UNIVERSITY OF CALIFORNIA

Los Angeles

Stochastic Compute-In-Memory Hardware Accelerator for Intelligent Edge Devices

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Electrical and Computer Engineering

by

Jiyue Yang

2023

ABSTRACT OF THE DISSERTATION

Stochastic Compute-In-Memory Hardware Accelerator for Intelligent Edge Devices

by

Jiyue Yang

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Los Angeles, 2023

Professor Sudhakar Pamarti, Chair

Deep learning is creating many new applications on edge devices such as autonomous driving, industrial robotics, and wearable health care. Edge devices demand hardware operating at low power, but processing with a high throughput. Traditional digital Von Neumann architecture such as CPU and GPU is limited by the data movement's cost. The computing challenge demands more efficient memory technology and architectures. Voltage-Controlled Magnetic Tunneling Junction (VC-MTJ) is an emerging MRAM solution that can achieve much higher density than SRAM and more efficient write operation than other MRAM technologies. VC-MTJs not only can be used as memory devices but also in cryptography and probabilistic computing applications such as Stochastic Computing (SC). VC-MTJ's special voltage-controlled switching behavior can achieve stable, but random, switching probability after a long pulse and removes the requirement of calibration circuit. We have demonstrated a VC-MTJ based TRNG in 65nm, which passed the NIST randomness tests. Compute-In-Memory (CIM) is an emerging solution to move computing inside the memory to avoid data access. Entire array can be activated for computing and, therefore, breaks the bandwidth limits of Von Neumann architecture. Stochastic Computing (SC) is an approximate computing method that uses extreme tiny bit-serial logic gates as computing unit to achieve massive parallelism on chip. Combining SC and CIM removes the costly Analog-to-Digital converter (ADC) of traditional CIM architecture and achieves high energy efficiency. The compact SC computation units can also achieve massive throughput density inside memory. In the second half of this thesis, we propose combining the benefits of SC and CIM as Stochastic Compute-In-Memory

(SCIM) accelerators. We have demonstrated two variants of the SCIM solutions: (1) An SCIM accelerator in 65nm supporting full CNN operations on chip with 8-bit precision for image classification applications. The memory stores pre-converted stochastic bit stream and achieves high energy efficiency by in-memory SC MAC units. (2) An SCIM accelerator in 12nm for high-speed object tracking application using event cameras. The memory embeds in-situ stochastic number generator to allow binary number storage and achieves >30x higher throughput density than state-of-the-art works.

The dissertation of Jiyue Yang is approved.

Subramanian Srikantes Iyer

Chih-Kong Ken Yang

Puneet Gupta

Sudhakar Pamarti, Committee Chair

University of California, Los Angeles

2023

To My Parents for Their Love and Support.

Table of Contents

List of Tables

List of Figures

Acknowledgements

First, I would like to thank my advisor Professor Sudhakar Pamarti. The completion of this thesis research would not be possible without his support and advice. I am very grateful of his active involvement in my research work and insightful guidance he provided. He has never been tired of helping students. I also had the privilege to witness his leadership in our lab and research projects that require close collaboration with other groups with different area of expertise, from which I have gained courage and wisdom to take on challenging tasks.

I would like to thank Professor Puneet Gupta, whom I have also worked very closely with in all my PhD projects. Professor Gupta has provided many insightful advice from his expertise in hardware system and architecture design. Professor Pamarti and Gupta lead the FRANC project. They provided a lot of feedback of my research during weekly meeting, design review and paper editing sessions. I am also grateful for Professor Chih-Kong Ken Yang and Professor Subramanian S. Iyer for being in my committee and their invaluable comments during exams and thesis review process.

I would also like to thank my lab mates and friends: Dr. Shi Bu, Vinod Kurian Jacob, Dr. Wojciech Romaszkan, Dr. Tianmu Li, Haris Suhail, Stephen Bauer, Mohammadreza Zeinali, Dr. Neha Sinha, Dr. Hani Esmaeelzadeh, Dr. Albert Lee, Dr. Di Wu, Haoran He, Bingqian Dai, Dr. Kejian Shi, Dr. Jiacheng Pan and Dr.Sida Li for countless discussion we had on academic topics and personal life. My PhD also wouldn't be possible without the support from my girlfriend Jocelyn Lu.

I am forever grateful to my parents for their love and support .

Curriculum vitae

2016	B.S., Electrical and Computer Engineering
	Cornell University, Ithaca, NY.
2018	M.S., Electrical and Computer Engineering
	University of California, Los Angeles, CA.
2021	Qualcomm Innovation Fellowship
	Qualcomm, San Diego, CA.
2022	Best Student Design Award
	2022 Asian Solid State Circuit Conference.
2023	Hardware R&D Intern
	Broadcom, Irvine, CA.

Chapter 1

Introduction

Deep learning is creating many new applications on edge devices such as autonomous vehicle, smart industrial robotics and wearable health care. Many of these applications require immediate response to the environment, but only have a limited energy budget. Edge computing is an emerging concept to process data on the edge devices to avoid slow and costly communicate to the cloud. Since sensor data is preserved locally, edge devices can provide enhanced privacy to users. Although edge computing can unlock many new features, it faces severe challenges from applications that demand both high throughput and low power at the same time.

The gap between the massive amount of data in deep learning models and the energy profile of conventional computing hardware poses serious challenges. The number of parameters and operations in deep learning models are several orders of magnitude higher than classical machine learning models, shown in Fig.1.1(right). Typical neural networks for image classification require 10-100M parameters and 1-20G operations [3][4][5]. The emerging neural network for large language models require the number of parameter in trillions [6][7]. However, accessing and communicating data consumes significantly higher energy compared to actual computation. The off-chip DRAM read consumes almost 100$\times$ higher energy than on-chip SRAM read, and 1000$\times$

higher compared to Multiplication and Accumulation (MAC) operations, shown in Fig.1.1(left). Efficiently using the data on chip to reduce the number of DRAM access can significantly improve the energy efficiency [8]. Researchers are also coming up with more efficient architectures and computing paradigms such as Compute-In-Memory (CIM) and Stochastic Computing (SC), which are the main focuses of this thesis.

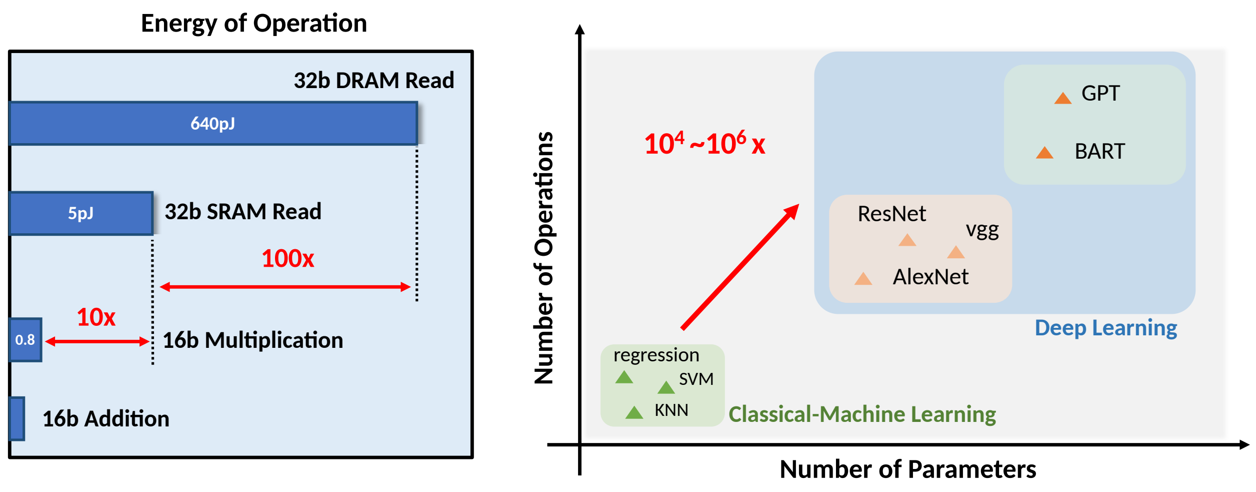


Figure 1.1: Energy consumption of different operation (left); Number of parameter and operations in machine learning and deep learning models (right).

1.1 Compute-In-Memory Accelerator

Von Neumann architecture has been widely adopted in CPU and GPU processors. Data is accessed from memory sequentially and transmitted to processor for computing. The array structure of the memory only allows one row to be accessed at time. The bandwidth limit of the memory and the long data path between memory and processor constrains the computing throughput and energy efficiency. Compute-In-Memory (CIM) accelerator is new architecture that moves computing next to the data. The comparison between Von-Neumann architecture and CIM is shown in Fig.1.2. The memory cells in the CIM array is custom design to embed computation logic. Usually, weight coefficients are stored in the memory; inputs are transmitted to the CIM array and applied to the computing word lines. Each CIM cell includes both the storage element and a multiplier unit.

Multiplication between the stored weight bit and the applied input bit is performed inside the CIM cell. The multiplication results are accumulated in the same column, equivalently achieving a dot-product operation. Analog-based CIM performs accumulation in analog domain such as current and charge, but suffers from compute errors[9][10][11]. Transistor's non-linearity, local mismatch and Process/Voltage/Temperature(PVT) variations cumulatively contribute to the degradation of the dynamic range. The analog signals also require bulky and power-hungry Analog-to-Digital Converters(ADC), which reduce the energy efficiency and area density. Digital-CIM uses digital adder tree inside the memory to perform accurate addition between accumulation results, but the adder tree requires large area overhead [12][13][14].

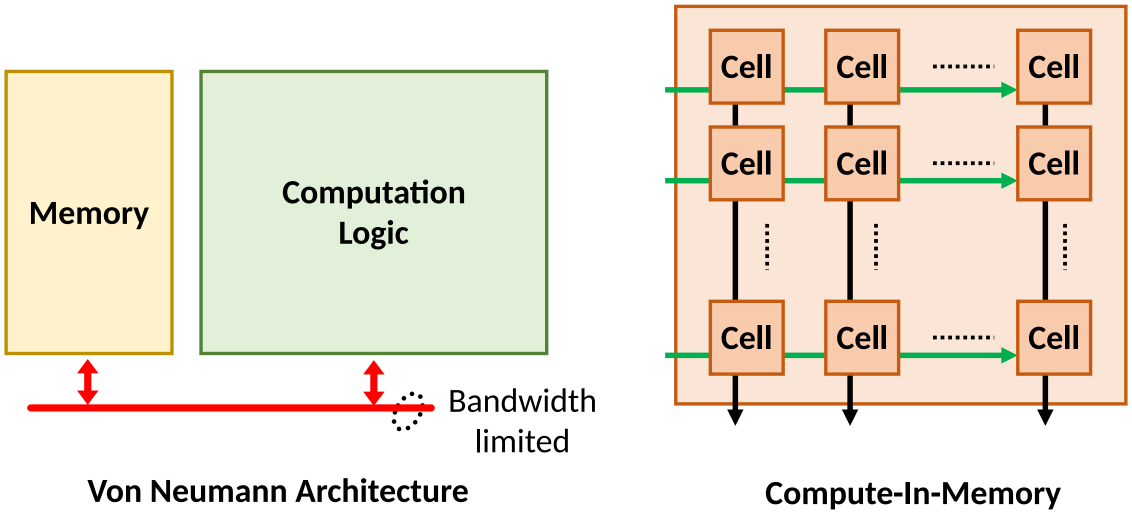
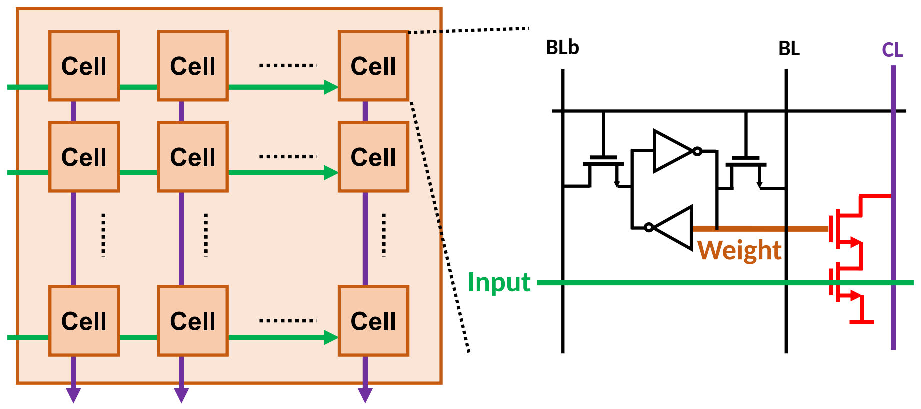


Figure 1.2: Von Neumann and Compute-In-Memory Architectures.

1.2 Analog Compute-In-Memory

An example of analog Compute-In-Memory array based on SRAM is shown in Fig.1.3. Each cell consists of a storage element and a multiplier which performs a multiplication operation between the stored weight bit and the input bit applied at word line. Two cascaded transistors form a simple 1-bit multiplier, which has a truth table similar to the AND logic gate. The multiplication result is ON/OFF state.



1b Multiplication Truth Table

input	weight	output
1	1	ON
1	0	OFF
0	1	OFF
0	0	OFF

Figure 1.3: Multiplication operation of transistor-based analog CIM.

The accumulation is performed on a shared bit line where multiplier's results are added as analog signal such as current and charge. Fig.1.4 shows three main categories of analog CIM differentiated by the types of analog signal: (I) Transistor's current, (II) Resistive device's current such as (RRAM/MRAM), (III) Charge.

1.2.1 Compute Accuracy

Accumulation operation of the analog CIM suffers from errors from local mismatch and PVT variations. A unified model is created to analyze the dynamic range of the computation at ADC's input as a function of mismatch properties for all three types of CIMs. Transistor-based CIM (Type I) accumulate transistor's current from multiplier cells, shown in Fig.1.4 (left). Transistor's nonlinearity causes the current to change when drain voltage is different. Furthermore, transistors in different cells draw different current due to local mismatches. In 12nm CMOS technology, saturation current of a minimal-sized transistor shows $3-5\%$ variation compared to its nominal current ($\Delta I/I$).

CIMs based on non-volatile devices (Type II) such as MRAM and RRAM can achieve much higher density using an 1-Transistor 1-Resistor (1T1R) cell as the storage element, shown in Fig.1.4 (mid). The stored bit is represented as the resistance state (HIGH/LOW) and some non-volatile devices can even store multi bits. The 1T1R cell can be directly used as an multiplier, or sometimes, two cells form a differential structure to keep a constant common-mode signal. Similar to transistor's current, nonvolatile devices suffer from mismatch of device's resistance, and, therefore current. Besides, non-volatile devices only have a limited ratio between resistance states, while transistors can be fully turned off to achieve a very large ON/OFF ratio. The small resistance ratio diminishes the dynamic range and leads to computation errors.

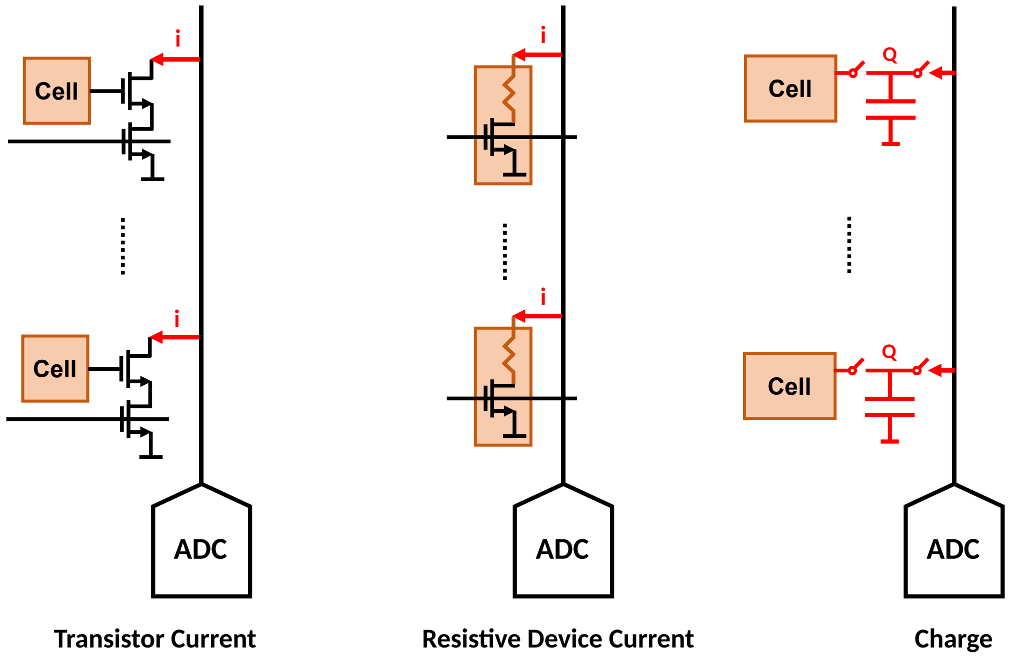


Figure 1.4: Analog CIM using transistors, resistive devices and charge.

Charged-based CIMs (Type III) uses capacitors to store multiplication result and accumulates charges on the shared bit line. It achieves significantly higher compute accuracy due to the good matching properties of capacitors, but its cell area is much larger compared to other two CIM types. Metal-Oxide-Metal (MOM) capacitors on CMOS' backend can achieve mismatch property ($Q/\Delta Q$) lower than 1% [15] and do not suffer from nonlinearity problems as transistors.

A unified model is created to compare the achievable dynamic range of these three types of CIMs, assuming the errors are dominated by local mismatches. PVT variations and nonlinearity will further degrade the dynamic range, so the results from this model can be used as an optimistic trend line. when N rows are turned on together, in the absence of device mismatches, the summed signal can be one of at most (N+1) possible levels. Typically, a column ADC is designed to reliably resolve these levels either in the current domain, or after converting into a proportional voltage, or time domain. Fig.1.6 (left) shows the quantization levels assuming current or charges are converted to voltage that has a $LSB = V_H - V_L$.

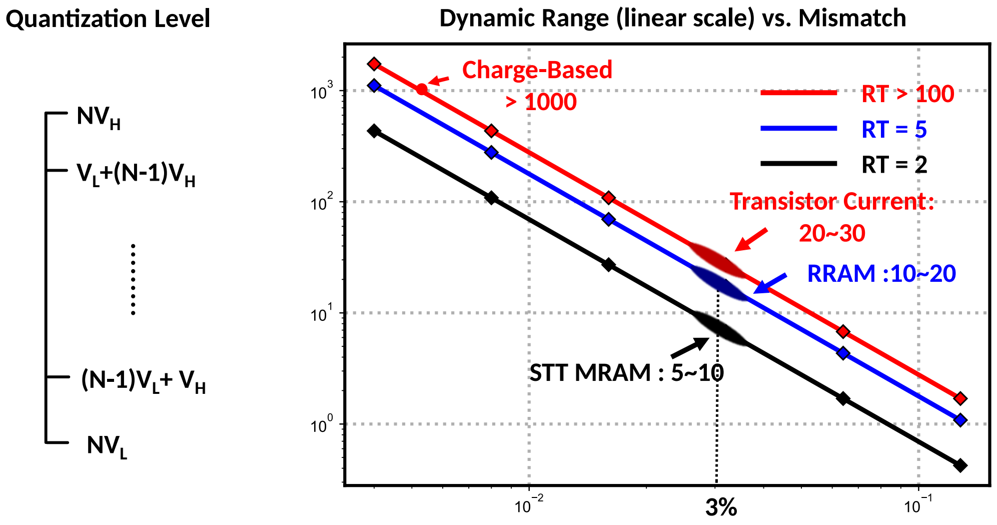


Figure 1.5: Dynamic range of the CIM dot product at ADC's input vs standard deviation of local mismatch.

Invariably, mismatches between the cells in different rows degrade the effective resolution and the total variation increases with the number of rows. Assuming that the mismatches are independent zero-mean Gaussian random variables with a normalized standard deviation of σ, the worst-case standard deviation of the MAC sum is $\sigma_{sum} = \sqrt{N}\sigma$. To reliably achieve no degradation of dynamic range, half the LSB should be greater than $3\sigma_{sum}$. It can be shown that:

$$N \leq [\frac{1}{6\sigma}(1 - \frac{1}{RT})]^2 \tag{1.1}$$

$$RT = I_H/I_L \text{ or } Q_H/Q_L$$

The dynamic range is inversely proportional to the mismatch and the ON/OFF ratio, RT, as shown in Fig.1.6(right) which plots N vs σ for different values of RT=2, 5 or >1000. Now, three types of analog CIMs can be compared. SRAM-based CIM that sums transistor's currents has >1000 cell ON/OFF ratio but the mismatch in the minimum sized FETs can easily be up to 3-5% limiting operation to only about 8-32 simultaneously enabled rows. For non-volatile deviced-based CIM, dynamic range of only $5-20$ can be achived. STT-MRAM with TMR ratio of 200% has been reported and corresponds to RT = 3 [16] . However, due to the larger resistance value of the access transistor compared to the MTJ resistance and MTJ resistance variations, the effective bit cell ON/OFF ratio is much lower. In fact, [17] claims that the tail bit in a large STT-MRAM array only has 20% TMR ratio. Since both the access transistor and the MTJ contribute to mismatch, a 3% σ-mismatch is optimistic and would limit the effective number of rows to 8. RRAM has a much higher ON/OFF ratio (5-10) compared to STT-MRAM but a 3% device mismatch would limit the number of rows to 20 during compute; state-of-the-art in RRAM based CIM has demonstrated 16 rows [18], [19]. In contrast, charge-based CIM using large MOM capacitors, which owing to their relatively large size, achieve much better matching and can achieve more than 1000 parallel rows computation without reduction in dynamic range [10].

1.2.2 Energy Efficiency

ADC Energy Consumption

The energy efficiency of the analog CIM is significantly limited by the power-hungry Analog-to-Digital Converters (ADC). Assume the ADC has an Effective Number of Bits (ENOB) of N_{ADC} and consumes an energy of E_{ADC} per input sample. For a CIM column with N_{row} rows accumulating in parallel, the ADC needs to quantize between N_{row} discrete levels without sacrificing signal's dynamic range: $2^{N_{ADC}} \geq N_{row}$. The energy consumption of the CIM column normalized by the number of inputs is:

$$\frac{E_{ADC}}{N_{row}} \geq \frac{E_{ADC}}{2^{N_{ADC}}} \tag{1.2}$$

This ADC performance metric is commonly reported and known as Walden Figure of Merit (FoM)[20]. The Walden FOM of the ADCs implemented in 65nm is plotted in Fig.1.6 as a function of sampling frequency. To match the throughput of CIM columns, ADCs typically operate at the frequency >100MHz and the Walden FoM of the state-of-the-art works is around 20fJ/conv-step, as highlighted in the red dotted box. Since one MAC is counted as two operations, the lowest ADC energy consumption normalized by the number of operations is 10fJ/Op. Based on these estimations, ADCs limit the CIM's overall energy efficiency to 100 TOP/S/W for 1b operation without considering any other energy consumption. For 8-bit operations and assume bit-serial/bit-parallel scheme described in [21], the energy efficiency is limited to 1.56 TOP/S/W.

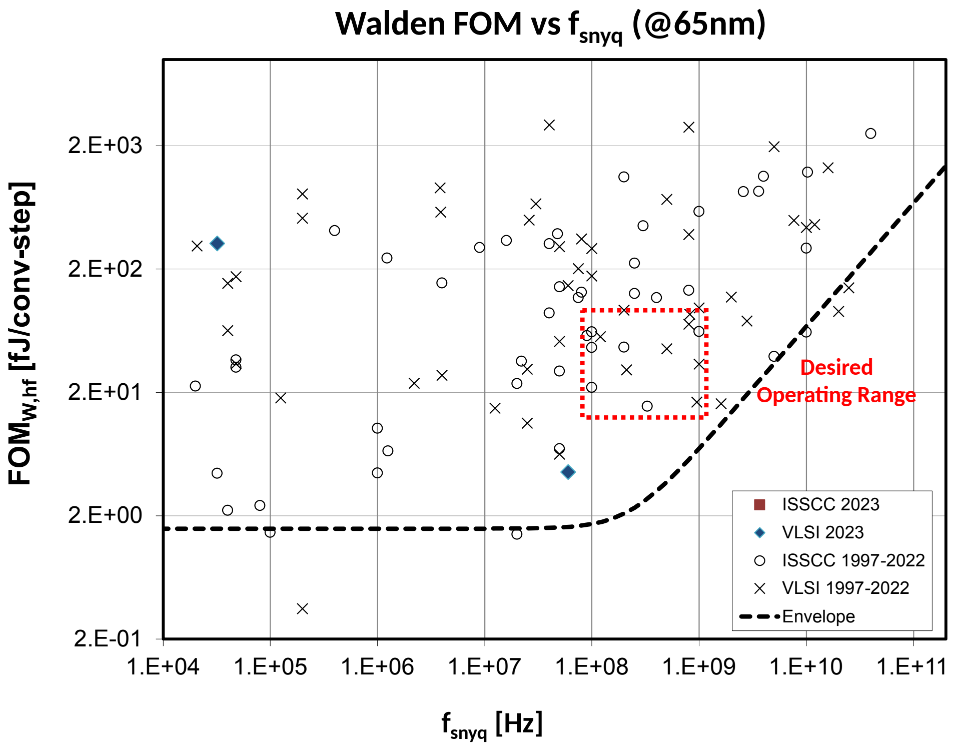


Figure 1.6: Walden ADC figure of merit (fJ/conversion step) vs Fnyqs implemented in 65nm and published in ISSCC and VLSI till 2023 [1].

Trade Off Between Energy Efficiency and Precision

Charge-based CIM achieves much higher compute accuracy by using a large capacitor to improve matching property, but its higher SNR comes with the sacrifice of energy efficiency. Since the charge is shared on the bit line during accumulation, the parasitic capacitance of the bit line reduces the signal's amplitude by a gain factor : $C_{signal}/(C_{signal}+C_{par})$, shown in Fig.1.7. In order to improve matching property and avoid signal loss, the bit cell capacitance is chosen to several times larger than the bit line's parasitic capacitance. Charging the bit cell capacitance leads to higher energy consumption compared to current-based CIM that only needs to charge bit line capacitance.

Energy efficiency of the CIM macro also trades off with the precision of ADCs. A tall column with large number of bit cells accumulating in parallel can reduce the average cost of the ADC, but sacrificing accuracy. N_{row} rows in parallel requires ADC to have a ENOB of $N_{ADC} \geq \log_2 N_{row}$ to capture the full precision, assuming the accumulation is accurate. Reducing the ADC precision for the applications that do not require high precision lowers the ADC energy per operation and also saves macro area. The dominant energy cost of a CIM macro and the trade off with precision is shown in Fig.1.7 (right). ADCs with full precision dominate the macro energy and require sacrificing $2-3$ bits of precision to bring down the cost. The charge-based compute uses bit cell capacitors to significantly increase the accuracy, but charging the capacitor becomes a big part of overall energy consumption.

1.3 Stochastic Computing

Stochastic Computing (SC) is a promising computing paradigm that becomes increasingly attractive in low-power deep learning applications on edge devices [22]. In SC, number is represented as a random binary bit stream and the fraction of 1s indicates its value. This unique representa-

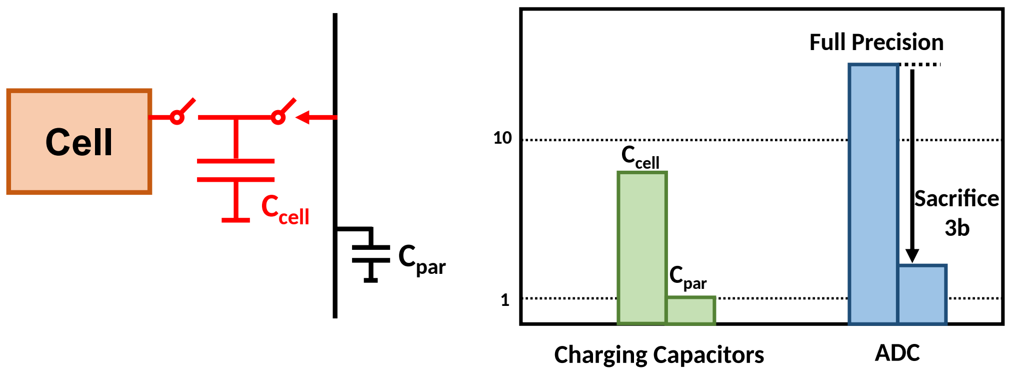


Figure 1.7: An example of charge-based CIM highlighting the size of bit cell capacitor compared to parasitic capacitance of the bit line (left); Dominant energy consumption of a CIM macro and trade off with precision.

tion enables extremely compact SC computing logic that is much smaller than the conventional digital logic. The SC multiplier is a single AND logic gate, and it is agnostic to the number precision. Longer bit stream leads to higher precision, therefore the same hardware can support programmable precision. To perform a multiplication equivalent to 8-bit binary precision, SC multiplier is 100× smaller in area, shown in Fig.1.8. The area density of SC provides an unique opportunity to increase parallelism and achieve a higher number of reuse of data. The area benefits, however, does not come for free. The SC logic gates need to compute 2^N cycles for N-bit precision. SC hardware is fully digital, which is deterministic and robust against errors suffered from analog compute. Due to these properties, embedding SC in memory is a very attractive solution that can remove ADCs or digital adder tree, which are required components in traditional analog or digital CIM solutions.

1.4 Voltage-Controlled MRAM and True Random Number Generator

Data movement cost poses a serious challenge of energy efficiency in conventional Von Neumann architectures, and the problem is worsened by the increased number of parameters in deep learning models. SRAM can achieve high read/write performance, but the SRAM cell occupies a large area

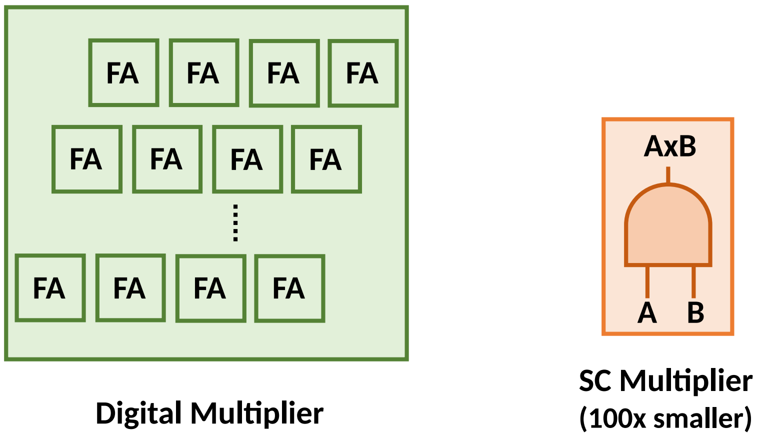
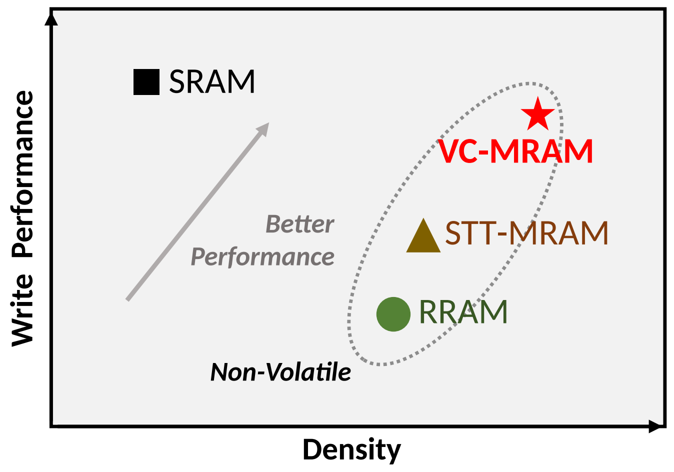
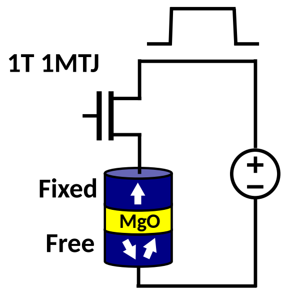


Figure 1.8: Digital and SC multiplier.

and transistors scaling in advanced technology nodes becomes more challenging. Non-Volatile memory such as Magnetic Random Access Memory (MRAM) has shown increasing popularity among mobile devices and micro-controllers as a replacement for last-level cache or Flash memory due to the dense 1-transistor bit cell. Spin-Transfer-Torque (STT) MRAM provides dense and non-volatile storage solution, but its slow and power-hungry write operation makes it less advantageous than SRAM, despite its nonvolatility [23]. Its high switching current also limits its density due to the sizing requirement of the access transistor. Voltage-Controlled (VC)-MRAM, also referred to as Magneto-Electric RAM (MeRAM), is a promising candidate to drastically improve the write performance and array density [24][25], Fig.1.9. The voltage-based writing mechanism and high resistance of VC-MTJ (larger than 10x of STT-MTJ) allows the access transistor to be minimal size. The Voltage-Controlled Magnetic anisotropy (VCMA) effect at the interface of free and barrier layer allows the voltage to modulate the perpendicular field. The free layer's magnetization will precess under the torque from in-plane field and switches to the opposite state in <1ns. The free layer becomes stable when voltage is removed.

VC-MRAM's special switching property can also be used to generate true random numbers, which are highly demanded in cryptography, statistical simulation and probability-based computing applications. When a voltage pulse is applied across VC-MTJ, the free layer starts precession and gradually converges to a metastable state due to damping effect. The switching probability





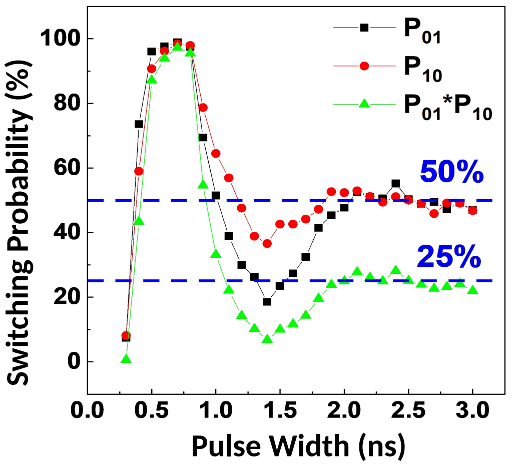


Figure 1.9: Digital and SC multiplier.

of the VC-MTJ reaches 50% after a 2ns pulse and remains static for longer pulse width, shown in Fig.1.9 (bottom right). The stable, but random, switching activities after a long voltage pulse removes the requirement for calibration circuits, which cause large area and energy overhead in many existing solutions.

1.5 Overview of This Thesis Research

The thesis consists of four chapters. Chapter 1 provides a brief introduction to the background and main approaches we take to solve the research problems in my PhD.

Chapter 2 presents a True Random Number Generator (TRNG) using Voltage-Controlled Magnetic Tunneling Junction (VC-MTJ). VC-MTJ is a new MRAM technology that uses Voltage-Controlled Magnetism (VCM) effect as the write mechanism. It has potential to improve the write performance by 10$\times$ compared to Spin-Transfer-Torque (STT) MRAM. Existing TRNG solutions harvest entropy from metastablity in cross-coupled inverters [26], jitters in ring oscillators [27][2] and random switching of STT MRAM [28][29]. Current solutions require calibration before random number generation and bias correction post-processing, which hinder the possibility to generate large number of random numbers in parallel and operate well under system variation or noise. VC-MTJ's free layer converges to the metastable state when a voltage pulse is applied across the MTJ. The longer the voltage pulse, the switching probability is closer to 50%, therefore removing the requirement for a calibration procedure. We have demonstrated a TRNG using VC-MTJs that passes the NIST tests. A light-weight digital bias correction algorithm is proposed to guarantee robust operation against malicious attack.

Chapter 3 presents a Stochastic Compute-In-Memory (SCIM) accelerator for neural network inference in image classification applications[30]. Embedding SC in memory is enabled by storing pre-converted stochastic numbers in SCIM macros and computing in a bit-parallel way. The SC accumulation uses OR logic, which does not require costly ADCs, and, therefore achieves very high density. The OR-accumulation suffers from nonlinearity if inputs are large, but efficient training has achieved comparable classification accuracy in CIFAR-10 and MNIST dataset as fixed-point implementation. We have also proposed a computation skipping technique that reduce the SC stream length by 4$\times$ when convolution layer is followed by a 2$\times$2 average pooling layer. We have built a complete Convolutional Neural Network (CNN) processor in 65nm using SCIM macros as the matrix-multiplication cores and supporting 8-bit full neural network operations on chip. The processor achieves peak energy efficiency of 7.96 TOP/S/W and the SCIM macro achieve energy efficiency of 20 TOP/S/W. A 14nm chip is also demonstrated with only SCIM macros and achieves energy efficiency of 35 TOP/S/W without average pooling function and 140 TOP/S/W with average pooling function.

Chapter 4 presents a Stochastic Compute-In-Memory (SCIM) accelerator for high-speed object tracking application using event camera. The SCIM accelerator storing the pre-converted stochastic numbers can achieve high energy efficiency, but it requires large area: N-bit number requires 2^N macros if computation skipping technique cannot be used. This work proposed embedding Stochastic Number Generator (SNG) in memory to enable storing binary numbers and SC stream is computed bit-serially. SCIM macros do not require ADCs. Increasing the number of multiplier cells sharing the same weight can increase throughput per area and wouldn't cause big energy or area overhead. The weight stochastic numbers from the SNG output are shared by 32 SC multiplier cells, which maximize a combined performance metric between energy efficiency and throughput per area. The SCIM macros are used to accelerate a filter-based object tracking algorithm for event camera and achieves 278M events/s. An early termination technique is proposed to skip computation with high-sparsity inputs, which improves both throughput and energy efficiency. A complete object tracking pipeline is implemented and shows very high tracking accuracy.

Chapter 2

True Random Number Generator Using Voltage-Controlled MTJ

True Random number generators (TRNG) are key components in cryptography applications. With the advent of the quantum computing, many of the traditional cryptography algorithms may be impaired or broken in a reasonable number of tries. To prevent security problems in the post-quantum cryptography, much longer keys are required [31]. This requires the TRNG hardware to have higher throughput and lower energy cost. Previous works have demonstrated hardware random number generators in CMOS technology using inverter's metastability [26] [32] and jitter in a ring oscillator [2] [27]. Most of them require dedicated circuit with large and power-consuming calibration and post processing circuits to remove system variations. The memory based TRNG reduces the cost of energy and area by reusing the memory array to generate random numbers. Previous works have explored TRNG based on STT-MRAM [28] [29], which exploits metastability in a current controlled spin-transfer-torque (STT) MRAM. However, the switching probability of the TRNG is highly sensitive to the amplitude and duration of the critical current. Given inevitable device variability, extensive calibration may be required to find qualified devices. Besides, the STT MRAM suffers from large energy consumption and limited endurance due to large write current.

To overcome those issues, we propose an in-memory TRNG using Voltage-Controller MRAM that does not require calibration of the write pulse. It improves energy consumption and endurance by having 50× larger resistance area (RA) product than STT-MRAM. Furthermore, a new bias-correction lightweight digital circuit is proposed to ensure high speed and robust randomness under potential magnetic field interference.

2.1 Overview of Existing Solutions

.

2.1.1 Metastability-based TRNG

The metastability of cross-coupled inverters can be harvested to generate random numbers and several metastability-based TRNGs have been successfully demonstrated in CMOS technology [26] [33] [34] [32] [35]. When cross-coupled inverters are enabled, usually by switching on the supply voltages, current from two inverters is racing to charge up their outputs. Thermal noise and local mismatch between inverters cause one inverter to charge its output faster than the other, and under the influence of the positive feedback, the inverter pair eventually settles to a stable state. Switching of the inverter can be extremely fast even under a racing condition, which provides a high throughput. The challenge of the metastability-based TRNG is to overcome the deterministic local mismatch between inverters. Complicated calibration circuit or post-processing circuit are required to remove the effect of mismatch [26] [33]. The TRNG demonstrated in [26] uses a control loop to program the delay of each inverter's start-up time, which requires a dedicated calibration processor to generate the configuration bits, shown in Fig.2.1. The system achieves throughput of 2.4Gbps and energy efficiency of 2.9pJ/bit. Another work [32] proposed a self-compensated solution to remove the inverter's mismatch that does not need a calibration loop. The large time constant of the mismatch compensation circuit leads to low throughput (10Kbps), but the circuit achieves a high energy efficiency of 0.186pJ/bit by using a 0.3V supply.

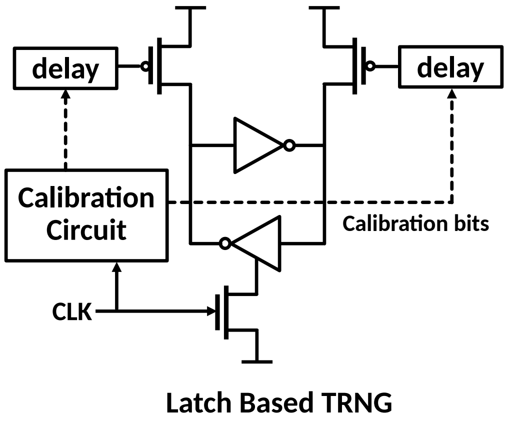


Figure 2.1: Concept of Metastability-based TRNG.

2.1.2 Ring Oscillator Based TRNG

Ring oscillator (RO) is an essential frequency generation block in many hardware systems. It uses compact digital logic gates, and, therefore has very compact layout. The phase noise performance is very poor compared to LC oscillator [36], which becomes an advantage, however, in generating true random numbers. The original RO-based TRNG uses an slow and uncorrelated clock to sample the fast clock generated from RO, shown in Fig.2.2. This solution leads to low throughput due to a long time required to accumulate jitter to achieve high entropy. For example, jitter that causes 0.01% frequency difference would require a slow clock that is 10000$\times$ slower to capture random phase changes. A TRNG based on this concept is demonstrated by IBM and used in the POWER7 processors, achieving a throughput of 2Mbps [37].

Other researchers have proposed solutions to increase the throughput of RO-based TRNGs and achieve robust operations under supply noise or PVT variations [27] [2] [38]. Researchers of [27] proposed to detect the beat frequency, Δf, of two free-running ring oscillators. The beat frequency is used to sample a counter that is running at the speed of the RO frequency. To use the same

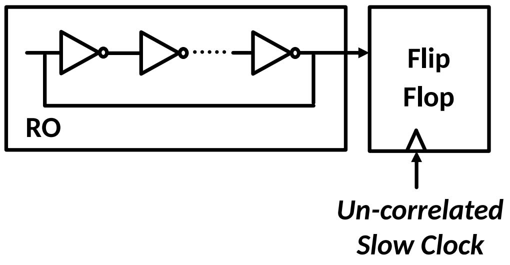


Figure 2.2: Concept of RO-based TRNG using a slow clock to sample fast RO.

example shown in the last paragraph: the jitter causes a 0.01% change of the main RO's frequency and the beat frequency between two ROs is 1% of the main RO's frequency. A phase change can be detected only 100× slower than the RO frequency, faster than the original solution. The ROs require a calibration loop to set the beat frequency in the desired range to achieve a high entropy and prevent overflow of counters. The raw random numbers generated from ROs are processed by a simple Von Neumann bias correction circuit and achieves a throughput of 2Mbps. The energy efficiency is 66pJ/bit [27].

Other researchers propose to start up the RO at 2× or 3× of the RO frequency by injecting edges at the intermediate stages and generate random numbers by digitizing the time it takes for the injected edges to collapse and return to stable 1× frequency [2] [38], shown in Fig.2.3. The TRNG with 2× start-up frequency achieves 2Mbps and 23pJ/bit. The TRNG with 3× start-up frequency achieves 23Mbps and 23pJ/bit. Still, the TRNG requires a calibration loop to ensure that the RO frequency is within the desired range to account for system variations. To improve the robustness of the RO-basd TRNG in a power supply attack, researchers of [39] proposed to use differential RO to achieve better immunity to supply noise. It achieves throughput of 9.9Mbps and energy efficiency of 42pJ/bit.

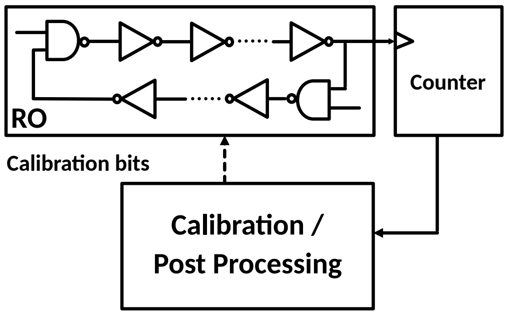


Figure 2.3: Diagram of the RO-based TRNG proposed by [2].

2.1.3 Memory-Based TRNG

Dedicated TRNG requires substantial area when a large number of true random numbers are demanded and researchers have proposed to reuse memory array to generate random numbers [28][40] [41] [29] [42]. Researchers of [42] uses the leakage current of an SRAM's column to generate random numbers. When the SRAM is idle, leakage current from the SRAM cells will discharge the bit line if the bit line buffer is disabled. The time for the bit line to reach a threshold is random due to the noise current. Since the leakage current is small, it takes long time to discharge the bit line. The TRNG achieves throughput of 3.6 Mbps and energy efficiency of 9.6pJ/bit [42]. [29] and [28] explore using Spin-Transfer-Torque (STT) MRAM to generate random numbers. STT MRAM shows random switching behavior when the write pulse is controlled at the preferred amplitude and duration. Researchers of [29] uses an off-chip calibration feedback loop to tune the write pulse in order to achieve random numbers with probability close to 50%. The calibration setup requires very fine control of the pulse width. [29] reported sub-50ps resolution required by the pulse width tuning circuit. Instead of randomly switch the STT MRAM, researchers of [28] uses the time to deterministically write the STT MRAM to generate random numbers. A comparator detects the moment when the MTJ state is changed and samples the counter clocked by a ring oscillator running very fast. The TRNG achieves throughput of 66Mbps and energy efficiency of 18pJ/bit. A calibration loop is require to set the write time in the desired range in order to prevent overflow of counters.

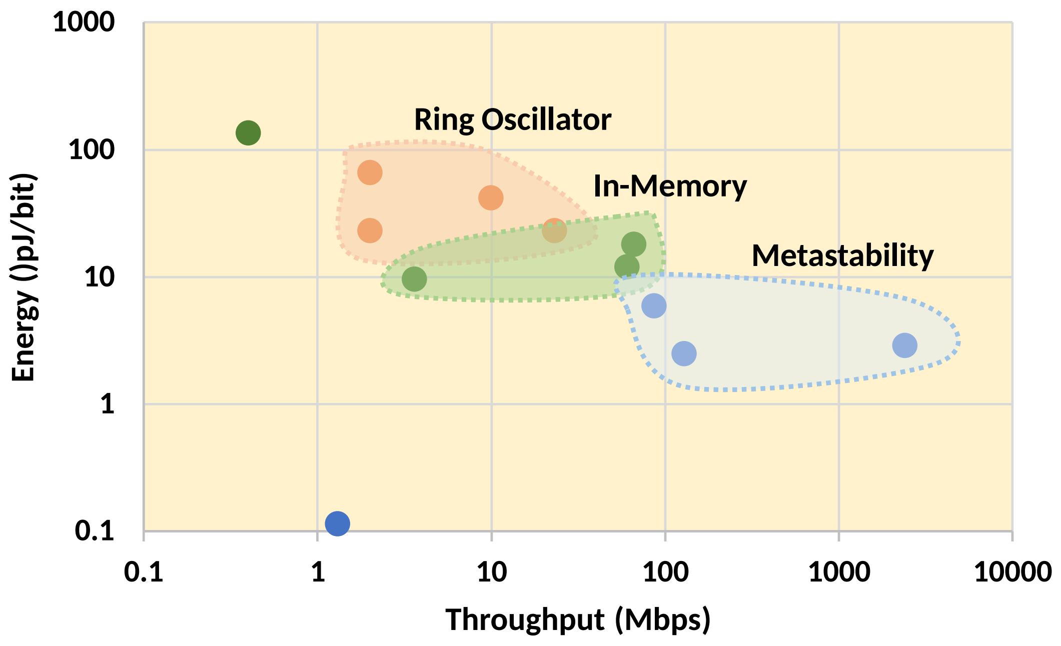


Figure 2.4: Energy and throughput of existing TRNG works.

2.1.4 Performance Overview of Existing Solutions

The performance of the existing TRNG solutions are compared and shown in Fig.2.4. The main performance metrics are energy consumption and throughput. The metastability-based TRNGs achieve both highest throughput (>100Mbps) and lowest energy consumption (<10pJ) because the cross-coupled inverters can switch at a very fast speed and draws very small dynamic power. This class of solutions, however, require the most complicated calibration process. When a large number of TRNGs are required, the calibration process might take a long time if all of them share the same calibration circuits. When the voltage or temperature conditions are changed, the calibration process is required again. The ring oscillator-based TRNGs have a low throughput (<10Mbps) and consumes significantly more energy (10−100pJ). The jitter in the RO requires a long time to accumulate to a detectable phase change, which is also the reason it consumes much more energy. The RO-based TRNG requires some calibration process in order to prevent overflow. The memory-based TRNG is a hybrid class since each type of device has a different mechanism to generate random numbers. The current memory-based TRNGs show performance between ring oscillator and metastability-based TRNGs. Energy consumption close to 10pJ/bit and throughput close to 100Mbps are demonstrated. Since the memory array is repurposed for the TRNG application, only a few circuit blocks are needed besides those used in normal memory operations.

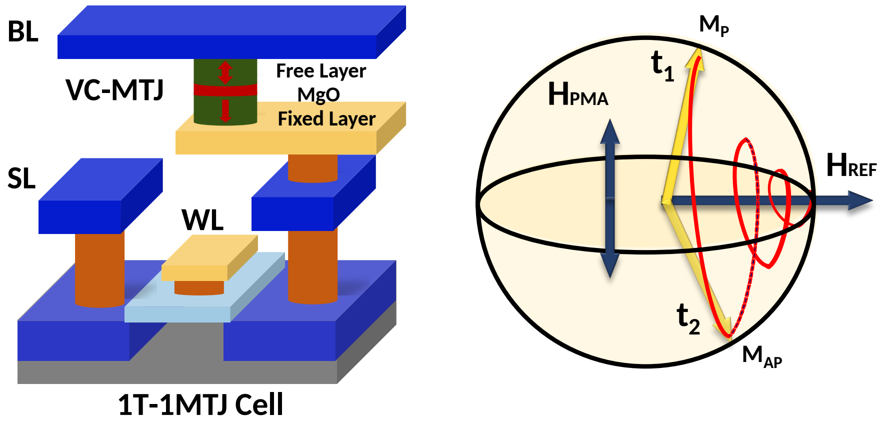


Figure 2.5: 1T-1MTJ cell; Voltage-controlled switching.

2.2 Voltage-Controlled Magnetic Tunneling Junction (MTJ)

2.2.1 Device Mechanism and Properties

Voltage-Controlled Magnetic Tunneling Junction (MTJ) uses Voltage-Controlled Magnetism (VCM) as the switching mechanism to achieve much faster speed compared to Spin Transfer torque (STT) [25][43]. The VC-MTJ has similar device structure as STT-MRAM but requires a thicker MgO layer to enhance the VCM effect and suppress STT effect, as shown in Fig.2.5. When a voltage is applied across the VC-MTJ, it instantly eliminates the Perpendicular Magnetic Anisotropy Field(HPMA). Due to the torque between free layer's magnetization and an in-plane reference field, the free layer will start a damped precessionas shown in Fig.2.5. If the voltage pulse is removed after half a precession period e.g., when the free layer magnetization reaches t_2 from

t_1, H_{PMA} will recover to the opposite direction and the free layer becomes stable. The precession is ultra-fast and previous works have shown 0.7ns switching speed [25].

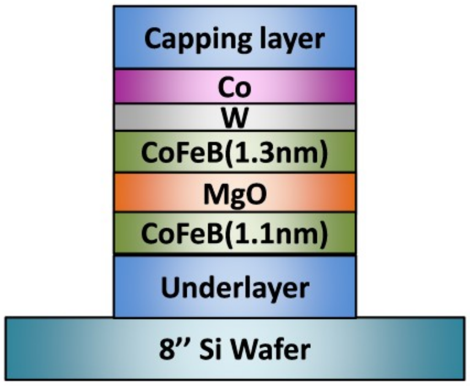


Figure 2.6: VC-MTJ film stack.

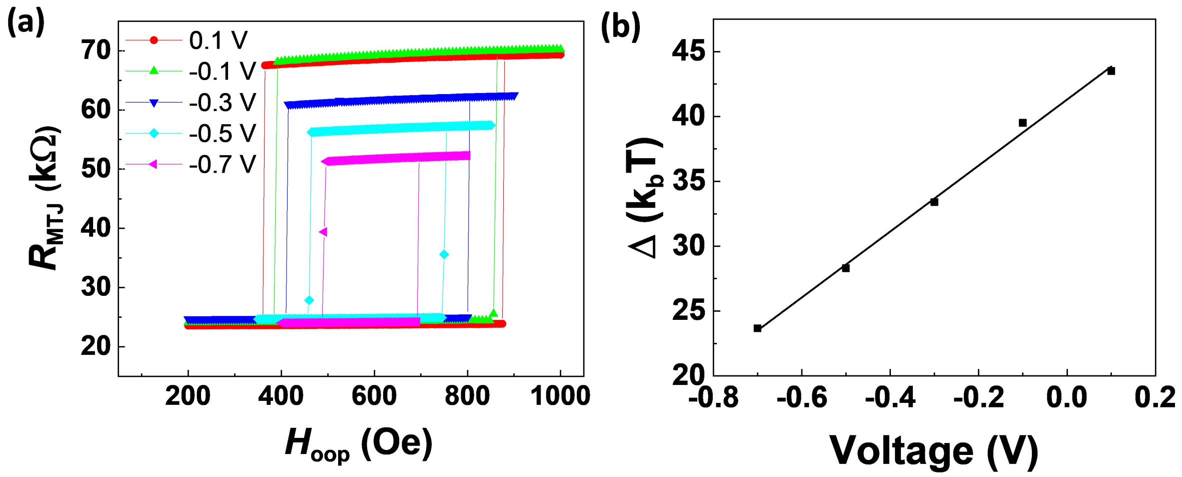


Figure 2.7: (a) R-H curves of free layer with various gate voltages. (b) Voltage dependence of thermal stability factor (Δ).

In this work, we have fabricated an array of VC-MTJs on an 8-inch silicon wafer. The film stack consists of a CoFeB free layer, CoFeB/W/Co fixed layer, and MgO tunneling barrier, shown in Fig.2.6. The MTJ pillars are patterned to the diameter of 80nm after 400°C annealing. Due to the thicker MgO layer, the RA of 200 $\Omega \times \mu m^2$ is achieved, which is about 50x larger than the STT-MRAM [44]. An external magnetic bias field is given to use as a processional axis and compensate for the fixed layer's stray field. Hysteresis loop for the free layer at various bias voltages are shown in Fig.2.7(a). Anti-parallel state's resistance of 65K ohm and TMR of 160% are achieved. The voltage-controller magnetic anisotropy (VCMA) can be extrapolated from the thermal stability vs voltage curve [25], shown in Fig.2.7(b). The VCMA coefficient of 41.5fJ/Vm is measured and corresponds to a write voltage of 1.4V.

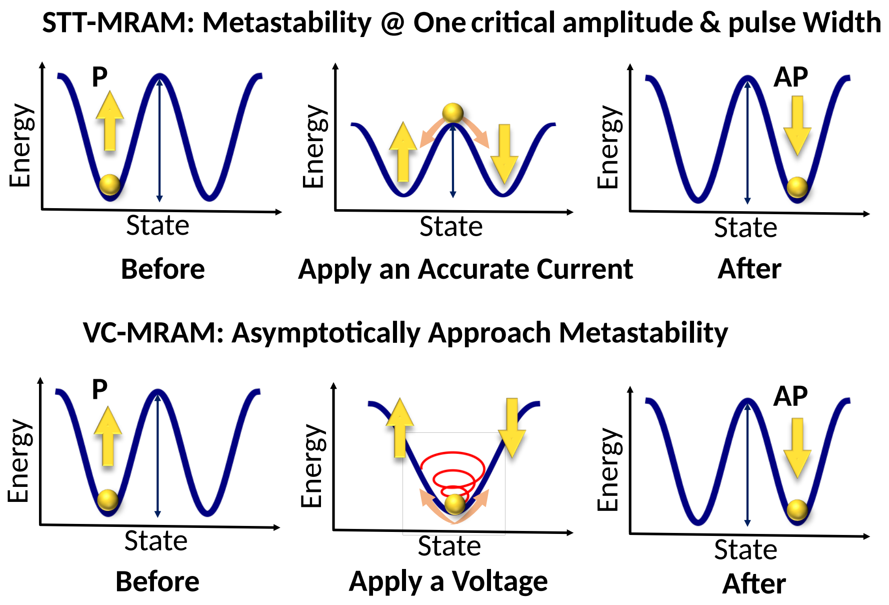
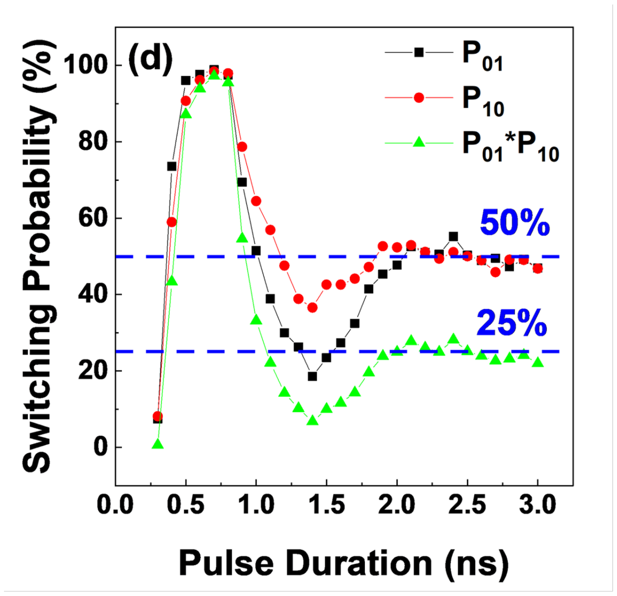

STT-MRAM: Metastability @ One critical amplitude & pulse Width
P
Energy
State
Before
Energy
State
Apply an Accurate Current
AP
Energy
State
After
VC-MRAM: Asymptotically Approach Metastability
P
Energy
State
Before
Energy
State
Apply a Voltage
AP
Energy
State
After

Figure 2.8: Comparison of stochastic mechanism between STT-MRAM and VC-MRAM.

2.2.2 Using VC-MTJ as Random Number Generators

The VC-MTJ has the unique property of converging to metastability asymptotically without the requirement of any calibration, making it a perfect solution of generating true random numbers inside the memory. A comparison of the random number generation mechanism between STT-MRAM and VC-MRAM is shown in Fig.2.8. STT-MRAM rely on a critical current that has a pre-determined amplitude and pulse width to achieve metastability. High resolution timing control of the write pulse for each device during the operation or calibration ahead of the operation is needed to achieve high entropy [29]. In contrary, VC-MRAM does not require calibration before or during the operation. When a voltage is applied, the free layer's magnetization precesses along the in-plane axis. It oscillates between P and AP state and asymptotically converges to the in-plane axis due to damping effect. The longer the voltage pulse is, the closer it is aligned to the in-plane direction, which corresponds to 50% probability. After the voltage pulse is removed, the free layer's magnetization is randomly switched to P or AP state under the influence of the thermal noise. The measured switching probability approaches 50% after the voltage pulse is applied 3n second, as shown in Fig.2.9.



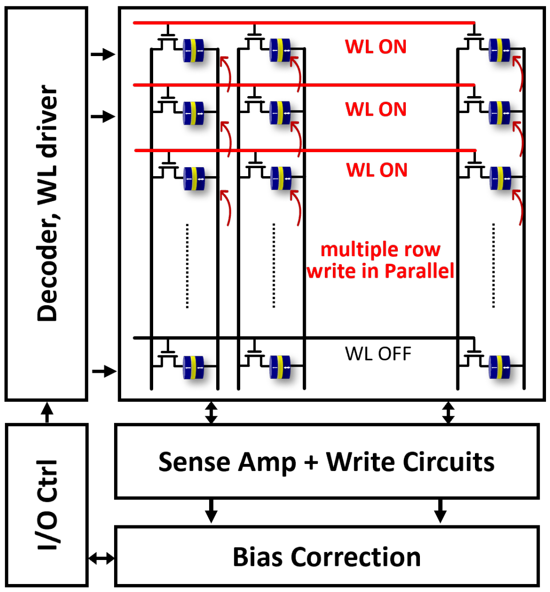


Figure 2.9: Switching probability vs pulse duration; VC-MTJ based TRNG architecture.

VC-MRAM makes the in-memory true random number generation of multiple rows possible when high throughput is required. The array arrangement of the memory makes only one row of the bitcells available for read or write at a time. Since STT-MRAM requires a calibration for each individual device inside the memory, the multi-row in-memory operation is not possible to achieve. However, since VC-MTJ does not need calibration, devices in multiple rows can generate random numbers at the same time. In a multi-row operation, several wordlines turn on together, as shown in Fig.2.9(right). The bitcells on the same bitline share the same write pulse. A longer pulse can make sure that most of the bitcells generate high-entropy random bits even if device variations are considered. A post processing circuit can remove any potential bias during read out. The multi-row in-memory RNG can significantly increase the throughput with no extra hardware cost.

2.3 VC-MTJ Write and Read Operation

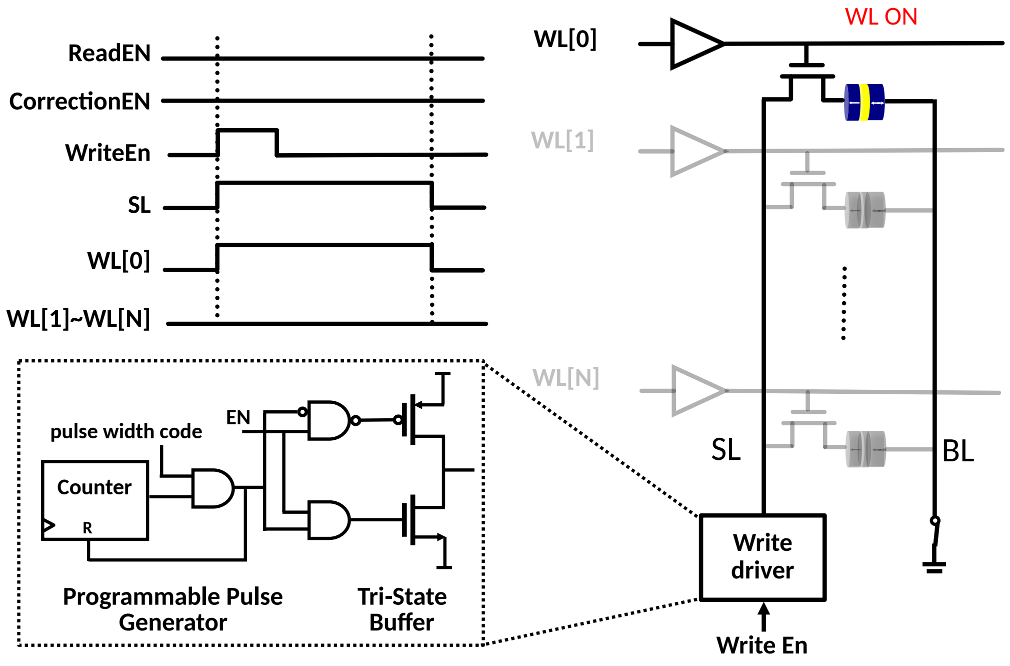


Figure 2.10: VC-MTJ write operation.

Raw random numbers are first generated by applying a long voltage pulse across the VC-MTJs. The Voltage-Controlled Magnetism (VCM) based write is uni-directional: the voltage in one polarity reduces the Perpendicular Magnetic Anisotropy (PMA), which is condition to switch the MTJ; the voltage in the opposite polarity can enhance the PMA and makes the magnetic moment more stable. In an VC-MTJ array the voltage pulse is applied to the source line (SL) during write operation, while current is sensed at the bit line (BL) during read opeartion, shown in Fig.2.10. The write pulse for the random number generation requires much longer duration than the deterministic write, and, therefore, can be efficiently generated using digital circuits. A counter generates a voltage pulse with tunable pulse width, which is synchronized with the system's clock. A tri-state buffer applies the voltage pulse on SL. It can be optionally turned off when the memory is idle to minimize the leakage current. The memory controller can turn on a single device or multiple devices togther by controlling the word line drivers to support high-throughput options.

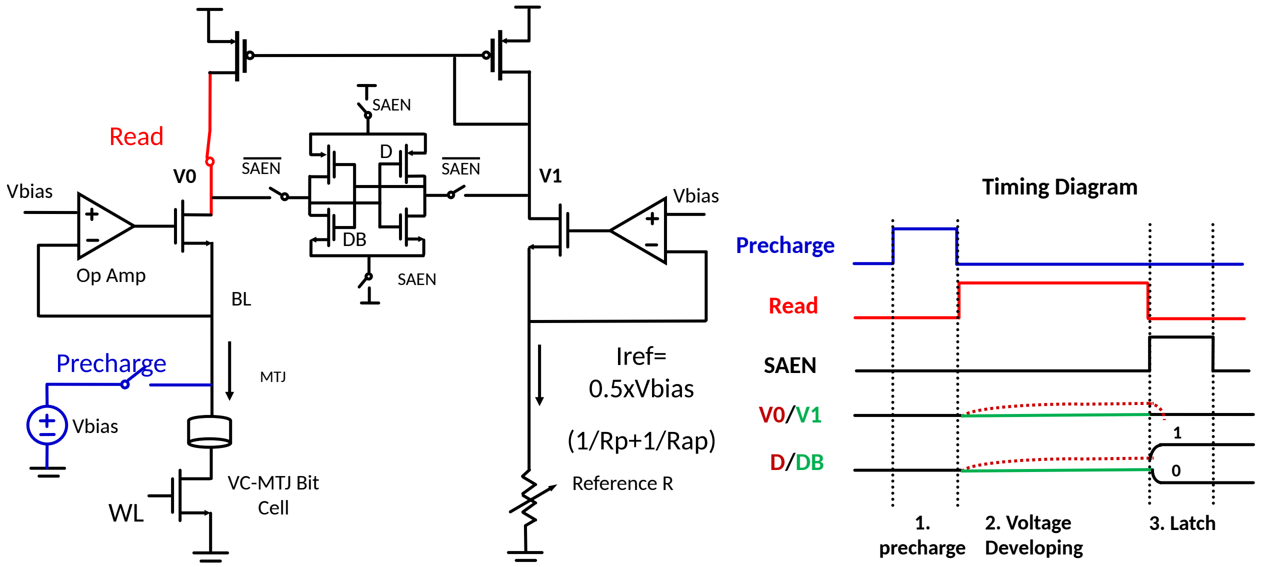


Figure 2.11: VC-MTJ write operation.

The conventional voltage-based sense amplifier compares the voltages between the MTJ's bitline and a reference bitline. However, the speed of sensing is limited by the large RC constant of the biltine parasitic capacitance and the MTJ resistance. The current sense amplifier can achieve faster speed by a common-gate stage which has a lower parallel resistance between MTJ and common gate stage's input impedance, therefore achieving lower time constant. The read operation's scheme and timing diagram are shown in Fig.2.11. A feedback amplifier regulates the bitline voltage and reduces the SA's input impedance by the loop gain. The difference between the MTJ and reference's current is amplified at the read node V0 that has much smaller capacitance. During the precharge phase, the bitline is pre-charged to a bias voltage. Then during the voltage-developing phase, the bitline voltage is kept constant at Vbias by the regulator. When the read switch is turned on, the current from the selected MTJ cell is passed to the read node V0. The pFET connected to V0 copies the current from a reference branch that sets the current at the middle of the MTJ's P and AP state. The currents between the MTJ and reference cell are subtracted and amplified at the end of voltage developing phase. In the latch phase, a voltage comparator compares V0 and V1 and latches the result to VDD or GND. A fast 5nsec sensing speed is achieved in the presence of large (pF) bitline capacitance due to the small time constant at V0.

2.4 Digital Bias Correction Circuits Using Stochastic Number Generator

In order to guarantee robust operation under system variation and potential malicious attacks, we propose a light-weight digital bias correction circuit to process the raw random numbers generated from VC-MTJs. Although the ideal switching probability after the metastable state of the VC-MTJ is 50%, multiple sources can cause a probability bias. For example, the stray field from the fixed layer may not be completely compensated by the external field. Malicious attackers may also apply an interference magnetic field externally to disturb the TRNG. The residue field can cause the free layer to bias towards a state, and thus causing probability errors in the random number output.

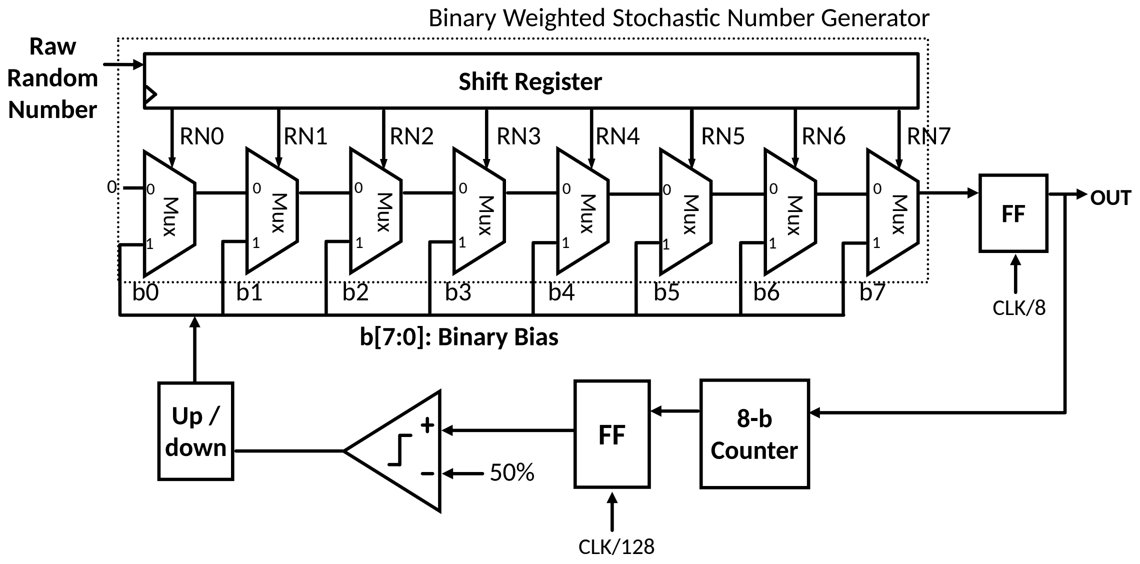


Figure 2.12: Bias correction circuit using binary-weighted stochastic number generator.

The proposed bias correction circuit is shown in Fig.2.12. The core of the bias-correction circuit is a Binary Weighted Stochastic Number Generator (BW-SNG) [45]. It is originally used to convert binary to stochastic numbers for applications of Stochastic Computing (SC) [46]. The BW-SNG can almost accurately convert a multi-bit binary number into a random bit stream such that the fraction of one is equal to the binary input. At every cycle, a raw random number (RN) is shifted into the buffer and converts a binary number (b) to random bit stream at OUT. Every 8-bit raw random numbers (RN_{0-7}) are used as the control signals for a chain of 8 multiplexers. The 0-selected input signal is the output from the previous multiplexer and the 1-selected input signal is each bit of the binary number b_{0-7}. Assuming the raw random numbers are Independent and Identically Distributed (I.I.D.), the output of the BW-SNG is a stochastic bit stream with probability represented as Equation.2.1.

$$P_{SNGOUT}(b, P_{RN}) = \sum_{n=1}^{8} b_{8-n} \times P_{RN}(1 - P_{RN})^{n-1} \tag{2.1}$$

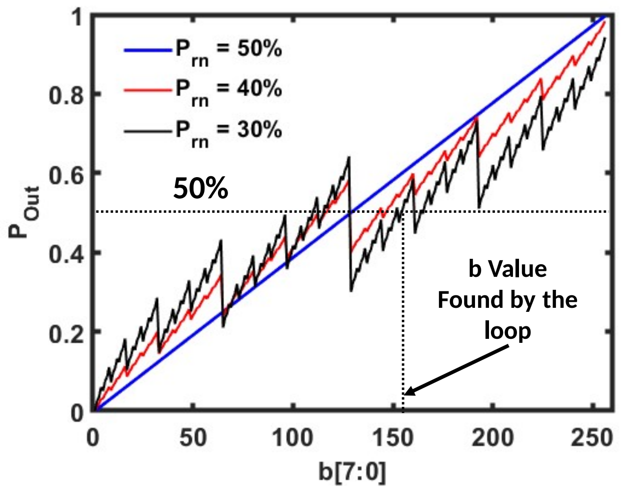
P_{RN}: probability of the raw random numbers, assuming I.I.D. conditions.

The probability tracking circuit in the feedback loop counts the number of 1s in 256 output bits serially and compare the probability with 50%. Binary number b is added or subtracted by 1 based on the comparison result until the output probability is close to 50%. The SNG OUT's probability (P_{SNGOUT}) is a function of b and P_{RN}. Assume certain P_{RN}, Fig.2.13 (top) plots P_{SNGOUT} vs. b for P_{RN} = 0.3, 0.4 and 0.5. The binary number b is initialized at 0.5 (1000000), then the feedback loop will find the closest solution for the optimization problem defined in Equation2.2:

$$\arg\min_{b}\{Max(P_{SNGOUT}(b, P_{RN}) - 0.5, 0)\} \tag{2.2}$$

The discontinuity of the output probability in the Fig.2.2(top) shows that even if the feedback loop converges to a local minimal, the corrected output might still have small probability errors.

Fig.2.2(bottom) plots the probability error ($P_{SNGOUT} - 0.5$) vs. P_{RN}. The error probabilities are mostly below 0.3% across all bias range. Better bias correction results can be achieved by using BW-SNG with higher precisions. Fig.2.2(bottom) also plots the error correction results using a 10-bit and 12-bit BW-SNG, which can achieve errors well below 0.1%.



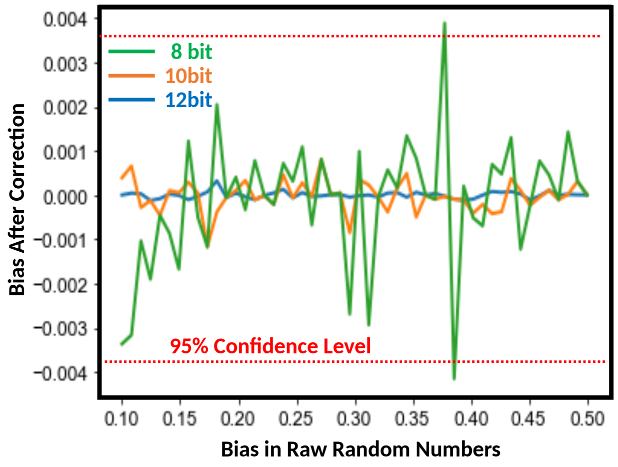


Figure 2.13: (Top) P_{SNGOUT} vs. b for P_{RN} =0.3, 0.4, 0.5; (Bottom) Probability error compared to 50% after correction vs P_{RN}.

2.5 Measurement and Evaluation

The VC-MRAM based TRNG is demonstrated by having the MRAM array circuitry fabricated in 65nm CMOS technology and VC-MTJ devices fabricated on another die. The core circuit occupies

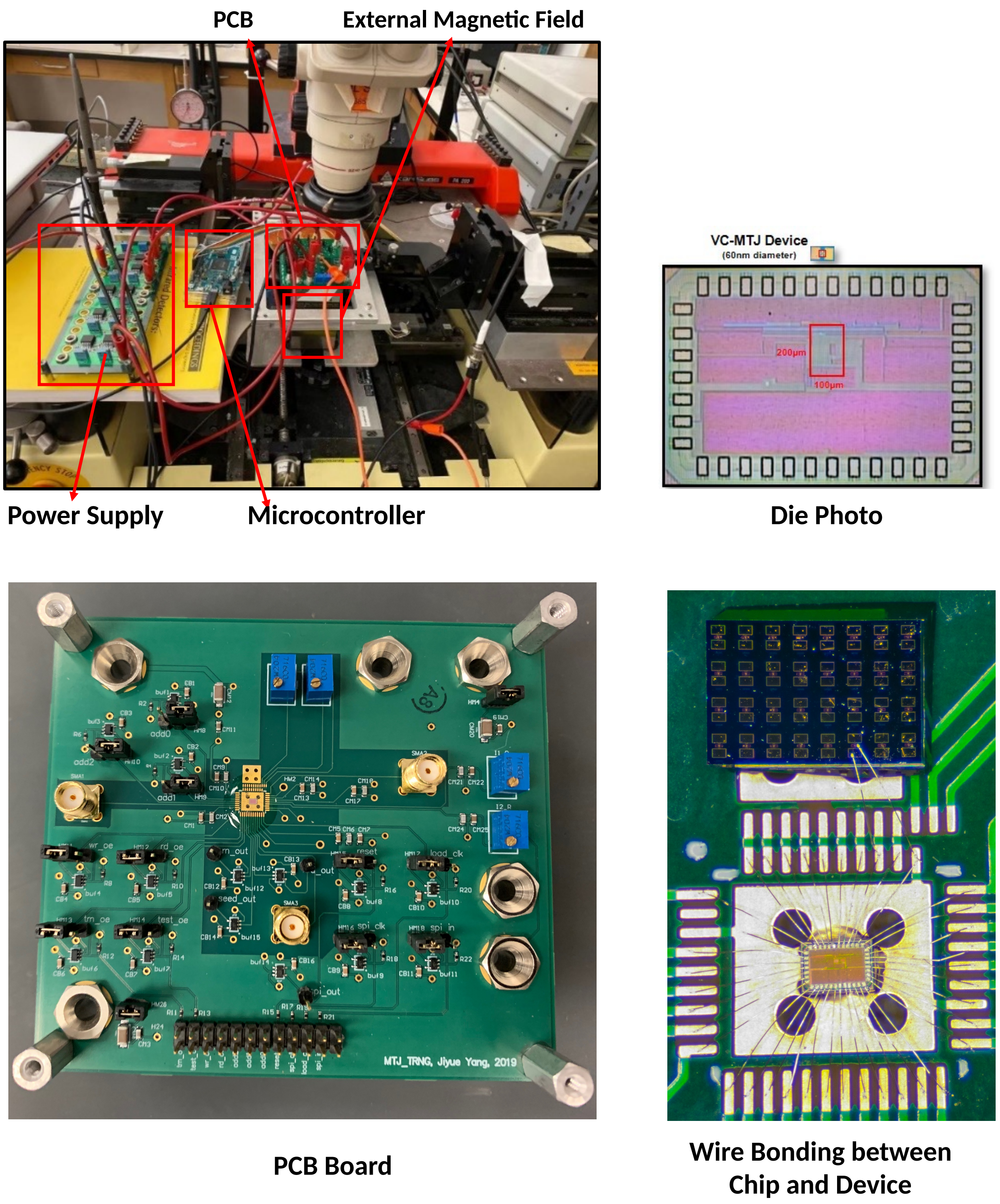
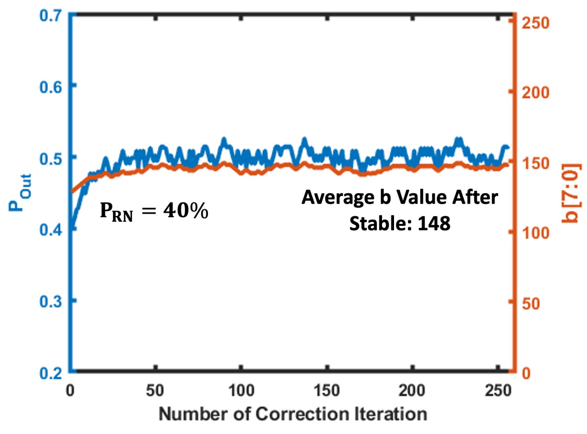


Figure 2.14: Photograph of TRNG chip and testing setup.



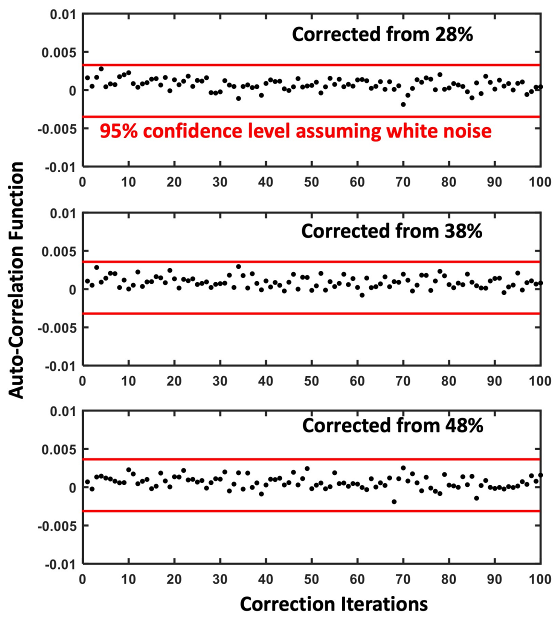
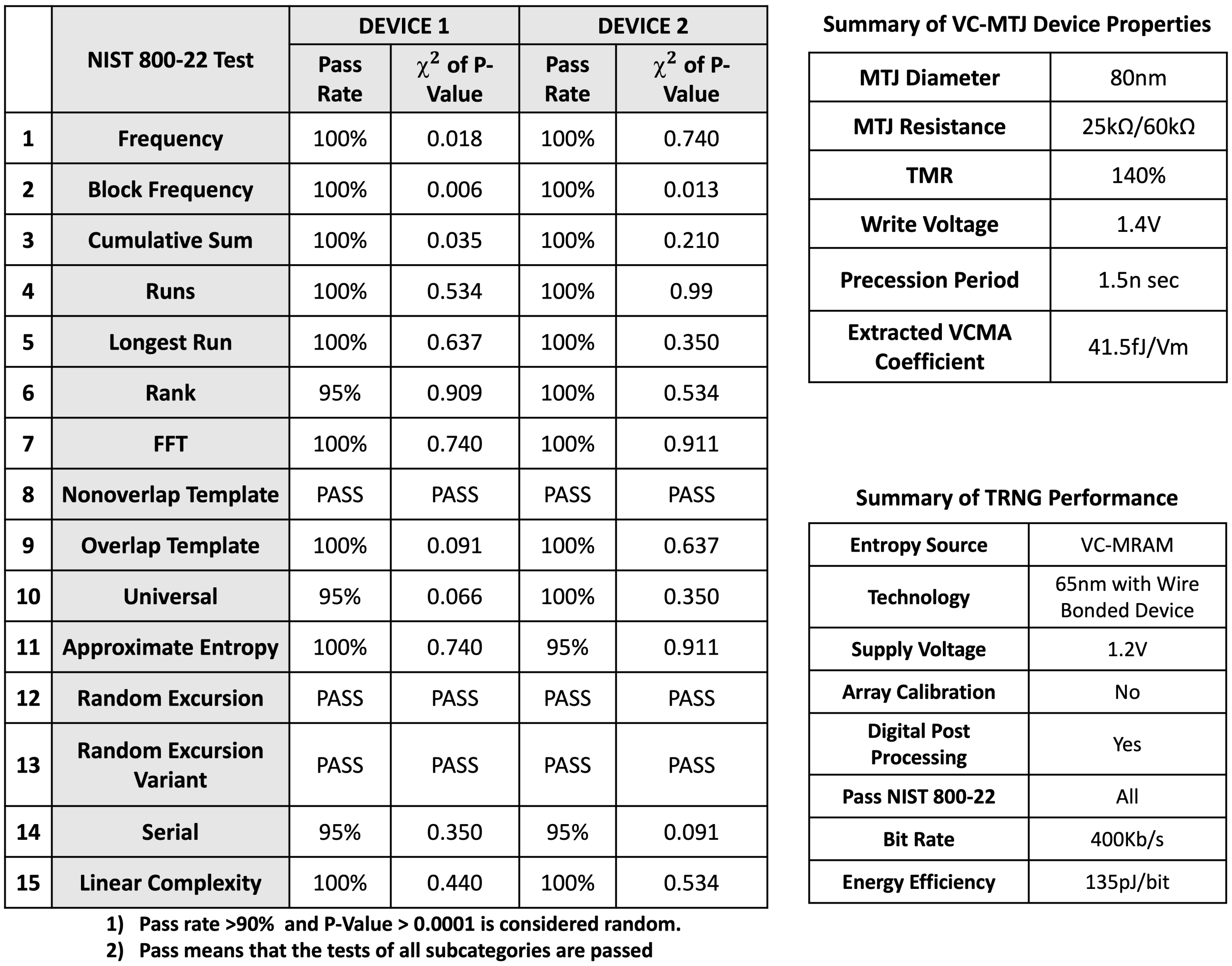


Figure 2.15: Correction output probability and b vs correction iteration (left); Autocorrelation function of streams corrected from multiple biases (right).

an area of only $20\mu m \times 70\mu m$. Eight VC-MTJ devices are connected in a column and wire-bonded with the CMOS chip, shown in Fig.2.14. Multiple VC-MTJs have been tested with the CMOS chip. The device properties of the VC-MTJ is summarized in Fig.2.16(right). It has a TMR of 150% and VCMA coefficient of 41.5fJ/Vm.

For each VC-MTJ, we collected multiple 2M bit random number streams for the NIST 800-22 randomness test. All VC-MTJs pass the NIST test. Fig.2.16(left) shows a summary of the NIST test results for two example devices. To test the bias correction circuit, we deliberately create a probability bias in the raw random numbers using a external magnetic field. Fig.2.15(left) shows the probability of the corrected random numbers and the binary weighted number b vs correction iterations during the probability tracking process. The raw random numbers are biased at 40% at the beginning. As the binary weighted number starts to increase, the output probability increases correspondingly and reaches a stable state of 50% probability within the first 20 cycles. Several different bias fields are also tested to show that probability tracking circuit can work in a wide range of raw random number bias probabilities. Fig.2.15(right) shows the autocorrelation function of the three corrected random bit streams biased at different starting probabilities. They are bounded within the threshold assuming that the bit streams are white noise with 95% confidence level.



	NIST 800-22 Test	DEVICE 1		DEVICE 2	
		Pass Rate	χ^2 of P-Value	Pass Rate	χ^2 of P-Value
1	Frequency	100%	0.018	100%	0.740
2	Block Frequency	100%	0.006	100%	0.013
3	Cumulative Sum	100%	0.035	100%	0.210
4	Runs	100%	0.534	100%	0.99
5	Longest Run	100%	0.637	100%	0.350
6	Rank	95%	0.909	100%	0.534
7	FFT	100%	0.740	100%	0.911
8	Nonoverlap Template	PASS	PASS	PASS	PASS
9	Overlap Template	100%	0.091	100%	0.637
10	Universal	95%	0.066	100%	0.350
11	Approximate Entropy	100%	0.740	95%	0.911
12	Random Excursion	PASS	PASS	PASS	PASS
13	Random Excursion Variant	PASS	PASS	PASS	PASS
14	Serial	95%	0.350	95%	0.091
15	Linear Complexity	100%	0.440	100%	0.534

1) Pass rate >90% and P-Value > 0.0001 is considered random.
2) Pass means that the tests of all subcategories are passed

Summary of VC-MTJ Device Properties

MTJ Diameter	80nm
MTJ Resistance	25kΩ/60kΩ
TMR	140%
Write Voltage	1.4V
Precession Period	1.5n sec
Extracted VCMA Coefficient	41.5fJ/Vm

Summary of TRNG Performance

Entropy Source	VC-MRAM
Technology	65nm with Wire Bonded Device
Supply Voltage	1.2V
Array Calibration	No
Digital Post Processing	Yes
Pass NIST 800-22	All
Bit Rate	400Kb/s
Energy Efficiency	135pJ/bit

Figure 2.16: NIST 800-22 test results; Summary of VC-MTJ device properties and TRNG performance.

The random number output has a throughput of 400Kb/s and consumes energy of 135pJ/bit. The throughput and energy efficiency are limited by the large parasitic capacitance of the I/Os between chip and VC-MTJ devices. The performance is expected to significantly improve if the VC-MTJs are integrated with CMOS.

2.6 Conclusion

In this work, we demonstrate a TRNG using Voltage-Controlled MTJs that don't need calibration of write pulse. The dynamic properties of VC-MTJ is investigated. We also show a light-weight digital bias correction circuit that can correct from a wide input bias range. Furthermore, the potential of using VC-MRAM as a in-memory random number generator is explored as a high performance solution.

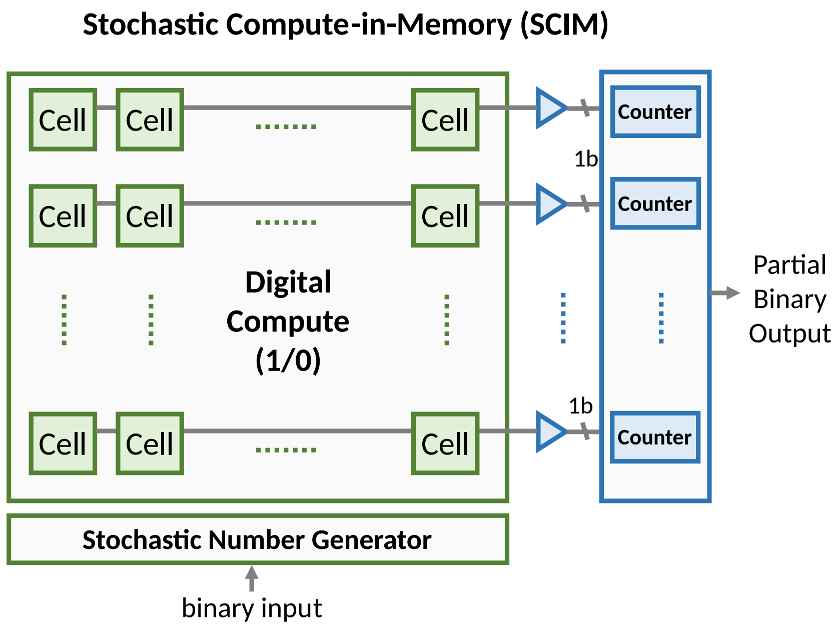
Chapter 3

An ADC-Less Stochastic Compute-In-Memory CNN Processor

3.1 Introduction

Deep Neural Networks (DNNs) have been widely used in the applications of computer vision, voice recognition and natural language processing [47] [48]. Deploying deep learning algorithms on mobile edge devices can significantly reduce decision latency and protect users' privacy, but the edge device's limited energy budget poses a big challenge on the hardware's energy efficiency. DNNs use a large number of convolution layers and parameters to improve accuracy, which leads to increasing model sizes. State-of-the-art models for ImageNet dataset require >100 MB of parameters and >109 of multiplication & accumulation (MAC) operations[49]. A general deep learning accelerator such as GPU and FPGA can process large DNN models on a workstation. It accommodates a wide range of applications and programmable computation precisions for training and inference. However, GPUs require a large power consumption that might exceed edge devices' power budget.

Recent works proposed custom deep learning accelerators for resource-constrained edge devices. The accelerators can achieve much higher energy efficiency for low-precision and inference-only neural network applications. [8] proposed a digital deep learning accelerator that uses efficient data flows to maximize data reuses in DNNs and significantly reduces the off-chip memory access costs. The accelerator builds an array of processing units with fixed point computation logic and local scratch pads. However, data movement costs from the on-chip memory dominates the overall energy consumption.



	Stochastic CIM	Analog CIM
Number Conversion	Digital (SNG, Counter)	Analog (ADC, DAC)
Area Density	High (1b Digital logic)	Low (large ADC)
Accuracy	Match INT 8b, Limited by SC error	Match INT 8b Limited by noise, mismatch
Latency vs Precision	Exponential (match INT 6-8b)	Square (Bit Serial INT)

Figure 3.1: High level diagram of a Stochastic Compute-In-Memory macro, comparison between SCIM and analog CIM. .

Compute-in-Memory (CIM) accelerator for deep learning is an emerging solution that provides high energy efficiency by embedding computation logic inside the memory array to reduce the data movement cost. The custom designed CIM macros are used for both memory and computation. During computation, multiple rows are activated at the same time to perform matrix-vector multiplications. Many recently proposed CIM solutions are based on computing in analog domain and converting to binary results by Analog-to-Digital Converters (ADC). Authors of [9] and [11] proposed accumulating bit cell's current on the bit line. However, the computation accuracy is significantly limited by the transistor's local mismatch and nonlinearity. Although large number (64−128) of rows are activated in parallel, only 3 4bit accurate computation accuracy can be achieved per dot-product operation. ADCs also cause large area and energy overhead. Other researchers have proposed charge-based compute [10] using metal-based capacitor embedded on top of the memory's bit cell at the back end. Multiplication results are stored as charges on the capacitor, and then accumulated on the bit line. Due to the large size of the capacitor, much better matching property is achieved. More than 2000 parallel rows and 8-bit computation accuracy are demonstrated. However, charging the bit cell's capacitor causes a significant amount of energy. Although capacitors have good matching properties, author of [50] found out that global process, voltages and temperature variations can greatly degrade the computation accuracy.

To alleviate analog compute's error and energy overhead problems, researchers have proposed robust digital Compute-In-Memory macro [50][12]. 1-bit multiplication is done in the bit cell and accumulated by in-memory digital adder tree. It achieves accuracy close to fixed-point computation, but adder trees degrade macro's density. Stochastic Computing (SC) is a digital probabilistic computing framework that uses random bit streams to represent numbers and simple bit-wise digital logic (AND, OR) gates to compute in the domain of probability. Previous work [51] has shown a deep learning accelerator using standard cell's digital circuit to implement SC. It has demonstrated significantly improved computation density and energy efficiency compared to digital accelerator. In this work, we propose Stochastic Compute-in-Memory (SCIM) deep learning accelerator which combines the benefits of SC and compute-in-memory architecture. Due to digital computation of SC, the CIM macro eliminates the large and power-hungry analog blocks: DACs/ADCs. The OR-based SC accumulation is embedded as the memory's wired-OR structure on the compute line. The accumulation is a 1-bit operation, and therefore does not cause area overhead. An efficient training for SC is performed and achieves comparable accuracy to INT8 on MNIST and CIFAR-10 classification. The DNN accelerator achieves energy efficiency up to 7.96TOPS/W, which is 2-3x higher than analog CIM solutions. The main contributions of our works are: 1) A bit-parallel dataflow that maps bit-wise stochastic computing to SCIM macros in parallel. 2) Storing pre-converted stochastic numbers in memory to achieve massive reuse of stochastic number generator 3) Computation skipping technique to reduce SC stream length when convolution layer is followed by average function.

3.1.1 Stochastic Computing

Stochastic Computing (SC) represents numbers and performs computation in the probability domain. The value of a number is represented by the probability of ones in a random binary bit stream. For example, in a stochastic stream of 8 bits A=01000001, 2 bits are one and 6 bits are zero. It represents the value of 2/8, Fig.3.2. Each bit represents an equal weight of 1/N, where N is the stream length. The position of the one should be randomly located in the bit stream to avoid correlation between different bit streams during computation.

The conversion from binary to stochastic numbers is done by Stochastic Number Generator (SNG) as shown in Fig.3.2. A randomness source is required. True random number generator (TRNG) can generate uncorrelated bit streams, but the high energy and area makes it impractical to implement. Pseudo-random number generators such as linear feedback shift register (LFSR) are commonly used. To avoid correlations between different SNGs, LFSRs with different polynomial functions or seeds are used, which can be easily programmed on the chip. The conversion can be implemented by comparator or a chain of multiplexer.

If the location of the ones in the SC bit stream are chosen randomly, the multiplication arithmetic of SC follows the rules of probability: $Prob(A \cap B) = Prob(A) \times Prob(B)$. This can be

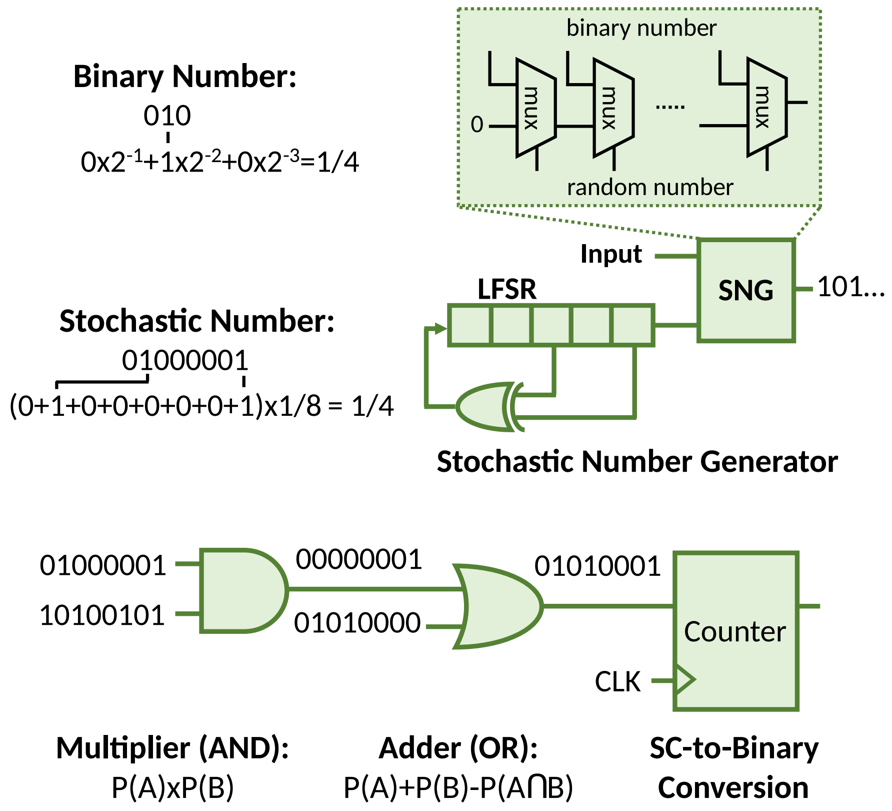


Figure 3.2: Number representation and basic building blocks of Stochastic Computing (SC).

achieved by AND gate in hardware. Addition can be implemented by multiplexer and OR gates [52]. Multiplexer achieves the scaled addition between input streams. The control bit randomly selects between the inputs, which performs an average function. When addition is performed between a large number of inputs, mux-based addition leads to small output amplitude due to the scaling factor. APC accumulates and adds the input streams by binary adder tree to generate binary output. Although it performs accurate addition, the adder hardware costs large area and energy consumption. The OR gate performs approximate addition follows the union of two random variables $Prob(A \cup B) = Prob(A = 1) + Prob(B = 1) - Prob(A \cap B) \approx Prob(A) + Prob(B)$. Although the OR-based accumulation does not implement the accurate addition function, a previous work proposed to account for the nonlinearity of the OR accumulation in neural network's training [46].

Previous works of SC proposed training methods to improve the computation errors in accumulation and stream generation when used in deep learning [46][53]. SNGs using pseudo random number generator create deterministic bit streams given the same inputs and stream. The training model can learn the errors in the fixed stream and also allow moderate sharing of random numbers between SNGs without degradation to the accuracy. State-of-the-art classification accuracy has been demonstrated for MNIST/CIFAR-10 data set with deep neural network that is comparable to 8-bit fixed-point implementations.

3.1.2 Motivation and Challenge of SC

Memory access consumes significantly higher energy than conventional binary computation. Reducing the number of on-chip and off-chip memory access is critical to achieve higher energy efficiency. Previous work of digital accelerator [8] has significantly improved energy efficiency by maximizing the reuse factor and reduces off-chip memory access. The energy consumption, however, is still dominated by the on-chip memory access. Stochastic Computing uses tiny bit-wise logic gates to achieve extremely high parallelism and further reduce the on-chip memory access. The previous work [8] has built a highly programmable Stochastic Computing deep learning accelerator using fully synthesizable standard cell digital gates. The density of the multiplication and accumulation (MAC) achieves $38.4K/mm^2$, which is >10x higher than fixed-point logic. The area includes all the necessary components for stochastic number conversion ,buffers and control logic.

Compute-In-Memory (CIM) architecture can achieve even higher reuse factor for two reasons: 1) No memory access is required for weight since computation happens inside the memory. Reloading of the CIM macro is necessary for larger network, but the cost can be reduced by a large reuse factor. 2) CIM breaks the bandwidth limits of the Von Neumann architecture by activating multiple rows or entire array for computation. The dense array can achieve very high parallelism. Combining the benefits of Stochastic Computing's light-weight digital computation with CIM's low data movement cost is an attractive solution. The Stochastic Compute-in-Memory (SCIM) accelerator embeds robust digital multiplication and accumulation circuits in the memory and does not suffer from analog noise. The accumulated output is a 1b signal at logic level and does not require large ADCs. Despite the benefits, there are a few challenges that need to be overcome: 1) Stochastic Number Generator (SNG)'s energy consumption is significantly larger than the SC's computation. A large SNG reuse factor is needed to reduce the average cost per operation. 2) The stream length of the stochastic number representation increases as power of two with number precision. The long stream length degrades the energy efficiency and the throughput. 3) SC's OR-based approximate accumulation is a nonlinear addition function. An efficient model of the accumulation function is needed during neural network training is required to achieve comparable accuracy to integer.

3.2 Stochastic Compute-In-Memory Processor

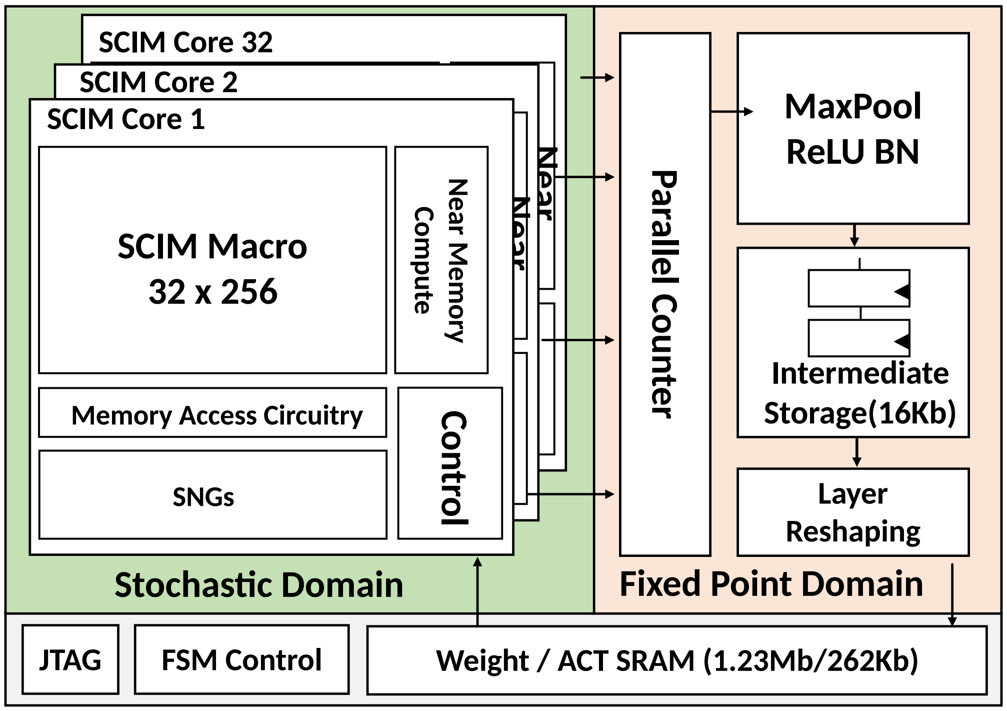


Figure 3.3: Architecture of Stochastic Compute-In-Memory (SCIM) accelerator.

Figure.3.3 shows an overview of the CNN processor based on the concept of Stochastic-Compute-In-Memory (SCIM). It has 32 SCIM macros that perform bit parallel processing of convolution between activations and weights in Stochastic Computing (SC) domain. The convolution results are converted to fixed-point domain by the parallel counters. ReLU, max pooling and batch normalization are performed in fixed-point. The layer outputs are stored in the output SRAM until the next layer starts processing.

3.2.1 Bit Parallel and In-Memory Compute Data Flow

Conventional Stochastic Computing (SC) operates serially since the bit stream is generated one bit at a time in sequence. Embedding bit-serial SC computation in memory requires also embedding stochastic number generators (SNG), which might occupy a large area. To enable stochastic computing-in-memory, entire bit streams can be stored in memory and computation will happen in parallel. A bit-parallel and in-memory SC data flow is designed to store input bit streams since activations are positive values and require less storage compared with weights. Unipolar stochastic representation directly maps probability of ones to range of [0, 1], therefore only representing positive numbers. Split-Unipolar SC representation is a simple way to use the difference between two unipolar bit streams, positive and negative stream, to represent signed number between -1 and 1. Fig.4 shows an example. For a negative number: W=-2/8, the magnitude 2/8 is encoded in the negative stream and the positive stream is zero. The probability difference between positive and negative stream is -2/8.

The Multiplication and Accumulate (MAC) arithmetic of the split-unipolar stochastic bit streams needs to account for the cross product between the operands' positive and negative streams. For neural networks, inputs to each layer are positive numbers. For example, the first layer's inputs are images with positive pixel intensities and the hidden layers' inputs are the outputs of nonlin-

Split-Unipolar Representation

positive stream: w_p : 0, 0, 0, 0, 0, 0, 0, 0
negative stream: w_n : 0, 1, 0, 1, 0, 0, 0, 0
$W = \mathrm{prob}(w_p) - \mathrm{prob}(w_n) = 0 - 2/8$

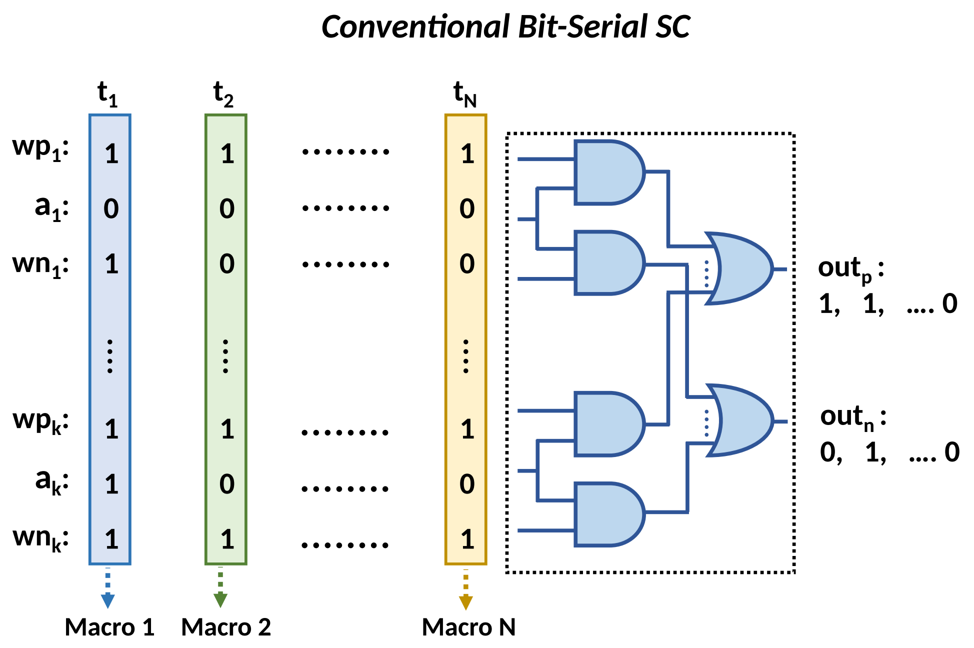
	Positive Stream	Negative Stream
Activation (Unsigned)	a	0
Weight (Signed)	wp	wn
Output (Signed)	outp	outn

Stochastic MAC of K inputs :

$$out_p = \bigcup_{k=1}^{K} a_k \cap wp_k$$

$$out_n = \bigcup_{k=1}^{K} a_k \cap wn_k$$

$$OUT = Prob(out_p) - Prob(out_n)$$



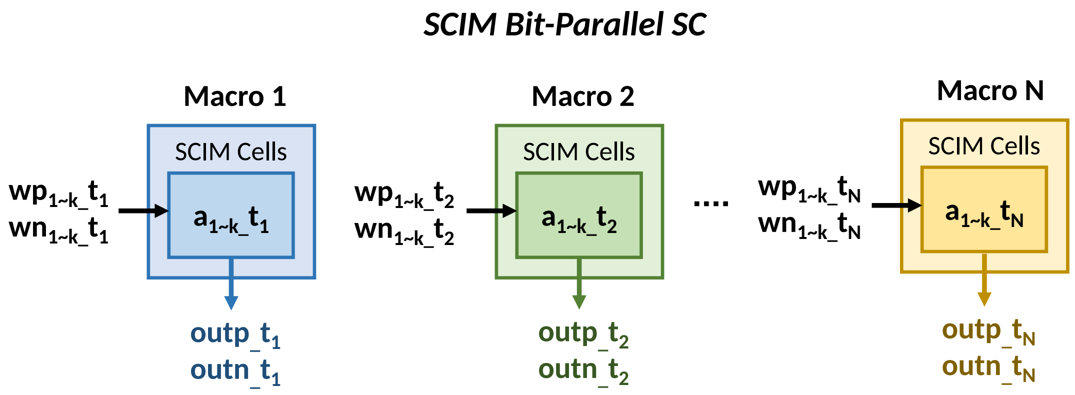


Figure 3.4: Split-Unipolar representation for inputs and weights. Bit-parallel processing and mapping to SCIM macros.

ear activation function such as ReLU and the negative values are clipped to zero. Inputs only need one unipolar bit stream to represent positive values. Weight parameters are signed numbers and require both positive and negative streams of split-unipolar representation. The cross product between input's positive stream: a, and weight's split-unipolar streams: wp/wn is illustrated in Fig.3.4. Assume both input and weight are 1-D vector with K elements. Each element of input a is multiplied with wp and wn by intersection AND logic: $a_k \cap wp_k$ and $a_k \cap wn_k$, and then accumulated by the union logic. The output streams of the two union operation are positive and negative stream of output: outp and outn, which are the split-unipolar representation of the dot product's result.

Fig.4 shows how bit wise processing of the conventional SC MAC operation is mapped to SCIM macros. Assume the stream length is N. Conventional bit-serial SC arithmetic process 1 bit at a time through the AND and OR gates. To perform stochastic computing in memory, each SCIM macro stores one bit representation of the input vector: macro1 stores $a_1\_t_1 \sim a_k\_t_1$, where $a_1 \sim a_k$ denotes the k elements of vector a and t_1 denotes 1st bit of stochastic bit stream. Macro2 stores $a_1\_t_2 \sim a_k\_t_2$ and macroN stores $a_1\_t_N \sim a_k\_t_N$. Each 1-bit representation of the weight vector is applied to macros: $wp_1\_t_1 \sim wp_k\_t_1$ and $wn_1\_t_1 \sim wn_k\_t_1$ are applied to macro-1. $wp_1\_t_N \sim wp_k\_t_N$ and $wn_1\_t_N \sim wn_k\_t_N$ are applied to macro-N. The dot product between inputs and weights are performed in the memory macros together. The bit-parallel processing unrolls the bit-wise operation in space so that all the output bits are computed and available in one cycle.

3.2.2 Stochastic Number Generator

A bottleneck with conventional SC is the large energy cost of the conversion from binary to SNs. Stochastic number generators (SNG) typically use pseudo-random number generators (PRNG) such as linear feedback shift register (LFSR) as the randomness source. The LFSR-based PRNG

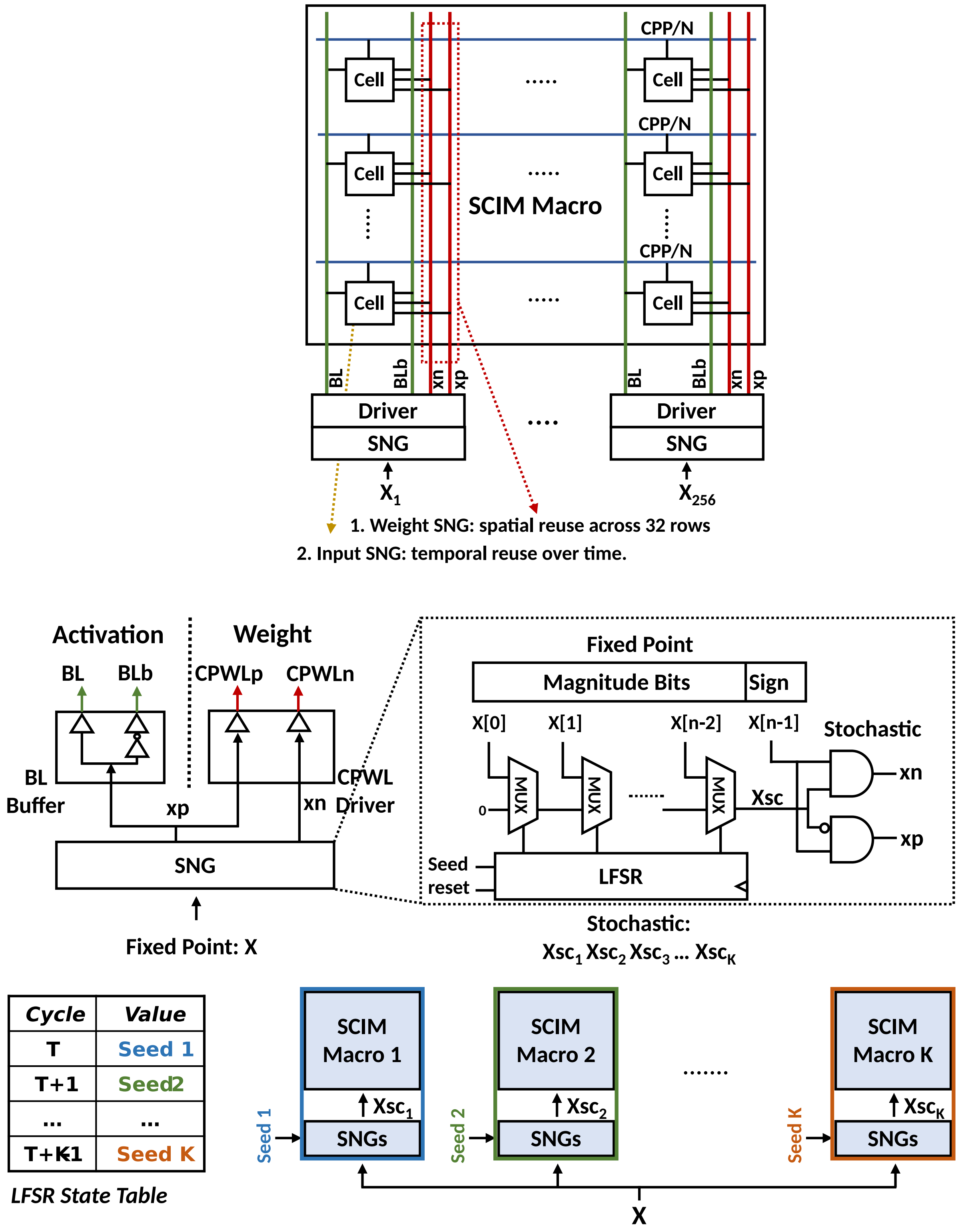


Figure 3.5: Stochastic number generator circuit; Dath path of SNG for input and weight; Seed schedule for parallel SNGs.

consumes 25x more energy than an SC MAC unit, which only performs 1-bit in-memory AND and OR operations. A large reuse factor of the SNG output is required to reduce the energy cost. Fig.5. shows the diagram of the SNG circuit. Each register of the LFSR controls one multiplexer. The multiplexers are cascaded, and each control signal selects between a binary bit of the input and the output of the previous multiplexer, Fig.5. N multiplexers are required to convert an N-bit binary number to stochastic bits. It can be shown that the SNG can almost accurately convert binary numbers if two conditions are met: 1) Maximal-length LFSR is used as the randomness source 2) The order of the LFSR's characteristic polynomial matches the bit width of the input binary number [53]. An N-bit maximal-length LFSR will iterate over all the possible combinations of N-bit numbers except zero and therefore has a period of 2N-1. Each N-bit combination is iterated exactly once. The most significant bit of the binary number is multiplexed to the SNG output for exactly 2^{N-1} times. Other bits are multiplexed with binary weighted frequencies: 2^{N-2}, 2^{N-3}, ... , 2^0, and therefore the frequency of one in the stochastic bit stream accurately represents binary input. Since the accelerator performs 8-bit computation with 1 bit for sign and 7 bits for magnitude. The LFSR size is chosen to be 7 bits. The demultiplexer at the output of the SNG selects between xp and xn based on the sign of the binary input.

SNG for Input Loading

The SCIM macro stores the pre-generated stochastic bit stream of inputs. Fig.3.5 shows how the per-column SNG supports both loading inputs and applying weights during computation. Each column has an SNG and driver for bit lines: BL/BLb and Compute word lines: CPWLP/N. The SNG converts the binary number X to split-unipolar stochastic representation: xp/xn. Since the activations use unipolar representation, only xp needs to be stored in the memory. During input loading, xp is sent to BL buffer, which drives BL/BLb and writes the xp to a bit cell. Each macro only stores one stochastic bit of the activation. A parallel stochastic number generation scheme supports the bit-parallel SC processing shown in Fig.3.5. The LFSR state table characterizes the changes in the shift registers at different cycles. To parallelize the SNG, each LFSR state is used to initialize the LFSR in different SCIM macros: seed 1 for the 1st macro and seed K for the Kth macro. The results of K SNGs in one conversion cycle are the same as a single SNG generating for K cycles.

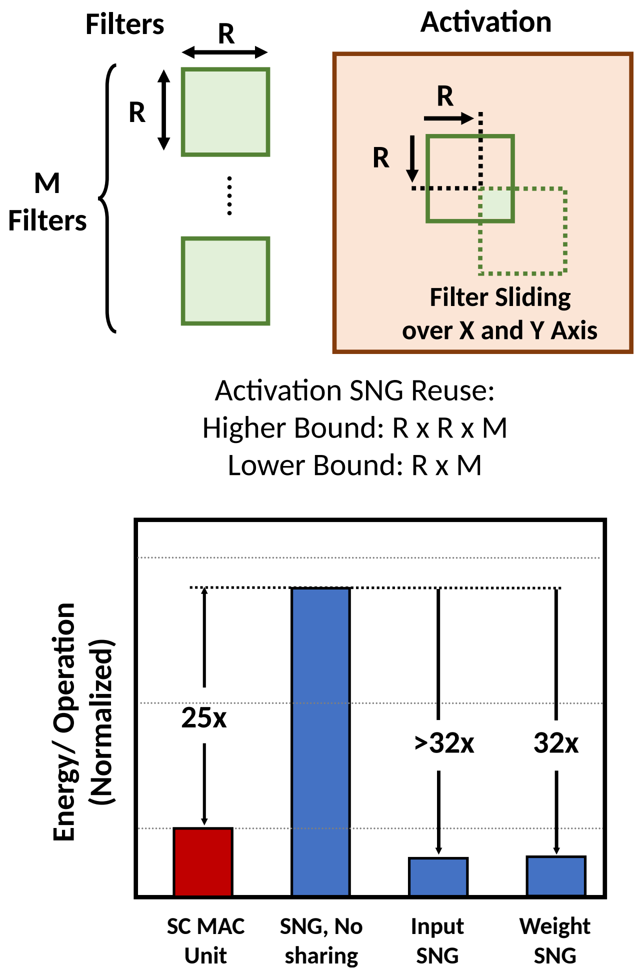


Figure 3.6: Reuse of the stored activations (top); Energy cost of SNG with and without reuse (bottom).

The stored activation stochastic bits can be reused from sliding windows and different output channels, shown in Fig.3.6. Assume the 2-D filter has the shape of RxR and there are M output channels, the activation can be reused with the maximal factor of RxRxM. The SNG cost is lowered by the same factor since SNG only consumes energy when activations are loaded into the memory.

The actual activation reuse factor depends on the size of each layer and whether the SCIM macro can support sliding windows across X and Y dimensions without reloading the activations. Our proposed accelerator computes Y-axis sliding windows in parallel due to the parallel operation of rows in the SCIM macro and X-axis sliding windows in serial. If the layer's X dimension is too large, the SCIM macro might not support sliding windows on the X axis without reloading the activations into the SCIM macro. The activation reuse over output channels can be easily supported with increased intermediate storage when the output channel size is large. Therefore, the lower bound of the activation SNG reuse factor is: RxM. In our experiment with CNNs supporting MNIST and CIFAR-10 classifications, the activation SNG reuse is >32.

SNG for Filters

The filters share the same SNGs as inputs. The binary filter coefficients are converted by SNGs in different SCIM macros in parallel to stochastic bits. The compute port word line (CPWLP/N) drivers buffer the split-unipolar bits xp and xn. All the rows in the SCIM macro operate in parallel, and therefore the energy cost of filter SNG and CPWL driver is shared by the number of rows: 32. The comparison of the energy cost of the activation SNG, filter SNG and SNG without any sharing is shown in Fig.3.6. Although the SNG energy is 25x higher than a single SC MAC unit, the activation and filter SNG energy cost is reduced by more than 32x due to reuse.

3.3 Stochastic Compute-In-Memory Macro

The Stochastic Compute-In-Memory (SCIM) macro embeds simple digital SC computation logic inside the memory to achieve high energy efficiency for matrix multiplication operations: AND gates for multiplication and OR gates for approximate accumulation. Each 1-bit stochastic representation of the activation is stored in the 10-T bit cell before the computation. The storage cell uses

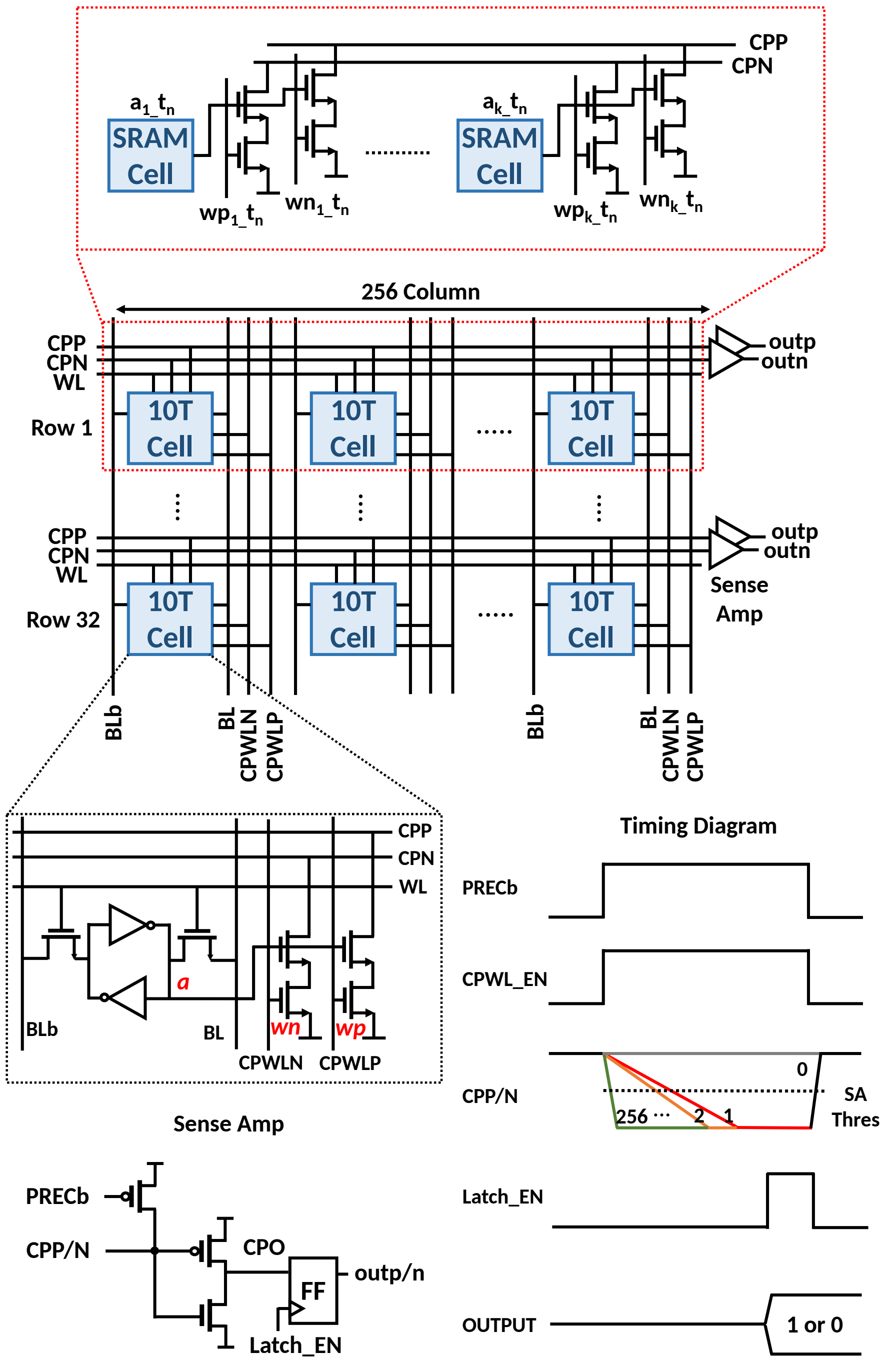


Figure 3.7: SCIM bit cell array; Circuit implementation of 10T SCIM cell and sense amplifier; Timing diagram of a compute operation.

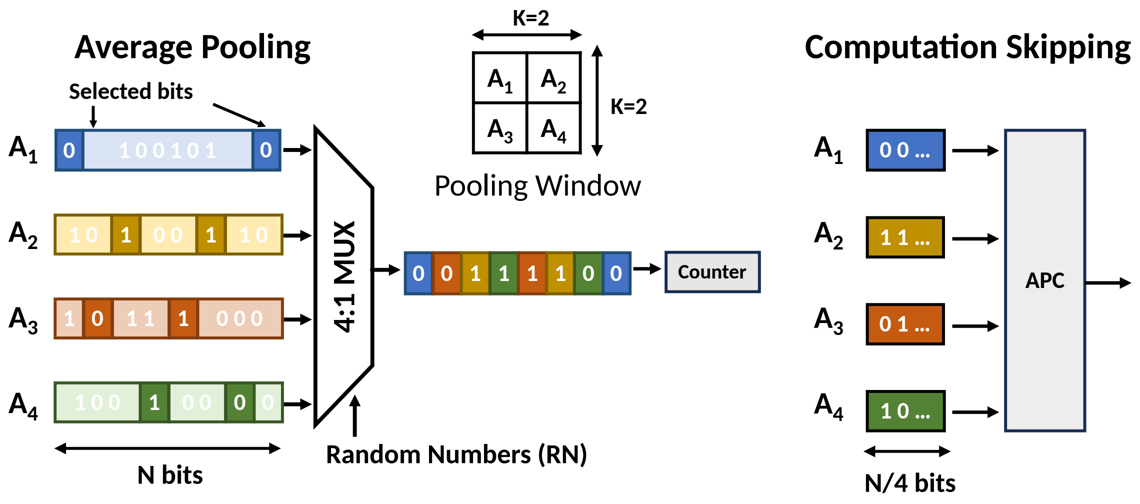
the standard 6-T SRAM cell design. The bit cell structure is shown in Fig.6 (bottom). Each pair of cascaded nMOS transistors performs an AND operation between the stored activation bit and a weight stochastic bit applied at the compute-wordline (CPWLP/CPWLN). The two multiplier circuits in the bit cell perform multiplication between activation (a) and weight (wp/wn), which correspond to one positive bit and one negative bit of a weight parameter. The shared compute port (CPP/CPN) across a row realizes a wired-OR operation between 256 bit cells. The CPWLP/N is routed in vertical direction and shared by all the bit cells in the same column. A simplified diagram of SC MAC units of one row is shown in Fig.3.7 (top). The computation follows the operation of the precharged pseudo-nMOS logic family. The timing waveform of the MAC operation is shown in Fig.6. The compute port is initially precharged to VDD and all the CPWLs are grounded. Once the computation starts, the compute port's precharging pMOS transistor is turned off and CPWL driver is enabled. If the OR accumulation result is zero, only leakage current will draw charges from the compute port. PVT and corner simulations show that the compute port only drops 12mV during the evaluation period. If at least one bit cell is conducting current, the OR accumulation result is one and the compute port will be discharged towards ground. The slowest discharging rate is when only one bit cell conducting current. The evaluation period is long enough for the compute port to be fully depleted for the worst discharging rate to ensure a robust operation. A simple inverter acts as a sense amplifier and buffer to provide tolerance to noise and coupling. Once the compute port evaluation is done, the 1-bit result is registered in a flip flop. The macro is 32 rows tall and 256 columns wide. Each row performs two 256-long dot product and the one macro contains 16.4K MAC units.

3.4 Computation Skipping for Average Pooling

A computation skipping technique is developed to save both latency and energy when the convolution layer is followed by an average pooling function. The Pooling layer is a necessary component in neural networks to make the output features less sensitive to the location of the input and reduces layer dimension to save computation complexity. Pooling based on the maximal or average value are both effective. Max pooling helps highlight the prominent pixel in the window while average pooling can smooth out the images. An averaging function can be implemented in stochastic computing using simple multiplexer logic. Consider an average pooling window of K by K. It is realized by a K^2:1 mux that selects one of the K^2 inputs randomly, Fig.3.8. The probability of the multiplexer's output is the average of input probabilities. For a 4-input average pooling function, each stochastic input is selected for 1/4 of its sequence length. For each input stream, the unselected stochastic bits do not contribute to the output and the computation for those bits can be avoided to save energy and latency, which is originally proposed by [46] . Each input only needs to compute for N/4 bits for an original N-bit stream. The shorter stream length equivalently scales each input by 0.25 and the four inputs are added by accumulative parallel counters (APC).

Stochastic Computing's energy consumption and latency increase exponentially versus computation precision due to its stochastic representation: N bit binary number requires 2^N stochastic bits and 2^N evaluation of bitwise SC logic. In contrast, fixed point digital logic's energy increases as N^2. An N-bit multiplier requires N^2 of full adders. Analog based CIM commonly uses bit-parallel bit-serial scheme to perform multi-bit multiplication [21], which also causes the energy to increase quadratically. Weight bits are stored as 1 bit per column and compute in parallel, but input bit is serially applied to the CIM macro in N cycles. For an N-bit computation, CIM requires N^2 of in-memory MAC and ADC evaluation. For 4-bit or lower precisions, the number of MAC evaluations is comparable between SC and ADC-based bit-serial CIM ($4^2 = 2^4 = 16$). However, an 8-bit SC computation requires $2^8 = 256$ MAC evaluations, which is 4x larger than the ADC-based bit-serial CIM.

A comparison of energy cost between CIM and SCIM macro is shown in Fig.3.9. The SCIM macro's energy cost is dominated by SNGs for input stochastic number conversion, counters for



$$Prob(Out) = \sum_{k=1}^{K^2} Prob(RN{=}k) \text{ x } Prob(Ak) = \frac{1}{K^2}\sum_{k=1}^{K^2} Prob(Ak)$$

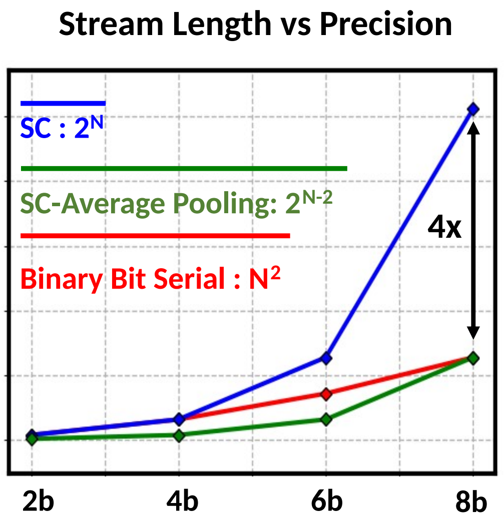
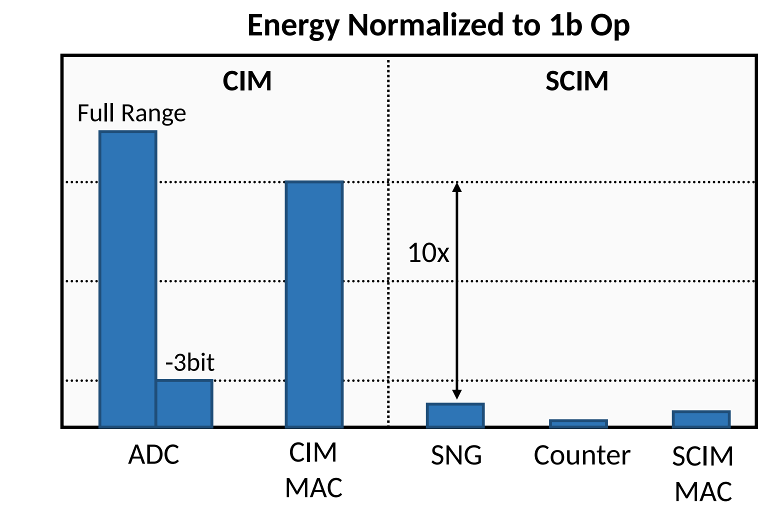


Figure 3.8: Computation skipping of average pooling function; Stream length vs precision function.

output binary conversion and in-memory SCIM MAC. The charge based CIM macro consumes energy from input driver, charge based CIM MAC and column ADC. The energy consumption of each block is normalized to 1-bit operation and plotted in Fig.9. The energy consumption of the charge based CIM uses energy breakdown from [21] demonstrated in 65nm technology. The ADC in the charge based CIM only accounts for 20% of the macro energy, but the ADC's precision is 3 bits less than the full dynamic range of the column dot product. If the ADC with the full



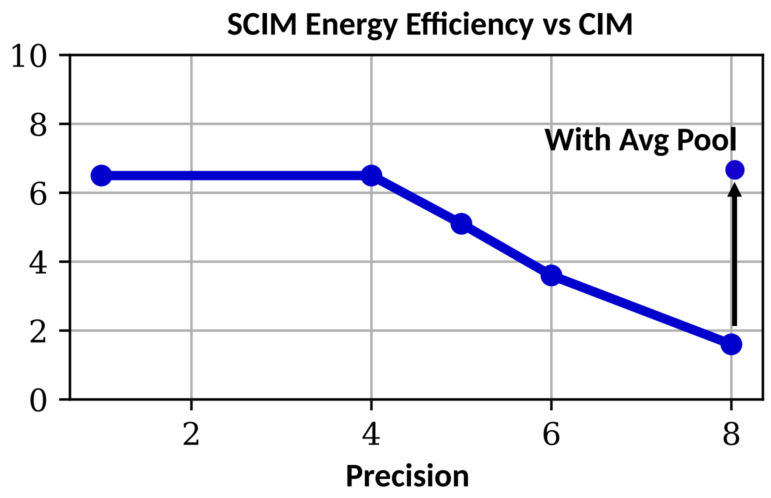


Figure 3.9: Energy cost comparison of all components in CIM and SCIM macros normalized to 1b Op; Energy efficiency improvement of SCIM over CIM for different bit precisions.

dynamic range is used, its energy will increase by 8x and dominate other parts of the energy. After summing up the components, the SCIM achieve 6x higher energy efficiency than CIM for 1-bit Op. For higher precision, SC benefits diminish due to the longer stream length. For example, 8-bit SC compute requires 256 1-bit Op, but CIM only requires 64 1-bit Op. The advantages of SCIM drop to 1.6x for 8-bit computation. The average pooling reduces the stream length by 4x and makes SCIM 6.4x more energy efficient than CIM.

3.5 Efficient Micro-Architecture for CNN

The proposed SCIM CNN accelerator supports end-to-end operation for convolutional neural network inferences and it is highly programmable to accommodate different layer topologies. In this section, we introduce different micro-architecture designs that support efficient implementation of CNNs.

3.5.1 Near-Memory Partial Binary Accumulation

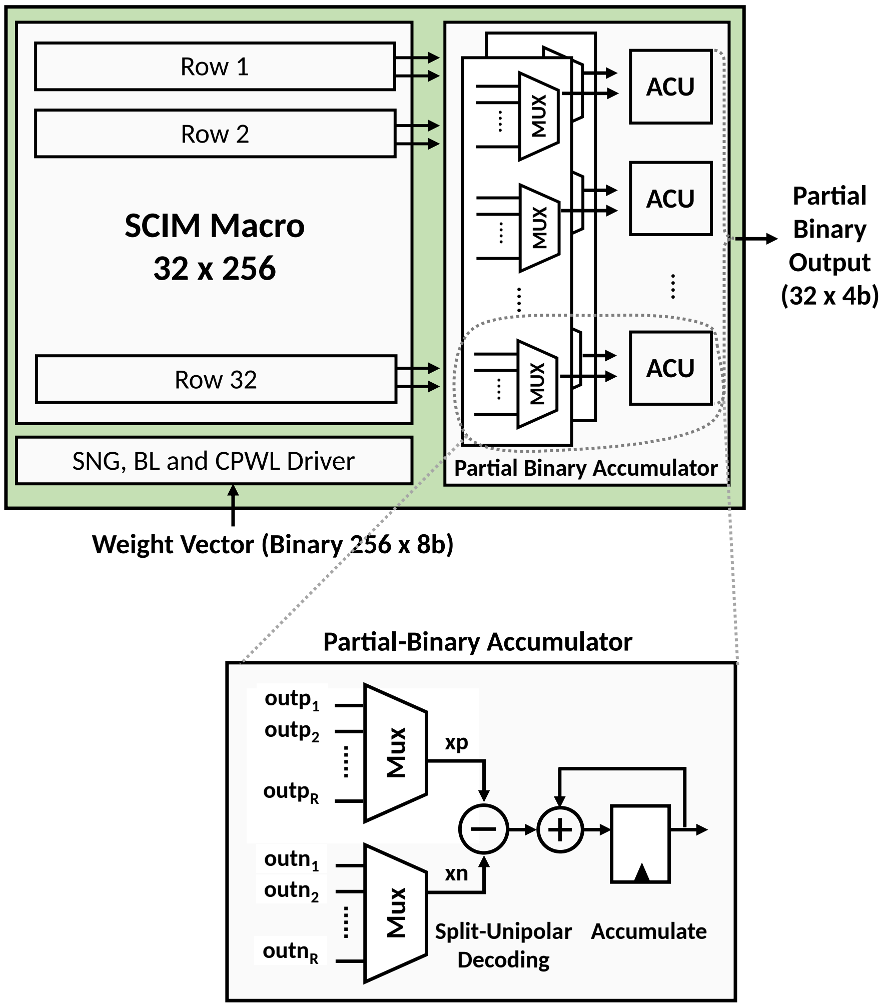


Figure 3.10: Near memory partial binary accumulator circuits.

The near-memory accumulation circuit supports partial binary accumulation between rows. Partial binary accumulation is a technique to improve accuracy by breaking large SC dot product and add partial SC outputs using fixed point adders. The in-memory dot product of two vectors in SC domain uses AND logic as the multiplier, then the multiplication results are accumulated by the wired-OR structure of the SCIM macro's compute port. The partial binary accumulator block has 32 rows, matching the height of SCIM macro. Each row has two row multiplexer circuits, a 1-bit split-unipolar number decoder and a binary integrator circuit. The multiplexer can select every 8 adjacent rows to the accumulator's input, which supports the row stationary dataflow discussed in the next section, Fig.3.10. One multiplexer selects among the positive stochastic bit of the rows' output (outp), and the other multiplexer selects the negative stochastic bit (outn). The split-unipolar decoding circuits subtract 1-bit outn from outp. The 2-bit subtraction result is accumulated by the fixed-point accumulation circuit.

3.5.2 Input and Row Stationary Dataflow for Conv Layer

The input activation's stochastic bits are stored in the SCIM macro due to the shorter stream length of the unsigned numbers compared to the signed weight parameters. An Input and Row Stationary (IRS) dataflow is used to maximize the input reuse during the convolution. Instead of flattening the 3-D activation tensor into a 1-D vector, only the depth channel is fattened to the same dimension as row so that rows can be reused during convolution sliding. Each activation row is stored in one row of the SCIM macro. A total of 32 activation rows are stored. The weight tensor is flattened in the same manner to keep the row dimension. Convolution operation prioritizes sliding across Y direction because the physical structure of the macro allows multiplication between rows to happen in parallel. Weight rows are applied to the macro serially in different cycles, which is supported by a ping-pong structure of FIFO cyclic shift registers. Each register stores one row of the weight kernel. The ping-pong structure enables computation and loading new weight kernels simultaneously to hide the latency of SRAM access. One FIFO provides weight parameters to 32 SCIM macros and the other FIFO loads the next kernel from SRAM. Once the computation of one FIFO is finished, roles of two FIFOs are switched and the multiplexer will select the new FIFO. The weight row is converted to stochastic bits by SNGs inside the macro and applied to the compute port word line (CPWL). Each SCIM row performs a dot product between an activation row and weight row. In the next cycle, FIFO register shifts and the next weight row is applied to the macro. The row multiplexers shift the sense amplifier's outputs to the next row's partial binary accumulator such that the row outputs are accumulated diagonally, as shown in Fig.3.11. The row-wise convolution is done until all the weight rows are shifted from the FIFO. The row multiplexers can select every 7 rows' output, which support 3x3, 5x5 and 7x7 convolution kernel size.

Convolution layer's dataflow is controlled by the on-chip finite state machine, which can be broken down into three recursive FOR loops: (a) convolution sliding across rows, (b) convolution sliding across columns and (c) computing different output channels. Sliding across rows (a) is prioritized because of the row-parallel structure of the SCIM array. The order between column sliding (b) and output channel (c) depends on the layer and kernel dimensions.

1) If the SCIM macro's width is larger than the flattened kernel width, the macro's row can store activations of more than one convolution sliding windows. The column sliding is scheduled first before loading a different weight kernel because sliding across the SCIM macro's columns to avoid reading new weights from SRAM. Since the kernel width is smaller than the macro width, the weight kernel needs to be shifted to match the CPWL of a sliding window. This is achieved by a programmable barrel shifter at the output of Ping-Pong FIFO registers, shown in Fig.3.11. Other CPWLs not inside the sliding window are masked by zero. The shifting step for each sliding window is calculated by a controller based on the current sliding location and stride. Once a kernel has finished convolution over the stored input activation, the next kernel is applied by switching the Ping-Pong FIFO. If the activation map's size is larger than the SCIM array width, new activations

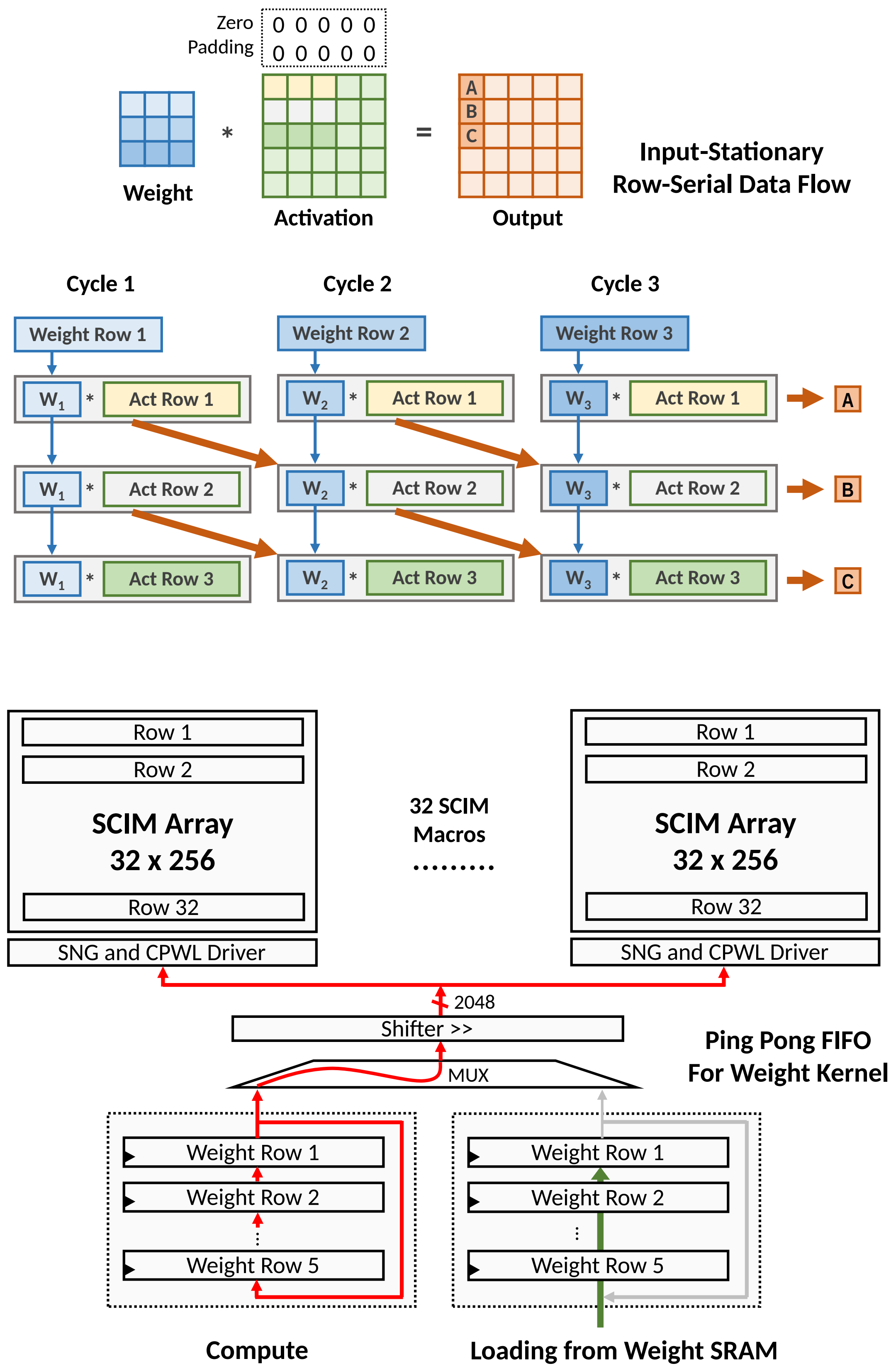


Figure 3.11: Input and row stationary data flow; Ping pong FIFO for weight loading.

need to load into the SCIM array to complete convolution over the entire activation map. Since the kernel width is smaller than the SCIM array width, the SCIM array is under utilized and the energy efficiency will degrade. This usually happens in the first few layers of neural networks when the number of output channels is small.

2) If the SCIM array width is equal or smaller than the flattend kernel width, the current activations stored in the SCIM array can only support one column-wise sliding window. Output channels (c) loop will be scheduled first because the energy cost of reading new weight kernels is smaller than loading inputs to the SCIM macros. To compute a different column-wise sliding window would require accessing input SRAM and loading 32 rows of input activation to the SCIM array. The kernel size (3x3 or 5x5) is much smaller than the height of activation map and therefore requires less SRAM accessing and communication energy. The output channels are scheduled to perform convolution over the current activations stored in the SCIM macros, then the SCIM array will be reloaded with new activation. Since the entire array is utilized for computation, the energy efficiency is the highest in this case. This usually happens for hidden layers in deep neural network where the flattened activation map's dimension is large.

3.5.3 Stochastic-to-Binary Conversion and Fixed Point Processing

The convolution outputs from the SCIM macros are converted to binary numbers in order to perform nonlinearity and scaling functions in fixed-point domain. The proposed accelerator performs bit-parallel stochastic processing where each SCIM macro computes one stochastic bit of the convolution operation in parallel. The stochastic outputs are converted to binary numbers by an array of parallel counters, Fig.3.12. Each SCIM macro produces 32 4-bit partial binary outputs. The outputs corresponding to the same convolution result in all 32 SCIM macros are passed to a parallel counter, which adds 32 binary numbers and produces a 9b result. The parallel counter uses a binary tree architecture with 2-stage pipeline to reduce the impact on the overall latency. The first pipeline register is placed after the third adder stage, which makes the delay of two pipeline sections about the same.

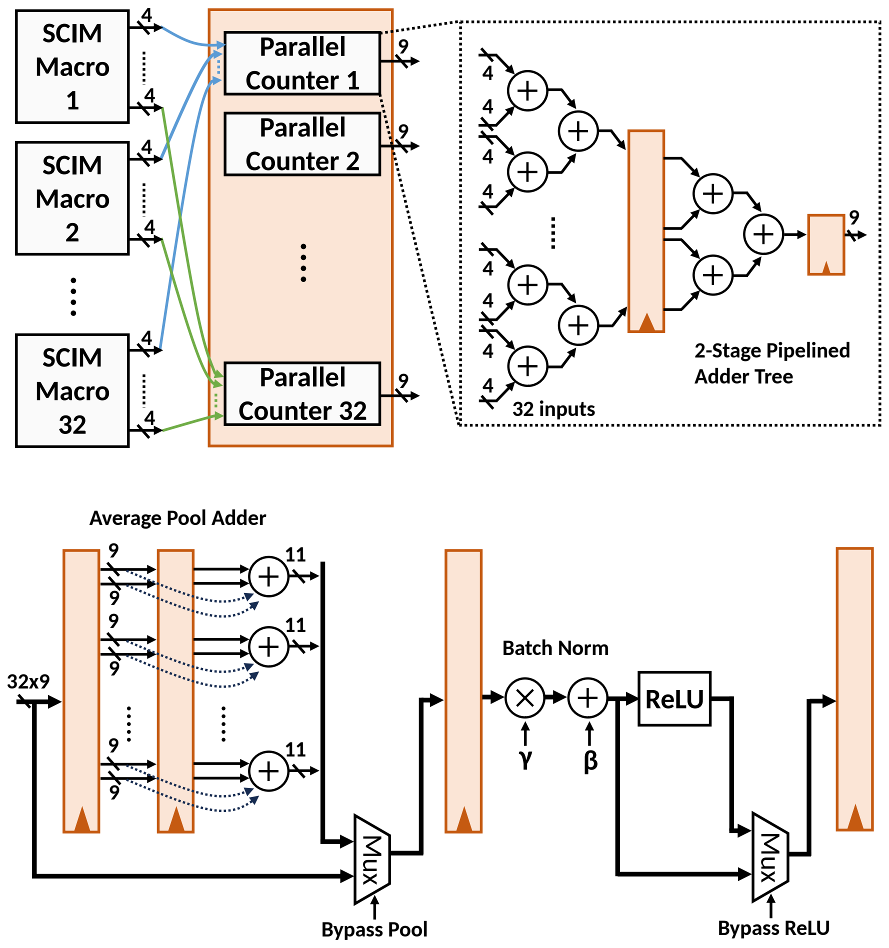


Figure 3.12: Parallel counter for stochastic to binary conversion; Fixed point domain processing.

After the convolution outputs from SCIM macros are converted to binary numbers, the output processing functions such as pooling, batch normalization and ReLU are done in fixed-point domain, Fig.3.12. They are implemented in pipelined stages and do not influence system's throughput. The first step of the fixed-point processing is average pooling. The computation skipping implemented in the Stochastic Computing domain reduces the stream length and therefore implicitly scales down the output value by the number of elements in the pooling window. To complete the average pooling function, the scaled outputs within pooling window need to be added. This design supports a pooling window of 2 x 2 and the inputs to the average pooling's adders are available at two different clock cycles. In the first clock cycle, the outputs from the parallel counters contain one column of the output feature map and they are stored in the shift registers next to the average pooling adders. The adjacent output feature map's column is computed after the kernel slides to the next column. Once the results of two adjacent output columns are available, the adder sums four inputs from the shift register. The average pooling can also be bypassed to support layer without the pooling function. The results are scaled with batch normalization's scaling parameter and then subtracted with bias. The batch normalization's outputs are then processed by ReLU function, which clips the negative values to zero. The outputs are loaded in the activation SRAM on chip until the start of the next layer.

3.5.4 OR-Accumulation and Neural Network Training

There are two main sources of random errors in the proposed SC system: 1) OR-based nonlinear accumulation function and 2) random bit stream representation generated from SNG. The OR-based accumulation is a nonlinear addition function. Take a 2-input OR operation as an example. If the two input stochastic bit streams have the probability of a and b, the output probability is a+b-ab. When multiple inputs are accumulated by the OR-based accumulation, the output can be derived as shown in the Fig.3.13, and approximately equals to $1 - e^{-s}$, where s is the accurate sum of all the inputs. The plot of OR-accumulator's outputs vs accurate sum of the inputs is shown in Fig.3.13 (top). The curve is close to a linear function in the range between 0 and 1. When the inputs become large, the accumulation saturates at 1 and becomes nonlinear. The backpropagation of the neural network training needs to account for the accumulation error. While modeling the stochastic computing bit by bit is the most accurate way, it takes impractically large amount of time to converge. We model the OR accumulation as a nonlinear activation function after an accurate accumulation, as proposed in [46]. The error in random bit stream representation is modeled by

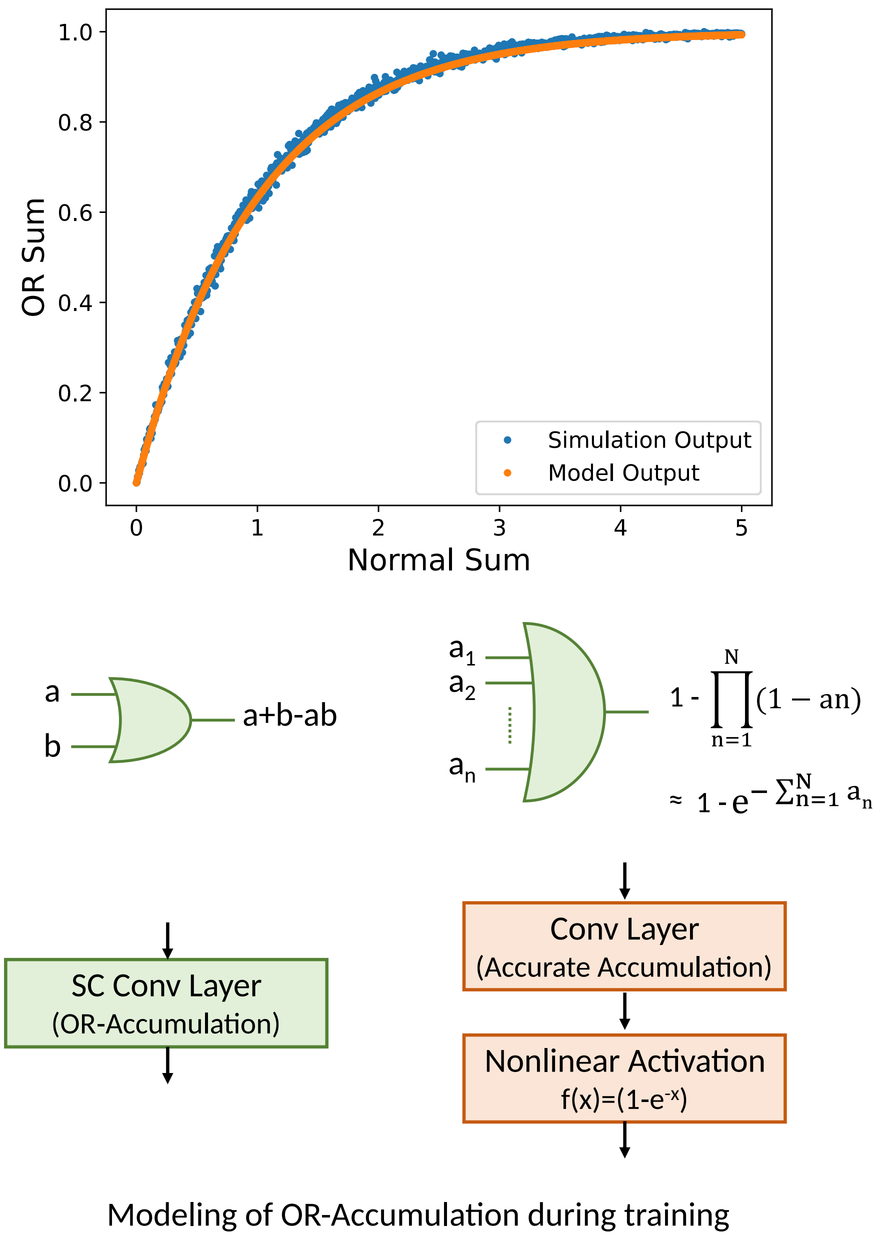


Figure 3.13: OR accumulation's output vs accurate sum. Modeling of OR-accumulation during training.

using the exact LFSR sequence and SNG during training. Multiplication is also modeled using AND operation between stochastic bit streams of input and weight. The multiplication outputs are converted to floating point and accurately added. The details of modeling the errors in stream generation is discussed in [53].

3.6 Prototype Measurement Results

The accelerator chip is fabricated in 65nm CMOS technology, and a macro test chip is fabricated in 14nm to better characterize the performance of the macro individually. The accelerator chip in 65nm has an area of 9.36 mm^2. A die photo is shown in Fig.3.16. The activation and weight SRAM are located at the bottom of the chip. 32 SCIM cores are placed in a 4x8 array and contain 260kb in-memory storage and 520Kb of MAC units in total, which achieves 130Kb/mm2 MAC density. The nominal voltage of the 65nm process is 1V. The macro's read/write and dot product function are verified at different supply voltage and operating frequency, shown in Fig.3.16. The maximal clock frequency at 1V is 5MHz. It is limited by a problem of on-chip power delivery structure and could achieve a much higher frequency and throughput given a more optimized design. The macro is fully functional down to 0.7V supply with a clock speed of 3.2MHz. The area breakdown of the SCIM core and whole chip is shown in Fig. 16. The SCIM core area is dominated by the bit cells and the sense amplifers only occupy 2% of area. SNG and row accumulator accounts for 15% and 6.25% of the SCIM core area. Input and Weight SRAM accounts for similar area as SCIM cores in the top level. The rest of the area is split among local register buffers, parallel counters, layer processing circuits, FSM and JTAG interface.

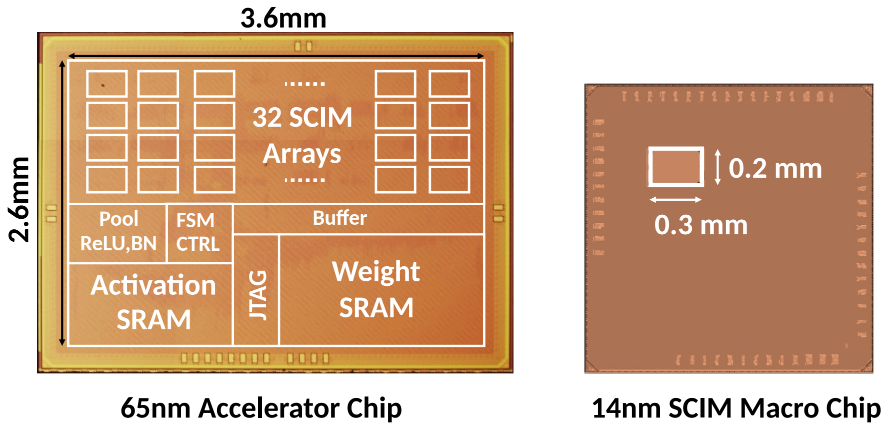


Figure 3.14: 65nm and 14nm Chip micrograph.

CNNs for MNIST and CIFAR-10 datasets are demonstrated on the 65nm accelerator chip, and

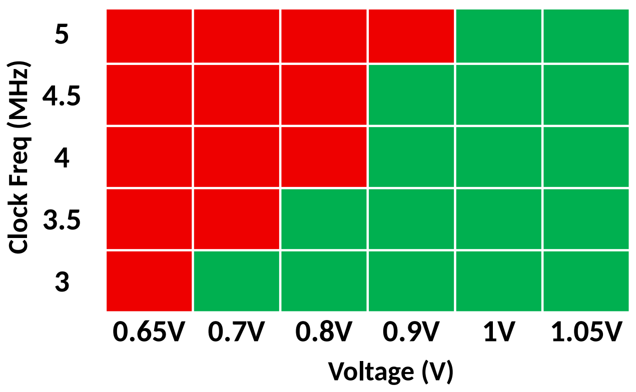


Figure 3.15: 65nm Chip Clock frequency vs Voltage.

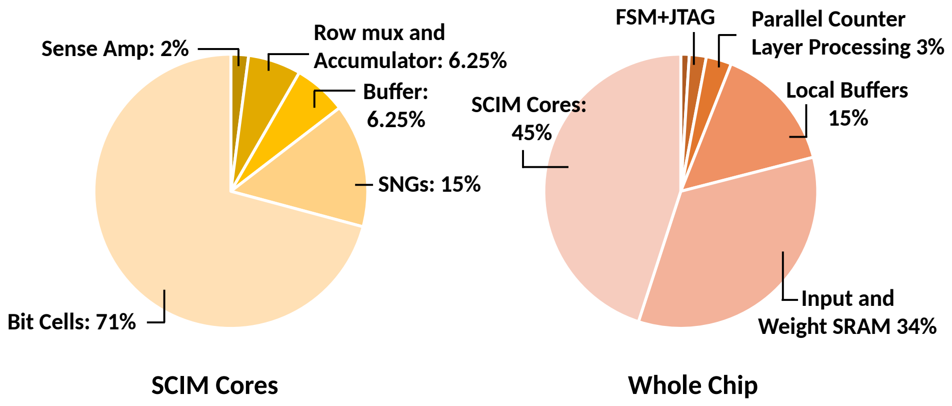


Figure 3.16: Area breakdown of SCIM cores and whole .

the performance is summarized in Table.3.1. We designed and trained CNN networks that can keep all parameters on the chip during computation. The measurement of energy consumption includes all components on the chip and there is no off-chip data movement during computation. For MNIST dataset, LeNet-5 topology is used. There are two CONV layers, each followed by average pooling to take advantage of SC's computation skipping and three fully connected layers. The classification results of the MNIST dataset achieve accuracy of 99.1% over 1000 test images, which is close to the training accuracy. Activation and weight both have 8-bit precision. Due to small size of filters used in LeNet-5, the peak macro utilization is only 5.1% and it causes significantly degradation of energy efficiency. The peak system energy efficiency for MNIST classification is 0.35 TOPS/W.

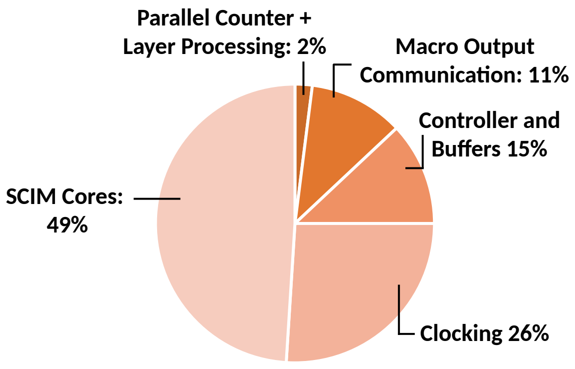
For CIFAR-10 dataset, we designed a 4-layer network called TinyConv-4. There are three con-

Dataset	MNIST				CIFAR-10			
VDD/CLK	0.8V/4MHz							
Performance Details	LeNet-5				TinyConv-4			
	Input	Layer	Utilization	E Efficiency	Input	Layer	Utilization	E Efficiency
	28x28x1	CONV (5x5)x6 Avg Pool 2x2	1.7%	0.15 TOPS/W	32x32x3	CONV (5x5)x32 Avg Pool 2x2	5.8%	0.4 TOPS/W
	14x14x6	CONV (5x5)x16 Avg Pool 2x2	5.1%	0.35 TOPS/W	16x16x32	CONV (5x5)x32 Avg Pool 2x2	31.3%	2.2 TOPS/W
	5x5x16	FC 25x120	4.8%	0.32 TOPS/W	8x8x32	CONV (5x5)x64 Avg Pool 2x2	15.6%	1.1 TOPS/W
	120x1	FC 120x64	1.5%	0.13 TOPS/W	1028x1	FC 256x10	3.1%	0.2 TOPS/W
	84x1	FC 64x10	1.1%	0.1 TOPS/W				
Bit Precision	8bit Weight, 8bit Act, 11bit Out							
INT Training Accuracy	99%				78%			
SC Training Accuracy	99.2%				75%			
Measured Accuracy	99.1%				73.5% (Top-1), 98.4%(Top-5)			
Energy/Classification	18.3μJ				35.6μJ			
Latency	5.85msec/frame				11.5msec/frame			

Table 3.1: Neural Network Demonstration

volution layers with 5x5 filters followed by 2x2 average pooling, batch normalization and ReLU function. The last layer is a fully connected layer generating the classification results. TinyConv-4 is trained in both floating point and SC. The classification accuracy using INT8 is 78%. The training accuracy using SC achieves 75%. The test accuracy measured on the SCIM accelerator is 73.5% in 1000 images. TinyConv-4 is larger than LeNet-5 and utilization problem is slightly relieved, summarized in Table.1. The largest macro utilization is 31.3% at the second layer and the peak system energy efficiency is 2.2 TOP/S/W. The second layer's filter size is 5x5x32. After flattening to 2D, the size becomes 5x160. The width of filter, 160, is smaller than the width of SCIM macro, 256.

A convolution layer with full macro utilization is tested to measure the peak energy efficiency of the accelerator. Activation and filter with a flattened width of 256 and no sparsity is applied in activation and weight. The energy efficiency at 0.8V supply is 5.75 TOP/S/W. The energy efficiency at different supply voltages is tested and shown in Fig.3.17. The peak energy efficiency at 0.7V supply is 7.96 TOP/S/W. A breakdown of the accelerator's energy consumption is shown in Fig.3.17. The most dominant components are SCIM cores (49%), clock communication (26%), controller and local buffers (15%). The energy of sending 32 SCIM cores' output to parallel



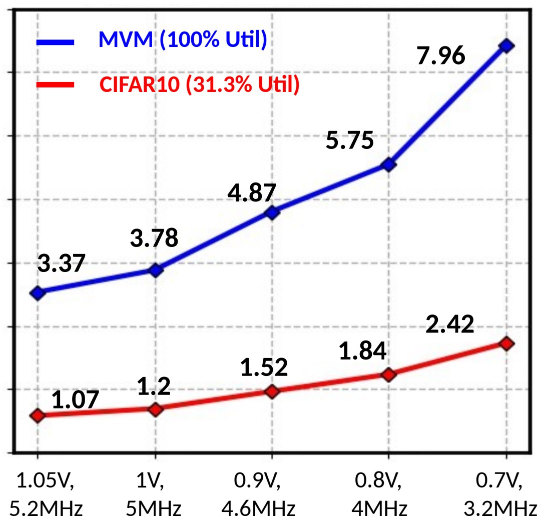


Figure 3.17: Energy breakdown and energy efficiency vs voltage.

counter and loading weights from SRAM to 32 SCIM cores account for 11%. Parallel counter and fixed-point processing account for 2%.

Another test chip is fabricated in 14nm to characterize the performance of the SCIM macros. The summary of the chip in comparison with 65nm accelerator chip is shown in Table 3.2. It has 16 SCIM macros of 32x32 array. The bit cell and sense amplifier design are the same as 65nm chip. Each row performs a 32-long dot product and generates 2 stochastic output bits (outp/outn). The 16 SCIM macros generate 32 stochastic bits for a given MAC operation, which is equivalent to 5b computation precision. The macro's outputs are directly accumulated by the parallel counter without row mux and accumulator at a frequency of 130MHz. The measured energy efficiency is 258 TOP/S/W. The energy efficiency scaled to 8b operation is 32.3 TOP/S/W. If average pooling is used in combination with the convolution layer, the 8b operation has an energy efficiency of 130 TOP/S/W.

A comparison table with other works is shown in Table 3.3. The 65nm chip packs 520Kb of in-memory MAC unit on chip, which is way larger than most of other works expect [10]. With smaller number of on-chip MAC units, more data movement is needed from on-chip buffer or off-chip DRAM, which might not be account in the energy measurement. Our work builds a complete

	14nm Macro Test Chip	65nm Accelerator Chip
SCIM Macro Size	32 x 32	32 x 256
Number of Macro	16	32
Number of 1b SC MAC	32.7K	542.3K
MAC Density	850 Kb/mm^2	130 Kb/mm^2
Supply Voltage	0.65 V	0.7 V
Maximal Frequency	130MHz	5MHz
Operations	SCIM Matrix Multiplication	SCIM Matrix Multiplication, Row mux and accumulator Average Pooling (2x2), ReLU, Batch Norm, CNN FSM Controller
SC Stream Length	32	64 or 256 (with avg pool)
Equivalent Precision	5b	6b or 8b (with avg pool)
System Energy Efficiency	Not Applied	7.96 TOP/S/W
Macro Energy Efficiency	258 TOP/S/W	20 TOP/S/W

Table 3.2: Summary of two chips.

CNN inference process on chip and keeps all the parameters on chip to account for all energy costs. The MAC density is also superior to other analog and digital MAC solutions. The 65nm chip has a density of 130Kb/mm2 and 14nm chip has a density of 860Kb/mm2, which is 2x higher than all other works. We report energy efficiency for 8b MAC operation and compare both system and macro efficiency. For SC, 8b computation requires processing 256-long stochastic bit stream. Energy efficiency with and without average pooling are reported. The 65nm accelerator has a peak system energy efficiency of 7.96 TOPS/W and 20 TOPS/W with average pooling. The 14nm macro chip achieves 140 TOPS/W for 8b compute with average pooling and 35 TOPS/W without average pooling.

3.7 Conclusion

In this work, we have demonstrated an ADC-less Stochastic Compute-In-Memory (SCIM) accelerator for convolutional neural networks. The accelerator has 32 SCIM macros storing the pre-converted stochastic numbers of activation and compute in a bit-parallel way. Since the OR-based

	Digital	Analog CIM			Digital CIM		SC	SCIM	
	Chen, ISSCC 16	**Dong, ISSCC 20**	**Jia, ISSCC 21**	**Yue, ISSCC 21**	**Yue, ISSCC 23**	**Chih, ISSCC 23**	**Romaszkan SSCL 22**	**This Work**	
Technology	65nm	7nm	16nm	65nm	28nm	22nm	14nm	**65nm**	**14nm**
Size	16mm^2	Not reported	25 mm^2	12 mm^2	3.75mm^2	Not reported	0.5mm^2	**9.4 mm^2**	**0.06mm^2**
Macro Area	Not Applied	0.0032mm^2	15 mm^2	1.7mm^2	0.27mm^2	0.2 mm^2	Not Applied	**4.2 mm^2**	**0.038mm^2**
Voltage	0.8-1.2V	1V	0.8V	0.65V (Digital) 1V (CIM)	0.4-0.7V	0.7-0.9V	0.6-0.9V	**0.7-1.05V (CIM), 0.8V(SRAM)**	**0.65 – 1V**
On-Chip CIM Size	Non CIM (168 PE)	4Kb	4.5Mb	64Kb	16.4Kb	64Kb	Non CIM (19.2K MAC)	**520Kb**	**32.7Kb**
CIM MAC Density	Non CIM	1250 Kb/mm^2	300 Kb/mm^2	37.6 Kb/mm^2	60.7Kb/mm^2	320Kb/mm^2	38.4Kb/mm^2	**130 Kb/mm^2**	**860 Kb/mm^2**
Precision	INT 16	INT 4	INT 8	Weight 4b, Act 8b	INT 8	INT 8	256b SC (INT8)	**256b SC (INT 8)**	**32b SC (INT 5)**
Peak System Energy Efficiency (8b Op)	0.332 TOPS/W	Not reported	Not reported	2.3 TOPS/W	12.8 TOPS/W	Not Applied	4.4 TOPS/W	**7.96 TOPS/W**	**Not Applied**
Peak Macro Energy Efficiency (8b Op)	Not applied	88 TOPS/W	30 TOPS/W	7.32 TOPS/W	68.7 TOPS/W	24.4 TOPS/W	Not applied	**5~20 TOPS/W**	**35~140 TOPS/W**
Throughput Density	0.3 TOPS/mm^2	29 TOPS/mm^2	0.2 TOPS/mm^2	0.014-0.46 TOPS/mm^2	1.52 TOPS/mm^2	0.15 TOPS/mm^2	0.3 TOPS/mm^2	**0.014 TOPS/mm^2**	**1.66 – 6.6 TOPS/mm^2**

Table 3.3: Comparison with other works.

SC accumulation is a 1-bit logic, the in-memory computation does not require an ADC. We have proposed a computation skipping technique to reduce the SC stream length by 4x when the average pooling layer is used. The accelerator has efficient architecture supporting full neural network application on chip: convolution is performed in SCIM macros and average pooling, ReLU, batch-norm are processed after stochastic numbers are converted to binary. Classification of MNIST and CIFAR-10 dataset are demonstrated with all parameters remain on chip. The accelerator has energy efficiency of 7.96 TOP/S/W for system and 20 TOP/S/W for macro. A test chip is fabricated in 14nm with only SCIM macro and shows 35 TOP/S/W in 8-bit operation and 140 TOP/S/W with computation skipping in average pooling layer.

Chapter 4

Stochastic-CIM Accelerator for Event Camera Object Tracking Application

4.1 Introduction

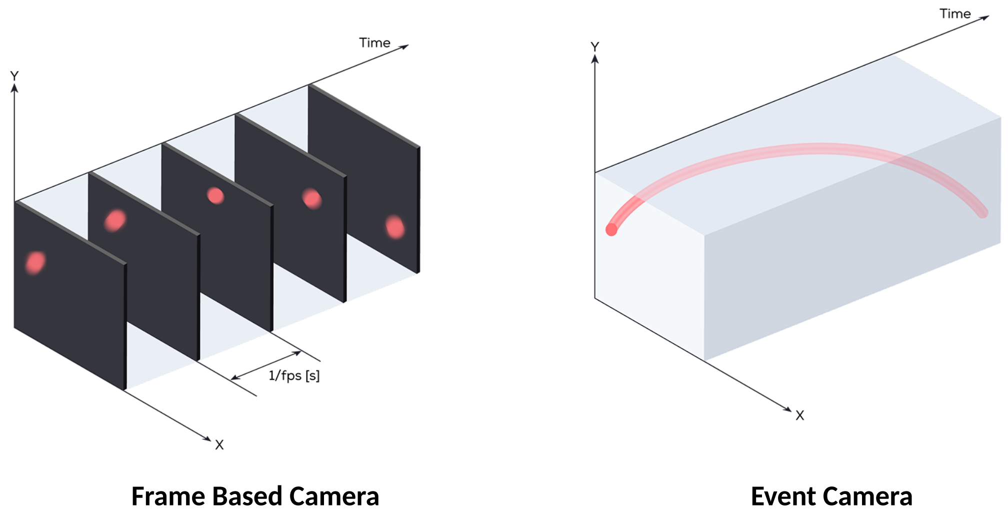


Figure 4.1: Comparison between frame and event camera.

Event Camera such as Dynamic Vision Sensor (DVS) is a bio-inspired imaging technology that has attracted many new computer vision applications recently [54] [55] [56]. DVS pixels only report changes of brightness and ignore static background that is not moving relative to the sensor. The sensor does not have a periodic sampling clock to read out pixels periodically like traditional frame-based cameras, Fig.4.1. Each pixel generates binary outputs (ON/OFF) asynchronously and can respond as fast as 1us. Due to these unique properties, event cameras can be used in high speed object tracking applications [57][58] [59][60]. However, hardware to process event camera's data faces serious challenges. The high-speed event output requires architectures with low latency and high energy efficiency. Researchers have experimented using event camera on Micro-Aerial Vehicles (MAV) to avoid high-speed obastacle and concluded that MAV needs to respond in mili seconds to avoid object moving faster than 10m/s [60]. For an event camera with a frame size of 640x480, the hardware needs to process > 100M events per second. A previous work has demonstrated an object tracking accelerator for event camera by near-SRAM circuits for accelerating spiking neural network for event data [61], achieving throughput of 11M events/s.

Compute-In-Memory (CIM) is an emerging accelerator architecture that can achieve extremely high throughput and energy efficiency by embedding computing logic inside memory [61][10][12]. Multiple rows or the entire CIM array are activated for computation in parallel, therefore breaking the bandwidth limits of memory in traditional Von Neumann architecture such as CPU/GPU. Previous CIM works have demonstrated analog-based computation where accumulation is done in current or charge domain and the analog output is converted to binary numbers by analog-to-digital converters (ADC) [10] [11]. ADC requires multiple cycles and long settling time to converge, therefore limiting the system's throughput. Typical ADC sampling rate reported in CIM accelerators is below 100MHz [10] [62] [1]. A way to increase the throughput in a given area (TOPS/mm^2) is to increase the number of multiplier cells per weight storage element. Each column of multiplier cells performing the dot product requires an ADC, and the area overhead of the ADC, therefore, significantly degrades the overall density. Stochastic Compute-In-Memory (SCIM) is an ADC-less CIM solution embeds Stochastic Computing's (SC) digital logic gate inside the memory and only requires 1-bit sense amplifier for each dot-product column that can operate a very fast rate [30]. SC represents binary number by the fraction of ones in a random bit stream, which requires 2^N bits for an N-bit number. Previous SCIM work [30] stores the entire stochastic bit stream inside the memory, and, therefore, occupies a large area.

In this work, we propose an SCIM accelerator that can achieve high throughput and energy efficiency requirement of the event camera. The SCIM macro stores binary number and reduces the number of storage cells for a parameter from 2^N to N. An in-situ Stochastic Number Generator (SNG) converts the binary number to stochastic bits in memory. The stochastic bits of every stored parameter are shared by 32 SC MAC units. Since the dot-product column does not need an ADC, operation frequency of $600-800$MHz is achieved. The event camera only captures the moving objects and removes noisy background. A convolutional filter-based algorithm is used that is highly effective and fast. The accelerator's architecture supports efficient convolution operation using both ternary events and multi-bit dense inputs to support flexible pre-pocessing of raw event streams. An early termination technique is proposed to turn off parts or entire operation of the chip when inputs have a high sparsity in the event processing mode. The accelerator achieves a throughput of 278-730 Mevents/s and system energy efficiency of 158 TOP/S/W.

4.2 Convolution Kernel: Stochastic Compute-In-Memory (SCIM) Macro

4.2.1 SCIM Macro Architecture

The Stochastic Computing-In-Memory (SCIM) macro accelerates the convolution operation unrolled as matrix-matrix multiplication between object tracking filters and input sliding windows Fig.4.2. The SCIM unit, consisting of a 6 memory cells, in-situ SNG, and a MAC array of 32 MACs, is the smallest compute primitive within the macro, and corresponds to a single weight. The SRAM-based memory cell array stores a 6-bit binary weight. We use 6-bit weights since the simulation has indicated it is sufficient for our target application. The in-situ Stochastic Number Generator (SNG) converts the weight from binary to a stochastic number using a 5-bit random

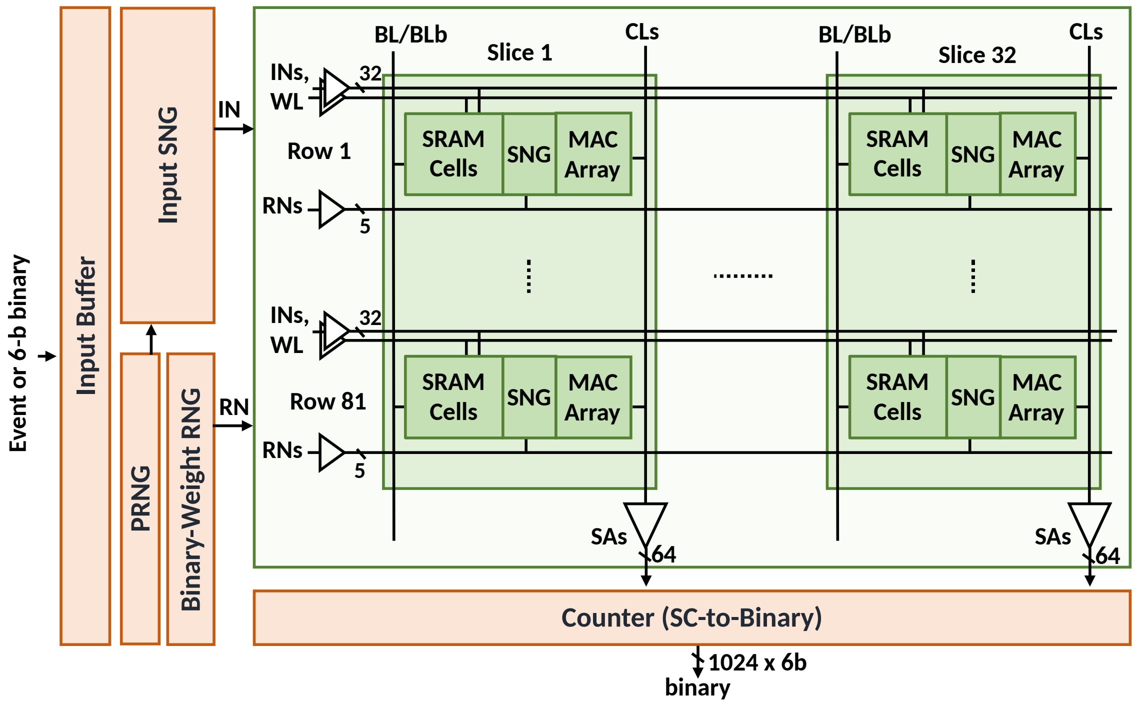


Figure 4.2: Stochastic Compute-In-Memory (SCIM) macro architecture.

number: $RN_1 - RN_5$. The random numbers are generated from the pseudo-random number generators, described in Section 4.2.3, next to the memory and shared across the SCIM units on the same row. The SNG output within each SCIM unit is multiplied with 32 inputs $(IN_1 - IN_{32})$ in an SC MAC array and accumulated with other SCIM units in the same SCIM slice. Each SCIM slice stores an unrolled, 9×9 filter and computes 32 outputs, using nine unrolled input half-rows (9x36, including overlap). We process ROI rows split in half vertically, due to the physical limitations of the macro layout. There are 32 SCIM slices within the macro that share the same inputs, each implementing a different filter. The outputs of the SCIM MAC are streams of 1-bit values and are converted to digital bits by the inverter at each Compute Line (CL). A replica slice is built to track the process, voltage, and temperature (PVT) variation of the circuit and generate the timing signals for the output sampling. The following sections will provide a detailed description of the in-situ SNG, near-memory random number generator, and the in-memory SC MAC units.

4.2.2 In-Situ Stochastic Number Generator

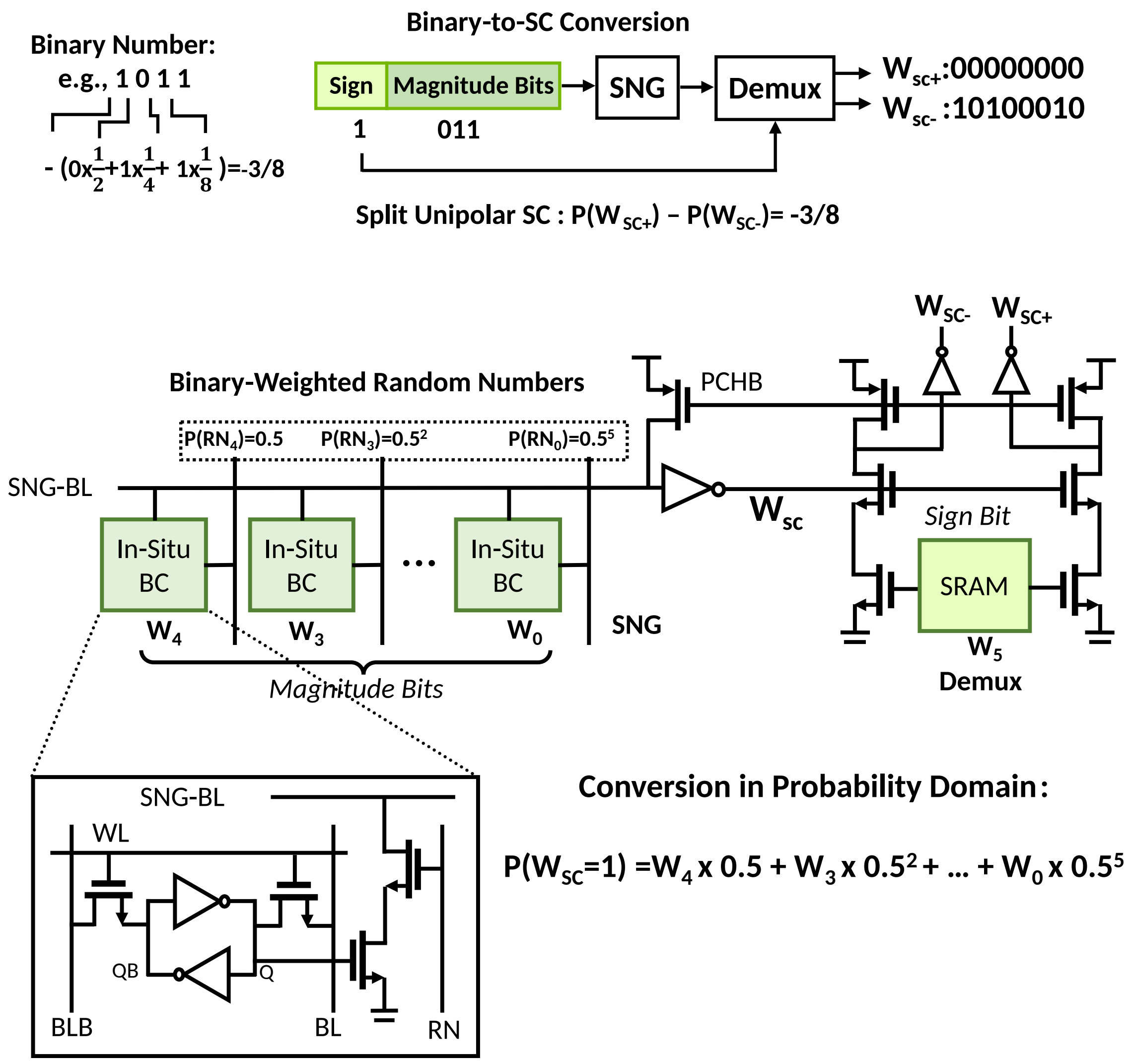
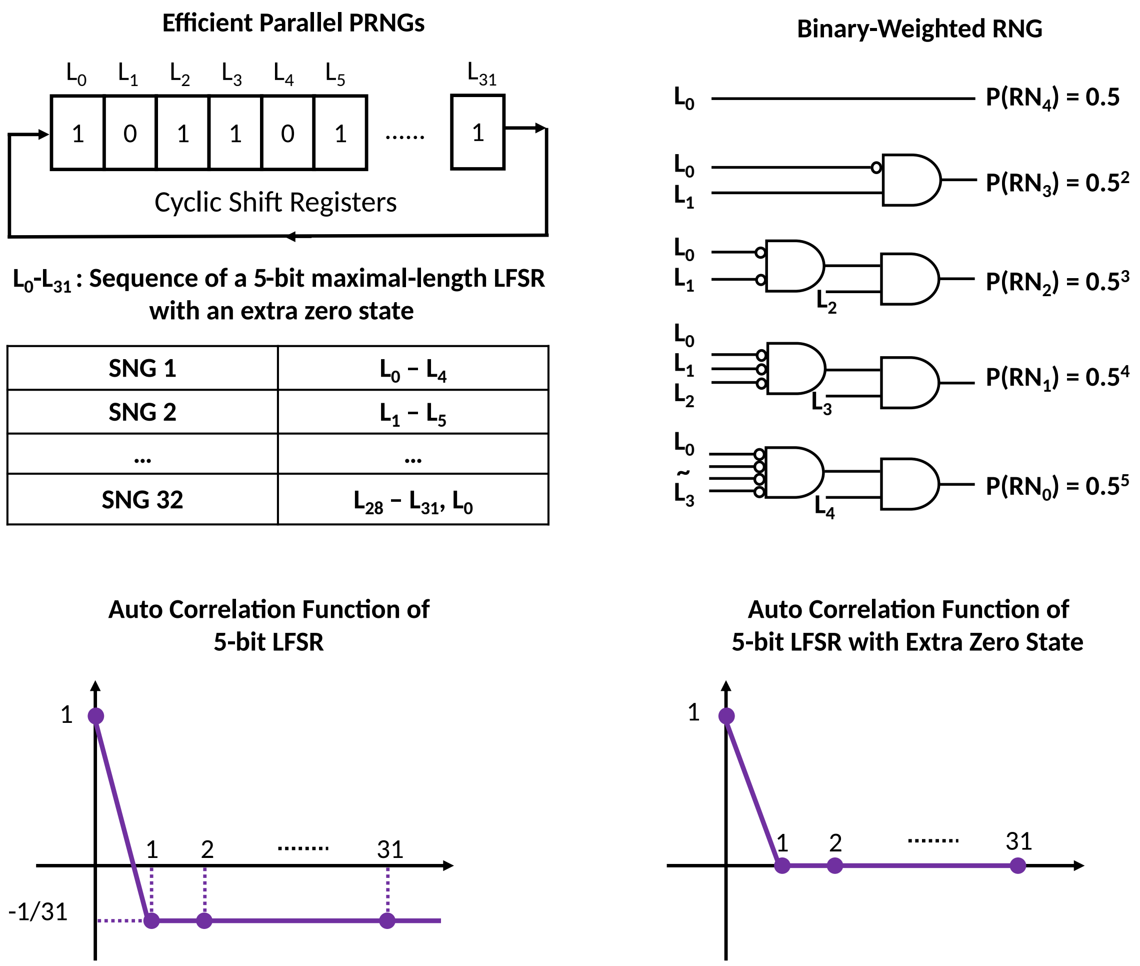


Figure 4.3: (Top) Binary-to-Stochastic conversion concept. (Bottom) In-Situ SNG's circuit diagram.

The compact Stochastic Number Generator (SNG) can be embedded inside the memory to achieve in-situ binary-to-stochastic number to allow storing binary numbers. Prior bit-parallel stochastic-CIM macro stores 2^N stochastic number in the memory for N-bit precision. By embedding in-situ SNGs, only N number of cells are required. Using the in-situ SNG also means that the implementation is now agnostic to the stream length, which was not the case for previous SCIM work where all the stochastic bits are computed in parallel [30]. Different stream length can be supported by varying the number of conversion cycles and the hardware remains the same.

The in-situ SNG supports conversion of signed numbers to split-unipolar stochastic representations. A block diagram of the in-situ SNG is shown in Fig.4.3. The magnitude bits of the binary number are first converted by the SNG circuit into an unsigned unipolar stochastic bit stream. The sign bit controls a demuliplexer that passes the unipolar bit stream to either W_{sc+} or W_{sc-}. W_{sc+} and W_{sc-} follows the convention of split-unipolar representation where the probability difference between two streams represents the bianry number [46]. The circuit implementation of the in-situ SNG is shown in Fig.4.3 (bottom). The SRAM memory cells store 6-bit binary weight numbers. The magnitude bits are selected by random numbers with binary-weighted probabilities. For example, the MSB(5^{th}) is selected by random numbers with a probability of 0.5, the 4^{th} bit is selected with a probability of 0.5^2, and the LSB is selected with a probability of 0.5^5. Each bit cell has two extra cascaded NMOS transistors beside the 6T SRAM cell to perform an AND operation between the stored binary bit and the random number. The output of AND logic in each cell is connected by a local bitline (SNG-BL), which performs a wired-OR operation to add the probability represented by each bit. An inverter amplifies in-situ SNG's local bit line and inverts the signal to maintain the correct logic. The sign bit stored in a memory cell controls passing the unipolar stream W_{sc} to either W_{sc+} or W_{sc-}. The unselected output is grounded to zero. An inverter is added at each output to generate the correct logic state and drive the next stage. The operation of the in-situ SNG is similar to domino logic [63]. The 'PCHB' signal controls the PMOS transistors that precharge the output before the operation starts and all the 'RNs' are grounded to avoid short circuit path through NMOS and PMOS transistors. Once the operation starts, precharge PMOS transistors are turned off and 'RNs' are applied to the in-situ SNG.



SNG 1	$L_0 - L_4$
SNG 2	$L_1 - L_5$
...	...
SNG 32	$L_{28} - L_{31}, L_0$

Figure 4.4: (Top) An efficient pseudo random number generator circuit; Binary-weighted RNG. (Bottom) Auto Correlation function of 5-bit LFSR and 5-bit LFSR with an extra zero state.

4.2.3 Efficient Near Memory Pseudo Random Number Generator (PRNG)

A shift register based pseudo-random number generation method and binary-weighted encoding are used to reduce the cost of the random number generation. Maximal-length Linear Feedback Shift Registers (LFSR) with the order of characteristic polynomial that matches the bit width of the binary number is shown to be able to accurately convert binary numbers to a stochastic [46] . The N^{th} order maximal length sequence (m-sequence) has a unique property that every N-bit combination except zero only appears once and has a period of $2^N - 1$. In this work, the m-sequence plus an additional zero state is stored in a cyclic shift register that rotates when clock is switching. The autocorrelation function of the m-sequence is $-1/31$. With the extra zero state,

Cost of PRNG (as number of flip flops)

Naive Implementation	Proposed
$N * (2^N - 1)$	2^N

Table 4.1: Cost of generating N^{th} order maximal length LFSRs with all possible seeds, evaluated as number of flip flops required.

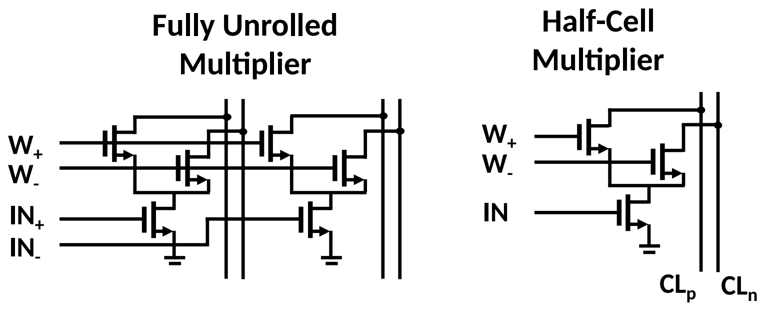
the autocorrelation function of the pseudo random number sequence becomes zero. Different taps of the shift registers are used by different SNGs. For example, SNG1 uses $\{L_0 - L_4\}$, SNG2 uses $\{L_1 - L_5\}$ ... SNG32 uses $\{L_{31}, L_0 - L_3\}$. Due to the zero autocorrelation of the pseudo-random number sequence, different SNG's random number does not have any correlation.

Storing the entire pseudo random number sequence has significant benefits of energy and area compared to fully-unrolled LFSRs. Consider a naive implementation of an array of N^{th} order LFSR that has all possible seeds, which result to $2^N - 1$ LFSRs. Each of LFSR requires N flip flops. The proposed method only requires 2^N flip flops and generate an extra zeros state, Table. 4.1. Both the energy consumption and area are reduce by N times using the proposed method.

The in-situ SNGs require random numbers with binary-weighted probabilities: $0.5, 0.5^2, 0.5^3, ...$ A random number generator converts outputs of the PRNG into binary-weighted probabilities and guarantees one-hot outputs to reduce switching activities. As an example shown in Fig.4.4, inputs $L_0 - L_4$ are generated from the PRNG. RN_4 is high when L_0 = 1, which happens 1/2 of the time in a period of the modified m-sequence. RN_3 is high when L_0 = 0 and L_1 = 1, which happens exactly 1/4 of the time, and so on for the rest of the output: RN_2, RN_1, RN_0 have probabilities of 1/8, 1/16 and 1/32. The circuit also guarantees that only one RN output is high in each step. The inversion plus AND gate makes sure that if a given RN_k is high, all the lower order $RN_j, j < k$ are zero. For example, if $RN_4 = 1$, then the rest of the RNs are all zero. The one-hot encoding reduces the switching activities of the random numbers, and, therefore, coupling noise.

4.2.4 In-Memory SC MAC Array

a) Multiplier Cell Supporting Signed and Unsigned Inputs



	IN	CL_p	CL_n
+ Phase	IN_+	$W_+ \times IN_+$	$W_- \times IN_+$
- Phase	IN_-	$W_+ \times IN_-$	$W_- \times IN_-$

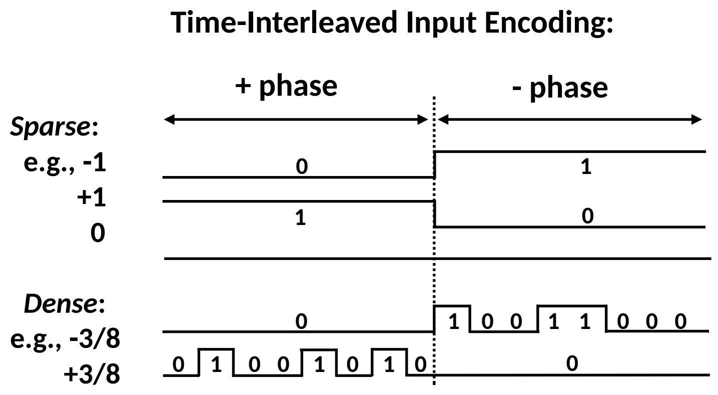


Figure 4.5: (Top) Implementation of fully unrolled SC multiplier and time-interleaved SC multiplier. (Bottom) Time-interleaved encoding of input for both event and dense numbers.

Previous work has demonstrated an in-memory SC MAC for unsigned inputs in neural network applications where activations are positive numbers after nonlinearity functions [30]. To support both unsigned and signed inputs, a half-cell multiplier is used to maintain area density and interleave split-unipolar streams in time. To support signed SC multiplication, one can fully unroll the cross product between input and weight's two bit streams as four 1-bit SC multipliers $(+/+, +/-, -/+,$ and $-/-)$, shown in Fig.4.5(top). If input is an unsigned number, however, half of the multiplier cell is not utilized. The larger multiplier cell leads to lower energy efficiency for both unsigned and signed number. Instead, the half-cell multiplier only uses half the area and the signed multiplication is supported by interleaving input's split-unipolar streams in time.

Scheduling of time-interleaved input bit streams is shown in Fig.4.5 (bottom). The positive/negative bit stream of the inputs are applied in $+/-$ phases. In $+$ phase, CL_p is the dot product between weight's positive stream and input's positive stream: $W_+ \times IN_+$, which has a positive sign. CL_n is the dot product between weight's negative stream and input's positive stream: $W_- \times IN_+$, which has a negative sign. When input is applied in the $-$ phase. The sign of CL_p and CL_n are flipped: CL_p has a negative sign and CL_n has a positive sign, summarized in Fig.4.5 (middle). Counters will accumulate CL_p and CL_n's output with their corresponding sign. The macro supports both event and fixed-point inputs. Event inputs are ternary numbers $-1/0/+1$, therefore do not need stochastic number generator and remain constant within the phase. Input streams for each of $-1/0/+1$ are shown in Fig.4.5. For fixed point inputs, SNGs convert them to split-unipolar stochastic streams and positive/negative streams are applied in $+/-$ phases.

SCIM Slice and Layout Optimization for Energy Efficiency

Analog Compute-In-Memory macro requires one analog-to-digital converter (ADC) for every dot-product column. Increasing the number of dot-product column sharing the same memory bit cells causes large area and energy overhead from ADCs. Since SC uses digital logic that does not require ADCs, the SCIM macro can embed 32 SC MACs for each weight and the sense amplifiers only account for 10% of macro area. Fig.4.6 shows the structure of one SCIM slice. Each row of the slice is one SCIM unit including 6 SRAM cells storing the binary weight, an in-situ SNG and an array of 32 SC MACs. 32 input lines are routed on top of SCIM units in each row. The computation happens in MAC array where 32 inputs multiply with the shared in-situ SNG's outputs: W_{sc+} and W_{sc-}. The SCIM slice has 81 rows; therefore one slice computes multiplication between an 81x1 weight vector and 81x32 input matrix.

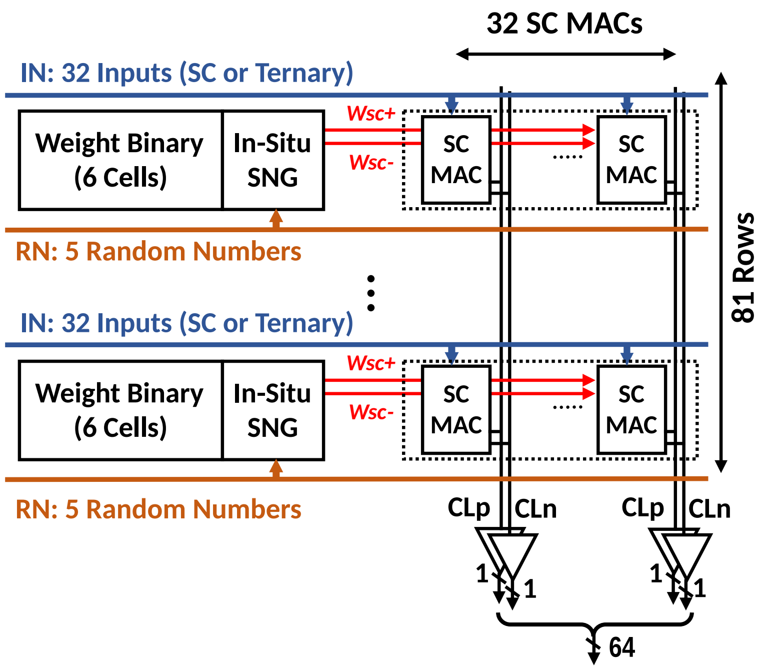


Figure 4.6: One SCIM slice.

The number of SC MACs sharing a weight, denoted as weight reuse factor, is chosen to optimize both energy efficiency and throughput per area. Increasing the reuse factor can amortize the cost of in-situ SNG and increase SCIM macro's throughput. The metal wiring of input or compute line can be routed on top of the memory cells without causing extra area penalty for a small reuse factor. When the reuse factor becomes too large, however, the energy efficiency starts to drop due to significant increase of area. Fig.4.7 shows the performance metric for different weight reuse factors. The metric achieves the optimal point for the reuse factor of 32. Larger reuse factors can achieve similar performance, but they lead to significant layout complexity, therefore we pick a reuse factor of 32.

The aspect ratio of the SCIM unit's layout impacts the energy consumption of SCIM unit, which is dominated by switching the parasitic capcitance on the input and output lines. Different layout options are compared to optimize energy efficiency. An SCIM unit includes six SRAM cells, an in-situ SNG and 32 SC MAC units. Three layout options are considered and the energy consumption of each option is estimated: 1) 1×6, SRAM cells are placed as a 1×6 array and

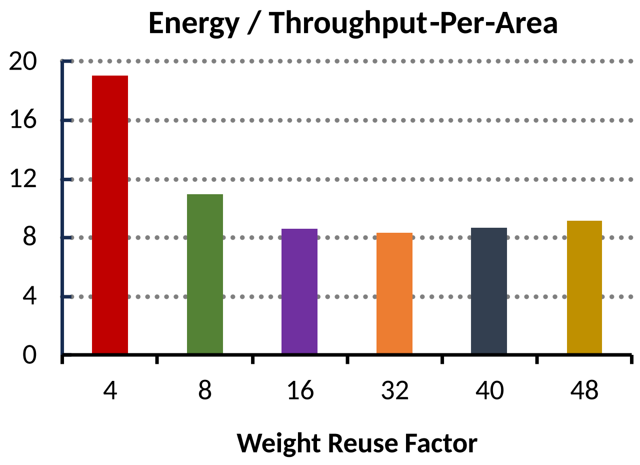


Figure 4.7: Energy divided by throughput per area metric for different weight reuse factors.

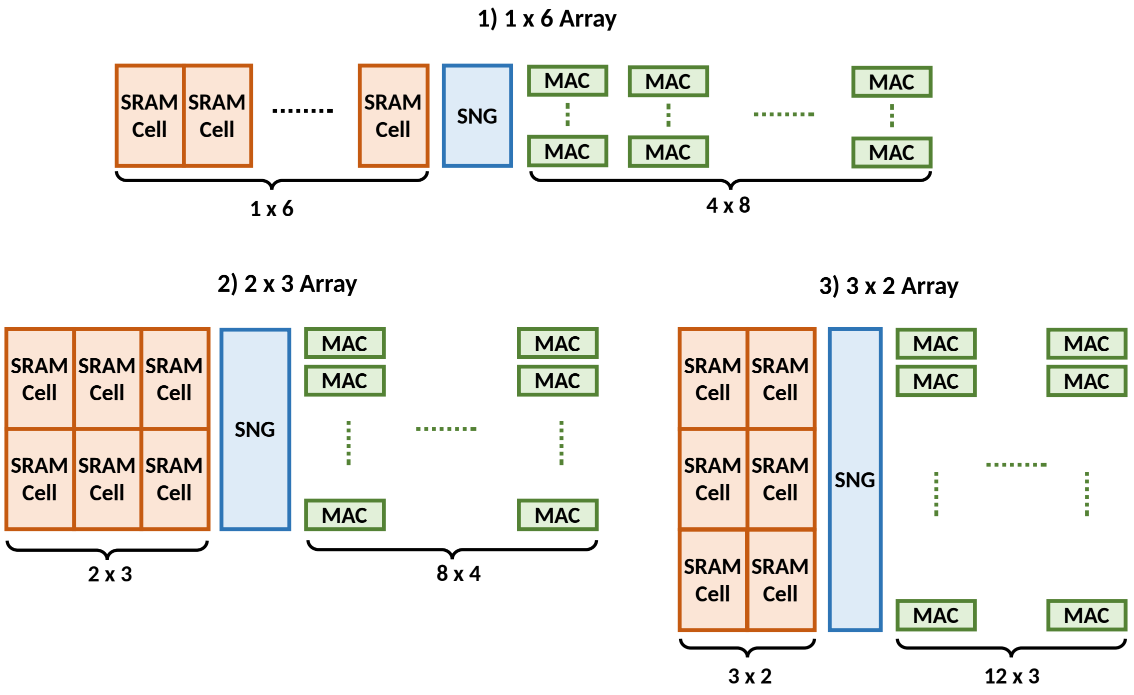


Figure 4.8: Three different options of laying out the SCIM unit.

MAC units are placed as a 4×8 array. 2) 2×3, SRAM cells are placed as a 2×3 array and MAC units are placed as 8×4 array. 3) 3×2, SRAM cells are placed as a 3×2 array and MAC units are placed as 12×3 array of 36 MAC units. In all three options, the height of the SRAM cells match the SC MAC array. The dimension of three options and the parasitic capacitance of the routing are estimated and verified by layout extraction. A switching activity of 50% is assumed for both input and compute line. The energy consumption of three options are shown in Fig.4.9. The 1x6 option consumes the most energy due the long distance of the input lines. The 2x3 and 3x2 options consume similar energy, but 2x3 option achieves the best energy efficiency. The SCIM unit with 2x3 array configuration is laid out in 12nm technology using the standard DRC rules, as shown in Fig.4.10. It occupies an area of 3.2μm x 7.5μm. The SRAM cells use 1/4 of the area and 32 SC MAC units dominate rest of the area. 32 input lines are routed in horizontal direction above the SCIM unit and 64 compute lines are rounted in vertical direction above the MAC array.

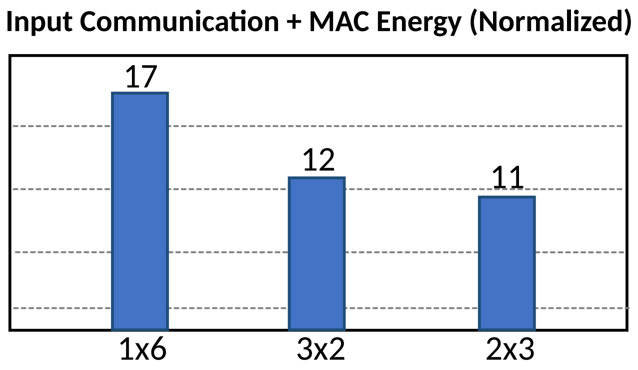


Figure 4.9: Energy of input communication and MAC operation for three different options.

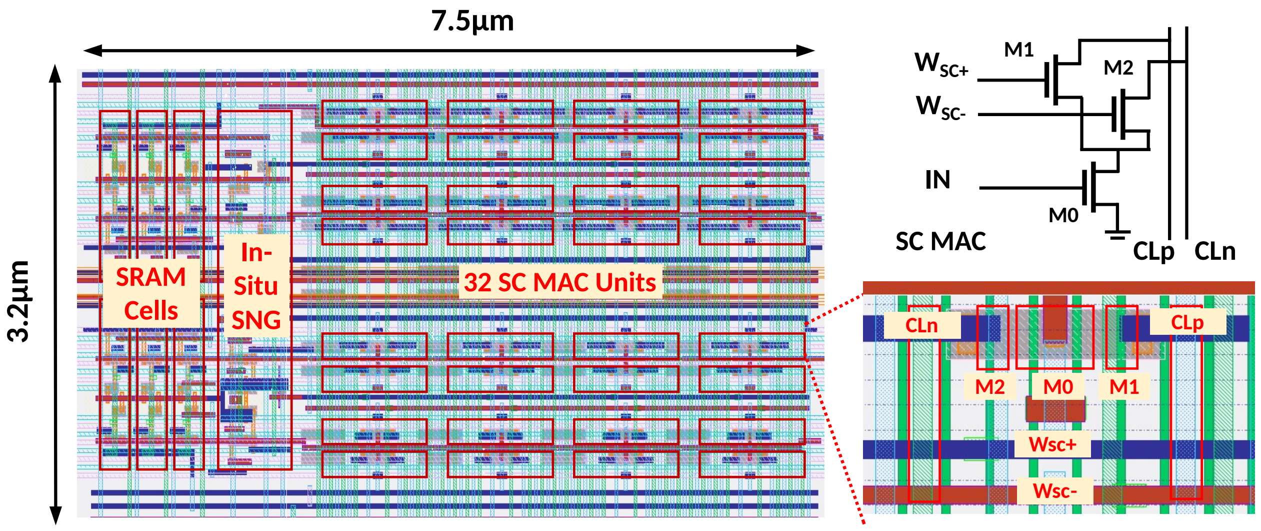


Figure 4.10: Layout of SCIM unit with 2x3 option in 12nm technology.

4.2.5 High Speed Sensing Circuitry with PVT Tracking

The event camera accelerator demands fast and low-power computation in SCIM macros. A high-speed sensing circuit is designed to guarantee fast and robust operations across Process, Voltage and Temperature (PVT) variations using a PVT tracking slice, shown in Fig.4.11. Each Multiplier cell performs an 1-bit multiplication such the ON/OFF of the cascaded NMOS circuit represents 1/0. The in-memory wired-OR accumulation is equivalent to a multi-input OR gate: if any of the inputs is ’true’, the output will be ’true’. The sensing circuit, therefore, needs to distinguish between no multiplier cell and at least one multiplier cell is conducting between the compute line and ground. The logic operation follows the domino logic such that the compute line is precharged before the computation. The sense amplifier can wait until the compute line to fully discharge to ground and then register the output, but the large parasitic capacitance on the compute line requires a long time to discharge in the worst case. To save sensing time and energy, the sense amplifier compares a voltage threshold when the compute line is only discharged for a fraction of VDD.

Transistor’s local mismatch and global PVT variation can lead to decision errors. A replica slice tracks the PVT variation and generates asynchronous timing signals with programmable sensing margin to account for mismatches. The circuit diagram of the sense amplifier is shown in Fig.4.12. The inverter has four programmable thresholds. Three transmission gates are cascaded between drains of nMOS and pMOS transistors in a normal inverter as a resistor ladder. Gates of the transmission gate are shorted to input voltage. Voltages at the four terminals of the transmission gates have the similar input-output transfer function of inverters, each of them, however, has a different switching threshold. The transfer function between input and four outputs are shown in Fig.4.12. The multiple thresholds are used to provide programmable sensing margins for read operation.

The PVT tracking slice replicates a normal computation slice and generates timing signal with extra sensing margin to account for local mismatch. 32 computing slices’ sense amplifiers (SA)

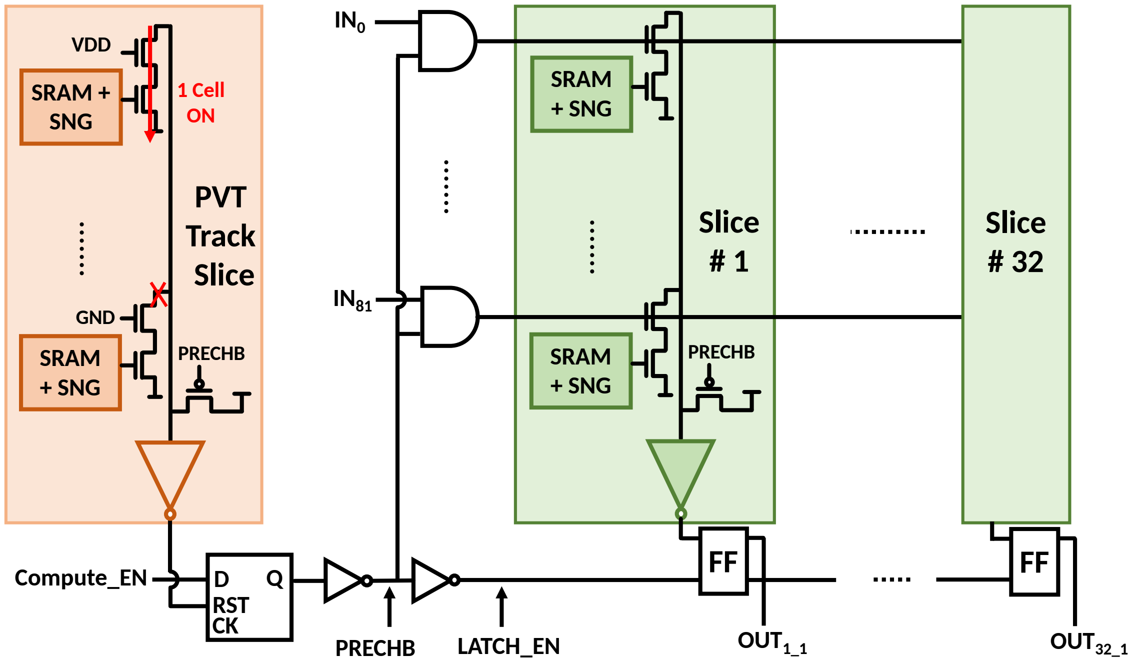
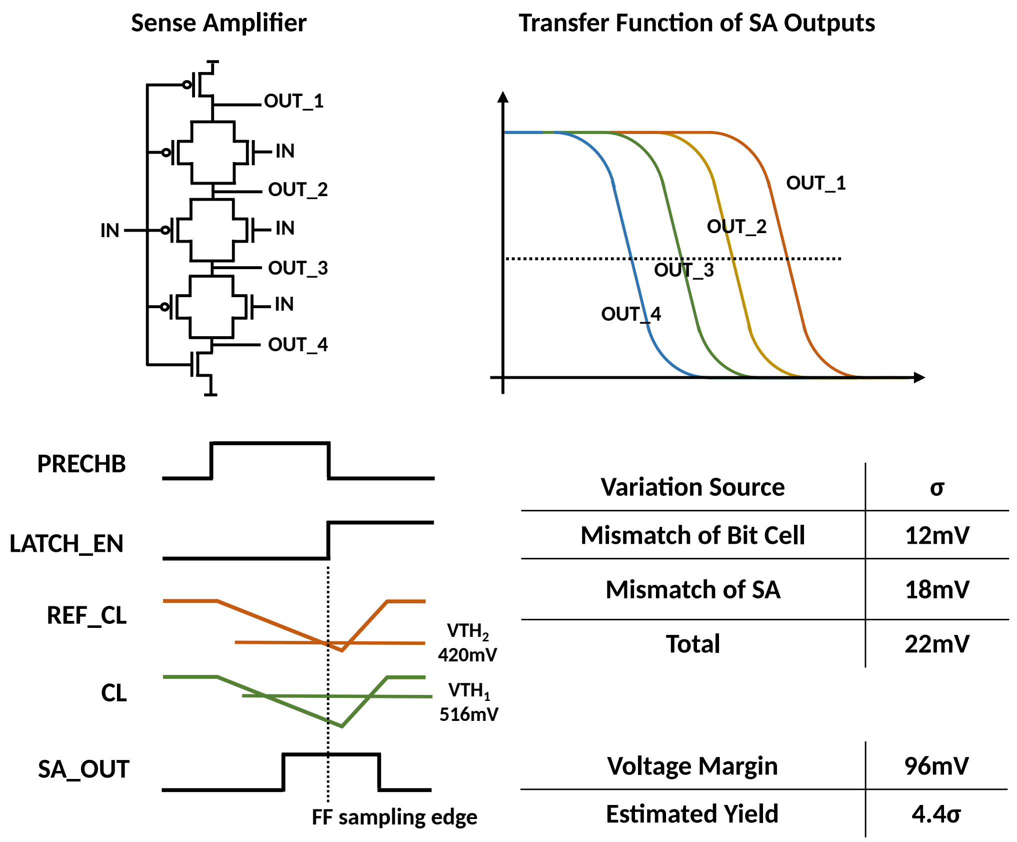


Figure 4.11: Read scheme of the SCIM macro.

are programmed with the highest threshold $Vth_1 = 516mV$. When one or more cells discharge the compute line to a voltage below Vth_1, the SA will switch to high. The tracking slice allows only one cell to discharge the compute line and generates a clock signal, **LATCH_EN**, if its compute line is discharged below the sense amplifier's threshold. **LATCH_EN** is shared across all the output flip flops to latch the sense amplifiers' outputs. The PVT tracking slice's sense amplifier is programmed with a lower threshold voltage to allow compute lines to discharge for a longer time before flip flop is latched, and, therefore has more voltage margin against local mismatches. The dominant sources of the mismatch come from multiplier cell's current and sense amplifier's threshold, summarized in 4.12. The total voltage variation has a standard variation of 22mV. If the Vth_2 is set for SA in tracking slice, there is a sensing margin of 96mV, which leads a yield of 4.4σ. The replica slice's sense amplifier can be programmed with lower threshold to allow more margin.



Variation Source	σ
Mismatch of Bit Cell	12mV
Mismatch of SA	18mV
Total	22mV

Voltage Margin	96mV
Estimated Yield	4.4σ

Figure 4.12: Sense amplifier with programmable thresholds; Sensing yield when local mismatches are considered.

4.3 Filter Based Object Tracking Pipeline

Event camera removes texture-rich static background and makes the filter-based algorithm highly effective for object detection and tracking applications [64]. Fig.4.13 shows our filter-based object tracking pipeline. The event inputs are accumulated in uniform time steps and quantized to ternary polarity format (-1/-/+1) as pseudo event frames to reduce the redundancy. The quantization can be also skipped and the event frames are processed in 6-bit precision. We use 32 convolution Gabor filters combining 8 different directions and 4 speeds, shown in Fig.4.14. Other tracking filters matching the specific shape of the object can also be used if prior knowledge is given. The object tracking pipeline includes convolution tracking filter, clustering, bounding box generation and Norfair multi-object tracker. The computation cost of each section of the pipeline is estimated.

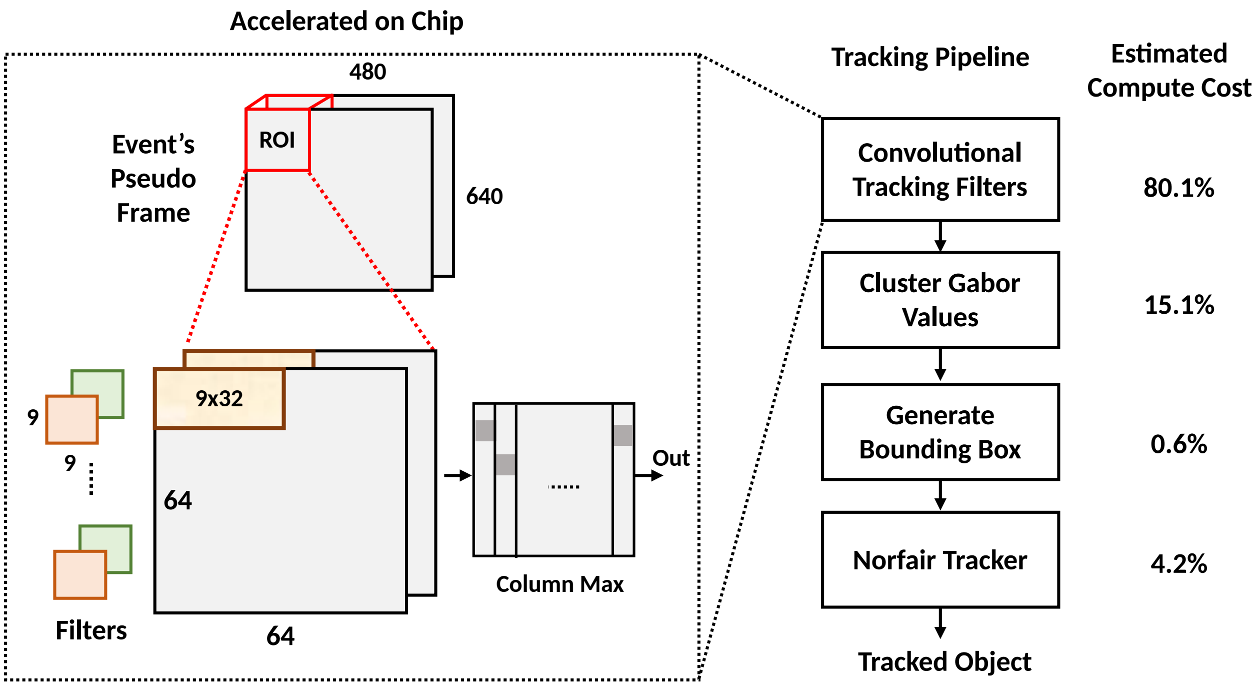


Figure 4.13: Object tracking pipeline.

Convolution tracking filters account for 80.1% of the cost, which is the most dominant part.

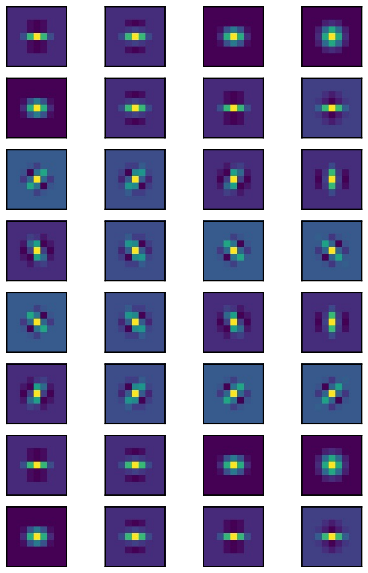


Figure 4.14: Gabor filters.

Convolution tracking filters are computed on the SCIM accelerator and the rest of tracking pipeline is implemented in Python. A Region-of-Interest (ROI) based processing is used to support

ROI proposal algorithms. The pseudo event frame is 640 pixel $\times$ 480 pixel. The frame is divided into a 8 $\times$ 11 ROI grid. Each ROI is 64 pixel $\times$ 64 pixel. Region proposal can predict the most likely ROI containing the object and only the predicted ROI is processed by the convolution filters. The SCIM accelerator takes two pseudo event frames as input. 32 of 9x9x2 convolution filters are stored in two SCIM macros. Each macro stores one time channel of the filters. The accelerator computes 32 of 9x9x2 convolution sliding windows in x direction. The rest of ROI is serially loaded to SCIM macros to complete convolution. The filter outputs are passed to a column max function that takes the maxima value of each column in ROI to save the output bandwidth.

4.4 Accelerator Architecture

Figure 4.15 shows the overall architecture of accelerator. The control logic is implemented as a finite-state machine (FSM) controlled through a set of programmable registers. The communication between the chip and external device is through the JTAG I/O interface. The accelerator is designed to process a single 64 $\times$ 64 ROI at a time with up to two time channels. Based on that, the architecture is organized into two columns, each consisting of an input SRAM, staging buffer, and a SCIM macro.

Each input SRAM is provisioned to buffer one time channel of a ROI and has a width of 64 to match ROI's width. The SRAM uses a bit-transpose layout to support different precision. For example, the 1st row stores the MSB, the 2nd row store (MSB-1) bit, and so on. For the event inputs with ternary precision, two bits are stored in two rows. For a dense frame with 6-bit precision, each number is stored across six rows.

Values from input SRAM are first read into staging buffers. The bit-transpose format is decoded after every N SRAM read cycles for an N-bit inputs. The staging buffer can optionally rotated to pass a certain time channel to a different SCIM macro, supporting convolution across time domain.

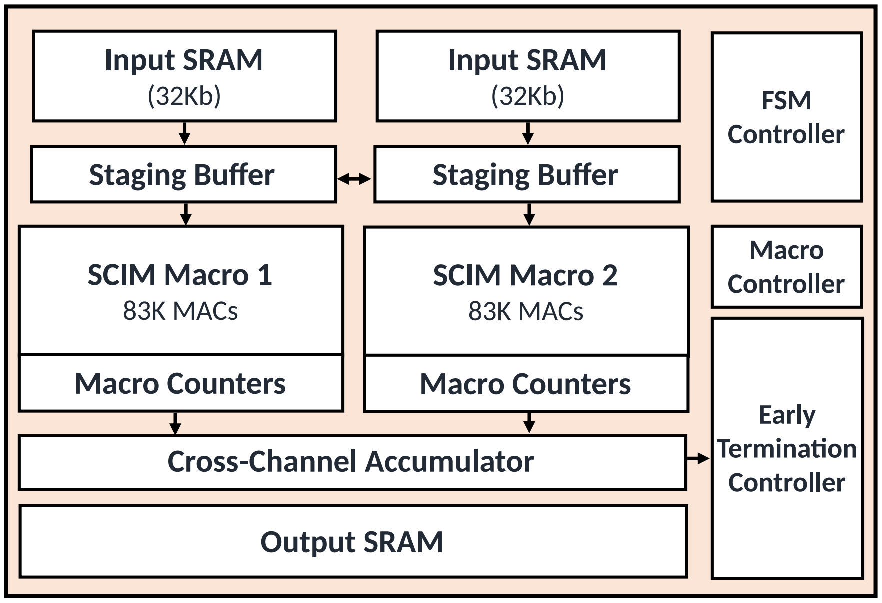


Figure 4.15: Architecture of the accelerator.

Since, as described in the previous Sections, the SCIM macro can only process half of the row at a time, staging buffers are provisioned for 36 2-bit values. Within each SCIM macro, the input values are written to the input buffers, where they are used to generate SC streams when the computation starts. Each bank has nine input buffers, which collectively hold nine rows of 36 values, making it possible to unroll to convolutional sliding windows spatially: the 9×36 inputs are unrolled to 32 sliding windows of 9×9 with a stride of one.

The weights are pre-loaded in the SCIM, and their streams are generated in situ, as described in the previous Section 4.2.2. Within each macro, a sliding 9x9 convolution is performed across nine input rows, generating 32 outputs, for 32 filters, for a total of 1024 outputs per macro. Outputs of each compute line, are fed into counters. After the computation is finished, counter outputs from different SCIM macros are sent to the global accumulator to implement a 9x9x2 filter.

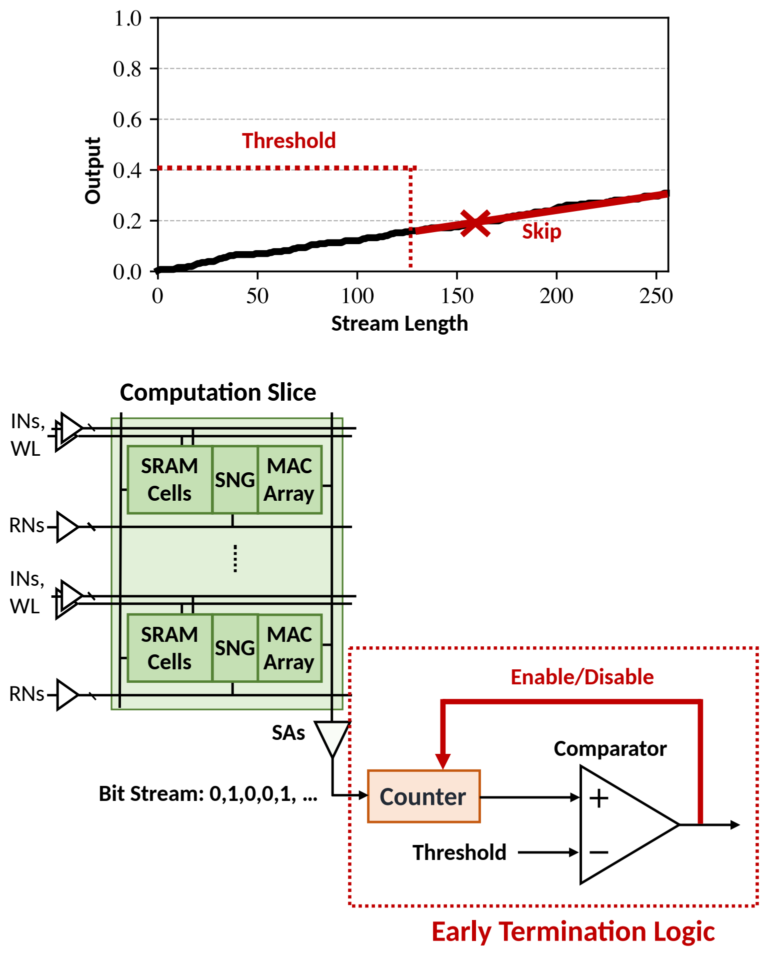


Figure 4.16: (Top) Concept of early termination; (Bottom) Early termination logic circuit.

4.4.1 Early Termination

SCIM macro's energy consumption is significantly reduced when switching activities are low, but the energy of synchronous logic such as counter and clock buffers remains constant. Three levels of termination logic are implemented to save energy at high sparsity level. The concept of early termination is shown in Fig.4.16 (top). The proposed SCIM accelerator computes bit-serially and counters accumulate the output bit stream in 64 cycles for 6b compute. Since each bit in the stochastic bit stream has an equal weight, counter's result grows linearly with bit stream length in average and the variation becomes smaller, which is known as stochastic computing's progressive precision [22]. The partial results of the counter are compared with a threshold. If it is smaller than the threshold, it shows a low correlation with the filter and the object is unlikely to show up in the convolution window. The remaining stream can be skipped to save both energy and time. The circuit implementation is shown in Fig.4.16 (bottom). A comparator compares counter output with the threshold and the result is used to turn off the counter.

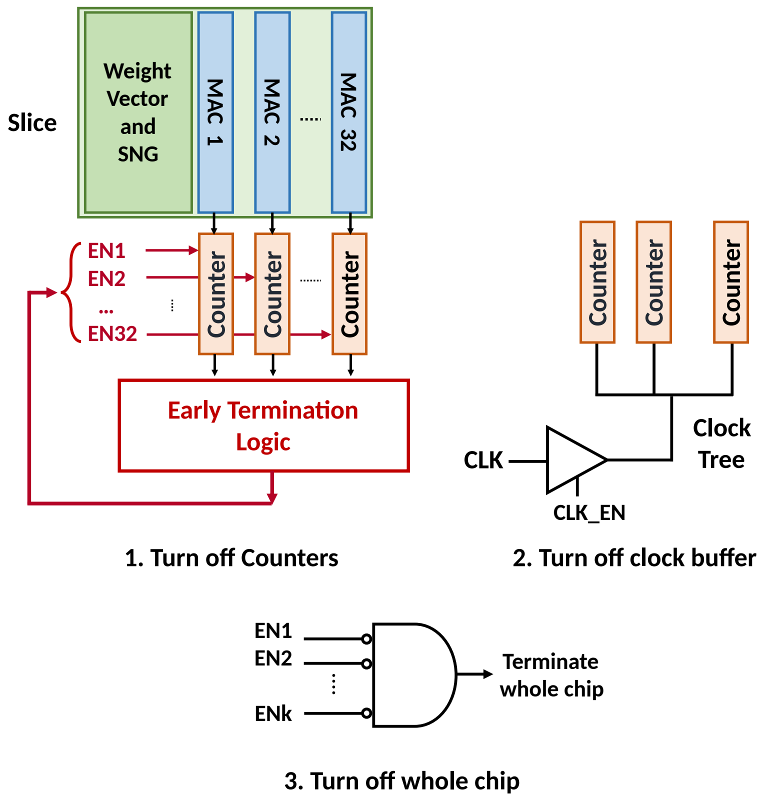


Figure 4.17: Three levels of early termination.

Three levels of early termination are shown in Fig.4.17. The first level can turn off individual counter in a slice. There are 1024 counters in an SCIM macro and each of them can be turned off individually. The second level turns off clock tree buffer. A group of counters share the same clock tree and applying gating signals at the clock buffer can cut down clock switching energy. The third level terminates all operation on the chip and move on to compute a new set of inputs. The criteria is met when all the counter's enable signals are low. The first and second categories only save energy, while the third category saves both energy and time. The termination criteria is checked for every 8 cycles to avoid energy overhead from the early termination's control logic. An example timing diagram of the early termination is shown in Fig.4.18. Input uses a time-interleaved encoding, therefore half of the 8 cycles computes positive input stream and the other half computes negative input stream. Criteria of three levels of termination are checked when **ET Check** signal is high. Counters that meet the level 1 early termination's condition will be disabled for the rest of the chip operation. Clock buffers are gated by level 2 early termination if groups of counters are disabled. Since the SC stream length is 64 for 6b compute, the early termination is checked for 8 times. If the level 3 early termination is triggered, it stops the chip's operation on the current ROI and new ROI needs to be loaded from the input SRAM.

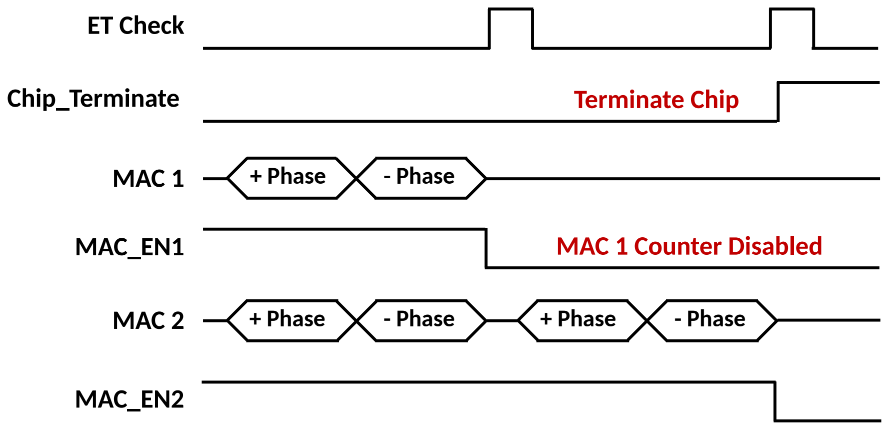


Figure 4.18: Timing diagram of early termination.

4.5 Results and Evaluation

The prototype chip is fabricated in GF 12nm LP technology. The testing setup and die photo are shown in Fig.4.19. The core area of the chip is $1mm \times 0.5mm$. A current mode logic differential-input and single-ended-output clock buffer is on chip to support high-frequency clock sent from external. Arduino micro-controller is used to interface with the chip's JTAG controller and PC.

The operating clock frequency of the chip at different supply voltages are characterized, shown in Fig.4.20. The system is working at frequency range of 600MHz-850MHz under supply voltages

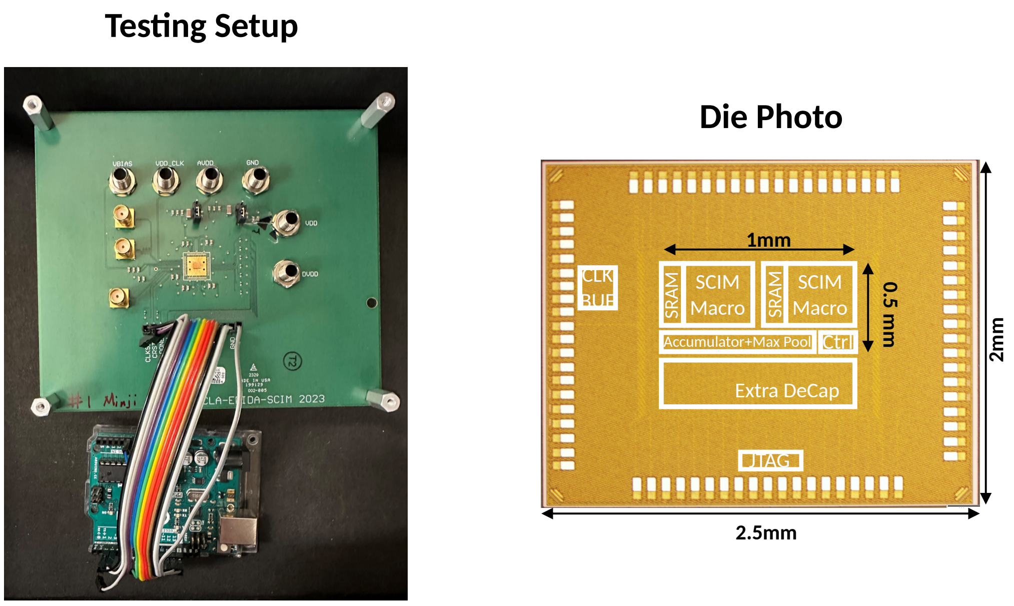


Figure 4.19: Testing setup and die photo.

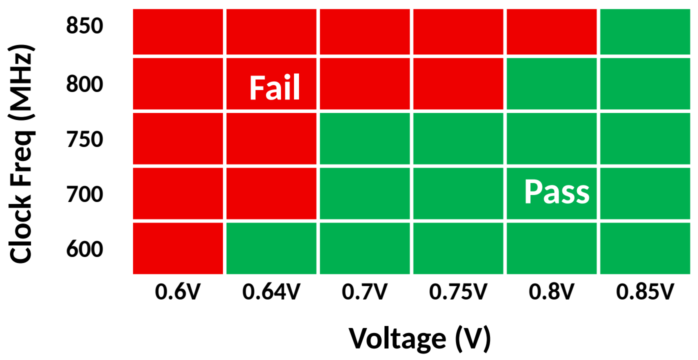


Figure 4.20: Clock frequency vs. supply voltages .

between 0.64 and 0.85V. The testing data set is acquired by a static event camera capturing a flying object. The events are accumulated every 2m second and quantized to ternary numbers as pseudo-event frames: positive numbers quantized to +1, negative numbers quantized to -1 and zero remains as 0. An example dense frame and event video of 250 frames of 480 pixel $\times$ 640 pixel are shown in Fig.4.21. The event frame is divided into 8 $\times$ 11 grid of ROIs. The sparsity of each ROI is measured and shown in Fig.4.21 (right). The ROIs without a moving object has a sparsity above 99%, while the ROI with the object has a sparsity around 98%.

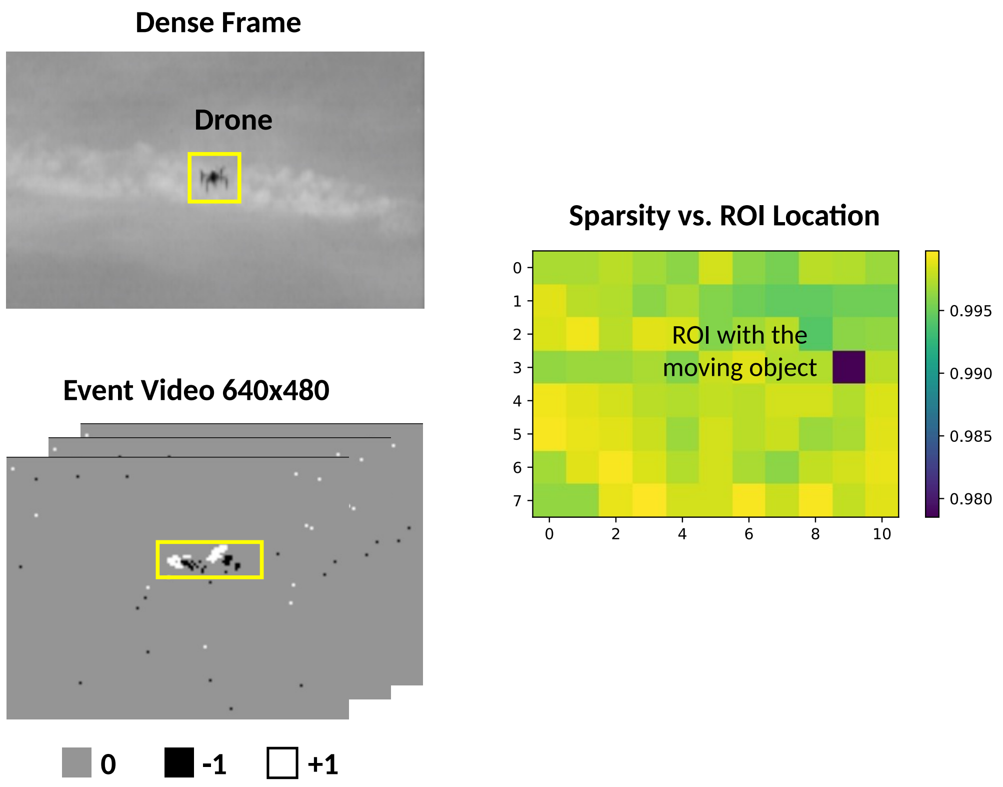


Figure 4.21: Dense frame and event data set; Sparsity of ROIs at different locations in an example frame .

4.5.1 Analysis of SC Errors

The SC's computation error compared to fixed point ground truth is very small due to high sparsity. In-situ stochastic number generation of weights using binary-weighted random numbers is accurate [45]. Multiplication is also accurate since input is a constant value within $+$ or $-$ phase so that the AND-based multiplication does not introduce any error. The OR-accumulation approximates the accurate summation of inputs when they are small, but suffers from nonlinearity when the sum of inputs is large [65]. Since the event images have very high sparsity (99%), the average sum of inputs is small and leads to very accurate accumulation using OR logic. The SC computation's results vs. fixed-point ground truth in one ROI is shown in Fig.4.22 (Top). Results of both show close match between each other. The average error between SC computation and the expected ground truth value is plotted in Fig.4.22 (Bottom). The errors are below one LSB with only few

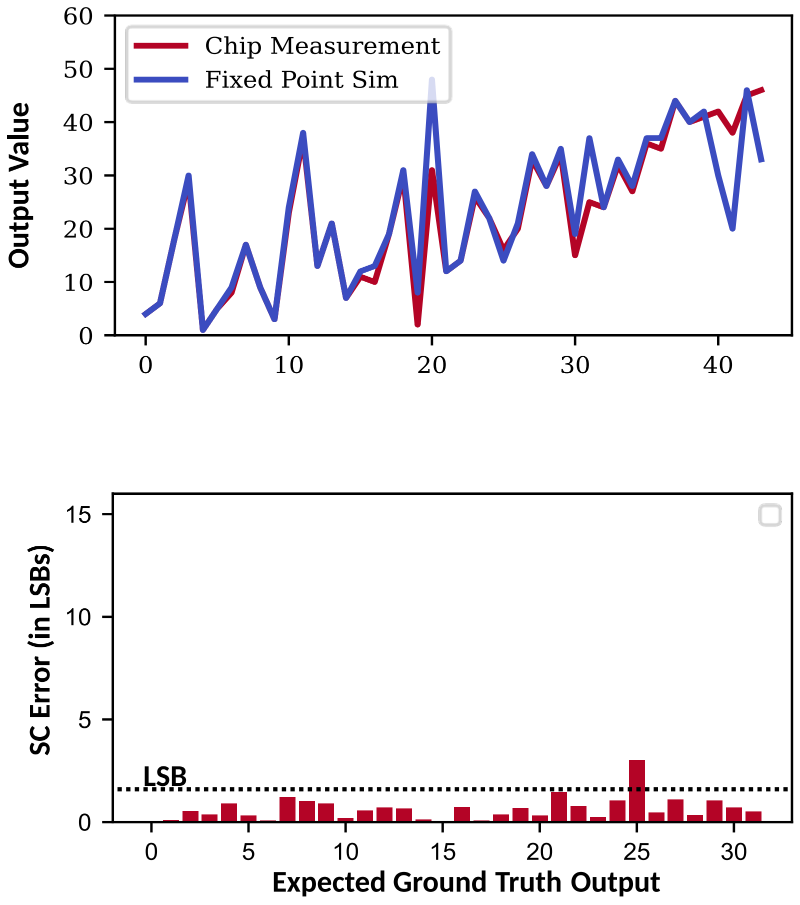


Figure 4.22: (Top) Filter output SC vs Fixed-point ground truth. (Bottom) SC error vs expected ground truth result.

exceptions, which do not impact the tracking results.

4.5.2 Object Tracking Demonstration

The object tracking is demonstrated using the pipeline describe in section 4.3. The convolution filters are accelerated on the SCIM chip and the rest of the pipeline is performed in python. The tracked object's location shows close match with ground truth, as shown in Fig.4.23 (top left and right). Without early termination, the tracked location has an average error distance of less than 5 pixels. When early termination is turned on, computation can be skipped to save energy. The higher the early termination (ET) thresholds, more computation is skipped, but tracking accuracy is degraded. The average error distance increases to 10 pixels when ET threshold is larger than 0.5. The object tracking performance is also evaluated by Higher Order Tracking Accuracy (HOTA) metric [66]. HOTA is an accuracy metric for Multi-Object Tracking (MOT) application combining scores of localization, association and detection. We choose to use HOTA-0 score as the performance metric which sets localization threshold as 0: localization is successful if intersection between predicted and ground-truth bounding is larger than zero. The HOTA-0 score is above 90 for threshold lower than 0.5 and drop to 70 when threshold is above 0.5, which might due to too much computation skipped by the early termination logic. The average execution time of different ET thresholds is summarized in Fig.4.23 (bottom right). Early termination using threshold of 0.375 skips 46% execution time and does not lead to accuracy drop compared to computation without early termination.

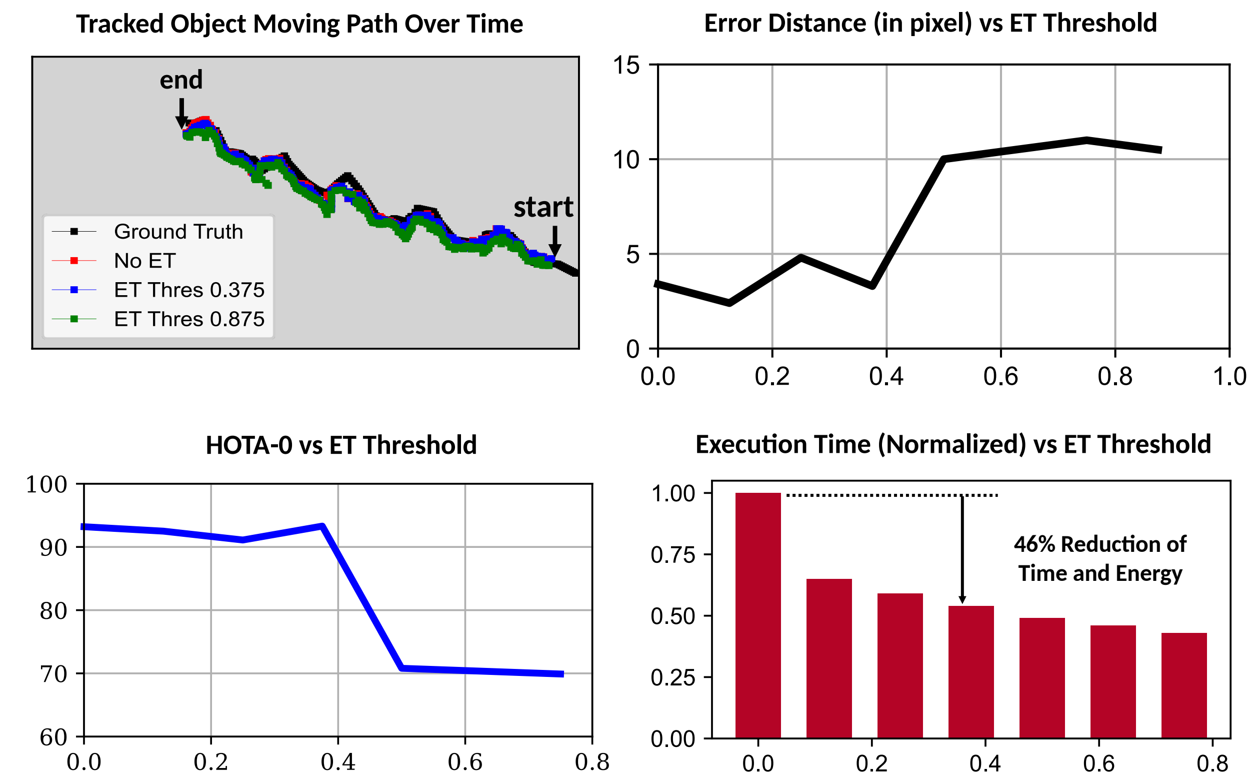
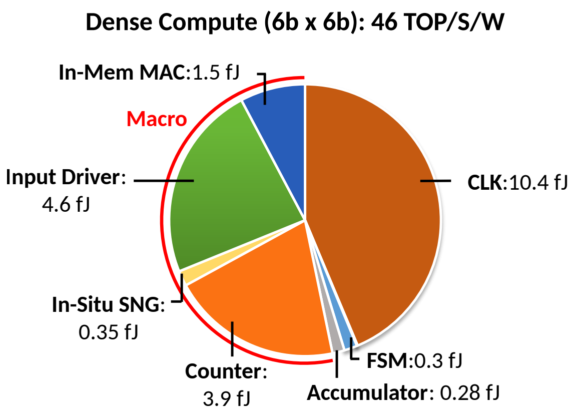
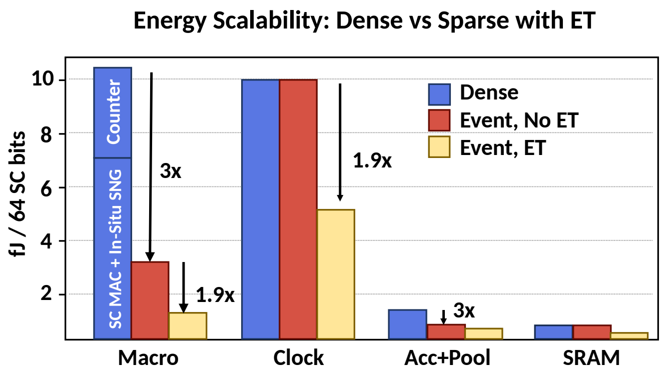


Figure 4.23: (Top Left) Tracked location over time for different early termination (ET) threshold; (Top Right) Distance error versus ground truth vs. ET threshold; (Bottom left) HOTA-0 score vs ET threshold; (Bottom right) Execution time vs ET threshold.

4.5.3 Energy Efficiency and Throughput

The energy efficiency is measured for both event and dense inputs. The energy breakdown of the accelerator is shown in Fig.4.24. The clock communication and SCIM macro account for most of the energy consumption. The energy of input driver and in-memory MAC units correspond to switching the input and output lines, which can be scaled with switching activities. Counter's energy consumption is not significantly changed with input's switching activity due to large clock switching energy. The energy is only reduced by 3$\times$ from dense to event processing. The clock communication energy is unchanged between dense and event processing. The level-3 early termination reduces both macro and clock communication energy by 1.9$\times$. The overall energy efficiency of macro and system is summarized in the table in Fig.4.24. For the event processing, the energy efficiency is 158 TOP/S/W for system and 495 TOP/S/W for macro.





	Event 99%	Frame 0%
Precision (Weight/IN)	6b x 2b	6b x 6b
Macro	495 TOP/S/W	86 TOP/S/W
System	158 TOP/S/W	46 TOP/S/W

Figure 4.24: Energy breakdown of the dense compute; Scalability between dense and sparse compute; Energy efficiency of macro and system.

The throughput of the accelerator is measured as number of events per second for different supply voltages. It ranges from 278 Mevents/s at 0.64V supply without early termination to 730

Mevents/s at 0.85V supply with early termination. The higher throughput benefits from the dense SC MAC array within each SCIM unit and the fast clock frequency. Analog CIMs using the bit-serial/bit-parallel scheme requires 6 clock cycles to compute a 6-bit MAC operation. Although SC requires 64 clock cycles for a 6-bit MAC operation, each SCIM unit packs 32 SC MACs. In average, each SC MAC only takes 2 clock cycles. which is 3 times faster. Due to SC's ultra-fast 1-bit operation, the SCIM macro operates at 600-850MHz, which is 6-8 times higher than the analog CIM solutions. The combination of the dense array and fast clock rate leads to 20-30 times higher throughput. When early termination is enable, it can reduce the stream length and helps increase the throughput further.

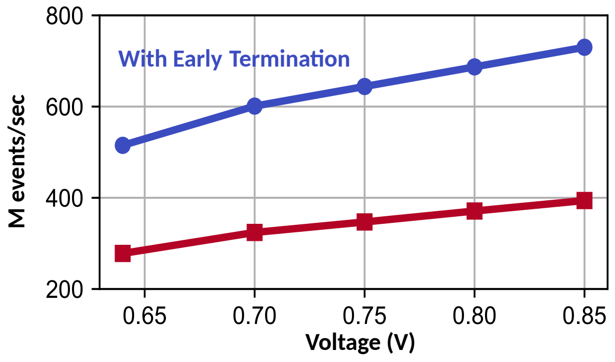
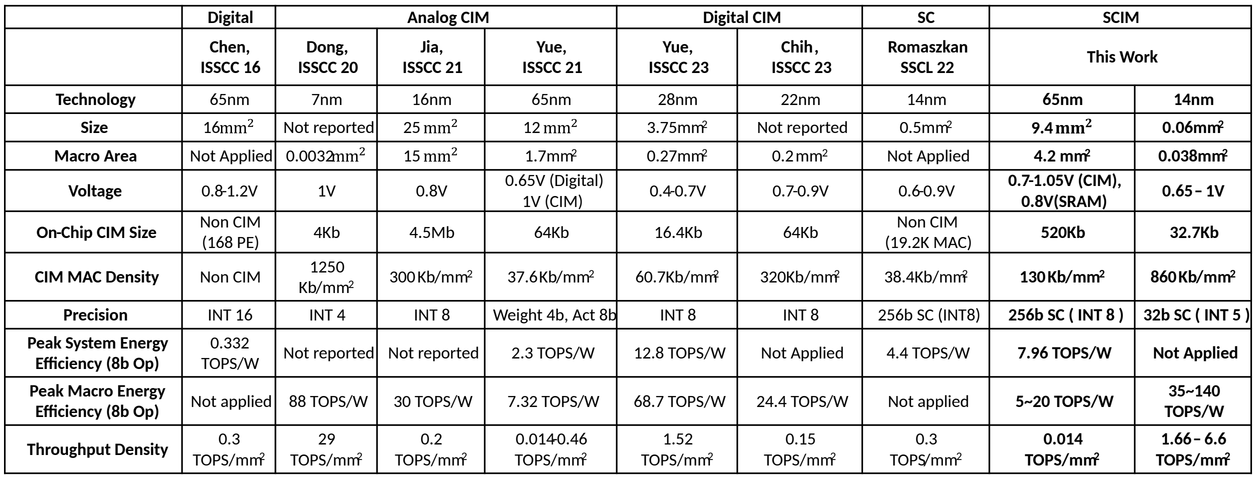


Figure 4.25: Throughput (Mevents/s) of the accelerator.

The comparison with other state-of-the-art works is shown in Fig.4.26. [61] is the only work that demonstrated object tracking application using event camera. It uses RRAM-based CIM macros for CNN acceleration and near-SRAM circuit to process event inputs, achieving event processing throughput of 11.1 Mevents/s and a throughput density around 3.7M events/mm^2. Our event camera accelerator achieves the energy efficiency of 158 TOP/S/W for event processing and 46 TOP/S/W for 6-bit dense image processing. The dense mode energy efficiency is much higher than [61] when scaled to 4b compute. The event processing throughput is 278M events/s at 0.64V supply and without early termination, which is 25x higher than [61]. The throughput density is 556M events/mm^2. We have also compared with digital CIM [13] [12], analog CIM [11] and SCIM with pre-converted stochastic numbers [30], but they are used for deep learning applications. Our work has shown dense-load throughput density >10x higher than other CIM works and comparable energy efficiency when scaled to the same computation precision.



	Digital	Analog CIM			Digital CIM		SC	SCIM	
	Chen, ISSCC 16	Dong, ISSCC 20	Jia, ISSCC 21	Yue, ISSCC 21	Yue, ISSCC 23	Chih, ISSCC 23	Romaszkan SSCL 22	This Work	
Technology	65nm	7nm	16nm	65nm	28nm	22nm	14nm	**65nm**	**14nm**
Size	16mm^2	Not reported	25 mm^2	12 mm^2	3.75mm^2	Not reported	0.5mm^2	**9.4 mm^2**	**0.06mm^2**
Macro Area	Not Applied	0.0032mm^2	15 mm^2	1.7mm^2	0.27mm^2	0.2 mm^2	Not Applied	**4.2 mm^2**	**0.038mm^2**
Voltage	0.8-1.2V	1V	0.8V	0.65V (Digital) 1V (CIM)	0.4-0.7V	0.7-0.9V	0.6-0.9V	**0.7-1.05V (CIM), 0.8V(SRAM)**	**0.65 – 1V**
On-Chip CIM Size	Non CIM (168 PE)	4Kb	4.5Mb	64Kb	16.4Kb	64Kb	Non CIM (19.2K MAC)	**520Kb**	**32.7Kb**
CIM MAC Density	Non CIM	1250 Kb/mm^2	300 Kb/mm^2	37.6Kb/mm^2	60.7Kb/mm^2	320Kb/mm^2	38.4Kb/mm^2	**130 Kb/mm^2**	**860 Kb/mm^2**
Precision	INT 16	INT 4	INT 8	Weight 4b, Act 8b	INT 8	INT 8	256b SC (INT8)	**256b SC (INT 8)**	**32b SC (INT 5)**
Peak System Energy Efficiency (8b Op)	0.332 TOPS/W	Not reported	Not reported	2.3 TOPS/W	12.8 TOPS/W	Not Applied	4.4 TOPS/W	**7.96 TOPS/W**	**Not Applied**
Peak Macro Energy Efficiency (8b Op)	Not applied	88 TOPS/W	30 TOPS/W	7.32 TOPS/W	68.7 TOPS/W	24.4 TOPS/W	Not applied	**5~20 TOPS/W**	**35~140 TOPS/W**
Throughput Density	0.3 TOPS/mm^2	29 TOPS/mm^2	0.2 TOPS/mm^2	0.014-0.46 TOPS/mm^2	1.52 TOPS/mm^2	0.15 TOPS/mm^2	0.3 TOPS/mm^2	**0.014 TOPS/mm^2**	**1.66 – 6.6 TOPS/mm^2**

Figure 4.26: Comparison with other works.

4.6 Conclusion

In this work, we have demonstrated a Stochastic Compute-in-Memory accelerator for event camera's high-speed tracking application. We propose embedding stochastic number generators in memory so that Stochastic Computing can take place right next to the stored data in binary format instead of storing pre-converted stochastic numbers. A massive 32 SC MAC units share in-situ SNG's output, which significantly increase throughput density. An early termination technique is proposed for Stochastic Computing to skip unnecessary computation when the preliminary outputs do not predict an object of interest. Our design has been validated by a 12nm chip hardware measurement and a complete object tracking pipeline in Python. The accelerator achieves a throughput of 278-730 Mevents/s and 158 TOP/S/W system energy efficiency, which is much better than the state-of-the-art works.

Bibliography

[1] B. Murmann, "ADC Performance Survey 1997-2023." [Online]. Available: https://github.com/bmurmann/ADC-survey.

[2] K. Yang, D. Blaauw, and D. Sylvester, "An all-digital edge racing true random number generator robust against pvt variations," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 4, pp. 1022–1031, 2016.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[7] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, "A survey of large language models," 2023.

[8] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[9] Q. Dong, M. E. Sinangil, B. Erbagci, D. Sun, W.-S. Khwa, H.-J. Liao, Y. Wang, and J. Chang, "15.3 a 351tops/w and 372.4gops compute-in-memory sram macro in 7nm finfet cmos for machine-learning applications," in *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, pp. 242–244, 2020.

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

[11] J. Yue, X. Feng, Y. He, Y. Huang, Y. Wang, Z. Yuan, M. Zhan, J. Liu, J.-W. Su, Y.-L. Chung, P.-C. Wu, L.-Y. Hung, M.-F. Chang, N. Sun, X. Li, H. Yang, and Y. Liu, “15.2 a 2.75-to-75.9tops/w computing-in-memory nn processor supporting set-associate block-wise zero skipping and ping-pong cim with simultaneous computation and weight updating,” in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, pp. 238–240, 2021.

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

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

[14] Y. D. Chih, P.-H. Lee, H. Fujiwara, Y.-C. Shih, C.-F. Lee, R. Naous, Y.-L. Chen, C.-P. Lo, C. Lu, H. Mori, W.-C. Zhao, D. Sun, M. E. Sinangil, Y.-H. Chen, T.-L. Chou, K. Akarvardar, H.-J. Liao, Y. Wang, M.-F. Chang, and T.-Y. J. Chang, “An 89tops/w and 16.3tops/mm2 all-digital sram-based full-precision compute-in memory macro in 22nm for machine-learning edge applications,” *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, pp. 252–254, 2021.

[15] V. Tripathi and B. Murmann, “Mismatch characterization of small metal fringe capacitors,” in *Proceedings of the IEEE 2013 Custom Integrated Circuits Conference*, pp. 1–4, 2013.

[16] V. B. Naik, K. Lee, K. Yamane, R. Chao, J. Kwon, N. Thiyagarajah, N. L. Chung, S. H. Jang, B. Behin-Aein, J. H. Lim, T. Y. Lee, W. P. Neo, H. Dixit, S. K, L. C. Goh, T. Ling, J. Hwang, D. Zeng, J. W. Ting, E. H. Toh, L. Zhang, R. Low, N. Balasankaran, L. Y. Zhang, K. W. Gan, L. Y. Hau, J. Mueller, B. Pfefferling, O. Kallensee, S. L. Tan, C. S. Seet, Y. S. You, S. T. Woo, E. Quek, S. Y. Siah, and J. Pellerin, “Manufacturable 22nm fd-soi embedded mram technology for industrial-grade mcu and iot applications,” in *2019 IEEE International Electron Devices Meeting (IEDM)*, pp. 2.3.1–2.3.4, 2019.

[17] H. L. Chiang, J. F. Wang, T. C. Chen, T. W. Chiang, C. Bair, C. Y. Tan, L. J. Huang, H. W. Yang, J. H. Chuang, H. Y. Lee, K. Chiang, K. H. Shen, Y. J. Lee, R. Wang, C. W. Liu, T. Wang, X. Bao, E. Wang, J. Cai, C. T. Lin, H. Chuang, H. S. P. Wong, and M. F. Chang, “Cold mram as a density booster for embedded nvm in advanced technology,” in *2021 Symposium on VLSI Technology*, pp. 1–2, 2021.

[18] C.-X. Xue, J.-M. Hung, H.-Y. Kao, Y.-H. Huang, S.-P. Huang, F.-C. Chang, P. Chen, T.-W. Liu, C.-J. Jhang, C.-I. Su, W.-S. Khwa, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, Y.-D. Chih, T.-Y. J. Chang, and M.-F. Chang, “16.1 a 22nm 4mb 8b-precision reram computing-in-memory macro with 11.91 to 195.7tops/w for tiny ai edge devices,” in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, pp. 245–247, 2021.

[19] W. Wan, R. Kubendran, C. Schaefer, S. Eryilmaz, W. Zhang, D. Wu, S. Deiss, P. Raina, H. Qian, B. Gao, S. Joshi, H. Wu, H.-S. Wong, and G. Cauwenberghs, "A compute-in-memory chip based on resistive random-access memory," *Nature*, vol. 608, pp. 504–512, 08 2022.

[20] R. Walden, "Analog-to-digital converter survey and analysis," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 539–550, 1999.

[21] H. Jia, H. Valavi, Y. Tang, J. Zhang, and N. Verma, "A programmable heterogeneous microprocessor based on bit-scalable in-memory computing," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 9, pp. 2609–2621, 2020.

[22] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Trans. Embed. Comput. Syst.*, vol. 12, pp. 92:1–92:19, 2013.

[23] D. Edelstein, M. Rizzolo, D. Sil, A. Dutta, J. DeBrosse, M. Wordeman, A. Arceo, I. C. Chu, J. Demarest, E. R. J. Edwards, E. R. Evarts, J. Fullam, A. Gasasira, G. Hu, M. Iwatake, R. Johnson, V. Katragadda, T. Levin, J. Li, Y. Liu, C. Long, T. Maffitt, S. McDermott, S. Mehta, V. Mehta, D. Metzler, J. Morillo, Y. Nakamura, S. Nguyen, P. Nieves, V. Pai, R. Patlolla, R. Pujari, R. Southwick, T. Standaert, O. van der Straten, H. Wu, C.-C. Yang, D. Houssameddine, J. M. Slaughter, and D. C. Worledge, "A 14 nm embedded stt-mram cmos technology," in *2020 IEEE International Electron Devices Meeting (IEDM)*, pp. 11.5.1–11.5.4, 2020.

[24] S. Kanai, M. Yamanouchi, S. Ikeda, Y. Nakatani, F. Matsukura, and H. Ohno, "Electric field-induced magnetization reversal in a perpendicular-anisotropy CoFeB-MgO magnetic tunnel junction," *Applied Physics Letters*, vol. 101, p. 122403, 09 2012.

[25] C. Grezes, F. Ebrahimi, J. Alzate, X. Q. Cai, J. Katine, J. Langer, B. Ocker, P. Amiri, and K. Wang, "Ultra-low switching energy and scaling in electric-field-controlled nanoscale magnetic tunnel junctions with high resistance-area product," *Applied Physics Letters*, vol. 108, p. 012403, 01 2016.

[26] S. K. Mathew, S. Srinivasan, M. A. Anders, H. Kaul, S. K. Hsu, F. Sheikh, A. Agarwal, S. Satpathy, and R. K. Krishnamurthy, "2.4 gbps, 7 mw all-digital pvt-variation tolerant true random number generator for 45 nm cmos high-performance microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 11, pp. 2807–2821, 2012.

[27] Q. Tang, B. Kim, Y. Lao, K. K. Parhi, and C. H. Kim, "True random number generator circuits based on single- and multi-phase beat frequency detection," in *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*, pp. 1–4, 2014.

[28] K. Yang, Q. Dong, Z. Wang, Y.-C. Shih, Y.-D. Chih, J. Chang, D. Blaauw, and D. Svlvester, "A 28nm integrated true random number generator harvesting entropy from mram," in *2018 IEEE Symposium on VLSI Circuits*, pp. 171–172, 2018.

[29] W. H. Choi, Y. Lv, J. Kim, A. Deshpande, G. Kang, J.-P. Wang, and C. H. Kim, "A magnetic tunnel junction based true random number generator with conditional perturb and real-

time output probability tracking," in *2014 IEEE International Electron Devices Meeting*, pp. 12.5.1–12.5.4, 2014.

[30] J. Yang, T. Li, W. Romaszkan, P. Gupta, and S. Pamarti, "A 65nm 8-bit all-digital stochastic-compute-in-memory deep learning processor," in *2022 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pp. 10–11, 2022.

[31] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.

[32] R. Zhang, X. Wang, L. Wang, X. Chen, F. Yang, K. Liu, and H. Shinohara, "A 0.186-pj per bit latch-based true random number generator with mismatch compensation and random noise enhancement," in *2021 Symposium on VLSI Circuits*, pp. 1–2, 2021.

[33] V. R. Pamula, X. Sun, S. Kim, F. u. Rahman, B. Zhang, and V. S. Sathe, "An all-digital true-random-number generator with integrated de-correlation and bias correction at 3.2-to-86 mb/s, 2.58 pj/bit in 65-nm cmos," in *2018 IEEE Symposium on VLSI Circuits*, pp. 1–2, 2018.

[34] S. Taneja and M. Alioto, "Fully synthesizable unified true random number generator and cryptographic core," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 10, pp. 3049–3061, 2021.

[35] X. Wang, R. Zhang, Y. Wang, K. Liu, X. Wang, and H. Shinohara, "A 0.116pj/bit latch-based true random number generator with static inverter selection and noise enhancement," in *2022 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1–4, 2022.

[36] A. Hajimiri, S. Limotyrakis, and T. Lee, "Jitter and phase noise in ring oscillators," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 6, pp. 790–804, 1999.

[37] J. S. Liberty, A. Barrera, D. W. Boerstler, T. B. Chadwick, S. R. Cottier, H. P. Hofstee, J. A. Rosser, and M. L. Tsai, "True hardware random number generation implemented in the 32-nm soi power7+ processor," *IBM Journal of Research and Development*, vol. 57, no. 6, pp. 4:1–4:7, 2013.

[38] K. Yang, D. Fick, M. B. Henry, Y. Lee, D. Blaauw, and D. Sylvester, "16.3 a 23mb/s 23pj/b fully synthesized true-random-number generator in 28nm and 65nm cmos," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 280–281, 2014.

[39] E. Kim, M. Lee, and J.-J. Kim, "8.2 8mb/s 28mb/mj robust true-random-number generator in 65nm cmos based on differential ring oscillator with feedback resistors," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 144–145, 2017.

[40] J. Yang, D. Wu, A. Lee, S. A. Razavi, P. Gupta, K. L. Wang, and S. Pamarti, "A calibration-free in-memory true random number generator using voltage-controlled mram," in *ESSDERC 2021 - IEEE 51st European Solid-State Device Research Conference (ESSDERC)*, pp. 115–118, 2021.

[41] N. Liu, N. Pinckney, S. Hanson, D. Sylvester, and D. Blaauw, "A true random number generator using time-dependent dielectric breakdown," in *2011 Symposium on VLSI Circuits - Digest of Technical Papers*, pp. 216–217, 2011.

[42] S. Taneja, V. K. Rajanna, and M. Alioto, "36.1 unified in-memory dynamic trng and multi-bit static puf entropy generation for ubiquitous hardware security," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, pp. 498–500, 2021.

[43] B. Dai, M. Jackson, Y. Cheng, H. He, Q. Shu, H. Huang, L. Tai, and K. Wang, "Special issue in honor of professor chia-ling chien review of voltage-controlled magnetic anisotropy and magnetic insulator," *Journal of Magnetism and Magnetic Materials*, vol. 563, p. 169924, 09 2022.

[44] S. Sakhare, M. Perumkunnil, T. H. Bao, S. Rao, W. Kim, D. Crotti, F. Yasin, S. Couet, J. Swerts, S. Kundu, D. Yakimets, R. Baert, H. Oh, A. Spessot, A. Mocuta, G. S. Kar, and A. Furnemont, "Enablement of stt-mram as last level cache for the high performance computing domain at the 5nm node," in *2018 IEEE International Electron Devices Meeting (IEDM)*, pp. 18.3.1–18.3.4, 2018.

[45] P. Gupta and R. Kumaresan, "Binary multiplication with pn sequences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 4, pp. 603–606, 1988.

[46] W. Romaszkan, T. Li, T. Melton, S. Pamarti, and P. Gupta, "Acoustic: Accelerating convolutional neural networks through or-unipolar skipped stochastic computing," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 768–773, 2020.

[47] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015.

[48] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

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

[51] W. Romaszkan, T. Li, R. Garg, J. Yang, S. Pamarti, and P. Gupta, "A 4.4–75-tops/w 14-nm programmable, performance- and precision-tunable all-digital stochastic computing neural network inference accelerator," *IEEE Solid-State Circuits Letters*, vol. 5, pp. 206–209, 2022.

[52] Z. Li, J. Li, A. Ren, R. Cai, C. Ding, X. Qian, J. Draper, B. Yuan, J. Tang, Q. Qiu, and Y. Wang, "Heif: Highly efficient stochastic computing-based inference framework for deep

neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 8, pp. 1543–1556, 2019.

[53] T. Li, W. Romaszkan, S. Pamarti, and P. Gupta, "Geo: Generation and execution optimized stochastic computing accelerator for neural networks," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 689–694, 2021.

[54] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128$\times$ 128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[55] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2022.

[56] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240 × 180 130 db 3 µs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[57] H. Chen, Q. Wu, Y. Liang, X. Gao, and H. Wang, "Asynchronous tracking-by-detection on adaptive time surfaces for event-based object tracking," in *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, ACM, Oct. 2019.

[58] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "EKLT: Asynchronous Photometric Feature Tracking Using Events and Frames," *International Journal of Computer Vision*, vol. 128, no. 3, pp. 601–618, 2020. Publisher: Springer US.

[59] A. Glover and C. Bartolozzi, "Event-driven ball detection and gaze fixation in clutter," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2203–2208, Oct. 2016. ISSN: 2153-0866.

[60] D. Falanga, S. Kim, and D. Scaramuzza, "How fast is too fast? the role of perception latency in high-speed sense and avoid," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1884–1891, 2019.

[61] A. S. Lele, M. Chang, S. D. Spetalnick, B. Crafton, S. Konno, Z. Wan, A. Bhat, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "A heterogeneous rram in-memory and sram near-memory soc for fused frame and event-based target identification and tracking," *IEEE Journal of Solid-State Circuits*, pp. 1–13, 2023.

[62] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8 μ j/86

[63] V. Kursun and E. Friedman, "Domino logic with variable threshold voltage keeper," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 1080–1093, 2003.

[64] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using

adaptive correlation filters," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2544–2550, 2010.

[65] T. Li, W. Romaszkan, S. Pamarti, and P. Gupta, "Rex-sc: Range-extended stochastic computing accumulation for neural network acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 12, pp. 4423–4435, 2023.

[66] J. Luiten, A. Osep, P. Dendorfer, P. H. S. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe, "HOTA: A higher order metric for evaluating multi-object tracking," *CoRR*, vol. abs/2009.07736, 2020.