SRAM-Based Digital Processing-In-Memory Architectures for Resource-Efficient Edge AI Acceleration

MS (Research) Thesis

By

AKASH NITIN SANKHE

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

MAY 2025

SRAM-Based Digital Processing-In-Memory Architectures for Resource-Efficient Edge AI Acceleration

A Thesis

Submitted in fulfillment of the

requirements for the award of the degree

of

Master of Science (Research)

by

AKASH NITIN SANKHE

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

MAY 2025

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **SRAM-Based Digital Processing-In-Memory Architectures for Resource-Efficient Edge AI Acceleration** in the fulfillment of the requirements for the award of the degree of **Master of Science (Research)** and submitted in the **Department of Electrical Engineering, Indian Institute of Technology Indore,** is an authentic record of my own work carried out during the time period from July 2023 to May 2025 under the supervision of Dr. Santosh Kumar Vishvakarma, Professor, Indian Institute of Technology Indore, Indore, India.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

18/05/2025

Signature of the Student with Date

(Akash Nitin Sankhe)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

19/05/2025

Signature of the Supervisor of MS-R Thesis with Date

(Prof. Santosh Kumar Vishvakarma)

Akash Nitin Sankhe has successfully given his MS-Research Oral Examination held on **07/07/2025**

July 7, 2025

Signature of Chairperson (OEB) with date

Prof. Anirban Sengupta

07/07/2025

Signature of Thesis Supervisor with date

07-07-2025

Signature of Convener, DPGC with date

7 July 2025

Signature of Head of Discipline with date

ACKNOWLEDGEMENTS

I am profoundly grateful to my MS Research thesis supervisor and mentor, Prof. Santosh Kumar Vishvakarma, for his unwavering support and encouragement throughout my academic journey. His belief in my abilities and his invaluable guidance have been constant sources of motivation, driving me to push my boundaries and exceed my expectations. I am deeply indebted to him for granting me the freedom to explore my research interests and allowing my innovative ideas to flourish.

I would also like to express my sincere appreciation to Prof. Sharad Kumar Singh, the Coordinator of my thesis evaluation committee, and all evaluation committee members. Their thoughtful evaluations and insightful questions have played a crucial role in broadening my research perspective.

I am immensely thankful to my family for their unyielding support throughout my master's studies. Their confidence in me and their love, sacrifices, and constant encouragement have been instrumental in my academic success. I will forever be grateful for their guidance, and their faith in me will continue to propel me towards greater accomplishments.

My heartfelt thanks also go to the Nanoscale Devices and VLSI Circuit System Design Lab (NSDCS) research group, particularly Dr. Gopal Raut, Dr. Narendra Dhakad, Mr. Radheshyam Sharma, and Mr. Mukul Lokhande, for their continuous guidance and support. I would also like to thank Dr. Adam Teman for his valuable input and guidance. I also appreciate the camaraderie and encouragement of my friends Mr. Akash Ahirwar, Mr. Hansraj Meena, Mr. Aman Chandrakar, Mr. Anand Kachale, Mr. Kishan Tripathi and Ms. Sneha Jha and labmates, Mrs. Neha Maheshwari, Mr. Sonu Kumar, Mr. Shashank Singh Rawat, Mr. Vikash Vishwakarma, Mr. Ankit Tenwar, Ms. Komal Gupta, and Mr. Sagar Patel, whose friendship made my time at the institute truly memorable...

Akash Nitin Sankhe

This Thesis is Dedicated

to

My Parents, My Brother & Sister-in-law,
My Grandparents and the Almighty God

ABSTRACT

The rapid expansion of AI applications has intensified the focus on energy-efficient and high-throughput compute-in-memory (CIM) operations for resource-constrained Edge-AI platforms. Recently, SRAM-embedded CIM hardware has emerged as a promising solution to mitigate von Neumann bottlenecks and has shown noteworthy improvements in energy efficiency and throughput for matrix-vector multiplication, a significant portion of neural networks. While PVT variations significantly impact traditional analog/mixed-signal (AMS) macros, the DCIM macros are more robust.

2D Interleaved Adder Tree-based DCIM macro that incorporates an 8-transistor SRAM bitcell capable of performing 1-bit multiplications and addressing the bit-flip issue from simultaneous activation of multiple array rows. A 2D interleaved adder tree using a novel 7T-based ripple carry adder (RCA) significantly reduces area overhead. The proposed 16Kb macro computes 64 parallel products per clock cycle and achieves 2× higher energy efficiency over recent SoTA works at 65nm CMOS. The macro is validated at 250MHz and demonstrates classification accuracy of 98.7%, 98.8% for 1A4W precision, and 99.1%, 97.8% for 4A4W precision on LeNet-5 using MNIST and A-Z datasets respectively.

RAPID-CIM, a novel reconfigurable digital CIM with M8T bitcell and an area-efficient hierarchical adder tree, to address performance drawbacks associated with conventional SoTA works. It achieves up to 16.7× improvement in energy efficiency and 8.6× enhancement in compute density compared to SoTA designs at CMOS 65nm. Furthermore, RAPID-CIM enables scalable bit-precision and sparsity-aware operations, reducing up to 3.2× operation cycles and memory bank usage by 30% while maintaining application accuracy within 97.8% Quality of Results (QoR).

DARE, a novel DPIM approach featuring an area-efficient A7T bitcell, composed of a 5T SRAM cell capable of subthreshold (ST) operation and a 2T bit-wise multiplication. The ST multiplications significantly reduce power consumption, enhancing energy efficiency up to 2.85×. A delay and area-optimized interleaved adder tree, comprised of power-gated Carry Skip Adder (CSkA) using FA-7T and FA-14T,

is integrated to rapidly accumulate partial products, leading to 3.2× improvement in compute density.

Thus, the proposed solutions, RAPID-CIM, DARE, and the 2D Interleaved adder tree-based DCIM macro, offer robust and scalable digital compute-in-memory solutions with high energy efficiency, compute density, and application accuracy, positioning them as well-versed optimizations for next-generation Edge-AI inference accelerators.

Contents

List of Figures

List of Tables

List of Abbrevations

DL	-	Deep Learning
ML	-	Machine Learning
AI	-	Artificial Intelligence
DNN	-	Deep Neural Network
SRAM	-	Static Random Access Memory
PCM	-	Phase-Change Memory
ReRAM	-	Resistive Random Access Memory
STT-RAM	-	Spin-Transfer Torque RAM
FeRAM	-	Ferroelectric RAM
IMC	-	In-Memory Computing
RCA	-	Ripple carry adder
CSkA	-	Carry skip adder
NVM	-	Non-Volatile Memory
PG	-	Pass Gate
PUN	-	Pull-Up Network
PDN	-	Pull-Down Network
SoC	-	System-on-Chip
ASIC	-	Application Specific Integrated Circuit
STA	-	Static Time Analysis
IC	-	Integrated Circuits

Chapter 1

Introduction

1.1 Introduction

Artificial Intelligence (AI) has rapidly become an integral part of modern digital systems, driving data-driven decision-making in applications such as voice assistants, facial recognition, autonomous vehicles, and personalized recommendations. These applications rely heavily on deep learning algorithms, which involve extensive multiply-and-accumulate (MAC) operations for tasks like image recognition, classification, and language processing. As datasets continue to grow, the computational complexity of these applications also increases, posing significant challenges to traditional von Neumann architectures. These architectures suffer from limited data bus bandwidth, memory latency, and instruction fetching bottlenecks, collectively referred to as the von Neumann bottleneck [11].

To meet the computational demands of AI workloads, especially on edge devices, there is a growing need for hardware accelerators that offer high throughput, energy efficiency, and compute density. Edge-AI devices, in particular, require quick event-triggered responses, low latency, and optimized performance per watt to support real-time inference. Runtime reconfigurable architectures that allow layer reuse and precision-scalable operations have emerged as promising solutions. Models like VGG-16 require 15.5 B MAC operations and 138M weights for processing a 224×224 RGB image, while mobile-BERT demands 4.5B MAC operations. Quantization techniques, such as 4-bit precision, have shown the ability to maintain acceptable inference accuracy while significantly reducing computational load [12].

Compute-In-Memory (CIM) has emerged as a promising alternative to mitigate the memory bottleneck and reduce the energy cost associated with data movement between memory and processing units. By performing computations within memory arrays, CIM architectures bypass the limitations of traditional memory hierarchies. CIM can be broadly categorized into analog and digital implementations. Analog CIM offers higher energy efficiency and throughput but suffers from significant drawbacks such as non-linearity, process variations, and high overheads from analog-to-digital converters (ADC). Digital CIM (DCIM), on the other hand, avoids data-conversion

errors, offers better scalability with technology nodes, and supports bit-precision reconfigurability [13, 14].

Static Random Access Memory (SRAM) is the preferred choice for implementing CIM in edge-AI Systems-on-Chip (SoC) due to its fast access time, high endurance, and no requirement for periodic refresh. In a typical DCIM macro, the architecture comprises an SRAM array for storage and computation, an address decoder, controller, peripheral circuitry, an adder tree, and an accumulator. The SRAM array stores weights and performs MAC operations with input feature maps in a dual-purpose manner. However, implementing digital MAC operations in SRAM arrays introduces challenges such as bit-flips caused by simultaneous activation of wordlines, requiring larger 8T or 10T bitcells with decoupled read ports. Moreover, adder trees used in the accumulation stage significantly impact power and area, consuming up to 65% of power and 45% of area in some designs [15].

Prior work has addressed these issues through innovations at the algorithmic, architectural, and circuit levels. Techniques such as sparsity-aware computing and pruning have shown up to 50% reduction in unnecessary computations. Circuit-level innovations include using AND-based MACs for reconfigurable precision and better accuracy compared to XNOR-MACs, and novel adder architectures to reduce area and power overhead. While analog CIM avoids threshold losses under constrained input voltages, digital CIM macros must address threshold loss and voltage drop issues that degrade inference accuracy.

Conventional adder architectures such as Ripple Carry Adders (RCAs), Carry Look-Ahead Adders (CLA), and Brent Kung Adders (BKA) offer various area-delay tradeoffs. Approximate adder designs like ACA, QuAd, ESA, and APEx aim to reduce power and area by modifying the carry chain or logic gates but are unsuitable for low-bit precision quantized AI workloads due to accuracy degradation. Interleaved adder tree structures with optimized full adders (e.g., 10T, 12T, or 28T) have been explored to reduce area and delay, yet they often fail to address intra-stage voltage drops and threshold loss from the SRAM bitcell [16].

1.2 Background and Motivation

The shift towards edge-AI and the decline of Moore's Law highlight the urgent need for alternative memory-compute architectures. Moore's Law, which predicted the doubling of transistors every two years, is challenged by quantum tunneling, leakage currents, and heat dissipation issues at advanced technology nodes. As transistor scaling slows, architectural innovations like CIM become critical.

A typical DCIM macro consists of the following components:

- **SRAM Memory Array:** Stores weights and inputs, often in a dual-purpose configuration.
- **Address Decoder and Controller:** Facilitates memory access and compute control.
- **Peripheral Circuitry:** Enables data path management and input application.
- **Adder Tree and Accumulator:** Aggregates partial products from each row of the array.

Several SoTA bitcell and adder tree designs have attempted to address the limitations of traditional DCIM architectures. For example, the 10T CONV-SRAM bitcell enhances dynamic range by decoupling write and MAC operations and utilizing charge-based accumulation. However, the extra transistors and analog components compromise storage density and energy efficiency. D6CIM enhances storage density using fixed-point arithmetic and time-shared hardware but suffers from throughput loss [17].

Adder trees built using RCA structures with 28T full adders dominate power and area consumption. Works like Sankhe et al. introduced interleaved adder tree structures using 10T and 28T full adders to manage voltage drops across stages. However, these designs often overlook threshold losses within same-stage RCA elements and from pass-transistor logic multiplication [2].

Our proposed solution introduces several key innovations:

- Wordline-independent 8T CIM bitcell for efficient AI computation.
- Novel FA14T and HA11T for area-efficient Ripple Carry Adder (RCA) architecture.
- A low-voltage 5T SRAM bitcell to minimize access power.
- A 2T bit-wise multiplier to reduce area overhead.
- A true-output power-gated Carry Skip Adder (CSkA) architecture built using FA-7T and FA-14T blocks.
- A NOR-MUX-based prefix carry scheme to optimize delay and critical carry path.
- Exploitation of sparsity to reduce computational workload.

Our approach addresses the limitations of both analog and digital CIM, achieving runtime reconfigurability, precision scalability, and enhanced resource efficiency. Tables 1.1 and 1.2 provide a comparative summary of existing bitcell and adder tree designs, focusing on transistor count, output precision, and compute accuracy. By overcoming prior limitations, our design presents a comprehensive solution for next-generation edge-AI acceleration.

Table 1.1: Overview of PIM bit cells from recent SoTA works

Parameters	Technology (nm)	No. of Transistors	Bitcell Area (um2)	Bitwise Mult.	Mult. Domain	Mult. Type	Accuracy Support	Layout Density
JETCAS'22 [**17**]	65	8	2.78	Yes	Analog	AND	Medium	Medium
TCAS-II'23 [**18**]	28	10	0.623	Yes	Analog	NA	NA	Medium
TNANO'23 [**19**]	65	10	4.5	Yes	Analog	XNOR	High	Low
DATE'24 [**20**]	65	8	2.52	Yes	Digital	AND	NA	High
TCAS'24 [**21**]	55	8	4.28	Yes	Digital	MUX	High	Low
ISQED'25 [**1**]	65	8	4.1	Yes	Digital	AND	Medium	High
JSSC'21 [**3**]	65	NA	10.528	Yes	Digital	XNOR	High	High
JSSC'19 [**22**]	65	10	NA	Yes	Analog	NA	Medium	Low
RAPID-CIM	65	8	4.21	Yes	Digital	AND	High	High
DARE-PIM	65	7	2.23	Yes	Digital	AND	High	High

Table 1.2: Comparative evaluation of SoTA adder tree works.

	Tech. (nm)	**Adder Tree**	**Reconfigurability**	**# Transistors/FA**	**Output Type**
JSSC'21 [**3**]	65	Conventional	No	28T	Non-Inv.
JSSC'24 [**4**]	28	Hybrid†	Partial†	12T	Inverted
ISCAS'23 [**7**]	40	Accurate RCAs	No	Interleaved (28T + 10T)	Non-Inv.
TCAS'24 [**8**]	55	RCA + MUX	No	26T	Inverted
Ours	65	Optimised RCAs	Yes	14T FA + 11T HA	Non-Inv.

Note: Here, the hybrid adder† tree approach refers to partial reconfigurability† between approximate compressors and accurate RCAs.

1.3 Research Objectives

The primary objective of this research is to investigate and develop a novel Compute in Memory architecture based on SRAM technology for Edge AI applications. The specific research objectives include:

- Designing a SRAM-based CIM architecture tailored for CAM operations, with a focus on improving speed, energy efficiency, and area efficient compared to conventional SoTA designs.

- Developing efficient circuit designs and performance tradeoff to boost computation within SRAM-based memory cells, leveraging the AI workload characteristics (sparsity).

- Implementing a architecture SRAM-based CIM system and conducting comprehensive performance evaluations to validate the effectiveness of the proposed architecture in real-world scenarios.

- Exploring the potential applications and practical implications of SRAM-based CIM for various computing tasks, such as neural networks, and AI inference.

1.4 Organization of Thesis

This thesis is organized into several chapters, each focusing on different aspects of the research conducted on Static RAM (SRAM) based Processing-in-Memory architectures for resource efficient edge AI acceleration. The chapters are structured as follows:

Chapter 1: This chapter provides the background and motivation for the research, highlighting the increasing demands of data-intensive applications and the need for high-speed, energy-efficient solutions. The literature review offers a comprehensive overview of Compute-In-Memory (CIM) or Processing-In-Memory (PIM), including existing CIM implementations and various types of CIM architectures. This chapter sets the stage for the novel contributions of this research by discussing the limitations of current technologies and the potential improvements offered by SRAM-based solutions. It introduces processing-in-memory architectures required for data-intensive applications. The chapter concludes with the research objectives and a brief overview of the thesis organization.

Chapter 2: This chapter details the design and implementation of the Area-optimized 2D Interleaved Adder Tree design to enhance the performance of DCIM processing. The work covers the structure of the novel FA-7T, its operation, incorporation into two dimensional interleaving adder tree structure including an analysis of the its impacts and a performance evaluation of the sparse DCIM array.

Chapter 3: This chapter details the design and implementation of Reconfigurable Adder Tree design to incentivize the digital CIM operations. The work covers the structure of the novel HA-11T and FA14T, its operation, incorporation into adder tree structure, including an analysis of the its impacts and a performance evaluation of the digital DCIM macro.

Chapter 4: This chapter details the design and implementation of low-voltage A7T design to enhance the performance of DCIM processing and power gated novel CsKA Adder Tree. The work covers the structure of the novel 2-bit CsKA with FA7T and FA14T, its operation, incorporation into adder tree structure including

an performance evaluation for the edge PIM macro.

Chapter 5: The final chapter summarizes the key findings and contributions of the research. It discusses the implications of the novel RAPID-CIM , DARE and 2D Interleaved adder based DCIM macro design for future SRAM-enabled CIM technology. Additionally, it outlines potential directions for future research to further enhance the performance and applicability.

Chapter 2

Area-optimized 2D Interleaved Adder Tree Design for Sparse DCIM Edge Processing

The proliferation of edge-AI applications demands energy-efficient and compact compute solutions capable of performing large-scale neural network inference within tight area and power constraints. Compute-in-memory (CIM) architectures have emerged as a promising alternative to conventional von Neumann systems by reducing the data movement overhead and improving compute throughput. However, existing digital CIM designs still have limitations, such as excessive transistor count in peripheral logic, low signal margins, and unreliable bitcell operations under parallel activation [15, 19]. This paper proposes a novel digital CIM (DCIM) macro that integrates a multiplication-capable 8T SRAM bitcell and a compact 2D interleaved adder tree tailored for low-power inference on pruned deep neural networks.

Traditional MAC-intensive neural networks executed at the edge often face challenges related to memory bottlenecks, compute inefficiency, and area overhead in digital CIM implementations. State-of-the-art CIM designs still struggle with energy inefficiency due to voltage drops, static power dissipation, and limited scalability concerning precision and sparsity. Motivated by these challenges, this work aims to enhance compute density and energy efficiency by rethinking bitcell functionality and adder tree organization. With a novel 8T bitcell for bitwise multiplication and an ultra-compact 7T full adder with a 2D interleaved structure, we significantly reduce transistor count while maintaining high performance—making the proposed macro a compelling choice for next-generation edge-AI inference engines.

The major contributions of this work are as follows:

- **Dual-Purpose 8T SRAM Bitcell:** We introduce an 8-transistor SRAM cell capable of multiplication, which addresses the bit flipping issue. Further, a CIM-enabled 16Kb cache is proposed with improved compute density.

- **Novel 7T Full Adder:** A novel area-efficient full adder with just seven transistors is proposed, effectively contributing to a 75% reduction in transistor count per adder.

- **Area-Efficient 2D Interleaved Adder Tree Design:** Adder tree with 7T full adder and 28T full adder (FA) interleaved RCAs enables parallel

accumulation operations with significantly large signal margin, reducing overall adder tree transistor count by 40%.

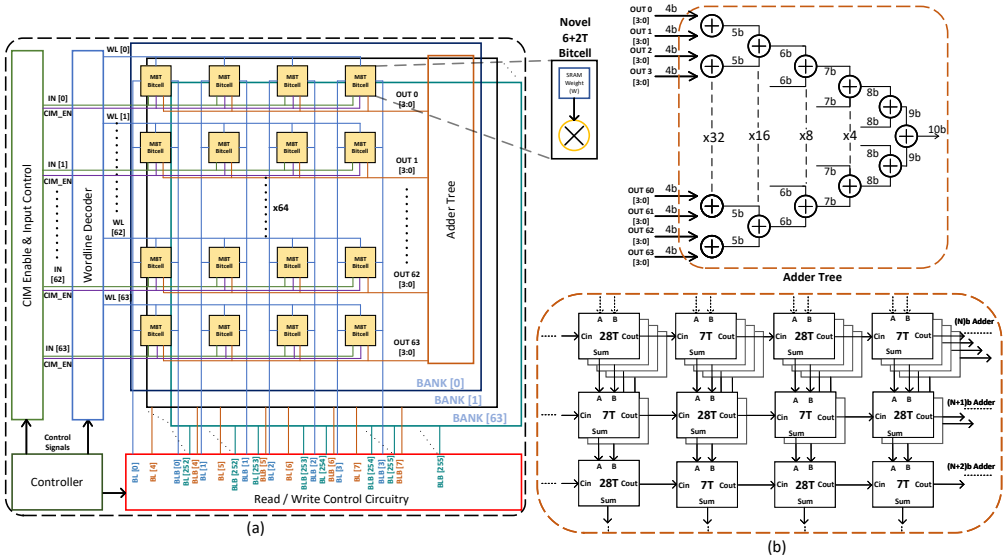

Figure 2.1: (a) Proposed DCIM architecture, (b) Proposed Adder Tree Structure. [1]

2.1 Proposed Digital Compute-in-Memory Architecture

A significant portion of NN inference is MVM/MAC computations, typically performed within DCIM macro. Thus, our work focuses on a performance-enhanced SRAM array and adder tree. The overall architecture for the proposed 16Kb DCIM macro is illustrated in Fig. 2.1a. The SRAM array consists of 64 banks, each with 64 rows and 4 columns(bit width=4), and is utilized to store binary filter weights as outlined in Table 2.1. The runtime programmable CIM operation is executed row-wise, assisted by a CIM_EN driver and selection multiplexers. The inputs are fed in a bit-serial fashion from MSB to LSB, generating partial products in every clock cycle. The weight precision is limited to 4-bit and can be scaled in multiples of 4 using multiple banks, while the input bit-width is adjustable as required, requiring N computation cycles for N-bits. The partial products for each kernel are computed

within a single SRAM bank and passed to an adder tree for a 10-bit sum. The MAC result is calculated from the partial sums with an accumulator circuitry over N+1 clock cycles.

2.1.1 Multiplication Capable 8T SRAM Bit-Cell

The proposed 8T SRAM bit cell, as demonstrated in Fig. 2.2, includes respective input activation (IN), weight (stored at Q node), and output product (MUL) for MAC computation. In addition to standard 6T SRAM normal read/write operations, two additional transistors (MP and MN) perform AND operation without affecting the wordline and bit lines. BL and BLB are pre-charged to VDD during the SRAM storage operation mode and enabled for read operation. The input (IN) is fed to the MP transistor upon enabling the CIM mode, and the multiplication output is obtained (MUL).

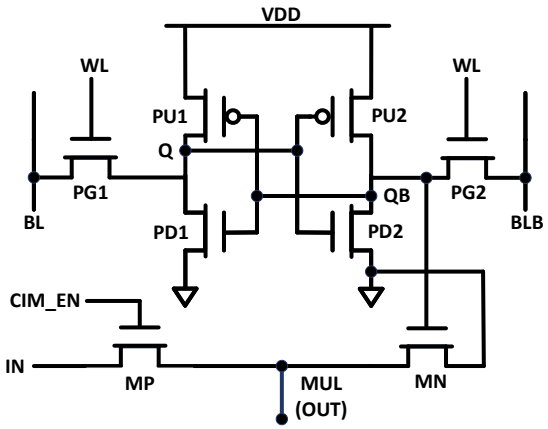

Figure 2.2: Multiplication capable 8T SRAM bitcell. [1]

Thus avoiding the need to turn on WL, unlike prior SOTA works [4, 23], and eliminates the potential bit flip due to the simultaneous activation of multiple rows. This also allows for concurrent weight updating when the input activation = 0, thus improving the energy efficiency and throughput. The multiplication functionality was validated through 100K Monte-Carlo simulations, and the standard deviation for the delay was at 1.77ps, which shows a 4× improvement over [9]. Hence, DCIM macro based on this 8T bitcell can efficiently produce accurate multi-bit partial

products. However, this work highlights the 4-bit weight computations from the edge-AI perspective.

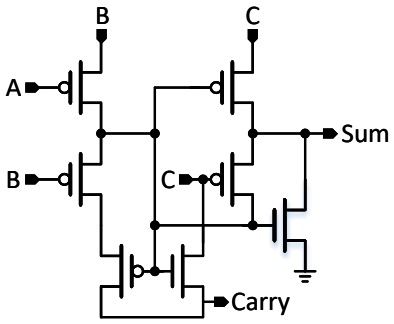

Figure 2.3: Schematic of 7T Full Adder. [1]

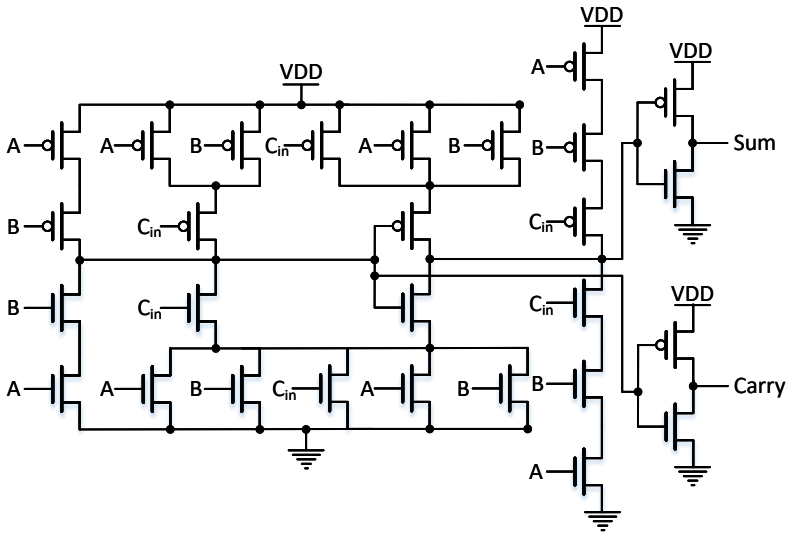

Figure 2.4: Schematic of 28T Full Adder. [1]

2.1.2 Proposed 7T Full Adder

DCIM faces reduced device efficiency due to the area overhead of the adder tree, whereas ACIM accumulates directly dot products from bit-line voltages. The traditional RCA-based adder tree utilizes 28 transistors for a single full adder. Thus, we introduced a novel 7T full adder, improving the transistor count and CMOS area by 75% and 80%, respectively, over its conventional counterpart. The schematic for the proposed FA and 28T FA are shown in Fig. 2.3, 2.4, respectively. Note that there

will be a voltage drop in the 7T FA, and to avoid the carry of this voltage drop to the next stage, a 2D interleaved adder tree structure is proposed. The pass-gate logic contributes to static power consumption, but the proposed adder tree significantly enhances compute density and noise margin. The simulation shows that the static power consumption is considerably smaller, only about 6.13% of dynamic power consumption, thus showcasing the proposed adder used within the adder tree in the DCIM macro as better suited for energy-efficient edge-AI devices.

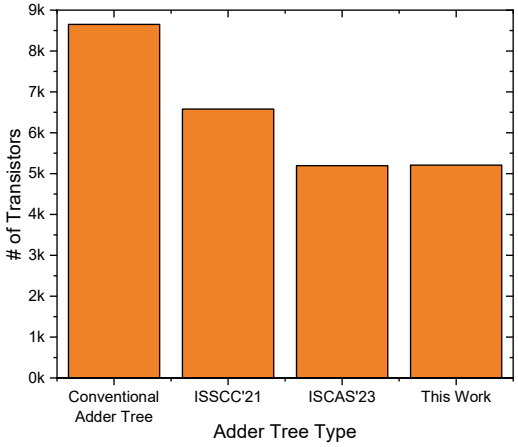

Figure 2.5: Comparison of Various Adder Tree Techniques. [1]

2.1.3 2D Interleaved Adder Tree

The 2D interleaved adder tree uses an interleaving structure of 28T FA and 7T FA in the same RCA and also an interleaving structure of such interleaved RCAs in between the N-bit RCA stage and N+2 bit RCA stage of the adder tree. From Fig. 2.1b, one can see the 2D interleaved structure of the tree; in the case of the first stage of the adder tree, the 4-bit RCA has the first adder as a 28T FA and then a 7T FA and so on, this is done to mitigate the threshold voltage loss incurred due to the PTL based multiplication from the bitcell. Now for the second stage of the adder tree, the 5-bit RCA will have FA starting and ending with 28T FAs with 7T FAs in between them; this is done to remove the voltage drop, which is introduced by the 7T FA which is at the last stage of 4-bit RCA of the previous stage. For the third

stage of the adder tree, the 6-bit RCAs will now have 7T FA as their starting adder, with the last FA being a 28T one, and so on, the adder tree structure is designed. Fig. 2.5 shows the total transistor counts of various implementations of the adder tree.

2.2 Evaluation Results & Discussion

The 8T SRAM bitcell occupies an area of 4.1 µm^2, 1.6× larger than a conventional 6T SRAM cell at a 65nm CMOS technology node. The simulation of the adder tree shows a static power consumption of 29.5 µW, which is relatively smaller than the dynamic power consumption of 481.3 µW at 1.2 V. The optimized novel 7T FA consumes an area of 4.77 µm^2. The 7T FA has an area reduction of 80% compared to the conventional 28T FA area. Fig. 2.6 and Fig. 2.7 show the accumulator circuit architecture and the timing diagram, respectively, for the CIM operation. For the 4-bit mode, the first 4 cycles are used for partial product and partial sum generation, and the 5th cycle is used to generate the final MAC output from the accumulator.

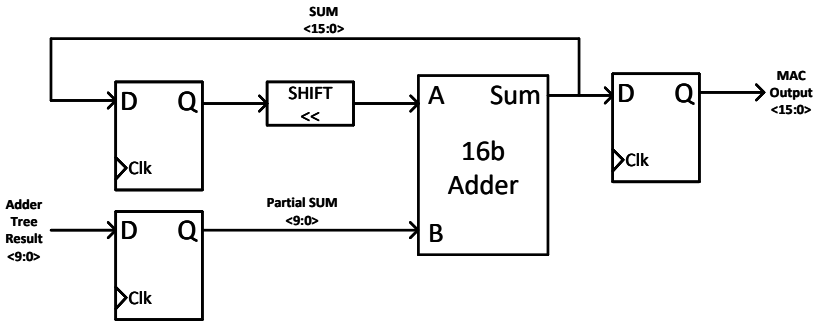

Figure 2.6: Structure of Accumulator. [1]

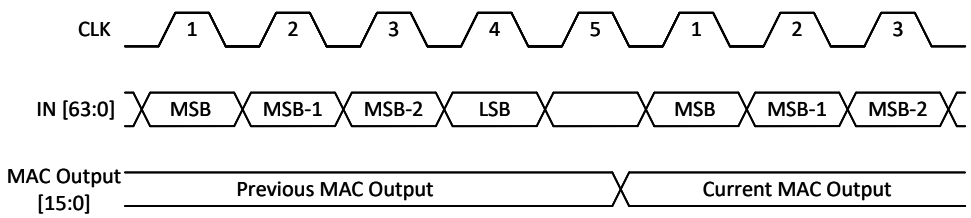

Figure 2.7: CIM Operational Timing Diagram. [1]

For real-time multi-character recognition, the LeNet-5 network (see Fig. 2.8) with additional Support Vector Classifier (SVC) with 45 binary classifiers is evaluated on both the MNIST and A-Z alphabet datasets. We developed a parameterized software evaluation model with *QKeras* library using *Python* 3.0 on the *Google Colab* platform and extracted weights in *CSV* format for manual mapping to DCIM macro. The computations were performed using 1-bit activation input/4-bit weight (1A4W) and 4-bit activation input/4-bit weight (4A4W). The classification accuracy was observed at 98.7%, 98.8% for 1A4W and 99.1%, 97.8% for 4A4W on the MNIST and A-Z alphabet datasets, respectively. We pruned the model by 40% without losing accuracy, reducing MAC operations significantly, and the same has been utilized for DCIM mapping evaluation. The hardware performance emulation model is defined on the following design parameters: DCIM array dimensions, activation and weight precision, memory banks required for layer execution, and reconfigurable hardware multiplexing.

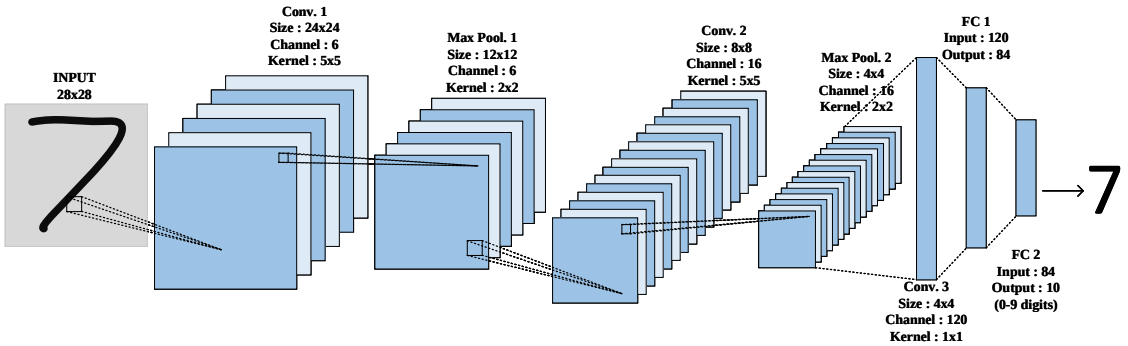

Figure 2.8: Visualization of LeNet-5 Network Architecture. [1]

Sparsity, achieved through network pruning, is crucial in improving DCIM energy efficiency. Pruning systematically reduces the number of parameters/connections in an NN by removing less significant weights, resulting in a sparse network structure. This introduces an opportunity to skip zero-based multiplication and addition operations during computation, which in turn significantly reduces the workload on the DCIM array. Thus, by executing the reduced workload on a small DCIM array, both power and macro utilization are minimized, which leads to improved energy efficiency. In our evaluation of the macro, we pruned the network by 40% with the

loss pr accuracy, maintaining a 99.8% quality of results (QoR). This resulted in a 27% reduction in MAC operations and SRAM bank utilization, as well as a 33% reduction in operating cycles required for AI workload computation with the proposed macro. These results highlight how the proposed sparse DCIM design effectively leverages the pruning technique to achieve substantial performance gains while maintaining accuracy and making it a highly efficient solution for edge-AI applications.

Table 2.1: Pruned LeNet-5 Specifications

Layer	**Kernal**	**Output**	**Banks Used**	**Op Cycles**
Input	-	$28 \times 28 \times 1$	NA	-
Conv. 1	$5 \times 5 \times 6$	$24 \times 24 \times 6$	6	384
Max Pool 1	2×2	$12 \times 12 \times 6$	NA	-
Conv. 2	$5 \times 5 \times 16$	$8 \times 8 \times 16$	32	48
Max Pool. 2	2×2	$4 \times 4 \times 16$	NA	-
Conv. 3	$1 \times 1 \times 120$	$4 \times 4 \times 120$	45	16
Flatten	NA	1920×1	NA	-
FC 1	84	10	168	3840
FC 2	10	1	20	64

* Off-chip operations and post-accumulation cycles are not considered.

The proposed DCIM macro is utilized by reprogramming weights layer by layer, featuring an off-chip bit-serialized ReLU activation function and control engine circuitry like [24]. The result combination accumulator, sub-sampling layer, softmax non-linear layer, and SVC are considered outside DCIM macro. This work evaluated layer-reuse philosophy, maintaining kernel-wise modularity and macro scalability to various network configurations, which is well suited for edge-AI applications. Table 2.1 illustrates the memory mapping for automatically loading LeNet-5 kernels layer by layer in the proposed macro, and Table 2.2 compares the proposed macro with the SoTA works. Conv. 2 layer has a filter size of 5×5, 6 input channels and 16 output channels as shown in Fig. 2.9.

Each set of 16 three-dimensional filters is mapped to one bank in the macro. Three banks must accommodate the entire kernel, as each bank is restricted to 64 words. Thus, a total of 32 banks are necessary to compute the Conv. 2 with 1A4W

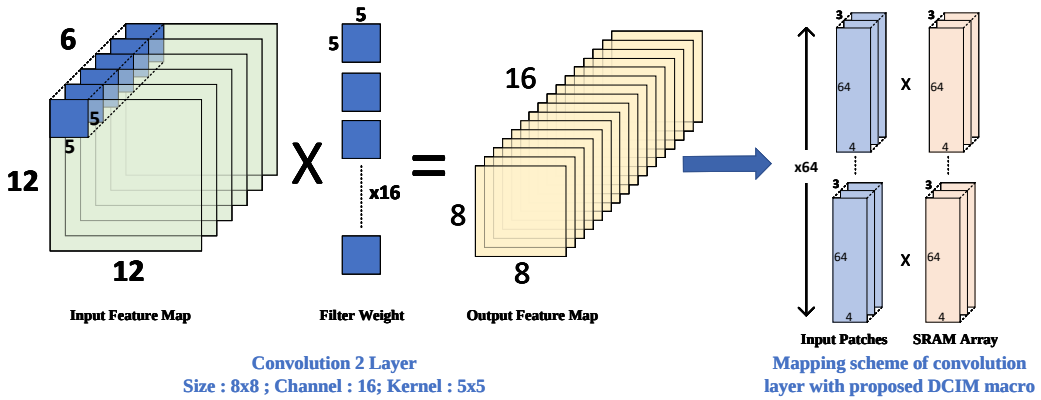

Figure 2.9: Convolution layer Mapping for the proposed DCIM macro. [1]

precision. The kernel slides 48 times over the previous feature map, necessitating 48 computation cycles to process the entire layer. The proposed DCIM architecture achieved 2× improvement in energy efficiency (480 TOPS/W) compared to state-of-the-art solutions.

Table 2.2: Performance Comparison with State-of-the-Art ACIM and DCIM Macros

Parameter	Chih et al.'21 [7]	Oh et al.'23 [9]	Yi et al.'24 [8]	Wang et al.'23 [23]	Sharma et al.'22 [5]	Kim et al.'21 [3]	Saragada et al.'23 [25]	This Work
Technology [nm]	22	28	55	40	65	65	65	65
Supply Voltage [V]	0.72	0.6-1.1	1.2	0.8	1	0.6	1.2	1.2
Cell Type	6T	6T+0.5T	8T	PT-8T	AND8T	NA	10T	8T
Bitcell Area [μm^2]	0.38	0.38	4.3	NA	2.78	10.5	4.5	4.1
Array Size [Kb]	64	16	64	16	16	16	16	16
Operating Frequency [MHz]	100	30-360	200	100	100	140	25	250
Weight Precision [bit]	4/8/12/16	8	4/8/12/16	4/8	4	1-16	1-4	4/8
Activation Precision [bit]	1-8	1-8	4/8/12/16	1-8	4	1-16	4	1-8
Model	NA	NA	ResNet-10	NA	VGG-8 Like	LeNet-5	CNN-Type	LeNet-5
Accuracy	NA	NA	93.7	NA	96	99.2	98.7	99.1
Throughput [TOPS]	0.83	NA	NA	0.8	0.42	0.57	0.8	2.2
Energy Efficiency [TOPS/W]	23	60	67	94	180	156	273	480

2.3 Conclusion

This paper presents a DCIM macro designed to address the challenges in state-of-the-art works, specifically focusing on reducing the overall area utilization of the SRAM array and the adder tree. The multiplication operation is achieved using an 8T SRAM bitcell, which eliminates potential bit flipping caused by the simultaneous

activation of multiple rows. The proposed DCIM macro also incorporates a novel 7T full adder used in the 2D interleaving adder tree, which mitigates the threshold voltage drop issue of the bitcell and the voltage drop from the RCAs. This design provides full output voltage swing and achieves a significant transistor count reduction of up to 40%, optimizing the adder tree design. The circuit-level simulation was done on Cadence Virtuoso and Mentor Graphics Calibre, with the software framework (NN mapping and accuracy) done using Python on Google Colab. The proposed DCIM macro achieves a maximum throughput of 2.2 TOPS at 250 MHz and at 1.2V utilizing CMOS 65nm Technology. The 16Kb DCIM macro is evaluated for 1-bit activation and 1-bit weight precision (1A4W) and 4-bit activation and 4-bit weight precision (4A4W) configurations for the LeNet-5 network using MNIST and A-Z alphabet datasets, achieving 98.7%, 98.8% for 1A4W and 99.1%, 97.8% for 4A4W inference accuracy, respectively. Compared to SOTA works, the improvements in energy efficiency (2×) in the proposed DCIM macro make it better suited for compact battery-powered edge-AI devices.

Chapter 3

RAPID: Re-configurable Adder Tree based Performance-Incentivized AI Digital CIM Macro

The increasing demand for real-time intelligence in edge devices necessitates computing platforms that are energy-efficient, reconfigurable, and capable of handling data-intensive deep neural network (DNN) workloads. Digital Compute-in-Memory (DCIM) architectures offer a promising path forward by minimizing memory-compute data movement. However, existing solutions suffer from limited precision support, inefficient adder tree designs, and low resource utilization under sparse neural workloads [13]. In this work, we present RAPID-CIM, a reconfigurable, energy-efficient SRAM-based DCIM macro built on a novel dual-purpose M8T bitcell and an area-optimized RCA employing new 11T and 14T adders. The architecture demonstrates high accuracy across standard DNN workloads while significantly improving energy efficiency and compute density.

State-of-the-art DCIM and ACIM designs often rely on high-overhead bit cells and bulky adder trees that restrict scalability, precision flexibility, and energy-efficient operation—especially under sparse network conditions prevalent in pruned DNNs. Moreover, the under-utilization of memory banks and static adder configurations leads to performance bottlenecks during workload execution. Motivated by these challenges, RAPID-CIM is designed to rethink bitcell and adder architectures with reconfigurable logic, enabling dual-purpose memory-compute fusion, low-power operation, and modular scalability. By integrating sparsity-aware acceleration and precision-tunable compute blocks, this work pushes the envelope in building compact, high-performance DCIM macros ideal for edge-AI platforms.

The key contributions of this work are as follows:

- **Novel M8T bitcell for DCIM array:** The proposed approach enables word line-independent multiplication with a novel M8T bit cell to address challenges associated with DCIM operations. The bit cell can work independently as a storage element and compute block without affecting each other's functionality.

- **Novel FA14T and HA11T for Area-efficient RCA:** The novel 11T half adder (HA11T) and 14T full adder (FA14T) provide a full rail-to-rail swing and an area-efficient adder tree with a ripple carry adder(RCA). The reductions in

# of transistors contribute to reduced area and enhanced compute density.

- **Enhanced Performance Reconfigurable DCIM Macro:** The proposed approach explores MUX-based reconfigurable adder tree architecture to address memory bank underutilization. The work significantly improves 16.7× energy efficiency and 8.6× compute density. The proposed macro has been evaluated for a wide range of workloads and diverse precision range under 40% sparsity factor, with application performance 97.8% QoR.

3.1 Proposed Design

AI inference focuses mainly on matrix computations (MVM/MAC), typically performed within DCIM macro.

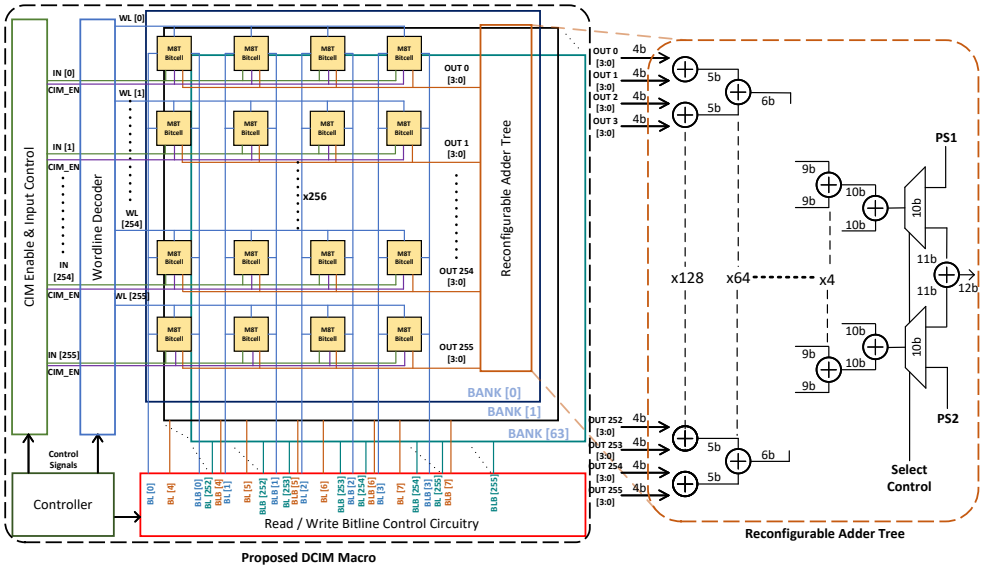

Figure 3.1: Proposed DCIM Macro. [2]

The fundamental elements of the proposed DCIM macro architecture include a novel M8T-based SRAM memory array, an optimized controller, and an accumulator with an area-efficient adder tree. Fig. 3.1 depicts the detailed circuitry for the proposed 64Kb macro. SRAM array is distributed into 64 banks, with 4 columns and 256 rows. The reconfigurable hierarchical adder tree is attached to the proposed

CIM array and assists in summing up outputs from each row concurrently passed to the accumulator within a single clock cycle. The key elements include novel M8T, HA11T, and FA14T, which are discussed in forthcoming subsections.

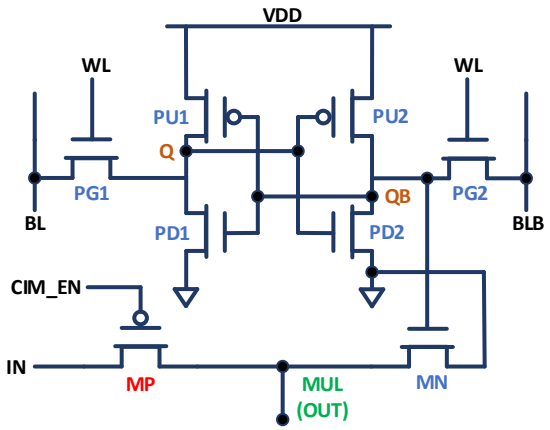

Figure 3.2: Schematic for Proposed M8T bit cell. [2]

3.1.1 M8T for Dual-purpose CIM array

The proposed M8T bit cell, as shown in Fig. 3.2, utilizes two transistors (MP and MN) for independent multiplication operation with a conventional 6T SRAM bitcell. With the CIM_EN signal, the pre-stored weights are multiplied with input data whenever required, apart from the traditional weight storage operation. The conventional 6T SRAM bitcell is used to store weight (W) data, and bitline (BL) and bitline bar (BLB) are pre-charged to VDD and enabled for the read operation. The input (IN) is fed from the source of the MP transistor, and the multiplication output (MUL) is obtained at the joint drain of MP and MN transistors during CIM mode. This avoids the need to turn on the wordline (WL) access transistor and eliminates the bit flip issue due to multiple rows' simultaneous activation. Using an MP transistor mitigates potential threshold voltage loss and provides rail-to-rail voltage swing at MUL. This marks a key difference between the proposed dual-purpose SRAM array and the SoTA SRAM-enabled CIM array.

$$\left(\frac{W}{L}\right)_{\text{MN}} = 2 \times \left(\frac{W}{L}\right)_{\text{MP}} \tag{3.1}$$

CIM_EN	IN	Weight	Q	QB	MUL	MUL (V)
1	X	X	X	X	No CIM Operation	
0	0	0	0	1	0	4.49E-06
0	0	1	1	0	0	0.3
0	1	0	0	1	0	0.11
0	1	1	1	0	1	1.2

Figure 3.3: CIM operation table with MUL logical value (In green) and actual voltage values. [2]

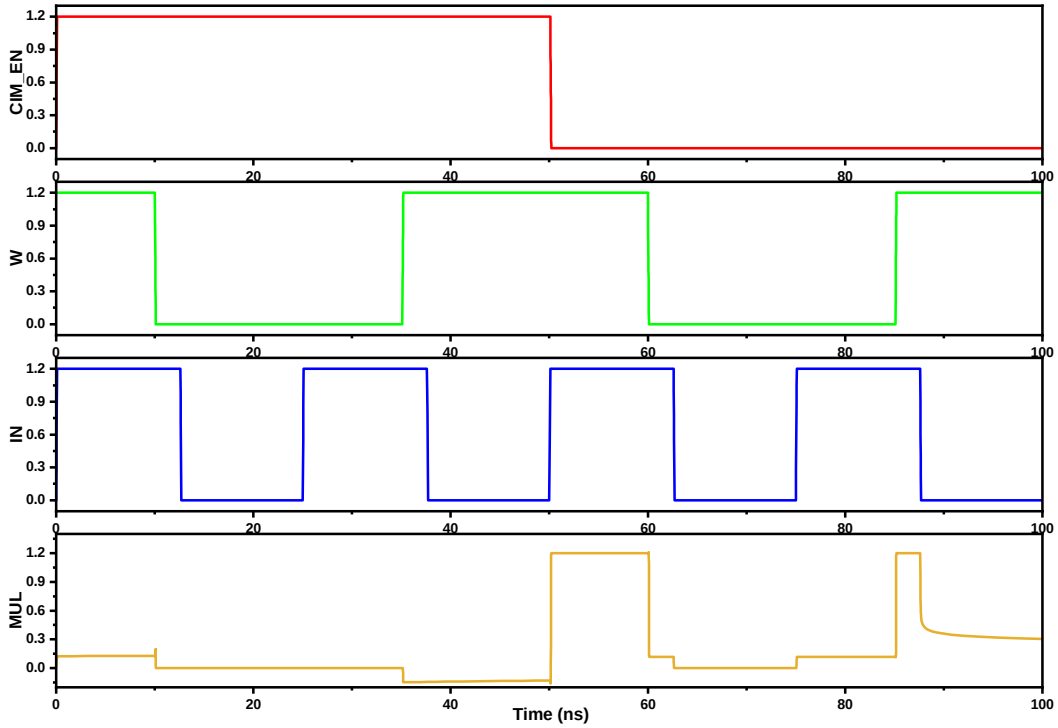

Figure 3.4: Simulation Waveform for Proposed M8T Bitcell. [2]

In CIM mode, the CIM_EN signal is low. If the input activation and data stored are logic 1, the MP transistor will pass logic 1 to the MUL port without threshold loss, thus performing the multiplication operation. Conversely, if input activation and data stored is logic 0, the MN transistor pulls the MUL node to the ground.

If the input activation is logic 0 and the data stored is logic 1, then the MUL is discharged through the MP transistor. Similarly, when the input activation is logic 1, and the data stored is logic 0, the MUL node is pulled down to near logic 0 levels due to the adequately sized MN (Eq. 3.1). This allows for simultaneous weight update when the input activation = 0, which improves energy efficiency and throughput. Fig. 3.3 showcases the detailed operation with logical and actual voltage levels for multiplication between Input and Weight. The simulation waveform for the bit-cell for all cases is shown in Fig. 3.4.

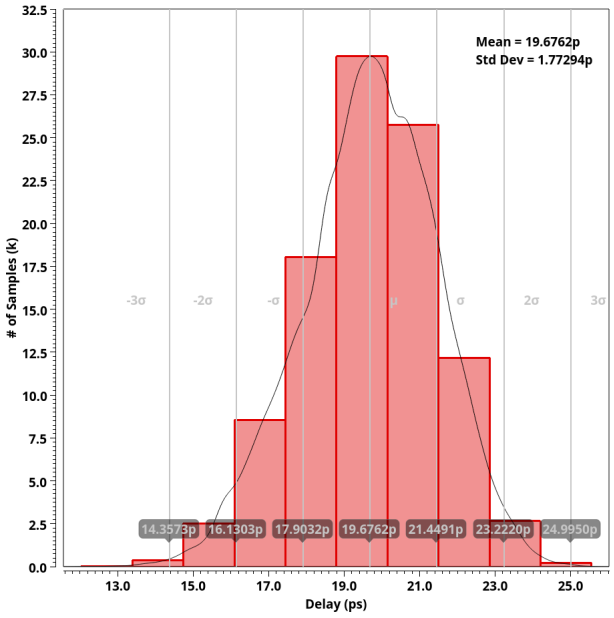

Figure 3.5: Monte-Carlo Simulations (Multiplication Delay) for proposed M8T Bitcell. [2]

The 8T SRAM bitcell occupies an area of 4.21 μm^2, 1.6× larger than a conventional 6T SRAM cell at a 65nm CMOS technology node. The 100K Monte-Carlo simulations for the multiplication delay are shown in Fig. 3.5, and the standard deviation for the delay demonstrated 4× improvement over [9]. Hence, DCIM macro based on the proposed M8T can efficiently produce accurate multi-bit partial products.

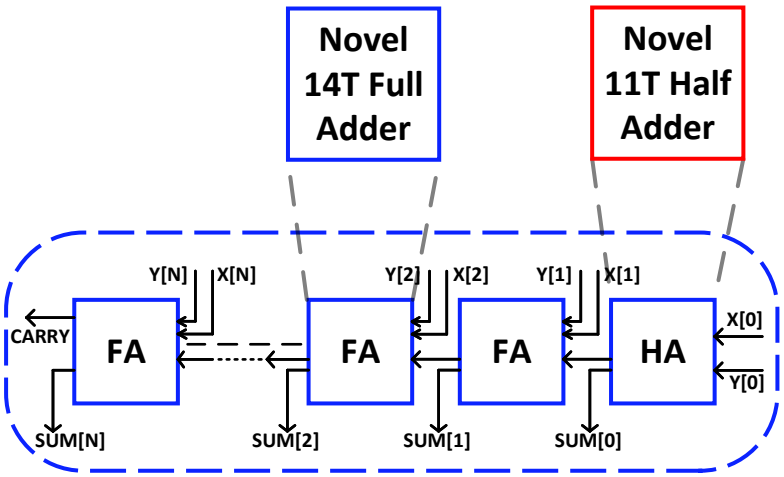

Figure 3.6: Ripple Carry Adder with proposed modifications. [2]

3.1.2 True output HA11T & FA14T for Area-efficient RCA

The partial product accumulation is done with the help of an adder tree in the DCIM array. However, it contributes to a significant loss in compute density compared to ACIM, thus making the accuracy-hardware tradeoff infeasible. Our work proposes an area-efficient ripper carry adder composed of HA11T and FA14T with transistor-level optimization without compromising functionality to address this, as shown in Fig. 3.6. The novel adder designs use the PTL approach to minimize redundant circuitry and dissipate low power. This approach also reduces parasitic capacitance and improves switching activity within reduced silicon area. The adder eliminates threshold voltage drops at output signals. It provides rail-to-rail output swing for true input signals, thus making it the ideal fundamental element for the RCA-based adder tree block.

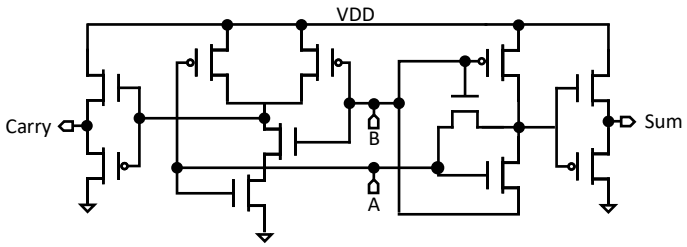

Figure 3.7: Schematic for Proposed 11T Half Adder. [2]

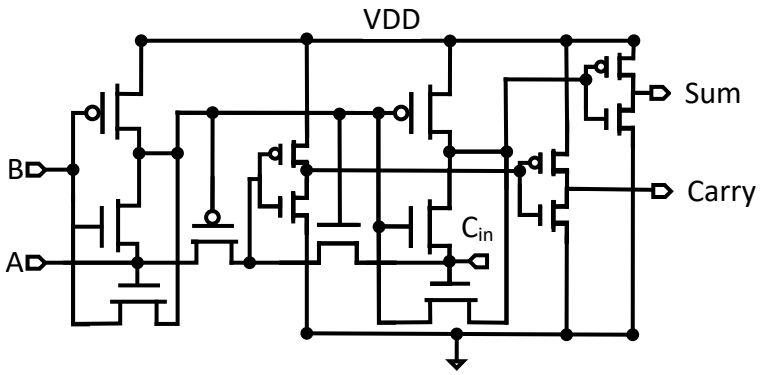

Figure 3.8: Schematic for Proposed 14T Full Adder. [2]

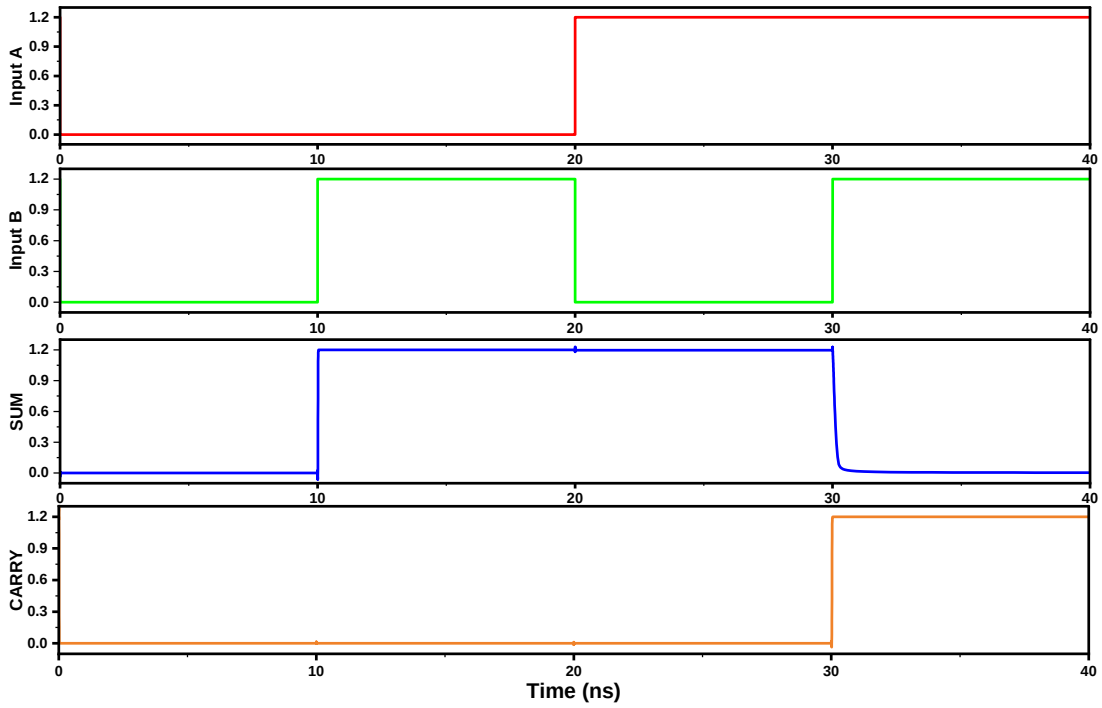

Figure 3.9: Simulation Waveform for Proposed 11T Half Adder. [2]

The proposed schematics and simulation waveforms for the proposed HA11T and FA14T are shown in Fig. 3.7, 3.8 and Fig. 3.9, 3.10, respectively. Unlike prior works [4, 7, 23], the proposed adder avoids the use of interleaving adder trees or the use of complementary FAs. The HA11T uses PTL XOR logic and NMOS AND logic, while FA14T utilizes two-stage XOR with PTL logic and AND-OR logic with CMOS logic, which does not affect the noise margin very similar to the conventional CMOS

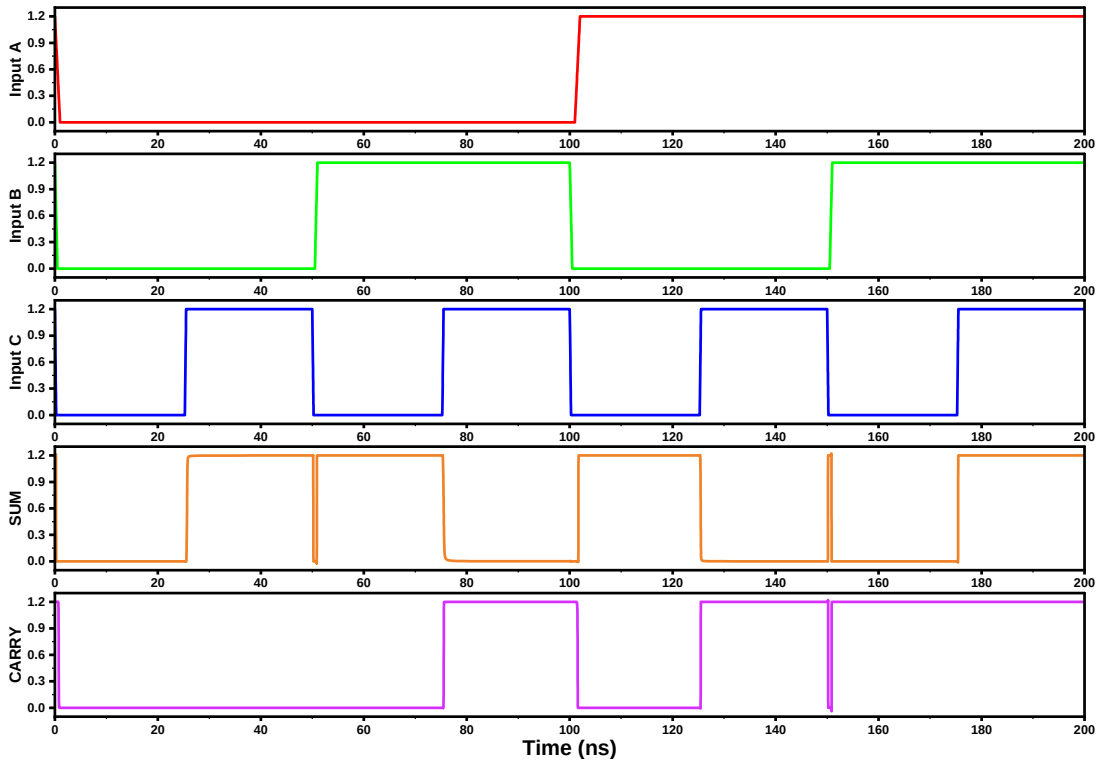

Figure 3.10: Simulation Waveform for Proposed 14T Full Adder. [2]

adder. The functionality is validated with post-layout simulation and observed that the adder tree consumes a tiny fraction in static power consumption, only 6.13% of dynamic power consumption with the proposed approach. The proposed HA11T and FA14T occupy an area of 5.39 μm^2, and 9.38 μm^2 respectively, which is approximately 4.4× and 2.25× smaller, compared to conventional 28T full adder at a 65nm CMOS technology node. The proposed 4-bit and 8-bit RCA minimizes area up to 36.45% and 38.2% compared to conventional RCA adder. This will scale linearly for huge savings in the adder tree.

3.2 Reconfigurable DCIM Macro Evaluation

The DCIM macro architecture focuses on critical components, such as the SRAM memory array, an adder tree, controller and peripheral circuitry, and an accumulator. Our dual-purpose DCIM array and area-efficient RCA contribute towards better

throughput with compute density and reduced power and delay to meet the needs of improved edge devices. The overall architecture for the proposed 64Kb DCIM macro is illustrated in Fig. 3.1. The dual-purpose DCIM array is programmed in a row-wise weight-stationary approach, assisted by a CIM_EN driver and selection multiplexers. The inputs are fed in a bit-serial fashion from LSB to MSB, generating partial products in every clock cycle. The weight precision is limited to 4-bit and can be scaled in multiples of 4 using multiple banks, while the input bit-width is adjustable as required, requiring N computation cycles for N-bits. The partial products for each kernel are computed within a single SRAM bank and passed to an adder tree for a 10-bit sum. The MAC result is calculated from the partial sums with an accumulator circuitry over N+1 clock cycles. A detailed illustration showcasing MAC computations is shown in Fig. 3.11, where multiplication is computed with an M8T-based SRAM array, followed by single cycle accumulation with the adder tree.

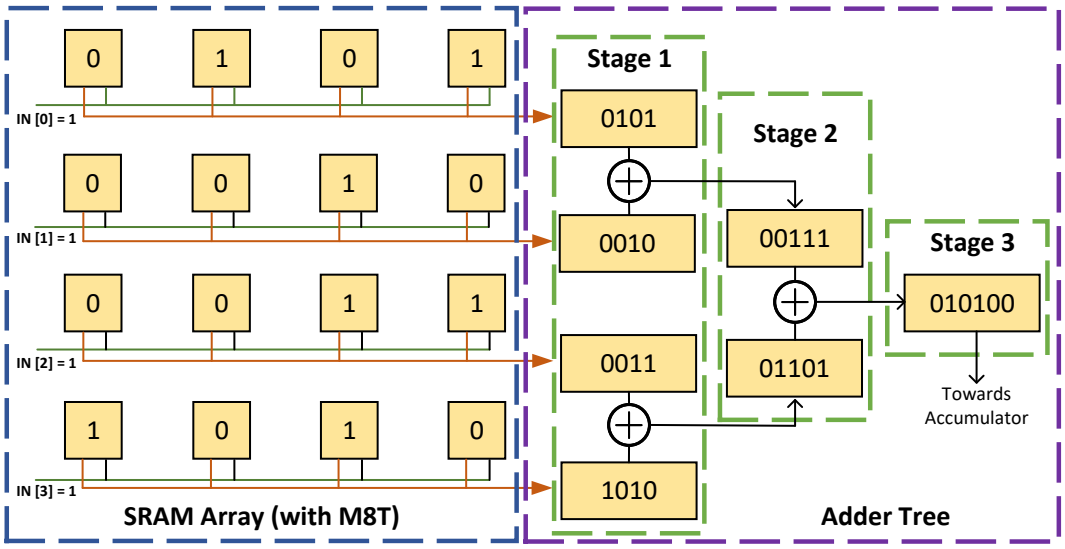

Figure 3.11: Illustration of 1A4W MAC Computation with the Proposed DCIM Macro. [2]

3.2.1 Reconfigurable adder tree for Incentivized performance

AI model evaluations showcased that DCIM macro is underutilized. Thus, we evaluated software evaluation of algorithms for resource-friendly, low latency, and

flexible precision computations, providing well-balanced optimizations across multiple metrics and delivering a performance-enhanced, lightweight DCIM macro. For the AI workloads evaluated, as shown in Fig. 3.12, we observed that the 40 % sparsity factor does not impact the application performance, thus helping us to accelerate DNN workloads with enhanced energy efficiency and compute density. However, this leads to fewer consecutive partial products than available rows in the DCIM array and a single accumulation associated with a memory bank. Thus, to address these, we decided to divide the adder into the order of 2 parts with the inclusion of multiplexers at the N-1 stage. Here, we discuss only dual accumulation and allowing the same DCIM array to be used for two MAC computations in two computation cycles at the cost of a minute delay. This contributes up to 3.2× reduction in operation cycles and a 30 % reduction in usage of memory banks, thus providing significant CIM acceleration and a primary source of increased throughput compared to prior works. The detailed analysis is discussed in the subsequent section.

Table 3.1: Alex Net Mapping

Layer	fmaps	Cout	Banks	Op. Cycles (x10^3)
Input	-	224 x 224 x 3	NA	-
Conv. 1	11 x 11 x 96 (Stride = 4)	55 x 55 x 96	96	9
MP1	3 x 3 (Stride = 2)	27 x 27 x 96	NA	-
Conv. 2	5 x 5 x 256 (Stride = 2)	27 x 27 x 256	4374	27
MP2	3 x 3 (Stride = 2)	13 x 13 x 256	NA	-
Conv. 3	3 x 3 x 384	13 x 13 x 384	1140	1.5
Conv. 4	3 x 3 x 384	13 x 13 x 384	2432	3
Conv. 5	3 x 3 x 256	13 x 13 x 256	2432	3
MP3	3 x 3 (Stride = 2)	6 x 6 x 256	NA	-
Flatten	NA	9216 x 1	NA	-
FC 1	4096	4096	4096	590
FC 2	4096	1000	2624	262
FC 3	1000	1	16	66

Note: For 1A4W, it takes one clock cycle per memory bank, as demonstrated in Fig. 3 (c). For 1A8W, two memory banks are required. For 2A4W, it takes two clock cycles, while for 4A4W, it requires four clock cycles (from LSB to MSB).

3.2.2 Evaluation for DNN performance

The proposed DCIM macro targets edge-node real-time multi-character recognition with custom additional 8-layered multi-class (45) support vector classifiers(SVC). The detailed neural network mapping for Alex-Net architecture with DCIM memory bank utilization and operation cycles are reported in Table 3.1. A parameterized software evaluation model with *QKeras* library using *Python* 3.0 on the *Google Colab* platform and extracted weights in *CSV* format for manual mapping to DCIM macro.

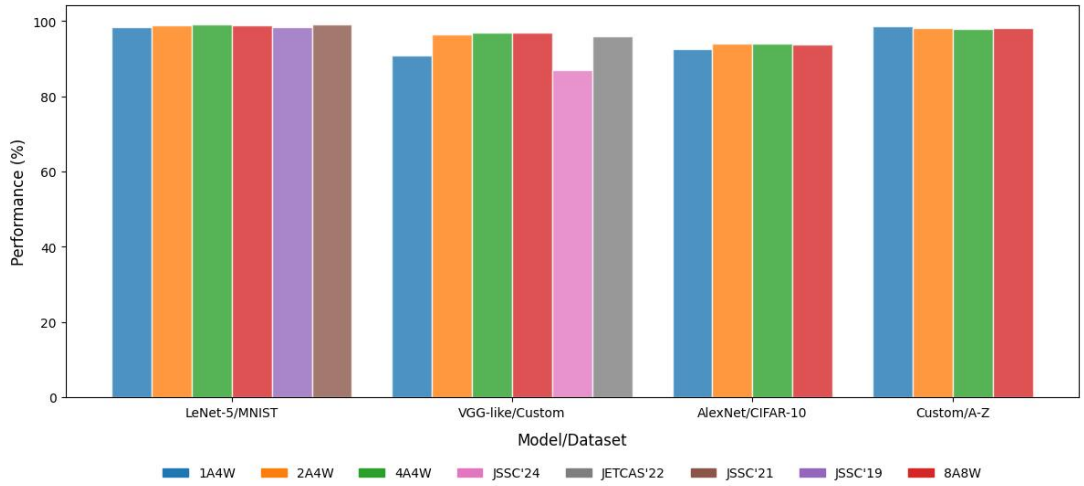

Figure 3.12: Comparison of inference accuracy with simulated Macro[1] compared with SoTA works [2], [3], [4], [5], [6].

The detailed precision with macro utilization tradeoff is shown in Fig. 3.12, which involves evaluation for LeNet-5/MNIST, VGG-like/Custom dataset, AlexNet/CIFAR-10 and Custom-8/A-Z Handwritten dataset at 1A4W, 2A4W, 4A4W and 8A8W and is compared with FP32-tensor and SoTA works [3–6]. The evaluation showcases the performance with 98.5 % QoR compared to FP32 as reported in [26] and outperforms SoTA works [3–6] with a significant improvement up to $0.5 - 4$ % in application accuracy.

The hardware performance emulation model is defined on the following design parameters: DCIM array dimensions, activation and weight precision, memory banks

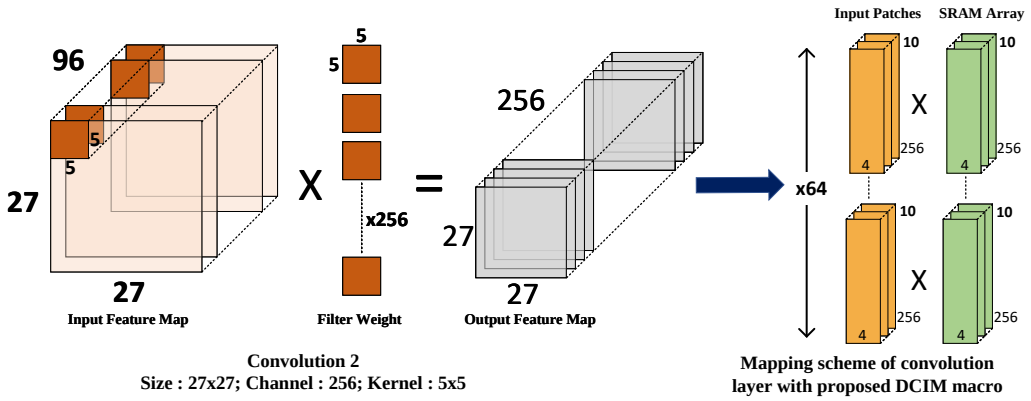

Figure 3.13: Convolution Layer Mapping for the Proposed DCIM Macro. [2]

required for layer execution, and reconfigurable hardware multiplexing. Table 3.1 illustrates the memory mapping for automatically loading Alex-Net kernels layer by layer in the proposed macro with macro mapping for convolution shown in Fig. 3.13. The proposed DCIM macro is utilized by reprogramming dense weights layer by layer, featuring an off-chip bit-serialized ReLU activation function and control engine circuitry in [26]. The result combination accumulator, sub-sampling layer, softmax non-linear layer, and normalization are considered outside DCIM macro. This work evaluated sparsity-aware layer-reuse philosophy, maintaining kernel-wise modularity and macro scalability to various network configurations, which helps to enhance performance for resource-heavy AI applications.

3.2.3 Performance metrics analysis

The implications of macro size, precision, and throughput for the proposed DCIM macro compared to SoTA works are discussed below to quantify the impact of this work. The proposed 64kb macro achieves 5.4× and 3.6× improvement in throughput compared to prior works [8] and [7], respectively. Our DCIM macro provides additional input precision support from $1-16$ bits and weight precision to 4/8/12/16 bits. At the same time, the prior works limit input precision to $1-8$ bits in [7] and 4/8/12/16 bits in [8] for similar weight precision support. The scaled proposed 16Kb macro shows 4.6×, 2.4× and 1.2× throughput improvements

compared to prior works [25], [5], and [9] respectively. However, the input precision is limited to 4-bit, $1-4$ bit, and $1-8$ bit, with weight precision to 4-bit, 4-bit, and 8-bit, strictly in the corresponding macros. [1] The increased throughput is primarily contributed by enhanced Operating frequency and reduction in AI workload due to sparsity exploitation.

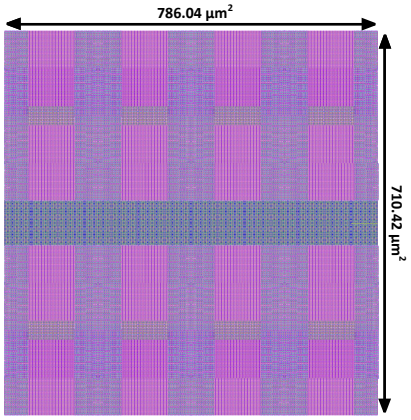

Figure 3.14: Layout of proposed DCIM macro. [2]

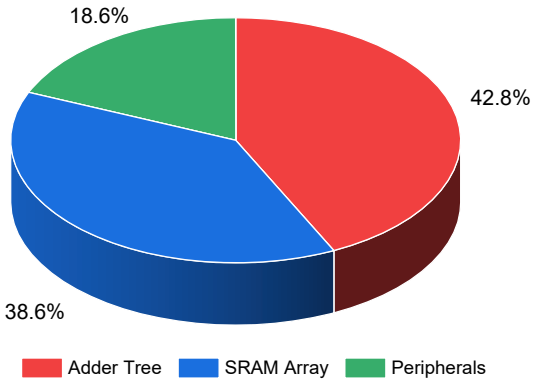

Figure 3.15: Area Utilization Breakdown of DCIM Macro. [2]

The layout and area breakdown of the proposed DCIM macro at the CMOS 65nm Technology node is illustrated in Fig. 3.14 and 3.15, respectively. The primary analysis focused on energy efficiency (TOPS/W) and compute density (TOPS/mm^2) assists in accessing the computational power of the device. The key factor driving

[1] **Note**: VGG-like/Custom refers to a custom 8-layer network implemented on the custom dataset (formed from selective images of MNIST, SVHN, and Car license plate detection).

the enhancement in compute density is incorporating an area-efficient reconfigurable adder tree for efficient macro utilization with a 40% sparsity factor. In contrast, the lower energy consumption is driven by design optimization, such as reduced static power consumption and separation of CIM operation.

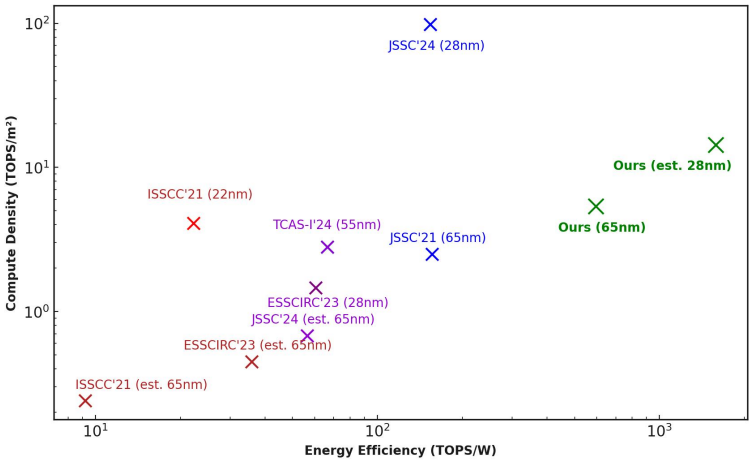

Figure 3.16: Performance analysis for proposed DCIM macro with SoTA works [2–4, 7–9]

The proposed DCIM macro achieves 16.7×, 9.2× and 3.8× better energy efficiency compared to [9], [8], and [3] respectively. The enhancements in compute density are by 8.6×, 4.1× and 1.9× compared to [9], [7], and [3] respectively. Technology scaling and macro size are adapted as detailed in Table 3.2 and Fig. 3.16 for fair comparison. The scaled proposed macro (28nm CMOS) estimates 14.7×, 9.6× and 6.3× higher energy efficiency compared to [9], [7], and [4] respectively and higher compute density by 8.6×, and 3.2× compared to [9], and [7] respectively. Increased compute density signifies a compact, cost-effective design, allowing more computations in reduced silicon area. Enhanced energy efficiency prolongs operational life in resource-constrained platforms while reducing operational costs associated with cooling and power consumption at cloud nodes.

Table 3.2: Performance comparison with SoTA ACIM and DCIM Macros

Parameter	ESSCIRC'23 [9]	JSSC'19 [6]	JETCAS'22 [5]	TNANO'23 [25]	JSSC'21 [3]	ISSCC'21 [7]	JSSC'24 [4]	TCAS-I'24 [8]	Ours
Tech. (nm)	28	65	65	65	65	22	28	55	65
MAC Operation	Digital	AMS	Analog	Digital	Digital	Digital	Digital	Digital	Digital
Cell Type	6T+0.5T	10T	AND8T	10T	NA	6T	8T	8T	M8T
Supply Voltage (V)	0.6-1.1	0.8-1.0	1	1.2	0.6-0.8	0.72	0.45-1.1	1.2	1.2
Array Size	16Kb	16Kb	16Kb	16Kb	16Kb	64Kb	16Kb	64Kb	64Kb
Op. Frequency (MHz)	200	5	100	25	138	100	250	200	250
Bitcell Area (μm^2)	0.39	NA	2.78	4.5	10.53	0.37	NA	4.28	4.21
Macro Area (mm^2)	0.59, 3.43*	NA	NA	NA	0.23	0.22, 1.87*	0.49, 2.84*	2.8	0.87
Input Precision (bit)	1-8	6	4	4	1-16	1-8	1-4	4/8/12/16	1-16
Weight Precision (bit)	8	1	4	1-4	1-16	4/8/12/16	1	4/8/12/16	4/8/12/16
Model	NA	LeNet-5	VGG-8	CNN-Type	LeNet-5	NA	VGG-8	ResNet-10	AlexNet
Accuracy	NA	98.3	96.05	98.67	99.2	NA	86.96	93.7	94.2
Throughput (TOPS)	2.40, 1.62†	0.64	0.41	0.82	0.56	0.83, 0.52†	4.8, 2.97†	1.28	4.67, 1.93‡

Note: Equivalent Macro area* is normalized quadratically and Throughput† at CMOS 65nm, with Tech Scaling [4, 6, 7, 9, 25];
Scaled throughput‡ for 16Kb of proposed DCIM macro, for a fair comparison with 16Kb Macros, CMOS 65nm [5, 6, 25].

3.3 Conclusion

This work introduces RAPID-CIM, a reconfigurable and energy-efficient SRAM-based DCIM macro, to address SoTA edge-AI acceleration challenges. The circuit-level simulation was done on Cadence Virtuoso and Mentor Graphics Calibre on a 65nm technology node, with the software framework (NN mapping and accuracy) done using Python on Google Colab. The proposed macro improves up to a maximum of 16.7× energy efficiency, 8.6× compute density, and a 3.2× reduction in Operation Cycles compared to prior works, with a novel M8T bitcell, scalable bit-precision reconfigurable adder tree. The detailed evaluations on AlexNet, LeNet-5, and custom multi-class classification models demonstrate accuracy improvement over SoTA methods with minimal resource overhead. Thus, the proposed solution is better suited for compact battery-powered edge devices.

Chapter 4

DARE: Delay and Area-optimized Reconfigurable Edge DPIM Macro

With the rapid proliferation of edge-AI applications, there is a growing demand for low-power and high-throughput compute-in-memory (CIM) architectures that overcome the memory-compute bottleneck of traditional systems. Processing-in-memory (PIM) architectures, particularly digital PIM (DPIM), offer promising improvements in energy efficiency and compute density for matrix-intensive workloads [21]. However, existing solutions often suffer from increased latency, large area overheads due to bulky adder trees, and poor adaptability to sparse neural workloads. In this work, we propose DARE—a Delay and Area-optimized Reconfigurable edge DPIM macro that combines a novel A7T bitcell for subthreshold PIM operation with a power-gated Carry Skip Adder (CSkA) based adder tree for optimized accumulation. The DARE macro offers dynamic precision reconfigurability and high sparsity tolerance while significantly improving energy efficiency and throughput.

Current digital PIM implementations face challenges in meeting edge devices' stringent energy and area constraints due to inefficient compute blocks, voltage swings, and static power dissipation in large adder trees [19]. Moreover, many designs lack runtime adaptability to varying precision and sparsity, which limits their flexibility in real-world AI deployments. DARE addresses these issues by introducing a dual-purpose A7T bitcell that enables bit-wise multiplication independent of the word line, along with a novel power-gated CSkA adder tree optimized with compact FA-7T and FA-14T adders. This architecture not only reduces delay and silicon footprint but also allows for reconfigurable precision computation and effective sparsity exploitation—achieving up to 2.85× energy efficiency and 3.2× compute density improvement over state-of-the-art designs.

The primary contributions of this work could be listed as:

- **Novel A7T bitcell:** We introduce A7T PIM bitcell to perform word-line independent bit-wise multiplication between weight stored in ST 5T SRAM bitcell and inputs passed to 2T multiplication block.

- **Novel Delay-optimized Power-Gated CSkA:** Our proposed Power Gated CSkA reduces area, latency, power, and energy consumption comparably to

SoTA adder design approaches. Our design uses novel FA-7T and FA-14T blocks and introduces delay-reduction logic.

- **Enhanced Performance Reconfigurable DPIM Macro:** The proposed approach achieves cumulative gains from power reduction in the ST PIM array, delay-area optimized Adder tree architecture, and simultaneous reconfigurable precision computation with sparsity-exploited workload reduction. Thus, our proposed DARE macro enhances energy efficiency and compute density by 2.85× and 3.2× compared to SoTA works, respectively.

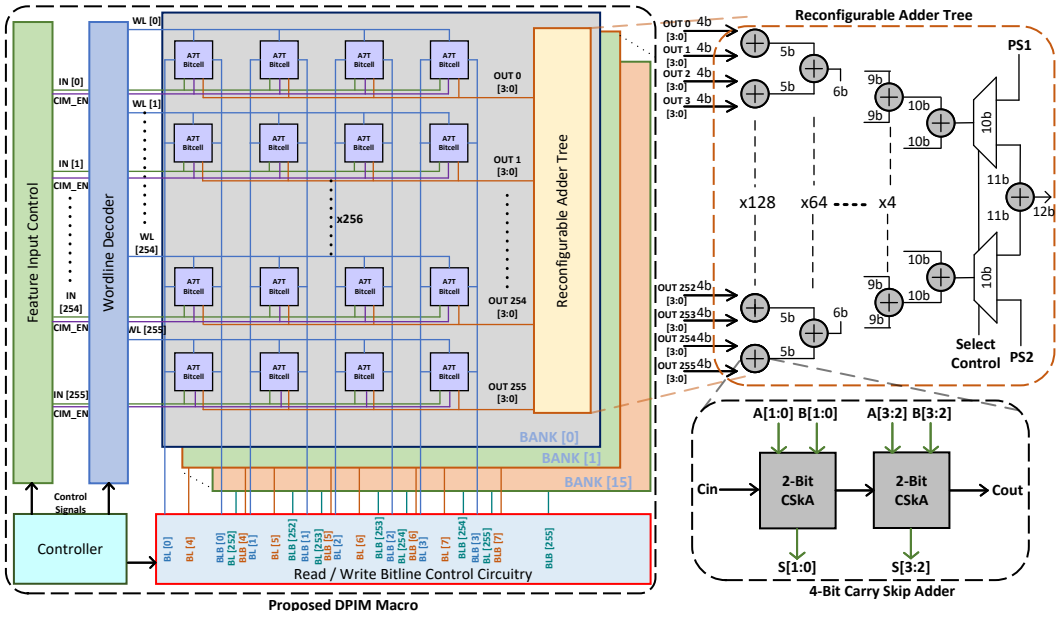

Figure 4.1: Architecture of the Proposed PIM Macro. [10]

4.1 Proposed DARE macro

AI inference is predominantly driven by matrix computations (MVM/MAC), where the DARE macro focuses on acceleration. The fundamental elements include a novel A7T-based PIM array, resource-efficient adder tree, enhanced performance controller, and off-chip circuitry- an accumulator, ReLU (AF), Max Pooling, and Batch Normalization and Quantization block. The detailed circuitry for the proposed

16kb macro is depicted in Fig. 4.1. The PIM array is distributed in 16 banks, with 4 columns and 256 rows; each bank is equipped with a reconfigurable hierarchical adder tree to accumulate outputs from each row concurrently within a single clock cycle. The key focus of this work was on the low voltage A7T and a modified power-gated Carry Skip Adder (CSkA) designed with FA-7T and an FA-14T.

4.1.1 Novel dual-purpose A7T based PIM array

The conventional 6T memory cell in the ST region is constrained by both read and write margins due to high sensitivity to process variations and device mismatch. The MOS drive strength exponentially varies with the threshold voltage (V_T), severely affecting performance. Traditional 65-nm 6T cell exhibits read and write margin failure around 800 mV and 700 mV respectively [27]. This worsens further with device mismatch and cell failure with technology scaling.

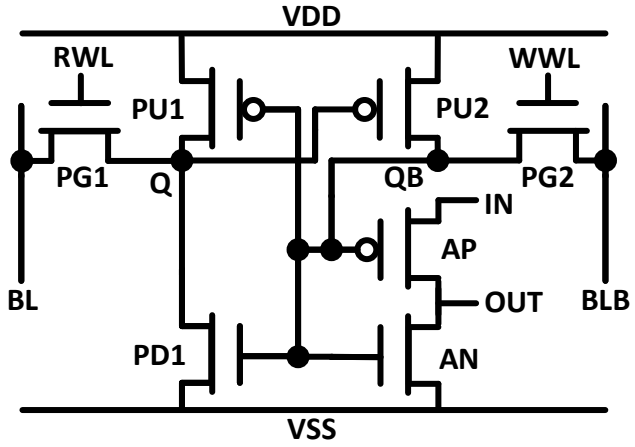

Figure 4.2: Schematic for Proposed A7T Bitcell. [10]

We overcome these challenges with a novel A7T PIM bitcell, which is comprised of subthreshold 5T SRAM bitcell [28], and 2T multiplication. The proposed bit-cell implements a single-ended read and differential/single-ended write with multiplication between input activation (IN) provided at the source of AP and complement of weight stored (QB). The output is obtained from the fused node between AN and AP. The schematic can be seen in Fig 4.2. The comparative PPA performance has been evaluated for different bit-cells and is discussed in Table 4.1.

Table 4.1: Comparative analysis of bit cells at 65nm CMOS Process

Parameters	**6T**	**8T [17]**	**6T+4T**	**6T+2T [20]**	**5T [28]**	**A7T (Ours)**
Bitcell Area (um^2)	1.75	2.35	3.67	2.52	1.55	2.23
Bitwise Multiplication	No	No	Yes	Yes	No	Yes
Layout Density	High	High	Low	High	High	High
Read Delay (ps)	106.2	102.7	127.5	108.2	104.6	108.3
Read Energy (fJ/bit)	62.4	58.8	69.2	63.5	55.9	59.2
Write Delay (ps)	169.6	164.3	216.7	173.2	157.2	168.1
Write Energy (fJ/bit)	140.4	133.1	155.8	139.7	142.3	140.7

The proposed cell behaves functionally very similar to a traditional 6T bit-cell and enables regular structure layout with slight modifications in the memory compiler. However, A7T lacks an NMOS pull-down transistor for the QB node. This has been addressed by low-threshold (LVT) PG2 and holding the right bit line (BLB) low during standby and read operations. This arrangement enables single-ended read operation through PG1 - left access transistor via an independent Read Word Line (RWL) signal. The differential write operations are carried out with RWL and the Write Word Line (WWL). The cell functionality is improved with high-threshold (HVT) pull-up PMOS transistors (PU1 and PU2).

4.1.1.1 Hold state Operation

The proposed cell behaves similarly to a standard cross-coupled latch for hold stage logical "0". When node Q is discharged, PU2 turns on ($V_{SGPU1} = V_{DD}$), allowing node QB to be fully charged to V_{DD}. Consequently, PD1 is activated ($V_{GSPD1} = V_{DD}$), ensuring that Q remains discharged. The enhanced A7T is more robust for holding the "0" state without a pull-down transistor beneath QB. For the hold "1" state, Q is charged to V_{DD}, and QB is discharged to the GND. PU_1 turns on ($V_{SGPU1} = V_{DD}$), holding Q high. The stronger leakage current from QB to the ground than from V_{DD} to QB ensures stability for the state- QB low.

4.1.1.2 Read Operation

The proposed cell behaves similarly to traditional 6T for differential read scheme. Both bit lines are pre-charged, and the access transistors are activated. This invasive state causes a rise in the voltage of the low internal data node (i.e., the node holding a "0"), which could trigger a potential bit flip. In the A7T cell, read access begins with BL pre-charged, keeping BLB in the standby state. Subsequently, a single-ended readout of node Q occurs once the RWL is asserted. If Q is high (the hold "1" state), there is no voltage drop across PG1, and all voltage levels remain unchanged. When Q is low (the hold "0" state), charge sharing between BL and Q forces the discharging of BL and results in a "0" readout.

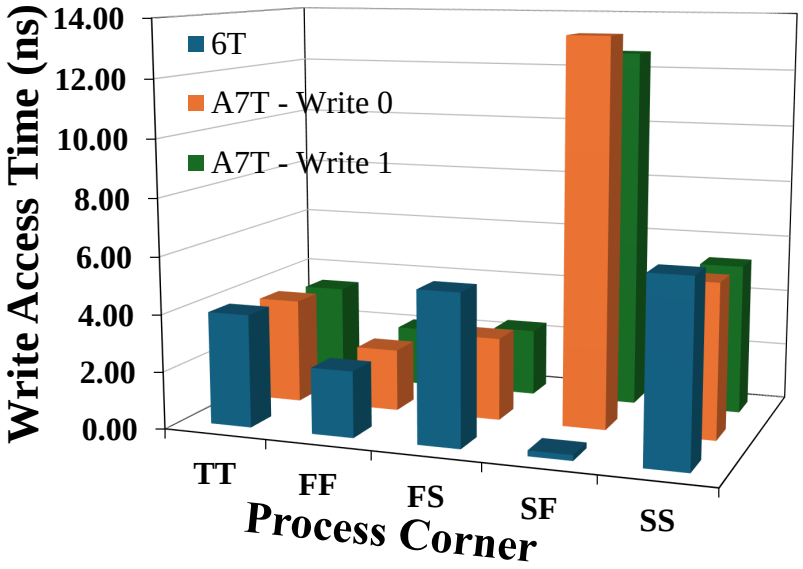

Figure 4.3: Comparison for Write delay across diverse process corners with A7T. [10]

4.1.1.3 Write Operation

The proposed cell uses a typical differential write operation similar to a conventional 6T bitcell. The bit lines are driven to opposite levels, and the word lines (WWL and RWL) are subsequently asserted. QB is pulled above the VT of PD1, enabling pull-down at node Q with charged BLB and the assertion of WWL. The write "0" operation can also be performed in a single-ended manner, while the former one

ensures a faster and more robust write operation. BL is charged, BLB is discharged, and both WLs are asserted for write "1" operation. QB must be discharged below the switching threshold of the left inverter (comprising of PU1 and PD1) to flip the bit-state successfully. The extensive comparison for the write access time across diverse process corners is demonstrated in Fig. 4.3.

IN (Input)	Weight (Q)	QB	OUT (V)
0	0	1	0 (0V)
0	1	0	0 (0.28V)
1	0	1	0 (0V)
1	1	0	1 (1.2V)

Figure 4.4: PIM functionality with logical OUT (In green) and actual voltage. [10]

4.1.1.4 Multiplication Operation

The A7T performs multiplication operation with 2T AND operation. It can be interpreted as a modified CMOS inverter, where the input activation is provided at the source of AP, and the complement of weight stored in SRAM bit-cell(QB) is applied as input to the combined gate of AP and AN. The output is obtained from the common drain between AP and AN. This ensures the direct connection between VDD-GND, a critical issue in [1, 29], and significant power consumption. The PIM functionality showcasing expected logical Output value and actual output observed can be seen in Fig. 4.4. The detailed Monte Carlo analysis for the multiplication delay is shown in Fig. 4.5, the Gaussian curve of mean $\mu = 19.68$ ps and deviation around mean $\sigma = 1.77$ ps.

4.1.2 Resource-efficient Power-Gated CSkA-based Adder tree

The partial product accumulation in the DPIM array is done with bulky adder trees. However, this approach is a major source of reduced compute density compared

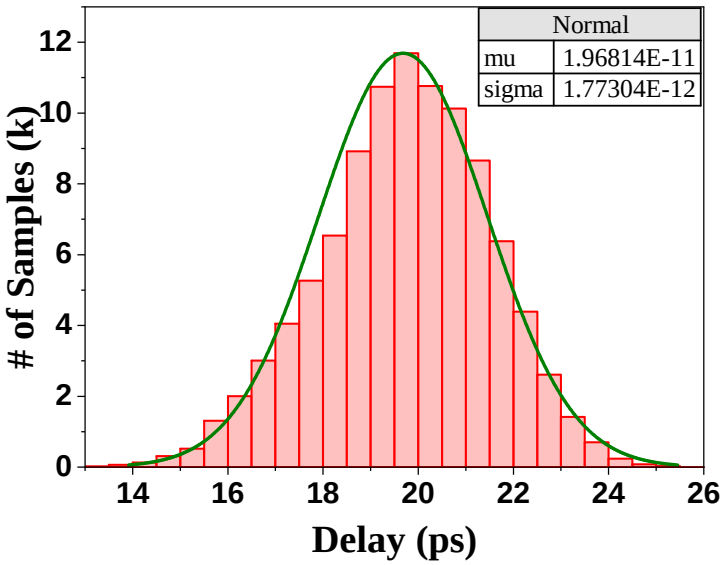

Figure 4.5: Monte-Carlo Simulations (Multiplication Delay) for proposed A7T Bitcell. [10]

to APIM. Several efforts are made to minimize area overhead, such as interleaved and complementary adder tree structures [16, 29–31]. We address it with an area, power, and delay-efficient power-gated Carry-Skip Adder (CSkA), which leverages FA-7T and FA-14T with transistor-level optimizations, as to achieve the required bit width, the even-bit adder configuration is implemented using a simple daisy chain of $(N/2)$ 2-bit power-gated CSkA full adherence simultaneously. This is the first work to adopt such an approach for optimizing adder tree efficiency. To substantially reduce the power consumption due to leakage, the internal supply voltage (VDD_PG) to the 2-bit CSkA sub-circuits is gated to either ground or to global supply (VDD); this is done using an LVT PMOS & an LVT NMOS device During computation mode the internal supply is connected to global supply via a power-gating control signal (PG_En) but in standby or write mode the power gating signal connects the internal supply to ground thus completely disconnecting the adder tree power supply [32]. To achieve the required bit width, the even-bit adder configuration is implemented using a simple daisy chain of $(N/2)$ 2-bit power-gated CSkA full adder. In contrast, the odd-bit configuration is realized by appending a single FA-14T after the final 2-bit CSkA stage. The proposed 4-bit CSkA achieves area and delay reductions of up to

64.5% and 70.78%, respectively, compared to conventional 4-bit RCA counterparts. This efficiency scales linearly, contributing to substantial area and delay savings in larger adder tree implementations.

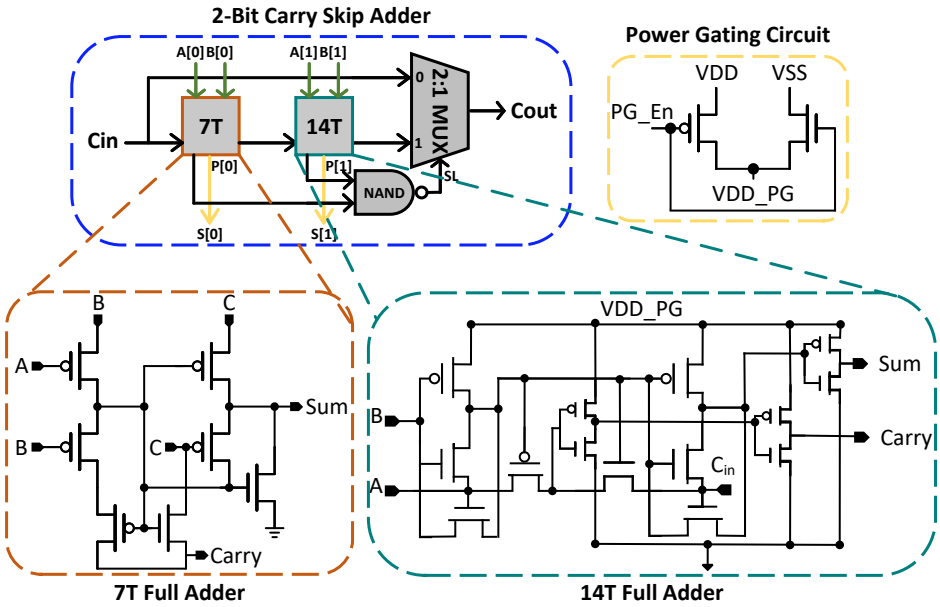

Figure 4.6: Proposed 2-Bit Carry Skip Adder Design. [10]

4.1.2.1 Full Adder 1: FA-7T

The conventional 28T CMOS adder functionality is realized with Pass Transistor Logic (PTL)-based FA-7T, gaining substantial reduction of 75% in transistor count and 80% in silicon area. The schematic for the FA-7T is illustrated in Fig. 4.6. FA-7T standalone suffers threshold voltage and cumulative signal degradation when incorporated in Ripple Carry Adder (RCA)-based architectures, particularly with bit width scaling. However, the proposed power-gated CSkA integrates a rail-to-rail swing, ensuring FA-14T next to each FA-7T, to effectively prevent this. The FA-7T occupies a compact silicon footprint of 4.77 μm^2, thus significantly enhancing computational density, albeit at the cost of a marginally increased static power dissipation up to 6.31%.

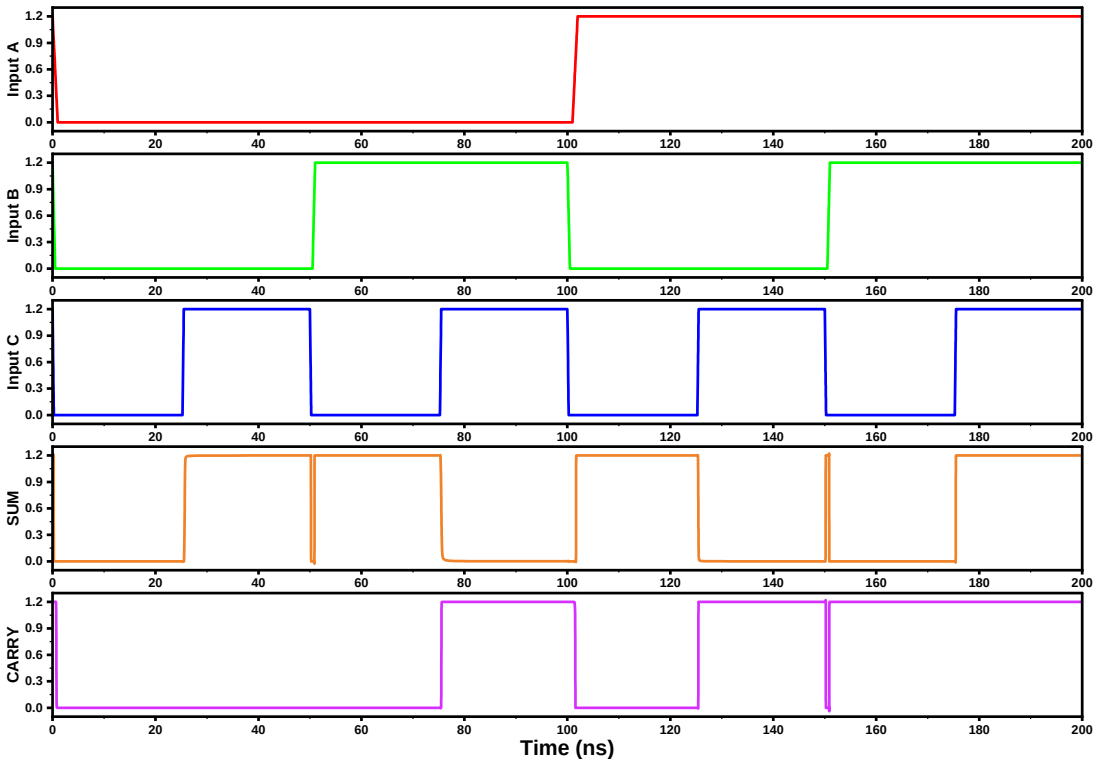

Figure 4.7: Output Waveform for FA-14T. [10]

4.1.2.2 Full Adder 2: FA-14T

The conventional 28T CMOS adder functionality can also be realized using a Hybrid CMOS-PTL logic-based FA-14T without compromising the output voltage levels. The FA-14T employs a two-stage XOR with PTL and AND-OR logic using CMOS, ensuring no effect on noise margin and rail-to-rail swing at the output, similar to the conventional CMOS adder. The FA-14T occupies an area of 9.38 μm^2, approximately 2.25× smaller compared to a conventional 28T full adder at the 65nm CMOS technology node. The schematic and simulation waveforms for the proposed FA-14T are depicted in Fig. 4.6 and Fig. 4.7, respectively. The stand-alone use of FA-14T does not provide significant benefits over SoTA RCA-based adder tree architectures. However, the proposed power-gated CSkA eliminates the need for interleaved or complementary adder tree structures [16, 29, 30], with FA-14T, thus achieving reduced area and delay as evident from Fig. 4.8.

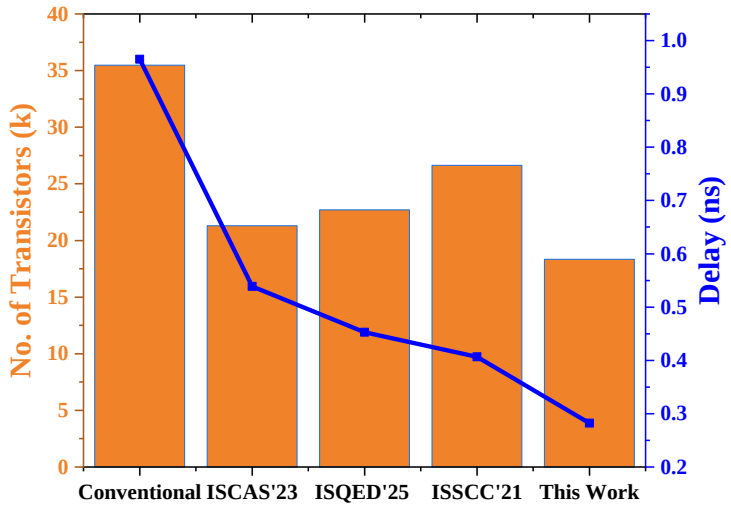

Figure 4.8: Comparison of Adder Tree Structures. [10]

4.2 Performance Analysis

The resource-efficient DARE macro contributes collectively to the enhanced throughput and compute density. The key factors in the power reduction are the ST dual-operation PIM array and the power-gated CSkA-based adder tree. In contrast, the reduced area/delay is attributed to the novel CSkA adder tree. The data flow is programmed using a row-wise weight-stationary approach with the controller. The weight precision can be scaled in the ratio of 4-bit precision, utilizing multiple memory banks. The Variable precision inputs (1-16 bits) are fed bit-serially starting with LSB, producing partial products every clock cycle, and can be controlled dynamically as and when required, such as N computation cycles for N-bits. 64 DPIM banks correspond to 64 MAC units operating parallel, and the partial products are accumulated with 12-bit accumulation. This can be illustrated with Fig. 4.9, where A7T bitcell array computes 1A4W multiplication, with adder tree single-cycle accumulation.

4.2.0.1 DNN Performance Evaluation

The proposed macro is part of a sub-system being developed under the Smart-City Project to enable Surveillance and Object Detection, Facial Recognition and Person

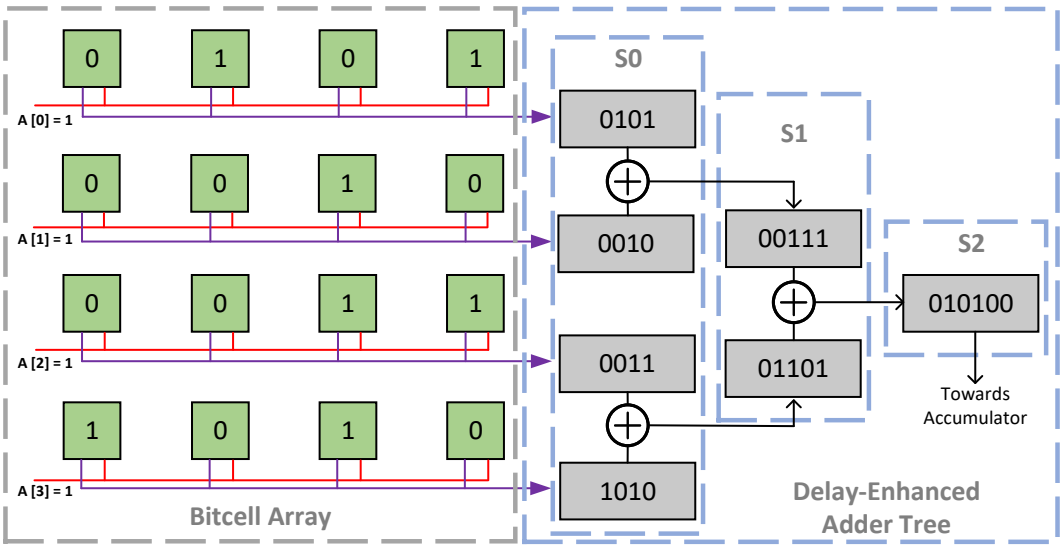

Figure 4.9: Illustration of 1A4W MAC Computation with the Proposed DPIM Macro. [10]

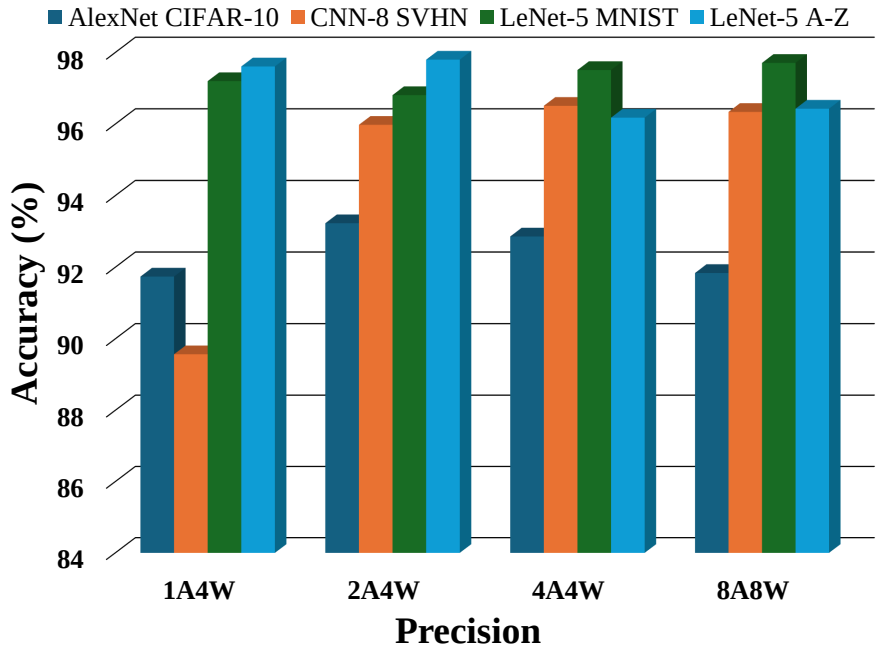

Figure 4.10: Comparison for Inference Accuracy for the Proposed PIM Macro. [10]

Tracking, Traffic Monitoring/Vehicle theft detection, etc. The key goal is to enable resource-efficient real-time classification for vehicles, pedestrians, and suspicious objects. Thus, we targeted the implementation of LeNet-5, AlexNet, and Custom CNN-8 models for MNIST, CIFAR-10, SVHN, and A-Z alphabet recognition datasets.

We have provided the application accuracy with a parameterized software evaluation model in Fig. 4.10. The model is built with $QKeras$, $FxP-math$ library and $Python$ 3.0 on the $Google\ Colab$ platform. The weights are extracted in CSV format and fed to DARE macro, post-training at 1A4W, 2A4W, 4A4W, and 8A8W precision. The custom VGG-like model with 45-class SVCs is evaluated for edge AI applications and achieves satisfactory accuracy up to 95% and 1.7× reduced power consumption. The accuracy evaluation demonstrates significant 98.5% QoR compared to FP32 and stays within 0.5-1% compared to SoTA works [17, 22, 30, 33]. We observed maximum enhancements up to 2.85× energy efficiency and 3.2× compute density when 65% sparsity is exploited with an accuracy loss of 0.5-1%.

4.2.0.2 Performance Metric Analysis

To evaluate computational trade-offs and maximize the proposed system's performance, compute density signifies higher throughput within a compact silicon footprint. At the same time, energy efficiency highlights the system's ability to handle AI workloads without rapidly depleting the emergency battery. We defined the hardware performance emulation model based on the following design parameters: DPIM array dimensions, activation and weight precision, memory banks used per layer execution, and reconfigurable hardware multiplexing. DARE macro is accessed by reprogramming dense weights layer-wise, followed by an off-chip bit-serialized ReLU activation function and Control engine circuitry similar to our work [1]. The off-chip components include a combination accumulator, sub-sampling layer, softmax non-linear layer, and normalization are optimized to support enhanced performance, with sparsity-aware layer-reuse philosophy, maintaining kernel-wise modularity and macro scalability.

The implications of macro size, precision, and throughput for the proposed DPIM macro compared to SoTA works are discussed in Table 4.2 to quantify the impact of this work. The proposed 16kb macro achieves 5.2×, 3.6×, and 1.3× improvement in throughput compared to prior works [1, 19, 33], respectively. The significant throughput improvements primarily come from enhanced operating

Table 4.2: Performance Comparison with SoTA APIM and DPIM Macros

Parameter	JSSC'21 [33]	JETCAS'22 [17]	TNANO'23 [19]	TCAS'23 [18]	TCAS'23 [14]	JSSC'25 [34]	DATE'24 [20]	ISQED'25 [1]	Ours
Tech. (nm)	65	65	65	65	28	28	65	65	65
MAC Operation	Digital	Analog	Digital	Digital	Analog	Digital	Digital	Digital	Digital
Cell Type	NA	AND8T	10T	7T	10T	Compact 8T	6T+2T	8T	A7T
Supply Voltage (V)	0.6-0.8	1	1.2	0.8-1.1	0.9	0.54-0.9	0.6-1.2	1.2	0.8-1.2
Array Size	16Kb	16Kb	16Kb	80Kb	8Kb	16Kb	4Kb	16Kb	16Kb
Op. Frequency (MHz)	138	100	25	200	NA	230	40	250	250
Bitcell Area (μm^2)	10.53	2.78	4.5	2.83	0.623	NA	2.25	4.1	2.23
Macro Area (mm^2)	0.23	NA	NA	0.473	0.051	0.028	0.365	NA	0.346
Input Precision (bit)	1-16	4	4	1-16	1-4	2-8	4/8	1-8	1-16
Weight Precision (bit)	1-16	4	1-4	1-16	1-4	2-8	4/8/12/16	4/8	4/8/12/16
Model	LeNet-5	VGG-8	CNN-Type	Inception V4	VGG-Like	ResNet-20	NA	LeNet-5	AlexNet
Accuracy	99.2	96.05	98.67	95.3	85.7	91.74	98.5	99.1	98.64
Throughput (TOPS)	0.56	0.42	0.8	0.41‡	1.08	0.122	2.52‡	2.2	2.89
Energy Efficiency (TOPS/W)	156	180	273	63	311.2	162	404	480	514
Compute Density (TOPS/mm^2)	2.5	NA	NA	0.86	10.06	4.4	6.92	7.92	8.34

Note: Scaled throughput‡ for 16Kb of proposed DPIM macro, for a fair comparison with 16Kb Macros, CMOS 65nm [17–20].

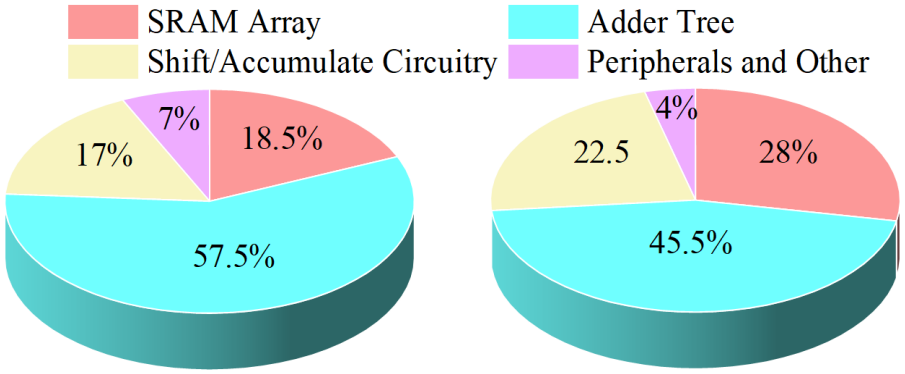

Figure 4.11: Area Breakdown of the Proposed PIM Macro. [10]

frequency, leading to more operations/cycles. The circuit-level simulation was done on Cadence Virtuoso and Mentor Graphics Calibre, with the software framework (NN mapping and accuracy) done using Python on Google Colab. The area and energy breakdown of the proposed DPIM macro at the CMOS 65nm Technology node is illustrated in Fig. 4.11. We have also provided the area and power consumption for off-chip components in Fig. 4.12. The primary analysis focused on energy efficiency (TOPS/W) and compute density (TOPS/mm^2) assists in accessing the computational power of the device. We also observed improvements in energy efficiency up to 3.2×,

2.85×, 1.8×, 1.2× compared to prior works [1, 17, 19, 33] respectively. We have also discussed the impact of supply voltage on energy efficiency in Fig. 4.13. Lower energy consumption is primarily driven by design optimization, which includes reduced power consumption and sparse workload execution, in addition to enhanced throughput. This is a key factor considering the operational life of resource-constrained devices.

Component	Area(μm²)	Power(μW)
Accumulator	806	177.76
ReLU (AF)	429	113.68
Max Pooling	988	121.28
Batch Norm.	523	135.17
Quantization	78	23.51

Figure 4.12: Energy Breakdown of Off-chip Components. [10]

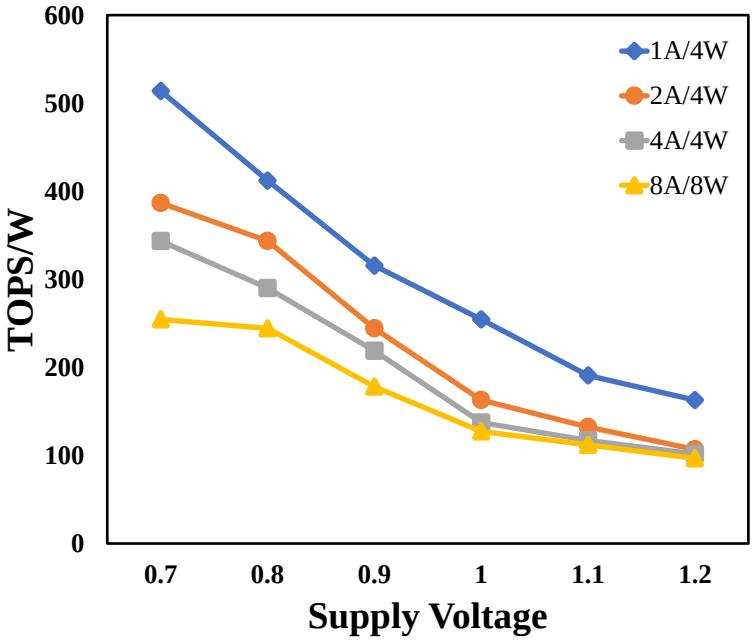

Figure 4.13: Performance Analysis of the Proposed PIM Macro in TOPS. [10]

The proposed DPIM macro achieves 3.2×, 1.85× and 1.1× enhanced compute density compared to prior works [1, 30, 33] respectively. This is primarily driven by

area efficiency, achieved through resource-efficient adder trees and reduced macro utilization due to sparse workload processing. This signifies a compact, cost-effective design with reduced silicon area.

4.3 Conclusion

This work introduces the resource-efficient performance-enhanced DARE macro, addressing the area-latency overheads of bulky adder tree architectures while reducing power consumption in the PIM array. The proposed approach utilizes an area-efficient AND7T (A7T) bitcell, integrating a low voltage 5T SRAM cell with 2T bit-wise multiplication in the PIM array, enhancing an energy efficiency up to 2.85×. Additionally, a delay, energy, and area-optimized interleaved adder tree, constituted with power-gated Carry Skip Adder (CSkA) with FA-7T and FA-14T, increases compute density by 3.2× compared to SoTA works. Moreover, skipping 65% of sparse computations in AI workloads helps reduce operation cycles by 27% while maintaining accuracy within 0.5–1% of prior approaches. We provide a detailed evaluation of AlexNet, LeNet-5, and custom multiclass classification models. Overall, the proposed solution balances available resource efficiency and computational performance, making it highly suitable for resource-constrained Edge-AI devices.

Chapter 5

Conclusion & Future Scope

This thesis presents digital compute-in-memory (DCIM) macro architectures such as RAPID-CIM, DARE, and a 2D Interleaved Adder Tree-based DCIM macro targeted at improving energy efficiency, compute density, and area utilization for compact, battery-powered Edge-AI devices.

The 2D Interleaved Adder Tree-based DCIM macro features an 8T SRAM bitcell that resolves the bit-flipping issue caused by multi-row activation and a novel 7T full adder that enables 40% transistor count reduction. The macro achieves a throughput of 2.2 TOPS at 250 MHz and 1.2 V on 65nm CMOS, providing 2× energy efficiency improvement over state-of-the-art solutions.

The RAPID-CIM architecture introduces a reconfigurable and energy-efficient SRAM-based DCIM macro, utilizing a novel M8T bitcell and scalable bit-precision reconfigurable adder tree. Compared to state-of-the-art designs, it achieves up to 16.7× energy efficiency, 8.6× compute density, and 3.2× reduction in operation cycles.

The DARE macro addresses area-latency overheads in adder trees by employing an area-efficient AND7T (A7T) bitcell (5T SRAM + 2T multiplier). It integrates a power-gated Carry Skip Adder (CSkA) with FA-7T and FA-14T cells, enabling 2.85× energy efficiency, 3.2× compute density improvement, and 27% reduction in operation cycles through 65% sparse computation skipping, all while maintaining application accuracy within 0.5–1% of prior works.

The macros were evaluated on AlexNet, LeNet-5, and custom multiclass classification models, demonstrating accuracy of 98.7%, 98.8% (1A4W) and 99.1%, 97.8% (4A4W) using MNIST and A-Z alphabet datasets. Collectively, these DCIM designs strike a balance between energy efficiency, compute density, area optimization, and application accuracy, making them highly suitable for resource-constrained edge-AI platforms.

While the proposed DCIM macros demonstrate significant improvements in energy efficiency and compute density, analysis reveals that the adder tree alone accounts for over 45% of the total power consumption and occupies nearly 60% of the overall area within the DCIM macro. This high resource overhead reduces

effective CIM memory utilization, limiting it to approximately 30% compared to the original SRAM array, thereby replacing a substantial portion of the actual storage memory. As a result, further optimization of the adder tree architecture is critical to reclaim memory footprint, reduce energy bottlenecks, and improve overall macro efficiency. Future efforts shall focus on developing ultra-low-power, area-compact, and computation-aware accumulation designs to enhance the scalability and practicality of DCIM solutions for edge-AI applications.

List of Publications

- **Akash Sankhe**, Mukul Lokhande, Radheshyam Sharma, Santosh Kumar Vishvakarma, "Area-optimized 2D Interleaved Adder Tree Design for Sparse DCIM Edge Processing", *In 2025 26th International Symposium on Quality Electronic Design (ISQED)*, vol. 26, pp. 1-6, Dec 2025. **(Accepted)**.

- **Akash Sankhe**, Narendra Dhakad, Radheshyam Sharma, Mukul Lokhande, Santosh Kumar Vishvakarma, "RAPID: Re-configurable Adder Tree based Performance-Incentivized AI Digital CIM Macro", *IEEE Transactions on Nanotechnology, 2025* **(Under Review)**.

- **Akash Sankhe**, Mukul Lokhande, Adam Teman, Santosh Kumar Vishvakarma, "**DARE**: **D**elay and **A**rea-optimized **R**econfigurable **E**dge DPIM Macro", *IEEE Transactions on Circuits and Systems Part II: Express Briefs, 2025* **(Manuscript Under Preparation)**.

Bibliography

[1] A. Sankhe *et al.*, "Area-optimized 2D Interleaved Adder Tree Design for Sparse DCIM Edge Processing", *In 2025 26th International Symposium on Quality Electronic Design (ISQED)*, vol. 26, pp. 1–6, Dec 2025.

[2] A. Sankhe, N. S. Dhakad, R. Sharma, M. Lokhande, and S. Kumar, "Rapid: Re-configurable adder tree based performance-incentivized ai digital cim macro", *Authorea Preprints*, Mar 2025.

[3] H. Kim, T. Yoo, T. T.-H. Kim, and B. Kim, "Colonnade: A reconfigurable sram-based digital bit-serial compute-in-memory macro for processing neural networks", *IEEE Journal of Solid-State Circuits*, vol. 56, no. 7, pp. 2221–2233, 2021.

[4] C.-T. Lin, D. Wang, B. Zhang, G. K. Chen, P. C. Knag, R. K. Krishnamurthy, and M. Seok, "Dimca: An area-efficient digital in-memory computing macro featuring approximate arithmetic hardware in 28 nm", *IEEE Journal of Solid-State Circuits*, vol. 59, no. 3, pp. 960–971, 2023.

[5] V. Sharma, J.-E. Kim, H. Kim, L. Lu, and T. T.-H. Kim, "A reconfigurable 16kb and8t sram macro with improved linearity for multibit compute-in memory of artificial intelligence edge devices", *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 2, pp. 522–535, 2022.

[6] 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*, vol. 54, no. 1, pp. 217–230, 2018.

[7] Y.-D. Chih, P.-H. Lee, H. Fujiwara, Y.-C. Shih, C.-F. Lee, R. Naous, Y.-L. Chen, C.-P. Lo, C.-H. Lu, H. Mori, *et al.*, "16.4 an 89tops/w and 16.3 tops/mm 2 all-digital sram-based full-precision compute-in memory macro in 22nm for machine-learning edge applications", in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, pp. 252–254, 2021.

[8] W. Yi, K. Mo, W. Wang, Y. Zhou, Y. Zeng, Z. Yuan, B. Cheng, and B. Pan, "Rdcim: Risc-v supported full-digital computing-in-memory processor with high energy efficiency and low area overhead", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 4, pp. 1719–1732, 2024.

[9] J. Oh, C.-T. Lin, and M. Seok, "D6cim: 60.4-tops/w, 1.46-tops/mm 2, 1005-kb/mm 2 digital 6t-sram-based compute-in-memory macro supporting 1-to-8b fixed-point arithmetic in 28-nm cmos", in *ESSCIRC 2023-IEEE 49th European Solid State Circuits Conference (ESSCIRC)*, pp. 413–416, 2023.

[10] A. Sankhe, M. Lokhande, A. Teman, and S. K. Vishvakarma, "Dare: Delay and area-optimized reconfigurable edge dpim macro", *Authorea Preprints*, Mar 2025.

[11] O. Krestinskaya, L. Zhang, and K. N. Salama, "Towards Efficient In-Memory Computing Hardware for Quantized Neural Networks: State-of-the-Art, Open Challenges and Perspectives", *IEEE Transactions on Nanotechnology*, vol. 22, pp. 377–386, Jul 2023.

[12] M. Lokhande, G. Raut, and S. K. Vishvakarma, "Flex-pe: Flexible and simd multi-precision processing element for ai workloads", *IEEE Transactions on Very Large Scale Integration Systems*, Dec. 2025.

[13] Z. Wang *et al.*, "A dual-domain compute-in-memory system for general neural network inference", *Nature Electronics*, pp. 1–12, Jan 2025.

[14] L. Lu and D. A. Tuan, "A 47 TOPS/W 10T SRAM-Based Multi-Bit Signed CIM With Self-Adaptive Bias Voltage Generator for Edge Computing Applications", *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, pp. 3599–3603, May 2023.

[15] V. Sharma, X. Zhang, N. S. Dhakad, and T. T.-H. Kim, "FlexDCIM: A 400MHz 249.1TOPS/W 64Kb Flexible Digital Compute-in- Memory SRAM Macro for CNN Acceleration", *IEEE Transactions on Circuits and Systems-I*, 2025.

[16] Y. D. Chih *et al.*, "An 89TOPS/W and 16.3 TOPS/mm^2 all-digital SRAM-based full-precision compute-in memory macro in 22nm for machine-learning edge applications", in *ISSCC*, vol. 64, pp. 252–254, 2021.

[17] V. Sharma *et al.*, "A reconfigurable 16Kb AND8T SRAM macro with improved linearity for multi-bit compute-in memory of artificial intelligence edge devices", *IEEE JETCAS*, vol. 12, no. 2, pp. 522–535, 2022.

[18] H. Kim, J. Mu, C. Yu, T. T.-H. Kim, and B. Kim, "A 1-16b reconfigurable 80kb 7t sram-based digital near-memory computing macro for processing neural networks", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, pp. 1580–1590, Jan 2023.

[19] P. K. Saragada *et al.*, "A configurable 10T SRAM-based IMC accelerator with scaled-voltage-based pulse count modulation for MAC and high-throughput XAC", *IEEE Trans. on Nanotechnology*, vol. 22, pp. 222–227, 2023.

[20] P. Tyagi *et al.*, "A 101 TOPS/W and 1.73 TOPS/mm2 6T SRAM-Based Digital Compute-in-Memory Macro Featuring a Novel 2T Multiplier", in *DATE*, 2024.

[21] W. Yi *et al.*, "RDCIM: RISC-V supported full-digital computing-in-memory processor with high energy efficiency and low area overhead", *IEEE Trans. on Circuits and Systems I*, vol. 71, no. 4, pp. 1719–1732, 2024.

[22] A. Biswas *et al.*, "CONV-SRAM: An energy-efficient SRAM with in-memory dot-product computation for low-power convolutional neural networks", *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, 2019.

[23] Z. Wang, H. Luo, Z. Peng, X. Chao, and Y. He, "An 8t sram based digital compute-in-memory macro for multiply-and-accumulate accelerating", in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2023.

[24] G. Raut, S. Karkun, and S. K. Vishvakarma, "An empirical approach to enhance performance for scalable cordic-based deep neural networks", *ACM Transactions on Reconfigurable Technology and Systems*, vol. 16, no. 3, pp. 1–32, 2023.

[25] P. K. Saragada, S. Manna, A. Singh, and B. P. Das, "A configurable 10t sram-based imc accelerator with scaled-voltage-based pulse count modulation for mac and high-throughput xac", *IEEE Transactions on Nanotechnology*, vol. 22, pp. 222–227, 2023.

[26] G. Raut, S. Karkun, and S. K. Vishvakarma, "An empirical approach to enhance performance for scalable CORDIC-based Deep Neural Networks", *ACM Trans. on Reconfigurable Technology and Systems*, vol. 16, no. 3, pp. 1–32, 2023.

[27] B. H. Calhoun and A. P. Chandrakasan, "A 256-kb 65-nm sub-threshold sram design for ultra-low-voltage operation", *IEEE Journal of Solid-State Circuits*, vol. 42, pp. 680–688, Mar 2007.

[28] A. Teman, A. Mordakhay, J. Mezhibovsky, and A. Fish, "A 40-nm sub-threshold 5t sram bit cell with improved read and write stability", *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, pp. 873–877, Jan 2012.

[29] Z. Wang *et al.*, "An 8T SRAM Based Digital Compute-In-Memory Macro For Multiply-And-Accumulate Acceleration", in *ISCAS*, pp. 1–5, 2023.

[30] C. T. Lin *et al.*, "DIMCA: An Area-Efficient Digital In-Memory Computing Macro featuring Approximate Arithmetic Hardware in 28 nm", *IEEE Journal of Solid-State Circuits*, 2024.

[31] C. He, Z. Wang, F. Xiang, Z. Dai, Y. He, J. Yue, and Y. Liu, "Lsac: A low-power adder tree for digital computing-in-memory by sparsity and approximate circuits co-design", *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, pp. 852–856, Aug 2024.

[32] V. Sharma, X. Zhang, N. S. Dhakad, and T. T.-H. Kim, "Flexdcim: A 400 mhz 249.1 tops/w 64 kb flexible digital compute-in-memory sram macro for cnn acceleration", *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–12, Mar 2025.

[33] H. Kim *et al.*, "Colonnade: A reconfigurable SRAM-based Digital bit-serial Compute-in-memory macro for processing neural networks", *IEEE Journal of Solid-State Circuits*, vol. 56, no. 7, pp. 2221–2233, 2021.

[34] H. Diao *et al.*, "A multiply-less approximate sram compute-in-memory macro for neural-network inference", *IEEE Journal of Solid-State Circuits*, vol. 60, pp. 695–706, Aug 2025.