



LUND UNIVERSITY

Near-Memory Computing Architectures for Scalable Edge AI Applications

Nouripayam, Masoud

2025

Document Version:

Publisher's PDF, also known as Version of record

[Link to publication](#)

Citation for published version (APA):

Nouripayam, M. (2025). *Near-Memory Computing Architectures for Scalable Edge AI Applications*. Electrical and Information Technology, Lund University.

Total number of authors:

1

Creative Commons License:

Unspecified

General rights

Unless other specific re-use rights are stated the following general rights apply:

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: <https://creativecommons.org/licenses/>

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Near-Memory Computing Architectures for Scalable Edge AI Applications

Doctoral Thesis

Masoud Nouripayam



LUND
UNIVERSITY

Department of Electrical and
Information Technology
Lund, November 2025

Academic thesis for the degree of Doctor of Philosophy, which, by due permission of the Faculty of Engineering at Lund University, will be publicly defended on Friday, 21 November, 2025, at 9:15 a.m. in lecture hall E:1406, Department of Electrical and Information Technology, Ole Römers Väg 3, 223 63 Lund, Sweden. The thesis will be defended in English.

The Faculty Opponent will be Professor Snorre Aunet, Norwegian University of Science and Technology (NTNU), Norway.

<i>Organisation:</i> LUND UNIVERSITY Department of Electrical and Information Technology Ole Römers Väg 3 223 63 Lund Sweden	<i>Document Type:</i> DOCTORAL THESIS	
	<i>Date of Issue:</i> November 2025	
	<i>Sponsoring Organisation(s):</i> Swedish Research Council (VR) – ClassIC (SSF-funded research center) – GlobalFoundries – Codasip	
<i>Author:</i> Masoud Nouripayam		
<i>Title:</i> Near-Memory Computing Architectures for Scalable Edge AI Applications		
<i>Abstract:</i> Artificial intelligence (AI) and machine learning (ML) are rapidly permeating nearly every aspect of modern life, from personal devices and autonomous systems to industrial automation and environmental monitoring. The growing demand for intelligence at the network edge is reshaping how computing hardware is conceived and built. Edge AI platforms are expected to deliver high throughput within tight energy and area budgets, operate reliably at low voltages, and adapt to diverse workloads, while data movement between processors and memory continues to dominate system cost. These trends position memory-centric computing as a compelling alternative to conventional architectures. Addressing these challenges begins with rethinking on-chip memory. A dual-port six-transistor (6T) static random-access memory (SRAM) was developed that combines high density with concurrent read and write capability, enabling reliable low-voltage operation and significant energy savings. Targeted at resource-constrained edge platforms, this design also achieves notable area reductions compared with conventional dual-port implementations. Building on this foundation, a synthesizable distributed-SRAM (SD-SRAM) framework was devised to extend the scalability of on-chip memories. By pushing the area-energy break-even point from 2–4 kb to about 14–16 kb and enabling modular partitioning with parallel access, this framework provides a robust memory substrate well suited to data-centric processing. Extending these memory innovations, a digital near-memory computing (NMC) architecture was integrated into RISC-V-based microcontroller units (MCUs), performing multiply-accumulate (MAC) operations directly within cache SRAM macros without altering the conventional SRAM architecture. The NMC engine supports configurable convolutional neural network (CNN) dataflows and achieves nearly two orders of magnitude higher throughput while sharply reducing internal data movement relative to processor-memory baselines. To further improve efficiency under strict energy constraints, a family of approximate multipliers with operand-aware error control provides tunable accuracy–cost trade-offs, enabling substantial reductions in area and power while preserving inference quality for error-tolerant workloads. These contributions converge in a scalable and modular NMC platform capable of flexible CNN acceleration at the edge. By tightly coupling local memory, approximate processing, and near-memory execution, this work advances the development of compact, high-throughput, and energy-efficient AI hardware. The presented concepts and implementations also establish a foundation for compiler–hardware co-optimization, automated NMC generation, and seamless integration into multimodal, sensor-driven embedded systems.		
<i>Keywords:</i> Data-Centric, Near-Memory Computing, SRAM, Artificial Intelligence, Convolutional Neural Network, Edge Computing, Hardware Acceleration, Approximate Computing, Energy-Efficiency		
<i>Classification System and/or Index Terms</i> e.g. Electronic Engineering, Nano Technology		<i>Language:</i> English
<i>Supplementary Bibliographical Information:</i> –		<i>ISBN (printed):</i> 978-91-8104-741-7
<i>Key title and ISSN:</i> Series of Licentiate and Doctoral Theses; 1654-790X, No. 189		<i>ISBN (digital):</i> 978-91-8104-742-4
<i>Recipient's Notes:</i>	<i>Number of Pages:</i> 170	<i>Price:</i>
	<i>Security Classification:</i> Unclassified	

General Permissions:
I, the undersigned, being the copyright owner and author of the above-mentioned thesis and its abstract, hereby grant to all reference sources permission to publish and disseminate said abstract.

Signature: Masoud Nouripayam

Date: October 2025

Near-Memory Computing Architectures for Scalable Edge AI Applications

Doctoral Thesis

Masoud Nouripayam



LUND
UNIVERSITY

Department of Electrical and
Information Technology
Lund, November 2025

Masoud Nouripayam
Department of Electrical and Information Technology
Lund University
Ole Römers Väg 3, 223 63 Lund, Sweden

Series of Licentiate and Doctoral Theses
ISSN 1654-790X, No. 189
ISBN 978-91-8104-741-7 (printed)
ISBN 978-91-8104-742-4 (digital)

© 2025 Masoud Nouripayam
This thesis is typeset using $\text{\LaTeX}2_{\epsilon}$ with the body text in Palatino and Goudy
Initials, headings in Helvetica.

Frontispiece: AI-generated illustration inspired by connectivity, embedded electronics, and the future of processing and communication.

Printed by Tryckeriet i E-huset, Lund University, Lund, Sweden.

No part of this thesis may be reproduced or transmitted in any form or by any means without written permission from the author. Distribution of the original thesis in full, however, is permitted without restriction.

*“Ställning, anseende, framtid.
Som om jag icke var dag och var stund vore beredd
att packa dessa kollin ombord på det första skepp,
som kommer lastat med en handling.
En verklig handling.”*

Hjalmar Söderberg

Abstract



ARTIFICIAL intelligence (AI) and machine learning (ML) are rapidly permeating nearly every aspect of modern life, from personal devices and autonomous systems to industrial automation and environmental monitoring. The growing demand for intelligence at the network edge is reshaping how computing hardware is conceived and built. Edge AI platforms are expected to deliver high throughput within tight energy and area budgets, operate reliably at low voltages, and adapt to diverse workloads, while data movement between processors and memory continues to dominate system cost. These trends position memory-centric computing as a compelling alternative to conventional architectures.

Addressing these challenges begins with rethinking on-chip memory. A dual-port six-transistor (6T) static random-access memory (SRAM) was developed that combines high density with concurrent read and write capability, enabling reliable low-voltage operation and significant energy savings. Targeted at resource-constrained edge platforms, this design also achieves notable area reductions compared with conventional dual-port implementations. Building on this foundation, a synthesizable distributed-SRAM (SD-SRAM) framework was devised to extend the scalability of on-chip memories. By pushing the area-energy break-even point from 2–4 kb to about 14–16 kb and enabling modular partitioning with parallel access, this framework provides a robust memory substrate well suited to data-centric processing.

Extending these memory innovations, a digital near-memory computing (NMC) architecture was integrated into RISC-V-based microcontroller units (MCUs), performing multiply-accumulate (MAC) operations directly within

cache SRAM macros without altering the conventional SRAM architecture. The NMC engine supports configurable convolutional neural network (CNN) dataflows and achieves nearly two orders of magnitude higher throughput while sharply reducing internal data movement relative to processor–memory baselines. To further improve efficiency under strict energy constraints, a family of approximate multipliers with operand-aware error control provides tunable accuracy–cost trade-offs, enabling substantial reductions in area and power while preserving inference quality for error-tolerant workloads.

These contributions converge in a scalable and modular NMC platform capable of flexible CNN acceleration at the edge. By tightly coupling local memory, approximate processing, and near-memory execution, this work advances the development of compact, high-throughput, and energy-efficient AI hardware. The presented concepts and implementations also establish a foundation for compiler–hardware co-optimization, automated NMC generation, and seamless integration into multimodal, sensor-driven embedded systems.

Popular Science Summary



WHAT if your glasses could describe the world around you in real time, translating street signs as you walk through a foreign city, pointing out hidden landmarks, or warning you about a car rushing up from behind? It may sound futuristic, like a new layer of intelligence we do not yet have, but technology is moving rapidly in that direction. The gadgets we already use every day, from phones and wearables to cars and home assistants, are closer to this vision than it seems. The challenge that remains is not a lack of processing power.

Modern chips can already perform enormous calculations. The real limitation is energy. Battery capacity is finite, and every milliwatt should be used efficiently. Looking closer at where that energy goes reveals something surprising: the bulk of the energy is consumed not by computation but by memory. Constantly moving data back and forth between memory and the processor drains energy and time, creating the true bottleneck.

It is like watching a comedian who, after every joke, has to pause, pull out a notebook, and read the next line. The show continues, but the rhythm breaks, the timing slips, and the audience loses focus. Computers face a similar problem when data is repeatedly shuttled between memory and processor, breaking the rhythm of computation.

This led researchers to a bold question: what if memory could also compute? Imagine that same comedian, but now she knows every joke by heart, can improvise when the crowd reacts, and keeps the performance flowing without missing a beat. That is the essence of memory-centric computing.

The first step in this thesis was to rethink the memory itself. Conventional memories can either read or write at a given moment, but not both. Enabling simultaneous operations usually requires additional hardware, which adds cost and complexity. To address this, a new memory design was proposed that allows reading and writing from different locations at the same time, improving efficiency while remaining compact.

Building on this foundation, the concept of distributed memory was explored, where a single large memory block is replaced by many smaller ones placed around the processing logic and operating in parallel. This approach reduces traffic bottlenecks and provides greater flexibility to construct memory structures tailored to specific tasks.

The major breakthrough came with merging memory and processing. Instead of continuously transferring data back and forth to a processor, a near-memory computing architecture was designed and prototyped, demonstrating that information can be processed directly where it is stored. This approach avoids unnecessary data movement, lowers energy consumption, and accelerates data-intensive tasks such as image recognition and real-time sensor analysis. Importantly, these concepts were not only explored in theory but also validated on fabricated integrated circuits.

Another contribution of this thesis is the development of an approximate computing technique. Inspired by the fact that many AI tasks can tolerate small errors, these circuits deliberately relax precision to gain substantial improvements in speed and energy efficiency. By skipping non-critical details, they act as efficient shortcuts that preserve the overall outcome while reducing cost. It is like the comedian slipping in an improvised line instead of sticking word-for-word to the script. The timing improves, and the audience still laughs.

Together, these innovations form a flexible computing platform where memory itself can execute neural networks, the same brain-inspired models that power modern AI. With adaptable dataflow and support for diverse workloads, this work points toward the next generation of intelligent, battery-driven devices.

This is the path taken in this thesis: moving from conceptual ideas to hardware prototypes that demonstrate compact, efficient, and responsive systems. Such devices will not only store data but also interpret the world and act on it, from glasses that explain your surroundings to vehicles that react instantly and assistants that feel truly present.

Acknowledgments



s my PhD journey nears its conclusion, I am deeply grateful to many individuals who have supported me along the way. I could not have reached this point without your encouragement and support.

I would like to begin by expressing my gratitude to my main supervisor, **Joachim Rodrigues**, for his support throughout this period. His understanding during challenging personal circumstances made a real difference, and his guidance has contributed to bringing this work to completion. I would also like to thank my co-supervisor, **Liang Liu**, for his technical input, feedback, and personal support, which have contributed to the development of this work. A heartfelt thanks to **Stefan Höst** and **Daniel Sjöberg**, two department heads over the years, who have always been approachable, kind, and supportive. The spotlight in research may fall on results, but it is the unseen hands that make them possible. I warmly thank **Linda Bienen**, **Elisabeth Nordström**, **Erik Göthe**, **Margit Billesö**, **Pia Bruhn**, **Anne Andersson**, and **Elisabeth Ohlsson**. Your behind-the-scenes support has made it possible for us to focus on our research, and panic in peace.

Erik Jonsson, EIT hade varit en betydligt svårare plats utan ditt outhärliga stöd. Tack också för alla roliga samtal och skämt vi har delat om allt möjligt genom åren. **Stefan Molund**, Du kommer att saknas, och jag hoppas att du njuter av din välförtjänta pension. **Bertil Lindvall**, stort tack för att du hjälpte mig att bli varm i kläderna när jag, något skakigt, skulle ta över i Helsingborg, även om det aldrig fick bli som det skulle. **Sirvan**, thank you for simply being who you are: down-to-earth, supportive, and considerate. While your decisive role in our research has been central to many EIT publications, what I value most is your very generous friendship.

I would like to thank **Pietro Andreani**, **Henrik Sjöland**, **Markus Törmänen**, **Erik Larsson**, **Per Andersson**, and **Viktor Öwall**. I am deeply grateful for the opportunity to conduct my research alongside you and to learn from your vast knowledge, which has enriched my work in countless ways.

Ali Zaher, Steffen, Mojtaba, Oskar, Farokh, Jesús, Sidra, MinKeun, Siyu, Therese, Jonas, and Victor Å.: thank you for the time we shared and for everything I learned from you. Each of you added something unique to my journey.

Arturo, from work-related frustrations and successes to personal matters, your kindness, friendship, and support have meant so much to me. I have always admired your ability to stay calm and manage stress, although at times to an almost annoyingly admirable degree. **Rikard**, I doubt you missed a single workshop, but more than your dedication, I remember your kindness and that ever-smiling face, even when things got hectic. **MJ**, you never stopped surprising us with your complicated relationship with chess and your mastery of wonderfully obscure, rarely used words in every language you know. Beneath your quiet demeanor lies a kind and gentle soul I deeply appreciate. **Hamid**, I have always admired your unshakable calm and your ability to keep things refreshingly simple, sidestepping the over-complications that most of us so easily fall into. **Lucas**, my vibrant Brazilian friend who, against all odds, is not obsessed with football, thank you for bringing boundless energy to our boring days. I have no doubt you are exactly where you are meant to be in the world of entrepreneurship.

My heartfelt appreciation to **William, Kristoffer, Alex, and Sergio** for the good laughs, afterworks, and coffee breaks, and for being genuinely easy to be around; and to **Dumitra, Lina, Love, Vilgot, Sijia, Wenbo, Ilayda, and Joel** for the encouragement, good company, and all the shared moments along the way. Hang in there, I promise you will survive this intense and stressful chapter, hopefully with your sanity mostly intact!

Babak, without your support the valuable experience I gained during my time at Xenergic would not have been possible. Above everything else, I am grateful for your friendship that always extended beyond work. **Hemanth, Reza**, thank you for all the technical conversations and the dark humor we shared over the years. **Xiao**, my good old mate, I have always admired your reliability and your drive as a true doer. You have been my steadfast companion throughout those hardworking years that ultimately culminated in my PhD, and I am deeply grateful for that. I am also grateful to **Tom, Ajay, Adam** and all my Xenergic colleagues for the discussions, the technical help, the joy of sharing Chinese and Indian food, and the unforgettable competitions to see who could handle the most chili.

I also want to express my gratitude to my dearest companions, **Iman** and **Baktash**. It is a rare fortune to meet people who not only share your outlook on life but also truly resonate with your way of being, and that is what has made our friendship so meaningful. Iman, I have always learned from and admired your depth of knowledge, your insight, and your unique way of seeing the world. Baktash, your thoughtfulness, joyful and kind spirit, and

near-wizardly command of technology have never ceased to impress me. You both have supported me in ways that go beyond words. **Behshid**, thank you for the joy our group of five shared, and for patiently enduring me, Iman, and Baktash through our endless conversations about everything and nothing.

My interactions have never been limited to just the ASIC corridor. I have always enjoyed the conversations and friendships with colleagues from across our EIT department, and I am grateful to **Ove, Fredrik Rusek, Fredrik Tufvesson, Vincent, Erik Lind, Johannes Svensson, Amir, Juan V., Ashkan, Hedieh, Saeedeh, Omar, Hassan, Sahar, Pegah, Saeed Bastani, Mahdi Rezayati, Björn, Emma, Enrico, Markus Hellenbrand, and Adam Jönsson.**

Farnaz Moradi, I truly miss the thoughtful, laughter-filled moments we shared during our time together in the department.

As a non-EU PhD student in Sweden, one cannot help but feel a certain sense of “gratitude” toward **Migrationsverket** for the seemingly endless limbo while waiting for one’s case to be processed. This, coupled with the impact of **COVID-19**, effectively prevented me from attending many planned international events abroad.

My gratitude also extends to the dear ones beyond the professional sphere, whose presence through both good and difficult times, and the comfort offered when it was needed most, even if not directly tied to work, have helped me persevere. In no particular order, thanks to **Mohsen T., Anne, Shahab, Neda, Farnaz, Mohsen Z., Behnaz, Axel, Mehdi H.H., Kiumars, Daniel D.K.**

Kasra, Rebecca, ni står mig så nära i hjärtat, tack för att ni alltid har funnits där för mig. **Sigrid, Andreas**, ni har hållit i mig under de tuffa dagarna, tack, var ni än är och vad ni än gör.

Mommy, Daddy, thank you for standing by me at every stage of life, whatever I have achieved is rooted in your love, patience, and sacrifice. To my lovely sisters, **Mojgan** and **Mojdeh**, thank you for your care and encouragement. **Mojdeh**, you are not only my sister but also a true friend, and your constant support means the world to me. To the beloved additions to our family – **Hamid, Mehdi, Amin, Kimia, Lily**, and little **Rosie** – your warmth and kindness have brightened our home and strengthened our bond.

And finally, my life partner, **Naghmeh**. We have shared countless ups and downs, and from the bottom of my heart, I know that whatever I have achieved, I did not achieve alone. Your unwavering presence, patience, and love have been with me every step of the way.

Masoud Nouripayam
Lund, Oct. 2025

Contents

Abstract	v
Popular Science Summary	vii
Acknowledgments	ix
Contents	xiii
Preface	xvii
Structure of the Thesis	xvii
Included Papers	xvii
Related Work	xviii
Acronyms and Abbreviations	xxi
.	xxi
INTRODUCTION	1
1 Motivation and Scope	3
1.1 The New Era of Analytics	3
1.2 The Past Is Prologue	4
1.3 Connectivity and Intelligence of Things	7
1.4 AI and Power Consumption	8
1.5 Emerging Computing Trends	9
1.6 Thesis Outline	12

2 Power Optimization: A Strategic Perspective **15**

2.1 Power Consumption in Digital Systems 15

2.2 Power Optimization Framework 16

2.2.1 Architecture, Algorithms, and Systems Optimization . . . 18

2.2.2 Circuit Level Optimization 20

2.2.3 Runtime Optimization 22

2.3 Thesis Contribution 24

3 Memory and Efficiency **25**

3.1 Memory System 25

3.2 SRAM 29

3.2.1 Six-Transistor (6T) SRAM: Architectural Foundation . . . 29

3.2.2 Performance, Reliability, and Multi-Port Realization . . . 31

3.2.3 Challenges in SRAM design 34

3.2.4 Power Optimization and Design-Challenge Mitigation . . 35

3.3 Standard-Cell Based Memories 36

3.3.1 SCM Implementation and Architecture 37

3.4 Hybrid Memory Structures 38

3.5 Emerging Memory Technologies 40

3.6 Thesis Contribution 41

4 Synthesizable Distributed SRAM **43**

4.1 From Fixed Macros to Flexible Integration 43

4.2 Architecture 44

4.2.1 Generation Flow 45

4.2.2 Bitline Hierarchy 46

4.3 Proposed Design 47

4.4 Results and Discussion 48

4.5 Thesis Contribution 51

5 Approximate Computing: Techniques and Design Strategies **53**

5.1 Turning Imperfection into Efficiency 53

5.2 Design Methodologies 55

5.3 Approximate Multiplier Design 58

5.3.1 Error Compensation Modules for AM Design 61

5.4 Accuracy and Quality Metrics 63

5.4.1 Application-Independent Metrics	63
5.4.2 Application-Specific Metrics	64
5.4.3 Error Analysis Methodologies	65
5.5 Thesis Contribution	66
6 Rebooting Computing: Beyond von Neumann Architectures	67
6.1 From Processor-Centric to Data-Centric Architectures	67
6.1.1 Historical Context and Resurgence	69
6.1.2 Taxonomy and Implementation Techniques	70
6.1.3 Architectural Paradigms	71
6.1.4 Memory Technology Enablers	75
6.1.5 Architectural Variations	77
6.1.6 Dataflow Strategies for Accelerating NN Workloads . . .	77
6.1.7 Representative Implementations	80
6.2 Energy, Performance, and System-Level Tradeoffs	81
6.2.1 Evaluation Metrics	81
6.2.2 Throughput and Energy Efficiency	82
6.3 Integration Challenges	84
6.3.1 Area, Density, and Device Non-Idealities	84
6.3.2 Software Stack, Execution, and Programming Support . .	84
6.4 Thesis Contribution	86
7 Conclusion and Future Directions	89
Bibliography	91

PAPERS	117
I A Low-Voltage 6T Dual-Port Configured SRAM with Wordline Boost in 28 nm FD-SOI	119
II An Energy-Efficient Near-Memory Computing Architecture for CNN Inference at Cache Level	125
III A Scalable All-Digital Near-Memory Computing Architecture for Edge AIoT Applications	131
IV NMCu-CNN: A Scalable Near-Memory Computing Co-Processor on a RISC-V MCU in 22 nm FDSOI	151
V A Scalable Hardware-efficient Approximate Multiplier for Error-Resilient Applications	159

Preface



THIS dissertation presents the academic work carried out during my doctoral studies within the Division of Integrated Electronic Systems at the Department of Electrical and Information Technology, Lund University, Sweden. The content is structured into two main sections:

STRUCTURE OF THE THESIS

- **INTRODUCTION**

Offers a comprehensive overview of the research domain and the key areas explored during my studies. It outlines the background, research motivations, significant findings, and concludes with reflections on the contributions and potential directions for future advancements in the field.

- **PAPERS**

The scientific contributions comprise five original, peer-reviewed publications that highlight the scope and impact of the research conducted, with an outline of my individual contributions to each work presented below.

INCLUDED PAPERS

The following papers form the main body of this thesis, and the corresponding published or draft versions are appended at the end.

Paper I: MASOUD NOURIPAYAM, J. RODRIGUES, X. LUO, T. JOHANSSON, AND B. MOHAMMADI, "A Low-Voltage 6T Dual-Port Configured SRAM with Wordline Boost in 28 nm FD-SOI", *IEEE 47th European Solid State Circuits Conference (ESSCIRC)*, pp. 459–462, Sep 2021, doi: 10.1109/ESSCIRC53450.2021.9567785.

► **Contribution:** I carried out the chip-level ASIC implementation and sign-off in STM 28 nm FD-SOI technology, performed simulations, conducted silicon bring-up and measurements, authored the manuscript under the supervision of the second and fifth authors, and presented the work at the conference.

Paper II: MASOUD NOURIPAYAM, A. PRIETO, V. K. KISHORELAL, AND J. RODRIGUES, “An Energy-Efficient Near-Memory Computing Architecture for CNN Inference at Cache Level”, *IEEE 28th International Conference on Electronics, Circuits, and Systems (ICECS)*, pp. 1–4, Nov. 2021, doi: 10.1109/ICECS53924.2021.9665530.

► **Contribution:** I proposed the main idea and architecture and, in collaboration with the second author, designed and developed the system, carried out the ASIC implementation and simulations, and authored the paper.

Paper III: MASOUD NOURIPAYAM, A. PRIETO, AND J. RODRIGUES, “A Scalable All-Digital Near-Memory Computing Architecture for Edge AIoT Applications”, *IEEE Access*, vol. 13, pp. 108609–108625, Jun. 2025, doi: 10.1109/ACCESS.2025.3582013.

► **Contribution:** I further developed the main idea and architecture and, in collaboration with the second author, designed and developed the system, carried out the ASIC implementation and simulations, and authored the paper.

Paper IV: A. PRIETO, MASOUD NOURIPAYAM, K. WESTRING, S. C. MOHEDANO, A. ALLEJORD, L. SVENSSON, P. ANDERSSON, AND J. RODRIGUES, “NMCu-CNN: A Scalable Near-Memory Computing Co-Processor on a RISC-V MCU in 22 nm FD-SOI”, *IEEE 38th International Workshop on Signal Processing Systems (SiPS)*, Accepted for presentation, Nov. 2025.

► **Contribution:** I designed and developed the system at the RTL level and co-authored the paper together with the first author. The ASIC implementation and silicon measurements were carried out in collaboration with the other co-authors.

Paper V: MASOUD NOURIPAYAM, S. C. MOHEDANO, A. PRIETO, AND J. RODRIGUES, “Scalable Hardware-Efficient Approximate Multipliers for Error-Resilient Applications”, *Integration, the VLSI Journal*, Submitted.

► **Contribution:** I proposed and developed the technique, conducted the block-level analyses, and authored the paper. The application-level analyses were performed in collaboration with the second author.

RELATED WORK

The following works are not included in the thesis, but summarize related work that I was involved in:

- A. PRIETO, MASOUD NOURIPAYAM, K. WESTRING, W. MARNFELDT, A. ALLEFJORD, S. C. MOHEDANO, V. ÅBERG, L. SVENSSON, P. ANDERSSON, AND J. RODRIGUES, “LUCIA: Compute Inter-chip Architecture for Multi-chiplet Acceleration in 22 nm FD-SOI”, *To be submitted to IEEE Custom Integrated Circuits Conference 2026*.
- A. BROKALAKIS, P. MALAKONAKIS, K. HARTEROS, D. ANDRONIKOUA, I. GALANOMATIS, C. SAVVAKOS, G. TANG, K. VADIVEL, G.J. VAN SCHAİK, S. IOANNIDIS, K. GEORGOPOULOS, G. CHRYSOS, A. ALLEFJORD, MASOUD NOURIPAYAM, A. PRIETO, J. RODRIGUES, I. MAVROIDIS, I. PAPAEF-STATHIOU, “REBECCA: Reconfigurable Heterogeneous Highly Parallel Processing Platform for safe and secure AI”, *Microprocessors and Microsystems (MICPRO)*, *Under review*.
- A. PRIETO, MASOUD NOURIPAYAM, K. WESTRING, A. ALLEFJORD, S. C. MOHEDANO, L. WALTHER, AND J. RODRIGUES, “Poster: An energy-efficient RISC-V based SoC accelerator for CNN inference”, *RISC-V Summit Europe*, Germany, 2024. [Online]. Available: https://riscv-europe.org/summit/2024/media/proceedings/posters/156_poster.pdf
- K. WESTRING, L. SVENSSON, P. ANDERSSON, MASOUD NOURIPAYAM, A. PRIETO, S. C. MOHEDANO, AND J. RODRIGUES, “Poster: Low-power acceleration of CNNs using near-memory computing on a RISC-V SoC”, *RISC-V Summit Europe*, Germany, 2024. [Online]. Available: https://riscv-europe.org/summit/2024/media/proceedings/posters/159_poster.pdf

Acronyms and Abbreviations

1R1W	one-read/one-write
2RW	two-read/write
5G	Fifth Generation Wireless Communication
6G	Sixth Generation Wireless Communication
6T	Six-Transistor
8T	Eight-Transistor
ABB	Adaptive Body Biasing
AC	Approximate Computing
Accu.	Accurate
ADC	Analog-to-Digital Converter
ADP	Area-Delay Product
AI	Artificial Intelligence
AIF	Approximate Integer Format
AIMC	Analog In-Memory Computing
AIoT	Artificial Intelligence of Things
ALU	Arithmetic Logic Unit
AM	Approximate Multiplier
AMD	Advanced Micro Devices
ANN	Artificial Neural Network
AOC	Accurate Output Count
AOI	AND-OR-Invert
ASIC	Application-Specific Integrated Circuit

ASIP	Application-Specific Instruction Processor
ASVG	Adaptive Supply Voltage Generation
BB	Body Bias
BC	Bitcell
BL	Bitline
BLB	Bitline-Inverse
BNN	Binary Neural Network
CAM	Content-Addressable Memory
CC	Cloud Computing
CEC	Consolidated Error Correction
CES	Consumer Electronics Show
CG	Clock Gating
CIM	Compute-in-Memory
CMOS	Complementary Metal-Oxide-Semiconductor
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CTSS	Compatible Time-Sharing System
DAC	Digital-to-Analog Converter
DAL	DNN Accuracy Loss
DC	Direct Current
DCS	Dynamic Core Scaling
DCT	Discrete Cosine Transform
DFS	Dynamic Frequency Scaling
DG	Data Gating
DIBL	Drain-Induced Barrier Lowering
DIMC	Digital In-Memory Computing
DL	Deep Learning
DMA	Direct Memory Access
DNN	Deep Neural Network
DRAM	Dynamic Random-Access Memory
DRV	Data Retention Voltage
DSE	Design Space Exploration
DSL	Domain-Specific Language
DSP	Digital Signal Processor
DVFS	Dynamic Voltage and Frequency Scaling
DVS	Dynamic Voltage Scaling

EB	Exabyte
EC	Edge Computing
ECC	Error Correcting Code
ECM	Error Compensation Module
ED	Error Distance
EDA	Electronic Design Automation
EDP	Energy-Delay Product
eDRAM	embedded Dynamic Random-Access Memory
ENIAC	Electronic Numerical Integrator and Computer
EQ	Energy-Quality
ER	Error Rate
ERM	Error Recovery Module
FA	Full-Adder
FD-SOI	Fully-Depleted Silicon-on-Insulator
FF	Flip-Flop
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FLOP	Floating-Point Operation
FLOPS	Floating-Point Operations per second
FoM	Figure of Merit
FP	Floating-Point
FP16	Half-Precision Floating-Point (16-bit)
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GB	Gigabyte
GCN	Graph Convolutional Network
GDP	Gross Domestic Product
GDSII	Graphic Data System II
GenAI	Generative Artificial Intelligence
GMAC	Giga Multiply-Accumulate Operations
GMACS	Giga Multiply-Accumulate Operations per second
GOPS	Giga Operations per second
GP	General-Purpose
GPC	General-Purpose Computing
GPP	General-Purpose Processor
GPU	Graphics Processing Unit
HA	Half-Adder

HBM	High Bandwidth Memory
HDC	Hyper-Dimensional Computing
HDD	Hard Disk Drive
HDL	Hardware Description Language
HLS	High-Level Synthesis
HMC	Hybrid Memory Cube
HVT	High Threshold Voltage
IBM	International Business Machines Corporation
IC	Integrated Circuit
IEA	International Energy Agency
Ifmap	Input Feature Map
IMC	In-Memory Computing
Intel	Integrated Electronics
IoE	Internet of Everything
IoT	Internet of Things
IP	Intellectual Property
ISA	Instruction Set Architecture
J	Joule
JPEG	Joint Photographic Experts Group
KAN	Kolmogorov-Arnold Network
KB	Kilobyte
K-map	Karnaugh map
LP	Low-Power
LSB	Least Significant Bit
LUT	Look-Up Table
LV	Low-Voltage
LVT	Low Threshold Voltage
MAC	Multiply-Accumulate
MAE	Maximum Absolute Error
MAV	Maximum Absolute Value
MB	Megabyte
MCU	Microcontroller Unit
MED	Mean Error Distance
MHz	Megahertz
MIPS	Million Instructions per second
MIT	Massachusetts Institute of Technology

ML	Machine Learning
MLC	Mid-Level Cache
MRAM	Magnetoresistive Random-Access Memory
MRE	Maximum Relative Error Distance
MRED	Mean Relative Error Distance
MSB	Most Significant Bit
MSE	Mean Square Error
MSSIM	Mean Structural Similarity Index Measure
MSV	Multiple Supply Voltage
MTCMOS	Multiple Threshold-Voltage CMOS
NAND	NOT AND
NDP	Near-Data Processing
NLR	No-Local-Reuse
nm	Nanometer
NMC	Near-Memory Computing
NMED	Normalized Mean Error Distance
NMOS	N-channel Metal-Oxide-Semiconductor
NMP	Near-Memory Processing
NMSE	Normalized Mean Square Error
NN	Neural Network
NoC	Network-on-Chip
NoEB	Number of Effective Bits
NOR	NOT OR
NTC	Near-Threshold Computing
NVM	Non-Volatile Memory
OAI	OR-AND-Invert
Ofmap	Output Feature Map
OPS	Operations per second
OS	Output-Stationary
PADP	Power-Area-Delay product
PCM	Phase-Change Memory
PD	Power Domain
PDP	Power-Delay Product
PDT	Pull-down transistor
PE	Processing Element
PG	Power Gating
PIM	Processing-in-memory

PLL	Phase-Locked Loop
PMD	Personal Mobile Device
PMOS	P-channel Metal-Oxide-Semiconductor
PnR	Place-and-Route
PP	Partial Product
PPA	Power-Performance-Area
PR	Pass Rate
PRED	Probability of Relative Error Distance
PSNR	Peak Signal-to-Noise Ratio
Psum	Partial Sum
PULP	Parallel Ultra-Low-Power
PUT	Pull-up transistor
PVT	Process-Voltage-Temperature
QoS	Quality of Service
RAM	Random-Access Memory
RBL	Read Bitline
RC	Resistance-Capacitance
RCA	Ripple-Carry Adder
RED	Relative Error Distance
ReRAM	Resistive Random-Access Memory
RISC-V	Reduced Instruction Set Computer V (fifth generation)
RMSE	Root Mean Square Error
RNM	Read Noise Margin
ROI	Return On Investment
ROM	Read-Only Memory
RS	Row-Stationary
RTL	Register-Transfer Level
SA	Sense Amplifier
SCM	Standard-Cell based Memory
SDL	Significance-Driven Logic
SNM	Static Noise Margin
SNN	Spiking Neural Network
SNR	Signal-to-Noise Ratio
SoA	State-Of-The-Art
SoC	System-on-Chip
SPEF	Standard Parasitic Exchange Format
SRAM	Static Random Access Memory

SSD	Solid State Drive
SSIM	Structural Similarity Index Measure
SSM	Static Segmented Multiplier
STA	Static Timing Analysis
STM	STMicroelectronics
STT-MRAM	Spin-Transfer-Torque Magnetoresistive Random-Access Memory
SVM	Support Vector Machine
SVR	Supply Voltage Ramping
SVT	Standard Threshold Voltage
TDC	Time-to-Digital Converter
TDFG	Tensor Dataflow Graph
TDP	Thermal Design Power
TG	Transmission Gate
TOPS	Tera Operations per second
TPU	Tensor Processing Unit
TSMC	Taiwan Semiconductor Manufacturing Company
TSV	Through-Silicon Via
TWh	Terawatt-hour
ULP	Ultra-Low-Power
ULV	Ultra-Low-Voltage
VarED	Variance of Error Distance
VGND	Virtual Ground
VLSI	Very-Large-Scale Integration
VOS	Voltage Over-Scaling
VR	Virtual Reality
VTC	Voltage-Transfer Curve
V_{TH}	Threshold Voltage
WCRE	Worst Case Relative Error
WL	Wordline
WNM	Write Noise Margin
WS	Weight-Stationary
XOR	Exclusive OR

INTRODUCTION

Motivation and Scope



THIS chapter presents the motivation and scope of the thesis by tracing the transformative journey of computing systems in an era of unprecedented data generation and connectivity. At its core lies the Internet of Everything (IoE), an evolution of the Internet of Things (IoT) that unifies systems, devices, people, processes, and data streams. The rapid pace of technological change has created a critical challenge: to meet the growing demand for seamless connectivity and energy-efficient, high-performance computing across applications ranging from smart homes to scientific supercomputing. Addressing these demands reveals the limits of conventional processor-centric von Neumann architectures and motivates the exploration of new computing paradigms capable of sustaining a data-driven world.

1.1 THE NEW ERA OF ANALYTICS

Data, often described as the world's most valuable resource [1], is central to modern technology and the *Fourth Industrial Revolution* [2], where the convergence of technological and biological systems is reshaping economies and societies. Digital globalization has created opportunities such as personalized healthcare, precision agriculture, and improved governance. This new era of analytics is defined by *datafication*, in which data steers decisions and enables efficiency, innovation, and transformative intelligence [3,4]. Over recent decades, technological advancements have halved computing costs roughly every 15 months (Figure 1.1a), accelerating the growth of IoT and Artificial Intelligence (AI). These trends have fueled a global data economy worth

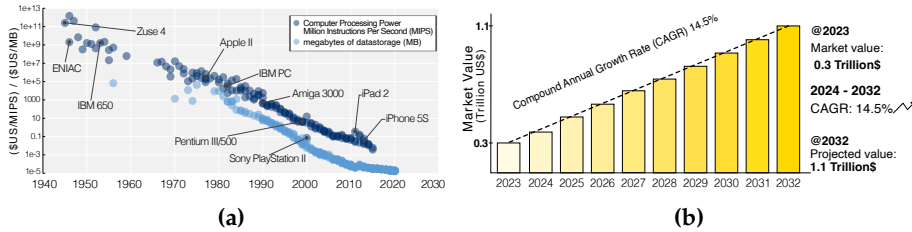


Figure 1.1: (a) Deflated costs using the Consumer Price Index (CPI), based on 2022 statistics from the U.S. Bureau of Economic Analysis, expressed in \$US/MIPS/MB (U.S. dollars per million instructions per second per megabyte) [6]; (b) global big data market revenue forecast [8].

trillions of U.S. dollars (Figure 1.1b), with data now contributing more to Gross Domestic Product (GDP) than physical trade [2, 5]. Within AI, Deep Learning (DL), particularly Convolutional Neural Network (CNN), has been central to this shift, advancing vision, language, and pattern recognition while driving unprecedented computational and memory demands. This transformation is underpinned by three key factors [6, 7]:

- a steady decline in data storage and processing costs,
- widespread deployment of IoT and digital technologies, and
- rapid advances in AI.

However, despite substantial progress, fully unlocking the potential of the data economy requires next-generation infrastructure, including advanced broadband networks and innovative data-centric computing frameworks.

1.2 THE PAST IS PROLOGUE *

The evolution of communication, connectivity, computing systems, and AI is intricately tied to *historical milestones* that have shaped today's technological landscape and set the stage for future innovations. From the telegraph to modern IoT networks, humanity's drive to bridge distances and share information has culminated in a world where billions of devices seamlessly exchange data, redefining connectivity (Figure 1.2a) [9–14].

Early milestones such as the Electronic Numerical Integrator and Computer (ENIAC) computer in 1949 [15], and the first experiments with localized networks [16] established the foundation for the advances in computational

* William Shakespeare, The Tempest, Act 2, Scene 1, "Antonio: ... what's past is prologue; what to come, In yours and my discharge."

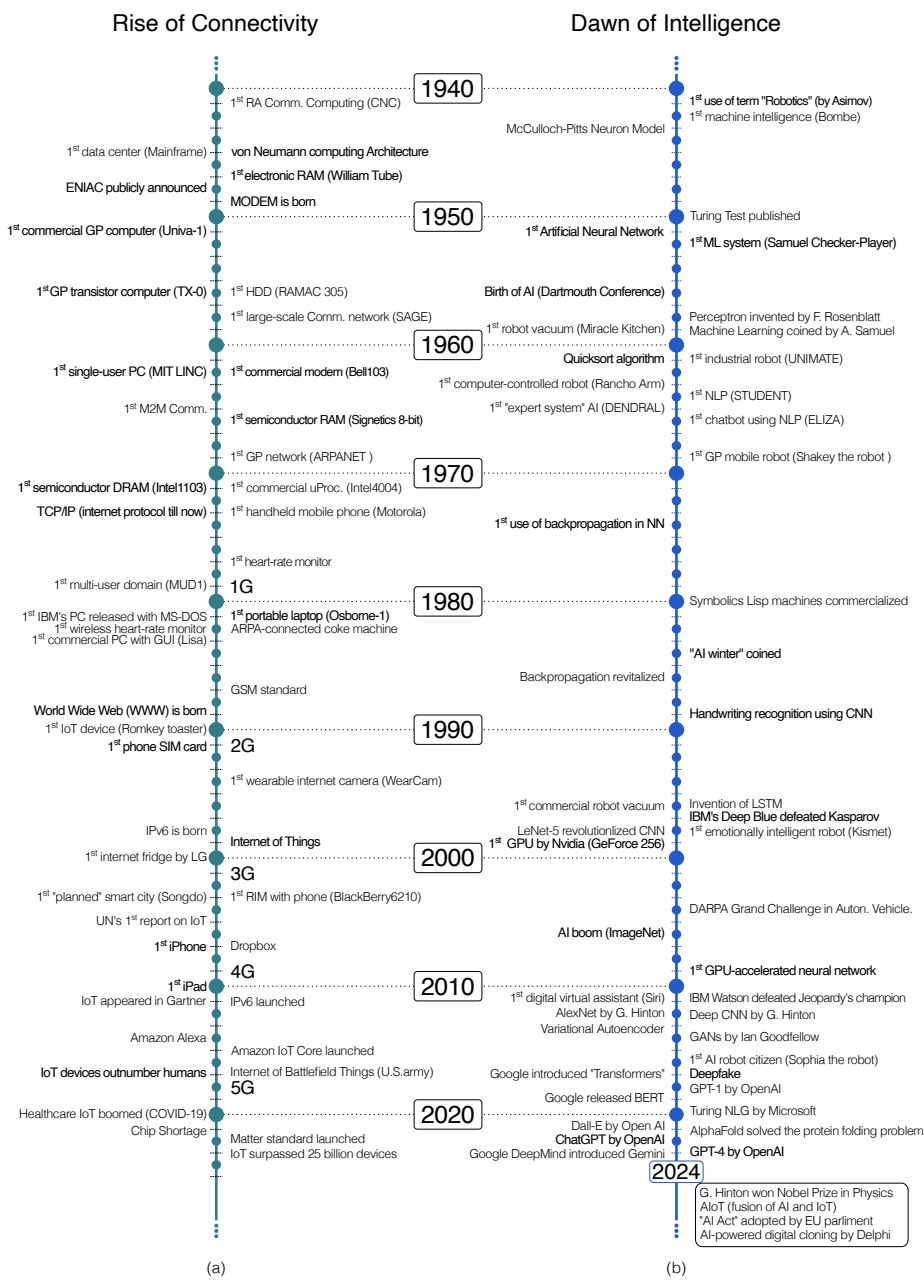


Figure 1.2: (a) Historical timeline of IoT; (b) major milestones in AI.

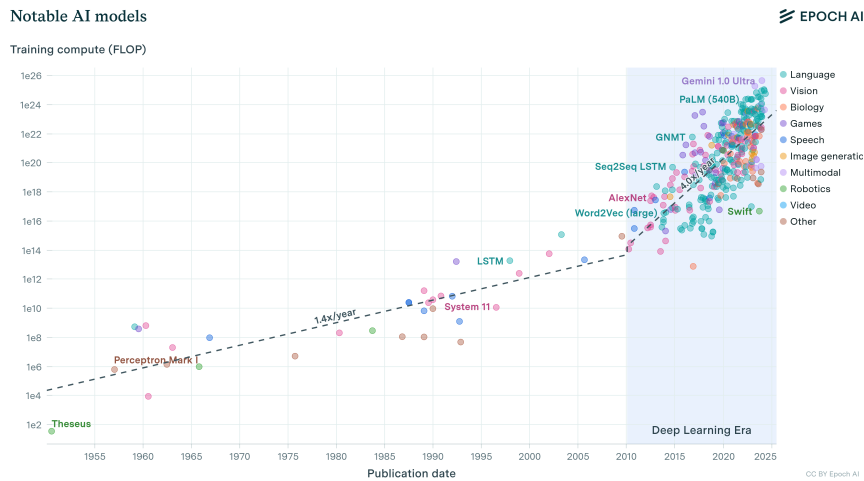


Figure 1.3: Historical ML compute trends (1955–2025), highlighting a $4\times$ annual growth since 2010 [18].

power and interconnectivity that accelerated in the decades leading up to the 1980s. The growing need for resource sharing during the 1960s and 1970s led to innovations like MIT’s Compatible Time-Sharing System (CTSS), which paved the way for personal computing systems, exemplified in the following decade by the Osborne-1 in 1981 [16]. The introduction of 1G cellular networks in the 1980s marked a transformative milestone, triggering rapid global expansion of connectivity [17].

This momentum drove major breakthroughs: IoT-connected devices outnumbered the global population in 2017, Fifth Generation Wireless Communication (5G) networks rolled out in 2018, and IoT-driven healthcare solutions played a vital role during the COVID–19 pandemic [17]. In parallel, AI progressed from early symbolic reasoning and neural models [16] to the foundational methods such as Rosenblatt’s perceptron and Samuel’s early Machine Learning (ML) algorithm (Figure 1.2b). However, the "AI Winter" of the 1970s and early 1980s slowed progress due to technological limitations [19].

Interest in AI resurged in the late 1980s, driven by advancements in semiconductor technology, laying the foundation for DL, large-scale model training, and cloud-based deployment. These advances dramatically accelerated both the scale and complexity of AI workloads. ML compute has grown exponentially since 2010 (Figure 1.3), averaging a $4\times$ increase per year [18]. This trajectory is closely linked to hardware progress and can be grouped into three eras [18]:

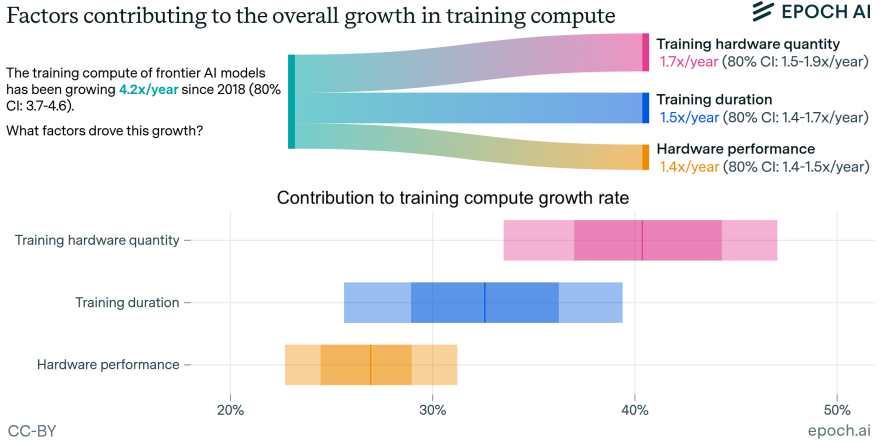


Figure 1.4: Decomposition of training compute growth into three drivers: hardware scale, compute per device, and training duration [20].

1. Pre-DL (before 2010): Compute demands increased steadily following Moore’s Law, doubling every 20 months.
2. DL (2010–2015): Compute demands doubling every 6 months.
3. Large-Scale Era (post–2015): Frontier models requiring $100\text{--}1000\times$ more compute than earlier systems in the same year.

Compute growth typically measured in Floating-Point Operations per second (FLOPS) reflects the total processing capability of hardware used for training and inference. The underlying drivers of this progression, as shown in Figure 1.4, include a $1.7\times$ annual increase in hardware scale, a $1.5\times$ rise in training duration, and a $1.4\times$ gain in per-device performance [20]. Supported by investments growing $2\text{--}3\times$ annually, these factors have enabled larger models and more powerful processing infrastructures.

1.3 CONNECTIVITY AND INTELLIGENCE OF THINGS [†]

The convergence of AI and IoT, often termed the Artificial Intelligence of Things (AIoT) [22], marks a key stage in the evolution of connected technologies. IoT provides vast, distributed data streams, while AI delivers the analytics and autonomous decision making needed to turn raw data into actionable insight. Together they create context-aware, adaptive systems that embody the

[†] "Intelligence of Things" was first introduced by Steve Koenig, Vice President of the Consumer Technology Association, in his "Tech Trends to Watch" presentation at Consumer Electronics Show (CES) 2020 [21].

Intelligence of Things, shifting the focus from isolated smart devices to intelligent networks and infrastructures.

Communication technology underpins this vision. Wireless networks must accommodate massive device counts and surging data volumes while meeting strict latency and energy requirements. Global IoT connections are projected to reach 38.8 billion by 2030, with monthly mobile traffic rising from 62 Exabyte (EB) in 2020 to over 5000 EB [23,24].

While 5G provides major gains in capacity and Quality of Service (QoS) with latencies around 1 ms, the rapid growth of AIoT workloads such as video-rich applications, high-resolution sensing, and machine-to-machine coordination is already testing its limits [25]. This is driving research toward Sixth Generation Wireless Communication (6G) networks, which aim for about 0.1 ms latency together with significantly higher energy efficiency to enable real-time, mission-critical applications such as autonomous driving and remote healthcare [24,26].

By embedding intelligence into billions of devices, AIoT equips these evolving networks to interpret, act, and adapt in real-time, making communication advances a core element of a broader intelligent system.

1.4 AI AND POWER CONSUMPTION

Gartner's 2024 analysis of technology trends (Figure 1.5) highlights the growing influence of AI-driven innovations. Among these, Generative Artificial Intelligence (GenAI) stands out, moving from theoretical exploration to large-scale deployment. Organizations increasingly use foundation models to achieve measurable returns on investment, fostering autonomous AI systems capable of dynamic interaction, decision making, and adaptation [27].

Graphics Processing Units (GPUs) first gained prominence by accelerating CNNs for large-scale training and inference, and have since evolved into the dominant computing platform for generative AI (GenAI) and other specialized accelerators. Their rapid adoption has sharply increased data-center energy demand, with facilities now scaling to million-GPU operations. Despite power-efficiency improvements achieved by industry leaders such as NVIDIA, AMD, and Intel, the overall consumption continues to rise, and GPU power use has increased by about 75 % in just two years. Although each new processor generation is more energy-efficient per operation, the escalating computational demands of large-scale AI models outpace these gains, driving up total energy per chip [28]. Advances in semiconductor technology help curb this trend. Taiwan Semiconductor Manufacturing Company (TSMC)'s 3 nm process delivers about 15 % higher performance than 5 nm while lowering power consumption by roughly 30 % [28]. Data from Intel, AMD, and NVIDIA confirm steady reductions in energy per operation (Figure 1.6a) [29].

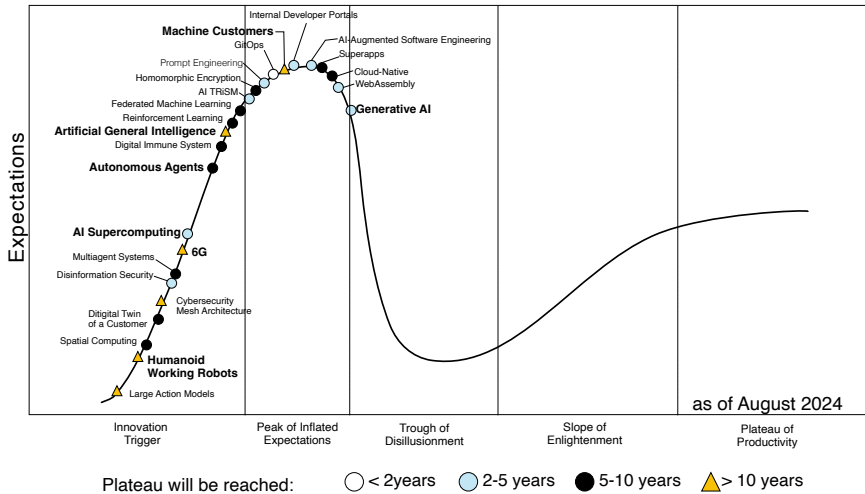


Figure 1.5: Key themes in emerging technology trends as of August 2024 [27].

Despite these improvements, AI's escalating power demands are expected to drive a new wave of data-center construction with integrated power plants [28]. Projections underscore the scale of this challenge: Morgan Stanley estimates AI power demand will grow about 70 % annually, reaching Spain's 2022 electricity use by 2027 [30]. Wells Fargo forecasts an increase from roughly 8 Terawatt-hour (TWh) in 2024 to over 650 TWh by 2030, with model training alone requiring about 400 TWh [28]. The International Energy Agency (IEA) projects that AI, data centers, and cryptocurrency together could draw up to 1050 TWh of electricity by 2026 [31].

Figure 1.6b illustrates this continuing upward trend, which raises both operational costs and environmental concerns. For example, Google's carbon emissions rose 48 % over five years, and Microsoft's increased by 30 % since 2020 [32,33]. A substantial portion of AI energy is spent on data movement, with up to 90 % of training energy consumed by memory access [34]. Together, these trends underscore the urgent need to rethink and redesign computing systems to meet AI's rapidly growing energy demands.

1.5 EMERGING COMPUTING TRENDS

The rapid growth of network-connected devices has exposed the limits of traditional centralized architectures. Rising data volumes, strict latency requirements, and bandwidth constraints, amplified by IoT proliferation and

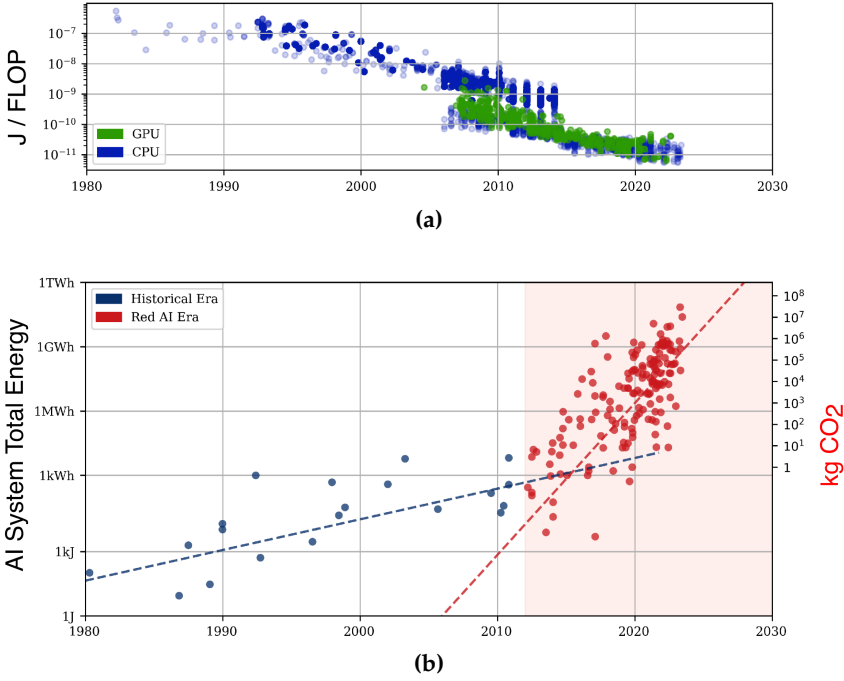


Figure 1.6: (a) Estimated energy (J/FLOP) based on thermal design power and throughput (FLOPS); (b) energy consumption (left axis) and corresponding carbon emissions (right axis) in the United States from training published AI models. ‘AI System Total Energy’ represents a computed metric accounting for hardware efficiency improvements [29].

embedded AI, demand alternative approaches [35]. Cloud Computing (CC) provides scalable resources, efficient data storage, and strong computational capabilities, but its centralized model struggles with real-time, context-aware workloads and the exponential growth of data. Large-scale data centers also pose significant energy and sustainability concerns [5,36].

Edge Computing (EC) addresses these challenges by processing data closer to its sources, reducing latency, bandwidth use, and network congestion while improving scalability, efficiency, security, and privacy [36]. Complementing CC, EC supports latency-sensitive and resource-constrained applications by enabling local decision making and improving energy efficiency, thereby enhancing overall user experience.

A central driver of this evolution is *edge AI*, performing inference and, in some cases, on-device learning where data is generated, including sensors, smartphones, drones, and autonomous vehicles [37]. Combining low-latency

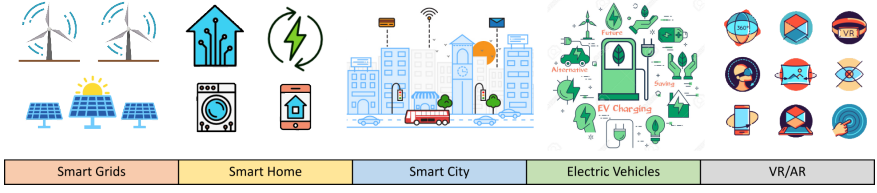


Figure 1.7: AIoT implementation within the EC ecosystem [35].

and privacy benefits of EC with ML capability, edge AI enables real-time analytics, predictive maintenance, and personalized healthcare while meeting strict power budgets and avoiding reliance on cloud infrastructure.

The broad EC ecosystem shown in Figure 1.7 underscores key elements needed to ensure efficient operation [38]:

- real-time support with high reliability and low latency,
- resource optimization across computing, networking, and energy use,
- scalable architectures adaptable to diverse workloads through automated edge IoT orchestration.

Despite their advantages, EC faces strict limits in compute capacity, device heterogeneity, storage, and energy budgets. The *memory wall*, a long-recognized widening performance gap between memory and processing [39, 40], is particularly acute in edge devices where resources are tighter. The rapid growth of AI workloads exacerbates this challenge as modern applications demand far higher memory bandwidth and computational performance, as shown in Figures 1.8a and 1.8b, far outpacing advances in memory and interconnect technologies [41].

Studies show that memory-intensive workloads, especially AI-driven ones, often fail to use available bandwidth efficiently [42]. Energy analyses indicate that even State-Of-The-Art (SoA) processors and accelerators can waste up to 60 % of total time and energy waiting for data or moving it between memory and processing units [43]. Investigations of Google’s edge Tensor Processing Units (TPUs) highlight two critical inefficiencies [44,45]:

- under-utilization of processing elements due to limited memory bandwidth, inefficient dataflows, and rigid data architectures,
- high energy cost of off-chip to on-chip data movement, with over 50 % of total energy spent on buffer caching.

These inefficiencies call for a paradigm shift from von Neumann architectures to memory-integrated computing models. Conventional systems, with physically separated compute and memory, incur frequent data transfers and significant energy overheads. In contrast, beyond-von-Neumann architectures

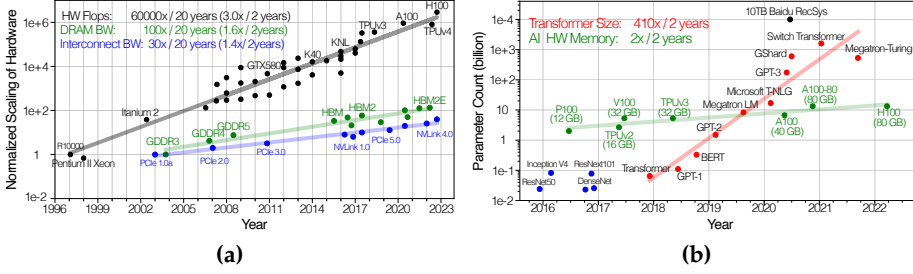


Figure 1.8: (a) Bandwidth scaling of interconnect and memory technologies alongside peak FLOPS [41]; (b) evolution of SoA model parameter counts and AI accelerator memory capacity (green dots). Transformer model parameters have increased exponentially by $410\times$ every two years, whereas single-GPU memory has scaled only $2\times$ in the same period [41].

co-locate storage and processing, reducing data movement, lowering latency, and improving energy efficiency.

This architectural evolution aligns with the broader vision of the *Intelligence of Things*, where edge devices act as distributed, intelligent components. By overcoming the inefficiencies of traditional designs, EC can meet the computational demands of modern AIoT applications and enable real-time analytics, enhanced security, and scalable networks.

Among emerging solutions, *Near-Data Processing (NDP)* stands out [46], including *Near-Memory Computing (NMC)* and *In-Memory Computing (IMC)* in both digital and analog forms [47]. By integrating memory and processing, these approaches minimize data movement, reduce energy use, and limit intermediate storage [34]. Eliminating constant data transfers and enabling localized computation make NDP particularly effective for data-intensive workloads [47, 48], positioning memory-centric computing as a key building block for future energy-efficient architectures.

1.6 THESIS OUTLINE

The Introduction is organized into seven chapters that trace the path from the initial motivation to the development of novel memory-centric architectures for edge AI. The presentation follows a **bottom-up** optimization strategy, progressing from device- and circuit-level innovations to micro-architectural, algorithmic, and system-level solutions. Figure 1.9 summarizes the main contributions and shows how each paper and chapter aligns with the layers of hardware abstraction.

Chapter 2 examines power consumption, identifying major sources of dissipation and surveying optimization techniques across abstraction layers. The discussion highlights key tradeoffs between flexibility, energy efficiency, and scalability that shape the design choices developed later in the thesis.

Chapter 3 narrows the focus to memory, reviewing hierarchy principles and examining the design and implementation of on-chip memory variants. This analysis highlights both the opportunities and the limitations that motivate the architectural innovations developed in the thesis.

Chapter 4 presents the architecture and implementation of a proposed synthesizable distributed on-chip memory and its generation flow, which provide modularity, scalability, and parallel access within a synthesizable framework. Silicon implementation and simulation results show how conventional high-density memory cells can be adapted into a flexible, energy-efficient Intellectual Property (IP) block.

Chapter 5 explores Approximate Computing (AC) as a complementary path to efficiency. It reviews key methodologies, error metrics, and multiplier design strategies, establishing the context and motivation for the thesis's contribution on scalable approximate multipliers with operand-aware error management.

Chapter 6 broadens the perspective to memory-centric computing, classifying in-memory and near-memory paradigms, reviewing enabling technologies, and analyzing representative implementations. System-level tradeoffs and integration challenges are examined to connect the thesis's logic- and architecture-level contributions with the wider computing landscape.

Chapter 7 concludes the thesis by synthesizing its contributions and positioning them within the broader trajectory of data-centric edge AI hardware research while outlining future directions such as compiler-hardware co-optimization and deployment in heterogeneous systems.

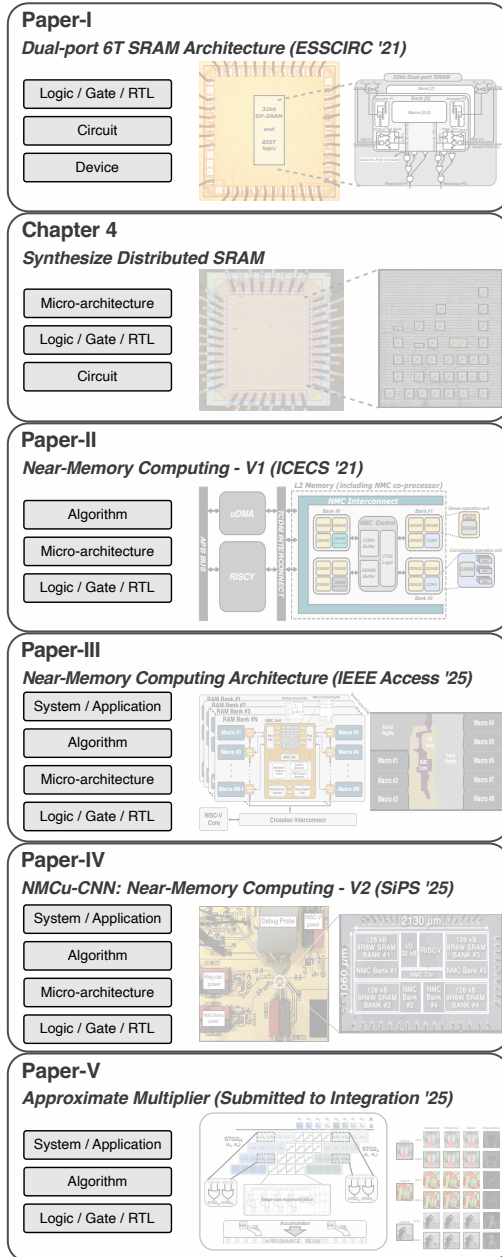


Figure 1.9: Overview of the thesis contributions mapped to the layers of hardware abstraction.

2

Power Optimization: A Strategic Perspective

 HIS chapter provides a comprehensive exploration of power consumption in digital Integrated Circuits (ICs), examining the fundamental sources of power usage and surveying techniques to optimize power and realize energy-efficient design across abstraction layers, with particular attention to Ultra-Low-Power (ULP) design for energy-constrained IoT and always-on systems.

2.1 POWER CONSUMPTION IN DIGITAL SYSTEMS

Complementary Metal-Oxide-Semiconductor (CMOS) technology underpins nearly all modern digital ICs, from microprocessors to IoT and AI accelerators, increasingly integrated as System-on-Chip (SoC) platforms. Decades of scaling and innovation have steadily improved speed and energy efficiency, yet power consumption remains a key challenge as device dimensions and supply voltages approach physical limits [49]. Understanding these power characteristics is essential for effective power optimization and adopting energy-efficient design strategies.

The total power consumption in digital systems consists of two components, dynamic (P_{dyn}) and static (P_{stat}), expressed as [50]:

$$P_{tot} = P_{dyn} + P_{stat}. \quad (2.1)$$

Dynamic power, consumed during active operation, includes switching (P_{switch}) and short-circuit (P_{sc}) components:

$$P_{dyn} = P_{switch} + P_{sc}. \quad (2.2)$$

P_{sc} arises from a temporary current flow between supply and ground during logic transitions, when both P-channel Metal-Oxide-Semiconductor (PMOS)

and N-channel Metal-Oxide-Semiconductor (NMOS) transistors conduct simultaneously. With continued CMOS scaling, its impact has been significantly reduced [50, 51].

The dominant component of P_{dyn} , namely P_{switch} , arises from charging and discharging load capacitance during logic transitions, and is given by [52]:

$$P_{switch} = f_{clk} \times C_{eff} \times V_{DD}^2 = \alpha \times f_{clk} \times C_L \times V_{DD}^2, \quad (2.3)$$

where f_{clk} is the clock frequency, V_{DD} the supply voltage, C_L the total load capacitance, and α the activity factor ($0 < \alpha < 1$), accounting for the fact that not all gates switch each clock cycle. Dynamic power depends quadratically on V_{DD} ($P_{dyn} \propto V_{DD}^2$), making voltage scaling one of the most effective power-reduction strategies [50, 52, 53]. In contrast, when the system is idle, leakage current (I_{Leak}) through nominally off transistors results in P_{stat} , which scales linearly with the supply voltage ($P_{stat} \propto V_{DD}$):

$$P_{stat} = I_{Leak} \times V_{DD}. \quad (2.4)$$

I_{Leak} is governed by factors such as transistor Threshold Voltage (V_{TH}), operating temperature, and device dimensions. Although I_{Leak} is largely independent of V_{DD} in first order, secondary effects such as Drain-Induced Barrier Lowering (DIBL) and channel-length modulation introduce a mild voltage dependence. As technology scales to ever smaller nodes, controlling I_{Leak} has become increasingly difficult, demanding advanced circuit techniques to limit static losses and maintain energy efficiency [52, 54].

2.2 POWER OPTIMIZATION FRAMEWORK

Power and energy optimization is a critical aspect of IC design, spanning multiple stages and addressing both P_{dyn} and P_{stat} . Some techniques can be applied during design, either based on predictable activity patterns or independent of workload variations. However, real-world systems frequently experience dynamic fluctuations in performance demand. These fluctuations necessitate adaptive runtime optimizations. For example, during the idle (standby) state, P_{dyn} is reduced to near zero, leaving P_{stat} as the dominant factor [54].

The optimization process involves tuning device parameters such as transistor size, V_{TH} and supply voltage, as well as making discrete design choices in logic style, circuit topology, and microarchitecture. Since optimizing all parameters simultaneously is computationally infeasible, abstraction and hierarchy are employed. Abstraction confines optimizations to specific design layers, such as microarchitecture-level instruction sets or circuit-level, while

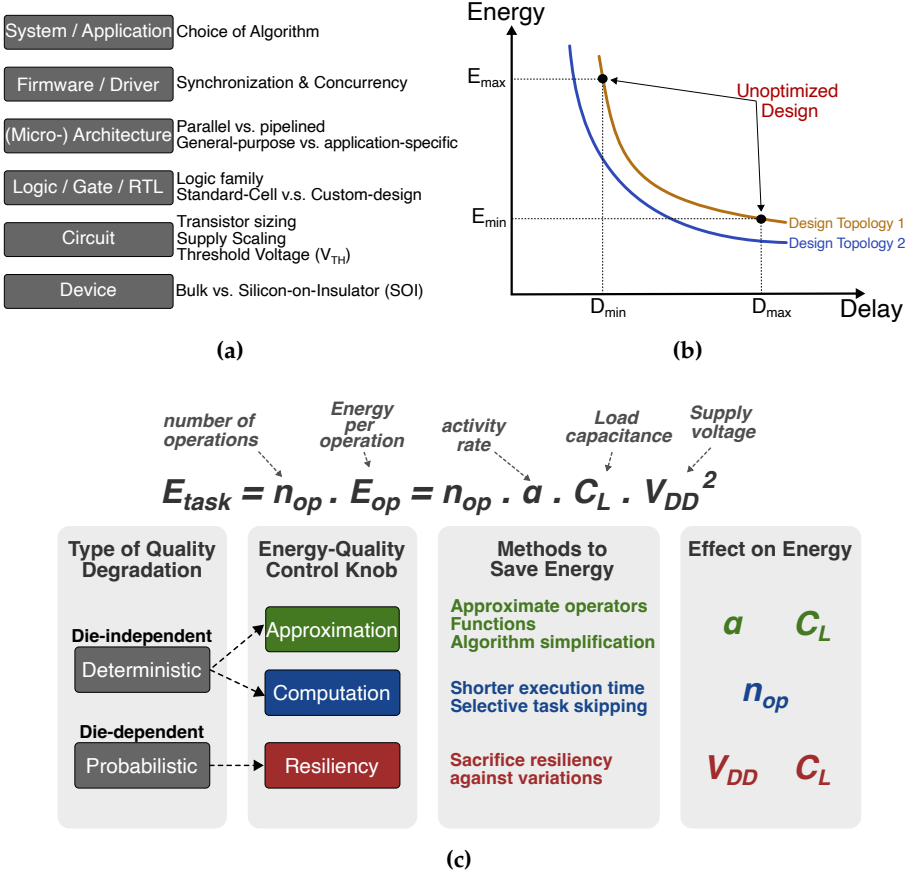


Figure 2.1: (a) Design abstraction stack [54]; (b) energy-time trade-off curves [56]; (c) taxonomy of methods trading energy for quality [55].

hierarchy enables larger systems to be built from modular components (Figure 2.1a) [54]. These layers are interdependent, requiring a meet-in-the-middle approach where high-level specifications define system goals and low-level constraints determine feasible implementations [54,55]. Analytical methods such as energy-delay sensitivity analysis are essential for guiding design choices, as they quantify how parameter adjustments affect both energy and delay. The energy-delay curve (Figure 2.1b) illustrates the Pareto-optimal frontier, where improving one metric inevitably worsens the other. These tradeoffs define the fundamental design limits of a system, which can be expressed through optimization objectives [56]:

- maximize performance within a power budget, constrained by a maximum energy E_{max} ,
- minimize energy consumption while meeting throughput requirements, constrained by a maximum delay D_{max} .

Beyond the energy-delay domain, further energy reduction can be achieved by exploiting the tradeoff between energy consumption and computational quality. As illustrated in Figure 2.1c, Energy-Quality (EQ) knobs introduce a separate optimization dimension, grouped into three main categories:

- approximation, where algorithmic or circuit-level simplifications reduce power at the cost of bounded accuracy,
- execution-time reduction, where early termination or loop perforation shortens computation and enables voltage scaling,
- resiliency tuning, where design margins are relaxed to exploit tolerance to variations in Process-Voltage-Temperature (PVT).

These approaches are typically combined with error detection and control, making them complementary to classical energy-delay optimization [55]. Energy-efficient, high-performance ICs are ultimately realized through the coordinated use of design-time and runtime optimizations. By leveraging abstraction, hierarchy, and multi-layered strategies, designers can balance energy, performance, and quality to meet the requirements of modern workloads [54]. Studies show that architecture-circuit co-optimization can significantly reduce energy consumption, depending on application goals [53].

2.2.1 ARCHITECTURE, ALGORITHMS, AND SYSTEMS OPTIMIZATION

Power optimization at the architecture, algorithm, and system levels provides substantial energy savings, often exceeding the impact of circuit-level techniques [54]. Higher-level approaches target inefficiencies across entire workloads, delivering orders-of-magnitude improvements by reducing supply voltage, load capacitance, and switching activity. These strategies influence both P_{dyn} and P_{stat} , extending optimization beyond device-level refinements.

The design implications of these approaches are shown in Figure 2.2a, which presents common hardware architectural options, and further detailed in Figure 2.2b, which maps these options onto the flexibility–performance–power–area and functional-diversity design space [54,57]. Together, they show that architectural optimization is not only about maximizing performance and energy efficiency but also about balancing flexibility and area overhead.

The hardware platform landscape can be viewed as a spectrum where *energy efficiency* gradually decreases and *flexibility* increases when moving from dedicated accelerators toward general-purpose processors. At the high-efficiency, low-flexibility end are *dedicated Application-Specific Integrated*

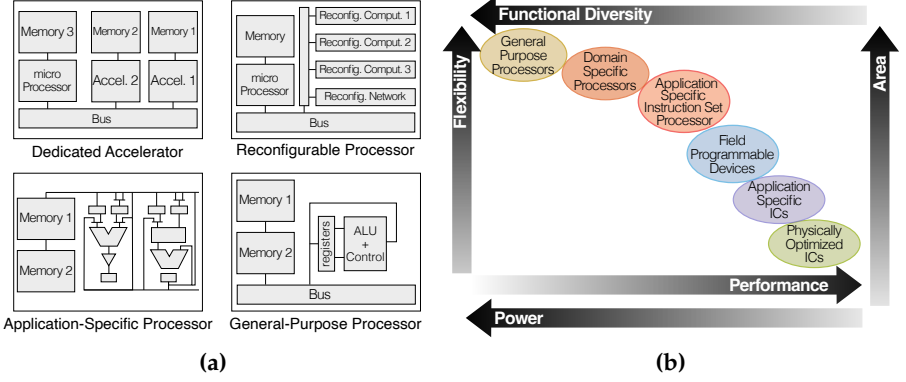


Figure 2.2: (a) Spectrum of hardware architectural platforms [54]; (b) corresponding tradeoffs on the flexibility-performance-power-area and functional-diversity planes [57].

Circuits (ASICs), which achieve exceptional energy efficiency for narrowly defined workloads but sacrifice programmability and incur significant area overhead. To introduce flexibility while retaining energy efficiency close to that of dedicated accelerators, *reconfigurable hardware architectures* such as Field-Programmable Gate Arrays (FPGAs) can be employed. These devices can be rewired to implement new functions or complete applications, offering a balanced tradeoff between efficiency, flexibility, and cost. Further along the flexibility axis are *Application-Specific Instruction Processors (ASIPs)* and *Digital Signal Processors (DSPs)*, which add custom instructions and optimized datapaths to deliver domain-specific efficiency while remaining more programmable than pure accelerators. At the opposite extreme are *General-Purpose Processors (GPPs)*, which deliver maximum flexibility through broad programmability and adaptability across diverse workloads but offer the lowest energy efficiency.

Beyond the selection of a hardware platform, several *architectural strategies* further improve energy efficiency [54,58]:

- **Parallelism** distributes input samples across multiple processing branches, reducing the required clock frequency and enabling supply-voltage scaling for significant power savings [56,58]. Area and synchronization overhead, along with workload-dependent applicability, are the main tradeoffs [54].
- **Pipelining** partitions long combinational paths into sequential stages separated by registers, allowing concurrent computation and reducing energy per operation compared with low-latency designs of similar

throughput [52,56,59]. Extra registers and switching activity introduce overhead and latency, effects that become more pronounced at low voltages.

- **Time-multiplexing**, or resource sharing, reuses hardware across multiple processing blocks over time, cutting area in proportion to the number of multiplexed branches and mitigating leakage in low-frequency or Low-Voltage (LV) designs [54,60]. Energy savings must be balanced against limits on parallel throughput.

Overall, effective architectural optimization requires a careful balance of concurrency, flexibility, and energy efficiency. Complementary techniques such as loop unfolding, retiming, and algebraic transformations including commutativity and distributivity can improve parallelism and lower energy. However, inefficiencies often result from over-dimensioned or overly generic designs, and the programmability that improves reusability typically reduces efficiency [54,57]. Thus, design choices should carefully navigate the tradeoffs illustrated in Figures 2.2a and 2.2b.

2.2.2 CIRCUIT LEVEL OPTIMIZATION

Circuit-level optimization is essential to improve system performance and energy efficiency, typically offering moderate savings [60]. Key techniques include [52,54]:

- **Logic Family Selection** directly impacts energy and performance. Static circuits are preferred in energy-sensitive or low-performance applications due to their low switching activity, robustness, and scalability, while dynamic circuits are suited for high-performance designs. Common families include Standard CMOS, Pseudo-NMOS, Pass Transistor, and Transmission Gates (TGs) Logic [52,58].
- **Gate Sizing** Adjusts drive strength to balance performance and power within constraints and improves manufacturability, and reliability [61].
- **Multiple Supply Voltages (MSVs)** are implemented through separate voltage islands, with higher levels assigned to performance-critical blocks and lower levels to less demanding functions. Empirical studies have shown that employing two discrete voltage levels provides an effective tradeoff between design complexity and power efficiency [54,62].
- **Multiple Threshold-Voltage CMOS (MTCMOS)** employs low- V_{TH} devices on critical paths to enhance performance, high- V_{TH} devices on non-critical paths to suppress I_{Leak} , and standard- V_{TH} devices for clock and control logic, achieving a balanced tradeoff between energy efficiency and performance (Figure 2.3a) [63].

- **Transistor Stacking** relies on the principle that two series-connected transistors exhibit lower I_{Leak} than a single off-device, owing to reduced gate-to-source voltage, reverse body biasing, and DIBL effects. This leakage reduction comes at the expense of reduced on-current, with optimal efficiency typically achieved using two stacked NMOS transistors, although the results depend on input patterns [52, 54, 63].
- **Power Gating (PG)** disconnects modules from the supply rails using header or footer sleep transistors, or maintains a retention voltage when data integrity need to be preserved (Figure 2.3b). This technique, however, introduces challenges such as start-up delay and wake-up energy, since parasitic capacitances must be recharged. To reduce the area and power overhead, the off-time must be sufficiently long. Moreover, power gates must be carefully sized to supply the required peak current during start-up while limiting I_{Leak} in idle states [54, 64, 65].
- **Adaptive Body Biasing (ABB)** dynamically adjusts transistor V_{TH} to control overall power consumption by balancing performance and leakage (Figure 2.3c). It also compensates for process variations, enhances reliability, and mitigates device aging, but requires triple-well technology and additional bias supplies [63, 66].
- **Supply Voltage Ramping (SVR)** lowers the supply either to zero (non-retentive) or to the Data Retention Voltage (DRV) level. It is highly effective in reducing power but requires on-chip voltage regulators, introduces longer wake-up times [53, 64].

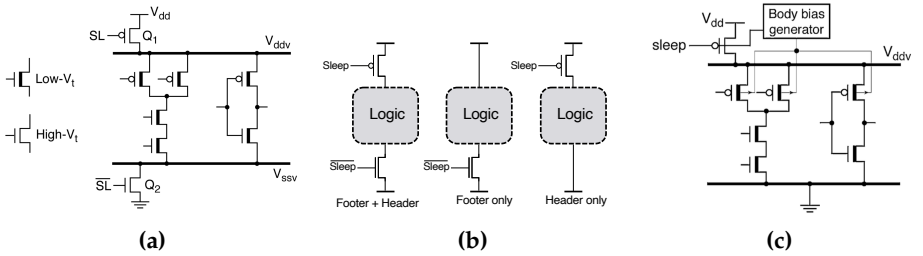


Figure 2.3: (a) MTCMOS PG [67]; (b) PG configurations: footer (NMOS) sleep transistors are more area-efficient, header (PMOS) sleep transistors are less area-efficient, and combining both improves leakage reduction [54]; (c) PG with Body Bias (BB) control [67].

Additionally, minimizing the effective switching frequency (f_{eff}) is essential for reducing P_{dyn} . This can be effectively achieved using techniques such as **Clock Gating (CG)** and **Data Gating (DG)** [54]. The clock distribution network

often consumes a large share of total power in CMOS circuits due to its high switching activity and large capacitive load. CG reduces f_{eff} by disabling the clock in idle modules (Figure 2.4a), thereby avoiding unnecessary transitions and spurious activity, and can be applied at multiple levels [65,68]:

- *Module*: disables the clock for specific functional modules,
- *Clock tree*: turns off branches of the clock distribution network,
- *Clock source*: turns off the on-chip clock source,
- *Full shutdown*: leaves only a minimal wake-up circuit active.

CG effectively lowers standby P_{dyn} but also introduces challenges such as clock skew, load variations, and increased design complexity [68]. DG, in contrast, targets non-clock signal paths by suppressing unnecessary toggles. For example, computational units such as adders, as illustrated in Figure 2.4b, or multipliers can be gated when inactive. This approach further reduces P_{dyn} but requires additional gating cells and may introduce extra timing delays on critical paths [69,70].

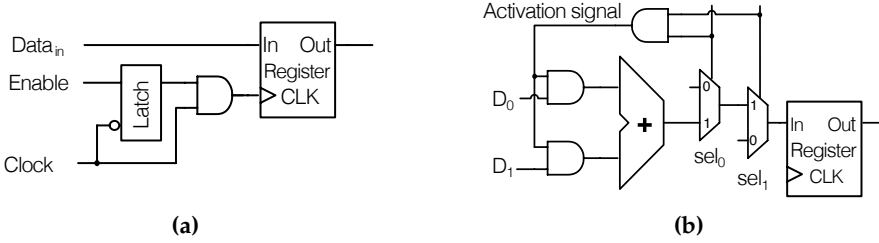


Figure 2.4: (a) Clock Gating (CG); (b) Data Gating (DG) [71].

2.2.3 RUNTIME OPTIMIZATION

In ULP systems, where workloads and supply conditions vary widely, runtime optimization is essential for maintaining energy efficiency under dynamic operating conditions. By adapting supply voltage, body bias, and operating frequency to real-time activity levels, process variations, and environmental factors, runtime methods enable optimal performance and power management while addressing thermal concerns, device aging, and reliability limitations that static design-time techniques cannot mitigate.

Dynamic Voltage Scaling (DVS) and **Dynamic Frequency Scaling (DFS)** are the most widely used runtime power management techniques (Figure 2.5a). When combined as **Dynamic Voltage and Frequency Scaling (DVFS)**, they provide quadratic power savings by lowering both supply voltage and

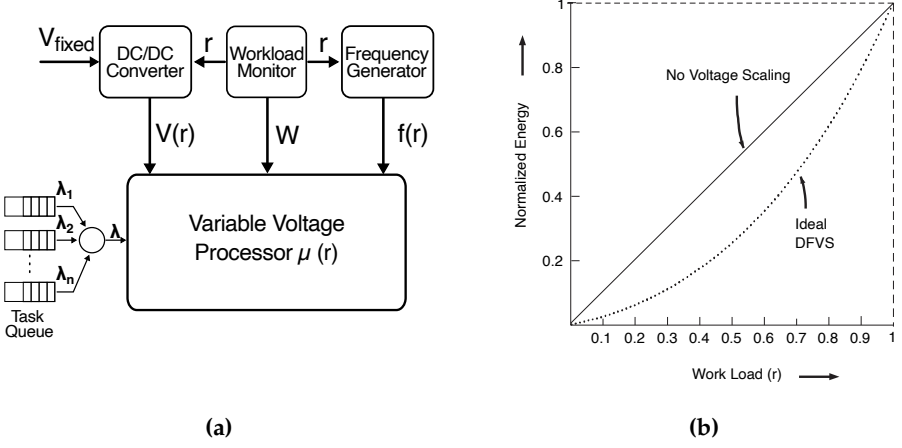


Figure 2.5: (a) DVFS control model, where utilization is monitored to estimate workload w and processing rate r , and the next-cycle frequency $f(r)$ and supply voltage $V(r)$ are set accordingly [72]; (b) energy versus workload under DVFS, with power dissipation decreasing approximately linearly (solid line) [72].

operating frequency, significantly reducing energy per operation without hardware modifications (Figure 2.5b) [59, 63]. However, the effectiveness of DVFS diminishes in advanced technology nodes where voltage scaling is constrained.

Dynamic Core Scaling (DCS) addresses this limitation by dynamically adapting datapath resources to run applications at reduced speeds, thereby lowering energy consumption without relying solely on voltage scaling. When used together with DVFS, DCS achieves higher energy efficiency across a broader range of operating scenarios [73].

Power Domains (PDs) further extend flexibility by enabling independent control of different chip regions, typically in combination with multiple supply voltages. Performance-critical modules can operate at full supply levels while less critical blocks run at lower voltages or enter standby states. Although PDs improve overall energy efficiency, they introduce overhead and complexity from generating multiple voltage rails and require careful level-shifter design to preserve signal integrity across power boundaries [62].

By integrating DVFS, DCS, and PDs, modern SoCs achieve high energy efficiency, adaptability, and performance scalability, ensuring reliable operation under diverse workloads and environmental conditions.

2.3 THESIS CONTRIBUTION

This thesis applies power optimization techniques across multiple abstraction levels. **Paper I**, focused on a standalone Static Random Access Memory (SRAM) design, employs several circuit-level methods, including body biasing, gate sizing, multiple supply voltages, transistor stacking, and PG.

Chapter 4, on Synthesizable Distributed SRAM, extends this perspective by examining power efficiency within a synthesizable memory framework. Standard circuit-level optimizations are integrated in full-custom cells such as the sense amplifier, tri-state buffer, and memory Bitcell (BC). In addition, the architectural organization places each memory array close to the logic blocks that use it most, which reduces parasitic capacitance and resistance in interconnects and thereby improves energy efficiency.

Papers II–IV broaden the scope by combining circuit-level techniques with architectural optimizations, targeting efficiency in larger system contexts. Within the strategic energy-flexibility-performance design space, the proposed NMC framework maximizes the complementary benefits of two hardware paradigms: highly energy-efficient dedicated accelerators and highly flexible programmable Microcontroller Units (MCUs). Finally, **Paper V** on approximate multiplier design, addresses power optimization at the algorithmic level, highlighting how error-aware arithmetic can reduce energy costs while preserving application-level accuracy.

3

Memory and Efficiency



HIS chapter delves into the organization of memory systems, examining their components, technical intricacies, and the critical role they play in enabling highly efficient processing paradigms.

3.1 MEMORY SYSTEM

The foundational architects of modern computing, A. W. Burks, H. H. Goldstine, and J. von Neumann, envisioned systems with *indefinitely large memory in which any word would be immediately accessible* [74]. Recognizing practical limits, they introduced a *hierarchy of memories* in which each level offers greater capacity at slower access. This concept, grounded in a cost-performance tradeoff, shaped the von Neumann memory hierarchy [75] and has maintained a balance of speed, capacity, and efficiency across decades of technological progress [76].

The organization of memory hierarchies is further guided by the *principle of locality*. Programs typically operate on a small working set of data, exhibiting temporal locality (reusing the same data within short intervals) and spatial locality (accessing data near recently used addresses). Because faster memories remain more expensive in any given technology, hierarchies are structured to balance cost and performance, preserving the low cost per byte of the cheapest memory while approaching the speed of the fastest available [76]. Figure 3.1 illustrates this conceptual hierarchy [76–78]:

- Registers: smallest and fastest Central Processing Unit (CPU) memory, holding operands during computation to exploit temporal locality.

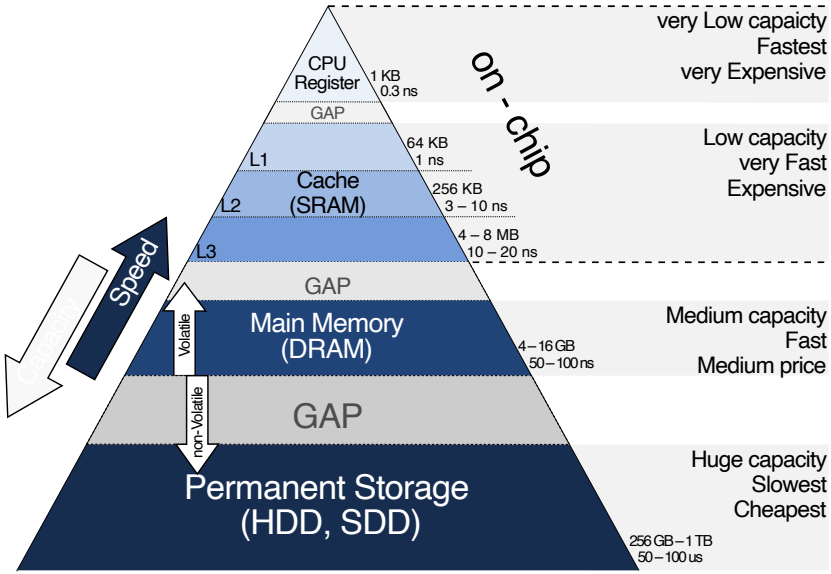


Figure 3.1: Conceptual memory hierarchy pyramid with representative capacities/latencies for Personal Mobile Device (PMD) (e.g., smartphones, tablets) and laptops/desktops, adapted from [76].

- Caches: high-speed buffers for frequently used data. Multi-level caches (L1 to L3) are typically employed to balance speed, size, and energy efficiency, though adding levels can bring diminishing returns.
- Main Memory: stores active data and instructions with far larger capacity than caches but lower speed.
- Secondary Storage: non-volatile long-term storage with the largest capacity and lowest cost.

At the technology level, Static Random Access Memory (SRAM) and Dynamic Random-Access Memory (DRAM) underpin these hierarchy levels with complementary roles. DRAM provides high storage density but requires periodic refresh to preserve data, which makes it the dominant technology for main memory [76]. In contrast, SRAM, used for on-chip caches, can retain data as long as power is supplied and offers the speed and reliability needed for rapid access [76,79]. As processors have incorporated multi-MB caches, particularly large second- and third-level caches, power consumption has become a major design concern. Caches can account for up to 50 % of total chip power [76].

The practical application of the memory hierarchy concept and the principle of locality is clearly reflected in commercial systems. Figure 3.2 shows the

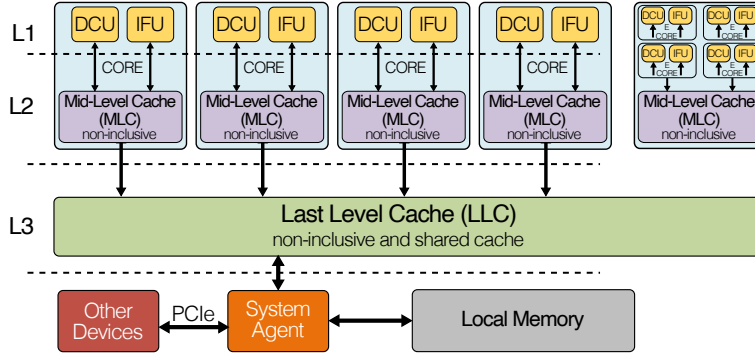


Figure 3.2: Intel Smart Cache technology in 13th–14th generation Core processors (2024) [80].

memory hierarchy of Intel’s 2024 high-end Core processors, which embody this well-established structure [76, 80].

Over the past decades, the widening gap between processor speeds and memory access times (Figure 3.3) has highlighted the critical role of memory hierarchies [76]. Processor improvements have steadily increased the rate of memory requests (inversely related to latency), while DRAM latency has advanced far more slowly. Designing effective hierarchies is further complicated by factors such as multiple banks, memory channels, and non-uniform access across processors. Although this performance gap expanded sharply for many years, the slowdown in single-core performance scaling has moderated its growth in recent generations [76].

To bridge this gap, caches have become a key element of modern memory hierarchy design. By storing frequently accessed data close to the processor, they achieve high hit rates and reduce average access latency so that most requests are served without reaching slower lower levels of the memory hierarchy. This efficiency often depends on the *inclusion property*, which ensures that each lower level contains a superset of the data stored in higher levels, simplifying coherence management and guaranteeing that main memory (or secondary storage in virtual memory) remains a complete repository of program data [76, 81]. Acting as non-user-addressable buffers, caches also reduce overall bandwidth demand, and their performance is typically evaluated by the miss rate, the fraction of accesses not served by the cache. Each miss incurs latency and energy penalties because data must be fetched from slower, lower levels [77, 78]. Beyond misses, bandwidth bottlenecks inside the hierarchy can create congestion and longer critical-path delays, limiting throughput. Studies show that increasing cache hierarchy bandwidth often

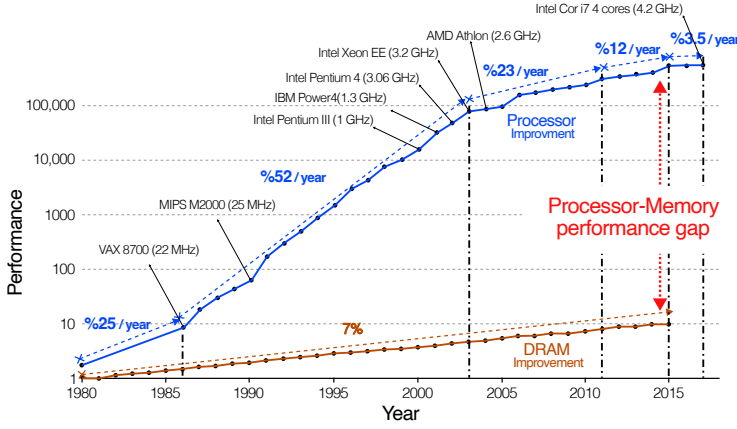


Figure 3.3: Trend of the growing processor-memory gap since 1980, measured as the difference between DRAM access latency and the single-core memory request interval [76].

provides greater performance gains than relying only on advanced off-chip memory technologies such as High Bandwidth Memory (HBM) DRAM [82]. Addressing bandwidth constraints across all hierarchy levels is therefore essential for balanced and efficient system performance.

On-chip memories have therefore become central to system performance across platforms ranging from high-performance servers to energy-constrained edge devices. The emphasis on energy efficiency in applications such as smart sensors and embedded systems drives the need for innovative memory solutions. SRAM remains the dominant on-chip technology, but its vulnerability in the Low-Voltage (LV) operating region has motivated the development of alternative solutions such as synthesizable Standard-Cell based Memories (SCMs), which offer robust operation at reduced voltages albeit with lower density [83–85]. Hybrid architectures combining SRAM and SCM balance density and reliability, while embedded Dynamic Random-Access Memory (eDRAM) has re-emerged as a candidate for high-density on-chip storage despite challenges in power and scalability.

The following sections examine SRAM, SCMs, hybrid architectures, and emerging memory options in detail, emphasizing their roles in enabling energy-efficient and reliable systems.

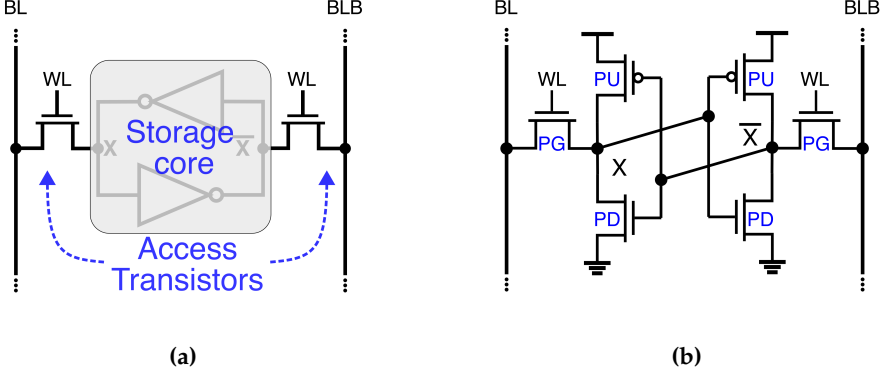


Figure 3.4: Schematics of: (a) simplified reference SRAM BC with two cross-coupled inverters and access transistors; (b) conventional 6T SRAM BC highlighting Pull-up transistors (PUTs), Pull-down transistors (PDTs), and pass transistors.

3.2 SRAM

SRAM is the dominant on-chip memory due to fast access, compatibility with mainstream CMOS processes, and strong density scaling [51]. However, advancing SRAM design remains challenging: unlike general-purpose computational logic, it must deliver very high density, low leakage, high-performance, and reliable data retention. Additionally, SRAM often occupies a substantial portion of the chip area, making power consumption and design choices critical factors that directly influence system-level performance and efficiency [86,87]. This section reviews architectural foundations, stability, multi-port realizations, power optimization, and key challenges with mitigation strategies.

3.2.1 Six-Transistor (6T) SRAM: ARCHITECTURAL FOUNDATION

The 6T SRAM, named for the six transistors in its core Bitcell (BC), is the most common topology in SRAM designs. Its fundamental architecture consists of two main components, illustrated in Figure 3.4a:

- **Storage element:** two cross-coupled inverters (four transistors) form a bistable latch that stores one bit.
- **Access path:** two access transistors connect the storage nodes to the complementary Bitline (BL) and Bitline-Inverse (BLB). Activating the Wordline (WL) turns on these transistors enabling data transfer between the BC and the BLs [88,89].

The compact and highly symmetric layout of the 6T BC delivers excellent integration density and strong immunity to electrical noise. During a read operation, charge transfer between the selected BC and the complementary BLs develops a small voltage difference, which the Sense Amplifier (SA) then amplifies to a full-swing digital level. The combination of differential BLs and fast sensing enables robust and energy-efficient readout, even at low supply voltages [90,91]. Together, these architectural features have established the 6T BC as the dominant choice in SRAM design [88,89,91,92].

Since SRAM typically accounts for a large fraction of total power in modern digital ICs, its energy efficiency directly affects overall system consumption [86]. The energy efficiency of SRAM is therefore commonly evaluated in two operating modes [86,87,92]:

- **Active mode:** P_{dyn} is consumed through charging and discharging of WLs and BLs during read and write operations.
- **Idle mode:** P_{stat} is dissipated due to leakage while data is retained but not accessed. In large arrays with long shared BLs, leakage can be significant and becomes more pronounced at advanced technology nodes [87,92].

Although reducing P_{dyn} often requires complex architectural and circuit co-optimization, the relatively simple 6T structure still permits effective, low-overhead techniques for power reduction [87,93]. Beyond power considerations, ensuring data integrity across PVT variations is a central requirement in SRAM design. This necessitates sufficient *stability*, encompassing both *writability* and *read stability*.

The Static Noise Margin (SNM) quantifies robustness as the maximum Direct Current (DC) disturbance the BC can tolerate without flipping state [90]. Graphically, SNM is represented as the side length of the largest square that fits between the Voltage-Transfer Curves (VTCs) of the two cross-coupled inverters (Figure 3.5a). In ideal conditions, a symmetric design maximizes SNM. However, in practice, V_{TH} variation, sizing mismatch, and layout imbalances distort the VTCs, degrading noise margins [90]. This effect is particularly severe during read operations (the worst-case condition), where the access transistors weaken the pull-down path and increase vulnerability to read disturb and potential bit flips [90].

Device-level sensitivity studies show how individual transistors in 6T BCs (Figure 3.4b) affect SNM and overall stability [90,94]:

- Pull-down transistors (PDTs) dominantly govern the drive strength and state stability in the BC, thereby strongly influencing SNM.
- Access transistors significantly affect Read Noise Margin (RNM) by disturbing internal node voltages.

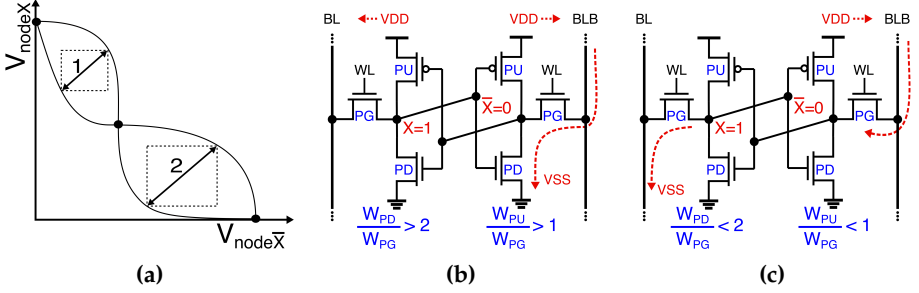


Figure 3.5: Evaluation of SNM in a 6T SRAM BC: (a) example butterfly VTC used to assess read, write, and hold operations independently; (b) read ‘1’ with required sizing to prevent read upset [94]; (c) write ‘0’ with required sizing to ensure writability [94].

- Pull-up transistors (PUTs) have comparatively less effect on SNM, due to lower drive strength and smaller sizing.
- Mismatch in transistor width or length (W/L) between the two halves of the BC lowers SNM, making careful symmetry essential for stability.

Figures 3.5b and 3.5c illustrate these effects and the required sizing for fault-free read and write operations [94]. As CMOS technology continues to scale, intrinsic device variations such as random dopant fluctuations and oxide thickness variation cause wider SNM distributions, reducing yield even in defect-free BCs. Maintaining robust operation in such environments requires tighter transistor matching and careful BC layout [90,94].

3.2.2 PERFORMANCE, RELIABILITY, AND MULTI-PORT REALIZATION

The demand for high-speed, energy-efficient processing has increased the importance of multi-port SRAMs, which enable multiple simultaneous accesses to the same memory bank. By supporting parallel operations, multi-port architectures significantly enhance memory bandwidth and system performance [95,96].

Extending the conventional 6T BC to multi-port operation, however, introduces significant design challenges. Since the same access path is shared for both reads and writes, concurrent activity can lead to contention, potential data corruption, and degraded stability [97–99]. Furthermore, incorporating multiple ports exacerbates the classic Power-Performance-Area (PPA) tradeoffs, as larger BLs and more complex peripheral circuitry increase area and power, while longer signal paths reduce access speed [100]. The key issues in multi-port SRAM design include:

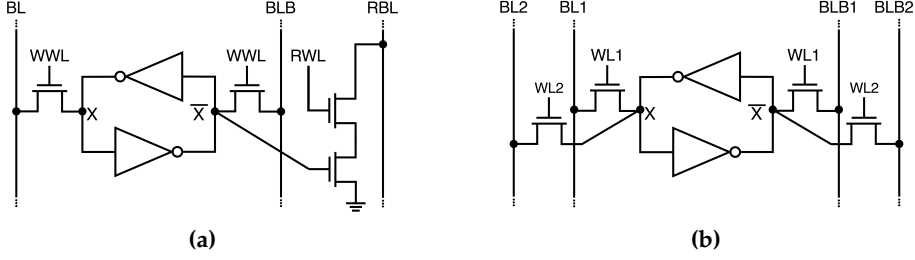


Figure 3.6: Proposed 8T SRAM BCs in literature: (a) 1R1W with decoupled read and write paths; (b) 2RW with coupled read and write ports [95].

- **Contention issue:** simultaneous access to the same location may cause: (a) full contention (one port writing while another reads) or (b) half contention (multiple concurrent reads) [101].
- **Area overhead:** each additional port requires larger BLs and extra peripheral circuitry, substantially increasing area [96,101].
- **SNM degradation:** concurrent operations disturb internal nodes, reducing SNM and endangering data integrity [97,99].
- **Disturb failures:** in same-row access mode, concurrent operations can lead to unintended read or write errors [102].
- **$V_{\min}$ degradation:** the minimum operating voltage often worsens in practice, reducing overall energy efficiency [102].

Because of these limitations, conventional 6T-based multi-port SRAMs struggle to scale efficiently. This has motivated the development of alternative architectures and custom BC topologies, which aim to balance parallelism with stability, area, and energy efficiency.

3.2.2.1 Eight-Transistor (8T) SRAM Architecture

The 8T BC is one of the most widely adopted alternatives to the conventional 6T design. By decoupling the read path, i.e., the Read Bitline (RBL), from the storage node (Figure 3.6a), the 8T BC implements a one-read/one-write (1R1W) configuration that overcomes key 6T limitations and provides greater robustness and versatility, albeit at the cost of a larger BC area compared to denser SRAM topologies.

Separating the read and write paths enables independent optimization of both operations, improving read stability and writability and reducing the risk of read disturb under LV or energy-constrained conditions [103]. This robustness makes 8T BCs attractive for multi-port implementations such as two-read/write (2RW) SRAMs (Figure 3.6b) [104].

The 8T structure also adapts well to energy-efficient and embedded applications. A decoupled read path maintains reliable operation even when the ratio of a transistor's on-state current (I_{ON}) to its off-state leakage current (I_{OFF}) is reduced. A high I_{ON}/I_{OFF} ratio ensures strong signal drive and low leakage, which improves noise margins and read stability. Lower access disturbance can further enhance robustness under harsh operating conditions. Designs such as Differential Data-Aware Power-supplied SRAM further exploit the data-independent leakage of the 8T BC to improve sub- V_{TH} energy efficiency [105–107]. The inherent multi-port capability of 8T SRAM provides another important advantage. It supports parallel operations without altering the base BC and typically uses a 6T BC as the core storage element. This capability is particularly valuable for IMC, where dot-product and similar operations can be executed directly within the array, yielding substantial throughput gains for AI and ML workloads [108].

Nevertheless, supporting multiple ports in 8T BCs increases design complexity, particularly in managing simultaneous read and write operations, which complicates layout and fabrication [104,106]. The additional functionality also introduces area overhead, a drawback that becomes especially critical in area-sensitive designs, since SRAM often occupies the largest portion of chip area.

3.2.2.2 Alternative Topologies in SRAM Design

Several SRAM topologies beyond the 6T and 8T BCs have been proposed in the literature, including 7T, 9T, 10T, and higher-transistor-count architectures. These designs are not direct successors to the 6T BC but represent diverse strategies to overcome its limitations. Each topology introduces unique enhancements, targeting challenges such as stability, leakage, writability, and robustness under Ultra-Low-Voltage (ULV) operation, thereby illustrating the breadth of innovation in SRAM design.

The 7T topology is often explored for ULV and energy-constrained environments. A single-BL 7T BC [111] achieves strong energy efficiency by boosting critical nodes, enabling a retention voltage as low as 200 mV and operational frequencies up to 150 MHz at 360 mV. Another 7T variant [109] improves sub- V_{TH} performance through a custom read-assist scheme that enhances the I_{ON}/I_{OFF} ratio, thereby reducing leakage while maintaining an area footprint comparable to the 6T BC (Figure 3.7a). Similarly, the design in [94] employs capacitive coupling to resolve conflicts during concurrent read and write operations, using two pass-gates to simplify the access structure.

The 9T BC emphasizes robustness across wide operating ranges. One approach [112] integrates Adaptive Supply Voltage Generation (ASVG), combining high- V_{TH} devices for leakage suppression with low- V_{TH} devices in

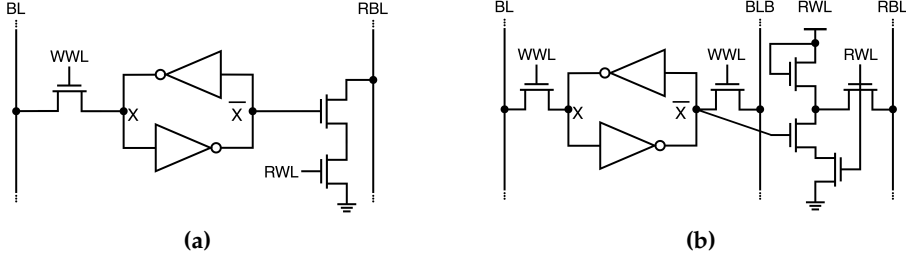


Figure 3.7: Examples of alternative SRAM BCs: (a) sub- V_{TH} 7T with read assist [109]; (b) high-performance 10T [110].

the read path to maintain speed. Another 9T design [113] supports dual-mode operation across 300 mV–1.2 V, effectively balancing low leakage in standby with reliable performance at higher voltages.

Further advancing the landscape, 10T SRAM designs often prioritize stability and leakage reduction. The BC in [114] incorporates a single-ended, decoupled 4T read path, improving RNM and cutting BL leakage by orders of magnitude compared to 6T. Another 10T variant [110] employs NMOS-only read ports to support near- V_{TH} operation, focusing on Low-Power (LP) consumption (Figure 3.7b). The proposed BC in [115] introduces a Schmitt-trigger inverter to achieve low standby power, enhanced read stability and writability in scaled technology nodes.

Higher-transistor-count designs target application-specific needs. The 11T SRAM in [116] employs a data-dependent power supply with a decoupled read path, significantly improving the I_{read}/I_{leak} ratio and reducing leakage, an appealing feature for IoT and other ULV use cases. The 12T design in [117] eliminates half-select issues of 6T arrays by using a cross-point write structure with row-column integration, while an isolated read path enhances SNM and ensures reliable access under challenging conditions.

In summary, these architectures address key SRAM design challenges by improving read stability, reducing leakage, and enhancing ULV robustness while complementing rather than replacing the 6T BC.

3.2.3 CHALLENGES IN SRAM DESIGN

Scaling trends and strict energy budgets expose SRAM to a set of interrelated challenges that directly affect its stability, performance, and efficiency. A key concern is voltage scaling in advanced technology nodes, as lowering supply voltage rapidly degrades reliability [87, 100, 118–120]. Reduced voltage shrinks the SNM, increasing susceptibility to read disturb and data corruption

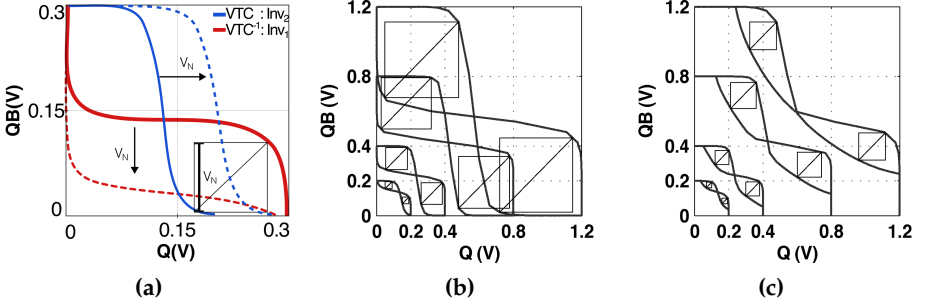


Figure 3.8: Impact of V_{DD} scaling on SRAM's SNM: (a) VTC for SNM extraction; when curve separation drops below SNM (e.g., $V_N = \text{SNM}$) the BC becomes mono-stable and loses data; (b) hold VTCs vs V_{DD} ; (c) read VTCs vs V_{DD} [121].

(Figures 3.8a, 3.8b, 3.8c) [88, 121]. Lower supply also weakens transistor drive strength, slowing switching and increasing access latency, which negatively affect the overall system performance [87].

Process variability further aggravates the reliability issues, as variations in V_{TH} or device dimensions widen the SNM distribution, producing a long tail of failing BCs even when the nominal design is stable [100, 120]. This variability lowers the fabrication yield and complicates large-scale SRAM deployment in deeply scaled nodes. At the same time, sub- V_{TH} leakage intensifies as voltage decreases, remaining one of the most persistent barriers to energy-efficient operation [114].

3.2.4 POWER OPTIMIZATION AND DESIGN-CHALLENGE MITIGATION

Robust, LP SRAM design requires coordinated techniques across abstraction levels to ensure stability across PVT while reducing P_{dyn} and P_{stat} :

- P_{stat} , dominated by sub- V_{TH} and gate leakage in idle BCs, mitigated by:
 - **PG and virtual power rails:** disconnect inactive blocks from the supply and ground using sleep transistors to suppress standby currents [114, 118, 122].
 - **Device/transistor-level techniques:** employ low-leakage or high- V_{TH} devices, MTCMOS, careful sizing/stacking, or robust BC topologies to reduce leakage without sacrificing performance [87, 88, 91, 106, 122].
 - **Data-retention biasing:** lower the supply of idle BCs to the minimum that maintains hold margin [87].
- P_{dyn} , dominant during read and write, can be reduced through:

- **Voltage scaling with assists:** operate near device V_{TH} for quadratic savings, using selective WL boosting and read-assist to preserve SNM and writability [87, 94, 100, 119, 120].
- **WL/BL optimization:** reduce BL swing with single-ended sensing, data-dependent precharge, and hierarchical/segmented BLs, plus compact layouts to lower parasitics and switching energy [89, 92, 93].

Beyond lower-level circuit optimizations, *System-level strategies* tune supply and access patterns to the workload:

- **DVS/DFS:** tune supply and clock dynamically to match workload demand in real-time [123–125].
- **Near-Threshold Computing (NTC):** bias entire arrays close to the device V_{TH} for energy savings, supported by assist circuits [88, 126].
- **Approximate and hybrid memories:** combine BC types (e.g., 8T for critical bits and 6T for less significant bits) or apply data-aware schemes such as Differential Data-Aware Power-supplied BCs to exploit error tolerance and state-dependent leakage [93, 105, 117, 127, 128].
- **Read/write-decoupled BCs and optimized peripherals:** isolate access paths with 8T or Schmitt-trigger BCs and use optimized SA and decoder designs together with innovative readout schemes to enhance stability and access speed in the LV regime [107, 122].

Taken together, these approaches show that no single technique suffices. Energy-efficient, reliable SRAM in advanced nodes relies on co-optimization across device, circuit, and architectural levels, tailored to the application and operating environment.

3.3 STANDARD-CELL BASED MEMORIES

Although 6T SRAM is the predominant on-chip memory, meeting requirements such as ULV operation, non-standard array sizes, or multi-port access remains challenging [129–131]. SCMs offer a flexible alternative, particularly for ULV systems with small, distributed memory blocks [132].

SCMs are described at the Register-Transfer Level (RTL) in Hardware Description Languages (HDLs) and synthesized using standard cells, enabling extensive configurability. Parameters such as word length, port count, bit/byte masking, and access latency can be tuned directly in HDL without custom memory compilers [133]. This automated flow shortens design cycles, supports portability across technology nodes, and integrates naturally into modern SoC development [131, 134].

A major strength of SCMs is their seamless placement within logic rows. Positioning SCM arrays close to the logic reduces routing overhead and

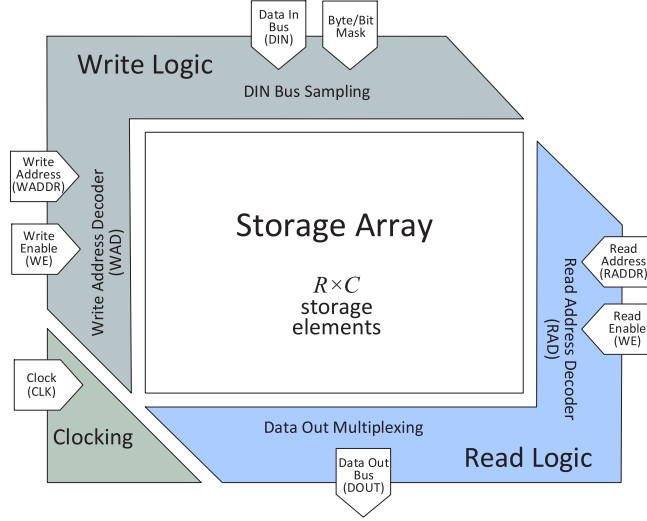


Figure 3.9: Building blocks of a generic SCM architecture, including storage array, read/write logic, and clock/control [131].

parasitic capacitance, which in turn improves timing and lowers P_{dyn} compared with stand-alone SRAM macros [135, 136]. Beyond layout flexibility, SCMs can incorporate independent read and write ports that minimize race conditions and enable parallel access [130, 132]. Robust operation in the LV region further extends reliable functionality into the sub- V_{TH} regime where SRAM typically fails [132].

In terms of efficiency, SCMs are most advantageous for small capacities on the order of kilobits [136, 137]. For larger arrays, however, the advantage diminishes because SCMs incur area penalties several times greater than optimized 6T SRAM [133, 134]. Nevertheless, in distributed low-capacity storage where SRAM periphery overhead dominates, SCMs can deliver superior energy-area efficiency and faster access, making them attractive for deeply energy-constrained IoT nodes and parallel compute engines [132].

3.3.1 SCM IMPLEMENTATION AND ARCHITECTURE

SCMs store data in standard-cell latches or Flip-Flops (FFs), avoiding common SRAM issues such as write contention, read disturb, and half-select conditions. They also operate reliably at supply voltages well below the limits of SRAM [132]. A generic SCM comprises a storage array with associated read/write logic and clock management, as shown in Figure 3.9 [131].

SCMs typically use latch- or FF-based cells but can be optimized with custom-characterized designs to reduce area, leakage, and delay [135]. Dual-bit pass-latch cells, for example, extend the area break-even with sub- V_{TH} SRAMs to about 2 kb for single-port and 4–6 kb for dual-port arrays [133], while recent 18T-SCM designs using AND-OR-Invert and OR-AND-Invert logic (AOI/OAI) further enhance density and efficiency. Although large SCMs (e.g., 256 kb) may occupy roughly eight times the area of 6T SRAM, several studies consistently report break-even points around 1 kb (single-port) and 4–6 kb (dual-port) [133, 134, 136].

The memory array is supported by efficient read/write logic. Write addresses are decoded into one-hot or one-cold signals to activate target cells, typically using CG-based schemes to lower switching activity and energy while reducing area [132]. Read addresses are similarly decoded to select target words, which can be accessed through multiplexers, tri-state buffers, or AOI/OAI-based daisy-chain decoders depending on design goals [133, 138].

SCMs integrate seamlessly with surrounding logic, and structured placement as a dedicated memory region can further improve density and energy. Grouping BLs near read multiplexers in an SRAM-like fashion reduces area by up to 25% for long word lengths and lowers read/write energy [131]. Even unstructured layouts remain more energy-efficient than 6T SRAMs in the LV region because NAND–NOR read multiplexers and aggressive CG consume much less energy than SRAM SAs.

Collectively, these architectural and layout techniques make SCMs well suited for distributed, low-capacity storage in energy-constrained systems where conventional SRAM cannot operate reliably or efficiently.

3.4 HYBRID MEMORY STRUCTURES

Hybrid memory structures combine different memory technologies to exploit their complementary strengths while mitigating individual limitations. This approach has become particularly relevant in embedded systems, IMC, and high-performance computing, where optimizing energy, performance, and area efficiency is critical [83, 84, 139–142].

A widely studied strategy integrates SCM and SRAM. With its fully digital design and reliable sub- V_{TH} operation, SCM is attractive for small, energy-critical memory regions, while 6T SRAM provides the density needed for larger blocks. This combination has been demonstrated in multi-level caches where SCM blocks of 0.5–2 KB act as Level-0 caches and larger 4–8 KB SRAM blocks serve as Level-1 caches, achieving robust LV operation with improved energy efficiency (Figure 3.10a) [83, 84, 142]. Although SCMs are less area-efficient

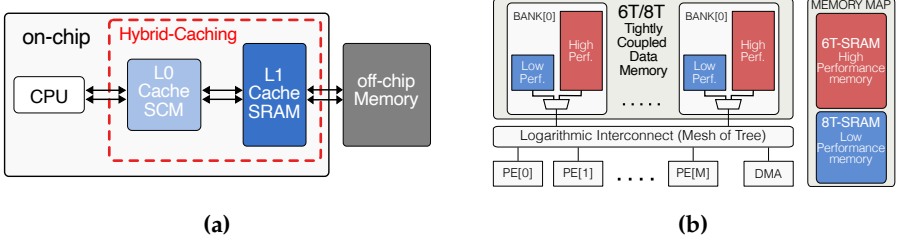


Figure 3.10: Hybrid memory examples: (a) SCM Level-0 for LV energy efficiency with larger SRAM Level-1 caches [83]; (b) combined 8T SRAM for LV robustness with 6T SRAM for high-density storage [85].

than SRAM for large arrays, studies show clear energy advantages for cache capacities below about 4 KB [83, 143].

In another approach, 6T and 8T SRAM are combined to exploit their complementary strengths: the 8T BC delivers reliable ULV-region operation, while the 6T BC provides compact, high-density storage. Such hybrid partitions are especially valuable in biomedical and IoT applications, where robustness under near- V_{TH} operation is critical (Figure 3.10b) [85]. Hybrid designs can also combine SRAM and *embedded Dynamic Random-Access Memory (eDRAM)*, creating efficient cache hierarchies where SRAM delivers fast access for frequently used data, while eDRAM provides higher capacity at lower area cost [144]. Combining SRAM with *emerging Non-Volatile Memories (NVMs)* has gained traction for AI accelerators and IMC [139–141, 144, 145]. In these designs, SRAM supplies fast read/write capability, while emerging NVMs contribute density, data retention, and low standby power, creating a balance of performance and energy efficiency.

Despite their promise, hybrid memories pose design challenges. Managing heterogeneous technologies demands complex control logic, multiple voltage domains, and careful synchronization [141, 144, 145].

System-level techniques are key to overcoming these issues. Compiler-driven heuristics and dynamic programming place frequently accessed data in the most suitable memory to cut access time, energy, and NVM wear [139, 142, 146]. Robust address mapping and control logic manage transfers across memories with different speeds, while eDRAM refresh and voltage regulation add complexity [84, 142]. High-speed interfaces remain critical to sustain throughput and fully exploit each technology [139, 145].

3.5 EMERGING MEMORY TECHNOLOGIES

Emerging memory technologies such as Phase-Change Memory (PCM), Spin-Transfer-Torque Magnetoresistive Random-Access Memory (STT-MRAM), Magnetoresistive Random-Access Memory (MRAM), and Resistive Random-Access Memory (ReRAM) are being actively developed as successors to conventional memories like SRAM and DRAM. They offer fast access, non-volatility, high density, and CMOS compatibility, positioning them as strong candidates for future high-performance and energy-efficient systems [147, 148].

Compared to conventional hierarchies, these technologies provide lower latency, with PCM and STT-MRAM approaching near-SRAM speeds while maintaining non-volatility [147]. Their performance also exceeds Hard Disk Drives (HDDs) and, for some workloads, Solid State Drives (SSDs), making them attractive for next-generation storage [149]. Commercial platforms, such as Intel's 3D XPoint demonstrate their density advantage and ability to bridge the gap between fast volatile memory and large-capacity storage by combining low latency with persistence [150, 151].

Each technology brings unique device-level strengths. PCM leverages phase transitions in chalcogenide materials for fast, durable storage suited to storage-class memory and IMC [152]. STT-MRAM encodes data magnetically, offering both speed and endurance for caches and embedded systems [147, 153]. ReRAM relies on resistive switching, enabling LP operation and excellent scalability for neuromorphic, IoT, and EC [154]. Architectural innovations such as 3D integration and cross-point (1T1R) arrays further raise density and energy efficiency while addressing issues like sneak-path currents and write overheads [153]. These attributes align well with IMC and NMC, where reducing data movement accelerates AI and ML workloads [155].

Despite their promise, several challenges remain. Reliability is affected by limited endurance and data-retention variability [148, 152], while extensive testing across operating conditions adds cost and complexity [152]. Scalability and manufacturability at advanced nodes are still open concerns [148]. System-level integration demands new controllers, software stacks, and compilers to accommodate different access characteristics [147, 156]. Endurance limits and write-energy costs can further restrict deployment in high-write or ULV environments [148].

Widespread adoption will ultimately depend on overcoming these technical challenges while establishing standards and design ecosystems. As these memories enter AI-driven systems, privacy, security, and ethical implications must also be addressed to ensure effective deployment [148].

3.6 THESIS CONTRIBUTION

One contribution of this thesis is a true dual-port SRAM architecture that uses conventional high-density 6T BC and single-ended read/write schemes rather than standard differential BLs. Achieving multi-porting with a 6T core without adding per-cell devices for extra ports is challenging because concurrent accesses degrade SNMs and aggravate read-disturb and writability conflicts, which can offset anticipated gains.

The architecture preserves 6T density while enabling concurrent access through three co-designed elements:

- a dual-rail organization that isolates read and write activity and reduces BL contention
- Boost-Assisted Single-Ended (BASE) read and write schemes that momentarily strengthen only the critical nodes during access while keeping the rest of the memory in the LV region
- LP peripherals, including a three-stage tree decoder that reduces switched and leakage paths, daisy-chained NAND/NOR readout without SAs, and data-aware LP precharge and write circuitry

For reliable ULV operation, the array biases the BCs near DRV to curb P_{dyn} . A single-cycle, tunable on-chip charge pump can supply an xV_{DD} overdrive to selected WLs and BLs rails on demand [111,157]. In the fabricated prototype, external sources provide boosting to allow wider tuning during testing, and the architecture remains fully compatible with an integrated charge pump. Additional LP measures include hierarchical BL segmentation to cut capacitance and speed up sensing, and Virtual Ground (VGND) implementation via a small tail-buffer stack that floats the node at the bottom of each global BL segment to remove the DC path to ground. This prevents unnecessary full-swing charge and discharge and substantially reduces standby and read I_{Leak} . Operation-dependent boosting of otherwise virtual rails restores margins only when needed. Together, these choices provide concurrent access without the area cost of 8T dual-port cells.

Paper I details the circuit implementation and silicon in 28 nm FD-SOI. Measurements show reliable dual-port functionality down to 0.29 V, energy of 1.35 fJ/bit-access at 0.54 V and 80 MHz, and approximately 62 % smaller array area than 8T dual-port baselines at comparable throughput. This contribution complements the embedded SRAM discussion in **Chapter 3** and underpins later system-level studies.

4

Synthesizable Distributed SRAM



THIS chapter introduces a synthesizable distributed SRAM (SD-SRAM) architecture, a flexible on-chip memory solution based on area-efficient 6T BCs and designed for seamless integration into larger computing systems. The proposed architecture has been realized as a synthesizable memory IP, successfully implemented and fabricated in silicon with funding and technical support from Xenergetic AB.

4.1 FROM FIXED MACROS TO FLEXIBLE INTEGRATION

On-chip memories are pivotal to system performance and energy consumption, particularly in energy-constrained platforms such as smart sensors and embedded devices where efficiency is critical. SRAM has long served as the primary high-speed memory in digital ICs, valued for its density and CMOS compatibility, but conventional implementations rely on foundry-specific hard macros or full-custom layouts, which restrict portability and design flexibility.

Table 4.1: Normalized area comparison of different storage cells in STM 28 nm FD-SOI technology. Area is normalized to a single 6T BC.

Storage Cell	Foundry 6T BC	D-Latch	D-FF
Area [n.u]	1	7	16

To address these limitations, SCMs have emerged as a fully digital and configurable alternative. However, their efficiency does not scale well beyond small capacities, as each BC requires a clock signal, resulting in complex clock distribution trees, routing congestion, and high P_{dyn} [131]. Despite

Table 4.2: Comparison between SRAM and SCM across key efficiency metrics, illustrating trade-offs in scalability, robustness, clocking, and density.

Feature	SRAM	SCM
Form	Fixed (hard macro)	Synthesizable (digital flow)
Scalability	Poor for small sizes	Scalable
Robustness	Limited	High(digital gate)
Clock Network	Controlled (low-power)	Irregular (high-power)
Memory density	High	Low
Suitability	large-size efficient	small-size efficient

these drawbacks, SCMs remain attractive for small, LV, energy-constrained blocks. Table 4.1 reports the normalized area cost of an optimized 6T SRAM BC compared with two common standard-cell storage elements used in SCMs, evaluated in a commercial 28 nm FD-SOI technology. The broader tradeoffs between SRAM and SCM are summarized in Table 4.2.

Combining the advantages of SRAM and SCM, this work presents the Synthesizable Distributed SRAM (SD-SRAM) as a solution for expanding the design space of on-chip memories. The architecture continues to use 6T BCs defined within each technology node but removes reliance on fixed, foundry-provided hard macros by coupling dense arrays with fully digital peripherals. Configurable sizing and distributed placement close to logic blocks improve data locality, while an HDL-driven design flow avoids the routing and placement constraints associated with analog hard macros. By combining the density of SRAM with the flexibility of SCM, SD-SRAM delivers a scalable and portable memory solution tailored for data-centric SoCs and memory-centric computing scenarios such as NMC and NDP.

4.2 ARCHITECTURE

The SD-SRAM architecture preserves the functional organization of conventional SRAMs, with key components such as the BC array, address decoder, precharge circuits, read and write paths, and a centralized timing unit (Figure 4.1a). Unlike hard-macro designs, these blocks are restructured for synthesizability and seamless integration into digital flows. The proposed SD-SRAM adopts the same main blocks, including the BC array, address decoder, precharge circuitry, read and write data paths, and timing control, but restructures them for synthesizability and flexibility (Figure 4.1b).

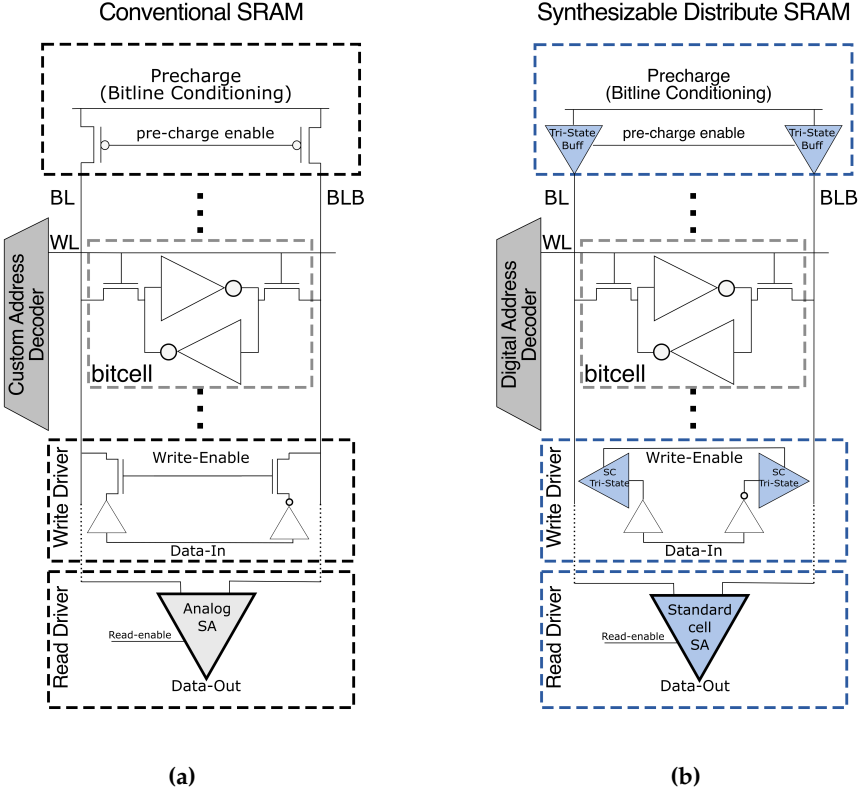


Figure 4.1: (a) Conventional SRAM architecture based on fixed hard macros; (b) proposed SD-SRAM architecture enabling flexible sizing, synthesizability, and distributed placement.

4.2.1 GENERATION FLOW

Each internal block of SD-SRAM is designed as a synthesizable digital module. Replacing analog periphery with digital equivalents allows integration through standard RTL and synthesis flows while requiring careful abstraction of timing-critical elements.

BC behavior is captured by monitoring charge transfer between storage nodes and BLs when the WL is asserted. This provides the delay and switching characteristics required for synthesis and verification. SAs are characterized by measuring the response to voltage differences between BLs when the sense-enable signal is active. In some cases a simplified digital buffer can substitute

for a full analog SA, offering easier integration at the cost of sensitivity. *Precharge units* and *write drivers* are implemented with tri-state logic gates that mimic the switching behavior of analog transistors in a purely digital form. *Timing* control is managed with counters instead of analog delay circuits. Each counter is configured with worst-case timing values extracted across process corners. During precharge, a counter starts when an access request is received and automatically deasserts the precharge signal once the threshold is reached. Similar logic governs read and write operations. An on-chip configurable ring oscillator provides a timing source, with external clock support as a backup. A multiplexer selects the clock source, and clock dividers generate different operating frequencies for evaluation (Figure 4.2a).

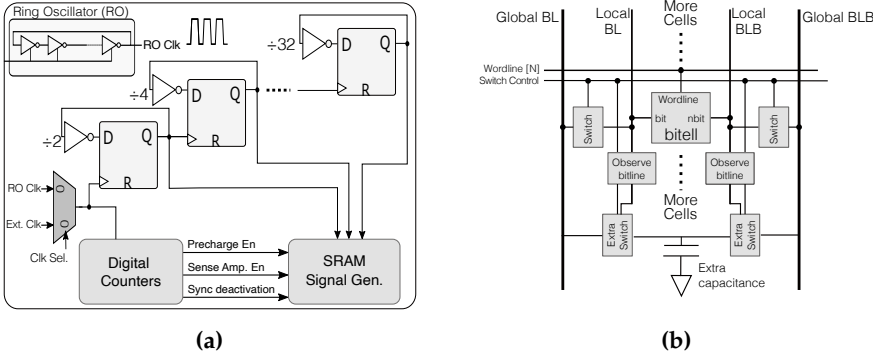


Figure 4.2: (a) Timing control unit of SD-SRAM based on digital counters and clock dividers; (b) segmented BL hierarchy with local and global lines to reduce capacitance and improve energy efficiency [158].

4.2.2 BITLINE HIERARCHY

The performance and area efficiency of memory arrays are heavily influenced by the BL length. Long BLs increase capacitance and reduce access speed, while short BLs create routing overhead and raise area cost. SD-SRAM addresses this tradeoff with a segmented hierarchy (Figure 4.2b) where groups of BCs connect to local BLs that are aggregated onto shared global lines through switches [158].

Segmentation lowers capacitance, allows multiple groups to share peripherals, and improves both energy and area efficiency. The segmentation depth must be chosen carefully. Too few BCs per group wastes density, while too many introduce switch delays that harm access time. In contrast, unsegmented

arrays dedicate a full read and write path to every BL, which improves raw speed but consumes excessive area and power for small to medium blocks. The segmented structure in SD-SRAM offers a balanced compromise, combining dense integration with scalable and energy-efficient performance.

4.3 PROPOSED DESIGN

The design flow begins with the creation and characterization of memory macros in four array sizes: 16×32 , 32×32 , 64×32 , and 128×32 . A 32-bit word length is selected to align with the data bus of the target platform, PULPino*. To meet the PULPino memory requirements, the macros are assembled into larger blocks. For instance, implementing an 8 KB private cache with 16×32 macros requires 16 instances. Memory macro placement is performed using the mixed-placement technology GigaPlace XL within the Cadence Innovus Implementation System [159]. Placement is guided by constraints such as keep-out regions, while signal routing follows the standard flow with the integrated NanoRoute engine. Wire lengths and parasitic capacitances are extracted for evaluation across different design scenarios.

Circuit-level simulations are conducted in Cadence Virtuoso using Spectre, with routing parasitics captured in Standard Parasitic Exchange Format (SPEF) during Place-and-Route (PnR) in Innovus. Four fundamental operations, write '1', write '0', read '1', and read '0', are used to evaluate the complete memory system, with detailed analysis of latency and energy. Read and write power are further partitioned across the main contributors including address decoders, read logic, and precharge unit.

Preliminary results at 200 MHz and 0.9 V show that larger arrays achieve better energy efficiency and higher operating frequency. The 128×32 array demonstrates the lowest power consumption (Figures 4.3a and 4.3b) and the shortest minimum clock period (Figure 4.4a), while maintaining an intrinsic read access time suitable for reliable sensing, defined as the delay to develop a 100–200 mV differential across the BLs (Figure 4.4b). These results establish the 128×32 configuration as the main SD-SRAM building block, while the 64×32 array remains a practical alternative when integration constraints apply.

To further scale capacity, SD-SRAM employs hierarchical access based on segmented BLs. For example, when 16×32 macros are deployed, four macro instances are grouped under a shared global BL, enabling hierarchical communication between local and global segments. This reduces capacitive loading and improves both scalability and access efficiency.

* PULPino: open-source single-core RISC-V SoC developed at ETH Zurich within the PULP platform, available at <https://github.com/pulp-platform/pulpino>.

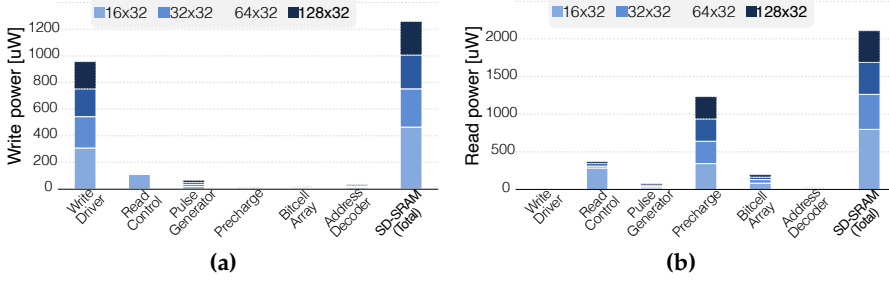


Figure 4.3: Power consumption of internal memory blocks during (a) write and (b) read operations for different array sizes (16x32, 32x32, 64x32, and 128x32).

The next section provides a comprehensive analysis of post-layout simulation results and includes comparisons with SoA designs.

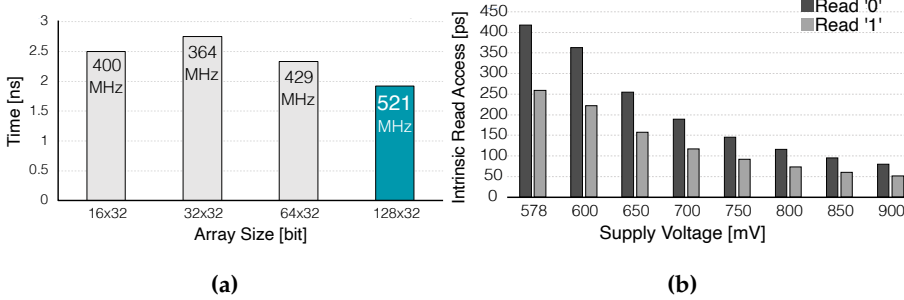


Figure 4.4: (a) Maximum operating frequency of SD-SRAM as a function of array size; (b) intrinsic read access time for the 128x32 configuration, measured as the delay to develop a 100–200 mV differential across BLs.

4.4 RESULTS AND DISCUSSION

Table 4.3 summarizes representative SCM implementations across technology nodes and highlights their density, energy, frequency, and voltage characteristics. Latch-based SCMs such as [143] operate reliably near- V_{TH} but suffer from poor density of $5.9 \mu\text{m}^2/\text{bit}$ at 65 nm. AOI and OAI based SCM in [134] improves configurability and integration but reports operating frequencies below 0.1 MHz near V_{min} , which limits broader applicability. More aggressive

Table 4.3: Comparison of representative SoA synthesizable memory implementations across different technology nodes.

Refs	PATMOS'16 [143]	TCASI'19 [134]	JSSC'22 [160]	ISCAS'17 [161]	ESSCIRC'14 [135]	This work
Technology	65 nm	40 nm	55 nm	20 nm FinFET	65 nm	28 nm
Bitcell structure	standard cell latch	standard cell 18T (OAI/AOI)	full-custom 13T	full-custom 14T (1W1R)	full-custom dual-bit latch	foundry 6T SRAM
Capacity (Array size)	16 kb (512 x 32)	90 kb (2048 x 45)	8 kb (8 x 16 x 64)	4 kb (32 x 128)	4 kb (128 x 32)	16 kb (4 x 128 x 32)
Area @Orig. node [$\mu\text{m}^2/\text{bit}$]	5.9	7.6	5	5.4	8.5	0.45
Area @28 nm scaled [$\mu\text{m}^2/\text{bit}$]	1.79	4	1.79	11.88	2.58	0.45
Vmin [V]	0.4	0.4	0.324	0.42	0.37	0.57
Fmax @Vmin [MHz]	14.5 (no VBB)	0.053 (Wr) 0.028 (Rd)	0.423 (Wr) 0.547 (Rd)	86	0.08@0.45 V	250
Fmax [MHz]	190@1.2V	32@1.1V (Wr) 27@1.1V (Rd)	150 (Wr) 113.4 (Rd)	3000@1.1V	20@0.7V	520@0.9V
E _{min} @Orig. node [fJ/bit]	38@0.4V (Wr) 21@0.4V (Rd)	130@0.5V (Wr) 30@0.7V (Rd)	54.1@0.4V (Wr) 32.8@0.4V (Rd)	39 (5pJ/cycle)	35@0.45 V	10.6@0.57V (Wr) 7.6@0.57V (Rd)
E _{min} @28 nm scaled [fJ/bit]	9@0.4V (Wr) 5@0.4V (Rd)	131.3@0.5V (Wr) 30.3@0.7V (Rd)	12.9@0.4V (Wr) 7.8@0.4V (Rd)	52.7	9	10.6@0.57V (Wr) 7.6@0.57V (Rd)
Area break-even	–	6 kb	8 kb	4 kb	4 kb	14-16 kb

custom designs, including the 13T SCM in [160] and the 14T SCM in [161], reduce V_{min} to below 0.35 V but incur significant area overhead and require full custom cell design, limiting portability. Dual-bit latch SCMs [135] achieve improved energy efficiency but scale poorly in advanced nodes.

In contrast, the proposed SD-SRAM achieves a normalized BC area of $0.45 \mu\text{m}^2/\text{bit}$ at 28 nm, significantly smaller than the $5\text{--}8.5 \mu\text{m}^2/\text{bit}$ range typical of prior works, while extending the SRAM break-even point to 14–16 kb. These results are obtained through a fully synthesizable RTL-to-Graphic Data System II (GDSII) flow, preserving compatibility with standard ASIC methodologies and avoiding the complexity of transistor level design.

Figure 4.5a shows that SD-SRAM extends the area break-even point with hard-macro SRAM to nearly twice that of previous SCMs, including latch based and AOI or OAI designs. Figure 4.5b further illustrates that density, defined as the ratio of BC area to total memory area, which remains within 3 % of hard-macro SRAM at 16 kb capacity. These results confirm that the proposed fully synthesizable flow outperforms conventional hard-macro SRAMs up to 14 kb, while extending the established boundaries of practical SCM design.

For energy efficiency, SD-SRAM operates reliably down to 570 mV. At this voltage, read power is $2.64 \mu\text{W}/\text{bit}$ ($84.56 \mu\text{W}/\text{word}$) and write power is $1.9 \mu\text{W}/\text{bit}$ ($60.87 \mu\text{W}/\text{word}$), leading to minimum access energies of 7.6 fJ/bit for read and 10.6 fJ/bit for write. These values, shown in Figure 4.6a, fall within or surpass the best reported for SCMs, which range from 21–38 fJ/bit

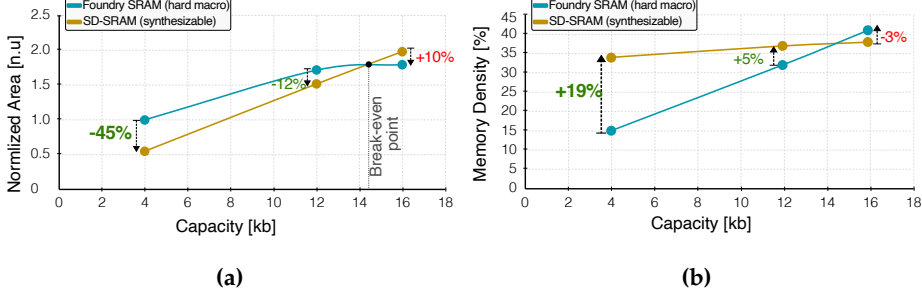


Figure 4.5: (a) Total area comparison between SD-SRAM and hard-macro SRAM across different capacities, indicating the area break-even point; (b) memory density comparison, defined as the ratio of BC area to total memory area.

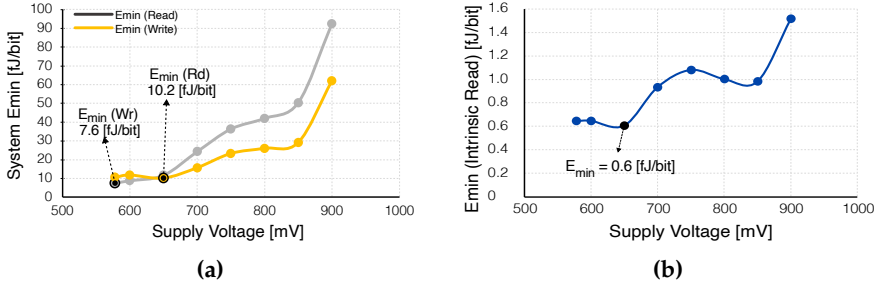


Figure 4.6: (a) Minimum energy point (E_{min}) for read and write operations at 250 MHz across different supply voltages; (b) intrinsic read energy obtained from Spectre post-layout simulations.

in [143] to 30–131 fJ/bit in [134]. Figure 4.6b, further illustrates the intrinsic read energy, measured from the assertion of a request to successful sensing by the amplifiers.

Performance results further confirm the observed energy trends. The design sustains 250 MHz at 0.57 V, which is more than $10\times$ faster than typical SCM implementations that operate below 1 MHz near- V_{TH} . Even optimized custom SCMs such as hybrid latch transistor approaches or 13T and 14T cells remain constrained by limited speed, while SD-SRAM achieves a balanced tradeoff between energy, area, and frequency. At nominal 0.9 V, it reaches 520 MHz, offering both high performance and portability.

Finally, scalability is demonstrated across a wide range of capacities. Although SCMs are often considered practical only below 6 kb [134, 162], the proposed SD-SRAM remains competitive at larger sizes. Figure 4.7 shows the physical layout with distributed placement of memory macros alongside computing logic, generated entirely within a standard digital flow, along with a die micrograph of fabricated chip. These results establish SD-SRAM as a synthesizable memory that combines SRAM class density with SCM level flexibility, redefining the scope of energy efficient and tightly integrated on-chip memory design.

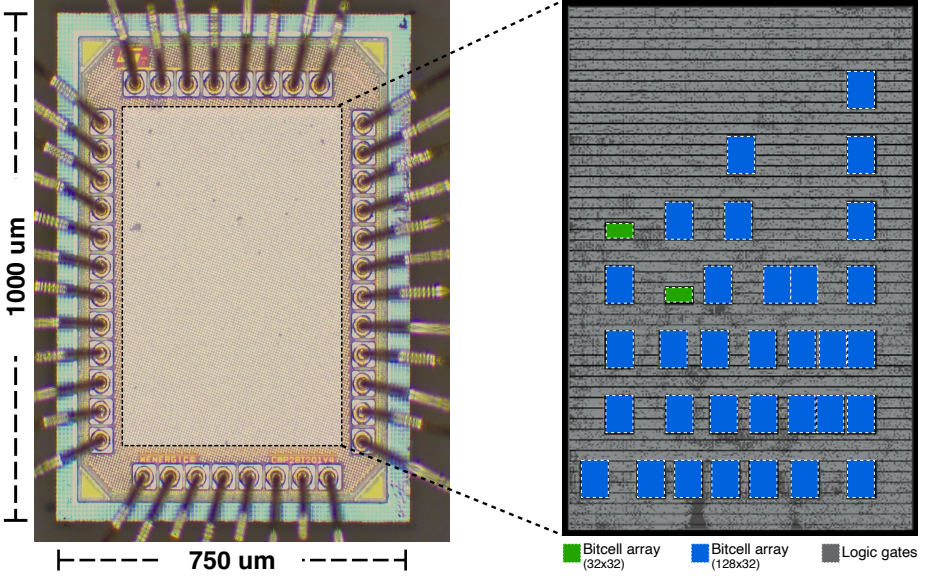


Figure 4.7: Distributed placement of SD-SRAM memory macros alongside logic blocks generated through a standard digital flow, with die micrograph of the fabricated chip.

4.5 THESIS CONTRIBUTION

This thesis develops a Synthesizable Distributed SRAM (SD-SRAM) architecture constructed from conventional 6T BCs and implemented entirely with synthesizable digital logic. The design introduces a modular structure with distributed banks and parallel access paths, enabling capacity scaling, concurrent accesses, and seamless integration into standard-cell-based ASIC flows. The architecture was realized through a full RTL-to-GDSII flow in

STM 28 nm FD-SOI technology. Although complete system-level silicon measurements were constrained by timing issues, functional verification and post-layout analysis validated the architectural concept and demonstrated its potential for energy-area advantages over conventional embedded SRAM implementations. A key benefit is the distributed placement of memory banks close to the logic blocks they serve, which reduces parasitic capacitance and resistance in on-chip interconnects. This lowers P_{dyn} and further improves overall energy efficiency.

Within the thesis, this contribution extends the discussion of **Chapter 3** by directly addressing the scalability and integration challenges of embedded memories. The proposed framework establishes a synthesizable and flexible memory IP suitable for energy-efficient edge platforms, while also forming a foundation for the near-memory computing exploration in subsequent chapters.

5

Approximate Computing: Techniques and Design Strategies



THIS chapter explores AC techniques, with a particular focus on approximate multiplication, examining their technical challenges and the critical role they play in enabling highly efficient processing paradigms for error-tolerant applications.

5.1 TURNING IMPERFECTION INTO EFFICIENCY

In the past few decades, advances in technology scaling have enabled an increasing number of transistors to be integrated within the same silicon footprint as previous technology nodes, thereby enhancing computational capabilities. However, this progress also introduces physical limitations, such as elevated power density and the emergence of dark silicon, a phenomenon in which large portions of a chip must remain powered down to manage thermal constraints [163]. Consequently, new design methodologies are required to sustain the efficiency gains achieved through technology scaling.

In many modern applications, such as big data analytics, ML, signal processing, and multimedia, full-precision computation is often unnecessary, as small numerical deviations rarely affect output quality or decision accuracy. Allowing controlled inaccuracies can therefore reduce hardware complexity and improve both energy efficiency and performance [164]. Moreover, due to perceptual limitations, small distortions in image, audio, or video output often go unnoticed, while noise and quantization during data conversion introduce imperfections that further limit achievable precision.

In this context, AC has emerged as a key paradigm to meet the growing demand for energy-efficient and high-performance Very-Large-Scale Integration

(VLSI) systems by exploiting the intrinsic error resilience of many workloads. This resilience, illustrated in Figure 5.1, arises from several factors [165]:

- lack of a single correct (*golden*) result,
- use of heuristics or suboptimal algorithms,
- noisy inputs and error-resilient computation patterns like aggregation and iterative refinement.

Studies indicate that a substantial portion of application runtime is spent on computations that can tolerate small inaccuracies [166].

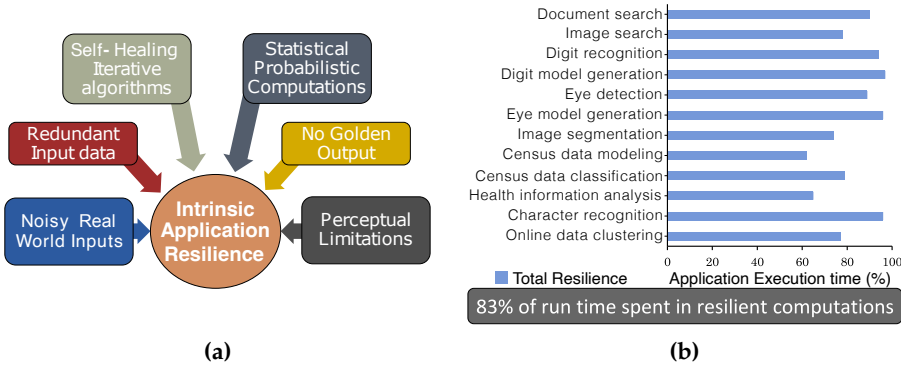


Figure 5.1: (a) Key sources of error tolerance; (b) representative application domains with inherent error resilience [165].

Consequently, strict enforcement of precision across the computing stack, from circuits and architectures to algorithms and software, often leads to significant inefficiencies. AC addresses this challenge by selectively relaxing accuracy constraints, enabling more aggressive optimizations and resource savings across multiple levels of abstraction [164].

A fundamental principle of AC is to achieve disproportionate benefits, i.e., obtaining substantial improvement in efficiency with minimal degradation in quality [165]. However, simplistic approximations of accurate designs often yield limited improvements, and many ad hoc circuits under-perform even basic truncation-based strategy. Functional approximation, such as the use of binary logarithms, offers a more promising alternative, enabling significant reductions in area and power consumption, but at the cost of reduced accuracy. Despite these advances, enhancing the accuracy of functional approximations with minimal hardware overhead remains an active area of research [164]. Adapting approximation strategies to the specific quality requirements of target workloads can significantly increase the obtained efficiency gains [167].

Nevertheless, AC also presents significant challenges. Effectively managing the tradeoff between energy savings and accuracy loss is critical, particularly in computations where errors cannot be easily tolerated. Both the choice of approximation method and the context in which it is applied have a strong impact on system performance and reliability.

Ongoing research continues to explore several open challenges, including the formulation of meaningful quality metrics, the development of robust verification techniques, the identification of components amenable to approximation, and the assurance of system reliability under approximate execution [166].

5.2 DESIGN METHODOLOGIES

Approximation techniques can be applied across multiple levels of abstraction, as illustrated in Figure 5.2.

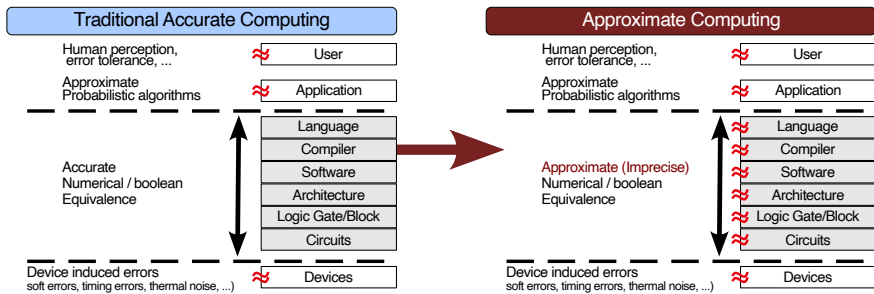


Figure 5.2: Comparison of accurate and approximate computing approaches across abstraction levels [168].

At **software level**, AC focuses on enabling programmers to explicitly manage approximations within their code. This includes developing approximation-aware programming languages and compilers [169], providing abstractions to specify which parts of a program can tolerate imprecision [170]. New approximate data types [171], such as the Approximate Integer Format (AIF) [172], can be introduced to allow developers to declare variables or operations where precision can be relaxed. Compilers can identify or be guided toward regions of code suitable for approximation, applying optimizations such as selectively skipping computations or reducing memory accesses [169]. In parallel and multi-threaded systems, approximation may also involve relaxing global synchronization and communication, thereby improving scalability and throughput [173].

At **architectural level**, AC explores the application of controlled imprecision across heterogeneous hardware components, including processors, memory subsystems, and interconnects [165]. GPPs can be enhanced to support approximate execution of selected instructions. Instruction Set Architectures (ISAs) may be extended with special approximate instructions [171], enabling compilers to specify tolerance boundaries without prescribing specific implementations [169]. Architectural techniques such as "Truffle" introduce dual-voltage operation, allowing lower voltages for approximate operations based on the instruction stream [170]. Additionally, coarse-grained approximation strategies can offload tolerant computations to dedicated LP cores [174].

Approximate memory designs, such as reducing DRAM refresh rates, demonstrate the tradeoffs between energy savings and data reliability [165,170]. Another technique generates hybrid SRAM arrays that combine 8T BCs for critical data and 6T BCs for non-critical data, allowing for aggressive voltage scaling and, thereby achieving substantial power savings while maintaining acceptable reliability levels [175].

At the *algorithm level*, the focus is on developing algorithms which can deliver *good enough* results, while potentially sacrificing strict precision to gain efficiency [165]. Several techniques and concepts are employed at this level:

- **Heuristics and Probabilistic Algorithms:** Algorithms such as Automatic Sequential Logic Approximation Methodology target near-optimal solutions with lower computational complexity and energy use, while introducing minor inaccuracies, as in Joint Photographic Experts Group (JPEG) compression's Discrete Cosine Transform (DCT) outputs [176]. On the other hand, probabilistic algorithms incorporate selective randomness in various stages of computation.
- **Significance-Driven Logic (SDL):** Prioritizes computations by their impact on output quality, allowing low-significance operations to be approximated with negligible effect on accuracy [165,176,177].
- **Incremental Refinement:** Algorithms such as Fast Fourier Transform (FFT) can terminate early to save energy, trading minor decreases in output quality for efficiency gains [177].
- **Application-Specific Modifications:** Restructures applications for approximation by prioritizing key features and identifying dominant operations (e.g., L1/L2 norms in clustering) that tolerate errors [165,176,177].
- **dependency relaxation and computation skipping:** Relaxes inter-iteration dependencies in iterative algorithms to enable parallelism and reduce synchronization overhead [173]. Techniques such as loop

perforation further save energy by selectively skipping iterations with controlled probability [166].

- **Algorithmic simplification:** Approximates complex operations like multiplication or division with simpler ones such as addition or subtraction, for example using Mitchell's binary logarithm approximation [164, 169, 178].

At the *circuit/hardware level*, AC improves energy efficiency and performance by modifying arithmetic units to operate at lower voltages or higher frequencies [164, 165]. Often referred to as approximate (inexact) hardware, this approach directly targets Boolean logic and arithmetic components [164, 179]. Figure 5.3a illustrates common circuit-level design approaches for AC units. Key techniques include:

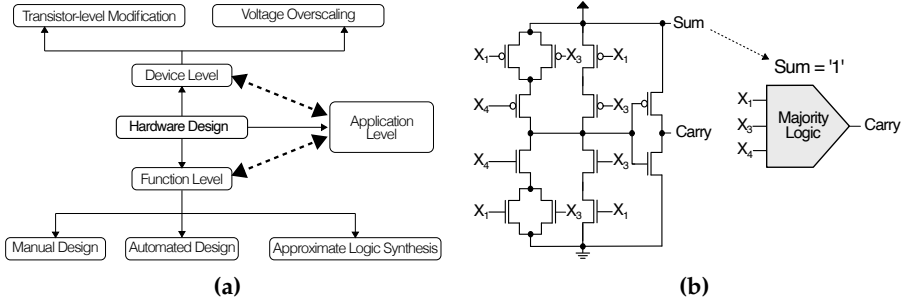


Figure 5.3: (a) Circuit-level design flow for AC [180]; (b) example of a 3-input majority gate at the transistor-level used in approximate logic [181].

- **Voltage Over-Scaling (VOS):** Reduces supply voltage below nominal levels, allowing timing violations in some combinational paths that lead to controlled errors [164].
- **Transistor-level and logic simplification:** Approximation can be introduced by modifying standard cells or adder structures at the transistor level or by heuristically simplifying logic functions through altered truth tables and omission of non-critical conditions. Both methods reduce layout complexity and gate count while preserving functional correctness within tolerable error bounds (Figure 5.3b) [169, 181, 182].
- **Majority Logic:** Uses majority voters (e.g., 3-input majority gates) where the output is '1' if at least two inputs are '1', expressed as $F = AB + BC + AC$ (Figure 5.3b) [181, 183].

- Approximate compressors and arithmetic units:** Simplify reduction trees by lowering logic complexity or output width [184–189]. In conventional multipliers, Half-Adders (HAs) and Full-Adders (FAs) act as binary compressors that combine multiple input bits into sum and carry outputs. For example, a FA functions as a 3:2 compressor. These compressors typically rely on Exclusive OR (XOR) logic, which increases propagation delay and limits performance. Approximate compressors address these issues by performing general X:Y bit compression using simplified logic, achieving higher efficiency. Building on these, approximate adders and Approximate Multiplier (AM) are constructed with simplified compressors or partial logic omission [179, 182, 184, 190, 191]. Figures 5.4a and 5.4b show examples of such approximate adder designs.

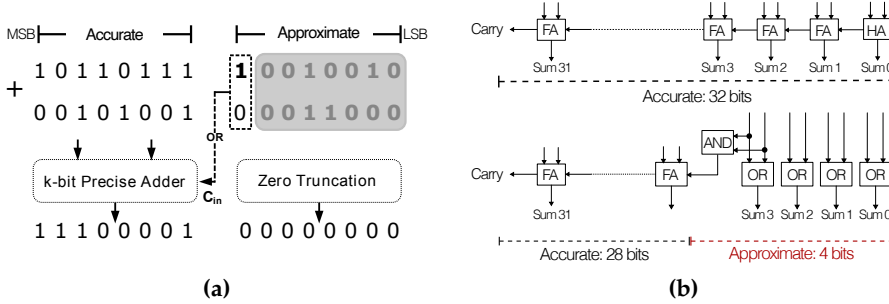


Figure 5.4: (a) Zero-truncation adder with OR-based carry prediction [192]; (b) 32-bit approximate RCA with 4-bit approximation in the LSB region [191].

5.3 APPROXIMATE MULTIPLIER DESIGN

Multiplication is among the most energy- and area-intensive operations in digital systems [193]. Approximate Multipliers (AMs) offer substantial benefits in error-tolerant applications by simplifying logic and reducing switching activity. The key design principle is logic simplification, i.e., reducing hardware complexity while maintaining acceptable accuracy.

Accumulation-tree and Partial Product (PP) reduction techniques collectively aim to reduce computational complexity and hardware cost in AMs by lowering the number of generated PPs, streamlining tree accumulation, and optimizing compressor structures for lower delay, power, and area overhead:

- **PP Perforation:** Selectively omits PPs, either statically by skipping fixed blocks of successive terms or dynamically based on bit significance to remove lower-impact bits [182,194].
- **Truncation:** Discards the least significant PPs to reduce bit-width and simplify compression [195]. Dynamic segmentation variants preserve only the most significant bits for further precision [196].
- **Bit Significance-Aware OR Compression:** Replaces lower-bit additions with OR gates to eliminate long carry chains while using approximate compressors for higher bit positions [190,194]. This approach enables scalable $2n \times 2n$ multipliers from modular $n \times n$ blocks, with the most significant block kept accurate (Figures 5.5a–5.5b).
- **Compressor and adder simplification** optimizes the reduction tree by replacing accurate HAs, FAs, and compressors with simplified variants to reduce depth, delay, and power [187,188,197–199]. Approximate 4–2 compressors are especially effective for balanced accuracy and efficiency [184,185]. Additional savings can be achieved through alternative logic reuse, using gates such as NOT AND (NAND) instead of AND for PP generation, lowering transistor count and power [198].

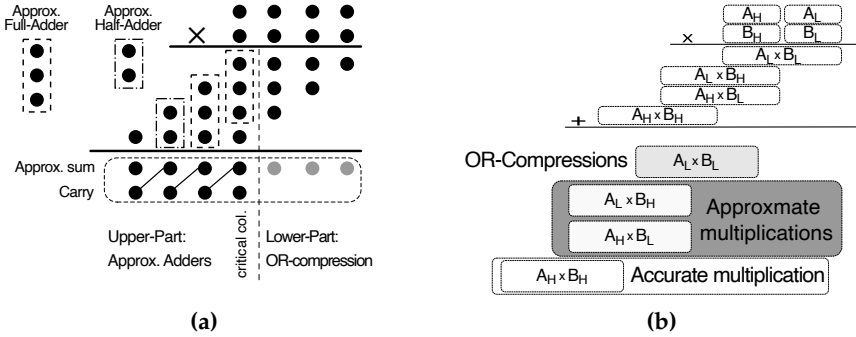


Figure 5.5: (a) Lower-Part-OR 4x4 AM; (b) scalable Lower-Part-OR structure for larger AMs [190].

Modular and Scalable Designs construct larger multipliers from approximate subblocks, enhancing reusability and tunability:

- **Modular approach:** Employs small approximate subblocks (e.g., 2×2 or 3×3) to construct higher-order AMs [200,201]. For instance, [200] uses two 3×3 and one 2×2 blocks in an 8×8 design tailored to Deep Neural Network (DNN) workloads, while [201] employs 4×2 and 4×4 modules

for flexible construction of power-of-two larger multipliers (Figures 5.6a and 5.6b).

- **Static Segmentation:** selects a fixed segment from each operand (typically the m Most Significant Bits (MSBs) out of n bits) for multiplication with smaller units, reducing logic while preserving partial precision [202].
- **Basic Block Modification:** uses modified Karnaugh map (K-map) or logic of small multipliers/compressors to produce approximate yet compact logic [197].

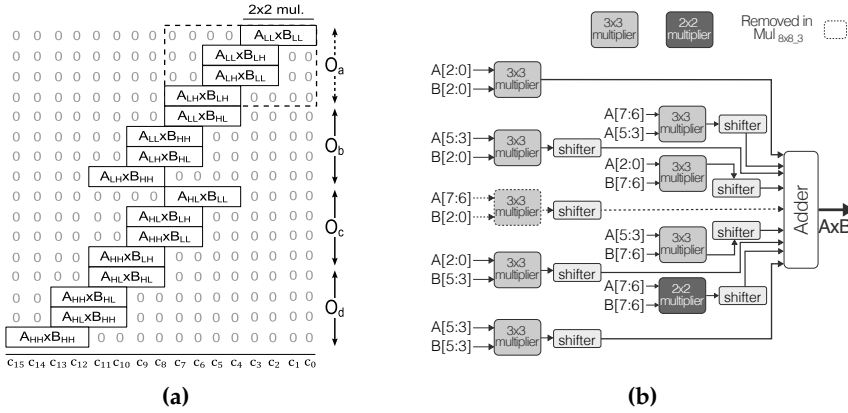


Figure 5.6: Modular 8-bit AM design: (a) using 2×2 and 3×3 blocks [200]; (b) using 2-bit subblocks [201].

Encoding and algorithmic approaches reduce logic overhead by optimizing input representation. A common example is Booth encoding, which can be approximated by simplifying the encoder's K-map to lower PP generation complexity. Hybrid schemes often apply radix-4 encoding to MSBs and radix-2 to LSBs [203,204]. **Automated and application-aware methods** further exploit input statistics or workload characteristics to generate AMs for a desired balance between accuracy and energy efficiency (Figure 5.7a) [180].

Architecture-aware implementations exploit platform-specific features to improve efficiency. FPGA-based designs leverage Look-Up Tables (LUTs) and reconfigurable logic to implement AMs optimized for hardware deployment (Figure 5.7b) [186], while VOS runs multipliers below nominal voltage, introducing limited timing-related errors to achieve substantial energy savings [205]. Building on these ideas, **hybrid techniques** combine multiple strategies,

such as truncation, approximate compressors, and modular construction, to balance power, area, and accuracy for specific applications [179,203].

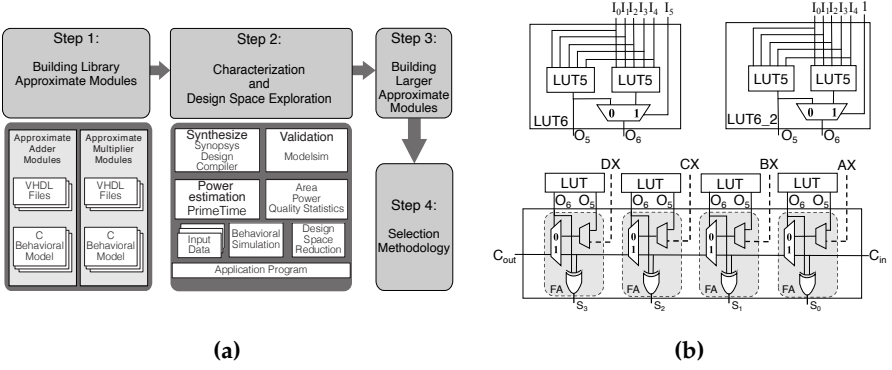


Figure 5.7: (a) Automated synthesis flow for AM design [206]; (b) LUT-based architecture for FPGA implementation [186].

5.3.1 ERROR COMPENSATION MODULES FOR AM DESIGN

To recover part of the lost precision, an Error Compensation Module (ECM), also known as an Error Recovery Module (ERM), corrects errors, particularly those arising during PP accumulation [184,188,189]. ECMs can be integrated at various levels of a multiplier architecture, either embedded within approximate compressors [185,188] or placed after truncated and simplified PP reduction units [187,207]. Their inclusion, however, depends on the specific approximation technique and is not universally required. Design efficiency and hardware overhead are critical, requiring careful architectural planning to ensure that compensation does not negate the gains from approximation. Several compensation methods have been proposed, each tailored to the type and location of approximation, including:

- **Correcting Bit Injection:** Negative errors produced by approximate compressors can be mitigated by inserting a constant logic '1' at a higher bit position, typically near the MSB, to offset the inherent negative bias. The ECM monitors input patterns and disables the correction when it would lead to overcompensation, i.e., when the approximate output is already accurate or positively biased [184].
- **Carry-In Based Error Compensation:** Correction signals from the ECMs serve as carry inputs to subsequent stages, enabling effective error mitigation with minimal hardware overhead. Error detection logic may

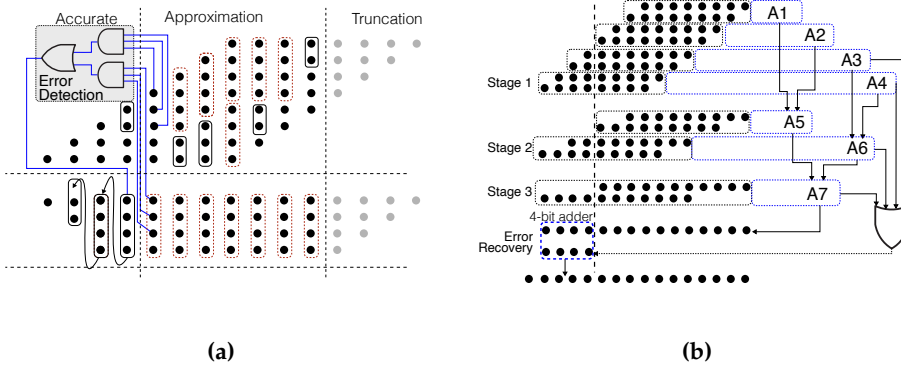


Figure 5.8: (a) ERM design combining exact and approximate compressors [185]; (b) OR-based partial error recovery using 4-MSBs of the error vector [199].

combine input bits or error flags using OR gates to generate a carry-in for the next column (Figure 5.8a) [185]. Some designs also derive a correction bit from the approximate region and inject it as a carry-in to the least significant column of the accurate region, achieving coordinated error correction across regions [187].

- **Configurable Partial Error Recovery:** Error signals generated during PP accumulation using approximate units are computed in parallel with the approximate sum. The accumulated error, estimated using OR-based logic due to the sparsity of 1s in the error vector, is partially recovered by adding a configurable number of its MSBs to the output through an accurate adder (Figure 5.8b) [199].
- **Consolidated Error Correction (CEC):** The CEC corrects the total accumulated output error instead of individual components by exploiting predictable error patterns in cascaded approximate adders. A predefined offset is added to the final result, effectively reducing overall error with minimal hardware and energy overhead [168,208].
- **Truncation Compensation:** In truncated multipliers, omitting lower bits introduces systematic errors that can be mitigated using carry estimation techniques based on Booth encoder outputs, simplified sorting networks, or K-map-derived low-overhead logic. One effective method generates compensation carries without referencing truncated columns by simplifying the K-map of the first non-truncated column, producing approximate carries that efficiently correct the output [164,182].

- **Static Segmented Multiplier (SSM) Correction:** SSMs approximate an $n \times n$ multiplication using smaller $m \times m$ units that operate on a selected portion of the multiplicand, with precision defined by the segment size. The input is divided into fixed segments, and the segment containing the leading one is used for computation. A correction term is added to the most significant segment result only when both operands have non-zero leading segments [202].

5.4 ACCURACY AND QUALITY METRICS

Accuracy and quality evaluation is central to the design of AMs. Metrics used for this purpose fall into two broad groups: *application-independent*, which quantify intrinsic error properties irrespective of workload, and *application-specific*, which measure whether errors remain tolerable in real tasks.

5.4.1 APPLICATION-INDEPENDENT METRICS

Application-independent metrics, first formalized in [209] and refined in later works, characterize the raw error behavior of AMs. They are typically grouped into four categories:

- **Distance-based metrics** quantify the deviation of approximate results from exact ones [164,177,179,187,190]. *Error Distance (ED)* measures absolute deviation, with *Mean Error Distance (MED)* giving its average across all inputs. *Normalized Mean Error Distance (NMED)* scales MED by the maximum possible output, making it largely invariant to word length and suitable for cross-design comparison. Relative deviation is captured by *Relative Error Distance (RED)*, with *Mean Relative Error Distance (MRED)* as the average across all outputs, less sensitive to input distribution and typically decreasing with AM size, and *Maximum Relative Error Distance (MRE)* as the maximum observed. Together, these metrics distinguish small absolute errors in large results from severe relative errors in small ones.
- **Error-rate metrics** measure how often errors occur [202,210,211]. *Accurate Output Count (AOC)* gives the number of correct results, while *Error Rate (ER)* and its complement, *Pass Rate (PR)*, report the frequency of incorrect or correct outputs. *Probability of Relative Error Distance (PRED)* estimates the probability that RED exceeds a threshold (e.g., 2%), useful for bounded-error applications.
- **Peak-deviation metrics** capture the most extreme cases, which may affect system reliability [183,202]. *Maximum Absolute Error (MAE)* is the

largest deviation, and $NmaxED$ normalizes it by the maximum correct output.

- **Statistical metrics** provide distribution-level insight [164, 184, 193, 194, 202]. *Mean Square Error (MSE)* emphasizes larger deviations, while *Root Mean Square Error (RMSE)* calculates the square root of MSE, offering an overall measure of error magnitude. *Variance of Error Distance (VarED)* indicates whether errors are predictable or highly variable. *Number of Effective Bits (NoEB)* estimates how many output bits remain consistently correct, offering a bit-level view of accuracy.

5.4.2 APPLICATION-SPECIFIC METRICS

Application-specific metrics measure the acceptability of errors in specific tasks, including:

- **Peak Signal-to-Noise Ratio (PSNR):** Widely used in image and video processing, PSNR relates the maximum signal to noise from approximation, indicating overall image quality (Figure 5.9) [194, 197, 211].
- **Structural Similarity Index Measure (SSIM):** SSIM and its mean version, Mean Structural Similarity Index Measure (MSSIM), measure perceptual similarity (Figure 5.9), capturing structural fidelity rather than raw pixel differences [168, 198, 211].
- **DNN Accuracy Loss (DAL):** In ML, DAL reports accuracy degradation when AMs replace exact ones [200].



Figure 5.9: PSNR and MSSIM evaluation of image sharpening using one proposed AMs for operational classes $m = 1$ to $m = 3$ [211].

A comprehensive assessment of AMs therefore requires both perspectives. Application-independent metrics define theoretical error bounds, while application-specific metrics reveal whether those bounds impact real-world functionality. Balancing the two allows principled design choices that exploit error tolerance where feasible and preserve accuracy where required.

5.4.3 ERROR ANALYSIS METHODOLOGIES

A comprehensive evaluation of AMs requires diverse methodologies, since their effects manifest across multiple abstraction levels, from raw arithmetic correctness to application-level quality of service. No single method provides a complete picture. Low-level simulations reveal intrinsic error behavior, statistical analyses capture error patterns, and application-level tests demonstrate practical impact on real workloads. Together, these methods form a comprehensive framework for quantifying the accuracy, efficiency, and resilience of approximate designs. As shown in Figure 5.10, the most widely used methodologies include:

- **Behavioral Modeling:** High-level C/C++ or MATLAB functional models enable fast exhaustive or randomized testing [179, 194, 198, 206, 207].
- **Exhaustive Simulation:** All input combinations are evaluated, feasible only for small designs (e.g., 8-bit AMs with 65536 cases) [190, 196].
- **Error Distribution Analysis:** Examines statistical error distributions to identify local patterns and quantify exact or near-exact results [194].
- **Application-Level Testing:** Integration into workloads such as image filtering (Gaussian blur, Sobel), ML (K-means, Support Vector Machine (SVM), DNNs), or DSP (FFT, Finite Impulse Response (FIR)), with quality measured via domain-specific metrics [179, 190, 198, 206].
- **Gate-Level Simulation:** Post-synthesis verification in tools such as ModelSim, with power estimation and Static Timing Analysis (STA) in Synopsys PrimeTime [206].

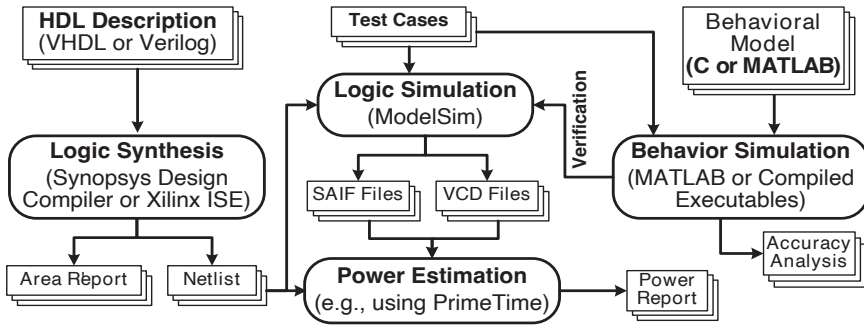


Figure 5.10: Evaluation flow for an AC unit: from application-level quality assessment to gate-level analysis [168].

5.5 THESIS CONTRIBUTION

This chapter presents a survey of different techniques in AC, with a focus on arithmetic units and multiplier design strategies. Building on this foundation, **Paper V** contributes a scalable AM architecture based on priority-select PP removal. In this design, stacked PPs in each column are selectively pruned so that more significant terms are preserved while less critical ones are discarded. By embedding approximation directly into PP generation, the approach reduces the reliance on HAs and FAs in the accumulation tree, thereby lowering power, area, and delay.

Scalability was a central focus, as this thesis emphasizes the importance of exploiting the benefits of both highly energy-efficient dedicated accelerators and highly flexible programmable MCUs for adapting to fast-evolving edge AI applications. The proposed AM aligns with this strategy by offering efficiency gains comparable to dedicated hardware while maintaining scalability needed for integration into programmable platforms.

In 8-bit implementations, the AM achieves up to 60 % area and 62 % power savings, with a 92 % improvement in Power-Area-Delay product (PADP) relative to a Wallace baseline, while maintaining bounded error metrics. Application-level evaluations on CNN inference and image processing confirm negligible accuracy loss, demonstrating suitability for error-tolerant embedded AI workloads.

Rebooting Computing: Beyond von Neumann Architectures



s data-intensive applications proliferate and technology scaling decelerates, the von Neumann architecture, characterized by a rigid separation between processing and memory, has increasingly become a performance and energy bottleneck. This chapter investigates the shift toward memory-centric computing paradigms, particularly IMC and NMC. This chapter provides a comprehensive discussion of architectural classifications, implementation techniques, enabling memory technologies, and system-level tradeoffs, with an emphasis on implications for AI and EC.

6.1 FROM PROCESSOR-CENTRIC TO DATA-CENTRIC ARCHITECTURES

For decades, the von Neumann architecture has been the dominant paradigm in computer system design, with memory and compute units connected by bandwidth-limited buses [212, 213]. While this data processing scheme has enabled the rapid advancement of general-purpose computing, a fundamental limitation known as the *von Neumann bottleneck* or *memory wall* remains [39, 214–216]. Moving data between processor and memory often dominates overall system cost, with off-chip memory access consuming up to $200\times$ more energy than computation [47, 217].

Furthermore, processors may spend over 60% of execution cycles stalled while waiting for data [43], leading to severe performance and energy inefficiencies that are especially critical in data-intensive workloads with high memory demand and tight energy budgets [43, 46, 218].

Adding to the problem is the slowdown of Moore’s law and the end of Dennard scaling, which have significantly diminished the traditional gains from transistor scaling [46, 212]. As a result, continued performance scaling

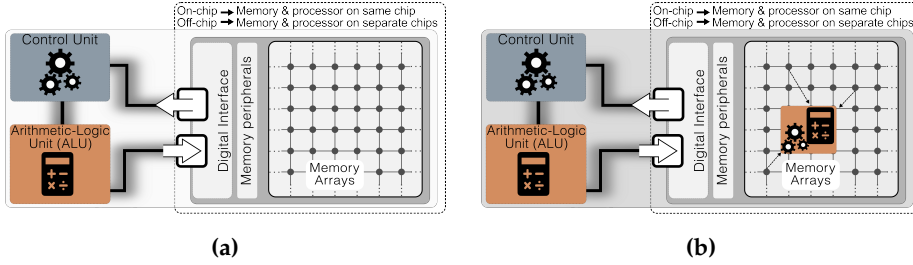


Figure 6.1: (a) Processor-centric system: data shuttled to the CPU; (b) memory-centric system: compute placed near/in memory to cut data movement [219].

through conventional processor-centric architectures (Figure 6.1a) has become increasingly unsustainable.

Over the years, various mitigation strategies have been explored, including the development of faster data buses, multi-level cache hierarchies, and hardware accelerators. However, these measures only alleviate the architectural inefficiency rather than resolving it. For example, cache hierarchies are ineffective for workloads with poor data locality and irregular access patterns, while accelerators, despite offering high computational throughput for a specific workload, are often limited by off-chip memory bandwidth and data transfer overheads [214, 219]. These limitations have motivated a shift toward *data-centric* or *memory-centric* architectures, which reduce data movement by placing computation closer to where the data resides (Figure 6.1b) [215, 220]. Embedding processing capabilities near- or in-memory structures promises substantial gains in both energy efficiency and throughput.

This transition is strongly driven by modern AI workloads, particularly DNNs used in vision, speech, autonomous systems, and scientific computing [221, 222]. These applications repeatedly access large amounts of data and are increasingly constrained by bandwidth and latency rather than raw compute capability. The exponential growth in ML model size and complexity, illustrated in Figure 6.2, has dramatically increased the number of operations and memory transactions required per inference or training cycle, further exposing the limitations of processor-centric systems.

This challenge is even more pronounced at the edge, where data is generated and needs to be processed locally under strict latency and energy constraints. Memory-centric computing is particularly well suited to edge AI, since reducing data movement enables faster and more energy-efficient operation. In practice, edge AI already supports predictive maintenance in industrial systems through direct sensor analysis, enables smart grid control and load

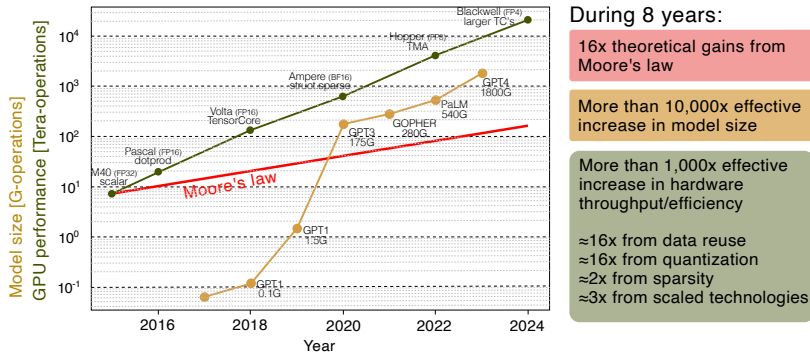


Figure 6.2: Decade-long trends in ML model size, GPU performance, and Moore's law [223].

balancing in energy networks [224], and even compact language models are now deployed to run locally, providing privacy-preserving services across distributed devices [225].

Building reliable and efficient edge AI systems, however, remains challenging due to limited resources, real-time requirements, and security concerns [226, 227]. Addressing these constraints requires frameworks that map AI models to resource-limited platforms. Examples include AIeS and NeuMap, which provide compact toolchains for embedded and neuromorphic systems [228, 229], bridging the gap between algorithm design and hardware deployment under real-world conditions.

6.1.1 HISTORICAL CONTEXT AND RESURGENCE

The concept of memory-centric computing, dates back to early explorations of the idea of fusing memory and logic in the 1960s–1970s [219, 220]. In 1978, John Backus of International Business Machines Corporation (IBM) Research challenged the dominance of sequential, processor-centric programming models in his seminal article, *“Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs”* [230]. In the 1990s, prototypes such as Vector IRAM (VIRAM) integrated vector processors with eDRAM to exploit data-level parallelism [231], but their adoption was constrained by fabrication incompatibilities between logic and memory processes.

The resurgence of memory-centric computing is driven by advances in 2.5D and 3D integration, in which Through-Silicon Vias (TSVs) and interposers enable heterogeneous stacking of memory and logic dies, with reduced communication overhead [46]. Technologies such as Hybrid Memory Cube

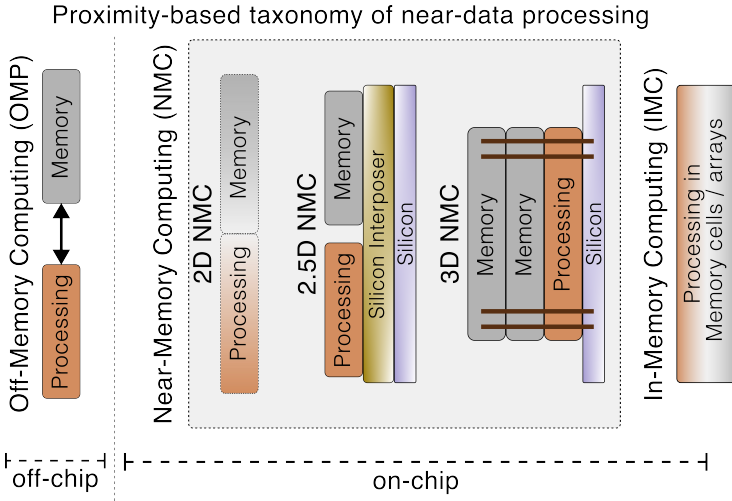


Figure 6.3: Proximity-based taxonomy: off-memory processing (OMP), Near-Memory Computing (NMC), and In-Memory Computing (IMC) [46].

(HMC) and High Bandwidth Memory (HBM) established the feasibility of near-memory computing [214], while Intel’s Optane DC Persistent Memory DIMMs marked a milestone in commercial deployment by the byte-addressable persistent main memory [232].

More recently, commercial solutions have demonstrated how these concepts can be applied in practice. UPMEM’s NDP DRAM embeds lightweight processors inside DRAM banks to offload simple compute kernels, Samsung’s HBM-Processing-in-memory (PIM) augments high-bandwidth memory with embedded logic for AI acceleration [233], and SK Hynix’s AiM integrates arithmetic units into memory stacks for efficient inference [234]. At the same time, SRAM-based prototypes such as Vecim [235] and BLADE [236] show how in-memory computing can perform logic operations directly within arrays, enabling energy-efficient dataflows. Emerging NVMs (e.g., ReRAM, PCM, MRAM) further broaden the design space by bridging the DRAM–storage gap and enabling in-situ operations [215,237].

6.1.2 TAXONOMY AND IMPLEMENTATION TECHNIQUES

Memory-centric computing spans two main paradigms that differ in the physical proximity of compute to data:

- **NMC:** Also referred to as Near-Memory Processing (NMP), this approach places compute units adjacent to memory arrays (on-die, in package, or in stacked dies) without modifying the memory cells. NMC

reduces data movement and alleviates memory bandwidth bottlenecks while preserving compatibility with conventional memory designs. Commercial realizations include Samsung's HBM-PIM.

- **IMC:** Also called Compute-in-Memory (CIM), in-situ computing, or PIM, performs computation directly within the memory array by exploiting the same structures used for storage. IMC eliminates data transfers entirely and is particularly well suited for parallel operations such as Multiply-Accumulate (MAC) and vector-matrix multiplications. SRAM-based platforms such as Vecim and BLADE demonstrate practical realizations of this approach.

As illustrated in Figure 6.3, these paradigms can be positioned within a broader proximity-based classification [46]:

- **Off-Memory Processing (OMP):** Processor and memory are separate chips linked by costly off-chip I/O. Off-chip memory may use NMC to return partially processed results instead of bulk raw data.
- **NMC:** Compute logic is tightly integrated with memory through advanced packaging and stacking, significantly reducing communication overhead.
- **IMC:** Computation is executed directly within the memory cells, removing the need for data transfer entirely.

This taxonomy and existing implementations demonstrate the evolution of memory-centric computing from concept to practice, with each class presenting unique tradeoffs in scalability, energy efficiency, and workload suitability. While NMC and IMC are often collectively referred to as NDP, we maintain this distinction to highlight key differences in integration complexity, data locality, and supported workloads. This classification provides a concrete framework for evaluating tradeoffs across architectural designs in terms of scalability, performance, and energy efficiency.

6.1.3 ARCHITECTURAL PARADIGMS

6.1.3.1 Near-Memory Computing

NMC places compute units next to memory arrays (same chip, interposer, or die stack) without modifying the memory cells, maintaining compatibility with existing memory architectures [47,219]. By executing computations close to memory and transmitting only computation results to the host processor, NMC reduces memory bandwidth pressure and improves energy efficiency, especially for memory-bound tasks like DNN inference, graph analytics, and embedded workloads [43,215]. Figure 6.4a depicts two NMC variants integrated into a RISC-V MCU for TinyML: NM-Caesar (area-critical) and NM-Carus (performance-oriented) [238]. Figure 6.4b presents a behavioral

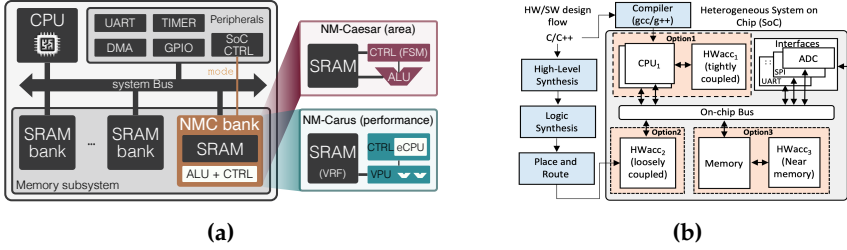


Figure 6.4: (a) NMC-enhanced RISC-V-based MCU variants (area (NM-Caesar) vs. performance (NM-Carus)) [238]; (b) HLS-based heterogeneous SoCs with tightly coupled accelerators and NMC [239].

High-Level Synthesis (HLS) prototype for heterogeneous NMC SoCs, which speeds prototyping but limits performance and energy efficiency due to the lack of low-level optimizations [239].

Packaging technologies such as 2.5D interposers and 3D-stacked memories (e.g., HMC, HBM) enable tight integration of logic and memory. Notable systems like TESSERACT embed processing cores in each HMC vault to accelerate graph workloads [47], while architectures such as HeNCoG utilize MRAM-based near-memory cores for Graph Convolutional Networks (GCNs) aggregation [240].

A subset of NMC integrates compute engines into storage devices, as in CompStor, and Deepstore, which embed ARM processors or FPGAs in SSDs to perform key-value lookups, encoding, and Hyper-Dimensional Computing (HDC) [215, 241, 242].

6.1.3.2 In-Memory Computing

IMC pushes computation into the memory arrays by reconfiguring standard memory cells or exploiting emerging memory devices with intrinsic compute capabilities [46, 216, 219]. Eliminating data transfers between separate compute and memory units yields high energy efficiency and throughput for massively parallel operations.

Figure 6.5 illustrates the application landscape of IMC, categorizing domains by computational precision, data access requirements, and complexity [219]. The spectrum includes:

- low-precision, highly parallel tasks such as deep learning inference, filtering, and compressed sensing,
- moderate-precision workloads including signal processing and optimization,

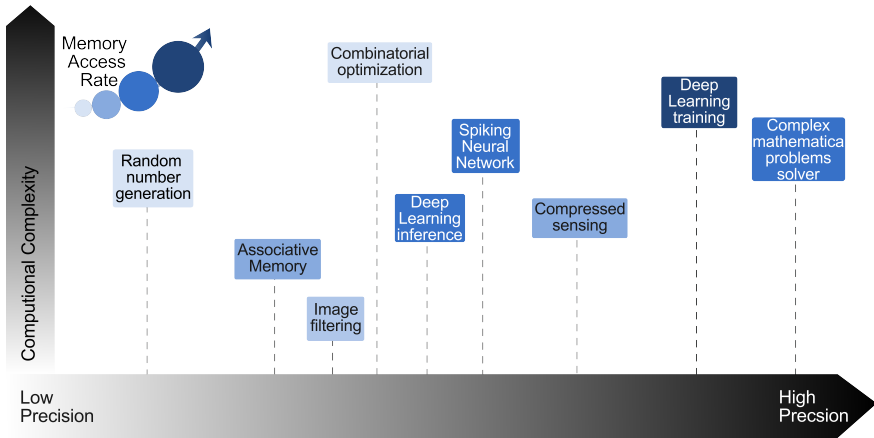


Figure 6.5: Application landscape for IMC vs. required precision, complexity, and memory access demands [219].

- high-precision applications such as scientific computing and linear solvers.

As shown in Figure 6.6, IMC implementations are typically grouped into two classes by implementation approach:

- **Digital In-Memory Computing (DIMC)** embeds deterministic logic within or near memory arrays, providing robustness to PVT variations, noise-free computation, and flexible bit-serial or bit-parallel dataflows [243, 244]. Examples include X-SRAM [245], SIMD RAM [246], and Colonnade for CNN inference [247].
- **Analog In-Memory Computing (AIMC)** leverages device properties such as current summation or charge sharing to perform massively parallel MAC operations, delivering high energy efficiency and throughput for workload-specific tasks [244, 248]. Its drawbacks include sensitivity to noise, device mismatch, and non-idealities, along with the overheads of Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs), that limit precision and scalability [249, 250].

Beyond these two categories, *Hybrid IMC (HIMC)* architectures combine analog computation with digital control to balance precision and efficiency, such as WDVR-RAM for accelerating Binary Neural Networks (BNNs) or digitally controlled IMC implementations [251, 252].

6.1.3.3 Analog vs. Digital Tradeoffs

AIMC performs arithmetic directly within memory cells by accumulating bit-wise multiplication results on the BL voltage [244]. By activating multiple

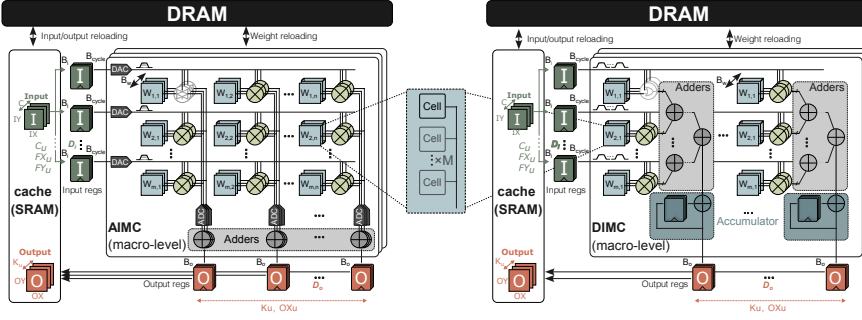


Figure 6.6: System-level view of SRAM-based AIMC (left) and DIMC (right) for CNN edge deployment [243].

memory words simultaneously, it achieves high parallelism and energy efficiency, as all BCs in an array contribute to a single MAC operation. Implementations exploit device-level effects such as voltage division [253], current discharge [254], charge sharing [249], and capacitive coupling [255]. However, these benefits come with several limitations:

- **Accuracy:** Susceptible to reduced Signal-to-Noise Ratio (SNR) from circuit noise, device mismatch, and ADC quantization, often degrading accuracy relative to digital baselines. Limiting the number of activated rows can improve SNR but reduces parallelism and efficiency [243,248].
- **Variability:** Sensitive to PVT variations and device non-idealities such as resistance variation, limited resistance ratios, and read disturbance. Calibration is typically required to mitigate these effects [216,237].
- **Peripheral Overhead:** ADCs and Time-to-Digital Converters (TDCs) are often power-hungry, though ADC-free architectures have been explored to reduce this cost [243].
- **Workload Suitability:** Well suited for bitwise and pointwise operations ($1 \times 1 \times N$) with high spatial unrolling, but less effective for diverse or irregular workloads [243].
- **Scalability:** Limited by signal integrity, parasitic effects, and the increasing resolution demand of ADCs, which limit array size and throughput. Calibration and synchronization overheads further challenge large-scale deployment [216,243].

In contrast, DIMC executes digital logic and arithmetic within or near memory arrays, offering deterministic accuracy and predictable scalability while avoiding the calibration, variability, and precision issues inherent in AIMC. While cell-level concurrency and raw computational density at lower bit precision are comparatively limited, DIMC maintains consistent performance

across workloads and often outperforms AIMC in depthwise or low-parallelism layers where analog inefficiencies dominate [216, 243, 244].

6.1.4 MEMORY TECHNOLOGY ENABLERS

The effectiveness of memory-centric computing depends on the capabilities of the underlying memory technologies, ranging from CMOS-compatible volatile memories to emerging NVM with distinct analog and digital behavior (Figure 6.7).

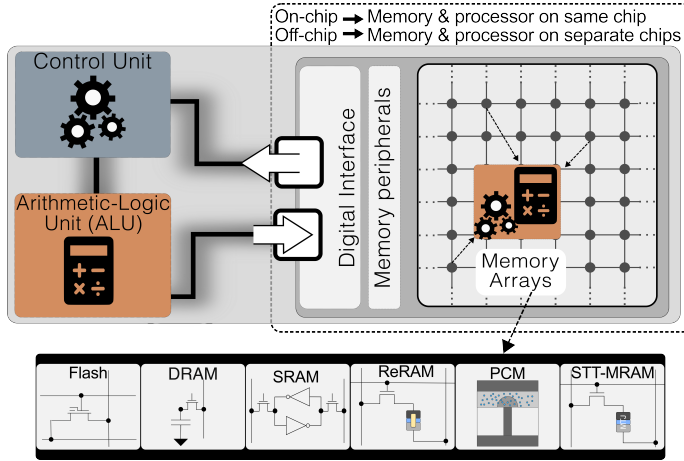


Figure 6.7: Memory technologies for memory-centric computing: volatile (SRAM, DRAM, 3D DRAM) and emerging NVM (ReRAM, PCM, MRAM) [219].

6.1.4.1 Volatile Memories

SRAM is the most widely adopted technology for IMC due to its high endurance (exceeding 10^{16} write cycles), low latency, large on/off ratios, and full CMOS compatibility. Its robustness and fast access further reinforce its suitability for compute integration [47]. In-situ arithmetic and logic operations can be realized by reconfiguring BLs or SAs [256]. Prototypes such as BLADE and X-SRAM exemplify in-cache and BC-level computation in standard 6T and extended SRAM arrays [236, 245].

DRAM offers greater density and capacity than SRAM, making it attractive for large-scale memory-centric computing. IMC/NMC in DRAM is typically enabled by charge sharing, multi-row activation, or BL precharge manipulation to implement bulk Boolean logic operations [257]. Commodity DRAM-based compute platforms like SIMDRAM demonstrate scalable in-place processing

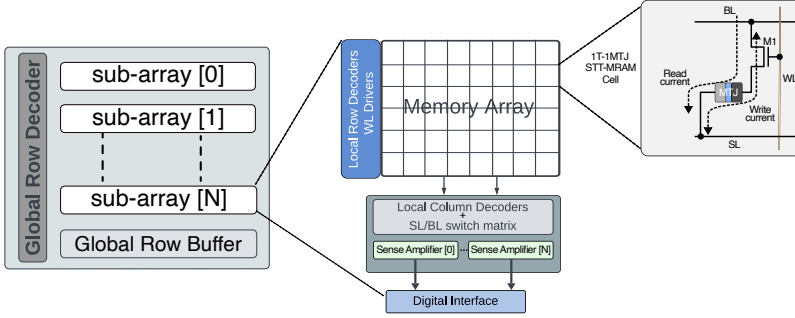


Figure 6.8: Overview of STT-MRAM-based IMC architecture [260].

without major architectural changes [246]. Adjusting timing parameters further extends DRAM’s capability for compute. However, leakage and frequent refresh requirements complicate IMC implementations [47,213]. Furthermore, even in optimized configurations, off-chip to on-chip transfers remains a bottleneck, limiting overall system efficiency.

3D DRAM technologies such as HMC and HBM integrate logic in the base die, enabling NMC without altering DRAM cells. Architectures including TESSERACT and Samsung’s HBM-PIM leverage this integration to provide high internal bandwidth and efficient NMC [46].

6.1.4.2 Non-Volatile Memories

ReRAM and **PCM** enable analog computation through conductance modulation in crossbar arrays, making them attractive for AIMC. Their limitations include resistance drift, finite endurance, and device variability [237]. **MRAM** and **STT-MRAM** support fast, energy-efficient digital operations and are compatible with CMOS process (Figure 6.8). Designs such as AMS-IMC and TS-NMC exploit its low leakage and fast reads for hybrid and NMC accelerators [213]. **FeFETs** combine storage and logic at low operating voltages, making them promising for neuromorphic architectures and Content-Addressable Memories (CAMs), though they face key limitations such as inter-gate coupling, variability in polarization switching, and endurance-voltage tradeoffs, challenging reliability and large-scale deployment [258,259]. **Read-Only Memory (ROM)** has also been revisited for IMC, especially in workloads with static weight matrices. ROM-based IMC is highly compact and energy-efficient, although it requires algorithm-level co-design due to its limited reconfigurability [216].

6.1.5 ARCHITECTURAL VARIATIONS

Architectural implementations of IMC and NMC differ not only in compute placement but also in the types of processing units employed, the integration strategies used, and the execution models they target. Key categories of *processing units* include:

- **Programmable platforms:** RISC-V cores, GPUs, and FPGAs offer flexibility and a high-degree of programmability [215,261].
- **Fixed-function accelerators:** ASIC-based designs (e.g., TS-CIM, PUMA, SIMDRAM) target specific inference or logic tasks for high-performance and energy efficiency [244,262].
- **Hybrid units:** Architectures like DIANA and Vecim combine analog compute cores for MACs with digital logic for control or activation functions [216,235].

In addition, several *integration and data movement optimization strategies* are central to improving system efficiency:

- **2D, 2.5D, and 3D integration:** TSV-based stacking (e.g., HBM, HMC) and chiplet-based integration enable vertical and lateral proximity of compute and memory, alleviating bandwidth bottlenecks and reducing latency [218,263].
- **Software-hardware co-design:** Compiler frameworks (e.g., SongC-NMC) and runtime schedulers manage task placement, coherence, and data mapping, enabling scalable deployment [264].

6.1.6 DATAFLOW STRATEGIES FOR ACCELERATING NN WORKLOADS

Dataflow selection is a critical determinant of throughput and energy efficiency in DNN workloads, particularly CNNs, which underpin a wide range of applications and have been the main target of architectural optimization in recent years. Figure 6.9a illustrates the convolutional layer processing flow and the data movement among operands in a typical CNN layer. Since data movement often dominates energy cost, modern dataflows aim to maximize data reuse within the on-chip hierarchy while reducing costly off-chip transfers [69,217]. Dataflows are commonly categorized by which operand (input activations (Input Feature Maps (Ifmaps), filter weights, Partial Sums (Psums), and Output Feature Maps (Ofmaps)) remains locally stationary [69]:

- **Weight-Stationary (WS):** Weights are fetched once and kept in Processing Element (PE)'s register file, optimizing weight reuse and MAC utilization [217]. Limited on-chip capacity, however, increases DRAM traffic and reduces Ofmap reuse (Figure 6.9b).
- **Output-Stationary (OS):** Psums remain in the PE until the corresponding output is fully computed, reducing accumulation energy and making OS

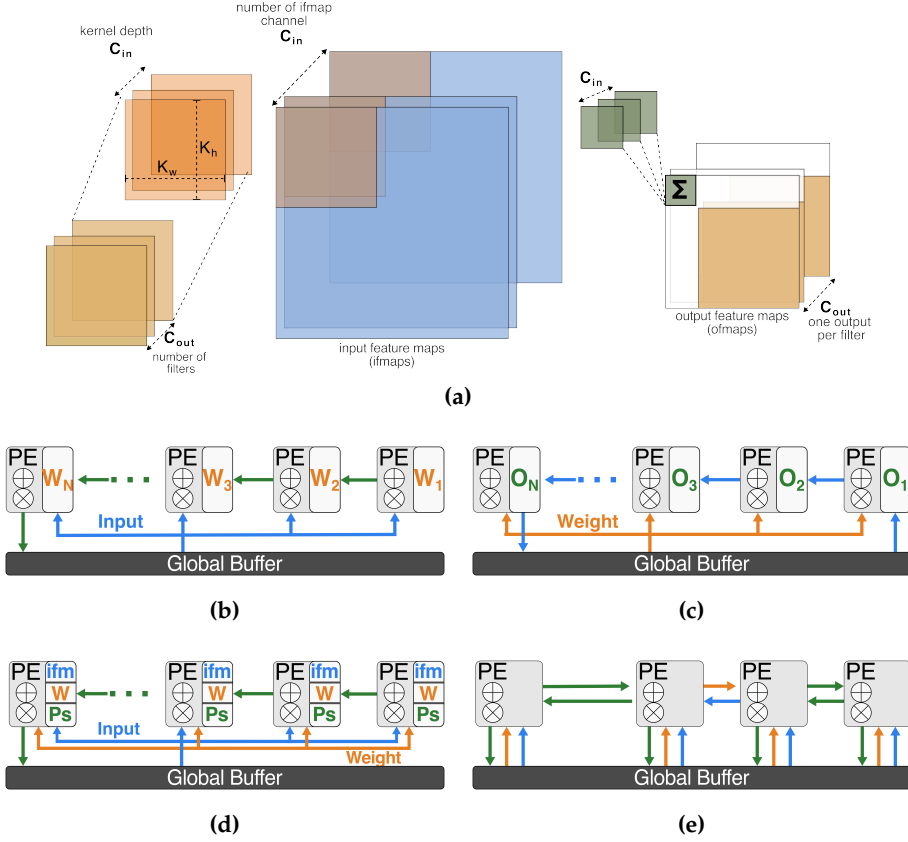


Figure 6.9: (a) Convolutional layer computation in a CNN; dataflows: (b) Weight-Stationary (WS); (c) Output-Stationary (OS); (d) Row-Stationary (RS); and (e) No-Local-Reuse (NLR) [265].

effective for layers with small output dimensions, such as in later stages of VGG-11 [266,267]. Variants like SOC-MOP, MOC-MOP, and MOC-SOP adjust the degree of parallelism based on the number of Ofmap planes and channels processed [217]. OS can also achieve relatively low DRAM traffic, indicating that optimized on-chip memory utilization could further improve its energy efficiency, which aligns well with NMC and IMC architectures (Figure 6.9c).

- **Row-Stationary (RS):** Introduced in Eyeriss, RS reduces movement for all operands by decomposing convolution into one-dimensional primitives whose row pairs remain stationary in PEs and reusing ifmaps/weights in local scratchpads, reducing access to DRAM [69]. RS typically offers

improved energy efficiency per convolution than WS or OS, though its peak throughput is also generally lower (Figure 6.9d).

- **No-Local-Reuse (NLR):** Keeps only Arithmetic Logic Unit (ALU) datapaths inside the PE while all data (ifmaps, weights, Psums) reside in the global buffer [217, 267]. Every cycle, weights are unicast and ifmaps multicast to the appropriate multipliers. Because PE count is independent of layer shape, NLR achieves very high raw throughput on large feature maps, but frequent global-buffer access leads to higher energy consumption (Figure 6.9e).

In addition to the established dataflows, several refinements have been introduced to enhance efficiency and adaptability. Enhanced Output-Stationary (EOS) adds dedicated activation register files to exploit activation reuse and cut SRAM traffic [266]. Hybrid Stationary (HS), as in FlexSpIM, dynamically selects WS or OS per layer to maximize operand stationarity [268]. Point-wise Stationary (RS-IS / WS-OS) combines outer and inner flows to efficiently map BNN layers [269]. Enhanced Weight-Stationary (EWS) incorporates a weight register file per PE to extend WS reuse inside systolic arrays [270].

Furthermore, the effectiveness of any dataflow is shaped by a set of design considerations, as its choice dictates how well memory traffic and computation overlap, including:

- **Operand reuse and sparsity:** Improves energy efficiency by higher local reuse or the ability to skip zero computations [271].
- **Bandwidth and utilization:** Limited data reuse requires higher Network-on-Chip (NoC) bandwidth to avoid stalls and PE underutilization [271].
- **Mapping flexibility:** fixed choice of dataflow risk PE underutilization when layer dimensions shrink, whereas reconfigurable mappings sustain utilization across diverse workloads [271].
- **Heterogeneous pairing:** Architectures such as Quartet combine optimal execution order with RS to improve efficiency [272].
- **Adaptability:** Emerging DNNs motivate reconfigurable dataflows such as FlexNPU [273].

In summary, WS provides the highest MAC utilization, OS reduces accumulation cost and remains competitive with RS in memory traffic, RS balances all operands efficiently, and NLR maximizes throughput though with higher energy overhead. To maintain both performance and energy efficiency, modern NMC/IMC designs increasingly combine or switch among these flows dynamically.

6.1.7 REPRESENTATIVE IMPLEMENTATIONS

Table 6.1 compares representative IMC and NMC implementations, illustrating diverse strategies and tradeoffs.

Table 6.1: Representative memory-centric systems: types, features, advantages, and limitations.

System	Type	Key Features / Applications	Advantages	Limitations
TESSERACT [47]	NMC (HMC)	Parallel graph processing using vault-local cores in 3D-stacked HMC	HMC vault-level parallelism	(i) Complex memory management, (ii) Limited applicability (not general-purpose)
IMPICA [214]	NMC (3D tech.)	Pointer chasing acceleration	Handles irregular memory access	Limited applicability (pointer-chasing)
HeNCoG [240]	NMC (MRAM)	(i) Energy-efficient GCN inference (ii) MAC fusion (iii) MAC reordering in MRAM-based NMC cores	(i) Reduced execution cycles and memory access via execution reordering	(i) Domain-specific, (ii) dependent on MRAM technology
SIMDRAM [246]	IMC (DRAM)	Bit-serial logic operations in commodity DRAM	Cost-efficient due to reuse of existing DRAM	(i) Limited to bit-serial logic, (ii) Low throughput
X-SRAM [245]	IMC (SRAM)	Boolean operations via multi-row activation	(i) Digital logic in SRAM, (ii) in-memory vector Boolean operations	(i) Limited to logic operations (ii) Not suitable for arithmetic-heavy workload
PUMA [262]	IMC (Memristor)	inference via analog crossbars + programmable digital control	(i) High energy efficiency, (ii) Scalable for large DNNs	(i) Complex peripheral circuits, (ii) Memristor write endurance issue
WDVR-RAM [274]	Hybrid IMC (SRAM)	Charge-domain architecture for AIoT workload	Binarized CNN in a wide dynamic voltage range	Limited applicability
NM-Caesar [238]	NMC (SRAM)	RISC-V + NMC for TinyML on embedded MCUs	(i) Programmable, (ii) Suitable for edge AI	(i) Limited throughput

NMC designs such as TESSERACT, IMPICA, HeNCoG, and NM-Caesar integrate compute near memories to accelerate data-intensive workloads from graph analytics to TinyML. They reduce data movement and improve efficiency but often rely on specific technologies (e.g., MRAM, 3D DRAM) and require custom runtimes. Wider adoption demands programmability and standardized software support. IMC prototypes such as SIMDRAM, X-SRAM, PUMA, WDVR-RAM, and PS-IMC embed compute within arrays to reduce data movement and improve efficiency. X-SRAM supports Boolean logic but is restricted to bitwise operations. WDVR-RAM is tailored to BNNs but lacks broader applicability. PUMA combines memristor crossbars with programmable ISAs but is limited by device variability and write latency. PS-IMC offers precision scaling at the cost of increased circuit complexity. IMC as a whole continues to face challenges in precision, reliability, and programmability, which call for advances in mixed-signal design, precision-scalable macros, and unified hardware-software integration. Together, these systems illustrate that no single approach dominates. The current landscape

is instead shaped by tradeoffs between flexibility, integration overhead, and application scope.

6.2 ENERGY, PERFORMANCE, AND SYSTEM-LEVEL TRADEOFFS

6.2.1 EVALUATION METRICS

Evaluating memory-centric architectures requires a comprehensive set of metrics that capture both hardware efficiency and system-level performance. Commonly reported figures include power consumption, throughput, latency, energy efficiency, and area utilization [223,248].

A central concept is *arithmetic intensity*, the ratio of computation to memory accesses, which indicates whether a workload is compute- or memory-bound [223]. The *roofline model* (Figure 6.10a) visualizes this relationship, mapping achievable performance against arithmetic intensity and showing whether throughput is constrained by peak compute or memory bandwidth [267,271]. This model helps identify performance bottlenecks and guide hardware-software co-optimization, which has been widely used in analyzing CNN accelerators to reveal their performance envelopes under specific architectural constraints [223,271]. However, as depicted in Figure 6.10b, the roofline curve is not fixed: architectural modifications or workload changes, such as memory hierarchy refinements or increased parallelism, can shift the operational bounds. Each step reflects the layered nature of optimization, where the ceiling of one configuration becomes the baseline for the next [271].

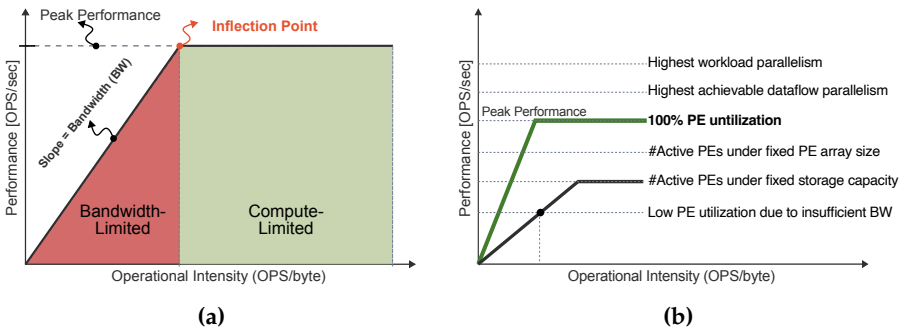


Figure 6.10: (a) Roofline model: performance vs. arithmetic intensity; (b) impact of architectural/workload changes on the roofline levels [271].

To compare performance and efficiency together, the *Energy-Delay Product* (EDP) is often used as a composite metric, capturing both efficiency and speed [275]. Modeling tools such as CACTI [261], DRAMSim3, and HSPICE are used to estimate latency, leakage, and area, while architectural simulators like gem5 enable system-level profiling. To complement these models, FPGA and ASIC prototypes remain essential for validating performance under real-world conditions [250,276,277]. Overall, the effectiveness of memory-centric architectures should be evaluated under realistic workloads, with emphasis on balancing energy, bandwidth, latency, and utilization [238,239,243].

6.2.2 THROUGHPUT AND ENERGY EFFICIENCY

6.2.2.1 Throughput

Performance is typically characterized using several complementary metrics. At the hardware level, raw throughput is measured in Operations per second (OPS), typically reported in Giga Operations per second (GOPS) or Tera Operations per second (TOPS), or in Giga Multiply-Accumulate Operations per second (GMACS) when focusing on MAC operations that dominate DNN workloads [69,223,271]. At the system level, performance is often expressed as task-level latency (e.g., inference time per image or sequence) or as relative speedup compared to CPU and GPU baselines. PE utilization, defined as the fraction of peak compute sustained under realistic workloads, is also a critical indicator [223]. These metrics align naturally with the roofline model: OPS or GMAC/s reflect the theoretical compute ceiling, while speedup, latency, and utilization expose how memory bandwidth and arithmetic intensity constrain practical performance [223]. Throughput can be further boosted by architectural modifications that increase arithmetic intensity, thereby improving compute utilization and enabling massive parallelism across in-memory arrays [223].

Recent implementations highlight this range. Instruction-level optimizations in Vecim achieve a $33.9\times$ speedup for small matrix operations [235]. Neural Turing Machine-inspired designs report $640\times$ acceleration over CPUs and $46\times$ over GPUs [278]. Memory-integrated architectures such as BLADE deliver up to $6\times$ improvements in image processing and encryption [236].

6.2.2.2 Energy Efficiency

The central objective of memory-centric computing is to lower energy consumption by reducing data movement between compute and memory. Off-chip DRAM access consumes orders of magnitude more energy than in-memory operations [69,217], making local compute within or near memory a key efficiency driver. Energy efficiency is typically reported at two

complementary levels. At the **hardware level**, it is expressed as operations per watt (OPS/W), often in GOPS/W or TOPS/W, or more precisely as GMACS/W when focusing on MAC operations that dominate DNN workloads [69].

At the **system level**, it is measured as energy per inference (J/inference), capturing the end-to-end cost of processing one input, including both computation and memory transfers [223]. The total task energy can be expressed in terms of the number of operations N_{op} , energy per operation E_{op} , memory accesses N_{Li} , and energy per byte at each memory level E_{Li} as:

$$E_{task} = N_{op} \cdot E_{op} + \sum_i (N_{Li} \cdot E_{Li}) \quad (6.1)$$

Recent implementations illustrate the range of achievable efficiency: Vecim reports 289 GOPS/W for Half-Precision Floating-Point (16-bit) (FP16) workloads [235], TS-CIM achieves 701 TOPS/W using time-domain Spiking Neural Network (SNN) acceleration [244], and SRAM-based IMC macros deliver 270 TOPS/W for Boolean logic [279,280].

Energy efficiency can be further improved through:

- **Quantization and sparsity**, which lower precision and skip redundant operations [277].
- **Dynamic power management**, such as clock gating and DVFS to adapt power to workload demand [260].
- **Circuit-level techniques**, such as sub- V_{TH} amplifier design and Lazy Convolution Bitline Reset [274].

6.2.2.3 Design Tradeoffs

Designing memory-centric systems requires navigating several hardware and software tradeoffs:

- **Area vs. Power vs. Performance:** Integrating compute into memory increases silicon area and leakage. Some designs can limit the incurred overhead [280], whereas others, such as Vecim, report up to 11 % [235].
- **Accuracy vs. Efficiency:** Quantization and pruning lower energy but may reduce model accuracy. Retraining is often necessary to recover performance [237].
- **Scalability vs. Cost:** Scaling requires greater bandwidth and capacity, often by adding DRAM or HBM, which increases system cost [263].
- **Utilization vs. Peripheral Overhead:** Large arrays risk underutilization unless closely matched to workload demands, and peripheral circuits can dominate power in lightly loaded regimes [243].

6.3 INTEGRATION CHALLENGES

Memory-centric architectures show clear potential, yet their system-level deployment remains challenged by device limits, programmability, thermal constraints in 3D integration, and secure execution in shared memory. Meeting these demands calls for holistic co-design across hardware and software layers.

6.3.1 AREA, DENSITY, AND DEVICE NON-IDEALITIES

Emerging memory technologies such as MRAM and ReRAM offer increased density and non-volatility but also introduce challenges. Write latency, endurance, and resistance variability hinder stable and reliable execution [216]. For example, scaled MRAM suffers from increased Resistance-Capacitance (RC) delay, affecting write margins [213]. AIMC designs are susceptible to PVT variations, limited ADC resolution, and device non-linearities [281,282]. Error Correcting Code (ECC) schemes, redundant logic, and PVT-aware SRAM design are necessary to ensure robustness.

6.3.2 SOFTWARE STACK, EXECUTION, AND PROGRAMMING SUPPORT

Mapping neural workloads to memory-centric accelerators relies on a well-integrated software stack. Full utilization of these architectures demands coordinated support across compiler toolchains, intermediate representations, runtime systems, and high-level frameworks. Such integration is vital to ensure programmability, portability, and performance across heterogeneous platforms.

6.3.2.1 Toolchains, Compilers, and Intermediate Representations

Compiler toolchains bridge high-level model descriptions to hardware-specific execution. They typically consist of frontends, intermediate abstractions, and backend optimizations tailored to data-centric execution. Key approaches include:

- **Auto-vectorization and ISA extensions:** For example, Vecim integrates RISC-V vector ISA with custom mixed-precision instructions to accelerate vectorized inference [235].
- **HLS tools** enable rapid prototyping on FPGA platforms, facilitating early hardware-software co-design [239].
- **DL framework integration:** allows for direct mapping from PyTorch or TensorFlow streamlines deployment workflows [238].

Advanced compiler backends further optimize execution by exploiting parallelism, restructuring data access, and balancing resource utilization:

- **Tensor Dataflow Graphs (TDFGs):** Introduced in Infinity Stream, TDFG expose fine-grained parallelism and optimize data layout for memory-centric execution [283].
- **Polyhedral compilation:** SongC exemplifies this approach by applying loop restructuring and memory reuse transformations for locality-aware mapping. It integrates multi-level graph abstractions with simulation-driven scheduling, using a Just-in-Time Evaluator to accelerate the scheduling search and an extended roofline model (Palace) to guide instruction generation and resource management in hybrid NMC/IMC systems [264].
- **Cost-based scheduling:** Hardware-aware performance models guide instruction placement across compute-memory hierarchies to achieve balanced utilization.

6.3.2.2 Data Mapping and Execution Models

Efficient data mapping and runtime coordination are critical for exploiting locality, reducing data movement, and ensuring balanced utilization in memory-centric architectures. Despite hardware advances, programmability remains a central bottleneck. As highlighted by Meta [284]:

“We’ve investigated applying PIM to our workloads and determined there are several challenges to using these approaches. Perhaps the biggest challenge of PIM is its programmability.”

These challenges reflect broader issues of workload partitioning, coherence, and task offloading, which require coordinated software and architectural solutions. Key strategies fall into three categories:

- **Data organization and layout:** Systems first shape the data itself to expose locality and parallelism. Partitioning, replication, and intelligent placement are used to balance load and keep producers and consumers close in the hierarchy, which reduces long-distance traffic and improves concurrency [260]. Complementary techniques such as compression, encoding, and data grouping shrink the working set and ease bandwidth pressure [260]. At the physical level, compiler-assisted BL alignment and intra-subarray mapping align logical operations with memory structures to minimize costly movements inside arrays [260].
- **Runtime execution strategies:** At runtime, dedicated systems coordinate offloading, scheduling, and synchronization between host and memory-side compute [46]. Code regions identified through behavioral synthesis and profiling are migrated to near-memory engines, as in NMPO and NM-Carus, to exploit locality while keeping the host in control [238,275]. When workload characteristics change, ML-guided schedulers adapt placement and pacing decisions to sustain utilization and avoid stalls

[285]. Pragmas and compiler directives provide a practical way for developers to guide mapping and establish lightweight synchronization without sacrificing portability [46].

- **Specialized scheduling and co-design:** scheduling policies and algorithms are co-shaped to the memory substrate. Structured reuse and layout-aware mapping for matrix-vector and matrix-matrix kernels keep operands resident where they are consumed, cutting redundant transfers [48, 260]. Sparsity-aware scheduling avoids needless activations and memory touches by exploiting zeros and fine-grained gating [223]. More broadly, algorithm-architecture co-design aligns computational patterns with device and array constraints to achieve balanced throughput and efficiency [216].

6.3.2.3 Programming Paradigms and Prototyping

Building on these execution models, *flexible programming* support is critical to make memory-centric architectures usable across diverse workloads [215]. Emerging paradigms such as functional programming, first articulated by Backus [230], eliminate mutable variables and separate logic from state, simplifying optimization and parallelization in data-centric contexts.

Finally, bridging research and practice requires accessible tools and reproducible infrastructures. Domain-Specific Languages (DSLs) are central to this effort, generating optimized code while abstracting hardware details, often compiling applications into dataflow graph representations that enable deployment across heterogeneous platforms with minimal modification [260]. Examples include MARVEL [286] and flexible cross-platform edge deployment frameworks [215]. Complementing these, open-source platforms such as the eXtensible Heterogeneous Energy-Efficient Platform (X-HEEP) provide reference flows and reproducible toolchains for integrating ULP edge accelerators [287].

6.4 THESIS CONTRIBUTION

This chapter advances the architectural contribution by developing and validating a scalable, all-digital, non-intrusive NMC framework for edge AI. Building on circuit-level optimizations from earlier chapters, the focus here shifts to system integration of NMC units inside MCU memory subsystems to bring compute closer to data while preserving general-purpose programmability.

Paper II defines the concept and design space. Using a RISC-V MCU baseline, it frames NDP for resource-constrained systems, motivates the MCU as a viable platform for multi-objective optimization, characterizes the

efficiency-flexibility tradeoff, and contrasts NMC with common MCU-centric optimizations for CNN workloads. Under matched conditions it reports several-fold gains in performance and energy efficiency over the baseline and representative SoA MCU-enhanced designs, and it introduces the evaluation methodology, workloads, and metrics used in this chapter.

Paper III turns the concept into a modular, platform-agnostic in-cache NMC architecture integrated with a RISC-V MCU. The design is validated on FPGA and by ASIC synthesis, and it provides a well-defined hardware/software interface, runtime-configurable kernels, and memory-resident dataflows. A multi-objective Pareto analysis compares the proposal with representative ASIC, FPGA, and MCU-enhanced multi- and many-core edge designs, positioning it within the broader landscape and reinforcing the benefits of digital NMC.

Paper IV brings the framework to silicon in 22 nm FD-SOI. The NMCu-CNN co-processor is embedded in the shared memory subsystem of an MCU and demonstrates runtime-configurable operation. Measured results reach up to 203 GOPS and 1.7 TOPS/W, confirming scalability and efficiency in a realistic LP setting.

Across **Papers II–IV** the emphasis is on balancing efficiency and flexibility by pairing the efficiency of near-memory fixed-function kernels with the flexibility of MCUs. By embedding NMC in shared memory structures and coordinating execution through a lightweight runtime and compiler hooks, it delivers portable gains without sacrificing programmability. The chapter also summarizes the relevant SoA and outlines the experimental setup and metrics used for fair comparison to contextualize the reported results.

Conclusion and Future Directions

THIS dissertation presents a comprehensive framework for memory-centric computing architectures tailored to energy-efficient edge AI systems, encompassing innovations from the circuit to the system level. The key contributions include a high-density dual-port 6T SRAM optimized for LP concurrent access, a synthesizable distributed-SRAM framework that surpasses conventional break-even points in both area and energy efficiency, and a scalable digital NMC architecture tightly integrated into RISC-V-based MCUs. The platform’s versatility is further enhanced by incorporating AMs that applies operand-aware hardware cost reduction, scalable across different word lengths, enabling tunable tradeoffs between accuracy and hardware-power cost. The proposed designs were implemented and validated on FPGA and silicon prototypes, demonstrating localized MAC operations and high computational efficiency on CNN workloads. Together, these developments establish a solid foundation for flexible and efficient memory-compute integration, paving the way for coordinated advances in hardware, compilers, and cross-layer optimization to support emerging edge intelligence.

FUTURE DIRECTIONS

Emerging Model Paradigms and Hardware Co-Design

Emerging AI workloads increasingly show diverse computational and dataflow patterns that differ from widely adopted network architectures. This trend suggests the need for tighter coupling between algorithmic models and hardware execution through flexible mechanisms that enable co-optimization of NMC architectures with both conventional Artificial Neural Networks (ANNs) and emerging models such as Spiking Neural Networks (SNNs) and Kolmogorov-Arnold Networks (KANs). SNNs offer sparse, event-

driven processing with notable energy advantages, yet practical hardware implementations face routing complexity, irregular memory demands, and control overheads. Similarly, KANs provide compact and expressive representations of complex functions, but their hardware realization remains at an early stage with limited toolchain support. Hybrid approaches, such as ANN-SNN or CNN-KAN fusions, could offer opportunities to combine the complementary strengths of different paradigms. However, they also pose important challenges for hardware programmability, memory organization, and dataflow control, particularly in multimodal edge AI deployments.

Architectural Flexibility

The modular and synthesizable NMC architecture proposed in this work can evolve to support broader model diversity and increasing workload complexity. Programmable, data-driven reconfiguration of memory–compute hierarchies would enable spatial or functional specialization at runtime. Greater dataflow flexibility and heterogeneous execution models are essential for efficiently processing multimodal inputs with diverse temporal and spatial characteristics. Progress in these directions will likely rely on tight compiler–hardware co-design. Strategies for data locality management, approximation control, and concurrency scheduling are needed across the full design stack, from ISA and hardware-description templates to system-level NMC generation flows and graph-based neural network compilers. These advances target intra-chip adaptability and enable NMC-based platforms to dynamically accommodate evolving algorithmic demands.

Scalable Cross-Layer Deployment

Beyond architectural adaptability, effective deployment of NMC requires scaling across multiple compute nodes and layers of the system stack. Domain-specific and lightweight NMC cores align well with applications such as sensor fusion, robotics, wearable healthcare, and augmented reality, where real-time and distributed processing near the sensor faces stringent power, area, and latency constraints. To reach their full potential in such scenarios, architectures need to be supported by mechanisms for coherent multi-node operation, efficient orchestration of on-chip and off-chip data movement, and standardized software ecosystems that expose these hardware capabilities to higher layers of the stack. Addressing these system-level integration challenges is crucial for transforming NMC from a promising architectural concept into a scalable, interoperable, and widely deployable edge AI computing fabric.

Bibliography

- [1] The Economist, "The world's most valuable resource is no longer oil, but data," 2017, accessed: 2025-09-03. [Online]. Available: <https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data>
- [2] A. S. Banga, "Fourth Industrial Revolution: A Global Economy Powered by Data," 2016, accessed: 2025-09-03. [Online]. Available: <https://www.weforum.org/stories/2016/01/a-global-economy-powered-by-data>
- [3] R. Kennedy, "The New Era Of Big Data," 2023, accessed: 2025-09-03. [Online]. Available: <https://www.forbes.com/councils/forbestechcouncil/2023/05/24/the-new-era-of-big-data>
- [4] C. Southerton, "Datafication," in *Encyclopedia of Big Data*, L. A. Schintler *et al.*, Eds. Springer International Publishing, 2020, pp. 1–4.
- [5] D. Kramer, "Will AI's growth create an explosion of energy consumption?" *Physics Today*, vol. 77, no. 4, pp. 28–29, 04 2024.
- [6] Organisation for Economic Co-operation and Development (OECD), "Measuring the Value of Data and Data Flows," OECD Publishing, Paris, Tech. Rep., 2022. [Online]. Available: <https://doi.org/10.1787/923230a6-en>
- [7] J. D'Souza, "Artificial Intelligence Statistics by Market Share, Challenges in Implementation, Revenue and Patents," 2024, accessed: 2025-09-03. [Online]. Available: <https://electroiQ.com/stats/artificial-intelligence-statistics>

- [8] Allied Market Research, "Big Data Analytics Market: Global Opportunity Analysis and Industry Forecast, 2024–2032," 2024, accessed: 2025-09-03. [Online]. Available: <https://www.alliedmarketresearch.com/big-data-analytics-market-A189561>
- [9] A. Braun, "History of IoT: A Timeline of Development," 2019, accessed: 2025-09-03. [Online]. Available: <https://www.iottectrends.com/history-of-iot>
- [10] Y. LeCun, "MNIST Demos on Yann LeCun's Website," 1998, accessed: 2025-04-11. [Online]. Available: <https://yann.lecun.com/exdb/lenet>
- [11] G. Press, "On Thinking Machines, Machine Learning, And How AI Took Over Statistics," 2021, accessed: 2025-09-03. [Online]. Available: <https://www.forbes.com/sites/gilpress/2021/05/28/on-thinking-machines-machine-learning-and-how-ai-took-over-statistics>
- [12] J. Norman, "Thomas Marill and Lawrence G. Roberts Conduct the First "Actual Network Experiment": History of Information," 1965, accessed: 2025-09-03. [Online]. Available: <https://www.historyofinformation.com/detail.php?id=836>
- [13] J. Romkey, "Toast of the IoT: The 1990 Interop Internet Toaster," *IEEE Consumer Electronics Magazine*, vol. 6, no. 1, pp. 116–119, 2017.
- [14] J. Jumper *et al.*, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, pp. 583–589, 2021.
- [15] E. K. Brockmeier, "The World's First General Purpose Computer Turns 75," 2021, accessed: 2025-09-03. [Online]. Available: <https://penntoday.upenn.edu/news/worlds-first-general-purpose-computer-turns-75>
- [16] Computer History Museum, "Timeline of Computer History," 2025, accessed: 2025-09-03. [Online]. Available: <https://www.computerhistory.org/timeline>
- [17] J. Bergström *et al.*, "The History of Mobile Internet: The Technology Transformation That Changed the Lives of Billions," 2024, accessed: 2025-09-03. [Online]. Available: <https://www.ericsson.com/en/report-s-and-papers/ericsson-technology-review/articles/mobile-miracles>
- [18] J. Sevilla *et al.*, "Compute Trends Across Three Eras of Machine Learning," in *International Joint Conference on Neural Networks (IJCNN)*, 2022, pp. 1–8.
- [19] Z. Thorwirth, "AI Winter: The Highs and Lows of Artificial Intelligence," 2021, accessed: 2025-09-03. [Online]. Available: <https://www.historyofdatascience.com/ai-winter-the-highs-and-lows-of-artificial-intelligence>

- [20] L. Emberson *et al.*, “Training compute growth is driven by larger clusters, longer training, and better hardware,” 2025, accessed: 2025-09-28. [Online]. Available: <https://epoch.ai/data-insights/training-compute-decomposition>
- [21] ASPENCORE Network, “CES 2020: Are you ready for the Intelligence of Things?” 2020, accessed: 2025-09-03. [Online]. Available: <https://iot.eetimes.com/ces-2020-are-you-ready-for-the-intelligence-of-things>
- [22] A. Marengo, “Navigating the Nexus of AI and IoT: A Comprehensive Review of Data Analytics and Privacy Paradigms,” *IEEE Internet of Things Journal*, vol. 27, p. 101318, 2024.
- [23] Ericsson Mobility Report, “IoT Connections Outlook,” 2024, accessed: 2025-09-03. [Online]. Available: <https://www.ericsson.com/en/report-s-and-papers/mobility-report/dataforecasts/iot-connections-outlook>
- [24] W. Jiang *et al.*, “The Road Towards 6G: A Comprehensive Survey,” *IEEE Open Journal of the Communications Society*, vol. 2, pp. 334–366, 2021.
- [25] M. Z. Chowdhury *et al.*, “6G Wireless Communication Systems: Applications, Requirements, Technologies, Challenges, and Research Directions,” *IEEE Open Journal of the Communications Society*, vol. 1, pp. 957–975, 2020.
- [26] D. C. Nguyen *et al.*, “6G Internet of Things: A Comprehensive Survey,” *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 359–383, 2022.
- [27] Gartner, “Gartner 2024 Hype Cycle for Emerging Technologies Highlights Developer Productivity, Total Experience, AI and Security,” 2024, accessed: 2025-09-03. [Online]. Available: <https://www.gartner.com/en/newsroom/press-releases/2024-08-21-gartner-2024-hype-cycle-for-emerging-technologies-highlights-developer-productivity-total-experience-ai-and-security>
- [28] B. Kindig, “AI Power Consumption: Rapidly Becoming Mission-Critical,” 2024, accessed: 2025-09-03. [Online]. Available: <https://www.forbes.com/sites/bethkindig/2024/06/20/ai-power-consumption-rapidly-becoming-mission-critical>
- [29] C. E. Tripp *et al.*, “Measuring the Energy Consumption and Efficiency of Deep Neural Networks: An Empirical Analysis and Design Recommendations,” *arXiv:2403.08151 [cs.LG]*, 2024.
- [30] Morgan Stanley Research, “Powering the AI Revolution,” 2024, accessed: 2025-09-03. [Online]. Available: <https://www.morganstanley.com/ideas/ai-energy-demand-infrastructure>

- [31] International Energy Agency, "Electricity 2024: Analysis and Forecast to 2026," 2024, accessed: 2025-09-03. [Online]. Available: <https://www.iea.org/reports/electricity-2024>
- [32] Google, "Environmental Report – Google Sustainability," 2024, accessed: 2025-09-03. [Online]. Available: <https://sustainability.google/reports/google-2024-environmental-report>
- [33] C. Hodgson, "Microsoft's emissions jump almost 30% as it races to meet AI demand," 2024, accessed: 2025-09-03. [Online]. Available: <https://www.ft.com/content/61bd45d9-2c0f-479a-8b24-605d5e72f1ab>
- [34] K. Bourzac, "Fixing AI's energy crisis," *Nature*, vol. 632, no. 8031, pp. 22–25, 2024, accessed: 2025-09-03. [Online]. Available: <https://www.nature.com/articles/d41586-024-03408-z>
- [35] T. Holmes *et al.*, "Energy Efficiency on Edge Computing: Challenges and Vision," *IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 1–6, 2022.
- [36] G. Carvalho *et al.*, "Edge computing: current trends, research challenges and future directions," *Computing*, vol. 103, no. 5, pp. 993–1023, 2021.
- [37] G. Han *et al.*, "Special Section on Emerging Trends Issues and Challenges in Edge Artificial Intelligence," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4172–4177, 2019.
- [38] W. Z. Khan *et al.*, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019.
- [39] W. A. Wulf *et al.*, "Hitting the memory wall: implications of the obvious," *ACM SIGARCH Computer Architecture News*, vol. 23, no. 1, pp. 20–24, 1995.
- [40] Z. Or-Bach, "A 1,000x improvement in computer systems by bridging the processor-memory gap," in *IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, 2017, pp. 1–4.
- [41] A. Gholami *et al.*, "AI and Memory Wall," *IEEE Micro*, vol. 44, no. 3, pp. 33–39, 2024.
- [42] J. Kim *et al.*, "Bandwidth Bottleneck in Network-on-Chip for High-Throughput Processors," in *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques (PACT '20)*. Association for Computing Machinery, 2020, pp. 157–158.
- [43] O. Mutlu, "Memory-Centric Computing," *arXiv:2305.20000 [cs.AR]*, 2023.

- [44] A. Boroumand *et al.*, “Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks,” in *30th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2021, pp. 159–172.
- [45] A. Boroumand *et al.*, “Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks,” in *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '18)*, 2018, pp. 316–331.
- [46] C. Zhang *et al.*, “A Survey of Memory-Centric Energy Efficient Computer Architecture,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 10, pp. 2657–2670, 2023.
- [47] A. A. Khan *et al.*, “The Landscape of Compute-near-memory and Compute-in-memory: A Research and Commercial Overview,” *arXiv:2401.14428 [cs.AR]*, 2024.
- [48] S. Angizi *et al.*, “MRIMA: An MRAM-Based In-Memory Accelerator,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 5, pp. 1123–1136, 2020.
- [49] T. Hiramoto, “Never-Ending CMOS Innovation: The Past, Present, and Future Perspectives,” in *2020 International Symposium on VLSI Technology, Systems and Applications (VLSI-TSA)*, 2020, pp. 3–4.
- [50] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Prentice Hall, 1996. ISBN 978-0-13-178609-7
- [51] N. H. E. Weste *et al.*, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Addison-Wesley, 2010. ISBN 978-0-321-54774-3
- [52] N. Reynders *et al.*, *Ultra-Low-Voltage Design of Energy-Efficient Digital Circuits*. Cham, Switzerland: Springer, 2015.
- [53] O. Azizi *et al.*, “Energy-performance tradeoffs in processor architecture and circuit design: A marginal cost analysis,” *ACM SIGARCH Computer Architecture News*, vol. 38, pp. 26–36, 2010.
- [54] J. M. Rabaey, *Low Power Design Essentials*. Boston, MA: Springer, 2009. ISBN 978-0-387-71712-8
- [55] M. Alioto, “Energy-quality scalable adaptive VLSI circuits and systems beyond approximate computing,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017, pp. 127–132.
- [56] B. D. Bingham *et al.*, “Modeling Energy-Time Trade-Offs in VLSI Computation,” *IEEE Transactions on Computers*, vol. 61, no. 4, pp. 530–547, 2012.

- [57] S. Huang *et al.*, “How Flexible is Your Computing System?” *ACM Transactions on Embedded Computing Systems*, vol. 21, no. 4, 2022.
- [58] B. R. Zeydel *et al.*, “Energy-Efficient Design Methodologies: High-Performance VLSI Adders,” *IEEE Journal of Solid-State Circuits*, vol. 45, no. 6, pp. 1220–1233, 2010.
- [59] A. B. Kahng *et al.*, “Enhancing the Efficiency of Energy-Constrained DVFS Designs,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 10, pp. 1769–1782, 2013.
- [60] D. Markovic *et al.*, “Methods for true energy-performance optimization,” *IEEE Journal of Solid-State Circuits*, vol. 39, no. 8, pp. 1282–1293, 2004.
- [61] T. Casagrande *et al.*, “GTFUZZ: A novel algorithm for robust dynamic power optimization via gate sizing with fuzzy games,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015, pp. 677–682.
- [62] A. Shapiro *et al.*, “Power Efficient Level Shifter for 16 nm FinFET Near Threshold Circuits,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 2, pp. 774–778, 2016.
- [63] J. Samandari-Rad *et al.*, “Power/Energy Minimization Techniques for Variability-Aware High-Performance 16-nm 6T-SRAM,” *IEEE Access*, vol. 4, pp. 594–613, 2016.
- [64] A. Chhabra *et al.*, “Low-Energy Power-ON-Reset Circuit for Dual Supply SRAM,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 5, pp. 2003–2007, 2016.
- [65] E. Macii *et al.*, “Integrating Clock Gating and Power Gating for Combined Dynamic and Leakage Power Optimization in Digital CMOS Circuits,” in *11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools*, 2008, pp. 298–303.
- [66] L. Anghel *et al.*, “Design-Time Exploration for Process, Environment and Aging Compensation Techniques for Low Power Reliable-Aware Design,” *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 581–590, 2022.
- [67] Y. Shin *et al.*, “Power gating: Circuits, design methodologies, and best practice for standard-cell VLSI designs,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 15, no. 4, 2010.
- [68] M. Cho *et al.*, “Adaptive clocking with dynamic power gating for mitigating energy efficiency & performance impacts of fast voltage droop in a 22nm graphics execution core,” in *IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, 2016, pp. 1–2.

- [69] Y.-H. Chen *et al.*, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.
- [70] H. Mahdavi *et al.*, "Improving Architectures of Binary Signed-Digit CORDIC With Generic/Specific Initial Angles," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 7, pp. 2297–2304, 2020.
- [71] y.-M. Yu *et al.*, "Passive and Semi-Passive Wireless Temperature and Humidity Sensors Based on EPC Generation-2 UHF Protocol," *IEEE Sensors Journal*, vol. 15, no. 4, pp. 2403–2411, 2015.
- [72] A. Pal, *Low-Power VLSI Circuits and Systems*, 1st ed. New Delhi, India: Springer India, 2014. ISBN 978-81-322-1936-1
- [73] W. Zhang *et al.*, "Extending Performance-Energy Trade-offs Via Dynamic Core Scaling," *IEEE Transactions on Computers*, vol. 70, no. 11, pp. 1875–1886, 2021.
- [74] A. W. Burks *et al.*, *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument*. Berlin, Heidelberg: Springer, 1982, pp. 399–413.
- [75] J. von Neumann, "First draft of a report on the EDVAC," *IEEE Annals of the History of Computing*, 1945.
- [76] J. L. Hennessy *et al.*, *Computer Architecture: A Quantitative Approach*, 6th ed. Morgan Kaufmann, 2017. ISBN 978-0-12-811905-1
- [77] D. Burger, "Memory systems," *ACM Computing Surveys*, vol. 28, no. 1, pp. 63–65, 1996.
- [78] R. Nair, "Evolution of Memory Architecture," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1331–1345, 2015.
- [79] W. Wei *et al.*, "Design and comparative evaluation of a hybrid Cache memory at architectural level," in *International Great Lakes Symposium on VLSI (GLSVLSI)*, 2016, pp. 125–128.
- [80] Intel, "13th Generation Intel Core and Intel Core 14th Generation Processors Datasheet," 2024, accessed: 2025-09-03. [Online]. Available: <https://www.intel.com/content/www/us/en/content-details/743844/13th-generation-intel-core-and-intel-core-14th-generation-processors-datasheet-volume-1-of-2.html>
- [81] J. Sim *et al.*, "FLEXclusion: Balancing cache capacity and on-chip bandwidth via Flexible Exclusion," in *39th Annual International Symposium on Computer Architecture (ISCA)*, 2012, pp. 321–332.

- [82] S. Dublisch *et al.*, "Evaluating and mitigating bandwidth bottlenecks across the memory hierarchy in GPUs," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2017, pp. 239–248.
- [83] H. Xu *et al.*, "Maximizing Energy Efficiency of on-Chip Caches Exploiting Hybrid Memory Structure," in *28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2018, pp. 237–242.
- [84] R. Braojos *et al.*, "A Synchronization-Based Hybrid-Memory Multi-Core Architecture for Energy-Efficient Biomedical Signal Processing," *IEEE Transactions on Computers*, vol. 66, no. 4, pp. 575–585, 2017.
- [85] D. Bortolotti *et al.*, "Hybrid memory architecture for voltage scaling in ultra-low power multi-core biomedical processors," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1–6.
- [86] A. Burg, *Gain-cell embedded DRAMs for low-power VLSI systems-on-chip*, 1st ed. Springer International Publishing, 2015.
- [87] N. Verma *et al.*, "Analysis towards minimization of total SRAM energy over active and idle operating modes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 9, pp. 1697–1709, 2011.
- [88] M. Kutila *et al.*, "Comparison of 130 nm technology 6T and 8T SRAM cell designs for Near-Threshold operation," in *IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2014, pp. 925–928.
- [89] J. Singh *et al.*, "Single ended 6T SRAM with isolated read-port for low-power embedded systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2009, pp. 917–922.
- [90] A. Pavlov *et al.*, *CMOS SRAM Circuit Design and Parametric Test in Nano-Scaled Technologies*, 1st ed. Springer Dordrecht, 2010.
- [91] M. E. Sinangil *et al.*, "A 28nm High-Density 6T SRAM with Optimized Peripheral-Assist Circuits for Operation Down to 0.6V," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2013, pp. 254–256.
- [92] A. Calimera *et al.*, "Design techniques and architectures for low-leakage SRAMs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 9, pp. 2001–2016, 2012.
- [93] N. Surana *et al.*, "Energy Efficient Single-Ended 6-T SRAM for Multi-media Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 5, pp. 1549–1559, 2019.

- [94] D. Takashima *et al.*, "A 7T-SRAM With Data-Write Technique by Capacitive Coupling," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 2, pp. 596–605, 2019.
- [95] J. P. Kulkarni *et al.*, "5.6 Mb/mm² 1R1W 8T SRAM Arrays Operating Down to 560 mV Utilizing Small-Signal Sensing With Charge Shared Bitline and Asymmetric Sense Amplifier in 14 nm FinFET CMOS Technology," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 229–239, 2017.
- [96] H. Mori *et al.*, "A 28-nm FD-SOI 8T Dual-Port SRAM for Low-Energy Image Processor With Selective Sourceline Drive Scheme," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 4, pp. 1442–1453, 2019.
- [97] J. Singh *et al.*, "A 2-port 6T SRAM bitcell design with multi-port capabilities at reduced area overhead," in *11th International Symposium on Quality Electronic Design (ISQED)*, 2010, pp. 131–138.
- [98] S. Ataei *et al.*, "A high performance multi-port SRAM for low voltage shared memory systems in 32 nm CMOS," in *IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 1236–1239.
- [99] K. Nii *et al.*, "2RW dual-port SRAM design challenges in advanced technology nodes," in *IEEE International Electron Devices Meeting (IEDM)*, 2015, pp. 11.1.1–11.1.4.
- [100] J. B. Kuang *et al.*, "The Design and Characterization of a Half-Volt 32 nm Dual-Read 6T SRAM," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 9, pp. 2010–2016, 2011.
- [101] V. Nautiyal *et al.*, "An ultra high density pseudo dual-port SRAM in 16nm FINFET process for graphics processors," in *30th IEEE International System-on-Chip Conference (SOCC)*, 2017, pp. 12–17.
- [102] Y. Ishii *et al.*, "A 28 nm Dual-Port SRAM Macro With Screening Circuitry Against Write-Read Disturb Failure Issues," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 11, pp. 2535–2544, 2011.
- [103] J. P. Kulkarni *et al.*, "A Read-Disturb-Free, Differential Sensing 1R/1W Port, 8T Bitcell Array," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 9, pp. 1727–1730, 2011.
- [104] S. Ishikura *et al.*, "A 45 nm 2-port 8T-SRAM Using Hierarchical Replica Bitline Technique With Immunity From Simultaneous R/W Access Issues," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 938–945, 2008.

- [105] M.-F. Chang *et al.*, "A differential data aware power-supplied (D2AP) 8T SRAM cell with expanded write/read stabilities for lower VDDmin applications," in *IEEE Symposium on VLSI Technology and Circuits*, 2009, pp. 156–157.
- [106] J.-J. Wu *et al.*, "A 45-nm Dual-Port SRAM Utilizing Write-Assist Cells Against Simultaneous Access Disturbances," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 11, pp. 790–794, 2012.
- [107] M. Khayat-zadeh *et al.*, "Average-8T Differential-Sensing Subthreshold SRAM With Bit Interleaving and 1k Bits Per Bitline," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 5, pp. 971–982, 2014.
- [108] A. Jaiswal *et al.*, "8T SRAM Cell as a Multibit Dot-Product Engine for Beyond Von Neumann Computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2556–2567, 2019.
- [109] S. Gupta *et al.*, "A 32-nm Subthreshold 7T SRAM Bit Cell With Read Assist," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 12, pp. 3473–3483, 2017.
- [110] S. Gupta *et al.*, "Low-Power Near-Threshold 10T SRAM Bit Cells With Enhanced Data-Independent Read Port Leakage for Array Augmentation in 32-nm CMOS," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 3, pp. 978–988, 2019.
- [111] B. Mohammadi *et al.*, "A 128 kb 7T SRAM Using a Single-Cycle Boosting Mechanism in 28-nm FD-SOI," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1257–1268, 2018.
- [112] S. Lutkemeier *et al.*, "A 65 nm 32 b Subthreshold Processor With 9T Multi-Vt SRAM and Adaptive Supply Voltage Control," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 1, pp. 8–19, 2013.
- [113] H. K. O. Berge *et al.*, "Design of 9T SRAM for dynamic voltage supplies by a multiobjective optimization approach," in *17th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2010, pp. 319–322.
- [114] N. Maroof *et al.*, "10T SRAM Using Half- V_{DD} Precharge and Row-Wise Dynamically Powered Read Port for Low Switching Power and Ultralow RBL Leakage," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 4, pp. 1193–1203, 2017.
- [115] E. Abbasian *et al.*, "A Reliable Low Standby Power 10T SRAM Cell With Expanded Static Noise Margins," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 4, pp. 1606–1616, 2022.

- [116] V. Sharma *et al.*, "A robust, ultra low-power, data-dependent-power-supplied 11T SRAM cell with expanded read/write stabilities for internet-of-things applications," *ACM Analog Integrated Circuits and Signal Processing*, vol. 98, no. 2, pp. 331–346, 2019.
- [117] V. Sharma *et al.*, "A write improved low power 12T SRAM cell for wearable wireless sensor nodes," *International Journal of Circuit Theory and Applications*, vol. 46, pp. 2314–2333, 2018.
- [118] M. A. Turi *et al.*, "Effective Low Leakage 6T and 8T FinFET SRAMs: Using Cells With Reverse-Biased FinFETs, Near-Threshold Operation, and Power Gating," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 5, pp. 954–963, 2015.
- [119] E. Morifuji *et al.*, "Power optimization for SRAM and its scaling," *IEEE Transactions on Electron Devices*, vol. 54, no. 4, pp. 714–721, 2007.
- [120] X. Yang *et al.*, "Robust 6T Si tunneling transistor SRAM design," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2011, pp. 1–6.
- [121] B. Calhoun *et al.*, "Static noise margin variation for sub-threshold SRAM in 65-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 7, pp. 1673–1679, 2006.
- [122] B. Mohammadi *et al.*, "A 128 kb single-bitline 8.4 fJ/bit 90MHz at 0.3V 7T sense-amplifierless SRAM in 28 nm FD-SOI," in *42nd IEEE European Solid-State Circuits Conference (ESSCIRC)*, 2016, pp. 429–432.
- [123] Y.-H. Chen *et al.*, "A Novel Cache-Utilization-Based Dynamic Voltage-Frequency Scaling Mechanism for Reliability Enhancements," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, pp. 820–832, 2017.
- [124] M. E. Sinangil *et al.*, "A Reconfigurable 8T Ultra-Dynamic Voltage Scalable (U-DVS) SRAM in 65 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 11, pp. 3163–3173, 2009.
- [125] M. Igarashi *et al.*, "A 28 nm High-k/MG Heterogeneous Multi-Core Mobile Application Processor With 2 GHz Cores and Low-Power 1 GHz Cores," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 92–101, 2015.
- [126] M. Ling *et al.*, "A Design of Timing Speculation SRAM-Based L1 Caches With PVT Autotracking Under Near-Threshold Voltages," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 12, pp. 2197–2209, 2021.
- [127] A. Saha *et al.*, "Significance driven hybrid 8T-6T SRAM for energy-efficient synaptic storage in artificial neural networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 6, pp. 793–797, 2018.

- [128] J. Kwon *et al.*, "Heterogeneous SRAM Cell Sizing for Low-Power H.264 Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 10, pp. 2275–2284, 2012.
- [129] D. Rossi *et al.*, "Energy-Efficient Near-Threshold Parallel Computing: The PULPv2 Cluster," *IEEE Micro*, vol. 37, no. 5, pp. 20–31, 2017.
- [130] H. Marinberg *et al.*, "Efficient Implementation of Many-Ported Memories by Using Standard-Cell Memory Approach," *IEEE Access*, vol. 11, pp. 94 885–94 897, 2023.
- [131] A. Teman *et al.*, "Power, Area, and Performance Optimization of Standard Cell Memory Arrays Through Controlled Placement," *ACM Transactions on Design Automation of Electronic Systems*, vol. 21, no. 4, 2016.
- [132] P. Meinerzhagen *et al.*, "Benchmarking of Standard-Cell Based Memories in the Sub-V_{th} Domain in 65-nm CMOS Technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 2, pp. 173–182, 2011.
- [133] O. Andersson *et al.*, "Ultra Low Voltage Synthesizable Memories: A Trade-Off Discussion in 65 nm CMOS," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 6, pp. 806–817, 2016.
- [134] X. Fan *et al.*, "Synthesizable Memory Arrays Based on Logic Gates for Subthreshold Operation in IoT," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 3, pp. 941–954, 2019.
- [135] O. Andersson *et al.*, "A 35 fJ/bit-access sub-V_T memory using a dual-bit area-optimized standard-cell in 65 nm CMOS," in *IEEE 40th European Solid State Circuits Conference (ESSCIRC)*, 2014, pp. 243–246.
- [136] P. Meinerzhagen *et al.*, "Towards generic low-power area-efficient standard cell based memory architectures," in *53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2010, pp. 129–132.
- [137] D. Esposito *et al.*, "Power-precision scalable latch memories," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.
- [138] J. Zhou *et al.*, "Standard Cell based Memory Compiler for Near/Sub-threshold Operation," in *27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2020, pp. 1–4.
- [139] K. Chen *et al.*, "Improving power efficiency of GPGPU's global memory by a hybrid memory approach," in *4th IEEE International Conference on Information Science and Technology*, 2014, pp. 660–664.
- [140] J. Hu *et al.*, "Towards energy efficient hybrid on-chip Scratch Pad Memory with non-volatile memory," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2011, pp. 1–6.

- [141] S. H. H. Nemati *et al.*, "A Hybrid SRAM/RRAM In-Memory Computing Architecture Based on a Reconfigurable SRAM Sense Amplifier," *IEEE Access*, vol. 11, pp. 72 159–72 171, 2023.
- [142] G. Tagliavini *et al.*, "Synergistic HW/SW Approximation Techniques for Ultralow-Power Parallel Computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 5, pp. 982–995, 2018.
- [143] J. Shiomi *et al.*, "Fully digital on-chip memory using minimum height standard cells for near-threshold voltage computing," in *26th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2016, pp. 44–49.
- [144] W. Wei *et al.*, "Design and comparative evaluation of a hybrid Cache memory at architectural level," in *International Great Lakes Symposium on VLSI (GLSVLSI)*, 2016, pp. 125–128.
- [145] H. Zhang *et al.*, "HD-CIM: Hybrid-Device Computing-In-Memory Structure Based on MRAM and SRAM to Reduce Weight Loading Energy of Neural Networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 11, pp. 4465–4474, 2022.
- [146] M. Qiu *et al.*, "Low-Power Low-Latency Data Allocation for Hybrid Scratch-Pad Memory," *IEEE Embedded Systems Letters*, vol. 6, no. 4, pp. 69–72, 2014.
- [147] Y. Xie *et al.*, "Emerging Memory Technologies," *IEEE Micro*, vol. 39, pp. 6–7, 2019.
- [148] E. Veera Boopathy *et al.*, "Advancements in Next-Generation Memory Technologies: Challenges and Applications," in *Interdisciplinary Approaches to AI, Internet of Everything, and Machine Learning*, D. Pandey *et al.*, Eds. IGI Global Scientific Publishing, 2025, pp. 25–48.
- [149] A. M. Caulfield *et al.*, "Understanding the Impact of Emerging Non-Volatile Memories on High-Performance, IO-Intensive Computing," in *SC '10: Proceedings ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, 2010, pp. 1–11.
- [150] Y.-H. Chang *et al.*, "Guest Editorial: IEEE Transactions on Computers Special Section on Emerging Non-Volatile Memory Technologies: From Devices to Architectures and Systems," *IEEE Transactions on Computers*, vol. 68, pp. 1111–1113, 2019.
- [151] A. Chen, "A review of emerging non-volatile memory (NVM) technologies and applications," *Solid-state Electronics*, vol. 125, pp. 25–38, 2016.
- [152] Y. Chen *et al.*, "Recent Technology Advances of Emerging Memories," *IEEE Design & Test of Computers*, vol. 34, pp. 8–22, 2017.

- [153] S. Yu *et al.*, “Emerging Memory Technologies: Recent Trends and Prospects,” *IEEE Solid-State Circuits Magazine*, vol. 8, pp. 43–56, 2016.
- [154] J. Handy *et al.*, “How Emerging Memories Extend Battery Life,” *Computer*, vol. 57, no. 3, pp. 113–116, 2024.
- [155] Z. Wang *et al.*, “Memory Technology: Development, Fundamentals, and Future Trends,” in *Advanced Memory Technology: Functional Materials and Devices*. Royal Society of Chemistry, Oct 2023.
- [156] S. Ghosh *et al.*, “Guest Editorial Emerging Memories—Technology, Architecture and Applications,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, pp. 105–108, 2016.
- [157] B. Mohammadi *et al.*, “An area efficient single-cycle xvdd sub-vth on-chip boost scheme in 28 nm fd-soi,” in *2016 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, 2016, pp. 229–232.
- [158] A. Stansfield, “Memory with local-/global bit line architecture and additional capacitance for global bit line discharge in reading,” US Patent US9406351B2, 2016, assignee: Surecore Limited, Leeds (GB). [Online]. Available: <https://patents.google.com/patent/US9406351B2>
- [159] Cadence Design Systems. (2020) Innovus Mixed Placer. Accessed: 2025-09-30. [Online]. Available: https://community.cadence.com/cadence_blogs_8/b/breakfast-bytes/posts/innovus-mixed-placer
- [160] J. Sun *et al.*, “An Ultra-Low-Voltage Bit-Interleaved Synthesizable 13T SRAM Circuit,” *IEEE Journal of Solid-State Circuits*, vol. 57, no. 11, pp. 3477–3489, 2022.
- [161] P. Dubey *et al.*, “A 0.42V high bandwidth synthesizable parallel access smart memory fabric for computer vision,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.
- [162] P. Meinerzhagen *et al.*, “A 500 fW/bit 14 fJ/bit-access 4kb standard-cell based sub-VT memory in 65nm CMOS,” in *Proceedings IEEE European Solid State Circuits Conference (ESSCIRC)*, 2012, pp. 321–324.
- [163] H. Esmaeilzadeh *et al.*, “Dark silicon and the end of multicore scaling,” in *38th Annual International Symposium on Computer Architecture (ISCA)*, 2011, pp. 365–376.
- [164] H. Jiang *et al.*, “Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications,” *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.
- [165] S. Venkataramani *et al.*, “Approximate computing and the quest for computing efficiency,” in *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.

- [166] V. K. Chippa *et al.*, “Analysis and characterization of inherent application resilience for approximate computing,” in *50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2013, pp. 1–9.
- [167] A. Alaghi *et al.*, “The Promise and Challenge of Stochastic Computing,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1515–1531, 2018.
- [168] M. Shafique *et al.*, “Invited: Cross-layer approximate computing: From logic to architectures,” in *53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.
- [169] Q. Xu *et al.*, “Approximate Computing: A Survey,” *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, 2016.
- [170] H. Esmailzadeh *et al.*, “Architecture support for disciplined approximate programming,” in *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XVII. New York, NY, USA: Association for Computing Machinery, 2012. ISBN 9781450307598 pp. 301–312. [Online]. Available: <https://doi.org/10.1145/2150976.2151008>
- [171] A. Sampson *et al.*, “Ener]: approximate data types for safe and general low-power computation,” in *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI ’11)*. New York, NY, USA: Association for Computing Machinery, 2011. ISBN 9781450306638 pp. 164–174.
- [172] M. Gao *et al.*, “A novel data format for approximate arithmetic computing,” in *22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2017, pp. 390–395.
- [173] J. Mengte *et al.*, “Exploiting the forgiving nature of applications for scalable parallel execution,” in *IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, 2010, pp. 1–12.
- [174] U. R. Karpuzcu *et al.*, “Accordion: Toward soft Near-Threshold Voltage Computing,” in *IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, 2014, pp. 72–83.
- [175] I. J. Chang *et al.*, “A Priority-Based 6T/8T Hybrid SRAM Architecture for Aggressive Voltage Scaling in Video Applications,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 2, pp. 101–112, 2011.
- [176] A. Ranjan *et al.*, “ASLAN: synthesis of approximate sequential circuits,” in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE ’14)*, 2014.

- [177] J. Han *et al.*, "Approximate computing: An emerging paradigm for energy-efficient design," in *18th IEEE European Test Symposium (ETS)*, 2013, pp. 1–6.
- [178] J. N. Mitchell, "Computer Multiplication and Division Using Binary Logarithms," *IRE Transactions on Electronic Computers*, vol. EC-11, no. 4, pp. 512–517, 1962.
- [179] V. Leon *et al.*, "Walking through the Energy-Error Pareto Frontier of Approximate Multipliers," *IEEE Micro*, vol. 38, no. 4, pp. 40–49, 2018.
- [180] Z. Li *et al.*, "Adaptable Approximate Multiplier Design Based on Input Distribution and Polarity," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 12, pp. 1813–1826, 2022.
- [181] F. Sabetzadeh *et al.*, "A Majority-Based Imprecise Multiplier for Ultra-Efficient Approximate Image Multiplication," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 11, pp. 4200–4208, 2019.
- [182] G. Zervakis *et al.*, "Design-Efficient Approximate Multiplication Circuits Through Partial Product Perforation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 10, pp. 3105–3117, 2016.
- [183] W. Liu *et al.*, "Design and Analysis of Majority Logic-Based Approximate Adders and Multipliers," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1609–1624, 2021.
- [184] H. Pei *et al.*, "Design of Ultra-Low Power Consumption Approximate 4–2 Compressors Based on the Compensation Characteristic," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 1, pp. 461–465, 2021.
- [185] A. G. M. Strollo *et al.*, "Low-Power Approximate Multiplier with Error Recovery using a New Approximate 4-2 Compressor," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–4.
- [186] Y. Guo *et al.*, "Power-Efficient and Small-Area Approximate Multiplier Design with FPGA-Based Compressors," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2024, pp. 1–5.
- [187] U. Anil Kumar *et al.*, "CAAM: Compressor-Based Adaptive Approximate Multiplier for Neural Network Applications," *IEEE Embedded Systems Letters*, vol. 15, no. 3, pp. 117–120, 2023.
- [188] A. Sadeghi *et al.*, "Energy Efficient Compact Approximate Multiplier for Error-Resilient Applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 12, pp. 4989–4993, 2024.

- [189] L. Sayadi *et al.*, "Two Efficient Approximate Unsigned Multipliers by Developing New Configuration for Approximate 4:2 Compressors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 4, pp. 1649–1659, 2023.
- [190] Y. Guo *et al.*, "Design of Power and Area Efficient Lower-Part-OR Approximate Multiplier," in *TENCON 2018 - 2018 IEEE Region 10 Conference*, 2018, pp. 2110–2115.
- [191] P. Balasubramanian *et al.*, "Approximate quasi-delay-insensitive asynchronous adders: Design and analysis," in *IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 1196–1199.
- [192] J. Lee *et al.*, "Design of a Low-Cost Approximate Adder with a Zero Truncation," in *International SoC Design Conference (ISOCC)*, 2020, pp. 69–70.
- [193] M. S. Ansari *et al.*, "Improving the Accuracy and Hardware Efficiency of Neural Networks Using Approximate Multipliers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 2, pp. 317–328, 2020.
- [194] I. Qiqieh *et al.*, "Energy-efficient approximate multiplier design using bit significance-driven logic compression," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017, pp. 7–12.
- [195] S. Vahdat *et al.*, "TOSAM: An Energy-Efficient Truncation- and Rounding-Based Scalable Approximate Multiplier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, pp. 1161–1173, May 2019.
- [196] N. Petra *et al.*, "Truncated Binary Multipliers With Variable Correction and Minimum Mean Square Error," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 6, pp. 1312–1325, 2010.
- [197] A. Momeni *et al.*, "Design and Analysis of Approximate Compressors for Multiplication," *IEEE Transactions on Computers*, vol. 64, pp. 984–994, 2015.
- [198] M. Ahmadinejad *et al.*, "Energy- and Quality-Efficient Approximate Multipliers for Neural Network and Image Processing Applications," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1105–1116, 2022.
- [199] C. Liu *et al.*, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1–4.

- [200] Y. Lu *et al.*, “Low Error-Rate Approximate Multiplier Design for DNNs with Hardware-Driven Co-Optimization,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022, pp. 3507–3511.
- [201] H. Waris *et al.*, “AxRMs: Approximate Recursive Multipliers Using High-Performance Building Blocks,” *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1229–1235, 2022.
- [202] A. G. M. Strollo *et al.*, “Approximate Multipliers Using Static Segmentation: Error Analysis and Improvements,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 6, pp. 2449–2462, 2022.
- [203] V. Leon *et al.*, “Approximate Hybrid High Radix Encoding for Energy-Efficient Inexact Multipliers,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 421–430, 2018.
- [204] G. Park *et al.*, “Simplified Compressor and Encoder Designs for Low-Cost Approximate Radix-4 Booth Multiplier,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 3, pp. 1154–1158, 2023.
- [205] B. Liu *et al.*, “A Reconfigurable Approximate Computing Architecture With Dual-VDD for Low-Power Binarized Weight Network Deployment,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 1, pp. 291–295, 2023.
- [206] S. Rehman *et al.*, “Architectural-space exploration of approximate multipliers,” in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.
- [207] F. Sabetzadeh *et al.*, “An Ultra-Efficient Approximate Multiplier With Error Compensation for Error-Resilient Applications,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 2, pp. 776–780, 2023.
- [208] S. Mazahir *et al.*, “An area-efficient consolidated configurable error correction for approximate hardware accelerators,” in *53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.
- [209] J. Liang *et al.*, “New Metrics for the Reliability of Approximate and Probabilistic Adders,” *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1760–1771, 2013.
- [210] P. J. Edavoor *et al.*, “Approximate Multiplier Design Using Novel Dual-Stage 4:2 Compressors,” *IEEE Access*, vol. 8, pp. 48 337–48 351, 2020.
- [211] R. Zendegani *et al.*, “AMCAL: Approximate Multiplier With the Configurable Accuracy Levels for Image Processing and Convolutional Neural Network,” *IEEE Access*, vol. 12, pp. 94 135–94 151, 2024.
- [212] Y. Yang *et al.*, “Rebooting Computing in Post Moore Era,” *Advanced Intelligent Systems*, vol. 4, no. 8, p. 2200161, 2022.

- [213] Y. Li *et al.*, “A Survey of MRAM-Centric Computing: From Near Memory to In Memory,” *IEEE Transactions on Emerging Topics in Computing*, vol. 11, no. 2, pp. 318–330, 2023.
- [214] G. Singh *et al.*, “A Review of Near-Memory Computing Architectures: Opportunities and Challenges,” in *Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 608–617.
- [215] D. Fakhry *et al.*, “A review on computational storage devices and near memory computing for high performance applications,” *Memories - Materials, Devices, Circuits and Systems*, vol. 4, p. 100051, 2023.
- [216] W. Sun *et al.*, “A Survey of Computing-in-Memory Processor: From Circuit to Application,” *IEEE Open Journal of the Solid-State Circuits Society*, vol. 4, pp. 25–42, 2024.
- [217] Y.-H. Chen *et al.*, “Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks,” in *ACM/IEEE 43rd International Symposium on Computer Architecture (ISCA)*, 2016, pp. 367–379.
- [218] S. Srinivasa *et al.*, “Trends and Opportunities for SRAM Based In-Memory and Near-Memory Computation,” in *22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 547–552.
- [219] A. Sebastian *et al.*, “Memory devices and applications for in-memory computing,” *Nature Nanotechnology*, vol. 15, no. 8, pp. 529–544, 2020.
- [220] G. Singh *et al.*, “Near-memory computing: Past, present, and future,” *Microprocessors and Microsystems*, vol. 71, p. 102868, 2019.
- [221] M. I. Jordan *et al.*, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015.
- [222] Z. Chen *et al.*, “Survey on AI Sustainability: Emerging Trends on Learning Algorithms and Research Challenges,” *IEEE Computational Intelligence Magazine*, vol. 18, no. 2, pp. 60–77, 2023.
- [223] M. Verhelst *et al.*, “How to Keep Pushing ML Accelerator Performance? Know Your Rooflines!” *IEEE Journal of Solid-State Circuits*, vol. 60, no. 6, pp. 1888–1905, 2025.
- [224] Y. Himeur *et al.*, “Edge AI for Internet of Energy: Challenges and Perspectives,” *arXiv:2311.16851 [cs.NI]*, 2023.
- [225] Y. Shen *et al.*, “Large Language Models Empowered Autonomous Edge AI for Connected Intelligence,” *IEEE Communications Magazine*, pp. 1–7, 2024.

- [226] M. Injadat *et al.*, “Machine Learning Towards Intelligent Systems: Applications, Challenges, and Opportunities,” *arXiv:2101.03655 [cs.LG]*, 2021.
- [227] W. Salhab *et al.*, “A Systematic Literature Review on AI Safety: Identifying Trends, Challenges, and Future Directions,” *IEEE Access*, vol. 12, pp. 131 762–131 784, 2024.
- [228] L. Wulfert *et al.*, “AlfES: A Next-Generation Edge AI Framework,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 6, pp. 4519–4533, 2024.
- [229] C. Xiao *et al.*, “Optimal Mapping of Spiking Neural Network to Neuromorphic Hardware for Edge-AI,” *Sensors*, vol. 22, no. 19, 2022.
- [230] J. Backus, “Can programming be liberated from the von Neumann style? a functional style and its algebra of programs,” *Commun. ACM*, vol. 21, no. 8, pp. 613–641, 1978.
- [231] C. Kozyrakis *et al.*, “Scalable processors in the billion-transistor era: IRAM,” *Computer*, vol. 30, no. 9, pp. 75–78, 1997.
- [232] Intel Corporation, “Intel® Optane™ DC Persistent Memory Product Brief,” Intel Corporation, Product Brief, 2019, accessed: 2025-09-30. [Online]. Available: <https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/optane-dc-persistent-memory-brief.pdf>
- [233] S. Lee *et al.*, “Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology : Industrial Product,” in *ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 43–56.
- [234] G. Kim *et al.*, “SK Hynix AI-Specific Computing Memory Solution: From AiM Device to Heterogeneous AiMX-xPU System for Comprehensive LLM Inference,” in *IEEE Hot Chips 36 Symposium (HCS)*, 2024, pp. 1–26.
- [235] Y. Wang *et al.*, “30.6 Vecim: A 289.13GOPS/W RISC-V Vector Co-Processor with Compute-in-Memory Vector Register File for Efficient High-Performance Computing,” in *IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67, 2024, pp. 492–494.
- [236] W. A. Simon *et al.*, “BLADE: An in-Cache Computing Architecture for Edge Devices,” *IEEE Transactions on Computers*, vol. 69, no. 9, pp. 1349–1363, 2020.
- [237] S. Myung *et al.*, “A 14 nm MRAM-Based Multi-bit Analog In-Memory Computing With Process-Variation Calibration for 72 Macros-Based Accelerator,” *IEEE Solid-State Circuits Letters (L-SSC)*, vol. 7, pp. 295–298, 2024.

- [238] M. Caon *et al.*, “Scalable and RISC-V Programmable Near-Memory Computing Architectures for Edge Nodes,” *IEEE Transactions on Emerging Topics in Computing*, pp. 1–15, 2025.
- [239] Q. Si *et al.*, “Optimizing Behavioral Near On-Chip Memory Computing Systems,” in *IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2022, pp. 27–33.
- [240] S.-E. Hwang *et al.*, “HeNCoG: A Heterogeneous Near-memory Computing Architecture for Energy Efficient GCN Acceleration,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2024, pp. 1–5.
- [241] S. Gupta *et al.*, “THRIFTY: Training with Hyperdimensional Computing across Flash Hierarchy,” in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–9.
- [242] V. S. Mailthody *et al.*, “DeepStore: In-Storage Acceleration for Intelligent Queries,” in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 224–238.
- [243] J. Sun *et al.*, “Analog or Digital In-Memory Computing? Benchmarking Through Quantitative Modeling,” in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–9.
- [244] K. Park *et al.*, “A 701.7 TOPS/W Compute-in-Memory Processor With Time-Domain Computing for Spiking Neural Network,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 72, no. 1, pp. 25–35, 2025.
- [245] A. Agrawal *et al.*, “X-SRAM: Enabling In-Memory Boolean Computations in CMOS Static Random Access Memories,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 12, pp. 4219–4232, 2018.
- [246] N. Hajinazar *et al.*, “SIMDRAM: a framework for bit-serial SIMD processing using DRAM,” in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS ’21)*, 2021, pp. 329–345.
- [247] H. Kim *et al.*, “Colonnade: A Reconfigurable SRAM-Based Digital Bit-Serial Compute-In-Memory Macro for Processing Neural Networks,” *IEEE Journal of Solid-State Circuits*, vol. 56, no. 7, pp. 2221–2233, 2021.
- [248] J.-s. Seo *et al.*, “Digital Versus Analog Artificial Intelligence Accelerators: Advances, trends, and emerging designs,” *IEEE Solid-State Circuits Magazine*, vol. 14, no. 3, pp. 65–79, 2022.
- [249] H. Valavi *et al.*, “A 64-Tile 2.4-Mb In-Memory-Computing CNN Accelerator Employing Charge-Domain Compute,” *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, 2019.

- [250] Y.-J. Chang *et al.*, “Multi-Logic Sense Amplifier (MLSA) Design for In-Memory Computing,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 1, pp. 371–381, 2023.
- [251] H. Zhang *et al.*, “WDVR-RAM: A 0.25–1.2 V, 2.6–76 POPS/W Charge-Domain In-Memory-Computing Binarized CNN Accelerator for Dynamic AIoT Workloads,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 10, pp. 3964–3977, 2023.
- [252] X. Liu *et al.*, “HNMC: Hybrid Near-Memory Computing Circuit for Neural Network Acceleration,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 11, pp. 4763–4767, 2024.
- [253] S. Yin *et al.*, “XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks,” *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, 2020.
- [254] M. E. Sinangil *et al.*, “A 7-nm Compute-in-Memory SRAM Macro Supporting Multi-Bit Input, Weight and Output and Achieving 351 TOPS/W and 372.4 GOPS,” *IEEE Journal of Solid-State Circuits*, vol. 56, no. 1, pp. 188–198, 2021.
- [255] Z. Jiang *et al.*, “C3SRAM: An In-Memory-Computing SRAM Macro Based on Robust Capacitive Coupling Computing Mechanism,” *IEEE Journal of Solid-State Circuits*, vol. 55, no. 7, pp. 1888–1897, 2020.
- [256] S. Aga *et al.*, “Compute Caches,” in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 481–492.
- [257] M. F. Ali *et al.*, “In-Memory Low-Cost Bit-Serial Addition Using Commodity DRAM Technology,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 1, pp. 155–165, 2020.
- [258] P. Meihar *et al.*, “Ferroelectric MirrorBit-Integrated Field-Programmable Memory Array for the TCAM, Storage, and In-Memory Computing Applications,” *IEEE Transactions on Electron Devices*, vol. 71, no. 5, pp. 2957–2962, 2024.
- [259] M. Benkhelifa *et al.*, “Dual-Bit FeFET for enhanced storage and endurance,” *npj Unconventional Computing*, vol. 2, no. 16, 2025.
- [260] A. Yusuf *et al.*, “Domain-Specific STT-MRAM-Based In-Memory Computing: A Survey,” *IEEE Access*, vol. 12, pp. 28 036–28 056, 2024.
- [261] L. Ke *et al.*, “Near-Memory Processing in Action: Accelerating Personalized Recommendation With AxDIMM,” *IEEE Micro*, vol. 42, no. 1, pp. 116–127, 2022.

- [262] A. Ankit *et al.*, “PUMA: A Programmable Ultra-efficient Memristor-based Accelerator for Machine Learning Inference,” in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 715–731.
- [263] J.-S. Seo, “Advances and Trends on On-Chip Compute-in-Memory Macros and Accelerators,” in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.
- [264] J. Lin *et al.*, “SongC: A Compiler for Hybrid Near-Memory and In-Memory Many-Core Architecture,” *IEEE Transactions on Computers*, vol. 73, no. 10, pp. 2420–2433, 2024.
- [265] B. Zhang *et al.*, “A Novel CONV Acceleration Strategy Based on Logical PE Set Segmentation for Row Stationary Dataflow,” *IEEE Transactions on Computers*, vol. 71, no. 6, pp. 1466–1478, 2022.
- [266] J. Sim *et al.*, “An Energy-Efficient Deep Convolutional Neural Network Inference Processor With Enhanced Output Stationary Dataflow in 65-nm CMOS,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 87–100, 2020.
- [267] C. Park *et al.*, “Roofline-Model-Based Design Space Exploration for Dataflow Techniques of CNN Accelerators,” *IEEE Access*, vol. 8, pp. 172 509–172 523, 2020.
- [268] N. Chauvaux *et al.*, “An Event-Based Digital Compute-In-Memory Accelerator with Flexible Operand Resolution and Layer-Wise Weight/Output Stationarity,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2025, pp. 1–5.
- [269] S. Liu *et al.*, “XNORAM: An Efficient Computing-in-Memory Architecture for Binary Convolutional Neural Networks with Flexible Dataflow Mapping,” in *2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2020, pp. 21–25.
- [270] C. Wang *et al.*, “EWS: An Energy-Efficient CNN Accelerator With Enhanced Weight Stationary Dataflow,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 7, pp. 3478–3482, 2024.
- [271] Y.-H. Chen *et al.*, “Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, 2019.
- [272] Y. Qiu *et al.*, “Quartet: A Digital Compute-in-Memory Versatile AI Accelerator With Heterogeneous Tensor Engines and Off-Chip-Less Dataflow,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–14, 2025.

- [273] A. Raha *et al.*, “FlexNPU: a dataflow-aware flexible deep learning accelerator for energy-efficient edge devices,” *Frontiers in High Performance Computing*, vol. 3, Jun. 2025.
- [274] H. Zhang *et al.*, “WDVR-RAM: A 0.25–1.2 V, 2.6–76 POPS/W Charge-Domain In-Memory-Computing Binarized CNN Accelerator for Dynamic AIoT Workloads,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 10, pp. 3964–3977, 2023.
- [275] S. Corda *et al.*, “NMPO: Near-Memory Computing Profiling and Offloading,” in *24th Euromicro Conference on Digital System Design (DSD)*, 2021, pp. 259–267.
- [276] A. Coluccio *et al.*, “Hybrid-SIMD: A Modular and Reconfigurable Approach to Beyond von Neumann Computing,” *IEEE Transactions on Computers*, vol. 71, no. 9, pp. 2287–2299, 2022.
- [277] J. Lim *et al.*, “Near-Memory Computing With Compressed Embedding Table for Personalized Recommendation,” *IEEE Transactions on Emerging Topics in Computing*, vol. 12, no. 3, pp. 938–951, 2024.
- [278] Y. Ju *et al.*, “A Differentiable Neural Computer for Logic Reasoning with Scalable Near-Memory Computing and Sparsity Based Enhancement,” in *IEEE European Solid State Circuits Conference (ESSCIRC)*, 2022, pp. 93–96.
- [279] Z. Xuan *et al.*, “A Brain-Inspired ADC-Free SRAM-Based In-Memory Computing Macro With High-Precision MAC for AI Application,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 4, pp. 1276–1280, 2023.
- [280] Z. Lin *et al.*, “In Situ Storing 8T SRAM-CIM Macro for Full-Array Boolean Logic and Copy Operations,” *IEEE Journal of Solid-State Circuits*, vol. 58, no. 5, pp. 1472–1486, 2023.
- [281] E. Ferro *et al.*, “A Precision-Optimized Fixed-Point Near-Memory Digital Processing Unit for Analog In-Memory Computing,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2024, pp. 1–5.
- [282] A. Mouallem *et al.*, “1T1R In-Memory Compute for Winner Takes All Application in Kohonen Neural Networks,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022, pp. 1561–1565.
- [283] Z. Wang *et al.*, “Infinity Stream: Enabling Transparent and Automated In-Memory Computing,” *IEEE Transactions on Computers*, vol. 71, no. 2, pp. 85–88, 2022.
- [284] M. Anderson *et al.*, “First-Generation Inference Accelerator Deployment at Facebook,” *arXiv:2107.04140 [cs.AR]*, 2021.

- [285] S. Maity *et al.*, “Unguided Machine Learning-Based Computation Offloading for Near-Memory Processing,” in *38th International Conference on VLSI Design and 23rd International Conference on Embedded Systems (VLSID)*, 2025, pp. 540–545.
- [286] A. M. Kumar *et al.*, “MARVEL: An End-to-End Framework for Generating Model-Class Aware Custom RISC-V Extensions for Lightweight AI,” *arXiv:2508.01800 [cs.AR]*, 2025.
- [287] S. Machetti *et al.*, “X-HEEP: An Open-Source, Configurable and Extendible RISC-V Microcontroller for the Exploration of Ultra-Low-Power Edge Accelerators,” *arXiv:2401.05548 [cs.AR]*, 2024.