Rethinking Computational Architectures at the Edge Through Asynchronous Temporal Streams

---

A

Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Rahul Sreekumar

December 2024

APPROVAL SHEET

This

Dissertation

is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Rahul Sreekumar

This Dissertation has been read and approved by the examing committee:

Advisor: Mircea R. Stan

Advisor:

Committee Member: Avik Ghosh

Committee Member: Steven M. Bowers

Committee Member: Ashish Venkat

Committee Member: Bhupendra Singh Reniwal

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

December 2024

Abstract

The escalating trend towards processing information and abstract features at the edge has compelled researchers to reconsider the methodology for performing these tasks with minimal energy consumption. Edge accelerators confront unique challenges that demand a reconsideration of the fundamental approach to algorithm execution. Notably, the sporadic nature of peak workload demands at the edge calls for a solution involving synchronizing the workload execution rate with the actual occurrence of events. This thesis explores the potential of architectures that compute information and process signals in the temporal domain, introducing the concept *stimulus-driven workload execution* by leveraging Asynchronous Stream Computation (ASC) principles to expedite edge Machine Learning workloads. A comprehensive analysis is conducted on the fundamental computational elements required for developing Neural Network (NN) models to evaluate the impact of an ASC-based computational paradigm.

A primary objective of this research is to investigate mechanisms addressing the disparity between conventional processing power and the high memory bandwidth required for storing high-dimensional data vectors. This goal is realized through the development of Compute-in-Memory (CiM) architectures, utilizing asynchronous streams to regulate the execution rate of data-intensive operations like Vector-Matrix Multiplications (VMMs), Multiply-and-Accumulate (MAC), and dot product, among others. To gauge the impact of these stream-based CiM tiles in practical Machine Learning scenarios, the thesis implements an end-to-end streaming Convolutional Neural Network image classifier model aligned with Asynchronous Stream Computation (ASC) principles. The implemented classifier architecture exhibits a scalable performance between 28 - 249 Frames/second (FPS) while maintaining an energy efficiency of 81.54 and 247.08 TOPS/W. Furthermore, the research encompasses the meticulous design of essential computational elements to facilitate and optimize this implementation.

The thesis also delves into reimagining the neural encoding scheme within Spiking Neural Networks (SNNs) using the proposed $\Sigma\Delta\Sigma$ encoding. This encoding technique enables dual-purpose neural connections, facilitating feature extraction while suppressing noise-like features from propagating through the network. These neurons possess unique noise-filtering characteristics, making them suitable for integration into models capable of robustly handling input feature-dependent noise and random temporal perturbations within the physical signal. We implemented reservoir computing networks in the spiking domain and demonstrated their effectiveness by implementing liquid-state machine models for time-series predictive engines. We also showed the robustness of the network against low-quality audio.

Acknowledgments

This dissertation would not have been possible without the mentorship, support, and love of many people and organizations. I am profoundly grateful to everyone who has participated in this journey.

First and foremost, I would like to thank my family and closest friends. The Ph.D. process was long and challenging, and I deeply appreciate the unwavering encouragement and love they provided every step of the way. To my advisor, **Mircea Stan**, I extend my heartfelt gratitude. His guidance and encouragement have been instrumental in shaping my research path. I am especially thankful for his rare quality of allowing me the freedom to explore new research horizons. His hands-on approach across diverse research disciplines allowed me to collaborate beyond my immediate field, elevating my knowledge and skill set as a researcher and an engineer. For that and much more, thank you, Professor Stan. To my mentor, **Avik Ghosh**, I am grateful for your expertise and insightful guidance. Your broad knowledge base across multiple research areas has significantly enriched my work. The many brief yet impactful conversations we shared have improved the quality of my research, shaped my approach to work, and reinforced my dedication to excellence.

To my early mentors in education, especially **Sunitha K. Beevi**, thank you for showing me what it means to be an inspiring educator. You have taught me lessons that extend well beyond academics.

A special thanks to **Beth Eastwood Beatty** for helping me navigate the complexities of university processes and for so much more. Your support made many challenges easier to face, and for that, I am incredibly thankful.

To my collaborators and friends at UVA and beyond, including **Nazmus Sakib**, **Lingxi Wu**, **Faiyaz Mullick**, **Patricia Gonzalez**, **Minseong Park**, **Bhupendra S. Reniwal**, **Melika Morsali**, **Sergiu Mosanu**, **Elisa Pantoja**, and everyone in the **High Performance Low Power** lab, thank you for your camaraderie and support. I look forward to a lifetime of friendship and collaboration with all of you.

Finally, my sincere gratitude to the **Semiconductor Research Corporation**

(SRC) for providing the resources that made this research possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Edge Machine Learning (ML) workloads frequently involve the utilization of high-dimensional feature vectors, and they exhibit a pronounced demand for memory, particularly in the context of Convolutional Neural Networks (CNNs). Nonetheless, the persistent challenge known as the "memory-wall" problem, characterized by a discernible performance gap between computational power and memory bandwidth, necessitates the exploration of innovative data computation architectures. One such solution is the adoption of a Compute-in-Memory (CiM)-based architecture, where arithmetic computations take place within the memory array. The ongoing research and development efforts in both academia [59] and industry [20] have yielded CiM-based accelerators demonstrating energy efficiency, surpassing 100 TOPS/W.

Edge accelerators face distinctive challenges owing to sporadic peak workload demands triggered by intermittent occurrences. Sporadic events are characterized by their aperiodic nature and the crucial importance of actionable information within each event. Current methodologies either necessitate the device to be in an "always on" mode, continuously "sniffing" for actionable events, or rely on a smaller processor to "wake up" the main processor in the presence of relevant information. Exacerbating this situation, a constrained energy budget complicates matters, requiring algorithms to operate in the most energy-efficient state.

While traditional architectures use wake-up processors for dual-mode performance scaling, employing techniques like Dynamic Voltage and Frequency Scaling (DVFS)[8]

for finer granularity, these approaches face scalability limitations and encounter challenges with emerging non-volatile memory (eNVM) devices, as noted in existing literature[71]. Conventional design for edge accelerators balances energy efficiency and accuracy, or performance and accuracy [29, 39]. Techniques such as neuron skipping, reduced bit precision, weight pruning, etc., are common but result in static solutions. However, as we transition to "ultra-edge" devices with renewable energy sources, energy cycle unreliability becomes a key bottleneck, causing potential downtime when energy reserves drop below critical thresholds.

We propose an innovative computational approach inspired by biological neural systems, utilizing Asynchronous Stream Computing (ASC) based temporal streams to encode information within the temporal domain. These streams represent data through the duty cycle and frequency of a digital signal, facilitating sparse event representation and seamless interaction with Computation-in-Memory (CiM) technologies. Our approach involves designing computational elements and neurons adhering to ASC principles, introducing "stimulus-based workload execution". In this paradigm, the rate at which computations are performed directly correlates with the amplitude and rate of change of the actionable event. This paradigm dynamically scales performance in real-time based on event occurrences, offering a more flexible solution. To address energy scalability bottlenecks in ultra-edge devices, we scale the frequency spectrum of the encoding technique, modifying computation energy. Our goal is to shift the design space from energy efficiency versus accuracy to energy efficiency versus performance, prioritizing workload execution at multiple energy thresholds, even if it entails reducing the computational throughput. We demonstrated the "stimulus-driven workload execution" technique with a Convolutional Neural Network (CNN) for image classification. Our evaluation assessed the accelerator's energy efficiency across different inference quantities. Employing a circuits-to-architecture design strategy, we meticulously simulated various computational elements, laying the groundwork for accelerator development. This dissertation includes the hardware module library to implement additional feature abstraction modules and processing elements, enabling the creation of diverse CNN models capable of handling complex

datasets, thus enhancing the versatility of our approach.

Given that our approach utilizes temporally encoded streams for conveying and computing information, there is a natural inclination to explore compatibility with spiking neurons, which operate in the temporal domain. Spiking neural models have garnered considerable attention recently due to their efforts to mimic the energy efficiency observed in biological systems [26]. These models encode information in sparse dimensions, offering enhanced energy efficiency and facilitating efficient feature representation by minimizing redundancy. In recent years, the research community has extensively explored various neural encoding techniques to replicate properties observed in biological systems, such as those found in audio cochlear [73] and dynamic vision-based systems [27]. This has led to sparser schemes such as Delta modulation (Δ) and Sigma-delta ($\Sigma\Delta$) encoding. These schemes enable feature communication between neurons only when they exceed a certain threshold, leading to even sparser feature representations between layers within a network.

In this dissertation, we extend the capabilities of delta-modulated neurons by introducing a novel evolution in this encoding approach termed Sigma-Delta-Sigma ($\Sigma\Delta\Sigma$/SDS) neural encoding. Our proposed encoding method stands out for its adeptness in filtering out noise-like components from critical features, achieved through a unique feedback mechanism and intrinsic delay within the encoding technique. This work aims to construct noise-robust spiking models capable of feature extraction even in the presence of noisy input stimuli.

1.1 Hypothesis and Contributions

Hypothesis: Reimagining edge computation through the utilization of asynchronous temporal streams emerges as a transformative approach, fostering an energy-scalable architecture and empowering the implementation of noise-robust neural encoding techniques tailored for spiking-based models. In defending this dissertation, we will make the following contributions to the state of the art:

- **Open source simulation framework:** We aim to create an open-source

framework using Python, dedicated to modeling the behavioral aspects of ASC computational elements. This framework will seamlessly integrate these elements, enabling the emulation of various neural network models. Its primary purpose is to expedite the development of asynchronous stream-based neural models, inspire future research, and offer benchmarks for device researchers. Moreover, the framework will facilitate the incorporation of device models, allowing researchers to simulate the performance of temporal-based in-memory computation platforms.

- **Redesign CNN computation elements in the ASC domain and generate customized cell library:** The simulation framework will facilitate a swift architecture assessment process, leveraging the speed of design-to-test inherent within software simulation. In parallel, we plan to unveil hardware IP modules to actualize robust ASICs from the simulated models. These modules will encompass standard cell libraries, featuring computational elements designed based on the open-source Process Design Kit (PDK) from Skywater 130nm technology. The intention is to make these modules readily available to the broader research community.

- **Develop noise-resilient spiking neural encoding utilizing Lava as the base framework**: To introduce our innovative *Sigma-Delta-Sigma* neuron, we plan to integrate our encoding technique into the Intel Labs Lava software platform. By implementing this neural encoding approach, our objective is to develop noise-resilient spiking neural network models capable of performing classification tasks even with lower-quality incoming features.

1.2 Organization

The remaining sections of the dissertation are organized as follows:

Chapter 2: Background introduces the current state-of-the-art (SoA) approaches to implementing edge-based ML accelerators and the challenges faced while

implementing these solutions.

Chapter 3: Asynchronous Stream Computing: Computing through Temporal Streams presents the fundamental concepts of ASC that enable an energy-scalable and precision-scalable computational paradigm and quantifies the cost for this scalability in terms of energy efficiency.

Chapter 4: EAS-CiM: Foundational Blocks for CNN models introduces the various circuit elements designed to implement computational elements for CNN models and presents benchmarking results based on datasets used by academia and industry.

Chapter 5: $\Sigma\Delta\Sigma$ Neuron: Noise-Resilient Spiking Neural Encoding introduces a novel spiking encoding mechanism and depicts its unique noise-invariant mechanism through information encoding benchmarking models and datasets used by the industry.

Chapter 6: Conclusions and Future Work summarizes the dissertation and further discusses the implications and future research directions of our work.

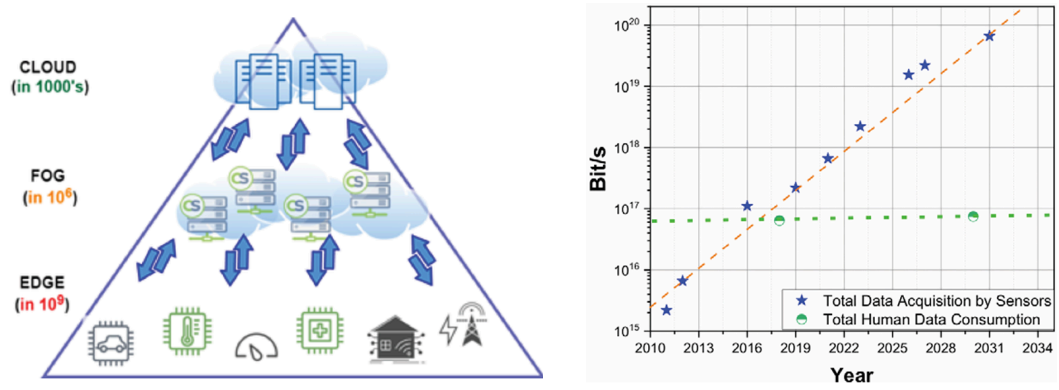
Chapter 2

Background

2.1 Specific Challenges in Edge Computing for ML

The rapid expansion of IoT applications is expected to lead to the deployment of approximately 41.6 billion IoT devices by 2025, collectively generating close to 79.4 zettabytes (ZB) of data globally [46]. To manage this substantial data influx and streamline processing, the predominant framework employed is a tiered pyramid structure, which divides data processing responsibilities across different layers (Fig. 2-1a). At the base layer, edge devices collect raw data directly from the environment and relay it upward to the intermediate "fog" layer. In this fog tier, preliminary network-level computations and routing occur, effectively serving as a conduit that reduces the load on higher layers. Finally, data reaches the cloud at the top of the pyramid, where intensive data analysis and computation take place. This tier is also the primary interface for end users, allowing them to access the processed information.

Despite its effectiveness, this pyramid structure has limitations, particularly in latency and real-time response. Consequently, recent advancements have focused on shifting more processing capabilities directly to the edge layer. By processing data closer to the source, edge devices can deliver actionable insights and reduce the dependency on cloud-based processing. This approach not only mitigates the latency associated with multi-tier data relays but also enhances privacy, as data remains closer to its origin. Edge computing, therefore, presents a promising alternative by enabling



(a) (b)

Figure 2-1: **(a) Present day pyramid distribution of computing devices, (b) Global sensory data generation and expected growth trend [2].**

timely responses and empowering end users with near-instant access to meaningful information without relying solely on the cloud for computation. Alongside the shift from cloud to edge computing, there is a compelling motivation to concentrate on edge-based processing capabilities. In recent years, we have witnessed a rapid surge in the volume of sensory data generated and collected by edge devices, a trend expected to grow exponentially over the next decade (Fig. 2-1b). However, a key insight emerges from this increase in data volume: despite exponential data growth, the actual information extracted has reached a point of equilibrium. This observation is crucial for information extraction strategies, highlighting the need for algorithms that can effectively compress raw data to yield "actionable information."

To address this, the development of optimized edge-based machine learning (ML) models becomes essential. Such models must not only be lightweight and efficient to operate at the edge but also capable of producing high-value insights from extensive data streams. Furthermore, advancing hardware architectures that can accelerate these ML algorithms on edge devices is equally critical. A holistic hardware-software co-design approach is needed to ensure the seamless integration and deployment of edge-oriented models and devices, creating a unified system that maximizes performance while minimizing latency and energy consumption.

The widespread deployment of IoT and edge devices brings unique challenges, particularly in environments where stable power supplies are either impractical or

impossible to maintain. For example, sensors deployed in remote or harsh conditions often cannot rely on "battery-powered" solutions due to logistical limitations, and frequent maintenance or replacement of batteries is often not feasible. Instead, energy harvesting systems provide an alternative by capturing ambient energy from sources like solar radiation, thermal gradients, and kinetic movement [51, 66]. However, these energy sources are inherently variable and intermittent, leading to power outages that disrupt device functionality and result in data loss within volatile memory.

To manage these issues, energy-efficient power management and data processing techniques such as Dynamic Voltage and Frequency Scaling (DVFS) [8, 50] and wake-up processing have been adopted. DVFS dynamically adjusts a processor's voltage and frequency based on workload demands, allowing the system to save power during low-activity periods by reducing the operating frequency and voltage. This technique minimizes energy consumption while maintaining a balance between power and performance. However, DVFS's effectiveness is often limited to certain levels of performance scaling, as it usually operates in discrete steps and may be constrained to a few predetermined power-performance states. This restriction means that while DVFS can help reduce power consumption, it cannot fully address the high variability in available power for energy-harvesting devices.

Complementary to DVFS, wake-up processing introduces an additional layer of power savings by utilizing a secondary, ultra-low-power processor that can handle basic tasks and activate the primary processor only when necessary [69, 55]. This approach significantly extends battery life by allowing the main processor to remain inactive during idle periods. While wake-up processing provides energy efficiency, it too has limitations due to its binary "on-off" nature, offering only two levels of performance—active or standby. As a result, it lacks the granularity needed to scale performance according to variable workloads, which is essential in environments where power availability and computational demands can fluctuate widely.

In the context of deploying machine learning (ML) algorithms at the edge, a key constraint arises from the "memory-intensive" nature of these workloads. Unlike traditional computational tasks, ML workloads often require significant amounts of

data to be moved repeatedly between memory and the processing unit, exacerbating the well-known "memory wall" problem. The memory wall describes the growing disparity between a processor's speed and the limited bandwidth of memory, creating a bottleneck where memory access becomes the dominant limitation on system performance. This issue is particularly acute in ML workloads, which rely heavily on data-intensive operations, such as matrix multiplications and convolutions, that demand frequent memory access. Consequently, as computational power continues to increase, the relatively slower memory access speed severely constrains the overall performance of ML models.

For edge computing, where power is limited and energy harvesting cycles are inconsistent, this memory bottleneck becomes even more drastic. The need for high-density, low-energy, non-volatile memory (NVM) devices that can handle frequent read-write cycles without excessive power consumption is paramount. This demand has driven research toward developing emerging Non-Volatile Memory (eNVM) technologies, such as Resistive RAM (ReRAM), Magnetoresistive RAM (MRAM), and Phase Change Memory (PCM) [37, 52, 62]. These eNVM technologies are attractive candidates due to their high memory density, non-volatility, and lower energy requirements for data storage and retrieval. Their inherent characteristics allow them to perform certain computations directly in memory, helping to alleviate the memory wall by reducing the need for data transfer between the processor and memory.

In edge computing, where power constraints and intermittent energy harvesting cycles are the norm, traditional power management techniques like Dynamic Voltage and Frequency Scaling (DVFS) fall short of addressing the demands of eNVM technologies. DVFS operates by lowering voltage and frequency to save energy during low-load conditions, which works well in conventional digital systems. However, with emerging NVMs, this technique is less effective as these devices have unique operational requirements, such as specific voltage thresholds for write and erase operations, which cannot simply scale with frequency and voltage without risking functionality. Therefore, applying DVFS in these contexts can introduce significant instability, data integrity issues, and non-linear energy consumption, rendering it incompatible with

the reliable, efficient operation of these memory devices [71].

For edge ML accelerators, accuracy—though important—is often a secondary consideration, particularly when small gains in precision demand disproportionate increases in energy. The primary goal is to ensure efficient computation that aligns with power constraints, enabling edge devices to deliver meaningful performance within the limitations of intermittent power sources. For these applications, a high degree of energy efficiency is paramount, even at the cost of peak performance or slight sacrifices in model accuracy. In scenarios where incremental gains in precision require a substantial increase in energy, it is more advantageous to prioritize scalable energy usage to extend device uptime and reduce latency. In the following sections, we will review current state-of-the-art solutions that address the aforementioned challenges, examining both their strengths and limitations, with a focus on how this dissertation aims to address these shortcomings.

2.2 Compute-in-Memory: An alternate to Von Neumann Architectures

Compute-in-memory (CiM) [34] represents a paradigm shift in traditional computing by integrating data storage and processing within the same memory array, thereby addressing the limitations of conventional von Neumann architectures where memory and processing are separated. At its core, CiM eliminates the need for frequent data transfers between memory and the processor by enabling computations to occur directly within memory cells. This is particularly advantageous in scenarios where intensive data access and manipulation are required, as CiM minimizes the delays and energy costs associated with data movement. This fundamental change is achieved through specialized memory arrays that can perform operations such as bitwise logic, matrix multiplications, and accumulations directly within the memory, thus accelerating processing while reducing power consumption.

One of the key benefits of CiM is the significant reduction in data movement costs, a known bottleneck in both performance and power consumption for conventional ar-

chitectures. With CiM, the memory wall—where memory bandwidth lags behind computational speed—is effectively mitigated, allowing for much faster data processing. This reduced data movement enhances performance and extends the battery life of energy-constrained devices, making CiM highly suitable for edge applications.

Given ML algorithms' high data access requirements, CiM is uniquely well-suited for ML acceleration, especially at the edge. ML workloads, including operations such as matrix multiplications and convolutional neural network (CNN) computations, are data-intensive and benefit greatly from reduced memory access times. In edge ML applications, where power and latency constraints are critical, CiM offers an efficient alternative to traditional architectures, delivering the computational power needed for complex ML tasks with minimal energy overhead. Integrating memory and computation within a CiM system allows for faster, low-energy data processing, which aligns with the stringent power budgets typical in edge deployments. Compute-in-Memory (CiM) encompasses a range of implementation strategies, primarily categorized into analog and digital approaches, each offering distinct advantages tailored to various computational needs. These different "flavors" of CiM provide flexibility in optimizing for factors such as energy efficiency, speed, and computational accuracy.

2.2.1 Analog CiM

Analog CiM (see Fig. 2-2a) leverages the analog properties of memory devices to perform computations directly within the memory array. This approach is particularly effective for operations like matrix-vector multiplications, which are fundamental to many machine-learning algorithms. By utilizing the inherent physical characteristics of memory elements, analog CiM can perform these operations in a highly parallel manner with significant energy savings. For instance, analog CiM implementations can exploit Ohm's law and Kirchhoff's current law to execute weighted summations across memory cells, which is ideal for accelerating neural network computations. This method offers substantial benefits in terms of energy efficiency and throughput, as it minimizes the need for digital-to-analog and analog-to-digital conversions that are typically required in traditional digital systems.

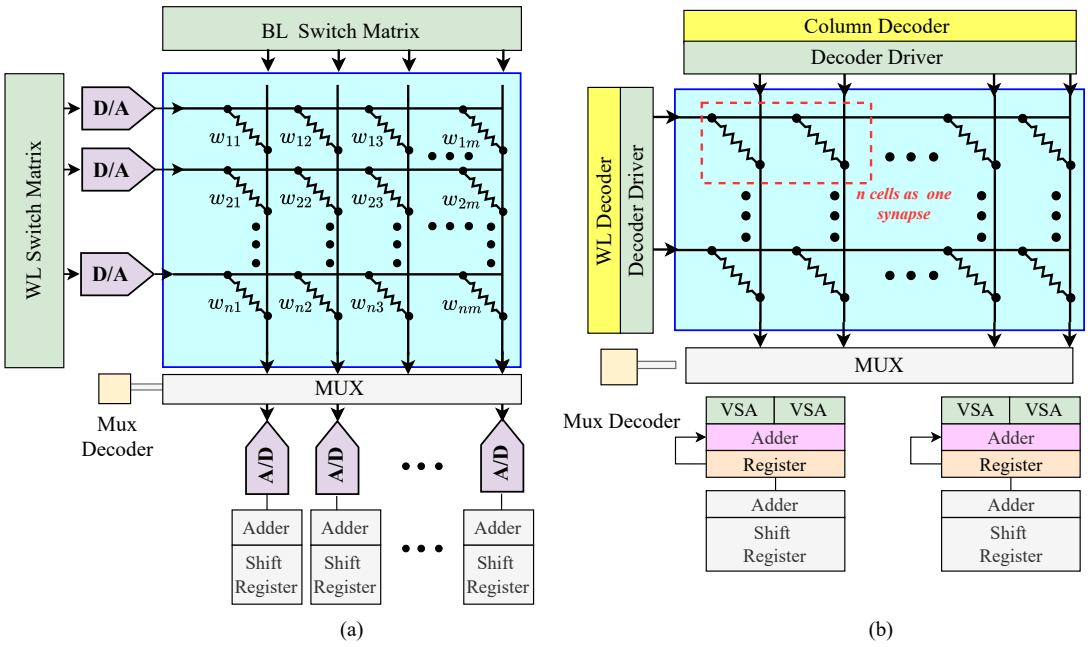


Figure 2-2: **Architectural diagram of (a) an Analog CiM, and (b) a Digital CiM.**

Ali Shafiee et al. [59] introduced the ISAAC architecture, a comprehensive framework integrating memristive crossbar arrays and digital components to form a pipelined structure, where neuron sets within each CNN layer are processed in the crossbar while intermediary values are stored in eDRAM buffers. This architecture assigns crossbars to specific layers, reducing reprogramming needs and optimizing chip area, which significantly enhances throughput, energy efficiency, and computational density in CNN tasks. Similarly, Qi Liu et al. [44] addressed Analog CiM bottlenecks, such as IR drop, with a sign-weighted 2T2R memristor array and a resolution-adjustable LPAR-ADC interface to balance accuracy and power. Additionally, researchers have explored adaptations of conventional memory cells in Analog CiM architectures. Zhengyu Chen et al. [13] introduced a 3-transistor (3T) analog memory cell (DARAM) that encodes weights as analog voltages, achieving a 10× reduction in transistor count over traditional SRAM-based cells. Further power efficiency was achieved through analog sparsity-based methods like compute-adaptive ADC skipping and weight-shifting, resulting in a 2× reduction in power consumption. However, analog CiM also faces challenges, such as sensitivity to noise, device variability, and non-idealities in analog

signal processing. These factors can affect the precision and reliability of computations, which might be a limitation for applications requiring high numerical accuracy.

2.2.2 Digital CiM

Digital CiM(see Fig. 2-2b), on the other hand, incorporates computational logic within the memory array using digital circuits. This approach can be implemented through modifications to conventional memory technologies, such as SRAM or DRAM, to embed logic gates that perform basic operations like addition, multiplication, and bitwise functions [67, 14, 65]. Digital CiM maintains the advantages of standard digital computation, such as robustness to noise and high precision, while still reducing data movement by co-locating processing and storage. Digital CiM implementations are generally more compatible with existing digital systems and fabrication processes, making them easier to integrate into current technology stacks. They offer a good balance between performance and reliability, which is crucial for applications where computational accuracy cannot be compromised.

Digital CiM architectures offer a distinct advantage in scalability and energy efficiency over analog designs, as bit vectors can be sequentially fed into the CiM tile, making digital precision more feasible. For example, Yu-Der Chih et al.[14] proposed a 6T SRAM-based, all-digital CiM macro optimized for energy-efficient MAC operations in CNNs. This design supports configurable bit-widths for both input activations (1-8 bits) and weights (4, 8, 12, or 16 bits), using bit-serial multiplication and parallel adder trees to maximize parallelism, energy efficiency, and throughput. Researchers have also focused on optimizing the read mechanisms at the memory bank level to further improve energy efficiency. For instance, Jian-Wei Su et al.[65] introduced a segmented-BL charge-sharing (SBCS) scheme for efficient MAC operations, maintaining high signal margin with low energy consumption. Supporting this approach, they developed a source-injection local multiplication cell (SILMC) to stabilize signal margin against transistor variations, along with a prioritized-hybrid ADC (Ph-ADC) for compact, low-power analog readout.

Each of these CiM approaches—analog and digital—brings unique benefits and

trade-offs. Analog CiM excels in energy efficiency and parallelism, making it well-suited for low-power, high-throughput applications like real-time image and signal processing at the edge. Digital CiM provides a more reliable and precise computational environment, which is advantageous for applications requiring exact calculations and compatibility with conventional digital systems. By leveraging the strengths of these different CiM implementations, edge ML accelerators can be tailored to meet the diverse demands of edge computing, providing efficient and scalable solutions that maximize performance while minimizing energy consumption.

2.3 Emerging Devices Facilitating Non-Von Neumann Architectures

Emerging non-volatile memory (eNVM) devices, such as memristors, phase-change memories (PCM), and magnetic devices, are drawing significant attention in edge machine learning for their potential to transform computation at the hardware level. These devices offer distinct advantages, including non-volatility, multi-bit storage precision, and reduced power consumption, making them highly suitable for energy-efficient, low-power edge applications. Recent works [59, 20, 67, 44] have demonstrated edge ML accelerators that leverage eNVM devices as both storage and computational elements, highlighting their versatility in these architectures.

These eNVM devices operate based on diverse physical principles. For example, RRAM (Resistive RAM) utilizes resistive switching by forming and breaking conductive filaments within a dielectric layer; PCM (Phase Change Memory) encodes data by transitioning between crystalline and amorphous states in a chalcogenide material; MRAM (Magnetoresistive RAM) employs magnetic tunnel junctions, storing data through the alignment of magnetic orientations; and FeFET (Ferroelectric FET) relies on electric field-induced polarization states within a ferroelectric layer, allowing data to persist without power. The choice of device depends on the system requirements, such as energy efficiency, storage precision, and endurance, for the

targeted ML acceleration.

Despite their promising features, deploying emerging devices in large-scale, practical applications poses several challenges, including device-to-device variations, limited fabrication yields, endurance constraints, leakage power, and reliability concerns over prolonged use. A primary barrier to widespread adoption is the limited compatibility of these devices with current CMOS fabrication techniques, which restricts scalable integration across various edge platforms. Overcoming these challenges is crucial for realizing the full potential of eNVM devices in edge ML applications. Depending on the device type and fabrication process, the severity of these issues can vary significantly. Comprehensive reviews by Je-Min Hung et al. [32] and An Chen [12] discuss these benefits and challenges in detail, particularly for CiM-based applications, as summarized in Table 2.1.

Table 2.1: Qualitative Summary of eNVM devices based on system metrics

Device	Energy Consumption	Endurance	Device Variations
RRAM	Medium (ns to µs write time)	Medium (10^8 to 10^12 cycles)	High (Stochastic behavior, intrinsic variability)
PCM	High (2 ns to 200 ns write time, high switching power)	High (10^8 to 10^15 cycles)	Medium (Disturbance issues, phase-change consistency)
MRAM	Low (1.3 ns to 25 ns read time)	Medium-High (approx. 10^8 cycles)	Low-Medium (Patterning, magnetic stability)
FeFET	Low (Field-driven, low-power)	Low-Medium (Reliability issues)	Medium-High (Charge trapping, parasititic effects)

2.4 Spiking Neural Networks: Insights from Biological Models

Biological systems, such as the brain, are capable of performing complex computations with very little power, far surpassing traditional artificial computing systems in this regard [15]. This energy efficiency, combined with the ability to manage large

amounts of sensory data and extract critical features for decision-making, has inspired researchers to explore computational models that emulate these biological processes. The focus is on creating systems that can replicate the brain's ability to operate in noisy, variable environments while minimizing energy consumption, making this approach particularly attractive for edge machine learning applications.

One of the most promising avenues in this area is the development and implementation of Spiking Neural Networks (SNNs). Spiking neural models have garnered considerable attention recently due to their efforts to mimic the energy efficiency observed in biological systems [26]. These models encode information in sparse dimensions, offering enhanced energy efficiency and facilitating efficient feature representation by minimizing redundancy. These networks mimic the way neurons in the brain communicate via spikes, transmitting information only when a certain threshold is reached, which conserves energy compared to continuous communication in traditional neural networks. Various research works have demonstrated the superior energy efficiency of SNNs for specific tasks. For example, SNNs have been successfully applied to image recognition [70] and sensory processing tasks, where they not only matched the performance of conventional deep learning models but also did so with significantly lower power requirements. Neuromorphic hardware platforms such as Intel's Loihi [16] and IBM's TrueNorth [4] have showcased these benefits in practical implementations, further proving their potential in energy-constrained environments like the edge.

2.4.1 Fundamental spiking encoding schema

In the early stages of spiking neural network (SNN) research, efforts primarily focused on mimicking the function of neurons found in biological systems, especially vertebrates. This bio-inspired approach introduced foundational neuron models, such as Integrate-and-Fire (IF), Leaky Integrate-and-Fire (LIF), and time-to-first spike (TTFS), laying the groundwork for encoding methods that capture neuron activation in response to stimuli. Initially, neural encoding aimed to replicate neuronal behavior by using rate encoding, where information is conveyed through the average firing

rate of spikes over a given time. This approach aligned well with basic biomimetic goals, yet it presented limitations in capturing the full potential of temporal dynamics observed in biological neurons. Broadly speaking, various encoding schemas can be classified irrespective of their biological mechanism into the following categories:

Rate Encoding: In rate encoding, information is conveyed through the average firing rate of neurons over a set period, with the critical information encoded in the density of spikes within that interval. The timing of individual spikes can be either deterministic or random, but it is the spike rate that primarily represents the encoded stimuli. Although computationally efficient, rate encoding is limited in applications requiring precise temporal dynamics. It aligns with traditional neural network methodologies but does not fully exploit the low-latency, event-driven capabilities inherent to SNNs.

Temporal Encoding: Temporal encoding techniques, such as time-to-first spike (TTFS) and spike-time-dependent encoding, capture information in the precise timing of individual spikes rather than in their overall rate. This approach enables faster, more energy-efficient communication by focusing on the exact moment of each spike. For example, TTFS encodes data in the time interval between a stimulus and the first spike, facilitating low-latency processing and effectively capturing temporal patterns. Temporal encoding supports biologically realistic processing features, such as phase locking and temporal synchrony. However, the precision required in spike timing demands a more robust implementation, as variations in these intervals can lead to sub-optimal information processing.

Population Encoding: Instead of relying on a single neuron, population encoding uses a group of neurons to represent information, with each neuron responding to different features or aspects of the data. This approach mirrors the organization of biological sensory systems, where diverse neuronal populations work together to process complex stimuli. Population encoding is especially effective in tasks requiring high-dimensional representations, offering robustness against noise and variation in individual neuron responses.

2.4.2 Evolution of Δ class neurons

Recently, we have seen a paradigm shift in encoding techniques, wherein the focus has shifted from biomimicry to emulating certain functional characteristics observed within biological neurons. The evolving landscape of temporal encoding techniques exhibits a significant divergence, marked by methodologies relying on the differential amplitude between consecutive feature samples in a time series format. This has led to sparser schemes such as Delta modulation (Δ) and Sigma-delta ($\Sigma\Delta$) encoding. These schemes enable feature communication between neurons only when they exceed a certain threshold, leading to even sparser feature representations between layers within a network. These spiking encoding schemes have shown notable effectiveness in fields like image and audio processing. For instance, Ilya Kiselev et al. [35] implemented a biologically inspired 64 $\times$ 2 channel silicon cochlea front end for stereo audio sensing, employing asynchronous Delta (Δ) modulation to encode analog-filtered signals into sparse event streams. This system extracted features through bandpass filters (BPFs) scaled across the human auditory range (8 Hz to 20 kHz), achieving efficient, event-driven outputs with a power consumption of only $55\mu W$ at a rate of 100k events/s.

The Delta neuron class has also been impactful in image and video processing, especially where real-time response is needed within strict energy constraints. Researchers at Dayton Engineering Advanced Projects Lab [27] developed a neuromorphic system for real-time vision-based autonomous drone navigation using Intel's Loihi 2 processor. Their approach utilized a spiking Sigma-Delta Neural Network (SDNN) to directly process high-resolution camera data onboard, generating control actions in real time. Leveraging Loihi 2's spiking capabilities, the SDNN achieved robust, energy-efficient navigation under low size, weight, and power (SWaP) constraints, emulating sparse, asynchronous processing similar to human perception. Similarly, Waseem Sharif et al. [60] introduced a spiking SDNN for enhancing spatial-temporal resolution in event cameras, aimed at improving real-time visual processing. To address the challenges of low resolution and sparse data typical of event cameras,

their method incorporated binary spike integration with a spatiotemporal learning mechanism for optimized spatial and temporal feature extraction. Evaluations on datasets like NMNIST, CIFAR10-DVS, and ASL-DVS showed a 17x improvement in event sparsity, with greater operational efficiency than traditional ANNs.

2.5 Barriers to Designing Hardware for Edge-based Neural Network Accelerators

Mapping ML workloads to edge accelerators presents significant challenges, largely due to the limitations of edge hardware in memory, computational power, and energy efficiency. Many ML models, like Convolutional Neural Networks (CNNs) and Deep Neural Networks (DNNs), are designed for powerful cloud environments but are often tailored for edge deployment through techniques like model compression and quantization. However, these adaptations come with trade-offs in accuracy, as reducing precision (e.g., INT8 or INT4) can lead to information loss. To address edge-specific needs, unique models such as Spiking Neural Networks (SNNs) and Binary Neural Networks (BNNs) have also been developed, specifically optimized for energy efficiency and reduced memory usage. Despite their potential, these models often require non-standard training approaches, and developing robust training techniques remains challenging. Successfully implementing edge AI requires extensive design space exploration, balancing trade-offs in energy efficiency, performance, precision, and latency to achieve optimal results.

Furthermore, most edge devices deployed today are inference-only, meaning they process data but cannot learn or train in real time, limiting the adaptability of edge AI in dynamic environments. Without continual learning capabilities, these models cannot adapt to changing conditions, such as variations in lighting or temperature, which may impact data distributions and, consequently, model performance. To enable effective edge AI, efforts are needed from both sides: algorithm developers must design models specifically suited to edge constraints, while hardware engineers should create

customized architectures that accelerate specific model types. Additionally, limited memory bandwidth on edge hardware makes efficient memory management crucial, underscoring the need for cohesive, hardware-aware model design that achieves real-time, resilient performance in diverse conditions.

Developing hardware for edge-based CNN accelerators faces several critical barriers. First, power and energy constraints are paramount, as edge devices often rely on limited power sources, making it challenging to achieve high performance without sacrificing energy efficiency. Most current ML architectures for edge devices aim to balance accuracy with either performance or energy efficiency. For example, Longwei Huang et al. [31] introduced a precision-scalable RISC-V-based Deep NN processor optimized for extreme edge platforms, achieving a balance between energy efficiency and accuracy scalability. Their processor supports diverse precision levels (2-bit to 16-bit fixed-point) for inference, enhancing accuracy while preserving data privacy. By reusing multi-precision integer multipliers and optimizing FPGA resource allocation, they achieved a 1.1 - 14.6× increase in energy efficiency and a 16.5× boost in floating-point throughput for on-device learning compared to previous architectures.

However, while focusing on accuracy vs performance is essential, the overlooked additional hardware cost is substantial. Precision-scalable designs demand reconfigurable hardware and an increased silicon footprint, but these resources may see limited utilization. Precision scalability is inherently linked to computational demand, which, at the edge, is often sporadic. Thus, while the logic of addressing high computational loads may justify additional hardware, the practical, intermittent demands at the edge highlight a trade-off between hardware cost and actual utility.

Second, memory limitations are significant; edge devices lack the extensive memory resources typical in cloud or high-performance computing, complicating handling large CNN models and frequent memory accesses. ML algorithms are inherently memory-intensive, demanding substantial data storage and retrieval during training and inference. Modern ML models, especially deep learning networks, use large matrices and tensors to represent weights, activations, and gradients. As discussed in Section 2.2, CiM offers a novel and orthogonal alternative to conventional Von Neu-

mann architectures, which have traditionally served as edge hardware accelerators.

Mark Horowitz [30] estimated that, in modern superscalar processors, over 50% of the processor's die energy cost comes from cache and memory management. This issue is even more pronounced when mapping CNN/DNN workloads to an edge device, where innovative techniques are essential, both in ML algorithm design and hardware data flow management. For example, Arnab Raha et al. [54] developed FlexNN, a framework implementing a novel sparsity-based acceleration logic. By analyzing the sparsity in activations and weights, FlexNN bypasses unnecessary computations, reducing memory access and data movement costs, achieving a speedup of 1.8×–3.3× for dense workloads and approximately 1.8× for semi-sparse workloads.

The brain processes vast amounts of information with minimal energy, inspiring the development of computing systems that mimic this efficiency. SNNs offer an event-driven processing approach, where computation is triggered only when necessary, inherently supporting energy-efficient processing. This feature is highly beneficial for edge computing, where power constraints are critical. As described in Section 2.4, SNNs excel in representing information sparsely, making them a promising solution for feature extraction at the edge, where events tend to be sporadic. The ability of spiking models to encode information with minimal redundancy and reduced dimensionality is particularly valuable in such contexts.

The practical implementation of SNNs remains challenging, with reliable training often proving cumbersome and leading researchers to adopt proxy model approximations. Unlike conventional Artificial Neural Networks (ANNs), which benefit from backpropagation as a champion algorithm, SNNs lack a similarly robust training method. Common approaches include localized training algorithms that leverage neuron-level plasticity, though these often achieve lower accuracy on standard datasets, and ANN-to-SNN conversion, which, while effective, is not a true SNN approach. This conversion method trains the model as an ANN and then approximates activation functions for SNN adaptation.

For instance, Xiangwen Wang et al. [68] extensively reviewed supervised learning algorithms for SNNs, assessing them across five key criteria: spike train learning abil-

ity, offline and online processing support, locality of learning rules, stability of optimal solutions, and compatibility with spiking neuron models. Some algorithms are limited to learning single spikes, while others are capable of handling entire spike trains, which makes them more versatile for complex tasks. Certain methods support both offline and online learning, though others are restricted to one mode, limiting their flexibility. While single-layer and some recurrent SNNs maintain locality, multilayer feed-forward and certain recurrent SNNs lack this property, which impacts scalability. Stability in achieving optimal solutions varies across methods, with some providing reliable performance and others lacking consistency. Additionally, certain algorithms, such as gradient descent-based approaches, are compatible only with specific neuron models, while others are more widely applicable.

In today's edge ML landscape, where practical applications and relevant datasets demand robust performance, the theoretical energy efficiency of SNNs is not always fully realized. This limitation often arises from the dataset characteristics and training procedures required. For instance, Lei Deng et al. [17] examined SNN performance across datasets, showing that SNNs perform well on simpler, ANN-oriented datasets like MNIST, where they maintain accuracy with lower compute costs. On more complex datasets, such as CIFAR-10, SNNs struggle to match ANN accuracy and may require longer processing times. SNNs excel on SNN-specific workloads, such as DVS-CIFAR-10, where sparse activity patterns enable lower computation costs. In sum, while SNNs are effective on simpler or SNN-specific datasets, they are less suitable for complex, ANN-oriented tasks.

Moreover, the irregular, spike-based communication patterns in SNNs pose challenges for mapping onto conventional digital hardware, which is optimized for continuous data flows. Current state-of-the-art hardware like Loihi [16] and SpiNNaker [49] are digital emulations of biological neurons, designed to implement SNN models on digital platforms. However, these digital implementations often limit the full energy efficiency potential that SNNs could theoretically achieve. Additionally, designing neuromorphic hardware that captures the complex dynamics of biological neurons while remaining efficient in silicon demands innovative solutions. Key challenges such

as scalability, precision in spike generation, and power management during spike processing remain active research areas. Bridging the gap between SNNs' energy-efficient promise and the practical constraints of hardware is a primary focus for researchers aiming to make these models viable for real-world edge applications.

Chapter 3

Asynchronous Stream Computing: Computing Through Temporal Streams

In applications where energy efficiency takes precedence over accuracy, Stochastic Computing offers an energy-efficient solution, because arithmetic and logical operations can be carried out with simpler gates. Engineers also realized that accuracy and precision can often be traded off for other metrics, such as power and/or cost, depending on the requirements of the application [3]. The overall accuracy of these operations is a knob that can be adjusted by varying the length of the stochastic number [19]. The fact that the accuracy of the system is related to the length of the stochastic number in a conventional stochastic system leads to a trade-off between accuracy and throughput. Stochastic Computing streams/numbers can be generated by employing Random Number Sources, Linear Feedback Shift Registers (LFSRs), noisy emerging technologies, etc. [22, 72]. Asynchronous streams can be created through a Pulse Density Modulation (PDM) method, and are often generated through clockless Sigma-Delta ($\Sigma\Delta$) Modulator circuits [36].

Asynchronous Stochastic Computing (ASC) is an alternative to the conventional stochastic methodology introduced by Gaines [21], where an analog value/ event can be represented as a stochastic stream, p, and is represented by the average duration

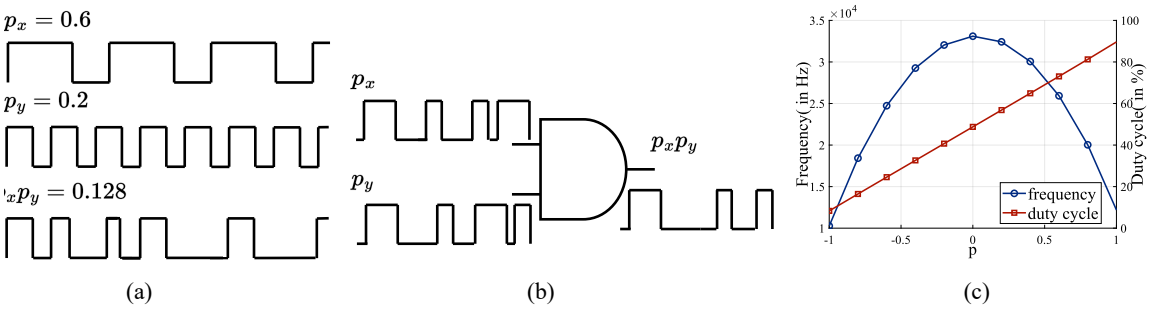


Figure 3-1: **Signal properties of (a) different value ASC streams (b) Multiplication operation using AND gate, and, (c) Temporal characteristics with relation to the input value, *p*.**

for which the stream takes a high (V_h) voltage level. Fundamentally, ASC encodes the incoming analog information in the time domain, and the amplitude of the signal correlates with the stream's instantaneous frequency, f and duty cycle, δ, through the relations below [23]:

$$\frac{f}{f_c} = 1 - p^2 \qquad \delta = \frac{1+p}{2} \tag{3.1}$$

, where f_c is the natural frequency of the modulator when the input is zero and p is the normalized value of the input to the maximum value. Fig. 3-1a exhibits transient signals that represent different values of p. Arithmetic operations such as multiplication can be implemented through simple logic gates such as AND, as depicted in Fig. 3-1b. From a feature encoding perspective, any feature vector can be mapped to a set space of $p \in [-1 \ +1]$, which translates to an output frequency and duty cycle, as depicted in Fig. 3-1c.

3.1 Energy-Efficient Computing: A Scaling Perspective

To understand the energy and frequency dynamics involved while generating asynchronous streams, it is necessary to understand the fundamental concepts of an Asynchronous Continuous Time $\Sigma\Delta$ modulator as shown in Fig.3-2a. The modulator operates on the difference between the input source, I_{in}, and the output signal V_{out}. To operate on the difference, V_{out} is represented in the current domain by a switched

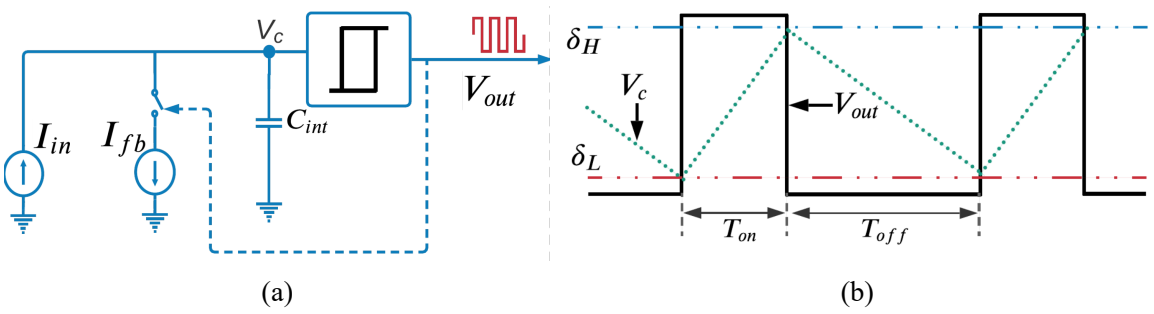


Figure 3-2: **(a) Conceptual diagram for ΣΔ modulator, (b) Timing diagram of the transient signals to determine the instantaneous frequency of the stream.**

current source, I_{fb}, and the difference is integrated on the capacitor node, C_{int}. The buildup of the voltage, V_c causes the output of the Schmitt trigger to switch when the switching threshold of the trigger is reached. The output pulse, V_{out}, has a constant amplitude, an instantaneous output frequency (f), and duty cycle (δ). These temporal parameters depend on the input signal dynamics, thus characterizing the stream as discrete in amplitude but continuous in time. The relation between f, δ and the input value is described in Eq. 3.1.

3.1.1 Energy & Frequency Model

In the energy model for the ΣΔ modulator, three primary components are used to calculate the energy dissipation: energy dissipated to the load (E_L), capacitive energy dissipation (E_{int}) and energy required to drive the feedback current mirror circuit (E_{fb}) and account for all the dynamic and static energy dissipation within the module. The frequency of the stream is set by the amplitude of the input signal, it would imply that the number of transitions and switching activity would be dependent on the input value. Therefore, the energy required for generating a single output stream pulse is the energy required per stream transition. The expressions for each of these components are described below:

$$E_{int} = C_{int} * (\delta_H^2 - \delta_L^2)$$

where δ_H, and δ_L are the upper and lower switching threshold voltages of the Schmitt trigger within the $\Sigma\Delta$ modulator.

$$E_L = C_L * V_{DD}^2$$

$$E_{fb} = V_{DD} * I_{fb} * \frac{1}{f}$$

Therefore, the total energy dissipation per transition would be;

$$E_{tot} = E_{int} + E_L + E_{fb} \tag{3.2}$$

where C_L, V_{DD} are the load capacitance and supply rail of the $\Sigma\Delta$ modulator circuit, respectively. To calculate the instantaneous output frequency of the stream, the transitional period of the stream must be calculated. For an analog input, I_{in}, there would be a time-varying voltage buildup of V_c on the capacitor, C_{int}. The timing diagram of the stream is depicted in Fig.3-2b and the respective transitional periods are derived below:

$$T_{on} = \frac{2 * C_{int} * \delta_{hys}}{I_{fb} + I_{in}}$$

$$T_{off} = \frac{2 * C_{int} * \delta_{hys}}{I_{fb} - I_{in}}$$

$$\implies T = 2 * C_{int} * \delta_{hys} \left[\frac{2I_{fb}}{I_{fb}^2 - I_{in}^2} \right]$$

$$f = \frac{I_{fb}^2 - I_{in}^2}{4 * C_{int} * \delta_{hys} * I_{fb}} \tag{3.3}$$

A key deduction that can be made from (3.3), is the identification of knobs that can be utilized to understand how effectively can one scale the stream throughput while following a minimum energy cost curve.

3.1.2 Sensitivity Analysis for efficient scaling

To analyze the sensitivity of energy to the instantaneous frequency due to a change in a design knob, it is necessary to identify knobs that would critically affect the design space. By carrying out a sensitivity analysis for each knob, it is possible to understand the trade-off one would achieve regarding energy and frequency metrics. A design knob, with a larger sensitivity, can be a more effective knob in optimizing the circuit for a given region. To be chosen as a design parameter, they need to be uncorrelated and scalable. This allows us to explore the effect each parameter exclusively has on the metrics under consideration. The sensitivity of a parameter, X, can be defined as [36];

$$S_x(X) = \frac{\frac{\partial P_{tot}}{\partial x}}{\frac{\partial D}{\partial x}}\Bigg|_{x=X} \tag{3.4}$$

In our analysis, the design optimization is carried out by employing I_{fb}, C_{int}, and δ_{hys} as the knobs. To normalize the design space exploration, we consider a reference design with a constant output frequency, F_{ref}, and the corresponding energy dissipation is calculated for that reference point. Every other design point within the space exploration is considered an incremental/decremental point regarding the reference point.

In the case of C_{int}, a higher value results in a larger time constant and therefore, causes V_c to reach the switching threshold voltages at a slower rate, resulting in a lower frequency. δ_{hys} has a similar effect wherein increasing/decreasing the parameter, effectively affects the transitional period lengths and hence lowers/increases the frequency. This is reflected in (3.5) & (3.6) and shows that their respective sensitivities have diminishing returns as the value of the two design knobs are increased. The diminishing returns for C_{int} can be attributed to the fact that the reduction in E_{int} as the frequency increases is also reflected by the negative value of sensitivity in the equations. In the case of δ_{hys}, there is a quadratic effect and the relative drop

in sensitivity occurs at a much faster rate. The dynamic range of δ_{hys} is the limiting factor that prevents it from scaling the frequency across the entire range. However, this can be attributed to the fact that the scalable nature of δ_{hys} is often limited by the supply voltage and is reflected in the analysis. The deviation in energy per transition due to a change in C_{int} can be calculated as;

$$\frac{\partial E_{tot}}{\partial C_{int}} = \delta_{hys}^2 + 2\delta_L\delta_{hys} + \frac{4\delta_{hys}}{1-p^2}$$

The variation in frequency due to a change in C_{int}, is formulated below;

$$\frac{\partial f}{\partial C_{int}} = \frac{-1(I_{fb}^2 - I_{in}^2)}{2I_{fb}\delta_{hys}C_{int}^2}$$

Therefore, the sensitivity of the total energy per transition to the frequency due to a change in C_{int} is derived below;

$$S_{C_{int}} = \frac{\dfrac{\partial E_{tot}}{\partial C_{int}}}{\dfrac{\partial f}{\partial C_{int}}} = \frac{4I_{fb}(C_{int}\delta_{hys})^2\left[\dfrac{\delta_{hys}}{2} + \delta_L + \dfrac{2}{1-p^2}\right]}{-1(I_{fb}^2 - I_{in}^2)} \tag{3.5}$$

Similarly, the sensitivity of the other knobs can be derived and is calculated below;

$$S_{\delta_{hys}} = \frac{\dfrac{\partial E_{tot}}{\partial \delta_{hys}}}{\dfrac{\partial f}{\partial \delta_{hys}}} = \frac{4I_{fb}(C_{int}\delta_{hys})^2\left[\delta_{hys} + \delta_L + \dfrac{2}{1-p^2}\right]}{-1(I_{fb}^2 - I_{in}^2)} \tag{3.6}$$

$$S_{I_{fb}} = \frac{\dfrac{\partial E_{tot}}{\partial I_{fb}}}{\dfrac{\partial f}{\partial I_{fb}}} = \frac{8I_{fb}(C_{int}\delta_{hys})^2 V_{DD}}{-1(I_{fb}^2 - I_{in}^2)^2\left[\dfrac{1}{I_{in}^2} + \dfrac{1}{I_{fb}^2}\right]} \tag{3.7}$$

The sensitivity of energy to the relative output frequency due to a change in I_{fb} as

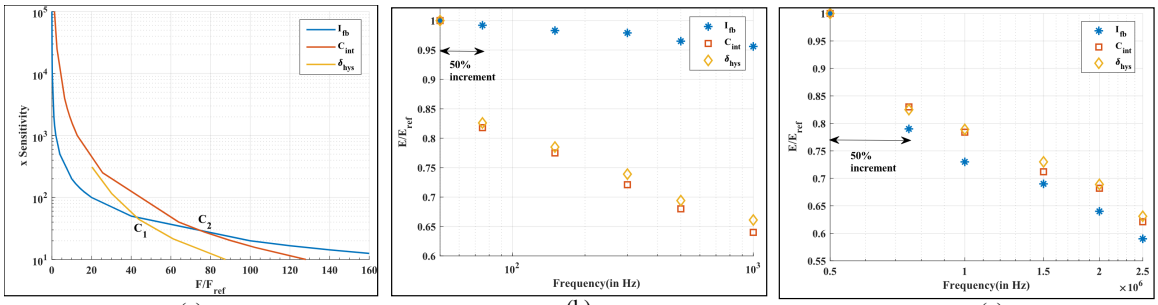


Figure 3-3: **(a)Sensitivity analysis of design knobs vs frequency scaling, Single Knob Optimization scenario for (b) low-frequency region, and (c) high-frequency applications.**

depicted by (3.7), exhibits a plateauing effect, where the relative change in energy as the frequency is increased is almost constant, owing to the static energy costs associated with the feedback current mirror. This implies that I_{fb} sensitivity drops from its maximum value at the origin at a much lower frequency and then plateaus out, and the relative decrease in sensitivity is much lower than that of the other two design knobs. However, a critical cross-over point in the design space is when the diminishing returns of δ_{hys} & C_{int} drops below the sensitivity of the I_{fb} at points C_1 and C_2, respectively, as seen in Fig. 3-3. This implies that, as the frequency scales, I_{fb} would prove to be a more efficient design knob after the respective cross-over points.

Single Knob Optimization: We utilize our insights gained from the sensitivity analysis to carry out a case study to show how various knobs can be effective in different regions within the frequency scaling domain. To understand the energy and frequency dynamics, it is necessary to understand the scenarios that would occur during a practical design process. In most design approaches, it is often impractical to have a dynamic tuning property for every knob within the circuit. Therefore, depending upon the application and the required frequency scalability, it would be highly beneficial to determine the knob that would be the most effective before design and would result in a single knob optimization problem. To analyze single-knob optimization, we consider two frequency scaling applications targeting sub-1 kHz applications, such as body temperature and blood pressure monitors, as depicted in Fig.3-3b. Experimental simulations reveal that when C_{int} and δ_{hys} are used as separate scaling knobs, the relative energy savings are approximately 18% with a 50%

increment in frequency, validating their effectiveness at lower frequencies. Conversely, when I_{fb} serves as a single knob, the energy savings are less than 4%. In another scenario, focusing on intravascular ultrasound signal processing applications operating in the 100's of kHz to MHz range (Fig.3-3c), using I_{fb} as the scaling knob is more effective, with approximately 21% savings compared to the 17.5% savings with C_{int} and δ_{hys}, showcasing its efficacy beyond the inflection point highlighted in Fig. 3-3a.

3.1.3 Scalable ASC: Circuit Implementation

The insights gained from the sensitivity analysis and single knob optimization problem, are validated by designing and simulating a scalable throughput ΣΔ Modulator. The practicality of the optimization strategy is proved by designing a scalable arithmetic unit that is capable of MAC operations within a microarchitecture system. The effect of scaling of throughput on the overall accuracy of the operation is also analyzed. The ΣΔ modulator circuit and ASC arithmetic modules were designed using the 65nm TSMC LP process, and the simulated results utilizing Cadence Virtuoso are depicted in this section.

3.1.4 Variable Stream Rate ΣΔ Modulator

To demonstrate the effectiveness of each knob and how the sensitivity analysis results would be implemented in a practical application, a variable stream rate ΣΔ Modulator has been designed, as shown in Fig. 3-4. The three design knobs, I_{fb}, C_{int} and δ_{hys}, have dynamic scaling capabilities through programmable elements, as seen in Figure 3-4. By properly sizing a current mirror's transistor, M_P and utilizing switchable elements, $M_{N,sw}$ a combination of I_{fb} values can be generated ranging from 1 nA - 25 nA. Similarly, a programmable Metal-insulator-Metal (MiM) capacitor bank has been designed to scale C_{int} from 50fF to 300fF. The parasitic and switching noise components that would be introduced by utilizing a bank of switchable capacitors can be minimized through proper sizing and allocating a reset period based on the worst-case settling time required for the bank before the generation of an ASC stream.

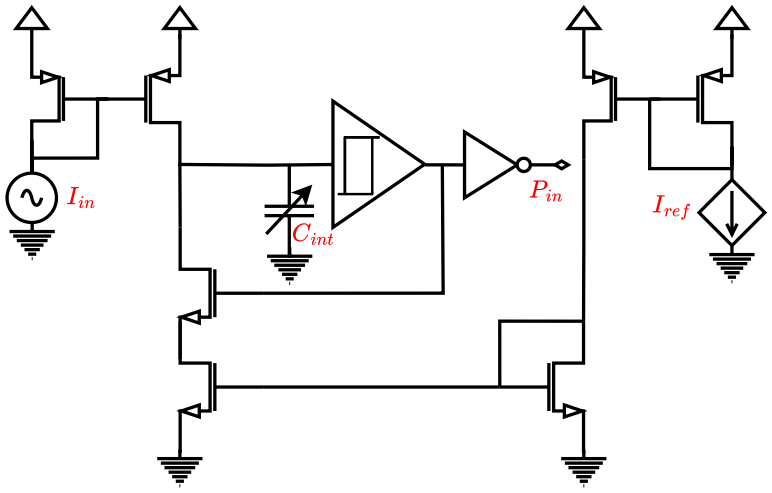


Figure 3-4: **Schematic diagram of programmable ΣΔ Modulator**

To implement a variable hysteresis loop for the Schmitt trigger, the feedback strength of the circuit can be altered thereby varying the upper and lower switching threshold. The resulting hysteresis loop of a 4-bit programmable Schmitt trigger is shown in Fig.3-4. The frequency and duty cycle characteristics of the scalable modulator over the dynamic input range of $0 - 0.8p$, depict the scaling effect and have been plotted for throughput ranges from 2kHz to 100 kHz.

3.1.5 Scalable ASC Arithmetic Unit

In our previous works, it has been proven that arithmetic operations such as multiplication can be carried out by employing simple logic gates (AND/XOR) [25] and addition of streams can be implemented in a current-driven and integration method [24]. As the adder operates on a similar current integration principle, a secondary level of Pareto optimization can be carried out. Multiply-Accumulators (MAC) units often account for more than 500 Million MAC (MMAC) operations in IoT edge systems [53]. This provides a significant incentive to design scalable MAC units that can be energy-efficient across a wide throughput range. The designed arithmetic unit can be utilized for MAC operations, and the computational throughput of the system can be altered by scaling the knobs. The implemented schematic model of the ASC arithmetic unit is depicted in Fig. 3-5a.

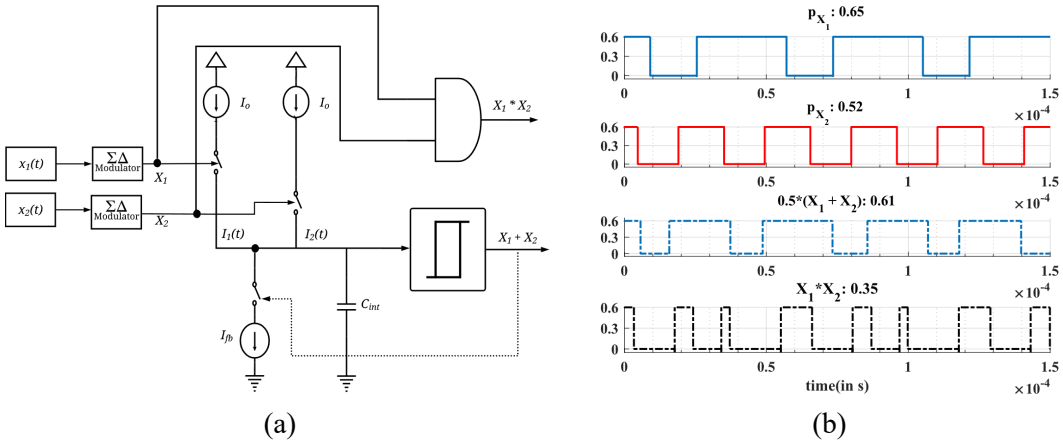


Figure 3-5: **Schematic diagram of the implemented arithmetic unit in ASC.**

Another key metric in the computation of streams is the operation's accuracy, which depends on how uncorrelated the streams are. The measure of non-correlation between two streams is defined by the Stochastic Computing Correlation (SCC)[25] for streams X_1 and X_2 and is calculated using (3.8) and a transient simulation depicting the results of computation for two streams is plotted in Fig. 3-5b.

$$SCC(X_1, X_2) = \begin{cases} \frac{p_{X1 \wedge X2} - p_{X1} p_{X2}}{min(p_{X1} p_{X2}) - p_{X1} p_{X2}}, & \text{if } p_{X1 \wedge X2} \geq p_{X1} p_{X2} \\ \frac{p_{X1 \wedge X2} - p_{X1} p_{X2}}{p_{X1} p_{X2} - max(p_{X1} + p_{X2} - 1, 0)}, & \text{otherwise} \end{cases} \tag{3.8}$$

The non-correlation between streams can be varied through the design knobs, which in turn affect the natural frequency of the modulator. From a micro-architectural level, the optimization strategy is to ensure the scaling of the average throughput of the system, in constraint to a minimum energy penalty. This would ensure an energy-efficient scaling arithmetic unit that can be deployed within distributed computing systems.

From an application's perspective, the validity of the sensitivity analysis is proved by implementing the above-mentioned scalable arithmetic unit. A reference design point with an average throughput of 100Hz was chosen for both arithmetic operations. By scaling each design knob, individually their optimal throughput range has been determined. The non-coherence of the streams is guaranteed, by appropriately

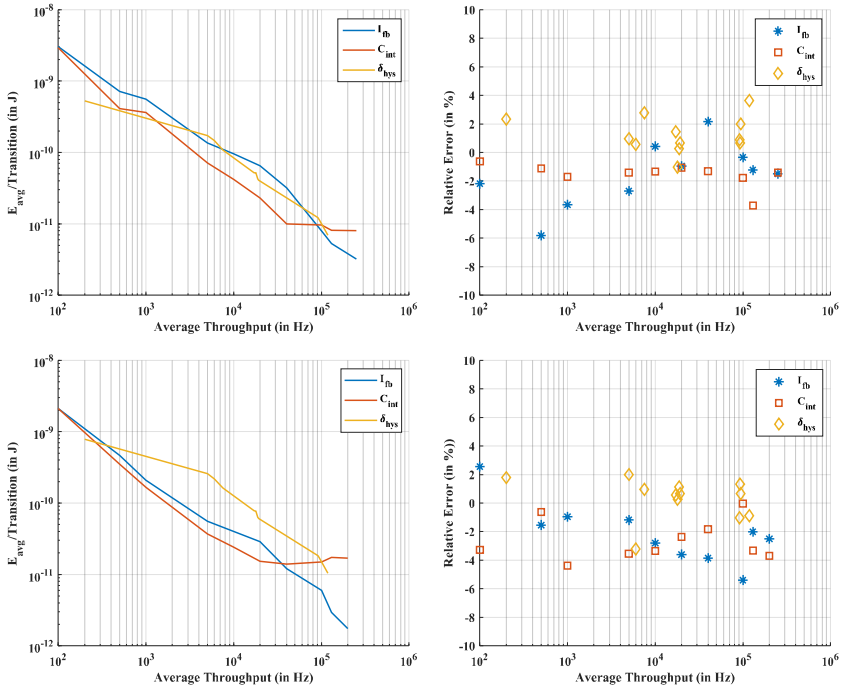


Figure 3-6: **Design space exploration of ASC arithmetic unit depicting a)** $E_{avg}/transition$ **for addition, b) Relative Error for addition, c)** $E_{avg}/transition$ **for multiplication, d) Relative Error for multiplication as a function of throughput**

programming the knobs to reflect variations in f_c. While utilizing I_{fb} as a knob, one must ensure a constant ratio of p, to validate a fair comparison. This can be done by scaling the input value in accordance with the change in I_{fb}. It must be noted that, while the accuracy of the operations depends on the non-coherence between two streams, there also lies a dependency on the actual value of the input itself [25]. This error can be attributed to the noise introduced as the input value instantaneous approaches f_c, resulting in limit cycle frequency components. The simulated results of the arithmetic unit are depicted in Fig. 3-6.

As it can be seen from Fig. 3-6 a) and c), C_{int} and δ_{hys} prove to be the preferred design knob for low-frequency applications. The savings achieved by utilizing I_{fb} as a knob, is evident at frequencies from 100KHz and upwards. The error analysis depicted in Fig. 3-6 b) and d), verifies that the implemented solution is energy-efficient whilst ensuring that the relative error is limited within a desirable range, ±4%, for our

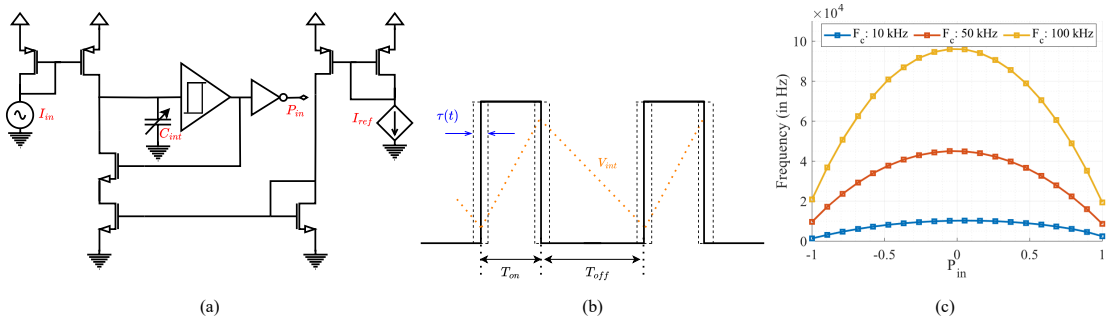


Figure 3-7: **(a) Schematic representation of ASC stream generation circuit, (b) effect of jitter on ASC stream characteristics, and, (c) Scalable frequency characteristics attained with implemented modulator circuit.**

current analysis. Scaling an arithmetic unit's throughput, without compromising the overall accuracy, establishes a practically feasible solution that can be designed for stochastic computing-based micro-architectures.

3.2 Precision in Information Encoding via ASC

.

3.2.1 Analytical Model for ASC Precision

Consider an ASC stream $x(t)$ generated by a $\Sigma\Delta$ modulator (Fig. 3-7a) for a static input p_k, and its Fourier representation can be formulated as:

$$x(t) = p_k + 4\delta_k \sum_{n=1}^{+\infty} \text{sinc}(n\delta_k) \cos\left(2\pi \frac{n}{T_k} t\right) \tag{3.9}$$

where δ_k and T_k represent the duty cycle and period of the stream. To model the precision of ASC streams, we analyze how noise, such as random circuit variations, flicker, and thermal noise, manifests during encoding. In this work, jitter is the primary metric used to evaluate encoding robustness. Assuming Gaussian-distributed jitter, $\tau(t) \sim \mathcal{N}(0, \sigma^2)$, it introduces pulse-to-pulse variations in duty cycle and frequency. Let $\tau(t)$ represent the jitter (Fig. 3-7b), affecting both the duty cycle and frequency as:

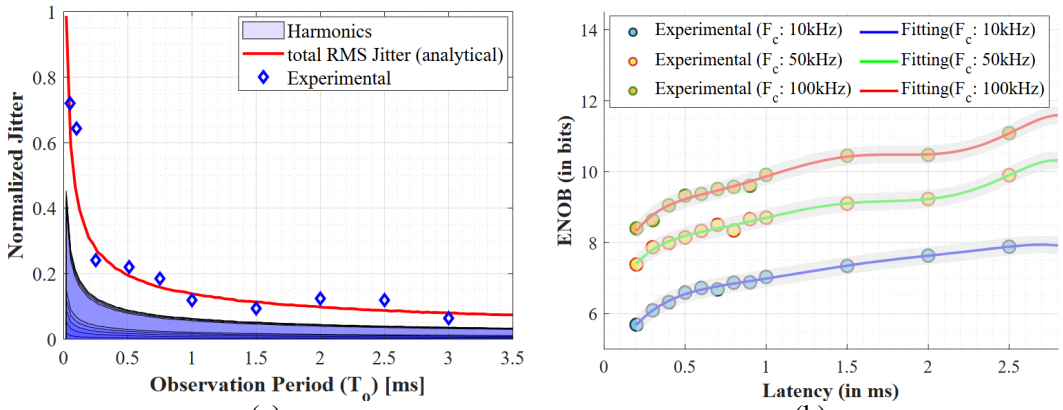


Figure 3-8: **(a) Analytical and experimental plot depicting how jitter scales vs T_o (b) ENOB vs latency plots for different $\Sigma\Delta$ modulator frequencies.**

$$\delta_k^{\text{jitter}}(t) = \delta_k + \Delta\delta_k(t), \quad f_k^{\text{jitter}}(t) = f_k + \Delta f_k(t)$$

where $\Delta\delta_k(t)$ and $\Delta f_k(t)$ are jitter-induced variations. Jitter also modulates the phase and amplitude of Fourier components, affecting the $\text{sinc}(n\delta_k)$ terms, leading to:

$$\begin{aligned} x_k^{\text{jitter}}(t) \approx p_k + 4\delta_k^{\text{jitter}}(t) \sum_{n=1}^{\infty} \text{sinc}(n\delta_k^{\text{jitter}}(t)) \\ \left[\cos\left(2\pi n \frac{t}{T_k}\right) - 2\pi n \frac{\tau(t)}{T_k} \sin\left(2\pi n \frac{t}{T_k}\right)\right] \end{aligned} \tag{3.10}$$

The assumption of zero-mean jitter holds for an infinite observation period. However, for a finite period, T_o this is not the case, and as T_o increases, the system aggregates multiple pulses, enhancing robustness. Averaging the jitter over T_o, the effective jitter follows $\tau_{\text{avg}}(T_o) \sim \mathcal{N}(0, \frac{\sigma^2}{T_o})$. The duty cycle and frequency over T_o are modeled as:

$$\delta_k^{\text{jitter}}(T_o) = \delta_k + \Delta\delta_k(T_o), \quad f_k^{\text{jitter}}(T_o) = f_k + \Delta f_k(T_o)$$

As T_o increases, jitter effects on both duty cycle and frequency diminish. The Fourier series, averaged over T_o, shows reduced phase noise:

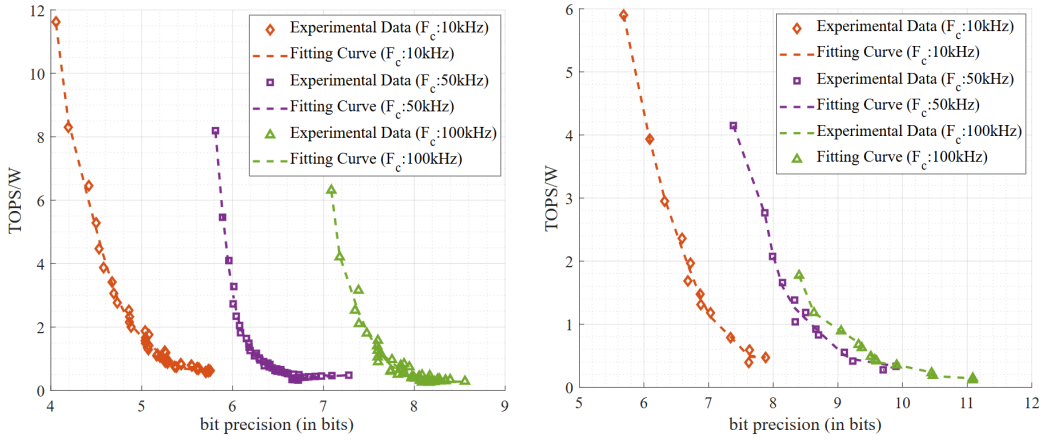


Figure 3-9: **Evaluation of bit precision vs TOPS/W for scaled addition with (a) ASDM-LP, and (b) ASDM-RP based implementation.**

$$x_k^{\text{jitter}}(T_o) \approx p_k + 4\delta_k^{\text{jitter}}(T_o) \sum_{n=1}^{\infty} \text{sinc}(n\delta_k^{\text{jitter}}(T_o)) * \left[\cos\left(2\pi n \frac{t}{T_k}\right) - 2\pi n \frac{\tau_{\text{avg}}(T_o)}{T_k} \sin\left(2\pi n \frac{t}{T_k}\right)\right] \tag{3.11}$$

As T_o increases, the impact of jitter decreases, reducing phase noise and enhancing the robustness of the encoding process. Fig. 3-8a shows the effect of T_o on the jitter within the stream. SPICE-simulated results are compared against the analytical model, and the values are normalized for circuit independence. This jitter contributes to information loss and can be utilized for calculating the Signal-to-Noise Ratio (SNR), where $SNR = \dfrac{\delta_k^2}{var(\Delta\delta_k)}$. This SNR can then be used to determine the Effective Number of Bits (ENOB), thus quantifying stream encoding precision. Fig.3-8b presents the experimental ENOB at various modulator natural frequencies (Fig. 3-7c) (F_c) as a function of T_o, which manifests as latency in the encoding process. Scaling F_c increases the pulse count "n" for a given T_o, reducing average jitter and improving encoding precision, but at the cost of higher dynamic power.

The analysis of computational efficiency metrics, examining ENOB (Effective Number of Bits) versus latency, offers valuable insights for developing variable-precision

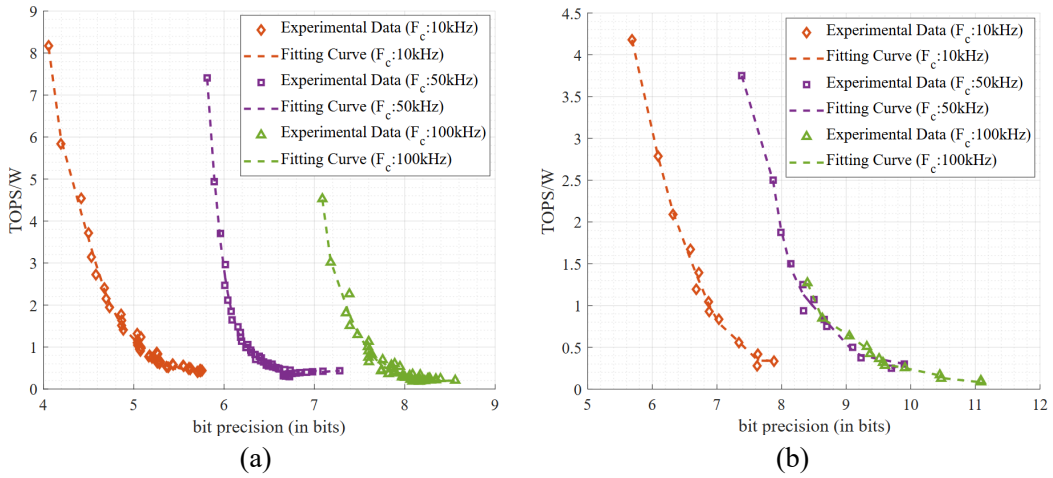


Figure 3-10: **Evaluation of bit precision vs TOPS/W for dot product operation with (a) ASDM-LP, and (b) ASDM-RP based implementation.**

accelerators. Understanding how computational efficiency varies with bit precision is essential for designing systems capable of exploring a broad design space to optimize performance and power. In this study, we introduce two primary circuit implementation profiles targeting high-performance (HP) and low-power (LP) applications, respectively. The HP design utilizes a programmable Schmitt trigger-based $\Sigma\Delta$ modulator, while the LP design is implemented through a Dynamic Leakage Suppression (DLS) inverter. Detailed descriptions of these designs are provided in Section 3.3.

To evaluate scalable precision, we simulate both a scaled addition and a single-cell dot product operation using an ASC stream and a 1T1R cell. Fig. 3-9 illustrates how computational efficiency (TOPS/W) changes with bit precision across both implementations and different natural frequencies. From a TOPS/W perspective, scaling the natural frequency (F_c) emerges as a key lever for achieving higher precision within a given TOPS/W target. Specifically, increasing F_c by a factor of 5 across implementations yields an average precision gain of 1.5-1.85 bits. The DLS (LP) approach nearly doubles computational efficiency at moderate precisions (5-7 bits), making it well-suited for applications prioritizing energy efficiency at moderate accuracy levels. In contrast, the Schmitt trigger (HP) implementation excels at higher precisions ($\geq$8 bits), though its efficiency gains diminish as precision further increases.

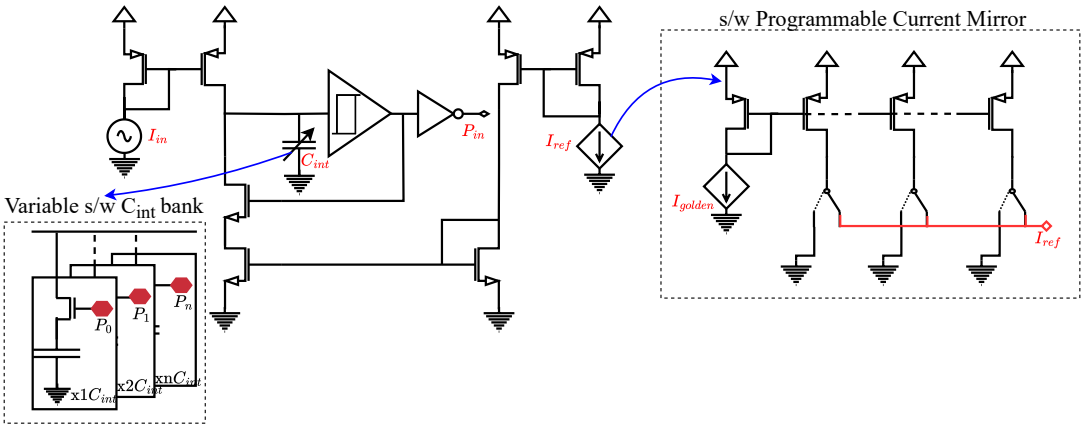


Figure 3-11: **Circuit diagram of implemented ΣΔ Modulator.**

This pattern also holds for the dot product operation, shown in Fig. 3-10, where an overall reduction in TOPS/W is attributed to the added memory read energy cost, which is factored into the efficiency metric. Scaling the natural frequency similarly enhances computational efficiency for each bit precision level. A key advantage of ASC-based precision scaling lies in offering two design knobs—stream encoding frequency and operation latency—to optimize efficiency versus precision. This dual-tuning capability provides an additional degree of design flexibility that conventional digital hardware lacks without reconfigurable fabric architectures. Consequently, this dual-approach strategy offers a clear framework for selecting the most effective implementation based on specific precision and TOPS/W objectives.

3.3 ΣΔ Modulator Implementation & Design Metrics

Based on the discussion in Section 3.1 and identifying key frequency regions of interest from an application standpoint, we have developed two implementations of the ΣΔ modulator circuit. The first design focuses on achieving a programmable modulator circuit to validate the energy scaling of the stream encoding. For this purpose, we implemented a modulator circuit with a programmable Schmitt trigger, as shown in Fig. 3-11. As described in Section3.1.3, the integration capacitor (C_{int}) can be scaled between 50-300 fF using a programmable MiM capacitor bank based on switched elements. Additionally, to control the reference current injected into

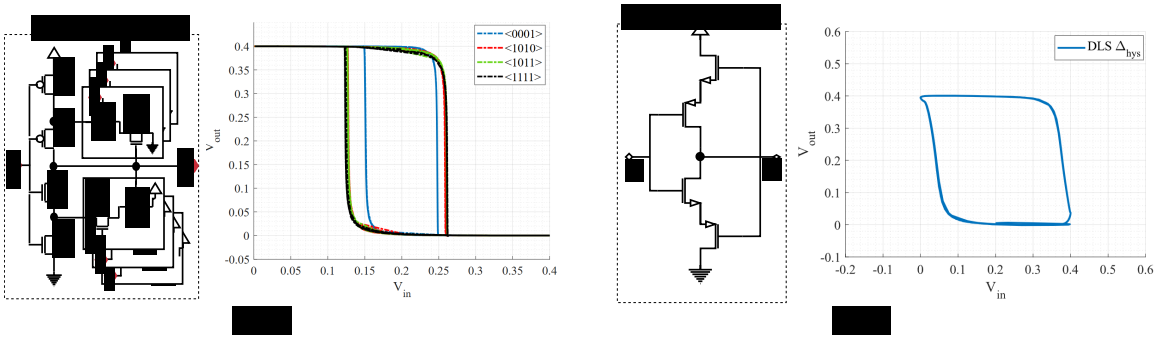


Figure 3-12: **Circuit diagram and hysteresis loop of (a) Programmable Schmitt Trigger, and (b) DLS inverter**

the circuit, we incorporated programmable two-way transmission gates with a binary scaling method, allowing current values to be adjusted between 1-25 nA.

The hysteresis of the Schmitt trigger is made programmable by varying the strength of positive feedback to the input inverter pair. This approach enables control over the width of the bistable circuit's hysteresis loop (see Fig. 3-12a). In the continuous encoding of the ΣΔ modulator, where circuits must remain "always on," static and leakage power become the primary contributors to the overall power consumption. A Dynamic Leakage Suppression (DLS)-based inverter circuit [43] is employed to address this. This design achieves super-cutoff feedback through header and footer elements, significantly reducing leakage power. This super-cutoff approach establishes distinct rising and falling switching thresholds, creating hysteresis and enabling the DLS inverter to function effectively as a comparator within the ΣΔ modulator circuit, as shown in Fig. 3-12b. However, due to the increased RC delay and larger hysteresis in the DLS inverter, the optimal operating frequency range of the DLS-based modulator is limited to 10-50 kHz.

The modulator circuits are implemented using the TSMC 65nm LP PDK and simulated in Cadence Virtuoso. Power consumption analysis focuses on static and leakage power, which dominate the modulator's efficiency. Although ASC encoding supports sparse representation, it also requires circuits to remain continuously active for asynchronous stream generation upon input. Fig.3-13 shows average leakage power savings achieved with the DLS (LP) version of the modulator, yielding

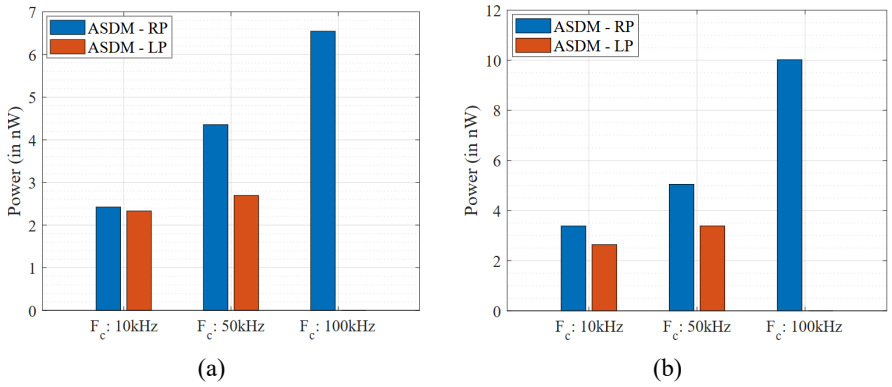


Figure 3-13: **Comparison between RP and LP implementations for (a) static + leakage power, and (b) Total power consumption.**

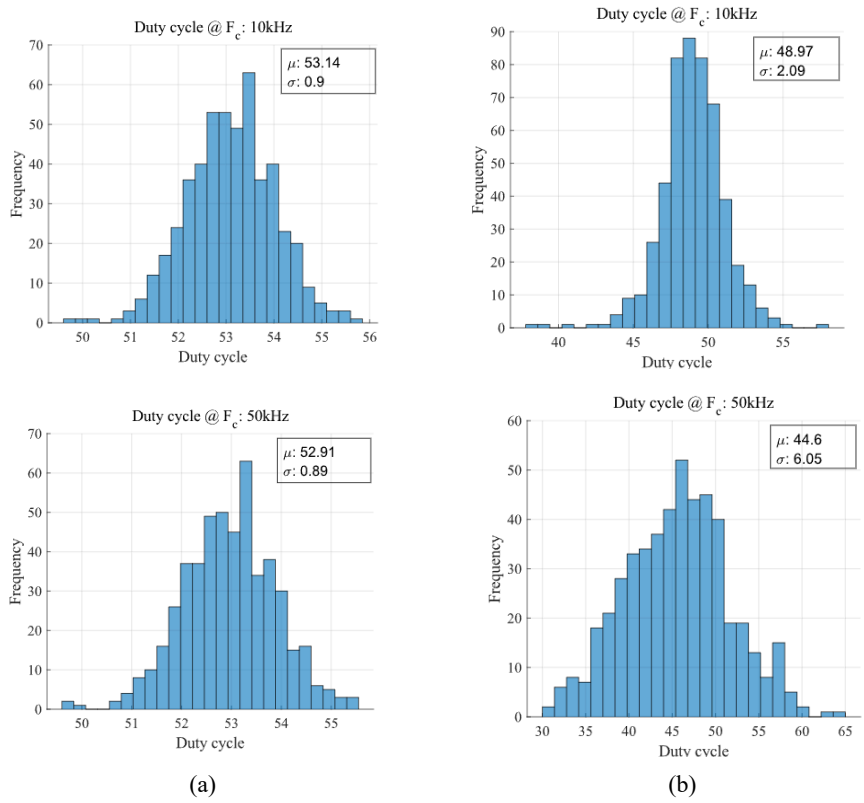


Figure 3-14: **Monte Carlo Analysis depicting temporal variations for (a) RP, and (b) LP implementation at different f_c values.**

1.1–2.43x savings when F_c is in the 10–50 kHz range. Lower savings at 10 kHz result from higher C_{int} values needed to maintain the natural frequency in the LP version, leading to increased capacitive dissipation. Based on the energy scaling strategy from Section3.1.1 and adjustments to modulator knobs, total power consumption ranges from 3.28–10.1 nW at 10–100 kHz for the RP implementation, compared to 2.56–3.48 nW at 10–50 kHz for the LP version (see Fig. 3-13b).

From a power efficiency perspective, the LP implementation is preferable across all frequency ranges. However, analysis of the encoding noise generated by these circuits reveals an impact on stream encoding precision, as detailed in Section 3.2. Fig.3-14 shows how process and mismatch variations influence the stream's temporal signature (duty cycle). The DLS inverter, due to variable switching points, exhibits higher sensitivity to fluctuations, leading to reduced encoding stability and increased duty cycle deviations at similar natural frequencies. Additionally, pulse-to-pulse variations in the temporal evaluation of the stream are pronounced in the DLS inverter, resulting in lower achievable precision for the DLS-based modulator circuit, as discussed in Section. 3.2. This analysis underscores a trade-off between power efficiency and encoding stability in selecting an implementation.

Chapter 4

EAS-CiM: Foundational block for CNN models

4.1 EAS-CiM: A Unified Platform for accelerating ML workloads

Compute-in-Memory (CiM) offers an orthogonal approach to the conventional Von Neumann architecture by performing arithmetic operations directly within the memory array, thereby mitigating the "memory wall" bottleneck. As described in Section 2.2, in a conventional analog-based CiM tile, the input vector is applied as an analog voltage in a parallel-read manner through simultaneous row-wise memory access. As illustrated in Fig. 4-1a, let the synaptic weights associated with a neuron, w_1, w_2, w_n, be stored as conductances within memristor cells. Inputs to the neuron, represented as analog or digital voltage values, are applied along the BitLine of each row. According to Kirchhoff's current law, the resulting cell current is effectively the product of the incoming voltage amplitude and the conductance stored in each memory cell. The accumulated current on the Select Line represents the dot product of the input activations and synaptic weights stored within the memory array. This read process, conducted across multiple columns, enables a high degree of parallelism and significantly boosts throughput. In our approach, instead of directly applying analog

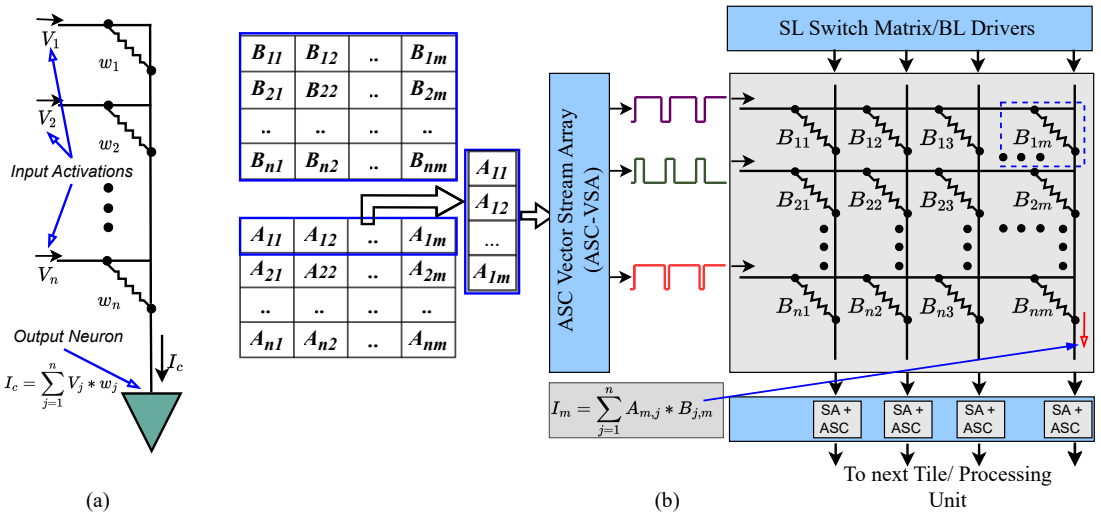


Figure 4-1: **Schematic representation of (a) conventional Analog CiM operation, and, (b) proposed CiM tile with ASC stream interface.**

voltages, we encode the analog information into ASC streams, which are then applied to the word line (WL) accessing all 1T1R memory cells within a row (see Fig. 4-1b. These ASC streams serve as control vectors for reading the conductance values programmed into each cell. Unlike traditional Analog CiM, where the generated cell current is proportional to the amplitude of the applied voltage, our implementation achieves a read operation in which the average cell current is proportional to the ASC value p embedded within the stream, dictated by the stream's effective duty cycle. The accumulated cell currents, which share a common select line (SL), collectively produce a weighted sum current. This operation can be mathematically represented as:

$$I_k = \sum_{i=1}^{m} (p_i(t) * G_{i,k}) \tag{4.1}$$

,where i and k, represent the row and column index within the array and $p_i(t)$, $G_{i,k}$ stand for the input stream vector and conductance of the 1T1R cell in the $< i, k >$ position within the array, respectively. The current generated within each column represents the dot product result and is converted back into an asynchronous stream.

The stream consists of multiple pulse instances, with pulse density determined by the stream frequency, which in turn depends on the input value. Each pulse within the stream initiates a single read operation, creating a successive accumulation of current on the Select Line (SL). This accumulated current is aggregated over time on the

output capacitor of the Sigma Delta modulator, effectively performing repetitive dot product operations. The repetitive nature of these operations helps average out any random temporal variations in the devices across multiple read iterations, ensuring robust operation. A detailed analysis of the precision of ASC streams can be found in the analytical model described in Section 3.2.

As the incoming array of vector stream encodes information in the temporal domain, the resulting current in each SL of the crossbar array will encompass different frequency components that pertain to the incoming elements. To ensure a faithful representation of the desired computation is attained, the output of each column needs to be encoded to a frequency range that covers the span of these multiple frequencies. The range of output frequencies must also reflect the scale of computation in terms of the number of rows that are accessed at the same time, i.e., the matrix order. To understand the encoding process, let us assume that any incoming vector element, $A_{ij} \in [-A_{min}\ A_{max}]$ is normalized to the stochastic stream range, $p_{in(ij)} \in [-1\ +1]$ and that would correspond to an equivalent frequency range of $f_{in} \in [0\ f_c']$. On a similar note, to ensure that the integrity of information is maintained, an output frequency range ($f_{out} \in [0\ F_c]$) is required at the output column ASC core such that $\frac{F_c}{f_c'} = \gamma$, where γ is the computational scaling factor and is a design space knob. Next, we focus on how input vector element streams will be re-encoded at the output column once it undergoes a dot product operation with the corresponding memory cells in the ASC-PiM tile. Let's consider an input element, A_{ij} applied as a control vector to a 1T1R cell that is programmed with a conductance whose value maps to $B_{ij} \in [-B_{min}\ B_{max}]$. For the sake of simplicity, let us consider the equivalent conductance value G_{ij}, such that $B_{ij} = 1$ and therefore, the individual dot product would result in $A_{ij} * B_{ij} = A_{ij}$. To understand the relation between the input vector and computed output vector, we encode the input vector A_{ij} as p_i, wherein;

$$\frac{f_i}{f_c'} = (1 - p_i^2) \implies f_c' = \frac{f_i}{(1 - p_i^2)} \tag{4.2}$$

Based on our initial definition of γ, f_c' can be redefined as $f_c' = \frac{F_c}{\gamma}$ and substituting

that into eq.(4.2):

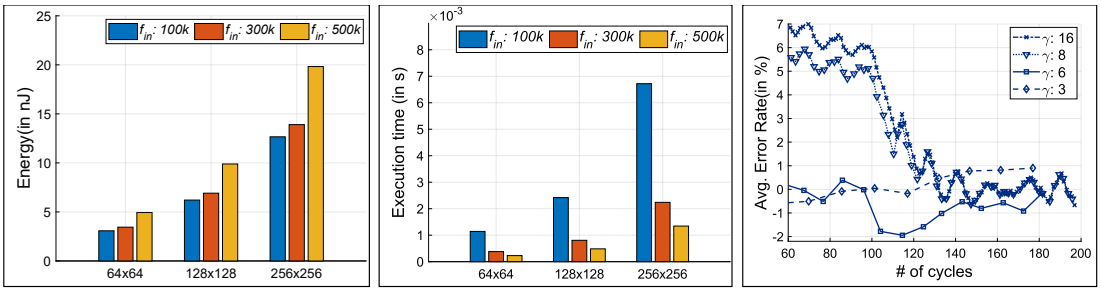
$$\implies F_c = \frac{\gamma * f_i}{(1 - p_i^2)} \tag{4.3}$$

Now, the resulting dot product value $A_{ij} * B_{ij}$, would be encoded by the output column ASC as P_{out}, wherein $\frac{F_{out}}{F_c} = (1 - P_{out}^2)$ and replacing F_c with eq. (4.3):

$$\frac{F_{out}}{\frac{\gamma * f_i}{(1 - p_i^2)}} = (1 - P_{out}^2) \implies \boxed{P_{out} = \sqrt{1 - \frac{F_{out} * (1 - p_i^2)}{\gamma * f_i}}} \tag{4.4}$$

Equation (4.4) establishes the relationship between the scaled output vector and the corresponding encoded set of input vectors. Intuitively, the input analog signal range wouldn't be the same as the current generated on the SL due to the conductance ranges, cells/column, parasitics, etc., and both these ranges are mapped to the same $p \in [-1\ 1]$ scale. This relationship allows us to understand the mapping irrespective of the device, crossbar array, and input range parameters.

To evaluate the EAS-CiM implementation, Vector-Matrix Multiplications (VMM) with order sizes ranging from 64x64 to 256x256 are simulated. To understand the effect of the average frequency of the stream vector on the performance of the CiM operation, we scale the natural frequency f_c of the streams from 100 - 500 kHz. While accelerating a 256x256 VMM operation and scaling the stream frequencies from 500–100 kHz, the average energy savings is roughly 1.6x. This effect is more pronounced for lower matrix orders (64x64), where the energy savings are over 2.1x (Fig. 4-2a). This is because as the input frequency range scales down, the energy contribution by the ASC components sharply reduces, but the increased read times result in a larger memory energy cost. This results in reduced energy savings as the overall crossbar array size increases from 64x64 to 256x256. On the other hand, these energy savings come at the cost of a longer execution time. An alternate viewpoint to this scaling is that given an abundance of energy reserves, the architecture will be able to execute VMM operations at a faster rate by encoding the vectors on a



(a) (b) (c)

Figure 4-2: **Evaluation of stream-based CiM VMM operations.**

higher stream frequency range. By scaling the stream rate from 100-500 kHz, the architecture can accelerate the multiplication operation approximately 4.7-5.1x times across the 3 order sizes (Fig. 4-2b). When accelerating a 64x64 MMM operation, the best-case execution time is under 0.23 ms, whereas the average execution time is under 1.3 ms for an order size of 256.

Fig. 4-2c assesses MMM operation accuracy by varying γ for a fixed input frequency. The $\Sigma\Delta$ core integrates the input signal over time, yielding an averaged dot product at the output. For $\gamma \geq 8$, initial error rates are high due to insufficient integration time but stabilize over time. Conversely, with low output stream frequencies ($3 \leq \gamma \leq 7$), the modulator achieves stable output with fewer cycles, indicating latency within the operation. However, lower γ values reduce throughput, unsuitable for high-performance applications.

4.2 Interfacing ASC with eNVM-based CiM tiles

Memristors are emerging memory devices that can store analog information as resistance [62] and can be utilized in 2D arrays as hardware implementations of vector-matrix multiplication (VMM), allowing for in-memory computing for machine learning applications [52]. However, the device variations arising from random noise sources [71, 12] pose a substantial risk of errors in implementing VMM within CiM architectures, complicating their widespread adoption in CiM memory arrays.

We fabricated a HfO_2 device with a Ta/Pt (50/25 nm) top electrode, a HfO_2 (5 nm) memristive medium, and a Ti/Pt (5/30 nm) bottom electrode. The fabricated

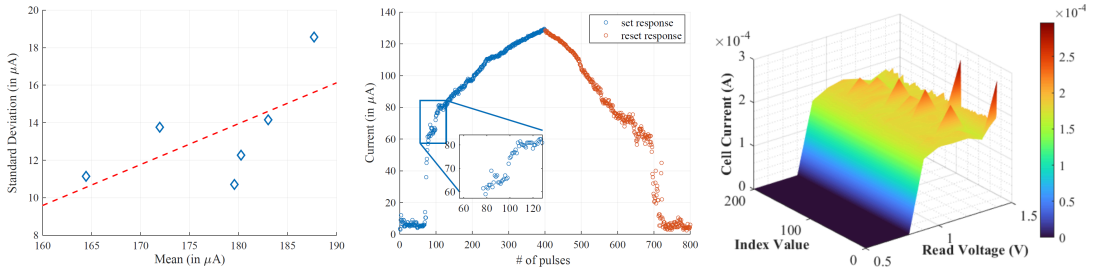


Figure 4-3: **(a) set and reset behavior, and (b) extracted fitting curve characteristics for fabricated HfO_2 device. (c) contour plot depicting fabricated device variations.**

device allows us to assess experimental intrinsic variations in nonvolatile resistive switching attributed to significant rearrangements of oxygen vacancies[52]. This random switching during programming results in a variable set of current characteristics, which also depend on the applied read voltage, as illustrated in Fig. 4-3a&c. To embed the variations of the fabricated device into our classifier model framework, the deviations within the 1T1R cell current values are extracted based on a fitting curve as depicted in Fig. 4-3b.

For EAS-CiM to be compatible with existing in-memory computational frameworks, the layout of a conventional crossbar memory array must be preserved. Critical modifications have been implemented within the memory kernel to facilitate the asynchronous read operation over an input feature-bound memory read period. For instance, as depicted in Fig.4-4a, we modify the conventional 1T1R cell (dashed grey box) by swapping the conventional WLs and Bit Line's (BLs) (dashed blue box) and applying the ASC stream onto the access transistor. By engineering the dimensions of the access transistor, we can control the voltage drop across the memristive device, thus minimizing the possibility of reprogramming the cell during the read, and a similar strategy can be adopted for a 1R-based cell device as well.

To demonstrate the linear relationship between the incoming stream and the programmed conductance of a cell, the average cell current for both the High Resistance State (HRS) and Low Resistance State (LRS) is shown in Fig. 4-4b. To account for these variations in our framework, a comprehensive Monte Carlo analysis was performed to assess deviations in cell current caused by both stream variations and device

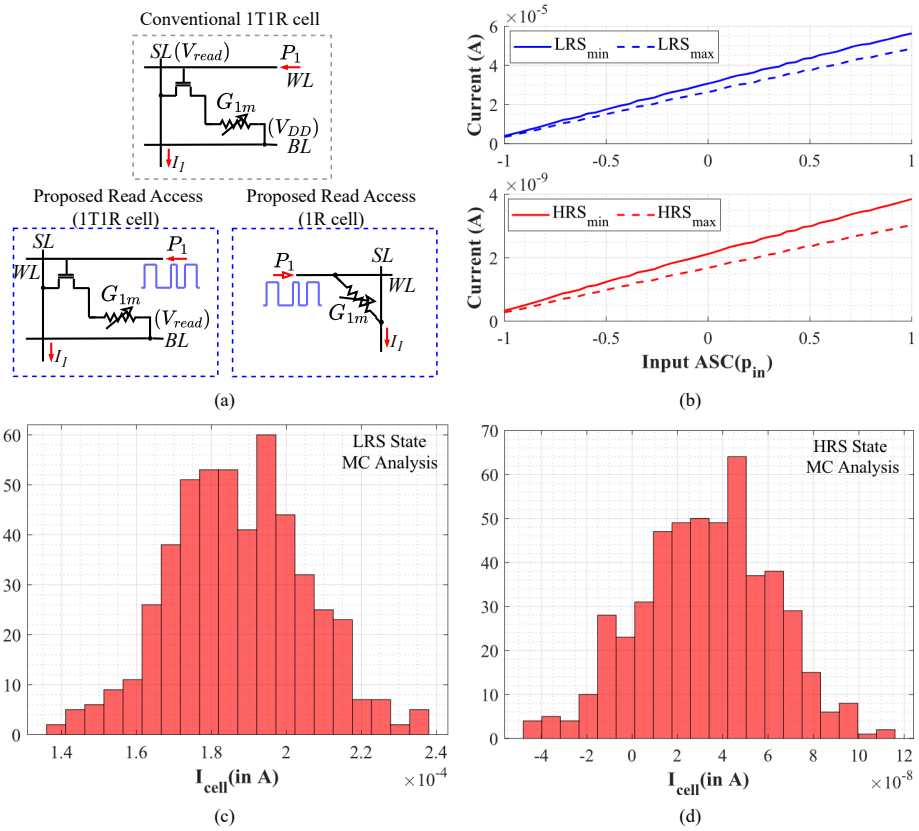


Figure 4-4: **(a)Reconfigured access terminal for 1T1R ad 1R configuration, (b) Average cell current vs p_{in}, Monte Carlo Analysis of average cell current at (c) HRS state, and (d) LRS state..**

characteristics, as depicted in Fig. 4-4c & d. These variations are then integrated into the EAS-CiM Python-based simulation framework.

4.3 Stimulus Awaren Computing: A Key to Energy-Efficient and Scalable Computing

To understand the qualitative benefit of implementing a stream-driven VMM/MMM computational architecture, we take a segment from a moving MNIST image dataset, as shown in Fig. 4-5a. Our architecture leverages the concept of a *stimulus-driven workload execution rate* to integrate incoming feature information into asynchronous streams, governing computational operations such as MAC, activations, and pooling. Fig. 4-5b represents the heat map of the frequencies for each pixel-encoded stream,

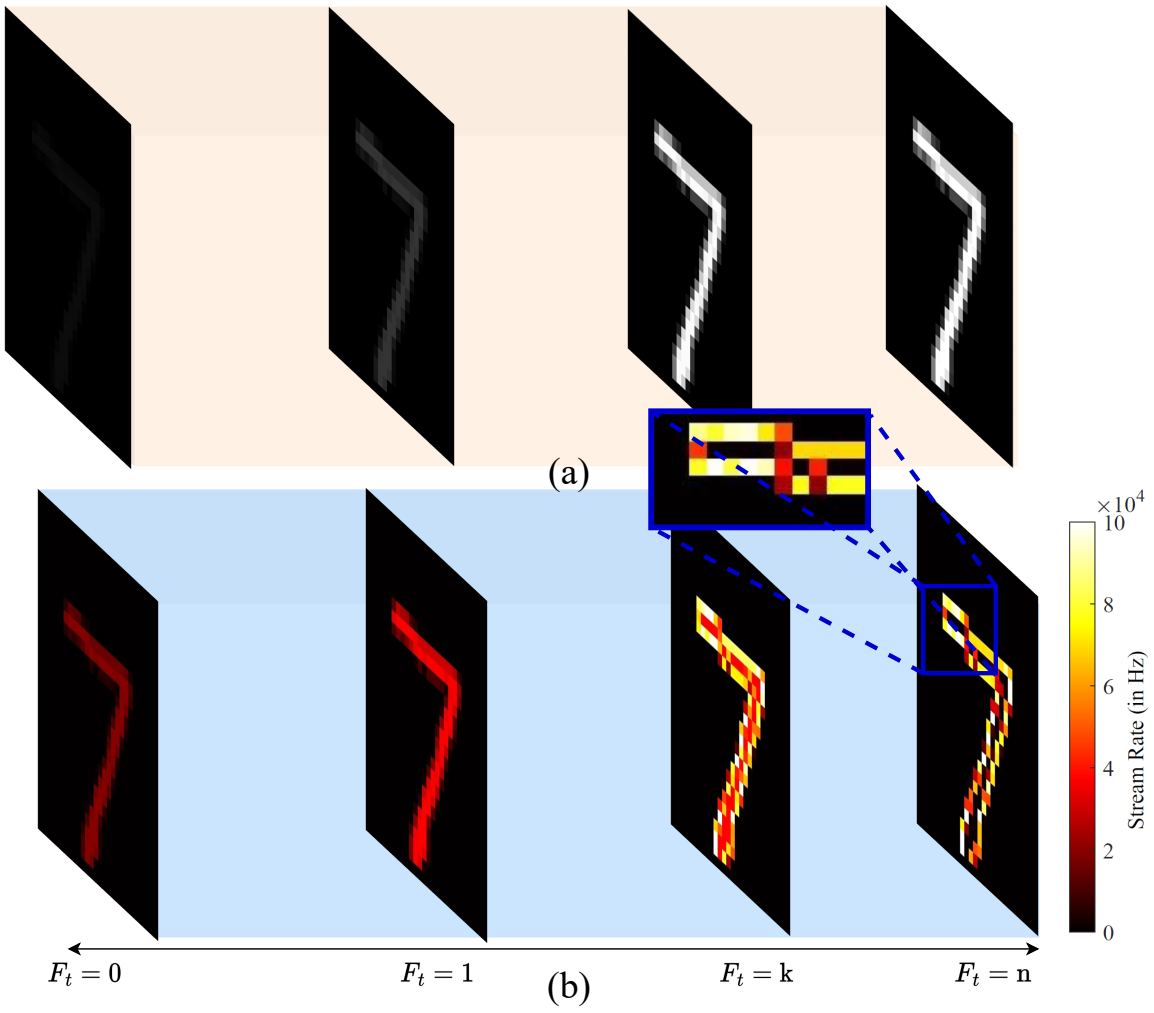


Figure 4-5: **Concept of stimulus-driven work rate depicted with a moving MNIST-dataset image frame.**

which is generated by a P2S converter and will be discussed in Section. 4.3.1.

Our architecture leverages the concept of a *stimulus-driven workload execution rate* to integrate incoming feature information into asynchronous streams, governing computational operations such as MAC, activations, and pooling. In Fig.4-5a, a frame-by-frame segment of the moving MNIST dataset is shown, accompanied by a heat map of encoded stream frequencies in Fig.4-5b. Within this approach, the magnitude of features between successive frames increases in value. This is to emulate real-world conditions such as object detection, where there can be a sudden surge in feature intensity (appearance of an object within a frame) for a brief period, with the remainder of the time experiencing static features. Based on our encoding technique,

high-stream frequency clusters are formed along the gray edges of the digit and represent the critical feature patterns within each frame. This aspect of the encoding technique is observed in the highlighted blue box depicted in Fig. 4-5b, where the highest stream rate occurs where the pixels are gray. This encoding can be modified to enable a similar trend wherein critical features are represented with high temporal frequencies. On a similar note, black and white pixels that stray away from the edges of the digit are encoded at a lower rate and emphasize EAS-CiM's ability to suppress the work rate when the incoming feature doesn't contain critical information. This particular aspect is beneficial in applications such as edge detection, as our encoding method allows for the suppression of non-edge-based features without having to implement feature-specific extraction techniques to perform a similar action.

Within EAS-CiM, the featured encoded stream vectors control the rate of memory access to perform in-memory kernel computations and thus control the number of MAC operations carried out over an inference period. For instance, the streams associated with gray pixels, when applied as a memory read control vector onto the CiM tile, will result in a larger number of dot product operations/second in comparison to a black or white pixel. This allows our architecture to perform more iterative computations on critical features and carry out minimal to no computations on features that are insignificant. From a general time-series-based data classification, where there could be periods of sporadic activity, the stream-based access of the kernel ensures that the rate of feature computation is directly proportional to the activity within the time-series data. Stream frequency curves can be adjusted by varying key circuit parameters, detailed in Section 4.3.1. The energy cost of shifting the frequency curve depends on circuit sensitivity and operating frequency region [64]. This allows an energy-scalable temporal architecture, contributing to a scalable performance aligned with workload execution energy costs. To ensure that this rate of computation can be scaled across every layer, each arithmetic and non-linear computational element is visualized in a manner that adheres to ASC principles.

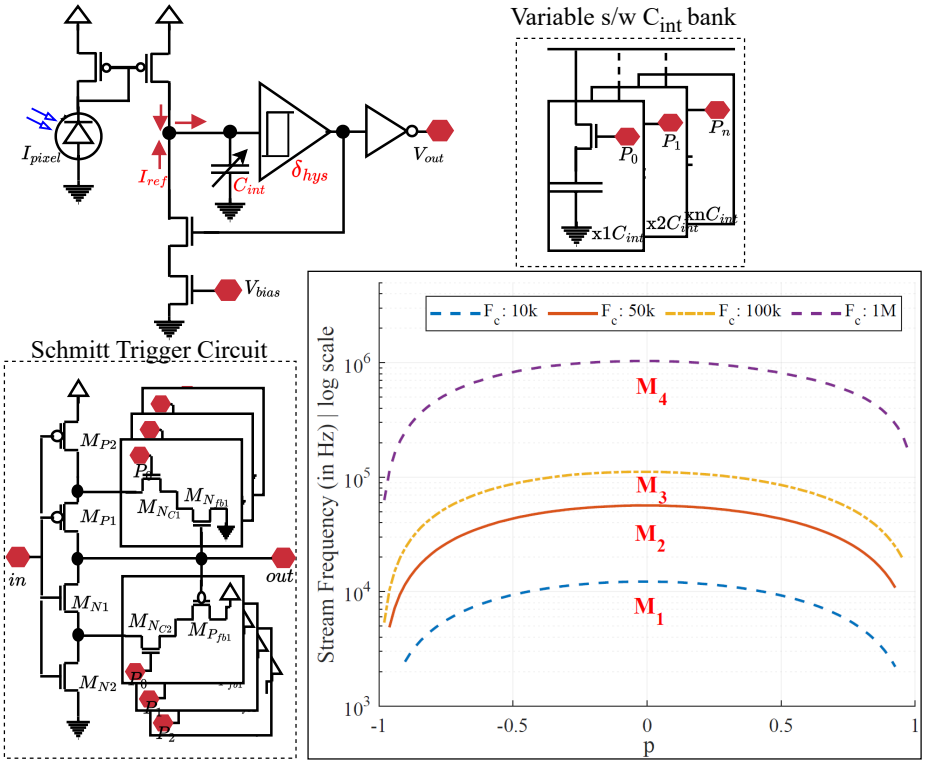


Figure 4-6: **Schematic diagram of P_2S converter and temporal characteristics at different frequency ranges (bottom)**

4.3.1 Event-driven Pixel to stream (P_2S) converter

To allow for seamless integration with the ASC-based CiM tiles, it is necessary to implement a continuous time-based Feature eXtractor (FeX) module that reflects the extracted feature in a manner that can be utilized as a stream-based control vector. In this work, a modified sigma-delta ($\Sigma\Delta$) modulator circuit is utilized as a P_2S converter (Fig. 4-6) and encodes the incoming pixel value into an asynchronous stream, with temporal characteristics mentioned in Section. 3.1. The modulator operates on the difference between the input pixel value, which is represented as a current source, I_{pixel}, and the output stream V_{out}. To operate on the difference, V_{out} is represented in the current domain by a switched current source, I_{ref}, and the difference is integrated on the capacitor node, C_{int}. The output pulse, V_{out}, has a constant amplitude and an instantaneous output frequency (f) and duty cycle (δ), as described in Section 3.1. As briefly described in Section 3.1, the modulator can scale its stream rate by varying the natural frequency (f_c) and shifting the entire frequency curve. This allows us to

scale the memory access rate and number of arithmetic operations carried out in a second. From an applications perspective, this aspect translates into another original contribution of our work, where we can scale performance metrics such as Tera Operations/second (TOPS) or IPS. However, as we scale the frequency of the stream, it also affects the energy costs associated with the generation of the stream. Therefore, we must opt for the least energy cost path to scale the temporal characteristics, and this is achieved through programmable elements in the form of I_{ref}, C_{int} and δ_{hys} (depicted in dashed boxes in fig. 4-6). Fig. 4-6 (bottom right) depicts a series of temporal characteristics ($M_1 - M_4$) that can be achieved by varying a combination of these design elements. The impact of these design elements on energy costs has been analyzed [64], enabling us to allocate specific frequency regions where each circuit element optimally scales. A similar approach to scale the stream frequencies of all ASC elements within EASI-CiM has been implemented and enables the entire architecture to traverse an energy-efficiency vs performance design curve. At this juncture, it is critical to distinguish between the **stimulus-driven work rate** and **scalable energy efficiency** principles. The encoding technique of ASC enables a dynamic & natural implementation of the stimulus-driven work rate and can be visualized as traversing a unique frequency curve (e.g., M_1) on the temporal characteristic. To scale the energy efficiency, one must *reprogram* the encoding to traverse from one frequency curve to another (e.g., $M_1 \rightarrow M_3$), and in turn, affects the computational performance.

4.3.2 Activation functions within EASI-CiM

The activation function forms a critical component within any neural network as it introduces non-linearity within the network and improves the abstraction of complex features. Activation functions designed within the framework reflect the non-linearity in a stream-based manner, thereby facilitating truly stream-based communication between layers. The tanh activation function modifies the ASC-SD neuron stream output from the CiM tile into a current source that is fed into a tanh I-V converter circuit designed to model the transfer function curve. Fig. 4-7a depicts the circuit

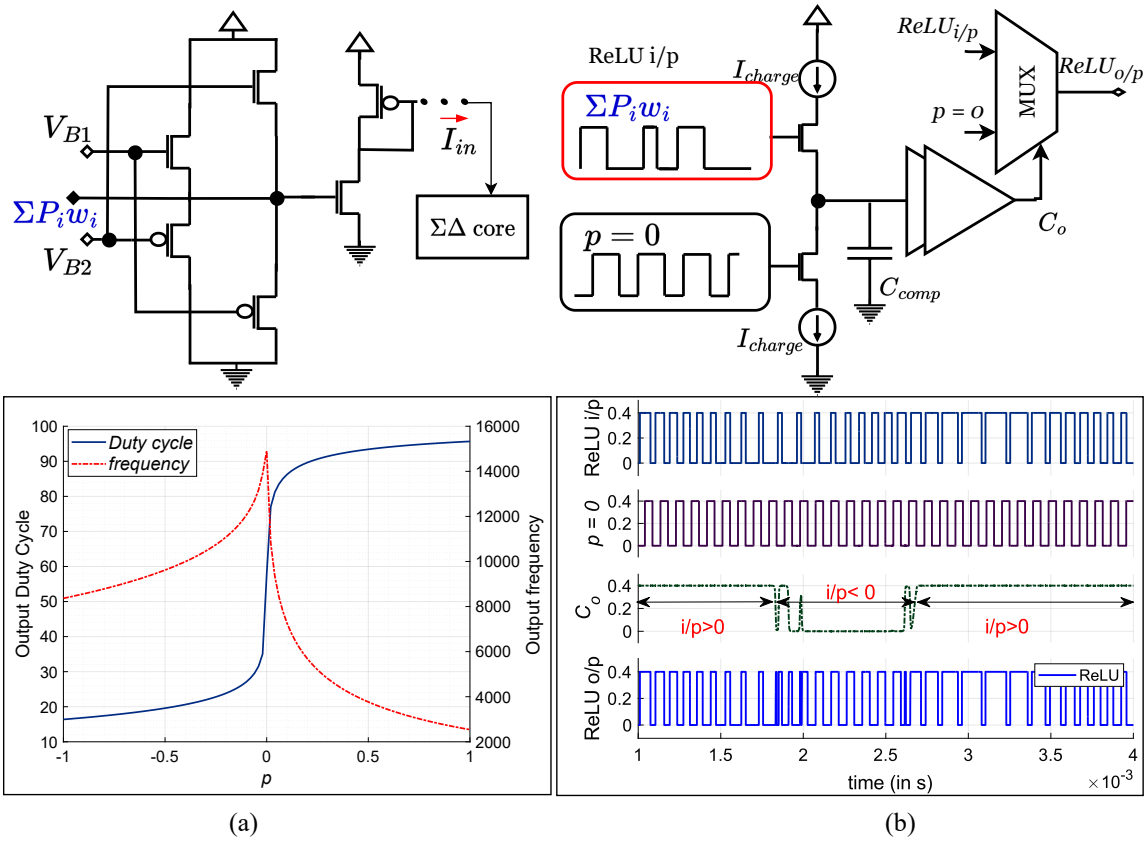


Figure 4-7: **(a) Circuit diagram of ReLU activation unit and (b) operational transient characteristics.**

along with the temporal characteristics of the transfer curve. As the duty cycle maintains a linear relation with the stream value, p, it can be used to determine the faithful representation of the activation function.

The mathematical operation that occurs within a ReLU function is the comparison operation between the incoming vector and a bias value. A comparison operation is mathematically equivalent to a negated addition and observing if the output is > 0 or ≤ 0. This operation is implemented through a CMOS-based charge pump circuit, where the switched current sources are controlled by the incoming vector and bias vector, and the resulting charge accumulation is utilized to drive a rail-to-rail comparator element. The resulting comparison acts as a select line control signal for a MUX, which allows for the transmission of the incoming stream, or p= *bias*, depending upon the comparison result, as illustrated in Fig. 4-7a. It should be noted that the working principles also apply when a non-zero bias needs to be applied.

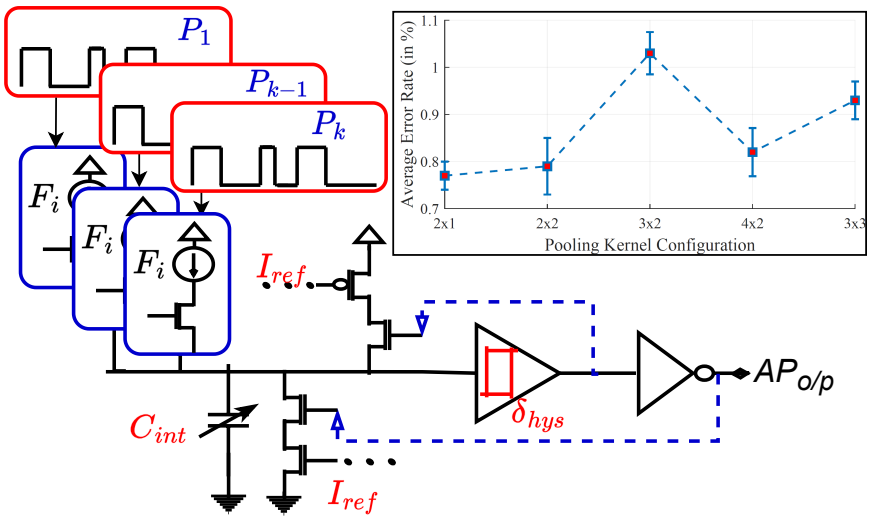


Figure 4-8: **Implemented schematic diagram of pooling kernel with average error rate).**

Fig. 4-7b, reflects the transient operational characteristics under multiple conditions wherein, the incoming i/p vector $<$ bias and vice-versa. The comparator output C_0 drives the selector signal for the MUX and transmits either the input stream or "zero"-equivalent vector to the next neuron layer.

4.3.3 Pooling Layers within EASI-CiM

Pooling layers are implemented to minimize the sensitivity of feature maps and generate condensed feature maps across layers within a CNN. Stream-based pooling kernels translate the aggregating streams to compress information along with desensitizing the pooled vector due to the multiple occurrences of the stream across a period. In our NN model, we implement an average pooling layer, with the first step being the translation of incoming streams into a cumulative single analog value that represents the summation operation. We utilize an array of voltage-controlled current sources (VCCS) in the form of switched cascode current mirrors. The cascode configuration ensures a higher input impedance and minimizes the switching effects caused by the multiple input streams. The number of VCCS elements depends on the kernel size of the pooling layer, where each input stream is mapped to a unique VCCS element. To implement the averaging part of the operation, a modified $\Sigma\Delta$ modulator core with

a scaled adder circuit is utilized. The scaling factor of the adder is attributed to the current integration principle, and the scaling aspect is reflected by the ratio of the max current generated by an individual VCCS element to the reference current (I_{ref}) within the modulator. Fig. 4-8 represents the circuit implementation of an average pooling kernel along with an error graph that depicts the simulated average error for different kernel configurations. We opt for integrating an average pooling kernel due to its compatibility with our operational schema based on simple circuit laws like Kirchhoff's Current Law (KCL).

4.4 Benchmarking Edge ML models with EAS-CiM

4.4.1 Scalable Emulation Platform for EAS-CiM

At the foundational tier, ASC-based elements are designed and extensively tested using a commercial 65nm PDK technology. These elements are rigorously simulated within the Cadence Virtuoso environment to characterize the behavior of individual computational components and to capture the dynamics with corner and silicon variations captured through Monte Carlo analyses. As detailed in Section 4.2, extensive experimental evaluations focused on variations within the fabricated HfO_2 device were conducted to understand its behavior under real-world conditions. The simulations focus on accurately modeling both the core functionality and the performance deviations introduced by physical variations in the CMOS circuits and fabricated HfO_2 device. Through exhaustive simulation, the low tier provides detailed insights into the electrical properties and limits of each ASC-based circuit component, ensuring that the behavior and limitations observed in hardware are faithfully represented in higher tiers. This tier establishes a robust foundation for accurately modeling the EAS-CiM architecture at all subsequent levels.

The mid-tier of the framework is dedicated to capturing and analyzing network-level non-idealities that arise when scaling the ASC-based design, specifically addressing critical issues such as IR drop and RC delay. In this tier, we examine the effects of

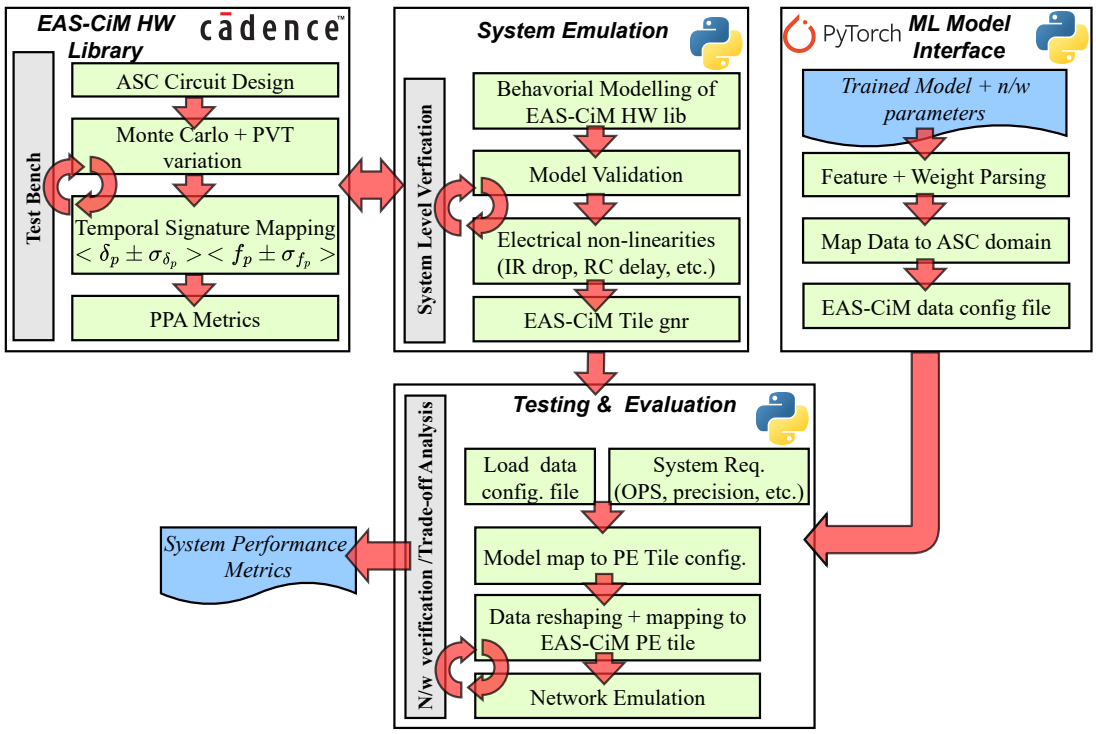


Figure 4-9: **Process & design flow overview for EAS-CiM framework.**

IR drop, a voltage reduction that occurs along the select lines (SL) due to inherent line resistance. This phenomenon can lead to diminished signal strength and variability in the voltages received by cells further along the line, resulting in uneven current distributions and impacting the overall accuracy of read operations. Another significant non-ideality considered at this level is RC delay, which manifests as a result of combined resistance and capacitance along the word lines (WL) during the read process. RC delay introduces timing skew, causing variations in the arrival times of signals across different parts of the network. This mismatch affects the temporal alignment of ASC streams at different stages, potentially leading to inconsistencies, which can be quantified as the source of noise/jitter in the encoding process as mentioned in Section. 3.2, that represents the computational outcome.

By incorporating these non-idealities into the model, the mid-tier enables a detailed evaluation of how they impact the fidelity of ASC streams and their resulting duty cycles. Understanding these effects allows for refined configuration of tile dimensions, and routing strategies to mitigate IR drop and RC delay as much as

possible. Ultimately, this tier provides a critical bridge between device-level behavior and network-level performance, ensuring that the cumulative impact of non-idealities is well accounted for as the design scales in complexity.

At the highest tier of the EAS-CiM framework, we developed a custom Python-based environment that models the architecture's complex transient and event-driven behavior, providing a high-fidelity emulation of the EAS-CiM tile. This Python framework encapsulates the transient behaviors and non-idealities of each computational element described in previous sections, offering functions that accurately reflect the variations, delays, and non-linearities present in the physical circuits. By abstracting these characteristics, the framework delivers a realistic representation of the circuit-level dynamics of the ASC-based architecture. Communication within the framework is designed around asynchronous packets, which carry critical parameters such as the duty cycle and frequency of each pulse within a stream. This approach captures the temporal signature of each pulse without storing or transmitting large arrays of transient voltage values for each stream, significantly reducing memory and computational overhead. As a result, the framework remains lightweight and scalable, even for complex models, while preserving the integrity of the ASC-based computations. The overall process and design flow of the framework is depicted in Fig. 4-9.

To enhance flexibility and scalability, the Python framework is designed with a modular, "plug-and-play" architecture. Users can easily select and integrate specific computational elements within the framework, allowing for customizable configurations that can be adapted to a variety of network topologies and operational requirements. This modularity supports dynamic adjustments in network depth, width, average stream frequency, 1T1R memory cell types, activation functions, and more, enabling seamless adaptation to different model configurations. Additionally, this Python environment integrates with ML training frameworks like PyTorch, which allows mapping CNN model architectures, layer dimensions, and computational tasks to equivalent implementations within the EAS-CiM framework. This end-to-end emulation provides an efficient and versatile tool for analyzing, optimizing, and scaling ASC-based computations for diverse ML tasks.

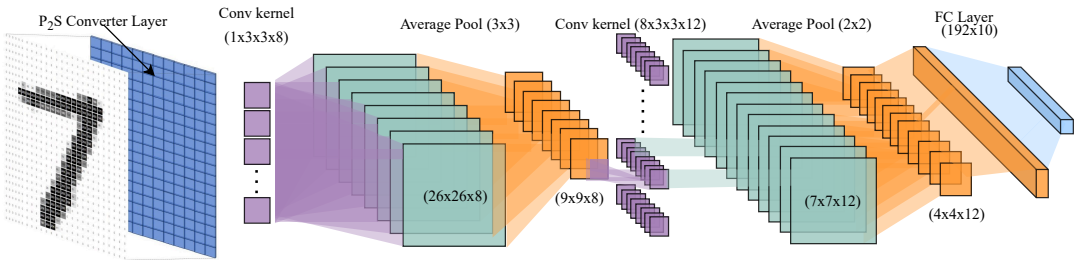


Figure 4-10: **Implemented CNN model for MNIST digit classification.**

4.4.2 EASI-CiM: Event-driven Image Classifier with EAS-CiM

Fig. 4-10 represents the CNN topology to implement an MNIST image classifier, where a 28x28 input image feature map is streamed into a 3x3 convolution kernel that comprises 8 channels, resulting in a 26x26 filter output. A Rectified Linear Unit (ReLU) function is employed as the activation function on the filter output. The convolution layer is followed by an average pooling layer with a 3x3 kernel size to compress the feature map to 9x9 across 8 channels. In the second layer of the network, the 8 input channels are split into 12 output channels after pushing them through a 3x3 convolution kernel. The final average pooling kernel size is 2x2 and is fed into the final Fully Connected (FC) Layer that maps to the output labels. A softmax function is utilized to distribute the classification probabilities across the different labels. The optimal height and width for the CiM tile are determined, based on non-linearities that incur within the tile, as discussed in Section 4.4.1. The convolution kernel is reorganized and mapped onto the CiM PE tile, to carry out the required MAC operations. An array of activation and pooling kernels are present within each PE tile, to map subsequent layers of the network. Tiles are positioned to enable spatial processing of data, as the stream-like nature of ASC prevents the temporary storage of intermediary output and requires feature vectors from one layer stream onto the next layer, and so forth.

The CNN model, trained on PyTorch, uses the MNIST dataset [40], with 60k images for training and 10k for testing. Training and backpropagation are performed using the Adadelta algorithm from the Keras optimizer package, with a learning rate of 0.003, essentially a stochastic gradient descent method. The architecture's in-

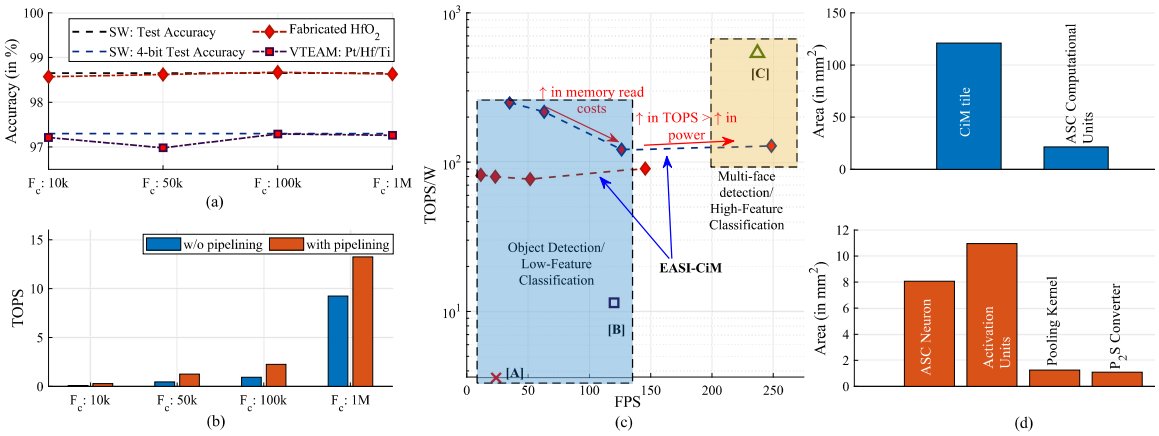


Figure 4-11: **Evaluation of the EASI-CiM image classifier based on the MNIST handwritten image dataset.**

ference operations are simulated within the aforementioned custom Python design environment, completing the design loop of the EAS-CiM framework. As mentioned earlier, another key aspect of our work is the ability to shift the entire stream frequency characteristics and scale the energy efficiency of the computation. To analyze this aspect from a performance perspective, we shift/scale the stream frequency of the entire architecture to four distinct (f_c) ranges, which mimic varying levels of real-world energy reserve conditions.

Inference Accuracy: Fig. 4-11a, exhibits the simulated inference accuracy results of the network. The fabricated HfO_2 device is programmed with floating precision weight values, as these devices exhibit reliable analog programming of conductances [52]. We also implemented an instance of the well-known theoretical VTEAM-based Pt/HfO_2/Ti RRAM model and retrained the network with quantized 4-bit precision weights based on the stable conductance levels derived from the fitting characteristics [38]. Our simulations show minimal degradation in accuracy as we scale to different frequency operating regions. The maximum deviation between the software (SW) testing accuracy and simulated inference accuracy across the frequency ranges for both implementations of the CiM tile is $\leq 0.34\%$. On a similar note, the robustness of the network against the different flavors of the memristive device is also evaluated across the different operating regions. The variance between the simulated inference accuracy in comparison to the SW test accuracy is approximately 0.41%.

Temporal Pipelining: One aspect of the framework that limits the overall performance of the accelerator is the inability to *store* intermediary results between layers due to the stream-like nature of the information. This hinders the possibility of implementing performance improvement techniques such as pipe lining within the architecture. However, the inherent latency within the architecture owing to the internal delay generated by the ASC components can be repurposed as a *temporal buffer period* between stages. This buffer period can be visualized as a temporal pipeline stage where the buffer period lies between any two computational layers within the network. It allows us to feed new feature vectors into the network as the current inference workload cycles itself through these buffer periods across the layers. The internal delay for the ASC computational units has an inverse relation to the natural frequency of the modulator and depends on the magnitude of the input itself to a certain extent. However, to analyze the extent to which the temporal pipelining can be inserted into the network, we need to normalize the internal delay to the natural time period of the modulator. In our evaluation, we observe that the normalized delay reduces drastically as the average stream frequency of the architecture increases. As we scale from 10 kHz to 1 MHz, this delay reduces by approximately 46.5% and represents a diminishing return to the temporal pipelining benefit as we scale the frequency. To better understand the benefits of temporal pipelining, we utilize TOPS as the quantitative metric, as depicted in Fig. 4-11b. When the natural frequency of the architecture is 10 kHz, a 3.05x improvement in TOPS is observed with the temporal pipelining technique. However, this benefit is reduced to 1.4x when the frequency is 1 MHz, with the performance peaking at approximately 13.21 TOPS.

Scalable Energy Efficiency: As mentioned in previous sections, our architecture can scale the overall stream frequency and attain a variable performance. However, in the case of edge accelerators, executing a workload at the lowest energy cost often takes precedence over performance. With the MNIST image classifier model, we calculate the IPS performance for EASI-CiM as it scales across the four operating regions. As each inference translates to a single-frame classification, the following evaluations are measured in terms of Frames/second (FPS). Fig. 4-11c rep-

resents the scalable FPS performance achieved by the architecture and its associated energy efficiency, which is measured in terms of TOPS per Watt (TOPS/W), where the orange dashed plot represents the performance without temporal pipelining, and the blue dashed line represents performance with pipelining. For a lower stream rate, EASI-CiM can achieve an efficiency of 245 TOPS/W and can process approximately 35 FPS. In comparison with other classifiers (A)[41] and (B)[42] that provide similar performance, our architecture roughly outperforms by an order of 10. As we increase the stream rate to scale the performance of the accelerator, the energy efficiency drops due to the associated ASC computational units and memory peripheral costs. For instance, as we scale the stream rate from 10 kHz to 100 kHz, the overall increase in FPS is 3.6x, at the cost of a drop in energy efficiency by 48.3%. If we further scale the stream rate, we see a slight improvement in efficiency because the performance improvement outweighs the energy costs. At the highest operating region, the accelerator is capable of performing 248 FPS and at par with other image classifiers (C)[9] from a performance perspective. A critical contribution of our work is the ability to traverse an energy efficiency vs performance curve, which can be impactful for edge accelerators. From an application perspective, an image classifier such as an object detection camera will benefit from EASI-CiM due to its scalable efficiency. For instance, when the camera needs to operate under strict energy constraints due to a drop in energy reserve/battery levels, it can scale down its performance. This enables the accelerator to still perform meaningful classifications such as edge detection, which requires around 50 FPS, and operate at a higher energy efficiency. When the energy reserves are back to normal levels, the accelerator can perform high-feature classifications such as multi-face image recognition that would demand a higher performance of $\approx$ 200-300 FPS. This allows the accelerator to navigate the steep variations within an energy cycle and adapt its performance accordingly. This would be in stark contrast to conventional accelerators that freeze their workload execution when the energy reserves cross below a critical level.

EASI-CiM Area Metrics: Fig. 4-11d describes the area contributions made by the different components within the architecture. The evaluated area for the

accelerator is 142.28mm^2, with approximately 83.65% occupied by the CiM tile and memory peripherals. The area for an individual 1T1R cell within the CiM tile is evaluated based on the fabricated HfO_2 device area and is measured to be 26.14μm^2. Within the ASC computational units, the ReLU and softmax activation units account for 51.3% of the total 21.36mm^2. The primary reason behind this increased area penalty is due to the need for an additional $\Sigma\Delta$ core per activation unit to generate the required bias. If the network topology can account for this overhead and reorganize the bias values such that a majority of neurons share a similar bias (p=0), it would significantly reduce the area overhead. One benefit of the EASI-CiM architecture is the ability to directly apply the P_2S converter-generated streams onto the CiM tile as control vectors. This eliminates the need for additional conversion units and accounts for less than 5.2% of the ASC computational units area.

Stimulus-driven Execution: To visualize the effects of the *stimulus-driven work rate* within the accelerator, it's essential to analyze the relationship between the input "stimulus" feature vector and the rate of operations across network layers. Fig.4-12 displays a heatmap representing data flow and the corresponding arithmetic operational rate across the convolutional layers. The incoming array of pixel density values is transformed into an asynchronous stream vector array, where the stream frequency is proportional to the encoding described in Section4.3.1. These streams drive read operations on the CiM tile, programmed with convolution kernel weights. From the figure, we observe that the highest read operations required to generate the filter output in the first convolution layer correspond to stream vectors containing critical information, such as edge pixels, in the input image. Each read operation translates to a dot product within the convolution layer, indicating that the stimulus intensity directly impacts workload execution.

Similarly, the number of averaging operations per second is closely correlated to the intensity of filter outputs from preceding convolution operations, as higher intensities require increased processing. However, as one traverses through the layers, this correlation becomes less evident due to the progressive abstraction of features and the increasing number of channels. In deeper layers, critical features are distributed

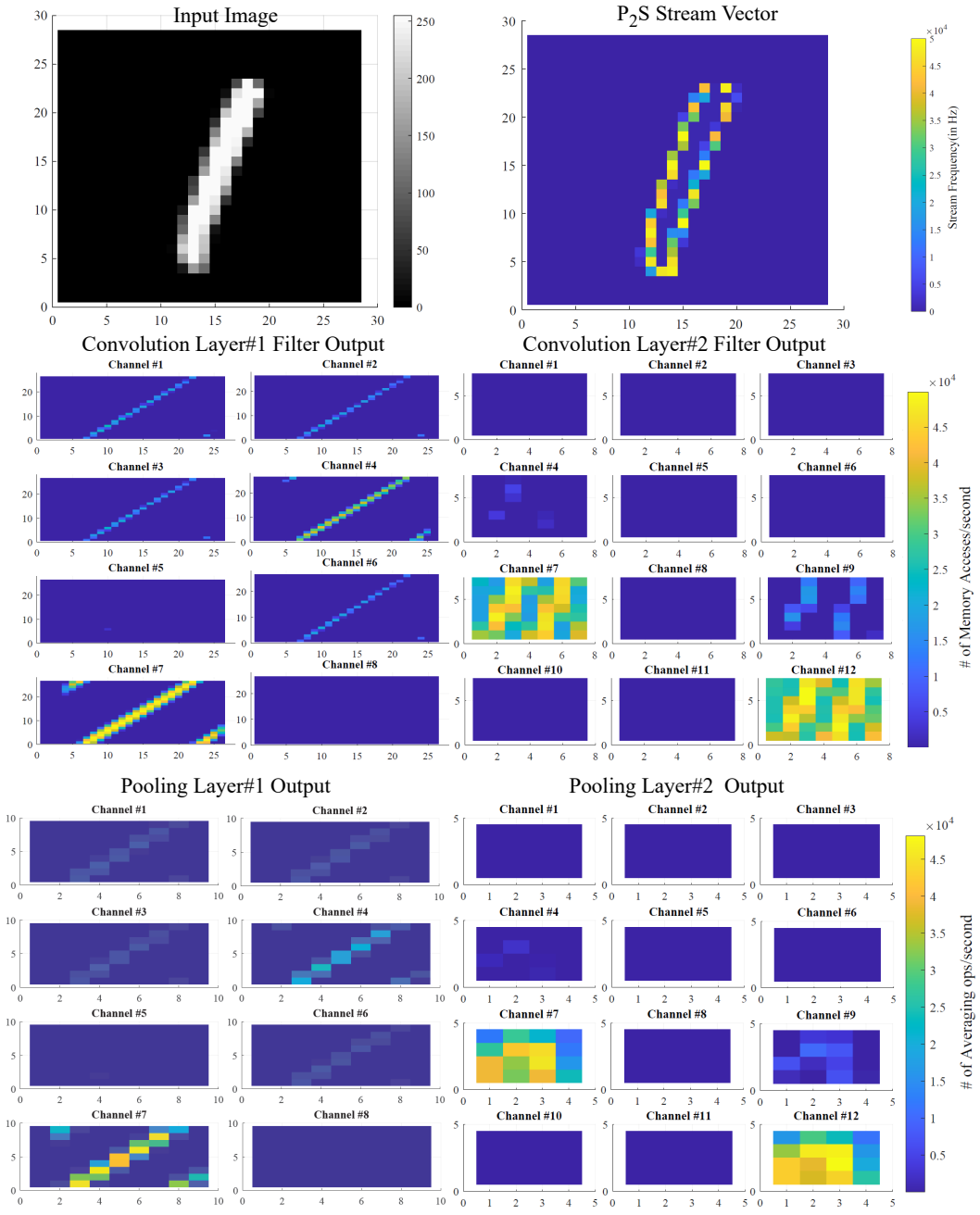


Figure 4-12: **Evaluation of stimulus-driven work-rate during MNIST classification.**

across multiple channels, and intermediary activations become less directly linked to specific input pixel values. This results in reduced determinism in layer-wise activations, as the criticality of individual neurons is governed more by hierarchical feature representations learned through the model rather than by stimulus intensity alone.

4.4.3 Word2Vector Mapping Engine based on EAS-CiM with Scalable Precision

Fixed-precision accelerators are inefficient for lower-priority workloads typical of edge ML applications, and using low precision during critical workloads can affect computational accuracy. This underscores the need for variable precision accelerators, and recent works have exhibited a balance between hardware complexity and precision scaling [7, 45, 61]. Most of these approaches rely on reconfigurable hardware fabric, which often results in an increased silicon footprint and leakage power, reducing energy efficiency. Unlike traditional approaches that rely on additional hardware, we scale the encoding precision through a *latency vs precision* design approach (as mentioned in Section. 3.2), enabling high precision during peak workloads and energy savings during lighter tasks. To evaluate this aspect, a CiM-based precision-scalable architecture is implemented and is optimized for Word2Vector algorithms, enhancing edge NLP tasks with efficiency measured in TOPS/W.

To assess the precision scaling of the CiM tile architecture and its computational efficiency, we accelerated the Word2Vec algorithm [11][57], widely used in NLP for learning word embeddings—dense vectors that capture semantic relationships. The algorithm encodes words into high-dimensional vectors (Fig. 4-13a), and during inference, this vector is compared to trained embeddings to determine semantic relations. The precision of these embeddings is critical for accuracy, and scaling it during inference is key to optimizing Word2Vec models [75] for edge deployment. To evaluate this algorithm, we utilized the Text8 dataset [1], which is a preprocessed version of the first 100 million Wikipedia characters and is widely used to provide a large, real-world corpus that helps capture meaningful semantic relationships. We train the Word2Vec

model using the Text8 dataset, then scale and quantize a 200-word long subset of the embeddings to INT4/8/12/16 formats.

In our framework, we achieve scalable input data precision by extending the operation period, T_o. Since training and encoding reference vectors is a one-time cost, dynamically adjusting the precision of these reference vectors is impractical. Therefore, we program the reference vectors in 16-bit integer (INT16) format as conductance values in the crossbar array. This high-precision INT16 embedding is split into separate positive and negative arrays and mapped accordingly to distinct instances of our Python-emulated CiM tiles. By keeping reference vector precision fixed and scaling only the precision of the input vectors, we strike a balance between computational efficiency and model performance. During testing, we similarly split INT4, INT8, and INT12 input test vectors into positive and negative embeddings, which are then applied as ASC streams to compute the dot product similarity with the pre-encoded reference embeddings. Extending the observation period T_o increases the aggregation of redundant pulses, effectively enhancing stream precision. This scaling approach allows us to evaluate semantic relations between mapped words through the dot product operation with varying input precision.

We assess the impact of quantization on semantic distortion and vector ranking mismatch using two primary metrics: average Euclidean distance and Rank Biased Overlap (RBO), both measured against 16-bit precision embeddings as the baseline. Figure 4-13b demonstrates a significant 42.16% reduction in the average Euclidean distance as stream precision increases from 4 to 12 bits. This improvement in Euclidean distance reflects a closer alignment of lower-precision embeddings with the high-precision reference, thus enhancing semantic similarity. Additionally, the reduction in Euclidean distance is accompanied by a 1.27x decrease in the standard deviation of embedding mismatches, indicating a more consistent and reliable similarity computation across the tested embeddings. This notable reduction in variance further underscores the stability and robustness of higher-precision streams in preserving semantic meaning. However, these improvements in accuracy are achieved at a cost: increasing precision from 4 to 12 bits leads to a substantial 16.7x increase in

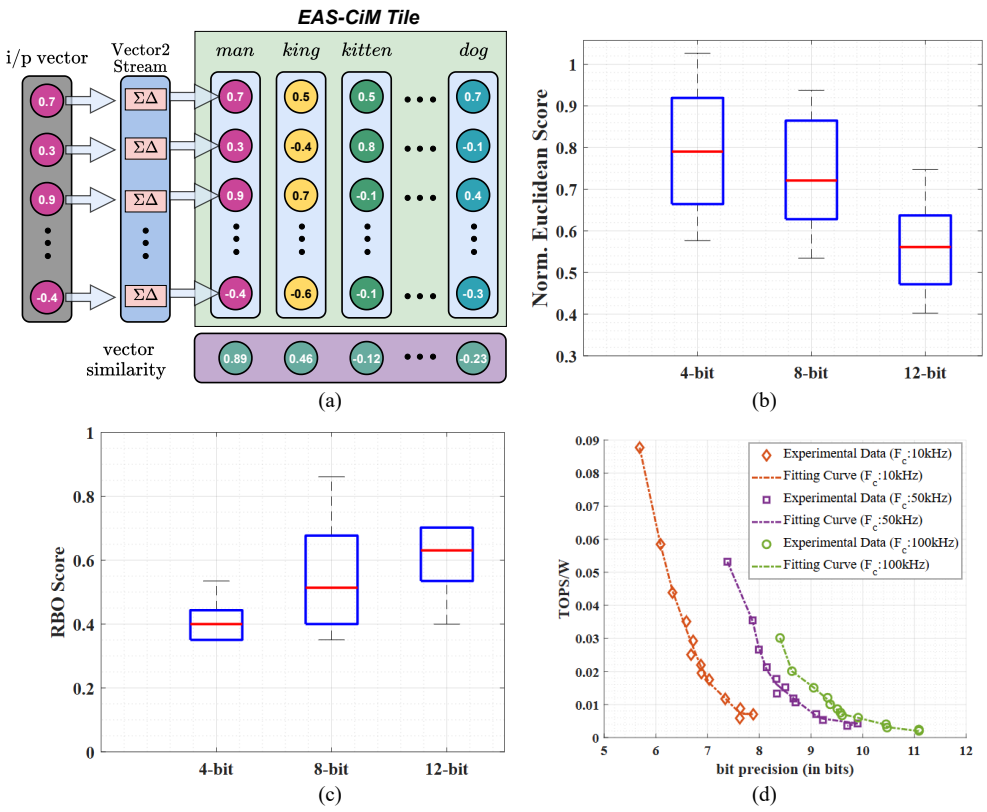


Figure 4-13: **(a) Mapping Word2Vec to EAS-CiM, bit precision vs (b) Euclidean score ,(c) RBO score, and (d) TOPS/W**

the observation period T_o and incurs a 5x rise in frequency scaling costs.

The effects of quantization reduction extend beyond semantic accuracy and directly impact the ranking of word vectors, as indicated by our analysis of RBO scores. Quantization reduces the rank stability of word embeddings, causing a reordering in the similarity-based ranking of words—a direct consequence of lowered precision. Figure 4-13c illustrates a 53.45% decline in the RBO score as input precision is reduced from 12 to 4 bits, highlighting a significant degradation in ranking consistency. This loss in ranking fidelity indicates that lower-precision embeddings fail to capture finer semantic distinctions, which are crucial for applications where ranking is critical. The drop in RBO score as precision decreases reinforces the importance of encoding precision for applications requiring accurate similarity assessments between words.

From a computational efficiency standpoint, the gains in semantic accuracy and ranking fidelity at higher precision come at a steep cost. Increasing precision results

in a 43.23x reduction in computational efficiency, as shown in Figure 4-13d. This efficiency drop arises from the increased observation period required to accumulate additional pulses and improve stream precision. While higher precision enhances semantic accuracy and stabilizes rankings, it imposes considerable costs in terms of computational time and frequency scaling, emphasizing the trade-off between precision and efficiency. This trade-off is significant, as it enables flexibility in balancing precision requirements with efficiency goals based on the specific workload.

Overall, the results underscore the architecture's ability to scale computation precision by dynamically adjusting the aggregation time of encoded streams, allowing for flexible design choices that cater to varying levels of precision and efficiency. Higher precision significantly enhances semantic accuracy and ranking performance, making it suitable for tasks with stringent accuracy requirements. However, the corresponding drop in computational efficiency indicates that for tasks where efficiency is a priority, lower precision may be preferable. This balance between precision and efficiency highlights the versatility of the EAS-CiM architecture in adapting to diverse workload demands and illustrates its potential for optimizing performance in edge computing applications where resource constraints are paramount.

Chapter 5

$\Sigma\Delta\Sigma$ Neuron: Noise-Resilient Spiking Neural Encoding

5.1 Sigma-Delta and Sigma-Delta-Sigma Neuron: A Comparative Analysis

In accordance with Section 2.4, recent years have witnessed the emergence of advanced spiking neural encoding techniques tailored for specific applications. Our research contributes to this landscape by closely aligning our encoding technique with temporal-based neural encoding, emphasizing the encoding of features within the temporal signature of an asynchronous stream.

The evolving landscape of temporal encoding techniques exhibits a significant divergence, marked by methodologies relying on the differential amplitude between consecutive feature samples in a time series format. This paradigm shift is exemplified in notable instances such as the implementation of spiking neural encoding-based Dynamic Vision Sensor (DVS) cameras [74] and audio cochlear feature extractors [35], both incorporating spiking networks with Asynchronous Delta Modulation (ADM). Building upon this foundation (depicted in Fig. 5-1a), researchers have introduced the asynchronous Sigma Delta (SD) encoding technique, characterized by partitioning neural encoding into two distinct phases: Sigma decoding and Delta encoding.

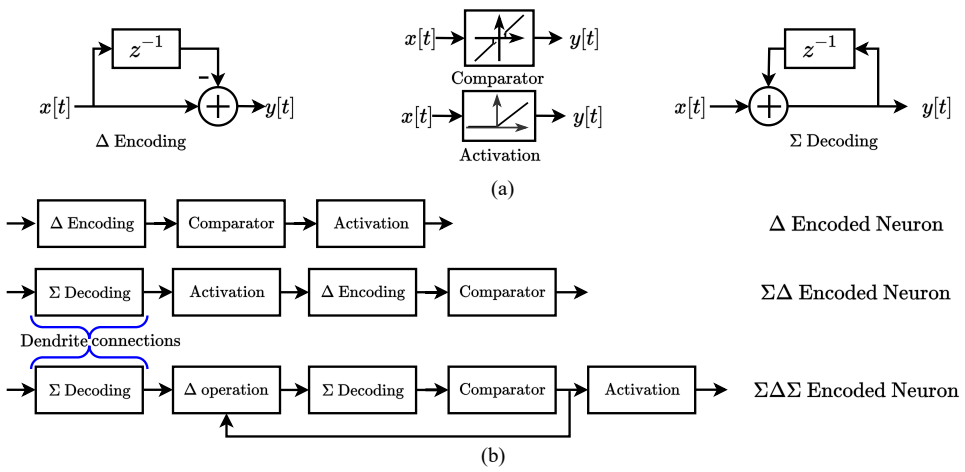


Figure 5-1: **(a) Symbolic representation of fundamental building blocks for differential encoding (b) Neural Encoding representation for Δ, ΣΔ and proposed ΣΔΣ neurons.**

The versatility of the SD encoding technique is highlighted through its successful application in video processing applications [63], showcasing substantial enhancements in energy efficiency. This progression signifies a significant stride in the temporal encoding landscape, reflecting a dynamic evolution with promising implications for diverse applications. From an encoding standpoint, a defining feature that distinguishes these neural encoding techniques from existing methods is how activations become increasingly sparse as information traverses successive layers of neurons. This phenomenon is driven by the encoding techniques' mechanism, where a post-neural spike occurs when the incoming feature/activation surpasses a predefined Δ threshold or deviates from its previous activation value by a certain Δ (as shown in Fig. 5-1b).

We extend the capabilities of delta-modulated neurons by introducing a novel evolution in this encoding approach termed Sigma-Delta-Sigma (ΣΔΣ/SDS) neural encoding. Our proposed encoding method stands out for its adeptness in filtering out noise-like components from critical features, achieved through a unique feedback mechanism and intrinsic delay within the encoding technique. Our inspiration for developing this schema stems from observations of noise-invariant neurons found in specific bird species [47] and resonant frequency selective traits observed in primate auditory systems [10]. Our research aims to emulate and enhance these functionalities using our proposed scheme, aiming to construct noise-robust spiking models capable

of feature extraction even in the presence of noisy input stimuli.

From an applications perspective, it is imperative to understand the motivation for developing a noise-invariant neural encoding schema. In Machine Learning (ML) models, noise is often introduced into training features to prevent over-fitting and to ensure robust inferences when faced with non-deterministic variations within test features. However, injecting an inaccurate noise profile during training can result in under-fitting and distorted feature representations, ultimately undermining model performance [18]. Through our proposed neural encoding schema, we aim to train models on ideal feature sets without the need for noise injection, while still being able to perform feature extraction with distorted, low-quality data. During testing, the neural connections prioritize the transmission of significant features while attenuating noise components. By implementing this neural encoding approach, our objective is to develop noise-resilient spiking neural network models capable of performing classification tasks even with lower-quality incoming features. Through these innovations, we contribute to advancing the field by bridging the gap between neural encoding techniques and practical neural computing applications.

5.2 $\Sigma\Delta\Sigma$ neuron: Noise-robust Encoding schema

In this work, we draw upon the aforementioned encoding techniques and propose our novel SDS neural encoding (Fig. 5-1d). At this juncture, it is important to understand how the SDS neuron is an evolution in the class of delta-modulated neurons. To begin with, the SD neuron is essentially a Δ modulated neuron with a Σ decoding block that integrates dendrite connections from multiple pre-synaptic neurons. Our proposed SDS neuron is essentially a $\Delta\Sigma$ modulated neuron with a similar pre-synaptic Σ decoding block. Another critical difference within our encoding technique is the internal Δ operation block. Within the previously mentioned Delta and SD neuron classes, the delta operation is essentially performed on the current input and a delayed version of the input feature. However, within the SDS neuron, this delta operation performs a negated addition between the input feature and the output spike.

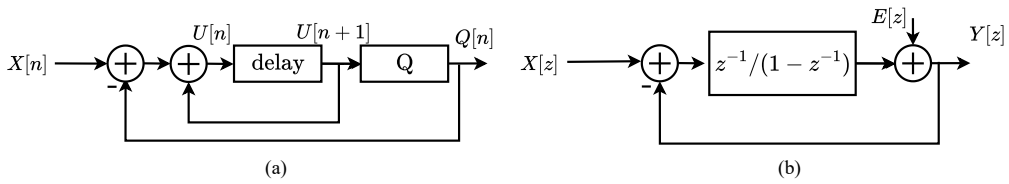


Figure 5-2: **Schematic representation of (a) Time domain signal, and (b) z-transform diagram for the encoding scheme.**

From an encoding perspective, the SDS neural encoding stands out from its predecessors due to two distinct signal processing features. Firstly, a portion of the soma output from the comparator element is subtracted from the input stimuli at the dendrite terminal. This introduces a negative feedback mechanism, represented by the Δ operation block within the encoding scheme. Secondly, an additional integration phase is embedded within the encoding process which accumulates the aforementioned Δ operation over multiple time steps. These elements effectively impart two different filtering characteristics to the incoming stimuli and noise-like features.

To examine the filtering properties of the neural encoding scheme, we construct the time-domain representation and the equivalent discrete Z-transform signal flow diagram, as depicted in Fig. 5-2. By analyzing the signal flow, we can derive the output $Q[n]$ as follows:

$$\begin{aligned} Q[n] &= X[n-1] + Q[n] - U[n] - Q[n-1] + U[n-1] \\ &= X[n-1] + e[n] - e[n-1] \end{aligned}$$

, where X[n], U[n], and e[n] represent the discrete input stimuli, integrator input, and error vector, respectively. The z-transform for the designated output Y(z) is:

$$Y(z) = X(z)z^{-1} + E(z)(1 - z^{-1}) \implies \frac{Y(z)}{X(z)} = z^{-1} \tag{5.1}$$

$$\implies \frac{E(z)}{X(z)} = 1 - z^{-1} \tag{5.2}$$

, where Eq. 5.1 and Eq. 5.2 represent the deduced signal and noise transfer function, respectively. As evident from these equations, the input signal features undergo a low-

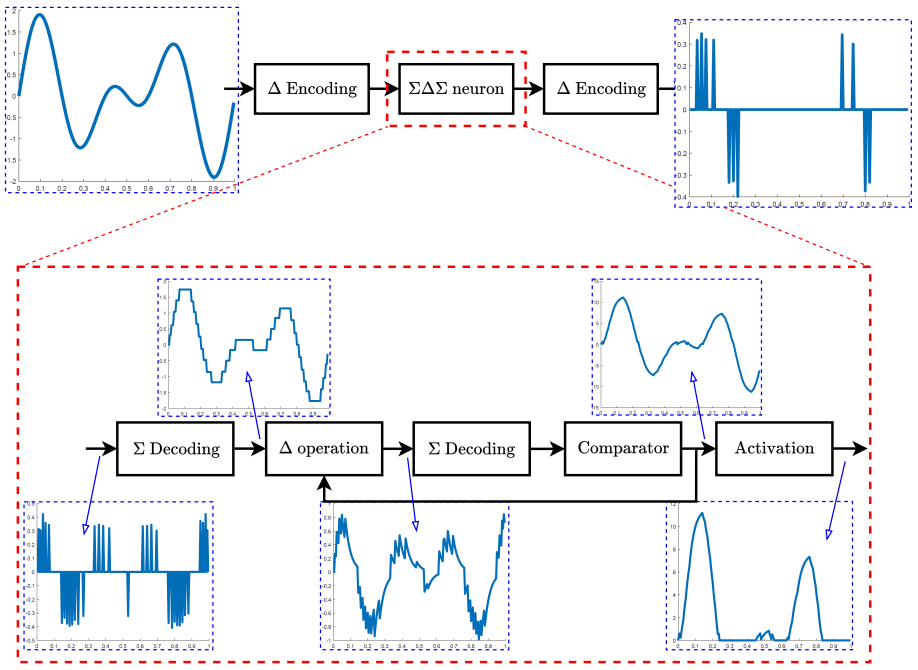


Figure 5-3: **Internal feature transformation and transient operative characteristics of the proposed SDS encoding schema.**

pass filtering operation while the noise features are shifted towards higher frequency regions and attenuated at the lower end of the spectrum. This mechanism allows us to diminish noisy components from critical features, provided these elements lie beyond the corner frequency of the low-pass filter and are modulated by the feedback strength and integration period of the neuron. It is noteworthy that, to maintain a similar sparse feature density profile to its predecessors, each SDS neuron is linked with a delta encoding block depicted in Fig. 5-3, showcasing the internal transformation occurring to an incoming feature at each encoding step within the neuron.

5.2.1 ΣΔΣ performance against noise profiles

The analytical model shows that the discrete implementation of the ΣΔΣ (SDS) neural encoding exhibits a low-pass filtering characteristic. For practical ML applications, it's crucial to understand how this proposed encoding responds to different noise profiles, as real-world data is rarely noise-free. The two most common noise profiles used to simulate real-world conditions in ML models, especially for audio and sensory time-series data, are Gaussian and Pink noise. Gaussian noise, or white noise,

has a flat spectral density, meaning it affects all frequencies equally, making it ideal for testing a model's robustness to random, high-frequency disturbances. Pink noise, in contrast, has a spectral density that decreases with frequency, closely resembling many natural and biological processes. This makes Pink noise particularly relevant for audio and sensing applications, where lower frequencies are prominent.

Incorporating these noise profiles into the evaluation of ML models is essential for simulating real-life conditions. Thus, understanding how the SDS encoding performs under these noise profiles is necessary before deploying it in practical ML scenarios. To this end, we use a fully connected (FC) neural population, where a single-tone signal is fed into the population's input, and the spiking activity at the output is measured. The weights in this layer are randomized with a set seed to ensure that no pre-trained noise filtering characteristics are imparted to the layer. The baseline spiking activity is recorded with an ideal (noise-free) input signal, serving as a reference for comparison. When noise is added to the input at varying SNR levels, deviations in the spiking activity from the baseline indicate potential encoding degradation, reflecting the filtering capability of the proposed neural encoding schema.

To quantify the encoding and filtering performance of the SDS neuron under these conditions, the Kullback-Leibler (KL) divergence is used. KL divergence measures how one probability distribution diverges from a reference distribution, making it useful for assessing noise impact on neural encoding fidelity. When noise alters the input signal, the resulting spiking patterns may deviate from the expected, noise-free distribution, which ideally represents the true information content. Higher KL divergence values indicate greater divergence between the noisy and baseline distributions, suggesting that noise has degraded the encoding fidelity.

To evaluate the performance of the SDS encoding against other spiking encoding techniques, a similar fully connected (FC) layer was simulated using a LIF spiking model and $\Sigma\Delta$ neural encoding. Fig. 5-4 presents the KL divergence scores for the three encoding schemas across three FC layer sizes (10, 50, and 100 neurons) under varying SNR levels, where the input signal was injected with AWGN and Pink noise. This evaluation reveals four key findings relevant to encoding fidelity.

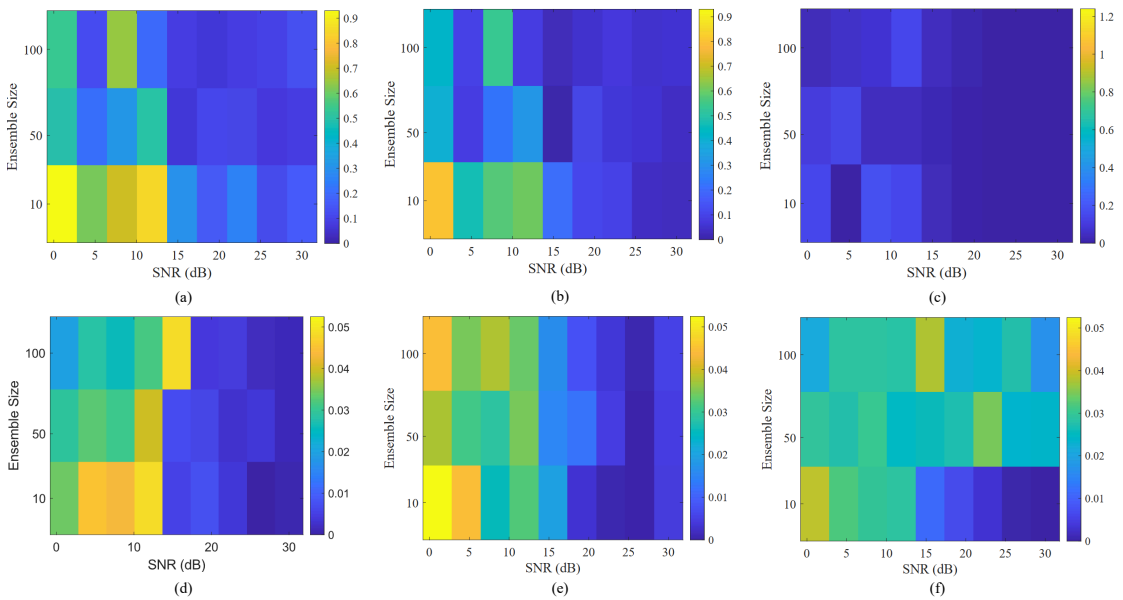


Figure 5-4: **KL divergence heatmap across varying SNR values and ensemble sizes for AWGN profile with (a) LIF, (b) ΣΔ, and (c) ΣΔΣ encoding, and for Pink Noise profile with (d) LIF, (e) ΣΔ, and (f) ΣΔΣ encoding.**

Firstly, as anticipated, spiking activity degrades (indicated by higher KL scores) as SNR decreases, regardless of the encoding type, noise profile, or neural population size. However, this degradation diminishes with larger neuron populations, suggesting that increased layer sizes introduce redundancy, which helps mitigate information loss caused by noise. Secondly, from the perspective of SDS encoding, neural encoding is more robust to a flatter noise profile, such as AWGN, compared to a low-frequency dominant pink noise profile. This trend aligns with our analytical model: in the case of pink noise, the SDS encoding encounters more "in-band" noise due to its low-pass filtering characteristics. This effect becomes evident when comparing KL scores across similar neural population sizes and SNR conditions for SDS encoding. For example, under the worst conditions (smallest neural population of 10 and lowest SNR of 1 dB), the network experiences approximately an 18.67x relative degradation in encoding quality.

Thirdly, compared to other encoding mechanisms, SDS shows superior noise rejection. When injected with pink noise and using a layer size of 10 neurons, the SDS-based implementation exhibits a 2.13x relative degradation, whereas the SD

and LIF implementations experience higher degradations of 4.62x and 4.27x, respectively. This emphasizes the inherent noise-filtering capacity of SDS encoding, even in an untrained model. Finally, increasing the neural population size further amplifies these advantages. As shown in Fig. 5-4c, SDS encoding maintains spiking activity deviation under 1.21x with larger populations, compared to 3.21x and 3.87x degradations in LIF and SD encodings, respectively. This underscores the robustness and effectiveness of SDS encoding under noisy conditions.

5.2.2 Effect of signal parameters on $\Sigma\Delta\Sigma$ neural encoding

In the previous section, we introduced a proxy measure for the filtering characteristics of the $\Sigma\Delta\Sigma$ (SDS) neuron and examined the impact of additive white Gaussian noise (AWGN) on its ability to retain encoded information. Due to the inherent Δ characteristics, applications involving audio and time-series data align particularly well with the spiking behavior of our proposed encoding schema. However, to thoroughly assess the suitability of this encoding schema for audio-based applications, it's essential to expand the evaluation to include additional parameters critical for audio fidelity, such as *loudness* (inversely related to the distance of the source) and *pitch selectiveness* (related to signal frequency). These parameters are vital for understanding the encoding schema's robustness across varying audio signal characteristics.

Accurately evaluating the fidelity of this neural encoding schema to these parameters requires more than a single spiking neuron. Thus, we employ an ensemble model—a group of interconnected spiking neurons that work in concert to process and encode information. Each neuron within the ensemble responds to distinct aspects of the input signal by generating spikes that reflect specific signal features. The performance of this ensemble can be quantitatively assessed through metrics such as spike rate and normalized mean square error (NMSE), which provide direct measures of how accurately information is reconstructed from spike-based signals. In this context, loudness is defined as an amplitude ratio, normalized to the maximum amplitude encountered during training, which establishes a consistent baseline for comparison.

In Fig. 5-5a, we illustrate how loudness impacts the reconstructive ability of the

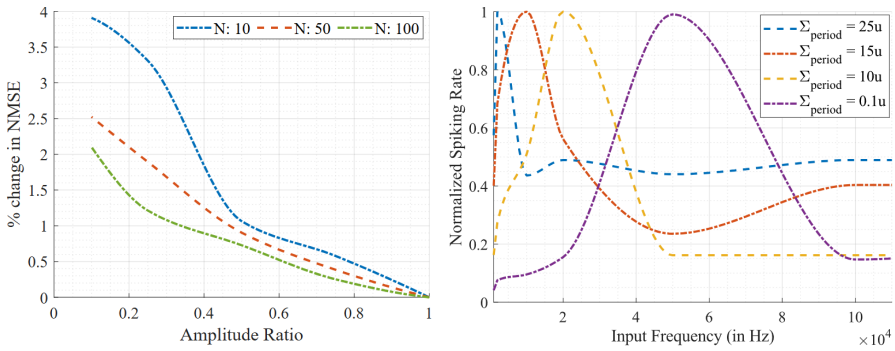


Figure 5-5: **Effect of (a)Loudness of signal on NMSE (b) Pitch selectiveness on information encoding quality.**

neural ensemble, quantified through NMSE. This analysis reveals two key findings. First, with smaller neuron populations, NMSE sharply increases as amplitude decreases, highlighting a significant drop in reconstruction capability under low-loudness conditions. Specifically, the evaluation shows a 4% increase in NMSE when the ensemble size is limited to 10 neurons, suggesting that even in the absence of noise, the SDS model struggles to accurately reconstruct low-amplitude signals with a minimal neuron count. However, as we scale up the neural population ($N \geq 50$), the relationship between amplitude and reconstruction becomes more linear, indicating a more predictable and stable performance. This outcome is particularly beneficial for two reasons. First, a linear relationship enables the introduction of a gain or "boost" at the front end of the accelerator, allowing the input signal to reach an amplitude that provides the precise feature encoding required by the application. Second, for applications where the expected signal attenuation is known, scaling the neural population from the point where linearity begins serves as an effective design parameter. This dual approach—boosting the input amplitude or adjusting the neural population size—provides flexibility in configuring the model to achieve accurate encoding based on the application's specific requirements.

To examine the filtering effects and scalability of the SDS neural encoding schema, we experimented with analogous to amplitude scaling, but instead of adjusting the input amplitude of the signal, we varied its frequency. The low-pass frequency response

of the neuron was modulated by altering the delay within its feedback loop. This was achieved by increasing the integration period—or summation window—in the Sigma block, emulating the integration process within a digital asynchronous environment like Lava. The integration period here can be roughly defined as the number of samples accumulated within a sliding summation window. This adaptable summation window showcases inherent flexibility in SDS neurons, allowing control over frequency sensitivity, which is beneficial for applications such as audio processing.

In Fig.5-5b, we illustrate the results of this analysis, using the normalized spiking rate of the ensemble as an indicator to approximate the filtering behavior. A key finding is that, instead of observing a simple low-pass filtering effect, we see a band-pass filtering response. This result highlights an aspect not fully captured in the initial analytical model: the input signal undergoes a Δ encoding operation before entering the SDS ensemble (as detailed in Section 5.2). This Δ operation introduces a high-pass filtering effect that combines with the SDS neuron's low-pass behavior, resulting in an overall band-pass response.

By reducing the neuron's summation period, we decrease the delay within the feedback loop, enabling higher spiking activity in response to high-frequency signals. Conversely, increasing this period shifts the peak spiking activity towards lower frequencies, as indicated by the leftward shift in peak activity in Fig. 5-5b.

5.3 Incorporating stochastic devices within $\Sigma\Delta\Sigma$

Replicating characteristics of spiking neurons, such as stochastic variations in firing threshold [5], is challenging with conventional silicon devices. Extensive research has explored the use of emerging device technologies to emulate these characteristics within spiking neuron models. For example, Maruan Al-Shedivat et al. [6] proposed a memristor-based stochastically spiking neuron. In their approach, a memristor model captured intrinsic stochastic behavior consistent with experimentally observed phenomena. They implemented a stochastic Spike Response Model (SRM)–based neural soma circuit, leveraging memristive non-determinism for efficient spike generation.

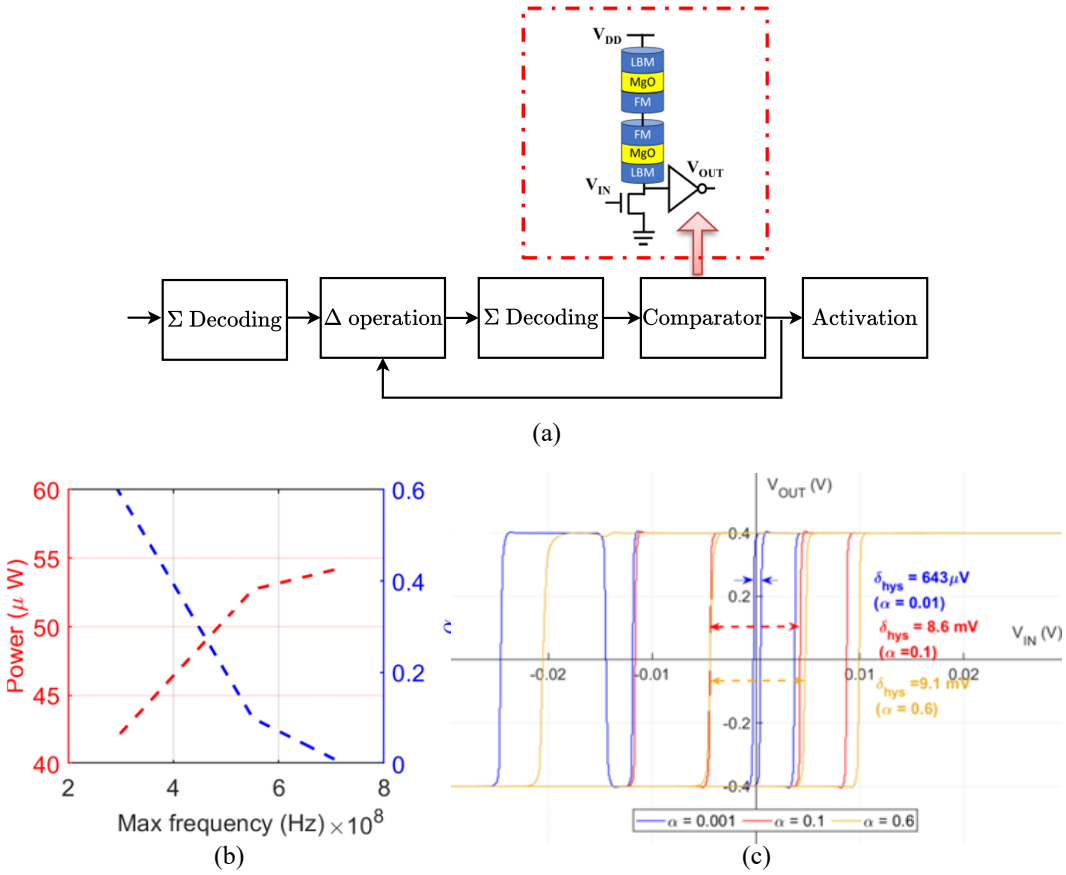


Figure 5-6: **(a)Double MTJ-based BSN module, (b) power and α trade-off for varying spike rates, and (c) hysteresis of BSN module for different values of α.**

Their results demonstrated the feasibility of using such neurons for probabilistic sampling and adaptive pattern recognition, contributing to scalable neuromorphic systems. To that extent, we propose a novel double magnetic tunnel junction (MTJ) based spiking neuron that leverages low barrier magnets (LBM) stochasticity to emulate the firing threshold variability. Our research replaces the comparator unit within the previously mentioned SDS neuron with a stochastic-threshold-based comparator element.

5.3.1 BSN inspired SDS Neuron Modeling

To develop an MTJ-based stochastic comparator, we build upon the previously designed Binary Stochastic Neuron (BSN)[28]. Integrating the BSN as a stochastic

threshold comparator within the SDS neuron requires controlling the BSN's switching frequency to manage the delay in the feedback loop of neural encoding, thereby influencing the filtering characteristics of the spiking neuron. This ensures minimal signal degradation within the SDS neuron's internal feedback loop. Using SPICE-based LLG solvers, we found that incorporating a second MTJ into the BSN (see Fig.5-6a) results in four possible resistance combinations based on the alignment of low barrier magnets (LBM) with the fixed layer (FM): $R_P - R_P - R_P$ for HIGH voltage, $R_{AP} - R_P - R_{AP}$ for LOW voltage, and two intermediate states: $R_{AP} - R_P - R_P$ and $R_P - R_P - R_{AP}$. These intermediate states prompt the NOT gate to adjust them, resulting in a 'sparser' firing rate, which can be used as a tuning mechanism to control the neuron's stochasticity, and its electrical behavior can be formulated as;

$$V_{\mathrm{OUT}}(t)_{\mathrm{BSN}} = \frac{V_{DD}}{2}\left(\tanh\left[b \cdot \{V_{\mathrm{IN}}(t) + V_{\mathrm{IN}}(t-1)\}\right] + a \cdot V_{\mathrm{rnd}}(t)\right) \tag{5.3}$$

To manipulate the stochastic behavior of the BSN-based SDS neuron (BSN-SDS), we vary the Gilbert damping coefficient (α) of the LLG equations [28], which alters the switching frequency of the LBM. By adjusting the α values, we can modify the average firing rate, but this also impacts the associated power consumption, as depicted in Fig. 5-6b. To implement the BSN-SDS neuron in a spiking model, we convert the device behavior into an analytical model based on the BSN [28] and the SDS model. To ensure the accuracy of this analytical model, we calibrate it against SPICE simulations and measure the hysteresis (δ_{hys}) of the BSN-SDS neuron for different α values, as shown in Fig. 5-6c. This calibration controls the internal delay within the neuron and subsequently affects the rate of change of $V_{IN}(t)$ to the BSN.

5.4 Benchmarking $\Sigma\Delta\Sigma$ Neural Encoding & Results

5.4.1 Noise-Robust Information Encoding & Decoding with Ensemble model

To evaluate the noise-filtering capabilities of our proposed encoding, we utilize an ensemble-based signal regenerative model. This ensemble model serves the dual purpose of assessing feature preservation and noise filtering simultaneously while extracting features from the incoming signal. By employing the ensemble model, we aim to replicate real-world scenarios where the nature of noise during inference is uncertain necessitating training with idealized features and subsequent fine-tuning for non-deterministic noise conditions. However, using our proposed neural encoding, models trained with idealistic features still demonstrate robust noise performance during testing, obviating the need for fine-tuning. In Fig. 5-7a, we illustrate the pair of ensemble models employed to validate our hypothesis. Ref. Ensemble 1 processes the incoming time-series feature through a fully connected layer of neurons, generating a spike train. They are then fed into the second ensemble with a similar structure, which, under ideal conditions, reconstructs the original signal from the incoming spike activations. By implementing the ensemble model, we ensure that the SDS neuron can faithfully encode information in the spiking domain and decode the spike train to retrieve the original signal.

During the training phase, the first reference ensemble model weights are updated using idealistic features with an SNR value $\geq$ 100dB. The weights for the second ensemble model are obtained from the first ensemble through a transpose operation and a scaling factor g_{ij}. In the subsequent testing phase, Reference Ensemble I encounters degraded signal features through the injection of artificial noise profiles and is characterized by their respective SNR values. The signal output regenerated by Ensemble II is then assessed for its fidelity against the ideal signal, measured by the normalized mean square error (NMSE). The ensemble model is trained using the Nengo PyTorch-Spiking package, which facilitates the emulation of spiking neuron models based on

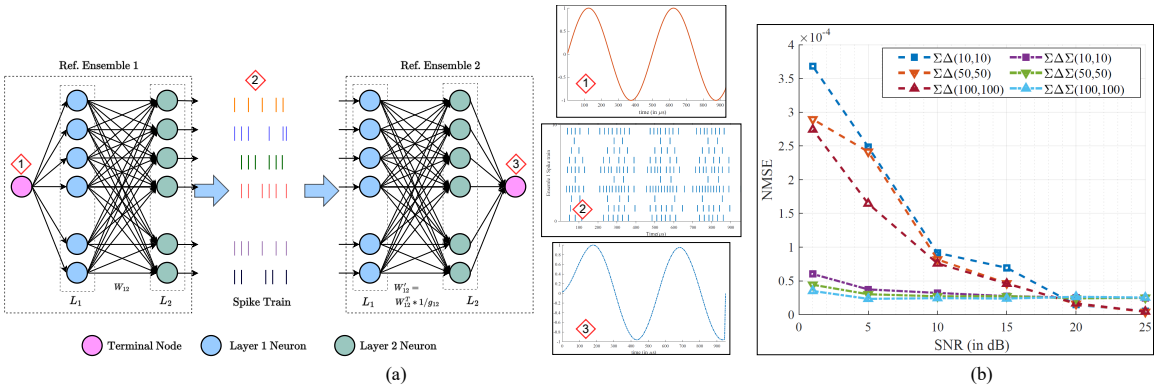


Figure 5-7: **(a)Implemented Ensemble model with transient feature transformation and (b) Robustness (NMSE variation) of ensemble model against SNR degradation.**

the ΣΔ neural encoding scheme. To evaluate the performance of the neural encoding, we compare our results with a baseline ensemble model featuring ΣΔ neuron banks. Additionally, we vary the ensemble size to assess the network's performance as more features are extracted from a larger neuron bank. Fig. 5-7b demonstrates the network's performance across different test cases, with the reconstruction error (NMSE) calculated for input features spanning a range of SNR values. The critical findings within this simulation can be summarized as follows:

In a controlled setup where the ensemble size remains constant, the SDS neuron-based ensemble model demonstrates a remarkable ability to outperform the SD baseline model. Specifically, at an SNR of 1 dB for the input features, the SDS ensemble achieves an average performance improvement of approximately 6.2× over the baseline. This result underscores the resilience of the SDS-based neural encoding, which is adept at filtering out noise-like artifacts even when the spectral power of the noise closely approximates that of the signal. The enhanced noise suppression capabilities of the SDS neurons allow the ensemble to reliably process and retain information, thereby elevating the model's robustness in noisy environments. For an ensemble with a fixed neuron count, specifically N=50, the performance degradation as the SNR is reduced from 20 dB to 1 dB is markedly lower in the SDS-encoded model compared to the baseline. When using SDS encoding, the performance degradation rate is approximately 1.482×, in stark contrast to the 57.56× degradation observed with the baseline model. This observation confirms the capability of the SDS neuron-

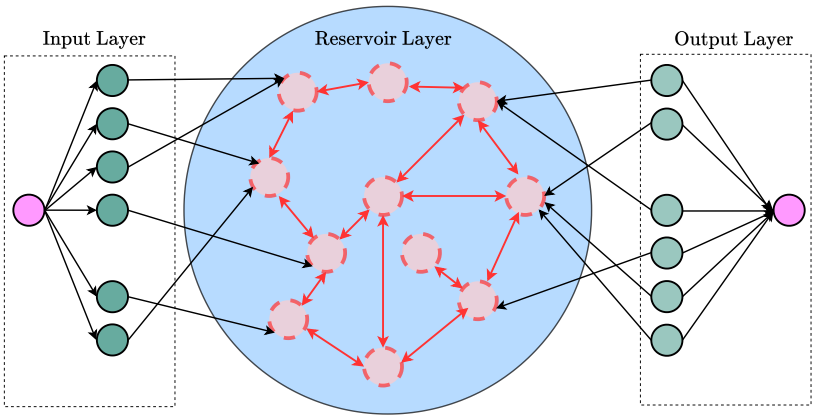


Figure 5-8: **Block level representation of a LSM model**

based encoding scheme to uphold stable, regenerative performance across varying noise conditions. Such consistency reflects the SDS ensemble's efficiency in dealing with low-quality input data, even though the model was initially trained on higher-quality, ideal data. The reduced sensitivity to SNR variations indicates that the neural encoding scheme is well-suited for applications where the data quality may fluctuate, maintaining reliable feature representation and model output.

In examining the effect of ensemble size on feature extraction capabilities, it becomes evident that increasing the neuron bank size enhances performance. When the number of neurons per layer(N) is raised from 10 to 100, there is a notable improvement in the average Normalized Mean Square Error (NMSE) by a factor of 1.418. Comparatively, for the baseline model, the NMSE reduction is around 1.425×. These values suggest that the SDS encoding scheme supports effective feature extraction even as the ensemble size scales. This consistency implies that SDS neurons continue to effectively filter noise while isolating meaningful signal features, enhancing the overall robustness of the ensemble model's feature extraction mechanism. This scalability in feature extraction with increasing neuron count demonstrates the SDS ensemble's capability to retain performance while handling larger neuron populations, an essential trait for processing complex data or operating under inferior conditions.

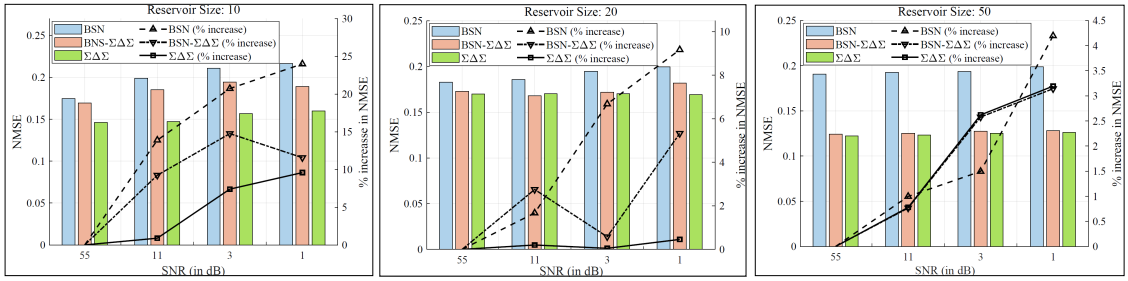


Figure 5-9: **Performance of LSM with various neural encoding schema against SNR degradation for (a) 10x10 (b) 20x20 (c) 50x50 reservoir sizes.**

5.5 Liquid State Machine and interfacing with emerging device models

To assess our proposed neural encoding in a complex learning model, we develop a Liquid State Machine (LSM) based on the Echo State Network (ESN) model outlined in [48]. The LSM model is structured with three layers: the input layer, which maps incoming time-series features to a collection of recurrently connected neurons forming the "reservoir" layer of the network and the output layer. Both the input and reservoir layers have randomized weights distributed according to an initial seed matrix. Connections from the reservoir layer extend to the output layer, where weights are trained using a pseudo-inverse approach and optimized to minimize the least squares fitting (LSQ) error.

Fig. 5-8, represents the LSM model, and in our simulation, we test out the model with different reservoir sizes of 10,20 and 50. To evaluate the dual capabilities of our neural encoding, we train the output layer based on an ideal feature set (SNR $>$ 50 dB) for a training period that constitutes roughly 75% of the entire feature set available for both training and testing. During the testing phase, artificially corrupted features are fed to the predictive model based on a pre-determined SNR value. We evaluated the model against three different neural encoding schemas; the BSN model based on which the original ESN model is generated, our proposed BSN-$\Sigma\Delta\Sigma$ neuron, and conventional $\Sigma\Delta\Sigma$ neuron. Our simulations, illustrated in Fig. 5-9a-c, represent the robustness of the network for various reservoir sizes as the SNR is varied from 55

(clean signal) to 1 dB, with the following observations being made:

- **Encoding variations minimize as reservoir size increases:** For smaller reservoirs (N=10), the conventional $\Sigma\Delta\Sigma$ neuron outperforms the BSN adaptation of our proposed encoding scheme by an average factor of 1.26x. However, this disparity diminishes as the reservoir size increases (N=50). With larger reservoirs, both encoding schemes demonstrate similar NMSE values and robustness to SNR degradation (% increase in NMSE). This suggests that while the increased stochasticity within the BSN comparator may initially lead to losses when the model is small, these effects are minimized as the model becomes capable of extracting more features from the reservoir. As a result, the true noise-filtering capabilities of the encoding scheme become more prominent.

- **Resilience to noise degradation has diminishing returns:** At a given reservoir size and SNR value, the proposed encoding scheme outperforms the baseline (BSN) version by an average of 1.14 - 1.52x, showcasing its robustness under specific noise conditions. When evaluating the model's robustness within a fixed reservoir size as noise injection increases, we observe greater benefits with smaller reservoirs (N=10,20). Here, the average increase in NMSE is $>$ 11.15x smaller than the corresponding change in the baseline implementations. However, for a larger reservoir (N=50), this advantage diminishes to 2.16x. This underscores the model's capability to learn from inferior features regardless of the encoding scheme.

Chapter 6

Conclusions and Future Work

6.1 Summary

There is a critical need for innovative computational architectures tailored to accelerate edge machine learning under the constraints of high-dimensional feature vectors and substantial memory demands. This work was motivated by the persistent "memory-wall" issue, where the performance gap between computational power and memory bandwidth challenges efficient operation in edge environments, especially those with limited power and intermittent energy availability.

To tackle these challenges, we proposed a shift from the traditional balance of energy efficiency versus accuracy toward a model that prioritizes energy efficiency and performance scalability. This reimagined balance is essential for next-generation ultra-edge applications that demand flexible, adaptive architectures. Additionally, by exploring compatibility with spiking neural models and temporal encoding schemes, we demonstrated the potential of bio-inspired computational frameworks to enhance energy efficiency and achieve sparse data representation.

Central to our approach was the development of an event-driven Compute-in-Memory (CiM) architecture, integrated with Asynchronous Stream Computing (ASC) principles, introducing a "stimulus-driven workload execution" model. This model allows computational rates to adapt dynamically to the amplitude and temporal characteristics of incoming data, thereby enabling energy-efficient operations and real-time

responsiveness even in energy-variable scenarios.

A primary contribution of this work was the presentation of a systematic approach and analytical model for optimizing the asynchronous stream generation module, focused on energy efficiency and scalable precision. This scalability was achieved by identifying critical design parameters within the design space and optimizing them using sensitivity balancing and Pareto optimization techniques. Although not all design knobs can scale universally, this work provided insights into the impact of each parameter based on application requirements, facilitating a more targeted design strategy. Furthermore, we analyzed the effects of jitter as a noise source within the encoding process, demonstrating how precision could be scaled through specific design knobs.

Building upon this scalable foundation, we developed a novel asynchronous stimulus-driven workload execution technique that enables our architecture to execute arithmetic operations at a rate proportional to both the intensity and rate of change of events. In the context of this work, we applied this approach to an MNIST image classification task, where computational units implemented with ASC principles dynamically adjusted performance based on energy demands. Our evaluations demonstrated an energy efficiency range of 81.54 - 247.08 TOPS/W with a performance range of 11 - 248 FPS, illustrating the flexibility of our architecture to traverse an energy-efficiency versus performance curve and its potential impact on edge devices. Additionally, we incorporated a *latency vs. precision* trade-off, formalizing key design parameters such as average observation period and stream frequency, which allowed for precision scaling without additional circuitry. This approach, when applied to a Word2Vec algorithm for encoding word embeddings, achieved a 47.81% improvement in semantic accuracy as precision increased from 4 to 12 bits, providing a computational efficiency trade-off without the usual hardware overhead.

Leveraging the similarities between ASC streams and spiking models, we devised a novel $\Sigma\Delta\Sigma$ encoding scheme aimed at developing noise-resilient spiking neurons. This approach significantly enhanced the neurons' ability to encode and decode features amidst high noise, especially within an ensemble model framework. To validate the

robustness of this neural encoding scheme, we tested it against various noise profiles, including AWGN and pink noise. For practical application, we implemented a time-series prediction model based on a Liquid State Machine (LSM), that exhibited a 1.86x improvement in prediction accuracy for moderate noise levels. These models demonstrated enhanced noise resilience and reliable performance for smaller network sizes, even under heavily corrupted signal conditions—an advantage critical for real-world applications.

6.2 Impact

The work presented in this dissertation has been disseminated widely among academic and industry audiences through multiple publications at workshops and conferences and showcased at an annual research review meeting. The originality and impact of our research have also catalyzed a three-year industry-academia collaboration funded by the Semiconductor Research Corporation (SRC) under the SRC-TxACE program. Recognitions from the academic community underscore the contributions of this work: we received the Best Paper Award at ISQED'24 for our EAS-CiM research, and our SDS spiking neural work presented at MWSCAS'24 earned an invitation to submit an extended version to the IEEE Open Journal of Circuits and Systems.

6.2.1 EAS-CiM framework

Several research projects have begun to utilize our emulation framework and spiking neural encoding schema including a current research project which utilizes our framework to implement a scalable precision architecture based on skyrmionic-based racetrack memory and utilizes a temporal encoding method that bears similarity with ASC streams [56].

6.2.2 Industry Penetration

Our research has led to a three-year funded collaboration under the SRC-TxACE program, bringing us into partnership with industry researchers from Intel Research Labs, MediaTek, and IBM. Notably, there is strong industry interest in our EAS-CiM research, highlighting its potential for advancing edge computing architectures.

6.3 Future Architecture Research

This dissertation introduces a novel computational paradigm, that enabled the development of novel edge-based architectures capable of accelerating ML workloads, both CNN and SNN-based implementations. There are a few future directions this research could target and continue in. One direction could be to explore additional architectures and models that would benefit from the ASC computational paradigm.

6.3.1 FPGA inspired Distributed ASC computing

Our approach centers on implementing Compute-in-Memory (CiM)-based machine learning architectures to efficiently map Convolutional Neural Network (CNN) models, addressing memory bottlenecks by interfacing Asynchronous Stream Computing (ASC) streams with CiM tiles. This integration alleviates memory bandwidth constraints, significantly reducing data movement costs and enhancing energy efficiency. However, to fully unlock the potential of this architecture, advancements in reliable, high-yield multi-bit precision embedded non-volatile memory (eNVM) devices are essential. While substantial progress has been made in eNVM technology, challenges remain that limit the widespread adoption of these devices in mainstream industry applications.

An alternative approach involves the development of distributed computing cells, where computational logic and memory blocks are co-located within a single computational unit. Unlike CiM, where operations occur directly in the memory array, this approach uses memory as a conventional storage element but minimizes data

movement due to the physical proximity of memory and logic within each cell. This strategy enables the use of conventional memory technologies, such as floating-gate devices, which are more compatible with existing CMOS fabrication processes. Additionally, by organizing these computational cells into a 2D grid, this design supports a scalable and reconfigurable architecture reminiscent of FPGA architectures, offering the flexibility to adapt to various applications.

6.3.2 Benchmarking ΣΔΣ and EAS-CiM on Complex Models

In **Chapter 4** and **Chapter 5**, we demonstrated the advantages of event-driven computation and the noise robustness of the proposed ΣΔΣ (SDS) encoding, quantifying its performance on edge ML models. For future research, an exciting direction would be to accelerate models such as Echo State Networks (ESNs) and deeper CNNs tailored for edge machine learning tasks, including Keyword Spotting (KWS) and predictive maintenance for rotary machinery. This could be achieved by extending the stimulus-driven workload execution approach introduced in this work.

Another potential direction is to benchmark the SDS encoding within continuous audio-based digit recognition models using a Liquid State Machine (LSM) network. Prior reservoir models have achieved a Word Error Rate (WER) of 3% or lower with fewer than 30,000 parameters [33, 58]. However, these implementations often experience performance degradation of 10% or more when signal quality declines by 15 dB. It would be valuable to explore whether our SDS encoding technique could reduce this degradation under noisy conditions, enhancing the robustness of edge applications in challenging environments.

6.3.3 Exploring Additional Machine Learning Models

We believe that the ASC computational approach presented here has the potential to accelerate models beyond CNNs and SNNs. Future research could explore the application of these techniques to other architectures, including Long Short-Term Memory (LSTM) networks and Recurrent Neural Networks (RNNs), potentially expanding

the scope of ASC in enhancing energy efficiency and performance for a broader range of machine learning models.

Bibliography

[1] Text8 dataset. http://mattmahoney.net/dc/textdata.html. Accessed: [08, 01, 2024].

[2] Decadal Plan for Semiconductors - SRC — src.org. https://www.src.org/about/decadal-plan/, 2021. [Accessed 28-10-2024].

[3] R. Akhtar and F. A. Khanday. Stochastic computing: Systems, applications, challenges and solutions. In *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, pages 722–727, 2018.

[4] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, Brian Taba, Michael Beakes, Bernard Brezzo, Jente B. Kuang, Rajit Manohar, William P. Risk, Bryan Jackson, and Dharmendra S. Modha. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10):1537–1557, 2015.

[5] Maruan Al-Shedivat, Rawan Naous, Gert Cauwenberghs, and Khaled Nabil Salama. Memristors empower spiking neurons with stochasticity. *IEEE journal on Emerging and selected topics in circuits and systems*, 5(2):242–253, 2015.

[6] Maruan Al-Shedivat, Rawan Naous, Gert Cauwenberghs, and Khaled Nabil Salama. Memristors empower spiking neurons with stochasticity. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(2):242–253, 2015.

[7] Mark Anders, Himanshu Kaul, Sanu Mathew, Vikram Suresh, Sudhir Satpathy, Amit Agarwal, Steven Hsu, and Ram Krishnamurthy. 2.9tops/w reconfigurable dense/sparse matrix-multiply accelerator with unified int8/inti6/fp16 datapath in 14nm tri-gate cmos. In *2018 IEEE Symposium on VLSI Circuits*, pages 39–40, 2018.

[8] Mario Bambagini, Mauro Marinoni, et al. Energy-aware scheduling for real-time systems: A survey. *ACM Trans. Embed. Comput. Syst.*, 2016.

[9] Daniel Bankman, Lita Yang, et al. An always-on 3.8 μj 86% cifar-10 mixed-signal binary cnn processor with all memory on chip in 28-nm cmos. *IEEE Journal of Solid-State Circuits*, 2019.

[10] Daniel Bendor and Xiaoqin Wang. The neuronal representation of pitch in primate auditory cortex. *Nature*, 2005.

[11] Damien Bouchabou, Sao Mai Nguyen, Christophe Lohr, Benoit LeDuc, and Ioannis Kanellos. Using language model to bootstrap human activity recognition ambient sensors based in smart homes. *Electronics*, 10(20):2498, 2021.

[12] An Chen. A review of emerging non-volatile memory (nvm) technologies and applications. *Solid-State Electronics*, 125:25–38, 2016.

[13] Zhengyu Chen, Xi Chen, and Jie Gu. 15.3 a 65nm 3t dynamic analog ram-based computing-in-memory macro and cnn accelerator with retention enhancement, adaptive analog sparsity and 44tops/w system energy efficiency. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 240–242. IEEE, 2021.

[14] Yu-Der Chih, Po-Hao Lee, Hidehiro Fujiwara, Yi-Chun Shih, Chia-Fu Lee, Rawan Naous, Yu-Lin Chen, Chieh-Pu Lo, Cheng-Han Lu, Haruki Mori, et al. 16.4 an 89tops/w and 16.3 tops/mm 2 all-digital sram-based full-precision compute-in memory macro in 22nm for machine-learning edge applications. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 252–254. IEEE, 2021.

[15] Manon Dampfhoffer, Thomas Mesquida, Alexandre Valentian, and Lorena Anghel. Are snns really more energy-efficient than anns? an in-depth hardware-aware study. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(3):731–741, 2022.

[16] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

[17] Lei Deng, Yujie Wu, Xing Hu, Ling Liang, Yufei Ding, Guoqi Li, Guangshe Zhao, Peng Li, and Yuan Xie. Rethinking the performance comparison between snns and anns. *Neural Networks*, 121:294–307, 2020.

[18] Oussama Dhifallah and Yue M. Lu. On the inherent regularization effects of noise injection during training, 2021.

[19] R. P. Duarte and H. C. Neto. Stochastic processors on FPGAs to compute sensor data towards fault-tolerant iot systems. In *2018 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8, 2018.

[20] Karl Freund. Ibm research says analog ai will be 100x more efficient. yes, 100x.

[21] B. R. Gaines. Stochastic computing. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*. ACM, 1967.

[22] B. R. Gaines. *Stochastic Computing Systems*, pages 37–172. Springer US, Boston, MA, 1969.

[23] P. Gonzalez-Guerrero, X. Guo, and M. Stan. Sc-sd: Towards low power stochastic computing using sigma delta streams. In *2018 IEEE International Conference on Rebooting Computing (ICRC)*, 2018.

[24] P. Gonzalez-Guerrero, X. Guo, and M. R. Stan. ASC-FFT: Area-efficient low-latency FFT design based on asynchronous stochastic computing. In *2019 IEEE 10th Latin American Symposium on Circuits Systems (LASCAS)*, pages 117–120, 2019.

[25] Patricia Gonzalez-Guerrero, Xinfei Guo, and Mircea Stan. SC-SD: Towards low power stochastic computing using sigma delta streams. In *2018 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8, 2018.

[26] Joon-Kyu Han, Seong-Yun Yun, Sang-Won Lee, Ji-Man Yu, and Yang-Kyu Choi. A review of artificial spiking neuron devices for neural processing and sensing. *Advanced Functional Materials*, 32(33):2204102, 2022.

[27] Steven Harbour, Ben Sears, Stephen Schlager, Matt Kinnison, John Sublette, and Alex Henderson. Real-time vision-based control of swap-constrained flight system with intel loihi 2. In *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, pages 1–6, 2023.

[28] Orchi Hassan, Supriyo Datta, and Kerem Y Camsari. Quantitative evaluation of hardware binary stochastic neurons. *Physical Review Applied*, 15(6):064046, 2021.

[29] Yintao He, Ying Wang, Yongchen Wang, Huawei Li, and Xiaowei Li. An agile precision-tunable cnn accelerator based on reram. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–7, 2019.

[30] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14, 2014.

[31] Longwei Huang, Chao Fang, Qiong Li, Jun Lin, and Zhongfeng Wang. A precision-scalable risc-v dnn processor with on-device learning capability at the extreme edge. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 927–932, 2024.

[32] Je-Min Hung, Xueqing Li, Juejian Wu, and Meng-Fan Chang. Challenges and trends indeveloping nonvolatile memory-enabled computing chips for intelligent edge devices. *IEEE Transactions on Electron Devices*, 67(4):1444–1453, 2020.

[33] Azarakhsh Jalalvand, Fabian Triefenbach, David Verstraeten, and Jean-Pierre Martens. Connected digit recognition by means of reservoir computing. In *Interspeech*, 2011.

[34] Donghyuk Kim, Chengshuo Yu, Shanshan Xie, Yuzong Chen, Joo-Young Kim, Bongjin Kim, Jaydeep P Kulkarni, and Tony Tae-Hyoung Kim. An overview of processing-in-memory circuits for artificial intelligence and machine learning. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 12(2):338–353, 2022.

[35] Ilya Kiselev, Chang Gao, and Shih-Chii Liu. Spiking cochlea with system-level local automatic gain control. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(5):2156–2166, 2022.

[36] D. Kościelnik and M. Miśkowicz. Asynchronous sigma-delta analog-to digital converter based on the charge pump integrator. *Analog Integrated Circuits and Signal Processing*, 55:223–238, 2008.

[37] Duygu Kuzum, Rakesh G. Jeyasingh, et al. Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano Letters*, 2011.

[38] Shahar Kvatinsky, Misbah Ramadan, et al. Vteam: A general model for voltage-controlled memristors. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2015.

[39] Cecilia Latotzke and Tobias Gemmeke. Efficiency versus accuracy: a review of design techniques for dnn hardware accelerators. *IEEE Access*, 9:9785–9799, 2021.

[40] Y. LECUN. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*.

[41] Martin Lefebvre et al. 7.7 a 0.2-to-3.6tops/w programmable convolutional imager soc with in-sensor current-domain ternary-weighted mac operations for feature extraction and region-of-interest detection. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, 2021.

[42] Ziwei Li, Han Xu, et al. A 2.17w@120fps ultra-low-power dual-mode cmos image sensor with senputing architecture. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2022.

[43] Wootaek Lim, Inhee Lee, Dennis Sylvester, and David Blaauw. 8.2 batteryless sub-nw cortex-m0+ processor with dynamic leakage-suppression logic. In *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, pages 1–3, 2015.

[44] Qi Liu, Bin Gao, Peng Yao, Dong Wu, Junren Chen, Yachuan Pang, Wenqiang Zhang, Yan Liao, Cheng-Xin Xue, Wei-Hao Chen, et al. 33.2 a fully integrated analog reram based 78.4 tops/w compute-in-memory chip with fully parallel mac computing. In *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 500–502. IEEE, 2020.

[45] Wenjian Liu, Jun Lin, and Zhongfeng Wang. A precision-scalable energy-efficient convolutional neural network accelerator. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(10):3484–3497, 2020.

[46] Carrie MacGillivray, David Reinsel, and Michael Shirer. The Growth in Connected IoT Devices is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast — businesswire.com. https://www.businesswire.com/news/home/20190618005012/en/The-Growth-in-Connected-IoT-Devices-is-Expected-to-Generate-79.4ZB-of-Data-in-2025-According-to-a-New-IDC-Forecast, 2019. [Accessed 28-10-2024].

[47] R Channing Moore, Tyler Lee, and Frédéric E Theunissen. Noise-invariant neurons in the avian auditory cortex: hearing the song in noise. *PLoS computational biology*, 2013.

[48] Md Golam Morshed, Samiran Ganguly, and Avik W. Ghosh. Choose your tools carefully: a comparative evaluation of deterministic vs. stochastic and binary vs. analog neuron models for implementing emerging computing paradigms. *Frontiers in Nanotechnology*, 2023.

[49] Eustace Painkras, Luis A Plana, Jim Garside, Steve Temple, Francesco Galluppi, Cameron Patterson, David R Lester, Andrew D Brown, and Steve B Furber. Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE Journal of Solid-State Circuits*, 48(8):1943–1953, 2013.

[50] Saroj Kumar Panda, Man Lin, and Ti Zhou. Energy-efficient computation offloading with dvfs using deep reinforcement learning for time-critical iot applications in edge computing. *IEEE Internet of Things Journal*, 10(8):6611–6621, 2022.

[51] Chulsung Park and Pai H Chou. Ambimax: Autonomous energy harvesting platform for multi-supply wireless sensor nodes. In *2006 3rd annual IEEE communications society on sensor and ad hoc communications and networks*, volume 1, pages 168–177. IEEE, 2006.

[52] Minseong Park, Yuan Yuan, Yongmin Baek, Andrew H Jones, Nicholas Lin, Doeon Lee, Hee Sung Lee, Sihwan Kim, Joe C Campbell, and Kyusang Lee. Neuron-inspired time-of-flight sensing via spike-timing-dependent plasticity of artificial synapses. *Advanced Intelligent Systems*, 4(3):2100159, 2022.

[53] Antonio Pullini, Davide Rossi, Igor Loi, Alfio Di Mauro, and Luca Benini. Mr. wolf: A 1 GFLOP/s energy-proportional parallel ultra low power soc for IoT edge processing. In *ESSCIRC 2018 - IEEE 44th European Solid State Circuits Conference (ESSCIRC)*, pages 274–277, 2018.

[54] Arnab Raha, Deepak A Mathaikutty, Soumendu K Ghosh, and Shamik Kundu. Flexnn: A dataflow-aware flexible deep learning accelerator for energy-efficient edge devices. *arXiv preprint arXiv:2403.09026*, 2024.

[55] Davide Rossi, Francesco Conti, Manuel Eggiman, Stefan Mach, Alfio Di Mauro, Marco Guermandi, Giuseppe Tagliavini, Antonio Pullini, Igor Loi, Jie Chen, et al. 4.4 a 1.3 tops/w@ 32gops fully integrated 10-core soc for iot end-nodes with 1.7 μw cognitive wake-up from mram-based state-retentive sleep mode. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 60–62. IEEE, 2021.

[56] Mohammad Nazmus Sakib, Rahul Sreekumar, Xinyuan Zhu, Tommy Tracy, and Mircea R. Stan. Processing-in-memory with temporal encoding. In *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 56–61, 2022.

[57] H Sankar, V Subramaniyaswamy, Varadarajan Vijayakumar, Sangaiah Arun Kumar, R Logesh, and AJSP Umamakeswari. Intelligent sentiment analysis approach using edge computing-based deep learning technique. *Software: Practice and Experience*, 50(5):645–657, 2020.

[58] Benjamin Schrauwen, Jeroen Defour, David Verstraeten, and Jan Campenhout. The introduction of time-scales in reservoir computing, applied to isolated digits recognition. pages 471–479, 09 2007.

[59] Ali Shafiee, Anirban Nag, et al. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. 2016.

[60] Waseem Shariff, Joe Lemley, and Peter Corcoran. Event-stream super resolution using sigma-delta neural network. *arXiv preprint arXiv:2408.06968*, 2024.

[61] Vishal Sharma, Hyunjoon Kim, and Tony Tae-Hyoung Kim. A 64 kb reconfigurable full-precision digital reram-based compute-in-memory for artificial intelligence applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(8):3284–3296, 2022.

[62] Xia Sheng et al. Low-conductance and multilevel cmos-integrated nanoscale oxide memristors. *Advanced Electronic Materials*, 2019.

[63] Sumit Bam Shrestha, Jonathan Timcheck, Paxon Frady, Leobardo Campos-Macias, and Mike Davies. Efficient video and audio processing with loihi 2, 2023.

[64] Rahul Sreekumar and Mircea R. Stan. Microarchitecture optimization for asynchronous stochastic computing. In *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, 2021.

[65] Jian-Wei Su, Yen-Chi Chou, Ruhui Liu, Ta-Wei Liu, Pei-Jung Lu, Ping-Chun Wu, Yen-Lin Chung, Li-Yang Hung, Jin-Sheng Ren, Tianlong Pan, et al. 16.3

a 28nm 384kb 6t-sram computation-in-memory macro with 8b precision for ai edge chips. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 250–252. IEEE, 2021.

[66] J Taneja, J Jeong, D Culler, and Modeling Design. Capacity planning for micro-solar power sensor networks., in the proc. of the 7th international conference on information processing in sensor networks. In *IPSN SPOTS*, volume 8, 2008.

[67] Fengbin Tu, Yiqi Wang, Zihan Wu, Ling Liang, Yufei Ding, Bongjin Kim, Leibo Liu, Shaojun Wei, Yuan Xie, and Shouyi Yin. A 28nm 29.2 tflops/w bf16 and 36.5 tops/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)*, volume 65, pages 1–3. IEEE, 2022.

[68] Xiangwen Wang, Xianghong Lin, and Xiaochao Dang. Supervised learning in spiking neural networks: A review of algorithms and evaluations. *Neural Networks*, 125:258–280, 2020.

[69] Yiqun Wang, Yongpan Liu, Shuangchen Li, Daming Zhang, Bo Zhao, Mei-Fang Chiang, Yanxin Yan, Baiko Sai, and Huazhong Yang. A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops. In *2012 Proceedings of the ESSCIRC (ESSCIRC)*, pages 149–152. IEEE, 2012.

[70] Yuan-Kai Wang, Shao-En Wang, and Ping-Hsien Wu. Spike-event object detection for neuromorphic vision. *IEEE Access*, 11:5215–5230, 2023.

[71] Zhongrui Wang, Huaqiang Wu, Geoffrey W Burr, Cheol Seong Hwang, Kang L Wang, Qiangfei Xia, and J Joshua Yang. Resistive switching materials for information processing. *Nature Reviews Materials*, 5(3):173–195, 2020.

[72] M. Yang, J. P. Hayes, D. Fan, and W. Qian. Design of accurate stochastic number generators with noisy emerging devices for stochastic computing. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 638–644, 2017.

[73] Minhao Yang, Chen-Han Chien, Tobias Delbruck, and Shih-Chii Liu. A 0.5v 55w 64×2-channel binaural silicon cochlea for event-driven stereo-audio sensing. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 388–389, 2016.

[74] Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A dynamic vision sensor with 1% temporal contrast sensitivity and in-pixel asynchronous delta modulator for event encoding. *IEEE Journal of Solid-State Circuits*, 50(9):2149–2160, 2015.

[75] Changsheng Zhao, Ting Hua, Yilin Shen, Qian Lou, and Hongxia Jin. Automatic mixed-precision quantization search of bert. *arXiv preprint arXiv:2112.14938*, 2021.