Processing-in-Memory DRAM Architectures for Neural Network Applications

Processing-in-Memory-DRAM-Architekturen für neuronale Netzwerkanwendungen

Vom Fachbereich Elektrotechnik und Informationstechnik
der Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau
zur Verleihung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Dissertation

von

Chirag Sudarshan

geboren in Bangalore, Indien

Tag der Einreichung:	12. Juni 2023
Tag der mündlichen Prüfung:	24. Oktober 2023
Dekan des Fachbereichs:	Prof. Dr. rer. nat Marco Rahm
Vorsitzender der Prüfungskommission:	Prof. Dipl.-Ing. Dr. Gerhard Fohler
1. Berichterstatter:	Prof. Dr.-Ing. Norbert Wehn
2. Berichterstatter:	Prof. Dr. Ahmed Hemani

Eidesstattliche Erklärung

Ich erkläre an Eides statt,

dass ich die vorliegende Dissertation selbst angefertigt habe und alle von mir benutzten Hilfsmittel angegeben habe,

dass ich die Dissertation oder Teile hiervon noch nicht als Prüfungsarbeit für eine staatliche oder wissenschaftliche Prüfung eingereicht habe, und

dass ich nicht die gleiche oder eine andere Abhandlung bei einem anderen Fachbereich oder einer anderen Universität als Dissertation eingereicht habe.

Kaiserslautern, 12. Juni 2023

Chirag Sudarshan

Preface

Most of the presented contributions have therefore been generated in close cooperations with a number of colleagues and partners. To clearly distinguish between what has been discovered solely on my own and in interaction with others, in the following I will use first person singular (i.e. “I”, “my”...) or first person plural (i.e. “we”, “our”...), respectively.

Abstract

Emerging applications based on machine learning and Deep Neural Networks (DNNs) are data-driven and memory intensive. Hence, there is a recent shift from compute-centric architectures to memory-centric architectures. In this context, Processing-in-Memory (PIM) is a new emerging compute paradigm to overcome the memory issues such as bandwidth limitation, long data access latency, and very high data access energy, which are encountered in conventional compute platforms for memory-intensive applications. The key idea of PIM is to integrate the computation units inside the memory to exploit the massive internal data parallelism and to minimize data movement energy consumption. Thereby achieving the highest energy efficiency and/or increased throughput. Researchers have investigated various memories, such as SRAM, DRAM, and emerging memories (i.e. RRAM, FEFET, MRAM, etc.) as PIM candidates. The drawbacks for SRAMs and emerging memories for PIM are 1) SRAM-based PIM accelerators have a low memory capacity and 2) emerging memories are not technologically as mature as DRAMs or SRAMs. Unlike SRAMs and RRAMs, DRAMs satisfy both technological maturity and high memory capacity requirements, which makes them a prominent candidate for PIM. There is a recent surge in research on DRAM-based PIM, both from academia and industries, but it is relatively less compared to SRAM or emerging memory-based PIM. Very recently in the year 2021-2022, major memory vendors like Samsung and SK Hynix have published their respective initial engineering samples of DRAM-PIM.

The commodity DRAM technology is different in comparison to standard CMOS logic technology-compatible SRAMs or some emerging memories. It is highly optimized for the density, yield, and thus cost per bit. Commodity DRAM architectures also do not explicitly contain computation units apart from the peripheral logic for data access. Hence, most state-of-the-art DRAM-PIM architectures integrated the computation units at the output of the DRAM bank (i.e. bank peripheral region) without modifying the core structure of the existing commodity DRAM bank architecture. Thereby sufficing a very appealing feature known as Dual-Mode that allows these architectures to be employed as both high-density main memory and high energy-efficient accelerator. In a commodity DRAM, the highest data parallelism and lowest data movement energy is in the Sub-Array region, i.e., inside the DRAM bank. E.g. the amount of parallel data available near the DDR4 bank output is 128-bits while in the Sub-Array region, it is 2 KB. As a result, a few state-of-the-art architectures explored the integration of computation units inside the DRAM banks to further increase the throughput and energy efficiency in comparison to the bank peripheral region approach. However, these architectures required the modification of a highly optimized DRAM Sub-Array design. Furthermore, the bank's internal region like near the Sub-Array region also imposes high area constraints to the new computation units.

In this thesis, I investigate a series of novel DRAM-PIM architectures for DNN inference and training that enable the integration of computation units inside the commodity DRAM bank without majorly modifying the existing commodity bank design and strictly the Sub-Array design to profit from the DRAM road-map. Additionally, I also ensure seamless integration of the DNN computation units by fulfilling the area constraints of the near Sub-Array region. My architectures will fulfill the Dual-Mode features while achieving the highest throughput and energy efficiency in comparison to State-of-the-art DRAM-PIM architectures. Furthermore, I will also extend my research to address the memory issues in non-PIM conventional platforms. I propose a novel DRAM architecture referred to as Dual-Sense-Amplifier (DSA) that reduces the DRAM data access latency which results in increased bandwidth utilization without increasing energy consumption. All my novel DRAM architectures will be fully compatible with any type of commodity DRAM architecture (e.g. HBM, GDDR, etc.) and retain most of its circuit designs. Finally, I investigate the reduction of the DRAM energy without modifying the commodity DRAM architecture. For this, I propose a lean, low power, and low area DRAM memory controller architecture that is tailored for IoT/embedded applications and leverages on the transprecision computing methodology.

Acknowledgments

This thesis is the result of research I conducted between 2017 and 2023 at the Microelectronic Systems Design Research Group at the Rhineland-Palatinate Technical University of Kaiserslautern-Landau. I have been supported and encouraged in some way or the other by various colleagues, friends, and family during this period. There are a few individuals I would like to thank specifically.

My heartfelt gratitude to my supervisor Prof. Dr.-Ing. Norbert Wehn for giving me an opportunity to work on this topic. My appreciation goes out to him forever for his invaluable guidance that has shaped me not only as a good researcher but as a dynamic individual as well. I always enjoyed discussions with him and gained a lot of academic and industrial knowledge in the field of microelectronics. I am extremely grateful for all the support he has given me in my academic and personal life. My special thanks to Prof. Dr. Ahmed Hemani for his interest in my thesis and for serving on my thesis committee.

I would like to extend my thanks to Dr.-Ing. Christian Weis for sharing all his industrial knowledge related to DRAMs. I am very grateful for all my colleagues with whom I had the chance to work in the Microelectronic Systems Design Research Group and other collaborating institutes: M.Sc. Jan Lappas, M.Sc. Taha Soliman, Prof. Dr.-Ing. Matthias Jung, Dr.-Ing. Deepak M. Mathew, Dr.-Ing. Vladimir Rybalkin, M.Sc. Lukas Steiner, M.Sc. Muhammad Mohsin Ghaffar, M.Sc. Mohammad Hassani Sadi, M.Sc. Kira Kraft, Dipl.-Inf. Claus Kestel, Dipl.-Ing. Matthias Herrmann, M.Sc. Oliver Griebel, Prof. Dr.-Ing. Stefan Weithoffer, M.Sc. Johannes Feldmann, M.Sc. Maximilian Schöffel, M.Eng. Lucas Johannsen, M.Sc. André Lucas Chinazzo, M.Sc. Jonas Ney, M.Eng. Carl Rheinländer, M.Sc. Frederik Lauer, Dipl.-Math. Uwe Wasenmüller, M.Sc. Menbere Tekleyohannes, Dr.-Ing. Javier Alejandro Varela, Eng. Éder Zulian, Dr.-Ing. Bilal Hammoud, M.Sc. Mohammadreza Esmaeilpour Quchani, M.Sc. Hussien Abdo, M.Sc. Mohamed Moursi, and Dr. Dimitrios Stathis. Without all their support my research could have never been complete.

This journey would not have been smooth without Martina Jahn; thanks to all her administrative and non-technical support. Many thanks to the technical support team: Dipl.-Ing. (FH) Hans-Peter Goldhammer, Dipl.-Ing. (FH) Roland Volk, Andreas Christmann, and Markus Müller.

Thanks to Dr. Asha Srinivas, M.Sc. K. Shashidhar, and Dr. M. Nagabhushanam for all their valuable guidance during the early stages of my education. This acknowledgment would be incomplete without thanking my parents, my wife, and my brother for their continuous encouragement and a lifetime of support.

Contents

1 Introduction **1**

1.1 Motivation 5

1.2 Scientific Research Gaps 7

1.2.1 Minimal Modifications to Commodity DRAM Architecture 7

1.2.2 Computation Unit Placement Location in a Commodity DRAM . . 9

1.2.3 Scalability 11

1.3 Contributions and Thesis Outline 11

2 DRAM Background **19**

2.1 DRAM Device Architecture 19

2.1.1 Sub-Array 23

2.1.2 Primary Sense Amplifiers 24

2.2 DRAM Protocol and Memory Controller 26

3 Design Space Exploration **33**

3.1 Design Dimension 1: Computation Unit Location 33

3.1.1 Inside Sub-Array Computation 34

3.1.2 Near Sub-Array Computation 36

3.1.3 Bank Peripheral Region Computation 37

3.1.4 IO Region Computation 40

3.2 Design Dimension 2: Computation Unit Data Format 40

3.3 Design Dimension 3: Memory Controller 42

3.4 Summary 43

4 DRAM PIM Architecture for DNN Inference - Digital Approach 45
4.1 Background: Power-of-two Quantization . . . 46
4.2 Architecture . . . 48
4.2.1 Primary Processing Unit (PPU) . . . 48
4.2.2 Bank-Level Architecture . . . 52
4.2.3 Device-Level Architecture . . . 54
4.3 Experimental Setup . . . 55
4.4 Results . . . 56
4.4.1 Accuracy Results . . . 56
4.4.2 Area Results . . . 57
4.4.3 Performance Results . . . 59
4.4.4 Energy Efficiency Results . . . 59
4.4.5 DRAM-PIM Comparision . . . 61
4.5 Summary & Discussion . . . 63

5 DRAM PIM Architecture for DNN Inference - Mixed-Signal Approach 65
5.1 Background: Bit-Decomposed MAC . . . 65
5.2 Mixed Signal DRAM-PIM Architecture . . . 67
5.2.1 PPU Architecture . . . 69
5.2.2 Bank-level Architecture . . . 70
5.2.3 Device-Level Architecture . . . 75
5.2.4 PSA Region Floorplan . . . 76
5.2.5 Data Mapping . . . 77
5.3 Experimental Setup . . . 77
5.4 Variation Analysis . . . 79
5.5 Results . . . 82
5.5.1 Accuracy Evaluation . . . 82
5.5.2 Area Evaluation . . . 83
5.5.3 Performance Evaluation . . . 84
5.5.4 Power Evaluation . . . 85
5.5.5 Comparison . . . 86
5.6 Summary . . . 88

6 DRAM PIM Architectural Approaches for Scalability 91
6.1 Architectural Approach for Inference + Training 92
6.1.1 PPU Architecture 92
6.1.2 Device-Level Architecture 94
6.2 Unified Interface and Protocol for External Control 96
6.3 Summary 99

7 A Novel DRAM Architecture for Improved Bandwidth Utilization and Latency Reduction using Dual-Page Operation 101
7.1 Architecture 103
7.2 Experimental Setup 106
7.3 Results 107
7.3.1 Area Evaluation 107
7.3.2 DSA Timing Evaluation 109
7.3.3 Average Response Latency and Bandwidth Evaluation 109
7.3.4 Energy and Power Evaluation 110
7.4 Summary 111

8 DRAM Memory Controller Design for Transprecision Applications 113
8.1 DDR3 DRAM Memory Controller for Transprecision Computing 113
8.1.1 Memory Controller Features for Transprecision Computing 114
8.1.2 Memory Controller Architecture 116
8.1.3 Results 118
8.2 Summary 120

9 Conclusion and Future Work 121

Zusammenfassung 125

Acronyms 131

Bibliography 133

List of Figures 147

List of Tables 153

List of Algorithms 155

Chapter 1

Introduction

Nowadays, we have various advanced systems such as autonomous vehicles, humanoid robots, smart devices, virtual assistants, virtual reality, etc. The three key domains that have contributed significantly to these innovations are *Artificial Intelligence* (AI), Microelectronics, and High-speed Wireless Communication. It is interesting to note that the 1940s marked a major milestone for all these domains, viz. the first mathematical model of biological neuron (1943) [1], the invention of a point-contact transistor (1947) [2,3] and the mathematical theory of communication by Shannon (1948) [4]. This thesis research targets the domain of Microelectronics for AI applications.

Over the years, the field of microelectronics has made phenomenal advancements. The key driving factor for the microelectronics industry is the famous forecast projected by Gordon Moore in the year 1965 (denoted as Moore's Law) [5]. He speculated that the number of transistors in Integrated Circuit (IC) chips doubles every year, which was later updated to every two years. Today's ICs contain about 100 billion sub-nm transistor nodes that are $700\times$ smaller, $3600\times$ faster, $90000\times$ energy efficient, and $6000\times$ cheaper than the 1971's semiconductor technology nodes [6]. By 2030, microelectronic companies like Intel are targeting one trillion transistor nodes per chip, with an estimated increase of $10\times$ compared to the year 2022 [7]. The continuous miniaturization efforts to keep up with Moore's Law enable the design of powerful computation platforms with 100s of compute cores (i.e. throughput scaling) and memory ICs with GBs of data capacity (i.e. density scaling).

The progress in computation platforms is extensively leveraged by AI, along with the abundant amount of data gathered through social media and Internet-of-Things, to achieve groundbreaking results in the field of Image classification and Natural Language Processing. Today's AI domain is mainly driven by the biological brain neuron inspired *Deep Neural Network* (DNN) models. There have been several DNN models proposed over the last decade to improve prediction accuracy and support various datasets/applications. With every new DNN model, the already large parameter size is continuing to increase exponentially (i.e. memory intensive, refer Figure 1.1). The total number of computation operations

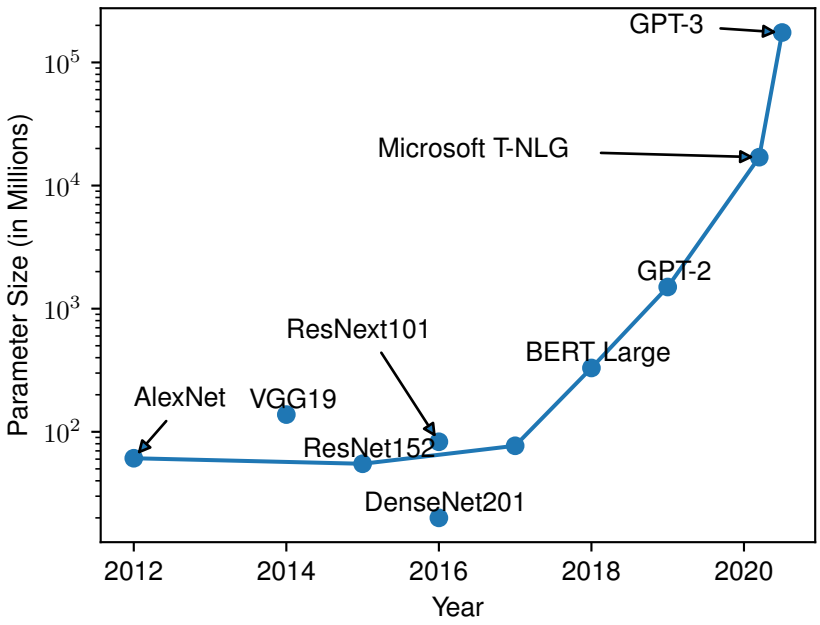

Figure 1.1: DNN parameter size increasing over the years. Source [8]

for DNN models and the required computational throughput is also increasing [8]. Moore's Law effectively aids in meeting the high computation throughput and memory-footprint requirements of various DNN models. Despite this, the relatively slow interconnect between the computation core and memory devices imposes a major challenge on computation platforms, creating a bottleneck to the effective utilization of peak throughput.

The memory-intensive nature of DNN results in a frequent exchange of data between the computation cores and memory devices (i.e. both on-chip and off-chip). Over the generations of semiconductor technology, the wire-based interconnect bandwidth of memory devices did not scale at the same rate as peak computation throughput or memory density due to the increased influence of RC parasitics [9]. This slow scaling of these wire-based interconnects limits the memory interface's maximum frequency and total number of parallel data pins. This is especially the case for external interfaces. Consequently, many memory devices' (particularly external memories) are failing to deliver the data at computation demanded bandwidth and are resulting in throughput saturation at a much lower value as compared to peak throughput [10]. This challenge is known as 'Memory Wall' [11] and is one of the critical challenges in many conventional computation platforms like GPU, TPU, and other custom accelerators for DNN. It is important to first differentiate the memory bandwidth vs memory latency to better understand the memory wall challenge. Peak memory bandwidth is defined as the product of frequency and data bus interface width. On the other hand, memory latency is the total amount of time required to access a data word from the memory devices, which depends on the memory type and memory request

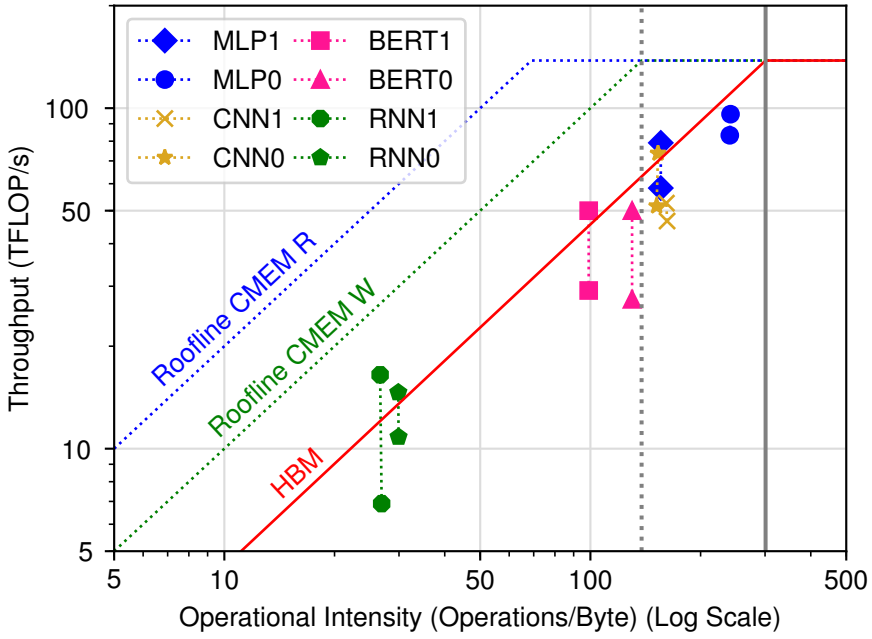

Figure 1.2: Roofline model for Google TPUv4i. The lower point and higher points show the results of TPUv4i with 16 MB on-chip memory and 144 MB on-chip memory respectively. The operation intensity (i.e. num of operations per memory access) for lower points is with reference to HBM2 and for higher points is with reference to large on-chip memory. The solid vertical line and dashed vertical line depict the ridge point for the HBM2 and large on-chip scenarios, respectively. Source [13]

scheduling policy [12]. The effective utilization of peak bandwidth majorly depends on memory latency. Reducing the average data access latency using an efficient scheduling policy increases the overall bandwidth utilization. Memory latency particularly plays an important role in the case of external memories like DRAMs, due to their long and internal state-dependent varied latencies (e.g. 4 clks to 40 clks), as compared to on-chip memories like SRAMs that have low and fixed latencies (e.g. 1 clk or 2 clk). The memory wall challenge arises when the computation demanded memory bandwidth is higher than the actual memory bandwidth utilization or peak memory bandwidth.

As an example for showcasing the memory wall issue, I consider a DNN inference accelerator chip from Google called as TPUv4i [13]. Figure 1.2 shows the roofline model [14] of Google's TPUv4i accelerator for two scenarios: 1) with nominal 16 MB on-chip memory + 8 GB off-chip DRAM (lower data points) and 2) with high 144 MB on-chip memory + 8 GB off-chip DRAM (upper data points). The roofline model is used to identify if an application is computation or memory bound based on its operation intensity (operations/byte of data access) relative to the ridge point, i.e. to the left of the ridge point is memory-bound and to the right is compute-bound. For a nominal on-chip memory case, the throughput of all the networks is bound by DRAM devices in spite of connecting each core with one of

Table 1.1: Comparison of computation units energy vs memory data access energy. Source [13]

Operation		**pJ/operation**	
		45 nm	7 nm
Add	Int8	0.03	0.007
	Int32	0.1	0.03
	BFoalt16	–	0.11
	IEEE FP16	0.4	0.16
	IEEE FP32	0.9	0.38
MUL	Int8	0.2	0.07
	Int32	3.1	1.48
	BFoalt16	–	0.21
	IEEE FP16	1.10	0.34
	IEEE FP32	3.7	1.31
Mem Size		**pJ/64-bit access**	
SRAM	8 KB	10	7.5
	32 KB	20	8.5
	1 MB	100	14
DRAM		$\approx$ 45 nm	$\approx$7 nm
	DDR3/4	1300	1300
	HBM2	–	250-450
	GDDR6	–	350-480

the highest bandwidth-providing DRAM devices (i.e. an 8 GB HBM2 device with 614 GB/s bandwidth). TPUv4i compensates for DRAM-bound challenges by adding a large block of additional 128 MB on-chip memory called CMEM. This CMEM in TPUv4i occupied 28% of the total chip area ($\approx$400 mm^2) implemented in 7 nm technology. It also has a high read and write bandwidth of 2 TB/s and 1 TB/s, respectively. Despite the high capacity, high bandwidth, and low latency of on-chip memory, the throughput of larger networks like RNN (i.e. 254 MB) is still constrained by the on-chip memory and memory hierarchy. This problem will be more severe, for relatively lower bandwidth DRAMs like DDR3/4, LPDDR4/5, etc., for other extremely large networks like GPT-2, GPT-3, etc., and for DNN training (10$\times$ more memory intensive than inference [15])

The Data accesses from memory devices consume orders of magnitude higher energy than computation operations. This is another major constraint to the conventional computation platforms, where memory devices contribute a predominant portion of the overall energy. Table 1.1 compares the data access energy from on-chip SRAM and off-chip DRAM vs the computation operations' energy consumption in 45 mn and 7 nm technology. Data access energy from on-chip memories like SRAMs and off-chip memories like DRAMs consumes orders of magnitude higher energy than computation operations. The farther the data is from the computation unit, the higher the data access energy. As a result, the con-

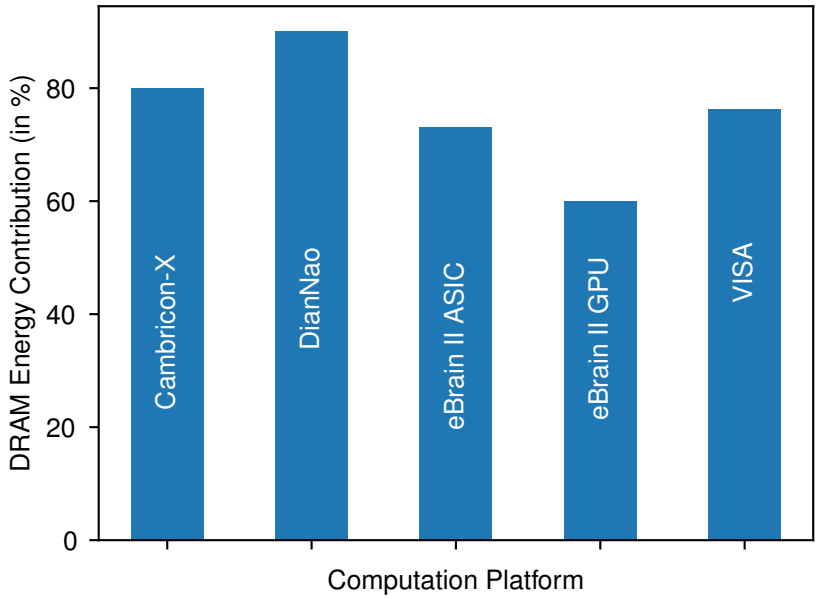

Figure 1.3: DRAM energy contribution in various DNN accelerators. Source [16–19]

tribution of external main memories like DRAM is the highest in many conventional DNN computation platforms. Figure 1.3 shows the percentage of energy consumed by DRAM devices on DNN accelerators. More than 50% of the total energy consumption was contributed by DRAM devices. In summary, the overall energy efficiency and performance of conventional computation platforms are majorly impacted by memory devices, particularly external devices like DRAM. Hence, there has been a recent shift from compute-centric architectures to memory-centric architectures.

1.1 Motivation

Processing-in-Memory (PIM) is a new emerging compute paradigm that overcomes the aforementioned memory issues (i.e. memory-wall and high data access energy) by integrating computation units inside memory devices. The tight coupling of computation units and memory arrays enables the effective utilization of high internal data bandwidth and reduced data movement energy. For example, the internal bandwidth at the bank output in one of the latest HBM2 DRAM devices is at least 4x higher than the device's external bandwidth [20]. Similarly, the energy consumed for moving data to the bank output from the memory cells is 50% lower than the external off-chip data transfers [20]. The internal bandwidth is further increased and data moment energy is further reduced as the computation units are moved closer to the memory cells.

Over the past 3 to 5 years, PIM accelerators have been extensively investigated with different memory types, like SRAM, DRAM, and emerging memory technologies (e.g. RRAM, FeRAM, MRAM, PCM, etc.) as PIM candidates. Initially, SRAM-based and emerging memory-based PIM accelerators gained a lot of attention from academia and industries. This is because of two main reasons: (1) accessibility of the respective memory technology and (2) compatibility with standard logic technology (i.e. valid for SRAMs and many emerging memory types). However, there are fundamental drawbacks in using SRAMs or emerging memories as PIM. Emerging memories are not technologically as mature as DRAMs or SRAMs and suffer from high device variations that directly impact DNN accuracy, especially for analog-based PIM [21, 22]. Another challenge of using the emerging memories in the context of DNN training is the low endurance [23–25] that restricts the number of training cycles to a few thousand. For example, the low endurance value of FeFET memories (E.g. 10^6) and RRAM/PCM memories (E.g. 10^8) constraints the training cycles to a low value of 400 and 40000 cycles, respectively (i.e. $Total_TrainingCycles = Endurance \div Weight_Updates_per_TrainingCycle$). The above calculation is for an example DNN training with 50 epochs and 50 batches per epoch that will have a resulting weights update of 2500 times per training cycle. In comparison to the emerging memories, SRAMs and DRAMs have high endurance of $> 10^{16}$. Additionally, most emerging memory macros have low density and a limited capacity of 1 Mb to 100 Mb [23], which is much lower than the memory footprint of DNNs (Figure 1.1). The authors of [26] employed 1Mb RRAM macros to design a large capacity 8Gb PIM accelerator chip, but it resulted in a high power consumption of 62W/chip. On the contrary to [26], the authors of [27] confined the PIM memory capacity size to the Mb range but integrated large global buffers and external DRAMs for DNN data storage that resulted in DRAM consuming 94% of total energy. Similarly, SRAM-based PIM accelerators also have a low memory capacity and are not suitable for networks with a memory footprint in the order of Gigabits. Unlike emerging memories and SRAMs, DRAMs satisfy technological maturity, endurance, and high memory capacity requirements. Furthermore, DRAMs are one of the most widely used external main memories due to their high density and high capacity. However, DRAM-based PIMs are relatively less explored compared to SRAMs or emerging memory based PIMs. From an energy overhead perspective (in conventional computation platforms), accessing a data word (e.g. 64-bit) from an external DRAM consumes $10\times$ to $50\times$ more energy compared to accessing the data from a local SRAM memory (Table 1.1). Due to these aforementioned criteria, this research work focuses on DRAM device-based PIM. At the start of this thesis work, to the best of my knowledge there existed only two other publications targeting DRAM-PIM architectures i.e. AMBIT [28] (2017) and DRISA [29] (2017). Since then, there is a surge in publications on DRAM-PIM architectures, both from academia and industries [20, 30–40], but it is relatively less compared to SRAM or emerging memory-based PIM. Very recently in the year 2021/22, major memory vendors like Samsung and SK Hynix have also published their respective first engineering samples of DRAM-PIM in [34] (FIMDRAM, 2021) and [37] (Hynix-AiM,

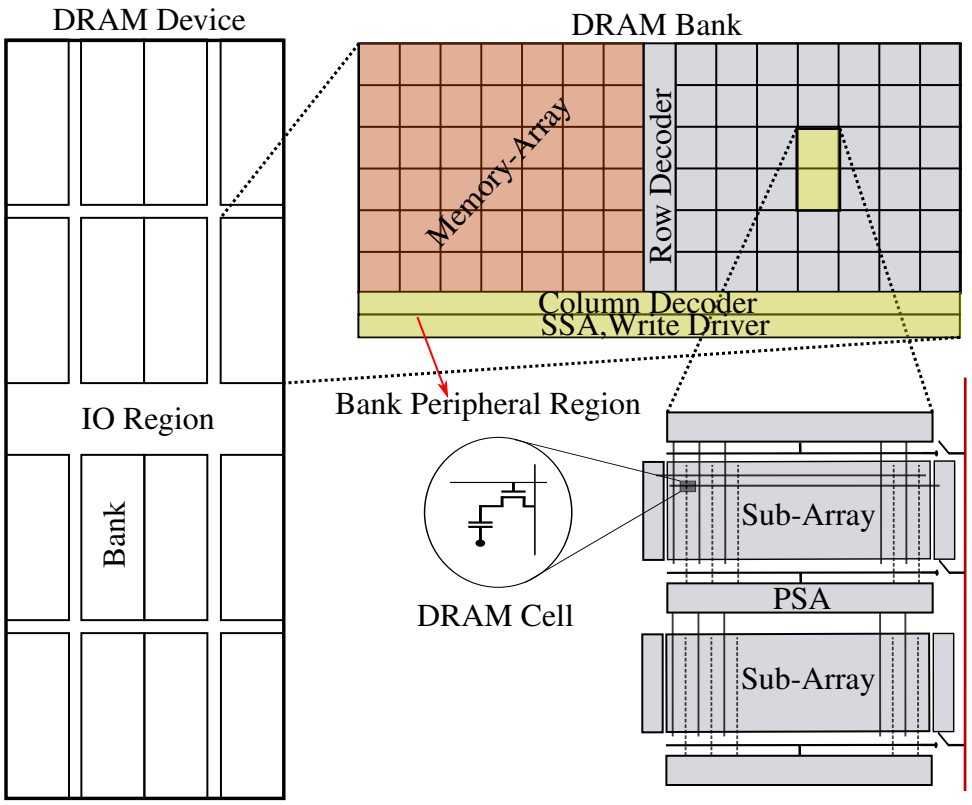

Figure 1.4: DRAM hierarchical structure. The DRAM device height-to-width ratio, number of banks and bank placement is based on a real 10 nm, 2nd gen, 12Gb commodity LPDDR5 chip [41].

2022). DRAM-PIM accelerator chips from start-ups like UPMEM are also commercially available [35].

1.2 Scientific Research Gaps

The main goal of this thesis is to research on energy-efficient DRAM PIM architecture design. This section describes the research gaps and challenges addressed in this thesis.

1.2.1 Minimal Modifications to Commodity DRAM Architecture

The DRAM market is driven by minimizing cost-per-bit and maximizing bits-per-mm^2. Commodity DRAM architecture is a hierarchical and repetitive structure of Sub-Array, Memory-Arrays, and Banks (refer Figure 1.4). These internal blocks of DRAMs, particularly Sub-Arrays, are highly optimized for density, yield, and thus cost per bit. The Sub-Arrays are designed using specialized buried gate *Recessed Channel Array Transistor* (RCAT) or *Sphere-shaped Recessed Channel Array Transistor* (SRCAT) or Saddle Fin cell

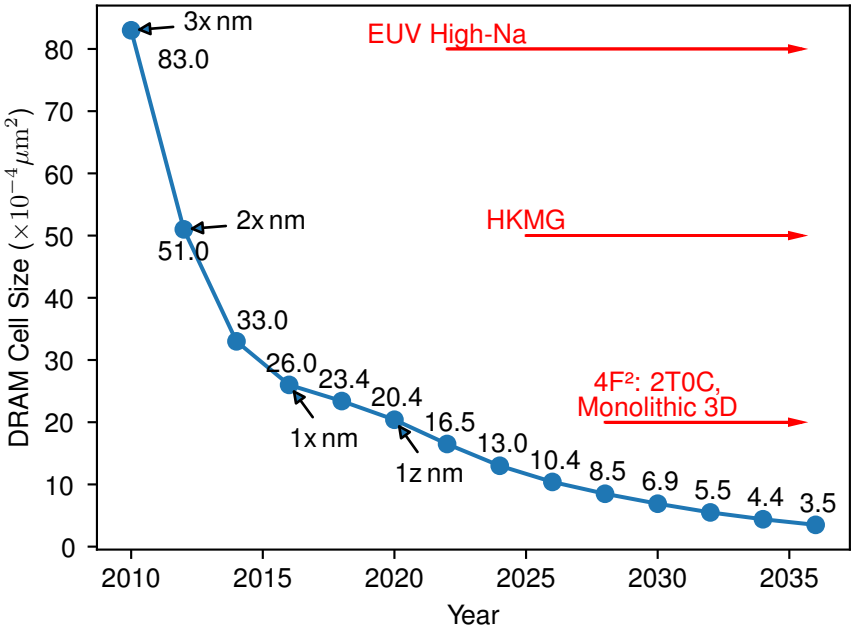

Figure 1.5: Scaling of DRAM cell size over the years and future predictions. 3x nm, 2x nm, 1x nm, and 1z nm process node sizes are approximately equal to 32 nm, 28 nm, 18 nm and 12 nm, respectively. Source [42]

access transistors, slanted active area, stacked cylindrical cell capacitors, shared drain terminal between the cell access transistors, fine bitline pitch, and buried wordlines [43–48] (refer Chapter 2 for more details). Furthermore, to continuously increase the bit-density and keep up with Moore's Law, DRAM vendors are constantly thriving to reduce the cell size and optimize the existing Sub-Array design. DRAM vendors are investing in several technological and material innovations, like High-K, EUV, $4F^2$ cell, etc., to continue the cell size scaling and increase the density. Please note, F defines the smallest feature size in a given DRAM technology, E.g. bitline pitch. Figure 1.5 shows the scaling of DRAM cell size over the years and future predictions [42]. The cell size shrinking is slowing down in recent times in comparison to the early 2010s. All these factors showcase the criticality of the DRAM Sub-Array design in today's times. Hence, it is important for DRAM-PIM architectures to be compatible with the existing commodity DRAM memory blocks and avoid modification of the Sub-Array design. A few state-of-the-art DRAM-PIM architectures [28, 30, 31, 33] proposed to modify these highly optimized Sub-Arrays with special compute rows (refer Chapter 3), which results in design modifications and impacts the overall density.

Goal 1: *The first goal of this thesis is to investigate new DRAM-PIM architectures that strictly do not modify the Sub-Arrays and are comparable to commodity DRAM counterparts in terms of density and capacity. Furthermore, it also aim to be compatible with exist-*

ing commodity DRAM architectures, incur minimal modifications to the dense structure of banks, reuse most of the existing DRAM circuitry, and maintain the thermal design power within the range of commodity DRAM die (i.e. 1-2 W). Overall, by sufficing this goal, the new DRAM-PIM architectures implicitly support "Dual-Mode" feature that enables them to concurrently operate as both a high-density main-memory and an energy-efficient accelerator. Thereby aligning with the roadmap of DRAM vendors.

Please note that the dual-mode goal was set at the beginning of this thesis in 2017 and was first published in 2019 [49]. This dual-mode is inline with the goal of industrial DRAM-PIM architectures like Samsung's FIMDRAM [34] which was later published in 2021. Samsung's FIMDRAM architecture 3D integrated four conventional DRAM dies over the PIM dies to compensate for density loss by the PIM dies. This allows FIMDRAM to simultaneously operate as both a high-density main memory and an energy-efficient accelerator. Apart from the 3D approach, my goal at the start of this thesis was to investigate the possibility of dual-mode feature within the DRAM die, thereby supporting this feature for both 2D (e.g. DDR5, LPDDR5) and 3D (HBM) scenarios.

1.2.2 Computation Unit Placement Location in a Commodity DRAM

The primary building block of DNN is the *Multiply-Accumulate* (MAC) operation, whose hardware implementation is extremely area-intensive and power-hungry. In this context, one of the most crucial architectural decisions is the placement location of MAC units in the commodity DRAM architecture without violating the aforementioned goals in Section 1.2.1. As mentioned before, commodity DRAM architectures are dominated by the area, density, and yield-optimized memory blocks. Additionally, the commodity DRAMs do not explicitly contain computation units or have a dedicated region for computation units apart from the peripheral logic for external data accesses and error correction codes. The possible locations to integrate the computation units within a commodity DRAM are 1) *Inside Sub-Array*, 2) *Near Sub-Array* or *in Primary Sense Amplifier (PSA) region*, 3) *Bank peripheral region* and 4) *IO region* (Figure 1.4). The farther the computation units are placed from Sub-Arrays, parallelism is reduced and more energy is required to transfer data. E.g. the amount of parallel data available near the DDR4 bank peripheral region is reduced to 256-bits per cycle as compared to 1024-bits near each Sub-Array or 2 KB per DRAM page. However, as the computation units are moved closer to the Sub-Array, the engineering cost to integrate the new computation units increases and leads to the modification of highly optimized memory blocks.

The state-of-the-art DRAM-PIM publications have proposed to integrate computation units in all four regions. *Inside Sub-Array* approach is proposed in publications like [28, 30, 31, 33] that implement a majority function and perform bulk AND or OR operations. However, all these architectures require modification of the Sub-Array design with special rows and dual contact DRAM cells (refer Chapter 3). The next possible location for integrating computation units is the *PSA region*. It offers a similar data parallelism and data

movement energy as "Inside Sub-Array". Contrary to Sub-Array design, the PSA region, Bank peripheral region, and IO region are compatible with standard CMOS logic design, and the transistors of these regions are comparable to logic technology with planar transistors [50]. This enables the integration of logic gates in PSA region but under certain area restrictions. The commodity DRAM PSA region is highly sensitive towards additional logic and is constrained especially by the Sub-Array width (i.e., area restriction), fine bit-line pitch, and the limited number of available metal layers (i.e., three or four) in the DRAM process. The PSA sharing in *Open Bitline* (OBL) architecture further restricts the integration of large computation units in this region (refer Chapter 3). Metal tracks have to be reserved for bit-lines and reference bit-lines to pass through the computation units and connect to the respective PSAs, which imposes area restrictions. The prior publications that integrate the DNN computation units in the PSA region are [29, 32]. However, these architectures proposed to reorganize the bank and sub-array dimensions. Their DRAM-PIM consisted of 1024 banks, eight Sub-Arrays per bank, and 256×2096 Sub-Array size, which are not consistent with commodity DRAMs.

The *bank peripheral region* and *IO region* have no such physical/area restrictions and enable the seamless integration of new computation blocks without modifying the existing memory bank design. This approach is used in Samsung's FIMDRAM [20, 34] and McDRAM [39,40], UPMEM [35], and Hynix's Newton [36] and AiM [37] architectures. However, these architectures encounter reduced data parallelism and higher energy consumption due to the frequent movement of data between the PSA and bank periphery. Additionally, the number of parallel MAC operations computed is limited by the amount of data available near the bank peripheral region (i.e. 256-bit per clock cycle). Further, throughput scaling has to rely mainly on increasing the number of banks, which is currently limited to 32 banks as per commodity DRAM standards published by JEDEC [51–57]. In this context, *PSA region* not only has high parallel data of 1024-bits per Sub-Array and lowest data movement energy but also enables throughput scaling by parallelly operating multiple Aub-Arrays in each bank (e.g. 64 per bank).

Goal 2: *The second goal of this thesis is to explore various design choices and leverage approximate techniques to integrate the computation units like MAC in the PSA region, rather than confining only to the bank peripheral region. The new PIM architecture will partition the computation unit into Primary Processing Unit (PPU) and Secondary Processing Unit (SPU), where the light-weight PPU are integrated in the PSA region and substantial SPU logic is integrated in the bank peripheral region. Thereby exploiting the advantages of both regions to the fullest. Additionally, the goal is also not to violate the area constraints of PSA region, enable seamless integration, and incur minimal modifications to the existing DRAM bank architecture/design.*

1.2.3 Scalability

In the past, most of the DRAM-PIM publications have mainly targeted DNN inference. However, DNN training is 10× more memory intensive as compared to inference [15]. As mentioned in Section 1.2.2, the computation kernel in DNN is MAC, whose integration in PSA region is extremely challenging due to area constraints. The challenge is further escalated for DNN training as it requires 32-bit floating-point computations versus inference that have achieved state-of-the-art results with integer 8-bit [58, 59]. The 32-bit floating-point multiplication unit requires approximately 20× more power (Table 1.1) and a much larger area than the 8-bit integer multiplication [60]. Hence, the DRAM PIM architecture design has to be scalable such that the PSA region is capable of supporting various types of computation units without violating the aforementioned goals in Sections 1.2.1 and 1.2.2. The number of new commands and the types of compute commands of DRAM PIM vary from vendor to vendor due to the lack of a standard for DRAM-PIM. This results in the vendor-specific modification of the DRAM-PIM interface and memory controller to support new commands, which is contrary to the commodity DRAM counterparts. Hence, it is also important to design a unified interface and external control protocol that is compatible with DRAM-PIM architectures from multiple vendors. Thereby enabling the drop-in replacement of DRAM-PIM chips from one vendor with another vendor without any hardware modifications.

Goal 3: *The Third goal of this thesis is to propose DRAM PIM architecture that is scalable for any new computation unit with minimal modifications to the core structure and to investigate a unified interface that is compatible across the vendors.*

1.3 Contributions and Thesis Outline

The main objective of this thesis is to investigate novel DRAM PIM architectures for DNN applications, which address the aforementioned scientific research gaps and challenges. Additionally, this thesis also contributes towards the optimization of DRAM sub-system's latency, energy, and bandwidth utilization for conventional non-DRAM-PIM computation platforms. Figure 1.6 show the outline of this thesis. Chapter 2 gives the necessary background on DRAMs. The vast design space of DRAM-PIM, the associated design challenges, and the related works are discussed in Chapter 3. The novel DRAM-PIM architectures that accomplish the goals listed in Sections 1.2.1 and 1.2.2 are presented in two parts in Chapter 4 (digital approach) and Chapter 5 (mixed-signal approach). The necessary DNN background required for these novel DRAM-PIM architectures are explained in the respective chapters. The novel architectural approaches to satisfy the scalability goal listed in Section 1.2.3 are presented in Chapter 6. Chapter 7 presents a new DRAM architecture called *Dual-Sense-Amplifier* (DSA) that aims to optimize latency and effectively utilize the available DRAM bandwidth. This new DRAM architecture reduces the memory wall issues in the conventional non-DRAM-PIM platforms. In Chapter 8, the design

of DRAM memory controller for Transprecision applications is presented. The purpose of this memory controller is to reduce commodity DRAM energy consumption in conventional non-DRAM-PIM architectures. The related work for the contributions in Chapter 7 and Chapter 8 is presented in their individual chapters.

The contributions of this thesis are previously published in the following conference and journal papers: [49, 61–68]. Patents are filed for DRAM-PIM architectures [69, 70] and for DSA architecture presented in this thesis [71]. In total this thesis has resulted in 9 research publications [49, 61–68] and 3 patent fillings [69–71]. In addition to these publications and patents, during my thesis research period between 2017 to 2023, I have also co-authored 13 more research publications [72–84] and I'm the co-inventor of one patent [85]. Most of the presented contributions have been generated in close cooperation with many colleagues and partners. To clearly distinguish between the contributions discovered solely on my own and in interaction with others, henceforth I will use first-person singular (i.e. "I", "my"...) and first-person plural (i.e. "we", "our"...), respectively. This work was partly funded by the EU project OPRECOMP under grant No.732631, the German Federal Ministry of Education and Research (BMBF) within the project Open6GHuB under grant 16KISK004, Carl Zeiss foundation under the grant "Sustainable Embedded AI", the German Federal Ministry of Education and Research (BMBF) within the project ZuSE-KI-AVF under grant 16ME0379, and Huawei Technologies Co., Ltd., Chengdu, China. A summary of the novel contributions and their respective publications are listed below.

- **Design Space Exploration (Chapter 3)** [65]
  Most of the prior DRAM-PIM publications have proposed architectures with specific design approaches and techniques. The majority of prior publications do not provide a comprehensive view of the DRAM-PIM design space and a critical assessment of the various design dimensions. In this thesis, I first present a detailed analysis of the DRAM-PIM's design space. My analysis covers the following design dimensions of the DRAM-PIM design space: 1) computation unit placement location, 2) computation data format, and 3) memory controller and interface modifications required for external control or programming of PIM In this chapter, I explore the various design constraints and implementation challenges of incorporating computation units into the commodity DRAM architecture. I also explore the impact of computation units' placement location, within the commodity DRAM architecture, and computation units' data format on overall throughput and energy efficiency. This chapter provides new insights into various constraints of commodity DRAM architectures from a PIM perspective. The findings presented in this chapter establish the foundation for all the investigations and design decisions of novel DRAM-PIM architectures presented in Chapters 4, 5, and 6. In addition, it also briefs the state-of-the-art DRAM-PIM architectures. To the best of my knowledge, I investigated all the available DRAM-PIM architecture publications [20, 28–38, 86–89] with the emphasis on DNN for this design space assessment.

- **DRAM PIM Architecture for DNN Inference (Chapter 4 and 5) [49, 63, 64]**
  As a starting point, I targeted to design DRAM-PIM architectures for inference task and suffice only Goal 1 and Goal 2. As mentioned in Sections 1.2.1 (Goal 1) and 1.2.2 (Goal 2), I aim to investigate new DRAM-PIM architectures that suffice the following factors: (1) *Dual-Mode*: a DRAM-PIM architecture that functions both as an accelerator and as a high-density DRAM. (2) Integrate the new computation units in the PSA region without majorly modifying the bank design, retaining the basic structure of commodity DRAMs, and strictly not modifying the highly optimized Sub-Array design. More specifically, the architectures aim to reuse most of the existing DRAM circuitry, internal buses, and internal operations to execute neural network inference and transfer data between the computation units. (3) Ensure seamless integration of the DNN computation units by fulfilling the area constraints of the PSA region. In comparison to the state-of-the-art, this is one of the first works to explore the feasibility of integrating the computation units in the PSA region of the commodity DRAM banks without majorly modifying its optimized structure.

  The foremost design decision in my architectures is the partition of computation units into two entities, i.e., a lean PPU integrated in the PSA region and a larger SPU integrated near the bank peripheral region. This split between the first and second entities is required because the surroundings of the DRAM Sub-Array are constrained by limited area, which is not the case near the bank periphery. The computation kernel in DNN is MAC, these MAC operations require area-intensive and power-hungry multipliers, whose integration in PSA region is extremely challenging. Hence, my designs are based on the following two algorithmic optimizations: 1) Logarithmic quantization that employs shifter-based approximate multiplier, and 2) bit-decomposed MAC. I present the PPU, SPU, and overall device-level architecture design for both algorithmic optimizations. This contribution is divided into two chapters, where I present a logarithmic quantization based architecture - digital approach in Chapter 4 and the bit-decomposed MAC based mixed signal approach in Chapter 5. My research work was the first to present a mixed-signal architecture for DRAM-PIM. The mixed signal approach achieved the best results and required the lowest modifications to the PSA region among the two. This novel 8 Gb DRAM PIM device achieves a peak performance of 142.8 GOPS (23.813 GOPS/bank) while consuming a power of 945.765 mW, which results in the highest energy efficiency of 151.07 GOPS/W in comparison to the prior DRAM-PIM publications. The area overhead of the proposed PIM device is 12.63% in comparison to a commodity 8 Gb DRAM in a 2y nm technology.

  The research works published in [63, 64] were conducted in close collaboration with my colleague M.Sc. Taha Soliman from CR/AEX2, Robert Bosch GmbH. In all the publications, my main contributions were architectural exploration/investigation, circuit design, and hardware evaluations to meet the aforementioned goals (Section 1.2) and to respect the area constraints of underlying commodity DRAM architecture

without major modifications. In addition to my contribution, the following parts of these publications' contributions are presented in my colleague's PhD thesis. (1) Logarithmic quantization based DRAM-PIM architecture [63]: Taha Soliman contributed to the data mapping and designed the python-model of my architecture. This model was substituted with the circuit-level results obtained by me to compute the device-level results such as accuracy, throughput, power consumption, and energy efficiency. (2) Bit-decomposed MAC based mixed-signal DRAM-PIM architecture [64]: Taha Soliman made contributions to the algorithmic evaluations and approximations of bit-decomposed MAC for low DNN accuracy loss. The data mapping for this architecture was done by both Taha Soliman and me, where I did the data mapping at Sub-Array and bank-level, and Taha Soliman did the data mapping at the device level. In summary, my main research contribution is to leverage the existing approximation techniques to investigate novel hardware architectures for DRAM-PIMs that are compatible with commodity DRAMs and achieve the highest energy efficiency in comparison to state-of-the-art DRAM-PIMs.

- **DRAM PIM Architectural Approaches for Scalability (Chapter 6)** [65]
  The previous architectures explicitly targeted inference and addressed only Goals 1 and 2 without considering scalability goal (i.e. Goal 3). In this chapter, I present two scalable architectural approaches. My first architectural approach, referred to as *Quasi Open Bitline* (QOBL) enables seamless integration of computation units (both floating-point and integer) without any modification of the DRAM Sub-Array or PSA design. Due to the integration of the larger computation units like floating-point MAC in the PSA using QOBL approach, the DRAM density is reduced and the "Dual-Mode" feature is violated (i.e. a high-density main memory and an energy-efficient accelerator). To address the density reduction of QOBL, I propose a DRAM-PIM device-level architecture composed of two types of banks, i.e., larger conventional banks and high parallel PIM banks. This is the first work to present architectural approaches that enable the integration of floating-point units in the PSA region without major design modifications to existing commodity DRAM circuits and without considerable loss of memory capacity for dual-mode operation. Finally, I propose a new unified interface and protocol for DRAM PIM that eliminates vendor-specific hardware modifications to the memory controller. These aforementioned contributions go beyond the considerations of the state-of-the-art DRAM-PIM architectures.

- **A Novel DRAM Architecture for Improved Bandwidth Utilization and Latency Reduction using Dual-Page Operation (Chapter 7)** [67, 71]
  In order to reduce the impact of memory-wall in conventional computing platforms, data access latency has to be reduced and the bandwidth utilization has to be increased. The major latency-introducing operations in conventional DRAM are page-misses (i.e., opening a new DRAM page/row within a bank) and refresh operations (refer Chapter 2). Enabling the simultaneous opening of multiple rows within a bank

is one way to hide these latencies. Hence, the authors of [90–93] proposed DRAM architectures that enable multiple rows per bank. However, these architectures lack flexibility and restrict the opening of multiple rows within a Sub-Array. It also requires the memory controller to know the bank's internal structure and row arrangement for avoiding collisions. Hence, I present a novel DRAM architecture referred to as *Dual-Sense-Amplifier* (DSA) that is capable of flexibly opening any two rows within a sub-array or a bank without any collisions or page restrictions. The memory controller also does not require the know-how of bank's internal structure. Further, this architecture allows access of data during a refresh phase of the DRAM without stalling the data bus. The proposed architecture is adaptable to any type of DRAM e.g. HBM, GDDR, etc., and replaces any commodity DRAM in the conventional platforms. My architecture retains most of the commodity DRAM design/circuits and results in minimal DRAM modifications. A DSA-based 8 Gb DRAM device increases the bandwidth utilization by 17.53% and the average response latency is reduced by 31.59 ns (≈25 clk cycles) compared to commodity DRAM with row-bank-column-bankgroup address mapping. The proposed DSA architecture has a low area overhead of 9.6%, for a 2y nm technology 8 Gb device, compared to a commodity DRAM device. A US patent is filed for this DSA architecture [71]. The contributions of this chapter in prior published in [67]. This publication was one of the selected papers of IEEE International Symposium on Circuits and Systems (ISCAS) 2021, and all the selected papers were published in a special issue of the IEEE Transactions on Circuits and Systems—Part II: Express Briefs (TCAS-II) [94].

- **DRAM Memory Controller Design for Transprecision Applications (Chapter 8)** [68]
  In my thesis, I not only propose architectural solutions that modify the commodity DRAM architecture but also investigate the possibility of reducing the DRAM energy by investigating novel DRAM memory controller architecture. Energy consumption is one of the major challenges for the advanced *System on Chips* (SOC). This is addressed by adopting heterogeneous and approximate computing techniques. One of the recent evolution in this context is the transprecision computing paradigm. The idea of transprecision computing is to consume an adequate amount of energy for each operation by performing dynamic precision reduction. The impact of the memory subsystem plays a crucial role in such systems. Hence, the energy efficiency of a transprecision system can be further optimized by tailoring the memory subsystem to transprecision computing. In this chapter, I present a lean, low power, low latency memory controller (ASIC design) that is appropriate for transprecision SOC. The memory controller consumes an average power of 129.33 mW at a frequency of 500 MHz and has a total area of 4.71 mm^2 for UMC 65 nm process.

  The ASIC implementation of the memory controller was part of EU project OPRECOMP under grant agreement No.732631. My colleague M.Sc. Jan Lappas from Mi-

croelectronic Systems Design Research Group, RPTU - Kaiserslautern designed the mixed-signal physical layer (PHY) block, while I designed the digital blocks of the memory controller. During my thesis research period, I also ported the DDR3 memory controller to DDR4 and successfully demonstrated it on FPGA platform. I also open-sourced my DDR4 memory controller [95]. The FPGA version of my memory controller is employed in project ZuSE-KI-AVF under grant 16ME0379 [82].

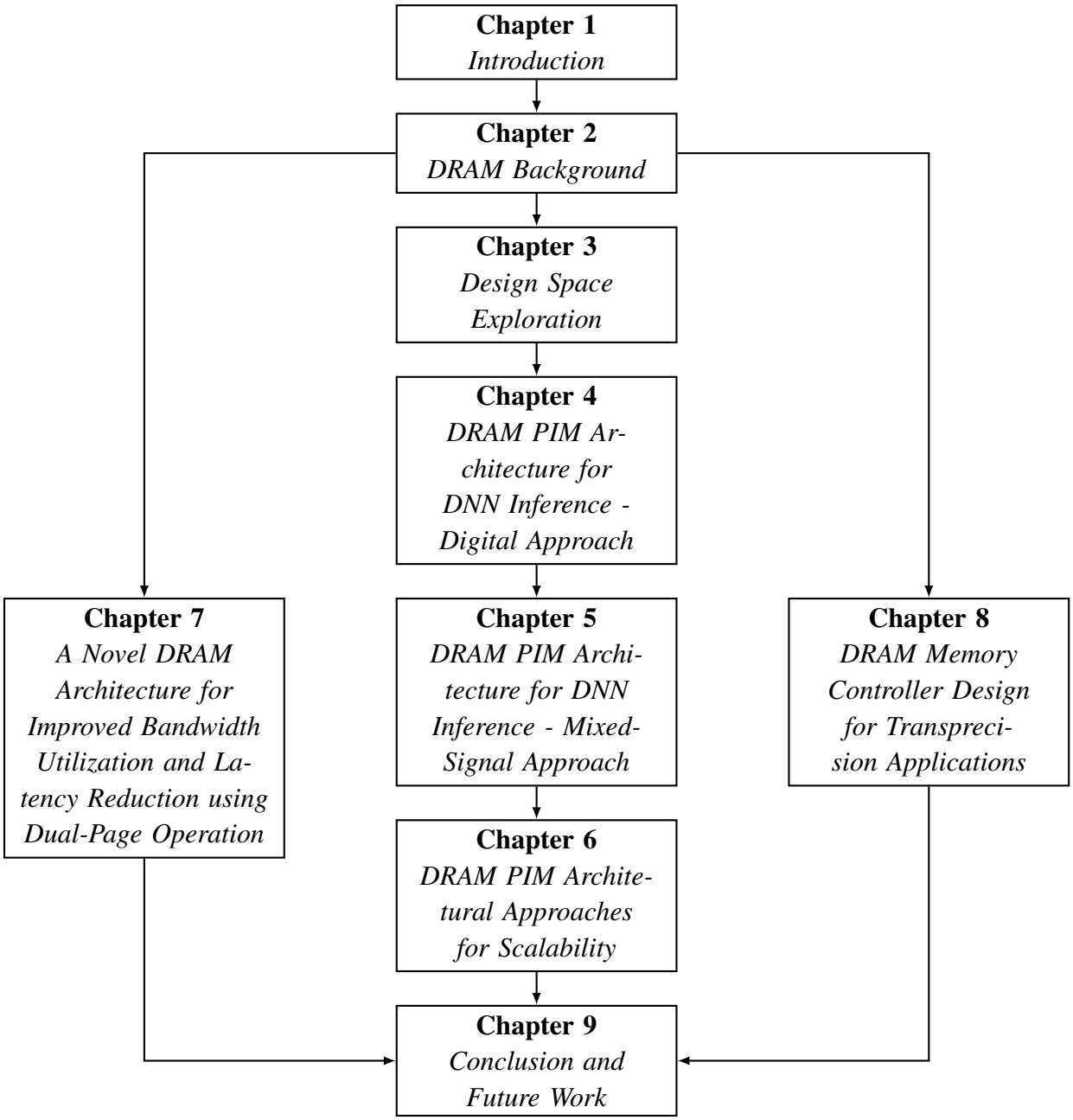

Figure 1.6: Thesis outline.

Chapter 2

DRAM Background

This chapter provides an overview of DRAM architecture, which is relevant to this thesis. DRAMs are widely used as main-memory devices in most computation systems. In today's times, all commodity DRAMs are by default Dual Data Rate (DDR) and Synchronous. The commodity DRAM standards are defined by Joint Electron Device Engineering Council (JEDEC), which is now known as JEDEC Solid State Technology Association. For the rest of this thesis, the commodity DRAM standards published by JEDEC [51–57] are referred as "JEDEC standards". JEDEC has categorized DRAMs into multiple types, depending on the power and bandwidth requirements of the computing system. These DRAM types are as follows: DDRx for general-purpose applications (e.g. DDR3, DDR4, and DDR5), LPDDRx for low-power applications (e.g. LPDDR4 and LPDDR5), GDDRx for graphics applications (e.g. GDDR5 and GDDR6), and HBMx for high bandwidth demanding applications (e.g. HBM1 and HBM2). In the following sections, I describe the basic architecture and operation principle of commodity DRAMs that are common for all types.

2.1 DRAM Device Architecture

The DRAM devices in computation platforms (e.g. accelerator cards) are typically organized in one of three ways: 1) DIMM-based, 2) Device-based, and 3) 3D/2.5D-integrated, as shown in Figure 2.1. Figure 2.2 shows the architecture of a single DRAM device or a die. It consists of a set of banks that operate in parallel. The number of banks per DRAM die depends on the DRAM type and is defined by the JEDEC standards (e.g. eight in LPDDR4). Each DRAM bank consists of storage elements that are arranged as an array of rows and columns. It is a hierarchical structure consisting of DRAM cells, Sub-Arrays, and Memory-Arrays for efficient wiring, increased speed, and reduced power consumption [97]. In this thesis, I refer to a row of Sub-Arrays as Horizontal-Block or H-Block as shown in Figure 2.2. Figure 2.3 shows the detailed view of the H-Block, where all the Sub-Arrays within the H-Block operate in parallel. To access the data from a DRAM device, an activate

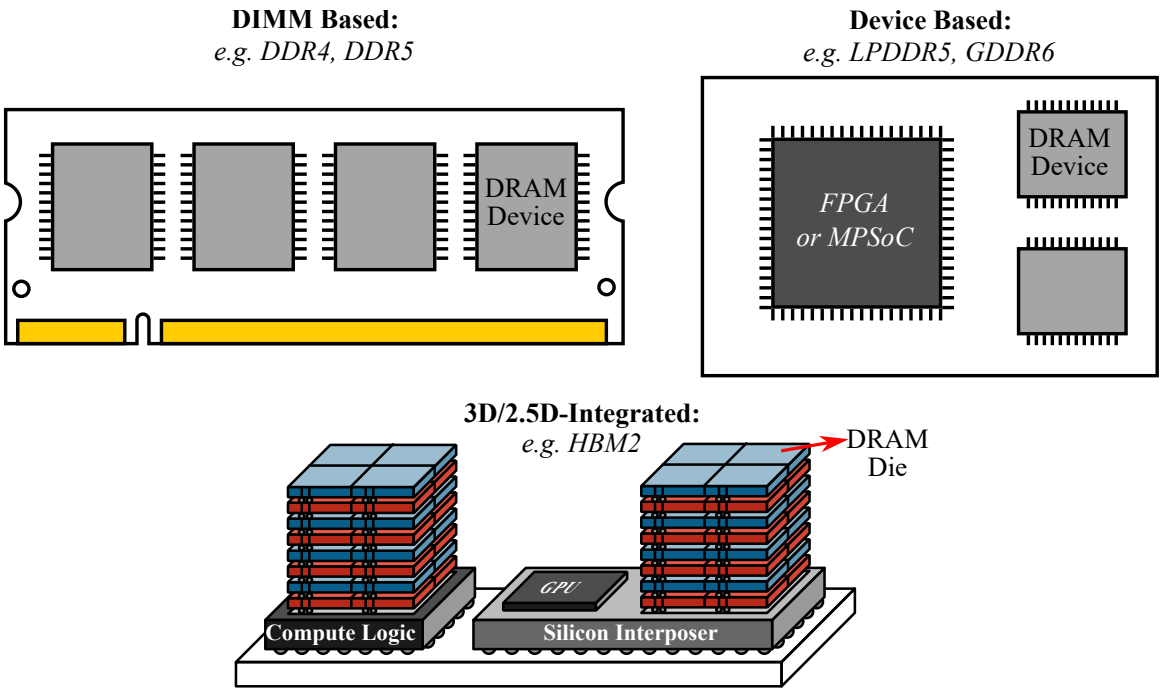

Figure 2.1: Different DRAM organization in computation systems (Image courtesy [96]).

command (ACT) is issued that enables a single row or page within a bank. Depending on the row address sent along with ACT command the appropriate *Master Wordline* (MWL) and *Local Wordline* (LWL) within a bank is driven high. All the Sub-Arrays of an H-Block share the MWLs and each MWL is further connected to four LWLs. The row decoder performs the first round of decoding that enables one MWL and the second round of decoding is performed in the *Local Wordline Driver* (LWLD) which enables one of the LWLs within a Sub-Array. The same LWL is enabled in all the Sub-Arrays of an H-Block. This enabling initiates the charge sharing between the DRAM cell capacitors of a Sub-Array row and the respective *Local Bitlines* (LBLs), which is sensed by the *Primary Sense Amplifiers* (PSAs). The concurrently sensed data by the PSAs of all the Sub-Arrays of an H-Block creates an illusion of a large row buffer or a page (e.g., $1024 \times 16 = 2\,\text{KB}$ page size). The detailed description of charge sensing and waveform of PSA is presented in Section 2.1.2. Please note, there is one PSA per LBL (i.e. 1024 PSAs for 1024-bit wide Sub-Array), and the PSAs are distributed equally and interleaved on both sides of Sub-Array due to fine pitch of LBL. The center hole region, shown in Figure 2.2, incorporates Sub-Array local control circuits and other auxiliary circuits (e.g. *Column Select Enable* (CSLEN) switches) of each Sub-Array.

After the row activation, column commands i.e. read (RD) or write (WR) are issued to access or update the desired columns of the activated page. In case of the read operation, the column decoder enables the requested *Column Selectlines* (CSLs) to access the data from the desired columns of the page to the bank peripheral region (i.e. transferring data from PSA to *Secondary Sense Amplifier* (SSA) via *Master Bitlines* (MBLs)). Similarly, for

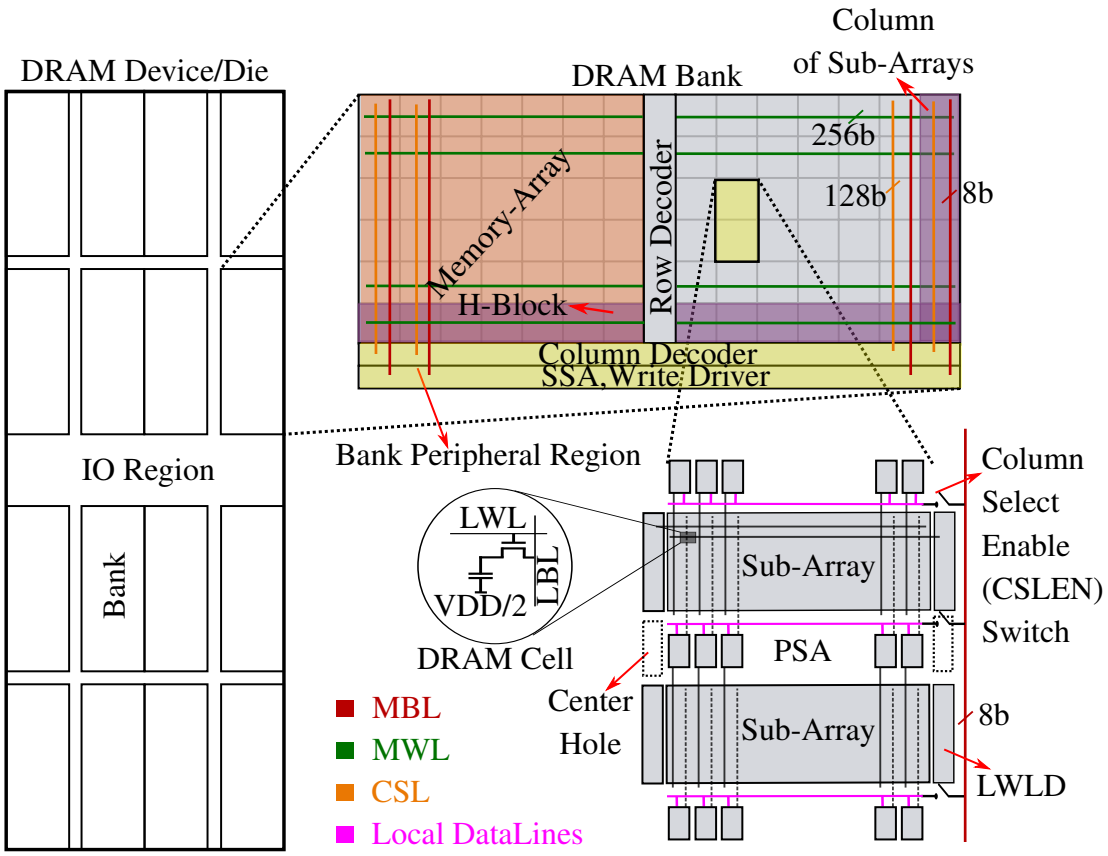

Figure 2.2: DRAM hierarchical architecture.

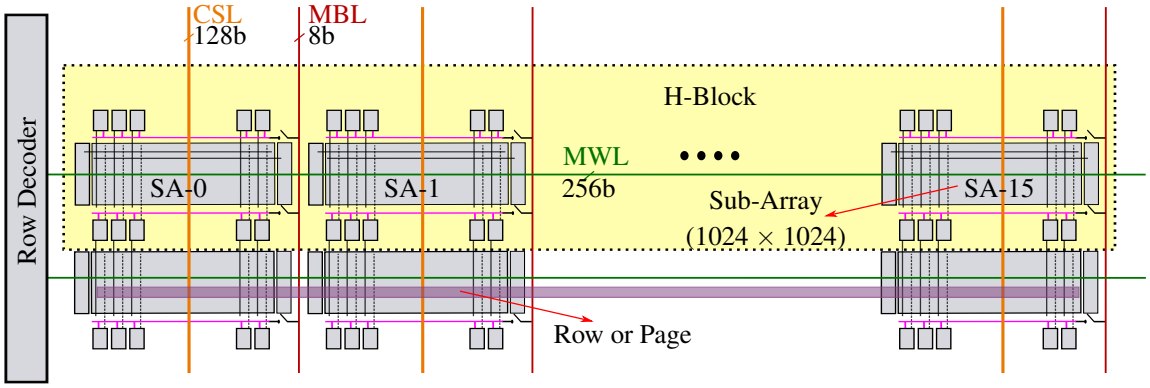

Figure 2.3: Detailed view of the H-Block.

the write operation, the data is transferred from the write-drivers to the desired columns of the page (i.e. Write driver to PSAs via the MBLs). This data access is in a fixed granularity that is defined by Equation 2.1 (e.g. $16 \times 8 = 128$-bit in DDR4), where Burst Length is abbreviated henceforth as BL. Consequently, the number of SSAs and write drivers integrated in the Bank peripheral region is also defined by Equation 2.1 (e.g. 128). The whole 128-bit

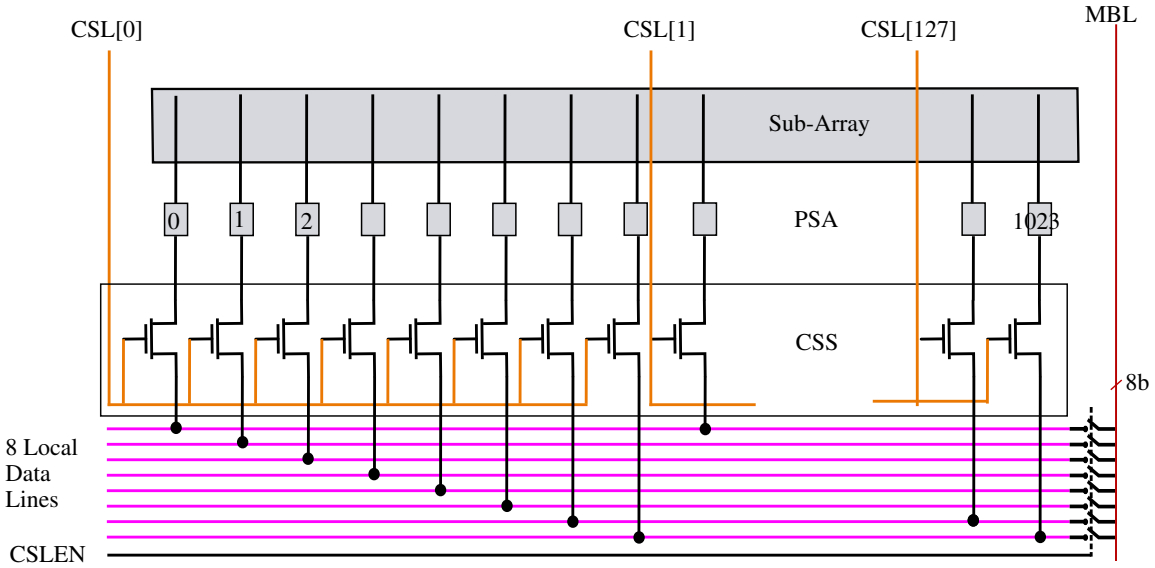

Figure 2.4: Circuit diagram for column selection from PSAs to MBLs. For ease of representation, all PSA outputs are shown on the same side of the Sub-Array, but in the actual design, the PSAs and CSS are interleaved on both sides of the Sub-Array.

data is not accessed from one single Sub-Array of an H-Block, but rather an equal number of bits are accessed from each Sub-Array of the H-Block (E.g. 4 or 8, or 16). For example, an H-Block with 16 Sub-Arrays and a data access granularity of 128-bit will access 8-bits from each Sub-Array. This value defines the number of MBLs per column of Sub-Arrays (e.g. 8-bit, refer Figure 2.2). Figure 2.4 shows the circuit diagram for accessing 8-bit data from the PSAs of each Sub-Array. This figure shows all the PSAs and the associated column circuits on the same side of the Sub-Array for ease of representation. However, in the actual design these blocks will be equally distributed on both the sides of Sub-Array as shown in Figure 2.2 and 2.3. The *Column Select Switches* (CSS) block of each Sub-Array is used to access the requested columns of the PSA data. Each CSL is connected to eight CSS and for a 1024 wide Sub-Array there are 128 CSLs per column of Sub-Arrays (refer Figure 2.2). To access 8-bit data from each Sub-Array of an H-Block, one CSL is enabled per column of Sub-Arrays. Similar to MWLs, CSLs and MBLs are also shared by all the Sub-Arrays within a column of Sub-Arrays. CSLEN signal of each H-Block is used to connect only one Sub-Array per column of Sub-Arrays to the MBLs. Once the 128-bit data (i.e. *Data Bus Inteface Width* $\times$ *Bursts Length*) is parallelly accessed by the SSA, it is forwarded to the IO region through a shared internal data bus (i.e. shared by all banks). This data is sequentially sent out through the high speed, dual data rate, external data bus interface (e.g. 16-bit $\times$ BL/2 clock cycles). Only the DRAM external interface frequency is operated at a high frequency and the rest of the device is operated at a lower frequency called core frequency. The frequency ratio between the external and core frequency is typically BL/2. In order to access the data from another row within the same bank, a precharge

(PRE) command has to be issued before activating a new row. The detailed description of precharge operation is presented in Section 2.1.2. In the following sections, I present the details of the Sub-Array block and the PSA block circuit.

$$\textit{Data Access Width within a Bank} = \textit{Data Bus Inteface Width} \times \textit{Bursts Length} \quad (2.1)$$

2.1.1 Sub-Array

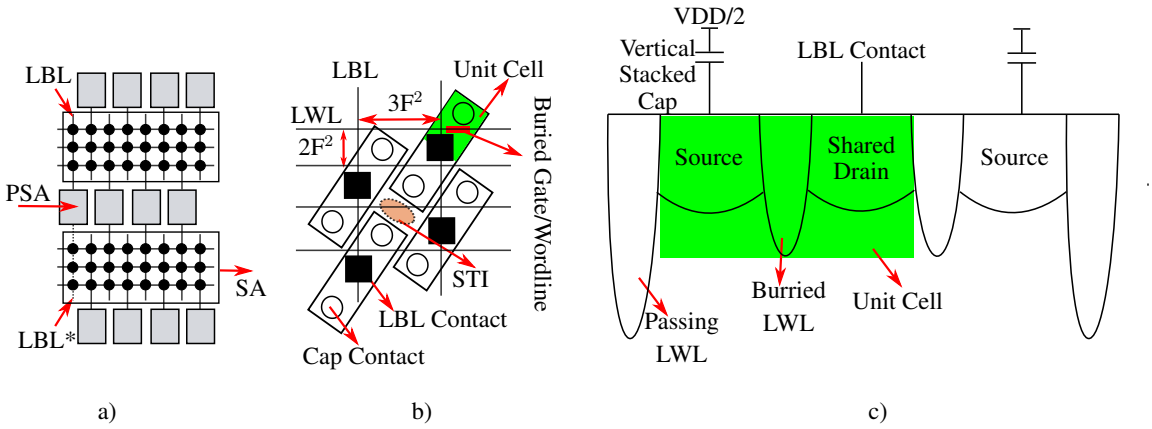

Figure 2.5: Sub-Array architecture and cross-section. a) Simple illustration of Sub-Array (SA), b) $6F^2$ Sub-Array architecture and slanted active area, c) Sub-Array cross-section showing buried gate, stacked cell capacitors, and shared drain terminal. Source [45–48]

The data in DRAM is stored as a charge in the DRAM cell. Each DRAM cell, also referred as 1T1C cell, consists of one access transistor and one storage capacitor. These DRAM cells are grouped as a two-dimensional array called as Sub-Array consisting of 1T1C cell, LWL, and LBL. The Sub-Arrays are designed using specialized buried gate *Recessed Channel Array Transistor* (RCAT) or *Sphere-shaped Recessed Channel Array Transistor* (SRCAT) or Saddle Fin cell access transistors, slanted active area, stacked cylindrical cell capacitors, shared drain terminal between the cell access transistors, fine bitline pitch, and buried wordlines [43–48] (Figure 2.5). The LBLs have a fine pitch of 52 nm for 18 nm DRAM technology [98], and DRAM vendors have further reduced this value in the latest technologies. In comparison to this value, the metal 1 pitch in a 12 nm FinFET based logic technology is 64 nm [99]. Hence, the Sub-Arrays are highly optimized for area, high density, and yield. Unlike Sub-Array the transistors in PSA, bank peripheral, and IO region are comparable to conventional planar transistors used in a logic technology [50] (i.e. not buried gate SRCAT transistors, like in Sub-Array) and are lagging behind the logic technology in terms of scaling the peripheral transistor [100]. This is due to multiple DRAM-process requirements such as low leakage, cost-effectiveness, and high

thermal resistance [50]. DRAM vendor's current roadmap is towards the High-k Metal Gate (HKMG) [50, 100] for the non-Sub-Array regions.

2.1.2 Primary Sense Amplifiers

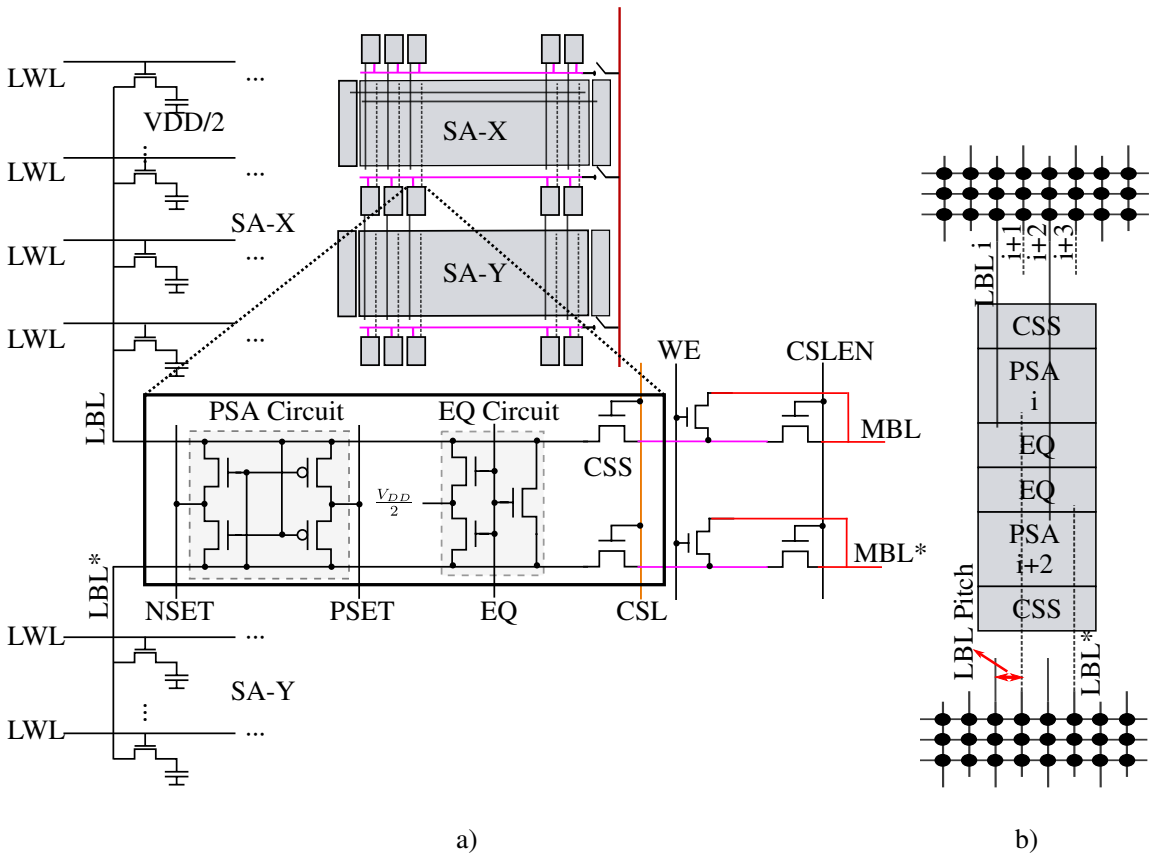

Figure 2.6: DRAM Primary Sense Amplifier (PSA) region circuit design and example floorplan. a) Circuit design [97], b) Example floorplan. In this figure Sub-Array is abbreviated as SA.

Figure 2.6.a shows the detailed circuit diagram the PSA region. This region consists of three main circuits viz. PSA circuit, Equalizer (EQ) circuit and CSS. PSA circuit is a cross-coupled differential amplifier that senses the voltage difference between LBL and reference LBL (LBL*). The LBL and LBL* are from the two vertically neighboring Sub-Arrays. For instance, when sensing the data from the Sub-Array X (SA-X, in Figure 2.6), the PSA circuit uses the LBL of vertically neighboring Sub-Array (SA-Y) as the reference line and vice versa. This array structure is called *Open Bitline* (OBL) array architecture and almost all recent commodity DRAMs employ this for high density. Prior to the sensing of cell charge, it is mandatory to maintain LBL and LBL* at an equilibrium voltage (i.e. VDD/2). The equalizer circuit is responsible for precharging the LBL and LBL* to VDD/2. CSS are

used to access the data to/from the PSAs from/to the SSAs via MBL and local datalines. Similar to LBL, the MBL and local datalines are also differential signals.

Figure 2.6.b shows the example floorplan of the PSA region (i.e., including the CSS and EQ). For the sake of simplicity, only a single side of the Sub-Array is shown, while the same blocks are present on either side of the Sub-Array. In commodity DRAMs, the typical width of each PSA is equal to four times the LBL pitch. To match the LBL pitch with PSA pitch, two design techniques are adapted. First, the PSAs of each Sub-Array are equally distributed and interleaved on both sides of the Sub-Array (Figure 2.6.a). Second, two consecutive PSAs on the same side are placed one after the other (Figure 2.6.b). These are applicable to the CSS and EQ circuits as well.

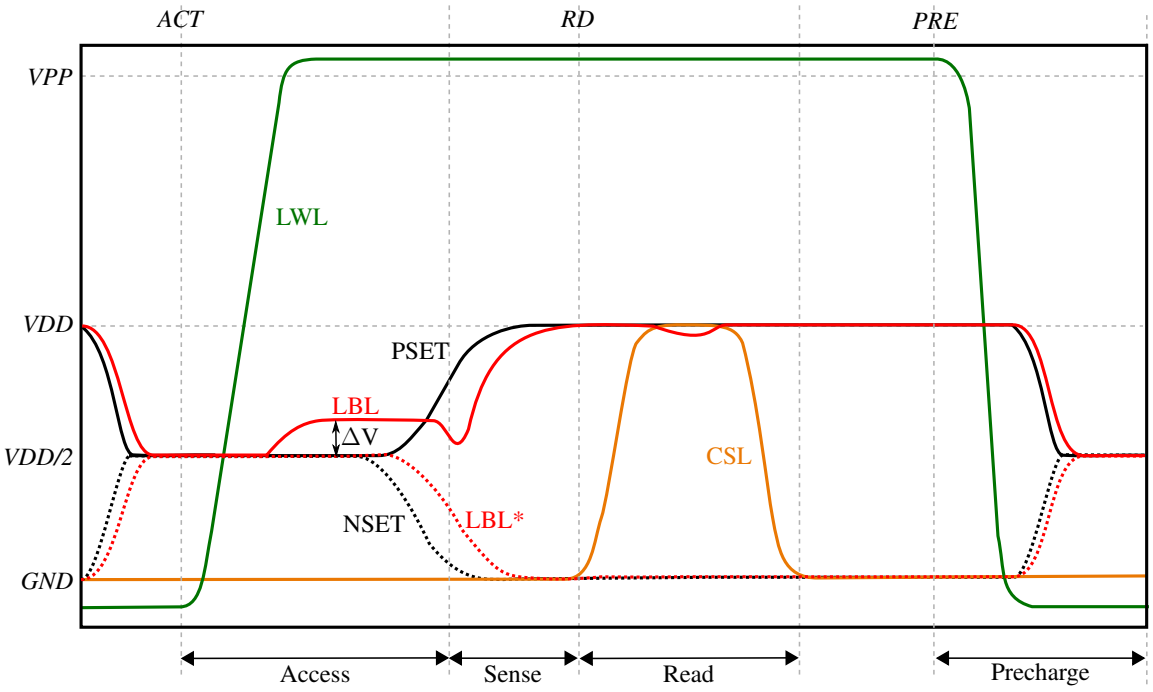

Figure 2.7: DRAM data sensing waveform [97] (Image courtesy [96]).

Figure 2.7 shows the PSA operation and waveform for sensing the cell charge. Initially, LBLs and supply lines of PSAs (i.e. NSET and PSET) are precharged to VDD/2 by the EQ circuit. The data access starts with the ACT command that enables the MWL and LWL of the requested DRAM row. The LWL is set to a voltage called VPP which should be at a voltage that is at least equal to VDD plus cell transistor threshold voltage (Vth). This is to ensure that a full logic-1 can be written to the cell capacitor through the NMOS cell transistor. After enabling LWL, the cell capacitor discharges the stored charge to the LBL. This is called as an access phase where charge sharing between cell capacitor (i.e. C_{cell}) and LBL wire capacitance C_{LBL} is happening. The charge sharing in the access phase changes the voltage of LBL from VDD/2 to VDD/2 + ΔV. The voltage ΔV is positive if the cell value (V_{cell}) is logic-1 and negative if the cell value is logic-0. The actual voltage

value of ΔV is given by Equation 2.2. The reference LBL continue to be at a precharged voltage VDD/2 in the access phase. The time duration of the access phase is based on the worst-case analysis.

$$\Delta V = V_{cell} \cdot \frac{C_{cell}}{(C_{cell} + C_{LBL})} \tag{2.2}$$

After the predefined access duration, the supply lines of PSA are enabled (i.e. NSET to GND and PSET to VDD) by the DRAM internal control circuit to start the sensing phase. Sensing is the process of amplifying the voltage difference (ΔV) between the LBL and LBL* to VDD and ground level. The cross-coupled NMOS latch of PSA is responsible for pulling one of the bitlines (i.e. either LBL or LBL*) towards the ground voltage and the PMOS latch will pull the other towards VDD. For example, if the ΔV is positive then the NMOS latch pulls LBL* towards the ground and the PMOS latch pulls LBL towards the VDD. At the end of the sensing phase, the voltage on LBL and LBL* reaches a stable value of VDD and ground, respectively, if the cell data is logic-1 and vice-versa if logic-0. In other words, the voltage value of LBL and LBL* corresponds to the true and complementary value of the cell data. The cell capacitor charge is also restored back to the full level in the background since the LBL continues to be connected to the cell capacitor. This phase is called write-back and happens in the background after the sensing phase. The total time for activating a row within a Sub-Array or a page within an H-Block is equal to the worst-case access phase time plus the worst-case sensing phase time. Post activating, CSLs are enabled to write or read the data to/from PSA. In order to activate a new row in the same bank, a precharge (PRE) command has to be issued to equalize the LBLs, NSET, and PSET to VDD/2. The DRAM cell has to be refreshed (REF) periodically to restore the leaked cell charge. As per JEDEC standards, all cells have to be refreshed once every 64ms at 80°C and 16 ms or 32 ms for higher temperatures. Internally, DRAM uses ACT and PRE commands pair to restore the cell charge during refresh. For every REF command, DRAM sequentially restores the charge of 4 to 16 pages per bank depending on the DRAM capacity. REF command is issued at an interval called t_{REFI}, which is typically 7.8 μs or 3.9 μs to restore all cells every 64 ms or 32 ms, respectively.

2.2 DRAM Protocol and Memory Controller

DRAM protocol defines the command sequence, state machine, and timing constraints that have to be respected for accessing the data from a DRAM device. The memory controller is a specialized block integrated between the processor core and external DRAM devices (Figure 2.8) that suffices the DRAM protocol specified in JEDEC standards. It internally has two sub-blocks 1) a digital frontend and 2) an analog Physical layer (PHY). The digital frontend converts the host memory request as per the DRAM command sequence or state

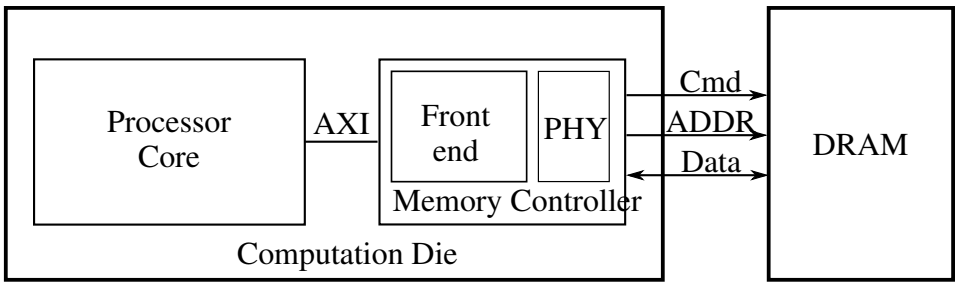

Figure 2.8: Computation system top-level overview.

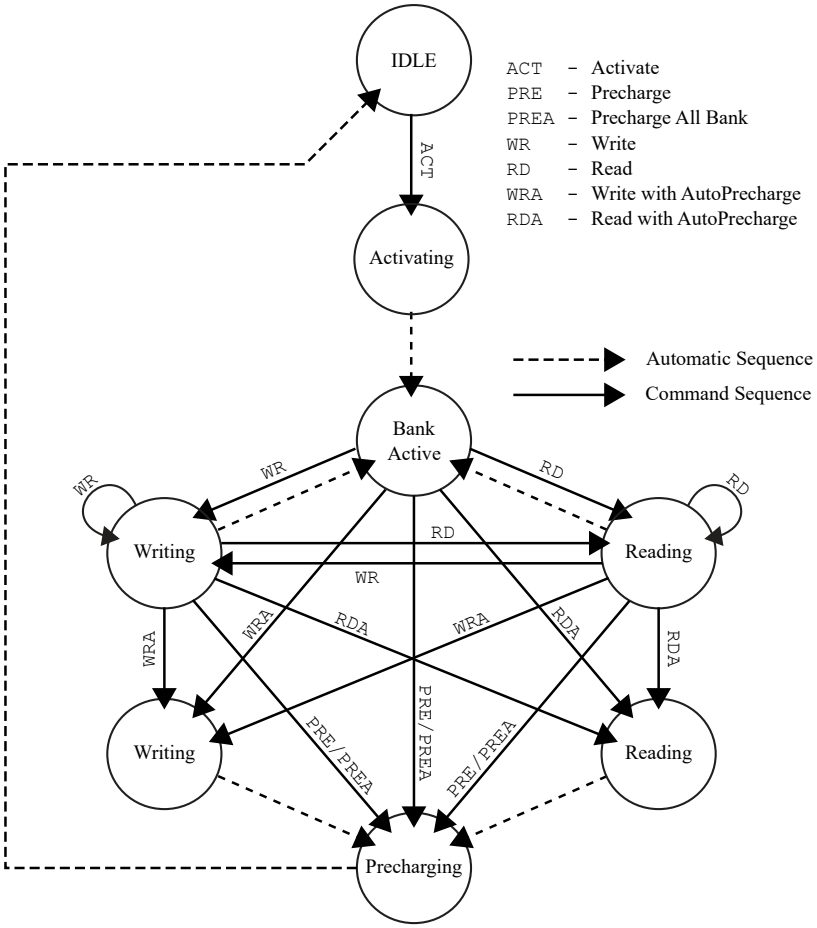

Figure 2.9: State machine of a single DRAM bank [51] (Image courtesy [96]).

machine and respects all the timing delays between the commands. The analog PHY consists of drivers and receivers for the high-speed external interface. The detailed architecture of the memory controller is proposed in Chapter 8. In this section, I brief the state machine and key timings that are generic to all DRAM types. JEDEC standards [51–57] provide a detailed description of DRAM protocol and timings for each type of DRAM.

Figure 2.9 shows the state machine of a single bank. As aforementioned in Section 2.1,

the DRAM bank is initially in an ideal state and an ACT command has to be issued prior to the column commands (i.e. read or write). Once the row or page is active, column commands can be issued consecutively to access the data from that specific page. In order to access the data from a new page within the same bank, the active page has to be closed using PRE command and a new row has to be opened (i.e. new ACT). This is referred to as **Page-Miss** and is one of the high latency-consuming operations in DRAM (e.g. 32.5 ns to 56.25 ns). The DRAM also supports the special commands called Read Auto-Precharge (RDA and) and Write Auto-Precharge (WRA). These commands inherently close the page after the data access without the memory controller explicitly sending the PRE command. These commands are used in the memory controller for closed page-policy where the page has to be immediately closed after one data access, while the standard RD and WR are used in open-page policy that continues to keep the page open until the page-miss is encounter. The advanced memory controllers use the combination of these policies called adaptive page-policy [101] in order to reduce the page-miss penalty. Strict timing has to be maintained between each pair of commands or for each transaction of the state machine. Table 2.1 summarizes the timing constraints for consecutive commands to the same the bank. For instance, time t_{RCD} (i.e. access time plus sensing time) has to be awaited prior to sending the RD or WR commands after ACT. Figure 2.10 shows the example timing diagram for single bank access with various timing delays. It also highlights the page-miss latency. The page-miss latency in a DDR4 device operated at 800 MHz vary from 32.5 ns (i.e. page-miss RD to RD latency = $t_{RTP} + t_{RP} + t_{RCD}$) to 56.25 ns (i.e. page-miss WR to WR latency = $t_{WL} + BL/2 + t_{WR} + t_{RP} + t_{RCD}$).

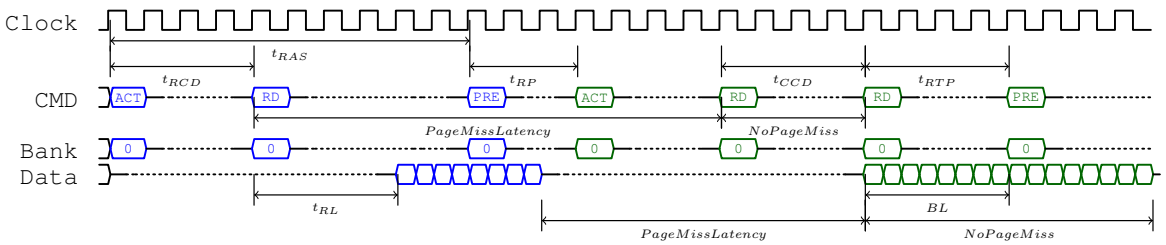

Figure 2.10: Example waveform for DRAM timing delays within a bank.

Relative to same-bank timing constraints, the inter-bank timing constraints are very few. Table 2.2 summarizes the timing constraints for consecutive commands to different banks. One of the new timing constraints for interbank commands is t_{RRD} (i.e. ACT-ACT), which is required to avoid power spikes or large voltage dips on the power lines due to concurrent ACT. The new DRAM types like LPDDR5 also have a similar timing restriction for interbank PRE commands. The column-to-column command timing constraints are the same for both interbank and interbank. This is due to the shared internal bus between the bank peripheral region and the IO circuit. DRAM also provides a special command called PREA (Precharge-All) that is used to concurrently close the active pages of all the banks. This command is very useful prior to the refresh operation that requires all banks to be

Table 2.1: Timing delays within a bank.

Type	Cmd	Timing	Definition
Row-Row	ACT-PRE	t_{RAS}	*Row Access Strobe*: Minimum time for which a row or a page has to be active
	PRE-ACT	t_{RP}	*Row Precharge*: Minimum precharge duration
	ACT-ACT	t_{RC}	*Row Cycle*: Minimum duration between to two consecutive ACT assuming PRE in between
Col-Col	RD-RD WR-WR	t_{CCD}	*Column to Column Delay*: Minimum time two consicutive RDs or WRs. tCCD = BL/2 $\times$ tCK, where tCK is Interface Clk period and BL is Burst Length. tCCD also defines the internal core clock of DRAM
	RD Cmd to RD Data	t_{CL} or t_{RL}	*Column Latency or Read Latency*: Minimum wait time for RD data to be available on DRAM Data bus after receiving RD command.
	WR Cmd to WR Data	t_{CWL} or t_{WL}	*Column Write Latency or Write Latency*: Minimum wait time for WR data to be sent by memory controller after sending WR command.
	RD-WR	-	$RL + BL/2 + 2tCK - WL$
	WR-RD	t_{WTR}	*Write to READ*: Minimum time between the last write data burst to RD cmd. Delay between WR cmd to RD cmd = $t_{WL} + BL/2 + t_{WTR}$
Row-Col & Col-Row	ACT-RD ACT-WR ACT-RDA ACT-WRA	t_{RCD}	*Row to Column Delay*: Minimum wait time after ACT command to send column commands (i.e. RD or WR)
	RD-PRE	t_{RTP}	*Read to Precharge*: Minimum time RD and PRE
	WR-PRE	t_{WR}	*Write Recovery*: Minimum time between the last write data burst to PRE cmd. It is a duration required to overwrite the cell value after receiving all data burst Delay between WR cmd to PRE cmd = $t_{WL} + BL/2 + t_{WR}$
	RDA-ACT	-	$tRTP + tRP$
	WRA-ACT	-	$WL + BL/2 + tWR + tRP$
	RDA-PRE WRA-PRE	-	Redundant

precharged. PREA introduces four additional timing constraints (i.e. ACT-PREA, PREA-ACT, RDA-PREA, WRA-PREA) whose time delay is the same as within the bank delay. There are no timing delays between the other row-to-row, row-to-column, and column-to-row commands (e.g. ACT to PRE of different banks), which is effectively due to bank parallelism. In addition to these timings, Table 2.2 also presents the refresh-related tim-

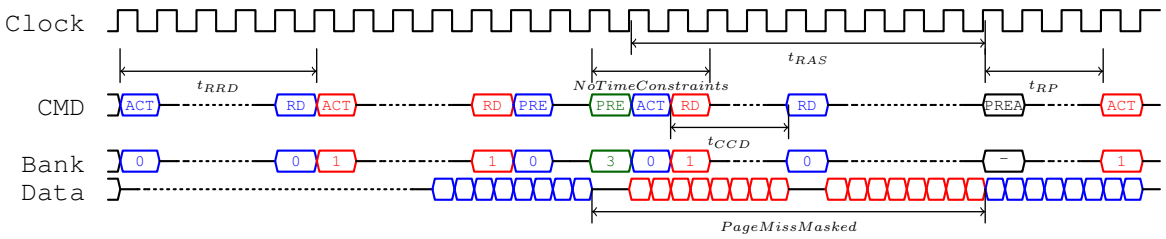

Figure 2.11: Example waveform for DRAM timing delays between different banks.

ings. The memory controller has to stall the data access request every tREFI to perform the refresh operation. DRAM JEDEC standards also present the timing for other operations such as mode-register, low-power, and self-refresh. These operations are to be performed only when the DRAM is in an ideal state. These timings are not of major relevance to this thesis, as all my architectural investigations are related to DRAM active mode. Figure 2.11 shows the example timing diagram for multi-bank access. This figure also shows the advantage of bank parallelism to mask the page-miss penalty i.e. while bank 0 is stalled due to a page-miss, the data is accessed from bank 1 which already had an activated/open row.

Table 2.2: Timing delays for different banks.

Type	Cmd	Timing	Definition
Row-Row	ACT-ACT	t_{RRD}	*Row to Row Delay*: Minimum duration between to two consecutive ACTs of different banks
Col-Col	Same as Table 2.1		
Pre-All	ACT-PREA	t_{RAS}	Same as ACT-PRE in a single bank (Table 2.1)
	PREA-ACT	t_{RP}	Same as PRE-ACT in a single bank (Table 2.1)
	RDA-PREA	t_{RTP}	Same as RD-PRE in a single bank (Table 2.1)
	WRA-PREA	t_{WR}	Same as WR-PRE in a single bank (Table 2.1) WRA cmd to PREA cmd = $WL + BL/2 + tWR$
Other Row-Row, Row-Col & Col-Row	ACT-PRE PRE-ACT ACT-RD ACT-WR ACT-RDA ACT-WRA RD-PRE WR-PRE RDA-ACT WRA-ACT RDA-PRE WRA-PRE	-	No timing constraints - Only 1 tCK delay required
Refresh	REF-REF	t_{REFI}	*Refresh Interval*: Time Interval between two REF cmds
	REF-ACT	t_{RFC}	*Refresh Cycle*: Refresh time duration or minimum time between REF to ACT
	PREA-REF	t_{RP}	Same as PRE-ACT in a single bank (Table 2.1)

Chapter 3

Design Space Exploration

As presented in Chapter 1, the primary focus of this thesis is to investigate novel DRAM-PIM architectures that fulfill the three aforementioned research goals i.e., *Goal-1*) avoid Sub-Array modification, compatibility with commodity DRAM, and Dual-Mode, *Goal-2*) integrate computation units near the Sub-Array (i.e. in PSA region) to exploit high internal parallelism and low data movement, and *Goal-3*) architectural scalability for DNN training and unified interface for all future generations of DRAM-PIM. For these goals to be achieved, it is imperative to first explore the DRAM-PIM design space and investigate the following factors: 1) computation unit area vs energy-efficiency gains in various locations within the commodity DRAM architecture, 2) area constraints in various regions of the commodity DRAM architecture, 3) impact of modifying the existing commodity DRAM designs on memory-density and Dual-Mode feature of DRAM-PIM, 4) data format vs DNN accuracy vs computation unit area tradeoff, and 5) memory controller and interface modifications required for external control or programming of PIM. Figure 3.1 shows the design space of DRAM-PIM and the three design dimensions that are discussed in detail in this chapter. This chapter also briefs the state-of-the-art DRAM-PIM architectures and their design choices. To the best of my knowledge, I review all the available state-of-the-art DRAM-PIM architecture publications [20, 28–38, 86–89] with the emphasis on DNN for this design space assessment. All the findings presented in this chapter laid the groundwork for my investigations on novel DRAM-PIM architectures and architectural approaches proposed in Chapters 4, 5 and 6.

3.1 Design Dimension 1: Computation Unit Location

In this section, I present the area constraints and impact of integrating the computation units in various possible locations within the commodity DRAM. The four key locations to integrate the computation units in a commodity DRAM architecture are 1) Inside Sub-Array, 2) Near Sub-Array or Inside PSA region, 3) Bank peripheral region and 4) IO region.

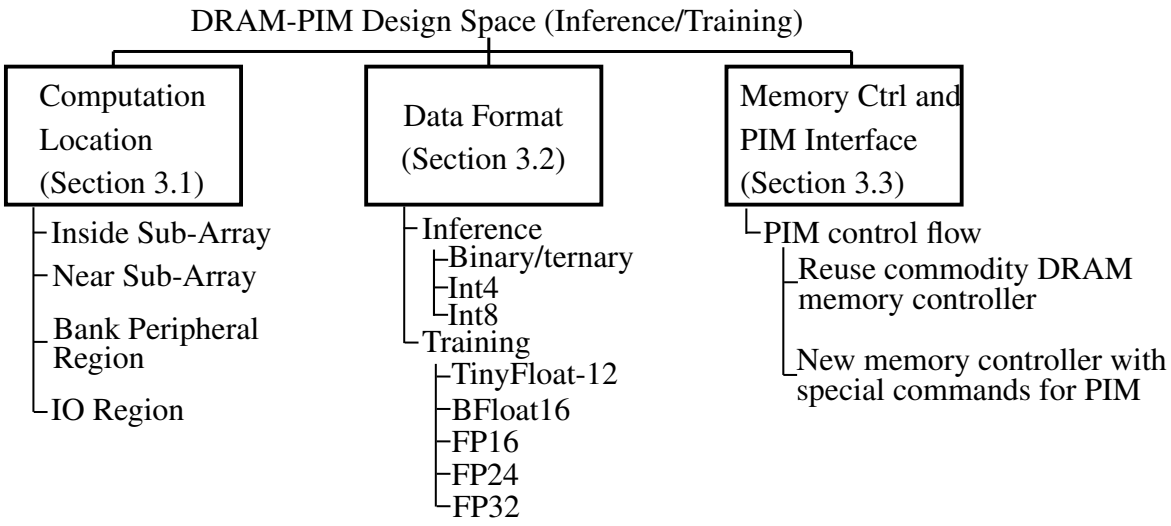

Figure 3.1: Three main dimensions of DRAM-PIM design space.

3.1.1 Inside Sub-Array Computation

As mentioned in Chapter 1, PIM architectures are designed to exploit the high internal data parallelism of memory devices and low data movement energy. In this context, the highest data parallelism and the lowest data movement energy within a DRAM is available at a Sub-Array level. For example, the amount of parallel data available at Sub-Array level is as high as 1024 bits/Sub-Array (or 2 KB per page), while at a bank peripheral region or IO region, this value is reduced to 128 or 256 bits per cycle. Hence, Sub-Array level based DRAM-PIM architectures have the potential to achieve maximum throughput and energy efficiency. The Ambit [28] architecture was one of the first DRAM-PIM publications to propose ”Inside Sub-Array” approach to compute bulk logic operations like AND, OR and NOT. The basic operation principle of Ambit is to simultaneously activate three rows within a Sub-Array to implement a majority function (i.e. row OP, row A and row B in Figure 3.2.a and Figure 3.2.b). This majority function is employed to perform bulk AND or OR operation during the DRAM activation process (i.e. cell charge sensing). Figure 3.2.b shows the example for AND operation, where the storage value of OP cell defines the operation type (i.e. OP = 0 for AND operation and OP = 1 for OR operation). For functional completeness, Ambit architecture also proposes to implement NOT operation using a special row of *Dual Contact Cells* (DCCs) within a Sub-Array. Figure 3.2.c shows the cell structure of a DCC. This cell has two access transistors, where one is connected between the cell capacitor and the bitline (LBL) and the other is connected between the cell capacitor and reference bitline (LBL*). The LBL and LBL* are tied to the true and complementary value after the sensing operation by the PSA. Hence, connecting both LBL and LBL* to the cell capacitor enables the storage of either true or complementary value of a AND or OR operation output computed using the majority function. It is important to note that the DRAM Sub-Arrays are designed using a single access-transistor plus a single cell-capacitor (i.e. 1T1C cell) and do

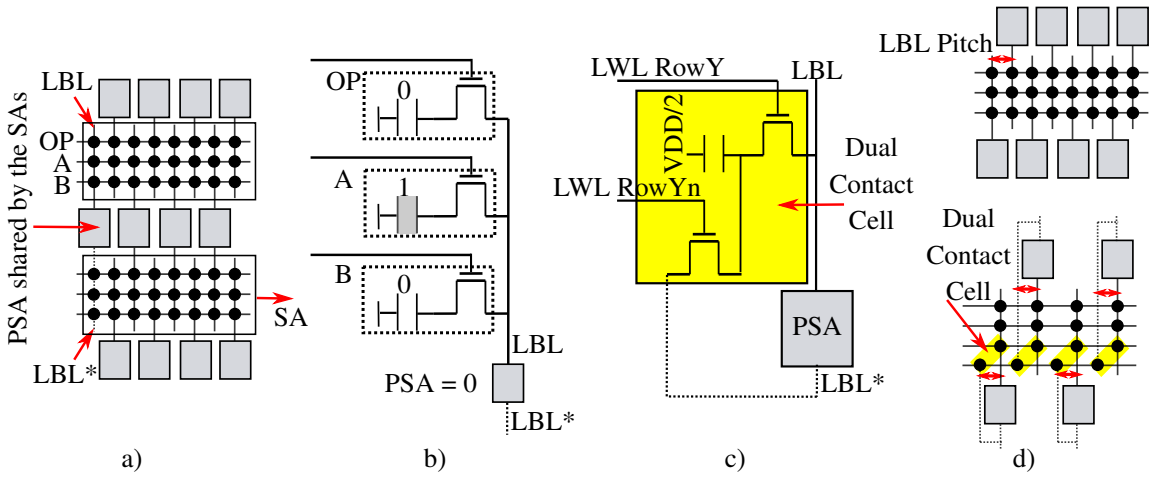

Figure 3.2: Ambit implementation. a,b) Basic operation principle of Ambit, c) Dual Contact Cell (DCC) proposed by Ambit to implement NOT operation, d) Comparison of commodity DRAM Sub-Array vs. modifications required to Sub-Array for connecting LBL* to the DCC.

not explicitly have DCCs. This leads to the modification of the commodity DRAM Sub-Array, which is highly optimized for density and yield. Any modification to the Sub-Array design has to be avoided due to the reasons previously described in Section 1.2.1.

Almost all recent commodity DRAM architectures employ *Open Bitline* (OBL) array architecture, as it offers high density. In OBL array architecture, the LBL and LBL* correspond to two different Sub-arrays (i.e. vertical neighbors) as shown in Figure 3.2.a and Figure 3.3.a. It is important to note that the LBLs have a fine pitch of 52 nm for 18 nm DRAM technology [98], and DRAM vendors have further reduced this value in latest technologies. In comparison to this value, the metal 1 pitch in a 12 nm FinFET based logic technology is 64 nm [99]. The LBL pitch in DRAM technology is precisely adjusted for minimal crosstalk and maximum density. Hence, the new feedback connection of LBL* to a DCC will have a direct impact on the memory density. Figure 3.2.d shows one of the example modifications that reduced the memory density of Sub-Array for LBL* connections. The spacing of LBL to route the LBL* from the PSA to a DCC of a given Sub-Array has to be doubled so as to not violate the minimum pitch.

The Ambit architecture is adapted by publications like DrACC [30], LACC [31], and DRISA [29] to implement DRAM-PIM architectures for DNN inference. DrACC and LACC further proposed to include multiple other special rows to perform XOR, NOT and shift operations within a Sub-Array. The special rows of DrACC and LACC not only required the new connection between the LBL* and DCC but also included connections between the rows within the Sub-Array, which affect the density and regular structure of the Sub-Array. On the other hand, DRISA proposed to integrate CMOS logic gates for NOT operation in the PSA region rather than modifying the Sub-Array (refer to Section 3.1.2 for more details). The Ambit architecture is also adapted in Max-PIM [33], a DRAM-PIM

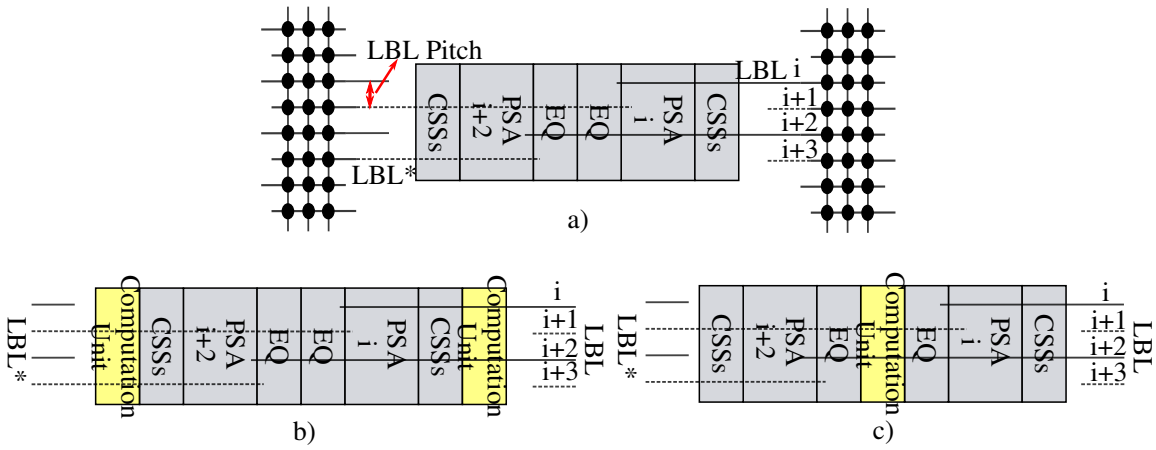

Figure 3.3: PSA floorplan of conventional vs. with computation unit for PIM. a) Conventional PSA region, b/c) example computation unit integration in the PSA region, where metal tracks have to be reserved in the computation unit for LBL and LBL* to be connected to respective PSA. All three floorplan figures are rotated by 90° in a clockwise direction due to space considerations.

architecture designed for a big data application that searches for minimum and maximum value in a big array. Similar to DRISA, the authors of Max-PIM also proposed to integrate logic gates in the PSA region (i.e. XNOR). However, this architecture still required the modification of the Sub-Array design as it relies on DCCs for other logic operations. Hence, I aim to explore the design approaches that avoid modification of commodity DRAM Sub-Array.

3.1.2 Near Sub-Array Computation

The ”Near Sub-Array” approach integrates computation units in the PSA region without modifying the Sub-Array design. The data parallelism and data movement energy for this approach is similar to the ”Inside Sub-Array” approach (i.e. highest and lowest, respectively). However, unlike Sub-Array the PSA region's physical design is compatible with the standard CMOS logic design and the transistors of this region are comparable to conventional planar transistors (refer Chapter 2, Section 2.1.1). This enables the integration of CMOS-based logic gates in PSA region, but with certain *area constraints* that are outlined in this paragraph. The new computation units have to be integrated near the Sub-Array without violating the OBL structure of the commodity DRAM architecture (refer Figure 3.3.a). Additionally, it is also important to not modify the existing circuits of the PSA region as it might have a direct impact on the data sensing, which is designed to sense a minute charge stored in the cell capacitors (e.g. cell capacitance value of 10 nm class DRAM is 10 fF [42]). Hence, PIM architectures with ”Near Sub-Array” approach have to ensure the following physical criteria: First, the integration of new computation units has to retain the balanced wire load/capacitance at the PSA inputs, as PSA is a differential

amplifier. This can be ensured by adopting one of the two possible floorplans presented in Figure 3.3.b and c. Please note, the floorplan shown in Figure 3.3 is for an example scenario where a PSA's width is equal to four LBL pitch. Second, as shown in Figure 3.3, the new computation units integrated near the Sub-Array have to reserve metal lines to allow LBLs/LBLs* to be connected to the respective PSAs (i.e. LBLs/LBLs* pass through computation unit). This makes the routing within the new computation unit extremely difficult, especially considering the fact that the DRAM process has a very limited number of metal layers of three or four. Third, the new computation units increase the length of the LBLs, which increases the charge sensing time and in the worst case impacts the correct sensing of data if the height of the computation units is beyond the design considerations of the PSA. This imposes height restriction to the new computation units in order to avoid the redesign of the PSA. Overall, these aforementioned factors impose area restrictions to the new computation units in the PSA region.

The compute kernel of DNN is Multiply Accumulate (MAC) unit, which requires area and power-intensive multipliers. The integration of these area-intensive MAC units (8-bit or higher precision) have to be without violating the three aforementioned area constraints of the PSA region. An evaluation done by the authors of [39] showed that the integration of 8-bit MAC units near each Sub-Array results in $2\times$ area overhead or reduces the memory density by half. The "Near Sub-Array" approach is employed in the following DRAM-PIM architectures: DRISA [29] and SCOPE [32]. These two architectures addressed the area restrictions of the PSA region by redesigning the DRAM bank with only 16 Sub-Arrays, as opposed to memory density optimized commodity DRAM bank structure with 256 to 1024 Sub-arrays per bank. To increase the throughput of DRAM-PIM, the number of banks per die is increased to 1024, while the commodity counterparts have a maximum of 32 banks. Consequently, [32] presented a total chip area that is $4.4\times$ larger than commodity 8Gb DRAM. Additionally, [31] reported that DRISA and SCOPE consume extremely high power of 98 W per 2 Gb die and 176 W per 8 Gb die, respectively. These values are way higher than the typical power consumption of commodity DRAM die (e.g. 1-2 W) or its cost-effective packaging. The design approach of DRISA and SCOPE directly contradicts the goal of being compatible with commodity DRAM architecture and sufficing Dual-Mode feature mentioned in Goal 1 (Section 1.2.1).

Challenge 1: *Seamless integration of computation units like multipliers and adders (i.e. $>$8-bit) in the PSA region of commodity DRAM architecture without modifying dense OBL structure.*

3.1.3 Bank Peripheral Region Computation

In this approach, the computation units are integrated at the output of a bank without modifying the core design of DRAM bank. The DRAM-PIM architectures such as Samsung's McDRAM [39, 40] and FIMDRAM [34], Hynix's Newton [36] and AiM [37],

[86–88], [89], and UPMEM-PIM [35] place computation units in the bank peripheral region. The main advantage of the bank peripheral region is its relaxed area and physical restrictions compared to the PSA and Sub-Array regions, which enables the integration of area-intensive units. However, the available data bits per cycle at the bank output (i.e., prefetch) are reduced to a maximum of 256 bits (i.e., defined by the number of MBLs). In comparison, the data parallelism at the PSA region is up to 2 KB per page. This limits the number of parallel computation units integrated at the output of the bank to a value defined by Equation 3.1, which further limits the throughput of a bank. Increasing the number of MBLs per bank in an area-optimized commodity DRAM bank design is difficult as all the wires including the power lines are tightly packed due to the limited number of metal layers. Additionally, the data prefetch size per bank is predefined by the JEDEC specification for a given DRAM type (e.g. LPDDR, DDR, GDDR, HBM etc.). Increasing these values for the sake of higher throughput violates the JEDEC specification in the memory-mode operation of DRAM-PIM, and the DRAM bank has to be redesigned to support different prefetch sizes for computation and memory mode.

$$Num_of_Units_per_Bank = Num_of_MBLs_per_Bank / Data_Word_Size \quad (3.1)$$

The peak computation throughput is also alternatively increased by increasing the number of parallel operating banks. However, JEDEC specification has predefined the number of banks for a given DRAM type. Additionally, increasing the number of banks in PIM architectures also increases the computation overhead and reduces the memory density. For example, Samsung's FIMDRAM with 16 banks reduces the density per die to half of the commodity DRAM counterpart with the same number of banks and the same technology. This value would further reduce upon increasing the number of banks. Furthermore, the area overhead is expected to increase for future DRAM-PIMs, as the computational capability and functionalities are increased to support a wide sector of applications. Hence, this compromises the density and capability to operate as a high-capacity main memory. Hence, irrespective of the placement location, it is very important to maintain low area overhead with the maximum worst-case upper limit density loss of 25% to 50% compared to commodity counterparts of the same type and technology.

Another way to increase the throughput in this approach is to reduce the data access latency within the bank. The data access latency within the bank (i.e., PSA to SSA) defines the peak operating frequency of computation units, which mainly depends on the bank height, wire capacitance/resistance, operating voltage, and technology node. Please note, memory vendors do not disclose the internal data access latency or internal data access energy consumption values. One of the major factors that impact the data access latency is the DRAM bank height which varies for different DRAM types. Table 3.1 compares the bank height of various recent DRAM types, which is proportional to the internal data access latency and energy consumption. One of the LPDDR4-based DRAM-PIM, McDRAM, operated the computation units at 250 MHz due to a data access latency of 4 ns (i.e. long bank height), although by design the computation unit is capable of operating at up to

Table 3.1: Bank height comparison for various DRAM types.

DRAM Type	Technology[a]	Die Capacity (in Gb)	Die Dimention $W \times H$ (in mm)	Bank Height[b] (in mm)
DDR4 [98]	1Y ($\approx 17\,nm$)	8	8.08×4.19	1.83
DDR5 [103]	1Y	16	9.2×8.2	1.92
LPDDR4 [98]	1Y	8	4.28×8.03	3.55
LPDDR5 [41]	10 nm 2^{nd} Gen	12	4.4×12.44	2.65
LPDDR5 [104]	10 nm 3^{nd} Gen	16	5.609×10.86	3.122
HBM2 [105]	20 nm	8	12×8	0.762
HBM2E [106]	10 nm 2^{nd} Gen	16	11×10	0.904
FIMDRAM [20, 34] (HBM2)	20 nm	4	10.28×8.2	**0.49**
AiM [37] (GDDR6)	1Y($\approx 17nm$)	4	8.4×4	**0.532**

[a] Some DRAM vendors specify the 10 nm class DRAM technology as 1Z(≈14 nm), 1Y(≈17 nm), and 20 nm class as 2X (≈28 nm), 2Y(≈25 nm), 2Z(≈22 nm).
[b] Bank height is proportional to data access latency within the bank

1 GHz. Many prior DRAM-PIM publications (e.g. McDRAM, FIMDRAM) operated their computation units in the rage of 200 MHz to 300 MHz due to data access latency. On the other hand, Hynix-AiM shows that the data access latency can be reduced to as low as 1 ns for smaller bank heights (like in GDDR6) and operate computation units at 1 GHz. Additionally, the reduced bank height also reduces the energy consumption for accessing data within the bank as compared to other DRAM types with large bank heights but a similar technology node. As shown in the table, the choice of DRAM type by FIMDRAM and Hynix-AiM is associated with a low bank height either to reduce latency and to increase energy efficiency within a bank. Achieving a similar throughput and energy efficiency in other DRAM types like LPDDR5, DDR5, etc. with large bank heights is challenging. As per Samsung, the LPDDR5 and DDR5 are the target DRAM types for DRAM-PIMs in edge applications [102]. The large bank heights also provide an additional advantage of maintaining the area overhead per bank to low values. For example, in comparison to their commodity counterpart, the LPDDR4-based McDRAM V1 [39], and V2 [40] showed a low area overhead of 4.5% and 5.4%, respectively, with large banks. Hence, maximizing the memory cell density, but compromising on throughput and energy efficiency.

***Challenge 2**: Increasing the computation throughput without decreasing the density or violating JEDEC specifications.*

***Challenge 3**: Achieving low data access energy within the bank, especially for DRAM types with large bank heights.*

3.1.4 IO Region Computation

This approach integrates all or a portion of the computation units in the IO region of the DRAM, as it has the fewest physical restrictions. Figure 3.4.a compares the data access energies of the different regions of a commodity 20 nm HBM2 DRAM. The energy required to transfer the data between the bank and IO region is $3\times$ higher than the energy within a bank. Architectures like Hynix-AiM proposed to place only the large shared global buffers in this region, as this region has the least area constraints. The shared buffers also reduce the area overhead of computation units near the banks. Figure 3.4.b shows one of the architectural configurations of Hynix-AiM, where one of the operands of the computation units is received from the shared buffer while the other is from the respective bank. In this configuration, the transfer of data from the shared buffer in the IO region and the computation unit in the bank peripheral region consumes a major portion of energy per computation operation. Alternatively, receiving the data from a neighboring bank (Figure 3.4.c) results in disabling half of the computation units and reduced throughput. The shared buffer is also used to transfer data between the banks, which is vital for DNNs such as Convolution Neural Networks, where one input feature map is convoluted with various kernels. This results in high energy consumption for the exchange of data between banks. On the contrary, architectures like FIMDRAM do not support bank-to-bank communication. A few architectures like GradPIM [38] place a single block of computation per set of banks (e.g. four) called as Bank Group. This approach is in between the "Bank Peripheral Region" and "IO Region" approaches that encounter similar drawbacks as IO region.

Challenge 4: *Enabling low energy data exchange between the banks, and between banks and buffers.*

3.2 Design Dimension 2: Computation Unit Data Format

Recently, DNN accelerators are exploring a wide range of low-bit-width computation (i.e., ≤ 8 bit) for inference and training (refer to Figure 3.5). The low bit computation in conventional accelerators reduces the area and power consumption per MAC unit and increases the overall throughput. Figure 3.5 shows various data formats employed in DNN accelerators and their respective accuracy degradation compared to 32-bit floating point format for different networks and datasets. DRAM-PIMs have also explored different data formats such as binary/ternary [29,30], Int8 [31,32,39], and floating point [34,36,37]. Reducing the data width results in higher throughput due to an increase in the number of data words fetched per data access within the bank (i.e. an increase of ratio $\#MBLs/DataWordSize$). It also reduces the overall data access energy within the bank as fewer data accesses are required to fetch the given total number of data words. For a #MBL size of 128 bits, an 8-bit format reduced the data access energy by 50% and increased the throughput by $2\times$ compared to 16-bit formats. Prior DRAM-PIM architectures [31, 32, 39] have adapted low bit-width

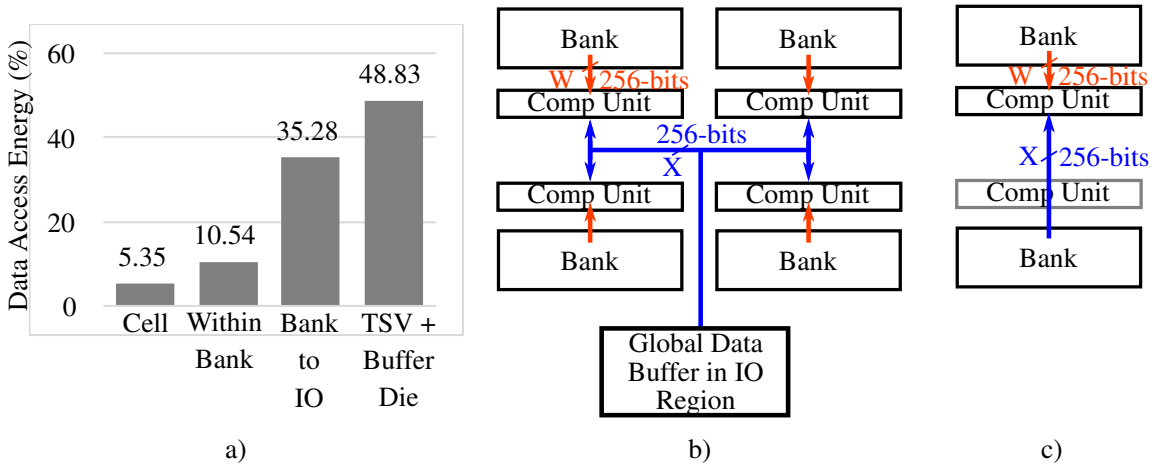

Figure 3.4: High energy consumption for transferring data to and from IO region. a) Energy breakdown of read operation in 20 nm standard HBM2 device [20], b) Hynix-AiM distributing input feature map data to all banks from a global buffer in IO region: High energy consumption, c) Hynix-AIM fetching both input feature map data and weight from the bank: throughput reduction by half [37].

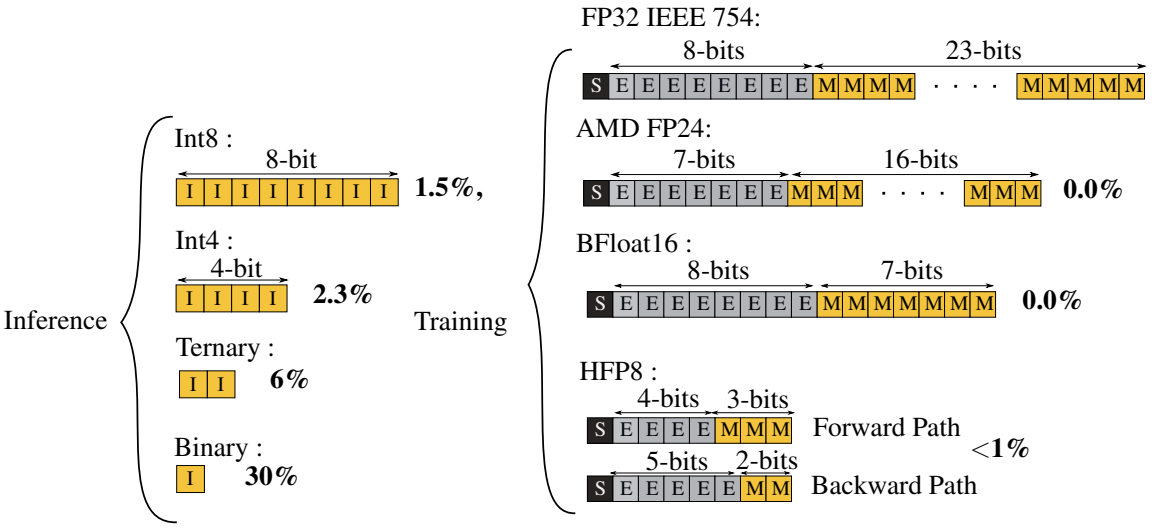

Figure 3.5: Data formats employed in state-of-the-art standard DNN accelerators for inference and training and their accuracy degradation compared to FP32. Accuracy degradation values are extracted from the sources [107–110] for inference and from [111, 112] for training.

(e.g. 8-bit) data type and successfully demonstrate for DNN inference tasks with minor low accuracy loss. However, DRAM-PIMs such as [34, 37] that target DNN training are mostly confined to a 16-bit floating-point format called Brain-Float (BFloat16) that achieve similar accuracy as 32-bit floating-point format [111]. Hence, it is also very important to explore the low bit-width training for DRAM-PIMs to further increase the energy-efficiency.

One of the recent trends in the context of DNN training is to employ the 8-bit floating-point data format, which is proposed in [112–114] for Non-PIM computation platforms. The main challenge of using the 8-bit floating-point data format for DNN training tasks is its limited dynamic range compared to the 32-bit floating-point data format. The aforementioned research works address this issue by employing a suitable scaling factor to adjust the data range of the 8-bit format. However, these publications only demonstrate the proof of concept but do not discuss the detailed technique to dynamically adjust the scaling factor within a hardware accelerator. For real-world applications, where the data sets and DNN models are evolving continuously, the scaling factor should be adapted dynamically on-the-fly within the accelerator. As a result, a new online statistical analysis methodology has to be adopted in order to successfully employ an 8-bit format for DNN training either in DRAM-PIM or any conventional accelerator. Especially for DRAM-PIM, the methodology has to additionally consider all the area constraints within a commodity DRAM architecture.

Challenge 5: *Proposing a new online statistical analysis methodology to adapt an 8-bit data format for DNN training in DRAM-PIM without any accuracy loss.*

3.3 Design Dimension 3: Memory Controller

A conventional DRAM sub-system consists of a block called a memory controller that converts the host processor's read and write requests into a DRAM-specific protocol and timings. In the context of DRAM-PIM, the memory controller plays a vital role in external control and programming. In this section, I investigate the state-of-the-art approaches for external control and programming of a DRAM-PIM. FIMDRAM and UPMEM-PIM architectures are designed with the aim to be a drop-in replacement for JEDEC-compliant commodity DRAM. The external control of these architectures reuses the existing commodity DRAM memory controller and interface. The same conventional DRAM commands (e.g. Activate, Write, Read) are employed for both memory mode and compute mode. In the memory mode, the computation instructions are offloaded to the PIM engines using the write command to a dedicated address. The DRAM-PIM device is switched from memory mode to computation mode after the desired program is loaded. The change to computation mode is done either by activating a reserved row (in FIMDRAM) or by sending a special sequence of commands (e.g., sequence of ACTs) to the DRAM (in UPMEM-PIM) to start the computation. In the case of FIMDRAM, during the computation operations, the conventional write and read commands are used to increment the internal program counter. After finishing the program execution, the DRAM-PIM is switched back to memory mode to access the computation results and load new data for the next round of computations. One of the major drawbacks of this approach is the disabling of data accesses until the entire computation program is finished (i.e. Non-PIM requests are stalled). This limits the effective utilization of the Dual-Mode feature of the DRAM-PIMs. The alternative DRAM-PIM

control strategy is to introduce new commands (e.g. ADD, MUL, etc.) for PIM operations, which is proposed in Hynix-AiM [37], McDRAM [40] and Grad-PIM [38]. This enables the differentiation of Non-PIM data access requests and PIM requests that allow the interleaving of data access requests and computation requests. Hence, simultaneously operating in both memory mode and computation mode, provided that the specified timing dependencies between the commands are satisfied. The DRAM-PIM command interface in the second approach is varied according to the number of new compute commands supported by a given PIM architecture. For example, McDRAM [40] introduce 10 new commands for their architecture, while Hynix-AiM [37] introduced 16 new commands. This results in the vendor-specific DRAM interface and vendor-specific memory controller design, which is not the case for commodity DRAM counterparts. Hence, impacts the compatibility of the memory controller for various PIM devices from different vendors. The challenge here is to design the external interface and control protocol that exploits the advantages of both approaches.

Challenge 5: *Designing a unified external interface and protocol that is inter-vendor compatible and enables simultaneous operation of memory mode and computation mode.*

3.4 Summary

This chapter presented the design space exploration of DRAM-PIM architectures for inference and training tasks, discussed the design approaches of state-of-the-art DRAM-PIM architectures, and highlighted the various critical challenges for different design dimensions. This study is used as the basis for all the design decisions of new architectures and novel approaches presented in Chapters 4, 5, and 6.

Chapter 4

DRAM PIM Architecture for DNN Inference - Digital Approach

As a starting point for DRAM-PIM architectural investigations, I target only the inference tasks. I propose two novel DRAM-PIM architectures for DNN inference in this thesis (i.e. Chapter 4 & 5). Both the architectures are designed to be fully compatible with commodity DRAM architectures by strictly avoiding the modification of highly optimized Sub-Array design and retaining the core structure of commodity DRAMs. More specifically, the architectures aim to reuse most of the existing DRAM circuitry and internal buses. Consequently, the architectures function as both an energy-efficient accelerator and as a high-capacity standard DRAM (i.e. Dual-Mode feature).

The commodity DRAM architectures are dominated by the density and yield-optimized memory arrays and do not explicitly contain computation units apart from the peripheral logic for external data transfers. An important decision for this architectural investigation is the identification of the appropriate location for new PIM computation units inside the commodity DRAM without majorly modifying the bank design. The straightforward solution is to integrate the computation units near the bank-peripheral region [34, 37, 39, 40], but as presented in Section 3.1.3 of Chapter 3, this approach encounters limited data parallelism and consumes high data access energy. For example, the amount of parallel data available at a bank peripheral region is a low value of 128 or 256 bits per cycle, while the near Sub-Array (i.e. PSA) region has 1024 bits/Sub-Array (or 2 KB per page). Further, the data access energy in PSA region is 10$\times$ less than the bank peripheral region [39]. Hence, one of the aims of this thesis is to integrate the DNN computation units in the PSA region for achieving higher energy efficiency by exploiting high data parallelism and low data movement energy. However, the PSA region is sensitive to the area-intensive computation units like multipliers (refer Challenge 1 in Chapter 3). To respect the area constraint of PSA region, my architectures partition the DNN computation units into two entities. The first entity is a lean, low-area computation unit referred to as Primary Processing Unit (PPU) that is integrated in the PSA region. The second entity referred to as the Secondary

Processing Unit (SPU) consists of all the other substantial units and is integrated in the bank peripheral region that tolerates a larger area of computation units. Thereby exploiting the advantages of both regions. Contrary to my approach, the state-of-the-art DRAM-PIM architectures DrAcc [30] and LAcc [31] proposed to modify the Sub-Array with special compute rows. The other architectures like DRISA [29] and SCOPE [32] proposed reorganizing the bank and Sub-Array dimensions of DRAMs into 1024 banks and 256×2096 Sub-Arrays, which are not typical for commodity DRAM architectures. Most state-of-the-art DRAM-PIM architectures discussed in Chapter 3 either integrate all the computation units near the Sub-Array or in the bank peripheral region without effectively exploring the partition approach. In this chapter and the next chapter, I investigate the novel PIM architectures that are compatible with commodity DRAM architectures to support the Dual-Mode feature, incur minimal modifications to the existing circuits, exploit the advantages of PSA region for high energy efficiency without violating its area constraints, and finally enable seamless integration (i.e. Goal 1 & 2, Chapter 1).

All my DRAM-PIM architectures investigations are based on a DDR4-type commodity DRAM architecture. However, the architectures are compatible with any type of DRAM (e.g. HBM2, GDDR6). Additionally, all the architectural investigation and evaluation results presented in this thesis are for a single 2D DRAM die, which can be further extended to DIMM or 3D organization in the future. I propose a novel digital approach based DRAM-PIM architecture in this chapter, and a novel mixed-signal approach based DRAM-PIM architecture in the next chapter. In both these chapters, I only focus on Goal 1 & 2 as an initial step, while novel architecture approaches to meet Goal 3 (i.e. scalability goal, Section 1.2.3) are proposed in Chapter 6. The digital approach architecture relies on a shifter-based approximate multiplier whose area is compact enough to be placed in the PSA region. More specifically, the architecture relies on Power-of-two quantization of neural network weights, which allows multiplications to be transformed into one or two shift operations for approximate computing. The contributions of this chapter are previously published in [49, 63].

4.1 Background: Power-of-two Quantization

As shown in Equation 4.1, the key computation operation of any artificial neuron in DNN is MAC , where o_n is the output value of neuron n, x_i is the i-th input of neuron n (out of a total of I inputs), $w_{n,i}$ is the i-th weight, b_n is the bias, and ϕ is a non-linearity function such as *Rectified Linear Unit* (ReLU) [115].

$$o_n = \phi\left(\sum_{I-1}^{i=0} x_i \cdot w_{n,i} + b_n\right) \tag{4.1}$$

$$\begin{gathered} x_q \cdot w_q \approx s_b \cdot (x_q << b) + s_c \cdot (x_q << c) \\ \textit{where } w_q \approx s_b \cdot 2^b + s_c \cdot 2^c \end{gathered} \tag{4.2}$$

In most modern DNN inferences, the weights and input values are quantized to 8 bits, which has negligible accuracy loss compared with 32-bit floating-point representations [116]. However, the 8-bit MAC operation requires area and power-hungry multipliers that are very large to be integrated in the PSA region (refer Section 3.1.2). This challenge is addressed by adapting a Power-of-two quantization that rounds each 8-bit weight to the nearest power-of-two value to implement a low area approximate multipliers [116]. There are two variants of this approach: 1) Two-hot quantization, and 2) One-hot quantization. The two-hot quantization approach is represented by Equation 4.2. Each 8-bit quantized weight is converted to two 3-bit values b and c that indicate the number of shifts and two 1-bit values s_b and s_c that indicate the sign. In one-hot quantization, the 8-bit quantized weights are simply rounded to the nearest power-of-two value (i.e 2^a). Through these power-of-two representations, the multiplication of x_i and $w_{n,i}$ is transformed into a one or two-shifter-based approximate multiplier that drastically reduces the computation unit area in comparison to a standard 8-bit multiplier. However, the one-hot quantization method has a high rounding error for large values, which results in a significant DNN accuracy degradation (i.e. $>10\%$) [116]. In order to exploit the advantages of one-hot quantization (i.e. single shifter for multiplication), a retraining scheme is employed to recover the accuracy losses. This retraining step is required once per network post-training. Algorithm 1 shows the pseudocode for retraining. First, the floating-point weights w in each retraining iteration t are converted to 8-bit linear quantization (W_q) and then one-hot quantization $W_{q_{1hot}}$. The one-hot quantized weights are used to calculate the DNN inference output y_t at iteration t, which is used to compute the DNN cost function $C(y)$ (e.g. Cross-entropy cost function). The gradient of $C(y)$ with respect to weights w is computed as in Equation (4.3). Finally, the weights w are updated as in Equation (4.4). This process is continued until the *Convolutional Neural Network* (CNN) accuracy is within the pre-defined accuracy tolerance (e.g. -2%). This enables the adoption of one-hot quantization in addition to two-hot quantization for approximating the multiplication operation in my DRAM-PIM architecture for DNN inference. The retraining of networks to recover the accuracy loss of one-hot quantization is not my individual contribution and it is performed in collaboration with M.Sc. Taha Soliman and his colleagues from CR/AEX2, Robert Bosch GmbH. The scope for my research is to leverage the existing approximation techniques to investigate the hardware architectures for DRAM-PIMs that are compatible with commodity DRAMs.

$$\frac{\partial C(y)}{\partial w} = \frac{\partial C(y)}{\partial y}\frac{\partial y}{\partial W_{q_{1hot}}}\frac{\partial W_{1hot}}{\partial W_q}\frac{\partial W_q}{\partial w} \tag{4.3}$$

$$w_{t+1} = w_t + \Delta w = w_t - \frac{\partial C(y_t)}{\partial w_t} \quad , \tag{4.4}$$

Input: Pre-trained DNN with floating-point weights w and accuracy y_0, accuracy tolerance ϵ; input data with 8-bit linear encoding x_q
Result: Optimized DNN weights in one-hot encoding $w_{q_{1hot}}$
do
  $w_q = Q_{linear}(w)$; Linear quantization
  $w_{q_{log}} = Q_{log2}(w_q)$; one-hot quantization
  $y_t = DNN(x_q, w_{q_{log}})$; DNN inference
  if $|y_t - y_0| > \epsilon$ **then**
    Compute cost function $C(y_t)$;
    Back-propagation of $\frac{\delta C(y_t)}{\delta w}$;
    Update floating-point weights w;
    $t = t + 1$;
  end
while $|y_t - y_0| > \epsilon$;

Algorithm 1: CNN Optimization with one-hot quantization

4.2 Architecture

In this section, I present my first DRAM-PIM architecture, i.e. "a novel DRAM-based PIM architecture for one-hot and two-hot quantized DNN inference". The architecture is designed for 8-bit input data and 4-bit one-hot quantized weights (i.e. 1-bit sign and 3-bit shifts). In the case of two-hot quantization, two 4-bit weights are used per multiplication operation. As mentioned in the beginning of this chapter, the DNN computation units are partitioned into two entities, i.e. PPU and SPU, in order to reduce the computation area near the DRAM Sub-Array or in the PSA region. The PPUs are designed to be lean and produce only the intermediate MAC results of DNN artificial neurons. All the other subsequent computations are processed by the SPUs that are integrated in the bank peripheral region. This architecture considers a commodity DRAM Sub-Array dimension of 1024×1024, but it is also applicable to other standard Sub-Array dimensions.

4.2.1 Primary Processing Unit (PPU)

A PPU consists of shifters and adders that compute the intermediate MAC results of an artificial neuron. The architectural choice for selecting the number of shifters in the PPU is based on the adder unit size and its area. As the number of inputs of the adder is increased, the adder tree depth is also increased, which increases the area of PPU. For example, the 512 input binary adder tree in DRISA contributed up to 50% of the total DRAM-PIM area [29]. Furthermore, each input of the adder in this architecture has to be 16-bit, as the input data to the shifter is 8-bit whose output will be a maximum of 16-bit. This is also a major contributor to the adder area. Hence, as a starting point, I limit the number of adder inputs to four and consequently four shifters per PPU. To compensate the throughput

reduction due to low number of shifters in PPU (i.e. to increase the effective utilizing the data parallelsim of PSA region), I propose to integrate multiple PPUs near each Sub-Array (E.g. four).

Figure 4.1 shows the architecture of PPU. Each Sub-Array consists of four PPUs, two on each side of the Sub-Array. Similar to PSAs in commodity DRAM architectures, the PPUs are also shared with their neighboring Sub-Arrays in this architecture. For the sake of the image clarity, the reference LBL (LBL*) connections from a vertically neighboring Sub-Array to the respective PSAs are not shown in Figure 4.1. However, in the actual design, the LBLs* have to pass through the PPUs to be connected to the PSAs. The foremost component of the proposed PPU is the *Shadow Latch* (SHL). The number of SHLs are same as PSAs per Sub-Array. The circuit design of SHL is also same as PSA, but smaller in size are compared to PSA (i.e 1/4th the size of a PSA). SHLs are used to buffer the DNN weights that can be reused for multiple computations by changing the input data in the PSAs. The procedure for sensing weights and input data is as follows:

- The input data and weights are stored in a separate rows within a Sub-Array.
- Initially, LBLs are connected to both PSAs and SHLs. The PIM control unit sends ACT command to a weight row, whose data are sensed by both PSAs and SHLs.
- As soon as the sensing is completed (i.e. latency t_{RAS}, refer Chapter 2), SHLs are disconnected from LBLs, but the weights continue to be buffered in SHLs. Further, PSAs and LBLs are precharged to activate the input data row within a Sub-Array.
- After the precharge duration (t_{RP}), the PIM control unit sends ACT command to an input data row, which is sensed and buffered by the PSAs.

The DNN computations are started after sensing the weights and input data. The *Column Select Switches* (CSS) block one (i.e. CSS-1) selects four input data from the PSAs to the four shifters and CSS-2 selects four weight data from the SHLs to the four shifters of each PPU. The circuit design of the CSS-1 and CSS-2 are identical to a commodity DRAM CSS block. The 3-bit shift input of the weight data is represented in an 8-bit one-hot encoding format (e.g. 5 = 3bit'101 = 8bit'0001_0000), where the position of logic-1 bit represents the number of shifts. This representation simplifies the shifter design to a switch based shifter circuit, as shown in Figure 4.2. For the sake of simplicity and page space considerations, a shifter with 2-bit data input, 4-bit shift input in one-hot encoding and 4-bit data output is shown in Figure 4.2. However, the shifter that is employed in this architecture consists of 8-bit data input, 8-bit shift input (corresponding to 3-bit binary shift input), and 16-bit data output. The 16-bit outputs from the four shifters are added and the 18-bits adder output is truncated to obtain a 16-bit intermediate MAC result, which is latched in the result latch. This truncation is done to ensure that the transfer of intermediate MAC results via MBLs are in multiples of 8-bit (i.e. aligning with the number of MBLs). The transfer of

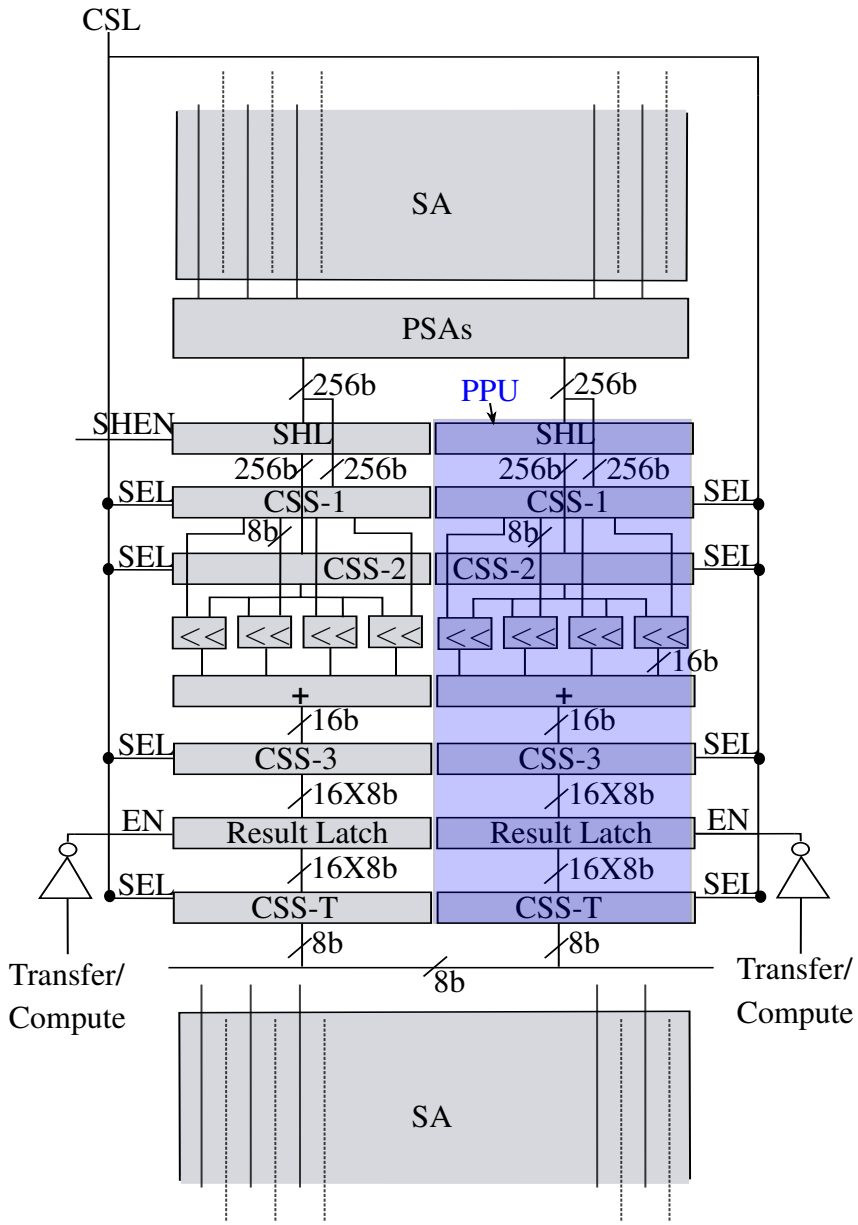

Figure 4.1: Architecture 1: PPU architecture. For image clarity the connection of LBL* to PSA is not shown. In this figure Sub-Array is abbreviated as SA.

intermediate results to the SPU is done by reusing the existing circuitry from commodity DRAM architecture such as MBLs, SSAs, and CSLs. CSS-T block connects the output of result latches to the MBLs. This is similar to a commodity DRAM read command, where the data is transferred from the PSAs to SSAs via MBL, where CSS block select the required data column. The sign bit of the network weights is encoded within the 8-bit shift input and is also passed as an input to the adder. Note that the sign-bit is encoded in the MSB-bit of the shift input. Consequently, the weight data +7 is represented as 1100_0000 that could be decoded with a simple AND-logic.

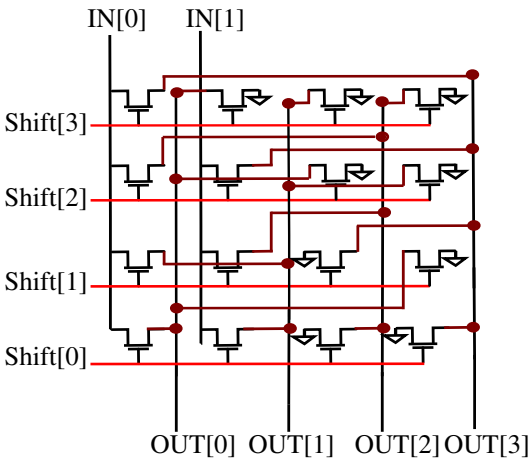

Figure 4.2: Architecture 1: Example Shifter Circuit for 2-bit Input.

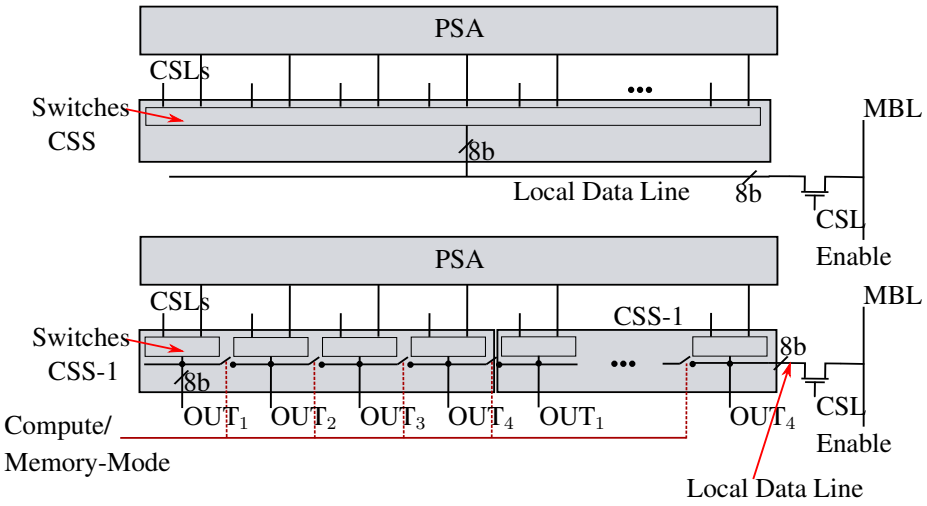

Figure 4.3: Architecture 1: Conventional CSS block (top) compared with CSS-1 block (bottom) of this architecture.

The transfer of data from SSA to PSA or vice-versa for conventional write or read operations in memory-mode is via CSS-1 blocks. Figure 4.3 compares the CSS block design of commodity DRAM (i.e. top) with the CSS-1 block design of this architecture (i.e. bottom). In commodity DRAMs, the CSS block selects the appropriate PSA output based on the CSL control signals inputs and transfers this selected 8-bit or 16-bit output via the local data lines and MBLs. The CSS-1 block is similar is design to commodity DRAM counterparts. This figure shows two CSS-1 block, which corresponds to two PPUs on each side of the Sub-Array. The CSS-1 block is internally divided into four blocks of switches, but the design of each block of switches is same as the conventional CSS block. In the compute-mode, the four blocks of switches select four inputs to the shifters (i.e. OUT1 to OUT4). In the memory-mode, the output of all the switches blocks are connected to mimic

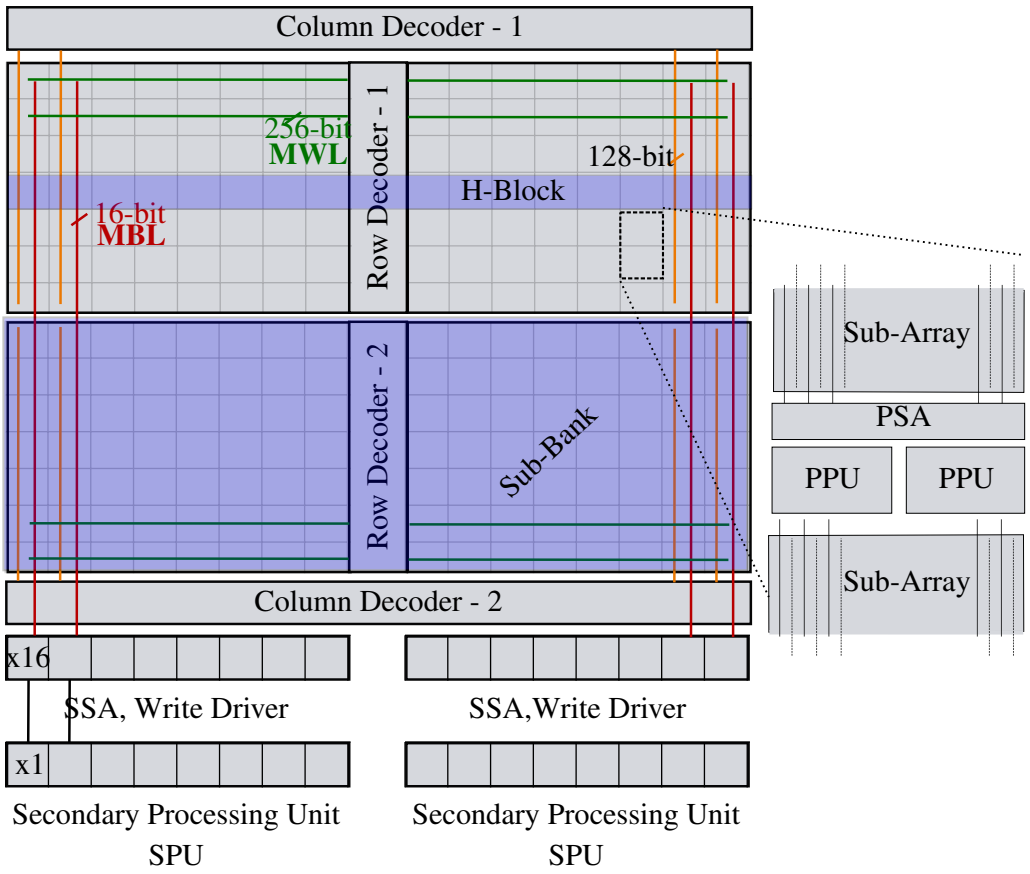

Figure 4.4: Architecture 1: PIM Bank Architecture.

local data lines and only one block of switches is active to select an 8-bit or 16-bit data from the PSA that are transferred to SSA via local data line and MBL.

4.2.2 Bank-Level Architecture

Figure 4.4 shows the bank-level architecture. In a typical commodity DRAM bank, all the Sub-Array of an H-Block (a H-Block is a row of Sub-Arrays) are parallelly enabled for the data accesses as explained in Chapter 2, Figure 2.3. This Sub-Array parallelism is also exploited in this DRAM-PIM architecture to increase the computation parallelism (i.e. higher throughput). For example, an H-Block with 16 Sub-Array will enable 64 parallel PPU per bank (i.e. 16×4 PPU per Sub-Array). The banks of this architecture are partitioned into two sub-banks to enable parallel operation of two H-Blocks within a bank. This enables the implementation of a pipeline structure within a bank. The pipeline structure is as follows - when the PPUs of a H-block in one sub-banks is computing the intermediate results, the PPUs in a H-Block of another sub-bank transfers its previously computed intermediate results to the SPUs for further processing. This pipeline is required to mask the long stall of $t_{RP} + t_{RAS} + t_{RP} + t_{RCD}$ (e.g. 24 cycles) required for sensing the input data

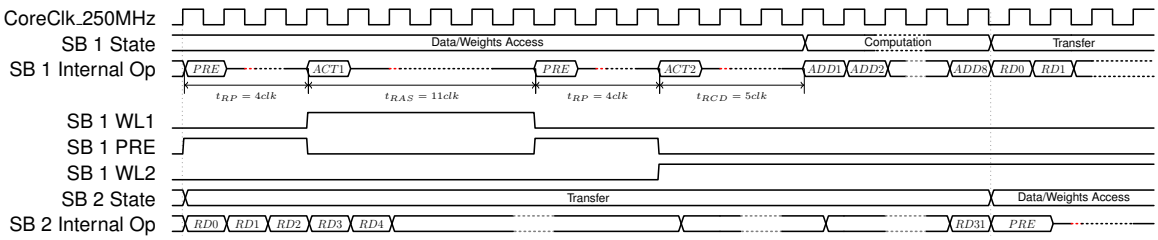

Figure 4.5: Architecture 1: PIM Bank operation timing diagram.

and weights. Please refer Chapter 2 for all the timing definitions. This is elaborated using an example timing diagram (Figure 4.5) of the bank-level operation. In this example, sub-bank-1 (SB 1) is operating in computing the intermediate results of the current iteration and sub-bank-2 (SB 2) is transferring the intermediate results of the previous iteration. A total of $t_{RP} + t_{RAS} + t_{RP} + t_{RCD}$ cycles latency (e.g. 24 cycles) is introduced for this sensing weights and input data, where ACT refers to activate command and PRE refers to precharge command. This is followed by the MAC operations in PPU (i.e. ADD1-ADD8). The PPUs have to be inevitably stalled until the sensing is finished. However, to avoid the stalling of SPU as well, I proposed a pipelined bank-level architecture. The transfer of intermediate results computed in a H-Block of SB 2 in parallel to the sensing+computation in a H-block of SB 1 ensures that the SPUs are effectively utilized without stalling during the sensing. For example, a 1024-bit wide Sub-Array with four PPUs require eight cycles to process all the sensed data and 32 cycles to transfer all the computed 512-bit intermediate results. Once all the intermediate results are transferred and computation of all the sensed data is finished, the role of SB 1 and SB 2 is switched.

The partitioning of banks into two sub-banks is in accordance with commodity DRAM architectures. This approach of having individual row-decoders per sub-bank but shared MBLs, SSAs, control logic, and power network was first proposed in my 2021 publication [63]. This approach does not require any modifications to the bank circuits, except for the additional multiplexer near the row decoder input, and is the simplest approach to enable two H-Blocks within a bank. A similar approach is also adapted by one of the memory vendor's 2022 DRAM-PIM publications (i.e.Hynix-AiM [37]) to activate two separate pages (one per H-Block) within a bank. There are other state-of-the-art publications [90–93] that propose different architectures to activate multiple rows within a bank, but require modifications to row-decoder (refer Chapter 7).

Secondary Processing Unit (SPU): SPUs are integrated near the SSAs or bank peripheral region. This PIM architecture has one SPU for every column of Sub-Arrays, E.g. 16 SPUs per bank if an H-Block has 16 Sub-Arrays. SPUs consists of a 32-bit sequential adder, buffers, max-pooling unit, and ReLU neural network activation function. The sequentially accessed intermediate results from the PPUs are accumulated in the accumulator. The computation of 32-bit final MAC result in a SPUs, corresponds to the MAC output of a single neuron n. The buffers store the temporary/final results of the MAC operation

and also enable out of order executions. After the computation of final 32-bit MAC results, the results are forwarded to the activation function unit and quantization unit to generate 8-bit output o_n (refer Equation 4.1) of output feature map.

4.2.3 Device-Level Architecture

The commodity DRAMs have multiple parallel operating banks that have a shared internal data and command bus. This bank parallelism is exploited in this architecture for increasing the computation parallelism. The total number of available banks in a DRAM architecture are partitioned into two sets referred to as *computation bank-set* and *storage bank-set*. The computation bank-set is responsible for computing the output feature map of an arbitrary CNN layer (e.g. CNN layer 3). The DNN output o_n from the computation bank-set, which correspond to the input data of the next CNN layer (e.g. CNN layer 4), are concurrently stored in the appropriate memory locations of storage bank-set. This pipeline approach mask the write-back latencies required to store the input data of the next layer. The functionality of the bank-sets are swapped after all the computations of a particular CNN layer (e.g. layer 3) is finished. Further, the computation of the next CNN layer (e.g. layer 4) is initiated in a bank-set that was previously configured for storage. For example, is bank-set A is configured as a computation bank-set and bank-set B is configured as a storage bank-set for CNN layer 3, then post completion of layer 3 computation, bank-set A is configured as a storage bank-set and bank-set B is configured as a computation bank-set for CNN layer 4.

In order to transfer the output feature maps data (i.e. o_n) between the bank-sets, the internal data bus and the IO circuitry that manages the transfers of data between the device IO and the banks are reused. After a bank computes a specific number of output feature map data, the PIM control circuitry generates an internal transfer command to initiate bank to bank transfer. The data bus is time multiplexed between the banks of the computation bank-set to avoid the collision while transferring the output feature maps to banks of storage bank-set. In some cases, the same output feature maps have to be written to multiple banks for parallel processing of the same feature maps with different weight kernels. Hence, I introduce multi-bank commands in this architecture, where a given command is sent to multiple banks. The multi-bank command in commodity DRAMs is applicable only for precharge-all (PREA) and refresh (REF), but not for commands like read (RD), write (WR), and activate (ACT). This architecture extends the multi-bank command to other commands such as RD and WR. Hence, allowing the same output feature data to be concurrently written to the addressed memory arrays in multiple banks. Similar to conventional read and write commands, the activation of appropriate rows within a bank using the ACT commands is required prior to the internal transfers. The control circuit can manage the activation of appropriate rows without violating any timings by leveraging the deterministic nature of DNN inference.

The bank to bank internal data transfers proposed in this architecture, which was previously published in my publications [63] (2021) and [62] (2020). It is unique to PIM and is not available in commodity DRAM architectures. A similar approach is also adapted in Hynix's DRAM-PIM architecture [37] (2022). The multi-bank commands proposed in this thesis, which was also previously published in my publications [63] (2021) and [62] (2020), is also proposed by Samsung's 2021 DRAM-PIM publication (i.e. FIMDRAM [34]) for ACT command.

Control Logic: This architecture assumes the necessary modifications to the memory controller to send the new PIM commands. The detailed protocol for external control of the PIM is described in Chapter 6. The main focus of this chapter and the next chapter is to investigate the most feasible architecture that enable integration of computation units in the PSA region.

4.3 Experimental Setup

My PIM architecture is constructed on top of a state-of-the-art 8 Gb commodity DRAM architecture with 8 banks, 64 H-Blocks per bank, 16 sub-arrays per H-Block, a 1024×1024 standard sub-array dimension, and a 2 KB page. The area of the commodity DRAM Sub-Array is calculated using the reverse engineered unit cell dimensions, LBL pitch, and LWL pitch (e.g. 66 nm) of 2y nm technology (i.e. between 25 nm to 27 nm) based 8Gb DRAM dies [117], [118]. Please note, DRAM technology is typically represented like 2y, 1y ($\approx$17 nm), 1z ($\approx$14 nm), etc. The area of the other standard DRAM circuits (e.g., decoders, sense amplifiers, etc.) is evaluated using the DRAM architecture exploration tool DRAMSpec [119]. The required 2y nm DRAM technology input parameters for DRAMSpec are extracted from multiple sources and articles [48,117,118,120–124]. To provide an indication of the correctness of the area evaluation methodology, I compare the estimated 8 Gb standard DRAM die area calculated with this methodology (i.e., 74.44 mm^2) to the area of 8 Gb commodity DRAM dies from memory vendors i.e. Samsung's DDR4 (i.e., 59.00 mm^2) [118], Hynix's DDR4 Gen-1 (i.e., 76.0 mm^2) [117] and Micron's DDR4 (i.e., 58.48 mm^2) [125].

The latency and energy of the DRAM circuits are estimated for a core-clk of 250 MHz using an in-house enhanced circuit-level DRAM model (SPICE) that can be equipped with different kinds of SPICE transistor models and PDKs, such as FreePDK [126] with PTMs [127], Cadence PDK models, and as well proprietary DRAM transistor models. This DRAM model includes the wire parasitics extracted from Qimonda's 45nm DDR3 design. The circuit-level results and timing correctness of this model are presented in [128] using BSIM4 model cards built on Low Power PTM transistor models that are adapted to DRAM technology. We updated this DRAM model to 2y nm DRAM technology based on the scaling results presented in [129].

The area, latency and energy evaluation of the PIM computation units are performed using an adapted UMC 65 nm Low-Leakage CMOS technology. For example, the number of metal layers to three in order to align with DRAM technology The use of 65 nm logic technology is based on the evidences showed by authors of [130] that the peripheral transistors in a 2y nm DRAM technology is approximately equivalent to a 50-65 nm logic technology. All the computation unit results are extracted post place and route using Synopsis' IC Compiler.

The CNN benchmarks used in our experiments are ResNet20 [131], SqueezeNet [132], VGG16 [133] and ResNet18. ResNet20 is trained with the CIFAR10 dataset [134] and all the other networks are trained with the ImageNet dataset [135]. The overall throughput and power consumption for this architecture is evaluated with a cycle-accurate python-model. This model is back annotated with the latency and energy results obtained of each individual PIM operations.

4.4 Results

In this section, I present the experimental results of my new digital approach based DRAM-PIM architecture. As mentioned in Chapter 1, this architecture research work published in [63] was conducted in close collaboration with my colleague M.Sc. Taha Soliman from CR/AEX2, Robert Bosch GmbH. All the architectural investigations, design choices and the calculation of circuit-level results like area, latency and energy consumption of DRAM-circuits and PIM units are my individual contributions. The algorithmic evaluations, accuracy results and the cycle-accurate python-model implementation of the proposed architecture, which is required for the overall performance and energy efficiency evaluation, are done together with with M.Sc. Taha Soliman.

4.4.1 Accuracy Results

Before presenting the architectural results, it is important to show that the accuracy loss of the one-hot quantization is minimal. Figure 4.6 presents the CNN inference accuracies with linear 8-bit quantization, one-hot, two-hot, and 32-bit floating-point. These accuracy results include the error introduced by truncating the 2 LSBs from the adder output in PPU to obtain a 16-bit intermediate MAC result, as described in Section 4.2.1. VGG16 did not require any retraining because the accuracy of this network for one-hot quantization without retraining is within the 5% margin.

Retraining Overhead - Our quantization-aware training algorithm has a retraining overhead that dependent on the type of network and its dataset. One-hot quantized ResNet20 for CIFAR10 is retrained over 60 epochs, which represents an overhead of 33% w.r.t. the original training time. SqueezeNet and ResNet18 require only 12 retraining epochs, which

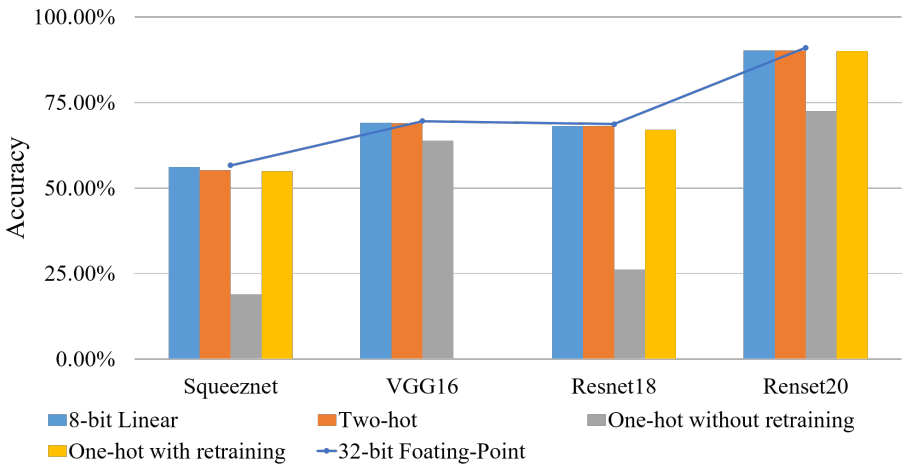

Figure 4.6: Architecture 1: CNN inference accuracy compression of all aforementioned quantization. Courtesy: Taha Soliman

represents an overhead of only 6.7% and 10.8% respectively. In this context, one epoch refers to one training cycle on the entire training dataset. Similar to network training, the aforementioned overhead appears only once per network per dataset.

4.4.2 Area Results

The area evaluations of all the relevant circuits of this architecture is shown in Table 4.1. I evaluated the area results for three architecture variants, (1) all H-Blocks are enhanced with PPUs (referred as All-MAC-rows) (2) a limited number of H-Blocks (16 in this case) in each bank are enhanced with PPUs (referred as 16-MAC-rows) (3) 8-MAC-rows. The preferred variant is 8-MAC-rows as it achieves the highest memory density (bits/mm^2) and the lowest area overhead of 6.6% in comparison with the baseline standard DRAM die. This enables the proposed DRAM-PIM architecture to be employed as both high memory density host memory and energy efficient accelerator, i.e. Dual-Mode feature. The memory space of the computational Sub-Arrays in 8-MAC-rows variant is not sufficient to store all the weights of larger networks like VGG-16. In such cases, only the network weights corresponding to initial layers are stored in the available space. The remaining network weights are stored in non-computational Sub-Arrays and the weights in computational Sub-Arrays are overwritten on a timely basis along with the writing of output feature map data in the storage bank-set. However, the easy solution is to employ a suitable architecture variant. All the architecture variants result in a similar performance and power consumption, since only 2 H-Blocks per bank are actively operating at any given instance of time. The layout of the PPU's adder block is shown in Figure 4.7.

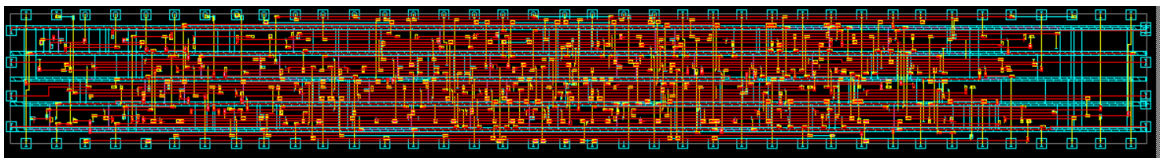

Figure 4.7: Architecture 1: PPU's adder layout @65 nm.

Table 4.1: Architecture 1: Area results.

Block	**Area** $Height \times Width$ [μm^2]
Sub-array (SA)[a]	83.68×74.036
PSAs + CSS ($512\times$), per side of Sub-Array	10×74.036
Sub-array with PSAs[b]	93.68×74.036
Bank without computation units[a]	6329.2×1444.58
Column decoder	125×1444.58
Row decoder	6329.2×250
8 Gb Standard DRAM Die	**74.4442 (mm^2)**
SHL ($512\times$), per side of Sub-Array	2.5×74.036
Shifters ($8\times$), per side of Sub-Array	5×74.036
CSS-1, CSS-2, CSS-3	0.5×74.036
Adder ($2\times$), per side of Sub-Array	15.47×74.036
Result Latch, per side of Sub-Array	2.5×74.036
SPU Adder, per column of Sub-Arrays	820
SPU Buffers ($32bit \times 16$)	1231.2
Max pool, per column of Sub-Arrays	548.3
Bank - PIM (8-MAC-rows)	6759.01×1444.58
Bank - PIM (16-MAC-rows)	6974.77×1444.58
Bank - PIM (All-MAC-rows)	8323.27×1444.58
8 Gb PIM Die (8-MAC-rows)	**6.6%** overhead
8 Gb PIM Die (16-MAC-rows)	**10.02%** overhead
8 Gb PIM Die (All-MAC-rows)	**30.96%** overhead

[a] including redundancy

[b] SA area considers only single side of PSA due to PSA sharing

Table 4.2: Architecture 1: Latency and Energy of various operations for the core clock frequency of 250 MHz.

Operation	Latency,[ns]	Energy,[pJ]
ACT (2KB page)	t_{RCD}=12, t_{RAS}=30	614
PRE (2KB page)	t_{RP}=10	314
One Basic unit	3.76	1.14
RD/WR (PSA-SSA)	t_{CCD}=4 (1 Clk)	418/438
SPU - per column of Sub-Array	3.13	1.8
Bank to Bank communication	3 clks	1pJ/bit
Device Background Power	-	34mW

4.4.3 Performance Results

The latency and energy of various operations that are required for the calculation of the overall performance and power consumption using the Python model are summarized in Table 4.2. For instance, activating a 2KB bank page (row) takes around 12 ns and consumes around 614 pJ. The performance of the proposed DRAM-PIM architecture is compared with the two non-PIM architectures, i.e. Eyeriss v1 [136] (ASIC) and Eyeriss v2 [137] (FPGA) DNN accelerators. Eyeriss v1 is clocked at 200 MHz, contains 164 processing elements (PEs) for performing MAC operations, 181.5 KB of on-chip memory and is implemented in a 65 nm logic technology. Eyeriss v2 is implemented in a Xilinx Zynq UltraScale+ XCZU9EG device operating at 300MHz, containing 192 PEs, 246 KB of on-chip memory, and is extended to support one-hot and two-hot quantization. The FPGA implementation and its results are provided by Taha Soliman and his team from CR/AEX2, Robert Bosch GmbH. The performance results of all three architectures are compared in Figure 4.8, which is measured in terms Giga operations per second (GOPS), where one OP is equal to one MAC operation. In the figure, the Eyeriss v1 is referred to as the *ASIC* and Eyeriss v2 is referred to as the FPGA. Note that we deliberately chose the Eyeriss v1 as the representative of an ASIC-accelerator because the performance numbers presented in [136] take into account the effect of external memory and on-chip buffers, which is not the case in many research publications related to hardware DNN accelerators. My novel DRAM-PIM architecture has an edge over the other implementations in terms of throughput and clearly highlights the advantages of PIM architectures. The aforementioned conventional systems were limited by the global buffer size and the data/weights access latency of memory hierarchy.

4.4.4 Energy Efficiency Results

Figure 4.9 compares the energy efficiency of the praposed architecture against the other aforementioned conventional systems, i.e. Eyeriss v1 (ASIC) and Eyeriss v2 (FPGA). In

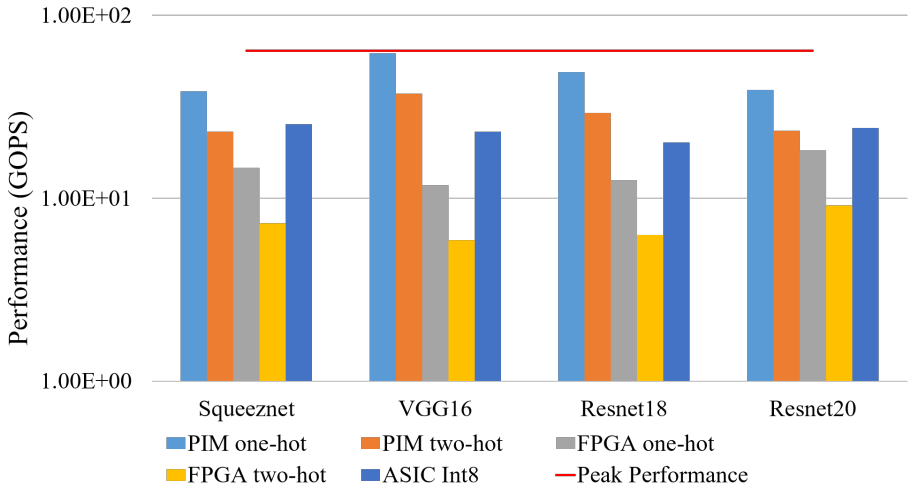

Figure 4.8: Architecture 1: Performance (GOPS) Comparison.

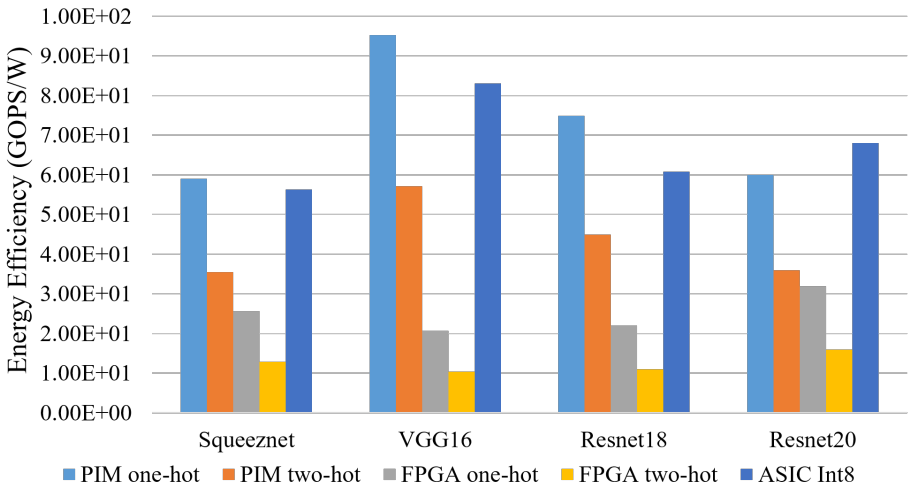

Figure 4.9: Architecture 1: Energy Efficiency (GOPS/W).

case of FPGA, only the core chip energy is considered. My architecture outperforms the other systems in terms of the energy efficiency for memory intensive networks such as VGG16 and Resnet18. In case of the smaller network size like Squeezenet and smaller datasets like CIFAR10 (refer Resnet20 results), Eyeriss (i.e. ASIC) is comparable to PIM architecture due to reduced memory accesses (to both external memory). The energy efficiency of one-hot quantization is on an average 40% higher than two-hot.

4.4.5 DRAM-PIM Comparision

In this section, I compare the proposed DRAM-PIM architecture with five state-of-the-art DRAM-PIM architectures i.e. re-evaluated 8Gb DRISA [32], LAcc [31], SCOPE [32], UPMEM [35] and McDRAM [39] (refer Table 4.3). I consider only these five DRAM-PIM architectures because at the time of publishing my novel digital approach architecture [63] in the year 2021, these architectures were the available state-of-the-art. In the following chapter, I compare all the prominent DRAM-PIM state-of-the-art architectures up to date against the novel architectures presented in this thesis. The proposed digital approach architecture achieves a normalized peak throughput per bank of 8 GOPS/bank, which is higher than state-of-the-art architectures and equal to McDRAM architecture. Please note that the number of computation banks considered in the peak throughput calculation of McDRAM is eight, while my architecture limit the number of computation banks to four out of the total eight banks. Thereby resulting in the same normalized peak throughput per bank in both, as both architectures have a total of eight banks. However, within a single bank, McDRAM architecture computes a maximum of 32 8-bit MAC operations per 4 ns clock cycle, while my architecture is capable of computing 64 8-bit MAC operations in the same timeframe. Hence, the peak throughput would be double if my architecture also considered 8 computation banks instead of four. The power consumption of all the state-of-the-art architectures is higher than the architecture proposed in this work. Especially, LAcc and SCOPE consume a very high power that is $8\times$ and $267\times$, respectively, higher than my architecture. LAcc [31] proposed to perform bit-wise multiplication and bit-wise addition using special rows in each Sub-Array. This approach require a high number of activate and precharge operations for computing add operation (i.e. 20 ACT and PRE), which is the major contributors to the high power consumption. The higher power of UPMEM and McDRAM is due to the internal read operations that are required to fetch data to bank peripheral region for every MAC operation. These internal reads consume higher energy as compared to accessing data near the PSA and as compared to computation unit energy. E.g. UPMEM's DDR4-based DRAM PIM die consumes $\approx 150\,pJ$ for transferring 64-bit data, while the computation unit (DPU) consumes only $\approx\ 20\,pJ$. Contrarily, my architecture requires an internal read operation for every second MAC operation. As a result, this architecture achieves the highest energy efficiency among all, e.g. $2\times$ compared to LAcc and SCOPE. Furthermore, the power consumption of this architecture is within the accepted range of a commodity DRAM device and its packaging (i.e. 1 W), while SCOPE requires special packaging and cooling to withstand such a high power consumption. Please note, the fair comparison of the proposed architecture to SCOPE [32] is not straightforward because SCOPE is designed with 1024 highly modified banks to achieve a performance that is comparable to GPU. On the other hand, my aim is to design a DRAM-PIM architectures that are capable of functioning both as DNN accelerator and standard DRAM (or host memory) without any major modification to the bank structure and design.

Table 4.3: Comparison of DRAM based PIM architectures.

Parameter	In Sub-Array	Near Sub-Array		Bank Peripheral	
	LAcc [31]	SCOPE [32]	This work All-Digital [63]	UPMEM [35]	McDRAM [39] (C/D) [a]
DRAM process	25nm	22nm	2ynm	2xnm	20nm
Capacity per die [Gb]	8	8	8	4	8
Num of banks	32	1024	8 (4+4) [b]	8	8
Page Size	4KB	4KB [c]	2KB	2KB	2KB
Operation precision	INT8	INT8	INT8	INT8	INT8
DRAM core frequency [MHz]	-	124	250	500	250
Area [mm^2]	54.8	273.38	79.41	-	-
Area overhead [%]	**1**	342.24	6.6	-	**4.7**
Power [W]	5.3	176.4	**0.66**	≈ 1.2	1.6 [d]
Power Density [W/cm^2]	9.67	64.53	**0.83**	-	-
Peak Performance per die [TOPS] [e]	0.27227	**7.2**	0.064	0.03125	0.064
Effi. [GOPS/W]	51.3	40.8	**96.96**	26.041	40
Normalized GOPS/bank [f]	4.25	3.45	**8.0**	3.9	8.0
Computation unit placement	SA	PSA	PSA	Bank IO	Bank IO
SA modified	Yes	No	No	No	No
Seamless Integration	No	No	No	Yes	Yes

[a] In C/D architecture variant of MCDRAM, the computation units are integrated near the bank peripheral (32 INT8 MAC units per bank).

[b] Computation banks + storage banks

[c] Single bank has 16 Sub-Arrays, each of dimension 256×2048. All Sub-Arrays operate in parallel, hence equivalent to 4KB page size

[d] Peak Power. For a fair comparison, McDRAMs peak power is re-evaluated with the energy numbers presented in Table 4.2 (i.e.$E_{256bitRead} * freq * Numofbanks$). Please note that the computation unit power is excluded for McDRAMs and only read power is considered in this calculation.

[e] 1 operation = 1 MAC (except UPMEM). [f] Normalized to 2KB page size - linear scaling and normalized based on number of computation banks.

4.5 Summary & Discussion

This architecture was the first attempt in this thesis to address Challenges 1, 2, & 3 (i.e. to increase the throughput and energy efficiency per bank by placing the computation units near the PSA). However, this architecture has further room for optimizations:

- Retraining: The one-hot quantization requires retraining to recover the accuracy loss. Result section 4.4.1 shows that the accuracy loss after retraining is less than 1% for CNN. However, this might not be the case for all other networks like LSTM, Transformers, etc. Although, the architecture is also feasible for two-hot quantization, more research is required for its ubiquitous applicability for all networks.
- Integration: The proposed digital approach architecture integrated several new units in the PSA region i.e. four CSS blocks, SHLs, result latch, adder and shifters. In contrary, a typical PSA region in commodity DRAMs only has PSA, equalizer, and CSS blocks. As a result, the area overhead per Sub-Array in this architecture is approximately 30%. This will also increase the LBL length by 30%, which will have adverse impact on the sensing. Furthermore, this will also impact the seamless integration of computation units in the PSA region.

Hence, I propose another new DRAM-PIM architecture in Chapter 5.

Chapter 5

DRAM PIM Architecture for DNN Inference - Mixed-Signal Approach

In this chapter, I present my second DRAM-PIM architecture for DNN inference to address the limitations of my first architecture and to further increase energy efficiency. This architecture is based on a mixed-signal approach as compared to the previous all digital approach. To the best of my knowledge, this is the first DRAM-PIM architecture to propose a mixed signal approach. The bit-decomposed MAC and *Weighted Current Summation* (WCS) techniques are employed to implement the MAC units in the PSA region using only AND gates and wired accumulation. By employing the analog MAC units, this architecture exploits the PSA region's analog behavior to achieve the highest energy efficiency compared to all DRAM-PIM state-of-the-art (i.e. 151 GOPS/W) and also results in extremely low area overhead per Sub-Array of 4%. The area intensive *Analog to Digital Convertor* (ADC) are integrated in the bank peripheral region, so the area overhead in the PSA region is maintained low. Furthermore, the bit-decomposed MAC does not require any retraining, unlike one-hot quantization employed in my first architecture. Similar to previous architecture, this architecture also aims to be compatible with commodity DRAM architectures to support the Dual-Mode feature, incur minimal modifications to the existing circuits, exploit the advantages of PSA region for high energy efficiency without violating its area constraints, and finally enable seamless integration (i.e. Goal 1 & 2, Chapter 1). The contributions of this chapter are previously published in [64].

5.1 Background: Bit-Decomposed MAC

This section describes the generic idea of Bit-Decomposed MAC using CNN as an example, but it is applicable to any type of DNN. CNNs are a type of DNNs that employ distinct kernel windows to extract positional features from the input data. The kernels slide over the input feature map in a raster scan manner and compute MAC operations for each position.

Table 5.1: Architecture 2: Example bit-decomposed MAC operation for $1 \times 1 \times 4$ kernel, $W_p = 3$, $I_p = 3$, $\{w_{000}, w_{001}, w_{002}, w_{003}\}$ = {2, 1, -3, -1} and $\{x_{000}, x_{001}, x_{002}, x_{003}\}$ = {0, -2, 1, 3}.

Clk	Unit 0	Unit 1	Unit 2	Partial MAC output
1	$\sum_{c=0}^{3} x_{00c}[0]$ $\&w_{00c}[0]$ **= 2**	$\sum_{c=0}^{3}(x_{00c}[0]$ $\&w_{00c}[1])$ $<< 1$ **= 2**	$((\sum_{c=0}^{3}(x_{00c}[0]$ $\&w_{00c}[2])$ $<< 2)\cdot(-1))\cdot 1$ **= -8**	**-4**
2	$\sum_{c=0}^{3}(x_{00c}[1]$ $\&w_{00c}[0])$ $<< 1$ **= 4**	$\sum_{c=0}^{3}(x_{00c}[1]$ $\&w_{00c}[1])$ $<< 2$ **= 4**	$((\sum_{c=0}^{3}(x_{00c}[1]$ $\&w_{00c}[2])$ $<< 3)\cdot(-1))\cdot 1$ **= -8**	**0**
3	$((\sum_{c=0}^{3}(x_{00c}[2]$ $\&w_{00c}[0])$ $<< 2)\cdot 1)\cdot(-1)$ **= -4**	$((\sum_{c=0}^{3}(x_{00c}[2]$ $\&w_{00c}[1])$ $<< 3)\cdot 1)\cdot(-1)$ **= 0**	$((\sum_{c=0}^{3}(x_{00c}[2]$ $\&w_{00c}[2])$ $<< 4)\cdot(-1))\cdot(-1)$ **= 0**	**-4**
Final MAC output				**-8**

The input feature map of an arbitrary convolution layer is of dimensions (W, H, C), where W and H are the spatial size of the input feature map and C is the number of input channels. The kernel window has the dimension (I, J, C) where I, J are spatial size of the kernel and C is the number of channels. Each convolution layer has K such kernel windows. The output feature map calculation is based on Equation 5.1, where w is a single element from the kernel window or weight matrix, x is a single element of the input feature map, f is the activation function, and b_k is the bias.

$$Y_{m,n,k} = f\left(\sum_{i=0}^{I-1}\sum_{j=0}^{J-1}\sum_{c=0}^{C-1} w_{i,j,c,k}\cdot x_{m+i,n+j,c} + b_k\right) \tag{5.1}$$

$$\begin{aligned} Y_{m,n,k} = f\Bigg(\Bigg(\sum_{v=0}^{I_p-1}\Bigg(\sum_{u=0}^{W_p-1}\Bigg(\sum_{i=0}^{I-1}\sum_{j=0}^{J-1}\sum_{c=0}^{C-1}(x_{m+i,n+j,c}[v] \\ \&w_{i,j,c,k}[u]) << (u+v)\Bigg)\cdot w_s\Bigg)\cdot i_s\Bigg) + b_n\Bigg) \end{aligned} \tag{5.2}$$

$$w_s = \begin{cases} +1 & u < W_p - 1 \\ -1 & u = W_p - 1 \end{cases}$$

$$i_s = \begin{cases} +1 & v < I_p - 1 \\ -1 & v = I_p - 1 \end{cases}$$

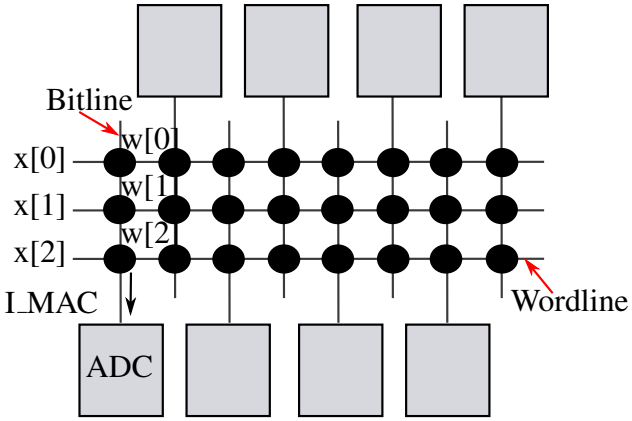

Figure 5.1: Basic implementation of bit-decomposed WCS MAC in emerging memories.

Equation 5.2 shows the MAC operation decomposed to the bit-level operations, where W_p and I_p are the precision of weights and input data, respectively. To extend this technique to the 2s complement numbers format, i_s and w_s are included in the Equation 5.2. This bit-decomposition of the MAC operation enables a new level of parallelism for CNN computations. Additionally, it also enables the exploitation of sparsity that exists at a bit-level to facilitate a new set of approximations and computation power reduction. Table 5.1 shows the example of bit-decomposed MAC operation for the kernel and input data dimension of $1 \times 1 \times 4$. In this example, $W_p = 3$, $I_p = 3$, $\{w_{000}, w_{001}, w_{002}, w_{003}\}$ = {2, 1, -3, -1} and $\{x_{000}, x_{001}, x_{002}, x_{003}\}$ = {0, -2, 1, 3}. The number of cycles required to finish the bit-decomposed MAC depends on the number of parallel computation units, which is also three in this case. This example considers a kernel of dimension $1 \times 1 \times 4$, but the same example can be extended to other dimension kernels by converting $I \times J \times C$ kernels to $1 \times 1 \times (C \cdot I \cdot J)$ kernels.

5.2 Mixed Signal DRAM-PIM Architecture

The bit-decomposed MAC converts the multiplication operation to an AND operation, and the WCS technique replaces the digital adder with a wired current-based adder. The bit-decomposed WCS MAC is employed in many SRAM and emerging memory-based PIM architectures [138–140]. Figure 5.1 shows the basic principle of SRAM/emerging memory PIM architectures that employ the bit-decomposed WCS MAC. The data input bits 'X' are concurrently applied to multiple wordlines and the weight bits are stored in the cells. The memory cell compute the AND operation ($x[v]$ & $w[u]$) and accumulate a current on bitlines that correspond to the partial MAC output. Such an architectural approach cannot be adapted to a DRAM-based PIM because of the following drawbacks: (1) enabling multiple rows in a DRAM will result in charge accumulation on bitlines and the charge stored in the DRAM cells are very small. As a result, it is very hard to differentiate the multiple levels.

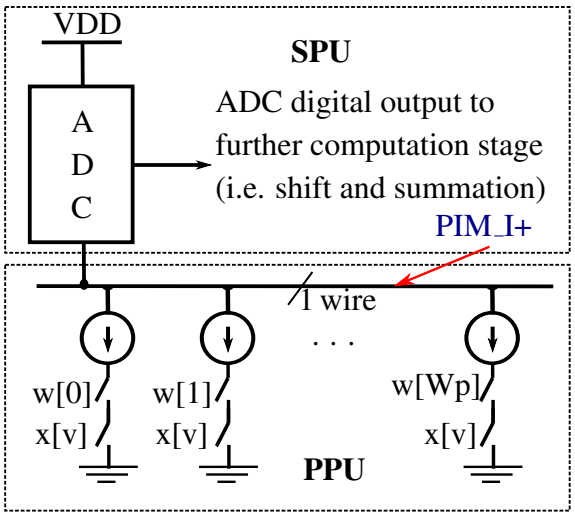

Figure 5.2: Architecture 2: Basic implementation of bit-decomposed WCS MAC in this DRAM-PIM architecture.

(2) ADCs are to be integrated in the area-constrained PSA region to convert the analog MAC outputs on the bitlines to the digital values for further processing. The integration of ADCs in the area-constrained PSA region is highly challenging. Hence, *the entire architecture has to be rethought and have to adapt a very distinct architecture approach for DRAM-based PIMs to fully exploit the advantage of the bit-decomposed WCS MAC (i.e. low area overhead for multiplication operation).*

In this section, I present the novel mixed-signal based DRAM PIM architecture for 8-bit DNN inference. As aforementioned, the architecture relies on the bit-decomposed WCS MAC technique to eliminate the integration of the area-intensive standard MAC units in the PSA region. Additionally, it is designed to eliminate the need for integrating the ADCs in the PSA region. Similar to my first architecture, the DNN computation unit is partitioned into two entities (refer Figure 5.2), 1) a lean analog Primary Processing Unit (PPU) integrated in the PSA region and 1) a larger mixed-signal Secondary Processing Unit (SPU) integrated near the bank periphery. This split between the first and second entities is required because the surroundings of the DRAM Sub-Array is constrained by limited area, which is not the case near the bank periphery. I first describe the design of the analog PPU, followed by the bank-level and device-level architecture. Lastly, I will also show that this architecture enables the seamless integration of computation units in the commodity DRAM architecture. Similar to my first architecture, this architecture also exploits the inherent parallelism of the commodity DRAM structure, i.e. bank-level and sub-array level parallelism, for enhancing the computation throughput.

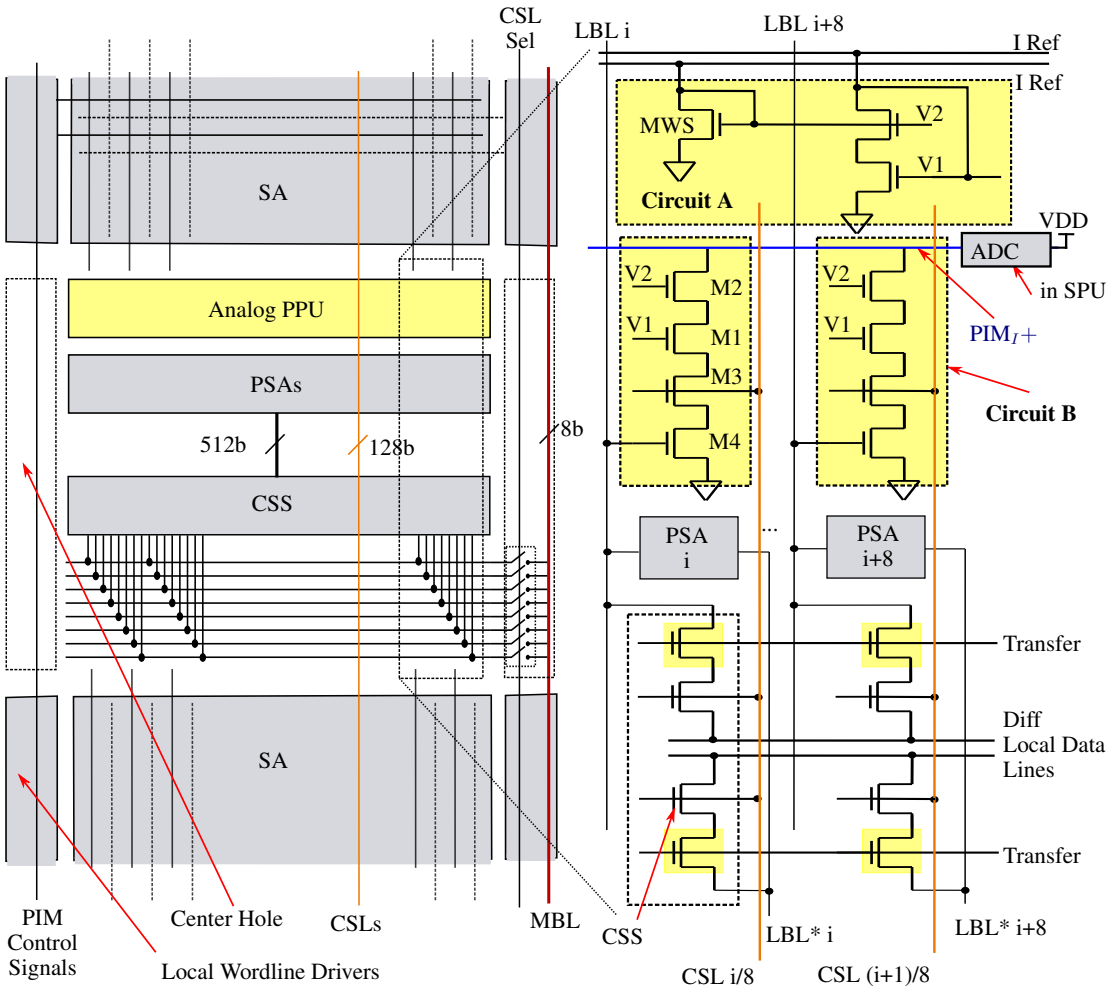

Figure 5.3: Architecture 2: Sub-Array-level Architecture with analog PPU. The yellow blocks show the new circuits. For image clarity, the connection of LBL* to PSA is not shown in left figure. In this figure Sub-Array is abbreviated as SA.

5.2.1 PPU Architecture

The analog PPU is responsible for performing the bit-wise AND operation and the wired current accumulation operation, which are the primary operations of bit-decomposed WCS MAC (refer Section 5.1). Figure 5.2 shows the basic implementation of bit-decomposed WCS MAC. If the AND operation output $x[v]$ & $w[u]$ is true for an arbitrary bit position, then the analog PPU will sink a constant amount of current (I_{out}) from the PIM_I+ line connected to VDD through ADC. The cumulative current on a PIM_I+ line due to concurrent current sink from multiple bit positions corresponds to the partial MAC output of the bit-decomposed WCS MAC. Figure 5.3 shows the Sub-Array-level architecture and the detailed analog PPU design, where the yellow blocks represent the new additional blocks to the existing PSA region. Prior to the start of computation, the weight bits $w[u]$ that are

stored in the Sub-Array are latched in the PSAs using a standard DRAM activate command (i.e. ACT). The input feature map data bits $x[v]$ are driven from the SSA region by reusing the CSLs wires (explained in Section 5.2.2). The computation unit is enabled after sensing of $w[u]$ by PSAs and driving of input feature map bits on . The transistors M3 and M4 compute the AND of $w[u]$ bit available on LBL and $x[v]$ available on . In order to sink a fixed amount of current (i.e. I_{out}) from each bit position, the analog PPU relies on current mirror circuits, i.e., Circuit A and B (refer Figure 5.3). A typical wide-swing cascode current mirror circuit is employed in this architecture [141]. "Circuit A" is responsible for generating a gate voltages to the "Circuit B", which is equivalent to a reference current I-Ref. These gate voltages ensure that the current through the "Circuit B" (i.e. I_{out}) is equal to I-Ref, provided both transistors M3 and M4 are enabled (i.e. $x[v]$ *&* $w[u]$). The analog PPU is enabled by turning on the I-Ref generator circuit and disabling the transfer signal. Please note, the cell charge degradation does not impact the PIM_I+ current, as the analog PPU input is from the PSAs whose output is at a stable value of VDD or ground after sensing cell charge using ACT command.

The proposed PPU micro architecture is very similar to the exsisting CSS block. Figure 5.4 compares to conventional CSS block with the analog PPU's "Circuit B" block. In conventional DRAM each CSL line is connected to the eight *Column Select Switches* (CSS) corresponding to eight bit positions of the PSAs, i.e., if a single CSL line is enabled the CSS will connect eight PSA outputs to the respective eight differential local datalines, which is further connected to MBLs. Similarly, the analog PPU also connects one CSL input (or data input in computation mode) to eight "Circuit Bs". This corresponds to the parallel computation of multiple outputs that have the same input feature map data but different weight inputs (refer Equation 5.2). The data mapping is discussed in detailed in Section 5.2.5. An additional similarity to the CSS is the integration of one "Circuit B" unit per LBL or per PSA bit position (refer Figure 5.3). However, the "Circuit A" unit is shared by multiple "Circuit B" units (e.g., eight). The sizing of all the transistors in the PPU are the same as the CSS transistor, except for the MWS transistor (refer "Circuit A") that has the same width but a larger length (e.g., $4\times$). As a result, the MWS transistor is constructed by concatenating multiple transistors of the same length as the CSS transistor. The additional control logic required at SA-level is integrated in the center hole area (refer Figure 5.3 for center hole location in the PSA region).

5.2.2 Bank-level Architecture

Figure 5.5 shows the bank-level architecture. The bank architecture supports two modes of operation, i.e. transfer mode and compute mode. In transfer mode, all the data paths from a conventional DRAM architecture is preserved, i.e., PSA $\rightarrow$ CSS $\rightarrow$ MBL $\rightarrow$ SSA and vice-versa. In computation mode, the data path is as follows: PSA $\rightarrow$ PPU $\rightarrow$ PIM_I+ $\rightarrow$ SPU. Initially, the weights w are stored in the Sub-Arrays. The computation starts after the sensing of the weights stored in the Sub-Arrays by their respective PSAs. E.g. 1024

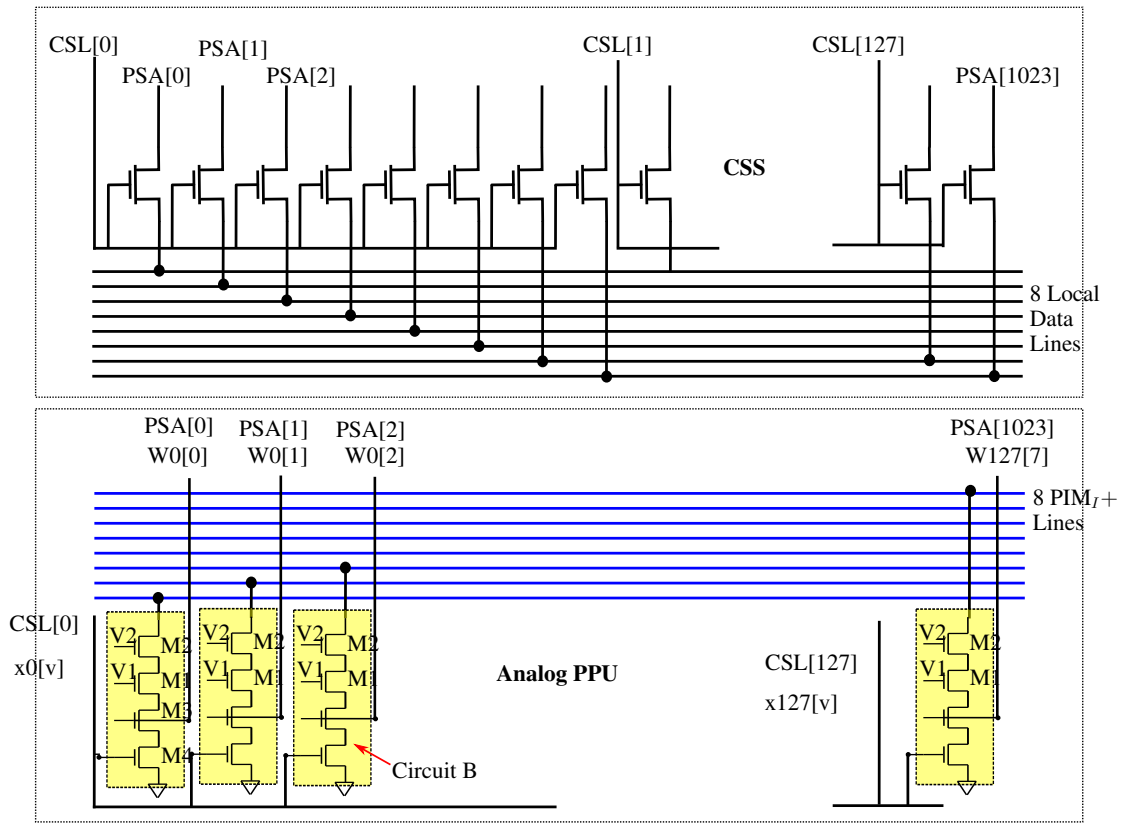

Figure 5.4: Architecture 2: (Top) CSS connection to local data lines. (Bottom) Hardwired connection between "Circuit B" and PIM_I+ lines. Shows the similarity between the CSS connection and "Circuit B" connection. For the ease of representation, all PSA output are shown on same side, but in the actual design the "Circuit B" and CSS are interleaved on both side of Sub-Arrays. An example input data and weights available on PSA and CSL, respectively, is shown is green color.

weight bits are sensed per Sub-Array. This sensing process is the same as in conventional DRAM Sub-Arrays (i.e. ACT) and require a time of tRCD (i.e., row to column delay, refer Chapter 2 for all timing definitions). After the sensing time tRCD, the input feature map data obtained from the external memory controller or via bank-bank transfer (explained in Section 5.2.3) is applied to the CSL from the SSA region. The weight bits latched in the PSAs are ANDed with the corresponding CSL bits in the analog PPU. Depending on the number of logic-1 outputs of the AND operation, the analog PPU will results in the equivalent cumulative current on a PIM_I+ line. This means that the energy consumption of the analog PPU is proportional to the number of logic-1 outputs. The cumulative current on a PIM_I+ line is then sensed by the ADC, whose output is transferred to the digital computation units for further processing. There are eight PIM_I+ lines per H-Block (a H-Block is a row of sub-arrays) and all the analog PPU units in the H-Block shares these eight PIM_I+ lines (refer Figure 5.5). The connection between the analog PPU unit and the PIM_I+ lines is hardwired, every eighth bit position is connected to the same PIM_I+

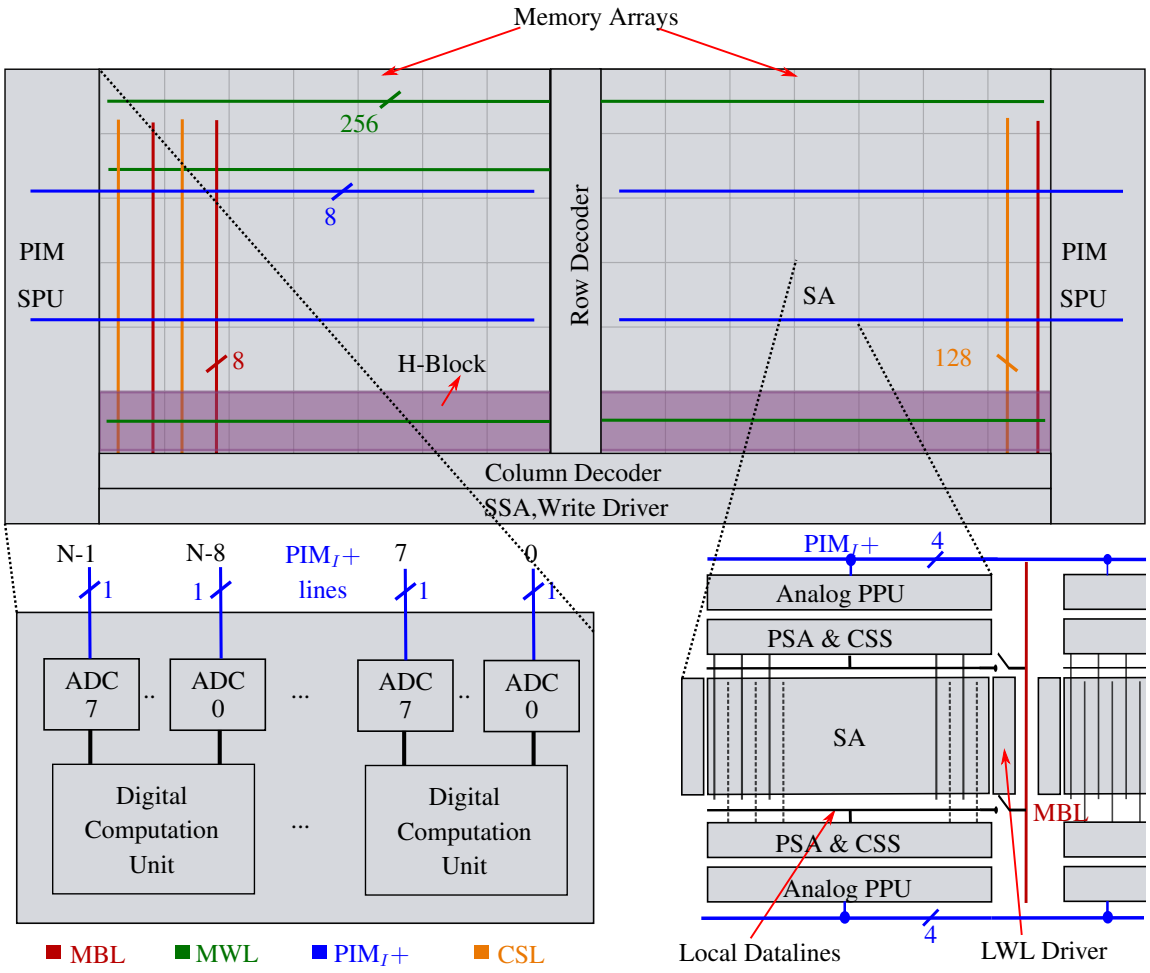

Figure 5.5: Architecture 2: Bank-level mixed-signal PIM Architecture. In this figure Sub-Array is abbreviated as SA.

(refer Figure 5.4). This pattern is similar to the PSA/CSS to local dataline connection. By following the same connection pattern for computation and storage (i.e CSS to local dataline), the need for additional data re-arrangement units is eliminated. Since the bank in this architecture is constructed using two memory arrays (refer Figure 5.5) that exhibit certain degree of parallelism, it is exploited to increase the parallelism within the bank. The H-block is partitioned into two individual blocks (i.e. one in either memory array), and the analog PPUs in each half of the H-Block will have their individual eight PIM_I+ lines. Hence, there are sixteen PIM_I+ lines per H-Block, i.e., eight per half of the H-block. The architecture also offers flexibility to dynamically vary the maximum number of bits ('C') accumulated on each PIM_I+ line. This is achieved by driving only 'C' CSL with the data input. The value of 'C' is configured to the number of input data channels 'C' (refer Section 5.1) of a given convolution layer, e.g., 3 for shallow layers and 128 or 256 for deep layers. I restrict the value of 'C' to a maximum value of 128 to avoid employing ADCs with

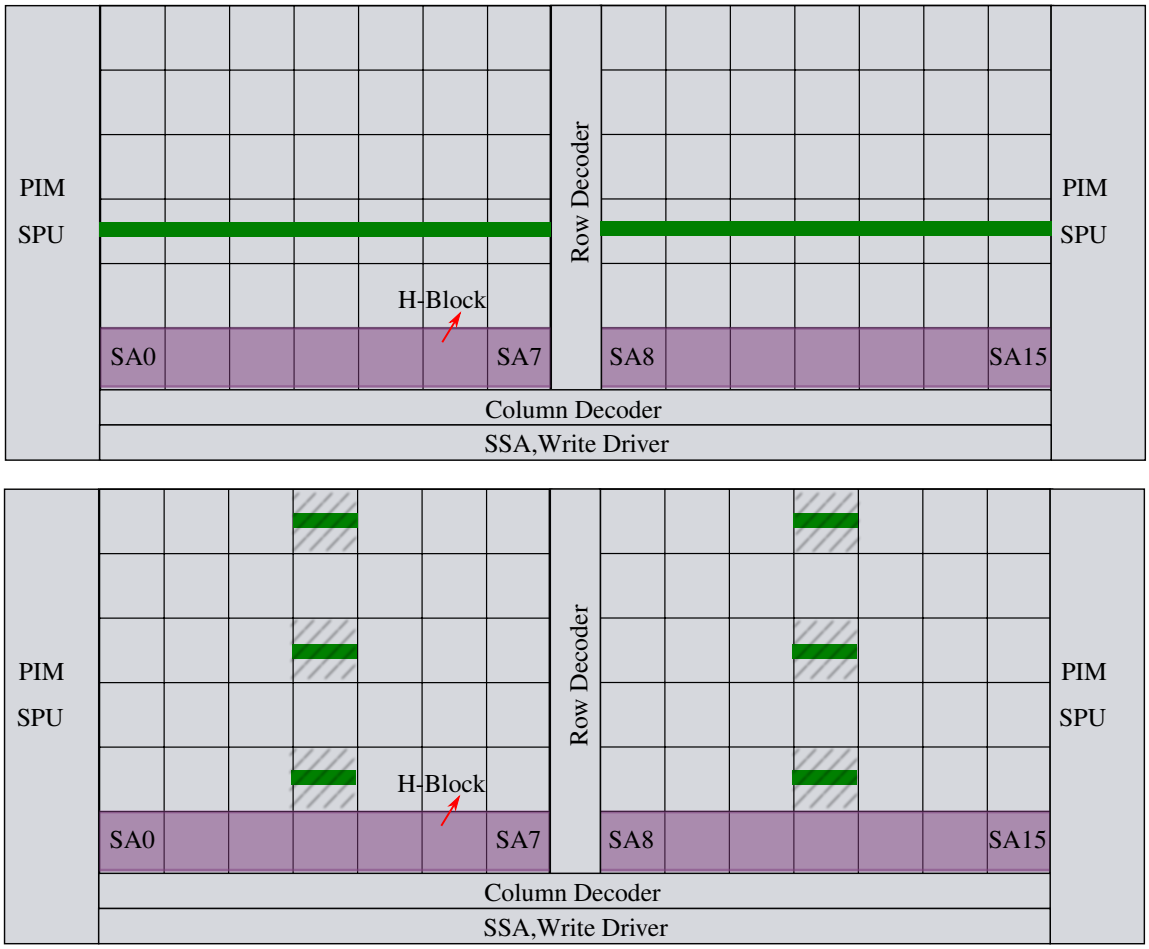

Figure 5.6: Architecture 2: Conventional activation (top) vs Vertical activation for P=2 and Q=3 (bottom). In this figure Sub-Array is abbreviated as SA.

higher bit precision that are power and area hungry. Hence, multiple iterations are required for CNN inference with larger 'C' values.

Typically, in a DRAM, one H-Block is enabled per bank (i.e., one active row) at any given instance of time. As a result, there exist only sixteen concurrently active PIM_I+ lines. In order to further increase the parallelism within the bank i.e., more than sixteen active PIM_I+ lines, this architecture proposes a new activation technique referred as *Vertical Activation* (VACT), where 'P' number of Sub-Arrays in each of the 'Q' H-Blocks are activated. Figure 5.6 shows the example for P=2, Q=3. In this example, SA-4 and SA-12 (i.e. stripped Sub-Arrays) in three H-blocks are activated instead of 16 Sub-Arrays activated in a single H-blocks. The 'P' Sub-Arrays in each of the 'Q' H-block are aligned in column direction and none of the 'Q' H-blocks are neighbors to the other due to sharing of PSA region. The architecture employs one control signal per column of Sub-Arrays to enable/disable the LWLD and PSA circuits of the desired Sub-Arrays in an H-block. Furthermore, a minor modification to the row decoder circuit to enable multiple MWLs ensures

the concurrent activation of 'P' Sub-Arrays in 'Q' H-Blocks. Hence, a higher parallelism of $Q \times 16$ active PIM_I+ lines is obtained per bank. The parallelism within the bank is on-the-fly configured by overwriting the 'P' and 'Q' values in the mode registers. Please note, at a Sub-Array level the operation of the vertical activation and the associated precharge is the same as conventional activation and precharge without requiring any modifications to the PSA or LWLDs. The vertical activation also does not disrupt the cell charge restoration process, which is automatically done during the activation operation, or the conventional refresh process. This architecture maintains the maximum product of 'P' and 'Q' to be equal to the number of Sub-Arrays per H-block (e.g. 16) so that the total activation energy is the same as conventional counterparts. Additionally, the row decoder modification in this case is less critical because, (1) row decoder is in the peripheral region of the bank and (2) row decoder has to enable the same row is each of the 'P' Sub-Arrays in all 'Q' H-blocks.

The architecture restricts the number of H-Blocks integrated with the analog PPU to eight, where the total number of H-Blocks per DRAM bank is in the range of 32 to 64 for an 8Gb device. The vertical activation and precharge feature is also confined to these eight H-Blocks (i.e. PIM enhanced H-Blocks). The total amount of memory available for CNN inference per bank is equal to $PageSize \times SAHeight \times 8 \times 2$ (e.g. $2KB \times 1024 \times 8 \times 2 = 32MB$), $\times 2$ is due to the PSA region sharing with neighboring H-blocks. This restriction ensures that the total die area overhead is low and maintain a high memory density similar to the commodity DRAMs. Retaining the high memory density gives a distinct advantage to this novel architecture in terms of yield. *If the PIM blocks in an arbitrary die or set of dies are not functional due to permanent manufacturing faults in computation units, these dies can still be used as conventional high memory density DRAMs. This is possible only because the architecture preserved all the data paths and the control logic from the conventional DRAM. Hence, the architecture retains the yield of the underlying commodity DRAM technology.* This architecture also makes an effort to retains all the routing paths of their conventional counterparts, i.e., the new blocks will not impact the original routing. The new PIM_I+ lines are routed in the same metal layer as MWL and is routed over the analog PPU circuitry without having to modify the layout/macro of any other convectional circuits in the PSA region. The macros have to be shifted in vertical direction to make space for analog PPU. Note that, DRAM technology is very conservative with limited number of metal layers (e.g., three or four) and such decisions play a major factor for industrial acceptance of the novel PIM architectures.

SPU: The Secondary Processing Unit (SPU) consists of ADC and digital computation blocks (refer Figure 5.5). The SPU is integrated vertically on either sides of the bank. Each SPU has one current sensing 5-bit ADC [140] per PIM_I+ line and a digital computation unit (referred as stage2-computation) for every eight ADCs. The stage2-computation block of the SPU consists of shifters, 32-bit adder, 2s Complement unit, ReLU activation function and a quantization unit. The SPU also includes a current source array to generate the reference current to the ADCs. This architecture employs a 5-bit current based flash ADC, which is presented in our previous work [140]. Employing a low bit precision 5-bit ADC

ensures that the power contribution by the ADC on the overall device's power is low. The use of 5-bit ADC in this architecture is based on the statistical analysis of CNN kernels. Our analysis shows that the CNN kernels exhibit a very high sparsity for most networks. Hence, a 5-bit ADC that is capable of sensing I_{Ref} to $32 \times I_{Ref}$ is sufficient for all the possible values of 'C'. In Section 5.5, we show the impact of 5-bit ADC on CNN inference accuracy to be minimal. The statistical analysis and approximation techniques to employ 5-bit ADC instead of 8-bit is beyond the scope of this thesis. These techniques and the ADC design are elaborated in detail in the PhD thesis of my colleague Taha Soliman. In case of the network that does not exhibit such high sparsity or approximation is not possible, then the 'C' value is configured accordingly to the lower value (e.g., 64 or 32). In other words, the usage of this architecture is not solely confined to networks for which the approximation is possible.

5.2.3 Device-Level Architecture

The banks in a DRAM device are capable of operating in parallel. This PIM architecture exploits this inherent parallelism to enable concurrent computations. The banks are configurable to function either as a computation bank or storage bank (i.e. PPU disabled, behaves like a convention DRAM bank). The storage banks are used for temporary buffering of output feature maps of an arbitrary DNN layer that is required for the computations of the next layer. The architecture also supports the internal transfer of the data between the computation banks and storage banks. The internal 128-bit data bus that connects the bank output with the device IO is reused for bank-bank communication as well. All the banks have two ping-pong buffers of size 8×128 that is used to buffer the input and output feature map data required for the computations/storage during the next eight cycles (i.e., to support pipelining). For the external control of the PIM operations, four new commands are introduced: (1) vertical activation - VACT (2) vertical precharge - VPRE (3) bank-bank transfer - BBR or BBW and (4) compute - CM. The vertical activation command specifies the row address 'A' (e.g., 10 bits) that has to be activated within a Sub-Array and the 5-bit configuration. Depending on the 'P' and 'Q' values configured in the mode register, the possible vertical activation configurations are preloaded in the mode registers. E.g. in the case of P=2 and Q=8, the configuration 5'b0000 may specify the activation of A^{th} row in SA0 and SA8 (i.e., Sub-Array in column zero and eight) in all eight PIM H-Blocks. The preloaded configuration ensures that the VACT command is specific to the H-Blocks integrated with the analog PPUs and equal number of Sub-Arrays are activated in both the memory arrays for P$>$1. Such preloading of configurations is used in commodity DRAMs as well. The bank-bank command BBR/BBW is similar to a convention DRAM read/write command (RD/WR), except that the data is transferred between the banks. Hence, the bank-bank command requires the activation (ACT) and precharge (PRE) of the desired row in the storage bank prior to the transfer. The compute command specifies the number of inputs to be driven by the CSL (i.e., 'C' value) and the bit position to be selected from the 8×128

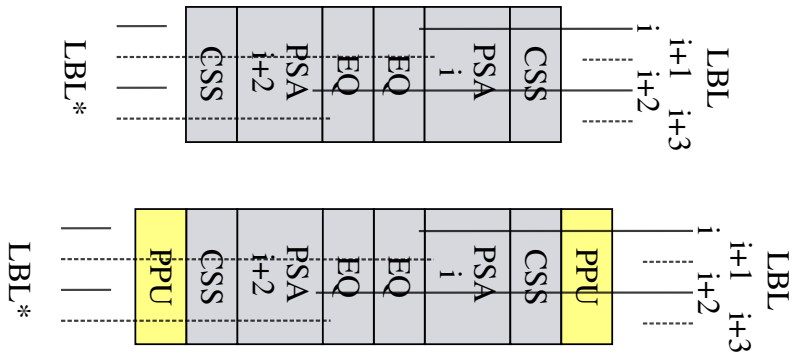

Figure 5.7: Architecture 2: Comparing the example PSA region floorplan of conventional DRAM with DRAM-PIM.

buffer as an input data (i.e $x[v]$), which is applied to CSLs. Any of these new commands can be optionally sent to multiple banks for simultaneous operation. The concurrent multi-bank operation is employed in other type of DRAM-PIMs as well, e.g. FIMDRAM [34]. The architecture also allows the possibility to write to the ping-pong buffer from an external memory controller using the conventional write command. All the new operations in this PIM architecture follow a specific timing, similar to conventional DRAM operations. Therefore, the external memory controller has to respect the specified timings before performing the next operations, which results in ease of control flow. The timings for vertical activation, vertical precharge and bank-bank transfer follow the same timings as the conventional activation, precharge and read/write operations, respectively (refer Section 5.5.3). This is because these operations are similar at a Sub-Array-level.

5.2.4 PSA Region Floorplan

In this section, I describe the ease of integration of the analog PPU without modifying the design and layout of the PSA. Figure 5.7 compares the example floorplans of the conventional PSA region (i.e., including the CSS block and equalizer block (EQ)) and PIM architecture's PSA region. For the sake of simplicity, only single side of theSub-Arrayis shown in Figure 5.7, while the same blocks are present on the either sides of the Sub-Array. In conventional DRAMs, the width of each PSA is equal to four LBL pitch. As a result, two PSA circuits and its respective CSS blocks are placed one after the other on each side of the Sub-Array. The alternating LBLs and reference LBLs (LBL*) pass over the CSS block to connect to the respective PSAs. The LBLs over the CSS block maintain a spacing of two LBL metal pitch. These metal gaps are utilized by the CSS block for other relevant routing and vias. Similar to CSS block design, the analog PPU block is also partitioned into two blocks, constructed using the same transistor as CSS block, and use the LBL metal gaps for its routing. The PPU blocks are deliberately integrated next to the CSS block or at the borders of the PSA region so that the rest of the macros are preserved.

The key factors that make the PPU design highly feasible to be integrated near the Sub-Array or in the PSA region in comparison to other prior DRAM-PIM architectures [29, 32] or my digital approach architecture are: 1) high similarities between the CSS block and current mirror "Circuit B", 2) low design complexity, 3) very low area overhead. Additionally, partitioning the analog PPU into two blocks is also one of the key architectural decisions that maintain the increase of LBL/LBL length, due to integration of new blocks, to be uniform and results in a balanced load on the PSA. Hence, the proposed architecture design has ensured the maximum reuse of existing macros. The only modification that is required to the existing PSA region design is the inclusion of the additional enable transistors (for transfer signal) in series with the CSS (refer Figure 5.3), which is relatively trivial compared to the modification of the whole PSA macros.*

5.2.5 Data Mapping

Figure 5.8 shows the example data mapping of an 8-bit $1 \times 1 \times 128$ weight kernel in a Sub-Array. The input feature map data of the same dimension (i.e., $1 \times 1 \times 128$) is applied on CSLs, but one bit position (out of eight) per iteration. Hence, each Sub-Array computes 128 MAC operations for every eight computation cycles. A single bank is capable of concurrently computing 16 such kernel operations, assuming P=2 and Q=8. For simplicity reason, Figure 5.8 shows all the PSAs, PPU and PIM_I+ lines on the same side of the Sub-Array. However, these units are interleaved on either sides of the Sub-Array in the actual design (refer Figure 5.5). For the ease of representation, Figure 5.8 shows the mapping of an example kernel of dimension $1 \times 1 \times 128$, but the same data mapping is applicable for other dimension kernels as well, e.g. $I \times J \times C$ kernel can be converted to $1 \times 1 \times (C \cdot I \cdot J)$ kernel. Although, most of the analyses in this chapter is for the configuration P=2, Q=8 and C=128, I retain P, Q and C as reconfigurable parameters for exploring other data mapping strategies in the future.

5.3 Experimental Setup

My PIM architecture is constructed on top of a state-of-the-art 8 Gb commodity DRAM architecture with 8 banks, 64 H-Blocks per bank, 16 sub-arrays per H-Block, a 1024×1024 standard sub-array dimension, and a 2 KB page. The area of the commodity DRAM Sub-Array is calculated using the reverse engineered unit cell dimensions, LBL pitch, and LWL pitch (e.g. 66 nm) of 2y nm technology (i.e. between 25 nm to 27 nm) based 8Gb DRAM dies [117], [118]. Please note, DRAM technology is typically represented like 2y, 1y ($\approx$17 nm), 1z ($\approx$14 nm), etc. The area of the other standard DRAM circuits (e.g., decoders, sense amplifiers, etc.) is evaluated using the DRAM architecture exploration tool DRAMSpec [119]. The required 2y nm DRAM technology input parameters for DRAMSpec are extracted from multiple sources and articles [48, 117, 118, 120–124]. To provide an

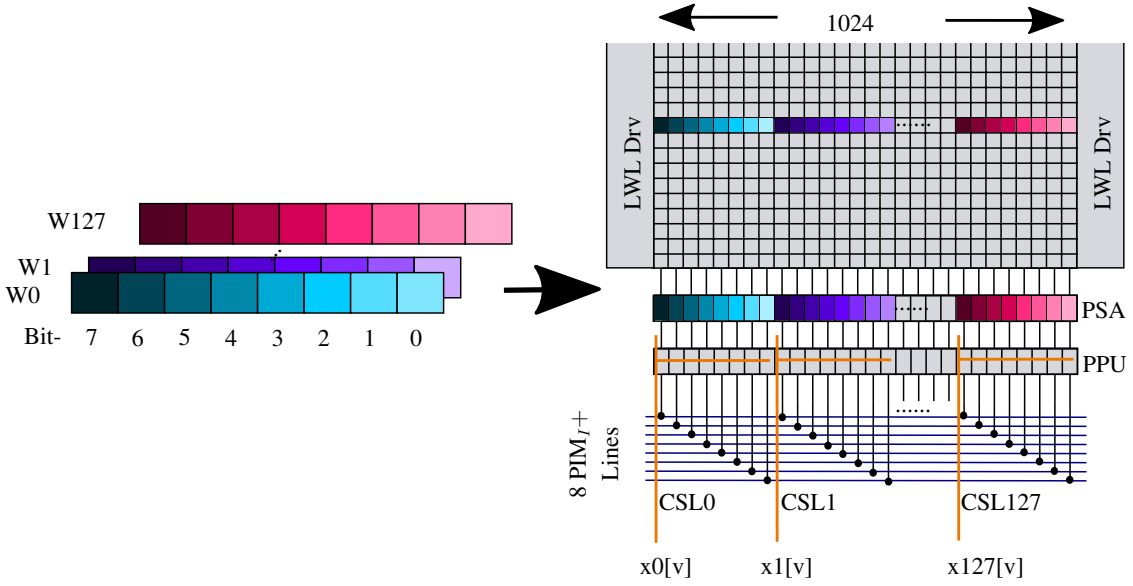

Figure 5.8: Architecture 2: Example mapping of $1 \times 1 \times 128$ kernel to a 1024 cell wide Sub-Array.

indication of the correctness of the area evaluation methodology, I compare the estimated 8 Gb standard DRAM die area calculated with this methodology (i.e., 74.44 mm^2) to the area of 8 Gb commodity DRAM dies from memory vendors i.e. Samsung's DDR4 (i.e., 59.00 mm^2) [118], Hynix's DDR4 Gen-1 (i.e., 76.0 mm^2) [117] and Micron's DDR4 (i.e., 58.48 mm^2) [125].

The latency and energy of the DRAM circuits are estimated for a core-clk of 250 MHz using an in-house enhanced circuit-level DRAM model (SPICE) that can be equipped with different kinds of SPICE transistor models and PDKs, such as FreePDK [126] with PTMs [127], Cadence PDK models, and as well proprietary DRAM transistor models. This DRAM model includes the wire parasitics extracted from Qimonda's 45nm DDR3 design. The circuit-level results and timing correctness of this model are presented in [128] using BSIM4 model cards built on Low Power PTM transistor models that are adapted to DRAM technology. We updated this DRAM model to 2ynm DRAM technology based on the scaling results presented in [129]. The PPU evaluation is also performed in SPICE using the same wire and transistor models as the DRAM model. The PIM_I+ wire parasitic resistance of 1.2 Ohms/um and capacitance of 0.18 fF/um is considered for latency and energy calculation of PPU. These values are same as MWL resistance and capacitance, since both the lines are routed on same metal layer.

The area, latency and energy evaluation of the PIM SPU computation units are performed using an adapted STM 28 nm Low-Leakage CMOS technology. For example, the number of metal layers to three in order to align with DRAM technology The computation unit area, power and timing results are extracted post place and route (Synopsis' IC Compiler). The length of PPU has to be the same as Sub-Array and the height of PPU

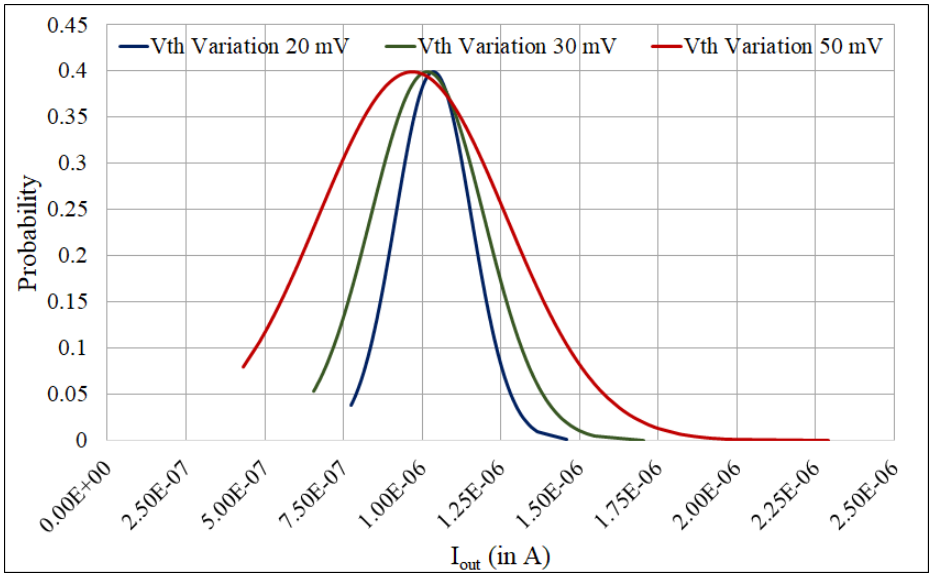

Figure 5.9: Architecture 2: Impact of threshold (Vth) variation on a single current mirror output - 500 Monte-Carlo simulations.

is manually calculated based on the worstcase height dimension of the CSL transistors (i.e. 500 nm).

The CNN benchmarks used in this chapter are ResNet20 [131], SqueezeNet [132] and VGG16 [133]. ResNet20 is trained with the CIFAR10 dataset [134] and all the other networks are trained with the ImageNet dataset [135]. The simulation framework ProxSim [142] is used to evaluate the CNN accuracies with the presented approximations and the possible architecture variations. The overall system-level performance and power consumption for the proposed architecture is evaluated with a behavioral level Python model.

5.4 Variation Analysis

In this section, I present the impact of device variations in analog PPU on the final convolution result. Such an analysis is very crucial, as the output of the current mirror is highly sensitive to the device variations such as channel length, channel width, oxide thickness, threshold voltage, etc. The analog PPU design minimizes the impact of variations by duplicating the current mirror ”Circuit A” for every eight ”Circuit Bs” (refer Figure 5.3), thereby ensuring the devices within a given current mirror circuit to exhibit high spatial correlation. Please note, ”Circuit A” and ”Circuit B” together forms a current mirror circuit. Although, the high spatial correlation minimizes the device-device sizing variation, but the threshold voltage variation due to random dopant fluctuations (RDF) is persistent. Hence, this section predominantly presents the threshold voltage variation analysis.

Prior to the threshold variation analysis, the foremost analysis I perform is to verify the linear relation between the cumulative current on PIM_I+ line and number of logic-1 out-

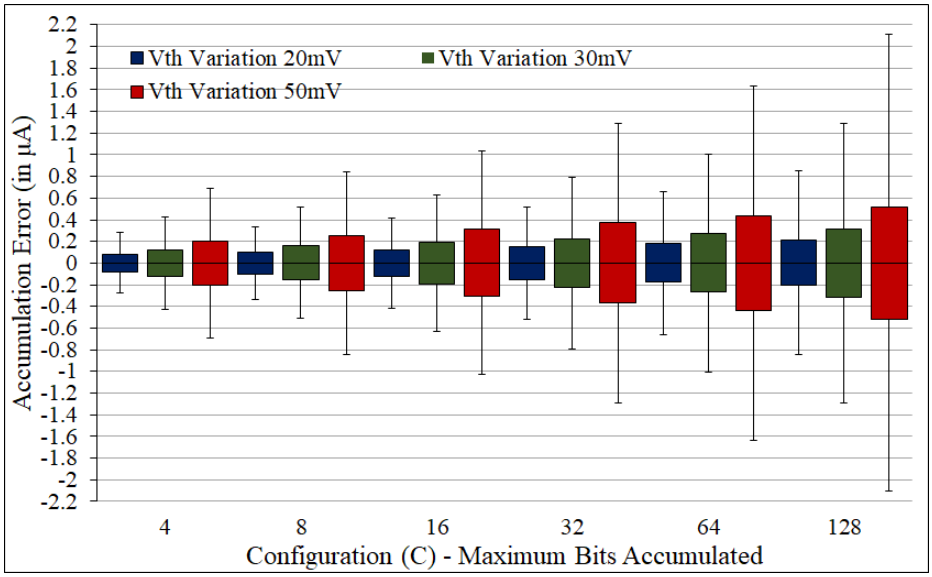

Figure 5.10: Architecture 2: Impact of threshold (Vth) variation on PIM_I+ line current accumulation results - 10000 simulations of convolution operation using WCS-based MAC. The min and max value corresponds to 5th and 95th percentile, respectively.

puts of the AND operation. The SPICE simulations of PPU, including the wire resistance and capacitance, confirm the expected linear behavior for an example scenario where 32 "Circuit B" outputs are connected to a single PIM_I+ line. This linear behavior is disrupted mainly due to threshold voltage variations. The impact of threshold voltage variation in the current mirror circuit is reduced by replacing the NMOS M1 and M2 of "Circuit B" (refer Figure 5.3) with N equal sized series connected NMOS transistors (same dimention as CSS transistor). Hence, the variation effects are averaged and the deviation across the current mirror circuits' output I_{out} is minimized. The rest of the results in this paper are for such an adapted current mirror circuit, where 'N' is equal to four. Figure 5.9 shows the probability distribution curves of a single current mirror circuit output I_{out} based on 500 Monte-Carlo simulations for the maximum threshold voltage variation of 20 mV, 30 mV and 50 mV, where the mean I_{out} is $\approx 1\,\mu A$. The distributions are then substituted to the analog PPU equivalent python model to compute the current accumulation error on a PIM_I+ lines. In each iteration, the model calculates the current accumulation error on PIM_I+ lines and propagates the accumulation value to the ADCs (python submodule) to calculate the impact of the variation on the digital output of the ADC. The rest of the results in this section are based on 10000 simulations of convolution operation using this model. The model is simulated for six different configurations (C), where each configuration indicates the maximum number of bits accumulated per iteration on a PIM_I+ line. In a CNN inference, the value of 'C' corresponds to the depth of the input feature map of a particular layer, which influences the active number of CSLs driven with the data input in this architecture (assuming the data mapping presented in Section 5.2.5 is employed). Therefore, it is important to analyze the error behavior for different layers i.e. shallow layers and deep

Table 5.2: Architecture 2: ADC output absolute error - 10000 simulations of convolution operation. Median, 95th percentile and max error are shown. *The error values are in absolute values and not in bits*.

		Configuration (C)					
		4	**8**	**16**	**32**	**64**	**128**
ADC output - **Error** (20 mV Vth variation)	Median	0	0	0	0	0	0
	95th	0	0	1	1	1	1
	max	1	1	2	3	5	6
ADC output - **Error** (30 mV Vth variation)	Median	0	0	0	0	0	0
	95th	0	1	1	1	1	2
	max	1	2	3	5	7	9
ADC output - **Error** (50 mV Vth variation)	Median	0	0	0	0	0	0
	95th	1	1	1	2	2	3
	max	2	3	6	9	11	17

layers. Figure 5.10 shows the box plot of accumulation error on a single PIM_I+ line due to transitor variation of current mirror circuit. Please note that the x-axis of Figure 5.10 indicates only the maximum number of ones to be accumulated, the actual number of ones in each iteration are randomly chosen by the model to mimic the realistic scenario. Table 5.2 summarizes the deviation of ADC digital output from the expected value due to the transistor variations. The median error value is at zero, and the 95th percentile error can go up to 3, which corresponds to 2-bit error. In worst cases scenarios the ADC output error is at very high value of 17 (i.e. 4-bits), but the probability of these values are very low. E.g. the probability of error value 17 of C=128 and 50 mV Vth variation is 2.13×10^{-7}. Similarly, for error value = 1, C=4 and Vth variation = 20 mV, probability of error is 4.6×10^{-5}. The results in Table 5.2 are considering the ADCs of bit precision $\log_2(C)$, but a 5-bit ADC is considered for the accuracy calculations.

Calibration: The effects of device variations in the analog PPU are further addressed in SPU block as well. During the DRAM calibration phase, the predefined weights and data are used as input to the analog PPU and the output of the ADC is measured for each PIM_I+ line and each analog PPU. The ADCs and its references are then tuned appropriately to avoid the propagation of accumulation error to the digital blocks. Furthermore, measuring the output of the ADCs using predefined input also enables the identification of suitable bias for the network, which can further mitigate the impacts of variations on convolution output. These techniques address not only the device variation issues, but also the aging effects. Since the calibration is done seldom, the time consumed to finish the calibration phase is not important. Additionally, the identification of the predefined pattern is beyond the scope of this paper.

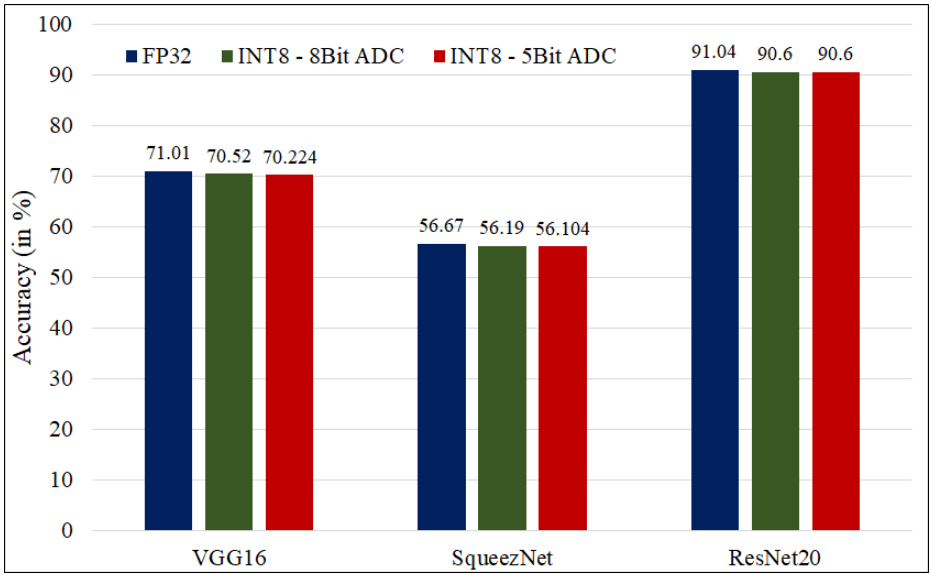

Figure 5.11: Architecture 2: CNN inference accuracy (in %) comparison for various networks. INT8-5Bit ADC and INT8-8Bit ADC corresponds to the accuracy of the proposed PIM device with 5-bit and 8-bit ADC, respectively. INT8-5Bit ADC and INT8-8Bit accuracy results consider 30 mV threshold variation.

5.5 Results

In this section, I present the experimental results of the new mixed-signal DRAM-PIM architecture. All the evaluations are for the following configurations 8-bit input feature map data, 8-bit weights, P=2, Q=8, six computations banks and two storage banks. As mentioned in Chapter 1, this architecture research work published in [64] was conducted in close collaboration with my colleague M.Sc. Taha Soliman from CR/AEX2, Robert Bosch GmbH. All the architectural investigations, design choices, floorplan analysis, Sub-Array level data mapping, variation analysis, calculation of circuit-level results like area, latency and energy consumption, and DRAM-PIM architecture's peak performance/power-consumption calculations are my individual contributions. The algorithmic approximations, accuracy results, and the behavioral python-model implementation of this architecture, which is required for the network specific DRAM-PIM performance and energy efficiency evaluation, are in collaboration with M.Sc. Taha Soliman.

5.5.1 Accuracy Evaluation

The foremost result presented in this chapter is the CNN inference accuracy achieved by the proposed 8-bit precision PIM architecture with 5-bit ADC (i.e. labeled as INT8-5Bit ADC) in comparison to the 32-bit floating-point (FP32) operation (refer Figure 5.11). The accumulation error distribution presented in Figure 5.10 is propagated to the ProxSim framework [142] for CNN accuracy degradation calculation. The accuracy results presented in

Table 5.3: Architecture 2: Area evaluation results of all the building blocks of Standard DRAM with PIM and the mixed-signal based DRAM-PIM

Block	**Area** $Height \times Width$ [μm^2]
Sub-array (SA) [a]	83.68×74.036
PSAs + CSS ($512\times$), per side of Sub-Array	10×74.036
Sub-array with PSAs [b]	93.68×74.036
Column decoder, SSAs	125×1444.58
Row decoder	6329.2×250
Bank without computation units [a]	6329.2×1444.58
8 Gb Standard DRAM Die	**74.4442 (mm^2)**
Analog PPU, per side of Sub-Array	4×74.036
PIM SPU - per side of bank	$BankHeight \times 50$
Bank - PIM (8-MAC-rows)	6673.56×1544.576
8 Gb PIM Die (8-MAC-rows)	**83.8526 (mm^2)**
Overhead	**12.63%**

[a] including redundancy
[b] SA area considers only single side of PSA due to PSA sharing

figure are for 30 mV threshold variation. Figure 5.11 also presents the accuracy degradation by the proposed PIM device integrated with 8-bit ADC and 30 mV threshold variation. In all the cases, the accuracy degradation by the proposed PIM device is marginal (i.e. $< 1\%$). The further sub-sections presents the area, performance and the power consumption results of my novel PIM architecture.

5.5.2 Area Evaluation

Based on the methodology described in Section 5.3, the area evaluations of all the relevant circuits of the proposed PIM architecture are shown in Table 5.3. For instance, the area of the 5-bit ADC is $4.6\,\mu m^2$ and the stage2-computation block is $397.55\,\mu m^2$. This height of the SPU should be equal to the bank height, and the width of the SPU is conservatively estimated as $50\,\mu m$, which is sufficient for all the ADCs, reference current generator and stage2-computation blocks. Hence, the resulting area overhead of the proposed 8 Gb DRAM PIM die in comparison to standard commodity 8 Gb DRAM is 12.63%. This is a moderate value compared to Samsung's FIMDRAM die area overhead of $\approx 40\%$. In comparison to the digital approach architecture, the mixed-signal architecture reduces the area overhead per sub-array from 30% to 4% and it also effectively addresses the area constraint challenge of PSA region.

Table 5.4: Architecture 2: Timing and Latency of the new PIM commands. A few timings in JEDEC DRAM specification are presented in terms of interface clock cycles (Ck) that is equal to $4\times$ core clock cycle of the proposed architecture. Assumption: CWL = 12Ck, CL=14Ck, BL=8.

Operation	Equivalent conventional DDR4 DRAM Operation	Latency,[ns]
VACT	ACT	12
VPRE	PRE	10
VACT - VACT [a]	ACT - ACT	45
BBR - BBR	RD - RD	4
BBW - BBW	WR - WR	4
BBR - BBW	RD - WR	8
BBW - BBR	WR - RD	18
CM	-	8

[a] Same bank

5.5.3 Performance Evaluation

Table 5.4 presents the latencies of various operations that are required for the calculation of the overall performance. The latency for computation command CM (i.e. t_{CM}) is 8 ns, where 2 ns is for CSL charging to VDD, 4 ns is for current accumulation on PIM_I+ lines and ADC sensing, and 2 ns is for CSL discharging to zero. In a conventional DRAM bank RD operation, CSLs are enabled only for half the core clock period (i.e. 1 core clock = 4 ns), allowing another half for CSLs to be discharged to zero before performing the next RD. Hence, I consider the worst case charging and discharging duration of CSLs to be 2 ns. An additional reason for considering worst case CSL charging duration is that the gate terminal of the transistor M3 connected to CSL in "Circuit B" has to be tied to VDD for the proper functioning of the current mirror and sink a current of I_{out}. During the charging and discharging of the CSLs the ADCs are disabled or are in standby mode. Equation 5.3 shows the peak performance calculation for C=128, where $t_{compute}$ is the total computation time (including the VACT and VPRE latencies) required to finish 'C' MAC operations using eight PIM_I+ lines. Based on the data mapping methodology described in Section 5.2.5, eight CM cycles are required for an 8-bit MAC operation (i.e., input data and weight precision is 8-bit), as a result $t_{compute}$ is equal to $t_{VACT} + 8 \times t_{CM} + t_{VPRE}$. The multiplication of factor two in Equation 5.3 is due to the fact that the two memory arrays within the bank operate parallelly (i.e. P=2). Hence, in $t_{compute}$ time period a total of $128 \times 2 \times 8$ MAC operations are performed. The resulting peak performance of a PIM bank is 23.813 Gops, where one operation is equal to one MAC. Equation 5.3 did not consider the latency of stage2-computation or bank-bank transfer because of the pipeline structure, i.e.

Table 5.5: Architecture 2: Energy consumption of various operations for the core clock frequency of 250 MHz.

Operation	Energy/Power
ACT (2KB page)	614 pJ
PRE (2KB page)	314 pJ
Read/Write (PSA-SSA) - 128-bit	418/438 pJ
CSL + Driver	1.5 pJ/bit
Bank to Bank communication (BB) [a]	2 pJ/bit
ADC [b]	1.272 mW
Stage2-Computation(St2-C) [b]	0.26 mW
Device Background Power	34mW

[a] Excluding read/write within the bank
[b] Single unit

PPU+ADC, stage2-computation and bank-bank transfer operate in parallel. Equation 5.3 considered only the longest latency i.e. PPU+ADC, which defines the peak performance of this architecture. However, the contribution of all the units is considered in power calculations presented in Section 5.5.4. Figure 5.12 shows the performance achieved in the proposed novel PIM architecture for various networks obtained from the behavioral-level python model. This behavioral-level python model calculates the utilization of computation units using a network-specific data mapping. The peak performance and peak power of the proposed architecture is then scaled by this python model as per the utilization percentage to obtain the achieved performance and power results for a given network. The highest performance of 115.321 Gops is observed for the VGG16 network.

$$\begin{aligned} GOPS_{Bank} &= (C \cdot 2 \cdot Q)/t_{compute} \\ &= (C \cdot 2 \cdot Q)/(t_{VACT} + 8 \cdot t_{CM} + t_{VPRE}) \\ GOPS_{Bank} &= 23.813\, GOPS \end{aligned} \tag{5.3}$$

5.5.4 Power Evaluation

Table 5.5 summarizes the energy/power consumption of all the operations of the proposed PIM-device. For instance, activating a 2KB bank page (row) consumes around 614 pJ, which is same for VACT with P=2 and Q=8. The energy for BBR/BBW commands is equal to read/write energy plus bank-bank communication energy. Further, the use of 5-bit ADC has resulted in a low ADC power consumption of 1.27 mW for 250 MHz frequency.

These energy and latency results are used to calculate the overall power of the devices.

$$\begin{aligned} P_{PPUperBank} =& VDD(I^{*}_{out} \cdot NumOfOnesPerPIMLine \\ & \cdot NumOfPIMLinesPerHblock \\ & \cdot Q \cdot 2) + ((E_{CSL} \cdot C \cdot 2)/t_{compute}) \end{aligned} \tag{5.4}$$

$$\begin{aligned} P_{device} =& (((E_{VACT} + E_{VPRE})/t_{compute}) \\ & + P_{PPUperBank} + (2 \cdot Q \cdot P_{St2-C}) \\ & + (2 \cdot Q \cdot (P_{ADC}/2))) \cdot 6 + (8 \cdot P_{BBR}) \\ & + (6/9) \cdot P_{BBW}) + P_{Background} \\ P_{device} =& 945.76\,mW \end{aligned} \tag{5.5}$$

$$Efficiency_{device} = 151.07\,GOPS/W \tag{5.6}$$

Equation 5.4 shows the power consumption of all the PPUs in a bank, where I^{*}_{out} is the mean current sink of a single current mirror circuit. The multiplication of factor two in Equation 5.4 is due to the fact that the proposed architecture considers two memory arrays per bank. Equation 5.5 shows the peak power evaluation for the entire device with six computation banks and two storage banks. Only half of the ADC power is considered in Equation 5.5, because the ADCs are active only for 4 ns out of t_{CM} duration of 8 ns. The contribution from the storage banks to the total power consumption is based on the number of reads and write performed in the storage banks for the duration $t_{compute}$. Eight C-bits are read from the storage bank in a single $t_{compute}$ duration to performs the computations (considering input data precision is 8-bit), which results in eight bank-bank reads (BBRs) per $t_{compute}$ duration. Typically, these eight BBRs are performed in the previous $t_{compute}$ duration and the input data are buffered in ping-pong buffer (refer Section 5.2.3). Each computation bank writes $2 \times Q$ output data (8-bit) in every nine $t_{compute}$ cycles for a $3 \times 3 \times C$ kernel. This results in a total of 6 bank-bank writes (BBWs) from six computation banks in nine $t_{compute}$ cycles. Hence, Equation 5.5 considers $8 \times P_{BBR}$ and $(6/9) \times P_{BBW}$, where P_{BBR} and P_{BBR} are the power consumption of a single 128 bit bank-bank read and write, respectively (E.g. $P_{BBR} = (E_{RD} + (128 \times E_{BB}))/t_{compute}$). The peak power consumption of the proposed PIM device is 945.76 mW, where the value of 'C' is configured to 128. Hence, the energy efficiency of the novel mixed-signal DRAM-PIM architecture is 151.07 GOPS/W. The power consumption results of the proposed PIM devices for various CNN network obtained from the behavioral-level python model is shown in Figure 5.12.

5.5.5 Comparison

In this section, I compare the proposed DRAM-PIM architectures of this thesis with six well-known state-of-the-art DRAM-PIM architectures that are published till-date i.e. LAcc [31],

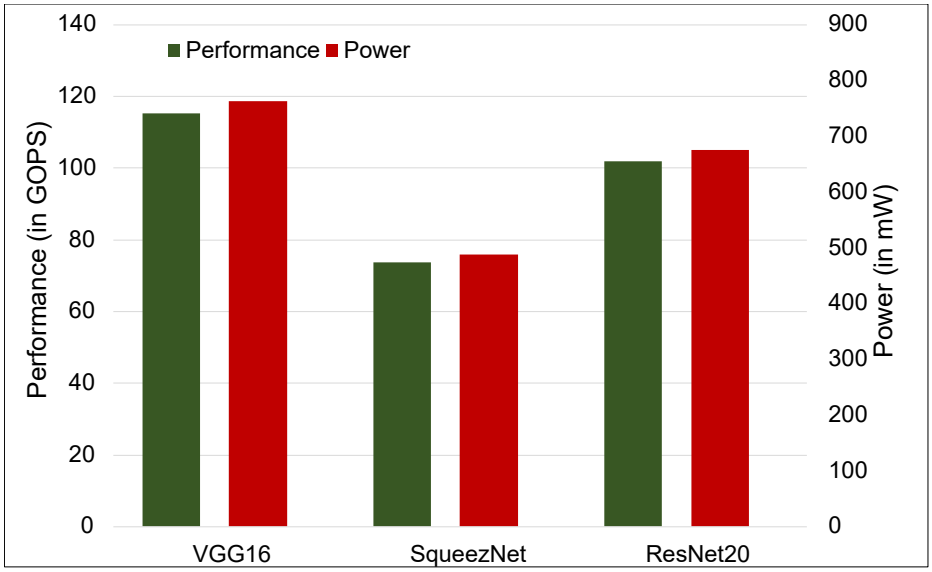

Figure 5.12: Architecture 2: Performance (in GOPS) and power consumption (in mW) of the proposed mixed-signal DRAM-PIM architecture for various CNNs.

SCOPE [32], UPMEM [35] and McDRAM [39], FIMDRAM [20,34] and Hynix-AiM [37] (refer Table 5.6). The proposed mixed-signal based PIM architecture achieves the highest throughput of 23.813 GOPS per DRAM bank, which is $4.96\times$, $2.97\times$, $1.4\times$ and $2.97\times$ higher than Samsung's FIMDRAM (HBM2 based), McDRAM(C/D) (LPDDR5 based), Hynix's AiM (GDDR6 based) and the proposed digital approach architecture, respectively. The area overhead of the mixed-signal architecture is 12.63% per DRAM die. Although, this value is higher than many of the prior works, but it is still a moderate value compared to the FIMDRAM ($\approx 40\%$). The engineering samples of FIMDRAM published by Samsung in [34] showed that the DRAM vendors are willing to tolerate higher area overhead. In comparison to the my digital architecture, the mixed-signal architecture reduces the area overhead per Sub-Array from 30% to 4%. The mixed-signal architecture effectively intigrated the computation units within the area constraints of PSA region. This architecture also enabled seamless integration of new computation unit in the PSA region with minimal modifications in comparison to digital architecture.

The power consumption of both the proposed architecture is within the accepted range of a commodity DRAM devices and its packaging (i.e., $< 1\,W$). The proposed PIM architectures additionally achieve the highest efficiency compared to state-of-the-art DRAM-PIM architectures, i.e ([31, 32, 35, 39]).

The fair comparison of the proposed architectures to SCOPE [32] is challenging because SCOPE is designed with 1024 highly modified banks to achieve a performance that is comparable to GPU. On the other hand, my aim is to design a DRAM-PIM architectures that are capable of functioning both as DNN accelerator and standard DRAM (or host memory) without any major modification to the bank structure and design. LAcc [31] consume a very high power that is $5.88\times$ higher than the mixed-signal architecture. LAcc [31] pro-

posed to perform bit-wise multiplication and bit-wise addition using special rows in each Sub-Array. This approach require a high number of activate and precharge operations for computing add operation (i.e. 20 ACT and PRE), which is the major contributors to the high power consumption.

5.6 Summary

In this chapter, I proposed a novel mixed-signal DRAM-based PIM architecture that relies on a bit-decomposed WCS technique to realize the DNN inference. To the best of my knowledge, this is the first DRAM-PIM architecture to adopt a mixed-signal approach. The use of analog current mirror-based MAC units ensured that the area overhead at the PSA region is low. It also facilitated the seamless integration of computation units (i.e. analog PPUs) in the PSA region. Overall, this architecture results in minor modifications to the highly optimized PSA region, and all the substantial modifications were confined to the bank periphery. The evaluation results show that the architecture is extremely energy efficient (i.e. 151.07 GOPS/W) and achieves the highest performance of 23.813 GOPS/bank. Additionally, the presented PIM architecture for an 8Gb device also exhibits a low area overhead of 12.63% with a comparable memory density as commodity DRAMs. I also presented the impact of device variations in the analog PPUs on final convolution results and various techniques to mitigate the error due to such variations. The key downside of this architecture is that it is only feasible for inference tasks and not for training. The DNN training typically require floating-point computation units, while the analog PPU is only capable of computing integer 8-bit MAC operation. Hence, new architectural approaches have to be explored to enable the integration of the floating-point MAC units in the PSA region without violating the area constraints, which is described in detail in Chapter 6.

Table 5.6: Comparison of DRAM based PIM architectures.

Parameter	LAcc [31]	SCOPE [32]	UPMEM [35]	Samsung		Hynix	This Work	
				FIM DRAM [34]	McDRAM [39] (C/D) [a]	AiM [37]	Digital [63]	Mixed [64]
DRAM process	25nm	22nm	2xnm	20nm	20nm	1ynm	2ynm	2ynm
Capacity per die [Gb]	8	8	4	4	8	4	8	8
Num of banks	32	1024	8	32	8	16	8 (4+4) [b]	8 (6+2) [b]
Page Size	4KB	4KB [c]	2KB	2KB	2KB	2KB	2KB	2KB
Operation precision	INT8	INT8	INT8	FP16	INT8	FP16	INT8	INT8
DRAM core frequency [MHz]	-	124	500	300	250	1000	250	250
Area [mm^2]	54.8	273.38	-	-	-	-	79.41	83.8526
Area overhead [%]	**1**	342.24	-	≈ 40	4.7	-	6.6	12.63
Power [W]	5.3	176.4	≈ 1.2	-	1.6 [d]	0.66	0.66	0.945 [d]
Power Density [W/cm^2]	9.67	64.53	-	-	-	-	0.83	0.9
Peak Performance per die [TOPS] [e]	0.27227	**7.2**	0.03125	0.1536	0.064	256	0.064	0.1428
Effi. [GOPS/W]	51.3	40.8	26.041	-	40	-	96.96	**151.07**
Normalized GOPS/bank [f]	4.25	3.45	3.9	4.8	8.0	16	8.0	**23.813**
Computation unit placement	SA	PSA	Bank IO	Bank IO	Bank IO	Bank IO	PSA	PSA
SA modified	Yes	No	No	No	No	No	No	No
Seamless Integration	No	No	Yes	Yes	Yes	Yes	No	Yes

[a] In C/D architecture variant of MCDRAM, the computation units are integrated near the bank peripheral (32 INT8 MAC units per bank).

[b] Computation banks + storage banks

[c] Single bank has 16 Sub-Arrays, each of dimension 256×2048. All Sub-Arrays operate in parallel, hence equivalent to 4KB page size

[d] Peak Power. For fair comparison, McDRAMs peak power is re-evaluated with the energy numbers presented in Table 5.5 (i.e.$E_{256bitRead} * freq * Numofbanks$). Please note that the computation unit power is excluded for McDRAMs and only read power is considered in this calculation.

[e] For fair comparison, 1 operation = 1 MAC (except UPMEM). FINDRAMs's TOPS result is converted accordingly, which is equal to $300\ Mhz \times 32 \times 16$ i.e. 32 PCUs per die, 16 MUL and ADD units per PCU or 16 MACs per PCU.

[f] normalized to 2KB page size - linear scaling

Chapter 6

DRAM PIM Architectural Approaches for Scalability

In this chapter, I propose new architectural approaches to enable the DRAM-PIM architectures for DNN training tasks and support other computation features. The computations of DNN training predominantly employ a floating-point format. However, the shifter based MAC units of architecture 1 and the bit-decomposed WCS based MAC units of architecture 2, integrated in the PSA region, are suitable only for integer computations and not for floating-point computations. The integration of floating-point MAC in the PSA region is infeasible due to its area constraints (refer Chapter 3 - Challenge 1). Hence, most state-of-the-art DRAM-PIM architectures either support only inference tasks or integrate the floating-point MAC units in the bank peripheral region. In this chapter, I propose a novel DRAM-PIM architectural approach called *Quasi Open Bitline* (QOBL) that enables the integration of floating-point MAC units in the PSA region without violating the area constraints. Similar to architectures 1 & 2, this approach also avoids modifying the Sub-Array and supports the Dual-Mode feature. Additionally, the architectures 1 & 2 only focused on Goal 1 (i.e. no Sub-Array modification) and Goal 2 (integration in PSA region), but this approach also suffices Goal 3, where the DRAM-PIM architecture is scalable for any new computation units in the PSA region without violating the area constraints.

In addition to the proposal of novel architecture that is scalable for any new computation units, the Goal 3 also targeted to investigate a unified and external control protocol that is compatible across the vendors. The computation commands proposed by each DRAM-PIM architecture vary from vendor to vendor. For example, Samsung's McDRAM [39] introduce 10 new commands and Hynix-AiM [37] introduced 16 new commands. This results in vendor-specific hardware modifications of DRAM PIM's external interface and the DRAM PIM's external memory controller, which does not align with the commodity counterparts. To address this issue, DRAM-PIM architectures from Samsung (i.e. FIM-DRAM [34]) and UPMEM [35] proposed architectures that employ the commodity DRAM interface and its memory controller without requiring any hardware modifications. How-

ever, their approach restricts the data access from a given DRAM-PIM while the computation is in progress. This stalls the data requests and inhibits the effective utilization of the Dual-Mode feature. Hence, I propose a new unified DRAM-PIM interface and external control protocol that is feasible for any DRAM-PIM architecture and is capable of concurrent processing of both data and computation requests. This new protocol allows a DRAM-PIM die from any vendor or generation to be a drop-in replacement of the other without modifying the interface or memory controller hardware design.

Finally, all the architectural approaches presents in this chapter address five out of six challenges highlighted in Chapter 3 (i.e. Challenge 1,2,3,4, and 6). The low bit-width data-type for DNN training (i.e. Challenge 5) is out of the scope of this thesis, which will be addressed in the thesis work of my colleague M.Sc. Mohammad Hassani Sadi from Microelectronic Systems Design Research Group, Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau. All the findings of this chapter are previously published in [65]. The main aim of this chapter is to investigate the most feasible future architectural directions/approaches for DRAM-PIM. However, the detailed architectural assessment with application evaluations and comparative study is reserved for future work due to the time period constraint of this thesis.

6.1 Architectural Approach for Inference + Training

In this section, I present a new DRAM-PIM architectural approach that is suitable for the computation of both inference and training tasks. In addition, this approach is scalable for accommodating other computation units without violating the area constraints of PSA region or compromising the memory density that impact the Dual-Mode goal.

6.1.1 PPU Architecture

As presented in Section 3.1.2 of Chapter 3, the PSA region imposes area constraints to the PIM computation units, especially to the area intensive multipliers (i.e. Challenge 1). One of the primary reasons for the area constraints in the PSA region is due to the widely employed *Open Bitline* (OBL) array structure and its PSA sharing. Hence, I present a new array structure called as *Quasi Open Bitline* (QOBL) that eliminate the PSA sharing and enable the integration of any PIM computation units. I first compare the two well-known commodity DRAM array architectures, i.e. *Folded Bitline* (FBL) and OBL, from the PIM perspective. Figure 6.1.a and 6.1.b show the legacy FBL and OBL array architectures. The key advantage of the OBL array architecture over FBL is its high memory density. As a result, almost all recent commodity DRAM bank architectures are designed as per OBL. In OBL, the LBL and reference LBL (LBLs*) inputs to the PSA (differnetial amplifier) are from the vertically neighboring Sub-Arrays, which result in mandatory PSA sharing. Due to this PSA sharing, the new PIM computation units in OBL has to retain the balanced

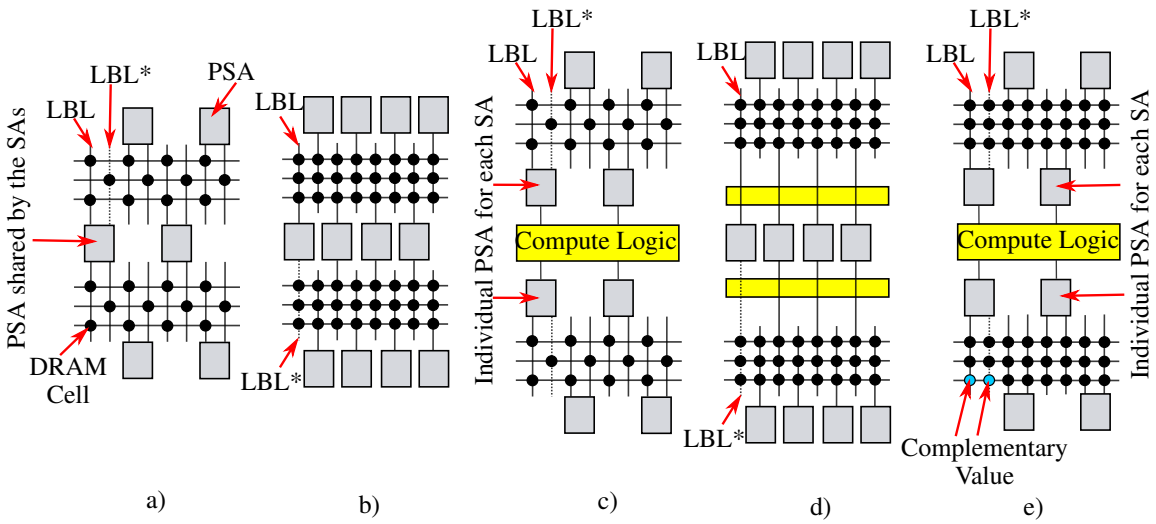

Figure 6.1: Comparing DRAM BitLine array architectures from PIM perspective: a) Conventional Folded BitLine (FBL) b) Conventional Open BitLine (OBL) c) Folded BitLine with new compute logic d) Open BitLine with new compute logic e) Quasi BitLine (QOBL) with new compute logic. QOBL eliminating the PSA sharing and enabling the integration of large computation units in the PSA region without modifying the Sub-Array design.

wire load of LBLs, reserve metal lines to allow LBLs/LBLs* to be connected to the respective PSAs, and ensure that the length of LBLs is not drastically increased beyond the consideration of PSA design (Figure 6.1.d). All these factors imposes area constraints to the new computation units, especially considering the number of metal layers in DRAM technology is three or four. On the contrary, LBL and reference LBL (LBLs*) in FBL are from the same sub-array. This allow FBL Sub-Arrays to have an individual set of PSAs without mandatory sharing of PSAs between the vertically neighboring Sub-Arrays (Figure 6.1.c). Hence, enabling larger space between the Sub-Arrays for the new computation units without encountering the OBL like area constraints. Additionally, the computation unit height neither impacts the LBL length nor requires any design/layout modifications to the existing PSA. Hence, FBL is more favorable choice for PIM while the OBL is for high density memory design. However, in order to suffice the Dual-mode feature, the DRAM-PIM architectures have to adapt OBL structure, despite the advantages of FBL. Almost all the recent commodity DRAM banks and Sub-Array are designed as per OBL structure to achieve the highest memory density. Consequently, it is important to reuse the OBL Sub-Array to be compatible with the commodity counterparts and avoid redesign of Sub-Array. Hence, I present a new QOBL array architecture for DRAM-PIM that reuses the existing OBL-based Sub-Array design to achieve the FBL-like advantages with OBL. Figure 6.1.e shows the QOBL array architecture that emulates the FBL structure using OBL arrays and its PSA design. It is important to note that QOBL does not require any modifications to the Sub-Array or PSA design of OBL except for a minor interconnection modification in

the PSA region, i.e., LBL to PSA interconnection. In QOBL, two consecutive LBLs from a Sub-Array are connected to the same PSA, where one of them is the reference line. This eliminates the dependency on the vertically neighboring Sub-Array and makes space for the new computation logic without area restrictions. Hence, mimicking the advantages of FBL with OBL arrays. By designing a DRAM-PIM bank with QOBL array type, floating point computation units that are feasible for both inference and training can be integrated in the PSA region. Furthermore, this approach is capable of supporting various other types of computation units without violating the area constraint of PSA region. Thereby sufficing the goal, ease of scaling the PSA region computation capabilities over the various generations of DRAM-PIM. One of the main disadvantages of QOBL is the reduction of the storage capacity of each Sub-Array to half. This is because QOBL uses two consecutive cells of same row for storing a single binary value, as compared to one cell in OBL. Furthermore, this capacity loss per bank is increased up to 75%. This calculation is based on the consideration that the area of computation unit integrated in the PSA region is the same as Sub-Array area and the computation units are integrated near all the Sub-Arrays of the bank. Hence, it is important to recover the capacity and overall cell density loss of QOBL in order to not impact the Dual-Mode. The device-level architectural approach to compensate the capacity loss of QOBL is presented in the next section. The circuit-level analysis of QOBL is proposed in a 2001 publication [143], but the authors only compared QOBL with OBL for LBL noise analysis. My work is the first to explore and propose the benefits of QOBL for DRAM-PIM architectures.

6.1.2 Device-Level Architecture

The current trend in the state-of-the-art DRAM-PIM architectures is to employ the same PIM bank design for both computation-mode and storage-mode. Figure 6.2.a shows an example DRAM-PIM architecture from Hynix [37], where all the banks are the same by design. Similar approach is employed in my DRAM-PIM architecture 1 & 2. Adapting this approach with QOBL array type incurs high loss of capacity (i.e. up to 75%), which will impact the dual-mode goal of this thesis. To address the capacity reduction of QOBL, I propose a DRAM-PIM device composed of two types of banks, i.e., conventional banks with OBL array type and PIM banks with QOBL array type. The conventional banks are used for high capacity data storage purpose, while the PIM bank is optimized for energy efficient computations. I first elaborate on the conventional bank design and followed by the PIM bank design. The conventional bank design is exactly the same as commodity DRAM bank design (i.e. with OBL array type). The commodity bank design with greater than or equal to 512 Mb of capacity is chosen as the conventional bank to leverage high capacity (E.g. LPDDR5, DDR5, LPDDR4, DDR4, DDR5 banks). As an example, Figure 6.2.b shows the bank capacity and the dimensions of a real 1y nm LPDDR5 commodity DRAM die [41], which is adapted as a conventional bank in this architecture (Figure 6.2.c). The computation banks are designed using QOBL array type where computation units are partitioned as

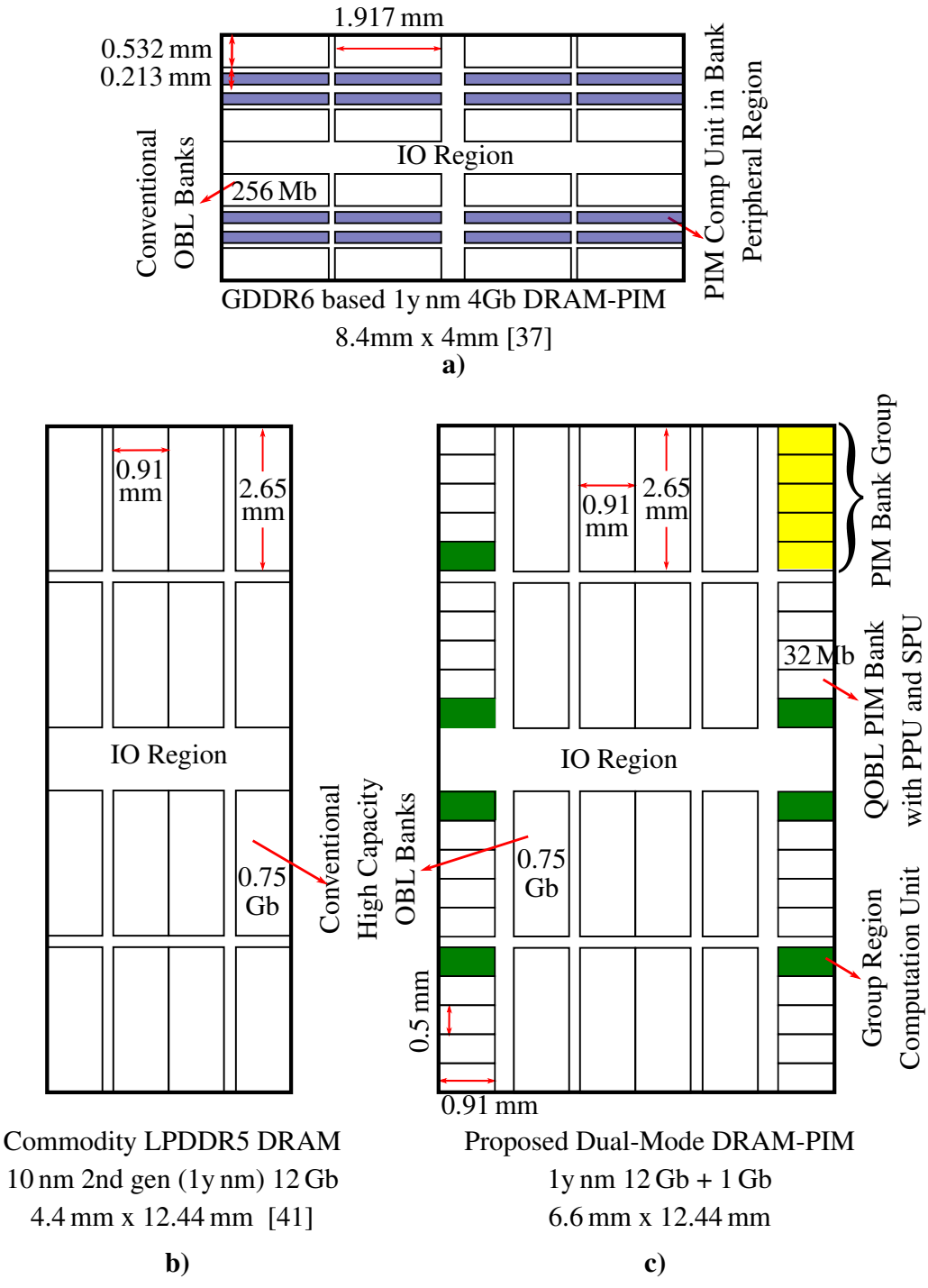

Figure 6.2: Device-level architecture comparison. a) State-of-the-art GDDR6 DRAM-PIM device with OBL array type bank. b) Commodity LPDDR4 DRAM device with OBL array type bank. c) Proposed DRAM-PIM with two types of banks i.e. conventional OBL based high capacity banks and new QOBL based PIM banks.

PPU and SPU. The capacity of these banks is targeted to be low (e.g., 32 Mb or 16 Mb), which results in much smaller bank height than conventional banks (Figure 6.2.c). In the example design shown in Figure 6.2.c, I consider the computation bank width of 0.91 mm, which is the same as conventional bank. The computation bank height of 0.5 mm for 32 Mb bank capacity in my architecture is a rough estimation that includes all the worst-case computation unit overheads. This value is similar to the bank height of 256 Mb GDDR6 based Hynix-AiM bank (i.e. 0.532 mm). These small bank dimensions enable the integration of a high number of computation banks (i.e., $>= 32$) in the given area (i.e., increased bank parallelism for computation), and the small bank height results in low data access energy, similar to GDDR6 based Hynix-AiM [37]. Hence, the JEDEC specifications that have to be satisfied for the normal memory operation are ensured by the conventional banks as it is the exact replica of commodity DRAM banks, while the high throughput demand and low bank internal data access requirement for energy-efficient computation is ensured by the PIM banks. This addresses Challenge 2 (i.e. increasing the computation throughput without decreasing the capacity or violating JEDEC specifications) and Challenge 3 (i.e. low data access energy within the bank), which is previously highlighted in Chapter 3. The further advantage of the proposed device-level architecture is that the DRAM-PIM die can be sold as a conventional DRAM die by disabling all the computation banks if in case one of the computation banks is damaged due to manufacturing defects. Additionally, the proposed DRAM-PIM architecture can be scaled for higher throughput or higher density by adjusting the number of computation banks and conventional banks. The die area of the proposed DRAM-PIM is estimated to be 79.2 mm^2 for 14 Gb (i.e. computation 1 Gb, conventional 12 Gb) that results in the density of 181.01 Mb/ mm^2. In comparison, Hynix-AiM has an area of 33.6 mm^2 for 4 Gb die that result in the density of 121.90 Mb/ mm^2 (i.e. 48% lesser than the proposed QOBL).

The high energy consumption of bank-to-bank communication (Challenge 4) is addressed by integrating the individual data buffers at a bank group level rather than one central global buffer at a IO region like in Hynix-AiM [37]. This is an extension to the partitioning approach presented in architecture 1 and 2 (i.e. PPU and SPU), where the shared data buffers and control units are integrated at a bank group-level (refer to the green block in Figure 6.2). This flexibility increases the data reusability within the bank group and reduces the frequent exchange of data through the IO region. The placement of a computation unit at the output of a bank group is also proposed in GradPIM [38], but this architecture placed all the computation unit in the bank group region without exploiting the bank parallelism within the bank group.

6.2 Unified Interface and Protocol for External Control

As explained in Section 3.3 of Chapter 3, the external control of DRAM-PIM is designed in one of the following approaches: 1) reuse the commodity DRAM interface and memory

controller, 2) new vendor-specific DRAM-PIM interface and memory controller. The first approach enables the DRAM-PIM devices to be a drop-in replacement to the conventional DRAMs without incurring any hardware modifications to the host processor, memory controller, or interface. In this approach, the conventional DRAM commands (e.g. Activate, write, and read) are used for both Non-PIM data access requests in normal memory mode and PIM requests in compute mode. Hence, the DRAM-PIM banks can either operate in a normal mode or in PIM compute mode at any given time. Figure 6.3 shows the example operation flow of Samsung's FIMDRAM [34] that employs approach 1. Initially, DRAM-PIM is in the normal mode, during which the host processor accesses the data similar to conventional DRAM. The compute engines are also programmed during normal mode using conventional write commands. The computation is started by changing the mode from normal to PIM. Once a given bank is in the computation mode, the host processor either has to wait until the finish of all the programmed computations or stop the ongoing computation to access the data from that bank to the host core. This limits the host processor's data access flexibility from the DRAM-PIM. In the second approach, the programming of DRAM-PIM is not by reusing the conventional DRAM commands, but rather the commodity DRAM command bus is extended to support the new PIM commands (e.g. ADD, MUL, MAC8). This allows the memory controller to alternate between the PIM request and Non-PIM request without having to wait until the whole program is finished. Hence, enabling the effective utilization of the Dual-Mode feature of DRAM-PIM. The command interface width in this approach depends on the number of new PIM commands introduced in a given type of DRAM-PIM. For example, McDRAM [39] introduce 10 new commands, Hynix-AiM [37] introduced 16 new commands. This results in a vendor-specific DRAM-PIM interface, which does not align with the commodity DRAMs, where all vendors employ the same interface for a given generation of DRAM. Furthermore, a dedicated memory controller for each vendor has to be designed, which is not the case for commodity DRAM memory controllers. For example, a DDR5 memory controller designed as per JEDEC specification is compatible with DDR5 memory devices from any memory vendor. The lack of standardization for DRAM-PIM is the main cause for the adaptation of different control protocols by each vendor. Hence, I propose a new unified interface and protocol for the external control of DRAM-PIM that addresses the disadvantages of both approaches. The aim here is as follows: 1) Design a generic interface that can accommodate any number of PIM commands without requiring any hardware modifications to the memory controller, and 2) Concurrent operation in both computation mode and normal memory mode for efficient utilization of Dual-Mode feature.

In this new protocol, I propose to introduce a single new command called *PIM Instruction* (PI) that is common to all PIM commands. This new PI command is similar to the conventional write command (WR) and uses address (ADDR), bank (BA), bank group (BG), and data (DQ) bus. The data bus of PI is used to send the PIM opcodes to be executed by the specified bank or bank group of DRAM-PIM. Depending on the type of PIM opcode encoded in the data bus of PI, the memory controller waits for a specified time

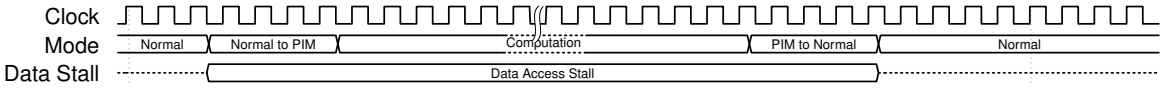

Figure 6.3: Samsung's FIMDRAM operation flow [34]. The data access request is stalled until the PIM mode is changed to normal mode.

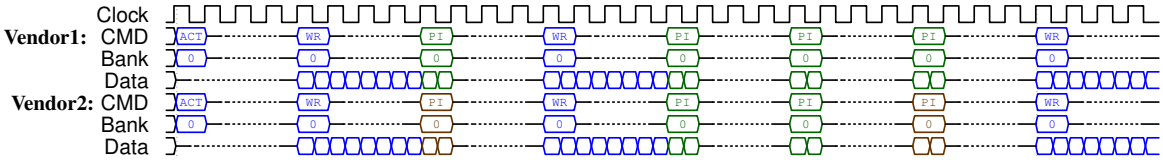

Figure 6.4: Unified Protocol for External Control of PIM. PI is the new generic PIM Instruction command, whose 2 burst data indicate the actual computation commands. In this example, Vendor1 has only one type of PIM command (Green color) and Vendor2 has two types of PIM commands (Brown color). This protocol interleaves the PIM compute operations and data access operations without stalling the data access requests for a long time. Furthermore, Vendor2 sends the new PIM commands to the DRAM-PIM by reusing the PI and does not require any modification to the CMD interface with every new generation.

before processing the next data or compute request. This is similar to commodity DRAM latency between the memory operations. E.g. read and write operation latency, precharge to activate latency, etc. Figure 6.4 shows the example use case of the PI command. I present this example for the following interface configurations: data bus width of 16, PIM internal command width of 32-bit (similar to FIMDRAM), write data burst length of 8, PI burst length of two, a number of PIM internal commands for Vendor1 is one, and a number of PIM internal commands for Vendor2 is two. The proposed approach is generic for any number of vendor commands and command length. The dedicated PI command enabled the differentiation of Non-PIM and PIM requests, which allows the interleaving of these requests. Further, irrespective of the vendor and DRAM-PIM generation any new PIM compute command can be embedded in the data bus of the PI command, which eliminates the need for vendor-specific memory controllers. Thereby, addressing the disadvantages of both state-of-the-art approaches (i.e. Challenge 6). Please note, this example does not include the actual latencies between the commands and also does not include the WL delay between the write command and write data. The purpose of this figure is to demonstrate the basic operation principle of the proposed new protocol.

The proposed new approach requires the memory controller to be modified once to include the new PI command. To understand the feasibility of memory controller modification in this context, I refer to the trends in commodity DRAM protocols and timing changes across different DRAM types (i.e. DDR, HBM, LPDDR, GDDR). Each of these DRAM types has different timings, protocol, command width, interface frequency, etc. The modification of memory controllers is inevitable for each new type of DRAM. However, once

the memory controller design is finished for a given DRAM type as per the JEDEC standard, it is compatible with DRAM devices of any memory vendor for that type. Similarly, the DRAM-PIM architectures adapting the proposed external control protocol require the memory controller to be modified once, after which the memory controller is compatible with DRAM-PIMs from any vendor. For the memory controller to support various PIM command latencies from different vendors, a reconfigurable timing register file is incorporated, similar to conventional memory controllers.

In [87] (Feb 2022) and in [88] (Sept 2022) a research group from Korea University in collaboration with Hynix also proposed a new interface protocol that is capable of servicing both PIM requests and Non-PIM requests without any long stalls. These works were published during the same time period as [65], in which I published my protocol. Both my protocol and their protocol are capable of sufficing inter-vendor compatibility and avoiding memory controller modifications for each new version of DRAM-PIM. There were two prominent approaches to achieve these criteria: 1) a new command PI that is common for all PIM requests, where the PIM opcodes are embedded in the data field (approach adopted in my work), and 2) address-based differentiation that reuses the commodity DRAM interface protocol (approach adapted in [87] and [88]). In the latter approach, each PIM computation operation is assigned dedicated address encoding and it requires address compactors inside the DRAM-PIM die to decode the required PIM or Non-PIM operation. We explored the first approach as it is easy to differentiate the PIM and Non-PIM commands in comparison to address lookup table-based differentiation. Further, it is easily scalable for embedding new PIM opcodes in future generations without requiring the modification of the internal control circuitry of DRAM-PIM architectures. Please note, the inclusion of the PI command to the commodity DRAM interface does not result in any major modification to the existing commodity DRAM interface.

6.3 Summary

In this chapter, I presented new DRAM-PIM architectural approaches that is suitable for training and inference tasks. The QOBL array architecture enabled the integration of floating-point MAC units in the PSA region without encountering the area constraints. The device-level architecture with two types of banks enabled the compensation of QOBL's capacity loss and sufficed the dual-mode operation of DRAM-PIM. The new unified interface and protocol eliminated the vendor-specific memory controller design similar to commodity DRAM counterparts. Overall, these architectural approaches addressed five out of six major DRAM-PIM challenges addressed in Chapter 3. In the future, a detailed behavioral-level and system-level analysis of these novel approaches have to be performed for further quantifying the advantages.

Chapter 7

A Novel DRAM Architecture for Improved Bandwidth Utilization and Latency Reduction using Dual-Page Operation

In this thesis, I aim to address the DRAM memory bound issues highlighted in Chapter 1, not only by proposing novel PIM architectures (Chapter 4 to 6), but also by proposing novel DRAM architectural solutions for conventional non-PIM accelerator platforms. In this chapter, I will propose a new reduced latency DRAM architecture to address the memory-bound issues in conventional non-PIM accelerator platforms. As mentioned in Chapter 1, many state-of-the-art computing systems and accelerators encounter performance saturation, mainly due to limited DRAM bandwidth and long latencies. The peak bandwidth of a DRAM device is defined as Data Bus Interface Width $\times$ Interface Frequency $\times$ Dual Data Rate. DRAM vendors are constantly thriving to increase the interface frequency and/or the interface width in each generation of DRAM to achieve a higher peak bandwidth. However, the effective utilization of the available peak bandwidth strongly depends on the DRAM devices' internal latencies. To maximize the DRAM bandwidth utilization, the internal data access latencies have to be low. Depending on the internal state of a DRAM bank, the data access latency varies largely. For example, a DDR4 device operated at 800 MHz [52] has a data access latency varying between 5 ns to 56.25 ns. This large variation limits the maximum utilization of the peak bandwidth. One of the high latency consuming operations within DRAM bank is the Page-Miss. It happens when a currently open page/row within a bank has to be closed and a new page/row has to be opened before accessing the data from the new page. The page-miss latency in a DDR4 device operated at 800 MHz vary from 32.5 ns (i.e. page-miss RD to RD latency = $t_{RTP} + t_{RP} + t_{RCD}$) to 56.25 ns (i.e. page-miss WR to WR latency = $t_{WL} + BL/2 + t_{WR} + t_{RP} + t_{RCD}$). Avoiding the page-misses or masking the page-miss latency can increase bandwidth utilization upto 95% of the peak

bandwidth [81]. The impact of page-miss latency is addressed at several levels in state-of-the-art DRAM memory sub-systems. At the DRAM device level, the number of banks is increased to achieve higher bank parallelism, which masks the page-miss latency, i.e. while bank A is stalled due to a page-miss, data can be accessed from bank B which already has an activated/open row. At the memory controller level, address mapping techniques [144], adaptive page-policy [101], and schedulers [145] are employed to exploit bank parallelism and reduce page-misses. However, the effectiveness of these approaches is strongly influenced by the application knowledge and its memory access pattern. Hence, the impact of the page-miss latency is persistent when the DRAM accesses are not adequately scheduled and channelized to the same bank more frequently in a given period of time.

A new research direction in this context is to explore a new level of parallelism within a bank. The aim is to design a DRAM bank that is capable of activating multiple rows within the same bank. One of the first research publications in this direction is Sub-Array-Level Parallelism (SALP) [90]. The authors of SALP proposed an architecture variant called MASA that exploits the Sub-Array parallelism to activate multiple rows/pages within a bank, i.e. one page per H-Block (i.e. Horizontal-Block or row of Sub-Arrays, refer to Figure 7.1). One of the key drawbacks of SALP architecture is the lack of flexibility to activate a second page within a currently active H-block, which is referred to as page restriction henceforth. This page restriction is also applicable to the neighboring H-Blocks of a currently active H-Block due to PSA sharing (i.e. page-miss is persistent). As a result, this approach cannot flexibly activate any two pages within a bank. Furthermore, the memory controller for SALP has to be aware of the row address boundaries of the H-Blocks, which depends on the number of rows per Sub-Array. The number of rows per Sub-Array or the H-Block address boundary is not available from the DRAM specification and is a vendor-specific confidential design-dependent value, which varies from one generation of DRAM to other. Additionally, DRAM banks also internally scramble a few pages or a set of rows to fix manufacturing defects during die testing. This scrambling is die-specific and the memory controller has to be aware of the address boundaries of these scrambled rows. The prior works [91–93] presented different variants of SALP architectures that partition the DRAM bank into multiple sub-entities and activate one partial page (i.e. half-page or quater-page) per sub-entity. Similar to SALP, these architectures also lack flexibility and encounter page-misses within the sub-entity.

In this chapter, I propose a novel DRAM Sub-Array level architecture referred to as *Dual-Sense-Amplifier* (DSA) that is capable of flexibly opening any two rows per Sub-Array or any two pages per bank without page-restrictions. The proposed architecture is compatible with any type of commodity DRAM architecture (e.g. HBM, GDDR, etc.) and it retains most of the commodity DRAM circuits and results in minimal DRAM modifications. DSA explicitly ensures that the Sub-Array design that is highly optimized for area and density is not modified. Additionally, the memory controller does not need to be aware of the internal design-specific Sub-Array dimensions or row address boundaries for the activation of the second row. The bandwidth of a DSA-based 8 Gb DRAM device increases by

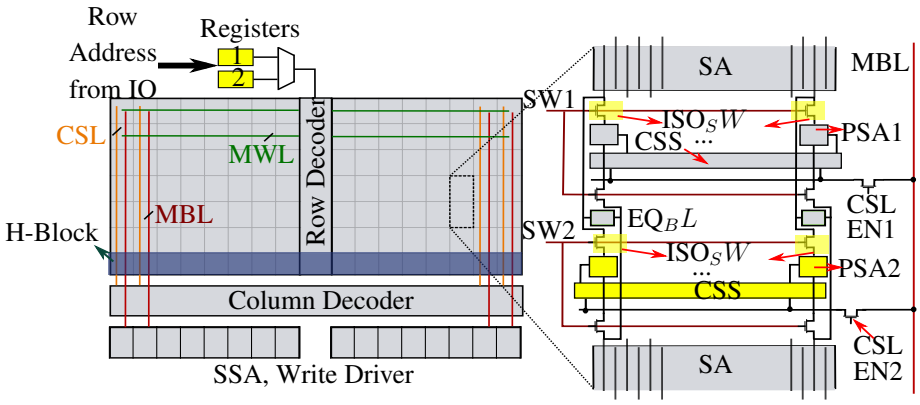

Figure 7.1: Dual-Sense-Amplifier (DSA) Architecture. In this figure Sub-Array is abbreviated as SA.

17.53 % and the average response latency reduces by 31.59 ns (≈25 clk) compared to commodity 8 Gb DRAM with row-bank-column-bankgroup address mapping. The proposed DSA architecture has a low area overhead of 9.6 % for a 2y nm technology 8 Gb device, and a negligible energy consumption overhead compared to the commodity 8 Gb DRAM. A further advantage of this architecture is the ability to access the data during a refresh phase of the DRAM without having to stall the data bus until its completion. Overall, the main aim of DSA architecture is to investigate a new level of parallelism that can be exploited within the DRAM banks for increased bandwidth utilization without encountering page-restriction issues. A US patent is filed for this DSA architecture [71]. The contributions of this chapter in prior published in [67]. This publication was one of the selected papers of IEEE International Symposium on Circuits and Systems (ISCAS) 2021, and all the selected papers were published in a special issue of the IEEE Transactions on Circuits and Systems—Part II: Express Briefs (TCAS-II) [94].

7.1 Architecture

This section describes the detailed architecture of the DSA. Figure 7.1 shows the DSA bank architecture, which is capable of activating two rows within a Sub-Array. The yellow blocks in the figure depict the new additional blocks to the existing PSA-region. The foremost requirement for activating two rows in a Sub-Array is the availability of a second set of PSAs (i.e. PSA2) and its associated control circuitry. The availability of two sets of PSAs per Sub-Array enables the activation of two individual rows within a Sub-Array i.e. opening of row1 by PSA1 and row2 by PSA2. However, the LBLs and the reference LBLs (LBLs*) that connect to the PSAs have to be shared by both sets of PSAs using the isolation switches (ISO_SW), as shown in Figure 7.1 i.e. LBLs are connected to PSA1 while activating row1 and LBLs are connected to PSA2 while activating row2. Note that the isolation switches

in the path of LBLs and PSAs will not impact the cell charge sensing process by the PSAs, since similar switches are proposed in a few prior DRAM architectures [146–148]. The signals SW1 and SW2 to the ISO_SW are driving higher voltage (i.e. $> VDD$) to avoid level degradation. LBLs and LBLs* have to be precharged prior to the activation of a second row within a Sub-Array. This precharge operation reuses the existing equalizer circuit, referred to as EQ_BL. The time required to precharge the LBLs and LBLs* is referred to as t_{RP_BL} and is less than the conventional precharge duration t_{RP}, which includes the precharge duration of PSAs as well.

The operation flow of the DSA architecture is as follows. Initially, the LBLs and both sets of PSAs are in a precharged state. The activation of the first row (e.g. R1) is similar to activation in commodity DRAMs, i.e enable the PSAs, enable LWL, charge sharing between cell capacitor and LBLs, and charge sensing by PSA1 (refer Chapter 2 for DRAM background). During the activation of row R1, the LBLs and LBLs* are connected to PSA1. After the time t_{RCD} (i.e. activate to read/write timing), the LBLs and LBLs* are disconnected from the PSA1 and are precharged by EQ_BL to enable the activation of a second row (i.e. R2) within a Sub-Array. This process disrupts the write-back process from PSAs to the respective cells of row R1, but the data continue to be buffered in PSA1. During the precharge of LBLs, the LWL corresponding to R1 is also disabled. After t_{RP_BL} duration, the LBL are connected to PSA2. The memory controller sends a second activate command to activate row R2 belonging to the same Sub-Array. The LWL of corresponding to R2 is enabled and the data of row R2 is sensed by the PSA2. Similar operation steps are followed for the activation of a second row in a vertically neighbouring Sub-Array. This eliminates the page restriction that was encountered in SALP Architecture. The memory controller can read/write data to/from either of the PSAs by selecting the appropriate control signal CSEL_EN1 and CSEL_EN2. For this, an additional column address bit is required in DSA architecture to differentiate the read and write data between the two rows, which can be encoded within the available address I/Os of commodity DRAMs.

In the commodity DRAMs, the LBLs are always connected to DRAM cell and PSAs, which simultaneously updates both PSA values and cell values during the write operation. In the case of DSA, the LBLs are disconnected from PSAs. This updates only the PSAs' value but not the corresponding cell values of the desired row. Hence, DSA architecture introduces a new command called WRB (i.e. Write Back command), which the memory controller has to send to the DRAM in order to write back the updated data in PSAs to the corresponding cells. The WRB command will enable the LWLs of the appropriate row, connect the LBLs to the appropriate PSA set (i.e. PSA1 or PSA2), and initiates the write-back from the PSA to the cell capacitor. This enabling of LWLs and connection of LBLs will not affect the data buffered in PSA, since the PSA value is in a stable state after activation. Only a minor voltage change in the PSAs is observed and the PSAs quickly recover to the original value. The row addresses required to enable the appropriate MWL and LWLs for write-back are buffered in the internal registers in each bank (Refer Figure 7.1). The memory controller only has to send a single bit register address, i.e. Reg1 or Reg2,

along with the WRB command instead of a large row address. This approach is especially advantageous for DRAMs with a multi-cycle command protocol (e.g. LPDDR4) because the WRB command requires only two cycles instead of four, i.e. two additional cycles for sending the row address are saved. The activation of a second row belonging to a different H-Block that does not share any PSAs with the already activated H-Block also follows the same operational steps, i.e. LBL disconnection after t_{RCD}, disabling of R1 LWL/MWL, etc. Although these H-Blocks are capable of being operated concurrently, this architecture does not exploit this feature to eliminate the need for the memory controller to be aware of the vendor-specific number of rows per Sub-Array and H-Block row address boundaries. This maintains uniform timing and control flow in the memory controller for the activation of the second row within a bank irrespective of the row location.

The activation of the third row in DSA requires the memory controller to send the PRE command that precharges the PSAs and LBLs. However, the WRB command has to be sent by the memory controller prior to the PRE command. An intelligent scheduling of WRB commands can potentially mask the latency induced by them, e.g. WRB commands can be sent in the next clock cycle of WR commands. Table 7.1 shows all the new timing constraints of the DSA architecture in addition to the timings that are mentioned in the DRAM device specification. For instance, the time between two consecutive ACT commands within a bank is t_{RCD} + t_{RP_BL} + $3 \cdot t_{CK}$, where the additional 3 clock cycles are for the sequential enabling of the desired control signals. Figure 7.2 shows the example timing diagram of DSA that increases the bandwidth utilization and reduces the data access latency in comparison to the commodity DRAM. During the refresh phase, only one page has to closed/precharged while retaining the second page in an active state. This enables data access from the second active page during the refresh process, but new pages cannot be opened. Overall, DSA architecture eliminates the page restriction problem of state-of-the-art DRAM architectures that enable the activation of multiple pages within a bank.

Table 7.1: Additional Timing Constraints for DSA. All new timings are applicable for the commands sent to the same bank

Command Sequence	**Timing Constraint**
WRB $\rightarrow$ PRE	t_{WRB} + $3 \cdot t_{CK}$
WRB $\rightarrow$ ACT	t_{WRB} + t_{tRP_BL} + $3 \cdot t_{CK}$
ACT $\rightarrow$ ACT	t_{RCD} + t_{RP_BL} + $3 \cdot t_{CK}$
ACT $\rightarrow$ WRB	t_{RCD} + $3 \cdot t_{CK}$
WRB $\rightarrow$ WRB	t_{WRB} + $3 \cdot t_{CK}$

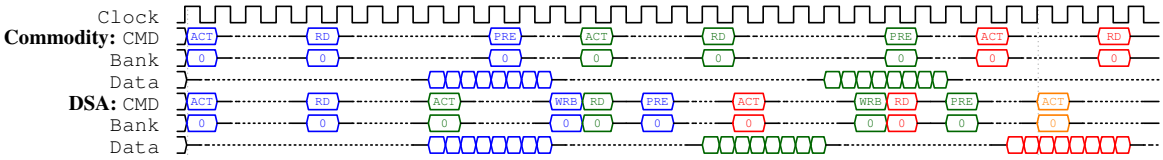

Figure 7.2: DSA latency reduction as compared to commodity DRAM.

7.2 Experimental Setup

The proposed DSA architecture is constructed on top of a state-of-the-art 8 Gb commodity DRAM architecture with 8 banks, 64 H-Blocks per bank, 16 sub-arrays per H-Block, a 1024×1024 standard sub-array dimension, and a 2 KB page. The area of the commodity DRAM Sub-Array is calculated using the reverse engineered unit cell dimensions, LBL pitch, and LWL pitch (e.g. 66 nm) of 2y nm technology (i.e. between 25 nm to 27 nm) based 8Gb DRAM dies [117], [118]. Please note, DRAM technology is typically represented like 2y, 1y (≈17 nm), 1z (≈14 nm), etc. The area of the other standard DRAM circuits (e.g., decoders, sense amplifiers, etc.) is evaluated using the DRAM architecture exploration tool DRAMSpec [119]. The required 2y nm DRAM technology input parameters for DRAMSpec are extracted from multiple sources and articles [48,117,118,120–124]. To provide an indication of the correctness of the area evaluation methodology, I compare the estimated 8 Gb standard DRAM die area calculated with this methodology (i.e., 74.44 mm^2) to the area of 8 Gb commodity DRAM dies from memory vendors i.e. Samsung's DDR4 (i.e., 59.00 mm^2) [118], Hynix's DDR4 Gen-1 (i.e., 76.0 mm^2) [117] and Micron's DDR4 (i.e., 58.48 mm^2) [125].

The latency of the DRAM bank circuits are estimated using an in-house enhanced circuit-level DRAM model (SPICE) that can be equipped with different kinds of SPICE transistor models and PDKs, such as FreePDK [126] with PTMs [127], Cadence PDK models, and as well proprietary DRAM transistor models. This DRAM model includes the wire parasitics extracted from Qimonda's 45nm DDR3 design. The circuit-level results and timing correctness of this model are presented in [128] using BSIM4 model cards built on Low Power PTM transistor models that are adapted to DRAM technology. We updated this DRAM model to 2ynm DRAM technology based on the scaling results presented in [129].

To demonstrate the practical applicability, we conducted several experiments with the bandwidth-demanding applications presented by the authors of [144], i.e. Image Rotation, Convolutional Neural Network (CNN) and 3D Image Rotation. The bandwidth (BW) and average response latency (ARL) results for comparison of the DSA 8 Gb device, the commodity DDR4 8 Gb device, and the SALP 8 Gb device are computed using the open-source cycle-accurate SystemC/TLM-based DRAM simulator called DRAMSys [149, 150]. The energy/power estimations of the aforementioned device architectures are computed using the open-source tool called DRAMPower [151] that is coupled to DRAMSys and fed with

the issued DRAM commands. The power models in DRAMPower are calibrated with the measured currents, such as I_{DD0} (i.e. single activate to precharge current), I_{DD2N} (i.e. precharge standby current - all bank), I_{DD3N} (i.e. active standby current - all bank), I_{DD4R}/I_{DD4W} (i.e. burst read/write current), I_{DD5}/I_{DD6} (i.e. refresh/self-refresh current), which are provided by the vendor specifications. The average response latency, bandwidth evaluations, and power consumption evaluation of DSA are for a DDR4-like device with an interface clock frequency of 800 MHz, dual data rate, data bus width 16 ($\times$16), 2KB page size, and a burst length of eight. The current values provided by a DDR4 specifications [152] that match the aforementioned device configuration are used as the base for obtaining the DSA current values and for its power evaluation.

The MASA architecture variant of SALP [90] with a configuration of 8-subarrays-per-bank is used for the comparisons (referred to as SALP-8). According to the authors of [90], 8 subarrays-per-bank means the DRAM bank is partitioned into 8 sub-banks that are capable of opening one page per sub-bank. The memory controller of all the architectures (i.e. DSA, SALP, and Commodity) is configured with a standard Row-Bank-Column-BankGroup (RBCBg) address mapping, open page-policy, and First Ready - First Come First Serve (FR-FCFS) scheduling scheme. Additionally, the memory controller of SALP is configured with the row mirror address mapping on top of the RBCBg mapping to uniformly distribute the incoming traffic to all sub-banks, which reduces the traffic congestion for a single sub-bank or H-Block and the impact of page restrictions.

7.3 Results

This section presents the evaluation results of the proposed DSA architecture. The results are presented for an 8 Gb device.

7.3.1 Area Evaluation

Based on the methodology described in Section 7.2, the area evaluations of all the relevant circuits of the presented DSA architecture are shown in Table 7.2 for a 2y nm DRAM technology. The DSA has a low area overhead of 9.6% compared to a commodity DRAM device with a similar configuration, i.e. 8 Gb, 8 banks, 2 KB page, etc. The height of the PSA region of commodity DRAM is estimated to be 10 μm, which includes PSAs, EQ_BL and column select switches. I conservatively estimate the additional height required for the second set of PSAs to be $1.1\times$ the commodity DRAM PSA region height, considering the routing complexity and additional isolation switches. This equates the total PSA-region height to 21 μm.

Table 7.2: Area evaluation results

Block	**Area** $Height \times Width$ **[**μm^2**]**
Commodity 8 Gb DRAM	
Sub-array (SA)[a]	83.68×74.036
PSAs ($512\times$), per side of Sub-Array	10×74.036
Sub-array with PSAs[2]	93.68×74.036
Column decoder, SSA, Drivers	125×1444.58
Row decoder, Drivers	6329.2×250
Bank [a]	6329.2×1444.58
Die	**74.4442 (mm^2)**
DSA 8 Gb DRAM	
Sub-array with PSAs[b]	104.68×74.036
Bank	7035.2×1444.58
Die	**81.6283 (mm^2)**

[a] Including redundancy
[b] SA area considers only single side of PSA due to PSA sharing

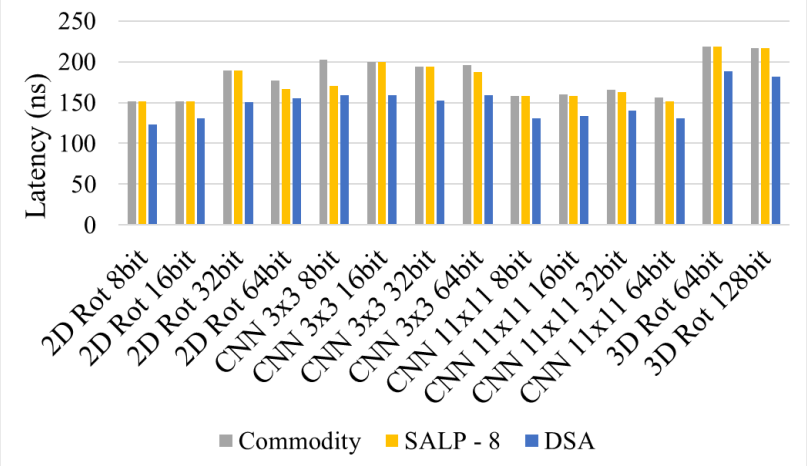

Figure 7.3: Average Access Response Latency comparison of DSA, SALP and commodity 8Gb devices for quad-core system.

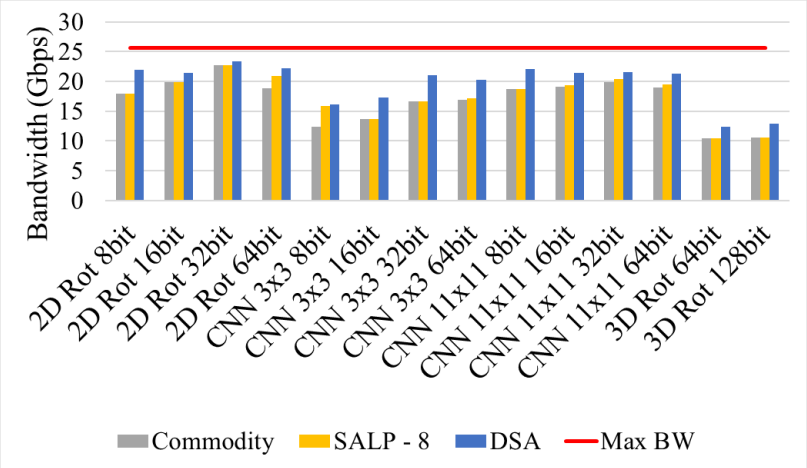

Figure 7.4: Bandwidth Utilization comparison of DSA, SALP and commodity 8Gb devices for quad-core system.

7.3.2 DSA Timing Evaluation

SPICE evaluations show the timing related to activation of the second row within a bank is the same as commodity DRAM (i.e. t_{RCD}). The new DSA-specific operations, WRB (t_{RP_WRB}) and LBL precharge (i.e. t_{RP_BL}) is 7.07 ns and 5.16 ns, respectively. The precharge of the whole activated row, i.e. both PSAs and LBLs, is same as commodity DRAM (i.e. t_{RCD}). The connection and disconnection of LBLs from the PSAs showed minor changes on the sensed voltage value in PSAs, but the sense amplifier recovered and stabilized to the original sensed value in less than 3 ns, worst case. Additionally, the evaluations show the marginal impact on ACT timing t_{RCD} due to additional LBL loads (i.e. ISO_SW). All the other commodity DRAM timings required for the further evaluations, which are also applicable for DSA and SALP (e.g. t_{RRD}, t_{CCD}, Write Latency, Read Latency, t_{RAS}, etc.), are extracted from a DDR4 specifications [152]. Refer to Chapter 2 for all the timing definitions.

7.3.3 Average Response Latency and Bandwidth Evaluation

In this section, I compare the average response latency (ARL) and bandwidth (BW) results of DSA, SALP-8, and commodity DRAM 8 Gb device for various applications (refer to Figure 7.3 and Figure 7.4). The presented system-level analysis is for a quad-core system, where all cores are executing the same application and each core is allocated a memory space required by the application, similar to an accelerator environment. A standard RBCBg address mapping is used to maximize the bank parallelism, which is especially beneficial for commodity memory sub-systems. On average across all test applications, the ARL of the DSA memory sub-system (i.e. DSA DRAM + controller) is 31.59 ns ($\approx 25\,clk$) and 27.28 ns ($\approx 22\,clk$) lower than commodity DRAM and SALP, respectively. As a result,

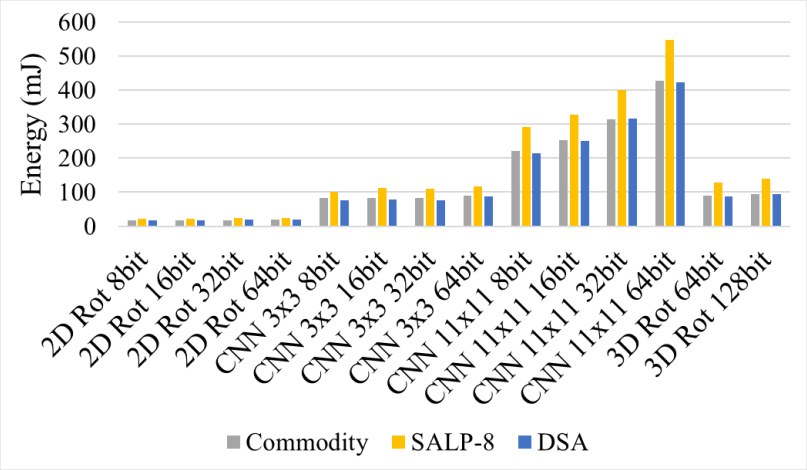

Figure 7.5: Energy consumption comparison of DSA, SALP and commodity 8Gb devices for quad-core system.

the bandwidth achieved with the DSA device is on average 2.76 Gbps (i.e. 17.53%) and 2.25 Gbps (i.e. 14%) higher compared to the commodity DRAM and SALP, respectively. A maximum ARL reduction of 41 ns and a maximum bandwidth improvement of 26% in comparison to commodity DRAM is observed for the 32 bit quantized CNN application with a 3×3 kernel. In summary, the bandwidth utilization in DSA is higher compared to SALP for the aforementioned applications.

For many applications, SALP achieved small improvements over the commodity DRAM. The reason for this is the page-restrictions or page-misses within the sub-bank. At a sub-bank level, the operation of SALP is similar to a commodity DRAM bank operation. The efficient utilization of the available sub-bank parallelism using memory controller optimizations like an application-specific address mapping (ASAM) can further improve the BW utilization of SALP. However, this aforementioned optimization approach (i.e. ASAM) results in BW utilization improvement for DSA as well and is not the scope of this thesis. The main aim of this architecture is to demonstrate the new level of parallelism within bank and Sub-Array which can be exploited by the DRAM memory controllers for increased bandwidth utilization without encountering page-restriction issues.

7.3.4 Energy and Power Evaluation

DSA circuit analysis shows that the activation of more than one page in each bank in the DSA and SALP architectures will majorly increase the standby current called I_{DD3N}, while the other currents are similar to a commodity DRAM device[1]. The increase of I_{DD3N} in DSA and SALP-8 devices is directly proportional to the number of activated pages per bank (i.e. NumPagePerBank). The new I_{DD3N} is shown in Equation 7.1, where $I_{DD2N}^{Commodity}$ depict

[1] Assuming same device configuration and the same design of standard DRAM circuits

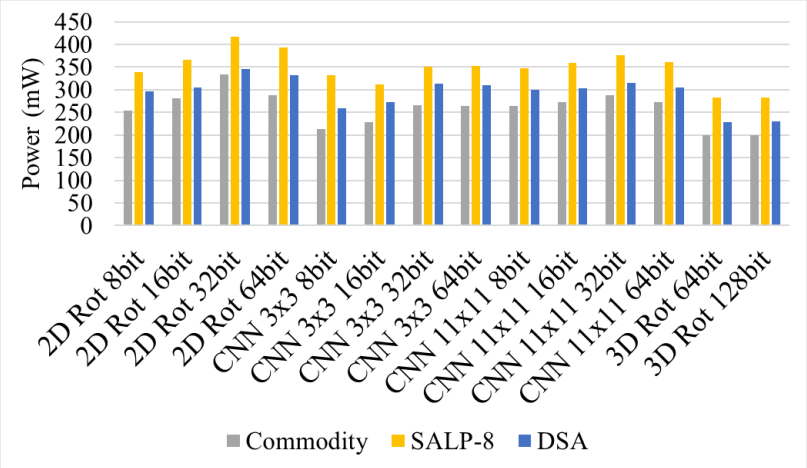

Figure 7.6: Power consumption comparison of DSA, SALP and commodity 8Gb devices for quad-core system.

the standby current in commodity DRAM when all banks are precharged and $I_{DD3N}^{Commodity}$ depict the standby current in commodity DRAM when all bank has one active page per bank.

$$I_{DD3N}^{New} = I_{DD2N}^{Commodity} + ((I_{DD3N}^{Commodity} - I_{DD2N}^{Commodity}) \times NumPagePerBank) \quad (7.1)$$

Figure 7.5 and Figure 7.6 compares the energy and power consumption of DSA, SALP-8, and commodity DRAM 8 Gb device for various applications. On average DSA consumes 14.59% higher power while increasing the bandwidth by 17.53% compared to the commodity DRAM. As a result, the DSA device consumes approximately the same energy as the commodity DRAM while it improves bandwidth utilization. The reduced page misses and page restrictions within a Sub-Array or bank compensate for the increase of standby current I_{DD3N}^{New}.

7.4 Summary

This chapter presented a novel DRAM Sub-Array level architecture called *Dual-Sense-Amplifier* (DSA) that flexibly activates two rows within a Sub-Array or a bank without any page restriction. The architecture reused most of the commodity DRAM circuit designs, incurred minimal modifications, and is fully compatible with any type of commodity DRAM, i.e. HBM, LPDDR, etc. Additionally, the memory controller does not need to be aware of the vendor-specific Sub-Array dimensions for the activation of the second row, which is not the case for SALP architecture. The area overhead of DSA for 2y nm technology is 9.6% compared to the commodity 8 Gb device. The DSA architecture reduced the average response latency by 31.59 ns, and the resulting average bandwidth improvement is 17.53% higher compared to commodity counterparts operated at a frequency of 800 MHz.

Moreover, the energy consumption of the DSA architecture is approximately the same as commodity DRAM.

Chapter 8

DRAM Memory Controller Design for Transprecision Applications

In this chapter, I explore the architectural solutions to address the DRAM memory-wall issues (i.e. high energy consumption and long data-access latency) at a memory controller level rather than modifying the DRAM architecture (i.e. like in Chapter 4 to Chapter 7). A memory controller is a specialized block between the processor core and external DRAM devices (Figure 8.1) that suffice the complex protocol and various timings specified in the JEDEC standard. I present the architecture and hardware implementation of the standard DDR3 DRAM memory controller, which is further extended with additional modules that reduce the DRAM energy and overall latency.

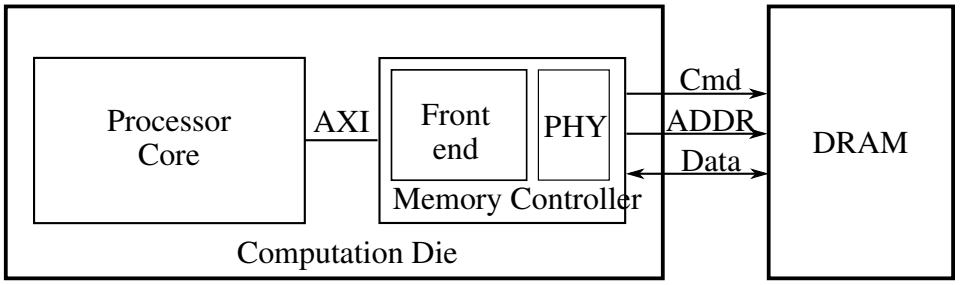

Figure 8.1: Computation system top-level overview.

8.1 DDR3 DRAM Memory Controller for Transprecision Computing

Approximate computing is one of the effective techniques to overcome the energy scaling barrier of computing systems by compromising the accuracy of results. Inspired by

approximate computing, *transprecision computing* [153] has emerged as a new computing paradigm that offers a dynamic precision reduction to achieve higher energy efficiency without inheriting any errors in the final output. In other words, the accuracy of the final result is the same as that of traditional full-precision computing. The dynamic precision reduction is achieved by spanning all the layers of the computing system i.e. from the algorithm to the hardware that supports a variety of precision settings. The objective of H2020 European project OPRECOMP is to implement transprecision computing platforms for applications ranging from *Internet-of-Things* (mW Platforms - ASIC) to *High Performance Computing* (KW platforms - FPGA). OPRECOMP project uses an open-source RISC-V based Parallel Ultra-Low-Power Processing Platform (PULP) [154]. The high degree of customization offered in RISC-V architecture enables the design of energy-efficient transprecision computation units. OPRECOMP project also aims to adapt the memory subsystem for transprecision computing to further optimize the energy consumption of external memories like DRAMs. As a part of OPRECOMP project, the Microelectronic Systems Design Research Group, Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau contributed towards the design of DRAM memory controller adapted for transprecision computing. My contribution is the identification of the new additional modules for the memory controller and the design/implementation of the Memory controller's entire Frontend block. The analog blocks of the memory controller are designed by my colleague M.Sc. Jan Lappas. Please note that the lack of open-source ASIC implementation of the memory controller required us to implement our memory controller from scratch. This memory controller is designed for mW platform with a single DDR3 DRAM device and 8-bit data bus, but is scalable to other platforms with a wider DRAM bus. The designed DDR3 memory controller is integrated with the PULP cluster to demonstrate the advantage of transprecision computing for IoT/embedded applications i.e. mW platform, Urania Chip [155].

8.1.1 Memory Controller Features for Transprecision Computing

As shown in Chapter 1, DRAMs consume a major portion of the overall system energy in many memory-intensive applications like neural networks. This is partially due to the DRAM operations such as refresh, activation, and precharge operations that contribute high energy [156]. For example, the refresh operation in a single 4Gb DDR3 DRAM chip consume up to 15% of the total DRAM energy [157]. The impact of the refresh further increases in the next generation of high-density DRAM devices (i.e. 64Gb devices) consuming up to half of total DRAM energy [157, 158]. In this context, one of the fundamental features is to adapt the idea of approximate computing to the DRAM (i.e *Approximate DRAM*) [159–161]. The key knobs of approximating the DRAM is to vary the refresh rate in order to enable the trade-off between energy efficiency, performance, and reliability. This technique allows the processing units to store the application data in an appropriate refresh/reliability zone that ranges from no refresh to a high refresh rate. For example,

the data are stored in no refresh zone if the application updates the data at a rate faster than the required refresh period of the DRAM [162]. Similarly, the MSB bits of a data block can be stored in a high refresh zone while storing the LSB bits in a no-refresh or moderate refresh zone depending on the application error resilience. DRAMs are typically refreshed using the Auto-Refresh command, where the user has no control over the rows that are internally refreshed. An alternative approach is to use Activate and Precharge commands to mimic the refresh operation while providing the flexibility of refreshing the individual rows of DRAM (i.e. fine granular refresh). A typical approximate DRAM employs this fine granular refreshing technique in order to refresh different zones of reliability at varied rates. This approach is previously proposed in [158, 163, 164]. One of the recent approaches to realize a fine granular refresh is using *Optimized Row Granular Refresh (ORGR)* methodology that is proposed by my colleague Dr.-Ing. Deepak M. Mathew in his PhD thesis [165, 166]. They present a reverse engineering method to determine the user-unknown minimum DRAM timings to realize the fine granular refresh that consumes the lowest latency in comparison to Auto-Refresh and other fine granular refresh techniques that consider JEDEC specified standard Activate to Precharge timing. The refresh latency of ORGR is 45% lower than Auto-Refresh while proving the added benefit of flexibly refreshing any row. Some of the recent DRAM types like LPDDR4 or GDDR6 have a new feature called Per-Bank-Refresh that is capable of refreshing a single bank, but similar to Auto-Refresh the Per-Bank-Refresh also lacks the flexibility to refresh the individual rows within a bank. Please note, there are several research works that have been published to reduce the overhead of refresh [167–170] at several levels of computation system. My aim in this thesis was to incorporate and implement ORGR in the DRAM memory controller for transprecision computing within the scope of OPRECOMP project.

The second important feature is to exploit the application knowledge. General-purpose memory controllers are confined to online scheduling techniques that only have a local view of the executed application. However, many applications have deterministic memory access patterns (e.g. neural networks), which can be exploited to improve bandwidth and energy. My colleague Prof. Dr.-Ing. Matthias Jung presented a novel methodology to generate an *Application-Specific Address Mapping* (ASAM) in his PhD thesis [81, 144, 171]. This approach has a global view of the application and exploits the application knowledge to optimally map the data to DRAM locations that decrease the total number of row misses, i.e. reduced number of precharge and activate operations. The authors of [144] showed up to 9x and 8.6X improvements in bandwidth utilization and energy efficiency by employing this technique.

The aforementioned research works presented the proof of concepts mainly by simulations. To the best of our knowledge, this is the first memory controller hardware implementation to combine all the previously discussed advanced techniques. With these features inside the memory controller, the energy efficiency of a transprecision system is further optimized as the memory subsystem is tailored to transprecision computing. My key contributions in this thesis are the identification of the new additional modules for the

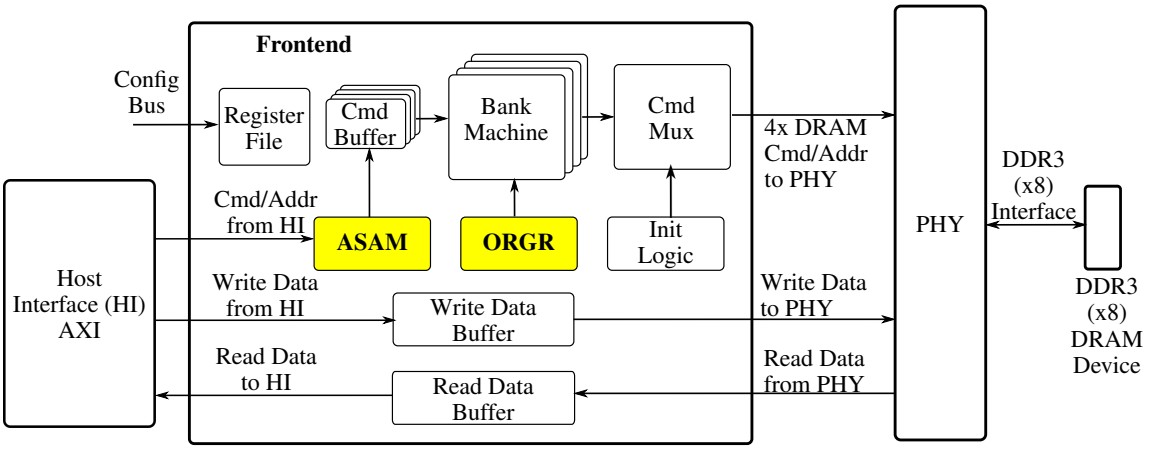

Figure 8.2: Transprecision DDR3 memory controller architecture

memory controller and the design/implementation of the memory controller's entire frontend block with these additional modules. Please note that the lack of open-source ASIC implementation of the memory controller required us to implement our memory controller from scratch. The key features of this memory controller's frontend block are as follows:

- Lean, low latency, and low power.
- Enables fine granular refresh control using ORGR.
- Supports the exploitation of application knowledge using ASAM.

8.1.2 Memory Controller Architecture

Figure 8.2 shows the architecture of the transprecision memory controller. I only present the architecture of the frontend, which is my contribution to the memory controller design. It is designed to satisfy the mW platform requirements i.e. low power, low area and low latency. The frequency ratio between the PHY and the frontend is 4:1, similar to state-of-the-art memory controllers, such as [172, 173]. This allows the frontend to be operated at a lower clock frequency (i.e. same frequency as the processor core) to meet the lower power requirements of the mW platform. In order to compensate for the frequency difference and avoid stalling of the PHY, the frontend issues $4\times$ DRAM commands to the PHY per slow clock cycle (i.e. commands corresponding to next 4 fast cycles).

The ASAM block is a dedicated address decoder that translates the incoming address bits from the *Host Interface* (HI) like AXI interface to an equivalent DRAM row, column, and bank addresses as per the configured custom address map. The ASAM block incorporates a configurable address scrambling hardware as shown in Figure 8.3. This custom

address map is identified offline based on the mathematical models and application knowledge, which is configured to a register file of the memory controller during the initialization phase. The identification of the custom address map is described in [144] and is not the scope of this thesis. The incoming logical address from the HI is typically 32 bit, out of which only 30 bits are valid since the maximum density of a DDR3 device is 8Gb. Note that the addresses from HI are byte addressable and the requests are in the granularity of a cache line (i.e. 8 bytes). The lower 3 bits of HI address are directly mapped to the lower 3 bits of the DRAM column address (C2-C0). The remaining 27 bits of the HI address are scrambled by the $27\times$ 27 bit multiplexers to determine the DRAM addresses i.e. bank (B2-B0), row (R15-R0), and column (C10-C3).

The translated addresses and the corresponding HI command (i.e. read or write) are forwarded to one of the eight command buffers (FIFO) depending on the bank address. The *Bank Machines* (BM) consecutively processes the incoming traffic associated with its respective DRAM bank. A BM keeps track of the currently active row of its respective bank and translates the incoming transactions to a sequence of DRAM commands as per the JEDEC protocol. The command sequence depends on the current state of the bank and the target row of the incoming transaction. For example, the read request is translated to Precharge, Activate, and Read commands if the currently active row in the DRAM is different from the row corresponding to the read requested The BM also guarantees all bank-specific timings such as tRCD, tRTP, tWR, tRAS, and tRP (refer Chapter 2 for timing definitions). The *Command Multiplexer* (Cmd Mux) prioritizes the DRAM commands from multiple BMs, ensures that the bus-related timings (i.e. inter-bank timing) like tWTR, tRTW, tRRD etc. are maintained, and packs $4\times$ DRAM commands. The *Init* block handles the initialization sequence of the DRAM as specified in the DDR3 specification [51]. Until the initialization is finished, the rest of the memory controller is stalled. The write data from the HI is stored in the write buffer and is forwarded to the PHY along with its corresponding write command. The read data that arrives after RL clock cycles (i.e. read latency) is stored temporarily in read buffer and forwarded to the host processor unit via HI. The data bus width of the frontend is 64 bits i.e. DRAM Burst Length $\times$ DQ width (8×8). The configuration of the DRAM timings, mode register settings, and other internal parameters required by the frontend and the PHY is done via the 8 bit *Configuration Bus* (Config Bus) that employs a custom protocol similar to *Serial Peripheral Interface* (SPI).

The ORGR block manages the DRAM refresh operation using the optimized fine granular refresh technique. The ORGR block consists a set of counters to track the refresh interval of different DRAM zones of reliability. As the counter expires, the ORGR sends the corresponding BM the row addresses to be refreshed. That appropriate BM will stall its further transactions to its respective bank and services the ORGR request with the sequence of ACT and PRE commands (i.e. fine granular refresh) with reduced DRAM timings. These ACT and PRE commands are given the highest priority by the Cmd Mux. The reverse engineering methodology presented in [166] to identify the user unknown minimum timings is executed during the initialization. However, it is not triggered at every initialization due

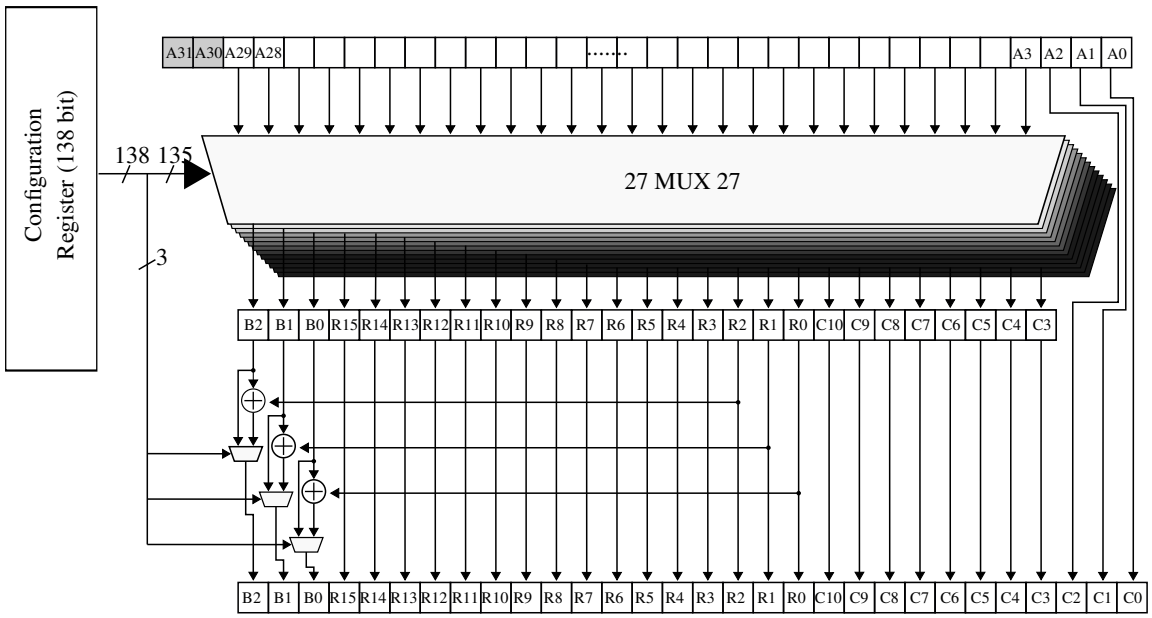

Figure 8.3: ASAM Architecture

to the fact that the identified minimum timings in most cases remain unmodified for the entire course of operation of that device. Hence, it is triggered once, and the last known minimum timing values are configured via the config bus from the software during further initialization phases. The config bus is also used to define the DRAM reliability zones and their respective refresh intervals.

8.1.3 Results

The memory controller is implemented using the UMC 65 nm Low-Leakage CMOS bulk technology. The synthesis, place and route, and timing/power analysis of the digital logic are carried out using Synopsis' DesignCompiler, IC Compiler, and PrimeTime, respectively. The circuit level simulations and the full custom layout of the analog blocks of the PHY are done using Cadence Spectre and Virtuoso. The DRAM model provided by multiple vendors and the bit true model of our PHY are used for the functional verification (post-layout) of the DDR3 controller and the PHY. Figure 8.4 shows the floor plan and layout of our memory controller designed for mW platform. The pin pitch of the data IOs is 200 μm and the ADDR/CMD IOs have a pin pitch of 100 μm. All the results associated with the AXI4 and Frontend are obtained by me and the results associated with the PHY block are from Jan Lappas I present the PHY results in addition to the frontend block to provide the reference of comparison.

The area distribution of the memory controller consisting of frontend, PHY digital logic, and PHY IO transceiver is shown in Table 8.1. The core of the memory controller (i.e.

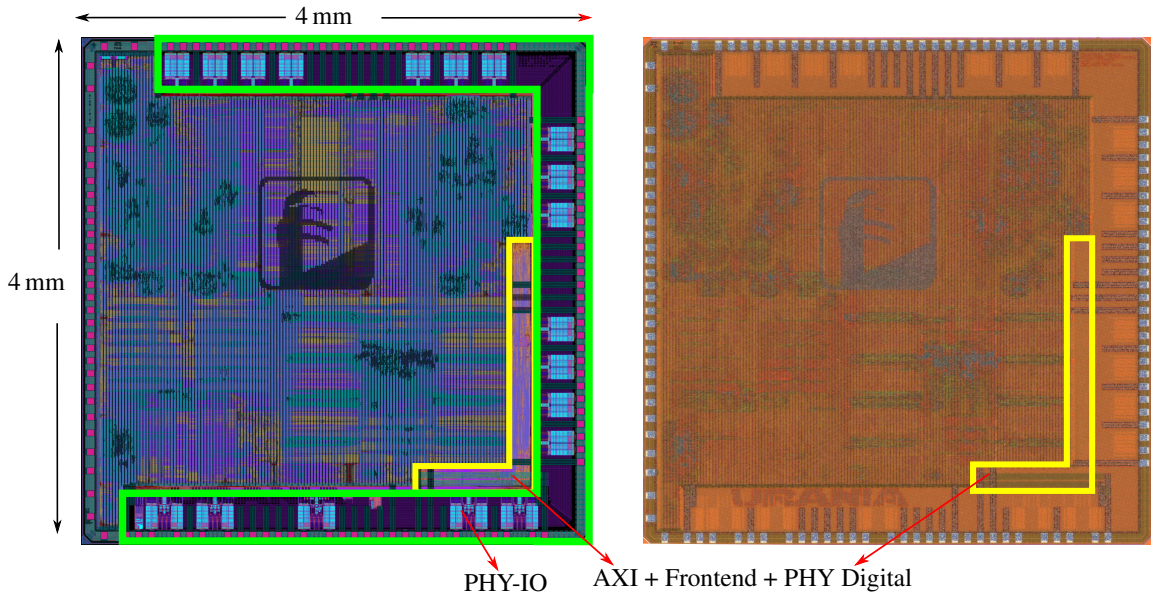

Figure 8.4: DDR3 memory controller post-layout and die photo

Table 8.1: Area Distribution of the Controller and PHY

Component	*Area*
PHY-IO transceiver	3.820 mm^2
PHY-Digital	0.551 mm^2
Frontend	0.339 mm^2

frontend) is extensively lean, consuming only 2% of the total chip area. The frontend of the controller also has a very low latency of 3 slow clock cycles plus 1 slow clock cycle for processing the host interface (AXI4) requests. The post-layout results show that the DDR3 controller achieves a performance of 533 MHz (PHY clock) leading to a data rate of 1066 Mbit/pin/s under worst-case conditions (i.e slow process, low VDD and high temperature). The peak frequency of the design is limited due to the package (QFN64) used for the mW demonstrator. The power estimation of the memory controller is done using a random trace with an equal number of reads and writes as an input to the controller that resulted in a data bus utilization of 60%. Table 8.2 shows the distribution of the estimated power for the frequencies 166 MHz and 500 MHz. Note that the vendors of commercially

Table 8.2: Power Distribution of the Controller and PHY

Component	*Power at 166 MHz*	*Power at 500 MHz*
Frontend + PHY Digital	6.312 mW	18.764 mW
PHY-IO ADDR/CMD block	19.98 mW	59.96 mW
PHY-IO DATA block	15.96 mW	50.61 mW

available DDR3 controllers do not disclose the power and performance values of their IPs. Hence, it is not feasible to compare our results against the other controllers. To the best of our knowledge, there were no open source memory controllers for ASICs at the time of our memory controller design (i.e. between mid 2017 to early 2019) and during the publication of the new contributions in [68] (2019). This makes the comparison of this controller even more challenging.

8.2 Summary

In this work, I presented a memory controller that is tailored for transprecision computing targeting IoT/embedded applications. The frontend of this memory controller adopted several advanced techniques, such as approximate DRAM, sophisticated refresh policy, and optimal address mapping and data placement by exploiting application knowledge. The experimental results show that the memory controller's design is lean, has low latency, and low power. Furthermore, this memory controller's frontend module is migrated to FPGA platform to be integrated with Xilinx PHY module and has been open-sourced [95]. The FPGA version is also adapted in the BMBF project ZuSE-KI-AVF under grant 16ME0379 [82]. Finally, the proposed hardware implementation of memory controller for commodity DRAMs will also be the reference of comparison to future memory controllers implemented for DRAM-PIM architectures.

Chapter 9

Conclusion and Future Work

The main objective of this thesis was to investigate novel DRAM PIM architectures for DNN inference and training. I had set three main research goals which are as follows: 1) Investigate new DRAM-PIM architectures that are compatible with existing commodity DRAM architectures, incur minimal modifications to the dense structure of DRAM banks, strictly do not modify the Sub-Arrays, and reuse most of the existing DRAM circuitry. By sufficing this goal, the new DRAM-PIM architectures implicitly support an appealing feature called "Dual-Mode" that enables them to operate as both a high-density main memory and an energy-efficient accelerator. 2) Explore architectures that enable the seamless integration of computation units in the PSA region (i.e. near Sub-Array) without violating its area constraints. The focus of this goal was to exploit the highest data parallelism and lowest data movement energy consumption of the PSA region for achieving the highest energy efficiency while maintaining a low area overhead in the area sensitive PSA region. 3) Propose novel DRAM PIM architectures that is scalable such that the PSA region is capable of supporting new computation units with minimal modifications to the core structure. Thus aiming to extend the DRAM-PIM architectures for other memory-intensive applications. The third goal also focused on investigating a novel interface protocol that enables DRAM-PIM die from any vendor or generation to be a drop-in replacement of the other without modifying the memory controller hardware design or its interface.

The first part of this research focused only on DNN inference and Goal 1 & 2. I presented two novel architectures (i.e. digital and mixed-signal approach) that adapted approximation techniques such as shifter based MAC and bit-decomposed analog MAC, respectively, to minimize the area overhead in the PSA region. All the substantial area-intensive logic was integrated in the bank peripheral region. My research work was the first to present a mixed-signal architecture for DRAM-PIM. It achieved the best results and required the lowest modifications to the PSA region among the two. This novel 8 Gb DRAM PIM device achieves a peak performance of 142.8 GOPS (23.813 GOPS/bank) while consuming a power of 945.765 mW, which results in the highest energy efficiency of 151.07 GOPS/W in comparison to the prior DRAM-PIM publications. The area overhead of the proposed

PIM device is 12.63% in comparison to a commodity 8 Gb DRAM in a 2y nm technology, while the digital architecture had an area overhead of 6.6%. However, in the area sensitive PSA region of near Sub-Array region, the mixed signal architecture has 4% overhead with seamless integration compared to 30% in case of digital architecture. For the mixed-signal architecture, I also presented the impact of device variations and various techniques to mitigate the error due to such variations.

In the second part, I proposed novel DRAM-PIM architectural approaches to enable the computation of both inference and training tasks. I presented a new array architecture called QOBL that enabled seamless integration of area-intensive computation units in the PSA region without any modification of the Sub-Array or PSA design. Thereby ensuring scalability goal in addition to Goal 1& 2. However, the DRAM-PIM bank with QOBL has a capacity loss per bank up to 75%. Hence, to compensate for the capacity loss, I presented a new device-level architecture with two types of banks where larger conventional commodity DRAM banks are used for high-capacity data storage purposes and smaller QOBL based PIM banks for energy-efficient computation. This DRAM-PIM device resulted in a cell density of 181.01 Mb/ mm^2 for 1y nm DRAM technology, which is 48% more than one of the memory vendors DRAM-PIM architecture i.e. Hynix-AiM. Furthermore, I also proposed a new unified DRAM-PIM interface and protocol that eliminated the need for vendor-specific memory controller design, similar to commodity DRAM counterparts. It also enabled the servicing of both PIM and Non-PIM requests without any long stalls. In the future, a detailed system-level analysis of these novel approaches has to be performed to further quantify the advantages.

In the third part, I proposed novel DRAM and memory controller architecture to address the memory wall and high DRAM energy consumption issues in the conventional Non-PIM compute platforms. I presented a novel DRAM architecture referred to as *Dual-Sense-Amplifier* (DSA) that is capable of flexibly opening any two rows within a Sub-Array or a bank without any collisions or page restrictions. This architecture reduces the data access latency and increases bandwidth utilization by masking the page-miss penalty within the DRAM bank. A DSA-based 8 Gb DRAM device increases the bandwidth utilization by 17.53%, reduces the average response latency by 31.59 ns (≈25 clk cycles), and consumes approximately the same energy as compared to commodity DRAM. Furthermore, it has a low area overhead of 9.6%, for a 2y nm technology 8 Gb device compared to a commodity DRAM device. Finally, I presented a lean, low latency, and low power memory controller that is tailored for transprecision computing targeting IoT/embedded applications. This memory controller adopted several advanced techniques, such as approximate DRAM, sophisticated refresh policy, and optimal address mapping to exploit application knowledge and reduce the overall DRAM energy. To the best of our knowledge, this is the first open-source memory controller hardware implementation to combine all these advanced techniques. The memory controller consumes an average power of 129.33 mW at a frequency of 500 MHz, has a total area of 4.71 mm^2 for UMC 65 nm process and the frontend of the controller (i.e. excluding PHY) has a latency of three clock cycles.

In total this thesis has resulted in 9 research publications [49, 61–68] and 3 patent filings [69–71]. In addition to these publications and patents, during my thesis research period between 2017 to 2023, I have also co-authored 13 more research publications [72–84] and I'm the co-inventor of one patent [85].

Zusammenfassung

Neu aufkommende Anwendungen, die auf maschinellem Lernen und Deep Neural Networks (DNNs) basieren, sind datengetrieben und speicherintensiv. Daher gibt es in jüngster Zeit eine Verschiebung von rechenzentrierten Architekturen hin zu speicherzentrierten Architekturen. In diesem Kontext ist Processing-in-Memory (PIM) ein neu aufkommendes Rechenparadigma, um Speicherprobleme wie Bandbreitenbegrenzung, lange Datenzugriffslatenz und sehr hohe Datenzugriffsenergie zu bewältigen, die bei konventionellen Rechenplattformen für speicherintensive Anwendungen auftreten. Die enge Kopplung von Berechnungseinheiten und Speicherarrays ermöglicht die effektive Ausnutzung der hohen internen Datenbandbreite und verringert die Energie, die zur Bewegung der Daten notwendig ist. Forscher haben verschiedene Speicher wie SRAM, DRAM und neu aufkommende Speicher (d.h. RRAM, FEFET, MRAM usw.) als PIM-Kandidaten untersucht. Die Nachteile von SRAMs und neu aufkommenden Speichern für PIM sind, dass 1) SRAM-basierte PIM-Beschleuniger eine geringe Speicherkapazität besitzen und 2) neu aufkommende Speicher technologisch nicht so ausgereift sind wie DRAMs oder SRAMs. Im Gegensatz zu SRAMs und RRAMs sind DRAMs technologisch ausgereift und weisen gleichzeitig eine hohe Speicherkapazität auf, was sie zu einem vielversprechenden Kandidaten für PIM macht. Darüber hinaus sind DRAMs aufgrund ihrer hohen Dichte und Kapazität einer der am weitesten verbreiteten externen Hauptspeicher. DRAM-basierte PIMs sind jedoch im Vergleich zu PIMs, die auf SRAMs oder neu aufkommenden Speichern basieren, relativ wenig erforscht. Aus der Perspektive des Energiemehraufwands (bei konventionellen Rechenplattformen) verbraucht der Zugriff auf ein Datenwort (z.B. 64 Bit) in einem externen DRAM 10 bis 50 mal mehr Energie als der Zugriff auf die Daten in einem lokalen SRAM-Speicher (Tabelle 1.1). Aufgrund dieser genannten Kriterien konzentriert sich diese Forschungsarbeit auf DRAM-Device-basiertes PIM. Vor Beginn dieser Dissertationsarbeit im Jahr 2018 gab es meines Wissens nur zwei weitere Veröffentlichungen, die sich mit DRAM-PIM-Architekturen befassten, nämlich AMBIT [28] (2017) und DRISA [29] (2017). Seitdem gibt es einen Anstieg von Veröffentlichungen über DRAM-PIM-Architekturen sowohl aus der akademischen Welt als auch aus der Industrie [20, 30–40]. Erst kürzlich in den Jahren 2021/22 haben auch führende Speicherhersteller wie Samsung und SK Hynix ihre jeweiligen ersten Engineering-Samples von DRAM-PIM in [34] (FIMDRAM, 2021) und [37] (Hynix-AiM, 2022) veröffentlicht. DRAM-PIM-Beschleunigerchips von Start-ups wie UPMEM sind ebenfalls kommerziell erhältlich [35].

Die Commodity-DRAM-Technologie unterscheidet sich von Standard-CMOS-Logiktechnologie-kompatiblen SRAMs oder einigen aufkommenden Speichertypen. Sie ist stark auf Dichte, Ausbeute und damit Kosten pro Bit optimiert. Commodity-DRAM-Architekturen enthalten auch keine Berechnungseinheiten abgesehen von der Peripherielogik für den Datenzugriff. Deshalb integrieren die meisten State-of-the-Art-DRAM-PIM-Architekturen die Berechnungseinheiten am Ausgang der DRAM-Bank (d. h. Bankperipheriebereich), wodurch die vorhandene DRAM-Bankarchitektur wiederverwendet werden kann. Dabei wird ein sehr praktisches Feature bekannt als Dual-Mode ermöglicht, was bedeutet, dass die Architekturen sowohl als Hauptspeicher mit hoher Dichte als auch als energieeffizienter Beschleuniger eingesetzt werden können. Bei einem Commodity-DRAM liegt die höchste Datenparallelität und die niedrigste Datenbewegungsenergie im Sub-Array-Bereich vor, d.h. innerhalb der DRAM-Bank. Beispielsweise beträgt die Menge an parallelen Daten, die am DDR4-Bankausgang verfügbar ist, 128 Bits, während sie im Sub-Array-Bereich 2 KB beträgt. Folglich haben einige State-of-the-Art-Architekturen [28, 30, 31, 33] die Integration von Berechnungseinheiten innerhalb der DRAM-Bänke untersucht, um den Durchsatz und die Energieeffizienz im Vergleich zum Bankperipheriebereichsansatz weiter zu erhöhen. Diese Architekturen erforderten jedoch die Modifikation eines stark optimierten DRAM-Sub-Array-Designs. Einige frühere Veröffentlichungen wie [29,32] integrierten DNN-Berechnungseinheiten in der Nähe des Sub-Arrays oder im PSA-Bereich, was keine spezifische PIM-Modifikation des Sub-Arrays erfordert. Diese Architekturen strukturierten jedoch die Bank- und Sub-Array-Abmessungen von DRAMs in 1024 Banken und 256×2096 Sub-Arrays um, was für Commodity-DRAM-Architekturen untypisch ist. Darüber hinaus ist der Commodity-DRAM-PSA-Bereich sehr sensibel gegenüber zusätzlicher Logik und wird insbesondere durch die Sub-Array-Breite, den geringen Bit-Line-Abstand und die begrenzte Anzahl verfügbarer Metalllagen (d. h. drei oder vier) im DRAM-Prozess restriktiert. Das PSA-Sharing in der OBL-Architektur restriktiert die Integration großer Berechnungseinheiten in diesem Bereich zusätzlich. Metallbahnen müssen reserviert werden, um Bit-Lines und Referenz-Bit-Lines durch die Berechnungseinheiten hindurchzuführen und mit den entsprechenden PSAs zu verbinden, was zu Flächenbeschränkungen führt. Daher gibt es drei wesentliche Forschungslücken:

- Untersuchung von DRAM-PIM-Architekturen, die die Berechnungseinheit innerhalb der Commodity-DRAM-Bankarchitektur mit minimalen Modifikationen platziert, die das Dual-Mode-Feature unterstützen und die die Sub-Arrays nicht modifizieren.

- Exploration von Architekturen, die die nahtlose Integration von Berechnungseinheiten im PSA-Bereich (d. h. in der Nähe des Sub-Arrays) ermöglichen, ohne die Flächenbeschränkungen zu verletzen.

- Vorschlag einer DRAM-PIM-Architektur, die skalierbar ist, sodass der PSA-Bereich in der Lage ist, jede flächenintensive Berechnungseinheit zu unterstützen. Untersuchung einer neuen einheitlichen Schnittstelle und eines neuen einheitlichen Proto-

kolls, zu denen alle Hersteller kompatibel sind, ohne das Speicherinterface oder das Hardware-Design des Speichercontrollers zu verändern.

Das Hauptziel dieser Arbeit ist es, neue DRAM-PIM-Architekturen für DNN-Anwendungen zu untersuchen, die die genannten Forschungslücken adressieren. Darüber hinaus trägt diese Arbeit auch zur Optimierung der Latenz, Energie und Bandbreitenausnutzung des DRAM-Subsystems bei konventionellen Nicht-DRAM-PIM-Rechenplattformen bei. Eine Zusammenfassung der neuartigen Beiträge wird kapitelweise aufgelistet.

Design Space Exploration (Chapter 3)
Der Großteil der früheren Veröffentlichungen bietet keinen umfassenden Überblick über den DRAM-PIM-Entwurfsraum und keine kritische Bewertung der verschiedenen Entwurfsdimensionen. Meine Analyse deckt die folgenden Entwurfsdimensionen des DRAM-PIM-Entwurfsraums ab: 1) Platzierung der Berechnungseinheiten, 2) Datenformat für die Berechnungen und 3) Speichercontroller- und Interfacemodifikationen, die für die externe Steuerung oder Programmierung des PIM benötigt werden In diesem Kapitel untersuche ich die verschiedenen Entwurfsrestriktionen und Implementierungsherausforderungen bei der Integration von Berechnungseinheiten in die Commodity-DRAM-Architektur. Die in diesem Kapitel vorgestellten Erkenntnisse bilden die Grundlage für alle Untersuchungen und Entwurfsentscheidungen zu neuartigen DRAM-PIM-Architekturen. Darüber hinaus wird der aktuelle Stand der Technik von DRAM-PIM-Architekturen kurz dargestellt.

DRAM PIM Architecture for DNN Inference (Chapter 4 and 5)

Mein Ziel ist es, neue DRAM-PIM-Architekturen für DNN-Inferenz zu untersuchen, die die oben genannten Forschungslücken schließen. Die wichtigste Entwurfsentscheidung in meinen Architekturen ist die Aufteilung der Berechnungseinheiten in zwei Teile, d.h. eine schlanke PPU, die in der PSA-Region integriert ist, und eine größere SPU, die nahe der Bankperipherie-Region integriert ist. Diese Aufteilung zwischen der ersten und zweiten Einheit ist erforderlich, um die Flächenbeschränkungen der PSA-Region einzuhalten. Um den Flächen-Overhead in der PSA-Region noch weiter zu verringern, werden in meinen Architekturen zwei algorithmische Optimierungen vorgenommen: 1) Logarithmische Quantisierung, die einen Shifter-basierten näherungsweisen Multiplizierer verwendet, und 2) Bit-zerlegtes MAC. Dieser Beitrag ist in zwei Kapitel unterteilt, in denen ich in Kapitel 4 eine volldigitale auf logarithmischer Quantisierung basierende Architektur und in Kapitel 5 einen auf Bit-zerlegtem MAC basierenden Mixed-Signal-Ansatz vorstelle. Meine Forschungsarbeit war die erste, die eine Mixed-Signal-Architektur für DRAM-PIM vorstellte. Von diesen beiden erzielte der Mixed-Signal-Ansatz die besten Ergebnisse und erforderte die geringsten Modifikationen in der PSA-Region. Dieses neuartige 8-Gb-DRAM-PIM-Device erreicht in der Spitze eine Rechenleistung von 142,8 GOPS (23,813 GOPS/-Bank) bei einem Stromverbrauch von 945,765 mW, was im Vergleich zu den früheren

DRAM-PIM-Veröffentlichungen die höchste Energieeffizienz von 151,07 GOPS/W ergibt. Der Flächen-Overhead des vorgeschlagenen PIM-Devices beträgt 12,63% im Vergleich zu einem Commodity-8-Gb-DRAM in einer 2y-nm-Technologie.

DRAM PIM Architectural Approaches for Scalability (Chapter 6)

Ich habe neuartige DRAM-PIM-Architekturansätze vorgeschlagen, um sowohl die Berechnung von Inferenz- als auch von Trainingsprozessen zu ermöglichen. Ich habe eine neue Array-Architektur namens QOBL vorgestellt, die eine nahtlose Integration von flächenintensiven Berechnungseinheiten in der PSA-Region ermöglicht, ohne dass das Sub-Array- oder PSA-Design geändert werden muss. Mein erster Architekturansatz, der als QOBL bezeichnet wird, ermöglicht die nahtlose Integration von Berechnungseinheiten (sowohl Gleitkomma als auch Ganzzahl) ohne jegliche Änderung des DRAM-Sub-Array- oder PSA-Designs. Allerdings verliert die DRAM-PIM-Bank mit QOBL bis zu 75% ihrer Speicherkapazität. Um diesen Kapazitätsverlust auszugleichen, habe ich eine neue Architektur auf Device-Ebene mit zwei Arten von Bänken vorgestellt, wobei größere herkömmliche DRAM-Bänke für die Speicherung von Daten mit hoher Kapazität und kleinere PIM-Bänke auf QOBL-Basis für energieeffiziente Berechnungen verwendet werden. Dieses DRAM-PIM-Device erreichte eine Zelldichte von 181,01 Mb/mm^2 in der 1y-nm-DRAM Technologie, was 48% mehr ist als die Zelldichte des industriellen DRAM-PIMs Hynix-AiM. Darüber hinaus habe ich ähnlich wie bei Commodity-DRAMs eine neue, einheitliche DRAM-PIM-Schnittstelle und ein Protokoll vorgeschlagen, die einen herstellerspezifischen Speichercontroller überflüssig machen. Dies ermöglichte zudem die gemischte Abarbeitung von PIM- und Nicht-PIM-Anfragen ohne lange Wartezeiten.

A Novel DRAM Architecture for Improved Bandwidth Utilization and Latency Reduction using Dual-Page Operation (Chapter 7)

In meiner Dissertation werden nicht nur neuartige PIM-Architekturen vorgestellt, sondern auch neuartige DRAM-Architekturen für herkömmliche Nicht-PIM-Berechnungsplattformen vorgeschlagen. Um die Auswirkungen des "Memory Wallsïn konventionellen Berechnungsplattformen zu reduzieren, muss die Datenzugriffslatenz verringert und die Bandbreitenausnutzung erhöht werden. Eine der wesentlichen latenzverursachenden Operationen in konventionellem DRAM sind "Page Misses"(d. h. das Öffnen einer neuen DRAM Speicherseite/Speicherzeile innerhalb einer Bank). Das gleichzeitige Öffnen mehrerer Zeilen innerhalb einer Bank ist eine Möglichkeit, diese Latenzen zu verbergen. Ich stelle eine neuartige DRAM-Architektur vor, die als *Dual-Sense-Amplifier* (DSA) bezeichnet wird und in der Lage ist, zwei beliebige Zeilen innerhalb eines Sub-Arrays oder einer Bank ohne Kollisionen oder Beschränkungen flexibel zu öffnen. Die vorgeschlagene Architektur ist an jeden DRAM-Typ anpassbar, z. B. HBM, GDDR usw., und ersetzt jedes Commodity-DRAM in den herkömmlichen Plattformen. Meine Architektur behält den Großteil des DRAM-Designs/Schaltkreise bei und führt nur zu minimalen Änderungen im DRAM. Ein DSA-basiertes 8-Gb-DRAM-Device erhöht die Bandbreitenausnutzung um 17,53 Die vor-

geschlagene DSA-Architektur hat einen geringen Flächen-Overhead von 9,6

DRAM Memory Controller Design for Transprecision Applications (Chapter 8)
In meiner Dissertation schlage ich nicht nur Lösungen vor, die die Commodity-DRAM-Architektur verändern, sondern untersuche auch die Möglichkeit, die DRAM-Energie mithilfe einer neuartigen DRAM-Speichercontrollerarchitektur zu reduzieren. Der Energieverbrauch ist eine der größten Herausforderungen für ein fortschrittliches System-on-Chip. Dieses Problem wird durch den Einsatz heterogener und näherungsweiser Berechnungsmethoden adressiert. Eine der jüngsten Entwicklungen in diesem Zusammenhang ist das Paradigma des "Transprecision Computing". Die Idee des "Transprecision Computing"besteht darin, für jede Operation eine angemessene Energiemenge zu verbrauchen, indem die Präzision dynamisch reduziert wird. Die Auswirkungen des Speichersubsystems spielen bei solchen Systemen eine entscheidende Rolle. Daher kann die Energieeffizienz eines Transprecision-Systems weiter optimiert werden, indem das Speichersubsystem auf "Transprecision Computingßugeschnitten wird. In diesem Kapitel stelle ich einen schlanken Speichercontroller (ASIC-Design) mit geringem Stromverbrauch und niedriger Latenz vor, der für Transprecision-Systeme geeignet ist. Der Speichercontroller verbraucht durchschnittlich 129,33 mW bei einer Frequenz von 500 MHz und hat eine Gesamtfläche von 4,71 mm^2 im UMC 65-nm-Prozess.

Diese Arbeit wurde teilweise durch das EU-Projekt OPRECOMP unter dem Förderkennzeichen 732631, durch das Bundesministerium für Bildung und Forschung (BMBF) im Rahmen des Projekts Open6GHuB unter dem Förderkennzeichen 16KISK004, durch die Carl-Zeiss-Stiftung unter der Förderkennzeichen SSustainable Embedded AI", durch das Bundesministerium für Bildung und Forschung (BMBF) im Rahmen des Projekts ZuSE-KI-AVF unter dem Förderkennzeichen 16ME0379 und durch Huawei Technologies Co. finanziert.

Acronyms

ADC	Analog to Digital Convertor 65, 68
AI	Artificial Intelligence 1
CNN	Convolutional Neural Network 47, 56, 63, 65, 67, 79, 82
CSL	Column Selectline 20, 22, 26, 50, 51, 70, 71, 76, 79
CSLEN	Column Select Enable 20, 22
CSS	Column Select Switches 22, 24, 49–51, 63, 70, 148
DCC	Dual Contact Cell 34–36
DNN	Deep Neural Network 1–6, 9–11, 13, 14, 37, 42, 45, 54, 59, 88, 91, 126, 127, 147
DSA	Dual-Sense-Amplifier 11, 12, 15, 103
FBL	Folded Bitline 92–94
LBL	Local Bitline . . . 20, 23–26, 34, 35, 37, 49, 50, 63, 70, 92–94, 103, 104, 109, 148
LWL	Local Wordline 20, 23, 25, 104
LWLD	Local Wordline Driver 20, 73, 74
MAC	Multiply-Accumulate 9–11, 13, 14, 37, 40, 46–49, 53, 59, 61, 65, 67, 68, 91, 99, 121, 127
MBL	Master Bitline 20–22, 25, 49–53, 70
MWL	Master Wordline 20, 22, 25
OBL	Open Bitline 10, 24, 35–37, 92–95, 126, 150
PIM	Processing-in-Memory . . 5–14, 34, 36, 38, 45, 46, 48, 49, 52–56, 58–60, 67, 68, 77, 78, 83, 91–93, 101, 120, 121, 126–128, 148
PPU	Primary Processing Unit 10, 13, 45, 48–53, 56–58, 70, 71, 78, 96, 127, 148, 149

PSA Primary Sense Amplifier 9–11, 13, 14, 20–26, 33–38, 45–52, 55, 63, 65, 68, 70, 73, 74, 83, 87, 88, 91–94, 99, 103, 104, 107, 109, 121, 122, 126–128

QOBL Quasi Open Bitline 14, 91–96, 99, 122, 128, 150

RCAT Recessed Channel Array Transistor 7, 23
ReLU Rectified Linear Unit 46, 53, 74

SHL Shadow Latch 49, 63
SOC System on Chips 15
SPU Secondary Processing Unit 10, 13, 46, 48, 50, 52, 53, 78, 96, 127
SRCAT Sphere-shaped Recessed Channel Array Transistor 7, 23
SSA Secondary Sense Amplifier 20–22, 25, 38, 50–53

VACT Vertical Activation 73

WCS Weighted Current Summation 65, 67–69, 91, 149

Bibliography

[1] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[2] W. Brattain. *Bell Labs logbook*, pages 7–8, 24. 1947.

[3] W. Shockley. The theory of p-n junctions in semiconductors and p-n junction transistors. *The Bell System Technical Journal*, 28(3):435–489, 1949.

[4] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[5] G. E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter*, 11(3):33–35, 2006.

[6] W. M. Holt. 1.1 Moore's law: A path going forward (ISSCC16 Presentation). `https://www.youtube.com/watch?v=LbdwbsBbODM&t=296s`, 2016.

[7] P. Gelsinger. Semiconductors Run the World (Hot Chips 34 Keynote). `https://youtu.be/I0PSz2aDjQI?t=661`, 2022.

[8] A. Gholami, Z. Yao, S. Kim, M. W. Mahoney, and K. Keutzer. AI and Memory Wall. *RiseLab Medium Post*, 2021.

[9] T. D. Forums. Parasitics: an old problem reaches new heights, March 1, 2008.

[10] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pages 1–12, New York, NY, USA, 2017. ACM.

[11] W. A. Wulf and S. A. McKee. Hitting the memory wall: Implications of the obvious. *SIGARCH Comput. Archit. News*, 23(1):20–24, mar 1995.

[12] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens. Memory access scheduling. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, page 128–138, 2000.

[13] N. P. Jouppi, D. Hyun Yoon, M. Ashcraft, M. Gottscho, T. B. Jablin, G. Kurian, J. Laudon, S. Li, P. Ma, X. Ma, T. Norrie, N. Patil, S. Prasad, C. Young, Z. Zhou, and D. Patterson. Ten Lessons From Three Generations Shaped Google's TPUv4i : Industrial Product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–14, 2021.

[14] S. Williams, A. Waterman, and D. Patterson. Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, apr 2009.

[15] N. Jouppi, D. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson. A Domain-Specific Supercomputer for Training Deep Neural Networks. *Communications of the ACM*, 63:67–78, 06 2020.

[16] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen. Cambricon-X: An accelerator for sparse neural networks. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12, 2016.

[17] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam. DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning. *SIGARCH Comput. Archit. News*, 42(1):269–284, feb 2014.

[18] D. Stathis, C. Sudarshan, Y. Yang, M. Jung, S. Jafri, C. Weis, A. Hemani, A. Lansner, and N. Wehn. eBrainII: A 3 kW Realtime Custom 3D DRAM integrated ASIC implementation of a Biologically Plausible Model of a Human Scale Cortex. *Journal of Signal Processing Systems*, 11 2019.

[19] H. Zhang, X. Wu, Y. Du, H. Guo, C. Li, Y. Yuan, M. Zhang, and S. Zhang. A Heterogeneous RISC-V Processor for Efficient DNN Application in Smart Sensing System. *Sensors (Basel, Switzerland)*, 21, 2021.

[20] S. Lee, S.-h. Kang, J. Lee, H. Kim, E. Lee, S. Seo, H. Yoon, S. Lee, K. Lim, H. Shin, J. Kim, O. Seongil, A. Iyer, D. Wang, K. Sohn, and N. S. Kim. Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology : Industrial Product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021.

[21] Y.-F. Kao, W. C. Zhuang, C.-J. Lin, and Y.-C. King. A Study of the Variability in Contact Resistive Random Access Memory by Stochastic Vacancy Model. *Nanoscale Research Letters*, 13(1):213, Jul 2018.

[22] A. Chen and M. Lin. Variability of resistive switching memories and its impact on crossbar array performance. In *2011 International Reliability Physics Symposium*, pages MY.7.1–MY.7.4, April 2011.

[23] G. Molas and E. Nowak. Advances in Emerging Memory Technologies: From Data Storage to Artificial Intelligence. *Applied Sciences*, 11(23), 2021.

[24] B. Li, B. Yan, and H. H. Li. An Overview of In-memory Processing with Emerging Non-volatile Memory for Data-intensive Applications. *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, 2019.

[25] S. Beyer, S. Dünkel, M. Trentzsch, J. Müller, A. Hellmich, D. Utess, J. Paul, D. Kleimaier, J. Pellerin, S. Müller, J. Ocker, A. Benoist, H. Zhou, M. Mennenga, M. Schuster, F. Tassan, M. Noack, A. Pourkeramati, F. Müller, M. Lederer, T. Ali, R. Hoffmann, T. Kämpfe, K. Seidel, H. Mulaosmanovic, E. T. Breyer, T. Mikolajick, and S. Slesazeck. FeFET: A versatile CMOS compatible device with game-changing potential. In *2020 IEEE International Memory Workshop (IMW)*, pages 1–4, 2020.

[26] M. Imani, S. Gupta, Y. Kim, and T. Rosing. Floatpim: In-memory acceleration of deep neural network training with high precision. In *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, pages 802–815, 2019.

[27] X. Peng, S. Huang, H. Jiang, A. Lu, and S. Yu. Dnn+neurosim v2.0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(11):2306–2319, 2021.

[28] V. Seshadri, D. Lee, T. Mullins, H. Hassan, A. Boroumand, J. Kim, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry. Ambit: In-memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, pages 273–287, New York, NY, USA, 2017. ACM.

[29] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie. DRISA: A DRAM-based Reconfigurable In-Situ Accelerator. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, pages 288–301, New York, NY, USA, 2017. ACM.

[30] Q. Deng, L. Jiang, Y. Zhang, M. Zhang, and J. Yang. DrAcc: A DRAM Based Accelerator for Accurate CNN Inference. In *Proceedings of the 55th Annual Design Automation Conference*, DAC '18, pages 168:1–168:6, New York, NY, USA, 2018. ACM.

[31] Q. Deng, Y. Zhang, M. Zhang, and J. Yang. LAcc: Exploiting Lookup Table-based Fast and Accurate Vector Multiplication in DRAM-based CNN Accelerator. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.

[32] S. Li, A. O. Glova, X. Hu, P. Gu, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie. SCOPE: A Stochastic Computing Engine for DRAM-Based In-Situ Accelerator. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 696–709, 2018.

[33] F. Zhang, S. Angizi, and D. Fan. Max-PIM: Fast and Efficient Max/Min Searching in DRAM. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 211–216, 2021.

[34] Y.-C. Kwon, S. H. Lee, J. Lee, S.-H. Kwon, J. M. Ryu, J.-P. Son, O. Seongil, H.-S. Yu, H. Lee, S. Y. Kim, Y. Cho, J. G. Kim, J. Choi, H.-S. Shin, J. Kim, B. Phuah, H. Kim, M. J. Song, A. Choi, D. Kim, S. Kim, E.-B. Kim, D. Wang, S. Kang, Y. Ro, S. Seo, J. Song, J. Youn, K. Sohn, and N. S. Kim. 25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications. In *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, volume 64, pages 350–352, 2021.

[35] F. Devaux. The true Processing In Memory accelerator. In *2019 IEEE Hot Chips 31 Symposium (HCS)*, pages 1–24, 2019.

[36] M. He, C. Song, I. Kim, C. Jeong, S. Kim, I. Park, M. Thottethodi, and T. N. Vijaykumar. Newton: A DRAM-maker's Accelerator-in-Memory (AiM) Architecture for Machine Learning. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 372–385, 2020.

[37] S. Lee, K. Kim, S. Oh, J. Park, G. Hong, D. Ka, K. Hwang, J. Park, K. Kang, J. Kim, J. Jeon, N. Kim, Y. Kwon, K. Vladimir, W. Shin, J. Won, M. Lee, H. Joo, H. Choi, J. Lee, D. Ko, Y. Jun, K. Cho, I. Kim, C. Song, C. Jeong, D. Kwon, J. Jang, I. Park, J. Chun, and J. Cho. A 1ynm 1.25V 8Gb, 16Gb/s/pin GDDR6-based Accelerator-in-Memory supporting 1TFLOPS

MAC Operation and Various Activation Functions for Deep-Learning Applications. In *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, volume 65, pages 1–3, 2022.

[38] H. Kim, H. Park, T. Kim, K. Cho, E. Lee, S. Ryu, H.-J. Lee, K. Choi, and J. Lee. GradPIM: A Practical Processing-in-DRAM Architecture for Gradient Descent. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 249–262, 2021.

[39] H. Shin, D. Kim, E. Park, S. Park, Y. Park, and S. Yoo. McDRAM: Low Latency and Energy-Efficient Matrix Computations in DRAM. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2613–2622, 2018.

[40] S. Cho, H. Choi, E. Park, H. Shin, and S. Yoo. McDRAM v2: In-Dynamic Random Access Memory Systolic Array Accelerator to Address the Large Model Problem in Deep Neural Networks on the Edge. *IEEE Access*, 8:135223–135243, 2020.

[41] H.-J. Chi, C.-K. Lee, J. Park, J.-S. Heo, J. Jung, D. Lee, D.-H. Kim, D. Park, K. Kim, S.-Y. Kim, J. Park, H. Cho, S. Lim, Y. Choi, Y. Lim, D. Moon, G. Park, J.-H. Jang, K. Lee, I. Hwang, C. Kim, Y. Son, G.-Y. Kang, K. Park, S. Lee, S.-Y. Doo, C.-H. Shin, B. Na, J. Kwon, K. R. Kim, H. Choi, S.-K. Choi, S. Chang, W. Bae, H.-J. Kwon, Y.-S. Sohn, S.-J. Bae, K.-I. Park, and J.-B. Lee. 22.2 An 8.5Gb/s/pin 12Gb-LPDDR5 SDRAM with a Hybrid-Bank Architecture using Skew-Tolerant, Low-Power and Speed-Boosting Techniques in a 2nd generation 10nm DRAM Process. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 382–384, 2020.

[42] J. Choe. DRAM Scaling Trend and Beyond. `https://www.techinsights.com/blog/dram-scaling-trend-and-beyond`, 2022.

[43] J. Choe. Memory Technology 2021: Trends & Challenges. In *2021 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, pages 111–115, 2021.

[44] J. M. Park, Y. S. Hwang, S.-W. Kim, S. Y. Han, J. S. Park, J. Kim, J. W. Seo, B. S. Kim, S. H. Shin, C. H. Cho, S. W. Nam, H. S. Hong, K. P. Lee, G. Y. Jin, and E. S. Jung. 20nm DRAM: A new beginning of another revolution. In *2015 IEEE International Electron Devices Meeting (IEDM)*, pages 26.5.1–26.5.4, 2015.

[45] C. Lim, K. Park, and S. Baeg. Active Precharge Hammering to Monitor Displacement Damage Using High-Energy Protons in 3x-nm SDRAM. *IEEE Transactions on Nuclear Science*, 64(2):859–866, 2017.

[46] A. Spessot and H. Oh. 1T-1C Dynamic Random Access Memory Status, Challenges, and Prospects. *IEEE Transactions on Electron Devices*, 67(4):1382–1393, 2020.

[47] X. Liu, Y. Sun, N. Wang, J. Jeon, B. Wu, and K. Cao. Saddle Fin Structure Effects on the DRAM Access Transistor Performance. In *2021 IEEE 4th International Conference on Electronics Technology (ICET)*, pages 50–53, 2021.

[48] A. Das. Hynix DRAM layout, process integration adapt to change, 2012.

[49] C. Sudarshan, J. Lappas, M. M. Ghaffar, V. Rybalkin, C. Weis, M. Jung, and N. Wehn. An In-DRAM Neural Network Processing Engine. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2019.

[50] A. Spessot and H. Oh. 1T-1C Dynamic Random Access Memory Status, Challenges, and Prospects. *IEEE Transactions on Electron Devices*, 67(4):1382–1393, 2020.

[51] JEDEC Solid State Technology Association. DDR3 SDRAM Standard - JESD79-3F, 2012.

[52] JEDEC Solid State Technology Association. DDR4 SDRAM Standard -JESD79-4D, 2021.

[53] JEDEC Solid State Technology Association. DDR5 SDRAM Standard - JESD79-5B, 2022.

[54] JEDEC Solid State Technology Association. Low Power Double Data Rate 4 (LPDDR4) - JESD209-4D, 2021.

[55] JEDEC Solid State Technology Association. Low Power Double Data Rate 5 (LPDDR5) - JESD209-5B, 2021.

[56] JEDEC Solid State Technology Association. GRAPHICS DOUBLE DATA RATE (GDDR6) SGRAM STANDARD - JESD250C, 2021.

[57] JEDEC Solid State Technology Association. High Bandwidth Memory DRAM (HBM1, HBM2) - JESD235D, 2021.

[58] G. Armeniakos, G. Zervakis, D. Soudris, and J. Henkel. Hardware Approximate Techniques for Deep Neural Network Accelerators: A Survey. *ACM Comput. Surv.*, 55(4), nov 2022.

[59] S. Kim, G. Park, and Y. Yi. Performance Evaluation of INT8 Quantized Inference on Mobile GPUs. *IEEE Access*, 9:164245–164255, 2021.

[60] B. Darvish Rouhani, D. Lo, R. Zhao, M. Liu, J. Fowers, K. Ovtcharov, A. Vinogradsky, S. Massengill, L. Yang, R. Bittner, A. Forin, H. Zhu, T. Na, P. Patel, S. Che, L. Chand Koppaka, X. SONG, S. Som, K. Das, S. T, S. Reinhardt, S. Lanka, E. Chung, and D. Burger. Pushing the Limits of Narrow Precision Inferencing at Cloud Scale with Microsoft Floating Point. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 10271–10281. Curran Associates, Inc., 2020.

[61] V. Rybalkin, C. Sudarshan, C. Weis, J. Lappas, N. Wehn, and L. Cheng. Efficient Hardware Architectures for 1D- and MD-LSTM Networks. *J. Signal Process. Syst.*, 92(11):1219–1245, nov 2020.

[62] M. M. Ghaffar, C. Sudarshan, C. Weis, M. Jung, and N. Wehn. A Low Power In-DRAM Architecture for Quantized CNNs Using Fast Winograd Convolutions. In *The International Symposium on Memory Systems*, MEMSYS 2020, page 158–168, New York, NY, USA, 2020. Association for Computing Machinery.

[63] C. Sudarshan, T. Soliman, C. De la Parra, C. Weis, L. Ecco, M. Jung, N. Wehn, and A. Guntoro. A Novel DRAM-Based Process-in-Memory Architecture and Its Implementation for CNNs. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, ASPDAC '21, page 35–42, New York, NY, USA, 2021. Association for Computing Machinery.

[64] C. Sudarshan, T. Soliman, J. Lappas, C. Weis, M. H. Sadi, M. Jung, A. Guntoro, and N. Wehn. A Weighted Current Summation Based Mixed Signal DRAM-PIM Architecture for Deep Neural Network Inference. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 12(2):367–380, 2022.

[65] C. Sudarshan, M. Hassani Sadi, L. Steiner, C. Weis, and N. Wehn. A Critical Assessment of DRAM-PIM Architectures - Trends, Challenges and Solutions. In *Embedded Computer Systems: Architectures, Modeling, and Simulation*, 2022.

[66] C. Sudarshan, T. Soliman, T. Kämpfe, C. Weis, and N. Wehn. FeFET versus DRAM based PIM Architectures: A Comparative Study. In *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–6, 2022.

[67] C. Sudarshan, L. Steiner, M. Jung, J. Lappas, C. Weis, and N. Wehn. A Novel DRAM Architecture for Improved Bandwidth Utilization and Latency Reduction Using Dual-Page Operation. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 68(5):1615–1619, 2021.

[68] C. Sudarshan, J. Lappas, C. Weis, D. M. Mathew, M. Jung, and N. Wehn. A Lean, Low Power, Low Latency DRAM Memory Controller for Transprecision Computing. In D. N. Pnevmatikatos, M. Pelcat, and M. Jung, editors, *Embedded Computer Systems: Architectures, Modeling, and Simulation*, pages 429–441, Cham, 2019. Springer International Publishing.

[69] C. Li, W. Yang, G. Li, C. Sudarshan, and N. Wehn. Neural network computing device and method, and computing device (WO2020238843A1), 2020.

[70] C. Li, W. Yang, G. Li, C. Sudarshan, and N. Wehn. Long-short term memory LSTM network computing device and computing device (CN112862059A), 2021.

[71] A. Guntoro, C. Sudarshan, C. Weis, L. L. Ecco, T. Soliman, and N. Wehn. Memory device and method for performing consecutive memory accesses (US20220383937A1), 2022.

[72] T. Soliman, F. Müller, T. Kirchner, T. Hoffmann, H. Ganem, E. Karimov, T. Ali, M. Lederer, C. Sudarshan, T. Kämpfe, A. Guntoro, and N. Wehn. Ultra-Low Power Flexible Precision FeFET Based Analog In-Memory Computing. In *2020 IEEE International Electron Devices Meeting (IEDM)*, pages 29.2.1–29.2.4, 2020.

[73] M. Jung, S. A. McKee, C. Sudarshan, C. Dropmann, C. Weis, and N. Wehn. Driving into the Memory Wall: The Role of Memory for Advanced Driver Assistance Systems and Autonomous Driving. In *Proceedings of the International Symposium on Memory Systems*, MEMSYS '18, page 377–386, New York, NY, USA, 2018. Association for Computing Machinery.

[74] D. M. Mathew, M. Schultheis, C. C. Rheinländer, C. Sudarshan, C. Weis, N. Wehn, and M. Jung. An analysis on retention error behavior and power consumption of recent DDR4 DRAMs. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 293–296, 2018.

[75] K. Kraft, C. Sudarshan, D. M. Mathew, C. Weis, N. Wehn, and M. Jung. Improving the error behavior of DRAM by exploiting its Z-channel property. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1492–1495, 2018.

[76] D. Stathis, C. Sudarshan, Y. Yang, M. Jung, C. Weis, A. Hemani, A. Lansner, and N. Wehn. eBrainII: a 3 kW Realtime Custom 3D DRAM Integrated ASIC Implementation of a Biologically Plausible Model of a Human Scale Cortex. *Journal of Signal Processing Systems*, 92:1323 – 1343, 2019.

[77] M. Jung, K. Kraft, T. Soliman, C. Sudarshan, C. Weis, and N. Wehn. Fast Validation of DRAM Protocols with Timed Petri Nets. In *Proceedings of the International Symposium on Memory Systems*, MEMSYS '19, page 133–147, New York, NY, USA, 2019. Association for Computing Machinery.

[78] K. Kraft, D. M. Mathew, C. Sudarshan, M. Jung, C. Weis, N. Wehn, and F. Longnos. Efficient Coding Scheme for DDR4 Memory Subsystems. In *Proceedings of the International Symposium on Memory Systems*, MEMSYS '18, page 148–157, New York, NY, USA, 2018. Association for Computing Machinery.

[79] S. Müelich, S. Bitzer, C. Sudarshan, C. Weis, N. Wehn, M. Bossert, and R. F. Fischer. Channel Models for Physical Unclonable Functions based on DRAM Retention Measurements. In *2019 XVI International Symposium "Problems of Redundancy in Information and Control Systems" (REDUNDANCY)*, pages 149–154, 2019.

[80] C. Weis, M. Jung, F. Zulian, C. Sudarshan, D. M. Mathew, and N. Wehn. The Role of Memories in Transprecision Computing. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2018.

[81] M. V. Natale, M. Jung, K. Kraft, F. Lauer, J. Feldmann, C. Sudarshan, C. Weis, S. Krumke, and N. Wehn. Efficient generation of application specific memory controllers. In *The International Symposium on Memory Systems*, MEMSYS 2020, page 233–247, New York, NY, USA, 2021. Association for Computing Machinery.

[82] G. B. Thieu, S. Gesper, G. Payá-Vayá, C. Riggers, O. Renke, T. Fiedler, J. Marten, T. Stuckenberg, H. Blume, C. Weis, L. Steinert, C. Sudarshan, N. Wehn, L. M. Reimann, R. Leupers, M. Beyer, D. Köhler, A. Jauch, J. M. Borrmann, S. Jaberansari, T. Berthold, M. Blawat, M. Kock, G. Schewior, J. Benndorf, F. Kautz, H.-M. Bluethgen, and C. Sauer. ZuSE Ki-Avf: Application-Specific AI Processor for Intelligent Sensor Signal Processing in Autonomous Driving. In *2023 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6, 2023.

[83] S. Müelich, C. Sudarshan, C. Weis, M. Bossert, R. F.H. Fischer, and N. Wehn. Multi-Valued Physical Unclonable Functions Based on Dynamic Random Access Memory. In *The International Symposium on Memory Systems*, MEMSYS 2020, page 126–136, New York, NY, USA, 2021. Association for Computing Machinery.

[84] L. Steiner, C. Sudarshan, M. Jung, D. Stoffel, and N. Wehn. A framework for formal verification of dram controllers. In *ACM/IEEE International Symposium on Memory Systems (MEMSYS 2022)*, 10 2022.

[85] K. Kraft, D. M. Mathew, C. Sudarshan, M. Jung, C. Weis, N. Wehn, F. Longnos, G. LI, and W. Yang. Memory access technology and computer system for reducing data error probability (US11521674B2), 2022.

[86] W. J. Lee, C. H. Kim, Y. Paik, J. Park, I. Park, and S. W. Kim. Design of Processing-"Inside"-Memory Optimized for DRAM Behaviors. *IEEE Access*, 7:82633–82648, 2019.

[87] C. H. Kim, W. J. Lee, Y. Paik, K. Kwon, S. Y. Kim, I. Park, and S. W. Kim. Silent-PIM: Realizing the Processing-in-Memory Computing With Standard Memory Requests. *IEEE Transactions on Parallel and Distributed Systems*, 33(2):251–262, 2022.

[88] Y. Paik, C. H. Kim, W. J. Lee, and S. W. Kim. Achieving the Performance of All-Bank In-DRAM PIM With Standard Memory Interface: Memory-Computation Decoupling. *IEEE Access*, 10:93256–93272, 2022.

[89] B. Kim, J. Chung, E. Lee, W. Jung, S. Lee, J. Choi, J. Park, M. Wi, S. Lee, and J. H. Ahn. MViD: Sparse Matrix-Vector Multiplication in Mobile DRAM for Accelerating Recurrent Neural Networks. *IEEE Transactions on Computers*, 69(7):955–967, 2020.

[90] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu. A case for exploiting subarray-level parallelism (SALP) in DRAM. In *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, pages 368–379, 2012.

[91] T. Zhang, K. Chen, C. Xu, G. Sun, T. Wang, and Y. Xie. Half-DRAM: A high-bandwidth and low-power DRAM architecture from the rethinking of fine-grained activation. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pages 349–360, 2014.

[92] C. Zhang and X. Guo. Enabling efficient fine-grained DRAM activations with interleaved I/O. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6, 2017.

[93] S. Lym, H. Ha, Y. Kwon, C.-k. Chang, J. Kim, and M. Erez. ERUCA: Efficient DRAM Resource Utilization and Resource Conflict Avoidance for Memory System Parallelism. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 670–682, 2018.

[94] Special Issue on the 2021 IEEE International Symposium on Circuits and Systems. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS*, 68(5), 2021.

[95] DDR4 Memory Controller with XILINX Phy. `https://github.com/oprecomp/DDR4_controller`, 2021.

[96] M. Jung. *System-Level Modeling, Analysis and Optimization of DRAM Memories and Controller Architectures*. University of Kaiserslautern, 2017.

[97] B. Keeth, R. Baker, B. Johnson, and F. Lin. *DRAM Circuit Design : Fundamental and High-Speed Topics*. 01 2008.

[98] TechInsights. 1Y DRAM Analysis Product Brief, 2019.

[99] GlobalFoundries. 12lp+ technology design manual, 2021.

[100] U. Kwon. SK hynix Leading the Way in the HKMG Revolution, 2022.

[101] J. Fang, J. Lu, and M. Cai. Bank partitioning based adaptive page policy in multi-core memory systems. In *2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 240–243, 2015.

[102] J. H. Kim, S.-h. Kang, S. Lee, H. Kim, W. Song, Y. Ro, S. Lee, D. Wang, H. Shin, B. Phuah, J. Choi, J. So, Y. Cho, J. Song, J. Choi, J. Cho, K. Sohn, Y. Sohn, K. Park, and N. S. Kim. Aquabolt-XL: Samsung HBM2-PIM with in-memory processing for ML accelerators and beyond. In *2021 IEEE Hot Chips 33 Symposium (HCS)*, pages 1–26, 2021.

[103] D. Kim, M. Park, S. Jang, J.-Y. Song, H. Chi, G. Choi, S. Choi, J. Kim, C. Kim, K. Kim, K. Koo, S. Song, Y. Kim, D. U. Lee, J. Lee, D. Kim, K. Kwon, M. Han, B. Choi, H. Kim, S. Ku, Y. Kim, J. Kim, S. Kim, Y. Seo, S. Oh, D. Im, H. Kim, J. Choi, J. Chung, C. Lee, Y. Lee, J.-H. Cho, J. Chun, and J. Oh. 23.2 A 1.1V 1ynm 6.4Gb/s/pin 16Gb DDR5 SDRAM with a Phase-Rotator-Based DLL, High-Speed SerDes and RX/TX Equalization Scheme. In *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 380–382, 2019.

[104] Y.-H. Kim, H.-J. Kim, J. Choi, M.-S. Ahn, D. Lee, S.-H. Cho, D.-Y. Park, Y.-J. Park, M.-S. Jang, Y.-J. Kim, J. Choi, S.-W. Yoon, J.-W. Jung, J.-K. Park, J.-W. Lee, D.-H. Kwon, H.-S. Cha, S.-H. Cho, S.-H. Kim, J. You, K.-H. Kim, D.-H. Kim, B.-C. Kim, Y.-K. Kim, J.-H. Kim, S.-K. Choi, C.-Y. Kim, B.-W. Na, H.-I. Choi, R. Oh, J.-D. Ihm, S.-J. Bae, N. S. Kim, and J.-B. Lee. 25.2 A 16Gb Sub-1V 7.14Gb/s/pin LPDDR5 SDRAM Applying a Mosaic Architecture with a Short-Feedback 1-Tap DFE, an FSS Bus with Low-Level Swing and an Adaptively Controlled Body Biasing in a 3rd-Generation 10nm DRAM. In *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, volume 64, pages 346–348, 2021.

[105] K. Sohn, W.-J. Yun, R. Oh, C.-S. Oh, S.-Y. Seo, M.-S. Park, D.-H. Shin, W.-C. Jung, S.-H. Shin, J.-M. Ryu, H.-S. Yu, J.-H. Jung, H. Lee, S.-Y. Kang, Y.-S. Sohn, J.-H. Choi, Y.-C. Bae, S.-J. Jang, and G. 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*, 52(1):250–260, 2017.

[106] C.-S. Oh, K. C. Chun, Y.-Y. Byun, Y.-K. Kim, S.-Y. Kim, Y. Ryu, J. Park, S. Kim, S. Cha, D. Shin, J. Lee, J.-P. Son, B.-K. Ho, S.-J. Cho, B. Kil, S. Ahn, B. Lim, Y. Park, K. Lee, M.-K. Lee, S. Baek, J. Noh, J.-W. Lee, S. Lee, S. Kim, B. Lim, S.-K. Choi, J.-G. Kim, H.-I. Choi, H.-J. Kwon, J. J. Kong, K. Sohn, N. S. Kim, K.-I. Park, and J.-B. Lee. 22.1

A 1.1V 16GB 640GB/s HBM2E DRAM with a Data-Bus Window-Extension Technique and a Synergetic On-Die ECC Scheme. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 330–332, 2020.

[107] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. 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)*, June 2018.

[108] R. Banner, Y. Nahshan, and D. Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[109] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained Ternary Quantization, 2016.

[110] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized Neural Networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[111] D. Kalamkar, D. Mudigere, N. Mellempudi, D. Das, K. Banerjee, S. Avancha, D. T. Vooturi, N. Jammalamadaka, J. Huang, H. Yuen, J. Yang, J. Park, A. Heinecke, E. Georganas, S. Srinivasan, A. Kundu, M. Smelyanskiy, B. Kaul, and P. Dubey. A Study of BFLOAT16 for Deep Learning Training, 2019.

[112] S. K. Lee, A. Agrawal, J. Silberman, M. Ziegler, M. Kang, S. Venkataramani, N. Cao, B. Fleischer, M. Guillorn, M. Cohen, S. M. Mueller, J. Oh, M. Lutz, J. Jung, S. Koswatta, C. Zhou, V. Zalani, M. Kar, J. Bonanno, R. Casatuta, C.-Y. Chen, J. Choi, H. Haynie, A. Herbert, R. Jain, K.-H. Kim, Y. Li, Z. Ren, S. Rider, M. Schaal, K. Schelm, M. R. Scheuermann, X. Sun, H. Tran, N. Wang, W. Wang, X. Zhang, V. Shah, B. Curran, V. Srinivasan, P.-F. Lu, S. Shukla, K. Gopalakrishnan, and L. Chang. A 7-nm Four-Core Mixed-Precision AI Chip With 26.2-TFLOPS Hybrid-FP8 Training, 104.9-TOPS INT4 Inference, and Workload-Aware Throttling. *IEEE Journal of Solid-State Circuits*, 57(1):182–197, 2022.

[113] X. Sun, J. Choi, C.-Y. Chen, N. Wang, S. Venkataramani, V. V. Srinivasan, X. Cui, W. Zhang, and K. Gopalakrishnan. Hybrid 8-bit floating point (HFP8) training and inference for deep neural networks. *Advances in neural information processing systems*, 32, 2019.

[114] P. Micikevicius, D. Stosic, N. Burgess, M. Cornea, P. Dubey, R. Grisenthwaite, S. Ha, A. Heinecke, P. Judd, J. Kamalu, et al. FP8 Formats for Deep Learning. *arXiv preprint arXiv:2209.05433*, 2022.

[115] A. F. Agarap. Deep Learning using Rectified Linear Units (ReLU). *CoRR*, abs/1803.08375, 2018.

[116] S. Vogel, J. Springer, A. Guntoro, and G. Ascheid. Self-Supervised Quantization of Pre-Trained Neural Networks for Multiplierless Acceleration. In *DATE '19*.

[117] J. Choe. SK hynix' 21 nm DRAM Cell Technology: Comparison of 1st and 2nd generation, 2017.

[118] J. Choe. Samsung 18 nm DRAM cell integration: QPT and higher uniformed capacitor high-k dielectrics, 2017.

[119] C. Weis, A. Mutaal, O. Naji, M. Jung, A. Hansson, and N. Wehn. DRAMSpec: A High-Level DRAM Timing, Power and Area Exploration Tool. *International Journal of Parallel Programming*, 45, 11 2016.

[120] D. U. Lee, K. W. Kim, K. W. Kim, H. Kim, J. Y. Kim, Y. J. Park, J. H. Kim, D. S. Kim, H. B. Park, J. W. Shin, J. H. Cho, K. H. Kwon, M. J. Kim, J. Lee, K. W. Park, B. Chung, and S. 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.

[121] TechInsights. Samsung 18 nm DRAMAnalysis, 2017.

[122] TechInsights. Micron Technology MT43A4G40200NFA-S15 ES:A HMC Gen2 - Memory Functional Analysis. `https://w2.techinsights.com/l/4202/2019-08-28/2hbr19/4202/248106/Sample_Report_MFR_1810_802_Memory_Floorplan_Analysis.pdf`, 2018.

[123] TechInsights. TECHNOLOGY ROADMAP of DRAM for Three Major manufacturers: Samsung, SK-Hynix and Micron, 2014.

[124] J. C. Lee, J. Kim, K. W. Kim, Y. J. Ku, D. S. Kim, C. Jeong, T. S. Yun, H. Kim, H. S. Cho, Y. O. Kim, J. H. Kim, J. H. Kim, S. Oh, H. S. Lee, K. H. Kwon, D. B. Lee, Y. J. Choi, J. Lee, H. G. Kim, J. H. Chun, J. Oh, and S. H. Lee. 18.3 A 1.2V 64Gb 8-channel 256GB/s HBM DRAM with peripheral-base-die architecture and small-swing technique on heavy load interface. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 318–319, 2016.

[125] J. Choe. Micron's 1x DRAMs Examined, 2018.

[126] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal. FreePDK: An Open-Source Variation-Aware Design Kit. In *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, pages 173–174, 2007.

[127] W. Zhao and Y. Cao. Predictive Technology Model for Nano-CMOS Design Exploration. *J. Emerg. Technol. Comput. Syst.*, 3(1):1–es, Apr. 2007.

[128] K. Chandrasekar, C. Weis, B. Akesson, N. Wehn, and K. Goossens. Towards variation-aware system-level power estimation of DRAMs: An empirical approach. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–8, 2013.

[129] J. B. Park. *3D-DATE: A Circuit-Level Three-Dimensional DRAM Area, Timing, and Energy Model.* PhD thesis, 2018.

[130] M. Sung, S. Jang, H. Lee, Y. Ji, J. Kang, T. Jung, T. Ahn, Y. Son, H. Kim, S. Lee, S. Lee, J. Lee, S. Baek, E. Doh, H. Cho, T. Jang, I. Jang, J. Han, K. Ko, Y. Lee, S. Shin, J. Yu, S. Cho, J. Han, D. Kang, J. Kim, J. Lee, K. Ban, S. Yeom, H. Nam, D. Lee, M. Jeong, B. Kwak, J. Park, K. Choi, S. Park, N. Kwak, and S. Hong. Gate-first high-k/metal gate dram technology for low power and high performance products. In *2015 IEEE International Electron Devices Meeting (IEDM)*, pages 26.6.1–26.6.4, 2015.

[131] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *CVPR '15*.

[132] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. 2016.

[133] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.

[134] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *University of Toronto*, 2009.

[135] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, and et. al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vision*, 115(3), Dec. 2015.

[136] Y. Chen, T. Krishna, J. Emer, and V. Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 262–263, 2016.

[137] Y. Chen, T. Yang, J. Emer, and V. Sze. Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, 2019.

[138] A. Biswas and A. P. Chandrakasan. CONV-SRAM: An Energy-Efficient SRAM With In-Memory Dot-Product Computation for Low-Power Convolutional Neural Networks. *IEEE Journal of Solid-State Circuits*, 54(1):217–230, 2019.

[139] L. Wang, W. Ye, C. Dou, X. Si, X. Xu, J. Liu, D. Shang, J. Gao, F. Zhang, Y. Liu, M.-F. Chang, and Q. Liu. Efficient and Robust Nonvolatile Computing-In-Memory Based on Voltage Division in 2T2R RRAM With Input-Dependent Sensing Control. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 68(5):1640–1644, 2021.

[140] T. Soliman, F. Müller, T. Kirchner, T. Hoffmann, H. Ganem, E. Karimov, T. Ali, M. Lederer, C. Sudarshan, T. Kämpfe, A. Guntoro, and N. Wehn. Ultra-Low Power Flexible Precision FeFET Based Analog In-Memory Computing. In *2020 IEEE International Electron Devices Meeting (IEDM)*, pages 29.2.1–29.2.4, 2020.

[141] R. J. Baker. *CMOS Circuit Design, Layout, and Simulation*. Wiley-IEEE Press, 4th edition, 2010.

[142] C. De la Parra, A. Guntoro, and A. Kumar. ProxSim: GPU-based Simulation Framework for Cross-Layer Approximate DNN Optimization. In *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1193–1198, 2020.

[143] T. Takahashi, T. Sekiguchi, R. Takemura, S. Narui, H. Fujisawa, S. Miyatake, M. Morino, K. Arai, S. Yamada, S. Shukuri, M. Nakamura, Y. Tadaki, K. Kajigaya, K. Kimura, and K. Itoh. A multigigabit DRAM technology with 6F/sup 2/ open-bitline cell, distributed overdriven sensing, and stacked-flash fuse. *IEEE Journal of Solid-State Circuits*, 36(11):1721–1727, 2001.

[144] M. Jung, I. Heinrich, M. Natale, D. M. Mathew, C. Weis, S. Krumke, and N. Wehn. ConGen: An Application Specific DRAM Memory Controller Generator. In *Proceedings of the Second International Symposium on Memory Systems*, MEMSYS '16, pages 257–267, New York, NY, USA, 2016. ACM.

[145] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens. Memory Access Scheduling. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, ISCA '00, pages 128–138, New York, NY, USA, 2000. ACM.

[146] B. Keeth, R. J. Baker, B. Johnson, and F. Lin. *DRAM Circuit Design: Fundamental and High-Speed Topics*. Wiley-IEEE Press, 2nd edition, 2007.

[147] A. Kotabe, Y. Yanagawa, R. Takemura, T. Sekiguchi, and K. Itoh. Asymmetric cross-coupled sense amplifier for small-sized 0.5-v gigabit-dram arrays. In *2010 IEEE Asian Solid-State Circuits Conference*, pages 1–4, 2010.

[148] H. Luo, T. Shahroodi, H. Hassan, M. Patel, A. G. Yağlıkçı, L. Orosa, J. Park, and O. Mutlu. Clr-dram: A low-cost dram architecture enabling dynamic capacity-latency trade-off. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 666–679, 2020.

[149] M. Jung, C. Weis, and N. Wehn. DRAMSys: A flexible DRAM Subsystem Design Space Exploration Framework. *IPSJ Transactions on System LSI Design Methodology (T-SLDM)*, August 2015.

[150] L. Steiner, M. Jung, F. S. Prado, K. Bykov, and N. Wehn. Dramsys4.0: A fast and cycle-accurate systemc/tlm-based dram simulator. In *International Conference / Workshop on Embedded Computer Systems: Architectures, Modeling and Simulation*, 2020.

[151] K. Chandrasekar, C. Weis, Y. Li, B. Akesson, O. Naji, M. Jung, N. Wehn, and K. Goossens. DRAMPower: Open-source DRAM power & energy estimation tool. `http://www.drampower.info`, Last Access 15.08.2019.

[152] Micron Technology, Inc. DDR4 SDRAM (MT40A512M16).

[153] A. C. I. Malossi, M. Schaffner, A. Molnos, L. Gammaitoni, G. Tagliavini, A. Emerson, A. Tomás, D. S. Nikolopoulos, E. Flamand, and N. Wehn. The transprecision computing paradigm: Concept, design, and applications. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1105–1110, March 2018.

[154] D. Rossi, A. Pullini, I. Loi, M. Gautschi, F. K. Gürkaynak, A. Teman, J. Constantin, A. Burg, I. Miro-Panades, E. Beignè, F. Clermidy, P. Flatresse, and L. Benini. Energy-Efficient Near-Threshold Parallel Computing: The PULPv2 Cluster. *IEEE Micro*, 37(5):20–31, Sep. 2017.

[155] A. Kurth, W. Roenninger, O. Castaneda, C. Weis, J. Lappas, C. Sudarshan, and B. Muheim. Urania. `http://asic.ethz.ch/2019/Urania.html`, 2019.

[156] C. Weis, A. Mutaal, O. Naji, M. Jung, A. Hansson, and N. Wehn. DRAMSpec: A High-Level DRAM Timing, Power and Area Exploration Tool. *International Journal of Parallel Programming*, 45(6):1566–1591, Dec 2017.

[157] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu. RAIDR: Retention-Aware Intelligent DRAM Refresh. In *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, pages 1–12, Washington, DC, USA, 2012. IEEE Computer Society.

[158] I. Bhati, Z. Chishti, S.-L. Lu, and B. Jacob. Flexible auto-refresh: enabling scalable and energy-efficient DRAM refresh reductions. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, pages 235–246. ACM, 2015.

[159] M. Jung, D. Mathew, C. Weis, and N. Wehn. Approximate Computing with Partially Unreliable Dynamic Random Access Memory - Approximate DRAM. In *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, pages 100:1–100:4, New York, NY, USA, 2016. ACM.

[160] J. Lucas, M. Alvarez-Mesa, M. Andersch, and B. Juurlink. Sparkk: Quality-Scalable Approximate Storage in DRAM. In *The Memory Forum*, June 2014.

[161] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn. Flikker: Saving DRAM Refresh-power Through Critical Data Partitioning. *SIGPLAN Not.*, 46(3):213–224, Mar. 2011.

[162] M. Jung, E. Zulian, D. Mathew, M. Herrmann, C. Brugger, C. Weis, and N. Wehn. Omitting Refresh - A Case Study for Commodity and Wide I/O DRAMs. In *1st International Symposium on Memory Systems (MEMSYS 2015)*, Washington, DC, USA, October 2015.

[163] S. M. A. H. Jafri, H. Hassan, A. Hemani, and O. Mutlu. Refresh Triggered Computation: Improving the Energy Efficiency of Convolutional Neural Network Accelerators. *ACM Trans. Archit. Code Optim.*, 18(1), dec 2021.

[164] K. K.-W. Chang, D. Lee, Z. Chishti, A. R. Alameldeen, C. Wilkerson, Y. Kim, and O. Mutlu. Improving DRAM performance by parallelizing refreshes with accesses. In *High Perfor-*

mance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on, pages 356–367. IEEE, 2014.

[165] D. M. Mathew. *Advanced Heterogeneous Memory Subsystemsfor Energy-Constrained Computing*. PhD thesis, University of Kaiserslautern, 2020.

[166] D. M. Mathew, F. Zulian, M. Jung, K. Kraft, C. Weis, B. Jacob, and N. Wehn. Using Run-Time Reverse-Engineering to Optimize DRAM Refresh. In *International Symposium on Memory Systems (MEMSYS17)*, 2017.

[167] J. Wang, X. Dong, and Y. Xie. ProactiveDRAM: A DRAM-initiated retention management scheme. In *32nd IEEE International Conference on Computer Design (ICCD)*, pages 22–27, Oct 2014.

[168] S. Baek, S. Cho, and R. Melhem. Refresh now and then. *IEEE Transactions on Computers*, 63(12):3114–3126, 2014.

[169] P. J. Nair, C.-C. Chou, and M. K. Qureshi. Refresh Pausing in DRAM Memory Systems. *ACM Trans. Archit. Code Optim.*, 11(1):10:1–10:26, Feb. 2014.

[170] A. G. Yağlikçi, A. Olgun, M. Patel, H. Luo, H. Hassan, L. Orosa, O. Ergin, and O. Mutlu. HiRA: Hidden Row Activation for Reducing Refresh Latency of Off-the-Shelf DRAM Chips. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 815–834, 2022.

[171] M. Jung. *System-Level Modeling, Analysis and Optimization of DRAM Memories and Controller Architectures*. PhD thesis, University of Kaiserslautern, 2017.

[172] Cadence Inc. Cadence Denali DDR Memory IP. `http://ip.cadence.com/ipportfolio/ip-portfolio-overview/memory-ip/ddr-lpddr`, October 2014, last access 18.02.2015.

[173] Synopsys, Inc. DesignWare DDR IP. http://www.synopsys.com/IP/InterfaceIP/DDRn/Pages/, 2015, Last Access: 18.02.2015.

List of Figures

1.1 DNN parameter size increasing over the years. Source [8] 2

1.2 Roofline model for Google TPUv4i. The lower point and higher points show the results of TPUv4i with 16 MB on-chip memory and 144 MB on-chip memory respectively. The operation intensity (i.e. num of operations per memory access) for lower points is with reference to HBM2 and for higher points is with reference to large on-chip memory. The solid vertical line and dashed vertical line depict the ridge point for the HBM2 and large on-chip scenarios, respectively. Source [13] 3

1.3 DRAM energy contribution in various DNN accelerators. Source [16–19] . 5

1.4 DRAM hierarchical structure. The DRAM device height-to-width ratio, number of banks and bank placement is based on a real 10 nm, 2nd gen, 12Gb commodity LPDDR5 chip [41]. 7

1.5 Scaling of DRAM cell size over the years and future predictions. 3x nm, 2x nm, 1x nm, and 1z nm process node sizes are approximately equal to 32 nm, 28 nm, 18 nm and 12 nm, respectively. Source [42] 8

1.6 Thesis outline. 17

2.1 Different DRAM organization in computation systems (Image courtesy [96]). 20

2.2 DRAM hierarchical architecture. 21

2.3 Detailed view of the H-Block. 21

2.4 Circuit diagram for column selection from PSAs to MBLs. For ease of representation, all PSA outputs are shown on the same side of the Sub-Array, but in the actual design, the PSAs and CSS are interleaved on both sides of the Sub-Array. 22

2.5 Sub-Array architecture and cross-section. a) Simple illustration of Sub-Array (SA), b) $6F^2$ Sub-Array architecture and slanted active area, c) Sub-Array cross-section showing buried gate, stacked cell capacitors, and shared drain terminal. Source [45–48] 23

2.6 DRAM Primary Sense Amplifier (PSA) region circuit design and example floorplan. a) Circuit design [97], b) Example floorplan. In this figure Sub-Array is abbreviated as SA. 24
2.7 DRAM data sensing waveform [97] (Image courtesy [96]). 25
2.8 Computation system top-level overview. 27
2.9 State machine of a single DRAM bank [51] (Image courtesy [96]). 27
2.10 Example waveform for DRAM timing delays within a bank. 28
2.11 Example waveform for DRAM timing delays between different banks. . . . 30

3.1 Three main dimensions of DRAM-PIM design space. 34
3.2 Ambit implementation. a,b) Basic operation principle of Ambit, c) Dual Contact Cell (DCC) proposed by Ambit to implement NOT operation, d) Comparison of commodity DRAM Sub-Array vs. modifications required to Sub-Array for connecting LBL* to the DCC. 35
3.3 PSA floorplan of conventional vs. with computation unit for PIM. a) Conventional PSA region, b/c) example computation unit integration in the PSA region, where metal tracks have to be reserved in the computation unit for LBL and LBL* to be connected to respective PSA. All three floorplan figures are rotated by 90°in a clockwise direction due to space considerations. 36
3.4 High energy consumption for transferring data to and from IO region. a) Energy breakdown of read operation in 20 nm standard HBM2 device [20], b) Hynix-AiM distributing input feature map data to all banks from a global buffer in IO region: High energy consumption, c) Hynix-AIM fetching both input feature map data and weight from the bank: throughput reduction by half [37]. 41
3.5 Data formats employed in state-of-the-art standard DNN accelerators for inference and training and their accuracy degradation compared to FP32. Accuracy degradation values are extracted from the sources [107–110] for inference and from [111, 112] for training. 41

4.1 Architecture 1: PPU architecture. For image clarity the connection of LBL* to PSA is not shown. In this figure Sub-Array is abbreviated as SA. . 50
4.2 Architecture 1: Example Shifter Circuit for 2-bit Input. 51
4.3 Architecture 1: Conventional CSS block (top) compared with CSS-1 block (bottom) of this architecture. 51
4.4 Architecture 1: PIM Bank Architecture. 52
4.5 Architecture 1: PIM Bank operation timing diagram. 53

4.6 Architecture 1: CNN inference accuracy compression of all aforementioned quantization. Courtesy: Taha Soliman 57
4.7 Architecture 1: PPU's adder layout @65 nm. 58
4.8 Architecture 1: Performance (GOPS) Comparison. 60
4.9 Architecture 1: Energy Efficiency (GOPS/W). 60

5.1 Basic implementation of bit-decomposed WCS MAC in emerging memories. 67
5.2 Architecture 2: Basic implementation of bit-decomposed WCS MAC in this DRAM-PIM architecture. 68
5.3 Architecture 2: Sub-Array-level Architecture with analog PPU. The yellow blocks show the new circuits. For image clarity, the connection of LBL* to PSA is not shown in left figure. In this figure Sub-Array is abbreviated as SA. 69
5.4 Architecture 2: (Top) CSS connection to local data lines. (Bottom) Hardwired connection between "Circuit B" and PIM_I+ lines. Shows the similarity between the CSS connection and "Circuit B" connection. For the ease of representation, all PSA output are shown on same side, but in the actual design the "Circuit B" and CSS are interleaved on both side of Sub-Arrays. An example input data and weights available on PSA and CSL, respectively, is shown is green color. 71
5.5 Architecture 2: Bank-level mixed-signal PIM Architecture. In this figure Sub-Array is abbreviated as SA. 72
5.6 Architecture 2: Conventional activation (top) vs Vertical activation for P=2 and Q=3 (bottom). In this figure Sub-Array is abbreviated as SA. 73
5.7 Architecture 2: Comparing the example PSA region floorplan of conventional DRAM with DRAM-PIM. 76
5.8 Architecture 2: Example mapping of $1 \times 1 \times 128$ kernel to a 1024 cell wide Sub-Array. 78
5.9 Architecture 2: Impact of threshold (Vth) variation on a single current mirror output - 500 Monte-Carlo simulations. 79
5.10 Architecture 2: Impact of threshold (Vth) variation on PIM_I+ line current accumulation results - 10000 simulations of convolution operation using WCS-based MAC. The min and max value corresponds to 5th and 95th percentile, respectively. 80
5.11 Architecture 2: CNN inference accuracy (in %) comparison for various networks. INT8-5Bit ADC and INT8-8Bit ADC corresponds to the accuracy of the proposed PIM device with 5-bit and 8-bit ADC, respectively. INT8-5Bit ADC and INT8-8Bit accuracy results consider 30 mV threshold variation. 82

5.12 Architecture 2: Performance (in GOPS) and power consumption (in mW) of the proposed mixed-signal DRAM-PIM architecture for various CNNs. . 87

6.1 Comparing DRAM BitLine array architectures from PIM perspective: a) Conventional Folded BitLine (FBL) b) Conventional Open BitLine (OBL) c) Folded BitLine with new compute logic d) Open BitLine with new compute logic e) Quasi BitLine (QOBL) with new compute logic. QOBL eliminating the PSA sharing and enabling the integration of large computation units in the PSA region without modifying the Sub-Array design. 93

6.2 Device-level architecture comparison. a) State-of-the-art GDDR6 DRAM-PIM device with OBL array type bank. b) Commodity LPDDR4 DRAM device with OBL array type bank. c) Proposed DRAM-PIM with two types of banks i.e. conventional OBL based high capacity banks and new QOBL based PIM banks. 95

6.3 Samsung's FIMDRAM operation flow [34]. The data access request is stalled until the PIM mode is changed to normal mode. 98

6.4 Unified Protocol for External Control of PIM. PI is the new generic PIM Instruction command, whose 2 burst data indicate the actual computation commands. In this example, Vendor1 has only one type of PIM command (Green color) and Vendor2 has two types of PIM commands (Brown color). This protocol interleaves the PIM compute operations and data access operations without stalling the data access requests for a long time. Furthermore, Vendor2 sends the new PIM commands to the DRAM-PIM by reusing the PI and does not require any modification to the CMD interface with every new generation. 98

7.1 Dual-Sense-Amplifier (DSA) Architecture. In this figure Sub-Array is abbreviated as SA. 103

7.2 DSA latency reduction as compared to commodity DRAM. 106

7.3 Average Access Response Latency comparison of DSA, SALP and commodity 8Gb devices for quad-core system. 108

7.4 Bandwidth Utilization comparison of DSA, SALP and commodity 8Gb devices for quad-core system. 109

7.5 Energy consumption comparison of DSA, SALP and commodity 8Gb devices for quad-core system. 110

7.6 Power consumption comparison of DSA, SALP and commodity 8Gb devices for quad-core system. 111

8.1 Computation system top-level overview. 113

8.2 Transprecision DDR3 memory controller architecture 116
8.3 ASAM Architecture 118
8.4 DDR3 memory controller post-layout and die photo 119

List of Tables

1.1 Comparison of computation units energy vs memory data access energy. Source [13] 4

2.1 Timing delays within a bank. 29

2.2 Timing delays for different banks. 31

3.1 Bank height comparison for various DRAM types. 39

4.1 Architecture 1: Area results. 58

4.2 Architecture 1: Latency and Energy of various operations for the core clock frequency of 250 MHz. 59

4.3 Comparison of DRAM based PIM architectures. 62

5.1 Architecture 2: Example bit-decomposed MAC operation for $1 \times 1 \times 4$ kernel, $W_p = 3$, $I_p = 3$, $\{w_{000}, w_{001}, w_{002}, w_{003}\}$ = {2, 1, -3, -1} and $\{x_{000}, x_{001}, x_{002}, x_{003}\}$ = $\{0, -2, 1, 3\}$. 66

5.2 Architecture 2: ADC output absolute error - 10000 simulations of convolution operation. Median, 95th percentile and max error are shown. *The error values are in absolute values and not in bits*. 81

5.3 Architecture 2: Area evaluation results of all the building blocks of Standard DRAM with PIM and the mixed-signal based DRAM-PIM 83

5.4 Architecture 2: Timing and Latency of the new PIM commands. A few timings in JEDEC DRAM specification are presented in terms of interface clock cycles (Ck) that is equal to $4\times$ core clock cycle of the proposed architecture. Assumption: CWL = 12Ck, CL=14Ck, BL=8. 84

5.5 Architecture 2: Energy consumption of various operations for the core clock frequency of 250 MHz. 85

5.6 Comparison of DRAM based PIM architectures. 89

7.1 Additional Timing Constraints for DSA. All new timings are applicable for the commands sent to the same bank 105
7.2 Area evaluation results 108

8.1 Area Distribution of the Controller and PHY 119
8.2 Power Distribution of the Controller and PHY 119

List of Algorithms

1 CNN Optimization with one-hot quantization 48

Lebenslauf

Persönliche Daten

Name:	Chirag Sudarshan
Geburtsort:	Bangalore, Indien
Familienstand:	Verheiratet

Ausbildung

Schule:	Assumption English Primary School, Bangalore 1996–2003 St. Ann's High School, Bangalore 2003–2006 PVP Polytechnic, Bangalore 2006–2009
Hochschulstudium:	Grund- und Hauptstudium: Elektronik und Kommunikation Ramaiah Institute Of Technology, Bangalore 08/2009–05/2012 (Bachelor im Mai 2012)
	Grund- und Hauptstudium: Elektronik und Informationstechnik Vertiefungsrichtung: Eingebettete Systeme Technische Universität Kaiserslautern 10/2014–02/2017 (Master of Science (M.Sc.) im Februar 2017)

Beruflicher Werdegang

6/2012 - 8/2014	Software-Ingenieur Robert Bosch Engineering and Business Solutions Limited Bangalore
5/2017 - 5/2023	Wissenschaftlicher Mitarbeiter Arbeitsgruppe Entwurf Mikroelektronischer Systeme Fachbereich Elektrotechnik und Informationstechnik Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau