



# Enabling Emerging, Heterogeneous Memory Systems

## Citation

Pentecost, Lillian. 2022. Enabling Emerging, Heterogeneous Memory Systems. Doctoral dissertation, Harvard University Graduate School of Arts and Sciences.

## Permanent link

<https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37372073>

## Terms of Use

This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at <http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA>

## Share Your Story

The Harvard community has made this article openly available.  
Please share how this access benefits you. [Submit a story](#).

[Accessibility](#)

HARVARD UNIVERSITY  
Graduate School of Arts and Sciences




DISSERTATION ACCEPTANCE CERTIFICATE

The undersigned, appointed by the

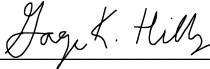
Harvard John A. Paulson School of Engineering and Applied Sciences  
have examined a dissertation entitled:

“Enabling Emerging, Heterogeneous Memory Systems”

presented by: Lillian Coston Pentecost

Signature   
*Typed name:* Professor D. Brooks

Signature   
*Typed name:* Professor G. Wei

Signature   
*Typed name:* Professor G. Hills

April 22, 2022

# Enabling Emerging, Heterogeneous Memory Systems

A DISSERTATION PRESENTED  
BY  
LILLIAN COSTON PENTECOST  
TO  
THE DEPARTMENT OF COMPUTER SCIENCE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY  
IN THE SUBJECT OF  
COMPUTER SCIENCE

HARVARD UNIVERSITY  
CAMBRIDGE, MASSACHUSETTS  
APRIL 2022

©2022 – LILLIAN COSTON PENTECOST  
ALL RIGHTS RESERVED.



## Enabling Emerging, Heterogeneous Memory Systems

### ABSTRACT

Optimizing data storage and data movement remain critical roadblocks to overall computing performance and efficiency. These barriers are due to a convergence of motivating factors that begins with the memory wall and limitations of well-established memory technologies and design paradigms, and is compounded by increasingly data-intensive applications. This shift can and should be answered by many research thrusts to address the density and efficiency of memory systems, including technological heterogeneity in on-chip memory, increased specialization and cross-computing-stack choices in system and device design, and re-thinking the relationship and co-location of memory and compute resources. This thesis proposes and evaluates both specific solutions and broad design methodologies, uncovering enormous potential for increased memory system efficiency while unlocking numerous doors for further exploration and innovation.

# Contents

TITLE PAGE	i
COPYRIGHT	ii
ABSTRACT	iii
TABLE OF CONTENTS	iv
LISTING OF FIGURES	vi
ACKNOWLEDGMENTS	ix
PREVIOUS WORK	xi
o INTRODUCTION	1
o.o Publications & Research Experiences . . . . .	2
o.1 Thesis Contributions . . . . .	4
o.2 Thesis Organization . . . . .	7
1 BACKGROUND: MEMORY TECHNOLOGIES & HOW TO LEVERAGE THEM	9
1.0 Motivation: Limitations of Today’s Memory Systems . . . . .	10
1.1 Motivation: Data-Intensive Applications . . . . .	17
1.2 Embedded, Non-Volatile Memory (eNVM) Technology Landscape . . . . .	22
1.3 Identifying Cross-Stack Design Considerations . . . . .	32
2 RELIABILITY AS A FIRST-ORDER DESIGN CONCERN	35
2.0 Application-Aware Resilience Studies . . . . .	38
2.1 Fault Modeling of Multi-Level-Cell (MLC) eNVMs . . . . .	50
2.2 Customized, Iso-Accuracy MLC storage of DNN weights . . . . .	58
2.3 Fault Tolerance In-the-Loop with Sparsity and Error Mitigation . . . . .	64
2.4 MLC FeFET Memory for DNN Inference and Graph Processing . . . . .	71

3	<b>MAXNVM: MAXIMIZING MEMORY EFFICIENCY FOR ML ACCELERATORS</b>	<b>87</b>
3.0	End-to-End, Co-Design Methodology for MLC eNVM . . . . .	90
3.1	Maximizing Storage Density and Inference Efficiency (Evaluation) . . . . .	93
3.2	MEMTI: Optimizing eNVM for Visual Multi-Task Inference . . . . .	105
3.3	EdgeBERT: Optimizations for Multi-Task NLP Inference . . . . .	115
4	<b>NVMEXPLORER: CROSS-STACK MEMORY DESIGN AND OPTIMIZATION</b>	<b>120</b>
4.0	An Efficient, Extensible Design Space Exploration Framework . . . . .	123
4.1	Supporting DNN Inference Under Varying Operating Conditions . . . . .	138
4.2	Complementing and Accelerating Graph Processing . . . . .	143
4.3	Probing General-Purpose Applications: eNVM as LLC . . . . .	147
4.4	Case studies in eNVM, System, and Application Co-Design . . . . .	150
5	<b>CONCLUSION: FUTURE MEMORY SYSTEM OPPORTUNITIES AND INNOVATIONS</b>	<b>157</b>
5.0	Themes to Unlock Future Memory Efficiency . . . . .	158
5.1	Innovations on the Horizon, and How to Leverage Them . . . . .	162
5.2	Summary: Cross-Computing-Stack Memory Efficiency Opportunities . . . . .	165
	<b>REFERENCES</b>	<b>193</b>

# Listing of figures

1.0	Memory Technology Hierarchy (Schematic) . . . . .	12
1.1	Example Memory Array Organization (Schematic). . . . .	14
1.2	Energy and Latency of Memory Array Access, SRAM (Example). . . . .	15
1.3	Deep Neural Network Layer Schematic (Examples) . . . . .	19
1.4	eNVM Publication Survey (2016-2020) Summary . . . . .	28
1.5	eNVM Projected Array Comparison (2017) . . . . .	31
1.6	Example of Interdependent Design Choices (Schematic). . . . .	33
2.0	Ares Method Schematic . . . . .	41
2.1	Ares Fault Injection Schematic . . . . .	42
2.2	Ares Per-DNN-Model Fault Tolerance Results . . . . .	44
2.3	Resilience of Datatype of DNN Weights . . . . .	47
2.4	Fault Tolerance Across DNN Training Runs . . . . .	49
2.5	Multi-Level-Cell (MLC) CTT Test Chip . . . . .	51
2.6	Multi-Level-Cell (MLC) CTT Test Chip Measurements . . . . .	52
2.7	Multi-Level-Cell (MLC) Programming (Cell-Level Schematic) . . . . .	53
2.8	Multi-Level-Cell (MLC) Programming (Array-Level Schematic) . . . . .	55
2.9	Multi-Level-Cell (MLC) Programming Reliability Impact of the ADC . . . . .	56
2.10	Multi-Level-Cell (MLC) Programming (Schematic) . . . . .	59
2.11	DNN Weight Storage Results, Fixed Point Encoding . . . . .	60
2.12	DNN Weight Storage Results, Clustered Encoding . . . . .	62
2.13	Index Synchronization for Bitmask Error Mitigation . . . . .	67
2.14	Resilience of Sparse-Encoded Data to Faults . . . . .	69
2.15	Optimal Sparse Encoding and Error Mitigation Strategies . . . . .	70
2.16	FeFET Device Diagram and MLC-3 Characterization . . . . .	72
2.17	FeFET Array Architecture Details . . . . .	73
2.18	FeFET MLC Programming Schematic . . . . .	76
2.19	FeFET MLC Programming Details . . . . .	78
2.20	FeFET MLC Programming Device Results . . . . .	79
2.21	FeFET Fault Rate Shmoo Plot . . . . .	81
2.22	FeFET Application Accuracy . . . . .	82

2.23	FeFET Co-Design ALBERT Array Details . . . . .	85
3.0	MaxNVM Methodology . . . . .	91
3.1	MaxNVM Architecture Block Diagrams . . . . .	94
3.2	MaxNVM On-Chip DNN Inference Memory Array Area . . . . .	96
3.3	MaxNVM On-Chip DNN Inference Memory Array Read Energy . . . . .	97
3.4	MaxNVM On-Chip DNN Inference Results, Performance . . . . .	98
3.5	MaxNVM On-Chip DNN Inference Results, Power/Energy . . . . .	99
3.6	MaxNVM Iso-Area Performance Results . . . . .	101
3.7	MaxNVM Intermittent Operation Results . . . . .	103
3.8	MEMTI Architecture Block Diagram . . . . .	109
3.9	MEMTI Accuracy and Energy Results . . . . .	113
3.10	EdgeBERT Design Elements, Optimizations . . . . .	115
3.11	EdgeBERT Memory Optimization Results . . . . .	118
4.0	NVMExplorer Methodology Overview . . . . .	123
4.1	NVMExplorer Example Array Characterization . . . . .	127
4.2	NVMExplorer Data Dashboard Snapshot . . . . .	129
4.3	CHAMPVis Data Dashboard . . . . .	131
4.4	NVMExplorer Array-Level Validation . . . . .	137
4.5	NVMExplorer DNN Array-Level Metrics . . . . .	139
4.6	NVMExplorer DNN Power, Energy-per-Inference . . . . .	140
4.7	NVMExplorer Intermittent DNN Operation . . . . .	142
4.8	NVMExplorer Graph Processing Results Power . . . . .	144
4.9	NVMExplorer Graph Processing Results Latency, Lifetime . . . . .	145
4.10	NVMExplorer LLC Array-Level Metrics . . . . .	148
4.11	NVMExplorer LLC Case Study Power . . . . .	149
4.12	NVMExplorer LLC Case Study Latency, Lifetime . . . . .	150
4.13	NVMExplorer BEOL FeFET Comparison . . . . .	151
4.14	NVMExplorer Area Efficiency Trade-Off . . . . .	153
4.15	NVMExplorer Write Buffer Study . . . . .	154
4.16	NVMExplorer MLC Comparisons . . . . .	156

I DEDICATE THIS WORK TO MY ENDLESSLY SUPPORTIVE FAMILY AND FRIENDS.

# Acknowledgments

I HAVE MANY PEOPLE to thank who have supported me over the past six years and who have helped shape me as a researcher and a scientist.

First, thank you to my advisors, David and Gu, who have been champions of my research and helped me build confidence and perspective throughout my PhD. You've given me space, time, and guidance to explore my many interests in and out of computer architecture and encouraged me in dabbling across research areas and projects, which has made my time as a PhD student immensely enjoyable and made the kind of research I do uniquely possible. It has been a pleasure to work with you both as the research group has grown and changed in exciting ways over the years.

Next, I'd like to thank Professor Gage Hills for sitting on my dissertation committee. It has been such a treat to hear your thoughts on my work and see your success at Harvard, and I appreciate the time and energy you bring to group meetings and research discussions. I'd also like to thank Professors Sasha Rush and James Mickens for serving on my qualification committee and providing feedback and encouragement at a time when I was awash in paper rejections. I've felt very fortunate to be surrounded by knowledgeable and thoughtful researchers throughout my time at Harvard.

Next, I'd like to thank my fellow graduate students and close collaborators at Harvard who have made my work possible and (crucially) kept me engaged, excited, and accountable in project collaborations over the years. To my favorite sounding-board and close friend Udit Gupta, it has been such a joy to grow as researchers and figure out our goals alongside each other. Thanks for being such a steadfast and creative collaborator, delightful co-worker, and, of course, diligent email-proofreader as we have organized events and coordinated projects together over the years. A special thank you also to other mentors I've been fortunate to cross paths with, namely Brandon Reagen for teaching me how to be a grad student and how to have a conversation with David and Gu, and Akshitha Sriraman for sitting me down and asking me to think critically about my research style and my research goals. Thank you to Marco Donato for working closely with me on many, many projects. I continue to learn a ton from you, and I'm very glad to know you as a mentor, a collaborator, and a peer. Thank you also to the many other graduate students and friends who made my time at Harvard productive and fun, to Thierry Tambe for being constantly impressive and generous with time and effort, to Siming Ma for sharing expertise (and data!), to Abdulrahman Mahmoud for refreshing and relatable perspectives and insights, to Brian Plancher for mind-melding about teaching and career goals, and many, many others who I won't list but who have shaped my last six years. I've also

had the pleasure of working with numerous undergraduate researchers over the years, and I continue to be so impressed and refreshed by the perspectives they bring. And, a special thank you to Glenn Holloway for many patient emails and direct support making this research possible.

Thank you to research collaborators across institutions as well, including the many wonderful and insightful folks I've met through the ADA center. A special thank you to my co-first-author buddies on some favorite collaborations over the years, namely Mehdi Sharifi at Notre Dame and Alexander Hankin at Tufts. I'd like to especially thank Alex for all your patience and effort putting together NVMEExplorer. I appreciate your focus and your enthusiasm (the much-needed optimism to my skepticism at times) and look forward to our continued collaboration.

I've been fortunate to do several research internships, from which I've collected skills, great experiences, and awesome mentors. Thank you to the memory systems research team at IBM, to the AI, Silicon, and Performance team at MSR, and to NVIDIA research for incredible exposure to new research ideas, as well as memorable summer experiences. I'd like to especially thank the team of mentors and collaborators who I've grown close with in the architecture research group at NVIDIA – Aamer Jaleel, Po-An Tsai, Angshuman Parashar, Joel Emer, and Bill Dally, who continue to push me to be a better scientist and a more thoughtful researcher through our conversations.

I would not have been nudged towards computer systems research and, more fundamentally, would not be the person I am today if not for the foundation and the incredible mentors I had as an undergraduate in physics and computer science at Colgate University. A special thank you to John Stratton, who took time to teach me how to read a research paper and how to frame a question. Our work together helped me realize the impact that the right mentor at the right time can have on someone's whole trajectory, and I'm very grateful. Thank you also to Ken Segall and to Elodie Fourquet, whose confidence in me as a student and as a scientist bolstered me in difficult times, and who let me wrestle and have real ownership of research projects as an undergraduate.

Thank you to the Bok Center for Teaching and Learning for support, guidance, and collaboration throughout my experience as a pedagogy fellow. My participation in seminars and critical discussions over the years has been so influential and informative as an academic, a researcher, and an educator and the many ways those roles intersect. A special thank you to John Girash, a wonderful and thoughtful mentor who was a real source of constancy and joy working and teaching together throughout an otherwise challenging and isolating year.

A huge thank you to my family, to my parents who have supported me in every way and been loving and patient throughout. Thanks to Joe and Linda for trivia nights and fun dinners, supplying plenty of beer and good times over the years. Thanks also to my close friends who have been such a special part of my life for so long. There are too many more important people to name, but I'll lastly thank my friend Allison for being a constant companion and great ranting-partner over the years of our PhD programs, and my best friend and concert-buddy Emily for our breakfasts and cake-baking and travels and many fun times together.

Finally, I thank my partner, Sean, for every laugh, every cup of coffee, every proofread and debug session, and for making me so happy, inspired, and thankful through these years of hard work. Also, I thank our precious dog, Mipha, for getting me out of the house every day, and for being the most beautiful, energetic, and talented creature on the planet.



# Previous Work

Portions of this dissertation appeared in the following works:

**L. Pentecost\***, A. Hankin\*, M. Donato, M. Hempstead, G. Wei, D. Brooks. “NVMExplorer: A Framework for Cross-Stack Comparisons of Embedded Non-Volatile Memories” IEEE International Symposium on High-Performance Computer Architecture (HPCA) Proceedings, April 2022. *\*Authors contributed equally.*

**L. Pentecost\***, M. Sharif\*, R. Rajaei, A. Kazemi, Q. Lou, G. Wei, D. Brooks, K. Ni, X. Hu, M. Niemier, M. Donato. “Application-Driven Design Exploration for Dense Ferroelectric Embedded Non-Volatile Memories (eNVMs)” ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED 2021), July 2021. *\*Authors contributed equally.*

T. Tambe, C. Hooper, **L. Pentecost**, T. Jia, E. Yang, M. Donato, V. Sanh, P. Whatmough, A. Rush, D. Brooks, G. Wei. “EdgeBERT: Sentence-Level Energy Optimizations for Latency-Aware Multi-Task Natural Language Processing Inference” MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, October 2021.

E. Pyne\*, **L. Pentecost**, U. Gupta, G. Wei, D. Brooks. “Quantifying the impact of data encoding on DNN fault tolerance” Workshop on Performance Analysis of Machine Learning Systems (FastPath at ISPASS 2020), August 2020. *\*Undergraduate Mentee.*

**L. Pentecost\***, U. Gupta\*, E. Ngan\*, G. Wei, D. Brooks, J. Beyer, M. Behrisch. “CHAMPVis: Comparative Hierarchical Analysis of Microarchitectural Performance Visualization” Workshop on Programming and Performance Visualization Tools at Supercomputing 2019 (ProTools at SC19), November 2019. *\*Authors contributed equally.*

M. Donato, **L. Pentecost**, D. Brooks, G. Wei. “MEMTI: Optimizing On-Chip Nonvolatile Storage (eNVM) for Visual Multi-task Inference at the Edge” IEEE Micro, Vol. 39, Issue 6, November 2019.

**L. Pentecost**, M. Donato, B. Reagen, U. Gupta, S. Ma, G. Wei, D. Brooks. “MaxNVM: Maximizing DNN Storage Density and Inference Efficiency with Sparse Encoding and Error Mitigation” 52nd IEEE/ACM International Symposium on Microarchitecture (MICRO-52), October 2019.

B. Reagen, U. Gupta, **L. Pentecost**, G. Wei, and D. Brooks. “Ares: A Framework for Quantifying the Resilience of Deep Neural Networks” Design Automation Conference (DAC) 2018, June 2018.

M. Donato, B. Reagen, **L. Pentecost**, U. Gupta, D. Brooks, G. Wei. “On-Chip Deep Neural Network Storage with Multi-Level eNVM” Design Automation Conference (DAC) 2018, June 2018.

*I was excited at something new, always liked something new, but give credit to everybody who helped. I didn't do anything alone.*

Katherine Johnson, 2017

# 0

## Introduction

DATA MOVEMENT AND DATA STORAGE remain a critical roadblock to overall computing performance and efficiency. Data-intensive workloads, often running on increasingly heterogeneous systems (i.e., computers that comprise more than one type of processing unit or more than just a general-purpose CPU), would particularly benefit from drastic increases to on-chip memory capacity and storage density, as well as the energy efficiency and performance of memory access, as

discussed in detail in Chapter 1.

This thesis identifies and explores game-changing design choices that could enable substantial gains in future memory systems, and also provides far-reaching approaches to identify additional design choices and optimizations. In a variety of application and system settings, my work reveals that exposing and co-optimizing design choices at different levels of the computing stack (for example, algorithmic optimizations alongside architecture and device-level choices) is the bedrock of future efficiency gains and effective future memory system design. For example, several works presented in this dissertation identify embedded, non-volatile memory (eNVM) technologies as a compelling opportunity to address on-chip memory efficiency, but the design space is complex and each memory cell configuration has unique characteristics that can and should be calibrated to system needs. This work has the dual contribution of proposing and evaluating specific memory system solutions, often specifically designing towards optimal on-chip memory for machine learning hardware accelerators, as well as providing reusable and extensible methods, tools, and takeaways to inform a broader class of future memory systems.

## 0.0 PUBLICATIONS & RESEARCH EXPERIENCES

Throughout this dissertation, I will present specific system designs and evaluations and the accompanying methodologies and tools that enable efficient, effective future memory system design and optimization. Much of this work is underpinned and informed by experiences early in my PhD contributing to the design, fabrication, and testing of specialized accelerators for deep neural network (DNN) inference, consistently in tandem with application-level optimizations<sup>243,81</sup>. I was also fortunate to complete research internships in the first half of my PhD that honed deep skills in workload characterization and hardware profiling at IBM (2016) and Microsoft Research (2018). Additionally, I contributed in the nascent stages of MLPerf, a benchmarking effort to standardize

and democratize expectations and conditions for measuring and reporting machine learning performance, in the form of discussions for benchmark selection and responsibilities in benchmark and dataset testing and validation <sup>159</sup>.

Through my development and study of various machine learning algorithms, and particularly in managing the training and optimization of different DNNs, the resilience of DNN parameters and execution to various forms of normalization and value manipulation inspired the development of Ares, an application-level framework for quantifying the resilience of DNNs <sup>193</sup>. Ares proved extensible and expressive for a variety of application and fault-mode pairings, and I collaborated and advised projects leveraging Ares to study resilience of different number representations and data formats <sup>191</sup>, sensitivity to hyperparameters and bit-flip errors during training <sup>191</sup> and DNN resilience to approximate multiplication techniques <sup>186</sup>. Crucially, I also spent time early in my PhD understanding and developing fault models for multi-level-cell (MLC) programmed embedded non-volatile memory technologies. Specific programming choices (e.g., number and spacing of programmed values into a memory cell) could be customized according to the number formats and encoding strategies I was already studying deeply to optimize DNN weight storage <sup>60</sup>, and Ares was once again leveraged and extended to represent and quantify application-level impacts of different DNN parameter storage schemes.

After observing that cross-computing-stack interdependent design factors and sensitivities to cell-level settings were clear determinants of potential memory solution efficiency <sup>60</sup>, it became clear that a system-level evaluation and more comprehensive design study could unlock the huge potential of MLC eNVMs, which led to the MaxNVM project <sup>183</sup>. The building blocks of MaxNVM, in turn, were sufficiently modular and robust, and I applied similar co-design strategies to exciting new application spaces and system contexts (e.g., multi-task DNN inference <sup>59</sup>, graph processing kernels <sup>204</sup>, hybrid SRAM-eNVM memory systems, and natural language processing <sup>226</sup>).

I continually developed and applied these building blocks, including a memory-fault-centric im-

plementation of the PyTorch Ares interface, in new contexts, revealing open research questions and complex design considerations along the way. I realized that the versatility and efficacy of a first-order design guidance tool could be an invaluable framework for a broader set of design studies. This realization spurred the formalization and development of NVMExplorer<sup>185</sup>. My background and experiences in data visualization platforms, including a web-based tool for visualizing microarchitectural performance counters<sup>184</sup>, made it clear that the wide adoption and accessibility of such a design space exploration framework would be boosted by providing the users from different backgrounds and expertise with comprehensive and interactive ways to explore, filter, and refine their memory system design solutions. The resulting framework, NVMExplorer, in turn quickly and effectively spurred additional collaborative efforts in optimizing device and circuit design choices towards application-level and system-level constraints and goals<sup>204,226</sup>.

## 0.1 THESIS CONTRIBUTIONS

Several key themes and takeaways for the broader discipline of memory systems architecture and computer architecture will emerge and recur throughout the discussion of my PhD work:

- **Reliability:** Careful consideration of hardware reliability must be a first-order design consideration in future memory systems, owing both to the unique properties of emerging technology solutions and application-level definitions of program accuracy and resilience.
- **Designing towards heterogeneity:** Specialized hardware components communicating and coordinating to form a single, efficient system presents unique challenges and opportunities to the computer architect and system designer.
- **Flexibility:** In customizing a memory architecture towards emerging technologies, system requirements, and application properties, the extreme efficiency made possible by specializa-

tion must be balanced by a sense of sufficient flexibility and generality to support application changes and broader use-cases.

- **Technology-aware principles and design methodologies:** Numerous emerging technology solutions hold promise for improving the efficiency of computing systems at every scale, particularly in the context of memory solutions, but integrating them effectively will require exposing and balancing their properties with creativity and collaboration.
- **First-order design space exploration to inform and shape further study:** Probing design considerations and quantifying their impacts at many layers of the computing stack is an intensive, complex task, demanding tools and methods to effectively and efficiently explore such trade-offs at a high level to guide further investigation or investment.

Through the course of my PhD, I've unearthed and evaluated a selection of promising architecture solutions garnering appreciable potential benefits and takeaways compared with competitive baseline systems, a few examples of which include:

- Fault-prone, highly dense emerging memory technologies can be effectively employed to store sparse-encoded deep neural network weights for image classification. However, there is a tension between the two: sparse encoding increases fault vulnerability, limiting the efficacy of a fault-prone dense memory. Employing a proposed, lightweight error mitigation scheme with faulty, dense memory and sparse encoding can provide higher storage density than any of the techniques in isolation, providing up to  $29\times$  memory area reduction compared to less-faulty memory solutions<sup>183</sup> (Chapter 2.3).
- Work presented in this thesis also probes the viability of a specific, even-less-mature memory technology solution in multiple application contexts (image classification, natural language processing, social network graph search) and identifies device- and circuit-level opportunities

for innovation and co-design for efficient future memory systems<sup>204</sup>. For example, memory area can be reduced by over  $12\times$  and energy-per-memory-access by  $2.6\times$  relative to a competitive baseline without degrading accuracy for a natural language processing task as a result of customizing memory device properties and programming schemes (Chapter 2.4).

- A proposed and evaluated new memory system for NVDLA<sup>212</sup>, an industry-grade convolutional neural network (CNN) accelerator, used co-designed emerging memory technologies to unlock up to  $3.5\times$  lower energy per image classification task and  $3.2\times$  lower system power, enabling entirely on-chip ResNet50 inference in about  $2\text{mm}^2$  of total system area<sup>183</sup> (Chapter 3.1).
- System-level performance and energy efficiency of embedded, multi-task image processing improves dramatically by integrating a hybrid memory system of both emerging and traditional memory technologies, again made possible by tuning application optimizations and system settings for compression and data format<sup>59</sup>. For example, energy per input image frame reduces by over  $10\times$  while reducing the memory area by about  $3\times$  (Chapter 3.2).
- Applying fault-prone, dense memory technologies towards other applications, such as natural language processing, similarly achieves significant efficiency benefits (e.g., reducing energy-per-input-sentence due to memory access by  $66,000\times$ )<sup>226</sup>, while showing that the intuitions and infrastructure developed in this thesis and used to evaluate emerging technologies can be effectively generalized and empower efficient future studies (Chapter 3.3).

These evaluations are made possible by a selection of software tools, simulation frameworks, and benchmarking efforts I've built or contributed to over the course of my PhD, which are described and cited throughout this thesis. Many of these contributions and evaluation efforts culminated in the development of NVMExplorer<sup>185</sup>, a unified platform to explore the viability of emerging



memory technologies in specific application and system settings. The NVMExplorer approach reveals cross-stack dependencies and optimization opportunities, in addition to reproducing and expanding previous published studies, (e.g., <sup>183</sup>, <sup>84</sup>, <sup>204</sup>). NVMExplorer similarly enables co-design studies of application properties, system constraints, and devices in order to bridge the gap between architects and device designers for future memory solutions. Example co-design studies in Chapter 4 reveal both opportunities and potential disconnects among current research efforts<sup>185</sup>.

## 0.2 THESIS ORGANIZATION

In Chapter 1, I present background on the fundamental costs and limitations of current on-chip memory solutions, as well as introducing the properties and potential of a selection of eNVMs that can be considered to address memory limitations in a variety of computing systems. Next, Chapter 2 introduces reliability as a key design consideration for the next generation of memory systems, describing the tools and modeling efforts I have contributed to throughout my PhD to quantify memory reliability in application-specific contexts and presenting results from several publications that highlight interdependent choices in data format, number representation, circuit- and device-level design choices, and algorithmic optimizations. Each of the studies in Sections 2.2-2.4 apply application-aware reliability analysis to improve memory system efficiency.

Chapter 3 deepens the findings of Chapter 2 by carefully co-designing specific system solutions for deep neural network inference across different application domains, optimization goals, and operating conditions. Together with Chapter 4, Chapter 3 details system evaluations that reveal significant energy/power benefits and increased storage density via comprehensive and technology-aware cross-computing-stack co-design. Chapter 4 goes above and beyond the evaluation of specific solutions by proposing and describing an open-source, modular design space exploration framework (NVMExplorer) to empower future collaborations and effective memory systems research

where application experts and device designers can similarly identify and contextualize different optimizations and system, algorithm, and technology choices. Chapter 5 revisits several key themes and takeaways, and proposes potential future contributions and research directions informed and enabled by the work presented in this dissertation.

# 1

## Background: Memory Technologies & How to Leverage Them

EFFICIENCY AND INNOVATION in future computer systems is increasingly bound by the capabilities of the memory system. The convergence of fundamental scaling limitations, increasing hardware specialization and system heterogeneity, and a broad set of essential computing applications

that are voraciously data-intensive prompt consideration and investment in new memory design strategies and technologies. In this chapter, I motivate the need for fundamental breakthroughs and fresh approaches in improving memory systems, as well as introduce several compelling research directions for addressing the memory needs of modern and future computing systems.

## 1.0 MOTIVATION: LIMITATIONS OF TODAY'S MEMORY SYSTEMS

Even before Moore's Law waned and ended, the slowing of scaling trends has been disproportionate in favoring the efficiency of compute vs. memory<sup>249</sup>. Namely, the improvement in microprocessor speed was observed to exceed the improvement in DRAM memory speed year-over-year. In blunt terms, the inevitable conclusion of this trend is that the fastest, most efficient possible processing unit will result in no system-level performance benefit because it will be stalled, waiting on interactions with memory. The disproportionate scaling in terms of achievable power efficiency, data delivery, and cost-effective density is sometimes referred to as the 'memory wall' problem, and it motivates much need for reinvention and careful consideration of memory system design choices.

The memory subsystem has remained a crucial performance bottleneck in the intervening years of innovation in computing at different scales (e.g., the emerging prominence and diversity of mobile phones and embedded technology, as well as datacenter-scale computing), with critical applications facing limits in terms of both the availability and utilization of memory bandwidth. The quantity of data required by applications often out-paces the power, energy, and physical distance to store sufficient data capacity. As such, there are broad classes of applications that can be identified by whether their end-to-end performance is limited by the availability and utilization of compute resources (**compute-bound**) versus those limited by the availability and utilization of memory resources (**memory-bound**). The identification of compute-bound vs. memory-bound application behavior on a given system is sometimes visualized via the Roofline Model<sup>90</sup>, and analysis is typi-

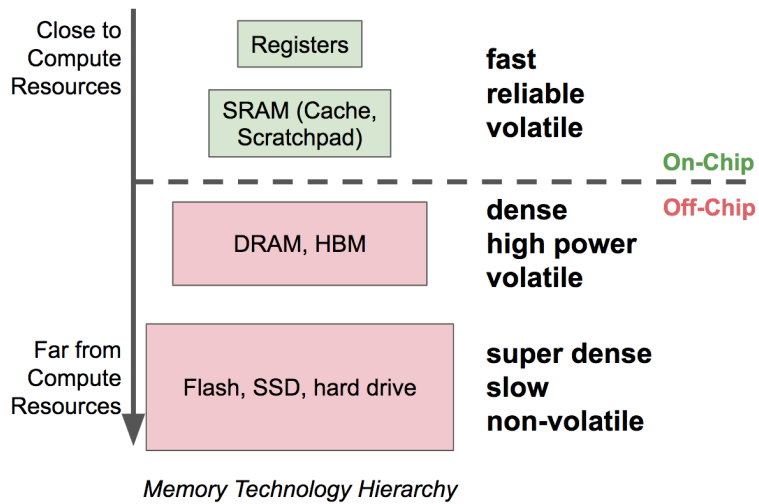
cally conducted via microarchitectural performance analysis (described in Chapter 4.0.5).

In response to the explosive number and variety of data-intensive, often memory-bound applications that are critical in computing today (from facial recognition and image processing on mobile phones, to recommendation systems and graph processing in the datacenter, to a variety of scientific computing applications), this thesis focuses on innovating and intervening in the design of memory systems, with a particular eye towards heterogeneity in two senses: (1) the integration of technological heterogeneity on-chip, that is, emerging and alternative memory device technologies, and (2) customizing memory systems towards hardware accelerators and heterogeneous systems. To understand these trends further and to motivate the solutions and methods put forth in Chapters 2-4, I next provide some discussion and key terms in memory architecture and organization, then dive into the strengths and limitations of today's most pervasive memory technologies (SRAM and DRAM). Then, I describe the area, latency, and energy costs of memory access with a simple example and identify cell-level and memory architecture design trade-offs that will recur in later design studies. Finally, this section will touch upon some compelling research and industry efforts to address the memory wall that are outside the scope of this thesis.

### 1.0.0 SRAM, DRAM, AND THEIR LIMITATIONS

Different memory technologies find use in different parts of a traditional memory hierarchy due to their varying characteristics and varying system requirements. What are the most common memory technologies used in computing systems today, and how are they typically employed in memory hierarchies? Answering these two questions will motivate and inform what the research goals are for augmenting, replacing, or improving these technologies in future designs.

In Figure 1.0, different memory technologies and key terms are organized by their typical relative capacity and proximity to compute units, with some relevant characteristics listed on the right going from top to bottom. In most modern computing systems, the memory resources physically closest



**Figure 1.0:** A majority of modern computing systems rely on SRAM for fast on-chip memory resources physically close to compute resources (green), rely on additional off-chip (red) memory resources with increased density (e.g., DRAM) during computation, and also incorporate higher-capacity, non-volatile memory sources for permanent storage, as described in Section 1.0.0.

to compute resources is implemented using SRAM, and is integrated directly on the same chip as compute resources for fast, efficient data access (shown in green in Figure 1.0). Higher-capacity memory resources are accessed via an interface to another fabricated component (e.g., a DRAM chip or Flash memory), referred to throughout this thesis as “off-chip” (shown in pink/red in Figure 1.0). SRAM and DRAM are volatile technologies, meaning they must constantly receive power to retain stored data values, while Flash is a non-volatile memory technology, meaning it retains data values in memory when powered off. While the relative capacities and protocols applied to each memory technology may vary depending on system context, the broad organization of on-chip SRAM resources for performant memory, off-chip DRAM for density, and a more physically distant, slower, non-volatile resource for permanent storage is visible in mobile phones, laptops, and datacenters alike.

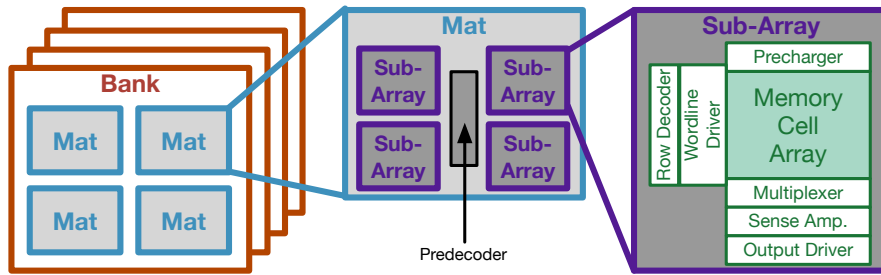
SRAM provides reliable, fast on-chip memory, used as cache memory in a vast majority of mod-

ern systems of varying scales. However, one can only provision so much SRAM on-chip, and there are power and efficiency trade-offs when doing so. When data needs outstrip the capacity of on-chip memory, higher-capacity, off-chip resources repopulate the on-chip resources with the more frequently used or otherwise critical data for use by compute resources. An enormous amount of microarchitectural design effort and system-level innovation is poured into retaining, rejecting, or fetching the right data from off-chip at the right time in order to minimize data movement and/or maximize overall system efficiency due to the costs (energy, time or otherwise) associated with off-chip access. This is true both historically, in continued development of cache protocols for general-purpose systems at varying scales, and in specialized hardware, where energy and power due to data movement remains a key limitation to system efficiency.

The memory wall has already forced system designers to think beyond the memory technology hierarchy in Figure 1.0, and current and future generations of computing systems will require different tools, methods, and technologies to make meaningful breakthroughs. To achieve fundamental future improvements, particularly for applications that are heavily memory-bound and for specialized systems with unique data needs, system designers must bring denser, more efficient memory resources closer to compute resources.

### 1.0.1 MEMORY ORGANIZATION AND MEMORY ARRAY ARCHITECTURE

This section summarizes some fundamental costs of storing and retrieving data. With SRAM properties and limitations as context, we can zoom in on an example of the costs associated with data storage and data movement in a memory array architecture. What are the area, energy, and latency trade-offs when memory devices are organized into subarrays, banks, and arrays? This discussion will introduce architectural design choices and trade-offs that are unique to each proposed physical mechanism for storing bits. As alternative memory technologies for on-chip storage solutions are introduced and evaluated in future chapters, some array architecture characteristics introduced here

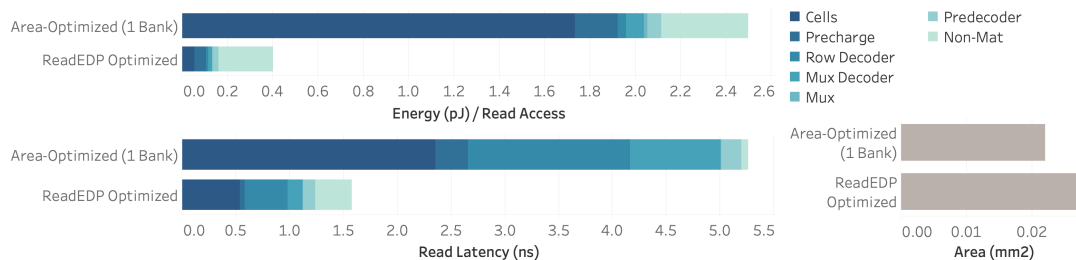


**Figure 1.1:** A memory array is comprised of a set of banks (left), each of which in turn may exhibit a hierarchical organization into mats and sub-arrays. Within each sub-array, there is additional periphery circuitry to read and write data in and out of a set of memory cells (right), adapted from<sup>60,62</sup>.

will be revisited and innovated in the context of optimizations that consider memory-device-level attributes, all the way up to practical, application-level impacts.

There are a variety of architectural choices when organizing memory cells and sub-arrays into larger blocks of accessible memory, some of which are evident in observing a basic memory array organization in Figure 1.1. An array of individual memory devices with corresponding circuitry to read and write data values is labeled as a sub-array, and those sub-arrays are, in turn, hierarchically organized into mats and banks. Large cell arrays can consume significantly more power and take more time to read/write to, so provisioning a block of memory using many smaller sub-arrays with a network of multiplexing and routing to determine when and how to fetch data can reduce the average energy and latency per access. Hierarchical organization choices can also enable simultaneous parallel access to multiple columns of data, sub-arrays, mats, or banks, improving the bandwidth of the overall memory. Sub-array dimensions, column muxing and complexity of selection circuitry, and hierarchical organization into mats and banks are just a sampling of design choices that impact the area efficiency, latency, and energy of access to the memory array. These trade-offs will vary per-technology, and can be tuned according to the system setting and use case. Figure 1.1 shows a schematic example of the memory organization considered by a previously-validated memory array simulation tool<sup>62</sup>, applicable both in the context of an SRAM array and for embedded, non-volatile





**Figure 1.2:** For a small (8KB) SRAM array under two different organization choices (simulated using NVSim<sup>62</sup>), different memory organization choices impact the area efficiency, latency, and energy of access. Organizing the 8KB array as a single bank (Area-optimal) requires less area, while using 16 smaller sub-arrays (the read-energy-delay-product-optimal (ReadEDP) choice) drastically reduces energy and latency of access at the cost of additional periphery circuitry and routing from sub-array to port. The energy and latency per access can be further broken down into the contributions from individual architecture components, as labeled in Figure 1.1.

memory technologies (eNVMs).

Figure 1.2 presents a simple example of how the components of a memory array contribute to the access energy and latency for a simple example under two different architectural organizations. An area-optimal array organization uses a single bank; the resulting read latency and energy-per-access are then dominated by the subarray components. In contrast, a read-energy-delay-product-optimal organization will drastically reduce the latency and energy of access by fracturing the array into multiple banks and subarrays. As a result, the area slightly increases due to duplicated sensing circuitry and network overhead to route data from subarray to port, and non-subarray energy and latency comprises a larger portion of total access costs. Thus, memory architecture choices and optimization goals heavily determine the performance and efficiency of memory resources, and application performance and efficiency, in turn, can be sharply impacted by these choices. The precise trade-offs and optimal memory architectures will vary significantly depending on array capacity and optimization goals. These design considerations are made even more complex by how periphery overheads and cell characteristics differ for alternative, emerging memory technologies discussed in Section 1.2.

### 1.0.2 3-D INTEGRATED MEMORY SOLUTIONS

Another key innovation that has been deployed in several contexts is using the third dimension (e.g., stacking chips on top of one another and connecting them vertically) to improve the effective density and potential bandwidth of memory systems. This is a well-established technique in the context of stacked DRAM using through-silicon-vias (e.g., Hybrid Memory Cube<sup>75</sup>) or the proliferation of High-Bandwidth-Memory (HBM) for increased bandwidth in GPU systems (graphics processing units, e.g.,<sup>158</sup>). The radical increase in effective density and bandwidth provided by 3D-stacked memory solutions has also been applied towards specialized systems for deep neural networks (e.g.,<sup>73</sup>). However, these performant, high-density solutions remain power-intensive, suffer efficiency limitations, and many, being DRAM-based, are volatile memory solutions.

3D-integration techniques have also been demonstrated as compatible with several of the technologies presented in Section 1.2<sup>247,41</sup>. Though the studies presented in this thesis do not consider 3D-integrated memory solutions, typically due to the energy and power constraints of edge and embedded computing contexts considered, there are many compelling memory system design directions left for future work described in Chapter 5.

### 1.0.3 PROCESSING-IN-MEMORY, PROCESSING-NEAR-MEMORY ARCHITECTURES

For several of the technologies leveraged in this thesis, their analog device properties allow them to be used effectively as processing elements<sup>222,50,105,89</sup>, as discussed in Section 1.2. For example, researchers have crafted circuit and array architectures to add or multiple input currents as values, and applied such systems towards neural network inference and training<sup>42,8,201</sup>. Additionally, there is a long history of investigation towards near-data processing (i.e., co-locating compute resources or performing computation in a memory process like DRAM to bridge the distance and unlock efficiency for data-intensive tasks<sup>140</sup>). While these approaches are exciting disruptions of the traditional

memory hierarchy (e.g., Figure 1.0) and have seen application in some targeted settings, they are accompanied by distinct manufacturing and scaling challenges.

Processing-in-memory and near-memory will remain important and exciting paradigms in the future, but the challenges to their scalability, capacity, and density are somewhat orthogonal to the study of emerging materials innovations and devices as memory cells. Though understanding and methodologies for design space exploration at the device and circuit level of emerging memory devices, such as are presented in the following chapters, could be incredibly informative towards in-/near-memory architectures in the future, specific evaluations fall outside the scope of the work presented in this thesis. As the next section will highlight, the size and data-intensiveness of mission-critical applications necessitates sheer density and energy efficiency of on-chip memory and data delivery more broadly. Thus, the scope of this thesis aims to be creative in augmenting and improving memory systems, while keeping them distinct from compute elements, to best complement the enormous diversity of modern heterogeneous computing systems.

## 1.1 MOTIVATION: DATA-INTENSIVE APPLICATIONS

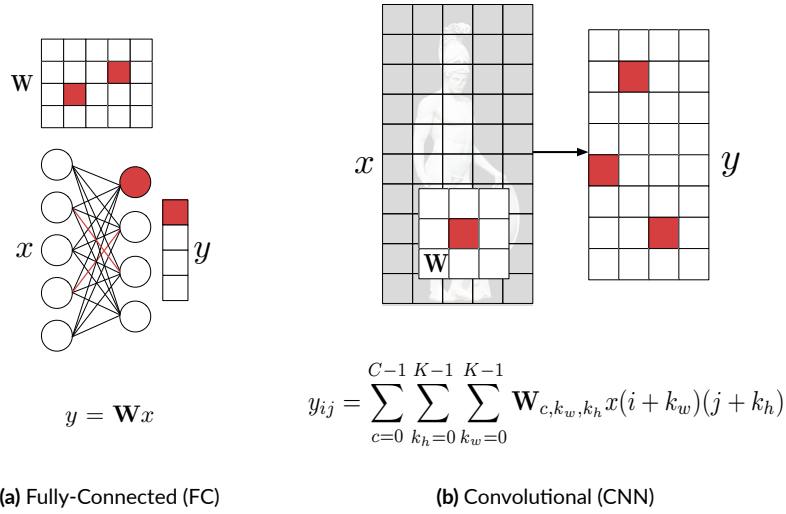
In response to the end of Moore's Law, architects and system designers have gravitated towards specialized hardware solutions and increasingly heterogeneous computing systems in order to achieve the next generation of performance and power advantages over previous systems. The past decade has seen an explosion of investment and interest in specialized hardware and heterogeneous computing platforms. As this trend continues, customized memory system support will continue to be essential. In this Section, I will begin by briefly reviewing examples of heterogeneous systems and the importance of hardware accelerators (Section 1.1.0). Next, I describe the specific needs and ongoing architectural innovations around two broad classes of data-intensive workloads that are the target of studies presented in this dissertation: Deep Neural Networks (DNNs, Section 1.1.1) and

the processing and analytics performed on very large graphs (Section 1.1.2).

### 1.1.0 HARDWARE ACCELERATORS, HETEROGENEOUS SYSTEMS

Specialization allows for significant gains in compute efficiency. Identifying applications that will benefit from specialization and whose acceleration will provide benefits to the overall system setting has become a touchstone of computer architecture research and future system efficiency. For example, datacenter architectures now rely on a huge mix of server-class CPUs, GPUs, FPGAs, other accelerators, and various network and memory interfaces to serve user requests, and high-performance-computing has similarly benefited from the integration of hardware accelerators<sup>87,111,114</sup>. Alternatively, resource-constrained and battery-powered devices (e.g., mobile phones, deployed sensor nodes, drones, self-driving cars) have similarly shifted towards a model of many accelerator blocks and integrated components, for example in a system-on-chip (SoC). In fact, the number of identifiable, specialized IP blocks in Apple iPhone SoCs has increased dramatically over the last decade, from fewer than 10 in the A4 to over 40 in the A11 (iPhone X), comprising a majority of on-chip area<sup>203</sup>. These blocks need to maintain and access data, as well as communicate with one another, and data transfer and data management become even more critical (e.g.,<sup>155,182</sup>).

These specialization efforts at varying scales of computing power and complexity require customizing computing blocks (i.e., accelerators) and end-to-end systems to the needs and characteristics of the most pervasive or otherwise bottlenecked pieces of software that will execute on these systems. The efficiency and effectiveness of specialized hardware accelerator blocks rely on their understanding of application-level properties (i.e., which patterns of computation, memory access, and individual operations to support), as well as sufficient programmability and flexibility (i.e., if the target application changes slightly in terms of size or pattern, is the accelerator still useful?). Thus, cross-computing-stack design considerations and first-order understandings of application-level impacts of system designs become necessary tools in the face of heterogeneous system design and



**Figure 1.3:** Two common types of neural network layers, either fully-connected (left) or convolutional (right) can be summarized as operations on an input ( $x$ ) using trained parameters referred to as weights ( $W$ ) to produce a transformed output ( $y$ ), adapted from <sup>193,192</sup>.

optimization.

The rise of heterogeneous computing platforms is similarly reflected in the sheer volume of academic publications, silicon-based start-ups, and industry investment in hardware accelerators for a range of target applications (indisputably dominated by innovations and investment in machine learning)<sup>192,159</sup>. In these myriad proposals, dense data storage, efficient data movement, and minimizing reliance on off-chip memory resources remain essential challenges. Thus, innovation and improved efficiency in on-chip memory would provide drastic improvement to systems with different target applications, different design constraints, and different scales of operation.

### 1.1.1 DEEP NEURAL NETWORKS

Deep learning is a field of machine learning that leverages large neural networks (often referred to as Deep Neural Networks or DNNs) to tackle challenging classification and regression tasks. In recent years, DNNs have become essential to a huge breadth of application domains, including

image recognition and detection, language processing, and translation. DNNs have two operating modes: training and inference. Training is the process of fitting neural network parameters (weights) to labeled data. Inference is the process of using a previously trained DNN to predict labels for new input data. Figure 1.3 provides highlighted examples of the inputs, weights, and outputs, as well as the equations describing the basic operations of two common types of neural network layers.

Perhaps the most prominent example of the co-development of data-intensive applications and specialized hardware solution has been the seismic rise of acceleration of tensor operations in machine learning algorithms and in DNNs in particular. Machine learning researchers and engineers over the past two decades realized that the computation underlying their algorithms was often deployable and exhibited enormous performance benefits when offloaded to a graphics processing unit (GPU). Where neural-network based solutions were previously impractical due to their intensity and runtime, neural network algorithms trained using GPUs began outstripping state-of-the-art metrics in classification and prediction tasks<sup>197,88</sup>. GPU manufacturers like NVIDIA responded in force by building out support and tailoring GPUs to the needs of such algorithms, leading to a proliferation of GPU-based machine learning acceleration. This increase in popularity, together with the continued proliferation of low-power embedded devices, has similarly motivated the design of DNN-specific hardware accelerators<sup>192,111</sup>. While many energy-efficient DNN hardware implementations have been proposed, a major challenge remains: the large memory requirement to store DNN parameters. Although entirely on-chip or on-device storage would guarantee better inference efficiency, limited on-chip SRAM capacity inevitably leads to reliance on costly off-chip memory accesses to DRAM or cloud-based execution of tasks like speech recognition.

An incredible body of work and many continued research efforts over the past decade have targeted reducing the computational complexity and memory requirements of DNNs. These techniques include innovations in data format (such as reduced precision even down to binarized DNNs), model architectures (such as reducing depth, or removing trained parameters), and training

techniques (such as transfer learning, adaptive techniques)<sup>192</sup>. However, the overarching trend of state-of-the-art proposals is a staggering increase in memory requirements to push the boundaries of machine intelligence, as well as sometimes unforeseeable changes in data flow and data access that complicate deployment to existing, fixed neural network accelerators<sup>126,226</sup>. Thus, the continued development and specialization of systems to support and accelerate DNNs and related, data-intensive applications must balance flexibility in the face of future trends with the persistent need for increased memory capacity and efficiency. Careful consideration of algorithmic advancements and memory-reducing optimizations play a key role in maximizing the effectiveness of system and architecture decisions, as explored in detail in future chapters.

The myriad proposals for reducing DNN size and complexity often degrade the achievable application-level accuracy (e.g., reducing the likelihood of correctly classifying input images by up to 10% while reducing storage requirements by an order of magnitude<sup>94</sup>). While many efforts have considered trading accuracy for efficiency in deep learning systems, the most convincing demonstration of a new technique for a practical system must address and preserve a DNN's baseline model accuracy. One proposal for quantifying and bounding acceptable variation in the inherently stochastic baseline performance of DNN's on a given task is **Iso-Training Noise (ITN)**<sup>192</sup>, which is employed in several studies presented in this dissertation (e.g.,<sup>193,183</sup>). The intuition for this method is that accuracy varies for DNNs repeatedly trained with identical hyperparameters. The resulting variance in the accuracy can be used as a bound for final classification error. As long as model alterations do not result in error exceeding this bound, they are said to be indistinguishable from ITN and therefore iso-accuracy. Examples of ITN for different DNNs and application of ITN as a reasonable bound on acceptable model accuracy are found in Chapters 2.0, 2.3, 2.4, and 3.1.

### 1.1.2 GRAPH PROCESSING

Analytics and processing tasks on very large graph structures underlie a variety of important and emerging applications today, including social network analysis, data mining, computational chemistry, and, perhaps most recently and prominently, COVID-19 drug discovery<sup>17,13,135,262</sup>. There has been a recent rash of interest and exploration of graph processing acceleration and optimization in the systems research community<sup>55,83,265,39,11,54,54,13</sup>. These works span system settings, from FPGA-based acceleration of graph kernels<sup>11</sup> to hardware accelerators<sup>54</sup> to general-purpose cache replacement policies<sup>13</sup>. Each of these works highlights and designs towards the large memory capacity and bandwidth required for efficient execution of graph processing<sup>17,83</sup>.

The challenges of customizing the memory architecture to support graph processing in a variety of system settings are similar to deep neural networks purely in that they are both data-intensive in terms of bandwidth and capacity. Otherwise, at least in broad strokes, the challenges of graph processing are quite distinct. Namely, memory access patterns exhibit much more randomness than the embarrassingly predictable patterns when operating on a fixed neural network model architecture, and graph-related kernels (e.g., search, extracting graph properties) tend to require a more balanced mix of read-write traffic than DNN use cases considered in this thesis. Additionally, the resilience of graph processing to different value manipulations and fault models is less thoroughly studied than for DNNs, as discussed and probed in Chapters 2.4 and 4.2.

### 1.2 EMBEDDED, NON-VOLATILE MEMORY (ENVM) TECHNOLOGY LANDSCAPE

This section briefly reviews the device physics and relevant low-level details of various embeddable, non-volatile memory technologies. Each technology offers unique design challenges and opportunities as a storage device, and are at widely varying stages of development and investment across industry and academic efforts. More detailed surveys and comparisons of subsets of these technologies



are helpful resources (e.g., <sup>31</sup>) in understanding the capabilities and physical mechanisms for each eNVM proposal. The following sections will briefly describe each technology, with an emphasis on both promising and limiting characteristics of each technology. For more established technologies (i.e., RRAM, PCM, and STT), I reference compelling recent examples and mention their broad characteristics, while for more emerging solutions (i.e., FeFET, CTT), I provide some discussion of their physical design and operation. Finally, this section also discusses the eNVM landscape as a whole, including contextualizing the potential benefits and limitations of different technologies and describing multi-level-cell programming.

### 1.2.0 RESISTIVE RAM (RRAM, RERAM)

Resistive RAM (RRAM) encompasses a wide variety of cell-design implementations that all encode data via the variable resistance of a thin layer of material<sup>92,121,190,109,79</sup>, most frequently a metal oxide<sup>254,255,49,128,29</sup>. RRAM is particularly compelling and relatively mature among eNVM proposals, having been demonstrated by major industry fabrication facilities at relatively mature technology nodes<sup>48,77,104</sup>. Previous and contemporary work has also employed RRAM to neural network acceleration<sup>239,95,183</sup>, including leveraging the variable resistance readout for analog compute capabilities in neuromorphic and compute-in-memory architectures<sup>247,235,38,261,162,215,141,42,201</sup>. Such processing-in-memory implementations for deep neural networks are part of a broader class of research probing the analog computation capabilities of RRAM cells<sup>147,252,68,222,36,246,50,149,21</sup>.

There are compelling RRAM solutions using either diode access for crossbars or CMOS access for more traditional memory array architectures. Crossbar arrays offer the best cell area (as low as  $4F^2$ ), but they are subject to higher access times than using a dedicated CMOS transistor to access the RRAM cell<sup>258,120,12,148</sup>. The larger cell area can be overcome by increasing storage density via multi-level-cell (MLC) programming<sup>209,33,251,37,189,146,269,268,53</sup>. RRAM arrays have also been demonstrated in 3D-integrated settings (e.g., 3D-stacked dies of RRAM, vertical integration of

RRAM)<sup>236,153,23,96,139,138,250</sup>. However, RRAM device and array design is still an area of active development and innovation, with constantly evolving studies of reliability and endurance<sup>171</sup>, in addition to alternative cell configurations and array architectures<sup>45,18,263,180,4,154,15,145,150,143,156,241,151</sup>.

### 1.2.1 PHASE-CHANGE MEMORY (PCM, PCRAM)

Phase-Change Memory (PCM), like RRAM, relies on a variable resistance in a material to encode a value. However, PCM relies on the transition between two phases of the target material, namely crystalline (low resistance) or amorphous (high resistance). PCM arrays have been the subject of study and demonstration for a longer period of time than most other eNVMs considered, and have been demonstrated in 20nm-90nm technology nodes<sup>248,47,46,10,218</sup>. PCM similarly requires careful study of reliability, endurance, and cell structure and fabrication variations<sup>40,43,152,118</sup>, and has been recently probed as compatible with 3D-integration<sup>41</sup>, processing-in-memory architectures<sup>74</sup>, and multi-level-cell-programmed implementations<sup>78,119,221,52</sup>.

### 1.2.2 FERROELECTRIC-BASED MEMORY CELLS (FeRAM, FeFET)

While Ferroelectric RAM (FeRAM) is a more mature memory solution with a DRAM-like structure relying on a ferroelectric capacitor<sup>179,259</sup>, FeFET-based eNVM is an emerging, embeddable technology that achieves much higher density by modifying a FET with a layer of ferroelectric material<sup>161,58,174,166,196,175,108,231</sup>. More precisely, the FeFET device is made by replacing the normal gate dielectric in a MOSFET with a ferroelectric layer<sup>108</sup>. For this reason, FeFET devices can be easily integrated in existing CMOS processes, especially when high- $k$  dielectrics such as hafnium oxide can be used as the ferroelectric layer<sup>161</sup>.

FeFET devices can be programmed to different values through partial polarization switching of the ferroelectric layer, and have been demonstrated as CMOS-compatible memory devices down

to 14nm<sup>205,27,227,64,124,131,132,122,70,206</sup>. This is achieved by tuning the portions of the switched ferroelectric domains in a multi-domain FeFET, and this mechanism can be leveraged to program a single FeFET as a multi-level-cell memory device<sup>5,58</sup>, for analog compute capabilities<sup>6,223,168,107</sup>, and 3D integration<sup>69</sup>, though these are recent efforts under active research development. Recent works<sup>5,166,168,64</sup> use FeFETs for ultra-dense, low-leakage, and fast memories.

### 1.2.3 MAGNETORESISTIVE RAM (SPIN-TRANSFER, SPIN-ORBIT TORQUE)

Two popular implementations of magnetoresistive RAM (MRAM)<sup>65,169,76</sup> rely on different physical mechanisms to encode data via the spin of electrons in a magnetic tunneling junction (MTJ) element<sup>214,142,16,110</sup>. Spin-Transfer-Torque (STT) is the more mature of the two, with compelling applications for last-level-cache<sup>170,98,7,229,177,176,106</sup>, other embedded and storage-class demonstrations<sup>133,3,173,57,113,25,51,230,32</sup>, and explorations into their applicability in near-memory-computing contexts<sup>30,105,89</sup>. By contrast, Spin-Orbit Torque (SOT) is an emerging solution with exciting potential for dense, performant storage rivaling STT characteristics and capabilities, though current implementations tend to be less dense and their scalability is not as proven<sup>172,93,71</sup>. Over the past six years, the studies in this thesis have focused on STT as a promising eNVM solution in multiple system contexts, though future work should incorporate SOT more centrally in the design space of eNVM options. STT-RAM is under active development to verify and improve reliability<sup>238,44,72,256,237,217,207</sup> and scalability<sup>164,22,240,129,61,130,134,216,211,198</sup>, as well as probing the possibility of different device and array architectures and MLC-programming via vertically stacked MTJs<sup>208,199,97,228,19,267,266</sup>.

#### 1.2.4 CHARGE-TRAP TRANSISTORS (CTT)

Previous work has shown that a single, standard-sized NMOS device can be used as a cost-effective embedded non-volatile memory cell<sup>117,116,115,165</sup>, including multi-level-cell<sup>157</sup> and process-in-memory<sup>63</sup> capabilities. Data is stored in the resulting memories, often referred to as charge trap transistors (CTTs), by trapping charge in the gate oxide using hot-carrier injection, which alters the threshold voltage of the device. Charge trapping impacting the threshold voltage is itself a well-studied effect typically indicative of device aging<sup>165</sup>.

As a result, a single transistor can be programmed to exhibit different saturation currents, which are read out and decoded as distinct stored values, as discussed in Section 1.2.6 and studied in detail in Chapter 2.1. In this way, CTT-based memory arrays can be fabricated in industrial-grade, cutting-edge standard CMOS technologies with zero added manufacturing cost. In addition to low cost, CTT has many other desirable properties: the memory arrays are analogous to programmable NOR ROM arrays and have similar-scale read latency; devices display very low leakage currents, as stored information is preserved in the transistor's threshold voltage with high retention independent of the applied voltage; and as each cell consists of only a single NMOS transistor, density can be extremely high.

The benefits of CTTs come at the expense of two undesirable properties, limiting their deployment and rendering them ineffective for many general-purpose compute tasks: (1) extremely long write-latencies and (2) potential for high cell fault rates. Writes involve changing the physical device properties, which takes 100ms or more. Write latency is long because CTTs are programmed by iteratively injecting increments of charge and reading until a desired shift is achieved. This is an imperfect process – charge injection is a random process that inevitably leads to a spread in current distribution, and this spread translates to increasing likelihood of misreading the stored value. Work presented in Chapters 2 and 3 demonstrate how CTT may yet be useful for on-chip MLC weight

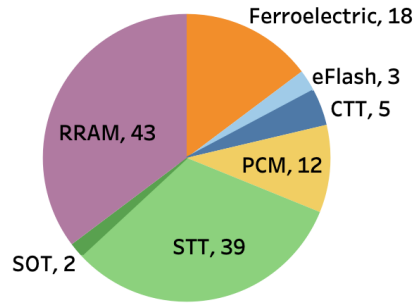
storage for DNN inference accelerators<sup>60,183</sup>.

#### 1.2.5 OTHER MEMORY TECHNOLOGIES (EDRAM, ROM, ETC.)

There are additional alternative memory technologies with varying advantages and limitations. For example, the storage requirements for the DNN architecture for deeply embedded inference, where weights need not be re-written, could be met by read-only memories (ROM) as well. ROMs ensure the best density for storing read-only parameters. However, ROMs also require configuring the network at fabrication time, which makes the design less scalable and cost-effective. One-time programmable (OTP) memories such as anti-fuse, while being amenable to post-fabrication configuration, are far less dense than other memory solutions, even when compared to SRAM.

Embedded DRAM (eDRAM) is another compelling response to the need for dense and performant on-chip memory, and over the past 10-15 years eDRAM has seen deployment in industry products (e.g., the Xbox 360<sup>9</sup>) and, more recently, in accelerator systems for data-intensive applications<sup>35,83</sup>. While eDRAM continues to be developed, optimized, and deployed, issues of technology scaling persist, and the complexity and power-intensiveness of DRAM refresh preclude its use in energy-constrained devices and environments. In contrast, embedded non-volatile memories typically have extremely low leakage and can be powered off without losing data, while still offering increased density, adequate performance, and CMOS-compatibility. In fact, several of the eNVMs discussed require few to no CMOS process changes compared with the complexity and potential fabrication issues of embedding DRAM.

It is also possible to embed traditional flash memory (eFlash), which is provided as a production-grade on-chip memory solution by top industry foundries (e.g., for IoT settings from TSMC<sup>232</sup>) despite known issues and large overheads due to reliability of eFlash technology<sup>210,253</sup>. eFlash has been deployed as a compelling solution for increased on-chip density in IoT and automotive applications, but has several key limitations to wider adoption and future development<sup>144</sup>. Most promi-



**Figure 1.4:** Number of ISSCC, VLSI, and IEDM papers published per eNVM technology, 2016-2020, showing sustained interest in RRAM, STT, and emerging interest in ferroelectric-based devices, taken from <sup>185</sup>.

nently, eFlash has not proved compatible with advanced technology nodes, and it exhibits markedly higher power consumption and typically worse read/write performance than RRAM, MRAM, and other eNVMs considered in this work.

#### 1.2.6 MULTI-LEVEL NVM STORAGE

Storing multiple bits in a single memory cell (multi-level-cell, MLC) is desirable for increasing storage density, and has been demonstrated using many eNVM devices, including resistive random access memories (ReRAM), phase change memories (PCMs), charge-trap transistors (CTT), and FeFETs <sup>31,115,183</sup>. In order to discriminate among the different programmed levels, the memory sensing circuit needs to perform an analog to digital conversion (ADC), decoding each detected level of current or resistance from the memory cell to the corresponding binary word <sup>53</sup>. However, the relatively high area and power overhead of the ADC can limit the relative benefits of MLC implementations compared to single-level-cell (SLC) storage. Thus, MLC NVM storage, while offering exciting opportunities for increased density, introduces performance/power/area trade-offs and may be accompanied by reliability issues, as studied in detail for several technologies in Chapter 2.

### 1.2.7 DISCUSSION: OPPORTUNITIES AND LIMITATIONS

In evaluating the varied landscape of non-volatile memory devices, we are interested in identifying implementations that achieve low read latency, high storage density, and proven ability to scale to advanced process nodes. Figure 1.4 summarizes the number of publications pertaining to each flavor of eNVM over 5 years of the top device and circuit conference venues, including both academic and industry demonstrations and manufactured examples. There is clear variety and active research in the design, optimization, and application of eNVM solutions, but there is a severe imbalance of available data and detail across these technologies according to device maturity, corporate transparency, and the potentially limited scope of demonstration in each published example.

For a sampling of memory-array-level capabilities of eNVMs, Table 1.0 shows several published and validated examples, spanning different technology nodes (90nm to 20nm) and architectures (crossbar vs. CMOS-access), taken from <sup>183</sup>. Several RRAM solutions have been demonstrated in relatively advanced technology nodes, and CMOS-access architectures for embedded applications indicate compelling storage density and read characteristics (i.e., lower area and comparable read latency compared to iso-capacity SRAM). STT strictly improves storage density and read/write performance compared to RRAM and PCM owing to demonstration in a more advanced technology node (28nm) and higher area efficiency despite larger cell footprint. The advantages of PCM are less uniform; though 40nm PCM appears to exhibit lower write latency than similar-capacity RRAM examples, the publication lacks sufficient data to contextualize this with the achievable density and read characteristics. This incomplete context and inconclusive comparison at the device or array level is representative of a key research challenge in trying to identify and evaluate the most relevant and most compelling memory solutions for a particular use case.

To better contextualize and more precisely evaluate the trade-offs among eNVM proposals, Figure 1.5 shows simulated, iso-capacity 4MB memory array characteristics derived from pub-

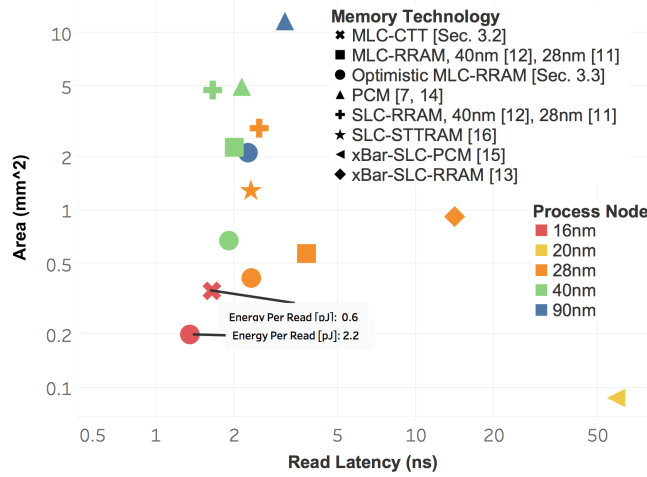
Reference	eNVM type	Tech. Node	Access	Cell Area	Capacity	Array Area	Read latency	Write latency
<sup>29</sup>	RRAM	28nm	CMOS	39 F <sup>2</sup>	1Mb	0.56 mm <sup>2</sup>	6.8ns	500ns - 100 $\mu$ s
<sup>128</sup>	RRAM	40nm	CMOS	53 F <sup>2</sup>	1.4Mb	0.28 mm <sup>2</sup>	10ns	—
<sup>23</sup>	RRAM	24nm	diode	4 F <sup>2</sup>	32Gb	130.7 mm <sup>2</sup>	40 $\mu$ s	230 $\mu$ s
<sup>52</sup>	MLC PCM	90nm	CMOS	25 F <sup>2</sup>	256Mb	120 mm <sup>2</sup>	320ns	—
<sup>47</sup>	PCM	40nm	CMOS	—	1Mb	—	—	120ns
<sup>46</sup>	PCM	20nm PRAM	diode	4 F <sup>2</sup>	8Gb	59.4 mm <sup>2</sup>	120ns	150ns - 100 $\mu$ s
<sup>256</sup>	STT	28nm	CMOS	75 F <sup>2</sup>	1Mb	0.214 mm <sup>2</sup>	2.8ns	20ns

**Table 1.0:** Characterization of several example non-volatile memory arrays circa 2017, adapted from <sup>183</sup>.

lished characteristics, optimized for read energy-delay-product in NVSim <sup>62</sup>. We also include an optimistically-scaled version of a RRAM array based on a 10F<sup>2</sup> cell size as a way of evaluating the maximum potential of promising technology advances, which is shown scaled to several process nodes in Figure 1.5. As demonstrated in proceeding studies (e.g., Section 3.1), compromising on the read latency may not be acceptable for DNNs and other data-intensive use cases, as this could reduce available bandwidth. While the need to accommodate larger access devices requires some area overhead, the arrays implemented using CMOS access devices provide read latencies more in line with potential application requirements. Based on these iso-capacity results, we can identify a subset of eNVM solutions that maximize density and maintain read performance in future evaluations.

To push the boundaries of dense on-chip storage, we are interested in evaluating memories with the capability of storing multiple bits in a single cell using MLC storage. In fact, the projected memory array characterizations shown in Figure 1.5 include several MLC design points. MLCs encode multiple bits of data per cell, which offers maximal storage density with minimal read latency and read energy for the examples shown compared to corresponding SLC (one-bit-per-memory-cell) solutions. However, MLCs come with design challenges (e.g., reliability issues, as studied in Chapter 2) and additional opportunities for optimization (Chapter 3). Both the opportunities and limitations of results in Figure 1.5 lay foundation for the modeling approach that is proposed, expanded, and developed into a more comprehensive design space exploration and evaluation framework in Chapters 3 and 4.





**Figure 1.5:** Characterization of published and extrapolated eNVM proposals comparing area and read latency for a fixed capacity (4MB) using read-energy-delay-product-optimized results from NVSim<sup>62</sup>; section and citation numbers are to references in original publication context, taken from<sup>183</sup>.

Table 1.1 summarizes device characteristics per-eNVM extracted from surveyed publications in 2016-2020 (Figure 1.4). This data, in turn, informs a library of cell characteristics and technology capabilities further developed and leveraged in Chapter 4. For each eNVM listed, we consider a range of published properties of array-level and device-level characterizations. Parameter ranges for a subset of parameters that are particularly relevant to guide system design and determine viability are shown for illustration purposes in Table 1.1.

In Table 1.1, there are several key characteristics that an architect will care about for a memory technology, including the range of demonstrated cell area [ $F^2$ ], demonstrated fabrication process [nm], endurance [write-cycles before degradation], and data retention [s]. It is important to note that the technology classes considered are at different levels of maturity. For example, SOT is a relatively recent technology, and while it boasts very impressive write speed and lower write current compared to STT, published data is extremely limited. We also see that endurance varies by multiple orders of magnitude across different technologies. Thus, adoption will depend on the write

	SRAM	PCM	STT	SOT	RRAM	CTT	FeRAM	FeFET
Cell Area [ $F^2$ ]	146	25-40	14-75	[20]	4-53	1-12	-	4-103
Tech. Node [ $nm$ ]	7-16	28-120	22-90	[1000]	16-130	14-16	40	45
Read Latency [ $ns$ ]	0.5-1.5	[1-100]	1.3-19	1.4-11	3.3-2e3	-	14	-
Write Latency [ $ns$ ]	0.5-1.5	10-3e4	2-200	0.35-17	5-1e5	6e7-2.6e9	14-1e3	0.93-1.3e3
Read Energy [ $pJ$ ]	1.1-2.4	-	0.21-1.2	-	1e-3	-	0.001	-
Write Energy [ $pJ$ ]	1.1-33	-	0.6-4.5	[0.015-8]	0.68	-	-	0.0003-0.01
Endurance [Cycles]	N/A	$10^5-10^{11}$	$10^5-10^{15}$	-	$10^3-10^8$	$10^4$	$10^4-10^{11}$	$10^7-10^{11}$
Retention [s]	N/A	$10^8-10^{10}$	$10^8$	$10^8$	$10^3-10^8$	$10^8$	$10^5-10^8$	-

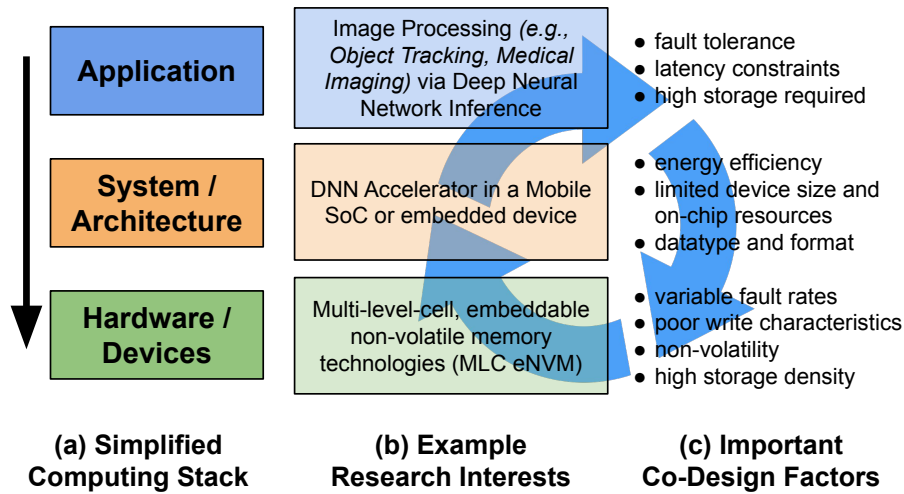
**Table 1.1:** High-level listing of memory cell technologies and ranges for key characteristics; recent publications are complemented by simulation and industry references to form technology cell definitions discussed, indicated by [X], though some emerging technologies have limited available data for certain metrics, indicated by '-', adapted from <sup>185</sup>.

intensity of target applications and system dynamics, so incorporating memory lifetime estimation becomes a critical design consideration.

For some technologies considered, the survey of published examples 2016-2020 provided insufficient data to conduct cell-level characterization, or even to fill in some cells in Table 1.1. Insufficient data could be for reasons of propriety from industry fabrication, waning interest in a given technology resulting in fewer up-to-date publications, or experimental constraints. However, to guide system design, it is critical to have some concept of the possible range of values associated with these parameters. In those cases where SPICE models for a technology are available, we used device and circuit simulations to fill in missing parameters. Alternatively, we consider older publications (dating back to 2009) and consult with device experts to reason about cell and array parameters. This includes identifying either optimistic or pessimistic ranges of achievable cell characteristics to establish reasonable bounds of the design space per technology, as described in Chapter 4.

### 1.3 IDENTIFYING CROSS-STACK DESIGN CONSIDERATIONS

Data-intensive applications such as DNNs and graph analytics have emerged as dominating workloads in the current computing landscape and have influenced many research efforts and advances in computer architecture in the past few years. Specialized hardware architectures deliver outstanding



**Figure 1.6:** The optimizations and opportunities, as well as the design methodologies and frameworks, proposed in the following chapters both expose and exploit the interdependent nature of design choices at the memory device, architecture, system, and application level of the computing stack.

performance and computational efficiency. However, the size and complexity of critical workloads continues to outstrip available on-chip memory, making data movement and efficient on-chip storage a challenge in scaling these applications. Efficient future memory systems must leverage information across the layers of the computing stack – that is, identifying and exploiting opportunities to alter the applications, the system architecture, and the hardware/devices in a collaborative manner to improve end-to-end metrics like energy-efficiency, performance, and accuracy. Design considerations and optimization goals become interdependent across the levels of the computing stack, as pictured in Figure 1.6, highlighting example research interests and important factors for co-design methods that are featured in the remainder of this thesis.

Embedded non-volatile memories offer a promising technology alternative to traditional on-chip SRAM<sup>31</sup>. Smaller memory cell size and the ability to store multiple bits in a single memory cell can translate to a denser array implementation, which can be optimized to store entire DNNs on-chip<sup>183,59</sup>. In addition to storage density improvements, non-volatility increases energy efficient

due to lower leakage power and the ability to retain information in power-off state for intermittent computing. The advantages of eNVMs are often countered by lackluster write performance and reduced reliability, which may be exacerbated with multi-level cell programming.

In today's varied application space, meeting different memory read and write access patterns requires careful consideration of eNVM memory characteristics. An application-driven approach can provide a clear path towards design optimization targeting a specific application and coincidentally offer insight regarding which device-level specifications such as reliability, endurance, write performance, and cell area should be further improved to serve a wider set of applications. Discussions in the following chapters will present both application-driven studies to guide better device development and careful system-level optimizations and evaluations to determine the viability of different memory solutions in specific system contexts.

*“Marilla, isn’t it nice to think that tomorrow is a new day with no mistakes in it yet?”*

*“I’ll warrant you’ll make plenty in it,” said Marilla. “I never saw you beat for making mistakes, Anne.”*

*“Yes, and well I know it,” admitted Anne mournfully. “But have you ever noticed one encouraging thing about me, Marilla? I never make the same mistake twice.”*

*“I don’t know as that’s much benefit when you’re always making new ones.”*

*“Oh, don’t you see, Marilla? There must be a limit to the mistakes one person can make, and when I get to the end of them, then I’ll be through with them. That’s a very comforting thought.”*

L.M. Montgomery, “Anne of Green Gables”, 1908

# 2

## Reliability as a First-Order Design Concern

HARDWARE IS NOT PERFECT, but conventional computing systems typically require near-perfect execution to guarantee correctness. In the shift towards increasingly heterogeneous systems, specialized hardware blocks, and integration of different memory technologies, taking a critical approach to hardware reliability and application resilience can unlock opportunities for increased efficiency. Such a critical approach will require a re-imagining, re-examining, or even retracting of existing

efforts in the deep and complex field of resilience and error correction, as well as a willingness to incorporate the unique fault and failure modes of emerging technology choices in order to extract potential system benefits. In the context of supporting specific classes of workloads, as for hardware accelerators or in computing systems of varying scales, driving resilience studies with a consideration of application-specific metrics, including accuracy, performance, and fault tolerance, is a key advantage of the work presented in this chapter, with a specific emphasis on the co-design of deep neural networks (DNNs) and emerging, embedded non-volatile memory (eNVM) technologies.

Resilience and error correction standards in general-purpose CPUs tend to be brittle and typically require fault probabilities as rare as on the order of  $10^{-15}$ <sup>2</sup>. This level of correctness levies a high design cost at both the device and microarchitecture levels, but has some key advantages. Namely, an operating application can reliably assume perfect accuracy of the hardware, and the user or developer requires no specific awareness or understanding of potential faults or failures. However, relaxing these requirements can enable significant savings, and certain classes of emerging, data-intensive workloads (e.g., deep neural networks) exhibit increased fault tolerance compared to conventional workloads (e.g., one-in-every-thousand values being manipulated or mis-read<sup>194</sup>), providing a plethora of optimization opportunities, particularly in specialized hardware systems.

Ample opportunities to improve performance and efficiency are possible at varying levels of the computing stack if the requirements for absolute correctness can be relaxed. For example, at the architecture level, conventional devices can be pushed to their operational limits by increasing clock frequency to boost performance or reducing  $V_{dd}$  to maximize energy efficiency. Additionally, DNNs require substantial amounts of memory to store weights and intermediate values. While modern high-density storage solutions (i.e., flash memories) require expensive redundancy and error correction hardware, such costly safeguards may be overkill for DNNs. At the device level, many emerging technologies promise high efficiency, but have seen little adoption in general-purpose contexts due to their instability and poor yield. At the fabrication level, design rules may be aggressively

pushed to maximize densities at the expense of reliability. These opportunities motivate us to thoroughly understand the extent of resilience in order to safely push the limits of DNN hardware from architectures to devices.

This chapter first presents and describes a DNN-specific, application-level fault injection framework: *Ares*<sup>193</sup>, including some background on hardware fault and prior work in fault modeling and resilience analysis. *Ares* studies and simulates faults at the application level while executing DNNs directly on GPUs, which is necessary for conducting large-scale fault studies in a timely matter as part of a larger system design process. Traditional fault injection frameworks based on dynamic instrumentation can experience kernel slowdowns of up to  $488 \times$ <sup>86</sup>, making them prohibitively slow to sweep many fault patterns across a range of fault rates varying orders of magnitude. As such, *Ares* can provide accurate, first-order design space exploration that can guide lower-level tools on where to conduct detailed microarchitectural fault analysis. Additionally, Section 2.0.1 will review characterization of DNN fault tolerance, which may be broadly applied across a range of devices and architectures that run DNNs. This is vital because DNNs run on CPUs, GPUs, and accelerators, and the optimal microarchitecture for each DNN hardware device is still widely debated<sup>192</sup>.

Next, Section 2.1 presents fault models and reliability-related characterization efforts across different memory technologies. These fault modeling efforts, in turn, underpin the exploration and careful co-design of specific memory solutions paired with different application characteristics and algorithmic optimizations.

The remainder of this chapter (Sections 2.2-2.4) will present several critical examples of the memory system benefits and fundamental device-application trade-offs that can be uncovered by taking reliability as a first-order design concern. Namely, I will discuss multi-level-cell (MLC) encoding strategies for optimized, dense DNN weight storage using CTT devices (Section 2.2<sup>60</sup>), the intersection of MLC programming choices for both CTT and RRAM with sparse-encoded data and sparse encoding strategies (Section 2.3<sup>183</sup>), and the application of the resulting design methods to

maximize storage density without impacting application accuracy for FeFET-based storage for both natural language processing and graph search workloads (Section 2.4<sup>204</sup>). I also leveraged the intuitions and methodology developed in this chapter towards a collaboration applying approximate matrix multiplication methods to DNN inference and robotics kernels<sup>186</sup>, which is omitted from the following discussion for brevity.

## 2.0 APPLICATION-AWARE RESILIENCE STUDIES

Today, deep neural networks (DNNs) are being deployed at all computing scales—from energy-constrained IoT devices to cost-optimized data centers. Given the complexity and growing scale of DNN deployment, work has mainly focused on optimizing performance and energy efficiency. The initial Ares work<sup>193</sup> explored and quantified the inherent resilience of DNNs to hardware-level faults, opening up new directions for design and optimization strategies.

Designing towards a specific application space (DNNs) allows us to re-examine the (possibly overprovisioned) reliability standards and correction strategies. Alternatively, emerging memory technologies have unique failure modes and fault models by virtue of their unique underlying physics. The application-level impacts of various storage technologies, schemes, and systems must be studied fresh and remain front-of-mind during system design and optimization processes for maximum efficiency.

This section will first discuss a bit of context about hardware faults, and in understanding the potential resilience of DNNs. Next, we present some preliminary results from Ares<sup>193</sup> to build intuition about DNN fault tolerance and the impacts of application-level-resilience-aware design. Additionally, this section will describe ways Ares has been extended and modified to support fault injection during training and under varying number representations and data formats<sup>191</sup>. Studying DNN resilience under more fixed definitions of hardware errors (e.g., bit-flips and power failures)



will prepare us to intersect these intuitions with additional application, system, and device-level optimizations including sparse encodings and varying eNVM solutions.

### 2.0.0 HARDWARE FAULTS

Hardware designers have been forced to consider hardware reliability since the inception of the field. There are many sources of faults in modern semiconductor chips and storage media; manufacturing process variation, voltage noise, and temperature contribute to unpredictable circuit delays that force designers to use worst-case margins. Attempts to minimize these wasteful margins risk the occurrence of timing violation faults under certain operating conditions. SRAM circuits are exposed to large delay variation because they use smaller devices than logic cells, and there are a vast number of cells on each chip, which increases the chance of an outlier. Similarly, heavily-scaled DRAM and flash memory cells now store so few electrons to encode each bit that they are occasionally flipped by common noise events. Cosmic particle strikes are a concern for some applications, as in datacenters, because they result in single-event upsets. Datapaths can also fault when operating conditions introduce excess delay. Another source of fault for both memory and datapaths can be self-induced via approximate circuits. The Ares work focused on static faults in memory, which are pertinent to DNNs due to the large storage requirement for weights and intermediate states<sup>193</sup>.

Faults can be further classified into transient and static varieties. Transient faults come and go over time and are caused by abnormal conditions or events (e.g., resonant supply voltage noise and particle strikes). Static faults persist in the affected device and occur in cases such as “weak” SRAM bit cells due to process variation or flash life-time wear problems. Because static faults persist in time, we see them as a superset of transient faults, and studying their effects first provides a lower bound on DNN fault tolerance. While there are well-known methodologies for identifying and quantifying reliability per-component at an architectural level<sup>167,234</sup> or for application-level studies of architectural resilience<sup>233</sup>, Ares aims to perform fault injection and quantify fault tolerance at an

application-level, using an understanding of fundamental technology fault models.

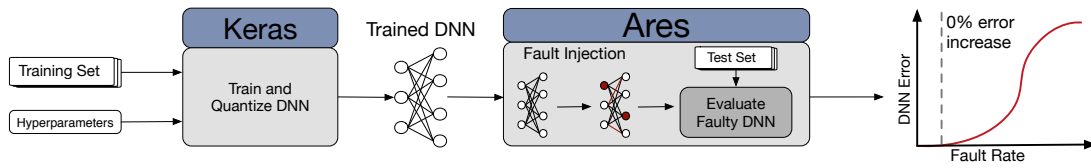
### 2.0.1 ALGORITHMIC RESILIENCE OF DNNs

Previous work suggests DNNs can be robust to faults<sup>192</sup>. For example, eliminating individual nodes or parameters leads to a graceful degradation in model accuracy, as also described in Chapter 1.1.1. Furthermore, regularization and simplification techniques can also be employed to develop, bolster, or otherwise reflect DNN robustness, as DNN models demonstrate correct operation with perturbed parameters. However, these cases are not representative of how faults typically manifest in hardware. Thus, the Ares work intersected some of the most popular DNN optimizations and simplification techniques with bit-flip fault models and error rates to develop a fuller picture of the fault tolerance of DNNs.

The hardware community has also shown interest in understanding DNN fault tolerance. For automotive applications, transient faults can lead to problematic image misclassifications; in order to meet ISO standards, techniques to improve reliability are required<sup>137</sup>. Finally, Minerva<sup>194</sup> proposes fault mitigation techniques to reduce SRAM supply voltage in order to save energy while preserving inference accuracy in fully connected DNNs.

### 2.0.2 ARES: A TOOL FOR QUANTIFYING THE RESILIENCE OF DNNs

We developed Ares to be a fast, scalable fault injection framework that enables rapid fault analysis demonstrated with fully connected (FC), convolutional (CNN), and recurrent architectures such as gated recurrent unit (GRU) based DNNs. We found that model-specific, more-aggressive data types can provide  $10\times$  more resilience, and fault tolerance can vary across models by several orders of magnitude. The basic methodology flow of Ares is depicted in Figure 2.0. Treating training as a large, one-time cost that has been performed in a fault-free environment, we focus on the execu-



**Figure 2.0:** Given a trained and quantized DNN model, Ares can simulate the impact of transient and static fault models on DNN execution, resulting in concrete application-level impacts (e.g., image classification accuracy) under varying fault rates, adapted from <sup>193</sup>.

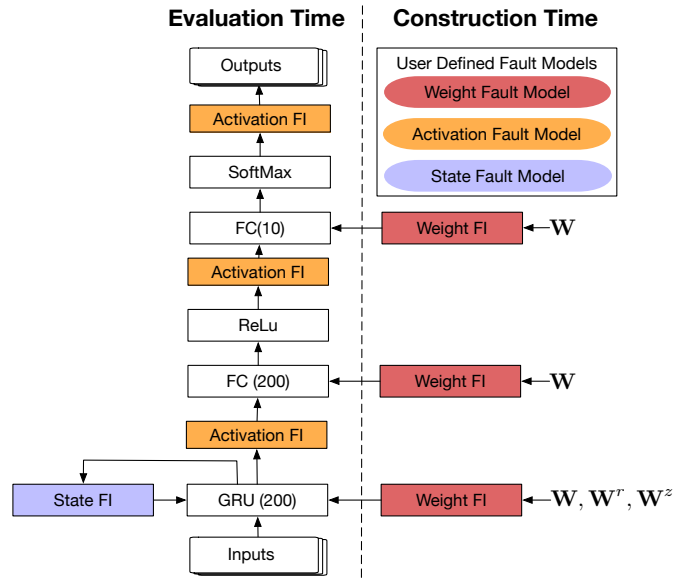
tion of inference. We can characterize the accuracy of trained models by performing inference on a particular set of previously unseen inputs.

The studies presented as part of the demonstration of Ares also revealed optimization opportunities by considering faults at a per-layer granularity. For example, within a DNN, the per-layer fault tolerance can vary by up to  $2781\times$ . We also found that increases in fault rate incur graceful degradation in accuracy for FC and GRU layers, and less so for CNN layers. Additionally, the different types of data of DNNs (i.e., weights, activations, and hidden state) exhibit different fault tolerance behaviors; the activations of a model can be up to  $50\times$  more resilient than the weights.

### 2.0.3 FAULT INJECTION MECHANISMS

Ares establishes baseline classification error by executing inference on a pre-trained DNN with floating point data types for all structures (defined as Iso-Training Noise, as in Chapter 1.1.1). Next, structures are quantized to desired fixed point representation (no accuracy loss is permitted).

Ares has two modes of fault injection: static (for trained weights) and dynamic (for activations). Static faults are injected off-line, before inference is executed. Injecting faults statically is preferred as it introduces no performance overhead during execution. Dynamic faults are injected during the execution of a DNN inference. In Ares, the overheads of dynamic fault injection are minimized by using native tensor operations to emulate fault behavior. So long as the fault model under study can be cast as an element-wise operation or a linear transformation of the state (e.g., random or



**Figure 2.1:** Ares is an application-level fault injection framework for simulating both static and dynamic fault models, incorporated in program execution either during evaluation or off-line during model construction, taken from<sup>193</sup>.

systematic noise), it can be implemented as a tensor operation and run on a GPU. For GRU layers, we inject faults into hidden states at each time step between state updates, as highlighted in Figure 2.1. The maximum slow down from dynamic fault injection is less than  $3.5 \times$ .

Ares performs fault injection at designated points across the weights, activations, and hidden states, depending on the DNN topology and planned experiment. Each fault injection experiment requires the bit error rate (BER) and faulty structures to be specified. Fault patterns are generated by sampling a uniformly random distributed process, which identifies which bits will fault per structure. Ares models faults by using the bit-level fault pattern to mutate stored values. By sweeping fault rates, the user can then analyze the fault tolerance of the DNN.

To further illustrate how Ares performs fault injection, Figure 2.1 lays out the network topology and fault injection points for TiGRU: a three layer DNN with one GRU and two FC layers. Ares is built on top of Keras<sup>1</sup>, with an alternate PyTorch<sup>181,191</sup> implementation with similar scope and

functionality.

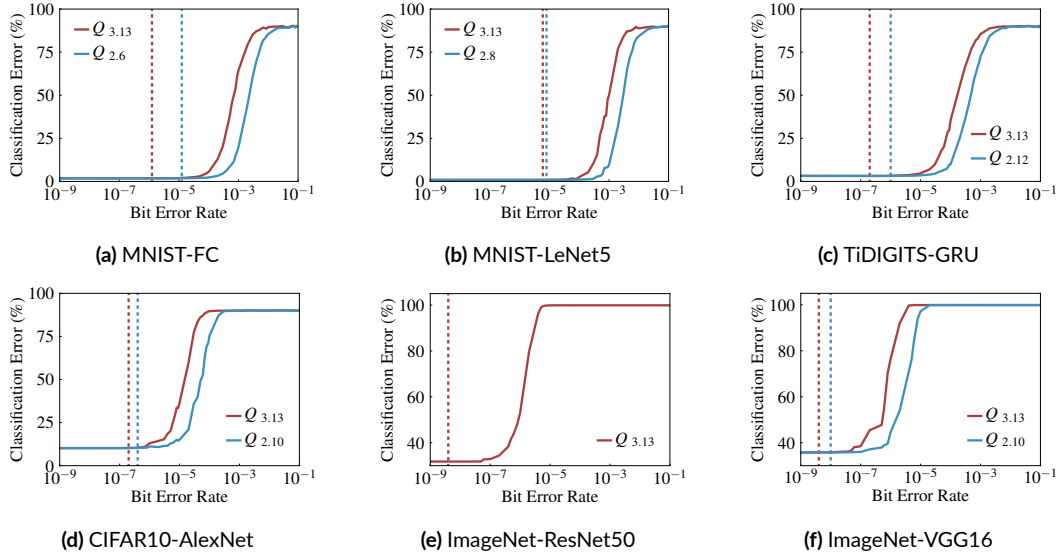
Fault injection is performed at two stages: construction time (static injection) and evaluation time (dynamic injection). Once the DNN is trained, the weights are known, and Ares injects weight faults at construction time by manipulating saved weight values. In contrast, injecting faults into activations requires changes to the program execution. Activation and state fault injection operators are implemented as GPU-compatible element-wise tensor operations. Datapath faults can also occur (e.g., in MAC units), which Ares models by manipulating the data before and after compute operations. The presented results focus on memory faults at all memory fault points, as a precursor for modified and optimized memory system solutions in the proceeding sections. To demonstrate that Ares can accurately capture bit error behavior exhibited by real hardware, we validated simulation results against measurements using a DNN accelerator capable of inducing and measuring SRAM faults<sup>242,194</sup>, as detailed in<sup>193</sup>.

#### 2.0.4 MODELS

Six diverse DNNs are used to evaluate Ares. The datasets used to train these DNNs represent major application domains in which DNNs are commonly used: image classification (*CiFar-10*, *MNIST*, *ImageNet*) and speech classification (*TIDIGITS*)<sup>125,197</sup>. Datasets have varying sizes and models vary greatly in depth and composition. For larger DNNs, we group clusters of similar, adjacent layers into *layer blocks* (LBs) to increase interpretability and ease organization.

#### 2.0.5 SUMMARY OF DNN INFERENCE RESILIENCE TO HARDWARE FAULTS

We use Ares to quantify the relationship between faults and accuracy in DNNs. Results reveal over an order of magnitude variance of fault tolerance across models, between layers, and across structures. Hence, Ares enables exploration of optimizations using different aspects of DNN resilience.



**Figure 2.2:** The tolerable bit-error-rate (BER) before classification accuracy degrades (marked by vertical, dashed lines per model) varies per DNN and per quantization choice, taken from<sup>193</sup>.

We simulate the impact of bit errors at a model granularity by injecting faults across all weights in each model. Figure 2.2 shows the resulting BER-accuracy curves, and the dashed vertical lines indicate the highest tolerable BER without loss of inference accuracy. The two vertical lines show two quantized types for each model: one using only the minimum number of bits required to run inferences without loss of accuracy (blue) and the other (red) a global type that is sufficiently accurate across all DNNs considered.

All models show heavily thresholded accuracy degradation: small BERs have a negligible impact on accuracy up to a certain threshold point. At BERs beyond this point, model accuracy degrades exponentially. The point of 0% accuracy loss is consistently orders of magnitude lower than the knee of this curve. Between these points, the curves typically exhibit a gradual decline in accuracy. Depending on the application, this could be an interesting operating regime in which a loss in accuracy may be exchanged for efficiency gains.

**Resilience varies across DNNs.** There is a large spread in fault sensitivity between DNNs; de-

pending on the model and data type, the knee of the BER-accuracy curve can vary by multiple orders of magnitude. Considering the unified data type for all models, we see that the 0% accuracy loss can vary from  $4 \times 10^{-9}$  (VGG16) to  $6 \times 10^{-6}$  (LeNetCNN). These results suggest protection mechanisms and optimizations used (e.g., ECC and optimal voltage reduction) should be engineered on a per-model basis.

**Quantization impacts resilience.** To improve efficiency, DNNs are quantized to minimize the total number of bits. We study weight fault tolerance with two data types for each network: a unified  $Q_{3.13}$  (i.e., 3 integer and 13 fractional bits) and a per-model optimized type.  $Q_{3.13}$  is the minimal type needed for no loss of accuracy across all models, with ResNet50 requiring the most bits. While a single global type would be needed to build flexible hardware that supports all 6 DNNs, model-specific types are the most aggressive quantization per DNN without increasing baseline error.

Figure 2.2 shows that  $Q_{3.13}$  (red) consistently results in lower fault tolerance than model-specific types (blue). For LeNetFC, the optimized  $Q_{2.6}$  data type is  $10\times$  more fault tolerant. This is because the number of integer bits used (3 versus 2) determines the range of representable values, and models other than ResNet50 can be clipped to just 2 integer bits with no loss in accuracy. The unnecessarily large range of possible values allows for faults of greater magnitude to occur, and hence increases the potential impact of a fault (e.g., flipping the MSB of a near-zero valued weight results in a greater change in value if 3 integer bits are used rather than 2). Flipped superfluous fractional bits do not have as much of an impact on parameter value. By reducing the number of integer bits from 3 to 2, model-specific quantizations consistently demonstrate increased fault tolerance.

**Some image classes are consistently mispredicted.** We found the variance of the distribution of test-case mispredictions (i.e., the number of mispredictions for each class at a sampled fault pattern) is positively correlated with classification error. While fault patterns with high per-class misprediction variance are outliers, it can skew the average classification error. In LeNetCNN, 100 samples at a fault rate of  $1 \times 10^{-3}$  resulted in an average error of 12.2% while the median was only 6.8%.

There are two ways a network can exhibit high class-misprediction variance: one class can constantly be misclassified or half the classes tend to be wrong and half correct. We found the latter case had a greater effect on model classification error. It is worth noting we also found specific fault patterns where a single class was consistently mispredicted.

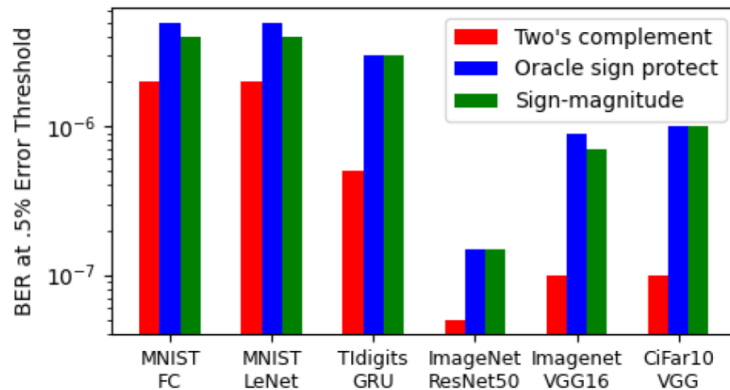
#### 2.0.6 SENSITIVITY TO DATA FORMATS

How weights are represented in memory changes the impact of certain bit flips, potentially affecting the model's vulnerability. In a standard two's complement representation, flipping the sign bit is equivalent to subtracting the largest expressible value for positive weights. DNN weights are clustered near zero, with fewer than one-in-a-million weights in CiFar10-VGG12 larger than 1.5, so sign bit flips are likely to have a large impact on the magnitude of the weight. To determine the extent of this impact, Ares is used to simulate oracle protection to the sign bit of weights, where flips that would occur in the sign bit are silently suppressed. Fault injections are performed on the weights of the six baseline models used in Ares (MNIST FC, MNIST LeNet5, ImageNet VGG16, ImageNet ResNet50, CiFar10 VGG12 and TIDIGITS GRU), with per-model customized quantization at iso-accuracy, as detailed in <sup>193</sup>.

For each model, an instance is trained, then trials are performed across 50 bit error rates (BER) in the range  $[1e-9, 1e-3]$ . For each trial, the test set error after fault injection is recorded. At each BER, 20 trials are averaged to estimate the mean model degradation at that fault rate. Taking the model test error with no fault injections as the baseline, the maximum bit error rate with less than a relative .5% increase in error is calculated and displayed for different NNs per data format in Figure 2.3. Compared to standard two's complement, oracle protection of only the sign bit gives up to an order of magnitude improvement in fault tolerance, as shown in Figure 2.3.

Unfortunately, completely protecting sign bit errors may require architectural tradeoffs, such as storing the bits on a different memory technology or with error correction, which introduces





**Figure 2.3:** Sign-magnitude encoding results in models tolerating up to  $10\times$  higher bit error rates (BERs) than two's complement, nearly matching the effect of oracle protection of the sign bit, taken from <sup>191</sup>.

overhead. This study achieved the benefits of suppressing sign errors without special protection by changing the bit representation of the weights to sign-magnitude (SM). In sign-magnitude encoding, flipping the sign bit transforms  $x$  to  $-x$  while leaving the absolute value unchanged. If  $M$  is the largest representable value, for  $x$  such that  $|x| \leq M/2$ , a sign bit error changes  $x$  by less in sign-magnitude representation than in two's complement. For values close to 0, the effect on weight magnitude of a sign bit error is much larger in two's complement than sign-magnitude. For CiFar-VGG12, which has an approximately normal distribution of weights with mean zero and standard deviation less than .02, for over 99.9% of weights a sign bit flip in 2c will have more than  $10\times$  greater impact on weight magnitude than the equivalent flip in SM. We use Ares to analyze the same set of model instances using sign-magnitude representation. For CiFar10-VGG, ImageNet ResNet50 and TiDigits GRU, sign-magnitude provides equal resilience as oracle protection in terms of tolerable bit error rate (Figure 2.3), achieving all of the benefit without special protection of sign bits. For the MNIST FC and LeNet and ImageNet VGG16 DNNs, the maximum sign-magnitude BER is within 20% of that achieved with oracle protection. Thus, SM representation could allow sign bits to be stored on the same memory as other data, while obtaining the benefit of “protecting” them

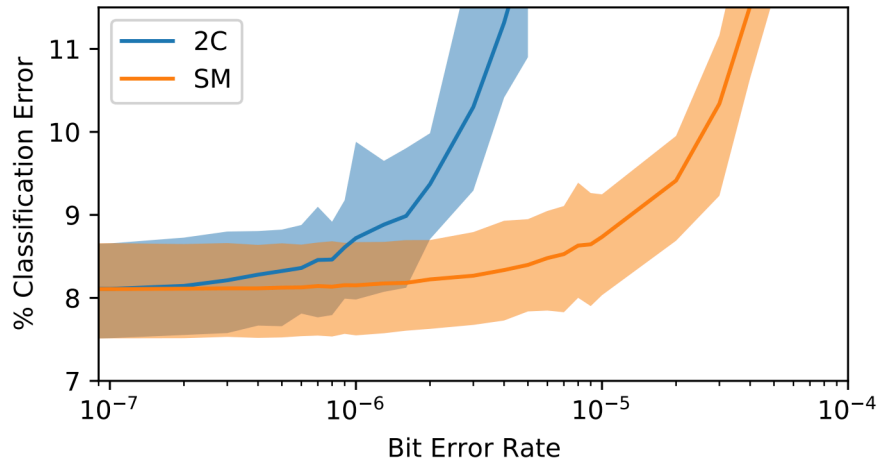
from error. Despite the design and implementation costs, moving to a sign-magnitude encoding for weights could result in  $10\times$  higher acceptable bit error rate while using the same number of bits per weight value.

### 2.0.7 DNN RESILIENCE ACROSS TRAINING RUNS

Implicit in experiments that examine the fault tolerance of a single instance of a model<sup>137,193</sup> is the assumption that identically trained instances have equivalent fault tolerance. However, this assumption does not necessarily hold. Even after isolating a specific dataset, model architecture, and training hyperparameters, we find a wide distribution of fault tolerance. Furthermore, changes to bit representation can affect the variability of resilience to faults (i.e., how predictable the fault tolerance and resulting accuracy is), in addition to the absolute level of fault tolerance. As with variance of hardware characteristics between devices, variance in the fault tolerance of models on a fixed device may result in a requirement to design for the worst case. Thus, studying the variability in fault tolerance has implications for provisioning hardware if a specific model architecture is continually re-trained and re-deployed<sup>87</sup>.

This follow-on study to Ares<sup>191</sup> quantified the variation of fault tolerance across training runs of a VGG12 model on the CiFAR10 dataset, as summarized in Figure 2.4. Each model instance is trained to convergence with identical hyperparameters (e.g., learning rate = 0.1, L2 =  $5e4$ ) and results in classification accuracy well within previously measured bounds<sup>192,193</sup>. For each potential fault rate, as in the previous study (Section 2.0.5), uniformly random bit flips are injected across all weight values, with 50 trials per BER per model instance.

A key finding, made clear in Figure 2.4 is that there is nontrivial variation in fault tolerance across identically-trained models, which should inform future quantification and analysis of DNN reliability and fault tolerance. In fact, if we consider the BERs at which different model instances experience 0.5% error degradation, we observed up to  $4\times$  discrepancy for two's complement number



**Figure 2.4:** Mean and min/max (indicated by shaded region) classification error across 20 instances of VGG12 trained on CiFar10 for varying BERs and different data encodings (two's complement, 2c, and sign-magnitude, SM). SM exhibits slower degradation and lower variance, visible as a smoother curve, taken from <sup>191</sup>.

representation. In more concrete terms, a design target that identified a one-every-million-bits error rate permissible, if applied to another instance of an identically trained model and re-deployed, could actually experience accuracy degradation below acceptable threshold  $4\times$  as often as predicted. For sign-magnitude (SM) encoding, BER at the 0.5% threshold varies by up to  $2.6\times$ . In fact, for a CiFar-VGG instance retrained with identical hyperparameters, assuming equal fault tolerance results in over 5% increase in misclassified images across the test set. Classification error for each model instance initially degrades smoothly below a BER of approximately  $2 \times 10^6$  for SM and  $2 \times 10^{-7}$  for 2c, then rapidly decay (Figure 2.4).

These results suggest that the acceptable BER to minimize impact on classification accuracy differs even among identically trained instances. A given model can be associated with a specific BER above which the model accuracy is unacceptable across all instances, but architects working at lower error tolerances must consider per-instance variation. For situations where models must adapt on device, or where multiple training runs sweeping fault tolerance are cost-prohibitive, sign-magnitude

encoding allows tighter control on the worst case error at a given bit error rate, in addition to lower mean error. More broadly, these results reveal the sensitivity of co-design strategies to the naturally stochastic behavior of DNN training and the important role of bringing re-training, fine-tuning, and other algorithmic choices in-the-loop with reliability analysis and system design choices, as investigated in other studies presented in this dissertation (e.g., Chapters 3.2, 3.3).

## 2.1 FAULT MODELING OF MULTI-LEVEL-CELL (MLC) eNVMS

Next, we discuss the underlying physical behaviors and circuit design considerations that inform the development of fault models for multi-level-cell programming of several compelling eNVM technologies. These fault models, in turn, inform system-level and application-level evaluations of eNVM memory solutions across several projects presented in this dissertation.

One key challenge in MLC storage is that an increase in number of levels is more susceptible to device-to-device variations, making the memory less reliable. In addition, the ADC cost in terms of area and power can limit in some cases the effectiveness of MLC implementations. In order to discriminate among different programmed levels, the memory uses analog to digital converters (ADCs). The ADCs translate each programmed  $I_D$  target level to the corresponding binary word. Individual circuit and device design choices have direct impacts on the resulting reliability of MLC-programmed memory cells and memory array architectures.

### 2.1.0 MULTI-LEVEL CTT CHARACTERIZATION

As discussed in Chapter 1.2.4, the benefits of CTTs come at the expense of two undesirable properties, limiting their deployment and rendering them ineffective for many general-purpose compute tasks: (1) extremely long write-latencies and (2) potential for high cell fault rates. Despite these limitations, several works, including those presented in this thesis, have demonstrated CTT-based mem-

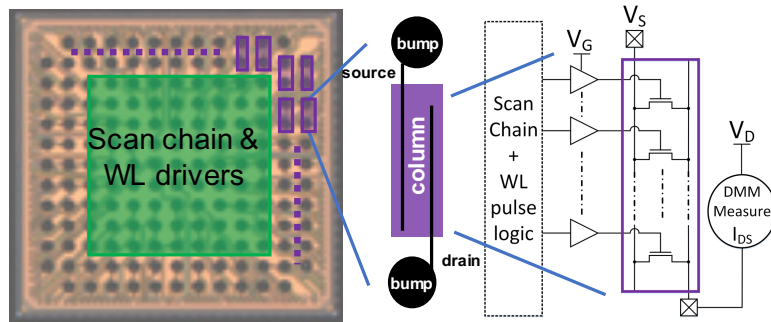
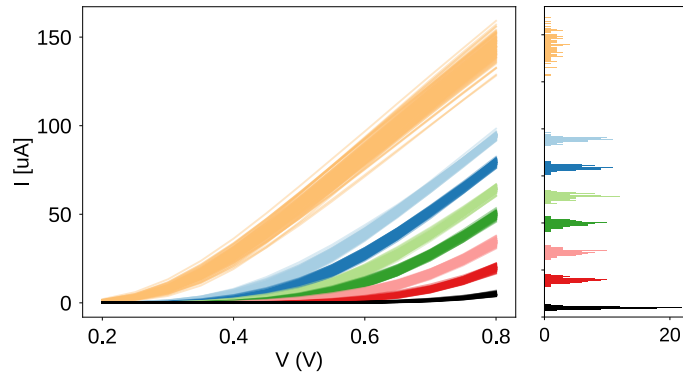


Figure 2.5: Test chip die photo and column schematic for Charge-Trap-Transistors (CTT), adapted from <sup>183,157</sup>.

ories as a near-ideal solution for enabling efficient, embedded DNN inference with careful architectural co-design. This is possible because deeply-embedded DNN weights are updated infrequently and repeatedly used to make inferences, and DNNs are known to be fault tolerant.

Several projects presented in this thesis rely on direct measurements of fabricated MLC-CTT test structures in a commercial 16nm FinFET process <sup>183,157</sup>. A die photo of the chip is shown in Figure 2.5, which contains 36 columns of 128 cells each; internal scan chain and driver circuits mimic wordline drivers. The bumps expose column bitlines to flexibly read and write individual cells via external test equipment. Figure 2.6 (left) shows the distribution of read currents at different wordline supply voltages for 8-level programmed CTTs (a 3-bit MLC). The different colors represents unique levels (i.e., programmed values), and each level is measured from 128 unique devices.

Figure 2.6 (right) shows a slice of the current distributions at the nominal wordline voltage of 0.8V. The yellow cluster corresponds to unprogrammed cells with intrinsic  $V_{th}$  variations inherent to the process technology. Other program levels exhibit tighter distributions due to the iterative write-and-check process. In both cases, the histograms are well approximated using a Gaussian distribution. As more levels are encoded per cell, the current distributions tend to overlap, which increases the probability of misreading the programmed value. This plot highlights two co-design opportunities: first, as more levels are encoded, the current bands tend to overlap, increasing the



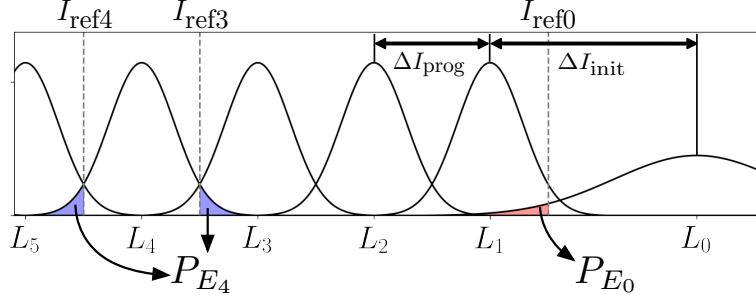
**Figure 2.6:** For programmed MLC-CTT test devices, we measure the distribution in current corresponding to each programmed level across voltage (left) and at fixed 0.8V (right), adapted from <sup>183,157</sup>.

probability of read errors. Given the wider distribution produced by intrinsic process variations, we purposely add separation between the unprogrammed and first programmed state to minimize read error probability of the unprogrammed state.

### 2.1.1 GENERALIZABLE MULTI-LEVEL-CELL FAULT MODELS

In developing a fault model for eNVM implementations, we focus on two primary sources of uncertainty: (1) the intrinsic randomness associated with the process of setting the value of different programmed levels in a memory cell, and (2) the effect of mismatches in the sensing circuitry.

A fault in an MLC is defined as a device being incorrectly read as a level adjacent to the intended one. The probability of a fault occurring for a given level,  $L_n$ , is determined by two reference thresholds:  $I_{\text{ref } n}$  and  $I_{\text{ref } n-1}$ ; reference thresholds discretize current ranges of the cell. Figure 2.7 shows the current level distributions for a generic MLC.  $\Delta I_{\text{init}}$  represents the distance between the mean value of the initial state,  $L_0$ , and the mean value of the first programmed state,  $L_1$ . The remaining programmed levels are equally spaced by  $\Delta I_{\text{prog}}$ . These deltas are tunable and determine the fault rate between levels, which introduces co-design opportunities.



**Figure 2.7:** For a generic multi-level-programmed cell (MLC), each level can be represented as a gaussian, and the probability of mis-reading a programmed value corresponds to the overlap or intersection of adjacent programming distributions, labeled as  $P_E$ , adapted from <sup>60</sup>.

Figure 2.7 highlights the probabilities of two possible cell faults:  $P_{E_0}$  and  $P_{E_4}$ . We first consider the case of a fault in a cell set to the initial state,  $L_0$ , to be incorrectly read as a cell in the first programmed state,  $L_1$ . The probability of this fault ( $P_{E_0}$ ) occurring is given by the total probability of the initial state's distribution beyond the reference current  $I_{\text{ref}0}$ .

$$P_{E_0} = P(I_{\text{cell}} < I_{\text{ref}0}) = \int_{-\infty}^{I_{\text{ref}0}} P_{L_0}(x) dx \quad (2.1)$$

In the case of a cell programmed to a specific level (i.e., not in the initial state), the cell could fail by being read as either of the two adjacent levels. In Figure 2.7, this is illustrated as  $P_{E_4}$ . Thus, the probability of a fault is given by the sum of the probabilities of having  $L_4$  fall in either of the erroneous ranges:

$$P_{E_4} = [1 - P(I_{\text{cell}} < I_{\text{ref}3})] + P(I_{\text{cell}} < I_{\text{ref}4}) \quad (2.2)$$

Given  $\Delta I_{\text{init}}$  and the number of levels per cell, a detailed fault model for CTT-MLC is constructed.  $\Delta I_{\text{prog}}$  results from partitioning the remaining  $\Delta I$  after  $\Delta I_{\text{init}}$  is allocated. This allows us to make  $P_{E_0}$  arbitrarily small in order to protect the initial state at the cost of increased fault rates in the programmed levels. The effects of current to voltage conversion on the distribution must also

be taken into account.

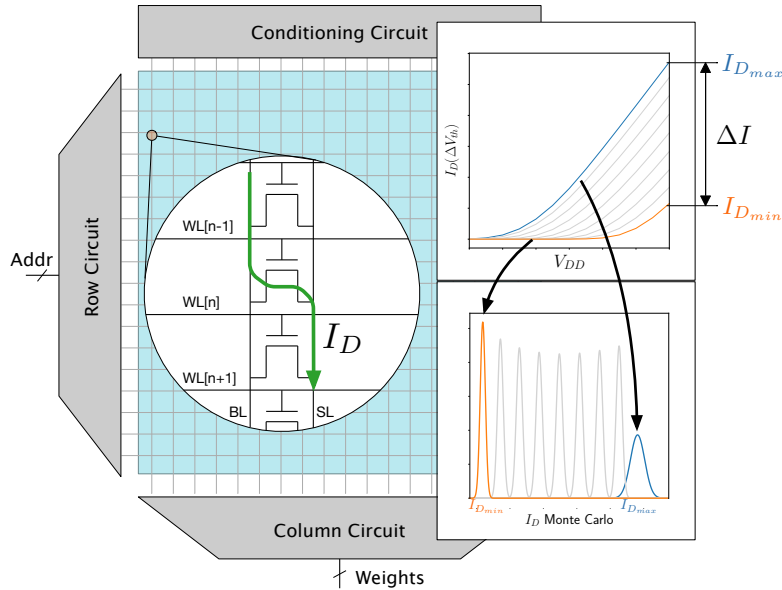
The intrinsic distributions for the stored levels in a RRAM implementations are extracted from published data for 3-bit MLC programming<sup>268</sup>. For MLC-CTT, inter-level fault rates are determined by directly measuring the current distributions from our test chip, as shown in Figure 2.6. For both technologies, current distributions can be modeled as gaussian distributions, and the overlap of level distributions determines the rate at which that value will be misread as an adjacent programmed level. In both cases, we use SPICE simulations to derive the distributions at the output of the current-to-voltage converter. When packing many levels into a single cell, the likelihood of misreading a programmed level can be high in both CTT and RRAM (e.g., fault rates for MLC-3 range from  $10^{-3}$  to  $10^{-5}$ ). However, errors typically result in reads to an adjacent level\*. By carefully arranging how data is encoded into MLCs and exploring the impact of varying number of bits per cell on DNN classification error, the effects of faults can be mitigated (Section 2.3). For each cell design, we complement measured or extracted characteristics with SPICE simulations to determine equivalent current or resistance and read voltage descriptions for each programmed level<sup>60,183,59</sup>.

The design and optimization of sensing circuitry also determines the fault rate of a multi-level-programmed memory cell for the eNVMs discussed. The sensing circuitry must perform the analog-to-digital conversion from the measured cell current to the binary-valued memory array output, as summarized in Figure 2.8. More precisely, an  $n$ -bit MLC outputs a current level of  $I_i$  in the read mode when programmed to level  $i$ . The sense amplifier (SA) circuit detects the level (one of  $2^n$ ) of the cell from  $I_i$  and outputs an  $n$ -bit signal. The designed SA circuit amplifies  $I_i$ , compares it to some quantized level, and concludes the saved level delivering an  $n$ -bit output. The analog-to-digital converter (ADC) employed by the SA circuit is similar to a flash-type ADC and has  $2^n - 1$  quantized levels for an  $n$ -bit MLC. The SA circuit, in turn, suffers from D2D variation and affects the reliability of the memory.

---

\*Misread probability of non-adjacent level is  $1.5 \times 10^{-10}$  or below.

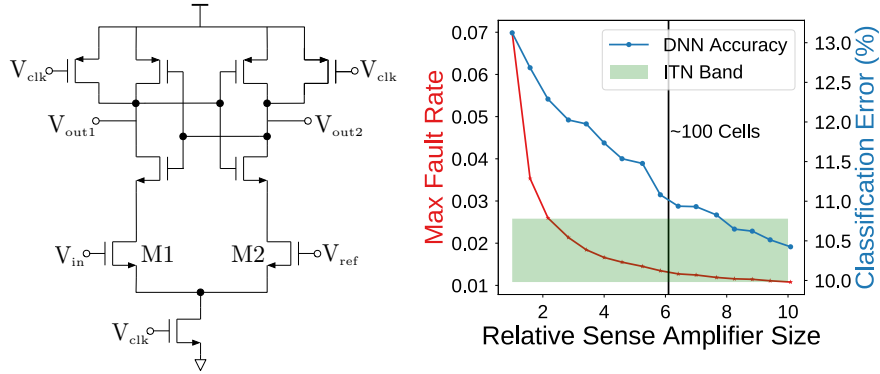




**Figure 2.8:** At a memory array level, the programmed level per MLC device correlates to a current value in a fixed, programmable range, and is directed to sensing circuitry for read-out, adapted from <sup>60</sup>.

Reading back the stored value requires converting the programmed analog level to a binary word, and can be done using parallel sensing or sequential sensing schemes. Parallel sensing is similar to using a flash ADC, and requires each bitline to have dedicated sense amplifiers for each possible stored level. Sequential sensing uses a single sense amplifier and recovers the stored binary word iteratively for each bit. While sequential sensing reduces the overall number of sense amplifiers, we noticed that implementing parallel sensing with small sense amplifiers does not incur an excessive area penalty.

Several of the works presented in this thesis focus on a specific sense amplifier (SA) design, which is characterized by having low static power and input referred offset primarily determined by the input differential pair of transistors (Figure 2.9, left). We characterize the input offset voltage for a latch-based sense amplifier <sup>33</sup> through Monte Carlo SPICE simulations. For these simulations, we swept the width of the input transistors as they are primarily responsible for setting the input offset voltage. These results allow us to quantify the resulting inter-level fault rate per device. For example,



**Figure 2.9:** Example schematic for a sensing circuit for multi-level programming (left), highlighting the relative size of M1/M2 as a determinant of fault rate, and simulated fault rate results for 3-bit MLC programming including ADC (right), plus application-level impact on ResNet/CiFar10 image classification error, adapted from <sup>183</sup>.

in Figure 2.9, right, varying the relative sense amplifier size has a direct impact on both the raw fault rate and the resulting impact on classification accuracy for a workload of interest (CiFAR-10 image classification using a ResNet model).

Based on these simulations, we choose a SA size such that the overhead incurred due to SA choice for the overall array in our final results never exceeds 1%, and the inherent inter-level MLC fault rates are altered by less than  $2\times$ . For the readout architecture, we consider a parallel sensing scheme, whose operation is similar to a flash ADC: the bitline is connected to a number of SAs equal to  $N - 1$ , where  $N$  is the number of levels that can be stored per cell, with each SA connected to the appropriate reference voltage. A parallel sensing scheme can decode the stored value in a single conversion step. However, the number of required SAs increases exponentially with the number of stored bits. This overhead is mitigated by multiplexing the memory array columns. The resulting design is integrated with NVSim<sup>62</sup> to characterize the overall memory architecture.

### 2.1.1.2 ARES FRAMEWORK EXTENSIONS

Several projects presented in this thesis extended Ares to model MLC eNVM faults, accommodate sparse-encoded values, and simulate error mitigation strategies. Based on the results from Section 2.1.1, we generated a inter-level fault probability map for each MLC configuration (either CTT or RRAM). The baseline fault probability map is modified to include the effects of the sensing circuitry by shifting the mean of each level distribution by an amount sampled according to the characterized input referred offset.

Fault injection is performed by first converting the weight values into an MLC representation. Then, for each eNVM cell, we sample a gaussian random variable from the appropriate level distribution and check if the result crosses the thresholds which are used to sense the cell's content. If they do, a fault has occurred and the faulty memory cell's value is updated to the value represented by the adjacent level in Section 2.3. After determining fault locations, the modified values are used to perform inference across the entire test dataset. Experiments are repeated over many trials, and presented results are averaged over many unique generated fault maps.

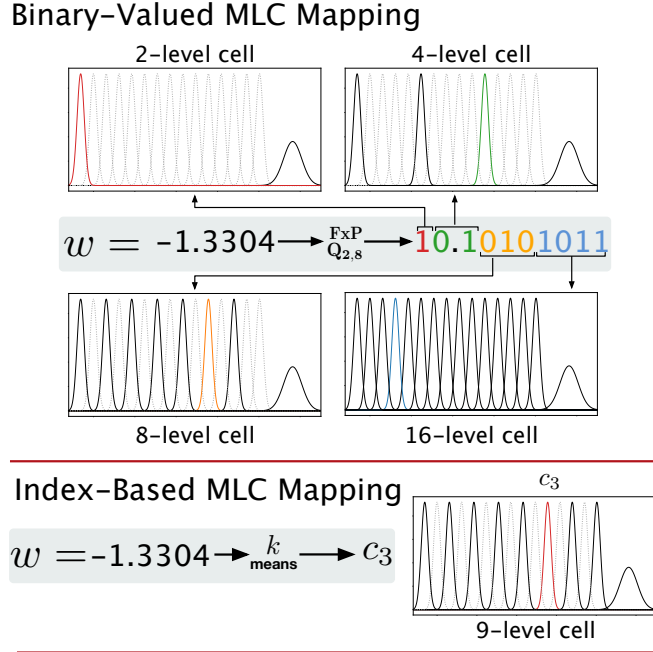
Sparse encodings require separate fault injections on each structure (e.g., the bitmask and non-zero weight values), and we vary the number of bits per cell used to store each structure. When implementing support for ECC, we must ensure a level-to-level fault correlates to a correctable (single bit flip) error by using Gray coding to store binarized values in the MLCs. Dynamic error correction and mitigation strategies are integrated such that faults are detected and values are updated as appropriate prior to evaluating model accuracy. Ares enables the evaluation of DNN classification error over many randomly seeded trials for various MLC eNVM configurations and encoding strategies. From this analysis, we definitively determine the optimal number of bits stored per MLC and the minimal number of memory cells required for each encoding strategy for each DNN such that there is no loss in accuracy.

## 2.2 CUSTOMIZED, ISO-ACCURACY MLC STORAGE OF DNN WEIGHTS

This study describes how CTT-MLCs can be leveraged to eliminate off-chip neural network (NN) weight accesses<sup>60</sup>. To address the faults incurred from storing multiple bits per device, these experiments leverage the fault model and fault infrastructure presented in Section 2.1 to vary how many cells and levels per cell to use for fixed-point weight representations. To optimize this implementation, we co-design DNN weights and CTT-MLC device properties by: (i) clustering the weight values to require fewer memory cells, (ii) using non-sequential level encodings to mitigate the effects of faults, and (iii) pruning the network to skew weight values and leverage the non-uniformity of fault probability in CTT-MLCs. While this section focuses on CTT-based eNVMs, this approach is deepened and applied to other MLC eNVM technologies in Sections 2.2 and 2.4 and again in Chapter 3.

### 2.2.0 DATA FORMATS

This study considers two types of weight quantization: fixed-point and clustering. **Fixed-point** data type quantization is a well-known and effective hardware optimization technique; reducing the width of data types can substantially reduce area and power dissipation while improving performance. In NNs, fixed-point quantization can be aggressively applied to weights. The 32-bit floating point types used for training weights can be reduced to use only 8-12 bits for inference without compromising accuracy. The second weight quantization approach we consider is ***k*-means clustering**<sup>192</sup>, in which NN weights are mapped onto a set of  $k$  values on a per-layer basis. Clustering is advantageous as only the indexes need to be stored per weight and  $\lceil \log k \rceil$  is typically significantly less than the number of bits required with fixed-point quantization. The overhead for storing the look-up table of the actual  $k$  weight values is negligible (e.g., 4-16 8bit values per NN layer).

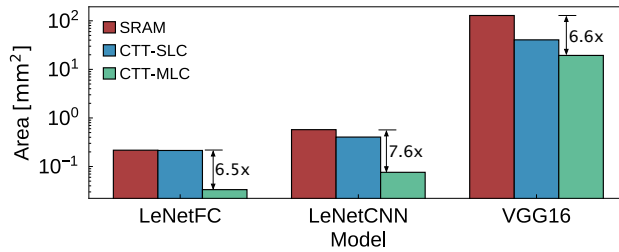


**Figure 2.10:** In assigning DNN weight values to MLC programmed levels, data format and programming choices will determine reliability of storage; mis-read MSBs may have outside impact on application accuracy and may be selectively stored reliably in 2-level cells, or programming settings can be customized for clustered DNN values corresponding to non-power-of-two number of programmed levels, adapted from<sup>60</sup>.

### 2.2.1 DATA ENCODINGS TO MLC STORAGE

As an example, consider the conversion of the value  $(-1.3304)_{10}$  and its fixed-point representation  $(10.10101011)_2$  using two integer bits and eight fractional bits. If this value is stored using binary encoding with CTT-SLC, then 10 cells are needed (one per bit). The same value can be stored using only 3 CTT-MLCs if the most dense, 16 level (4 bits), cell configuration is used. This uniform configuration reduces the area cost per bit at the expense of a higher fault rate. Since faults on higher order bits have a disproportionate impact on the stored value, we anticipate that it is advantageous to selectively mitigate these faults.

Figure 2.10 shows an alternative encoding to MLCs better tuned to both memory cell properties and DNN fault tolerance: a non-uniform encoding where 4 cells are used with 2, 4, 8, and 16 levels



**Figure 2.11:** Area reductions for fixed-point NN weights, with no loss in DNN accuracy, for MLC-CTT storage vs. on-chip SRAM, adapted from <sup>60</sup>.

per cell, from most-significant-bit (MSB) to least-significant-bit (LSB). The example shows that the sign and integer portion of the weight can be protected by using a CTT-MLC with fewer levels. Compared to uniform encoding, this non-uniform encoding requires only one additional cell and substantially reduces the fault rates in weight MSBs.

Storing cluster indexes (Figure 2.8, bottom) enables two optimization opportunities. The first technique is an intra-cell optimization to protect the initial state from faults by increasing  $\Delta I_{init}$ . This technique is particularly effective when the majority of the parameters are assigned to a single cluster, a property we actively enforce with weight pruning. We further consider distinct ways to map clusters to levels to mitigate the magnitude and fault rates, varying the relative level-to-level fault rates of the most common cluster indexes vs. ordering by corresponding weight value. The second technique is an inter-cell, multi-cell per weight optimization. If more than one cell is required to store the cluster index, then some clusters can be more protected via the asymmetrical fault rates of MLC and non-uniform encoding, (a non-uniform allocation of cluster index bits across cells).

### 2.2.2 REDUCING WEIGHT STORAGE FOOTPRINT (EVALUATION)

In Figure 2.11, the area benefits of using CTT-MLC eNVMs with a fixed point encoding are measured against two baselines: SRAM and CTT-SLC (storing one bit per CTT cell). We evaluate

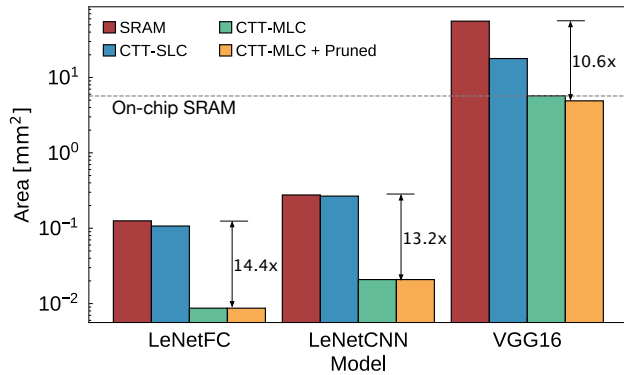
Model	Weights (#)	Error (%)	Q (int.frac)	Encoding	# Clusters (Max)	Config (lvls/cell)	Area (mm <sup>2</sup> )
LeNetFC	270K	1.91%	2.6	FxP	-	4,8,8	0.033
				C	16	16	0.008
				P+C	12	12	0.008
LeNetCNN	600K	0.85%	2.8	FxP	-	4,4,8,8	0.076
				C	8	8	0.021
				P+C	11	11	0.021
VGG16	135M	37.8%	2.10	FxP	-	4,4,4,4,16,16	40.53
				C	64(CNN) / 8(FC)	2,2,4,4 (CNN) / 8 (FC)	5.7
				P+C	64(CNN) / 9(FC)	4,4,4 (CNN) / 9 (FC)	4.9

**Table 2.0:** For each model, we report the quantization (Q) in the form integer.fractional bits, the number of clusters, and the configurations for each encoding scheme: fixed-point (FxP), clustered (C), and pruned & clustered (P+C). For VGG16, the two values of number of clusters, config, and level map given represent the values for CNN layers vs. FC layers, respectively, adapted from<sup>60</sup>.

the effectiveness of CTT devices to store the weights of three prototypical NNs listed in Table 2.0: LeNetFC is a three-layer fully-connected (FC) network, LeNetCNN is a five-layer convolutional neural network (CNN), and VGG16 is a much larger CNN that also contains large, prunable FC layers<sup>213</sup>. LeNetFC and LeNetCNN use the well-known MNIST dataset for handwritten digit classification, and VGG16 uses the popular ImageNet dataset of colored images to be classified into 1000 possible classes<sup>197</sup>.

The fault injection simulations are implemented using Ares<sup>193</sup>. For both fixed-point and cluster representations, a corresponding encoding transform function is defined. Starting from a trained model, the encoding transform is applied to the original parameters on a per-layer basis. Next, the faulty cells are randomly chosen using the error probability based on the number of levels per cell and the specific level value. The transformed parameters are used to evaluate the accuracy of the model for a specific encoding configuration. To find the configuration that minimizes the area footprint while maintaining model accuracy, all possible configurations of levels per cell and number of cells are tested for each NN and each variation of data format.

A comprehensive exploration of data encoding strategies and NN accuracy impacts identified that fixed-point CTT-MLC encoding requires 3 transistors per parameter for LeNetFC and 6 for VGG16, as listed in Table 2.0. Compared to an SRAM baseline, this results in a total area reduc-



**Figure 2.12:** Area reductions for clustered NN weight values, with no loss in DNN accuracy, for MLC-CTT storage vs. on-chip SRAM, adapted from<sup>60</sup>.

tion of up to  $7.6\times$ , as shown in Figure 2.11. Clustering is often preferable to fixed-point quantization because only cluster index pointers are stored. All three NN models require between 8 and 64 clusters to preserve model accuracy, which requires just 3 to 6 bits to represent each parameter, as summarized in Table 2.0.

When encoding clustered parameters in CTT-MLC, the benefits are immediate. For LeNetFC and LeNetCNN, each parameter can be stored in a single transistor because only 16 and 8 clusters are needed to preserve accuracy for each model. Compared to fixed-point quantization,  $k$ -means clustering saves  $3.8\times$  and  $3.6\times$  area when storing LeNetFC and LeNetCNN in CTT-MLCs. Compared to SRAM, storing the clusters in CTT-MLCs saves  $14.4\times$  and  $13.2\times$  for LeNetFC and LeNetCNN, respectively, as shown in Figure 2.12.

### 2.2.3 CUSTOMIZING WITH PER-LAYER, PRUNED DNN WEIGHTS

One additional step of customization that unlocks significant density benefits begins with the observation that in the VGG16 network, 89.5% of the parameters are in the FC layers. These FC layers need only 8 or 9 clustered values, while the CNN layers need up to 64 to maintain model accuracy



(Table 2.0). Using the same number of cells for parameters in the FC and CNN layers is wasteful as, once again, we need a single transistor per parameter to encode the parameters in the FC layers. For VGG16, 1 cell is used for each FC parameter and 4 cells are used for each CNN parameter, and this use of heterogeneous configurations saves us  $3\times$  area. For VGG16, CTT-MLCs provide  $9.8\times$  total area savings compared to SRAM.

This study also leveraged the particular MLC properties of CTT devices, where the initial state (see Figure 2.7) provides a unique opportunity for optimization: because it can be deliberately separated from the programmed states, the fault probability can be skewed to protect the most frequent cluster. To further leverage this protected state, the potentially high sparsity of stored weight values allows us to selectively map zero-valued weights to the less-fault-prone cell level. In VGG16, we prune 97% of all parameters in the first FC layer, which has over 100M parameters. This effectively protects a much larger proportion of the parameters compared to the unpruned network. In the CNN layers, pruning reduces the number of cells needed to store parameters from 4 to 3, leading to a further area reduction of  $1.17\times$  over CTT-MLC with clustering. Resulting sparsity and corresponding CTT-MLC area are reported in Table 2.0.

#### 2.2.4 DISCUSSION

The preceding results provide a concrete demonstration of the potential of concurrent optimization of MLC eNVM and neural network storage requirements. The storage of fixed-point parameters can be optimized using a non-uniform encoding that protects the sign and integer bits using fewer levels per cell, and this solution provides up to  $7.6\times$  area savings. As a further optimization, using  $k$ -means clustering together with MLC storage requires just a single transistor for each parameter in fully-connected NN layers. Additionally, pruning NN parameters and fine-tuning the cell level distributions protects the most frequent stored values and allows for more aggressive encoding configurations for CNN layers as well. The concurrent adoption of these optimizations reduces the

memory footprint of VGG16 to a total area of 4.9 mm<sup>2</sup>, which can be reasonably integrated in a modern SoC. While this work focused on density-reliability trade-offs for a specific eNVM implementation, the interactions between fault tolerance and storage formats, as well as the co-design methodology developed in this work, proved extendable to other system, application, and technology settings.

### 2.3 FAULT TOLERANCE IN-THE-LOOP WITH SPARSITY AND ERROR MITIGATION

In this section, we will study how DNN weight resilience under MLC programming varies as we tune application-level choices such as quantization, clustering, and sparsity. Additionally, we will study the intersection of resilience with system-level storage choices such as selective/dynamic MLC programming, sparse encodings, and memory allocation and partition choices. This section extends previous demonstrations of CTT memory using the previously described multi-level-programmed fabricated test chip (MLC-CTT, Section 2.1.0), and our careful algorithmic and architectural co-design can overcome the high fault rates that accompany high-density MLC storage. A more comprehensive system-level evaluation is reserved for Chapter 3.

#### 2.3.0 SPARSE ENCODINGS

Employing lossless sparse encodings to reduce required storage would be strictly beneficial for traditional memory technologies like SRAM. However, sparse-encoded values are no longer particularly fault tolerant due to the vulnerability of encoding metadata structures compared to the resilience of weight values. Therefore, additional analysis is required to determine optimal MLC storage schemes with sparse encodings<sup>183</sup>.

**Compressed Sparse Row Storage (CSR)** uses three data structures to encode a sparse matrix: an ordered list of all non-zero data elements by row (Weight Values), the column indexes of each non-

Model	LeNet5	VGG12	VGG16	ResNet50
Dataset	MNIST	CiFar10	ImageNet	ImageNet
Layers	4	12	16	54
Parameters	600810	7899840	138084352	24585472
Classification Error	0.83%	10.38%	35.07%	31.15%
Error Bound	0.05%	0.40%	0.57%	1.02%
Cluster Index Bits	4	4	6	7
Sparsity (% zero-valued)	89.9%	40.9%	81.1%	64.84%
16b Size	1.26MB	15.4MB	270MB	70MB
Pruned & Clustered (P+C)	316KB	3.86MB	101MB	30.6MB
CSR	84KB	3.78MB	30.2MB	25.1MB
BitMask	107KB	3.23MB	35.5MB	11.2MB

**Table 2.1:** DNN models for MaxNVM evaluation including baseline classification error, computed error bound, and storage requirements under varying quantization and encoding schemes, taken from <sup>183</sup>.

zero element (Column Index), and counters of the non-zero elements for each row in the original matrix (Row Counter). The relative overhead of CSR varies proportionally with sparsity. Model layers with less sparsity do not benefit from this encoding, so CSR is applied on a per-layer basis where worthwhile. Convolution layer weights are typically 3-D (filter width, height, and channels), and thus must be mapped to 2-D to be packed using CSR. NVDLA has specific requirements for data formatting into the convolutional core, which dictates a 2-D mapping that can be unpacked to be compatible with the datapath <sup>212</sup>.

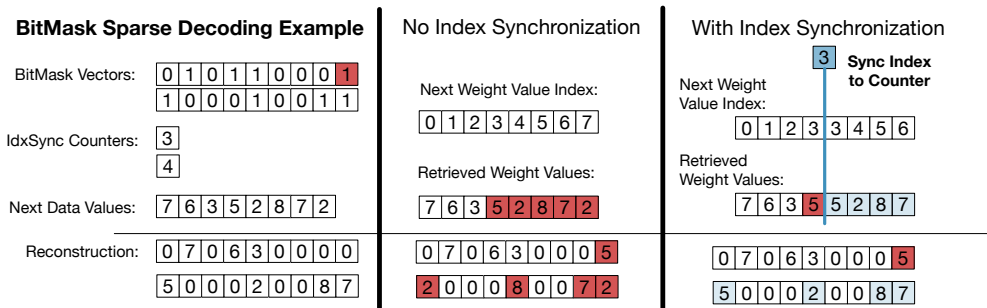
**BitMask Sparse Encoding Format** is natively supported in the NVDLA framework, and it utilizes an indicator bitmask for whether each weight is zero-valued and stores all non-zero data values in packed, 128 byte aligned groups. A simple example of a bitmask-based sparse encoding is also provided in Figure 2.13. We refer to this bitmask-based sparse encoding method as ‘**BitM**’, and we maintain compatibility with the defined NVDLA sparse format <sup>212</sup>. The effective compression of each encoding method is shown in Table 2.1, and the relative overhead of each method is dictated by the sparsity of the weights.

### 2.3.1 ERROR MITIGATION

Existing, well-established schemes to mitigate the impact of faults are effective in the context of detecting and correcting eNVM-related faults for DNN inference execution. However, we find that existing schemes are often over-provisioned in light of the relative fault tolerance of stored DNN weight values, and we explore variations as well as propose stripped-down alternative strategies for maintaining accuracy of DNN inference under fault-prone MLC storage with minimal overhead.

**Error-correcting codes (ECC)** are a pervasive mechanism to detect and correct errors in stored values. The basic mechanism is to, at the granularity of a block or page of data, maintain metadata (e.g., Hamming-style parity bits) computed according to the correct stored values, which can then be re-computed and validated against the data block when retrieved at another time. In this way, ECC introduces a nominal, constant overhead in terms of storage requirements and error-checking. Many variants of ECC exist in terms of what metadata is collected and maintained, how much metadata is used, and whether it is hard-coded into the memory array architecture or software-implemented. ECC may be deployed at different granularities and levels of the memory hierarchy.

Different amounts of relative ECC overhead were tested for each model in the studies that follow based on the number of correctable errors encountered, but over many trials the lowest overhead configuration (maintaining 24 parity bits for each 4KB of row counter values for CSR, for example) is sufficient to safely allow MLC-3 to be used. The resulting ECC bit storage overhead is strictly less than 1% per layer across all models. Additionally, ECC is single error correct, double error detect (SEC-DED), but the probability of a DED within the row counter structure, even for the largest model considered (ImageNet-VGG16), is on the order of  $10^{-18}$ . This probability is less than the standards for mass-produced standard memories<sup>2</sup>, so our design accepts this risk.



**Figure 2.13:** Introducing a lower-overhead approach to protecting bitmask-style sparse encoding than ECC; single bit-flip errors can be catastrophic in the BitMask structure by resulting in mis-assigned data values to the entire remaining matrix during reconstruction, but lightweight counters can prevent faults from propagating to preserve task accuracy by periodically synchronizing the index into the vector of non-zero data values, taken from <sup>183</sup>.

### 2.3.2 LIGHTER-WEIGHT ERROR MITIGATION WITH INDEX SYNCHRONIZATION

For even lighter weight error mitigation in the BitM sparse encoding, we additionally propose a method in which we store a counter for the number of non-zero (or zero, depending on which is lower) weight values we should have read in each 128 byte aligned block. When an error in the bitmask causes the wrong number of weights to be read, the counter is used to dynamically update where to read from next in the packed data array. We refer to this dynamic error mitigation strategy as **Index Synchronization of the bitmask or ‘IdxSync’**. Index synchronization of the bitmask does not correct any errors within 128 byte groups— rather, it prevents errors in previously read values from propagating to cause additional errors during weight matrix reconstruction, as demonstrated with a simple decoding example in Figure 2.13 and detailed in <sup>183</sup>.

### 2.3.3 CROSS-STACK EVALUATION OF SPARSE ENCODING CHOICES

The model optimizations and sparse storage schemes employed to minimize the burden of storing the DNN model parameters significantly impact the fault tolerance during inference tasks. Metadata structures become a significant portion of the storage needs per DNN layer, but bit errors in

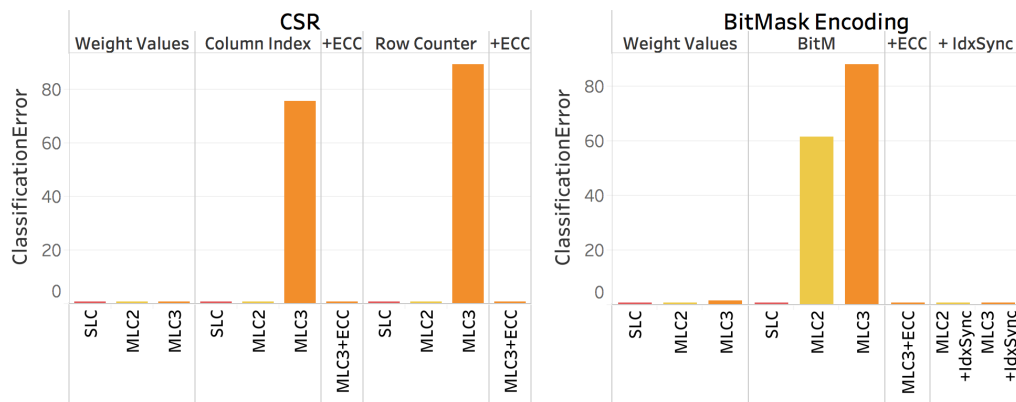
the metadata of any of the sparse encoding formats can have catastrophic impact when decoding the original matrix of values.

Thus, co-design with our carefully developed MLC eNVM fault models is required to enable dense, efficient storage. This evaluation uses the Ares fault-injection framework (Section 2.o) to quantify the impact of different encoding strategies on DNN classification error. This methodology was also applied across a wider range of models and encoding strategies with MLC-RRAM, as further explored in Chapter 3. The results of these experiments guide us in incorporating existing and proposed error correction and mitigation techniques in order to maximize the effectiveness of dense MLC eNVM storage for the execution of DNN inference.

#### 2.3.4 VULNERABILITY OF ENCODING STRATEGIES

Sparse encoding strategies allow fewer bits to be used when storing DNN weights, but experiments reveal that this compact data storage is in general much less tolerant of inter-level MLC faults. For all DNNs, weights that could be safely stored by their cluster index values in MLC-3 can no longer be safely stored in MLC-3 with sparse encoding. This is an incredibly important observation – as an example, there are layers of ImageNet-VGG16 in which, despite CSR enabling fewer bits to be stored, this does not overcome the area penalty of storing that layer’s parameters sparsely in SLC or MLC-2 rather than storing them densely in MLC-3.

These instances can be explained by the extreme vulnerability of certain sparse encoding data structures, as highlighted for MNIST-LeNet5 (Figure 2.14). This example demonstrates that the row counter (RC) and column index data structure of CSR exhibit exceptionally high vulnerability – storing with 3 bits per cell, which is equivalent to approximately one level-to-level fault in the row counter vector, causes a pronounced degradation in DNN accuracy. This is because a single misread RC value may cause an offset when reading from the non-zero data values such that all remaining non-zero data values are incorrectly assigned during re-construction. Similarly, column index

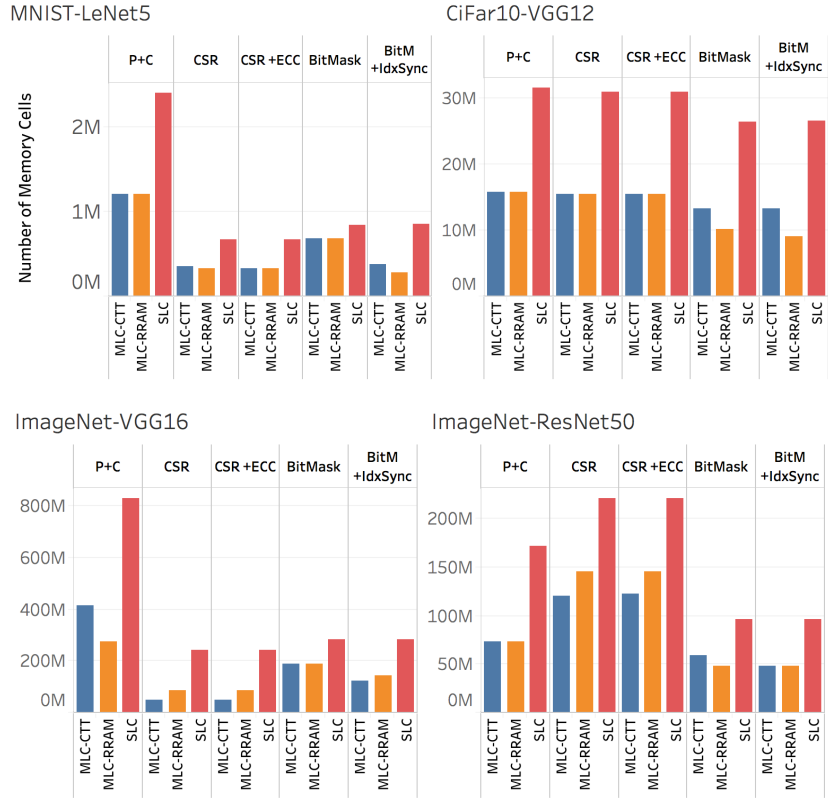


**Figure 2.14:** Introducing sparse encodings will have a direct impact on the resilience of our stored values. Existing, well-established schemes to mitigate the impact of faults are effective in the context of detecting and correcting eNVM-related faults for DNN inference execution, but a lower-overhead proposed technique (IdxSync) offers a compelling alternative, taken from <sup>183</sup>.

values (CI) are stored as relative indexes to the previous non-zero value within each row. Thus, a misread CI value may offset the remaining assigned data values, though the impact will be restricted to a particular row. The vulnerability of column indexes may be mitigated by using absolute rather than relative column indexes, but we find that this requires strictly higher overhead than integrating lightweight ECC. Relative to CSR, the bitmask-style sparse encoding is even more vulnerable, as shown in the right portion of Figure 2.14. A single bit flip in the bitmask will cause all remaining non-zero data values to be mis-assigned when reconstructing the matrix (illustrated in Figure 2.13). Thus, the bitmask structure cannot safely be stored in MLCs without some protective technique.

### 2.3.5 IMPACT OF ERROR CORRECTION AND MITIGATION TECHNIQUES

Looking at the classification error for the sample model in Figure 2.14, we see that both ECC and the IdxSync method enable the use of MLC-3 to store the bitmask without degrading model accuracy. The bitmask can account for a significant portion of the total storage per DNN layer (up to 65% of total bits to be stored in a given layer), so enabling denser storage of this structure can result



**Figure 2.15:** The storage strategy in terms of data format, sparse encoding, and MLC programming varies per layer for each DNN studied, with lightweight error mitigation via IdxSync corresponding to the lowest number of memory cells required for several models by enabling use of MLC-3 at no loss in application-level accuracy, taken from <sup>183</sup>.

in significant area benefits. For all models considered, applying IdxSync is sufficient to enable the safe use of MLCs for the bitmask. This configuration requires less storage overhead compared to ECC and reduces the decoder complexity compared to Hamming-style ECC.

Figure 2.15 summarizes the total number of memory cells required for each pairing of DNN model and encoding scheme such that there is no loss in application-level accuracy (i.e., image classification accuracy). In several cases, integrating index synchronization with the bitmask-style encoding results in the lowest required storage footprint by enabling use of MLC for the BitMask. However, the optimal storage strategy per DNN depends on model properties (per-layer sparsity, bits



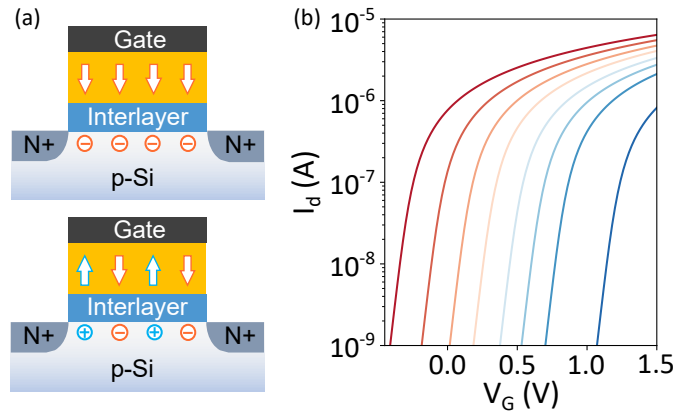
per weight value) and desired optimization target (lowest area dictating fewest cells, performance, energy) based on use case, so we continually leverage the results of these fault tolerance studies in determining proposed memory solutions in Section 2.4 and Chapter 3.

#### 2.4 MLC FeFET MEMORY FOR DNN INFERENCE AND GRAPH PROCESSING

This section considers yet another MLC-capable, promising eNVM candidate (FeFETs), develops detailed fault models and resiliency studies, and introduces several driving applications that can be co-designed with memory device properties to maximize storage density and energy efficiency. These studies, detailed in <sup>204</sup>, took full advantage of the intuitions, tools, and methods described to co-design MLC CTT and RRAM towards image classification DNNs and applied them to the unique properties of FeFETs and additional, data-intensive applications with unique fault tolerance properties (a DNN for natural language processing and graph processing kernels).

This work advanced the study of FeFET as a compelling eNVM solution by modeling and quantifying the impact of device-level design choices such as size and programming scheme on the accuracy of target workloads. FeFET memories promise dense storage with competitive read characteristics, but their viability and achievable density in a realistic application use case, such as DNN inference or graph analytics, was not previously explored. Optimizations targeting device size and iterative programming schemes must be co-designed with application-level metrics and memory array-level properties to maintain benefits, especially in multi-level-cell configurations.

This section begins by evaluating the device-to-device (D2D) variation for different FeFET cell areas using a previously validated device model <sup>58</sup>. Based on these preliminary device-level results, a proposed, customized programming scheme can increase the programming reliability for single-level cell (SLC) and multi-level cell (MLC) configurations. To quantify the relationship between resulting device characteristics and memory array performance, this work built on existing extensions of



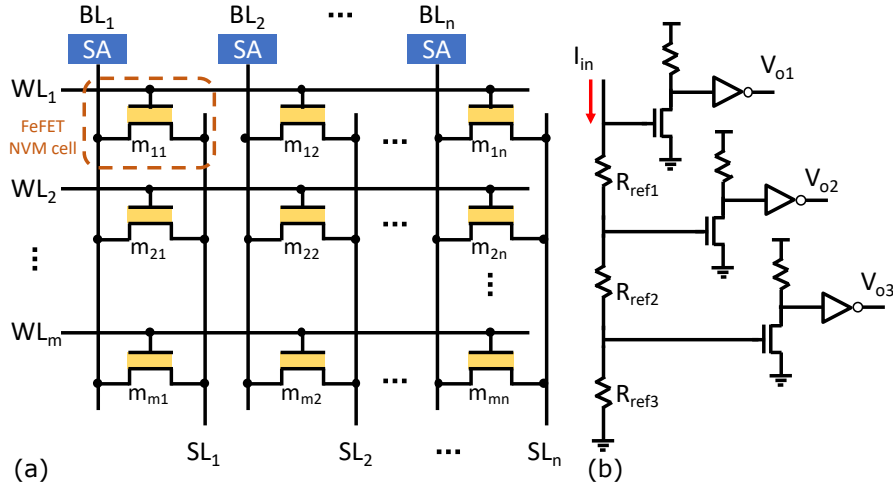
**Figure 2.16:** (a) FeFET device schematic; (b) Transfer characteristics of a FeFET device in 8 different resistance levels for 3-bit-per-cell (MLC-3) programming, adapted from <sup>204</sup>.

NVSim<sup>62</sup> tool to support a FeFET cell model and MLC programming<sup>183</sup>. Additionally, we develop a FeFET fault model based on the variability both at the memory and sensing circuitry level, and we extend an application-level fault injection framework<sup>193,185</sup> to evaluate the impact of FeFET device size and programming scheme on application accuracy.

#### 2.4.0 FEFET DEVICE AND ARRAY ARCHITECTURE

FeFETs, like other emerging devices, suffer from device-to-device variation issues. The sensitivity to variation increases when the number of levels increases in a multi-level-programming scenario. This problem even gets worse in smaller FeFET devices<sup>5</sup>. In the presented results, we account for D2D variations by employing a stochastic model of polarization switching in FeFET devices<sup>58,204</sup>. This device-level model has been previously experimentally validated<sup>58</sup>.

Figure 2.16 shows the programmability at 3 bits or 8 distinct levels per cell. This work employs 1FeFET AND arrays (Figure 2.17), where the source lines (SLs) connect the elements along the columns), which are more promising than the other types of FeFET memories (e.g., NOR arrays, where the SLs connect the elements along the rows) for two reasons: first, the cell size is equal or



**Figure 2.17:** (a) AND array memory architecture highlighting a single FeFET NVM cell; (b) Customized multi-level sensing circuit for four programmed (2-bit) MLC, taken from<sup>204</sup>.

even smaller than NOR arrays; second, the parallel bit-lines and source-lines reduce the write disturbance<sup>58,196</sup>. Note that the write disturbance can be further mitigated by applying a half-bias scheme (e.g., applying  $V_{write}/2$  to deselected cells)<sup>5</sup>.

The model uses Monte Carlo sampling on independent polarization domains in the ferroelectric layer. In doing so, it captures the essential behaviors required to perform scalability and reliability studies on FeFET memories, including (i) D2D variation as the cell size changes (different number of  $10nm \times 10nm$  ferroelectric domains); (ii) stochasticity of domain switching; and (iii) the accumulation of domain switching probability when a train of pulses are applied to the gate of the FeFET device. Thus, we can effectively project the impact of write schemes and device size on D2D variation, and, consequently, the impact of D2D variation on application level accuracy. Using this model and a bottom-up approach, we can effectively project the impact of D2D variation in terms of application level accuracy.

This work employs 1FeFET AND arrays (Figure 2.17, where the source lines (SLs) connect the elements along the columns), which are more promising than the other types of FeFET memories

(e.g., NOR arrays, where the SLs connect the elements along the rows) for two reasons: first, the cell size is equal or even smaller than NOR arrays; second, the parallel bit-lines and source-lines reduce the write disturbance<sup>58,196</sup>. Note that the write disturbance can be further mitigated by applying a half-bias scheme (e.g., applying  $V_{write}/2$  to deselected cells)<sup>5</sup>. Additionally device-level details are provided in Chapter 1.2.2.

#### 2.4.1 TARGET APPLICATIONS

We evaluate the impact of FeFET reliability on target applications that are (1) data-intensive in terms of required capacity and read bandwidth and (2) have infrequent or highly batched write accesses. In embedded and mobile SoCs, the high density and compelling efficiency of FeFET memories could be crucial in enabling on-device or otherwise power-efficient execution of these critical applications, and performance will not be debilitated by potentially long writes.

Deep Neural Networks (DNNs) for vision tasks and natural language processing (NLP) are well-studied driving applications that have demonstrated compatibility with multi-level-cell (MLC) eNVM storage<sup>183,59,226</sup>. DNN inference performance and power-efficiency benefits significantly from increased on-chip memory density, particularly for storage of weight parameters that are infrequently updated. Thus, we evaluate the viability of FeFET MLC memories for two representative DNN workloads: ResNet18 for image classification (CiFAR10), and the ALBERT transformer-based model for natural language understanding (MNLI)<sup>88,126</sup>. Another critical category of data-intensive workloads are those that operate on large graphs<sup>17</sup>, such as search tasks on social network graphs, though resilience of graph formats to MLC eNVM storage has not been addressed in prior work. Thus, we additionally evaluate the viability of FeFET MLC memories for storing social network graphs with sample graphs representing Wikipedia article voting patterns and anonymized social circles from Facebook<sup>135</sup>.

#### 2.4.2 ARRAY-LEVEL MLC FeFET SIMULATION

The FeFET Monte Carlo current model allows us to characterize the behavior of FeFET memory cells in isolation. NVMEExplorer leverages extensions to NVSim<sup>62,204</sup> to extrapolate our device-level characterization study to memory array architecture, as discussed further in Chapter 4. NVSim does not natively support FeFET memory cells, so we added a customized memory cell definition<sup>204</sup>. Moreover, we modify NVSim to estimate the costs of MLC sensing circuitry.

We integrate a new memory cell definition using the energy, delay, and area results collected from SPICE simulations. Our evaluation considers two programming schemes, which are discussed in detail in Section 2.4.5. For the read operation of the SLC memories, we employ a simple voltage based sense amplifier. However, for the MLC memories, we integrated the energy, area, and delay results that we collected in HSPICE simulations for our design discussed in Section 2.1.1. For the single-pulse programming scheme, we derive the set and reset energy and latency for different cell sizes from HSPICE simulations. The write energy and latency for other programming schemes use the average number of set and reset programming pulses computed by sampling 1500 FeFET cells for D2D variation, in addition to estimated energy and latency due to write circuitry.

To model the write energy, we first obtained the energy required to program the FeFET cell for a single reset pulse and a single set pulse with different gate sizes using SPICE simulation. Second, we calculated the average number of sets and resets needed for 1500 FeFET cells (to have enough data points) to reach their target levels for different programming schemes (single-pulse and write-verify). Then, we can calculate the write energy of FeFET cells by multiplying the number of set and reset pulses to the obtained pulse energy. An additional change to improve model fidelity is that FeFETs fabrication requires adding a ferroelectric layer on top of a traditional MOSFET gate, so the gate equivalent oxide thickness is increased and, therefore, the gate capacitance. We captured this in NVSim by setting FeFET gate capacitance  $1.73 \times$  bigger than the default CMOS gate capacitance<sup>58</sup>.

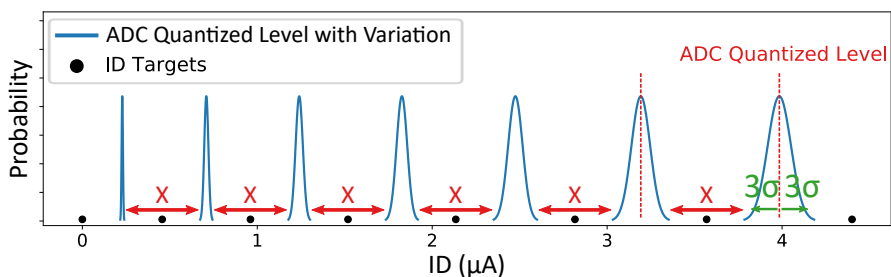


Figure 2.18: ADC quantized levels with variation as a Gaussian function with  $3\sigma$  deviation of 5% and target currents for 3-bit FeFET MLC, taken from<sup>204</sup>.

### 2.4.3 SENSING CIRCUIT DESIGN

For a 1-bit read operation, this work employed a simple voltage-based sense amplifier characterized in SPICE. The same sense amplifier model is employed to build a parallel sensing scheme for MLC FeFET operation. As discussed in Section 2.1.1, a parallel sensing scheme compares the current from a memory cell against  $2^n - 1$  reference levels and returns an  $n$ -bit binary word. Figure 2.17 depicts the circuit schematic of the sensing circuit for a 2-bit read operation. The operation is similar to a Flash ADC and is similarly affected by D2D variation, which in turn has an impact on memory reliability, as in the RRAM and CTT fault models developed in Section 2.1.1.

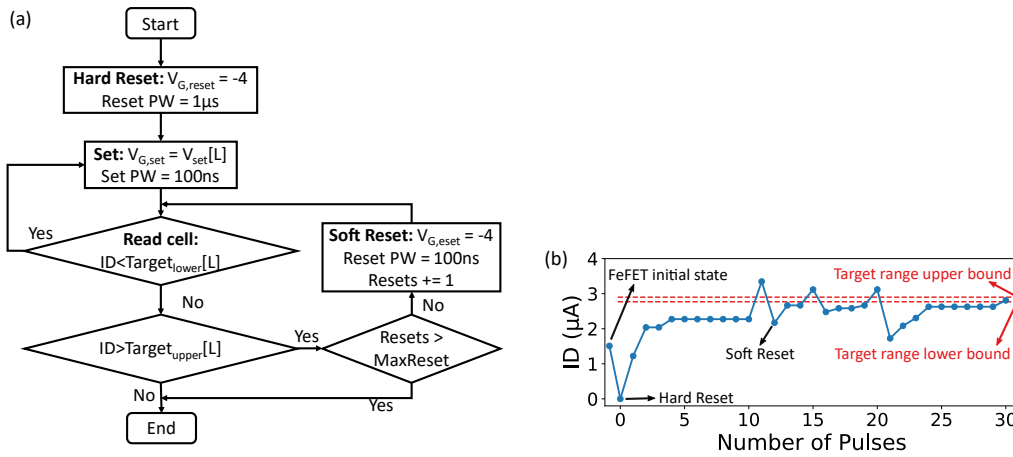
We model the effects of D2D variations on transistor dimensions, resistance values, etc., as a Gaussian function with a  $3\sigma$  deviation of 5%. As a result, the quantized levels show variability proportional to the threshold currents. Based on this constraint, we propose spacing the MLC programming currents such that the sensing threshold distributions are equally spaced. This approach uses the extra margins in low-current levels to distribute the read errors due to sensing threshold shifts more evenly across the entire programming window, as depicted in Figure 2.18. Figure 2.18 shows a MLC-programmed schematic at 3 bits or 8 distinct levels per cell, displaying changes in variation per ADC quantized level over the programmable range of current values. This is distinct behavior from the technologies considered in previous sections (i.e., CTT and RRAM), but is sim-

ilarly expressible as a set of normal distributions for integration with application-level resilience analysis. The energy, latency, and area for the resulting design is incorporated in NVSim to model sensing at the memory array architecture level.

#### 2.4.4 CROSS-APPLICATION FAULT INJECTION FRAMEWORK

Evaluating device non-idealities at the application level requires a balance of fault model accuracy and simulation performance. We approach this problem by building a fault-injection simulator that integrates with existing application-level frameworks. As the focus of this work is on DNN and graph applications, we leverage two popular Python packages, namely PyTorch<sup>181</sup> and SNAPpy<sup>136</sup> to execute representative workloads. Previous work has explored the evaluation of faults in DNN models<sup>193</sup> and extended the study to eNVM errors in DNN applications<sup>60</sup>. We extend existing MLC eNVM fault modeling to simulate FeFET memories based on the model discussed in Section 2.1.1 and ADC quantized level variations. Moreover, we use a general input interface to process parameters from both DNN and graph workloads, a major extension and generalization of the capabilities of Ares<sup>193,183</sup>. For each target application, the fault injection tool reads in the application data to be stored in FeFET-based memory and applies a quantization transform followed by MLC encoding based on the selected configuration, quantifying application-level accuracy impacts over many random trials.

Stored data values are assigned to FeFET current levels, and we sample the current based on the FeFET Monte Carlo model. The currents computed in this first sampling process are then compared against current sensing thresholds which are also sampled based on ADC variations. The sensed currents are then converted back to quantized values and used to repeat workload execution and evaluate the effect of memory faults on the target application.



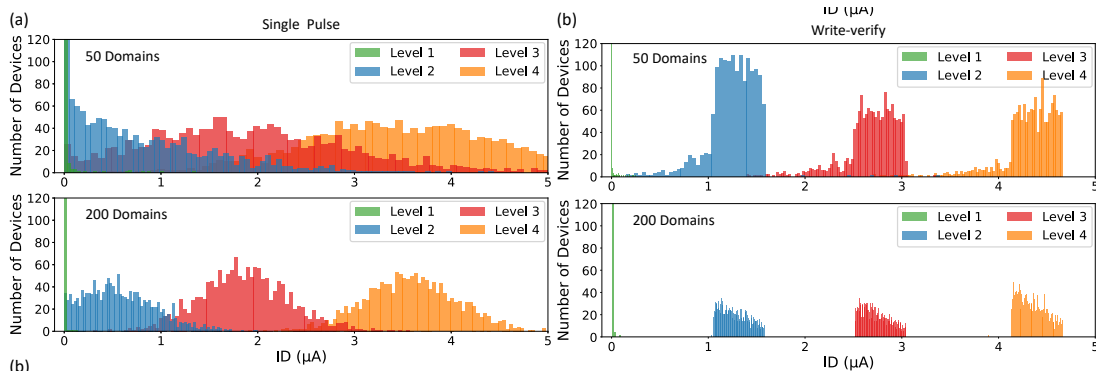
**Figure 2.19:** (a) Proposed write-verify programming schematic; (b) Example of pulse by pulse level tuning process of an individual FeFET device into the target range, taken from<sup>204</sup>.

#### 2.4.5 OPTIMIZATION STUDY – PROGRAMMING SCHEMES

We consider two approaches for programming multi-level FeFETs, (i) single pulse; and (ii) write-verify. The single-pulse approach first performs a hard reset of the device (a  $-4V$ ,  $1\mu\text{s}$  pulse), followed by one pulse of amplitude dictated by the target level. An issue with this approach is that as we move to smaller FeFET devices, (i.e., lower number of domains) the D2D variation increases to a point where the rate at which we mis-read a programmed level will increase significantly. While appealing in terms of simplicity, the single-pulse scheme is not robust to D2D variation for smaller FeFET cells, as evident in the overlap of resulting current distributions in Figure 2.20 for 50-domain cells.

To solve this problem and realize multi-level FeFET cells with tightened cell distribution, we proposed a write-verify scheme described in Figure 2.19. First, we do a hard reset on the device to make sure that the device reaches the lowest state. Next, the set voltage is chosen based on the target level with the pulse width of  $100\text{ns}$ . The set pulse is then applied to the gate of the FeFET device. After the first set pulse, the cell is read and the device current is measured. If the measured current is smaller than the lower bound of the target programmed range, another set pulse is applied to the





**Figure 2.20:** (a) Current distribution of 2-bit FeFET MLCs with single pulse programming for 1500 cells; (b) Current distribution of 2-bit FeFET MLCs with write-verify programming for 1500 cells, taken from<sup>204</sup>.

gate of the FeFET. However, if the measured current is larger than the upper bound of the target range (and we have not exceeded the maximum number of resets), a soft reset pulse is applied on the gate of FeFET. Finally if the device is in the targeted range we stop the programming.

The write-verify scheme proposed in<sup>204</sup> (Figure 2.19) helps combat D<sub>2</sub>D variations by producing tighter level distributions. This scheme leverages the stochastic behaviour of the FeFET domain switching to reach the targeted level. The advantage of this approach is that it reduces the overhead in the write circuitry while reaching the tighter cell current targets. This scheme cannot be employed for other emerging devices such as PCM since they need programming pulses with increasing amplitude and width to reach the desired current targets. The soft reset pulse has two advantages: First, its pulse width is 10× shorter than the hard reset pulse, which will increase the performance of the write. Second, it does not fully reset the device to the zero state because of the shorter pulse. This gives us more control to set the device into the targeted range. A similar strategy has been applied to RRAM devices<sup>33</sup>, but this has never been studied for FeFETs before to the best of our knowledge.

Our evaluation showed that only a small subset of devices (less than 0.1% for 200-domain cells) fails to reach the target range after 10 soft reset cycles. Therefore, we fix the maximum number of

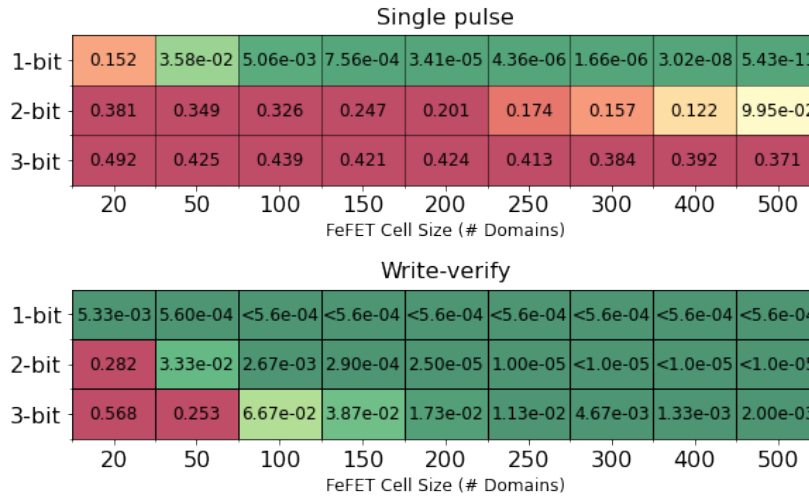
soft resets to prevent prohibitively long and energy-consuming programming sequences. The programming sequence is terminated either if the measured device current is within a target range, or if the number of applied soft reset pulses reaches its maximum count (Figure 2.19).

Figure 2.20 shows the current distribution of 1500 2-bit FeFET MLCs programmed using single-pulse and write-verify programming schemes. Figure 2.20 shows that there are large overlapping areas between different levels if the single-pulse scheme is used. The level distribution overlap worsens when the number of domains is reduced from 200 to 50. Figure 2.20 shows that using write-verify scheme, overlaps can be mitigated for 200 domain cells. However, even with the write-verify scheme, there is some overlap in 50 domain cells.

#### 2.4.6 MEMORY DESIGN TRENDS

For each of the programming schemes discussed in Section 2.4.5, we carefully consider the fault characteristics of the FeFET memory as we vary the number of programmed levels per cell and the cell size. The highest resulting inter-level fault rate per cell design and programming scheme is shown in Figure 2.21. Our fault model additionally captures the asymmetry of inter-level fault rates (i.e., when it is more likely for the programmed level to be mis-read as a lower level than a higher one), which is increasingly apparent in 2-bit and 3-bit programming. Thus, the raw maximal fault rate alone does not capture the complexity of the potential impact of MLC FeFET behavior on application accuracy. However, we note that the fault rates when we program 2 bits per cell, on average, exhibit more asymmetry than 3 bits per cell (i.e., it is more likely for a cell's value to be mis-read as a lower value than a higher one). Figure 2.21 highlights those fault rates which are strictly higher than similarly provisioned MLC-programmed RRAM storage as a proxy for the viability of different FeFET memory configurations.

As discussed in Chapter 1.1, there are multiple classes of important workloads that can be tolerant of long write latency and would benefit immensely from increased storage density and effi-

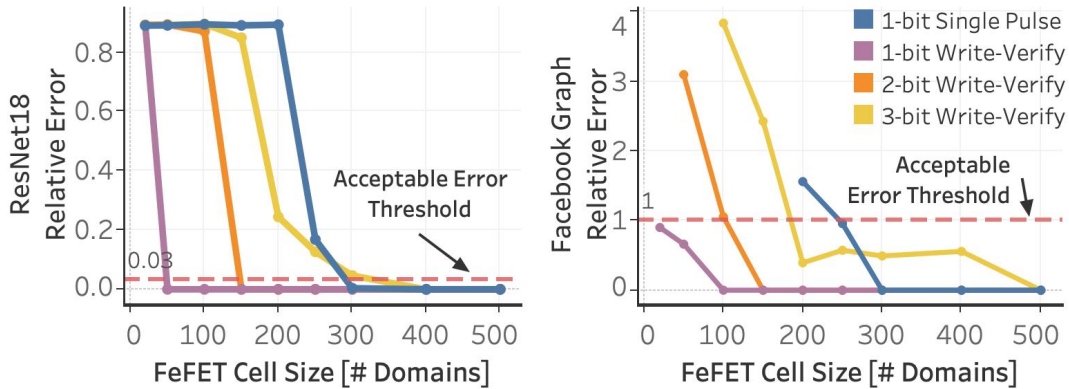


**Figure 2.21:** Shmoo plot of maximum read fault probability as a function of cell size and bits per cell for single pulse vs. write-verify programming, colored according to whether the raw fault rate is higher (red), lower (green), or similar to MLC-programmed RRAM taken from<sup>204</sup>.

ciency. However, these workloads may exhibit different resilience to MLC fault characteristics. In light of the relatively high error rates for MLC programming across cell size shown in Figure 2.21, it is critical that we evaluate application-level implications to identify the densest allowable storage scheme without degrading application accuracy.

#### 2.4.7 APPLICATION-LEVEL EVALUATION

We evaluate two DNN workloads which differ both in terms of target application and eNVM utilization. The first is image classification using ResNet18. In this case, we consider a system in which the full model is stored in the FeFET memory. Figure 2.22 shows how application error relative to baseline classification accuracy increases significantly as cell size decreases across programming schemes. For 1-bit, single-pulse programming (no verify), the minimum FeFET cell size that preserves inference accuracy is 300 domains. Introducing the write-verify scheme allows the memory cell size to be scaled more aggressively, with 1- and 2-bit storage displaying minimum cell sizes of 50



**Figure 2.22:** Fault injection studies are performed per cell size and programming scheme to evaluate resulting application error, as measured by relative degradation from baseline accuracy. The minimum cell size that keeps relative error below the acceptable threshold dictates densest storage per scheme, taken from<sup>204</sup>.

and 150 domains, respectively.

The second DNN we consider is natural language inference running on ALBERT. BERT-based DNNs are often re-used across multiple tasks (e.g., translation vs. sentiment analysis), so we propose to store the shared embedding parameters in FeFET memory, while the task-specific portion of the network is stored separately, an approach that has proven successful in maximizing energy efficiency for multi-task DNN inference<sup>59</sup>. In this case, the 1-bit, single-pulse configuration requires 200 domains, while for the write-verify cases, 1-, 2-, and 3-bit configurations can be scaled down to 50, 150, and 150, respectively (summarized in Table 2.2). Compared to the relative vulnerability of 3-bit MLC for ResNet18, partitioning ALBERT to store more vulnerable parameters in SRAM provides the opportunity for more aggressive scaling in terms of FeFET cell size.

To determine the viability of different FeFET cell designs and programming schemes for graph analytics tasks, we measure the average impact to the accuracy of a set of breadth-first search queries on the example graphs as a proxy for maintaining network structure and resilience to faults<sup>17</sup>. We store input graphs as adjacency matrices and perform fault injection to quantify resilience to varying MLC storage and programming schemes. The relative error increase for the Facebook graph,

BPC	Programming	ResNet18	ALBERT	Wiki	Facebook
1	Single Pulse	300	200	250	250
1	With Verify	50	50	50	20
2	With Verify	150	150	150	150
3	With Verify	400	150	400	200

**Table 2.2:** Summary of minimal cell size in number of domains per programming scheme and bits per cell (BPC) without application accuracy degradation, adapted from<sup>204</sup>.

for example, (Figure 2.22), demonstrates workload-specific sensitivity to FeFET fault characteristics distinct from ResNet18 in terms of allowable cell size. The minimum cell size per workload and per programming scheme is summarized in Table 2.2. For example, we see that the Wikipedia input graph is resilient to FeFET storage at 50 domains, while both graph workloads are amenable to 2-bit FeFET storage down to 150 domains. Due to the varying sparsity and structure of the two graph workloads studied, we observe different fault resilience in the 3-bit configuration, and the Wikipedia voting graph degrades in terms of average query accuracy when using a cell with fewer than 400 domains. Each mis-read value in the adjacency matrix corresponds to an erasure or erroneous addition of a connection between two graph nodes, and it is possible that each mis-read value has a disproportionate impact on a graph that is less clustered and more sparsely affiliated (as in the Wikipedia graph) than one with more disparate, strongly connected social circles (as in the Facebook graph), but further analysis would be required to justify this argument.

#### 2.4.8 DISCUSSION

Our results show that implementations targeting inherently robust applications can lead to more efficient memory array architectures. For example, in most of the workloads we considered in this work, the SLC configuration using a write-verify programming scheme results in a memory array design which has the best combined density and read performance. This result can be explained by considering that replacing the SLC configuration with MLC2 requires increasing the cell size by  $3\times$

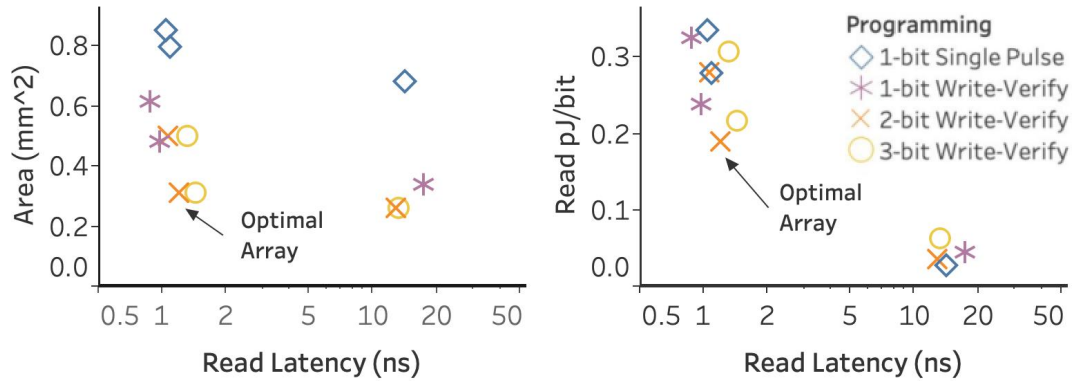
Workload	Opt. Scheme	Storage Needs	# Cell Domains	Area ( $mm^2$ )	Read Latency (ns)	Read Energy (pJ / bit)	SET Latency ( $\mu$ s)	SET Energy (pJ / bit)
ResNet18	SLC	24MB	50	1.686	1.866	0.461	1.47	0.745
ALBERT	MLC2	4MB	150	0.313	1.20	0.189	1.80	0.438
Wikipedia	MLC2	6MB	150	0.682	1.268	0.278	1.80	0.422
Facebook	MLC2	2MB	150	0.241	0.976	0.169	1.80	0.234
SRAM (16nm)	Ref.	4MB	N/A	3.9	1.3	0.5	0.001	0.5

**Table 2.3:** Summary of optimal programming schemes, device characteristics, and resulting memory array characteristics provisioned per workload, adapted from<sup>204</sup>.

in order to preserve application-level accuracy.

Though all of the examined workloads are amenable to MLC FeFET storage, in order to derive the most efficient FeFET storage solution per application, we must consider the achievable zero-accuracy-loss storage density in conjunction with memory array performance, area, and energy characterization. For example, we find that SLC FeFET under the write-verify scheme can be safely used down to a cell size of 50 domains or smaller (Table 2.2), so this programming scheme consistently provides highly dense, zero-accuracy-loss memory. On the other hand, our simulation on ALBERT give us an opportunity to uncover the potential of MLC storage solutions for data-intensive, fault-resilient workloads.

Figure 2.23 also demonstrates that 2-bit storage is a compelling design in terms of density and read characteristics, with 3-bit showing competitive performance in all metrics, though not providing the pareto-optimal configuration highlighted in Figure 2.23 and summarized for each workload in Table 2.2. The storage density achieved here is a full order of magnitude higher than that of 16nm SRAM, in which a 4MB array requires about  $4mm^2$  according to NVSim evaluation, cited as a reference point in Table 2.3. Similarly, the observed read latency and energy per bit tends to match or improve upon SRAM. We repeat analysis of area, latency, and read energy per bit of provisioned memory arrays per-application to identify FeFET memory arrays that minimize latency and energy. In this way, the FeFET arrays highlighted in Table 2.3 will cause minimal impact to workload execution time while providing high density, energy-efficient embedded storage.



**Figure 2.23:** For an example workload (ALBERT, 4MB capacity), we show key characteristics for FeFET memory arrays with acceptable application accuracy across programming schemes and NVSim optimization targets. From these, we select optimal array configurations summarized in Table 2.3 that minimize array area, read latency, and read energy per access, taken from <sup>204</sup>.

It is important to note that these design points are not fully optimized, which indicates that even higher density could be achieved with architectural enhancements <sup>183,185</sup>. For example, incorporating lightweight error mitigation or error correction strategies could enable use of even higher density storage solutions, such as 3-bit MLC with smaller cell sizes. Alternatively, more sophisticated programming schemes could be used to further tighten the current distribution of MLC FeFET configurations. However, more complicated programming schemes would add area, latency, and energy overhead due to write circuitry.

This work is a demonstration of the viability of MLC-programmed FeFET eNVMs with an application-driven evaluation of these memories in light of DNN and social network graph resilience and memory access patterns. The presented methodology and studies provide exposure to the critical trade-offs among write costs, storage density, fault characteristics, and application impact that will drive future development of highly-dense FeFET memory solutions. These results build confidence that MLC FeFET storage could be effectively leveraged for both deep neural network inference and graph processing tasks, even in the absence of more involved optimizations. Yet, these

results are limited by the lack of integration with system-level optimizations and a more comprehensive system evaluation, which is the focus of Chapter 3.



*If it's so beautifully arranged on the plate, you know someone's fingers have been all over it.*

Julia Child, "The French Chef"

# 3

## MaxNVM: Maximizing Memory Efficiency for ML Accelerators

DEEP NEURAL NETWORKS (DNNs) are a critical, driving application in a variety of systems and domains, from object-detection and object-tracking for self-driving cars, to classification and data analysis tasks on embedded sensor nodes and implanted devices, to diagnoses and drug discovery

in the medical field<sup>192</sup>. Particularly in deeply-embedded computing environments, efficiency is paramount – both in terms of performance, operating power and thermal impacts, and in terms of energy usage and potential impacts on battery life and device lifetime.

For state-of-the-art DNN hardware accelerators, fetching weights from DRAM persists as a primary performance and energy bottleneck, limiting inference efficiency<sup>192</sup>. Ideally, DNNs weights would be stored entirely in on-chip memory, such that DNN inference can be fully executed on-chip without costly DRAM data fetching. However, even with aggressive weight compression and layers of algorithmic optimization, capacity requirements of modern DNNs often remain unrealistic for storage in on-chip SRAM.

Emerging, embedded non-volatile memory (eNVM) technologies are one promising solution for eliminating DRAM inefficiencies, as described in Chapter 1. eNVMs provide high-capacity, low read-latency storage and are significantly more dense than SRAM. Many eNVMs, including RRAM, PCM, and CTT, also have multi-level cell (MLC) capabilities, allowing multiple bits to be packed into a single device to further increase density. eNVMs are not perfect—two main drawbacks are decreased reliability and high write latency. First, while single-level eNVMs are reliable, achieving the ultra-high densities necessary to eliminate DRAM for DNN inference requires aggressive MLC designs. However, these MLC designs can incur high memory access fault rates. Second, eNVMs offer fast read access, typically on the same order as SRAM. However, writes can be orders of magnitude slower, as well as more costly in terms of energy or operating conditions, as they often alter the physical property of the storage material. Given DNN inference in embedded settings often require infrequent weight updates and are implicitly fault tolerant (as explored in Chapter 2), we identify MLC eNVMs as a potential solution to improve DNN inference efficiency by storing weights on-chip and eliminating DRAM accesses.

With algorithm-hardware co-design, DNN inference is a near-ideal fit for MLC-eNVM as the benefits (high density, fast read, low-power) can be had without suffering from the cons (fault-

prone, slow writes). This chapter focuses on the demonstration and evaluation of MLC eNVM technologies applied towards highly-efficient DNN inference settings, in each case employing optimizations that span and exhibit inter-dependencies among multiple layers of the computing stack.

As an efficient baseline, MaxNVM starts at the algorithmic level, applying clustering, pruning, and sparse weight encoding (CSR and BitMask). To optimize for fault-prone MLC storage, MaxNVM co-optimizes the sparsity of DNNs with the storage density of MLC eNVMs, which directly impacts the reliability of the storage medium, to reduce the memory requirements without sacrificing accuracy. MaxNVM unveiled that sparse-encoded weights to be more vulnerable than dense weights, which limits MLC eNVM density, as previewed in Chapter 2.3. This chapter pushes the limits of storage density through increasing the levels-per-cell in MLCs and by applying protective mechanisms, ECC and IndexSync (proposed), in-the-loop with application-level metrics like accuracy and achievable frame rate and system constraints like total on-chip area and hardware accelerator bandwidth requirements.

To demonstrate the efficacy of our co-design approach on the actively evolving eNVM landscape, MaxNVM conducted detailed evaluations of two fundamentally different eNVMs: RRAM and CTT. Both are extremely dense, MLC-capable, and can be made CMOS-compatible for on-chip integration with logic in advanced process technology nodes. CTTs have fast read latency and are very low-power, but incur inordinately long write latency. Compared to CTTs, RRAM has a faster write latency, but is less energy-efficient.

The MaxNVM project conducted a crucial first step in developing models for RRAM and CTT, including MLC-programming, by extending NVSim<sup>62</sup> in ways that were, in turn, significantly generalized and expanded in other research projects I have led or contributed to over the course of this thesis work<sup>204,59,226,185</sup>. RRAM cell parameters and fault models are taken from published work, and MaxNVM implemented two models to represent projected<sup>268,269,53</sup> and demonstrated<sup>252,255,128</sup> RRAM scaling to smaller technology nodes. CTT parameters and fault distribu-

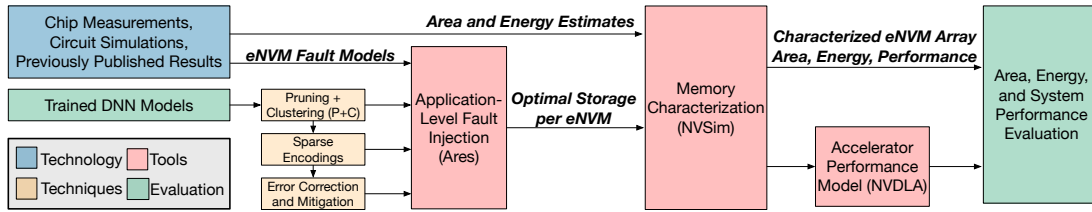
tions are derived from a measured chip prototype<sup>183,157</sup>.

Several studies presented in this chapter leverage NVDLA—a state-of-the-art, industrial-grade CNN accelerator. NVDLA is an open-source accelerator architecture from NVIDIA, providing performance modeling and comprehensive documentation so as to be extendable and effective as a system model for academic projects<sup>212,178</sup>. Experiments in Section 3.1 show that even large models, e.g., VGG16 and ResNet50<sup>213,88,197</sup>, can reasonable fit on-chip with the MaxNVM co-design approach. CTT results in the lowest energy-per-inference design point at up to  $3.2\times$  reduced energy per inference. In a system where weights are updated more frequently, RRAM presents a nice compromise of writing weights orders of magnitude faster while sacrificing approximately 20% energy efficiency. In a case study, we constrain all on-chip memory to fit within  $1\text{mm}^2$  and sweep the percentage given to SRAM and eNVM. The study concludes a balance of eNVM and SRAM is ideal to maximize performance and efficiency.

After detailing the system configurations and evaluations and discussing the compelling results of MaxNVM, this chapter briefly summarizes two additional projects in specialized DNN accelerators to which I have contributed. In the contexts of both Multi-Task Image Processing (MEMTI<sup>59</sup>, Section 3.2) and Natural Language Processing (EdgeBERT<sup>226</sup>, Section 3.3), total system efficiency and on-chip storage density are significantly improved by proposals integrating multi-level-cell-programmed eNVMs, and each system design benefited from cross-computing-stack considerations in data format, fault tolerance, and memory technology characteristics.

### 3.0 END-TO-END, CO-DESIGN METHODOLOGY FOR MLC eNVM

The MaxNVM co-design strategy incorporates optimizations and techniques at algorithmic and architectural levels. After developing a fault model based on technology-specific device characteristics and SPICE models of sensing circuitry, we use a previously-validated fault injection framework



**Figure 3.0:** MaxNVM summary of the tools, optimizations, and intermediate results used in final system evaluations; technology characteristics, specific system requirements, and a variety of application-level optimization opportunities inform the proposed MaxNVM system designs, taken from <sup>183</sup>.

to determine whether a given storage scheme impacts DNN accuracy <sup>193</sup>, as described in Chapter 2. We leverage and extend well-known tools to model the energy, performance, and area of the proposed systems at the memory-array <sup>62</sup> and system-architecture <sup>178</sup> levels. The interaction of these methods as they contribute to the final evaluation is summarized in Figure 3.0.

### 3.0.1 MODEL OPTIMIZATIONS

Proposed memory systems are evaluated against a competitive and realistic baseline by enforcing iso-accuracy and incorporating common optimizations on the chosen DNNs, as summarized in Table 2.1. All presented energy, performance, and area improvements maintain model accuracy within computed Iso-Training error bounds, as defined in <sup>192</sup> and briefly described in Chapter 1.1.1. We consider DNNs of different sizes: one CNN model for the small MNIST hand-written digits dataset, two more realistic and well-known DNN models (VGG16 and ResNet50) for the much larger ImageNet dataset, and another VGG-like topology for the mid-size CiFar10 dataset to span the gap between these cases <sup>213,88,125,197</sup>. Magnitude-based weight pruning with retraining is used to sparsify DNN weights, as the models are widely known to be over-parameterized. The resulting proportion of zero-valued weights is listed in Table 2.1, and further model optimizations are performed without retraining.

One popular technique to reduce the number of bits required to store each DNN weight is to

reduce precision using fixed-point quantization. Depending on the dynamic range of the DNN weight values, the number of integer and fractional bits can be drastically reduced, even to a few or a single bit per weight at some loss in model accuracy<sup>99,14</sup>. Another way to reduce the number of bits required to represent each weight value is to use  $k$ -means clustering for each layer of the DNN<sup>192</sup>. For the models considered, we found that all the weight values within a given layer can be represented by 16 to 128 unique clustered values at no loss of accuracy. Thus, each weight can be encoded as a 4 to 7 bit cluster index value, with a simple look up table per layer to map indexes back to values. We find clustering uses strictly fewer bits per weight than fixed-point quantization without significant re-training for all DNNs (Table 2.1). For more details on model optimizations and DNN simplification techniques, please see Chapters 1.1.1 and 2.3.

### 3.0.2 ERROR CORRECTION AND MITIGATION

Vulnerability to faults changes when weights are stored using sparse encodings, as explored in Chapter 2.3. A simple strategy to mitigate the increased classification error is to store more vulnerable structures using fewer bits per cell (e.g., SLC or MLC<sub>2</sub>)<sup>60</sup>. We additionally explore when it is beneficial to incorporate Hamming-style, parity-based ECC. The configuration used is the lightest-weight ECC considered for NAND flash memories<sup>200</sup>. Note that if values are binary-encoded in a MLC, a single level-to-level fault is not equivalent to a single bit flip, so for precise error detection and correction, Gray coding is used to represent ECC-protected values in MLCs. MaxNVM additionally proposes and evaluates a lighter-weight error mitigation technique for bitmask-encoded sparse data, which is described in fixmeSection.

NVSim Parameter	Value	NVDLA Baseline	NVDLA-64	NVDLA-1024
Data Width	8 - 128	Convolutional Buffer Size	128KB	256KB
Banks	$(1 \times 1) - (N \times N)$	Number of MACs	64	1024
Mats	$(1 \times 1) - (N \times N)$	SRAM Capacity	512KB	2MB
Optimization Targets	Area	Frequency	1 GHz	1 GHz
	Read Latency	Datapath Area	$0.55 \text{ mm}^2$	$2.4 \text{ mm}^2$
	Read EDP	SRAM BW	6 GB/s	25 GB/s
	Read Energy	DRAM Read BW		25 GB/s
	Leakage Power	LPDDR4 DRAM Power	100 mW	200 mW

**Table 3.0:** NVSim, baseline NVDLA parameters<sup>62,212,28</sup> for use in system-level evaluation of MaxNVM, taken from<sup>183</sup>.

### 3.0.3 MEMORY MODELING

We integrate new memory cell definitions in NVSim using the energy, performance, and area measurements from our test chip, published work, and SPICE simulations of different sensing circuitry topologies<sup>62</sup>. NVSim evaluates all possible memory bank configurations for a given capacity, optimization target, and number of bits per cell (Table 3.0). Pareto-optimal points are chosen from NVSim outputs to fit within area, performance, and/or bandwidth constraints according to use case in presented results.

### 3.1 MAXIMIZING STORAGE DENSITY AND INFERENCE EFFICIENCY (EVALUATION)

Leveraging the identification of maximal-density, accuracy-preserving MLC eNVM storage for DNN weights in Chapters 2.2 and 2.3, this section describes the system-level evaluation and implications of integrating such a memory solution with a DNN accelerator system.

#### 3.1.0 BASELINE ACCELERATOR DESIGN (NVDLA)

NVIDIA deep learning accelerator (NVDLA) is an industry-grade, open source architecture solution to accelerate DNN inference. Baseline system parameters are given in Table 3.0, and a block diagram is shown in Figure 3.1, left<sup>178</sup>. NVDLA supports CNN and FC layer execution, and all

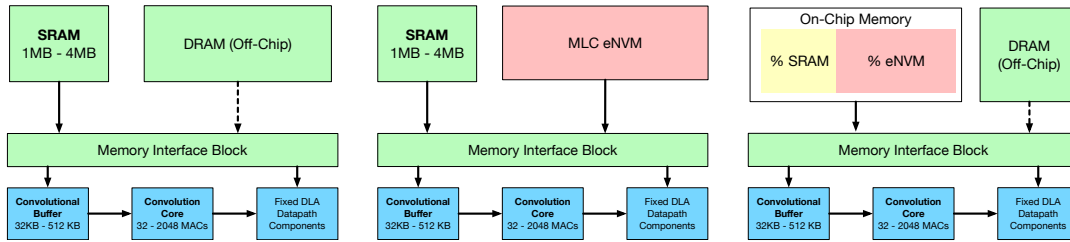


Figure 3.1: Baseline NVDLA (left) vs. Proposed Architectures (center, right) from MaxNVM, taken from<sup>183</sup>.

computation in MaxNVM evaluations is performed using this datapath. By demonstrating our proposed memory integrated with this system, we highlight the opportunity for eNVM integration with existing systems for efficient DNN inference.

By default, all DNN weights are retrieved per-layer and per-tile from off-chip DRAM before being staged in the convolutional buffer for computation. Any inputs to the network and intermediate results are read from and written to on-chip SRAM of a configurable size in the system design, and the memory interface block manages access both to the on-chip global SRAM buffer and off-chip DRAM resources. Inputs and weights may be configured as 8b or 16b width. The baseline NVDLA system optionally allows for bitmask-encoded sparse weights, and the MaxNVM work additionally considered CSR-encoded sparse weights. Additional architectural characteristics and design parameters are listed in Table 3.0 and described in more detail in<sup>178</sup>.

### 3.1.1 FULLY ON-CHIP DNN INFERENCE EVALUATION OVERVIEW

MaxNVM considers a completely self-contained inference accelerator that stores all of the weights in on-chip eNVM and does not require external DRAM, as indicated in Figure 3.1, center. This scenario would be especially important for deeply embedded applications such as IoT devices and implanted medical devices where component cost, performance, and energy constraints can be extreme, and entirely on-chip execution could additionally provide privacy benefits<sup>56,127</sup>.

To evaluate the area, energy, and performance of both CTT and RRAM on a per-model basis,



we use model optimization results to determine the number of bits required per encoding and perform fault tolerance analysis. These determine the appropriate number of bits per cell and total number of memory cells required for a given encoding without loss in classification accuracy (covered in Chapters 1.1.1, 2.3). Next, we use our extensions of NVSim to characterize memory systems that are sufficient to accommodate the size of each model under each encoding strategy. Finally, pareto-optimal points for characterized memories of four different design points are identified per model to minimize the read energy-delay-product and area. The resulting system performance is evaluated using the NVDLA performance model and compared to the performance of a baseline configuration relying on off-chip DRAM for weight storage in (Figure 3.1, left).

This study quantifies the increase in die area needed to store the considered DNNs entirely in on-chip memory while maintaining competitive performance and achieving reduced power and reduced energy per inference. The number of cycles to execute an inference per input image (frame) is computed using the NVDLA performance model with the characterized read bandwidth and read latency determined by NVSim for each eNVM array. Some amount of on-chip SRAM is retained in the NVDLA system evaluation (512KB or 2MB in our designated NVDLA configurations, Table 3.0) to manage storage of intermediate values between layers of the DNNs, as the write latency of CTT is prohibitively high (on the order of  $ms^{115}$ ) and can still be orders of magnitude higher for MLC-RRAM than SRAM write latency (e.g., 160 – 640ns).

### 3.1.2 AREA AND DYNAMIC READ ENERGY

An interface to eNVM replaces NVDLA’s interface to off-chip DRAM for DNN weights, and we demonstrate that aggressive MLC configurations of both CTT and optimistically scaled RRAM can store the sparsely-encoded, ECC protected weights of ImageNet-VGG16 (about 32MB Capacity) in  $2mm^2$  and  $1.3mm^2$ , respectively. Thus, as a result of exhaustive design space exploration and a rigorous evaluation scheme, all DNN weights can reasonably fit in on-chip, non-volatile

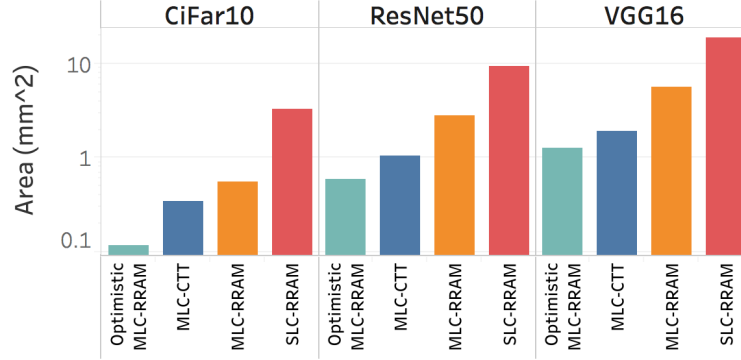
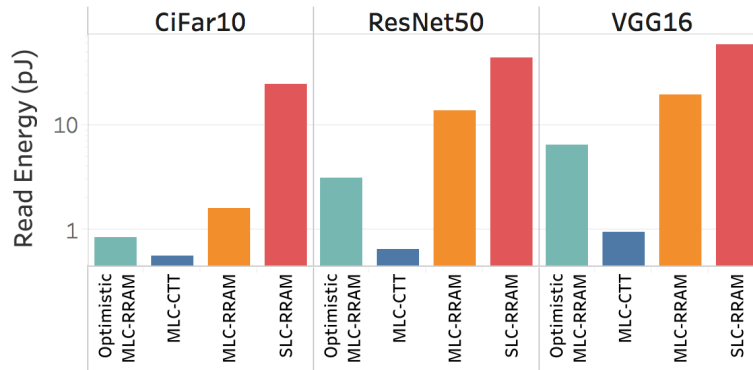


Figure 3.2: Array-Level area of various on-chip memory solutions for storing all DNN weights in eNVM, taken from <sup>183</sup>.

memory in an area equivalent to 1-2MB of SRAM capacity in modern process nodes. Even SLC-programmed RRAM fits all of ResNet50 (12MB capacity under sparse encoding) in under  $10\text{mm}^2$  (Figure 3.2). With additional model optimizations, or more relaxed error bounds, MLC configurations could be leveraged even more aggressively at some loss in DNN classification accuracy, depending on use case. This study strictly avoids loss of accuracy. Even so, these models consume reasonable area for on-chip local memory storage using MLC eNVM configurations.

Figures 3.2 and 3.3 summarizes the area and dynamic read energy per access of four distinct on-chip memory solutions. The area for each model is the read-energy-delay-product optimal point characterized by our extensions to NVSim that will hold all the DNN parameters in the most optimal encoding for each memory technology as determined in Chapter 2.3. Even relative to storing the same optimized and sparse-encoded weights in SLC-RRAM, the MLC-CTT array requires an average of  $9.6\times$  less area while maintaining competitive performance within 10% of the NVDLA baseline. Furthermore, our MLC extension of the SLC-RRAM achieves an average area benefit of over  $3.2\times$ , and the optimistically scaled MLC-RRAM (Optimistic MLC-RRAM) enables an average area benefit of  $20\times$ . The larger relative area benefit between MLC-CTT and the Optimistic MLC-RRAM for CiFar10-VGG12 (Figure 3.2) derives both from the inherent storage density of



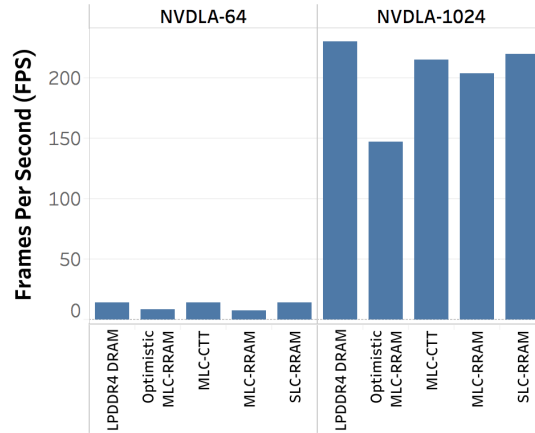
**Figure 3.3:** Array-Level read energy per access of various on-chip memory solutions for storing all DNN weights in eNVM, taken from <sup>183</sup>.

the Optimistic MLC-RRAM and the fact that more bits per cell can be safely used with the Optimistic MLC-RRAM (consistently 3 bits per cell) than with MLC-CTT (2 bits per cell) due to their respective fault characteristics and the impact on classification error, as exposed in Chapter 2.3.4.

The dynamic read energy for each of the memory proposals varies by orders of magnitude, even for equivalent-capacity characterized array (Figure 3.3). MLC-CTT is consistently lower energy per access than even the Optimistic MLC-RRAM solution by over  $4\times$ , and also tends to maintain a lower read latency and higher read bandwidth (up to 9 GB/s).

### 3.1.3 SYSTEM PERFORMANCE

Considering a real application-specific metric (frames-per-second), even the SLC consistently meets a target of 60fps, for example, even for the largest models. Though eNVM proposals do not surpass the maximal FPS possible with NVDLA (e.g., Figure 3.4), the best performance for each model consistently exceeds 60 frames per second with the NVDLA-1024 configuration, well above the image processing frame rate for standard motion pictures <sup>160</sup>. The latency overhead of MLC sensing tends to negate the effective bandwidth increase of MLC storage. This is seen in Figure 3.4; SLC-RRAM is competitive with MLC-CTT in terms of frames (i.e., inferences) processed per second. We note



**Figure 3.4:** Achievable frame rate (inferences-per-second) for baseline NVDLA (LPDDR4 weight memory) vs. Proposed Architectures from MaxNVM, taken from <sup>183</sup>.

that these performance estimates do not include the decoding overheads for sparse encodings and cluster index values. However, the overhead to reconstruct and decode weight values for both strategies will be minimal and consistent in the baseline and eNVM-based systems.

Figure 3.5 compares the average power consumption and total energy per inference for ResNet50 when operating at maximum performance. We consider two fixed NVDLA baseline datapath configurations given in Table 3.0, for which the power consumption is publicly available <sup>212</sup>, and assume additional power consumption of LPDDR4 DRAM running at 1GHz is 200mW to form a conservative bound of potential improvements via integration of on-chip eNVM <sup>28</sup>.

The NVDLA-64 design is representative of a resource-constrained DNN accelerator, and the energy consumption of memory accesses for weight fetches is *reduced by over 100x* (DRAM vs.  $1\text{mm}^2$  of on-chip MLC-CTT), resulting in overall average system power reduction of  $3.2\times$ . NVDLA-1024 provides considerably higher performance (Figure 3.4) and overall higher efficiency inference (lower dynamic energy per inference), though the average power is higher. In this scenario, we see similar trends in MLC-CTT reducing energy consumption due to weight fetching by up to  $13.6\times$ . However, due to the higher baseline power of the larger convolutional core and buffer, the total

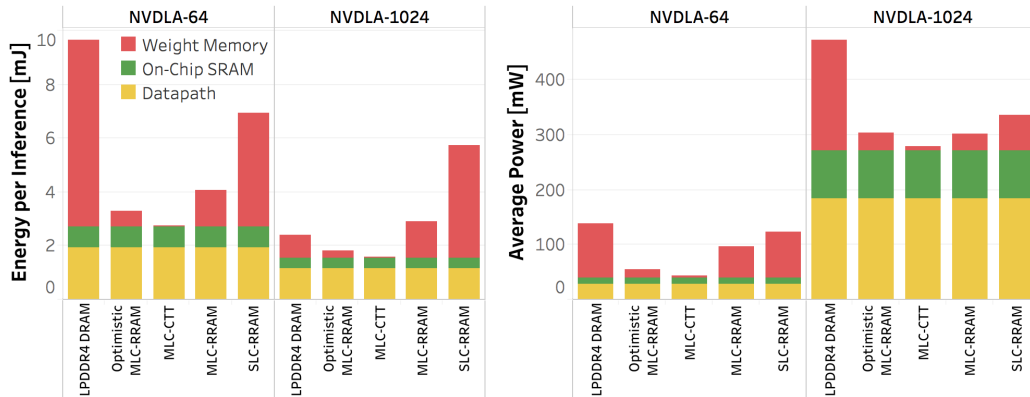


Figure 3.5: Total energy-per-inference and average system power for baseline NVDLA (LPDDR4 weight memory) vs. proposed architectures from MaxNVM, taken from <sup>183</sup>.

relative system power reduction is up to  $1.6\times$ .

Of the three evaluated MLC proposals, MLC-CTT achieves the best system performance, lowest power, and lowest energy per inference. Compared to the optimistically scaled MLC-RRAM, higher read bandwidth and lower energy per read result in 20% overall lower energy per inference for MLC-CTT for NVDLA-64. These demonstrated benefits are relative to energy and performance characteristics of convolutional models using a highly-optimized CNN accelerator datapath that maximizes re-use of fetched values, so energy reduction due to memory fetches would be increasingly beneficial in other resource-constrained contexts that exhibit less re-use of fetched parameters, as explored further in future works, for example in Sections 3.2 and 3.3. Compared to the energy usage of the NVDLA datapath and SRAM in isolation for ResNet50 inference, we see that the power for the eNVM solutions are a fraction of the the compute pipeline for the aggressive MLC proposals (either optimistically scaled MLC-RRAM or MLC-CTT based on our fabricated test chip), and the overall system power reduces by up to 40% with the added benefit of low leakage power and the ability to turn off the eNVM between inferences and retain weight values. Table 3.1 gives a summary of area, energy, and performance for optimal storage for each design point.

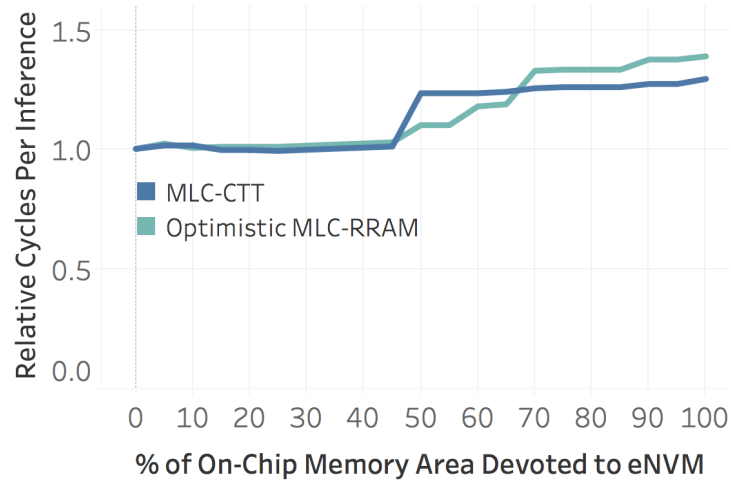
Model	Memory Tech	Encoding	BPC	[MB]	Area [ $mm^2$ ]	Read [ns]	FPS
CiFar10	Opt. MLC-RRAM	BitM+IdxSync	3	4	0.12	5.1	132
	MLC-CTT	BitMask	2	4	0.35	1.6	2286
	MLC-RRAM	BitM+IdxSync	3	4	1.3	4.9	633
	SLC-RRAM	BitMask	1	4	3.4	1.7	2967
VGG16	Opt. MLC-RRAM	CSR+ECC	3	32	1.3	4.2	102
	MLC-CTT	CSR+ECC	3	32	2.0	2.0	142
	MLC-RRAM	CSR+ECC	3	32	5.7	3.2	131
	SLC-RRAM	CSR	1	32	19.2	5.2	147
ResNet50	Opt. MLC-RRAM	BitM+IdxSync	2	12	0.6	2.1	147
	MLC-CTT	BitM+IdxSync	2	12	1.0	1.9	215
	MLC-RRAM	BitM+IdxSync	2	12	2.8	1.4	203
	SLC-RRAM	BitMask	1	12	9.6	2.5	219

**Table 3.1:** Summary of optimal storage per eNVM proposal, characterized per DNN. ‘BPC’ is the maximum number of bits per cell used across DNN layers, ‘MB’ is the approximate capacity, ‘FPS’ is the maximum frames per second for NVDLA-1024, and ‘Read’ is the Read Latency of each eNVM array in ns, taken from <sup>183</sup>.

### 3.1.4 REDUCING OFF-CHIP DRAM ACCESS FOR LARGER MODEL SIZES

New DNNs tend to improve accuracy by increasing model size <sup>192</sup>. To understand what happens when a model does not fit in on-chip eNVM, we consider a hybrid solution, shown in Figure 3.1, right. We allocate a fixed on-chip area allowance for all memory (SRAM and eNVM), and weights and activations that do not fit on-chip are stored and fetched from DRAM.

We choose an on-chip memory area budget of  $1mm^2$ , enough to accommodate about 1MB of SRAM under various NVSim optimization targets. With eNVM, smaller DNNs fit within this area constraint. The results in Figure 3.6 are for the largest DNN, ImageNet-VGG16, and highlight the performance impact of incorporating varying amounts of on-chip space to eNVM, which effectively increases on-chip memory capacity in a constrained environment. Note that the eNVM is not used as cache for DRAM; on-chip CTT/RRAM and DRAM store mutually exclusive sets of model weights and both are fed directly into the accelerator’s datapath. By partitioning fixed on-chip memory area between SRAM for intermediate storage and MLC eNVM for weight storage, we reduce costly DRAM accesses while drastically increasing total on-chip memory capacity. For each possible



**Figure 3.6:** Partitioning a fixed on-chip area between SRAM and a highly-dense eNVM solution is strictly beneficial until the SRAM capacity becomes insufficient to store intermediate results, which sharply degrades inference performance (e.g., at 45-60% of area devoted to eNVM), taken from <sup>183</sup>.

area partition, we use NVSim to characterize the maximal capacity and minimal read latency and energy within that area constraint and select pareto-optimal points.

Depending on how the on-chip memory area is partitioned, inference execution could be bottlenecked by weight retrieval from DRAM for weights not in the eNVM, or by read/write traffic of intermediates that do not fit in SRAM. We extend the NVDLA performance model to selectively read certain weights from eNVM rather than DRAM, and maximize eNVM benefits by greedily selecting the weights of otherwise DRAM-bottlenecked layers to store first. There is some initial benefit from alleviating the weight retrieval DRAM bottleneck when some amount of eNVM is allotted, because the number of DRAM accesses reduces and the energy to access either eNVM is orders of magnitude lower than accessing DRAM. However, performance sharply degrades when on-chip SRAM storage can no longer hold the working set of intermediate values to feed the convolutional core and execution gets increasingly bottlenecked on writing to and fetching intermediate values from DRAM. The write characteristics of both alternatives are not sufficient to buffer inter-

mediate values or to re-write weights during inference, as explored in Section 3.1.6.

### 3.1.5 INTERMITTENT OPERATION; BENEFITS OF NON-VOLATILITY

As yet, we have demonstrated incredible capabilities in terms of storage density and energy efficiency of MLC eNVM storage for on-chip DNN inference. However, we have not explicitly taken advantage of the non-volatility of the proposed embedded memories, and this property becomes extremely beneficial in intermittent operation use cases. eNVM arrays effectively have a huge additional energy advantage because there is no leakage power (even leakage power for sensing is at least  $100\times$  less than SRAM leakage according to NVSim). Though we have thus far only examined the significant benefits in terms of read dynamic energy during inference, another key advantage of both RRAM and CTT memory is that they are non-volatile, and thus can be powered off when not in use and need not incur any leakage power between inferences. This makes these design points and other eNVM technologies a particularly attractive solution for edge devices, as weights can be stored locally, off-line, and without any leakage power cost between inferences, in stark contrast to the constant leakage power required by SRAM or power-hungry DRAM refresh.

Depending on how frequently inferences occur (i.e., required frame rate for an image processing task<sup>264,67</sup>), eNVMs have an inherent relative benefit by virtue of not needing to reload weight values when powered on or, alternatively, keeping DRAM and SRAM powered to avoid this cost. Figure 3.7 summarizes how the average energy per inference of the evaluated MLC eNVM solutions compares to the baseline system either remaining fully powered to retain DNN weights (“DRAM always on”) vs. waking up the system for each inference (“DRAM wake up”). Each time the baseline system is woken up, there is an energy overhead to load the entire DNN’s weights from main memory into DRAM and an increased inference latency to load the entire first DNN layer before the inference task can begin computation, while these costs are not applicable to eNVM technologies that retain their values when powered off between inferences.



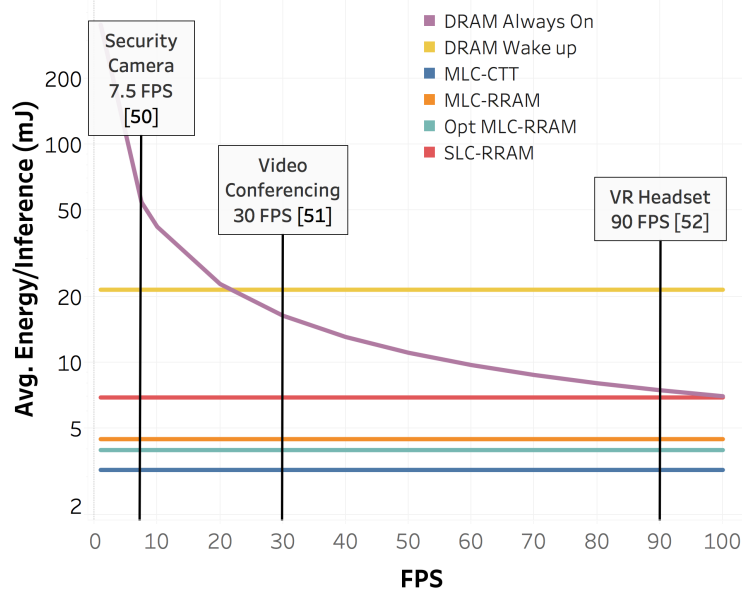


Figure 3.7: Average Energy-per-Inference for baseline NVDLA vs. proposed architectures at varying frame rates; for typical frame rates, it is strictly beneficial to leverage eNVM storage and wake the system per-inference, taken from <sup>183</sup>.

If the DRAM power stays constant while the frequency of inferences increases (i.e., frames processed per second, FPS), the average energy per inference decreases as the system spends less time idle. Conversely, if the system is woken up for each inference, the average energy per inference is constant so long as the frequency (FPS) can be met by the NVDLA-1024 system, which Figure 3.4 showed is clearly possible for the range of FPS considered. For frame rates below 22 FPS, such as would be relevant for object detection tasks for security camera systems<sup>82</sup>, it is more energy efficient to wake up the baseline system per inference, but our proposed MLC eNVM solutions maintain a relative  $5.3\times$  (MLC-RRAM) to  $7.5\times$  (MLC-CTT) reduced energy per inference.

Energy savings are similar for the typical range of frame rates used for video conferencing and other standard image processing tasks (e.g., 30 FPS<sup>264</sup>). Though the average energy per inference of the always-on baseline system approaches the performance of MLC eNVM alternatives at higher frame rates, we note that even operating at the FPS mandated to support virtual reality headsets

Model	Memory Technology	Approximate Total Write Time
CiFar10	OPTMLCRRAM	13ms
	MLC CTT	2.6 <b>minutes</b>
	MLC RRAM	33ms
	SLC RRAM	3ms
ResNet50	OPTMLCRRAM	117ms
	MLC CTT	15.7 <b>minutes</b>
	MLC RRAM	94ms
	SLC RRAM	4.7ms
VGG16	OPTMLCRRAM	254ms
	MLC CTT	12.2 <b>minutes</b>
	MLC RRAM	636ms
	SLC RRAM	23ms

**Table 3.2:** Optimistic total time to write all DNN weights per eNVM solution, taken from <sup>183</sup>.

(at least 90 FPS<sup>67</sup>), MLC eNVM proposals achieve  $1.7\times$  (MLC-RRAM) to  $2.5\times$  (MLC-CTT) lower energy per inference. Thus, optimized MLC eNVM solutions are particularly compelling for applications with lower frame-rate requirements, and these benefits would be exaggerated for deeply embedded systems with less frequent wake-ups.

### 3.1.6 WRITE LATENCY

Table 3.2 gives an approximate total time to write each model’s entire set of weights to each eNVM design point, simulating a full weight update after retraining, perhaps, based on best-case-scenario write latency from previous work (e.g., as extracted in the survey presented in Section 1.2). Depending on a given application space and resource constraints, waiting on the order of minutes may or may not be reasonable when re-writing of DNN weights is required. However, in sensor nodes, mobile devices, and other constraint-driven devices in which DNN inference is a key workload, periodic down-time for synchronization and charging may be permissible.

The asymmetric costs of eNVM writes and reads are further discussed, and another key approach is to identify a subset of application data that will be less-frequently-written, but frequently read, to maximize the benefits of on-chip eNVM storage and data reuse. This general strategy is probed

in order to optimize support for multi-task inference in both image-processing (Section 3.2) and natural language processing (Section 3.3) domains in follow-on works to MaxNVM.

### 3.1.7 MAXNVM SUMMARY

Table 3.1 provides the energy, area, and performance per model. The results emphasize the need for a comprehensive co-design methodology. Considering the storage density of MLC and energy advantages due to non-volatility, even unoptimized DNNs or other reasonably fault-tolerant applications may leverage these memories. However, choosing the optimal encoding strategy, MLC configuration, and whether to include error mitigation techniques varies between models and eNVM technologies. Thus, a rigorous evaluation of the design space per DNN, as our methodology effectively executes, maximizes efficiency gains.

## 3.2 MEMTI: OPTIMIZING ENVM FOR VISUAL MULTI-TASK INFERENCE

Embedded non-volatile memories (eNVMs) are under active consideration to provide higher density than SRAM and ameliorate the need for power-hungry DRAM storage. However, the benefits of eNVMs come at the cost of larger write energy and write latency, as exposed and explored in Section 3.1. Moreover, limited eNVM write endurance is an obstacle to the adoption of certain technologies if DNN parameter values require frequent updates, as discussed in Chapter 1. For instance, embedded devices for robotics or augmented reality applications often required a combination of multiple inference tasks, including image classification, object detection, and action recognition. These cases highlight the need for scalable solutions that can flexibly accommodate DNN parameters for multiple tasks.

MEMTI<sup>59</sup> presented a DNN model and memory co-design solution that leverages a machine learning technique to reduce eNVM writes, while enabling systems to efficiently perform multiple

inference tasks. Maximizing the reuse of the learned parameters across different DNN-dependent vision tasks without re-training enforces the assumption of infrequent writes: parameters shared by multiple tasks are trained and written only once, and therefore are highly suitable for eNVM storage; in contrast, the remaining parameters can be re-trained to accommodate new inference tasks, and stored in SRAM. In addition to the storage density benefits, we evaluate how the process of re-training specific parameters can be used to recover from accuracy loss due to the adoption of denser, fault-prone multi-level eNVM storage. To the best of our knowledge, this was the first attempt to co-design non-volatile memories and DNN models through multi-task learning approaches.

### 3.2.0 TRANSFER LEARNING FOR DNN AND MEMORY CO-DESIGN

Generalizing deep learning architectures to enable different application domains and more varied inference tasks serves as a way of supporting more powerful and versatile models. For example, prior work in the machine learning community<sup>112</sup> combined several building blocks for translation, speech, and visual inference that can be trained on all desired tasks simultaneously or on each task separately. In either case, however, introducing new inference tasks would require updating the entire set of model parameters. Other works have leveraged the concept of transfer learning to improve the performance of a single DNN on different datasets<sup>260</sup>. These approaches are based on the observation that many visual inference tasks share low-level features, such as edge and shape detection, in the front-end layers, and become more task-specific as the computation moves closer to the classification layers. However, in order to preserve inference accuracy, transfer learning approaches either share only a limited number of front-end layers or fine-tune parameters by re-training the transferred features from one inference task to another. A recent proposal applies transfer learning to create a synthesizable fixed-parameter feature extractor<sup>244</sup>. However, hard-wiring the feature extractor in logic prevents any future capability of fine-tuning the parameters, limiting the amount of cross-task weight sharing and flexibility.

Dataset	cifar100	aircraft	daimlerpedcls	gtsrb	ucf101
# images	50K	7K	30K	40K	9K
# classes	100	100	2	43	101
Full model	72.78%	40.98%	99.88%	99.97%	73.77%
Only adapters	79.61%	43.8%	99.51%	99.94%	73.16%
Parameters overhead	10.4%	10.4%	10.1%	10.2%	10.4%
Training speed-up	4×	2×	1.35×	3.23×	4.74×

**Table 3.3:** Summary of dataset characteristics, and maximum training accuracy for the model trained entirely from scratch on each dataset or using residual adapters on a pre-trained network. Pre-trained shared parameters on ImageNet, with 67.65% accuracy, taken from<sup>59</sup>.

While all these techniques enable a single DNN model to perform different inference tasks, they still require updating a considerable portion of parameters to achieve maximum adaptation. MEMTI pursued a specific transfer learning technique for which the learned parameters can be generalized across multiple vision inference tasks by maximizing DNN parameter reuse and enabling efficient inference on embedded devices. The high degree of DNN parameters reuse reduces memory traffic requirements, which makes non-volatile memories a compelling solution for retaining shared parameters on-chip without incurring costs associated with frequent memory writes.

### 3.2.1 MULTI-TASK LEARNING MODEL

Our design is based on the DNN architecture presented in<sup>195</sup>, which uses residual adapter modules as a way to parameterize a generic ResNet network. These parametric modules are themselves residual blocks which use  $1 \times 1$  filters and skip connection. In this setting, the number of domain-specific parameters, which comprises adapter filters, batch normalization, and fully-connected classifier parameters, can be reduced to roughly 10% of the total model size.

For our experiments, we integrate the residual adapter modules in a ResNet26 network. The baseline network is pre-trained on ImageNet, which is standard practice in transfer learning and model fine-tuning techniques. The pre-trained version for ImageNet achieves top-1 accuracy of 67.65%. The ResNet26 weight parameters obtained during pre-training are the backbone of this

multi-task inference system as they are reused for running inference on any additional visual task. The degree of adaptation is tested against five datasets selected to be representative of popular image processing tasks including classification (cifar100, aircraft), object detection (German Traffic Signs, Daimler pedestrian classification), and action recognition (UCF101 Dynamic Images).

Table 3.3 summarizes the best accuracy in the case of the model being either trained entirely from scratch or only for the task-specific parameters. As anticipated, for all datasets, the adapters overhead is around 10%. The accuracy of the network trained using adapters is always better than or comparable to training the entire network independently for each dataset. In addition, we observe that the modified model converges to the best accuracy in fewer training epochs, which results in training speed-up reported in Table 3.3.

### 3.2.2 NON-VOLATILE MEMORY TECHNOLOGIES FOR MULTI-TASK INFERENCE

The landscape of non-volatile memories includes a wide range of emerging technologies<sup>56,31</sup>, as highlighted in Chapter 1.2. These memories are generally characterized by high energy efficiency and high storage density, which can be further increased by programming multiple levels in a single cell. MEMTI focussed on a specific eNVM implementation (RRAM). Various implementations such as phase-change memories (PCM), embedded Flash (eFlash), or ferroelectric memories (FeRAM) can also be used for MLC storage. On the other hand, STT magnetic memories (STT-MRAM), while having the best write and read performance<sup>56</sup>, are not as suitable because compelling MLC implementations with comparable density have not been demonstrated to date, as described in Chapter 1.2.

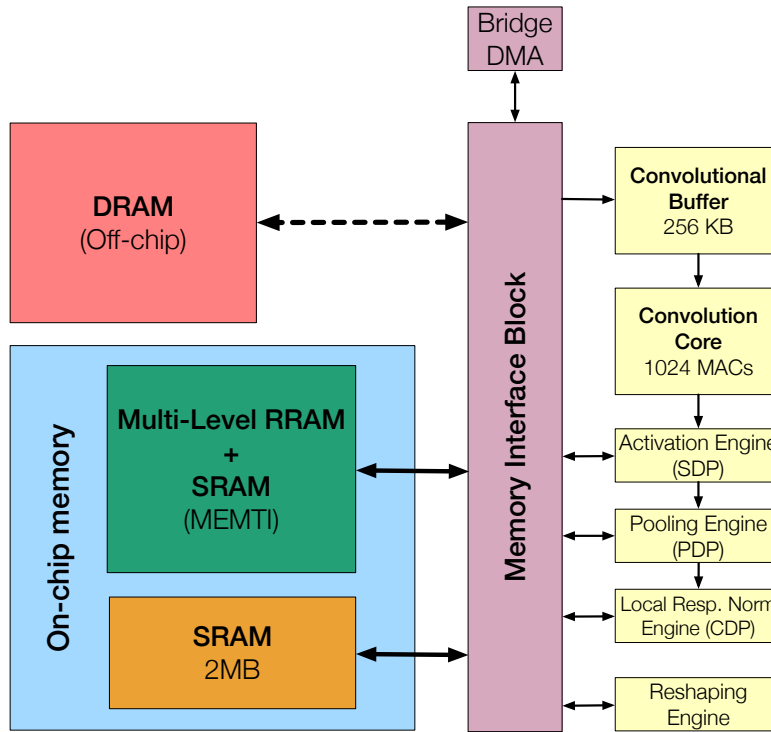


Figure 3.8: NVDLA system diagram, with additional optional interface to multi-level-cell RRAM for on-chip weight storage in a multi-task image processing setting, taken from <sup>59</sup>.

### 3.2.3 MEMORY SYSTEM OVERVIEW

In order to complement the properties of residual adapter networks and dense MLC RRAM storage, we propose MEMTI, a Memory system for Efficient Multi-Task Inference. A large fraction of the parameters in a residual adapter network is shared across multiple applications, and can be efficiently stored in MLC RRAM, while application-specific parameters can be stored in SRAM. By partitioning on-chip memory area between RRAM and SRAM, as shown in Figure 3.8, we achieve the best trade-off between storage density for the shared parameters and fast and energy-efficient updates for the task-specific parameters (SRAM). Off-chip DRAM stores multiple sets of task-specific parameters. The resulting memory hierarchy takes advantage of RRAM non-volatility for intermittent operation by powering down the system between inferences. In this scenario, only a small

portion of the model parameters must be written to SRAM during power up or task switching. Moreover, storing task-specific parameters in more robust memory allows us to mask MLC RRAM faults via retraining.

#### 3.2.4 EVALUATION FRAMEWORK

To evaluate the proposed memory architecture, we quantify the impact of RRAM fault characteristics and MLC encoding on inference accuracy, memory architecture and array properties, and system-level performance. The fault model is derived from previous work integrating eNVM device and circuit-level fault characteristics with DNNs evaluation frameworks to allow for extensive memory and DNN co-design space exploration<sup>60,183</sup>. We model MLC RRAM faults based on stochastic level distribution which arise from the random nature of memristors programming. When multiple levels are programmed in a single RRAM cell, the distributions overlap can be used to extrapolate the read fault probabilities for each level, as described in Chapter 2.1.

To evaluate the DNN accuracy under different storage schemes, we use a version of the residual adapter architecture implemented in PyTorch<sup>181</sup>. We modified the existing implementation by adding transform functions to manipulate the values according to different multi-level encoding and compression techniques. The resulting error map is then integrated in a DNN evaluation framework to simulate the impact MLC RRAM faults on inference accuracy, an extension of<sup>193</sup>. The level distributions are extrapolated from measured MLC RRAM characteristics<sup>268</sup>. Based on the MLC RRAM fault probabilities, we sample the value of the stored weight matrix based on a predefined multi-level encoding configuration to evaluate the impact on the model accuracy. In addition, we improve the fault model by including the effects of the sensing circuitry on the read error probability, also described in Section 2.1.

The corresponding framework is used to drive the design towards a solution that would minimize the on-chip memory footprint without increasing the inference error. After identifying the



best MLC encoding without loss in accuracy, we perform a memory design space exploration using a modified version of NVSim<sup>62</sup>, as in MaxNVM<sup>183</sup>. The impact of off-chip memory accesses is quantified using a model of LPDDR4 DRAM. Finally, we integrate the resulting memory hierarchy with a proven CNN accelerator architecture developed by NVIDIA (NVDLA), which, combined with NVSim results and DRAM estimates, allows us to evaluate the system energy and performance for different application scenarios.

### 3.2.5 SYSTEM-LEVEL CHARACTERIZATION

As shown in Figure 3.8, the baseline NVDLA system in this particular evaluation comprises a convolutional core with 1024 MAC units fed by a convolutional buffer and supplemented by several additional computational units for pooling and data transformation operations. NVDLA also supports a memory interface block and DMA that fetches model weights per layer from off-chip DRAM and leverages on-chip SRAM (2MB) to buffer inputs and intermediate results of computation between layers in the DNN. We flexibly integrate MEMTI with the NVDLA performance model as an additional memory interface to leverage for model weights, either in addition to or in place of fetching parameters from off-chip DRAM, reusing the more flexible performance model interface put forth in MaxNVM<sup>183</sup>.

For a competitive multi-task inference application, we evaluate the performance, energy, and area for the NVDLA system when executing three inferences per input frame using three representative visual tasks, namely image classification (cifar100), object detection (gtsrb), and action recognition (UCF101). This series of tasks computed per input frame would be appropriate, for example, for an autonomous vehicle or a drone processing sensory data to understand and interact with the surrounding environment. For this application, we set the target operating frequency to 30 frames per second (FPS), or 90 inference tasks per second, which satisfies a breadth of applications<sup>160</sup>.

### 3.2.6 BASELINE, DRAM-RELIANT EVALUATION

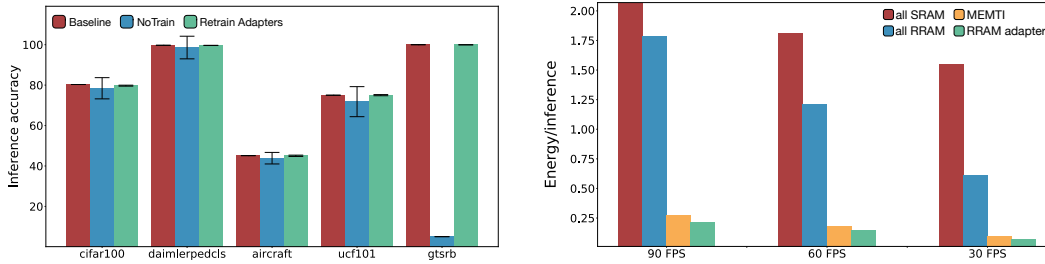
As a baseline case, we assume that the accelerator is continuously processing input frames and fetching both shared and task-specific parameters from off-chip DRAM for each layer’s computation. This DRAM-only, always-on operation consumes a total power of 493mW and a peak performance of 749FPS. The estimated power includes datapath, DRAM refresh, and on-chip SRAM leakage. At this stage, the on-chip SRAM is exclusively used for storing the input features and intermediate values. We compute the energy per frame at peak performance to be 1.17mJ.

### 3.2.7 PROVISIONING FOR ON-CHIP SRAM

We first show the case in which we allocate enough on-chip SRAM to store the entire set of parameters for a single task. For a system designed to run a single inference task, having the option of storing all the network parameters on chip allows to reduce the memory access energy by  $40\times$ . This result demonstrates the strong impact of off-chip memory access on the entire system energy. These high energy savings are however impractical to realize with SRAM since for a 22nm technology node, we estimate a total area of at least  $6.55\text{mm}^2$ , over  $10\times$  larger than the on-chip SRAM area for the baseline NVDLA configuration. Moreover, when we consider the full system energy in the multi-task scenario described above, the periodic parameter updates and SRAM leakage power reduce the energy savings to  $0.64\times$ .

### 3.2.8 IMPROVING STORAGE DENSITY WITH ENVM

As a first step towards reducing both power consumption and memory footprint we consider storing the weights on-chip using MLC RRAM. Without applying any application-level optimization, a multi-task operation would still require updating all the model parameters stored in RRAM when switching to different inference tasks. While this type of operation has a clear downside dictated



**Figure 3.9:** Accuracy Results for MEMTI (left) and Energy vs FPS for the different design configurations normalized to the DRAM baseline (right). The power savings of the RRAM-based design for higher frame rates is exacerbated by the frequent RRAM writes, making the design less efficient than the DRAM-based baseline. On the other hand, the energy per inference for MEMTI and RRAM adapters is strictly better than the baseline, taken from <sup>59</sup>.

by the RRAM write endurance, our system level evaluation exposes other limitations. Although the overall leakage power and on-chip memory area can be reduced to 298mW and 0.347mm<sup>2</sup> respectively, the energy per inference increases to 14.58mJ. This is caused by the combined effect of RRAM write energy and latency. These examples highlight the need for a solution capable of balancing on-chip memory density and write performance.

### 3.2.9 MEMTI FOR INTERMITTENT OPERATION

MEMTI reduces memory write costs by replacing the RRAM portion storing the task-specific parameters with SRAM. This is possible thanks to the adoption of residual adapters in the DNN network. For the resulting design, the total system power is 362mW and the energy per inference is 1.56mJ, which is comparable with the baseline. Isolating the costs associated with weight storage compounds the benefits introduced by MEMTI: power consumption decreases by 3.9 $\times$ , with an area overhead of 1.16mm<sup>2</sup>. The resulting peak performance is 429FPS, well above application requirements. Intermittent operation is where MEMTI truly stands out by taking advantage of non-volatility. In this scenario, we fix the operation at 30FPS and power down the system between frames, which reduces the energy per inference by 10.65 $\times$ , as highlighted in Figure 3.9.

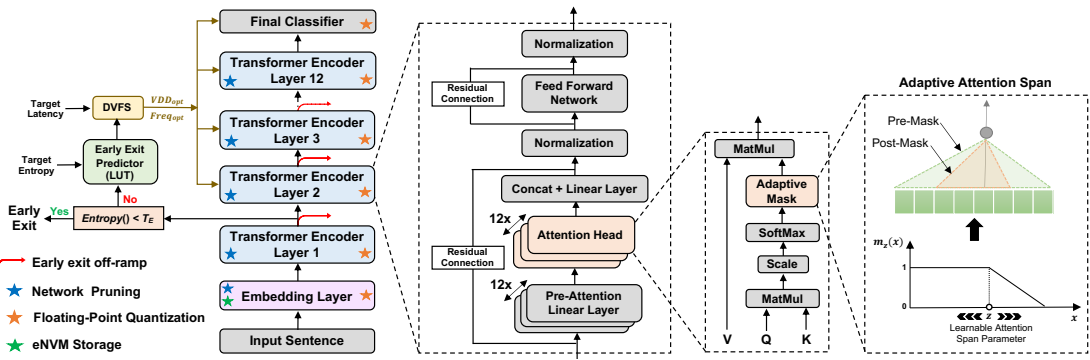
	Power [mW]	Max FPS	WMem Area [mm <sup>2</sup> ]	Saved energy	On-chip SRAM	On-chip RRAM
all DRAM	493	749	–	1×	2MB	–
all SRAM	634	485	6.55	0.64×	8.5MB	–
all RRAM	298	47	0.347	1.62×	2MB	2.2MB
MEMTI	344	429	1.16	10.65×	2.7MB	2MB
RRAM adapters	301	396	1	13.6×	2MB	4MB

**Table 3.4:** Summary of power, performance and area for the four design configurations considered in this work. The energy savings are normalized to the all DRAM configuration for the intermittent multi-task operation over three tasks running at 30 FPS. On-chip RRAM shows the physical memory capacity (i.e. number of cells), taken from<sup>59</sup>.

### 3.2.10 A RRAM-BASED SPECIALIZED DESIGN

Alternatively, we consider the case specifically tailored for the three chosen tasks. Using residual adapters still reduces the weights storage requirements by a factor of 2.3×. However, based on previous results, we use SLC RRAM to store the task-specific parameters and preserve inference accuracy. Therefore, we store the entire set of parameters in MLC and SLC RRAM. Removing the need for additional SRAM reduces the overall power to 343mW, and the area to 1mm<sup>2</sup>. Allocating enough memory for storing all the parameters on chip increases the energy savings compared to the baseline by 13.6×, making this design the most area and energy efficient. Nonetheless, MEMTI maintains the advantage in terms of flexibility and robustness to RRAM errors thanks to ease of reprogrammability for the task-specific parameters, for which the memory capacity is determined only by the network structure and therefore is independent from the breadth of tasks considered in a specific application.

Figure 3.9, right, shows the relationship between FPS and energy per inference normalized to the DRAM case. The all SRAM and all RRAM configurations are heavily penalized by the inability of efficiently implement a multi-task inference system. On the other hand, a co-design of the memory and DNN model using residual adapters shows much higher energy savings compared to the baseline. Table 3.4 summarizes the results at 30 FPS for the different configuration cases.



**Figure 3.10:** EdgeBERT design elements, a comprehensive codesign intersected with the memory density and efficiency considerations of MaxNVM to improve storage density and enable clever on-chip weight storage of ALBERT shared weights, in conjunction with several interdependent algorithmic optimizations and strategic use of dynamic voltage and frequency scaling (DVFS), taken from <sup>226</sup>.

### 3.3 EDGEBERT: OPTIMIZATIONS FOR MULTI-TASK NLP INFERENCE

Transformer-based language models such as BERT provide significant accuracy improvement to a multitude of natural language processing (NLP) tasks <sup>126</sup>. However, their hefty computational and memory demands make them challenging to deploy to resource-constrained edge platforms with strict latency requirements. The EdgeBERT project <sup>226</sup> proposed and evaluated an in-depth algorithm-hardware co-design for latency-aware energy optimizations for multi-task NLP. EdgeBERT employs several optimizations with implications across the computing stack, summarized in Figure 3.10, including entropy-based early exit predication in order to perform dynamic voltage-frequency scaling (DVFS), at a sentence granularity, for minimal energy consumption while adhering to a prescribed target latency. Computation and memory footprint overheads are further alleviated by employing a calibrated combination of adaptive attention span, selective network pruning, and floating-point quantization, as a result of a reimagining of the data format, pruning, and quantization in previous presented studies with the particular characteristics and optimization goals of NLP inference tasks using the ALBERT, transformer-based DNN model <sup>126</sup>.

Furthermore, in order to maximize the synergistic benefits of these algorithms in always-on and

intermediate edge computing settings, EdgeBERT specialized a 12nm scalable hardware accelerator system, integrating a fast-switching low-dropout voltage regulator (LDO), an all-digital phase-locked loop (ADPLL), as well as, high-density embedded non-volatile memories (eNVMs) wherein the sparse floating-point bit encodings of the shared multi-task parameters are carefully stored. Altogether, latency-aware multi-task NLP inference acceleration on the EdgeBERT hardware system generates up to  $7\times$ ,  $2.5\times$ , and  $53\times$  lower energy compared to the conventional inference without early stopping, the latency-unbounded early exit approach, and CUDA adaptations on an Nvidia Jetson Tegra X2 mobile GPU, respectively.

### 3.3.0 CO-DESIGNING MLC STORAGE FOR EDGE NLP INFERENCE

The ALBERT model’s hefty storage requirements comprise two different classes of parameters: (1) task-specific encoder weights, and (2) word embedding parameters. Word embeddings are deliberately fixed during fine-tuning and reused across different NLP tasks, making them a prime target for dense, MLC eNVM storage with less frequent writes. The EdgeBERT methodology sought to avoid the energy and latency costs of reloading the word embeddings from off-chip memory for different tasks by storing these shared parameters in embedded non-volatile memories (eNVMs). eNVM storage also enables energy-efficient intermittent computing because the embedding weights will be retained if and when the system-on-chip powers off between inferences. However, despite their compelling storage density and read characteristics, eNVMs exhibit two main drawbacks: potentially high write cost and decreased reliability<sup>183</sup>.

Fortunately, the word embeddings are acting as read-only parameters on-chip, which makes them highly suitable for eNVM storage, but previous work highlights the need to study the impacts of faulty, highly-dense RRAM storage on DNN task accuracy<sup>193,60,183,59</sup>. On the other hand, encoder weights need to be updated when switching across different NLP tasks. To prevent the energy and latency degradation that would follow from updating the encoder weight values in eNVMs, we map

	SLC		MLC <sub>2</sub>		MLC <sub>3</sub>	
	mean	min	mean	min	mean	min
MNLI	85.44	85.44	85.44	85.44	85.42	85.25
QQP	90.77	90.77	90.77	90.77	90.75	90.61
SST-2	92.32	92.32	92.32	92.32	91.86	90.83
QNLI	89.53	89.53	89.53	89.53	<b>88.32</b>	<b>53.43</b>
Area Density ( $mm^2/MB$ )	0.28		0.08		0.04	
Read Latency ( $ns$ )	1.21		1.54		2.96	

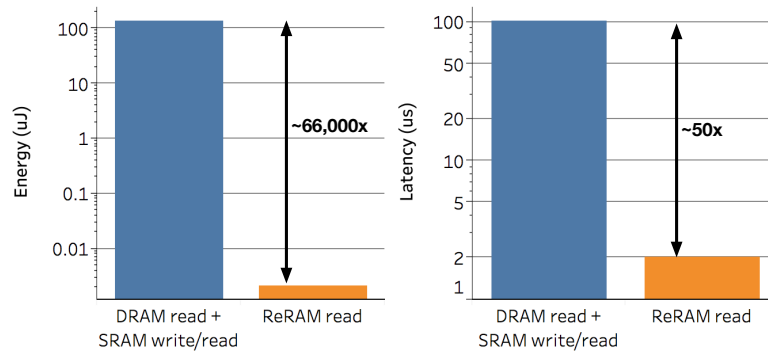
**Table 3.5:** Results of fault injection simulations impact of RRAM embedding storage on overall task accuracy. SLC=single-level cell (1 bit per cell). MLC<sub>2</sub>= 2 bits per cell. MLC<sub>3</sub> = 3 bits per cell, adapted from<sup>226</sup>.

the natural partition of shared and task-specific parameters to eNVMs and SRAMs, respectively<sup>59</sup>.

This work specifically considers dense, energy-efficient Resistive RAM (RRAM) arrays as an on-chip storage solution for shared embedding parameters (see Chapter 1.2.0). RRAMs arose as a compelling memory solution for this use case for their relative maturity and demonstrated read characteristics. There is a larger design space of opportunities to be explored with other emerging MLC-capable NVMs, but it is difficult to quickly and effectively identify eNVM solutions for a particular application and system setting, which is a research issue addressed in Chapter 4.

EdgeBERT evaluates the robustness of storing the 8-bit quantized word embeddings in MLC RRAM storage. In order to quantify the trade-offs between storage density and task accuracy, we use cell characteristics of 28nm RRAM programmed with varying number of bits per cell<sup>193</sup>, and evaluate 100 fault injection trials per storage configuration to identify robust eNVM storage solutions. Resulting per-task accuracy and memory array characteristics are summarized in Table 3.5.

After pruning, we store non-zero compressed embedding weights using a bitmask-style sparse encoding. Previous work demonstrated that DNN weight bitmask values are vulnerable to MLC faults, so the bitmask is protectively stored in lower-risk SLC devices, while we experiment with MLC storage for the non-zero data values<sup>183</sup>. Table 3.5 uncovers exceptional resilience to storing word embeddings in MLC RRAM. Across many fault injection trials, we observe that MLC<sub>2</sub>



**Figure 3.11:** Replacing accesses to off-chip DRAM for word embedding storage with embedded RRAM drastically reduces both the memory energy (left) and latency (right) across inference tasks, taken from <sup>226</sup>.

(RRAM programmed at 2 bits-per-cell) does not degrade accuracy across multiple tasks, while MLC<sub>3</sub> exhibits potentially catastrophic degradation in minimum accuracy and an appreciable decline in average accuracy for the QNLI task. Based on this observation, the EdgeBERT accelerator system leverages MLC<sub>2</sub> ReRAMs for word embedding storage.

### 3.3.1 RESULTS AND DISCUSSION

Fig. 3.11 illustrates the immense gains of leveraging an MLC eNVM configuration during single-batch inference after SoC power-on. In EdgeBERT, ALBERT embeddings would only need to be read from the integrated ReRAM buffers due to being statically pre-loaded. The conventional operation dictates reading the embedding weights from off-chip DRAM, then writing them to dedicated on-chip volatile SRAM memories so they can be reused for future token identifications. The EdgeBERT approach enforces a latency and energy advantage that is, respectively, 50× and 66,000× greater than the overhead costs in the conventional operation. The non-volatility of this embedded storage means that these benefits can further scale with the frequency of power cycles.

Though EdgeBERT conducted a more limited design space exploration than MaxNVM with respect to eNVM storage solutions, EdgeBERT was able to quickly and efficiently leverage the in-



tutions and methods of MaxNVM to identify and evaluate MLC eNVM array configurations that satisfied the application and system requirements.

*Even in a world of accelerating change, it is still difficult to convince people that new ways of doing things can be better and cheaper.*

Grace Hopper, "Computers in the Navy", 1976

# 4

## NVMEExplorer: Cross-Stack Memory Design and Optimization

THE WIDE ADOPTION OF data-intensive algorithms to tackle today's computational problems introduces new challenges in designing efficient computing systems to support these applications. Hardware specialization has shown potential in supporting state-of-the-art machine learning and

graph analytics across several computing platforms. However, data movement remains a major performance and energy bottleneck. As repeated memory accesses to off-chip DRAM impose an dominating energy cost, embedded memory systems demand increased on-chip storage density and energy efficiency beyond what is currently possible with SRAM.

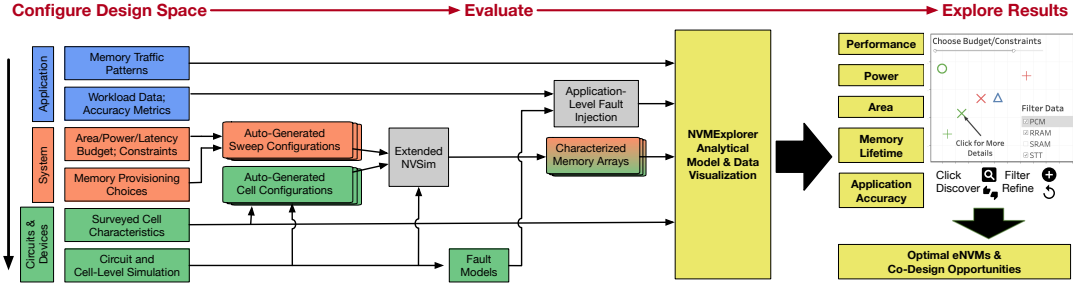
In recent years, CMOS-compatible, embedded nonvolatile memory (eNVM) research has transitioned from articles and technical reports to manufacturing flows and product lines. These technologies hold incredible promise toward overcoming the memory wall problem. For example, one approach inspired by these new technologies combines the advantages of highly specialized architectures with the benefits of non-volatile memories by leveraging analog compute capabilities<sup>42,201,215,50</sup>. On the other hand, the need for optimized on-chip storage solutions and memory innovation applies both to specialized hardware accelerators and for general-purpose CPU systems as well. More broadly, prior works have unveiled incredible potential improvements in storage density and energy efficiency by employing eNVMs across various architecture domains<sup>84,102,183</sup>. With many publications showcasing the benefits of eNVM storage technologies, it is critical for system designers to be able to explore their varying capabilities and empower efficient future on-chip storage. Unfortunately, the architecture and broader research community lacks a holistic tool to quantify the system and application-level implications of memory cell technologies and to make informed decisions while navigating the vast eNVM design space.

In the past five years, consistent interest in RRAM and STT was accompanied by emerging solutions with different physical properties such as FeFET-based memories, as discussed in Chapter 1.2. Each published example offers compelling and distinct trade-offs in terms of read and write characteristics, storage density, and reliability. In addition, the space of eNVM technologies is constantly evolving with certain technologies moving out of fashion or into production. Given the fluidity and complexity of this design space, application experts and system designers need to be able to evaluate which cell technologies are most likely to provide better efficiency, higher storage density, or im-

improvements on other key metrics in the context of different computing demands. Similarly, device designers and memory architects need high-level guidance to co-design their innovations toward more practical and maximally beneficial future, heterogeneous memory systems.

NVMEExplorer is an end-to-end design space exploration framework that addresses key cross-stack design questions and reveals future opportunities across eNVM technologies under realistic system-level constraints, while providing a flexible interface to empower further investigations<sup>185</sup>. For example, a user can specify constraints such as target application latency, area budget, and memory access characteristics to identify a subset of eNVM proposals which are best matched for those design goals. Furthermore, the output of this framework can be analyzed to identify the limiting cell-level parameters and array characteristics for each eNVM. To enable ease-of-use and iterative studies, NVMEExplorer also provides a web-based, interactive data visualizations for analyzing cell-level properties, array-level characteristics, and application-level impacts across eNVM technologies. Many of these data visualizations were made publicly available, and underlying evaluation tools are open-source<sup>185</sup>.

After describing NVMEExplorer, this chapter briefly presents a related project highlighting the potential impact of effective, accessible data visualization for efficient system design and optimization<sup>184</sup>. Next, this chapter reviews several application-driven case studies to explore and analyze eNVM storage solutions for DNN inference acceleration, graph processing, and general-purpose compute. Each eNVM is viable in certain contexts, and the most compelling eNVM is dependent on application behavior, system constraints, and device-level choices. This finding suggests the existence of many possible architecture-device co-design opportunities, also including specialization of application-level constraints and characteristics, which is the focus of Section 4.4. Interestingly, the presented co-design studies reveal both opportunities and potential disconnects between the priorities of architects and device-designers in maximizing memory efficiency in a realistic system context.



**Figure 4.0:** NVME Explorer Overview; There are three main stages, as discussed in Section 4.0: (1) a user configures their design space of interest, either with specific constraints and characteristics or leveraging broad, provided configurations; (2) the evaluation automatically generates corresponding configurations, runs a backend memory array simulator and/or application-level fault injection trials, and uses an analytical model to extrapolate results to application-level and system-level metrics of interest; (3) a user explores results interactively, discovering opportunities for optimization and optimal memory system configurations, as well as filtering and refining the design space, adapted from <sup>185</sup>.

#### 4.0 AN EFFICIENT, EXTENSIBLE DESIGN SPACE EXPLORATION FRAMEWORK

At a high level, NVME Explorer is a comprehensive design space exploration (DSE) framework integrating application-level characteristics, system constraints, and circuit and device parameters in a publicly-available, simple-to-use flow. The overall structure of NVME Explorer relies on three stages, described in more details in the following subsections:

1. A comprehensive cross-stack configuration interface to specify the design space of interest. This configuration spans the computing stack from application (blue) and system (orange) down to circuits and devices (green).
2. An evaluation engine which automatically generates configurations, simulates memory arrays, processes application behavior, computes key metrics such as performance, power, area, accuracy, and lifetime, and generates meaningful visualizations. Evaluation steps which extend existing tools are shaded grey in Fig. 4.0.
  - (a) First, a modified and extended version of NVSim extrapolates cell characteristics to memory array-level metrics

- (b) Additionally, application data and memory fault models can be connected to an application-level fault injection tool to quantify the impacts of memory faults
  - (c) Then, the analytical model estimates overall dynamic power, latency, and memory lifetime as dictated by application use-case and system settings
3. An interactive, web-based visualization tool to aide discovering, filtering and refining eNVM design points.

#### 4.0.0 CROSS-STACK CONFIGURATION

To evaluate and compare eNVM solutions in system settings, it is not just cell or even array-level characteristics of a particular technology that matter. Rather, viable solutions depend on the area, power, and latency budget of a system and how applications running on that system interact with the memory. NVMExplorer provides a rich interface for configuring key application, system, and circuit and device parameters.

At the application level, the user inputs information about memory traffic, which may include the number of read and write operations, their proportion relative to the total number of memory accesses, and how accesses are spread out over execution time. These configuration parameters may be fixed values (e.g., characterization results of a specific workload) or provided as ranges to generate generic memory traffic patterns. With either type of application input, NVMExplorer will analytically evaluate a given memory specification over a range of application scenarios. Some applications may have additional demands or metrics which are tightly related to memory technology characteristics. For example, machine learning applications or approximate computing methods may trade-off relaxed accuracy for performance and energy, and NVMExplorer also provides an interface for designers to study the application interactions and implications of fault-prone eNVM solutions.

At the system level, the user has the freedom to evaluate a wide variety of memory configura-

tion options by either setting performance, power, and area constraints and optimization goals or by choosing memory array specifications such as capacity, multi-level programming, bank configuration, and more. The circuits and devices level of the design space configuration comprises per-technology memory cell characteristics, in addition to sensing and programming circuitry choices. NVMEexplorer also provides a database of eNVM cell configurations derived from ISSCC, IEDM, and VLSI publications, but it is also possible (and encouraged!) for users to extend the current NVMEexplorer database with new simulation-based (*i.e.* SPICE or TCAD models), measured, or projected circuit and device properties. Once the full-stack specifications are set, NVMEexplorer automatically generates configuration files, which are used as input to the evaluation engine.

#### 4.0.1 EVALUATION ENGINE

NVMEexplorer natively supports a wide range of design space configurations including application choices and generic memory traffic patterns, system constraints, and eNVM cell characteristics. However, the user is also free to configure their own design space. Given the auto-generated cell and system-level sweep configurations, the evaluation engine produces memory array architecture characterizations and computes application- and system-level power, performance, area, and reliability metrics. NVMEexplorer combines a customized memory array simulator, an application-level fault injection tool, and an analytical model to perform these evaluations.

To characterize memory arrays, NVMEexplorer relies on a customized version of NVSim, a previously validated tool to compute array-level timing, energy, and area<sup>62</sup>. NVMEexplorer builds on existing efforts to extend NVSim to support multi-level cells and ferroelectric-based eNVMs<sup>204,183</sup>. In addition, NVMEexplorer provides a more user-friendly configuration interface to ease data collection and post-processing. We introduce the capabilities of NVMEexplorer in comparing eNVMs in Section 4.0.2.

Results of cell-level and circuit-level simulations are used to parameterize fault models and per-

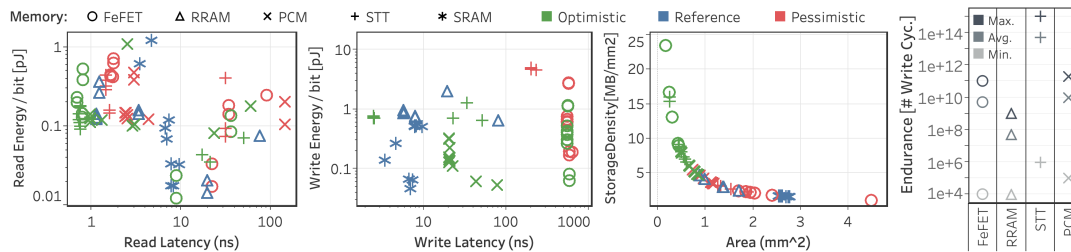
form application-level fault injection, as in Chapter 2. For performance estimations, in lieu of cycle-accurate simulation, NVMEexplorer utilizes a long-pole, bandwidth driven model that takes memory access latency and available read/write bandwidth and compares aggregated access latency per workload execution and per second of execution to workload access statistics. This is similar in spirit to performance models in <sup>66,178</sup>, and it serves the primary purpose of identifying memory solutions that cause application slowdown, rather than predicting precise latency implications. To extract other critical application-level metrics, such as energy, NVMEexplorer aggregates the read and write access energy based on the number of application accesses and array energy-per-access with the leakage power, scaling according to use-case and wake-up frequency for intermittent operation. Similarly, memory lifetime is extrapolated by applying the average reported endurance to the write access pattern per workload and the use-case.

#### 4.0.2 EXAMPLE ARRAY-LEVEL COMPARISON

Figure 4.1 presents example array characterization output generated by NVMEexplorer after evaluating various eNVM configurations implemented in a 22nm node. The design points are color-coded to highlight optimistic (green), pessimistic (red), or reference (blue) designs across surveyed publications per cell technology, as described in more detail in Section 4.0.7. The figure also reports the characteristics of 16nm SRAM as a comparison point. For each technology, we show array characterization under different optimization goals, which result in a variety of internal array architectures (e.g., bank arrangement and column muxing for read-out, as summarized in Chapter 1.0). For example, we observe a wide range for the read-energy-per-bit of an iso-capacity SRAM array. This result reflects the effect of different array optimization targets (read energy-delay product, write characteristics, area) on the internal bank configuration and periphery overhead.

This preliminary study already provides a few key takeaways. Each eNVM is able to attain read access characteristics competitive with SRAM, with the exception of an array characterized with





**Figure 4.1:** NVMExplorer example array characterization (4MB, various optimization targets), including read and write characteristics per access, storage density, and projected endurance based on the minimum, maximum, and average reported endurance across survey results in Chapter 1.2, taken from <sup>185</sup>.

pessimistic underlying PCM cell characteristics. However, write access characteristics vary dramatically across published eNVM examples, in addition to the range of reported endurance per technology. The tension between these properties and potential storage density (even in the absence of multi-level cell programming) indicates that array-level comparison in isolation may guide a system designer towards sub-optimal solutions. For example, a FeFET-based memory may seem a fitting choice for high-density, read-performant storage, but performance and energy efficiency of those memories are highly shaped by application traffic patterns and underlying cell assumptions. Thus, the cross-stack nature of data exploration supported by NVMExplorer is essential in guiding system-level choices and further investigation.

#### 4.0.3 FAULT MODELING AND RELIABILITY STUDIES

In addition to characterizing memory performance, power, area, and lifetime, NVMExplorer extends previously validated efforts in application-level fault injection to provide an interface for fault modeling and reliability studies <sup>193</sup>. Users can provide an expected error rate or more detailed, technology-specific fault models and storage formats to perform fault injection trials on application data stored in different eNVMs. To quantify the impact on application-specific metrics of accuracy, the fault injection tool is tightly integrated with application libraries for data-intensive workloads,

including PyTorch for DNNs and SnapPY for graph processing<sup>181,135</sup>, as well as numpy for generic application data. As a demonstration, we perform SPICE simulation and extract fault characteristics associated with single-level vs. multi-level cell (SLC vs. MLC) programming and sensing circuitry characteristics. In this work, we consider a subset of eNVMs, namely, RRAM, CTT, and FeFET, whose fault characteristics could be derived from existing modeling efforts<sup>183,204,60</sup>. We use our extended fault injection framework to simulate the impact of storing workload data in SLCs vs. MLCs and with varying sensing circuitry designs and parameters. Armed with these additional capabilities, NVMEplorer can replicate the results of previous considerations of eNVM storage reliability<sup>183</sup>, in addition to providing a broader platform for studying the interactions between programming choices, cell characteristics, and application accuracy.

#### 4.0.4 DATA VISUALIZATION AS A CORNERSTONE OF DESIGN SPACE EXPLORATION

The figures in this work are snapshots from NVMEplorer’s interactive web-based data visualizations, built using Tableau<sup>225</sup>, many of which are freely available<sup>185</sup>. A screenshot of one such view is provided in Figure 4.2. In each study, results are filtered and constrained according to system optimization priorities and application use cases, as described in the text. The basic NVMEplorer data visualization dashboard presents power, performance, area, and memory lifetime results across all user-configured sweep results (e.g., many application traffic patterns, array provisioning choices, and/or eNVM cell configurations) alongside array-level metrics for a holistic design exploration experience. A user can filter results in terms of important constraints (e.g., latency or accuracy targets, power or memory area budget) and identify design points of interest. While several features of these visualizations are evident in the figures in this work, including dynamic filtering across plots, click-and-drag to narrow the design space, and pop-up details about results, the reader is encouraged to use their imagination in how they might explore and filter the data shown in alternative ways according to their interests, questions, or confusions.

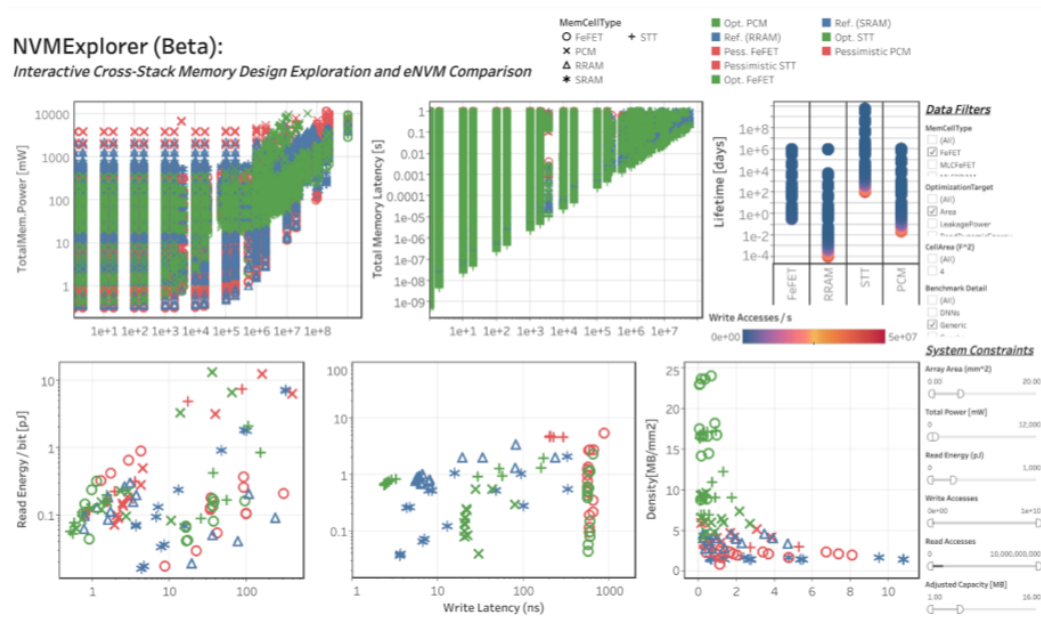


Figure 4.2: NVMExplorer dashboard summary, as appeared on live webpage in November 2021.

NVMExplorer is designed to be navigable and extensible to allow for users to conduct studies which are relevant to the design spaces they care about. A user can easily explore tweaking different device parameters or architecture constraints to see how it affects the results and metrics. This holistic approach makes application-architecture-device co-design easily accessible because it directly exposes trade-offs between device parameters and architecture metrics like performance and power for specific application settings. Using the interactive data visualization web tool, users can quickly filter results to understand results in an intuitive way and to conduct one-off studies related to a specific co-design target. Sections 4.1-4.3 explore different application-driven case studies using NVMExplorer, and Section 4.4, explores different architecture-device co-design opportunities. The application-driven case studies include a case study on eNVMs for DNN inference on IoT devices, a system agnostic graph workload study, and a study on an eNVM-based last-level cache. These case studies demonstrated some specific questions that can be answered using NVMExplorer. The co-design opportunities include both device-driven opportunities as well as architecture-driven

opportunities. The device-driven opportunities include alternative FeFET fabrication choices, trading area efficiency for increased performance, and variation in multi-level programming advantage across technologies. The architecture-driven opportunities includes intermittent operation and write buffering.

#### 4.0.5 CHAMPVIS: COMPARATIVE, HIERARCHICAL PERFORMANCE ANALYSIS

Effective and accessible data visualization can accelerate the process of designing and optimizing efficient computing systems. In this section, I briefly present a data visualization and analysis tool I developed towards identifying performance bottlenecks in general-purpose systems called CHAMPVIS<sup>184</sup>. Additionally, I'll describe how the design and implementation of CHAMPVIS informed the importance of providing an accessible, interactive simulation and data visualization framework as part of NVMEexplorer.

The first step to system and algorithmic optimization is identifying performance bottlenecks. The computer systems community has made significant strides in building tools for bottleneck analysis. TopDown<sup>257</sup> records the performance impact of these events and outlines a structured, hierarchical methodology to categorize critical bottlenecks. At the highest level, modern computers are organized into three major stages: (1) the frontend, which determines what work to perform, (2) the processing unit, which computes the work, and (3) backend, which stores the final output for future use in memory. Each of these stages can be broken down into a hierarchy of sub-stages, up to 4-levels, down to the utilization of individual hardware components (e.g., floating point vector computation unit or L1 cache for a CPU core). This methodology was adopted in commercial tools including Intel VTune<sup>103</sup>. However, these tools only consider a single application and hardware platform at a time, making comparative analysis laborious and time-consuming. CHAMPVIS is a web-based visualization platform that enables using TopDown-style hierarchical performance analysis across multiple applications.

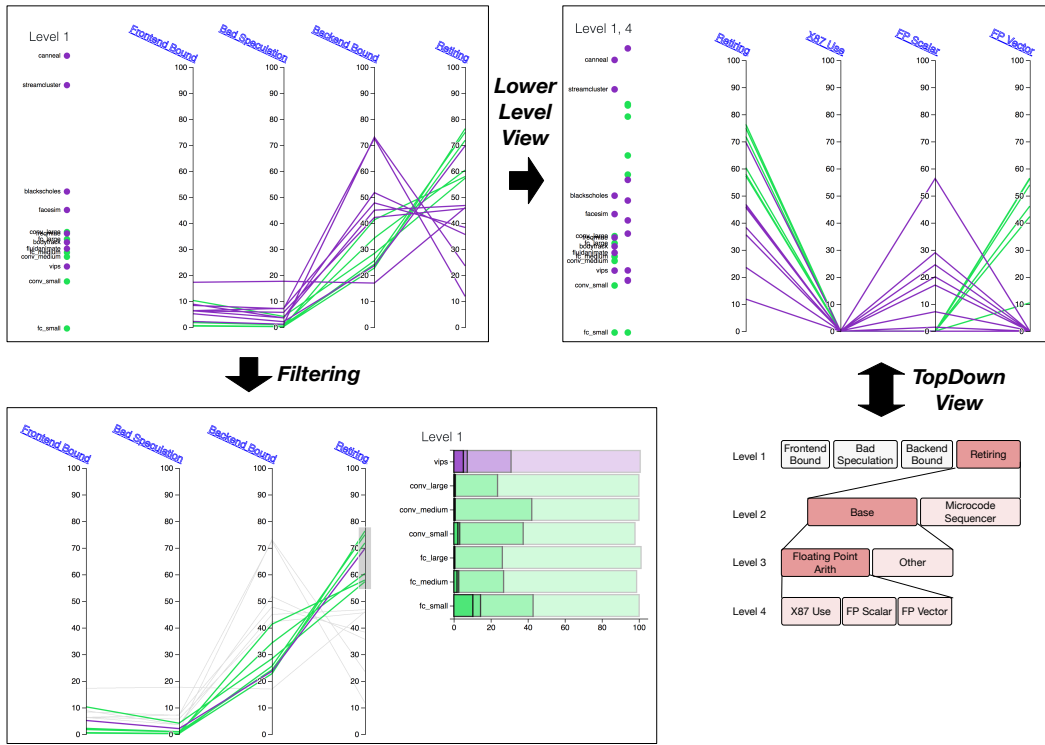


Figure 4.3: CHAMPVis representative figures; a user may flexibly explore TopDown<sup>257</sup> statistics across many workloads or hardware platforms, filtering to identify and classify performance bottlenecks at a glance, taken from<sup>184</sup>.

Supporting comparative performance analysis across applications and hardware platforms is key for datacenter-scale optimizations. Typical datacenters contain thousands of machines which are grouped into tens of machine types, and each essential application spans thousands of lines of code<sup>87</sup>. By analyzing a diverse set of applications or hardware platforms, performance optimizations can be applied to a wider range of inefficiencies. For example, Google and Facebook apply TopDown performance analysis within their datacenters<sup>114,219</sup>. These works led to solutions that reduced total datacenter cycles by up to 2%. However, researchers are currently limited to studying only the most abstract level in TopDown — deeper evaluations would require visualization techniques tailored to datacenter-scale hierarchical comparisons.

To enable productive, efficient, and user-friendly performance comparisons, I helped to develop

CHAMPVis, from which some representative images are presented in Figure 4.3. CHAMPVis facilitates identifying similarities and differences across different software applications and hardware platforms. CHAMPVis was developed considering several key design decisions such as how to visually relate performance metrics at different levels of the hierarchical hardware structure, how to guide users through the analysis process, and how to compare the performance of different applications and hardware platforms across multi-dimensional metrics. CHAMPVis allows users to directly interact with and extract useful details from hierarchical performance analysis data while simultaneously making effective comparisons. This closes the loop between identifying an important performance bottleneck and comparing results across software applications or hardware platforms. This will enable datacenter system engineers and software application developers to more productively optimize performance. Thinking critically about how to represent and interact with data for effective exploration and analysis in the CHAMPVis project inspired the development of openly-available and interactive data visualizations as part of the NVMExplorer methodology.

#### 4.0.6 COMPARISON TO, INTEGRATION WITH OTHER TOOLS AND FRAMEWORKS

Previous work in evaluating eNVMs can be characterized as either focusing on device-level and array-level evaluations, or providing in-depth cross-stack analysis for a particular combination of eNVM and application target. Table 4.0 codifies the key differences between NVMExplorer and some related works. Survey works such as the Stanford Memory Trends<sup>220</sup> maintain a list of key parameters, like storage capacity and write energy, while previously validated array-level characterization tools, such as NVSim<sup>62</sup>, characterize timing, energy, and area of eNVM-based memory structures. DESTINY<sup>187</sup> modifies NVSim to evaluate 3D integration.

To evaluate eNVMs in a system setting, prior work typically integrates NVSim with a system simulator. DeepNVM/DeepNVM++<sup>102,101</sup> enables cross-layer modeling, optimization, and design space exploration of MRAM-based technologies in the context of GPU cache for DNNs using

		Tech. Surveys		Array Simulators		Arch-Specific Frameworks			NVMExplorer
		IRDS <sup>100</sup>	Trends <sup>120</sup>	NVSim <sup>92</sup>	Destiny <sup>187</sup>	Neuro-Sim+ <sup>34</sup>	NVMain <sup>188</sup>	Deep-NVM++ <sup>104</sup>	
NVM	RRAM	✓	✓	✓	✓	✓			✓
	STT		✓	✓	✓	✓	✓	✓	✓
	SOT		✓					✓	✓
	PCM		✓	✓	✓	✓	✓		✓
	CTT								✓
	FeRAM	✓	✓						✓
	FeFET		✓			✓			✓
Circuits	MLC					✓			✓
	Fault Modeling					✓			✓
Architectural Simulator / Use Case						PIM + DNNs	gem5	GPGPU-sim + DNNs	Analytical; many included
App-Aware Evaluation	Accuracy					✓			✓
	Memory Lifetime						✓		✓
	Operating Power					✓		✓	✓
	Latency					✓	✓	✓	✓

**Table 4.0:** NVMExplorer leverages existing efforts by extending NVSim, while enabling cross-stack DSE across multiple use cases and domains, including more breadth than previous works, and providing a unified platform to explore and iterate design, adapted from <sup>185</sup>.

GPGPUSim. NVMain<sup>188</sup> enables evaluation of eNVM-based main memory using gem5<sup>20</sup>. NeuroSim+<sup>34</sup> focuses on evaluation of processing-in-memory for DNN inference and training. While these frameworks are great examples of domain-specific explorations and evaluations, NVMExplorer can evaluate a variety of system and application domains, in addition to offering reliability analysis, additional metrics such as memory lifetime, and a database of technology cell characteristics and configurable device parameters.

Existing works such as these provide limited or otherwise domain-specific design space exploration frameworks. The time required to sift through non-volatile memory roadmaps, learn the growing number of NVM simulators, and piece together toolflows to answer questions about specific architectures and applications is throttling the pace of co-design between architects and device designers and inhibiting the advancement of technologically-heterogeneous memory systems. In contrast, NVMExplorer’s cross-stack and analytical approach provides efficient evaluation of critical application-aware metrics such as application accuracy and resulting memory lifetime with sufficient fidelity to guide future design studies. NVMExplorer offers more breadth by including application-, system-, and device-level considerations, and accommodating a wider range of devices without requiring a separate system simulator.

NVMEplorer’s modular design allows for efficient and effective integration of other underlying memory characterization and simulation tools as the research space continues to grow and evolve. For example, at the configuration stage, NVMEplorer can interface with profiling tools and memory trace generators to designate application traffic rates and patterns of interest. For example, memory access statistics generated using the Sniper Multi-Core Simulator<sup>26</sup> are used as representative cache behavior in Section 4.3. Also at the configuration stage, NVMEplorer has the potential to directly interface with cell, device, and circuit simulation tools (e.g., HSPICE) to configure memory cell characteristics, programming settings, and array component properties and overheads. At the evaluation stage, NVMEplorer is sufficiently modular that the memory array-level characterization tool can be exchanged for alternative simulators to evaluate distinct memory solutions, such as 3D-stacked architectures using Destiny<sup>187</sup> or cryogenic operation using CryoMem<sup>163</sup>. Both array characterization results and application-level metrics can then be harnessed at the exploration stage to guide iteration of the design space or directly as inputs to a broader simulation framework for more detailed evaluation (e.g., to parameterize a new memory system component in gem5<sup>20</sup>, or to take advantage of existing tools for thermal analysis<sup>85</sup>).

#### 4.0.7 TECHNOLOGY LANDSCAPE AND EXAMPLE CELL PARAMETERIZATION

NVMEplorer provides a broad survey of published eNVM examples, which can be parameterized so that systems experts can make meaningful, high-level comparisons across technologies despite different underlying trade-offs and maturity. We validated this approach per-technology against fabricated memory arrays<sup>185</sup>.

This work was informed by a compilation of device-level and array-level data across eNVM technologies, as summarized in Chapter 1.2. We sourced the majority of the cell-level parameters from ISSCC, IEDM, and VLSI publications and focus primarily on works from 2017-2020 to reflect the most recent range of achievable behavior per technology. Previous efforts detailed the physical



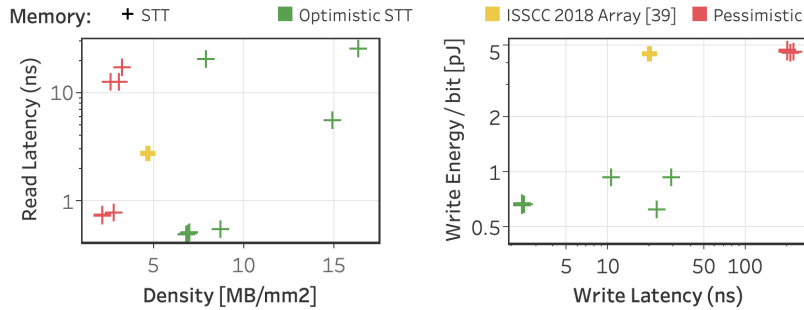
properties and limitations per technology<sup>31</sup>, while NVMExplorer focuses on compiling sufficient cell-level details and leaning on existing technology models to provide a broad and practical database of cell definitions. While we hope these extracted cell definitions are helpful to the community in calibrating the current state-of-the-art, NVMExplorer is extensible as the design space continues to evolve, as demonstrated in Section 4.4.

Comparing eNVMS at varying stages of development and with varying underlying physical properties is a challenging task. The case studies in this work aim to provide high-level guidance and relative judgments about which eNVM cell technologies are worthy of further investigation under specific system and application constraints. Thus, rather than focus in on specific, physically accurate cell configurations, NVMExplorer provides the bounds of what is conceivable per eNVM technology across the full range of published recent academic work. We liken identifying and evaluating these bounds per-technology to forming the poles of a tent that encompasses the full extent of eNVM properties, so we call the extrema in terms of cell-level characteristics (i.e., smallest, lowest read energy, best retention vs. largest cell size, lowest endurance) the device-level “tentpoles”. In an actively evolving technology space, this approach allows us to make meaningful classifications about which technologies are potentially adoptable solutions. These modeling choices are classified into two fixed cell configurations for applicable technologies. We validate that the “tentpoles” of the cell-level design space result in array-level characterization that provides coverage of published memory array properties, as discussed in Section 4.0.8.

For the technology classes most represented in our survey, we compute which published example has the best-case and worst-case storage density in terms of Mb/F<sup>2</sup>, and this data serves as the foundation of the bounds of the cell-level design space; those points which are most and least dense across recent published examples. Any critical cell-level parameters not reported with those cell definitions are assigned values (e.g., read characteristics and programming settings) using the best (lowest power, highest efficiency) or worst (highest power, lowest efficiency) value per metric across

all other recent publications with sufficient supporting data. These best-case and worst-case technologies per class form the tentpoles of the underlying cell design space, and we label these fixed cell definitions as “optimistic” or “pessimistic” accordingly. For the purposes of the case studies presented in published works<sup>185</sup>, all array- and application-level results are produced using these fixed underlying optimistic and pessimistic cell properties, though we note that a user of NVMEexplorer can draw either on these constructed, bounding example cells or on the full database of surveyed configurations, or on fully customized definitions with respect to cell size, access properties, and operating conditions (e.g., read/write voltage, temperature). Corresponding fault models and error rates for reliability studies are extracted after optimistic vs. pessimistic cell-level properties are fixed, as discussed in one of the presented case studies.

This approach is helpful for many reasons: for one, these extremes help us answer exploratory questions about what we will likely see in the near future; secondly, comparing the best-case of one technology to the worst-case of another can help gauge less mature technologies against more mature reference points; thirdly, if such optimistic configurations are untenable or even pessimistic configurations are attractive in a specific system setting, we can build confidence for further exploration and more detailed modeling efforts without implementing and attempting to meaningfully compare many many cell definitions with insufficient data. A limitation of this methodology is that inherent trade-offs between certain parameters for a technology may not be linked (e.g., area, latency, and retention for STT); however, this amalgam of cell properties represent the full spectrum of achievable characteristics per technology, rather than specific fabricated results. As a point of additional comparison, the results shown in the following studies include a reference cell configuration for RRAM as a relatively mature eNVM, with parameters derived from a specific industry result<sup>128</sup>.



**Figure 4.4:** Example memory-array-level validation of reference (yellow)<sup>61</sup> vs. optimistic (green) and pessimistic (red) underlying cell ‘tentpole’ configurations; metrics of a fabricated STT array from the literature<sup>61</sup> fall within the bounds of those generated using our device-level ‘tentpole’ characterization, taken from<sup>185</sup>.

#### 4.0.8 VALIDATION

Our array-level area, energy, and latency characterizations rely on the previously-validated procedures of NVSim to extrapolate cell-level configurations and array design constraints to optimized memory layouts and properties<sup>62</sup>. However, in employing our “tentpole” approach, it is critical that we verify that array-level results using our optimistic and pessimistic underlying cell characteristics fully cover and match expectations of existing manufactured and published eNVM solutions.

Whenever possible, we select publications with array-level characterizations for a given technology, and compare those results to iso-capacity memory arrays modeled through our “tentpole” approach. Figure 4.4 shows an example of such an exercise. We compare a 1MB STT-RAM array published at ISSCC in 2018<sup>61</sup> to optimistic and pessimistic STT design points produced by NVM-Explorer. Here, we note that our tentpole results effectively represent the range of actual array properties by producing metrics that are both higher and lower, but similar in magnitude, to the reference STT-RAM array. The studies presented in this work consider only validated configurations for which we were able to either complete this validation exercise or run SPICE-level simulations. It is worth noting that NVMExplorer is set up to evaluate all cell technologies in Table 1.1 (e.g., though SOT is a compelling emerging solution, our survey found insufficient array-level data for validation,

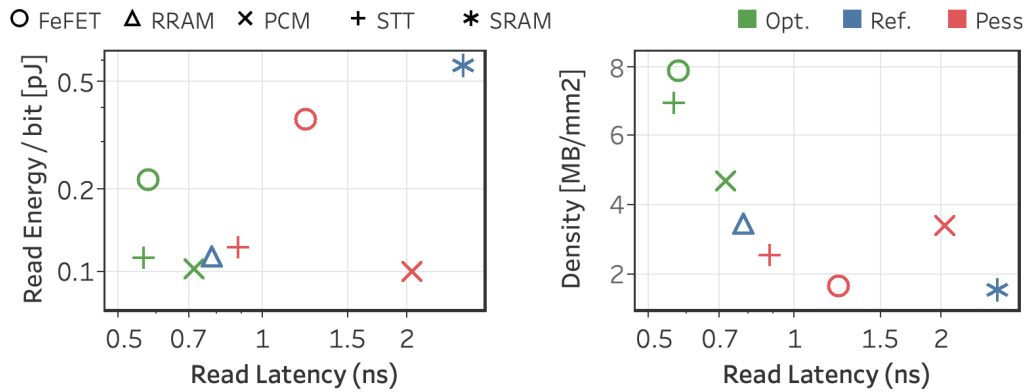
so it is omitted in this Chapter’s evaluations). System validation and application characteristics are derived from existing, state-of-the-art references.

#### 4.1 SUPPORTING DNN INFERENCE UNDER VARYING OPERATING CONDITIONS

Prior studies have demonstrated the potential benefits of eNVM storage for Deep Neural Network (DNN) inference accelerators<sup>183,59,239</sup>, albeit with limited scope in terms of eNVM technologies and cross-stack parameters considered. NVMEplorer empowers researchers to approach a broader set of questions that compare eNVMs in different storage scenarios (e.g., limited to weights vs. storage of DNN parameters and intermediate results) and system constraints (e.g., strict area budget, or power budget). In this section, we consider two distinct use cases for a DNN inference accelerator: continuous operation, as in image processing per frame of a streamed video input, and intermittent operation, where the system is woken up per inference task and can leverage the non-volatility of eNVM by retaining DNN parameters on-chip in power-off state between inferences.

##### 4.1.0 CONTINUOUS OPERATION

We consider the commonly-used and well-studied NVDLA<sup>212</sup> as a base computing platform (as in Chapter 3.1) and compare its 2MB SRAM with iso-capacity eNVMs. We use the NVDLA performance model<sup>178</sup> to extract realistic memory access patterns and bandwidth requirements of the on-chip buffer. More specifically, we evaluate the power and performance of accesses to on-chip memory storing ResNet26 weights for single-task image classification using the ImageNet dataset vs. multi-task image processing, comprising object detection, tracking, and classification, at a consistent frame rate of 60 frames-per-second, as is typical for HD video. We additionally consider the impact of storing activations in eNVM, but this ostensibly ignores endurance limitations.



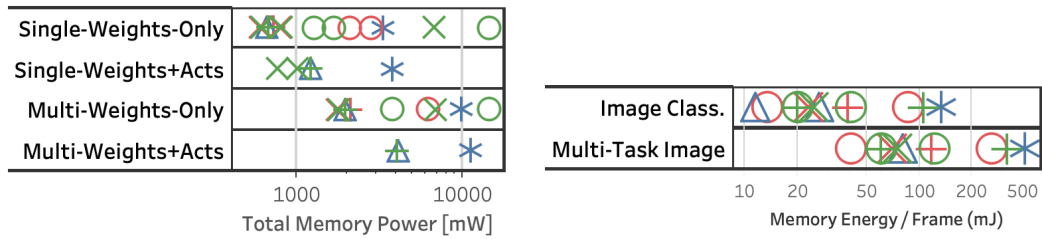
**Figure 4.5:** Read characteristics and storage density for 2MB arrays, provisioned to replace on-chip SRAM for NVDLA, taken from <sup>185</sup>.

#### 4.1.1 ARRAY CHARACTERISTICS

First, we observe the read and storage density characteristics for 2MB arrays using the cell-level tent-poles of several promising eNVM technology classes, as shown in Figure 4.5 compared with SRAM. Notice that read energy effectively divides arrays into two tiers. STT, PCM, and RRAM offer lower read energies and competitive read latencies vs. SRAM. In contrast, FeFET-based eNVMs suffer from higher read energies, but optimistic FeFET offers the highest storage density with low latency. At similar low latency, optimistic STT offers 6× higher density over SRAM. PCM and RRAM outperform SRAM in terms of both read latency and storage density. While such comparative insights can readily be extracted from this pair of plots, there are other important dimensions to also consider, and NVMExplorer facilitates more comprehensive analyses that consider the impact of application priorities and system-level use cases on eNVM design decisions.

#### 4.1.2 APPLICATION-LEVEL METRICS

Figure 4.6, left, summarizes total operating power (both dynamic access and leakage power) for the 2MB memory arrays characterized in Figure 4.5 and accessed according to traffic patterns of dif-



**Figure 4.6:** The most energy-efficient eNVM varies under different DNN inference use cases, such as continuous (left, operating power) vs. intermittent (right, reporting energy per input image frame); these results exclude eNVM solutions that are unable to meet application latency and accuracy targets, taken from <sup>185</sup>.

ferent ResNet deployment scenarios, i.e., single- vs. multi-task and weights-only vs. storing both weights and activations. These results exclude eNVM candidates that cannot support 60 FPS operation nor maintain DNN accuracy targets. Recall NVMExplorer includes fault injection wherein high eNVM fault rates can degrade model accuracy to unacceptable levels. While not explicitly shown here, NVMExplorer exposes numerous additional interactions for users to probe and build intuition. For example, while total memory power increases as the number of accesses per frame increases to compute multiple tasks, the ratio of read-to-write traffic stays roughly the same. Hence, the relative power of eNVM arrays also remains similar. In particular, PCM, RRAM, and STT all offer over  $4\times$  reduction in total memory power over SRAM. One important reason for this is that SRAM leakage power will dominate compared to eNVM solutions, even under high traffic. Of the energy-efficient solutions, STT offers best performance (lowest application latency per frame). In contrast, optimistic FeFET offers higher storage density while maintaining 60FPS and a  $1.5\text{-}3\times$  power advantage over SRAM.

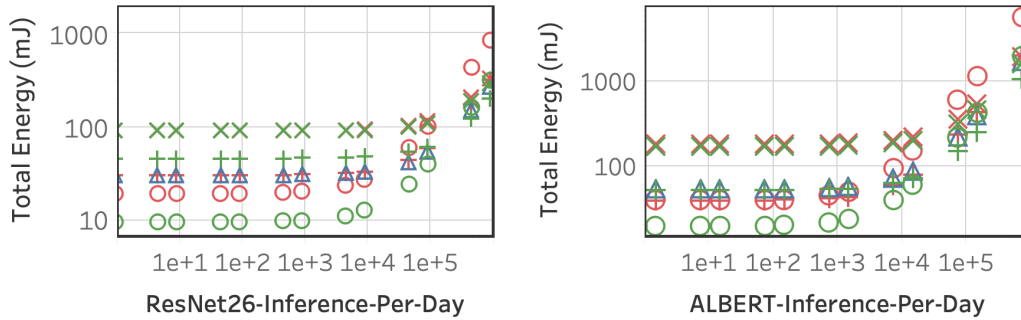
#### 4.1.3 INTERMITTENT OPERATION

Let us now consider eNVM storage for two additional use cases that alter system-level optimization goals and corresponding eNVM selection, further highlighting the flexibility and ease of exploration

the NVMExplorer framework offers. A major advantage of storing DNN weights in eNVMs is that non-volatility supports intermittent operation that powers off the accelerator between inferences. Using SRAMs would either consume leakage power to keep the weights memory powered on or consume power to restore the weights from off-chip memory, e.g., by incurring a latency and energy penalty by fetching from DRAM. In this use case, we provision monolithic eNVM storage to hold all DNN weights (e.g., up to 32MB for Natural Language Processing (NLP) tasks). For image processing, all weight memory accesses are to eNVM, eliminating the wake-up latency and power associated with loading parameters on-chip, in addition to reducing distance between compute system and higher-capacity memory. Previous work demonstrated that careful optimization between DNN properties and MLC eNVM storage can further increase storage density<sup>183,204</sup>.

Figure 4.6, right, compares the resulting memory-energy-per-inference across eNVMs for both single-task image classification and multi-task image processing, as determined by the total number of accesses to retrieve all DNN weights over the course of processing one input frame. The lowest-energy technology choice differs between the single vs. multi-task inference and, perhaps more interesting, both are eNVM candidates with *lower* storage density (RRAM and pessimistic FeFET), as opposed to the highest density options (STT and optimistic FeFET), which hints at a cross-stack prioritization of read performance as opposed to cell size reduction, as probed further in Section 4.4. We repeat this study for single task vs. multi-task natural language processing using the ALBERT network, a relatively small-footprint, high-accuracy, transformer-based DNN<sup>126</sup>.

We find that frequency of wake-up and specific target application traffic patterns play a critical role in selecting preferred eNVM candidates. To further study this result, we dig into the implications of intermittent operation and compare the total energy versus the number of inferences per day, showing a continuum of wake-up frequency that may arise (e.g., deployed solar-powered agricultural sensors or satellites, or a voice-enabled assistant executing NLP tasks on wake-up). The left plot of Figure 4.7 shows total memory energy as a function of inferences per day for image classi-



**Figure 4.7:** The eNVM storage solution (iso-capacity arrays provisioned per task, optimized for ReadEDP) that minimizes total memory energy consumption varies according to system wake-up frequency and DNN inference task; All solutions shown maintain application accuracy and a  $< 1s$  latency per inference, taken from<sup>185</sup>.

fication. Here, total memory energy is presented as a proxy for device battery life. From the figure, we observe that when the number of inferences per day is sufficiently low (less than  $1e5$ ), optimistic FeFET yields the lowest energy. At higher wake-up frequency, optimistic STTs take over because of the relatively lower energy-per-access. Figure 4.7, right, investigates the impact on an NLP workload. While results are similar, optimistic STT emerges as the best technology at lower inference rates (as compared to image classification), because sentence classification is a particularly intensive task, requires higher operating power than ResNet26.

Table 4.1 summarizes the preferred eNVM technology across different use cases and tasks, with “Opt. eNVM” indicating the preferred choice under optimistic underlying cell characteristics and “Alt. eNVM” indicating the preferred technology assuming pessimistic assumptions and reference points, and table entries for intermittent operation are selected at a fixed wake-up rate. As shown by these results, non-volatility of on-chip storage resources is particularly compelling for resource-constrained systems that experience intermittent power supply (e.g., deployed solar-powered agricultural sensors or satellites) or otherwise operate intermittently (e.g., a voice-enabled assistant executing NLP tasks on wake-up). Across a range of device wake-up frequencies and per-wake-up compute patterns, we observe that several eNVMs become compelling, and the preferred NVM



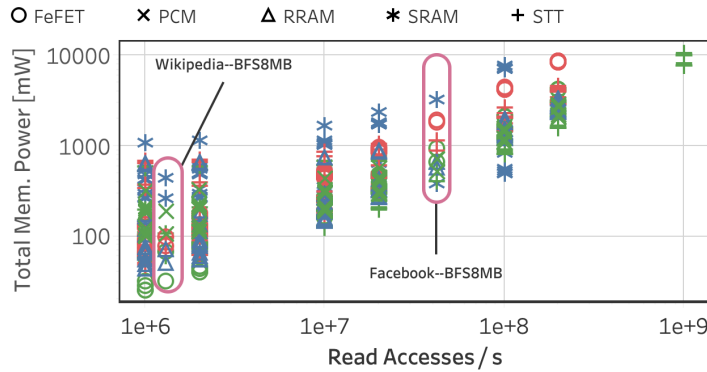
Use Case	Inference Task	Data Storage	Priority	Opt. eNVM	Alt. eNVM
Continuous (60IPS)	Single-Task Image Classification	Weights Only	Low Power	PCM	PCM
			High Density	FeFET	CTT
		Weights + Acts	Low Power	PCM	RRAM
			High Density	STT	RRAM
	Multi-Task Image Processing	Weights Only	Low Power	PCM	RRAM
			High Density	FeFET	CTT
		Weights + Acts	Low Power	STT	RRAM
			High Density	STT	RRAM
Intermittent (1IPS)	Single-Task Image Classification	Weights Only	Low Energy/Inf	RRAM	RRAM
			High Density	FeFET	CTT
	Multi-Task Image Processing	Weights Only	Low Energy/Inf	FeFET	FeFET
			High Density	FeFET	CTT
	Sentence Classification (ALBERT)	Embeddings Only	Low Energy/Inf	RRAM	RRAM
			High Density	FeFET	CTT
		All Weights	Low Energy/Inf	STT	RRAM
			High Density	FeFET	CTT
	Multi-Task NLP (ALBERT)	All Weights	Low Energy/Inf	STT	RRAM
			High Density	FeFET	CTT

**Table 4.1:** Summary of preferred eNVM under varying DNN use case, task, storage strategy, and optimization priority. Several distinct eNVM solutions are best suited in different cases under either optimistic underlying cell characteristics (“Opt eNVM”) or considering only reference and pessimistic configurations (“Alt eNVM”), taken from<sup>185</sup>.

choice for further investigation varies depending on both of these factors.

## 4.2 COMPLEMENTING AND ACCELERATING GRAPH PROCESSING

Our second case study explores the potential benefits of using eNVMs for graph processing, which imposes an entirely different set of constraints in terms of memory read and write characteristics. Graph processing comprises many read-dominated tasks with less predictable data reuse than DNNs (e.g., search kernels), but still involves write traffic and, overall, is incredibly data-intensive in terms of memory bandwidth and capacity. As an initial exploration of compatibility and viability between graph processing workloads and eNVM storage solutions, we consider the total power and resulting memory lifetime per technology under generic traffic patterns covering the range of read and write bandwidths for critical graph tasks, as described in previous workload characterization efforts<sup>17</sup>. As a proof of concept in a specific system, we additionally evaluate eNVM storage solutions under



**Figure 4.8:** Memory power for traffic patterns encompassing graph processing demands, including specific graph kernels as labeled. The lowest power solution depends on the expected read traffic, taken from<sup>185</sup>.

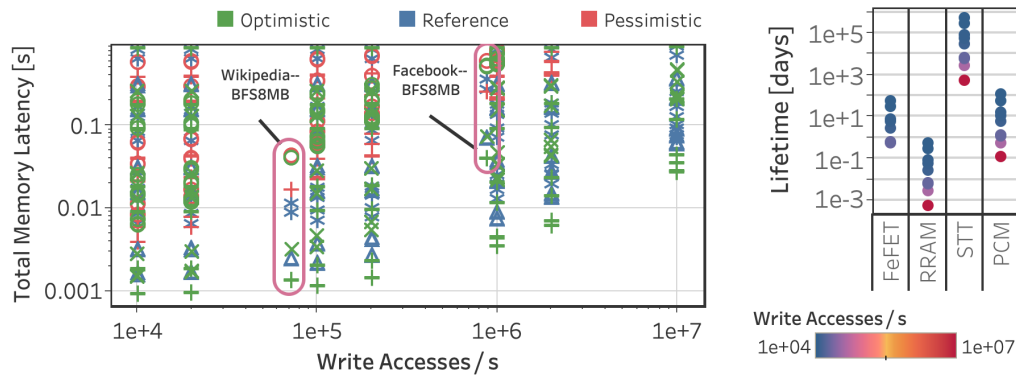
access patterns for benchmarks executed on a domain-specific accelerator<sup>83</sup>.

#### 4.2.0 ANALYSIS FOR GENERIC TRAFFIC PATTERNS

We consider different memories experiencing a range of generic traffic patterns representing graph processing kernels (i.e., read access rates from 1-10GB/s and write access rates from 1-100MB/s)<sup>17</sup>. NVMEexplorer provides a wide array of critical metrics to compare and user-configurable visualizations to extract important trends and limitations. For example, in Figures 4.8 and 4.9, we choose to display total memory power against read traffic, as number of read accesses becomes a dominant factor in total power for read-dominated workloads, and total memory latency against write traffic, as overall performance for several eNVMs is strongly determined by write traffic.

#### 4.2.1 POWER

As shown by Figure 4.8, total memory power generally increases with read access rate and the lowest power solution depends on the application traffic load. For applications that exhibit fewer than  $10^7$  read accesses per second, optimistic FeFET is a clear winner, while pessimistic FeFET and RRAM are next best candidates. On the other hand, for higher rates of read traffic (e.g.,  $> 10^8$ ), optimistic



**Figure 4.9:** Memory latency and projected lifetime for traffic patterns encompassing graph processing demands, including specific graph kernels as labeled. FeFET solutions fail to match SRAM performance. STT provides superior performance and memory lifetime, taken from <sup>185</sup>.

STT is best. For mid-range read access rates, PCM and RRAM are also viable solutions sometimes offering the lowest power solution. However, this relationship alone does not dictate memory technology choice.

#### 4.2.2 LATENCY

A slightly different and more consistent story emerges when we analyze the impact of different eNVMs on overall memory latency (both read and write) versus write access rates, shown by Figure 4.9. While there is a clear preference for optimistic STT, RRAM and optimistic PCM are also worth considering. In contrast, most pessimistic eNVM technologies and all FeFET-based solutions are significantly inferior, even failing to match SRAM performance for many traffic patterns. When we additionally consider projected memory lifetime, STT emerges the clear winner overall. Note that the right chart of Figure 4.9 plots the memory lifetime assuming continuous operation at a particular write access rate. Hence, the highest write traffic always yields the lowest lifetime. While RRAM seemed promising based on performance and power, it has the worst endurance and lowest lifetimes.

### 4.2.3 ANALYSIS FOR DOMAIN-SPECIFIC SYSTEMS

In addition to relying on generic traffic to represent the full range of expected load of graph processing, NVMExplorer can also be leveraged to answer a more specific design question: For performance targets and traffic patterns to a specific storage resource in a graph processing accelerator system, which eNVMs offer compelling characteristics that warrant further investigation? To this end, Figures 4.8 and 4.9 also include points, identified in pink, corresponding to memory traffic to run breadth-first search on two different social network graphs<sup>135</sup>. Traffic patterns are extracted from throughput and accesses reported for the compute stream of a domain-specific graph processing accelerator utilizing an 8MB eDRAM scratchpad<sup>83</sup>. In the baseline system, about 90% of the energy is spent on the eDRAM scratchpad (not including DRAM controller energy), with an operating power of at least 3.1W at the 32nm process technology node as reported from Cacti<sup>83,91</sup>. We analyze the benefits of replacing the 8MB eDRAM scratchpad with an iso-capacity eNVM array provisioned to meet the cited latency target (1.5ns).

If we exclude RRAM due to low lifetime projections, FeFET, PCM, and STT all offer significantly lower memory power (about 2-10× lower than SRAM) and even pessimistic STT offers consistent performance. These observations, based on a realistic graph processing use case extracted from prior work, are consistent with the results generated using generic traffic patterns. Again, optimal technology choice depends on higher, system-level optimization goals, and NVMExplorer provides critical insights in the presence or absence of a specific system simulator results.

If the high-level goal is to maximize storage density, FeFET is highly attractive, but severely limited by poor write latency (unable to meet application latency expectations under the higher range of traffic patterns). Rather than prematurely eliminating FeFET, designers can leverage NVMExplorer to study the impact of relaxing or adapting application targets or to explore co-design solutions that target improvements to the underlying technology or architecture, as in Section 4.4.0.

### 4.3 PROBING GENERAL-PURPOSE APPLICATIONS: eNVM AS LLC

Improved density and energy efficiency could revolutionize general-purpose on-chip storage, and recent efforts have endeavored to replace high-performance memories, like caches, with eNVM-based alternatives<sup>123,84,101</sup>. However, caches must handle a large volume of writes depending on the application, so the achievable write latency and endurance per eNVM comes to the forefront of design considerations. While the improved density and energy efficiency of eNVMs could revolutionize general-purpose on-chip storage, the open question of achievable endurance and write access characteristics per technology cannot be overlooked and needs to be a primary factor in determining a cache technology replacement.

In this study, we consider the last-level cache (LLC) of a high-performance desktop processor, similar to Intel's 14nm, 8-core Skylake. The memory hierarchy includes a private 32 KiB L1I\$; a private 32 KiB L1D\$; a private 512 KiB L2\$ (non-inclusive, write-back); and a shared ring 16MiB L3\$ with 64 B line, 16 ways (inclusive and write-back). The system includes DRAM with 2 channels, 8 B/cycle/channel, 42cycles + 51 ns latency. Representative application behavior comes from SPECrate CPU2017 (integer and floating point), and we warm-up the cache for 500M instructions and simulate for 1-billion instructions in detail using the Sniper simulator<sup>24,26</sup>. This provides application modeling data for a 16MB LLC (e.g., reads, writes, execution time per benchmark) that are inputs to the application configuration interface of NVMExplorer. When replacing the last level cache with an NVM, we use an iso-capacity configuration. Another possibility is to fill up the same amount of physical area that an equivalent SRAM-based last level cache would consume (which would lead to significant capacity advantage for NVMs).

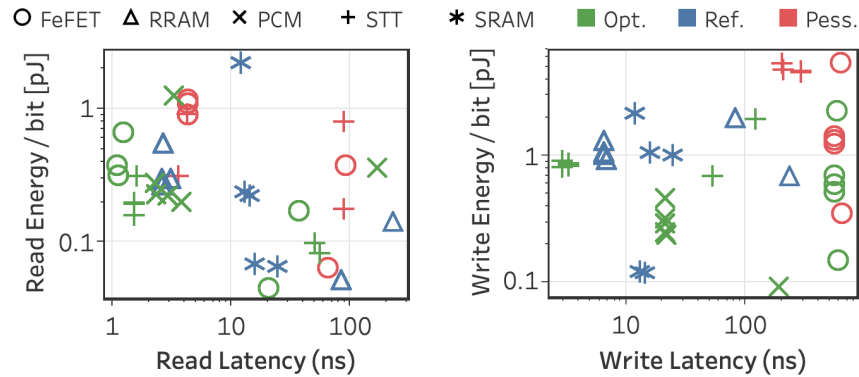
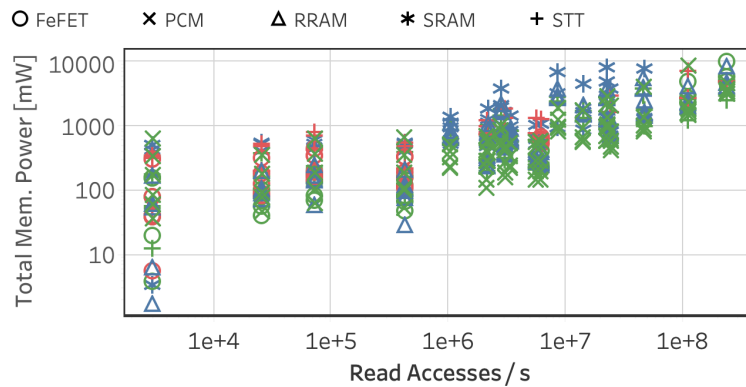


Figure 4.10: Array access characteristics in for consideration of replacing (iso-capacity) a 16MB LLC, taken from <sup>185</sup>.

#### 4.3.0 ARRAY CHARACTERISTICS

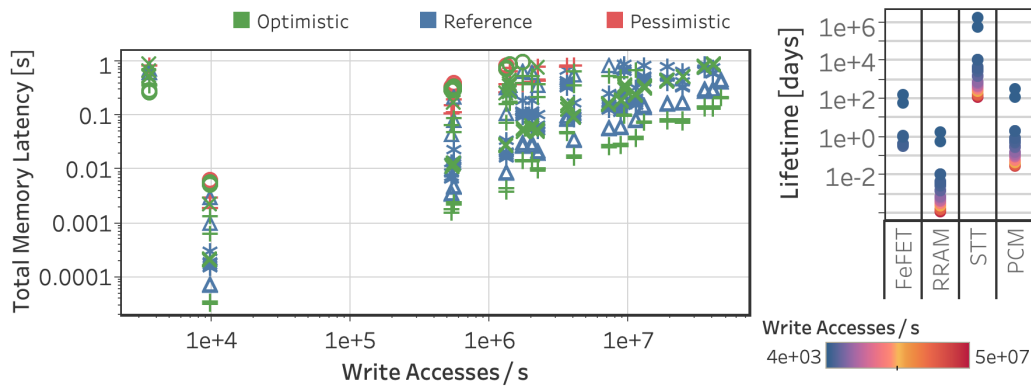
First we focus on the array characteristics of the different memory technologies in isolation, as shown in Figure 4.10. From the left plot, we note a competitive range of read energy and read latency does not reveal a clear winner. For example, if read energy per access is highest priority, FeFET, RRAM or even SRAM offer array configurations that trade access latency for energy efficient, while STT and optimistic FeFET offer pareto-optimal read characteristics. For writes, a PCM-based last level cache appears to minimize energy per access. On the other hand, only STT and RRAM are able to beat SRAM write latency. Again, we find array characteristics in isolation do not offer sufficient guidance to choose the best eNVM for LLC, and NVMEexplorer allows us to go further. In terms of storage density, STT shows over an order of magnitude reduction in area compared to SRAM (from about  $10mm^2$  to about  $1mm^2$  across optimization targets), though under pessimistic underlying cell characteristics, we note that PCM or RRAM offers higher density than STT.



**Figure 4.11:** Memory operating power under continuous operation across SPEC benchmark traffic to a 16MB LLC shows preferred eNVM depends on traffic demands and optimization goal. All solutions shown meet per-benchmark read/write demands. For high-traffic benchmarks, STT provides lowest power, lowest latency, and longest projected lifetime, taken from <sup>185</sup>.

#### 4.3.1 POWER, PERFORMANCE, AND LIFETIME

Now we start to focus on the effects of the actual application behavior on array characteristics in the context of the last level cache. It is, therefore, important to also consider system-level application behavior. Figures 4.11 and 4.12 show the resulting power, performance, and lifetime when using different eNVMs as LLC and assuming memory traffic from SPEC2017 benchmarks. Figure 4.11 shows total memory power versus read access rate, where each column of points corresponds to a particular benchmark traffic pattern. We again see that the lowest power eNVM solution depends on the traffic pattern. In broad terms, RRAM and FeFET fair better for lower read access rates while PCM is better for higher rates until STT emerges best for the highest rates. In terms of memory access latency (Figure 4.12) with respect to write access rates, STT is usually the best choice, though arrays unable to meet application bandwidth are excluded. Lastly, Figure 4.12, right, compares lifetimes across the eNVM technologies for a range of write access rates. Again, STT offers the best longevity on average. However, depending on the use case of your desktop machine, PCM and FeFET may also be compelling options (e.g., if read-dominated workloads such as xalancbmk



**Figure 4.12:** Memory latency and projected lifetime under continuous operation across SPEC benchmark traffic to a 16MB LLC shows preferred eNVM depends on traffic demands and optimization goal. All solutions shown meet per-benchmark read/write demands. For high-traffic benchmarks, STT provides lowest power, lowest latency, and longest projected lifetime, taken from <sup>185</sup>.

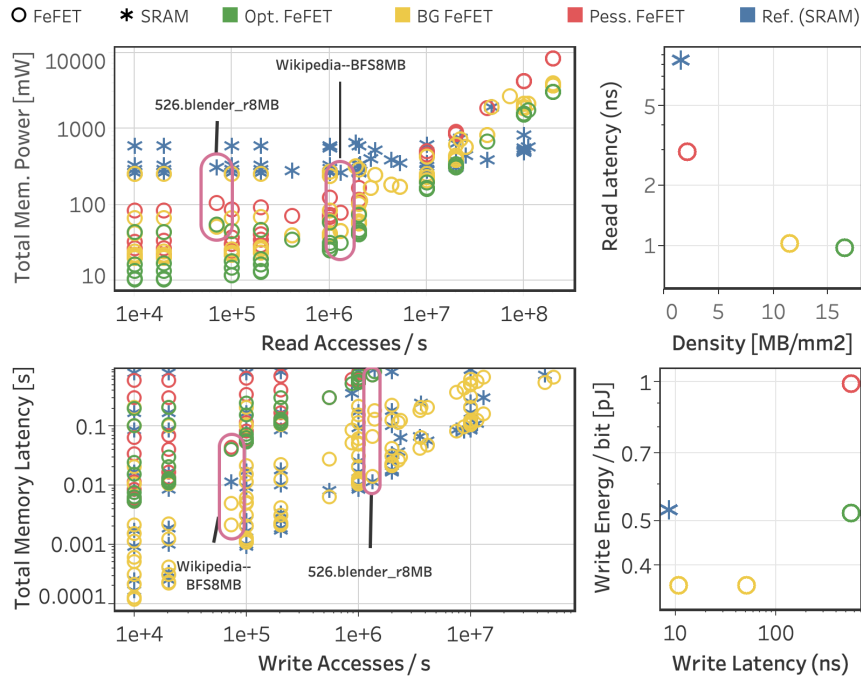
dominate an end user’s desktop machine, rather than more write-intensive scientific computing workloads). RRAM, on the other hand, does not appear viable as an LLC.

Interestingly, the winning technology is dependent on the read accesses per second seen by the last level cache only when they are sufficiently low. Even more so is the fact that it is actually the pessimistic STT technology which outperforms all others for the case where the read accesses per second is  $10^4$ . For higher values of read accesses per second, the optimistic version of STT is the clear winner. In terms of performance, total memory latency is consistently lowest for the optimistic STT regardless of number of write accesses per second.

#### 4.4 CASE STUDIES IN ENVM, SYSTEM, AND APPLICATION CO-DESIGN

Each scenario presented so far in this chapter presented unique optimization goals and system priorities and, in each case, we compare how each eNVM’s power, performance, and area fairs relative to similarly-provisioned SRAM or DRAM in a baseline system. The first case study showed that each eNVM candidate has distinct advantages as scratchpad memory for a DNN accelerator, depending





**Figure 4.13:** Back-gated (BG) FeFETs provide the high density and low operating power required by graph processing benchmarks while maintaining SRAM-comparable performance, and these solutions begin to close the performance gap between non-BG FeFET and other memory technologies across SPEC2017 benchmarks, taken from <sup>185</sup>.

on task and use cases (Table 4.1). The second case study explored – in a system-agnostic manner – promising eNVMs across expected memory traffic of graph analytics tasks and reveals that STT, PCM, and FeFET memories are all compelling solutions depending on guiding optimization target. Third, STTs emerge as the most attractive candidate to replace SRAMs in the LLC of a modern desktop CPU, consistent with prior studies <sup>84,98,7</sup>.

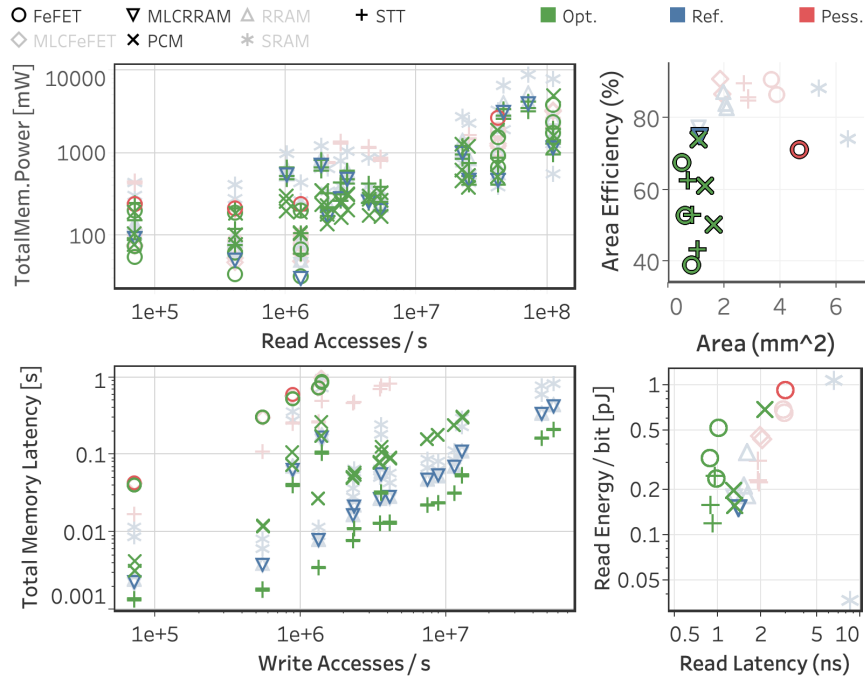
NVMEexplorer enables systematic studies that traverse the immense design space and provide navigable results, which can be progressively refined down to the most promising eNVM candidates. NVMEexplorer’s outputs offer insights into fundamental limitations per eNVM that may either be addressed by innovations at the device level or architectural solutions, and such co-design opportunities are the focus of the following discussions.

#### 4.4.0 DEVICE-DRIVEN CO-DESIGN OF FeFET-BASED ENVM

Previous FeFET-based device characterization and modeling efforts have exhibited write pulses on the order of  $100ns-1\mu s$ . However, alternative FeFET fabrication strategies in early development stages, such as back-gated FeFETs<sup>205</sup>, offer compelling potential advancements in write latency ( $10ns$  programming pulse) and projected endurance ( $10^{12}$ ). Section 4.2 and Chapter 2.4 noted that a primary limitation of FeFETs in the context of graph processing is an inability to meet the application latency targets under higher write traffic. Thus, using the underlying cell properties of back-gated FeFETs reported in<sup>205</sup>, we can rapidly re-examine the viability of FeFET-based memory and probe whether this change could make a difference in the viability of FeFET-based memory for graph processing and other workloads of interest.

Figure 4.13 shows the total memory power and total memory latency of an 8MB memory array of back-gated FeFETs (in yellow) compared to using previous FeFET models (red, green) and SRAM (blue). We examine these metrics under a range of read and write traffic patterns which are inclusive of the graph benchmarks described in Section 4.2 and the SPEC benchmarks used in Section 4.3, but here showing access patterns for an 8MB capacity LLC. Three different array organizations are tested which represent three different optimization targets: read latency, read energy, and read energy-delay-product (EDP). The underlying array-level characterization is shown in Figure 4.13, right. From the array characterization, we observe that the back-gated FeFETs show a slight increase in read energy per access and slight decrease in storage density compared to prior state-of-the-art cells. However, we observe that they enable comparable application latency to SRAM across a wide range of write traffic where previous FeFET versions fall short. Furthermore, back-gated FeFETs results in the lowest operating power over most of the range of read accesses per second, including for the example graph processing benchmark, Wikipedia-BFS8MB.

Based on these observations, we posit that back-gated FeFET memory may close the performance

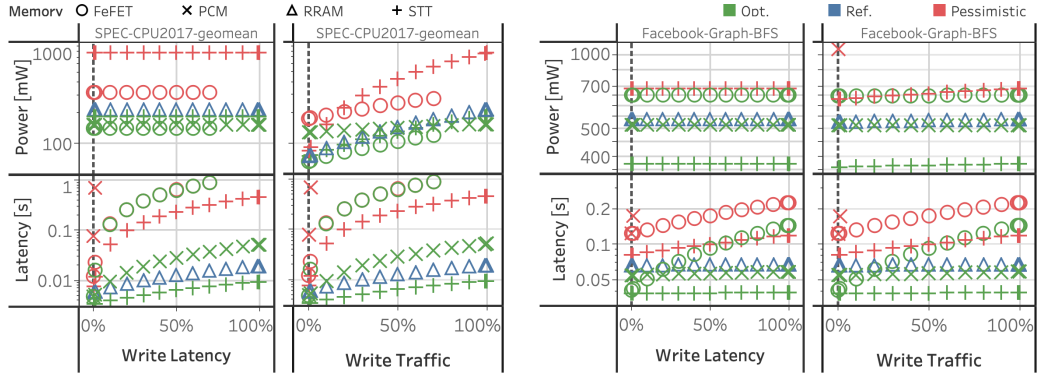


**Figure 4.14:** Results for 8MB arrays are filtered according to a maximum area efficiency (top right). Arrays with lower area efficiency are highlighted across all views and tend to result in low memory latency and power across many traffic scenarios, taken from <sup>185</sup>.

gap between prior FeFETs and other memory technologies (including SRAM) and unlock additional application domains. NVMEExplorer’s ability both to quickly and efficiently gauge the impact of cell-level innovations and to match emerging device designs to compelling use cases can enable productive future co-design collaborations. This feedback loop is mutually beneficial in providing direct motivation for further device development and encouraging system designers to integrate more energy-efficient, highly dense on-chip memory.

#### 4.4.1 TRADE AREA EFFICIENCY FOR PERFORMANCE

One theme we can highlight across the architecture-driven case studies in Sections 4.1-4.3 is that the subset of characterized results that exhibit lower area efficiency (i.e., internal array architectures



**Figure 4.15:** Masking write latency or reducing write traffic via introduction of a write caching scheme could make a broader set of eNVM technologies viable, taken from <sup>185</sup>.

that do less amortization of periphery and sensing overhead) also tend to result in lower total memory latency across many traffic scenarios. This is perhaps counter-intuitive given the effort spent in the devices community to manufacture very small cell sizes. We also note that in Figure 4.14, where such design points are highlighted across the plots, that slight advantages in terms of energy-per-access (e.g., Opt. STT and PCM compared to FeFET) tend to correlate to large total power advantages in high-traffic scenarios. As such, pointing out to device designers the greater relative impact of reduced energy per access rather than decreased cell size could usher in a more productive, product-ready set of eNVM technologies. Additionally, we observe that reducing energy per write access for STT and RRAM would drastically improve their relative power advantage for data-intensive applications, even at a cost of relatively lower area efficiency or storage density.

#### 4.4.2 WRITE BUFFERING CHANGES THE PERFORMANCE LANDSCAPE

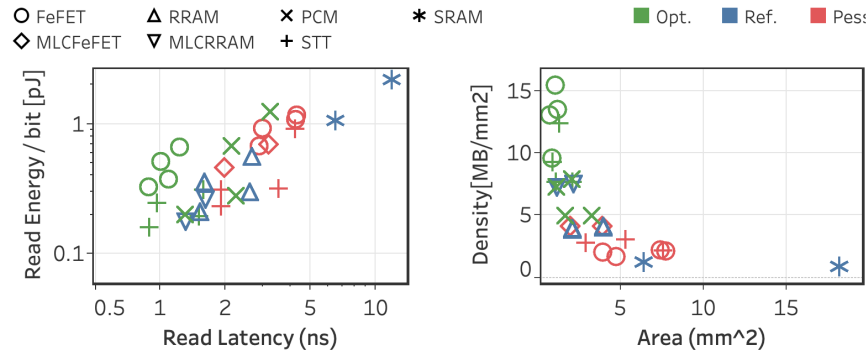
In conjunction with technology innovations to reduce write latency, adoption of a wider set of eNVMs in general-purpose computing contexts could be made possible by employing existing architectural techniques to mask poor write characteristics. For example, in an effort to extend memory lifetime and mask the performance impact of write access, a more performant technology (e.g., SRAM,

or STT) could be employed as a write-buffer. Rather than employ a costly and engineering-intensive cycle-accurate simulator to gauge the impact of provisioning a write buffer, NVMEExplorer enables an analytical study under user-specified traffic patterns to narrow the space of eNVMs worthy of further simulation and design effort. This approach answers high-level questions regarding whether write-buffering could make a difference in making additional eNVMs viable for applications with significant write traffic, and, if so, how much benefit would need to be extracted?

For illustrative purposes, we consider a simple write cache that would hold write requests to the eNVM, write back to eNVM when the buffer is full, and allow in-place updates in the case of multiple writes to the same address before an update to eNVM. Figure 4.15 shows the results for this study for SPEC2017 and Facebook-Graph-BFS. Just buffering the writes will mask the effective write latency experienced by the system, while a write cache that allows updates could additionally reduce traffic and extend lifetime. In particular, we look at the effects of masking write latency and reducing write traffic on total memory latency and power. We observe that for Facebook-Graph-BFS, if the write traffic load is reduced by at least half, FeFET emerges as a performant option, while STT remains the lowest power solution for this particularly high-traffic workload. STT and RRAM are still the optimal technology choices for SPEC2017 in terms of performance, but write-buffering could empower FeFETs as a lower-power alternative if latency could be masked or write traffic to the eNVM could be reduced by at least 25%. Write buffering could additionally benefit RRAM-based solutions by effectively extending the memory lifetime in the face of otherwise-problematic endurance characteristics.

#### 4.4.3 REVISITING FAULT MODELS, ERROR MITIGATION

While programming multiple bits per memory cell is an important strategy for increasing storage density across many eNVMs, previous work has revealed that MLC eNVMs may exhibit significantly higher fault rates that must be carefully considered in conjunction with application re-



**Figure 4.16:** When we consider multi-level cells (MLC)s and filter out memory solutions that don't provide acceptable ResNet18 inference accuracy after fault injection, we note MLC RRAM is denser and more performant than SLC RRAM, while MLC FeFET is only sufficiently reliable for larger cell sizes (red), taken from <sup>185</sup>.

silience <sup>183,204</sup>. NVMEexplorer enables efficient and broad probing of reliability vs. storage density by providing an application-agnostic fault injection tool and templates for technology-specific fault models. To demonstrate, we quantify the application accuracy for ResNet18 image classification under weight storage in SLC vs. 2-bit MLC across multiple technologies for which there exists sufficient cell and circuit level data to produce detailed fault models. The density vs. reliability trade-off is distinct for each technology. For example, Figure 4.16 displays 8MB and 16MB characterized arrays, including 2-bit MLC RRAM and 2-bit MLC FeFET, filtered such that only those arrays meeting application latency requirements and maintaining image classification accuracy are included. Note that these results replicate previous efforts that indicate that image classification inference is robust to 2-bit MLC RRAM storage (we also verified this for CTT-based memories with fault modeling details provided in <sup>183,60</sup>), while we show that MLC FeFET devices only exhibit acceptable accuracy for larger cell sizes. This is because smaller FeFETs are more difficult to program reliably due to device-to-device variation <sup>204</sup>. Portions of NVMEexplorer were leveraged to quantify cell- and circuit-level trade-offs specific to MLC FeFETs in greater depth to determine optimal cell provisioning and writing schemes for target applications <sup>204</sup>.

*I never am really satisfied that I understand anything; because, understand it well as I may, my comprehension can only be an infinitesimal fraction of all I want to understand about the many connections and relations which occur to me.*

Ada Lovelace

# 5

## Conclusion: Future Memory System Opportunities and Innovations

AS TECHNOLOGY SCALING REACHES PHYSICAL LIMITS and critical applications become increasingly data-intensive, fundamental improvements in efficiency are bound to effective and innovative memory system design choices. This thesis has reviewed and revealed a variety of concrete de-

sign choices and system opportunities for improving memory density and efficiency, particularly highlighting the promise of integrating embedded non-volatile memory solutions in a variety of computing systems. In addition to specific system solutions and key design choices, a central contribution of this work is to develop and make available methodologies and tools to empower future exploration of technology innovations, application characteristics, system design, and the critical intersections and interactions of these choices across the computing stack.

Several key themes have emerged in the discussion of this dissertation which will continue to be critical to future memory system design. In this section, I will describe the themes of reliability, flexibility, and technology-awareness and how they can and should inform future work in heterogeneous memory systems. Additionally, I will introduce other innovations and design considerations, such as temperature-aware evaluation, floorplan-aware memory architecture design for accelerators, environmentally-aware analysis of technology and fabrication choices, and exciting integration choices such as 3D-stacked memory and chiplet-based systems, and describe how future opportunities in these research spaces intersect and are made more immediately explorable, tangible, and feasible by the methods, tools, and processes described in this thesis.

## 5.0 THEMES TO UNLOCK FUTURE MEMORY EFFICIENCY

In this section, I review several over-arching design principles that emerged through the discussion of my dissertation work, and I comment on how these themes and principles will continue to shape the field of computer architecture, including some specific projects and studies I have considered that are in-progress or otherwise outside the scope of this work.



## 5.0.0 RELIABILITY

Chapter 2 probes the many ways not only that technology-level and system-level choices (e.g., MLC programming, data encoding) have direct and potentially catastrophic impacts on target application (in terms of accuracy, performance, and more), but also that exposing and exploring the distinct application-level impacts of varying system and technology reliability in the memory system can unlock incredible benefits in terms of increased memory density and memory system efficiency. These findings were concretized and integrated in a broader set of memory system evaluations in Chapters 3 and 4. In fact, as computing systems naturally gravitate towards increased specialization and heterogeneity in the face of the end of Moore’s law, critical examination of reliability assumptions prompts a huge opportunity to *relax* and/or *customize* storage settings, data formats, and technology choices towards the specific needs and scope of particular applications and domains of interest.

This thesis has heavily focused on the cross-stack consideration of potentially decreased reliability in emerging memory technologies in conjunction with the relative fault tolerance of deep neural network inference as an application space relevant to myriad computing domains and systems. However, presented work in general-purpose compute settings (Chapter 4.3) and towards acceleration of graph processing and broader classes of machine learning (Chapters 4.2, 3.2, 3.3) suggest that the bleeding edge of memory efficiency in a variety of domains and system settings will require the proposed approach to re-thinking and re-calibrating resilience analysis and mitigation strategies in tight connection with memory technology design choices and application properties. While there is evidence that the specific software tools for fault modeling, fault injection, and incorporating reliability into system-level design objectives that I’ve contributed to and developed through my PhD work (namely, Ares<sup>193</sup> and nvmFI within NVMExplorer<sup>185</sup>) will continue to carry the frontier and enable fascinating and essential future studies in memory reliability, I hope that the exposure and discussion of the interactions between data format (including numerical representations and sparse

encodings) and low-level memory cell and circuit optimizations (e.g., MLC programming and sensing design) will exhibit lasting impact on future memory solutions.

### 5.0.1 FLEXIBILITY

While a significant portion of the system solutions and memory architectures proposed throughout this thesis are heavily customized towards a single application or a subset of an application space, an increasingly crucial vector of any proposed architecture is the balance of customization via co-design vs. flexibility to support a sufficient range of workloads. The identification and segmentation of a ‘sufficient range’ of supported applications or workloads is highly system-dependent, and increasingly heterogeneous systems require careful consideration of not only what to accelerate and dedicate computational units towards, but also where and how memory should be allocated, physically distributed on a chip, and accessed and shared among system resources to best support end-to-end applications. In this light, the definition of flexibility alludes not only to a range of application characteristics, but also to the programmability and abstraction built into a customized system in terms of system support (e.g., user interface, development stack, memory management protocols) and circuit-level support (e.g., dynamic allocation and selection of programming settings, perhaps based on reliability concerns).

Ongoing work aims to taxonomize and sufficiently standardize both the design process for the next generation of accelerators and the appropriate dataflow, memory allocation, and scheduling paradigms to maximize utilization and efficiency<sup>224</sup>. However, this flavor of co-design is as yet insufficient without critically examining data movement energy and opportunities for increased overall energy efficiency in the form of decoupled, physically distributed memory banks that are interchangeable and dynamically allocatable to emulate the memory hierarchy and partitioning that best supports a given workload of interest at compile-time. I’m investigating such a proposal in an orthogonal but conceptually relevant research effort that aims to identify where and when the the

fracturing and physical distribution of a memory array architecture is worthwhile. In this proposal, smaller, individually accessible and addressable buffer units are cheaper per-access, but the relative cost and lack of area efficiency of physically distributed memory resources among compute units must be overcome by a combination of sufficiently stripped-down network routing (made possible by mostly-static scheduling paradigms<sup>182</sup>) and effective data mapping to maximize data reuse to physically co-located data buffers. A key finding in this effort is that the individual buffer units, to maximize potential efficiency, must be flexibly able to be defined and partitioned according to application properties, and then data allocation must be optimally identified during compile-time, through a cross-computing-stack design methodology.

#### 5.0.2 TECHNOLOGY-AWARENESS

Across a variety of contexts and leveraging a range of underlying memory technologies, this thesis has reviewed and evaluated concrete strategies to expose or otherwise customize lower-level design choices (i.e., memory cell, circuit, array optimizations) to the broader system architecture and even to the design and optimization of an application of interest to unlock striking gains. While reliability and flexibility are two critical considerations that can and should be built into future design goals and metrics, the core of computer architecture innovation and memory system efficiency in the future will be exploring how, when, and to what extent to bring awareness of technology-level details into the fore. The results presented throughout this thesis unearth the ways in which this is exacerbated as one considers integrating eNVMs, but the perspective of co-design and technology-awareness is similarly critical in understanding, adapting, and deriving benefits from advanced system design, fabrication, and integration choices more broadly, as discussed next.

## 5.1 INNOVATIONS ON THE HORIZON, AND HOW TO LEVERAGE THEM

In this section, I briefly review opportunities and limitations for a set of broad future research thrusts, in each case contextualizing in terms of their viability for heterogeneous future memory systems and the efficacy of the methods and findings presented in this dissertation.

### 5.1.0 TEMPERATURE-AWARE SYSTEM ANALYSIS AND EVALUATION

One design parameter largely overlooked in the preceding chapters is the impact of temperature. There are several ways in which thermal effects may impact analysis of the viability, effectiveness, and efficiency of proposed memory system solutions. For example, recent work observed that hot-spots and uneven heat distribution in CPUs are exacerbated at advanced technology nodes (e.g., 7nm vs. 14nm or 16nm), leading to performance degradation, reliability issues, and potential damage to components<sup>85</sup>. Further study is required to know whether such issues may be compounded or possibly alleviated by replacing high-leakage-power SRAM with an eNVM, and whether sensitivity of temperature fluctuations would impact eNVM susceptibility to faults in different system contexts. Additionally, environmental conditions and cooling methods of the system can have significant performance and power implications as well. In fact, recent work has proposed cryogenic operation (i.e., cooling an entire chip via contact with liquid nitrogen) as a potentially power-efficient solution for off-chip DRAM and even for the CPU, drastically reducing the effective leakage and improving performance of the cache subsystem<sup>163</sup>.

As additional innovations and system proposals continue to emerge, it is essential to incorporate temperature-dependencies and to contextualize potential benefits in a broader system context. For example, future work should investigate whether potential power overheads of cooling can be sufficiently reduced to justify cryogenic operation, and how relative power benefits of cryo-operation may compare to the energy efficiency and low leakage of eNVM solutions for the LLC.

### 5.1.1 ENVIRONMENTAL-IMPACT-AWARE SYSTEM ANALYSIS AND EVALUATION

In the face of our changing climate, the indisputably high carbon footprint of fabrication, development, and use of technology should be front-of-mind of researchers aiming to shape and inform the next generation of computing systems. Moore's Law describing the number of transistors per integrated chip has always, at heart, been an economic argument, in terms of justifying the research and development towards increased density and performance, but the time is past for technology scaling efforts and innovations to be dually justified by economic incentives and environmental impacts. Ongoing work has identified design challenges and opportunities in quantifying carbon footprint and designing future devices with environmental impact in-the-loop<sup>80,245</sup>. The modeling efforts recently put forth to quantify environmental impacts of computing can and should be integrated with the design tools for memory systems put forth in this thesis.

As yet, the environmental impact both in terms of embodied carbon and operation-related footprint of devices integrating eNVMs has not been analyzed, and shaping optimization goals and metrics to include minimization of carbon emissions is a crucial and imminent change in mindset. Quantifying and justifying changes to manufacturing, procurement of unique materials, and integration choices become central questions in judging the viability, promise, and limitations of each of the eNVMs studied in this thesis. For example, one proposal to elide the costs and ensuing carbon footprint of fabrication in advanced technology nodes is to instead roll back to a less advanced node (e.g., 22nm-45nm) for the dual benefit of reduced complexity and resources in manufacturing, as well as research and development, and lower power density during operation. Considering the open questions in terms of efficient scaling across memory technologies and system solutions, there is much ground to cover and many delicate, cross-computing-stack interactions to consider in reducing carbon footprint of end-to-end systems. Namely, increasing conversation and collaboration from materials engineers and industry fabrication plants up to computer architects and application

developers could change future computing systems by leveraging device-workload interactions and identifying potential concessions in the hallowed metrics of performance and cost in order build greener technology.

### 5.1.2 3D-INTEGRATED MEMORY

3D-stacked dies as well as monolithic 3D integration of memory (either standalone or integrated with compute layers) exhibits exciting potential performance and density benefits, but comes with design challenges<sup>41,73</sup>. Probing this trade-off per-memory-technology is a critical first step, but realistic analysis of end-to-end scalability, efficiency, and environmental impact will be needed. Many proposals already exist in this space and enable dramatic bandwidth and significant performance, but such systems must be carefully calibrated to identify when and where it is an appealing design choice (e.g., in light of potential thermal implication, potential additional costs, and more). There are ample, near-future opportunities to identify co-design choices in the space of 3D-stacked and monolithic 3D implementations of various memory technologies to augment the memory subsystem of both general purpose and highly specialized computing systems.

### 5.1.3 CHIPLETS – IMPLICATIONS FOR MEMORY DESIGN SPACE EXPLORATION

There are many well-documented limitations of large, monolithic chip designs, both in terms of yield and fabrication process at advanced technology nodes and in the drastically increasing complexity, time, and engineering effort involved to design and validate such a design. Multi-Chip Modules (MCMs), is a compelling emerging alternative design paradigm comprised of individually designed and fabricated chiplets, then integrated into a larger system<sup>202</sup>. A chiplet-based design introduces challenges in the efficiency of inter-chiplet communication, but has been demonstrated to scale well in supporting applications requiring beefy compute and high on-chip memory capacity, as

for deep neural network inference.

A chiplet-based integration strategy is also an opportunity to combine and communicate among distinct chip designs, including those fabricated with different processes or those dedicated for different designs and purposes (e.g., a subset of distributed chiplets for technologically heterogeneous memory resources, such as those integrating different NVM technologies). For this reason, MCMs are a compelling design paradigm for integrating and making efficient use of tightly-coupled eNVM resources in a large system, and the proposed tools and methodologies in this dissertation could be extended to model the interactions and access properties of a chiplet-based design. The relevance of varying reliability, flexibility, and technology-awareness are heightened in this context. The critical intersections of application access characteristics, data formats, and network properties unveiled in this work will remain relevant to these critical design space explorations and system developments.

## 5.2 SUMMARY: CROSS-COMPUTING-STACK MEMORY EFFICIENCY OPPORTUNITIES

Future memory system efficiency will clearly be guided by reliability, flexibility, and technology-awareness. The work presented in this dissertation can be effectively complemented by improved metrics and methodologies (e.g., quantifying environmental impact of design choices, incorporating temperature-awareness and integration innovations). Continuing the attitudes and design methods developed in this thesis will be critical to transformations and efficiency gains in future memory systems. In particular, exposing technology characteristics to higher levels of the computing stack necessitates conversation and approachable methods for collaborative design among application experts, system architects, circuit designers, device physicists, and more. Building accessible, extensible, and thoughtful frameworks and software tools for design space exploration and technology modeling is a bridge towards a more diverse and creative era of memory architecture solutions, to which I'm excited to shape and contribute as a researcher.

# References

- [1] (2015). Keras: The python deep learning library.
- [2] (2017). Solid state drive (ssd) requirements and endurance test method. <https://www.jedec.org/standards-documents/focus/flash/solid-state-drives>.
- [3] Aggarwal, S., Almasi, H., DeHerrera, M., Hughes, B., Ikegawa, S., Janesky, J., Lee, H. K., Lu, H., Mancoff, F. B., Nagel, K., Shimon, G., Sun, J. J., Andre, T., & Alam, S. M. (2019). Demonstration of a reliable 1 gb standalone spin-transfer torque mram for industrial applications. In *IEEE International Electron Devices Meeting (IEDM)*.
- [4] Alayan, M., Vianello, E., Navarro, G., Carabasse, C., Barbera, S. L., Verdy, A., Castellani, N., Levisse, A., Molas, G., Grenouillet, L., Magis, T., Aussenac, F., Bernard, M., DeSalvo, B., Portal, J. M., & Nowak, E. (2017). In-depth investigation of programming and reading operations in rram cells integrated with ovonic threshold switching (ots) selectors. In *IEEE International Electron Devices Meeting (IEDM)*.
- [5] Ali, T., Polakowski, P., Kuhnel, K., Czernohorsky, M., Kampfe, T., Rudolph, M., Patzold, B., Lehninger, D., Muller, F., Olivo, R., Lederer, M., Hoffmann, R., Steinke, P., Zimmermann, K., Muhle, U., Seidel, K., & Muller, J. (2019). A multilevel fefet memory device based on laminated hso and hzo ferroelectric layers for high-density storage. In *IEEE International Electron Devices Meeting (IEDM)*.
- [6] Ali, T., Seidel, K., Kuhnel, K., Rudolph, M., Czernohorsky, M., Mertens, K., Hoffmann, R., Zimmermann, K., Muhle, U., Muller, J., Van Houdt, J., & Eng, L. M. (2020). A novel dual ferroelectric layer based mfmfis fefet with optimal stack tuning toward low power and high-speed nvm for neuromorphic applications. In *IEEE Symposium on VLSI Technology*.
- [7] Alzate, J. G., Arslan, U., Bai, P., Brockman, J., Chen, Y. J., Das, N., Fischer, K., Ghani, T., Heil, P., Hentges, P., Jahan, R., Littlejohn, A., Mainuddin, M., Ouellette, D., Pellegren, J., Pramanik, T., Puls, C., Quintero, P., Rahman, T., Sekhar, M., Sell, B., Seth, M., Smith, A. J., Smith, A. K., Wei, L., Wiegand, C., Golonzka, O., & Hamzaoglu, F. (2019). 2 mb array-level demonstration of stt-mram process and performance towards l4 cache applications. In *IEEE International Electron Devices Meeting (IEDM)*.



- [8] Ando, K., Ueyoshi, K., Orimo, K., Yonekawa, H., Sato, S., Nakahara, H., Ikebe, M., Asai, T., Takamaeda-Yamazaki, S., Kuroda, T., & Motomura, M. (2017). Brein memory: A 13-layer 4.2 k neuron/0.8 m synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm cmos. In *2017 Symposium on VLSI Circuits*.
- [9] Andrews, J. & Baker, N. (2006). Xbox 360 system architecture. *IEEE Micro*, 26(2), 25–37.
- [10] Arnaud, F., Zuliani, P., Reynard, J. P., Gandolfo, A., Disegni, F., Mattavelli, P., Gomiero, E., Samanni, G., Jahan, C., Berthelon, R., Weber, O., Richard, E., Barral, V., Villaret, A., Kohler, S., Grenier, J. C., Ranica, R., Gallon, C., Souhaite, A., Ristoiu, D., Favennec, L., Caubet, V., Delmedico, S., Cherault, N., Beneyton, R., Chouteau, S., Sassoulas, P. O., Vernhet, A., Le Fric, Y., Domengie, F., Scotti, L., Pacelli, D., Ogier, J. L., Boucard, F., Lagrasta, S., Benoit, D., Clement, L., Boivin, P., Ferreira, P., Annunziata, R., & Cappelletti, P. (2018). Truly innovative 28nm fdsoi technology for automotive micro-controller applications embedding 16mb phase change memory. In *IEEE International Electron Devices Meeting (IEDM)*.
- [11] Asiatici, M. & Ienne, P. (2021). Large-scale graph processing on fpgas with caches for thousands of simultaneous misses. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)* (pp. 609–622).
- [12] Baek, J., Kim, S., Park, J., Park, J., & Kwon, K. (2015). A reliable cross-point mlc reram with sneak current compensation. In *2015 IEEE International Memory Workshop (IMW)*.
- [13] Balaji, V., Crago, N., Jaleel, A., & Lucia, B. (2021). P-opt: Practical optimal cache replacement for graph analytics. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (pp. 668–681).
- [14] Bankman, D., Yang, L., Moons, B., Verhelst, M., & Murmann, B. (2018). An always-on cifar-10 mixed-signal binary cnn processor with all memory on chip in 28nm cmos. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*.
- [15] Barlas, M., Grossi, A., Grenouillet, L., Vianello, E., Nolot, E., Vaxelaire, N., Blaise, P., Traoré, B., Coignus, J., Perrin, F., Crochemore, R., Mazen, F., Lachal, L., Pauliac, S., Pellissier, C., Bernasconi, S., Chevalliez, S., Nodin, J. F., Perniola, L., & Nowak, E. (2017). Improvement of hfo2 based rram array performances by local si implantation. In *IEEE International Electron Devices Meeting (IEDM)*.
- [16] Bayram, I., Eken, E., Kline, D., Parshook, N., Chen, Y., & Jones, A. K. (2016). Modeling stt-ram fabrication cost and impacts in nvsim. In *2016 Seventh International Green and Sustainable Computing Conference (IGSC)*.
- [17] Beamer, S., Asanovic, K., & Patterson, D. (2015). Locality exists in graph processing: Workload characterization on an ivy bridge server. In *IEEE International Symposium on Workload Characterization*.

- [18] Belmonte, A., Radhakrishnan, J., Goux, L., Donadio, G. L., Kumbhare, P., Redolfi, A., Delhougne, R., Nyns, L., Devulder, W., Witters, T., Covello, A., Vereecke, G., Franquet, A., Spampinato, V., Kundu, S., Mao, M., Hody, H., & Kar, G. S. (2019). Co active electrode enhances cbram performance and scaling potential. In *IEEE International Electron Devices Meeting (IEDM)*.
- [19] Bi, X., Mao, M., Wang, D., & Li, H. H. (2017). Cross-layer optimization for multilevel cell stt-ram caches. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*.
- [20] Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., Hestness, J., Hower, D. R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M. D., & Wood, D. A. (2011). The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2), 1–7.
- [21] Bojnordi, M. N. & Ipek, E. (2016). Memristive boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*.
- [22] Boujamaa, E. M., Ali, S. M., Wandji, S. N., Gourio, A., Pyo, S., Koh, G., Song, Y., Song, T., Kye, J., Vial, J. C., Sowden, A., Rathor, M., & Dray, C. (2020). A 14.7mb/mm<sup>2</sup> 28nm fdsoi stt-mram with current starved read path, 52ohm/sigma offset voltage sense amplifier and fully trimmable ctat reference. In *IEEE Symposium on VLSI Circuits*.
- [23] Bricalli, A., Ambrosi, E., Laudato, M., Maestro, M., Rodriguez, R., & Ielmini, D. (2016). Sio<sub>x</sub>-based resistive switching memory (rram) for crossbar storage/select elements with high on/off ratio. In *IEEE International Electron Devices Meeting (IEDM)*.
- [24] Bucek, J., Lange, K.-D., & v. Kistowski, J. (2018). Spec cpu2017: Next-generation compute benchmark. In *ACM/SPEC International Conference on Performance Engineering*.
- [25] Carboni, R., Ambrogio, S., Chen, W., Siddik, M., Harms, J., Lyle, A., Kula, W., Sandhu, G., & Ielmini, D. (2016). Understanding cycling endurance in perpendicular spin-transfer torque (p-stt) magnetic memory. In *IEEE International Electron Devices Meeting (IEDM)*.
- [26] Carlson, T. E., Heirman, W., Eyerman, S., Hur, I., & Eeckhout, L. (2014). An evaluation of high-level mechanistic core models. *ACM Transactions on Architecture and Code Optimization (TACO)*.
- [27] Chan, C. Y., Chen, K. Y., Peng, H. K., & Wu, Y. H. (2020). Fefet memory featuring large memory window and robust endurance of long-pulse cycling by interface engineering using high-k alon. In *IEEE Symposium on VLSI Technology*.
- [28] Chandrasekar, K., Weis, C., Li, Y., Goossens, S., Jung, M., Naji, O., Akesson, B., Wehn, N., & Goossens, K. (2012). Drampower: Open-source dram power and energy estimation tool. <http://www.drampower.info>.

- [29] Chang, M., Wu, J., Chien, T., Liu, Y., Yang, T., Shen, W., King, Y., Lin, C., Lin, K., Chih, Y., Natarajan, S., & Chang, J. (2014). 19.4 embedded 1mb reram in 28nm cmos with 0.27-to-1v read using swing-sample-and-couple sense amplifier and self-boost-write-termination scheme. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*.
- [30] Chang, T., Chiu, Y., Lee, C., Hung, J., Chang, K., Xue, C., Wu, S., Kao, H., Chen, P., Huang, H., Teng, S., & Chang, M. (2020). A 22nm 1mb 1024b-read and near-memory-computing dual-mode stt-mram macro with 42.6gb/s read bandwidth for security-aware mobile devices. In *IEEE International Solid-State Circuits Conference - (ISSCC)*.
- [31] Chen, A. (2016). A review of emerging non-volatile memory (nvm) technologies and applications. *Solid-State Electronics*. Extended papers selected from ESSDERC 2015.
- [32] Chen, E., Apalkov, D., Diao, Z., Driskill-Smith, A., Druist, D., Lottis, D., Nikitin, V., Tang, X., Watts, S., Wang, S., Wolf, S. A., Ghosh, A. W., Lu, J. W., Poon, S. J., Stan, M., Butler, W. H., Gupta, S., Mewes, C. K. A., Mewes, T., & Visscher, P. B. (2010). Advances and future prospects of spin-transfer torque random access memory. *IEEE Transactions on Magnetics*.
- [33] Chen, J., Wu, H., Gao, B., Tang, J., Hu, X. S., & Qian, H. (2020). A parallel multibit programming scheme with high precision for rram-based neuromorphic systems. *IEEE T-ED*.
- [34] Chen, P., Peng, X., & Yu, S. (2017a). Neurosim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures. In *IEEE International Electron Devices Meeting (IEDM)*.
- [35] Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y., & Temam, O. (2014). Dianna: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In *ASPLOS*.
- [36] Chen, W., Lin, W., Lai, L., Li, S., Hsu, C., Lin, H., Lee, H., Su, J., Xie, Y., Sheu, S., & Chang, M. (2017b). A 16mb dual-mode reram macro with sub-14ns computing-in-memory and memory functions enabled by self-write termination scheme. In *IEEE International Electron Devices Meeting (IEDM)*.
- [37] Chen, W., Lu, W., Long, B., Li, Y., Gilmer, D., Bersuker, G., Bhunia, S., & Jha, R. (2015). Switching characteristics of w/zr/hfo<sub>2</sub>/tin reram devices for multi-level cell non-volatile memory applications. *Semiconductor Science and Technology*.
- [38] Chen, W. H., Li, K. X., Lin, W. Y., Hsu, K. H., Li, P. Y., Yang, C. H., Xue, C. X., Yang, E. Y., Chen, Y. K., Chang, Y. S., Hsu, T. H., King, Y. C., Lin, C. J., Liu, R. S., Hsieh, C. C., Tang, K. T., & Chang, M. F. (2018). A 65nm 1mb nonvolatile computing-in-memory reram macro with sub-16ns multiply-and-accumulate for binary dnn ai edge processors. In *IEEE International Solid-State Circuits Conference - (ISSCC)*.

- [39] Chen, X., Huang, T., Xu, S., Bourgeat, T., Chung, C., & Arvind (2021). *FlexMiner: A Pattern-Aware Accelerator for Graph Pattern Mining*. IEEE Press.
- [40] Cheng, H. Y., Chien, W. C., Kuo, I. T., Lai, E. K., Zhu, Y., Jordan-Sweet, J. L., Ray, A., Carta, F., Lee, F. M., Tseng, P. H., Lee, M. H., Lin, Y. Y., Kim, W., Bruce, R., Yeh, C. W., Yang, C. H., BrightSky, M., & Lung, H. L. (2017). An ultra high endurance and thermally stable selector based on teasesise chalcogenides compatible with beol ic integration for cross-point pcm. In *IEEE International Electron Devices Meeting (IEDM)*.
- [41] Cheng, H. Y., Kuo, I. T., Chien, W. C., Yeh, C. W., Chou, Y. C., Gong, N., Gignac, L., Yang, C. H., Cheng, C. W., Lavoie, C., Hopstaken, M., Bruce, R. L., Buzi, L., Lai, E. K., Carta, F., Ray, A., Lee, M. H., Ho, H. Y., Kim, W., BrightSky, M., & Lung, H. L. (2020). Si incorporation into assege chalcogenides for high thermal stability, high endurance and extremely low vth drift 3d stackable cross-point memory. In *IEEE Symposium on VLSI Technology*.
- [42] Chi, P., Li, S., Xu, C., Zhang, T., Zhao, J., Liu, Y., Wang, Y., & Xie, Y. (2016). Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. *ACM SIGARCH Computer Architecture News*.
- [43] Chien, W. C., Cheng, H. Y., BrightSky, M., Ray, A., Yeh, C. W., Kim, W., Bruce, R., Zhu, Y., Ho, H. Y., Lung, H. L., & Lam, C. (2016). Reliability study of a 128mb phase change memory chip implemented with doped gasbge with extraordinary thermal stability. In *IEEE International Electron Devices Meeting (IEDM)*.
- [44] Chih, Y., Shih, Y., Lee, C., Chang, Y., Lee, P., Lin, H., Chen, Y., Lo, C., Shih, M., Shen, K., Chuang, H., & Chang, T. J. (2020). A 22nm 32mb embedded stt-mram with 10ns read speed, 1m cycle write endurance, 10 years retention at 150c and high immunity to magnetic field interference. In *IEEE International Solid-State Circuits Conference - (ISSCC)*.
- [45] Chiu, Y.-C., Hu, H.-W., Lai, L.-Y., Huang, T.-Y., Kao, H.-Y., Chang, K.-T., Ho, M.-S., Chou, C.-C., Chih, Y.-D., Chang, T.-Y., & Chang, M.-F. (2019). A 40nm 2mb reram macro with 85% reduction in forming time and 99% reduction in page-write time using auto-forming and auto-write schemes. In *Symposium on VLSI Technology*.
- [46] Choi, Y., Song, I., Park, M., Chung, H., Chang, S., Cho, B., Kim, J., Oh, Y., Kwon, D., Sunwoo, J., Shin, J., Rho, Y., Lee, C., Kang, M. G., Lee, J., Kwon, Y., Kim, S., Kim, J., Lee, Y., Wang, Q., Cha, S., Ahn, S., Horii, H., Lee, J., Kim, K., Joo, H., Lee, K., Lee, Y., Yoo, J., & Jeong, G. (2012a). A 20nm 1.8v 8gb pram with 40mb/s program bandwidth. In *2012 IEEE International Solid-State Circuits Conference*.
- [47] Choi, Y., Song, I., Park, M. H., Chung, H., Chang, S., Cho, B., Kim, J., Oh, Y., Kwon, D., Sunwoo, J., Shin, J., Rho, Y., Lee, C., Kang, M. G., Lee, J., Kwon, Y., Kim, S., Kim, J., Lee, Y. J., Wang, Q., Cha, S., Ahn, S., Horii, H., Lee, J., Kim, K., Joo, H., Lee, K., Lee, Y. T., Yoo,

- J., & Jeong, G. (2012b). A 20nm 1.8v 8gb pram with 40mb/s program bandwidth. In *2012 IEEE International Solid-State Circuits Conference*.
- [48] Chou, C., Lin, Z., Lai, C., Su, C., Tseng, P., Chen, W., Tsai, W., Chu, W., Ong, T., Chuang, H., Chih, Y., & Chang, T. (2020). A 22nm 96kx144 rram macro with a self-tracking reference and a low ripple charge pump to achieve a configurable read window and a wide operating voltage range. In *IEEE Symposium on VLSI Circuits*.
- [49] Chou, C., Lin, Z., Tseng, P., Li, C., Chang, C., Chen, W., Chih, Y., & Chang, T. J. (2018). An n40 256kx44 embedded rram macro with sl-precharge sa and low-voltage current limiter to improve read and write performance. In *IEEE International Solid - State Circuits Conference - (ISSCC)*.
- [50] Chou, T., Tang, W., Botimer, J., & Zhang, Z. (2019). Cascade: Connecting rrams to extend analog dataflow in an end-to-end in-memory processing paradigm. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*.
- [51] Chung, S., Kishi, T., Park, J. W., Yoshikawa, M., Park, K. S., Nagase, T., Sunouchi, K., Kanaya, H., Kim, G. C., Noma, K., Lee, M. S., Yamamoto, A., Rho, K. M., Tsuchida, K., Chung, S. J., Yi, J. Y., Kim, H. S., Chun, Y. S., Oyamatsu, H., & Hong, S. J. (2016). 4gbit density stt-mram using perpendicular mtj realized with compact cell structure. In *IEEE International Electron Devices Meeting (IEDM)*.
- [52] Close, G. F., Frey, U., Morrish, J., Jordan, R., Lewis, S. C., Maffitt, T., BrightSky, M. J., Hagleitner, C., Lam, C. H., & Eleftheriou, E. (2013). A 256-mcell phase-change memory chip operating at 2+bit/cell. *IEEE Transactions on Circuits and Systems I: Regular Papers*.
- [53] Cong Xu, Dimin Niu, Muralimanohar, N., Jouppi, N. P., & Yuan Xie (2013). Understanding the trade-offs in multi-level cell rram memory design. In *Design Automation Conference (DAC)*.
- [54] Dadu, V., Liu, S., & Nowatzki, T. (2021). Polygraph: Exposing the value of flexibility for graph processing accelerators. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)* (pp. 595–608).
- [55] Dadu, V., Liu, S., & Nowatzki, T. (2022). Systematically understanding graph accelerator dimensions and the value of hardware flexibility. *IEEE Micro*, (pp. 1–1).
- [56] Daly, D. C., Fujino, L. C., & Smith, K. C. (2018). Through the looking glass - the 2018 edition: Trends in solid-state circuits from the 65th isscc. *IEEE Solid-State Circuits Magazine*, 10(1), 30–46.
- [57] Datta, D., Dixit, H., Agarwal, S., Dasgupta, A., Tran, M., Houssameddine, D., Chauhan, Y. S., Shum, D., & Benistant, F. (2017). Quantitative model for switching asymmetry in

- perpendicular mtj: A material-device-circuit co-design. In *IEEE International Electron Devices Meeting (IEDM)*.
- [58] Deng, S., Yin, G., Chakraborty, W., Dutta, S., Datta, S., Li, X., & Ni, K. (2020). A comprehensive model for ferroelectric fet capturing the key behaviors: Scalability, variation, stochasticity, and accumulation. In *IEEE Symposium on VLSI Technology*.
- [59] Donato, M., Pentecost, L., Brooks, D., & Wei, G. (2019). Memti: Optimizing on-chip non-volatile storage for visual multitask inference at the edge. *IEEE Micro*.
- [60] Donato, M., Reagen, B., Pentecost, L., Gupta, U., Brooks, D., & Wei, G.-Y. (2018). On-chip deep neural network storage with multi-level envm. In *Proceedings of the 55th Annual Design Automation Conference*.
- [61] Dong, Q., Wang, Z., Lim, J., Zhang, Y., Shih, Y., Chih, Y., Chang, J., Blaauw, D., & Sylvester, D. (2018). A 1mb 28nm stt-mram with 2.8ns read access time at 1.2v vdd using single-cap offset-cancelled sense amplifier and in-situ self-write-termination. In *IEEE International Solid - State Circuits Conference - (ISSCC)*.
- [62] Dong, X., Xu, C., Xie, Y., & Jouppi, N. P. (2012). Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.
- [63] Du, Y., Du, L., Gu, X., Wang, X., & Chang, M. F. (2017). A memristive neural network computing engine using cmos-compatible charge-trap-transistor (CTT). *CoRR*.
- [64] Dunkel, S., Trentzsch, M., Richter, R., Moll, P., Fuchs, C., Gehring, O., Majer, M., Wittek, S., Muller, B., Melde, T., Mulaosmanovic, H., Slesazeck, S., Müller, S., Ocker, J., Noack, M., Löhr, D. ., Polakowski, P., Muller, J., Mikolajick, T., Hontschel, J., Rice, B., Pellerin, J., & Beyer, S. (2017). A fefet based super-low-power ultra-fast embedded nvm technology for 22nm fdsoi and beyond. In *IEEE International Electron Devices Meeting (IEDM)*.
- [65] Endoh, T., Honjo, H., Nishioka, K., & Ikeda, S. (2020). Recent progresses in stt-mram and sot-mram for next generation mram. In *IEEE Symposium on VLSI Technology*.
- [66] Esmaeilzadeh, H., Blem, E., Amant, R. S., Sankaralingam, K., & Burger, D. (2011). Dark silicon and the end of multicore scaling. In *38th Annual International Symposium on Computer Architecture (ISCA)*.
- [67] Facebook Technologies (2019). Oculus guidelines for virtual reality performance optimization. <https://developer.oculus.com/documentation/pcsdk/latest/concepts/dg-performance-guidelines/>.
- [68] Feng, X., Li, Y., Wang, L., Yu, Z. G., Chen, S., Tan, W. C., Macadam, N., Hu, G., Gong, X., Hasan, T., Zhang, Y. W., Thean, A. V. Y., & Ang, K. W. (2019). First demonstration

of a fully-printed mos2rram on flexible substrate with ultra-low switching voltage and its application as electronic synapse. In *Symposium on VLSI Technology*.

- [69] Florent, K., Pesic, M., Subirats, A., Banerjee, K., Lavizzari, S., Arreghini, A., Di Piazza, L., Potoms, G., Sebaai, F., McMitchell, S. R. C., Popovici, M., Groeseneken, G., & Van Houdt, J. (2018). Vertical ferroelectric hfo<sub>2</sub> fet based on 3d nand architecture: Towards dense low-power memory. In *IEEE International Electron Devices Meeting (IEDM)*.
- [70] Fujii, S., Kamimuta, Y., Ino, T., Nakasaki, Y., Takaishi, R., & Saitoh, M. (2016). First demonstration and performance improvement of ferroelectric hfo<sub>2</sub>-based resistive switch with low operation current and intrinsic diode property. In *IEEE Symposium on VLSI Technology*.
- [71] Fukami, S., Anekawa, T., Ohkawara, A., Zhang, C., & Ohno, H. (2016). A sub-ns three-terminal spin-orbit torque induced switching device. In *IEEE Symposium on VLSI Technology*.
- [72] Gallagher, W. J., Chien, E., Chiang, T., Huang, J., Shih, M., Wang, C. Y., Weng, C., Chen, S., Bair, C., Lee, G., Shih, Y., Lee, C., Lee, P., Wang, R., Shen, K. H., Wu, J. J., Wang, W., & Chuang, H. (2019). 22nm stt-mram for reflow and automotive uses with high yield, reliability, and magnetic immunity and with performance and shielding options. In *IEEE International Electron Devices Meeting (IEDM)*.
- [73] Gao, M., Pu, J., Yang, X., Horowitz, M., & Kozyrakis, C. (2017). Tetris: Scalable and efficient neural network acceleration with 3d memory. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '17*.
- [74] Giannopoulos, I., Sebastian, A., Le Gallo, M., Jonnalagadda, V. P., Sousa, M., Boon, M. N., & Eleftheriou, E. (2018). 8-bit precision in-memory multiplication with projected phase-change memory. In *IEEE International Electron Devices Meeting (IEDM)*.
- [75] Gokhale, M., Lloyd, S., & Macaraeg, C. (2015). Hybrid memory cube performance characterization on data-centric workloads. In *Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms, IA3 '15* New York, NY, USA: Association for Computing Machinery.
- [76] Golonzka, O., Alzate, J., Arslan, U., Bohr, M., Bai, P., Brockman, J., Buford, B., Connor, C., Das, N., Doyle, B., Ghani, T., Hamzaoglu, F., Heil, P., Hentges, P., Jahan, R., Kencke, D., Lin, B., Lu, M., Mainuddin, M., Meterelliyoz, M., Nguyen, P., Nikonov, D., O'brien, K., Donnell, J. O., Oguz, K., Ouellette, D., Park, J., Pellegren, J., Puls, C., Quintero, P., Rahman, T., Romang, A., Sekhar, M., Selarka, A., Seth, M., Smith, A. J., Smith, A. K., Wei, L., Wiegand, C., Zhang, Z., & Fischer, K. (2018). Mram as embedded non-volatile memory solution for 22fl finfet technology. In *IEEE International Electron Devices Meeting (IEDM)*.

- [77] Golonzka, O., Arslan, U., Bai, P., Bohr, M., Baykan, O., Chang, Y., Chaudhari, A., Chen, A., Clarke, J., Connor, C., Das, N., English, C., Ghani, T., Hamzaoglu, F., Hentges, P., Jain, P., Jezewski, C., Karpov, I., Kothari, H., Kotlyar, R., Lin, B., Metz, M., Odonnell, J., Ouellette, D., Park, J., Pirkle, A., Quintero, P., Seghete, D., Sekhar, M., Gupta, A. S., Seth, M., Strutt, N., Wiegand, C., Yoo, H. J., & Fischer, K. (2019). Non-volatile rram embedded into 22ffl finfet technology. In *Symposium on VLSI Technology*.
- [78] Gong, N., Chien, W., Chou, Y., Yeh, C., Li, N., Cheng, H., Cheng, C., Kuo, I., Yang, C., Bruce, R., Ray, A., Gignac, L., Lin, Y., Miller, C., Perri, T., Kim, W., Buzi, L., Utomo, H., Carta, F., Lai, E., Ho, H., Lung, H., & BrightSky, M. (2020). A no-verification multi-level-cell (mlc) operation in cross-point ots-pcm. In *IEEE Symposium on VLSI Technology*.
- [79] Goux, L., Belmonte, A., Celano, U., Woo, J., Folkersma, S., Chen, C. Y., Redolfi, A., Fantini, A., Degraeve, R., Clima, S., Vandervorst, W., & Jurczak, M. (2016). Retention, disturb and variability improvements enabled by local chemical-potential tuning and controlled hour-glass filament shape in a novel wwo<sub>3</sub>al<sub>2</sub>o<sub>3</sub>cu cbram. In *IEEE Symposium on VLSI Technology*.
- [80] Gupta, U., Kim, Y. G., Lee, S., Tse, J., Lee, H. S., Wei, G., Brooks, D., & Wu, C. (2020). Chasing carbon: The elusive environmental footprint of computing. *CoRR*, abs/2011.02839.
- [81] Gupta, U., Reagen, B., Pentecost, L., Donato, M., Tambe, T., Rush, A. M., Wei, G.-Y., & Brooks, D. (2019). Masr: A modular accelerator for sparse rnns. In *2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT)* (pp. 1–14).
- [82] Haldas, M. (2019). Cctv camera recording video frame rate comparison. <https://www.cctvcamerapros.com/CCTV-Video-Frame-Rate-Comparison-s/739.htm>.
- [83] Ham, T. J., Wu, L., Sundaram, N., Satish, N., & Martonosi, M. (2016). Graphicionado: A high-performance and energy-efficient accelerator for graph analytics. In *49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.
- [84] Hankin, A., Shapira, T., Sangaiyah, K., Lui, M., & Hempstead, M. (2019). Evaluation of non-volatile memory based last level cache given modern use case behavior. In *IEEE International Symposium on Workload Characterization (IISWC)*.
- [85] Hankin, A., Werner, D., Amiraski, M., Sebot, J., Vaidyanathan, K., & Hempstead, M. (2021). Hotgauge: A methodology for characterizing advanced hotspots in modern and next generation processors. In *2021 IEEE International Symposium on Workload Characterization (IISWC)* (pp. 163–175).
- [86] Hari, S. K. S., Tsai, T., Stephenson, M., Keckler, S. W., & Emer, J. (2017). Sassifi: An architecture-level fault injection tool for gpu application resilience evaluation. In *2017 IEEE*



*International Symposium on Performance Analysis of Systems and Software (ISPASS)* (pp. 249–258).

- [87] Hazelwood, K., Bird, S., Brooks, D., Chintala, S., Diril, U., Dzhulgakov, D., Fawzy, M., Jia, B., Jia, Y., Kalro, A., Law, J., Lee, K., Lu, J., Noordhuis, P., Smelyanskiy, M., Xiong, L., & Wang, X. (2018). Applied machine learning at facebook: A datacenter infrastructure perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*.
- [88] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *CoRR*.
- [89] He, Z., Angizi, S., & Fan, D. (2017). Exploring stt-mram based in-memory computing paradigm with application of image edge extraction. In *2017 IEEE International Conference on Computer Design (ICCD)*.
- [90] Hennessy, J. L. & Patterson, D. A. (2012). *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 5 edition.
- [91] Hewlett Packard (2017). Cacti 7.0.
- [92] Ho, C., Chang, S., Huang, C., Chuang, Y., Lim, S., Hsieh, M., Chang, S., & Liao, H. (2017). Integrated hfo2-rram to achieve highly reliable, greener, faster, cost-effective, and scaled devices. In *IEEE International Electron Devices Meeting (IEDM)*.
- [93] Honjo, H., Nguyen, T. V. A., Watanabe, T., Nasuno, T., Zhang, C., Tanigawa, T., Miura, S., Inoue, H., Niwa, M., Yoshiduka, T., Noguchi, Y., Yasuhira, M., Tamakoshi, A., Natsui, M., Ma, Y., Koike, H., Takahashi, Y., Furuya, K., Shen, H., Fukami, S., Sato, H., Ikeda, S., Hanyu, T., Ohno, H., & Endoh, T. (2019). First demonstration of field-free sot-mram with 0.35 ns write speed and 70 thermal stability under 400c thermal tolerance by canted sot structure and its advanced patterning/sot channel technology. In *IEEE International Electron Devices Meeting (IEDM)*.
- [94] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861.
- [95] Hsieh, E., Zheng, X., Nelson, M., Le, B., Wong, H., Mitra, S., Wong, S., Giordano, M., Hodson, B., Levy, A., Osekowsky, S., Radway, R., Shih, Y., Wan, W., & Wu, T. (2019). High-density multiple bits-per-cell 1t4r rram array with gradual set/reset and its effectiveness for deep learning. In *IEEE International Electron Devices Meeting (IEDM)*.
- [96] Hsueh, F. K., Shen, C. H., Shieh, J. M., Li, K. S., Chen, H. C., Huang, W. H., Wang, H. H., Yang, C. C., Hsieh, T. Y., Lin, C. H., Chen, B. Y., Shiao, Y. S., Huang, G. W., Wong, O. Y.,

- Chen, P. H., & Yeh, W. K. (2016). First fully functionalized monolithic 3d iot chip with 0.5 v light-electricity power management, 6.8 ghz wireless-communication vco, and 4-layer vertical reram. In *IEEE International Electron Devices Meeting (IEDM)*.
- [97] Hu, G., Gottwald, M. G., He, Q., Park, J. H., Lauer, G., Nowak, J. J., Brown, S. L., Doris, B., Edelstein, D., Evarts, E. R., Hashemi, P., Khan, B., Kim, Y. H., Kothandaraman, C., Marchack, N., O'Sullivan, E. J., Reuter, M., Robertazzi, R. P., Sun, J. Z., Suwannasiri, T., Trouilloud, P. L., Zhu, Y., & Worledge, D. C. (2017). Key parameters affecting stt-mram switching efficiency and improved device performance of 400c-compatible p-mtjs. In *IEEE International Electron Devices Meeting (IEDM)*.
- [98] Hu, G., Nowak, J. J., Gottwald, M. G., Brown, S. L., Doris, B., D'Emic, C. P., Hashemi, P., Houssameddine, D., He, Q., Kim, D., Kim, J., Kothandaraman, C., Lauer, G., Lee, H. K., Marchack, N., Reuter, M., Robertazzi, R. P., Sun, J. Z., Suwannasiri, T., Trouilloud, P. L., Woo, S., & Worledge, D. C. (2019). Spin-transfer torque mram with reliable 2 ns writing for last level cache applications. In *IEEE International Electron Devices Meeting (IEDM)*.
- [99] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 29*.
- [100] IEEE (2020). International roadmap for devices and systems (IRDS) 2020 edition.
- [101] Inci, A., Isgenc, M. M., & Marculescu, D. (2020a). DeepNVM++: cross-layer modeling and optimization framework of non-volatile memories for deep learning. *preprint arXiv:2012.04559*.
- [102] Inci, A. F., Meric Isgenc, M., & Marculescu, D. (2020b). Deepnvm: A framework for modeling and analysis of non-volatile memory technologies for deep learning applications. In *Design, Automation Test in Europe Conference Exhibition (DATE)*.
- [103] Intel Corporation (2018). Intel vtune. <https://software.intel.com/en-us/vtune>.
- [104] Jain, P., Arslan, U., Sekhar, M., Lin, B. C., Wei, L., Sahu, T., Alzate-vinasco, J., Vangapaty, A., Meterelliyoz, M., Strutt, N., Chen, A. B., Hentges, P., Quintero, P. A., Connor, C., Golonzka, O., Fischer, K., & Hamzaoglu, F. (2019). 13.2 a 3.6mb 10.1mb/mm<sup>2</sup> embedded non-volatile reram macro in 22nm finfet technology with adaptive forming/set/reset schemes yielding down to 0.5v with sensing time of 5ns at 0.7v. In *IEEE International Solid-State Circuits Conference - (ISSCC)*.
- [105] Jain, S., Ranjan, A., Roy, K., & Raghunathan, A. (2018). Computing in memory with spin-transfer torque magnetic ram. *IEEE Trans. Very Large Scale Integr. Syst.*

- [106] Jan, G., Thomas, L., Le, S., Lee, Y., Liu, H., Zhu, J., Iwata-Harms, J., Patel, S., Tong, R., Serrano-Guisan, S., Shen, D., He, R., Haq, J., Teng, J., Lam, V., Annapragada, R., Wang, Y., Zhong, T., Torng, T., & Wang, P. (2016). Achieving sub-ns switching of stt-mram for future embedded llc applications through improvement of nucleation and propagation switching mechanisms. In *IEEE Symposium on VLSI Technology*.
- [107] Jerry, M., Chen, P., Zhang, J., Sharma, P., Ni, K., Yu, S., & Datta, S. (2017). Ferroelectric fet analog synapse for acceleration of deep neural network training. In *IEEE International Electron Devices Meeting (IEDM)*.
- [108] Jerry, M., Dutta, S., Kazemi, A., Ni, K., Zhang, J., Chen, P.-Y., Sharma, P., Yu, S., Hu, X. S., Niemier, M., & Datta, S. (2018). A ferroelectric field effect transistor based synaptic weight cell. *Journal of Physics D: Applied Physics*.
- [109] Jiang, Z., Wang, Z., Zheng, X., Fong, S., Qin, S., Chen, H. Y., Ahn, C., Cao, J., Nishi, Y., & Wong, H. P. (2016). Microsecond transient thermal behavior of hfox-based resistive random access memory using a micro thermal stage (mts). In *IEEE International Electron Devices Meeting (IEDM)*.
- [110] Jin, Y., Shihab, M., & Jung, M. (2014). Area, power, and latency considerations of stt-mram to substitute for main memory. In *Proc. ISCA*.
- [111] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P.-l., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Ghaemmaghami, T. V., Gottipati, R., Gulland, W., Hagmann, R., Ho, C. R., Hogberg, D., Hu, J., Hundt, R., Hurt, D., Ibarz, J., Jaffey, A., Jaworski, A., Kaplan, A., Khaitan, H., Killebrew, D., Koch, A., Kumar, N., Lacy, S., Laudon, J., Law, J., Le, D., Leary, C., Liu, Z., Lucke, K., Lundin, A., MacKean, G., Maggiore, A., Mahony, M., Miller, K., Nagarajan, R., Narayanaswami, R., Ni, R., Nix, K., Norrie, T., Omernick, M., Penukonda, N., Phelps, A., Ross, J., Ross, M., Salek, A., Samadiani, E., Severn, C., Sizikov, G., Snelham, M., Souter, J., Steinberg, D., Swing, A., Tan, M., Thorson, G., Tian, B., Toma, H., Tuttle, E., Vasudevan, V., Walter, R., Wang, W., Wilcox, E., & Yoon, D. H. (2017). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture, ISCA '17*.
- [112] Kaiser, L., Gomez, A. N., Shazeer, N., Vaswani, A., Parmar, N., Jones, L., & Uszkoreit, J. (2017). One model to learn them all. *CoRR*, abs/1706.05137.
- [113] Kan, J. J., Park, C., Ching, C., Ahn, J., Xue, L., Wang, R., Kontos, A., Liang, S., Bangar, M., Chen, H., Hassan, S., Kim, S., Pakala, M., & Kang, S. H. (2016). Systematic validation of 2x nm diameter perpendicular mtj arrays and mgo barrier for sub-10 nm embedded stt-mram with practically unlimited endurance. In *IEEE International Electron Devices Meeting (IEDM)*.

- [114] Kanev, S., Darago, J. P., Hazelwood, K., Ranganathan, P., Moseley, T., Wei, G.-Y., & Brooks, D. (2015). Profiling a warehouse-scale computer. In *International Symposium on Computer Architecture (ISCA)* (pp. 158–169).
- [115] Khan, F., Cartier, E., Kothandaraman, C., Scott, J. C., Woo, J. C. S., & Iyer, S. S. (2016). The impact of self-heating on charge trapping in high- $k$ -metal-gate nfts. *IEEE Electron Device Letters*, 37(1), 88–91.
- [116] Khan, F., Cartier, E., Woo, J. C. S., & Iyer, S. S. (2017). Charge trap transistor (ctt): An embedded fully logic-compatible multiple-time programmable non-volatile memory for high- $k$ -metal-gate cmos technologies. *IEEE Electron Device Letters*.
- [117] Khan, F., Moy, D., Anand, D., Schroeder, E. H., Katz, R., Jiang, L., Banghart, E., Robson, N., & Kirihata, T. (2019). Turning logic transistors into secure, multi-time programmable, embedded non-volatile memory elements for 14 nm finfet technologies and beyond. In *Symposium on VLSI Technology*.
- [118] Khwa, W., Chang, M., Wu, J., Lee, M., Su, T., Yang, K., Chen, T., Wang, T., Li, H., Brightsky, M., Kim, S., Lung, H., & Lam, C. (2017). A resistance drift compensation scheme to reduce mlc pcm raw ber by over 100 $\times$  for storage class memory applications. *IEEE Journal of Solid-State Circuits*.
- [119] Khwa, W. S., Chang, M. F., Wu, J. Y., Lee, M. H., Su, T. H., Yang, K. H., Chen, T. F., Wang, T. Y., Li, H. P., BrightSky, M., Kim, S., Lung, H. L., & Lam, C. (2016). 7.3 a resistance-drift compensation scheme to reduce mlc pcm raw ber by over 100 $\times$  for storage-class memory applications. In *IEEE International Solid-State Circuits Conference (ISSCC)*.
- [120] Kim, S. G., Lee, J. C., Ha, T. J., Lee, J. H., Lee, J. Y., Park, Y. T., Kim, K. W., Ju, W. K., Ko, Y. S., Hwang, H. M., Lee, B. M., Moon, J. Y., Park, W. Y., Gyun, B. G., Lee, B., Yim, D., & Hong, S. (2017). Breakthrough of selector technology for cross-point 25-nm reram. In *IEEE International Electron Devices Meeting (IEDM)*.
- [121] Kim, W., Hardtdegen, A., Rodenbücher, C., Menzel, S., Wouters, D. J., Hoffmann-Eifert, S., Buca, D., Waser, R., & Rana, V. (2016). Forming-free metal-oxide reram by oxygen ion implantation process. In *IEEE International Electron Devices Meeting (IEDM)*.
- [122] Kobayashi, M., Ueyama, N., Jang, K., & Hiramoto, T. (2016). Experimental study on polarization-limited operation speed of negative capacitance fet with ferroelectric hfo<sub>2</sub>. In *IEEE International Electron Devices Meeting (IEDM)*.
- [123] Korgaonkar, K., Bhati, I., Liu, H., Gaur, J., Manipatruni, S., Subramoney, S., Karnik, T., Swanson, S., Young, I., & Wang, H. (2018). Density tradeoffs of non-volatile memory as a replacement for sram based last level cache. In *ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*.

- [124] Krivokapic, Z., Rana, U., Galatage, R., Razavieh, A., Aziz, A., Liu, J., Shi, J., Kim, H. J., Sporer, R., Serrao, C., Busquet, A., Polakowski, P., Müller, J., Kleemeier, W., Jacob, A., Brown, D., Knorr, A., Carter, R., & Banna, S. (2017). 14nm ferroelectric finfet technology with steep subthreshold slope for ultra low power applications. In *IEEE International Electron Devices Meeting (IEDM)*.
- [125] Krizhevsky, A. (2016). Learning Multiple Layers of Features from Tiny Images. Technical Report.
- [126] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942.
- [127] Lane, N. D., Bhattacharya, S., Georgiev, P., Forlivesi, C., & Kawsar, F. (2015). An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices. In *Proceedings of the 2015 International Workshop on Internet of Things Towards Applications, IoT-App '15*.
- [128] Lee, C., Lin, H., Lien, C., Chih, Y., & Chang, J. (2017). A 1.4mb 40-nm embedded rram macro with 0.07um<sup>2</sup> bit cell, 2.7ma/100mhz low-power read and hybrid write verify for high endurance application. In *2017 IEEE Asian Solid-State Circuits Conference (A-SSCC)*.
- [129] Lee, K., Bak, J. H., Kim, Y. J., Kim, C. K., Antonyan, A., Chang, D. H., Hwang, S. H., Lee, G. W., Ji, N. Y., Kim, W. J., Lee, J. H., Bae, B. J., Park, J. H., Kim, I. H., Seo, B. Y., Han, S. H., Ji, Y., Jung, H. T., Park, S. O., Kwon, O. I., Kye, J. W., Kim, Y. D., Pae, S. W., Song, Y. J., Jeong, G. T., Hwang, K. H., Koh, G. H., Kang, H. K., & Jung, E. S. (2019). 1gbit high density embedded stt-mram in 28nm fdsoi technology. In *IEEE International Electron Devices Meeting (IEDM)*.
- [130] Lee, K., Chao, R., Yamane, K., Naik, V. B., Yang, H., Kwon, J., Chung, N. L., Jang, S. H., Behin-Aein, B., Lim, J. H., K, S., Liu, B., Toh, E. H., Gan, K. W., Zeng, D., Thiyagarajah, N., Goh, L. C., Ling, T., Ting, J. W., Hwang, J., Zhang, L., Low, R., Krishnan, R., Zhang, L., Tan, S. L., You, Y. S., Seet, C. S., Cong, H., Wong, J., Woo, S. T., Quek, E., & Siah, S. Y. (2018). 22-nm fd-soi embedded mram technology for low-power automotive-grade-l mcu applications. In *IEEE International Electron Devices Meeting (IEDM)*.
- [131] Lee, M. H., Chen, P., Fan, S., Chou, Y., C.Kuo, Tang, C., Chen, H., Gu, S., Hong, R., Wang, Z., Chen, S., Liao, C., Chen, K., Chang, S. T., Liao, M., Li, K., & Liu, C. W. (2017). Ferroelectric al:hfo<sub>2</sub> negative capacitance fets. In *IEEE International Electron Devices Meeting (IEDM)*.
- [132] Lee, M. H., Fan, S., Tang, C., Chen, P., Chou, Y., Chen, H., Kuo, J., Xie, M., Liu, S., Liao, M., Jong, C., Li, K., Chen, M., & Liu, C. W. (2016). Physical thickness 1.x nm ferroelectric hfzrox negative capacitance fets. In *IEEE International Electron Devices Meeting (IEDM)*.

- [133] Lee, T., Yamane, K., Kwon, J., Naik, V., Otani, Y., Zeng, D., Lim, J., Sivabalan, K., Chiang, C., Huang, Y., Jang, S., Hau, L., Chao, R., Chung, N., Neo, W., Khua, K., Thiyagarajah, N., Ling, T., Goh, L., & Siah, S. (2020). Fast switching of stt-mram to realize high speed applications. In *IEEE Symposium on VLSI Technology*.
- [134] Lee, Y., Song, Y., Kim, J., Oh, S., Bae, B.-J., Lee, S., Lee, J., Pi, U., Seo, B., Jung, H., Lee, K., Shin, H., Jung, H., Pyo, M., Antonyan, A., Lee, D., Hwang, S., Jang, D., Ji, Y., & Jung, E. (2018). Embedded stt-mram in 28-nm fdsoi logic process for industrial mcu/iot application. In *Symposium on VLSI Technology*.
- [135] Leskovec, J. & Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection.
- [136] Leskovec, J. & Sosič, R. (2016). Snap: A general-purpose network analysis and graph-mining library. *ACM TIST*.
- [137] Li, G., Hari, S., Sullivan, M., Tsai, T., Pattabiraman, K., Emer, J., & Keckler, S. W. (2017a). Understanding error propagation in deep learning neural network (dnn) accelerators and applications. In *SC*.
- [138] Li, H., Li, K. S., Lin, C. H., Hsu, J. L., Chiu, W. C., Chen, M. C., Wu, T. T., Sohn, J., Eryilmaz, S. B., Shieh, J. M., Yeh, W. K., & Wong, H. P. (2016a). Four-layer 3d vertical rram integrated with finfet as a versatile computing unit for brain-inspired cognitive information processing. In *IEEE Symposium on VLSI Technology*.
- [139] Li, H., Wu, T. F., Rahimi, A., Li, K. S., Rusch, M., Lin, C. H., Hsu, J. L., Sabry, M. M., Eryilmaz, S. B., Sohn, J., Chiu, W. C., Chen, M. C., Wu, T. T., Shieh, J. M., Yeh, W. K., Rabaey, J. M., Mitra, S., & Wong, H. P. (2016b). Hyperdimensional computing with 3d vrram in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition. In *IEEE International Electron Devices Meeting (IEDM)*.
- [140] Li, S., Niu, D., Malladi, K. T., Zheng, H., Brennan, B., & Xie, Y. (2017b). Drisa: A dram-based reconfigurable in-situ accelerator. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-50 '17*.
- [141] Li, Z., Chen, P. Y., Xu, H., & Yu, S. (2017c). Design of ternary neural network with 3-d vertical rram array. *IEEE Transactions on Electron Devices*.
- [142] Liao, Y., Pan, C., & Naeemi, A. (2020). Benchmarking and optimization of spintronic memory arrays. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*.
- [143] Lin, C., Hung, J., Lin, W., Lo, C., Chiang, Y., Tsai, H., Yang, G., King, Y., Lin, C. J., Chen, T., & Chang, M. (2016). A 256b-word length rram-based tcam with 1ns search-time and 14x improvement in word length-energy efficiency-density product using 2.5t1r cell. In *IEEE International Solid-State Circuits Conference (ISSCC)*.

- [144] Lin, C. S., Huang, W. T., Huang, A., Yang, Y. H., Hsu, Y. L., Yang, S. K., Liu, J., Tseng, H. W., Huang, J., Chou, B. Y., Huang, K., Chang, W. K., Chang, D., Chien, C. H., Yeh, H., Liu, P. W., Hsieh, C. D., Chuang, H., & Kalnitsky, A. (2020). An approach to embedding traditional non-volatile memories into a deep sub-micron cmos. In *IEEE Symposium on VLSI Technology*.
- [145] Lin, Y. H., Ho, Y. H., Lee, M. H., Wang, C. H., Lin, Y. Y., Lee, F. M., Hsu, K. C., Tseng, P. H., Lee, D. Y., Chiang, K. H., Wang, K. C., Tseng, T. Y., & Lu, C. Y. (2017). A comprehensive study of 3-stage high resistance state retention behavior for tmo rerams from single cells to a large array. In *IEEE International Electron Devices Meeting (IEDM)*.
- [146] Liu, J., Hsu, C., Wang, I., & Hou, T. (2015). Categorization of multilevel-cell storage-class memory: An rram example. *IEEE Transactions on Electron Devices*.
- [147] Liu, Q., Gao, B., Yao, P., Wu, D., Chen, J., Pang, Y., Zhang, W., Liao, Y., Xue, C., Chen, W., Tang, J., Wang, Y., Chang, M., Qian, H., & Wu, H. (2020). A fully integrated analog rram based 78.4tops/w compute-in-memory chip with fully parallel mac computing. In *IEEE International Solid-State Circuits Conference - (ISSCC)*.
- [148] Liu, T., Yan, T. H., Scheuerlein, R., Chen, Y., Lee, J. K., Balakrishnan, G., Yee, G., Zhang, H., Yap, A., Ouyang, J., Sasaki, T., Addepalli, S., Al-Shamma, A., Chen, C., Gupta, M., Hilton, G., Joshi, S., Kathuria, A., Lai, V., Masiwal, D., Matsumoto, M., Nigam, A., Pai, A., Pakhale, J., Siau, C. H., Wu, X., Yin, R., Peng, L., Kang, J. Y., Huynh, S., Wang, H., Nagel, N., Tanaka, Y., Higashitani, M., Minvielle, T., Gorla, C., Tsukamoto, T., Yamaguchi, T., Okajima, M., Okamura, T., Takase, S., Hara, T., Inoue, H., Fasoli, L., Mofidi, M., Shrivastava, R., & Quader, K. (2013). A 130.7mm<sup>2</sup> 2-layer 32gb rram memory device in 24nm technology. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*.
- [149] Liu, Y., Wang, Z., Lee, A., Su, F., Lo, C., Yuan, Z., Lin, C., Wei, Q., Wang, Y., King, Y., Lin, C., Khalili, P., Wang, K., Chang, M., & Yang, H. (2016). A 65nm rram-enabled nonvolatile processor with 6× reduction in restore time and 4× higher clock frequency using adaptive data retention and self-write-termination nonvolatile logic. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*.
- [150] Liu, Y., Wang, Z., Lee, A., Su, F., Lo, C. P., Yuan, Z., Lin, C. C., Wei, Q., Wang, Y., King, Y. C., Lin, C. J., Khalili, P., Wang, K. L., Chang, M. F., & Yang, H. (2016). A 65nm rram-enabled nonvolatile processor with 6x reduction in restore time and 4x higher clock frequency using adaptive data retention and self-write-termination nonvolatile logic. In *IEEE International Solid-State Circuits Conference (ISSCC)*.
- [151] Lo, C. P., Chen, W. H., Wang, Z., Lee, A., Hsu, K. H., Su, F., King, Y. C., Lin, C. J., Liu, Y., Yang, H., Khalili, P., Wang, K. L., & Chang, M. F. (2016). A rram-based single-nvm

- nonvolatile flip-flop with reduced stress-time and write-power against wide distribution in write-time by using self-write-termination scheme for nonvolatile processors in iot era. In *IEEE International Electron Devices Meeting (IEDM)*.
- [152] Lung, H. L., Ho, Y. H., Zhu, Y., Chien, W. C., Kim, S., Kim, W., Cheng, H. Y., Ray, A., Brightsky, M., Bruce, R., Yeh, C. W., & Lam, C. (2016). A novel low power phase change memory using inter-granular switching. In *IEEE Symposium on VLSI Technology*.
- [153] Luo, Q., Xu, X., Gong, T., Lv, H., Dong, D., Ma, H., Yuan, P., Gao, J., Liu, J., Yu, Z., Li, J., Long, S., Liu, Q., & Liu, M. (2017). 8-layers 3d vertical rram with excellent scalability towards storage class memory applications. In *IEEE International Electron Devices Meeting (IEDM)*.
- [154] Lv, H., Xu, X., Yuan, P., Dong, D., Gong, T., Liu, J., Yu, Z., Huang, P., Zhang, K., Huo, C., Chen, C., Xie, Y., Luo, Q., Long, S., Liu, Q., Kang, J., Yang, D., Yin, S., Chiu, S., & Liu, M. (2017). Beol based rram with one extra-mask for low cost, highly reliable embedded application in 28 nm node and beyond. In *IEEE International Electron Devices Meeting (IEDM)*.
- [155] Lyons, M. J., Hempstead, M., Wei, G.-Y., & Brooks, D. (2012). The accelerator store: A shared memory framework for accelerator-based systems. *ACM Trans. Archit. Code Optim.*, 8(4).
- [156] Ma, J., Chai, Z., Zhang, W., Govoreanu, B., Zhang, J. F., Ji, Z., Benbakhti, B., Groeseneken, G., & Jurczak, M. (2016). Identify the critical regions and switching/failure mechanisms in non-filamentary rram (a-vmco) by rtn and cvs techniques for memory window improvement. In *IEEE International Electron Devices Meeting (IEDM)*.
- [157] Ma, S., Donato, M., Lee, S. K., Brooks, D., & Wei, G.-Y. (2019). Fully-cmos multi-level embedded non-volatile memory devices with reliable long-term retention for efficient storage of neural network weights. *IEEE Electron Device Letters*, 40(9), 1403–1406.
- [158] Macri, J. (2015). Amd's next generation gpu and high bandwidth memory architecture: Fury. In *2015 IEEE Hot Chips 27 Symposium (HCS)* (pp. 1–26).
- [159] Mattson, P., Cheng, C., Diamos, G., Coleman, C., Micikevicius, P., Patterson, D., Tang, H., Wei, G.-Y., Bailis, P., Bittorf, V., Brooks, D., Chen, D., Dutta, D., Gupta, U., Hazelwood, K., Hock, A., Huang, X., Kang, D., Kanter, D., Kumar, N., Liao, J., Narayanan, D., Oguntebi, T., Pekhimenko, G., Pentecost, L., Janapa Reddi, V., Robie, T., St John, T., Wu, C.-J., Xu, L., Young, C., & Zaharia, M. (2020). Mlperf training benchmark. In *Proceedings of Machine Learning and Systems*, volume 2 (pp. 336–349).
- [160] Microsoft (2019). Understanding frames per second.



- [161] Mikolajick, T., Schroeder, U., & Slesazeck, S. (2018). Hafnium oxide based ferroelectric devices for memories and beyond. In *VLSI-TSA* (pp. 1–2).
- [162] Milo, V., Pedretti, G., Carboni, R., Calderoni, A., Ramaswamy, N., Ambrogio, S., & Ielmini, D. (2016). Demonstration of hybrid cmos/rram neural networks with spike time/rate-dependent plasticity. In *IEEE International Electron Devices Meeting (IEDM)*.
- [163] Min, D., Byun, I., Lee, G.-H., Na, S., & Kim, J. (2020). *CryoCache: A Fast, Large, and Cost-Effective Cache Architecture for Cryogenic Computing*, (pp. 449–464). Association for Computing Machinery: New York, NY, USA.
- [164] Miura, S., Nishioka, K., Naganuma, H., Nguyen, T. V. A., Honjo, H., Ikeda, S., Watanabe, T., Inoue, H., Niwa, M., Tanigawa, T., Noguchi, Y., Yoshiduka, T., Yasuhira, M., & Endoh, T. (2020). Scalability of quad interface p-mtj for 1x nm stt-mram with 10 ns low power write operation, 10 years retention and endurance 1011. In *IEEE Symposium on VLSI Technology*.
- [165] Miyaji, K., Shinozuka, Y., & Takeuchi, K. (2012). Zero Additional Process, Local Charge Trap, Embedded Flash Memory with Drain-Side Assisted Erase Scheme Using Minimum Channel Length Width Standard Complementary Metal-Oxide-Semiconductor Single Transistor Cell. *Jpn. J. Appl. Phys.*, 51.
- [166] Mo, F., Tagawa, Y., Jin, C., Ahn, M., Saraya, T., Hiramoto, T., & Kobayashi, M. (2019). Experimental demonstration of ferroelectric hfo<sub>2</sub> fet with ultrathin-body igzo for high-density and low-power memory application. In *Symposium on VLSI Technology*.
- [167] Mukherjee, S., Weaver, C., Emer, J., Reinhardt, S., & Austin, T. (2003). A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36*. (pp. 29–40).
- [168] Mulaosmanovic, H., Ocker, J., Müller, S., Noack, M., Müller, J., Polakowski, P., Mikolajick, T., & Slesazeck, S. (2017). Novel ferroelectric fet based synapse for neuromorphic systems. In *Symposium on VLSI Technology*.
- [169] Naik, V. B., Lim, J. H., Yamane, K., Zeng, D., Yang, H., Thiyagarajah, N., Kwon, J., Chung, N. L., Chao, R., Ling, T., & Lee, K. (2019). Superior endurance performance of 22-nm embedded mram technology. In *IEEE International Reliability Physics Symposium (IRPS)*.
- [170] Naik, V. B., Yamane, K., Lim, J. H., Lee, T. Y., Kwon, J., Aein, B., Chung, N. L., Hau, L. Y., Chao, R., Zeng, D., Otani, Y., Chiang, C., Huang, Y., Pu, L., Thiyagarajah, N., Jang, S. H., Neo, W. P., Dixit, H., Aris, S. K., Goh, L. C., Ling, T., Hwang, J., Ting, J. W., Zhang, L., Low, R., Balasankaran, N., Seet, C. S., Ong, S., Wong, J., You, Y. S., Woo, S. T., & Siah, S. Y. (2020). A reliable tddb lifetime projection model verified using 40mb stt-mram macro at sub-ppm failure rate to realize unlimited endurance for cache applications. In *IEEE Symposium on VLSI Technology*.

- [171] Nail, C., Molas, G., Blaise, P., Piccolboni, G., Sklenard, B., Cagli, C., Bernard, M., Roule, A., Azzaz, M., Vianello, E., Carabasse, C., Berthier, R., Cooper, D., Pelissier, C., Magis, T., Ghibaudo, G., Vallée, C., Bedeau, D., Mosendz, O., De Salvo, B., & Perniola, L. (2016). Understanding rram endurance, retention and window margin trade-off using experimental results and simulations. In *IEEE International Electron Devices Meeting (IEDM)*.
- [172] Natsui, M., Tamakoshi, A., Honjo, H., Watanabe, T., Nasuno, T., Zhang, C., Tanigawa, T., Inoue, H., Niwa, M., Yoshiduka, T., Noguchi, Y., Yasuhira, M., Ma, Y., Shen, H., Fukami, S., Sato, H., Ikeda, S., Ohno, H., Endoh, T., & Hanyu, T. (2020). Dual-port field-free sot-mram achieving 90-mhz read and 60-mhz write operations under 55-nm cmos technology and 1.2-v supply voltage. In *IEEE Symposium on VLSI Circuits*.
- [173] Nguyen, V. D., Sabon, P., Chatterjee, J., Tille, L., Coelho, P. V., Auffret, S., Sousa, R., Prejbeanu, L., Gautier, E., Vila, L., & Dieny, B. (2017). Novel approach for nano-patterning magnetic tunnel junctions stacks at narrow pitch: A route towards high density stt-mram applications. In *IEEE International Electron Devices Meeting (IEDM)*.
- [174] Ni, K., Chakraborty, W., Smith, J., Grisafe, B., & Datta, S. (2019). Fundamental understanding and control of device-to-device variation in deeply scaled ferroelectric fets. In *Symposium on VLSI Technology*.
- [175] Ni, K., Li, X., Smith, J. A., Jerry, M., & Datta, S. (2018). Write disturb in ferroelectric fets and its implication for 1t-1f1t and memory arrays. *IEEE Electron Device Letters*.
- [176] Noguchi, H., Ikegami, K., Abe, K., Fujita, S., Shiota, Y., Nozaki, T., Yuasa, S., & Suzuki, Y. (2016a). Novel voltage controlled mram (vcm) with fast read/write circuits for ultra large last level cache. In *IEEE International Electron Devices Meeting (IEDM)*.
- [177] Noguchi, H., Ikegami, K., Takaya, S., Arima, E., Kushida, K., Kawasumi, A., Hara, H., Abe, K., Shimomura, N., Ito, J., Fujita, S., Nakada, T., & Nakamura, H. (2016b). 4mb stt-mram-based cache with memory-access-aware power optimization and write-verify-write read-modify-write scheme. In *IEEE International Solid-State Circuits Conference (ISSCC)*.
- [178] NVIDIA (2017). Nvidia deep learning accelerator (nvdl): a free and open architecture that promotes a standard way to design deep learning inference accelerators. [nvidia.org](http://nvidia.org).
- [179] Okuno, J., Kunihiro, T., Konishi, K., Maemura, H., Shuto, Y., Sugaya, F., Materano, M., Ali, T., Kuehnel, K., Seidel, K., Schroeder, U., Mikolajick, T., Tsukamoto, M., & Umebayashi, T. (2020). Soc compatible 1t1c feram memory array based on ferroelectric hfo<sub>5</sub>zro<sub>5</sub>o<sub>2</sub>. In *IEEE Symposium on VLSI Technology*.
- [180] Pang, Y., Wu, H., Gao, B., Wu, D., Chen, A., & Qian, H. (2017). A novel puf against machine learning attack: Implementation on a 16 mb rram chip. In *IEEE International Electron Devices Meeting (IEDM)*.

- [181] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*.
- [182] Pellauer, M., Shao, Y. S., Clemons, J., Crago, N., Hegde, K., Venkatesan, R., Keckler, S. W., Fletcher, C. W., & Emer, J. (2019). Buffets: An efficient and composable storage idiom for explicit decoupled data orchestration. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '19* (pp. 137–151). New York, NY, USA: Association for Computing Machinery.
- [183] Pentecost, L., Donato, M., Reagen, B., Gupta, U., Ma, S., Wei, G.-Y., & Brooks, D. (2019a). Maxnmv: Maximizing dnn storage density and inference efficiency with sparse encoding and error mitigation. In *Proceedings of the 52nd International Symposium on Microarchitecture*.
- [184] Pentecost, L., Gupta, U., Ngan, E., Beyer, J., Wei, G.-Y., Brooks, D., & Behrisch, M. (2019b). Champvis: Comparative hierarchical analysis of microarchitectural performance. In *2019 IEEE/ACM International Workshop on Programming and Performance Visualization Tools (ProTools) at Super Computing (SC19)* (pp. 55–61).
- [185] Pentecost, L., Hankin, A., Donato, M., Hempstead, M., Wei, G.-Y., & Brooks, D. (2022). Nvmexplorer: A framework for cross-stack comparisons of embedded non-volatile memory solutions. In *The 28th IEEE International Symposium on High-Performance Computer Architecture (HPCA-28)*.
- [186] Plancher, B., Brumar, C. D., Brumar, I., Pentecost, L., Rama, S., & Brooks, D. (2019). Application of approximate matrix multiplication to neural networks and distributed slam. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)* (pp. 1–7).
- [187] Poremba, M., Mittal, S., Li, D., Vetter, J., & Xie, Y. (2015). Destiny: A tool for modeling emerging 3d nvm and edram caches. In *Design, Automation Test in Europe Conference Exhibition (DATE)*.
- [188] Poremba, M. & Xie, Y. (2012). Nvmain: An architectural-level main memory simulator for emerging non-volatile memories. In *IEEE Computer Society Annual Symposium on VLSI*.
- [189] Prakash, A., Park, J., Song, J., Woo, J., Cha, E., & Hwang, H. (2015). Demonstration of low power 3-bit multilevel cell characteristics in a taox-based rram by stack engineering. *IEEE Electron Device Letters*.
- [190] Puglisi, F. M., Larcher, L., Pan, C., Xiao, N., Shi, Y., Hui, F., & Lanza, M. (2016). 2d hbn based rram devices. In *IEEE International Electron Devices Meeting (IEDM)*.

- [191] Pyne, E., Pentecost, L., Gupta, U., Wei, G.-Y., & Brooks, D. (2020). Quantifying the impact of data encoding on dnn fault tolerance. In *International Workshop on Performance Analysis of Machine Learning Systems (FastPath ISPASS 2020)*.
- [192] Reagen, B., Adolf, R., Whatmough, P., Wei, G.-Y., & Brooks, D. (2017). Deep Learning for Computer Architects. *Synth. Lect. Comput. Archit.*, 12(4), 1–123.
- [193] Reagen, B., Gupta, U., Pentecost, L., Whatmough, P., Lee, S. K., Mulholland, N., Brooks, D., & Wei, G. (2018). Ares: A framework for quantifying the resilience of deep neural networks. In *55th Design Automation Conference (DAC)*.
- [194] Reagen, B., Whatmough, P., Adolf, R., Rama, S., Lee, H., Lee, S. K., Hernandez-Lobato, J. M., Wei, G.-Y., & Brooks, D. (2016). Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *ISCA*.
- [195] Rebuffi, S.-A., Bilen, H., & Vedaldi, A. (2017). Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*.
- [196] Reis, D., Ni, K., Chakraborty, W., Yin, X., Trentzsch, M., Dünkel, S. D., Melde, T., Müller, J., Beyer, S., Datta, S., Niemier, M. T., & Hu, X. S. (2019). Design and analysis of an ultra-dense, low-leakage, and fast fefet-based random access memory array. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*.
- [197] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*.
- [198] Saida, D., Kashiwada, S., Yakabe, M., Daibou, T., Hase, N., Fukumoto, M., Miwa, S., Suzuki, Y., Noguchi, H., Fujita, S., & Ito, J. (2016). Sub-3 ns pulse with sub-100 microa switching of 1x–2x nm perpendicular mtj for high-performance embedded stt-mram towards sub-20 nm cmos. In *IEEE Symposium on VLSI Technology*.
- [199] Sato, H., Honjo, H., Watanabe, T., Niwa, M., Koike, H., Miura, S., Saito, T., Inoue, H., Nasuno, T., Tanigawa, T., Noguchi, Y., Yoshiduka, T., Yasuhira, M., Ikeda, S., Kang, S. Y., Kubo, T., Yamashita, K., Yagi, Y., Tamura, R., & Endoh, T. (2018). 14ns write speed 128mb density embedded stt-mram with endurance 1010 and 10yrs retention 85c using novel low damage mtj integration process. In *IEEE International Electron Devices Meeting (IEDM)*.
- [200] Semiconductor, C. (2017). What types of ecc should be used on flash memory?
- [201] Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J. P., Hu, M., Williams, R. S., & Srikumar, V. (2016). Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*.

- [202] Shao, Y. S., Clemons, J., Venkatesan, R., Zimmer, B., Fojtik, M., Jiang, N., Keller, B., Klinefelter, A., Pinckney, N., Raina, P., Tell, S. G., Zhang, Y., Dally, W. J., Emer, J., Gray, C. T., Khailany, B., & Keckler, S. W. (2019). Simba: Scaling deep-learning inference with multi-chip-module-based architecture. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO '52* New York, NY, USA: Association for Computing Machinery.
- [203] Shao, Y. S., Reagen, B., Wei, G.-Y., & Brooks, D. (2015). The aladdin approach to accelerator design and modeling. *IEEE Micro*, 35(3), 58–70.
- [204] Sharifi, M. M., Pentecost, L., Rajaei, R., Kazemi, A., Lou, Q., Wei, G.-Y., Brooks, D., Ni, K., Hu, X. S., Niemier, M., & Donato, M. (2021). Application-driven design exploration for dense ferroelectric embedded non-volatile memories. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '21)*.
- [205] Sharma, A. A., Doyle, B., Yoo, H. J., Tung, I. C., Kavalieros, J., Metz, M. V., Reshotko, M., Majhi, P., Brown-Heft, T., Chen, Y. J., & Le, V. H. (2020). High speed memory operation in channel-last, back-gated ferroelectric transistors. In *IEEE International Electron Devices Meeting (IEDM)*.
- [206] Shibayama, S., Xu, L., Migita, S., & Toriumi, A. (2016). Study of wake-up and fatigue properties in doped and undoped ferroelectric hfo<sub>2</sub> in conjunction with piezo-response force microscopy analysis. In *IEEE Symposium on VLSI Technology*.
- [207] Shih, M., Wang, C., Lee, Y., Wang, W., Thomas, L., Liu, H., Zhu, J., Lee, Y., Jan, G., Wang, Y., Zhong, T., Torng, T., Wang, P., Lin, D., Chiang, T., Shen, K., Chuang, H., & Gallagher, W. J. (2016). Reliability study of perpendicular stt-mram as emerging embedded memory qualified for reflow soldering at 260c. In *IEEE Symposium on VLSI Technology*.
- [208] Shih, Y., Lee, C., Chang, Y., Lee, P., Lin, H., Chen, Y., Lin, K., Yeh, T., Yu, H., Chuang, H., Chih, Y., & Chang, J. (2018). Logic process compatible 40nm 16mb, embedded perpendicular-mram with hybrid-resistance reference, sub-microa sensing resolution, and 17.5ns read access time. In *IEEE Symposium on VLSI Circuits*.
- [209] Shim, W., Seo, J.-s., & Yu, S. (2020). Two-step write-verify scheme and impact of the read noise in multilevel rram-based inference engine. *Semi. Sci. Tech.*
- [210] Shin, H., Kim, J., Kang, S., & Kwak, S. (2020). A 28nm 10mb embedded flash memory for iot product with ultra-low power near-1v supply voltage and high temperature for grade 1 operation. In *IEEE Symposium on VLSI Circuits*.
- [211] Shum, D., Houssameddine, D., Woo, S. T., You, Y. S., Wong, J., Wong, K. W., Wang, C. C., Lee, K. H., Yamane, K., Naik, V. B., Seet, C. S., Tahmasebi, T., Hai, C., Yang, H. W., Thiagarajah, N., Chao, R., Ting, J. W., Chung, N. L., Ling, T., Chan, T. H., Siah, S. Y., Nair, R.,

- Deshpande, S., Whig, R., Nagel, K., Aggarwal, S., DeHerrera, M., Janesky, J., Lin, M., Chia, H., Hossain, M., Lu, H., Ikegawa, S., Mancoff, F. B., Shimon, G., Slaughter, J. M., Sun, J. J., Tran, M., Alam, S. M., & Andre, T. (2017). Cmos-embedded stt-mram arrays in 2x nm nodes for gp-mcu applications. In *Symposium on VLSI Technology*.
- [212] Sijstermans, F. (2018). The nvidia deep learning accelerator. In *Hot Chips*.
- [213] Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.
- [214] Slaughter, J. M., Nagel, K., Whig, R., Deshpande, S., Aggarwal, S., DeHerrera, M., Janesky, J., Lin, M., Chia, H. J., Hossain, M., Ikegawa, S., Mancoff, F. B., Shimon, G., Sun, J. J., Tran, M., Andre, T., Alam, S. M., Poh, F., Lee, J. H., Chow, Y. T., Jiang, Y., Liu, H. X., Wang, C. C., Noh, S. M., Tahmasebi, T., Ye, S. K., & Shum, D. (2016). Technology for reliable spin-torque mram products. In *IEEE International Electron Devices Meeting (IEDM)*.
- [215] Song, L., Qian, X., Li, H., & Chen, Y. (2017). Pipelayer: A pipelined reram-based accelerator for deep learning. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*: IEEE.
- [216] Song, Y. J., Lee, J. H., Han, S. H., Shin, H. C., Lee, K. H., Suh, K., Jeong, D. E., Koh, G. H., Oh, S. C., Park, J. H., Park, S. O., Bae, B. J., Kwon, O. I., Hwang, K. H., Seo, B. Y., Lee, Y. K., Hwang, S. H., Lee, D. S., Ji, Y., Park, K. C., Jeong, G. T., Hong, H. S., Lee, K. P., Kang, H. K., & Jung, E. S. (2018a). Demonstration of highly manufacturable stt-mram embedded in 28nm logic. In *IEEE International Electron Devices Meeting (IEDM)*.
- [217] Song, Y. J., Lee, J. H., Shin, H. C., Lee, K. H., Suh, K., Kang, J. R., Pyo, S. S., Jung, H. T., Hwang, S. H., Koh, G. H., Oh, S. C., Park, S. O., Kim, J. K., Park, J. C., Kim, J., Hwang, K. H., Jeong, G. T., Lee, K. P., & Jung, E. S. (2016). Highly functional and reliable 8mb stt-mram embedded in 28nm logic. In *IEEE International Electron Devices Meeting (IEDM)*.
- [218] Song, Z. T., Cai, D. L., Li, X., Wang, L., Chen, Y. F., Chen, H. P., Wang, Q., Zhan, Y. P., & Ji, M. H. (2018b). High endurance phase change memory chip implemented based on carbon-doped ge<sub>2</sub>sb<sub>2</sub>te<sub>5</sub> in 40 nm node for embedded application. In *IEEE International Electron Devices Meeting (IEDM)*.
- [219] Sriraman, A., Dhanotia, A., & Wenisch, T. F. (2019). Softsku: optimizing server architectures for microservice diversity at scale. In *Proceedings of the 46th International Symposium on Computer Architecture* (pp. 513–526): ACM.
- [220] Stanford Nanoelectronics Lab (2020). Stanford memory trends.
- [221] Stanisavljevic, M., Pozidis, H., Athmanathan, A., Papandreou, N., Mittelholzer, T., & Eleftheriou, E. (2016). Demonstration of reliable triple-level-cell (tlc) phase-change memory. In *2016 IEEE 8th International Memory Workshop (IMW)*.

- [222] Su, F., Chen, W. H., Xia, L., Lo, C. P., Tang, T., Wang, Z., Hsu, K. H., Cheng, M., Li, J. Y., Xie, Y., Wang, Y., Chang, M. F., Yang, H., & Liu, Y. (2017). A 462gops/j rram-based non-volatile intelligent processor for energy harvesting ioe system featuring nonvolatile logics and processing-in-memory. In *Symposium on VLSI Technology*.
- [223] Sun, X., Wang, P., Ni, K., Datta, S., & Yu, S. (2018). Exploiting hybrid precision for training and inference: A 2t-1fefet based analog synaptic weight cell. In *IEEE International Electron Devices Meeting (IEDM)*.
- [224] Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12), 2295–2329.
- [225] Tableau (2022). Tableau: Business intelligence and analytics software.
- [226] Tambe, T., Hooper, C., Pentecost, L., Jia, T., Yang, E.-Y., Donato, M., Sanh, V., Whatmough, P., Rush, A. M., Brooks, D., & Wei, G.-Y. (2021). Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21 New York, NY, USA: Association for Computing Machinery.
- [227] Tan, A. J., Pesic, M., Larcher, L., Liao, Y. H., Wang, L. C., Bae, J. H., Hu, C., & Salahuddin, S. (2020). Hot electrons as the dominant source of degradation for sub-5nm hzo fefets. In *IEEE Symposium on VLSI Technology*.
- [228] Thomas, L., Jan, G., Le, S., Serrano-Guisan, S., Lee, Y., Liu, H., Zhu, J., Iwata-Harms, J., Tong, R., Patel, S., Sundar, V., Shen, D., Yang, Y., He, R., Haq, J., Teng, Z., Lam, V., Liu, P., Wang, Y., Zhong, T., & Wang, P. (2017). Probing magnetic properties of stt-mram devices down to sub-20 nm using spin-torque fmr. In *IEEE International Electron Devices Meeting (IEDM)*.
- [229] Thomas, L., Jan, G., Serrano-Guisan, S., Liu, H., Zhu, J., Lee, Y., Le, S., Iwata-Harms, J., Tong, R., Patel, S., Sundar, V., Shen, D., Yang, Y., He, R., Haq, J., Teng, Z., Lam, V., Liu, P., Wang, Y., Zhong, T., Fukuzawa, H., & Wang, P. (2018). Stt-mram devices with low damping and moment optimized for llc applications at ox nodes. In *IEEE International Electron Devices Meeting (IEDM)*.
- [230] Tillie, L., Nowak, E., Sousa, R. C., Cyrille, M., Delaet, B., Magis, T., Persico, A., Langer, J., Ocker, B., Prejbeanu, I., & Perniola, L. (2016). Data retention extraction methodology for perpendicular stt-mram. In *IEEE International Electron Devices Meeting (IEDM)*.
- [231] Trentzsch, M., Flachowsky, S., Richter, R., Paul, J., Reimer, B., Utess, D., Jansen, S., Mu-laosmanovic, H., Müller, S., Slesazeck, S., Ocker, J., Noack, M., Müller, J., Polakowski, P., Schreiter, J., Beyer, S., Mikolajick, T., & Rice, B. (2016). A 28nm hkmg super low power embedded nvm technology based on ferroelectric fets. In *IEEE International Electron Devices Meeting (IEDM)*.

- [232] TSMC (2019). eflash documentation  
<https://www.tsmc.com/english/dedicatedfoundry/technology/specialty/eflash>.
- [233] Venkatagiri, R., Mahmoud, A., Hari, S. K. S., & Adve, S. V. (2016). Approxilyzer: Towards a systematic framework for instruction-level approximate computing and its application to hardware resiliency. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (pp. 1–14).
- [234] Walcott, K. R., Humphreys, G., & Gurumurthi, S. (2007). Dynamic prediction of architectural vulnerability from microarchitectural state. *SIGARCH Comput. Archit. News*, 35(2), 516–527.
- [235] Wan, W., Kubendran, R., Eryilmaz, S. B., Zhang, W., Liao, Y., Wu, D., Deiss, S., Gao, B., Raina, P., Joshi, S., Wu, H., Cauwenberghs, G., & Wong, H. P. (2020). A 74 tmacs/w cmos-rram neurosynaptic core with dynamically reconfigurable dataflow and in-situ transposable weights for probabilistic graphical models. In *IEEE International Solid-State Circuits Conference - (ISSCC)*.
- [236] Wang, C., McClellan, C., Shi, Y., Zheng, X., Chen, V., Lanza, M., Pop, E., & Wong, H. P. (2018). 3d monolithic stacked 1t1r cells using monolayer mos2 fet and hbn rram fabricated at low (150c) temperature. In *IEEE International Electron Devices Meeting (IEDM)*.
- [237] Wang, C., Shih, M., Lee, Y., Wang, W., Thomas, L., Lee, Y., Liu, H., Zhu, J., Jan, G., Wang, A., Zhong, T., Wang, P., Lin, D., Chen, C., Chang, C., Weng, C., Chiang, T., Shen, K., Gallagher, W. J., & Chuang, H. (2017). Impact of external magnetic field on embedded perpendicular stt-mram technology qualified for solder reflow. In *IEEE International Electron Devices Meeting (IEDM)*.
- [238] Wang, C. Y., Shih, M. C., Weng, C. H., Chen, C. H., Chang, C. Y., Wang, W., Chiang, T. W., Hung, A., Chuang, H., & Gallagher, W. J. (2020). Reliability demonstration of reflow qualified 22nm stt-mram for embedded memory applications. In *IEEE Symposium on VLSI Technology*.
- [239] Wang, Z., Li, Z., Xu, L., Dong, Q., Su, C. I., Chu, W. T., Tsou, G., Chih, Y. D., Chang, T. Y. J., Sylvester, D., Kim, H. S., & Blaauw, D. (2020). An all-weights-on-chip dnn accelerator in 22nm ull featuring 24x1 mb erram. In *2020 IEEE Symposium on VLSI Circuits*.
- [240] Wei, L., Alzate, J. G., Arslan, U., Brockman, J., Das, N., Fischer, K., Ghani, T., Golonzka, O., Hentges, P., Jahan, R., Jain, P., Lin, B., Meterelliyoz, M., O'Donnell, J., Puls, C., Quintero, P., Sahu, T., Sekhar, M., Vangapaty, A., Wiegand, C., & Hamzaoglu, F. (2019). A 7mb stt-mram in 22ffl finfet technology with 4ns read sensing time at 0.9v using write-verify-write scheme and offset-cancellation sensing technique. In *IEEE International Solid-State Circuits Conference - (ISSCC)*.



- [241] Wei, Z., Katoh, Y., Ogasahara, S., Yoshimoto, Y., Kawai, K., Ikeda, Y., Eriguchi, K., Ohmori, K., & Yoneda, S. (2016). True random number generator using current difference based on a fractional stochastic model in 40-nm embedded rram. In *IEEE International Electron Devices Meeting (IEDM)*.
- [242] Whatmough, P. N., Lee, S. K., Brooks, D., & Wei, G. (2018). Dnn engine: A 28-nm timing-error tolerant sparse deep neural network processor for iot applications. *IEEE Journal of Solid-State Circuits*.
- [243] Whatmough, P. N., Lee, S. K., Donato, M., Hsueh, H.-C., Xi, S., Gupta, U., Pentecost, L., Ko, G. G., Brooks, D., & Wei, G.-Y. (2019a). A 16nm 25mm<sup>2</sup> soc with a 54.5x flexibility-efficiency range from dual-core arm cortex-a53 to efpga and cache-coherent accelerators. In *2019 Symposium on VLSI Circuits* (pp. C34–C35).
- [244] Whatmough, P. N., Zhou, C., Hansen, P., Venkataramanaiah, S. K., Seo, J., & Mattina, M. (2019b). Fixynn: Energy-efficient real-time mobile computer vision hardware acceleration via transfer learning. In *Proceedings of Machine Learning and Systems 2019, MLSys*.
- [245] Wu, C., Raghavendra, R., Gupta, U., Acun, B., Ardalani, N., Maeng, K., Chang, G., Behram, F. A., Huang, J., Bai, C., Gschwind, M., Gupta, A., Ott, M., Melnikov, A., Candido, S., Brooks, D., Chauhan, G., Lee, B., Lee, H. S., Akyildiz, B., Balandat, M., Spisak, J., Jain, R., Rabbat, M., & Hazelwood, K. (2021). Sustainable AI: environmental implications, challenges and opportunities. *CoRR*, abs/2111.00364.
- [246] Wu, H., Yao, P., Gao, B., Wu, W., Zhang, Q., Zhang, W., Deng, N., Wu, D., Wong, H. P., Yu, S., & Qian, H. (2017). Device and circuit optimization of rram for neuromorphic computing. In *IEEE International Electron Devices Meeting (IEDM)*.
- [247] Wu, J., Mo, F., Saraya, T., Hiramoto, T., & Kobayashi, M. (2020). A monolithic 3d integration of rram array with oxide semiconductor fet for in-memory computing in quantized neural network ai applications. In *IEEE Symposium on VLSI Technology*.
- [248] Wu, J. Y., Chen, Y. S., Khwa, W. S., Yu, S. M., Wang, T. Y., Tseng, J. C., Chih, Y. D., & Diaz, C. H. (2018). A 40nm low-power logic compatible phase change memory technology. In *IEEE International Electron Devices Meeting (IEDM)*.
- [249] Wulf, W. A. & McKee, S. A. (1995). Hitting the memory wall: Implications of the obvious. *SIGARCH Comput. Archit. News*, 23(1), 20–24.
- [250] Xu, X., Luo, Q., Gong, T., Lv, H., Long, S., Liu, Q., Chung, S. S., Li, J., & Liu, M. (2016). Fully cmos compatible 3d vertical rram with self-aligned self-selective cell enabling sub-5nm scaling. In *IEEE Symposium on VLSI Technology*.

- [251] Xu, X., Tai, L., Gong, T., Yin, J., Huang, P., Yu, J., Dong, D. N., Luo, Q., Liu, J., Yu, Z., Zhu, X., Wu, X. L., Liu, Q., LV, H., & Liu, M. (2018). 40x retention improvement by eliminating resistance relaxation with high temperature forming in 28 nm rram chip. In *IEEE International Electron Devices Meeting (IEDM)*.
- [252] Xue, C., Huang, T., Liu, J., Chang, T., Kao, H., Wang, J., Liu, T., Wei, S., Huang, S., Wei, W., Chen, Y., Hsu, T., Chen, Y., Lo, Y., Wen, T., Lo, C., Liu, R., Hsieh, C., Tang, K., & Chang, M. (2020). A 22nm 2mb rram compute-in-memory macro with 121-28tops/w for multibit mac computing for tiny ai edge devices. In *IEEE International Solid-State Circuits Conference - (ISSCC)*.
- [253] Yamauchi, T., Yamaguchi, Y., Kono, T., & Hidaka, H. (2016). Embedded flash technology for automotive applications. In *IEEE International Electron Devices Meeting (IEDM)*.
- [254] Yang, C. F., Wu, C. Y., Yang, M. H., Wang, W., Yang, M. T., Chien, T. C., Fan, V., Tsai, S. C., Lee, Y. H., Chu, W. T., & Hung, A. (2020a). Industrially applicable read disturb model and performance on mega-bit 28nm embedded rram. In *IEEE Symposium on VLSI Technology*.
- [255] Yang, J., Xue, X., Xu, X., Lv, H., Zhang, F., Zeng, X., Chang, M., & Liu, M. (2020b). A 28nm 1.5mb embedded 1t2r rram with 14.8 mb/mm<sup>2</sup> using sneaking current suppression and compensation techniques. In *IEEE Symposium on VLSI Circuits*.
- [256] Yang, T., Li, K., Chiang, Y., Lin, W., Lin, H., & Chang, M. (2018). A 28nm 32kb embedded 2t2mtj stt-mram macro with 1.3ns read-access time for fast and reliable read applications. In *IEEE International Solid - State Circuits Conference - (ISSCC)*.
- [257] Yasin, A. (2014). A top-down method for performance analysis and counters architecture. *ISPASS 2014*.
- [258] Yasuda, S., Ohba, K., Mizuguchi, T., Sei, H., Shimuta, M., Aratani, K., Shiimoto, T., Yamamoto, T., Sone, T., Nonoguchi, S., Okuno, J., Kouchiyama, A., Otsuka, W., & Tsutsui, K. (2017). A cross point cu-rram with a novel ots selector for storage class memory applications. In *Symposium on VLSI Technology*.
- [259] Yoo, H. K., Kim, J. S., Zhu, Z., Choi, Y. S., Yoon, A., MacDonald, M. R., Lei, X., Lee, T. Y., Lee, D., Chae, S. C., Park, J., Hemker, D., Langan, J. G., Nishi, Y., & Hong, S. J. (2017). Engineering of ferroelectric switching speed in si doped hfo<sub>2</sub> for high-speed 1t-feram application. In *IEEE International Electron Devices Meeting (IEDM)*.
- [260] Yosinski, J., Clune, J., Bengio, Y., & Lipsons, H. (2014). How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NeurIPS'14* (pp. 3320–3328). Cambridge, MA, USA: MIT Press.

- [261] Yu, S., Li, Z., Chen, P., Wu, H., Gao, B., Wang, D., Wu, W., & Qian, H. (2016). Binary neural network with 16 mb rram macro chip for classification and online training. In *IEEE International Electron Devices Meeting (IEDM)*.
- [262] Zeng, X., Song, X., Ma, T., Pan, X., Zhou, Y., Hou, Y., Zhang, Z., Li, K., Karypis, G., & Cheng, F. (2020). Repurpose open data to discover therapeutics for covid-19 using deep learning. *Journal of Proteome Research*, 19(11), 4624–4636.
- [263] Zhang, F., Zhang, H., Shrestha, P. R., Zhu, Y., Maize, K., Krylyuk, S., Shakouri, A., Campbell, J. P., Cheung, K. P., Bendersky, L. A., Davydov, A. V., & Appenzeller, J. (2018). An ultra-fast multi-level mte2-based rram. In *IEEE International Electron Devices Meeting (IEDM)*.
- [264] Zhang, X., Xu, Y., Hu, H., Liu, Y., Guo, Z., & Wang, Y. (2013). Modeling and analysis of skype video calls: Rate control and video quality. *IEEE Transactions on Multimedia*.
- [265] Zhang, Y., Liao, X., Jin, H., He, L., He, B., Liu, H., & Gu, L. (2021). Depgraph: A dependency-driven accelerator for efficient iterative graph processing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (pp. 371–384).
- [266] Zhang, Y., Zhang, L., Wen, W., Sun, G., & Chen, Y. (2012). Multi-level cell stt-ram: Is it realistic or just a dream? In *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.
- [267] Zhao, H., Xue, L., Chi, P., & Zhao, J. (2017). Approximate image storage with multi-level cell stt-mram main memory. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.
- [268] Zhao, L., Chen, H.-Y., Wu, S.-C., Jiang, Z., Yu, S., Hou, T.-H., Wong, H. . P., & Nishi, Y. (2014a). Improved multi-level control of rram using pulse-train programming. In *Proceedings of Technical Program - 2014 International Symposium on VLSI Technology, Systems and Application (VLSI-TSA)*.
- [269] Zhao, L., Chen, H.-Y., Wu, S.-C., Jiang, Z., Yu, S., Hou, T.-H., Wong, H.-S. P., & Nishi, Y. (2014b). Multi-level control of conductive nano-filament evolution in hfo2 rram by pulse-train operations. *Nanoscale*.