Deep In-Memory Computation Based Multiply and Accumulate Architecture For Machine Learning Applications

Thesis submitted by

Hemant Kumar
2021EEN2015

under the guidance of

Dr. Kaushik Saha,
Indian Institute of Technology, Delhi

in partial fulfilment of the requirements
for the award of the degree of

Master of Technology

Department of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY DELHI

May 2023

CERTIFICATE

This is to certify that the M.Tech project titled **Deep In-Memory Computation Based Multiply and Accumulate Architecture For Machine Learning Applications**, submitted by **Hemant Kumar (2021EEN2015)**, to the Indian Institute of Technology, Delhi, for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by him under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr.Kaushik Saha
Electrical Engineering Department
IIT-Delhi ,110016

Place: New Delhi

Date: 15th May 2023

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to IIT Delhi and the Electrical Engineering department for granting me this invaluable opportunity to work on this project. Their visionary objective of promoting curriculum-oriented projects has provided me with an extraordinary platform. I extend my heartfelt thanks to my project guide, Dr. Kaushik Saha, for his invaluable guidance and support throughout the design and development phases. Without his expertise and encouragement, I would not have gained a comprehensive understanding of the key concepts and fundamentals.

I would also like to acknowledge the unwavering support of all the faculty members in the department. Their guidance and excellence in their respective fields have been instrumental in the success of my project. Additionally, I am grateful for the provision of facilities, laboratories, instrumental support, and necessary resources.

Lastly, I express my immense gratitude to Dr. Kaushik Saha for his continuous support and encouragement in pursuing projects aligned with my field of interest. His guidance has been pivotal in making this endeavor a reality.

ABSTRACT

Traditional computing architectures face a persistent challenge in achieving optimal performance due to the inherent speed disparity between processors and memory. This discrepancy leads to a bottleneck where the processor is often starved of data, as memory struggles to deliver information at high rates. To overcome this limitation, a promising solution has emerged: the diffusion of boundaries between memory and processors, with a focus on processing inside memory.

This thesis investigates the development of a variation-tolerant in-memory computing architecture tailored for low-power neuromorphic systems. Specifically, it proposes a novel architecture to perform multiply and accumulate. The architecture harnesses the power of highly parallel computing facilitated by Deep in-memory (DIMA) processing.

However, the analog nature of DIMA architectures poses challenges in terms of sensitivity to variations in process, voltage, and temperature, particularly in scenarios with low voltage conditions where bitline voltages exhibit diminished swings. To address these issues, the thesis presents a novel architecture for computing the multiply and accumulate (MAC) operations involved in SVM cost function minimization. To evaluate its effectiveness, the architecture is thoroughly characterized under varying supply voltages and noise variance.

Moreover, the proposed architecture prioritizes low power consumption and area efficiency, making it suitable for integration into neuromorphic computing systems. By seamlessly integrating memory and processing capabilities, the architecture strives to alleviate the speed bottleneck and enhance system performance. The thesis provides a proof of concept for this architecture and offers insights into its behavior under different operating conditions. It also contributes in terms of novelity in Analog to Digital converter architecture and the Scaling network.

Ultimately, this research contributes to the advancement of low-power, area-aware neuromorphic computing systems by enabling efficient and variation-tolerant in-memory processing. By addressing the limitations of traditional computing architectures, this thesis paves the way for improved performance and energy efficiency in data-intensive applications.

Contents

List of Tables

List of Figures

ABBREVIATIONS

IITD	Indian Institute of Technology, Delhi
ADC	Analog to digital converter
SAR	Successive Approximation Register
BCA	Bit Cell Array

Chapter 1

INTRODUCTION

There has been an increasing interest in the applications of machine learning and artificial intelligence. Applications range from education, healthcare to smart appliances. Algorithms employed in these applications are generally computationally heavy and have to work under low-power conditions (for example IoT devices). Other problems faced are large data set and frequent training requirement.

Traditional computing works with the separation of memory and processor, where there is constant to-and-fro motion of data between the two. However, although with the advancement of technology processors have become fast, memory still lags behind in speed. This creates a speed bottleneck wherein the processor is starved of data as the memory is not capable of delivering at high rates.

One of the solutions that has emerged is the diffusion of boundary between the memory and processor, with the concept of processing inside the memory. This newer paradigm of architectures has been classified as Near memory architectures, Logic in memory architectures or the Deep in-memory architectures. In-memory computing architectures are potential solution for the speed bottleneck problem faced by the current architectures. They, when, incorporated into AI/ML applications, provide avenues for low power, low latency computing. This in turn will make portable solutions possible to the challenging problems, opening avenues for improving human quality of life for example IoT in healthcare.

In memory computing inherently suffers from the limitation of low-compute SNR. However, when incorporated into ML algorithms, which inherently are robust to errors, there is a comparable performance of systems as is there in the Von Neuman architecture but at much less power consumption and delay. The averaging effect involved in the multiply and accumulate operation also lowers the effect of statistical noise at the output of the in-memory computation-based architecture. A recent publication on SVM implements a variation tolerant in-memory machine learning classifier via on chip classification [1]. This architecture exploits highly parallel computing which is possible with Deep in-memory (DIMA) processing. However, since DIMA's are inherently analog, they are sensitive to the process, voltage, and temperature variations especially under low voltage conditions where the bitline voltages have lower swings. Paper develops a solution with on-chip training, which according to the authors, adds to the robustness of the design. Chip-specific training weights are better suited to take PVT induced variations into account. Thesis discusses a novel architecture to compute the multiply and accumulate (MAC) operation involved in the SVM cost function

minimization. It presents a proof of concept for the architecture and it's characterization under varying supply voltages and noise variance. It provides an architecture for incorporation into low power and an area-aware neuromorphic computing.

1.1 Basics and Need of a Deep In Memory Computation

Deep In-Memory Computation (DIMC) is an emerging paradigm in computer architecture that aims to address the performance and energy efficiency challenges associated with traditional von Neumann architectures. In DIMC, the boundary between memory and computation is blurred by performing certain computations directly within the memory units, such as RAM or non-volatile memory.

The need for DIMC arises from the widening performance gap between processors and memory in conventional computing systems. While processors have become increasingly fast and powerful, the speed of memory access has not kept pace with these advancements. This leads to a bottleneck in data transfer between the processor and memory, known as the "memory wall" problem. As a result, the processor is often starved of data, leading to reduced overall system performance.

DIMC offers several advantages to overcome these limitations:

1. **Reduced Data Movement:** By performing computations within the memory itself, DIMC eliminates or reduces the need for data movement between the processor and memory. This reduces the data transfer overhead, alleviating the memory wall problem and improving overall system performance.
2. **Improved Energy Efficiency:** DIMC can significantly reduce the energy consumption associated with data movement. Since data is processed directly within the memory, there is a reduction in the amount of data transferred over long distances, resulting in lower energy consumption.
3. **Increased Parallelism:** DIMC enables highly parallel processing by leveraging the massive parallelism inherent in memory arrays. This allows for efficient execution of data-intensive tasks, such as machine learning algorithms and data analytics, by exploiting the inherent parallelism of memory.
4. **Enhanced Scalability:** DIMC offers potential scalability advantages as memory arrays can be easily expanded or stacked, providing a scalable architecture for accommodating large-scale data-intensive applications.

Overall, the basics and need for Deep In-Memory Computation stem from the desire to overcome the performance and energy efficiency challenges posed by the memory wall problem in traditional computing systems. DIMC provides a promising approach to bridge the processor-memory performance gap and unlock new opportunities for efficient and high-performance computing applications.

1.2 Features of a Deep In Memory Architecture:

We can define an ideal Deep In-Memory Architecture (DIMA) by specifying the following properties (1):

1. **Use of a standard Bit-Cell Architecture (BCA):** The idealized DIMA relies on a conventional memory BCA, which maintains memory density while ensuring efficient storage and retrieval of data. By leveraging a standard BCA, the idealized DIMA capitalizes on established memory design principles and technologies.

2. **Row parallelism:** In the idealized DIMA, all rows within the BCA are simultaneously activated during computational operations. This approach allows for the exploitation of extensive parallelism, enabling multiple calculations to be performed simultaneously. By activating all rows concurrently, the idealized DIMA maximizes computational throughput and accelerates data processing.

3. **Bit-Line (BL) computations:** The idealized DIMA goes beyond conventional memory architectures by incorporating analog computations on the BLs. This unique feature enables valuable analog operations to be executed directly within the memory units. By harnessing the inherent parallelism of memory arrays, the idealized DIMA leverages the BLs to perform computations, significantly reducing the need for frequent data transfers between the processor and memory. This reduction in data movement leads to improved efficiency and performance.

4. **Delayed decision:** To enhance processing capabilities, the idealized DIMA implements a strategy of delayed decision-making. This involves performing analog computations on the BL outputs, allowing for the postponement of hard decisions. Digitization, comparison, and sense amplification are examples of analog operations carried out in a massively parallel fashion. The design of column-pitch-matched analog circuits is crucial to enable these computations, ensuring coordinated and efficient processing within the DIMA.

By incorporating these properties, the idealized DIMA aims to maximize memory density, exploit parallelism, and delay decision-making processes. This architectural concept paves the way for efficient and high-performance in-memory computations, reducing the data movement overhead and enabling complex analog operations directly within the memory arrays.

1.3 Variations of In Memory Architectures:

One of the current trends in AI compute architectures involves a notable departure from the conventional Von Neumann architecture, as depicted in Figure 1.1(a). This traditional architecture, although still widely used, strives to minimize memory accesses through techniques such as efficient data flow, data reuse, and computation reduction. However, it maintains the fundamental separation between the memory and the processor.

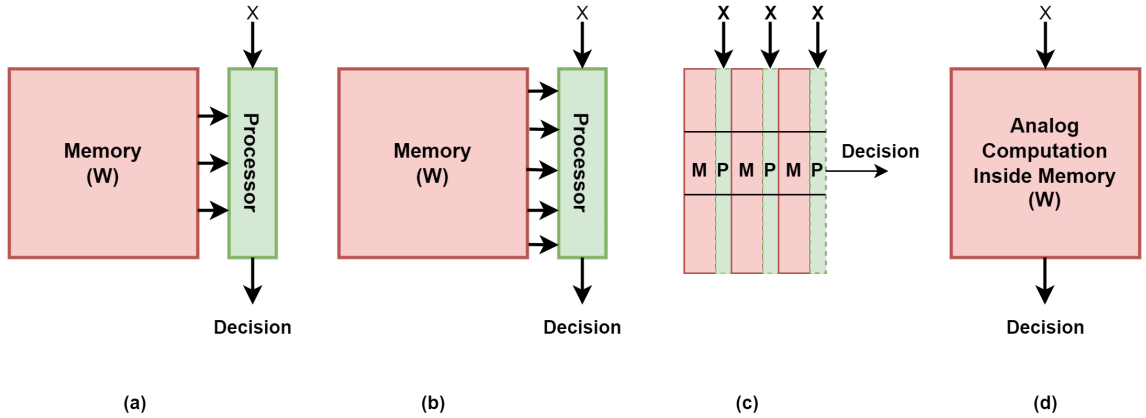


Figure 1.1: Alternate architectures: (a)Von Neumann b) Near-memory c) Logic In Memory (d) Deep In Memory, where X is an input vector and W is the weight vector stored in SRAM. (1)

In contrast, there has been a growing interest in near-memory architectures, as illustrated in Figure 1.1(b). These architectures aim to address the challenges associated with memory access costs by distributing computations around memory banks or employing advanced 3-D technologies that physically stack memory on top of CMOS dies. The objective is to maximize the proximity between computation and memory, thereby reducing data movement and enhancing overall system performance.

Another promising approach is the logic-in-memory (LIM) architecture, as shown in Figure 1.1(c). LIM architectures utilize memory bitcells (BCs) that incorporate embedded digital logic, enabling computations to be performed directly within the memory. However, this approach necessitates specialized BCs, such as 16T in CMOS. While LIM architectures offer the advantage of in-memory computations, they introduce additional design complexities and may not fully leverage the optimized design processes employed for conventional BC arrays (BCAs).

Deep in-memory architectures (DIMAs), depicted in Figure 1(d), take a unique approach by integrating analog computations in close proximity to the memory BCA. This integration directly addresses the limitations of the memory-processor interface, allowing for significant reductions in the energy-delay product (EDP) compared to equivalent von Neumann architectures. DIMA has demonstrated remarkable achievements, with multiple integrated circuit (IC) prototypes showcasing more than 100× reduction in EDP for decision-making tasks.

In summary, the emerging near-memory, LIM, and DIMA architectures represent a paradigm shift that aims to bring computation closer to memory, transcending the limitations of traditional von Neumann architectures. These advancements hold considerable promise for improving energy efficiency, reducing data movement overhead, and achieving

enhanced overall performance in AI computing.

1.4 Motivation

The Von Neumann architecture, while widely used and highly prevalent in computing systems, have certain limitations

1. **Memory Bottleneck:** In the Von Neumann architecture, there is a separation between the memory and the processor, leading to a significant memory bottleneck. The constant data movement between the memory and processor can cause delays and limit overall performance.
2. **Limited Parallelism:** Von Neumann architecture is inherently sequential, which limits parallel processing. Instructions are executed one after the other, reducing the potential for parallel execution and impacting performance.
3. **Instruction and Data Fetching Overhead:** Von Neumann architecture requires fetching instructions and data from memory, introducing additional overhead. Sequential fetching can result in inefficiencies and slower processing, especially with large data or complex algorithms.
4. **Memory Access Conflict:** Sharing the same bus for data and instructions, Von Neumann systems may face conflicts when multiple processes attempt to access memory simultaneously. This contention can lead to performance degradation.
5. **Power Consumption:** The constant data transfer between processor and memory in Von Neumann architecture increases power consumption. Energy is consumed in moving data, limiting energy efficiency, especially in power-sensitive applications.
6. **Limited Scalability:** Von Neumann architecture may struggle to scale performance as applications and data complexity grow. The sequential instruction execution and memory bottleneck can hinder scalability, making it challenging to handle demanding computational tasks effectively.

To address the aforementioned limitations, novel compute architectures have emerged as viable alternatives. Among these, Deep In-Memory Computation stands out as a promising solution. It offers numerous advantages over the conventional Von Neumann architecture, as previously elucidated. To facilitate comprehension, the key benefits are succinctly summarized below.

1. **Improved Memory Access:** One of the key advantages of Deep In-Memory Architecture (DIMA) is its ability to bring computation closer to memory. By doing so, DIMA reduces the data movement overhead that is typically associated with traditional architectures. This architectural design allows for direct processing within the memory units, minimizing the need for frequent data transfers between the processor and memory. As a result, DIMA significantly enhances memory access efficiency, leading to improved overall system performance.

2. **Parallelism and Speed:** DIMA leverages the inherent parallelism of memory units to enable highly parallel processing. By performing computations directly within the memory arrays, DIMA can execute multiple operations simultaneously, leading to enhanced parallelism. Moreover, the close proximity of computation and memory in DIMA reduces latency, resulting in faster processing speeds. This combination of parallelism and speed contributes to the overall efficiency and effectiveness of DIMA.

3. **Energy Efficiency:** Another notable advantage of DIMA is its improved energy efficiency compared to traditional architectures. By minimizing data movement and reducing the need for frequent memory access, DIMA significantly reduces energy consumption. The ability to perform computations within the memory itself helps mitigate power-hungry data transfers and enhances overall energy efficiency. This energy efficiency is particularly beneficial in power-constrained environments and contributes to the sustainability of DIMA-based systems.

4. **Reduced Data Transfer:** DIMA reduces the data transfer overhead between the processor and memory. By performing computations directly within the memory arrays, DIMA minimizes the need for extensive data movement. This reduction in data transfer requirements alleviates bandwidth constraints and mitigates the latency issues associated with data movement. As a result, DIMA allows for more efficient utilization of system resources and contributes to improved overall performance.

5. **Enhanced Performance for Data-Intensive Applications:** DIMA's proximity of computation to memory is particularly advantageous for data-intensive applications, such as artificial intelligence and machine learning. By enabling in-memory processing of large datasets, DIMA minimizes data movement, accelerates computations, and improves overall performance for these demanding workloads. This capability is especially beneficial in scenarios where processing large volumes of data in a timely manner is critical.

6. **Scalability:** DIMA provides scalability for both memory capacity and computational capability. As the size of memory arrays increases, the computational power of DIMA scales proportionally. This scalability is crucial for handling complex and large-scale computational tasks efficiently. Additionally, DIMA's architectural design allows for flexible expansion of memory capacity, accommodating the growing demands of data-intensive applications and enabling future scalability.

In memory computing inherently suffers from the limitation of low-compute SNR. However, when incorporated into ML algorithms, which inherently are robust to errors, there is a comparable performance of systems as is there in the Von Neuman architecture but at much less power consumption and delay. The averaging effect involved in the multiply and accumulate operation also lowers the effect of statistical noise at the output of the in-memory computation-based architecture.

A recent publication (7) on SVM implements a variation tolerant in-memory machine learning classifier via on chip classification . This architecture exploits highly parallel computing which is possible with Deep in-memory (DIMA) processing. However, since DIMA's are inherently analog, they are sensitive to the process, voltage, and temperature variations

especially under low voltage conditions where the bitline voltages have lower swings. Paper develops a solution with on-chip training, which according to the authors, adds to the robustness of the design. Chip-specific training weights are better suited to take PVT induced variations into account.

Thesis discusses a novel architecture to compute the multiply and accumulate (MAC) operation involved in the SVM cost function minimization. It presents a proof of concept for the architecture and it's characterization under varying supply voltages and noise variance. It provides an architecture for incorporation into low power and an area-aware neuromorphic computing. Along with this, thesis also discusses certain novel architectures to perform scaling and a modified Successive Approximation Register based ADC.

Chapter 2

Literature review

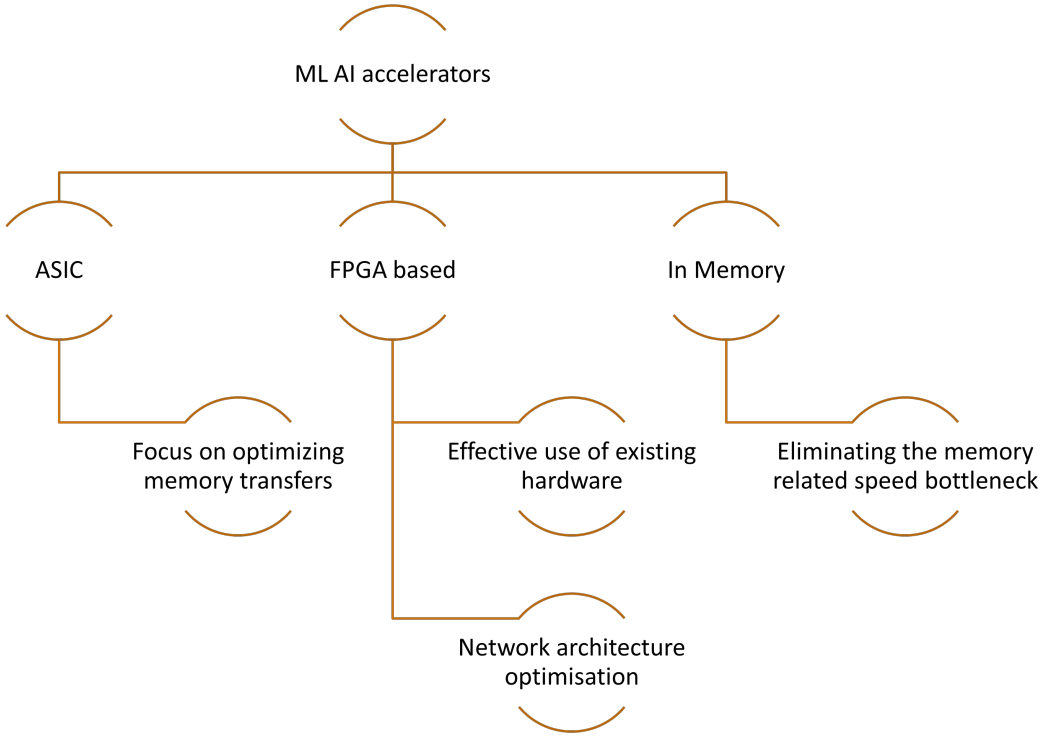


Figure 2.1: Classification of ML-AI Hardware Accelerators (1) (2) (3) (4)

2.1 ASIC based Accelerators

Application Specific Integrated Circuits for the purpose of AI/ML accelerators has been an active area of research. Major focus has been to optimise the memory transfers.The recent trend in machine learning involves the use of increasingly large neural networks, with billions of parameters, to achieve state-of-the-art accuracy on challenging tasks. However, these large networks pose hardware limitations, particularly regarding memory access and computational efficiency. From a machine learning perspective, a neural network with 1 billion parameters may seem substantial. However, when considering the hardware aspect, it is manageable. Assuming each parameter requires 64 bits, the total storage needed is only 8 GB, and there are indications that even fewer bits could suffice. While 8 GB can be too large for a single chip, a viable solution is to construct a dedicated machine-learning computer consisting of multiple chips. Each chip would house specialized logic and sufficient RAM

to store the entire neural network, eliminating the reliance on main memory. By tightly interconnecting these chips through a dedicated mesh, it becomes feasible to implement even the largest existing deep neural networks (DNNs) with improved performance. This approach offers notable advantages such as reduced energy consumption and area costs compared to traditional CPUs or GPUs. It has been used in (8). Researchers have used 16-bit weights for CNNs and DNNs and have stored them on chip. Authors report a remarkable speedup of 450.65x and an average energy reduction of 150.31x compared to GPU's.

Another work on similar idea has been done in (9). Authors present PuDianNao, an ML accelerator which can carry out operations of seven prominant ML techniques, including SVM, K-nearest neighbours, regression etc. Their focus have been to enhance ML techniques efficiency by working on locality properties, improving access to computational tasks and primitives using on-chip storage. There findings underscore the efficiency and versatility of PuDianNao as an ML accelerator. By supporting multiple ML techniques and optimizing their computational characteristics, PuDianNao exhibits superior performance and energy efficiency, surpassing conventional GPU architectures.

ShiDianNao (10), a hardware accelerator, is designed specifically for real time detection and recognition algorithms. It exploits a property of CNNs which is weight sharing. It significantly reduces the memory footprint of the neural network. This advantageous characteristic allows for the complete mapping of a CNN onto a Static Random-Access Memory (SRAM), eliminating the need for accessing Dynamic Random-Access Memory (DRAM) for weight data. Furthermore, by strategically placing the CNN accelerator in close proximity to the image sensor, the remaining DRAM accesses too can be eliminated, including those required for inputs and outputs.

2.2 FPGA based Accelerators

The versatility, adaptability, and rapid performance of FPGAs have propelled them into the spotlight, making them an ideal choice for developing accelerators. ML applications demand extensive computational power and intricate processing, along with the need for minimal latency and impressive throughput. FPGAs excel in meeting these requirements by leveraging genuine parallelism, unlike CPU-only implementations. On the other hand, ASICs offer distinct advantages in having less size, increased energy efficiency, and more processing speed. ASICs can be meticulously optimized for compactness, and can attain remarkable speeds compared to FPGAs. But, they have got an disadvantage when it comes to reconiguration after they have been fabricated. This indeed is a hallmark of FPGAs. Ultimately, the decision to opt for either an FPGA or ASIC hinges on the application's unique demands and the user's priorities concerning reconfigurability and design constraints.

In their work, Qiu et al. (11) introduced a novel FPGA-based accelerator for Con-

volutional Neural Networks (CNNs), with a specific focus on image classification tasks in embedded platforms. The design primarily targets the CONV and FC layers of the CNN, which are critical components and are heavy in terms of computation and memory accesses.To efficiently handle these expensive operations, the researchers employed singular value decomposition techniques to optimize the weight matrix of the fully connected layer. This approach effectively reduced the memory footprint and resource consumption. Additionally, they incorporated a specialized convolver design module to enhance utilisation of memory by decreasing it's accesses and also minimized the need of bandwidth.

In order to address memory-bandwidth limitations, the architecture was devised to have dynamic precision capability for the data quantization. This technique enabled efficient storage and processing of data by adaptively adjusting the precision levels based on specific requirements. Overall, the proposed FPGA-based CNN accelerator by Qiu et al. leverages innovative strategies such as SVD weight decomposition, specialized convolver design, and dynamic precision data quantization to optimize performance, memory utilization, and bandwidth requirements in embedded platforms.

Apart from maximizing the utilization of current hardware components, researchers have dedicated their efforts to optimizing deep neural networks (DNNs) through the exploration of network architecture modifications. The primary goal of these optimizations is to reduce the size of the models while simultaneously improving hardware efficiency. To accomplish this, one approach involves the utilization of model compression and pruning techniques. These techniques specifically target large and intricate networks by eliminating redundant neurons, compressing weight values into binary representations, or reducing the precision of the weights. Through the application of these optimization methods, the overall size and complexity of the network can be significantly reduced, leading to potential improvements in both performance and power efficiency.

Gao et al. [33] introduced an innovative pruning technique which exploited the entropy concepts for deep neural networks (DNNs) with aim of reducing network complexity while adhering to performance and latency constraints. The authors focused on addressing the challenges associated with latency, complexity of the network, and enabling use of DNN on edge devices with limited resources. Their proposed approach involves integrating an pruning regularizer based on the entropy ideas with existing pruning algorithms. The key idea is to reduce the search space of the decoder by decreasing the information entropy of the DNN and hence reducing latency. This technique is particularly beneficial for use cases which require a, DNN and decoder, configuration.

2.3 In Memory based Accelerators

In recent years, there has been a growing interest in deep in-memory computing, also known as processing-in-memory (PIM). This approach aims to address the challenges associated with data transfer between ALU and memory components. While the idea of computing in-memory has been around for some time, practical implementation and exploration of its benefits have gained traction more recently. The key advantage of in-memory processing is its ability to alleviate bandwidth limitations, reduce latency, and minimize energy consumption caused by data transfers between memory and processors. By enabling efficient and parallel computations, PIM-based hardware accelerators have garnered significant attention, particularly for demanding applications like deep neural networks (DNNs). Researchers are actively investigating the potential of PIM to enhance the performance and energy utilisation of DNN applications. The exploration of PIM-based hardware accelerators represents a promising avenue for advancing high-performance computing and unlocking new possibilities for DNN applications. The ongoing research and development in this field highlight the potential transformative impact of in-memory computing in the realm of intense, efficient computing.

The utilization of SRAM-based processing-in-memory, for instance in IMAC (12) and in another work related to XNOR-SRAM (13), has gained attention for accelerating convolutional neural networks (CNNs). IMAC shares similarities with accelerators using ReRAM architectures like ISAAC, but faces challenges related to SRAM leakage, resulting in lower efficiency and performance. Whereas, XNOR -SRAM decreases complexity of convolutions by reducing weights to binary and ternary, leading to significantly higher energy efficiency compared to ISAAC, a ReRAM-based accelerator.

Another approach, DRISA, proposed by Li et al., takes advantage of DRAM technology. DRISA employs computing using bitlines. In such configuration several logic gates are placed on each bitline, enabling parallel computations at the bit level. Unlike XNOR-SRAM (13), DRISA (14) supports more complex operations with improved precision by involving and embedding several layers of logic gates. However, this approach requires significant modifications to the DRAM memory architecture and results in a considerably larger chip size. Despite these challenges, DRISA demonstrates impressive performance, achieving a throughput of 147 frame/s for eight-bit binary weighted AlexNet set. These diverse SRAM-based and DRAM-based PIM architectures contribute to the advancement of CNN acceleration, offering options to optimize energy efficiency and throughput in deep learning applications.

Additional PIM architectures have emerged for CNN/DNN acceleration, including XNOR-POP (15) and SCOPE (16). These architectures employ bitline-based computing to improve performance and energy efficiency.

DrAcc (17), developed by Deng et al., focuses on ternarized weighted CNN inferences and incorporates carry look-ahead addition. Its throughput is lower than DRISA, DrAcc achieves a remarkable 49× improvement in energy efficiency with only a minimal 2% of extra chip area. Another case is of a PIM architecture which exploits DRAM. SCOPE (16) is such an accelerator which uses stochastic approx. computing to accelerate Convolutional Neural Network operations. Despite its larger chip area, approximately four times that of traditional DRAM chips, SCOPE achieves an impressive throughput of above 2500 frames/s for AlexNet dataset. XNOR-POP (15) is a accelerator which uses binary weighted CNN. It employs bitwise XNOR operations, similar to XNOR-SRAM, resulting in superior area and energy efficiency.

These PIM architectures provide diverse design options, offering opportunities to optimize both performance and energy efficiency in CNN/DNN acceleration applications.

2.4 Analog to Digital Converters

Analog-to-digital converters (ADCs) are essential components in modern electronic systems, responsible for converting continuous analog signals into discrete digital representations for further processing. Among the various ADC architectures available, two commonly used ones are the Successive Approximation Register (SAR) ADC and the Flash ADC.

The SAR ADC is renowned for its versatility, moderate speed, and high accuracy. It operates through an iterative process that approximates the analog input voltage by comparing it with a reference voltage using a binary search algorithm. Starting with the most significant bit (MSB) of the digital output, the SAR ADC successively determines whether the input voltage is higher or lower than the comparator's threshold voltage. Based on the comparison result, it adjusts the corresponding bit in the digital code until convergence is achieved, resulting in an accurate digital representation of the analog signal. SAR ADCs are favored for their low power consumption, suitability for applications with moderate speed and resolution requirements, and their ability to handle a wide range of input signals.

On the other hand, Flash ADCs are known for their exceptional speed but at the cost of increased complexity and power consumption. A Flash ADC employs a parallel architecture, consisting of a resistor ladder network and a set of comparators. The resistor ladder network divides the reference voltage into multiple equal voltage levels, while the comparators simultaneously compare the analog input voltage against these reference levels. Each comparator determines which voltage level the input voltage matches or exceeds, generating a unique digital output code corresponding to that particular voltage level. Flash ADCs offer rapid conversion rates due to the parallel comparison of the input signal, making them suitable for applications demanding high-speed signal acquisition. However, the main drawbacks of Flash ADCs are their limited resolution and the requirement for a large number

of comparators, resulting in increased power consumption, complex circuitry, and higher manufacturing costs.

When selecting an ADC architecture, one has to consider the specific requirements of the application, such as the desired speed, resolution, power consumption, and cost. SAR ADCs are often preferred for applications that prioritize power efficiency, moderate speed, and high accuracy. On the other hand, Flash ADCs find their niche in applications demanding fast conversion rates, where resolution and power consumption take a backseat. Ultimately, the choice between SAR ADC and Flash ADC depends on striking the right balance between the performance requirements and the trade-offs associated with power consumption, complexity, and cost.

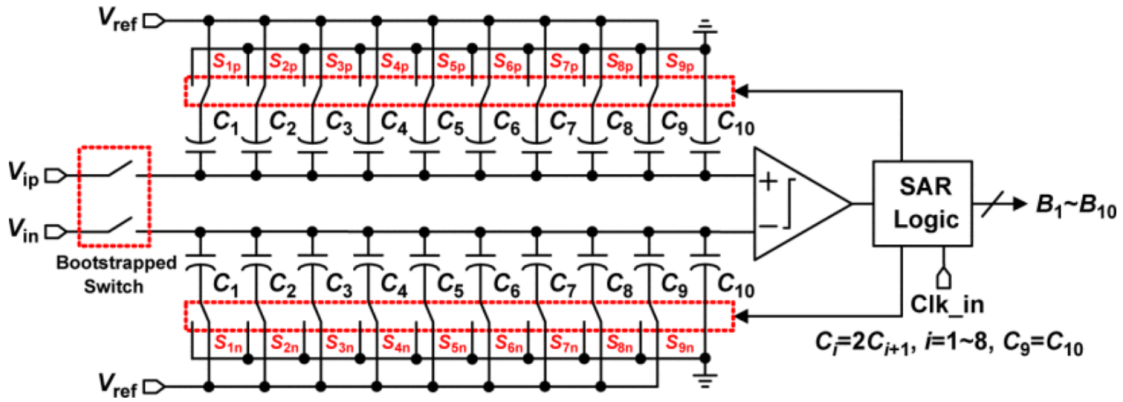


Figure 2.2: SAR ADC architecture for differential signals, exploiting monotonic switching. Adapted from (5).

This work (5) introduces a novel approach to designing a low-power 10-bit, 50-MS/s successive approximation register (SAR) analog-to-digital converter (ADC). The proposed ADC utilizes a unique capacitor switching procedure that ensures monotonically increasing or decreasing voltage levels. This switching procedure offers significant advantages over conventional methods, resulting in a remarkable reduction of approximately 81% in average switching energy and 50% in total capacitance. In addition to the improved switching procedure, ADC in this work employs an enhanced comparator to address the signal-dependent offset caused by input common-mode voltage variation. This enhancement effectively mitigates the impact of voltage fluctuations on the ADC's performance, leading to improved accuracy and reliability in the conversion process.

Other architectures like in (18), where the authors target the settling issues faced by the capacitive DAC present in ADC. This issue is prominant at higher sampling rates like 100MS/s and at advanced technology nodes because there is an increased interconnect line impedance which in turn impedes the charge transfer. Researchers in (19) have used incorporated both SAR and digital slope architecture topology to enable operation at higher resolutions. SAR part of the architecture is to provide a coarse output and it is further

resolved by digital slope ADC to produce final high resolution. For the purpose of this thesis a modified version of (5) is used and will be detailed in the subsequent chapters.

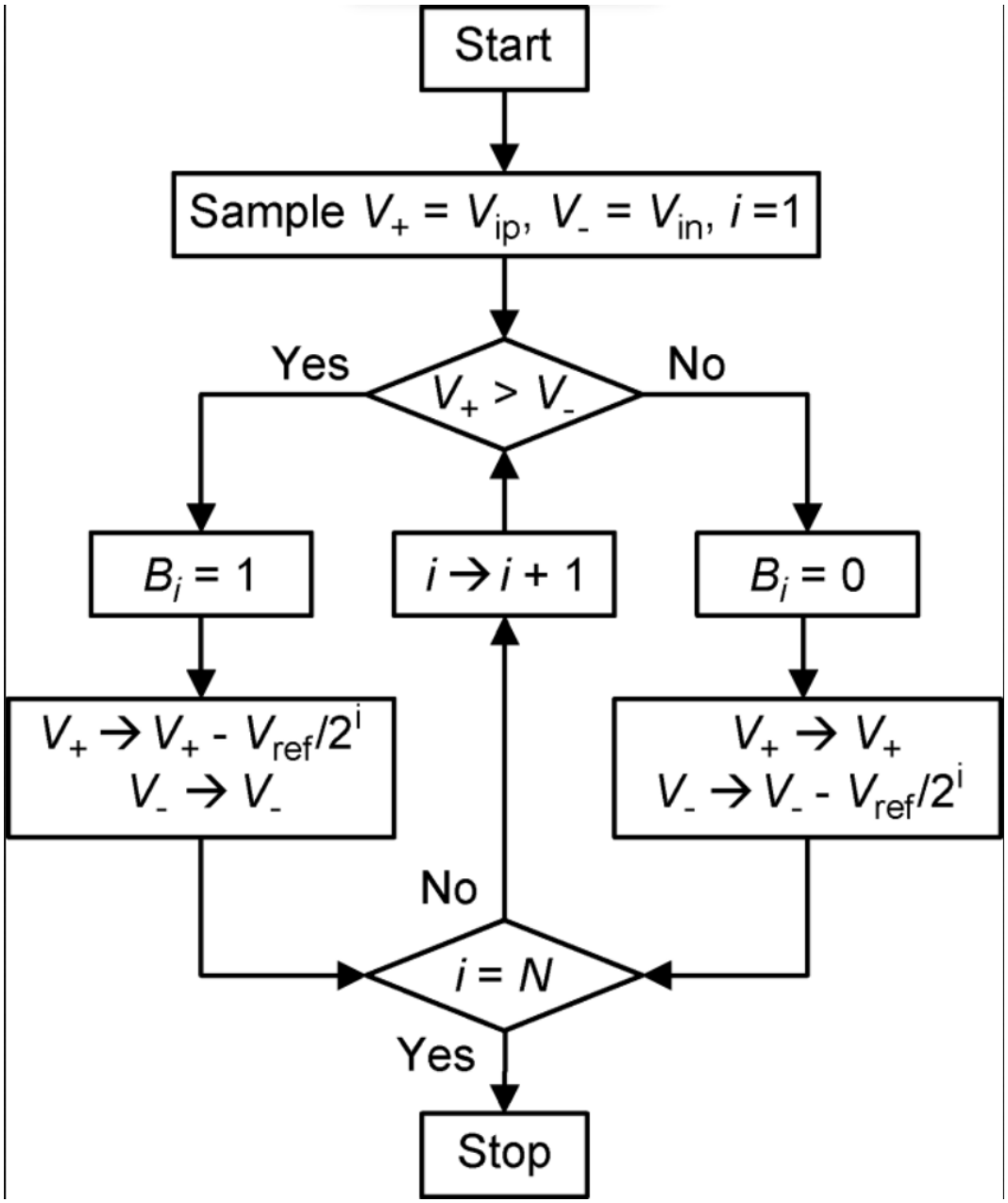


Figure 2.3: Flow chart for the monotonic switching. Adapted from (5).

Chapter 3

Design and Methodology

An AI-ML accelerator, also known as an Artificial Intelligence-Machine Learning accelerator, is a specialized hardware component or system designed to accelerate the performance of artificial intelligence and machine learning tasks. It is specifically optimized to execute the computationally intensive operations involved in AI and ML algorithms, such as matrix multiplication, vector operations, and convolutional operations.

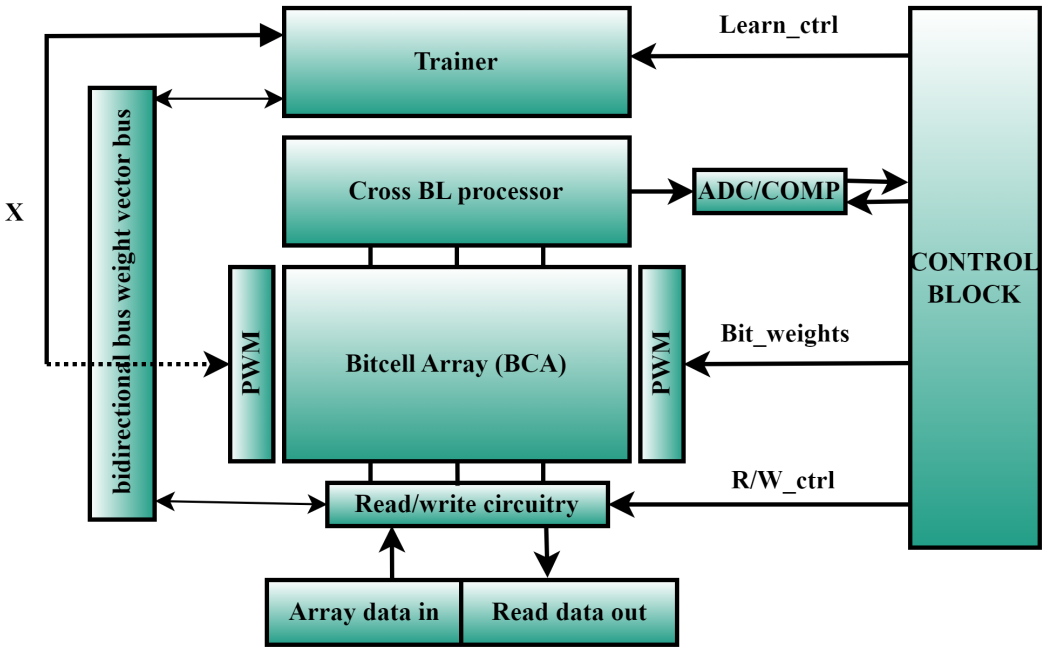


Figure 3.1: Illustration of DIMA based SVM accelerator architecture

Different Sub-blocks in Figure 3.1 :

1. **Bitcell Array (BCA):** An array of modified bit-cells, storing the multi-bit weight vector.
2. **PWM:** PWM drivers driving the lines with binary-duration pulse widths. They assign binary weights to the input and weight vectors.
3. **Read/write circuitry:** Used to store(read) multi-bit weights to(from) bit cells
4. **Cross BL processor:** Circuitry to compute final addition of voltages accumulated across bit lines.

5. **Control Block:** Used to generate control signals for the accelerator.
6. **Trainer:** Used to perform on-chip training of weights.
7. **ADC/COMP:** Used to digitize the accumulated analog voltage and perform decision on a sample under test.
8. **X:** Multi-bit input vector is fed to PWM drivers.

The dot product, $W^T X$ is the main computation which is required for the computing the inference, in a number of machine learning algorithms. Following section illustrates this with an example of SVM algorithm.

3.1 Brief Overview of a SVM

Support Vector Machine classifier involves finding of a maximum margin hyperplane which separates the two classes. It predicts the class label $y_k = 1$ according to the expression,

$$\begin{aligned} W^T X_k + b > 0 \text{ if } \hat{y_k} = +1 \\ W^T X_k + b < 0 \text{ if } \hat{y_k} = -1 \end{aligned} \tag{3.1}$$

$\hat{y_k}$ is the prediction, $X_k = [x_{(0,k)}, x_{(1,k)}, \ldots\ldots x_{(L-1,k)}]$ is a L- dimensional input vector, weight vector $W^T = [w_{(0,)} w_{(1,)} \ldots\ldots w_{(L-1)}]^T$ and b is the scalar bias. Here, $x_{(i,k)}$ and w_j are multi-bit scalars. SVM cost function is as follows

$$Q(W, X_k) = \frac{\lambda}{2}(||W||^2 + b^2) + \{1 - y_k(W^T X_k + b)\}_+ \tag{3.2}$$

λ is the regularization factor, y_k is the true label for training data input vector X_k, and the expression $\{x\}_+ = max\{0, x\}$. Stochastic gradient descent (SGD) algorithm is used to minimize the cost function and it minimizes the averaged $Q(W, X_k)$ (over the training set) by updating the weight vector.

The product $W^T X$ is computed in-memory using a 12-T bit cell and the partial products obtained are then aggregated using capacitors. Weight vector, W, is stored in the bit-cell array and the input X is applied to the in batches.

3.2 Detailed Functional Description of the Modules

The following section details the functional description of each of the module used in the architecture. It details the architecture as well as the design specifications.

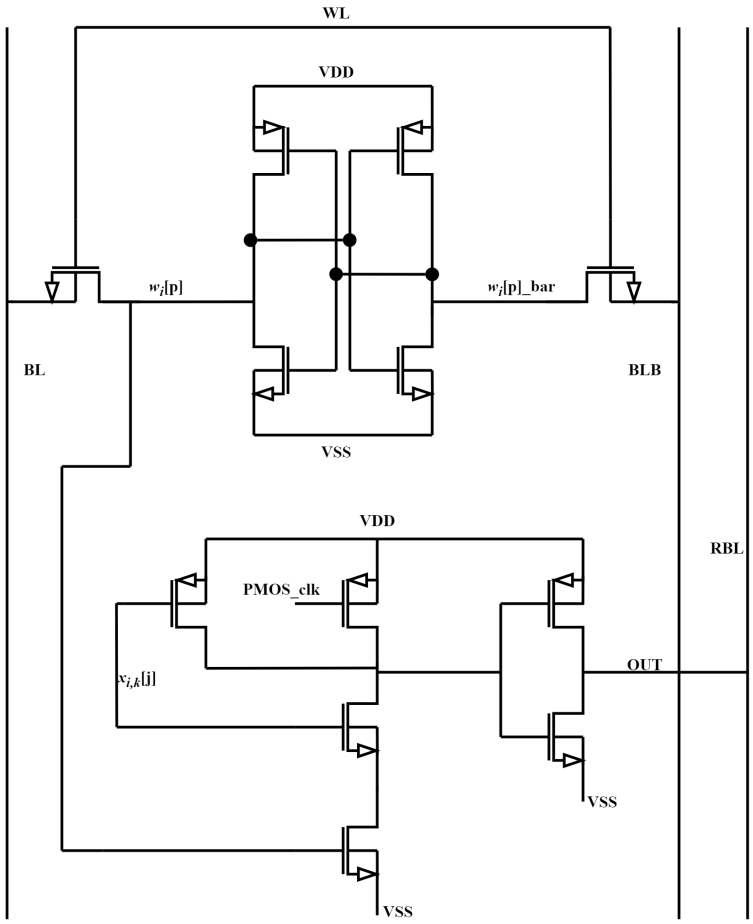


Figure 3.2: 12-T cell for computing two bit multiplication

3.2.1 Modified 12-Transistor Bit cell

Conventional 6T SRAM cell has been modified to include circuitry for computing two-bit multiplication. Figure 3.2 shows the 12-T cell.
In the bit cell, $x_{i,k}[j]$ denotes the j^{th} bit of the multi bit-scalar $x_{i,k}$ (Equation 3.1). Signal PMOS_clk is to assign binary weights to the weight bit, $w_i[p]$, stored inside the SRAM cell. It is driven by a PWM driver located outside the SRAM cell. Output terminal of the cell is connected to the RBL (Read Bit Line). RBL is used to accumulate the outputs from multiple 12-T cells connected in parallel to it. A capacitor is connected with it, to perform integration of current from the cells.

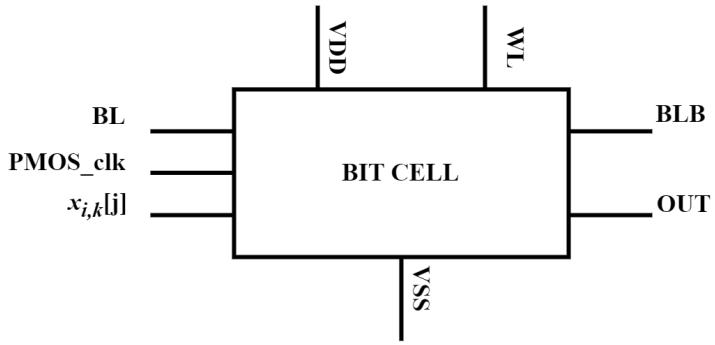


Figure 3.3: Block Diagram of the 12-T bit cell with inputs and outputs

3.2.2 Bit Cell Array (BCA)

Bit cells are connected in array fashion as shown in Figure 4. The module shown is to compute the multiplication of two 8-bit numbers, which in this case would be the multiplication of input vector x_i and w_j. Mathematically,

$$|w| = 2^7 w_{15} + 2^6 w_{14} + 2^5 w_{13} + 2^4 w_{12} + 2^3 w_{11} + 2^2 w_{10} + 2w_9 + w_8 \tag{3.3}$$

$$|x| = 2^7 x_7 + 2^6 x_6 + 2^5 x_5 + 2^4 x_4 + 2^3 x_3 + 2^2 x_2 + 2^1 x_1 + x_0 \tag{3.4}$$

$$\begin{aligned} |w||x| = (2^7 w_{15} + 2^6 w_{14} + 2^5 w_{13} + 2^4 w_{12} + 2^3 w_{11} + 2^2 w_{10} + 2w_9 + w_8) \\ (2^7 x_7 + 2^6 x_6 + 2^5 x_5 + 2^4 x_4 + 2^3 x_3 + 2^2 x_2 + 2^1 x_1 + x_0) \end{aligned} \tag{3.5}$$

$$\begin{aligned}|w||x| = (2^7w_{15} + 2^6w_{14} + 2^5w_{13} + 2^4w_{12})(2^7x_7 + 2^6x_6 + 2^5x_5 + 2^4x_4)+\\ (2^3w_{11} + 2^2w_{10} + 2^1w_9 + w_8)(2^3x_3 + 2^2x_2 + 2x_1 + x_0)+\\ (2^3w_{11} + 2^2w_{10} + 2^1w_9 + w_8)(2^7x_7 + 2^6x_6 + 2^5x_5 + 2^4x_4)+\\ (2^7w_{15} + 2^6w_{14} + 2^5w_{13} + 2^4w_{12})(2^3x_3 + 2^2x_2 + 2^1x_1 + x_0)\end{aligned} \tag{3.6}$$

And hence the design is split into four parts to implement each of the four terms.

Bit-line Capacitance and Frequency of Operation

Bit-line capacitor value and frequency of MAC operation has to be chosen such that the accumulated value is proportional to the time for which it gets integrated. Mathematically, for a RC charging circuit

$$V = V_0(1 - \exp(-\frac{t}{RC})) \tag{3.7}$$

This can be reduced to give linear variation with Δt as $V = V_o(1 + \frac{\Delta t}{RC})$; provided either C is very large, or integration time is small. Here, the expression is expanded using binomial expansion and higher order terms are neglected. But large on-chip capacitance is undesirable and hence need to optimally decide on the value of capacitance and the period of integration. Also, problem with large integration period is the voltage saturation which is observed and as a result the integrated value is no longer proportional to the integration period. This situation can be improved by decreasing the driving strength of the individual bit-cell and hence in the design, the cells used in the computation logic of 12-T cell are **minimum sized.**

Need of Scaling

The bit cell array has been divided into four separate sections (Figure 5). The pulse widths used for each section are the same. However, since the final accumulated voltage should reflect the actual scaling factors present in the multiplication, there is a need of a scaling before the voltages from the four blocks are added. This scaling is proposed to be done using a summer circuit implemented using an op-amp.

3.2.3 Pulse Width Modulation Drivers

In the design, Pulse width modulated drivers are used to drive the PMOS_clk and $x_{i,k}[j]$ inputs of the bit cell (Figure 3.3). PWM drivers' pulse widths are chosen in powers of 2

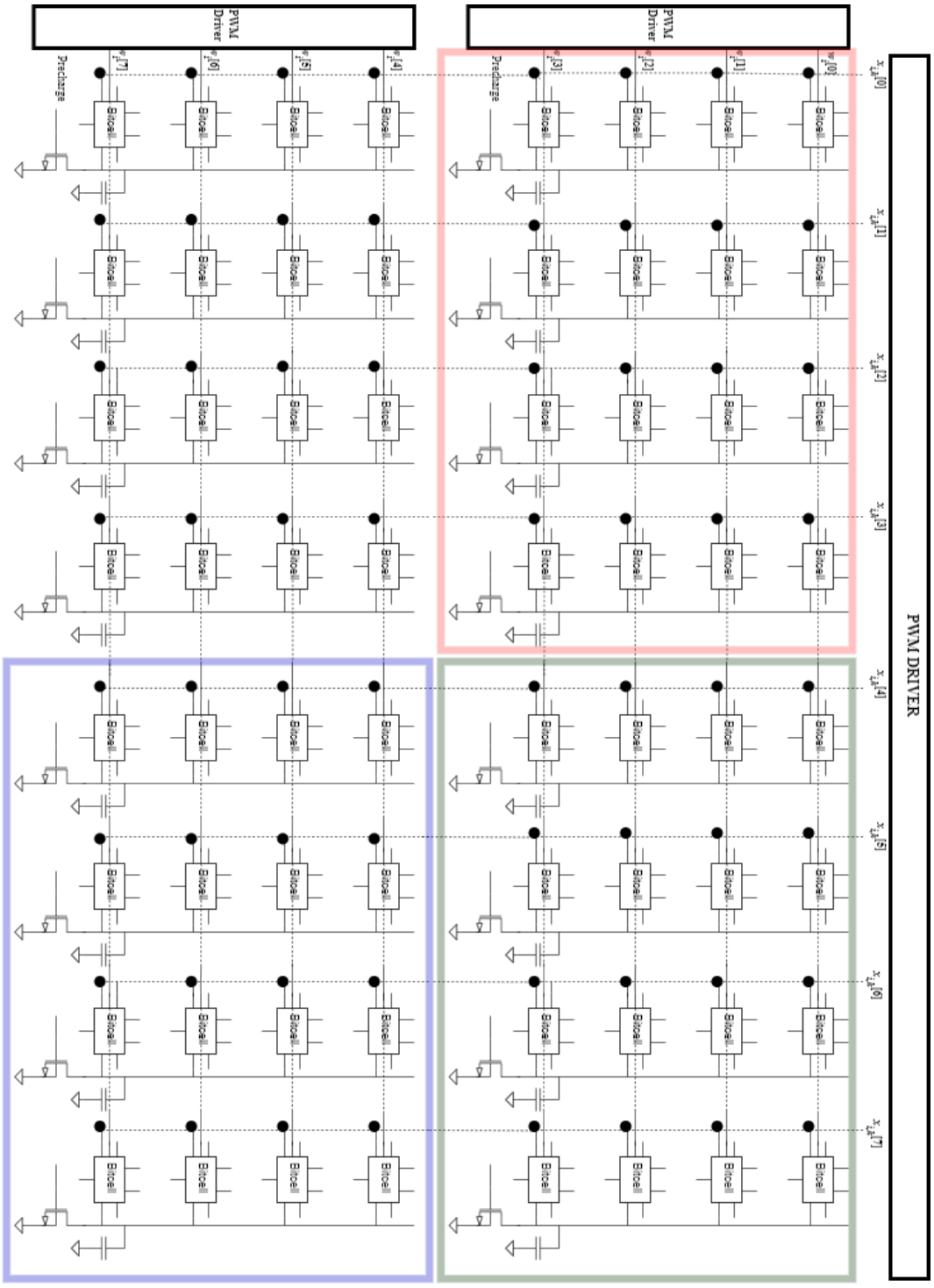


Figure 3.4: Organization of bit-cells to compute product. Bit-cell output is connected to corresponding RBL. VDD, VSS, BL, BLB and WL connections are hidden to improve readability

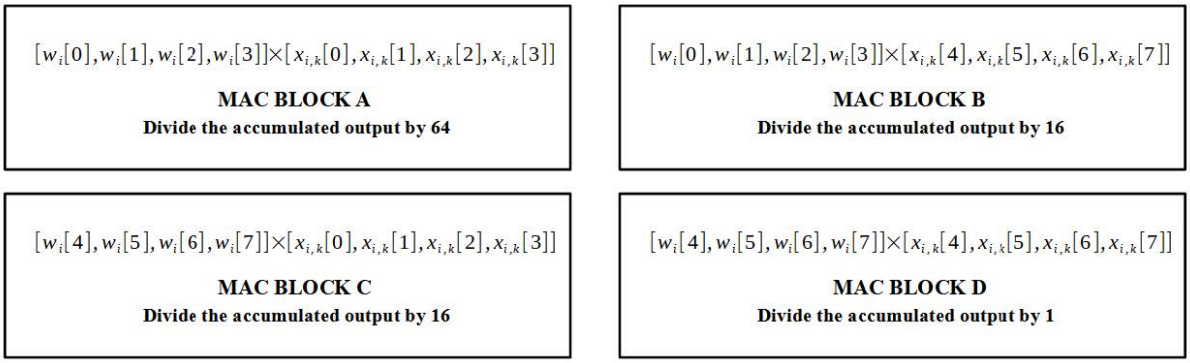


Figure 3.5: Sections in the architecture, also highlighted in Figure 3.4

(Figure 3.6).

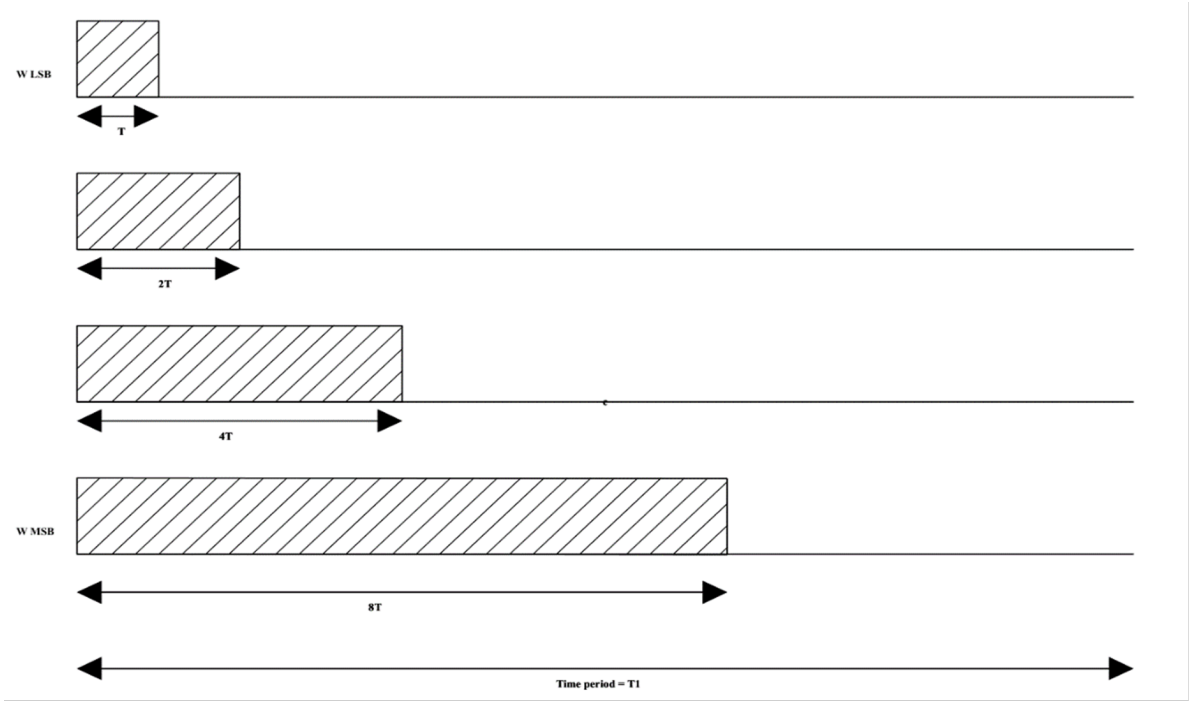
Idea behind selection of pulse widths is to ensure that output reflects the binary multiplication of weight and input bits. Since, accumulation essentially involves integration of currents from the individual cells, binary time periods make sure that this integration time is modulated according to the binary weights of the quantities involved. Another point to notice here is the way the periods for the weights and input vectors are selected. Pulse widths for the input vector driver is a multiple of the pulse period for the weights. So, the number of cycles for which weight vector effects output (through computation), is dependent on the pulse width of the input. Hence, by selecting the x-input pulse width as powers of 2, we ensure power of 2 cycles of computation at the output. It, also, intuitively reflects the mechanism behind multiplication and accumulation which is taking place.

3.2.4 Pre-Charge Circuit

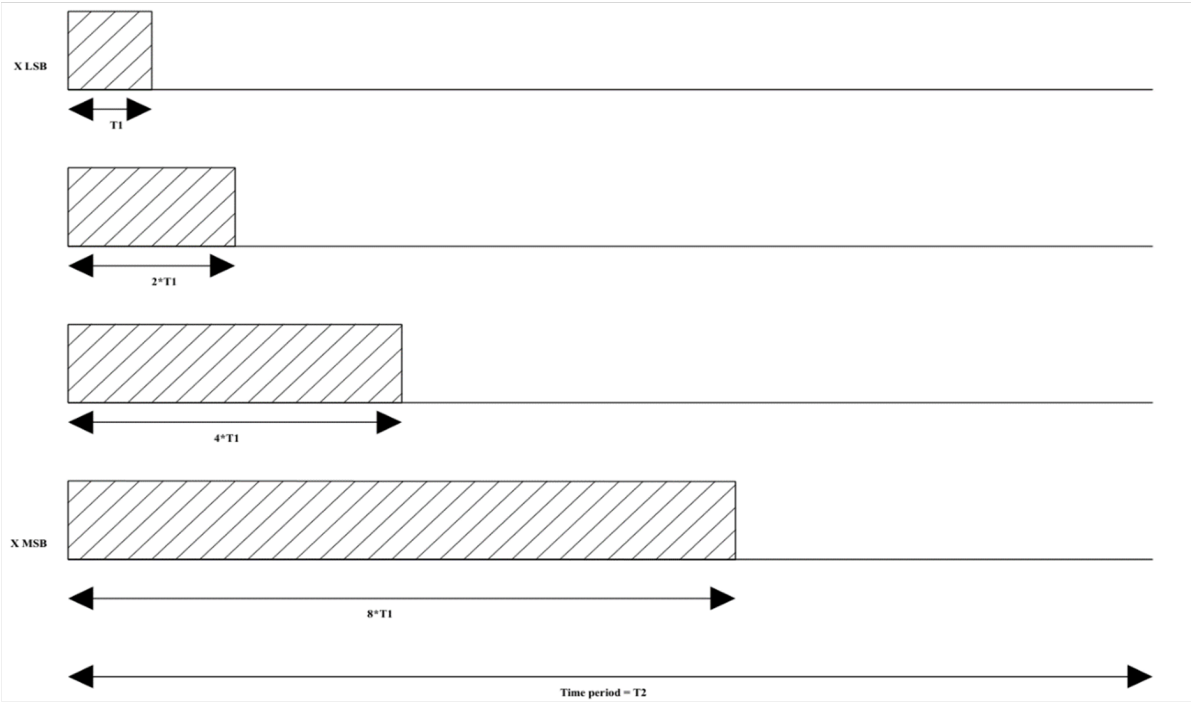
Multiply and accumulate operation is preceded by a pre-(dis)charge phase, which is done to ensure that the previously accumulated voltage doesn't affect the accumulated value in the current cycle. It is implemented using a NMOS connected to each of the bit line. It is controlled by the pre-charge signal generated by the control circuitry. The width of NMOS is to be selected appropriately taken into account the time available for the discharge and the size. Large sized transistor adds to parasitics and also an area overhead. Dis-charge phase also adds a time overhead to the total time required for the mac operation and hence is desired to be as low as possible.

3.2.5 Sensing and Scaling the Accumulated Result

There are multiple ways to sample the value accumulated on the bit line capacitors. Important design requirement of any such circuit is to have minimum leakage and low noise



(a)



(b)

Figure 3.6: (a) PWM pulses for weight vector (b) PWM pulses for input vector

addition. Loading effect should also be minimum as otherwise the accumulated value on the capacitor will get disturbed. The two designs were tried and tested during the project. First one was to sample the accumulated voltage using a switch and then use the held value for further processing like scalar multiplication and final accumulation. But this adversely affects the frequency of operation of circuit. With such a design, there will be a need of dedicated sampling period, during which no accumulation on the concerned capacitor can take place. This is an overhead similar to what is required for the pre-discharge phase at the beginning of each multiply and accumulation cycle. Further samplers are notoriously known for non-ideal effects like charge injection and clock feedthrough which introduce non-linear effects like gain and offset error. **Cap-hold T-switch** (6) (Figure 3.7) sampler circuit was designed and simulated. The circuit topology has a stack of transistors and has additional capacitors which during sample phase, gets charged to input voltage. These two specifications keep the drain to source voltages of the transistors in low range, limiting the leakage because of subthreshold current. Simulation results also proved the above assertions and there was significantly less leakage compared to simple transmission logic switch. Problem associated with the topology is the loading effect of additional capacitors, increased delay and the area overhead.

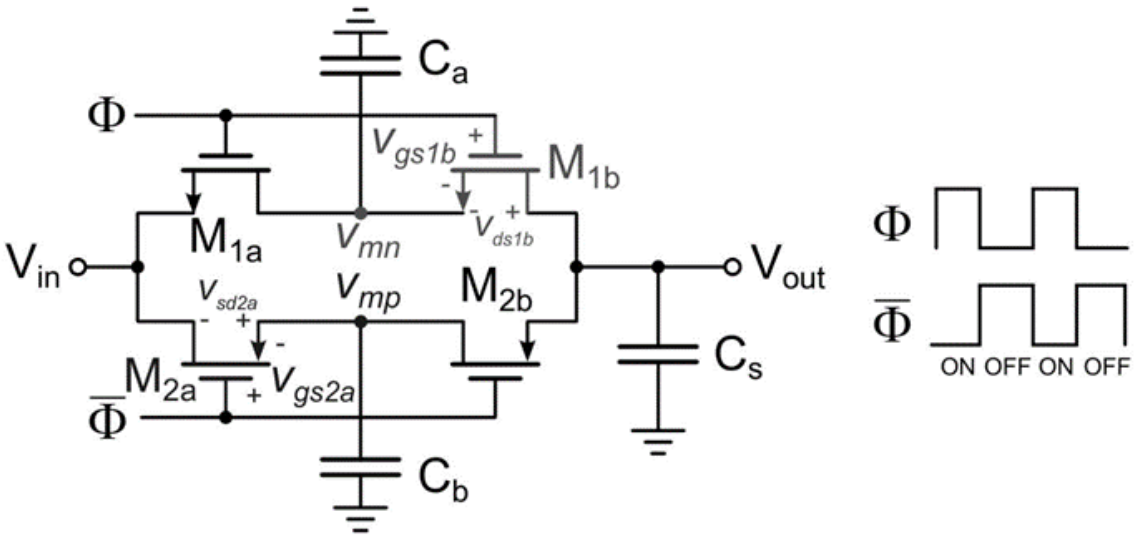


Figure 3.7: Cap-hold T switch (6)

Another way to sample the accumulated voltage is to do using a non-inverting amplifier. Since the input is at the high impedance node, there is no loading effect (except because of gate leakage, which is, however, small). This approach also helps remove the need of dedicated sampling overhead which was there in case of sampling using a switch. Simultaneous accumulation and sensing is possible in this approach. A non-inverting amplifier using two stage amplifier is designed. In this, first stage comprised of 5T OTA, followed by a common source amplifier (Figure 8). In both stages, PMOS configuration was used as the input

common mode varied in the low voltage range. The designed circuit worked well for a range of common mode value, however it failed to work properly for voltages below 100mV. This necessitated the need of rail-to-rail voltage amplifier and hence a circuit topology with both nmos and pmos stages incorporated in the signal amplification will have to be explored in the future work.

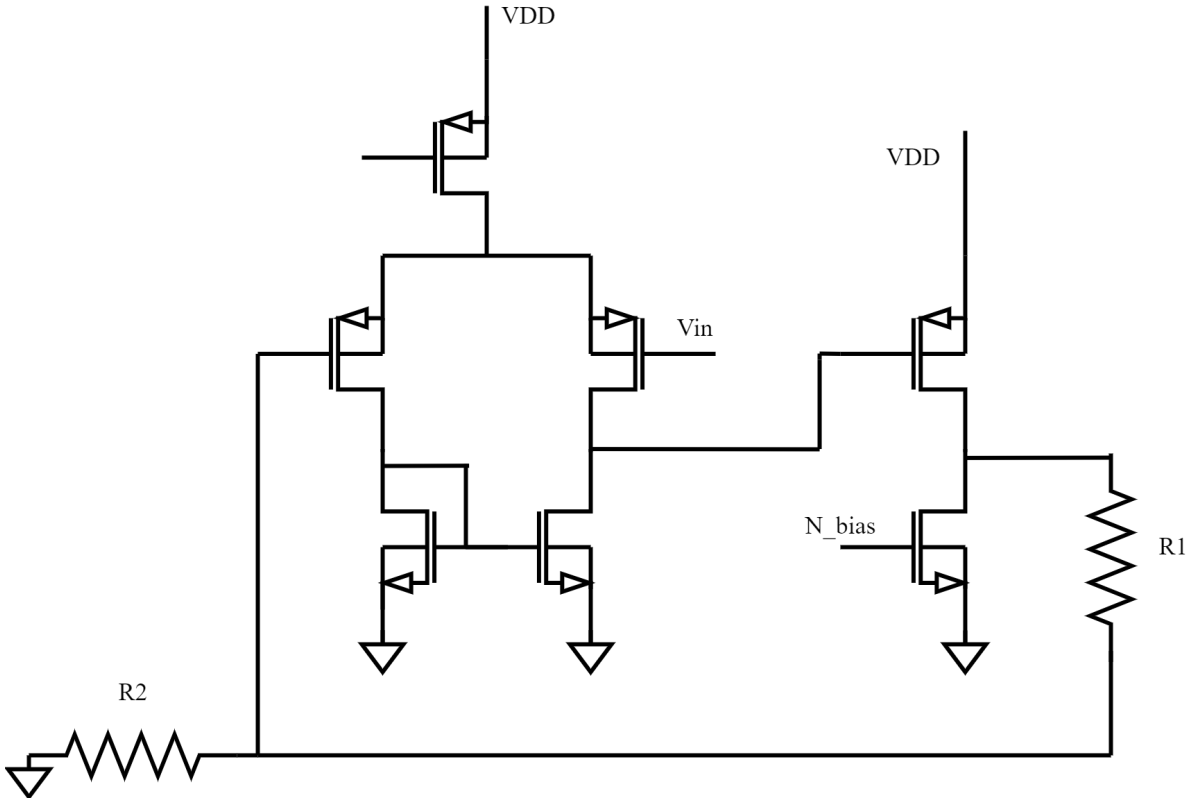


Figure 3.8: Non-inverting amplifier

This work uses switch based topology with dedicated control circuitry to operate the switches appropriately. Some of the observations while arriving at the final topology are, first, size of transistors need to be appropriately chosen to minimise the problems due to charge injection. Additionally a small capacitor can be connected at the junctions where switch connects a net to ground. This will help stabilizing that node voltage when switch is turned off.

Figure 3.9 shows the ciruit topology used to accumulate the voltages across bit lines before being sent to the scaling network. Switches S1, S2, S3, S4, S5, S6, S7 are switched ON and rest are switched OFF, when the accumulation is on individual bit-lines i.e. when the computation is taking place. After it is over and sensing phase starts, Switches S8, S9 and S10 are switched ON and rest are switched OFF. This makes the capacitors connect in series configuration. Output is then sensed from the out terminal.

Figure 3.10 depicts a novel, simple, capacitor charge sharing based circuit. It has been

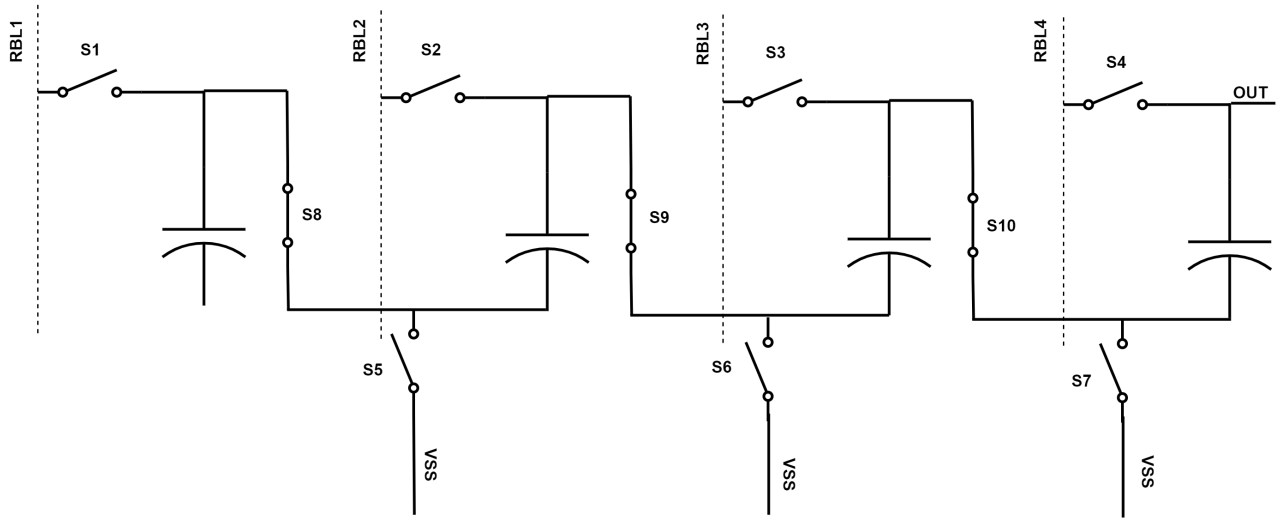


Figure 3.9: Accumulating the voltages across bit lines

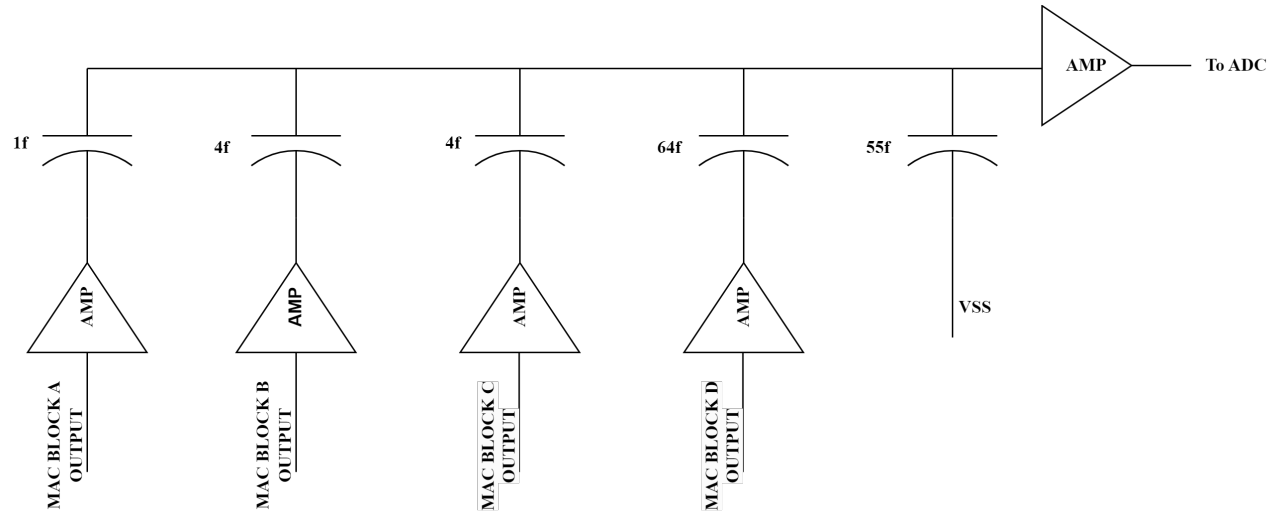


Figure 3.10: Scaling Network

used to achieve the scaling of voltages as discussed in figure 3.5.Here there is an additional scaling down by a factor of 2. Major design problem faced was in the design of the amplifier blocks as they need to supply sufficient current to charge the network while continuing to act as voltage buffer. Another point to consider is the initialisation of the network before use. It has been achieved by connecting capacitor nodes to ground via switches which are controlled by control signals. This block is further connected to Analog to Digital Converter with the help of an buffer, to avoid loading effect.

3.2.6 Successive Approximation Register (SAR) ADC

Analog to Digital Converter (ADC) is required to interface the analog components with the digital counter parts. Here in this architecture, computations are being done in analog domain and an ADC is required to return to digital domain. Successive Approximation Register (SAR) based ADC has been used in this work. In recent years, there have been notable advancements in the operational speed of successive approximation register (SAR) analog-to-digital converters (ADCs). Traditionally, SAR ADCs were characterized by their requirement for several comparison cycles to complete a single conversion, resulting in limited operational speed. However, with the continuous scaling down of CMOS device feature sizes, SAR ADCs have undergone significant improvements.

SAR ADC with monotonic capacitor switching procedure has several advantages such as reduced average switching energy, total capacitance and speed-related performance improvements. It requires only NMOS transistors while switching, and as a result, improved speed compared to PMOS counterparts.

Single ended ADC used here requires 10 comparison cycles to compute 8-bit output. Figure 3.11 shows the architecture used. Here, two reference voltages are required. These can be varied with appropriate change in the control circuitry. For the purpose of the thesis, Vin_Plus and Vin_Minus are sampled to VDD. Operation starts with a sampling phase where the input voltage is sampled onto the top plates of capacitors connected to the net Vin_Minus. Similarly, VDD is sampled on the net Vin_Plus. During this phase, bottom plates of capacitors is connected to Vref. This is followed by sequential switching of bottom plates to VSS beginning with capacitors connected to Vin_Plus. If comparator output is low, corresponding switch of capacitor which is connected to Vin_Minus is connected to VSS. This is followed for next 9 cycles to get 8-bit digital output.

The Figure 3.12 illustrates the above mentioned functioning of the ADC. Here, levels in black denote the voltage level at Vin_Plus where as in red denote the voltage level at Vin_Minus. Problems faced during designing of this block were, first, it was the initialisation of the capacitor network present inside the ADC. The buffer connecting the scaling

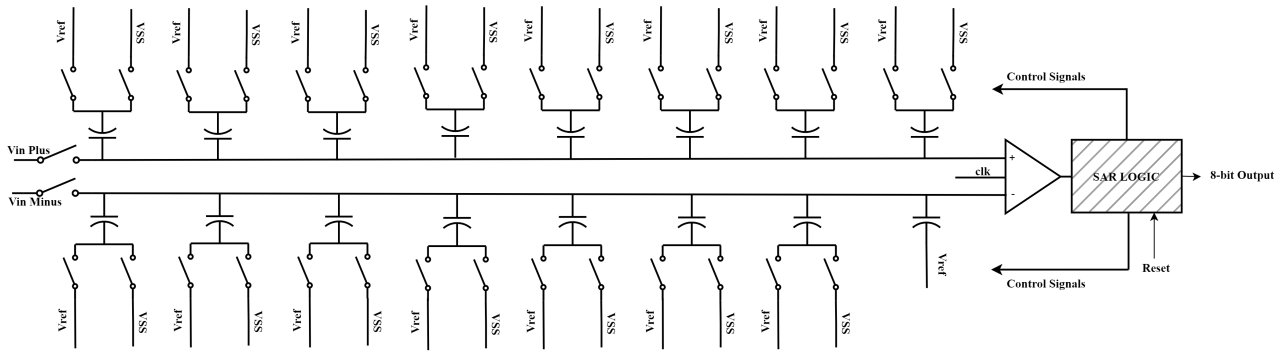


Figure 3.11: Successive Approximation Register ADC with Monotonic Capacitor Switching

network to the ADC need to supply excessively large current in a very small period of time. This problem was mitigated by incorporating a diode connnected NMOS at the input terminal of the ADC and ground. This was to quickly bring the input node voltage to close to zero. Negative feedback present in the diode configuration inherently made it to switch between cut-off and saturation according to node voltage. Note that this strategy worked in this architecture because ADC input voltage is a small voltage and hence lower input node voltage will reduce the loading on the buffer.

Another constraint was in the selection of capacitor's capacitance values. There is a trade-off between loading effect of large capacitors compared to voltage stabilisation problem in small capacitors. Very small capacitors though charge fast but are very prone to coupling and kick back noise present because of the comparator. The largest capacitance value used in this architecture is 640f F, while others are in multiples of two as required by a binary ADC.

3.2.7 Level Down Shifter

Simulation results proves the functioning of the circuit even at decreased supply voltages, paving a way to low-power design. Level down shifter was designed and simulated to give a stepped-down output voltage level of close to 0.5 V. It's performance is tested against PVT variations (Monte-carlo simulation) and varying capacitor loads.

3.2.8 Voltage Buffer

The circuit topology as depicted in 3.8 is used for the voltage buffer. This buffer need to operate at a varying input DC levels and have to supply sufficiently large current to load a mainly capacitive network which it is connected. Figure 3.10 denotes the various buffers which are required in the given architecture. As explained in the section on SAR ADC, the

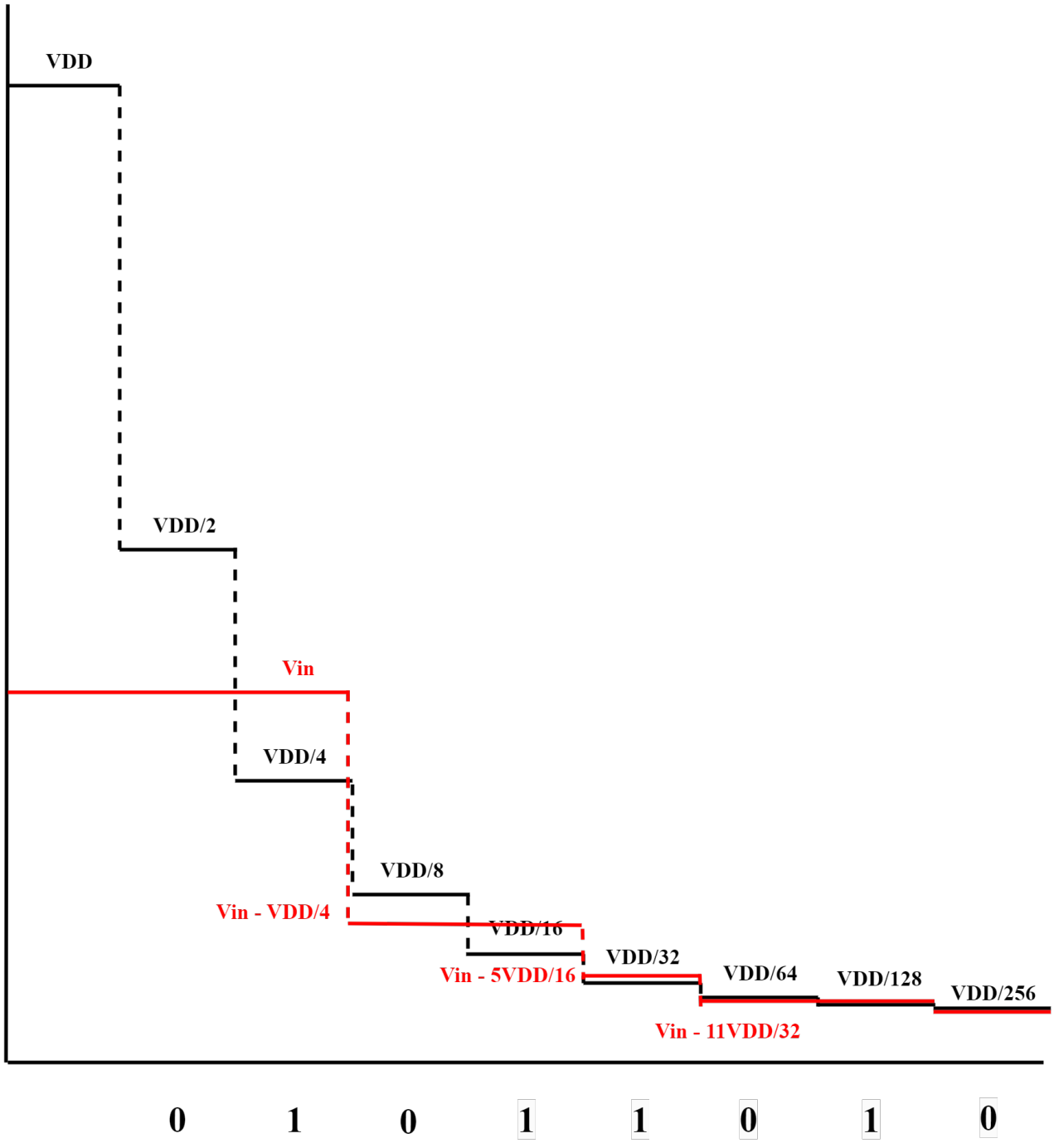


Figure 3.12: Flow chart for the voltage drop

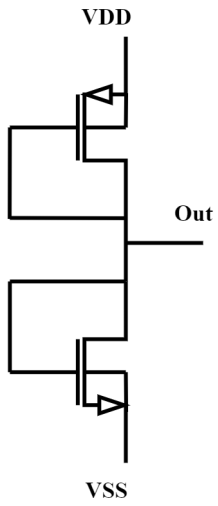


Figure 3.13: Level down shifter

buffer connecting the network to the ADC need to have a NMOS in diode configuration to quickly stabilise the node voltage.

3.2.9 Control Circuit

The Control circuit is designed in Verilog and synthesised using Digital Synthesis tool. It is a 29 state finite state machine with initial 5 states required only when there is a requirement to write into SRAM cells. Rest 24 states are required to perform computation. Out of these 16 states are required for integration of current on capacitors where as rest 8 states were for sensing and scaling. ADC itself require 10 states for conversion and it's operation is controlled and coordinated using the reset signal.

Chapter 4

Results and Discussion

Bit cell array architecture as given in Figure 3.4 is tested at 1 GHz clock frequency with a throughput of 41.6 MOPS. Control circuitry has been designed to generate control signals as well as the PWM pulses with a scheme as given in the Figure 3.6. Bit line capacitors are taken as 4pF, 2pF and 1pF, according to the binary weight which needs to be assigned. Architecture is also tested with varying input frequency, gate drive strengths and capacitor loads. Optimum value of capacitor value and frequency is selected so as to avoid output voltage saturation and hence maintain linear characteristics.
Voltage level shifter is tested and characterized against varying PVT conditions using Monte-Carlo simulation. Further in a standalone testbench, capacitor load is varied to mimic the effect of large capacitance present on the network.

4.1 Validation of Architecture

4.1.1 Under Varying Supply Levels

For the proof of concept, 4-bit input is multiplied with 4-bit weights (stored in SRAM cell) and the bit line accumulated voltage is observed. Input bits as well as Weights are taken to be all 1 and hence the accumulated voltage on the RBL's should be in the multiples of 2.

Table 4.1: Bit Cell Array Accumulated Voltages

Global VDD Level	**VDD=600m**		**VDD=500m**		**VDD=400m**	
Input Voltage Level	1V	600m	1V	500m	1V	400m
Bit Position						
RBL3	314.5m	264m	79.6m	67.3m	4.89m	4.2m
RBL2	143m	125m	36.7m	32m	2m	1.6m
RBL1	84m	81m	19.4m	17.3m	0.5m	0.3m
RBL0	38.5m	38m	8.5m	7.6m	0.2m	0.1m

Architecture is then tested for functionality under varying supply voltages and input voltage levels. Voltages are varied from 400mV to 1 V with two sets of observations. One where input voltage level is kept constant at 1 V and the other in which the input voltage range is kept equal to the supply voltage level (Global VDD level). As can be observeed

from the data in 4.1, trend of multiplication by a factor of two is observed. However at lower input voltage level i.e VDD =400m, there is not enough accumulation of voltage and at higher voltage level saturation effect is observed.

4.1.2 Under Varying Input Noise Levels

To test the robustness of the architecture against noise, gaussian zero mean noise is injected from the input side as well as from the output. Noise variance is varied from 0 to 60mV and the accumulated voltages are observed. This experiment is conducted for Global VDD level equal to 600m as well as 500m.

A general trend that can be observed from the table 4.2 is the small deviation in the observed voltages with noise variance. Performance of the circuit is better when the input voltage level is equal to 1 Volts i.e when it is not scaled with the global VDD level. This is, as was discussed in the previous chapter, because the transistors are in the triode region compared to sub-threshold and hence more linear behavior.

Table 4.2: Bit Cell Array Accumulated Voltage under varying Input Noise Variance

Global VDD Level	**VDD=600m**		**VDD=500m**	
Noise Variance	30m	60m	30m	60m
Bit Position				
RBL3	319m	310m	79.6m	78.2m
RBL2	145m	140m	36.3m	35.5m
RBL1	87m	82.2m	19m	18.1m
RBL0	40m	37.5m	8.23m	7.8m

Noise was also injected directly onto the bit-lines and as can be theoretically expected, the averaging effect of accumulation nulls out the effect of noise.

4.1.3 Cap-Hold T Switch Characterisation

Low leakage switch is characterized by measuring it's leakage current for a pulse input (hence maximum VDS). Table 4.3 proves the low leakage nature of the switch topology and hence has been used in the main architecture.

Table 4.3: Leakage Specifications of Switch

Max Current During Sample Phase	1.1mA
Max Leakage Current During Hold Phase	225pA

4.1.4 Scaling Network Verification

Scaling Network is tested for it's functionality with varying inputs in-order to establish the correctness of the theoretical results. Results establish the theoretical derivation and hence has been used in the main architecture.

Table 4.4: Functionality Verification of Scaling Network

V_{128}	V_{32}	V_{32}	V_2	**Result(Simulation)**	**Result(Theory)**
140m	140m	140m	140m	79.7m	79.84m
128m	32m	32m	2m	4m	4m

4.1.5 Successive Approximation Register ADC Specifications

Successive Approximation Register has been tested with varying inputs and ENOB calculation has been done. As can be inferred from 3.11, net Vin_PLUS is sampled to VDD and pin Vref is connected to VDD for the purpose of computation.

Table 4.5: Analog to Digital Converter Specifications

Specification (Unit)	**Value**
Supply Voltage (V)	1
Frequency of Operation (V)	50MS/s
Input Range (V_{p-p})	1
ENOB	5.98

4.1.6 Buffer Specifications

Non-inverting amplifier is tested against varying input common mode voltage and the open loop gain is measured over the bandwidth against each test case. This is then used as a buffer to drive the various capacitive loads as was discussed in the previous chapter.

Table 4.6: Buffer Specifications

Specification (Unit)	**Value**
Supply Voltage (V)	1
Open loop gain (V)	23
Stages	2

4.1.7 Overall Specifications

Overall functionality of the circuit has been tested against varying input vector and corresponding results have been reported.

Table 4.7: Architecture Output with Buffer

X[7:0]	W[7:0]	Accumulated Voltage	ADC Output
8'b11111111	8'b11111111	69.46m	8'b00001111
8'b01111111	8'b11111111	35.2m	8'b00000111
8'b00111111	8'b11111111	21m	8'b00000101
8'b11111111	8'b01111111	29m	8'b00000111
8'b11111111	8'b00111111	17.5m	8'b00000011
8'b01111111	8'b01111111	18m	8'b00000011

Table 4.8: Architecture Output With Ampilfier of gain 13

X[7:0]	W[7:0]	Accumulated Voltage	ADC Output
8'b11111111	8'b11111111	843m	8'b11010111
8'b01111111	8'b11111111	413m	8'b01101001
8'b00111111	8'b11111111	251m	8'b00111111
8'b11111111	8'b01111111	376m	8'b01010111
8'b11111111	8'b00111111	211m	8'b00110101
8'b01111111	8'b01111111	219m	8'b00110111

4.2 Conclusion

The work done in the thesis provides a proof of concept for the novel bit-cell array architecture (Figure 3.4) for multiply and accumulate operation. The architecture has been characterized against varying supply voltages, input levels and noise levels. It works best with supply level close to 0.5 volts and input level of 1 volt and its potential application involve low power and area aware computing. Further for the sensing of accumulated voltage, switch capacitor based charge sharing network has been used. The architecture for scaling as in Figure 3.10 is also novel. Cap hold T-switch switch architecture (Figure 3.7) gives an extremely low leakage current, as supported by simulation results, useful for low leakage applications. Also, in order to scale down the voltage, a simple level-down shifter circuit supporting varying cap load can be designed using diode connected transistors.
Further for analog to digital conversion, SAR based ADC with monotonic switching has been used and verified. The architecture has been modified to be used with single ended signals. Since, the overall architecture involves approximate computing, specifications of the ADC can be relaxed. Buffers play an integral role in interconnecting different modules and hence need proper designing, ensuring linear operation and ability to supply sufficient current to the subsequent stages.

4.3 Future Scope

The search for alternate architectures will continue to remain, given the shortcomings of Von Neumann architecture and growing processing requirements. Work in this thesis has explored one such architecture based on in-memory computation topology. Further refinement of architecture can be done in following ways and thus constitutes the future scope.

1. **Design of Voltage Buffer :** Buffer is an essential component of the architecture and it need to work for a wide range of input DC voltages. It's architecture can be further explored to provide better linearity.
2. **Sensing Circuit :** Current architecture uses capacitors and switches to sense the accumulated voltage on the bitline capacitors. But this strategy leads to scale down of voltage during sensing. Alternate way could be to use an non-inverting amplifier but again it should be linear for the wide range of input DC voltages.
3. **Analog to Digital Converter Circuit :** This needs to optimally designed given the approximate nature of calculation and hence there can be scope for further relaxation in the ENOB requirement.
4. **Integration into Support Vector Machine Accelerator :** Such an application will act as a guide for the actual requirements for the specifications of various blocks used in the architecture.

Bibliography

[1] M. Kang, S. K. Gonugondla, and N. R. Shanbhag, "Deep in-memory architectures in sram: An analog approach to approximate computing," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2251–2275, 2020.

[2] C.-J. Jhang, C.-X. Xue, J.-M. Hung, F.-C. Chang, and M.-F. Chang, "Challenges and trends of sram-based computing-in-memory for ai edge devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 5, pp. 1773–1786, 2021.

[3] R. Sehgal and J. P. Kulkarni, "Trends in analog and digital intensive compute-in-sram designs," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–4, 2021.

[4] S. Bavikadi, A. Dhavlle, A. Ganguly, A. Haridass, H. Hendy, C. Merkel, V. J. Reddi, P. R. Sutradhar, A. Joseph, and S. M. Pudukotai Dinakarrao, "A survey on machine learning accelerators and evolutionary hardware platforms," *IEEE Design Test*, vol. 39, no. 3, pp. 91–116, 2022.

[5] C.-C. Liu, S.-J. Chang, G.-Y. Huang, and Y.-Z. Lin, "A 10-bit 50-ms/s sar adc with a monotonic capacitor switching procedure," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, pp. 731–740, 2010.

[6] B. Wang, S. Wang, and M.-K. Law, "On low-leakage cmos switches," in *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1–5, 2021.

[7] S. K. Gonugondla, M. Kang, and N. Shanbhag, "A 42pj/decision 3.12tops/w robust in-memory machine learning classifier with on-chip training," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pp. 490–492, 2018.

[8] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam, "Dadiannao: A machine-learning supercomputer," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 609–622, 2014.

[9] D. Liu, T. Chen, S. Liu, J. Zhou, S. Zhou, O. Teman, X. Feng, X. Zhou, and Y. Chen, "Pudiannao: A polyvalent machine learning accelerator," *SIGPLAN Not.*, vol. 50, p. 369–381, mar 2015.

[10] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "Shidiannao: Shifting vision processing closer to the sensor," in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, pp. 92–104, 2015.

[11] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '16, (New York, NY, USA), p. 26–35, Association for Computing Machinery, 2016.

[12] M. F. Ali, A. R. Jaiswal, S. Kodge, A. Agrawal, I. Chakraborty, and K. Roy, "Imac: In-memory multi-bit multiplication and accumulation in 6t sram array," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, pp. 2521–2531, 2020.

[13] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "Xnor-sram: In-memory computing sram macro for binary/ternary deep neural networks," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, 2020.

[14] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "Drisa: A dram-based reconfigurable in-situ accelerator," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 288–301, 2017.

[15] L. Jiang, M. Kim, W. Wen, and D. Wang, "Xnor-pop: A processing-in-memory architecture for binary convolutional neural networks in wide-io2 drams," in *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1–6, 2017.

[16] S. Li, A. O. Glova, X. Hu, P. Gu, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "Scope: A stochastic computing engine for dram-based in-situ accelerator," in *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-51, p. 696–709, IEEE Press, 2018.

[17] Q. Deng, L. Jiang, Y. Zhang, M. Zhang, and J. Yang, "Dracc: a dram based accelerator for accurate cnn inference," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2018.

[18] C.-C. Liu, S.-J. Chang, G.-Y. Huang, Y.-Z. Lin, C.-M. Huang, C.-H. Huang, L. Bu, and C.-C. Tsai, "A 10b 100ms/s 1.13mw sar adc with binary-scaled error compensation," in *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, pp. 386–387, 2010.

[19] C.-C. Liu, M.-C. Huang, and Y.-H. Tu, "A 12 bit 100 ms/s sar-assisted digital-slope adc," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 12, pp. 2941–2950, 2016.