LUND UNIVERSITY

3D Integration Technology and Near-Memory Computing for Edge AI

Prieto, Arturo

2025

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Prieto, A. (2025). *3D Integration Technology and Near-Memory Computing for Edge AI*. Electrical and Information Technology, Lund University.

Total number of authors:
1

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

Download date: 02. Jul. 2026

3D Integration Technology and Near-Memory Computing for Edge AI

Doctoral Thesis

Arturo Prieto

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, 12 December, 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 Dr.-Ing. Dietmar Fey, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany.

Organisation: LUND UNIVERSITY Department of Electrical and Information Technology Ole Römers Väg 3 223 63 Lund Sweden	*Document Type:* DOCTORAL THESIS
	Date of Issue: November 2025
Author: Arturo Prieto	*Sponsoring Organisation(s):* EU Horizon 2020 3D-MUSE Chips-JU REBECCA (101097224) SSF Research Center classIC

Title:
3D Integration Technology and Near-Memory Computing for Edge AI

Abstract:
In the era of artificial intelligence of things (AIoT), distributed processing on local devices is growing in popularity. This stems from the need to reduce data transfer to and from central servers, with concerns about privacy and latency encouraging the processing of AI applications on edge devices. However, the computational demands of these applications require an increase in the processing capabilities of resource-constrained edge devices with reduced memory and energy deployment. In this thesis, solutions for improving edge AI implementations are evaluated considering two approaches: technology integration and hardware architecture design.

Higher performance through increased technology integration has focused on scaling transistor dimensions. However, the manufacturing process is increasingly expensive and faces technical challenges in the development of new breakthroughs. Evaluation of the third dimension has emerged as a promising alternative to scaling, which enables stacking of semiconductor components with 3D interconnections. Different technologies present different integration strategies, where 3D sequential integration (3DSI) enables small pitch for 3D contacts, allowing for high-integration circuits. A library of standard cells has been designed and characterized according to 3DSI, enhancing the high-integration capabilities of the technology for digital designs. This library compiles the required predefined logic cells that can be used in the design of a digital integrated circuit (IC).

The design of ICs as a foundation for edge AI is focused on enhancing memory and computing resources to improve the processing capabilities of such platforms. Computing architectures are traditionally based on the concept of von Neumann architecture, which distinguishes computing and memory units as two independent entities. However, near-memory computing (NMC) is presented as a viable alternative to the von Neumann architecture that brings computation closer to memory. NMC is non-intrusive to the conventional low-level structure of SRAM and enhances memory bandwidth for hardware acceleration. The integration of accelerators into resource-constrained platforms has been evaluated, expanding the functionality with custom hardware tailored for computation-intensive AI workloads. Furthermore, flexibility has been achieved by providing modularity to the design architecture.

The proposed architectures are evaluated by programs that highlight the performance of integrated AI hardware accelerators into edge devices, emphasizing the importance of software and hardware co-design. The contributions of this thesis focus on 3DSI technology circuit design and NMC architectures evaluating performance, energy and area efficiency.

Keywords:
3D Integration Technology, Near-Memory Computing, Edge Computing, Artificial Intelligence, More-than-Moore, Convolutional Neural Network, Hardware Acceleration, SRAM

Classification System and/or Index Terms –		*Language:* English
Supplementary Bibliographical Information: –		*ISBN (printed):* 978-91-8104-745-5
Key title and ISSN: Series of Licentiate and Doctoral Theses; 1654-790X, No. 190		*ISBN (digital):* 978-91-8104-746-2
Recipient's Notes:	*Number of Pages:* 180	*Price:*
	Security Classification: 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: *Arturo Prieto* *Date:* 12 November 2025

3D Integration Technology and Near-Memory Computing for Edge AI

Doctoral Thesis

Arturo Prieto

LUND
UNIVERSITY

Department of Electrical and
Information Technology
Lund, November 2025

Arturo Prieto
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. 190
ISBN 978-91-8104-745-5 (printed)
ISBN 978-91-8104-746-2 (digital)

© 2025 Arturo Prieto
This thesis is typeset using LaTeX 2ε.

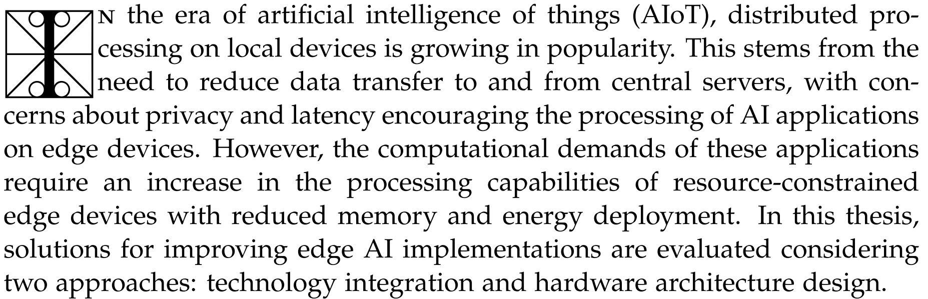
Frontispiece: *Expansion of an Integrated Circuit.*

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.

To my Dad

Abstract



In the era of artificial intelligence of things (AIoT), distributed processing on local devices is growing in popularity. This stems from the need to reduce data transfer to and from central servers, with concerns about privacy and latency encouraging the processing of AI applications on edge devices. However, the computational demands of these applications require an increase in the processing capabilities of resource-constrained edge devices with reduced memory and energy deployment. In this thesis, solutions for improving edge AI implementations are evaluated considering two approaches: technology integration and hardware architecture design.

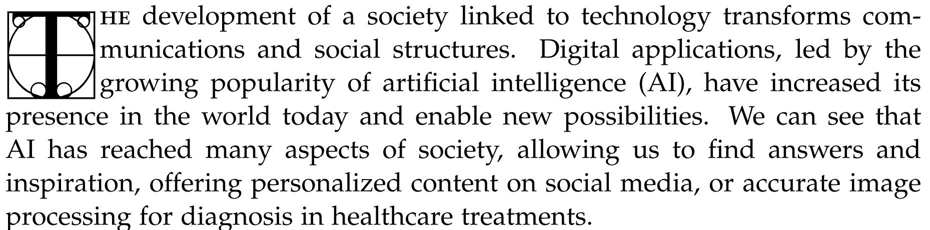
Higher performance through increased technology integration has focused on scaling transistor dimensions. However, the manufacturing process is increasingly expensive and faces technical challenges in the development of new breakthroughs. Evaluation of the third dimension has emerged as a promising alternative to scaling, which enables stacking of semiconductor components with 3D interconnections. Different technologies present different integration strategies, where 3D sequential integration (3DSI) enables small pitch for 3D contacts, allowing for high-integration circuits. A library of standard cells has been designed and characterized according to 3DSI, enhancing the high-integration capabilities of the technology for digital designs. This library compiles the required predefined logic cells that can be used in the design of a digital integrated circuit (IC).

The design of ICs as a foundation for edge AI is focused on enhancing memory and computing resources to improve the processing capabilities of such platforms. Computing architectures are traditionally based on the concept of von Neumann architecture, which distinguishes computing and memory units as two independent entities. However, near-memory computing (NMC) is presented as a viable alternative to the von Neumann architecture

that brings computation closer to memory. NMC is non-intrusive to the conventional low-level structure of SRAM and enhances memory bandwidth for hardware acceleration. The integration of accelerators into resource-constrained platforms has been evaluated, expanding the functionality with custom hardware tailored for computation-intensive AI workloads. Furthermore, flexibility has been achieved by providing modularity to the design architecture.

The proposed architectures are evaluated by programs that highlight the performance of integrated AI hardware accelerators into edge devices, emphasizing the importance of software and hardware co-design. The contributions of this thesis focus on 3DSI technology circuit design and NMC architectures evaluating performance, energy and area efficiency.

Popular Science Summary



THE development of a society linked to technology transforms communications and social structures. Digital applications, led by the growing popularity of artificial intelligence (AI), have increased its presence in the world today and enable new possibilities. We can see that AI has reached many aspects of society, allowing us to find answers and inspiration, offering personalized content on social media, or accurate image processing for diagnosis in healthcare treatments.

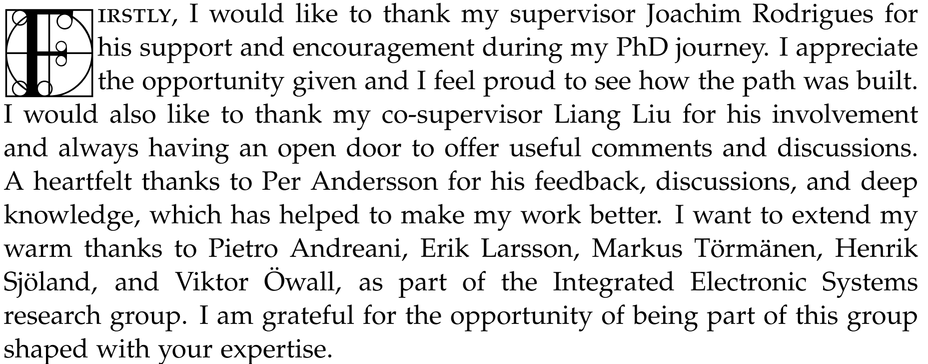
Vast amounts of data and mathematical calculations are required to provide AI applications that support interactions in our daily life with technology. If we want to avoid sending the data to cloud servers, so that we can keep everything on our personal devices ensuring privacy, we need battery-powered devices that can cope with the workload. The technology in our hands is increasing the possibilities in a digitally connected world, with smartphones and electronic devices expanding their capabilities in communication, access to information, or entertainment. However, the extended abilities provided to such systems does not come for free.

Higher energy consumption is the result of more complex tasks and AI applications that can handle diverse functions. The energy consumption of electronic devices is more dominant and has greater relevance in the world today. At the same time, data privacy is a concern, so keeping the processing on local devices avoids privacy breaks due to data transfer, but demands more processing capabilities from battery-powered devices. In order to maintain battery life and still use complex applications, it is necessary to design hardware that can cope with the workload of computation-demanding scenarios. Microchips serve as the foundational pillar for electronic devices that are required for a technology-connected society.

New challenges come as a consequence of the development of digital applications that require more computational power. Concerns about energy use are directed to increase the battery life of electronic devices. However, performance and area cost cannot be overstated, which requires hardware engineers to look into multidisciplinary solutions that cover a wide range of use cases and possibilities.

Specialized hardware needs to be designed to efficiently process the digital applications of the future. For that reason, the evaluation of alternative hardware solutions is necessary. From the technology perspective, microchip integration has conventionally followed a miniaturization strategy to increase density. In this thesis, 3D integration is considered as an alternative to miniaturization to improve efficiency. From a hardware design point of view, memory and computation are conventionally separated. Near-memory computation is proposed in this thesis work as a way to bring data storage and processing closer to each other. I have considered AI examples to evaluate my research contributions and evaluated new integration technologies, together with new hardware solutions, that can efficiently process such applications.

Acknowledgments



Firstly, I would like to thank my supervisor Joachim Rodrigues for his support and encouragement during my PhD journey. I appreciate the opportunity given and I feel proud to see how the path was built. I would also like to thank my co-supervisor Liang Liu for his involvement and always having an open door to offer useful comments and discussions. A heartfelt thanks to Per Andersson for his feedback, discussions, and deep knowledge, which has helped to make my work better. I want to extend my warm thanks to Pietro Andreani, Erik Larsson, Markus Törmänen, Henrik Sjöland, and Viktor Öwall, as part of the Integrated Electronic Systems research group. I am grateful for the opportunity of being part of this group shaped with your expertise.

I am very grateful to have shared this PhD journey with Masoud. I enjoyed working together, but especially the friendship and personal relationship we built, extending my thanks to Naghmeh. Thanks to Lucas, I am glad to have met you during the master and all the moments we shared together that I keep as good memories. Thanks to MJ for the interesting discussions that I always cherished during my visits to your office.

My heartfelt appreciation to my friends in the large room. Your relentless work has been an inspiration, but above all I appreciate the afterwork and personal time we shared together. To Kristoffer for all the good laughs and chats, to Sergio for organizing fantastic dance events, to Alex and his deep knowledge of beer, to William for every lunch, and to Linus for the time we shared together.

I would like to thank Babak and Hemanth for inspiring me to take this journey. To Victor for his knowledge and feedback that has helped me improve my work. Thanks to Baktash, Iman and Hamid for the lunches we had together, and I would also like to thank Rikard, Mojtaba, Steffen, Jesús,

Ilayda, Sidra, Dumitra, Lina, Vilgot, Sijia, Love, Josep, Giovani, Wenbo, Joel, Siju, and MinKeun for the time we shared together at the department.

I appreciate the work carried out by Sirvan at the lab, in exchange for some lessons of Spanish. Furthermore, I would like to thank Elisabeth Nordström, Elisabeth Ohlsson, Linda Bienen, Margit Billesö, Erik Göthe, Erik Jonsson and Stefan Molund for your dedication and for making it possible for us to focus on our work. I extend my thanks to Stefan Höst, Daniel Sjöberg and Michael Lentmaier for your work at the department, and the environment you create with your efforts.

Thanks to Juan and Laura, I am very grateful to have you in my life. To Mahdi, thank you for bringing boundless energy every time we meet. To Umar for the good times together. Thanks to Yeilse for your friendship and the moments of disconnection we shared. I cherish all the evenings spent drinking tea with Kinna, and I am very grateful to have shared a corner with Caty and Berta before starting this PhD journey.

Finally, I am grateful for the unconditional support from my family. I have followed in the footsteps of the role model that I saw in my dad. I always felt the support of my mum, and her desire for me to come back home. Thanks to my sister Marta for her words of encouragement when I needed them most. Thanks also to Andrés, and my nieces Sofía and Sara for brightening every moment. I would like to thank everyone who has surrounded me from Valencia to Lund, and has influenced me to become the person I am today.

Arturo
Lund, November 2025

Contents

Abstract **v**

Popular Science Summary **vii**

Acknowledgments **ix**

Contents **xi**

Preface **xv**

Structure of the Thesis xv

Included Papers xvi

Related Work xvii

Acronyms and Abbreviations **xix**

. . . . xix

INTRODUCTION **1**

1 Motivation **3**

1.1 Technology in Everyday Life 3

1.2 Compute-intensive Applications 4

1.3 Cloud and Edge Computing 5

1.4 Semiconductor Value Chain 6

1.5 Thesis Focus 7

2 Background 9
2.1 Artificial Intelligence . . . 9
2.1.1 Applications . . . 10
2.1.2 Convolutional Neural Networks . . . 11
2.2 Moore's Law . . . 13
2.2.1 More-than-Moore and More-Moore . . . 14
2.3 Memory . . . 14
2.3.1 Computer Memory Hierarchy . . . 15
2.3.2 Memory Wall . . . 16
2.4 von Neumann Architecture . . . 17
3 Integration Technologies 19
3.1 CMOS . . . 19
3.2 3D Stacking . . . 21
3.2.1 Silicon Interposer . . . 22
3.2.2 3D Bonding . . . 22
3.3 3D Sequential Integration . . . 23
3.3.1 Integration Granularity . . . 25
3.3.2 Challenges . . . 26
3.4 Thesis Contribution . . . 26
4 Hardware Architecture 29
4.1 Processing Platforms . . . 29
4.1.1 General-Purpose . . . 30
4.1.2 FPGA . . . 31
4.1.3 ASIC . . . 31
4.2 Hardware Accelerators . . . 32
4.2.1 Co-Integration . . . 33
4.3 Data-centric Computing . . . 34
4.3.1 In-Memory Computing . . . 35
4.3.2 Near-Memory Computing . . . 36
4.4 Thesis Contribution . . . 37
5 Conclusion and Future Work 39
Bibliography 41

PAPERS 51

I High-Density Standard Cell Library for Sequential 3D Integrated Circuits 55

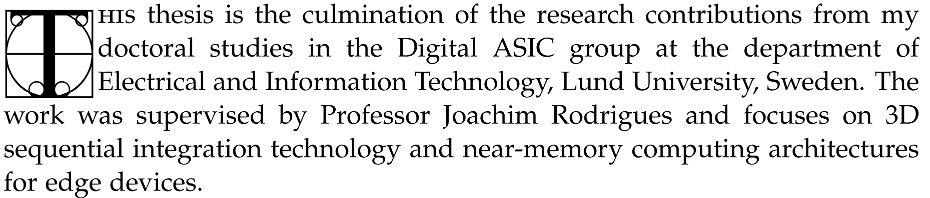
II An Energy-Efficient Near-Memory Computing Architecture for CNN Inference at Cache Level 69

III A Scalable All-Digital Near-Memory Computing Architecture for Edge AIoT Applications 83

IV NMCu-CNN: A Scalable Near-Memory Computing Co-Processor on a RISC-V MCU in 22 nm FDSOI 127

V LUCIA: Compute Inter-chip Architecture for Multi-chiplet Acceleration in 22 nm FDSOI 143

Preface



THIS thesis is the culmination of the research contributions from my doctoral studies in the Digital ASIC group at the department of Electrical and Information Technology, Lund University, Sweden. The work was supervised by Professor Joachim Rodrigues and focuses on 3D sequential integration technology and near-memory computing architectures for edge devices.

STRUCTURE OF THE THESIS

This thesis is structured in two parts. The first part is an introductory section that provides a summary of the research field. The second part is a collection of the research papers that condense my contribution to the field.

- **INTRODUCTION**
  A broader and more comprehensive view of the areas explored during my doctoral studies. It presents the main topics and challenges considered in the papers, connects the work together, and includes a contextualization of my contributions.

- **PAPERS**
  The papers forming the main body of the thesis are reproduced in the second part and listed in the following order, including an outline of my individual contributions.

INCLUDED PAPERS

The following papers form the main body of this thesis and the respective published or draft versions are appended in the back.

Paper I: A. Prieto and J. Rodrigues, "High-Density Standard Cell Library for Sequential 3D Integrated Circuits", *IFIP/IEEE 32nd International Conference on Very Large Scale Integration (VLSI-SoC)*, Oct. 2024, pp. 1-4.

Contribution: I proposed the design of logic cells using 3D sequential integration technology and evaluated their implementation on benchmark circuits, writing the paper under the supervision of the second author, and presenting a poster at the conference.

Paper II: M. Nouripayam, A. Prieto, V. K. Kishorelal and J. Rodrigues, "An Energy-Efficient Near-Memory Computing Architecture for CNN Inference at Cache Level", *28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, Nov. 2021, pp. 1-4.

Contribution: I defined specifications of the architecture, developed the design and performed simulations of the system, co-authoring the paper together with the first author, and presenting the work at the conference.

Paper III: M. 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.

Contribution: I defined specifications of the architecture, developed the design and performed simulations of the system, co-authoring the paper together with the first author.

Paper IV: A. Prieto, M. Nouripayam, K. Westring, S. Castillo Mohedano, A. Allfjord, L. Svensson, P. Andersson and J. Rodrigues, "NMCu-CNN: A Scalable Near-Memory Computing Co-Processor on a RISC-V MCU in 22 nm FDSOI", *IEEE Workshop on Signal Processing Systems (SiPS)*, 2025.

► *Accepted and presented.*

Contribution: I defined specifications of the architecture and developed the design, co-authoring the paper together with the second author, and presenting the work at the conference. ASIC implementation and measurements were accomplished in collaboration with the other authors.

Paper V: A. Prieto, M. Nouripayam, K. Westring, W. Marnfeldt, S. Castillo Mohedano, A. Allfjord, L. Svensson, V. Åberg, P. Andersson and J. Rodrigues, "LUCIA: Compute Inter-chip Architecture for Multi-chiplet Acceleration in 22 nm FDSOI".

▶ *Submitted for review.*

Contribution: I defined specifications of the architecture, developed the design and wrote the paper under the supervision of the last author. ASIC implementation and measurements were accomplished in collaboration with the other authors.

RELATED WORK

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

Paper vi: M. Nouripayam, S. Castillo Mohedano, A. Prieto and J. Rodrigues, "Scalable Hardware-Efficient Approximate Multipliers for Error-Resilient Applications".

▶ *Submitted for review.*

Paper vii: A. Brokalakis, P. Malakonakis, K. Harteros, D. Andronikou, I. Galanomatis, C. Savvakos, G. Tang, K. Vadivel, G.-J. van Schaik, S. Ioannidis, K. Georgopoulos, G. Chrysos, A. Allfjord, M. Nouripayam, A. Prieto, J. Rodrigues, I. Mavroidis and I. Papaefstathiou, "REBECCA: Reconfigurable Heterogeneous Highly Parallel Processing Platform for safe and secure AI".

▶ *Submitted for review.*

Acronyms and Abbreviations

3DSI	3D Sequential Integration
ADC	Analog-to-Digital Converter
AI	Artificial Intelligence
AIMC	Analog In-Memory Computing
AIoT	Artificial Intelligence of Things
ALU	Arithmetic Logic Unit
AR	Augmented Reality
ASIC	Application-Specific Integrated Circuit
ASIP	Application-Specific Integrated Processor
BEOL	Back End of Line
CMOS	Complementary Metal-Oxide-Semiconductor
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DIMC	Digital In-Memory Computing
DL	Deep Learning
DRAM	Dynamic Random-Access Memory

EDA	Electronic Design Automation
FDSOI	Fully Depleted Silicon-on-Insulator
FEOL	Front End of Line
FPGA	Field-Programmable Gate Array
GP	Ground Plane
GPP	General-Purpose Processor
GPU	Graphics Processing Unit
HBM	High-Bandwidth Memory
HPC	High Performance Computing
IC	Integrated Circuit
IMC	In-Memory Computing
IoT	Internet of Things
ISA	Instruction Set Architecture
M3D	Monolithic 3D
MAC	Multiply-Accumulate
MCU	Microcontroller Unit
MIV	Monolithic Inter-tier Via
ML	Machine Learning
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
NMC	Near-Memory Computing
NMOS	N-channel Metal-Oxide-Semiconductor
NN	Neural Network
NVM	Non-Volatile Memory
PE	Processing Element
PMOS	P-channel Metal-Oxide-Semiconductor
RAM	Random-Access Memory

RISC Reduced Instruction Set Computer

RRAM Resistive Random-Access Memory

SoA State-of-the-Art

SRAM Static Random-Access Memory

TSV Through-Silicon Via

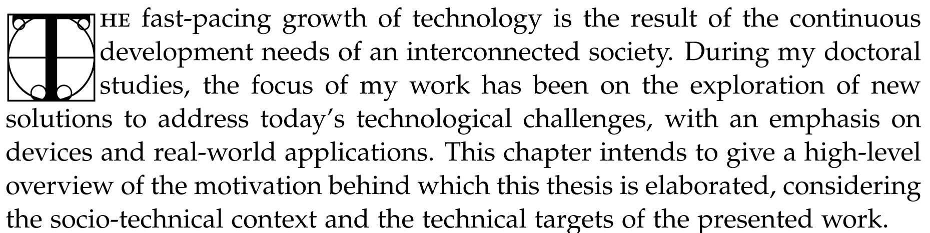
VGG Visual Geometry Group

VR Virtual Reality

INTRODUCTION

1

Motivation



The fast-pacing growth of technology is the result of the continuous development needs of an interconnected society. During my doctoral studies, the focus of my work has been on the exploration of new solutions to address today's technological challenges, with an emphasis on devices and real-world applications. This chapter intends to give a high-level overview of the motivation behind which this thesis is elaborated, considering the socio-technical context and the technical targets of the presented work.

1.1 TECHNOLOGY IN EVERYDAY LIFE

During the past three decades, the influence of technological advances in the digital world has become an intrinsic element of society [1]. Computers, smartphones, and digital services are fundamental to everyday life in modern society [2]. However, social structures and processes affect the progress of technologies, driving innovation changes that occur as a social process [3]. The growing interdependency between technology and society demands the development of platforms that fit the needs of daily life to interact with the world around us.

From telecommunications to healthcare, automotive, and computing systems, microelectronics serves as an underlying principle of technology that shapes modern life. In an increasingly interconnected and technology-driven world, the importance of microelectronics has become a foundational pillar of innovation [4]. Semiconductor materials support the fabrication of basic microelectronic components, such as transistors, whose evolution and progression have led to nanometric dimensions and large volume integrations [5].

The evolution of physical components leads to needs in the microelectronics sector and brings new challenges. At the same time, digital applications are becoming more computationally intensive, increasing the requirements of performance-enhanced platforms, with energy consumption emerging as a more topical challenge. Performance improvement through technology scalability faces physical limitations, and alternative integration strategies are considered to overcome these constraints.

1.2 COMPUTE-INTENSIVE APPLICATIONS

The growth of digital services with a variety of compositions, and the data generated as a result, demands solutions that specialize in data-centric processing platforms that handle a high intensity of computations [6]. A pivotal service embracing digital transformation is artificial intelligence (AI), which is becoming increasingly popular and covers a wide range of use cases. The deployment of efficient computing as a result of the growth of AI applications is a challenge to address. Vast volumes of data and mathematical operations are required to generate AI applications that are used in different aspects of society. Similarly, the metaverse, which encompasses augmented reality (AR) and virtual reality (VR) experiences, is leading to highly distributed and tightly integrated compute-intensive networks with large datasets for next-generation digital experiences [7]. Such applications have become increasingly complex, basing their topology on the processing of large datasets to define their functionality.

Big data is the term used to describe large volumes of data that require advanced techniques and technologies to store, distribute, control, and analyze information [8]. Big data extracts knowledge by processing the correlation between data and providing an effective decision-making capacity in healthcare, finance, or social media applications [9]. The vast amounts of data generated by different users and platforms require solutions with different processing properties. In that respect, AI is presented as an application that offers suitable analysis by processing large datasets. With the use of AI in a wider range of conditions, polyvalent platforms that compute efficiently under different circumstances are crucial.

Furthermore, higher computational requirements by digital applications used in everyday life demand platforms with enhanced computational capabilities. Compute-intensive applications are the result of more complex tasks that perform more operations and process larger amounts of data, thereby making the invested processing power more significant. Different types of processing platforms are available for compute-intensive applications, ranging from local edge devices to central cloud servers.

1.3 CLOUD AND EDGE COMPUTING

Digital applications can be processed on different types of platforms. Depending on the processing location, devices are grouped as cloud and edge devices, as shown in Figure 1.1. Cloud processing is performed on remote servers with high storage and computing capabilities, while edge devices have limited storage and energy budget. Processing on edge devices keeps data close to the source, addressing latency and privacy issues derived from data transfer between the source and the cloud.

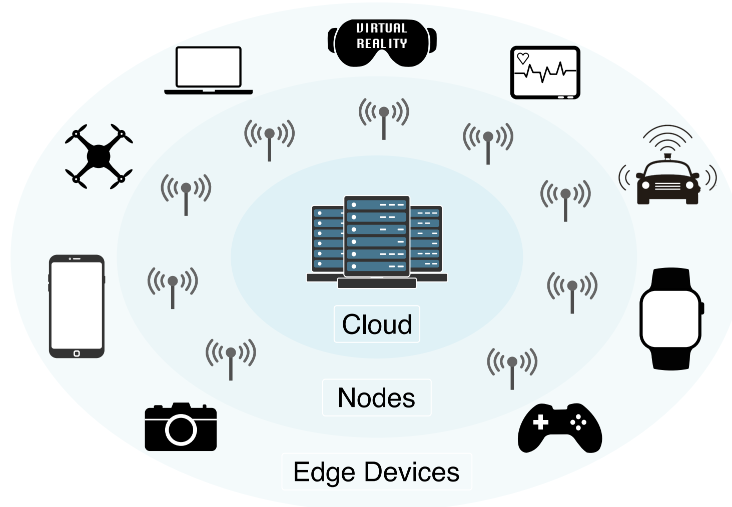


Figure 1.1: Processing platforms based on cloud and edge devices, including connection nodes.

The internet of things (IoT) is a network of connected devices with embedded sensors that collect information. Edge IoT involves processing data locally, and includes devices with integrated sensors and chips with various capabilities for different applications that are used to facilitate smart home, smart healthcare, or smart cities [10].

Processing AI applications on edge devices has emerged as an alternative to transferring large amounts of data across the network, considering transmission delay, cost, and privacy leakage concerns [11]. However, edge devices span resource-limited computing systems. The combination of AI applications with edge platforms results in Edge AI, linking intelligence applications to a broad collection of connected systems [12]. Bringing AI to the edge enables innovative and useful applications in local devices and alleviates bandwidth demand on central cloud servers [13]. However, building applica-

tions for edge devices, characterized by resource-constrained hardware architectures limiting intensive processing, implies higher complexity compared to building them for the cloud. Performance and energy efficiency are presented as challenges for hardware designers to improve such platforms, which can be done at different stages of the semiconductor value chain.

1.4 SEMICONDUCTOR VALUE CHAIN

The semiconductor value chain represents the consecutive stages that go into the process of creating a microchip, from tools to design, manufacturing technology, assembly, and software applications [14], as shown in Figure 1.2. Each step in the value chain includes a large number of actors that play a role in the process. The semiconductor value chain is at the core of the digital economy considering its highly interconnected influence and capital intensive properties [15].

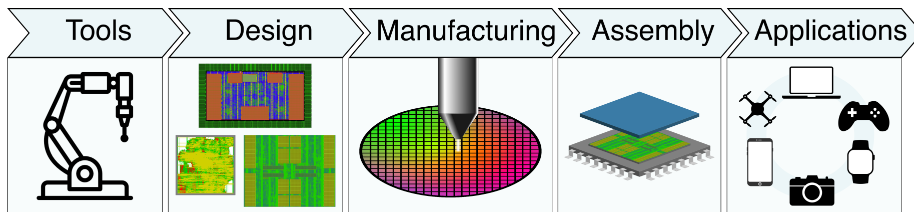


Figure 1.2: Semiconductor value chain.

Microchips can be found everywhere, from smartphones, computers, or cars to gaming consoles, image sensors, and domestic appliances, which constitutes the main technology driver for the continuous digital transformation. The intricate process of the semiconductor industry to create an integrated circuit (IC) covers the following parts in each stage:

1. **Tools**: Software tools for electronic design automation (EDA), available for the generation of chips with a large number of transistors.
2. **Design**: A chip blueprint is sketched according to the specified requirements and functionality.
3. **Manufacturing technology**: Silicon is used as raw material in most chip designs, with increased research in alternative materials for optimized solutions. Manufacturing facilities are specialized in the conversion of the design sketch to physical implementation.
4. **Assembly**: Semiconductor assembly is required to obtain final hardware products.

5. **Applications**: Software programming is performed to define the functionality of manufactured products. The synergy between hardware and software is the key enabler to obtain efficient solutions.

1.5 THESIS FOCUS

The focus of this thesis is mainly on the design and manufacturing technology stages of the semiconductor value chain, with consideration of applications. However, the importance of other stages cannot be overseen, and the included papers have submerged into various parts of the chain for research exploration.

In **Paper I**, the possibilities of 3D integration technology are evaluated as an alternative to scaling. Furthermore, the adaptation of tools for EDA has been considered, since available commercial tools were not automated for the partition of digital designs in 3D ICs.

Papers II and **IV** present chip implementations optimized for edge computing, with simulation and measurement results including relevant comparisons to the state-of-the-art (SoA).

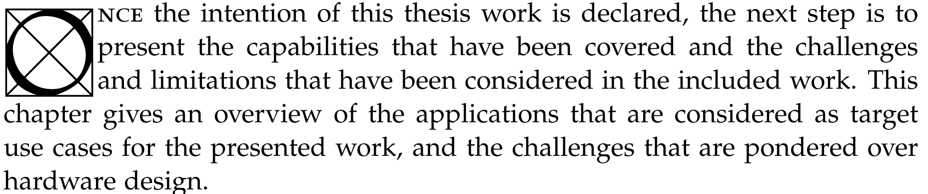
Paper III provides a framework for the efficient processing of AI applications on edge devices. Flexibility and scalability are achieved with a versatile design adapted to the evolving processing requirements of AI.

In **Paper V**, an optimized architecture for multi-chiplet systems is evaluated. The proposed implementation exposes the benefits of parallel and modular processing architectures that enhance the performance of edge devices, avoiding manufacturing limitations.

The design of efficient hardware is ineffective if there is no software that can explore its benefits. For this reason, improvements in hardware and software need to go hand in hand with relevant applications used as benchmark models. The included papers in this thesis have been evaluated with popular use cases, and a detailed analysis of the proposed solutions is presented.

2

Background



ONCE the intention of this thesis work is declared, the next step is to present the capabilities that have been covered and the challenges and limitations that have been considered in the included work. This chapter gives an overview of the applications that are considered as target use cases for the presented work, and the challenges that are pondered over hardware design.

2.1 ARTIFICIAL INTELLIGENCE

AI describes the capability of computing programs to mimic human intelligence and has found utility in multiple use cases. The combination of different fields such as computer science, biology, psychology, philosophy, and many other disciplines produces AI applications that achieve remarkable results in speech recognition, image processing, or natural language processing [16]. The expansion of IoT, with its growing network of links between interconnected sensors and devices, is generating massive volumes of data continuously expanding the global datasphere [17]. The annual size of the global datasphere presented by the international data corporation is shown in Figure 2.1, where it was predicted that 175 ZB of digital data would be created in 2025 and will continue to grow exponentially. Rapid development of AI, together with the necessary analysis of large amounts of information in the era of big data, urges the combination of edge computing and AI to bring applications closer to the data collection source [18].

AI introduced into IoT platforms delivers AIoT, where distributed processing on local devices fulfills intelligent applications [19]. AIoT achieves flexibility with enhanced interactivity between devices for distributed collab-

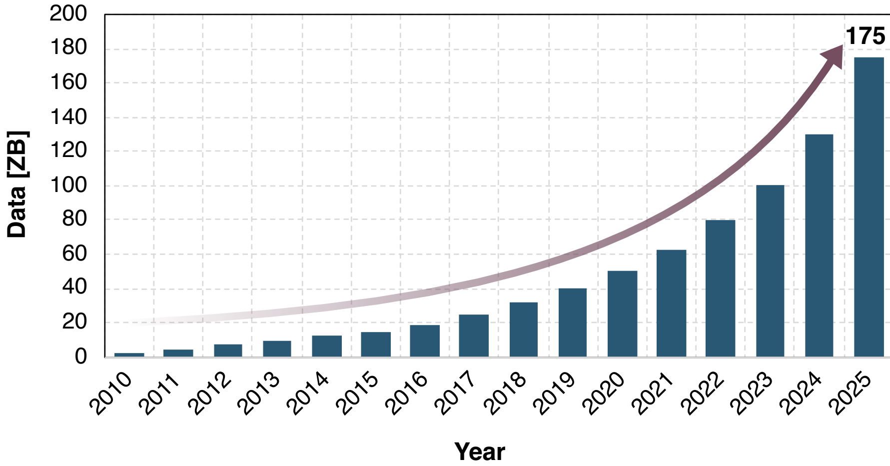


Figure 2.1: Annual size of the global datasphere [20].

oration and the combination of data collection with intelligent applications. AI models learn from the environment and make accurate predictions that are applied to various types and datasets, increasing the utility of AIoT systems. Different applications that are processed on edge benefit from the ability provided by AI models.

2.1.1 APPLICATIONS

The use of AI on edge devices generates a wide range of applications focusing on five main categories: communication networks, healthcare, security, image processing, and voice analysis [21]. The performance of telecommunication systems can be optimized by AI through efficiently distributing resources and enhancing security capabilities to protect data transmission. In addition, AI applications have improved different healthcare procedures by personalizing treatments and increasing the accuracy of the diagnosis, presenting transformative potential for improvement in the field that requires to be balanced with ensured ethical practices [22].

Image and speech processing are highly relevant information sources for human intelligence activities. For that reason, many AI applications focus on image recognition and classification. Image data is provided to computers that can perform mathematical operations to identify patterns and provide analytical results similar to human capabilities.

2.1.2 CONVOLUTIONAL NEURAL NETWORKS

The relationship between the levels of AI applications is shown in Figure 2.2. As part of AI, machine learning (ML) is a group subset characterized by the improvement of applications without explicit programming, mimicking a learning process [23]. According to the premises of ML, deep learning (DL) contains machines that are prepared with representation learning methods based on the identification of data characteristics [24]. At an embedded level of DL, convolutional neural networks (CNNs) are defined as the type of neural network (NN) that mimics brain interconnections.

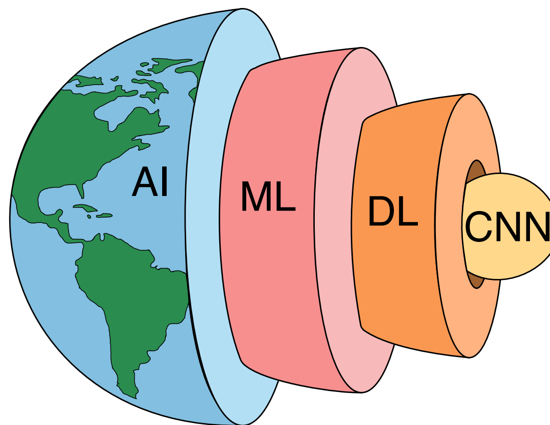


Figure 2.2: Relationship between the different levels under AI that mimic human intelligence.

CNN is a popular type of DL network commonly used in image processing, data analysis, or object detection and classification. The mathematically modeled neurons in a CNN represent brain interconnections as individual computational units that process inputs to generate output data. CNNs are categorized as supervised learning networks in which neurons are trained with labeled data provided for comparison [25]. Based on the predicted result and the ground truth labeled result, the neurons of the network are adjusted, defining the training process to increase prediction accuracy. Training and inference are the two phases required in the use of CNNs:

- **Training**: Initial process to prepare neurons for detection and classification, which requires large amount of data processing and results in self-optimized neurons through learning.
- **Inference**: Subsequent phase where the trained model is prepared for identification of input features and uses the previously defined neurons to make predictions.

The supervision categorization arises from the availability of labeled data that the network uses for comparison and tuning of the neurons to increase the classification accuracy. The generation of datasets as part of supervised learning is a demanding task, since ground truth results need to be provided as labeled data. ImageNet [26] is an example of a labeled dataset with more than 15 million images belonging to 22,000 categories that is available and can be used to prepare NN models for image classification.

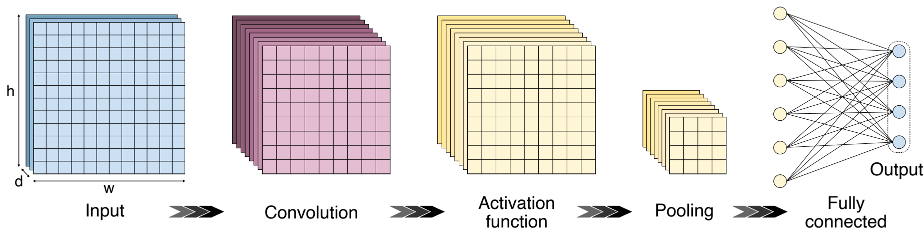


Figure 2.3: CNN dataflow including layer types.

CNNs combine convolutional, activation functions, pooling, and fully connected layers, in addition to the input and output data layers, as shown in Figure 2.3:

- **Input**: Parameters organized into three dimensions according to height (h), width (w), and depth (d).
- **Convolutional**: Layers that perform the mathematical operation of convolution, where filters are shaped in the form of matrices that target the recognition or identification of specific patterns in an input.
- **Activation function**: Application of non-linearity to the data to allow the modeling of complex patterns.
- **Pooling**: Down-sampling mechanism to reduce the size of data between layers.
- **Fully connected**: Layers that serve as a classification of the extracted features, performing matrix multiplication of all input and output neurons to define their relationship.
- **Output**: Establishes the classes specified for the model with results that define the probability of an input to classify in each class.

The parameters, or activations, that define inputs and outputs of each layer use the nomenclature ifmap and ofmap, respectively, while filters are also known as kernels. The filters contain the parameters assigned to neurons that are used in mathematical calculations for data prediction and classification. CNN models include abundant parameters, which require extensive memory footprint and processing capabilities to perform a large number of operations.

A popular fine-tuned CNN model is the 16-layer CNN, which features 13 convolutional layers combined with pooling and activation functions and 3 fully connected layers, developed by the visual geometry group (VGG) named VGG-16 [27]. This work focused on the demonstration of the importance of network depth with concatenated convolutional layers to achieve better recognition and classification [28]. VGG-16, together with MobileNet [29] and ResNet [30], are popular examples of SoA CNN models used for AI evaluation.

2.2 MOORE'S LAW

Gordon Moore postulated in 1965 that the number of transistors would double every two years, coining the term Moore's law [31], as shown in Figure 2.4. The increased requirements of new applications need the integration of more transistors, which demand higher computing capabilities of hardware platforms. Traditionally, scaling down the size of transistors has been the trend that persists in Moore's law. Even when the transistor scaling rate has slowed, the number of transistors continues to increase [32].

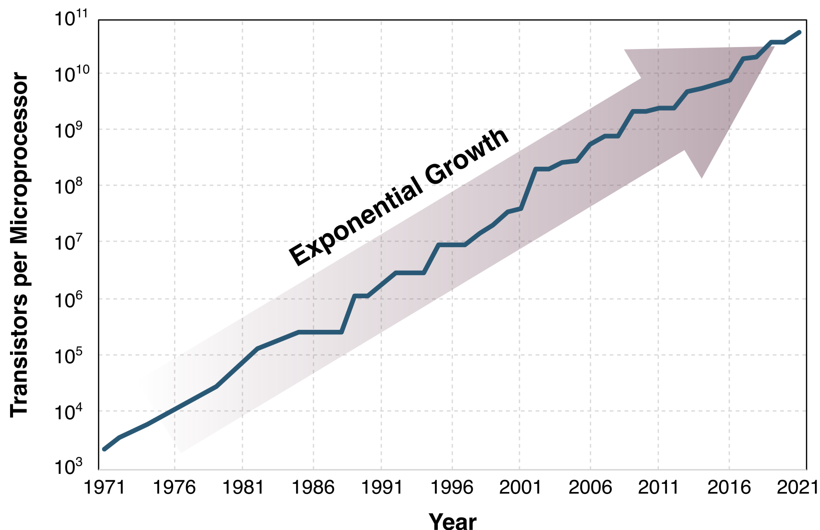


Figure 2.4: Moore's law pondering the progression of the number of transistors per microprocessor [33].

Although Moore's law has demonstrated to prevail true for a long time, the expense of scaling in order to fit larger numbers of transistors has motivated alternative solutions to scaling, referred to more-than-Moore technologies.

More-than-Moore represents the diversification of technologies that intend to represent an alternative to transistor scaling. Alternatives to Moore's law are considered by evaluating new devices for memory and logic, new architectures for computing, new devices that combine memory and logic, or new integration processes [34].

2.2.1 MORE-THAN-MOORE AND MORE-MOORE

More-than-Moore covers emerging technologies that derive from silicon rather than just transistor miniaturization with Moore's law, exploring heterogeneous system solutions focusing on smart sensors, smart energy, and heterogeneous integration of different components in the same package [35]. The expansion of technologies that serve as a basis for platforms with intense requirements for a wide range of applications is a major challenge for more-than-Moore functional diversification. The IoT network of devices certainly motivates the development of emerging technologies with a wide range of cutting-edge applications with uninterrupted energy and computational power requirements [36].

A major trend is the evaluation of the third dimension in ICs to increase the integration of transistors. The limitation of transistor scaling pushes for more-Moore, where 3D integration enables higher transistor density. 3D integration and packaging have evaluated the addition of more layers, allowing vertical interconnection and stacking dies [37].

2.3 MEMORY

In a computing platform, the central processing unit (CPU) is considered the heart of the computer, while memory is the physical space where data and control commands, known as instructions, are located [38]. Memories are fundamental in computing platforms to store data and provide access to enable the execution of computer tasks. In computing platforms, memories are responsible for enabling the access to information with processors moving data into memory and retrieving it for computation. The speed at which CPUs perform calculations is important, as is the memory bandwidth to determine the speed of the overall system. Therefore, processing speed is associated with the data access capabilities that memories provide.

Scaling up the properties of processing platforms to fulfill the requirements of the big data era and high performance computing (HPC) workloads influence memories [39]. The development of heterogeneous processing platforms leads to a higher demand for memory capacity and bandwidth according to the requirements of increasingly popular compute-intensive applications [40].

2.3.1 COMPUTER MEMORY HIERARCHY

Modern computer systems have a memory hierarchy to organize data in different levels, whose position is defined according to their proximity to the processing units [41]. The computer memory hierarchy is established by creating different memory levels with different data locations, as shown in Figure 2.5. A memory system with different levels of abstraction creates a modularized system that allows designers to optimize individual subsystems. Each level of the memory hierarchy has different properties that are defined according to the needs of the computing platforms considering storage capacity, cost, and speed, where closer to a CPU is closer to computational units.

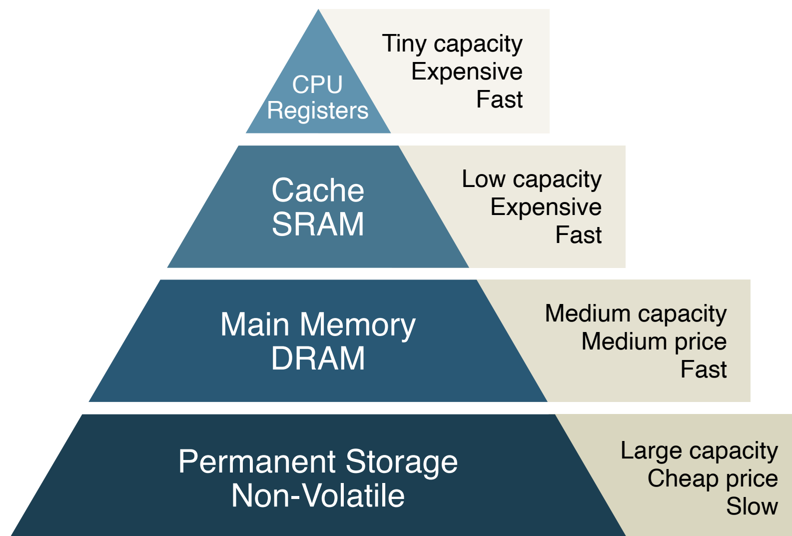


Figure 2.5: Memory levels in computing systems.

The CPU is on top, considered as level zero, using registers as storage elements, which have reduced capacity and allow for fast access. Static random-access memory (SRAM) is used as cache memories in the first level of the memory hierarchy with extended capacity compared to registers and still allowing fast access. Dynamic random-access memory (DRAM) is on the second level and is used as the main memory, presenting more storage space with slower communication than higher levels. Finally, non-volatile memory (NVM) is used as the last level offering high storage capacity at the expense of slower data bandwidth. NVMs keep the stored data even when the machine is shut down, as opposed to register and random-access memories (RAMs) that are used as temporary storage of active data.

The memory hierarchy helps to optimize data access and reuse in computer systems, improving system efficiency. However, memory still faces speed limitations compared to processing units.

2.3.2 MEMORY WALL

The memory wall is a limitation produced by the difference in speed between the processing and memory devices [42]. Figure 2.6 presents the progression of processor speed and DRAM bandwidth of different generations, indicating the performance gap known as the memory wall. The different levels of memory hierarchy help bridge the performance gap between the processor and the memory, keeping frequently accessed data at lower levels. However, deploying larger DNN models has to face a capacity challenge, where the limited on-device physical memory constrains the efficient processing of large models [43].

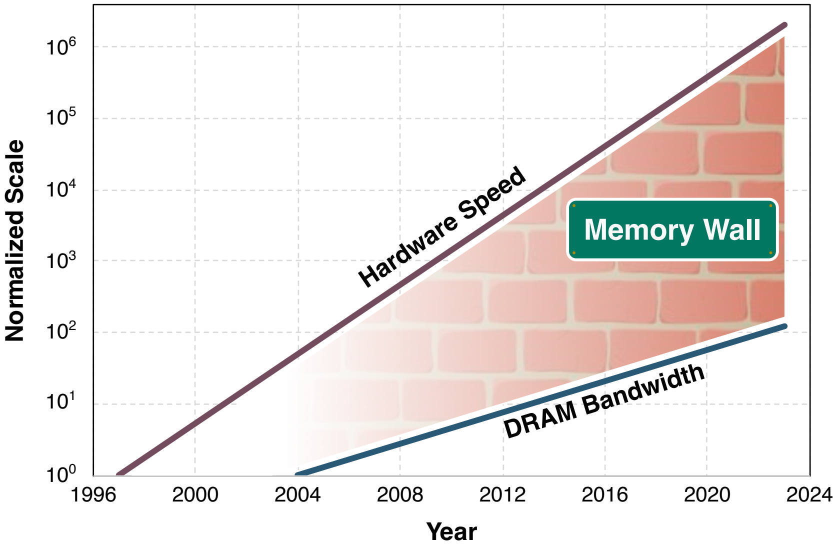


Figure 2.6: Progression of normalized processing speed and DRAM bandwidth, outlining the difference between processing and memory devices [42].

The problem posed by the memory wall presents an opportunity for hardware designers to explore solutions surrounding the way memory is utilized. Improving the use of memory access is a synonym of increasing performance in data-intense applications. The design of computing architectures is essential, as data parallel architectures have achieved high-scalable and cost-effective solutions for HPC applications [44].

2.4 VON NEUMANN ARCHITECTURE

Computing architectures are traditionally based on the concept presented by John von Neumann, referred to as von Neumann architecture [45]. The computer architecture distinguishes the CPU and the memory unit as two independent entities communicating through an interconnect bus, as shown in Figure 2.7. The CPU includes the control unit and the arithmetic logic unit (ALU), responsible for arithmetic operations such as addition, subtraction, multiplication, division or combinational logic functions. The CPU executes instructions to process data that is stored in the memory unit, and interacts with the input/output interface to receive/send data from peripheral devices. The von Neumann computer architecture presents data and instructions in the same memory unit, in comparison to the Harvard computer architecture, where the key difference is that instructions and data have independent storage entities.

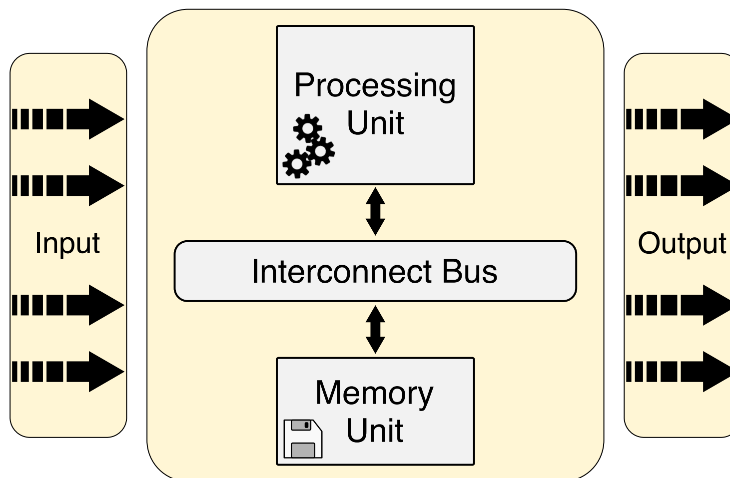
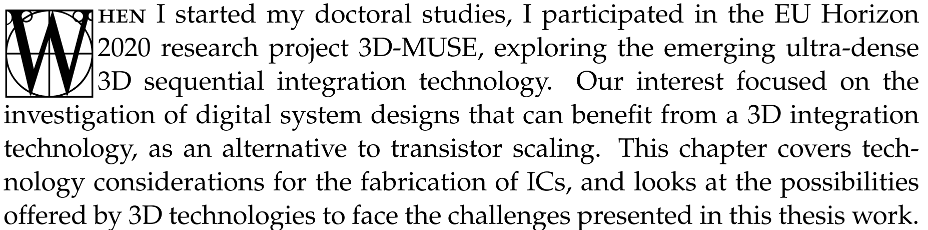


Figure 2.7: Traditional concept of computer architecture represented as the von Neumann architecture.

The memory wall issue is latent in such architectures, where data movement is conditioned to the memory bandwidth. The architecture introduced by von Neumann includes several challenges regarding resource constraints and flexibility that limit the processing capabilities of nowadays applications. The connection link between the CPU and the memory unit limits the capacity to handle the computational requirements of intelligent systems, defining the von Neumann bottleneck [46].

The research of alternatives to von Neumann architecture is comprehended around the beyond-von Neumann solutions, focusing on the exploration of hardware solutions to mitigate the memory wall [47]. The conventional von Neumann architecture has a simple sequential programming model, while beyond-von Neumann looks into more complex models to combine processing and storage devices to improve overall computing.

3 Integration Technologies



When I started my doctoral studies, I participated in the EU Horizon 2020 research project 3D-MUSE, exploring the emerging ultra-dense 3D sequential integration technology. Our interest focused on the investigation of digital system designs that can benefit from a 3D integration technology, as an alternative to transistor scaling. This chapter covers technology considerations for the fabrication of ICs, and looks at the possibilities offered by 3D technologies to face the challenges presented in this thesis work.

3.1 CMOS

Complementary metal-oxide-semiconductor (CMOS) is the conventional technology used in the fabrication of ICs, where the fundamental element is the transistor. CMOS considers two types of metal-oxide-semiconductor field-effect transistor (MOSFET), n-channel (NMOS) and p-channel (PMOS), used to create the logic in digital circuit designs. The fabrication of digital circuits using CMOS technology covers two main phases illustrated in Figure 3.1, front end of line (FEOL) and back end of line (BEOL). FEOL is the process by which transistors and other active devices are patterned according to the design defined for the IC. The next step is BEOL, in which the metals are deposited, creating the connection wires between active devices of FEOL, completing the IC fabrication.

Process variations are a critical aspect in the fabrication of CMOS technology ICs [48]. Deviations in the semiconductor manufacturing process cause the properties of the fabricated devices to differ from those expected. The yield, as the ratio of chips that work correctly according to their specifications compared to the total number of chips manufactured, fluctuates as the dimen-

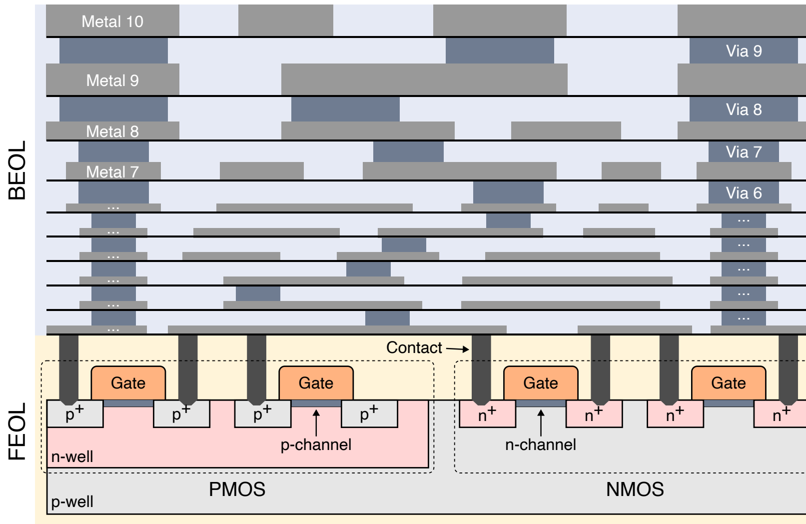


Figure 3.1: FEOL and BEOL processes in IC fabrication including transistors, metals, vias and contacts.

sions of technology get smaller. However, advancements in semiconductor manufacturing have improved the management of variations to ensure that ICs work under the conditions that are expected.

The cost of CMOS fabrication has been kept low by scaling devices according to new generations of technology. Reducing the size of MOSFETs allows for an increase in both the switching speed and the number of transistors per chip, improving the performance of large scale ICs [49]. However, smaller technology nodes contribute to a higher leakage current that consumes power when the circuit is not active, an effect of the physical properties of transistors. Performance is integrally related to chip power, which encourages optimizations at the system level to reduce energy avoiding the reduction of system performance [50].

The higher transistor density of smaller technology nodes comes with increased congestion, producing longer interconnection wires with inherent parasitics degrading the performance of a system. Continuous downsizing of technology nodes improves transistor operation delay at the cost of rapidly increasing wire delay, affecting power consumption in high performance chips [51]. The delay of a wire increases quadratically as a function of the length of the interconnections [52], which motivates design approaches to reduce the distance between active devices. 3D technologies present alternatives to conventional CMOS, where both high integration density and reduced wire interconnection lengths can be achieved.

3.2 3D STACKING

An IC is built upon a silicon die, or can be partitioned into individual circuits, each on an independent silicon die, and integrated together to create a complete system. As an alternative to scaling following Moore's law, different technologies evaluate stacking circuits, or functional parts of a circuit, to generate 3D ICs. In comparison to conventional CMOS, where transistors and active devices are in the same layer, the third dimension is provided by stacking multiple active layers with vertical interconnections. Two main manufacturing approaches are considered in the investigation of 3D ICs according to their integration strategy: (1) stacking and bonding of multiple dies and (2) monolithic integration [53].

Instead of using vertical interconnections, a 2.5D integration method is obtained by partitioning a system into silicon dies and integrating them together with a silicon interposer [54]. Figure 3.2 shows 2.5D and 3D integration technologies, where the properties of each strategy and the pitch size of the interconnections are illustrated. A silicon interposer connects dies side-by-side, while 3D bonding allows communication between stacked dies, and monolithic 3D sequential integration (3DSI) uses vertical interconnections within dies.

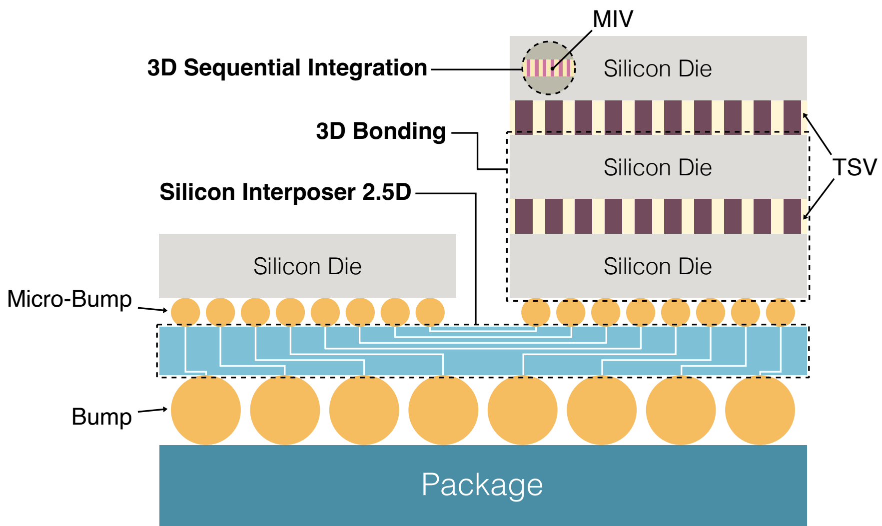


Figure 3.2: 3D integration technologies accommodated in the same package to illustrate a comparison of their integration strategies.

3.2.1 SILICON INTERPOSER

A silicon interposer is used as a common medium for connecting units in an IC package, offering better integration than passive silicon for circuit interconnections. The advanced packaging technique connects multiple circuit dies in a single package, creating a 2.5D integration. The use of a silicon interposer offers high interconnect density through silicon fabrication that provides increased die connectivity to achieve high speed and throughput.

A technique to improve the limitation of the memory wall is based on the use of a silicon interposer to connect processing units and memory with high density chip-to-chip interconnection [55]. Bringing more memory capacity closer to processing units reduces the memory latency and increase the bandwidth. High-bandwidth memory (HBM) is a popular implementation using a silicon interposer for improved external memory access with a shorter connection between processor and memory dies [56].

3.2.2 3D BONDING

Two types of interconnection are considered for bonding stacked dies in advanced semiconductor packaging. Through-silicon via (TSV), where 3D contacts are vertical interconnections that pass through a silicon wafer connecting individual dies [57], and copper-to-copper (Cu-Cu) bonding, where interconnections are created by joining copper surfaces in a strong and conductive path [58]. TSVs are used for connecting dies to each other, but also provide silicon interposer connectivity. An accurate manufacturing process for TSV has been a priority to reduce interconnection failures and achieve an appropriate level of maturity [59].

A comparison between TSV and Cu-Cu bonding used to interconnect stacked dies, and monolithic inter-tier via (MIV) used in 3DSI, is shown in Table 3.1. The pitch offered by MIV is up to 50 times smaller than the TSV or Cu-Cu contacts, allowing for a finer grained integration than die bonding [60].

Table 3.1: Comparison between different 3D integration technologies [60].

3D Contact	**TSV [57]**	**Cu-Cu [58]**	**MIV [61]**
Diameter (µm)	3	1.7 - 3	0.1 - 0.25
Pitch (µm)	10	3.4 - 10	0.2 - 0.5 *
Density (per mm^2)	1×10^4	1×10^4	$\sim 25 \times 10^6$

* Assumed to double the MIV diameter

3.3 3D SEQUENTIAL INTEGRATION

3DSI, also known as monolithic 3D (M3D), offers a higher density of interconnections achieved by the fabrication of 3D ICs in a single wafer, including vertical contacts [61]. More-Moore was the initial driver for 3DSI to continue the development of Moore's law for denser integration in high performance applications [62]. However, at the same time, the integration of independent active layers allows the use of different technology nodes. Digital, analog, and mixed-signal circuits can be integrated in the same 3D IC using technology nodes that are more suitable for each functionality, appearing as a promising breakthrough in more-than-Moore technologies.

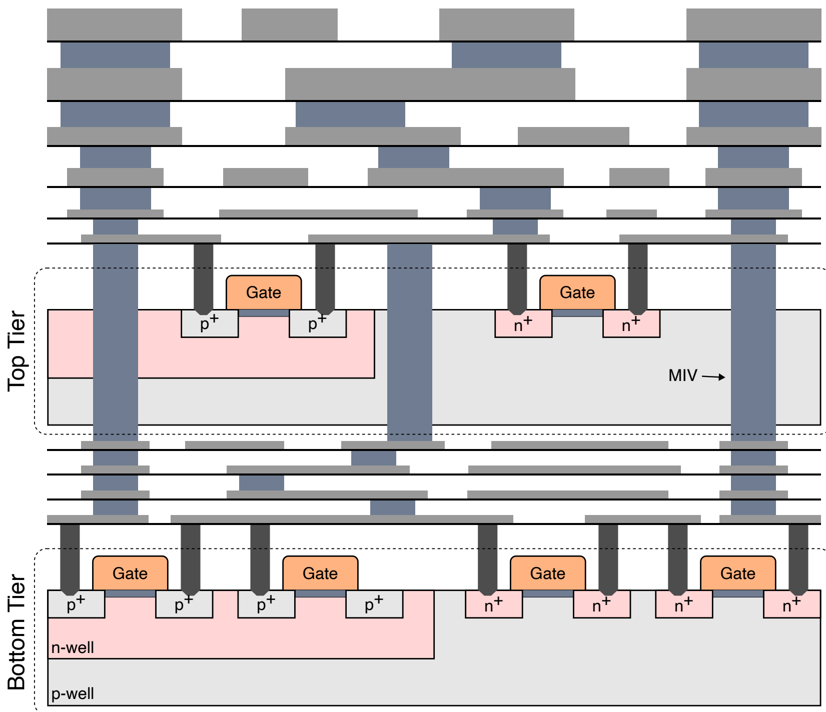


Figure 3.3: 3DSI with transistors distributed between top and bottom tiers using MIV interconnections.

Sequential integration of transistors in active layers, referred to as tiers, enables 3D integration by stacking tiers that use MIV for interconnections between active layers, as shown in Figure 3.3. The manufacturing process starts with the FEOL for the active layer of the bottom tier and then continues with the BEOL of the bottom tier metal layers. After that, the process is

repeated with the top tier, completing the 3D design with the metals that are used for the connections to other parts of the circuit. MIV is the 3D contact that is used to establish connections between top and bottom tiers, by interconnecting the highest metal layer of the bottom tier with the lowest metal layer of the top tier.

The monolithic stacking of active layers in the same die causes the activity of the bottom tier transistors to influence the active layer of the top tier. A ground plane (GP) can be introduced between both tiers, as an inter-tier isolation plane, to reduce the sensitivity of the top tier to the influence of voltage variations in the bottom tier [63]. The integration of an isolation GP implies the requirement of additional design space rules to include MIVs for vertical interconnections, which limits the integration density. However, [63] has demonstrated that for digital designs, there is a minor coupling effect between tiers, and the inter-tier GP can be avoided. Furthermore, results in [64] have shown that the coupling between tiers is high enough to influence analog applications, but they are within the boundaries of local variability for digital circuits.

The higher integration properties of 3DSI come at the cost of thermal budget constraints for the top tier [62]. During the fabrication process, the temperatures of top tier transistors need to be high enough to ensure functionality, but not too high that harm bottom tier transistors. The MOSFET in the top tier are processed at low temperature ($\approx 500\,°C$) to preserve the integrity of the bottom tier FEOL and BEOL [65]. However, the use of low temperature transistors reduces the performance of the system and expose reliability problems that can limit functionality.

The higher interconnection density offered by 3DSI allows to increase the performance and memory bandwidth with shorter connections to improve memory wall limitations. Memory and computing logic can be placed on different tiers and use MIV for shorter and denser interconnections, presenting 3D integration as a promising alternative [66]. The design of a two-tier cell structure SRAM is proposed in [67], with access capability from both active layers that maintains memory density and enhances computing performance by multiple data access. An SRAM design technique that combines boolean functions and memory cells using 3DSI is presented in [68], where multiplication and summation are performed within memory, narrowing the boundaries of computation and storage.

The high density properties of 3DSI enable different types of custom circuit partitioning strategies in tiers, offering promising prospects for 3D integration technology as an alternative to conventional CMOS scaling.

3.3.1 INTEGRATION GRANULARITY

The design approaches for digital circuits using 3DSI technology can be classified on a granularity scale according to their partitioning strategies, as shown in Figure 3.4. The higher granularity on the scale is related to the use of vertical interconnections between smaller units, which at the same time represents a higher density of MIVs.

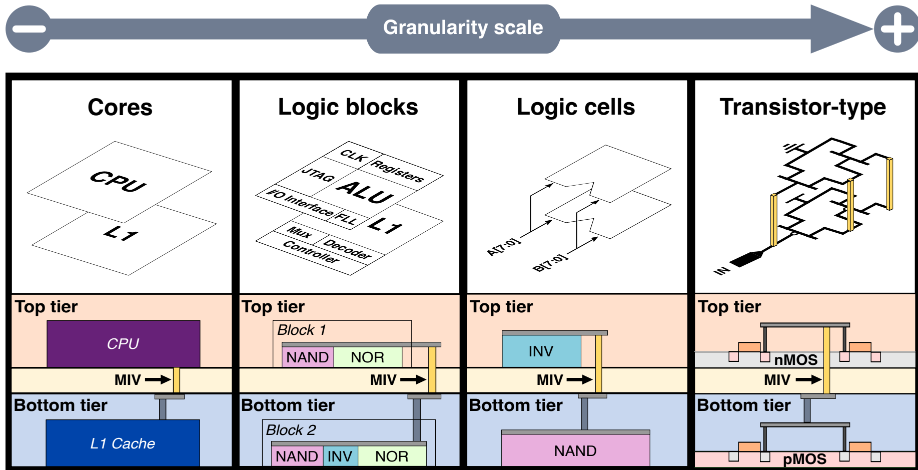


Figure 3.4: Granularity scale for different partitioning styles with 3DSI.

The various granularities are defined according to the following spectrum:

- **Cores**: At the lowest granularity level, units of a computing platform are placed independently, i.e., core and memory are allocated in different tiers, and data communication is fulfilled with MIVs.
- **Logic blocks**: Modules that define the main units of a computing platform, i.e., ALU, registers, or cache memories, are distributed between tiers offering higher placement flexibility to reduce interconnection length with 3D contacts.
- **Logic cells**: Combinational and sequential logic cells are the fundamental building components of a logic block. In [69], a platform is proposed for the design of high quality 3DSI circuits based on the use of conventional logic cells distributed in both tiers.
- **Transistor-type**: At the highest end of the scale with the finest granularity, NMOS and PMOS are separated, generating pull-up and pull-down networks of a logic cell in the top and bottom tiers, respectively [70]. This partition of tiers enables individual optimization according to transistor-type for customized logic cells.

3.3.2 CHALLENGES

Despite advancements over conventional CMOS, the use of 3DSI for the fabrication of ICs poses challenges for the semiconductor industry to become a regular 3D integration technology.

The partitioning of digital designs into individual active layers for 3D integration requires EDA tools with the ability to perform such partitions. Several commercial EDA tools support the design of IC using 3D methodologies, including Cadence Integrity 3D-IC, Synopsys 3DIC Compiler and Siemens Calibre [71]. However, these tools focus mainly on TSV and 3D contacts with a lower interconnection density. Commercial EDA tools based on conventional CMOS technology do not consider multiple tiers for 3DSI design, generating engineering overhead in order to efficiently synthesize and route digital designs in the third dimension. SoA publications in the literature present solutions that focus on the proposal of strategies that, using commercial EDA tools, generate high quality ICs designed with 3DSI technology [69,72].

Furthermore, an increased density affects the temperature of ICs, especially in active layers away from cooling systems, which has become more pertinent in modern microelectronics. The temperature characteristics of 3DSI are first addressed in [73], identifying the most relevant factors and modeling the thermal behavior of such circuits. The study introduced in [74] presents a temperature management analysis that reports thermal performance similar to TSV-based integrations due to the tight thermal coupling of the stacked tiers.

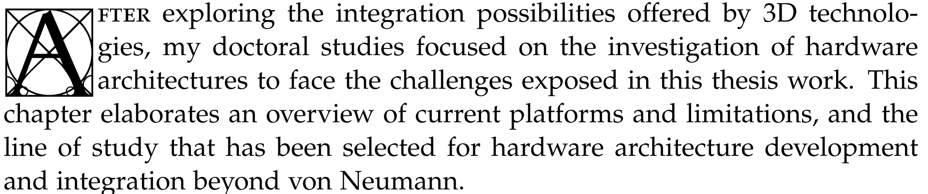
The maturity of 3DSI technology needs to achieve a high yield and ensure that functionality is adequate to become a regular integration technology. The challenges depicted by high density integration and different technology solutions need to overcome manufacturing processes obstacles.

3.4 THESIS CONTRIBUTION

As part of this thesis, another level in the granularity scale between logic cells and transistor-type is evaluated in **Paper I**, defining an intra-cell partitioning. The higher integration density offered by 3DSI allows to generate logic cells that are designed according to 3D structure premises. The proposed intra-cell integration strategy is evaluated by designing a library of sequential and combinational logic cells defined in the 3D domain, with input/output ports on a single tier, offering an interconnection solution that is compatible with commercial EDA digital design tools. Our adopted partitioning strategy is based on dividing the structure of logic cells between both tiers using MIVs efficiently to obtain high density cells that reduce interconnection lengths

in large designs. The library of logic cells includes the implementation of commonly used cells in the design of digital circuits, offering a reduced area solution for designs with high integration density. Implementations using the proposed library demonstrate to reduce the area of evaluated benchmark designs and, more relevant, reduce the length of interconnections as the main contributor to energy consumption, especially in smaller technology nodes.

4 Hardware Architecture



After exploring the integration possibilities offered by 3D technologies, my doctoral studies focused on the investigation of hardware architectures to face the challenges exposed in this thesis work. This chapter elaborates an overview of current platforms and limitations, and the line of study that has been selected for hardware architecture development and integration beyond von Neumann.

4.1 PROCESSING PLATFORMS

Edge AI envisions bringing certain cloud computing utilities to edge devices for applications that require a fast response time [75]. The combination of AI applications with edge devices offers intelligent information processed directly at the data source, addressing latency, security, and bandwidth limitations for data transmission to the cloud [76]. However, edge devices are battery powered systems that require low power design solutions to ensure efficient energy utilization.

The hardware architecture of a processing platform determines the ability of a system to perform different applications. Edge devices are designed on the basis of standalone or customized processing platforms with different technical specifications, which can be classified for performance and flexibility according to their capabilities. Three main categories are considered, general-purpose processor (GPP), field-programmable gate array (FPGA), and application-specific integrated circuit (ASIC), as shown in Figure 4.1.

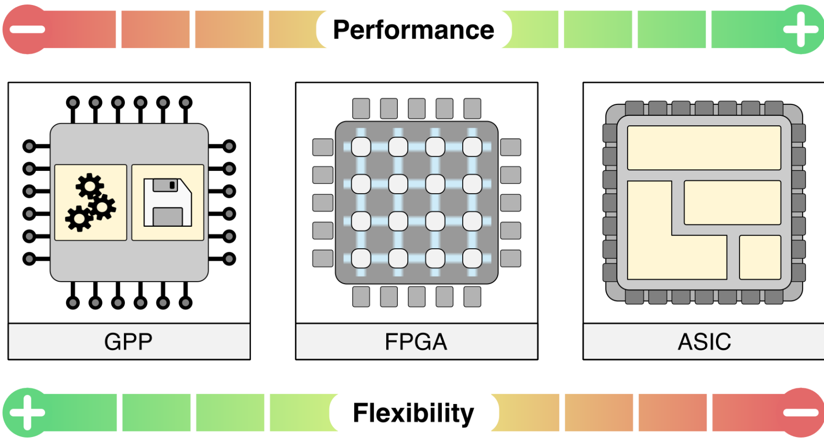


Figure 4.1: Performance and flexibility trade-off between different processing platforms.

4.1.1 GENERAL-PURPOSE

GPP is a highly resource-constrained platform that is designed to cover a wide range of instructions and arithmetic operations that are not tailored to any particular application. The high flexibility that GPPs can achieve comes at the cost of reduced performance, which limits the efficient processing of compute-intensive applications.

RISC-V is an open-source instruction set architecture (ISA) based on reduced instruction set computer (RISC) principles, supporting computer architecture designs and hardware implementations [77]. A CPU can be defined with a RISC-V core that executes instructions performing arithmetic operations and interacting with memory units.

A microcontroller unit (MCU) is a popular GPP platform, where core, memory, and peripheral units are connected to create a complete system capable of performing multiple types of operation based on instructions. Conventional MCUs are based on the von Neumann architecture design, where CPU and memory units are separated and communicate via an interconnection bus.

The graphics processing unit (GPU) is a more specialized circuit for digital image processing. GPUs are used to accelerate computer graphics using a hardware architecture designed specifically for most common operations found in image processing applications, improving performance, and reducing energy consumption. However, GPUs can be used for general-purpose processing by executing workloads that conventionally run on CPUs.

4.1.2 FPGA

An FPGA is a configurable integrated circuit that can be reprogrammed after manufacturing using configurable logic blocks and switch modules, as shown in Figure 4.2a. FPGAs represent another step in performance improvement with customized hardware generation on the platform [78]. The flexibility of FPGAs depends on the specifications of the hardware architecture generated according to the variety of defined applications to evaluate. Applications of a different nature require reprogramming on the platform, so flexibility is limited by the constraints defined by a design. FPGA has proven to be a suitable solution for compute-intensive workloads by customizing the hardware architecture of on-chip logic blocks with a dedicated pipeline to enhance performance [79]. The FPGA solution presented in [80] provides better results than optimized software running a binarized NN on a GPU.

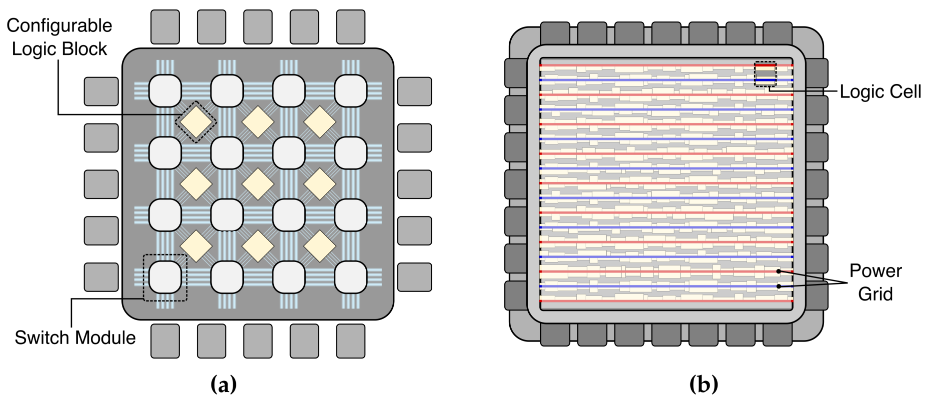


Figure 4.2: Low-level hardware architecture of (a) an FPGA, and (b) an ASIC.

4.1.3 ASIC

An ASIC can achieve a higher degree of performance at the expense of higher specialization with the use of standard logic cells, as shown in Figure 4.2b. ASICs are hardware architectures designed for specific applications with fixed functionality after fabrication. However, the higher customization properties achievable with ASICs offer higher performance and energy efficiency. ASIC requires a longer development time involving design, fabrication, and testing, while FPGA can be reprogrammed as needed, avoiding manufacturing time. To overcome the limited flexibility of ASIC, an application-specific integrated processor (ASIP) combines a general-purpose processor with dedicated hardware units to accelerate application-specific recurrent operations [81].

4.2 HARDWARE ACCELERATORS

The design of hardware accelerators on FPGA or ASIC is a common approach to incorporate AI models on resource-constrained edge devices, outperforming GPUs with specialized circuits for digital processing [82]. The computing requirements of AI applications have increased the popularity of accelerators that reroute data-intensive workloads to specialized architectures designed for improving performance and energy efficiency. While general-purpose units perform basic arithmetic operations, hardware accelerators execute resource-intensive workloads in parallel. In [83], an analysis of accelerators for AI systems is presented, evaluating different ML architectures, their implementation, and the required performance, area, and energy efficiency trade-offs.

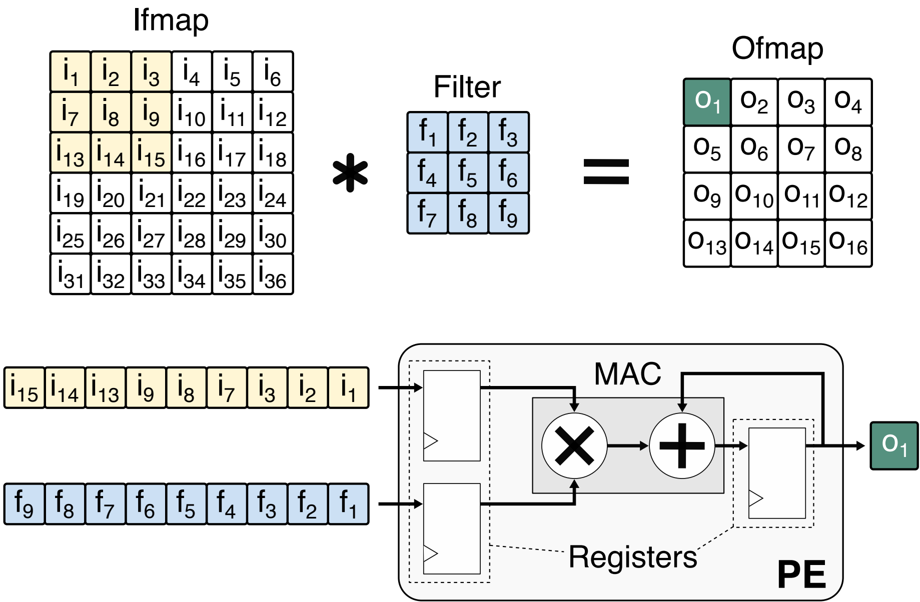


Figure 4.3: Convolutional process for a hardware accelerator architecture.

CNNs, as common ML applications, are popular applications considered for hardware acceleration, since convolutional operations and matrix multiplication are linear operations that can be optimized with specialized hardware units. The general convolutional process including the mathematical operations to calculate an ofmap parameter is shown in Figure 4.3. Processing elements (PEs) are used as the arithmetic block for calculations, using multiply-accumulate (MAC) units and registers that are commonly used as intermediate storage units for input, filters, and partial accumulation

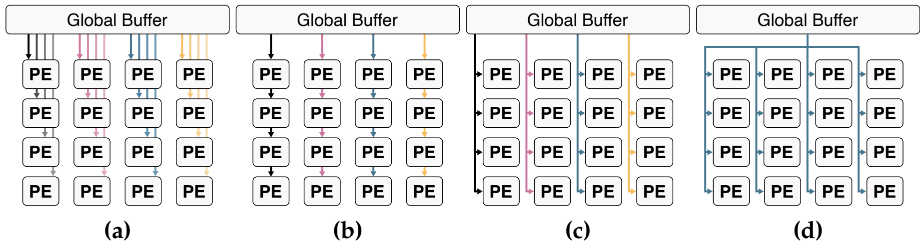


Figure 4.4: Data distribution for different hardware accelerators designs, (a) unicast, (b) systolic array, (c) multicast, and (d) broadcast.

results. The integration of multiple PEs enables extended parallelization of convolutional operations in accelerator architectures.

Figure 4.4 shows different data organization mechanisms for hardware accelerators [84]. The distribution of data across PEs is defined by the bandwidth and the level of parallelization of the network architecture. In a unicast network, the global buffer used as intermediate storage gathers unique data for each PE, requiring the highest bandwidth to access multiple parallel data, while the systolic array network moves data between PEs. Lower bandwidth alternatives are presented for multicast and broadcast networks with higher data reuse, where multiple PEs receive the same data. Different types of data distribution strategies can be employed for ifmap, filters, or ofmaps in a hardware architecture, combining bandwidth and data reuse properties to optimize computation.

4.2.1 CO-INTEGRATION

Hardware accelerators can be integrated into processing platforms and configured to carry out the compute-intensive workloads. The approach followed for the integration of hardware accelerators plays an important role in the efficiency of the proposed solutions. Dedicated memory integrated for hardware acceleration operations ensures data control and storage availability. However, private memory units come at the price of reduced integration density.

An architecture that has emerged as a promising approach to improve yield and enhance edge devices capabilities for hardware acceleration is the chiplet. It is conceived as an independent unit for parallel, distributed, and modular processing. Multi-chiplet architectures are implemented to create larger computing platforms, reducing the yield limitations as an alternative to single-chip designs with more computing capabilities. Different challenges come with a multi-chiplet integration design, since partition of workloads into a modular design requires dedicated strategies to optimize computing.

4.3 DATA-CENTRIC COMPUTING

An emerging paradigm focused on the importance of systematic data management is data-centric AI [85]. The efficiency of systems dedicated to AI applications is achieved by improving the quality and refinement of data, improving ML models and platforms. Specialized and tailored models for specific applications are difficult to transfer from one problem to another, which also affects processing platforms, motivating the development of solutions that work in different paradigms [86].

The importance of efficient data processing is highlighted by the increased amount of data generated and applications with data-centric requirements. Conventional computing systems require to move data from different levels of memory for computation on processing units. Figure 4.5 shows the energy cost of accessing data from different levels of the memory hierarchy, relative to the energy dedicated to a MAC operation [87]. Certainly, the energy dedicated to data movement is a dominant factor in the total energy cost of different hardware platforms. The high cost associated with off-chip main memory access encourages data-centric computing and control techniques to ensure the design of energy efficient processing systems.

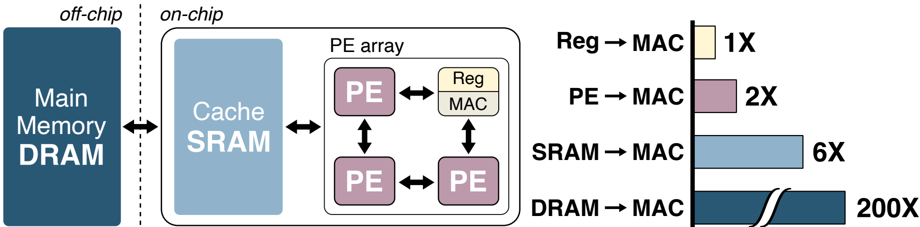


Figure 4.5: Energy cost of data movement relative to a MAC operation for a SoA accelerator architecture [87].

Efficient data processing on resource-constrained edge devices requires to optimize data movement and perform computations in or near the location where the data resides [88]. In contrast to general-purpose processing platforms, data-aware architectures include mechanisms that improve data movement with memory management and control optimizations. Addressing memory wall limitations with data-centric computing approaches requires a paradigm in which memory systems integrate computing capabilities [89]. There are two main trends centered on the idea of bringing computation closer to memory: in-memory computing (IMC) and near-memory computing (NMC), which present different techniques and integration strategies illustrated in Figure 4.6.

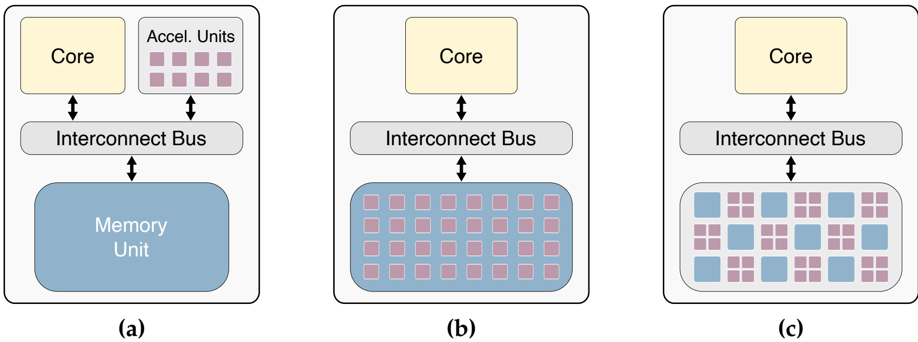


Figure 4.6: Processing architectures including (a) hardware accelerator units, and memory-computation integration strategies: (b) IMC, and (c) NMC.

4.3.1 IN-MEMORY COMPUTING

IMC is a technique based on the physical integration of memory and computing logic, involving structural modifications or additional circuitry to support computation in storage units [90]. Two integration approaches are commonly used, classified as analog IMC (AIMC) and digital IMC (DIMC). AIMC performs computation in the analog domain, offering high energy efficiency and parallelization, which compromises the accuracy of results due to the effect of intrinsic circuit noise and mismatches [91]. AIMC implies modifying part of the memory architecture, peripheral or cell, where analog voltage levels define a value and require an analog-to-digital converter (ADC) in the peripheral circuit. In contrast, DIMC are implemented with digital logic circuitry defining multiplication and addition operations integrated within memory cells.

IMC techniques are available for implementation on different memory levels, i.e., off-chip DRAM have integrated computing logic producing IMC off-chip, and NVM architectures have high storage density and can efficiently perform parallel matrix multiplications for hardware acceleration. Resistive Random-Access Memory (RRAM) has high density, low power, and multilevel operations to perform operations in the AIMC domain with a high parallel memory array [92]. The IMC on-chip is achieved by modifying the low-level memory structure of SRAMs, limiting the scalability and integration of highly optimized memory cells.

Domain-specific hardware accelerators based on the IMC integration approaches present high performance results and low power consumption. However, intrinsic modifications of memory architectures limit flexibility and the ability to process different types of applications.

4.3.2 NEAR-MEMORY COMPUTING

As an alternative to modifying the low-level structure of memory devices, NMC techniques place computation logic near storage units, avoiding intrusive integration that modifies the optimized hardware architecture of memory cells. Hardware architectures employing NMC techniques benefit from increased memory bandwidth with a partitioned memory sub-system to face the limitations of the memory wall.

Resource-constrained edge devices can expand their potential with architectures that enhance the computing capabilities of the platform with tailored processing units. NMC techniques allow the integration of hardware accelerators as co-processors with shared memory resources, with optimized functionality to improve the performance and energy efficiency for compute-intensive AI applications. Integration of NMC architectures on hardware platforms requires a custom memory interface to synchronize accesses from conventional processors and accelerators with independent data controls. While general-purpose cores use standard load and store memory operations, hardware accelerators defined with NMC techniques can parallelize access to memory, optimizing the bandwidth for extended data processing.

Cache memories built with SRAM are a primary component of CPUs that contribute greatly to the efficient utilization of memory resources. Many works have recently explored the opportunities offered by NMC techniques, with active integration of computing units near SRAM in the CPU cache hierarchy [90]. An NMC work that shares the last level cache between parallel compute units accessing internal SRAM sub-arrays and an eight-core RISC-V processor is presented in [93]. The high memory bandwidth of an array of shared SRAMs that form the cache memory is enhanced with an NMC architecture that bypasses the bandwidth bottleneck processing where data is stored. Using RISC-V cores combined with compute units customized for MAC operations achieves the programmability of general-purpose processors together with high performance gains. Alternatively, the placement of hardware accelerators next to on-chip memory is evaluated in [94], showing performance and energy improvements compared to coupling them and moving data through an on-chip bus. Furthermore, the versatility of flexible memory access for an NMC technique is explored in [95] with two architecture approaches that target energy efficiency on edge devices running ML algorithms.

Although IMC techniques offer a higher level of parallelization with the computation located in memory, NMC avoids low-level SRAM alterations, increasing processing flexibility with customized PEs. The use of NMC techniques transforms computer paradigms to present beyond-von Neumann architectures that efficiently process AI applications en edge devices.

4.4 THESIS CONTRIBUTION

The design and fabrication of hardware architectures integrated on edge device platforms using NMC techniques is evaluated as part of this thesis. CNN applications are considered as benchmark for acceleration to evaluate the performance and energy efficiency improvements of parametrized hardware architectures. The first NMC architecture was presented in **Paper II**, testing the integration on an MCU platform with a shared memory subsystem, where a compatible memory control was considered and evaluated. Flexibility and scalability have been assessed during the design stages to offer hardware architectures capable of adapting to varying algorithmic demands and platform requirements, as demonstrated in **Paper III**. Furthermore, the presented solutions offer integration on cache memories with a digital NMC framework compatible with conventional SRAM architectures. Two chips, fabricated in 22 nm fully depleted silicon-on-insulator (FDSOI) technology, are presented in **Papers IV** and **V**. The results show an NMC technique that is efficiently integrated in a resource-constrained platform and an architecture for multi-chiplet implementations fabricated and tested for parallel computation of distributed workloads. The modular design enables multiple partitioning approaches with high performance per chip and a chiplet-to-chiplet communication interface. The designed and evaluated NMC techniques are compared with relevant SoA works and presented as solutions that deliver high performance, energy efficiency, and area density.

5 Conclusion and Future Work

THE content of this chapter is the result of a reflection considering the different topics that have been covered during my doctoral studies, gathering the conclusion of this thesis from a technology and hardware architecture point of view.

The challenges introduced by current and future technological applications require hardware platforms that can cope with such demands. The properties offered by edge computing, combined with AI applications, make edge AI an increasingly relevant concept that links compute-intensive workloads to resource-constrained connected systems. In order to imbue higher computing capabilities in processing platforms for edge AI applications, this thesis has evaluated 3D integration technologies and hardware architectures based on the beyond-von Neumann computing paradigm.

The exploration of 3DSI, which emerged as a continuation of the development of Moore's law for the digital design of ICs, has demonstrated the benefits of higher integration density. In this thesis work, an intra-cell partitioning strategy has been evaluated for the design of logic cells in a 3D domain that can be coupled with conventional EDA tools. However, several challenges arise from the requirements of reliable 3D technology, from temperature limitations to manufacturing maturity to achieve high yield and ensure functionality.

Alternatively, the design of hardware architectures based on an NMC technique has been evaluated to define beyond-von Neumann solutions for the requirements of compute-intensive applications. The presented NMC technique has focused on the integration of tailored PEs near SRAM macros to obtain hardware accelerators working as co-processors on an MCU platform. Flexibility and scalability are considered in the design of the proposed systems to obtain solutions that can cope with different application requirements.

The results have demonstrated that the proposed architectures offer an NMC solution that improves the performance, energy efficiency, and area density of resource-constrained systems compared to SoA designs.

In conclusion, this thesis has evaluated relevant aspects of ICs and proposed solutions to improve SoA considering challenges and limitations.

FUTURE WORK

The combination of 3D integration technologies with optimized hardware architectures can offer highly integrated chip architectures that improve the processing of current and future technological applications. The placement of computing units closer to memory can be achieved by combining NMC techniques with 3DSI, where memory and hardware acceleration units are stacked with shorter interconnections. Bringing NMC techniques with 3DSI technology for hardware acceleration needs to be tightly coupled to algorithmic models to achieve significant advances, and address system-level integration challenges.

Bibliography

[1] I. Levin and D. Mamlok, "Culture and Society in the Digital Age," *Information*, vol. 12, no. 2, p. 68, 2021.

[2] N. Thell, "Technology in Everyday Life," *Symbolic Interaction*, 2024.

[3] R. Volti and J. Croissant, *Society and Technological Change*. Waveland Press, 2024.

[4] Strategy&, "Forging Germany's Digital Destiny: The Imperative of a Sustainable Microelectronics Strategy," 2023.

[5] O. Bonnaud and L. Fesquet, "Microelectronics at the Heart of the Digital Society: Technological and Training Challenges," in *34th Symposium on Microelectronics Technology and Devices (SBMicro)*, 2019, pp. 1–4.

[6] P. Singh, "Systematic Review of Data-Centric Approaches in Artificial Intelligence and Machine Learning," *Data Science and Management*, vol. 6, no. 3, pp. 144–157, 2023.

[7] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Compute- and Data-Intensive Networks: The Key to the Metaverse," in *1st International Conference on 6G Networking (6GNet)*, 2022, pp. 1–8.

[8] A. Gandomi and M. Haider, "Beyond the Hype: Big Data Concepts, Methods, and Analytics," *International Journal of Information Management*, vol. 35, no. 2, pp. 137–144, 2015.

[9] J. Wang, C. Xu, J. Zhang, and R. Zhong, "Big Data Analytics for Intelligent Manufacturing Systems: A Review," *Journal of Manufacturing Systems*, vol. 62, pp. 738–752, 2022.

[10] J. Pan and J. McElhannon, "Future Edge Cloud and Edge Computing for Internet of Things Applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2018.

[11] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[12] C. Hao, J. Dotzel, J. Xiong, L. Benini, Z. Zhang, and D. Chen, "Enabling Design Methodologies and Future Trends for Edge AI: Specialization and Codesign," *IEEE Design & Test*, vol. 38, no. 4, pp. 7–26, 2021.

[13] T. Meuser, L. Lovén, M. Bhuyan, S. G. Patil, S. Dustdar, A. Aral, S. Bayhan, C. Becker, E. d. Lara, A. Y. Ding *et al.*, "Revisiting Edge AI: Opportunities and Challenges," *IEEE Internet Computing*, vol. 28, no. 4, pp. 49–59, 2024.

[14] A. Thadani and G. C. Allen, "Mapping the Semiconductor Supply Chain," *Center for Strategic and International Studies*, vol. 11, 2023.

[15] A. Ciani and M. Nardo, "The Position of the EU in the Semiconductor Value Chain: Evidence on Trade, Foreign Acquisitions, and Ownership," JRC working papers in economics and finance, Tech. Rep. 2022/3, 2022.

[16] C. Zhang and Y. Lu, "Study on Artificial Intelligence: The State of the Art and Future Prospects," *Journal of Industrial Information Integration*, vol. 23, p. 100224, 2021.

[17] H. Hua, Y. Li, T. Wang, N. Dong, W. Li, and J. Cao, "Edge Computing with Artificial Intelligence: A Machine Learning Perspective," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[18] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, 2020.

[19] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A Survey of Recent Advances in Edge-Computing-Powered Artificial Intelligence of Things," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 849–13 875, 2021.

[20] D. Reinsel, J. Gantz, and J. Rydning, "The Digitization of the World: From Edge to Core," *Framingham: International Data Corporation*, vol. 16, pp. 1–28, 2018.

[21] T. Sipola, J. Alatalo, T. Kokkonen, and M. Rantonen, "Artificial Intelligence in the IoT Era: A Review of Edge AI Hardware and Software," in *31st Conference of Open Innovations Association (FRUCT)*, 2022, pp. 320–331.

[22] S. V. Chinta, Z. Wang, A. Palikhe, X. Zhang, A. Kashif, M. A. Smith, J. Liu, and W. Zhang, "AI-Driven Healthcare: Fairness in AI Healthcare: A Survey," *PLOS Digital Health*, vol. 4, no. 5, p. e0000864, 2025.

[23] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine Learning: An Artificial Intelligence Approach*. Springer Science & Business Media, 2013.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[25] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions," *Journal of big Data*, vol. 8, no. 1, p. 53, 2021.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[27] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556*, 2014.

[28] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," *arXiv:1803.01164*, 2018.

[29] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv:1704.04861*, 2017.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[31] G. E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, 1965.

[32] M. S. Lundstrom and M. A. Alam, "Moore's Law: The Journey Ahead," *Science*, vol. 378, no. 6621, pp. 722–723, 2022.

[33] K. Rupp, "Microprocessor Trend Data," 2022.

[34] T. N. Theis and H.-S. P. Wong, "The End of Moore's Law: A New Beginning for Information Technology," *Computing in Science & Engineering*, vol. 19, no. 2, pp. 41–50, 2017.

[35] W. Arden, M. Brillouët, P. Cogez, M. Graef, B. Huizing, and R. Mahnkopf, "More-than-Moore White Paper," *Version 2*, 2010.

[36] F. Iacopi and F. Balestra, *More-than-Moore Devices and Integration for Semiconductors*. Springer, 2023.

[37] C. Hartfield, W. Harris, A. Gu, M. Terada, V. Viswanathan, L. Jiao, and T. Rodgers, "Emerging Technologies for Advanced 3D Package Characterization to Enable the More-than-Moore Era," *ECS Transactions*, vol. 109, no. 2, p. 15, 2022.

[38] R. Nair, "Evolution of Memory Architecture," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1331–1345, 2015.

[39] A. Hazarika, S. Poddar, and H. Rahaman, "Survey on Memory Management Techniques in Heterogeneous Computing Systems," *IET Computers & Digital Techniques*, vol. 14, no. 2, pp. 47–60, 2020.

[40] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," in *5th IEEE International Memory Workshop*, 2013, pp. 21–25.

[41] S. A. Przybylski, *Cache and Memory Hierarchy Design: A Performance Directed Approach*. Morgan Kaufmann, 1990.

[42] A. Gholami, Z. Yao, S. Kim, C. Hooper, M. W. Mahoney, and K. Keutzer, "AI and Memory Wall," *IEEE Micro*, vol. 44, no. 3, pp. 33–39, 2024.

[43] Y. Kwon and M. Rhu, "Beyond the Memory Wall: A Case for Memory-Centric HPC System for Deep Learning," in *51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2018, pp. 148–161.

[44] B. Jang, D. Schaa, P. Mistry, and D. Kaeli, "Exploiting Memory Access Patterns to Improve Memory Performance in Data-Parallel Architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 1, pp. 105–118, 2011.

[45] R. Eigenmann and D. J. Lilja, "Von Neumann Computers," *Wiley Encyclopedia of Electrical and Electronics Engineering*, vol. 23, pp. 387–400, 1998.

[46] J. Backus, "Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs," *Communications of the ACM*, vol. 21, no. 8, pp. 613–641, 1978.

[47] D. Kimovski, N. Saurabh, M. Jansen, A. Aral, A. Al-Dulaimy, A. B. Bondi, A. Galletta, A. V. Papadopoulos, A. Iosup, and R. Prodan, "Beyond von Neumann in the Computing Continuum: Architectures, Applications, and Future Directions," *IEEE Internet Computing*, vol. 28, no. 3, pp. 6–16, 2023.

[48] K. J. Kuhn, M. D. Giles, D. Becher, P. Kolar, A. Kornfeld, R. Kotlyar, S. T. Ma, A. Maheshwari, and S. Mudanai, "Process Technology Variation," *IEEE Transactions on Electron Devices*, vol. 58, no. 8, pp. 2197–2208, 2011.

[49] H. Iwai, "CMOS Technology-Year 2010 and Beyond," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 3, pp. 357–366, 1999.

[50] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein, "Scaling, Power, and the Future of CMOS," in *IEEE International Electron Devices Meeting (IEDM)*, 2005, pp. 7–15.

[51] K. Banerjee, S. Souri, P. Kapur, and K. Saraswat, "3-D ICs: A Novel Chip Design for Improving Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 602–633, 2001.

[52] R. Ho, K. Mai, and M. Horowitz, "The Future of Wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490–504, 2001.

[53] S. S. Salvi and A. Jain, "A Review of Recent Research on Heat Transfer in Three-Dimensional Integrated Circuits (3-D ICs)," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 11, no. 5, pp. 802–821, 2021.

[54] F. Sheikh, R. Nagisetty, T. Karnik, and D. Kehlet, "2.5D and 3D Heterogeneous Integration: Emerging Applications," *IEEE Solid-State Circuits Magazine*, vol. 13, no. 4, pp. 77–87, 2021.

[55] E. Beyne, D. Milojevic, G. Van der Plas, and G. Beyer, "3D SoC Integration, Beyond 2.5D Chiplets," in *IEEE International Electron Devices Meeting (IEDM)*, 2021, pp. 3.6.1–3.6.4.

[56] D. U. Lee, K. S. Lee, Y. Lee, K. W. Kim, J. H. Kang, J. Lee, and J. H. Chun, "Design Considerations of HBM Stacked DRAM and the Memory Architecture Extension," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2015, pp. 1–8.

[57] H. Chaabouni, M. Rousseau, P. Leduc, A. Farcy, R. El Farhane, A. Thuaire, G. Haury, A. Valentian, G. Billiot, M. Assous *et al.*, "Investigation on TSV Impact on 65nm CMOS Devices and Circuits," in *International Electron Devices Meeting*, 2010, pp. 35.1.1–35.1.4.

[58] R. Taibi, L. Di Cioccio, C. Chappaz, L.-L. Chapelon, P. Gueguen, J. Dechamp, R. Fortunier, and L. Clavelier, "Full Characterization of Cu/Cu Direct Bonding for 3D Integration," in *Proceedings 60th Electronic Components and Technology Conference (ECTC)*, 2010, pp. 219–225.

[59] J. Wang, F. Duan, Z. Lv, S. Chen, X. Yang, H. Chen, and J. Liu, "A Short Review of Through-Silicon Via (TSV) Interconnects: Metrology and Analysis," *Applied Sciences*, vol. 13, no. 14, p. 8301, 2023.

[60] H. Sarhan, S. Thuries, O. Billoint, and F. Clermidy, "An Unbalanced Area Ratio Study for High Performance Monolithic 3D Integrated Circuits," in *IEEE Computer Society Annual Symposium on VLSI*, 2015, pp. 350–355.

[61] P. Batude, B. Sklenard, C. Fenouillet-Beranger, B. Previtali, C. Tabone, O. Rozeau, O. Billoint, O. Turkyilmaz, H. Sarhan, S. Thuries *et al.*, "3D Sequential Integration Opportunities and Technology Optimization," in *IEEE International Interconnect Technology Conference*, 2014, pp. 373–376.

[62] P. Batude, L. Brunet, C. Fenouillet-Beranger, F. Andrieu, J.-P. Colinge, D. Lattard, E. Vianello, S. Thuries, O. Billoint, P. Vivet *et al.*, "3D Sequential Integration: Application-Driven Technological Achievements and Guidelines," in *IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 3.1.1–3.1.4.

[63] P. Sideris, J. Lugo-Alvarez, P. Batude, L. Brunet, P. Acosta-Alba, S. Kerdiles, C. Fenouillet-Beranger, G. Sicard, O. Rozeau, F. Andrieu *et al.*, "Inter-Tier Dynamic Coupling and RF Crosstalk in 3D Sequential Integration," in *IEEE International Electron Devices Meeting (IEDM)*, 2019, pp. 3.4.1–3.4.4.

[64] P. Sideris, L. Brunet, G. Sicard, P. Batude, and C. Theodorou, "Impact of Inter-Tier Coupling on Static and Noise Performance in 3D Sequential Integration Technology," in *Joint International EUROSOI Workshop and International Conference on Ultimate Integration on Silicon (EUROSOI-ULIS)*, 2019, pp. 1–4.

[65] A. Tsiara, X. Garros, L. Brunet, P. Batude, C. Fenouillet-Béranger, K. Triantopoulos, M. Cassé, M. Vinet, F. Gaillard, and G. Ghibaudo, "Performance and Reliability of a Fully Integrated 3D Sequential Technology," in *IEEE Symposium on VLSI Technology*, 2018, pp. 75–76.

[66] Y. Cheng, X. Guo, and V. F. Pavlidis, "Emerging Monolithic 3D Integration: Opportunities and Challenges from the Computer System Perspective," *Integration*, vol. 85, pp. 97–107, 2022.

[67] S. Srinivasa, X. Li, M.-F. Chang, J. Sampson, S. K. Gupta, and V. Narayanan, "Compact 3-D-SRAM Memory with Concurrent Row and Column Data Access Capability using Sequential Monolithic 3-D Integration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 4, pp. 671–683, 2018.

[68] S. Srinivasa, W.-H. Chen, Y.-N. Tu, M.-F. Chang, J. Sampson, and V. Narayanan, "Monolithic-3D Integration Augmented Design Techniques for Computing in SRAMs," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[69] S. Thuries, O. Billoint, S. Choisnet, R. Lemaire, P. Vivet, P. Batude, and D. Lattard, "M3D-ADTCO: Monolithic 3D Architecture, Design and Technology Co-Optimization for High Energy Efficient 3D IC," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 1740–1745.

[70] C. Yan and E. Salman, “Mono3D: Open Source Cell Library for Monolithic 3-D Integrated Circuits,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 3, pp. 1075–1085, 2018.

[71] L. Zhu and S. K. Lim, “Design Automation Needs for Monolithic 3D ICs: Accomplishments and Gaps,” in *60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–4.

[72] S. Bobba, A. Chakraborty, O. Thomas, P. Batude, T. Ernst, O. Faynot, D. Z. Pan, and G. De Micheli, “CELONCEL: Effective Design Technique for 3-D Monolithic Integration targeting High Performance Integrated Circuits,” in *16th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2011, pp. 336–343.

[73] S. K. Samal, S. Panth, K. Samadi, M. Saedi, Y. Du, and S. K. Lim, “Fast and Accurate Thermal Modeling and Optimization for Monolithic 3D ICs,” in *51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2014, pp. 1–6.

[74] C. Santos, P. Vivet, S. Thuries, O. Billoint, J.-P. Colonna, P. Coudrain, and L. Wang, “Thermal Performance of CoolCube™ Monolithic and TSV-based 3D Integration Processes,” in *IEEE International 3D Systems Integration Conference (3DIC)*, 2016, pp. 1–5.

[75] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, “Edge Computing: A Survey,” *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019.

[76] V. Shankar, “Edge AI: A Comprehensive Survey of Technologies, Applications, and Challenges,” in *1st International Conference on Advanced Computing and Emerging Technologies (ACET)*, 2024, pp. 1–6.

[77] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovi, “The RISC-V Instruction Set Manual. Volume 1: User-level ISA, Version 2.0,” 2014.

[78] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, and T. Zhou, “A Survey on Edge Computing Systems and Tools,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1537–1562, 2019.

[79] J. Cong, V. Sarkar, G. Reinman, and A. Bui, “Customizable Domain-Specific Computing,” *IEEE Design & Test of Computers*, vol. 28, no. 2, pp. 6–15, 2010.

[80] E. Nurvitadhi, D. Sheffield, J. Sim, A. Mishra, G. Venkatesh, and D. Marr, “Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC,” in *International Conference on Field-Programmable Technology (FPT)*, 2016, pp. 77–84.

[81] M. Capra, R. Peloso, G. Masera, M. Ruo Roch, and M. Martina, "Edge Computing: A Survey on the Hardware Requirements in the Internet of Things World," *Future Internet*, vol. 11, no. 4, p. 100, 2019.

[82] A. Samanta, I. Hatai, and A. K. Mal, "A Survey on Hardware Accelerator Design of Deep Learning for Edge Devices," *Wireless Personal Communications*, vol. 137, no. 3, pp. 1715–1760, 2024.

[83] T. Mohaidat and K. Khalil, "A Survey on Neural Network Hardware Accelerators," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 8, pp. 3801–3822, 2024.

[84] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, *Efficient Processing of Deep Neural Networks*. Springer, 2020.

[85] J. Jakubik, M. Vössing, N. Kühl, J. Walk, and G. Satzger, "Data-Centric Artificial Intelligence," *Business & Information Systems Engineering*, vol. 66, no. 4, pp. 507–515, 2024.

[86] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, Z. Jiang, S. Zhong, and X. Hu, "Data-Centric Artificial Intelligence: A Survey," *ACM Computing Surveys*, vol. 57, no. 5, pp. 1–42, 2025.

[87] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," in *ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 367–379.

[88] N. Hajinazar, "Data-Centric and Data-Aware Frameworks for Fundamentally Efficient Data Handling in Modern Computing Systems," *arXiv preprint arXiv:2109.05881*, 2021.

[89] J. Gómez-Luna, I. El Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware," in *12th International Green and Sustainable Computing Conference (IGSC)*, 2021, pp. 1–7.

[90] D. Fujiki, X. Wang, A. Subramaniyan, and R. Das, *In-/Near-Memory Computing*. Springer, 2021.

[91] J. Sun, P. Houshmand, and M. Verhelst, "Analog or Digital In-Memory Computing? Benchmarking Through Quantitative Modeling," in *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–9.

[92] D. Ielmini and G. Pedretti, "Resistive Switching Random-Access Memory (RRAM): Applications and Requirements for Memory and Computing," *Chemical Reviews*, 2025.

[93] G. K. Chen, P. C. Knag, C. Tokunaga, and R. K. Krishnamurthy, “An Eight-Core RISC-V Processor With Compute Near Last Level Cache in Intel 4 CMOS,” *IEEE Journal of Solid-State Circuits*, vol. 58, no. 4, pp. 1117–1128, 2023.

[94] Q. Si and B. C. Schafer, “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.

[95] M. Caon, C. Choné, P. D. Schiavone, A. Levisse, G. Masera, M. Martina, and D. Atienza, “Scalable and RISC-V Programmable Near-Memory Computing Architectures for Edge Nodes,” *IEEE Transactions on Emerging Topics in Computing*, vol. 13, no. 3, pp. 1003–1018, 2025.

PAPERS

Paper I

High-Density Standard Cell Library for Sequential 3D Integrated Circuits

Research efforts to push the integration density of circuits with technologies that transcend Moore's law have gained significant attention in recent years. This study investigates the silicon area gains of Sequential 3D technology, utilizing the third dimension of integrated circuits by accommodating nMOS and pMOS transistors in two stacked tiers with high-density and low-pitch 3D vias. The efficiency of the proposed integration strategy is exemplified through the design of a library with high-density 3D standard cells, including sequential and combinational logic. The integration of 3D vias within the standard cells mitigates the effort required for inter-tier connections during the routing of integrated circuits. Subsequent analysis indicated an average silicon area reduction of 36 % in comparison to commercially available libraries with purely planar cells. The proposed 3D cells have been incorporated into a commercial design flow for a 28 nm process technology and have been benchmarked using examples of large-scale integration designs, indicating an area and wirelength reduction of 44 % and 23 %, respectively.

©2024 IEEE. Reprinted, with permission, from
A. Prieto, and J. Rodrigues,
"High-Density Standard Cell Library for Sequential 3D Integrated Circuits," in *IFIP/IEEE 32nd International Conference on Very Large Scale Integration (VLSI-SoC)*, Oct. 2024, pp. 1-4.

I. INTRODUCTION

Technology scaling for integrated circuits (ICs) has slowed down considerably in recent years. However, transcending Moore's law, research on more-than-Moore technologies has gained increased attention, driven by the demand for high-performance computing, e.g., artificial intelligence, 6G networks and virtual reality. Consequently, higher computational demands require higher transistor density, and the corresponding complexity comes with significantly increased congestion and longer wire connections with inherent parasitics that impact the system performance. Moreover, core utilization will be reduced as signal routing may require a larger space between cells, and thus, the total silicon cost will increase.

A promising more-than-Moore technology that offers high integration density is Sequential 3D (S3D). This technology is realized by stacking a layer of transistors, which is fabricated in a sequential process, on top of another layer of transistors [1, 2]. In S3D, each active layer is referred to as a tier, realizing a 3D implementation by having tiers stacked above each other. Tiers are connected by 3D vias, known as Monolithic Inter-tier Vias (MIVs), which have the advantage of being the same size as conventional vias in a silicon process, and having up to 50 times smaller pitch than Through Silicon Vias (TSVs). These properties result in a significantly higher integration density, outperforming other 3D integration technologies [3].

The implementation of 3D circuits comes with challenges in tier integration and associated costs regarding stacking. On the bottom tier, standard nMOS and pMOS transistors are fabricated, including its Back-End-Of-Line (BEOL). Thereafter, 3D vias are integrated, followed by the manufacturing of the top tier. This procedure in the manufacturing process is extremely critical for achieving a sufficient yield. Therefore, the manufacturing temperature needs to be kept below 500 °C to avoid degradation of the bottom structures [4].

A performance limiter of today's circuits is the parasitics of lengthy interconnections, caused by the physical properties of ICs. S3D offers different implementation possibilities, based on various integration approaches, which reduce overall wirelength and silicon area. A cell library for S3D integrated circuits, designed with nMOS and pMOS transistors separation on top and bottom tier, respectively, is presented in [5]. The reduced pitch size offered by MIV enables the division of tiers for a specific transistor type, i.e., nMOS and pMOS, creating 3D cells with all transistors of the same type concentrated in the same tier. Alternatively, a division of logic cells in tiers, rather than transistor separation, is evaluated in [6]. This analysis exposes the required research on placement techniques, taking into account routing congestion in 3D integrated circuits for efficient connections between tiers. In [7], arithmetic logic blocks for multiplications are evaluated, making a circuit integration

with several layers stacked above each other, and employing vertical pillars for the connection of multiplier elements. These works show different possibilities with 3D technology to obtain high integration density. However, the aforementioned 3D ICs are realized by accommodating individual logic cells, or transistor types, in each tier. An alternative is the evaluation of 3D cell designs by partitioning the cell circuit across tiers.

This work investigates the integration density and area-wirelength reduction efficiency of S3D technology by evaluating the design and integration of 3D cells, where nMOS and pMOS transistors are freely accommodated on both tiers. A library of 3D standard cells is created for large-scale designs.

The remainder of this manuscript is organized as follows: Section II presents the proposed integration for the library of high-density 3D cells, and Section III discusses silicon area gains and the use of the presented cells as part of benchmarked examples. Finally, Section IV concludes this study.

II. HIGH-DENSITY 3D STANDARD CELL LIBRARY

Sequential 3D technology offers various levels of partitioning. This section will provide background information on the integration levels and details on the defined intra-cell partition for the generation of our 3D standard cell library.

II.A. SEQUENTIAL 3D INTEGRATION

3D circuits can be realized by considering different implementation styles, as shown in Fig. 1. At the lowest granularity level, an entire core is placed on one tier, and data communication is realized with vertical connections between tiers. Conventional 2D cells are used, increasing the engineering overhead for finding an optimal partition of the design that reduces routing congestion on both tiers and achieves an efficient area implementation. On the other end of the scale, reaching the finest granularity, the nMOS and pMOS transistors are strictly separated, i.e., the pull-down and pull-up networks of a logic cell are in the top and bottom tier, respectively, and MIVs realize transistor connections. By making a transistor-type partition, higher integration density can be achieved using more MIVs, at the cost of increased parasitics arising from more inter-tier connections. The technique proposed in this study closes the design gap between separating the tiers by logic cells or transistor type, seeking high integration density with efficient inter-tier connections. This is accomplished by freely integrating nMOS and pMOS on either tier, with MIVs for intra-cell connections.

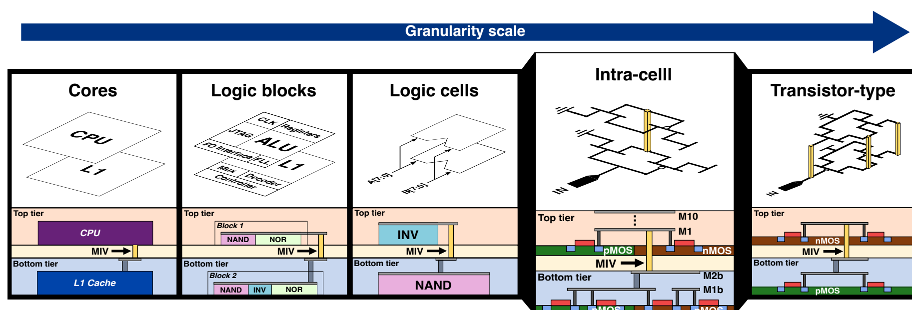


Fig. 1: Granularity scale for different partitioning possibilities with Sequential 3D integration technology, including high-density 3D cells integrated as intra-cell partition with nMOS and pMOS transistors accommodated freely on two tiers.

The stack of transistors and metal layers for intra-cell partition with Sequential 3D integration technology is shown in Fig. 1. Both top and bottom tiers include nMOS and pMOS transistors, with two metal layers (M1b and M2b) dedicated to the bottom for internal cell connections including power rails, and ten (M1-M10) to the top for routing. The connection between tiers is realized by MIVs, which have the same physical dimensions as other vias between metal layers.

With intra-cell partition approach, transistors are efficiently integrated in 3D cells with the same distance between power rails as conventional 2D cells, providing a large design space for routing tracks with reduced congestion of inter-tier connections. Consequently, the bottlenecks derived from 1) routing congestion of MIVs produced by using 2D cells on 3D stacking, and 2) reduced cell height with more MIVs that limit routing tracks on transistor-type partition, are addressed with a solution that offers high-density integration with MIVs included in cells.

In Sequential 3D integration, the stacking of transistors generates a coupling effect that makes the top tier more sensitive to voltage variations in the bottom tier. The introduction of an isolation plane between both tiers for avoiding undesirable interaction has been evaluated in [8], where a polysilicon Ground Plane (GP) is introduced sequentially in the manufacturing of 3D integrated circuits. When inserting an isolation plane, extra design space rules are required for including MIVs, which reduces the integration density. However, results in [9] demonstrate that there is minor coupling between top and bottom tiers for digital ICs, therefore the introduction of an inter-tier GP is not needed for purely digital Sequential 3D circuits. For the design of high-density 3D cells, the GP is not considered in order to achieve

high integration density. The trade-off between inter-tier coupling effect and integration density for 3D cells will require further technology analysis.

An engineering overhead for S3D technology is produced by IC design tools and their ability to efficiently synthesize and route in the 3rd dimension. However, the design technique proposed in this study, integrating MIVs in the cell design, has the advantage of a routing effort that is comparable to 2D technology. The routing step is fulfilled at the top tier, while at the bottom tier only power rails are connected. The use of a conventional IC design flow reduces the design overhead and tool dependency for S3D circuits.

II.B. 3D CELLS LIBRARY DESIGN

An extensive number of transistors are employed in the process of designing large-scale integrated circuits. In order to create digital designs, pre-designed and pre-characterized building blocks are commonly used to implement different logic functions. A selection of combinational and sequential standard cells, as fundamental building blocks, are compiled in a library. In the generation of large-scale designs, the building blocks defined in the library are selected and combined for producing the logic of the design.

The realization of sequential and combinational logic cells is evaluated to study the efficiency of the proposed 3D integration strategy. The partition is based on dividing the structure of standard cells in top and bottom tiers. When using MIVs, intermediate metal layers between tiers are introduced adding extra metal connections in the cell. For a standard cell with intra-cell partitioning, the cell structure is divided targeting the efficient use of MIVs by limiting inter-tier connections and achieving high-density integration.

As an illustrative example, the 3D integration of a flip-flop (DFF) cell, including the partition of the transistors in the schematic, is shown in Fig. 2. The schematic and layout views show MIVs employed for inter-tier connections. The input-output interface, Data-in (D), Clock Pulse (CP) and Data-out (Q), is on the top tier, while the bottom tier accommodates transistors for obtaining high-density cell integration. The routing step for digital designs is performed by connecting 3D cells interface ports at the top tier. The presented cells are fully characterized for inclusion in a large-scale design flow.

III. EXPERIMENTAL SETUP AND RESULTS

This section details the advantages of S3D technology for silicon area and wirelength gains by evaluating design optimizations with high-density 3D standard cells.

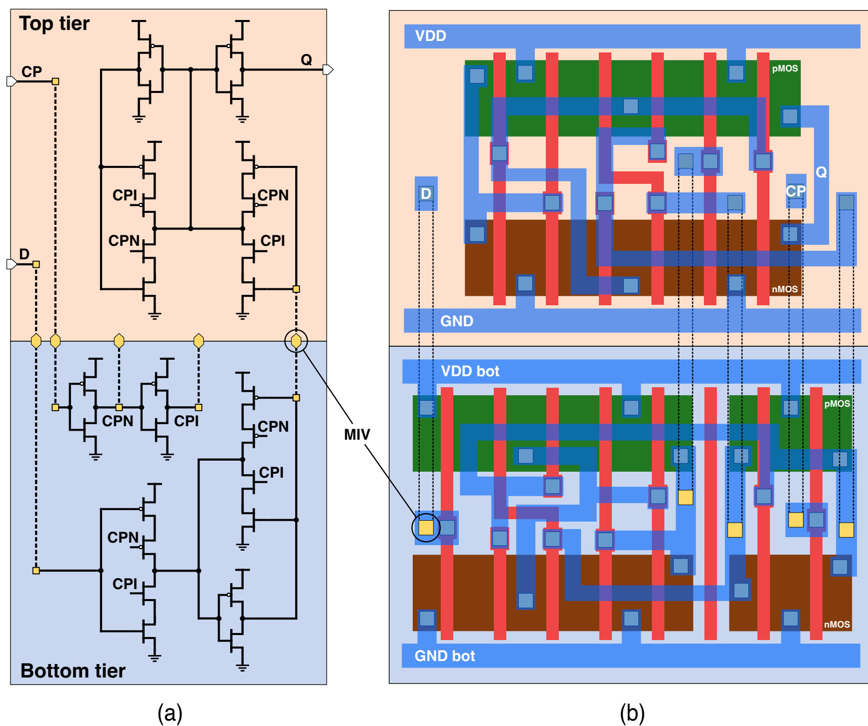


Fig. 2: Circuit partition of a DFF in two tiers showing MIVs for the 3D cell and circuit connections. (a) Schematic, and (b) layout.

III.A. METHODOLOGY

Different considerations for Sequential 3D integration technology are taken into account for this evaluation. Advances in S3D implementation enable the use of the same technology, i.e., 28 nm, in both stacked tiers [10]. Design rules for MIVs are considered to be the same as for other vias employed to connect metal layers, with the same area and space distance. As mentioned in section II.A, a ground plane is not required between tiers, considering negligible the coupling effect due to tier-to-tier interference for digital circuits.

The 3D cells compiled to a library are designed on schematic level with their correspondent physical layout. The standard cells partition for top and bottom tiers are created in independent views using Cadence Virtuoso, and verified with commercial 28 nm design rules. MIVs are modeled as connections between cell sub-circuits accommodated in each tier, introducing a 20 Ω resistance between M2b and M1, with irrelevant lateral coupling.

Different large-scale digital circuits, with the number of cells employed in the integration of each design, are presented in the benchmark evalua-

tions of [5, 6]. Silicon area gains are estimated by evaluating the proposed implementations with logic cells of our designed 3D standard cell library, and comparing to a commercial library. These benchmarks will provide representative results regarding the integration density achievable with our 3D standard cell library. Impact on power and timing exceeds the scope of this evaluation, and will be considered for future analysis.

The 3D cells are characterized to be included in a commercial design flow. The top and bottom tier cell partitions are separated for an individual realization, adapting the use of 3D cells with conventional IC design tools. For a digital circuit implementation, the cells position is defined by the placement of the top tier partitions. Then, routing of the signals in the top will finalize this design step. The power rails included in the bottom tier of standard cells define the mesh for the power of the bottom transistors.

III.B. RESULTS

In [11], benchmark implementations with different partition granularities and S3D optimizations are evaluated. For this work, the focus is on large-scale designs that will benefit from increased area and wirelength efficiency, performing the evaluation in simulation.

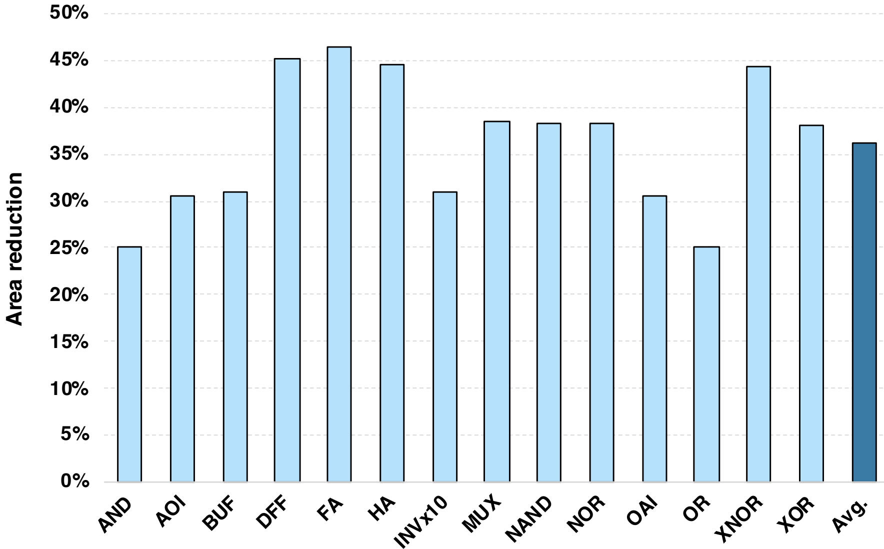


Fig. 3: Area reduction of designed 3D standard cell library compared to a commercial library.

The silicon area gain, presented as area reduction percentage, of the 3D cells library compared to a commercial library, is shown in Fig. 3. The average area reduction considering all logic cells in the library is estimated as 36 %. Logic cells with a larger transistor count achieve higher area gains, since a larger cell layout benefits more from the higher integration density achieved with the proposed technology.

Comparison with other works using an academic benchmark circuit, s38584, and a RISC-V processor, is shown in Table 1. The area reduction is estimated as the area difference between using our 3D standard cell library and a commercial 2D library. The wirelength reduction is estimated as the difference in the total length of wires in a layout generated for each case. The density of inter-tier connections is measured as MIV/cell, which determines the number of MIV employed in the circuit integration as a function of the number of cells of the design. A smaller factor indicates lower congestion of 3D vias, with reduced inherent parasitics of inter-tier connections, benefiting routing.

Table 1: Area, wirelength and MIV density comparison using benchmark circuit designs.

Source	[5]	[6]	**This work**	
Granularity	Transistor	Logic cells	Intra-cell	
Circuit	s38584	RISC-V	s38584	RISC-V
Frequency [MHz]	500	555	500	
Area red. [%]	29,5 - 38,3	23,6	39,0	44,0
Wirelength red. [%]	12,4 - 17,2	6,5	20,5	23,1
MIV/cell	4	2 - 9	3	

In [5], different large-scale integration designs are presented, and each of them is evaluated with three types of 3D cells based on transistor level partitioning with transistors accommodated on type-specific tiers (nMOS on top and pMOS on bottom). The three proposed implementations of 3D cells, each considering a different number of routing tracks, represent different percentages of area and wirelength reduction gathered in a range for the case of the benchmark circuit s38584 included in the comparison. In [6], two cases of Monolithic 3D (M3D) implementation for a RISC-V circuit are presented. The partitioning method accommodates SRAM memories in one tier and a RISC-V processor in the other, using MIVs for inter-tier connections. The implementation of the logic that is part of the RISC-V processor is considered for our evaluation.

Considering the circuit s38584, the use of the proposed 3D standard cell library achieves an area gain similar as the most efficient area implementation of [5]. However, the number of MIV per cell is reduced by 25 %, reducing the congestion of inter-tier connections. Considering the RISC-V design, the area gain is 20 % higher with up to 67 % less MIV/cell compared to [6]. Employing the same IC design flow as a planar implementation, our 3D standard cell library reduces the wirelength 23 % compared to a commercial 2D library. The proposed 3D integration method achieves higher wirelength reduction than the aforementioned works, showing an efficient Sequential 3D integration solution for reducing routing congestion on large-scale circuits.

Wire delay grows quadratically with the routing wirelength [12], which motivates the use of high-density cells to reduce distances between cells. They have the advantage of a reduced area and routing cost, which results in higher efficiency in terms of density, performance and energy. The presented evaluation is considered as a baseline scenario, where similar gain possibilities are expected for larger designs.

IV. CONCLUSION

This study presents the design of a 3D standard cell library using MIVs for intra-cell connections, which facilitates a high-density Sequential 3D integration in two tiers. The designed library is compatible with commercial IC design tools with a routing effort comparable to conventional technologies. The proposed design of 3D cells has an average of 36 % silicon area gain compared to commercial libraries. The implementation method is benchmarked using examples of large-scale integration circuits, showing up to 44 % silicon area gain, and 23 % wirelength improvement, with reduced congestion of 3D vias compared to other works.

V. ACKNOWLEDGMENT

This work has been supported by the EU's Horizon 2020 funding scheme in the project 3D-MUSE.

REFERENCES

[1] S. Bobba et al., "CELONCEL: Effective design technique for 3-D monolithic integration targeting high performance integrated circuits," *16th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 336-343, 2011.

[2] P. Vivet et al., "Monolithic 3D: an alternative to advanced CMOS scaling, technology perspectives and associated design methodology challenges," *25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 157-160, 2018.

[3] H. Sarhan, S. Thuries, O. Billoint and F. Clermidy, "An Unbalanced Area Ratio Study for High Performance Monolithic 3D Integrated Circuits," *IEEE Computer Society Annual Symposium on VLSI*, pp. 350-355, 2015.

[4] I. Radu et al., "Ultimate Layer Stacking Technology for High Density Sequential 3D Integration," *International Electron Devices Meeting (IEDM)*, pp. 1-4, 2023.

[5] C. Yan and E. Salman, "Mono3D: Open Source Cell Library for Monolithic 3-D Integrated Circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 3, pp. 1075-1085, 2018.

[6] S. Thuries et al., "M3D-ADTCO: Monolithic 3D Architecture, Design and Technology Co-Optimization for High Energy Efficient 3D IC," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1740-1745, 2020.

[7] E. Giacomin, F. Catthoor and P. -E. Gaillardon, "Area-Efficient Multiplier Designs Using a 3D Nanofabric Process Flow," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-5, 2021.

[8] P. Sideris, A. Peizerat, P. Batude, C. Theodorou and G. Sicard, "Inter-tier Coupling Analysis in Back-illuminated Monolithic 3DSI Image Sensor Pixels," *International Conference on Modern Circuits and Systems Technologies (MOCAST)*, Thessaloniki, pp. 1-4, 2021.

[9] P. Sideris et al., "Inter-tier Dynamic Coupling and RF Crosstalk in 3D Sequential Integration," *IEEE International Electron Devices Meeting (IEDM)*, pp. 3.4.1-3.4.4, 2019.

[10] T. Mota-Frutuoso et al., "3D sequential integration with Si CMOS stacked on 28nm industrial FDSOI with Cu-ULK iBEOL featuring RO and HDR pixel," *International Electron Devices Meeting (IEDM)*, pp. 1-4, 2023.

[11] S. Bobba, A. Chakraborty, O. Thomas, P. Batude, V. F. Pavlidis and G. De Micheli, "Performance analysis of 3-D monolithic integrated circuits," *IEEE International 3D Systems Integration Conference (3DIC)*, pp. 1-4, 2010.

[12] R. Ho, K. W. Mai and M. A. Horowitz, "The future of wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490-504, 2001.

Paper II

An Energy-Efficient Near-Memory Computing Architecture for CNN Inference at Cache Level

A non-von Neumann Near-Memory Computing architecture, optimized for CNN inference in edge computing, is integrated in the cache memory sub-system of a microcontroller unit. The NMC co-processor is evaluated using an 8-bit fixed-point quantized CNN model, and achieves an accuracy of 98 % on the MNIST dataset. A full inference of the CNN model executed on the NMC processor, demonstrates an improvement of more than 34× in performance, and 28× in energy-efficiency, compared to the baseline scenario of a conventional single-core processor. The design achieves a performance of 1.39 GOPS (at 200 MHz) and an energy-efficiency of 49 GOPS/W, with negligible area overhead of less than 1 %.

©2021 IEEE. Reprinted, with permission, from
M. Nouripayam, A. Prieto, V. K. Kishorelal, and J. Rodrigues,
"An Energy-Efficient Near-Memory Computing Architecture for CNN Inference at Cache Level," in *28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, Nov. 2021, pp. 1-4.

I. INTRODUCTION

Edge computing based neural networks (NN), convolutional neural networks (CNN) in particular, have shown a great potential in processing and transforming complex and large datasets into predictive outputs [1, 2]. Edge computing devices have the advantage of processing at proximity of data collection sources, and deliver higher energy-efficiency, shorter latency, and less hardware cost compared to cloud computing [3]. However, these devices are mostly resource constrained, e.g., energy, computational power, memory availability and associated silicon cost [4].

Depending on flexibility and performance, a computing platform employed for CNN edge devices, can be mainly divided into embedded application specific hardware accelerators (ASICs), Field Programmable Gate Array (FPGA) solutions, digital signal processors (DSPs), and MCU-based designs. CNN ASICs are attractive for their high performance and low-power characteristics, but they usually suffer from higher cost and lack of reconfigurability and flexibility for edge computing devices [5]. FPGA-based accelerators also offer high performance, however, they require a much larger power budget (usually 1-10 W) [20], which is a power expensive solution in tightly resource-constrained edge devices.

These limitations pronounce the need for MCU-based CNN designs, motivated by their low-cost, low-power budget ($\leq$ 1 W) and higher flexibility resulting from software programmability. However, performance limitations of MCU's conventional von Neumann architecture, i.e., the well-known "memory-wall" issue [6], as well as memory-intensive requirements of NNs, increasingly demand different optimization strategies [7].

In order to improve the efficiency of MCU-based CNN designs, considering the NN requirements, one approach is to focus on system level optimizations, such as specialized programming libraries for CNN computations [8, 9], or instruction set architecture (ISA) extension for supporting NN operations [10]. Another approach is more hardware-oriented, including design of a new core capable of handling computation-intensive NN operations [11], dedicated hardware accelerators and multi-core MCU-based systems [12, 13]. However, the aforementioned studies are based on the von Neumann architecture, and come with limited performance improvement, design complexities, and/or substantial hardware cost.

Near-memory computing (NMC) and computation-in-memory (CIM), as non-von Neumann architectures, are popular data-centric approaches, targeting the memory-wall issue by bridging the performance gap between memory and computation. These architectures, optimized for data movement reduction, are applicable to any level of a memory subsystem and improve

the system energy-efficiency and latency by moving computations spatially closer to the data location [14]. CIM realization techniques require in depth integration of computing logic into memories, and come with the cost of aggressive modifications of the internal memory architecture [15,16]. Furthermore, CIM poses different challenges, such as unconventional programming models, lack of efficient data mapping mechanisms, as well as power and hardware overhead [17,18]. Advantageously, NMC techniques do not require alterations in the internal structure of the memories, delivering high energy-efficiency and low latency at a limited hardware overhead.

In this work, an energy-efficient NMC architecture, realized as a co-processor, is integrated in the cache hierarchy of an MCU, i.e., the PULP open-source platform. The proposed design keeps flexibility of the system at a high degree by making use of the main processor for general purpose applications, and executing CNN operations using the NMC unit. The co-processor is designed using conventionally available SRAM solutions, which has the advantage of, 1) retaining area efficiency of foundry supplied SRAM, and 2) being non-disruptive in a conventional ASIC design flow.

The structure of this paper is organized as follows: Section II presents the proposed NMC architecture, and Section III demonstrates the experimental results compared with a baseline scenario and puts this work in perspective with other edge implementations. Finally, Section IV concludes this study.

II. PROPOSED NMC ARCHITECTURE

This section details different aspects of the proposed architecture, and starts with a brief introduction of the developed CNN model, followed by a description of the optimized data flow, as well as the constituting sub-units of the NMC co-processor. This architecture is an example for a tiny ML application using a CNN model on a benchmarked MNIST dataset.

II.A. NEURAL NETWORK MODEL

For the evaluation of the proposed NMC unit, a simplified CNN model for image classification is developed. The structure of the CNN model is shown in Fig. 1a, combining single convolution and pooling, as well as two fully-connected (dense) layers. An efficient hardware implementation is realized by converting the 32-bit floating-point network parameters to 8-bit fixed-point data format. Furthermore, in order to reduce the number of bits for data movement, computation outputs are truncated to 8-bit for data storage. The performed simulations have shown that the quantization

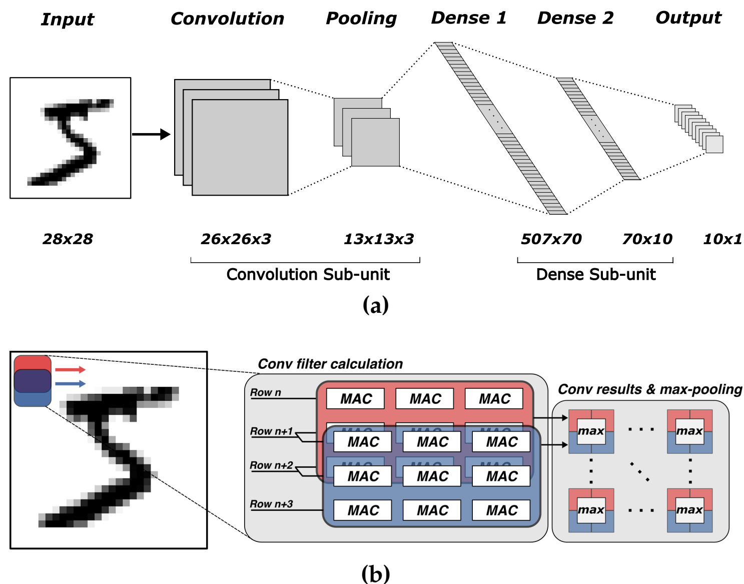


Fig. 1: (a) CNN model for MNIST dataset processing, (b) NMC convolution and max-pooling data-flow.

and precision reduction, have a marginal effect in the final accuracy of the network, delivering an average inference accuracy of 98%.

II.B. NMC UNIT OPERATIONS

The integration of an NMC architecture as a co-processor in the cache hierarchy makes recurrent data-accesses (load and store), that are required for processing by the main core, unnecessary. Advantageously, this integration in the memory sub-system provides direct data access, reduces the memory footprint and improves data management in the computation units. Ultimately, the reduction in data access cost results in an increased energy-efficiency and throughput, which in turn facilitates the employment of parallel computation units. The main operation performed in CNN processing is multiply-and-accumulate (MAC). Accordingly, a total number of 30 MAC units, split respectively in 18 and 12 dedicated MACs for convolution and fully-connected layers, are employed for the parallel computation of CNN operations. A data flow optimization is utilized so that multiple rows of an input image are accessed simultaneously by convolution filters, which facilitate the efficient generation of max-pooling results. This technique provides the benefit of performing convolution and pooling as part of one single operation, with no need for intermediate storage of convolution outputs.

Fig. 1b presents the data flow for the convolution and max-pooling steps. Allocating 18 dedicated MAC units for convolution creates a fully parallelized process of two matrices from an input image (using 3x3 convolution filters in two arrays of 9 MAC units), and generates two rows of convolution outputs at once, which are used for on-the-go computation of max-pooling. Consequently, the throughput and memory read-access are optimized considerably, due to the fact that only final results of max-pooling (in a succession of convolution-pooling) are stored in the memory.

II.C. NMC ARCHITECTURE

The NMC architecture is developed bearing in mind the overall energy-efficiency of the entire system. A scalable and MCU platform-agnostic NMC architecture is accomplished by employing existing memory resources, meeting the goal of a non-disruptive design flow. The memory-wall issue is tackled with this technique, which practically provides increased bandwidth and uses the memory resources more efficiently than the case of running an NN on MCU using general-purpose instructions. For demonstration purpose, the PULPissimo single-core microcontroller, as a low-power MCU, is chosen [19]. The dedicated computation logic is tailored for CNN application, providing efficient memory accesses and reduced data movement through the system.

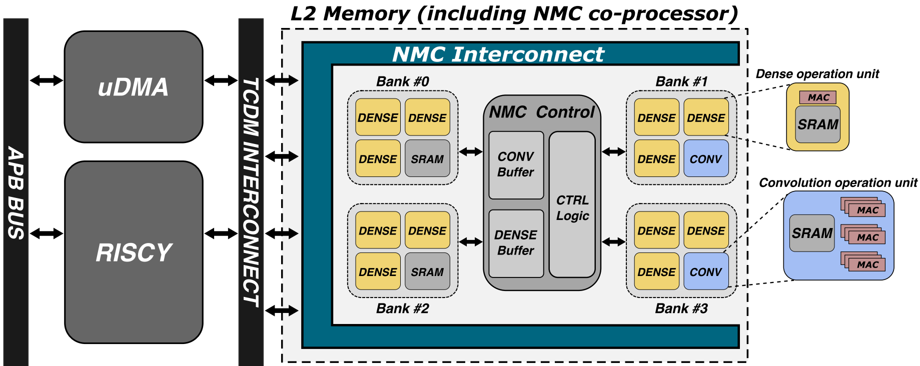


Fig. 2: NMC architecture integrated in the memory sub-system of PULPissimo.

Fig. 2 shows a high-level representation of the MCU architecture with the integrated NMC co-processor. The memory sub-system is adapted by including the NMC control, NMC interconnect, re-purposed SRAM macros and their dedicated MAC units.

NMC Control: orchestrates all operations in the network and organizes the data accesses, including buffers targeting an efficient data allocation for convolution and dense operations performed in the memory banks.

NMC Interconnect: functions as a wrapper around the memory banks and NMC sub-units, and establishes all the communications between RISCY core, TCDM interconnect bus and memory banks.

NMC Convolution unit: accelerates convolution by employing 18 dedicated MAC units near two SRAM macros and a parallel flow over 3x3 convolution filters. The pixel rows of an image are read from the memory, stored in an input buffer, and thereafter, the MAC units concurrently perform convolution. The weights for the convolution layer are stored in the MAC units associated register file, creating a weight-stationary taxonomy. Therefore, the reuse of the weights is maximized in each MAC unit, which reduces the cost of energy expensive data movement by memory accesses. The organization of MAC units minimizes the computation stall time of pooling operation by the parallel calculation of two convolutions per cycle, resulting in one pooling output every two cycles, which is stored back in the output SRAM.

NMC Dense unit: performs the dense layer operations while the NMC Control organizes the flow by, 1) fetching the distributed dense layer weights from 12 SRAM macros to their dedicated MAC units, and 2) flattening the outputs of the pooling layer in a buffer and distributing them among the MAC units in parallel.

The number of MAC units and size of storage buffers may be adapted for larger application scenarios, with low-power, memory-efficient and high throughput considerations for running neural networks at-the-edge.

III. DISCUSSION

In this section, the performance improvements of the proposed non-von Neumann NMC architecture are evaluated. The target platform for integration of the NMC co-processor is the well-researched PULPissimo, a single-core SoC from the low-power PULP MCU [19]. The 512 kB Level 2 (L2) cache memory, consisting of four re-purposed memory banks, is the design space of the NMC co-processor, and equips the system with accelerated CNN inference computations. Thus, the overall workload of the MCU is reduced, which frees resources for simultaneous execution of general-purpose operations. The proposed design accommodates MAC units in the memory sub-system, achieving a lower area cost compared to multi-core or dedicated hardware-accelerator alternatives [7]. Specifically, the re-purposed SRAM banks in the MCU cache memory, helps to avoid the employment of extra dedicated memories, which is usually required for hardware accelerators. The results

from synthesis in a 28 nm FD-SOI process technology show that the NMC unit of this work is realized with a negligible area overhead (<1%).

This study is framed by the evaluation of a baseline and target scenario, performing a CNN full inference. The baseline and target scenarios are characterized by employing a single-core MCU and the NMC unit in cache, respectively. Table 1 represents the results of this study, as well as positions the proposed design to other MCU-based NNs. Performance is evaluated by post-synthesis simulations, using back-annotated toggle information, and reveals that a performance of 1.39 GOPS and an energy-efficiency of 49 GOPS/W at 200 MHz is achieved by integrating the NMC unit in the MCU. These results, compared to the baseline scenario, demonstrate a considerable performance and energy-efficiency improvement of 34× and 28×, respectively. The performance numbers are based on cycle counts and are obtained by a hardware timer, which captures the time before and after running the application in each of the examined scenarios.

Table 1: Comparison of the proposed NMC architecture with the baseline MCU and other edge implementations.

Source	**[20]**			**[21]**		**[22]**	**[23]**	**This Work**	
	GAP-8	**STM32H7**	**STM32H4**	**GAP-8**	**STM32F4**			**Baseline**	**Target**
Platform	MCU	MCU	MCU	MCU	MCU	FPGA	APSoC	MCU	
Core	8	1		8	1	—	—	1	
Design level	Library (PULP-NN)	Library (CMSIS-NN)		ISA opt.	—	Accel.	Accel.	—	NMC @Cache
Dataset	CIFAR-10			Freesound		MNIST	MNIST	MNIST	
Network	CNN			BNN		LeNet-5	LeNet-5	CNN	
Data width	INT-8			Binary		16-bit	—	8-bit	
Clock [MHz]	170	480		150	480	—	650	200	
Performance [GOPS]	1.07	0.14	0.03	1.5	0.15	20.3	46.15	0.04	1.39
Energy Eff. [GOPS/W]	16.1	0.97	1.7	31.3	0.61	30.03	16.19	1.72	49

The proposed NMC, see Table 1, can achieve 1.3× and 3× higher performance and energy-efficiency compared to GAP-8 MCU in [20], and similar performance with 1.5× higher energy-efficiency in comparison to GAP-8 MCU in [21]. Moreover, the proposed NMC delivers a range of 9-46× higher performance, as well as 28-80× higher energy-efficiency compared to the presented commercial MCUs. Compared to the proposed MCU-based design, the realizations in [22, 23] achieve a higher performance with dedicated NN hardware accelerators, but a notable lower energy-efficiency of 1.6× and 3×, respectively. Additionally, FPGA-based designs offer less flexibility and require higher power budget, which in turn make these solutions less attractive for a tightly-constrained edge device.

In Fig. 3a, the performance improvement for the target scenario, with reference to the baseline implementation, is presented. The results are represented as operations per cycle (OP/cycle). For a full inference, the baseline MCU only completes 0.2 OP/cycle, however, using the NMC co-processor, it is possible to achieve ≈7 OP/cycle. The two scenarios are evaluated individually to compare how convolution, pooling and dense operations are improved by the NMC sub-units. The simulation results reveal that the dense operation in the MCU platform requires around 1.5× more cycles than convolution-pooling combined. However, by employing the NMC unit, a more balanced computation time is achieved, as the cycle count for the dense layer computation is 1.1× of convolution-pooling. As shown in Fig. 3a, the NMC unit delivers a performance improvement of 28× for convolution-pooling and 38× for dense, respectively. This, in turn, can be translated to an overall performance improvement of 34×.

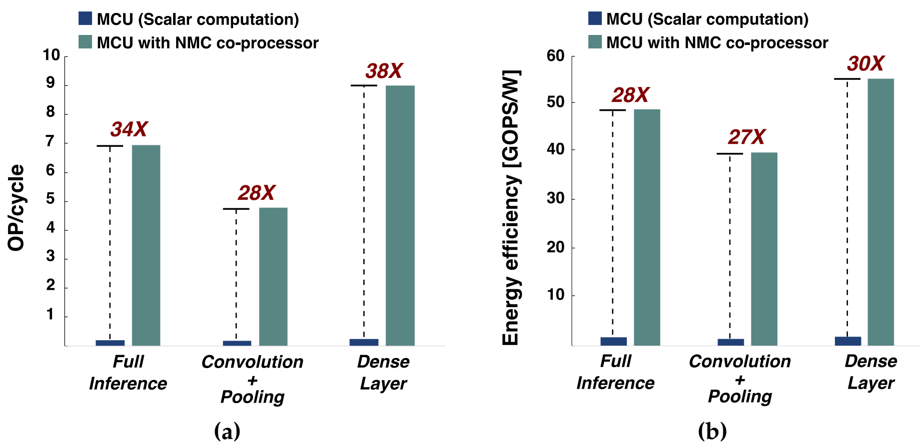


Fig. 3: Comparison between MCU and NMC at cache for (a) performance, (b) energy-efficiency.

The energy-efficiency improvement for the target scenario is shown in Fig. 3b. The results show a similar energy-efficiency improvement for the convolution-pooling combined, and the dense operations. This, in turn reflects that the NMC sub-units are well-balanced for computation of different CNN layers. The improvement for convolution-pooling is 27×, and for dense operations 30×, which is translated to an overall energy-efficiency gain of 28×. The proposed NMC architecture is portable to other processor platforms, and similar performance improvements are expected.

IV. CONCLUSION

This study presents a non-von Neumann NMC architecture for CNN image classification on edge devices. The hardware-friendly NMC architecture is integrated as a co-processor in the cache level of an MCU, by interleaving the memory with the computation logic, while deploying conventional SRAM. The proposed architecture is scalable for CNN applications and is MCU platform-agnostic, with the benefit of pushing computation logic closer to data locations as a solution to the well-known memory-wall challenge. This study is carried out on a 8-bit fixed-point quantized CNN model on the MNIST dataset. The experimental results show that the NMC design is able to deliver the performance and energy-efficiency of 34× and 28× over the baseline study, which is a single-core general purpose MCU. The designed NMC unit delivers a peak performance of 1.39 GOPS and energy-efficiency of 49 GOPS/W when running at a frequency of 200 MHz.

ACKNOWLEDGMENT

The authors are grateful to Vetenskaprådet and H2020 3D-MUSE for financial support.

REFERENCES

[1] W. G. Hatcher and W. Yu, "A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends," *IEEE Access*, pp. 24411-24432, 2018.

[2] A. Ganguly, R. Muralidhar and V. Singh, "Towards Energy Efficient non-von Neumann Architectures for Deep Learning," *20th International Symposium on Quality Electronic Design (ISQED)*, pp. 335-342, 2019.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, pp. 637-646, 2016.

[4] W. Yu et al., "A Survey on the Edge Computing for the Internet of Things," *IEEE Access*, pp. 6900-6919, 2018.

[5] A. Reuther et al.,"Survey and Benchmarking of Machine Learning Accelerators," *IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1-9, 2019.

[6] S. McKee, "Reflections on the Memory Wall," *Proceedings of the 1st Conf. on Comp. fontiers*, 2004.

[7] D. Rossi et al., "PULP: A parallel ultra-low power platform for next generation IoT applications," *IEEE Hot Chips 27 Symposium (HCS)*, pp. 1-39, 2015.

[8] A. Capotondi, M. Rusci, M. Fariselli and L. Benini, "CMix-NN: Mixed Low-Precision CNN Library for Memory-Constrained Edge Devices," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 871-875, 2020.

[9] X. Wang, M. Magno, L. Cavigelli and L. Benini, "FANN-on-MCU: An Open-Source Toolkit for Energy-Efficient Neural Network Inference at the Edge of the Internet of Things," *IEEE Internet of Things Journal*, pp. 4403-4417, 2020.

[10] Garofalo et al., "XpulpNN: Accelerating Quantized Neural Networks on RISC-V Processors Through ISA Extensions," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 186-191, 2020.

[11] M. Gautschi et al., "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 2700-2713, 2017.

[12] F. Conti, P. D. Schiavone, L. Benini, "XNOR Neural Engine: A Hardware Accelerator IP for 21.6-fJ/op Binary Neural Network Inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 2940-2951, 2018.

[13] E. Flamand et al., "GAP-8: A RISC-V SoC for AI at the Edge of the IoT," *IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 1-4, 2018.

[14] M. Gao, G. Ayers and C. Kozyrakis, "Practical Near-Data Processing for In-Memory Analytics Frameworks," *International Conference on Parallel Architecture and Compilation (PACT)*, pp. 113-124, 2015.

[15] S. Yin, Z. Jiang, J. Seo and M. Seok, "XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks," *IEEE Journal of Solid-State Circuits*, pp. 1733-1743, 2020.

[16] J. -W. Su et al., "16.3 A 28nm 384kb 6T-SRAM Computation-in-Memory Macro with 8b Precision for AI Edge Chips," *IEEE International Solid- State Circuits Conference (ISSCC)*, pp. 250-252, 2021.

[17] G. Singh et al., "A Review of Near-Memory Computing Architectures: Opportunities and Challenges," *21st Euromicro Conference on Digital System Design (DSD)*, pp. 608-617, 2018.

[18] K. -T. Tang et al., "Considerations of Integrating Computing-In-Memory and Processing-In-Sensor into Convolutional Neural Network Accelerators for Low-Power Edge Devices," *Symposium on VLSI Technology*, pp. T166-T167, 2019.

[19] P. D. Schiavone et al., "Quentin: an Ultra-Low-Power PULPissimo SoC in 22nm FDX," *IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, pp. 1-3, 2018.

[20] Garofalo A, Rusci M, Conti F, Rossi D, Benini L. "PULP-NN: accelerating quantized neural networks on parallel ultra-low-power RISC-V processors," *Phil. Trans. R. Soc*, 2019.

[21] Cerutti. G et al., "Sound event detection with binary neural networks on tightly power-constrained IoT devices." *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 19-24. 2020.

[22] W. Chen et al., "An Asynchronous Energy-Efficient CNN Accelerator with Reconfigurable Architecture," *IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pp. 51-54, 2018.

[23] V. R. Laguduva et al., "Dissecting Convolutional Neural Networks for Efficient Implementation on Constrained Platforms," *33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID)*, pp. 149-154, 2020.

Paper III

A Scalable All-Digital Near-Memory Computing Architecture for Edge AIoT Applications

With the growing need to process large volumes of data, edge computing near data collection sources has become increasingly important. However, the resource constraints of edge devices require more efficient data processing techniques. Near-memory computing (NMC) presents an efficient solution, especially for data-intensive applications, by enabling processing that is both energy-efficient and hardware optimized. This work introduces a platform-agnostic NMC architecture tailored for convolutional neural network (CNN) workloads, integrated into the shared cache memory subsystem of a microcontroller unit (MCU). An open-source RISC-V MCU is chosen as the target platform due to its flexibility and low-power architecture. The NMC co-processor, operating alongside the general-purpose RISC-V core, forms a multi-core system-on-chip that combines low hardware cost with high energy efficiency, while maintaining a high degree of flexibility. The proposed design offers a configurable architecture capable of processing a wide range of CNN models with a computational efficiency of 94 %. For evaluation purposes, widely recognized CNN benchmark models are utilized, showing a performance of 96 GOPS and an energy efficiency of 1828 GOPS/W for 8-bit precision at 200 MHz. These results represent a significant improvement over both highly customized state-of-the-art hardware accelerators and multi-core MCU solutions.

©2025 IEEE. Reprinted, with permission, from
M. Nouripayam, A. Prieto, and J. Rodrigues,
"A Scalable All-Digital Near-Memory Computing Architecture for Edge AIoT Applications," in *IEEE Access*, vol. 13, pp. 108609-108625, Jun. 2025.

I. INTRODUCTION

The rapid growth of data-centric applications, particularly Artificial Neural Networks (ANNs), has driven a paradigm shift in computing architectures, highlighting the need for efficient and scalable processing solutions. ANNs, highly effective at executing complex tasks across diverse applications, typically require data centers and cloud infrastructures to meet their intensive computational demands. However, growing concerns about latency, energy consumption, data privacy, and security are accelerating the adoption of edge computing. By processing data closer to its source, edge computing improves energy efficiency, reduces latency, and mitigates communication bandwidth limitations [1]. Despite the advantages of local data processing, edge computing platforms face inherent challenges, including limited energy budgets, restricted compute capacity, and constrained memory. Additionally, the rapid evolution of artificial intelligence (AI) algorithms and emerging applications poses challenges related to costly system transformations and complex adaptation across diverse hardware technologies [2].

Addressing edge AI challenges requires hardware architectures that strike a balance in the efficiency–flexibility design space, particularly to support dynamic and evolving AI workloads. Flexibility in this context refers to the ability to deploy low-power, high-performance, and reconfigurable edge AI hardware that adapts to varying algorithmic demands while maintaining energy efficiency and scalability [3]. Edge computing hardware spans a wide spectrum, from energy-efficient application-specific integrated circuits (ASICs) to more versatile, but less energy-efficient, general-purpose processors such as microcontroller units (MCUs) [4]. MCUs are valued for their programmability, cost-effectiveness, and low power consumption ($\leq 1W$). However, their performance is constrained by the traditional von Neumann architecture, including memory bandwidth bottlenecks and the high energy cost of data transfers [5].

Hardware platform limitations are closely linked to the "memory wall", a fundamental performance bottleneck caused by the widening gap between processor speed and memory access latency. In data-centric applications, data movement is the dominant contributor to power consumption [6]. From an energy perspective, as illustrated in Fig. 1a, off-chip data transfers consume up to 200$\times$ more energy than a multiply-accumulate (MAC) operation, while even accessing on-chip memory requires 6$\times$ more energy [7]. The energy cost of data movement highlights the critical role of on-chip memory in improving performance and energy efficiency, especially in resource-constrained edge devices. Consequently, effective memory utilization and organization are essential and require a delicate balance of memory hierarchy, access patterns, capacity, and hardware cost.

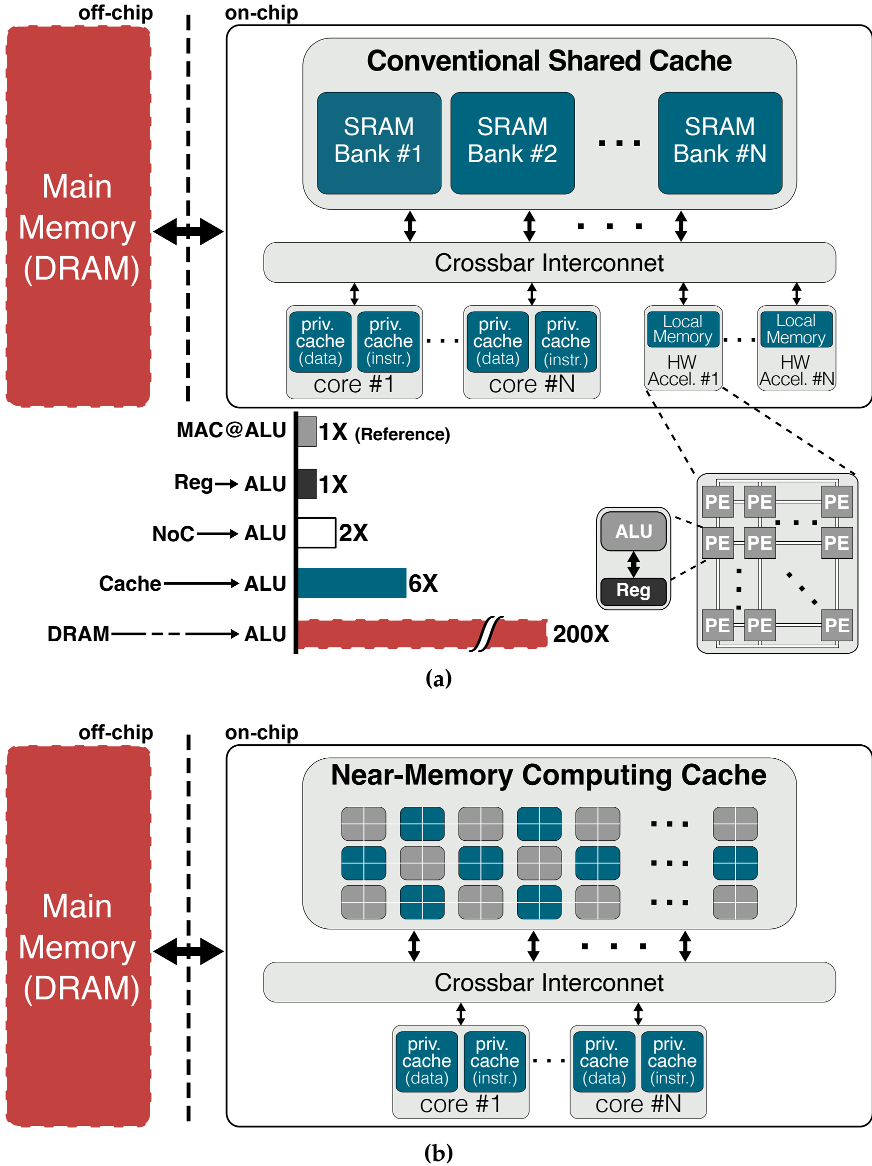


Fig. 1: (a) A conventional computing architecture with memory subsystems, and the normalized energy cost of data movement with reference to one computation at an Arithmetic Logic Unit (ALU) [7], (b) a near-memory computing cache architecture.

Techniques such as near-memory computing (NMC), illustrated in Fig. 1b, have emerged as promising solutions to overcome the limitations of conventional architectures. By performing computations near data, NMC significantly reduces data movement cost, thereby improving both performance and energy efficiency [8]. However, general-purpose central processing units (CPUs) remain essential in AI systems, managing critical operations of pre/post-processing, data movement, and non-computational control tasks, often becoming major sources of system latency. Conventional architectures integrating CPUs, ASICs, and direct memory access (DMA) engines experience processor stalls and underutilization. For instance, deep neural network (DNN) computations typically account for only 12-50 % of the total runtime, with the remainder dominated by CPU-bound operations and data transfers [9]. To overcome processing bottlenecks, a holistic design approach is required that exploits NMC to reduce unnecessary data movement by optimizing memory access patterns and data paths for applications with large memory footprints. Co-optimizing data access and computation is crucial for achieving high energy efficiency and sustained performance in edge computing platforms.

This article introduces a computationally efficient architecture that integrates an NMC technique to address memory and energy constraints in edge devices, while capitalizing on the low-power, flexible, and reprogrammable nature of MCUs. The key contributions of this work are summarized as follows:

- An in-cache digital NMC framework for edge AI acceleration, integrated within an MCU's memory subsystem. The design enables tight coupling between compute and storage while preserving efficiency of foundry-provided SRAM structures.
- A platform-agnostic implementation validated through FPGA integration and ASIC synthesis using a commercial CMOS technology demonstrating scalability, energy efficiency, and area feasibility.
- Development of a modular and configurable NMC architecture, with independent control of processing units embedded in memory banks, supporting concurrent execution of neural networks adapted to workload demands.
- CPU-transparent convolutional neural network (CNN) execution, combining compile-time control of tiling, loop scheduling, and memory partitioning with runtime coordination via interrupt-driven synchronization and shared memory access with a RISC-V.

The remainder of this paper is structured as follows: Section II presents the relevant methods and studies as the theoretical background and challenges that lead to the suggested technique in this paper. Section III details the

modules of the proposed NMC architecture. Post-synthesis analysis of power and performance of the design is presented in Section IV. Finally, Section V concludes the paper by summarizing the key challenges and contributions of this work.

II. RELATED WORK

While various studies attempt to mitigate performance and energy limitations of von Neumann architecture, several challenges persist, including resource constraints, hardware reliability, and architectural trade-offs in flexibility, processing efficiency, technology portability, and design complexity. To establish the rationale behind our design choices, this section reviews existing research on memory access optimizations and utilization in constrained environments. The advancements in NMC are highlighted, considering the impact on performance and energy efficiency and the remaining limitations that have motivated our proposed solution.

II.A. DOMAIN-SPECIFIC EDGE ACCELERATORS

Domain-specific accelerators (DSAs) such as systolic arrays [10], vector processing units (VPUs) [11], and multi-core clusters [12] are widely used to enhance inference performance through spatial parallelism and data reuse. These architectures typically depend on on-chip scratchpad memory (SPM) to avoid cache overhead and require high bandwidth between buffers and DRAM to sustain low-latency computation. However, DSAs face scalability and memory access challenges, including increased energy consumption from frequent data transfers and the complexity of integrating memory-centric computation into AI pipelines. Ensuring data consistency across distributed memory systems and reducing host CPU involvement remain ongoing challenges [13].

While DSAs are efficient for specific tasks, their rigidity limits adaptability across workloads. To address versatility, reconfigurable architectures offer a more adaptable alternative. Solutions such as FPGAs and multi-core clusters [14] provide hardware flexibility and increased programmability while preserving parallelism. However, reliance on dedicated local memory introduces complex data management at the software level. Despite gains in throughput, energy efficiency remains limited, and static reconfiguration often leads to suboptimal resource utilization [15]. FPGAs typically are less energy efficient and unsuitable for edge IoT applications, while low-power variants lack the computational capacity for demanding AI workloads [16]. Hybrid FPGA-SoCs, with embedded FPGAs (eFPGAs), enable customizable

acceleration, but are restricted by their dependence on mature technology nodes such as 180 nm and 130 nm [17], limiting performance and energy efficiency. A major constraint of FPGA-SoCs is the dependency on external memory, reinforcing the inefficiencies of conventional architectures. Unlike traditional FPGAs, eFPGAs lack dedicated block SRAMs, and instead rely on the host SoC's memory subsystem, increasing data movement and reducing energy efficiency. Furthermore, integrating eFPGAs into SoCs introduces ≈ 25 % area overhead and increased power consumption due to programmability. Further challenges include routing congestion, interconnect limitations, and the complexity of integrating eFPGAs in heterogeneous systems requiring high-speed, low-latency communication [16, 17].

Although DSA architectures efficiently handle parallel workloads and spatial data reuse, they continue to rely on local memory and shift the burden of data movement to the host CPU. To overcome data inefficiencies, our proposed design integrates CPU-transparent NMC acceleration for CNN workloads directly within the MCU's shared memory. This approach preserves the MCU's flexibility, reduces unnecessary data transfers, and enhances efficiency by combining programmable accelerators with NMC.

II.B. NEAR-MEMORY VS. IN-MEMORY COMPUTING

Near-memory computing and in-memory computing (IMC) both aim to overcome the limitations of the von Neumann architecture by reducing data movement [8]. IMC integrates computation directly into memory, either at the bitcell level or within peripheral circuits, effectively downscaling data transfer latency. Hybrid IMC designs split computation between memory arrays and periphery, using analog-to-digital converters (ADCs) and sensing circuits, making them suitable for Boolean logic operations [18]. Additionally, by embedding computation into memory arrays, IMC often disrupts conventional memory structures and introduces circuit complexity, which increases design and integration challenges. In contrast, NMC places processing elements (PEs) adjacent to memory, preserving a logical separation between storage and computation. This architectural modularity preserves memory array integrity and enables flexible deployment across memory hierarchy levels, offering greater scalability and adaptability, especially for energy-efficient, high-throughput systems.

IMC implementations span various memory technologies, each with tradeoffs. Emerging non-volatile memory (eNVM), including FeFET, STT-MRAM, and PCM, enable high-density in-memory computation while offering benefits such as reduced power consumption and improved access speeds [19], but face reliability issues. DRAM-based IMC, on the other hand, suffers from costly off-chip data transfers that hinder scalability [20, 21].

SRAM-based IMC, including analog (AIMC) and digital (DIMC), are well researched techniques for energy-efficient, low-latency edge AI [22]. AIMC enables parallel bit-wise operations via bitline computing and multiple word-line activation [23], but is inherently intrusive to conventional SRAM architectures, limiting scalability and integration. AIMC also suffers from precision issues due to process, voltage, and temperature (PVT) variations and incurs significant area overhead from analog circuitry and ADCs [24]. As a result, AIMC adoption needs to prioritize overall system-level efficiency over peak performance for balanced deployment [25].

DIMC, in contrast, provides greater flexibility, programmability, and robustness by executing digital MAC operations at the SRAM cell level with full-precision accumulation in registers. However, additional logic comes with increased area and energy overhead. Moreover, energy and area scale with word length, requiring trade-offs between efficiency and precision. Challenges such as energy overhead, data consistency, and the lack of standardized programming frameworks further hinder programmability and system-level optimizations [26].

Unlike IMC, NMC places compute units near memory, using dedicated logic and high-bandwidth interconnects to reduce data movement without disrupting internal memory structures [27,28]. NMC is particularly effective for multi-bit operations and computationally intensive tasks, offering greater architectural flexibility and energy-efficiency. A behavioral-level NMC design using high-level synthesis (HLS) proposes generating loosely coupled accelerators near on-chip memory, offering modest flexibility for rapid hardware development [29]. However, reliance on synthesis tools and high-level abstraction overlook low-level optimizations and implementation complexity, ultimately limiting efficiency gains. RISC-V based NMC architectures, such as NM-Caesar and NM-Carus, provide scalable and configurable solutions with low integration effort via compatible interfaces with SRAM, offering memory-compute interoperability and suitability for embedded MCUs, though with limited peak performance [30].

Our in-cache digital NMC integration introduces a flexible framework with a non-disruptive design flow, enabling scalable and low-power architectures that balance performance and energy efficiency, well-suited for next-generation edge AI. By combining a programmable RISC-V core with optimized memory access and enhanced computing capabilities, the proposed framework supports general-purpose NMC based MPSoCs with configurable CNN accelerators near SRAM. The framework adaptability enables efficient execution of diverse edge AI workloads and promotes the practical deployment of energy-efficient AI hardware in resource-constrained environments.

III. NMC ARCHITECTURE ORGANIZATION

The proposed design is the result of an extensive design space exploration (DSE) that evaluates key metrics and trade-offs among essential design parameters. The multivariate DSE is illustrated in Fig. 2, providing a comparative assessment across a representative set of architectures. Performance and energy efficiency are directly extracted from the respective studies and ranked comparatively to ensure a fair evaluation across different designs and platforms. For general-purpose programmability, the highest ranking is assigned to designs featuring at least one programmable core, while architectures with fixed, predefined operations have limited programmability. Similarly, dynamic configurability is ranked based on the number of supported modes and operations, reflecting the design's adaptability and reconfigurability. Non-idealities of IMC include noise susceptibility, reducing robustness compared to all-digital designs.

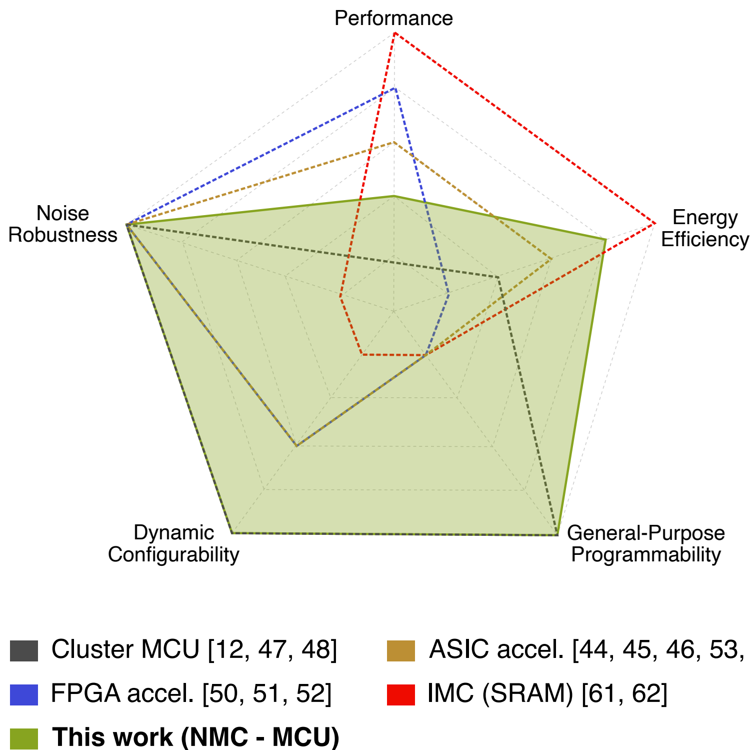


Fig. 2: Comparison between different designs and platforms in terms of: performance, energy-efficiency, general-purpose programmability, dynamic runtime configurability, and noise robustness.

Given the rapidly evolving landscape of edge AI applications and their varying demands, our proposed design emphasizes flexibility and cost-effectiveness to ensure competitiveness and adaptability. Moreover, high performance and energy efficiency are maintained, achieving a well-balanced trade-off between scalability, cost, and computational power.

Existing research highlights domain-specific accelerators based on IMC architectures for their exceptional performance and low power consumption per processed bit. However, lack of flexibility and reduced overall system efficiency limit adaptability to diverse application requirements. To overcome processing limitations, our proposed framework integrates the computational efficiency of near-memory computing with the versatility of a general-purpose microcontroller, ensuring seamless integration into existing architectures, while maintaining lower design costs and enhanced flexibility. The NMC approach achieves an optimal balance between performance, power efficiency, and adaptability, effectively leveraging the strengths of both paradigms to meet the evolving requirements of AI-driven applications.

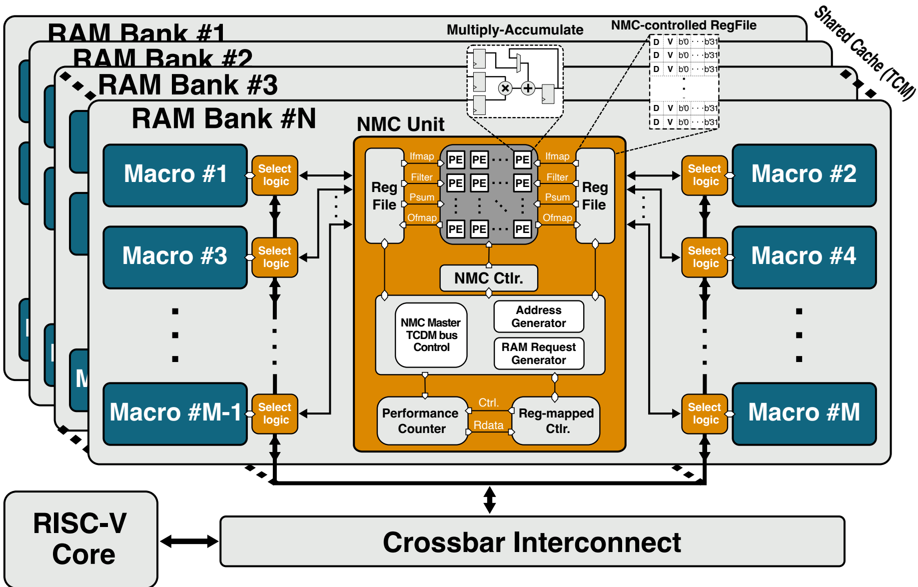


Fig. 3: Architecture of NMC units integrated in the memory subsystem of an MCU platform, including bank partitioning, memory macros, and processing and control logic for hardware acceleration.

The integration of an NMC design in the L2 shared cache memory structure of an MCU platform, as illustrated in Fig. 3, effectively transforms the hardware architecture into a multi-core parallel data processing system. By

addressing data consistency, synchronization overhead, and memory access contention, this proposal leverages multi-core parallelism to integrate seamlessly within MCU platforms and maximize both throughput and energy efficiency.

The shared L2 cache is implemented as a high-speed SPM, a well-established approach in real-time embedded systems due to its high density, speed, and predictable access patterns [31]. Scratchpads, mapped onto the processor's address space within a predefined range, are commonly employed to simplify caching logic and maintain full functionality without contention in multi-core environments. In SPMs, commonly known as "software-controlled caches", data transfer and allocation tasks are managed entirely by software [32]. By eliminating the need for a complex hardware caching controller, SPMs enhance energy efficiency, particularly in applications with predictable data access patterns. Furthermore, the software-driven approach offers flexibility, allowing the programmer/compiler to explicitly manage the appropriate memory regions for data transfers [33].

Performance is further optimized by partitioning the shared cache into multiple memory banks and enabling access through an interleaved addressing scheme. A partition configuration enables multi-channel communication, thereby increasing the I/O bandwidth for the NMC design and improving memory access efficiency. Additionally, the NMC units reduce unnecessary data movement by directly accessing data stored in the L2 memory banks. This internal memory access strategy improves data lifecycle efficiency, reducing the need for frequent data transfers and intermediate storage overhead. Consequently, the system achieves enhanced computational and energy efficiency, along with optimized memory bandwidth, ensuring a balanced trade-off between flexibility, performance, and power.

III.A. MEMORY STRUCTURE AND BANDWIDTH

In SoCs, dedicated local buffers are commonly used to enhance memory bandwidth and performance. Large-capacity buffers help to reduce data movement, but come with high area and energy overhead. Conversely, smaller buffers, although more area-efficient, are less effective in reducing data transfers. In hardware acceleration systems, SRAM can occupy 40–90% of the area and dominate the power consumption [34]. Shared caching balances area, power, and bandwidth by allowing multiple accelerators to access a common memory pool instead of relying on dedicated local buffers [35, 36]. By merging dedicated accelerator SRAMs and buffers into a shared memory configuration, data duplication and unnecessary temporary storage are eliminated, leading to improved area efficiency, lower power consumption, and enhanced performance. Beyond efficient memory sharing, a well-optimized

memory organization that divides large memory blocks into smaller macros further optimizes effective load capacitance (C_{eff}) and effective frequency (F_{eff}), reducing power consumption and access latency.

In the proposed architecture, the L2 SPM is implemented as a Tightly Coupled Memory (TCM), shared between RISC-V and NMC units, to enable low-latency and high-throughput data access. To mitigate contention and support concurrent access, the SPM is partitioned into banks, each with a dedicated data bus. While a large number of banks reduces access conflicts, greater interconnect complexity is introduced, degrading latency. For example, as reported in [37], scaling from 2 to 128 banks within a crossbar interconnect results in a frequency drop of 12 %. With the RISC-V core and NMC units working as independent consumers, a banking factor of two per consumer has empirically shown to minimize conflicts without incurring high interconnect complexity ($\#Banks = \#Consumers \times BankingFactor$) [38], resulting in a four-bank memory subsystem. The address space of each bank is further partitioned into eight memory macros, for data distribution in a system with increased memory bandwidth. From the system-level perspective, the RISC-V core accesses all banks uniformly, remaining agnostic to internal organization, while NMC units exploit the macro-level granularity for parallel data streaming to tailored PEs. Although the number of macros can be scaled up to support larger memory bandwidth, arbitrarily increasing the macro counts may degrade memory density and expand data bus complexity. Thus, our eight-macro-per-bank, two-bank-per-consumer scheme strikes a practical balance between performance, area efficiency, and interconnect complexity.

To further illustrate the impact of the proposed memory organization, Fig. 4 presents timing diagrams for memory access under various scenarios involving the RISC-V core and NMC units. Each memory bank consists of M memory macros and is coupled with a single NMC unit. In the baseline scenario (Fig. 4a), the RISC-V's memory bandwidth is limited to a single macro access per cycle. In contrast, as shown in Fig. 4b, the active NMC unit accesses multiple macros concurrently, as long as they are not already used by the RISC-V. The proposed macro-level parallelism improves the effective bandwidth available to the NMC unit without requiring additional physical ports. The proposed access arbitration enables NMC units to utilize all available macros within a bank, significantly increasing its throughput. The resulting bandwidth for both the NMC architecture and the RISC-V core is given by:

$$BW_{NMC} = (\#Banks) \times (\#Macros) \quad [\frac{access}{cycle}] \tag{1}$$

$$BW_{Core} = (\#Banks) \quad [\frac{access}{cycle}] \tag{2}$$

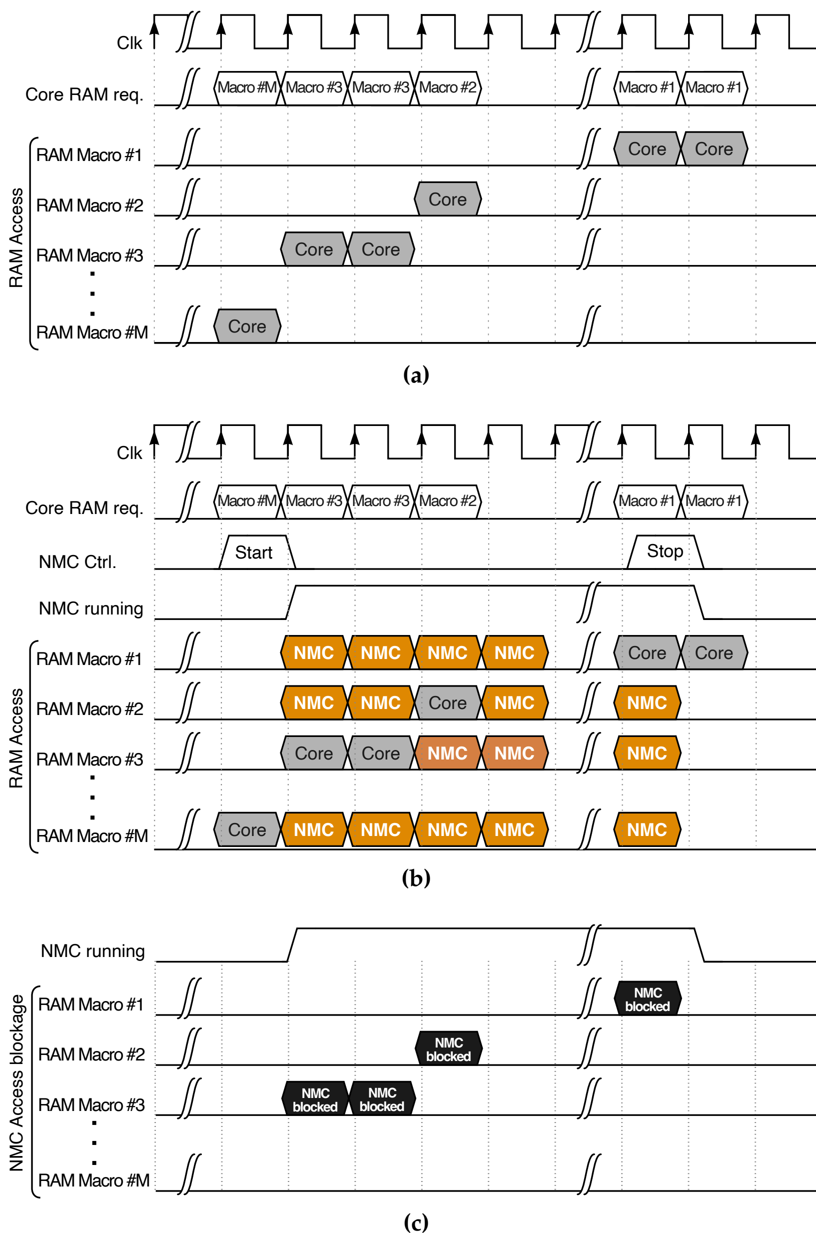


Fig. 4: (a) RAM access in a RISC-V system, (b) RAM access in a system with concurrent memory requests from the NMC unit and RISC-V, and (c) blocked RAM access to an NMC unit due to incoming request from RISC-V.

Equations (1) and (2) show that each NMC unit has #*Macros*$\times$ higher memory bandwidth than the RISC-V. With eight macros per bank, each NMC unit can achieve up to 8$\times$ the memory bandwidth of the RISC-V core per bank, resulting in a total of 32$\times$ higher bandwidth across 4 banks considering the RISC-V core employs single-I/O access. While the RISC-V core operates with a fixed memory bandwidth of one I/O per clock cycle per bank, the NMC benefits from scalable bandwidth due to the configurable number of memory macros per bank.

III.B. SYNCHRONIZATION AND CACHE MANAGEMENT

In multi-core systems including peripheral units, memory contention becomes essential when accessing a shared TCM. The proposed design incorporates a multi-level contention management scheme to address multiple memory access requests. At the global level, a round-robin arbiter fairly schedules access across different memory banks, allocating requests in a queue, and granting a turn in successive cycles. This arbitration mechanism inherently ensures fairness and prevents starvation by avoiding monopolization of the memory resources. In addition to round-robin arbitration, the NMC architecture includes a configurable control mechanism that alternates access between the RISC-V core and NMC units. Memory access requests are governed by programmable intervals, allowing dynamic adaptation to varying workload patterns. However, under high-throughput workloads, multiple requesters may target the same memory macro simultaneously. In such cases, relying solely on the previously mentioned arbitration mechanisms may cause delays on critical operations. Therefore, the NMC architecture introduces a fine-grained conflict resolution mechanism by suppressing requests from NMC units when a RISC-V access is requested simultaneously to the same memory macro, granting precedence to RISC-V. Higher memory bandwidth of NMC enables a selective prioritization that maintains arbitration fairness at the system level, but adds adaptive responsiveness for time-sensitive requests. Additionally, an interleaved addressing scheme distributes consecutive addresses across memory banks, reducing contention by up to 75 % [39].

At the software level, memory throughput is improved through cache-aware resource allocation. Although shared memory is implemented as a software-managed scratchpad rather than a hardware-controlled cache, principles of utility-based partitioning still apply. In parallel processing, memory regions are explicitly assigned to the application that benefits most, rather than the one with the highest access rate [40]. The partitioning strategy divides the shared SPM into two distinct sections at compile time, one allocated to the RISC-V core and the other to the NMC architecture, effectively managing the memory space and ensuring concurrent execution

without data collision. Partitions are determined at compile time based on application-specific memory requirements, ensuring deterministic memory behavior while preventing access conflicts and avoiding the need for complex and costly runtime coherence or locking mechanisms. For CNN workloads, where large datasets are frequently accessed across multiple memory locations, assigning a larger partition to the NMC unit enhances performance by reducing memory access latency and data movement overhead. Partitions are static during execution to preserve determinism, and reconfigurable between runs, allowing the system to adapt to different workloads using the general-purpose programmability of the RISC-V core.

Moreover, to coordinate task-level synchronization, the NMC units raise an interrupt to the RISC-V core upon completion or termination of a compute task. The dedicated interrupt line informs the RISC-V to resume post-processing, allowing low-latency coordination between producer and consumer tasks.

III.C. NMC UNIT EXTERNAL COMMUNICATION

The proposed NMC technique optimizes data movement (to and from memory), while enhancing efficient utilization of the data lifecycle. Nevertheless, NMC units need to communicate with other system modules to ensure synchronization and coherence. This external communication is facilitated through the TCM protocol, which provides single-cycle, low-latency access. At the system level, NMC units function as the primary unit, issuing memory access requests as needed.

The NMC units are controlled via memory-mapped registers defined on the TCM interconnect, allowing each individual unit to be configured independently and to send or receive data using a specific set of virtual addresses. The addresses are designated within the system memory map exclusively for NMC unit operations. As an alternative to NMC unit direct single-cycle access to memories, other possibilities consider establishing communication with the RISC-V through direct memory access (DMA), or advanced high-performance bus (AHB).

III.D. NMC UNIT OPERATION CONTROLLER

The single-cycle access to NMC units through memory-mapped registers on the TCM interconnect protocol enables instantaneous control via software. Control signals are generated using inline functions in C/C++ source code, allowing independent configuration of each NMC unit. Architectural modularity allows concurrent execution of multiple workloads, with each NMC unit configured for a distinct task of the same CNN execution, or independent

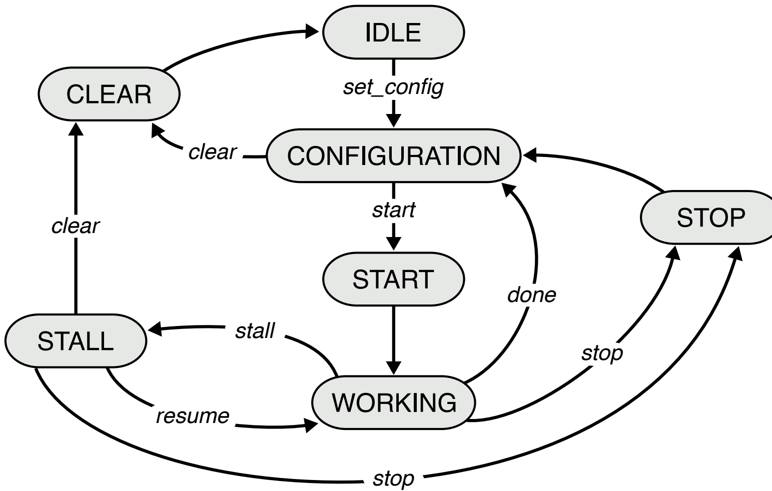


Fig. 5: NMC unit state machine including operational states with control signals.

neural networks. The modular structure supports scalable parallelism and model-level isolation, enabling adaptation to diverse application demands. Furthermore, the NMC control mechanism enables software-configurable and CPU-transparent CNN execution, combining compile-time control of tiling, loop scheduling, and memory partitioning with runtime coordination via interrupt-driven synchronization and shared memory access.

Depending on the operation type, control is established using either a one-hot or multiple-hot encoding scheme. Through the multiple-hot encoding scheme, one (or multiple) NMC unit enters one of the following states, as illustrated in the state machine diagram in Fig. 5:

- *Configuration*: each NMC unit is configured using hot encoding scheme through the set_config function. Furthermore, for validation purposes, the configuration register within each NMC unit is individually accessible by a get_config function.
- *Start*: the NMC unit initiates an operation upon receiving the signal from the *nmc_start* function. Each NMC unit starts either simultaneously or sequentially, based on the assigned priority and order of operations.
- *Stop*: the NMC unit halts an operation upon receiving the *nmc_stop* signal from software programming to terminate its computation cycle.
- *Clear*: the NMC unit is reset using the *nmc_clear* function during $Configuration/Stall$ states, allowing to reprogram as needed.

- *Stall*: the NMC unit is paused or resumed externally using *nmc_stall* and *nmc_resume* functions.
- *Working*: each NMC unit is in operation mode according to the configuration, with *nmc_done* signal to indicate the end of processing.

Furthermore, the NMC architecture incorporates a control mechanism that regulates the access to the shared memory, alternating between two distinct periods: *access_allowed* and *access_not_allowed*. During the *set_config* phase, an NMC unit is configured with its access rate, along with the parameters for a designated CNN operation. The configurable memory access allows the NMC unit's operational period to be dynamically adjusted, serving as a tuning parameter to manage priorities, particularly when the RISC-V core is handling high-priority and memory-intensive tasks. The memory control mechanism ensures flexibility and fair access distribution, optimizing resource allocation based on real-time workload demands. In summary, the NMC unit includes the following memory control states:

- *NMC_memory_access_allowed*
- *NMC_blocked_by_core_access*
- *NMC_blocked_by_configuration*

III.E. ACCELERATOR MODULES

Our proposed NMC architecture consists of the following modules for CNN acceleration:

- Repurposed RAM macros for input feature map (IFMap) and output feature map (OFMap) data
- NMC-controlled register file
- Address generator
- Access controller
- PEs

The CNN acceleration modules, enhanced memory bandwidth, and flexibility of the proposed memory structure, result in an energy efficient architecture that is capable of executing CNNs of diverse characteristics. The modular nature of the NMC technique allows for a cost-effective integration of additional modules, thereby expanding the capability to support a wider variety of workloads. Details of the accelerator are provided below.

1) RAM MACROS FOR IFMAP/OFMAP

As introduced in Section III-A, the proposed memory organization employs both bank- and macro-level partitioning to optimize bandwidth, area, and parallelism. Three design factors define the number of memory macros in the breakdown structure of the NMC architecture: total memory capacity, number

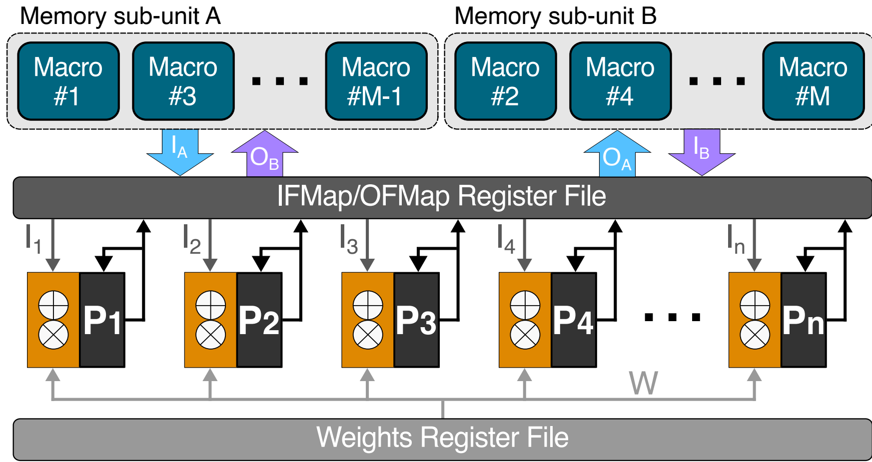
of banks, and number of PEs. Each bank is divided into two macro groups dedicated to storing IFMap and OFMap data, and managing the associated operations of CNN layers executed within the NMC unit. A ping-pong mapping scheme, also known as double-buffering, synchronizes the loading of IFMap and the reloading/storing of OFMap data, as shown in Fig. 6a. This partitioning prevents data hazards and computational stalls in case IFMap and OFMap data were stored in overlapping memory regions. Combined with the macro-level memory organization, the ping-pong mapping ensures that input activations are never overwritten by intermediate outputs, eliminating the need for additional control logic.

2) NMC-CONTROLLED REGISTER FILE

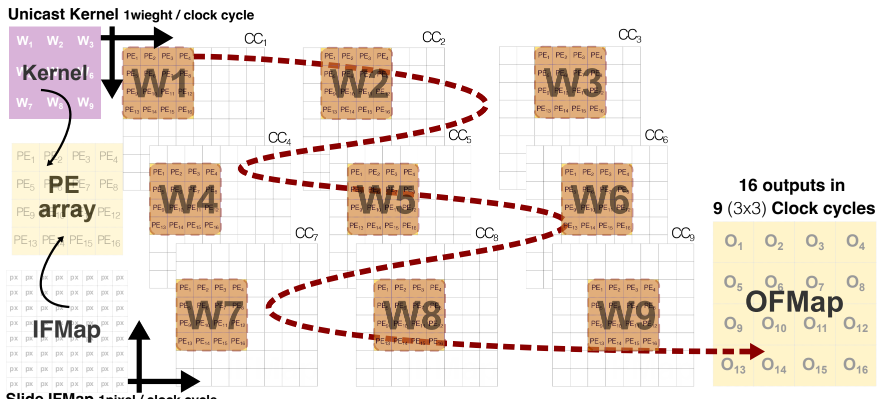
Two multi-port 1 KB register files, serving as intermediate buffers for IFMap/OFMap data and kernel weights, are employed. Given 16 PEs in each NMC unit and the allocation of memory macros for IFMap/OFMap data, the proposed setup features a 16R/16W buffer with a 1:1 aspect ratio, consisting of 16 rows, each containing 16 words of 32-bit. The intermediate buffer temporarily stores smaller chunks of data for spatial processing, reducing energy costly memory accesses. Additionally, the small-sized intermediate buffer helps to achieve high utilization of the IFMap data lifecycle. The NMC-controlled register file adds an additional level to the existing memory hierarchy, thereby enhancing computational performance. The intermediate buffer is conventionally considered as a means of addressing memory bandwidth limitations. To accommodate variations of system requirements, the relationship between the register file size, memory macros, and PEs is parameterized.

The proposed register file is equipped with a data replacement policy, with each data word containing two control bits: *Dirty bit (D)* and *Valid bit (V)*. Using control bits, similar to a hardware-controlled cache, the NMC-controlled register file tracks the lifecycle of data elements. The following scenarios based on the combination of *"DV"*, according to occurrence order, are provided:

- "00": No data is loaded into the register file.
- "01": Valid data is loaded into the pointed address in the register file but has not yet been used.
- "11": The data at the referenced address is utilized, but its lifecycle is not yet complete, as it will be used in subsequent operations.
- "10": The data in the pointed address is used, its lifecycle is over, and can be replaced with new data for the upcoming operations.



(a)



(b)

Fig. 6: (a) Illustration of output stationary dataflow for CNN processing and the employed IFMap-OFMap mapping (ping-pong structure) of memory macros within a single bank including partial sums accumulation (P), and (b) an example of sliding IFMap pixels and 3×3 kernel elements across PEs, producing 16 complete outputs in 9 clock cycles.

3) ADDRESS GENERATOR

The address generator module defines memory addresses according to the dataflow specified for CNN operations. NMC data access in memories is defined by *IFMap/OFMap element column*, *IFMap/OFMap element row*, and *input/output channel* parameters that dictate the position of an element in a CNN

layer. The NMC architecture includes hardware control parameters to specify memory addresses from which data elements are loaded, stored, or processed. The shared memory between the RISC-V and NMC units is governed by the address generator, which checks accessibility and generates the proper addresses for load/store operations in both IFMap/OFMap memory macros and intermediate buffers.

4) ACCESS CONTROLLER

The access controller manages the synchronization of the memory load/store operations for the IFMap and OFMap elements, together with the processing operations of a CNN layer. Access to memory macros and the computation process are governed by the parameters that define the CNN layer, determining whether requests for load or store operations are granted or blocked through the generation of memory access request signals. Layer operation and memory access requests are generated based on the status of the *valid bit* in the buffer of the specified access element and the current availability of memory, which is synchronized with access requests originating from the RISC-V.

5) PE ARRAY

The PEs, 16 in each NMC unit, are arranged in a grid structure. Each PE independently computes intermediate results by performing multiplication of IFMap data with kernel weights and accumulation of partial sums (psum). Two control signals govern the PEs operation: one triggers the loading of a psum from the OFMap memory macros, while the other initiates the next kernel multiplication.

The architecture flexibly supports configurable PE grid arrangements. Aligning the number of PEs per row with the number of memory macros allocated for IFMap and OFMap storage enhances computational efficiency by enabling seamless parallel data access. This alignment reduces the risk of stalls caused by overlapping compute, load, and store operations. As computational efficiency depends on the number of active PEs during various phases of layer execution, the proposed NMC architecture is optimized for both compute performance and data access efficiency throughout the data lifecycle.

III.F. DATAFLOW

The execution of a CNN layer involves a specific sequence of operations and data organization patterns across memory and compute units. A dataflow defines how input data is ordered and processed to generate outputs, sig-

nificantly influencing required memory bandwidth, data reuse, and computational efficiency. The NMC architecture features a parameterized and flexible dataflow, designed to be configurable for a wide range of CNN workloads. The dataflow for processing each CNN layer is determined by five key parameters:

- **IFMap size:** input feature map dimensions.
- **Kernel size:** weight matrix dimensions.
- **Input channel count:** number of IFMap channels.
- **Kernel count:** number of OFMap channels.
- **Stride:** convolutional traversal step size over IFMap.

Different dataflows, including unicast, multicast, broadcast, systolic array, and tree structures, can be tailored to specific application and system requirements [41]. The NMC unit, as illustrated in Fig. 6a, adopts a weight-unicast, output-stationary (OS) CNN dataflow to enhance hardware adaptability by optimizing local accumulation, reducing write-back memory traffic, and enabling configurability of the processing scheme. This dataflow exploits the lifecycle efficiency of IFMap data while leveraging convolutional data reuse within the array. The processing of single OFMap channels (SOC) and multiple OFMap-plane data (MOP), referred to as SOC-MOP OS, is considered. Each cycle, the process controller streams IFMap data of one channel into the PE array, while broadcasting a single kernel weight across all PEs to compute partial sums. An example of a 3×3 kernel operation is illustrated in Fig. 6b, showing how input data and kernel weights slide through the PEs, where 16 output values are generated over 9 cycles. Keeping intermediate results stationary within the array ensures efficient spatial reuse and reduced memory accesses, increasing the utilization of computational units. The choice of an OS dataflow provides high PE utilization via local psum accumulation and reduced IFMap data transfers.

However, the SOC-MOP OS dataflow is a trade-off for layers where weight parallelization is more dominant, like fully-connected layers in CNN, which require higher memory bandwidth to achieve high processing capabilities. Although the dataflow is fixed after synthesis, the system allows to mitigate limitations by compile-time configurability:

- layer-specific configurations, along with compile-time tuning of tiling factors, loop order, and data partitioning, to support varying CNN architectures;
- flexible memory and address mapping strategies;
- flexible co-processing with the RISC-V core for handling unsupported/irregular layers via hybrid execution.

To maintain continuous PE activity and hide memory latency, the architecture executes four concurrent operations, as shown in Fig. 7: layer execution, IFMap loading, OFMap storing, and OFMap reloading (for psum accumulation).

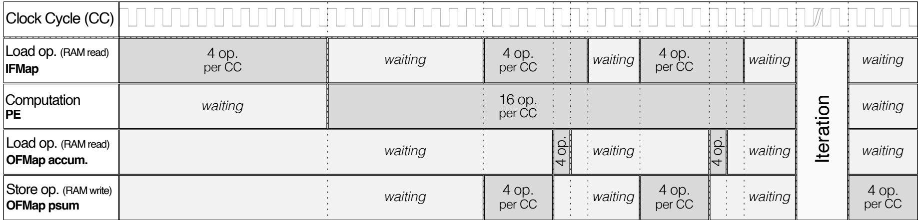


Fig. 7: Timing diagram of an NMC unit acceleration workload, defining the four simultaneous processes as part of the NMC's dataflow, and including the number of operations per clock cycle (CC).

The use of a small-sized register file, as intermediate buffer, guarantees uninterrupted PE computations and enables parallel processing. Memory access overhead is reduced by overlapping layer execution with IFMap loading. Furthermore, the CNN dataflow behavior is governed by the same set of layer parameters used in address generation, defining the spatial and channel structure of each layer and determining the pattern of concurrent data processing.

During convolution, each kernel channel accumulates partial results to generate the corresponding OFMap channels. This process is repeated for all kernels. The lifecycles of fetched IFMap data and kernel weights are efficiently utilized by storing the intermediate OFMap values in memory, and reloading them for accumulation as required by the layer's execution sequence. Accumulation is then completed by restoring OFMap data back into the PEs, ensuring that OFMap data restoration is directly proportional to the number of IFMap channels. Since the number of output channels (defined by kernel count) typically exceeds the number of input channels, the proposed dataflow reduces memory access by reusing IFMap data across kernel passes, thereby avoiding repeated input memory loads for each OFMap channel. The IFMap data is initially stored in SRAM macros and subsequently transferred to register files, in accordance with the dataflow strategy. The IFMap storage pattern ensures alignment with the processing sequence of each NMC unit, supporting consistent and predictable memory accesses throughout execution.

III.G. DESIGN MODULARITY AND FLEXIBILITY

Each memory bank accommodates one NMC unit, and the number of banks is configurable, enabling modular scalability for different architecture requirements. However, since each NMC unit embedded in a cache bank acts as a primary component on the TCM interconnect, excessive scaling may lead to increased overhead from additional data buses. Therefore, it is important to maintain a balance between scaling the number of banks and NMC units to increase parallelism, while keeping hardware costs and complexity low. Utilizing the NMC modular structure and adjusting architecture parameters such as the number of memory banks, the number of RAM macros per bank, and the number of PEs, the architecture can be flexibly configured to meet different performance and application needs.

Beyond hardware-level adjustments, our NMC architecture offers a remarkable flexibility compared to fixed-function ASICs. A wide range of CNN models are supported without requiring hardware re-synthesis, with tiling, loop ordering and memory partitioning execution parameters defined at compile time. Each NMC unit is independently configurable and operated, allowing concurrent execution of different network layers or models. In conjunction with the modular NMC architecture, the general-purpose RISC-V core provides additional programmability to manage dynamic workloads, tasks coordination, and system adaptability to application-specific requirements. The synergy between RISC-V core and NMC units enables efficient workload distribution and multi-model support within a single hardware instance.

IV. RESULTS AND DISCUSSION

This section presents an analysis of computational efficiency, data movement, performance, energy efficiency, and area. Additionally, the simultaneous operation of the NMC unit and the RISC-V is evaluated. While the design has been validated both on FPGA and through ASIC integration, the results presented in this article are based on the ASIC implementation. The design is synthesized using a standard ASIC flow using Cadence Genus in GlobalFoundries 22 nm FD-SOI technology at a nominal supply voltage of 0.8 V. Post-synthesis power consumption is estimated through simulation-based power analysis, using Value Change Dump (VCD) switching activities, and further analyzed with Synopsys PrimeTime. The power of the entire system, including the NMC units integrated within the memory subsystem of the MCU platform, is considered in the evaluation. A cycle-accurate performance evaluation is ensured by including dedicated performance counters in the

NMC units. The organization of the NMC unit allows flexibility in excluding the performance counters from hardware synthesis, or retaining them in the design for simulation purposes.

IV.A. NMC UNIT CONFIGURATION AND DATAFLOW

The proposed NMC architecture is integrated into the open-source platform PULPissimo, which is a single-core RISC-V SoC derived from the low-power PULP MCU [42]. The shared cache, composed of four repurposed memory banks of 128 KB each, defines the design space of the NMC architecture, enabling highly efficient CNN inference acceleration.

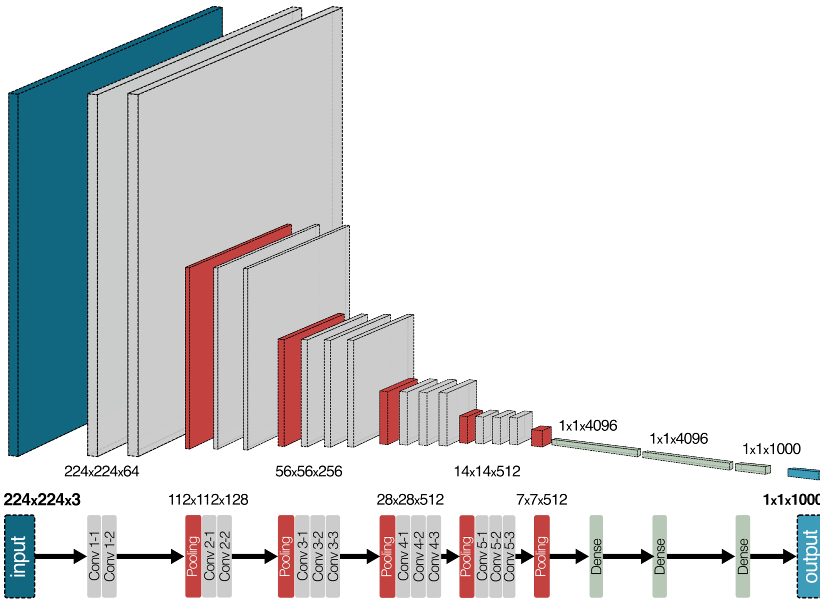


Fig. 8: VGG-16 architecture having convolutional layers of 3×3 kernel with stride 1, and max-pool layers of 2×2 with stride 2.

The proposed design is benchmarked using convolutional layers of three popular neural networks: MobileNet, ResNet, and in more detail VGG-16. VGG-16 comprises a total of 16 layers, including 13 convolutional layers and 3 fully-connected layers, as shown in Fig. 8 [43]. The resulting quantized model achieved a top-5 accuracy of 89.2 %, preserving inference fidelity to a competitive degree for edge deployment. For evaluation purposes, emphasis is placed on the workload of convolutional layers, given that fully-connected operations account for less than 1 % of the total network workload. Moreover, the scalability and flexibility of the architecture is realized by parameterizing

Table 1: Configuration of each convolutional layer of VGG-16 processed in the NMC architecture, including layers tiling.

Configuration	**Convolutional Layer**								
	1-1	**1-2**	**2-1**	**2-2**	**3-1**	**3-2 / 3**	**4-1**	**4-2 / 3**	**5-1 / 2 / 3**
IFMap size	224	224	112	112	56	56	28	28	14
#Channel	3	64	64	128	128	256	256	512	512
#Kernel	64	64	128	128	256	256	256	512	512
Kernel size	3	3	3	3	3	3	3	3	3
Tiling partition									
#IFMap tile	4	4	2	2	1	1	1	1	1
#Chan. tile	1	16	16	32	32	64	16	32	16
#Kern. tile	16	16	32	32	64	64	64	64	32
Tile size									
IFMap tile	56	56	56	56	56	56	28	28	14
#Chan. per tile	3	4	4	4	4	4	16	16	32
#Kern. per tile	4	4	4	4	4	4	8	8	16

various aspects of the design. Since the RISC-V system processes data in 32-bit units and the memory word length aligns with this size, the parameters are aggregated into a 32-bit word to achieve a single clock cycle configuration step.

The NMC units are configured for sequential processing of CNN layers. The configuration parameters, required for processing each convolutional layer of VGG-16, are presented in Table 1. Due to the large size of each layer, partial computation via tiling (segmentation) of layer parameters becomes necessary, as illustrated in Fig. 9. Each layer adopts a specific tiling pattern based on configurations optimized for NMC acceleration. The relationships between tiling parameters and resulting tile sizes are exemplified in Equations 3, 4, and 5 for layer 1-1, with further details provided in Table 1.

$$IFMap_{tile} = \frac{IFMap_{size}}{\#IFMap_{tile}} = \frac{224}{4} = 56 \tag{3}$$

$$\#Channel_{per\ tile} = \frac{\#Channel}{\#Channel_{tile}} = \frac{3}{1} = 3 \tag{4}$$

$$\#Kernel_{per\ tile} = \frac{\#Kernel}{\#Kernel_{tile}} = \frac{64}{16} = 4 \tag{5}$$

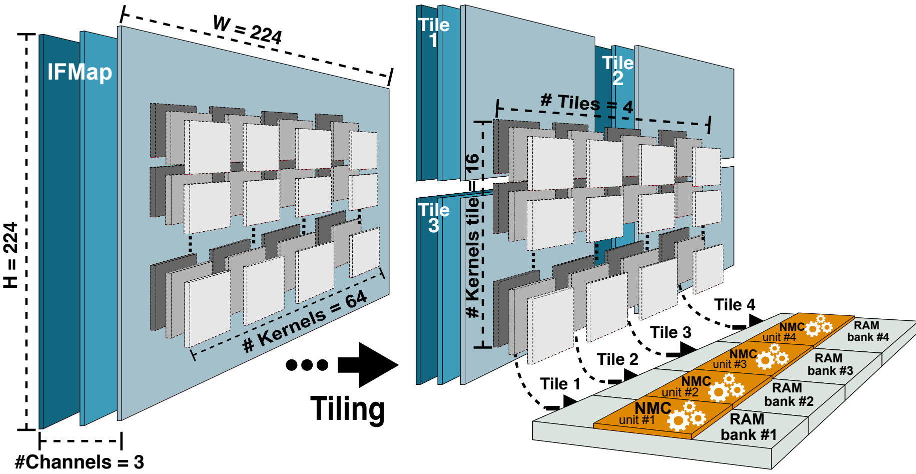


Fig. 9: Tiling illustration for the distribution of a convolutional layer into NMC units. IFMap and kernels are partitioned in tiles, generating processing batches that can fit in the memory subsystem of the design.

In the VGG-16 analysis, a bank-tiling strategy is applied, where each memory bank holds the data of a single tile. Based on layer characteristics, either IFMap or kernel data partition is assigned per bank. This organization enables efficient parallel processing across banks and complements the efficiency of NMC data access.

IV.B. COMPUTATIONAL EFFICIENCY

This section evaluates computational efficiency of NMC, with emphasis on in-core performance depending on on-chip data movement, memory bandwidth, and operational intensity. In this context, computational efficiency is defined as PE activity, representing the extent to which each PE operates at its peak capacity. Assuming each PE performs at most one MAC per cycle, computational efficiency is measured as the average number of MAC operations executed per cycle per PE. In this study, an NMC unit includes 16 PEs arranged in a grid of 4×4, capable of executing up to 16 operations per clock cycle. The first convolutional layer of the three aforementioned CNN models is used to benchmark computational efficiency, as shown in Fig. 10. Inputs of size 224×224 pixels with three channels are evaluated across four kernel configurations: 1, 16, 32, and 64. The measured data is presented alongside a trendline, smoothed using a moving average filter to mitigate noise. Larger IFMaps demand greater parallelism, thereby increasing computational efficiency.

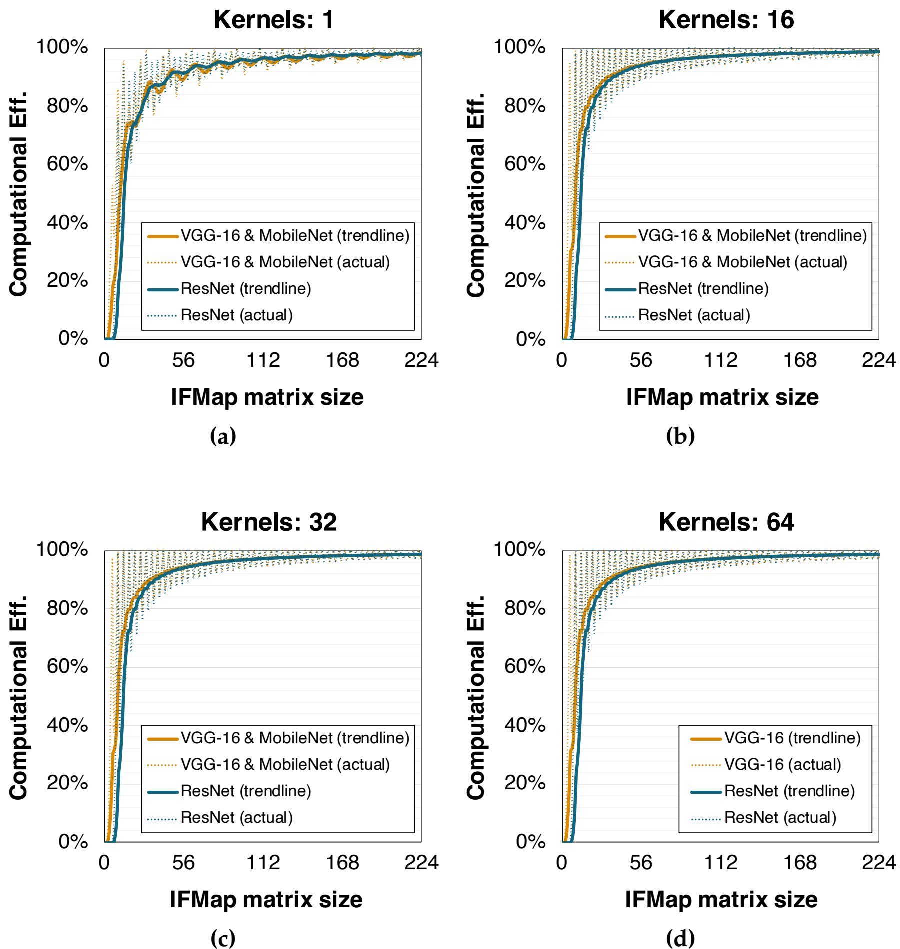


Fig. 10: Computational efficiency for the first convolutional layer of VGG-16, MobileNet and ResNet processing (a) 1 kernel, (b) 16 kernels, (c) 32 kernels, and (d) 64 kernels.

To assess performance on deeper layers, we evaluate the configurations listed in Table 1 for the VGG-16 model. As shown in Fig. 11, the NMC architecture achieves a computational efficiency of 96 % when processing layers CONV1 to CONV4. Efficiency drops to 73 % for CONV5 layers due to the smaller tiling sizes within the employed dataflow, as larger IFMap tiles yield higher efficiency compared to smaller ones. Overall, the architecture achieves

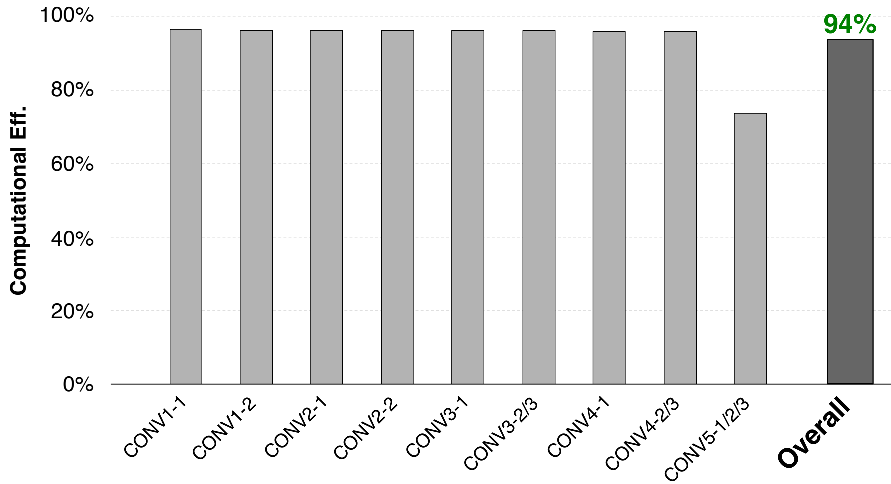


Fig. 11: Computational efficiency for processing VGG-16 convolutional layers, calculated based on the utilization of PEs for each individual layer.

an average computational efficiency of 94 % across the entire network, which is comparable to state-of-the-art (SoA) implementations.

IV.C. MEMORY ACCESSES

System efficiency, considering unavoidable energy-expensive memory accesses, is evaluated as the ratio of memory accesses to the amount of data processed during runtime. In this study, a memory access refers to on-chip memory read and write operations during CNN processing. Fig. 12 illustrates the relationship between the number of parameters in each convolutional layer, and the required memory accesses for both the NMC units and the RISC-V running the same workload. The use of the proposed NMC architecture effectively reduces on-chip memory accesses by a factor of 3.6×, compared to the baseline scenario, where the CNN workload is managed solely by the RISC-V core.

IV.D. PARALLEL PROCESSING: NMC VS RISC-V

In multi-core systems with a shared memory, balanced and fair access is essential for maintaining consistent performance across all cores. To evaluate parallel processing, shared scratchpad memory access is enforced for both the RISC-V and NMC units, enabling direct assessment of system's behavior under simultaneous access conditions. In this scenario, the CONV1-1 layer of

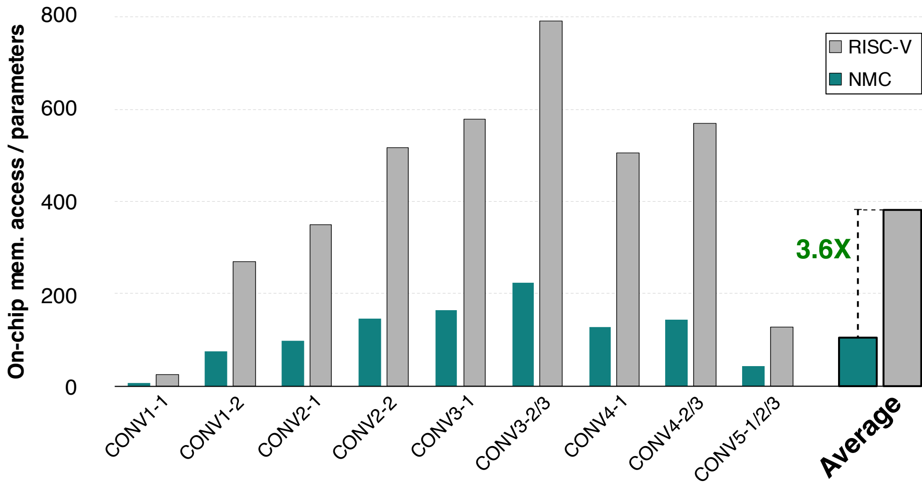


Fig. 12: Comparison of memory accesses between NMC architecture and RISC-V core processing VGG-16 convolutional layers.

VGG-16 is mapped to the NMC units, while the final dense layer is assigned to the RISC-V core. The system supports two operational modes:

- Sequential processing: the RISC-V performs pre/post processing tasks, while the NMC units handle the execution of an intensive CNN workload.
- Parallel processing: both existing co-processing units execute tasks concurrently.

In parallel operational mode, non-overlapping memory regions are explicitly allocated to avoid data hazards and the need for dynamic coherence mechanisms. Task-level dependencies are resolved through a producer-consumer execution model, coordinated via a dedicated interrupt line raised by the NMC units to the RISC-V core upon task completion or termination. This mechanism enables efficient resumption of post-processing without the overhead of polling or unnecessary stalls.

The analysis of both operational modes is illustrated in Fig. 13a, showing that the RISC-V processor, despite handling only approximately 20 % of the total workload, remains active for a longer duration in sequential mode. In contrast, the NMC units are responsible for 80 % of the workload, and require less of the total active cycles. By enabling simultaneous memory access without contention, the parallel processing mode achieves a 38 % reduction in total active cycles compared to sequential execution, improving the performance of the multi-core system.

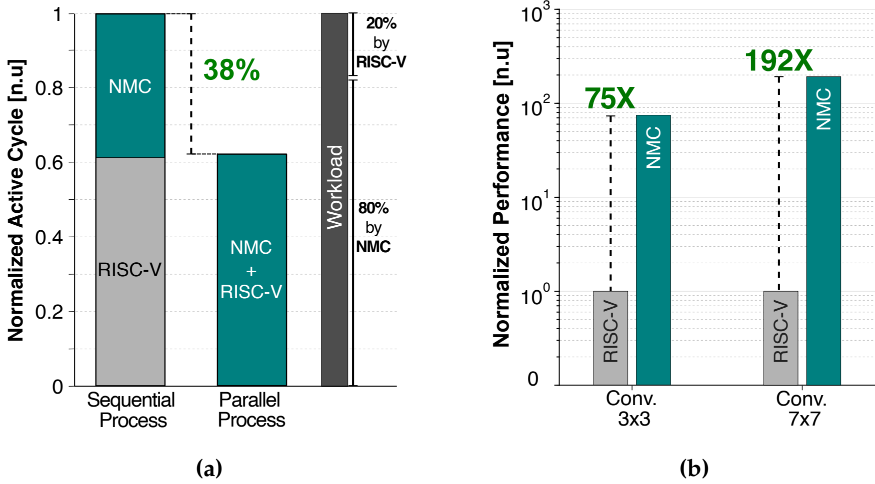


Fig. 13: (a) Comparison of sequential and parallel processing using the RISC-V core and NMC units, (b) Performance comparison between the RISC-V and NMC units for different convolutional layer kernel sizes.

The improvement in parallel processing is enabled by the memory partitioning strategy, which forms a modular NMC unit per bank architecture. The effective memory bandwidth for hardware acceleration is significantly increased by providing CPU-transparent access to distinct memory regions. In addition to memory-level optimizations, the proposed dataflow enhances data reuse and reduces intermediate storage overhead through a computation-aware execution strategy, resulting in superior computational performance of the NMC units compared to the RISC-V core. The performance gap between NMC units and RISC-V, when executing convolutional layers with varying kernel sizes, is illustrated in Fig. 13b.

IV.E. PERFORMANCE AND ENERGY EFFICIENCY

Processing at the edge can be realized on various platforms, with the choice of platform being crucial for balancing performance and energy efficiency. Fig. 14 illustrates the trade-offs in performance and energy efficiency among SoA CNN edge solutions across different platforms, including operating frequency ranges. To ensure consistency, all performance metrics are normalized to 8-bit precision by scaling with respect to operand word length. For example, the performance of a 16-bit design is scaled by a factor of 2, consistent with the reported results in [48], under the assumption that

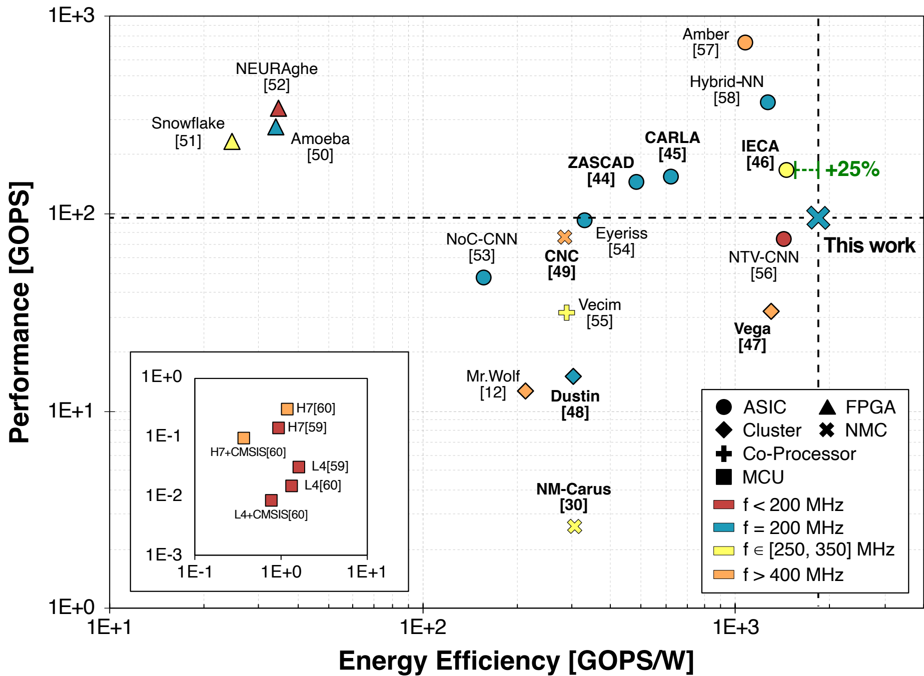


Fig. 14: Performance and energy efficiencies comparison of various SoA works, considering different platforms with results normalized to 8-bit, including our NMC architecture.

its datapath can be adapted to execute two parallel 8-bit operations. As shown in Fig. 14, ASICs and FPGAs achieve a higher performance with a larger number of PEs offering faster computational capabilities. However, ASIC-based accelerators are tailored to a specific application, and thus, lack flexibility. FPGA-based solutions are configurable, but at the lower end of energy efficiency. Moreover, MCU-based designs, positioned in the lower-left region, exhibit reduced performance and energy efficiency, driving various studies into optimization techniques for this platform. SoA co-processors and clusters typically employ multiple RISC-V cores to handle computation-intensive workloads. The proposed NMC architecture integrates specialized PEs and a dedicated control unit directly within the shared memory, enhancing both efficiency and performance.

The hardware analysis of the NMC architecture, alongside SoA works, is summarized in Table 2. Performance is expressed in giga operations per second (GOPS), where each MAC accounts for two operations, i.e., one multiplication and one addition. Performance and energy efficiency exclude

Table 2: Comparison of the proposed NMC architecture with other edge computing implementations.

Source	ZASCAD [44]	CARLA [45]	IECA [46]	Vega [47]	Dustin [48]	CNC [49]	NM-Carus [30]	This work
Solution	Accel.	Accel.	Accel.	Cluster	Cluster	NMC	NMC	NMC
Technology	65 nm	65 nm	55 nm	22 nm	65 nm	Intel 4	65 nm	22 nm
Area [mm ^{2}]	6	6.2	2.75	12	10	1.92	0.42	1.45
Precision	16-bit	16-bit	16-bit	8 to 32-bit	2 to 32-bit	8-bit	8 to 32-bit	8 to 32-bit
PEs	192	196	168	27 + 10×RV	16×RV	128 + 8×RV	4	64 (128)
Memory [KB]	36.9	85.5	109	1728	208	512	32	512
Clock [MHz]	200	200	250	450	205	1150	330	200
Performance * [GOPS]	145	154.8	168 †	32.2 †	15 †	75.8	2.6 †	**96** (192)
Energy Eff. * [GOPS/W]	481.8	626.8	1466 †	1300 †	303 †	285	306.7 †	**1828**
Area Eff. * [GOPS/mm^2/MB]	6.18 ‡	14.8 ‡	30.55 ‡	4.53	2.16 ‡	5.53 ‡	1.37 ‡	**33.1**
Comp. Eff. [%] (Network)	94 (VGG-16)	98 (VGG-16)	87 § (VGG-16)	66 § (repVGG)	65 § (ResNet8)	26 (MANN)	79 § (Conv2D)	**94** (VGG-16)

* Normalized as function of 8-bit precision. † Peak theoretical. ‡ Technology scaling according to [63]. § Projected from reported results.

expenses of external data accesses to DRAM. The proposed architecture achieves the highest energy efficiency among all evaluated works, specifically 25 % higher than IECA [46], 28 % higher than Vega [47], and up to 84 % higher than other designs. PEs and data access mechanisms consume the largest amount of power during computation, and architectures with larger PE arrays tend to incur higher energy costs. The high energy efficiency of this design is primarily attributed to the high computational efficiency of PEs and reduction of redundant accesses through an optimized dataflow, with additional contribution from the underlying implementation technology. The flexibility of the proposed NMC architecture enables configuration tailored to a specific application. Modularity implies that performance is adjustable by the number of PEs, precision, and memory capacity, as illustrated in Fig. 15. Thus, the architecture has the advantage of being as flexible as an MCU platform, while providing a performance achieved with rigidly custom designed accelerators. IECA [46], ZASCAD [44], and CARLA [45] report higher raw throughput in their respective setups. However, performance projections for a 128-PE configuration of our architecture indicate superior performance compared to the competitive designs analyzed in the study. This advantage is primarily enabled by efficient scalability of the architecture and enhanced parallelism.

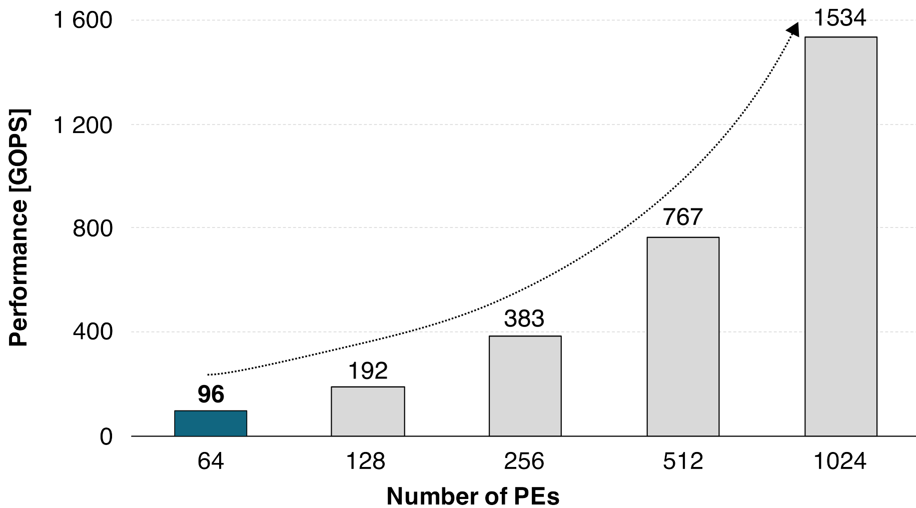


Fig. 15: Architecture scalability relation to performance at 200 MHz, according to the number of PEs.

In terms of computational efficiency, the proposed NMC architecture outperforms all evaluated works, with only ZASCAD [44] and CARLA [45], achieving comparable level of efficiency. Multi-core RISC-V cluster-based designs, such as Vega [47] and Dustin [48], offer flexibility through low-power general-purpose cores. However, their lower computational efficiency for CNN workloads results in reduced performance and energy efficiency compared to the proposed design. A clock frequency of 200 MHz is selected to ensure a fair comparison with the evaluated SoA techniques. At this frequency, the proposed MCU-based NMC architecture achieves 1828 GOPS/W energy efficiency and a high computational efficiency of 94 %, enabling enhanced processing capabilities for various CNN models. Notably, compared to the evaluated NMC designs, i.e., NM-Carus [30] and CNC [49] in Table 2, the proposed NMC architecture demonstrates superior performance and efficiency for its nature of integrating tailored PEs within the memory subsystem of an MCU platform.

IV.F. AREA

The 512 KB shared cache memory occupies a significant portion of the SoC area. Fig. 16 presents the synthesis-based area distribution for a single memory bank within the cache memory subsystem, including the area breakdown of an NMC unit logic. Each bank consists of eight memory macros and an NMC unit, with SRAM cells occupying 69 % of the area, while the remaining

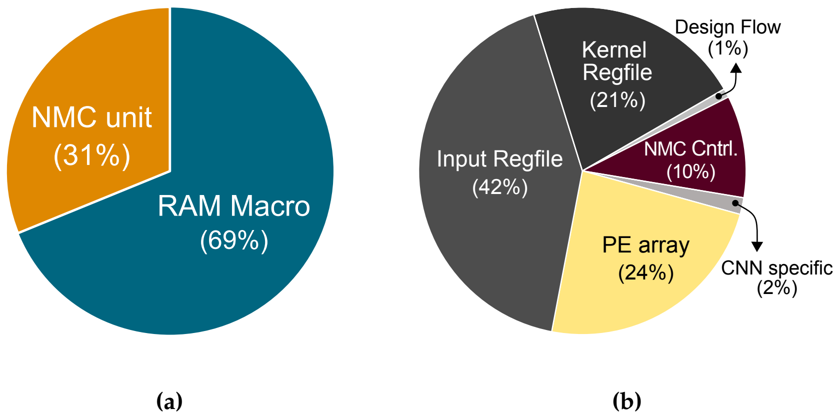


Fig. 16: (a) Area decomposition of a single memory bank consisting of memory macros and an NMC unit, (b) area decomposition of NMC components.

31 % is allocated to NMC components, as shown in Fig. 16a. Overall, the NMC unit integrated into the cache memory subsystem accounts for one-third of the total SoC area. Additionally, Fig. 16b illustrates the internal area distribution of the NMC unit, including PE array and intermediate register files (RegFile). Gate-level synthesis results reveal that the MCU platform with the integrated NMC architecture consists of 7.2 million gate equivalents (GE), measured in minimum-sized two-input NAND gates, with 5.3 million GE dedicated to the RISC-V core, private and shared memories, and SoC's peripherals. As shown in Table 2, the area efficiency, measured in GOPS/mm^2/MB to account for memory size, is 33.1 for the proposed design. This represents an 8% improvement over IECA [46], a 2.2× gain over CARLA [45], and between 5.3× and 24× higher efficiency compared to the remaining designs, which accounts for an efficient memory utilization in our design. Fig. 17 illustrates a layout view for the physical placement of a single NMC unit and its associated logic blocks in relation to the SRAM macros, reflecting the macro-level partitioning strategy. The complete system, comprising 32 macros and four NMC units across four memory banks, replicates the representation of the presented single NMC unit. According to floorplan constraints, the system can be adapted to different physical placement scenarios.

IV.G. RESEARCH CHALLENGES AND FUTURE WORK

While the proposed NMC architecture demonstrates high energy efficiency and scalability for CNN workloads, some constraints arise from its current

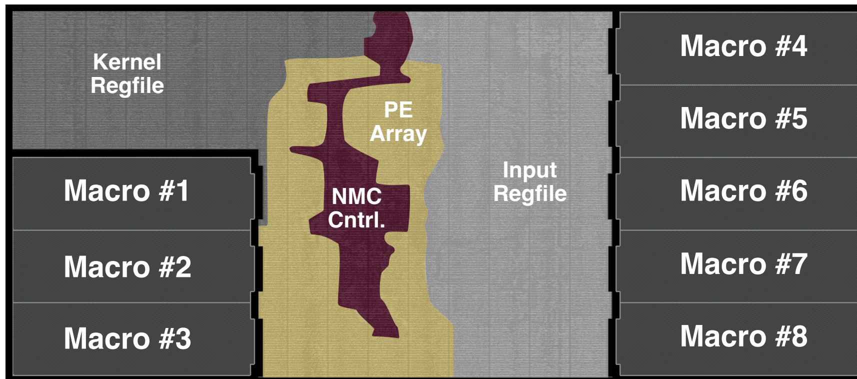


Fig. 17: NMC unit layout including memory macros and hardware acceleration components.

optimization focus. The design targets dense computation patterns, making it well-suited for many edge AI tasks. Although sparse CNNs, GCNs, and irregular models, e.g., RNNs or Transformers, can be executed using dense representations, the architecture requires hardware level optimizations to exploit sparsity, including zero-skipping or sparse data decoding mechanisms, which are promising directions for future work. The use of conventional SRAM simplifies integration with a standard CMOS design flow and ensures predictable behavior, but adapting to emerging memory technologies may require changes to accommodate differing latency, endurance, or control requirements. While the architecture supports modular scaling, increasing the number of NMC units introduces interconnect and arbitration complexity, requiring careful balancing in larger systems. Overall, the modular and software-configurable design provides a solid foundation for extending support to broader workloads and memory platforms in future iterations.

V. CONCLUSION

This study presents an efficient near-memory computing hardware acceleration architecture for processing NNs (specifically CNN) on edge devices. The NMC architecture is integrated as a co-processor in the shared cache system of an MCU, by interleaving the memory with computation logic in a non-intrusive manner, while employing conventional SRAM. The proposed design demonstrates scalability across diverse CNN applications through a fully parameterized architecture and platform-agnostic integration, enabling computation to be efficiently pushed closer to the data source. The study

is benchmarked with popular CNN models and compared to state-of-the-art solutions. The design evaluation is capable of delivering 96 GOPS and 1828 GOPS/W for 8-bit precision at a clock frequency of 200 MHz. An efficient hardware solution is demonstrated by achieving a computational efficiency of 94 % during simultaneous operation with a RISC-V core. The modular design and software-configurable execution further distinguish the architecture from ASICs tailored to a specific application, enabling broader adaptability across diverse edge AI workloads. While the current implementation is optimized for dense CNNs, the architecture can functionally evaluate sparse and irregular models using dense representations. However, realizing the efficiency benefits of sparsity would require additional hardware support, which is considered for future extensions of our work.

ACKNOWLEDGMENT

The authors are grateful to Vetenskapsrådet, and to the SSF financed classIC Research Center for financial support.

REFERENCES

[1] Véstias, M.P., "A Survey of Convolutional Neural Networks on Edge with Reconfigurable Computing," *Algorithms*, vol. 12, no. 8, 2019, Art. no. 154.

[2] W. Shi et al., "Edge Computing: Vision and Challenges," *IEEE Internet of Things J.*, vol. 3, no. 5, pp. 637-646, Oct. 2016.

[3] S. Huang et al., "How Flexible is Your Computing System?," *ACM Trans. Embed. Comput. Syst.*, vol. 21, no. 4, Jul. 2022, Art. no. 37.

[4] A. Reuther et al., "Survey and Benchmarking of Machine Learning Accelerators," *IEEE High Perform. Extreme Comput. Conf. (HPEC)*, 2019, pp. 1-9.

[5] A. Capotondi et al., "CMix-NN: Mixed Low-Precision CNN Library for Memory-Constrained Edge Devices," *IEEE Trans. Circuits Syst. II: Express Br.*, vol. 67, no. 5, pp. 871-875, May. 2020.

[6] T. -J. Yang et al., "A method to estimate the energy consumption of deep neural networks," *51st Asilomar Conf. Signals, Syst., Comput.*, 2017, pp. 1916-1920.

[7] Y. -H. Chen et al., "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," *ACM/IEEE 43rd Annual Int. Symp. Comput. Archit. (ISCA)*, 2016, pp. 367-379.

[8] A. A. Khan et al., "The Landscape of Compute-near-memory and Compute-in-memory: A Research and Commercial Overview," 2024, *arXiv:2401.14428v1 [cs.AR]*.

[9] Y. Ju et al., "A General-Purpose Compute-in-Memory Processor Combining CPU and Deep Learning with Elevated CPU Efficiency and Enhanced Data Locality," *IEEE Symp. VLSI Technol. Circuits*, 2023, pp. 1-2.

[10] Y. Ju and J. Gu, "A 65nm Systolic Neural CPU Processor for Combined Deep Learning and General-Purpose Computing with 95% PE Utilization, High Data Locality and Enhanced End-to-End Performance," *IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2022, pp. 1-3.

[11] M. Cavalcante et al., "Ara: A 1-GHz+ scalable and energy-efficient RISC-V vector processor with multiprecision floating-point support in 22-nm FD-SOI," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 2, pp. 530–543, Feb. 2020.

[12] A. Pullini et al., "Mr.Wolf: An Energy-Precision Scalable Parallel Ultra Low Power SoC for IoT Edge Processing," *IEEE J. Solid-State Circuits*, vol. 54, no. 7, pp. 1970-1981, Jul. 2019.

[13] S. S. Sahoo et al., "On-chip Memory in Accelerator-based Systems: A System Technology Co-Optimization (STCO) Perspective for Emerging Device Technologies," *IEEE 37th Int. Syst.-on-Chip Conf. (SOCC)*, 2024, pp. 1-6.

[14] F. Conti et al., "A 12.4TOPS/W @ 136GOPS AI-IoT system-on-chip with 16 RISC-V, 2-to-8b precision-scalable DNN acceleration and 30%-boost adaptive body biasing," *IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2023, pp. 21-23.

[15] J. Li et al., "An FPGA-Based Energy-Efficient Reconfigurable Convolutional Neural Network Accelerator for Object Recognition Applications," *IEEE Trans. Circuits Syst. II: Express Br.*, vol. 68, no. 9, pp. 3143-3147, Sep. 2021.

[16] P. D. Schiavone et al., "Arnold: An eFPGA-Augmented RISC-V SoC for Flexible and Low-Power IoT End Nodes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 29, no. 4, pp. 677-690, Apr. 2021.

[17] F. Renzini et al., "A fully programmable eFPGA-augmented SoC for smart power applications," *IEEE Trans. Circuits Syst. I: Regul. Pap*, vol. 67, no. 2, pp. 489-501, Feb. 2020.

[18] G. Li et al., "A Digital SRAM-Based Computing-in-Memory Macro Supporting Parallel Maintaining for Network Management," *IEEE Solid-State Circuits Lett. (SSCL)*, vol. 7, pp. 327-330, Oct. 2024.

[19] Y. Li et al., "A Survey of MRAM-Centric Computing: From Near Memory to In Memory," *IEEE Trans. Emerg. Topics Comput.*, vol. 11, no. 2, pp. 318-330, Apr. 2023.

[20] R. Medina et al., "Bank on Compute-Near-Memory: Design Space Exploration of Processing-Near-Bank Architectures," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 43, no. 11, pp. 4117-4129, Nov. 2024.

[21] L. Ke et al., "Near-Memory Processing in Action: Accelerating Personalized Recommendation With AxDIMM," *IEEE Micro*, vol. 42, no. 1, pp. 116-127, Jan. 2022.

[22] S. Srinivasa et al., "Trends and Opportunities for SRAM Based In-Memory and Near-Memory Computation," *22nd Int. Symp. Qual. Electron. Des. (ISQED)*, 2021, pp. 547-552.

[23] J. Zhang et al., "In-Memory Computation of a Machine-Learning Classifier in a Standard 6T SRAM Array," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 915-924, Apr. 2017.

[24] X. Si et al., "A 28 nm 16-kb Sign-Extension-Less Digital-Compute-in-Memory Macro With Extension-Friendly Compute Units and Accuracy-Adjustable Adder-Tree," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* vol. 32, no. 11, pp. 2164–2168, Nov. 2024.

[25] J. Sun et al., "Analog or Digital In-memory Computing? Benchmarking through Quantitative Modeling," *IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2023, pp. 1-9.

[26] K. -T. Tang et al., "Considerations of Integrating Computing-In-Memory and Processing-In-Sensor into Convolutional Neural Network Accelerators for Low-Power Edge Devices," *Symp. VLSI Technol.*, 2019, pp. T166-T167.

[27] Y. -C. Wu et al., "DEA-NIMC: Dynamic Energy-Aware Policy for Near/In-Memory Computing Hybrid Architecture," *IEEE 36th Int. Syst.-on-Chip Conf. (SOCC)*, 2023, pp. 1-6.

[28] M. Nouripayam et al., "An Energy-Efficient Near-Memory Computing Architecture for CNN Inference at Cache Level," *28th IEEE Int. Conf. on Electron. Circuits Syst. (ICECS)*, 2021, pp. 1-4.

[29] Q. Si and B. C. Schafer, "Optimizing Behavioral Near On-Chip Memory Computing Systems," *IEEE 33rd Int. Conf. Appl.-specific Syst., Arch. Processors (ASAP)*, 2022, pp. 27-33.

[30] M. Caon et al., "Scalable and RISC-V Programmable Near-Memory Computing Architectures for Edge Nodes," *IEEE Trans. Emerg. Topics Comput.*, pp. 1-15, Apr. 2025.

[31] I. Puaut and C. Pais, "Scratchpad memories vs locked caches in hard real-time systems: a quantitative comparison," *Des. Automat. Test Eur. Conf. Exhib.*, 2007, pp. 1-6.

[32] A. Ben Abdallah, "Computational Frameworks: 3 - Heterogeneous Computing: An Emerging Paradigm of Embedded Systems Design," *Comput. Frameworks: Syst., Models Appl.*, pp. 61-93, Dec. 2017.

[33] J. Cong et al., "An energy-efficient adaptive hybrid cache," *IEEE/ACM Int. Symp. on Low Power Electron. Des. (ISPLED)*, 2011, pp. 67-72.

[34] M. J. Lyons et al., "The accelerator store: A shared memory framework for accelerator-based systems," *ACM Trans. Archit. Code Optim.*, vol. 8, no. 4, pp. 1–22, Jan. 2012.

[35] M. Dehyadegari et al., "A tightly-coupled multi-core cluster with shared-memory HW accelerators," *Int. Conf. Embedded Comput. Syst. (SAMOS)*, 2012, pp. 96-103.

[36] T. L. Fischer et al., "Analysis of Shared Cache Interference in Multi-Core Systems using Event-Arrival Curves," *Int. Conf. Real-Time Netw. Syst. (RTNS)*, 2023, pp. 22-23.

[37] J. Ax et al., "CoreVA-MPSoC: A Many-Core Architecture with Tightly Coupled Shared and Local Data Memories," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 5, pp. 1030-1043, May. 2018.

[38] L. Benini et al., "P2012: Building an ecosystem for a scalable, modular and high-efficiency embedded computing accelerator," *Des. Autom. Test Eur. Conf. Exhib.*, 2012, pp. 983-987.

[39] R. Wittig et al., "Statistical Access Interval Prediction for Tightly Coupled Memory Systems," *IEEE Symp. Low-Power High-Speed Chips (COOL CHIPS)*, 2019, pp. 1-3.

[40] M. K. Qureshi and Y. N. Patt, "Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches," *39th Annual IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, 2006, pp. 423-432.

[41] H. Kwon et al., "MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects," *23rd Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, 2018, pp. 461–475.

[42] P. D. Schiavone et al., "Quentin: an Ultra-Low-Power PULPissimo SoC in 22nm FDX," *IEEE SOI-3D-Subthreshold Microelectron. Technol. Unified Conf. (S3S)*, 2018, pp. 1-3.

[43] K. Simonyan et al., "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014, *arXiv:1409.1556v6 [cs.CV]*.

[44] A. Ardakani et al., "Fast and Efficient Convolutional Accelerator for Edge Computing," *IEEE Trans. Comput.*, vol. 69, no. 1, pp. 138-152, Jan. 2020.

[45] M. Ahmadi et al., "CARLA: A Convolution Accelerator With a Reconfigurable and Low-Energy Architecture," *IEEE Trans. Circuits Syst. I: Regul. Pap.*, vol. 68, no. 8, pp. 3184-3196, Aug. 2021.

[46] B. Huang et al., "IECA: An In-Execution Configuration CNN Accelerator With 30.55 GOPS/mm^2 Area Efficiency," *IEEE Trans. Circuits Syst. I: Regul. Pap.*, vol. 68, no. 11, pp. 4672-4685, Nov. 2021.

[47] D. Rossi et al., "Vega: A Ten-Core SoC for IoT Endnodes With DNN Acceleration and Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode," *IEEE J. Solid-State Circuits*, vol. 57, no. 1, pp. 127-139, Jan. 2022.

[48] G. Ottavi et al., "Dustin: A 16-Cores Parallel Ultra-Low-Power Cluster With 2b-to-32b Fully Flexible Bit-Precision and Vector Lockstep Execution Mode," *IEEE Trans. Circuits Syst. I: Regul. Pap.*, vol. 70, no. 6, pp. 2450-2463, Jun. 2023.

[49] G. K. Chen et al., "An Eight-Core RISC-V Processor With Compute Near Last Level Cache in Intel 4 CMOS," *IEEE J. Solid-State Circuits*, vol. 58, no. 4, pp. 1117-1128, Apr. 2023.

[50] X. Wu et al., "Amoeba: An Efficient and Flexible FPGA-Based Accelerator for Arbitrary-Kernel CNNs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 32, no. 6, pp. 1086-1099, Jun. 2024.

[51] V. Gokhale et al., "Snowflake: An efficient hardware accelerator for convolutional neural networks," *IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2017, pp. 1-4.

[52] P. Meloni et al., "Exploring NEURAghe: A Customizable Template for APSoC-Based CNN Inference at the Edge," *IEEE Embed. Syst. Lett.*, vol. 12, no. 2, pp. 62-65, Jun. 2020.

[53] J. Gao et al., "An NoC-based CNN Accelerator for Edge Computing," *15th IEEE Int. Conf. ASIC (ASICON)*, 2023, pp. 1-4.

[54] Y. -H. Chen et al., "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127-138, Jan. 2017.

[55] 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," *IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2024, pp. 492-494.

[56] J. Sim et al., "An Energy-Efficient Deep Convolutional Neural Network Inference Processor With Enhanced Output Stationary Dataflow in 65-nm CMOS," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 1, pp. 87-100, Jan. 2020.

[57] A. Carsello et al., "Amber: A 367 GOPS, 538 GOPS/W 16nm SoC with a Coarse-Grained Reconfigurable Array for Flexible Acceleration of Dense Linear Algebra," *IEEE Symp. VLSI Technol. Circuits*, 2022, pp. 70-71.

[58] S. Yin et al., "A 1.06-to-5.09 TOPS/W reconfigurable hybrid-neural-network processor for deep learning applications," *IEEE Symp. VLSI Technol. Circuits*, 2017, pp. C26-C27.

[59] A. Garofalo et al., ”PULP-NN: accelerating quantized neural networks on parallel ultra-low-power RISC-V processors,” *Phil. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 378, no. 2164, Dec. 2019, Art. no. 20190155.

[60] A. Burrello et al., "A Microcontroller is All You Need: Enabling Transformer Execution on Low-Power IoT Endnodes," *IEEE Int. Conf. Omni-Layer Intell. Syst. (COINS)*, 2021, pp. 1-6.

[61] C. Yu et al., "A 65-nm 8T SRAM Compute-in-Memory Macro With Column ADCs for Processing Neural Networks," *IEEE J. Solid-State Circuits*, vol. 57, no. 11, pp. 3466-3476, Nov. 2022.

[62] C. Yu et al., "A Dual 7T SRAM-Based Zero-Skipping Compute- In-Memory Macro With 1-6b Binary Searching ADCs for Processing Quantized Neural Networks," *IEEE Trans. Circuits Syst. I: Regul. Pap.*, vol. 71, no. 8, pp. 3672-3682, Aug. 2024.

[63] A. Stillmaker et al., "Scaling equations for the accurate prediction of CMOS device performance from 180nm to 7nm," *Integration, the VLSI J.*, Vol. 58, pp. 74-81, Jun. 2017.

Paper IV

NMCu-CNN: A Scalable Near-Memory Computing Co-Processor on a RISC-V MCU in 22 nm FDSOI

This paper presents NMCu-CNN, a near-memory computing (NMC) co-processor for the hardware acceleration of convolutional neural networks (CNNs) on a low-power, flexible MCU platform. The scalable architecture and the selected platform enable adaptability to the rapidly evolving Edge AIoT landscape, where energy and performance requirements are constrained. Application-tailored NMC units, equipped with an optimized CNN dataflow, are integrated into the shared memory address space of a RISC-V-based MCU. The proposed architecture supports highly flexible runtime configurability, achieving 94 % computational efficiency. Fabricated in 22 nm FDSOI technology, NMCu-CNN delivers a performance of 203 GOPS and an energy efficiency of 1716 GOPS/W (1.7 TOPS/W) benchmarked on convolutional layers of a CNN model, outperforming the processing capabilities of other state-of-the-art techniques.

©2025 IEEE. Reprinted, with permission, from
A. Prieto, M. Nouripayam, K. Westring, S. Castillo Mohedano, A. Allfjord, L. Svensson, P. Andersson, and J. Rodrigues,
"NMCu-CNN: A Scalable Near-Memory Computing Co-Processor on a RISC-V MCU in 22 nm FDSOI," in *IEEE Workshop on Signal Processing Systems (SiPS)*, 2025.

I. INTRODUCTION

In the era of Artificial Intelligence of Things (AIoT), the growing volume of data from distributed sources, such as sensors, coupled with rising concerns about latency and privacy, is driving the need for local data processing [1]. Higher processing demands have accelerated the adoption of edge computing, where data is processed near its source to reduce communication overhead and improve responsiveness [2]. Among edge workloads, Convolutional Neural Networks (CNNs) are especially prominent due to their effectiveness in vision-centric tasks such as image classification, object detection, and segmentation [3]. However, deploying CNNs at the edge is challenging due to limited computational and energy resources, necessitating hardware-efficient processing techniques.

In-memory and near-memory computing (IMC and NMC) have emerged as key near-data processing paradigms for accelerating CNN inference on resource-constrained platforms. IMC tightly embeds computation within memory arrays, achieving high peak performance for fixed-function models. However, disruption of conventional memory architectures limits scalability, increasing design complexity, and offering limited system-level performance compared to peak throughput in niche applications [4]. In contrast, digital NMC offers a more scalable and flexible alternative that integrates logic in close proximity of memory with standard design flows. As neural networks evolve and edge devices are expected to support diverse applications of varying sizes, efficient NMC integration into general-purpose programmable cores enables high-performance while preserving modularity and specialization.

This work introduces NMCu-CNN, a near-memory co-processor designed to efficiently address the challenges of AIoT applications based on the principles of our NMC framework presented in [5]. The architecture is tailored for seamless integration into low-cost, RISC-V-based microcontroller units (MCUs), and embedded in the shared memory subsystem. The proposed NMC architecture offers the following advantages: (1) runtime configurability, enabling adaptation of the hardware to application-specific requirements and support for a wide range of CNN models and sizes; (2) high-performance, low-power parallel processing, achieved by efficiently offloading computation-intensive edge AI tasks to NMC units while preserving the flexibility of a software programmable main core; and (3) low integration effort, ensured by compatibility with foundry-supplied optimized SRAMs and standard digital design flows, allowing for scalable parallelism through the modular addition of multiple NMC units.

The remainder of this paper is organized as follows. Section II describes the hardware implementation of the proposed NMC architecture and the adopted dataflow. Measurement results are presented in Section III, followed by a

discussion and comparative analysis with state-of-the-art (SoA) in Section IV. Finally, Section V summarizes the key challenges and contributions of this work.

II. NMCU-CNN IMPLEMENTATION

This study presents the hardware integration of an NMC co-processor in a RISC-V platform. Key design metrics and trade-offs, i.e., performance, energy efficiency, programmability, and configurability, are explored to meet the diverse and dynamic computational demands of edge AI. NMC was chosen as a suitable technique for hardware acceleration considering throughput and energy. Moreover, from a system perspective, NMC is a leading solution in terms of scalability, engineering overhead, and technology portability. To support a wide range of CNN models with varying sizes and characteristics, the architecture incorporates a parameterized hardware architecture and dataflow, specified at RTL level.

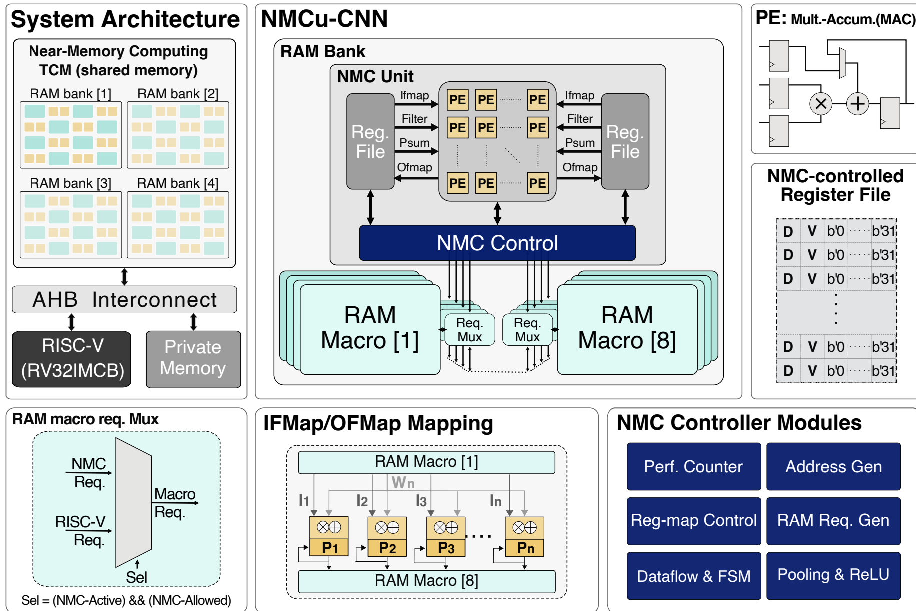


Fig. 1: High-level view of our proposed NMCu-CNN architecture including system integration and design modules.

II.A. HARDWARE ARCHITECTURE

NMCu-CNN is implemented as a fully digital co-processor, integrated into the memory subsystem of a RISC-V-based MCU using Codasip Studio [6], a commercial design platform for customizable low-power processors. At system level, the design features a 512 KB tightly coupled memory (TCM) divided into four banks. A physical memory partition reduces contention during concurrent memory access of multiple processors. Prior studies demonstrated that a ratio of two between the number of memory banks and processors is appropriate [7, 8], and thus, adopted in this architecture. The proposed memory partitioning forms the foundation of a multiprocessor system-on-chip (MPSoC), comprising an NMC co-processor, implemented as four independently configurable units, alongside a 32-bit general-purpose RISC-V core (RV32IMCB), all sharing access to the unified address space of a scratchpad memory (SPM), as shown in Fig. 1.

The design supports architecture scalability through a configurable memory partitioning scheme at both bank and macro levels. At bank level, each NMC unit operates in a dedicated memory bank, allowing independent process execution per bank. At macro level, fine-grained partitioning enhances effective memory bandwidth by allocation of memory macros to the processing elements (PEs), which are tailored for an efficient hardware acceleration. The proposed architecture consists of eight macros per bank, increasing memory bandwidth up to 32× compared to the RISC-V core. The macro fine-grain memory structure is transparent to the RISC-V, avoiding specific software intervention. However, macro level partitioning is explicitly structured to apply parallelism and locality to the NMC units, allowing efficient data streaming and scheduling.

Control and synchronization between NMC units and RISC-V core is organized via dedicated I/O channels over the advanced high-performance bus (AHB) interconnect. Memory access requests, from both the RISC-V core and the NMC units, are arbitrated to the shared RAM macros within each bank, as shown in Fig. 2. A Round-Robin arbitration mechanism schedules concurrent accesses across the eight macros in each bank, ensuring consistent latency and fair access while avoiding starvation. To further mitigate memory contention and prevent data corruption, a utility-based static partitioning scheme, adjustable at runtime, is used to isolate memory regions. Task-level dependencies are resolved through interrupt-based synchronization, using a dedicated interrupt line from each NMC unit to the RISC-V core, avoiding the need for polling and reducing processor overhead.

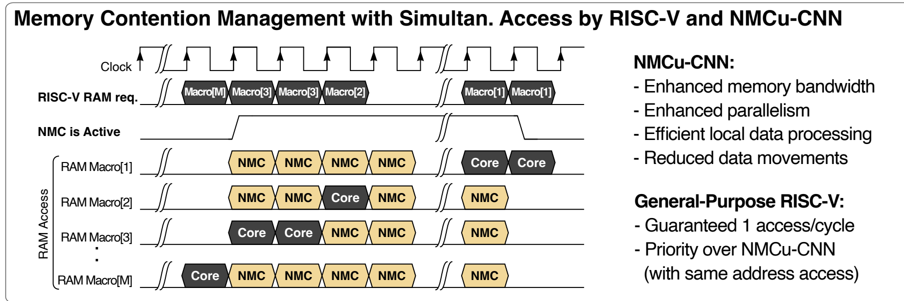


Fig. 2: NMCu-CNN and RISC-V core memory access waveforms.

II.B. DATAFLOW

Runtime configurability and workload adaptability are central to NMCu-CNN, which contains a customizable dataflow, tailored to the optimized data movement requirements of specific applications. The flexible organization of memory banks and macros promotes design reuse, and also guarantees long-term adaptability. Combined with the general-purpose programmability of the RISC-V core, the architecture efficiently handles workloads of different nature in a combined NMC-RISC-V computing scenario, ensuring sustained relevance as AI workloads evolve.

1) CONFIGURABILITY

Each NMC unit is configured through software-controlled memory-mapped registers, accessible via C/C++ software programming on the RISC-V core. This interface enables individual control and data movement using a dedicated range of virtual addresses. CNN-specific settings are encoded as 32-bit words and transmitted to NMC units in two consecutive vectors using predefined function calls, as illustrated in Fig. 3. The first vector specifies layer parameters such as input feature map (IFMap) size, number of channels, kernel size, and output feature map (OFMap) size. The second vector contains auxiliary settings, including quantization mode, padding, pooling, and activation function. Both vectors are stored in the configuration registers of each NMC unit. All external control of the NMC units is managed from software, and upon completion of any operation, a dedicated interrupt line triggers a task-synchronization signal to the RISC-V core.

2) EXECUTION MODEL

The architecture adopts an output-stationary (OS) weight-unicast dataflow for convolutional operations considering pooling sampling. This dataflow

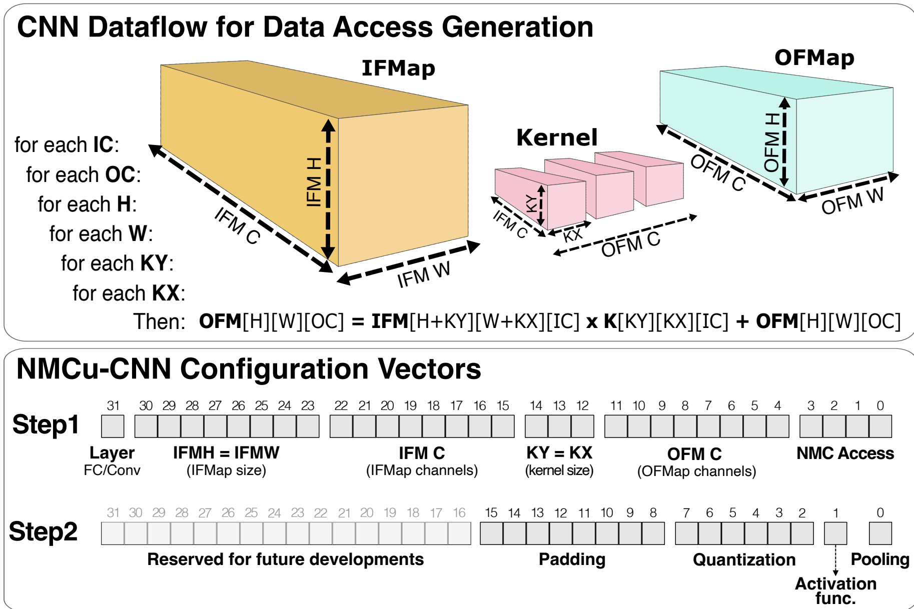


Fig. 3: CNN dataflow and setup configuration for controlling NMCu-CNN.

strategy reduces the need for temporary memory storage of partial outputs and applies IFMap data reuse, significantly reducing energy-intensive data movement and improving overall efficiency. The execution follows a scheduling model that is aware of data dependencies, reducing idle cycles and avoiding redundant memory transactions. A double buffering ping-pong scheme enables concurrent data transfer and computation, ensuring continuous utilization of processing resources.

The PEs are arranged in a grid and equipped with local registers and accumulators to perform multiply-accumulate (MAC) operations. The array size is determined at RTL level based on the available memory macros per bank and the intermediate buffer allocation for input and output streaming. In the implemented configuration, each bank includes a 4×4 PE array per NMC unit, totaling 64 PEs across the system. This configuration provides sufficient internal bandwidth to support double buffering and stalling-free streaming, allowing high-throughput operation under the OS dataflow model.

III. MEASUREMENT RESULTS

NMCu-CNN is fabricated in GlobalFoundries (GF) 22 nm FDSOI technology. The chip is organized into multiple power domains, where memories and

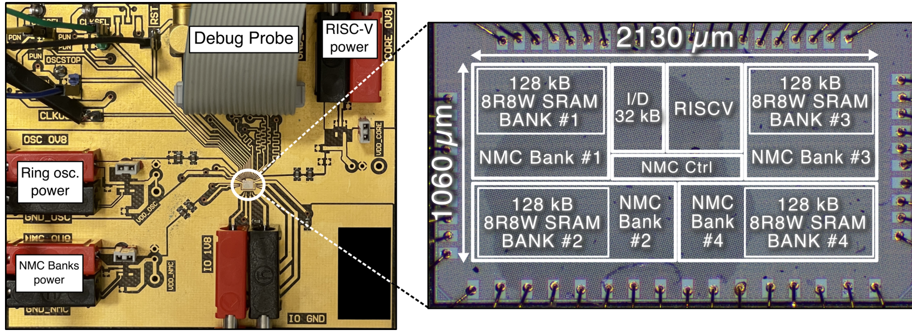


Fig. 4: Custom-designed PCB for measurement (left), and micrograph of the fabricated chip (right).

NMC units are grouped together. Thereby, a flexible testing and performance evaluation under various conditions is enabled. A die micrograph of the fabricated SoC, along with the custom-designed printed circuit board (PCB), is shown in Fig. 4. For benchmarking, the NMC units are configured with representative convolutional layers of VGG-16 neural network according to the configuration setup defined in Fig. 3.

Measurements are performed at room temperature (≈ 25°C), using an on-chip tunable ring oscillator, which facilitates frequency and supply voltage (V_{DD}) scaling. Maximum operating frequency and computational performance are characterized by varying V_{DD} of the memory-NMC units. The minimum operating voltage, which passes the benchmark tests without accuracy loss, is recorded at 600 mV, supporting a frequency of 100 MHz, see Fig. 5a. The nonlinearities in frequency response to V_{DD} stem from variations of the nonlinear control of the ring oscillator and second-order device effects reducing effective threshold voltage. Accuracy measurements sweeping V_{DD} and clock frequency are presented in Fig. 5b, where color-coded boxes represent computation accuracy.

Multiple configuration scenarios are evaluated by activating one, two, and four NMC units. During a full system utilization, all 64 PEs operate concurrently. The modular architecture enables an advantageous scalability, where performance increases proportionally to the number of active NMC units. A controlled disposition of active banks demonstrates the adaptability and scalability of the proposed parameterized architecture, as illustrated in Fig. 6a. The maximum measured performance, for 8-bit precision operations, is 203 GOPS at 420 MHz, with an overall computational efficiency of 94 %. A voltage sweep from 600 mV to 950 mV results in a 4.2× improvement in performance. Furthermore, the energy efficiency of the NMCu-CNN across different configurations is presented in Fig. 6b. Parallel processing using four

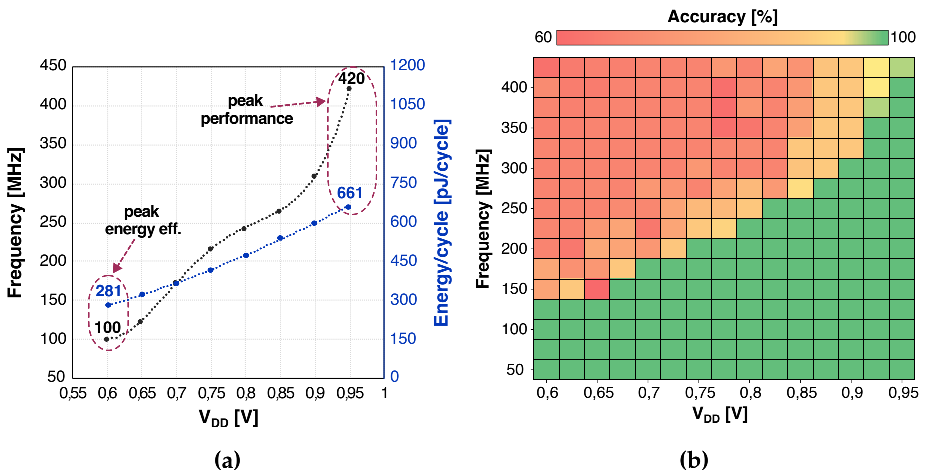


Fig. 5: (a) Operational frequency across the supply voltage range V_{DD}, and (b) Shmoo plot of frequency and computation accuracy as function of V_{DD}.

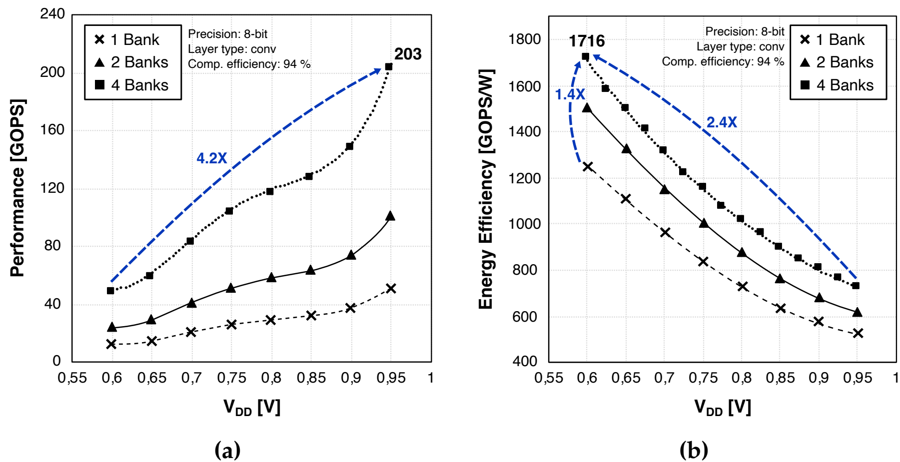


Fig. 6: Design modularity evaluation as a function of active NMC-memory banks and their impact on (a) performance and (b) energy efficiency.

NMC units, compared to a single unit, leads to a 1.4× improvement in energy efficiency in the low-voltage region, i.e., 600 mV. Operating at higher frequencies achieves a higher performance, whereas energy efficiency improves at lower voltages, which stems from the quadratic dependence on the operating voltage. Consequently, when utilizing four NMC units in parallel, reducing the supply voltage from 0.95 V to 0.6 V results in a 2.4 × improvement in

energy efficiency. The peak energy efficiency is observed under low-voltage operation, reaching 1716 GOPS/W (1.7 TOPS/W) at 100 MHz and 600 mV.

IV. DISCUSSION

To highlight the significant improvements in metrics achieved by the tailored NMCu-CNN architecture, an identical convolutional workload is executed by both the RISC-V, working as central core of the MCU platform, and NMC units. A comparison of active power consumption of the general-purpose RISC-V core and NMCu-CNN, including the computational efficiency of the four banks during runtime, is shown in Fig. 7a. Due to the significantly higher performance and reduced latency, NMCu-CNN completes workloads in a much shorter time frame, resulting in a transient rise in power profile confined to a brief execution period. In contrast, the RISC-V core consumes power continuously throughout the entire execution period, leading to a higher total energy consumption. Furthermore, Fig. 7b presents a normalized comparison of performance and energy efficiency for both architectures under the same workload. The results show that NMCu-CNN delivers a 361× increase in performance and a 28× improvement in energy efficiency compared to the RISC-V baseline.

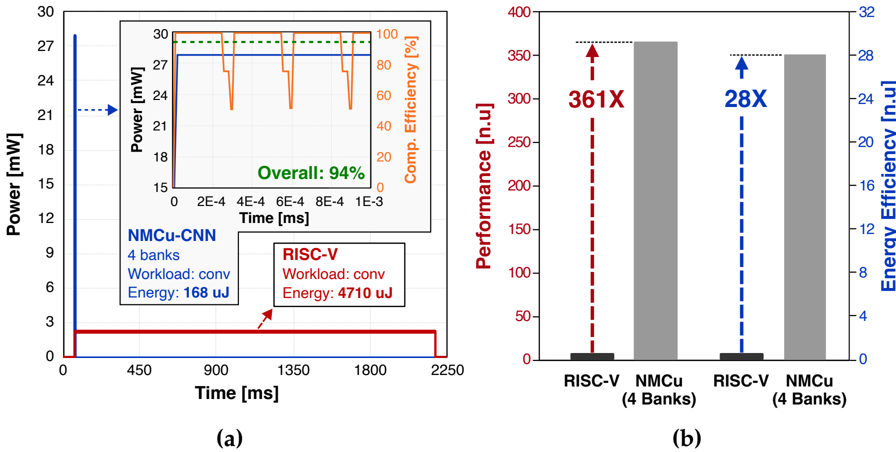


Fig. 7: (a) Power consumption as a function of time including NMCu-CNN computational efficiency, and (b) comparison of NMCu-CNN and RISC-V processing capabilities with performance and energy efficiency for the same convolutional workload.

Table 1: Measurement results and comparison with SoA.

Source	CNC [9]	Darkside [10]	SamurAI [11]	Vecim [12]	This work
Technology	Intel 4	65 nm	28 nm	65 nm	22 nm
Type	NMC	Cluster	Accelerator	IMC	NMC
Die area [mm^2]	1.9	12	4.5	4	2.3
Voltage [V]	0.55 - 0.85	0.75 - 1.2	0.45 - 0.9	1	0.6 - 0.95
Max. Freq. [MHz]	1150	290	350	250	420
Precision [bits]	8	2 to 32	8	8/bf16/fp16	8 to 32
Memory [KB]	512	384	464	16	512
Processing Units	128	8×RISC-V	64	-	64
Performance [GOPS] *	75.8	17	36	32	**203** [a]
Area Eff. [GOPS/mm^2] †	11.1	10.1	8	56.4	**88.3** [a]
Energy Eff. [GOPS/W]	285	191	1300	289	**1716** [b]
Comp. Eff. [%]	26	91	87	-	**94**

* Normalized to 8-bit precision. † Scaled to 22 nm according to [13].
[a] @420 MHz (0.95 V). [b] @100 MHz (0.6 V).

The measurement results are summarized and compared with other SoA works in Table 1. To ensure a fair and accurate comparison, variations in precision and area of technology nodes are considered, and the reported values are normalized accordingly. Performance is represented in giga operations per second (GOPS), normalizing the results to 8-bit precision. Additionally, area and energy efficiency are calculated according to chip dimensions, considering technology scaling, and power consumption, respectively. A compute near last level cache (CNC) design using an NMC technique is implemented in [9], where performance and energy efficiency are achieved by extending RISC-V to perform MAC operations on an eight-core processor. Darkside [10], includes a RISC-V cluster with three specialized digital accelerators, while SamurAI [11] combines a machine learning accelerator together with a RISC-V. Alternatively, [12] includes a vector co-processor with IMC, where performance is evaluated by matrix multiplication. The results demonstrate that our work achieves 2.7× higher performance and 8× higher area efficiency than the NMC technique in [9]. Our NMCu-CNN achieves the highest energy efficiency, i.e., more than 30 % higher than [11]. Compared to other SoA designs, NMCu-CNN achieves more than 6× higher energy efficiency. Although part of the energy gain is attributed to the use of an advanced technology

node, the margin remains large enough to still outperform existing solutions. Furthermore, the higher computational efficiency of NMCu-CNN reflects a higher ratio of executed operations to the theoretical peak, outperforming the other works evaluated in this study.

The 2.3 mm^2 area of our SoC implementation includes a RISC-V core as central processing unit, 32 KB of CPU private memory, and four repurposed shared memory banks integrating our NMCu-CNN design near SRAM macros, as shown in Fig. 8. The area breakdown shows that ≈ 69 % of the design is occupied by SRAMs, while NMC logic represents ≈ 29 % of the SoC, offering a digital integration of less than half the memory area.

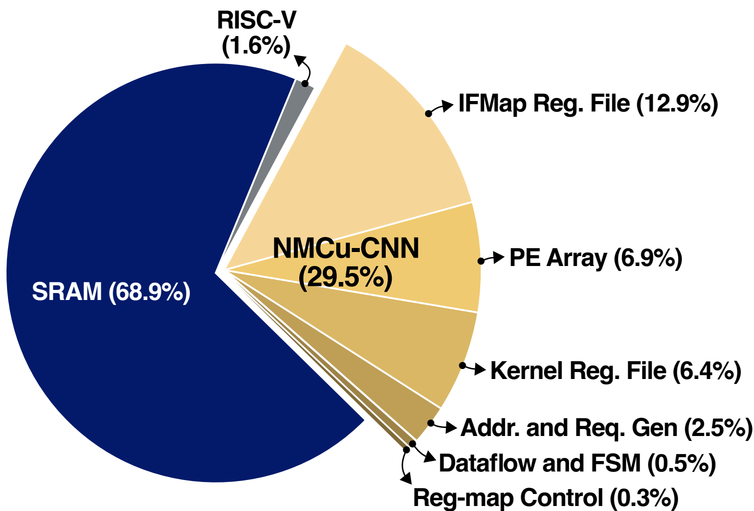


Fig. 8: Area breakdown, including NMCu-CNN integrated near SRAM.

V. CONCLUSION

In this article, we present NMCu-CNN, a parametrized NMC architecture with runtime configurability for CNN workloads. NMCu-CNN is integrated as a co-processor in the memory subsystem of a low-power RISC-V-based MCU. The non-intrusive architecture, fabricated in 22 nm FDSOI technology, is fully compatible with foundry-supplied SRAMs, and provides enhanced memory bandwidth for hardware acceleration. The measurement results indicate that, with an overall computational efficiency of 94 %, the design achieves a performance of 203 GOPS, and an energy efficiency of 1716 GOPS/W (1.7 TOPS/W).

ACKNOWLEDGMENT

The authors are grateful for financial support from Vetenskapsrådet, the SSF-financed classIC Research Center, ELLIIT, Chips-JU REBECCA project under grant agreement no. 101097224, Codasip for the IP, and GlobalFoundries for silicon fabrication.

REFERENCES

[1] J. Zhang and D. Tao, "Empowering Things With Intelligence: A Survey of the Progress, Challenges, and Opportunities in Artificial Intelligence of Things," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 7789-7817, 2021.

[2] N. Abbas, Y. Zhang, A. Taherkordi and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450-465, 2018.

[3] M. Hassanpour, M. Riera, and A. Gonzalez, "A Survey of Near-Data Processing Architectures for Neural Networks," *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 66–102, 2022.

[4] K. -T. Tang et al., "Considerations of Integrating Computing-In-Memory and Processing-In-Sensor into Convolutional Neural Network Accelerators for Low-Power Edge Devices," *Symposium on VLSI Technology*, 2019, pp. T166-T167.

[5] M. Nouripayam, A. Prieto and J. Rodrigues, "A Scalable All-Digital Near-Memory Computing Architecture for Edge AIoT Applications," in *IEEE Access*, 2025.

[6] Codasip, "White paper: Product overview," https://codasip.com/wp-content/uploads/2024/07/Codasip-Product-Brief-Fusion-EN.pdf (accessed 2025-03-26)

[7] J. Ax et al., "CoreVA-MPSoC: A Many-Core Architecture with Tightly Coupled Shared and Local Data Memories," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 5, 2018, pp. 1030-1043.

[8] L. Benini et al., "P2012: Building an ecosystem for a scalable, modular and high-efficiency embedded computing accelerator," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2012, pp. 983-987.

[9] G. K. Chen et al., "An Eight-Core RISC-V Processor With Compute Near Last Level Cache in Intel 4 CMOS," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 4, pp. 1117-1128, 2023.

[10] A. Garofalo et al., "Darkside: 2.6GFLOPS, 8.7mW Heterogeneous RISC-V Cluster for Extreme-Edge On-Chip DNN Inference and Training," *IEEE 48th European Solid State Circuits Conference (ESSCIRC)*, 2022, pp. 273-276.

[11] I. Miro-Panades et al., "SamurAI: A Versatile IoT Node With Event-Driven Wake-Up and Embedded ML Acceleration," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 6, pp. 1782-1797, 2023.

[12] Y. Wang, M. Yang, C. -P. Lo and J. P. Kulkarni, "30.6 Vecim: A 289.13 GOPS/W RISC-V Vector Co-Processor with Compute-in-Memory Vector Register File for Efficient High-Performance Computing," *IEEE International Solid-State Circuits Conference (ISSCC)*, 2024, pp. 492-494.

[13] A. Stillmaker et al., "Scaling equations for the accurate prediction of CMOS device performance from 180nm to 7nm", *Integration, the VLSI Journal*, vol. 58, pp. 74-81, 2017.

Paper V

LUCIA: Compute Inter-chip Architecture for Multi-chiplet Acceleration in 22 nm FDSOI

Chiplet-based architectures have emerged as a robust system-in-package (SiP) design paradigm, enabling scalability and parallel processing for disaggregated high-performance computing systems. In this work, we present LUCIA, an architecture for multi-chiplet systems optimized for convolutional neural network (CNN) acceleration through configurable hardware engines and distributed workload mapping. The design integrates near-memory computing (NMC) units in close proximity to memory banks and is tightly coupled with a low-power RISC-V processor. Dedicated memory control units facilitate efficient intra-chiplet and inter-chiplet data transfers, minimizing latency and maximizing bandwidth utilization. Fabricated in 22 nm FDSOI technology, LUCIA achieves a peak operating frequency of 565 MHz and demonstrates an energy-area efficiency of 0.83 TOPS/W/mm^2, outperforming comparable state-of-the-art implementations.

©2025 IEEE. Reprinted, with permission, from
A. Prieto, M. Nouripayam, K. Westring, W. Marnfeldt, A. Allfjord, S. Castillo Mohedano, V. Åberg, L. Svensson, P. Andersson, and J. Rodrigues,
"LUCIA: Compute Inter-chip Architecture for Multi-chiplet Acceleration in 22 nm FDSOI".

I. INTRODUCTION

The rapidly expanding ecosystem of artificial intelligence of things (AIoT) with distributed processing on local devices demands hardware architectures that provide high performance while adhering to the memory and energy constraints of edge devices. The implementation of large designs on a single monolithic die carries a high risk of yield issues, as even a single fabrication defect can render the entire die unusable [1]. Consequently, disaggregating a well-optimized large system into interconnected chiplets to form a parallel, distributed, and modular processing architecture has emerged as a promising approach to improve yield and enhance edge devices capabilities for hardware acceleration. However, chiplet-based architectures require dedicated workload partitioning, task scheduling, and inter-chip communication, which poses significant challenges.

Manticore [2] and Occamy [3] introduce large chiplet-based architectures with many RISC-V cores for data parallelization, using a passive silicon interposer as the interconnection medium. As an alternative to a large cluster of RISC-V cores, the integration of tailored hardware acceleration units with low-power general-purpose platforms enables to combine resources, offering efficient solutions to process computation-intensive tasks on edge devices.

This work introduces a customized near-memory computing (NMC) integration technique [4], purposefully engineered to exploit the parallelism and data locality of a multi-chiplet architecture. The proposed architecture processes convolutional neural network (CNN) models through modular workload distribution and task mapping strategies for high performance, energy, and area efficiency. The contribution of this work centers on a versatile and scalable processing platform, accomplished by an efficient integration of computing and memory units with an intra-chiplet data transfer scheme. Furthermore, the proposed architecture includes a low-cost peripheral memory access interface for inter-chiplet communication.

II. LUCIA IMPLEMENTATION

The LUCIA architecture integrates near-memory computing (NMC) units alongside the memory banks of a RISC-V-based processing platform enabling tightly coupled hardware acceleration, as shown in Fig. 1. This shared integration of compute and memory forms the architectural backbone of a scalable implementation for a chiplet, where distributed NMC banks minimize data movement and enhance parallelism.

In this study, a custom-developed printed circuit board (PCB) that meets the demands of redistribution layer (RDL) is used for chiplet assembly, emulating

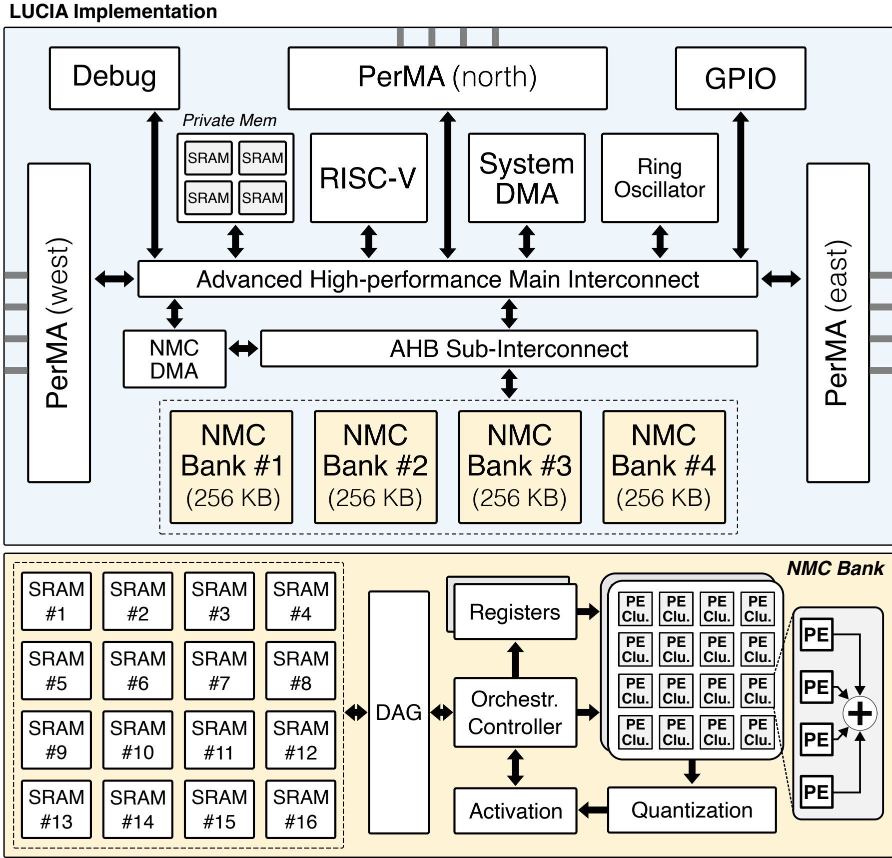


Fig. 1: High-level view of the LUCIA implementation with memory access units and embedded NMC units for hardware acceleration.

the use of an organic or silicon interposer. The proposed framework establishes a foundational scenario for achievable higher-density integration in multi-chiplet systems through advanced packaging techniques. Data transfer flexibility is enhanced by equipping LUCIA with two specialized memory access units: a direct memory access (DMA) engine for intra-chiplet data movement, and a peripheral memory access (PerMA) unit designed for low-latency inter-chiplet communication. To mitigate contention between NMC bank units and peripheral modules, the DMA engine is partitioned into two independent units: system-DMA and NMC-DMA. While the system-DMA functions as a conventional controller for general data transfers, the NMC-DMA is dedicated to orchestrating high-throughput communication across

memory banks, ensuring conflict-free access and sustained performance. LUCIA is equipped with three PerMA modules: east and west links for communication with adjacent chiplets, and a north link interfacing a central host.

The memory sub-system is partitioned into four memory banks, integrating one NMC unit each. The hardware acceleration NMC units are configurable to operate either cooperatively or independently, allowing flexible task scheduling within the multi-bank architecture, and thereby supporting distributed edge AI applications.

II.A. MEMORY SUB-SYSTEM

The on-chip memory, as part of a tightly integrated sub-system, facilitates concurrent and low-latency access by both the RISC-V core and the NMC units. This is achieved through a shared banking strategy that minimizes load capacitance and maximizes operating frequency, while mitigating access contention [5]. By provisioning multiple accelerator engines with a unified memory pool, the design eliminates data duplication and reduces overall storage requirements, resulting in substantial improvements in area efficiency, power consumption, and memory bandwidth utilization.

In alignment with these design principles, each bank of the LUCIA memory sub-system is partitioned into sixteen 16 kB SRAM macros, enabling high-throughput parallel access and significantly increasing aggregate bandwidth for hardware acceleration. The nominal memory bandwidth is preserved for RISC-V core accesses and external interfaces to maintain system-level consistency. In contrast, the bandwidth available to NMC units is scaled by a factor of 16—from 32 bit/cycle to 512 bit/cycle—matching internal macro parallelism and supporting wide concurrent data transfers. This elevated communication rate imposes stricter demands on data orchestration and introduces control overhead, which can impact implementation efficiency. For that reason, intra-chiplet DMA modules retain the baseline 32 bit/cycle transfer rate for streamlined integration, and PerMA modules utilize a 2-bit transmitter-receiver interface to balance throughput and complexity. Additionally, the architecture incorporates a tunable on-chip ring oscillator (RO) for dynamic voltage and frequency scaling (DVFS), alongside GPIO and debug ports for external control and real-time monitoring via the RISC-V core.

Each NMC bank is configured to accelerate convolution and matrix multiplication workloads using 8-bit precision for multiplication and 32-bit registers for accumulation. Each embedded NMC architecture integrates a dedicated data address generator (DAG) module that orchestrates fine-grained access control to SRAM, coordinated by an orchestration controller implementing an output-stationary scheduling strategy. Within each bank, two

register arrays—holding input and weight data, feed into dual 4×4 processing element (PE) matrices. Each PE cluster performs four parallel multiply-accumulate (MAC) operations and locally accumulates partial sums, enabling independent output computation. A high-throughput data scheduler ensures continuous parallel access to input data, leveraging enhanced memory bandwidth and tightly coupling compute with memory. For convolutional operations, the input channel data is distributed across memory banks, with the scheduler optimized for local data reuse, thus maximizing parallelism and minimizing data movement, as illustrated in Fig. 2.

Inference-time quantization is applied in a pipelined fashion to output computations, followed by activation via a dedicated module that supports ReLU and other nonlinear functions through configurable lookup tables, whose parameters are dynamically loaded during system initialization. A register-mapped control interface orchestrates the operation of the LUCIA architecture, with start, stop, and configuration registers defining the execution state of each NMC bank. Each NMC–DMA module is programmed with memory pointers, data dimensions, and access sequencing to ensure conflict-free transfers and prevent data overwrites, while maintaining high-throughput inter-bank communication. The PerMA modules act as lightweight, autonomous external communication endpoints, eliminating the need for register-mapped configuration and reducing control overhead. Synchronization between the RISC-V core and NMC banks is achieved via an interrupt-driven mechanism, while inter-bank data streaming is managed by the DMA engine to maintain balanced workload distribution. The DMA engines support multiple data partitioning schemes, enabling flexible reuse of input activations and weights across accelerator engines, thereby optimizing memory bandwidth, reducing redundancy, and improving overall system efficiency.

II.B. VERSATILE CHIPLET CONFIGURATION

LUCIA is optimized for high-efficiency acceleration of convolutional and fully-connected layers in CNNs, with a modular chiplet-based design that enables scalable extension to broader neural network (NN) workloads. Fig. 2 illustrates the NN partitioning strategy across the multi-bank system: Partition A distributes input activations across banks to exploit parallel data access, while Partition B assigns kernel weights across banks to enable concurrent computation. This partitioning scheme maximizes compute density and memory bandwidth utilization, while minimizing external communication overhead.

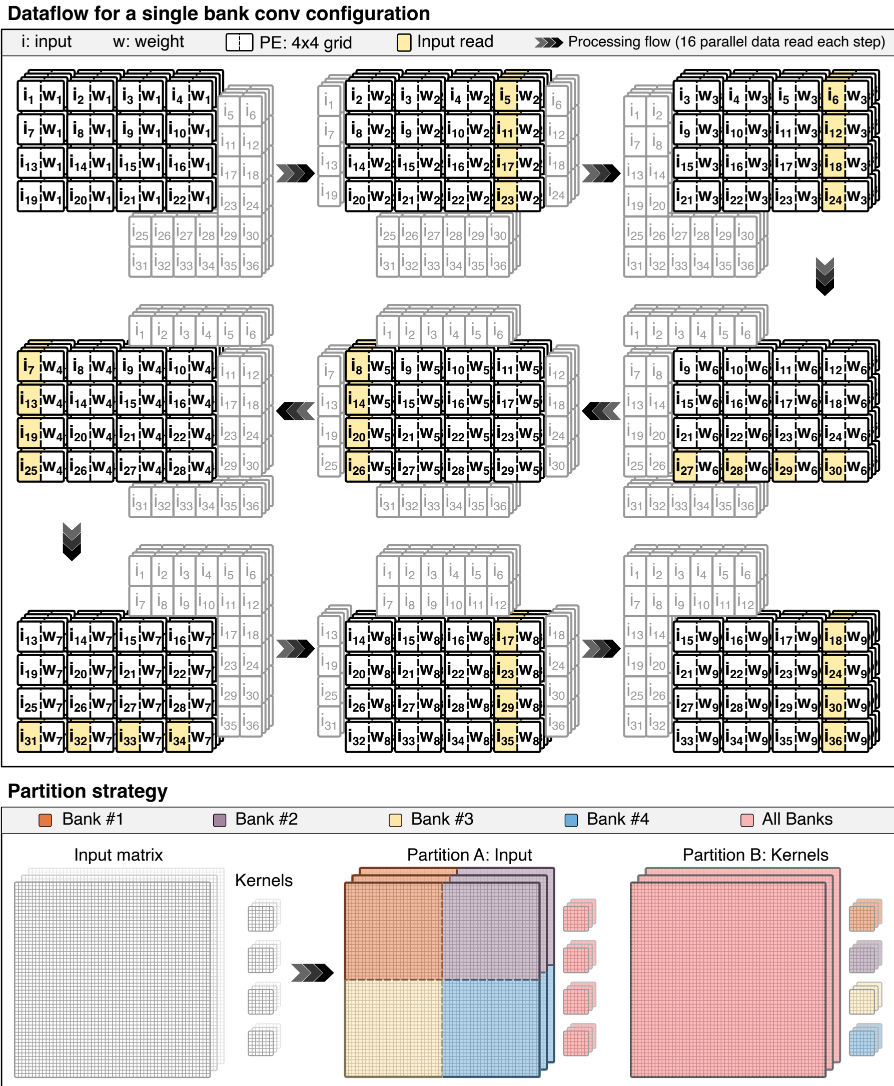


Fig. 2: Data scheduler for convolution operations with parallel memory accesses, and partition strategy for multi-bank computation.

III. MEASUREMENTS

LUCIA is fabricated in GlobalFoundries (GF) 22 nm FDSOI technology, occupying a compact core area of 2.28 mm × 1.78 mm. A custom-designed PCB serves as the integration platform, providing high-speed I/O interfaces, clock

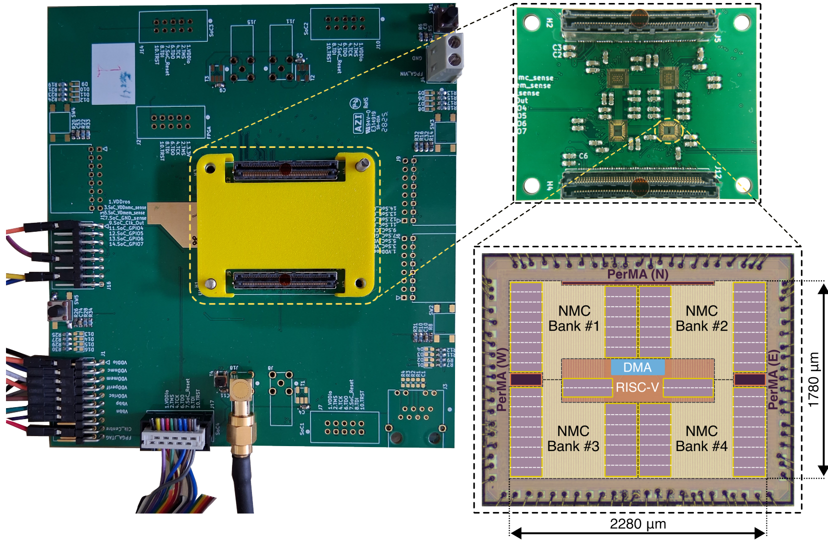


Fig. 3: Measurements test platform including PCB for multi-chiplet assembly, with a zoom on LUCIA architecture and low-level modules.

distribution, debug access, and power delivery for all parts of the design. The measurement setup offering up to four chiplets assembly, illustrated in Fig. 3, includes a micrograph of the LUCIA implementation, highlighting the spatial organization of the NMC memory banks. The architecture is centered around a RISC-V core, which acts as a general-purpose central unit, surrounded by memory blocks (shown in purple) and peripheral modules. Three PerMA units enable external communication, while DMA engines are strategically placed to facilitate high-throughput internal data movement. Measurements are conducted at ambient temperature, with dedicated debug ports enabling fine-grained chiplet programming and evaluation. Convolutional layers are used as benchmark workloads to assess performance and energy efficiency across varying configurations, i.e., from single-bank to multi-bank NMC processing. The voltage-frequency characterization is performed for the four-bank configuration, as shown in Fig. 4a, revealing two distinct operating regimes: a low-power mode optimized for energy-per-cycle, and a high-performance mode supporting elevated throughput. The execution time evaluation of four VGG-16 convolutional layers tiles considering a partition strategy A, with input data distributed across the four NMC banks, is shown in Fig. 4b. Inter-bank data transfer using the custom NMC-DMA engine reduces execution time by 34 % compared to transferring data between banks using the RISC-V core.

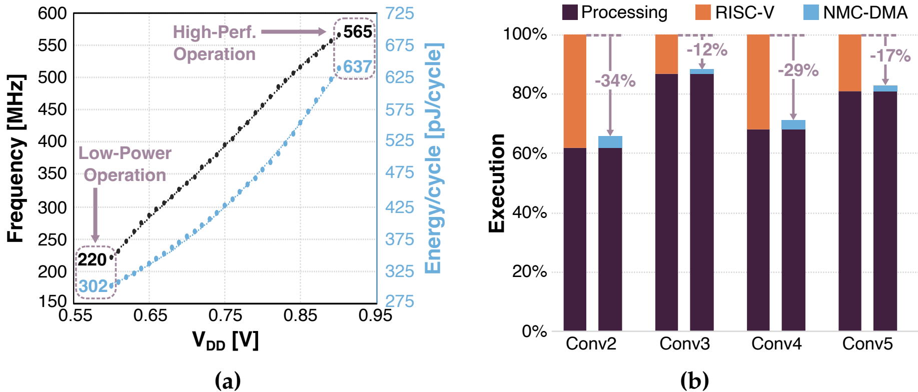


Fig. 4: (a) Measured operational frequency and energy response to the supply voltage (V_{DD}), and (b) VGG-16 convolutional layers execution time with intra-chip data transfers comparison between RISC-V and NMC-DMA.

Performance evaluation, shown in Fig. 5a, is conducted across multiple active-bank configurations, with each MAC operation counted as two arithmetic operations. Under full parallel activation of four NMC banks, the LUCIA implementation achieves a peak throughput of 0.58 TOPS. Energy efficiency, detailed in Fig. 5b, reaches 3.4 TOPS/W in the low-power operating regime with four active banks. To minimize idle power, the NMC units employ fine-grained clock gating (CG), achieving a 28 % improvement in energy efficiency during single-bank operation. Additionally, measured leakage power informs the impact of power gating (PG) on inactive banks, with PG contributing an average 15 % energy efficiency gain, particularly pronounced at elevated voltages due to second-order leakage effects. The area composition of LUCIA is illustrated in Fig. 5c, where SRAM blocks account for approximately 50 % of the total silicon footprint. This includes both the shared memory sub-system used for hardware acceleration and the private memory regions allocated to the RISC-V core.

IV. DISCUSSION

A summary of relevant state-of-the-art (SoA) implementations is provided in Table 1, encompassing both single-chip and multi-chiplet architectures. Two notable single-die solutions are considered: the systolic neural CPU (SNCPU) presented in [6], which offers a reconfigurable architecture capable of operating as a multi-core RISC-V processor or a systolic CNN accelerator;

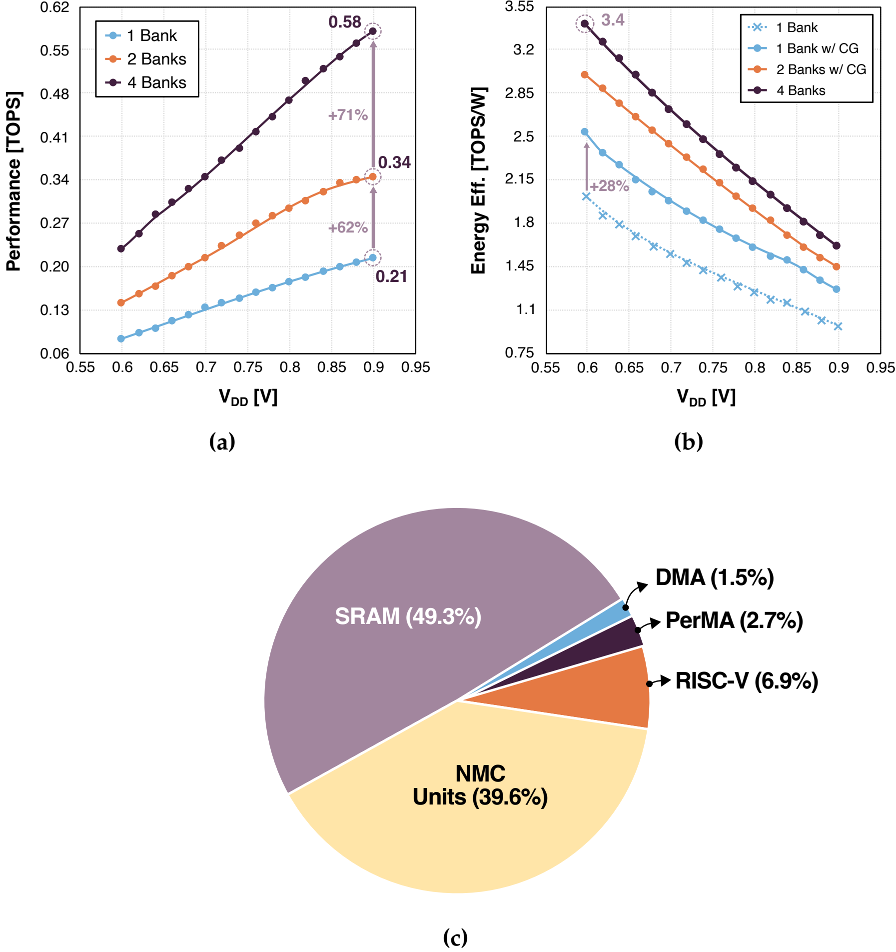


Fig. 5: Design modularity evaluation as a function of active NMC banks and their impact on (a) performance, (b) energy, and (c) area.

and the Siracusa platform in [7], which integrates a RISC-V cluster with a tightly coupled accelerator for efficient AI processing. Multi-chiplet systems are represented by the NetFlex multi-chiplet package (MCP) in [8], designed for scalable networked inference, and the SPGEMM chiplet-based accelerator in [9], targeting sparse matrix multiplication workloads. While these works address the growing computational demands of modern AI models, LUCIA advances the architectural paradigm by explicitly mapping compute-intensive workloads onto a multi-bank topology. It leverages spatial partitioning

Table 1: Measurement results and comparison with SoA.

Source	ISSCC’22 [6]	ESSCIRC’23 [7]	VLSI’22 [8]	CICC’25 [9]	**This work**
Technology	65 nm	16 nm	22 nm	4 nm	22 nm
Design	SNCPU	Clust. + Acc.	MCP	NMC	NMC
Configuration	1-chip	1-chip	4-chip	16-chip	1-chip
Core Area [mm^2]	4.5	16 *	11.1	16	4.1
Voltage [V]	0.5 - 1	0.6 - 0.8	0.6 - 0.9	0.8 - 1.1	0.6 - 0.9
Max. Freq. [MHz]	400	530	492	500	565
Int Precision	8	2 to 8	16	8	8
Memory [MB/chip]	0.15	6.5	2.5	16	1
PEs/chip	100	36	1024	128	512
Performance [TOPS]	0.08	0.38 (int8)	1.07	2 ‡	**0.58**
Energy Eff. [TOPS/W]	1.82 †	2.9 † (int8)	7.19 †	8.7 ‡	**3.4** †
Energy-Area Eff. [TOPS/W/mm2]	0.4	0.18 (int8)	0.65	0.54 ‡	**0.83**

* Silicon die area † @ Low V_{DD} ‡ Dense computation mode

of compute tiles and memory banks, tightly integrated with near-memory computing (NMC) units, to achieve scalable acceleration with reduced data movement and improved memory locality.

LUCIA achieves an energy-area efficiency of 0.83 TOPS/W/mm^2, surpassing comparable SoA implementations fabricated in equal or more advanced technology nodes. This result underscores the effectiveness of LUCIA’s distributed architecture in delivering high throughput and energy efficiency per unit area. The system’s modular design enables scalable deployment across multiple chiplets, each tightly coupled to near-memory compute banks. Future integration on a silicon interposer or through advanced 3D packaging technologies could further enhance inter-chiplet communication density, reduce latency, and unlock additional performance gains through high-bandwidth, low-power interconnects.

V. CONCLUSION

In this article, we present LUCIA as an architecture for a multi-chiplet implementation with NMC hardware acceleration units integrated in memory banks of a RISC-V-based platform. LUCIA is manufactured using 22 nm

FDSOI technology and allows the configuration of partitioned workloads in multiple memory banks with enhanced bandwidth. The results of the measurements show an energy-area efficiency of 0.83 TOPS/W/mm^2, with a performance of 0.58 TOPS in the presented single-chiplet architecture.

ACKNOWLEDGMENT

The authors are grateful for financial support from Vetenskapsrådet, the SSF financed classIC Research Center, and the Chip JU REBECCA project (101097224); Codasip for the RISC-V IP, and GlobalFoundries for silicon fabrication.

REFERENCES

[1] P. Iff, M. Besta, M. Cavalcante, T. Fischer, L. Benini and T. Hoefler, "HexaMesh: Scaling to Hundreds of Chiplets with an Optimized Chiplet Arrangement," *60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1-6.

[2] F. Zaruba, F. Schuiki and L. Benini, "Manticore: A 4096-Core RISC-V Chiplet Architecture for Ultraefficient Floating-Point Computing," in *IEEE Micro*, vol. 41, no. 2, pp. 36-42, 2021.

[3] G. Paulin et al., "Occamy: A 432-Core 28.1 DP-GFLOP/s/W 83% FPU Utilization Dual-Chiplet, Dual-HBM2E RISC-V-Based Accelerator for Stencil and Sparse Linear Algebra Computations with 8-to-64-bit Floating-Point Support in 12nm FinFET," *IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, 2024, pp. 1-2.

[4] M. Nouripayam, A. Prieto and J. Rodrigues, "A Scalable All-Digital Near-Memory Computing Architecture for Edge AIoT Applications," in *IEEE Access*, vol. 13, pp. 108609-108625, 2025.

[5] L. Benini, E. Flamand, D. Fuin and D. Melpignano, "P2012: Building an ecosystem for a scalable, modular and high-efficiency embedded computing accelerator," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2012, pp. 983-987.

[6] Y. Ju and J. Gu, "A 65nm Systolic Neural CPU Processor for Combined Deep Learning and General-Purpose Computing with 95% PE Utilization, High Data Locality and Enhanced End-to-End Performance," *IEEE International Solid-State Circuits Conference (ISSCC)*, 2022, pp. 1-3.

[7] M. Scherer et al., "Siracusa: A Low-Power On-Sensor RISC-V SoC for Extended Reality Visual Processing in 16nm CMOS," *IEEE 49th European Solid State Circuits Conference (ESSCIRC)*, 2023, pp. 217-220.

[8] T. Chou et al., "NetFlex: A 22nm Multi-Chiplet Perception Accelerator in High-Density Fan-Out Wafer-Level Packaging," *IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, 2022, pp. 208-209.

[9] S. R. Srinivasa et al., "A 68 TOPS/W, 256MB SRAM Sparse GEMM Accelerator Tiled Across 16, 4nm Near Memory Compute (NMC) Chiplets Disaggregated 2.5D System," *IEEE Custom Integrated Circuits Conference (CICC)*, 2025, pp. 1-3.