

# Architecture Exploration and Optimization of Heterogeneous Many-Core Compute and Memory Architectures with Architectural Extensions

Présentée le 10 septembre 2021

Faculté des sciences et techniques de l'ingénieur  
Laboratoire des systèmes embarqués  
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

**Yasir Mahmood QURESHI**

Acceptée sur proposition du jury

Prof. A. M. Alahi, président du jury  
Prof. D. Atienza Alonso, directeur de thèse  
Prof. A. Coskun, rapporteuse  
Dr J. Picorel, rapporteur  
Prof. A. Burg, rapporteur



There is only one thing that makes a dream impossible to achieve:  
the fear of failure  
— Paulo Coelho

To my loving parents;  
To my amazing and wonderful wife;  
And to my beautiful lively kids.



# Acknowledgements

To being with, first and fore-most, I will to thank and express my deepest gratitude to my thesis director **Prof. David Atienza** for not only giving me the opportunity to pursue my Ph.D. at the Embedded Systems Laboratory (ESL), but also supervising and guiding me throughout this journey. This Ph.D. thesis would not have been possible without his continuous support and advise. I will also like to thank him for helping me grow professionally and enabling me to gain a wide range of skills not just from research perspective, but as a person as well. I am also grateful to him for creating a conformable and amiable environment at work and taking care of our needs in the lab. Additionally, I will like to thank him for all the ESL activities including annual Christmas dinners, summer BBQs and ESL team building activities, which is something that is very unique to ESL. Finally, I appreciate and am grateful to him for being compassionate, considerate and always listening to my personnel circumstances and accommodating them accordingly.

I will like take this opportunity to thank and and express my gratitude to the jury members comprising of **Prof. Alexandre Alahi, Prof. Andreas Burg, Prof. Ayse Coskun, and Dr. Javier Picorel**, for taking out time from their busy schedules to evaluate this thesis and providing valuable feedback.

Then, I will like to thank **Prof. Marina Zapater**, who was my supervising post-doc and go to person throughout my Ph.D. I am grateful to her for helping and guiding me whenever I was stuck in something, encouraging and supporting me whenever there was a rejected paper and finally, reviewing and helping me with different publications, which would not have been possible without her valuable feedback. I will like to thank **Prof. Katzalin Olcoz, Prof. Sonia Gonzalez-Navarro and Prof. Oscar Plata**, with whom I have various publications in different collaborations. I am also particularly grateful to **Miguel Peon Quiros** and **Luis Costero** for reviewing and proof-reading this thesis and providing valuable feedback on it.

Furthermore, I will like thank ESL's administrative assistant **Homeira Salimi** taking care of all the administrative matters, from booking hotels and flights abroad for conferences to organization of Ski trips, ESL dinners and birthdays. I will like to appreciate and thank the IT administrators at ESL, including **Rodolphe Buret, Mikael Doche** and **Thomas Christoph** for resolving my day-to-day IT related problems, and enabling me to carry out my research seamlessly. I will also like to thank the post-doctoral researchers in the lab, including **Ruben Braojos, Amir Aminifar, Leila Cammoun, Giovanni Ansaloni, Adriana Arza, Alexandre Levisse, Dennis Majoe and Tomas Teijeiro**, with whom I worked directly or indirectly, either for work, research or just a chat in the kitchen. Moreover, I will like to express my gratitude to my office mates, **Loris, Gregoire,**

## Acknowledgements

---

**Josh** and **Nacho**, for sharing the nice ELG-132 office with me and having good conversations on a range of topics. I will like to extend my gratitude to all the past and present ESL Ph.D. students, including **Ali Pahlevan, Arman Iranfar, Soumya Basu, Benoît Denkinger, Dionisije Sopic, Elisabetta De Giovanni, Artem Andreev, Halima Najibi, Farnaz Forooghifar, Flavio Ponzina, Renato Zanetti, William Simon, Marco Rios, Darong Huang, Alireza Amirshahi, Saleh Baghersalimi, Simone Machetti, Giulio Masinelli, Rafael Morillas, Una Pale, Hossein Taji, Pengbo Yu** and **Silvio Zanolli**, for the great moments and wonderful time spent with them. It is these colleagues that make ESL international and multi-cultural.

This journey would not have been possible without all the new friends that I made during my stay in Lausanne. I will like to thank the **Pakistani-Students Society Lausanne** and all my Pakistani friends here in Switzerland, who were there for me whenever I needed them and for arranging the wonderful Eid BBQ's. I will also grateful to all my friends at **GMU** for arranging the amazing Ramadan Iftars every year. I will take this opportunity to give a shout-out and express my gratitude to my friend **Anas**, who was always there for me and my family, since our arrival in Switzerland.

Furthermore, I will to thank all my family for their continuous support. In particular, I will like to start by expressing my deepest gratitude to my **father Nasir Mahmood Qureshi**, who always wished and dreamed for me to be become a Ph.D. scholar. A dream that he narrowly missed himself, and wished for one of his children to complete. It was he who convinced and motivated me to pursue this journey. I am proud and glad that I could fulfil his dream, and he could see the start of this journey, however, I feel sad and grieved at the same time, as he is no longer with us to witness the completion my Ph.D., one of the most important milestones in my life. Next, I will like to thank my **mother Tallat Nasir**, for all her support and prayers for my success, and taking time to talk to me every single day trying to keep me motivated and going. I will also like thank my brothers **Waqas** and **Haris**, for being very supportive and taking care of the matters back home in Pakistan on my behalf. I also want to thank my **father-in-law Dr. Qamar Mubashir** and **mother-in-law Kulsoom Mubashir** for believing in me and being very encouraging and supportive throughout the journey.

Last but not the least, I will like to say special thanks to my wonderful and amazing wife **Ramsha**, who was very supportive throughout this journey from the very start of making the decision to pursue a Ph.D. I really appreciate her for being very understanding during the course of my Ph.D. It was her who motivated me daily, kept me focused on my research by taking care of our beautiful kids and all the house-hold stuff, and prepared delicious meals and dinners everyday. The effort and contributions she did in enabling me to be focused on my research are beyond words. Moreover, I will to say a big thanks to my two little cute sons, **Arham** and **Affan**, who are the joy of my life, and relieved me of all the work and research related stress everyday, whenever I returned back home. They made this journey really wonderful. It would not have been the same experience without the joyful moments I had with them, and to which I looked forward everyday when coming back home.

*Lausanne, August 22, 2021*

Yasir Mahmood Qureshi

# Abstract

The expeditious proliferation of Internet connectivity and the growing adoption of digital products have transformed various spheres of our everyday lives. This increased digitization of society has led to the emergence of new applications, which are deployed all the way from High Performance Computing (HPC) systems and cloud servers to mobile devices. These new emerging applications like video analytics, autonomous driving, natural language processing, content recommendation, bioinformatics, and genome sequencing, have different performance/energy requirements, and are deployed on a variety of different platforms. To meet the performance and Quality-of-Service (QoS) constraints, cloud servers, and HPC platforms are comprised of many-core processors. Even the processors in today's mobile and edge devices are multi-core. To optimize energy efficiency and performance, a system-level simulator is required. This simulator must be capable of simultaneously executing multi-threaded applications in a full-system environment with a complete operating system (OS), on a heterogeneous many-core system.

In this thesis, I present gem5-X, a system-level simulation platform to optimize many-core heterogeneous compute and memory architectures. Gem5-X exploration and optimization methodology is also proposed to optimize both the compute and memory sub-system for multi-thread applications. Gem5-X extends gem5 with architectural extensions for the compute and memory sub-systems, including in-caching computing accelerator, clustered heterogeneous cores, in-memory computing engine, and 3D stacked High Bandwidth Memory v2 (HBM2). It also adds support enhancements to gem5, including, Virtual Machine (VM) support, profiling support in the simulator using gperf profiler, enhanced checkpointing, and file sharing between the simulated and the host system. Finally, all the extensions and enhancements are fully supported in Full System (FS) mode of gem5-X, enabling booting and running a full Linux stack on top of the simulator, allowing almost any application running in a Linux system to be executed using gem5-X.

To demonstrate the capabilities of gem5-X, various compute-dominated and memory-dominated applications are used as case studies. These state-of-the-art applications include video encoding, video analytics, VMs in the cloud, Binary Neural Networks (BNNs) and Recurrent Neural Networks (RNNs) as compute-intensive workloads. In addition to compute-dominated workloads, memory-dominated applications, including genome sequence alignment based on Next Generation Sequencing (NGS) techniques, and Convolutional Neural Networks (CNNs) are presented as case studies to demonstrate the memory sub-system extensions of gem5-X. Gem5-X exploration

## Abstract

---

methodology is used to optimize architectures for these applications, achieving both performance and energy benefits. All the applications and workloads used in this thesis are case studies to showcase the capabilities of gem5-X. Gem5-X is generic and can be used to optimize and explore architectures for any multi-threaded (or single-threaded) application that can run on a Linux-based OS.

**Keywords:** *Many-core, multi-core, heterogeneous architectures, architecture exploration, simulator, gem5, gem5-X, in-cache computing, in-memory computing, HBM, 3D ICs, SPM, 3D stacked memory, CNN, NGS, genome sequencing, High Performing Computing (HPC), Cloud Computing*



# Résumé

L'augmentation de la connectivité internet et de l'utilisation de produits digitaux ont transformé différentes facettes de notre vie quotidienne. La digitalisation de notre société a fait apparaître de nouvelles applications qui sont déployées sur tout l'éventail de l'infrastructure informatique, des serveurs cloud et centre de Calcul Haute Performance (HPC) jusqu'aux appareils mobiles. Ces applications, telles que l'analyse vidéo, la conduite autonome, le traitement du langage naturel, ou le séquençage du génome, ont des exigences différentes en matière de performances/-énergie et sont déployées sur une large gamme d'appareils. Pour répondre aux contraintes de performances et de qualité de service, les serveurs cloud et les plateformes HPC sont composés de processeurs multicœurs. Même les processeurs des téléphones mobiles et des dispositifs périphériques d'aujourd'hui sont multicœurs. Pour optimiser leur efficacité énergétique et leurs performances, un simulateur du système complet est nécessaire. Ce dernier doit être capable d'exécuter simultanément des applications multithread dans un environnement système complet avec un système d'exploitation (OS) complet, sur un système multicœurs hétérogène.

Dans cette thèse, je présente gem5-X, une plateforme de simulation au niveau système pour optimiser les architectures de calculs et de mémoire hétérogènes à plusieurs cœurs. Gem5-X propose aussi une méthodologie pour l'exploration et l'optimisation à la fois du sous-système de calcul et de la mémoire, pour les applications multithread. Gem5-X étend gem5 avec des extensions architecturales, notamment un accélérateur de calcul en cache, des cœurs hétérogènes groupés, un moteur de calcul en mémoire et une mémoire à bande passante élevée (HBM2) empilée en 3D. Gem5-X apporte également plus de support à gem5, y compris, mais sans s'y limiter, la prise en charge du profilage dans le simulateur à l'aide du profileur gperf. Toutes les extensions et améliorations sont entièrement prises en charge en mode Système complet de gem5-X, permettant de démarrer et d'exécuter une couche Linux complète sur le simulateur et ainsi d'exécuter presque toutes les applications fonctionnant sur un système Linux.

Pour démontrer les capacités de gem5-X, diverses applications dominées par leur demande en calculs et en mémoire sont utilisées comme études de cas. Ces applications de pointes incluent l'encodage vidéo, l'analyse vidéo, les machines virtuelles dans le cloud, les réseaux de neurones binaires (BNN) et les réseaux de neurones récurrents (RNN), l'alignement de séquences génomiques basé sur les techniques de séquençage de nouvelle génération (NGS) et les réseaux de neurones à convolution (CNN). La méthodologie d'exploration Gem5-X est utilisée pour optimiser les architectures pour ces applications, permettant à la fois des avantages en termes de

## Résumé

---

performances et d'énergie.

**Mots-clés :** *Architectures multicœurs, architectures hétérogènes, exploration d'architecture, simulateur, gem5, gem5-X, calcul en cache, calcul en mémoire, HBM, circuits intégrés 3D, SPM, mémoire 3D empilé, CNN, NGS, séquençage du génome, Calcul Haute Performance (HPC), calcul dans le cloud*

# Contents

<b>Acknowledgements</b>	<b>i</b>
<b>Abstract (English)</b>	<b>iii</b>
<b>Résumé(Français)</b>	<b>v</b>
<b>List of Figures</b>	<b>xiii</b>
<b>List of Tables</b>	<b>xix</b>
<b>Acronyms</b>	<b>xxi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 The Power Wall Problem . . . . .	2
1.2 The Memory Wall Problem . . . . .	3
1.3 Thesis Contribution . . . . .	3
1.3.1 The Gem5-X Simulator . . . . .	4
1.3.1.1 Architectural Extensions . . . . .	4
1.3.1.2 Support Enhancements . . . . .	5
1.3.2 Compute-Dominated Architecture Exploration . . . . .	6
1.3.2.1 Architecture Optimization for Video Encoding . . . . .	6
1.3.2.2 Heterogeneous Architecture for Video Analytics . . . . .	7
1.3.2.3 Near-Threshold-Computing (NTC) Servers for VMs in the Cloud . . . . .	8
1.3.2.4 In-Memory Computation of Artificial Intelligence (AI) Work- loads . . . . .	9
1.3.3 Memory-Dominated Architecture Exploration . . . . .	10
1.3.3.1 Genome Sequence Alignment . . . . .	10
1.3.3.2 Scratchpad Memories (SPMs) for CNNs . . . . .	11
<b>2 The Gem5-X Simulator</b>	<b>13</b>
2.1 Introduction . . . . .	13
2.2 Related work . . . . .	15
2.3 Architectural Extensions . . . . .	16

## Contents

---

2.3.1	Compute Sub-System . . . . .	17
2.3.1.1	ISA Extension . . . . .	17
2.3.1.2	Custom Accelerator . . . . .	17
2.3.1.2.1	<i>Memory Mapped Accelerator</i> . . . . .	18
2.3.1.2.2	<i>Custom Instruction Accelerator</i> . . . . .	18
2.3.1.3	BLADE In-cache Computing Engine . . . . .	18
2.3.1.4	Computational Memory . . . . .	21
2.3.1.5	Core Clustering . . . . .	22
2.3.1.6	Heterogeneous Cores . . . . .	22
2.3.1.7	Core Count . . . . .	24
2.3.2	Memory Sub-System . . . . .	24
2.3.2.1	High Bandwidth Memory v2 (HBM2) . . . . .	24
2.3.2.2	ScratchPad Memory (SPM) . . . . .	25
2.3.2.3	Heterogeneous Memories . . . . .	27
2.3.3	Gem5-X Parameters . . . . .	27
2.4	Support Enhancements . . . . .	28
2.4.1	ARMv8 64-bit Full System (FS) Support . . . . .	28
2.4.2	Gperf Profiler . . . . .	29
2.4.3	Enhanced Checkpointing . . . . .	29
2.4.4	9P over Virtio . . . . .	29
2.4.5	Virtual Machines (VMs) . . . . .	29
2.5	Validation . . . . .	30
2.6	HBM2 Model Bandwidth Analysis . . . . .	31
2.6.1	BW Analysis Methodology . . . . .	31
2.6.2	Bandwidth Analysis Results . . . . .	33
2.7	Power Models and Area . . . . .	35
2.8	Architecture Exploration and Optimization Methodology . . . . .	35
2.8.1	Single-Step Architecture Exploration and Optimization Methodology . . . . .	35
2.8.1.1	Phase 1: Application Characterization . . . . .	36
2.8.1.2	Phase 2: Architecture Optimization . . . . .	36
2.8.1.3	Phase 3: Milestones . . . . .	38
2.8.2	Two-Step Architecture Exploration and Optimization Methodology . . . . .	38
2.9	Summary . . . . .	39
<b>3</b>	<b>Compute-Dominated Architecture Exploration</b> . . . . .	<b>41</b>
3.1	Introduction . . . . .	41
3.2	Related work . . . . .	43
3.2.1	Video Encoding . . . . .	43
3.2.2	Video Analytics . . . . .	43
3.2.3	Near-Threshold-Computing (NTC) Servers for VMs in the Cloud . . . . .	44
3.2.4	In-Memory Computation of AI Workloads . . . . .	44
3.2.4.1	Accelerating BNNs with RRAMs . . . . .	44

3.2.4.2	Accelerating LSTMs with Analog In-Memory Computation .	45
3.3	In-Cache Computing Accelerator for Video Encoding . . . . .	46
3.3.1	Video Encoding . . . . .	46
3.3.2	Experimental Setup . . . . .	47
3.3.3	Profiling and Bottlenecks . . . . .	47
3.3.4	Gem5-X Extensions for Video Encoding . . . . .	48
3.3.5	Strategies for Architecture Optimization . . . . .	48
3.3.5.1	Sweeping the Cache Sizes . . . . .	48
3.3.5.2	Acceleration with In-cache Computing . . . . .	49
3.3.6	Architectural Exploration and Results . . . . .	51
3.3.6.1	Low Resolution Video . . . . .	51
3.3.6.2	Medium Resolution Video . . . . .	51
3.3.6.3	High Resolution Video . . . . .	52
3.3.7	Video Encoding on Many-core Multi-application Systems . . . . .	53
3.4	Heterogeneous Architecture for Video Analytics . . . . .	54
3.4.1	Video Analytics Application . . . . .	55
3.4.1.1	Video Analytics Application Structure . . . . .	55
3.4.1.2	Video Encoding . . . . .	56
3.4.1.3	Image classification using CNNs . . . . .	56
3.4.2	Gem5-X Extensions . . . . .	58
3.4.3	Exploration and Optimization Methodology for Real-Time Video Analytics .	59
3.4.4	Architecture Exploration and Optimization of Video Encoding Kernel .	59
3.4.4.1	Experimental Setup . . . . .	59
3.4.4.2	Profiling the Video Encoding Kernel . . . . .	60
3.4.4.3	Architecture Exploration - Results and Analysis . . . . .	60
3.4.5	Architecture Exploration and Optimization of MobileNet Kernel . . . .	62
3.4.5.1	Performance Comparison - CPU vs GPU . . . . .	63
3.4.5.2	MobileNet Scaling and Clustering . . . . .	65
3.4.5.3	Architecture Exploration - Results and Analysis . . . . .	65
3.4.6	Architecture Exploration and Optimization of Video Analytics . . . . .	67
3.4.6.1	Heterogeneous Architecture . . . . .	67
3.4.6.2	Architecture Exploration - Results and Analysis . . . . .	68
3.5	Near-Threshold-Computing (NTC) Servers for VMs in the Cloud . . . . .	72
3.5.1	Overview of the Proposed system . . . . .	73
3.5.1.1	Application Description . . . . .	73
3.5.1.2	Server and Data Center Architecture . . . . .	74
3.5.1.3	QoS Degradation Constraint for VMs . . . . .	74
3.5.2	Server and Data Center Power Models . . . . .	74
3.5.2.1	Cores . . . . .	75
3.5.2.2	Last-Level-Cache (LLC) . . . . .	75
3.5.2.3	Memory Controller, Peripherals, IO and Motherboard . . . . .	75
3.5.2.4	DRAM . . . . .	75

## Contents

---

3.5.3	Gem5-X Extensions . . . . .	76
3.5.4	Experimental Results . . . . .	76
3.5.4.1	Simulation Framework Validation . . . . .	76
3.5.4.2	Server-level Results . . . . .	77
3.5.4.3	Energy Efficiency . . . . .	77
3.5.4.4	Trade-offs Discussion . . . . .	77
3.6	ISA Extensions for In-Memory Computation of AI Workloads . . . . .	78
3.6.1	Accelerating BNNs with RRAMs . . . . .	78
3.6.1.1	CNN Background . . . . .	79
3.6.1.2	Architecture of the Binary Dot Product Engine . . . . .	80
3.6.1.3	Integration within the CPU Pipeline . . . . .	81
3.6.1.4	Performance and Energy Evaluation Methodology . . . . .	83
3.6.1.5	Experimental Results . . . . .	84
3.6.2	Accelerating LSTMs with Analog In-Memory Computation . . . . .	86
3.6.2.1	Analog in-memory computing - Background . . . . .	86
3.6.2.2	DL acceleration with analog in-memory computing . . . . .	87
3.6.2.3	Integration of Analog In-memory Computing (AIMC) Cores . . . . .	88
3.6.2.4	ISA extensions for analog in-memory computing cores . . . . .	89
3.6.2.5	Experimental setup . . . . .	89
3.6.2.6	Analog in-memory computing core parameters . . . . .	89
3.6.2.7	LSTMs - Recurrent Neural Networks (RNNs) . . . . .	90
3.6.2.8	Experimental Results . . . . .	91
3.7	Summary . . . . .	93
<b>4</b>	<b>Memory-Dominated Architecture Exploration</b>	<b>95</b>
4.1	Introduction . . . . .	95
4.2	Related Work . . . . .	96
4.2.1	Architecture for NGS Application Domain . . . . .	96
4.2.2	SPM for CNNs . . . . .	98
4.3	Design Space Exploration for Genome Sequence Alignment . . . . .	98
4.3.1	Introduction . . . . .	98
4.3.2	Sequence Alignment Applications . . . . .	101
4.3.2.1	FM-index . . . . .	101
4.3.2.2	Bowtie2 . . . . .	102
4.3.2.3	BWA-MEM . . . . .	103
4.3.2.4	HISAT2 . . . . .	103
4.3.3	Methodology for Architecture Exploration . . . . .	103
4.3.3.1	Memory Exploration and Optimization . . . . .	105
4.3.3.2	Compute Core Exploration and Optimization . . . . .	105
4.3.3.3	Final Architecture . . . . .	106
4.3.4	Architectural Exploration and Simulation Framework . . . . .	106
4.3.4.1	Experimental Setup . . . . .	106

4.3.4.2	Power Models . . . . .	107
4.3.4.3	Gem5-X Extensions . . . . .	107
4.3.5	Architectural Exploration Parameters . . . . .	108
4.3.6	Architecture Exploration Results . . . . .	108
4.3.6.1	Bowtie2 Architectural Exploration . . . . .	109
4.3.6.1.1	<i>HBM2 vs DDR4</i> . . . . .	109
4.3.6.1.2	<i>HBM2 (no-LLC) vs DDR4</i> . . . . .	111
4.3.6.1.3	<i>Performance-Energy Scaling with Core Count</i> . . . . .	112
4.3.6.1.4	<i>Performance-Energy Scaling with Frequency</i> . . . . .	114
4.3.6.2	BWA-MEM Architectural Exploration . . . . .	116
4.3.6.2.1	<i>HBM2 vs DDR4</i> . . . . .	116
4.3.6.2.2	<i>HBM2 (no-LLC) vs DDR4</i> . . . . .	117
4.3.6.2.3	<i>Performance-Energy Scaling with Core-Count and Frequency</i> . . . . .	117
4.3.6.3	HISAT2 Architectural Exploration . . . . .	119
4.3.6.3.1	<i>HISAT2 Scaling Problem</i> . . . . .	119
4.3.6.3.2	<i>Clustered Architecture</i> . . . . .	120
4.3.6.3.3	<i>HBM2 vs DDR4</i> . . . . .	120
4.3.6.3.4	<i>HBM2 (no-LLC) vs DDR4</i> . . . . .	121
4.3.6.3.5	<i>Performance-Energy Scaling with Core-Count and Frequency</i> . . . . .	121
4.3.6.4	Discussion of the Results . . . . .	123
4.4	SPM Memories for CNNs . . . . .	125
4.4.1	Application . . . . .	127
4.4.2	Profiling of the Application . . . . .	128
4.4.3	SPM architecture for Alexnet CNN . . . . .	129
4.4.3.1	Tiling of the Convolution Layers to Reduce Activations Transfer Size . . . . .	129
4.4.4	Experimental Setup . . . . .	131
4.4.5	Experimental Results . . . . .	131
4.4.5.1	Performance Results . . . . .	131
4.4.5.2	Energy Results . . . . .	133
4.5	Summary . . . . .	135
<b>5</b>	<b>Conclusion and Future Work</b>	<b>137</b>
5.1	Exploration is the Key . . . . .	137
5.2	Future Work . . . . .	140
5.2.1	Short-Term . . . . .	140
5.2.1.1	Consolidation of Architectural Extensions . . . . .	140
5.2.1.2	Custom Accelerators . . . . .	141
5.2.1.3	Dynamic Voltage and Frequency Scaling (DVFS) Support in Gem5-X . . . . .	141

## Contents

---

5.2.1.4	Processing-in-Memory (PIM) . . . . .	141
5.2.2	Long-Term . . . . .	142
5.2.2.1	Integrate 3D-ICE Thermal Simulator with Gem5-X . . . . .	142
5.2.2.2	RISC-V Instruction Set Architecture (ISA) Support . . . . .	142
5.2.2.3	ISA Heterogeneity . . . . .	143
5.2.2.4	Graphics Processing Unit (GPU) Modelling . . . . .	143
5.2.2.5	Parallelization of Gem5-X . . . . .	143
<b>Bibliography</b>		<b>145</b>
<b>Curriculum Vitae</b>		<b>167</b>



# List of Figures

2.1	Gem5-X simulation framework. . . . .	14
2.2	Gem5-X architectural extensions. . . . .	17
2.3	Cache subarray with AND/NOR/XOR bitline computing on values A=0 and B=0. Bitwise operations are performed by first (a) precharging the bitlines, then (b) activating multiple wordlines, thus discharging the bitlines through the connected bitcells. . . . .	19
2.4	The highlighted operands in sets 2 and 4 have matching offsets and set LSBs, and differing set MSBs, indicating that they share bitline logic, but not Local Groups (LGs). . . . .	20
2.5	Simplified block diagram of a generic processor 5-stage pipeline integrating the Computational Memory (CM) in execute stage of the pipeline. . . . .	22
2.6	Multi-core heterogeneous clusters in gem5-X. Each cluster supports multiple cores which can either be in-order or Out-of-Order (OoO) cores for a given cluster with its own cluster cache. If the cluster cache is the Last-Level-Cache (LLC), then the clusters connect directly to the main memory via a cross-bar interconnect. However, if a shared L3 is the LLC then the clusters are connected to the L3 shared cache, which ultimately connects to the main memory. . . . .	23
2.7	3D stacked HBM2 Architecture. . . . .	24
2.8	Private Scratchpad Memory (SPM) architecture in gem5-X. . . . .	25
2.9	Shared SPM architecture in gem5-X, with the SPM being shared between two consecutive Central Processing Unit (CPU) cores. . . . .	26
2.10	Gem5-X platform with architectural exploration parameters. These configurable parameters include the core count, choice of core as either in-order or OoO, ISA extensions, BLADE in-cache computing engine in L1-D cache, cluster cache (L2/LLC), common LLC for all the cores/clusters, and the main memory as either DDR4, HBM2 or both. Caches and memory sizes are configurable as well. . . .	27
2.11	Gem5-X support enhancements. . . . .	28
2.12	Methodology for validation of the simulation platform. . . . .	30
2.13	Methodology for bandwidth (BW) analysis using the STREAM benchmark. . .	32
2.14	BW scaling of HBM2 with respect to the number of channels and core frequency in comparison to DDR4, for both 8 in-order and 8 OoO cores when running the STREAM benchmark. . . . .	33
2.15	BW scaling with frequency for 16 cores. . . . .	34

## List of Figures

---

2.16	BW scaling of HBM2 with core count and core frequency with no LLC. . . . .	34
2.17	Gem5-X single-step multiple-phase architectural exploration methodology. . .	36
2.18	Steps for selection of appropriate architecture. . . . .	37
2.19	Two-Step architectural exploration and optimization methodology. . . . .	39
3.1	Half and Quarter Pixel Finite Impulse Response (FIR) Filter (numbers represent size in pixels). . . . .	49
3.2	Energy and area comparison for low resolution video. . . . .	51
3.3	Energy and area comparison for medium resolution video. . . . .	52
3.4	Energy and area comparison for high resolution video. . . . .	52
3.5	Video analytics application composed of two kernels running concurrently, i.e., video encoding using Kvazaar [1] and image classification/detection using CNNs.	55
3.6	Image partitioning and resize for input image resolution of (a) 300x200 pixels, (b) 640x480 pixels. . . . .	57
3.7	Performance, energy and area comparison for Kvazaar video encoding of 300x200 resolution video. . . . .	60
3.8	Performance, energy and area comparison for video encoding of 640x480 resolution video. . . . .	61
3.9	MobileNet performance and energy comparison. . . . .	64
3.10	Clustered architecture for multiple MobileNet instances. . . . .	65
3.11	MobileNet frames-per-second (FPS) scaling with core count (OoO) for HBM2 (8-Ch) and DDR4 with different number of channels. The horizontal lines are the threshold that should be met for scenarios in Table 3.7. . . . .	66
3.12	MobileNet performance and energy benefit of using HBM2 (8-Ch) over DDR4 with different number of channels for OoO cores. . . . .	66
3.13	Heterogeneous architecture for video analytics. . . . .	67
3.14	Video Analytics FPS for Kvazaar and Mobilenet clusters for 300x200 resolution video for drone navigation when using HBM2, DDR4 and on Hikey960 platform.	68
3.15	Percentage performance and energy benefit when using HBM2 (8-Ch) in comparison to DDR4 with different number of channels and Hikey960 platform for video analytics of 300x200 resolution video for drone navigation. . . . .	69
3.16	Video Analytics for 640x480 resolution video for video surveillance and autonomous cars and Advance Driver-Assistance Systems (ADAS) systems. 10-core Kvazaar cluster is used for video encoding in both scenarios. A 4-core OoO cluster is used for MobileNet in case of surveillance, and three 4-core OoO clusters are used for autonomous cars and ADAS. . . . .	70
3.17	Execution time normalized to QoS limit for different workloads. . . . .	77
3.18	Server efficiency as UIPS/Watt under different core frequencies. . . . .	78

3.19	Block diagram of the proposed Binary Dot-Product Engine (BDPE). Fig. 3.19a illustrates the control logic and the alternative mechanism to perform the bit-wise XNOR of the data and the kernel (both supplied as an input), in case the kernel is not stored in the Resistive Random-Access Memory (RRAM) array. Fig. 3.19b depicts the RRAM-based convolutional block used in this work inspired by [2].	80
3.20	Simplified block diagram of a generic processor pipeline integrating the BDPE.	81
3.21	Format of the new instructions added to the ARMv8 Instruction Set Architecture (ISA) to allow the processor to issue instructions to the BDPE. The <i>opcodes</i> 10000011000 and 11000011000 were re-purposed to specify the custom instructions. <i>rm</i> and <i>rn</i> specify addresses of 64-bit registers; <i>imm6</i> represents a 6-bit immediate; and <i>rd</i> specifies the address of the destination 64-bit register.	82
3.22	Simple example that illustrates the process of storing the most used kernels inside the RRAM and running a BNN using the novel BDPE. . . . .	82
3.23	Results showing the performance improvements due to using BDPE. Figure 3.23a shows the relative reduction on memory accesses during the inference phase of YoloV3 for five runs where the usage of the BDPE varies between 10% and 50%. Figure 3.23b depicts the execution time of the inference phase of YoloV3 and the relative performance improvements varying the usage of the BDPE. . . . .	84
3.24	Results showing the energy efficiency improvements due to using BDPE. Figure 3.24a shows the total energy spent when executing the baseline and five runs where the usage of the BDPE varies between 10% and 50%. Figure 3.24b shows, for the same scenarios the energy spent by the system excluding the main memory (DRAM). . . . .	85
3.25	(a) Proposed design of the Analog In-Memory Computing (AIMC) core. The core communicates with the CPU through digital input and output signals while the array of unit cells performs the Matrix-vector multiply (MVM) operations in an analog fashion. (b) Schematic illustration of the MVM operation across one of the bit lines. . . . .	87
3.26	(a) The proposed processor system with the AIMC core integration. (b) A classic in-order pipe-lined CPU instruction cycle. The AIMC core exists alongside the Arithmetic Logic Unit (ALU) within the execution stage of the CPU. . . . .	88
3.27	RNN with one layer Long-Short-Term-Memory (LSTM) with fully connected dense layer. The weights for both the layers are stored in the AIMC core. . . . .	91
3.28	This figure shows the trends in performance per architecture, per LSTM network size. Figure (a) shows the total run time of the LSTM, figure (b) shows the memory intensity (number of LLC misses per 1000 instructions, or LLCMPKI), and figure (c) shows the total energy. . . . .	92
3.29	Factor improvement in (a) performance and (b) energy statistics with the proposed AIMC-based architecture over CPU+Single-Instruction-Multiple-Data (SIMD) implementation. . . . .	92

## List of Figures

---

4.1	Phases for (a) FM-index and (b) Genome Sequencing Applications. . . . .	102
4.2	Extended architecture exploration and optimization methodology. . . . .	104
4.3	Architectural block diagram of experimental setup in gem5-X. . . . .	106
4.4	Bowtie2 performance and energy benefit of HBM2 vs DDR4 at 2GHz. The bars represent percentage benefits and the points with lines show absolute value. (a) Percentage performance benefits along with sequencing time. (b) Corresponding percentage energy benefits along with absolute energy values. . . . .	110
4.5	Bowtie2 performance and energy benefits of near memory compute HBM2 with no-LLC systems in comparison to DDR4 with LLC systems at 2GHz core frequency. (a) Percentage performance benefits along with sequencing time. (b) Corresponding percentage energy benefits along with absolute energy values. . . . .	111
4.6	Bowtie2 performance scaling with number of cores at 2GHz using HBM2 as main memory. . . . .	113
4.7	Bowtie2 energy scaling with number of cores at 2GHz using HBM2 as main memory. . . . .	113
4.8	Bowtie2 performance scaling with frequency for different core types and LLC size. . . . .	115
4.9	Bowtie2 performance and energy of all configurations surpassing 32 Knights Landing (KNL) cores performance. . . . .	116
4.10	BWA-MEM performance and energy benefits of HBM2 in comparison to DDR4 with LLC of 2MB/8-cores. . . . .	117
4.11	BWA-MEM performance scaling with core-count and frequency. LLC is fixed at 2MB/8-cores. . . . .	118
4.12	BWA-MEM energy scaling with core-count and frequency. LLC is fixed at 2MB/8-cores. . . . .	118
4.13	Clustered architecture with 4-cores per cluster for HISAT2. . . . .	120
4.14	HISAT2 performance and energy benefits of HBM2 in comparison to DDR4 with LLC of 512KB/4-cores. . . . .	121
4.15	HISAT2 performance scaling with core-count and frequency. LLC is fixed at 512KB/4-cores. . . . .	122
4.16	HISAT2 Energy scaling with core-count and frequency. LLC is fixed at 512KB/4-cores. . . . .	123
4.17	Passing activations from one CNN layer to the next through the cache hierarchy. . . . .	126
4.18	Passing activations from one CNN layer to the next through the SPM. . . . .	126
4.19	A multi-core CNN system with pipelined inferences. . . . .	127
4.20	Convolution in a CNN layer. . . . .	130
4.21	Alexnet performance comparison of cache-only and cache-SPM system. Total memory size, comprising of L1-D, LLC and SPM, is also compared for the two systems. . . . .	132
4.22	Performance and total memory size comparison of cache-only and cache-SPM system with an LLC of 2MB and 512KB, respectively. . . . .	132
4.23	Performance scaling of Alexnet in a SPM-based system with scaling of both L1-D and LLC size. . . . .	133

4.24 Alexnet energy comparison of cache-only systems with SPM and cache system.	134
4.25 Alexnet Energy-Delay-Product (EDP) comparison of cache-only systems with SPM and cache system. . . . .	134



# List of Tables

2.1	Implemented HBM2 architecture. . . . .	25
2.2	Validation error of gem5-X when compared against a real JUNO platform. . . .	31
3.1	Initial architecture for video encoding. . . . .	47
3.2	FIR filter speed-up for different video resolutions when using BLADE. Application acceleration and energy reduction for different video resolutions, when using BLADE in-cache computing engine (for both FIR filter and Sum-of-Absolute-Difference (SATD) block) with ARM in-order cores in comparison to ARM in-order cores with NEON SIMD. . . . .	50
3.3	Speed-up using HBM2 instead of DDR4. . . . .	53
3.4	Video Analytics Application Scenarios. . . . .	56
3.5	Time, core power (at 2GHz) and memory power for video encoding of 300x200 resolution video, while meeting 24 FPS requirement. . . . .	62
3.6	Time, core power (at 2GHz) and memory power for video encoding of 640x480 resolution video, while meeting 24 FPS requirement. . . . .	62
3.7	MobileNet classification rate for various application scenarios. . . . .	63
3.8	Time, core cluster power (at 2GHz) and memory power for video analytics case study scenarios. . . . .	71
3.9	Near-Threshold-Computing (NTC) server and Cavium ThunderX QoS analysis in terms of execution time (s). . . . .	76
3.10	Profile of a trimmed version of YoloV3 regarding the frequency of use of each convolutional kernel during the inference phase. . . . .	80
3.11	Total energy spent by the BDPE and the CPU during the inference phase of YoloV3 XNOR-Net. . . . .	85
3.12	Gem5-X FS Mode System Configurations. . . . .	90
3.13	AIMC core performance and energy figures. . . . .	90
3.14	RNN Network Parameters. . . . .	91
4.1	Parameters for architectural exploration and optimization. . . . .	108
4.2	LLC Sizes and scaling with number of cores. . . . .	108
4.3	Initial profiling architecture. . . . .	129
4.4	Alexnet profiling on a cache-only system in gem5-X. . . . .	129
4.5	Parameters for SPM-based architecture. . . . .	131

**List of Tables**

---

4.6 Alexnet L1-D miss rate on SPM-based system in gem5-X. . . . . 133

5.1 case study Applications vs. Gem5-X Architectural Extensions. . . . . 140



# Acronyms

<b>3D-ICE</b>	3D Interlayer Cooling Emulator
<b>ACL</b>	ARM Compute Library
<b>ADAS</b>	Advance Driver-Assistance Systems
<b>ADC</b>	analog-to-digital converters
<b>AI</b>	Artificial Intelligence
<b>AIMC</b>	Analog In-Memory Computing
<b>ALU</b>	Arithmetic Logic Unit
<b>ASIC</b>	Application-Specific Integrated Circuit
<b>AWS</b>	Amazon Web Services
<b>BDPE</b>	Binary Dot-Product Engine
<b>BNN</b>	Binary Neural Network
<b>BW</b>	bandwidth
<b>BWT</b>	Burrows-Wheeler Transform
<b>CISC</b>	complex instruction set computer
<b>CM</b>	Computational Memory
<b>CNN</b>	Convolutional Neural Network
<b>CONV</b>	convolutional
<b>CPU</b>	Central Processing Unit
<b>DAC</b>	digital-to-analog
<b>DL</b>	Deep Learning
<b>DNN</b>	Deep Neural Network

## Acronyms

---

<b>DSL</b>	domain-specific language
<b>DTB</b>	device tree binary
<b>DVFS</b>	Dynamic Voltage and Frequency Scaling
<b>EDP</b>	Energy-Delay-Product
<b>FIR</b>	Finite Impulse Response
<b>FPGA</b>	Field-Programmable Gate Array
<b>FPS</b>	frames-per-second
<b>FS</b>	Full System
<b>GFM</b>	Graph FM-Index
<b>GPU</b>	Graphic Processing Unit
<b>HBM2</b>	High Bandwidth Memory v2
<b>HEVC</b>	High Efficiency Video Coding
<b>HGFM</b>	Hierarchical Graph FM-Index
<b>HMC</b>	Hybrid Memory Cube
<b>HPC</b>	High Performance Computing
<b>IC</b>	Integrated Circuit
<b>IoT</b>	Internet-of-Things
<b>ISA</b>	Instruction Set Architecture
<b>KNL</b>	Knights Landing
<b>L1-D</b>	L1 data
<b>L1-I</b>	L1 instruction
<b>LG</b>	Local Group
<b>LLC</b>	Last-Level-Cache
<b>LSTM</b>	Long-Short-Term-Memory
<b>MAE</b>	Mean Absolute Error
<b>mAP</b>	mean Average Precision
<b>ML</b>	Machine Learning

<b>MLP</b>	Multi-Layer Perceptron
<b>MSB</b>	Most Significant Bit
<b>MSHR</b>	miss-status-holding-registers
<b>MVM</b>	Matrix-vector multiply
<b>NGS</b>	Next Generation Sequencing
<b>NN</b>	Neural Network
<b>NPU</b>	Neural Processing Unit
<b>NTC</b>	Near-Threshold-Computing
<b>NVM</b>	Non-volatile Memory
<b>OOB</b>	out-of-the-box
<b>OoO</b>	Out-of-Order
<b>OS</b>	operating system
<b>PCM</b>	Phase-Change Memory
<b>PE</b>	Processing-Elements
<b>PIM</b>	Processing-in-Memory
<b>PPA</b>	performance, power and area
<b>pthread</b>	POSIX thread
<b>QoS</b>	Quality-of-Service
<b>RISC</b>	reduced instruction set computer
<b>RNN</b>	Recurrent Neural Network
<b>ROI</b>	Region-Of-Interest
<b>RRAM</b>	Resistive Random-Access Memory
<b>SATD</b>	Sum-of-Absolute-Difference
<b>SE</b>	System Emulation
<b>SIMD</b>	Single-Instruction-Multiple-Data
<b>SPM</b>	Scratchpad Memory
<b>SVHN</b>	Street View House Numbers

## Acronyms

---

<b>TDP</b>	Thermal Design Power
<b>TSV</b>	Through-Silicon-Via
<b>UIPS</b>	User Instructions per Second
<b>UTBB</b>	Ultra-Thin Body and Buried Oxide
<b>VM</b>	Virtual Machine
<b>WA</b>	workload automation
<b>WFM</b>	wait-for-memory
<b>WPP</b>	Wavefront Parallel Processing

# 1 Introduction

The past two decades have seen a rapid expansion and reach of internet and online services. From 2000 to 2019, internet penetration has increased more than 8 times from 6.3% to over 53% of the world population [3, 4]. In addition, more than 93% of the world population has internet connectivity as of 2019 [5]. This increased internet connectivity has enabled various online services which had a significant impact on our everyday lives. From weather reports in the morning, daily navigation and route planning on our way to work, social networking, video streaming, web search, online financial services, e-commerce and the list goes on, we are almost continuously connected to the Internet.

The world's connectivity infrastructure was put to a real test with the outbreak of the novel coronavirus (COVID-19) [6]. The abrupt closure of workplaces, educational institutes, shops and shopping centres to contain the spread of the virus had a drastic impact on the lives of people. However, due to the internet connectivity and availability of various online services, many of the effects were mitigated with workplaces and educational institutes switching instantly to remote working and remote learning, respectively. Online shopping became the new norm for many, with e-commerce representing 17% of the global retail trade [7]. With people confined to their homes during lockdown, online entertainment and video streaming services also saw a surge, with 72% weekly increase of Netflix users in the US [8].

With the emergence of the Internet-of-Things (IoT), we will be connected more than ever to the Internet and rely even more on the online services [9], in addition to the increased internet usage and digitization during the COVID-19 pandemic. IoT has evolved from being a buzz word in early 2000s to actual products and applications like smart cities [10], smart supply chain management for real-time tracking [11], connected and autonomous cars [12], e-health via remote monitoring and wearable sensors [13], and geographically distributed environment and weather sensors [14], just to mention a few. To reduce the communication overhead and save on communication energy, computing the collected data on the edge sensor nodes is preferable. However, these edge IoT and mobile devices are constrained in energy as well as computational power. Therefore, with the emergence of increasingly complex applications, some of these applications or a part of

them have to be computed on the cloud servers to which the edge node is connected. Hence, we have emerging applications which are deployed all the way from edge and mobile platforms to cloud servers and High Performance Computing (HPC) systems. Thus, it is essential to optimize performance and minimize the energy consumption of both the connected energy constrained edge devices as well as high-end servers. These power hungry data centres consume around 1% (200 TWh) of the global energy [15]. To put into perspective, this is half of the electricity used for transport worldwide [16].

### 1.1 The Power Wall Problem

Moore's Law [17] and the Dennard Scaling Model [18] describe the performance scaling in the semiconductor industry for decades. According to Moore's Law the number of transistors per unit area double every 18 to 24 months. Dennard scaling, which is the scaling of transistor size and voltage by the same factor, has helped chip designers maintain the power budget of the chip, i.e., constant power per unit area, despite having more transistors per unit area. However, Dennard scaling ignored the effects of leakage current and threshold voltage, due to which there is an exponential dependence of leakage power on threshold voltage. This has constrained the threshold and supply voltage scaling, resulting in increased power density of the chip with each technology node [18]. Consequently, not all the transistors on the chip can be powered *ON* at the same time due to constrained Thermal Design Power (TDP) budget of the chip, because of the physical limitation of cooling technologies and packaging. This is commonly referred to as the Dark Silicon problem [19, 20, 21]. Hence, many-core scaling faces a *Power Wall* problem.

Heterogeneous multi-core architectures can help in pushing back and alleviating the *Power Wall* problem [22]. Heterogeneity enables applications or different kernels of the application to be mapped on different computing resources, depending on the performance requirements and energy constraints. Consequently this improves performance as well as energy efficiency [23]. The heterogeneity can be of various forms including functional heterogeneity or micro-architectural heterogeneity. Functional heterogeneity refers to heterogeneity at the processing engines or core level, with different functional behavior like general-purpose processors, Graphic Processing Units (GPUs) and special purpose accelerators [22]. Micro-architectural heterogeneity refers to processors that have cores with the same Instruction Set Architecture (ISA), but different performance-power characteristics. These processors have a combination of simple energy efficient in-order cores along with high-performance Out-of-Order (OoO) cores. Tasks and applications are allocated according to their performance requirements, to have the best energy efficiency as well as performance. The big.LITTLE architecture from ARM is an example of micro-architectural heterogeneous processor [24]. Deeply heterogeneous architectures combine all the forms of heterogeneity discussed above (i.e., functional and micro-architectural heterogeneity) to have a complete heterogeneous system. However, there are various challenges for the heterogeneous computing, including communication between different compute cores, programming, integration, scheduling and scalability. Furthermore, design space exploration, in the quest to find the best performance and energy efficient heterogeneous architecture, using a

fast, validated and full system simulator is also a challenge in designing heterogeneous systems.

Besides heterogeneity, Near-Threshold-Computing (NTC) is a promising approach in mitigating the Dark Silicon problem [22]. In NTC, performance is sacrificed for energy efficiency, by operating the transistors at a voltage slightly greater than the transistor threshold voltage, thus bringing down power consumption due to the quadratic relation between dynamic power and operating voltage [25].

## 1.2 The Memory Wall Problem

In addition to the Dennard scaling and the Dark Silicon problem on the compute side, the second problem limiting the performance and energy efficiency of the computing systems is the *Memory Wall* problem. The *Memory Wall* problem refers to limited I/O and memory bandwidth available to the processing cores on the chip [26]. The emerging applications, whether it be a Convolutional Neural Network (CNN) deployed on the edge device or video analytics in the cloud or a genome sequence alignment running on a HPC system, all face the *Memory Wall* problem. These applications either have large working data-sets which do not fit on the on-chip caches, or memory access patterns that are unpredictable, increasing the cache miss-rate. This leads to deterioration in the overall performance and energy efficiency because of the high number of memory accesses. Cache hierarchies have hidden the large latencies associated in accessing the DRAM memory, but this does not solve the *Memory Wall* problem for applications with large data or random memory access patterns. Emerging memory technologies like 3D die-stacked memories [27] are explored as alternatives to the traditional DDR memories with magnitude of improvement (10x) in bandwidth [28] and hence, help in bringing down the *Memory Wall*. Moreover, the emerging Non-volatile Memories (NVMs) [29] are promising in alleviating the *Memory Wall* problem by enabling to perform the computations within the memory [30].

## 1.3 Thesis Contribution

Today's emerging applications are restricted in performance and energy efficiency by both the *Power Wall* and the *Memory Wall*. To mitigate the effects of both the walls, this thesis proposes and demonstrates the exploration of different heterogeneous compute and memory architectures to achieve performance- and energy-optimized architectures, both for the compute and memory sub-systems. Architecture *Exploration* is the key enabler for performance and energy efficient architectures. It is this exploration that enables to explore new ideas and analyse if they are beneficial in improving the system performance and energy efficiency. Therefore, I present in this thesis a system-level architectural simulation platform, gem5-X, to explore novel architectures for improved performance and energy efficiency across all levels of computing platforms, from the edge sensors to high end servers in the cloud. The major contributions of this thesis are summarized as follows:

- I present and develop the gem5-X architectural simulator, which is based on gem5 simulator [31], with a variety of architectural extensions and support enhancements completely integrated and compatible with each other out-of-the-box (OOB) in Full System (FS) mode.
- To demonstrate the capabilities of gem5-X architectural simulator, to explore and optimize architectures both for the compute and memory sub-systems, various emerging and state-of-the-art applications are used as case studies. However, gem5-X is generic and can be used to explore and optimize architectures for any application, other than the case studies as well.
- For the compute-dominated case study applications, I explore and optimize architectures, enabled by gem5-X architectural extensions including heterogeneous computing, in-cache computing accelerators, NTC servers and Computational Memories (CMs) integrated within the Central Processing Unit (CPU) cores.
- I develop and propose to use 3D stacked High Bandwidth Memory v2 (HBM2) and Scratchpad Memories (SPMs) models in gem5-X to alleviate the *Memory Wall*, when optimizing the architecture for memory-dominated case study workloads.

In this section, the chapter-wise contributions of the thesis will be briefly presented, in the architectural quest of achieving performance- and energy-optimized architectures for state-of-the-art emerging applications of today.

### 1.3.1 The Gem5-X Simulator

To explore many-core heterogeneous compute and memory architectures, a system-level simulator is required for fast design space exploration and optimization. The gem5-X simulator is proposed and presented in this thesis with two main system-level exploration goals. In particular, it is used for architectural exploration and for optimization of heterogeneous many-core systems in FS mode with full Linux stack. Gem5-X is "*a gem5-based full-system simulator with architectural eXtensions*". Gem5-X extends gem5 [31] with architectural extensions and support enhancements. Gem5-X is presented and discussed in detail in Chapter 2 of this thesis.

#### 1.3.1.1 Architectural Extensions

Architectural extensions in gem5-X are added for both the compute and the memory sub-systems and are supported in FS mode. These extensions enable the support for different general purpose compute cores, including ARMv8 in-order and OoO cores, with the number of cores scaling up to 256. Accelerators can be added and integrated with ease in gem5-X at different levels in the compute sub-system. They can be integrated within the CPU pipeline, or added within the caches (as in-cache computing engine), or as a separate compute engine. Gem5-X ISA extensions enable



seamless integration of the accelerators within the compute sub-system. Furthermore, gem5-X enables heterogeneity and is able to simulate different core types (including in-order and OoO) along with different accelerators in the same system together, to help alleviate the *Power Wall* problem. In addition to heterogeneity, gem5-X also supports core clustering, with each cluster having its own shared cache. Clustering enables different independent applications or kernels of the same application with very little data sharing to be assigned to different clusters, which alleviates the cache thrashing problem in the shared cache.

To alleviate the *Memory Wall* problem, gem5-X supports various architectural extensions to have an efficient and optimized memory sub-system. It supports 3D stacked HBM2 [32], based on 3D stacked DRAMs. This 3D stacking of DRAM banks is enabled by Through-Silicon-Vias (TSVs). Gem5-X also supports traditional memories like DDR4, alongside HBM2 in the same system. This enables one to have a heterogeneous memory sub-system, with data from different applications or different kernels of the same application allocated to different memories according to bandwidth (BW) requirements of the respective application or kernels. In addition to supporting a heterogeneous memory sub-system with regards to the main memory, the gem5-X simulation platform also supports SPM or Software-Programmable Memories, tightly coupled to the computing cores. SPMs are at the same level as an L1 cache, but with the data managed completely in software, as opposed to hardware controlled caches. Hence, they are useful when the programmer wants the data to be readily available near the compute core and avoid it being evicted in the hardware controlled cache. SPMs can either be private or shared between compute cores. Shared SPMs are particularly useful for core-to-core communication or data transfer, bypassing the entire cache hierarchy. Communication between cores was one of the challenges that was described earlier for heterogeneous computing systems. Hence, shared SPMs also help to resolve that. Gem5-X supports both private and shared SPMs.

#### 1.3.1.2 Support Enhancements

Gem5-X has various support enhancements that enable it to include the architectural extensions discussed above both for the memory and compute sub-systems. One of the most prominent features and support enhancements of the gem5-X simulation platform is that it supports all the architectural extensions, whether in the compute or memory sub-system, in FS simulation mode. FS mode enables to boot an entire Linux based operating system (OS). Gem5-X comes with Ubuntu 16.04 OS, up-gradable to Ubuntu 18.04 OS disk image. FS mode allows for almost any multi-threaded application running on top of a modern Linux to be ported and simulated in gem5-X, thus, facilitating a multi-threaded and multi-core architectural exploration for that application utilizing different architectural extensions. Furthermore, the *gperf* [33] profiler support in gem5-X allows fast profiling and seamless analysis of multi-threaded applications within the simulator. Moreover, the support for Virtual Machines (VMs) using the Linux LXC containers is also added to allow simulating cloud workloads using VMs. Furthermore, an architectural exploration and optimization methodology is proposed to obtain a performance-energy optimized architecture for applications within gem5-X. Checkpointing has been enhanced

## Chapter 1. Introduction

---

in gem5-X and support for file sharing between the host and simulated system has been added, for faster simulation turnaround time.

The gem5-X simulator has been validated for both ARMv8 64-bit in-order and OoO cores with a validation error of up to 4% against a commercially available ARM JUNO platform [34]. Moreover, a detailed BW analysis and validation of the HBM2 memory model in gem5-X is presented in this thesis, in comparison to the DDR4 model. The BW analysis also looks into the scaling of HBM2 BW with varying number of memory channels.

Gem5-X is generic and can be used in to simulate and optimize any multi-threaded application. To demonstrate the architectural exploration methodology and the utilization of various architectural extensions and support enhancements in gem5-X, different emerging applications deployed in real world scenarios are used as case studies.

The case study applications are broadly classified as either compute-dominated or memory dominated, depending on whether the application is compute intensive or memory intensive. Therefore, the architecture exploration is also either compute-dominated or memory-dominated. Compute-dominated architectural exploration does not imply that the memory sub-system will not be optimized, but it means that the main focus will be on the compute sub-system for compute intensive workloads, as most of the performance and energy benefits will come from it. However, optimization of the memory-subsystem for these applications will also be looked at, but the improvements will be lower. Same applies to memory-intensive workloads, where the most performance and energy benefits will be achieved by improving the memory sub-systems. However, an efficient compute sub-system will still be required and explored for these workloads.

### 1.3.2 Compute-Dominated Architecture Exploration

To showcase the gem5-X capabilities and demonstrate its architectural extensions, I will optimize and explore architectures for state-of-the-art emerging compute intensive applications in this thesis. The applications used as case studies represent application domains, and the optimized architectures obtained for each application domain can be considered as architecture templates for other applications in that domain. The compute-dominated case study applications include video encoding, video analytics, VM banking workloads, Binary Neural Networks (BNNs) and Long-Short-Term-Memory (LSTM) based Recurrent Neural Networks (RNNs). In this section different compute intensive applications and the respective gem5-X extensions used will be presented briefly. Chapter 3 of the thesis presents in detail the architecture exploration and optimization for these compute-dominated workloads.

#### 1.3.2.1 Architecture Optimization for Video Encoding

In 2018, 58% of the total down stream internet traffic was video streaming, as presented by [35]. Furthermore, amid the COVID-19 pandemic and lockdowns imposed in various countries

around the world, people were confined to their houses. Online video content was one of the main entertainments for both adults and children alike, with a 72% weekly increase of Netflix users in the US [8]. Hence, there is an increasing need for a performance-energy-optimized video encoding application, as the online videos are encoded prior to transmission to reduce BW utilization. Therefore, video encoding is used as one of the case study applications to demonstrate gem5-X capabilities. Video encoding targets the video processing application domain. Kvazaar [1], an open source state-of-the-art High Efficiency Video Coding (HEVC) application, is used for video encoding.

Kvazaar is first profiled, both on the ARM JUNO platform [34] as well as in gem5-X. Valgrind [36] and gperf [33] profilers are used in JUNO and gem5-X, respectively. The Finite Impulse Response (FIR) filter and Sum-of-Absolute-Difference (SATD) blocks in the motion estimation kernel of video encoding are found to be the dominating computationally intensive execution blocks. The cache utilization is also found to be high, with around 5% L1-D cache miss rate, indicating high data locality in the cache.

BLADE, an in-cache computing accelerator [37, 38], is proposed to accelerate both the FIR filter and SATD blocks of the video encoding application, by enabling computation directly within the cache. BLADE is implemented and integrated in the L1-D cache of the CPU cores in gem5-X. To enable BLADE to be used in the gem5-X FS mode, the ARMv8 ISA [39] is extended using the reserved opcodes, to implement new BLADE instructions. These new instructions are used within the application via in-line assembly calls in C/C++.

Videos of three different resolutions are encoded to determine the performance and energy benefits of using the BLADE in-cache accelerator for video encoding. Videos of 1920x1080, 416x240 and 176x144 pixels, corresponding to high, medium and low resolutions, are encoded with Kvazaar. These resolutions are representative of user playback devices from a low end mobile to a high resolution TV set. Results demonstrate that accelerating both the FIR filter and the SATD block using BLADE leads to 15% performance benefits, 76% energy savings and 31% less area when compared to a system architecture without BLADE.

#### 1.3.2.2 Heterogeneous Architecture for Video Analytics

Real-time video analytics application, also referred to as the emerging "killer-app" [40], is deployed across a variety of computing platforms, from low power edge devices to high performance cloud servers. It is used in different emerging domains including video surveillance for security and safety [41, 42], autonomous drone navigation for drone rescue missions and parcel deliveries [43], and autonomous cars equipped with Advance Driver-Assistance Systems (ADAS) [44].

Video analytics is composed of two main kernels, video encoding and image classification and detection, running simultaneously in parallel with each other. For a seamless user experience, video encoding has to meet the 24 frames-per-second (FPS) Quality-of-Service (QoS) constraint, whereas image classification has real-time constraints. These real-time constraints for the image

## Chapter 1. Introduction

---

classification are different in different use cases. Three case study scenarios of surveillance, drone navigation and autonomous cars with ADAS for video analytics will be used in this thesis. Kvazaar, as discussed in the previous section, is used for the video encoding kernel. MobileNet [45], a CNN-based image classifier, is used as the image classification kernel of video analytics.

A two-step optimization and exploration methodology is used for architecture exploration of the video analytics application. Step 1 focuses on architecture optimization for individual kernels. Step 2 combines all the architecture optimizations for different kernels on a single platform with all kernels co-allocated together to have a complete application. The complete architecture is then optimized with all the kernels executing together.

Using the two-step architecture exploration methodology, a clustered heterogeneous system for the video encoding application with 3D stacked HBM2 memory is proposed. ARMv8 in-order cores along with BLADE in-cache computing engine are proposed for the video encoding kernel. For the MobileNet CNN, I propose ARM OoO cores as they are found to be the most performance and energy efficient for it. Hence, ARM in-order cores with BLADE and ARM OoO cores, both in separate compute clusters, are proposed for the complete video analytics. HBM2 is also proposed and utilized to relieve any memory bottlenecks in the system. Results show that using heterogeneous cores along with HBM2 gives 30% performance speed-up, 54% energy savings and 43% area benefit, while meeting the target performance constraints of the application.

### 1.3.2.3 Near-Threshold-Computing (NTC) Servers for VMs in the Cloud

Due to the increase in the number of applications and services hosted in the cloud one cannot turn away from the importance of cloud computing. Video streaming services, online banking, cloud storage services, web hosting services, just to name a few, are all hosted in the cloud. Hence, the cloud data centers are increasing in the world, with the increase in these cloud based applications and services. Therefore, I target to explore architectures to maximize the performance and energy efficiency of cloud servers.

Server processors are generally multi-core, and due to the Dennard scaling have a Dark Silicon problem, as was discussed earlier in Section 1.1. NTC can help in overcoming these power bottlenecks, by operating the cores at a voltage slightly higher than the transistor threshold. Exploiting the performance-power trade-offs in NTC, it will be demonstrate that the strict QoS constraints can be met for the cloud servers. The Ultra-Thin Body and Buried Oxide (UTBB) in 28nm FD-SOI technology is used for NTC, in contrast to traditional bulk CMOS technology.

Linux LXC containers based VMs running synthetic workloads similar to that of real banking applications, are used as case study for cloud server workloads. These workloads perform matrix multiplication and manipulation, representing real world financial analysis. Gem5-X VM enhancement is used to simulate VMs within the system. Furthermore, for power analysis of an NTC server, an accurate power model for the UTBB FD-SOI process technology in NTC servers, proposed in [25, 46] will be utilized. Moreover, I present a performance validation

done against real servers (Intel x86 and ARM64), and propose NTC servers with improved performance. Finally, the Dynamic Voltage and Frequency Scaling (DVFS) setup for NTC servers is investigated along with energy and performance trade-offs with regards to QoS requirements.

#### **1.3.2.4 In-Memory Computation of Artificial Intelligence (AI) Workloads**

Artificial Intelligence (AI) and Machine Learning (ML) algorithms and systems have a great influence in our day-to-day life, with these systems being deployed from low power edge sensors and devices to high performance servers in the cloud and HPC platforms. Be it a search for something of interest online, route navigation and recommendation, or online translation from one language to another, these AI based systems are becoming a part of our lives. These algorithms are not only computationally intensive, but also have large working data-sets and are hence compute and memory intensive.

To have an optimized architecture for AI and ML applications, I propose to optimize architectures from both compute and memory perspectives simultaneously, and hence, use in-memory computation, also referred to as CM. CM improves both performance and energy efficiency by enabling multiple compute operations to be performed in parallel in Single-Instruction-Multiple-Data (SIMD) way, within the memory. Furthermore, avoiding the data movements from memory to compute core, saves both time and energy and therefore, leads to better overall performance and energy reduction. Gem5-X enables the modeling of the CM engine as an accelerator and supports integrating it within the execution stage of the CPU pipeline. To use the integrated CM engine in FS mode, gem5-X ISA extensions are used to extend the ARMv8 ISA [39], using the reserved opcodes to have new instructions for the CM engine. To demonstrate the CM engine architectural extension in gem5-X, two case study workloads are used, targeting two different technologies for the CM engine.

First, I propose to use Resistive Random-Access Memory (RRAM) as Binary Dot-Product Engine (BDPE) to be used in BNNs [47, 48]. BNNs are suitable for energy and resource constrained edge devices as both the input and the weights of the convolutional layers are binary. Therefore, the convolutions in BNNs are the bit-wise XNOR of the input and the kernel followed by a bitcount. YoloV3 [49] CNN for object detection is converted to a simplified BNN version to demonstrate the RRAM based BDPE. YoloV3 BNN is then profiled to identify the most frequently used kernels. These kernels are stored in the RRAM which is integrated in the CPU pipeline. Using the RRAM-based BDPE, a performance benefit of 11.3% is achieved along with energy savings of 7.4% for the YoloV3 BNN.

Then, I present an Analog In-Memory Computing (AIMC) engine to accelerate RNNs based on LSTM [50]. The AIMC engine is integrated into the execution stage of the CPU pipeline, like that of BDPE for BNNs. AIMC facilitates parallel analog computation, thus reducing the computational complexity of LSTMs. A non-volatile Phase-Change Memory (PCM) based AIMC core [51] for LSTMs is proposed. Performance and energy benefits of up to 12.4x

and 12.3x, respectively, are achieved, in comparison to a baseline system without in-memory computation.

### 1.3.3 Memory-Dominated Architecture Exploration

The applications deployed on a computational platform, either one on an HPC server or on an edge device, can be compute or memory intensive. For compute-dominated workloads, optimizing the compute-system has the most impact on improving the performance and energy. Likewise, for memory intensive workloads, optimizing the memory sub-system to overcome the Memory Wall, has the most influence in increasing the performance and reducing the energy consumption of the overall system. To demonstrate the capabilities of the gem5-X simulation platform in optimizing the memory sub-system, memory intensive applications including genome sequence alignment and CNNs are used as case studies. Chapter 4 of the thesis discusses in detail the architecture exploration and optimization for these memory-dominated workloads. In this section, the two case study memory-intensive applications and the respective gem5-X extensions used will be presented briefly.

#### 1.3.3.1 Genome Sequence Alignment

The importance of genome sequencing cannot be overlooked due to the impact it has had in the areas of bioinformatics, food microbiology and drug discovery [52, 53, 54]. It has had a central role during the COVID-19 pandemic [6], right from the development of test kits and vaccines to the tracings of different variants worldwide. Genome sequencing [52, 55], a memory bounded workload, is the process of determining the DNA sequence or the order of bases (As, Cs, Gs, and Ts) making up the organism's genome. Next Generation Sequencing (NGS) is a high-throughput genome sequencing method, typically deployed on HPC architectures which are extremely power hungry and high performance.

NGS applications have a pointer-chasing nature which comes from the full-text indexing strategies they use, such as the FM-index, based on the Burrows-Wheeler Transform (BWT) [56, 57] for fast sequence alignments. Three different state-of-the-art and widely used NGS applications, namely, Bowtie2 [58], BWA-MEM [59] and HISAT2 [60], are used as case studies for design space exploration of NGS applications. The gem5-X architectural exploration methodology is extended for the NGS application domain, for optimal performance and energy. Using this methodology I propose the use of 3D stacked HBM2, instead of DDR4, along with ARMv8 cores, as a replacement for HPC class state-of-the-art Intel Xeon Phi 7210 Knights Landing (KNL) [61], and demonstrate that fewer ARM in-order as well OoO cores can outperform a higher number of KNL cores for NGS. An overall performance improvement of 68% and energy savings of 71% are obtained thanks to the use of HBM2 instead of DDR4. I also investigate the optimization of the cache sub-system and demonstrate that in some cases, HBM2 without any Last-Level-Cache (LLC) can outperform traditional memory hierarchies with caches and DDR4 for sequence alignment, resulting in energy savings as well. Moreover, it is shown that by

using frequency scaling one can achieve up to 59% and 61% energy savings for ARM in-order and OoO cores, respectively. Lastly, I demonstrate that many ARMv8 in-order cores at 1.5GHz match the performance of fewer OoO cores at 2GHz, while attaining 4.5x energy savings.

### 1.3.3.2 Scratchpad Memories (SPMs) for CNNs

CNNs gained popularity and became the state-of-the-art de-facto standard for image processing, since the Alexnet [62] CNN was unveiled in 2012. These CNNs are extensively used in everyday imaging tasks on our mobile devices, as well as in the cloud, in applications like video analytics for surveillance. CNNs are one of the workloads that can be considered both compute intensive as well as memory intensive. Previously, the MobileNet CNN was considered as a compute-dominated workload in the context of video encoding, as in Section 1.3.2.1. However, CNNs are also very memory intensive, due to the large number of kernel weights and biases, as well as transfer of activations between different CNN layers. Therefore, I will optimize the architecture for CNNs for the memory sub-system.

Activations from one CNN layer to another in a multi-threaded CNN, with each layer on a different compute core, need to traverse the whole cache hierarchy. This is detrimental, both in terms of performance and energy, as the shared cache becomes the memory bottleneck. The shared SPM extension in gem5-X can alleviate this memory bottleneck with activations being passed between two layers allocated on consecutive cores, through the shared SPM between two consecutive cores. Thus the activation transfer bypasses the cache hierarchy. Alexnet CNN is used as a case study to demonstrate the use of a shared SPM architecture in gem5-X. To reduce the memory footprint and allow for the activations to fit in a physically feasible SPM size, the convolutions in the CNN are tiled. The results show that an SPM-based architecture for CNNs activation transfer is 1.85x faster and 13% energy efficient, as compared to a baseline system with caches only.

To summarize, the *Power Wall* and the *Memory Wall* are the two main hurdles, in achieving the performance and energy efficiency for today's emerging applications. Heterogeneous compute and memory technologies are proposed to overcome these walls. However, there are various challenges in designing and exploring heterogeneous compute and memory architectures, including design-space exploration, integration and executing all of it in a FS environment. In this thesis, I present and propose gem5-X architectural simulator with variety of architectural extensions both for the compute and memory sub-systems, to explore heterogeneous architectures along with emerging memory technologies in a realistic FS setup. Both compute and memory intensive workloads are used as case studies to demonstrate architectural exploration and optimization with various extensions and support enhancements enabled by gem5-X.





## 2 The Gem5-X Simulator

### 2.1 Introduction

The rapid growth of online services and increasing adoption of digital products have influenced all spheres of our lives. This increased digitization of the society has led to the emergence of new applications, which are deployed all the way from High Performance Computing (HPC) systems and cloud servers to mobile devices. Consequently, this situation has led to data centres consuming around 1% (200 TWh) of the global energy in 2019 [15]. These new emerging applications like video analytics [40], autonomous driving [44], natural language processing [63], content recommendation [64], bioinformatics [65] and genome sequencing [66], have different performance/energy requirements, and are deployed on a variety of different platforms. These platforms must be able to serve a diverse range of multi-threaded applications to multiple users, simultaneously. To meet the performance and Quality-of-Service (QoS) constraints, cloud server and HPC platforms are comprised of many-core processors, as described by [67]. The processor cores might be asymmetric with different micro-architectures, such as in-order and Out-of-Order (OoO) cores used by the ARM big.LITTLE architecture [24], with thread level allocation policies appropriately allocating the workload to respective cores, meeting QoS constraints under limited power budget. To optimize energy efficiency and performance, a system-level simulator is required, capable of simultaneously executing multi-threaded applications in a Full System (FS) environment with a complete operating system (OS), on a heterogeneous many-core system. Furthermore, the simulator should also be capable of a detailed system-level profiling to identify the bottlenecks, therefore, helping in designing strategies and architectures to alleviate these bottlenecks and enable fast architectural exploration methodologies to assess novel techniques at the system level.

This chapter presents gem5-X, *"a gem5-based full-system simulator with architectural eXtensions"*. Gem5-X is a simulation framework that supports fast profiling and seamless analysis of multi-threaded applications. It supports architectural extensions both for the compute and memory sub-systems, and enables exploration of various architectural parameters, to have an overall performance-energy-optimized architecture. Gem5-X allows seamless simulation of any

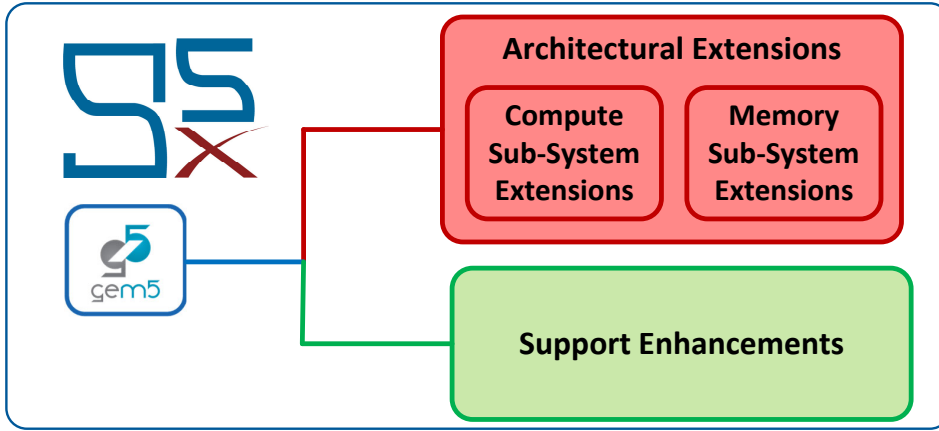


Figure 2.1 – Gem5-X simulation framework.

multi-threaded application running on top of a modern Linux operating system, for detailed architectural analysis and optimization. The contributions brought by the gem5-X simulation framework are split into architectural extensions and support enhancements on top of gem5, as shown in Fig. 2.1, with all of them completely integrated in FS mode. Furthermore, a methodology to profile applications within gem5-X, identify bottlenecks, and validate architectural modifications and extensions for application performance and energy optimization is also provided with the gem5-X simulation framework.

In particular, the following contributions to gem5-X simulation framework are presented in this chapter:

- The gem5-X framework, which enhances gem5, developed by [31], with a full-system simulation of ARM-64 in-order and OoO architectures running on a modern Linux OS. The in-order and OoO cores are tuned and validated for performance against a real ARM JUNO platform developed by [34] with a Mean Absolute Error (MAE) below 4%. Gem5-X is open-sourced to the community, enabling innovative extensions of ARM 64-bit architectures.
- The gem5-X architectural extensions comprising both compute sub-system and memory sub-system extensions. These extensions are supported in gem5-X FS mode.
- The gem5-X support enhancements, enabling in-simulator profiling, enhanced checkpointing and file sharing between host and simulated systems.
- The 3D stacked High Bandwidth Memory v2 (HBM2) memory model in gem5-X is presented, with interleaving to uniformly distribute the memory traffic. It is then analysed with the STREAM benchmark, giving insight into memory bandwidth (BW) scaling and the potential BW bottlenecks in the system.

- The gem5-X methodology to profile applications, identify bottlenecks, and validate architectural modifications and extensions for application acceleration.

This chapter is organized into different sections, starting with the overview of state-of-the-art and related work in Section 2.2. Then the gem5-X architectural extensions are presented in Section 2.3 and the support enhancements in Section 2.4. Furthermore, the validation of the gem5-X platform is presented in Section 2.5, followed by complete BW analysis of 3D stacked HBM2 memory model in gem5-X in Section 2.6. Moreover, power models and area estimates for simulated systems using gem5-X are presented in Section 2.7. Finally, the gem5-X architecture exploration and optimization methodology is presented in Section 2.8, which will be utilized in the next chapters, when exploring and optimizing architectures using gem5-X.

## 2.2 Related work

State-of-the-art complex applications deployed across the range of compute devices from edge devices to servers in the cloud require new innovative and heterogeneous architectures to run optimally in terms of performance and energy efficiency [68]. Full system architectural simulators are required for fast architectural exploration with accurate performance and energy estimates, to give an insight during the initial design phase by enabling early deployment of the application on the simulation platform, and reduce the time-to-market of new products [69, 70].

Quite a lot of research has gone into developing architectural simulators, but each has its own shortcomings. Sniper [71] is a multi-core parallel simulator with fast turnaround time, but its main drawback is that it only supports traditional x86 architectures. Simics [72] is another architectural simulator enabling applications to run on different hardware platforms. It is combined with Simflex [73] for timing information, but is limited to SPARC architectures only. Gem5 [31] is a FS architectural simulator being widely used both in academia and industry as it supports multiple Instruction Set Architectures (ISAs), like x86, ARMv7, ARMv8, MIPS and ALPHA. In addition to a variety of ISAs, it also supports different Central Processing Unit (CPU) models for these ISAs, like atomic, in-order and OoO CPU models, as well as multiple caching protocols and coherences. On the memory side, it supports many traditional and emerging memories. Further, gem5 supports full system simulations via different Linux based operating systems like Ubuntu and Android, enabling applications to run as they would do on a real platform, making it the best candidate for architectural exploration. However, the higher accuracy and sequential nature of gem5 results in slower simulation and long turnaround times. One of the major drawbacks of gem5 is that not all components work out-of-the-box (OOB) i.e., in all simulation modes. In particular, the Hybrid Memory Cube (HMC) is a memory type supported in gem5, but unavailable in FS mode. Similarly, system stability is not guaranteed for all component combinations. The Linux distributions and kernels provided are quite old with minimal package installation, and they can demonstrate FS mode, but are incapable of exploiting all the features in FS mode. Even so, gem5 is chosen as the base simulation platform as its flexibility allows for straightforward architectural extensions.

The gem5-X simulation framework enhances gem5 with OOB system-level support for many-core ARM-64 architectures, architectural extensions and new memory models. It enables seamless FS simulation with profiling support, to analyze the speedup and gains of the functional blocks within the application as a result of new architectural extensions and optimizations. It also supports file sharing between the host machine and the simulated system using workload automation (WA), which is built into the Linux kernel provided for gem5-X. Therefore, the simulation turnaround time is reduced, when debugging the application on a novel architecture within gem5-X. On the architectural side, gem5-X supports heterogeneous architectures, like in-order and OoO cores along with custom accelerators like an in-cache computing engine [37]. In-cache computing allows massive Single-Instruction-Multiple-Data (SIMD)-like operations to be performed in the cache hierarchy as proposed by [74]. The in-cache computing architecture incorporated in gem5-X is similar to BLADE proposed by [37], targeted for the L1 cache of ARM-based many-core systems, as opposed to the Last-Level-Cache (LLC), as in Neural Cache proposed by [75]. In addition to the heterogeneity at the compute side, gem5-X also supports heterogeneous memory types like traditional DDR4 alongside 3D stacked HBM2, thus, enabling a highly heterogeneous system, with full Linux stack. The BW analysis on HBM2 memory model in gem5-X is also performed using the STREAM benchmark [76] to validate that it provides the required BW in comparison to DDR4, as will be discussed in Section 2.6. To the best of my knowledge, this is the first work that simulates a complete Linux based system with clustered heterogeneous compute cores (in-order and OoO) with in-cache computing engine along with 3D stacked HBM2 memory.

With increasing growth and popularity of cloud based services, many of the emerging workloads can be deployed either on the cloud or on HPC systems. Gem5-X supports Virtual Machines (VMs) which can be deployed in the cloud infra-structure. It can also be used to explore and optimize architectures for HPC workloads.

### 2.3 Architectural Extensions

The gem5-X simulation platform is the enabler for the architectural exploration and optimization of emerging state-of-the-art multi-threaded applications, running on a full Linux system. Gem5 can be modified at any level of the architecture, from the multi-core pipeline through the interconnects and cache down to the DRAM. Therefore, gem5-X enhances gem5 with novel architectural extensions, enabling performance, power and area optimizations of the architecture for a given target application. It modifies the gem5 core to support extensions such as Scratchpad Memories (SPMs), HBM2, modified CPU pipelines, in-cache computing architectures and ARMv8 ISA extensions. All these architectural extensions are supported in gem5-X FS mode and can be implemented and profiled to evaluate their effectiveness for the target application. Furthermore, the architectural extensions added in gem5-X are generic and not specific to the applications used as case study to demonstrate these extensions in Chapter 3 and Chapter 4 of this thesis. Therefore, they can be used for optimizing any other application in the future, namely, by utilizing the gem5-X exploration and optimization methodologies.

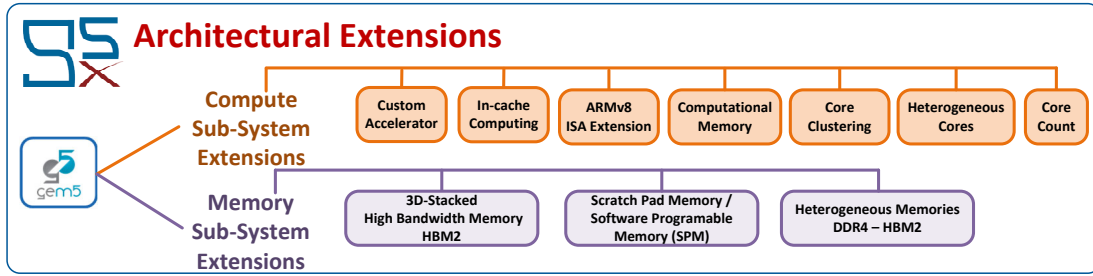


Figure 2.2 – Gem5-X architectural extensions.

The architectural extensions in gem5-X are classified into *compute sub-system* and *memory sub-system* extensions as shown in Fig. 2.2. Hence, I will now look into the compute and memory sub-systems of the gem5-X platform.

### 2.3.1 Compute Sub-System

Gem5-X enables different architectural extensions for the compute sub-system, which be will discussed next.

#### 2.3.1.1 ISA Extension

To support an accelerator or to add a new functional unit within the CPU core, gem5-X supports ISA extension, using the reserved op-codes of the original ISA specifications. Adding a custom instruction using one of the unused opcodes enables seamless integration of a new functional unit or accelerator in FS mode in gem5-X with full Linux stack. The accelerator or the functional unit can be used by issuing in-line assembly calls to the new instruction in C/C++. The added instruction, when decoded, issues the request to the accelerator/functional unit.

To support the new instruction, the decoder in the gem5 ISA domain-specific language (DSL) is modified. New flags are also added with the instruction, so that the accelerator/functional unit can recognize and handle it accordingly. Furthermore, the added flags record the statistics of the new instruction and associated accelerator/functional unit.

#### 2.3.1.2 Custom Accelerator

Custom accelerators are used to speed-up specific tasks or applications using dedicated hardware. This increases both performance and energy efficiency of the overall system. Gem5-X supports adding custom accelerators to the system in addition to general purpose processors. In particular, two approaches can be used to integrate the accelerator in the system with general purpose processors, namely, through system-memory map or a custom instruction extension.

### 2.3.1.2.1 *Memory Mapped Accelerator*

The custom accelerator can be integrated in the system via memory map by modifying the system memory map in gem5-X. The memory map thus shall include both the control and data registers of the accelerator. The system memory map in gem5-X need to be modified in the FS configuration script to support the accelerator in FS mode. Furthermore, a DMA can be setup with the accelerator to directly move data to and from the memory, without traversing it through the CPU cores.

The memory mapped accelerator approach is recommended for modelling accelerators that are either off-chip, or those who have a large number of instructions or commands. This enables these accelerators to be used with any ISA CPU core in gem5-X. To access the memory mapped accelerators in gem5-X, one can either develop device drivers for them or *mmap()* them from within the target application in FS mode.

### 2.3.1.2.2 *Custom Instruction Accelerator*

The custom accelerator can also be integrated in the system in gem5-X via custom instructions. In this approach, the accelerator is tightly coupled and integrated with the CPU core. Gem5-X ISA extension using the reserved op-codes in the ISA specification are used in this approach, as discussed earlier in Section 2.3.1.1.

Custom instruction is the appropriate approach for accelerator integration when accelerator needs to be tightly integrated with the CPU core and if there are just a few (5-6) instructions for the accelerator. The accelerator can be used in FS mode using inline assembly calls to the custom instructions.

### 2.3.1.3 **BLADE In-cache Computing Engine**

BLADE [37, 38] is a hardware extension for CPU caches that enables computing directly within the cache. BLADE operations are performed by first precharging the bitlines of the cache subarray, as illustrated in Fig. 2.3a. Then, two wordlines are activated simultaneously, thus allowing the contents of two rows of bitcells to be connected to the bitlines, as illustrated in Fig. 2.3b. The bitlines are discharged according to the contents of the bitcells, resulting in an *and* operation on the bitline and a *nor* operation on the inverse bitline. These signals can be further combined via a *nor* gate to achieve a *xor* operation. Finally, further processing allows complex operations such as addition and multiplication to be performed [37, 38]. The operation results are then written back to the cache. Application runtime is improved in two ways; first, data movement is reduced, and second, SIMD operations can be performed on multiple operands simultaneously, for example 128 1-byte operations in a cache with 2 subarrays and 64-byte wordlines. It also reduces energy consumption, as the computation is performed in the cache, hence saving energy both on the internal bus transactions as well as static energy for the idle core, when the data is being fetched.

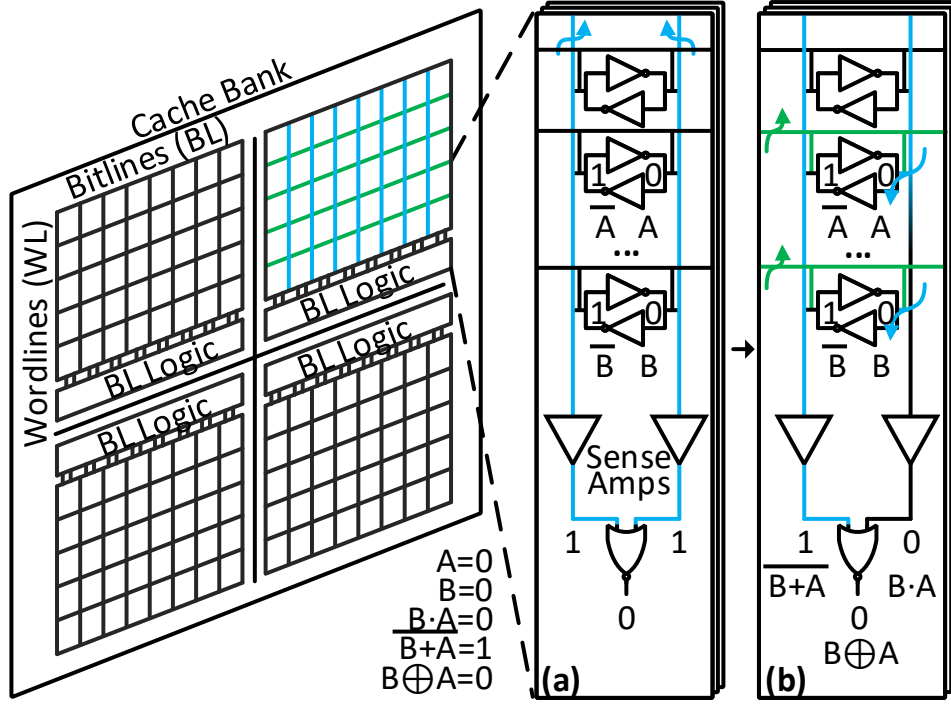


Figure 2.3 – Cache subarray with AND/NOR/XOR bitline computing on values  $A=0$  and  $B=0$ . Bitwise operations are performed by first (a) precharging the bitlines, then (b) activating multiple wordlines, thus discharging the bitlines through the connected bitcells.

BLADE provides acceleration primarily for applications that exhibit high cache locality and data computing regularity, as it performs SIMD operations on many physically successive operands.

To add an in-cache computing architecture to gem5-X, gem5's L1 cache model is modified, as discussed by [31], to simulate in-cache addition, subtraction, multiplication, shifting, greater-/less than and absolute operations in a timing and architecture-accurate manner. BLADE is implemented within the private L1 cache, as this provides a favorable trade-off between area footprint and functionality, as well as simplifying cache coherence considerations, in contrast to implementation in shared caches. BLADE can be implemented in lower level caches, providing an increase in the number of possible parallel operations, in exchange for a greater area overhead and coherence complexity.

To support BLADE in FS mode, a custom instruction is added using one of the unused opcodes in the ARMv8 ISA specification [39]. A new *cachecompute* flag is also added with the instruction, so that the cache controller can recognize it as a cache compute request and handle it accordingly. The flag is also used to record the statistics for in-cache computing engine in gem5-X. The accelerator can then be used by issuing in-line assembly calls to the new instruction in C/C++.

When an in-cache instruction is decoded by the CPU, the required operations are scheduled and their operands are loaded into the cache from main memory. The in-cache computing operands must share bitlines to be eligible for in-cache computing operations. Sets that can interact with

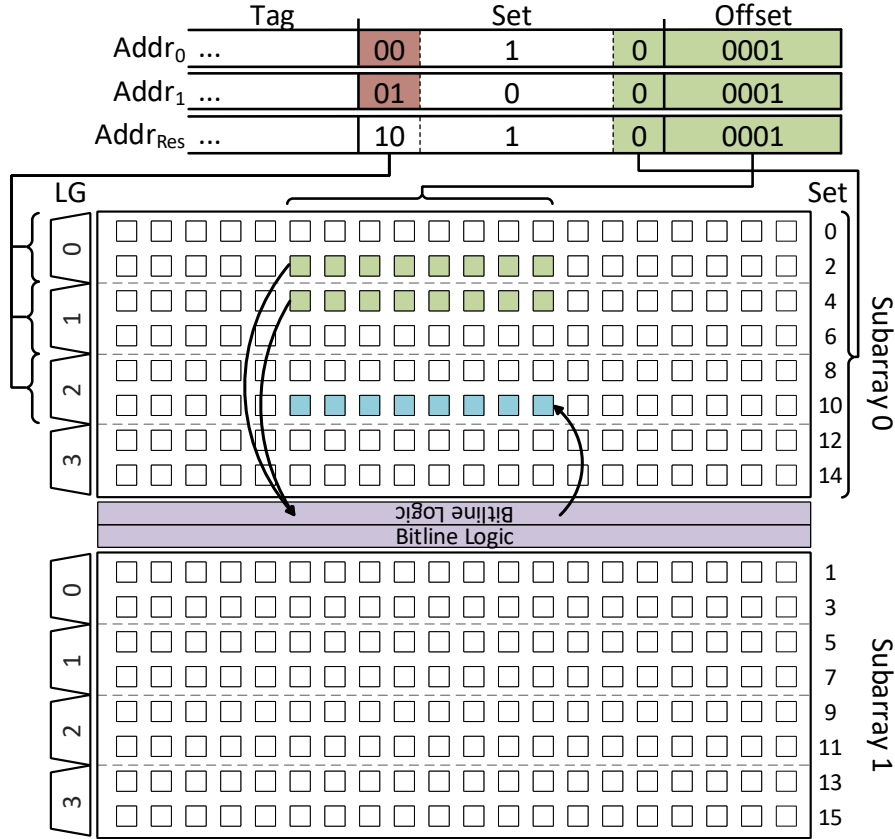


Figure 2.4 – The highlighted operands in sets 2 and 4 have matching offsets and set LSBs, and differing set MSBs, indicating that they share bitline logic, but not Local Groups (LGs).

each other are considered local, and therefore the requirements placed upon operands can be called operand (or data) locality constraints, as discussed in [38]. These constraints depend on the geometry of the cache, as factors such as cache size, subarray size, and associativity affect which sets share bitlines. However, all variations in cache geometry can be abstracted to three constraints on the operand memory addresses, as shown in Fig. 2.4:

- The offset bits between two operands must match, guaranteeing operand alignment within a cache block.
- A certain number of set LSBs must match, guaranteeing that the operands share the same subarray.
- A certain number of MSBs must differ in order to avoid data corruption, due to activation of multiple wordlines. Local bitlines are used in BLADE to divide groups of wordlines into Local Groups (LGs), where all wordlines in an LG share an local bitline pair, which is connected to the global bitline, as discussed in [38].

These data locality constraints are ensured by reserving 1GB of cacheable memory that can be



*mmaped* by an application, allowing a fine grained control of where operands are stored in the cache. The target application is also modified to guarantee data alignment during operations to be performed in-cache. Applications need to be modified at the source code level to support BLADE operations. BLADE has been incorporated in the L1 cache for ARM in-order cores in gem5-X.

By integrating the in-cache computing architecture into gem5-X, its performance can be measured on top of a full software and Linux kernel stack, allowing events such as context switching, cache line eviction, and performance loss due to complex data accesses to be evaluated. This is important when assessing the real-world applicability of any architectural innovation, and its ability to generalize to other applications.

To guarantee timing accuracy and estimate power consumption, the in-cache architecture was developed and simulated in 28nm bulk CMOS technology using Cadence Virtuoso, as discussed by [77]. The power and timing values were extracted and converted to cycle counts that are integrated into the gem5-X's event scheduler.

### 2.3.1.4 Computational Memory

Computational Memory (CM) is a memory technology like Resistive Random-Access Memory (RRAM) or Analog In-Memory Computing (AIMC) core that can store computational data as well as perform compute operations within the memory.

Gem5-X enables CM extension by integrating it in the execution stage of the CPU pipeline, as shown in Fig. 2.5. The gem5-X ISA extension mechanism is utilized for the CM to enable it in FS mode and to tightly couple it to the CPU. Therefore, it is integrated as a new functional unit in the CPU execute stage. A key benefit of this configuration is that the transactions between the CPU and CM core are reduced to the order of single nanoseconds. This is as opposed to standalone CM accelerators where an off-chip communication with the CPU is necessary to access its rich digital functionalities. Software modification to the application is required to use the CM, using in-line assembly calls to the new instructions in C/C++.

To ensure correct timing and energy consumption, circuit level simulations in Cadence Virtuoso is used to extract the timing and power values, which are then integrated within the CM core model in gem5-X.

Simple memory model in gem5-X is extended and used to model load/store functionality of the CM core. The compute functionality is implemented in conjunction with gem5-X ISA extension. The CM core is designated as a functional unit in gem5-X with its own flags. Hence, whenever the decoder in the CPU pipeline encounters a CM instruction, it forwards that to the CM core. Then, the compute engine within the CM core receives the instructions along with designated flags, and performs the corresponding operation.

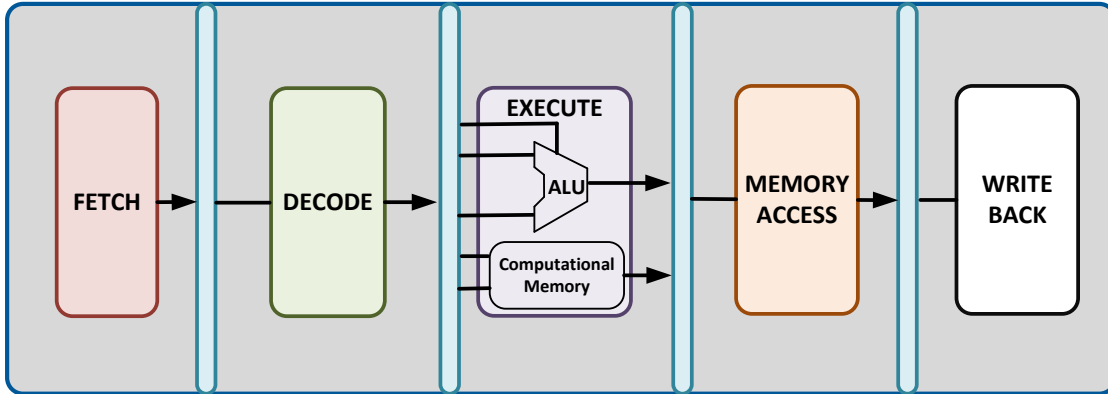


Figure 2.5 – Simplified block diagram of a generic processor 5-stage pipeline integrating the CM in execute stage of the pipeline.

### 2.3.1.5 Core Clustering

Gem5-X enables clustering of the compute cores, both for heterogeneous as well as homogeneous core types. Each cluster supports multiple cores and has its own shared cache. If there are only 2-levels of caching in the memory sub-system, the shared cache for each cluster is the LLC. However, if there are 3-levels of caching, the shared cache of each cluster is the L2 which then connects to the system L3 cache, as shown in Fig. 2.6. System architects can further expand the cluster implementation with each cluster having its own independent clock. Core clustering is fully supported in gem5-X FS mode. To add the core clustering support in gem5-X, cache configuration script, base CPU model files and FS configuration script are modified.

Clustering enables independent applications to execute on independent multi-core clusters with their own independent shared cache (L2 or LLC) in a multi-core system. Moreover, if there are independent kernels within an application that do not share resources with other kernels of the application, clustering enables to allocate these independent kernels on different multi-core clusters, each with its own independent shared cache. The benefit of this clustering approach is the avoidance of cache thrashing. Furthermore, as the clusters can operate independently at different clock frequencies, this can help in energy savings while meeting the performance requirements, as the performance and clock requirements of each cluster can be different.

### 2.3.1.6 Heterogeneous Cores

Gem5-X supports ARMv8 64-bit energy-efficient in-order cores as well as ARMv8 64-bit high performance OoO cores. To this end, I added support for heterogeneous architecture simulation enabling both in-order and OoO cores to be simulated simultaneously in FS mode, with different applications or application kernels being allocated efficiently to different core types to maximize performance, as well as energy efficiency. Furthermore, heterogeneous cores can be combined with clustering in gem5-X with each cluster having its own shared cache, and different core types

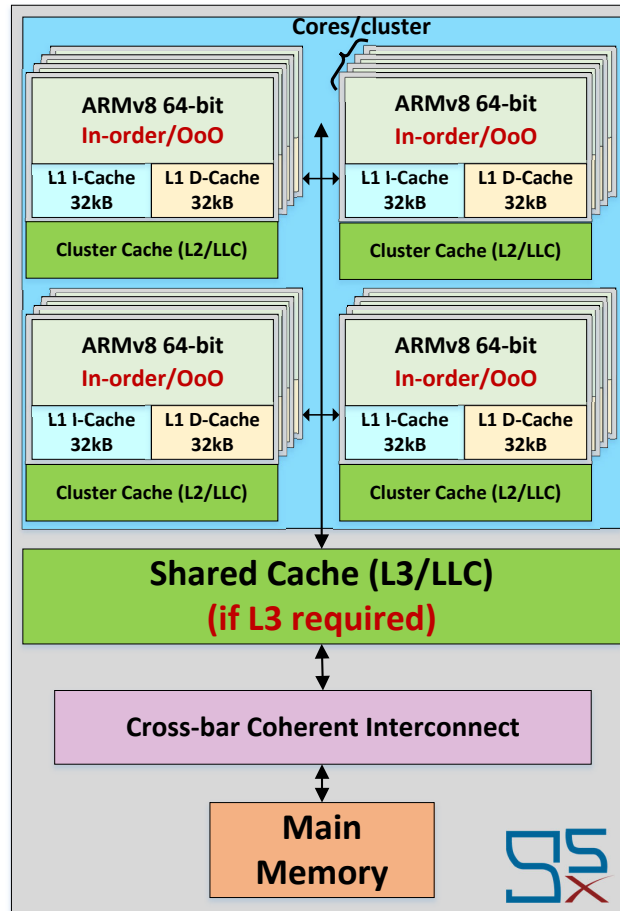


Figure 2.6 – Multi-core heterogeneous clusters in gem5-X. Each cluster supports multiple cores which can either be in-order or OoO cores for a given cluster with its own cluster cache. If the cluster cache is the LLC, then the clusters connect directly to the main memory via a cross-bar interconnect. However, if a shared L3 is the LLC then the clusters are connected to the L3 shared cache, which ultimately connects to the main memory.

in different clusters, i.e., an in-order cores cluster and an OoO core cluster, as shown in Fig. 2.6. Real heterogeneous platforms also take a similar clustering approach with each cluster having its own shared cache, as in the ARM JUNO [34] and Hikey960 [78] ARM big.LITTLE platforms.

To support heterogeneous cores in gem5-X, FS configuration script, system simulation script, cache configuration script and base CPU model files are modified. The modifications in the system simulation script ensures that checkpoints can be resumed for two different CPU models (in-order and OoO) within the same simulation.

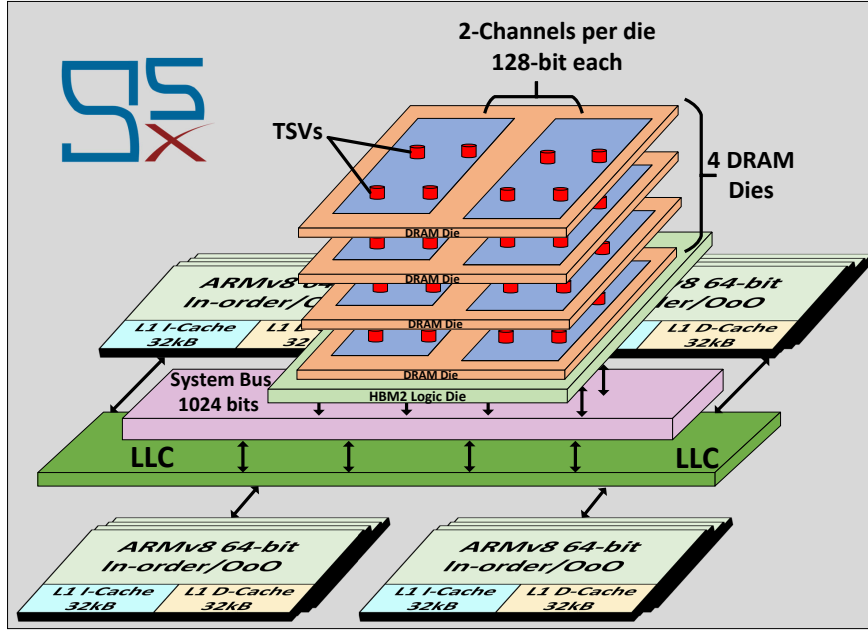


Figure 2.7 – 3D stacked HBM2 Architecture.

### 2.3.1.7 Core Count

Gem5-X enables many-core simulation with support for 256 cores simulation. To support 256 cores in FS mode, the Linux kernel is updated with a maximum core count of 256 cores. Moreover, the device tree binary (DTB) files are also updated to support many core simulation in gem5-X.

## 2.3.2 Memory Sub-System

The memory sub-system is as essential as the compute sub-system, especially in the case of memory intensive applications. Gem5-X adds various extensions in the memory sub-system, which will be discussed next.

### 2.3.2.1 High Bandwidth Memory v2 (HBM2)

The HBM2 memory, as described by [32], is based on 3D stacked DRAM banks connected by Through-Silicon-Vias (TSVs). It can achieve a BW of up to 307.2 GB/s [79], enabled by multiple channels. It supports up to 8 channels with each channel being 128-bit wide. To implement the functional behavior of the HBM2 memory model in gem5-X, the DRAM controller model of gem5 is extended according to the architectural details of HBM2, as summarized in Table 2.1. To have 8-channels with memory interleaving, 8 DRAM controllers, each 128 bits wide, are initialized. All 8 DRAM controllers are connected to a 1024-bit wide system bus, which connects to the cache hierarchy. For accurate timing estimates the timing values presented by [80] are

Table 2.1 – Implemented HBM2 architecture.

Parameter	Value	Parameter	Value
Clock period (tCK)	0.833ns	Channel width	128 bits
Bandwidth	2.4Gbps/pin	#Channels	8
#I/Os	1024 pins	Ranks per channel	1
Banks per rank	16	Burst length	4
Bank groups per ranks	4	-	-

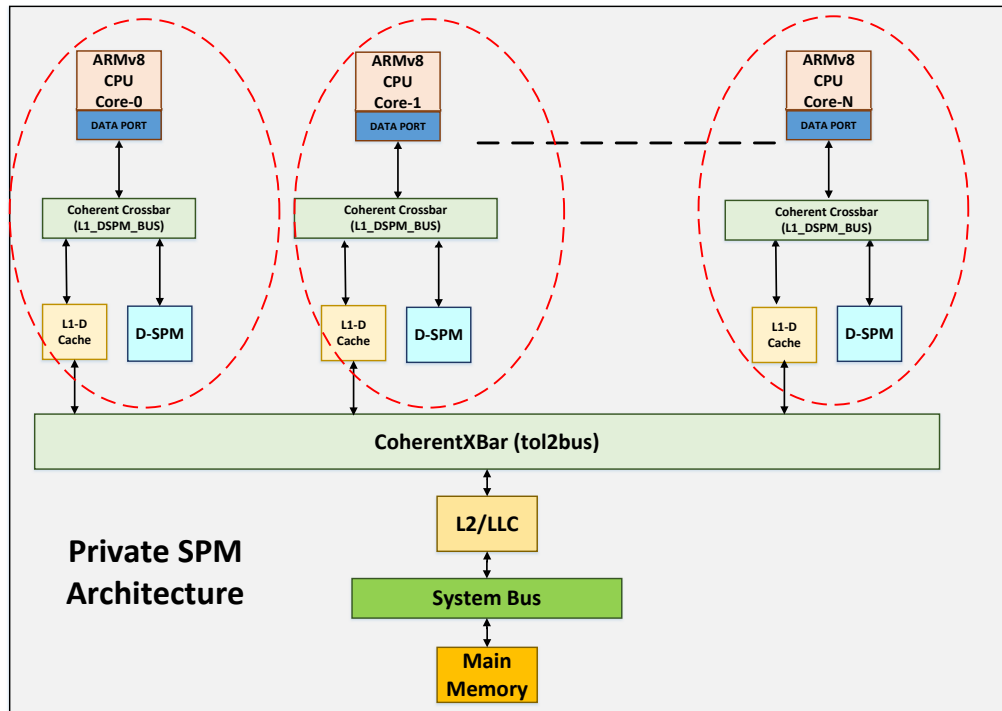


Figure 2.8 – Private SPM architecture in gem5-X.

used.

The memory interleaving among different channels of the HBM2 model uniformly distributes the memory accesses across all the channels. No additional support is required from the software perspective, thus enabling any application to be executed either on a traditional DDR4 or HBM2 based memory system in FS mode, without modifying the software. Figure 2.7 shows the 3D stacked HBM2 with multiple channels in gem5-X. Each channel is 128-bits wide, hence, for 8-channels HBM2, the system bus is configured to be 1024-bits wide.

### 2.3.2.2 ScratchPad Memory (SPM)

SPM, also referred to as software-programmable memory, is a tightly coupled memory to the CPU core at the same level as the L1-D cache. These memories are particularly useful when there

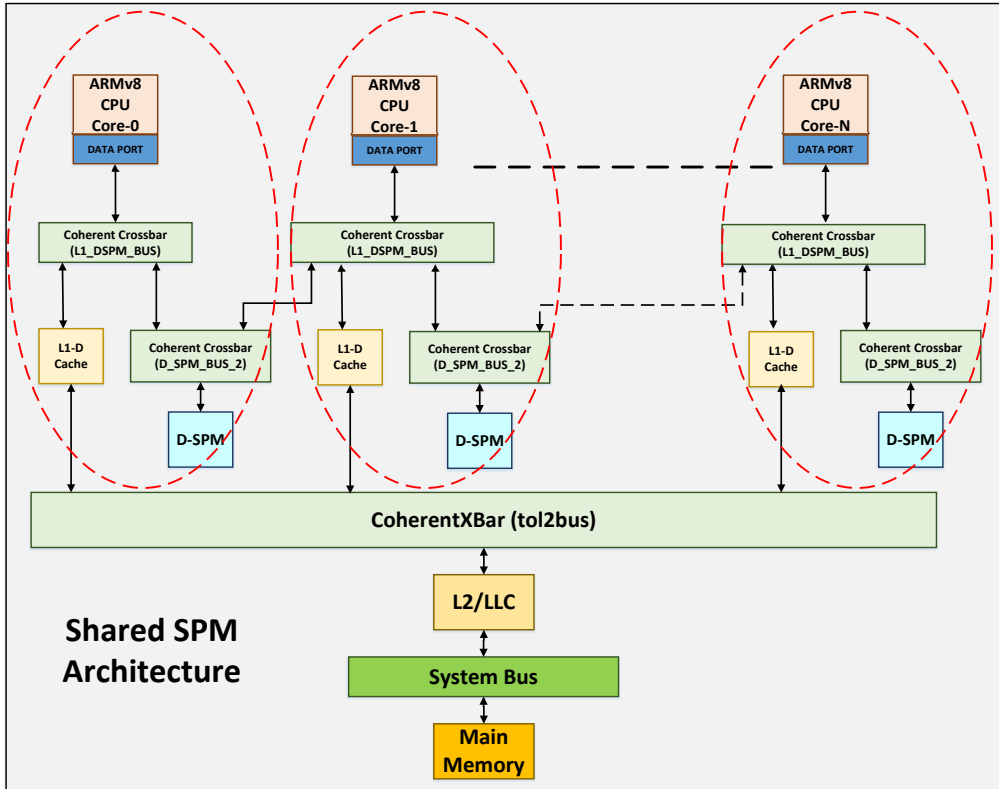


Figure 2.9 – Shared SPM architecture in gem5-X, with the SPM being shared between two consecutive CPU cores.

is a high memory contention in the cache and the working data-set gets evicted from the cache, due to cache thrashing. Since SPMs are software controlled, in contrast to hardware controlled caches, the programmer can store the working data-set in the SPMs, hence, keeping it close to the CPU.

Gem5-X supports SPMs in the memory sub-system. The SPMs can either be private to each core in the system, as shown in Fig. 2.8 or they can also be shared between two consecutive cores in the system, as shown in Fig. 2.9. A private SPM can only be accessed by the core to which it is attached. In case of a shared SPM, it can be accessed by the consecutive cores to which it is attached. In Fig. 2.9, the D-SPM 0 is shared between core-0 and core-1, the D-SPM 1 is shared between core-1 and core-2 and so on.

SPMs are supported in FS mode in gem5-X. To enable this support, the memory map of the SPMs is separate from the cache hierarchy and the main memory. Therefore, to allocate data on the SPM, *mmap()* is used from within the application in FS mode. Hence, the data allocated on SPM is not cached and with no copy on the main memory. Simple memory model in gem5-X is used to model the SPM, with timing values configured the same as that of an L1 cache. The cache configuration, base CPU and FS configuration files are modified to enable the support of SPMs in gem5-X.

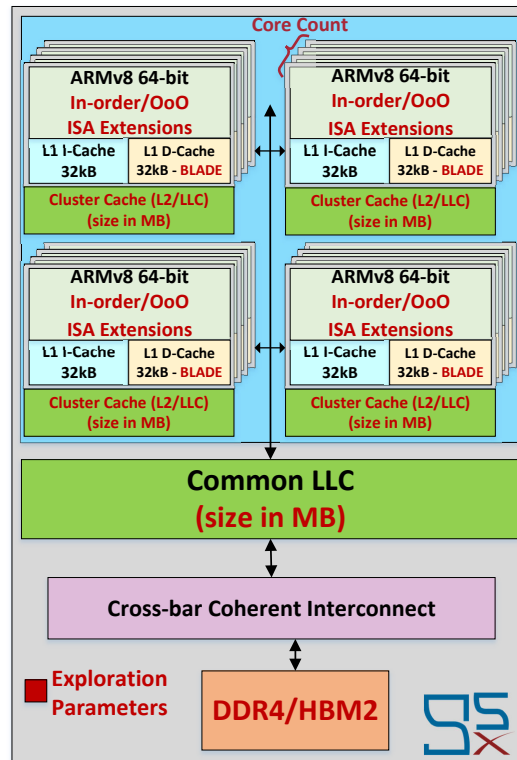


Figure 2.10 – Gem5-X platform with architectural exploration parameters. These configurable parameters include the core count, choice of core as either in-order or OoO, ISA extensions, BLADE in-cache computing engine in L1-D cache, cluster cache (L2/LLC), common LLC for all the cores/clusters, and the main memory as either DDR4, HBM2 or both. Caches and memory sizes are configurable as well.

### 2.3.2.3 Heterogeneous Memories

Gem5-X also supports heterogeneous memory types like traditional DDR4 along with 3D stacked HBM2 simultaneously in the same system in FS mode. Both memories are *mmap*ed separately in the gem5-X configuration file. During the Linux boot-up, one of the memories is defined as the base memory, with the full address space available to the kernel. The other memory can then be allocated using *mmap()*, from within the application, if required. By default, allocations are done to the base memory defined during the kernel boot.

### 2.3.3 Gem5-X Parameters

Gem5-X simulation platform enables architectural exploration with architectural extensions both in the compute and memory sub-systems. Figure 2.10 shows the gem5-X platform with the different configurable exploration parameters highlighted in red.

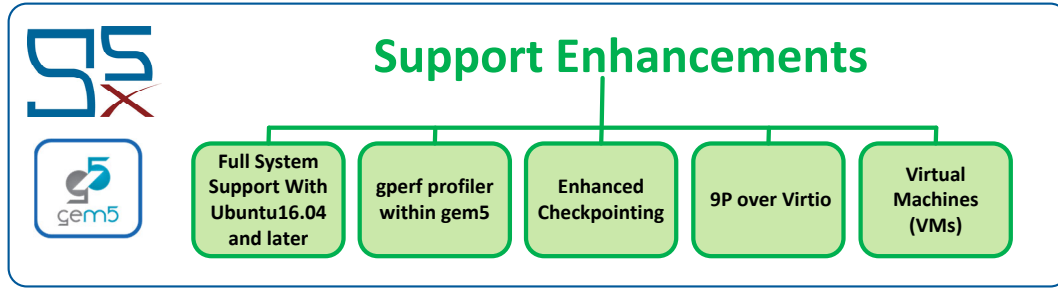


Figure 2.11 – Gem5-X support enhancements.

## 2.4 Support Enhancements

To enable gem5-X to integrate all architectural extensions from the software perspective and also to decrease the simulation turnaround time, various support enhancements are added to gem5, as shown in Fig. 2.11.

### 2.4.1 ARMv8 64-bit Full System (FS) Support

The vanilla gem5 provides FS support for ARMv8 (aarch64), however, this support is limited. The default disk images that come with gem5 are outdated and older versions of the OS like Ubuntu 14.04 distribution with minimal installed packages and disk storage space limited to 3GB. This situation limits the porting of new emerging applications to gem5. Moreover, the system stability is not guaranteed in gem5 FS mode, as the simulation crashes with higher core count, and all the system components are either not compatible with each other or not supported in FS mode.

Gem5-X enables stable ARMv8 (aarch64) FS support based on Ubuntu 16.04 and later disk images with Linux kernels v4.3, v4.13 and v5.0. Gem5-X FS support provides 30GB storage to allow larger applications, benchmarks and data-sets to be stored with ease on the disk images provided. The FS mode in gem5-X is thoroughly tested with all architectural extensions including accelerators, CM cores, heterogeneous systems along with emerging memories including 3D stacked HBM2 and SPM. The required Linux kernel configurations and libraries for all the compute and memory components as well as for support enhancements are provided with gem5-X for seamless OOB FS integration. The FS mode in gem5-X supports the *pthread* library to allow thread-level parallelism on a multi-core system, which is required by most applications. This extension to gem5 allows the installation and execution of any application that runs on a regular Linux system, using both in-order and OoO ARMv8 cores.

To run experiments on an application or benchmark in gem5-X, it needs to be on the disk image. For this, the disk image can be mounted as in any Linux system using the Linux *mount* command. Then a user can *chroot* into the image using QEMU. QEMU [81] allows emulating the ARMv8



64 bit image on an x86 host machine. Therefore, once in the emulation mode, user can install any benchmark or application on the disk image, which will later be used during the simulation in gem5-X.

### 2.4.2 Gperf Profiler

Profiling capabilities within FS mode on gem5-X have been included, by installing the gperf profiler on the disk image. The *gperf* statistical profiler developed by [33] provides profiling capabilities on gem5-X itself with minimal overhead, enabling the identification of application bottlenecks and exploration of the effectiveness of architectural modifications and extensions.

### 2.4.3 Enhanced Checkpointing

Checkpointing in gem5 drastically reduces simulation time in FS mode. Gem5-X enhances the existing checkpointing in gem5 by marking the Region-Of-Interest (ROI) of applications. Gem5-X simulations are launched using a simple functional CPU, run until the ROI and checkpointed. Then, simulations are switched to either in-order or OoO detailed models. This method vastly reduces the time required to setup simulation of applications. Moreover, checkpointing reduces the burden of the debugging process, by checkpointing just before the point-of-failure and then resuming with the debug mode.

### 2.4.4 9P over Virtio

I utilize the 9P protocol developed by Bell [82] over a virtio device driver developed by [83], to allow fast modification of files without modifying the root file system in gem5-X FS mode. While this feature is available in vanilla gem5, it is not enabled by default and has no kernel support. Both of these features are provided in gem5-X. Once Linux is booted, a folder on the host machine can be mounted within gem5-X to access files on the host system. This enables file sharing and fast data transfer between the host system and simulated system in gem5-X. Without 9P mounting, every time a program is modified, one needs to reload the disk image required for FS simulation and reboot Linux. In gem5, this process can take up to 20-30 minutes every time a program is modified, a bottleneck that gem5-X eliminates.

### 2.4.5 Virtual Machines (VMs)

Gem5-X supports VMs, virtualized via Linux LXC containers. The LXC support in FS mode is added by enabling it in the Linux kernel as well as the disk image. The VM support in gem5-X enables the capability to simulate cloud workloads in gem5-X, in a virtualized fashion.

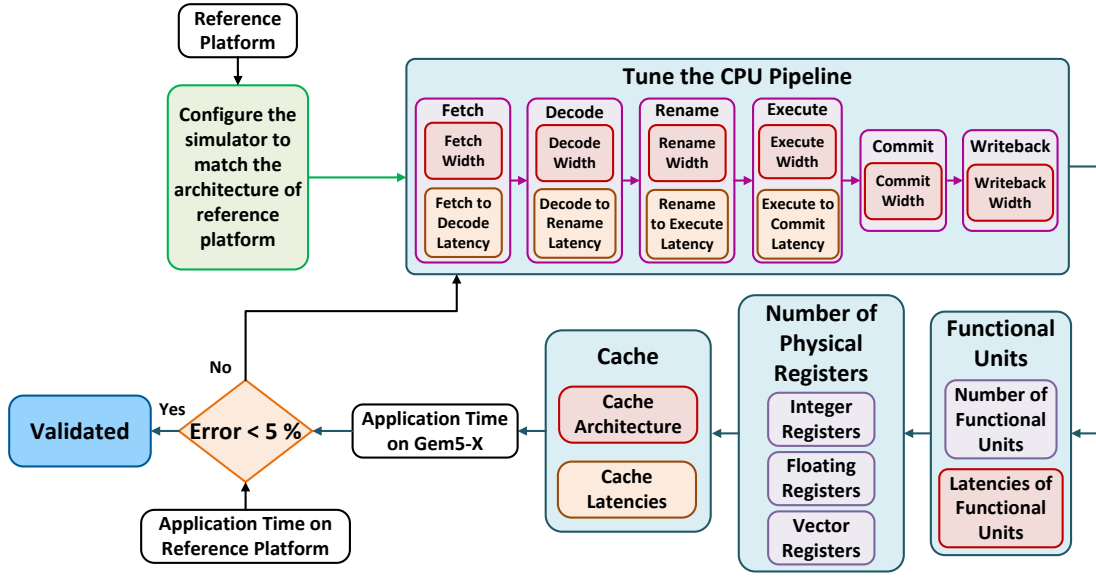


Figure 2.12 – Methodology for validation of the simulation platform.

## 2.5 Validation

Validation of the simulation platform is necessary to have confidence in the results of the simulation. To perform the validation of gem5-X simulation platform, a validation methodology is proposed, as shown in Fig. 2.12.

As a first step of the validation methodology, the simulation platform is configured to match the architecture of a reference platform. For this step, gem5-X was configured to simulate the architecture of a real ARM JUNO platform [34]. JUNO is an ARM big.LITTLE platform based on ARMv8 64-bit ISA, with both ARM in-order and OoO cores. This enables, to validate both type of cores in gem5-X against the real cores in ARM JUNO. The architectural configuration of gem5-X in this step includes, number of cores, core frequencies, cache levels, cache sizes and the memory size.

The next step involves the tuning of the CPU pipeline stages in gem5-X. The widths and latencies for fetch, decode, rename, execute, commit and writeback stages of the CPU pipeline are tuned during this step. As a starting point for these values ARM Cortex-A57, as presented in [84] for OoO cores and the ARM Cortex-A53, as presented in [85] for in-order cores, are utilized.

Furthermore, the number of functional units and the latencies of the functional units are tuned for in-order and OoO cores. The functional units include, the integer Arithmetic Logic Unit (ALU), integer multiplication and division unit, floating point unit, vector processing units and memory read and memory write units. Then, the number of physical integer, floating point and vector registers are tuned according to the specifications of in-order and OoO cores in the reference platform.

Table 2.2 – Validation error of gem5-X when compared against a real JUNO platform.

Video Resolution (pixels)	In-order Core - Error	OoO Core - Error
176x144	2.4%	1.93%
416x240	4.3%	1.85%
1920x1080	2.7%	4.25%

Moreover, the cache heirarchy is tuned in accordance with the cache sub-system of the reference platform. The cache latencies for L1-D, L1-I and LLC are tuned. The cache architecture is also configured in terms of set associativity of the caches, as compared to the reference platform.

Once all the parameters are tuned, a target application or a benchmark is run on both the real platform (ARM JUNO in this case) and the simulated platform in gem5-X. The difference in terms of execution time is checked. If the error is less than 5%, then the simulation platform is considered to be validated. However, if the validation error is more than 5%, then the parameters in CPU pipeline, functional units, physical registers and caches are tuned to bring down the validation error.

For validation of gem5-X, a real-time video encoding application Kvazaar, developed by [1] is used. Different video resolutions are used for encoding with Kvazaar for validation purposes. The validation errors for both in-order and OoO cores for various video resolutions are shown in Table 2.2. This table indicates that gem5-X has a maximum validation error of 4.3%.

Kvazaar is used for validation as it utilizes all the components (all CPU functional units, including NEON SIMD, caches, memory) in the system. The network and disk models in gem5-X are not tuned, as the purpose of this thesis is to explore novel compute and memory architectures (and not disk or networks). As a result, the performed validation of the simulation framework provides confidence in the results provided by gem5-X.

## 2.6 HBM2 Model Bandwidth Analysis

After validating the ARMv8 in-order and OoO compute cores in gem5-X with a validation error of up to 4%, as discussed in the previous section, the BW analysis of HBM2 memory model in gem5-X is presented in this section. The BW analysis is performed to validate that HBM2 provides the required BW in comparison to DDR4 and also to check that the memory traffic is being uniformly distributed among different memory channels. STREAM [76], which is a well-known memory BW benchmark, is used for the HBM2 BW analysis.

### 2.6.1 BW Analysis Methodology

Figure 2.13 shows the methodology for the analysis of HBM2 BW and its comparison to DDR4 in gem5-X using the STREAM benchmark, by changing various architectural parameters in the

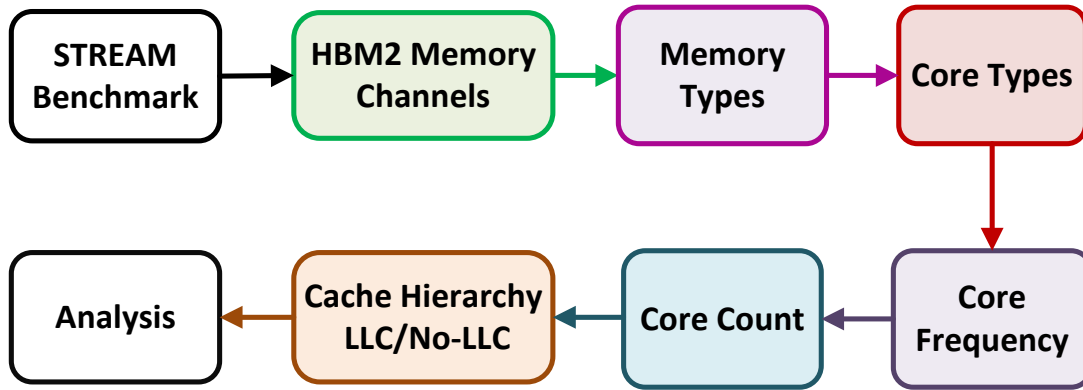


Figure 2.13 – Methodology for BW analysis using the STREAM benchmark.

system.

- **Input:** The STREAM benchmark is used as a benchmark on the ARM based platform. The input array size of the benchmark is set to 100 million, which is equivalent to a memory requirement of 2.2GB. The array size is chosen to have a trade-off between memory utilization and simulation turnaround time, which increases linearly with the increase in array size.
- **Memory Channels:** STREAM is executed for HBM2 with 1, 2, 4 and 8 memory channels, to observe the effect of channel count on BW.
- **Memory Types:** Both HBM2 and DDR4 memories are used as memory types to compare their performance. DDR4 is used with 1, 2 and 4 memory channels, as the number of channels scale from 1 to 4 in commercially available DDR4 based systems, when moving from low-power mobile devices to high-end server-class systems [86, 87, 88].
- **Core Types:** All the benchmark experiments are run for both ARMv8 in-order and OoO cores.
- **Core Frequency:** All the benchmark experiments are repeated at 1GHz, 2GHz and 4GHz core frequencies. The 4GHz frequency is used just for scaling analysis, as I am not aware of any ARM cores operating at 4GHz.
- **Core Count:** 8 and 16 cores are benchmarked to analyze the effect of core count on BW.
- **Cache Hierarchy:** Changes of the cache hierarchy are also looked into, namely, the effects of no-LLC system in comparison to system with LLC.
- **Analysis:** In addition to analysing the results after changing each architectural parameter discussed in this methodology, the results are analysed globally, and the experiments are run again with any changes in architectural parameters necessary.

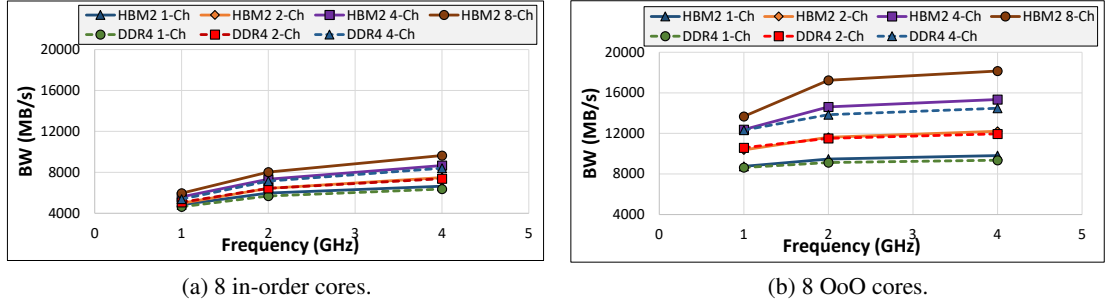


Figure 2.14 – BW scaling of HBM2 with respect to the number of channels and core frequency in comparison to DDR4, for both 8 in-order and 8 OoO cores when running the STREAM benchmark.

### 2.6.2 Bandwidth Analysis Results

Deploying the BW analysis methodology, I now look into the BW analysis results of HBM2 in comparison to DDR4.

- Figure 2.14 shows the scaling of HBM2 BW with the number of channels and core frequency, as compared to DDR4 with 1, 2 and 4 channels, for 8-core ARM in-order and OoO systems with an LLC (L2) of 1MB. It can be seen that the BW increases with the number of channels. HBM2 performs better than DDR4 by 5% for both in-order and OoO cores for the same number of channels, whereas 8-channel HBM2 gives up to 34% and 48% more BW as compared to single channel DDR4, for in-order and OoO cores, respectively. Moreover, 8-channel HBM2 gives up to 12% and 19% more BW as compared to 4-channel DDR4. BW scales with the core frequency for 8 in-order cores as in Fig. 2.14a. It can also be seen that OoO cores utilize much more of the available BW as compared to in-order cores at the same frequency. However, the BW scaling with frequency is low after 2 GHz. To investigate this further, STREAM is run with higher number of cores, as discussed next.
- The STREAM benchmark is run for 16 ARM in-order and OoO cores, both with LLC. Figure 2.15 shows that BW saturates for OoO cores at higher frequency, and BW utilization of in-order cores converge to that of OoO cores. If the BW of 8-OoO cores in Fig. 2.14b is compared to that of 16-OoO cores in Fig. 2.15, it is observed that the BW does not scale with the increase in the core count of OoO cores. However, it does scale with the number of in-order cores. The BW of 8-channel HBM2 is 44%-46% higher than 1-channel DDR4, and 17%-19% higher than 4-channel DDR4. Hence, I will only look into 8-channel HBM2 BW scaling analysis for now.
- Through the analysis of the data path between the compute cores and memory, it is observed that the LLC is the bottleneck on the available BW, which explains why OoO cores do not exhibit a linear scaling with the number of cores. On the other hand, the reduced BW scaling is also an indication of better performance as the memory is being accessed

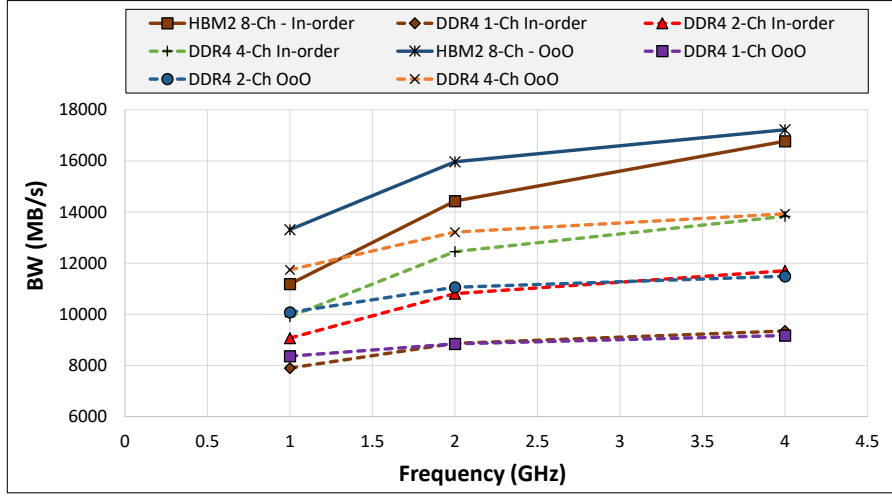


Figure 2.15 – BW scaling with frequency for 16 cores.

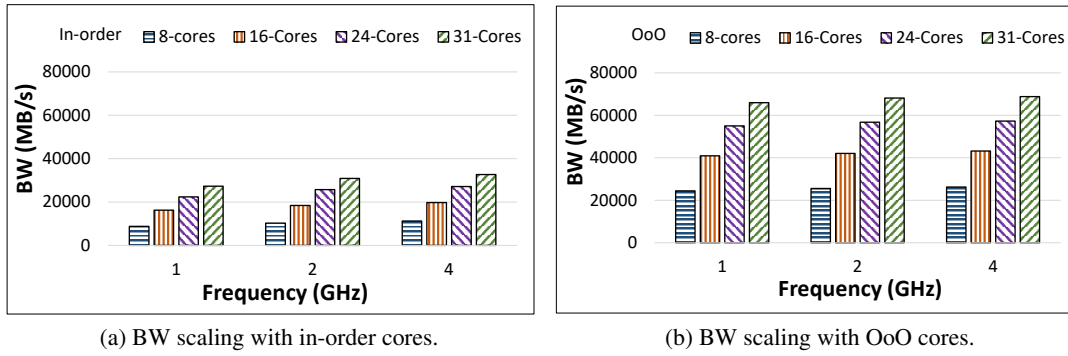


Figure 2.16 – BW scaling of HBM2 with core count and core frequency with no LLC.

less frequently due to the caching effects of LLC. However, the LLC is removed from the system and the STREAM benchmark is run again. As shown in Fig. 2.16, the BW scales with number of cores both for in-order and OoO cores without the LLC. From Fig. 2.16a and Fig. 2.16b, it is also evident that OoO cores can utilize almost 2x more BW as compared to in-order cores. These figures show that the BW utilization depends on the core count and type of cores. Thus, the BW provided by HBM2 is available, the greater the number of cores, the more its utilization. However, it is observed that the BW utilization does not scale linearly in relation to the core frequency and remains almost constant.

- The L1 cache is investigated to see if it is causing any bottlenecks in relation to BW scaling with frequency. The size of miss-status-holding-registers (MSHR)<sup>1</sup> is changed from 4 to 10. However, there was no change in BW for in-order cores, and around 16% BW

<sup>1</sup>MSHRs keep track of outstanding cache misses, thus enabling multiple accesses and outstanding misses in the cache

improvement for OoO cores when using 10 MSHRs instead of 4. However, the change with frequency scaling was still quite constant. Hence, the reason for almost constant BW being accessed across different frequencies is not due to L1, but results from the fact that the core frequency to issue memory request is high compared to HBM2 serving those requests. Thus, whenever the request goes to HBM2, the cores are stalled waiting for memory to respond to those read/write accesses. As STREAM is a memory intensive benchmark, changing the core frequency does not improve the stressed BW of the memory.

## **2.7 Power Models and Area**

In addition to performance analysis using gem5-X, I am also interested in the power and energy consumption of the systems. However, gem5 or McPAT power model, as proposed by [89] and [90], respectively, are not used as they both are for ARMv7 32-bit ISA, whereas I am using ARMv8 64-bit cores. For the CPU energy analysis, the power model for 28nm CMOS bulk technology node for ARM 64-bit in-order and OoO cores proposed in [46] are used. For in-order cores, the energy ratio between A57 and A53 cores at different frequencies as proposed by [91, 92] is used. The power model accounts for core active, wait-for-memory (WFM) and static energy (in J/cycle), and the LLC read and write energy (in J/access). For the memory power models, power values as reported in [93] and [94] are used for DDR4 and HBM2, respectively. Counters in gem5-X statistics like active CPU cycles, WFM cycles, cache read and writes hits and main memory accesses are used for power modeling.

For area estimates, the values reported in [92] and [91] for ARM OoO and in-order cores, respectively, are used.

## **2.8 Architecture Exploration and Optimization Methodology**

An architectural exploration and a flexible optimization methodology is presented for any given application using the gem5-X simulation platform. If the application has one kernel or if there is just one application, a single-step methodology comprising different phases is used. However, if there are multiple kernels in the application or multiple applications allocated on the system, then a two-step methodology is deployed, which encompasses the single-step methodology. Both methodologies are discussed in this section.

### **2.8.1 Single-Step Architecture Exploration and Optimization Methodology**

This architecture exploration methodology is used to optimize a single kernel or application in the system. The methodology, as shown in Fig. 2.17, has 3 phases: application characterization, architecture optimization and milestones. Each phase is then further divided into different stages. The different phases of the methodology are discussed in this section.

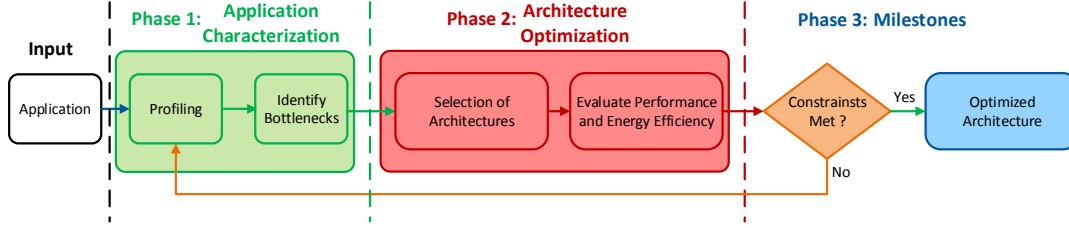


Figure 2.17 – Gem5-X single-step multiple-phase architectural exploration methodology.

### 2.8.1.1 Phase 1: Application Characterization

To optimize an architecture for any given application, the application is first profiled to identify bottlenecks.

- **Profiling:** Profiling provides two important insights. Firstly, it identifies the application kernels that are compute and memory intensive, and secondly, it gives information about the system resources being stressed by these kernels. In the gem5-X methodology, valgrind [36] is used as profiler when profiling on a hardware platform to collect performance counter statistics (e.g., instruction counts, cache misses, etc.). Valgrind adds some code instrumentation to the application, hence it is very slow. Therefore, the *gperf* statistical profiler developed by [33] is also used on both hardware and in the gem5-X simulator to profile applications with minimal profiling overhead, and generate call graph trees showing what percentage of the total time each kernel takes.
- **Identify Bottlenecks:** Once the profiling information is available, the kernels that consume most of the execution time can be identified. Then, using the profiling data, the resource utilization by these kernels can be further analyzed, providing insights about the resources that are bottle-necked.

### 2.8.1.2 Phase 2: Architecture Optimization

Once the application is characterized and bottlenecks are identified, the architecture is optimized by alleviating the bottlenecks and improving performance and energy efficiency using a two-stage process.

#### 1. Selection of Architectures

Gem5-X enables different architectural strategies and allows to select from a range of architectural extensions and tune various architectural parameters, as shown in Fig. 2.18. Once the bottlenecks are identified via profiling in phase 1 of the methodology, they provide guidance in selection of appropriate architectural choices. These architectural choices are



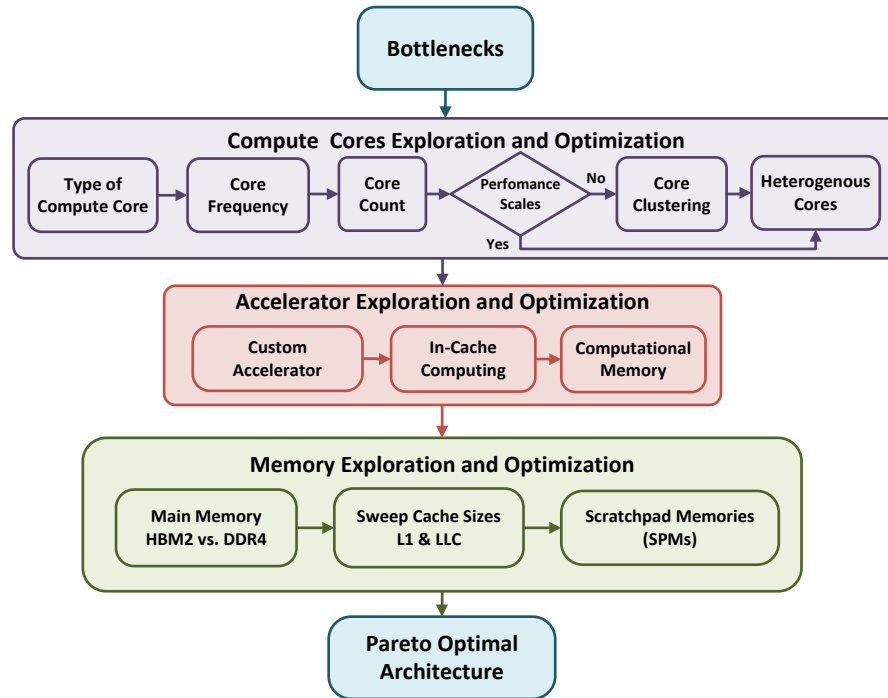


Figure 2.18 – Steps for selection of appropriate architecture.

divided broadly into three main steps: 1) Compute cores, 2) accelerators, and 3) memory exploration and optimizations.

- **Compute Cores Exploration and Optimization:** The compute core sub-system is explored and optimized in this step for selection of appropriate architectural choices, as depicted in Fig. 2.18, based on the bottlenecks identified. The type of cores, as either energy efficient in-order cores or high performing OoO, the core frequencies and the number of cores are varied to meet the required performance and power constraints. It is also checked if the performance scales with the core count. If it does not scale, core clustering extension of gem5-X is used with multiple instances of the application deployed on each individual cluster. Heterogeneous cores in gem5-X can also be utilized if it is efficient (with regards to performance and energy) to run some part of the application on one type of core and the others on another core type.
- **Accelerator Exploration and Optimization:** Accelerators can be used to overcome compute bottlenecks in an application. The choice and selection of accelerator is driven by profiling data. If a full-custom accelerator is required, gem5-X enables that either by modifying the system memory map or a custom instructions for the accelerator. One of the accelerators enabled by gem5-X is the in-cache computing architecture discussed in Section 2.3. If a kernel involves many operations upon the same data chunk, resulting in low memory access times but high computation cost and processor-cache traffic, it is a good candidate for in-cache acceleration. Moreover,

CM core accelerators are enabled by gem5-X, which can be integrated within the execution stage of the CPU pipeline. They are particularly useful when a kernel is both compute and memory intensive.

- **Memory Exploration and Optimization:** After optimizing the compute sub-system, the memory sub-system is optimized for an overall performance/energy-optimized system. This involves selection of the main memory as either traditional DDR4 or HBM2, as presented in Section 2.3. If the system is bottlenecked at the main memory, due to low available memory BW, the HBM2 is the better choice to speed-up the system by easing the memory bottleneck. Moreover, to select the appropriate cache size, it is possible to sweep the cache sizes at all levels like at L1 and LLC. Furthermore, depth of the cache hierarchy can also be explored to decide how many cache levels are required. Lastly, if the workload involves data transfer between the cores, SPMs extension in gem5-X can be utilized to bypass the cache hierarchy and enable direct data transfer between the cores.

### 2. Evaluation of Performance and Energy Efficiency

Once the optimized architecture is obtained using the strategies discussed in the previous section, the performance and energy efficiency improvements achieved are evaluated at the system level, as well as application level. I also look at the cost in terms of area of the optimized system.

#### 2.8.1.3 Phase 3: Milestones

In the third and final phase of the gem5-X methodology, it is checked if the optimized system meets the power, performance and area constraints. If any one of the constraints is not achieved, I go back to phase-1 and iterate over the whole methodology again. I iterate until all the constraints are met and milestones achieved, obtaining an optimized architecture. If multiple architectures meet the constraints, I use Pareto optimal architecture points, w.r.t. power, performance and area.

### 2.8.2 Two-Step Architecture Exploration and Optimization Methodology

I propose a two-step methodology to explore and optimize architectures for applications that are composed of multiple kernels, or when multiple independent applications are allocated simultaneously on a platform. Figure 2.19 shows the two-step methodology for architectural exploration and optimization of this multiple kernel scenario.

The first step is the local architectural optimization for each kernel based on its own bottlenecks and compute/memory requirements. The optimization is on both compute and memory fronts, exploiting their respective architectural parameters as presented in Fig. 2.19, Step 1, which is the same as single-step methodology shown in Fig. 2.17. Once all the kernels are independently architecturally optimized using the single methodology, they are co-allocated together and the architecture is globally optimized for all kernels, if further necessary, as in Fig. 2.19, Step 2.

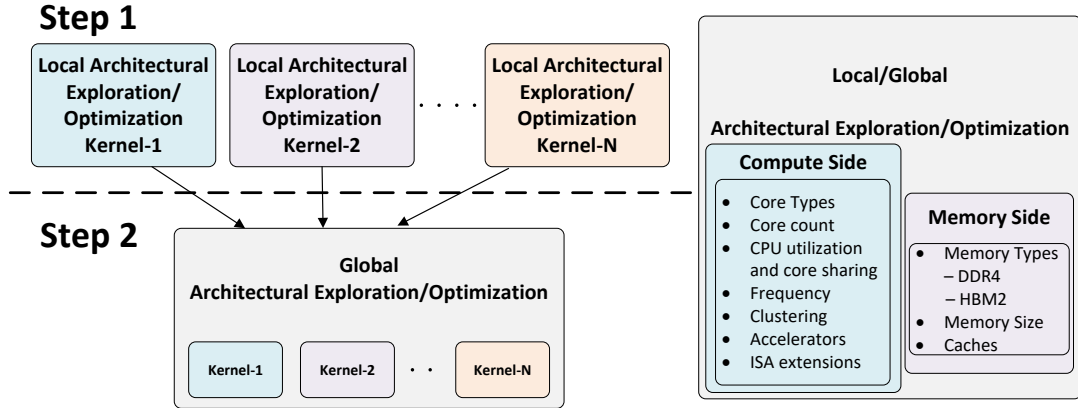


Figure 2.19 – Two-Step architectural exploration and optimization methodology.

The optimization strategy in Step 2 is the same as for local optimization in Step 1, both on the compute and memory sides, hence leaving limited room for further optimization, unless there are conflicting architectural requirements between different applications. In such a scenario, the focus is on minimum performance requirements or QoS for each kernel, and then further optimize the architecture.

The methodologies discussed in this section will be used in the following chapters to explore and optimize architectures for the case study applications.

## 2.9 Summary

In this chapter, I have presented the gem5-X simulation platform to enable exploration of many-core heterogeneous architectures to optimize performance and energy consumption in new emerging dynamic applications. Compared to gem5, the latest version of gem5-X supports validated ARMv8 in-order and OoO cores in FS mode with a validation error of up to 4%. Also gem5-X supports multiple heterogeneous ARMv8 cores along with heterogeneous memories including DDR4 and the new 3D stacked HBM2 in FS mode, completely integrated and tested, which gem5 does not support straight OOB. In addition to heterogeneous compute cores, gem5-X enables exploration of clustering configurations of compute cores. Moreover, gem5-X supports accelerators to be integrated within the system like the BLADE in-cache computing engine, by capitalizing on the ISA extension support. Furthermore, in comparison to gem5, gem5-X enables profiling support using the gperf profiler, supports advanced check-pointing to help reduce the simulation turnaround time and comes with WA to enable file sharing between host machine and the simulated system. All in all, in this chapter I have also proposed a new two-step architectural exploration and optimization methodology of new many-core architectures for new dynamic applications and benchmarks. Moreover, using the system configuration capabilities of the gem5-X framework and exploration methodology for many-core systems, I also analyzed the benefits of HBM2 vs. DDR4 in the STREAM benchmark.

The gem5-X simulation framework is open-sourced to the community, enabling OOB, fast simulation of many-core ARM 64-bit heterogeneous architectures with innovative architectural extensions. It is readily available for download online from <https://esl.epfl.ch/gem5-x> [95]. A technical reference manual for gem5-X has also been published online [96], which includes a quick start guide, as well as instructions on how to use different architectural extensions and support enhancements in gem5-X.

The work in [97], [98], [38] and [37] were published as a result of this chapter. The main contributions from [97] are the gem5-X architectural extensions, including ISA extensions and in-cache computing engine along with gem5-X support enhancements and single-step exploration methodology. The validation of gem5-X is also one of the main contributions in [97]. The work in [98] extended the work in [97] with support for core clustering, heterogeneous cores, heterogeneous memory sub-system along with HBM2 BW analysis and the two-step architectural exploration methodology. The main contribution of the work in [38] and [37] is the implementation of BLADE in-cache computing engine in gem5-X.

## 3 Compute-Dominated Architecture Exploration

### 3.1 Introduction

The adoption of digital products along with their associated online cloud-based services are experiencing an unprecedented growth. We interact and use these services and products in a variety of different spheres of our daily lives. Among them, video streaming services like YouTube, Netflix, etc., machine learning and Convolutional Neural Networks (CNNs) based video analytics applications like surveillance, security, drone navigation, self driving cars, personal digital assistants, augmented reality, etc., and cloud based services like social networking, online banking etc., are some of the noticeable ones. These applications are deployed all the way from cloud data centres and High Performance Computing (HPC) systems to the edge nodes and mobile devices. Consequently, there has been a rapid growth in the number of data centers in the world, leading to unsustained energy consumption, estimated to be at 1% of the global energy demand in 2019 [15]. This results in an escalating demand to meet the power and performance requirements of the server platforms hosting these different services and applications. Moreover, these platforms comprise many-core processors to meet the performance and Quality-of-Service (QoS) constraints while hosting a range of multi-threaded applications for multiple users, simultaneously, as described by [67]. The applications and services on these multi-core systems are either compute-dominated or memory-bounded. In case of compute-dominated applications, to optimize for performance, power and area (PPA), optimizing the compute subsystem maximizes the optimization for PPA. Memory optimization will also improve the PPA for these compute-dominated app, as memory is also the essential part of the system, but with a lesser impact.

In this chapter, I will present and explore optimized architectures using gem5-X simulation platform, for state-of-the-art emerging compute-dominated applications, including video encoding, video analytics comprising of video encoding and CNN-based image classification, Binary Neural Networks (BNNs), Recurrent Neural Networks (RNNs) like Long-Short-Term-Memory (LSTM) and banking workload in Virtual Machines (VMs). These workloads are deployed all the way from cloud servers and data centres to mobile devices. I will use these

### Chapter 3. Compute-Dominated Architecture Exploration

---

applications, as case studies to demonstrate the use of gem5-X methodology discussed in Chapter 2 and the compute sub-system architectural extensions of gem5-X, also presented in Chapter 2. These applications are case studies to show-case the capabilities and architectural extensions added to gem5-X, which is generic and can be used to explore and optimize architecture for any other application. Thanks to the gem5-X architectural extensions presented in Chapter 2, including Instruction Set Architecture (ISA) extensions, accelerator support (like in-cache computing), core clustering, heterogeneous compute cores, Computational Memory (CM) and 3D stacked High Bandwidth Memory v2 (HBM2), which enables to explore and optimize the architectures for improved performance and energy efficiency for the case study applications. Furthermore, the gem5-X support enhancements, helps not only to reduce the simulation turnaround time, but also allows to simulate the case study applications with different architectural extensions in the Full System (FS) mode with full Linux stack, as in a real system.

The main contributions in this chapter, as well as its organization is summarised as follows:

- Overview of state-of-the-art and related work for architectures of different case study applications is presented in Section 3.2.
- Architectures for real-time video encoding application for various video resolutions are explored and optimized using the gem5-X optimization methodology and utilizing the in-cache computing accelerator in Section 3.3. This provides performance benefits of up to 15% and energy benefits of up to 76% over an ARM-based baseline system.
- A heterogeneous architecture is proposed for a complete video analytics application, comprising of video encoding and CNN-based image classification, in Section 3.4. In addition to the heterogeneity on the compute side comprising ARM in-order cores, Out-of-Order (OoO) cores and in-cache computing engine along with clustering of the compute cores, high bandwidth 3D stacked HBM2 memory is also being utilized to alleviate any memory bottlenecks in the system. Gem5-X two-step methodology is used for architectural exploration and optimization for this case study application. Performance benefits of up to 30% and energy savings of up to 54% are achieved when using heterogeneous cores along with HBM2, while meeting the performance constraints of the video analytics application.
- Near-Threshold-Computing (NTC) servers are proposed for VMs in the cloud in Section 3.5, enabled by gem5-X VM support enhancement. Energy and performance trade-off with regards to QoS constraints for cloud workloads are discussed and presented in Section 3.5.
- In-memory computational cores based on Resistive Random-Access Memories (RRAMs) and Analog In-Memory Computing (AIMC) engine are proposed for accelerating BNNs and RNNs, respectively, in Section 3.6. These CMs are integrated within the execution stage of the Central Processing Unit (CPU) pipeline, thanks to the gem5-X ISA extensions. Performance and energy benefits of 12.4x and 12.3x, respectively, made possible due to the tightly coupled integration of the CMs in the CPU, are also presented in Section 3.6.

## 3.2 Related work

### 3.2.1 Video Encoding

Video encoding is an essential step in online video streaming, to reduce the size of actual data being transmitted. High Efficiency Video Coding (HEVC) [99] is a state-of-the-art video encoding standard and offers twice the compression of its predecessors, but at the cost of significantly increased computational cost [100]. To improve the throughput, the authors in [101], parallelize the HEVC encoder on multi-core processors, using tiles and Wavefront Parallel Processing (WPP) techniques. The work in [102], accelerates the HEVC encoder via a Field-Programmable Gate Array (FPGA) based accelerator. HEVC encoder on ARM-based multi-core platform is demonstrated in [103], using the parallelization techniques discussed in [101]. In all these works, the effort in improving the performance and energy is focused on either multi-threading or using hardware accelerator.

I propose to use in-cache computing engine, integrated in the L1-D cache of the CPU, to process the compute intensive kernels of the application with high cache locality. The in-cache computing accelerator is capable of processing a large working data-set, similar to a Single-Instruction-Multiple-Data (SIMD) format. Hence, I propose a heterogeneous system comprising of CPU cores with accelerators tightly coupled to them in their respective L1-D caches, in a multi-core system. The in-cache accelerator provides both performance and energy gains, firstly by processing a larger data-set and secondly, by eliminating the data transfer overhead between the CPU and the L1 cache.

### 3.2.2 Video Analytics

Real-time video analytics is an emerging application which is used in variety of fields such as video surveillance, traffic monitoring, augmented reality and self-driving smart cars [40]. Graphic Processing Units (GPUs) based video analytics have been proposed in [104] and [105] for high performance and throughput. For increased performance along with energy efficiency, Application-Specific Integrated Circuit (ASIC) and FPGA based video analytics hardware accelerators have also been proposed, as in [106, 107]. However, quite a lot of effort and time is required for the development of an ASIC. Furthermore, applications developed to run on ASIC cannot be updated/upgraded if there are new improvements to the algorithm. Therefore, I propose a CPU-based video analytics architecture, which is deeply heterogeneous along with 3D HBM2 memory, with different applications kernels running on different types of compute cores, along with an in-cache computing accelerator. The in-cache computing accelerator is not application specific, but runs basic computational blocks of the application. The performance and energy efficiency of the proposed CPU-based architecture with 3D stacked HBM2 is compared to that of an embedded low power GPU, and it is demonstrated that the CPU-based system is better for video analytics both in terms of performance and energy efficiency.

### 3.2.3 Near-Threshold-Computing (NTC) Servers for VMs in the Cloud

Recent work in the area of energy-efficient server design focuses on presently-shipping enterprise servers, with traditional x86 architectures [108]. These servers had traditionally been designed to meet performance goals, without energy efficiency as a design constraint. Only recently, with the stagnation of Dennard Scaling [18], and the resulting power-limited servers, NTC turned into a key technology to improve energy efficiency. Previous work on near-threshold many-cores mainly focused on single voltage domain and multiple frequency domain architectures [109]. However, other recent works on processors in FD-SOI demonstrated the near-threshold capabilities of the technology, capable to run a dual-core CortexA9 processor at 1 GHz at the supply voltage of 0.6V [110]. The work presented in [25] was the first one proposing the usage of NTC servers in Ultra-Thin Body and Buried Oxide (UTBB) FD-SOI technology. The proposed server architecture [25] was based on ARM cores optimized for latency-critical scale-out applications. Nonetheless, the power model proposed in that work did not include a detailed characterization of the uncore components. Moreover, the target Cavium ThunderX servers [111] were neither based on FD-SOI technology nor validated for virtualized applications. A new and much more accurate power model using 28nm FD-SOI process technology is utilized, along with a server architecture resembling available ARM-based ThunderX server, but optimized to run virtualized applications, as proposed in [46]. The power model of the near-threshold FD-SOI processors is based on the same characterization adopted by the authors of [25], but extensively modified and extended based on an exhaustive characterization of the different components of the Cavium ThunderX server (L2 memory power, memory controller, IOs, DDR memory) according to extensive measurements performed on the real hardware, fitted and ported to the 28nm FD-SOI technology to be used for the exploration performed using gem5-X.

### 3.2.4 In-Memory Computation of AI Workloads

#### 3.2.4.1 Accelerating BNNs with RRAMs

The need to deploy CNN on edge devices enabling the mobile devices and edge sensors for image classification and detection has led to the creation of new CNN models capable of being executed efficiently in such compute and energy constrained systems [45]. In CNNs, approximately 90% of the operations realized are convolutions [112]. Hence, convolutional layers became the main targets for optimization in CNNs. Consequently, two main approaches aiming at optimizing the execution of such layers were adopted: using dedicated hardware accelerators; and reducing the precision of the operands.

The use of custom accelerators attempts to optimize the execution of CNNs by exploiting hardware level parallelism [113, 114] and offloading workload to near-data accelerators [115, 116]. However, most of these solutions do not comply with the limitations of edge devices since they require a significant amount of hardware resources to be implemented, which is hardly feasible in the context of such energy and cost constrained systems. Furthermore, accelerators do



not provide performance benefits whenever the workload associated to the dataset is not enough to overcome the communication overhead with main memory [117]. For that, the datasets have to be significantly big, and since edge devices are usually used for processing rather small datasets, accelerators may also not be suited for such systems.

CNN models using low-precision operands, such as bfloat16 [118], were also created to speed computation in convolutional layers. Ultimately, BNNs [47, 48] only use one bit to represent the weights. Moreover, in XNOR-Net BNNs [47] both the input and the weights of the convolutional layers are binary, thus convolutions are performed by simply executing the bit-wise XNOR of the input and kernel followed by a bitcount. Although accuracy is sacrificed to some level by such heuristics [47], the resultant memory savings and performance improvements allow some XNOR-Net CNNs to be executed by edge devices. This allows such systems to execute complex workloads, such as facial recognition, which is a real-time task that has to be performed efficiently [119]. Moreover, in most BNNs, the convolutional kernels are rather small (typically  $3 \times 3$ ,  $5 \times 5$  or  $7 \times 7$ ), thus it is expected that a significant part of their data is redundant. For instance, in  $3 \times 3$  binary kernel, there can only be  $2^9 = 512$  possible combinations. There will be reuse of these 512 kernels, hence, resulting in redundancy in BNNs.

Exploiting the redundancy in BNNs, I propose a Binary Dot-Product Engine (BDPE) that locally stores the most used kernels in RRAMs and efficiently implements binary convolution to accelerate convolutional layers. Since the BDPE is meant to be integrated into the pipeline of a CPU, it does not introduce communication overheads, which is one of the main drawbacks of using dedicated accelerators.

#### 3.2.4.2 Accelerating LSTMs with Analog In-Memory Computation

Matrix-vector multiply (MVM) are one of the main operations in various Deep Learning (DL) workloads such as RNNs, CNNs, and Multi-Layer Perceptrons (MLPs) [120, 121]. MVM operations are especially amenable for in-memory acceleration, paving the way for significant energy gains and speed-ups for DL workloads.

One approach to exploit in-memory computing for DL is to design stand-alone accelerators where multiple CM cores and associated digital logic blocks are interconnected by a suitable communication fabric [122, 123]. In such an accelerator, weights associated with different neural network layers can be mapped to different CM arrays and data can be propagated via array-to-array communication. Yet, these works do not offer great implementation flexibility. The data flow and array-to-array connectivity may have to be revised for different neural network models. Moreover, the digital logic block typically offers limited functionality. For instance, only a small set of activation functions can be supported. DL algorithms change much faster than the development time of custom hardware accelerators; the accelerators should offer sufficient flexibility for supporting diverse workloads. One way to enrich the digital logic supported by the CM accelerators is to add local CPUs [124, 125]. However, these implementations lack a

complete full-stack hardware and software ecosystem.

One prominent approach for moving towards a more general-purpose architecture is to bring computing closer to the CPU with in-cache computing [38]. Here, the SRAM cache array's bit line/word line structure is exploited to perform a large number of parallel bitwise and arithmetic computations in a SIMD-like manner. However, the cache size (only on the order of kilobytes) limits the size of the operands and the number of operations that can be executed in parallel.

I propose a novel system architecture, where AIMC cores [126] are integrated directly into the pipeline of a general-purpose CPU. The design allows megabytes of data to be stored and processed in parallel in AIMC cores in only constant time complexity, while still having access to the rich digital capabilities of the CPU. The existing hardware and software stack can be leveraged with extensions to accommodate the AIMC core. RNN based LSTMs, [50] which are quite well suited for natural language processing [63], are used as case study DL workload for the AIMC core.

### 3.3 In-Cache Computing Accelerator for Video Encoding

#### 3.3.1 Video Encoding

To demonstrate the gem5-X framework and methodology, real-time video encoding is used as one of the case study applications. Video encoding is chosen because video streaming represents 58% of the overall downstream traffic in 2018, as presented by [35], hence, the need for a performance-energy-optimized video encoding application. For this purpose, Kvazaar [1], a state-of-the-art open source HEVC application, compliant with H.265 coding standard, is used as real time video encoding application. HEVC offers twice the compression of its predecessors, but at the cost of significantly increased computational cost [100].

In the HEVC encoder, the most complex block is the motion estimation of the video, which plays a critical role in compression (and therefore, bandwidth) and quality. The goal of online encoding is to serve videos to users in real-time, i.e. achieving a sustained frame rate of 24 frames-per-second (FPS), regardless of the video resolution. Real-time HEVC encoding is achieved by means of thread-level parallelization of different blocks. Kvazaar comes with a wider range of parallel processing capabilities. It is very well optimized from the software perspective, leaving limited headroom for further software-based optimization.

In accordance with the gem5-X methodology, as discussed in Chapter 2, I first profile and assess the bottlenecks of real-time video encoding application when running on ARM-64 in-order and OoO architectures equipped with the NEON SIMD accelerator [127]. To demonstrate the capabilities of gem5-X for exploiting architectural extensions, BLADE, the novel cache computing engine described in [37] and discussed in Section 2.3.1.3, is used for accelerating Kvazaar video encoding. BLADE executes a large number of operations simultaneously in-cache,

### 3.3. In-Cache Computing Accelerator for Video Encoding

Table 3.1 – Initial architecture for video encoding.

Parameter	Value	Parameter	Value
Core ISA	ARMv8 64-bit	# In-order, # OoO cores	4, 4
Core Frequency	2GHz	DDR4 size	4GB
L1-I cache, L1-D cache	32kB, 32kB	LLC	1MB

being a promising solution to reduce the computation time. The benefits of in-cache computing are assessed in terms of performance and energy consumption from the system-level perspective across varying frequencies and core counts, within a fixed area limit.

#### 3.3.2 Experimental Setup

- **Application:** To demonstrate the gem5-X methodology, experiments are run using Kvazaar for video encoding for three video resolutions: high, medium and low, which correspond to 1920x1080, 416x240 and 176x144 pixels, respectively.
- **Power Model:** To compute energy values, the power model for 28nm bulk CMOS A57 OoO cores is used, as proposed in Section 2.7 of Chapter 2. For in-order cores, the energy ratio between A57 and A53 cores at different frequencies is used, as proposed by [91, 92]. The power model includes core active, wait-for-memory (WFM) and static energy (in J/cycle), and the Last-Level-Cache (LLC) read and write energy (in J/access).
- **Hardware Architecture:** As a starting point for the architecture exploration, the ARM JUNO platform [34] is modelled in gem5-X, with 4 OoO cores instead of 2 to have a fair comparison between different architectures and core types, each with 4 cores as a starting point. The simulated architecture is summarized in Table 3.1.

#### 3.3.3 Profiling and Bottlenecks

The first step in the gem5-X methodology is to profile the target application to identify memory and compute bottlenecks. Valgrind [36] is used first on the JUNO platform to profile Kvazaar. The profiling data shows that the Finite Impulse Response (FIR) filter and the Sum-of-Absolute-Difference (SATD) are the two main blocks in the application that represent 21% and 26% of overall instructions executed, respectively. The remaining 53% of the computation is spread in chunks of less than 10% (in average 7%). L1 cache read miss counts for FIR filter and SATD blocks were as low as 4.8% and 5.2%, respectively, demonstrating high data locality. Kvazaar is also profiled on the ARMv8 64-bit JUNO platform in both in-order and OoO using gperf. The profiling results are consistent in both types of cores as well as across the profiler used, in terms of the bottlenecked functions. Moreover, percentages only differ between 1% to 4% from Valgrind to gperf due to gperf's statistical profiling.

Profiling demonstrates that the FIR filter and SATD blocks are the primary bottlenecks in Kvazaar.

Due to the high locality in the L1 cache and the relatively simple arithmetic operations they perform, they are potential candidates for the case study architectural extension of in-cache computing.

### 3.3.4 Gem5-X Extensions for Video Encoding

To increase the performance and explore different energy-efficient platforms for video encoding on the basis of the profiling step, two gem5-X compute sub-system architectural extensions will be utilized:

1. **BLADE in-cache computing accelerator:** The BLADE in-cache computing accelerator is integrated in the L1-D cache of the CPU, as discussed in Section 2.3.1.3 of Chapter 2. When the CPU decodes the in-cache computing instruction, the required operations are scheduled and its operands are loaded into cache from main memory. Data locality constraints are enforced upon the operands to guarantee architectural accuracy, as discussed by [128] and [37]. In order to perform an operation between two operands, they must share the same bitlines. This is ensured by reserving 1GB of cacheable memory that can be *mmaped* by an application, allowing fine grained control of where operands are stored in the cache. The target application, Kvazaar in this case, is modified to guarantee data alignment during operations to be performed in-cache.
2. **ARMv8 ISA extensions:** In-cache computing architecture is supported in FS mode in gem5-X, by extending the ARMv8 ISA [39], using reserved op-codes. The added instruction, when decoded, issues an in-cache computing request to the cache controller. A new *cachecompute* flag with the instruction is also added, so that the cache controller can recognize it as a cache compute request and handle it accordingly. This instruction can be issued through in-line assembly from any C or C++ program.

### 3.3.5 Strategies for Architecture Optimization

#### 3.3.5.1 Sweeping the Cache Sizes

As discussed in the previous section, FIR filter and SATD, the primary bottlenecks in Kvazaar, exhibit high cache locality. To explore the effect of cache size, I use gperf in gem5-X and vary the size of the L1 and LLC, in accordance with Phase-2 of the single-step exploration methodology presented in Section 2.8. It is observed that for an L1 of 32KB, varying the LLC size from 512KB to 16MB provides 6% application speed-up. Similarly, for an LLC of 16MB, increasing the L1 from 8KB to 128KB provides a 3.2% speed-up. As the speed-up is not significant, indicating that data fits in all cache sizes, a 32KB L1 cache and a 1MB LLC is selected for the remainder of architecture exploration for Kvazaar video encoding, as they represent an adequate trade-off between energy, performance, and area.

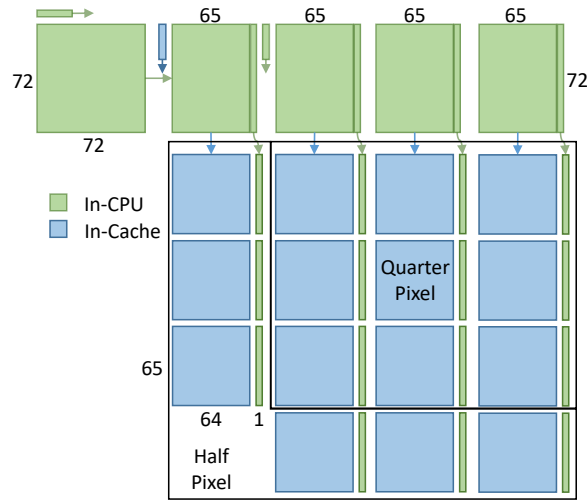


Figure 3.1 – Half and Quarter Pixel FIR Filter (numbers represent size in pixels).

#### 3.3.5.2 Acceleration with In-cache Computing

FIR Filter and SATD blocks are accelerated using in-cache computing, as they are the primary bottlenecks in Kvazaar and exhibit high cache locality in the L1 cache. Each block involves many operations upon the same data chunk, resulting in low memory access times, but high computation cost and processor-cache traffic, making them good candidates for in-cache acceleration.

1. **FIR filter:** The FIR filter block provides half and quarter pixel interpolation between two blocks of pixels, typically in 72x72 pixel blocks. Four 8-tap filters are utilized to perform interpolation. Four intermediate pixel blocks are computed first, and then six half pixel and nine quarter pixel blocks are computed, resulting in two instances of filtering being performed over the pixel blocks. To support the necessary data alignment as described in Section 2.3.1.3 in Chapter 2, the first 4 instances of FIR filtering are performed in-CPU and the results are stored in the necessary data aligned state in the 1GB of reserved space. This allows the next 15 instances of FIR filtering to be performed in-cache. Each instance of FIR filtering over a 72x72 pixel block requires 4'160 instances each of 32 bit multiplications, adds, shifts, greater than, and less than calculations, all of which can be performed in-cache, resulting in 312'000 in-cache operations being performed per function call. Finally, as FIR filtering over a 72x72 block of pixels results in an output block of 65x65, resulting in an unaligned data block, I trim the extra column of pixels and perform these calculations in-CPU, storing them in a separate array and maintaining data alignment, as demonstrated in Fig. 3.1.
2. **SATD Block:** SATD is performed between each of the half and quarter pixel blocks computed as described in the previous section, and a reference pixel block, and involves subtracting each pixel in the reference block from its corresponding pixel in the computed block. The Hadamard frequency transform of these blocks is then taken in the horizontal

### Chapter 3. Compute-Dominated Architecture Exploration

Table 3.2 – FIR filter speed-up for different video resolutions when using BLADE. Application acceleration and energy reduction for different video resolutions, when using BLADE in-cache computing engine (for both FIR filter and SATD block) with ARM in-order cores in comparison to ARM in-order cores with NEON SIMD.

Video Resolution	Filter Speed-up	Application Speed-up	Overall Energy Reduction
Low Resolution	88%	15.36%	11.5%
Medium Resolution	62.34%	14.7%	14.1%
High Resolution	64.64%	14.4%	16.47%

and vertical direction. The absolute values of these transformed values are added together for a single final value that represents the difference between the original and computed block.

Similar to the FIR filter, the SATD block can be accelerated by performing the subtractions and horizontal Hadamard transform in-CPU, and the results saved to the reserved memory. Then the absolute vertical Hadamard transform can be performed in-cache. To take advantage of data locality, each 64x64 pixel block is tiled into sub-blocks of 8x64 pixels, all operations performed on these tiles atomically. The vertical Hadamard transform requires 2'048 additions, 2'048 subtractions and 4'096 absolute value operations per 64x64 block. 4'096 additional additions are needed to reduce the absolute results to a single cost value, 3'584 of which are performed in-cache.

Table 3.2 shows the speed-up of the individual FIR filter and the overall application speed-up obtained via in-cache computing with 4 in-order cores, using gem5-X. Note that all comparisons in this section are against in-order (and OoO) ARM architectures equipped with a Neon SIMD accelerator. The FIR filter block is accelerated by 62% to 88% depending on the video resolution. The SATD acceleration is lower, reaching ~5% maximum, as not all SATD blocks can be accelerated, due to the alignment constraints on the operands for in-cache computing, as discussed in Section 2.3.1.3. However, accelerating SATD blocks is still very beneficial in terms of energy. The BLADE in-cache computing engine reduces the energy consumption, as it operates on larger data set using the SIMD approach, as discussed in Section 2.3.1.3 of Chapter 2. Furthermore, it reduces the data movement between cache and the functional units in the CPU, as the operations are performed in cache, which reduces the energy consumption. Accelerating both the FIR filter and the SATD block gives ~15% application speed-up across low, medium and high resolution videos compared to an in-order core without in-cache computing. Moreover, as presented in Table 3.2, an overall energy reduction of 11.5% to 16.47% is also achieved when both FIR filter and the SATD block are accelerated with BLADE, for different video resolutions.

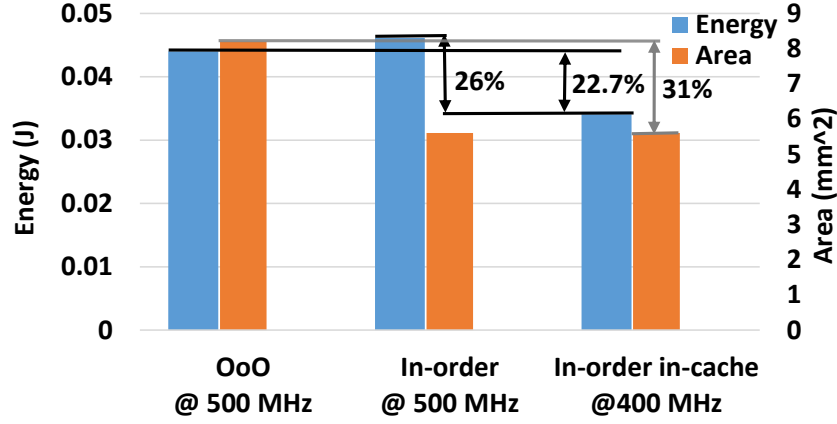


Figure 3.2 – Energy and area comparison for low resolution video.

### 3.3.6 Architectural Exploration and Results

Gem5-X allows to assess different architectures with and without in-cache computing. For the Kvazaar video encoding, the goal is to minimize the energy consumption while meeting the 24 FPS requirement for all video resolutions.

#### 3.3.6.1 Low Resolution Video

I start by assessing all configurations capable of satisfying 24 FPS for low resolution videos within a fixed area budget of 4 OoO cores, i.e  $8.2mm^2$ , as described by [92]. Figure 3.2 shows the energy consumption and area for various systems when running 24 FPS of a low resolution video. It can be seen that 8 in-order cores with in-cache computing at 400MHz are 22.7% and 26% more energy efficient when compared to 4 OoO cores at 500MHz and 8 in-order cores at 500MHz, respectively. Since in-order cores are 3 times smaller than OoO cores [92, 91], 8 in-order cores take 31% less area w.r.t. to 4 OoO cores. The area overhead of the L1 in-cache computing unit is 0.5% of the core area using the estimates by [75] and [129], which is not significant. Hence, it will not be considered for the rest of the chapter. The number of cores and frequency values are optimal, in terms of area and energy efficiency, when doing a sweep across different core counts at different frequencies.

#### 3.3.6.2 Medium Resolution Video

Figure 3.3 shows the energy consumption and area for optimal in-order and in-order with in-cache computing systems in comparison to OoO system when satisfying 24 FPS of a medium resolution video. It can be seen that 8 in-order cores with in-cache computing at 950MHz gives 44% energy benefit w.r.t. 4 OoO cores at 1.2GHz and area benefit of 31%. In comparison to 8 in-order cores without in-cache computing at 1.1GHz, the in-order system with in-cache computing is 36% more energy efficient. The number of cores and the operating frequencies for each architecture is optimal in terms of energy and area while meeting the 24 FPS requirement. They are obtained by

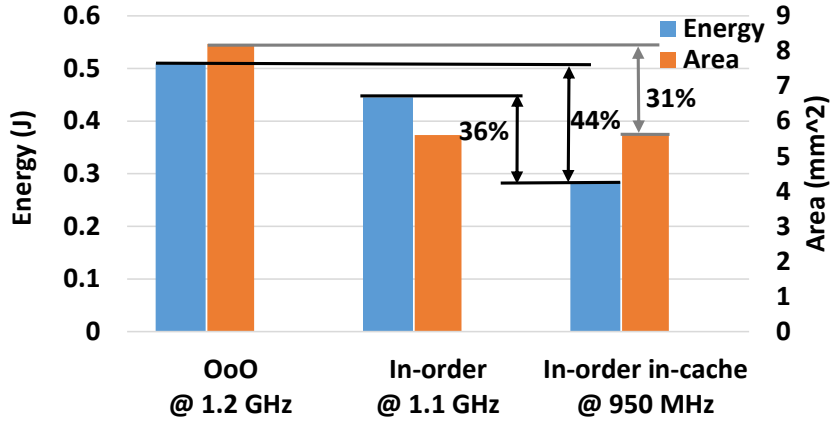


Figure 3.3 – Energy and area comparison for medium resolution video.

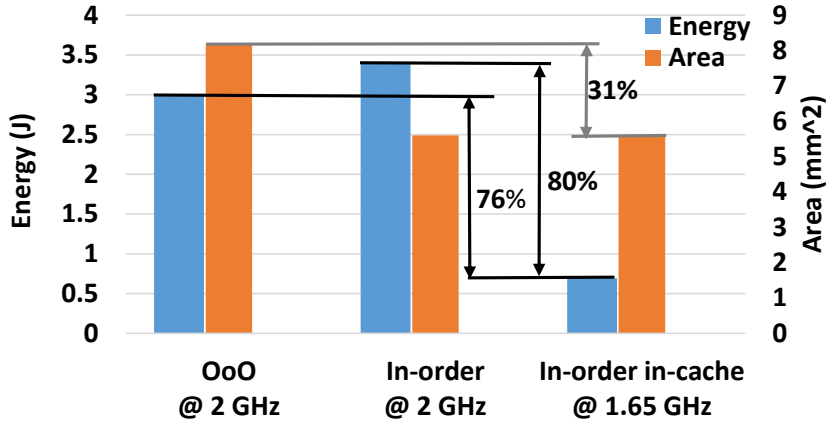


Figure 3.4 – Energy and area comparison for high resolution video.

doing a parameter sweep across both the number of cores and their frequencies.

### 3.3.6.3 High Resolution Video

Finally, the high resolution videos are assessed, with results consistent to that of lower resolutions, as shown in Fig. 3.4. In-order cores with a core count of 8 operating at 1.65 GHz with in-cache computing accelerator provide the best results in terms of energy efficiency, with energy savings of 76% in comparison to 4 OoO cores operating at 2GHz and 80% in comparison to 8 in-order cores operating at 2GHz. The area benefit for 8 in-order cores with in-cache computing is 31% in comparison to 4 OoO cores. Sweeping the core count and their operating frequencies enabled me to obtain the optimal architectures in terms of energy efficiency and area.

Figure 3.4 also reveals that the energy savings of in-order cores with in-cache computing in comparison to both in-order and OoO cores without in-cache computing, increases for higher video resolution as compared to lower and medium resolution videos, as shown in Fig. 3.2 and Fig. 3.3, respectively. This situation implies that, as the computational requirements increase, the



### 3.3. In-Cache Computing Accelerator for Video Encoding

Table 3.3 – Speed-up using HBM2 instead of DDR4.

Application	Medium Resolution + Alexnet Speed-up	High Resolution + Alexnet Speed-up
Kvazaar Speed-up	7.72%	2.8%
Alexnet Speed-up	8.35%	7.48%

in-cache computing performs better as it has larger data chunks to process and achieves more energy savings. The energy benefits obtained for all three video resolutions are attributed to the lower operating frequency used, the in-order cores and the in-cache computing unit.

#### 3.3.7 Video Encoding on Many-core Multi-application Systems

To demonstrate the many-core and multi-application simulation capabilities of gem5-X, a realistic server scenario is simulated by concurrently executing Alexnet image classification inference [62], along with real-time encoding (provided by Kvazaar), as in a video analytic application. This will be explored in much more detail in the next Section 3.4 for architecture exploration of video analytics application. In this section, I want to showcase the multi-application capabilities in gem5-X. Alexnet [62] is a CNN-based image classification algorithm. An Alexnet model in the ARM Compute Library (ACL) framework, developed by [130] is used. The system uses in-cache computing to accelerate the dominant blocks in Kvazaar. The in-simulator gperf profiler is useful in this case to look for new bottlenecks in the Kvazaar application, on the architecture with in-cache computing, when co-located with Alexnet, which is a memory intensive application. Analysis of the Kvazaar application for medium resolution and high resolution video reveals that memory functions, like *memcpy* and *memset* together contribute 16.5% and 10% respectively towards execution time. Profiling Alexnet reveals a 23.5% *memset* function contribution.

To accelerate the memory functions described above, HBM2 is used instead of the conventional DDR4 due to its high bandwidth (BW). The experimental setup includes HBM2 with 8 in-order cores. Kvazaar is allocated to the 4 cores equipped with in-cache computing, while Alexnet is allocated to the remaining 4 cores, all operating at 2GHz. Table 3.3 shows the percentage speed-up achieved when using HBM2 instead of DDR4. It can be seen that AlexNet is accelerated by more than 7% for both medium and high resolution videos. Moreover, Kvazaar achieves a higher acceleration in case of medium resolution as compared to high resolution, because the contribution of memory functions is higher in medium resolution videos as compared to high resolution ones.

To conclude, I have demonstrated in this section the architecture exploration and optimization of a video encoding application using the gem5-X methodology, architectural extensions and support enhancements. The fast design space exploration enabled by gem5-X allowed to properly select the optimal number, type and operating frequency of the cores giving an optimized architecture. To further demonstrate the system-level capabilities and functionality of gem5-X, I co-simulated

real-time encoding along with a deep learning CNN and alleviated the memory bottlenecks using HBM2. The gem5-X FS support enabled me to explore and optimize the application in a Linux based system, as in a real platform. Moreover, the gperf support enhancement enabled me to analyse the performance gains of BLADE for the individual kernels (i.e., FIR filter and the SATD block). Furthermore, the enhanced checkpointing support in gem5-X made it possible to reduce the simulation turnaround time by 20-30 minutes per simulation, as when resuming from the checkpoint, the core operating frequency as well as cache sizes can be changed. This enables multiple experiments to be resumed from the same checkpoint.

### 3.4 Heterogeneous Architecture for Video Analytics

Real-time video analytics dubbed as the next-generation "killer-app" [40], is used in a wide range of domains, from video surveillance, for security and monitoring, safety on construction sites, traffic monitoring and thermal monitoring to identify patients with fever as well as for fire hazards [41, 42]. Real-time video analytics is also used for autonomous drone navigation and rescue drone missions, e.g. during earthquakes, floods or rescuing people at sea [43, 131]. Autonomous drones are also being deployed for deliveries of parcels by many companies around the globe. The emerging market of smart autonomous cars along with their Advance Driver-Assistance Systems (ADAS), use video analytics as one of the core components of the system to detect obstacles, traffic signs and signals to navigate the car accordingly [44, 132]. Hence, this new emerging "killer-app" is being deployed all the way from low power edge devices to high performance cloud servers.

In this section, I demonstrate the capabilities of gem5-X to seamlessly analyze multi-threaded multi-kernel applications or multiple applications, enable various architectural extensions, and explore various architecture parameters, to have an overall performance-energy-optimized architecture. The video analytics application is used as a case study to demonstrate the architectural and system-level characterization capabilities of the gem5-X framework. Then, I demonstrate the utilization of gem5-X two-step optimization methodology based on local exploration and optimization for each individual kernel and then a global optimization for the whole application to have a complete optimized system, as discussed in Section 2.8.2. Each kernel can have its own optimized architecture, which might be different from another kernel, but collectively all those locally optimized kernels make up the complete optimized application. Hence, a heterogeneous system-level architecture is developed to ensure that all the kernels and system components are working coherently with each other.

Video analytics is a combination of two kernels, namely, video processing (or video encoding) along with image classification and detection. The video encoding kernel has to meet the performance and QoS constraints of processing at 24 FPS for a seamless user viewing experience. Image classification has real-time constraints in real scenarios like in surveillance to identify and alert of a potential security breach or safety alert on a construction site, collision avoidance in drone navigation and safety features and decision processing in autonomous cars and ADAS

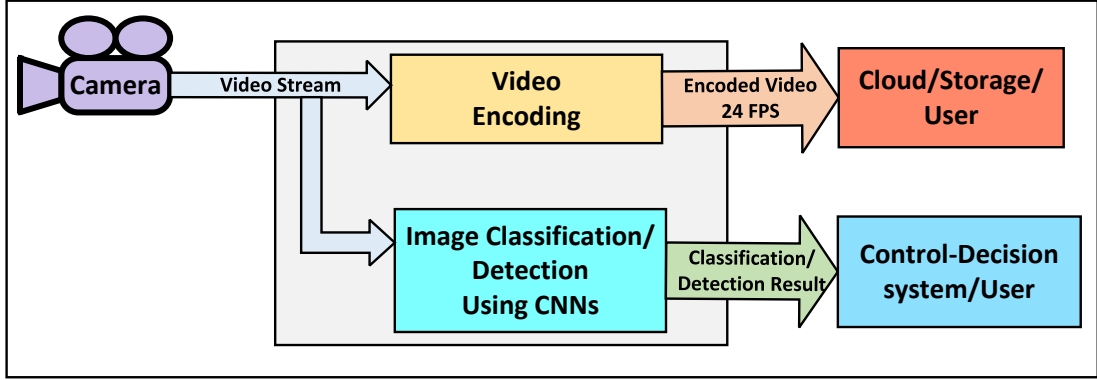


Figure 3.5 – Video analytics application composed of two kernels running concurrently, i.e., video encoding using Kvazaar [1] and image classification/detection using CNNs.

systems. These different use-cases of video analytics on different architectures, like surveillance in the cloud and drone navigation on the edge, require to have performance- and energy-optimized architectures while meeting the real-time constraints. The optimized architecture will enable longer battery life for edge devices and lower energy costs as well availability of resources to serve more users on the cloud. Hence, the proposed gem5-X system-level simulator is used to optimize the performance and energy of the complete system for multi-threaded applications on different architectures, either on the edge or in the cloud.

#### 3.4.1 Video Analytics Application

The gem5-X simulation framework supports full Linux stack and is generic enough to run and optimize any multi-threaded application, but to demonstrate its capabilities, real-time video analytics application is used as a case study. The choice of this particular application is due to two reasons. Firstly, video analytics is deployed and used in different spheres of our daily life, e.g., in surveillance for safety and security, drone navigation and autonomous cars. These different scenarios also present a challenge to system architectures because of different compute capabilities and energy constraints of the systems, all the way from the edge device to the cloud servers. The second reason for using video analytics as a case study application is because of the complexity of the application itself and the fact that it is composed of two distinct kernels, video encoding and image classification using CNNs. This gives me the opportunity to exploit and demonstrate the capabilities of gem5-X in optimizing various compute and memory sub-systems to have an overall optimized architecture.

##### 3.4.1.1 Video Analytics Application Structure

As depicted in Fig. 3.5, real-time video analytics consists of two kernels running in parallel alongside each other, namely, video encoding/processing and image classification and detection [133, 134]. Video encoding is used to encode the video stream and transmit it to the end user or to store it in the cloud, e.g. in a security video surveillance system the video is being streamed

Table 3.4 – Video Analytics Application Scenarios.

Case	Video Resolution (pixels)	Image Classification (FPS)
Surveillance: Construction Site/Pedestrians [41], [42]	640x480	5
Drone Navigation [43], [131]	300x200	10
Autonomous Driving and ADAS [44], [132]	640x480	15

to a control room or stored in the cloud [41]; in case of a drone navigation system it is being transmitted to the drone pilot [131]. In addition to video encoding, the video frames from the video stream are being analyzed simultaneously by the image classification application. CNN-based image classification may be used for facial recognition, counting people, potential hazard on a construction site in case of surveillance application [42]. It is also used for obstacle detection in drone navigation [43, 131] and autonomous cars [44, 132]. In a drone rescue mission, it can be used by drones to identify people at sea, in flood or in the rubble during an earthquake. In essence, both the video encoding and image classification needs to run side-by-side for a fully functional video analytics. The video encoding needs to process 24 FPS for a seamless user experience. Simultaneously, the image classification needs to classify a number of FPS depending on the application scenario. Table 3.4 summarizes the various application scenarios of video analytics application with different resolutions for video encoding and image classification/detection rate.

In the following sections, the video encoding and image classification kernels will be presented, both of which together make up the video analytics.

### 3.4.1.2 Video Encoding

Video encoding is an essential part of video analytics. Kvazaar [1], a state-of-the-art open source HEVC application, compliant with H.265 coding standard, is used as real time video encoding kernel. As discussed in Section 3.3, FIR filter and the SATD block are the two dominating blocks in Kvazaar that represent 21% and 26% of overall instructions executed, respectively. The remaining 53% of the computation is spread in chunks of less than 10% (in average 7%) instructions in other blocks. Hence, I will focus on defining a suitable architecture for the various case study scenarios with gem5-X to optimize the execution of the FIR filter and SATD block in Kvazaar.

### 3.4.1.3 Image classification using CNNs

The video analytics includes video encoding, which is either send to the end user or stored in the cloud, and CNN based visual recognition as discussed in [133, 134, 135]. CNN based image recognition is now becoming a standard in both the industry and academia for computer vision tasks because of the impressive results it has achieved [136]. All the case study scenarios

### 3.4. Heterogeneous Architecture for Video Analytics

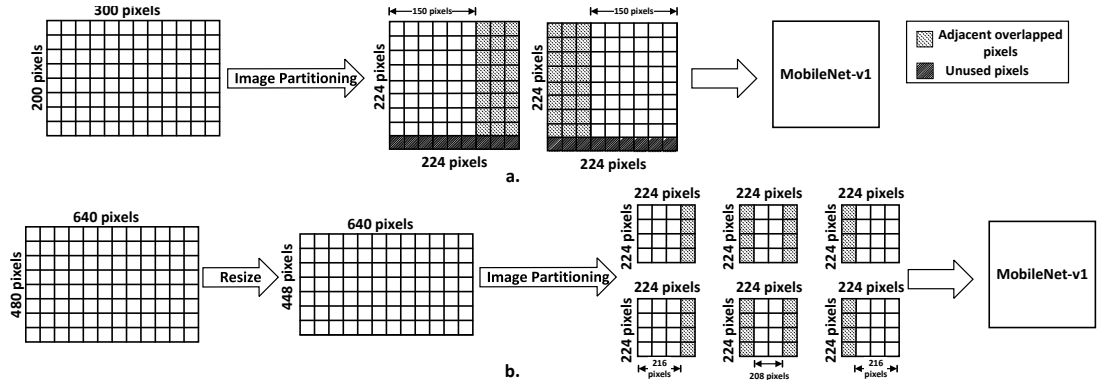


Figure 3.6 – Image partitioning and resize for input image resolution of (a) 300x200 pixels, (b) 640x480 pixels.

of surveillance, drone navigation and ADAS use CNNs for the visual recognition part of the application, as discussed in [42], [131] and [44], respectively. Because of ample research in CNN architectures, there are a variety of CNNs to choose from [137]. The choice to deploy a particular CNN depends on various factors such as accuracy, inference time, memory requirements, computation complexity and the size of the network. The video analytics scenarios presented in Table 3.4 vary from being deployed on the edge device like in drone navigation all the way up to high-end cloud servers like in a surveillance system. Therefore, the CNN should meet the memory, computational complexity and performance requirements in all the scenarios.

MobileNet [45] is an efficient CNN architecture that can be deployed on the edge as well as in the cloud. It has a Top-1 accuracy of 70.4% on the Imagenet benchmark, which is better than other complex popular CNNs, like GoogleNet, Alexnet and Squeezenet. MobileNet-v1 is also quite small in size and complexity, and feasible to be deployed on the edge node. MobileNet has previously been used in surveillance [42], drone navigation [138, 139] and autonomous cars and ADAS [140]. As most of the edge nodes are based on ARM architectures, the ACL [130] framework is used to deploy MobileNet. The limitation of MobileNet is that it can only process image sizes of up to 224x224 pixels, smaller than the case study scenarios listed in Table 3.4. Hence, for these images to be processed by MobileNet, image partitioning and resizing (if required) is performed, as shown in Fig. 3.6. For the 300x200 pixel frame, it is partitioned into two 224x224 images. Since the resolution of the resulting two 224x224 images is greater than the original image, extra pixels are used to overlap the split images horizontally. This overlap allows to detect objects that would have otherwise been split between images. The unused pixels at the bottom of the images are ignored and filled with black, as shown in Fig. 3.6a. For the frame resolution of 640x480, it is first resized to 640x448 pixels, so that it can evenly be distributed vertically to a 224 pixel size. After that, it is partitioned into six 224x224 pixel images, with the extra pixels being used to overlap the adjacent images to avoid splitting of objects between images, as shown in Fig. 3.6b. The partitioned images are then sent to MobileNet-v1 for image recognition.

MobileNet has been used as it is a small, fast and high accuracy CNN, which makes it feasible to be used both on the energy constrained edge device, as well as on the servers in the cloud. It has also previously been used in the case study scenarios listed in Table 3.4. However, any other CNN can be used instead of MobileNet, and the methodology used for optimization will remain the same.

### 3.4.2 Gem5-X Extensions

For the real-time video analytics application, comprising of video encoding and CNN based image classification kernels, the following compute and memory sub-system architectural extensions of gem5-X are utilized:

1. **BLADE In-cache Computing Accelerator:** The BLADE in-cache computing accelerator is used for accelerating the FIR filter and SATD blocks of the video encoding kernel, as discussed already in Section 3.3.4. As the in-cache computing engine has data alignment constraints, the Kvazaar kernel is modified to ensure data alignment during operations to be performed in-cache.
2. **ARMv8 ISA Extensions:** Gem5-X ISA extension capability is utilized by extending the ARMv8 ISA [39], using reserved op-codes. ISA extension enables the BLADE in-cache computing to be used in FS mode in gem5-X, as discussed in Section 3.3.4.
3. **Heterogeneous Compute Cores:** As the video analytics application comprises two different kernels, the video encoding and image classification based on CNNs, their compute requirements are also different. Hence, one kernel of the application might have better performance and energy efficiency on in-order cores and the other might run better on OoO cores. The gem5-X heterogeneous compute core extension, as discussed in Section 2.3.1.6 of Chapter 2, along with in-cache computing accelerator support will be utilized in such a scenario, which will be discussed later in Section 3.4.6.
4. **Core Clustering:** Core clustering in gem5-X enables independent L2 or LLC caches for different kernels of the application, as discussed in Section 2.3.1.5. Hence, clustering of cores alleviates the memory BW bottleneck on the L2/LLC, as each cluster has its own L2/LLC. As the video encoding application consists of two main kernels, core clustering will improve the performance, as the working data-set for each kernel will reside independently in its own cluster's cache. Thus, this also helps in avoiding the cache thrashing.
5. **HBM2 3D Stacked Memory:** The video analytics application is primarily compute dominated, with compute intensive tasks in both of its kernels, video encoding and CNN based image classification. However, when both the kernels are co-allocated together for a complete video analytics, the memory utilization increases. To alleviate any memory bottlenecks and increase the memory BW, 3D stacked HBM2 memory model in gem5-X

is used to improve both performance and energy efficiency, as it provides a high BW of up to 307.2 GB/s [79], enabled by multiple channels. The HBM2 memory model in gem5-X has already been discussed in Section 2.3.2.1 of Chapter 2. A detailed BW analysis of HBM2 in comparison to DDR4 has also been already presented in Section 2.6.

#### 3.4.3 Exploration and Optimization Methodology for Real-Time Video Analytics

To have an optimal architecture in terms of performance and energy efficiency for the case study video analytics, the two-step exploration methodology is used, as discussed in Section 2.8.2 of Chapter 2. Therefore, as in Step 1 of the methodology, separate architectural exploration and optimization is performed for each of the two kernels of the application, Kvazaar and MobileNet. Once the architectures have been optimized separately, both in terms of performance and energy efficiency, for both kernels, they are co-allocated together on a single platform as a video-analytics application. During co-allocation, the architectural optimizations for both kernels are combined together into a single architectural platform. Finally, in accordance with Step 2 of the methodology, global optimization is performed with both kernels co-allocated on a single architecture.

I will now discuss the architecture exploration and optimization separately for video encoding Kvazaar kernels and CNN-based image classification using MobileNet kernel. Finally, I will demonstrate the co-allocation of these two kernels and combine their respective architectural optimizations on a single platform for complete video analytics application.

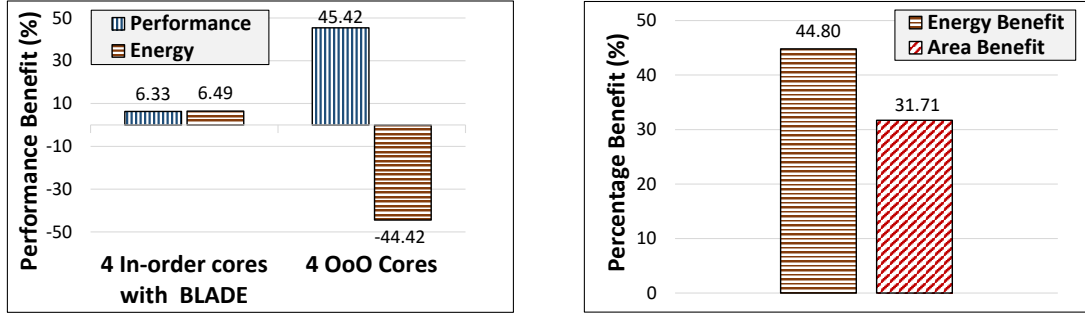
#### 3.4.4 Architecture Exploration and Optimization of Video Encoding Kernel

Kvazaar [1] is an open-source HEVC transcoding application, capable of performing both online encoding and decoding. For video analytics, the key element to consider is the video encoding part of Kvazaar. The performance requirement for encoding is 24 FPS to have a seamless user experience. Hence, for all video resolutions in Table 3.4, 24 FPS requirement needs to be met. As discussed in Section 3.4.1.2 and in [97], FIR filter and SATD are the two dominating blocks in the Kvazaar encoder. Hence, I will focus on optimizing the architecture for these significantly contributing blocks to have an architecture optimized for the overall kernel.

##### 3.4.4.1 Experimental Setup

The same experimental setup as for the video encoding application, presented in Section 3.3.2 and presented in Table 3.1 based on ARM JUNO platform [34] is used as a starting point. The power modeling modeling includes the core, LLC and memory power, as discussed in Sections 2.7 and 3.3.2.

The experimental setup is based on general-purpose compute cores (ARMv8 cores) and not



(a) Performance and energy benefit in comparison to 4 in-order cores.

(b) Energy and area benefit of 4 in-order cores with BLADE compared to 2 OoO cores.

Figure 3.7 – Performance, energy and area comparison for Kvazaar video encoding of 300x200 resolution video.

ASICs, as quite a lot of effort and time is required for the development of an ASIC. A software-based solution can ease this with just downloading and optimizing the code and running it on available general-purpose compute system. Furthermore, applications developed to run on ASIC cannot be updated/upgraded if there are new improvements to the algorithm. However, a CPU-based system allows for updating the currently deployed application with updated or new algorithms on the same platform.

### 3.4.4.2 Profiling the Video Encoding Kernel

Kvazaar was profiled both on ARM JUNO and in the gem5-X simulated system, as discussed already during the architecture exploration for the video encoding application in Section 3.3.3. FIR and SATD blocks were found to be the dominating blocks with 21% and 26% instructions respectively, and hence they were potential candidates for optimization using the BLADE in-cache computing engine, as discussed already in Section 3.3.5.2.

### 3.4.4.3 Architecture Exploration - Results and Analysis

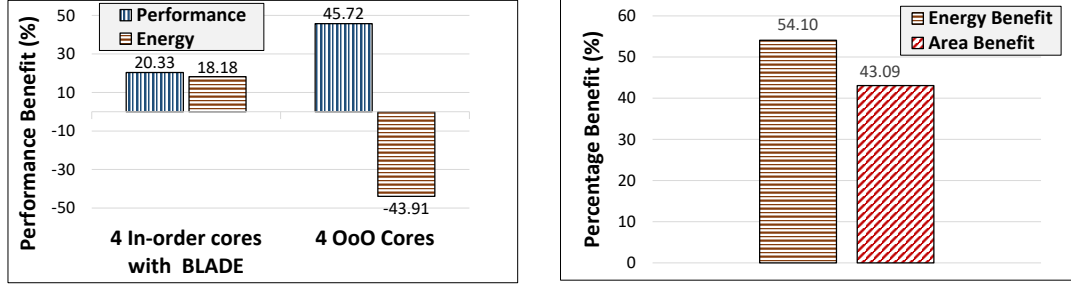
There are two video resolutions, 300x200 pixels and 640x480 pixels that cover all three-video analytic case study scenarios, namely, surveillance, drone navigation and autonomous cars, as presented in Table 3.4. Therefore, for the architectural exploration and optimization of Kvazaar, these two resolutions will be used. The baseline architecture will be as in Table 3.1, with ARMv8 64-bit in-order cores with the core count dependent on video resolution to meet the 24 FPS requirement. BLADE will be used as an accelerator along with ARMv8 in-order cores. The performance will also be compared to ARMv8 OoO cores.

- *300x200 Resolution:*

Figure 3.7a shows the performance and energy benefits of ARM 4-core in-order system



### 3.4. Heterogeneous Architecture for Video Analytics



(a) Performance and energy benefit in comparison to 4 in-order cores. (b) Energy and area benefit of 10 in-order cores with BLADE compared to 6 OoO cores.

Figure 3.8 – Performance, energy and area comparison for video encoding of 640x480 resolution video.

with BLADE and ARM 4 OoO cores in comparison to the baseline of ARM 4 in-order cores, all operating at 2GHz and DDR4 as main memory, while processing 300x200 resolution video at 24 FPS or more. It is observed that 4 in-order cores with BLADE are 6.33% performance efficient and 6.49% energy efficient, as compared to in-order system without BLADE. OoO cores on the other hand improve performance by 45% but at the cost of consuming 44.44% more energy than 4 in-order cores. Hence, the OoO core count is scaled down to 2-cores, to just meet the 24 FPS and not more.

Also, 4 in-order cores are selected with BLADE as the reference, and their energy and area compared to 2-OoO cores. Now both systems have the same performance to encode at 24 FPS, but 2 OoO cores consume 45% more energy along with an area overhead of 31% as depicted in Fig. 3.7b. Therefore, a 4-core in-order system with BLADE is the optimal architecture for 300x200 resolution video.

- 640x480 Resolution:** For 640x480 resolution video I start with a core count of 4-cores for ARM in-order, in-order with BLADE and OoO cores at 2GHz. As shown in Fig. 3.8a, 4 in-order cores with BLADE perform 20% better than in-order cores without BLADE in terms of FPS, along with 18% less energy consumption. The performance and energy benefit for in-order cores with BLADE is more in comparison to 300x200 resolution video, as a larger data set exploits more cache capacity and provides more opportunities for using BLADE vector processing in the cache. The OoO cores improve performance by 45%, but consume 44% more energy in comparison to in-order without BLADE, similar to 300x200 resolution video. However, with 4-cores none of these architectures meet the 24 FPS requirement. Hence, the core count is increased until the 24 FPS requirement is met. For in-order cores without BLADE, the number of cores are increased to 12, to reach 24 FPS, whereas, with BLADE only 10 cores are necessary, indicating a performance benefit of 20%. The number of OoO cores at 2GHz required to encode 24 FPS are 6. While processing 24 FPS for a 640x480 resolution video, I compare the energy and area required for OoO cores in comparison to in-order cores with BLADE. It is found that 10 in-order cores with BLADE are 54% more energy efficient compared to 6 OoO cores, and at the

### Chapter 3. Compute-Dominated Architecture Exploration

Table 3.5 – Time, core power (at 2GHz) and memory power for video encoding of 300x200 resolution video, while meeting 24 FPS requirement.

Parameter	300x200 Video Resolution		
	In-order 4-Cores	OoO 2-Cores	In-order 4-Cores with BLADE
Time (s)	0.98	1.0942	0.918
Core Power (W)	0.971	1.460	0.8567
Memory Power (W)	0.0291	0.0259	0.141

Table 3.6 – Time, core power (at 2GHz) and memory power for video encoding of 640x480 resolution video, while meeting 24 FPS requirement.

Parameter	640x480 Video Resolution		
	In-order 12-Cores	OoO 6-Cores	In-order 10-Cores with BLADE
Time (s)	1.006	1.092	0.962
Core Power (W)	3.835	6.448	2.987
Memory Power (W)	0.175	0.137	0.444

same time taking 43% less area as well, as shown in Fig. 3.8b.

The BLADE in-cache computing engine reduces the energy consumption, as it operates on larger data set using the SIMD approach as discussed in [37, 97]. Furthermore, it reduces the data movement between cache and the functional units in the CPU, as the operations are performed in cache, which reduces the energy consumption. For the video encoding kernel, Table 3.5 and Table 3.6 show the core and memory power for 300x200 and 640x480 resolution videos, respectively, while meeting the 24 FPS requirement. Here it can be seen that the energy reductions mainly come from the core power. In-order cores with BLADE in-cache computing has the least power, as compared to OoO cores, as well as the in-order cores without BLADE. In summary, this situation leads to low power ARM in-order cores with BLADE to match the performance of high-performance ARM OoO cores with 54% less energy budget.

Therefore, for the video encoding portion of video analytics, ARM64 bit in-order cores along with BLADE in-cache computing engine will be used, as it produces the best energy efficiency and area occupancy, while achieving the required 24 FPS.

#### 3.4.5 Architecture Exploration and Optimization of MobileNet Kernel

MobileNet [45] is a state-of-the-art CNN used for image classification, with low computational cost designed for deployment on mobile and edge devices. As image classification and detection is necessary for video analytics, MobileNet will be used for this purpose. The ARM ACL [130] framework is used to deploy an ARM optimized version of MobileNet, which is used in gem5-X

### 3.4. Heterogeneous Architecture for Video Analytics

Table 3.7 – MobileNet classification rate for various application scenarios.

Case	FPS - Original Resolution	FPS - Split Images
Surveillance (640x480)	5	30
Drone Navigation (300x200)	10	20
Autonomous Driving and ADAS (640x480)	15	90

based experiments to look into further architectural exploration and optimization choices.

MobileNet allows input image size of 224x224 pixels, whereas the three case study scenarios in Table 3.4 have 300x200 and 640x480 pixel resolution. As discussed in Section 3.4.1.3, each 300x200 and 640x480 frame is split into 2 and 6 images of 224x224 pixels as inputs to MobileNet, respectively. Therefore, the resulting FPS requirement for MobileNet is higher as shown in Table 3.7.

#### 3.4.5.1 Performance Comparison - CPU vs GPU

GPUs are increasingly being used both in training and inference of the CNNs architectures due to their high performance capability of processing vectored data, making them a perfect fit for CNNs [141, 142]. Then, as the case study scenarios involve edge devices, I looked into energy efficient GPUs for mobile and edge computing.

Nvidia Jetson Nano is a low power state-of-the-art GPU designed for Artificial Intelligence (AI) tasks on embedded and edge devices [143]. I compare the performance and energy consumption for MobileNet inference between ARM in-order cores, OoO cores and Jetson Nano. The Nvidia Jetson Nano platform [143, 144] comes with 128 Nvidia Maxwell GPU cores integrated with 4 ARM Cortex A-57 cores. Furthermore, the Nvidia Jetson Nano has better software support, as compared to other platforms like ARM Mali GPU [145] and Ethos Neural Processing Unit (NPU) [146]. It can run the networks implemented in Tensorflow directly and has the optimized versions of the networks in TensorRT [147], an optimized framework to deploy CNNs on Nvidia GPUs.

MobileNet is deployed on Jetson Nano using the Nvidia TensorRT framework [147]. The energy statistics on Jetson Nano are collected using the Nvidia *tegrastats* utility. Jetson Nano is set to its high-performance mode (10W mode) with its 128 GPU cores operating at 921MHz. The CPU and GPU in the Jetson Nano are in separate power domains [144], and in high-performance mode, the power rails VDD\_CPU and VDD\_GPU for CPU and GPU, respectively, are both set to 1.322V. These power rails only account for the CPU and GPU compute cores of Jetson Nano and not any other peripherals or I/Os, as they are in separate power domains. Both ARM in-order and OoO cores are configured in gem5-X to operate at 2GHz with 32KB L1 instruction and data cache. The LLC is configured to 1MB for 4 cores. As it will be discussed later in Section 3.4.5.2, MobileNet performance scales linearly with multiple threads on multi-core system up to 4 cores. Hence, for an 8-core simulation, two instances of MobileNet are launched, each using 4-cores

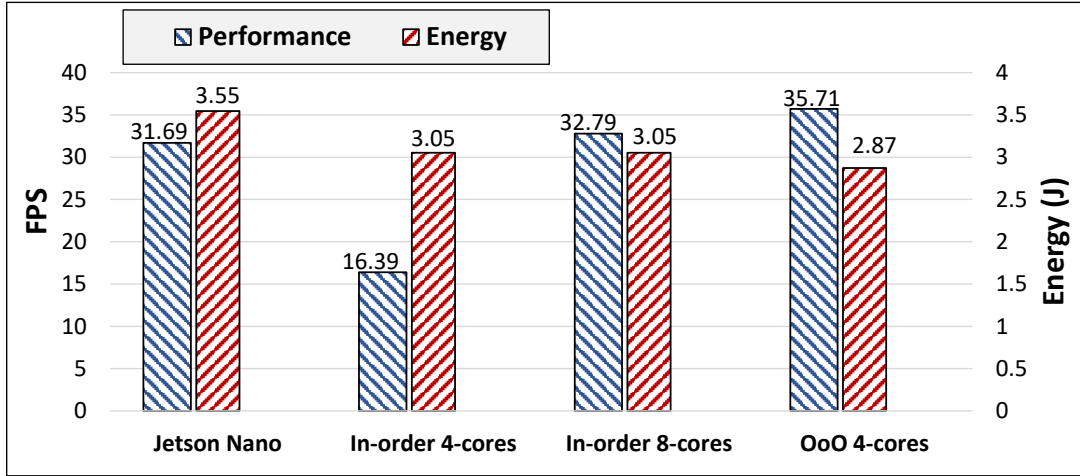


Figure 3.9 – MobileNet performance and energy comparison.

independently, so that the performance scales with core count. Thus, I use a clustering approach which will be discussed in detail in Section 3.4.5.2. Then, HBM2 is used as memory instead of DDR4, which is an architectural optimization, as discussed in detail in Section 2.3.2.1 and Section 2.6 of Chapter 2.

Figure 3.9 shows the performance in terms of FPS achieved by MobileNet on different architectures for 224x224 pixel input image. This figure shows that ARM 8 in-order and 4 OoO cores surpass the performance of Jetson Nano, by 3.3% and 11%, respectively. Hence, 4 OoO cores achieve the highest performance. The reason as to why the lower number of ARM CPU-based system matches or outperforms Jetson Nano comes from the fact that ARM CPU-based system is operating at 2GHz, whereas, Jetson Nano GPU-based platform with 128 GPU cores is operating at 921MHz. Regarding energy comparison, the energy is computed for processing 21 MobileNet images. Jetson Nano consumes the highest energy as can be seen in Fig. 3.9, whereas OoO cores are the most energy efficient. OoO cores are 27% more energy efficient as compared to Jetson Nano and 6% in comparison to 8 in-order cores. The energy of 8 in-order cores is the same as 4-in-order cores, as 8 in-order cores achieve double the FPS (half the execution time) in comparison to 4 in-order cores. Overall, OoO cores with HBM2 are the best both in terms of performance and energy efficiency for processing MobileNet. Jetson Nano consumes more energy in comparison to ARM CPU-based system because the high number of GPU compute cores (128 cores) in Jetson contribute largely towards the high energy consumption [148]. Secondly, the energy for the Jetson Nano platform includes the GPU energy as well as the 4 ARM Cortex-A57 cores which are mostly idle and not used for the actual computation of the MobileNet CNN, but are used as the host cores for the GPU. As these cores are mostly idle, their static energy adds to the total energy of the Jetson Nano platform.

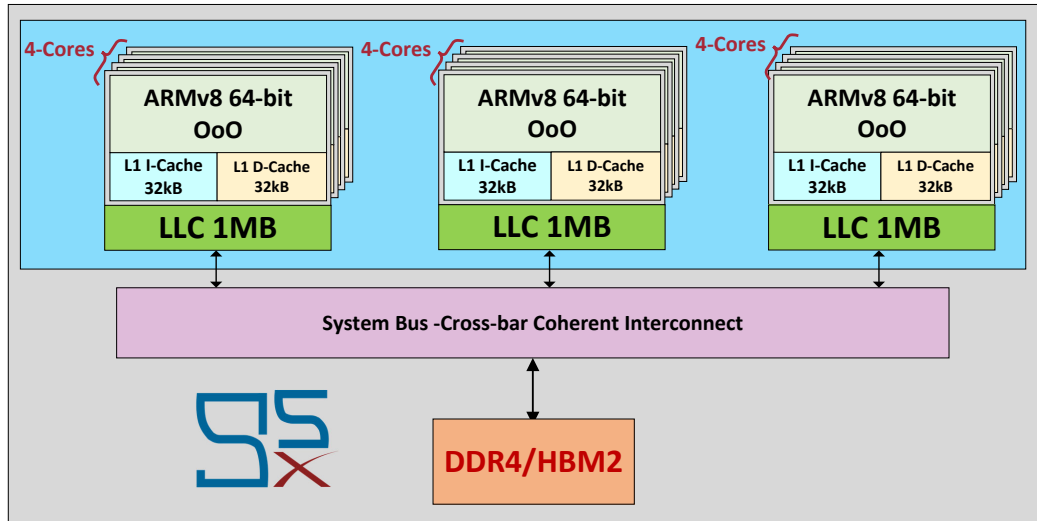


Figure 3.10 – Clustered architecture for multiple MobileNet instances.

#### 3.4.5.2 MobileNet Scaling and Clustering

To achieve higher FPS, e.g., 90 FPS for an ADAS system, more cores are required to process the images, as well as the use of multiple threads with one thread per core. Therefore, it is investigated how MobileNet scales with the number of cores.

It is found that FPS scales linearly up to 4 cores, however, after that the performance does not scale up with core counts and number of threads. This is due to the fact that since MobileNet is a light weight network, the cost of sharing data among higher number of cores affects the performance per core, thus suppressing the overall performance.

To overcome this issue of scaling, multiple independent MobileNet instances are launched in parallel, with each instance using a maximum of 4 cores. Hence, I instantiate 3 clusters each with 4 cores in gem5-X. Each cluster has its own LLC, as shown in Fig. 3.10, so that there is no thrashing in the cache.

#### 3.4.5.3 Architecture Exploration - Results and Analysis

For the compute side optimization of MobileNet, it was concluded in Sections 3.4.5.1 and 3.4.5.2 that ARM OoO cores are the best in terms of compute core performance, and that to use clustering for parallel processing and launching separate instance of MobileNet on each cluster.

Next, regarding the memory side exploration, 8-channel HBM2 is considered, as CNNs are memory intensive applications due to weight and bias accesses of the network. For the three-video analytics application case study scenarios, the FPS requirement needs to be met for the visual recognition portion, as in Table 3.7. Hence, I run experiments with ARM OoO cores at 2GHz, with the architecture shown in Fig. 3.10, both with DDR4 and HBM2 memory types.

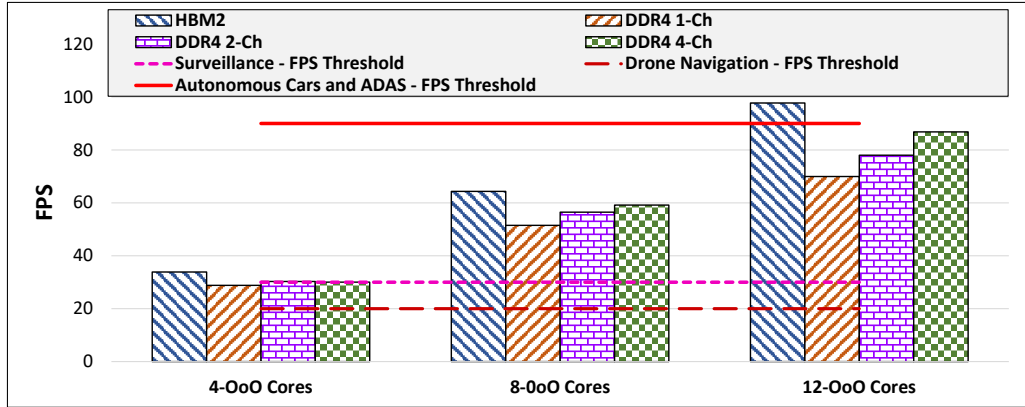


Figure 3.11 – MobileNet FPS scaling with core count (OoO) for HBM2 (8-Ch) and DDR4 with different number of channels. The horizontal lines are the threshold that should be met for scenarios in Table 3.7.

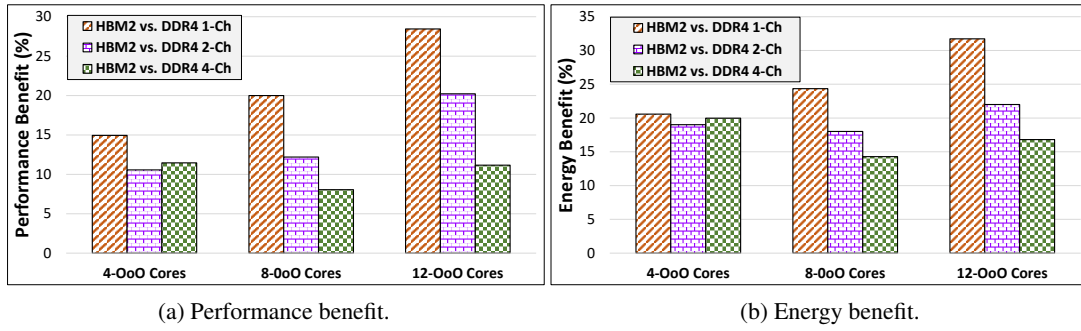


Figure 3.12 – MobileNet performance and energy benefit of using HBM2 (8-Ch) over DDR4 with different number of channels for OoO cores.

Figure 3.11 shows that 4 OoO cores with HBM2 meets the FPS requirement for drone navigation and surveillance scenarios, whereas, 4 OoO cores with DDR4 (1 to 4 channels) only meet the FPS requirement for drone navigation. However, using HBM2 instead of DDR4 results in energy benefit of up to 21%, as shown in Fig. 3.12b. For ADAS and autonomous smart cars, where the requirement is 90 FPS, 3 instances of MobileNet are launched, each on a 4-core cluster. Figure 3.11 shows that 12 OoO cores with HBM2 meet the 90 FPS requirement. Again 12 OoO cores with DDR4 (1 to 4 channels) fail to reach the required FPS. I also run experiments with 8 OoO cores with two MobileNet instances, and see that it scales well with the clustering approach, from 4-core to 8-core to 12-core systems.

Finally, I compare the percentage performance and energy benefit of using 8-channel HBM2 instead of DDR4, with number of channels varying from 1 to 4, for MobileNet. Figure 3.12a shows that the performance benefit varies from 8% to 28%, as it scales with the number of cores as well as with the number of DDR4 channels. Similarly, as shown in Fig. 3.12b, there are energy savings of up to 32% for 12 core system, when using HBM2 instead of single channel

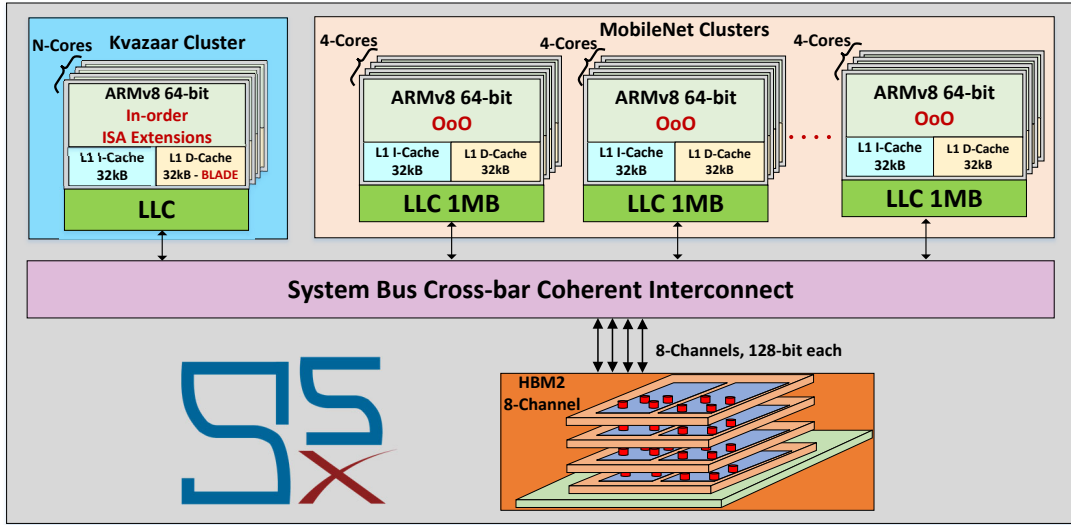


Figure 3.13 – Heterogeneous architecture for video analytics.

DDR4. The energy savings when using HBM2 instead of 4-channel DDR4 is between 16% to 20%, depending on the number of compute cores.

Therefore, ARM OoO core system with 8-channel HBM2 is the best performing system both in terms of performance and energy efficiency for MobileNet based visual recognition tasks.

#### 3.4.6 Architecture Exploration and Optimization of Video Analytics

Once the Kvazaar video encoding and MobileNet visual recognition kernels are independently optimized for performance and energy efficiency, I move to Step 2 of the optimization and exploration methodology, presented in Section 2.8.2 of Chapter 2. Then, the global architectural optimization and exploration is performed with all the application kernels co-allocated on the same platform, running in parallel. This complete analysis results in heterogeneous architectures with different clusters, as each locally optimized architecture might be different from one kernel to another, which share the crossbar interconnect and memory.

##### 3.4.6.1 Heterogeneous Architecture

Figure 3.13 shows the heterogeneous architecture proposed for the complete video analytics application. There is a separate compute cluster for Kvazaar, with ARM in-order cores and ISA extensions to use the in-cache computing engine, BLADE, integrated into the L1-D. Similarly, there are multiple clusters for MobileNet with 4 ARM OoO cores per cluster. The number of clusters depends on the required FPS. All of the clusters have their own LLC. HBM2 is used as the main memory in the system, as it offers more BW compared to DDR4 and it is necessary for MobileNet to achieve the required FPS. All the cores and clusters are 100% utilized for both

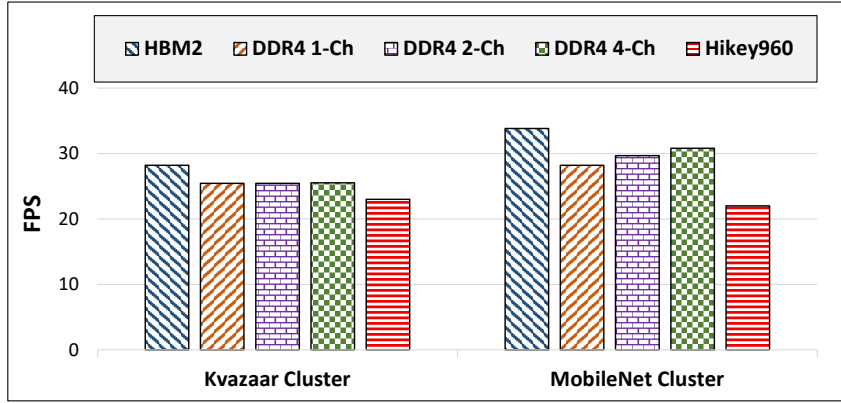


Figure 3.14 – Video Analytics FPS for Kvazaar and Mobilenet clusters for 300x200 resolution video for drone navigation when using HBM2, DDR4 and on Hikey960 platform.

the Kvazaar and MobileNet kernels. Hence, there will be no resource sharing between different kernels in terms of compute cores and their components, as it will not benefit the application performance. I will look into the performance and energy benefits of using HBM2 for Kvazaar as well as the overall system in the next section.

### 3.4.6.2 Architecture Exploration - Results and Analysis

For video analytics, the primary goal is to meet the QoS and performance requirements as set out in Tables 3.4 and 3.7 and improve the energy efficiency as much as possible. As there are three scenarios, one with a resolution of 300x200 pixels (drone navigation) and two with 640x480 pixels (surveillance and autonomous cars), I will explore and discuss the results according to the video resolutions.

- **300x200 resolution:** It was concluded in Section 3.4.4.3, that 4 in-order cores at 2GHz with BLADE in-cache computing is the optimal architecture to meet 24 FPS requirement for the Kvazaar video encoding part of video analytics. For visual recognition using MobileNet, 4 OoO cores at 2GHz were optimal in terms of performance and energy to meet the FPS requirement of 20 FPS for drone navigation, with HBM2 as the main memory. Hence, the architecture in Fig. 3.13 will have one Kvazaar cluster with 4-cores and LLC of 1-MB and one Mobilenet cluster of 4 OoO cores.

Figure 3.15a shows a performance benefit of 10% and 16.6% for Kvazaar and MobileNet clusters, respectively, when HBM2 is utilized instead of single-channel DDR4. The FPS values for both clusters with HBM2 and DDR4 are shown in Fig. 3.14. The performance benefit is approximately 9% for both clusters when using HBM2 instead of 4-channel DDR4. It is also observed that an energy benefit of 6% and 20.7% for Kvazaar and MobileNet is achieved, respectively, when HBM2 is compared to single-channel DDR4. When compared to 4-channel DDR4, the energy benefit is 2.5% and 18% for Kvazaar



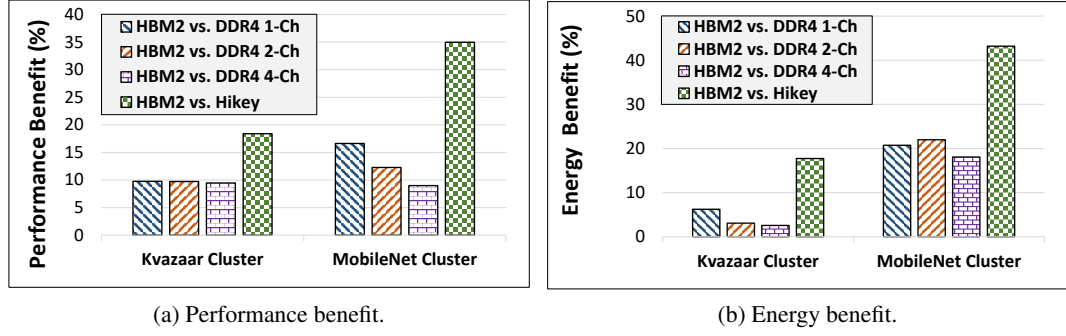


Figure 3.15 – Percentage performance and energy benefit when using HBM2 (8-Ch) in comparison to DDR4 with different number of channels and Hikey960 platform for video analytics of 300x200 resolution video for drone navigation.

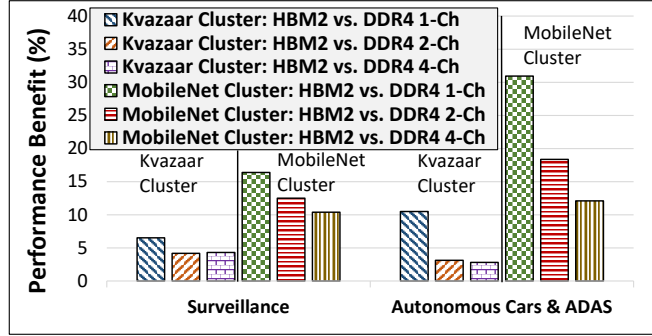
and MobileNet, respectively. As shown in Fig. 3.14, the achieved FPS are more than the minimum required for both kernels (24 FPS for Kvazaar, and 20 FPS for MobileNet). Moreover, the proposed architecture with HBM2 is also compared to commercially available ARM big.LITTLE Hikey960 platform [78], which has 4 in-order and 4 OoO ARM cores. As shown in Fig. 3.14, the Hikey960 achieves the least FPS, leading to 18% and 35% performance benefit of the proposed architecture using HBM2 over Hikey960, for Kvazaar cluster and MobileNet cluster, respectively, as shown in Fig. 3.15a. With regards to energy, energy savings of 18% and 43% are achieved over Hikey960, for Kvazaar and MobileNet clusters, respectively, as shown in Fig. 3.15b.

To further optimize the architecture, I move to Step 2 of the optimization methodology, to optimize for energy efficiency as the performance requirements have already been met. Both kernels scale linearly with frequency. Hence, the frequencies of both clusters are reduced to meet minimum performance requirements, therefore, enabling more energy savings. Reducing the frequency of Kvazaar 4-core cluster from 2GHz to 1.72GHz, results in an energy saving of 32% while achieving 24 FPS. Similarly, for the MobileNet cluster the frequency is reduced from 2GHz to 1.2GHz resulting in an energy saving of 59%.

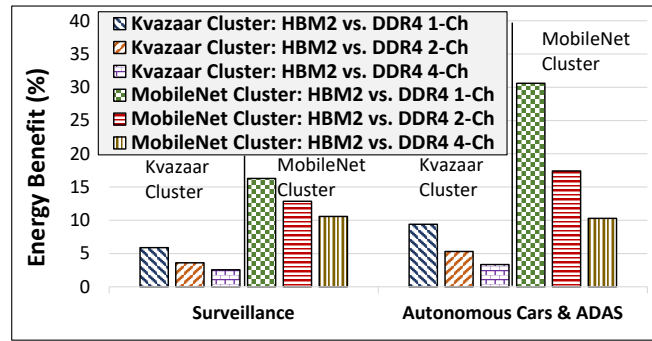
- **640x480 resolution:**

For the 640x480 resolution, two case study scenarios exist. First, one of video surveillance with MobileNet requirement of 30 FPS. Second, autonomous cars and ADAS with MobileNet are used for image recognition with a requirement of 90 FPS. The video encoding for both scenarios is required to execute at 24 FPS. Hence, as discussed in Section 3.4.4.3, a 10-core in-order cluster at 2GHz with BLADE is instantiated for Kvazaar, with HBM2 as main memory instead of DDR4. For MobileNet clusters, a 4-core OoO cluster at 2GHz meets the 30 FPS requirement and three 4-core clusters meet the 90 FPS requirement as discussed in Section 3.4.5.3.

The performance benefit of 10-core Kvazaar cluster varies from 6.5% to 10.5% when using HBM2 instead of single-channel DDR4, while meeting the 24 FPS requirement, as shown



(a) Percentage performance benefit.



(b) Percentage energy savings.

Figure 3.16 – Video Analytics for 640x480 resolution video for video surveillance and autonomous cars and ADAS systems. 10-core Kvazaar cluster is used for video encoding in both scenarios. A 4-core OoO cluster is used for MobileNet in case of surveillance, and three 4-core OoO clusters are used for autonomous cars and ADAS.

in Fig. 3.16a, and between 2.8% to 4.3% when comparing HBM2 to 4-channel DDR4. It can be seen that the percentage benefit of the Kvazaar cluster for 640x480 resolution video, when HBM2 is being used instead of DDR4, is less as compared to 300x200 video resolution, as in Fig. 3.15a. The reason being that, the higher the resolution, the more efficiently BLADE in-cache computing and caching is used as compared to lower resolution video. Hence, lower BW utilization of main memory and less performance benefit. The corresponding energy savings of using HBM2 for the Kvazaar cluster is between 6% to 10% and 2.6% to 3.3%, in comparison to single-channel and quad-channel DDR4, respectively, as depicted in Fig. 3.16b. For the MobileNet clusters, the performance and energy efficiency increase with the scaling of 4-core clusters from 1 to 3 for surveillance and ADAS, respectively. The performance improvement of using HBM2 is up to 30% with a corresponding energy benefit of 30% in comparison to single-channel DDR4. When compared to 4-channel DDR4, the performance and energy benefits of using HBM2 are up to 12% and 10%, respectively, for the MobileNet clusters. Comparing the results of MobileNet cluster in Fig. 3.16 to that of Fig. 3.15, it can be observed that the performance

### 3.4. Heterogeneous Architecture for Video Analytics

Table 3.8 – Time, core cluster power (at 2GHz) and memory power for video analytics case study scenarios.

Parameter	Clustered Architecture with HBM2			Clustered Architecture with DDR4		
	Drone Navigation	Surveillance	ADAS	Drone Navigation	Surveillance	ADAS
Time Kvazaar Frame (ms)	35.45	38.11	39.47	39.29	40.76	42.83
Core Power Kvazaar Cluster (W)	0.865	2.943	2.873	0.832	2.924	2.940
Time MobileNet Frame (ms)	29.55	29.41	10.465	35.44	35.17	15.015
Core Power MobileNet Cluster (W)	4.490	4.738	14.215	4.723	4.732	15.645
Memory Power (W)	0.198	0.511	1.222	0.631	1.672	3.302

and energy savings are higher for three 4-core MobileNet clusters when it is co-allocated with Kvazaar than when it is standalone. This is because, when more applications are being allocated on a single platform sharing the same memory, the memory BW requirements increase, which HBM2 serves better than DDR4.

The overall energy reduction when using HBM2 instead of DDR4, is due to the fact that HBM2 not only has lower energy per access in comparison to DDR4 [93, 94], but also enables faster memory access via 8 memory channels. The faster access in turn implies less stalling of the processor pipelines when waiting for data, resulting in overall energy reduction. Table 3.8 summarizes the core power, memory power, and execution time per frame for both kernels of video analytics in drone navigation, surveillance and ADAS scenarios. It can be seen that the run-time to process each frame is less when using HBM2 instead of DDR4. Secondly, the memory power when using HBM2 is less as compared to when using DDR4. Therefore, lower memory power and fast processing time for each frame (both for video encoding and MobileNet kernels), contributes to the energy benefits achieved when using HBM2 instead of DDR4.

In these case study scenarios for surveillance and autonomous cars, Step 2 of the methodology is not used, as the performance requirements are just met with the proposed architectures. Thus, I cannot further reduce the frequency to optimize for energy, as was the case in drone navigation, because this will degrade the performance below the minimum requirement.

In summary, capitalizing on the gem5-X simulation platform, an optimized heterogeneous

architecture is proposed for a video analytics application, composed of ARM in-order cores along with BLADE in-cache computing engine being used for the video encoding portion and ARM OoO core clusters for CNN based image recognition with HBM2 as the main memory. This heterogeneous architecture meets all the performance requirements and at the same time is energy efficient in all the three case study scenarios. In this case study, Step 2 of the methodology was used to optimize for energy consumption by reducing the frequency for drone navigation scenario. Other than this, as the two kernels of the application do not give rise to any conflicting requirements, no trade-offs have to be made. Moreover, the methodology is generic if an application requires further architectural exploration during Step 2. For instance, in case of different kernels of the application running on different cores, when allocated together on a single system, might all share the same LLC. In that case, the strategy to follow in Step 2 would be to optimize LLC size to fit the working data set of all kernels. In another instance, the memory requirement for individual kernels might be sufficient, but when co-allocated, the application might run out-of-memory. Thus, in Step 2 one will select appropriate memory size, forcing to even change the memory technology being used, as some memories like the 3D stacked HBM2 are limited in terms of the memory size, as compared to traditional DDR4 memories. Therefore, Step 2 of the methodology can be used to optimize the architecture for the compute and memory sub-systems in cases when conflicting requirements arise after co-allocating all the kernels of the application together, in addition to the tuning of core frequencies for optimal performance and energy. Furthermore, the two-step methodology is generic and not specific to any application or application domain. It can be used to explore and optimize architecture for any application domain. The advantage of using the methodology for a general-purpose CPU-based system, is that it gives an optimized architecture for the application domain and not specific to an application. This enables updating the application or replacing the application or one of its kernels with a better application or kernel, which is not possible in ASICs, as they are application specific. This has been demonstrated by using MobileNet for image classification kernel in this section, instead of AlexNet, which was used in the previous Section 3.3.7. From Fig. 3.15, it can be seen that using an 8-core system with 4-cores allocated to Kvazaar and 4-cores for MobilNet, the performance benefit of using HBM2 instead of DDR4 is up to 10% and 16.6% for Kvazaar and MobileNet, respectively, in comparison to 7.72% and 8.35% performance benefit for Kvazaar and AlexNet, respectively, in previous Section 3.3.7 in [97], in a similar 8-core system.

### 3.5 Near-Threshold-Computing (NTC) Servers for VMs in the Cloud

Cloud computing has recently been brought into focus in both academia and industry due to the increase of applications and services. Consequently, there has been a rapid growth in the number of data centers in the world, leading to increased energy consumption, estimated to be at 1% of the global energy usage [15].

To maximize energy efficiency (i.e., performance per watt), customized server architectures increase throughput by identifying and eliminating the bottlenecks of conventional server processors. However, as an effect of post-Dennard scaling [18], energy reduction in deep sub-micron

technologies has lagged behind, resulting in power-limited servers.

A promising approach to overcome the power bottlenecks is NTC. NTC takes advantage of the quadratic dependency between supply voltage and dynamic power consumption, by lowering the operating voltage to a value slightly higher than the transistor threshold, increasing energy efficiency at the expense of reduced performance. However, for current cloud applications, NTC allows to optimize the trade-off between performance and power, emerging as a promising approach to overcome the power-wall [25].

From a technology viewpoint, the UTBB FD-SOI technology has demonstrated its suitability for NTC. In contrast to traditional bulk technology, FD-SOI features a significantly increased voltage range and even higher performance for the same energy thanks to the better behavior of transistors at low voltage [110]. The 28nm FD-SOI technology process is currently employed for mass production by Samsung and ST Microelectronics, the 20nm technology is being produced by GlobalFoundries while the 12nm node is on the strategic roadmap [149]. With respect to FinFET technology, FD-SOI provides a cost-sensitive solution for low-power (both active and leakage) systems without increasing die cost [150], making it a suitable solution for next-generation NTC servers.

In the following sections, I utilize an accurate power model for the UTBB FD-SOI process technology in NTC servers, together with a performance validation against real servers (Intel x86 and ARM64), and propose a performance-improved architecture for NTC servers. I will also assess the performance and efficiency of virtualized workloads on three architectures: (i) x86, (ii) ARM-based Cavium ThunderX, and (iii) the proposed NTC server, which modifies and improves the efficiency of the ThunderX architecture. I will also investigate the Dynamic Voltage and Frequency Scaling (DVFS) setup and its effect on performance and power for NTC server.

#### 3.5.1 Overview of the Proposed system

##### 3.5.1.1 Application Description

The applications consist of VMs, virtualized via Linux LXC containers, and running synthetically generated workloads that resemble batches of real banking applications, as reported by the industry partners. These VMs perform batch financial analysis, mainly based on matrix multiplication and manipulation, and both their CPU and memory utilization can be tuned. For realistic CPU and memory usage, one week of traces of Google Cluster [151] are used, which provide the CPU and memory utilization for over 600 VMs, reported every 5 minutes (memory utilization is varying in the range of 2% to 32%). Therefore, for profiling purposes, the workload is divided in three categories, according to the per-VM memory utilization:

1. Low-mem for average memory usage of 70MB (7%)
2. Mid-mem for 255MB (25%)

### 3. High-mem for 435MB (43%)

Moreover, in order to run the experiments in worst-case scenarios, the workloads are tuned to maximum CPU utilization.

#### 3.5.1.2 Server and Data Center Architecture

As a starting point for the server architecture, Cavium ThunderX platform [111] is chosen. However, for the target applications, as discussed in Section 3.5.1.1, the Cavium performance was slower (from 1.5x to 3.5x) than the x86 platform with similar characteristics, and unable to meet QoS constraints, as discussed later (see Table 3.9 and Section 3.5.4.1). This was due to an inappropriate memory subsystem design for the target applications considered and the choice of in-order cores. Hence, the original architecture is modified with ARMv8 OoO cores instead of the in-order cores. A 16-core CPU (instead of 48 in ThunderX) is modelled to achieve a lower simulation turnaround time, as it was experimentally observed that the model linearly scaled up for higher number of cores. The memory sub-system was also updated, by including L1 instruction cache (I-cache) and data cache (D-cache) of 64KB and 32KB, respectively. A LLC of 16MB is modeled. A total memory size of 16GB is considered using a DDR4 memory model with memory controller [152]. DDR4 is clocked at 1200MHz with a peak BW of 19.2GB/s. Gem5-X FS mode is used to boot Ubuntu 16.04 and run VM workloads on top of it.

#### 3.5.1.3 QoS Degradation Constraint for VMs

Because banking applications are virtualized batch jobs, their QoS constraints are defined in terms of the maximum allowable degradation (i.e., increase in their execution time), which in this case is defined as 2x [108], w.r.t. a baseline execution in a 16-core Intel Xeon X5650 running at 2.6GHz, with 12MB LLC and 128GB of RAM clocked at 1333MHz, in which one LXC container (VM) per core is executed. 16 LXC containers are created for 16 cores on the server. Then, CGROUPS are used to allocate one container per core. The *time* command in Linux is used to get the total execution time for each workload with different memory utilization.

#### 3.5.2 Server and Data Center Power Models

The overall NTC server power model used is based on the work of Ali et al. in [25, 46]. The power model in [25, 46] is extracted by combining direct measurement on a commercial ARMv8 based server [153] with power measurements of real prototypes implemented in 28nm FD-SOI technology and operating in near-threshold [110, 154], allowing for a very accurate system power estimation. Four main contributors are considered to the overall power consumption of the server: 1) the core region composed of the A57 cores logic and the L1/L2 caches, 2) the LLC, 3) the memory controller, peripherals, IO subsystem and motherboard, and 4) the DRAM banks.

#### 3.5.2.1 Cores

For the core power, the power model proposed by Ali et al. [25, 46] is used, in which the authors combine the 28nm FD-SOI power and performance model of a recent Cortex A9 implementation of STM in 28nm bulk and FD-SOI, considering the differences in pipeline length ratio and critical path between Cortex A57 and Cortex A9. In the proposed model in [25, 46], these parameters are extracted by comparing the different voltage to frequency ratio (extracted via the CPUFreq Linux driver) present in the Samsung Exynos processor family. The Cortex A57 is 1.17x faster (higher-frequency) than the Cortex A9. This information is combined, by the authors of [25, 46], with the active and static energy per clock cycle at the different DVFS levels from the Samsung Exynos 5433 processors to scale its energy figures to the STM 28nm FD-SOI technology by using the trends reported in [110]. These numbers also account for the L1 and L2 cache power consumption. When in WFM state, the core region consumes 24% less power than when active. This number is reported to be measured empirically on an Intel Xeon v3 processor. Then, for the proposed NTC power model in [25, 46], the performance and power model is extended to the NTC region fitting a template extracted from measurements of a 28nm UTBB FD-SOI near-threshold parallel processor [154],

#### 3.5.2.2 Last-Level-Cache (LLC)

The LLC power model proposed by Ali et al. [25, 46] is utilized. The authors of [25, 46], modeled LLC by measuring the leakage power for a 256KB SRAM block in 28nm UTBB FD-SOI and read and write energy [pJ/Access] for 128-bit wide accesses. All these values are reported for different voltage levels.

#### 3.5.2.3 Memory Controller, Peripherals, IO and Motherboard

Ali et al. [25, 46] empirically measure the memory controller, peripherals and IO subsystem power consumption overhead of an Intel Xeon v3 CPU. This power consumption is split in two parts: (i) a constant component which accounts for the static and fix dynamic power cost needed to keep these subsystems on, and (ii) a component proportional to the operating condition. The constant part causes a 11.84W overhead in all operating points, while the proportional one ranges from 9 to 1.6W in the operational range. Finally, the same motherboard power consumption is assumed than in the Cavium ThunderX server, which is of 15W for a low fan speed, and with 1 SSD disk.

#### 3.5.2.4 DRAM

The DRAM power proposed by Ali et al. [46] is used. The authors of [46] obtained this power model with direct measurement on a real server platform based on Intel Xeon v3 architecture. The final model contains the empirical measurement of an idle power value of 15.5 mW/GB

Table 3.9 – NTC server and Cavium ThunderX QoS analysis in terms of execution time (s).

Application	Intel x86 @2.66 GHz	2x Degraded Intel (QoS limit)	Cavium @2GHz	NTC Server @2GHz
low-mem	0.437	0.873	0.733	0.582
mid-mem	1.564	3.127	5.035	2.926
high-mem	3.455	6.909	11.943	6.765

of DRAM, which increases to 155 mW/GB when the banks are activated. On top of this static power, an energy consumption of 800 pJ/Byte is reported.

### 3.5.3 Gem5-X Extensions

As the target application consists of VMs, virtualized via Linux LXC containers, and running synthetically generated workloads, as discussed in Section 3.5.1.1, the VM support enhancement and extension provided by gem5-X will be used.

VMs are supported in gem5-X using Linux LXC containers, as discussed in Section 2.4.5 of Chapter 2. To enable virtualization in gem5-X, configuration parameters related to virtualization and LXCs were enabled in the Linux kernel. LXCs were also added to the gem5-X disk image to have a complete VM support in gem5-X.

### 3.5.4 Experimental Results

#### 3.5.4.1 Simulation Framework Validation

Gem5-X cycle-accurate simulator is used to simulate the server architecture described in Section 3.5.1.2. In order to understand the effect of DVFS on performance, I compute the QoS degradation, taking as a baseline the execution time on the x86 server discussed in Section 3.5.1.3. Then, the virtualized applications are simulated in gem5-X for different frequency levels ranging from 2.5GHz down to 100MHz.

To validate the correctness of the results provided by the gem5-X simulator, the applications are run on two real hardware platforms based on x86 and ARM. I compared the execution times of Cavium ThunderX with the ones obtained via gem5-X while matching the exact same architectural configuration. The error obtained was below 10% in terms of execution time, showing that gem5-X is able to accurately simulate the workloads.

The execution time for each workload, on all three platforms are shown in Table 3.9. The QoS limit is a 2x degradation of the execution time on x86 based platform, as already discussed. The Cavium server exhibits the worst execution time. After the modifications undertaken, the proposed NTC server architecture outperforms Cavium by a factor of 1.25x to 1.76x. These results are due to the improved memory sub-system and the incorporation of the OoO processor



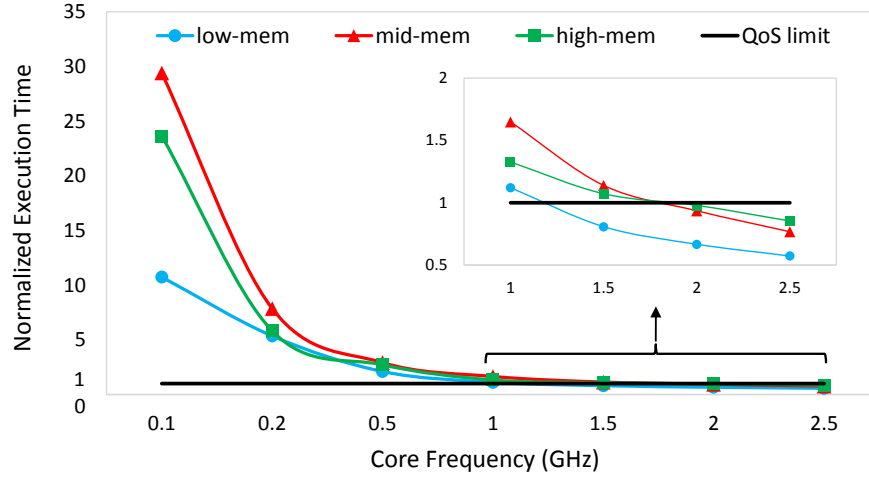


Figure 3.17 – Execution time normalized to QoS limit for different workloads.

in the proposed architecture.

#### 3.5.4.2 Server-level Results

QoS requirements of virtualized applications make it unclear whether this technology is suitable for server processors. To check for QoS requirements being met for VM workloads on NTC server, normalized execution time to QoS limit is shown in Fig. 3.17. It can be seen that high-mem and mid-mem workloads meet QoS requirement till a minimum frequency of 1.8GHz, whereas low-mem can scale down to 1.2GHz. In conclusion, I am able to reduce the frequency of the cores while meeting the 2x degradation constraint for virtualized applications.

#### 3.5.4.3 Energy Efficiency

Fig. 3.18 shows the benefits of reducing DVFS on server energy efficiency (i.e., the total number of User Instructions per Second (UIPS) at the chip level, divided by the total power consumption of the server). The optimal efficiency point is around 1.2GHz for high-mem, and around 1.5GHz for low-mem and mid-mem. The energy efficiency decreases with increasing memory utilization, firstly, because of higher active memory power, and secondly, because more memory accesses increase the amount of stalls and the WFM cycles.

#### 3.5.4.4 Trade-offs Discussion

As shown by [25], workloads can tolerate low frequencies if only core power is considered, thus enabling NTC operation to reduce core power consumption. However, not all server components scale with the core voltage, shifting the most energy-efficient point to a higher frequency. The results showed that frequency can be reduced to 1.2GHz for high-mem and 1.5GHz for low-mem

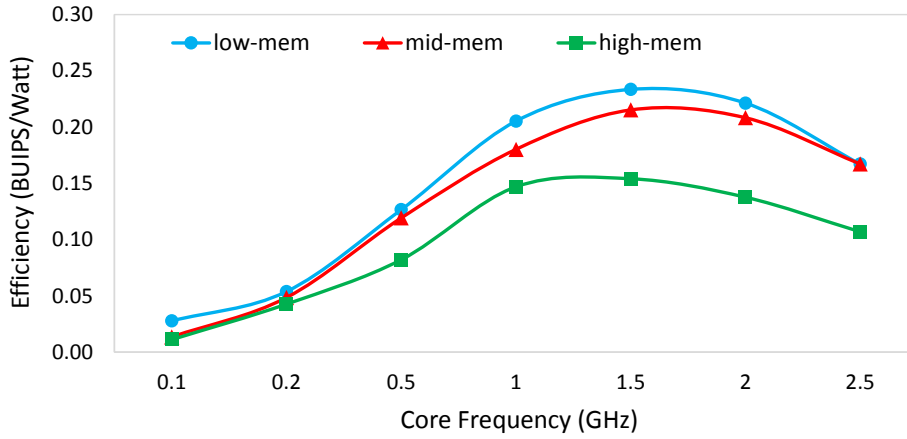


Figure 3.18 – Server efficiency as UIPS/Watt under different core frequencies.

and mid-mem. But, to guarantee the QoS requirements, the frequency level should be scaled up to 1.8GHz for mid-mem and high-mem; while for low-mem (CPU-bounded tasks) the optimal frequency (i.e., 1.5GHz) still meets the QoS limit.

### 3.6 ISA Extensions for In-Memory Computation of AI Workloads

The advent of AI and Machine Learning (ML) algorithms and systems, fueled by Big Data, are having an ever increasing impact in different spheres of our lives. These systems are being used and developed for autonomous cars, surveillance, medical imaging and diagnosis, drug discovery and online content recommendation to name a few. These AI and ML algorithms are deployed in different execution platforms all the way from high performance cloud servers to energy constrained edge devices. These algorithms are computationally intensive, along with extensive memory accesses. I propose to use in-memory computations or CM for these computational and memory intensive workloads, as it improves performance and energy by allowing multiple operations to be performed in parallel in the memory.

Gem5-X enables in-memory computation allowing the CM to be integrated in the execution stage of the CPU pipeline, as discussed in Section 2.3.1.4 of Chapter 2. CM extension in gem5-X can be utilized in FS simulation mode, thanks to the ISA extensions in gem5-X. Using the gem5-X ISA extensions, new instructions for the operations of CM are added to the instruction set.

I will present and use two CMs in this section: RRAMs and analog in-memory CM for BNNs and LSTM workloads, respectively.

#### 3.6.1 Accelerating BNNs with RRAMs

The role played by Internet-of-Things (IoT) in the advent of Big Data [155], which requires the execution of complex AI algorithms in latest smart embedded systems (also called edge

devices [156]), has created the necessity of developing new ML algorithms that can target compute and energy constrained systems. As BNNs have been proposed for these edge devices, I propose a BDPE based on RRAMs for improved performance and energy efficiency of the BNNs. The most frequently used kernels are stored in the RRAM, reducing the overall memory accesses. Since RRAMs excel by their ability of enabling in-memory computing capabilities while providing storage support [157], the novel BDPE uses a robust and energy efficient RRAM-based convolutional block based on [2] and operates in the digital domain. Furthermore, I propose to integrate the BDPE in the execute stage of the CPU pipeline, hence reducing the data transfer and communication overhead. Consequently, using the RRAM based BDPE for BNNs leads to performance and energy benefits.

#### 3.6.1.1 CNN Background

CNNs relate to a class of Neural Networks (NNs) that are commonly applied to image analysis. Such networks are dominated by convolutional layers, where the input is convoluted by a kernel, and the produced output is passed to the next layer. The operation of CNNs is divided into two phases: the training phase and the inference phase. Although the devised mechanism can be applied to both phases, the computing power and energy required by the training phase goes beyond edge devices. Therefore, only the inference phase is targeted by this work.

In the image analysis domain, CNNs can be used for multiple purposes, such as image classification [158, 159], or object detection [160], such as pedestrians detection [161, 162] or face recognition [163]. YoloV3 [49, 164] is a state-of-the-art CNN for real-time object detection. It divides the image into regions and predicts bounding boxes and associated probabilities for each region. By sizing the network, YoloV3 also allows trading accuracy for performance by reducing the number of network layers. This represents an advantage for edge devices characterized by low computing power. When binarized using the definitions in [47], the XNOR-Net version of a given configuration of YoloV3 shows approximately the same mean Average Precision (mAP) as the full-precision network, while reducing the size of the weights approximately 32× and thus executing up to 58× faster. For the Street View House Numbers (SVHN) dataset [165], the YoloV3 XNOR-Net shows a mAP decrease of only 0.43%, while increasing the number of detections by 75%. YoloV3 is a widely accepted benchmark, thus it is used to evaluate the proposed BDPE.

From the same authors of YoloV3, Darknet [166] is a C-based ML framework compatible with a large number of NN models. When executed in Darknet, BNNs can be accelerated using CPU primitives that can efficiently perform the binary convolution of two 64-bit vectors. For that purpose, the elements of the small kernels are shifted and combined in vectors of 64 bits, thus reducing the data redundancy that can be found between combinations. For the trimmed version of YoloV3 XNOR-Net trained with the SVHN dataset, all the generated combinations of small kernels are different, making it impossible to take advantage of data redundancy. However, as shown in Table 3.10, a small percentage of combinations is frequently used. For instance, 0.07%

### Chapter 3. Compute-Dominated Architecture Exploration

Table 3.10 – Profile of a trimmed version of YoloV3 regarding the frequency of use of each convolutional kernel during the inference phase.

Execution frequency	Kernels	Total executions	Percentage
169	115,200	19,468,800	45.64
676	18,432	12,460,032	29.21
2,704	1,152	3,115,008	7.30
10,816	320	3,461,120	8.11
43,264	96	4,153,344	9.74

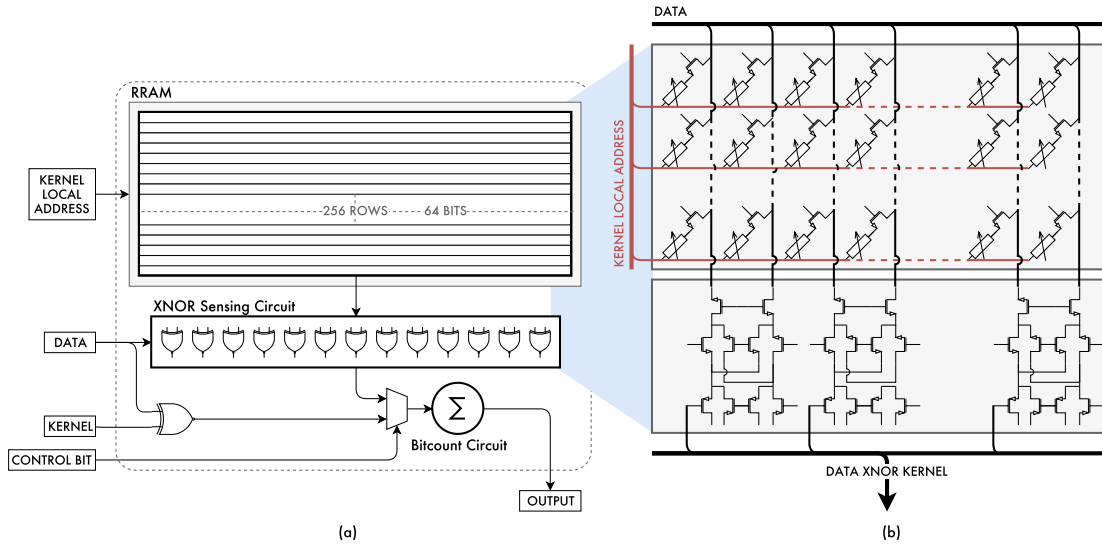


Figure 3.19 – Block diagram of the proposed BDPE. Fig. 3.19a illustrates the control logic and the alternative mechanism to perform the bit-wise XNOR of the data and the kernel (both supplied as an input), in case the kernel is not stored in the RRAM array. Fig. 3.19b depicts the RRAM-based convolutional block used in this work inspired by [2].

of the combinations (designated from now on as kernels) are used in 9.74% of the total number of convolutions. Thus it is possible to significantly accelerate the total amount of convolutions only storing a small percentage of kernels locally. Due to Darknet's performance, it is used for evaluation purposes.

#### 3.6.1.2 Architecture of the Binary Dot Product Engine

The devised BDPE aims at improving the performance of binary convolution while reducing the number of data transfers by locally storing the most used kernels. Accordingly, the base block used, proposed in [2], implements the binary dot product using the compute capabilities of RRAM, while also providing storage. The operands involved in the binary dot product are a constant binary kernel stored in the RRAM array in the form of resistance values and a binary vector supplied as an input. By selecting the appropriate kernel local address and applying the

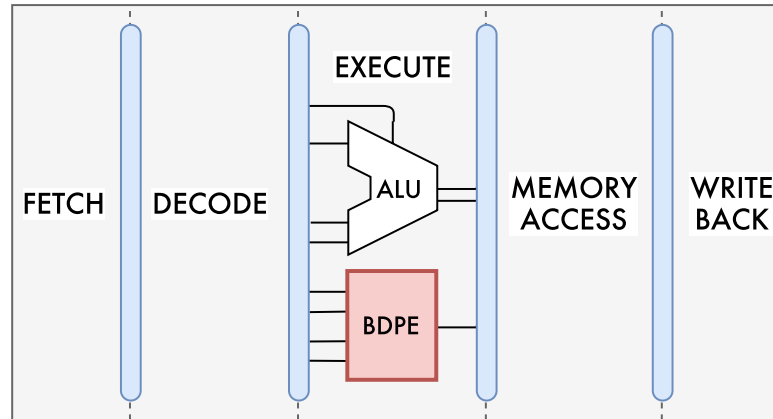


Figure 3.20 – Simplified block diagram of a generic processor pipeline integrating the BDPE.

input data to the RRAM array, the XNOR phase of the convolution is performed as a memory readout using custom XNOR sense amplifiers [2]. Then, a fully-digital combinational circuit counts the number of logic ones to determine the result of the binary dot product operation. The block diagram of the proposed RRAM-based convolutional block is depicted in Fig. 3.19b. Using the proposed mechanism has two main advantages. First, by locally storing the most used kernels, the data movements are substantially reduced, thus decreasing the overall energy consumption. Second, since it is capable of performing a dot product in one clock cycle, it also increases system's performance.

To use the devised engine in a CPU, a control path is added to the original block, which also unlocks the possibility of performing the XNOR of two inputs as an alternative to using the primary RRAM-based XNOR mechanism. Fig. 3.19a illustrates the block diagram of the BDPE.

The secondary XNOR mechanism is built from regular CMOS gates. Depending on the *control bit* obtained from the decoded *opcode*, the output of the alternative XNOR mechanism can be used as input to the *Bitcount Circuit*. In that case, the result is based on the kernel coming from a processor register rather a kernel stored in the RRAM. Such a functionality is particularly useful when the required convoluting kernel is not stored in the RRAM array.

#### 3.6.1.3 Integration within the CPU Pipeline

The integration of the proposed BDPE with a conventional ARMv8 core is divided into three phases, namely: (1) the integration of the new functional unit with the processor's pipeline; (2) the creation of new instructions in the ISA to use the BDPE; (3) compiler support to use the new ISA instructions on the software side.

The new functional unit is integrated into the processor's pipeline in the *Execution* stage. It receives the operands from the *Decode* stage, similarly to the Arithmetic Logic Unit (ALU), and passes the result to the *Execute/Memory Access* pipeline register, as shown in Fig. 3.20.

### Chapter 3. Compute-Dominated Architecture Exploration

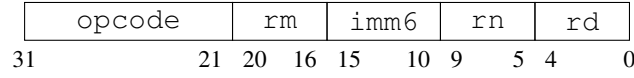


Figure 3.21 – Format of the new instructions added to the ARMv8 ISA to allow the processor to issue instructions to the BDPE. The *opcodes* 10000011000 and 11000011000 were re-purposed to specify the custom instructions. *rm* and *rn* specify addresses of 64-bit registers; *imm6* represents a 6-bit immediate; and *rd* specifies the address of the destination 64-bit register.

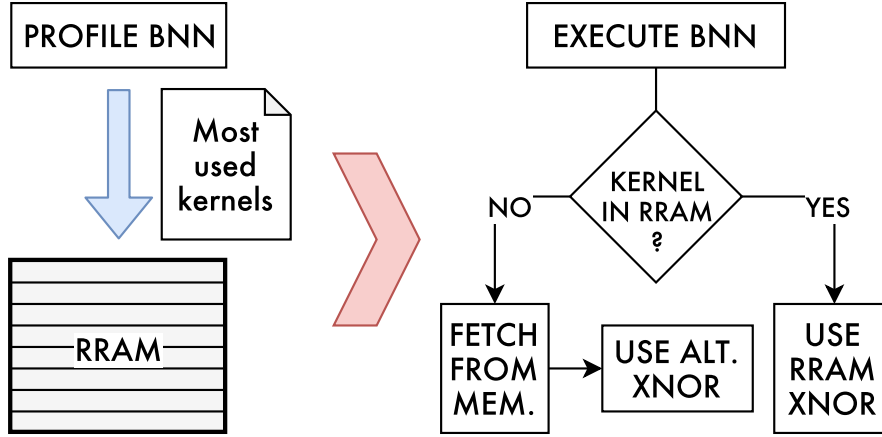


Figure 3.22 – Simple example that illustrates the process of storing the most used kernels inside the RRAM and running a BNN using the novel BDPE.

According to the ARM Architecture Reference Manual for the ARMv8-A architecture profile [39], the ARMv8 ISA has unused *opcodes* that can be re-purposed to expand the functionality of the CPU. Using two of the unused *opcodes*, two instructions were created and assigned to the novel BDPE.

Fig. 3.21 illustrates the format of the new instructions and denotes the purpose of each distinct set of bits. Each of the new instructions is decoded in the *Decode* stage such that the content of the register specified by *rm* serves as input data of the BDPE; the content of the register represented by *rn* is the input kernel; *imm6* specifies the address of the kernel stored in the RRAM array; and the second Most Significant Bit (MSB) of the *opcode* designates the control bit.

After determining which kernels to store in the RRAM and the respective RRAM addresses, an additional phase is added to the compilation workflow to replace regular binary dot products using the processor’s ALU with the corresponding instruction using the BDPE. By controlling the *opcode* (and consequently the *control bit*), either the output of the RRAM array or the output of the alternative XNOR mechanism is used to calculate the final result of the binary dot product operation.

To sum up, as shown in Fig. 3.22, the workflow for running a BNN using the novel BDPE is divided into profiling and execution.

During the profiling, the BNN is used to perform a single inference while the kernel space is

### 3.6. ISA Extensions for In-Memory Computation of AI Workloads

---

profiled, selecting the most frequently used kernels. The selected kernels are stored in the RRAM, and a configuration file is generated containing the information about the content of the RRAM. Then, the CNN is recompiled, and the code responsible for implementing the binary convolution is replaced by custom code that utilizes the BDPE. If the kernel being used is stored in the RRAM, the compiler inserts a special instruction to perform the binary convolution using the RRAM array. Otherwise, the compiler inserts a load instruction to fetch the kernel from memory, followed by a special instruction that performs the binary convolution using the two data inputs of the BDPE.

#### 3.6.1.4 Performance and Energy Evaluation Methodology

The Darknet framework together with gem5-X is used to evaluate the performance of the modified ARMv8 architecture. Darknet is set to operate as a trimmed version of the YoloV3 XNOR-Net CNN, and gem5-X is configured to emulate the ARMv8 in-order core with and without the devised BDPE. Darknet is adapted to allow support for gem5-X System Emulation (SE) mode by compiling all the inputs of the network (the configuration files, the previously trained weights for the SVHN [165] dataset and the input image) into a single executable binary file. Additionally, the Darknet framework is modified at assembly level to use the custom BDPE instead of the processor's ALU when performing 64-bit binary convolutions.

To determine the most used kernels and populate the RRAM array, the following two-step procedure is used: (1) Darknet is run using gem5-X and the kernel space is profiled; (2) The most used kernels are selected and stored in the RRAM. After populating the RRAM, the gem5-X module responsible for emulating the BDPE is rebuilt. Because the framework is not recompiled, gem5-X in SE mode mapped the data structures to the same addresses used in (1), and the application flow is kept the same except for the binary convolutions involving the most frequently used kernels stored in the RRAM. In those cases, the RRAM array is used instead of the alternative XNOR mechanism to perform the XNOR operation. The complete system featuring the modified ARM in-order cores and four DRAM ranks of 1GB each operating at 1200MHz is simulated and the entire workflow of Darknet is executed.

As a result of running the modified version of Darknet, gem5-X produces timing results, statistics on memory accesses and usage of the CPU's several modules. Such results are used to estimate the energy consumption, as described next.

For the energy efficiency and power assessment of the proposed architecture, 28nm FD-SOI power models for ARMv8 in-order cores are used, as discussed in Section 3.5.2. The Delay and power model of the devised BDPE are obtained through circuit-level electrical simulations using a commercial 28nm FD-SOI design kit in Cadence Innovus.

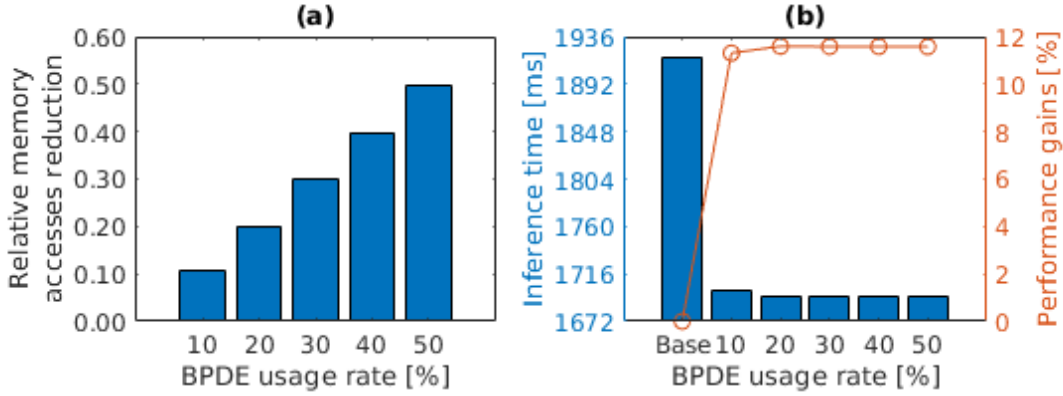


Figure 3.23 – Results showing the performance improvements due to using BDPE. Figure 3.23a shows the relative reduction on memory accesses during the inference phase of YoloV3 for five runs where the usage of the BDPE varies between 10% and 50%. Figure 3.23b depicts the execution time of the inference phase of YoloV3 and the relative performance improvements varying the usage of the BDPE.

### 3.6.1.5 Experimental Results

To better evaluate the impact of the BDPE in the performance of the targeted ARMv8 CPU, five scenarios are considered where the RRAM usage rate (percentage of convolutions that use kernels locally stored in the RRAM) varies between 10% and 50%, when executing YoloV3 XNOR-Net.

By offloading the execution of binary convolutions to the BDPE, the kernels are not requested from the main memory when they are locally stored in the RRAM array. Therefore, a reduction in memory accesses equal to the RRAM usage rate is observed, as shown in Fig. 3.23a. Moreover, over 99% of the memory accesses reduction happens at the L1 cache. Thus, the system counts with the maximum benefits of caching effects.

All in all, as illustrated in Fig. 3.23, for a usage rate of 10% the performance improvement is 11.3%. Also, the performance gains show no significant variation with the RRAM usage rate. This effect has two main causes: (1) both the data paths in the BDPE take exactly one cycle to perform a binary convolution; (2) due to caching effects, the convolutional kernels are stored in the L1 cache 94% of the time, substantially reducing the time required to fetch them. Consequently, using the alternative method for performing the XNOR of the kernel and the input data takes approximately the same time as using the RRAM array and does not impact negatively the overall performance. Nevertheless, thanks to the gem5-X simulation framework, that enabled me to perform this analysis and architectural exploration.

The total energy spent by the baseline system (ARM in-order core) and the five scenarios using the BDPE is illustrated in Fig. 3.24a. Then, Fig. 3.24b shows the energy consumption for the same circumstances subtracted by the energy spent by the DRAM. As illustrated in Table 3.11, the total energy spent by the BDPE is negligible when compared with the rest of the system, and so the energy savings are mostly due to the reduction in the execution time. As the execution



### 3.6. ISA Extensions for In-Memory Computation of AI Workloads

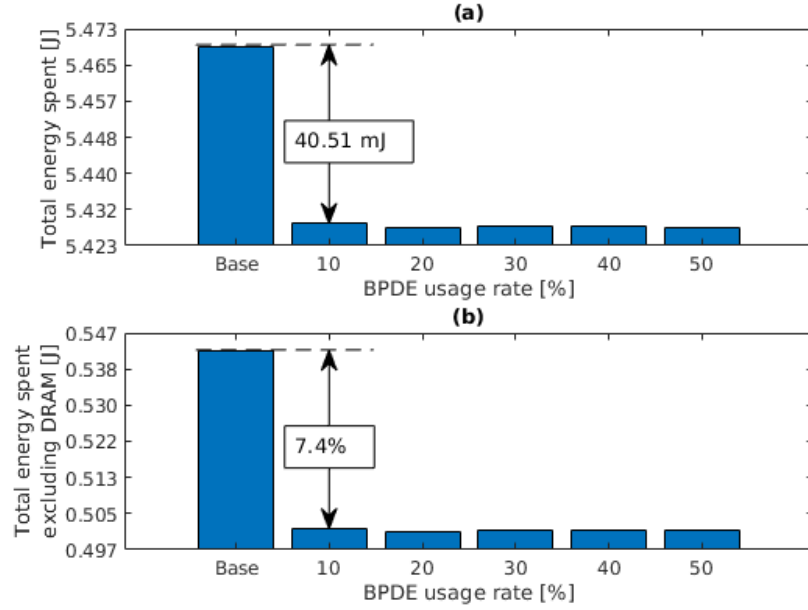


Figure 3.24 – Results showing the energy efficiency improvements due to using BDPE. Figure 3.24a shows the total energy spent when executing the baseline and five runs where the usage of the BDPE varies between 10% and 50%. Figure 3.24b shows, for the same scenarios the energy spent by the system excluding the main memory (DRAM).

Table 3.11 – Total energy spent by the BDPE and the CPU during the inference phase of YoloV3 XNOR-Net.

RRAM usage rate [%]	Baseline	10	20	30	40	50
BDPE [ $\mu$ J]	0	0.870	0.279	0.271	0.263	0.255
CPU [ $\mu$ J $\times 10^6$ ]	0.542	0.502	0.501	0.501	0.501	0.502

time is approximately constant regardless of the RRAM usage rate, so are the energy savings. When considering only the processing system (excluding the DRAM main memory), the use of the BDPE allows for average energy savings of 7.4%.

The advantages allowed by the devised BDPE are tightly coupled with the considered baseline CPU and the used CNN model. Since an ARM in-order core (modeled on ARM Cortex-A53) is used, which is a high-efficiency CPU, the compute power and energy efficiency enabled by the baseline puts it among the most efficient new edge devices. Nevertheless, the use of the devised BDPE allows achieving performance improvements and energy savings at the cost of a minor area overhead. It is also worth saying that should the baseline be a more rudimentary processing system (e.g., an ultra-low power embedded system), the novel BDPE would allow for bigger improvements.

The ML framework Darknet, optimized to achieve the best performance on CPUs, eliminated the

possibility of taking advantage of kernel redundancy, which would increase the RRAM usage rate and consequently allow for higher performance improvements and energy savings. To circumvent this and increase redundancy among kernels, the use of techniques such as weight clustering are proven to be efficient, while sacrificing little accuracy [167]. Nevertheless, since YoloV3 XNOR-Net uses a small subset of the kernels in most convolutions, significant data redundancy was still achieved.

### 3.6.2 Accelerating LSTMs with Analog In-Memory Computation

The latency associated with accessing data from the lower levels of the memory hierarchy is a key performance bottleneck for a wide range of applications, in particular for the increasingly prominent artificial intelligence-related workloads [168]. To accelerate these workloads, AIMC is a highly promising computing paradigm where the computation takes place in-memory [126]. In addition to alleviating the data movement, the AIMC core allows a significant reduction in the computational complexity facilitated by millions of memory devices performing the analog computation in parallel. Matrix-vector multiply (MVM) operations are particularly suitable for in-memory acceleration, enabling both performance and energy gains for various DL workloads like CNNs, RNNs and MLPs [120, 121].

I propose to integrate AIMC core within the execution state of the CPU pipeline, enabling fast data transfer and processing between the tightly coupled AIMC core and the CPU ALU and the register file (RF).

#### 3.6.2.1 Analog in-memory computing - Background

AIMC offers significant advantages in terms of energy and performance owing to two key properties. First, the computation takes place in the memory and therefore, the expensive data movement can be avoided. Secondly, the computation can be done in a massively parallel and analog manner by exploiting the physical attributes and state dynamics of memory devices, as well as circuit laws. Both charge-based (such as SRAM, DRAM, Flash) and emerging resistance-based memory technologies (such as Phase-Change Memory (PCM) and RRAM) can be exploited to build AIMC cores [126]. I will focus on PCM for AIMC. The multi-level storage capability (ability to achieve a continuum of resistance/conductance values) and non-volatility makes PCM a very attractive candidate for inference applications [51].

MVM operations, which form the bulk of computation for DL workloads, can be implemented in the following way in a PCM crossbar array, as shown in Fig. 3.25. The elements of an  $M \times N$  weight matrix are represented as the conductance values of memory devices. Each element of an input vector is translated into the duration of voltage pulse with fixed amplitude  $V$ . The voltage pulses are applied simultaneously to  $M$  word lines and each memory device contributes to the current flowing through one of the  $N$  bit lines, with an amount directly proportional to its conductance  $G$  (Ohm's law). The total current integrated on each of the bit lines over a certain

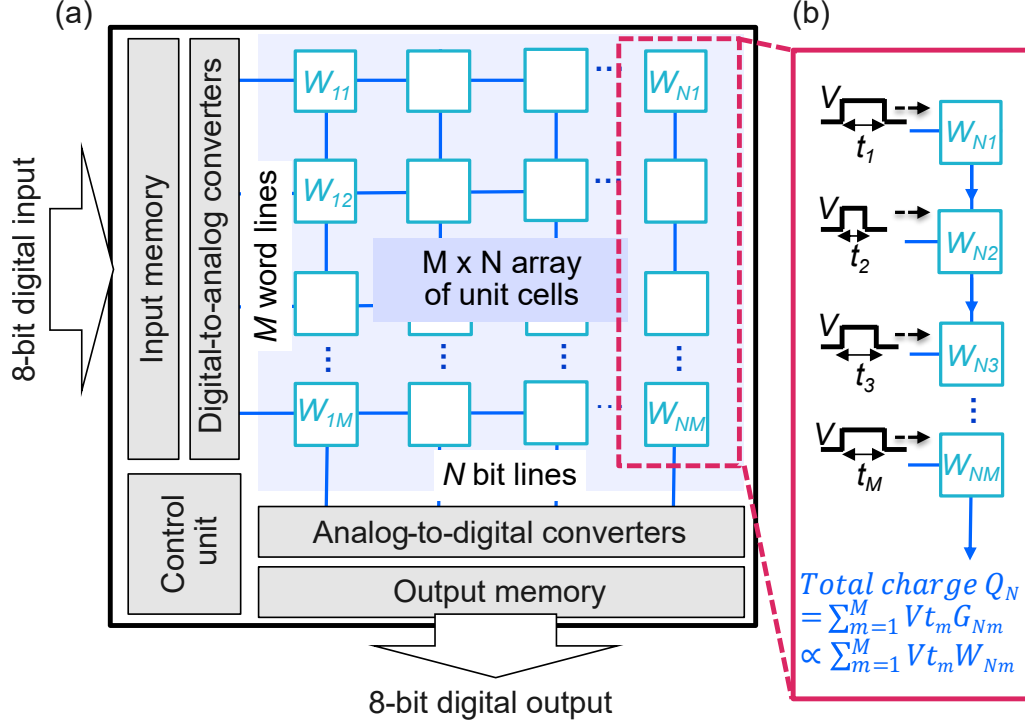


Figure 3.25 – (a) Proposed design of the AIMC core. The core communicates with the CPU through digital input and output signals while the array of unit cells performs the MVM operations in an analog fashion. (b) Schematic illustration of the MVM operation across one of the bit lines.

period of time  $t_{int}$  is indicative of the result of the dot product between the  $M$ -element vector and a column of the  $M \times N$  matrix (Kirchoff's current law). Hence, the multiplication of an  $M \times N$  matrix with an  $N$ -element vector (or in other words approximately  $2 \times M \times N$  operations) can be performed in a constant amount of time, typically less than a few hundreds of nanoseconds.

### 3.6.2.2 DL acceleration with analog in-memory computing

A potential drawback of AIMC is the reduced computing precision owing to the analog mode of operation; the result of the MVM operation is deciphered from the sum of currents across the bit lines. Some of the contributing factors for the reduced precision are non-idealities associated with memory devices (noise and other temporal conductance fluctuations) and circuit-related effects such as IR-drop. Moreover, the AIMC array typically interfaces with the rest of the AIMC arrays and the other system components through digital signals. This necessitates digital-to-analog (DAC) and analog-to-digital converters (ADC) to be employed. The conversion resolution and accuracy of the data converters, and scaling input and output signals to the range of the data converters are also contributing factors to the analog computing precision.

The reduced computing precision of AIMC poses challenges to perform DL inference with an

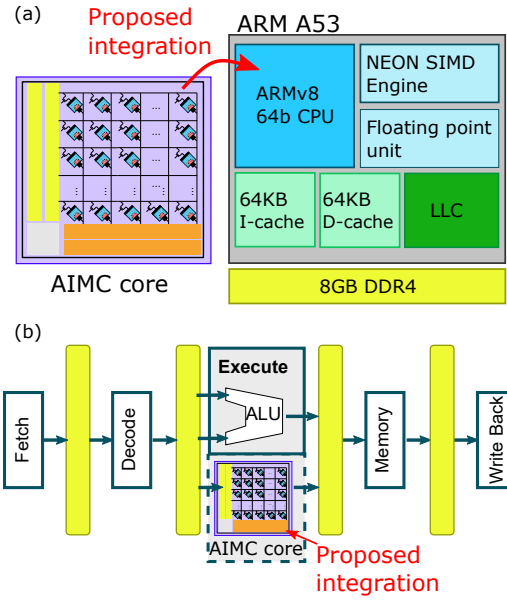


Figure 3.26 – (a) The proposed processor system with the AIMC core integration. (b) A classic in-order pipe-lined CPU instruction cycle. The AIMC core exists alongside the ALU within the execution stage of the CPU.

accuracy comparable to conventional digital floating-point implementations. However, innovations at the algorithmic level as well as AIMC array design choices can ensure iso-accuracy with digital implementations. For instance, custom training strategies that consider memory device non-idealities are shown to be particularly effective in experimentally achieving high inference accuracies with only a marginal loss [121]. Representing weights with multiple memory devices to increase the weight representation precision [122, 169] and applying proper weight compensation schemes [121] to account for temporal device non-idealities are other techniques that can be adopted for an increased computing precision.

In the presence of device-related non-idealities, the precision of MVM on PCM-based arrays is shown to be comparable to a 4-bit fixed-point implementation [170] or even to an 8-bit fixed-point implementation with suitable innovations in device design [171]. Note that 8-bit precision is sufficient to reach state-of-the-art inference accuracies for a wide range of networks [172, 173].

#### 3.6.2.3 Integration of Analog In-memory Computing (AIMC) Cores

The proposed integration of an AIMC core into the CPU system is depicted in Fig. 3.26a. The AIMC core is implemented within the execute stage of the processor, alongside the ALU, as shown in Fig. 3.26b. A key benefit of this configuration is that the transactions between the CPU and AIMC core are reduced to the order of single nanoseconds. This is as opposed to standalone AIMC accelerators where an off-chip communication with the CPU is necessary to access its rich digital functionalities. Furthermore, AIMC enables the stationary weights to be stored within the PCM crossbar array, which eliminates the need for the data of the weights to be fetched from

off-chip main memory and bring it to the CPU.

In addition to the PCM crossbar array in the AIMC core, there are signed 8-bit DACs with input memory/registers and signed 8-bit ADCs with output memory/registers for the data conversion, shown as yellow and orange blocks in Fig. 3.26a, respectively. The digital control unit coordinates the core operation and interfaces with the CPU.

#### 3.6.2.4 ISA extensions for analog in-memory computing cores

For the AIMC core integration, the ARMv8 ISA has been extended with three custom instructions using three previously unused opcodes.

The instructions are used as follows:

- *CM\_INITIALIZE* is used prior to inference to program the AIMC core. After inference has started, the program packs 8-bit inputs into an argument register.
- *CM\_QUEUE* is then called to place the packed inputs into the input memory of the AIMC core. Additional argument registers are used to specify the number of valid inputs packed in the aforementioned argument register, as well as the input memory index. Once all of the inputs are queued into the input memory,
- *CM\_PROCESS* is called to operate the AIMC core by converting the values from input memory into analog voltages, performing the MVM operation with the stationary weights, and storing the results in the output memory after digitizing them. Finally,
- *CM\_DEQUEUE* is called to retrieve packed 8-bit outputs from the AIMC output memory and place them in a destination register. The argument registers specify the number of packed outputs to retrieve starting from what index.

#### 3.6.2.5 Experimental setup

All of the experiments are run in gem5-X FS mode. The system specifications are listed in Table 3.12. Experiments are run for three different system configurations to represent the three different use-cases: *HPC*, *Mid-Tier* (such as mid-low end devices), and *IoT* (edge devices).

Power models as presented in Section 2.7 of Chapter 2 are used to calculate the energy consumption of the ARM based system.

#### 3.6.2.6 Analog in-memory computing core parameters

Table 3.13 reports the performance and energy metrics of the AIMC core estimated from hardware measurements and chip designs in 14nm technology node [120]. The signed weight is represented

Table 3.12 – Gem5-X FS Mode System Configurations.

System	IoT	Mid-Tier	HPC
CPU Core Model	In-order CPU		
ISA	ARMv8 (AArch64)		
CPU Core Frequencies	0.8GHz	1.3GHz	2.3GHz
L1 Data/Instruction Cache Size	32kB	64kB	
L2 (Last-Level) Cache Sizes	256kB	512kB	1MB
Memory Model	8GB DDR4 @ 1200MHz		

Table 3.13 – AIMC core performance and energy figures.

Operation latency of <i>CM_PROCESS</i>	100ns
Operation latency of <i>CM_QUEUE</i> , <i>CM_DEQUEUE</i>	1ns
Read energy per weight	50fJ
AIMC core performance power	20 TOPS/W

with a pair of PCM devices. Only one of the two PCM devices is programmed according to the sign of the weight. The PCM devices of the pair are physically located on separate consecutive bit lines and the differential current is integrated at the ADCs.

The AIMC core is implemented in gem5-X and a direct line modelled between the CPU and the AIMC core within the ARMv8 ISA templates and custom instructions, where operation latencies, configured as the number of CPU cycles, are implemented as a function of the CPU frequency.

#### 3.6.2.7 LSTMs - Recurrent Neural Networks (RNNs)

RNNs, namely LSTMs [50], are used in the experiments. An LSTM consisting of one cell and one fully-connected dense layer is considered, as shown in Fig. 3.27. A *softmax* layer follows the dense layer. LSTMs can greatly benefit from AIMC acceleration, as each of the four gates in an LSTM cell has fully-connected layers. Moreover, the implementation requires a variety of activation functions (*sigmoid*, *tanh*, *softmax*), which can be performed with sufficient precision on the CPU with the proposed architecture.

Various sizes for the cell input, hidden layer and dense layer output are considered, as listed in Table 3.14. Mapped to a single AIMC core are weights and biases for both the cell and dense layers (*Parameters* in Table 3.14). For the LSTM cell, weights associated with each of the gates are organized side-by-side in the AIMC core; the mapping of the LSTM cell weights to AIMC cores is described in more detail in [120]. The inference is run for only 10 steps ( $N_{inf}$ ).

To demonstrate that analog computation incurs only minimal accuracy loss for inference, the largest LSTM has been trained to perform character recognition on the PTB data set [174]. To assess the inference accuracy, the BPC metric [175] was used. The floating-point (FP32) implementation yields a BPC of 1.33 on the full test set while the AIMC implementation

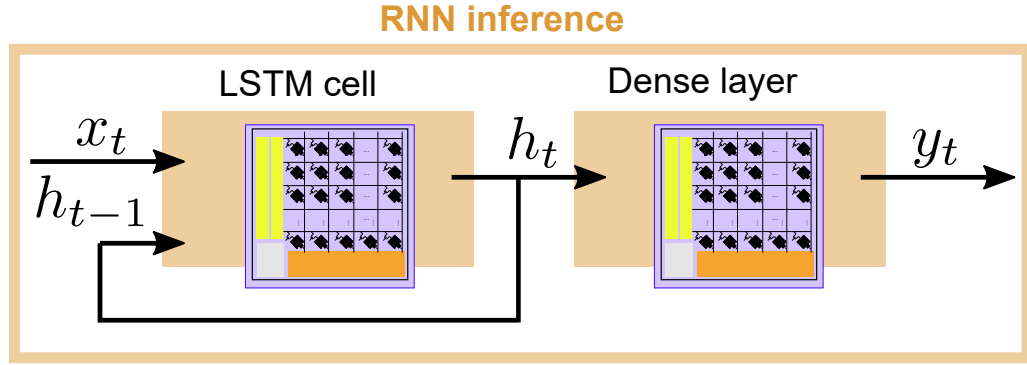


Figure 3.27 – RNN with one layer LSTM with fully connected dense layer. The weights for both the layers are stored in the AIMC core.

Table 3.14 – RNN Network Parameters.

Experiment	1	2	3	4	PTB
Input $x$	24	50	50	100	50
Hidden state $h$	96	192	254	512	750
Output $y$	256	512	778	1024	50
Inference steps $N_{\text{inf}}$	10				
Parameters	71k	285.4k	508.3k	1.78M	2.4M
Working set (INT8)	72.2kB	287.2kB	516.7kB	1.8MB	2.4MB

(simulations with 8-bit signed input, 8-bit signed output, 1.5% Gaussian conductance noise to model PCM device non-idealities) results in 1.35 BPC on the same test set without resorting to any special training methodology.

### 3.6.2.8 Experimental Results

In the experiments, inference is defined as a Region-Of-Interest (ROI), which is a part of run time that consists of loading inputs, propagating inputs through the layers and applying necessary digital functions, and storing outputs. The CPU+SIMD implementation (denoted as *digital*) is compared with the proposed architecture (denoted as *analog*).

In the CPU+SIMD implementation, vector input, weights and biases are loaded from memory in order to perform the MVM operations in the LSTM cell. When running the experiments on the proposed architecture, the AIMC core is used for computing the MVMs. The inputs first queued to the AIMC core through a 32-bit bus and in a similar manner, the outputs are dequeued. As the weights are stationary on the AIMC core, no weight loading is required and the working set only comprises of the inputs and the outputs. The internal cell state of the LSTM cell (FP32) is stored and updated for each inference. The LSTM cell requires *tanh* and *sigmoid* as activation functions

### Chapter 3. Compute-Dominated Architecture Exploration

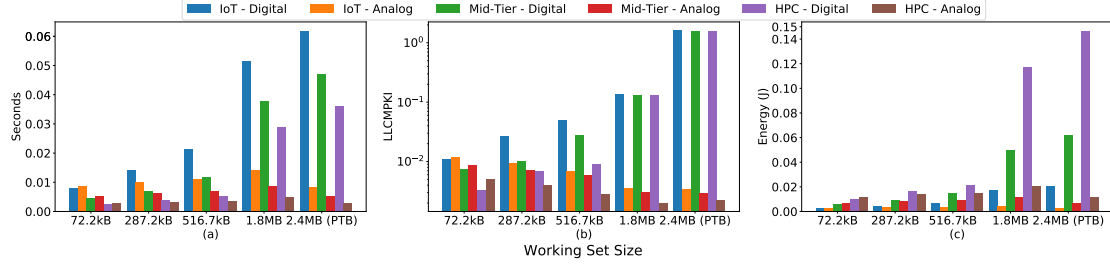


Figure 3.28 – This figure shows the trends in performance per architecture, per LSTM network size. Figure (a) shows the total run time of the LSTM, figure (b) shows the memory intensity (number of LLC misses per 1000 instructions, or LLCMPKI), and figure (c) shows the total energy.

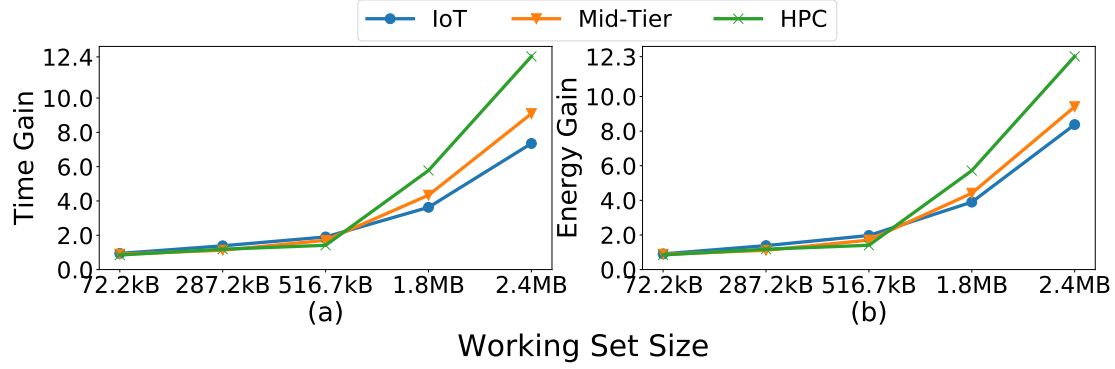


Figure 3.29 – Factor improvement in (a) performance and (b) energy statistics with the proposed AIMC-based architecture over CPU+SIMD implementation.

while a *softmax* is used for the dense layer. The activation functions and element-wise LSTM cell operations are performed in FP32 precision.

The aggregate results for the LSTM experiments, including PTB character recognition are shown in Fig. 3.28. Greater performance and energy gains are achieved with the analog test bench in comparison to its digital counterpart for larger networks. It is observed that the gains are correlated with the LLC size and the working set size, with the gains remaining relatively constant (around 1.3x, average) until the working set size exceeds the LLC size. The IoT, Mid-Tier, and HPC systems have 256kB, 512KB, and 1MB of LLC, respectively. The working set exceeds the LLC sizes of the IoT, Mid-Tier and HPC system for experiments 2, 3 and 4 respectively. Interestingly, for the smallest experiment where the 72.2KB working set fits on the LLC for all the system configurations, the digital implementation slightly outperforms the analog one in terms of time and energy. This suggests that the digital implementation is more favorable when the CPU can easily access the data.

Finally, Fig. 3.29 shows the factor improvement in performance and energy of the proposed approach over the CPU+SIMD implementation. The HPC system has both the largest time and



energy gains (12.4x and 12.3x, respectively). The mid-tier system sees time and energy gains of 9.1x and 9.4x, respectively, and the IoT system sees time and energy gains of 7.4x and 8.4x, respectively.

### 3.7 Summary

In this chapter, I have presented and explored architectures for performance and energy optimization for state-of-the-art emerging compute-dominated applications using gem5-X. These new emerging dynamic applications include video encoding, video analytics comprising video encoding and CNN-based image classification, BNNs, RNNs like LSTMs and banking workload in VMs. These workloads are deployed on different platforms all the way from cloud servers and data centres to mobile devices and were used as case studies to demonstrate the use of gem5-X methodology along with the compute sub-system architectural extensions of gem5-X. Gem5-X is generic and can be used to explore and optimize architecture for any other application.

I have demonstrated integrating and using in-cache computing engine in the L1-D cache to accelerate FIR filter and SATD blocks in the video encoding application. Gem5-X in-simulator profiling capability was also used to first profile the video encoding application and identify the potential bottlenecks. By properly selecting the optimal number, type and operating frequency of the cores, it was demonstrated that in-order cores with in-cache computing achieve the best performance and energy efficiency while meeting QoS requirements. Using video analytics as a case study, I have demonstrated the heterogeneous compute core and clustering capabilities of gem5-X along with heterogeneous memories including DDR4 and the new 3D stacked HBM2 in FS mode. Three case study scenarios of video analytics are used, namely, surveillance, drone navigation and autonomous cars and ADAS, for architecture exploration and optimization, as each has different performance requirements. In the end, thanks to gem5-X and the exploration methodology, an optimal clustered heterogeneous architecture with ARM in-order cores cluster along with a BLADE in-cache computing engine, embedded within L1-D cache was defined as optimal option to process the video encoding kernel, as well as an OoO core cluster was included to process the CNN based image classification kernel of video analytics application. Overall, gem5-X allows to select the right number of clusters and their operating frequencies for different case study scenarios according to the performance requirement in each case. Furthermore, on the memory side, gem5-X proved that the use of HBM2 for the video analytics application led to both performance and energy benefits. Hence, I demonstrated that using gem5-X enables a truly complete and fast design space exploration for many-core architectures.

Furthermore, I proposed NTC servers based on the FD-SOI process technology. The NTC server architecture was modeled and simulated in gem5-X with virtualized workload as the target application, enabled by VM support in gem5-X. The efficiency of the target application was evaluated on three different platforms: (i) x86, (ii) ARM-based Cavium ThunderX, and (iii) proposed NTC server. Moreover, the energy vs. performance trade-offs were explored, when VMs with different CPU utilization and memory footprint characteristics are executed.

### Chapter 3. Compute-Dominated Architecture Exploration

---

Finally, I demonstrated the utilization and integration of CM in the execution stage of the CPU pipeline, by capitalizing on ISA extensions support in gem5-X. BDPE was implemented using RRAMs to accelerate the executions of BNNs. Similarly, the architecture for LSTM-based RNNs was optimized for both performance and energy using analog in-memory computation.

Hence, I demonstrated that using gem5-X enables a truly complete and fast design space exploration for many-core architectures along with novel architectural extensions. The work in [97], [98], [38], [37], [46] and [176] were published as a result of this chapter. The main contribution from [97] to this chapter is the architectural exploration for video encoding Kvazaar case study application, using the in-cache computing accelerator. Furthermore, the work in [38] and [37] present in detail the implementation of BLADE in-cache computing engine in gem5-X. The work in [98] presents architectural exploration and optimization for video analytics application and demonstrates the support for core clustering, heterogeneous cores, and 3D stacked HBM2 memory in gem5-X. The scale-out NTC servers and VM support in gem5-X is demonstrated in the work published in [46]. Finally, the contribution of [176] to this chapter is the use of RRAMs in implementing BDPE within gem5-X for optimal execution of BNNs, in the context of CM integrated within the CPU pipeline.

## 4 Memory-Dominated Architecture Exploration

### 4.1 Introduction

High Performance Computing (HPC) systems hosted on data centers and cloud servers have accelerated the fast paced digitization of the society enabling greater efficiency in all phases of our daily lives. However this comes at the price of a significant increase in computational resources, resulting in data centers consuming 1% (200TWh) of the global energy demand in 2018 [15]. There is a wide range of applications hosted by these data centers for HPC systems, from weather forecasts or particle physics to genomics and precision medicine. These applications are either compute-dominated or memory dominated. In HPC data centers, where performance is the main evaluation metric, x86 based Central Processing Units (CPUs) along with Graphic Processing Units (GPUs) constitute the backbone and de facto industry standard of the processing infrastructure. Recent studies as in [177, 178] have evaluated using energy-efficient ARM cores in the HPC domain. In [179, 46], authors have also looked into ARM based data centers. These works agree that ARM-based systems are more energy efficient as compared to x86 based systems. The benchmarks used for evaluation of performance are compute bounded. In Chapter 3, I also presented different compute-dominated applications in various application domains, including video processing, Convolutional Neural Network (CNN) based image processing, Virtual Machine (VM) based cloud workloads and Artificial Intelligence (AI) applications including Binary Neural Networks (BNNs) and Recurrent Neural Networks (RNNs). All the optimized architectures for these compute-dominated applications, obtained using gem5-X, are ARM-based systems. The works [177, 178] also look into the memory bandwidth (BW) of ARM based systems with traditional DDR as main memory, for memory bounded applications and benchmarks, and conclude that the ARM systems fall behind x86. However, the authors in [178] suggest that ARM based systems might benefit from future 3D high bandwidth memories.

In this chapter, I will explore and optimize architectures for memory-dominated applications, using gem5-X and ARM-based energy efficient systems. In particular, I will present Next Generation Sequencing (NGS), which is a memory intensive application with random memory access pattern, and demonstrate that an ARM based-system with 3D stacked High Bandwidth

Memory v2 (HBM2) can outperform a power hungry and high performing Intel Knights Landing (KNL) system [61], which also comes with 3D stacked memory. Furthermore, I will also look in to optimizing multi-core, multi-threaded version of CNNs, which are both compute and memory intensive, using Scratchpad Memories (SPMs) or software programmable memories, tightly coupled to the CPU core. With the help of gem5-X, it will be demonstrated that using SPMs to pass activations between different layers of CNN, mapped to different CPU cores, leads to both performance and energy benefits.

The main contributions in this chapter, as well as its organization are summarised as follows:

- The single-step architectural exploration methodology is extended for the memory bounded NGS application domain, for optimal performance and energy. It is shown in Section 4.3 that by optimizing the memory sub-system using this extended architectural exploration methodology, HBM2 with no Last-Level-Cache (LLC) can outperform traditional memory hierarchies with caches and DDR4 for sequence alignment for some NGS applications. Furthermore, combining the memory sub-system with compute core optimization, the architectural design space is explored with different core types, core count, frequency and LLC size with HBM2 as main memory for NGS. Moreover, it is demonstrated that many ARMv8 in-order cores surpass or match the performance of fewer ARMv8 Out-of-Order (OoO) cores saving up to 2x energy when both are operating at same frequency and up to 4.5x energy benefits when in-order cores are operating at lower frequency than OoO cores.
- I propose the use of ARMv8 cores along with HBM2, as a replacement for state-of-the-art Intel Xeon Phi 7210 KNL processor with integrated 3D-stacked MCDRAM memory, and demonstrate that 16 ARM OoO cores can match the performance of 32 KNL cores for various case study NGS applications. I also show that 28 ARM in-order cores match the performance of 32 KNL cores for NGS case studies, in Section 4.3.
- A shared SPM architecture is proposed for inter-core data transfer, using Alexnet [62] CNN, as the case study in Section 4.4. The shared SPM enables the activations between consecutive CNN layers to be transferred without going through the cache hierarchy, providing 1.85x performance improvement and 13% energy savings over a baseline cache-only system. To reduce the memory requirements for activations transfer, tiled convolutions in the CNN convolutional layers are also proposed.

## 4.2 Related Work

### 4.2.1 Architecture for NGS Application Domain

HPC systems, usually based on x86 cores, have traditionally been used for fast NGS and sequence alignment process. These x86 CPU-based architectures are also the first point of execution and testing when a new sequencing algorithm is being developed. The widely used BWA-MEM

was tested on Intel Xeon 5420 running at 2.5GHz system, as in [59]. Similarly, Bowtie2 has been reported to run on Intel Xeon X5550 Nehalem running at 2.66 GHz rented from Amazon Web Services (AWS) system, as in [58]. Previous works in [180] and [181] have been done on improving the scalability of well known and state-of-the-art-aligners like BWA-Mem, Bowtie2 and HISAT2 [60] on Intel based HPC architectures like Intel KNL, Skylake and Broadwell architectures [180].

In addition to x86 CPU based systems, GPUs have also been explored for high-throughput sequence alignment task, as in [182] and [183]. The Arioc GPU aligner in [182] achieves up to 10x speedup in comparison to CPU-based system for the seed and extend stage of BWA-MEM. However, the authors have not looked into energy comparisons. Moreover, Intel Xeon X5670 CPUs running at 2.93GHz were used along with the GPUs, which cannot be used as a standalone system. In addition, using GPUs for read alignments is challenging from a software perspective, as one has to manage Single-Instruction-Multiple-Data (SIMD) threading as well as memory management, including data layout and data transfers between CPU and GPU [184]. Dynamic programming dependencies along with memory intensive tasks in sequence alignment add to the challenges of using parallel threaded GPU implementation of NGS [185]. Therefore, GPUs have not been widely adopted for NGS so far.

Sequence alignment accelerators both on Application-Specific Integrated Circuits (ASICs) and Field-Programmable Gate Arrays (FPGAs) have been developed previously. The GenAx accelerator [186] provides around 31.7x speedup as compared to 14-core Xeon E5 server. The authors in [187] implement a dataflow architecture on FPGA for Smith-Waterman Matrix-fill and Traceback stages, widely used in sequence aligners like BWA-MEM [59] and Bowtie2 [58]. However, this only accelerates the Smith-Waterman part of the sequence alignment application and the other stages of the application have to be run on the CPU. In [188], the authors implement Smith-Waterman accelerator on FPGA, which is attached as a co-processor to the IBM POWER8 CPU, and achieves 1.6x speed-up compared to CPU-based version. The DRAGEN platform [189] from Illumina is another state-of-the-art FPGA-based sequence aligner which operates in hybrid hardware-software configuration with a dual Intel Xeon processor. The DRAGEN architecture is proprietary and has not been disclosed, but it is reported to achieve 16-18x speed-up compared to BWA-MEM on a software based system [190]. Despite the speed-up that the sequence alignment accelerators can achieve, it takes a significant amount of time and effort for these accelerators to be developed, either on FPGAs or ASICs. On one hand, if they are on FPGAs, which are resource constrained, and hence, the application have to be scaled out on multiple compute nodes [52]. Additionally, application developed for one FPGA might not be compatible across different FPGA generations. On the other hand, in the case of ASICs, if a new sequencing algorithm is developed, it cannot be adopted for the same ASIC platform. Due to these factors, CPU-based sequence alignment is still widespread and is commonly used for research, as it is fast to deploy and the same system can be used with follow-up versions of the application.

On the memory front, NGS applications are memory intensive. High bandwidth memories like 3D stacked HBM2 are well suited for these applications. In [191], authors use HBM2 with

the Smith-Waterman algorithm on FPGA-based platform and report 2x speed-up compared to DDR4 memory system. Processing-in-Memory (PIM) exploiting HBM2 has been proposed in [192] for the seed location filter, used just before the alignment, and achieve 1.81x-3.65x speed-up compared to a state-of-the-art FastHASH seed location filter. However, in both these works significant software changes were done to use HBM2 with an accelerator on FPGA [191] or as a PIM [192]. Therefore, to the best of my knowledge, this is the first work where it is proposed the use of an ARM many-core compute sub-system along with a high bandwidth HBM2 based memory sub-system for an energy efficient sequence alignment system across three state-of-the-art and widely used NGS applications, namely, Bowtie2 [58], BWA-MEM [59] and HISAT2 [60].

### 4.2.2 SPM for CNNs

SPMs have widely been used in embedded systems since the early 2000s [193, 194]. They are particularly useful when there is a high memory contention in the cache and the working data-set gets evicted from the cache, due to cache thrashing and limited cache size. Previous works have also looked into SPM memory management at the compiler level [195, 196]. In the context of Deep Learning (DL) and CNNs, SPMs have been used in CNN hardware accelerators on ASICs like the Eyeriss [197] or on FPGAs as in [198]. The work in [199] is also an FPGA-based CNN accelerator using SPMs. SPMs have been proposed to be used as local private memory only for the Processing-Elementss (PEs) in [197], [198] and [199]. As the parameters in CNNs are usually very large, local SPMs allow the most frequently used parameters (kernel weights) to be stored close to PEs, reducing the time to fetch these parameters from the main memory. They are also used to store intermediate results of the PEs. However, SPMs have not been proposed to transfer data between different layers of a CNN. Furthermore, the previous works in [197, 198, 199], use SPMs in CNN accelerators.

To the best of my knowledge, this is the first work, where it is proposed to use SPMs for transferring data from one CNN layer to another, with different CNN layers running on separate compute cores. Furthermore, this is the first work to deploy SPMs to improve CNN inference performance on general purpose compute cores, in a multi-core system. Hence, it also paves the way forward for not only using SPMs on multi-core embedded systems, but also on many-core servers.

## 4.3 Design Space Exploration for Genome Sequence Alignment

### 4.3.1 Introduction

Next generation biomedical applications, like genome sequencing, are having an astounding impact in the fields of bioinformatics, cancer research, food microbiology and drug discovery [52, 200, 53, 54]. Genome sequencing is also one of the first steps in understanding a new

### 4.3. Design Space Exploration for Genome Sequence Alignment

---

disease, its effects, development of diagnostic tests and possible cure and vaccines for it. This process should be fast and efficient in case of a new disease outbreak, in order to curtail it, as has been evident during the recent outbreak of the novel coronavirus (COVID-19). SARS CoV-2, the virus that causes COVID-19, was first completely sequenced in China by 11 January 2020 and shared with the world community [6]. The Institut Pasteur in France sequenced the genome for COVID-19 in three days [6, 201]. Since the availability of the genome sequence of COVID-19, scientists and researchers around the world raced towards the development of vaccine and diagnostic kits, in order to cure and overcome the outbreak.

HPC architectures are usually used for fast NGS due to the complexity of the sequence alignment process. These HPC systems are extremely power hungry and performance is usually the only evaluation metric. However, performance alongside with energy should be considered together for enabling scalable HPC systems. HPC systems run both compute-bounded and memory-bounded applications. Genome sequencing [52, 55], which is the process of determining the DNA sequence or the order of bases As, Cs, Gs, and Ts making up the organism's genome, is among the latter type of applications due to pointer chasing [202]. NGS is a high-throughput genome sequencing method. Before the advent of NGS, Maxam and Gilbert [203] and Sanger along with his colleagues [204] came up with techniques to sequence DNA by fragmentation and chain termination, respectively, in the 1970s. The Sanger sequencing was further commercialized and became the de facto sequencing technique for 30 years. It has the honour of being the sequencing method used for the complete human genome in 2004 [205], through a 13-year effort under Human Genome project with an estimated cost of \$2.7 billion [206]. With the advent and rapid developments in NGS, the human genome was again sequenced in 5-months at a cost of \$1.5 million [207]. The common feature NGS platforms share is large parallel sequencing of clonally amplified or single DNA molecules separated spatially in a flow cell [206]. This is a departure from Sanger sequencing, which uses separate chain-termination for individual sequencing reactions. NGS is a huge parallel process that generates hundreds of mega-bases to giga-bases of nucleotide-sequence output in a single instrument run [206].

In sequence alignment, which is one of the first steps in NGS, a sequence read is aligned or checked against a genomic reference for regions of similarity [55]. This process must tolerate differences between the query read and the reference genome, due to errors in the sequencing process and genuine differences between organisms. In addition, the strategies used in sequence alignment have a pointer-chasing nature. They involve a repeated series of irregular memory access patterns through which they determine the memory address of the next (pointer) access, and the previous accessed data is required. Depending on the length of the sequence to be read, this pointer-chasing nature affects the performance of the application. In [208], it is shown that these NGS applications are memory bounded with 40% stalls due to memory, and 80% of these memory stalls are accounted for by long latency DRAM accesses. As these applications are memory bounded, having high performance compute nodes does not help in improving performance, but actually adds to the energy consumption of the system, due to underutilization of the processors. Instead, simpler ARM based architectures along with high bandwidth memories can help in having an energy efficient architecture for NGS with better performance compared to

existing solutions, leading to global energy savings for HPC data centers.

NGS applications use full-text indexing strategies, such as the FM-index, based on Burrows-Wheeler Transform (BWT) [56, 57] for fast sequence alignments. Additionally, Bowtie2 [58] is a new state-of-the-art NGS application based on FM-index, with efficient multi-threading capabilities. BWA-MEM [59] is a widely used sequence alignment application also based on BWT. Then, HISAT2 [60] is a graph based sequence alignment application and superior to both Bowtie2 and BWA-MEM in performance. HPC class compute resources like the Intel Xeon Phi KNL processors [61], Intel Skylake and Broadwell architectures, which support multiple threads, are used to run these NGS applications and maximize their performance [56, 180]. GPUs are also explored for genome sequencing as in [182]. In [209], authors utilize HPC type many-core x86 clusters for NGS. However, the underlying behaviour in NGS applications, whether it is based on FM-index or graph based search, behaves like pointer chasing, with random accesses to the memory and low cache locality. Thus, HPC nodes like KNL or GPUs are not efficient for these sort of memory bounded applications, as they underutilize and waste the compute resources.

ARM-based scale-out processors [210] have been evaluated for memory bounded data center applications, and reported to perform well as in [46], using traditional DDR4 memory. To the best of my knowledge, no studies have analysed ARM-based architectures along with 3D stacked memories for memory bounded genome sequence alignment applications. In this section, it will be demonstrated that random access memory bounded NGS workload can be executed on energy efficient ARM-based platforms, with performance at par or surpassing that of existing x86 or accelerators like KNL, given that there is enough memory BW available, such as the one provided by HBM2. KNL is selected as the x86 based comparison architecture instead of Intel Skylake or Broadwell. The reason is that KNL has a 3D stacked memory, making the comparison fair across the memory sub-system for both the proposed ARM-based system as well as x86-based system.

In this section, I extend the design space exploration methodology discussed in Section 2.8.1 to find performance, energy and area optimized architectures for the NGS application domain. As NGS applications are memory bounded, this extended methodology first focuses on optimizing the memory subsystem and then the compute sub-system. Using this methodology, an optimized architecture based on 3D-stacked HBM2 [79] alongside energy efficient ARMv8 64-bit compute cores is proposed. I use the gem5-X [97, 98] architectural simulator, an open source, validated and enhanced version of gem5 [31], with HBM2 memory model, presented and discussed in Chapter 2. Three widely used state-of-the-art NGS applications namely, Bowtie2 [58], BWA-MEM [59] and HISAT2 [60] with different search strategies are used as case study applications for performance and energy optimization. The main contributions of this section are as follows:

- By optimizing the memory sub-system using the extended exploration methodology, I demonstrate that many ARMv8 in-order cores surpass or match the performance of fewer ARMv8 OoO cores giving significant energy benefits. I also explore the use of ARMv8 cores along with HBM2 and show such a system can outperform Intel KNL based system with 3D stacked memory.



- I perform a frequency sensitivity analysis for ARM-based systems and demonstrate that an energy benefit of up to 59% and 61% can be achieved, for ARM in-order and OoO systems, respectively, when operating more cores at lower frequency, while matching or surpassing the performance, as compared to fewer cores at a higher frequency.

#### 4.3.2 Sequence Alignment Applications

Different optimized sequence alignment NGS applications exist, including some of the most widely used, relying upon the FM-index data structures and search algorithms. Among them, Bowtie2, BWA-MEM and HISAT2 are representative examples of these applications, and will be used as case study in the next sections. In this section these applications will be described briefly along with the FM-index.

##### 4.3.2.1 FM-index

FM-index is a data structure that allows fast substring searches over large texts [57]. FM-index is based on several data structures and algorithms, such as Suffix Array and BWT. Given a pattern or query  $Q$ , the FM-index allows to find all occurrences of  $Q$  in the text  $T$ . The search process takes the following two steps, as shown in Fig. 4.1a: *count* and *locate*. They are explained next:

1. **Count:** Count is a backward iterative process which performs two rank queries as highlighted in red in Fig. 4.1a and an addition per each character in  $Q$  (starting from the end). In FM-index, rank queries are typically performed using memory-consuming data structures, which store previously calculated data, making it a low computing, highly memory bound operation. As shown in Fig. 4.1a, in the highlighted yellow part, the count step requires accessing random sections of the memory in each iteration. The result of this step are pointers to the first and last position in the occurrences interval of  $Q$  in the sorted list of suffixes from  $T$ .
2. **Locate:** Locate uses the indexes of the rows to access the suffix array, where it finds the position of every occurrence of  $Q$  in the text  $T$ . Locate can be performed in one memory access at the cost of a higher memory footprint. With a reduced memory footprint, it also shows a random memory access pattern.

FM-index data structures are used in several well-known sequence alignment applications, such as Bowtie2, BWA and HISAT2 which will be discussed next. All these applications use the *seed and extend* techniques, depicted in Fig. 4.1b. The differences between these alignment applications are depicted through color coded blocks in Fig. 4.1b.

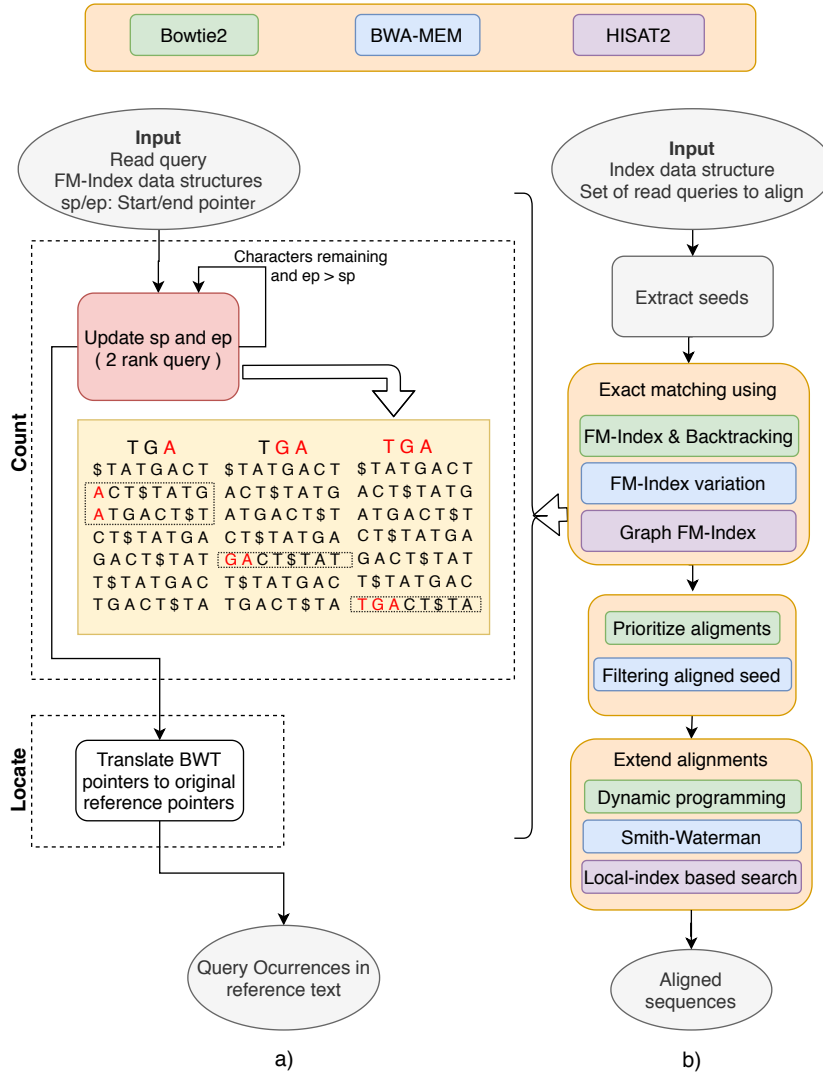


Figure 4.1 – Phases for (a) FM-index and (b) Genome Sequencing Applications.

#### 4.3.2.2 Bowtie2

Bowtie2 [58] is an open-source, ultra-fast and memory-efficient alignment application used for aligning DNA reads to large genomes, able to support gapped alignments. It relies upon the BWT and the FM-index algorithm to quickly find non-exact alignments that satisfy a specified alignment policy. Bowtie2 includes several novelties over the *seed and extend* basis, as the prioritization of seed alignments in order to reduce the computing power used, as shown Fig. 4.1b. Bowtie2 is scalable and supports multi-threading, and equally distributes the alignment tasks among different threads.

Bowtie2 indexes are optimized in order to use as little memory as possible. This way, a Bowtie2 index for the human genome uses around 3.24GB on disk, and has a memory footprint of just 1.3GB. Compared to other sequence alignment tools, Bowtie2 is 2.5-3x faster than the Burrows

---

### 4.3. Design Space Exploration for Genome Sequence Alignment

Wheeler Aligner (BWA) when both applications are searching for gapped alignments.

#### 4.3.2.3 BWA-MEM

BWA-MEM [59] is another widely used open-source sequence alignment algorithm based on FM-index data structures. It automatically chooses between local and end-to-end alignments, supports paired-end reads and performs chimeric alignment. This algorithm follows *seed-and-extend*, common in other sequence alignment applications as Bowtie2, but including some novelties. BWA-MEM includes an additional step which creates groups of seeds, and filters them in order to reduce unsuccessful seed extension at a later step. The seed extension step also includes a variation focused on reducing the computing time used by sub-optimal seed extension and prioritize end-to-end alignments over local ones.

As most sequence alignment applications, BWA-MEM processes a batch of reads at a time. This algorithm uses this feature to obtain both single-end and paired-end mappings. BWA-MEM also supports multi-threading.

#### 4.3.2.4 HISAT2

HISAT2 [60] is also an open-source NGS application based on the *seed-and-extend* and an extension of FM-index for graphs as opposed to the raw FM-index in previous applications, as depicted in Fig. 4.1b. Named the Hierarchical Graph FM-Index (HGFM), it is composed of a global Graph FM-Index (GFM), representing a population of human genomes and a large set of small GFM indexes, collectively covering the whole genome. These small indexes (named local indexes), combined with several alignment strategies, enable rapid and accurate alignment of sequencing reads.

According to results reported in [60], HISAT2 is faster than any other state-of-the-art sequence alignment algorithms like Bowtie2 and BWA-MEM. Although HISAT2 claims to be scalable and supports multi-threading, I noticed its scalability is limited, both on real hardware as well as in the gem5-X simulator, as will be discussed in Section 4.3.6.3.1.

Overall, there are similarities as well as some differences between the three genome sequencing case study applications, as has already been shown in Fig. 4.1b. I will explore architectures with optimal performance and energy for these NGS domain applications, taking advantage of the similarities of the applications, but flexible enough to cater for the differences between them.

### 4.3.3 Methodology for Architecture Exploration

The NGS application domain is usually memory bounded with random memory access pattern. As discussed in Section 4.3.2, the three case study applications exhibit a memory bounded behaviour with random access pattern, as in a pointer chasing and graph processing applications.

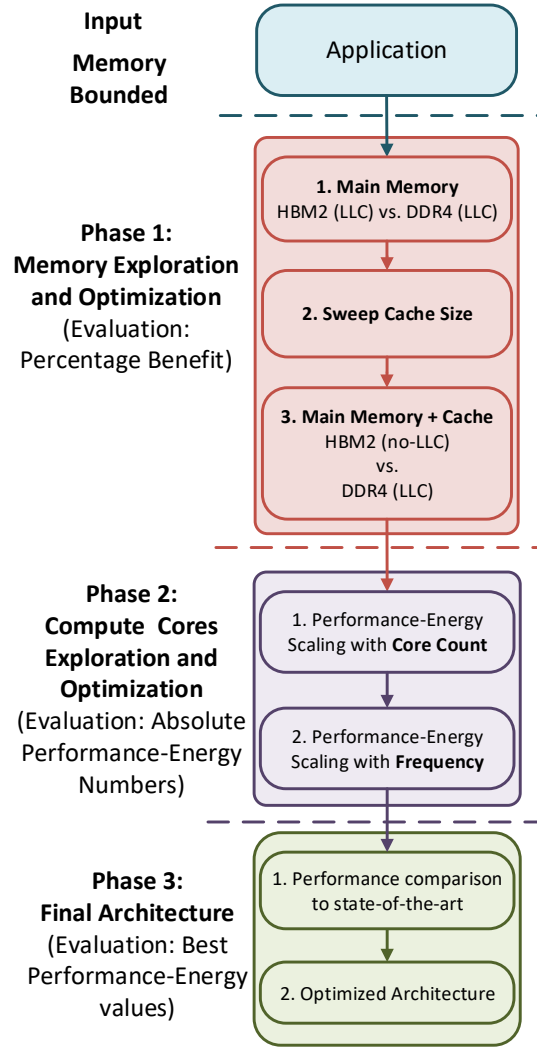


Figure 4.2 – Extended architecture exploration and optimization methodology.

The main compute operation in these applications is the comparison operation, performed when comparing the search sequence to a reference index.

Figure 4.2 shows the architecture exploration and optimization methodology. This methodology is an extension of the gem5-X exploration methodology, presented in Section 2.8.1. In comparison to the methodology described previously, there is no profiling phase in this extended methodology, as already discussed earlier in Section 4.3.1 that, according to [208], NGS applications are memory bounded with 40% stalls due to memory. Moreover, 80% of these memory stalls are accounted for by long latency DRAM accesses. Furthermore, the Phase 2 of the methodology in Section 2.8.1 is extended by all the phases of the methodology in Fig. 4.2, which is intended to optimize architectures for memory-dominated workloads like the NGS application domain. Therefore, the extended methodology first optimizes the memory sub-system and then the compute sub-system, as will be discussed in this section. This extended methodology comprises three phases.

#### 4.3.3.1 Memory Exploration and Optimization

The first phase is related to memory system exploration and optimization. It requires optimizing the entire memory sub-system comprising both the main memory and the cache. As the application is memory bounded with random access patterns, for the first step I analyse replacing the traditional DDR4 with a higher bandwidth memory like the 3D-stacked HBM2 and look into the performance and energy benefits. If the HBM2 based system with LLC outperforms DDR4 with LLC both in terms of energy and performance, it suggests that HBM2 should replace DDR4 in further optimization steps.

During the second step of the memory exploration phase, sweeping the LLC size while optimizing the cache sub-system is looked into. The LLC size providing the best performance is selected as the optimal one.

For the final and third step of the memory exploration phase, the cache sub-system is optimized with LLC being the main focus. A "no-LLC HBM2" system is explored and compared to an "LLC with DDR4" system, both for energy efficiency and performance. If the "no-LLC HBM2" system outperforms "LLC with DDR4" in terms of energy efficiency and performance, it is considered to be a potential candidate for an optimized architecture and is used during subsequent phases of the methodology, otherwise it is discarded. Looking into step-1 and step-3, I am effectively comparing three configurations, HBM2 (LLC) vs. DDR4 (LLC) vs. HBM2 (no-LLC). I do not compare to DDR4 (no-LLC), as it has a lower BW compared to HBM2, and without any caching effects of LLC, it has lower performance compared to DDR4 (LLC) and HBM2 (LLC) configurations.

During this first phase in the methodology, *percentage* performance/energy benefits are used instead of absolute values, as I am looking for architectural choices that are better performing and discarding the others.

#### 4.3.3.2 Compute Core Exploration and Optimization

After the identification of the best memory sub-systems in the first phase, the compute cores are optimized. Two types of cores are used: energy efficient in-order cores and high performance OoO cores. In the first stage of this phase, the performance and energy scaling of the Region-Of-Interest (ROI) with the number of compute cores is explored. The optimized core-count and core type are selected based on performance and energy metrics.

For the second stage of phase 2, I investigate scaling with the core frequency and analyse the core count and core types comparing their performance and energy. I also analyse the area constraints when comparing different core types.

Absolute performance/energy values are used during this phase of the methodology, to have an insight into the best performing system.

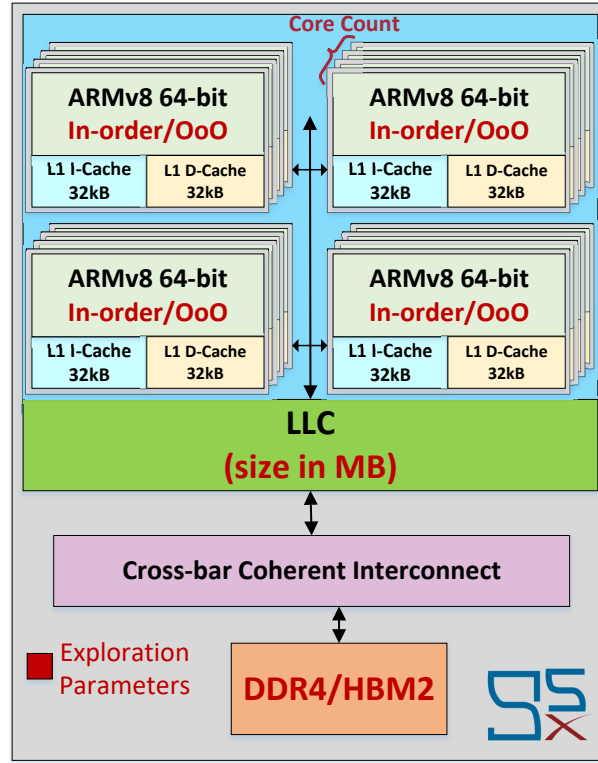


Figure 4.3 – Architectural block diagram of experimental setup in gem5-X.

### 4.3.3.3 Final Architecture

Finally, the optimized architecture is compared with state-of-the-art systems currently being used for the application under study. If the proposed architecture outperforms the state-of-the-art solutions, it is selected as the final optimized architecture.

### 4.3.4 Architectural Exploration and Simulation Framework

Architectural exploration is necessary to find the best architecture for an application, for a given optimization metric. Energy efficiency together with performance is considered as the primary optimization metrics. The gem5-X simulation framework enables us to perform fast architectural exploration for performance/energy-optimized architecture for any given application. It is capable of running multi-threaded applications on a many-core simulated system, as required for HPC applications.

#### 4.3.4.1 Experimental Setup

The gem5-X simulation platform is used in ARM Full System (FS) simulation mode. ARM FS mode is used with an Ubuntu 16.04 operating system (OS), as all the case study applications

### 4.3. Design Space Exploration for Genome Sequence Alignment

---

require various OS multi-threading support. As shown in Fig. 4.3, multiple ARMv8 64-bit in-order and OoO cores are used for the architectural exploration with L1 instruction (L1-I) and L1 data (L1-D) cache fixed at 32KB using the validated ARM JUNO platform [34] as starting point. All the cores are connected to the LLC, which then connects to the main memory of 4GB via a coherent cross-bar interconnect. Gem5-X statistics for the ROI are used for the performance analysis.

#### 4.3.4.2 Power Models

For energy evaluation, the power model for 28nm CMOS bulk technology node is used for ARM OoO and in-order cores, as presented in Section 2.7 of Chapter 2. The power model includes the core active, wait-for-memory (WFM) and static core energy, LLC read/write and static cache energy. For the memory power models, the DRAM power values as reported in [93] are used. Furthermore, counters in gem5-X statistics like active CPU cycles, WFM cycles, cache read and writes hits and main memory accesses are used for power modeling.

#### 4.3.4.3 Gem5-X Extensions

For the different case study NGS applications, the following memory and compute sub-system architectural extensions of gem5-X are utilized:

1. **High Bandwidth Memories:** High bandwidth memories like HBM2 [79] help in alleviating the memory bottleneck of memory bounded applications. I propose to use the 3D-stacked HBM2 memory for such workloads, attaining a BW of 307.2 GB/s. The 3D stacking has been made possible by Through-Silicon-Vias (TSVs), enabling the memory and the logic cores to be placed in the same die resulting in high bandwidth memory accesses. Gem5-X implements the HBM2 memory model by extending the DRAM controller model in gem5, as discussed in Section 2.3.2.1 and 2.6 of Chapter 2. For the power model of HBM2, the energy values of [94] are used.
2. **Core Clustering:** Core clustering in gem5-X enables independent L2 or LLC caches for each cluster, as discussed in Section 2.3.1.5. Therefore, it mitigates the memory BW bottleneck problem on the L2/LLC, as each cluster has its own L2/LLC. Core clustering can be utilized to improve application scaling, if the application performance does not scale with the increasing number of cores, due to L2/LLC being the bottleneck. This will be utilized for HISAT2 NGS application, due to its scaling problem, as discussed later in Section 4.3.6.3.1 and 4.3.6.3.2.

## Chapter 4. Memory-Dominated Architecture Exploration

Table 4.1 – Parameters for architectural exploration and optimization.

Parameter	Values
Core Type	ARMv8 in-order, ARMv8 OoO
Core Count	8 to 64 cores
Core Frequency	1GHz, 1.5 GHz, 2 GHz
LLC size	No LLC, 1MB, 1MB/8-cores, 2MB/8 cores as in Table 4.2
Memory Type	DDR4, HBM2

Table 4.2 – LLC Sizes and scaling with number of cores.

Core Count	Fixed Size LLC	LLC 1MB/8-cores	LLC 2MB/8-cores
8 cores	LLC = 1MB	LLC = 1MB	LLC = 2MB
16 cores	LLC = 1MB	LLC = 2MB	LLC = 4MB
24 cores	LLC = 1MB	LLC = 4MB	LLC = 8MB
28 cores	LLC = 1MB	LLC = 4MB	LLC = 8MB
32 cores	LLC = 1MB	LLC = 4MB	LLC = 8MB

### 4.3.5 Architectural Exploration Parameters

Following the optimization methodology discussed in Section 4.3.3, Table 4.1 summarizes the architectural parameters I sweep to get an optimized architecture, since they have the most impact on system performance and energy.

In the experiments, one software thread is pinned to each physical core. For systems with no-LLC, no more than 28 physical cores are simulated, because the simulation turn around time increases drastically with the number of cores and the scaling trend can already be captured with simulations of up to 28-cores. Similarly, for OoO cores a maximum of 32-cores with LLC are simulated. Lastly, for in-order cores additionally 64-core systems are simulated, which will be discussed in Section 4.3.6.

To explore the effects of varying LLC size, in addition to no-LLC and a fixed LLC of 1MB, the LLC size is also changed proportionally to the number of cores, so to have the same LLC size-to-core count ratio, as in Table 4.2. Two ratios are used, 1MB/8-cores and 2MB/8-cores. The ratio is restricted to 2MB/8-cores (at max.), as the LLC size will increase unrealistically large with the number of cores if this ratio is increased further. The ratio scales well with 8, 16 and 32 cores, but for 24 and 28 cores, according to the ratio of 1MB/8-core LLC size should be 3MB and 3.5MB, respectively. Since these sizes are not a power of 2, the LLC is scaled up to 4MB for 24 and 28 cores. Similarly, for 2MB/8-cores, a size of 8MB is used for both 24 and 28 cores.

### 4.3.6 Architecture Exploration Results

In this section I will explore and optimize architectures for performance, energy and area for the three genome sequencing applications discussed before in Section 4.3.2, using the optimization



### 4.3. Design Space Exploration for Genome Sequence Alignment

---

methodology in Fig. 4.2 presented in Section 4.3.3. ROI is the search and alignment phase of each application. Hence, the performance numbers are given in terms of execution time for ROI. The energy consumption corresponds to the energy in the ROI for the complete system including CPU cores, caches and memory.

As input data, a set of single-end queries generated by the Mason simulation tool [211] were used. These queries, with 200 symbols in average, have been searched in the Parus major (Great Tit) genome reference Parus\_major1.0.3 [212] which is composed of around 1 gigabases.

First an in-depth exploration of Bowtie2 is performed, and then using the insights gained during its architectural optimization, a similar exploration is done for BWA-MEM and HISAT2.

#### 4.3.6.1 Bowtie2 Architectural Exploration

For all the performance and energy results, I launched Bowtie2 in gem5-X, and performed 200K read alignments, which is a representative workload for sequence alignment and stresses the system resources. According to phase-1 of the methodology, first the performance and energy benefits of using HBM2 are explored, with and without LLC, as compared to DDR4, as in Section 4.3.6.1.1 and 4.3.6.1.2, respectively. Then, in accordance with phase-2 of the methodology, it will be discussed how performance-energy scales with core count and frequency, as in Section 4.3.6.1.3 and 4.3.6.1.4, respectively, in quest for an optimized architecture in phase-3.

##### 4.3.6.1.1 HBM2 vs DDR4

I first investigate the performance and energy benefit of using HBM2 instead of DDR4 for Bowtie2 with different core types, core count and LLC size. Figure 4.4a shows the performance benefit of using HBM2 instead of DDR4, along with absolute performance in terms of sequencing time. Hence, the baseline architecture for each bar in Fig. 4.4 is different and is composed of multi-core ARM in-order or OoO cores varying from 8-cores to 28-cores, which operate at 2GHz with DDR4 memory. The LLC for the baselines also varies as fixed 1MB, 1MB/8-core or 2MB/8-cores, namely:

- Architectures with HBM2 always outperform those with DDR4. The performance benefit is higher for OoO cores compared to in-order cores, as the OoO cores can further exploit the memory BW and, therefore, take more advantage of HBM2. The results show that OoO cores utilize 2x more memory BW in comparison to in-order cores, due to the speculative multiple issue of instructions in OoO cores.
- The performance benefit of using HBM2 increases with higher core count for in-order cores, as the increase in the core count results in a larger stress on memory BW, which HBM2 provides in comparison to DDR4. The BW utilization for HBM2 increases from 7GB/s to 22GB/s when varying the core count from 8 to 28 cores, in comparison to 6GB/s-12GB/s for DDR4.

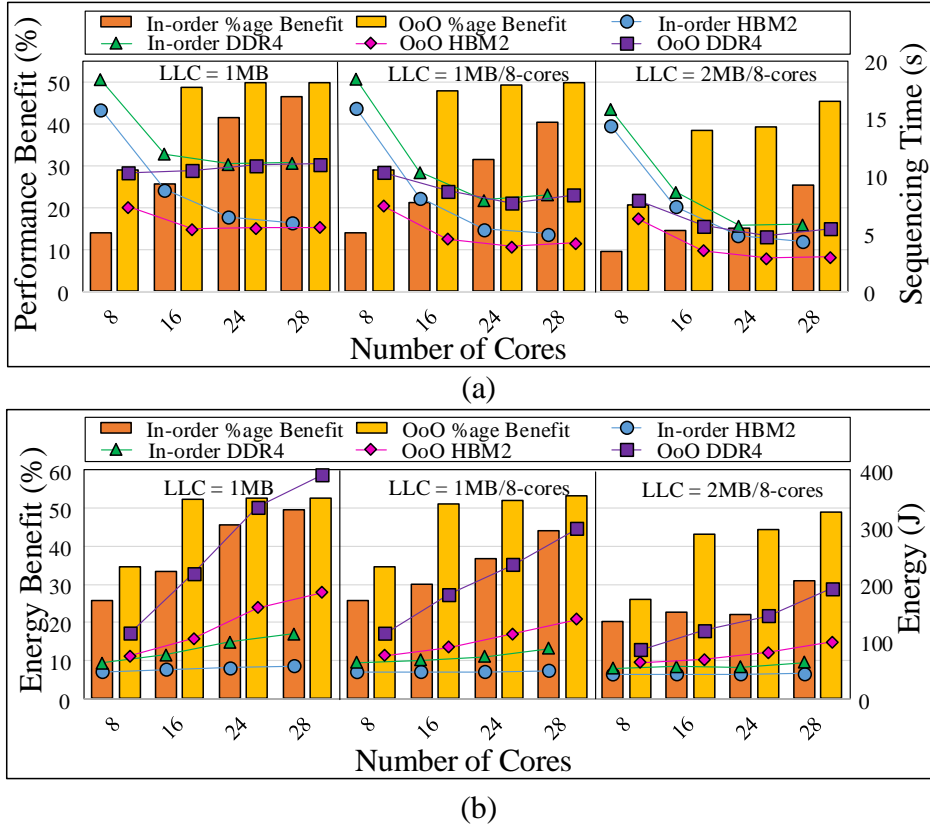


Figure 4.4 – Bowtie2 performance and energy benefit of HBM2 vs DDR4 at 2GHz. The bars represent percentage benefits and the points with lines show absolute value. (a) Percentage performance benefits along with sequencing time. (b) Corresponding percentage energy benefits along with absolute energy values.

- For OoO cores, the performance benefit of using HBM2 instead of DDR4 is constant for 16, 24 and 28 cores with fixed LLC size of 1MB and LLC of 1MB/8-core, but it scales with the core count for larger LLC size of 2MB/8-core. The reason being that LLC is a memory BW bottleneck for both HBM2 and DDR4 systems. As OoO cores are stressing the memory to a larger extent, the performance benefits are almost constant for smaller LLC size. However, larger LLC accommodates more search data and helps in alleviating the BW bottleneck problem, as the results show 2x increase in LLC read hits when increasing the LLC size from 1MB to 2MB/8-core. This is also the reason behind the decrease in percentage performance benefit as the LLC size for a given core count is increased. However, lower percentage benefit does not imply lower performance, as it can be seen in Fig. 4.4a that the absolute performance is always better with larger LLC size for a given core count.
- The performance benefit translates linearly into energy benefit, which scales in an identical (but scaled) way as that of performance, with core type, core count and LLC size. Thus, by replacing DDR4 with HBM2 as main memory, a performance benefit of up to 50% is

### 4.3. Design Space Exploration for Genome Sequence Alignment

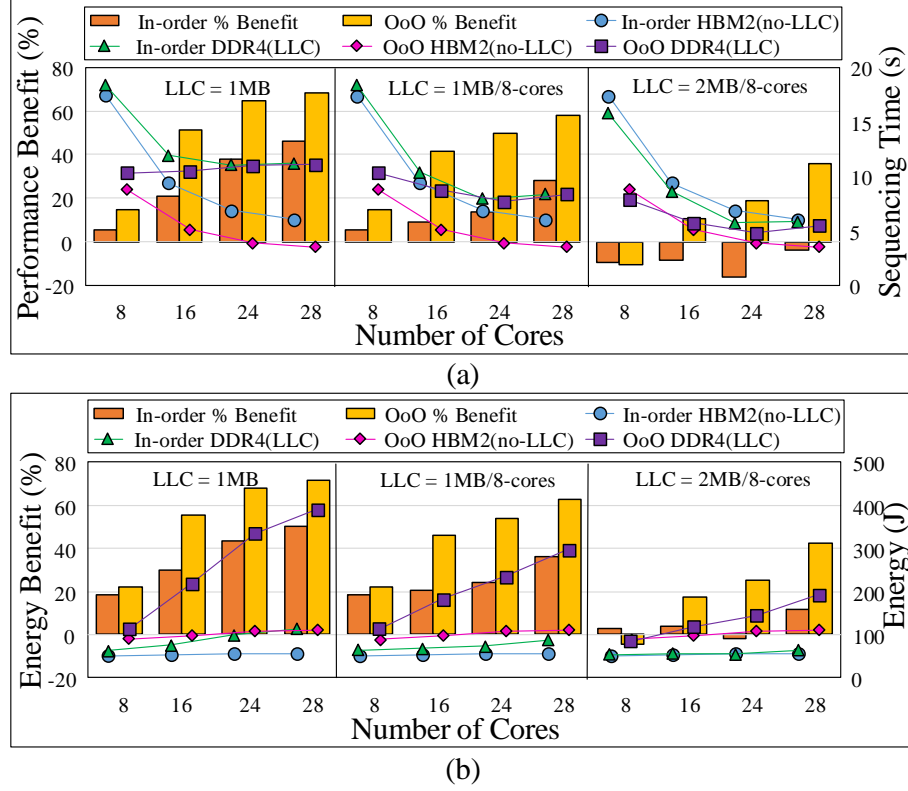


Figure 4.5 – Bowtie2 performance and energy benefits of near memory compute HBM2 with no-LLC systems in comparison to DDR4 with LLC systems at 2GHz core frequency. (a) Percentage performance benefits along with sequencing time. (b) Corresponding percentage energy benefits along with absolute energy values.

obtained, as shown in Fig. 4.4a, and energy benefit of up to 53% as depicted in Fig. 4.4b.

- Experiments are also run at 1GHz and 1.5GHz and with the same scaling trend observed for both performance and energy as that at 2GHz. So, using HBM2 instead of DDR4 bears performance and energy benefits.

#### 4.3.6.1.2 HBM2 (no-LLC) vs DDR4

I explore using HBM2 near to CPU core with no-LLC and compare it to a DDR4 system with LLC for Bowtie2. Figure 4.5 shows the percentage performance and energy benefit of HBM2 in a system with no-LLC as compared to DDR4 with LLC in the system at 2GHz core frequency. Hence, the baseline system has the same compute cores (in-order or OoO ARM cores) at 2GHz with DDR4 memory and LLC varies from 1MB, 1MB/8-cores to 2MB/8-cores, namely:

- OoO cores with HBM2 and no-LLC are always better than that of DDR4 with LLC both in terms of performance (by up to 68%) and energy (by up to 71.5%), except when the number of cores are 8 and LLC is 2MB. In this configuration, the number of cores is not large enough to fully exploit the memory BW and also the large size of LLC helps in

hiding away the latency to the memory, hence, LLC-DDR4 systems performs better. The same trend is true for the energy benefit of OoO cores.

- For in-order cores, the performance and energy benefits increase with the number of cores, except when LLC is larger (i.e. 2MB/8-cores), leading to slightly negative performance benefit. In this case, as in-order cannot stress the memory BW, LLC helps in hiding away the latency to the memory for the DDR4 system. However, energy benefits still remain, even if the performance benefits are negative.
- Experiments were also ran at 1GHz and 1.5GHz and the scaling trend is the same as that at 2GHz.

As no-LLC HBM2 outperforms DDR4 with LLC, this memory configuration will be further evaluated, as well as HBM2 with LLC, as it also outperforms DDR4 with LLC, when following further steps of the optimization methodology.

The methodology does not suggest exploring the DDR4 (no-LLC) configuration, as discussed in Section 4.3.3.1, since it is the worst performing memory configuration. In any case, as sanity check, experiments are performed for DDR4 (no-LLC) for both 28 ARM in-order and OoO cores at 2GHz. The results show that DDR4 (no-LLC) configuration performs at least 2x slower, as compared to HBM2 (no-LLC) for both in-order and OoO cores. Hence, as suggested by the proposed methodology, this memory configuration will not be considered.

### 4.3.6.1.3 Performance-Energy Scaling with Core Count

The scaling of performance and energy with the number of cores is explored for different core types and cache sizes with HBM2 as main memory. The scaling results with DDR4 will not be taken into account, as it was already shown in Section 4.3.6.1.1 and 4.3.6.1.2 that using HBM2 has performance and energy benefits over DDR4. Figure 4.6 and Fig. 4.7 show the performance and energy scaling with number of cores at 2GHz, for different configurations, as described next:

- There are a number of configurations in Fig. 4.6 that either match or outperform the performance of state-of-the-art 32 KNL cores operating in turbo boost at 1.5GHz (maximum KNL frequency) with 3D stacked memory, with 1 thread per core. E.g., it can be observed that 16 ARM OoO cores at 2GHz surpass the performance of 32 KNL cores at 1.5GHz. It can also be observed that 32 ARM in-order cores at 2GHz match the performance of 32 KNL cores at 1.5GHz. As it will be shown in Section 4.3.6.1.4, different ARM cores at 1.5GHz also match and outperform 32 KNL cores (operating at 1.5GHz). A comparison point of 32 KNL cores is chosen, as opposed to maximum 72 KNL cores, since the performance and energy scaling trends are captured with 32 KNL cores in comparison to 32 ARM OoO cores and 64 ARM in-order cores. In fact, this analysis already shows that this configuration surpasses or matches at least the performance of 32 KNL cores. Furthermore, 1 thread per core is used for KNL instead of maximum 4 threads per core for

### 4.3. Design Space Exploration for Genome Sequence Alignment

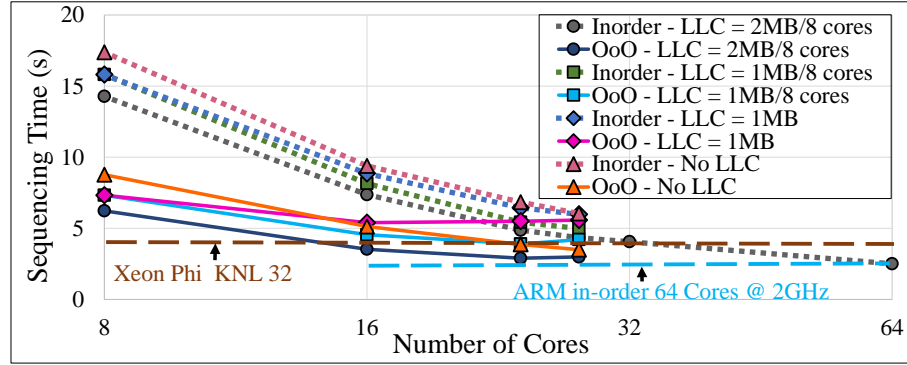


Figure 4.6 – Bowtie2 performance scaling with number of cores at 2GHz using HBM2 as main memory.

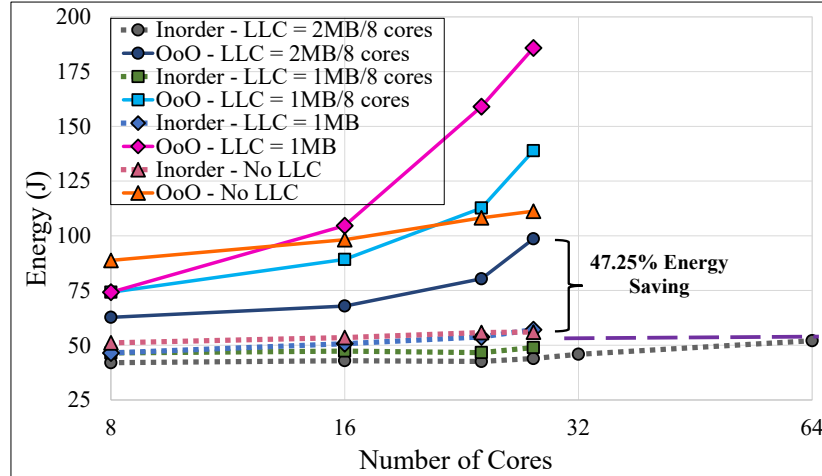


Figure 4.7 – Bowtie2 energy scaling with number of cores at 2GHz using HBM2 as main memory.

comparison to ARM based architectures. The reason being that if the number of threads are increased to 4 per core the performance does not improve, but actually deteriorates. In particular, when increasing the threads from 32 threads to 128 threads on 32 cores of Intel KNL, the performance deteriorates by 37.5% for Bowtie2. This is because the workload is memory intensive and increasing the threads increases the memory accesses and hence the bottleneck on already limited BW memory, which leads to performance deterioration.

- Performance improves when increasing the number of cores, except for OoO cores with LLC. This is because larger core count implies more memory requests through LLC, which gets bottle-necked, leading to performance stagnation. However, it can be seen that systems with HBM2 and no-LLC do not have this bottleneck and therefore, the performance improves with increasing core count. Systems with in-order cores do not exhibit this effect as in-order cores do not generate a lot of memory requests, thus, LLC does not get bottle-necked either.
- Many-core in-order system can match or outperform fewer OoO cores performance, with

much lower energy. To illustrate this, an additional simulation of 64 in-order ARM cores is run, as shown in Fig. 4.6 and Fig. 4.7. It can be seen that 64 ARM in-order cores (at 2GHz) with LLC of 2MB/8-cores, outperform 28 ARM OoO cores at 2GHz by 16.38% and 32 KNL cores (at 1.5GHz) by 37.5%. This leads to an energy benefit of 47.25% over 28 ARM OoO cores.

- If the area is considered, the area of a single OoO core for 28nm CMOS bulk ( $2.05mm^2$ ) is almost 3 times that of a single in-order core ( $0.7mm^2$ ) as reported in [97]. So, 64 ARM in-order cores take 23.8% less area when compared to 28 ARM OoO cores. Consequently, 64 ARM in-order cores with LLC of 2MB/8-cores is the most efficient architecture in terms of performance, energy and core area.
- In all in-order cores systems (with or without LLC) or OoO systems without LLC, as the performance scales with core count, the increase in the energy consumption is not with the same slope, but with a lower slope, as shown Fig. 4.7. This implies that the performance gain is higher with slight increase or almost constant energy.

Many ARMv8 in-order core system with LLC and HBM2 not only outperforms KNL but also fewer ARM OoO core system, in terms of performance, energy and area.

### 4.3.6.1.4 Performance-Energy Scaling with Frequency

I explore the performance and energy scaling with compute core frequencies varying from 1GHz to 2GHz, for different core types and LLC size. For the sizes of LLC, the extremes will be considered, i.e., no-LLC and LLC size of 2MB/8-cores. The trends for both fixed LLC size of 1MB and LLC of 1MB/8-cores are encapsulated between these two extremes, as at 2GHz in Section 4.3.6.1.3, hence, they will not be considered in this section for frequency scaling. Figure 4.8 shows the performance scaling of different architectures with HBM2, with different frequencies and number of cores. The results are discussed next.

- **Outperforming KNL:** I compare the performance of all the architectures against 32-KNL cores (running at 1.5GHz), and demonstrate that there are many-core ARM 64-bit architectures with HBM2 (for different core types, core count, frequency and LLC size), that can either match or surpass the performance line of 32-KNL cores, as shown in Fig. 4.8. The performance and energy of all the ARM architectures outperforming KNL are compared in Fig. 4.9. Firstly, it can be seen that the proposed architectures can match the performance of 32-KNL cores at 1.5GHz with as little as 24 OoO cores at 1.5GHz or 16 OoO cores at 2GHz with LLC of 2MB/8-cores. It can also be seen that 64 in-order ARM cores at 2GHz with LLC of 2MB/8-cores is the best configuration in terms of performance. However, in terms of energy efficiency, 64 in-order ARM cores at 1.5GHz with LLC of 2MB/8-cores is the best with 2.38x less energy when compared to operating at 2GHz.
- **Fewer OoO vs. many in-order cores:** It is observed that many in-order cores operating at lower frequency can match the performance of fewer OoO cores at higher frequency.

### 4.3. Design Space Exploration for Genome Sequence Alignment

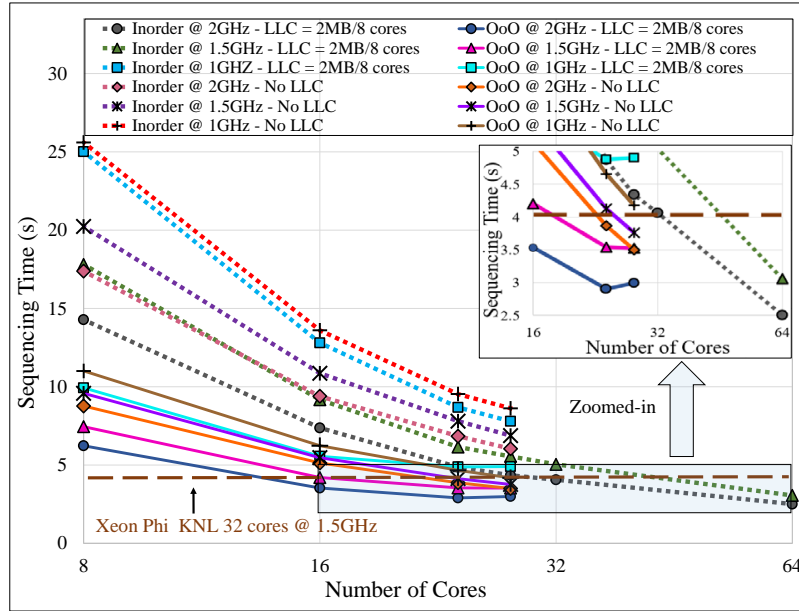


Figure 4.8 – Bowtie2 performance scaling with frequency for different core types and LLC size.

E.g., in Fig. 4.8 28 OoO cores with LLC of 2MB/8-cores 2GHz is outperformed by 16.5% in performance by 64 in-order cores at 2GHz, with energy savings of 1.9x as shown in Fig. 4.9. Furthermore, if the operating frequency of 64 in-order cores is reduced to 1.5GHz, it matches the performance of 28 OoO cores at 2GHz, but with energy savings of 4.5x as in Fig. 4.9. This also results in an area benefit of 23.8%.

- Performance Stagnation vs. No LLC System:** If the zoomed-in section of Fig. 4.8 is observed, it can be seen that systems with OoO cores and LLC have performance stagnated, due LLC not being able to provide enough BW as required by high number of OoO cores, even if HBM2 is the main memory. However, if the LLC is removed, the performance always scales with the increase in core count, and no performance stagnation in the no-LLC system. Therefore, a no-LLC system, is beneficial from scaling perspective for OoO cores as this translates into area savings. But Fig. 4.9 shows that no-LLC has slightly higher energy demand than corresponding LLC systems. So even though stagnation exist for OoO cores, a system with LLC still performs better in terms of performance and energy when compared to no-LLC system. This stagnation effect is not evident in 64-core in-order system with LLC. Thus, a many-core in-order system with LLC is best in terms of performance, energy, area and scaling.
- Frequency Scaling based Energy Savings:** Figure 4.8 and Fig. 4.9 show that 64 ARM in-order cores at 1.5GHz with LLC can surpass the performance of 32 in-order cores at 2GHz, resulting in an energy saving of 2.1x. Similarly, 24 OoO cores at 1.5 GHz match the performance of 16 OoO cores at 2GHz with energy savings of 2.2x.

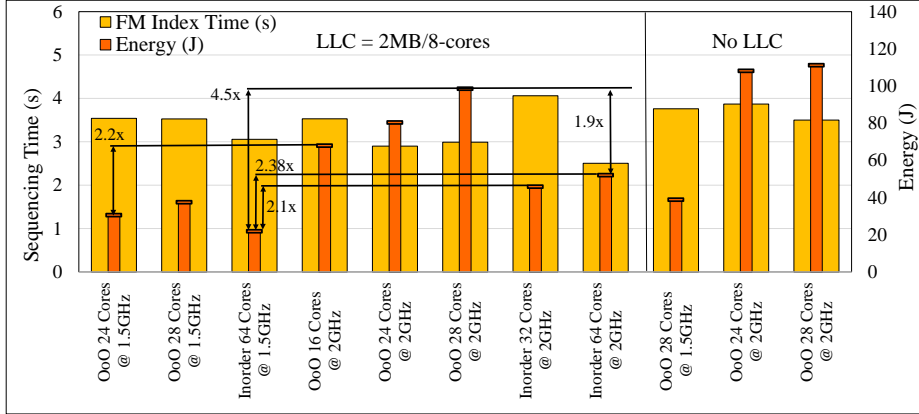


Figure 4.9 – Bowtie2 performance and energy of all configurations surpassing 32 KNL cores performance.

Many-core (64-cores) ARMv8 in-order system with LLC and HBM2 is the best performing when compared to KNL and fewer ARMv8 OoO cores. It is also more energy efficient and takes less area in comparison to fewer OoO cores. When targeting energy efficiency it operates best at 1.5GHz. However, if performance is being targeted, the same systems performs best at 2GHz.

### 4.3.6.2 BWA-MEM Architectural Exploration

Following the exploration methodology in Fig. 4.2 for Bowtie2, a performance/energy-optimized architecture was achieved. Similarly, following the methodology, BWA-MEM is the second genome sequencing application that will be used to explore optimal NGS architectures in this section. BWA-MEM is launched in gem5-X FS mode for 100K read alignments. The number of read alignments are reduced in BWA-MEM as it is almost twice as slow as Bowtie2, so 200K read for BWA-MEM was unfeasible in terms of simulation turnaround time. Similar to Bowtie2, using the exploration and optimization methodology, I first explore the performance and energy benefits of using HBM2, with and without LLC, as compared to DDR4. Then, I present how performance-energy scale with core count and frequency so to have an optimized architecture.

#### 4.3.6.2.1 HBM2 vs DDR4

I investigate the performance-energy benefits of using HBM2 instead of DDR4 for BWA-MEM with different core types, core count and frequency as shown in Fig. 4.10. Hence, the only difference between the baseline and the explored architecture is the use of HBM2 instead of DDR4. The LLC size is fixed at 2MB/8-cores, as discussed for Bowtie2, because this LLC size has the best performance. As BWA-MEM also uses a similar FM-Index strategy as Bowtie2 with some variation, the LLC size is kept the same. Key findings are summarized next:

- It can be observed in Fig. 4.10 that systems with HBM2 achieve up to 6.5% and 10.5%



### 4.3. Design Space Exploration for Genome Sequence Alignment

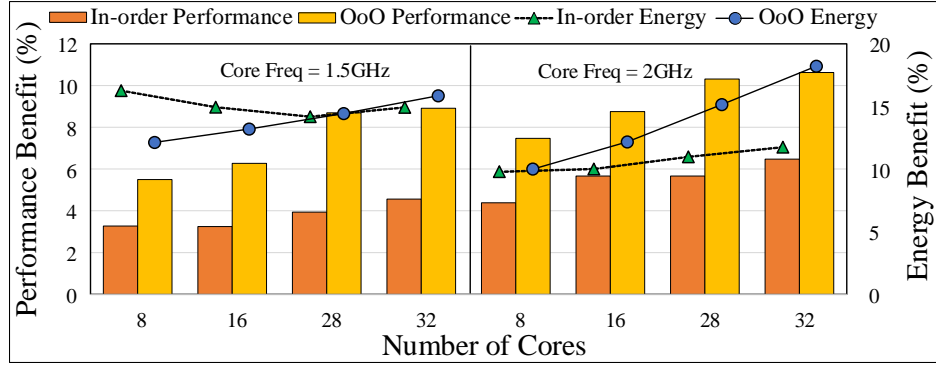


Figure 4.10 – BWA-MEM performance and energy benefits of HBM2 in comparison to DDR4 with LLC of 2MB/8-cores.

performance benefit for in-order and OoO cores, respectively, when compared to DDR4. The performance benefit scales and increases with the number of cores, as well as with the core frequency.

- The performance benefit translates into energy savings of up to 16% and 18% for in-order and OoO cores, respectively, when using HBM2 instead of DDR4.

Thus, using HBM2 instead of DDR4 is beneficial both in terms of performance and energy for BWA-MEM.

#### 4.3.6.2.2 HBM2 (no-LLC) vs DDR4

I explore using HBM2 with no-LLC and compare the energy and performance to a DDR4 system with LLC for BWA-MEM, in accordance with second step of phase-1 in the methodology in Section 4.3.3. For BWA-MEM, a no-LLC HBM2 system, as compared to DDR4 with LLC, is inferior in performance and energy efficiency for both in-order and OoO cores, in contrast to Bowtie2. Therefore, a no-LLC HBM2 memory configuration is discarded for further phases of the methodology.

The experiments are run for DDR4 (no-LLC) for both 28 ARM in-order and OoO cores at 2GHz, as a sanity check. The results show that DDR4 (no-LLC) configuration performs 28% and 38% slower, as compared to HBM2 (no-LLC) for in-order and OoO cores, respectively. Thus, in accordance with the methodology, DDR4 (no-LLC) configuration will not be explored.

#### 4.3.6.2.3 Performance-Energy Scaling with Core-Count and Frequency

I look into the performance and energy scaling of BWA-MEM with the number of cores as well as with the core frequency for both in-order and OoO cores in accordance with phase-2 of the methodology. The architectures of HBM2 with no-LLC will not be explored, as discussed in Section 4.3.6.2.2, that these systems are inferior in performance to DDR4 system with LLC.

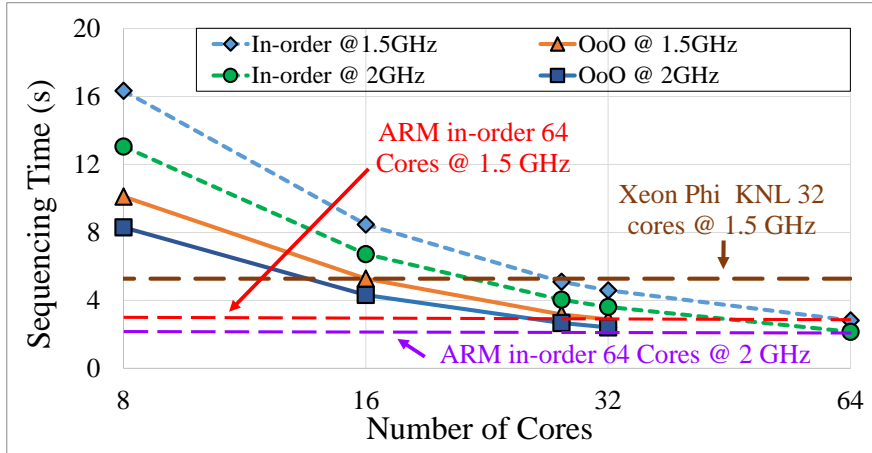


Figure 4.11 – BWA-MEM performance scaling with core-count and frequency. LLC is fixed at 2MB/8-cores.

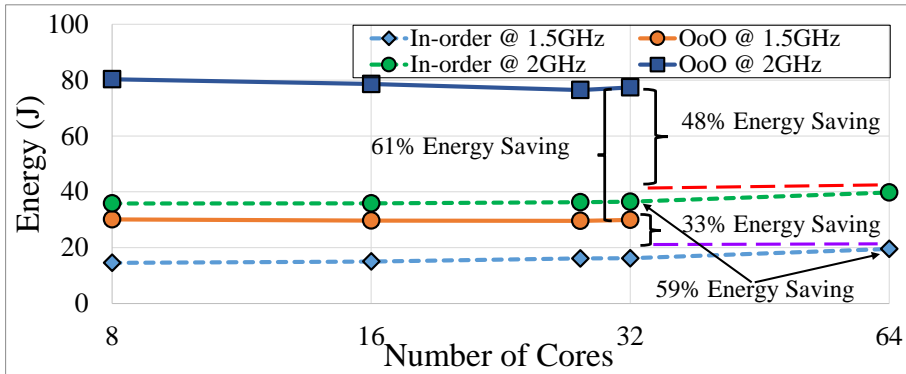


Figure 4.12 – BWA-MEM energy scaling with core-count and frequency. LLC is fixed at 2MB/8-cores.

DDR4 system with LLC will also not be considered, as it was concluded in Section 4.3.6.2.1, that HBM2 system with LLC outperforms DDR4 system with LLC, both in terms of performance and energy. Figure 4.11 and Fig. 4.12 show the performance and energy scaling of BWA-MEM, respectively, for different core-types, core frequency and core count with HBM2, as a main memory with LLC of 2MB/8-cores.

- As shown in Fig. 4.11, the performance of BWA-MEM scales with number of cores as well as with core frequency. It can also be seen that ARM OoO cores outperform in-order cores for a particular core count. However, many-core in-order system can match or outperform fewer OoO cores system. For example, 64 in-order cores match the performance of 32 OoO cores, with both operating at the same frequency (1.5GHz/2 GHz). This leads to energy savings, which are discussed next:

### 4.3. Design Space Exploration for Genome Sequence Alignment

---

- The energy scaling of BWA-MEM with both frequency and core-count is shown in Fig. 4.12. As the performance scales with core count, the energy curves are almost flat, indicating a very small increase in energy with the core count. So, 64 in-order cores, outperform 32 OoO cores at 2GHz with energy savings of 48%. For the same comparison at 1.5GHz, the energy saving is 33%. For 28nm technology node, 64 in-order cores occupy 32% less area when compared to 32 OoO cores.
- Figure 4.11 also shows that 16 OoO ARM cores at 1.5 GHz match the performance of 32 Intel KNL also at 1.5GHz. If compared with ARM in-order cores, which are three times smaller in area than ARM OoO, 28 in-order cores match the performance of 32 KNL cores for BWA-MEM.
- Figure 4.11 shows that 32 OoO cores at 1.5GHz match the performance of 32 OoO cores at 2GHz, giving an energy benefit of 61% as shown in Fig. 4.12. Also, it can be seen that 64 in-order cores at 1.5GHz surpass the performance of 32 in-order cores at 2GHz resulting in energy savings of 59%.

Overall, for BWA-MEM, 64 in-order ARM cores with HBM2 and LLC is the best architecture in terms of performance, energy efficiency and area, consequently, achieving the same optimized architecture than for Bowtie2.

#### 4.3.6.3 HISAT2 Architectural Exploration

Graph based genome sequencing application HISAT2, as discussed in Section 4.3.2.4, is the third and final application studied in the context of energy and performance efficient architecture for genome sequencing applications. HISAT2 was ported to compile for ARMv8 cores. The number of read alignments for HISAT2 are 500K, in-contrast to 100K for BWA-MEM and 200K for Bowtie2, as HISAT2 is faster than both Bowtie2 and BWA-MEM, giving approximately the same simulation turnaround time as for Bowtie2 and BWA-MEM.

##### 4.3.6.3.1 HISAT2 Scaling Problem

When HISAT2 is launched either on a simulation platform, like gem5-X, or natively on a server, it scales in multi-threading mode on a multi-core system (1-thread/core), up to a certain thread count, i.e., around 8 ARM cores in gem5-X and around 16 cores on an Intel Xeon server. After that, performance saturates and does not scale with number of cores. This is due to the lack of further parallelization support in HISAT2. To overcome it, multiple instances of HISAT2 are launched, with each instance using 4 threads (on 4 different cores). Therefore, to stress 16 cores, 4 instances of HISAT2 are launched, each using 4 cores, with the number of read alignments equally distributed among them. This does not affect the memory footprint for HISAT2, as it supports using memory-mapped I/O for reference index which many HISAT2 instances can share.

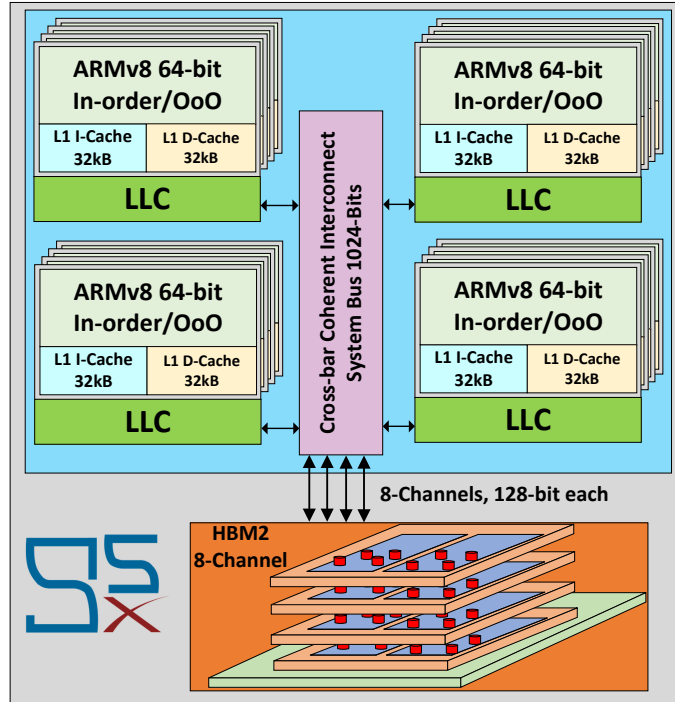


Figure 4.13 – Clustered architecture with 4-cores per cluster for HISAT2.

#### 4.3.6.3.2 Clustered Architecture

As discussed in the previous section, as HISAT2 scales to a limited number of threads, multiple instances of HISAT2 are launched, each using 4-cores. Hence, I propose a clustered architecture with 4-core clusters, each with its own LLC to prevent cache thrashing. This clustered architecture is shown in Fig. 4.13. Each cluster connects to a coherent system bus, which ultimately connects to the main memory, which is HBM2 in the figure, but can be DDR4 as well.

The LLC is set to 512KB for each cluster. This gives 1MB/8-cores, in contrast to 2MB/8-cores for Bowtie2 and BWA-MEM. The reason being that the dataset in HISAT2 is smaller and can fit in a smaller cache. I ran experiments in gem5-X with both 2MB/8-cores and 1MB/8-cores, and observed no difference in performance. Thus, the choice for a smaller cache leads to higher energy and area savings. The performance starts to deteriorate if the cache size is reduced further.

I will now describe the utilization of optimization methodology for HISAT2, as I did earlier for Bowtie2 and BWA-MEM.

#### 4.3.6.3.3 HBM2 vs DDR4

With the clustered architecture for HISAT2, I investigate the performance and energy benefits of using HBM2 in comparison to DDR4 (both with LLC) for different core types, core count and frequency as shown in Fig. 4.14. The LLC size is fixed at 512KB/4-cores (1MB/8-cores). Therefore, the baseline multi-core architecture differs from the proposed architectures only at the memory level with DDR4 being used for the baseline, namely:

### 4.3. Design Space Exploration for Genome Sequence Alignment

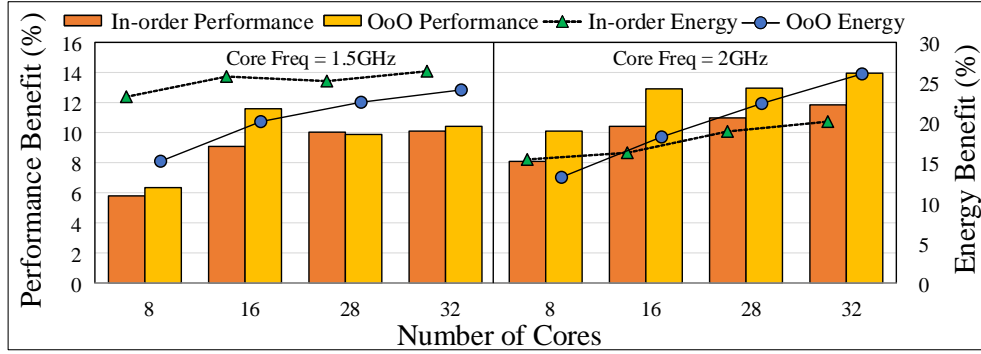


Figure 4.14 – HISAT2 performance and energy benefits of HBM2 in comparison to DDR4 with LLC of 512KB/4-cores.

- For in-order cores, an HBM2 based system performs up to 12% better when compared to DDR4 based system, and up to 14% better for OoO cores. The performance benefit increases with the number of cores as well as with the core frequency, as shown by the bars in Fig. 4.14.
- The energy benefit of using HBM2 instead of DDR4 for both in-order and OoO cores is up to 26%.

Therefore, using HBM2 instead of DDR4 is beneficial both in terms of performance and energy for HISAT2, in a system with LLC.

#### 4.3.6.3.4 HBM2 (no-LLC) vs DDR4

For HISAT2, HBM2 with no-LLC does not perform better than a DDR4 system with LLC in terms of performance as well as energy efficiency for both in-order and OoO cores, as was the case with BWA-MEM. Thus, no-LLC HBM2 configuration for HISAT2 will not be further investigated in the optimization methodology.

As a sanity check of the methodology regarding the DDR4 (no-LLC) configuration, experiments are run for DDR4 (no-LLC) for both 28 ARM in-order and OoO cores at 2GHz. The results show that DDR4 (no-LLC) configuration performs 35% and 38% slower, as compared to HBM2 (no-LLC) for in-order and OoO cores, respectively. Therefore, in accordance with the methodology, DDR4 (no-LLC) configuration will not be explored.

#### 4.3.6.3.5 Performance-Energy Scaling with Core-Count and Frequency

Figure 4.15 and Fig. 4.16 show the performance and energy scaling, respectively, for different core-types, core frequency and core count with HBM2 as a main memory with LLC of 512KB/4-cores. The architectures of HBM2 with no-LLC will not be explored, as discussed

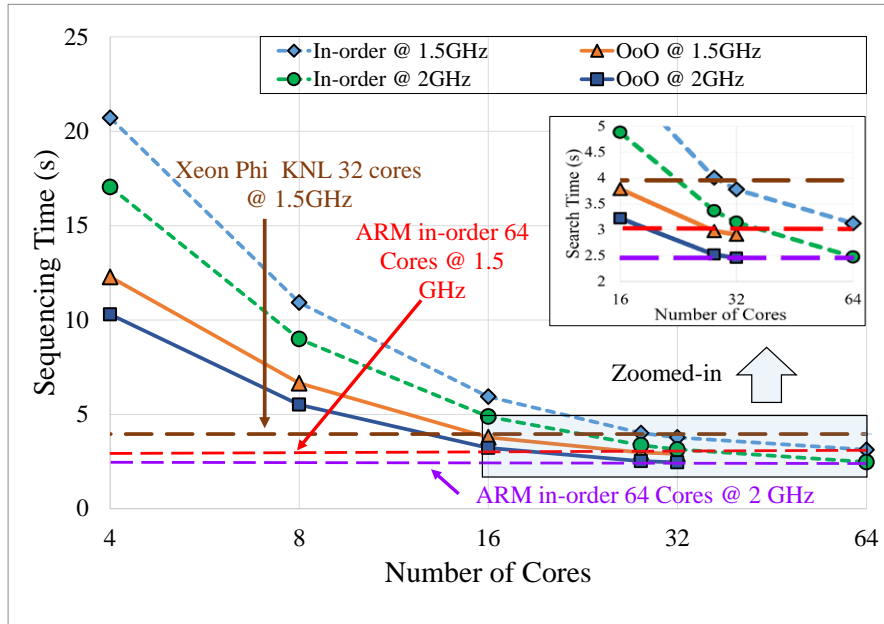


Figure 4.15 – HISAT2 performance scaling with core-count and frequency. LLC is fixed at 512KB/4-cores.

in Section 4.3.6.3.4 that these systems are inferior in performance to DDR4 system with LLC. DDR4 system with LLC will not be explored, as it was also concluded in Section 4.3.6.3.3, that HBM2 system with LLC outperforms DDR4 system with LLC, both in terms of performance and energy. Key findings are described in the following paragraphs:

- Figure 4.15 shows the performance scaling of HISAT2 with number of cores and core frequency. It can be also be seen that for a given core-count, ARM OoO cores outperform in-order cores. However, many-core in-order system can match the performance of fewer OoO cores system. For example, 64 in-order cores match the performance of 32 OoO cores, with both operating at the same frequency.
- Figure 4.16 shows the energy scaling of HISAT2 with both frequency and core-count. As the performance scales with core count, the energy curves are almost flat, indicating a very small increase in energy with the core count. As discussed above, 64 in-order cores match the performance of 32 OoO cores at 2GHz and giving energy savings of 2x. Similarly at 1.5GHz, 64 in-order cores consume 39% less energy than 32 OoO cores, and at the same time matching performance. For 28nm technology node, 64 in-order cores occupy 32% less area when compared to 32 OoO cores.
- Furthermore, the performance of ARM based architectures is compared with Intel KNL. Figure 4.15 shows that 16 OoO ARM cores at 1.5GHz match the performance of 32 Intel KNL also at 1.5GHz. If compared with ARM in-order cores, which are three times smaller in area than ARM OoO, 28 in-order cores at 1.5 GHz match the performance of 32 KNL

### 4.3. Design Space Exploration for Genome Sequence Alignment

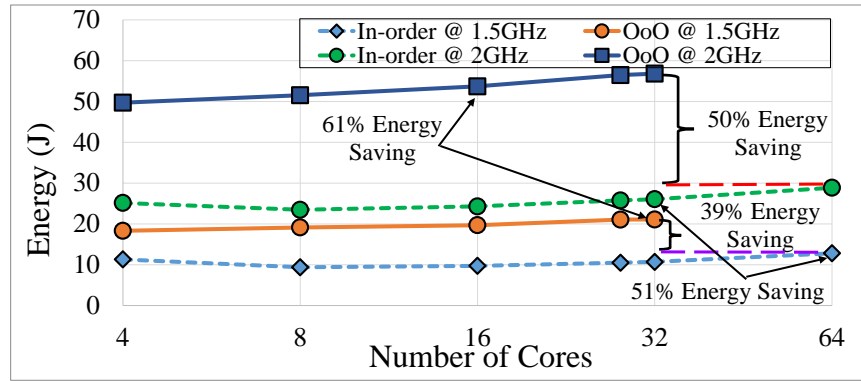


Figure 4.16 – HISAT2 Energy scaling with core-count and frequency. LLC is fixed at 512KB/4-cores.

cores for HISAT2.

- Finally, the performance of the same system configurations are compared at different frequencies. Figure 4.15 shows that 64 in-order cores at 1.5GHz match the performance of 32 in-order cores at 2GHz resulting in energy savings of 51%, as shown in Fig. 4.16. It can also be seen that 32 OoO cores at 1.5GHz surpass the performance of 16 OoO at 2GHz cores giving an energy benefit of 61%.

Overall, in the case of HISAT2, a system with 64 in-order ARM cores with HBM2 and LLC is the best architecture in terms of performance, energy efficiency and area.

#### 4.3.6.4 Discussion of the Results

After following the optimization methodology for three case study applications representing the NGS genome sequencing applications, the following conclusions can be drawn:

- In all the three applications, systems with HBM2 as main memory outperformed their counterparts with DDR4, both in terms of performance and energy efficiency. This leads to a performance benefit of 50%, 10.5% and 14% for Bowtie2, BWA-MEM and HISAT2, respectively, with corresponding energy savings of up to 53%, 18% and 26%, respectively. This is due to the fact that the three NGS applications are memory bounded and, therefore, benefit from high memory BW of up to 307.2 GB/s [79], which is provided by HBM2. It is also observed that the performance and energy benefits of using HBM2 are more for OoO cores compared to in-order cores, as OoO cores can exploit more BW and therefore, take more advantage of HBM2.
- For no-LLC HBM2 systems compared to LLC systems with HBM2 or DDR4, both BWA-MEM and HISAT2 showed that LLC with HBM2 or DDR4 performed better than no-LLC

HBM2 in terms of performance and energy. As discussed previously, HBM2 with LLC was better than DDR4 with LLC. Hence, HBM2 with LLC was selected as the optimized memory sub-system for both BWA-MEM and HISAT2. In case of Bowtie2, no-LLC HBM2 showed performance and energy benefits of up to 68% and 71.5%, respectively, as compared to DDR4 with LLC. Moreover, no-LLC HBM2 outperformed LLC with HBM2, when the LLC is small (1MB/8-core). However, if the LLC is large (2MB/8-cores), LLC with HBM2 was the best performing architecture in terms of performance and energy for Bowtie2.

3. Both ARM OoO and in-order cores outperformed HPC class Intel KNL with similar 3D-stacked memory as HBM2 (used in proposed ARM systems), with fewer or similar core counts at the same core frequency of 1.5GHz, for all the three applications. This is due to the fact that, as the applications are memory bounded, high performing Intel KNL cores do not help in improving the performance. Hence, energy efficient ARM cores coupled with HBM2 surpass the performance of Intel KNL.
4. For a given system, based on either ARM in-order or OoO cores, using frequency and core count scaling, it was shown that a system operating at a lower operating frequency can either match or surpass the performance of that at higher frequencies with energy savings of up to 55% (2.2x) for Bowtie2 and 61% for both BWA-MEM and HISAT2, when using OoO cores. Similarly, for in-order cores, an energy benefit of up to 52% (2.1x), 59% and 51% is achieved for Bowtie2, BWA-MEM and HISAT2, respectively.
5. Many energy-efficient ARM in-order cores either matched or outperformed fewer (almost half) high performance ARM OoO cores at the same operating frequency with energy savings of up to 47.5%(1.9x), 48% and 50% for Bowtie2, BWA-MEM and HISAT2, respectively. This is due to the memory bounded nature of the applications, which do not benefit from high performance cores. As discussed, since in-order are almost 3x smaller in area than OoO, it also led to core-area savings of 23.8% for Bowtie and 32% for both BWA-MEM and HISAT2. Furthermore, in case of Bowtie2, not only in-order cores surpass the performance of OoO cores, but that also at a lower operating frequency, resulting in 4.5x energy savings.
6. HISAT2 application showed performance scaling up to 8 cores in the system. To alleviate this problem, a clustered system was proposed with an instance of HISAT2 on each cluster (4-core cluster with 512KB LLC), and the read alignments distributed among all clusters.

Therefore, many-core ARM in-order system with HBM2 and LLC was the overall best performing system for all three applications in terms of performance, energy efficiency and area.



## 4.4 SPM Memories for CNNs

As of 2018, 41% of Americans owned a digital home assistant, e.g., Amazon's Alexa or Google's Home, and over 25% owned more than three smart home devices [213]. In the same year, 77% of Americans owned a smartphone, for the first time overtaking PC ownership at just 75% [214]. As edge device adoption accelerates, their utilization and influence has also increased in our everyday lives. Hence, the compute power must increase correspondingly to meet application demands. Many of these edge devices make up the Internet-of-Things (IoT) and are connected to the cloud. Hence, the architectures for the application deployed must be optimized all the way from the edge to the cloud. Moreover, as the adoption of edge devices increasingly supplant traditional compute environments such as servers, emerging workloads will increasingly resemble those of traditional servers in function and complexity. As edge devices are naturally limited in physical size and power, innovations must be developed that boost performance while maintaining low area and power overheads.

Since Alexnet [62] was unveiled in 2012, CNNs have become the premier neural net for tasks such as image and language processing [215, 216] and object detection [217]. Given their extreme utility, hardware support on mobile platforms for neural network inference and specifically CNNs has become increasingly attractive [218, 219]. CNNs are considered both compute and memory intensive. I have earlier looked into optimizing the architecture for them from the compute sub-system perspective when considering them as compute-intensive workloads, as in for MobileNet in the video analytics application in Section 3.4.5. However, in this section, I will consider CNNs as memory intensive, and optimize them from a memory sub-system perspective to overcome the Memory Wall.

CNNs are composed of multiple layers. The output of one layer is an input to the next one, with the inputs also referred to as activations. To increase the performance and throughput of the CNNs, they are multi-threaded on a multi-core platform, with different layers of the CNN mapped to different compute cores. To pass activations from one layer to another, they need to traverse through the whole cache hierarchy. As shown in Fig. 4.17, layer  $N$  of a CNN is mapped to core  $N$  and layer  $N+1$  is mapped to core  $N+1$ . To transfer activations from layer  $N$  to layer  $N+1$ , they need to go from L1 of core  $N$  to LLC and then to L1 of core  $N+1$  running the CNN layer  $N+1$ . Hence, the activations traverse the whole cache hierarchy. I propose to use SPM to pass activations from one CNN layer to another, in a multi-core multi-threaded CNN. SPM is a tightly coupled memory to the CPU core at the same level as the L1-D cache. When using SPMs to transfer the activations between the layers (and the cores they are running on), the cache hierarchy is by-passed, thus benefiting both performance and energy, as shown in Fig. 4.18. In this section, I discuss and present the performance and energy benefits of using SPMs for inter-core data transfer, using Alexnet [62] CNN, as the case study.

Gem5-X supports SPMs in the memory sub-system, as presented in Section 2.3.2.2 of Chapter 2. The SPMs can be configured as private SPMs in which they can only be accessible to a specific compute core, or they can be shared where they are accessible by two consecutive cores in the

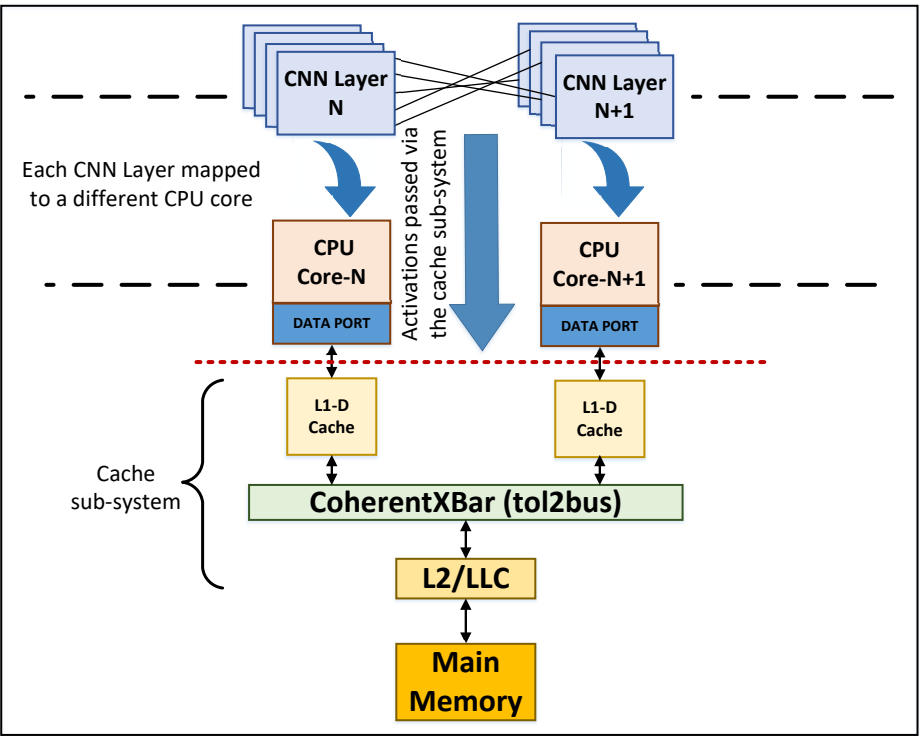


Figure 4.17 – Passing activations from one CNN layer to the next through the cache hierarchy.

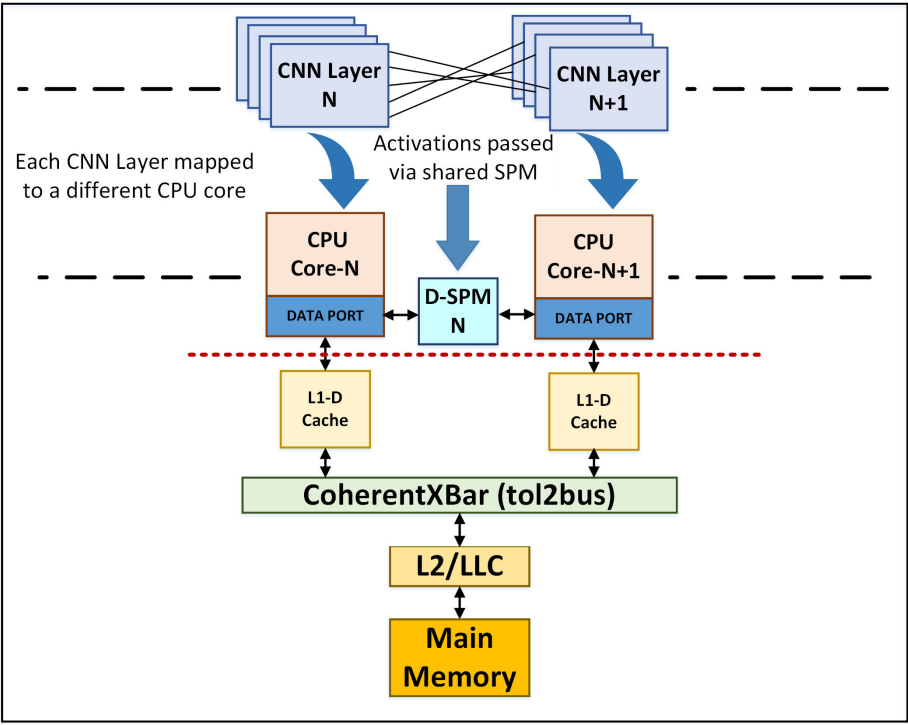


Figure 4.18 – Passing activations from one CNN layer to the next through the SPM.

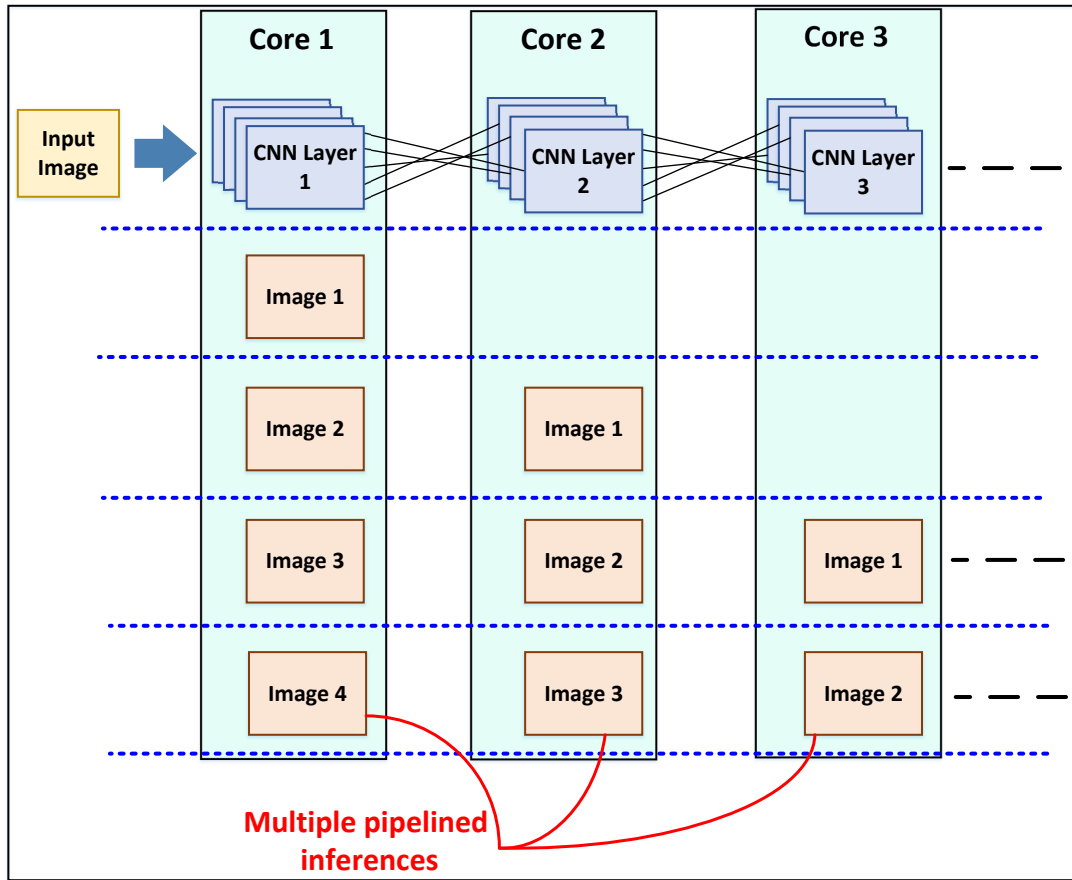


Figure 4.19 – A multi-core CNN system with pipelined inferences.

system. Since SPMs are software controlled, in contrast to the hardware controlled caches, the programmer can store the working data-set in the SPMs, hence, keeping it close to the CPU. The memory map of the SPMs is separate from the cache hierarchy and the main memory. Therefore, to allocate data on the SPM, *mmap()* is used from within the application. Hence, the data allocated on SPM is not cached and with no copy in the main memory.

#### 4.4.1 Application

In this section, Alexnet [62] will be used as a case study CNN, to demonstrate the utilization of SPMs to transfer data between different layers of the CNN running simultaneously on different cores. Alexnet model was implemented in C++ using a C/C++ based CNN solver, presented in [220]. A C++ CNN solver is used instead of python based frameworks like TensorFlow [221], as the memory management of the SPM is done manually by the programmer, which is not possible in python based frameworks.

The Alexnet implementation in [220] is single threaded. This implementation is first improved to incorporate and enable multi-threaded capabilities, using the Linux POSIX threads (pthreads)

[222]. The multi-threaded implementation of Alexnet enables the different layers of the network to run on different physical cores. The layers run simultaneously in a pipelined fashion on multiple compute cores, as shown in Fig. 4.19. For instance, when CNN layer 1 is processing image 3, CNN layer 2 is working on image 2 and layer 3 is processing image 1. Moreover, next layer is triggered only when data from previous layer is available. For instance, when layer 1 completes the processing of image 1, it triggers layer 2. Layer 2 then starts processing image 1 and layer 1 moves on to image 2, and so on. To maximize the throughput and avoid any bottlenecks, two buffers are used in a ping-pong strategy to transfer data between layers (in the implementation without SPM).

Once the CNN has been multi-threaded, the performance improves as multiple images are being processed in parallel on different layers. However, the memory bottleneck still exists, as the activations from one layer to another go through entire cache hierarchy, as discussed earlier. This offers a potential opportunity to use SPM, to transfer data between different cores, executing different CNN layers. However, it needs to be checked first that using SPM will be beneficial, and for that the CNN needs to be profiled, which will be discussed in the next section.

### 4.4.2 Profiling of the Application

To showcase the gem5-X SPM architectural extension, Alexnet [62] will be used as a case study CNN. SPM are beneficial in applications where the working data set does not fit onto the caches, or if there is cache thrashing, leading to high cache miss rates. In such a scenario, the programmer can manage the data in software keeping it in SPMs closer to the compute cores, instead of leaving it to be managed by the hardware controlled caches. However, it needs to be checked first if there is any need for SPMs, i.e., if the cache miss rates are high. Therefore, according to phase-1 of the single-step methodology, presented in Section 2.8.1 of Chapter 2, multi-core implementation of Alexnet is profiled in gem5-X. The architecture simulated for profiling in gem5-X is similar to that of ARM JUNO platform [34], except that all the compute cores are ARMv8 in-order cores, instead of big.LITTLE architecture in JUNO, comprising of both in-order and OoO cores. The simulated architecture is summarized in Table 4.3. Alexnet has 5 convolutional (CONV) layers, each of which is executed on a separate compute core. Two maxpool layers, one after the CONV4 layer and another after CONV5 layer are also pinned to separate physical cores. The fully connected (FC) layers are co-allocated together on a single physical core. Hence, the Alexnet CNN is multi-threaded on an 8-core system.

Table 4.4 shows the L1-D cache miss rate for different Alexnet layers executing on 8 ARM in-order cores. It can be seen that CONV3, CONV4 and CONV5, which are also the largest convolutional layers in Alexnet [62], have a L1-D cache miss rate of more than 20%. These cache misses are mainly due to the transfer of activations from one layer to another, as the L2/LLC miss rate is low and is 5%. Hence, the LLC is used to transfer the data from one layer to another. Therefore, this shows that there is potential for improving performance by bypassing the cache hierarchy and transferring activations via SPMs, with each SPM shared between two consecutive

Table 4.3 – Initial profiling architecture.

Parameter	Value	Parameter	Value
Compute Core ISA	ARMv8	Core Type	In-order
Number of Cores	8	Core Frequency	2GHz
L1-I cache, L1-D cache	32KB, 32KB	L1-I, L1-D associativity	2
L1 MSHRS	4	LLC MSHRS	20
LLC size	512KB	LLC associativity	2
Cache Coherence Protocol	MOESI	DDR4 size	4GB

Table 4.4 – Alexnet profiling on a cache-only system in gem5-X.

CNN Layer	Core Number	L1-D miss rate
CONV1	Core 0	0.6%
CONV2	Core 1	10%
CONV3	Core 2	27%
CONV4	Core 3	34%
MAXPOOL1	Core 4	10%
CONV5	Core 5	24%
MAXPOOL2	Core 6	8%
FC Layers	Core 7	6%

cores.

#### 4.4.3 SPM architecture for Alexnet CNN

After profiling the multi-threaded Alexnet CNN, it is evident that using SPM for activation transfer between the layers (on different cores), can help improve the performance and energy efficiency by bypassing the cache hierarchy, hence reducing the transfer time. To begin with, first, the size of SPM needs to be selected. The same SPM size will be used for all SPMs in the system. Hence, the size is governed by the biggest output activations. For Alexnet, the CONV3 layer has 384 output channels, which are the maximum number of channels among all the Alexnet layers. If the same dual buffer ping-pong strategy, as in the cache-only system is used, the memory requirement for transfer from CONV3 to CONV4 is 2MB. An 8-core system, with 8 2MB SPMs connecting consecutive cores, would imply 16MB for all SPMs. This is quite large and impractical for physical implementation of such a system. Therefore, software modifications are required to allow for a pipelined data-flow from one layer to the next, but with reduced memory requirements.

##### 4.4.3.1 Tiling of the Convolution Layers to Reduce Activations Transfer Size

Figure 4.20 shows a convolution layer in a CNN. Input activations with width  $W$ , height  $H$  and  $C$  number of channels are convoluted with  $C_{out}$  number of kernels, with each kernel of width

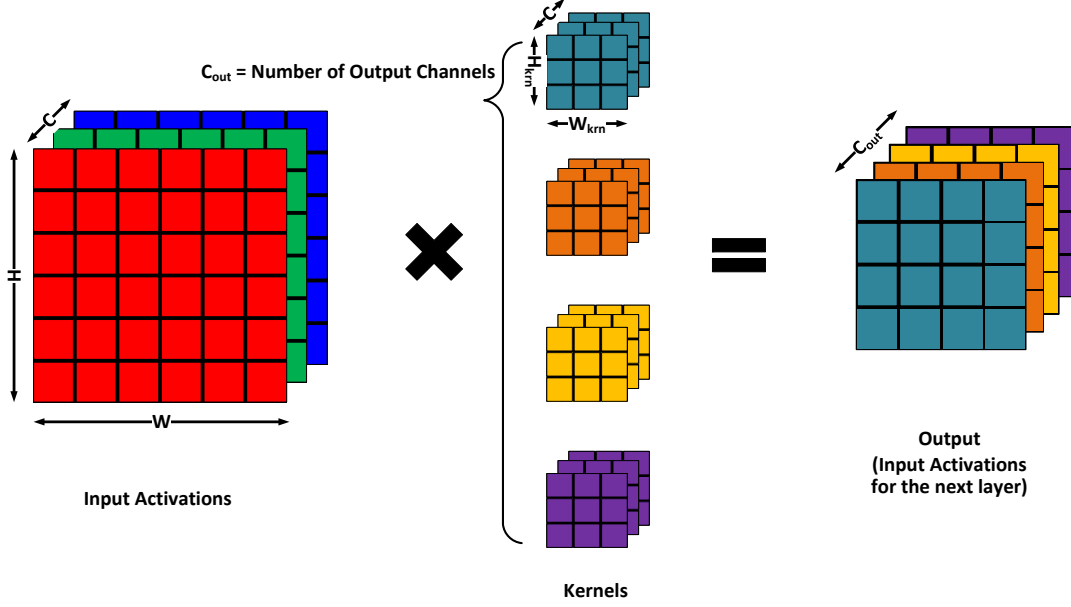


Figure 4.20 – Convolution in a CNN layer.

$W_{krn}$ , height  $H_{krn}$  and  $C$  number of channels. This results in an output with  $C_{out}$  number of channels. The output is the input for next CNN layer. In the cache-only implementation of Alexnet, each kernel is convoluted with input activations in a channel-wise fashion. This implies that all channels of kernel-1 are convoluted with all channels of input activations, to generate channel-1 of the output. Then kernel-2 is convoluted with the input to generate output channel-2 and so on. The drawback of this approach is that all the output channels need to be generated before they can be used as inputs to the next layer, and therefore, a large memory requirement to transfer the activations from one layer to another.

To use SPM for activation transfer from one layer to another with a smaller memory size, as shown in Fig. 4.18, memory re-use is necessary. Therefore, I propose a tiling based convolution operation, which enables the next convolution layer to start processing the output activations of the previous layer, without the need for all the output activations of the previous layer to be available, but just enough to trigger the next convolution layer. The minimum data required for any convolution layer to start is the data availability of all the channels with column and rows equivalent to  $W_{krn}$  and  $H_{krn}$  of that layer, respectively. Based on this, SPM size for all the layers is selected as that for the largest convolution layer, which is CONV4 for Alexnet with 384 channels of the input activations. Consequently this leads to a memory requirement of 119KB, hence a 128KB SPM is used. For Alexnet, in an 8 core system, 8 128KB SPMs are instantiated with each of them connected to the consecutive cores, as in Fig. 4.18.

The memory allocated for the activations in the SPM are stored in a circular FIFO, with empty and full conditions to control the data read/write operations. The maximum number of rows of the activations that are stored in the allocated memory are equal to the number of rows of the

Table 4.5 – Parameters for SPM-based architecture.

Parameter	Value	Parameter	Value
Compute Core ISA	ARMv8	Core Type	In-order
Number of Cores	8	Core Frequency	2GHz
Number of SPMs	8	SPM Size	128KB
L1-I cache, L1-D cache	32KB, 32KB	L1-I, L1-D associativity	2
L1 MSHRS	4	LLC MSHRS	20
LLC size	512KB	LLC associativity	2
Cache Coherence Protocol	MOESI	DDR4 size	4GB

kernel,  $H_{knn}$ , in next layer using these activations. Once the kernel has been convoluted with the data in the FIFO across all channels and columns, its slides down to next rows, by the stride set for that layer. The top rows are now no longer needed and can be re-used to store the next rows. As the data is stored in row-wise circular FIFO, the previous layer can now process more data and store it in the rows that can be re-used. In this way, the data flows in a tiled pipeline manner from one CNN layer to the next, with reduced memory requirements for SPM. The empty and full conditions need to be checked accordingly, for all SPMs, during the data transfer between the layers.

#### 4.4.4 Experimental Setup

To demonstrate the performance and energy benefits of using SPM for inter-layer data transfer in a multi-core CNN implementation, two experimental setups are used in gem5-X. First setup is a baseline system with caches to transfer activations between CNN layers, as shown in Fig. 4.17. This baseline cache-only system is summarized in Table 4.3. Then, the SPM-based system with shared SPM architecture, as presented in Fig. 4.18, is simulated in gem5-X with the architectural parameters summarized in Table 4.5.

#### 4.4.5 Experimental Results

##### 4.4.5.1 Performance Results

Figure 4.21 shows that Alexnet runs 1.85x faster on an SPM based system in comparison to cache-only system. This is because SPM enables the transfer of activations from one CNN layer to the next on consecutive compute cores, bypassing the cache hierarchy. However, as SPMs are added in the system, the total on-chip memory including L1-D, LLC and SPM, is 2.33x greater than the baseline cache-only architecture. Consequently, this increase in the total on-chip memory implies corresponding increase in the on-chip memory area. To have a more fair comparison, the LLC size is increased to 2MB for the cache-only architecture, but remains the same at 512KB for the SPM system. The reason being that in SPM based system, SPM is used to transfer the activations, and in cache-only system LLC is used for this purpose. Figure 4.22, shows the

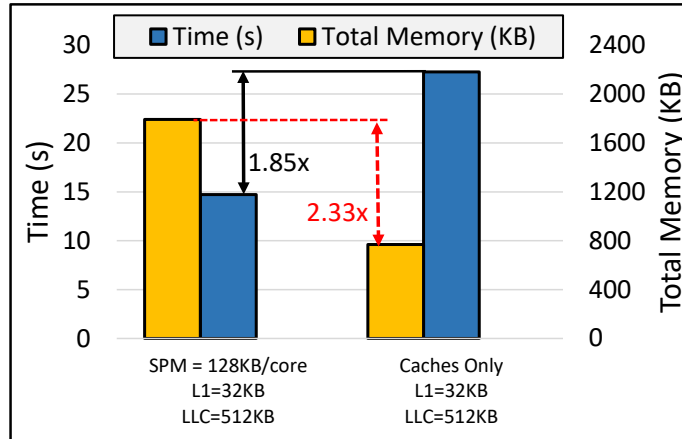


Figure 4.21 – Alexnet performance comparison of cache-only and cache-SPM system. Total memory size, comprising of L1-D, LLC and SPM, is also compared for the two systems.

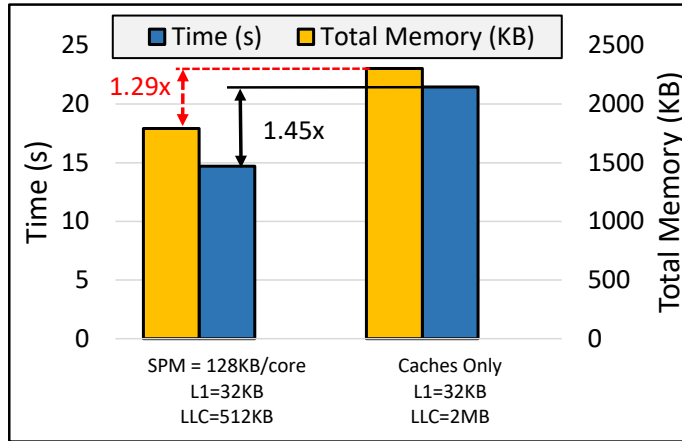


Figure 4.22 – Performance and total memory size comparison of cache-only and cache-SPM system with an LLC of 2MB and 512KB, respectively.

performance and total memory comparison of the two systems. The performance differences between the two systems is smaller as compared to before, but still SPM-based system is 1.45x faster than cache only system, although the cache-only architecture is 1.29x larger in terms of total memory. Hence, this demonstrates that even if the LLC is larger, but as the data needs to traverse through the complete cache hierarchy, the SPM-based system performs better.

Table 4.6 shows the L1-D miss rates for Alexnet on an 8-core SPM-based platform. When the L1-D miss rates are compared to that of a cache-only system in Table 4.4, it is evident that the miss rates have reduced considerably, by up to 3x for some layers. The scaling of performance is also explored in the SPM-based system, with reduction in LLC and L1-D cache sizes. Figure 4.23, shows that the performance remains the same, even if the LLC is reduced to 256KB from 512KB. If L1-D is also reduced to 16KB along with a reduced LLC of 256KB, the performance



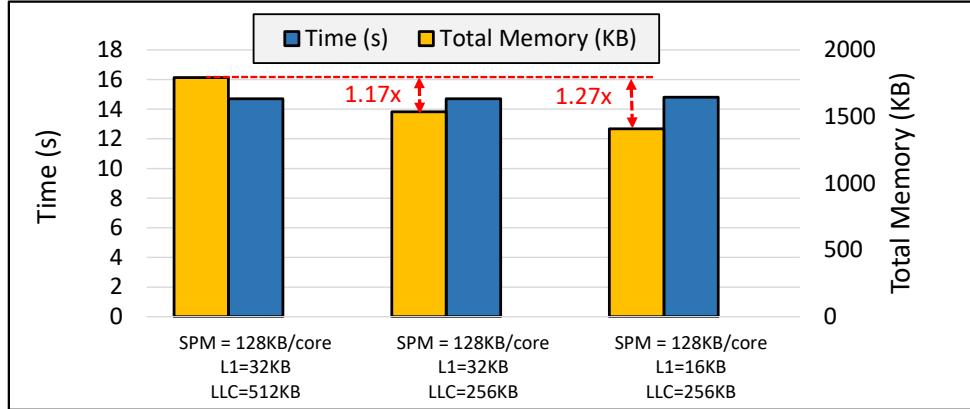


Figure 4.23 – Performance scaling of Alexnet in a SPM-based system with scaling of both L1-D and LLC size.

Table 4.6 – Alexnet L1-D miss rate on SPM-based system in gem5-X.

CNN Layer	Core Number	L1-D miss rate
CONV1	Core 0	1%
CONV2	Core 1	8%
CONV3	Core 2	9%
CONV4	Core 3	9%
MAXPOOL1	Core 4	3%
CONV5	Core 5	9%
MAXPOOL2	Core 6	1.7%
FC Layers	Core 7	7%

only degrades by 0.6%, which is negligible. This reduction in L1 and LLC size to 16KB and 256KB, respectively, further reduces the total on-chip memory by 1.27x, hence resulting in area benefit.

#### 4.4.5.2 Energy Results

SPMs are 30% smaller than a cache and consume around 30% less energy than a cache of the same size [223, 224]. As the SPMs in the proposed architecture are shared, hence, the SPM power model is derived from LLC power model, with 30% less energy. This derived SPM power model is used along with the one discussed in Section 2.7 of Chapter 2 for compute cores, LLC and main memory. Figure 4.24 shows the energy consumption of all systems discussed in the section above. It shows that SPM system with L1 of 32KB and LLC of 512KB is 13% more energy efficient than a cache based system with the same LLC size of 512KB. However, if the LLC size in the cache-only system is increased to 2MB, it consumes 7% less energy than the SPM-based system with 512KB LLC. The reason for this is that, DRAM access energy decreases, with the increase in LLC size, as DRAM accesses are decreased. Hence, the large LLC system

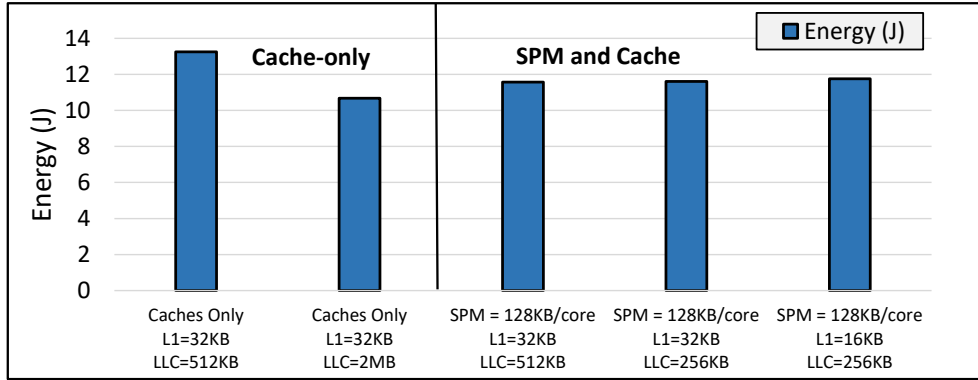


Figure 4.24 – Alexnet energy comparison of cache-only systems with SPM and cache system.

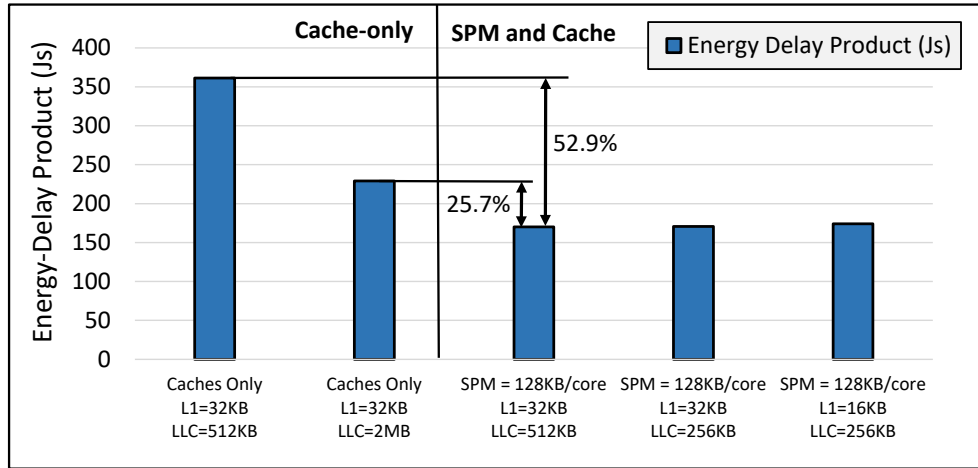


Figure 4.25 – Alexnet Energy-Delay-Product (EDP) comparison of cache-only systems with SPM and cache system.

consume less total energy in comparison to SPM-based system with smaller LLC. Furthermore, if cache sizes for both the L1-D cache and LLC are decreased in the SPM-based architecture, the energy does remains almost constant, with a difference of around 1%, as was the case with performance change.

The SPM-based system has better performance then the cache-only system, however, the cache-only system with a larger LLC consumes less energy. Hence, there is a trade-off between performance, energy and area. The selection of the architecture will depend on the actual performance, power and area (PPA) constraints.

However to decide which is the overall optimal system, the EDP metric is analysed for both systems. Figure 4.25 shows the Alexnet EDP comparison for all the cache-only and SPM-based systems. It is evident that SPM-based systems are outperform cache only systems, as their EDP is lower. The SPM-based architecture with L1-D of 32KB and LLC of 512KB outperform cache-

only platforms with LLC of 512KB and 2MB by 52.9% and 25.7%, respectively. The EDP of SPM-based architectures, with different cache sizes is almost constant. Therefore, in conclusion, using SPMs for transferring CNN activations from one layer to the next, is the optimal in terms of performance and energy efficiency, when compared to a system without SPMs.

## 4.5 Summary

In this chapter, I demonstrated the architecture exploration and optimization of memory-dominated workloads, using NGS and CNN, as case studies. Using the gem5-X architectural simulator and the proposed optimization methodology, architectures were explored to optimize the performance and energy of the system for three genome sequence alignment applications i.e., Bowtie2, BWA-MEM, and HISAT2. All these applications are memory bounded with random memory access. The analysis in this chapter has shown such memory bounded workloads do not require power hungry HPC compute nodes like Intel Xeon Phi KNL, but instead require improvements in memory BW to enhance the overall performance and energy. Furthermore, I have proven that by using high BW memories like HBM2 coupled with energy efficient compute cores, NGS applications achieve up to 68% performance and 71% energy benefit compared to a traditional system with DDR4. It was also demonstrated that a variety of architectures based on ARMv8 in-order and OoO cores with LLC and HBM2 outperform 32-core Intel KNL processor. Moreover, I have shown that by using frequency scaling, energy savings of up to 59% and 61% can be achieved for ARM in-order cores and OoO cores, respectively. Lastly, it was highlighted that up to 4.5x energy savings can be achieved using many simple in-order cores, instead of fewer complex OoO cores. Thus, power hungry HPC class resources like KNL are not required for efficiently executing NGS applications, but rather simple many-core ARM in-order architectures with HBM2 can be used instead. These newly proposed ARM-based many-core architectures perform much better in terms of both performance and energy efficiency.

Furthermore, I have demonstrated the utilization of SPMs for inter-core data transfer, bypassing the cache hierarchy. Using Alexnet CNN as a case study, it was shown that using SPMs for the transferring CNN activations from one core to the next, with the cores hosting consecutive CNN layers, results in an overall performance-energy-optimized system. Since SPMs are software controlled and with limited capacity, they need to be managed efficiently. In case of CNNs, this implied using a tiled convolution enabling me to re-use the memory allocated in SPMs. Finally, using the EDP as the evaluation metric, it was shown that SPM-based architectures are 52.9% and 25.7% more efficient than cache-only systems, with LLC of 512KB and 2MB, respectively.

The work in [66] was published as a result of this chapter, demonstrating the design space exploration for genome sequence alignment application. The results showed that the use of energy efficient ARM cores along with 3D stacked HBM2 memory for genome sequence alignment, is optimal in terms of performance and energy efficiency and outperforms Intel KNL based architectures used for genome sequencing.



## 5 Conclusion and Future Work

In today's world, digitization, enabled by state-of-the-art smart systems and ultra fast connectivity, has influenced and facilitated all spheres of our everyday lives. We interact with these smart devices, throughout our day, starting from smart electronic tooth brushes, getting updated on morning weather reports through connected smart personal assistants, using smart coffee machines to prepare the morning coffee blended to our taste, and then starting off our day to work on smart autonomous or semi autonomous cars. Depending on our field of work, these smart devices enable us to be more productive at work in ways like never before, including automation, smart supply chain management, providing e-health to remotely connected patients, and the list goes on. Finally, after returning from work, digitization and connected devices are still with us, for instance, when we go for a run or other sports with our smart watches, or for entertainment to do online gaming or watch online content hosted by various video streaming services. Lastly, to end the day we go back to sleep, but some of us with various health conditions wearing connected wearable sensors for continuous health monitoring. All these smart devices and services we use throughout the day expand over a wide range of computing platforms from computing and power constrained edge sensors all the way to high performance cloud servers. As these devices evolve, they are going to run much more complex and complicated tasks and become smarter. To scale the performance while minimizing the energy consumption, both the *Power* and *Memory Walls* need to be overcome.

In this thesis, I have presented a system-level architectural simulation platform, gem5-X, to explore novel architectures for improved performance and energy efficiency across all levels of computing platforms, from the edge sensors to high end servers in the cloud.

### 5.1 Exploration is the Key

Architecture *Exploration* is the common theme and key enabler for novel performance and energy efficient architectures, that I have presented from Chapter 2 to Chapter 4 of this thesis. It is this exploration that allows to test and try out new ideas and check if they are useful in improving the

## Chapter 5. Conclusion and Future Work

---

system performance and energy efficiency.

In Chapter 2 of this thesis, I presented the gem5-X simulation platform with architecture extensions and support enhancements for architecture exploration of many-core heterogeneous compute and memory architectures. Thanks to the gem5-X Full System (FS) mode, all the extensions and support enhancements are compatible and integrated in a simulated full Linux system. This has opened possibilities to explore architectures for any application (single or multi threaded) capable of running in a Linux based environment. FS mode also enabled to have profiling support in gem5-X which helps in determining the bottlenecks within an application. All-in-all, gem5-X enables fast architectural exploration in an ever evolving and fast paced digital landscape, as it allows to try new ideas for compute and memory architectures. The main contributions in Chapter 2 are as follows:

- Gem5-X architectural extensions and support enhancements to ease the way for architecture exploration of emerging state-of-the-art applications in FS mode. The architectural extensions are for both compute and memory sub-systems.
- Gem5-X was validated against a real ARM JUNO platform with a validation error of up to 4%.
- Gem5-X architecture exploration methodology was proposed in the quest to have a performance-power optimized architecture.

Next, to demonstrate the exploration capabilities of gem5-X, I optimized and explored architectures for both compute and memory-dominated workloads, using state-of-the-art emerging applications of today as case studies.

Chapter 3 of this thesis focused on architectural exploration for compute-dominated workloads. The case study applications used were video encoding, video analytics, Virtual Machines (VMs) based cloud workloads, Binary Neural Networks (BNNs) and Recurrent Neural Networks (RNNs). Following are some of the main contributions, when exploring architectures for these compute dominated case studies:

- An in-cache computing accelerator was proposed and integrated in the L1 data (L1-D) cache of the Central Processing Unit (CPU). This inclusion provides both performance and energy gains of up to 15% and 76%, respectively, for different resolutions of the video encoding application.
- Video analytics, comprising of the video encoding kernel and Convolutional Neural Network (CNN) based image classification kernel, was optimized for energy efficiency while meeting the real-time performance constraints using the two-step gem5-X architectural exploration methodology. The resulting architecture was deeply heterogeneous and composed of ARM in-order cores cluster with in-cache computing, ARM Out-of-Order (OoO)

cores cluster and 3D stacked High Bandwidth Memory v2 (HBM2). Thus, this architecture provides performance and energy benefits up to 30% and 54%, respectively.

- For VM based cloud workloads, Near-Threshold-Computing (NTC) servers were proposed. Gem5-X VM support enhancement made it possible to simulate the VMs.
- Computational Memory (CM) cores were proposed and integrated in the execute stage of the CPU pipeline for Deep Learning (DL) workloads, including BNNs and Long-Short-Term-Memory (LSTM) based RNNs. Binary Dot-Product Engine (BDPE) based on Resistive Random-Access Memories (RRAMs) was used to accelerate BNNs. Similarly, Phase-Change Memory (PCM) based Analog In-Memory Computing (AIMC) core was used for performance and energy gains of 12.4x and 12.3x, respectively, when running LSTMs.

Finally, in Chapter 4 of this thesis, I explored and optimized architectures for memory-dominated workloads, to demonstrate the memory sub-system architectural extensions in gem5-X. Two case study applications were used. First, Next Generation Sequencing (NGS) applications were used for genome sequence alignment, which is a High Performance Computing (HPC) class memory bounded workload. Three NGS applications, representative of the NGS application domain, were used as case studies. Then, I explored architectures for CNNs, from a memory sub-system perspective, rather than the compute sub-system, as they are both compute and memory intensive workloads. In Chapter 3, the compute sub-system was explored for them. The main contributions in Chapter 4 are as follows:

- It was demonstrated that for memory bounded workload like NGS, both ARM in-order and OoO cores with 3D stacked HBM2 memory, can match or surpass the performance of Intel Knights Landing (KNL) processor, also with 3D stacked memory.
- A many-core ARM in-order cores based system was able to match the performance of a fewer ARM OoO cores system, both with 3D stacked memory, giving 4.5x energy savings.
- A shared Scratchpad Memory (SPM) architecture was proposed to transfer activations between consecutive layers of the CNN, with each layer allocated to a different physical core. This proposed system avoided the entire cache hierarchy for activations transfer from one layer to the next, giving 1.85x performance and 13% energy benefits, compared to a cache-only baseline system.

Table 5.1 summarizes all the case study applications used in this thesis and the corresponding gem5-X architectural extensions, that enabled the architectural exploration for these applications both in the compute and memory sub-systems. Moreover, the architecture exploration was facilitated by various gem5-X support enhancements, particularly the FS support, which eased the execution of running multi-threaded applications on a Linux based operating system (OS) in gem5-X. Furthermore, the enhanced checkpointing enabled faster simulation turnaround time

Table 5.1 – case study Applications vs. Gem5-X Architectural Extensions.

Applications	Gem5-X Architectural Extensions						
	In-cache Computing	ISA Extensions	Computational Memory	Core Clustering	Heterogeneous Cores	Virtual Machines	SPMs
Video Encoding	✓	✓					✓
Video Analytics	✓	✓		✓	✓		✓
VMs in the Cloud						✓	
BNNs (YoloV3)		✓	✓				
RNNs (LSTM)		✓	✓				
NGS				✓		✓	
CNNs							✓

by reaching the Region-Of-Interest (ROI) fast. Finally, the gperf profiler support in gem5-X was very helpful in identifying the bottlenecks and analysing the improvements provided by the architectural optimizations.

## 5.2 Future Work

I will now present several ideas and research topics that can be pursued as future work in the follow-up of my contributions to this thesis. In Section 5.2.1, I will present short-term future work that is continuation and consolidation of different contributions to this thesis. Furthermore, I will also present long-term future research ideas in Section 5.2.2, to extend architecture modeling and exploration capabilities of gem5-X.

### 5.2.1 Short-Term

#### 5.2.1.1 Consolidation of Architectural Extensions

Table 5.1 shows all the gem5-X architectural extensions and the applications which utilize them. All the extensions are compatible with each other, and can be consolidated into a single architectural platform. I propose a case study to demonstrate that all these extensions can be used together in a single simulated system. Video analytics co-allocated along with VM-based cloud workload, as in a realistic cloud server, can be used to demonstrate this. Video analytics is already using all the extensions, except for CM cores, VMs and SPMs. VM workload with video analytics will utilize the VM extension in gem5-X. Furthermore, the CNN kernel in video analytics can be optimized for performance using CM cores like the AIMC core along with SPMs



for inter-layer activations transfer, hence extending the work on optimizing the architecture for video analytics in Chapter 3. Therefore, such a case study will demonstrate the utilization of all the architectural extensions at the same time in gem5-X.

### 5.2.1.2 Custom Accelerators

Tightly integrated accelerators, like the BLADE in-cache computing engine and CM cores with ISA extensions, have been demonstrated and simulated using gem5-X this thesis. However, I propose to simulate custom accelerators integrated using memory mapping in gem5-X. These accelerators can be utilized with any Instruction Set Architecture (ISA) CPU core system. Off-chip accelerators or accelerators not tightly integrated with the CPU can be modelled using this approach.

### 5.2.1.3 Dynamic Voltage and Frequency Scaling (DVFS) Support in Gem5-X

Dynamic Voltage and Frequency Scaling (DVFS) is a dynamic performance-power trade-off scaling technique. It allows for the operating frequency of the compute cores to be scaled down dynamically, when they are either idle or doing non-critical tasks. In addition, DVFS also enables thermal throttling of the cores and also limits the maximum power consumption. The lowering of the operating frequency also allows for the operating voltage to be scaled down as well. This leads to reduced power consumption, as the dynamic power is quadratically related to operating voltage and linearly related to operating frequency [225]. Therefore, DVFS is a widely adopted and deployed to save energy in almost all computing platforms, either on edge sensors or cloud servers. The work in [226] proposes DVFS in the gem5 simulator, on which gem5-X is based. However, DVFS does not come out-of-the-box with gem5. Moreover, DVFS management in [226] is only tested in a limited configuration with just a single OoO core. Therefore, I propose to implement and integrate DVFS management based on the work in [226], and then extend and develop it, so that it is supported in a many-core heterogeneous clustered architecture in gem5-X with different voltage and clock domains. This will enable more realistic performance and power modeling of platforms in gem5-X, along with its architectural extensions and support enhancements. Thus, this DVFS capability will extend the architectural extensions of gem5-X presented in Chapter 2. Furthermore, it can be used to dynamically change the operating frequency for the case study workloads in Chapter 3 and Chapter 4 of this thesis.

### 5.2.1.4 Processing-in-Memory (PIM)

Processing-in-Memory (PIM) has been a promising approach in overcoming the *Memory Wall* problem. In this thesis, I presented two in-memory computing architectures: 1) BLADE, an in-cache computing engine based on SRAMs and integrated in the L1-D cache of the CPU, 2) Non-volatile Memories (NVMs) based on in-memory computing accelerators, i.e., RRAMs based BDPE and PCM based AIMC cores, integrated in the execution stage of the CPU pipeline.

## Chapter 5. Conclusion and Future Work

---

However, both of these in-memory architectures were on-chip, closely coupled to the compute cores. In both of these extensions, the input data has to be brought from off-chip DRAM memory. Near-memory processing [227] or PIM [228] is a computing paradigm where the computation takes place directly near or on the main memory, hence, avoiding the need for data movement. Consequently, both the performance and energy efficiency of the system improves.

From a technology perspective, 3D stacked memories like HBM2 enable PIM and near memory computing. The logic layer at the very bottom of the 3D stacked DRAM banks in HBM2 can be used to implement an accelerator or a small CPU core like the ARM in-order core. I propose that the HBM2 model in gem5-X can be extended to support PIM or near-memory computing with ARM in-order cores integrated with the HBM2 model. This will open new avenues of exploration in overcoming the *Memory Wall* problem. Thus, the PIM will extend the HBM2 architectural extension in gem5-X presented in Chapter 2. Moreover, it can be used to optimize architectures for memory-dominated workloads, thus extending the work presented in Chapter 4 of this thesis.

### 5.2.2 Long-Term

#### 5.2.2.1 Integrate 3D-ICE Thermal Simulator with Gem5-X

3D Interlayer Cooling Emulator (3D-ICE) enables thermal modeling of both 2D and 3D integrated circuits (ICs) [229]. 3D-ICE takes as input a netlist file for the physical description of the Integrated Circuit (IC) (2D or 3D) and a floor-plan to describe the thermal power distribution. Integrating together 3D-ICE with gem5-X will enable system architects and designers to have a complete view of new explored architectures including performance, power, energy and thermal modeling. This integration will further extended the capabilities of gem5-X that are discussed in Chapter 2 of this thesis. It can be used in thermal modeling of architectures, particularly for compute-dominated workloads and the optimized heterogeneous architectures presented in Chapter 3.

#### 5.2.2.2 RISC-V Instruction Set Architecture (ISA) Support

RISC-V is an open source emerging ISA [230, 231], which is widely anticipated to be adopted and deployed to power the computing platforms of tomorrow. The main advantage of RISC-V is that it is open source, and anyone can freely innovate or modify it according to one's requirements. Due to this flexibility of RISC-V, the work in [232] demonstrates different RISC-V cores optimized for low power consumption. Given this openness and flexibility of RISC-V, I propose to integrate RISC-V ISA into gem5-X and support it in FS mode. This will enable gem5-X to explore architectures based on RISC-V, before actually implementing them in hardware, saving both cost and time. However, supporting RISC-V in gem5-X is not a trivial task, as it requires work on RISC-V ISA implementation, Linux disk image to boot up in gem5-X, compatible kernels, valid device tree configurations, address translation unit in gem5-X according to RISC-V ISA, and then finally validating all of it against a real RISC-V hardware platform. RISC-V ISA support can

clearly extend Chapter 2 of this thesis, with new ISA capabilities added to gen5-X. Furthermore, the compute-dominated workloads in Chapter 3 can be executed on RISC-V based architectures, and the performance/energy compared against the currently proposed ARM-based architectures.

### 5.2.2.3 ISA Heterogeneity

Gem5-X supports multiple ISAs including ARM, x86, MIPS, and ALPHA. However, currently, only one ISA can be simulated at a time. ISA heterogeneity refers to support multiple ISAs in the same simulated system. This implies having complex instruction set computer (CISC) cores (e.g. x86), as well as energy efficient reduced instruction set computer (RISC) cores (e.g. ARM, MIPS) integrated and simulated in the same system. Hence, applications can be allocated according to their Quality-of-Service (QoS) requirements to a suitable core to increase energy efficiency. This is a long-term future work, as it would require work on the Linux boot strategies supporting multiple ISAs, shared memory consistency models, task allocation management, device tree configurations, clocking strategies, just to mention few of the challenges. ISA heterogeneity extends Chapter 2 of this thesis, as it is an architectural extension to the gem5-X simulation platform.

### 5.2.2.4 Graphics Processing Unit (GPU) Modelling

Graphic Processing Units (GPUs) along with CPUs have been widely deployed on different computing platforms to process a range of applications, from energy efficient mobile devices to super computers [233]. Recently, in addition to graphics processing tasks, GPUs have also been wide adopted for training of Deep Neural Networks (DNNs) [234]. Furthermore, they have gained popularity in cryptocurrency mining [235]. Given the wide adoption of GPUs in various fields and their essential role in heterogeneous computing, I propose to model GPUs in gem5-X and validate the the performance against a real GPU in the future, extending Chapter 2 of this thesis. Compute-dominated workloads are the main target for utilizing GPUs to improve performance, thus extending Chapter 3 of this thesis.

### 5.2.2.5 Parallelization of Gem5-X

Gem5-X is based on gem5 which has a single thread simulation engine, and so gem5-X is a single threaded application. To speedup the simulation turnaround time, I propose to parallelize the gem5-X simulation engine into multi-threads on a multi-core host system. This is a tedious task, as gem5 is an event triggered simulator with a single global event queue. To parallelize it, the event queue needs to be distributed among different host CPU cores, but always be in synchronization with each other. However, if this parallelization is achieved, the simulation speed can scale with the number of cores on the host systems, thus reducing the simulation turnaround time. This parallelization feature enhances the supports capabilities of gem5-X, thus extending Chapter 2 of this thesis.



# Bibliography

- [1] Marko Viitanen, Ari Koivula, Ari Lemmetti, Arttu Ylä-Outinen, Jarno Vanne, and Timo D Hämäläinen. Kvazaar: Open-Source HEVC/H.265 Encoder. In *Multimedia Conference*, pages 1179–1182, 2016.
- [2] Edouard Giacomini, Tzofnat Greenberg-Toledo, Shahar Kvatinsky, and Pierre-Emmanuel Gaillardon. A robust digital rram-based convolutional block for low-power image processing and learning applications. *IEEE Transactions on Circuits and Systems (TCAS)*, 66-I(2):643–654, 2019.
- [3] International Telecommunication Union (ITU). ICT Facts and Figures. URL: <https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf>, 2015.
- [4] International Telecommunication Union (ITU). Measuring Digital Development Facts and Figures 2019. URL: [https://www.itu.int/en/ITU-D/Statistics/Documents/facts/FactsFigures2019\\_r1.pdf](https://www.itu.int/en/ITU-D/Statistics/Documents/facts/FactsFigures2019_r1.pdf), 2019.
- [5] International Telecommunication Union (ITU). Measuring Digital Development Facts and Figures 2020. URL: <https://www.itu.int/en/ITU-D/Statistics/Documents/facts/FactsFigures2020.pdf>, 2020.
- [6] ScienceDaily. Whole genome of the Wuhan coronavirus, 2019-nCoV, sequenced. URL: [www.sciencedaily.com/releases/2020/01/200131114748.htm](http://www.sciencedaily.com/releases/2020/01/200131114748.htm), 2020.
- [7] United Nations Conference on Trade and Development (UNCTAD). How COVID-19 triggered the digital and e-commerce turning point. URL: <https://unctad.org/news/how-covid-19-triggered-digital-and-e-commerce-turning-point>, 2020.
- [8] Inc. The NPD Group. Average Weekly Netflix Usage Rose 72% in the U.S. Since COVID-19 Pandemic Began, The NPD Group Says. URL: <https://www.npd.com/wps/portal/npd/us/news/press-releases/2020/average-weekly-netflix-usage-rose-72-in-the-us-since-covid-19-pandemic-began-the-npd-group-says/>, 2020.
- [9] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

## Bibliography

---

- [10] Hamidreza Arasteh, Vahid Hosseinneshad, Vincenzo Loia, Aurelio Tommasetti, Orlando Troisi, Miadreza Shafie-khah, and Pierluigi Siano. Iot-based smart cities: A survey. In *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, pages 1–6, 2016.
- [11] Mohamed Abdel-Basset, Gunasekaran Manogaran, and Mai Mohamed. Internet of things (iot) and its impact on supply chain: A framework for building smart, secure and efficient systems. *Future Generation Computer Systems (FGCS)*, 86:614–628, 2018.
- [12] Attila Csaba Marosi, Róbert Lovas, Ádám Kisari, and Ernő Simonyi. A novel iot platform for the era of connected cars. In *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*, pages 1–11, 2018.
- [13] Moeen Hassanaliheragh, Alex Page, Tolga Soyata, Gaurav Sharma, Mehmet Aktas, Gonzalo Mateos, Burak Kantarci, and Silvana Andreescu. Health monitoring and management using internet-of-things (iot) sensing with cloud-based processing: Opportunities and challenges. In *2015 IEEE International Conference on Services Computing (SCC 2015)*, pages 285–292, 2015.
- [14] Ravi Kishore Kodali and Snehashish Mandal. Iot based weather station. In *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pages 680–683, 2016.
- [15] IEA. Data centres and data transmission networks - Analysis. URL: <https://www.iea.org/reports/data-centres-and-data-transmission-networks>, 2019.
- [16] Nicola Jones. The information factories. *Nature*, 561(7722):163–6, 2018.
- [17] Robert R Schaller. Moore’s law: past, present and future. *IEEE Spectrum*, 34(6):52–59, 1997.
- [18] Muhammad Shafique, Siddharth Garg, Tulika Mitra, Sri Parameswaran, and Jörg Henkel. Dark silicon as a challenge for hardware/software co-design: Invited special session paper. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis, CODES ’14*, New York, NY, USA, 2014. Association for Computing Machinery.
- [19] Hadi Esmaeilzadeh, Emily Blem, Renée St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. In *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, pages 365–376, 2011.
- [20] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki. Toward Dark Silicon in Servers. *IEEE Micro*, 31(4):6–15, July 2011.
- [21] Michael B Taylor. Is Dark Silicon Useful? Harnessing the Four Horsemen of the Coming Dark Silicon Apocalypse. In *DAC Design Automation Conference 2012*, pages 1131–1136, June 2012.

- 
- [22] Muhammad Shafique and Siddharth Garg. Computing in the dark silicon era: Current trends and research challenges. *IEEE Design & Test (D&T)*, 34(2):8–23, 2017.
- [23] Eric S. Chung, Peter A. Milder, James C. Hoe, and Ken Mai. Single-chip heterogeneous computing: Does the future include custom logic, fpgas, and gpgpus? In *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 225–236, 2010.
- [24] ARM. big.LITTLE Technology Moves Towards Fully Heterogeneous Global Task Scheduling, 2013.
- [25] Ali Pahlevan, Javier Picorel, Arash Pourhabibi Zarandi, Davide Rossi, Marina Zapater, Andrea Bartolini, Pablo G Del Valle, David Atienza, Luca Benini, and Babak Falsafi. Towards near-threshold server processors. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 7–12. IEEE, 2016.
- [26] Wm. A. Wulf and Sally A. McKee. Hitting the Memory Wall: Implications of the Obvious. *ACM SIGARCH Computer Architecture News*, 23(1):20–24, March 1995.
- [27] Mitesh R Meswani, Sergey Blagodurov, David Roberts, John Slice, Mike Ignatowski, and Gabriel H Loh. Heterogeneous Memory Architectures: A HW/SW Approach for Mixing Die-Stacked and Off-Package Memories. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 126–136, Feb 2015.
- [28] Donghyuk Lee, Saugata Ghose, Gennady Pekhimenko, Samira Khan, and Onur Mutlu. Simultaneous multi-layer access: Improving 3D-stacked memory bandwidth at low cost. *ACM Transactions on Architecture and Code Optimization (TACO)*, 12(4):1–29, 2016.
- [29] Yuan Xie. Modeling, Architecture, and Applications for Emerging Memory Technologies. *IEEE Design & Test of computers (D & T)*, 28(1):44–51, January 2011.
- [30] Bin Gao. Emerging non-volatile memories for computation-in-memory. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 381–384, 2020.
- [31] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, August 2011.
- [32] Dong Uk Lee, Kyung Whan Kim, Kwan Weon Kim, Hongjung Kim, Ju Young Kim, Young Jun Park, Jae Hwan Kim, Dae Suk Kim, Heat Bit Park, Jin Wook Shin, Jang Hwan Cho, Ki Hun Kwon, Min Jeong Kim, Jaejin Lee, Kun Woo Park, Byongtae Chung, and Sungjoo Hong. 25.2 A 1.2V 8Gb 8-channel 128GB/s high-bandwidth memory (HBM) stacked DRAM with effective microbump I/O test methods using 29nm process and TSV. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 432–433, 2014.

## Bibliography

---

- [33] Google. gperftools. <https://github.com/gperftools/gperftools>, 2011.
- [34] ARM. ARM Versatile Express Juno r2 Development Platform, 2015.
- [35] SANDVINE. Global internet phenomena report. 2018. URL: <https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf>, 2018.
- [36] Nicholas Nethercote and Julian Seward. Valgrind: A framework for heavyweight dynamic binary instrumentation. In *Proceedings of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 89–100, 2007.
- [37] William Simon, Yasir Mahmood Qureshi, Alexandre Levisse, Marina Zapater, and David Atienza. BLADE: A BitLine Accelerator for Devices on the Edge. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI (GLVLSI)*, pages 1–6, 2019.
- [38] William Andrew Simon, Yasir Mahmood Qureshi, Marco Rios, Alexandre Levisse, Marina Zapater, and David Atienza. BLADE: An in-Cache Computing Architecture for Edge Devices. *IEEE Transactions on Computers (TC)*, 69(9):1349–1363, 2020.
- [39] ARM. ARM Architecture Reference Manual ARMv8, 2017.
- [40] Ganesh Ananthanarayanan, Paramvir Bahl, Peter Bodík, Krishna Chintalapudi, Matthai Philipose, Lenin Ravindranath, and Sudipta Sinha. Real-time video analytics: The killer app for edge computing. *Computer*, 50(10):58–67, 2017.
- [41] Anup Mohan, Ahmed S. Kaseb, Kent W. Gauen, Yung-Hsiang Lu, Amy R. Reibman, and Thomas J. Hacker. Determining the necessary frame rate of video data for object tracking under accuracy constraints. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, April 2018.
- [42] Anupam Sobti, Chetan Arora, and M Balakrishnan. Object detection in real-time systems: Going beyond precision. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1020–1028, 2018.
- [43] Viacheslav Moskalenko, Alona Moskalenko, Artem Korobov, and Viktor Semashko. The model and training algorithm of compact drone autonomous visual navigation system. *Data*, 4(1):4, December 2018.
- [44] Gowdham Prabhakar, Binsu Kailath, Sudha Natarajan, and Rajesh Kumar. Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving. In *2017 IEEE Region 10 Symposium (TENSYP)*, pages 1–6, 2017.
- [45] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *Computing Research Repository (CoRR)*, abs/1704.04861, 2017.



- 
- [46] Ali Pahlevan, Yasir Mahmood Qureshi, Marina Zapater, Andrea Bartolini, Davide Rossi, Luca Benini, and David Atienza. Energy proportionality in near-threshold computing servers and cloud data centers: Consolidating or not? In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 147–152, 2018.
- [47] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV (4)*, volume 9908 of *Lecture Notes in Computer Science*, pages 525–542. Springer, 2016.
- [48] Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *Computing Research Repository (CoRR)*, abs/1602.02830, 2016.
- [49] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 779–788. IEEE Computer Society, 2016.
- [50] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [51] Abu Sebastian, Manuel Le Gallo, Geoffrey W Burr, Sangbum Kim, Matthew BrightSky, and Evangelos Eleftheriou. Tutorial: Brain-inspired computing using phase-change memory devices. *Journal of Applied Physics*, 124(11):111101, 2018.
- [52] Bertil Schmidt and Andreas Hildebrandt. Next-generation sequencing: big data meets high performance computing. *Drug Discovery Today*, pages 712 – 717, 2017.
- [53] Jennifer Ronholm. Editorial: Game changer - next generation sequencing and its impact on food microbiology. *Frontiers in Microbiology*, page 363, 2018.
- [54] Jun Zhang, Rod Chiodini, Ahmed Badr, and Genfa Zhang. The impact of next-generation sequencing on genomics. *Journal of Genetics and Genomics (JGG)*, 38(3):95–109, 2011.
- [55] Heng Li and Nils Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5):473–483, 05 2010.
- [56] Jose M Herruzo, Sonia González-Navarro, Pablo Ibanez-Marin, Victor Vinals-Yufera, Jesus Alastruey-Benede, and Oscar Plata. Accelerating sequence alignments based on fm-index using the intel KNL processor. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 17(4):1093–1104, 2018.
- [57] Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 390–398. IEEE, 2000.
- [58] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature Methods*, pages 357–359, Mar 2012.

## Bibliography

---

- [59] Heng Li and Richard Durbin. Fast and accurate long-read alignment with burrows–wheeler transform. *Bioinformatics*, pages 589–595, January 2010.
- [60] Daehwan Kim, Joseph M Paggi, Chanhee Park, Christopher Bennett, and Steven L Salzberg. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nature Biotechnology*, 37(8):907–915, August 2019.
- [61] Avinash Sodani, Roger Gramunt, Jesus Corbal, Ho-Seop Kim, Krishna Vinod, Sundaram Chinthamani, Steven Hutsell, Rajat Agarwal, and Yen-Chen Liu. Knights landing: Second-generation intel xeon phi product. *IEEE Micro*, pages 34–46, Mar 2016.
- [62] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.
- [63] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of CNN and RNN for natural language processing. *Computing Research Repository (CoRR)*, abs/1702.01923, 2017.
- [64] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. Deep Learning Recommendation Model for Personalization and Recommendation Systems. *arXiv preprint arXiv:1906.00091*, 2019.
- [65] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. Deep learning in bioinformatics. *Briefings in Bioinformatics*, 18(5):851–869, 07 2016.
- [66] Yasir Mahmood Qureshi, Jose Manuel Herruzo, Marina Zapater, Katzalin Olcoz, Sonia Gonzalez Navarro, Oscar Plata, and David Atienza. Genome sequence alignment - design space exploration for optimal performance and energy architectures. *IEEE Transactions on Computers*, pages 1–1, 2020.
- [67] Wen-mei Hwu, Kurt Keutzer, and Timothy G. Mattson. The concurrency challenge. *IEEE Design & Test of Computers (D & T)*, pages 312–320, 2008.
- [68] Heejun Shim, Sang-Heon Lee, Yun-Sik Woo, Moo-Kyoung Chung, Jae-Gon Lee, and Chong-Min Kyung. Cycle-accurate verification of ahb-based rtl ip with transaction-level system environment. In *2006 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pages 1–4, April 2006.
- [69] Cagkan Erbas, Andy D. Pimentel, Mark Thompson, and Simon Polstra. A framework for system-level modeling and simulation of embedded systems architectures. *EURASIP Journal on Embedded Systems*, 2007:1–11, 2007.

- 
- [70] Roberto Varona-Gomez and Eugenio Villar. AADL Simulation and Performance Analysis in SystemC. In *2009 14th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 323–328, 2009.
- [71] Trevor E. Carlson, Wim Heirman, and Lieven Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2011.
- [72] Peter S Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, Gustav Hallberg, Johan Hogberg, Fredrik Larsson, Andreas Moestedt, and Bengt Werner. Simics: A full system simulation platform. *Computer*, pages 50–58, 2002.
- [73] Nikolaos Hardavellas, Stephen Somogyi, Thomas F. Wenisch, Roland E. Wunderlich, Shelley Chen, Jangwoo Kim, Babak Falsafi, James C. Hoe, and Andreas G. Nowatzky. Simflex: A fast, accurate, flexible full-system simulation framework for performance evaluation of server architecture. *ACM SIGMETRICS Performance Evaluation Review*, pages 31–34, 2004.
- [74] Supreet Jeloka, Naveen Bharathwaj Akesh, Dennis Sylvester, and David Blaauw. A 28 nm Configurable Memory (TCAM/BCAM/SRAM) Using Push-Rule 6T Bit Cell Enabling Logic-in-Memory. *IEEE Journal of Solid-State Circuits (JSSC)*, pages 1009–1021, 2016.
- [75] Charles Eckert, Xiaowei Wang, Jingcheng Wang, Arun Subramaniyan, Ravi Iyer, Dennis Sylvester, David Blaauw, and Reetuparna Das. Neural cache: Bit-serial in-cache acceleration of deep neural networks. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 383–396, 2018.
- [76] John D. McCalpin. Memory bandwidth and machine balance in current high performance computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pages 19–25, December 1995.
- [77] William Simon, Juan Galicia, Alexandre Levisse, Marina Zapater, and David Atienza. A fast, reliable and wide-voltage-range in-memory computing architecture. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.
- [78] 96Boards. Hikey960. URL: <https://www.96boards.org/product/hikey960/>, 2018.
- [79] Kyomin Sohn, Won-Joo Yun, Reum Oh, Chi-Sung Oh, Seong-Young Seo, Min-Sang Park, Dong-Hak Shin, Won-Chang Jung, Sang-Hoon Shin, Je-Min Ryu, Hye-Seung Yu, Jae-Hun Jung, Hyunui Lee, Seok-Yong Kang, Young-Soo Sohn, Jung-Hwan Choi, Yong-Cheol Bae, Seong-Jin Jang, and Gyoyoung Jin. A 1.2 V 20 nm 307 GB/s HBM DRAM With At-Speed Wafer-Level IO Test Scheme and Adaptive Refresh Considering Temperature Distribution. *IEEE Journal of Solid-State Circuits (JSSC)*, 52(1):250–260, 2017.
- [80] Niladrish Chatterjee, Mike O’Connor, Donghyuk Lee, Daniel R Johnson, Stephen W Keckler, Minsoo Rhu, and William J Dally. Architecting an Energy-Efficient DRAM

## Bibliography

---

- System for GPUs. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 73–84, Feb 2017.
- [81] QEMU. QEMU the FAST! processor emulator. URL: <https://www.qemu.org>, 2019.
- [82] Bell Labs. Plan 9 from bell labs. <https://9p.io/plan9/about.html>, 2003. Accessed Sep. 01, 2018.
- [83] OSDev. Virtio. <https://wiki.osdev.org/Virtio>, 2016. Accessed Aug. 28, 2018.
- [84] Jag Bolaria. Cortex-A57 extends ARM’s Reach High-End 64-bit CPU Strives for Servers. *Microprocessor Report*, 2012.
- [85] Kevin Krewell. Cortex-A53 is ARM’s Next Little Thing New CPU Core Brings 64 Bits to Big.Little, Mobile. *Microprocessor Report*, 11 2012.
- [86] Intel. Intel Atom x5-Z8350 Processor. <https://ark.intel.com/content/www/us/en/ark/products/93361/intel-atom-x5-z8350-processor-2m-cache-up-to-1-92-ghz.html>, 2016. Accessed Jun. 13, 2021.
- [87] Intel. Intel core i7-4790 processor. <https://ark.intel.com/content/www/us/en/ark/products/80806/intel-core-i7-4790-processor-8m-cache-up-to-4-00-ghz.html>, 2017. Accessed Jun. 13, 2021.
- [88] Intel. Intel xeon processor e5-1620. <https://ark.intel.com/content/www/us/en/ark/products/64621/intel-xeon-processor-e5-1620-10m-cache-3-60-ghz-0-0-gt-s-intel-qpi.html>, 2015. Accessed Jun. 13, 2021.
- [89] Basireddy Karunakar Reddy, Matthew J. Walker, Domenico Balsamo, Stephan Diestelhorst, Bashir M. Al-Hashimi, and Geoff V Merrett. Empirical cpu power modelling and estimation in the gem5 simulator. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–8, 2017.
- [90] Anastasiia Butko, Florent Bruguier, Abdoulaye Gamatié, Gilles Sassatelli, David Novo, Lionel Torres, and Michel Robert. Full-system simulation of big.little multicore architecture for performance and energy exploration. In *2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC)*, pages 201–208, 2016.
- [91] Andrei Frumusanu and Ryan Smith. Cortex a53 - performance and power. <https://www.anandtech.com/show/8718/the-samsung-galaxy-note-4-exynos-review/4>, 2015. Accessed Sep. 01, 2018.
- [92] Andrei Frumusanu and Ryan Smith. Cortex a57 - performance and power. <https://www.anandtech.com/show/8718/the-samsung-galaxy-note-4-exynos-review/6>, 2015. Accessed Sep. 01, 2018.

- 
- [93] Sukhan Lee, Hyunyeon Cho, Young Hoon Son, Yuhwan Ro, Nam Sung Kim, and Jung Ho Ahn. Leveraging power-performance relationship of energy-efficient modern DRAM devices. *IEEE Access*, pages 31387–31398, 2018.
- [94] Mike O’Connor, Niladrish Chatterjee, Donghyuk Lee, John Wilson, Aditya Agrawal, Stephen W. Keckler, and William J. Dally. Fine-grained dram: Energy-efficient DRAM for extreme bandwidth systems. In *MICRO*, pages 41–54, 2017.
- [95] Yasir Mahmood Qureshi. Gem5-X: A gem5-based simulator with architectural eXtensions. <https://esl.epfl.ch/gem5-x>, 2020. Accessed Jun. 13, 2021.
- [96] Yasir Qureshi, William Simon, Marina Zapater, Katzalin Olcoz, and David Atienza. Gem5-X Full System Manual. [https://eslweb.epfl.ch/masters/img/20200814gem5\\_X\\_TechnicalManual\\_v1.pdf](https://eslweb.epfl.ch/masters/img/20200814gem5_X_TechnicalManual_v1.pdf), 2020. Accessed Jun. 13, 2021.
- [97] Yasir Mahmood Qureshi, William Andrew Simon, Marina Zapater, David Atienza, and Katzalin Olcoz. GEM5-X: A GEM5-Based System Level Simulation Framework to Optimize Many-Core Platforms. In *Proceedings of the High Performance Computing Symposium, HPC ’19*, San Diego, CA, USA, 2019. Society for Computer Simulation International.
- [98] Yasir Mahmood Qureshi, William Andrew Simon, Marina Zapater Sancho, Katzalin Olcoz, and David Atienza Alonso. Gem5-x : A many-core heterogeneous simulation platform for architectural exploration and optimization. *ACM Transactions on Architecture and Code Optimization (TACO)*, 2021.
- [99] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 22(12):1649–1668, 2012.
- [100] Benjamin Bross. High efficiency video coding (HEVC) text specification draft 9 (SoDIS). In *11th JCT-VC meeting*, 2012.
- [101] Ari Koivula, Marko Viitanen, Jarno Vanne, Timo D. Härmäläinen, and Laurent Fasnacht. Parallelization of kvazaar hevc intra encoder for multi-core processors. In *2015 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 1–6, 2015.
- [102] Panu Sjövall, Janne Virtanen, Jarno Vanne, and Timo D. Härmäläinen. High-Level Synthesis Design Flow for HEVC Intra Encoder on SoC-FPGA. In *2015 Euromicro Conference on Digital System Design (DSD)*, pages 49–56, 2015.
- [103] Mohamed Maazouz, Nejmeddine Bahri, Noureddine Batel, Abdelmoughni Toubal, and Nouri Masmoudi. Parallel implementation of kvazaar hevc on multicore arm processor. In *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*, pages 1086–1091, 2016.

## Bibliography

---

- [104] Johan Barthélemy, Nicolas Verstaëvel, Hugh Forehead, and Pascal Perez. Edge-computing video analytics for real-time traffic monitoring in a smart city. *Sensors*, 19(9), 2019.
- [105] Puren Guler, Deniz Emeksiz, Alptekin Temizel, Mustafa Teke, and Tugba Taskaya Temizel. Real-time multi-camera video analytics system on GPU. *Journal of Real-Time Image Processing (JRTIP)*, 11(3):457–472, March 2013.
- [106] Hardik Sharma, Jongse Park, Balavinayagam Samynathan, Behnam Robatmili, Shahrzad Mirkhani, and Hadi Esmaeilzadeh. From tensors to fpgas: Accelerating deep learning. In *Hot Chips: Symposium on High Performance Chips*. IEEE/ACM, 2018.
- [107] Shang Wang, Chen Zhang, Yuanchao Shu, and Yunxin Liu. Live video analytics with fpga-based smart cameras. In *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*, pages 9–14, 2019.
- [108] Christina Delimitrou and Christos Kozyrakis. Optimizing resource provisioning in shared cloud systems. Technical report, Stanford University, 2014.
- [109] Ulya R. Karpuzcu, Abhishek Sinkar, Nam Sung Kim, and Josep Torrellas. Energysmart: Toward energy-efficient manycores for near-threshold computing. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 542–553, 2013.
- [110] David Jacquet, Frédéric Hasbani, Philippe Flatresse, Robin Wilson, Franck Arnaud, Giorgio Cesana, Thierry Di Gilio, Christophe Lecocq, Tanmoy Roy, Amit Chhabra, Chiranjeev Grover, Olivier Minez, Jacky Uginet, Guy Durieu, Cyril Adobati, Davide Casalotto, Frederic Nyer, Patrick Menut, Andreia Cathelin, Indavong Vongsavady, and Philippe Magarshack. A 3 GHz Dual Core Processor ARM Cortex TM -A9 in 28 nm UTBB FD-SOI CMOS With Ultra-Wide Voltage Range and Energy Efficiency Optimization. *IEEE Journal of Solid-State Circuits (JSSC)*, 49(4):812–826, 2014.
- [111] ThunderX Rattles Server Market. Thunderx rattles server market. *Microprocessor Report*, 29(6):1–4, 2014.
- [112] Jason Cong and Bingjun Xiao. Minimizing computation in convolutional neural networks. In *International conference on artificial neural networks (ICANN)*, volume 8681 of *Lecture Notes in Computer Science*, pages 281–290. Springer, 2014.
- [113] Weijia Chen, Hui Wu, Shaojun Wei, Anping He, and Hong Chen. An Asynchronous Energy-Efficient CNN Accelerator with Reconfigurable Architecture. In *2018 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pages 51–54. IEEE, 2018.
- [114] Baohua Sun, Lin Yang, Patrick Dong, Wenhan Zhang, Jason Dong, and Charles Young. Ultra power-efficient CNN domain specific accelerator with 9.3tops/watt for mobile and embedded applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1677–1685. IEEE Computer Society, 2018.

- 
- [115] Manqing Mao, Xiaoyu Sun, Xiaochen Peng, Shimeng Yu, and Chaitali Chakrabarti. A versatile reram-based accelerator for convolutional neural networks. In *2018 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 211–216. IEEE, 2018.
- [116] Kaiyuan Guo, Jincheng Yu, Xuefei Ning, Yiming Hu, Yu Wang, and Huazhong Yang. RRAM based buffer design for energy efficient CNN accelerator. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 435–440. IEEE Computer Society, 2018.
- [117] Alok Prakash, Siew-Kei Lam, Thambipillai Srikanthan, and Christopher T Clarke. Modelling communication overhead for accessing local memories in hardware accelerators. In *2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, pages 31–34. IEEE Computer Society, 2013.
- [118] Intel. Bfloat16 – hardware numerics definition. <https://software.intel.com/sites/default/files/managed/40/8b/bf16-hardware-numerics-definition-white-paper.pdf>, 2018. Accessed Jun. 13, 2021.
- [119] Yiran Shen, Tao Han, Qing Yang, Xu Yang, Yong Wang, Feng Li, and Hongkai Wen. Cs-cnn: Enabling robust and efficient convolutional neural networks inference for internet-of-things applications. *IEEE Access*, 6:13439–13448, 2018.
- [120] SR Nandakumar, Manuel Le Gallo, Christophe Piveteau, Vinay Joshi, Giovanni Mariani, Irem Boybat, Geethan Karunaratne, Riduan Khaddam-Aljameh, Urs Egger, Anastasios Petropoulos, et al. Mixed-precision deep learning based on computational memory. *Frontiers in Neuroscience*, 14:406, 2020.
- [121] Vinay Joshi, Manuel Le Gallo, Simon Haefeli, Irem Boybat, Sasidharan Rajalekshmi Nandakumar, Christophe Piveteau, Martino Dazzi, Bipin Rajendran, Abu Sebastian, and Evangelos Eleftheriou. Accurate deep neural network inference using computational phase-change memory. *Nature Communications*, 11(1):1–13, 2020.
- [122] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, 44(3):14–26, 2016.
- [123] Martino Dazzi, Abu Sebastian, Pier Andrea Francese, Thomas Parnell, Luca Benini, and Evangelos Eleftheriou. 5 parallel prism: a topology for pipelined implementations of convolutional neural networks using computational memory. In *Proceedings 33rd Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019.
- [124] Ben Feinberg, Uday Kumar Reddy Vengalam, Nathan Whitehair, Shibo Wang, and Engin Ipek. Enabling scientific computing on memristive accelerators. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018.

## Bibliography

---

- [125] Aayush Ankit, Izzat El Hajj, Sai Rahul Chalamalasetti, Geoffrey Ndu, Martin Foltin, R. Stanley Williams, Paolo Faraboschi, Wen-mei W Hwu, John Paul Strachan, Kaushik Roy, and Dejan S. Milojicic. PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'19)*, 2019.
- [126] Abu Sebastian, Manuel Le Gallo, Riduan Khaddam-Aljameh, and Evangelos Eleftheriou. Memory devices and applications for in-memory computing. *Nature Nanotechnology*, 2020.
- [127] Venu Gopal Reddy. Neon technology introduction. *ARM Corporation*, 4(1), 2008.
- [128] Shaizeen Aga, Supreet Jeloka, Arun Subramaniyan, Satish Narayanasamy, David Blaauw, and Reetuparna Das. Compute caches. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 481–492, 2017.
- [129] Wikichip. Cortex-a53 - microarchitectures - arm. [https://en.wikichip.org/wiki/arm\\_holdings/microarchitectures/cortex-a53](https://en.wikichip.org/wiki/arm_holdings/microarchitectures/cortex-a53), 2016. Accessed Jan. 7, 2019.
- [130] ARM. Arm compute library framework. <https://developer.arm.com/ip-products/processors/machine-learning/compute-library>, 2018. Accessed Jun. 13, 2021.
- [131] Elia Kaufmann, Antonio Loquercio, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: Learning agile flight in dynamic environments. *Computing Research Repository (CoRR)*, abs/1806.08548, 2018.
- [132] Jeng-Shyang Pan, Sen Ma, Shi-Huang Chen, and Chun-Sheng Yang. Vision-based vehicle forward collision warning system using optical flow algorithm. *Journal of Information Hiding and Multimedia Signal Processing (JIHMSP)*, 6:1029–1042, 07 2015.
- [133] Yuyang Liu, Ce Zhu, Min Mao, Fangliang Song, Frederic Dufaux, and Xiang Zhang. Video analytical coding: When video coding meets video analysis. *Signal Processing: Image Communication*, 67:48 – 57, 2018.
- [134] Xukan Ran, Haolanz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 1421–1429, 2018.
- [135] Tarek Elgamal, Shu Shi, Varun Gupta, Rittwik Jana, and Klara Nahrstedt. Sieve: Semantically encoded video analytics on edge and cloud. *arXiv preprint arXiv:2006.01318*, 2020.
- [136] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.
- [137] Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277, 2018.



- 
- [138] Dong-Hyun Kim, Yong-Guk Go, and Soo-Mi Choi. First-person-view drone flying in mixed reality. In *SIGGRAPH Asia 2018 Posters*, SA '18, New York, NY, USA, 2018. ACM.
- [139] Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. Bandwidth-efficient live video analytics for drones via edge computing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 159–173, 2018.
- [140] Yeong-Kang Lai, Chu-Ying Ho, Yu-Hau Huang, Chuan-Wei Huang, Yi-Xian Kuo, and Yu-Chieh Chung. Intelligent vehicle collision-avoidance system with deep learning. In *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 123–126, 2018.
- [141] Heehoon Kim, Hyoungwook Nam, Wookeun Jung, and Jaejin Lee. Performance analysis of cnn frameworks for gpus. In *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 55–64, 2017.
- [142] Leyuan Wang, Zhi Chen, Yizhi Liu, Yao Wang, Lianmin Zheng, Mu Li, and Yida Wang. A unified optimization approach for cnn model inference on integrated gpus. In *Proceedings of the 48th International Conference on Parallel Processing*, ICPP 2019, New York, NY, USA, 2019. ACM.
- [143] Nvidia. Nvidia jetson nano. <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/>, 2019. Accessed Jan. 15, 2021.
- [144] Nvidia. DATA SHEET NVIDIA Jetson Nano System-on-Module. <https://developer.nvidia.com/embedded/dlc/jetson-nano-system-module-datasheet>, 2019. Accessed Jan. 15, 2021.
- [145] ARM. Mali GPUs for Graphics Processing. <https://www.arm.com/products/silicon-ip-multimedia>, 2021. Accessed Jan. 21, 2021.
- [146] ARM. Arm Ethos-N series processors. <https://developer.arm.com/ip-products/processors/machine-learning/arm-ethos-n>, 2021. Accessed Jan. 21, 2021.
- [147] Nvidia. Nvidia tensorrt. <https://github.com/NVIDIA/TensorRT>, 2019. Accessed Jan. 15, 2021.
- [148] Sunpyo Hong and Hyesoon Kim. An integrated gpu power and performance model. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA)*, pages 280–289, 2010.
- [149] Peter Clarke. Globalfoundries preps 12nm fdsoi process. [http://www.eetimes.com/document.asp?doc\\_id=1330423](http://www.eetimes.com/document.asp?doc_id=1330423), 2016. Accessed Sep. 10, 2016.
- [150] Linley Gwennap. Fd-soi offers alternative to finfet. <https://www.globalfoundries.com/sites/default/files/fd-soi-offers-alternative-to-finfet.pdf>, 2016. Accessed Sep. 10, 2016.

## Bibliography

---

- [151] John Wilkes. More google cluster data, 2011.
- [152] Micron. 4Gb: x4, x8, x16 DDR4 SDRAM features. [https://www.micron.com/-/media/client/global/documents/products/data-sheet/dram/ddr4/4gb\\_ddr4\\_dram.pdf](https://www.micron.com/-/media/client/global/documents/products/data-sheet/dram/ddr4/4gb_ddr4_dram.pdf), 2020. Accessed Jun. 13, 2021.
- [153] Daniele Bortolotti, Simone Tinti, Piero Altoé, and Andrea Bartolini. User-space apis for dynamic power management in many-core armv8 computing nodes. In *2016 International Conference on High Performance Computing & Simulation (HPCS)*, pages 675–681. IEEE, 2016.
- [154] Davide Rossi, Antonio Pullini, Igor Loi, Michael Gautschi, Frank Kağan Gürkaynak, Adam Teman, Jeremy Constantin, Andreas Burg, Ivan Miro-Panades, Edith Beigné, Fabien Clermidy, Philippe Flatresse, and Luca Benini. Energy-efficient near-threshold parallel computing: The pulpv2 cluster. *IEEE Micro*, 37(5):20–31, September 2017.
- [155] Umar Ahsan and Abdul Bais. A review on big data analysis and internet of things. In *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 325–330. IEEE Computer Society, 2016.
- [156] Mahmoud Ammar, Giovanni Russello, and Bruno Crispo. Internet of things: A survey on the security of iot frameworks. *Journal of Information Security and Applications (JISA)*, 38:8–27, 2018.
- [157] Haitong Li, Tony F Wu, Subhasish Mitra, and H-S Philip Wong. Resistive ram-centric computing: Design and modeling methodology. *IEEE Transactions on Circuits and Systems (TCAS)*, 64-I(9):2263–2273, 2017.
- [158] Joris Guérin, Olivier Gibaru, Stéphane Thiery, and Eric Nyiri. CNN features are also great at unsupervised classification. *Computing Research Repository (CoRR)*, abs/1707.01700, 2017.
- [159] Uzi Chester and Joel Ratsaby. Machine learning for image classification and clustering using a universal distance measure. In *International Conference on Similarity Search and Applications (SISAP)*, volume 8199 of *Lecture Notes in Computer Science*, pages 59–72. Springer, 2013.
- [160] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *Computing Research Repository (CoRR)*, abs/1807.05511, 2018.
- [161] Eshed Ohn-Bar and Mohan M. Trivedi. To boost or not to boost? on the limits of boosted trees for object detection. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 3350–3355. IEEE, 2016.
- [162] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(4):743–761, 2012.

- 
- [163] Kah Kay Sung and Tomaso A. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 20(1):39–51, 2002.
- [164] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *Computing Research Repository (CoRR)*, abs/1804.02767, 2018.
- [165] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Neural Information Processing Systems (NIPS) Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.
- [166] Joseph Redmon. Darknet: Open source neural networks in c. <http://pjreddie.com/darknet/>, 2013–2016. Accessed Jun. 13, 2021.
- [167] Saransh Gupta, Mohsen Imani, Harveen Kaur, and Tajana Simunic Rosing. NNPI: A Processing In-Memory Architecture for Neural Network Acceleration. *IEEE Transactions on Computers (TC)*, 2019.
- [168] Onur Mutlu, Saugata Ghose, Juan Gómez-Luna, and Rachata Ausavarungnirun. Processing data where it makes sense: Enabling in-memory computation. *Microprocessors and Microsystems*, 2019.
- [169] H. Tsai, S. Ambrogio, C. Mackin, P. Narayanan, R. M. Shelby, K. Rocki, A. Chen, and G. W. Burr. Inference of long-short term memory networks at software-equivalent accuracy using 2.5 M analog phase change memory devices. In *2019 Symposium on VLSI Technology*, 2019.
- [170] Manuel Le Gallo, Abu Sebastian, Roland Mathis, Matteo Manica, Heiner Giefers, Tomas Tuma, Costas Bekas, Alessandro Curioni, and Evangelos Eleftheriou. Mixed-precision in-memory computing. *Nature Electronics*, 2018.
- [171] I. Giannopoulos, A. Sebastian, M. Le Gallo, V.P. Jonnalagadda, M. Sousa, M.N. Boon, and E. Eleftheriou. 8-bit precision in-memory multiplication with projected phase-change memory. In *2018 IEEE International Electron Devices Meeting (IEDM)*, 2018.
- [172] S. Kapur. Low precision RNNs: Quantizing RNNs without losing accuracy. *arXiv preprint arXiv:1710.07706*, 2017.
- [173] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [174] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 1993.

## Bibliography

---

- [175] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.
- [176] João Vieira, Edouard Giacomin, Yasir Qureshi, Marina Zapater, Xifan Tang, Shahar Kvatinisky, David Atienza, and Pierre-Emmanuel Gaillardon. A product engine for energy-efficient execution of binary neural networks using resistive memories. In *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 160–165, 2019.
- [177] Milan Radulovic, Kazi Asifuzzaman, Darko Zivanovic, Nikola Rajovic, Guillaume Colin de Verdière, Dirk Pleiter, Manolis Marazakis, Nikolaos Kallimanis, Paul Carpenter, Petar Radojković, and Eduard Ayguadé. Mainstream vs. emerging HPC: Metrics, trade-offs and lessons learned. In *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 250–257, 2018.
- [178] Nikola Rajovic, Paul M Carpenter, Isaac Gelado, Nikola Puzovic, Alex Ramirez, and Mateo Valero. Supercomputing with commodity cpus: Are mobile socs ready for HPC? In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2013.
- [179] Zhonghong Ou, Bo Pang, Yang Deng, Jukka K Nurminen, Antti Ylä-Jääski, and Pan Hui. Energy- and cost-efficiency analysis of ARM-based clusters. In *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 115–123, 2012.
- [180] Xinyuan Li, Lin Xu, and Jian Zhang. Improving the thread scalability and parallelism of bwa-mem on intel hpc platforms. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1858–1865, 2019.
- [181] Ben Langmead, Christopher Wilks, Valentin Antonescu, and Rone Charles. Scaling read aligners to hundreds of threads on general-purpose processors. *Bioinformatics*, 35(3):421–432, 2019.
- [182] Richard Wilton, Tamas Budavari, Ben Langmead, Sarah J Wheelan, Steven L Salzberg, and Alexander S Szalay. Arioc: high-throughput read alignment with GPU-accelerated exploration of the seed-and-extend search space. *PeerJ*, page e808, March 2015.
- [183] Ruibang Luo, Thomas Wong, Jianqiao Zhu, Chi-Man Liu, Xiaoqian Zhu, Edward Wu, Lap-Kei Lee, Haoxiang Lin, Wenjuan Zhu, David W. Cheung, Hing-Fung Ting, Siu-Ming Yiu, Shaoliang Peng, Chang Yu, Yingrui Li, Ruiqiang Li, and Tak-Wah Lam. SOAP3-dp: Fast, accurate and sensitive GPU-based short read aligner. *PLoS ONE*, 8(5):e65632, May 2013.

- 
- [184] Michael J Anderson, Bryan Catanzaro, Jike Chong, Ekaterina Gonina, Kurt Keutzer, Chao-Yue Lai, Mark Murphy, David Sheffield, Bor-Yiing Su, and Narayanan Sundaram. Considerations when evaluating microprocessor platforms. In *3rd USENIX Workshop on Hot Topics in Parallelism (HotPar '11)*, page 1, USA, 2011.
- [185] Ali Khajeh-Saeed, Stephen Poole, and J Blair Perot. Acceleration of the smith–waterman algorithm using single and multiple graphics processors. *Journal of Computational Physics*, 229(11):4247–4258, June 2010.
- [186] Daichi Fujiki, Arun Subramaniyan, Tianjun Zhang, Yu Zeng, Reetuparna Das, David Blaauw, and Satish Narayanasamy. Genax: A genome sequencing accelerator. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 69–82, 2018.
- [187] Konstantina Koliogeorgi, Nils Voss, Sotiria Fytraki, Sotirios Xydis, Georgi Gaydadjiev, and Dimitrios Soudris. Dataflow acceleration of smith-waterman with traceback for high throughput next generation sequencing. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pages 74–80, 2019.
- [188] Moniek Jaspers. Acceleration of read alignment with coherent attached fpga coprocessors. *Master Thesis, TU Delft*, 2015.
- [189] Neil A. Miller, Emily G. Farrow, Margaret Gibson, Laurel K. Willig, Greyson Twist, Byunggil Yoo, Tyler Marrs, Shane Corder, Lisa Krivohlavek, Adam Walter, Josh E. Petrikin, Carol J. Saunders, Isabelle Thiffault, Sarah E. Soden, Laurie D. Smith, Darrell L. Dinwiddie, Suzanne Herd, Julie A. Cakici, Severine Catreux, Mike Ruehle, and Stephen F. Kingsmore. A 26-hour system of highly sensitive whole genome sequencing for emergency management of genetic diseases. *Genome Medicine*, 7(1), September 2015.
- [190] Illumina. Accuracy improvements in germline small variant calling with the dragentm platform. <http://albiogen.ru/upload/documents/DRAGEN%20Accuracy%20Application%20Note.pdf>, 2019. Accessed Jun. 13, 2021.
- [191] Riadh Ben Abdelhamid and Yoshiki Yamaguchi. A block-based systolic array on an hbm2 fpga for dna sequence alignment. In *International Symposium on Applied Reconfigurable Computing (ARC)*, pages 298–313. Springer, 2020.
- [192] Jeremie S Kim, Damla Senol Cali, Hongyi Xin, Donghyuk Lee, Saugata Ghose, Mohammed Alser, Hasan Hassan, Oguz Ergin, Can Alkan, and Onur Mutlu. GRIM-filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies. *BMC Genomics*, 19(S2), May 2018.
- [193] R. Banakar, S. Steinke, Bo-Sik Lee, M. Balakrishnan, and P. Marwedel. Scratchpad memory: a design alternative for cache on-chip memory in embedded systems. In *Proceedings of the Tenth International Symposium on Hardware/Software Codesign. CODES 2002 (IEEE Cat. No.02TH8627)*, pages 73–78, 2002.

## Bibliography

---

- [194] P.R. Panda, N.D. Dutt, and A. Nicolau. Efficient utilization of scratch-pad memory in embedded processor applications. In *Proceedings European Design and Test Conference. ED TC 97*, pages 7–11, 1997.
- [195] Mahmut Kandemir and Alok Choudhary. Compiler-directed scratch pad memory hierarchy design and management. In *Proceedings 2002 Design Automation Conference (DAC)*, pages 628–633, 2002.
- [196] Sumesh Udayakumaran, Angel Dominguez, and Rajeev Barua. Dynamic allocation for scratch-pad memory using compile-time decisions. *ACM Transactions on Embedded Computing Systems (TECS)*, 5(2):472–511, May 2006.
- [197] Yu-Hsin Chen, Tushar Krishna, Joel S. Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits (JSSC)*, 52(1):127–138, 2017.
- [198] Yangjie Qi, Shuo Zhang, and Tarek M. Taha. Trim: A design space exploration model for deep neural networks inference and training accelerators. *arXiv preprint arXiv:2105.08239*, 2021.
- [199] Robert M. Radway, Andrew Bartolo, Paul C. Jolly, Zainab F. Khan, Binh Q. Le, Pulkit Tandon, Tony F. Wu, Yunfeng Xin, Elisa Vianello, Pascal Vivet, Etienne Nowak, H.-S. Philip Wong, Mohamed M. Sabry Aly, Edith Beigne, Mary Wootters, and Subhasish Mitra. Illusion of large on-chip memory by networked computing chips for neural network inference. *Nature Electronics*, 4(1):71–80, January 2021.
- [200] Jessica S Black, Manuel Salto-Tellez, Ken I Mills, and Mark A Catherwood. The impact of next generation sequencing technologies on haematological research – a review. *Pathogenesis*, pages 9 – 16, 2015.
- [201] Institut Pasteur. Institut Pasteur isolates strains of coronavirus 2019-nCoV detected in France. <https://pasteur.fr/en/press-area/press-documents/institut-pasteur-isolates-strains-coronavirus-2019-ncov-detected-france>, 2020. Accessed Jun. 13, 2021.
- [202] Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu. Accelerating pointer chasing in 3D-stacked memory: Challenges, mechanisms, evaluation. In *2016 IEEE 34th International Conference on Computer Design (ICCD)*, pages 25–32, 2016.
- [203] Allan M Maxam and Walter Gilbert. A new method for sequencing dna. *Proceedings of the National Academy of Sciences*, pages 560–564, 1977.
- [204] Frederick Sanger, Steven Nicklen, and Alan R Coulson. Dna sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, pages 5463–5467, 1977.

- 
- [205] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–945, October 2004.
- [206] Karl V Voelkerding, Shale A Dames, and Jacob D Durtschi. Next-Generation Sequencing: From Basic Research to Diagnostics. *Clinical Chemistry*, pages 641–658, 04 2009.
- [207] David A. Wheeler, Maithreya Srinivasan, Michael Egholm, Yufeng Shen, Lei Chen, Amy McGuire, Wen He, Yi-Ju Chen, Vinod Makhijani, G. Thomas Roth, Xavier Gomes, Karrie Tartaro, Faheem Niazi, Cynthia L. Turcotte, Gerard P. Irzyk, James R. Lupski, Craig Chinault, Xing zhi Song, Yue Liu, Ye Yuan, Lynne Nazareth, Xiang Qin, Donna M. Muzny, Marcel Margulies, George M. Weinstock, Richard A. Gibbs, and Jonathan M. Rothberg. The complete genome of an individual by massively parallel DNA sequencing. *Nature*, pages 872–876, 04 2008.
- [208] Raja Appuswamy, Jacques Fellay, and Nimisha Chaturvedi. Sequence alignment through the looking glass. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 257–266, 2018.
- [209] Jorge Gonzalez-Dominguez, Yongchao Liu, and Bertil Schmidt. Parallel and scalable short-read alignment on multi-core clusters using UPC++. *PLOS ONE*, page e0145490, January 2016.
- [210] Pejman Lotfi-Kamran, Boris Grot, Michael Ferdman, Stavros Volos, Onur Kocberber, Javier Picorel, Almutaz Adileh, Djordje Jevdjic, Sachin Idgunji, Emre Ozer, and Babak Falsafi. Scale-out processors. In *Proceedings of the 39th Annual International Symposium on Computer Architecture (ISCA)*, ISCA ’12, page 500–511, USA, 2012. IEEE Computer Society.
- [211] M Holtgrewe. Mason - A read simulator for second generation sequencing data. Technical Report 962, Freie Universitaet Berlin, 2010.
- [212] Animal Breeding and Wageningen University Genomics Centre. Parus\_major1.0.3 - Genome - Assembly - NCBI. [https://www.ncbi.nlm.nih.gov/assembly/GCF\\_001522545.1/](https://www.ncbi.nlm.nih.gov/assembly/GCF_001522545.1/), 2016. Accessed Jun. 01, 2021.
- [213] GfK. Gfk smart home study 2018. <https://insights.gfk.com/study-the-2018-gfk-smart-home-study>, 03 2018. Accessed Jun. 01, 2021.
- [214] Pew Research Center. <http://www.pewinternet.org/fact-sheet/mobile/>, 2018. Accessed Dec. 01, 2018.
- [215] Karen Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *Computing Research Repository (CoRR)*, abs/1409.1556, 2015.
- [216] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 160–167, 2008.

## Bibliography

---

- [217] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.
- [218] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. Eie: Efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 243–254, 2016.
- [219] Vinayak Gokhale, Jonghoon Jin, Aysegul Dundar, Berin Martini, and Eugenio Culurciello. A 240 g-ops/s mobile coprocessor for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, June 2014.
- [220] Flavio Ponzina, Miguel Peon, Andreas Burg, and David Atienza. E2cnns: Ensembles of convolutional neural networks to improve robustness against memory errors in edge-computing devices. *IEEE Transactions on Computers (TC)*, pages 1–1, 2021.
- [221] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, November 2016. USENIX Association.
- [222] Dave McCracken. Posix threads and the linux kernel. In *Ottawa Linux Symposium*, page 330, 2002.
- [223] Rajeshwari Banakar, Stefan Steinke, Bo-Sik Lee, Mahesh Balakrishnan, and Peter Marwedel. Scratchpad memory: a design alternative for cache on-chip memory in embedded systems. In *Proceedings of the Tenth International Symposium on Hardware/Software Codesign. CODES 2002 (IEEE Cat. No.02TH8627)*, pages 73–78, 2002.
- [224] Majid Namaki Shoushtari. *Software Assists to On-chip Memory Hierarchy of Manycore Embedded Systems*. PhD thesis, UNIVERSITY OF CALIFORNIA, IRVINE, 2018.
- [225] Stefanos Kaxiras and Margaret Martonosi. Computer architecture techniques for power-efficiency. *Synthesis Lectures on Computer Architecture*, 3(1):1–207, 2008.
- [226] Vasileios Spiliopoulos, Akash Bagdia, Andreas Hansson, Peter Aldworth, and Stefanos Kaxiras. Introducing dvfs-management in a full-system simulator. In *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 535–545, 2013.
- [227] Salessawi Ferede Yitbarek, Tao Yang, Reetuparna Das, and Todd Austin. Exploring specialized near-memory processing for data intensive operations. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1449–1452, 2016.



- 
- [228] Saugata Ghose, Amirali Boroumand, Jeremie S Kim, Juan Gómez-Luna, and Onur Mutlu. Processing-in-memory: A workload-driven perspective. *IBM Journal of Research and Development*, 63(6):3:1–3:19, 2019.
- [229] Arvind Sridhar, Alessandro Vincenzi, Martino Ruggiero, Thomas Brunschwiler, and David Atienza. 3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling. In *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 463–470, 2010.
- [230] Krste Asanović and David A Patterson. Instruction sets should be free: The case for risc-v. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146*, 2014.
- [231] Andrew Shell Waterman. *Design of the RISC-V instruction set architecture*. PhD thesis, UC Berkeley, 2016.
- [232] Pasquale Davide Schiavone, Francesco Conti, Davide Rossi, Michael Gautschi, Antonio Pullini, Eric Flamand, and Luca Benini. Slow and steady wins the race? a comparison of ultra-low-power risc-v cores for internet-of-things applications. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–8, 2017.
- [233] Sparsh Mittal and Jeffrey S Vetter. A survey of cpu-gpu heterogeneous computing techniques. *ACM Computing Surveys (CSUR)*, 47(4):1–35, 2015.
- [234] Nikko Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [235] Sainathan Ganesh Iyer and Anurag Dipakumar Pawar. GPU and CPU accelerated mining of cryptocurrencies and their financial analysis. In *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)**I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 599–604. IEEE, 2018.




## YASIR MAHMOOD QURESHI

Avenue de la Vallombreuse 81

1008 Prilly, Switzerland

+41 77 910 91 45

[yasirmahmoodqureshi@gmail.com](mailto:yasirmahmoodqureshi@gmail.com)

 yasirmahmoodqureshi

 <https://www.linkedin.com/in/yasir-mahmood-qureshi/>



---

## STRENGTHS

- RTL Designer for CPUs and Memories
- SoC/ASIC/FPGA Design, Embedded Systems
- Low Power Design Specialist
- Heterogeneous Systems Architect

---

## EDUCATION

- |   |                         |                    |
|---|-------------------------|--------------------|
| • <b>PhD Electrical Engineering</b>                             |                         | <b>2016 - 2021</b> |
| Swiss Federal Institute of Technology, Lausanne (EPFL)          |                         |                    |
| • <b>MSc. Embedded Computing Systems (Erasmus Mundus)</b>       | <b>Average: A Grade</b> | <b>2011 - 2013</b> |
| NTNU, Trondheim, Norway   | A - Grade               | 2012 - 2013        |
| University of Southampton, UK                                   | A - Grade 83.42%        | 2011 - 2012        |
| • <b>Bachelor of Electrical Engineering</b>                     | <b>GPA 3.99/4.00</b>    | <b>2006 - 2010</b> |
| National University of Sciences and Technology (NUST), Pakistan |                         |                    |

---

## WORK EXPERIENCE

- **Embedded Systems Laboratory (ESL), EPFL, Switzerland.** **2016 - 2021**  
*Doctoral Thesis - "Architecture Exploration and Optimization of Heterogeneous Many-Core Compute And Memory Architectures with Architectural Extensions"*
  - Development of Gem5-X multi-core heterogeneous simulator
  - System level architectural exploration and optimization for state-of-the-art video encoding, CNNs, and next generation genome sequencing (NGS) applications
  - Integration and utilization of in-cache computing accelerator in L1-D cache
  - Integration and utilization of RRAM as computational engine in CPU pipeline
  - Exploration of 3D stacked HBM2 memory with heterogenous compute cores for various applications
  - Scratch Pad Memory (SPM) integration and utilization for CNNs and RNNs within a CPU based system
- **ARM, Cambridge, UK.** **2015 - 2016**  
**Systems & Software Group (SSG). Job Title: 'Engineer'**
  1. **Design**
    - Block level design specifications for different ARM CoreSight SoC blocks
    - Implementing ARM CoreSight SoC components in RTL according to design specifications
    - Integrating different IP components together in RTL
    - Implementing clock domain crossings (CDC) and multiple clock domains in RTL
    - IP-XACT for Configurable IPs

## 2. Verification

- Built block level testbenches using UVM for verification
- Write top level test cases
- Power aware simulations
- Out of Box testing.

## • ARM, Cambridge, UK.

2013 - 2015

### **Systems & Software Group (SSG). Job Title: 'Graduate Engineer'**

As a Graduate Engineer at ARM, I did three graduate rotations

#### 1. Cortex-M7

- Worked on tarmac trace verification of Cortex-M7 CPU
- Created test cases to stress test the exception model for Cortex-M7
- Debug and fixed the cases for tarmac trace mismatches

#### 2. Dynamic Memory Controller (DMC)

- Formal verification for APB protocol
- Memory mapping of the system at RTL level
- Formal verification with clock domain crossing and multiple clock domains

#### 3. CoreSight SOC

- Built block level testbenches using UVM for verification

## • ATMEL, Norway.

2013

### **IC Design Division. Job Title: "Thesis Student"**

- Implementation of system consisting of AVR8 core, AMBA-APB interconnect and APB compliant peripherals in RTL
- Design of a bus matrix for AVR8 to APB interfacing in RTL
- Testing and verification of the complete system
- Synthesis of the system for low power optimization

## • ARM, Cambridge, UK.

2012

### **Processor Cores Division. Job Title: 'Summer Placement'**

Worked on system validation project for ARM Cortex-M and Cortex-R class CPUs which plugs the gap in not just having system level validation for these cores, but it goes beyond in having MATLAB providing realistic simulated scenarios of real-world control problems and doing RTL-MATLAB co-simulations.

- Specified the memory map of different peripherals in the system
- Setting up and running the ModelSim-Simulink/MATLAB co-simulation environment with the Cortex-M0+ integration kit.
- Loopback test in the ModelSim-Simulink/MATLAB co-simulation
- Used Simulink Coder for the auto generation of C code for the controller from its MATLAB model
- Implemented the RTL model of the DAC peripheral with some skewing effect and AHB slave interface
- Integrated DAC and the ADC peripheral into one system and do the loopback test with the MATLAB co-simulation

• **Center for Advanced Research in Engineering (CARE) Pvt Ltd Pakistan. 2010 - 2011**

**Job Title:** 'Design Engineer' [www.carepvtltd.com](http://www.carepvtltd.com)

- Development of software reconfigurable hardware architecture for Software Defined Radio (SDR).
- Implementation of PHY for SDR on multiplatform system, comprising Xilinx Spartan 3A-DSP FPGA, TI DSP and ARM GPP.
- Development and implementation of OFDM, GMSK, W-CDMA and FM waveforms for data and speech transmission over SDR
- Implementation for burst detection algorithm for WBNR waveform on Xilinx Spartan 6 FPGA.
- Design and Implementation of MAC protocol for Ad-Hoc radio networks.

• **Next Generation Intelligent Network Research Center, Pakistan. 2008**

**Job Title:** 'Research Student' [www.nexginrc.org](http://www.nexginrc.org)

- Worked as Research Student in Remote Patient Monitoring System (*RPMS*) *Project Team*, on the development of
- Clinical digital temperature sensor, digital blood pressure apparatus and digital pulse meter
- Data acquisition and transmission by PDA via GPRS

## TEACHING

• **Microprogrammed Embedded Systems 2017 - 2020**

- Lead Teaching Assistant (TA) for this course, in which programming a embedded system is taught using Nintendo DS as the emdedded platform.
- Lab sessions
- Exam preparation and evaluation
- Project evaluation

## TRAININGS

1. Comprehensive SystemVerilog – Doulos (17 March 2014 – 21 March 2014)
2. UVM Adopter Class – Doulos (12 August 2014 – 15 August 2014)

## TECHNICAL SKILLS

• **Tools and Packages**

Synopsys Design and Power Compiler	Cadence using AMS C35 and Cadence Encounter	ModelSim	Xilinx ISE and EDK
Synopsys VCS	Jasper Gold	IP-XACT	Synplify Pro
ModelSim – MATLAB Cosimulation	Altera Quartus II	MATLAB/Simulink	PrimeTime
QuestaSim	Formality	LTSpice	HAL
Code Composer Studio (CCS)	ARM ACL	AutoCAD	Gem5-X/Gem5 Simulator
Cortex M0+ Integration Kit (IK)	Cortex-M0 Design Start	MS-Office	

- **Programming Skills:**

SystemVerilog HDL	C/C++ programming language
Verilog HDL	Python
VHDL	Assembly language programming for ARM, x86
UVMS	M-file programming

- **Hardware Platforms:**

Virtex-II Pro FPGA	Raspberry Pi
Altera Cyclone IV	TI MSP430
Xilinx Spartan 3-A DSP, Spartan 6 FPGA	ARM JUNO Platform
TI DM 6446 DSP Processor	Nvidia Jetson Nano

---

## LANGUAGE SKILLS

1. **English** – Fluent
  2. **French** – Beginner (A1 – CEFR )
- 

## PUBLICATIONS

1. **Gem5-X: A Many-Core Heterogeneous Simulation Platform for Architectural Exploration and Optimization,**  
ACM Transactions on Architecture and Code Optimization (TACO), 2021
  2. **Genome Sequence Alignment-Design Space Exploration for Optimal Performance and Energy Architectures,**  
IEEE Transactions on Computers, 2020
  3. **An In-Cache Computing Architecture for Edge Devices,**  
IEEE Transactions on Computers, 2020
  4. **Gem5-X: A Gem5-Based System Level Simulation Framework to Optimize Many-Core Platforms,**  
Spring Simulation Conference (SpringSim), Tucson, AZ, USA, 2019
  5. **BLADE: A BitLine Accelerator for Devices on the Edge**  
GLSVLSI '19: Proceedings of the 2019 on Great Lakes Symposium on VLSI, 2019.
  6. **A Product Engine for Energy-Efficient Execution of Binary Neural Networks Using Resistive Memories,**  
IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC), 2019.
  7. **Energy proportionality in near-threshold computing servers and cloud data centers: Consolidating or Not?,**  
Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018.
  8. **Design and Layout of a Two Stage High Bandwidth Operational Amplifier,**  
WASET International Conference on Electrical, Computer, Electronics and Communication Engineering (ICECECE), Venice, 2012.
- 

## PROJECTS

### INTERNET OF THINGS (IOT)

#### Optimal Exercise Training System

Two STM32L151xD ARM Cortex M3 based custom boards integrated with ECG and PPG sensors along with 3D accelerometer is used for Machine Learning (ML) based training system for athletes. The system is connected via Bluetooth Low Energy (BLE) module to our Android

application. The whole system is optimized to be low power, with some application parts execution on the edge sensor nodes and others on the Android device.

## **MASTER THESIS**

### **Low-Power Optimized AMBA-APB Bus Connection For AVR8 Co-Processor**

- Design, implementation and integration of bus matrix to connect 8-bit AVR8 core with 32-bit APB for low power optimization in Verilog RTL
- AVR8 is Atmel's low power 8-bit CPU and AMBA-APB is an industry standard, low power, 32-bit bus interconnect by ARM for peripheral modules on SoCs.
- Implementation of different bus matrix designs for different optimization parameters
  - Low power design
  - High performance design
  - Low area design

## **AUTUMN SPECIALIZATION PROJECT**

### **Modeling of Cache Coherence Protocols**

- Study of memory consistency models and cache coherence protocols
- Modeling techniques and abstraction levels in modeling
- Investigating multi-processor simulators like GEM5 and MultiCacheSim for modeling of the cache coherence protocols

## **UNDERGRADUATE DESIGN PROJECT**

System level design and hardware implementation of Real-Time Signal Processor of a Phase-Coded Pulse Doppler Radar on Virtex-II FPGA for detection of high-altitude planes.

- IBM PowerPC405 used as main control processor
- Design and implementation of a co-processor for real time radar processing and its integration with PowerPC in RTL
- Design of optimized parallel and pipelined architecture of FFT engine
- Implementation of memory controller for fast access of DDR RAM

## **SEMESTER PROJECTS**

- **Realization of Digital Components:** Design and implementation of RSA encryption/decryption core on Altera Cyclone IV in VHDL.
- **Computer Design:** Design and implementation of a fully pipelined MIPS processor with branch predictor on Spartan6 FPGA in VHDL.
- **Digital System Synthesis:** Low Power Data Path Design using Clock Gating.
- **VLSI Design Project:** Design and Implementation of Solar Tracker and MPPT on Altera Cyclone IV in Verilog.
- **System on Chip:** Full Custom Design and layout of Op-Amp in Cadence AMS C35
- **System on Chip:** Design and Layout of Sequence Decoder in Synopsys Design Compiler and Cadence Encounter
- **Control Systems:** Position and velocity PID control of DC servo motor using PIC 18F452 microcontroller.
- **Power Electronics:** DC-DC Converter using Maximum Power Point Tracking for Solar Panels.
- **DSP:** Image Compression using Wavelet Transform.

- **Digital System Design:** Optimized design and implementation of Wallace Tree and Hybrid Adder on Xilinx Spartan-3E FPGA.
  - **Communications:** Simulation and noise analysis of QAM, QPSK, FSK, ASK, PAM, SSB and DSB in MATLAB
  - **Microcontroller:** Air Mouse based on accelerometer sensor using PIC Microcontroller.
  - **Digital Logic Fundamentals:** 1. 16-Bit ALU using discrete components. 2. Implemented a Laser Based Security System
  - **Electronics-II:** Full Duplex Mode Intercom System.
  - **Electronics-I:** Designing of  $\pm 30V$  Variable and  $\pm 5V$  Fixed Power Supply.
  - **Data Structures:** String Matching using Finite Automata Algorithm.
- 

## AWARDS/HONORS

- **EMECS-thon 2013** – Won the People’s Choice award in the 48 hours embedded marathon EMECS-thon competition at NTNU
  - **Won the prestigious Erasmus Mundus Scholarship** for 2011-2013.
  - **Presidents Gold Medal** for being the best university student for 2009-2010 at NUST
  - **Prime Minister’s Gold Medal** for *best CGPA* in the Department of Electrical Engineering in Bachelor’s of Engineering at NUST.
  - **Won Prime Minister’s Gold Medal Scholarship.**
  - **Maximum Performance Based Scholarships** in all semesters at NUST
  - **Declared Overall best student** in academics at NUST for 2009-2010
  - **Won MERIT Award** at Asia Pacific Information Communication Technology Alliances (APICTA) competition 2010 in Malaysia (Awarded to SDR Project)
  - **Won PASHA 2010 IT award** for best project on communication systems. (Awarded to SDR Project)
  - **Higher Secondary School Examination Scholarship** 2006 by F.BISE
  - **President of University Student Body**
  - **Awarded Plaque of Excellence** by Dean, College of E&ME, NUST. (7 times)
  - **Won Interschool Football Tournament, 2000** Leading my team as the captain
- 

## EXTRA-CURRICULAR ACTIVITIES

- **Fire Warden**, at ARM
  - **Vice President**, University Environment Club, NUST
  - **Liaison Head**, Society of Information and Communication Technology, NUST
  - **Speaker at MATLAB Workshop**, 2010 College of E&ME, NUST
  - **Organizer, Alumni Reunion**, 2010, NUST
-





