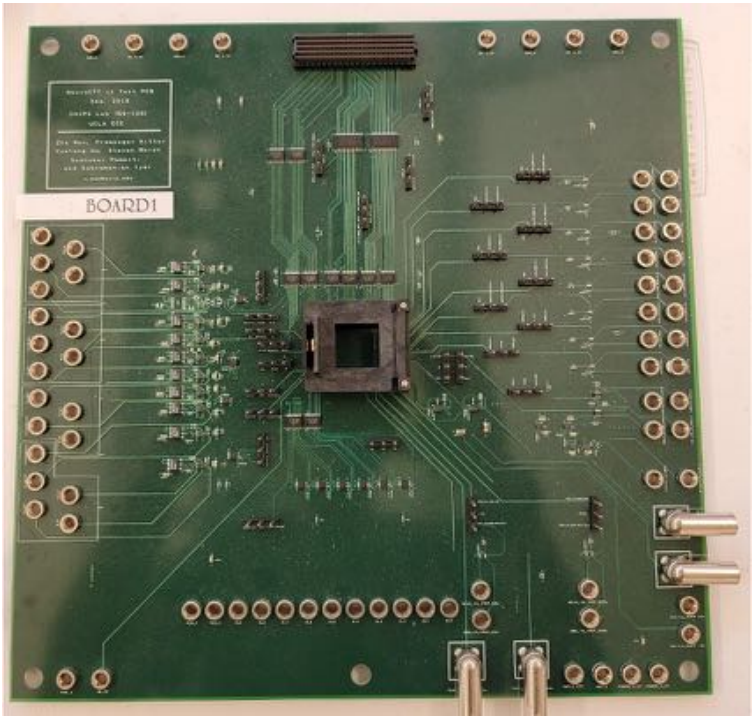


Figure 3.35: **NeuroCTT 0.1 Mainboard Design.** 12" × 12" Mainboard with C-QFN package Socket designed for use with Kyocera PB-C87729 84-pin C-QFN socket. Supplies are manually controlled via external supplies and banana connections.

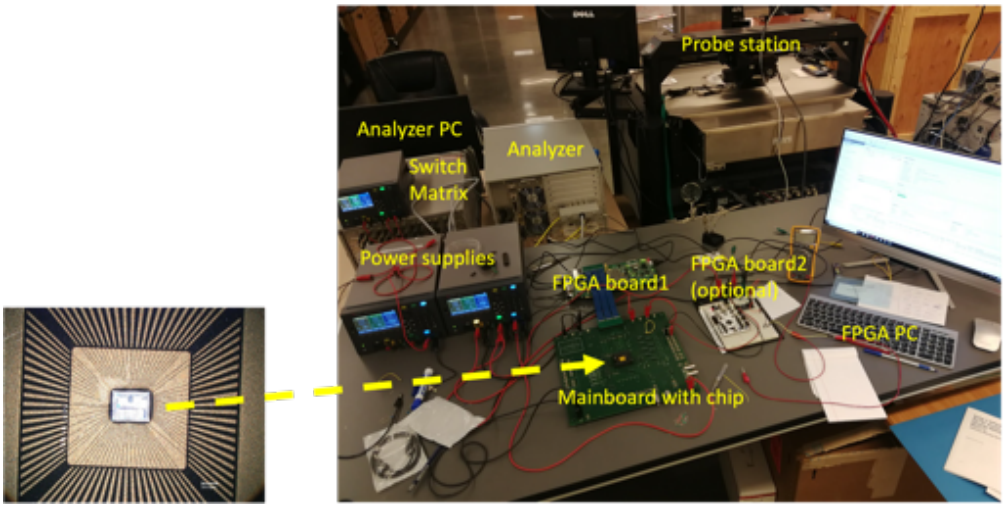
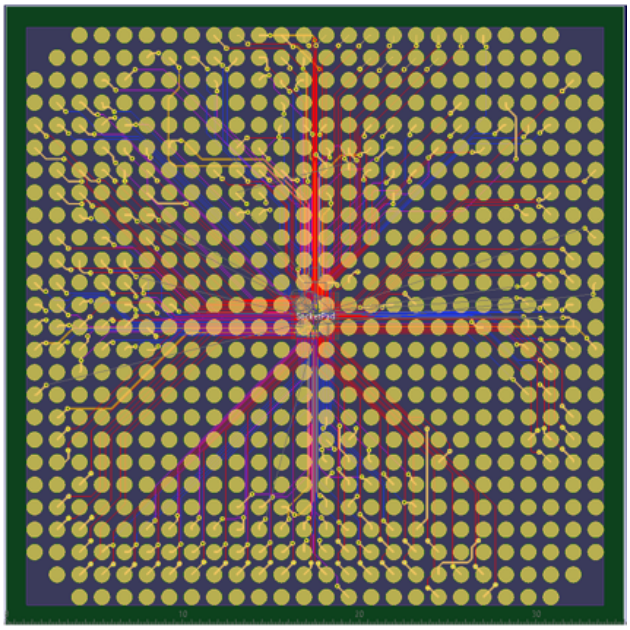


Figure 3.36: **NeuroCTT 0.1 Lab Test Setup.** Die bonded to 84-pin ceramic QFN package (*left*) then connected to Mainboard using Loranger C-QFN socket. Test setup includes external supplies, Xilinx Artix-7 FPGA AC701 Board, Analyzer, and PCs as well as Switch Matrix and Probe Station for probing discrete device macros.

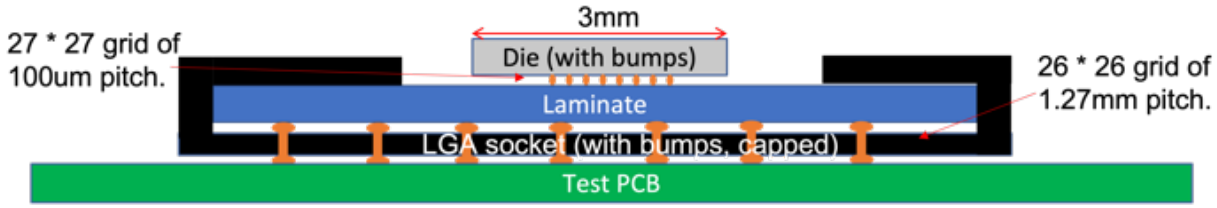
3.3.2 NeuroCTT 0.2 Infrastructure

NeuroCTT 0.2 IC was designed using a $100\mu m$ pitch Cu-pillar process, shown in Fig. 3.31. A 5-layer laminate was designed to fanout the 27×27 grid of $100\mu m$ pitch Cu-pillars to a 26×26 grid of $1.27mm$ pitch BGAs, shown in Fig. 3.37.

Due to *low-volume* vendor & manufacturing limitations, it was not possible to manufacture the required 5-layer laminate in order to package the chip using a traditional flip-chip packaging process. A *Si-IF*-enabled flip-chip package was conceived, inspired by previous *in-lab* work [BJP17, BJP18, JRN20]. The *Si-IF*-enabled package design, shown in Fig. 3.38, involves directly bonding the die to a Si-IF using $100\mu m$ pitch copper pads, then soldering the bonded Si-IF sample to a package laminate. The laminate is then connected to the testing PCB using $1.27mm$ pitch BGAs and an LGA socket.

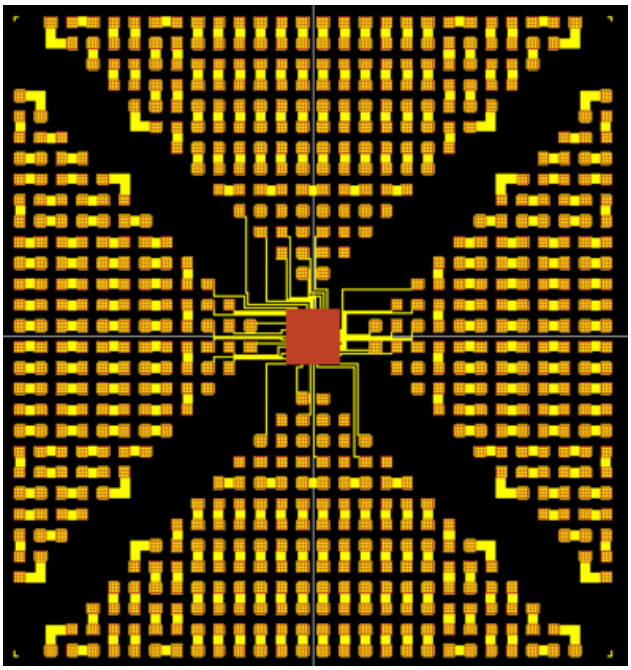


(a) Laminate Design

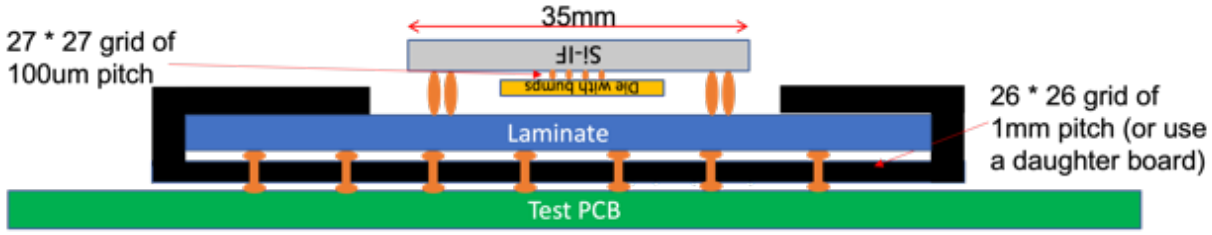


(b) *Cross Section* of Package Design

Figure 3.37: **NeuroCTT 0.2 Package Laminate Design.** (a) 5-layer laminate was designed to make connections between 27×27 grid of $100\mu m$ pitch Cu-pillars to 26×26 grid of $1.27\,mm$ BGAs. Packaged chip to be connected to testing PCB using LGA socket.



(a) *Si-IF* Design ($35 \times 35mm$)



(b) *Cross Section* of *Si-IF*-enabled Package Design

Figure 3.38: **NeuroCTT 0.2 *Si-IF*-enabled Package Design.** Due to vendor limitations with manufacturing 5-layer package laminates (Fig. 3.37a) to support $100\mu m$ pitch Cu-pillar process, a separate *Si-IF*-based package structure was also designed. In this design, a die is bonded directly to a 1-layer Si-IF (Technology Option 1a [CHI20]), which is then soldered to a laminate.

3.3.3 NeuroCTT 0.3 Infrastructure

Two versions of the testing infrastructure were implemented. The first version involved a package-less solution using a 7"×7" Mainboard and 2"×2" Mezzanine card, shown in Figures 3.39 and 3.40. Die was bonded directly to mezzanine card, shown in Fig. 3.41.

Six mezzanine card samples were bonded across 4 different bonding runs with an internal vendor (Center for High-Frequency Electronics (CHFE) at UCLA) and an external vendor located in Anaheim (IDAX Microelectronics Labs). All samples but one had critical ESD failures on the IO supply domain (VDDIO=$3.3V$) after wirebonding, rendering samples useless. The remaining sample had ESD failures on several digital and bias voltage inputs, but a clock signal was still able to be supplied to the chip and CLK_OUT signal observed using an oscilloscope. ESD protection was placed on-chip, but no external ESD protection was provided on mezzanine card design.

It was determined that unknown ESD-related issues were caused by bonding directly to the mezzanine card, so a second version of the testing infrastructure with ceramic packaging was designed. The second version consisted of (1) eliminating the mezzanine card concept, (2) wirebonding directly to a Kyocera PB-C87729 84-pin ceramic QFN (C-QFN) package, and (3) redesigning the Mainboard with a Loranger (P/N 03853 841 6217) C-QFN socket—shown in Fig. 3.42. The redesigned board consists of 281 components—a summary of major components is made available in Table 3.5.

External vendor QP Technologies performed 78-pad wirebonding to 84-pin C-QFN package. Example C-QFN wirebonding samples are shown in Fig. 3.43.

ESD-related issues were resolved after reverting back to traditional packaging route using 84-pin C-QFN package. Example data-scan chain output verification at 20MHz is visible on the oscilloscope in the Lab Test Setup shown in Fig. 3.44.

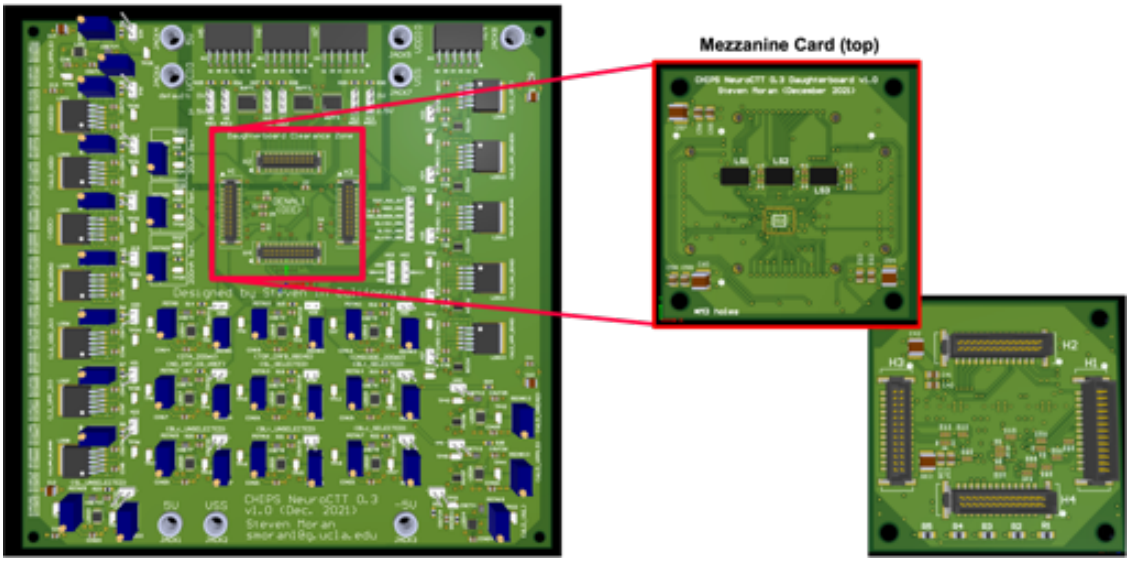


Figure 3.39: **NeuroCTT 0.3 Mainboard & Mezzanine Card Design.** 7" × 7" Mainboard (*left*) consists 3 external supply voltages ($5V, 3.3V, \&-5V$) and 26 LDO-generated supply & bias voltages ranging $-0.3V$ to $3.3V$—7 of which are digitally controlled by FPGA controller. Die is wirebonded to 2"×2" Mezzanine card (*right*) consisting of 78 wirebond pads at 10 mil ($254\mu m$) pitch (die: $100\mu m$ pitch). Mezzanine card is mostly passive with header connections on bottom-side, 78 decoupling capacitors ranging $300pF-100\mu F$, and 3 digital buffer/level-shifter ICs for digital IO paths.

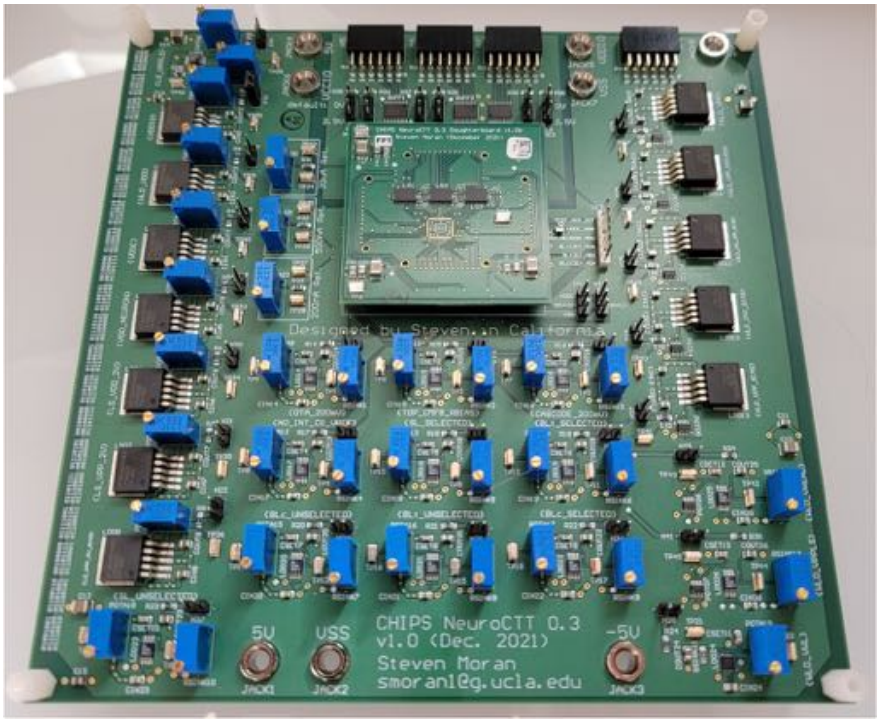
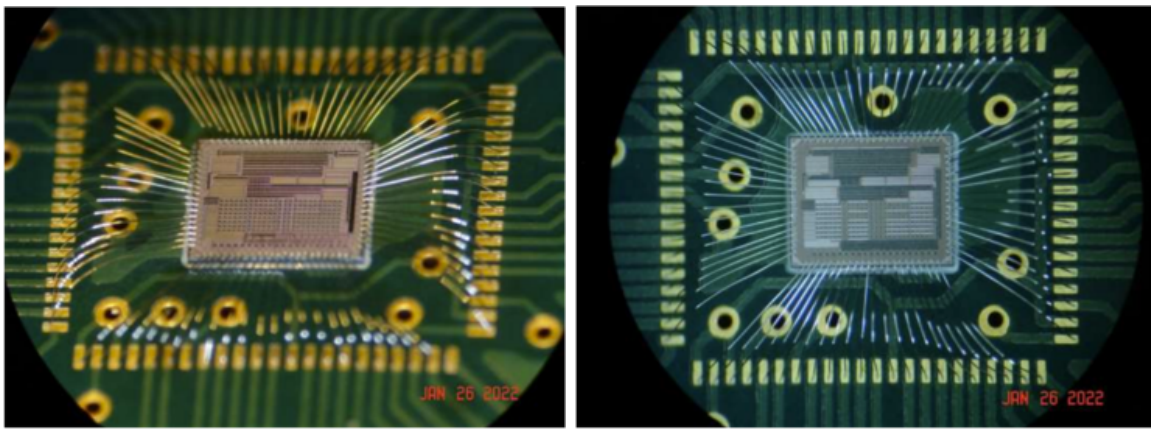
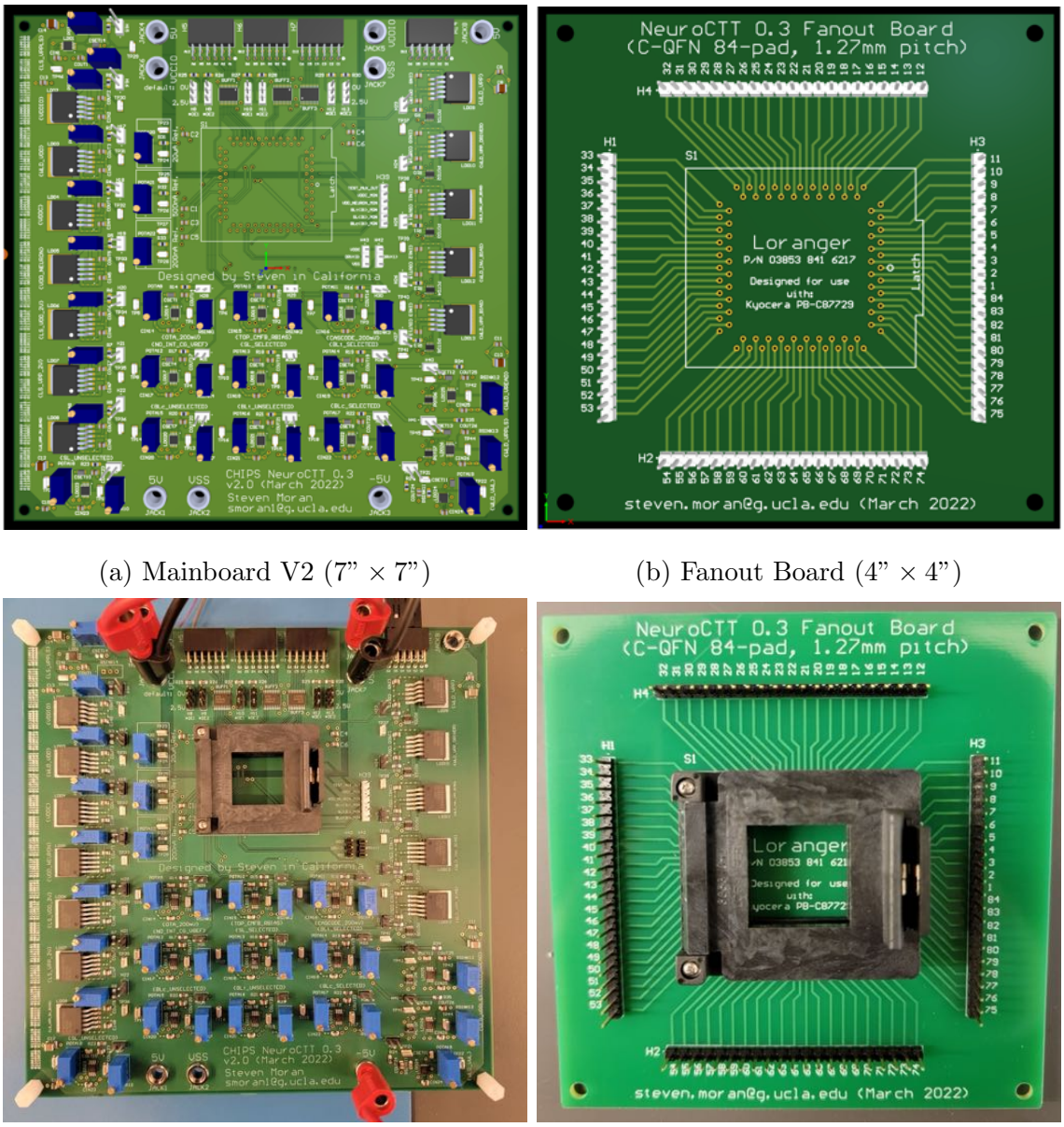


Figure 3.40: **Fabricated NeuroCTT 0.3 Mainboard with Mezzanine Card.**



(a) Sample 1 (b) Sample 2

Figure 3.41: **NeuroCTT 0.3 Die-to-Mezzanine Card Wirebonding.**



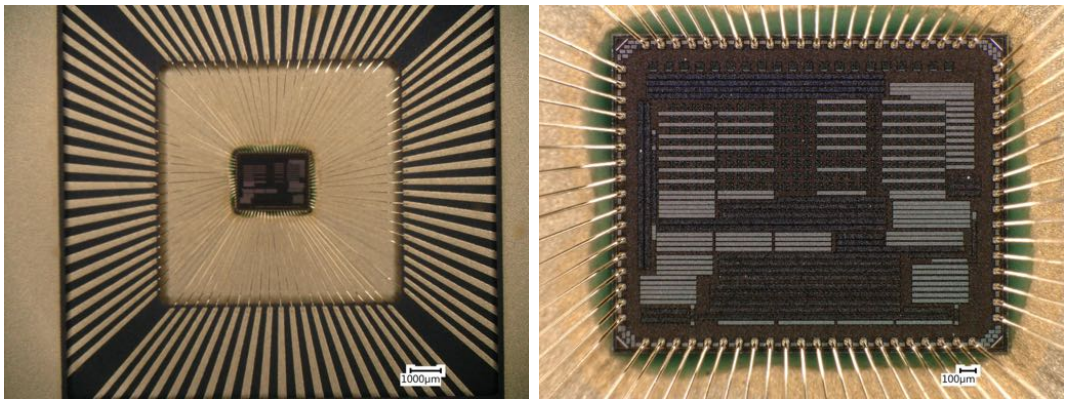
(a) Mainboard V2 (7" × 7")

(b) Fanout Board (4" × 4")

(c) *Fabricated* Mainboard V2

(d) *Fabricated* Fanout Board

Figure 3.42: **NeuroCTT 0.3 Mainboard (v2) & Fanout board designs.** (a) Second version design replaces mezzanine card header connectors with Loranger (P/N 03853 841 6217) 84-pin C-QFN socket, designed for use with Kyocera PB-C87729 ceramic package. (b) A simpler *Fanout* board was also designed with Loranger socket to simplify debugging as well. (c) & (d) represent actual *fabricated* boards.



(a) 50× Zoom (b) 100× Zoom

Figure 3.43: **NeuroCTT 0.3 C-QFN Package Wirebonding.** Dies were directly wirebonded to Kyocera PB-C87729 ceramic package.

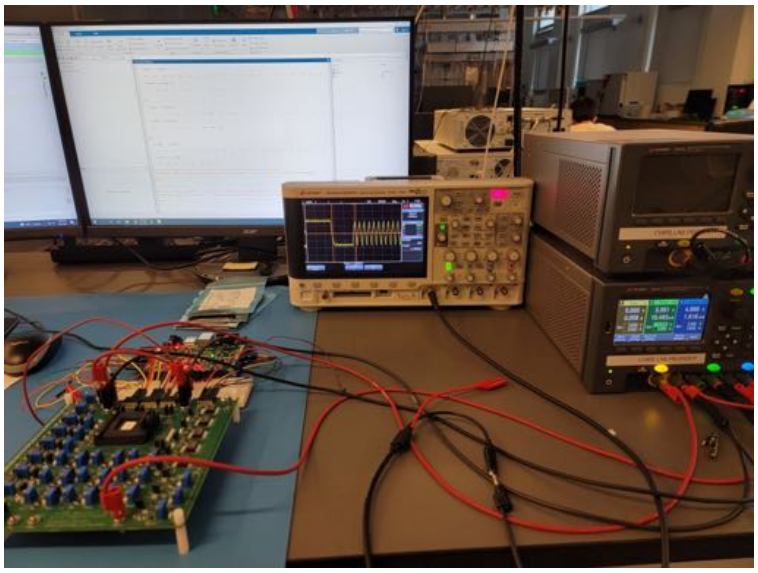


Figure 3.44: **NeuroCTT 0.3 Lab Test Setup.** Die bonded to 84-pin ceramic QFN package (*left*) then connected to Mainboard using Loranger C-QFN socket. Test setup also includes external supplies, Digilent Nexys A7-100T FPGA board, gateway PC, and oscilloscope. UART protocol is used to communicate between FPGA and computer via MATLAB terminal.

Component	Manufacturer	Quantity	Purpose
03853 841 6217	Loranger	1	84-pin C-QFN Socket
LP38500TS-ADJ/NOPB	TI	12	$0.6 - 5V$ LDO
LT3085EMS8E#PBF	Analog Devices	13	0-36V LDO ($10\mu A$)
LT3090EDD#TRPBF	Analog Devices	1	-36V-0V LDO ($50\mu A$)
CLVC540AQPWRG4Q1	TI	3	8ch Logic Inv. Buffer
n/a	Bourns	22	Misc. Pot.
AD5227BUJZ10-RL7	Analog Devices	5	10kΩ Dig. Pot.
AD5227BUJZ50-RL7	Analog Devices	2	50kΩ Dig. Pot.
n/a	Misc.	35	Res. ($50m - 40.2k\Omega$)
JMK325AC7107MM-P	Taiyo Yuden	4	$100\mu F$ 0805 X7S cap.
CGA4J1X7R0J106K125AC	TDK	49	$10\mu F$ 0805 X7R cap.
CGA4J1X7R0J685K125AC	TDK	1	$6.8\mu F$ 0805 X7R cap.
CL21B225KAFNFNE	Samsung	12	$2.2\mu F$ 0805 X7R cap.
AC0805KRX7R7BB104	Yageo	13	$0.1\mu F$ 0805 X7R cap.
5019	Keystone Elec.	45	Testpoints
n/a	Misc.	39	2/3/6/12 headers

Table 3.5: **Summary of NeuroCTT 0.3 Mainboard (v2) Components.**

3.4 Testing User Interface (UI)

Special care was taken when developing the FPGA testing interface to the chip in order to allow simple testing automation as well as to prevent frequent recompilation of FPGA verilog code. A MATLAB-based UART-interface was developed between the gateway computer and the Digilent Nexys A7-100T FPGA board. The UART-interface allows all possible chip and board operations to be entirely controlled by a MATLAB terminal, preventing any FPGA verilog recompilation except in the special case where the user opts to modify the chip operating frequency. This allows the MATLAB terminal to not only send data to/from the chip, but it also allows the user able to modify the board's digitally controlled LDO-generated supplies on the fly in order to support automated Pulsed-gate Voltage Ramp Sweep (PVRS) programming, as discussed previously in Section 2.2. The MATLAB-based UART interface between the Digilent Nexys A7-100T FPGA and the gateway computer over USB officially supports $9600 - 115,200\,baud/s$ ($\sim 4 - 60$ inference samples per second).

3.4.1 MATLAB-based Chip Configuration GUI

The NeuroCTT chip design includes a $1308-bit$ CONFIG_REG for storing various chip configuration parameters. These parameters for the most part require one-time configuration (per chip). A GUI was designed to simplify chip configuration and is shown in Figures 3.45 & 3.46. Users can save & load chip configuration files for simplified testing. Additionally, users can also test modified configuration parameters directly from the GUI by selecting 'Run INFERENCE (Zero Inputs)'. This allows the user to quickly determine the optimal set of neuron offset correction parameters by directly evaluating the output of the neuron when zero input is applied (e.g. output should be zero after correction).

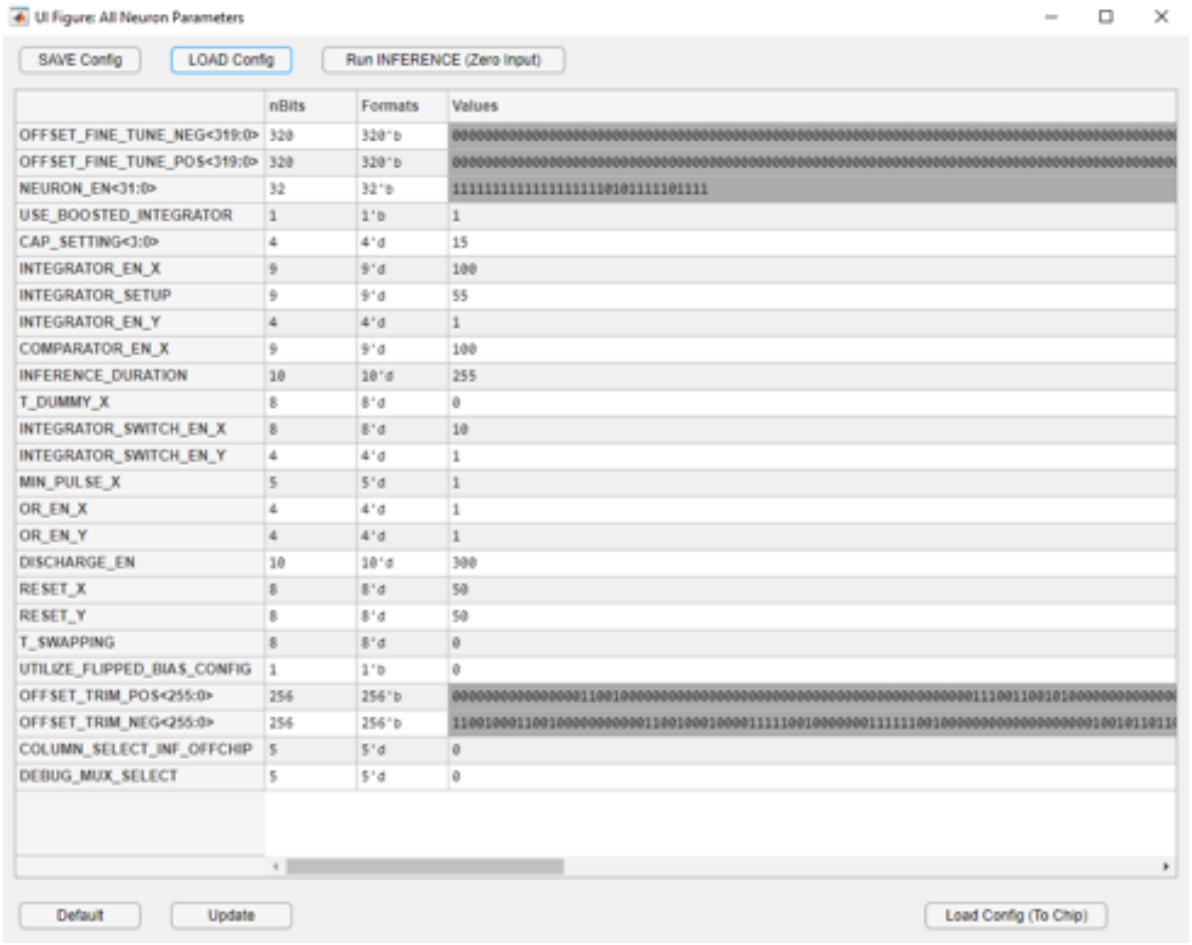


Figure 3.45: **NeuroCTT 0.3 Chip Configuration GUI.** Program allows user to update chip configuration parameters, load them directly to the chip, test the chip output for zero-input condition, and iteratively modify parameters, if necessary. Parameters can be saved to—and loaded from—an output file for reuse

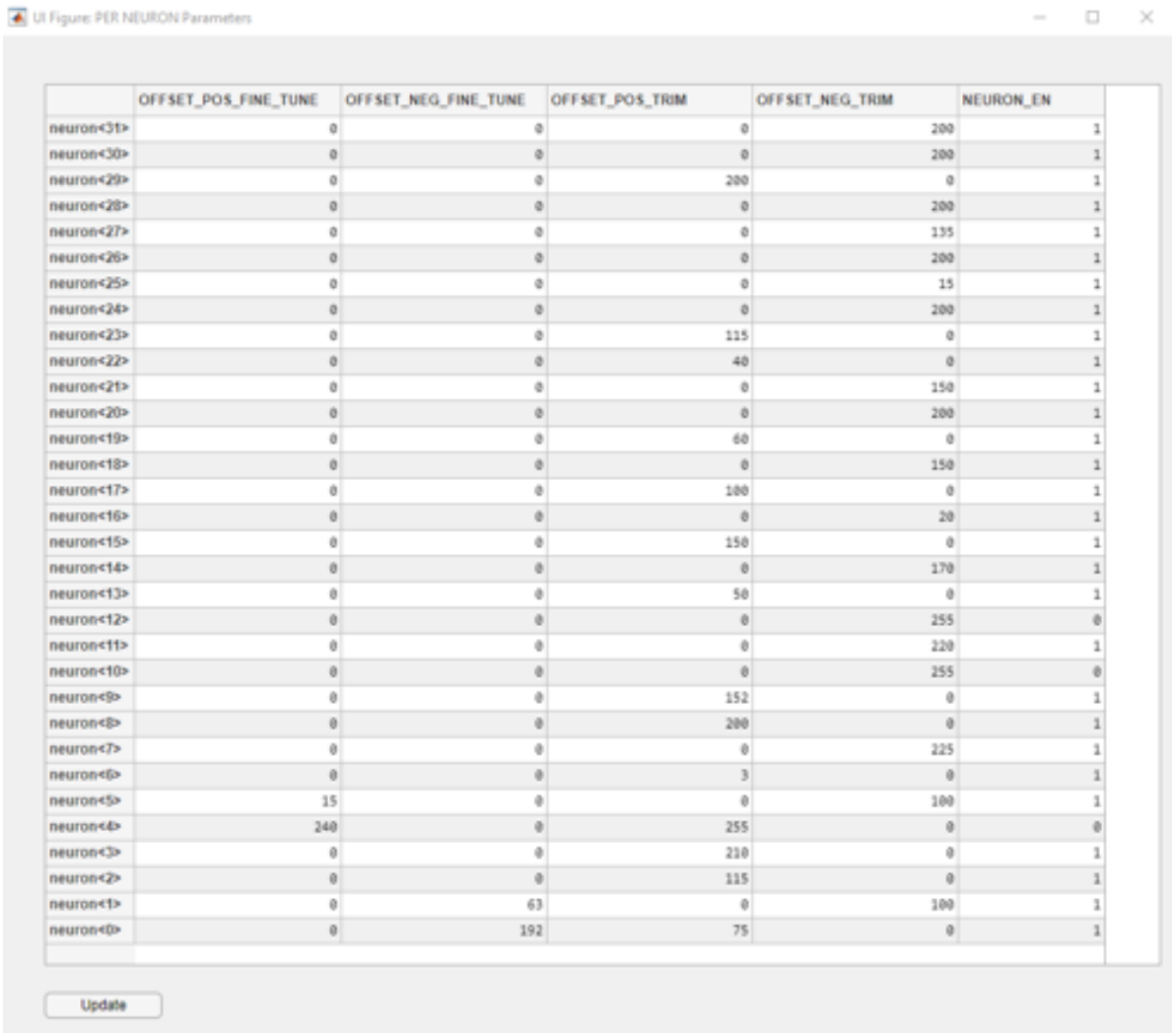


Figure 3.46: **NeuroCTT 0.3 Chip Configuration GUI: Neuron Parameters Window.** Neuron Parameters Window can be used to update parameters per neuron. After modifying parameters, 'Update' button will save push changes to the respective fields on the main window displayed in Fig. 3.45.

```
% Run Automated Inference Script
nRuns = 256;
nNeurons = 32;
results = zeros(nRuns,nNeurons);

for j = 1:1:255,
    input_vector = ones(1,256)*j;
    % Reformat input (Send MSB first, e.g. WL<255>)
    input_vector = fliplr(input_vector);
    % Send input to FPGA
    write(device,input_vector,"uint8");
    % Retrieve output from FPGA (40 Bytes)
    inference_outputs = read(device,40,"uint8");

    % Convert (40B) inference output into 32 x 10-bit vector
    OUTPUT_REG = "";
    for i = 1:1:40,
        OUTPUT_REG = OUTPUT_REG + dec2bin(inference_outputs(i),8);
    end
    OUTPUT_REG = convertStringsToChars(OUTPUT_REG);
    inference_outputs_decimal = zeros(1,32);
    for i = 1:1:32,
       inference_outputs_decimal(i) = bin2dec(OUTPUT_REG((1+10*(i-1)):10*i));
    end
end
```

Table 3.6: **MATLAB Automated *INFERENCE* Example Script.**

3.4.2 Automated Inference Script

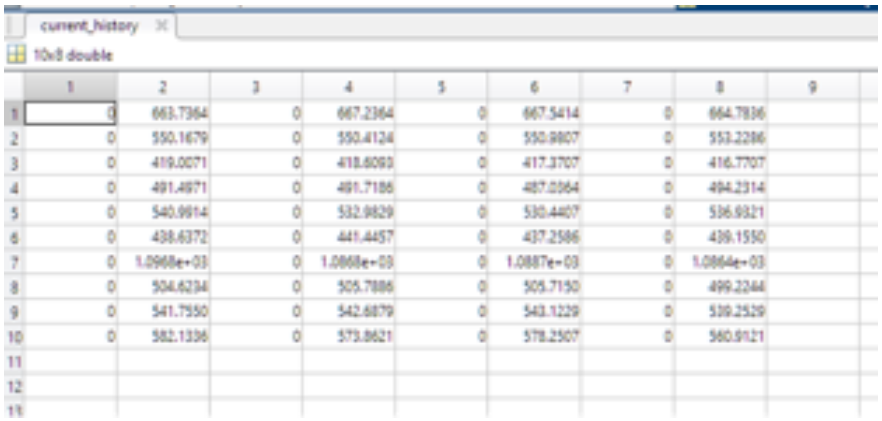
An Automated Inference Script allows for the user to specify a range of inputs to supply to the chip and retrieve the chip outputs for each set of inputs. Chip inputs for each inference sample are supplied as a 256B ($256 \times 8b$) input message followed by a 40B ($32 \times 10b$) output message sent back to the gateway computer for processing. FPGA code is designed to immediately recognize the 256B UART message, load the received $2kb$ data serially to the chip, execute the program on the NeuroCTT chip, retrieve the chip data output serially, and transmit the $320b$ output message back to the host/gateway computer.

```
% Iterate over device columns<0:31>
for j = 1:1:32,
    % Iterate over devices (aka WLs<0:255>)
    for i = 1:1:256,

        % (1) Create 2kb binary message to send to chip
        DATA_REG = data_reg_fun(j,i) % returns 2048-bit vector

        % (2) Format data to send over UART
        DATA_REG = convertStringsToChars(DATA_REG);
        DATA_REG_MESSAGE = zeros(1,256);
        for bb = 1:1:256,
            DATA_REG_MESSAGE(bb) = bin2dec(DATA_REG((1+8*(bb-1)):8*bb));
        end

        % (3) Send UART Data Command (Enable Specific CTT Device)
        write(device,DATA_REG_MESSAGE,"uint8");
        pause(0.1);

        % (4) Disable all CTT devices
        write(device,zeros(1,256),"uint8");
        pause(0.1);
    end
end
```

Table 3.7: **MATLAB Off-chip Verification Example Script.**

3.4.3 Automated Off-Chip CTT Device Weight Verification

Similarly, a MATLAB script can also be written for off-chip verification as well. In this case, the LDOs for the BLt, BLc, and SL selected pads must be first disconnected. The Keysight B1500A analyzer Source Management Unit (SMU) is then connected to the either the BLt/SL or BLc/SL depending on whether the user wants to read out the True (T) or Complement (C) weights. Finally, the user must reload the instruction with the VERIFY_T_OFFCHIP (or VERIFY_C_OFFCHIP) instruction prior to executing the script provided in Table 3.7. Example off-chip verification output measured by the analyzer is detailed in the following chapter and shown in Fig. 4.5. Measurement parameters were optimized in order to guarantee single device measurement repeatability with $<5nA$ error.



	1	2	3	4	5	6	7	8	9
1	0	663.7364	0	667.2364	0	667.5414	0	664.7836	
2	0	550.1679	0	550.4124	0	550.9807	0	553.2286	
3	0	419.0071	0	418.6093	0	417.3707	0	416.7707	
4	0	491.4871	0	491.7186	0	487.0364	0	494.2314	
5	0	540.9914	0	532.9829	0	530.4407	0	536.9321	
6	0	438.6372	0	441.4457	0	437.2586	0	439.1550	
7	0	1.0968e+03	0	1.0868e+03	0	1.0887e+03	0	1.0864e+03	
8	0	504.6234	0	505.7886	0	505.7150	0	499.2244	
9	0	541.7550	0	542.6879	0	543.1229	0	539.2529	
10	0	582.1336	0	573.8621	0	578.2507	0	560.9121	
11									
12									
11									

Figure 3.47: **Off-Chip Verification Measurement Repeatability.** Example shows 10 devices measured 4 times each using an external Keysight B1500A analyzer. Repeatable measurements demonstrated with $<5nA$ variability. Each row corresponds to a different device. Columns 2, 4, 6, and 8 correspond to runs 1-4, respectively.

3.4.4 Automated *On-Chip* CTT Device Programming

On-chip programming can be performed by specifying the target cell and loading in cell programming timing information (e.g. programming pulse width). Before executing the script, the user must modify the BL/SL as well as WL driver voltages to set the proper programming conditions. Example script is provided in Table 3.8. User can optionally add UART-based commands to modify the WL driver voltages (not shown in provided example script) in order to support Pulsed-gate Voltage Ramp Sweep (PVRS)-based programming.

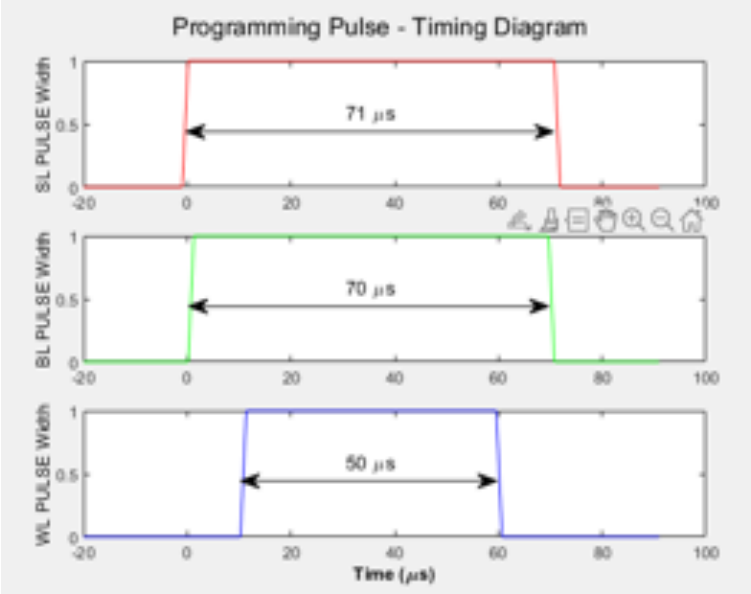
Figure 3.48 provides example output from the Automated *On-Chip* Programming Script, where the user can double-check the timing parameters prior to executing the programming instruction as the programming pulse width timing information is derived from the clock frequency of the chip and specified in terms of number of clock cycles. One the user has double-checked the programming and/or timing conditions, they may choose to continue with programming the selected devices for the provided timing conditions.

```
% On-Chip Programming
% Instructions: make sure to set INST = 1 (PRG_T) or 2 (PRG_C)

CLK_FREQ_MHZ = 20; % MHz

% all values in clock cycles @ specified CLK_FREQ_MHZ
BL_PULSE_WIDTH_X = 200; % Valid Range: 0-65535 (16 bits)
BL_PULSE_WIDTH_Y = 200; % Valid Range: 0-65535 (16 bits)
SL_START = 10; % Valid Range: 0-15 (4 bits)
SL_END = 10; % Valid Range: 0-15 (4 bits)
WL_PULSE_WIDTH = 1000; % Valid Range: 0-15 (4 bits)

create_prg_timing_digram(BL_PULSE_WIDTH_X,BL_PULSE_WIDTH_Y,SL_START,SL_END,WL_PULSE_WIDTH);

% Program 1 device at a time
% ------------------------------------------------------------

    % Iterate over device columns
    for j = 1:1:32, %32,
        % Iterate over devices (aka WLs)
        for i = 1:2:256, %256,

            % (1) Create 2kb PRG message to send to chip
            DATA_REG_PRG = data_reg_prg_fun(j,i,BL_PULSE_WIDTH_X,BL_PULSE_WIDTH_Y,SL_START,SL_END,
                WL_PULSE_WIDTH);

            % (2) Format data to send over UART
            DATA_REG_PRG = convertStringsToChars(DATA_REG_PRG);
            DATA_REG_PRG_MESSAGE = zeros(1,256);
            for bb = 1:1:256,
                DATA_REG_PRG_MESSAGE(bb) = bin2dec(DATA_REG_PRG((1+8*(bb-1)):8*bb));
            end

            % (3) Send UART Data Command (PRG Specified CTT Device)
            write(device,DATA_REG_PRG_MESSAGE,"uint8");

            % (4) Disable all CTT Devices
            write(device,zeros(1,256),"uint8");

        end
    end
end
```

Table 3.8: **MATLAB *On-chip* Programming Example Script.**

```
                        values    units    cycles    units_cycles
                        ______    _____    ______    ____________

    BL_PULSE_WIDTH_X      10        us       200      clk cycles
    BL_PULSE_WIDTH_Y      10        us       200      clk cycles
    SL_START             0.5        us        10      clk cycles
    SL_END               0.5        us        10      clk cycles
    WL_PULSE_WIDTH        50        us      1000      clk cycles

If values are correct, would you like to proceed with programming? [Y/N]: Y

Programming will commence.

DATA_REG string is correct length.
Device #1 Programmed (Column: 0, WL: 0)
Device #2 Programmed (Column: 0, WL: 2)

================================================

Total Devices Programmed: 2
```

(a) Automated Device Programming Output ($f_{CLK} = 20MHz$)



(b) PRG Pulse Timing Diagram

Figure 3.48: **NeuroCTT 0.3 Automated *On-Chip* Device Programming.** (a) Example Script output for 2 selected devices (WL[0] & WL[1] on COLUMN[0]). (b) Script also provides Programming Pulse Timing Diagram prior to starting device programming to make sure user has set correct pulse timing settings.

3.4.5 Automated Program-Verify Weight Fine-Tuning

An automated *Program-Verify* test setup was also created to iteratively program twin-cell devices until they have reached their respective target currents. Device weights are initially read out using the off-chip verification scheme with Keysight Analyzer detailed in Section 3.4.3. A JAVA-based 'ROBOT' [Ora20, Tak10] is utilized to automate the Keysight EasyEXPERT GUI in order to initiate, run, & parse the measurement outputs from the Keysight B1500A Analyzer. A MATLAB script then compares the measured *as-fabricated* device currents with the specified target currents, and a programming algorithm determines an appropriate programming scheme for each twin-cell device including pulse-width, V_{GS} & V_{DS} voltage conditions, and number of pulses. This information is provided to the Automated *On-Chip* Programming function detailed in Section 3.4.4. Device programming and verification is iteratively programmed on each device in the array until each device has reached their respective target current within some margin of error (e.g. $<20nA$).

CHAPTER 4

Hardware Results

The following chapter is organized in four sections which detail hardware results obtained with NeuroCTT_0.1 (*ZION*), NeuroCTT_0.2 (*GLACIER*), & NeuroCTT_0.3 (*DENALI*) circuit designs described in the previous chapter as well as results obtained using the CTT-Hardware-based Inference Realistic Circuit Universal Simulator (CIRCUS) Platform. Most recent silicon (NeuroCTT 0.3) was verified to be 100% functional within the testing confines of the chip and has yielded significant system validation results for all included system blocks. Additionally, system array-level programming & verification results have been correlated with past results obtained from discrete and array-level device macros.

4.1 NeuroCTT 0.1 Hardware Results

NeuroCTT 0.1 results were previously reported in [Wan20a]. The test setup is detailed in Section 3.3.1 and Fig. 3.36. On-chip data scan chain was verified at 40 MHz using external Xilinx Artix-7 FPGA AC701 controller.

As-fabricated ("Virgin") device weights from the on-chip 1024×10 twin-cell CTT array were read out using the Keysight B1500A analyzer and Source Measurement Unit (SMU). On-chip WL Drivers applied a $V_{GS} = \sim 0.2V$ to selected WL(s). The SMU applied a $V_{DS} = \sim 0.05 - 0.2V$ to the selected column (BLt/BLc) and measured the current for each device. *As-fabricated* device weights for 1024×10 twin-cell CTT devices is reported in Fig. 4.1.

On-chip programming was performed by activating the target cell's WL and applying

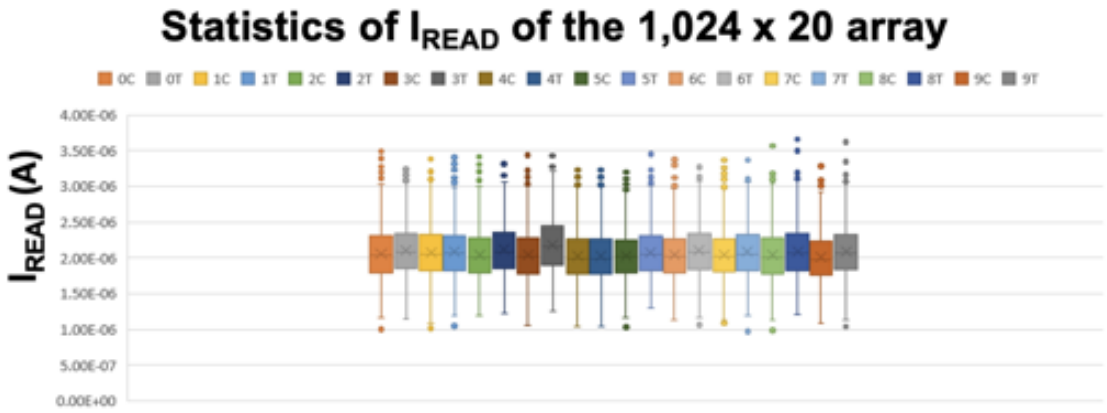
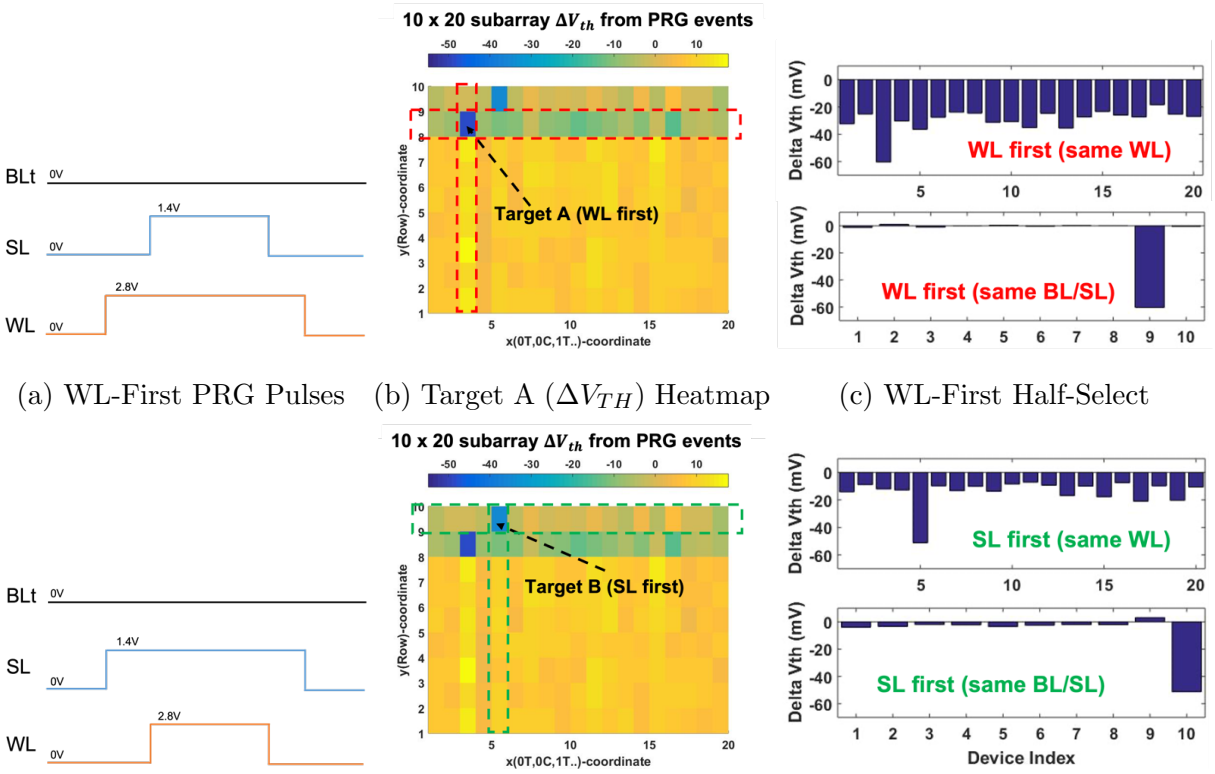


Figure 4.1: **NeuroCTT 0.1 Twin-Cell CTT-Array** (1024×10) ***As-Fabricated*** **Device Weights.**

a V_{GS} pulse of $1.8 - 2.7V$ while simultaneously applying an external pulse to the device's SL using an off-chip Keysight B1500A analyzer. This approach implements the Pulsed-gate Voltage Ramp Sweep (PVRS) method, previously shown in Fig. 2.5a. Half-selected devices along the same row and column of the target cell may experience some unintentional programming. Half-selected issues are studied for the NeuroCTT 0.1 CTT array design and reported in Fig. 4.2. Programming pulses can be applied as WL-first or SL-first as defined by Figures 4.2a and 4.2d.

Half-select results are suboptimal in both cases as devices are unfortunately overstressed by the off-chip SL pulse provided by the analyzer, due to timing limitations. Future designs include *on-chip* generated WL and SL programming pulses to reduce half-select exposure and the overall target cell programming time. Despite half-select issues in both cases, half-select issues were substantially less severe in the SL-first programming case as unselected devices were effectively exposed to $V_{GS} = 1.4V$ ($V_{WL} = 2.8V, V_{SL} = V_{BLc} = 1.4V$) instead of $V_{GS} = 2.8V$, as is the case momentarily when the WL is high before the SL goes high during WL-first programming.



(a) WL-First PRG Pulses (b) Target A (ΔV_{TH}) Heatmap (c) WL-First Half-Select

(d) SL-First PRG Pulses (e) Target B (ΔV_{TH}) Heatmap (f) SL-First Half Select

Figure 4.2: **NeuroCTT 0.1 WL-First & SL-First Programming.** Half-select issues are studied for WL-first (a-c) and SL-first (d-f) CTT device programming while programming two targets cells on the BLt, A & B, respectively. V_{BLc} was set to $1.4V$ in order to reduce half-select for unselected devices on the same WL as the target cell. Substantially less severe half-select issues were observed during SL-first programming.

	Panasonic [1] VLSI 2018	[2] NTHU ISSCC 2019	[4] Georgia Tech SSC-L 2020	[3] TU + NTHU ISSCC 2020	NeuroCTT 0.2 Design Target	CTT Possible Improvements
Network Type	MLP	CNN	CNN	MLP	**MLP**	—
# of layers	4	unknown (3x3)	1	2	**2**	—
Layer Info	194x64, 64x64, 64x64, 64x10	—	64x64	784x100, 100x10	**1024x256, 256x256**	
Capacity	2Mb	1Mb	128x64	158.8Kb	**655.4Kb**	—
CMOS Tech. Node	180nm	55nm	90nm	130nm	**22nm**	< 22nm
Device	RRAM	RRAM	RRAM	RRAM	**CTT**	*CTT*
Cell type	2T2R	1T1R	2T2R	2T2R	**2T**	*2T*
Cell Size	—	0.2025μm^2	2x0.25μm^2	3.38μm^2	**2.496μm^2 (0.52x4.8um)**	*0.936μm^2 (0.26x3.6μm)*
Cell Range (Single)	20-50μA	—	—	0.4-4.0μA	**100-700nA**	*1-15nA*
Max. Input Resolution	—	1-2b	1b	1b	**8b**	—
Max. Weight Resolution	—	4b signed	2b	3b signed	**5b signed**	—
Max. Output Resolution	1b	3b	1b	1-8b	**10b**	—
***[Claimed]* Peak Energy-Efficiency**	20.7 TOPS/W (1024 input inf. read)	53.17 TOPS/W (1-In-4-W-3-O) 21.9 TOPS/W (2-In-4-W-3-O)	25.1 TOPS/W (1-In-2-W-1-O)	78.4 TOPS/W (1-In-T-W-1-O) 0.61 TOPS/W* (1-In-T-W-8-O)	1242 TOPS/W* (1-In-5-W-1-O) 8.9 TOPS/W (8-In-5-W-8-O)	—
MNIST Accuracy (%)	90.8% (with MSMA) 87.3% (normal)	—	—	94.4% (1-In-T-W-8-O1-8-O2) 92% (1-In-T-W-2-O1-8-O2)	98% (8-In-5-W-8-O1-8-O2) **94% (8-In-5-W-2-O1-8-O2)**	—
***[Claimed]* 2-layer Inference Duration**	—	—	—	42ms (N1=8, N2=8) 77us (1-In-T-W-2-O1-8-O2)	0.96us (8-In-5-W-8-O1-8-O2) 0.645us (8-In-5-W-2-O1-8-O2)	—
2-Layer Inf. Cycle Time	—	—	—	~2^{N1+N2} cycles	~2^{N1}+2^{N2} cycles	—

Figure 4.3: **NeuroCTT 0.2 Design Target.** Comparison provided with [Moc18, Xue19, Liu20, He20]. Multi-layer Perceptron (MLP) design is best compared to RRAM MLP work in [Liu20]. CTT Improvements to lower inference currents are possible by using smaller channel width devices (e.g. $W = 428nm \rightarrow 80nm$) and by lowering the subthreshold gate voltage (e.g. $V_{GS} = 200mV \rightarrow 50mV$).

4.2 NeuroCTT 0.2 Hardware Results

Unfortunately, the main system was not testable in NeuroCTT 0.2 due to flip-chip packaging limitations. As of 2020, *low-volume* limitations prevented us from finding a reliable vendor to fabricate and bond to the required 4+ layer, 27×27 pad, & $100\mu m$-pitch packaging laminate. Despite the disappointing lack of results from the 2^{nd} design, a tremendous amount of work went into designing, validating, and demonstrating the feasibility of constructing a fully-analog, nonvolatile, and multi-layer in-memory compute engine. Anticipated performance numbers are reported in Table 4.1 and extensively evaluated in software using the CTT-Hardware-based Inference Realistic Circuit Universal Simulator (CIRCUS). A detailed architectural comparison of the designed system is provided in Fig. 4.3 and Table 4.1.

Item	Specification
IMC Cell Type	Twin-cell CTT Device
Inputs per Frame	1024× 8b inputs (8kb)
Number of Layers	2
$Layer_1$ Size	1024×256
$Layer_2$ Size	256×256
Input Resolution	8 bits
Output Resolution	10 bits
Chip Operating Frequency	800MHz ($T_{min} = 1.25ns$)
IO Frequency	20-250MHz
Maximum Throughput	10^6 FPS
Maximum Required Input Bandwidth	$8\,Gbps$
Energy per MAC	$\sim 360fJ$
Peak Energy-Efficiency	$\sim 8.9TOPS/W$
Normalized PRG Variance ($\sigma'_{PRG}/Range$)	6%
Retention (50hr at 85°C)	$\Delta\sigma'_{CTT} = \sim 0.5\%$ $\lvert\Delta\mu_{CTT}\rvert < 5nA$

Table 4.1: **NeuroCTT 0.2 Chip Specifications.**

Several *Si-IF*-enabled macros were included on the same die and successfully tested. These macros leveraged the *Si-IF* process with $5\mu m$ pillars at $10\mu m$ pitch [BJP17, BJP18, JRN20, CHI20]. In order to perform die-to-wafer bonding, a subset of the manufactured dies were pulled after metal 9 (last Cu layer). Waferpulled dies, Si-IF design, and thermocompression bonding results were previously shown in Fig. 3.32 and reported in [JRN20]. These macros provided crucial demonstrations of the Si-IF technology and SuperCHIPS communication protocol [JRN20]. Additionally, post-fabrication circuit tuning was also demonstrated by tuning the frequency of a ring oscillator using CTT devices, reported in [NI20].

4.3 NeuroCTT 0.3 (*DENALI*) Hardware Results

Promising hardware results were collected on the NeuroCTT 0.3 Chip design using the test setup shown in Fig. 3.44. The following subsections review (1) system-level block validation results, (2) system-level CTT device programming & verification results, and (3) multiply-and-accumulate functionality (*'Inference'*) results with programmed weights. While it is omitted from this dissertation, a follow-on publication is expected in the near future to detail in-hardware *inference* results using trained and mapped network weights.

4.3.1 System Block-Level Validation

Initial chip testing focused on validating all designed blocks within the NeuroCTT system including the (1) Logic System & Chip Interface, (2) GPIOs, (3) WL Drivers, (4) CTT-Array, (5) Array MUX, (6) Level-Shifted Control Logic, (7) Neuron Block, and (8) Neuron Time-to-Digital Converters (TDCs).

Logic System & Load Logic Verification:
The Logic System was first validated by confirming that all 4 scan chains were fully functional by repeatably sending data to the chip from the FPGA at various frequencies and confirming that the received data was correct. Additionally, known bit sequences are automatically loaded into all registers upon chip reset. The system included three serial-input scan-chains tasked with loading data into the DATA_REG (`2048'b`), CONFIG_REG (`1308'b`), and INST_REG (`5'b`) registers. The data register (DATA_REG) is nominally reserved for specifying the WL pulse-width modulated inputs. The configuration register (CONFIG_REG) specifies a variety of neuron offset compensation and timing control parameters. The instruction register (INST_REG) specifies one of up to 32 instructions that can be performed on the chip, where the loaded instruction is only executed if the `CHIP_PROGRAM_EXECUTE_EN` signal is enabled. All input scan chains were validated to be fully operational at 20MHz with the possibility of increasing the system & data load frequency further.

(a) Example Output: 35 cycles (b) Example Output: 70 cycles

Figure 4.4: **Example Neuron PWM Debug Outputs.** Two of the 32 neuron outputs were connected to a digital output pad that can be connected to an external oscilloscope for verifying the Neuron Time-to-Digital (TDC) converter output.

The final scan chain is an output-only scan chain corresponding to the `320'b` TDC output register (COUNT_REG) which allows users to retrieve the digitized neuron output from the chip upon request. The digitized output corresponds to the output of the Neuron's Time-to-Digital Converter (TDC) block. Two of the 32 neuron PWM outputs were directly connected to external debug pads, allowing the neuron PWM outputs to be measured externally by an oscilloscope—two example neuron PWM outputs shown in Fig. 4.4. The externally measured output was compared with the digitized output stored in the COUNT_REG across 100 trials and shown to be 100% across all runs. Special efforts were taken into consideration when designing the TDC block to prevent the possibility of a hold or setup-time violation corrupting the output of the TDC block and leading to undefined behavior.

#	Bias Signal Name	Typ. Range	Chip 2	Chip 3
1	CM_OFFSET_BIAS_PMOS_200nA	450-625mV	0.545mV	0.528mV
2	CM_OFFSET_BIAS_NMOS	250-400mV	0.314mV	0.323mV
3	CM_DISCHARGE_BIAS_PMOS_500nA	315-450mV	0.411mV	0.384mV
4	CM_DISCHARGE_BIAS_NMOS	375-500mV	0.431mV	0.425mV
5	COMPARATOR_BIAS	615-650mV	0.601mV	0.607mV
6	CM_VGS_INTEGRATOR_BIAS_PMOS_IREF_20uA	425-575mV	0.488mV	0.499mV
7	CM_VGS_INTEGRATOR_BIAS_NMOS_IREF_20uA	335-450mV	0.417mV	0.412mV
8	CM_VGS_COMPARATOR_BIAS_NMOS_IREF_20uA	310-430mV	0.365mV	0.379mV

Table 4.2: **NeuroCTT 0.3 Neuron Biasing Verification.** A debug mux was utilized to probe multiple bias voltages within the neuron integrator & comparator circuits to confirm proper biasing prior to performing any time-domain experiments. Table reports actual bias voltages measured from CHIP2 & CHIP3. All other chip samples have also been verified.

Neuron Verification:

Proper neuron operation was first verified by using the on-chip debug mux to probe several bias nodes within the integrator and comparator circuits. Power-gating logic was disabled in order to ensure all circuits were on and to allow for all dc bias voltages to be accurately measured using a multimeter. Measured bias voltages from three chips and typical voltage ranges obtained from simulations are provided in Table 4.2. All probed bias voltages were found to be within expected range. Additionally, neuron power consumption was externally measured by measuring the VDD_NEURON ($0.9V$) supply current—agreeing with expectations ($450\mu W \times 32\, neurons = \sim 14mW$).

WL Drivers:

WL Drivers were designed to support a range of inference (e.g. $\{-300mV, 200mV\}$), programming ($1.5-2.7V$), and erase ($-0.5V$) conditions, as detailed in Section 3.1.3. WL driver logic and output voltage swings for all modes of operation were verified by externally probing a dummy WL Driver through the debug mux. The dummy WL Driver was designed to duplicate logic for $WL\langle 0\rangle$. WL Driver operation was confirmed to be 100% operational and programmed output pulses were viewable via an external oscilloscope and matched specified digital WL driver inputs, further validating the DATA_REG scan chain.

CTT Array & Array Column Mux:

CTT Array & Array Column Mux functionality was first confirmed by reading out all of the *as-fabricated* (**PREF-PRG**) device weights using the Off-Chip Verification instruction and an external Keysight B1500A analyzer. The purpose of this experiment was to confirm that (1) the devices could be individually measured, (2) the array columns could be automatically switched between using the array mux & associated level-shifted logic, and (3) ensure column leakage currents are within expectations. Additionally, device measurements were validated against results obtained using discrete device and array macros. During the design phase, special care was taken to ensure sufficiently low-resistance routing to ensure efficient device programming. Array validation included ensuring that devices are programmable with reasonable retention. Cell programming is explored in more detail in a subsequent section.

Initial Off-Chip Verification results are critical for accurately characterizing in-array CTT devices. Off-chip verification is performed by configuring the array column mux, described in Section 3.1.5, in such a way that it allows an external tool to measure the I_{DS} of selected device(s). *As-fabricated* devices were read using an external Keysight B1500A analyzer and Source Measurement Unit, previously shown in Fig. 2.6, along with the chip's off-chip verification testing interface detailed in Section 3.4.3.

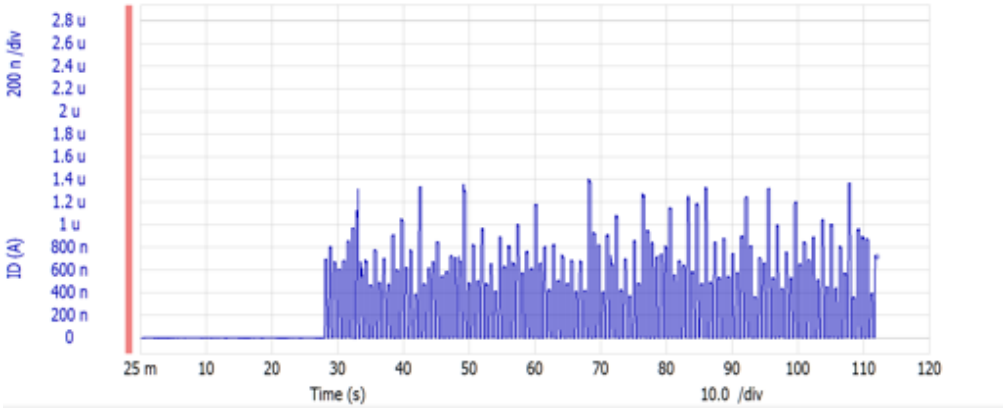
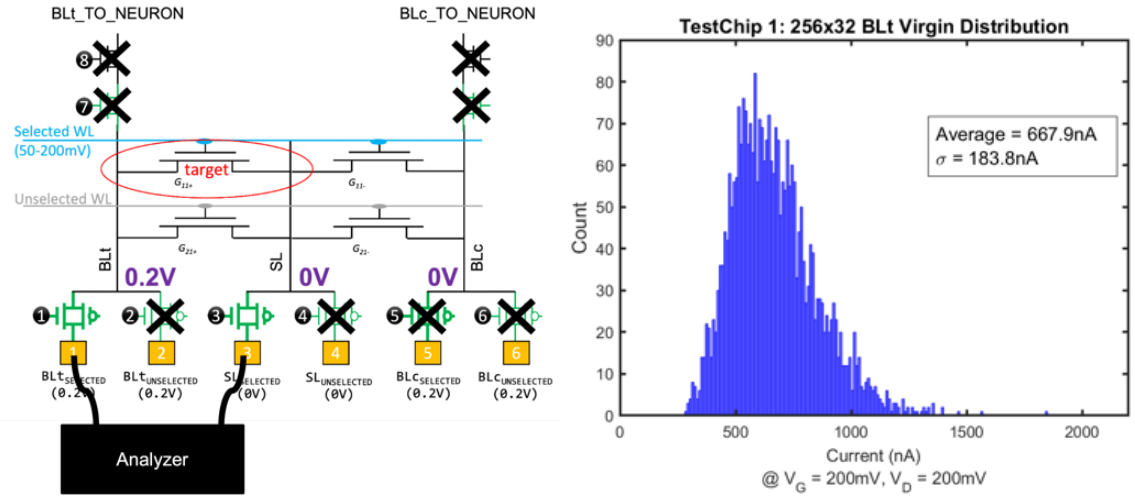


Figure 4.5: **Example Off-chip Verification Output using Analyzer.** CTT devices on-chip were read out one at a time using a Keysight B1500A Analyzer and Source Measurement Unit (SMU). A Matlab-based UART script was used to communicate with the chip and iteratively select & activate each device for $\sim 300ms$. All WLs were disabled in between each device reading (e.g. $I_{measured} \sim 0nA$) in order to simplify data extraction.

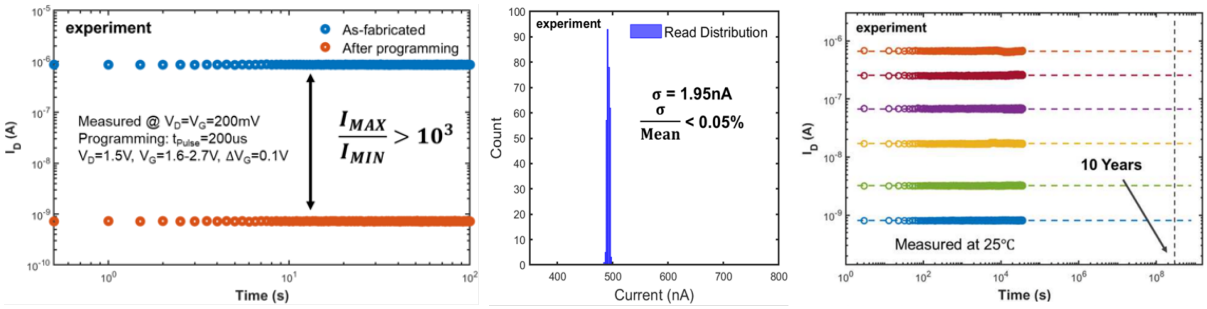
Figure 4.5 provides an example off-chip verification output for a partial column of devices as measured by the Keysight B1500A analyzer. A Matlab-based Off-chip Verification script, similar to that provided in Table 3.7, was used to communicate with the chip over UART and iterate over each of the devices one-at-a-time. Each device is activated for $\sim 300ms$ with an additional $\sim 300ms$ gap in between each device reading to simplify data parsing and analysis. The magnitude of each pulse in the figure corresponds to each device's subthreshold on-current (WL$\langle 0 \rangle$,WL$\langle 1 \rangle$, ...) at the specified $V_{GS} = 0.2, V_{DS} = 0.2V$ bias condition. Unselected devices are biased with $V_{GS} = -0.3V$ to minimize any column leakage currents.

Figure 4.6 displays the on-chip CTT array *as-fabricated* (**Pre-PRG**) device distribution for 256×32 BLt (True) devices, obtained by extracting and re-plotting the off-chip verification results shown in Fig. 4.5. *As-fabricated* device results correlate with expected results from simulation. Repeatable device measurements are shown in Fig. 4.7.



(a) (b)

Figure 4.6: **NeuroCTT 0.3 Twin-Cell CTT-Array *As-Fabricated* Device Weight Distribution.** (a) Keysight B1500A Analyzer and Source Measurement Unit (SMU) were connected to the chip's BLt/SL & BLc/SL pads, respectively, to measure each device. (b) Measured Weight Distribution for 256×32 CTT devices.



(a) (b) (c)

Figure 4.7: **Repeatable Device Measurements using External Analyzer.** (a) Current monitoring of a single as-fabricated CTT device before and after applying 12 programming pulses using PVRS. An $\sim$1000$\times$ difference in channel conductance is observed before and after programming ($\sim 800nA \rightarrow <1nA$). (b) 300 measurements over a 48-period were taken on a separate programmed device (target=500nA) and were shown to be repeatable ($<5nA$). (c) Six devices programmed to different target states and monitored for 10 hours.

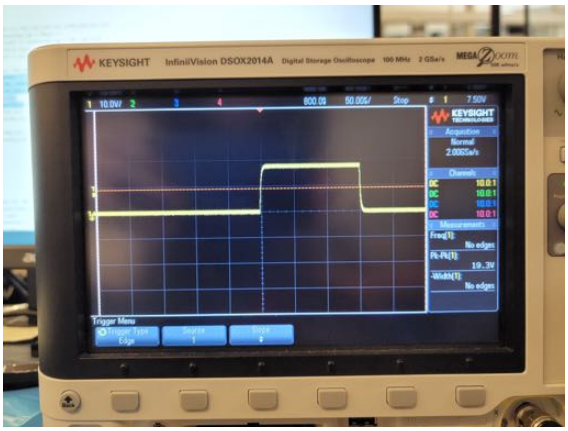


Figure 4.8: **NeuroCTT 0.3 Example SL Programming Pulse Measured Externally.** Programming with $150\mu s$ SL pulses was initiated on Column$\langle 0 \rangle$. *Unselected* columns observe $1.8V$ SL/BLt/BLc pulse to prevent half-select issues on devices connected to the activated WL. DEBUG_PAD_BLt$\langle 31 \rangle$ & DEBUG_PAD_BLc$\langle 31 \rangle$ debug pads were connected to an external oscilloscope and the SL pulse width was measured to be $150\mu s$—set by on-chip programming parameters. Applied SL Voltage was also verified (e.g. $V_{SL} = 1.8V$).

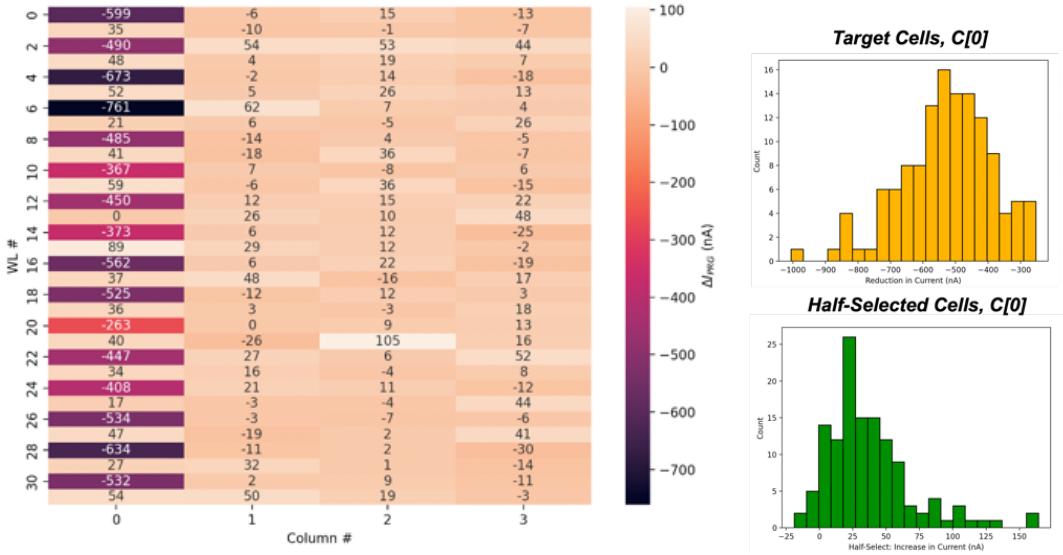
4.3.2 System-Level On-Chip Programming and Verification

Recent silicon (NeuroCTT 0.3) allows for programming pulse widths to be tuned down to $<50ns$ compared to $100\mu s$ minimum pulse widths previously used with device macro results via the Keysight B1500A analyzer. BL, SL, and WL voltage pulses are each individually tunable given the timing diagram shown previously in Fig. 3.16.

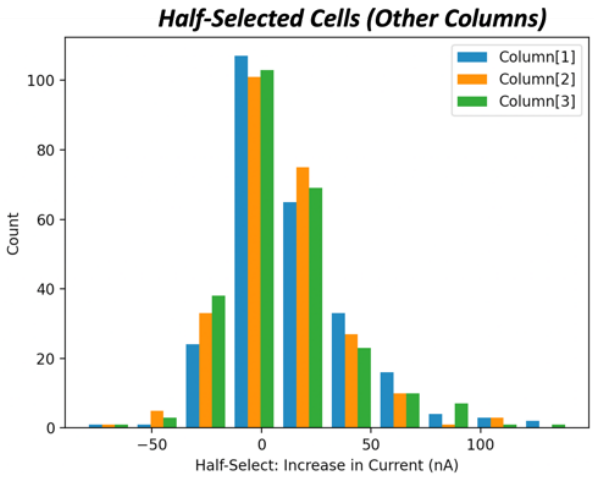
Prior to initiating extensive device programming studies, additional logic validation was performed by ensuring that appropriate programming pulse timing information was correct. An oscilloscope was connected to a set of array debug pads. A single programming pulse was applied to the array with SL pulse width digitally set to $150\mu s$. Figure 4.8 shows that an $150\mu s$ programming pulse was correctly applied to the array based on the specified timing parameters set using the script shown in Section 3.4.4.

On-Chip programming functionality was previously reviewed in Sections 3.1.5 & 3.4.5. Given that the CTT (Regular V_{TH}, nfet) devices are nominally rated for maximum V_{GS}, V_{DS} conditions of $<1V$, careful consideration was taken in order to limit device stress and programming duration—especially for *unselected* devices.

SL-first programming was utilized to limit half-select issues for *unselected* devices on the same column and/or WL—demonstrated via past programming studies on the NeuroCTT 0.1 tapeout (Section 4.1, Fig. 4.2). Intuitively, SL-first programming provides less stress to all *unselected* devices as the programming V_{GS} can range from $2-2.7V$ and SL-first programming ensures that the maximum positive gate voltage for *unselected* devices on the target WL is $<(V_{GS}-V_{DS})$, e.g. $2.7V-1.8V = 0.9V$. Initial programming results for 128 CTT devices after $10\times$ programming pulses at $V_{GS} = 2.2V$ and $V_{DS} = 1.8V$ are shown in Fig. 4.9. Target Cells experienced ΔI up to $-1000nA$ after programming. Half-selected cells observed a slight increase in current after programming which can be compensated for using an iterative program-verify technique.

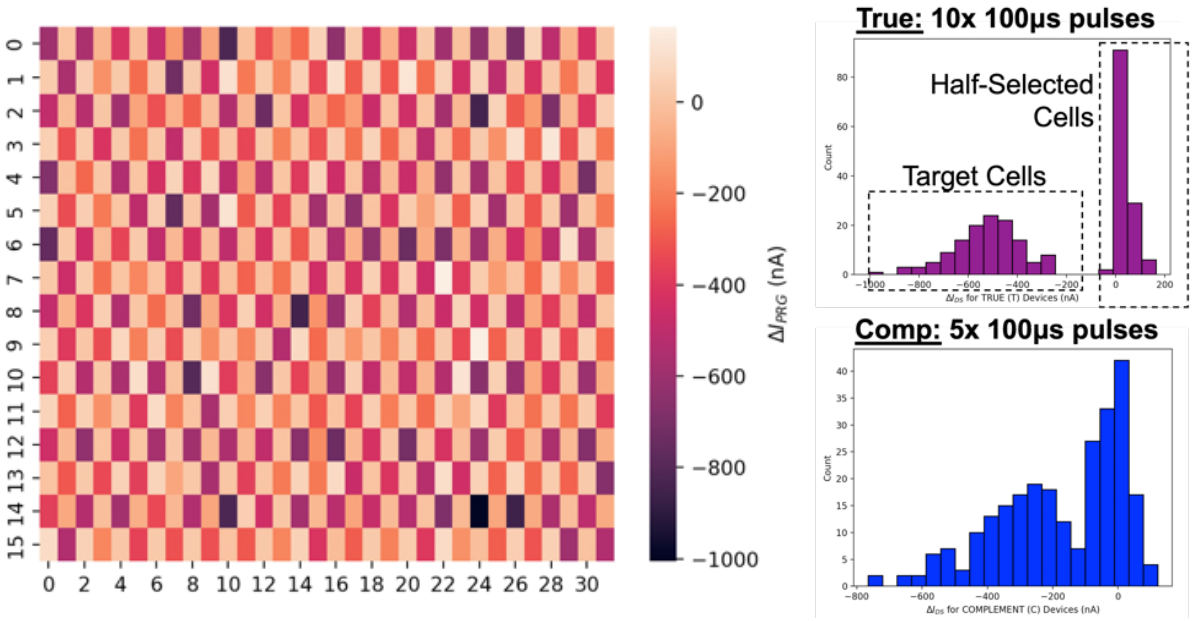


(a) (b)



(c)

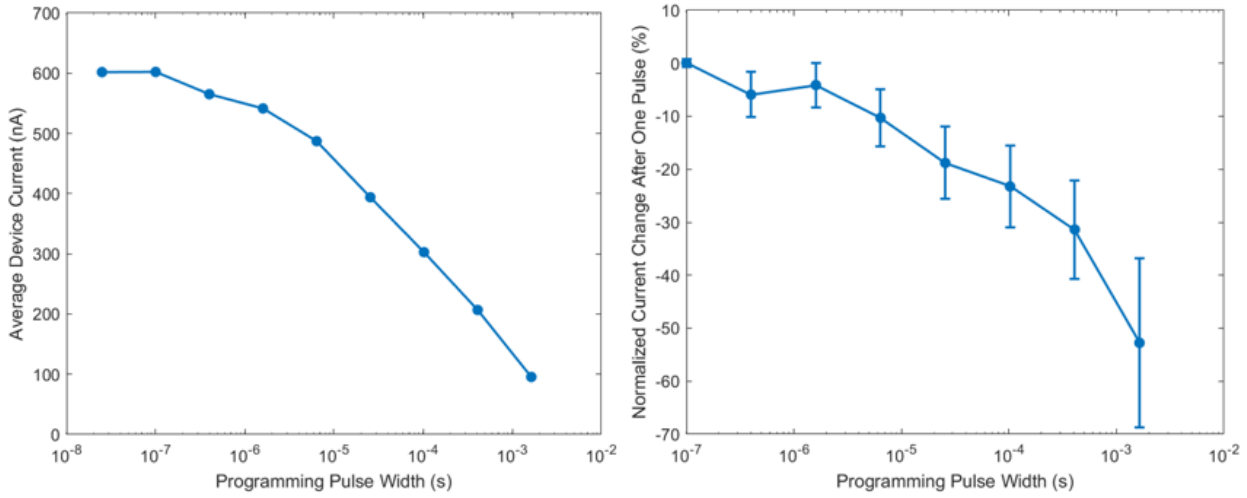
Figure 4.9: **Initial Half-Select Results after Target Cell Programming.** True devices on *even* WLs for $Column\langle 0\rangle$ were programmed using 10× pulses with $V_{GS} = 2.2V$ & $V_{DS} = 1.8V$. (a) Change in current for all array True cells ($Columns\langle 0:3\rangle$) after programming (subset of $WLs\langle 0{:}\ 31$ shown). (b) Distribution of ΔI_{WEIGHT} after programming for Target & Half-Selected cells on $Column\langle 0\rangle$. (c) Distribution of ΔI_{weight} for half-selected cells on other columns.



(a) (b)

Figure 4.10: **Checkerboard Array Programming of True & Comp Devices.** *Even* True devices received 10× programming ($V_{GS} = 2.2V$, $V_{DS} = 1.8V$) pulses & *odd* Comp. devices received 5× programming ($V_{GS} = 2.2V$, $V_{DS} = 1.8V$) pulses. (a) ΔI_{WEIGHT} after cell programming is shown for $WLs\langle 0{:}15\rangle$ and $Columns\langle 0{:}15\rangle$. *Odd* column #'s represent True devices and *even* represent Comp devices. (b) Distribution of ΔI_{WEIGHT} for Target True & Comp cells is also shown.

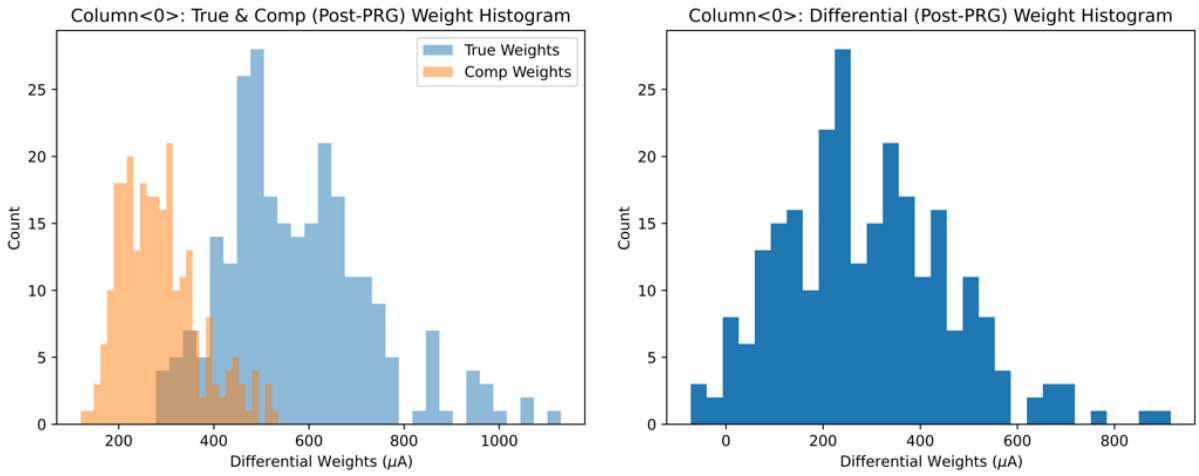
Additional array programming results are shown in Fig. 4.10. A checkerboard pattern was programmed in order to check for possible layout-dependent programming effects—no substantial effects were found. True devices on *even* WLs were programmed via 10× ($V_{GS} = 2.2V$, $V_{DS} = 1.8V$) pulses while Comp. devices on *odd* WLs were programmed via 5× pulses at identical conditions, all across $Columns\langle 0{:}15\rangle$. As shown in Fig. 4.10b, 10× programming pulses at the specified condition were sufficient to fully separate the target and *unselected* cell distributions.



(a) (b)

Figure 4.11: **Pulsed-Voltage Time Sweep (PVTS) Results.** (a) Mean device current for 128 devices programmed subsequently with increasing pulse-width gate voltage pulses: 100ns, 400ns, ..., 409.6μs, 1.64ms. (b) Distribution of 128 devices where each devices receives subsequent gate voltage pulses with similarly increasing pulse-widths.

In addition to the traditional Pulsed-Voltage Ramp Sweep (PVRS) programming methodologies reviewed in Section 2.2 and Figure 2.5a, the most recent chip allows for programmable duration pulses to be applied during cell programming. A Pulsed-Voltage Time Sweep (PVTS) programming scheme is proposed which would simplify future design implementations by potentially requiring a single programming bias condition to fine-tune program all devices rather than requiring ramping gate pulse voltages. Figure 4.11 highlights some initial PVTS-based programming results which provide some information about programming efficiency as a function of programming pulse-width. Figure 4.11a reveal that pulse-widths down to 100ns are rather ineffective at programming devices, likely because the CTT requires self-heating in order to efficiently trap charge and program the device; however, efficient programming with approximately linear current drops in between subsequently increasing pulses is shown for pulses >1.6μs.

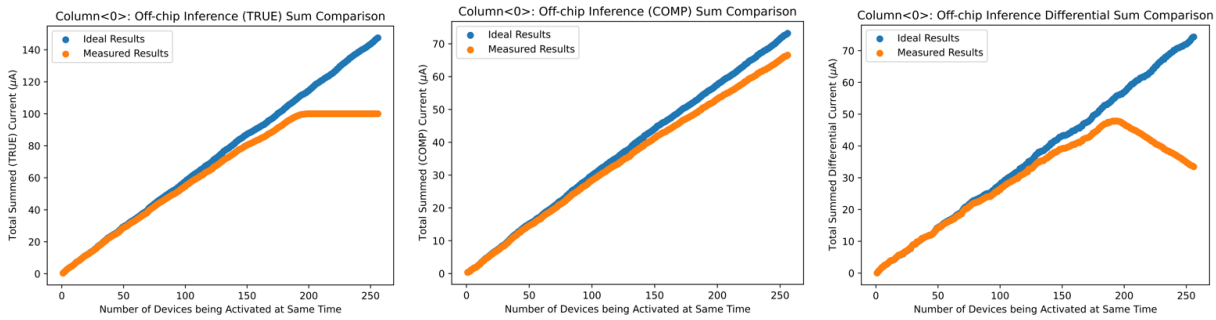


(a) (b)

Figure 4.12: **Example Positive Twin-Cell Weight Programming.** All 256 twin-cell devices were programmed to an arbitrary positive differential weight. True devices were left in their respective *as-fabricated* (Pre-PRG) state while complement devices were each individually programmed by applying 20× 100μs pulses at $V_{GS} = 2.2V$, $V_{DS} = 1.8V$. (a) True & Complement Device distributions after Comp. cell programming. (b) Corresponding differential twin-cell weights after Comp. Cell programming. (Chip 4, Column$\langle 0 \rangle$)

4.3.3 Demonstrating a MAC Engine with Programmed CTT Weights

A reliable MAC Engine with nonvolatile, CTT-based weights requires accurate cell programming, reasonable retention, and minimal measurement error. MAC Engine outputs with programmed weights are first evaluated using an external analyzer. Previously, device measurement repeatability using an external analyzer was explored on single devices, where the normalized 1σ error was shown to be <0.4% (e.g. 1.95nA) for a 500nA target cell, shown in Fig. 4.7b. Measurement accuracy of the summed differential current from a column of devices simultaneously enabled is evaluated in this section. A column of 256 twin-cell devices are first programmed to all positive differential weights by programming only the complement cells via 20× 100μs programming pulses, shown in Fig. 4.12.

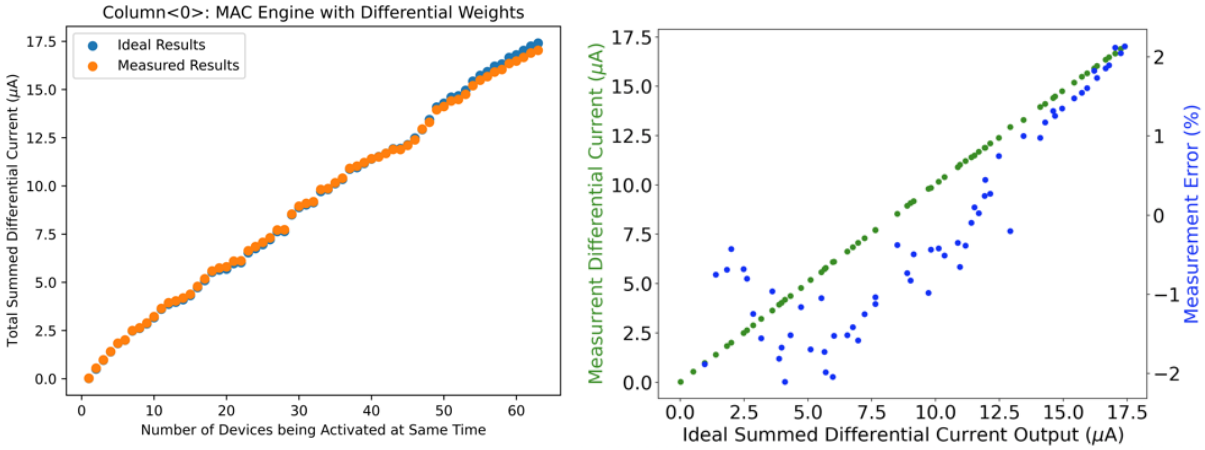


(a) (b) (c)

Figure 4.13: **256-input MAC Engine Results Measured using External Analyzer.** Positive twin-cell weights from Fig. 4.12 were utilized. (a) True & (b) Comp. devices were measured separately and compared against ideal results. (c) Differential sum results are shown. Flat-line shown in (a) and resulting impact on (c) is due to $100\mu A$ current compliance limit. (Chip 4, Column$\langle 0 \rangle$)

After programming, individual weights were measured one-at-a-time using an external analyzer for calculating ideal sums. Summed column output current was then measured using external analyzer when multiple inputs or devices were simultaneously enabled, shown in Fig. 4.13. I_{BLt} and I_{BLc} are separately measured, shown in Figures 4.13a & 4.13b. For increasing common-mode currents, the measured column currents begin to deviate from the ideal results due to IR drop in the Array Column Mux. It's also important to point out that the flat-line shown in Fig. 4.13a is not fundamental and only due to a $100\mu A$ current compliance limit set on the external Keysight B1500A analyzer. The demonstrated on-chip MAC Engine utilizes all system-level blocks including WL Drivers, CTT Array, CTT Array Column Mux, Level-Shifters, and On-Chip Logic, except for the Neuron circuit. Instead of utilizing the neuron, an external analyzer is utilized to compute the overall weighted-sum. Measurement error for sums $<30\mu A$ was demonstrated to be $<2\%$.

Future experiments were limited to a maximum of 64 inputs to manage measurement error due to IR drop in the Array Column Mux. Figure 4.14 demonstrates a subset of

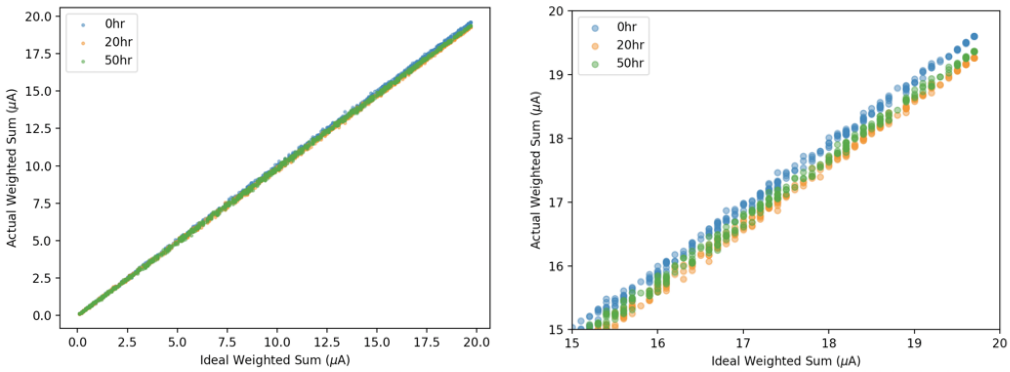


(a) (b)

Figure 4.14: **64-input MAC Engine Results Measured using External Analyzer.** Subset of 64 positive twin-cell weights ($WLs\langle 0{:}63\rangle$) from Figures 4.12 & 4.13 were utilized. (a) Differential Sum results shown as a function of number of inputs enabled. Some mismatch visible for large inputs due to array column mux IR drop for large common-mode currents. (b) Ideal sums vs. Measured Sums show excellent linearity in measurements with <2% measurement error for 64-input MAC. (Chip 4, Column$\langle 0\rangle$)

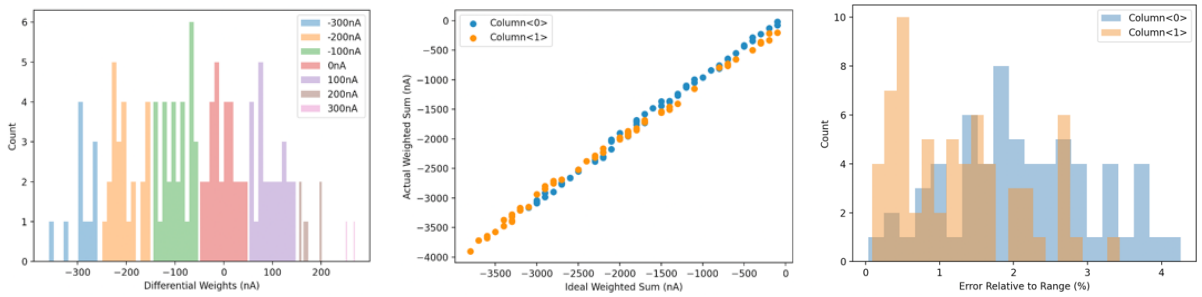
the results from the previous figure. Ideal vs. Measured Differential Output currents are plotted in Fig. 4.14b, showing a maximum measurement error of ±2% with respect to the differential output current. Measurement error due to IR drop in the Array Column Mux is not fundamental. Overall error is still dominated by programming variance.

Additional MAC Engine Results using the Array Macro results from Figures 2.7b & 2.9 are shown in Fig. 4.15. Separately, on-chip fine-tuning to seven differential target states was attempted and results are shown in Fig. 4.16. On-chip fine-tune programming requires further improvement. Iterative programming is limited by cumbersome manual switching after every verification step. A modified PCB board is being designed to fully automate the program-verify weight fine-tuning algorithm. In the future, program-verify can be further accelerated using on-chip neuron design for verification.



(a) (b)

Figure 4.15: **MAC Engine with 6 Target States.** MAC Engine Results shown for 480 devices programmed to one of 6 target states (100, 200, ..., 600nA), shown previously in Figures 2.7b & 2.9 before and after 20hr & 50hr baking at 85°C. A maximum of 60 devices were measured simultaneously. Full MAC Engine results are shown in (a). Zoomed in results for Ideal Sums >15μA are shown in (b). MAC Engine Results shown immediately after programming (0hr) and after 20hr & 50hr baking, respectively, at 85°C.



(a) (b) (c)

Figure 4.16: **MAC Engine with 7 Differential Target States.** (a) 128 Devices were programmed using on-chip programming to one of 7 differential target states. (b) MAC Engine Results for two columns of 64 devices are shown. (c) Relative error compared between summed current and ideal summed current w.r.t. to ideal target states. (Chip 3)

4.3.4 Demonstrating a MAC Engine with On-Chip Neuron

An on-chip neuron circuit with differential integrator and comparator circuit was designed to compute the PWM-based weighted sum of a column of up to 256 twin-cell devices. The integrator provides a 200mV virtual supply to the CTT Array ($V_{CTT,D}$) and integrate the differential column current. The comparator circuit converts the computed weighted sum (Q_{INF}) into a pulse-width modulated output (t_{PWM}) which can be applied as an input to subsequent network layers. Neuron circuit architecture & design is discussed in more detail in Section 3.1.6.

Preliminary neuron results and output linearity are discussed in this section and more detailed debugging are discussed in the following section (Section 4.3.5). The neuron pulse-width modulated output is designed to implement the Rectified Linear Unit (ReLU) activation function, shown previously in Fig. 3.23, where negative sums are set to zero. For positive sums, the neuron computes a linear output until the output reaches saturation.

Integration of positive-only differential twin-cell weights (from Fig. 4.12) using on-chip neuron circuit is shown in Fig. 4.17. Figures 4.17a-4.17c show integration results across 3 different columns and neuron circuits ($Columns\langle 0{:}2\rangle$) when selected inputs are activated for 16 cycles. For comparison, the experiment was repeated with selected inputs activated for 8 cycles, shown in Figures 4.17d-4.17f. Neuron output reaches saturation $\sim$2$\times$ faster when selected inputs are activated for 16 cycles compared to 8 cycles. Non-ideal neuron output variability is observed over multiple iterations with the same inputs. Output variability is observed to scale proportionally to $T_{INTEGRATION}$ (defined by Eq. 4.4), suggesting that an error current is being integrated and imposing an error term in the integrated output value. Sources of the output variability issue are discussed in further in the following section. Regardless, it was found that approximately linear output from the neuron with up to 7b resolution could be achieved by averaging the neuron output over multiple iterations, as shown in Fig. 4.18.

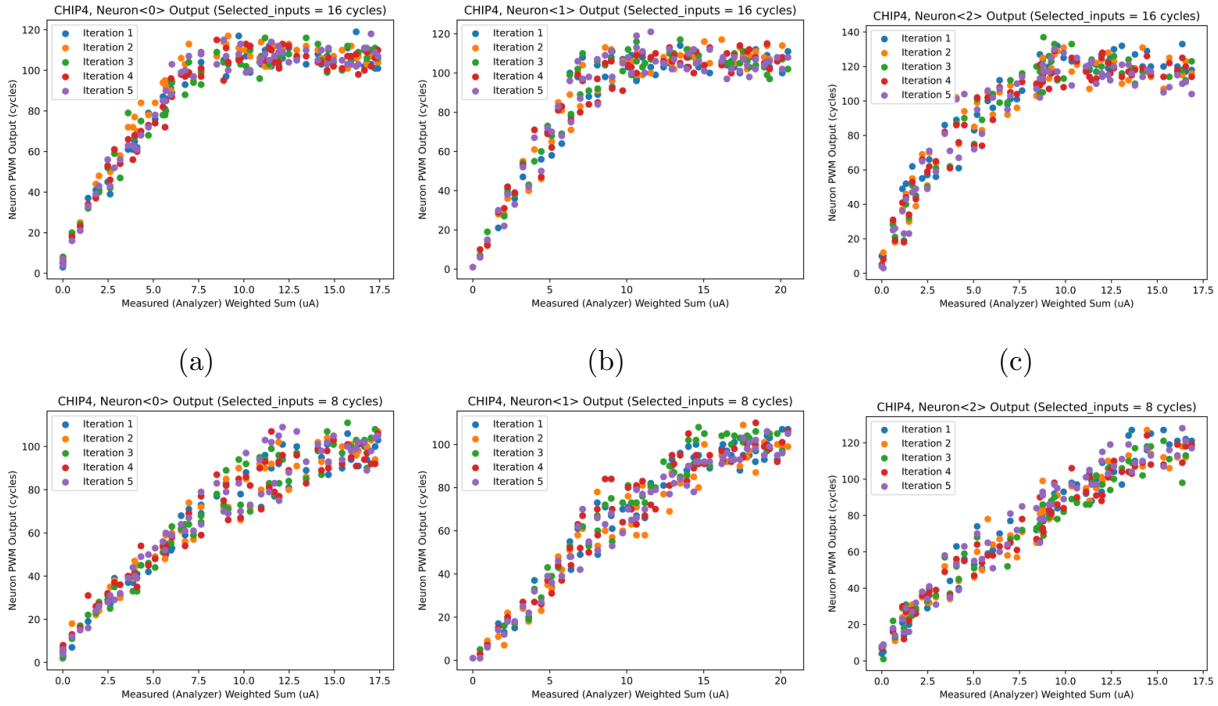
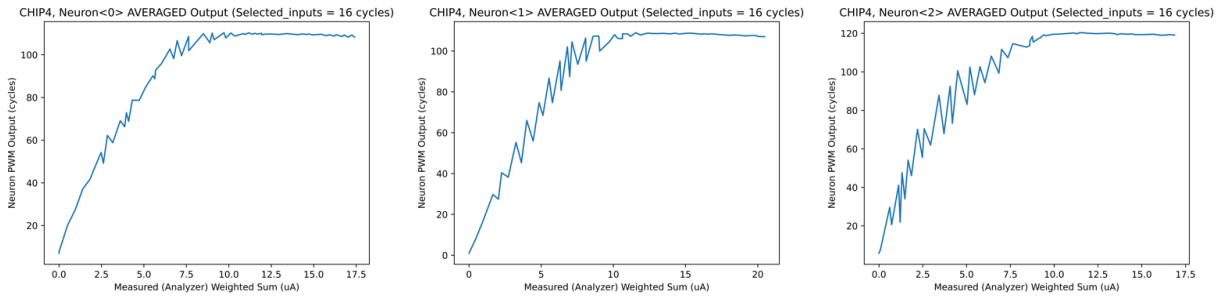


Figure 4.17: ***On-Chip* MAC Engine with Differential Integrator.** All twin-cell CTT devices on $Columns\langle 0{:}\ 2\rangle$ programmed to arbitrary positive weights, similar to those shown in Fig. 4.12. Digitized neuron output is plotted vs. ideal output measured using external analyzer for a random sample of 128 inputs (up to 128 inputs simultaneously enabled). (a-c) represent plots for $Neurons\langle 0{:}2\rangle$ where selected inputs are activated for 16 cycles. (d-f) Represent similar plots except selected inputs are activated for 8 cycles. All plots shown over 5 iterations to better understand output variability. (Chip 4, Column$\langle 0{:}2\rangle$)



(a) (b) (c)

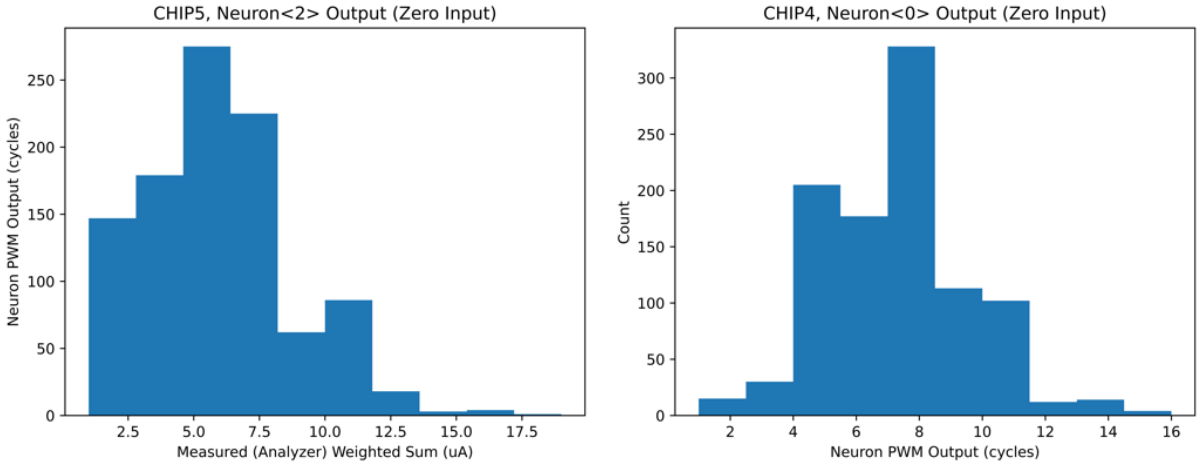
Figure 4.18: ***On-Chip* MAC Engine Neuron Output Averaged over 1000 Iterations.** Experiments similar to those shown in Figures 4.17a-4.17c were repeated for 1000 iterations, then averaged across all iterations. Averaged results show reasonable linearity—especially in (a)—and output saturation, as expected. Potential source(s) of output variability are discussed in the subsequent section. (Chip 4, Column$\langle 0{:}2 \rangle$)

4.3.5 Additional MAC Engine Debug Efforts

While the original neuron circuit was designed to operate at $800MHz$ and integrate a column of up to 256 twin-cell devices, the NeuroCTT 0.3 design specification was simplified with the intention of first validating the entire CTT-based system prior to designing a more complex chip. As a validation chip, it did not include a high-speed CLK IO driver or on-chip PLL—limiting the system testing to lower frequencies (e.g. $20-150MHz$). Future iterations of the chip will require a multi-clock domain logic to decouple the load (e.g. $20MHz$) logic from the inference ($\sim 800MHz$) logic.

The NeuroCTT 0.3 chip was highly successful as all system block functionality was validated including WL driver, CTT Array, CTT Array Column Mux, Array Level-Shifters, etc. and CTT devices were programmable with comparable retention to previously obtained array-level device macro results. Despite demonstrating output linearity and weighted-sum functionality using the on-chip neuron in the previous section, output resolution was lower than expected ($\sim 3b$)—demonstrated by Figures 4.17 & 4.18.

Debug studies were first performed to ensure that device noise from *programmed* CTT devices do not contribute additionally to neuron output variability compared to device noise from *as-fabricated* CTT devices. Figure 4.19 demonstrates that neuron output variability for the zero-input case is nearly identical for a neuron connected to a column of *as-fabricated* devices compared to a neuron that is connected to a column of *programmed* devices. In general, leakage currents for off devices ($V_G = -300mV$) should still be negligible which should not impact the neuron output for the zero-input case as demonstrated in Fig. 4.19.



(a) *As-Fabricated* Devices (b) Programmed CTT Devices

Figure 4.19: **Neuron Output for Zero-Input Case.** Neuron output for zero-input evaluated for neurons connected to a column of (a) *as-fabricated* devices and (b) programmed CTT devices to confirm that device programming is not the source of the neuron output variability. Output evaluated for $1,000$ iterations for both cases and histograms are shown in (a) and (b). No noticeable difference is present (e.g.$\sigma_{(a)} \sim \sigma_{(b)}$). In general leakage currents for off devices ($V_G = -300mV$) should still be negligible.

In addition, neuron output for a column of *as-fabricated* weights is shown in Fig. 4.20. Only *as-fabricated* twin-cell weights with a positive differential current were utilized to demonstrate neuron output linearity. Neuron output variability is similar to that shown previously in Figures 4.17 & 4.18—further demonstrating that device noise from programmed CTT devices does not contribute additionally to neuron output variability. Thus, the output variability is likely a circuit problem instead of a device problem. We explore whether or not testing the chip at low-frequencies is contributing to the output variability.

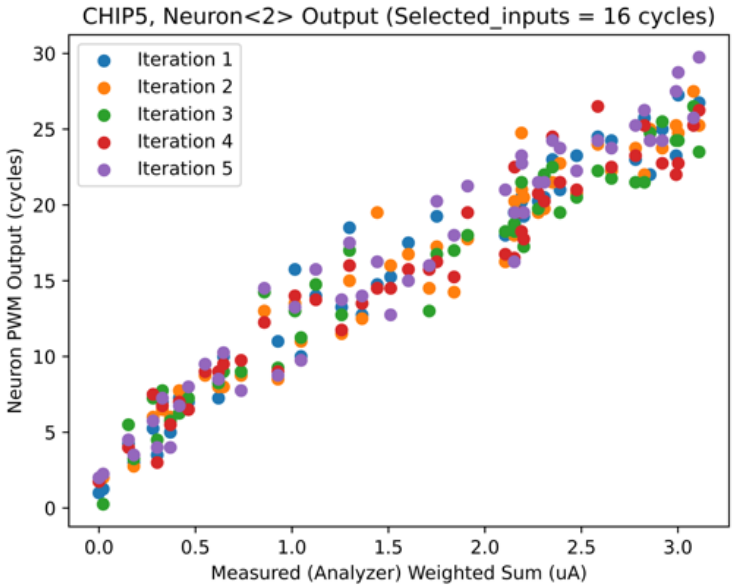


Figure 4.20: **Neuron Output with *As-Fabricated* Weights.** Output shows up to 50 twin-cell weights activated simultaneously with selected inputs set to 16 cycles. Only *as-fabricated* twin-cell weights with a positive differential current were utilized to demonstrate neuron output linearity. Neuron output variability is identical to that shown previously in Figures 4.17 & 4.18.

Testing the chip at low-frequencies (e.g. $20MHz$) limits the valid input range. Assuming that (1) the maximum output capacitance ($C_{INF} = 6.6pF$) is selected and (2) the maximum output voltage is $\sim 250mV$ (obtained from simulation), the maximum charge accumulated during integration is approximately:

$$Q_{MAX} = C \times V_{MAX} = \sim 1.65pC \tag{4.1}$$

Given that $I_{DISCHARGE} = 350nA$, the maximum neuron output at 20MHz was:

$$t_{PWM,MAX} = \frac{Q_{MAX}}{I_{DISCHARGE}} = \sim 5\mu s \sim 100\, cycles \tag{4.2}$$

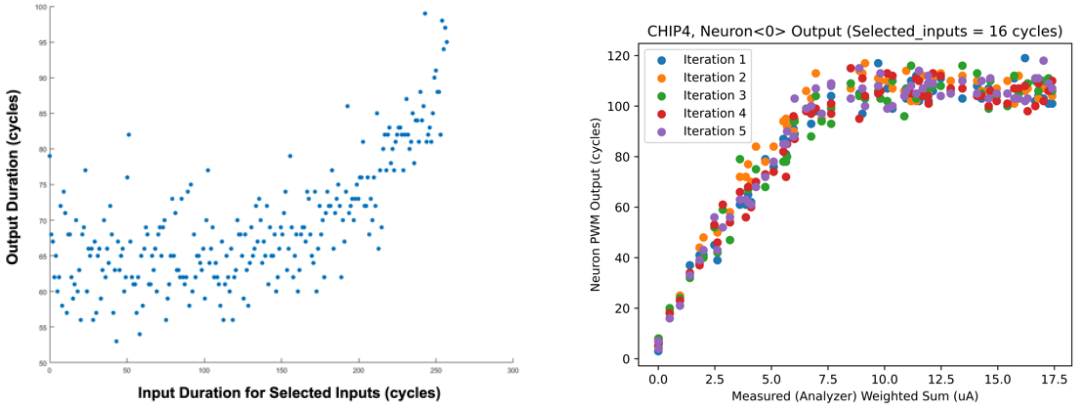
This implies that the maximum differential input current when applied for the maximum 8b input duration (e.g. 255 cycles) is:

$$I_{DIFF,MAX} = \frac{I_{BLt} - I_{BLc}}{2} = \frac{1.65pC}{255 \times 50ns} = \sim 130nA \tag{4.3}$$

Since such a small input current can saturate the neuron output when operated at 20MHz—40× lower than the designed operating frequency for the neuron circuit—the circuit may also be susceptible to noise sources as well. Previous neuron results shown in Fig. 4.17 suggest that the neuron output can fluctuate by $\pm 10\, cycles$ when the output is near saturation (e.g. $\sim 100\, cycles$). Such output fluctuations can be caused by small error currents (e.g. $\sim 10\, nA$) when operating the neuron at low frequencies.

The total time the neuron is on and integrating ($T_{INTEGRATION}$) corresponds to the total length of the inference and the discharge phases. The Inference Duration ($T_{INFERENCE}$) is set to a constant value for all input cases and must be $\geq t_{input,max}$. The length of the discharge phase corresponds to the time it takes to fully discharge the output capacitance, or $t_{PWM,out}$:

$$T_{INTEGRATION} = T_{INFERENCE} + t_{PWM,out} \tag{4.4}$$



(a) $T_{INFERENCE} = 255\,cycles$ (b) $T_{INFERENCE} = 16\,cycles$

Figure 4.21: **Improving Neuron Output Results.** (a) Initial neuron results for 32 simultaneously activated devices. Outputs are plotted against the pulse-width modulated input duration up to 255 cycles. Results are substantially improved in (b) where the inference duration was shortened from 255 cycles down to 16 cycles (results duplicated for convenience from Fig. 4.17a). Respective logic timing diagrams for both cases are provided in Fig. 4.22.

Initial neuron output results are shown in Fig. 4.21. In the first case (Fig. 4.21a), 32 devices were simultaneously enabled for $N = \{0, 1, 2, ..., 255\}$ cycles where $T_{INFERENCE} = 255$ in all test cases. Approximately linear output is only observed for input values beyond ~150 cycles. By shortening the inference duration by 16× and reducing $T_{INFERENCE}$ to 16 cycles, results were substantially improved upon. Figure 4.21b demonstrates clear linear output from the neuron for increasing input values when $T_{INFERENCE}$ was set to $16\,cycles$. Output variability scales for larger PWM outputs—corresponding to $t_{PWM,out}$ in Eq. 4.4. Maximum output variability (±10 cycles) occurs when the neuron output approaches saturation (e.g. $t_{PWM,out} \sim 100$ cycles).

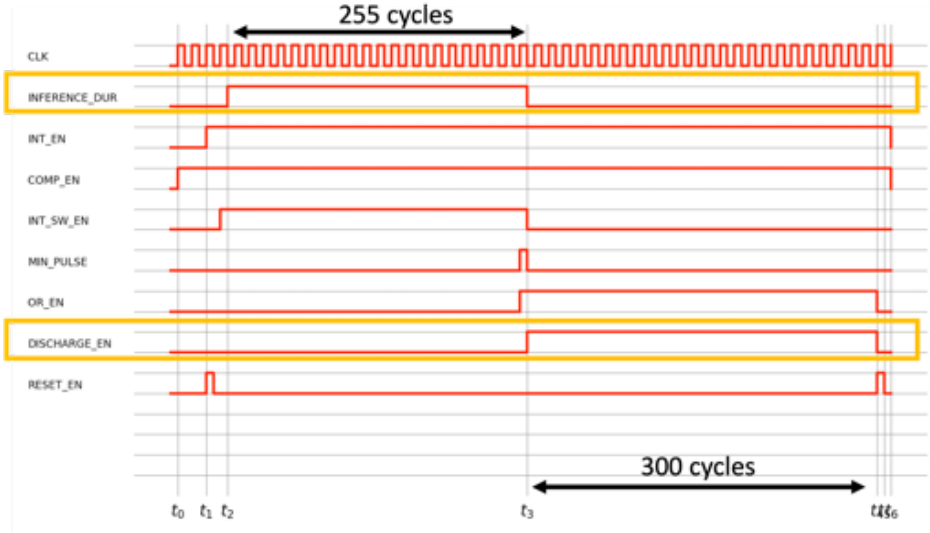
The neuron output variability's dependence on total integration time (Eq. 4.4) suggest that the neuron performance is limited by low-frequency (20MHz) testing constraints. When the inference period is increased from 16 → 255 cycles ($800ns \rightarrow 12.75\mu s$), any integrated

error current can lead to an appreciable error term in the accumulated charge, Q_{INF}. Error currents as small as $\sim 10nA$ may be sufficient to cause a ± 10 cycle output fluctuation. Figure 4.22 provide neuron logic timing diagrams which describe a subset of the related neuron logic signals for the two cases where $T_{INFERENCE}$ equals 255 & 16 cycles, respectively.

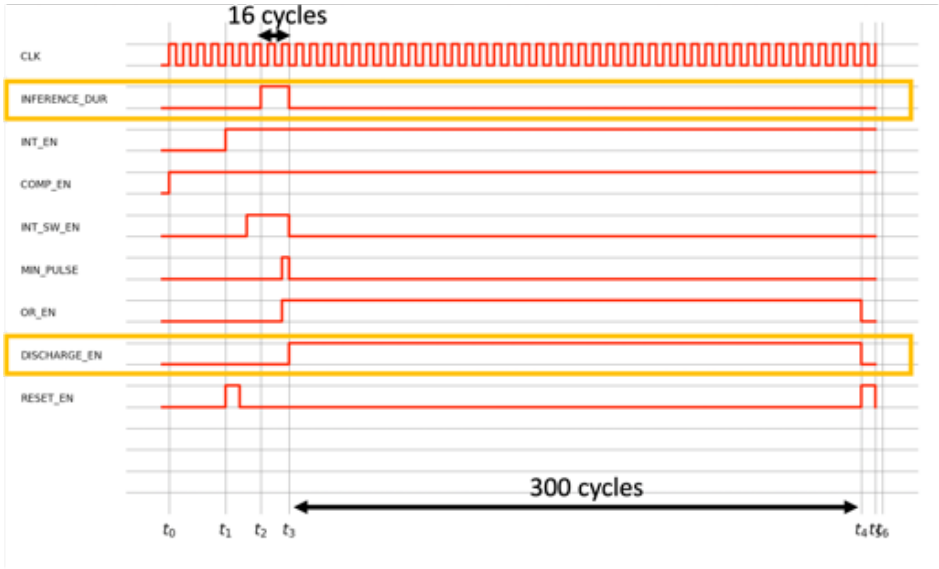
The integrator design was further interrogated via simulation and hardware studies to determine possible sources that may cause the suggested error currents. The integrator circuit relies on 3 external reference currents to properly bias the circuit including an (1) $I_{REF,20\mu A}$ reference for biasing the differential integrator, (2) an $I_{REF,500nA}$ reference for biasing the discharge source, & (3) a $I_{REF,200nA}$ reference for biasing the offset source. The discharge current is utilized to linearly discharge the integrator's output capacitor in order to convert the accumulated charge to an output pulse width (t_{PWM}) and the offset current is utilized to apply DC offset-cancellation settings to counteract process-induced mismatch effects. The three reference currents are replicated using on-chip current mirrors. During the design phase, these bias currents were treated as dc nodes and no internal decoupling capacitors were included. Unfortunately, supply noise coupling to the generated current mirror bias voltages was not considered. External decoupling capacitance was connected to the chip at the board-level on all three nodes with negligible output improvement.

The offset cancellation reference source ($I_{REF,200nA}$) was quickly ruled out as the main culprit as several neurons were testable with offset cancellation *fully-disabled* and the output fluctuations continued to persist without any reduction in magnitude. The discharge current source ($I_{REF,200nA}$) was similarly ruled out as the discharge duration was held constant in experiments where $T_{INFERENCE}$ was varied, previously shown in Fig. 4.21.

The magnitude of the neuron output fluctuations are strongly correlated to the length of the integration period, $T_{INTEGRATION}$. By both reducing the maximum input ($T_{INFERENCE}$) to 16 cycles, the neuron was enabled for a shorter duration leading to substantially improved neuron output results, as shown in Fig. 4.21b. It was determined that likely the differential integrator bias ($I_{REF,20\mu A}$) was susceptible to a small noise current ($<10nA$).

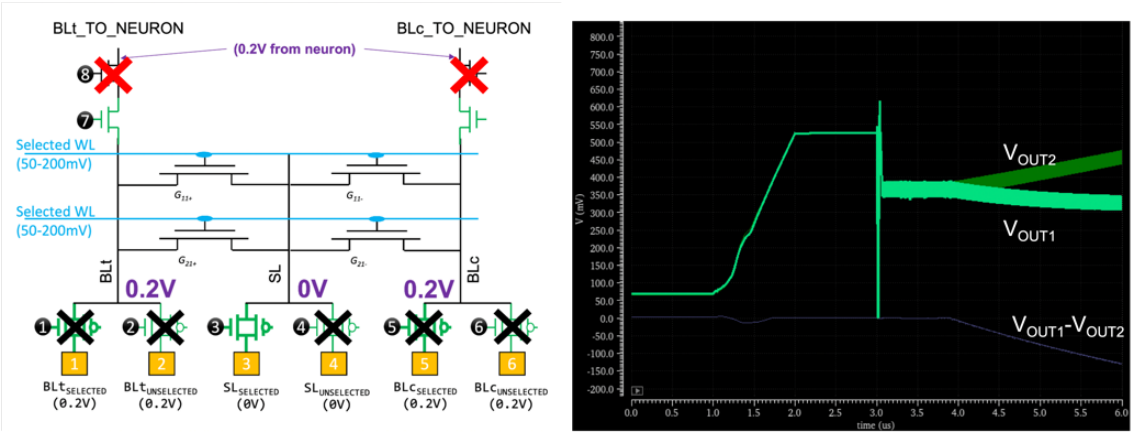


(a) $T_{INFERENCE} = 255\,cycles$



(b) $T_{INFERENCE} = 16\,cycles$

Figure 4.22: **Debugging with Neuron Logic Timing Diagrams.** Diagrams demonstrate the previously mentioned inference duration ($T_{INFERENCE}$) parameter as well as the hardware logic bug associated with the Array Switch (INT_SW_EN) where the array is disconnected from the neuron outside of the inference duration, causing a possible CM instability issue within the integrator, described in more detail in Fig. 4.23.



(a) (b)

Figure 4.23: **CM Stability Issue Introduced by Array Switch.** Logic originally intended to disconnect the CTT array from the neuron to prevent leakage currents when not intentionally integrating may cause CM instability issues within the integrator design. (a) The INTEGRATION_SWITCH corresponds to a switch that connects/disconnects the CTT Array from the neuron. The switch is only enabled during the Inference period, $T_{INFERENCE}$. (b) Simulation results showing possible CM instability. Fortunately, simulation results suggest that given its a CM instead of DM instability issue, it should not substantially affect the output results (see $V_{OUT1} - V_{OUT2}$). V_{OUT1} & V_{OUT2} correspond to the two output nodes associated with the differential integrator's output capacitor.

Additionally, it is worth pointing out a separate design issue that was identified during debugging. Originally, a switch (INTEGRATION_SWITCH_EN) was placed between the neuron and CTT array with intention of disconnecting the CTT array from the neuron when not actively integrating in order to prevent possible leakage currents. It was determined during design verification that this switch may cause CM instability issues for the neuron's differential integrator as it substantially varies the capacitance connected to the input of the circuit. The logic design intended to include both options: (1) leaving the switch always on & (2) enabling the switch only during 'inference'. Unfortunately, a discovered hardware logic bug disabled the former option from properly working. Figure 4.23b demonstrates

in simulation that while substantial ringing can be seen on the two nodes of the output capacitor (V_{OUT1} & V_{OUT2}), it does not have a substantial effect on the differential output result, $V_{OUT1} - V_{OUT2}$. Therefore, it is not expected to be the main culprit of the circuit's output variability. Regardless, it is necessary to resolve the issue in a future design iteration.

4.3.6 Final Insights

To reiterate, the NeuroCTT 0.3 chip was highly successful. All system-level blocks were fully verified including the WL drivers, CTT Array, Array Column Mux, Level-Shifters, etc. and can be reused as is in subsequent tapeouts. CTT devices array were efficiently programmed and results match previously demonstrated programming capabilities (e.g. ΔV_{TH}) shown on the discrete device and array macros. For the first time, a Pulsed-Voltage Time Sweep (PVTS) programming methodology was demonstrated which allows for fine-tune programming by simply tuning the programming pulse width rather than requiring ramping voltage supplies as is the case for the PVRS methodology. For the first time, an accurate & reproducible MAC Engine with programmed weights was demonstrated utilizing all on-chip components and an external Keysight Analyzer for measuring the total summed output current. For the first time, a MAC Engine with programmed weights and an on-chip neuron circuit & ReLU activation function were demonstrated. While neuron output resolution was limited, neuron functionality and linearity were properly demonstrated.

In the short-term, results from the NeuroCTT 0.3 chip can be further improved by exploring a couple different avenues. First, the PCB will be re-designed with a few modifications to fully automate the iterative fine-tune weight programming. Secondly, opportunities exist to further increase the on-chip clock frequency beyond 20MHz by optimizing FPGA timing parameters. It is also possible that varying the clock frequency might be possible by slowing it down to reliably load data onto the chip, then increasing the frequency during inference.

In the long-term, a re-spin will be required to fully demonstrated the concept and overall energy-efficiency. A subsequent design would require (1) PLL or high-speed CLK IO driver, (2) multi-clock domain to decouple the data load and 'inference' logic, (3) thorough decap placement on sensitive nodes, and (4) additional neuron modifications to further reduce mismatch-related issues. It is also worth demonstrating a multi-layer design where the neuron output is directly applied as an input to a subsequent layer.

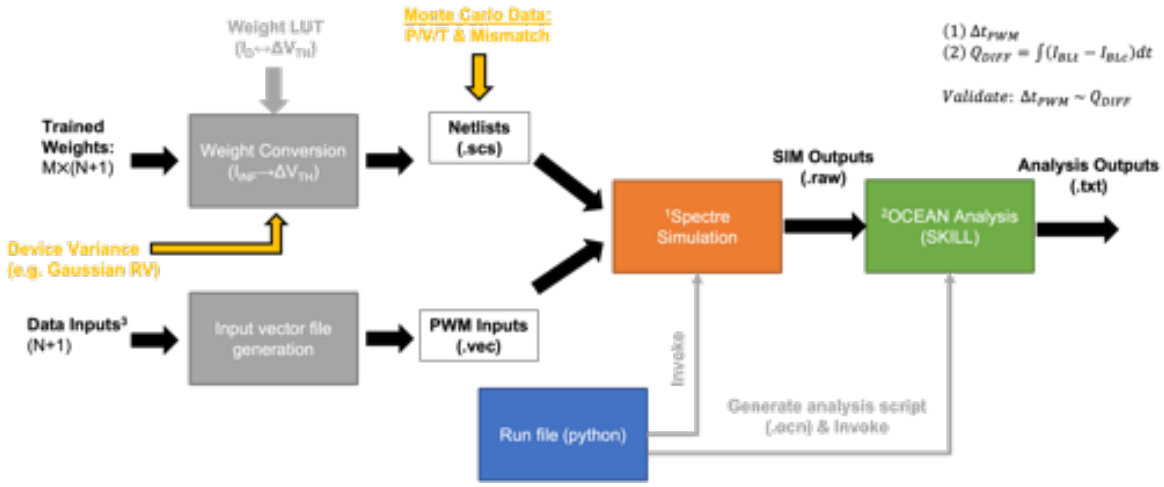


Figure 4.24: **CIRCUS Functional Overview.**

4.4 CIRCUS Hardware Simulator

CIRCUS (*CTT-Hardware-based Inference Realistic Circuit Universal Simulator*) is a simulation platform developed to study the effects of circuit-induced errors and device non-idealities on overall system performance. An example CIRCUS implementation flow for a local monte carlo (localmc) simulation including P/V/T and foundry-provided mismatch data is provided in Appendix C. A functional overview is provided in Fig. 4.24.

Twin-cell device weights correspond to the differential channel conductance (ΔG_{CH}) or on-current (ΔI_{WEIGHT}) for a given subthreshold bias point such as $V_{GS} = 200mV$, $V_{DS} = 200mV$—the weight for a single device as a function of different programming and erase steps is shown in Fig. 4.25a. Weights are encoded in simulation by utilizing a parameterized voltage source to encode the ΔV_{TH} for the True and Comp. devices, respectively, shown in Fig. 4.25b. A Lookup-Table (LUT) is utilized to map between device weights and the corresponding ΔV_{TH} value. Trained network weights are mapped to CTT devices using a Weight Conversion Script which generates spectre netlist files, shown in the overview provided by Fig. 4.24. Device variance can also be included by adding a normally-distributed error term to each of the CTT device conductances or threshold voltages (G_{CH} or V_{TH}).

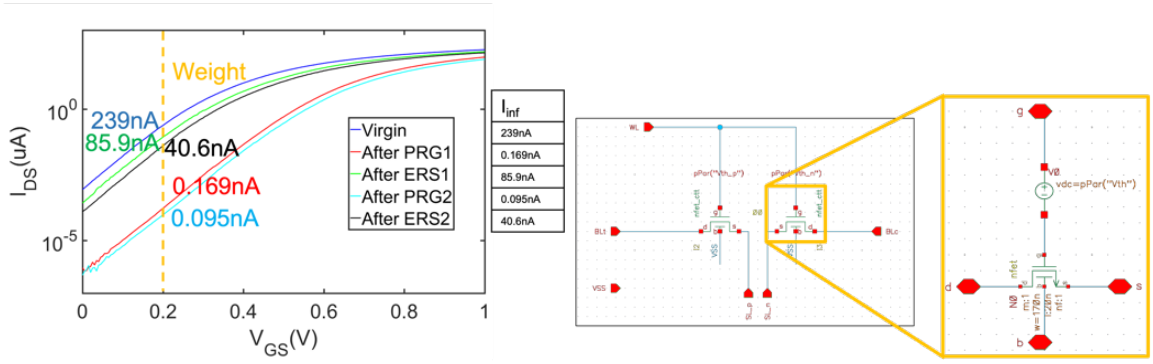


Figure 4.25: **CIRCUS CTT Device Model.** The CTT device utilizes the standard foundry-provided *nfet* device model. As shown in (a), a programmed CTT device can be modeled as an *as-fabricated* device with some threshold voltage shift, neglecting gate leakage & subthreshold degradation. In simulation, a parameterized voltage source is attached to the gate of the CTT device shown in (b), allowing for a programmed device 'weight' to be included.

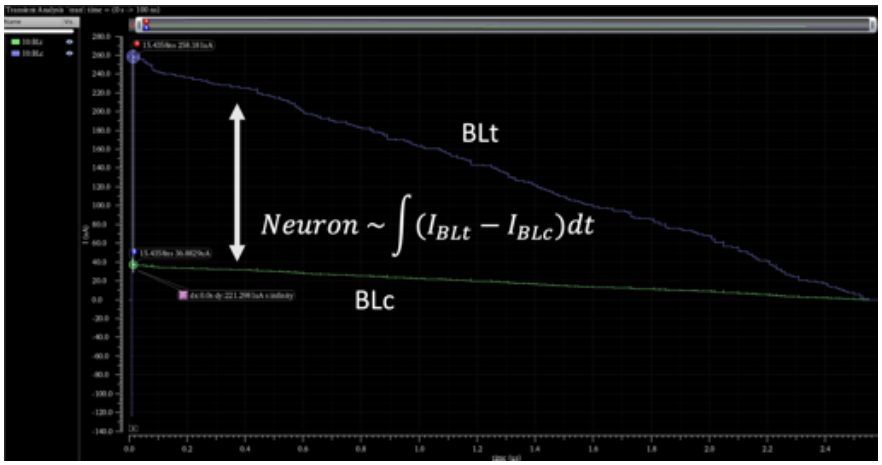
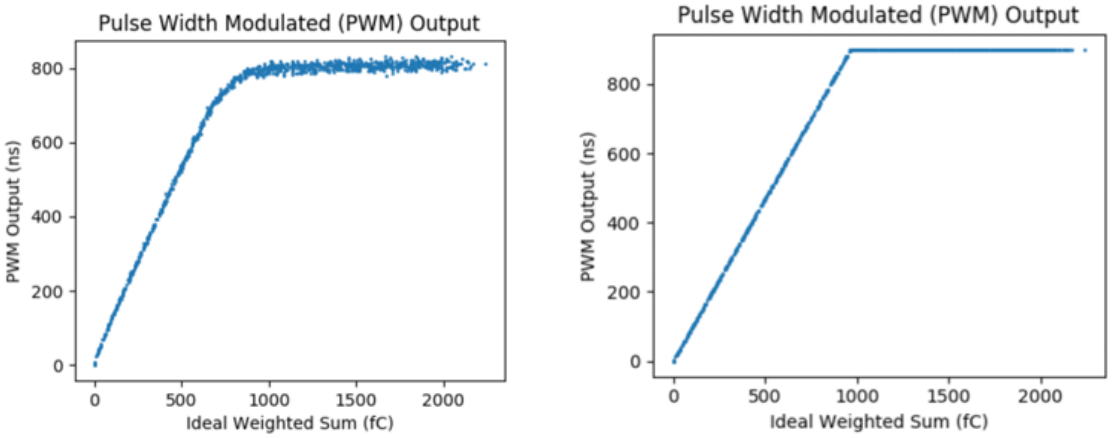


Figure 4.26: **CIRCUS Example CTT Array Differential Current Waveform.** Differential current waveform from CTT array is then integrated by the Neuron Circuit which produces a pulse-width-modulated output signal proportional to the total integrated current $t_{PWM} \sim \int_0^t (I_{BLt}(t) - I_{BLc}(t))\, dt$.



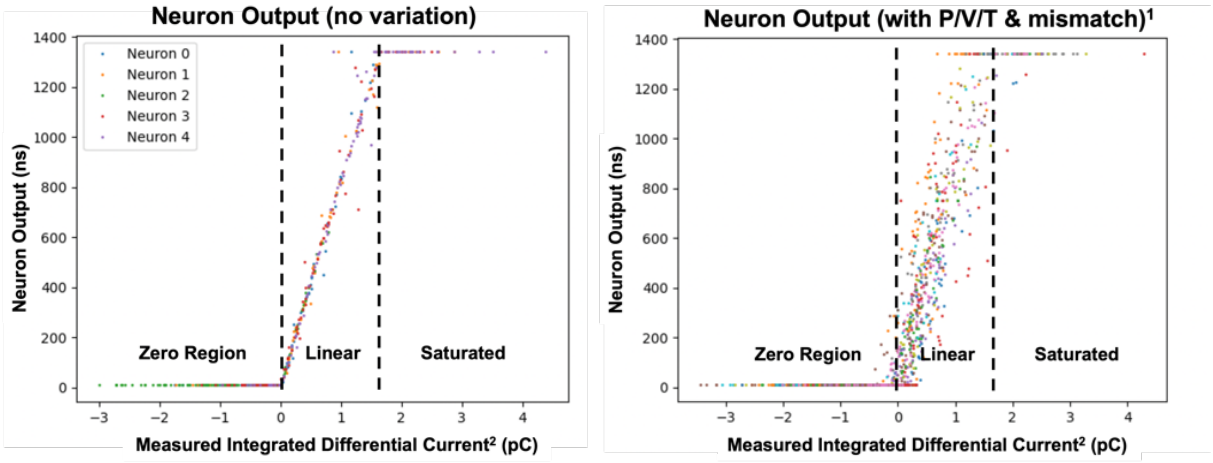
(a) Spectre Model

(b) VerilogA Model

Figure 4.27: **CIRCUS Neuron Output Simulation.** (a) Spectre-based Neuron Design with $C_{INF} = 1.8pF$ integrating capacitor was evaluated across 1000 different randomly-generated input patterns. Input patterns converted to 'Ideal Weighted Sum' by mathematically integrating the $I_{BLt}(t) - I_{BLc}(t)$ signal presented at the neuron integrator's input. (b) Equivalent VerilogA model can be used to speed up simulation run-time. In both cases, the neuron pulse-width-modulated (t_{PWM}) is zero for negative weighted sums and saturates for weighted sums $> 1pC$.

An example differential current input with positive weights and random PWM inputs for a 256×1 network is shown in Fig. 4.26. The neuron is tasked with integrating the differential input current and generating an output pulse that is proportional to the integrated result or 'weighted sum', $t_{NEURON,PWM} \propto \int_0^t (I_{BLt}(t) - I_{BLc}(t))dt$.

The CIRCUS simulator offers Spectre and Verilog-A models of the neuron circuit design. The Spectre model provides the actual circuit taped-out in the most recent NeuroCTT 0.3 chip where P/V/T effects are studied, while the Verilog-A model offers an idealistic model that can substantially accelerate simulation run-time. Example neuron output results for both models are shown in Fig. 4.27. The Spectre-model is mainly utilized for thorough design verification over a variety of corners.



(a) Without P/V/T

(b) Considering P/V/T & Mismatch

Figure 4.28: **CIRCUS Simulation for Evaluating Realistic Neuron Accuracy.** (a) Output of 5 Neuron & CTT Column Models with randomly initialized weights and 100 randomly supplied inputs without considering P/V/T effects. (b) Simulation run across 32 Randomly Initialized Neuron Models considering P/V/T effects and foundry-provided mismatch. [1]Each neuron model considering mismatch was manually calibrated using the offset calibration schemes to provide zero output for the zero-input scenario. [2]Measured Integrated Differential Current was calculated by integrating the differential input current from the CTT Array, $\int_0^t (I_{BLt}(t) - I_{BLc}(t))\, dt$.

Figures 4.28a & 4.28b clearly show that the neuron output consists of a (1) Zero region, (2) Linear region, and (3) Saturated region. The Zero & Linear Regions realize the Rectified Linear Unit previously shown in Fig. 3.23. The post-calibration Neuron Output when considering P/V/T and mismatch, shown in Fig. 4.28b, appears to be somewhat noisy, but for any given neuron is fairly consistent and linear. It's important to point out that offsets exist in both the neuron's differential integrator as well as differential comparator circuits. Additional offsets beyond those that can be specifically corrected for using previously mentioned offset cancellation schemes in Section 3.1.6, can be compensated for by programming the CTT devices to implement 'effective' weights that produce a specific desired output from the neuron circuit. This type of weight verification is considered *On-Chip* Verification where the neuron circuit during Inference can be repurposed to read out individual device weights by enabling a single twin-cell CTT device for a fixed input duration.

CHAPTER 5

Conclusions & Outlook

Crossbar architectures with nonvolatile weights and analog computation have the potential to greatly disrupt traditional hardware design targeting ML workloads. While this dissertation, focuses mostly on edge applications it is possible if a series of technical problems are addressed, crossbar architectures may become technologically & commercially viable for larger scale applications.

This dissertation provides a first-time demonstration of an *on-chip* analog MAC engine using the CTT as a nonvolatile *analog* synapse—fabricated in a commercial CMOS technology (GlobalFoundries 22FDX). Excellent device programmability is demonstrated on a group of 480 programmed CTT devices with $\sim 5b$ twin-cell resolution and worst-case normalized programming variance σ'_{PRG}<6%. Additionally, sufficient retention characteristics are reported with minimal change in programming variance and mean drift after baking for 50 hours at $85°C$. CTT devices are shown to be highly stable with <1% measurement error reported for 300 measurements taken on a single device over a 48 hour period.

For the first time, an *on-chip* CTT-based MAC Engine utilizing an external Keysight B1500A Analyzer was able to demonstrate highly accurate & linear weighted sums using programmed differential weights. Additional, an On-Chip CTT-based MAC Engine with an *on-chip* neuron comprising of differential integrator and comparator circuits was able to demonstrate linear output with $3b$ output resolution—limited by testing constraints and low-frequency operation. Further improvements are required to improve the resolution to the desired $6-8b$ output resolution including improving decap placement on sensitive bias/sup-

ply nodes and logic modifications to support testing at higher frequencies—up to $800MHz$. While reliable weighted sum computation was demonstrated, future work must also demonstrate energy-efficiency by performing this computation at speed (e.g. $800MHz$) and by properly power-gating digital & analog circuitry when idle.

For the first time, A Pulsed-gate Voltage Time Sweep (PVTS) programming methodology is introduced for *fine-tune* programming without requiring ramping voltage supplies as is required with the PVRS programming methodology. Further studies are required to determine whether a PVTS methodology can be exclusively used to accurately program devices.

An analysis of all the variances present is in a CTT-based analog in-memory compute (IMC) engine is provided, including the device programming variance, measurement error and device fluctuations due to effects such as RTN and $\frac{1}{f}$ noise, weight retention, temperature-dependent effects, circuit-induced non-idealities, and radiation effects such as Total-Ionizing Dose (described in more detail in Appendix A). The CTT-Hardware-based Inference Realistic Circuit Universal Simulator (CIRCUS) is introduced to study the net effect of all of these sources of variability on the overall systems output and validate that sufficient output resolution can still be obtained.

Finally, the experimental radiation tolerance of neuromorphic hardware is discussed in Appendix A. The IBM TrueNorth Neurosynaptic System is utilized as a baseline system for evaluating the resiliency of pseudo-non-von Neumann digital architectures where the IBM TrueNorth is found to be highly resilient to Single-Event Upsets (SEUs) or *soft errors*. The CTT device is evaluated as a candidate device for analog-based neuromorphic hardware. Given the nonvolatile nature of the CTT-based weights, it is found that the CTT is resilient to *soft errors* but is susceptible to large ΔV_{TH} shifts due to Total-Ionizing Dose (TID) which is especially more pronounced in fully-depleted SOI technologies. A related discussion on the effects of radiation on 3D-stacked technologies such as 3DS-DRAM Memories is provided in Appendix B which will become especially more relevant as larger systems continue to

adopt newer packaging and system technologies to meet the computational demand of new workloads. Future work is required to develop characterization methods for more complex 3D-stacked systems with increasingly more complex failure modes.

In summary, the CTT is highly competitive against other proposed analog NVM devices such as RRAM, PCM, SONOS, & STT-MRAM, just to name a few. The CTT is completely CMOS-compatible and exists as a 3-terminal device requiring no additional select transistor with sufficiently high $\frac{I_{ON}}{I_{OFF}}$ ratio. It has been demonstrated in all *gate-first* high-κ technology nodes and offers a sufficiently high on-resistance ($R_{ON}{\sim}M\Omega$) in the selected 22FDX technology node. Excellent device programmability & weight resolution as well as sufficient weight retention at higher temperatures has been demonstrated. CTT can be reliably *fine-tuned* to target weights and utilized in a MAC Engine to perform reliable & reproducible computation with nonvolatile weights.

5.1 Outlook

In order to demonstrate scalable crossbar architectures for larger applications beyond leaf applications, In-Memory Computing research must tackle a series of challenging technical problems including system scaling & network reconfigurability for rapidly evolving network topologies beyond Fully-Connected or Multi-Layer Perceptron (MLP) networks. Analog IMC-based compute are currently difficult to scale with a trade-off between reconfigurability and energy-efficiency. Scalable and reconfigurable systems almost certainly require a digital interface and/or router between network layers potentially, which may eliminate the energy benefits of performing computation in the analog domain. In addition, today's crossbar architectures are not able to as effectively exploit network sparsity as some of today's digital systems.

Designers utilizing pulse-width modulated (PWM) schemes, such as the one utilized by the NeuroCTT, must consider the trade-off between area and output resolution. Addition-

ally, while pulse-based architectures might be substantially smaller in size and simpler to design, they are typically not able to provide comparable throughput at the same input/output resolution. While pulse-based input architectures might suffer from lower throughput, they are promising for edge applications with smaller computational requirements. Pulse-based input architectures are able to leverage a simpler, lower-power, and smaller-area ADC in lieu of an integrator-based ADC, offering superior energy-efficiency.

Finally, crossbar architectures with larger number of inputs may suffer from a large common-mode to differential-mode ($\frac{I_{CM}}{I_{DM}}$) ratio leading to design challenges. As an example the NeuroCTT system may be required to handle input cases where the measured $\frac{I_{CM}}{I_{DM}} \sim 1000\times$, requiring a higher-power integrator with larger bias currents to properly manage the large common-mode currents.

5.2 Future Work

In the short-term, additional results may be obtainable from the current hardware by exploring a couple possible testing avenues. First of all, a simple PCB board redesign is required to fully automate the *fine-tune* program-verify algorithm using external Keysight B1500A analyzer for weight verification. Secondly, it is possible that the system frequency can be increased beyond $20MHz$ by (1) optimizing the load logic & timing between the FPGA and NeuroCTT 0.3 chip and (2) modulating the clock frequency. By modulating the clock frequency, it is possible that the clock frequency could be slowed down while loading data onto the chip and increased when performing inference computation.

A future tapeout will be required to demonstrate the concept at speed (e.g. $\sim 800MHz$) with sufficient output resolution and competitive energy-efficiency. Thorough block and system validation has been a significant result from this project. All of the included circuit blocks on NeuroCTT 0.3 have been fully verified as functional including the WL Drivers, CTT Array, Array Column Mux, Neuron, Time-to-Digital Converters (TDCs), and system

logic. Further, experiments were performed that demonstrate that devices within the NeuroCTT System's CTT Array can be efficiently programmed with comparable programming results to those obtained using the discrete & array device macros. These blocks can be fully reused in a future tapeout with the exception of the neuron block. While the neuron was verified at 20MHz with linear output, modest modifications are required to ensure proper output resolution may be obtained when testing at higher frequencies. These modifications include additional decoupling capacitor placement on sensitive bias & supply nodes as well as some device resizing to better handle possible device mismatch issues. In addition, a simple re-spin would require multi-clock domain logic (e.g. $SYS\_CLK = 20MHz$, $INF\_CLK = \sim 800MHz$) to decouple the load and inference logic as well as a high-speed CLK IO driver or licensed PLL IP block.

APPENDIX A

Effect of Memory-Related Errors in Neuromorphic Hardware

A.1 Digital-based Neuromorphic Computation

In contrast to analog-based neural networks, the nature of errors in digital systems is inherently different. In the absence of errors, a digital, non-stochastic system is almost always deterministic. For large digital systems, the dominant error or failure mechanism is often considered to be errors created within or while accessing the memory. These errors also include *soft errors* or Single-Event Upsets (SEUs) where high-energy particle strikes and corrupt data within the memory. On-chip Dynamic Random Access Memory (DRAM) and Static Random Access Memory (SRAM) are highly susceptible to bit errors, forcing server-grade and more complex systems to rely heavily on Error Correction Codes (ECC) and Cyclic Redundancy Checks (CRC) to detect errors and prevent instruction & data corruption due to a memory error [Ham50, PB61].

The IBM TrueNorth Synaptic Platform [Ami13, Mer14, Ess16, Saw16, Fur16] is utilized as a baseline system for evaluating the effects of errors on digital-based neuromorphic platforms, where memory-related errors are assumed to be the dominant failure mechanism. Non-ionizing radiation is utilized (1) as a means for injecting random faults or *soft errors* into the on-chip network and (2) studying the overall system tolerance to radiation effects for use in strategic environments.

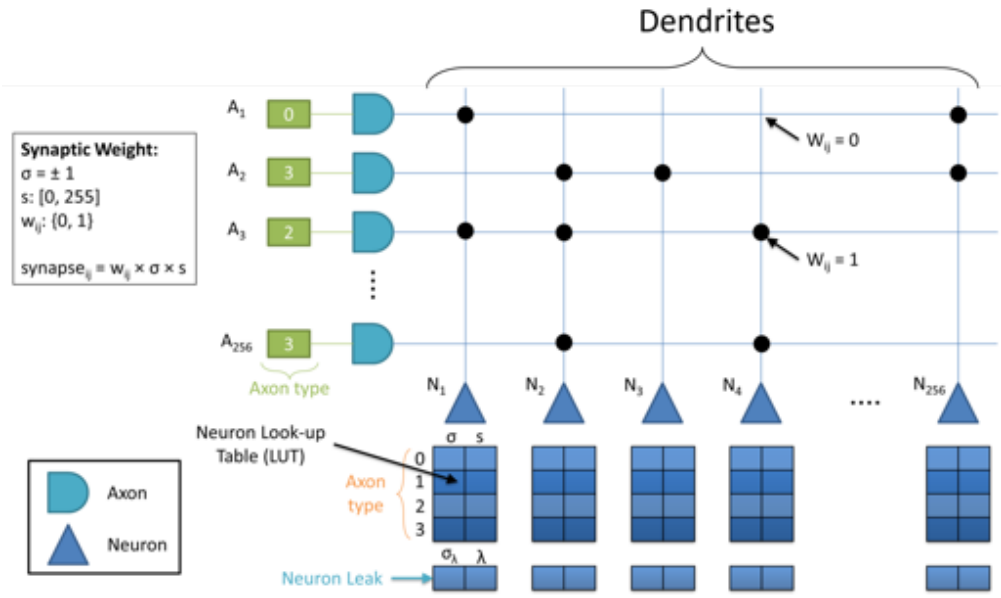


Figure A.1: **IBM TrueNorth Core Implementation.** TrueNorth chip consists of $4,096$ of these cores requiring 410Mb of on-chip SRAM storage and spans $\sim 400 mm^2$ in Samsung 28nm technology.

A.1.1 TrueNorth Architecture

Figures A.1, A.2, & A.3 provides information on the architecture of the IBM TrueNorth Neurosynaptic System. The TrueNorth consists of $4,096$ cores that emulate $\sim 1\,M$ neurons and $\sim 256\,M$ synapses while consuming $\sim 70\,mW$ at run-time. Each core includes 256 axonal inputs, a 256×256 synaptic crossbar, and 256 neuron outputs. Each neuron output represents a weighted-sum of the respective 256 axonal inputs:

$$V_j = \sum_{i=1}^{256} w_{ij} x_i \tag{A.1}$$

The IBM TrueNorth supports integer synaptic weights ranging [-255, 255]. A trade-off exists between programmability and the synaptic density. In order to co-optimize programmability and synaptic density, the TrueNorth utilizes 4-element synaptic weight lookup tables

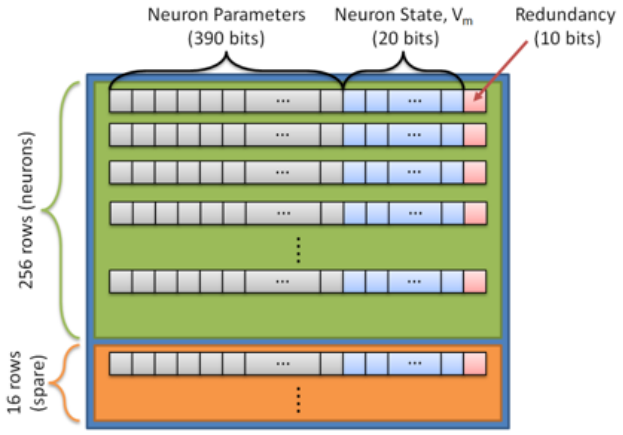
per neuron, hence supporting a maximum of 4 *unique* synaptic weight values per neuron (synaptic crossbar column), as shown in Fig. A.1. The chip stores a 2-bit axon type parameter for each synapse to index the lookup table and obtain its respective synaptic weight. Synaptic weights correlate the pre-synaptic neurons (axon inputs) to the post-synaptic output neurons, whose values are determined by the training of the network.

TrueNorth models are trained offline and then loaded onto the chip. Model parameters are stored in SRAM banks per core. Each SRAM bank consists of 102.5 Kb of memory, excluding redundancy bits and spare rows, as shown in Fig. A.2a. Figure A.2 provides detailed SRAM breakdown by model parameter type. Essential parameters include Synaptic weights, Axon Types, Neuron addressing, membrane potential, and membrane threshold parameters. Other additional stochastic parameters—classified under 'Other Neuron Parameter'—are available for emulating brain spiking patterns but are not necessary for running traditional neural network tasks such as Convolutional Neural Networks (CNNs).

A single TrueNorth chip consists of 4,096 *cores* arranged in a 64×64 array, depicted in Fig. A.3 (*middle*). The system includes a 24-bit addressing space which allows any neuron output to be routed and connected to any axonal input. This 24-bit address variable consists of 16-bit destination core address and an 8-bit destination core address, which is sufficiently large enough to support multi-chip configuration up to 16 chips (4×4), such as demonstrated with the TrueNorth NS-16e boar [Fur16].

A.1.2 TrueNorth EEDN Framework for CNNs

Convolutional Neural Networks (CNNs) can be implemented on the IBM TrueNorth using the Energy-Efficient Deep Neuromorphic (EEDN) networks training algorithm provided in [Ess16]. EEDN utilizes trinary weights, {-1,0,1}, to train and implement multi-layer CNNs specifically mapped for the IBM TrueNorth hardware, elaborated upon in Fig. A.4. Trinary weights help simplify the training difficulty given the synaptic weight programmability constraints alluded to earlier (e.g. 4-element LUTs per neuron). Each core on the chip can be



(a) SRAM per core

SRAM Component	Total Size
Total SRAM Memory	**410 Mb**
Synapses	**292 Mb**
Synaptic Crossbar (w_{ij})	256 Mb
Synaptic Weights (σ, s)	36 Mb
Axon	**2 Mb**
Axon Type	2 Mb
Neuron Addressing	**30 Mb**
Destination Core Address	18 Mb
Destination Axon Address	8 Mb
Axonal Delay	4 Mb
Other Neuron Parameters	**86 Mb**

(b) SRAM Model Parameters

Figure A.2: **IBM TrueNorth SRAM Design per core.** (a) Each core consists of 102.5Kb ($256\,rows \times 410\,bits$) of SRAM, not including redundancy bits and spare rows. Entire chip consists of 410Mb of SRAM ($4096\,cores \times 102.5\,Kb$). (b) TrueNorth Chip SRAM Breakdown by Model Parameter. SRAM memory is dominated by synaptic weight ($292\,Mb$) and neuron addressing ($30\,Mb$) parameters. Some of the other available neuron parameters such as stochastic parameters (e.g. neuron leakage, λ) are not required for solving traditional CNN-based classifications tasks using the TrueNorth.

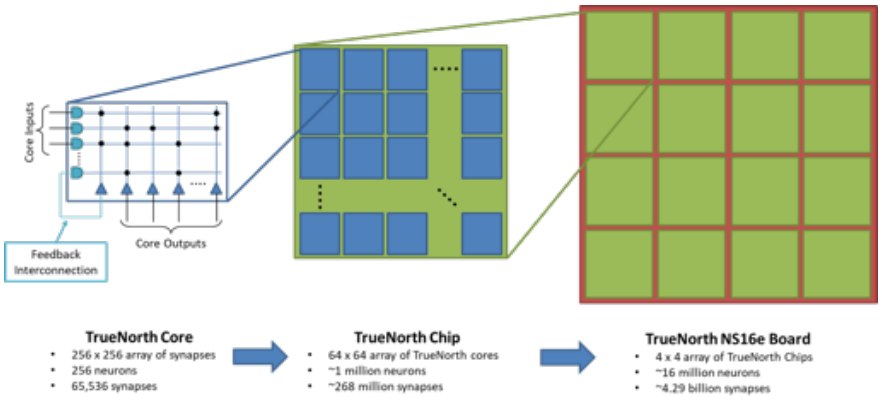


Figure A.3: **IBM TrueNorth Multi-Chip Configuration.** The IBM TrueNorth chip utilizes an address space that supports scaling the system up to a 16-chip (4×4) system. IBM has demonstrated the 16-chip configuration, known as the IBM TrueNorth NS-16e board, capable of running model files that emulate up to 16 million neurons and >4 billion synapses [Fur16].

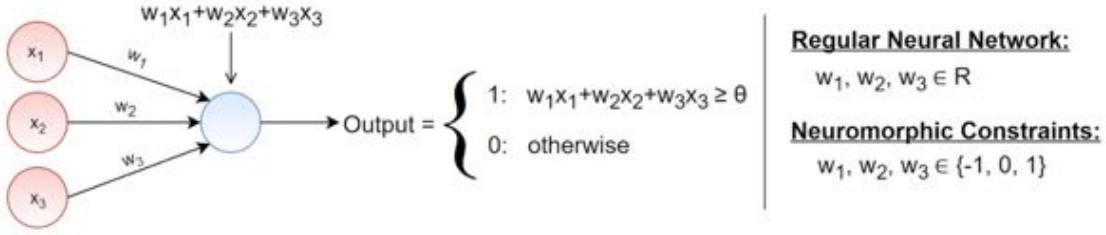


Figure A.4: **IBM TrueNorth EEDN Trinary Weight Constraint.**

used to implement one filter ($nFeatures = 1$) in the network.

Not all TrueNorth cores, however, can be utilized by the CNN. Since each neuron output can only be configured to transmit output spikes to a single-fixed address (axon input), splitter cores are required. Splitter cores require a large overhead as an entire core is required to replicate one neuron output up to 256×. By definition, each Convolutional Network Layer output is reused multiple times by subsequent layers as the filter is convolved across its input space for the next respective layer, requiring many splitter cores to implement most CNNs.

For the purpose of this work, MNIST, CIFAR-10, and CIFAR-100 datasets were utilized

(a) MNIST [LeC98, LCB10] (b) CIFAR-10 [Kri09] (c) CIFAR-100 [Kri09]

Figure A.5: **Commonly-Used Classification Datasets.** (a) MNIST dataset consists of 60,000 training images and 10,000 test images representing handwritten digits $(0-9)$. (b) CIFAR-10 dataset consists 50,000 training $(5,000/class)$ and 10,000 test $(1,000/class)$ images representing 10 different classes of objects. (c) CIFAR-100 dataset consists of 50,000 training $(500/class)$ and 10,000 test $(100/class)$ images representing 100 different classes of objects.

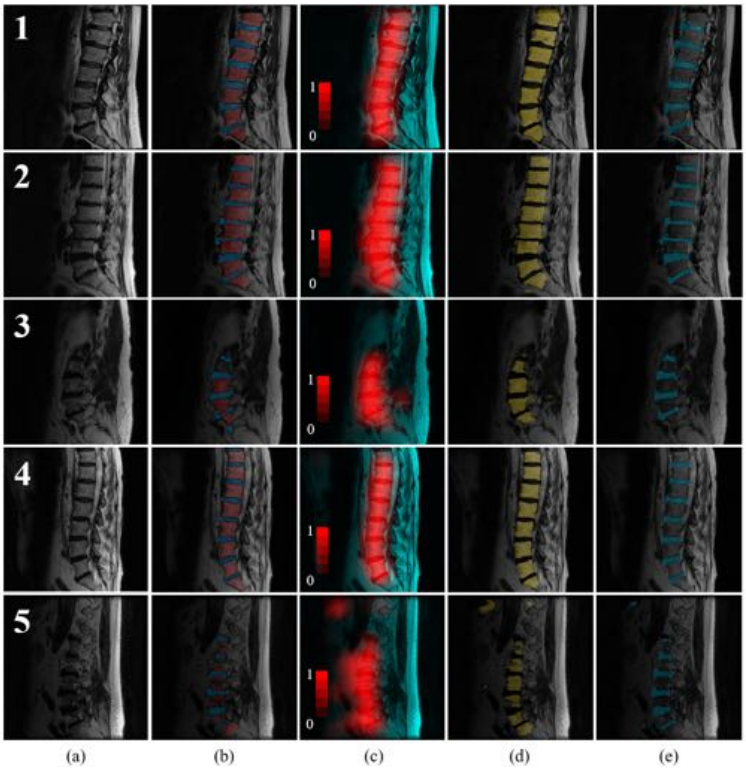


Figure A.6: **IBM TrueNorth for Spinal MR Image Segmentation Example.** Originally presented in [MW18]. (a) Original MR images. (b) Manual Segmentations perform by human rater. (c) Detection (localization) results where red pixels designate areas identified as belonging to the spinal canal. (d) Automated vertebrae segmentations. (e) Automated disk segmentations.

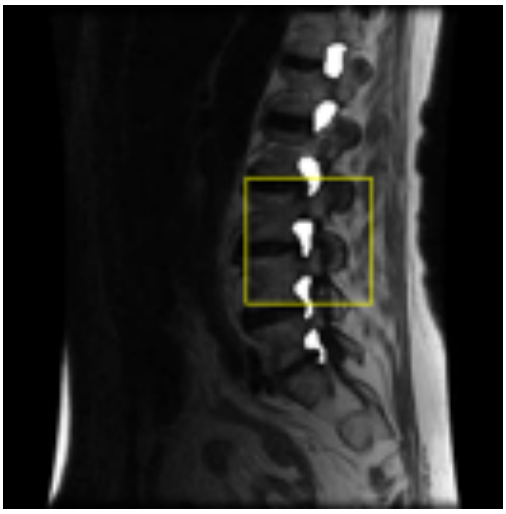


Figure A.7: **IBM TrueNorth Spinal Foramina Segmentation Example.** A similar network to that shown in [MW18] can also be utilized to segment more complex imaging features such as neuroforamina in the spinal canal which can be utilized to quantify and diagnose spinal-related diseases such as neuroforaminal stenosis as is discussed in [Gao19].

to study the effect of *soft errors* on CNNs running on a neuromorphic architecture such as the IBM TrueNorth. Additional work presented in [MW18], [WM18], & [Gao19] demonstrate that low-power neuromorphic platforms such as the IBM TrueNorth can be utilized for even complex medical segmentation tasks to identify features for automated image-based diagnosis, shown in Figures A.6 and A.7.

A.1.3 Experimental Setup: Vanderbilt Pelletron

The Vanderbilt Pelletron accelerator [McC15] offers several particles & ions with varying energy levels and range. Commonly used recipes and achievable ranges are detailed in Table A.1. As shown, most options provide a range of at most $10 - 100\mu m$. The IBM TrueNorth die is covered by an organic molding compound with a radiation absorption thickness of $t_{rad} = \sim 0.9mm$, requiring the chip to be chemically delidded prior to irradiating using the Vanderbilt Pelletron.

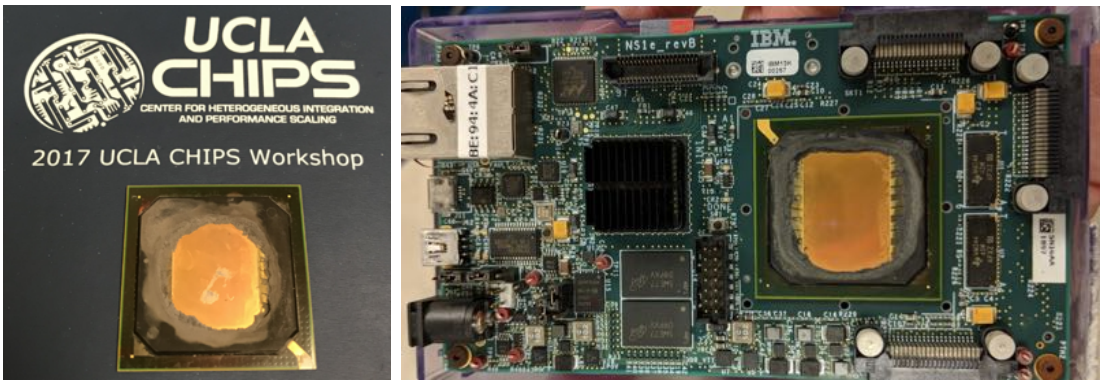
Incident Ion and Max. Energy	Metal				
	Al	Cu	Si	SiO2	W
Proton (4MeV)	130	54	148	182	40
Alpha (6MeV)	27.9	12.9	31.9	38.4	10.2
Oxygen ion (14.3MeV)	8.75	4.33	9.86	12.2	3.43
Chlorine ion (16.4MeV)	5.66	3.04	6.28	8.06	2.35

Table A.1: **Range of Potential Ions/Particles using the Vanderbilt Pelletron.** Ranges provided in μm.

The IBM TrueNorth was chemically delidded using foaming nitric acid followed by an acetone cleaning process. Protective tape was used to protect the surrounding board components. Delidding results are shown in Fig. A.8. Although the nitric acid is an oxidizing agent, the bond wires are gold (Au) instead of Copper (Cu), so it did not pose any issues to wirebond quality. Chip Back-End-of-Line (BEOL) thickness was estimated at $\sim 8\mu m$ which is supported by the range available of both Proton and Alpha particles. Ultimately, 4 MeV protons with varying fluxes were selected for all experiments.

The TrueNorth NS1e board was placed and aligned with the Vacuum Test Chamber using laser alignment (Fig. A.9a) and pumped down to $\sim 10^{-6}\, torr$. Power supply connection and communication with the board via Ethernet was faciliated by the Vacuum Test Chamber Feed-through (Fig. A.9b).

Additional images of the Vanderbilt Pelletron beamline from multiple vantage points are shown in Fig. A.10. More information on the Vanderbilt Pelletron can be found in [McC15].



(a) *Spare* Delidded Chip (b) Delidded Chip on NS1e Board

Figure A.8: **IBM TrueNorth Chip Delidding.** The initial delidding process was first demonstrated on a non-functional TrueNorth chip (shown in *(a)*) using a foaming nitric acid process followed by an acetone cleaning process without using any water. Protection tape was used to avoid wet chemicals from damaging surrounding board components.

(a) Inside Test Chamber (b) External Feed-through

Figure A.9: **Vanderbilt Pelletron Vacuum Test Chamber.** (a) TrueNorth NS1e Board was aligned inside the test chamber using the green alignment laser shown. (b) External feed-through provided power supply & local Ethernet connections to the NS1e board. Ethernet was used to send data between the NS1e board and the gateway computer.

(a) From the Test Chamber (b) From the Source

Figure A.10: **Vanderbilt Pelletron Accelerator Beamline.** Viewed from the (a) Vacuum Test Chamber and (b) the ion/particle source. More information can be found in [McC15].

A.1.4 Experimental Results: Fragile Corelet

Before running CNN-based experiments on the TrueNorth under irradiation, an initial *fragile* corelet (program) was loaded onto the chip—*'Randomly-connected Spontaneously Spiking Neurons (RCSSN)*. The RCSSN corelet was utilized to detect whether or not errors were detected in the on-chip SRAM after exposing the chip to 4 MeV protons utilizing the Vanderbilt Pelletron. Traditionally, this corelet was designed to detect post-fabrication die defects but is easily repurposed to detect SRAM bit errors. The RCSSN corelet executes in a pipelined fashion at a tick rate of 1 kHz and designed to run for 600,000 input spikes or $\sim$10 *minutes* without any errors. The corelet halts if the output at any point no longer matches the expected 'golden truth' output spike file. Results across 6 runs are shown in Fig. A.11.

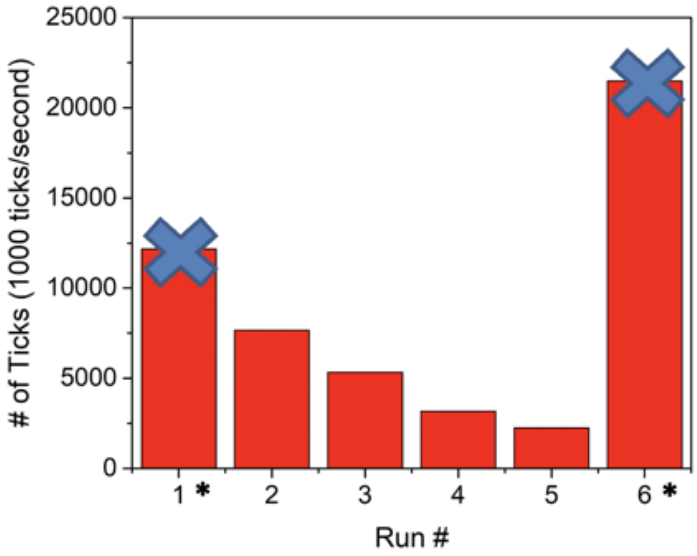


Figure A.11: **Fragile Corelet Performance under irradiation.** TrueNorth continuously exposed to 4 MeV protons with a flux of $2.5 \times 10^{-5}\, cm^{-2} s^{-1}$. Program (*'corelet'*) specifically designed to halt if any errors are detected by running it until program output no longer matches expected output. Runs 1 & 6 are discarded as there was a substantial latency between the start of the computation and the start of the beam exposure due to operator error.

Table A.2: **IBM TrueNorth CNN-based Classification Experimental Results.** Table consists of 8 experiments across 3 different datasets (MNIST, CIFAR-10, & CIFAR-100).

Exp. #	Run #	Dataset	Total exposure (s)	Flux ($cm^{-2}s^{-1}$)	Total Fluence (cm^{-2})	Initial Error Rate	Final Error Rate	Error Rate Increase	Net Additional Classification Errors	Net Classification changes ("mismatches")
3	16	MNIST	100	2.50E+05	2.50E+07	1.18%	1.18%	0.00%	0	39
3	19	MNIST	100	2.50E+05	2.50E+07	1.18%	1.22%	3.39%	4	22
4	1	MNIST	200	1.73E+05	3.46E+07	1.18%	1.16%	-1.69%	-2	48
4	4	MNIST	120	2.43E+05	2.92E+07	1.18%	1.23%	4.24%	5	19
4	7	MNIST	120	2.04E+05	2.45E+07	1.18%	1.15%	-2.54%	-3	34
4	2	CIFAR	150	1.95E+05	2.92E+07	16.69%	16.83%	0.84%	14	626
4	3	CIFAR100	100	1.91E+05	1.91E+07	44.46%	44.97%	1.15%	51	1864
4	6	CIFAR100	120	2.15E+05	2.58E+07	44.46%	45.42%	2.16%	96	2329

A.1.5 Experimental Results: Convolutional Neural Networks

A similar 14-layer CNN structure was trained on each of the 3 datasets of interest: MNIST, CIFAR-10, & CIFAR-100. Initial test accuracies of 98.82%, 83.31%, and 55.54% were achieved after training across the 3 datasets, respectively, using the EEDN training algorithm [Ess16]. Each of the 3 trained network models were then separately loaded onto the chip and irradiated during separate runs. Their classification accuracy was compared before & after radiation with model files being reloaded after each experimental run. Results for all runs are summarized in Table A.2 and were previously reported in[MCB19, BM19].

An additional two experimental runs utilizing an MNIST-trained model irradiated for 10 cycles of 10s is shown in Fig. A.12. After each cycle, the CNN test classification accuracy was evaluating by running the model using the full 10,000 test image dataset. One interesting observation from Table A.2 and Fig. A.12 quickly points out that while a large number of classification changes or *mismatches* might occur due to random changes (e.g. SRAM bit errors) in the on-chip CNN model, only a small subset of these classification changes appeared to actually result in additional classification errors. This is because incorrectly classified test images prior to irradiation were low-confidence answers to begin with and

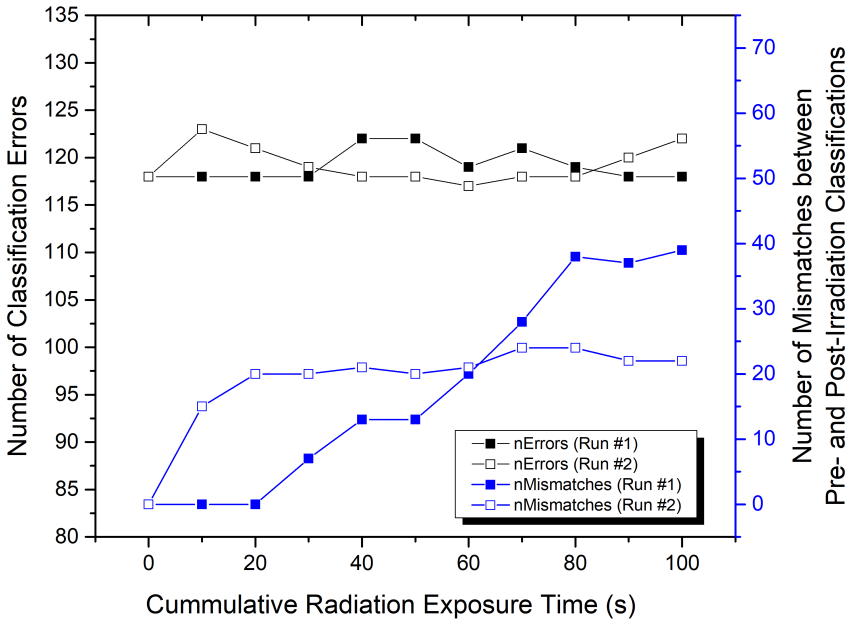


Figure A.12: **Effects of SEUs on MNIST-trained CNN on IBM TrueNorth.** Number of Classification Errors and Changes (*Mismatches*) vs. Cummulative Radiation Exposure time for two separate exposures of a TrueNorth system loaded with an MNIST-trained 14-Layer CNN model. Each run consisted of 100s cummulative exposure (10 cycles of 10s) to 4 MeV protons with a flux of $2.5 \times 10^{-5}\, cm^{-2} s^{-1}$. *Mismatches* defined as changes between pre- and post-irradiation output classification labels.

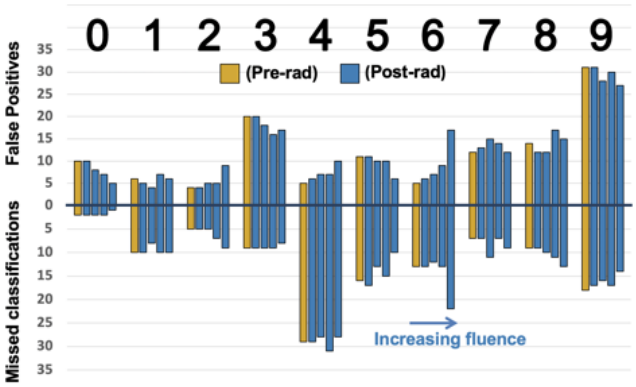


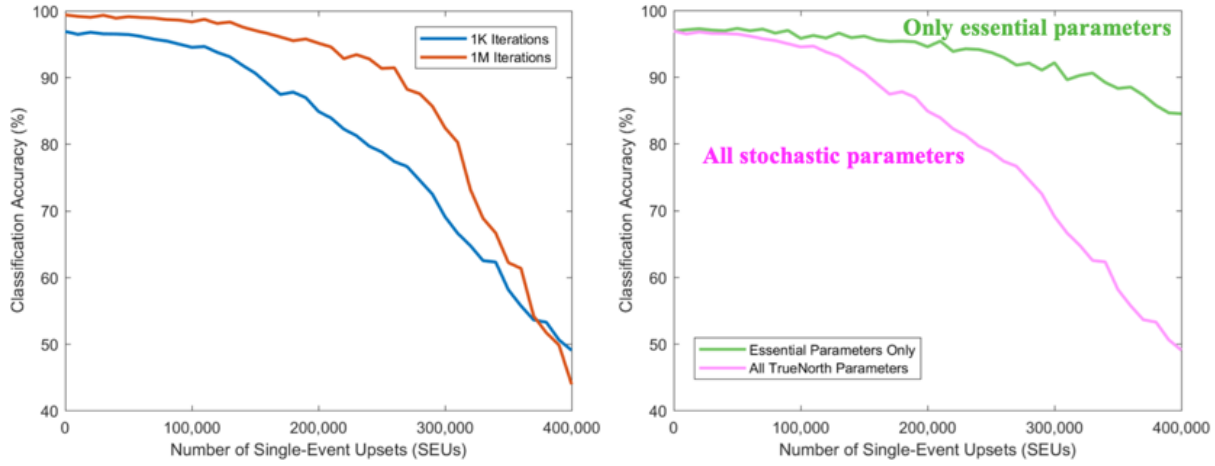
Figure A.13: **MNIST Classification Changes for Varying Fluences [BM19].** MNIST network exposed to 4 MeV protons with a flux of $2.5 \times 10^{-5}\, cm^{-2}s^{-1}$ for 0, 10, 15, 30, & 37 seconds. *False positives* are defined as outputs wrongly classified as specified output label and *missed classifications* are outputs that were not correctly classified as specified output label. For increasing fluence, *0's* were detected correctly more frequently while *6's* less frequently.

were most susceptible to classification changes from one incorrect class to another incorrect class.

Figure A.13 provides more detailed information for MNIST output classification changes (*mismatches*) for 5 different runs with varying exposure time. Some output classes (e.g. *0's*) observed increased classification performance for increasing fluence while others saw output performance degradation (e.g. *6's*).

A.1.6 Simulation Results

The IBM TrueNorth Neurosynaptic Core Simulator (NSCS) was separately utilized to emulate the TrueNorth system. The effects of randomly introduced SEUs or *soft* errors were studied in software using NSCS by taking a trained model file, injecting random bit errors or upsets, and running the file on the NSCS tool. Generally, networks trained for a larger



(a) (b)

Figure A.14: **IBM TrueNorth NSCS Simulator CNN Results.** Neurosynaptic Core Simulator (NSCS) framework was utilized to simulate the effects of randomly introduced bit errors on the network. (a) Simulated effects of bit errors on classification accuracy for two separately trained networks with varying training efforts (10^3 and 10^6 iterations), averaged across 10 runs. (b) Simulated effects of bit errors on EEDN-essential parameters vs. all network parameters (including stochastic parameters), averaged across 10 runs.

number of training iterations appeared to be more resilient to random bit errors as shown in Fig. A.14a. Additionally, some parameters were identified as more sensitive to bit errors depending on its function. Optional parameters such as stochastic parameters are not utilized by EEDN-trained CNNs on the IBM TrueNorth and are disabled by default, but random bit errors could enable some of these parameters and dramatically modify the behavior of the network. As shown in Fig. A.14b, when bit errors were introduced to all parameters including stochastic parameters, the classification accuracy dropped much faster than when bit errors were only introduced to EEDN-essential TrueNorth parameters. This hints that a redesigned digital neuromorphic platform with only CNN-essential parameters is expected to be more resilient to *soft errors* or bit errors in the network.

A.1.7 Further Design Insights

System crashing was observed on the IBM TrueNorth after exposing the system to irradiation. As long as the system was left in standby mode by pausing input spikes to the chip during irradiation, the system exclusively crashed during subsequent Test/Evaluation cycles, shown in Fig. A.15. Further, system crashing can be also induced during irradiation by continuously evaluating the network (e.g. cycling over 10,000 test input dataset for MNIST). This indicates that the system crashing was largely input-dependent.

It was determined that the likely contributor of system crashing was due to changes in neuron parameter addressing. As previously mentioned in Section A.1.1, The neuron (destination core) address supports an off-chip addressing space which supports larger systems of up to 4×4 TrueNorth chips. For a single-chip system such as the NS1e board, if neuron address parameters are corrupted by bit errors, it can cause neuron output spikes to be routed off-chip.This is not expected to occur under normal chip operating conditions, but may cause system crashing as the system was not designed to delete unsendable packets and eventually enters an undefined error state after some number of output spikes are routed to be sent off-chip. 'Problematic neurons' limited the total exposure of all experiments shown in Table A.2 as all experimental results eventually crashed after a certain number of irradiation cycles. All experiments were conducted by pausing or idling the model file during irradiation in order to reduce probability of system crashing.

A.1.8 Conclusions

Neuromorphic systems are inherently redundant. Simulations show that a neuromorphic architecture—such as the IBM TrueNorth—is capable of enduring extensive exposure to low-dose irradiation ($\sim 10^5\, SEUs$) while maintaining negligible reduction in classification accuracy. Furthermore, experimental results validate expectations by showing a maximum increase in classification error rate of 4.24% (1.18%→1.22%) for an MNIST-trained CNN

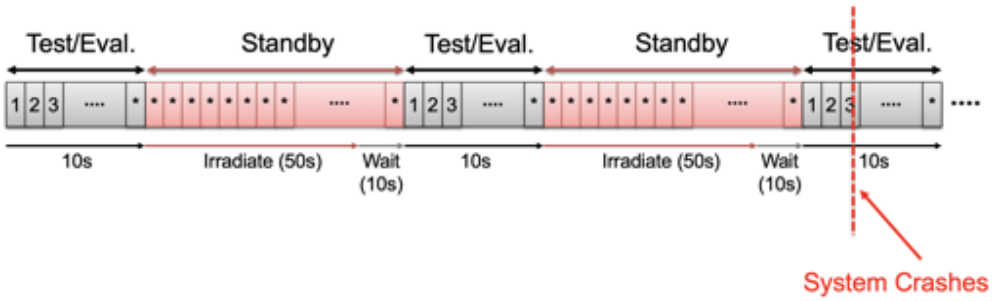


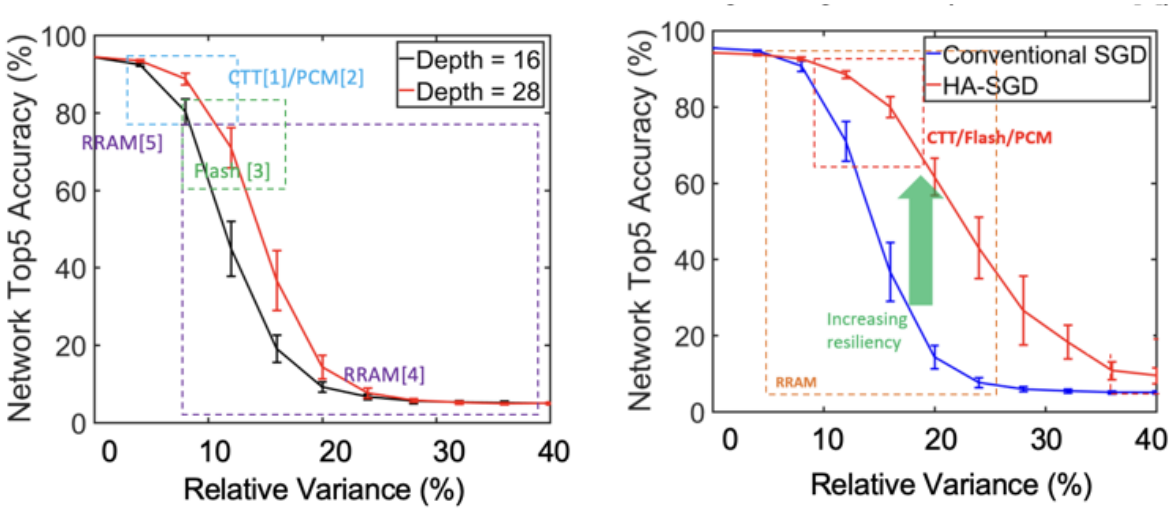
Figure A.15: **IBM TrueNorth System crashing after irradiation.** Example multi-step irradiation of MNIST CNN network. Network is first evaluated (pre-irradiation) to confirm expected initial network results, left idling during for a cycle of irradiation by pausing spike inputs, evaluating, irradiating, evaluating etc. System exclusively crashes during Test/Evaluation phase and not during the Standby/Irradiation phase.

after $120s$ exposure to $4\,MeV$ protons, shown in Table A.2. Neuromorphic architectures are ideal candidates for high-throughput and reliable operation in high-altitude, strategic, & radiation environments. Additionally, the effects of bit errors on the IBM TrueNorth can be further mitigated by system redesign with only CNN or EEDN-essential network parameters, as described by Fig. A.14b.

A.2 Analog-based Computation

Analog-based Neural Networks are susceptible to both circuit-induced errors and device non-idealities. Circuit-induced errors are architecture and input-dependent. They require detailed characterization in order to study how a system may perform across all process, voltage, & temperature ($P/V/T$) corners, device mismatches, inputs (x_i), and weights (w_{ij}). Other forms of characterization such as noise analysis and device-specific studies including relaxation, retention, & endurance may also be of interest.

The Charge-Trap Transistor, NeuroCTT Architecture, and Chip Testing Results were introduced in Chapters 2, 3, & 4. This section will focus on the performance & accuracy of



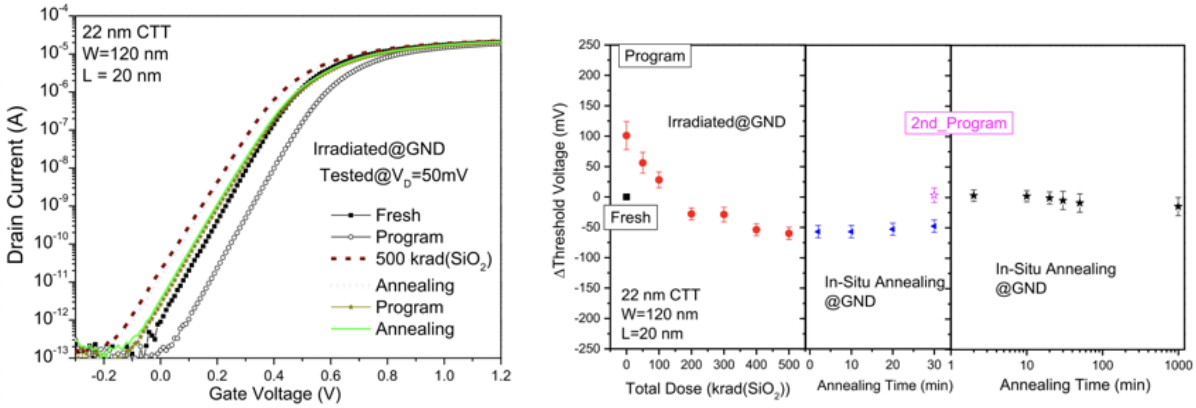
(a) Depth Comparison

(b) Training Algorithm Comparison

Figure A.16: **Effect of Relative Variance on Inference Accuracy.** A Wide-ResNet [ZK17] model was trained using the CIFAR-100 dataset [Kri09]. Relative variance defined as the $\frac{\sigma_{PRG}}{Range}$ and reflects the weight programming accuracy and retention. (a) Trained Network (Top 5%) Accuracy vs. Weight Relative Variance for two different networks of depth 16 & 28, respectively. (b) Trained Network (Top 5%) Accuracy vs. Weight Relative Variance for a Wide-ResNet (Depth=28) network trained using the Traditional Stochastic Gradient Descent (SGD) [Bot98, BCN18] and Hessian-Aware Stochastic Gradient Descent (HA-SGD) [WWZ22] algorithms.

analog nonvolatile memory-based neural networks using the NeuroCTT system as a baseline system for evaluation. Thorough accuracy and performance evaluation of analog in-memory computing (IMC) architectures based on emerging analog non-volatile memory (NVM) technologies was recently reported in [WWZ22, WMC19], a subset of which will be shared here.

Circuit-induced errors can also be characterized using the CIRCUS Platform introduced in Section 4.4 (and Appendix C), considering global process variation and mismatch across all temperature corners. Device non-idealities are characterized by studying the (1) weight programming error (e.g. $w_{trained} \neq w_{deployed}$), referred to as programming variance (σ_{PRG}) in this dissertation, (2) device shallow relaxation, (3) device retention, and (4) subthreshold degradation. Other variables may be of interest depending on the target device used. For example, some RRAM devices experience weight instability or large fluctuations over time especially in the high-resistance (HRS) state, shown in [Li21]. While MOSFET transistors are inherently susceptible to $\frac{1}{f}$ ('flicker') noise, these low-frequency fluctuations are substantially smaller in magnitude comparison to those shown in [Li21] and assumed to be negligible.



(a) I_{DS} vs. V_{GS} Curves

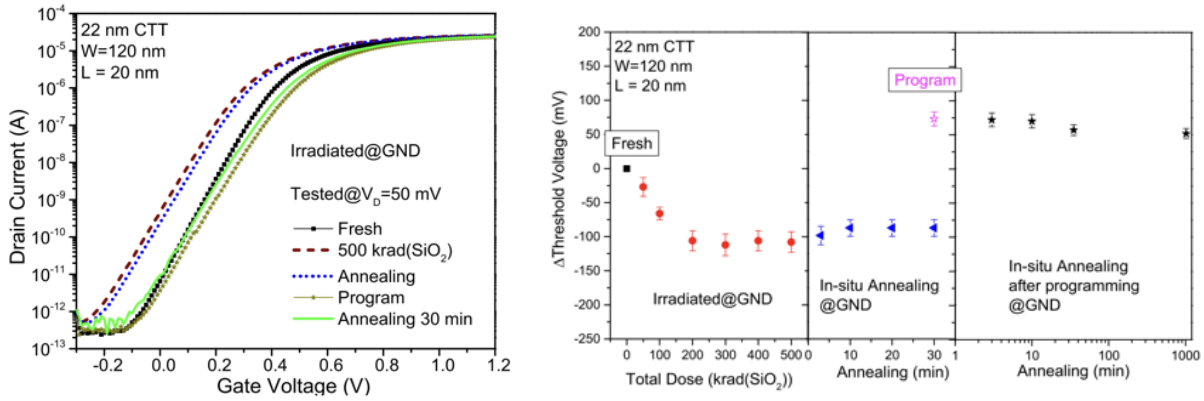
(b) ΔV_{TH} Shift

Figure A.17: **TID Effects on 22nm ($W = 120nm$) FDSOI Devices Programmed *Before* Irradiation**. Averaged across 3 devices. Previously reported in [BZG21].

A.3 Effects of Total Ionizing Dose (TID) on the CTT

Studies were performed to evaluate the effects of Total Ionizing Dose (TID) on 14LP and 22FDX-based CTT devices [BZG21]. While not fully relevant to the device operation in non-strategic operating environments, TID-effects provide valuable information about how these devices would fair in high-dose environments and clues about how to remedy new issues introduced by TID.

It was determined that FDSOI (e.g. 22FDSOI) technologies are highly-sensitive to TID effects due to hole-trapping in the Buried Oxide (BOX) layer with a net effect of reducing the device's threshold voltage ($-\Delta V_{TH}$). Figures A.17 & A.18 suggest that TID effects were due to hole-trapping in the BOX instead of trapping in gate dielectric, as a programming cycle with ΔV_{TH}~$200mV$ was achievable on devices that were programmed after irradiation. ΔV_{TH} is presumed to be fairly systematic across a chip and could be somewhat compensated for by applying back-gate biasing, though *Zheng et. al* [ZCX19] points out that only positive back-gate biases may be used for Forward-Body-biased (FBB) nFET devices to prevent p-n conduction, which would only further decrease the threshold voltage instead of negating the



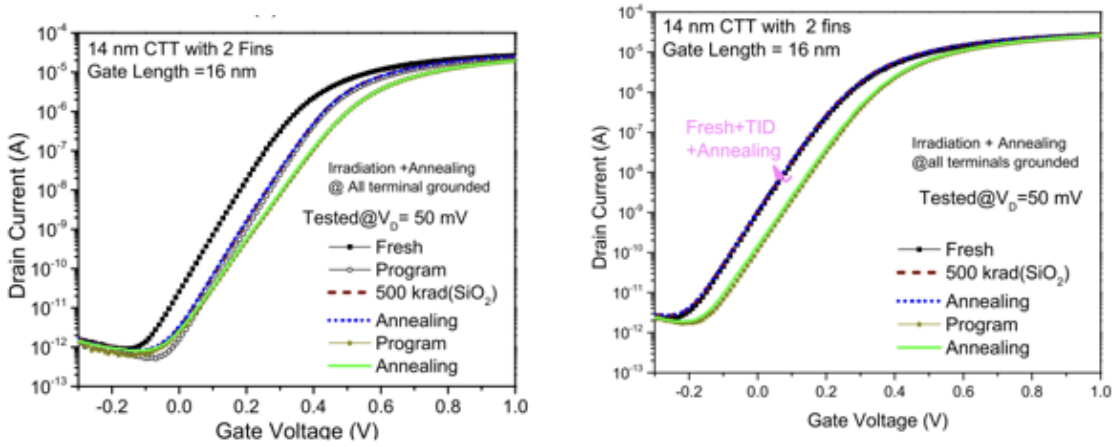
(a) I_{DS} vs. V_{GS} Curves

(b) ΔV_{TH} Shift

Figure A.18: **TID Effects on 22nm ($W = 120nm$) FDSOI Devices Programmed *After* Irradiation**. Averaged across 3 devices. Previously reported in [BZG21].

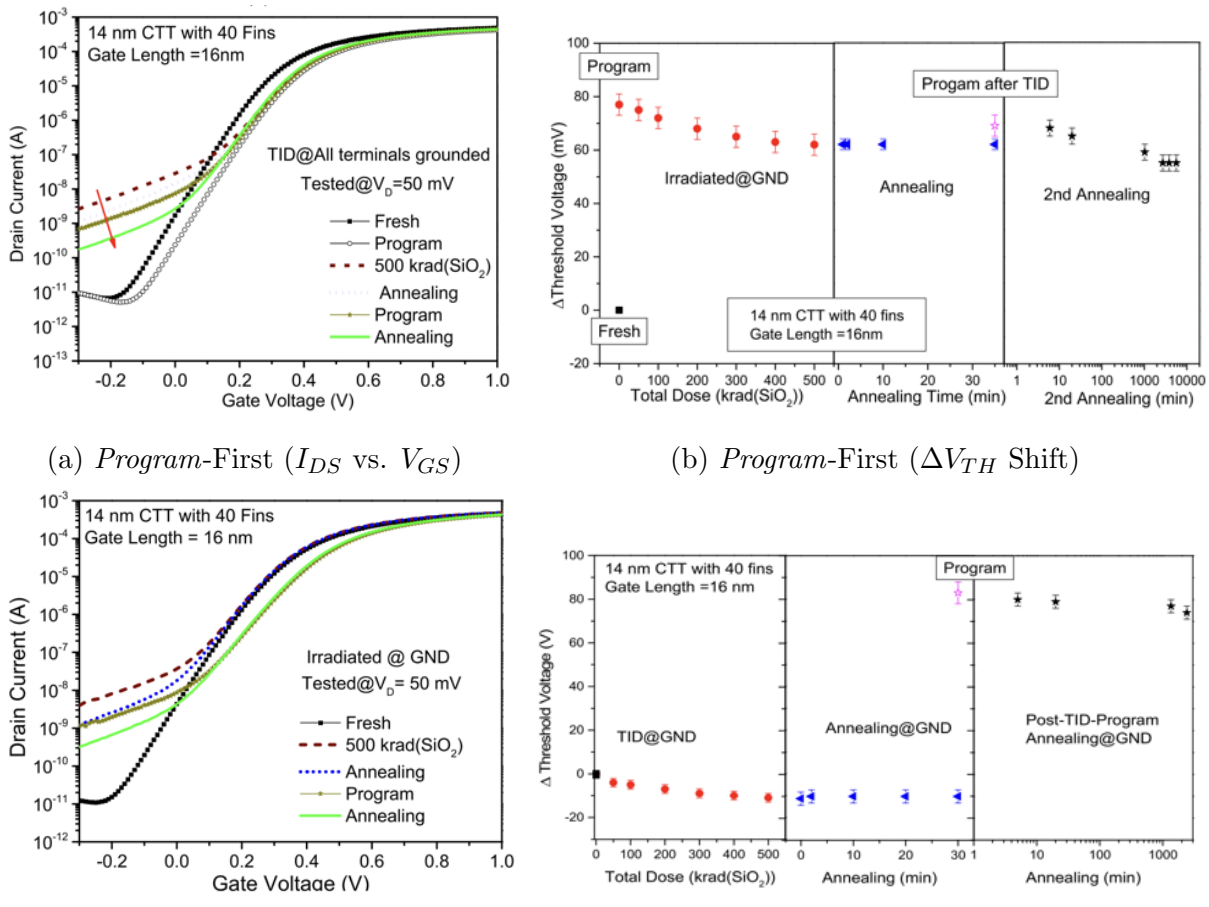
TID-induced threshold shift. TID effects may be alleviated by using a thicker BOX layer to reduce channel coupling. Additionally, PDSOI technologies have been shown to have substantially less sensitivity to TID effects.

Charge-trapping has also been demonstrated in bulk devices. TID studies were performed on GlobalFoundries 14nm low-power (14LP) FinFET devices as a control experiment. Significantly less ΔV_{TH} due to TID irradiation was observed, shown in Fig. A.20. Unfortunately, devices with more fins (e.g. $N_f = 40$) saw substantial off-state current as shown in Figures A.20a & A.20c, compared 2-fin devices shown in Fig. A.19. Both observations agree with current information available in the literature [HMA15, ZCX19].



(a) *Program*-First (I_{DS} vs. V_{GS}) (b) *Irradiate*-First (I_{DS} vs. V_{GS} Shift)

Figure A.19: **TID Effects on 14nm ($W_{eff} = 150nm, N_f = 2$) Devices.** (a) Typical I-V Curve for a device that is programmed first, then irradiated. (b) Typical I-V Curve for a device that is irradiated first, then programmed. Previously reported in [BZG21].



(a) *Program*-First (I_{DS} vs. V_{GS})

(b) *Program*-First (ΔV_{TH} Shift)

(c) *Irradiate*-First (I_{DS} vs. V_{GS})

(d) *Irradiate*-First (ΔV_{TH})

Figure A.20: **TID Effects on 14nm ($W_{eff} = 3\mu m, N_f = 40$) Devices.** (a) & (b) show data averaged across 3 devices that were programmed first, then irradiated. (c) & (d) show data averaged across 3 devices that were irradiated first, then programmed. Previously reported in [BZG21].

APPENDIX B

Radiation Effects on 3D-Stacked Architectures

While a majority of this dissertation focuses on edge-based applications, large-scale deep learning hardware and its overall system reliability is becoming increasingly important. This chapter concludes this dissertation by looking at the effects of errors on general-purpose 3D-stacked architectures, utilizing 3D-Stacked (3DS) DRAM as a tool for measuring the propagation of errors between multiple die layers.

Many larger-scale systems today leverage 3D memories including High-Bandwidth Memory (HBM) and 3D-Stacked (3DS) DRAM in order to store large multi-layer networks. As an example, the Generative Pre-trained Transformer 3 (GPT-3) model for generating human-like text has over 175 billion network parameters trained on $\sim 45TB$ of training data across multiple datasets [BMR20].

B.1 Experimental Design Challenges

Designing a set of experiments for evaluating the effects of radiation-induced soft errors on 3D-stacked architectures is nontrivial. First of all, 3D-stacked components are most commonly available today as packaged memory chips including the Hybrid Memory Cube (HMC) [Mic18c], High-bandwidth memory (HBM) [Mic18b], and 3D-stacked (3DS) DRAM [Mic18a]. These memory chips, especially HMC and HBM, are typically co-embedded with a processor on a silicon interposer and are difficult to procure in a discrete & standalone-testable form factor. Because of this, FPGA compatible platforms for accessing and testing

Figure B.1: **Xilinx Virtex Ultrascale+ HBM (VCU-128) FPGA Board.** Board includes Xilinx VU37P FPGA with $2 \times 32Gb$ 4H HBMs.

HBM memory were evaluated for reduced testing complexity. In the example of the Xilinx Ultrascale+ VU37P HBM FPGA (VCU-128 board), shown in Fig. B.1, the FPGA is co-embedded on a Silicon interposer with $2 \times 32Gb$ 4H HBMs [SA17] [Xil19] requiring the entire heat sink to be removed and the interposer to be partially delidded to expose the HBMs for irradiation.

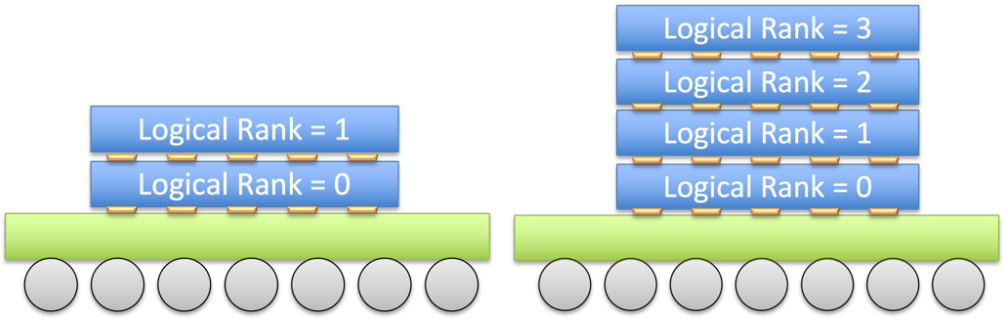
Secondly, removing the heatsink leads to thermal dissipation issues and excludes vacuum-based testing options such as utilizing the Vanderbilt Pelletron [McC15]. It also complicates the experiment as errors may not be exclusively introduced into the HBM memory, but may also affect the FPGA chip responsible for interfacing with and testing the HBM memory.

Xilinx recently demonstrated in [CM19] single-event evaluation of the same Xilinx Ultrascale+ HBM board as considered above, with limited insights on Single Event Effects (SEEs) on HBM memories and no spatial error information. Their SEE studies utilized $>10\,MeV$ neutron and $64\,MeV$ proton sources at the LANSCE Weapons Neutron Research Facility (WNR) and at Crocker National Laboratory (CNL), respectively.

Separately, HBM and HMC memories not only include memory dies but also include a logic die at the bottom of the stack. Decoupling the location of memory errors and the error source is further complicated by the fact that *soft errors* may be introduced to the memory dies and/or the the logic die.

B.2 3D-Stacked (3DS) Test Samples

As alluded to above, 3D-stacked (3DS) DRAM [Mic18a] was selected as it is (1) procurable in a discrete and testable 288-pin DDR4 DIMM form factor, (2) not co-embedded with a processor or FPGA die on a silicon interposer, (3) simple to chemically delid, and (4) testable using a memory test platform (MTP) such as the Innoventions Ramcheck LX (Fig. B.6) or the custom IBM Sputnik MTP (Fig. B.7). Overall, 3DS is also simpler to test and extract information from as it has no logic die as in the case for HMC & HBM memories.



(a) 2-Layer (2H) (b) 4-Layer (4H)

Figure B.2: **3DS Logical Rank Mapping.** 3DS DDR4 DIMMs include 2 physical ranks and 2/4 logical ranks for the 2H and 4H Stacks respectively. The logical ranks correspond to the stack height where the (a) 2H stack has 2 logical ranks and the (b) 4H stack has 4 logical ranks. Logical rank can be used to locate where *soft errors* occur in the memory stack during irradiation.

Radiation-induced *soft errors* are expected to be dominated by upsets within the bit cells & memory array itself, though one should still consider the effects of read/write/refresh circuitry on the overall bit error rate as this circuitry may also be sensitive to radiation.

Given the volatile nature of DRAM, *refresh* at regular (e.g. $64ms$) intervals is required. While decoupling the effect of radiation on peripheral circuitry and bit cells is ideal, the overall objective of this work is to evaluate the effects of radiation-induced errors on the the entire DRAM memory system (bit cells and peripheral circuitry).

Samsung 64GB (M393A8K40B22-CWD) and and 128GB (M393AAK40B42-CWD) 288-pin DDR4 DIMMs were utilized specifically in this study. The 64GB and 128GB DDR4 DIMM modules consist of 36× 2-layer (2H) and 4-layer (4H) memory devices, respectively. Four of the devices are reserved for ECC and ignored during all experiments. Each die in the both stacks consists of 8Gb DRAM. DIMM Memory hierarchy includes 2 physical ranks (side of DIMM), 2/4 logical ranks (64GB/128GB DIMMs), and 16 banks ($4\,banks/group \times 4\,bank\,groups$). In both cases, the logical rank indexes each die within its respective stacked-

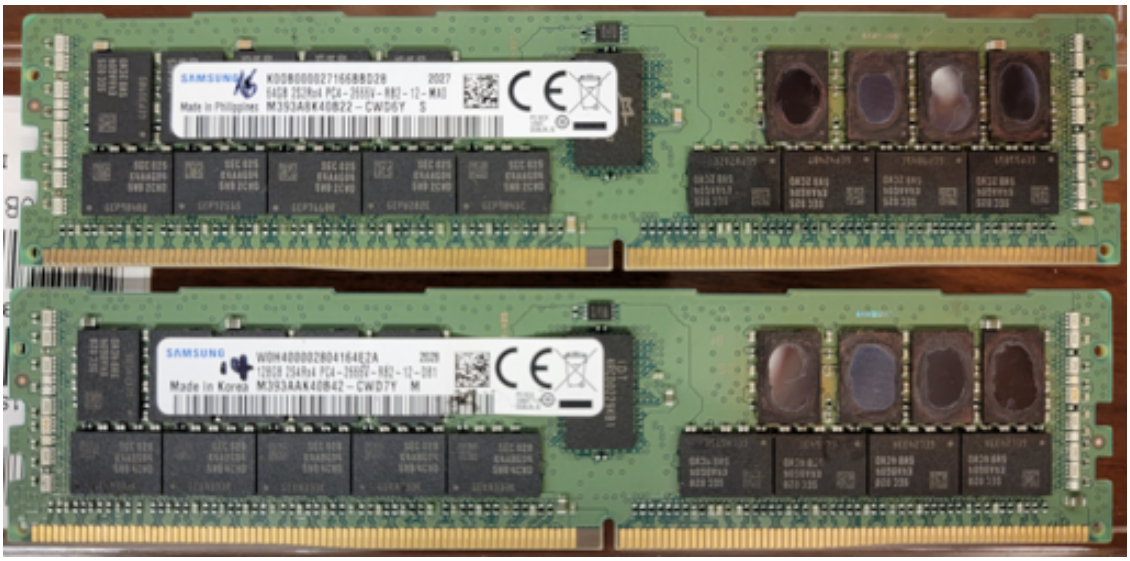


Figure B.3: **Delidded 3D-Stacked (3DS) DRAM Memory.** 288-pin DDR4 modules were utilized for testing. A 64GB, 2-layer stacked DIMM (*top*) and a 128GB, 4-layer stacked DIMM (*bottom*) were used for testing. Devices D21-D24 were delidded—device #'s [Sam17a] & [Sam17b]. Devices D25-D28, directly below the delidded devices, were also utilized for testing at the NASA Space Radiation Laboratory (NSRL).

DRAM device, shown in Fig. B.2.

64GB and 128GB DIMMs were prepared for testing using facilities at both Vanderbilt University and NASA Space Radiation Lab (NSRL). A Titanium-Sapphire Chirped Pulse Amplifier (CPA) Laser was utilized at Vanderbilt, which required sample delidding to expose the bare silicon prior to irradiation. Delidding was performed across 4 DRAM devices on both the 64GB and 128GB DIMMs, as shown in Fig. B.3. All delidded devices were confirmed to be fully functional after delidding using the IBM Sputnik MTP discussed in the next section.

Additionally, cross sections of both the 2H and 4H DRAM devices were taken by molding the devices in a resin and polishing. Cross sections are shown in Fig. B.4.

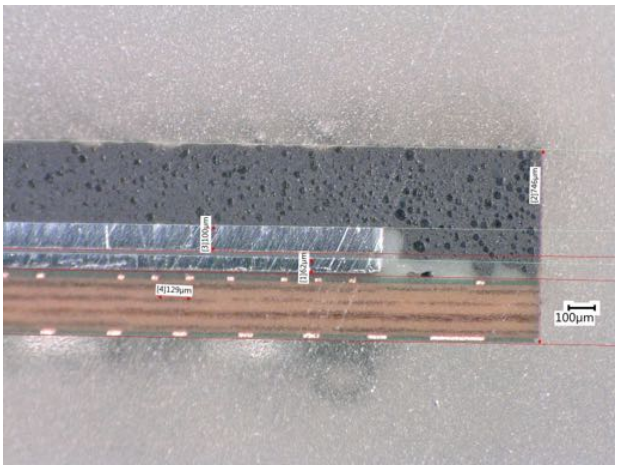
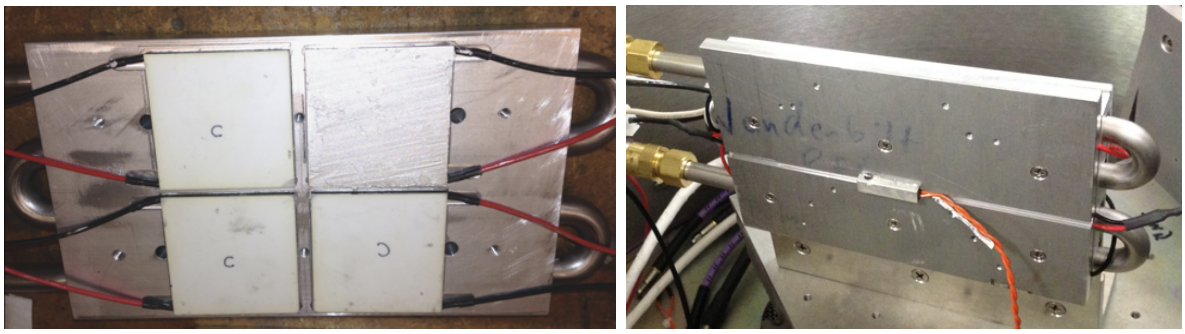


Figure B.4: **Cross Section of 2-Layer (2H) Samsung 3DS Memory Devices.** Stacked memory devices consist of similar 8Gb DRAM dies. Therefore, 2H and 4H stacked-DRAM devices consist of 16Gb and 13Gb, respectively.

B.3 Memory Test Platform (MTP)

After obtaining Samsung 2H (64GB) and 4H (128GB) 288-pin DDR4 DIMM samples, several MTP platforms were evaluated for testing. Multiple factors were taken into consideration including (1) programmability, (2) ability to shield the tester during irradiation, (3) thermal-dissipation issues, (4) accessing physical addresses, (5) commercial availability, and (6) software support.

Most importantly, the testing platform must be compatible with the experimental setup and be able to identify spatial information about where errors are introduced within the memory. Depending on the source of irradiation, there may be size, power, and thermal limitations. If testing is conducted in a vacuum, as is the case for SEU studies using the Vanderbilt Pelletron, thermal considerations are a large concern as heat sinks and fans do not provide sufficient cooling in a vacuum and must be replaced with a thermoelectric cooler, or *cold finger*. It's important to note that while its possible to perform some SEU studies at room pressure ($1\,atm$), this is typically only available at higher-energy facilities such as



(a) (b)

Figure B.5: **Vanderbilt University Thermoelectric Cooling System.** (a) 4 Thermoelectric (TE) cooling modules mounted in between heat exchanger and $0.25\,in$ thick aluminum plate. (b) Thermistor (*center*) was half-embedded into outer side of plate for measuring temperature input for external temperature controller. [McC17]

NASA Space Radiation Laboratory (NSRL).

Previously, it was shown in Section A.1 where an IBM TrueNorth Neurosynaptic System NS1e Board was irradiated in the Vanderbilt Pelletron under vacuum conditions; however, this setup did not require any special heat dissipation techniques as the chip is a low-power design ($\sim 70\,mW$) and was powered down in between runs to prevent any possible overheating or damage. An FPGA system (e.g. Virtex Ultrascale+ HBM VCU-128G, Fig. B.1) or a DDR4-based memory test platform is typically much higher-powered and requires thermal considerations.

Collaborators at Vanderbilt University developed a thermoelectric cooling system, detailed in [McC17] and shown in Fig. B.5, which has been demonstrated to cool a 30W test article in a vacuum chamber using electrical and liquid chamber feed-throughs (see Fig. A.10a). Utilizing a *cold finger* instead of a heat sink and fan allows higher power systems to be tested in vacuum conditions; however, it is difficult to both irradiate and cool the same part (e.g. *interposer*), which is required in the case of testing the Virtex Ultrascale+ HBM VCU-128G board at vacuum.

For the purpose of irradiating 3DS memory, three different DDR4-based memory test platforms were evaluated:

1. Linux Computer (Example test programs: memtest86, malloc, etc.)
2. Ramcheck LX Memory Test Platform (Fig. B.6)
3. IBM Sputnik Memory Test Platform (Fig. B.7)

First of all, Using a Linux-based computer comes with obvious limitations. Computers utilize virtual addressing spaces, making it difficult to evaluate error information for physical addresses. Programs such as memtest86 [Pas21] state that physical addressing decoding scheme information is proprietary for Intel-based systems. Other Linux commands such as memory allocation (*malloc*) were also evaluated but suffer from the same address decoding limitations as above. Additionally, its important for the reliability of the memory test platform to be decoupled from the DRAM memory under test.

Commercially available memory test platforms are typically quite limited in offering, large in form factor, and expensive. Additionally, project specifications required the memory tester to not only support DDR4 memory but also 3DS-DDR4 DIMM memory which consists of both physical and logical ranks. Due to quoted compatibility for 64GB+ DDR4-based DIMMs, the Ramcheck LX MTP (Fig. B.6) was procured and tested with the 64GB & 128GB DIMM modules. Unfortunately, it was found that the tester (1) was not compatible as stated, (2) required custom software support, and (3) was extremely limited by available testing functionality and not designed for detecting exactly where bit errors occurred. Additionally, the MTP manufacturer, Innoventions, ceased operations and was unable to offer any software support in order to run the DDR4 parts.

Procuring a system for testing 3DS-based DRAM memory proved rather challenging due to compatibility and experimental limitations mentioned above. The project was assisted by the support of IBM through the OpenPower Foundation. Collaborators at IBM provided

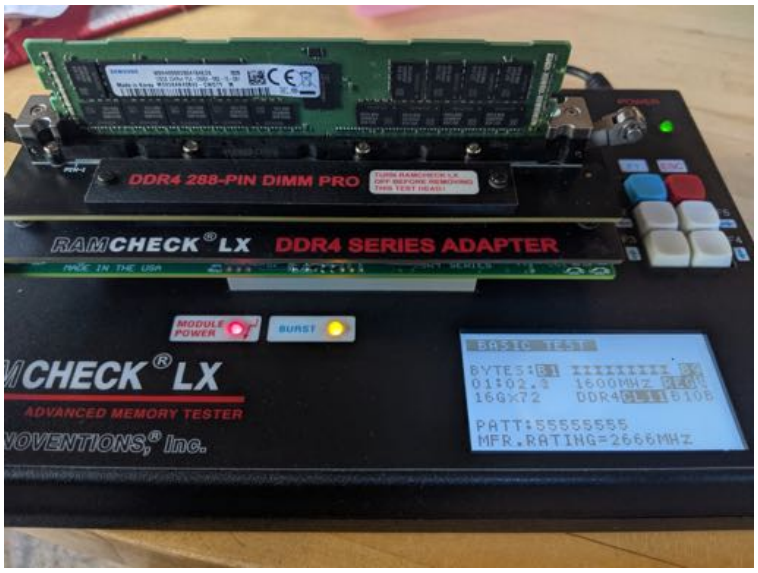
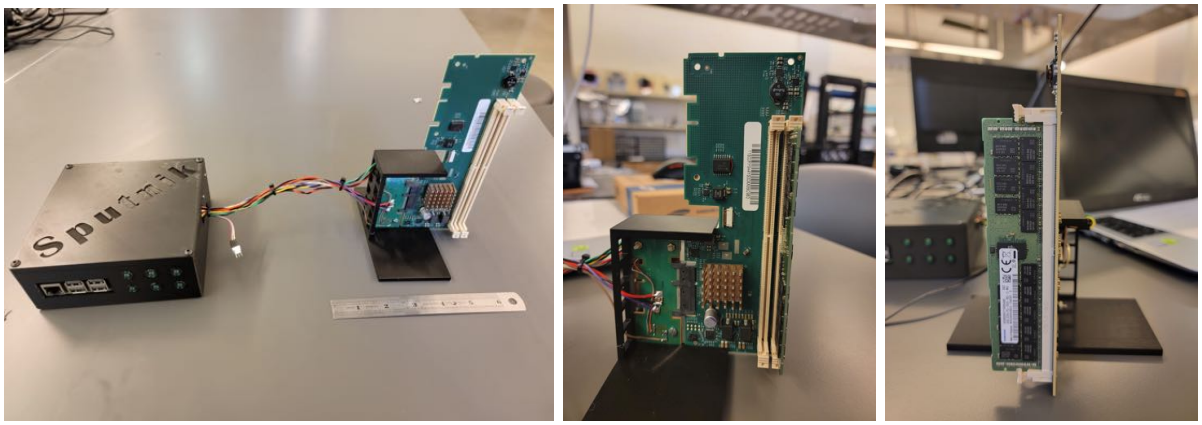


Figure B.6: **Innoventions Ramcheck LX Memory Test Platform.**



(a) (b) (c)

Figure B.7: **IBM Sputnik Memory Test Platform.** (a) Sputnik Memory Test Platform with Raspberry Pi-based control box (*left*) and Minnow card (*right*). Minnow card (b) front and (c) side views. Irradiation source placed such that ions incident on front-side of DDR4 DIMM in (c). Sputnik box and Minnow card communicate via I2C bus and user communicates with Sputnik box via gateway computer utilizing Ethernet protocol.

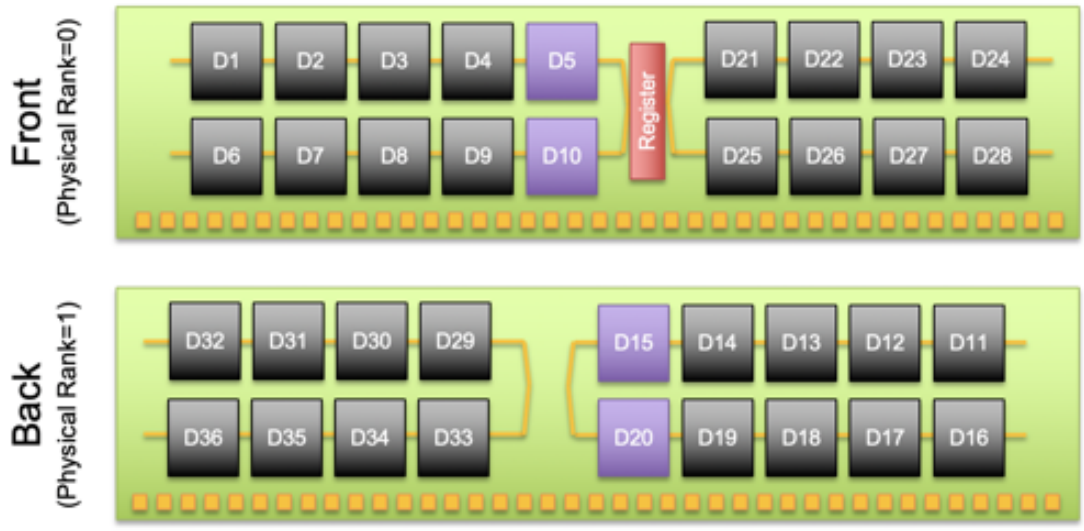


Figure B.8: **3DS DDR4 DIMM Device Layout.** Devices D5, D10, D15, & D20 (purple) are reserved for ECC and disabled during all irradiation experiments.

access to an internal, custom-design memory test platform, nicknamed *'Sputnik'*. The IBM Sputnik MTP (Fig. B.7) consisting a Raspberry Pi-based *'Sputnik'* control box and *Minnow card*. The two components utilize a low-frequency I2C protocol for communication. Gateway computer interfaces with *'Sputnik'* via local ethernet connection. The minnow card consists of a memory controller IC which interfaces with the 3DS DDR4 DIMM module. IBM OpenPOWER Foundation Cronus environment and MCBIST software tools were used to write known patterns into the memory array and detect errors after irradiation.

Example error report information for detected bit errors for a single memory address is found in Table B.2. Error report provides spatial information including physical rank (DIMM side), logical rank (die in stack, Fig. B.2), bank group (BG), bank array (BA), row (2^{17}), column (2^{10}), beat/burst pair, and *nibble* (or device). Nibbles correspond to specific devices on the DIMM depending on which physical rank is selected, detailed in Table B.1 & Fig. B.8.

Memory requests are 64B sent over 4 clock cycles at double-data rate (DDR). This con-

Nibble	Device (Phys. Rank=0)	Device (Phys. Rank=1)
nibble<0>	D6	D16
nibble<1>	D1	D11
nibble<2>	D7	D17
nibble<3>	D2	D12
nibble<4>	D8	D18
nibble<5>	D3	D13
nibble<6>	D9	D19
nibble<7>	D4	D14
nibble<8>	D25	D33
nibble<9>	D21	D29
nibble<10>	D26	D34
nibble<11>	D22	D30
nibble<12>	D27	D35
nibble<13>	D23	D31
nibble<14>	D28	D36
nibble<15>	D24	D32

Table B.1: **Nibble to Device Mapping per physical rank (side of DIMM)**. Each memory request is 64B sent over 4 clock cycles at double-data rate ($64b \times 4\,cycles \times 2\,bursts/cycle$). Each 64b segment is called a burst and clock cycle consists of 1 burst pair (or 2 bursts). Samsung 64GB (M393A8K40B22-CWD) and 128GB (M393AAK40B42-CWD) DIMMs consists of sixteen, 4-bit width ($\times 4$) dram devices per physical rank, excluding ECC devices. Each 64bit burst consists of $4\,bits/device \times 16$ devices. Each 4-bit block of data for each device is also called a *nibble*.

```
1     2022-02-17...: error trap:    0
2     2022-02-17...: dimm:          0
3     2022-02-17...: physicalrank: 0
4     2022-02-17...: logicalrank:   1
5     2022-02-17...: BG:            2
6     2022-02-17...: BA:            0
7     2022-02-17...: row:           0x05406
8     2022-02-17...: col:           0x00
9     2022-02-17...: beat pair:     0/1
10    2022-02-17...: BYTE              :   0011223344556677
11    2022-02-17...: NIBBLE            :   0101010101010101
12    2022-02-17...: ERRORS            :   ............M.X
13    2022-02-17...: burst 0 actual   : 0x1111111111111110
14    2022-02-17...: burst 0 expected: 0x1111111111111111
15    2022-02-17...: burst 1 actual   : 0x2222222222222220
16    2022-02-17...: burst 1 expected: 0x2222222222222222
```

Table B.2: **IBM Sputnik Error Trap Example.** Minnow card supports up to 2 DIMMs, 2 physical ranks (side of DIMM), 4 logical ranks (3DS stack height), 4 bank groups (BG), 4 bank arrays (BA), 17 row bits, and 10 column bits. Bursts (e.g. 0 & 1) are arranged LSB-first, nibble<0:15>. In this example, errors were detected on nibble<15> (physicalrank=0, device=D24) during burst 0 & 1, highlighted in red on lines 13 & 15. Nibble<13> (device D23) is ignored as device D23 was non-functional on utilized DIMM, denoted by an 'M' (masked) on line 12. Nibble-to-device mapping can be found in Table B.1 & Fig. B.8.

```
putscom mc regburst0 1111111111111111 -pall #burst 0
putscom mc regburst1 2222222222222222 -pall #burst 1
putscom mc regburst2 3333333333333333 -pall #burst 2
putscom mc regburst3 4444444444444444 -pall #burst 3
putscom mc regburst4 5555555555555555 -pall #burst 4
putscom mc regburst5 6666666666666666 -pall #burst 5
putscom mc regburst6 7777777777777777 -pall #burst 6
putscom mc regburst7 8888888888888888 -pall #burst 7
```

Table B.3: **Setting Expected IBM Sputnik Memory Pattern Example.** Memory controller (*mc*) chip is utilized for interfacing with DIMM. Written or expected memory pattern is specified by setting eight 64b registers (regburst0-regburst7) on the memory controller. Each 64b register is LSB first, nibble<0:15>. Example utilizes beat pattern resulting in `32'h12345678` to stored on each DRAM device for every valid memory address.

sists of 8 bursts of 64b of data, where two bursts (or a burst-pair) are sent each cycle. DRAM devices on the Samsung 64GB (M393A8K40B22-CWD) and 128GB (M393AAK40B42-CWD) DIMMs are 4-bit width (×4) devices. Each 4-bit packet of data (per device per burst) is called a *nibble*. Error information is provided for each memory address and on each beat/burst pair (e.g. burst pairs 0/1, 2/3, 4/5, 6/7). Arbitrary memory patterns can be specified such as the beat pattern (see Table B.3) as well as checkerboard, all zeros, and all ones patterns.

Thermal and other considerations still exist for the IBM Sputnik MTP as the memory controller IC on the *Minnow card* may consume $\sim 30W$ at run-time, making it difficult to use facilities such as the Vanderbilt Pelletron at vacuum without the use of a 'cold finger' as previously mentioned in Fig. B.5. The Vanderbilt Thermoelectric Cooler (designed in [McC17]) requires reasonable modifications for use with the Sputnik system to cool the memory controller IC (covered by heat sink in Fig. B.7b).

To reduce immediate testing complexity, testing was performed at room pressure ($\sim 1\,atm$) using the Vanderbilt University Pulsed-Laser and high-energy testing at NASA Space Radiation Laboratory (NSRL) facilities, detailed in subsequent sections B.4 and B.5.

B.4 Vanderbilt Pulsed-Laser Testing

Vanderbilt University offers Pulsed-Laser testing, which supports wavelengths varying $300\,nm$ to $3\,\mu m$ and $\sim 235\,fs$ pulses with $1\,kHz$ repetition rate [Van21, MBL04].

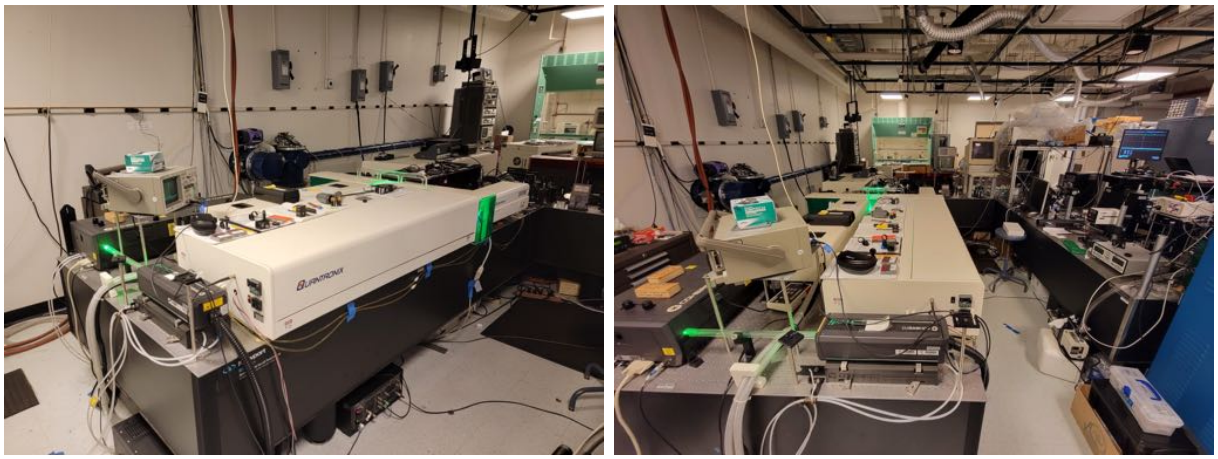
B.4.1 Titanium-Sapphire Chirped Pulse Amplifier (CPA) Laser

The Vanderbilt Pulsed-laser setup consists of a titanium-sapphire chirped pulse amplifier laser. A diode-pumped CW laser is coupled into a passively mode-locked Ti:sapphire laser. The mode-locked amplifier converts the CW laser input into a train of ultrashort pulses with a repetition rate of $\sim 80\,MHz$, which act as seed pulses for a second titanium-sapphire crystal pumped with a frequency doubled Nd:YAG laser at 532nm. The pulses are stretched in duration so they can be safely amplified and then compressed again. After making multiple passes through the gain medium, they are converted to $\sim 235\,fs$ pulses at 1 kHz repetition rate [Van21].

The system utilizes a Topas optical parametric generator (OPG) to convert broad spectrum Ti:sapphire pulses to a wide range of wavelengths (e.g. $300\,nm - 3\mu m$) using non-linear crystals. The laser beam diameter has been measured as $\sim 3.2\mu m$ for $\lambda = 1260nm$. The entire laser system is shown in Fig. B.9 with a mechanical stage for testing shown in Fig. B.10.

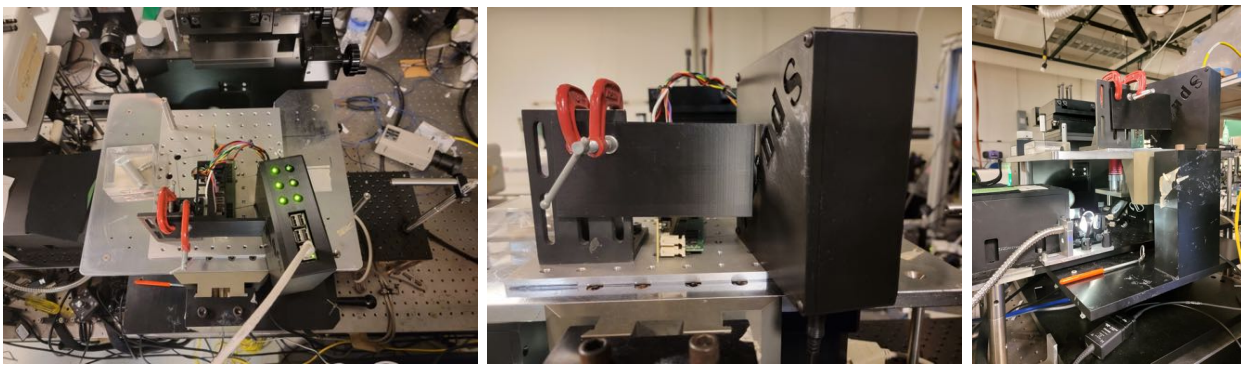
B.4.2 Laser-based Testing Results

Initial test results demonstrated that the Vanderbilt CPA-laser was able to induce Single-Event Effects in the 3DS memory using through-silicon two-photon absorption [MBL04].



(a) (b)

Figure B.9: **Vanderbilt University Ti-Sapphire-based Pulsed Laser.**



(a) (b) (c)

Figure B.10: **Vanderbilt University Pulsed Laser 3DS Test Setup.** (a) Sputnik Test setup (*bird's eye view*). (b) Sputnik test setup (*side profile*). DIMM is parallel to lens opening. (c) Mirrors mounted below objective used to focus beam on device under test (DUT).

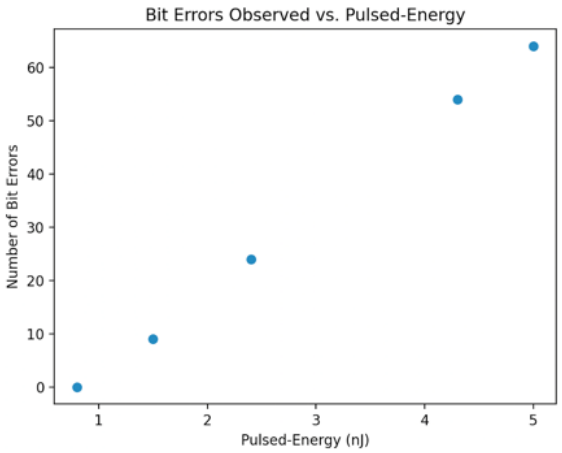
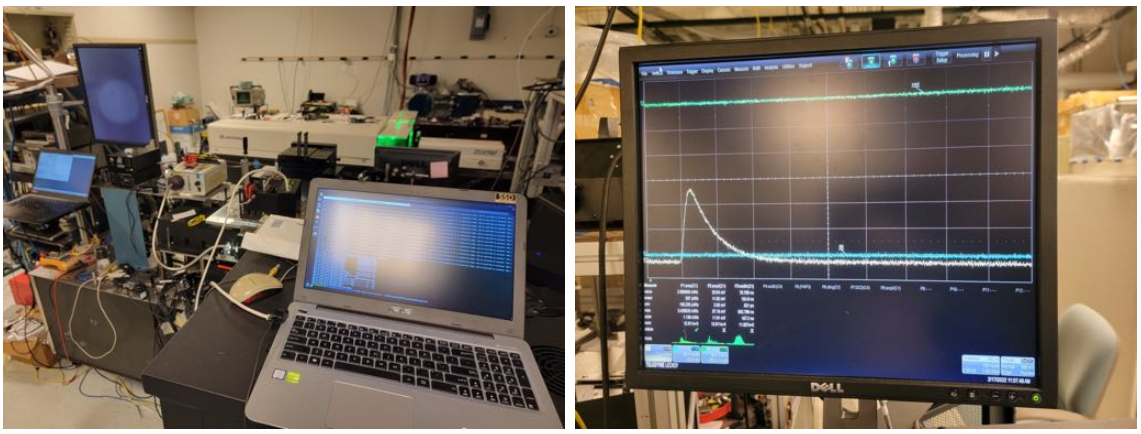


Figure B.11: **3DS Memory Bit Errors vs. Laser Intensity.** The laser was positioned over one of the memory arrays on a delidded device. The laser irradiated the exposed device at 5 different laser intensities, measured in terms of Pulse-Energy. In between runs, the DIMM memory was re-initialized with correct data.

The laser was aligned above an exposed memory device precisely above a memory array by visual inspection. The laser irradiated the exposed devices at 5 different Pulsed-Energy (nJ) intensities. Number of bit errors was measured after irradiation at each intensity level and plotted in Fig. B.11. The memory was reinitialized after each run.

An automated setup was developed to control the laser, mechanical stage, and memory test platform. The laser beam can be moved in $\sim 100\,nm$ steps—shown in Fig. B.12. The setup (1) writes a known pattern into the memory, (2) moves the laser beam and exposes the memory at a fixed X/Y location, and (3) evaluates the DUT to determine where errors occurred within the memory—and repeated for all specified X/Y locations using a simple bash script, shown in Fig. B.13.

The automated setup was used to expose 81 data points across a $4 \times 4mm$ area of the exposed back-side of the 3DS device, where the stage was moved in $500\mu m$ X/Y increments for each point. Results are shown in Figures B.14 & B.15. Testing at Vanderbilt University using the Titanium-Sapphire CPA Laser verified that memory errors can be induced in the



(a) (b)

Figure B.12: **Vanderbilt University Pulsed Laser 3DS Automated Testing.** (a) Setup including laser control computer (*left*) and memory test platform gateway computer (*right*). (b) A fraction of the beam is reflected using a low-incident angle waveplate onto a set of mirrors and into a photodiode, enabling the measurement of every laser pulse generated by the system.

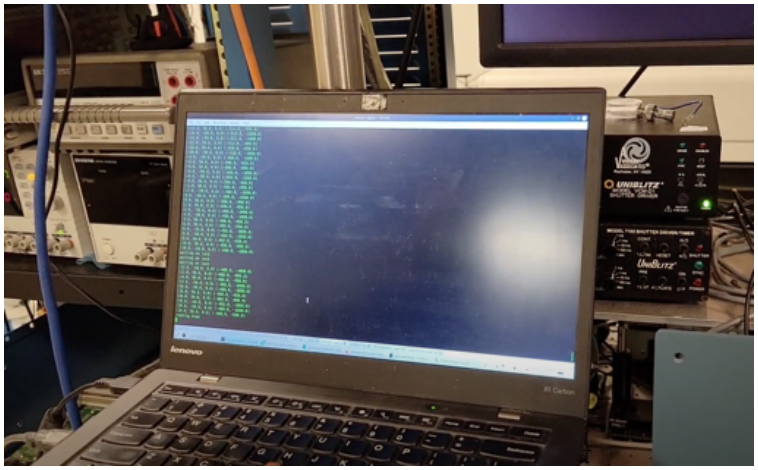
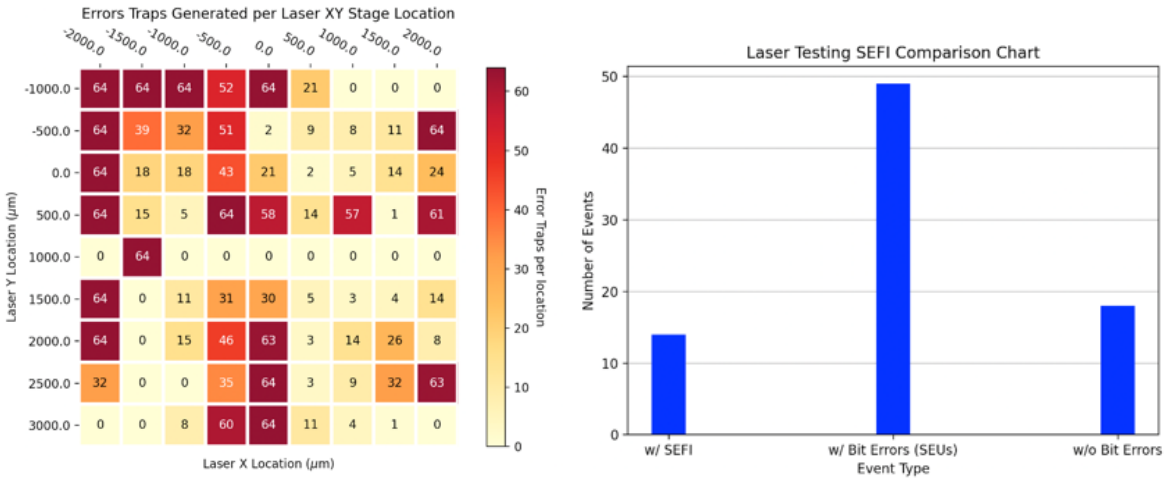


Figure B.13: **Automated Laser Testing Program.** Laser stage is mechanically controlled down to $\sim$100 nm resolution. Simple bash script automatically controls mechanical stage and Sputnik memory test platform and performs automated tests at specified X/Y locations.



(a) Heatmap (b) Histogram

Figure B.14: **Automated 3DS Laser Testing Results.** The Laser stage was automatically moved in 500μm X/Y increments, covering a $4mm \times 4mm$ area (81 total events). For each location, the DIMM was reloaded with correct data, irradiated with a laser, and then evaluated to see if errors occurred. (a) Heatmap of the number of error traps detected per X/Y location. After 64 error traps, the DIMM read operation is halted. (b) Histogram showing whether each event induced Single-Event Upsets (SEUs), Single-Event Functional Interrupts (SEFIs), or no detected memory/bit errors.

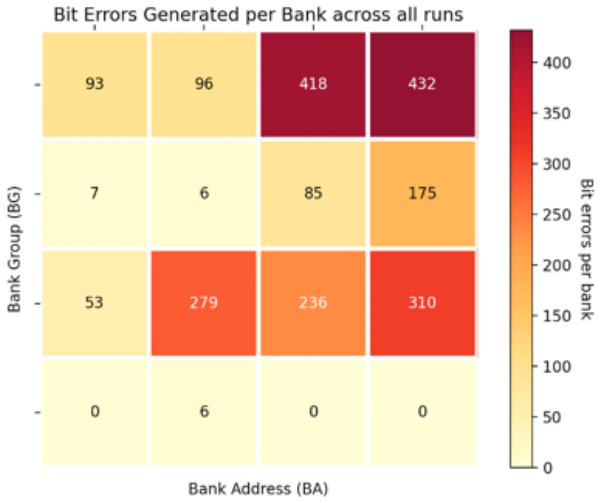
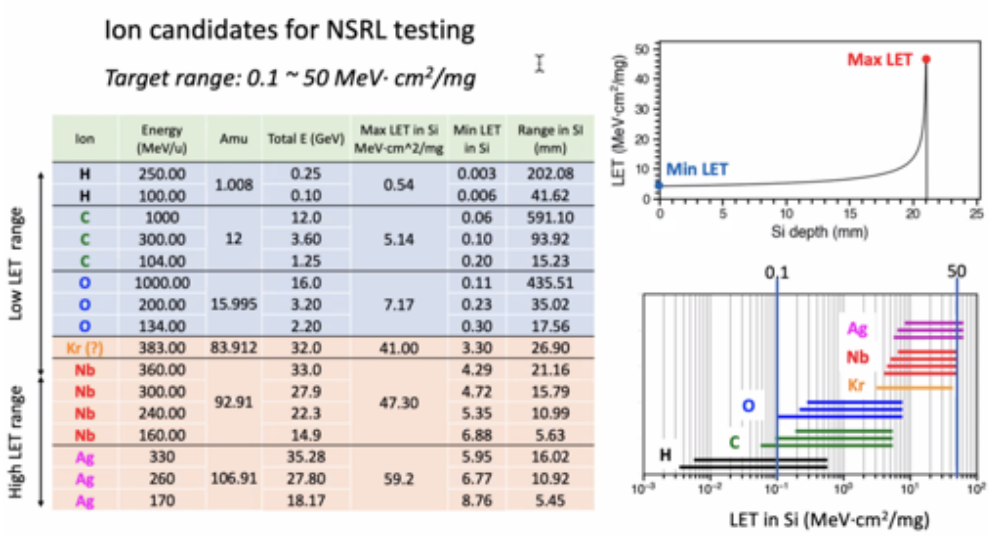


Figure B.15: **Automated 3DS Laser Testing Results: Bit Errors by Bank.** Laser was pulsed across 81 data points spanning a $4 \times 4mm$ area of the exposed device. Number of bit errors detected per device bank is shown.

top-die of the exposed 3DS memory device and accurately detected using the IBM Sputnik Memory Test Platform. Additionally, testing results shown in Fig. B.14 allude to the overall radiation sensitivity of the tested 3DS Memory devices that were observed during radiation studies at the NASA Space Radiation Laboratory (NSRL), described in the following section.



Ion	Energy (MeV/u)	Amu	Total E (GeV)	Max LET in Si MeV·cm^2/mg	Min LET in Si	Range in Si (mm)
H	250.00	1.008	0.25	0.54	0.003	202.08
H	100.00		0.10		0.006	41.62
C	1000	12	12.0	5.14	0.06	591.10
C	300.00		3.60		0.10	93.92
C	104.00		1.25		0.20	15.23
O	1000.00	15.995	16.0	7.17	0.11	435.51
O	200.00		3.20		0.23	35.02
O	134.00		2.20		0.30	17.56
Kr (?)	383.00	83.912	32.0	41.00	3.30	26.90
Nb	360.00	92.91	33.0	47.30	4.29	21.16
Nb	300.00		27.9		4.72	15.79
Nb	240.00		22.3		5.35	10.99
Nb	160.00		14.9		6.88	5.63
Ag	330	106.91	35.28	59.2	5.95	16.02
Ag	260		27.80		6.77	10.92
Ag	170		18.17		8.76	5.45

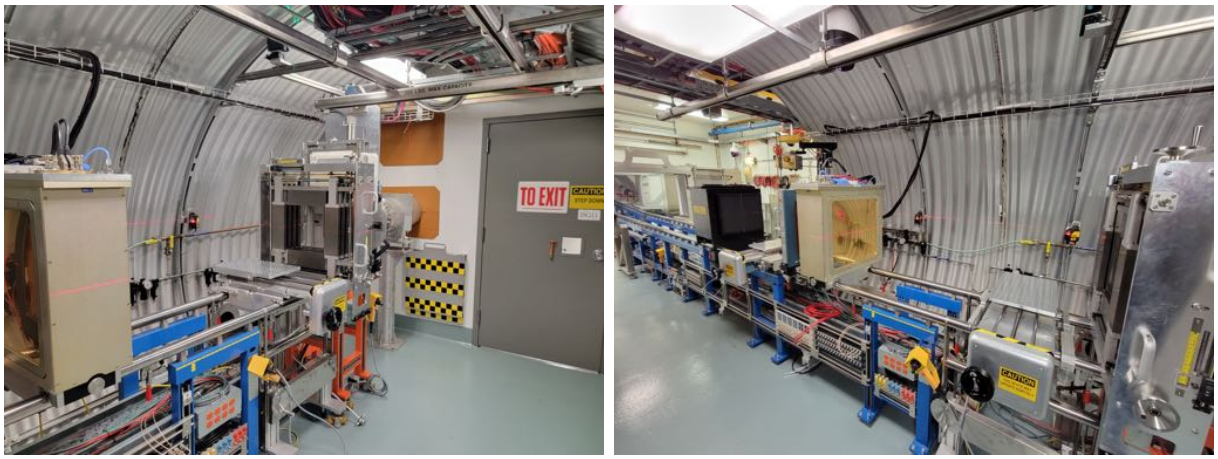
Figure B.16: **NSRL Testing Candidate Ions.** Energy, LET in Si, and Range in Si provided for each ion species.

B.5 NASA Space Radiation Lab (NSRL)

Additional testing was performed at the NASA Space Radiation Lab (NSRL) facility located at Brookhaven National Lab (BNL) in Long Island, NY. The NSRL facility itself was established to assess the risks of space radiation to human space travelers and equipment. It offers a variety of high-energy, non-ionizing radiation recipes for medical, biological, physical, and electronics testing.

B.5.1 Test Facility Capabilities

The ion beam supports a large range of energies and ion species. Full ion and energy capabilities as well LET curves, LET energy in Si ($MeV/(mg/cm^2)$), and range in Si can be found in [NAS22a, NAS22b]. A subset of candidate ions for 3DS testing are shown in Fig. B.16. The NSRL Beamline is shown in Fig. B.17.



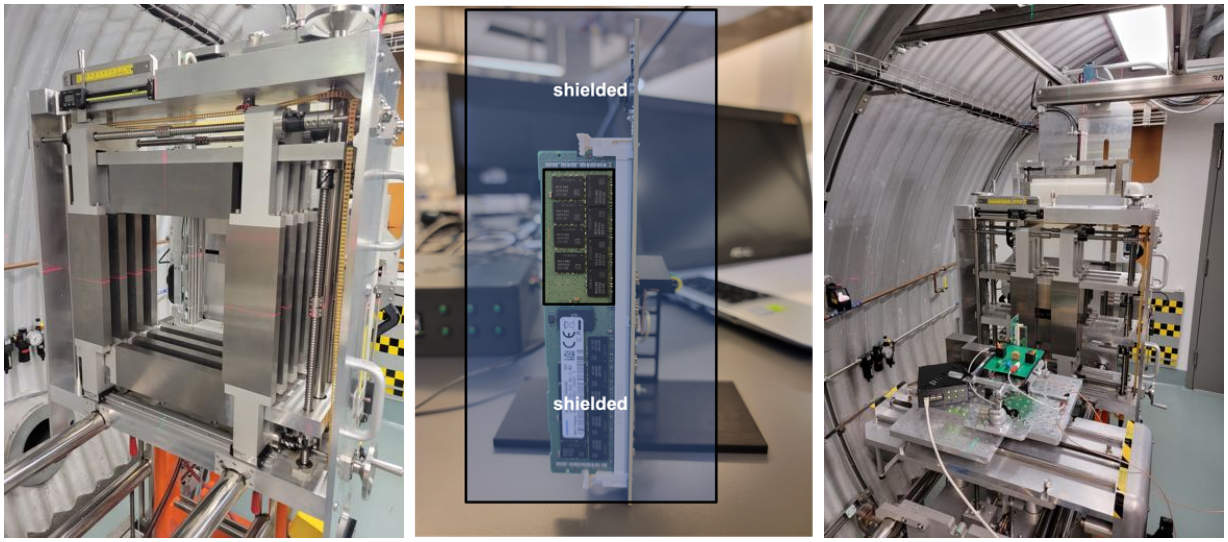
(a) Front (b) Back

Figure B.17: **NSRL Beamline.**

B.5.2 NSRL Testing Results

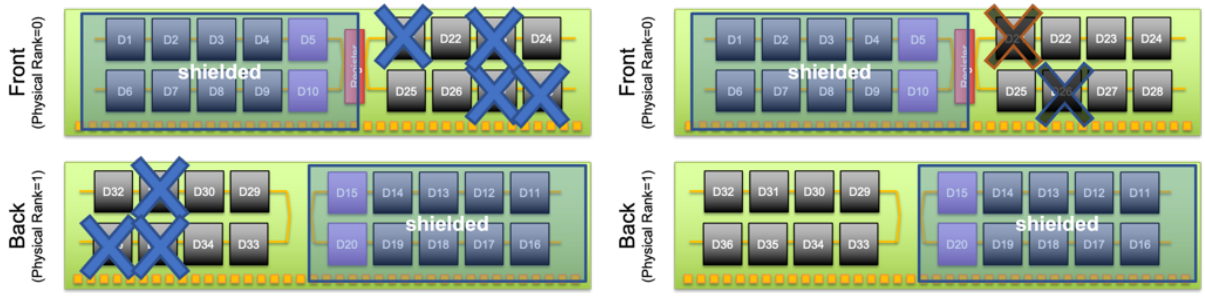
NSRL Tungsten Collimator with 3DS Test Setup is shown in Fig. B.18. The ion beam size is $\sim 10 \times 10\, cm$. A Tungsten collimator is used to limit exposure to specific areas of interest while shielding sensitive electronics components on the board. The collimator was adjusted to limit exposure to Devices D21-D28 on the front-side of the DIMM (Physical Rank=0) as shown in Fig. B.18b (DIMM Device layout defined in Fig. B.8).

Several experiments were performed over a 4-hour period using Si and C species. All experimental runs are summarized in Table B.4. In general, either no bit errors (SEUs) were detected or a Single-Event Functional Interrupt (SEFI) was triggered, during all runs except for Run 8. This was likely due to the low flux of each experimental run, but it hints at the substantial sensitivity of the 3DS memory to radiation. SEFI sensitivity correlates with what was seen during pulsed-laser testing described in Section B.4, where SEFIs were all but guaranteed if the laser was positioned directly above a sensitive area and/or circuit.



(a) Tungsten Collimator (b) Shielding Design (c) Experimental Setup

Figure B.18: **NSRL Adjustable Tungsten Collimator with 3DS Test Setup.** The collimator is shown in the fully retractable position with the red laser alignment lines visible. Collimator is designed to shield all but the 8 selected devices shown in (b). Actual experimental setup after aligning collimator is shown in (c).



(a) Run 24 (10.7 MeV/amu) (b) Run 25 (0 MeV/amu)

Figure B.19: **Example NSRL Runs with Degrader.** 'X' represents devices where bit errors were detected. Both Runs 24 & 25 are examples of runs with catastrophic SEFI events where multiple devices were affected despite the beam being mostly and fully degraded, respectively. Results hint that the 3DS sample was highly sensitive to *secondaries*.

Run	Species	Energy (MeV/amu)	LET	Fluence	Errors Collected	SEUs Collected	SEFI(s) Detected	SEFI Type
7	Si	370	0.5	1e+5	0	0	no	—
8	Si	370	0.5	1e+6	3	3	no	—
9	Si	370	0.5	1e+7	4593	0	yes	row(s)
11	C	300	0.1	6.5e+6	1651	0	yes	row(s)
12	C	300	0.1	1e+6	0	0	no	—
13	C	300	0.1	1e+6	0	0	no	—
14	C	300	0.1	1e+6	0	0	no	—
15	C	300	0.1	1e+6	0	0	no	—
16	C	300	0.1	1e+6	692	0	yes	column
17	C	300	0.1	1e+6	2085	0	yes	row
18	C	104	0.2	1e+6	0	0	no	—
19	C	104	0.2	1e+6	2735	0	yes	row
20	C	104	0.2	1e+6	0	0	no	—
21	C	104	0.2	2e+6	0	0	no	—
22	C	104	0.2	1e+6	335	0	yes	column
23	C	104	0.2	3e+6	585	0	yes	row
24	C	10.7	—	5e+6	1228	0	yes	multi-device
25	C	0	—	5e+6	624	1	yes	row
26	C	16.15	—	5e+6	171	0	yes	column
27	C	18	—	5e+6	512	0	yes	row

Table B.4: **NSRL Testing Run Summary**. All runs were performed on 4-layer (4H) 3DS Memory. Full 3DS DIMM memory readout was not performed every time an error was detected as system was designed to only report on errors on up to 64 memory addresses at a time. Runs with detected SEFI(s) were prematurely halted after reading out errors on 64-1280 addresses. 'Collected' errors reflect this rather than necessarily the total number of bit errors read out from the entire DIMM. Run 8 was unique as it was the only run with Single-Event Upsets (SEUs) detected only.

B.6 General Conclusions

It was determined that 3DS memory is highly-sensitive to radiation. As is true in general for smaller technology nodes, the critical charge to induce a memory upset is substantially lower than older technology nodes. The 3DS memory, however, was not only sensitive to Single-Event Upsets (SEUs) but also highly sensitive to Single-Event Functional Interrupts (SEFIs) as seen during both pulsed-laser testing at Vanderbilt University and heavy ion experiments at NASA Space Radiation Laboratory. Specifically during the laser testing, it was shown that if the laser was positioned directly over a sensitive circuit (e.g. redundancy latch), a SEFI could be repeatably induced at the same location across multiple experimental runs.

In general, studying the effects of radiation on specifically 3D Architectures provides new challenges beyond traditional testing. It was found that Variable-Depth Bragg Peak [BKM11] analysis is non-trivial in more complex 3D-stacked systems utilizing state-of-the-art technology nodes.

Our interpretation is that in principal, each layer in the memory stack can be independently examined given that the amount of attenuation per layer is likely negligible. It is, however, important to point out that NSRL experiments highlighted the the significant effect of secondaries, especially where in Run 25 the beam was fully degraded to 0 MeV/amu, but generated *secondaries* managed to trigger SEFI and SEU event(s) across multiple devices on the DIMM. Because of the overall system complexity and confounding variables, new techniques must be developed to study system reliability of more complex 3D-stacked memories and systems.

APPENDIX C

CTT-Hardware-based Inference Realistic Circuit Universal Simulator

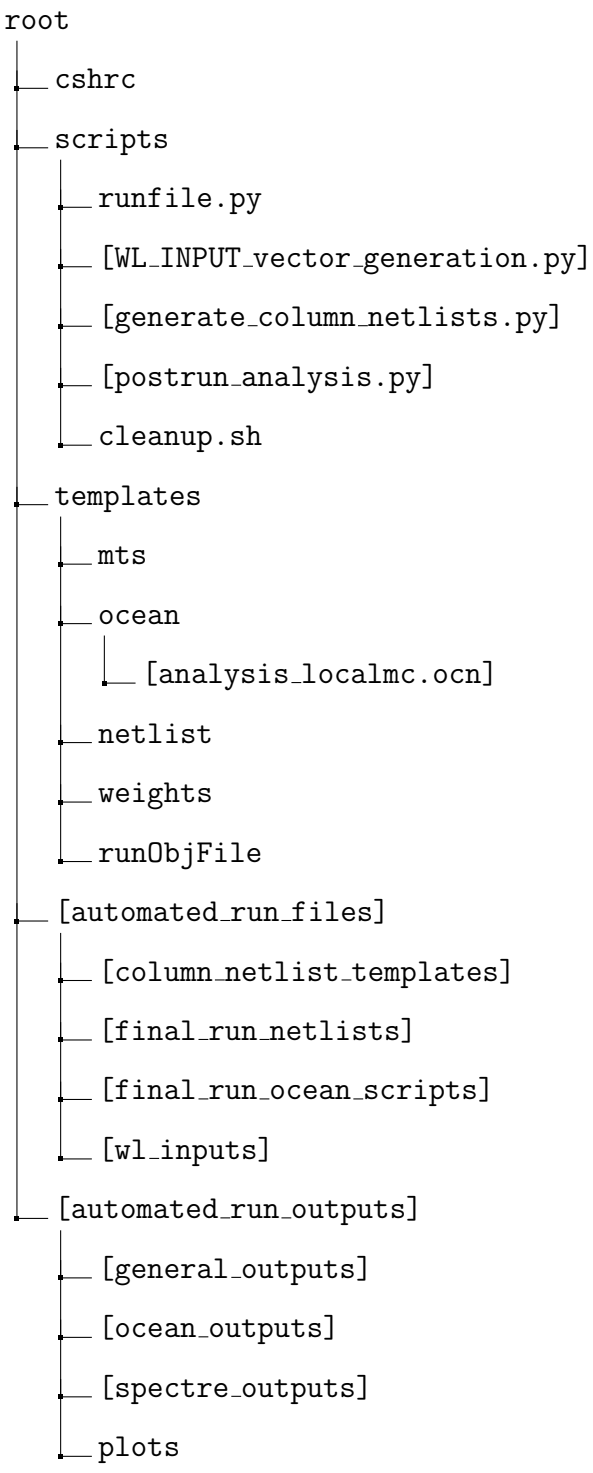
Example codebase is available here: `https://github.com/smoran1/CIRCUS_localmc`

C.1 CIRCUS Overview

CIRCUS is a simulation platform (Fig. 4.24) that allows for a schematic-based design to be automatically parsed & updated with specified input vectors and trained weights. Section C.2 describes how Trained weights are automatically mapped to CTT devices in simulation using an $I_{weight}(nA) \rightarrow \Delta V_{TH}(mV)$ Lookup Table which sets the value of the parameterized voltage attached to the device's gate such that it produces a specific on current when enabled with $V_{WL} = V_{GS} = 200mV$, previously shown in Fig. 4.25.

Section C.3 describes how digital inputs are mapped into the circuit simulation to control the WL driver circuits using vector file generation technique. Section C.4 combines the generated netlist and wl vector files from the previous two sections, and invokes the Cadence Spectre simulations followed by Cadence OCEAN Analysis to extract crucial data from the simulation output waveforms. Automated Data Analysis using Cadence OCEAN is described in more detail in section (Section C.5). Simulations with post-layout parasitic-extracted netlists are also possible for more accurate post-design verification. Spectre Accelerated Parallel Simulator (++aps) is utilized to reduce overall simulation run-time without compensating on simulation accuracy.

File Directory:



```
1  // Library name: NEUROCTT_0P3
2  // Cell name: nfet_ctt
3  // View name: schematic
4  // Inherited view list: spectre cmos_sch cmos.sch schematic veriloga ahdl
5  // pspice dspf
6  subckt nfet_ctt b d g s
7  parameters Vth=0
8      V0 (g net9) vsource dc=Vth type=dc
9      N0 (d net9 s b) nfet w=428n l=20n ...
10 ends nfet_ctt
11 // End of subcircuit definition.
12
13 // Library name: NEUROCTT_0P3
14 // Cell name: ARRAY_CTT_TWIN_CELL
15 // View name: schematic
16 // Inherited view list: spectre cmos_sch cmos.sch schematic veriloga ahdl
17 // pspice dspf
18 subckt ARRAY_CTT_TWIN_CELL BLc BLt SL VSS WL
19 parameters Vth_p Vth_n
20     I0 (VSS BLt WL SL) nfet_ctt Vth=Vth_p
21     I1 (VSS SL WL BLc) nfet_ctt Vth=Vth_n
22 ends ARRAY_CTT_TWIN_CELL
23 // End of subcircuit definition.
```

Table C.1: **Example ARRAY_CTT_TWIN_CELL Netlist Implementation.**

C.2 Spectre Netlist Parsing

Differential weights are stored using the ARRAY_CTT_TWIN_CELL model with parameterized Vth_p & Vth_n gate voltage sources, shown in Table C.1. Example netlist with device weights for a single column with 8 WLs after mapping to $\Delta V_{TH}(mV)$ parameters is shown in Table C.2.

Table C.3 provides a excerpt of the included `generate_column_netlists.py` script where netlist lines corresponding to twin-cell CTT devices are retrieved and modified with provided VTH_T and VTH_C weight parameters after mapping $I_{weight}(nA) \rightarrow \Delta V_{TH}(mV)$ values using a weight lookup table ($templates\backslash weights\backslash WEIGHT\_LUT\_TT.csv$).

```
I0\<0\> (BLc BLt SL VSS WL\<0\>) ARRAY_CTT_TWIN_CELL Vth_p=36.061044m Vth_n=15.503379m
I0\<1\> (BLc BLt SL VSS WL\<1\>) ARRAY_CTT_TWIN_CELL Vth_p=23.271323m Vth_n=31.856741m
I0\<2\> (BLc BLt SL VSS WL\<2\>) ARRAY_CTT_TWIN_CELL Vth_p=28.847506m Vth_n=13.605357m
I0\<3\> (BLc BLt SL VSS WL\<3\>) ARRAY_CTT_TWIN_CELL Vth_p=53.405000m Vth_n=28.581547m
I0\<4\> (BLc BLt SL VSS WL\<4\>) ARRAY_CTT_TWIN_CELL Vth_p=47.045000m Vth_n=31.757343m
I0\<5\> (BLc BLt SL VSS WL\<5\>) ARRAY_CTT_TWIN_CELL Vth_p=28.625991m Vth_n=44.195000m
I0\<6\> (BLc BLt SL VSS WL\<6\>) ARRAY_CTT_TWIN_CELL Vth_p=29.851701m Vth_n=48.395000m
I0\<7\> (BLc BLt SL VSS WL\<7\>) ARRAY_CTT_TWIN_CELL Vth_p=29.274721m Vth_n=26.502806m
```

Table C.2: **Example Spectre Netlist after Weight Insertion.**

```python
# Begin .scs file parsing
with open(spice_netlist_template) as f:

    # create [contents] data array of template file
    for line in f:
        contents.append(line)
    # Parse
    nLinesUpdated = 0
    for i in range(0,len(contents),1):
        # If line contains nfet_ctt (not dummy device):
        if(all(x in contents[i] for x in ['Vth_p','Vth_n']) and ('parameters' not in contents[i])):
            WL=contents[i].split()[0].split("<")[1].split("\\")[0]
            temp_line ='      I0\<' + str(WL)+'\> (BLc BLt SL VSS WL\<' + str(WL) \
                    + '\>) ARRAY_CTT_TWIN_CELL Vth_p=' + str(VTH_T[int(WL)]) \
                    + ' Vth_n=' + str(VTH_C[int(WL)] + '\n')
            updated_contents.append(temp_line)
            nLinesUpdated = nLinesUpdated+1 # sanity check (should be 256 lines / column)
        elif 'seed=12345' in contents[i]:
            temp_line = 'mc1 montecarlo numruns=1 seed=' + str(SEED_NUMBERS[neuron_num]) + ' variations=
                all sampling=standard \\\n'
            updated_contents.append(temp_line)
        else:
            updated_contents.append(contents[i])

    # Output file
    with open(output_netlist_directory + output_netlist_filename + str(neuron_num) + ".scs", "w") as f:
        for line in updated_contents:
            f.write(line)
```

Table C.3: **CIRCUS Column Netlist Generating Python Script.** Subset of provided `generate_column_netlists.py`.

C.3 Input Vector File Generation

Input Vector Files allow users to specify digital inputs (e.g. 8 bit integers between $0-255$), remap them, and then apply them as inputs within a circuit simulation. Example file setup and WL inputs for the first two clocks cycles of inference is shown in Table C.4. This file specifies the values of the 6 digital inputs to each of the WL Drivers for each of the 256 WLs ($WL\langle 0{:}255\rangle$).

Table C.5 provides an excerpt of the included `WL_INPUTS_vector_generation.py` script where specified digital inputs ($WL\langle 0{:}255\rangle$) are mapped to logic voltages for circuit input signals WL_VWL_EN$\langle 0{:}255\rangle$, WL_VREAD_EN$\langle 0{:}255\rangle$, WL_VPP_EN$\langle 0{:}255\rangle$, and their respective complements.

```
1  radix 4444444444444444444444444444444444444444444444444444444444444444
       4444444444444444444444444444444444444444444444444444444444444444
       4444444444444444444444444444444444444444444444444444444444444444
       4444444444444444444444444444444444444444444444444444444444444444
       4444444444444444444444444444444444444444444444444444444444444444
       4444444444444444444444444444444444444444444444444444444444444444
2
3  io i i i i i i
4  vname WL_VWL_EN<<0:255>> WL_VWL_EN_N<<0:255>> WL_VREAD_EN<<0:255>> WL_VREAD_EN_N<<0:255>> WL_VPP_EN
       <<0:255>> WL_VPP_EN_N<<0:255>>
5  tunit ns
6  trise 1
7  tfall 1
8  vil 0
9  vih 0.8
10 vol 0
11 voh 0.8
12
13 ; format time vector
14 ; t=0ns
15 0 0000000000000000000000000000000000000000000000000000000000000000
       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       0000000000000000000000000000000000000000000000000000000000000000
       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       0000000000000000000000000000000000000000000000000000000000000000
       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
16 ;Cycle: 1(t=3400ns)
17 3400 0000000820000000000080800000000000400020000001000004002000100000
       FFFFFFF7DFFFFFFFFFFF7F7FFFFFFFFFFFBFFFDFFFFFFEFFFFFBFFDFFFEFFFFF
       0000000820000000000080800000000000400020000001000004002000100000
       FFFFFFF7DFFFFFFFFFFF7F7FFFFFFFFFFFBFFFDFFFFFFEFFFFFBFFDFFFEFFFFF
       0000000000000000000000000000000000000000000000000000000000000000
       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
18 ;Cycle: 2(t=3405ns)
19 3405 0000000820000000000080800000000000400020000001000004002000100000
       FFFFFFF7DFFFFFFFFFFF7F7FFFFFFFFFFFBFFFDFFFFFFEFFFFFBFFDFFFEFFFFF
       0000000820000000000080800000000000400020000001000004002000100000
       FFFFFFF7DFFFFFFFFFFF7F7FFFFFFFFFFFBFFFDFFFFFFEFFFFFBFFDFFFEFFFFF
       0000000000000000000000000000000000000000000000000000000000000000
       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

Table C.4: **Example CIRCUS WL Vector File Output.** Output shown for the first 2 compute cycles (out of 256 input cycles.) Generated logic signals provide differential digital input signals to each of the 256 WL Drivers.

```
def generate_wl_input_vector_files(X_INPUTS):

    # (1) Generate WL Inputs per input-sample & per neuron (simulate 1 neuron @ a time)
    vec_contents = []
    nSamples = len(X_INPUTS)
    for run_num in range(0,nSamples,1):

        ...

        # (a) Prepare WL inputs for specific run #run_num
        # ***********************************************
        nWLs = len(X_INPUTS[0]) #
        WL_INPUTS = [0] * nWLs
        for index, val in enumerate(X_INPUTS[run_num],0):
            WL_INPUTS[index] = val

        ...

        # (d.1) Start Inference
        t_inf_start = 3400; #ns
        Tcycle = 5; #ns
        nMaxCycles = 256;
        for i in range(1, nMaxCycles+1, 1):
            time_step = Tcycle*(i-1) + t_inf_start
            line = str(time_step) + ' '
            vwl_en_temp = ''
            vwl_en_n_temp = ''

            # Convert 4-bits (1 hex char.) at a time
            for j in range(0,nWLs,4):
                tmp, tmp_n = convertWLInputsToHex(WL_INPUTS[j:j+3])
                vwl_en_temp = vwl_en_temp + tmp.split('0x')[1].upper()
                vwl_en_n_temp = vwl_en_n_temp + tmp_n.split('0x')[1].upper()

            vread_en_temp = vwl_en_temp
            vread_en_n_temp = vwl_en_n_temp
            line = line + vwl_en_temp + ' ' + vwl_en_n_temp + ' ' + vread_en_temp + ' ' + vread_en_n_temp

            # VPP
            line = line + ' 0'
            for _ in range(1,nWLs//radix,1):
                line = line + '0'
            line = line + ' F'
            for _ in range(1,nWLs//radix,1):
                line = line + 'F'

            vec_contents.append(';Cycle: ' + str(i) + '(t=' + str(time_step) + ')')
            vec_contents.append(line)
```

Table C.5: **CIRCUS WL Vector File Generation Python Script.** Subset of provided `WL_INPUT_vector_generation.py`.

C.4 Run File

Table C.6 provides an example runfile which (1) generates WL input vector files, (2) generates column netlist templates with imported weights, (3) creates run netlists with run and input information, (4) creates an ocean analysis script for each run from a template file, (5) invokes the Spectre circuit simulation, and (6) invokes Cadence OCEAN using the generated ocean analysis script to analyze the simulation results.

C.5 Data Analysis using Cadence OCEAN

Cadence OCEAN allows user to generate SKILL-based scripts for automatically parsing simulation outputs. An analysis script similar to that shown in Table C.8 is used to parse the `*.raw` simulation output and automatically extract important information such as the pulse-width of the neuron's output signal. Additionally, mathematical integration can be performed on specified signals such as the differential input current presented to integrator's input by the CTT array (e.g. $Q_{INF} \sim \int_0^T (I_{BLt}(t) - I_{BLc}(t))\, dt$).

Example CIRCUS analysis results for N=32 neurons and 50 simulation runs per neuron with random inputs and randomly initialized weights are plotted in Fig. 4.28 in Section 4.4.

```
# (1) Generate WL Input (.vec) Files
WL_INPUTS = np.random.randint(256, size=(nSamples, nWLs))
generate_wl_input_vector_files(WL_INPUTS)
# (2) Generate column netlist templates
np.random.seed(12977511)
SEED_NUMBERS = np.random.randint(500000,size=nNeurons)
T_WEIGHTS, C_WEIGHTS, T_WEIGHTS_VTH, C_WEIGHTS_VTH = generate_random_column_netlists_localmc(nNeurons,
    SEED_NUMBERS)
# (3) Generate run netlists
OFFSET_POS_TRIM_BITS = np.zeros(nNeurons)
OFFSET_NEG_TRIM_BITS = np.zeros(nNeurons)
neuron_run_list = list(range(0,nNeurons,1))
for run_num in range(1,nSamples+1,1):
    for neuron_num in neuron_run_list:
        contents = []
        with open(common_directory + template_neuron_netlist + str(neuron_num) + ".scs") as f:
            for line in f:
                contents.append(line)
            for i in range (0,len(contents),1):
                # (a) Insert 'vec_include' command with input vector file
                if 'vec_include' in contents[i]:
                    contents[i] = 'vec_include "' + wl_input_vec_filename + str(run_num) + '.vec" autostop
                        =false\n'
                # (b) Change OFFSET_POS_TRIM Settings
                ...
# (4) Generate ocean script files
for run_num in range(1,nSamples+1,1):
    for neuron_num in neuron_run_list:
        with open(common_directory + ocean_template_filename) as f:
            contents = []
            for line in f:
                contents.append(line)
            for i in range(0,len(contents),1):
                if 'openResults' in contents[i]:
                    contents[i] = 'openResults("' + common_directory + 'automated_run_outputs/
                        spectre_outputs/neuron_' + str(neuron_num) + '_run_' + str(run_num) + '.raw")\n'
                ...
# (5) Run Spectre Simulation
for run_num in range(1,nSamples+1,1):
    for neuron_num in neuron_run_list:
        scs_filename = common_directory + final_run_netlist + str(neuron_num) + '_run_' + str(run_num) + '
            .scs'
        os.system("spectre " + scs_filename + " ++aps +multithread")
# (6) Run Ocean analysis scripts
for run_num in range(1,nSamples+1,1):
    for neuron_num in neuron_run_list:
        ocean_filename = common_directory + final_run_ocean + str(neuron_num) + '_run_' + str(run_num) + '
            .ocn'
        os.system('ocean -restore ' + ocean_filename)
```

Table C.6: **CIRCUS Main Runfile Python Script.** Subset of provided `runfile.py` script.

```
openResults("/u1/ee/smoran2/Desktop/CIRCUS_NEW/neuron_sim_tt_85C_localmc/automated_run_outputs/
    spectre_outputs/neuron_0_run_10.raw")

x1 = 3.40e-06
x2 = 4.68e-06
BLt_integrated = integ(leafValue( i("I2:7" ?result "tran") "iteration" 1) x1 x2 "_")
BLc_integrated = integ(leafValue( i("I2:3" ?result "tran") "iteration" 1) x1 x2 "_")

rising_edge = cross(leafValue( v("NEURON_OUTPUT" ?result "tran") "iteration" 1 ) 0.45 0 'rising)
falling_edge = cross(leafValue( v("NEURON_OUTPUT" ?result "tran") "iteration" 1 ) 0.45 0 'falling)

PWM_0P9_total_time = 0.0

; Ideally the NEURON_OUTPUT should be a single pulse with one rising edge and one falling edge
if( length(rising_edge)==1 && length(falling_edge)==1 then
        PWM_0P9_total_time = nthelem(1 falling_edge)-nthelem(1 rising_edge)
        ;println(PWM_total_time)
; Condition for if the output is multiple pulses (e.g. 2 pulses with 2 rising edges & 2 falling edges)
else
        if(length(rising_edge) == length(falling_edge) then
                n = length(rising_edge)
                for( i 1 n
                        PWM_0P9_total_time = PWM_0P9_total_time + nthelem(i falling_edge) - nthelem(i
                            rising_edge)
                )
        )
)

p = outfile("../automated_run_outputs/ocean_outputs/neuron_0_run_10.txt" "w")
fprintf(p "PWM (0.9V) Total Duration: %g (sec)\n" PWM_0P9_total_time)
fprintf(p "BLt current (integrated) is: %g (C)\n" BLt_integrated)
fprintf(p "BLc current (integrated) is: %g (C)\n" BLc_integrated)
close(p)
```

Table C.7: **Example CIRCUS-generated OCEAN Analysis Script.** Analysis is performed on the Spectre simulation output (`*.raw`). The differential input current presented to the integrator by the CTT array is mathematically integrated and the output pulse width is calculated. Example is shown in Table C.8.

```
PWM (0.9V) Total Duration: 9.99515e-09 (sec)
BLt current (integrated) is: 3.34673e-12 (C)
BLc current (integrated) is: 3.99835e-12 (C)
```

Table C.8: **Example CIRCUS Run Output after Cadence OCEAN Analysis.**

APPENDIX D

NeuroCTT Design & Top-Level Verification

D.1 Digital I/O Architecture

A simple logic interface has been developed allowing a majority of the testing complexity to be shifted to the FPGA-to-MATLAB UART-interface discussed in Section 3.4. Chip logic operation consists of data load operations, instruction execution, & retrieving neuron time-to-digital (TDC) output from the chip. All data load & unload operations are performed using Serial-In (SI), Serial-Out (SO) communication protocols. Figures D.1 and D.2 demonstrate executing a program on the chip (PROGRAM_EXECUTE) as well as loading data, instruction, and configuration parameters to the chip. While, omitted here, output data can be similarly retrieved from the chip by sending a a one-cycle (RETRIEVE_NEURON_COUNT) pulse which leads to the contents of the 320b output register to be serially outputted from the chip the COUNT_DATA_OUT pin.

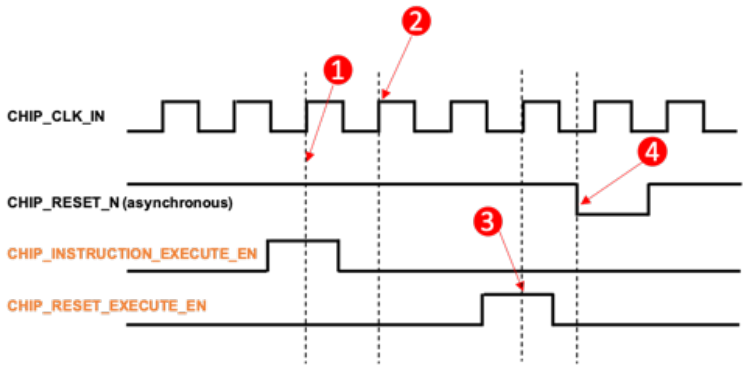
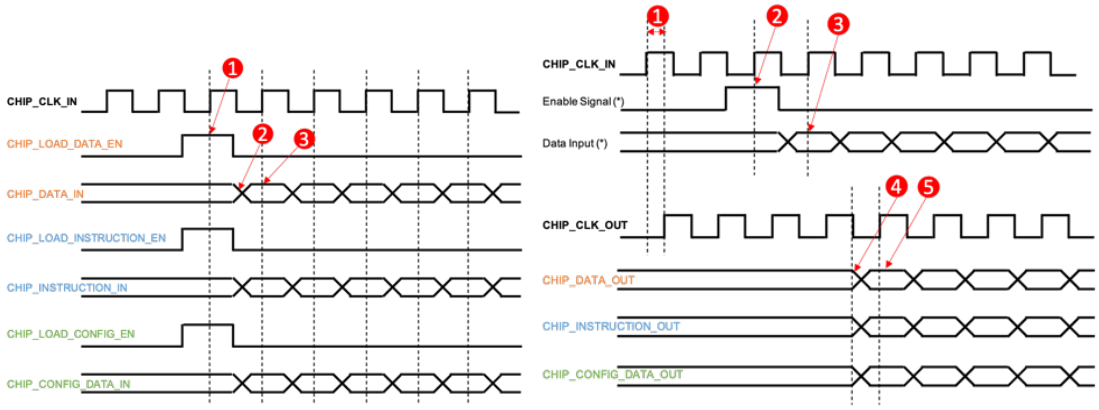


Figure D.1: **NeuroCTT 0.3 I/O Program Execute Enable.**



(a) Input Datapath (b) Output Datapth

Figure D.2: **NeuroCTT 0.3 I/O Architecture.** (a) There are 3 serial input data paths which load data into three input data shift registers: DATA_REG, INSTRUCTION_REG, and CONFIG_REG, respectively. Data load can happen concurrently or staggered (concurrent data load of the three datapaths is shown in the figure above). Input data is flopped on positive edge of clock. (b) Previous shift register contents are outputted MSB-first on respective output pins during a data load.

D.2 Testbench Vector Generator

Example codebase is available here:

`https://github.com/smoran1/Verilog_Test_Vector_Generator`

A python-based compiler or test vector generator was designed to generate CSV-based test vectors that can be used to run pre- and post-synthesis Verilog simulations as well as chip-level Analog Mixed-Signal (AMS) verification simulations.

Verilog Testbench Vector Generator File Directory:

```
root
├── functions.py
├── example_scripts
│   ├── [run_inference.py]
│   ├── [run_programming.py]
│   ├── [run_verification.py]
│   ├── [run_offchip_ver.py]
│   └── ...
├── [example_testbenches]
│   └── [LOGIC_SYSTEM_TB.v]
└── outputs
    ├── [IO_FILE_INFERENCE.csv]
    ├── [IO_FILE_PRG.csv]
    └── ...
```

The TB Vector Generator produces a CSV output file which can be loaded into a pre- or post-synthesis Verilog simulation for expediting chip design verification. An example test vector (CSV) output is provided in Table D.1.

```
1  RESET_N,LOAD_DATA_EN,DATA_IN,LOAD_INSTRUCTION_EN,INSTRUCTION_IN,LOAD_CONFIG_EN,CONFIG_DATA_IN,
       RETRIEVE_NEURON_COUNT,PROGRAM_EXECUTE_EN,RESET_COUNT_EN,ANALOG_NEURON_IN
2  1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
3  1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
4  1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
5  0,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
6  1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
7  1,0,0,1,0,0,0,0,0,0,00000000000000000000000000000000
8  1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
9  1,0,0,0,1,0,0,0,0,0,00000000000000000000000000000000
10 1,0,0,0,1,0,0,0,0,0,00000000000000000000000000000000
11 1,0,0,0,1,0,0,0,0,0,00000000000000000000000000000000
12 1,0,0,0,1,0,0,0,0,0,00000000000000000000000000000000
13 1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
14 1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
15 1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
16 1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
17 1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
18 1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
19 1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
20 1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
21 1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
22 1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
23 1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
24 1,1,0,0,0,0,0,0,0,0,00000000000000000000000000000000
25 1,0,0,0,0,0,0,0,0,0,00000000000000000000000000000000
26 ...
```

Table D.1: **Example TB Generator (CSV) Output File.** Each row corresponds to inputs that will be applied to the LOGIC_SYSTEM block at different clock cycles. The final column ("ANALOG_NEURON_IN") corresponds to example neuron pulse-width-modulated outputs to test functionality of the Time-to-Digital Converter (TDC) block.

D.2.1 Testbench Vector Generator: Header

Each python-based run-file begins with a header section defining necessary parameters for the compiler, shown in Table D.2. Users can customize the Setup or Chip Configuration Parameters shown in Table D.3. This section loads default values but can be modified later again in the script.

D.2.2 Test Vector Generator: Run Examples

After the *Header* and *Setup Parameters* sections, the user can utilize a predefined set of functions comprising data, instruction, and configuration loads to the chip as well as applying chip resets (RESET_N) and executing the loaded program (PROGRAM_EXECUTE_EN). Predefined functions are provided in the **functions.py** file. An example run file for running an INFERENCE simulation is provided in Table D.4.

D.3 Verilog Testbench

Tables D.5 & D.6 provide an example complementary testbench for loading the generated CSV vector file from Tables D.2- D.4 into the LOGIC_SYSTEM block.

```
import csv
from functions import *

run_name = "outputs/IO_FILE_INFERENCE"

input_signals = ['RESET_N', 'LOAD_DATA_EN', 'DATA_IN',\
    'LOAD_INSTRUCTION_EN', 'INSTRUCTION_IN', 'LOAD_CONFIG_EN',\
    'CONFIG_DATA_IN','RETRIEVE_NEURON_COUNT', 'PROGRAM_EXECUTE_EN',\
    'RESET_COUNT_EN', 'ANALOG_NEURON_IN']

instruction_dict = {
    # Programming Instructions
    "PRG_T" : 1,
    "PRG_C" : 2,
    "PRG_T_NO_FLOAT" : 3,
    "PRG_C_NO_FLOAT" : 4,
    "PRG_T_REVERSE" : 5,
    "PRG_C_REVERSE" : 6,
    "PRG_T_REVERSE_NO_FLOAT" : 7,
    "PRG_C_REVERSE_NO_FLOAT" : 8,
    "PRG_DEBUG" : 17,
    "PRG_TWIN_CELL" : 19,
    "PRG_TWIN_CELL_NO_FLOAT" : 20, # not implemented
    "PRG_TWIN_CELL_REVERSE" : 21,

    # Verification Instructions
    "VERIFY_OCV" : 9, # uses neuron
    "VERIFY_OCV_T" : 10, # same as VERIFY_OCV
    "VERIFY_OCV_C" : 11, # same as VERIFY_OCV
    "VERIFY_T_OFFCHIP" : 12,
    "VERIFY_C_OFFCHIP" : 13,
    "VERIFY_TWIN_OFFCHIP" : 14,
    "VERIFY_DEBUG" : 18,
    "VERIFY_OCV_CG_AMP_ONLY" : 25, # uses neuron

    # Inference Instructions
    "INF" : 15,
    "INF_ZERO_INPUT" : 16,
    "INF_MEASURE_BL_VOLTAGE" : 22,
    "INF_CG_AMP_ONLY" : 23,
    "INF_CG_AMP_ZERO_INPUT" : 24,
    "INF_DEBUG" : 26,
    "INF_ALWAYS_ON" : 27,
    "INF_OFF_CHIP" : 28,
    "INF_ORIGINAL_SWITCH_LOGIC" : 29,
    "INF_SWAPPING" : 30

}
```

Table D.2: **TB Vector Generator Run File: Header.**

```
# Setup Parameters
# ================================================
# Fine-tune offset correction
OFFSET_POS_FINE_TUNE = [0] * 32;
OFFSET_NEG_FINE_TUNE = [0] * 32;
# Trim offset correction
OFFSET_POS_TRIM = [0] * 32;
OFFSET_NEG_TRIM = [0] * 32;

NEURON_EN = [1] * 32;
neuron_parameters = {
    "T_CLK_MHz" : 100,          # Valid Range: 0-400 [MHz]
    "OFFSET_POS_FINE_TUNE" : OFFSET_POS_FINE_TUNE, # [32-parameters] Valid Range (10-bits): 0-1023 cycles
    "OFFSET_NEG_FINE_TUNE" : OFFSET_NEG_FINE_TUNE, # [32-parameters] Valid Range (10-bits): 0-1023 cycles
    "NEURON_EN" : NEURON_EN,
    "USE_BOOSTED_INTEGRATOR" : True,
    "CAP_SETTING" : "6.6pF",    # Options: {600fF, 1.8pF, 2.4pF, 3.0pF, 3.6pF, 4.2pF, 4.8pF, 5.4pF, 6.6pF}
    "INTEGRATOR_EN_X" : 30,       # Valid Range (9-bits): 0-511 cycles
    "INTEGRATOR_SETUP" : 20,      # Valid Range (9-bits): 0-511 cycles
    "INTEGRATOR_EN_Y" : 1,        # Valid Range (4-bits): 0-15 cycles
    "COMPARATOR_EN_X" : 30,       # Valid Range (9-bits): 0-511 cycles (Default = INTEGRATOR_EN_X)
    "INFERENCE_DURATION" : 255,    # Valid Range (10-bits): 0-1023 cycles (Formally known as 'MAX_WL_INPUT')
    "T_DUMMY_X" : 0,              # Valid Range (8-bits): 0-255 cycles (dummy wl input)
    "INTEGRATION_SWITCH_EN_X" : 10, # Valid Range (8-bits): 0-255 cycles
    "INTEGRATION_SWITCH_EN_Y" : 1, # Valid Range (4-bits): 0-15 cycles
    "MIN_PULSE_X" : 1,           # Valid Range (5-bits): 0-31 cycles
    "OR_EN_X" : 1,                # Valid Range (4-bits): 0-15 cycles
    "OR_EN_Y" : 1,                # Valid Range (4-bits): 0-15 cycles
    "DISCHARGE_EN" : 300,         # Valid Range (10-bits): 0-1023 cycles
    "RESET_EN_X" : 10,              # Valid Range (8-bits): 0-255 cycles
    "RESET_EN_Y" : 10,              # Valid Range (8-bits): 0-255 cycles
    "T_SWAPPING" : 100,                # Valid Range (8-bits): 0-255 cycles (applicable to INF_SWAPPING inst)
    "UTILIZE_FLIPPED_BIAS_CONFIG" : 0, # Valid Range (1-bit): 0-1 (0: normal_config, 1: flipped_config)
    "OFFSET_POS_TRIM" : OFFSET_POS_TRIM,     # [32-parameters] Valid Range (8-bits)
    "OFFSET_NEG_TRIM" : OFFSET_NEG_TRIM,     # [32-parameters] Valid Range (8-bits)
    "COLUMN_SELECT" : 0,            # Valid Range (5-bits): select neuron 0-31
    "DEBUG_MUX_SELECT" : 5,         # Valid Range (5-bits): 0: none selected, 1-15: selects respective input
}

# 1ms = 40,000 cycles @ 40MHz
prg_parameters = {
    "COLUMNS_SELECTED" : "11000000000000000000000000000000", # 32-bit debug param. (COLUMNS_SELECTED<0:31>)
    "BL_PULSE_WIDTH_X" : 100, # Valid Range (16-bits): 0-65535 cycles
    "BL_PULSE_WIDTH_Y" : 100, # Valid Range (16-bits): 0-65535 cycles
    "SL_START" : 5,  # Valid Range (4-bits): 0-15 cycles
    "SL_END" : 5,    # Valid Range (4-bits): 0-15 cycles
    "WL_PULSE_WIDTH" : 100 # Valid Range (16-bits): 0-65535 cycles
}
```

Table D.3: **TB Vector Generator Run File: Setup Parameters.**

```
input_vectors = []
check_neuron_parameters(neuron_parameters)
check_prg_parameters(prg_parameters)
check_debug_parameters(debug_parameters)

# Commands
# ================================================
idle(input_signals,input_vectors,2) #idle for 2 cycles
apply_reset(input_signals,input_vectors)

# (1) Initiate INF operation
instruction = "INF"
load_instruction(input_signals,input_vectors,instruction_dict,instruction) # max 5-bit value
idle(input_signals,input_vectors,10) #idle for 10 cycles

WL_DATA = [0] * 256
for i in range(128):
    WL_DATA[i] = 201;

# Expected neuron outputs (inputs to TDCs)
NEURON_OUTPUTS = [0] * 32
for i in range(32):
    NEURON_OUTPUTS[i] = i;

load_wl_data(input_signals,input_vectors,WL_DATA)
idle(input_signals,input_vectors,10) #idle for 10 cycles
load_wl_data(input_signals,input_vectors,WL_DATA)
idle(input_signals,input_vectors,10) #idle for 10 cycles
load_config_data(input_signals,input_vectors,neuron_parameters)
idle(input_signals,input_vectors,10) #idle for 10 cycles
load_config_data(input_signals,input_vectors,neuron_parameters)
idle(input_signals,input_vectors,10) #idle for 10 cycles
program_execute(input_signals,input_vectors,neuron_parameters,prg_parameters,instruction,NEURON_OUTPUTS)

# Retrieve inference data (neuron outputs: 320-bits)
retrieve_inference_data(input_signals,input_vectors)
idle(input_signals,input_vectors,10) #idle for 10 cycles

# Write .csv file
# ================================================
with open(run_name + '.csv','w') as csvfile:
    csvwriter = csv.writer(csvfile)
    # write header
    csvwriter.writerow(input_signals)
    # write data
    csvwriter.writerows(input_vectors)
```

Table D.4: **Example TB Vector Generator Run File INFERENCE Mode.** User (1) applies reset, (2) load "INF" instruction, (3) loads input or WL data, (4), loads neuron/chip configuration data, (5) executes the program, and (6) retrieves the neuron time-to-digital converted outputs.

```
module LOGIC_SYSTEM_TB();
// IO Filename
reg[63*8:0] input_filename = "IO/IO_FILE_INFERENCE.csv";
// ------------------------------------------
// Parameters, Signal Declarations & Instantiations
// ------------------------------------------
parameter HalfCycle = 5000; //ps
localparam Cycle = 2*HalfCycle;
...

LOGIC_SYSTEM DUT(
    // GPIO Inputs
    .CLK_IN(Clock), .RESET_N(reset_n), .LOAD_DATA_EN(load_data_en), .DATA_IN(data_in),
    .LOAD_INSTRUCTION_EN(load_instruction_en), .INSTRUCTION_IN(instruction_in),
    .LOAD_CONFIG_EN(load_config_en), .CONFIG_DATA_IN(config_data_in),
    .RETRIEVE_COUNT_DATA_EN(retrieve_count_data_en), .PROGRAM_EXECUTE_EN(program_execute_en),
    .RESET_COUNT_EN(reset_count_en),
    // Outputs
    ...
);

initial Clock = 1'b0;
always #(HalfCycle) Clock = ~Clock;
// ------------------------------------------
// Load Input Vector Files
// ------------------------------------------
reg[200*8-1:0] header;
reg isSimulationEnd;
reg startUpCompleted;
integer logFile;
integer inputFile;
initial begin
        inputFile = $fopen(input_filename,"rb");
        if (inputFile == 0)
        begin
                $display("Error at opening file: IO/IO_FILE.csv");
                $stop;
        end else begin
                //print & discard first line (header)
                if ($fscanf(inputFile, "%s\n",header) <1) begin
                        isSimulationEnd = 1;
                end else begin
                        $display("%s\n",header);
                end
        end
end
```

Table D.5: **Example Verilog TB with CSV Input Vector.**

```
// ------------------------------------------
// Apply Test Stimulus
// ------------------------------------------
initial begin
    ...
end

reg tmp_reset_n;
reg tmp_load_data_en, tmp_data_in;
reg tmp_load_instruction_en, tmp_instruction_in;
reg tmp_load_config_en, tmp_config_data_in;
reg tmp_retrieve_count_data_en;
reg tmp_program_execute_en;
reg tmp_reset_count_en;
reg [31:0] tmp_analog_neuron_in;
integer temp;
always @(negedge Clock) begin
        if (startUpCompleted) begin
                if ($feof(inputFile) != 0) begin
                        isSimulationEnd = 1;
                end else if ($fscanf(inputFile, "%b %b %b %b %b %b %b %b %b %b %b\n",
                     tmp_reset_n, tmp_load_data_en, tmp_data_in, tmp_load_instruction_en,
                     tmp_instruction_in, tmp_load_config_en, tmp_config_data_in, tmp_retrieve_count_data_en
                     , tmp_program_execute_en, tmp_reset_count_en, tmp_analog_neuron_in) <1) begin
                        isSimulationEnd = 1;
                end else begin
                        // Apply inputs
                        reset_n <= tmp_reset_n;
                        load_data_en <= tmp_load_data_en;
                        data_in <= tmp_data_in;
                        load_instruction_en <= tmp_load_instruction_en;
                        instruction_in <= tmp_instruction_in;
                        load_config_en <= tmp_load_config_en;
                        config_data_in <= tmp_config_data_in;
                        retrieve_count_data_en <= tmp_retrieve_count_data_en;
                        program_execute_en <= tmp_program_execute_en;
                        reset_count_en <= tmp_reset_count_en;
                        analog_neuron_in <= tmp_analog_neuron_in;
                end
        end
end
// End of Simulation
always @(isSimulationEnd) begin
        if (isSimulationEnd) begin
                $stop;
        end
end
```

Table D.6: **Example Verilog Testbench with CSV Input Vector (continued).** Each row of inputs in CSV test vector file is applied at the negative edge of the clock allowing for a half-cycle for setup time.

REFERENCES

[Aga18] Abien Fred Agarap. "Deep Learning using Rectified Linear Units (ReLU)." *arXiv*, pp. 1–7, 2018.

[Ami13] Arnon Amir et al. "Cognitive computing programming paradigm: A Corelet Language for composing networks of neurosynaptic cores." In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10, 2013.

[Ana21] AnandTech. "Apple Announces M1 Pro & M1 Max: Giant New Arm SoCs with All-Out Performance." `https://www.anandtech.com/show/17019/apple-announced-m1-pro-m1-max-giant-new-socs-with-allout-performance`, 2021. Accessed: 2022-02-13.

[APR20] Vineet Agrawal, V. Prabhakar, K. Ramkumar, Swatilekha Saha Long Hinh, Santanu Samanta, and Ravindra Kapre. "In-Memory Computing array using 40nm multibit SONOS achieving 100 TOPS/W energy efficiency for Deep Neural Network Edge Inference Accelerators." In *2020 IEEE International Memory Workshop (IMW)*, pp. 1–4, 2020.

[BCN18] Leon Bottou, Frank E. Curtis, and Jorge Nocedal. "Optimization Methods for Large-Scale Machine Learning." *arXiv*, pp. 1–95, 2018.

[BGK16] F. Merrikh Bayat, X. Guo, M. Klachko, N. Do, K. Likharev, and D. Strukov. "Model-based high-precision tuning of NOR flash memory cells for analog computing applications." In *2016 74th Annual Device Research Conference (DRC)*, 2016.

[BJP17] Adeel A. Bajwa, Siva C. Jangam, Saptadeep Pal, N. Marathe, T. Bai, Tak Fukushima, Mark Goorsky, and Subramanian S. Iyer. "Heterogeneous Integration at Fine Pitch ($\leq$10μm) using Thermal Compression Bonding." In *Proceedings of 67th IEEE Electronic Components and Packaging Technology (ECTC)*, pp. 1276–1284, 2017.

[BJP18] Adeel Bajwa, Siva C. Jangam, Saptadeep Pal, Boris Vaisband, Randall Irwin, Mark Goorsky, and Subramanian S. Iyer. "Demonstration of a Heterogeneously Integrated System-on-Wafer (SoW) assembly." In *Proceedings of 68th IEEE Electronic Components and Technology Conference (ECTC)*, pp. 1926–1930, 2018.

[BKM11] S. Buchner, N. Kanyogoro, D. McMorrow, C. C. Foster, Patrick M. O'Neill, and Kyson V. Nguyen. "Variable Depth Bragg Peak Method for Single Event Effects Testing." In *IEEE Transactions on Nuclear Science (TNS)*, volume 58, pp. 2976–2982, 2011.

[BM19] Rachel M Brewer, Steven L Moran, et al. "The Impact of Proton-Induced Single Events on Image Classification in a Neuromorphic Computing Architecture." *IEEE Transactions on Nuclear Science*, **67**(1):108–115, 2019.

[BMM21] Evelyn T. Breyer, Halid Mulaosmanovic, Thomas Mikolajick, and Stefan Slesazeck. "Perspective on ferroelectric, hafnium oxide based transistors for digital beyond von-Neumann computing." *Applied Physics Letters*, **118**(050501):1–7, 2021.

[BMR20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. "Language Models are Few-Shot Learners." In *arXiv*, pp. 1–75, 2020.

[Bot98] Leon Bottou. "Online Learning and Stochastic Approximations." Technical report, AT&T Labs–Research, 1998.

[BP01] Guo qiang Bi and Mu ming Poo. "Synaptic Modification by Correlated Activity: Hebb's Postulate Revisited." *Annual Review of Neuroscience*, **24**:139–166, 2001.

[BSN14] G.W. Burr, R.M. Shelby, C. di Nolfo, J.W. Jang, R.S. Shenoy, P. Narayanan, et al. "Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses) using phase-change memory as the synaptic weight element." In *2014 IEEE International Electron Devices Meeting (IEDM)*, pp. 1–4, 2014.

[BV20] S. Ashwin Balagopal and Janakiraman Viraraghavan. "Flash Based In-Memory Multiply-Accumulate Realisation: A Theoretical Study." In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2020.

[BZG21] Rachel M Brewer, En Xia Zhang, Mariia Gorchichko, Peng Fei Wang, Jonathan Cox, Steven L Moran, Dennis R Ball, Brian D Sierawski, Daniel M Fleetwood, Ronald D Schrimpf, Subramanian S Iyer, and Michael L Alles. "Total Ionizing Dose Responses of 22-nm FDSOI and 14-nm Bulk FinFET Charge-Trap Transistors." *IEEE Transactions on Nuclear Science*, **68**(5):677–686, 2021.

[CCG21] Zhengyu Chen, Xi Chen, and Jie Gu. "A 65nm 3T Dynamic Analog RAM-Based Computing-in- Memory Macro and CNN Accelerator with Retention Enhancement, Adaptive Analog Sparsity and 44TOPS/W System Energy Efficiency." In *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 9–10, 2021.

[Che18] Wei-Hao Chen et al. "A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors." In *International Solid-State Circuits Conference (ISSCC)*, pp. 494–495, 2018.

[CHI20] CHIPS. "DESIGN MANUAL FOR SILICON INTERCONNECT FABRIC TECHNOLOGY." Technical report, University of California Los Angeles, 2020.

[CLF21] Yu-Der Chih, Po-Hao Lee, Hidehiro Fujiwara, Yi-Chun Shih, et al. "An 89TOPS/W and 16.3TOPS/mm^2 All-Digital SRAM-Based Full-Precision Compute-In Memory Macro in 22nm for Machine-Learning Edge Applications." In *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 252–253, 2021.

[CM19] Yanran P. Chen, Pierre Maillard, et al. "Single Event Evaluation of Xilinx 16nm Ultrascale+ High-Bandwidth Memory Enabled FPGA." In *2019 IEEE Radiation Effects Data Workshop*, pp. 1–5, 2019.

[Cor20] Justin Correll et al. "A Fully Integrated Reprogrammable CMOS-RRAM Compute-in-Memory Coprocessor for Neuromorphic Applications." In *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, volume 6, pp. 36–44, 2020.

[Dav18] Mike Davies et al. "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning." In *IEEE Micro*, volume 38, pp. 82–99, 2018.

[Don20] Qing Dong et al. "A 351TOPS/W and 372.4GOPS Compute-in-Memory SRAM Macro in 7nm FinFET CMOS for Machine-Learning Applications." In *International Solid-State Circuits Conference (ISSCC)*, pp. 242–243, 2020.

[Ess16] Steven K. Esser et al. "Convolutional networks for fast, energy-efficient neuromorphic computing." In *Proceedings of the National Academy of Sciences (PNAS)*, volume 113, p. 11441–11446, 2016.

[Fur14] Steve B. Furber et al. "The SpiNNaker Project." In *Proceedings of IEEE*, volume 102, pp. 652–665, 2014.

[Fur16] Steve Furber. "Large-scale neuromorphic computing systems." *Journal of Neural Engineering*, **13**(5):1–14, 2016.

[Gao19] Bilwaj Gaonkar et al. "Quantitative Analysis of Neural Foramina in the Lumbar Spine: An Imaging Informatics and Machine Learning Study." *Radiology: Artificial Intelligence*, **1**:1–6, 2019.

[GI17] Xuefeng Gu and Subramanian S. Iyer. "Unsupervised Learning using Charge-Trap Transistors." In *IEEE Electron Device Letters*, volume 38, pp. 1204–1207, 2017.

[Gu18] Xuefeng Gu. *Charge-Trap Transistors for Neuromorphic Computing*. PhD thesis, University of California, Los Angeles, 2018.

[GWI19] Xuefeng Gu, Zhe Wan, and Subramanian S. Iyer. "Charge-Trap Transistors for CMOS-Only Analog Memory." In *IEEE Electron Device Letters*, volume 66, pp. 4183–4187, 2019.

[Ham50] Richard Wesley Hamming. "Error Detecting and Error Correcting Codes." *Bell System Technical Journal*, **29**(2):147–160, 1950.

[He20] Wangxin He et al. "2-Bit-Per-Cell RRAM-Based In-Memory Computing for Area-/Energy-Efficient Deep Learning." In *IEEE Solid-State Circuit Letters*, volume 3, pp. 194–197, 2020.

[Her09] Suzana Herculano-Houzel. "The human brain in numbers: a linearly scaled-up primate brain." *Frontiers in Human Neuroscience*, **3**(31):1–11, 2009.

[HMA15] Harold Hughes, Patrick McMarr, Michael Alles, Enxia Zhang, Charles Arutt, et al. "Total Ionizing Dose Radiation Effects on 14 nm FinFET and SOI UTBB Technologies." In *2015 IEEE Radiation Effects Data Workshop (REDW)*, pp. 1–6, 2015.

[Hun19] E. Hunt-Schroeder et al. "14nm FinFET 1.5Mb Embedded High-K Charge Trap Transistor One Time Programmable Memory Using Differential Current Sensing." In *2015 IEEE International Reliability Physics Symposium*, volume 1, pp. 233–236, 2019.

[Jan17] SivaChandra Jangam et al. "Latency, Bandwidth and Power Benefits of the SuperCHIPS Integration Scheme." In *Proceedings of 67th IEEE Electronic Components and Packaging Technology (ECTC)*, pp. 86–94, 2017.

[JCE10] Sung Hyun Jo, Ting Chang, Idongesit Ebong, Bhavitavya B. Bhadviya, Pinaki Mazumder, and Wei Lu. "Nanoscale memristor device as synapse in neuromorphic systems." In *Nano Letters*, volume 10, pp. 1297–1301, 2010.

[JLV18] Balaji Jayaraman, Derek Leu, Janakiraman Viraraghavan, Alberto Cestero, Ming Yin, John Golz, et al. "80-kb Logic Embedded High-K Charge Trap Transistor-Based Multi-Time-Programmable Memory With No Added Process Complexity." *IEEE Journal of Solid-State Circuits*, **53**(3):949–960, 2018.

[Jos20] Vinay Joshi et al. "Accurate deep neural network inference using computational phase-change memory." In *Nature Communications*, volume 11, pp. 1–13, 2020.

[JRN20] SivaChandra Jangam, Uneeb Rathore, Sumeet Nagi, Dejan Markovic, and Subramanian S. Iyer. "Demonstration of a Low Latency (<20 ps) Fine-pitch (≤10 μm) Assembly on the Silicon Interconnect Fabric." In *2020 IEEE 70th Electronic Components and Technology Conference (ECTC)*, pp. 1801–1805, 2020.

[JXH21] Chuan-Jia Jhang, Cheng-Xin Xue, Je-Min Hung, Fu-Chun Chang, and Meng-Fan Chang. "Challenges and Trends of SRAM-Based Computing-In-Memory for AI Edge Devices." In *IEEE Transactions on Circuits and Systems I: Regular Papers*, volume 68, pp. 1773–1786, 2021.

[Kha15] Faraz Khan et al. "The Impact of Self-Heating on Charge Trapping in High-k-Metal-Gate nFETs." In *IEEE Electron Device Letters*, volume 37, pp. 88–91, 2015.

[Kha17] Faraz Khan 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." In *IEEE Electron Device Letters*, volume 38, pp. 44–47, 2017.

[Kha20] Faraz Khan. *Charge Trap Transistors (CTT): Turning Logic Transistors into Embedded Non-Volatile Memory for Advanced High-k/Metal Gate CMOS Technologies*. PhD thesis, University of California, Los Angeles, 2020.

[Kim11] Yoon Kim et al. "Integration of 28nm MJT for 8-16Gb level MRAM with full investigation of thermal stability." In *2011 Symposium on VLSI Technology*, pp. 210–211, 2011.

[Kot15] Chandrasekara Kothandaraman et al. "Oxygen vacancy traps in Hi-K/Metal gate technologies and their potential for embedded memory applications." In *2015 IEEE International Reliability Physics Symposium*, pp. MY.2.1–MY.2.4, 2015.

[Kri09] Alex Krizhevsky. "Learning multiple layers of features from tiny images." Technical report, University of Toronto, 2009.

[KRP18] I. Kouznetsov, K. Ramkumar, V. Prabhakar, L. Hinh, , et al. "40 nm Ultralow-Power Charge-Trap Embedded NVM Technology for IoT Applications." In *2018 IEEE International Memory Workshop (IMW)*, pp. 1–4, 2018.

[LCB10] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database." *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, **2**, 2010.

[LeC98] Yan LeCun et al. "Gradient-Based Learning Applied to Document Recognition." *Proceedings of the IEEE*, **86**(11):2278–2324, 1998.

[LHY20] Yandong Luo, Xu Han, Zhilu Ye, Hugh Barnaby, Jae-Sun Seo, and Shimeng Yu. "Array-Level Programming of 3-Bit per Cell Resistive Memory and Its Application for Deep Neural Network Inference." In *Electron Device Letters*, volume 67, pp. 4621–4625, 2020.

[Li21] Haitong Li et al. "SAPIENS: A 64-kb RRAM-Based Non-Volatile Associative Memory for One-Shot Learning and Inference at the Edge." In *IEEE Transactions on Electron Devices*, volume 68, pp. 6637–6643, 2021.

[Lie21] Sean Lie. "The Multi-Million Core, Multi-Wafer AI Cluster." In *Hot Chips 33 (HC33)*, volume 33, 2021.

[Liu20] Qi Liu et al. "A Fully Integrated Analog ReRAM Based 78.4TOPS/W Compute-In-Memory Chip with Fully Parallel MAC Computing." In *International Solid-State Circuits Conference (ISSCC)*, pp. 500–501, 2020.

[LL20] Sung-Tae Lee and Jong-Ho Lee. "Neuromorphic Computing Using NAND Flash Memory Architecture With Pulse Width Modulation Scheme." *Frontiers in Neuroscience*, **14**(571292):1–10, 2020.

[MBL04] Dale McMorrow, Stephen Buchner, William T. Lotshaw, Joseph S. Melinger, Mike Maher, and Mark W. Savage. "Demonstration of Single-Event Effects Induced by Through-Wafer Two-Photon Absorption." *IEEE Transactions on Nuclear Science*, **51**(6):3553–3557, 2004.

[MCB19] Steven Moran, Jonathan Cox, Rachel Brewer, Brian Sierawski, and Subramanian S. Iyer. "Radiation Effects on Brain-Inspired Computing." In *GOMACTech-19, Artificial Intelligence & Cyber Security: Challenges and Opportunities for the Government*, pp. 1–6, 2019.

[McC15] Michael W. McCurdy et al. "Vanderbilt Pelletron - Low Energy Protons and Other Ions for Radiation Effects on Electronics." In *2015 IEEE Radiation Effects Data Workshop (REDW)*, pp. 1–6, 2015.

[McC17] Michael W. A. McCurdy. *"1.8 MeV PROTON RESPONSE OF THERMALLY STABILIZED GALLIUM NITRIDE RF POWER TRANSISTORS."*. Master's thesis, Vanderbilt University, Nashville, TN, 2017.

[Mer14] Paul A. Merolla et al. "A million spiking-neuron integrated circuit with a scalable communication network and interface." *Science*, **345**(6197):668–673, 2014.

[Mic18a] Micron. "3-Dimensional Stack (3DS) DDR4 SDRAM." `https://www.micron.com/-/media/client/global/documents/products/data-sheet/dram/ddr4/16gb_32gb_x4_x8_3ds_ddr4_sdram.pdf`, 2018. Accessed: 2022-02-23.

[Mic18b] Micron. "High Bandwidth Memory with ECC." `https://www.micron.com/-/media/client/global/documents/products/data-sheet/dram/hbm2e/8gb_and_16gb_hbm2e_dram.pdf`, 2018. Accessed: 2022-02-23.

[Mic18c] Micron. "Hybrid Memory Cube – HMC Gen2." `https://www.micron.com/-/media/client/global/documents/products/data-sheet/hmc/gen2/hmc_gen2.pdf`, 2018. Accessed: 2022-02-23.

[Moc18] Reiji Mochida et al. "A 4M Synapses integrated Analog ReRAM based 66.5 TOPS/W Neural-Network Processor with Cell Current Controlled Writing and Flexible Network Architecture." In *2018 IEEE Symposium on VLSI Technology (VLSI)*, pp. 175–176, 2018.

[MW18] Steven Moran, William Whitehead, et al. “Deep learning for medical image segmentation – using the IBM TrueNorth Neurosynaptic System.” In *SPIE Medical Imaging*, volume 10579, pp. 1–8, 2018.

[Nar21] Pritish Narayanan et al. “Fully On-Chip MAC at 14 nm Enabled by Accurate Row-Wise Programming of PCM-Based Weights and Parallel Vector-Transport in Duration-Format.” In *IEEE Transactions on Electron Devices (TED)*, volume 68, pp. 6629–6636, 2021.

[NAS22a] NASA Space Radiation Laboratory (NSRL). “NSRL User Guide Technical Data: Beam Ion Species and Energies.” `https://www.bnl.gov/nsrl/userguide/beam-ion-species-and-energies.php`, 2022. Accessed: 2022-03-08.

[NAS22b] NASA Space Radiation Laboratory (NSRL). “NSRL User Guide Technical Data: LET Range Plots.” `https://www.bnl.gov/nsrl/userguide/let-range-plots.php`, 2022. Accessed: 2022-03-08.

[Neu13] Jens Timo Neumann et al. “Mask effects for high-NA EUV: impact of NA, chief-ray-angle, and reduction ratio.” In *Proceedings of SPIE Advanced Lithography*, pp. 1–14, 2013.

[NI20] Sepideh Nouri and Subramanian S. Iyer. “Non-Volatile Wideband Frequency Tuning of a Ring-Oscillator by Charge Trapping in High-k Gate Dielectric in 22nm CMOS.” *IEEE Electron Device Letters*, **42**:110–113, 2020.

[NR22] Sumeet Singh Nagi, Uneeb Rathore, et al. “A 16nm 785GMACs/J 784-Core Digital Signal Processor Array with a Multilayer Switch Box Interconnect, Assembled as a 2×2 Dielet with 10µm-Pitch Inter-Dielet I/O for Runtime Multi-Program Reconfiguration.” In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 52–53, 2022.

[Ora20] Oracle. “Class Robot.” `https://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html`, 2020. Accessed: 2022-07-12.

[Pas21] PassMark®Software. “Memtest86 Technical Information.” `https://www.memtest86.com/troubleshooting.htm`, 2021. Accessed: 2022-03-07.

[PB61] William Wesley Peterson and David. T. Brown. “Cyclic Codes for Error Detection.” In *Proceedings of IRE*, volume 49, pp. 228–235, 1961.

[PSK13] S. Park, A. Sheri, J. Kim, J. Noh, J. Jang, M. Jeon, B. Lee, B. R. Lee, B. H. Lee, and H. Hwang. “Neuromorphic speech systems using advanced ReRAM-based synapse.” In *2013 IEEE International Electron Devices Meeting (IEDM)*, pp. 25.6.1–25.6.4, 2013.

[RBK12] William R. Reohr, John E. Barth, Toshi Kirihata, D. H. Leu, and Donald W. Plass. "High voltage word line driver.", U.S. Patent 8,120,968, Feb. 2012.

[Roc20] Kamil Rocki et al. "Fast Stencil-Code Computation on a Wafer-Scale Processor." In *arXiv*, pp. 1–12, 2020.

[SA17] Gaurav Singh, Sagheer Ahmad, et al. "Xilinx 16nm Datacenter Device Family with In-Package HBM and CCIX Interconnect." In *Hot Chips 29 (HC29)*, volume 29, 2017.

[Sam17a] Samsung. "288pin Registered DIMM based on 8Gb B-die (2H Stack, 64GB)." `https://www.samsung.com/semiconductor/global.semi/file/resource/2018/04/TSV_DDR4_8Gb_B_die_Registered_DIMM_Rev1.43_May.17.pdf`, 2017. Accessed: 2022-02-22.

[Sam17b] Samsung. "288pin Registered DIMM based on 8Gb B-die (4H Stack, 128GB)." `https://semiconductor.samsung.com/resources/data-sheet/20170731_TSV_128GB_only_DDR4_8Gb_B_die_Registered_DIMM_Rev1.53_Jun.17.pdf`, 2017. Accessed: 2022-02-22.

[Saw16] Jun Sawada et al. "TrueNorth Ecosystem for Brain-Inspired Computing: Scalable Systems, Software, and Applications." In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 130–141, 2016.

[SCL21] Jian-Wei Su, Yen-Chi Chou, Ruhui Liu, Ta-Wei Liu, Pei-Jung Lu, Ping-Chun Wu, Yen-Lin Chung, et al. "A 28nm 384kb 6T-SRAM Computation-in-Memory Macro with 8b Precision for AI Edge Chips." In *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 250–251, 2021.

[SRR08] P Jesper Sjöström, Ede A Rancz, Arnd Roth, and Michael Häusser. "Dendritic excitability and synaptic plasticity." *Physiol Rev.*, **88**(2):769–840, 2008.

[Tak10] Takeshi (Kesh) Ikuma. "GUI automation using a Robot." `https://undocumentedmatlab.com/articles/gui-automation-robot`, 2010. Accessed: 2022-07-12.

[Van21] Vanderbilt University. "VU Laser Facilities Guide.", 2021.

[Vir16] Janakiraman Viraraghavan et al. "80Kb 10ns read cycle logic Embedded High-K charge trap Multi-Time-Programmable Memory scalable to 14nm FIN with no added process complexity." In *IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, pp. 1–2, 2016.

[Wan20a] Zhe Wan. *Scalable and Analog Neuromorphic Computing Systems*. PhD thesis, University of California, Los Angeles, 2020.

[Wan20b] Zhe Wan et al. "Accuracy and Resiliency of Analog Compute-in-Memory Inference Engines." In *arXiv*, pp. 1–21, 2020.

[WM18] William Whitehead, Steven Moran, et al. "A Deep Learning Approach to Spine Segmentation Using a Feed-Forward Chain of Pixel-Wise Convolutional Networks." In *IEEE International Symposium on Biomedical Imaging (ISBI)*, pp. 868–871, 2018.

[WMC19] Zhe Wan, Steven Moran, Jonathan Cox, Xuefeng Gu, Vwani Rowchowdhury, and Subramanian S. Iyer. "Characterization Approaches to Test the Robustness of Neuromorphic Systems." In *GOMACTech-19, Artificial Intelligence & Cyber Security: Challenges and Opportunities for the Government*, pp. 1–6, 2019.

[WMS16] Jiyong Woo, Kibong Moon, Jeonghwan Song, Sangheon Lee, Myounghun Kwak, Jaesung Park, and Hyunsang Hwang. "Improved Synaptic Behavior Under Identical Pulses Using AlO_x/HfO_2 Bilayer RRAM Array for Neuromorphic Systems." In *IEEE Electron Device Letters*, volume 37, pp. 994–997, 2016.

[WTX21] Yin Wang, Hongwei Tang, Yufeng Xie, Xinyu Chen, Shunli Ma, Zhengzong Sun, Qingqing Sun, Lin Chen, Hao Zhu, Jing Wan, Zihan Xu, David Wei Zhang, Peng Zhou, and Wenzhong Bao. "An in-memory computing architecture based on two-dimensional semiconductors for multiply-accumulate operations." *Nature Communications*, **12**(3347):1–8, 2021.

[WWZ22] Zhe Wan, Tianyi Wang, Yiming Zhou, Subramanian S. Iyer, and Vwani P. Roychowdhury. "Accuracy and Resiliency of Analog Compute-in-Memory Inference Engines." *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, **18**(2):1–23, 2022.

[Xia22] T. P. Xiao et al. "An Accurate, Error-Tolerant, and Energy-Efficient Neural Network Inference Engine Based on SONOS Analog Memory." In *IEEE Transactions on Circuits and Systems*, volume 69, pp. 1480–1493, 2022.

[Xil19] Xilinx. "White paper: Supercharge Your AI and Database Applications with Xilinx's HBM-Enabled UltraScale+ Devices Featuring Samsung HBM2." Technical Report WP508, Xilinx, July 2019.

[XNS21] Shanshan Xie, Can Ni, Aseem Sayal, Pulkit Jain, Fatih Hamzaoglu, and Jaydeep P. Kulkarni. "eDRAM-CIM: Compute-In-Memory Design with Reconfigurable Embedded-Dynamic-Memory Array Realizing Adaptive Data Converters and Charge-Domain Computing." In *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 248–249, 2021.

[Xue19] Cheng-Xin Xue et al. "A 1Mb Multibit ReRAM Computing-In-Memory Macro with 14.6ns Parallel MAC Computing Time for CNN Based AI Edge Processors." In *International Solid-State Circuits Conference (ISSCC)*, pp. 388–389, 2019.

[Yan19] Yexin Yan et al. "Efficient Reward-Based Structural Plasticity on a SpiNNaker 2 Prototype." In *IEEE Transactions on Biomedical Circuits and Systems*, volume 13, pp. 579–591, 2019.

[YKC19] Taegeun Yoo, Hyunjoon Kim, Qian Chen, Tony Tae-Hyoung Kim, and Bongjin Kim. "A Logic Compatible 4T Dual Embedded DRAM Array for In-Memory Computation of Deep Neural Networks." In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1–6, 2019.

[ZCX19] Qiwen Zheng, Jiangwei Cui, Liewei Xu, Bingxu Ning, Kai Zhao, Mingjie Shen, Xuefeng Yu, et al. "Total Ionizing Dose Responses of Forward Body Bias Ultra-Thin Body and Buried Oxide FD-SOI Transistors." *IEEE Transactions on Nuclear Science*, **66**(4):702–709, 2019.

[Zha19] Jiawei Zhang. "Basic Neural Units of the Brain: Neurons, Synapses and Action Potential." In *arXiv*, pp. 1–38, 2019.

[ZK17] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks." *arXiv*, pp. 1–15, 2017.

[ZZP18] Xin Zheng, Ryan Zarcone, Dylan Paiton, Joon Sohn, Weier Wan, Bruno Olshausen, and H. S. Philip Wong. "Error-Resilient Analog Image Storage and Compression with Analog-Valued RRAM Arrays: An Adaptive Joint Source-Channel Coding Approach." In *2018 IEEE International Electron Devices Meeting (IEDM)*, pp. 3.5.1–3.5.4, 2018.