KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Low energy Digital In Memory Compute techniques for edge AI applications.

Weijie Jiang

Supervisors:
Prof. dr. ir. Wim Dehaene
Prof. dr. ir. Marian Verhelst

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

April 2025

Low energy Digital In Memory Compute techniques for edge AI applications.

Weijie JIANG

Examination committee:
Prof. dr. ir. Jean-Pierre Celis, chair
Prof. dr. ir. Wim Dehaene, supervisor
Prof. dr. ir. Marian Verhelst, supervisor
Prof. dr. ir. Nele Mentens
Prof. dr. ir. Kris Myny
Prof. dr. Joachim Rodrigues
(Far Away)
Mr. Prabhat Avasare
(Far Away)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

April 2025

Use of Generative AI
I did use the generative AI (ChatGPT, Deepseek) assistance tools. The tools are used to help me make abstraction, text clean-up, figure generation, and bib-tex export. They are also used generate the table from image, formula from image. The text, code, and images in this thesis are my own (unless otherwise specified). Generative AI has only been used in accordance with the KU Leuven guidelines and appropriate references have been added. I have reviewed and edited the content as needed and I take full responsibility for the content of the thesis.

Abstract

Edge AI applications demand ultra-low-power and efficient computing solutions that can handle increasingly complex workloads under strict energy and area constraints. Traditional Von Neumann architectures are fundamentally limited by the memory wall, leading to excessive energy consumption due to frequent data movement between memory and processing units. In response, In-Memory Compute (IMC) architectures—particularly Digital-In-Memory Compute (DIMC)—have emerged as promising alternatives.

This thesis investigates low-energy DIMC techniques tailored for edge AI, addressing key design challenges across the circuit, architecture, and system levels. The work begins with a systematic analysis of analog versus digital IMC, revealing the limitations of analog approaches in precision, scalability, and robustness. Subsequently, the design space of DIMC is explored in depth, with an emphasis on memory cell choices, macro-level architecture, and trade-offs between area and energy efficiency.

Two silicon-proven DIMC macro prototypes are presented, demonstrating effective techniques such as vertical p-sub layout, reverse precharge, and gray-code decoders for enhanced area and power efficiency. Building upon these, a DIMC-based system-on-chip (SoC), HUNBN, is designed for edge convolutional neural networks (CNNs). The SoC introduces innovations including split MAC workflows, efficient dataflow for sparse CNNs, and precision-scalable accumulation strategies.

Measurement results confirm that the proposed DIMC architectures offer state-of-the-art energy efficiency (up to 23.8 TOPs/W) with compact SRAM area usage, outperforming prior art across multiple metrics. This work paves the way for scalable, energy-efficient edge intelligence using digital in-memory compute.

List of abbreviations

IMC ..... *In Memory Compute*
SRAM ..... *Static Random-Access Memory*
ADC ..... *Analog Digital Converter*
MRAM ..... *Magnetoresistive Random-Access Memory*
RRAM ..... *Resistive Random-Access Memory*
PCM ..... *Phase Change Memory*
WL ..... *Word Line*
BL ..... *Bit Line*
IA ..... *Input Activation*
MAC ..... *Multiply and ACcumulate*
PE ..... *Processing Element*
CMOS ..... *Complementary Metal-Oxide-Semiconductor*
DIMC ..... *Digital In Memory Compute*
6T ..... *6-Transistor bit cell*
8T ..... *8-Transistor bit cell*
10T ..... *10-Transistor bit cell*
DRC ..... *Design Rule Check*
FoM ..... *Finger of Merit*
PSUM ..... *Partial SUM*
DNN ..... *Deep Neural Network*
INT ..... *INTeger data type*
FP ..... *Floating Point data type*
DVFS ..... *Dynamic Voltage Frequency Scaling*
PVT ..... *Process Voltage Temperature*
SoC ..... *System on Chip*
VDD ..... *Voltage supply*
PMOS ..... *P-type MOSFET*
NMOS ..... *N-type MOSFET*
MUX ..... *Multiplexer*
DL ..... *Deep Learning*

CNN ..... *Convolution Neural Network*
TOPs/W ..... *Tera OPerations per second per Walt*
GOPs ..... *Giga OPerations per second*
Kfps ..... *Kilo frame per second*

List of Symbols

Φ ..... *Control signal for timing*
re ..... *Read enable signal*
pe ..... *Precharge enable signal*
V_r ..... *Precharge voltage*
$\oplus$ ..... *XOR operation*

Contents

List of Figures

List of Tables

Chapter 1

In-Memory-Compute, from analog to digital

Complementary Metal-Oxide-Semiconductor (CMOS) technology has been one of the most transformative advancements in electronics, profoundly shaping nearly every aspect of our daily lives. Introduced in the 1960s, CMOS technology has become the foundation for modern integrated circuits, enabling the production of energy-efficient, compact, and high-performance microchips [62, 19]. From the microprocessors in our smartphones and laptops to the sensors in our smart home devices, CMOS technology has driven the miniaturization and affordability of electronic systems, making computing power accessible to billions of people worldwide.

Over the past few decades, significant advancements in transistor downscaling and technological innovations have allowed computing performance to keep up with the growing demands of modern applications [72, 63]. These improvements have been instrumental in enabling large-scale applications, such as Google searches and AI models like ChatGPT, by providing more powerful processing cores and greater on-chip memory capacity.

In addition to powering large-scale AI applications, technological advancements have also paved the way for a proliferation of edge AI applications. For example, modern cameras are now equipped with built-in edge detection algorithms capable of identifying objects in real time, enabling smarter and more responsive distributed intelligence systems [70, 41]. Similarly, in manufacturing, advanced robots integrated with edge AI capabilities are increasingly employed to streamline production processes, automate repetitive tasks, and enhance

overall efficiency [84]. These developments not only improve productivity but also enable more adaptive and intelligent systems across various industries, from smart surveillance to precision manufacturing.

Unlike large-scale AI applications, edge applications face a more challenging problem: how to compute complex and large AI models on edge devices within a stringent power budget [105]. For instance, autonomous drones require real-time processing of high-resolution video streams to identify and track objects while operating on limited battery capacity [53]. Similarly, wearable health devices must continuously analyze physiological data using sophisticated AI algorithms, all while ensuring prolonged battery life to avoid frequent recharging. These examples highlight the critical need for energy-efficient solutions in edge AI applications.

In fact, the main issue lies in the fact that, while transistor sizes have continued to shrink, the quality and efficiency of metal interconnects have not kept pace, limiting the energy scaling of on-chip data movement [81]. Such limitations are inherent to the traditional Von Neumann architecture, which relies on a clear separation between memory and computation units [65]. To address this fundamental constraint, In-Memory Compute (IMC) technology has emerged as a promising solution, as it integrates computation directly within memory instances to minimize data movement and improve energy efficiency [73, 82]. Since 2020, considerable research efforts have been directed toward developing and refining IMC technology.

This chapter will first provide a historical overview of IMC, exploring how it has evolved from conventional computing architectures. Next, it will discuss the limitations of current IMC designs, with a particular focus on comparing Analog-In-Memory Compute (AIMC) and Digital-In-Memory Compute (DIMC) approaches. Finally, the research content and objectives of this thesis will be briefly introduced.

1.1 The Von-Neumann architecture and the memory wall problem

1.1.1 The Von-Neumann architecture

The majority of computing systems are built upon the Von Neumann architecture. The Von Neumann architecture, as shown in Fig.1.1, is a computer system design that consists of a CPU, memory, and input/output components, all connected via a shared communication bus. In this architecture, both

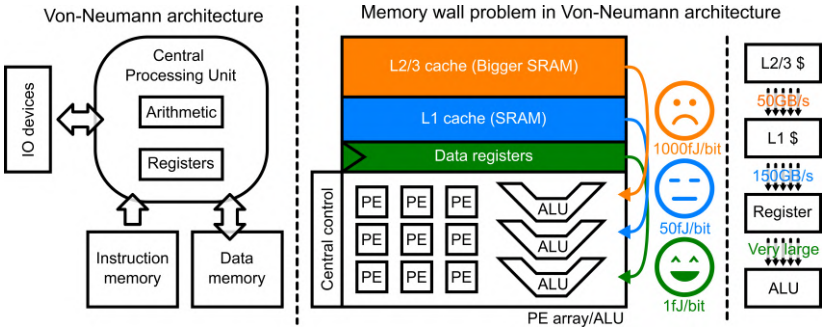


Figure 1.1: Illustration of the Von-Neumann architecture and the memory wall problems

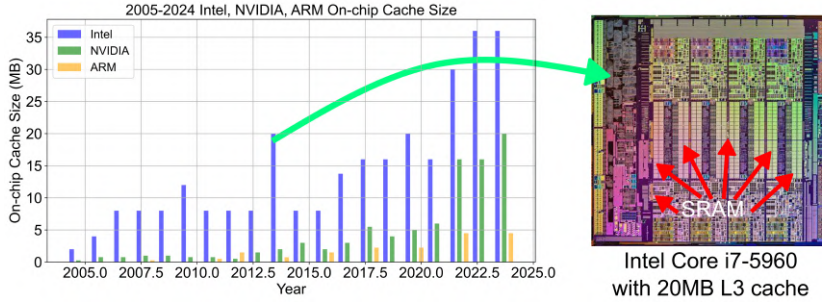


Figure 1.2: Evolution of on-chip cache size for Intel, Nvidia, and Arm CPU/GPU from year 2005 to 2024. [55]

program instructions and data are stored in the same memory space and accessed sequentially by the CPU. The CPU fetches instructions and data from memory, executes operations, and writes results back to memory, all through a unified data and address bus.

The Von Neumann architecture has performed exceptionally well over many years, largely due to its design simplicity due to its flexibility and programmability. However, in modern computing systems, one major challenge arises from the continuous growth of on-chip memory size, which places significant strain on the shared bus. This growth is evident in both commercial and academic chips.

Fig.1.2 illustrates the trend in L3 cache size across Intel CPUs, NVIDIA GPUs, and ARM CPUs from 2005 to 2024. The figure clearly highlights a trend:

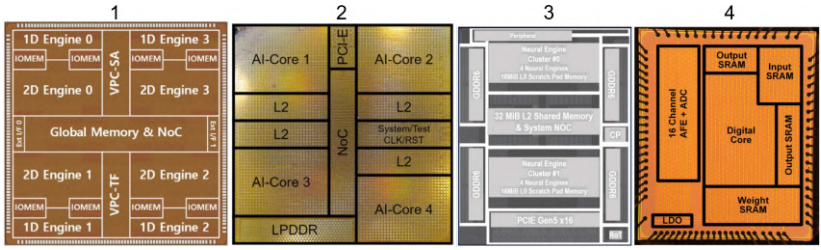


Figure 1.3: Example die photos from recent ISSCC publications. From 1-4 are [34], [118], [104], [33].

designers are increasingly integrating larger L3 caches into computing platforms, whether for desktop and laptop CPUs, embedded system CPUs, or GPUs. The rising cache sizes reflect the industry's response to the growing demands of data-intensive applications. Fig.1.2 includes a die photo of an Intel Core i7-5960 processor, which showcases the substantial area dedicated to SRAM on a commercialized processor chip.

From an academic perspective, the memory subsystem is also arguably the most critical component of modern integrated circuits. As depicted in Fig.1.3, which presents die photos from four recent ISSCC papers, a significant portion of chip area is allocated to SRAM. This extensive allocation underscores the crucial role of on-chip memory in achieving high performance and energy efficiency.

1.1.2 The memory wall problem

For computing system with large (relative to the chip area) on-chip cache, however, the Von Neumann architecture suffers from the memory wall problem. Two primary issues contribute to the memory wall problem.

- First, as cache size increases, the available data bandwidth does not increase proportionally. This limitation poses a major challenge for the PE arrays, which require efficient interconnections to manage data flow without exceeding bandwidth constraints. An effective strategy for addressing this issue is the implementation of systolic arrays, which optimize data movement by structuring computation in a highly parallel and pipelined manner. However, while this approach helps mitigate bandwidth bottlenecks, it comes at the cost of reduced system flexibility, as the interconnect design must be highly specialized to accommodate

specific computational patterns. Fortunately, most large AI models have rather regular computation patterns[89].

- Second, there is a notable loss in energy efficiency as the cache hierarchy moves to higher levels. As illustrated in Fig.1.1, while registers can supply data at an energy cost of just 1 fJ/bit, accessing data from L1 cache—typically implemented with SRAM—incurs a cost of approximately 50 fJ/bit. For much larger L2 and L3 caches, the energy cost can easily escalate to 1000 fJ/bit or more. This substantial increase in energy expenditure for data retrieval can significantly undermine the overall energy efficiency of the system. While advanced caching mechanisms can partially address the energy challenges in general CPU applications, *AI workloads present a more direct and fundamental issue: these applications demand substantial amounts of model data to be readily available near the computing units, necessitating direct access to large on-chip memory. This reliance on extensive memory resources exacerbates energy efficiency losses associated with traditional memory hierarchies, making it difficult to optimize power consumption effectively.* AI workloads present a more direct and fundamental issue: these applications demand near-real-time access to massive volumes of model parameters and intermediate results, which cannot be efficiently cached due to their sheer size and sparsity. Consequently, reliance on traditional memory hierarchies—which are optimized for localized data reuse rather than bulk memory access—creates critical energy bottlenecks.

For the Von Neumann architecture, the challenges of energy efficiency and memory bandwidth differ significantly between large AI applications and edge AI applications. For large-scale AI workloads running on GPU clusters, energy constraints are indeed relaxed due to the high computational throughput justifying power consumption. Memory bandwidth limitations are mitigated through advanced DRAM technologies like GDDR and HBM [2, 3], which provide terabytes-per-second bandwidth to sustain data-intensive tasks. In contrast, edge AI applications deployed on MCUs face **diametrically opposite constraints**:

- **Memory bandwidth is less critical** due to smaller model sizes and localized data processing.
- **Energy efficiency becomes the overriding priority**, as devices operate on batteries with strict power budgets (e.g., <10W for wearables).

To address this, advanced circuit techniques are increasingly employed to enhance energy efficiency by reducing data movement and optimizing on-chip processing, such as IMC.

1.2 Conquering the memory wall – from more registers to In-Memory-Compute

Researchers have been striving to overcome the memory wall by integrating more memory cells locally near computing units. One prominent trend is the expansion of on-chip register capacity. Early 32-bit Intel processors featured only 8 general-purpose registers, which doubled to 16 with the advent of 64-bit architecture [17].

Advancements in SIMD (Single Instruction, Multiple Data) technologies further expanded this capacity:

- **SSE (Streaming SIMD Extensions)**: Introduced 8 `XMM` registers (128-bit wide) for parallel processing of data (e.g., video encoding).
- **AVX (Advanced Vector Extensions)**: Doubled the register width to 256-bit (`YMM` registers), enabling faster calculations for AI inference tasks.
- **AVX-512**: Further extended to 512-bit (`ZMM` registers), allowing even larger-scale parallel computations (e.g., deep learning model activations).

This progression highlights the industry's effort to incorporate larger and more accessible on-chip register memory, significantly improving energy efficiency in CPUs for handling AI workloads.

In the domain of customized accelerators, where data movement patterns are typically fixed more regular and data structures are relatively simple, dedicated SRAM modules are commonly embedded directly into the datapath. These SRAMs serve as efficient, localized data storage units, reducing dependence on larger memory subsystems. As a result, the boundary between memory subsystems and processing units has increasingly blurred, creating architectures that seamlessly integrate memory into the datapath [83].

A natural question is: why not merge the datapath with the memory subsystem? In response, IMC has been proposed as an alternative approach. The fundamental difference is illustrated in Fig.1.4 using a matrix multiplication workload as an example. In a traditional processing element (PE) array, both input data and weights must be fetched from SRAM. Consequently, the memory

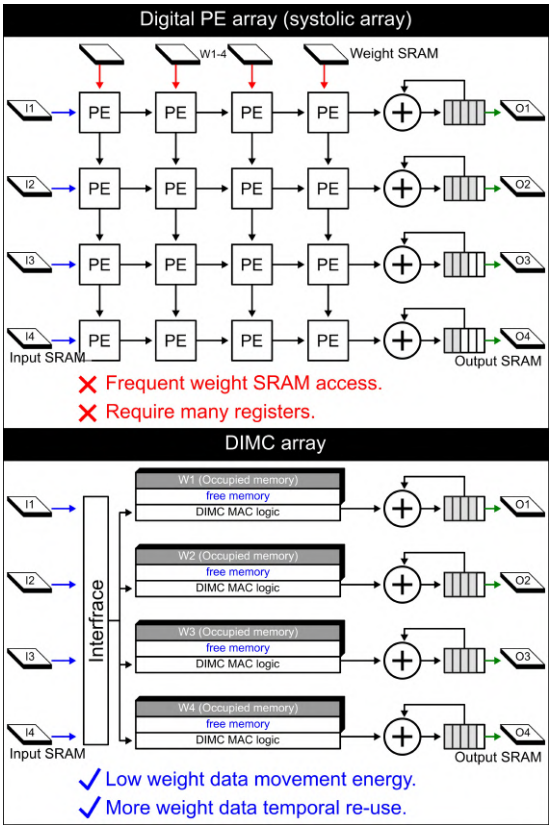


Figure 1.4: Comparison of a PE array and IMC computing array.

subsystem must meet the high energy and bandwidth demands associated with transferring both inputs and weights. In contrast, with IMC, weights are stored locally within the SRAM itself. This design eliminates the need for energy-intensive and bandwidth-consuming weight movement, significantly improving efficiency by reducing data transfer overheads.

1.3 Definition of IMC

In this section, we define the concept of IMC and highlight its key differences from traditional PE arrays. To aid understanding, we first revisit the circuit components of a PE array. A PE element typically consists of two main

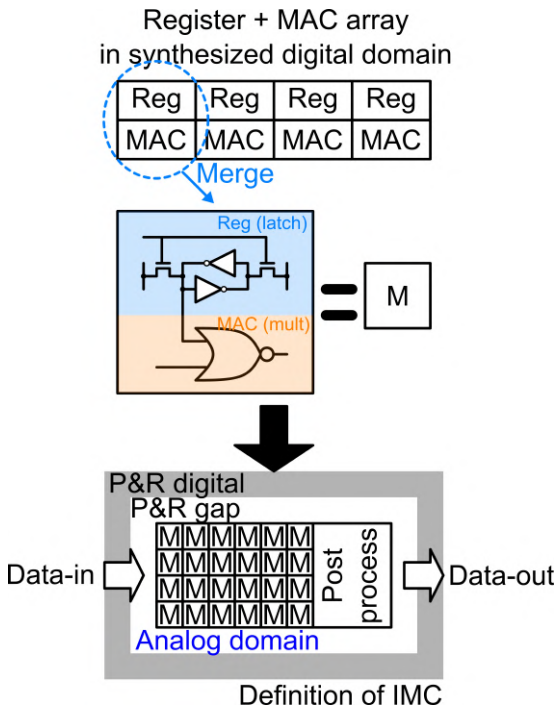


Figure 1.5: Definition of IMC.

components: a computing unit, such as a multiply-and-accumulate (MAC) circuit, and a register to cache input data temporarily. These elements are constructed using gates from the digital standard cell library and are synthesized, placed, and routed (P&R) in the digital design domain, as illustrated in Fig. 1.5. The PE array is essentially an organized arrangement of these PE elements, interconnected in ways tailored for specific computing tasks, such as matrix multiplication or vector processing.

In contrast, IMC rebuilds the smallest computational unit by merging the traditional components of a PE element into a highly optimized structure, referred to as an IMC cell. In an example configuration, the register is replaced with a compact 6T latch cell, while the MAC unit is substituted with a 1-bit multiplier, which drastically simplifies the design. Unlike standard digital cells, the IMC cell is not bound by the height and design constraints typically imposed by digital standard cells, allowing for greater flexibility in layout optimization.

An IMC array is then defined as a structured grid of these IMC cells,

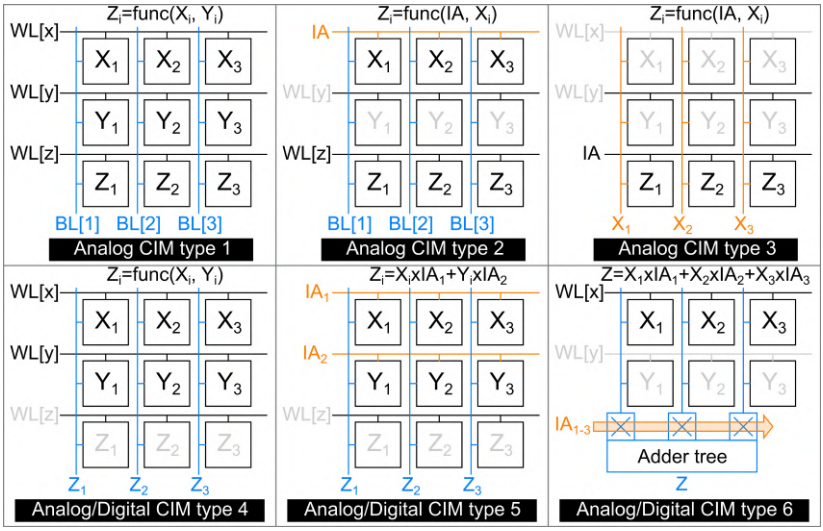


Figure 1.6: Six types of IMC designs.

complemented by additional post-processing circuitry to handle and process the analog outputs of the IMC cells. This setup transforms the IMC into a stream processor. Effectively, the IMC architecture compresses the functionality of numerous digital gates or analog circuit components into a streamlined, specialized macro, achieving substantial gains in energy efficiency and computational density. Based on the computational flow within the IMC array, IMC architectures can be broadly classified into six distinct types.

1.3.1 Six types of IMC

Fig.1.6 categorizes the existing types of IMC architectures. The IMC can be categorized based on the location of the input data and the destination of the output data.

Type 1 IMC employs an analog computation approach, representing true in-memory processing. During computation, multiple word-lines (e.g., n=3) are activated, selecting multiple rows of data from the memory cluster. Computations occur between n-1 words, with the results stored back into the n-th word. Since the written back mechanism happens inside the array, only memory types that support analog writing can be used. As a result, this architecture is commonly used in analog memory technologies, such as RRAM

[80], MRAM [111], PCM [40], and SRAM for applications like Ising model computations [6].

Type 2 IMC simplifies input sourcing by using the word-line (WL) to provide input activations (IA) while still storing results back into the same cell array. The computation occurs between the WL-provided input and data within the cell array. Since this type also requires internal analog writing, this approach is adopted by analog memory cells, including RRAM [77] and MRAM [50].

Type 3 IMC differs by sourcing IAs from the WL and incorporating an additional input, Xi, from the bit-line (BL) for computation. The results are computed directly within the memory cell and stored instantly. Like the previous types, Type 3 IMC leverages analog memory cells for efficient data processing [59, 113, 12].

Type 4 IMC performs computations by simultaneously activating n WLs. The data stored in each word is combined on the BLs, with the output generated and directed outside of the memory array. The computation can occur in either the analog or digital domain. For example, [46] describes an SRAM-based Content-Addressable Memory that opens n=4 WLs per cycle, while [23] presents another SRAM-based general-purpose processor capable of supporting a wide range of operators. Besides SRAM, several Type 4 IMCs also utilize non-volatile memory technologies, as demonstrated in [15, 43, 10].

Type 5 IMC also performs computations by activating n WLs, but it differs from Type 4 in that it applies the IA directly on the WL. The memory cell then produces the product of the IA and the stored value, sending the result on the BL. The accumulation can occur either in the analog domain, using current or charge summation, or in the digital domain, employing customized cells. Type 5 IMC is widely adopted in many non-volatile memory technologies [14, 11, 69] and is also prevalent in SRAM/DRAM-based IMC implementations [114, 44, 5, 9, 101, 108]. However, as discussed in this thesis, Type 5 IMC faces two significant challenges. First, digital Type 5 IMCs require non-standard memory cells, leading to reduced area efficiency. Second, analog Type 5 IMCs suffer from efficiency losses due to the non-idealities inherent in minimized SRAM circuitry.

Type 6 IMC performs computation not inside the memory array. Instead, a word is read on the BL and IAs are given separately to the multipliers near the memory cell array. In many cases, the results are also accumulated locally with digital adder tree. The type 6 IMC works in pure digial domain. Many recent IMC designs adopt the type 6 [79, 75], including the designs proposed in this thesis.

1.4 Analog-In-Memory-Compute

In the early stages of IMC research, most papers focused on Analog-In-Memory-Compute (AIMC) architectures. This focus was motivated by three key factors:

- Architectural Nature of IMC Macros: Unlike conventional digital cells, an IMC macro cannot be fully described using standard digital design methodologies. Its structure is more akin to an analog SRAM macro, requiring extensive low-level circuit simulation and validation. This inherent analog nature made it a natural choice to implement computations in the analog domain, leveraging the same design principles used for SRAM and similar memory technologies.
- Alignment with AI Model Requirements: AI models are known to be robust against small computational inaccuracies yet computational parallelism can drastically improve its energy efficiency. Analog computation inherently aligns with these requirements.
- Compatibility with Emerging Memory Technologies: Innovative memory technologies, such as Resistive RAM (RRAM), Magnetic RAM (MRAM), and Ferroelectric RAM (FeRAM), are intrinsically well-suited for analog computation. Especially the RRAM cell, which can be programmed by an analog voltage difference, and the programmed value is proportional to it. Moreover, the RRAM cell stores a continuous value, which is suitable for analog computation.

These factors made AIMC a prominent research focus, as shown in Fig.1.7, where its research activity peaked initially. However, significant limitations in AIMC have driven a growing shift toward digital approaches.

1.4.1 Limitations of ADC

For AIMC, the most critical component is the ADC. AIMC places stringent demands on the ADC, including the following:

- Precision Requirements: The ADC must provide sufficient precision for reliable operation. While AI models generally do not demand extremely high precision, a resolution of around 6 bits is typically required. This level of precision is already area-intensive for flash ADCs, making them less practical. Consequently, most designs employ SAR ADCs, which offer better area efficiency.

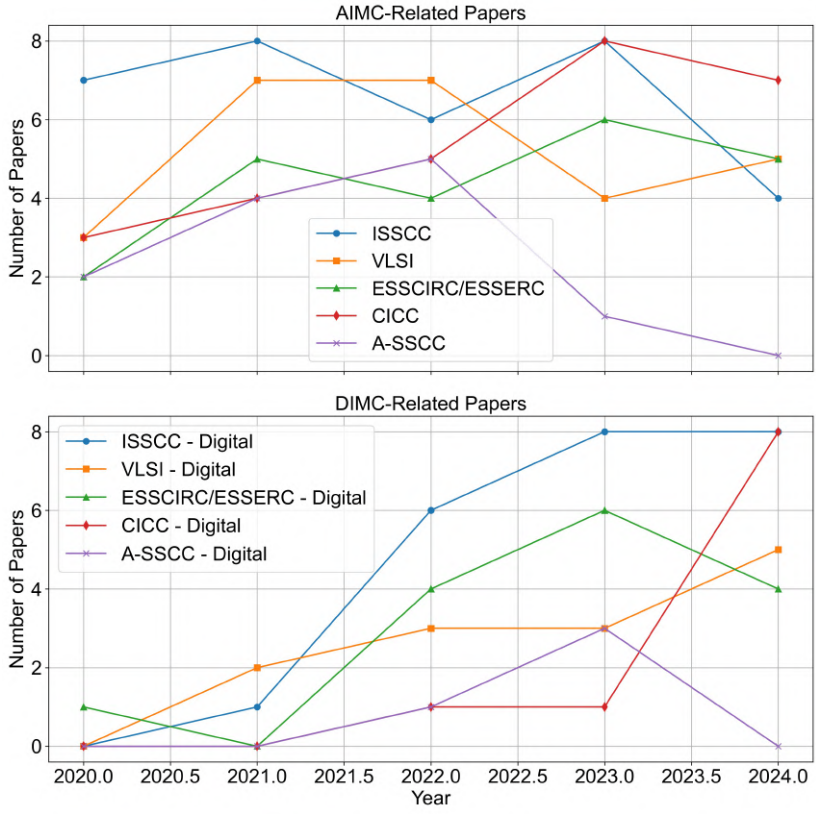


Figure 1.7: The number of papers related to IMC in ISSCC, VLSI, ESSCIRC(ESSERC), CICC, A-SSCC from year 2018 to year 2024.

- Energy Constraints: The ADC must operate with exceptionally low energy consumption. By definition, AIMC requires an ADC on each summation line (e.g., the BL), resulting in a large number of ADCs. If the energy efficiency of each ADC is insufficient, the cumulative energy consumption becomes a critical bottleneck.
- Performance Demands: AIMC applications require ADCs with high bandwidth to handle AI workloads effectively. While several efficient ADC designs have been proposed for low-bandwidth AIMC scenarios, achieving similar efficiency at a higher bandwidth remains a significant challenge.

These combined requirements—precision, energy efficiency, and high performance—highlight the complexity of integrating ADCs into AIMC systems.

Rather than focusing solely on circuit techniques to design ADCs that meet the stringent requirements, designers have explored methods to reduce the ADC workload. For instance, [98] proposes the use of a very low-precision time-to-digital converter (TDC), offloading partial product computations to the digital domain. Similarly, [27] suggests performing MAC operations only on the LSB portion using the ADC, while delegating the more critical MSB calculations to the digital domain. Taking this approach even further, [106] bypasses the accumulation in the MAC process altogether, instead performing aggregation at the bit level of the product using a simple 3-bit ADC. While these techniques effectively reduce the ADC workload, they come at the expense of the throughput benefits associated with analog computation, significantly diminishing its overall advantage.

Besides, the AIMC computation faces many non-ideality, and the computation itself also requires additional circuitry to ensure useful computation. The impact of those are discussed next.

1.4.2 Limitation of transistor variation

Variability is born with CMOS technology. Advanced nodes use FinFET technology. With their 3D structure and wrapped gate, FinFETs offer superior channel control compared to traditional planar CMOS, significantly reducing short-channel effects and sensitivity to threshold voltage variations from random dopant fluctuations. However, they are still susceptible to variations in fin width, height, and line-edge roughness, with these geometric inconsistencies becoming more challenging as dimensions shrink.

Transistor variability poses significant challenges in IMC cells, as these often utilize very small transistors that are highly susceptible to variations and performance fluctuations. In AIMC, the summation of bit-cell currents is expected to be proportional to the number of cells storing the value "1." To ensure correct operation, the accumulated errors from the individual cells must remain smaller than the current contribution of a single cell to avoid incorrect summation evaluations.

Fig.1.8 illustrates the current variation of a minimal transistor in a 6T SRAM cell. Even under ideal conditions, the variance of the DC current generated by a minimal transistor is approximately one-sixth of the mean. Assuming the operation of all 6T bit-cells is independent, the standard deviation of the current summation across #BL 6T bit-cells can be calculated as:

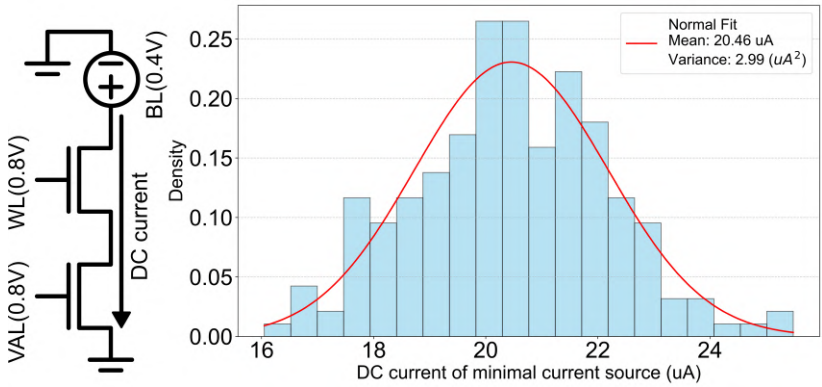


Figure 1.8: Current variation of minimal current source.

$$\sigma_{tot}^2 = \#BL \times \sigma_{6T}^2 \tag{1.1}$$

This implies that, assuming the standard deviation is less or equal to the mean, under the 3-sigma criterion, the maximum allowable number of BL is only $\#BL = \sqrt{Mean^2/(3 \times Variance)} = \sqrt{20.46^2/(3 \times 2.99)} = 6.8$, while under the 6-sigma criterion, it is reduced to just $\#BL = \sqrt{Mean^2/(6 \times Variance)} = \sqrt{20.46^2/(6 \times 2.99)} = 4.8$. In other words, to avoid false accumulation of bit-cell currents, only 4 bit-cells can be activated concurrently under the 6-sigma criterion. This severely limits the computational efficiency of the AIMC solution, as the small number of concurrently accessible bit-cells restricts parallelism and throughput.

1.4.3 Limitation of finite output impedance

Another limitation of transistors as current sources is their finite output impedance. This issue is particularly critical in AIMC applications, where the transistors typically have the minimum length permitted by the technology. Fig.1.9 illustrates the impact of finite output impedance on the accuracy of current summation.

In this simulation, we examine a typical AIMC current summation scenario where multiple minimal 6T current sources are connected to a shared BL. The BL is terminated with a 1kΩ resistor, representing the input impedance of a current-mode ADC. The plot demonstrates the relationship between BL voltage

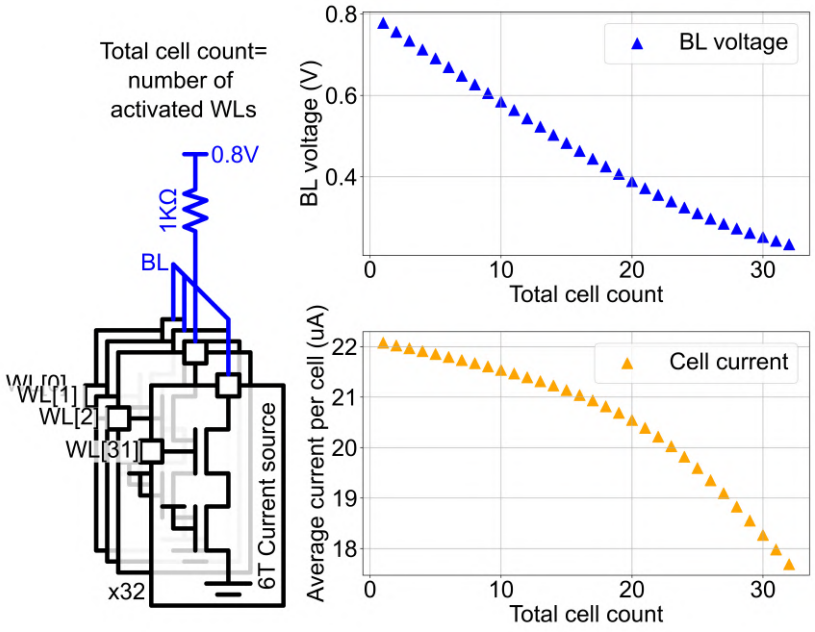


Figure 1.9: Output impedance of minimal current source.

and the number of 6T cells connected, as well as the average current contributed by a single 6T cell under varying summation conditions.

The results clearly show that the finite impedance of the small current sources causes significant BL voltage drops, leading to noticeable changes in the average current contribution of individual cells. In analog design, several circuit solutions can be used to address this problem.

Local Computing Cell

The most common approach to addressing the finite output impedance problem is the use of local computing cells (LCC) [31, 106]. The concept is illustrated in Fig.1.10. While the cell array utilizes optimized bit cells with minimal transistors, which suffer from high variability and poor output impedance, the LCC employs larger transistors with significantly increased length and width to mitigate these non-idealities.

In this architecture, the BLs native to the SRAM cells are no longer used

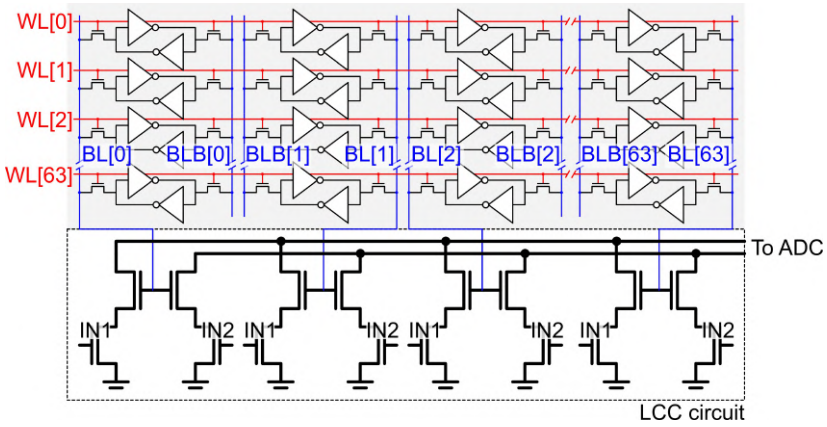


Figure 1.10: Idea of local computing cell (LCC).

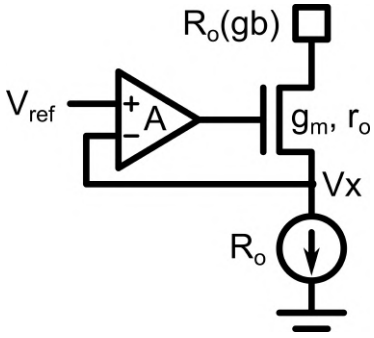


Figure 1.11: Idea of gain boosting.

for analog summation. Instead, they function as digital lines to control the on-off states of the large current sources within the LCC. These current sources generate analog currents based on the input voltages (IN1/IN2). The currents produced by each source are summed on a horizontal wire, and the combined current is then quantized by an ADC.

By utilizing LCC, the variability and finite output resistance issues of small 6T cell transistors are effectively resolved. However, this improvement comes at the cost of reduced parallelism. Despite this trade-off, LCC has garnered considerable attention in the research community and is more widely studied than alternative solutions.

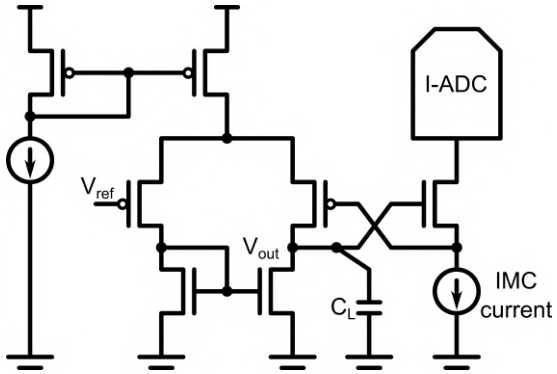


Figure 1.12: Simple OTA gain-boosting setup

Gain boosting

One method to increase the output impedance of a current source is through the gain-boosting technique. As illustrated in Fig.1.11, this approach employs an operational amplifier (OPAMP) in combination with a single NMOS transistor to regulate the output voltage of the original current source to a reference voltage, V_{ref}. It follows that

$$V_x \approx \frac{A}{A+1} V_{ref} \tag{1.2}$$

$$R_o(gb) \approx g_m r_o A R_o \tag{1.3}$$

This shows that with gain-boosting, the output impedance of the original current source is increased by the product of intrinsic gain and the gain of OPAMP. Apparently, the use of an OPAMP introduces energy efficiency drop. We would like to evaluate the energy cost of gain-boosting technique, relative to the power consumption from the IMC current summation.

Assuming a 5-transistor classic OTA is used for the gain-boosting stage, we can evaluate its power consumption based on the maximum IMC current and the load at the OTA's output. Keep in mind that only a simple OTA can be used as this circuit has to instantiated for every bit line. As a general rule of thumb, to achieve a specified gain-bandwidth product (GBW) for the OTA, its power consumption can be estimated as follows:

$$P_{OTA} = \frac{GBW}{2 \times 10^9} \times P_{IMC-Max} \tag{1.4}$$

The denominator term, 2×10^9, is an estimation based on an OTA with minimum transistor sizing. The power consumption of the OTA scales linearly with the required gain bandwidth product (GBW). The term $P_{IMC-Max}$ represents the peak IMC summation power, which corresponds to the maximum summation value on the BL. The OTA power scales with $P_{IMC-Max}$ for the following reason: When $P_{IMC-Max}$ increases, the feedback transistor needs to be biased with larger V_{gs}. If the feedback transistor is very small, then the V_{gs} is also large, requiring large V_{ds} to keep the transistor in saturation region. Because the BL voltage is set at V_{ref}, there is usually not enough voltage room for large V_{ds}. As a result, a large pass transistor is needed. However, this scaling increases the load capacitance, C_L, at the OTA's output node. Intuitively, for OPAMP, we have

$$\begin{aligned} GBW &\propto \frac{g_m}{C_L} \\ g_m &\propto \frac{I}{V_{gs}-V_t} \end{aligned} \tag{1.5}$$

Consequently, the power consumption of the OTA must increase to accommodate the higher load and maintain the desired performance.

The loop gain of the gain-boosting circuit can then be computed as:

$$\begin{aligned} A_{gb} &= \frac{GBW}{BW} \times g_m r_o \\ A_{gb} &= \frac{GBW}{BW} \times 15 \end{aligned} \tag{1.6}$$

where 15 is an average intrinsic gain of a transistor.

We can now calculate the power penalty associated with the gain-boosting technique. Consider a design where N 6T cells are summing on a BL, each contributing a current of 1μA. Let the output resistance of each cell be represented as r_o. Under these conditions, the following relationship holds:

$$R_o(gb) = \frac{A_{gb}}{N} \times r_o \tag{1.7}$$

$$P_{OTA} = \frac{GBW}{2\times10^9} \times N \times 1 \times 10^{-6} \times on_r \tag{1.8}$$

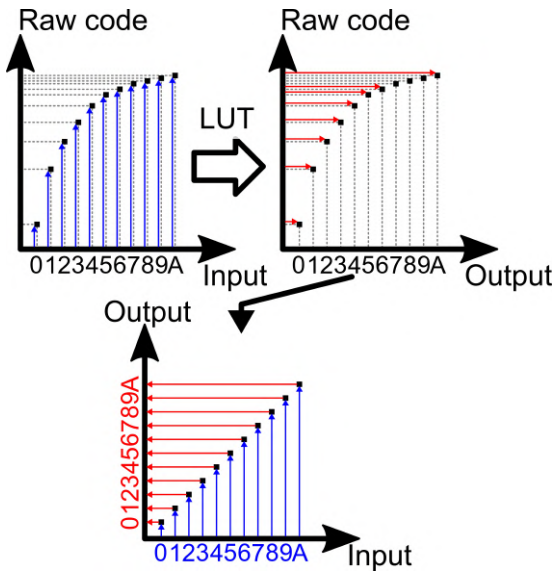


Figure 1.13: Idea of linearity compensation.

Here, on_r represents the percentage of time the DIMC current source remains active during a single clock cycle. Evidently, designing with N equal to the number of WLs leads to significant efficiency loss. However, even when N is limited to 10, the GBW is set to 2×10^9, derived from minimal sizing OTA on a $C_L = 5fF$, and on_r is reduced to 10%, the OTA power consumption still reaches $1\mu W$. It is important to note that a gain-boosting circuit is required for each BL. This relatively high energy demand makes the gain-boosting approach less attractive for AIMC applications.

As another solution, we can also biased the feedback transistor with a fixed voltage. Though energy efficient, it cannot either provide enough output resistance boost, or the pass transistor needs to be scaled exceptionally large.

Non-linearity correction

In many analog applications, device non-linearity is compensated for in the digital domain. Fig.1.13 illustrates this concept. Due to the previously discussed finite impedance issue, the IMC produces a distorted version of the summation value. This output, referred to as the raw code, is corrected using a pre-computed look-up table (LUT) that translates it back to the undistorted value.

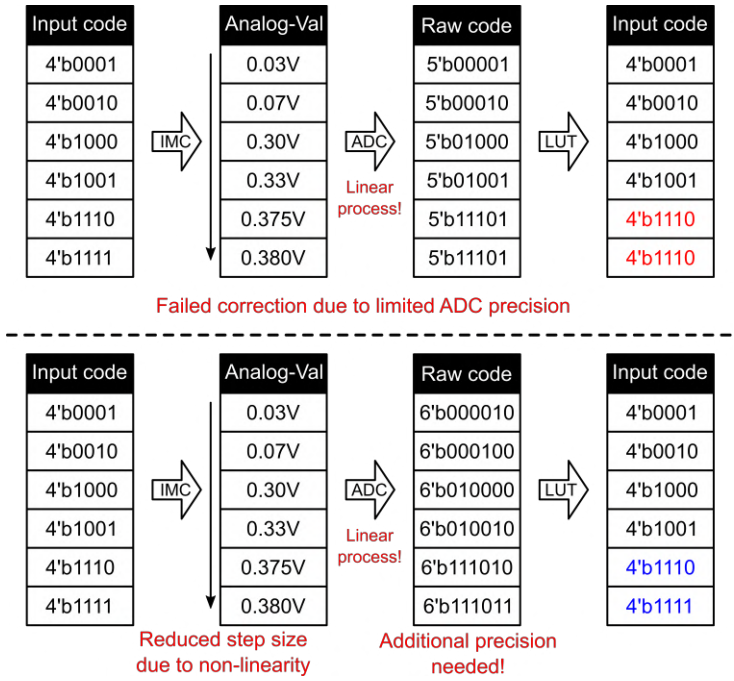


Figure 1.14: Additional precision for error correction.

Consequently, the clean, undistorted computation result is derived from the IMC array. With this correction technique, the IMC can operate effectively even under conditions where the current source exhibits finite impedance.

Despite its potential benefits, the correction technique faces three major challenges:

- Exponential Growth of LUT Entries: As the maximum summation value increases, the number of entries in the LUT grows exponentially, resulting in significant area overhead.
- Transistor Variation: Due to variations in transistor characteristics, the output impedance of 6T cells can differ across BLs, requiring a unique LUT function for each BL. This increases design complexity and area demands.
- Increased Bit Precision Requirements: This is the most critical issue. The LUT approach necessitates higher bit precision than typical

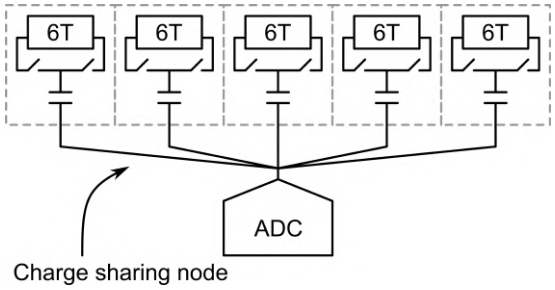


Figure 1.15: Charge sharing IMC with built-in capacitor.

implementations. Fig1.14 illustrates this problem: while the IMC input code may contain only 4 bits of information, the distortion caused by finite impedance results in non-uniform intervals for these input values. ADCs, however, typically convert analog signals into uniformly spaced digital values. When the ADC is limited to 4-bit resolution, this non-uniformity can lead to significant information loss. For example, when a 5-bit ADC is used, input codes 4'b1110 and 4'b1111 could be mapped to the same output value, 4'b1110, making it impossible to fully correct the non-linearity introduced by the IMC process. As a result, at least 6-bit ADC is needed for the correction.

The most straightforward solution is to increase the ADC's bit precision. However, for each additional bit, the area and energy costs of the ADC increase substantially. In energy-sensitive applications like AIMC, this penalty is unacceptable. Consequently, the error correction technique is not widely favored within the AIMC design community.

1.4.4 Limitation of area efficiency of capacitors

Instead of performing analog MAC operations in the current domain, which is prone to the previously discussed non-idealities, many researchers have explored implementing analog MAC in the voltage or charge domain with the aid of additional capacitors [49, 99]. The concept is illustrated in Fig.**??**.

In this approach, a unit capacitor is integrated into each SRAM cell. The original SRAM cell, combined with the capacitor and associated switches, forms a new composite cell, which serves as the fundamental building block of the IMC cell array. Computation is achieved through charge sharing among the capacitors in the selected cells of a BL.

This scheme offers two major benefits:

- Capacitors are inherently more resistant to process variations compared to transistors, resulting in significantly reduced fluctuation and non-linearity in the BL summation.
- The dynamic range of the BL voltage spans rail-to-rail, which simplifies the design and operation of subsequent ADC stages.

Nevertheless, capacitors are inherently area-inefficient. While some researchers argue that metal-fringing capacitors can reduce the area overhead of integrated capacitors [49], their overall efficiency remains constrained by the minimum spacing required to mitigate parasitic capacitance between surrounding cells. Consequently, the area efficiency of this approach is typically low, which is a critical limitation for memory cell design.

Researchers have made significant efforts to reduce the area overhead of charge-based IMC solutions. One approach involves using the BL as the unit capacitor [99]. While this method appears to reduce the area of each bit cell, it necessitates the use of additional local computing cells (LCC) for computation, ultimately lowering the overall area efficiency.

1.5 Digital-In-Memory-Compute

Unlike AIMC, which is affected by various non-idealities, digital circuits are largely free from these limitations. Given that AIMC designers have already minimized the analog workload to a very small subset, it may be more practical to shift entirely to the digital domain. As shown in Fig.1.7, recent publication trends highlight this shift. DIMC provides at least three key advantages:

1.5.1 Less susceptible to process variation

Digital circuits are inherently robust, thanks to their binary nature, which enables them to tolerate a wide range of variations. In AIMC, process variations can lead to issues such as false calculations and inaccurate ADC sampling. In contrast, variations in DIMC typically result only in differences in logic gate propagation delays, which are much easier to manage. Furthermore, modern computer aided design (CAD) tools are highly advanced and well-equipped to handle process variations in digital circuit design. Consequently, process variation in DIMC is much less problematic than it is in AIMC.

1.5.2 Compatible with advanced CMOS technology

In advanced CMOS technologies, scaling down digital circuits is significantly easier than analog circuits, largely due to the inherent robustness of digital designs. Studies from TSMC [26, 64, 25] have demonstrated that even at cutting-edge nodes such as 3nm, 4nm, and 5nm, DIMC consistently delivers robust operation and high throughput. In contrast, the analog circuit in AIMC benefits less from the transistor scaling. As a result, the AIMC designs are mostly designed with slightly older technologies, i.e., 28nm, 40nm.

1.5.3 Design flexibility

In addition to the technology-related advantages, DIMC also offers superior reconfigurability compared to AIMC. While large AI models are typically highly regular in structure, edge AI models often exhibit significant irregularities in network topology, operator precision, and computation sequences. This irregularity demands flexible architectures to accommodate diverse requirements. For AIMC, reconfiguration typically involves adjusting ADC precision, which is technically challenging and introduces considerable overhead in practice. In contrast, reconfiguring DIMC usually entails the addition or modification of logic gates, a process that incurs substantially less area and power overhead than modifying ADC configurations. Furthermore, the mature digital design ecosystem, including CAD tools and reusable IP blocks, makes implementing such reconfigurations more straightforward and efficient in DIMC. This flexibility is particularly advantageous for edge AI applications, where adaptability to diverse workloads is critical.

1.6 DIMC SoC design challenges

There are already many published DIMC based SoCs. [120] Introduced a DIMC-based scalable accelerator for AI inference, featuring 32 DIMC macros per chip. The design achieves 0.97 TOPs and 32.9 TOPs/W for 4-bit MAC operations at the system level. However, the DIMC capacity is limited to 1.7M-bit per chip, constraining the maximal model size. To address this, the authors proposed a chiplet approach to expand storage capacity. This approach, however, relies on a serialized chip-to-chip link, which incurs notable energy efficiency losses. [85] proposed a DIMC-based accelerator featuring a customized IMC cell array with 16 DIMC macros per chip. The design incorporates a Booth multiplication architecture to support both integer and floating-point

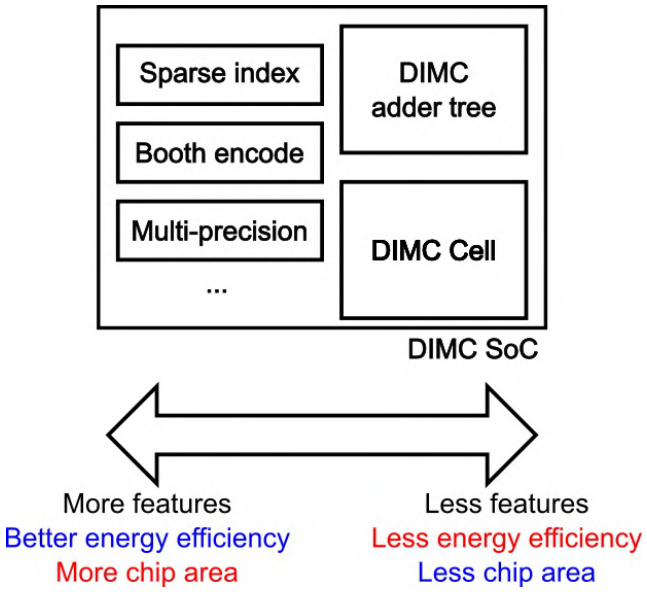


Figure 1.16: Area efficiency and energy efficiency trade-off for DIMC SoC

MAC operations, improving computation speed and energy efficiency. The accelerator achieves 1.35 TOPs and 19.5 TOPs/W for 8-bit MAC operations at the system level. However, the use of customized IMC cells, the inclusion of Booth multiplication, and the support for multi-mode MAC operations lead to significant losses in area efficiency. [87] proposed a DIMC-based accelerator tailored for transformer-based AI algorithms. The design emphasizes data flow optimization specific to transformers, targeting the longest data reuse patterns in input data, bit-level sparsity, and token sparsity. However, the hardware-level support required to accommodate these specialized features inevitably leads to a reduction in area efficiency.

1.6.1 Chip area for chip energy efficiency

While these DIMC-based SoCs achieve remarkable energy efficiency gain, they are usually achieved with usage of additional chip area. This can be observed from the significant tradeoff between area efficiency and energy efficiency in DIMC systems, as illustrated in Fig.1.16. Many advanced energy saving features rely heavily on additional circuitry and thus additional chip area.

For instance, to exploit the sparse data patterns common in AI models, many designs [86, 87] integrate sparse index modules. These modules skip multiply-with-zero computations and group operations into batches that share the same sparsity pattern, effectively improving energy efficiency by eliminating redundant operations. However, when dealing with edge models characterized by lower or irregular sparsity, additional logic circuits and memory are required to enable sparsity features. This increases the chip area overhead.

Another approach employed by researchers [39] involves integrating LUT-like logic, such as Booth encoding, to reduce the number of logic gates in the adder tree and lower computation energy. While this method improves energy efficiency, it comes at the cost of requiring additional DIMC memory for each computation. As the LUT size grows, the memory overhead becomes prohibitive, effectively shrinking the available DIMC memory for general computation and increasing chip area usage.

Similarly, supporting multi-precision reconfiguration allows the system to adjust computation precision based on operating conditions, optimizing energy efficiency. However, this flexibility introduces further tradeoffs, as additional circuitry is needed to handle multiple precision modes, leading to increased area consumption.

In summary, for DIMC SoCs, the tradeoff between chip area (for a given amount of DIMC memory and computation units) and equivalent energy efficiency is a fundamental consideration. It is interesting to review how different researchers consider this problem for different applications.

1.6.2 Definition of DIMC area efficiency and energy efficiency

Nevertheless, prior to delving into the published data regarding the area efficiency and energy efficiency of the DIMC, there are two points that require clarification:

- The energy efficiency of the DIMC is rather difficult to define precisely. The reason lies in the fact that energy efficiency is commonly measured in terms of tera-operations per second per Watt (TOPs/W). TOPs/W is typically derived from two types of measurements: the operation level TOPs/W and the task level TOPs/W. The operation level TOPs/W refers to the TOPs/W measured under the condition of maximum throughput. In theory, this metric is the most useful as it can be applied to all use cases. However, the majority of publications do not disclose such data. Conversely, the task level TOPs/W is more frequently utilized in the DIMC due to its lower measurement and programming complexity. Since different

designs are usually optimized for distinct types of models, the specific task employed for TOPs/W measurement varies from one design to another. This issue becomes even more pronounced for edge models, which possess numerous different layers and topologies. Moreover, the sparsity levels of different models also differ significantly. These data and model-dependent factors impede the DIMC community from reaching a consensus on a widely applicable benchmark, similar to what exists in the context of traditional CPU design. As a result, it becomes challenging to conduct easy and fair comparisons of energy efficiency among DIMCs. Despite these difficulties, we will still use TOPs/W reported in the literature for comparison. Throughout the remainder of this thesis, unless otherwise specified, TOPs/W will refer to the 50% sparsity level with the precision of the MAC provided.

- The area efficiency is associated with both the DIMC memory and DIMC computing units, which poses difficulties when it comes to making comparisons between different designs. There are primarily three significant issues in this regard. Firstly, the area efficiency of a design is tightly linked to the chip manufacturing process. For instance, a design fabricated using a 5nm process will intrinsically exhibit greater area efficiency compared to one produced with a 28nm process. Hence, it is essential to devise an effective method to eliminate the influence of technological differences on chips. Secondly, a common approach for evaluating the area efficiency of the computing cluster is through the metric of tera-operations per second per square millimeter (TOPs/mm^2). However, this metric has some drawbacks. To begin with, the metric inherently incorporates the frequency of the chip. This clearly gives an advantage to more advanced technologies and designs intended for high-performance applications rather than those for edge applications. Additionally, researchers typically report the TOPs/mm^2 and TOPs/W values under both the highest and the lowest frequencies. This is rather unrealistic since these two conditions can never be achieved simultaneously. Thirdly, the TOPs/mm^2 metric is exclusively determined by the number of DIMC computing units, without accounting for the DIMC memory units. For most designs, the measurement is done without considering the off-chip data moving energy. As a result, the design with less memory can only handle smaller models and vice versa. If considering only the TOPs/mm^2, it hinders fair comparisons between designs that prioritize greater computational power at the expense of local storage and those that emphasize increased local storage with reduced computational capabilities.

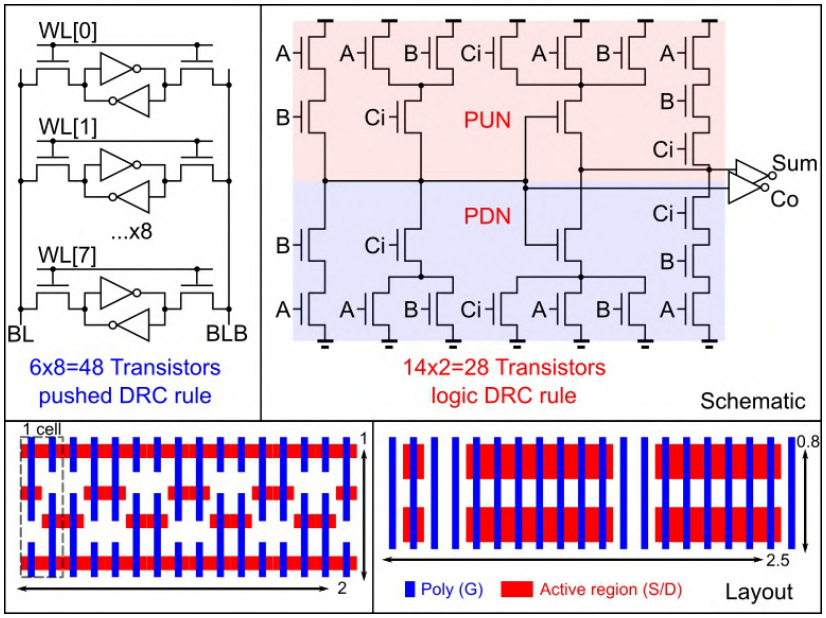


Figure 1.17: Comparison of 6T bit cell and a mirror full adder.

Technology (nm)	28	22	16	10	7	5	3
6T area (μm^2)	0.124	0.092	0.074	0.042	0.027	0.021	0.0199
Scaling factor	1.67	1.24	1.0	0.57	0.36	0.28	0.27

Table 1.1: Area scaling factor for different technologies

Consider both DIMC memory and computing units

To facilitate the establishment of a more suitable metric for area efficiency, let's explore the physical aspects of DIMC memory units and computing units. Fig.1.17 presents the schematic and layout of a 6T bit cell and a full adder standard cell. There are three reasons for using the full adder for area comparison. Firstly, it is the most compact cell within most standard cell libraries. Secondly, it constitutes the most fundamental cell in the design of adders and multipliers. Thirdly, in terms of poly spacing and source/drain spacing, it occupies roughly the same amount of chip area as 8 6T bit cells. For these reasons, we will define the area of either a full adder or 8 memory cells (1 byte) as a *DIMC unit*. Subsequently, the area efficiency will be denoted as the number of units per square millimeter.

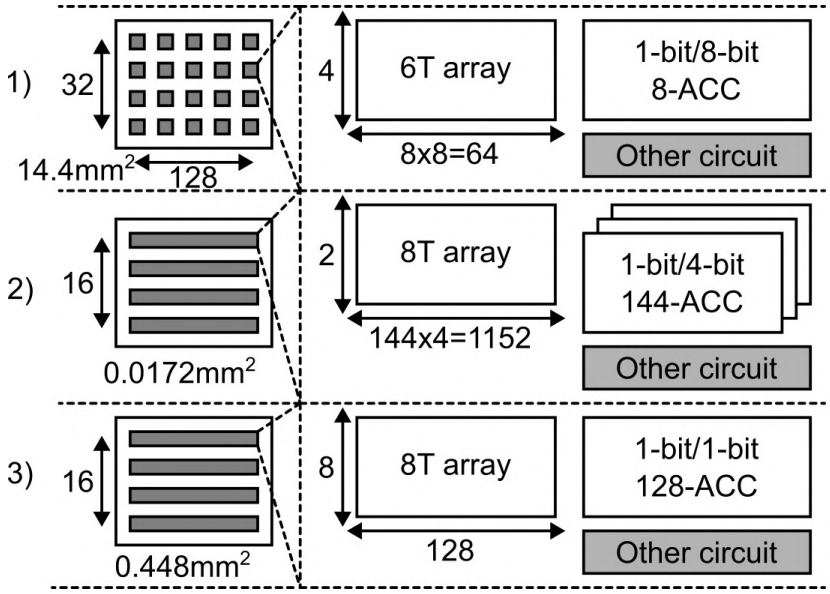


Figure 1.18: Illustration of DIMC chip area allocation. [87, 64, 38]

Compenstate for different technologies

The technology node used in a design is also an important factor that impacts the validity of area efficiency numbers. When the absolute chip area is considered, technology affects the metric. To decrease the influence of technology on it, we normalize the chip area to the 16 nm node using the area of the 6T bit-cell of that specific technology. The relevant information for this can be found online [4]. The reason we avoid using the technology value directly (in terms of minimal gate length) is that, in advanced technology nodes, the decrease in the area of standard cells doesn't maintain a proportional relationship with the technology scaling factor. The 6T SRAM bit-cell acts as a useful reference for the actual technology scaling factor among different technologies. Table 1.1 showcases the area of the 6T SRAM bit-cell for multiple technologies, along with their respective scaling factors. This way, it helps in providing a more consistent and comparable basis for evaluating area efficiency across different technological scenarios and designs, despite the variations in technology impact on chip areas and standard cell sizes.

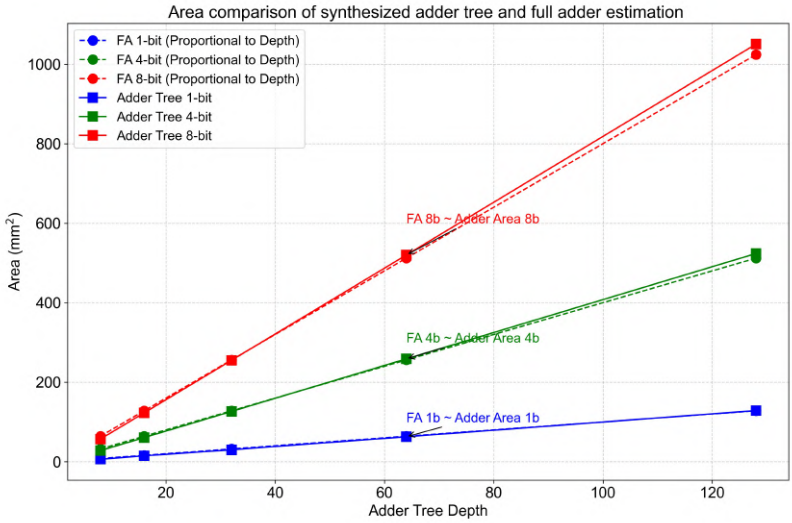


Figure 1.19: Area difference between synthesized adder tree and the full adder approximation.

1.6.3 Area efficiency computation

Fig. 1.18 illustrates the chip area utilization of a few example cases, showing how area efficiency is calculated for different chip designs. In example 1) [87], the chip consists of a cluster of 7,260 DIMC macros. Each macro contains a 4-by-64 6T bit cell array and a 1-bit/8-bit adder tree for 8 inputs (1-bit/8-bit 8-ACC). The "1-bit" indicates that the input activation is processed one bit at a time, while the "8-bit" indicates that an 8-bit weight is consumed in a single cycle. Since the input activation is only 1-bit, the multiplication reduces to a NAND gate, of which the area is negligible for this analysis. The adder tree in this design has a depth of 8. In example 2) [64], the chip consists of 16 DIMC macros. Each macro contains an 8-by-1152 8T bit cell array and three 1-bit/4-bit adder trees for 144 inputs. In example 3) [38], the chip consists of 16 DIMC macros. Each macro contains an 8-by-128 8T bit cell array and a 1-bit/1-bit adder tree for 128 inputs. The following formula is used to estimate the area of the adder tree for each design:

$$AT \approx \#TD \times \#IP \times A_{FA} \tag{1.9}$$

Where AT is the area of the adder tree, TD is the depth of the adder tree, IP is

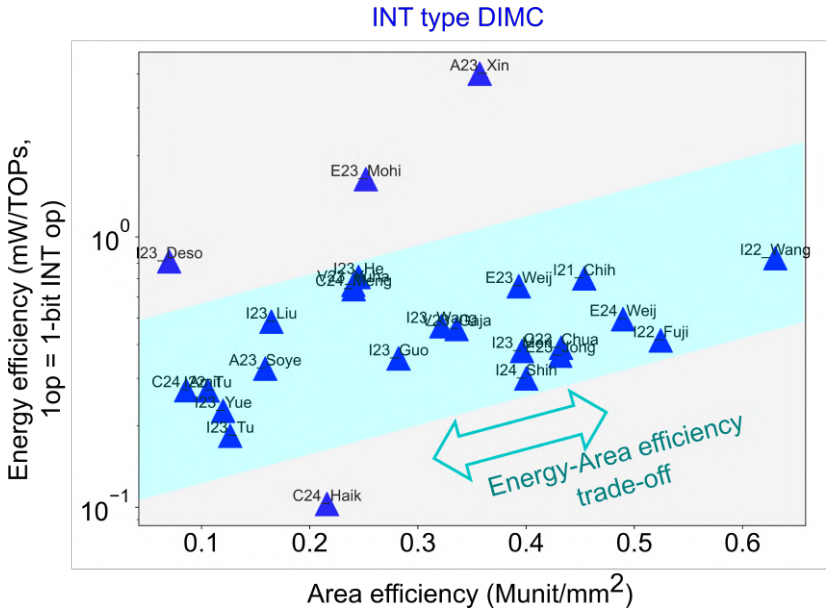


Figure 1.20: Area difference versus energy efficiency for state-of-the-art DIMC designs.

the precision of the input, and A_{FA} is the area of the full adder. In other words, we establish a one-to-one relationship between the area of the adder tree and the number of required full adders. We synthesized adder trees of different depths and input precision targeting minimal area and compared the area with the full adder approximation. Apparently, the counting of full adders provides an easy and effective way of estimating the adder tree area, as the Fig. 1.19. With this, we can derive the area efficiency for the three examples in Fig. 1.18. For examples 1, 2, and 3, the area efficiency in terms of units per millimeter square is 45,600 units/mm^2, 506,344 units/mm^2, and 14,000 units/mm^2, respectively.

DIMC macro area and energy efficiency trade-off

With the aforementioned knowledge, we can now analyze and compare DIMC designs, shedding light on the fundamental trade-offs inherent in their architectures. Fig. 1.20 illustrates the relationship between macro level energy efficiency and area efficiency for a range of recently published DIMC studies [16, 25, 102, 120, 85, 91, 86, 28, 64, 90, 110, 87, 60, 107, 33, 74, 30, 45, 13, 57, 21, 58, 78, 112, 97, 103, 66, 71, 93, 88, 115]. The data reveals a clear

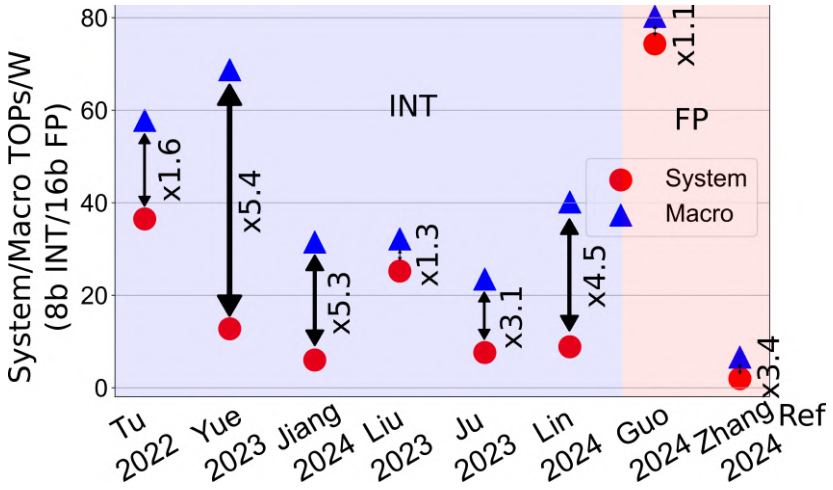


Figure 1.21: Macro and system energy efficiency gap for DIMC SoC designs.

negative correlation between energy efficiency and area efficiency, confirming the theoretical trade-off between these two metrics in DIMC designs.

This trade-off has significant implications for DIMC design and evaluation. The data supports our hypothesis that achieving high energy efficiency often comes at the cost of reduced area efficiency. This insight underscores the importance of carefully considering both metrics when designing or comparing DIMC architectures. Over-reliance TOPs/W is misleading, as this measure overlooks compromises in area efficiency. Such compromises can limit the practical benefits of the design, particularly in applications where overall system-level energy efficiency is critical. For example, Fig. 1.20 highlights a notable gap between macro-level and system-level energy efficiencies for state-of-the-art DIMC designs. This gap varies significantly across designs, suggesting the influence of additional factors such as workload characteristics, data movement overhead, and architectural constraints.

The observed gap between macro-level and system-level energy efficiency emphasizes the need for a holistic approach to DIMC design. System-level energy efficiency is influenced not only by the inherent energy efficiency of individual macros but also by their integration into the larger system. Designers must carefully balance these elements to optimize overall performance.

To summarize, DIMC designs inherently face a trade-off between area efficiency and energy efficiency. Recognizing this constraint, the core challenge for DIMC

designers lies in aligning the workload characteristics with the architecture. By minimizing the area required to reduce data movement energy and reallocating the remaining chip area to enhance DIMC energy efficiency, designers can achieve optimal system-level performance. This balancing act is particularly critical in edge AI applications, where resource constraints and energy demands are stringent. The subsequent section will delve deeper into these considerations and explore design strategies for optimizing DIMC architectures in edge AI scenarios.

1.7 How does DIMC help for edge AI applications.

Edge AI applications involve deploying artificial intelligence models directly on edge devices—such as smartphones, cameras, or IoT sensors—to enable local data processing. Edge AI is particularly critical for real-time inference tasks, including autonomous vehicle navigation and obstacle avoidance, smart camera-based surveillance and anomaly detection, continuous physiological monitoring in wearable health devices, and predictive maintenance in industrial IoT systems [32, 100, 7]. Local data processing ensures not only faster response times but also enhanced security by minimizing the transmission of confidential information.

To accommodate the resource constraints of edge hardware, AI models deployed in Edge AI are optimized for efficiency through techniques such as pruning, quantization, and knowledge distillation. These methods reduce model size and computational complexity. For instance, pruning removes redundant parameters, quantization reduces numerical precision (e.g., using 8-bit integers instead of floating-point values), and knowledge distillation transfers knowledge from large models to smaller, more efficient ones [92, 35, 41]. Lightweight architectures like MobileNet [41], ShuffleNet [116], and TinyML frameworks [7] exemplify these optimizations, achieving high inference speeds with minimal energy consumption and memory usage—key requirements for devices operating under tight energy budgets.

In this thesis, we focus on UNet, which is one of the edge CNN models that is widely adopted in field of image segmentation. The lightweight UNet variants have been developed to enable efficient image segmentation in edge applications. One such model is EdgeMedNet, a streamlined and precise UNet designed for medical image segmentation on edge devices. EdgeMedNet employs architectural optimizations to reduce computational complexity while maintaining high segmentation accuracy, making it suitable for deployment on low-power hardware [117].

Another example is L^3U-Net (Low-Latency Lightweight U-Net), which is tailored for real-time image segmentation on resource-constrained edge devices. L^3U-Net introduces a data folding technique that leverages parallel convolutional layer processing capabilities of CNN accelerators, significantly reducing inference latency. L^3U-Net achieved over 90% accuracy across multiple segmentation datasets, operating at 10 frames per second [56].

Additionally, LB-UNet (Lightweight Boundary-assisted UNet) has been proposed for skin lesion segmentation tasks on edge devices. LB-UNet incorporates a Group Shuffle Attention module to reduce model parameters and computational demands, facilitating deployment in resource-limited environments [51].

1.7.1 Properties of edge UNet models

Despite the increasing complexity of model topologies, the UNet architecture consistently retains three fundamental features: convolution operation, long skip connections, and transpose convolution operation.

- Convolution operation. The convolution operation is a core component of most UNet models, serving as the primary mechanism for feature extraction from images. Fig. 1.22 demonstrates the convolution process. The inputs include an input image and multiple convolution kernels. The input image is typically represented as a 3D tensor with dimensions corresponding to input width (IX), input height (IY), and the number of channels (not shown in Fig. 1.22). Similarly, each convolution kernel is a 3D tensor with dimensions for filter width (FX), filter height (FY), and the number of channels (also not shown). A convolution layer typically employs multiple convolution kernels in parallel, introducing an additional dimension, commonly referred to as the K-dimension. This dimension represents the number of kernels used in the layer. During the convolution process, a sampling window—matching the dimensions of the convolution kernel—is applied to the input image. Within this window, the convolution operation computes a dot product between the sampled image data and the kernel, producing an intermediate result. This result is then passed through a non-linear activation function to generate the final output of the layer. A widely used activation function is the Rectified Linear Unit (ReLU), which effectively sets negative values to zero, enhancing model performance and convergence. To generate the complete output feature map, the sampling window is systematically moved across the entire input image, with the process repeated for each convolution kernel. This approach enables the model to capture spatial patterns and hierarchical features across the image.

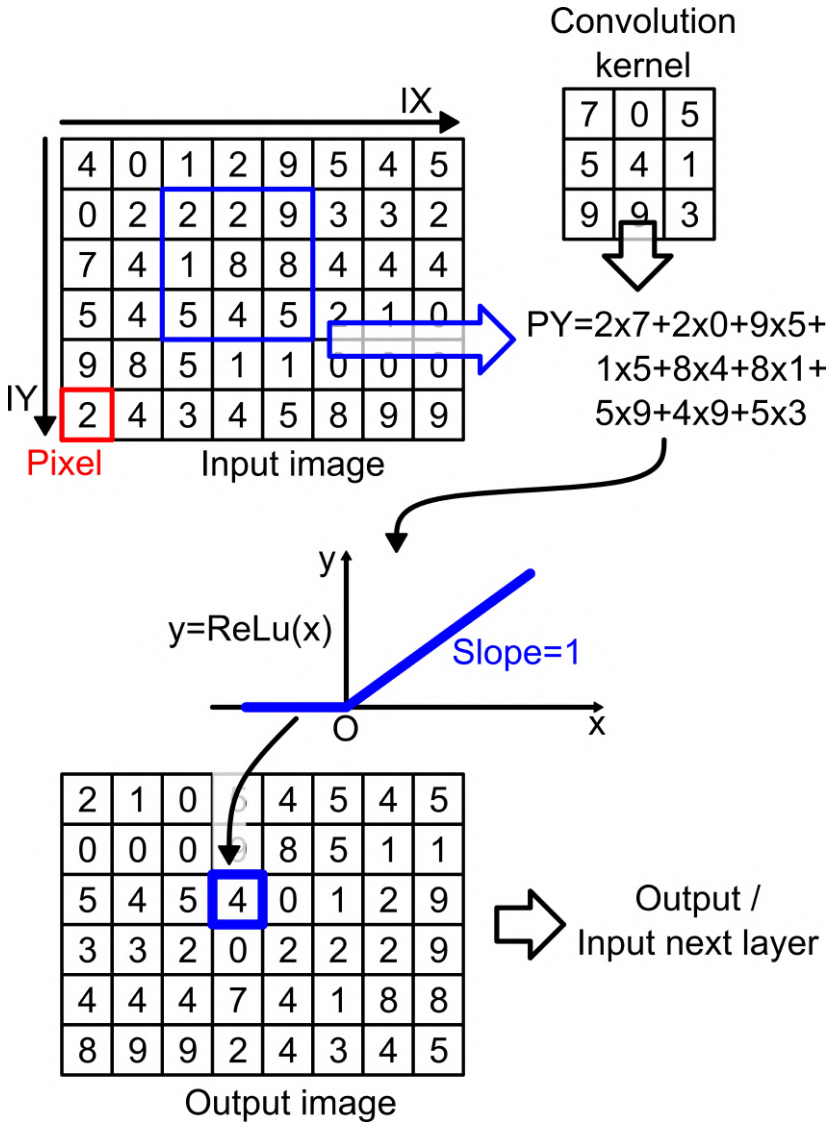


Figure 1.22: Illustration of convolution operation

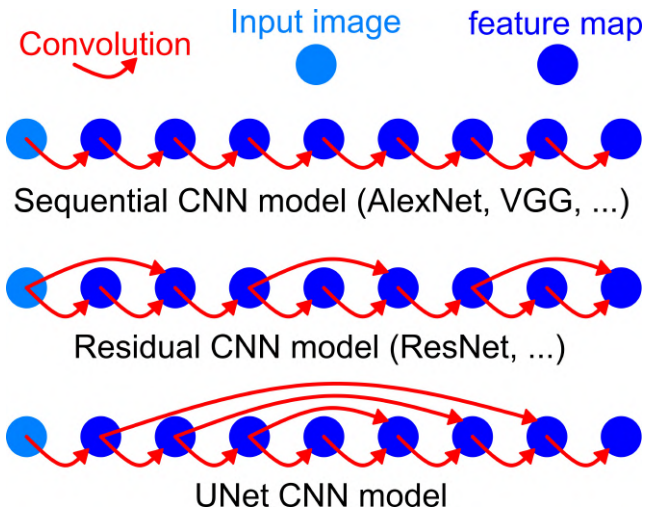


Figure 1.23: Long skip connections of UNet.

- Long skip connections. The UNet architecture incorporates multiple long skip connections, as illustrated in Fig. 1.23. In contrast, standard CNN typically follow a sequential structure, where the output of one layer directly serves as the input to the next, with no connections bypassing intermediate layers [54, 76]. This sequential nature of traditional CNNs often makes them challenging to train due to issues like the vanishing gradient problem [68]. To address this, ResNet introduced skip connections that bypass a specific number of layers, allowing gradient flow across deeper networks and improving trainability [37]. Building on this idea, UNet extends the use of skip connections by incorporating numerous long skip pathways between encoder and decoder stages. These connections preserve high-resolution spatial information and improve localization accuracy, which is crucial for tasks like image segmentation. However, skip connections have significant implications for memory management. The inclusion of long skip connections means that activations from earlier layers must be retained until they are reused later in the model. As a result, the memory requirements increase, posing challenges for resource-constrained systems and necessitating efficient memory management strategies to optimize performance.

- Transpose Convolution Operation. Image segmentation tasks requires generating an output image with the same dimensions as the input image with the segmentation results include. UNet achieves this by

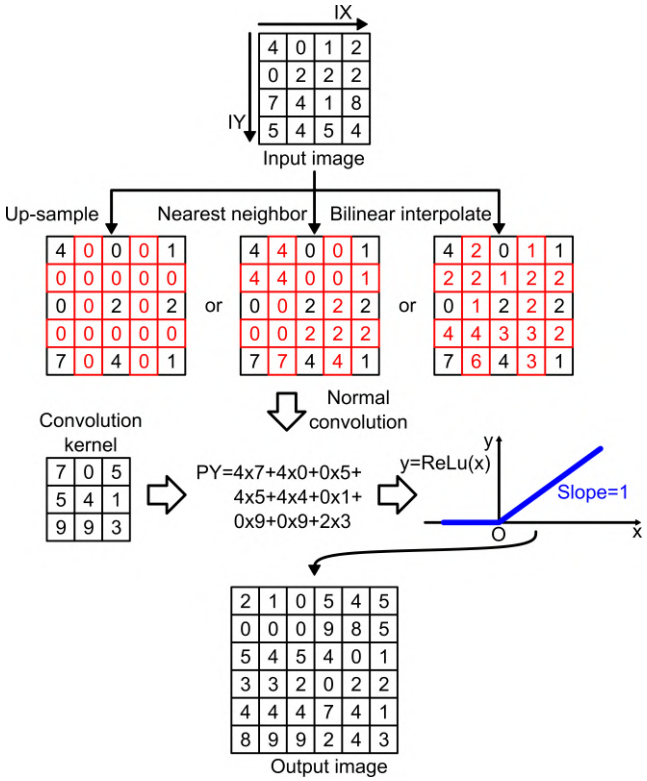


Figure 1.24: Transpose convolution.

employing the transpose convolution operation, as illustrated in Fig. 1.24. Unlike standard convolution, transpose convolution includes an additional upsampling stage before performing the convolution itself. The upsampling stage enlarges the spatial dimensions of the input feature map, and common upsampling methods include direct zero insertion (where zeros are inserted between input pixels), nearest-neighbor interpolation, and bilinear interpolation. These methods determine how the input data is expanded prior to applying the convolution operation. After the upsampling step, the remaining operations in transpose convolution are identical to those in standard convolution: the kernel slides over the upsampled feature map, computing dot products and generating the output. This process enables the model to effectively reconstruct high-resolution outputs while maintaining spatial relationships, which is critical for precise image segmentation.

1.7.2 DIMC's advantage over digital PE array

To demonstrate the advantages of DIMC over a digital PE array, let us consider the following edge UNet computation scenario:

- The model is fully stored on-chip, meaning the chip must be large enough to accommodate the entire model without relying on external memory.
- The model's parameter size is moderate, ranging from 1MB to 10MB, making the data movement energy comparable to the computation energy.
- The task's throughput requirements are non-negligible, necessitating a sufficiently large computation array to process the model within a reasonable timeframe. This is the case when e.g. the output of a camera has to be processed in real-time.

Fig. 1.25 illustrates the computational differences between a digital PE array and a DIMC array. In the case of the digital PE array, the convolution input data, weight data, and output data are all fetched from on-chip SRAM, which is relatively distant from the computation units in the PE array. Consequently, the total energy consumption is the sum of the input, output, and weight movement energy, along with the computation energy.

For an ideal DIMC array, where all input, weight, and output data are stored entirely within the array, data movement is confined within the DIMC array itself, significantly reducing energy consumption. In this scenario, the total energy is dominated by computation energy alone. However, achieving this ideal DIMC configuration is challenging, if not impossible, due to array size limitations and design complexities.

In most practical DIMC designs, only the weights are stored inside the array, while the input and output data remain in on-chip SRAM. As a result, the total energy consumption becomes the sum of the input and output movement energy, along with the computation energy. As a first-order approximation, if input, weight, and output movement energy are equal to computation energy, DIMC can achieve an approximate 25% reduction in energy consumption by limiting weight movement.

However, the actual energy gain is highly dependent on an efficient DIMC design. As previously discussed, DIMC architectures face a trade-off between area efficiency and energy efficiency, both of which can significantly influence the overall energy consumption.

For example, given a limited chip area, the designer must choose between allocating chip area to enhance DIMC energy efficiency or to improve its area

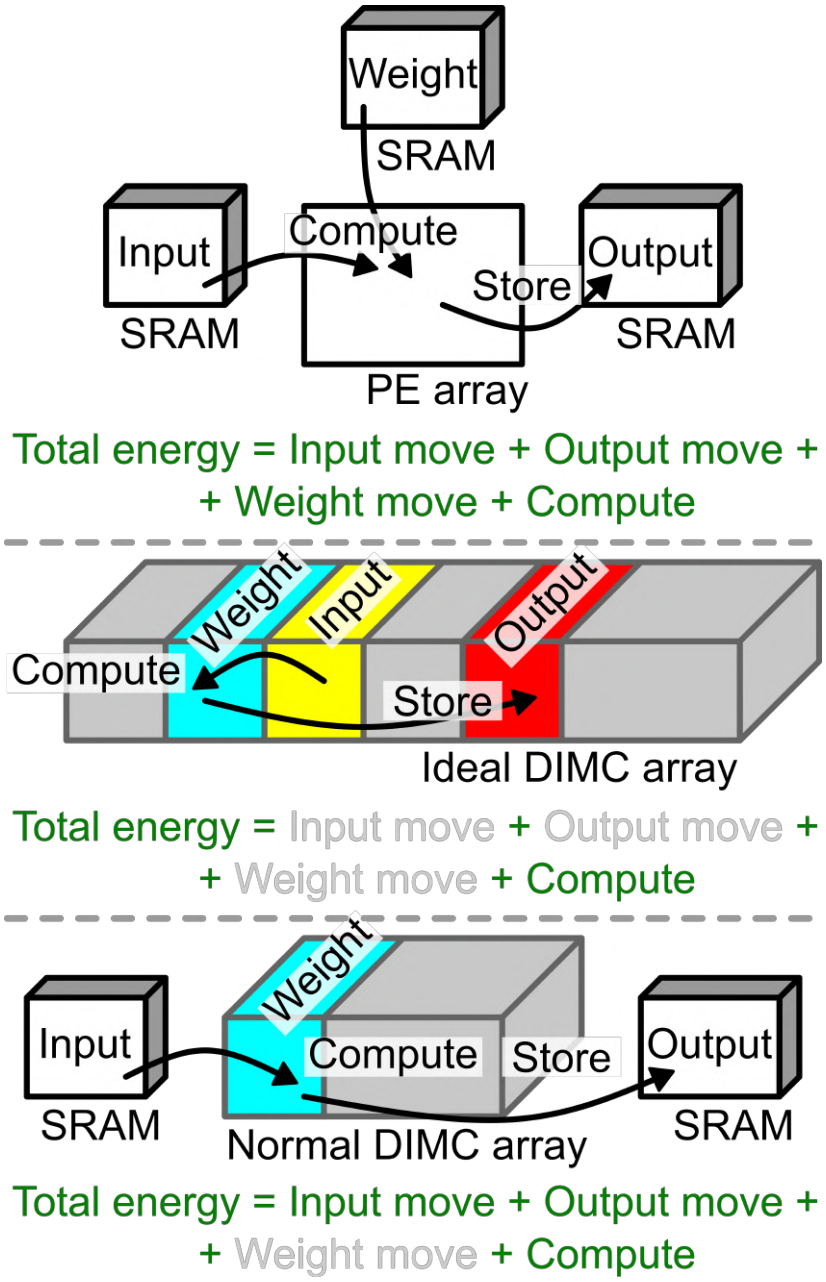


Figure 1.25: Edge UNet computation difference between digital PE array and DIMC.

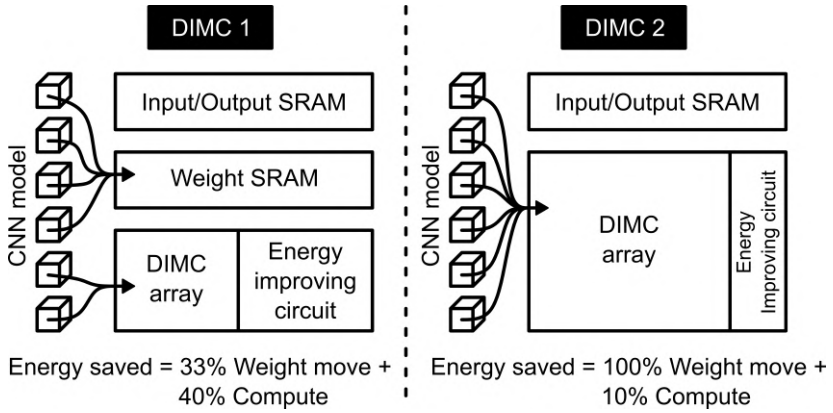


Figure 1.26: Energy saving for different DIMC designs for the limited chip area scenario.

efficiency. If additional chip area is used to improve energy efficiency, the entire model may not fit within the DIMC. In such cases, high-density on-chip SRAM is required to store the portions of the model that exceed the DIMC's capacity, as illustrated in the DIMC 1 case (Fig. 1.26). Suppose input, weight, output movement energy, and the computation energy are equal, and each takes 50%, the DIMC 1 case can save 33% of the weight moving energy, and the energy efficiency of the DIMC is improved by 40%, then the total energy savings in this scenario are calculated as $33\% \times 25\% + 40\% \times 25\% = 18.25\%$. Alternatively, if the designer prioritizes improving area efficiency, the entire model can fit within the DIMC, eliminating the need for additional SRAM. However, the energy efficiency of the DIMC may be slightly reduced due to the limited area available for energy efficiency enhancements. In this case, the total energy savings could be $100\% \times 25\% + 10\% \times 25\% = 27.5\%$.

In another example, when chip area is abundant, the optimal DIMC design changes. As shown in Fig. 1.27, with ample chip area, the DIMC 2 design over-fits the model, leaving a significant portion of the DIMC array unused. These inactive regions of the array do not contribute to energy savings for the system. In this scenario, the DIMC 1 design proves to be more effective, as the previously inactive DIMC area is utilized for computational energy savings. This not only eliminates weight movement energy but also further reduces total energy consumption.

In practice, the problem becomes more complex because the chip may need to support a variety of workloads. For instance, a UNet designed to segment only cars and humans might require just 1MB of weights and 3 skip connections. In

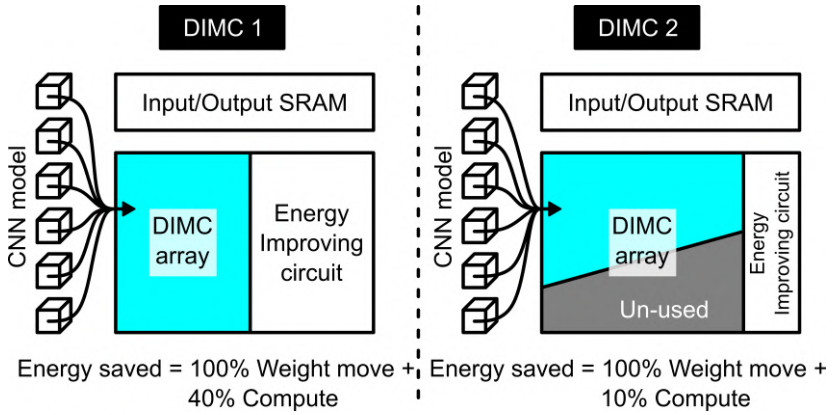


Figure 1.27: Energy saving for different DIMC designs for the abundant chip area scenario.

contrast, a UNet capable of segmenting cars, humans, traffic signs, animals, and labeling them could require 10MB of weights and 10 skip connections. Both models might be utilized depending on the system's power mode.

This means the system must be efficient across all possible scenarios, and the DIMC design should account for a wide range of workloads. The optimal DIMC configuration is highly dependent on the real-time workload distribution. If 90% of the time the larger workload is in use, a more area-efficient DIMC should be prioritized. Conversely, if 90% of the time the smaller workload is in use, a more energy-efficient DIMC is preferable, with weight SRAM providing additional storage when switching to the larger model.

Determining the optimal DIMC design within a large design space is an inherently complex problem. It extends beyond circuit design techniques to encompass considerations of algorithms, applications, and financial models. In this thesis, we do not delve deeply into this topic.

This thesis focuses on circuit improvement techniques for DIMC, with an example DIMC SoC designed specifically for edge UNet workloads. The results are validated through chip measurements conducted in 16nm FinFET technology. We present key circuit and system techniques aimed at enhancing the energy and area efficiency trade-off in DIMC systems, demonstrating their superiority.

Chapter 2

DIMC macro modeling

This chapter introduces the fundamental concepts of DIMC, outlining a modeling flow for each of its components. The DIMC architecture seamlessly integrates SRAM memory cells with computation logic, making it highly suitable for application-specific systems that require both efficient data storage and in-memory computation. Naturally, to model DIMC accurately, both the memory and computation circuits must be considered.

In the first section, we will explore different DIMC bit cell options, examining their advantages and drawbacks. The second section will focus on the impact of DIMC array dimensions on performance and efficiency. Finally, we will discuss the trade-offs involved in the DIMC multiply-and-accumulate (MAC) logic, highlighting key design considerations.

2.1 Basic concepts of DIMC

Fig.2.1 illustrates the fundamental concepts and parameters associated with DIMC sizing. Unlike traditional digital PE arrays, which are typically designed as matrix-matrix multiplication accelerators, DIMC macros are often configured as vector-matrix or vector-vector accelerators. These distinctions are depicted in Fig.2.1.a. In DIMC, full matrix-matrix multiplication tasks are usually decomposed into smaller matrix-vector or vector-vector multiplication operations, which are then scheduled along the time axis. This approach avoids implementing full matrix-matrix multiplication within a single cycle, primarily to reduce bit-line (BL) load.

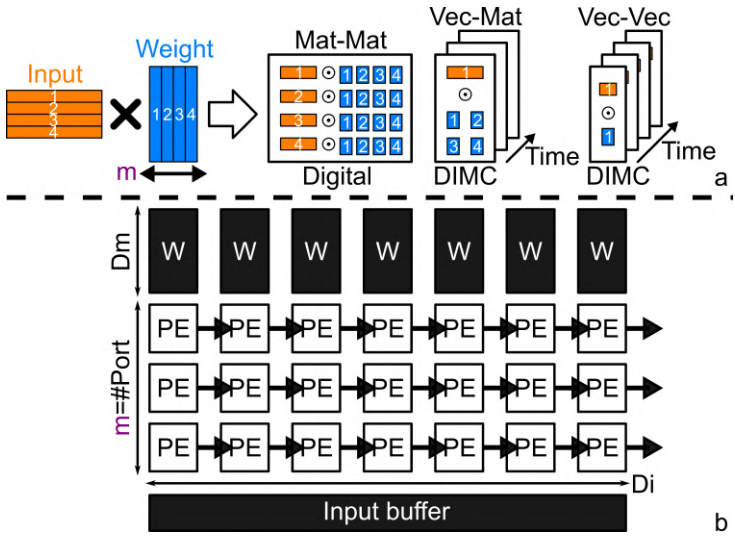


Figure 2.1: Basic concept and parameters of DIMC macro

Fig2.1.b presents a high-level overview of the DIMC macro organization. The weights are stored in a dedicated local weight memory within the DIMC, with the memory depth parameter represented as D_m. This indicates that for each row of processing elements (PEs), there are D_m different weight sets available for multiplication with the input vector. Each PE row performs a vector-vector multiplication operation, and the total number of rows determines the dimensionality of the weight matrix used for vector-matrix multiplication tasks. This hierarchical design enhances scalability and flexibility in DIMC macros, enabling efficient handling of diverse computation workloads.

Fig.2.2 illustrates the low-level organization of DIMC macros designed with different types of bit-cells. In Fig.2.2.a, the DIMC cell array is constructed using standard 6T bit-cells, preserving the original layout of the SRAM array. This configuration fully leverages the area efficiency of standard 6T/8T SRAM cells. The weight memory depth, D_m, corresponds to the total number of words in the array, while the input accumulation depth equals the number of BL/BLB pairs in the array. The standard SRAM array typically connects the bit line of every row, providing only one read/write port. In many DIMC implementations, the BL can be cut in the middle of the array, effectively creating two independent read ports on top and bottom, to increase the DIMC memory bandwidth. The DIMC MAC logic, which handles both multiplication and accumulation, is positioned near the SRAM array, leading to this configuration being commonly

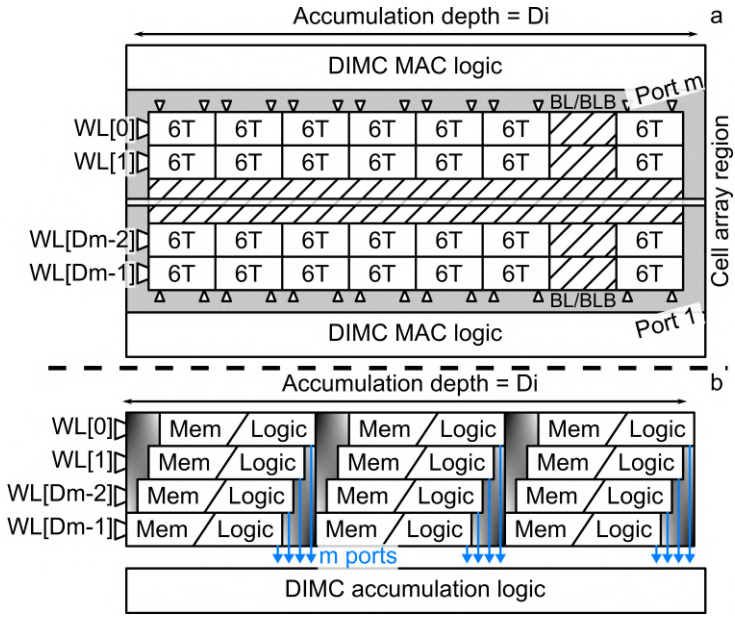


Figure 2.2: DIMC macro with different types of bit-cells.

referred to as "Near Memory Compute."

Fig.2.2.b depicts DIMC macros built with customized bit-cells that integrate multiplication functionality directly within each cell. These customized cells output their computed products through parallel metal buses connected to the DIMC accumulation logic located adjacent to the array. The weight memory depth, D_m, remains equivalent to the total number of words in the array. However, since each cell in a row requires additional metal wires to serve as DIMC read ports, the cell area increases significantly. In practice, to mitigate the challenges posed by increased metal spacing, designers often adopt a small D_m or group the outputs of multiple words to reduce the metal overhead of the output ports. This trade-off between functionality and area efficiency is critical in achieving a balanced DIMC design.

While the structure of a DIMC macro appears straightforward, it raises several important questions. For example, which type of cell is optimal for a given application? What parameters should be selected for specific models? How does energy efficiency scale when these parameters are adjusted? In this chapter, we will dive into this topic deeper.

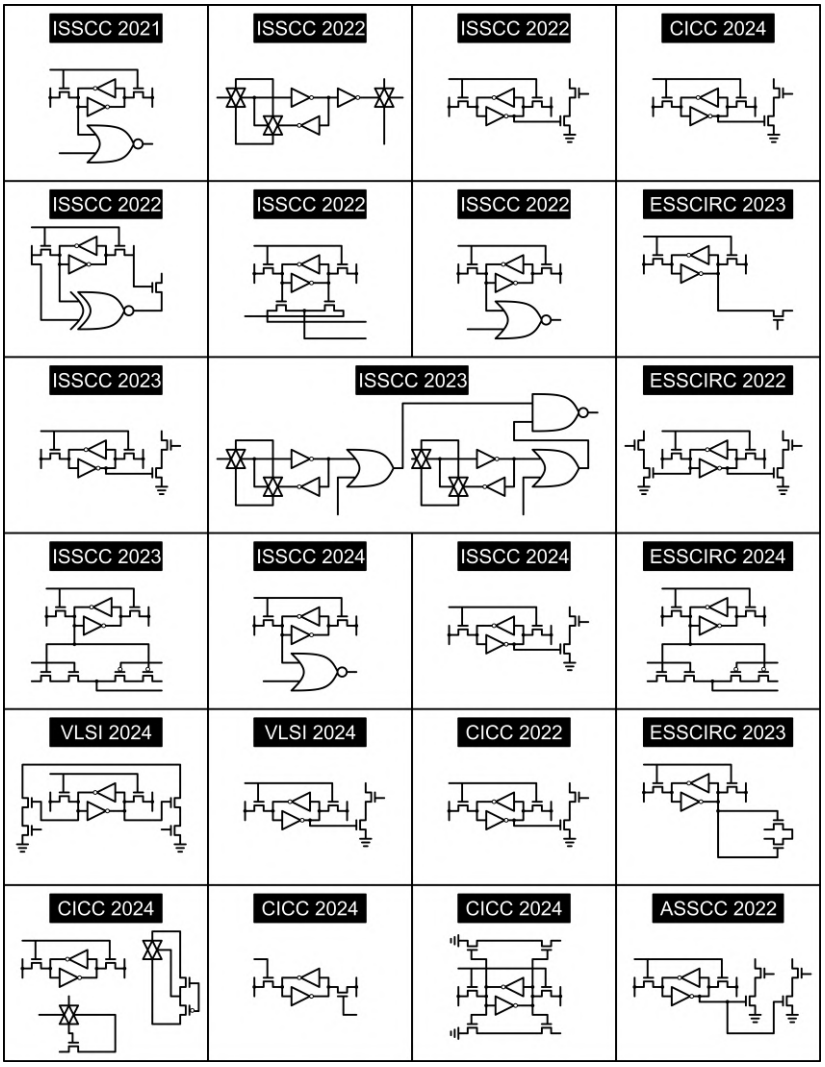


Figure 2.3: Non-6T bit cell used in the state-of-the-art.

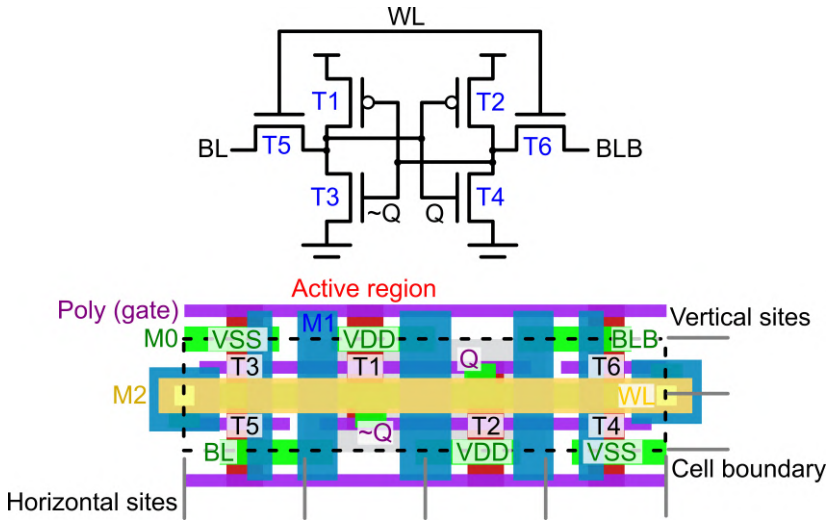


Figure 2.4: Circuit and layout of logic-rule 6T bit cell.

2.2 Choice of bit-cells

Many state-of-the-art designs employ non-6T bit cells for their DIMC macros. A comprehensive list of these non-6T bit cells is provided in Fig.2.3. These cells incorporate additional transistors and nodes compared to traditional 6T designs, which directly impact the area and parasitics of the cell. To thoroughly analyze these effects, we evaluate some representative cell types, and the classic 6T, separately. For all the cells, the layout is implemented using logic DRC rules.

2.2.1 Classic 6T

The classic 6T bit cell features a latch and pass transistors for data access. It is widely utilized across all technology nodes and serves as the fundamental building block for SRAM. Due to its critical role, foundries typically provide extensive support for it. For a first-order comparison, we will first implement the classic 6T bit cell using logic DRC rules.

The circuit and layout of the logic-rule 6T bit cell are shown in Fig.2.4. It comprises six transistors, labeled $T1$ to $T6$, and seven distinct nodes, including the power supply nodes (V_{DD} and V_{SS}). Advanced CMOS technologies adopt

what is referred to as the thin 6T layout, characterized by a high aspect ratio, hence the name "thin." Vertically, the pitch accommodates two transistors, while horizontally, it allocates space for four transistors. The total allocation for eight transistor locations ensures compatibility for forming arrays, where the boundary of one bit cell serves as the boundary of its neighbor. This symmetrical design is crucial to prevent concatenation issues.

In the 6T bit cell, all nodes are shared except for the internal storage nodes Q and $\overline{Q}$, which are concealed in the center and require a non-active region. The word line (WL) pin is shared on the left and right sides, while V_{DD}, V_{SS}, BL, and $\overline{BL}$ are shared at the top and bottom boundaries.

As a first-order approximation, the parasitic capacitance of the 6T bit cell is:

$$C_{WL} = 2C_{gs}, \quad C_{BL/\overline{BL}} = C_{gs} \tag{2.1}$$

This relationship implies that the parasitic capacitance on the bit lines (BL) is approximately half that of the word lines (WL). For SRAM designers, this is advantageous since the number of active bit lines during read/write operations is typically much greater than the number of active word lines. In many cases, the number of active bit lines is so big that many circuit techniques are adopted to further limit the swing on the bit line. However, for non-6T DIMC designs, where multiple WLs may be activated per cycle, the WL energy consumption becomes also significant. In subsequent sections, we will delve deeper into the WL and BL trade-offs in DIMC designs.

The classic 6T bit cell remains a popular choice for SRAM and DIMC designs due to its compactness and maturity. However, it is not well-suited for native digital computation. Some designers [22] have attempted to enable two-bit computations by simultaneously activating two WLs, but this approach requires both operands to reside within the cell array, leading to rigid data flow.

2.2.2 Classic 8T

The classic 8T bit cell builds upon the 6T bit cell by adding two additional NMOS pass transistors. It is widely utilized in advanced technologies due to its enhanced robustness and energy efficiency. Compared to the classic 6T bit cell, the 8T design is more robust as it eliminates read/write conflicts and consumes less dynamic energy through single-ended reads. It introduces more leakage because the two more pass transistors also introduce an additional leakage path. This becomes a serious issue for normal SRAM because the leakage of normal SRAM is usually more important than the dynamic energy. Despite this limitation, foundries usually also provide optimized 8T bit cells. We will examine the 8T bit cell under logic-rule DRC.

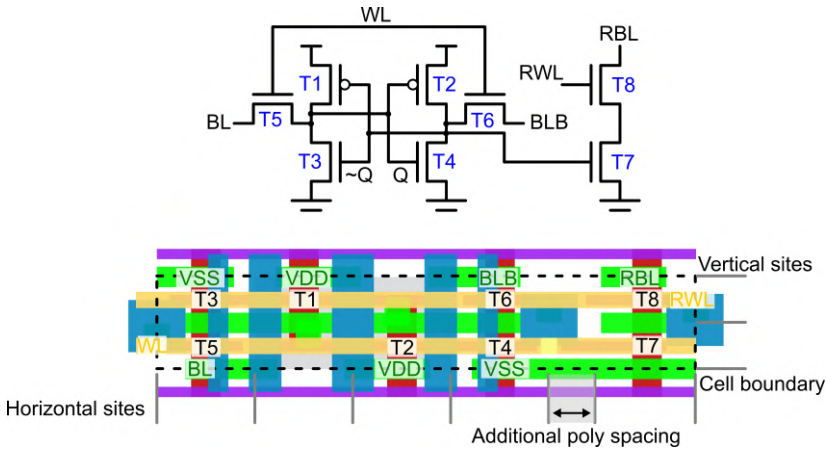


Figure 2.5: Circuit and layout of the logic-rule classic 8T bit cell.

The circuit and layout of the logic-rule 8T bit cell are shown in Fig. 2.5. The two additional NMOS pass transistors are labeled as $T7$ and $T8$. Fortunately, the inclusion of these two transistors introduces only two new shared nodes, which significantly mitigates concatenation issues when forming arrays. The parasitic characteristics of the 8T bit cell are also comparable to those of the 6T bit cell.

However, a critical scaling challenge arises with the 8T bit cell, particularly in the gate contact of $T8$. For $T7$, the gate is shared with $T3$, avoiding the need for an additional gate contact. In contrast, $T8$ requires a separate gate node. Although its gate contact can be placed at the cell boundary, $T8$ must share the same poly (gate) line with $T1$. To enable this, the poly must be cut between $T1$ and $T8$. Unfortunately, this poly-cutting process is especially challenging in FinFET technology, as it necessitates additional transistor spacing.

In summary, while the 8T bit cell offers significant advantages over the 6T bit cell in terms of robustness and energy efficiency, memory density remains a critical concern, particularly when scaling to advanced process nodes.

2.2.3 Non classic 8T

A variety of non-classic 8T bit cells have been proposed in the literature. One representative example is shown in Fig. 2.6 [91]. This bit cell incorporates an NMOS binary multiplier, enabling functionality such as AND and XOR

Type	Equation	RWL	RWLB	Q	Q	RBL
AND	$1 \times 1 = 1$	1	0	1	0	1
AND	$1 \times 0 = 0$	1	0	0	1	0
AND	$0 \times 1 = 0$	0	0	1	0	0
AND	$0 \times 0 = 0$	0	0	0	1	0
XOR	$0 \oplus 0 = 0$	1	0	0	1	0
XOR	$0 \oplus 1 = 1$	1	0	1	0	1
XOR	$1 \oplus 0 = 1$	0	1	0	1	1
XOR	$1 \oplus 1 = 0$	0	1	1	0	0

Table 2.1: Truth table for the DIMC bit cell in [91]

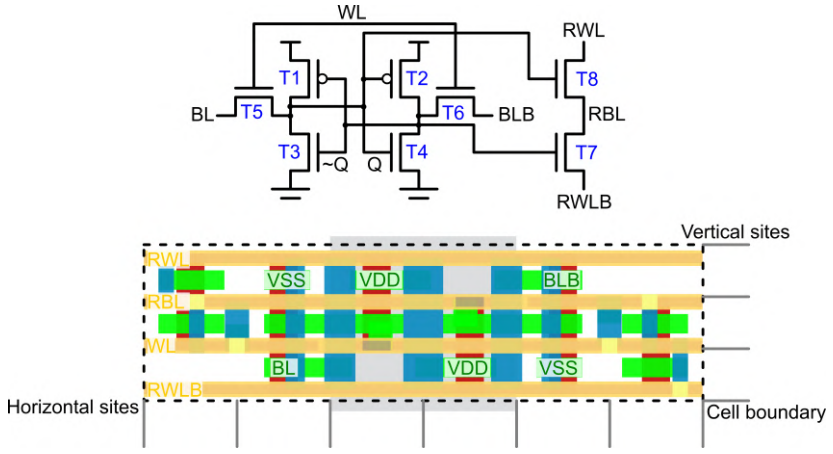


Figure 2.6: Circuit and layout of logic-rule non classic 8T bit cell.

operations by adjusting the RWL and $RWLB$ voltages. The corresponding truth table for the AND and XOR operations is provided in Table 2.1. By adding only two transistors, this DIMC bit cell significantly expands functionality.

Despite its advantages, the bit cell is not layout-friendly. The primary challenge arises from the addition of three horizontally routed nodes: RWL, $RWLB$, and RBL. In total, four nodes require separate horizontal metal layers, necessitating a total spacing of 8 times the minimum metal pitch if minimal width and pitch are used. In most technologies, this is much larger than the pitch allocated for two transistors. Fig. 2.6 presents an example layout supporting this arrangement, which requires three transistor sites vertically and two additional sites horizontally. Consequently, this type of 8T cell suffers from significant

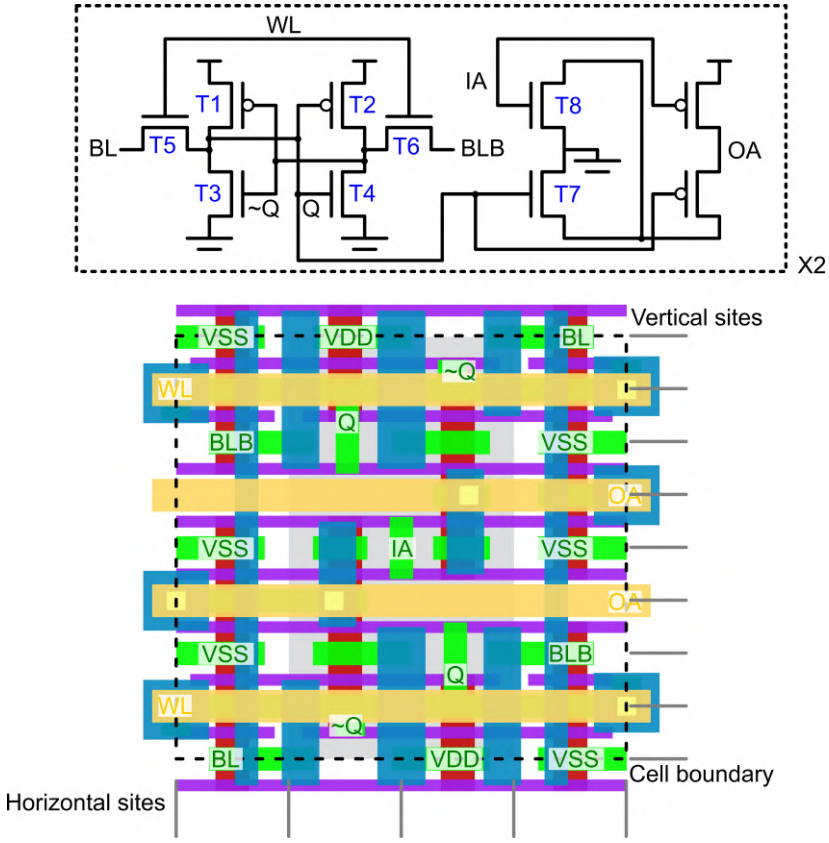


Figure 2.7: Circuit and layout of logic-rule 10T bit cell.

area inefficiency.

Furthermore, the computation in this bit cell is not performed using standard static CMOS logic but relies on pure NMOS logic. Since NMOS logic causes a voltage drop during pull-up operations, the output voltage of RBL cannot reach the full power supply level. This necessitates the use of level shifters or additional circuit support in the post-processing DIMC circuit, further reducing the overall area efficiency.

2.2.4 10T cell

A more robust multiplication circuit can be integrated into the 6T bit cell by adding two additional PMOS transistors to form a complete CMOS NOR gate, as illustrated in Fig. 2.7 [16]. This enhanced bit cell generates a full-swing multiplication result of the weight and input bit, which can be directly utilized by the nearby digital adder tree.

The design is also layout-friendly. By efficiently utilizing the gate poly and sharing cell regions between two bit cells, site utilization can be significantly optimized. However, the in-cell metal routing remains a major challenge. The complexity of the routing introduces significant yield and scaling difficulties, particularly at advanced technology nodes.

2.2.5 Foundry bit cell and logic DRC bit cell

Typically, foundries optimize the classic 6T and 8T bit cells to enable SRAM with improved density, performance, and manufacturability. These optimizations leverage advanced layout techniques and technology-specific design rules to maximize efficiency. The key differences in these optimizations are as follows:

- The horizontal site width is further minimized by reducing the width of the active region and decreasing the spacing between adjacent active regions. This results in a tighter, more compact layout, improving overall cell density.
- The additional area required for poly horizontal spacing is significantly optimized. This is particularly advantageous for the classic 8T bit cell, where horizontal poly spacing often contributes disproportionately to the overall cell area.

In general, foundry-provided bit cells are significantly more area-efficient compared to logic DRC bit cells. On average, an area reduction of approximately 70% can be achieved by transitioning to foundry bit cells. As a result, DIMC designs utilizing foundry-optimized cells can achieve significantly higher area efficiency, enabling more compact and energy-efficient architectures for advanced computing applications.

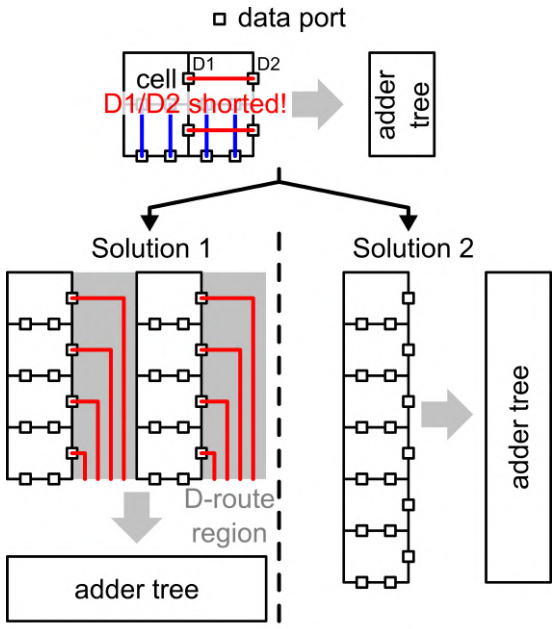


Figure 2.8: Cell concatenation problem for 10T and non classic 8T cell.

2.2.6 Area efficiency comparison for DIMC bit cells

Let us examine the area efficiency differences among the aforementioned bit cells. At this stage, the adder tree circuit is not taken into consideration; we focus solely on the memory cells within the DIMC unit (as defined in Chapter 1). To account for the integrated multiplier in the non-classic 8T and 10T cells, a 20% density bonus is applied to these cells.

It is important to note that not all bit cells can form a seamless array. This issue is depicted in Fig. 2.8. For DIMC cells that integrate additional multipliers, a separate output data port is required for each cell. Unlike the vertical bit line port, which is shared among all cells in the vertical dimension, the output data port, typically routed horizontally, must be individually separated for cells on the same horizontal line. This introduces significant routing congestion. As shown in Fig. 2.8, the two-by-two array cannot be formed because the direct routing of D_1 and D_2 causes an output short circuit.

Two solutions are proposed to address this issue:

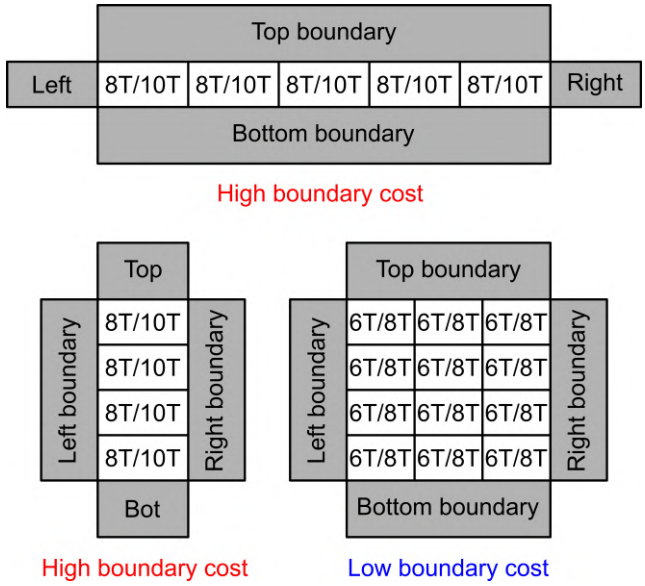


Figure 2.9: Illustration of area overhead on the boundary.

- **Solution 1**: Avoid horizontal routing of the output data by reserving a dedicated data-route (D-route) empty region for the output data. The width of the D-route region is proportional to the number of rows in the DIMC sub-array. This solution is well-suited for wide DIMC arrays but becomes inefficient for square DIMC arrays, as it attempts to fit n^2 wires into a 1D dimension, resulting in substantial area overhead.
- **Solution 2**: Abandon the conventional array shape of DIMC bit cells and adopt a one-by-n tall vector configuration for storage. The adder tree can then be positioned on the right side, directly accessing data from the output data port. However, this approach has a clear drawback: it severely restricts the aspect ratio of the DIMC array, leading to extreme proportions.

In contrast, classic 6T and 8T bit cells can form seamless arrays of any dimension, which is especially advantageous as the array size increases. The underlying reason is illustrated in Fig. 2.9. In advanced technologies, transistors must be properly terminated to avoid DRC violations. Termination regions, which consist of empty silicon, contribute negatively to the area efficiency of DIMC. For small arrays with high aspect ratios, the relative area occupied by termination regions is significant. Conversely, for large arrays with low aspect ratios, the

Cell type	DimX	DimY	Area efficiency (ratioed)
6T(DRC)	32	32	100%
8T(DRC)	32	32	74%
8T(NC)	1	32	19%
10T	1	32	22%
6T(FDR)	32	32	290%
8T(FDR)	32	32	210%

Table 2.2: Area utilization for different bit cells of 32-by-32 array.

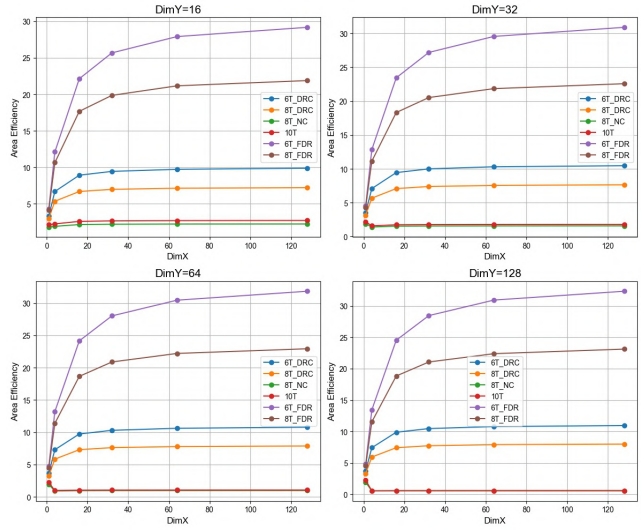


Figure 2.10: Area efficiency for different bit cells of different array dimension.

relative area occupied by these regions is minimal. To maximize area utilization, it is always desirable to build large arrays with low aspect ratios. This principle explains why the memory density of a typical register file is significantly lower than that of a standard SRAM.

Table 2.2 provides an overview of the area efficiency for various bit cells within a 32-by-32 array, assuming a consistent amount of cell termination area. For non-classic 8T and 10T bit cells, Solution 2 is applied. Even when considering the 20% bonus for the integrated multiplier in these non-classic bit cells, the penalty on area efficiency remains substantial.

Conversely, the use of foundry-optimized bit cells significantly enhances area

efficiency. A more detailed comparison across different array sizes is presented in Fig. 2.10. The results demonstrate that larger arrays are generally advantageous for improving area efficiency, although the rate of improvement diminishes as array size increases. It is important to note that this analysis considers only the impact of bit cell termination area. In practical DIMC designs, which include additional components such as decoders and pre-charge drivers, the point at which efficiency gains slow down occurs much later.

In summary, to achieve optimal area efficiency, it is recommended to utilize classic 6T and 8T bit cells, or even better, their foundry-optimized counterparts.

2.3 Choice of array dimension

An essential consideration for DIMC design is the dimension of the array. As previously mentioned, the array dimension has a significant impact on the area efficiency of the DIMC. In addition to area efficiency, it also plays a critical role in determining the timing, energy consumption, and speed of the DIMC macro.

In this section, we will conduct a detailed analysis of the trade-offs associated with classic 6T DIMC arrays. Our focus will be on the Type 6 DIMC discussed in Chapter I, as it represents the most commonly used configuration for 6T DIMC designs.

2.3.1 The lumped 6T DIMC array model

To assess the speed and energy consumption of the DIMC, it is crucial to analyze the parasitic effects within the DIMC array. In conventional SRAM, the RC lumped model is the most widely used approach for circuit simulation [47, 119]. Following this established methodology, we will apply the same model to evaluate the 6T DIMC array.

Basic structure

The basic lumped 6T DIMC model is shown in Fig. 2.11. Two key nodes of interest are the word line (WL) node and the bit line (BL) node. The WL node connects to all the pass transistors in a word and extends due to the thin-film structure of the 6T bit cell, resulting in several parasitic components.

First, the NMOS pass transistors introduce two parasitic capacitances: gate-to-source and gate-to-drain. Second, there are gate-to-bulk parasitic capacitances

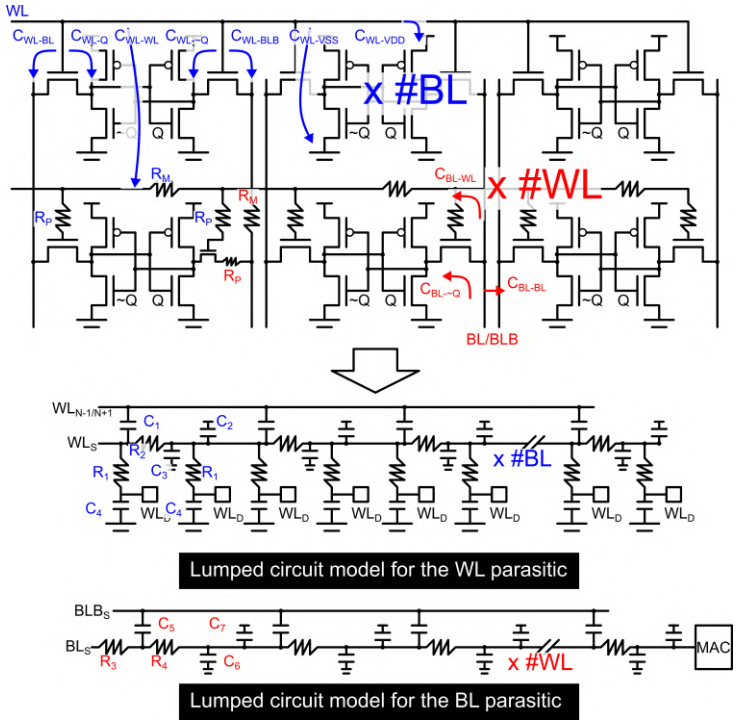


Figure 2.11: Lumped parasitic model for 6T DIMC array.

associated with the NMOS pass transistors. Third, a metal-to-metal parasitic capacitance exists. As a rule of thumb, only the parasitics between WL_n and $WL_{n-1/n+1}$ need to be considered.

The parasitic resistance of the WL consists of two components: the gate poly parasitic resistance and the WL metal parasitic resistance.

The BL node connects to all the pass transistors in a column and primarily consists of three parasitic capacitances: gate-to-drain, drain-to-bulk, and metal-to-metal. The parasitic resistance is composed of two main components: the drain resistance and the metal resistance. The equivalent RC lumped circuit model is shown in Fig. 2.11.

The key distinction between the WL and BL nodes lies in their network structure. The WL functions as a fan-out network, while the BL operates as an end-to-end network, extending from the source (BL_s) to the DIMC MAC logic. The values of these parasitic components for different array sizes are summarized in

C_1	C_2	C_3	C_4
1.0	0.33	1.3	4.0
C_5	C_6	C_7	
0.0	3.5	0.24	
R_1	R_2	R_3	R_4
1.0	0.52	0.875	0.01

Table 2.3: Parasitic value table for 6T SRAM (scaled).

Table 2.3. To maintain confidentiality and comply with NDA restrictions, these values are scaled without units.

Several important observations can be made:

- First, C_4 and C_6 are the largest parasitic capacitances. This indicates that the transistors contribute more parasitics than the interconnects.
- Second, C_5 is zero because the BL of two neighboring cells are separated by the WL contact, which introduces a large parasitic capacitance, C_6. This phenomenon can be understood as parasitic shielding.
- Third, the WL parasitic resistance is higher than the BL parasitic resistance, making the speed of the DIMC more sensitive to WL resistance than to BL resistance.

Model validation

To validate the effectiveness of the lumped 6T DIMC model, we compare it against the extracted netlist of a 32-by-32 DIMC array. Fig. 2.12 presents a waveform comparison between the lumped model and the extracted netlist of the DIMC array.

In this experiment, the WL signal is driven by an inverter and propagated to the destination bit-cell through either the extracted netlist or the lumped model. The bit-cell location is set at the end of the WL, with the fifth word line, WL[4], selected for reading. For the corresponding lumped model, a total of five cell segments are used for BL simulation, while 32 cell segments are employed for WL simulation.

To further validate the effectiveness of the lumped model across different array sizes, Fig. 2.13, Fig. 2.14, and Fig. 2.15 present the waveform comparisons for various array sizes and different choices of WL.

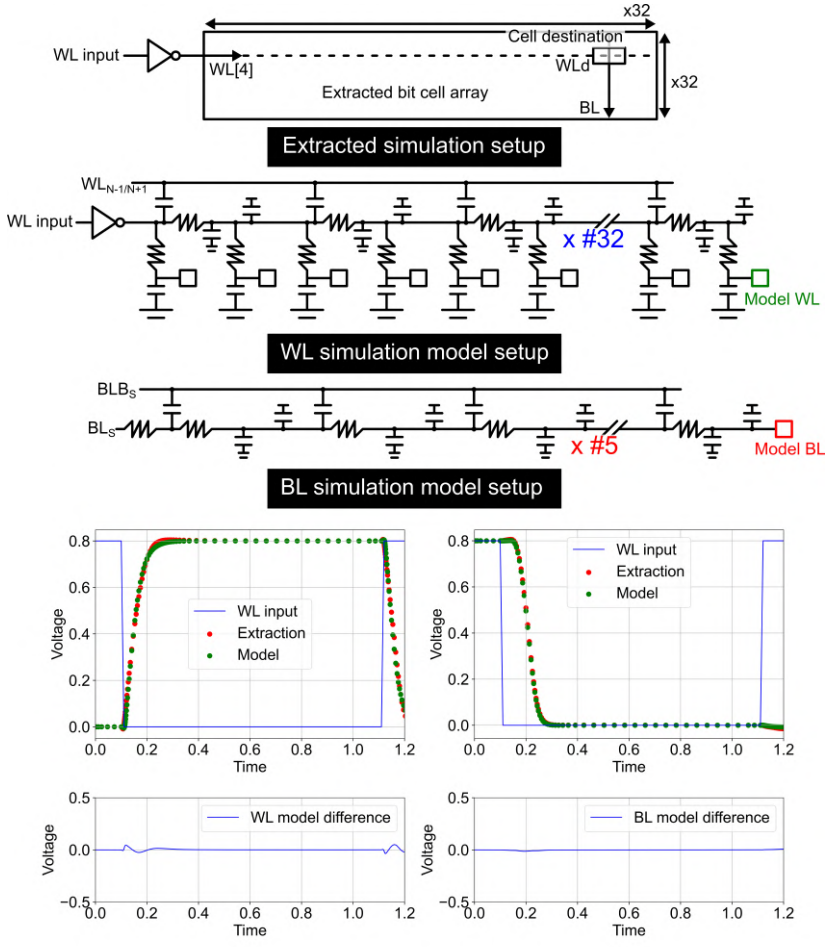


Figure 2.12: Comparison between the extracted and the modeled 32-by-32 6T DIMC array. The data are unit-less to avoid NDA violation.

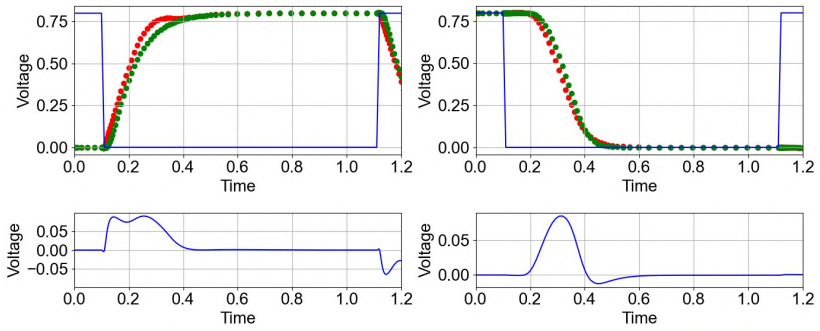


Figure 2.13: Comparison between the extracted and modeled 64-by-64 6T DIMC array on WL[14].

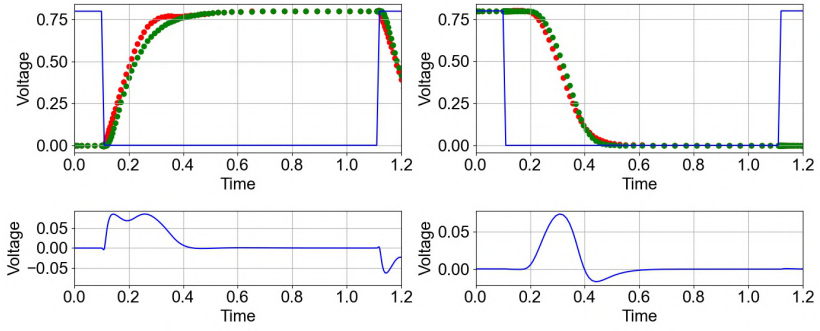


Figure 2.14: Comparison between the extracted and modeled 64-by-64 6T DIMC array on WL[50].

Notably, for the 64-by-64 DIMC array, the lumped model closely matches the extracted netlist, whereas for the 8-by-128 DIMC array, a significant mismatch is observed. This discrepancy arises because the selected WL is the 8th WL, which lies at the boundary of the cell array. The parasitic effects on this node are lower compared to those in the inner word lines, leading to a noticeable delay discrepancy where the lumped model exhibits a significantly longer delay than the extracted netlist.

However, this effect is generally minimal in most cell arrays for two reasons. First, an additional dummy cell ring is typically incorporated into the array design. Second, DIMC architectures are usually designed for the worst-case scenario, which corresponds to the inner word lines.

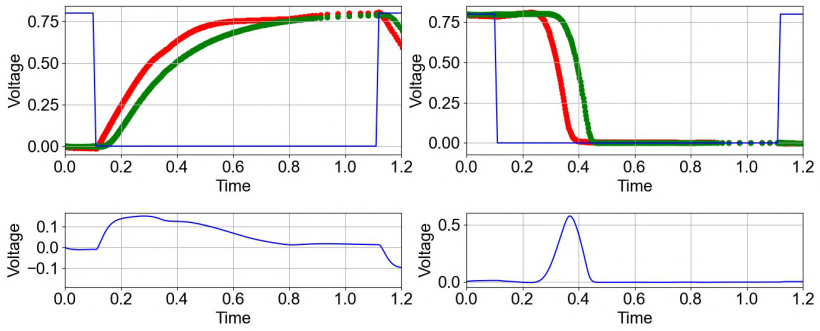


Figure 2.15: Comparison between the extracted and modeled 8-by-128 6T DIMC array on WL[7].

Overall, the waveform comparison at the destination bit-cell demonstrates that the lumped model provides a highly accurate approximation of the extracted netlist. This confirms that the bit-cell can be effectively evaluated without requiring complex extraction steps.

2.3.2 The DIMC trade-off

In this section, we establish the first set of DIMC trade-offs: considering only the bit-cell array, we analyze how DIMC area efficiency trades off against DIMC access speed and energy consumption. Notably, computation is not yet considered, which is why we use the term "accessing" instead of "operational."

DIMC array access can be decomposed into two stages: the WL access stage and the BL access stage. Theoretically, the BL begins to transition once the WL voltage exceeds the threshold voltage of the NMOS transistor. Since WL activity is intrinsically linked to BL activity, both contribute to the critical path of DIMC access.

For DIMC arrays with different aspect ratios, the timing contributions of the WL and BL vary significantly, influencing the overall access speed and efficiency.

WL and BL delay analysis

To analyze the delay contributions of the WL and BL, we first examine the WL delay. The WL setup is illustrated in Fig. 2.16, and the delays are simulated using the lumped model. The two key variables in this analysis are the width

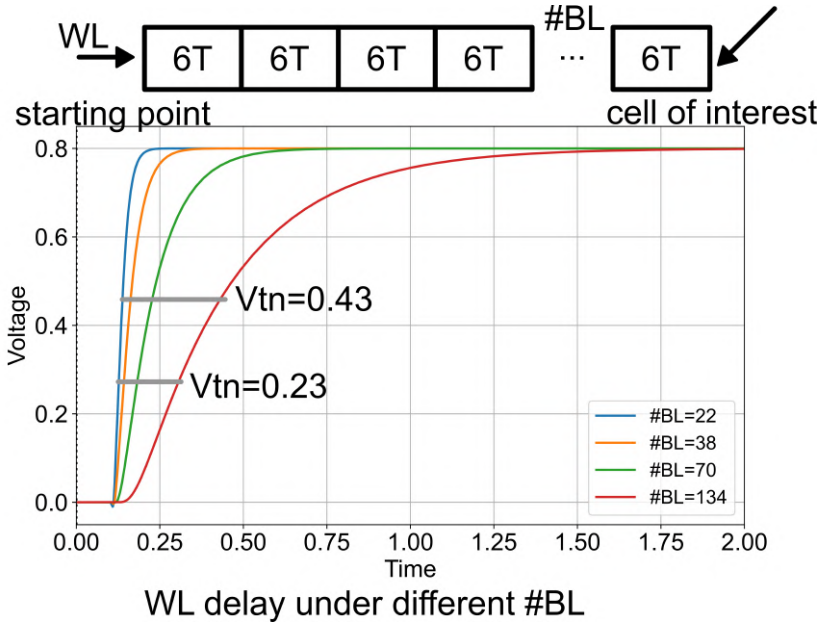


Figure 2.16: Delay analysis setup for the WL.

of the DIMC array, represented by the number of bit lines (#BL), and the threshold voltage of the NMOS pass transistor, V_{tn}. The bit-cell at the end of the WL is selected as the reference cell for evaluation. The WL delay is computed as the time elapsed from the initialization of the WL pulse until the voltage crosses V_{tn}.

Fig. 2.17 presents the delay map for different array sizes and varying threshold voltages (V_{tn}). On one hand, as the array size increases, the WL delay also increases. On the other hand, the impact of V_{tn} on WL delay varies significantly depending on the array size. For arrays with a small number of bit lines (#BL), the effect of V_{tn} is almost negligible. However, for arrays with a large #BL, the impact becomes substantial. As a result, in large DIMC arrays, transistor variation must be considered in timing analysis due to the increased sensitivity to V_{tn}.

The BL delay is derived using a similar setup, with two key differences, as illustrated in Fig. 2.18. First, the BL's source is now located within the bit-cell, while the destination, i.e., the point of interest (PoI), is the input to the DIMC MAC logic. At this stage, an inverter is used to simulate the load of the DIMC MAC logic. Second, for words positioned in the middle of the array, the BLs

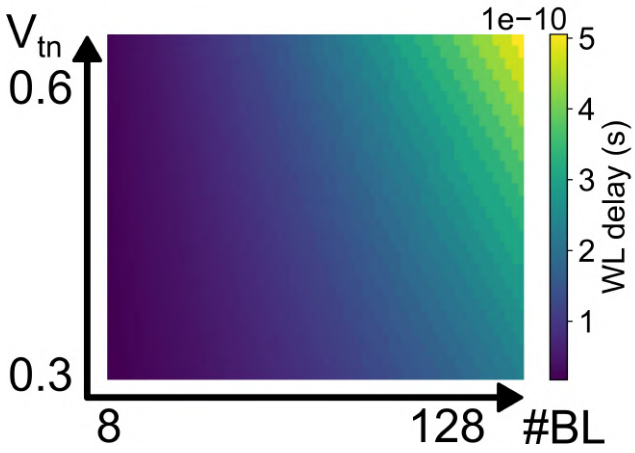


Figure 2.17: WL delay under different array sizes and different V_{tn}.

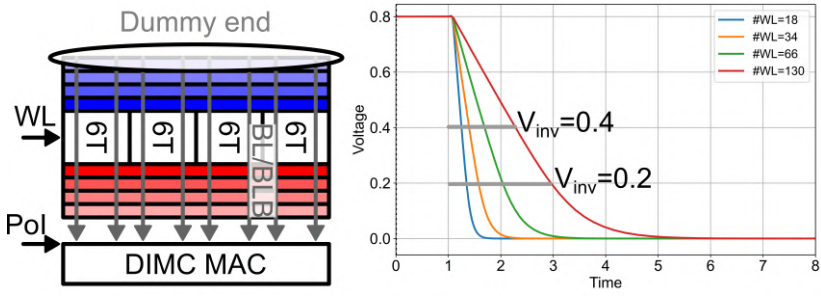


Figure 2.18: Delay analysis setup for the BL.

extend both upwards and downwards. Consequently, the selection of the word line has a significant impact on the BL parasitic structure. To simplify the discussion, we focus only on the word line located in the middle of the array. Fig. 2.18 presents the BL delay under different array sizes and inverter flip voltages (V_{inv}), which can be adjusted by varying the supply voltage of the DIMC MAC logic. Based on this setup, Fig. 2.19 illustrates the BL delay map under different conditions. From this, we can draw an important conclusion: For small array with small #WL, the V_{inv} does not impact the BL delay very much, while for the large array, the impact is significant. As a result, if the DIMC MAC circuit operates under a low voltage, extra BL delay margin must be considered.

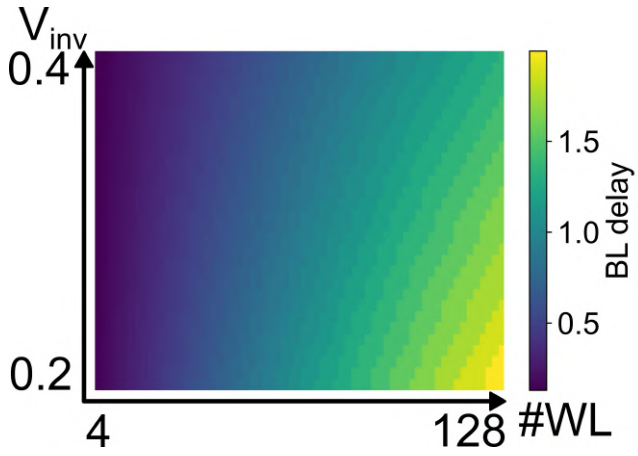


Figure 2.19: BL delay under different array sizes and different inverter flip voltage V_{inv}.

	#BL=8	#BL=64	#BL=128
#WL=4	0.371fJ	0.358fJ	0.357fJ
#WL=64	2.369fJ	2.356fJ	2.355fJ
#WL=128	4.496fJ	4.482fJ	4.481fJ

Table 2.4: Total WL and BL energy per bit accessed for different DIMC array sizes.

WL and BL energy analysis

The WL and BL energy are analyzed using the same setup as the delay analysis. The energy is computed by integrating the power over a single clock period. The key difference between the energy analysis and the timing analysis is that, in energy evaluation, the BL energy is the sum of all active BL energy, whereas in the timing analysis, only the worst BL is considered. Consequently, the BL plays a significantly larger role in overall energy efficiency compared to the WL. Intuitively, for energy-constrained edge applications, optimizing the BL is more critical than the WL to improve energy efficiency.

Table 2.4 presents examples of BL and WL energy consumption for DIMC arrays of different sizes. The total WL and BL energy are summed and then normalized by the number of bits accessed. Two key observations can be made: First, increasing the number of bit lines (#BL) has minimal impact on the total energy per accessed bit. Second, increasing the number of word

lines (#WL) significantly penalizes the total energy per accessed bit. This occurs because, while only one word is selected at a time, the unselected words in the #WL dimension introduce additional parasitic capacitances to the BLs, leading to increased energy consumption. From a pure energy-efficiency perspective, limiting #WL while increasing #BL is the most effective strategy to enhance both throughput and energy efficiency. In practice, this design choice is commonly adopted by designers.

As an aside, it should be pointed out that the leakage of the bit cell is also important to consider. Nevertheless, the leakage of the bit-cell array is less significant in DIMC than in traditional SRAM for two reasons: First, the DIMC typically adopts much smaller array size than the SRAM, and the leakage of the peripheral circuit and the DIMC MAC circuit is greater than the bit cell array. Second, the DIMC MAC circuit uses transistors with lower V_t to reduce the dynamic energy. The direct impact is the increase in leakage in the DIMC MAC circuit. As a result, we do not further discuss the leakage of the bit cell array in this thesis.

Delay versus energy versus area efficiency

We are now ready to establish the relationship among delay, energy, and area efficiency. To evaluate the combined impact of DIMC access delay and energy consumption, we use the energy-delay product (EDP) as a metric and compare it to area efficiency. Both parameters are plotted against different array sizes, as shown in Fig. 2.20. As the array size increases, the area efficiency of DIMC improves but eventually approaches the theoretical limit of the 6T bit cell. This suggests that a relatively large DIMC array is sufficient to maintain high area efficiency. On the other hand, increasing the number of word lines (#WL) has the most significant impact on energy efficiency, whereas increasing the number of bit lines (#BL) has a minimal effect.

To summarize, the first key conclusion for the raw DIMC array design is:

- *A DIMC array with a relatively small #WL but a large #BL is the most optimal configuration for maximizing both area efficiency and energy efficiency.*

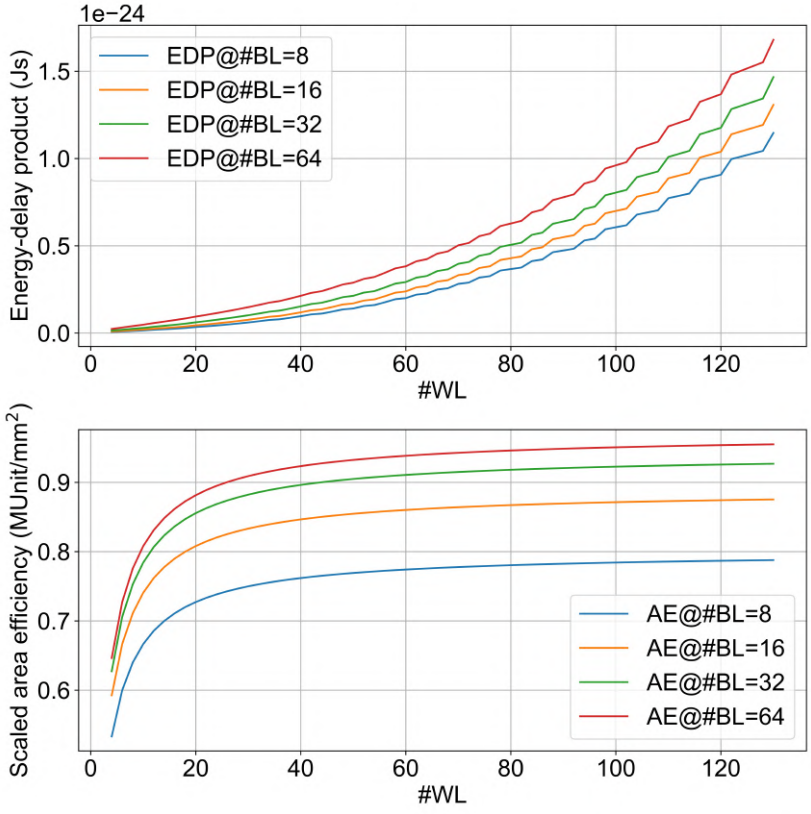


Figure 2.20: The EDP and area efficiency for DIMC array of different sizes.

2.4 Impact of DIMC decoder, write peripheral, and timing control circuit.

In the previous section, we discussed the trade-off between the energy-delay product (EDP) of the DIMC bit-cell array and its area efficiency. However, for the DIMC to function correctly, additional circuits are required. These primarily include the DIMC decoder for the WL, the peripheral circuit for BL pre-charging and cell read/write operations, and the timing control circuit that generates signals to regulate macro behavior. From an area utilization perspective, these circuits contribute to overhead, negatively impacting the area efficiency of the macro.

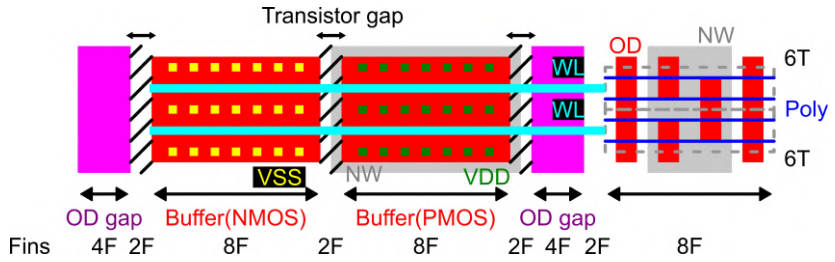


Figure 2.21: Comparison of DIMC decoder layout and thin-film 6T layout.

The DIMC decoder converts the binary WL signal into a one-hot encoded WL signal. There are two main challenges associated with the DIMC decoder circuit:

- The DIMC decoder must fit within the DIMC WL pitch, which corresponds to the width of the thin-film 6T layout along the WL dimension. This imposes stringent layout constraints on the decoder design.
- The last stage of the decoder must supply a significant amount of output current to drive the long WL.

As a result, the DIMC decoder typically occupies a considerable portion of the chip area. As a first-order estimation, we use the area of the WL buffer to assess the area impact of the DIMC decoder. Assuming that the last-stage buffer is the smallest buffer capable of driving a 128-length WL within 100 ps, a 16-fin transistor pair is required. Based on an example buffer layout shown in Fig. 2.21, where the buffer pitch aligns with the thin-film layout pitch, the buffer width is approximately four times that of the thin-film 6T layout. This implies that, in area efficiency computations, at least the width of four dummy cells must be accounted for when considering the impact of the DIMC decoder.

On top of this estimation, additional area overhead remains, as only the buffer area has been considered—not the entire decoder. This further penalizes the macro's area efficiency, necessitating optimizations to mitigate its impact, which will be addressed in the second prototype design (Chapter 3, section 3.2.3).

Similarly, the write peripheral and timing control circuits introduce significant area overhead, further reducing the area efficiency of the DIMC macro. To minimize this impact, shared write peripheral and synchronous DIMC techniques are proposed and will also be discussed in the two prototype designs (Chapter 3, section 3.1.2 and section 3.2.2, 3.2.4).

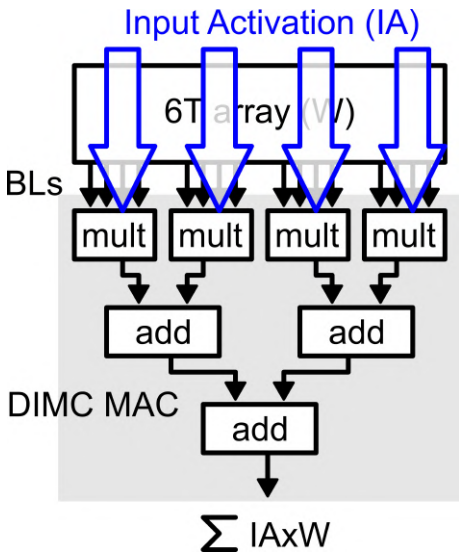


Figure 2.22: Illustration of the DIMC logic.

2.5 The DIMC MAC circuit

In this thesis, the DIMC MAC circuit computes the multiply-and-accumulate results of the DIMC weight (W) vectors and the input activation (IA) vectors. Fig. 2.22 illustrates the basic components of the DIMC MAC logic. It mainly consists of two parts: the multipliers (mult) that calculate the product of the IA data with the W data, and the adder tree that reduces these products into one or more summation values. In essence, the DIMC MAC logic is very similar to the sense amplifier of the SRAM, with the primary difference being whether the output is MAC data or raw data. In this section, we discuss the realization of the DIMC MAC logic with a focus on the speed/energy/area trade-offs.

2.5.1 The multiplication

The multiplier computes the product of the W and the IA. Fig. 2.23 illustrates the process for a 4-bit multiplication. The W value is first received from the WL. Since the multiplication is 4-bit, four BLs are required to provide these data. Each BL is multiplied with a corresponding 1-bit IA value, generating four partial products, P0 to P3. The 1-bit multiplication is implemented using simple AND gates. It is important to note that W must be broadcast to four sets

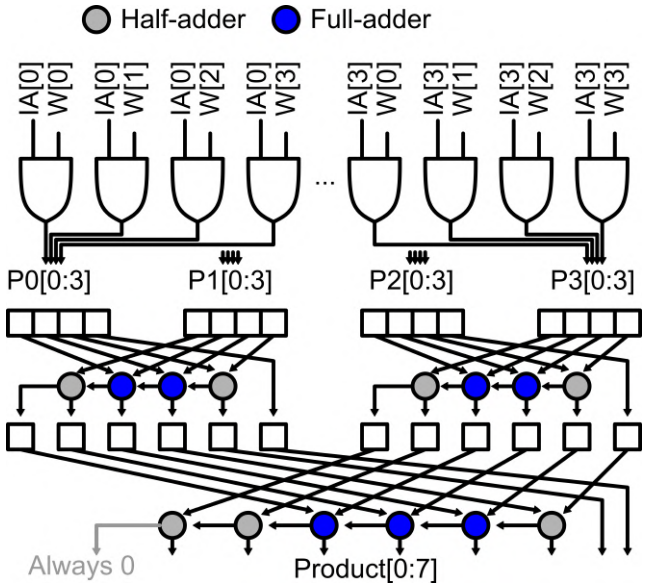


Figure 2.23: Data flow of 4-b multiplication.

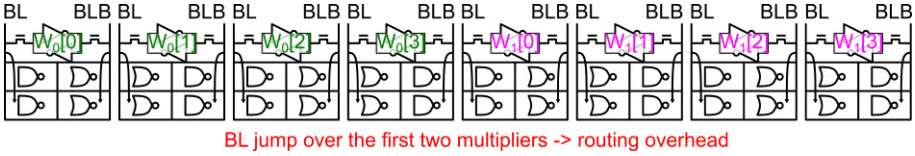


Figure 2.24: The routing congestion issue of the DIMC multiplier.

of AND gates to generate P0 to P3, which impacts the timing of the MAC logic. The 4-bit partial products P0 and P1 are then reduced into a 6-bit intermediate value. Similarly, P2 and P3 are reduced into another 6-bit intermediate value. These intermediate values are further reduced to produce the final 8-bit product. Notably, the carry of the first half-adder is always zero, as the product of two 4-bit numbers cannot exceed 8 bits. The reduction process is implemented using adders, with a ripple-carry adder used in this example. The first two full adders in the reduction process are replaced by half adders to optimize the design.

The design of the multiplier in DIMC presents three main challenges. First, the W values need to be broadcast, which introduces an additional load on the BL. As the number of bits on the BL increases, the speed degrades proportionally.

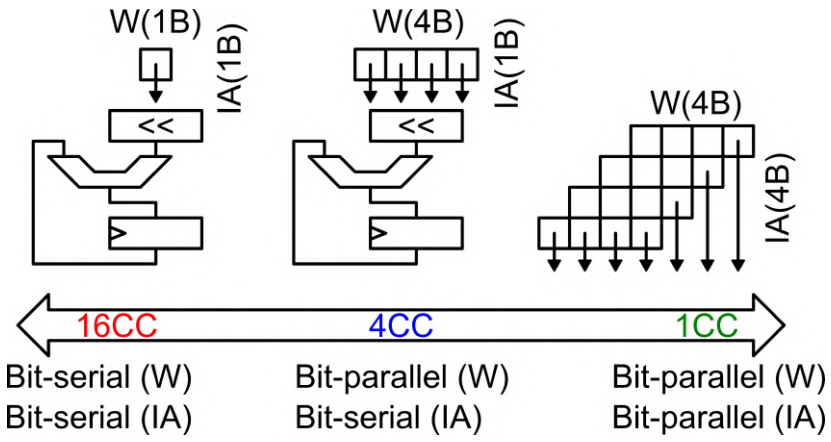


Figure 2.25: Illustration of different multiplication strategies.

Second, the 1-bit multipliers must be placed within the same region as the W data, as shown in Fig. 2.24. This requirement necessitates routing the BL onto higher metal layers toward the outer region of the 1-bit multipliers. For precisions exceeding 4 bits, this routing practice leads to significant congestion near the BL. Third, as illustrated in Fig. 2.24, the reduction logic is highly irregular. The input bits are unaligned, and the bit width increases at each reduction stage. These factors pose significant design challenges, particularly given the narrow width constraints imposed by the BL.

Many researchers have explored an alternative approach for multiplication, known as the bit-parallel/bit-serial scheme [64] (WJ: Add more here later!). The key difference is that, instead of performing the entire multiplication in a single clock cycle, the computation is distributed over multiple cycles to reduce the number of 1-bit multipliers placed near the BL.

As illustrated in Fig. 2.25, multiplication can be categorized into three schemes:

- **Bit-parallel/bit-parallel:** Computes the entire multiplication in a single clock cycle.
- **Bit-parallel/bit-serial:** Multiplies the full-precision W with a 1-bit IA in one clock cycle.
- **Bit-serial/bit-serial:** Computes only the product of a 1-bit W and a 1-bit IA in one clock cycle.

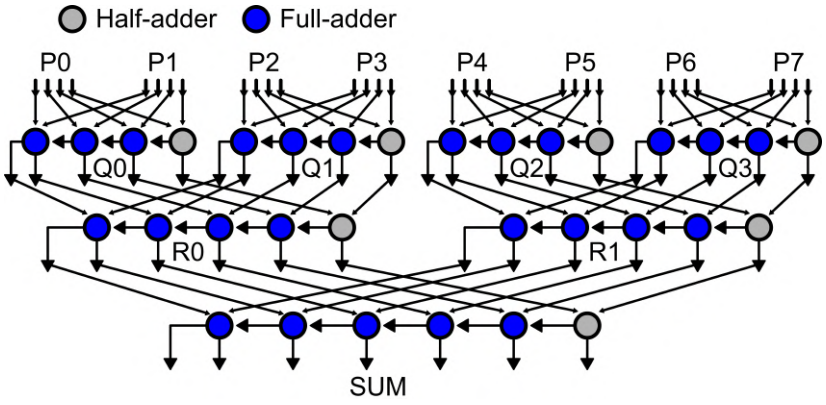


Figure 2.26: The architecture for a depth 8 4-bit adder tree.

Among these, the bit-serial/bit-serial scheme suffers from significant throughput loss, while the bit-parallel/bit-serial scheme offers a trade-off between throughput and reduced logic complexity in the multiplier. Another advantage of the bit-parallel/bit-serial approach is that the multiplier can be implemented using a simple NAND or NOR gate, which can be realized within the 10T cells discussed previously.

Despite its popularity, the bit-parallel/bit-serial scheme is less area-efficient. In fact, as shown in Fig. 2.24, there is typically enough space for two CMOS gates near the BL, making it possible to perform a 4-bit W $\times$ 2-bit IA multiplication in one clock cycle without incurring routing penalties. This observation will be further explored using two techniques in the prototype chip designs (Chapter 3, section 3.1.2 and section 3.2.1).

2.5.2 The adder tree

Following the multiplier, the adders perform the accumulation. DIMC architectures typically utilize a carry-ripple adder, as the precision of the DIMC MAC is generally not larger than 8 bits. Additionally, the bit-serial pattern inherently constrains the adder width. Fig. 2.26 illustrates the structure of an 8-depth, 4-bit adder tree. This adder tree progressively reduces multiple lower-precision values into a higher-precision sum, with the number of layers scaling logarithmically with the number of inputs. A key distinction between the adder tree and the reduction stage of the multiplier is that, in the adder tree, all inputs are equivalent. Because of this, most designers integrate the

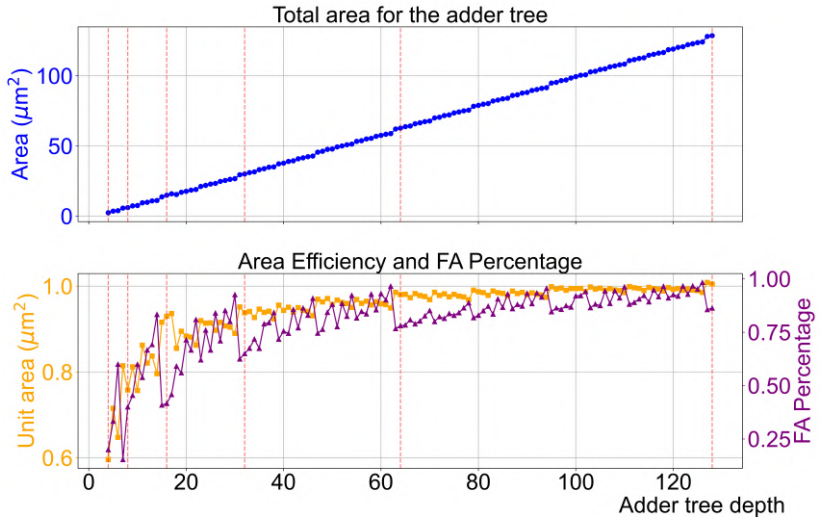


Figure 2.27: The synthesized area and area efficiency of 1-bit adder tree from depth 4 to depth 128.

adder tree close to the DIMC array, as it allows for optimized design replication across multiple instances.

As the depth of the adder tree increases, circuit design complexity rises. This is primarily due to the increasing precision required in deeper stages.

An interesting observation about adder trees is that, for deep adder trees, the optimal number of inputs is not necessarily a power of two. Fig. 2.27 presents the area of a 1-bit adder tree for depths ranging from 4 to 128, along with the corresponding area efficiency. The adder trees are synthesized for rapid comparison, and their area efficiency is defined by the average area occupied per input, referred to as the unit area. Naturally, a smaller unit area indicates a more efficient adder tree.

The red dotted lines in Fig. 2.27 highlight data points where the depth is a power of two. At these locations, the unit area exhibits a local maximum or is close to one, indicating poor area efficiency. Interestingly, the area efficiency of the adder tree strongly correlates with the proportion of full adders (FAs) in the total synthesized gate count. The percentage of FAs tends to be lower when the adder tree depth is a power of two, reducing efficiency. This phenomenon occurs because the FA compresses 3-bit inputs into 2-bit outputs, leading to better efficiency when the number of inputs is a power of three.

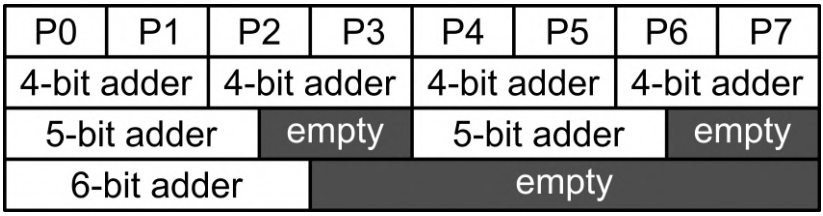


Figure 2.28: The placement difficulty for the adder tree.

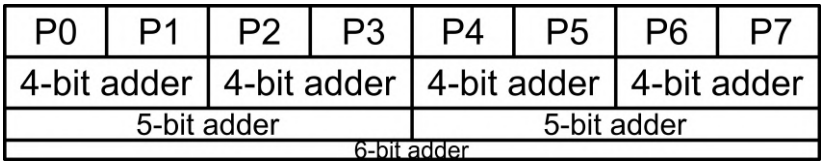


Figure 2.29: Thin adder solution for the placement problem.

However, it is important to note that this conclusion is derived from synthesis using a standard cell library, where the FA is optimized. More importantly, the synthesized adder tree exhibits a highly irregular dataflow, making customization more challenging. Last but not least, the DIMC FA cannot be made one-for-all, as a rigid, standard cell type design will lead to significant placement issue, which will be discussed right next. As a result, most designers continue to use power-of-two adder trees, with customized smaller NAND/NOR/INV gates to alleviate the placement problem.

Another challenge associated with the adder tree is placement complexity, as illustrated in Fig. 2.28. In the first stage, the 4-bit adders have a width equivalent to that of two multipliers, which generate the input values. However, in the second stage, the adders expand to 5-bit width, making them larger than the 4-bit adders. Consequently, the combined width of two 4-bit adders is smaller than that of two 5-bit adders, leading to empty filler regions on the second layer. This issue becomes even more pronounced in the third layer, where only a single adder is required. In most designs, the DIMC must be implemented as a rectangular hard IP block, preventing the utilization of these empty macro areas. This inefficiency significantly reduces the overall area efficiency of the DIMC.

One potential solution to this problem is to stretch the adder tree circuit by making the later-stage adders thinner. As shown in Fig. 2.29, the layout of the later-stage adders is adjusted to be thinner, allowing better utilization of the

empty macro area.

However, this technique presents three key challenges:

- **Limited area savings:** The reduction in area relies on decreasing the height of the later-stage adders. However, this reduction is typically minimal due to hard transistor constraints.
- **Increased layout complexity:** Modifying the height of the adders often necessitates a complete re-generation of the layout, significantly increasing manual design effort.
- **Routing congestion:** Later-stage adders involve a greater number of nodes, leading to higher routing complexity. Reducing their height exacerbates congestion, forcing the use of higher metal layers, which are a valuable resource at the system level.

In this thesis, we explore both the conventional approach and this alternative solution, incorporating optimizations in two prototype designs.

2.5.3 Area, timing, and energy analysis of the DIMC MAC logic

In the previous sections, we discussed the definition and implementation of the DIMC MAC logic. As a customized circuit, its layout must be designed to balance area and power consumption optimally. However, this process is highly time-consuming, resource-intensive, and difficult to automate. A natural alternative is a fully digital implementation using CAD tools. While this approach simplifies the design process, it may lead to suboptimal area and energy efficiency. This is because digital standard cell libraries are designed for general-purpose, irregular logic rather than the highly structured and regular DIMC logic. Nevertheless, performing a rule-of-thumb analysis of the DIMC MAC logic using traditional digital methodologies can help identify critical components that require customization within the DIMC. Meanwhile, less critical components can be efficiently implemented in the digital domain, striking a balance between design efficiency and performance.

The MAC revisited

Let us revisit the MAC logic. Fig. 2.30 illustrates the breakdown of the MAC logic. As previously introduced, it consists of three main components: the 1-bit

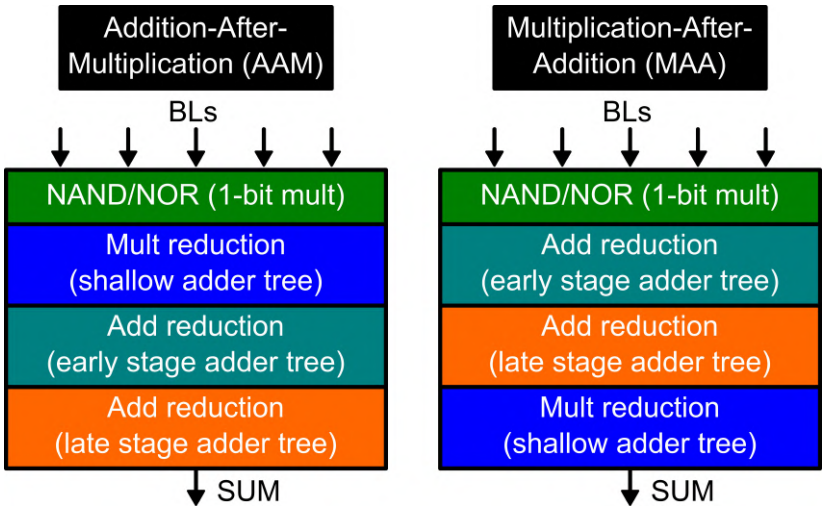


Figure 2.30: Breakdown of the MAC logic.

multipliers, the reduction adder tree for multiplication, and the adder tree for accumulation.

We further divide the accumulation adder tree is intentionally divided into two distinct parts:

- **Early stage adder tree** These consist of highly parallel, low-precision adders.
- **Late stage adder tree** These involve less parallelism but higher precision additions.

Since both multiplication and accumulation fundamentally rely on addition, their computational sequence can be interchanged. For example, the operations can be performed in the following orders:

- Mult-reduction first, followed by add-reduction. This strategy is denoted as Addition-After-Multiplication (**AAM**) strategy.
- Add-reduction first, followed by mult-reduction. This strategy is denoted as Multiplication-After-Addition (**MAA**) strategy.

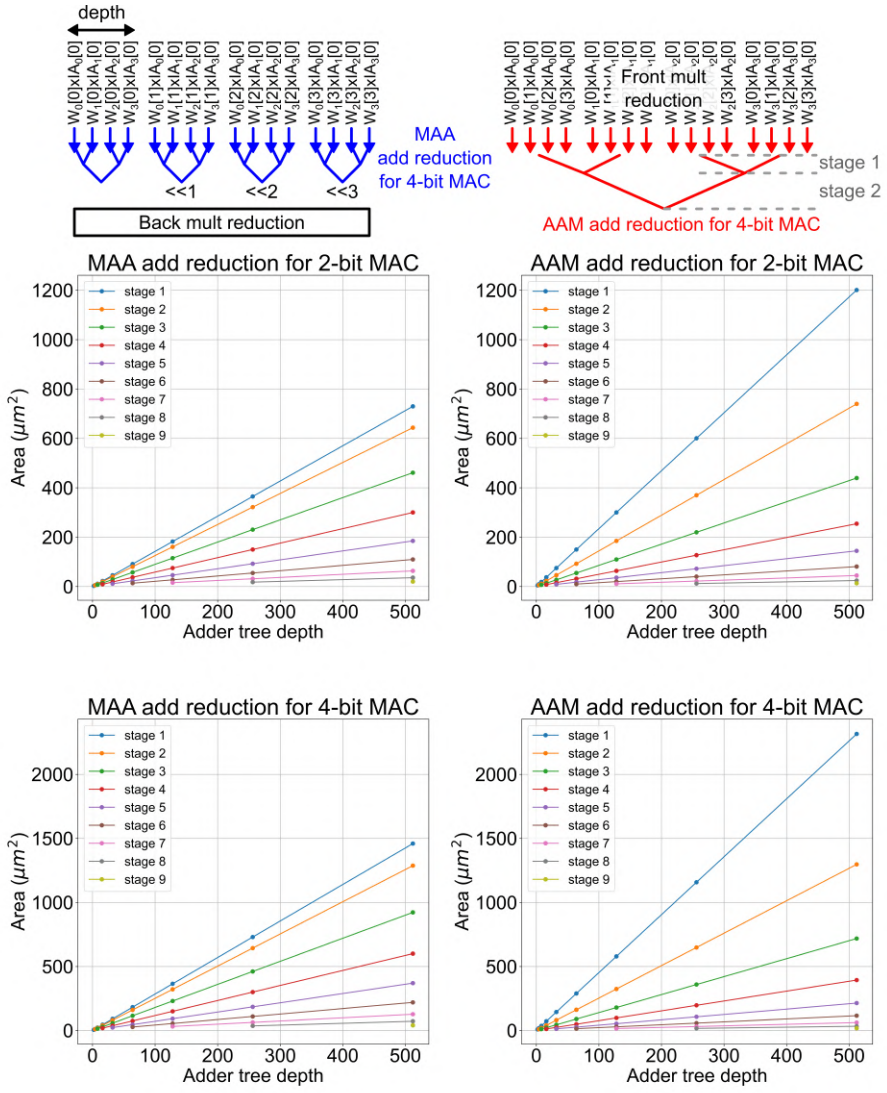


Figure 2.31: The area for different add reduction strategies under different MAC precisions.

By structuring the MAC logic in this manner, we can analyze the area and energy consumption of each part separately, enabling a more precise optimization strategy.

The area cost

As discussed, there are two possible strategies for the MAC. For these two strategies, the role of precision for the operands (weight and input activation) is different:

- **MAA** Higher precision increases the number of parallel adder trees, impacting the mult reduction circuit in the subsequent stages.
- **AAM** Inputs are treated as full-precision numbers, requiring a single adder tree for the add reduction.

Fig. 2.31 presents the area cost of the add reduction under different strategies and MAC precisions, with separate area evaluations for different adder tree depths. Two key observations emerge:

- **MAA add reduction is more area-efficient** than AAM add reduction, making it a more cost-effective choice.
- For deep adder trees, the area distribution varies significantly across stages. Notably, the first 3-4 stages contribute the most to the total area, suggesting that special optimization for early-stage adders could be beneficial.

Another advantage of MAA add reduction is its simplified adder tree design. Although the number of trees is higher, the bit width of the first adder in the front add reduction remains at 1-bit. More importantly, the trees for different bit positions are identical, allowing for efficient reuse of customized circuit designs and layouts.

Fig. 2.32 illustrates the area cost of different mult reduction circuits across varying MAC precisions. For the bit-parallel/bit-serial scheme, the AAM mult reduction is eliminated since a 1-bit product does not require further reduction. However, MAA mult reduction is essential to combine accumulation results from the MAA add reduction adder trees. Unlike adder tree for the add reduction, where area scales linearly with depth, the area required for MAA mult reduction grows at a slower rate.

In short, we could summarize a DIMC MAC area overhead table, as shown in Table.2.5. In this thesis, both strategies will be explored with two prototype chips with real silicon implementation.

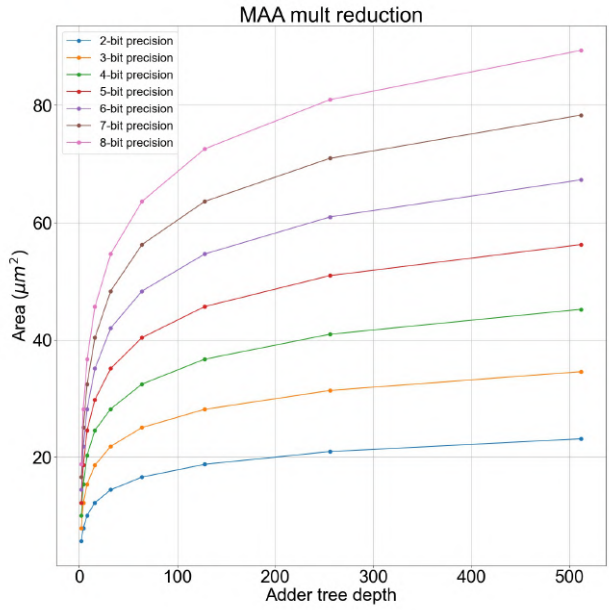


Figure 2.32: The area cost for different mult reduction circuit.

Circuit part	MAA add reduction	MAA mult reduction
Area overhead	Moderate	Low
Circuit part	AAM add reduction	AAM mult reduction
Area overhead	High	None

Table 2.5: Conclusion for the DIMC MAC area overhead.

The timing

In this section, we analyze the timing of the DIMC MAC circuit. Fig. 2.33 illustrates the critical path for the add-after-mult and mult-after-add strategies. In the **add-after-mult (AAM)** scheme, the critical path extends from the least significant bit (LSB) input of the first-stage add-reduction, propagates through the carry-out of the first stage, and continues toward the most significant bit (MSB) input of each subsequent stage. Assuming that the delays of the full-adder and half-adder are equivalent, and that the carry-out and sum port delays are also equivalent, the delay of the AAM scheme can be expressed as:

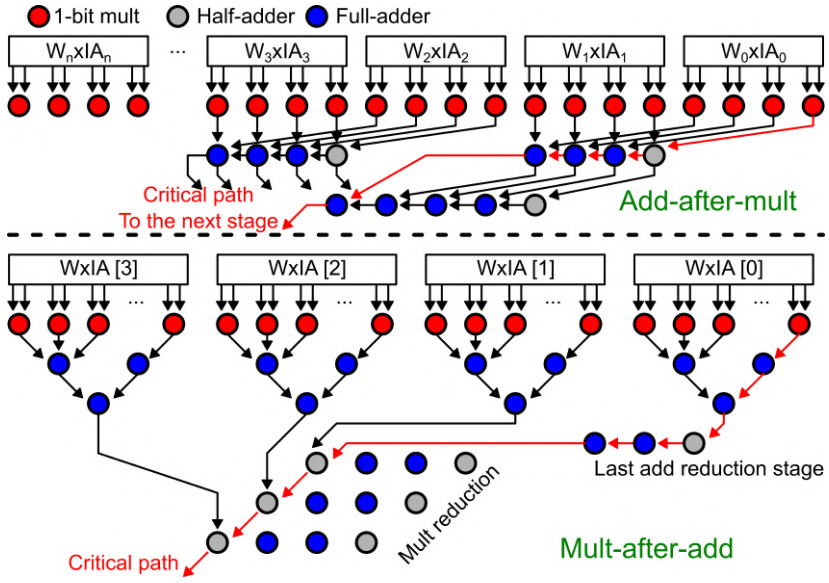


Figure 2.33: The critical path for add-after-mult and mult-after-add strategies.

$$Delay_{AAM} = Prec \times Delay_{FA} + (\log_2(\#inputs) - 1) \times Delay_{FA} \tag{2.2}$$

The first term $Prec \times Delay_{FA}$ corresponds to the delay contributed by the last stage carry ripple adder of the add reduction, and the second term $(\log_2(\#inputs) - 1) \times Delay_{FA}$ is the number of stages minus one.

For the **mult-after-add (MAA)** strategy, the critical path is more complex. Since the add reduction circuit is a 1-bit adder tree, the optimal circuit structure differs from that of a conventional multi-bit adder tree. A key difference arises from the compression efficiency of full adders compared to half adders. In a 1-bit adder tree, a full adder can process three input bits, making it more efficient in terms of compression rate. This is not a concern for the multi-bit adder tree used in the add-after-mult strategy, where full adders primarily operate within multi-precision adders. However, in the 1-bit adder tree, the first stage consists only of half adders, resulting in no initial compression.

One possible optimization is to replace the half adders in the first stage with full adders. The subsequent stages then follow a structure similar to that of the

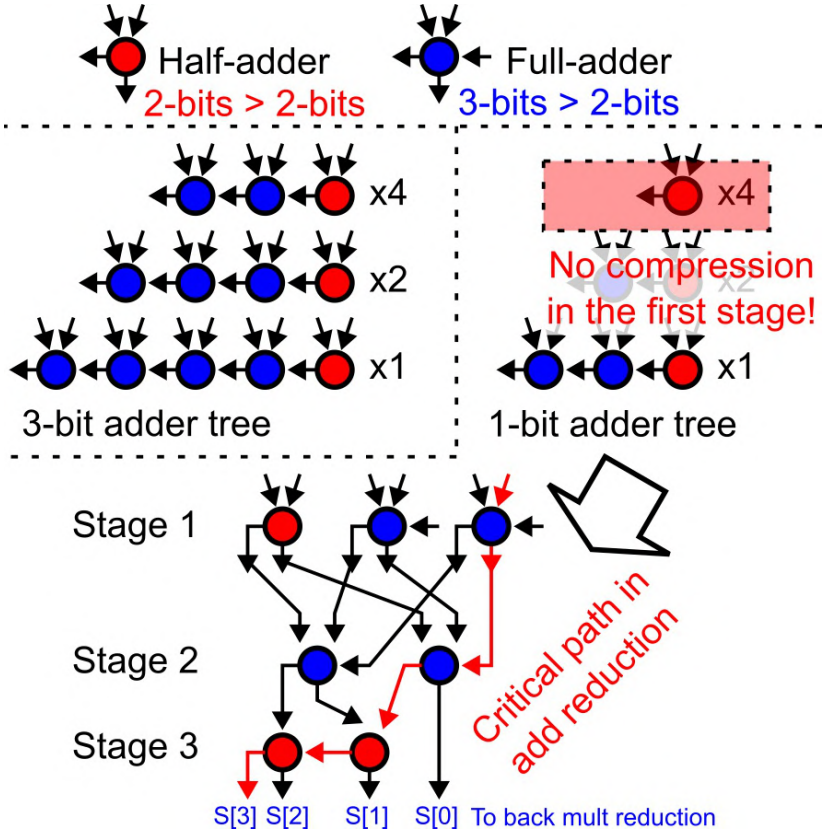


Figure 2.34: The difference between the 1-bit adder tree and the multi-bit adder tree.

mult reduction circuit. This approach reduces the overall gate count and area.

The critical path of this optimized architecture follows these stages:

1. From any input in the first stage of full adders,
2. To the full adder in the second stage,
3. To the last add reduction stage,
4. To the MAA mult reduction.

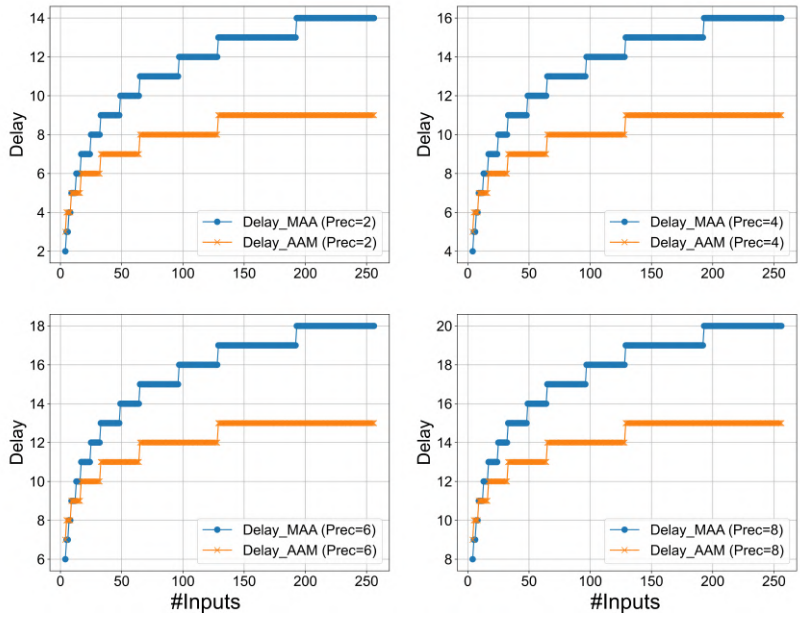


Figure 2.35: The delay difference between AAM and MAA for different numbers of inputs and precision.

Although the critical path of the add reduction already reaches the most significant bit (MSB), it must still propagate to the MSB part of the mult reduction circuit, as shown in Fig. 2.34.

The delay of the **mult-after-add (MAA)** scheme can be expressed as:

$$\begin{aligned} Delay_{MAA} = &(Prec-1) \times Delay_{FA} \\ &+ (\log_2(\#inputs) - 2) \times Delay_{FA} \\ &+ \log_2(\#inputs/3) \times Delay_{FA} \end{aligned} \tag{2.3}$$

Compared to the AAM delay, the MAA delay consists of three additional terms:

- $(Prec-1) \times Delay_{FA}$: Represents the delay of the back mult reduction circuit.
- $(\log_2(\#inputs) - 2) \times Delay_{FA}$: Represents the delay of the last-stage carry-ripple adder in the add reduction circuit. The reason this term is

Strategy	MAA	AAM
Speed	Slow	Fast

Table 2.6: Conclusion for the DIMC MAC timing.

not $(\log_2(\#inputs) - 1) \times Delay_{FA}$ is that the least significant bit (LSB) of the last-stage adder can be generated without requiring a full adder or half adder, as illustrated in Fig.2.34.

- $\log_2(\#inputs/3) \times Delay_{FA}$: Represents the number of stages where full adders are used in the first stage.

Clearly, the MAA strategy incurs a higher delay than the AAM strategy. For comparison, Fig. 2.35 presents the delay evaluation for varying numbers of inputs and precision levels. When the number of inputs is small, the delays of the MAA and AAM strategies are similar. However, as the number of inputs increases, the AAM strategy achieves significantly lower delay than the MAA strategy.

In summary, we conclude the timing analysis of the DIMC MAC logic, as detailed in Table 2.6.

The energy

In this section, we analyze the energy consumption of the DIMC MAC circuit. In general, the energy consumption of DIMC is primarily determined by the dynamic energy of the internal nodes within the MAC circuit. However, since the DIMC MAC typically accumulates a large number of inputs, it results in a long critical path. To address this, pipelining is often introduced to increase the throughput of the DIMC macro. Consequently, the energy consumption of the inserted pipeline registers must also be considered in the overall energy analysis.

Let us first examine the raw energy consumption of the DIMC MAC circuit. Fig. 2.36 and Fig. 2.37 illustrate the power consumption and energy efficiency of the AAM and MAA strategies under different conditions. The power values are derived from the synthesized MAC power, using 50 random input values to estimate the activity level of each node. The operating speed is set to 200 MHz.

Several key observations emerge from the analysis. First, the energy efficiency of the AAM strategy varies with precision. Lower-precision AAM MAC implementations are more efficient than higher-precision ones. In contrast, precision has less impact on the energy efficiency of the MAA MAC.

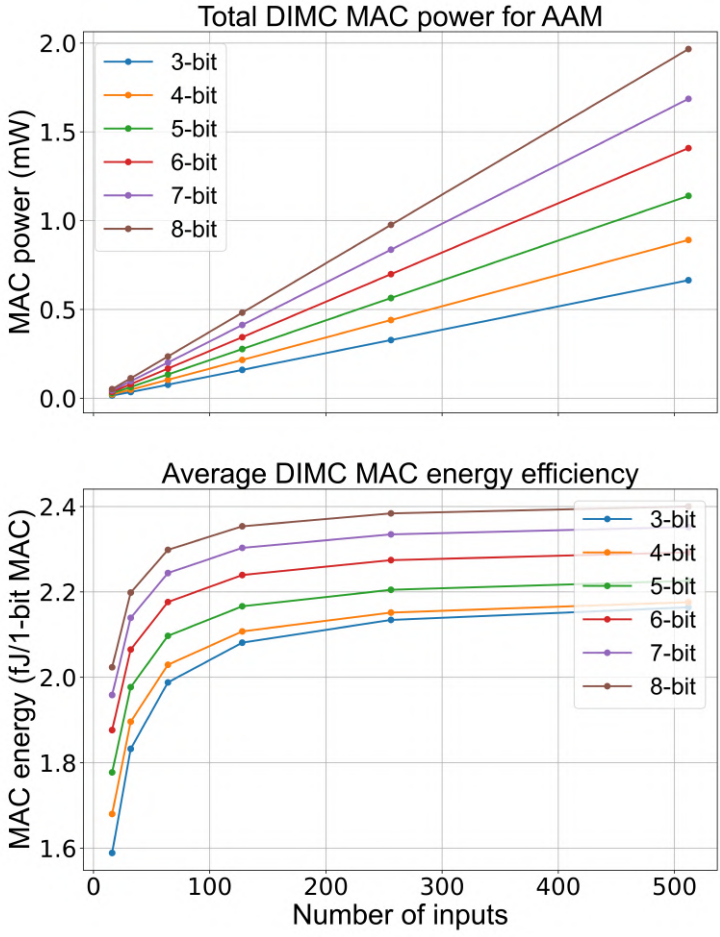


Figure 2.36: The power and energy efficiency of AAM strategy for different number of inputs and precisions.

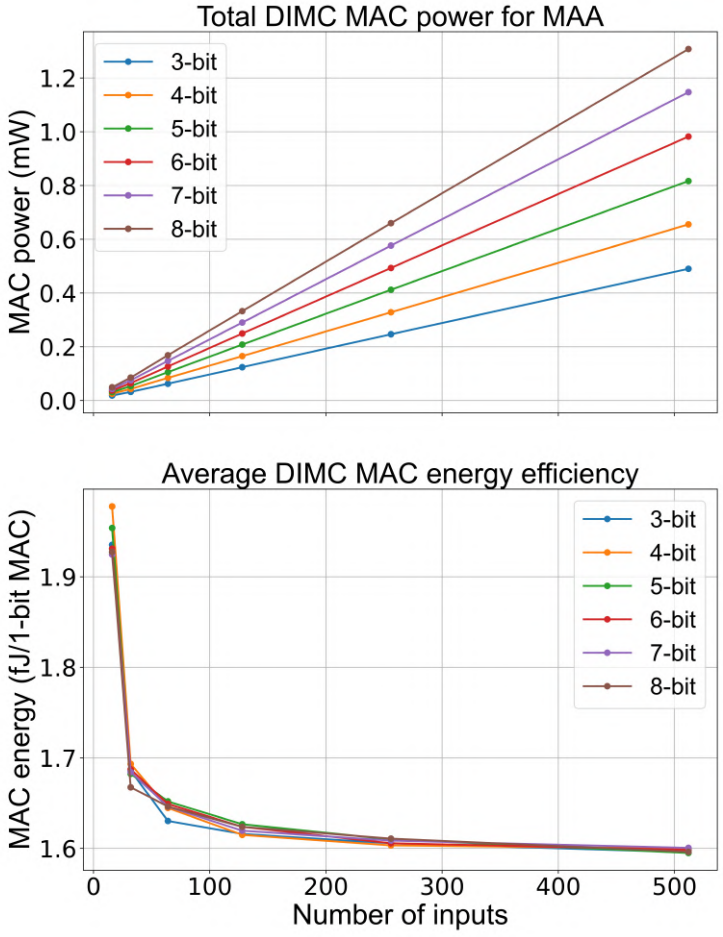


Figure 2.37: The power and energy efficiency of MAA strategy for different number of inputs and precisions.

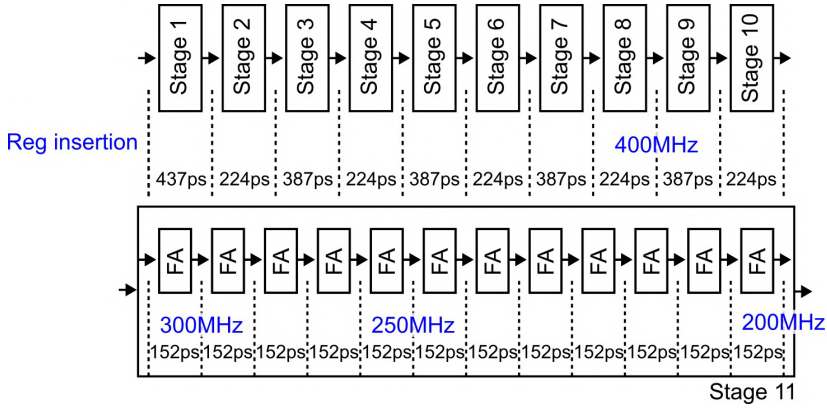


Figure 2.38: The critical path of 8-bit 2048 inputs DIMC MAC in AAM strategy.

The number of inputs also affects AAM MAC and MAA MAC differently. For AAM MAC, increasing the number of inputs reduces energy efficiency, whereas for MAA MAC, more inputs lead to higher energy efficiency. Despite these variations, a Base Energy Efficiency (BEE) limit of approximately 1.6 fJ per 1-bit MAC operation is observed across all cases. When scaled to 8-bit weight precision (WP) and input precision (IP), this results in an estimated MAC Energy Efficiency (EE) of:

$$EE := 2 \times (BEE \times 10^3 \times WP \times IP)^{-1} = \frac{1000 \times 2}{1.6 \times 8 \times 8} \approx 20 \text{ TOPs/W}$$

under nominal conditions, where the factor of 2 accounts for both multiplication and accumulation operations. The BEE is scaled to pJ per 1-bit MAC to make the EE in the unit of TOPs/W.

Apart from the raw DIMC MAC energy, the energy associated with registers must also be considered. This primarily consists of two components: the clocking energy and the intrinsic energy consumption of the registers.

To analyze this, we evaluate an 8-bit DIMC MAC with 2048 inputs under different timing constraints. The critical path is illustrated in Fig. 2.38. When no registers are inserted, the circuit operates at 200 MHz, and only the output registers contribute to energy consumption. However, at higher operating frequencies, additional registers must be inserted at intermediate stages to meet timing constraints. Consequently, the overall register energy consumption is directly proportional to the number of inserted registers.

Speed	200MHz	250MHz	300MHz	400MHz
Register energy	8.6fJ	15.0fJ	18.5fJ	63.6fJ
Energy/MAC	4.2aJ	7.3aJ	9.0aJ	31.1aJ

Table 2.7: Register energy of 8-bit 2048 AAM MAC under different speed constraints.

Strategy	MAA	AAM
Low-depth energy	High	Low
High-depth energy	Low	High

Table 2.8: Conclusion for the DIMC MAC energy.

Table.2.7 shows the register energy for different speed constraints. It increases drastically as the speed increases. Although it is still relatively small compared to the MAC, there is also the clock related energy, that is proportional to it. Such impact is more visible when evaluated on the system level. At this point, we only need to view the register as a fixed amount of extra overhead for a given clock frequency.

In short, when comparing the energy consumption of the MAA and AAM MAC, we have the conclusion as shown in Table.2.8

2.6 Conclusion

In this chapter, we discussed the modeling of DIMC macros. Starting from the choic of the bit cells, to the choice of the array dimension, to a detailed analysis on the DIMC MAC circuit. The main takeaway messages for this chapter are:

- Although the non-6T bit cell has higher throughput potential, the area efficiency compared to 6T bit cell, especially the foundry 6T bit cell, drops significantly. For energy efficient applications, the 6T bit cell is the best choice.
- A DIMC array with a relatively small #WL but a large #BL is the most optimal configuration for maximizing both area efficiency and energy efficiency.
- The DIMC peripheral circuits, including the decoder, write/read assist circuit, and other timing configuration circuit, need to be optimized to prevent the loss of area efficiency.

- The DIMC MAC circuit can be realized with two strategies: the multiplication-after-addition (MAA), and the addition-after-multiplication (AAM). From the energy and layout perspective, the MAA strategy is better thanks to its simpler adder tree structure. From the delay perspective, the AAM strategy is better because of the integration of multiplication in the adder tree structure.
- The adder tree part of the DIMC MAC circuit can be further decomposed into the early stages and the late stages. The early stages are highly parallel with simpler shallow adders, while the late stages are less parallel with more complex high precision adders. Applying different optimization strategies to them can save us energy without overwhelming design effort.

In the next chapter, we will design two prototype DIMC macros in 16nm FinFET technology.

Chapter 3

DIMC macro prototype designs

In this thesis, we develop two phototype DIMC chips. The first prototype is a DIMC macro featuring logic-rule DRC compliant SRAM bit-cells and incorporates several innovative techniques to minimize energy consumption. A novel reverse pre-charge read scheme is implemented, significantly reducing the read energy of the DIMC. Additionally, a charge reuse mechanism is applied to the bit-lines to further enhance overall energy efficiency. The MAC logic circuits are carefully optimized using a semi-standard cell approach, which boosts the area efficiency by increasing logic density within the DIMC macro. As a result, this design achieves an impressive 23.8 TOPs/W energy efficiency for 8-bit MAC operations and a throughput efficiency of 0.364 TOPs/mm^2. This first prototype was successfully validated and published at ESSCIRC 2023.

Building upon the success of the initial design, the second DIMC macro introduces several key advancements. It is made for edge UNet applications. First, the SRAM bit-cells are replaced with high-density pushed-rule 6T bit-cells, which significantly enhance the area efficiency of the macro. Second, a synchronous operational approach is adopted, greatly improving both the speed and robustness of the DIMC functionality. Finally, the MAC logic circuits are further refined with a fully customized layout, enabling even higher performance and energy efficiency. This enhanced design delivers an outstanding energy efficiency of 124 TOPs/W for 4-bit MAC operations and is capable of operating at frequencies exceeding 200 MHz. This second-generation DIMC macro will be utilized in future DIMC-based SoCs for comprehensive system-level validation, marking a significant step forward in energy-efficient edge AI

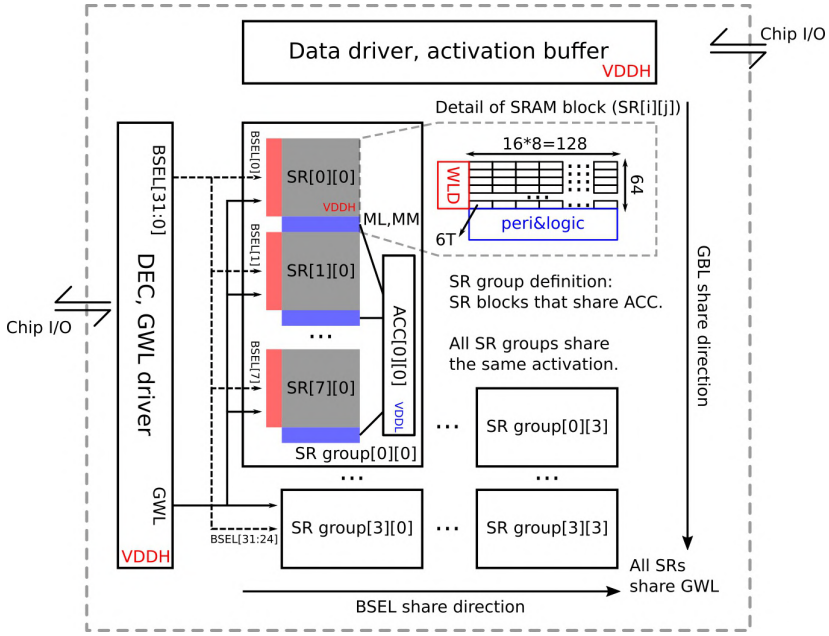


Figure 3.1: Architecture of the first prototype.

computing solutions.

3.1 First prototype

In this section, we introduce the first prototype DIMC macro design. This chip is designed for small 8-bit CNN models with 100KB model size, with around 0.5mm^2 core area. The focus of this prototype is finding energy saving techniques for the DIMC reading and computation. *The text of this section is based on the ESSCIRC 2023 publication.*

3.1.1 Chip architecture

Fig.3.1 shows the architecture of the proposed macro. It consists of 32*4 SRAM blocks (SR[i][j]) of 1kB each. Each SR encompasses word line drivers (WLD), local periphery and logic. In each SR, there are 1kB 6T unmodified bit-cells, which are arranged into 64 rows containing 128 bits each. Below the bit-cell

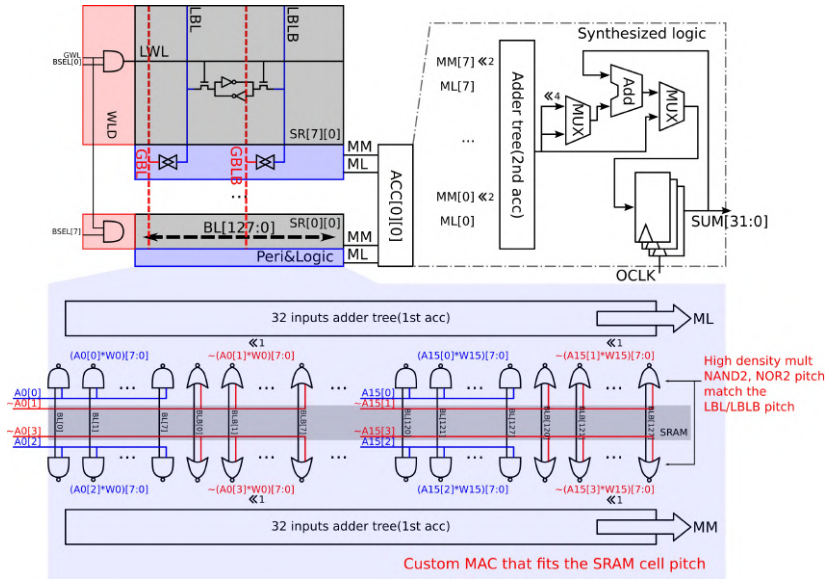


Figure 3.2: Details of SR and ACC blocks.

array, there are 16 MAC units. Partial multiplications are calculated in these units and split into the mac MSB (15b MM) and LSB (15b ML). ML and MM are calculated on two sides of the BL, leading to a very compact physical structure of the MAC units. The SRs are further combined into 16 SR groups: each column of 8 SRs makes up an SR group. In each SR group, an accumulation block (ACC[i][j]) is used to perform further accumulations on the ML and MM generated by the SR in the SR group. The results are buffered locally when partial sums are needed. On the top level of the DIMC macro, a global data driver and activation buffer are provided. A global address decoder (DEC) and global word line (GWL) drivers are used to generate global control signals for the macro. The system is split into two power domains: the VDDH domain, which supplies the bit-cells, WLD, DEC GWL driver, data driver and, activation driver and the VDDL domain, which supplies the local periphery and logic, and the ACC blocks.

Fig.3.2 details the SR and ACC blocks. To achieve high density, a 6T array with custom MAC is used. Each SR produces 16 8b weights (W) in every clock cycle. The input activation (IA) driver delivers 128 (4b) IA to all 16 SR groups, which are further divided equally into 8 parts for each SR in the SR group. In each SR, the received IAs are then MAC-ed with the W. The Ws are read on the local bit-line (LBL/LBLB) and sampled by static NAND2/NOR2 gates.

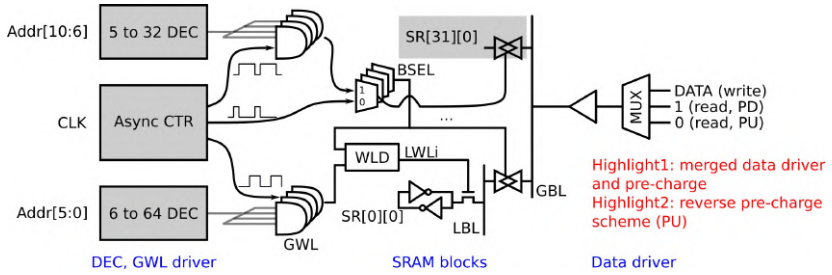


Figure 3.3: Periphery of the macro. Data driver and pre-charge circuit are merged. Reverse pre-charge scheme is used to lower the read energy.

To match the narrow LBL/LBLB pitch, this logic is optimized compared to standard cells in two ways: 1) power and ground contacts are shared because only NAND2 and NOR2 gates are used. 2) weight pins of NAND2/NOR2 gates are directly connected to the LBL/LBLB using the lowest metal layer, saving space for further routing. For each LBL/LBLB, 4 NAND2/NOR2 gates are used as 1b multiplier. All LBL/LBLBs in an SR produce 4*16 8b-products. These products are further split into two parts. In the first part, 32 products generated by the first 2b of IA are accumulated by the first adder tree, generating the LSB part (ML). The second part runs in parallel to the first part, generating the MSB part (MM) of the MAC results. These adder trees are also customized. They use customized logics cells with 4.5 tracks which are two times smaller than the normal 9 track standard cells. Next, each ML and MM generated by an SR group are shifted and accumulated further by second stage accumulation blocks (ACC) and buffered for further operations. These are synthesized using the traditional digital flow. In total 16 32b outputs (SUM[31:0]) are computed each clock cycle for the macro.

Fig.3.3 shows the periphery of the macro. It is based on a hierarchical local block-based SRAM design. This technique, see e.g. [18], allows to deal with variability more efficiently by using short bit-lines with limited area overhead. In each local block, the periphery includes only WLD and LBL to GBL connectors. Globally, a decoder (DEC) generates block select (BSEL) signals for SR selection, and GWL signals for within SR word selection. A control block (CTR) generates the timing signals. To reduce the area, the data driver and the pre-charge circuit are merged. More importantly, a reverse pre-charge scheme (PU) is implemented, which will be explained in detail later.

3.1.2 Energy saving techniques

Support for weight stationary data flow

We propose to implement a weight stationary data flow in our DIMC macro to reduce the weight read energy consumption. The weight stationary data flow is an often-used scheduling plan for CNN algorithms, especially for IMC. Different from other IMC macros, our macro does weight stationarity in two levels. Each MAC in an SR is shared over 64 weights locally, providing the first level of weight stationarity. On the bit-line, one word is read and buffered temporarily, providing the second level weight stationary at low cost. In the 6T array, a BL can store the value, i.e., keep its charge, for a period of time thanks to its relatively high parasitic capacitance. To reuse the charge on the LBL, instead of pre-charging the LBL after each SRAM read, the LBL are left floating until the next read/write cycle. In this macro, the LBL has a parasitic of 4fF. When VDDL=0.4V, the LBL stores 1.6fC charge. Suppose that in the average case, half of the bit-cells connected on this LBL store the reversed value, leading to leakage on the node. The total leakage current is then 25pA (leakage per 6T) * 32 (half of bit-cells) = 0.8nA, leading to around 1us valid time for the values on the LBL. During this 1us window, no further read operation on the bit-cells is needed if the same data is used.

The mapping of the weight stationary dataflow to our DIMC macro is demonstrated in Fig.3.4.a and 3.4.b showing two use cases. In case one when FX*FY*C >128 (Fig.3.4.a), W needs to be stored in multiple addresses of an SR group. It takes at least 2 steps to compute the output. In this case, weight stationarity can be achieved at the cost of output partial sum generation. For AIMC, to keep the accuracy of the model, the convolution needs to be completed for an output before truncated by the ADC, which prevents the use weight stationary dataflow in this case. Unlike AIMC, our macro provides a full precision partial sum, causing no precision lost from truncation of that partial sum. In this example, the 128th weights are read first, and the partial sum of the complete layer is computed and stored temporally. After that, the next 128th weights are read, and the MAC result is computed and accumulated with the partial sums generated from previous cycles. In case two when FX*FY*C <129 (Fig.3.4.b), W can be stored in a single address, which allows W temporal reuse without having partial sums.

Reverse pre-charge scheme

To further reduce the dynamic read energy, we propose a reverse SRAM pre-charge scheme to reduce the swing on the LBL. Instead of normal VDDH, the

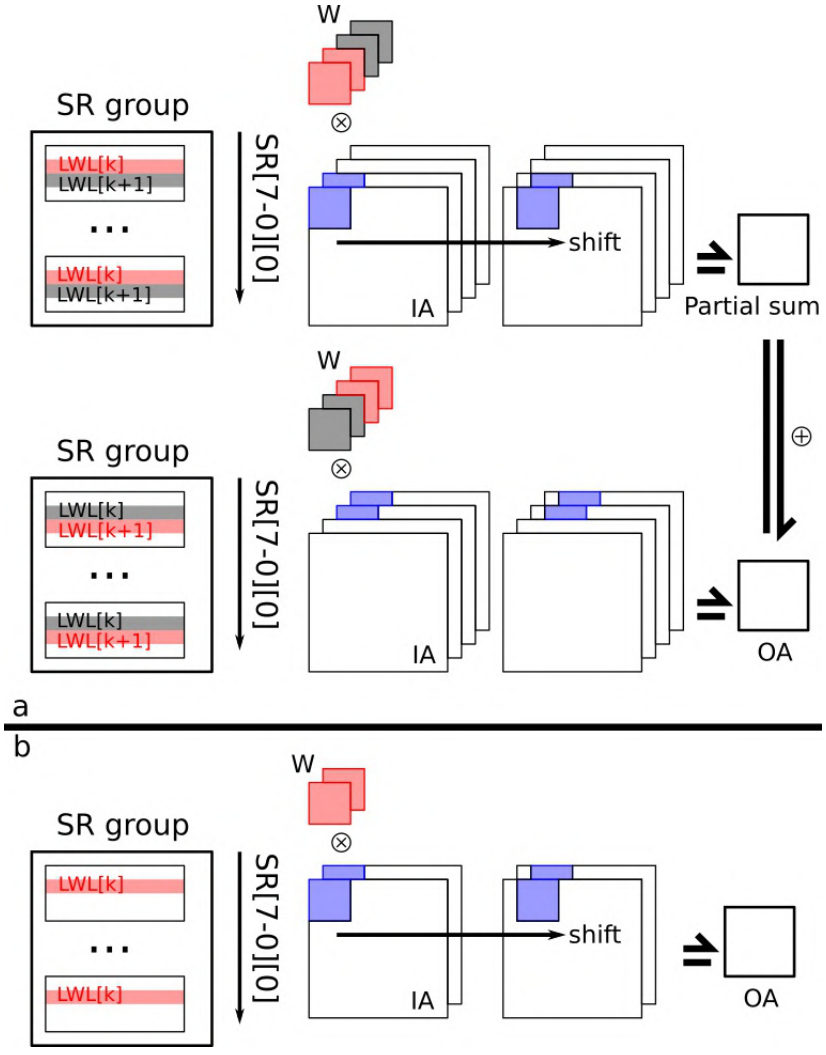


Figure 3.4: Applying weight stationary dataflow to the macro.

	# WL	BL parasitic capacitance	Number of destructive read
Case 1	64	4fF	0
Case 2	128	8fF	0
Case 3	256	16fF	2

Table 3.1: 200 points Monte-Carlo simulation for reading.

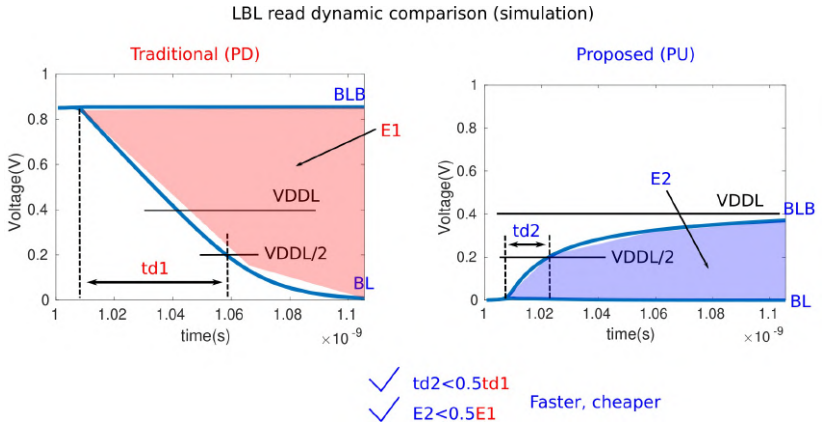
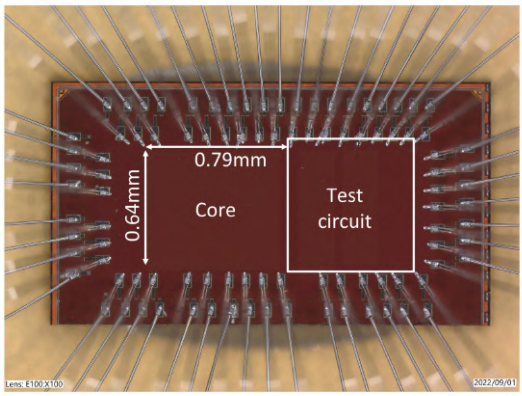


Figure 3.5: Comparison between the traditional pre-charge scheme and the proposed reversed pre-charge scheme.

LBL is pre-charged to VSS. Normally in big SRAM macros, this introduces bit-flipping when the number of word lines is large. This is because when the number of word lines is large, LBL has a large parasitic capacitance. During read, it takes a long time for the LBL to settle, increasing the chance of destructive read under variability induced worst case circumstances. Table.3.1 shows the read robustness for different number of word lines. For each word line count, parasitic extraction is performed, and a 200 points Monte-Carlo simulation is run. When the word line count is lower than 128, i.e., LBL parasitic smaller than 8fF, no bit flipping happens. When the word line count is large, destructive read could happen. For instance, when the word line count is 256, we observe 2 failed cases. In our design, the word line count is 64, which is safe with over 6-sigma safety margin.

There are two benefits of the proposed pre-charge scheme. First, the voltage swing on the LBL is limited to VDDH-Vtn thanks to the NMOS pass transistor for 6T read out. Compared to the normal full swing bit-line, the energy consumption on bit-line is reduced from 0.5*C*(VDDH2) to 0.5*C*(VDDH-Vtn)*VDDH, giving 50% energy saving for standard Vt devices. Second, when



Summary	
Technology	16nm
Core area(mm^2)	0.64mm*0.79mm =0.5mm^2
Macro size	128kB
Precision(I/W/O)	8/8/32
Supply(V)	0.6-1(VDDH) 0.28-0.75(VDDL)
Max frequency	91MHz@1V,0.75V
TP(TOPs)	0.182
Read energy (fJ/Byte)	76fJ(20MHz @0.6V,0.28V)
MAC energy (pJ/MAC)	0.08pJ(20MHz @0.6V,0.28V)
Efficiency (TOPs/W)	20.62(20MHz @0.6V,0.28V)

Figure 3.6: Chip photo of the first prototype and the basic information.

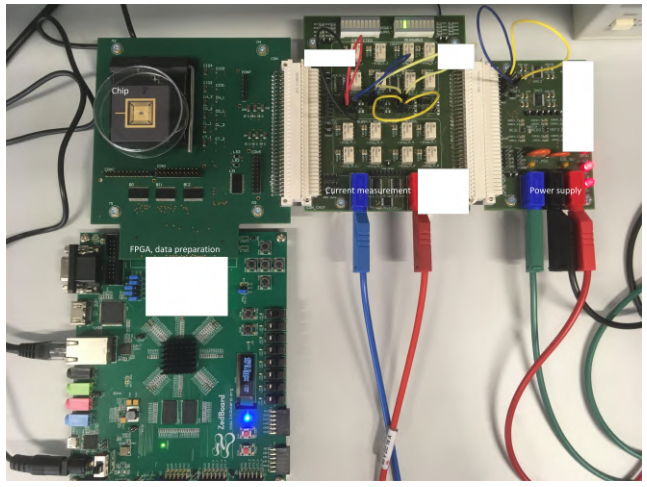


Figure 3.7: Measurement setup for the first prototype.

the bit-line is charging up, it reaches 50%VDDL threshold faster than in the normal full swing discharge case. Simulations, shown in Fig.3.5, show that the proposed scheme achieves 50% read time reduction and 50% energy reduction.

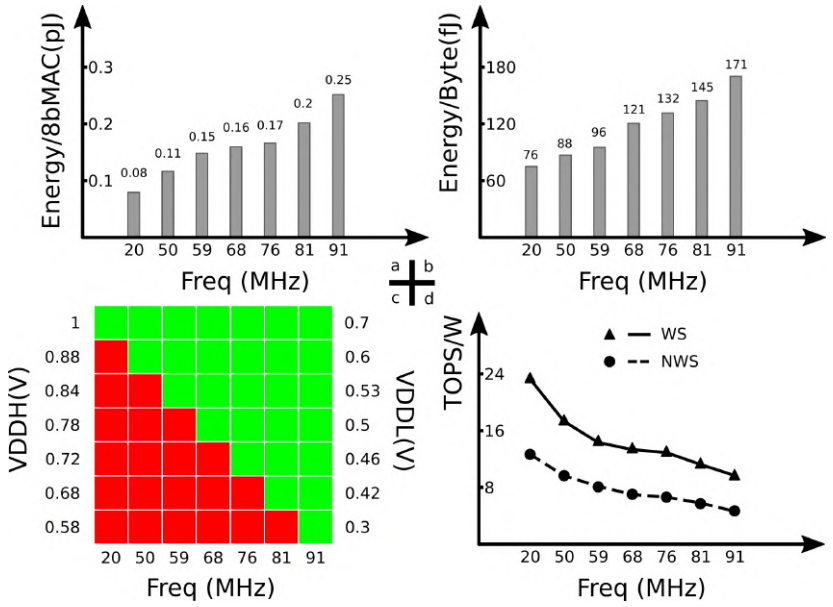
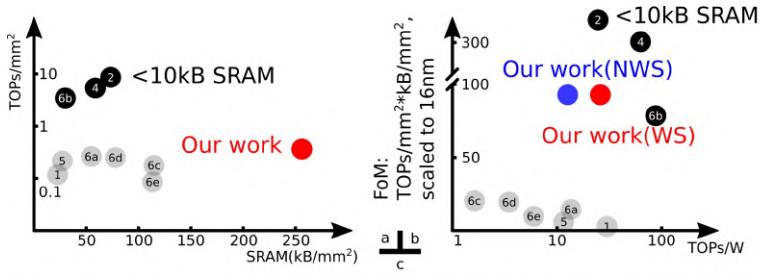


Figure 3.8: Measurement results. Byte energy denotes energy drawn from VDDH, MAC energy denotes energy drawn from VDDL.

Measurement results

The macro was produced in a 16 nm finFET CMOS technology. The chip photo is shown in Fig.3.6, and the measurement setup is shown in Fig.3.7. The measured shmoo plot of the DIMC chip is shown in Fig.3.8. The design is functional down to 20MHz at 0.58V VDDH and 91MHz at 1V VDDH. The logic shmoo plot is derived under the condition of the lowest VDDH when the macro is still functional. The logic can work at 20MHz with 0.3V VDDL, and at 91MHz with 0.7V VDDL.

The energy consumption of the macro is measured separately for VDDL and VDDH. The VDDL (logic) energy for each 8b MAC is shown in Fig.3.8.a. The macro achieves 0.08pJ/MAC at 20MHz, and 0.25pJ/MAC at 91MHz. The VDDH (SRAM) energy for each byte access is shown in Fig. 3.8.b. Thanks to the reverse pre-charge scheme, the chip achieves 76fJ/byte access at 20MHz, and 171fJ/ byte access at 91MHz. The total energy efficiency of the macro is shown in Fig.3.8.d. For weight stationary scheduling (WS) and non-weight stationary scheduling (NWS), the efficiency is different due to the different levels of weight re-use. In the WS case, we take 20x weight re-use for one read. The



	Our work	ISSCC'21 [1]	ISSCC'22 [3]	ISSCC'22 [5]	ISSCC'22 [2]	ISSCC'21 [4]
	SRAM Size > 10kB				SRAM Size < 10kB	
Technology	16nm	16nm	28nm	4nm	22nm	5nm
Type	DIMC	AIMC	AIMC	Digital	DIMC	DIMC
SRAM size	128kB	560kB	128kB	2048kB	8kB	8kB
Core area(mm^2)	0.5	25	NA	4.74	0.202	0.013
TP(TOPs)[1]	0.182	2.95	1.24	19.7	0.917	NA
TP density ($TOPs/mm^2$)[1,3]	0.364	0.118	NA	0.215	8.56	5.37
SRAM density (kB/mm^2)[3]	256	22.4	NA	27	74.8	58.7
FoM[1,3]	93.18	2.64	NA	5.78	640	315
Cell type	6T (logic)	8T1C	6T (foundry)	NA	6T (logic)	12T
Precision(I/W/O)	8/8/32	8/8/8	8/8/22	8/8/NA	8/4/16	4/4/14
Efficiency (TOPs/W)	23.8(WS)[2]	30.25	27.75	11.59	24.7	63

1, An 8bW*8bA MAC is defined as two ops. 2, WS: with weight stationary. 3, Density numbers are scaled to 16nm.

Figure 3.9: State-of-the-art comparison. All area density related numbers are scaled to 16nm. Points labeled with [1-5] are [49], [16], [98], [25], [67]. Points labelled with 6a-e are CNN processors reported in [29]. The macros < 10kB are in the table for completeness but are too small for any realistic AI workload.

number 20 is calculated from the data retention time on LBL 1 microsecond divided by clock period time 0.05us (20MHz). In the WS case, the chip achieves the maximum 23.8TOPs/W (8b) at 20MHz. In the NWS case, every MAC operation needs a weight read. In that case, the chip achieves 12.6TOPs/W at 20MHz. Fig.3.9.a shows the SotA area efficiency of recent AIMC, DIMC and digital accelerators. This design achieves the best SRAM density among all >10kB designs, while staying competitive with logic density. Designs which are smaller than 10 kB do not a represent memory size that is relevant for realistic applications. Fig. 3.9.b shows the overall FoM and energy efficiency diagram. More detailed comparisons are shown in Fig.3.9.c.

The area efficiency (AE) of this prototype, as defined in chapter I, section 1.6.2, can be calculated as following. First, the macro contains 128KB bit cell, which means 128K units. Second, the macro contains $16 \times 8 \times 16 = 2048$ 8bx4b MAC, which means $2048 \times 4 \times 8 = 65.5K$ units. Thus, $AE = (128K + 65.5K)/0.5mm^2 = 387KUnits/mm^2$.

3.2 Second prototype - improved macro

The first prototype successfully demonstrates an optimized DIMC design with the optimal array dimension, utilizing the AAM adder tree strategy and a bit-parallel/bit-half-parallel computation scheme. Optimization is achieved by leveraging both BL and BLB ports, where each port computes a product between a 1-bit input activation and an 8-bit weight. Consequently, the adder tree on each side processes only two partial products, mitigating placement challenges.

However, this macro faces four major issues:

- The bit cells are logic DRC-compliant cells, leading to a significant drop in area efficiency.
- The long shared write/pre-charge peripheral limits the macro speed to below 100 MHz.
- The reverse pre-charging scheme restricts the number of allowable WLs per macro, further reducing area efficiency.
- The empty MAC area issue, as discussed in Fig. 2.28, remains unresolved.

Due to these limitations, this macro is needs further improvement for deployment in applications requiring more realistic, larger model sizes. In this section, we

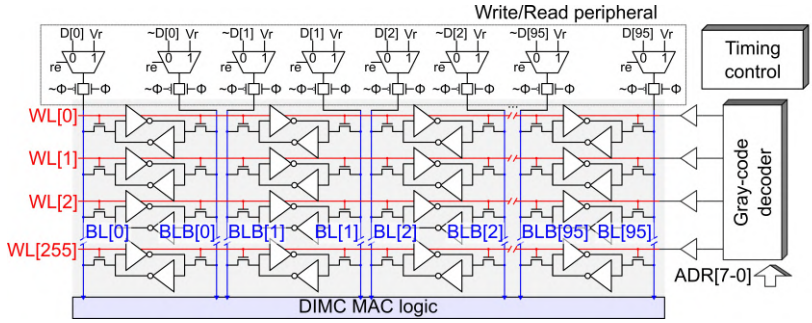


Figure 3.10: Ciruit details of DIMC macro memory.

introduce an improved macro design that addresses these four challenges. *The text of this section is based on the JSSC publication.*

3.2.1 Macro architecture

As Fig.3.10 shows, the improved DIMC macro utilizes a 6T SRAM bit-cell array, consisting of 256x96 pushed-rule 6T bit-cells. This configuration allows the memory to store up to 256x24 4-bit weights. The macro features a standard SRAM access pattern, delivering 24 4-bit weights per cycle in parallel. It then computes the dot product between the 24-element, 4-bit weight vector and the input vector. Address decoding is managed by a gray-code style decoder [61] that converts an address into a specific word line. By leveraging the minimal transition property of gray-code, the area of the decoder is optimized. The timing control circuit design is simplified by exploring synchronous operations.

Fig. 3.11 illustrates the DIMC MAC architecture for this design, which employs an MAA-style adder tree. The W data from the DIMC cell array are divided into four groups, each responsible for bit-level accumulation of four corresponding bits. Since the input vector has a length of two, 2-bit multipliers are required for the multiplication. In total, each group consists of 24 2-bit multipliers, generating 24 2-bit partial products. These partial products are then accumulated using four adder stages, producing a 7-bit output per group. The outputs are stored in flip-flops, and their voltage levels are restored using level shifters. The key innovations of this macro will be discussed in detail in the following sections.

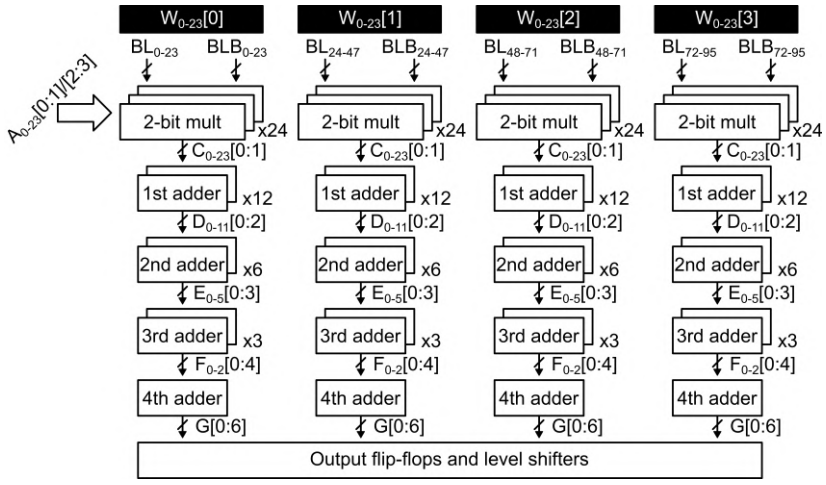


Figure 3.11: The details of the DIMC MAC.

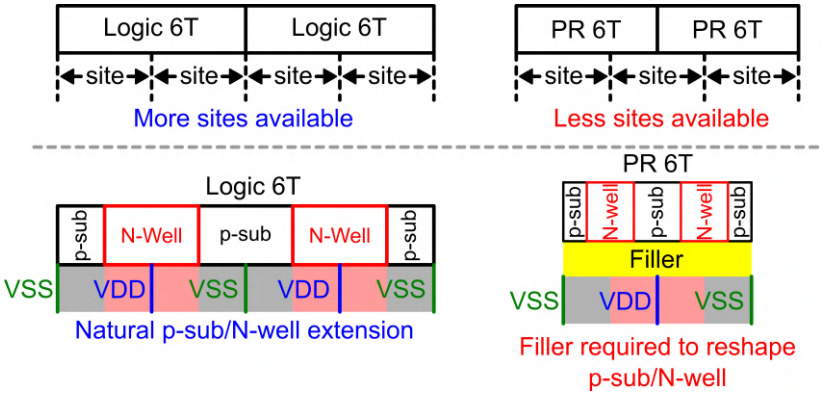


Figure 3.12: The difference between the logic rule 6T and pushed rule foundry 6T.

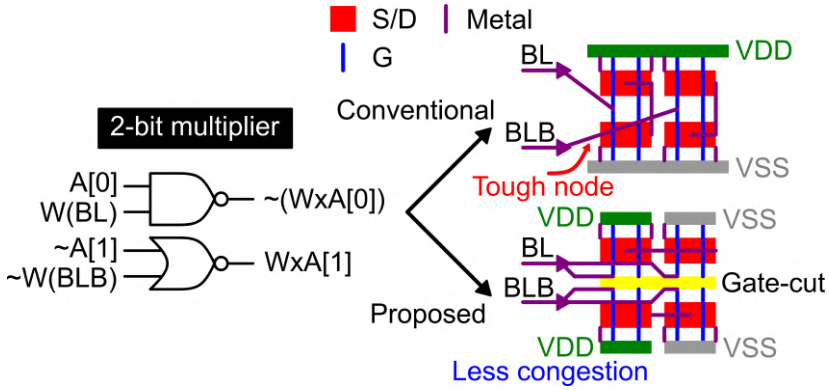


Figure 3.13: The multiplier layout optimization.

Pushed-rule 6T bit cell

In this design, the 6T bit cells are replaced with pushed-rule foundry 6T bit cells. The primary differences between the two implementations are illustrated in Fig. 3.12. First, the pushed-rule 6T DIMC provides fewer available sites compared to the logic-rule 6T DIMC. Second, in the logic-rule 6T design, the N-well and p-substrate regions of the cell array can be naturally extended to accommodate the DIMC MAC logic. However, in the pushed-rule cell array, such an extension is no longer feasible. As a result, additional filler regions are required to reshape the width of the p-substrate and N-well, leading to extra area overhead. To address these area overhead challenges, this design introduces several optimization techniques, which will be discussed in the following sections.

The multiplier

The improved macro features a 2-bit multiplier MAA DIMC MAC circuit, necessitating the integration of a 2-bit multiplier. This is achieved by incorporating a NAND/NOR pair for each BL/BLB pair. Since the multiplier receives inputs directly from the BL, it must also conform to the narrow pitch constraints of the BL/BLB.

As illustrated in Fig. 3.13, a standard NAND/NOR layout presents routing challenges when connecting BLB to the gate of the NOR, as it must pass through the NAND gate. To overcome this issue, this design introduces a MUX-like layout technique. Unlike conventional logic gates, which maintain the VDD metal track at the top and the VSS metal track at the bottom, the proposed

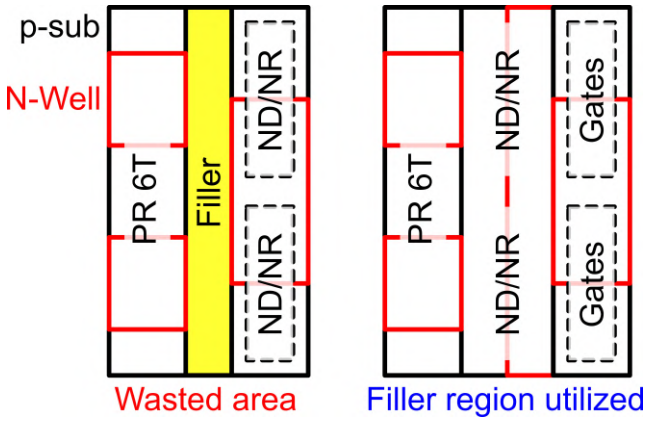


Figure 3.14: The area saving from vertical p-sub and N-well strip.

multiplier layout adopts a horizontally split NMOS and PMOS structure. In this configuration, the VSS pins are positioned on the left, while the VDD pins are located on the right.

This approach results in a significant structural change: instead of the traditional horizontal p-sub and N-well strips, the design employs a vertical p-sub strip on the left and a vertical N-well strip on the right. This property allows the filler area to be effectively utilized for multiplication, as demonstrated in Fig. 3.14.

Additionally, by arranging the NMOS and PMOS transistors horizontally for the same gate, BL/BLB routing to the gates is significantly simplified. The gates of the top and bottom transistors are separated using a gate-cut, resembling the layout of a transmission gate in a MUX design. By implementing this technique, the area overhead from the BL/BLB routing is reduced, improving the overall DIMC area efficiency.

The adders

The adders in the local adder tree are fully customized. In general, integrating a complete full-adder or half-adder within a DIMC MAC using pushed-rule 6T cells is highly challenging. This difficulty arises because the narrow width of these cells compresses the maximum number of available metal tracks in the gate region to just 4–5 tracks. Such limitations severely restrict the layout scaling of full-adders due to the constrained metal connection possibilities.

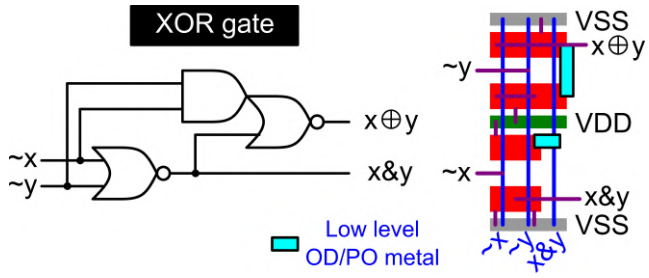


Figure 3.15: Optimized layout of the XOR gate with AND generation.

To address this issue, this design leverages alternative gate connection strategies to optimize the adder layout within the available space. Additionally, thin adder layout techniques are being explored to further mitigate placement constraints and improve integration efficiency.

After the multiplication, the 1st adder can be expressed as the following logic

$$\begin{aligned} z[0:2] &= x[0:1] + y[0:1] \\ z[0] &= x[0] \oplus y[0] \\ z[1] &= (x[1] \oplus y[1]) \oplus (x[0] \& y[0]) \\ z[2] &= (x[1] \& y[1]) || ((x[1] \oplus y[1]) \& (x[0] \& y[0])) \end{aligned} \tag{3.1}$$

Furthermore, the XOR ($\oplus$) operator can be decomposed to

$$\begin{aligned} x \oplus y &= (\overline{x \& y}) \& (x || y) \\ &= (\overline{x} || \overline{y}) \& (\overline{\overline{x} \& \overline{y}}) \end{aligned} \tag{3.2}$$

This approach enables the construction of all gates with maximum utilization of $\overline{x \wedge y}$ and other simple elements. Fig. 3.15 illustrates the optimized layout of the XOR gate, incorporating an additional AND gate. The key optimizations include:

- The NOR gate is implemented using a two-input gate, while the AOI (And-Or-Invert) is realized with a three-input gate.
- The gate poly for $\sim x$ and $\sim y$ is reused for both the NOR and AOI gates, improving layout efficiency.

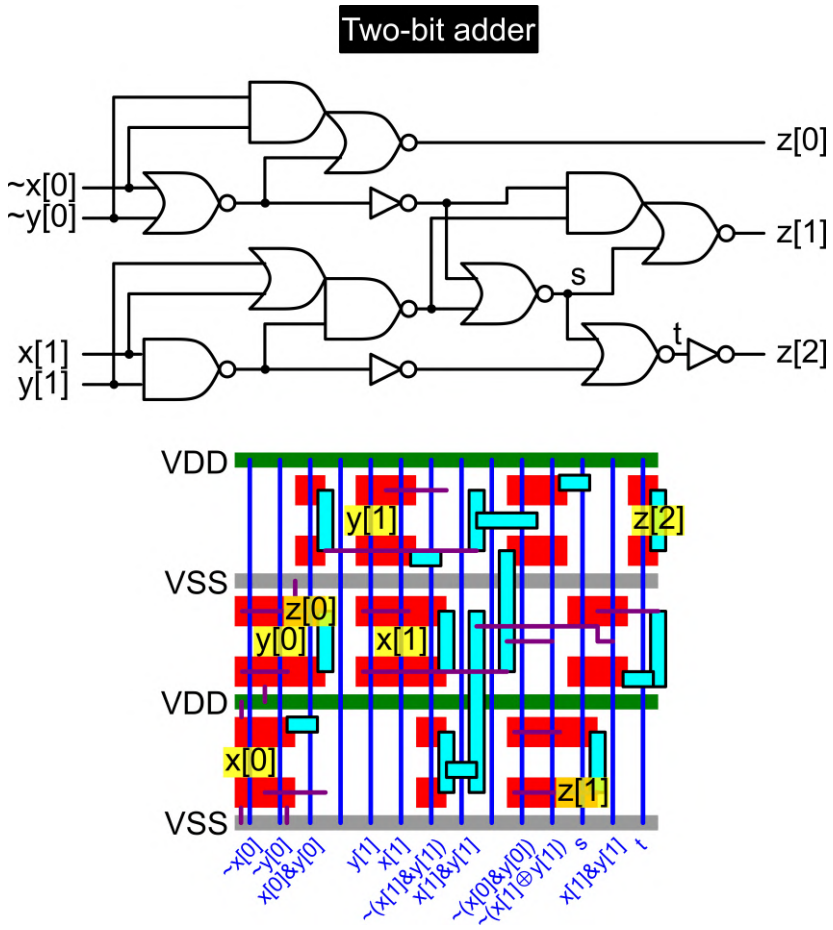


Figure 3.16: Optimized layout of the complete first 2-bit adder circuit.

- The gate-metal connections for $\sim x$ and $\sim y$ are vertically separated between the NOR and AOI gates to reduce input routing congestion.
- The $x \wedge y$ connection to the AOI is made using low-level OD/PO connection metal, which is typically restricted in place-and-route tools.

With these optimizations, the number of utilized metal layers—excluding the low-level OD/PO metal—is reduced to just one, significantly increasing routing flexibility for higher-level interconnections. Additionally, this approach results in a much more compact layout compared to a standard XOR implementation.

The same technique can be extended to the complete first addition stage, as illustrated in Fig. 3.16. In this design, all internal connections are implemented using M1 (depicted in purple) or low-level OD/PO metal, while the input and output ports (marked in yellow) can be routed using M2 or higher metal layers. The ports are evenly distributed vertically, effectively reducing congestion at higher routing levels. Compared to a standard cell implementation—which does not even include gate-to-gate connections—this optimized approach achieves a 30% reduction in area.

In general, we can decompose the carry ripple adder as the following:

$$\begin{aligned} z[0:N] &= x[0:N-1] + y[0:N-1] \\ z[0] &= x[0] \oplus y[0] \\ c_0 &= x[0] \& y[0] \\ z[i] &= x[i] \oplus y[i] \oplus c_{i-1} \\ c_i &= (x[i] \& y[i]) || ((x[i] \oplus y[i]) \& c_{i-1}) \\ z[N] &= c_{N-1} \end{aligned} \tag{3.3}$$

The most critical observation is that the $z[i]$ and c_i shares many common intermediate operators. For instance, $x[i]\&y[i]$ appears both in c_i and the $x[i] \oplus y[i]$ of $z[i]$, an the $(x[i] \oplus y[i]) \& c_{i-1}$ is also part of the $(x[i] \oplus y[i]) \oplus c_{i-1}$ according to the XOR decomposition. Exploring these properties allows us to maximally use the low level OD/PO metal or directly the PO (gate), encountering the routing congestion issue from the thin adder layout.

Fig. 3.17 provides an overview of the DIMC MAC layout. The overall height of the DIMC MAC remains consistent with the bit-cell array. As the adder stages progress, the layout becomes increasingly compact due to the efficient utilization of low-level OD/PO metal and gate poly. However, the width reduction is not applicable to the fourth adder stage due to extreme aspect ratio constraints. Despite this limitation, these optimizations result in a 30% area reduction compared to a synthesized DIMC MAC implementation.

3.2.2 Write/read peripheral

Unlike traditional SRAM, the write/read peripheral in DIMC must be minimized. This is because DIMC typically employs a very high BL to WL ratio to enhance throughput. However, this strategy significantly increases the replication of the write peripheral circuitry. Consequently, even a small reduction or increase in

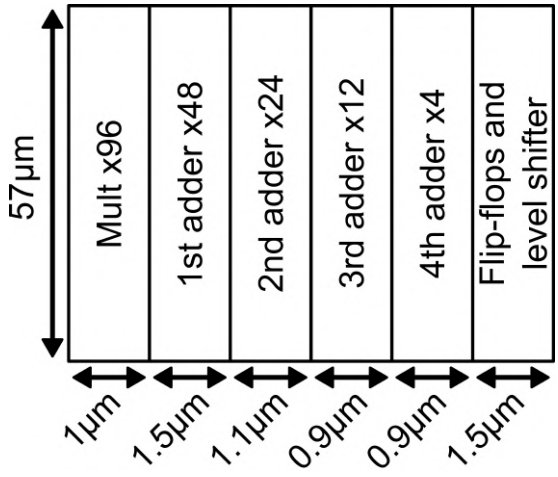


Figure 3.17: Summary on DIMC MAC dimension.

the area of the write peripheral has a substantial impact on the overall area efficiency of the macro. In this work, we explore two key differences between normal SRAM and DIMC:

- DIMC typically utilizes fewer word lines, resulting in significantly lower bit-line parasitic compared to traditional SRAM. Consequently, full-swing reading on the bit-line in DIMC is more energy efficient than in traditional SRAM.
- For models that can be fully stored within the DIMC, write operations are only required during the boot process. This significantly reduces the need for write-assist circuitry.

As illustrated in Fig.3.10, the write/read peripheral functions as a 2-way multiplexer, controlled by a read enable signal (re) that dictates the DIMC operation mode. When re is set to 0, 96 different data inputs ($D[0-95]$) are written into a selected word line. Conversely, when re is set to 1, a pre-charge voltage (Vr) is applied to the bit-line. The value of Vr is adaptable; for example, a lower Vr, such as 75% of the nominal Vdd, can improve reading speed and reduce energy consumption. However, this also decreases cell robustness and limits the maximum voltage that can be applied to the DIMC MAC logic. When Vr is set to VSS, it reduces to the reverse pre-charge scheme used in the first prototype, which would fail due to the increased number of word lines in this design. For the measurement, the Vr is set to be equal to the DIMC MAC

logic power supply. In all modes, changes occur only when the control signal Φ is active (high).

3.2.3 Gray-code style decoder

To further optimize the area efficiency of DIMC circuitry, this work proposes a gray-code style decoder to improve the decoder routing efficiency. As illustrated in Fig.3.18, the 8-bit address $A[7-0]$ is partitioned into three segments: $A[7-6]$, $A[5-3]$, and $A[2-0]$. A binary-to-thermometer conversion module generates the enable (ENi) signals to select one of four blocks, each containing 64 addresses. Within each block, the addresses are further subdivided into 8 smaller tiles, selected by decoding $A[5-3]$. Within a tile, the $A[0]$ bit functions as a special address signal that persists for half of the duty cycle. This bit is also encoded in gray code, ensuring only a single bit change between consecutive word line (WL) drivers. As illustrated in Fig.3.18, the proposed Gray-code decoder offers more opportunities for gate poly contact sharing in the NAND gates compared to the conventional binary code decoder. This reduces layout complexity. Moreover, it reduces the switching activities on the WL for linear address fetching. On average, the energy saving on the decoder thanks to the minimal bit flipping is 50%.

3.2.4 DIMC synchronous operation timing diagram

To further enhance macro-level energy efficiency, we implement synchronous operations for the DIMC, reducing the energy required for data reading within the DIMC. The timing diagram of the DIMC macro is depicted in Fig.3.19, highlighting three primary operations: write, read/compute, and refresh. During the weight write operation, weight data (D) is written to the word addressed by the address A. Since the write process inherently disrupts the memory state, the overlap between the control signal Φ in Fig.3.10 and the word line (WL) is not a concern, enabling continuous writes over consecutive clock cycles. In the read/compute operation, the macro reads a word from memory and performs the MAC calculation with the 2-bit input activation (I). In this mode, overlap between Φ and WL is not allowed, as it could lead to destructive reading. While traditional SRAM employs delay lines to address these issues, robust and variation-resilient delay lines are both area- and power-intensive, creating a significant bottleneck for DIMC arrays with numerous macros. In this work, synchronous reading/computing is proposed to avoid this overlap concern. The idea is to spread the Φ and WL pulse over two different clock cycles in a synchronous way. A pre-charge enable (pe) signal proceeds each

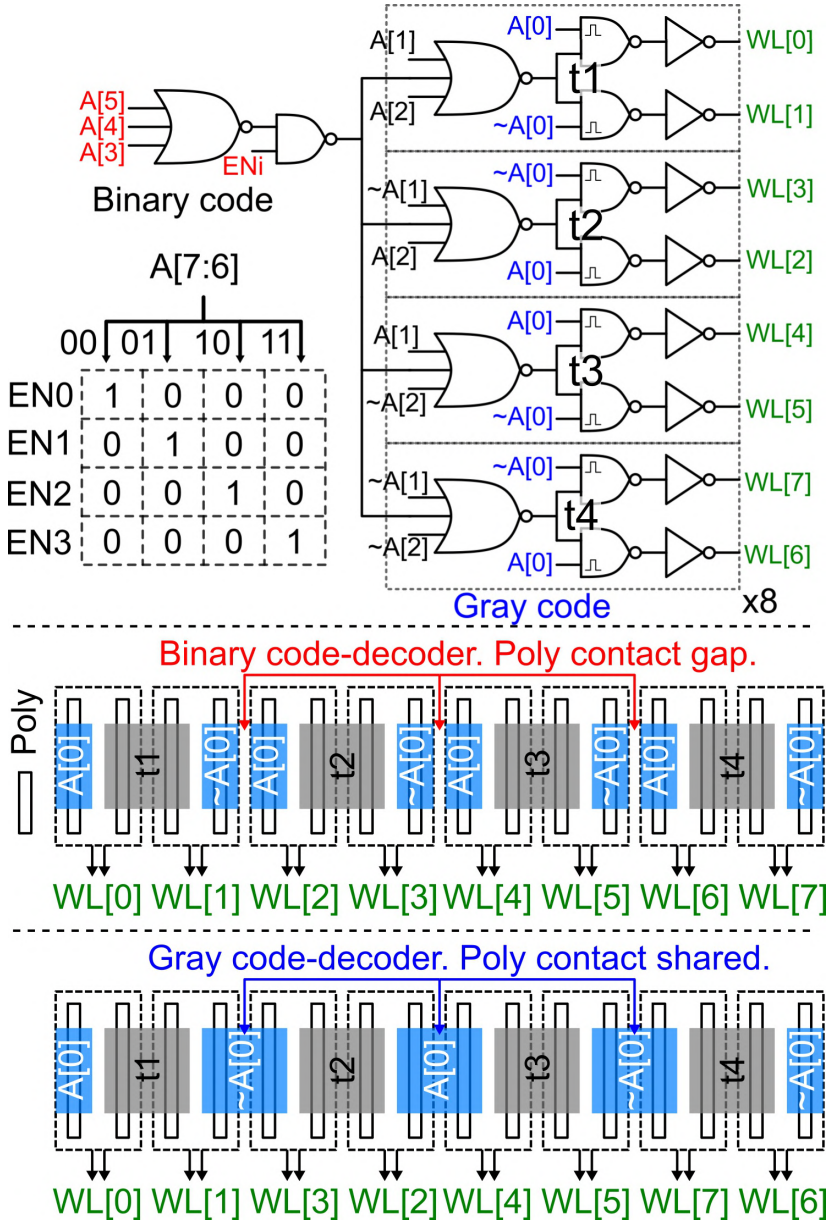


Figure 3.18: Proposed gray-scale decoder.

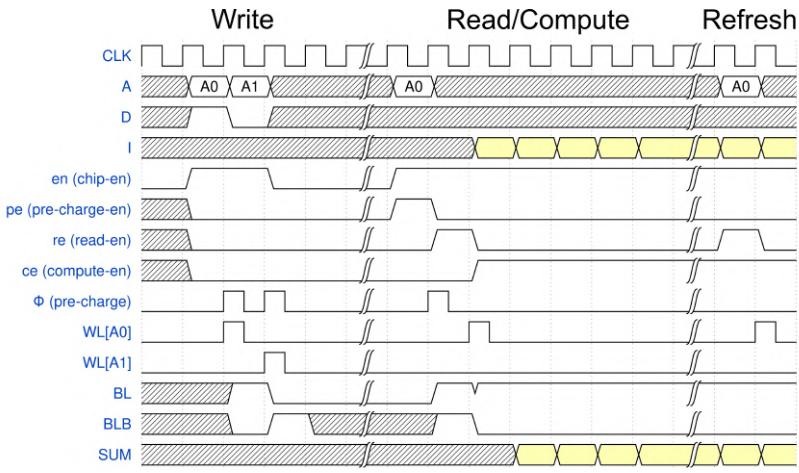


Figure 3.19: Timing diagram of the proposed DIMC macro.

read operation, generating a Φ pulse to pre-charge the BL. A read enable (re) signal follows right after, reading the selected word onto the BL. After a read operation, numerous compute operations can follow. Each SUM is the MAC result of the BL weight data and a different input (I).

Since BLs are floating nodes during the compute operation, a refresh pulse needs to be inserted on the WL after a while to recover the values on the BL to full swing. This mechanism does not affect throughput, as the BL values remain valid and *ce* stays high during refresh. It also has minimal impact on energy consumption, since the BL values are already close to the read values, resulting in only a small bit-line voltage change. However, this mechanism begins to degrade when there is reduced temporal reuse of weights. The impact of this will be examined in the measurement section by comparing among different workloads.

The adoption of synchronous operation significantly reduces the critical path delay during the read operation. Additionally, since the DIMC MAC logic is no longer part of the critical path, its power supply can be lowered to enhance energy efficiency. For subsequent measurements, both the DIMC MAC logic power supply and the pre-charge voltage (Vr) are set to the minimum voltage level at which correct computation results are still reliably obtained.

3.2.5 The area efficiency

The area efficiency of this macro can be calculated as follows. First, there are $96 \times 256 = 3KB$ bit cells per macro, which means 3K units. Second, a macro consists of 24 4bx2b MAC, which means $24 \times 4 \times 2 = 0.192K$ units. Third, the chip area of the macro is $65\mu m \times 67\mu m = 0.0044mm^2$. As a result, the $AE = (3K + 0.192K)/0.0044mm^2 = 725KUnits/mm^2$. Apparently, by adopting the foundry 6T bit cell, the second prototype achieves a much higher area efficiency compared to the first prototype.

3.3 Conclusion

In this chapter, we propose two DIMC macro prototypes in 16nm. Both prototypes are built upon a 6T bit cell array, and circuit techniques are applied to increase both the area efficiency and area efficiency.

For the first prototype, there are three takeaway conclusions:

- The energy efficiency can be significantly improved with BL/BLB as temporal storage element.
- The reverse pre-charging scheme lowers the DIMC read energy, but limits the allowable number of WL per macro, preventing the macro from scaling up.
- The logic 6T bit cell seriously limit the overall area efficiency of the macro.

Built on these lessons, the second prototype provides us three more takeaway conclusions:

- By making DIMC pre-charging, reading, computing, into different synchronous operations, the DIMC energy efficiency can be further improved. The impact on performance is very limited.
- By raising the pre-charging voltage to an intermediate voltage between the ground (VSS) and the supply (VDD), we can both save the DIMC reading energy and scale up the allowable number of WL per macro.
- The foundry 6T bit cell significantly improves the energy efficiency. Nevertheless, it also brings much more challenges to the customization of the DIMC MAC circuit. Two practices helps with customization: the MAA adder tree strategy and customizing only the early stage adder tree.

In the next chapter, the second prototype will be used in a system on chip to demonstrate the DIMC's energy saving potential for edge CNN applications.

Chapter 4

HUNBN, a DIMC based SoC for edge CNN applications

In this chapter, we introduce HUNBN, a DIMC-based System-on-Chip (SoC) designed for edge CNN applications. HUNBN integrates the second DIMC macro introduced in Chapter 3. We begin by presenting the system architecture, followed by a solution to the integration challenges using a novel level shifter circuit. Next, we discuss system-level optimizations for the SoC. Finally, we provide the measurement data for both the DIMC macro and the complete SoC.

4.1 System overview

The HUNBN SoC, depicted in Fig.4.1, comprises a general-purpose RISC-V core, a DIMC cluster, and a multi-bank input/output SRAM data memory. The input/output SRAM is divided into three sections, labeled P1, P2, and P3, each containing 3 banks of 65KB SRAM. This configuration is specifically optimized for common convolutional filters, as will be discussed later. The input/output SRAM can be accessed either via the AXI bus or directly from the DIMC array. The AXI bus utilizes a lightweight read/write interface designed for low-speed data transfers from the RISC-V core or off-chip sources. In contrast, the DIMC array employs three high-bandwidth stream buses to handle input activation, partial sums, and output activation efficiently, facilitating direct data transfer to and from the SRAM. The DIMC weight data are directly updated via the AXI bus. While the speed and bandwidth of DIMC weight writing are limited,

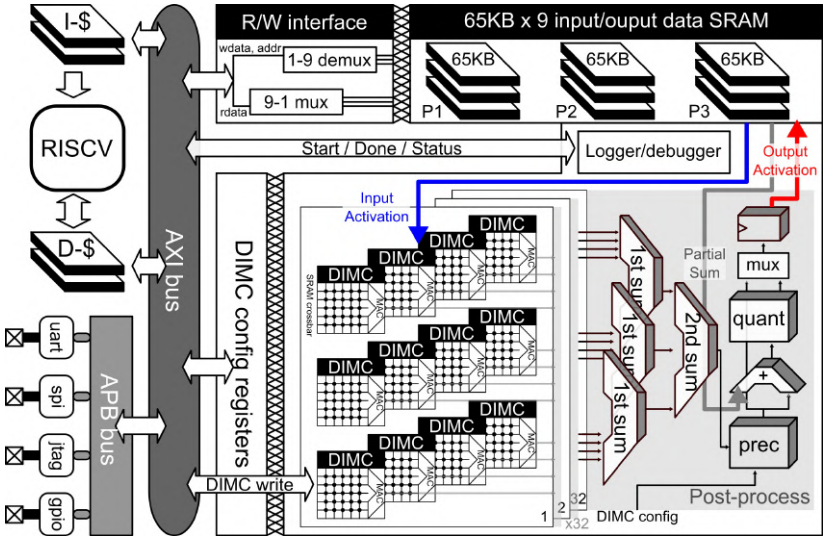


Figure 4.1: Architecture of HUNBN SoC.

this process is assumed to be a one-time cost, which does not affect performance during runtime.

The DIMC cluster serves as a convolutional layer accelerator with user-defined parameters. The input feature map size, the number of input/output channels, the starting address for the input/output data in SRAM, the weights in DIMC, and the type of convolution kernel (standard or transposed) can all be configured by the RISC-V host core. When properly configured, a CNN layer can be executed on a configurable subset of the 384 customized DIMC macros. These macros are organized into 32 groups, each consisting of 12 DIMC macros. All 32 groups share the same input activation vector, while each group is equipped with an independent post-processing module to handle the results produced by the DIMC macros.

Within each group, every DIMC macro features an SRAM bit cell array for weight storage and a local MAC unit to perform bit-level multiply-accumulate operations. In the post-processing module, the outputs from all macros within a group are further accumulated using two summation stages. The first-stage summation module processes the bit-level MAC results from each DIMC macro by shifting and adding them to form a product-level summation of 4-bit weights and activations. These product-level summations are then accumulated in packs of four. Within each DIMC group, the second-stage summation module then combines the three sums produced by the first stage. The output of the second

stage is an 18-bit full-precision value, representing the summation of 288 (24 from DIMC and 12 from post-processing) 4-bit by 4-bit products. It can be either quantized into a 12-bit partial sum, accumulated with a previous 12-bit partial sum and quantized into a 4-bit output, or directly quantized into a 4-bit output. The partial sums or output activations are directly stored in the data SRAM. Additionally, a logger/debugger unit is integrated into the cluster to monitor and control the behavior of the DIMC cluster, ensuring accurate operation and facilitating debugging. Details of each module will be discussed in subsequent sections.

4.2 Level shifter - Integrating DIMC into the SoC

The text of this section is based on the ISCAS publication.

The level shifter plays a crucial role in HUNBN. As introduced in the previous chapter, the DIMC MAC operates in a lower voltage domain compared to the digital voltage domain. To ensure correct functionality and lower the static power consumption, level shifters are needed to convert the signal from one domain to another.

There are mainly 4 important design aspects for level shifters:

- Speed. Speed is a critical factor for level shifters, and it should be optimized in both rising and falling directions.
- Power. Excessive power in level shifters presents two major issues. First, high power limits the number of cross-domain signals, which is undesirable for applications needing a high on-chip data bandwidth. Secondly, in power-constrained designs, a wider voltage conversion enables aggressive power reduction, but many level shifters struggle to scale down their power effectively with the voltage, hindering system-level power optimization.
- Conversion range. The conversion range refers to the maximum voltage difference between two power domains that a level shifter can handle. A wider range enables more versatile applications.
- Delay scaling. The delay scaling ability refers to the reduction of the delay when the voltage difference decreases. A level shifter with poor delay scaling introduces excessive additional delays during normal operation.

This thesis presents a novel Charge-Injection-Positive-Latch (CIPL) level shifter, addressing all of the above key aspects of level shifter design while maintaining a simple, pure feed-forward circuit.

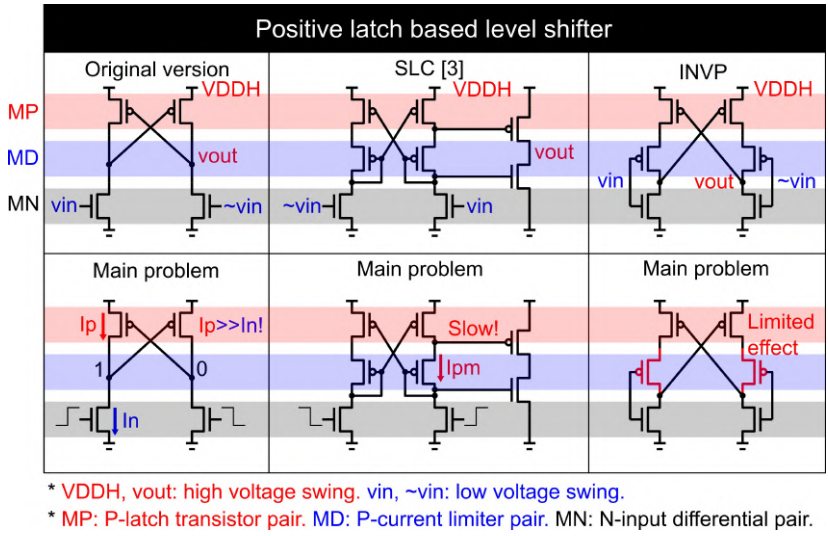


Figure 4.2: Popular positive-latch-based level shifter.

4.2.1 Existing level shifter review

In high-speed digital circuits, two primary types of level shifters are used: current-mirror-based and positive-latch-based level shifters. While current-mirror-based designs offer rapid zero-to-one transitions, they suffer from high leakage during one-to-zero switching. This inefficiency limits their widespread adoption. Modifications for current-mirror-based level shifters often focus on incorporating feedback systems for leakage cancellation after successful conversion [24, 48, 42]. While this approach helps to reduce the static power consumption, it introduces a trade-off by slowing down the speed of conversion. In contrast, positive-latch-based level shifters are more commonly used due to their balanced performance and reduced leakage, making them a better choice for most applications requiring efficient and reliable level shifting.

The original positive-latch-based level shifter, shown in Fig.4.2, includes a PMOS half-latch and an NMOS differential pair. It converts a differential voltage to a current, which the half-latch amplifies to full swing. However, during transitions with low input swing, especially below the threshold voltage, the PMOS operates under a large bias voltage, while the NMOS operates near sub-threshold. This creates a large imbalance between the PMOS current (Ip) and the NMOS current (In), preventing a proper state transition. To compensate for this, a highly aggressive NMOS-to-PMOS size ratio is required to balance the currents

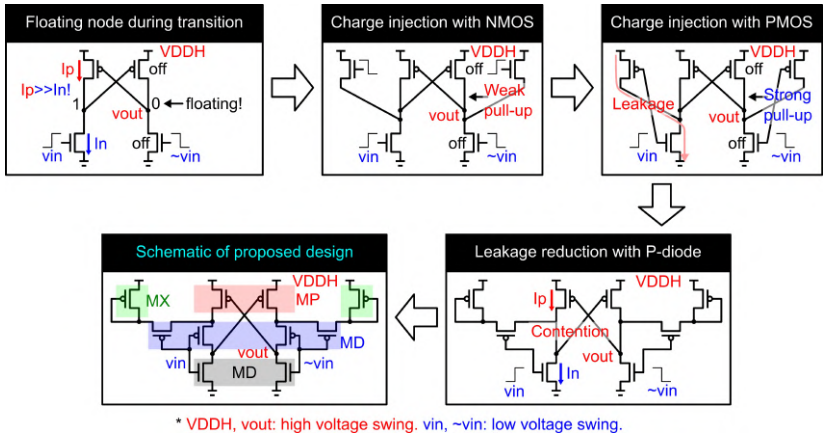


Figure 4.3: Schematic and design procedure of the proposed level shifter.

and ensure switching.

To address the issues in the standard design, two popular modifications exist. The first, the Split-Level Converter (SLC) [52], limits the PMOS current by using a diode-connected PMOS (MD) to lower the Vds of MP, reducing the Ip-to-In ratio during transitions. However, the SLC suffers from reduced voltage headroom, impacting the delay when VDDH-VDDL decreases. Another approach, the Inverter Positive Latch (INVP), improves the delay scalability by avoiding the Vtp drop. However, when the voltage difference is large, the MD transistor in the INVP experiences a high Vgs, resulting in weak current limiting, which leads to a reduced maximum conversion range.

4.2.2 Proposed design

The schematic of the proposed level shifter is shown in Fig.4.3. It consists of a standard INVP positive latch in the center, and 4 additional PMOS in the feed-forward path to help conversion. The operation of the circuit is described below.

Floating node in positive latch

In the standard positive-latch-based level shifter(Fig. 1), a state transition occurs when the NMOS current (In) overpowers the PMOS current (Ip) in one branch, while the other branch remains unused. Due to the lower Vgs of the

NMOS pair compared to the PMOS pair, Ip is significantly higher than In. As a result, the NMOS transistors must be oversized to achieve state flipping over a large conversion range. Throughout this process, the unused branch remains floating, contributing nothing to the transition. By injecting charge on this floating node, Ip can be significantly reduced before the state flipping, accelerating the transition without requiring oversized NMOS transistors. This disturbance can be achieved through charge injection at the floating node.

Charge injection with a pass transistor

A low-swing signal struggles to fully switch off the PMOS, but it can easily deactivate the NMOS. This led to the initial attempt at charge injection using NMOS pass transistors, which does not introduce any additional leakage in the idle state. However, when the input voltage approaches the sub-threshold region of the NMOS, the charge injection weakens significantly, limiting the ability to further improve the speed and conversion range. This issue presents a barrier to fully optimizing the performance under low-voltage conditions. In contrast, using a PMOS pass transistor for the charge injection significantly expands the conversion range, allowing a better performance across wider voltage domains. However, this approach introduces excessive leakage.

To reduce the leakage, a diode-connected PMOS (MX) is introduced to lower the effective supply of the PMOS pass transistor by Vtp. However, two issues remain: (1) while the leakage is partially reduced, it still poses a challenge, particularly for large conversion ranges, and (2) the Ip/In contention persists. To fully address these challenges, the PMOS half-latch is replaced with a standard INVP, and charge is injected onto the internal node instead of Vout. This approach offers two key advantages: (1) the stacking of the two PMOS transistors in MD further limits the leakage current, and (2) the Ip contention is reduced via the MD current limiter.

Waveform of the proposed level shifter

Example waveforms for the 300mV to 800mV conversion are shown in Fig.4.4. The conversion process can be divided into two stages. In stage 1, charge is injected into the floating node Q, significantly reducing the Ip on the other side. In stage 2, the P-latch resumes normal operation, but with much faster switching compared to the original P-latch, due to the reduced Ip-to-In ratio. It is also important to note that, while the 0-to-1 conversion requires both phases to complete, the 1-to-0 conversion only needs a single phase, as the charge injection alone is sufficient to trigger the Vout change.

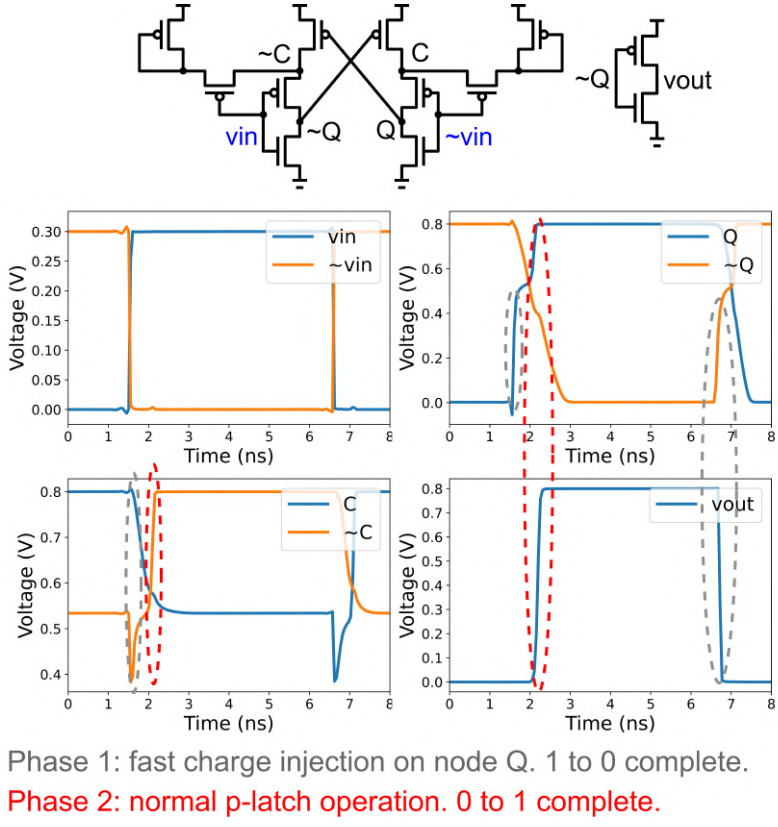


Figure 4.4: Example waveform for the proposed level shifter.

4.2.3 Simulation and comparison

The proposed design is compared with the work in [52, 36] using a 16nm FinFET technology. All level shifters are triggered by a 200MHz low-swing clock. The key performance metrics—average power over four clock cycles, 0-to-1 transition delay, 1-to-0 transition delay, are evaluated for all three designs. To assess the PVT robustness, simulations are performed across the TT, FF, and SS corners, as well as at temperatures of 27°C, 100°C, and -40°C. For a fair comparison, all designs are scaled to achieve a total fin count of 28/29, and all transistors are SVT transistors, with minimal length. For example, in our design, the NMOS pair is implemented with a 4-fin configuration, while the NMOS pair in the SLC design [52] uses 8 fins. Consequently, the increase in transistor count does not significantly impact the overall area. The conditions not shown on Fig. 4.5

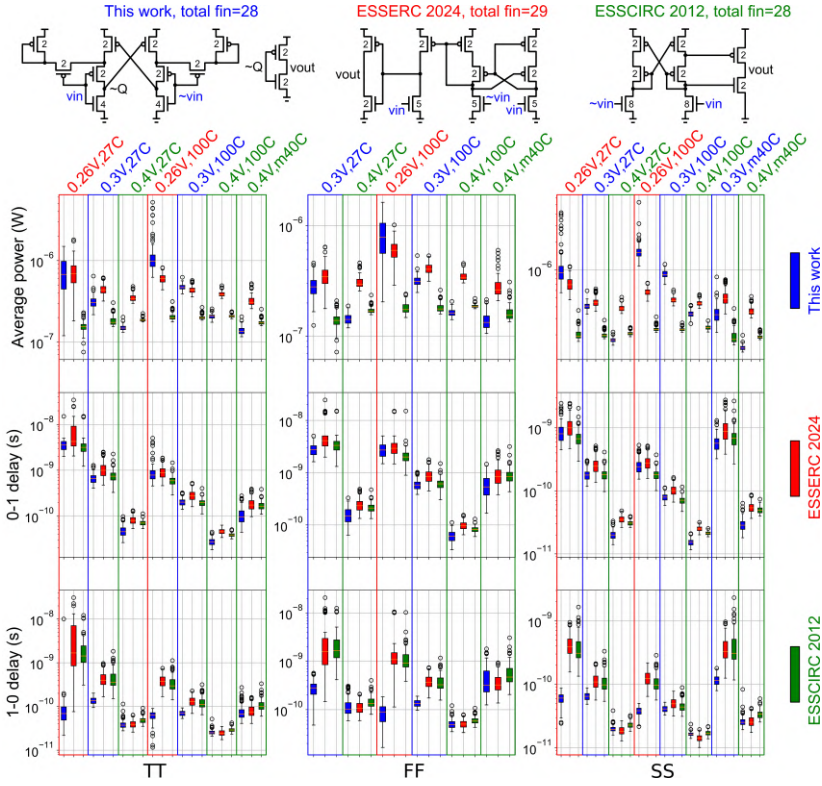


Figure 4.5: Monte-Carlo simulation results for the proposed design and [52, 36].

fail for all three designs.

As shown in Fig.4.5, the proposed design demonstrates the fastest speed under most conditions. The proposed design achieves a comparable speed to [52, 36] for the 0-to-1 conversion. More importantly, it demonstrates a significant speedup in the 1-to-0 conversion as the output voltage becomes valid within just one phase. While the power consumption of the proposed design is slightly higher than [52] for larger voltage conversions, it is lower than [52] when the voltage difference is reduced. In all cases, the proposed design maintains a controlled delay variation, especially when compared to the design in [36]. A comparison table is shown in Table.4.1.

Fig.4.6 presents another perspective of the comparison, plotting the conversion voltage against the conversion speed. The proposed design achieves a state-of-the-art conversion range, with significantly faster 1-to-0 conversion speeds.

Table 4.1: Comparison to other level shifters[1]

Reference	INVP	[52]	[36]	***This work***
Transistor count	8	8	10	10
0-to-1 delay (ns)[2]	Fail	0.73	0.94	0.64
1-to-0 delay (ns)[2]	Fail	0.4	0.42	0.13
Average power (nW)[2]	Fail	136	326	238
Minimal voltage (mV)[2]	360	240	250	260

[1]All numbers are derived under TT, 27°C corner.
[2]The number is derived for 300mV-to-800mV conversion.

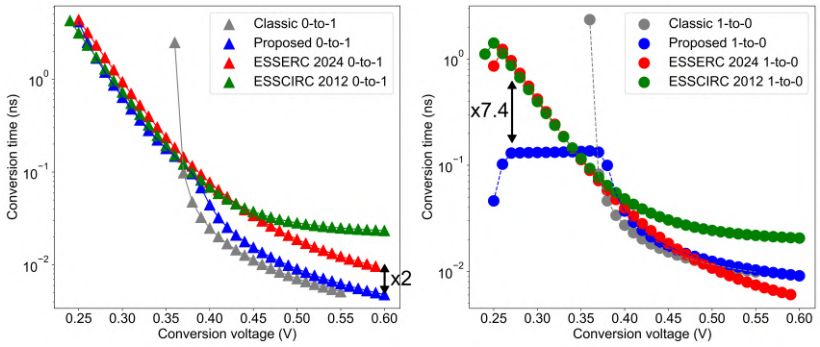


Figure 4.6: Conversion voltage to conversion time for different designs.

Additionally, the scaling of the conversion speed is highly competitive compared to the current state-of-the-art. Fig.4.7 presents the voltage sweep for both VDDL and VDDH. The results demonstrate a strong delay scalability with VDDL and a good performance under low VDDH conditions. This level shifter design will be used in the HUNBN for DIMC macro integration.

4.3 Main innovations on the SoC level

The text of this section is based on the JSSC publication.

4.3.1 Split MAC workflow

To fully exploit the efficiency and density of DIMC while minimizing manual circuit design effort, this work introduces a split MAC workflow. Fig.4.8

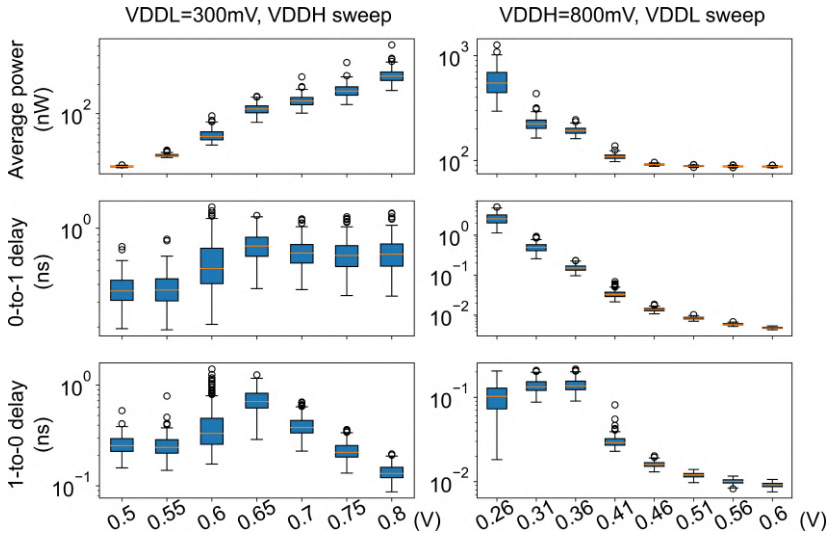


Figure 4.7: Conversion delay and average power of the proposed design.

compares the traditional DIMC MAC workflow with the proposed split MAC workflow. In the conventional DIMC approach, the manual design effort is concentrated in two areas: the multiplier, which scales with the precision of the weights and inputs, and the adder tree, whose complexity is proportional to the product of the precision and the logarithm of the number of inputs. This is because each stage of the adder tree requires a different bit-length adder, necessitating unique designs for each stage. In contrast, the proposed split MAC workflow integrates the early stages of the adder tree into the manually designed multiplication stage, while it pushes the rest of the adder tree into the synthesized logic. This approach offers two key advantages:

- The most repeated components, such as early-stage accumulation and bit-level multiplication, are fully optimized using a customized circuit design flow within DIMC. Compared to the baseline circuit implemented using a standard digital flow, a 20% area reduction is achieved. This gain results from customized logic gate height and utilizing the lowest metal layer, which digital tools cannot fully exploit. This part is treated as a hard macro in HDL.
- The more complex yet less reused components, including the partial product combiner and late-stage accumulation, are implemented at RTL level using a standard digital design flow.

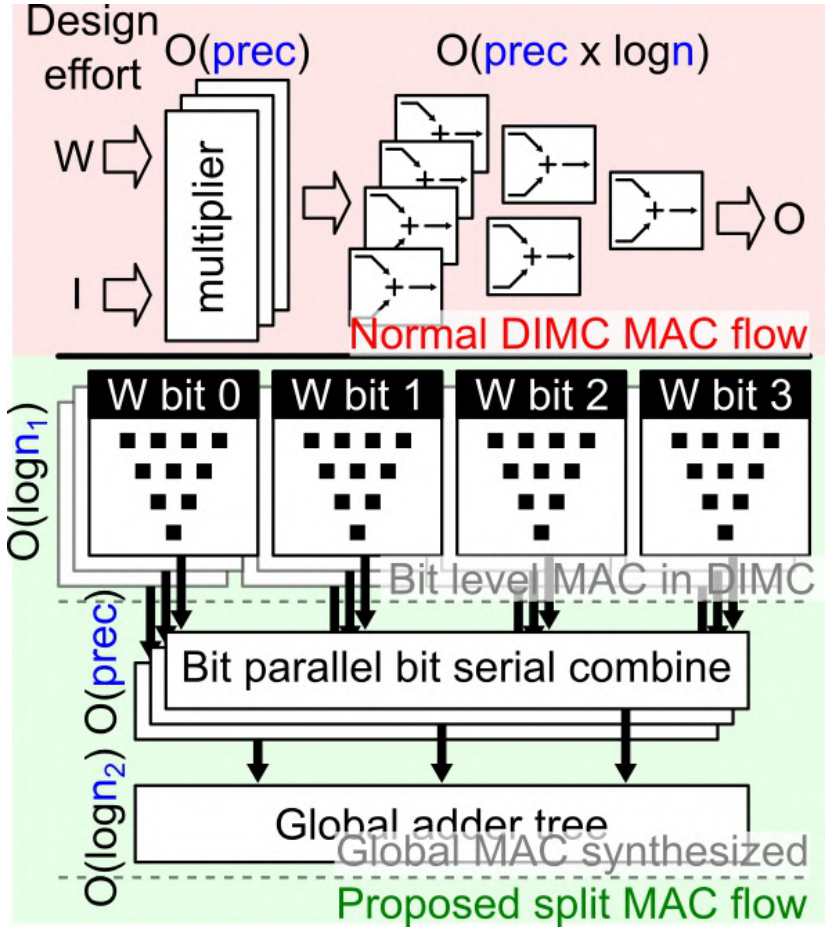


Figure 4.8: Comparison between normal DIMC MAC flow and proposed split MAC flow.

This strategy leverages the strengths of DIMC where it yields the most, while significantly reducing the overall design time.

4.3.2 CNN dataflow on SoC

CNNs are widely used AI models for visual tasks such as image recognition and segmentation. Fig.4.9 illustrates the computation flow of a CNN layer. It processes a 3-dimensional input tensor and a 4-dimensional kernel tensor,

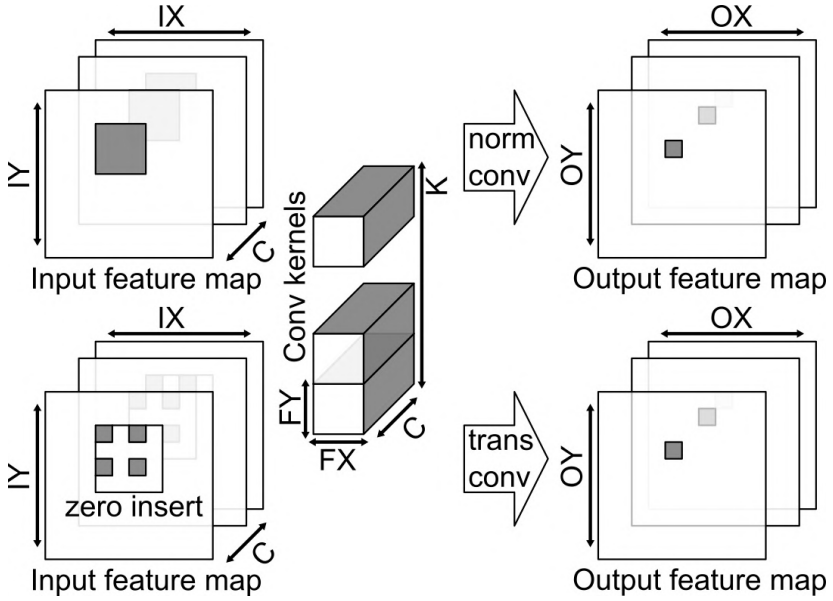


Figure 4.9: Computation of a general CNN layer.

performing matrix multiplication over each window of the input feature map to produce a 3-dimensional output tensor. Unlike traditional matrix multiplication, adjacent output pixels share many common input pixels, which allows for optimized data reuse and reduced data movement. The parameters shown in Fig.4.9 can vary depending on the type of convolution layer used. The HUNBN chip supports both normal CNN layers and transposed (up-sampling) CNN layers, with the primary difference being the input fetch pattern. In normal convolution, all pixels within the window are convolved with the kernel, while in transposed convolution, zeros are inserted to upsample the input feature map. The HUNBN chip also supports a wide range of layer dimensions (IX, IY, C, K, FX, FY) with varying temporal execution loops. For a complete set of the supported dimensions, refer to Appendix X. Four representative cases are illustrated below:

- Case 1, FX=3, FY=3, C=32, and K=32. This configuration serves as the cornerstone of all spatial unrolling. The C loop is split further into an inner C1 loop and an outer C2 loop, as illustrated in Fig.4.10.a. The 384 macros are arranged in a 3x4x32 constellation, where the FY=3, C1=4, and K=32 loops are mapped at the cluster level. The inner C2 and fx loops of the CNN are mapped within each macro. Since the macro performs

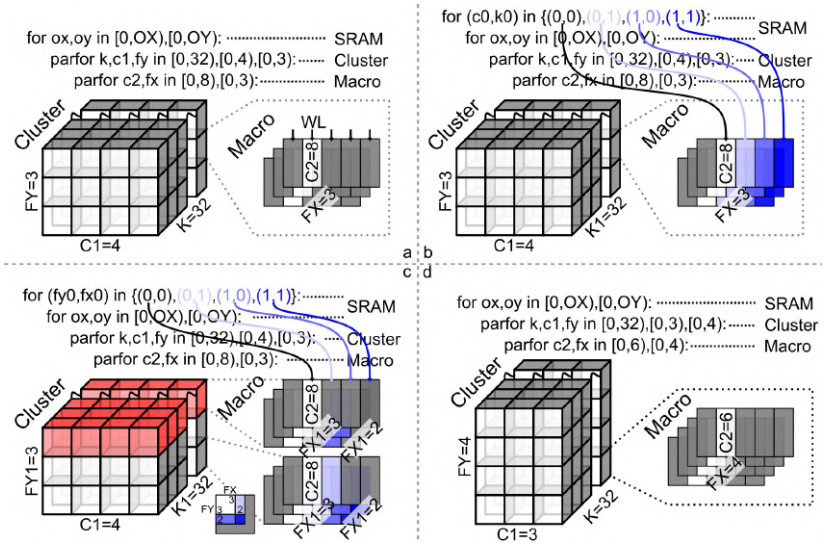


Figure 4.10: Illustration of CNN spatial mappings: Cases 1-4 are demonstrated in Figures a-d.

24 vector multiplications, up to FX=3 and C=8 can be supported within a single macro. The FY, K, and C1 loops, are computed in parallel in the cluster. The OX and OY loops, finally, are computed in the temporal domain, e.g., stored in input/ouput SRAM. This setup is optimal as it efficiently utilizes exactly one DIMC address. It also enables weight reuse at a single DIMC address throughout the entire layer computation process, maximizing the macro-level energy efficiency.

- Case 2: FX=3, FY=3, C>32, and K>32. This configuration extends Case 1. Here, the CNN layer requires more than one row of each DIMC macro, leading to storage across consecutive DIMC addresses, and an iteration across subsets of C and K in a temporal fashion. An example case with C=64 and K=64 is illustrated in Fig.4.10.b. The K0, C0 loop are outer loop indicated on Fig.4.10.b. The layer is computed using a weight-stationary dataflow, where the K0 and C0 loops are placed at the outer temporal loop, followed by OX and OY. This approach offers two key benefits: 1. Reduced Weight Reading Frequency: Weight-stationary dataflow minimizes the need for frequent weight reads, thus lowering the penalty from synchronous DIMC operations. 2. Improved Input Data Reuse: By leveraging data locality in the FX dimension through a sliding window approach, input data reuse is enhanced.

- Case 3: FX>3, FY>3, C=32, K=32. This configuration represents scenarios where the kernel size exceeds the baseline dimensions of FX=FY=3. An example with FX=FY=5 is illustrated in Fig.4.10.c. In this setup, the kernel is divided into sub-kernels by segmenting the dimension by 3 and rounding up to the nearest integer. The kernel is then split into four parts: the upper section containing a complete 3x3 and a 2x3 sub-kernel, and the lower section, which contains a 3x2 and a 2x2 sub-kernel part. These sub-kernels are stored in consecutive DIMC addresses. To prevent erroneous computations, zeros are inserted into the memory for the smaller parts. Macros within a cluster are categorized into highly utilized macros (shown in gray) and less utilized macros (shown in red). Due to the suboptimal mapping of the kernel, the performance in this case is lower compared to case 1 and case 2.
- Case 4: FX=4, FY=4, C=18, K=32. This configuration represents a specific scenario where the layer is a transposed convolution layer. The DIMC macros are organized into a 4x3x32 cluster, as shown in Fig.4.10.d. The loop parameters FY=4, C1=3, and K=32 are mapped to the cluster, while inside the macro, FX=4 and C2=6 are mapped to the weight vector at a single address. This spatial mapping fully utilizes DIMC's memory and MAC resources. However, on HUNBN, the input channel count is usually a multiple of 32, which misaligns with the optimal configuration (C=18) for the transposed convolution layer and complicates input/output data movement management. HUNBN provides special support for these irregular cases, which will be discussed in the following section.

4.3.3 Data movement optimization for input activation

To further enhance system-level energy efficiency, the input activation data movement from the SRAM to DIMC cluster is optimized towards the DIMC cluster architecture. Fig.4.11 illustrates the memory representation of an example input feature map. The 3-dimensional tensor is decomposed into four hierarchical levels to optimize memory storage and access.

At the first level, pixels across different channels are grouped into units called PixelPacks, with each pack containing pixels from 32 different channels. For layers with more than 32 channels, multiple PixelPacks are used to store the pixels at a given X/Y coordinate. Since each pixel is a 4-bit value, 32 pixels require 128 bits of storage, which aligns with the word length of the input/output SRAM. As a result, the PixelPack serves as the smallest storage unit for the feature map.

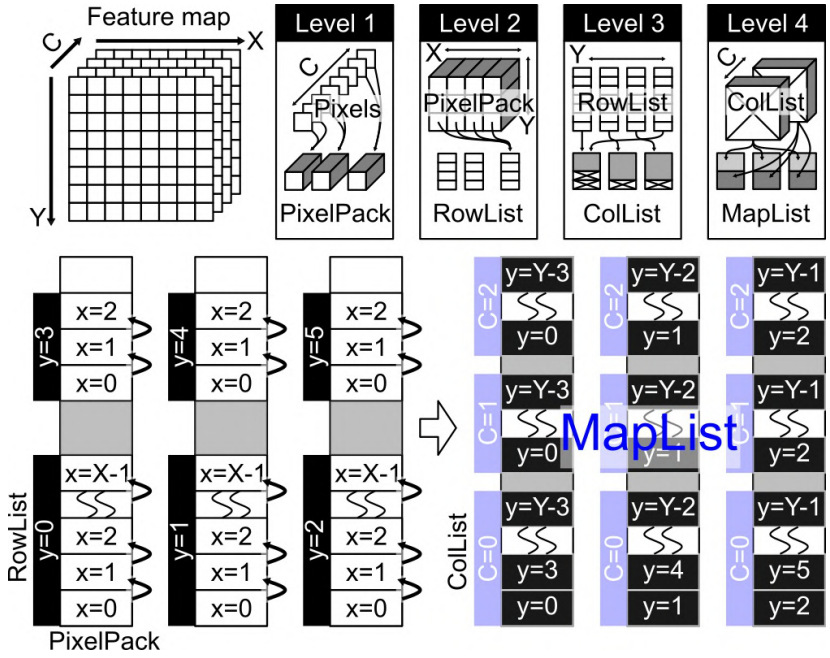


Figure 4.11: Illustration of CNN spatial mappings: Cases 1-4 are demonstrated in Figures a-d.

At the second level, PixelPacks corresponding to the same row (y-dimension) are arranged into linear structures called RowLists. A RowList contains the starting address and x-length of the feature map to locate the associated PixelPacks. For most CNN layers, this linear fetching mechanism is sufficient, significantly simplifying memory management.

At the third level, these RowLists are further organized into ColLists. The ColLists contain every pixel of the feature maps in groups of 32 channels, making them the smallest operable unit for layer-wise scheduling. For example, if a feature map contains 64 input channels, it requires two ColLists for storage. To avoid memory access conflicts, each ColList distributes the RowLists across three separate SRAM banks, where the location is determined by the RowList's modulo-3 congruence. With this configuration, three consecutive rows are distributed across different banks, ensuring single-bank access per cycle for an FX=FY=3 layer.

At the fourth level, ColLists that originate from different input channels are distributed across the SRAM. The complete set of ColLists is referred to as the

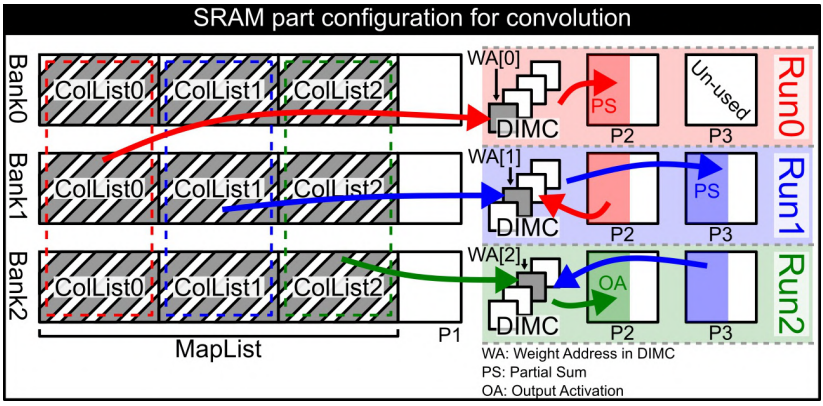


Figure 4.12: Illustration of an example SRAM part configuration.

MapList, which represents the entire feature map.

Before the convolution begins, the SRAM parts must be configured. Fig.4.12 shows an example of part assignment for a CNN layer with C=96. In this setup, P1 serves as the input activation supplier, P2 as the first partial sum buffer, and P3 as the second partial sum buffer. Given C=96 the input activation and weight tensor require 3 ColLists and 3 DIMC addresses. Consequently, 3 cluster runs are needed to compute the output activation. During these runs, P2 and P3 alternate roles as partial sum buffers. Data movement is optimized differently for normal 3x3 convolution compared to 4x4 transposed convolution.

Normal convolution

Fig.4.13 illustrates the data movement in a typical convolution operation. For simplicity, only run1 and a portion of run2 are discussed. Fig.4.13.a, 4.13.b, and 4.13.c depict the data movement process for run1.

During run1, the base address is initially set to 0, corresponding to the starting address of the first RowList in ColList1. An x-pointer is used to address the PixelPacks in RowList0, RowList1, and RowList2, with an offset based on the current x-coordinate. The PixelPacks are cached locally in a 3-depth register, which supplies the complete data set needed for one convolution. Concurrently, the same x-pointer is used to fetch the previous partial sum from P2 and store the current partial sum to P3. Upon completion of the first row, the computation proceeds to the second row. As shown in Fig.4.13.b, RowLists RL1, RL2, and RL3 are required. The ColList layout ensures that no bank

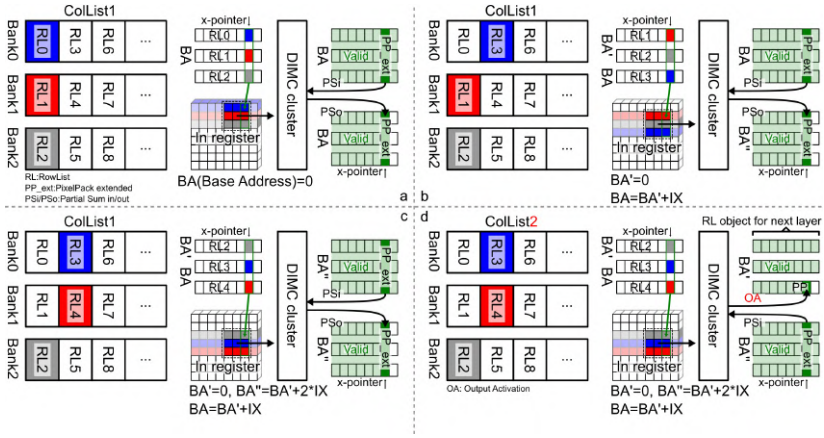


Figure 4.13: Data movement for normal convolution: Figures a-c illustrate the process of run1, while Figure d shows the differences between run1 and run2.

conflicts arise. RL1 and RL2 continue to use base address 0, while RL3's base address is set to 0+IX to correctly locate RL3 in bank0. Similarly, the partial sum is stored with an offset of 0+IX to prevent row data overwrite. This setup allows the computation of the third row to proceed without bank conflict as well, as shown in Fig.4.13.c.

A portion of run2 is illustrated in Figure 4.13.d. The input activation fetch pattern follows the same rules as in run1, with the primary differences occurring in the output. First, while run1 stores the partial sum in P3, run2 retrieves the partial sum from P3 instead of P2. Second, partial sums are stored using 12 bits, which do not fit within a standard PixelPack. To address this, three PixelPacks across three banks at the same address are combined to form an extended PixelPack for partial sum storage. In contrast, the output activation requires only 4 bits and thus fits within a single PixelPack in one bank. In this case, the x-pointer is set to access the correct bank according to the previously discussed rules, enabling seamless integration for the computation of the next layer.

Transpose convolution with FX=FY=4

Special support is provided for transpose convolution layers with FX=FY=4, introducing two key differences: 1. The input feature map is up-sampled by inserting zeros, requiring a more complex line-fetching mechanism. 2. The maximum input channels per cycle are reduced to 18, which can now span

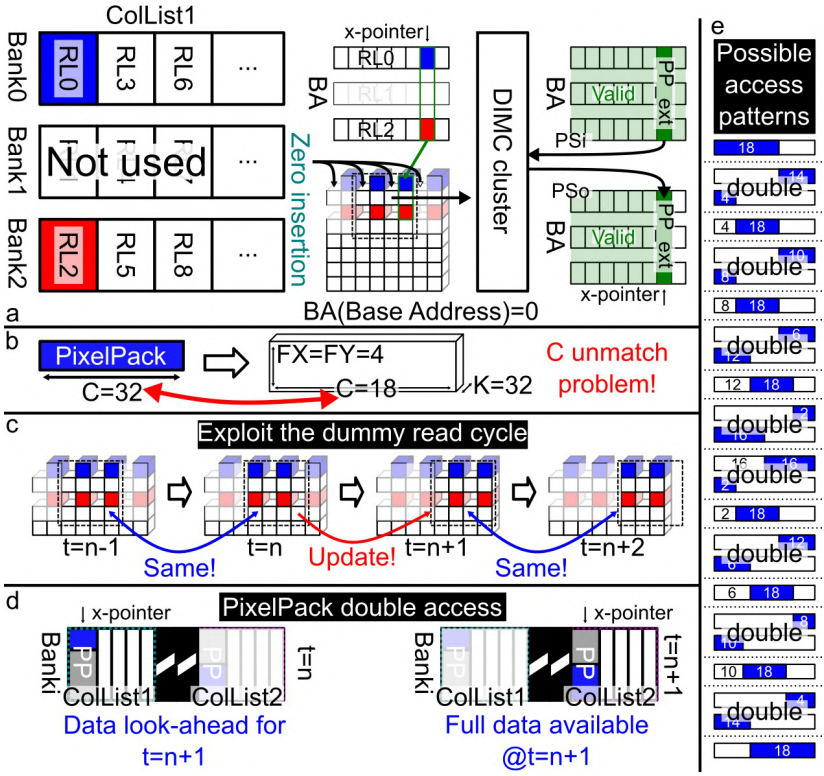


Figure 4.14: Data movement for FX=FY=4 transpose convolution.

across two PixelPacks, adding to the control overhead. Fig.4.14.a illustrates an example of data movement. In transposed convolution, zeros are inserted in one column, reducing the required input data bandwidth. However, due to the mismatch between the PixelPack C dimension and the spatial unrolling, two PixelPack accesses may be required to retrieve the complete set of input activations. Fortunately, with the insertion of zeros, the frequency of new input requests is effectively halved compared to standard convolution, as shown in Fig.4.14.c. This allows a data look-ahead mechanism to keep the DIMC operating at the same pace as in normal convolution. An example of this is shown in Fig.4.14.d: at time t=n, input data from the first portion of the input channels for the next pixel are accessed from the appropriate PixelPack in ColList1. At t=n+1, when new pixel data is required, the second portion is accessed from the correct PixelPack in ColList2. Altogether, there are 16 possible fine-grain PixelPack fetching patterns, as illustrated in Fig.4.14.e. Such mechanism enables full-speed execution of FX=FY=4 transpose convolution.

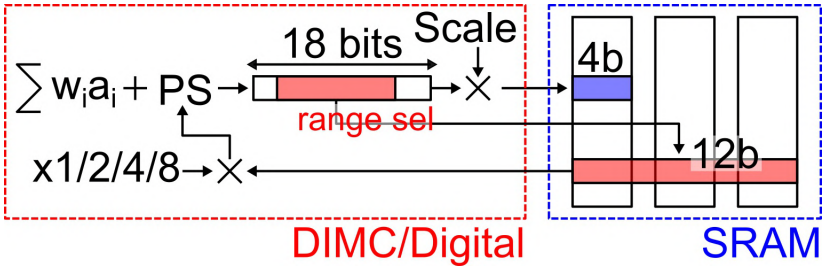


Figure 4.15: Precision control for large accumulation.

4.3.4 Precision control for the partial sum

Since a large number of values can be accumulated together, special attention is given to handling potential overflow situations. Fig.4.15 illustrates the four supported modes for managing this. While the partial sums stored in SRAM are limited to 12 bits, 18 bits are allocated for the original data. The user can select the dynamic range based on the accumulation depth of each layer. For early layers with fewer input channels, the first 12 bits can be used for partial sums. In deeper layers with a larger number of input channels, bits 4 through 16 can be selected to accommodate larger sums. This approach sacrifices precision in the LSB but effectively prevents overflow. In all modes, the final outputs are quantized to 4 bits whose dynamic range is compressed by a floating point number for scaling. Since layers with more input channels are quantized using smaller scaling factors, the impact of LSB differences on the final output is less significant.

4.4 Measurement results

4.4.1 Basic measurement

This chip is fabricated in TSMC 16nm FinFET technology. The chip is designed under 200MHz sign-off speed. Higher speed sign-off is possible but would in the given application only lead to increased power consumption. Fig.4.16 shows the measurement setup and a chip photo. The total core area is 1.9mm x 1.9mm. Fig.4.17 shows the measured shmoo plot of 4 samples. The definition of failure is the occurrence of the first error in any MAC results of all macros on one sample. All samples can run at maximal 165MHz at 0.88V core voltage. Fig.4.18 shows the frequency versus efficiency plot at both SoC level and macro level.

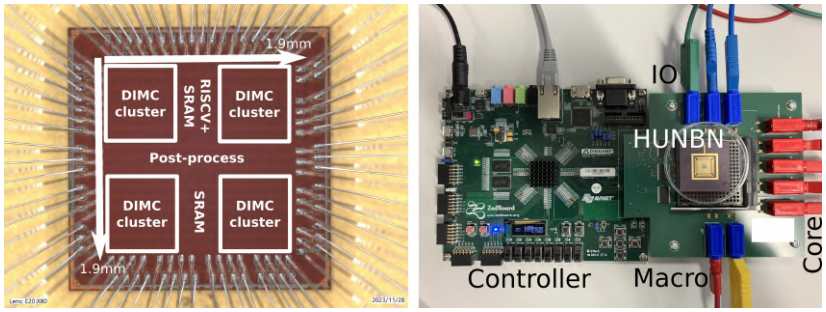


Figure 4.16: Measurement setup and chip photo.

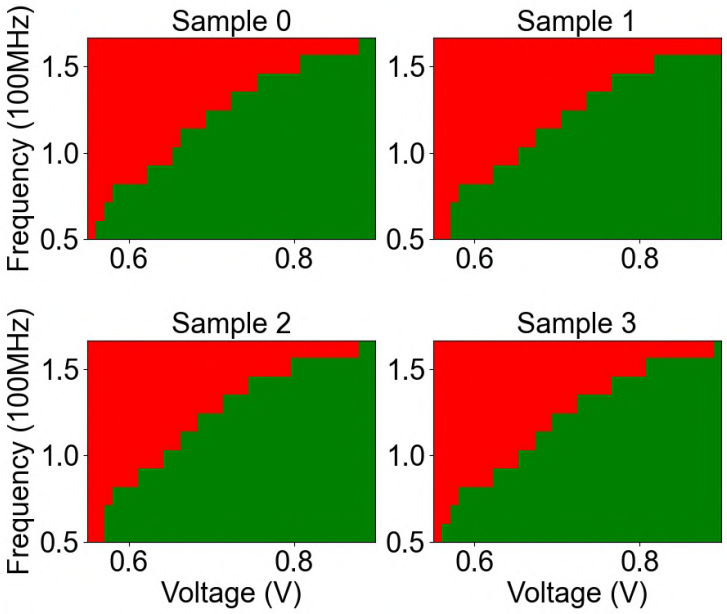


Figure 4.17: Shmoo plot for 4 measured samples.

The SoC level energy efficiency includes every energy source on-chip, excluding the IO power domain. This work achieves 24.28TOPs/W at system level and 126.3TOPs/W at macro level for purely random test data. Table 4.2 presents a comparison with state-of-the-art solutions. This work offers the highest on-chip IMC capacity, with a total memory density of 508 KB/mm^2. Using a figure of merit (FoM) defined as (MB/mm^2) $\times$ (#MACs/mm^2), this design achieves

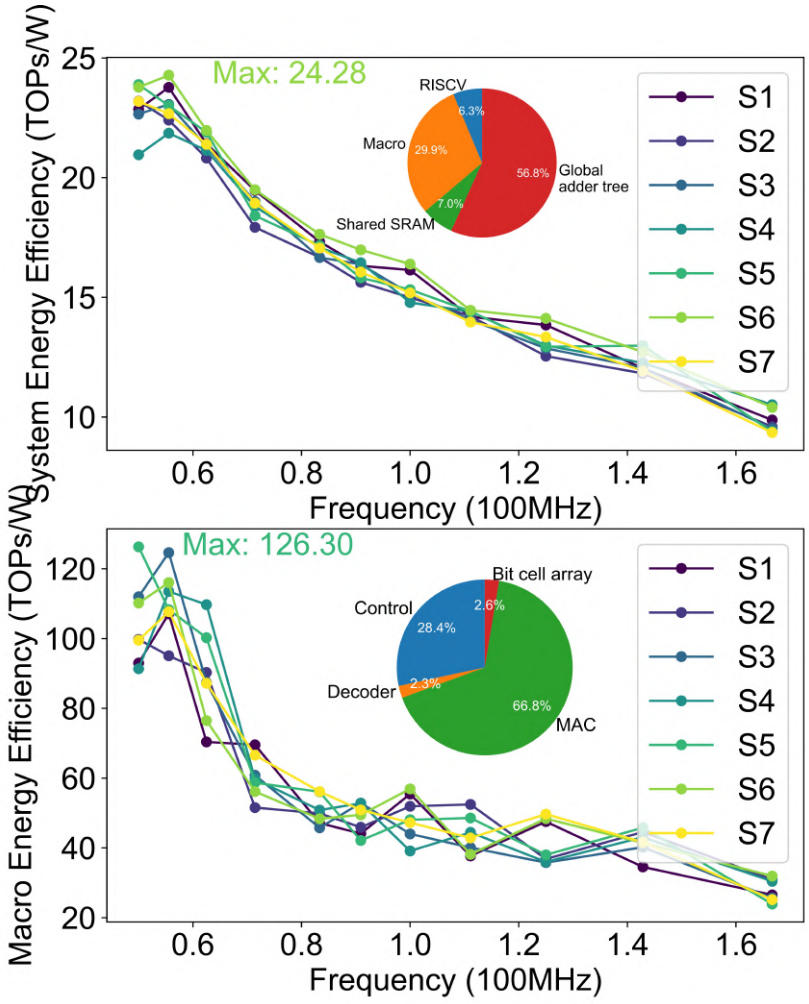


Figure 4.18: Energy efficiency of 7 measured samples under different frequency. The input and weight are pure random numbers.

0.64K, representing a 20% improvement over the best competitor. The FoM indicates that our chip is optimally suited for edge applications that prohibit off-chip DRAM access. Its superior macro-level area utilization efficiency reduces the required chip area for the same workload compared to other designs, conserving valuable chip space. This area savings can be redirected towards manufacturing cost reduction or allocated for other critical functions.

Table 4.2: Comparison to SoTA DIMC macros and SoCs.

Design	**Macro chips**			**SoC chips**			
Reference	[64]	[25]	[85]	[20]	[107]	[109]	***This work***
Technology	4nm	5nm	28nm	18nm	28nm	55nm	16nm
Technology scale (um^2)[a]	0.0199[b]	0.021	0.124	0.074[c]	0.124	0.4	0.074
Core area (mm^2)	0.0172	0.0133	6.69	4.2	4.536	12.87	3.61
IMC capacity (KB)	8	6.75	12	128	8	128	1180
Total memory (MB)	NA	NA	0.14MB	NA	0.552	1.392	1.835
Total memory density[d] (KB/mm^2)	125	144	35	NA	204	583	508
Total MAC density[d,e] (K/mm^2)	9	21	1.84	NA	1.5	0.86	1.27
Max throughput[e] (TOPs)	3.4	2.94	5.36	NA	1.64	1.02	1.53
SoC efficiency[e] (TOPs/W)	NA	NA	78[f]	NA	51[f]	5.62	24
FoM[g]	NA	NA	NA	NA	0.3K	0.5K	0.64K

[a]Technology scale is defined by the area of an SRAM bit cell.
[b,c]4nm bit cell area is missing. 3nm data is used. 18nm bit cell area is missing. 16nm data is used.
[d]All density numbers are scaled to 16nm by the technology scale.
[e]Scaled to 4bx4b MAC units. All ops are 4b. 1 MAC=2 ops. No sparsity considered.
[f]DIMC loading energy overhead not included in the number.
[g]FoM is defined as (MB/area)×(#MAC/area). Both areas are scaled based on technology scale.

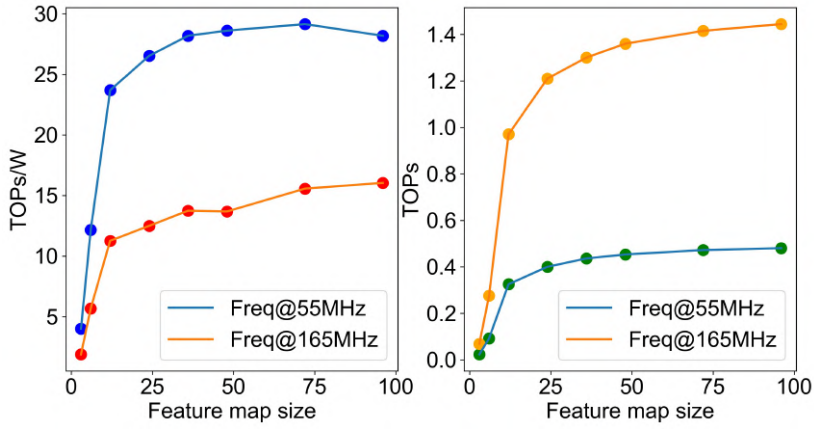


Figure 4.19: Energy efficiency and throughput versus different feature map size.

4.4.2 Evaluation on different layer type

Finally, HUHBN's flexibility and efficiency are demonstrated by mapping complete end-to-end workloads. Fig.4.19 illustrates the equivalent energy efficiency and throughput for convolution operations across varying feature map sizes. The convolution kernel is fixed with dimensions FX=3, FY=3, C=32, and K=32, while both input and weight sparsity levels are maintained at 50%. For small feature map sizes, the SoC experiences reduced energy efficiency and throughput due to frequent line switching and DIMC stalling compared to large feature map sizes. In this scenario, clock power and static energy consumption dominate the overall power usage. As the feature map size increases, both energy efficiency and throughput show significant improvement. However, this gain begins to plateau around a feature map size of 24, where the control power overhead becomes negligible. The maximum energy efficiency achieved is 29 TOPS/W at 0.42 TOPS. In conclusion, unless the feature map is extremely small, the SoC can operate at its full potential, delivering optimal performance.

Fig.4.20 illustrates the performance across different convolution kernels, varying in dimensions FX/FY and input channels C, while keeping the output channel fixed at K=32 for simplicity. A periodic pattern is observed in relation to both the input channel count and the filter size. The SoC operates most efficiently when the input channel count (C) is a multiple of 32, and the filter size (FX/FY) is a multiple of 3. Notably, the performance significantly degrades for cases like FX=FY=1 (e.g., point-wise convolution), due to severe under-utilization of the hardware resources.

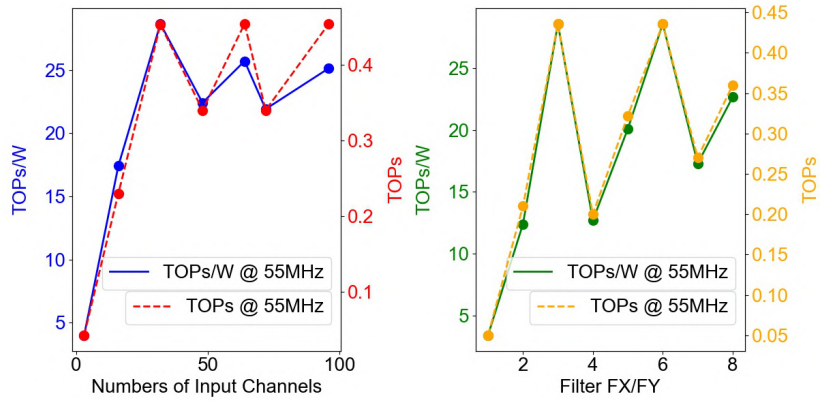


Figure 4.20: Energy efficiency and throughput versus different feature map size.

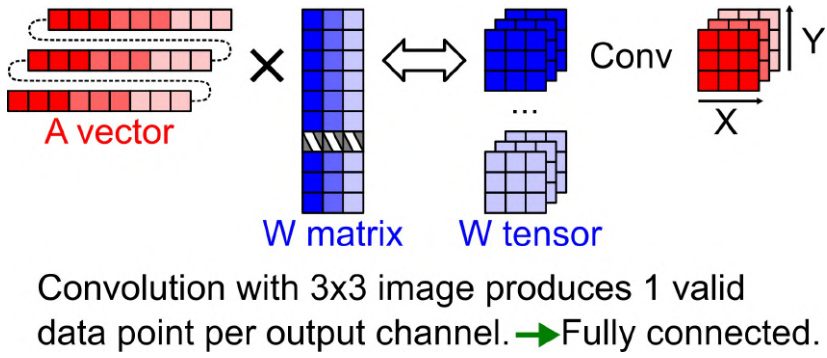


Figure 4.21: Replacing fully connected layer with convolution on 3x3 input feature map.

4.4.3 Evaluation on different workloads

Unet and ResNet8

A UNet model and ResNet8 have been fully mapped onto the SoC. The U-Net is a customized 4-bit model without significant accuracy drop. The parameters for the UNet are shown in Table.4.3. The ResNet8 and other benchmark are directly taken and quantized to 4b according to [8]. Fig.4.22 provides a visual of the U-Net architecture alongside measurement data for both U-Net and ResNet8. For the U-Net, the long skip connections are efficiently implemented on-chip. The SoC achieves an average system-level performance of 19 TOPS/W and 0.3

Layer number	C	K	FX/FY	IX/IY	Transposed?
1	3	32	3	96	No
2	32	32	3	96	No
3	32	64	3	48	No
4	64	64	3	48	No
5	64	128	3	24	No
6	128	128	3	24	No
7	128	256	3	12	No
8	256	256	3	12	No
9	256	128	3	12	Yes
10	256	64	4	24	Yes
11	128	5	4	48	Yes

Table 4.3: Parameters used for the example UNet model.

TOPS across all layers, consuming a total of 0.11 mJ per frame at 155 fps. For ResNet8, we achieve an average of 13.6 TOPS/W and 0.18 TOPS specifically for the convolution layers, with a total power consumption of 1.5 uJ per frame at 8.44 Kfps. The average system-level performance for UNet is 80% of the peak performance, and for ResNet8 is 57% of the peak performance, demonstrating the benefits of flexible dataflow support. Notably, the fully connected layer at the end of ResNet8 has been replaced by a convolution layer with dimensions FY=3, FX=3, applied on a 2D input with X-dimension=Y-dimension=3, as the Fig.4.21 shows. This adjustment gives equivalent computations. Apparently, such approximation results in many stalls in the system due to the limited size of the input feature map. All subsequent fully connected layers in the model are handled similarly to maintain computational efficiency.

MobileNetV1 and DS-CNN

Fig.4.22 shows the measurement data for MobileNetV1 [8] and DS-CNN [8]. For MobileNetV1, this work achieves 1.9 TOPs/W and 0.02 TOPs for MobileNetV1, and 3.7 uJ per frame at 2.88 Kfps. For DS-CNN, this work achieves 2.08 TOPs/W and 0.025 TOPs, and 5.09 uJ per frame at 2.44 Kfps. Unlike Unet and ResNet8, point-wise and depth-wise filters dominate these models, which lead to under utilization of the array. This explains the efficiency gap between MobileNetV1/DS-CNN and UNet/ResNet8.

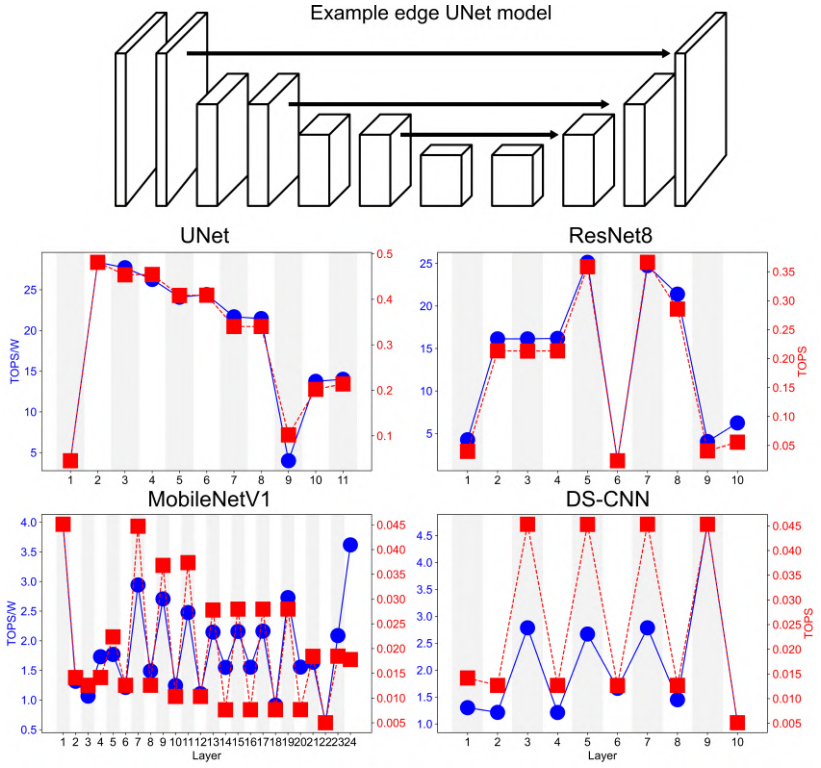


Figure 4.22: Performance analysis for UNet, ResNet8, MobileNetV1, and DS-CNN.

Deep-autoencoder

Deep-autoencoder [8] is also tested on the SoC. All gemm operators are replaced with approximate convolution layers. Fig.4.23 shows the measurement data for deep-autoencoder. This work achieves 0.9 TOPs/W and 5 GOPs, and 0.17 uJ per frame at 30.8 Kfps.

4.5 Conclusion

This work presents a high-density digital in-memory compute (DIMC) SoC designed for edge CNN applications. By utilizing pushed-rule memory cells, HUNBN can store the entire edge-based CNN within the DIMC memory,

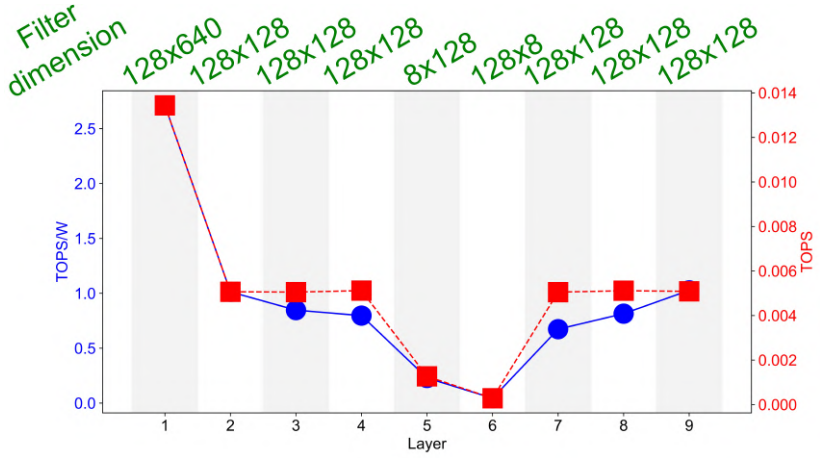


Figure 4.23: Performance analysis for deep-autoencoder.

significantly reducing energy costs associated with frequent on-chip weight loading. More importantly, through innovative DIMC operation techniques and a customized dataflow, HUNBN maximizes data reuse in CNNs, achieving exceptional energy efficiency and throughput. As a result, we achieve an SoC energy efficiency of 24 TOPs/W for edge CNN models. Furthermore, the performance is adaptable to other models, thanks to the high area utilization ratio and the built-in dataflow flexibility of the DIMC processor.

Chapter 5

Conclusion

5.1 Contributions

This thesis presented a complete exploration of low-energy Digital In-Memory Compute (DIMC) architectures, addressing the critical need for energy-efficient AI processing at the edge. Through the design and evaluation of novel DIMC macros and their integration into a custom System-on-Chip (SoC), several challenges inherent to In-Memory Compute were addressed, and multiple innovations were introduced.

The key conclusions from this work can be summarized as follows:

- **DIMC Macro Innovation:** Two prototypes of DIMC macros were developed in 16nm FinFET technology. The first prototype introduced novel energy-saving techniques such as reverse pre-charging and charge reuse. The second prototype enhanced both area and energy efficiency by adopting synchronous operation, using foundry-compatible 6T bit cells, and a Gray-code style decoder design.
- **Energy and Area Efficiency:** The second prototype achieved substantial improvements, demonstrating:
  - Area efficiency up to 725KUnit/mm^2.
  - Effective peripheral design that minimized overhead and supported high-density macro integration.
- **System-Level Integration:** A DIMC-based SoC named `HUNBN` was developed, integrating the optimized DIMC macros with a dedicated

level shifter, controller logic, and a precision-scalable MAC dataflow. It supports edge AI workloads with varying precision requirements and memory constraints.

- **Innovations in Dataflow and Scheduling:**
  - A split-MAC dataflow was proposed for pipeline efficiency and throughput maximization.
  - Dynamic partial sum precision control enabled flexible memory utilization with minimal accuracy loss.
  - Data reuse strategies and optimized activation placement reduced energy spent on memory access.
- **Measurement and Benchmarking:** Silicon measurements of the SoC show:
  - Peak system level energy efficiency of up to 24 TOPS/W at 0.6V.
  - Versatility across CNN layer types.
  - Robust performance across different chip samples.

In summary, this work provides a solid foundation for future DIMC-based edge inference accelerators. By combining circuit-level innovations with system-level co-design, it demonstrates that DIMC is not only viable but highly advantageous for energy-constrained AI applications.

5.2 Future works

The author identifies four promising directions for future researchers and engineers. Without loss of generality, we use the term IMC throughout this discussion, because some directions are relevant to AIMC.

- **Combining In-Memory-Compute with other algorithms.** This thesis primarily focuses on CNNs, which are among the most popular AI models in recent years. Researchers have also explored IMC architectures for matrix-multiplication-based models, such as transformers [87, 86, 60, 27], largely driven by the AI boom from 2018 to the present. However, many other interesting algorithms are also well-suited to IMC. For instance, the Ising model has recently been combined with IMC [6]. These algorithms typically require lower precision and benefit from high parallelism—characteristics that align well with IMC

capabilities. Similarly, polynomial-based algorithms, such as those used in Reed–Solomon codes [95], zero-knowledge proofs [96], and homomorphic encryption [94], rely on modular arithmetic. IMC's native support for integer operations and truncation procedures makes it a strong candidate for such applications, potentially enabling advances in security, coding schemes, and even NP problem-solving.

- **Flexibility modeling.** This thesis discusses the trade-offs in IMC design at the macro level, specifically between area efficiency and energy efficiency. However, this trade-off is further complicated by macro flexibility. As explored in Chapter 1, Section 1.7.2, choosing macros with varying energy and area efficiency significantly impacts system-level energy consumption. A more flexible macro may support a wider range of applications but often comes with hard-to-quantify design trade-offs. For instance, comparing [112] and [107], both using the same technology, shows a tenfold difference in BF16 MAC efficiency—primarily due to whether alignment is done in the macro ([112]) or digitally ([107]). Accurately modeling macro flexibility would greatly enhance IMC design comparisons.

- **Heterogeneous systems for edge applications.** In the HUNBN system, a variety of operations are supported, but this comes at the expense of system-level energy efficiency. As discussed earlier, modeling macro flexibility is difficult. An alternative is to design heterogeneous IMC cores, each optimized for specific tasks. For example, in CNNs, some cores can target standard convolution layers while others are tailored for transposed convolutions. This raises challenges in compiler design and on-chip interconnects, especially considering the asymmetry between IMC write and IMC read operations.

- **AIMC or DIMC?** Finally, we revisit the question posed in Chapter 1. For edge applications involving models that fit within on-chip SRAM, DIMC is the most viable. However, for larger models such as large language models (LLMs), SRAM capacity is insufficient. While some efforts aim to support LLMs on large DIMC systems with expansive SRAM [1], this approach lacks scalability. When model parameters reach gigabyte scale and off-chip memory movement becomes unavoidable, the energy inefficiency of analog circuits becomes less of a bottleneck. This motivates AIMC in Flash memory—leveraging Flash's scalability and combining storage with computation to achieve true "in-memory compute."

Appendix A

List of supported convolution layers on HUNBN

The supported CNN layers on HUNBN is given in this appendix. Note that for all the C loop conditions that have no upper-bound, the upper-bound can only be dynamically determined by the availability of the on-chip SRAM, thus not given.

- Group 1: $0 < C \leq 32$. K=any. $IX \leq 128$, $IY \leq 128$. $FX = FY = 3$. Stride=0/1. Transposed=0.
- Group 2: $C > 32$. K=any. $IX \leq 64$, $IY \leq 64$. $FX = FY = 3$. Stride=0/1. Transposed=0.
- Group 3: $0 < C \leq 32$. K=any. $IX \leq 64$, $IY \leq 64$. $FX = FY = 3$. Stride=0. Transposed=1.
- Group 4: $C > 32$. K=any. $IX \leq 32$, $IY \leq 32$. $FX = FY = 3$. Stride=0. Transposed=1.
- Group 5: $0 < C \leq 18$. K=any. $IX \leq 64$, $IY \leq 64$. $FX = FY = 4$. Transposed=1.
- Group 6: $C > 18$. K=any. $IX \leq 32$, $IY \leq 32$. $FX = FY = 4$. Transposed=1.

Bibliography

[1] d-Matrix - Ultra-low Latency Batched Inference for Generative AI — d-matrix.ai. https://www.d-matrix.ai/. [Accessed 01-04-2025].

[2] GRAPHICS DOUBLE DATA RATE (GDDR5) SGRAM STANDARD | JEDEC — jedec.org. https://www.jedec.org/standards-documents/docs/jesd212c. [Accessed 27-02-2025].

[3] HIGH BANDWIDTH MEMORY (HBM) DRAM | JEDEC — jedec.org. https://www.jedec.org/standards-documents/docs/jesd235a. [Accessed 27-02-2025].

[4] TSMC N3 Nodes Show SRAM Scaling is Hitting the Wall — techpowerup.com. https://www.techpowerup.com/309331/tsmc-n3-nodes-show-sram-scaling-is-hitting-the-wall?cp=1. [Accessed 13-01-2025].

[5] Ali, M., Agrawal, A., and Roy, K. Ramann: in-sram differentiable memory computations for memory-augmented neural networks. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design* (New York, NY, USA, 2020), ISLPED '20, Association for Computing Machinery, p. 61–66.

[6] Bae, J., Shim, C., and Kim, B. 15.6 e-chimera: A scalable sram-based ising macro with enhanced-chimera topology for solving combinatorial optimization problems within memory. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)* (2024), vol. 67, pp. 286–288.

[7] Banbury, C., and Reddi, V. J. Benchmarking tinyml systems: Challenges and opportunities. In *Proceedings of the NeurIPS TinyML Workshop* (2021).

[8] Banbury, C., Reddi, V. J., Torelli, P., Holleman, J., Jeffries, N., Kiraly, C., Montino, P., Kanter, D., Ahmed, S., Pau, D., et al. Mlperf tiny benchmark. *arXiv preprint arXiv:2106.07597* (2021).

[9] Biswas, A., and Chandrakasan, A. P. Conv-sram: An energy-efficient sram with in-memory dot-product computation for low-power convolutional neural networks. *IEEE Journal of Solid-State Circuits 54*, 1 (2019), 217–230.

[10] Breyer, E. T., Mulaosmanovic, H., Mikolajick, T., and Slesazeck, S. Perspective on ferroelectric, hafnium oxide based transistors for digital beyond von-neumann computing. *Applied Physics Letters 118*, 5 (02 2021), 050501.

[11] Burr, G. W., Shelby, R. M., Sidler, S., di Nolfo, C., Jang, J., Boybat, I., Shenoy, R. S., Narayanan, P., Virwani, K., Giacometti, E. U., Kurdi, B. N., and Hwang, H. Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element. *IEEE Transactions on Electron Devices 62*, 11 (2015), 3498–3507.

[12] Cassinerio, M., Ciocchini, N., and Ielmini, D. Logic computation in phase change materials by threshold and memory switching. *Advanced Materials 25*, 41 (2013), 5975–5980.

[13] Chang, E.-J., Xue, C.-X., Deshpande, C., Jedhe, G., Liang, J., Cheng, C.-C., Lin, H.-W., Lee, C.-D., Kumar, S., Jway, K. S., Guo, Z., Garg, R., Lu, A.-C., Lin, C.-H., Hsieh, M.-H., Lin, T.-Y., and Chen, C.-C. A 12-nm 0.62-1.61 mw ultra-low power digital cim-based deep-learning system for end-to-end always-on vision. In *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)* (2023), pp. 1–2.

[14] Chen, W.-H., Dou, C., Li, K.-X., Lin, W.-Y., Li, P.-Y., Huang, J.-H., Wang, J.-H., Wei, W.-C., Xue, C.-X., Chiu, Y.-C., King, Y.-C., Lin, C.-J., Liu, R.-S., Hsieh, C.-C., Tang, K.-T., Yang, J. J., Ho, M.-S., and Chang, M.-F. Cmos-integrated memristive non-volatile computing-in-memory for ai edge processors. *Nature Electronics 2* (2019), 420 – 428.

[15] Chen, W.-H., Lin, W.-J., Lai, L.-Y., Li, S., Hsu, C.-H., Lin, H.-T., Lee, H.-Y., Su, J.-W., Xie, Y., Sheu, S.-S., and Chang, M.-F. A 16mb dual-mode reram macro with sub-14ns computing-in-memory and memory functions enabled by self-write termination scheme. In *2017 IEEE International Electron Devices Meeting (IEDM)* (2017), pp. 28.2.1–28.2.4.

[16] Chih, Y.-D., Lee, P.-H., Fujiwara, H., Shih, Y.-C., Lee, C.-F., Naous, R., Chen, Y.-L., Lo, C.-P., Lu, C.-H., Mori, H.,

Zhao, W.-C., Sun, D., Sinangil, M. E., Chen, Y.-H., Chou, T.-L., Akarvardar, K., Liao, H.-J., Wang, Y., Chang, M.-F., and Chang, T.-Y. J. 16.4 an 89tops/w and 16.3tops/mm2 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)* (2021), vol. 64, pp. 252–254.

[17] Corporation, I. *Intel 64 and IA-32 Architectures Software Developer's Manual*. Intel Corporation, 2021. Available online: `https://software.intel.com/`.

[18] Cosemans, S., Dehaene, W., and Catthoor, F. A 3.6pj/access 480mhz, 128kbit on-chip sram with 850mhz boost mode in 90nm cmos with tunable sense amplifiers to cope with variability. In *ESSCIRC 2008 - 34th European Solid-State Circuits Conference* (2008), pp. 278–281.

[19] Dennard, R. H., Gaensslen, F. H., and Yu, H.-N. Design of ion-implanted mosfets with very small physical dimensions. *IEEE Journal of Solid-State Circuits 9*, 5 (1974), 256–268.

[20] Desoli, G., Chawla, N., Boesch, T., Avodhyawasi, M., Rawat, H., Chawla, H., Abhijith, V., Zambotti, P., Sharma, A., Cappetta, C., Rossi, M., De Vita, A., and Girardi, F. 16.7 a 40-310tops/w sram-based all-digital up to 4b in-memory computing multi-tiled nn accelerator in fd-soi 18nm for deep-learning edge applications. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 260–262.

[21] Diao, H., Luo, H., Song, J., Xu, B., Wang, R., Wang, Y., and Tang, X. A 28nm 128tflops/w computing-in-memory engine supporting one-shot floating-point nn inference and on-device fine-tuning for edge ai. In *2024 IEEE Custom Integrated Circuits Conference (CICC)* (2024), pp. 1–2.

[22] Eckert, C., Wang, X., Wang, J., Subramaniyan, A., Iyer, R., Sylvester, D., Blaaauw, D., and Das, R. Neural cache: Bit-serial in-cache acceleration of deep neural networks. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)* (2018), pp. 383–396.

[23] Eckert, C., Wang, X., Wang, J., Subramaniyan, A., Sylvester, D., Blaauw, D., Das, R., and Iyer, R. Neural cache: Bit-serial in-cache acceleration of deep neural networks. *IEEE Micro 39*, 3 (2019), 11–19.

[24] Fassio, L., Settino, F., Lin, L., De Rose, R., Lanuzza, M., Crupi, F., and Alioto, M. A robust, high-speed and energy-efficient ultralow-voltage level shifter. *IEEE Transactions on Circuits and Systems II: Express Briefs 68*, 4 (2021), 1393–1397.

[25] Fujiwara, H., Mori, H., Zhao, W.-C., Chuang, M.-C., Naous, R., Chuang, C.-K., Hashizume, T., Sun, D., Lee, C.-F., Akarvardar, K., Adham, S., Chou, T.-L., Sinangil, M. E., Wang, Y., Chih, Y.-D., Chen, Y.-H., Liao, H.-J., and Chang, T.-Y. J. A 5-nm 254-tops/w 221-tops/mm2 fully-digital computing-in-memory macro supporting wide-range dynamic-voltage-frequency scaling and simultaneous mac and write operations. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)* (2022), vol. 65, pp. 1–3.

[26] Fujiwara, H., Mori, H., Zhao, W.-C., Khare, K., Lee, C.-E., Peng, X., Joshi, V., Chuang, C.-K., Hsu, S.-H., Hashizume, T., Naganuma, T., Tien, C.-H., Liu, Y.-Y., Lai, Y.-C., Lee, C.-F., Chou, T.-L., Akarvardar, K., Adham, S., Wang, Y., Chih, Y.-D., Chen, Y.-H., Liao, H.-J., and Chang, T.-Y. J. 34.4 a 3nm, 32.5tops/w, 55.0tops/mm2 and 3.78mb/mm2 fully-digital compute-in-memory macro supporting int12 × int12 with a parallel-mac architecture and foundry 6t-sram bit cell. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)* (2024), vol. 67, pp. 572–574.

[27] Guo, A., Chen, X., Dong, F., Chen, J., Yuan, Z., Hu, X., Zhang, Y., Zhang, J., Tang, Y., Zhang, Z., Chen, G., Yang, D., Zhang, Z., Ren, L., Xiong, T., Wang, B., Liu, B., Shan, W., Liu, X., Cai, H., Sun, G., Yang, J., and Si, X. 34.3 a 22nm 64kb lightning-like hybrid computing-in-memory macro with a compressed adder tree and analog-storage quantizers for transformer and cnns. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)* (2024), vol. 67, pp. 570–572.

[28] Guo, A., Si, X., Chen, X., Dong, F., Pu, X., Li, D., Zhou, Y., Ren, L., Xue, Y., Dong, X., Gao, H., Zhang, Y., Zhang, J., Kong, Y., Xiong, T., Wang, B., Cai, H., Shan, W., and Yang, J. A 28nm 64-kb 31.6-tflops/w digital-domain floating-point-computing-unit and double-bit 6t-sram computing-in-memory macro for floating-point cnns. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 128–130.

[29] Guo, K., Li, W., Zhong, K., Zhu, Z., Zeng, S., Xie, T., Han, S., Xie, Y., Debacker, P., Verhelst, M., and Wang, Y. Neural network accelerator comparison.

[30] Guo, R., Wang, L., Chen, X., Sun, H., Yue, Z., Qin, Y., Han, H., Wang, Y., Tu, F., Wei, S., Hu, Y., and Yin, S. 20.2 a 28nm 74.34tflops/w bf16 heterogenous cim-based accelerator exploiting denoising-similarity for diffusion models. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)* (2024), vol. 67, pp. 362–364.

[31] Guo, R., Yue, Z., Si, X., Hu, T., Li, H., Tang, L., Wang, Y., Liu, L., Chang, M.-F., Li, Q., Wei, S., and Yin, S. 15.4 a 5.99-to-691.1tops/w tensor-train in-memory-computing processor using bit-level-sparsity-based optimization and variable-precision quantization. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)* (2021), vol. 64, pp. 242–244.

[32] Gupta, A., Verma, M., and Singh, S. K. Edge ai: Challenges and opportunities for next-generation applications. *IEEE Internet of Things Journal 9*, 4 (2022), 2012–2025.

[33] Hager, P. A., Moons, B., Cosemans, S., Papistas, I. A., Rooseleer, B., Loon, J. V., Uytterhoeven, R., Zaruba, F., Koumousi, S., Stanisavljevic, M., Mach, S., Mutsaards, S., Aljameh, R. K., Khov, G. H., Machiels, B., Olar, C., Psarras, A., Geursen, S., Vermeeren, J., Lu, Y., Maringanti, A., Ameta, D., Katselas, L., Hütter, N., Schmuck, M., Sivadas, S., Sharma, K., Oliveira, M., Aerne, R., Sharma, N., Soni, T., Bussolino, B., Pesut, D., Pallaro, M., Podlesnii, A., Lyrakis, A., Ruffiner, Y., Dazzi, M., Thiele, J., Goetschalckx, K., Bruschi, N., Doevenspeck, J., Verhoef, B., Linz, S., Garcea, G., Ferguson, J., Koltsidas, I., and Eleftheriou, E. 11.3 metis aipu: A 12nm 15tops/w 209.6tops soc for cost- and energy-efficient inference at the edge. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)* (2024), vol. 67, pp. 212–214.

[34] Han, D., Ryu, J., Kim, S., Kim, S., and Yoo, H.-J. 2.7 metavrain: A 133mw real-time hyper-realistic 3d-nerf processor with 1d-2d hybrid-neural engines for metaverse on mobile devices. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 50–52.

[35] Han, S., and Kang, J. Quantization and pruning techniques for efficient deep learning models. *IEEE Transactions on Artificial Intelligence 1*, 3 (2020), 284–299.

[36] Han, S., and Lee, Y. A complementary positive feedback-assisted current mirror-based level shifter for efficient wide voltage range operation. In *2024 IEEE European Solid-State Electronics Research Conference (ESSERC)* (2024), pp. 613–616.

[37] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.

[38] He, Y., Diao, H., Tang, C., Jia, W., Tang, X., Wang, Y., Yue, J., Li, X., Yang, H., Jia, H., and Liu, Y. 7.3 a 28nm 38-to-102-tops/w 8b multiply-less approximate digital sram compute-in-memory macro for neural-network inference. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 130–132.

[39] He, Y., Fan, S., Li, X., Lei, L., Jia, W., Tang, C., Li, Y., Huang, Z., Du, Z., Yue, J., Li, X., Yang, H., Jia, H., and Liu, Y. 34.7 a 28nm 2.4mb/mm2 6.9 - 16.3tops/mm2 edram-lut-based digital-computing-in-memory macro with in-memory encoding and refreshing. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)* (2024), vol. 67, pp. 578–580.

[40] Hoffer, B., Wainstein, N., Neumann, C. M., Pop, E., Yalon, E., and Kvatinsky, S. Stateful logic using phase change memory. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits 8*, 2 (2022), 77–83.

[41] Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[42] Huang, C., and Jiao, H. C3mls: An ultra-wide-range energy-efficient level shifter with ccls/cmls hybrid structure. *IEEE Journal of Solid-State Circuits 58*, 10 (2023), 2685–2695.

[43] Jain, S., Ranjan, A., Roy, K., and Raghunathan, A. Computing in memory with spin-transfer torque magnetic ram. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 26*, 3 (2018), 470–483.

[44] Jaiswal, A., Chakraborty, I., Agrawal, A., and Roy, K. 8t sram cell as a multibit dot-product engine for beyond von neumann computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27*, 11 (2019), 2556–2567.

[45] Jedhe, G., Deshpande, C., Kumar, S., Xue, C.-X., Guo, Z., Garg, R., Jway, K. S., Chang, E.-J., Liang, J., Wan, Z., and Pan, Z. A 12nm 137 tops/w digital compute-in-memory using foundry 8t sram bitcell supporting 16 kernel weight sets for ai edge applications. In *2023*

IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits) (2023), pp. 1–2.

[46] Jeloka, S., Akesh, N. B., Sylvester, D., and Blaauw, D. A 28 nm configurable memory (tcam/bcam/sram) using push-rule 6t bit cell enabling logic-in-memory. *IEEE Journal of Solid-State Circuits 51*, 4 (2016), 1009–1021.

[47] Jeon, I., Park, H., Yoon, T., and Jeong, H. High efficiency variation-aware sram timing characterization via machine-learning-assisted netlist extraction. *IEEE Transactions on Circuits and Systems II: Express Briefs 71*, 3 (2024), 1391–1395.

[48] Jeong, H., Kim, T.-H., Park, C. N., Kim, H., Song, T., and Jung, S.-O. A wide-range static current-free current mirror-based ls with logic error detection for near-threshold operation. *IEEE Journal of Solid-State Circuits 56*, 2 (2021), 554–565.

[49] Jia, H., Ozatay, M., Tang, Y., Valavi, H., Pathak, R., Lee, J., and Verma, N. 15.1 a programmable neural-network inference accelerator based on scalable in-memory computing. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)* (2021), vol. 64, pp. 236–238.

[50] Kim, D., Jang, Y., Kim, T., and Park, J. Bimdim: Area efficient bi-directional mram digital in-memory computing. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (2022), pp. 74–77.

[51] Kim, S., Park, H., and Choi, J. Lb-unet: A lightweight boundary-assisted unet for skin lesion segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2024), Springer, pp. 211–223.

[52] Kim, Y., Lee, Y., Sylvester, D., and Blaauw, D. Slc: Split-control level converter for dense and stable wide-range voltage conversion. In *2012 Proceedings of the ESSCIRC (ESSCIRC)* (2012), pp. 478–481.

[53] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)* (2012).

[54] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)* (2012), pp. 1097–1105.

[55] Leadbetter, R. Intel core i7-5960x review: Haswell-e, x99 chipset and ddr4, 2014. Accessed: 2024-10-31.

[56] Lee, C., Zhang, M., and Banerjee, A. L³u-net: Low-latency lightweight u-net for parallel cnn processors in real-time segmentation. *ACM Transactions on Embedded Computing Systems 21*, 3 (2022), 45–62.

[57] Lin, C.-T., Huang, P. X., Oh, J., Wang, D., and Seok, M. imcu: A 102-j, 61-ms digital in-memory computing-based microcontroller unit for edge tinyml. In *2023 IEEE Custom Integrated Circuits Conference (CICC)* (2023), pp. 1–2.

[58] Lin, C.-T., Oh, J., Lee, K., and Seok, M. Star-sram: 43.06-tflops/w, 1.89-tflops/mm2, 400-kb/mm2 floating-point sram-based digital computing-in-memory macro in 28-nm cmos. In *2024 IEEE Custom Integrated Circuits Conference (CICC)* (2024), pp. 1–2.

[59] Linn, E., Rosezin, R., Tappertzhofen, S., Böttger, U., and Waser, R. Beyond von neumann—logic operations in passive crossbar arrays alongside memory operations. *Nanotechnology 23*, 30 (jul 2012), 305205.

[60] Liu, S., Li, P., Zhang, J., Wang, Y., Zhu, H., Jiang, W., Tang, S., Chen, C., Liu, Q., and Liu, M. 16.2 a 28nm 53.8tops/w 8b sparse transformer accelerator with in-memory butterfly zero skipper for unstructured-pruned nn and cim-based local-attention-reusable engine. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 250–252.

[61] Mano, M. M., and Ciletti, M. D. *Digital Design: With an Introduction to the Verilog HDL, VHDL, and SystemVerilog*, 6 ed. Pearson, Boston, MA, USA, 2017.

[62] Mead, C., and Conway, L. *Introduction to VLSI Systems*. Addison-Wesley, 1980.

[63] Moore, G. Cramming more components onto integrated circuits. In *Electronics Magazine* (1965).

[64] Mori, H., Zhao, W.-C., Lee, C.-E., Lee, C.-F., Hsu, Y.-H., Chuang, C.-K., Hashizume, T., Tung, H.-C., Liu, Y.-Y., Wu, S.-R., Akarvardar, K., Chou, T.-L., Fujiwara, H., Wang, Y., Chih, Y.-D., Chen, Y.-H., Liao, H.-J., and Chang, T.-Y. J. A 4nm 6163-tops/w/b $\mathbf{4790 - TOPS/mm^2/b}$ sram based digital-computing-in-memory macro supporting bit-width flexibility and simultaneous mac and weight update. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 132–134.

[65] Neumann, J. V. First draft of a report on the edvac. *IEEE Annals of the History of Computing* (1945).

[66] Oh, J., Lin, C.-T., and Seok, M. D6cim: 60.4-tops/w, 1.46-tops/mm2, 1005-kb/mm2 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)* (2023), pp. 413–416.

[67] Park, J.-S., Park, C., Kwon, S., Kim, H.-S., Jeon, T., Kang, Y., Lee, H., Lee, D., Kim, J., Lee, Y., Park, S., Jang, J.-W., Ha, S., Kim, M., Bang, J., Lim, S. H., and Kang, I. A multi-mode 8k-mac hw-utilization-aware neural processing unit with a unified multi-precision datapath in 4nm flagship mobile soc. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)* (2022), vol. 65, pp. 246–248.

[68] Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning* (2013), PMLR, pp. 1310–1318.

[69] Patil, A. D., Hua, H., Gonugondla, S., Kang, M., and Shanbhag, N. R. An mram-based deep in-memory architecture for deep neural networks. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)* (2019), pp. 1–5.

[70] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 779–788.

[71] Saikia, J., Sridharan, A., Yeo, I., Venkataramanaiah, S., Fan, D., and Seo, J.-S. Fp-imc: A 28nm all-digital configurable floating-point in-memory computing macro. In *ESSCIRC 2023- IEEE 49th European Solid State Circuits Conference (ESSCIRC)* (2023), pp. 405–408.

[72] Schaller, R. R. Moore's law: Past, present, and future. *IEEE Spectrum 34*, 6 (1997), 52–59.

[73] Shafiee, A., et al. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)* (2016).

[74] Shih, M.-E., Hsieh, S.-W., Tsai, P.-Y., Lin, M.-H., Tsung, P.-K., Chang, E.-J., Liang, J., Chang, S.-H., Huang, C.-L., Nian, Y.-Y., Wan, Z., Kumar, S., Xue, C.-X., Jedhe, G., Fujiwara, H., Mori, H., Chen, C.-W., Huang, P.-H., Juan, C.-F., Chen,

C.-Y., Lin, T.-Y., Wang, C., Chen, C.-C., and Jou, K. 20.1 nve: A 3nm 23.2tops/w 12b-digital-cim-based neural engine for high-resolution visual-quality enhancement on smart devices. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)* (2024), vol. 67, pp. 360–362.

[75] Si, X., Tu, Y.-N., Huang, W.-H., Su, J.-W., Lu, P.-J., Wang, J.-H., Liu, T.-W., Wu, S.-Y., Liu, R., Chou, Y.-C., Zhang, Z., Sie, S.-H., Wei, W.-C., Lo, Y.-C., Wen, T.-H., Hsu, T.-H., Chen, Y.-K., Shih, W., Lo, C.-C., Liu, R.-S., Hsieh, C.-C., Tang, K.-T., Lien, N.-C., Shih, W.-C., He, Y., Li, Q., and Chang, M.-F. 15.5 a 28nm 64kb 6t sram computing-in-memory macro with 8b mac operation for ai edge chips. In *2020 IEEE International Solid-State Circuits Conference - (ISSCC)* (2020), pp. 246–248.

[76] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)* (2015).

[77] Song, Y., Wang, X., Wu, Q., Yang, F., Wang, C., Wang, M., and Miao, X. Reconfigurable and efficient implementation of 16 boolean logics and full-adder functions with memristor crossbar for beyond von neumann in-memory computing. *Advanced Science 9*, 15 (2022), 2200036.

[78] Sridharan, A., Zhang, F., Seo, J.-S., and Fan, D. Sp-imc: A sparsity aware in-memory-computing macro in 28nm cmos with configurable sparse representation for highly sparse dnn workloads. In *2024 IEEE Custom Integrated Circuits Conference (CICC)* (2024), pp. 1–2.

[79] Su, J.-W., Si, X., Chou, Y.-C., Chang, T.-W., Huang, W.-H., Tu, Y.-N., Liu, R., Lu, P.-J., Liu, T.-W., Wang, J.-H., Zhang, Z., Jiang, H., Huang, S., Lo, C.-C., Liu, R.-S., Hsieh, C.-C., Tang, K.-T., Sheu, S.-S., Li, S.-H., Lee, H.-Y., Chang, S.-C., Yu, S., and Chang, M.-F. 15.2 a 28nm 64kb inference-training two-way transpose multibit 6t sram compute-in-memory macro for ai edge chips. In *2020 IEEE International Solid-State Circuits Conference - (ISSCC)* (2020), pp. 240–242.

[80] Sun, Z., Ambrosi, E., Bricalli, A., and Ielmini, D. Logic computing with stateful neural networks of resistive switches. *Advanced Materials 30*, 38 (2018), 1802554.

[81] Sylvester, D., and Keutzer, K. Getting to the bottom of deep submicron. *IEEE Design and Test of Computers* (2006).

[82] Sze, V., Chen, Y.-H., and Yang, T.-J. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE 105*, 12 (2017), 2295–2329.

[83] Sze, V., Chen, Y.-H., Yang, T.-J., and Emer, J. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE 105*, 12 (2017), 2295–2329.

[84] Thrun, S. Robotic mapping: A survey. *Exploring Artificial Intelligence in the New Millennium* (2002).

[85] Tu, F., Wang, Y., Wu, Z., Liang, L., Ding, Y., Kim, B., Liu, L., Wei, S., Xie, Y., and Yin, S. A 28nm 29.2tflops/w bf16 and 36.5tops/w int8 reconfigurable digital cim processor with unified fp/int pipeline and bitwise in-memory booth multiplication for cloud deep learning acceleration. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)* (2022), vol. 65, pp. 1–3.

[86] Tu, F., Wu, Z., Wang, Y., Liang, L., Liu, L., Ding, Y., Liu, L., Wei, S., Xie, Y., and Yin, S. A 28nm 15.59µj/token full-digital bitline-transpose cim-based sparse transformer accelerator with pipeline/parallel reconfigurable modes. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)* (2022), vol. 65, pp. 466–468.

[87] Tu, F., Wu, Z., Wang, Y., Wu, W., Liu, L., Hu, Y., Wei, S., and Yin, S. 16.1 muitcim: A 28nm 2.24μJ/token attention-token-bit hybrid sparse digital cim-based accelerator for multimodal transformers. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 248–250.

[88] Um, S., Kim, S., Hong, S., Kim, S., and Yoo, H.-J. Log-cim: A 116.4 tops/w digital computing-in-memory processor supporting a wide range of logarithmic quantization with zero-aware 6t dual-wl cell. In *2023 IEEE Asian Solid-State Circuits Conference (A-SSCC)* (2023), pp. 1–3.

[89] Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[90] Wang, B., Xue, C., Feng, Z., Zhang, Z., Liu, H., Ren, L., Li, X., Yin, A., Xiong, T., Xue, Y., He, S., Kong, Y., Zhou, Y., Guo, A., Si, X., and Yang, J. A 28nm horizontal-weight-shift and vertical-feature-shift-based separate-wl 6t-sram computation-in-memory unit-macro for edge depthwise neural-networks. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 134–136.

[91] Wang, D., Lin, C.-T., Chen, G. K., Knag, P., Krishnamurthy, R. K., and Seok, M. Dimc: 2219tops/w 2569f2/b digital in-memory computing macro in 28nm based on approximate arithmetic hardware. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)* (2022), vol. 65, pp. 266–268.

[92] Wang, H., Xu, Z., and Liu, Y. Model compression techniques for edge ai: A review and case studies. *IEEE Transactions on Neural Networks and Learning Systems 32*, 11 (2021), 4897–4911.

[93] Wang, Y., Yang, M., Xie, S., Wang, M., and Kulkarni, J. P. Cimgn: An energy-efficient all-digital compute-in-memory graph neural network processor. In *ESSCIRC 2023- IEEE 49th European Solid State Circuits Conference (ESSCIRC)* (2023), pp. 477–480.

[94] Wikipedia. Homomorphic encryption — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Homomorphic%20encryption&oldid=1277514956`, 2025. [Online; accessed 01-April-2025].

[95] Wikipedia. Reed–Solomon error correction — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Reed%E2%80%93Solomon%20error%20correction&oldid=1283452259`, 2025. [Online; accessed 01-April-2025].

[96] Wikipedia. Zero-knowledge proof — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Zero-knowledge%20proof&oldid=1282625929`, 2025. [Online; accessed 01-April-2025].

[97] Wu, M., Ren, W., Chen, P., Zhao, W., Jing, Y., Ru, J., Wang, Z., Ma, Y., Huang, R., Jia, T., and Ye, L. S2d-cim: A 22nm 128kb systolic digital compute-in-memory macro with domino data path for flexible vector operation and 2-d weight update in edge ai applications. In *2024 IEEE Custom Integrated Circuits Conference (CICC)* (2024), pp. 1–2.

[98] Wu, P.-C., Su, J.-W., Chung, Y.-L., Hong, L.-Y., Ren, J.-S., Chang, F.-C., Wu, Y., Chen, H.-Y., Lin, C.-H., Hsiao, H.-M., Li, S.-H., Sheu, S.-S., Chang, S.-C., Lo, W.-C., Lo, C.-C., Liu, R.-S., Hsieh, C.-C., Tang, K.-T., Wu, C.-I., and Chang, M.-F. A 28nm 1mb time-domain computing-in-memory 6t-sram macro with a 6.6ns latency, 1241gops and 37.01tops/w for 8b-mac operations for edge-ai devices. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)* (2022), vol. 65, pp. 1–3.

[99] Wu, P.-C., Su, J.-W., Hong, L.-Y., Ren, J.-S., Chien, C.-H., Chen, H.-Y., Ke, C.-E., Hsiao, H.-M., Li, S.-H., Sheu, S.-S., Lo, W.-C., Chang, S.-C., Lo, C.-C., Liu, R.-S., Hsieh, C.-C., Tang, K.-T., and Chang, M.-F. A 22nm 832kb hybrid-domain floating-point sram in-memory-compute macro with 16.2-70.2tflops/w for high-accuracy ai-edge devices. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 126–128.

[100] Wu, Y., Zhang, J., and Gao, L. Privacy and latency optimization in edge ai applications: A survey. *ACM Transactions on Embedded Computing Systems 22*, 1 (2023), 45–68.

[101] Xie, S., Ni, C., Sayal, A., Jain, P., Hamzaoglu, F., and Kulkarni, J. P. 16.2 edram-cim: Compute-in-memory design with reconfigurable embedded-dynamic-memory array realizing adaptive data converters and charge-domain computing. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)* (2021), vol. 64, pp. 248–250.

[102] Yan, B., Hsu, J.-L., Yu, P.-C., Lee, C.-C., Zhang, Y., Yue, W., Mei, G., Yang, Y., Yang, Y., Li, H., Chen, Y., and Huang, R. A 1.041-mb/mm2 27.38-tops/w signed-int8 dynamic-logic-based adc-less sram compute-in-memory macro in 28nm with reconfigurable bitwise operation for ai and embedded applications. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)* (2022), vol. 65, pp. 188–190.

[103] Yang, M., Wang, Y., Xie, S., Lo, C.-P., Wang, M., Oruganti, S., Sehgal, R., and Kulkarni, J. P. Cilp: An arbitrary-bit precision all-digital compute-in-memory solver for integer linear programming problems. In *2024 IEEE Custom Integrated Circuits Conference (CICC)* (2024), pp. 1–2.

[104] Yu, C.-H., Kim, H.-E., Shin, S., Bong, K., Kim, H., Boo, Y., Bae, J., Kwon, M., Charfi, K., Kim, J., Kim, H., Shim, M., Ha, C., Shin, W., Yoon, J.-S., Chi, M., Lee, B., Choi, S., Kim, D., Woo, J., Yoon, S., Jo, H., Kim, H., Heo, H., Jin, Y.-J., Yu, J., Lee, J., Kim, H., Kang, M., Choi, S., Kim, S.-G., Choi, M., Oh, J., Kim, Y., Kim, H., Je, S., Ham, J., Yoon, J., Lee, J., Park, S., Park, Y., Lee, J., Hong, B., Ryu, J., Ko, H., Chung, K., Choi, J., Jung, S., Arthanto, Y. F., Kim, J., Cho, H., Jeong, H., Choi, S., Han, S., Park, J., Lee, K., Bae, S.-I., Bang, J., Lee, K.-J., Jang, Y., Park, J., Park, S., Park, J., Shin, H., Park, S., and Oh, J. 2.4 atomus: A 5nm 32tflops/128tops ml system-on-chip for latency critical applications. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)* (2024), vol. 67, pp. 42–44.

[105] Yu, W., Liang, F., He, X., Hatcher, W. G., Lu, C., Lin, J., and Yang, X. A survey on the edge computing for the internet of things. *IEEE Access 6* (2018), 6900–6919.

[106] Yuan, Y., Yang, Y., Wang, X., Li, X., Ma, C., Chen, Q., Tang, M., Wei, X., Hou, Z., Zhu, J., Wu, H., Ren, Q., Xing, G., Mak, P.-I., and Zhang, F. 34.6 a 28nm 72.12tflops/w hybrid-domain outer-product based floating-point sram computing-in-memory macro with logarithm bit-width residual adc. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)* (2024), vol. 67, pp. 576–578.

[107] Yue, J., He, C., Wang, Z., Cong, Z., He, Y., Zhou, M., Sun, W., Li, X., Dou, C., Zhang, F., Yang, H., Liu, Y., and Liu, M. A 28nm 16.9-300tops/w computing-in-memory processor supporting floating-point nn inference/training with intensive-cim sparse-digital architecture. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 1–3.

[108] Yue, J., Yuan, Z., Feng, X., He, Y., Zhang, Z., Si, X., Liu, R., Chang, M.-F., Li, X., Yang, H., and Liu, Y. 14.3 a 65nm computing-in-memory-based cnn processor with 2.9-to-35.8tops/w system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse. In *2020 IEEE International Solid-State Circuits Conference - (ISSCC)* (2020), pp. 234–236.

[109] Yue, J., Zhan, M., Wang, Z., He, Y., Li, Y., Yu, S., Sun, W., Jie, L., Dou, C., Li, X., Sun, N., Yang, H., Liu, M., and Liu, Y. A 5.6-89.9tops/w heterogeneous computing-in-memory soc with high-utilization producer-consumer architecture and high-frequency read-free cim macro. In *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)* (2023), pp. 1–2.

[110] Yue, Z., Wang, Y., Wang, H., Wang, Y., Guo, R., Tang, L., Liu, L., Wei, S., Hu, Y., and Yin, S. 7.7 cv-cim: A 28nm xor-derived similarity-aware computation-in-memory for cost-volume construction. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 1–3.

[111] Zabihi, M., Chowdhury, Z. I., Zhao, Z., Karpuzcu, U. R., Wang, J.-P., and Sapatnekar, S. S. In-memory processing on the spintronic cram: From hardware design to application mapping. *IEEE Transactions on Computers 68*, 8 (2019), 1159–1173.

[112] Zhang, B., Moon, S., and Seok, M. A 1-tflops/w, 28-nm deep neural network accelerator featuring online compression and decompression and

bf16 digital in-memory-computing hardware. In *2024 IEEE Custom Integrated Circuits Conference (CICC)* (2024), pp. 1–2.

[113] Zhang, H., Kang, W., Wang, L., Wang, K. L., and Zhao, W. Stateful reconfigurable logic via a single-voltage-gated spin hall-effect driven magnetic tunnel junction in a spintronic memory. *IEEE Transactions on Electron Devices 64*, 10 (2017), 4295–4301.

[114] Zhang, J., Wang, Z., and Verma, N. In-memory computation of a machine-learning classifier in a standard 6t sram array. *IEEE Journal of Solid-State Circuits 52*, 4 (2017), 915–924.

[115] Zhang, X., Sharma, V., Lu, Y., Jo, Y., and Kim, T. T.-H. A 400mhz 249.1tops/w 64kb fully-reconfigurable sram-based digital compute-in-memory macro for accelerating cnns. In *2023 IEEE Asian Solid-State Circuits Conference (A-SSCC)* (2023), pp. 1–3.

[116] Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[117] Zhang, Y., Liu, J., and Wang, H. Edgemednet: Lightweight and accurate u-net for medical image segmentation on edge devices. *IEEE Internet of Things Journal 10*, 4 (2023), 1234–1245.

[118] Zhong, Z., Wei, Y., Go, L. C., and Gu, J. 33.2 a sub-1j/class headset-integrated mind imagery and control soc for vr/mr applications with teacher-student cnn and general-purpose instruction set architecture. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)* (2024), vol. 67, pp. 544–546.

[119] Zhou, Z., and Zhang, G. The fast simulation model of sram. In *2006 8th International Conference on Solid-State and Integrated Circuit Technology Proceedings* (2006), pp. 1333–1335.

[120] Zhu, H., Jiao, B., Zhang, J., Jia, X., Wang, Y., Guan, T., Wang, S., Niu, D., Zheng, H., Chen, C., Wang, M., Zhang, L., Zeng, X., Liu, Q., Xie, Y., and Liu, M. Comb-mcm: Computing-on-memory-boundary nn processor with bipolar bitwise sparsity optimization for scalable multi-chiplet-module edge machine learning. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)* (2022), vol. 65, pp. 1–3.

Biography

Weijie Jiang was born in Hubei, China, in 1996. He pursued his undergraduate studies in microelectronics at Shanghai Jiao Tong University from 2014 to 2018, where he obtained his bachelor's degree. In 2018, he moved to Belgium to continue his education in electronic engineering at KU Leuven (Katholieke Universiteit Leuven), earning his master's degree in 2020. He subsequently began his PhD at KU Leuven under the supervision of Prof. Wim Dehaene, focusing on SRAM and in-memory compute circuit design.

List of publications

- W. Jiang, P. Houshmand, M. Verhelst and W. Dehaene, "A 16nm 128kB high-density fully digital In Memory Compute macro with reverse SRAM pre-charge achieving 0.36TOPs/mm2, 256kB/mm2 and 23. 8TOPs/W," ESSCIRC 2023- IEEE 49th European Solid State Circuits Conference (ESSCIRC), Lisbon, Portugal, 2023, pp. 409-412, doi: 10.1109/ESSCIRC59616.2023.10268774.
- W. Jiang, C. Caron, P. Avasare, M. Pauwels, M. Verhelst and W. Dehaene, "HUNBN, a 1.77MB Digital In-Memory-Compute SoC for edge applications achieving 126 TOPs/W (4b) at macro level and 24 TOPs/W at SoC level.," 2024 IEEE European Solid-State Electronics Research Conference (ESSERC), Bruges, Belgium, 2024, pp. 137-140, doi: 10.1109/ESSERC62670.2024.10719477.
- W. Jiang, C. Caron, P. Avasare, M. Pauwels, M. Verhelst and W. Dehaene, "HUNBN, a 16nm digital in-memory-compute SoC for edge CNN application achieving 24 TOPs/W (4b) at system level. ", in IEEE Journal of Solid-State Circuits. (Invited)
- W. Jiang, X. Zheng, J. Sun, G. Gielen, M. Verhelst, W. Dehaene, "CIPL: A Fast and Low-Power Level Shifter for Wide-Range Voltage Conversion", 2025 IEEE International Symposium on Circuits and Systems. (Accepted)

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
MICAS
Kasteelpark Arenberg 10, B-3001 Belgium
B-3001 Leuven
weijie.jiang@kuleuven.be
https://micas.esat.kuleuven.be/