In-Memory Computing based Machine Learning Accelerators: Opportunities and Challenges

KAUSHIK ROY

DIRECTOR, CENTER FOR BRAIN-INSPIRED COMPUTING

PURDUE UNIVERSITY

KAUSHIK@PURDUE.EDU

Machine (Deep) Learning Saga

1997

Deep Blue vs. Kasparov

~15000W

2011

Watson Wins Jeopardy

~200000W

2016

AlphaGo vs. Sedol

~300000W

2018

Autonomous Driving

~$$$$$

- **Advent of Deep Learning, 2012**
- **Fueled by powerful hardware - GPUs**

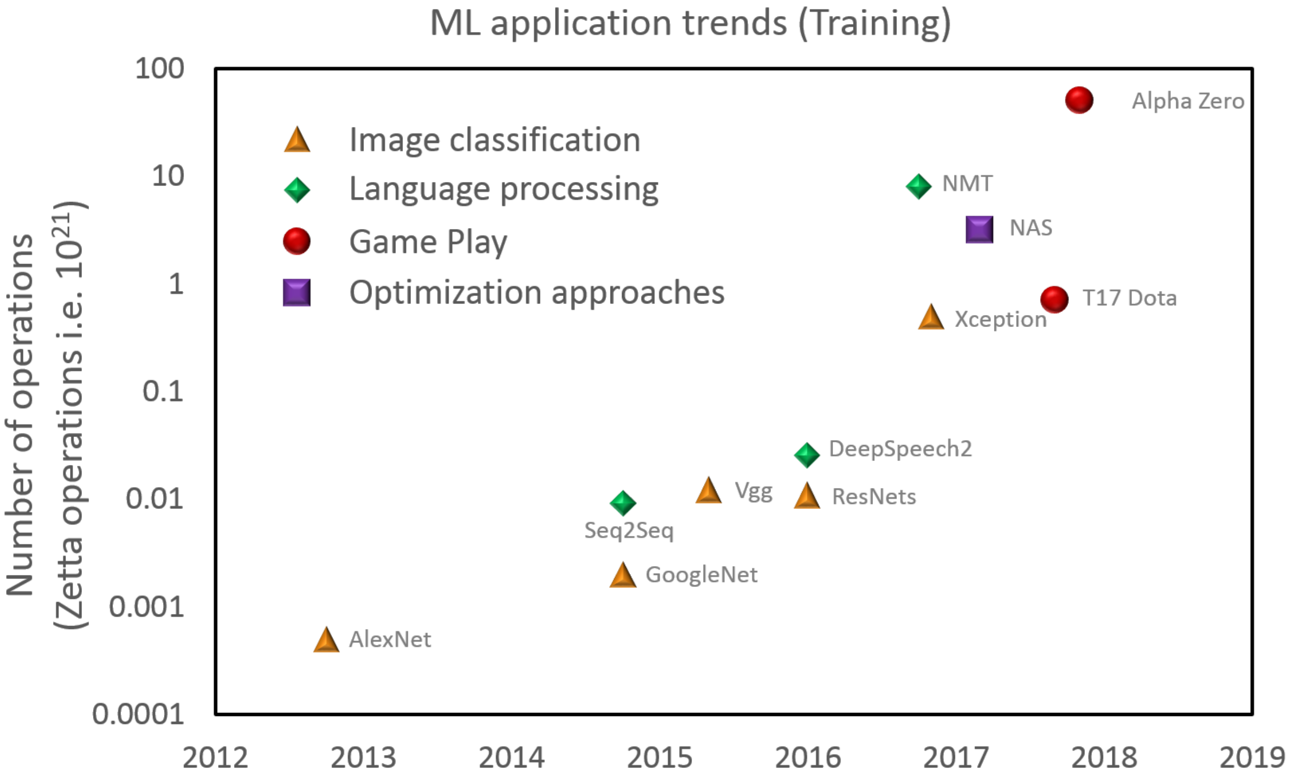
20W

Algorithm performance moving closer

Hardware cost moving farther

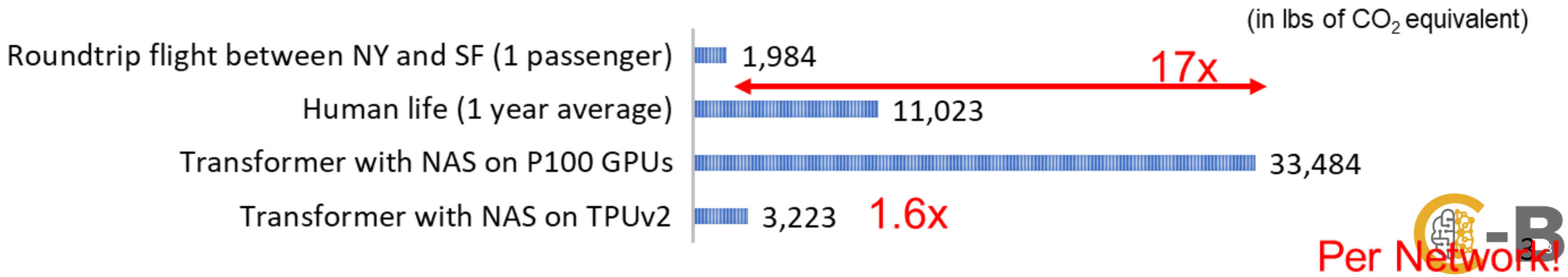
"Aspirations have grown faster than the technology available to satisfy them"

AI Compute Demands (Training)



ML application trends (Training)

COMMON CARBON FOOTPRINT BENCHMARKS



17x

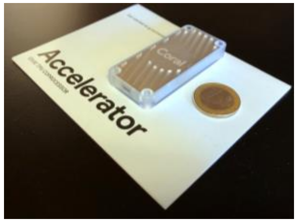
1.6x

Per Network!

Edge Intelligence: Efficiency Gap

- Case study: Object recognition in a smart glass with a state-of-the-art accelerator

+



Google Edge TPU

Retinanet DNN* on a smart glass

Performance	
Frames/sec	13.3
Battery Life	
Energy/op	0.5 pJ/op
Energy/frame	0.15 J/frame
Time-to-die (2.1WH)	**64 mins**

*300 GOPs/inference

Where do the in-efficiencies come from?

Algorithms

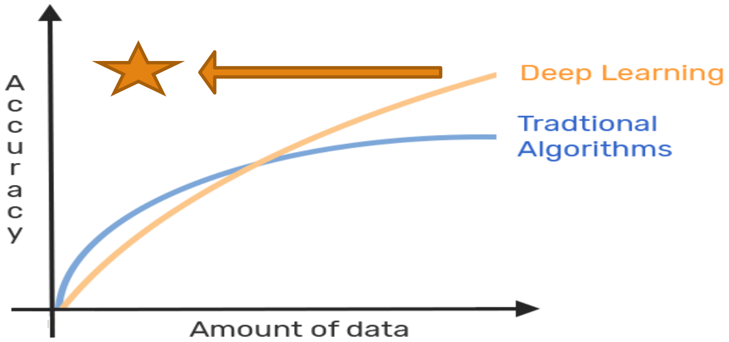
Hardware Architecture

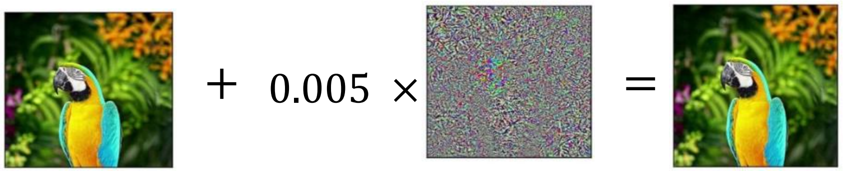
Circuits and Devices

Ref: Venkataramani, S., Roy, K. and Raghunathan, A. "Efficient embedded learning for IoT devices." In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 308-311. IEEE.

Beyond Compute Efficiency….

- Learning with less data
- Generalization & Robustness/ Security
- Lifelong learning



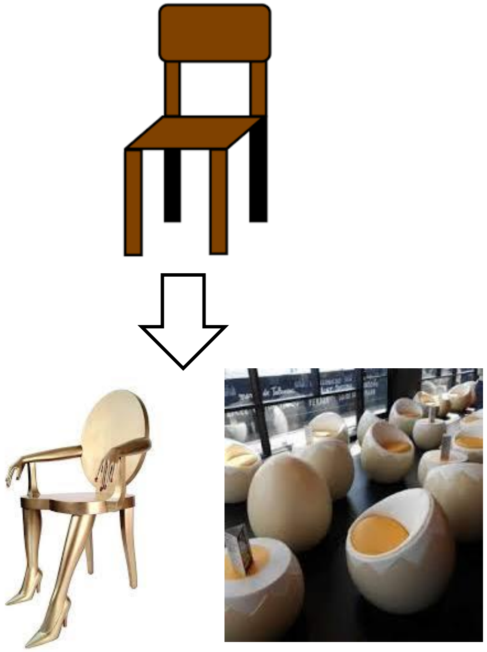


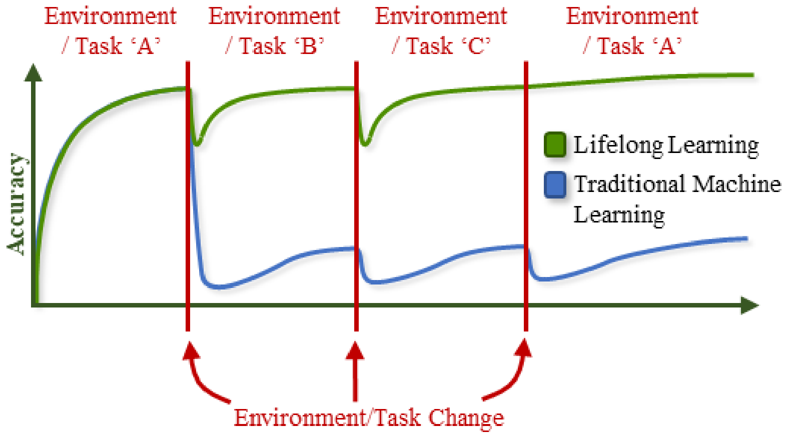
$X_{clean} + 0.005 \times \Delta = X_{adversary}$

97.3% confidence Macaw

Adversarial Perturbation

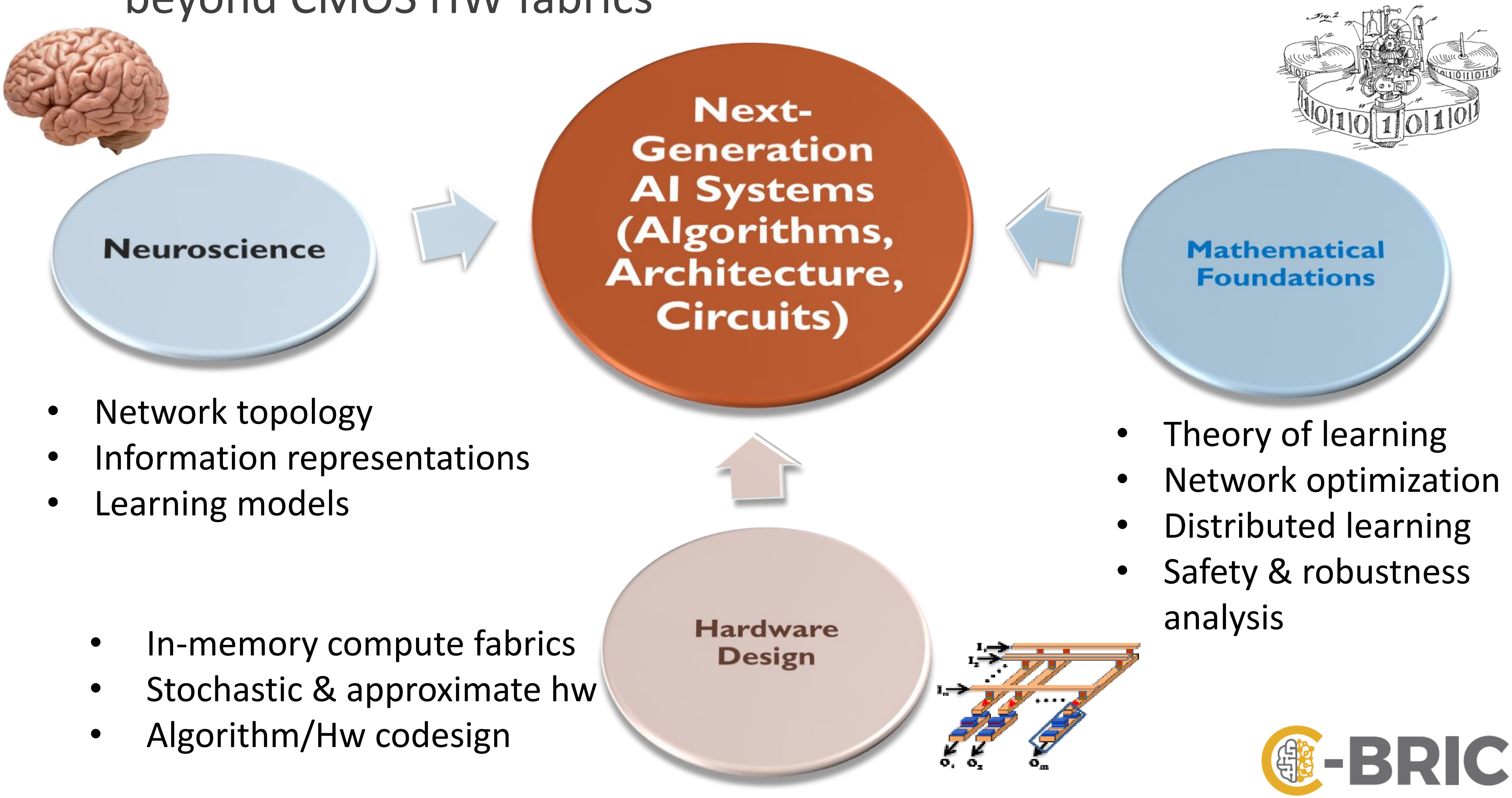
88.9% confidence Bookcase





Center for Brain-Inspired Computing (C-BRIC): Approach

- Design next-generation AI systems by drawing from neuroscience, mathematical foundations and using CMOS and beyond CMOS HW fabrics



Next-Generation AI Systems (Algorithms, Architecture, Circuits)

Neuroscience

- Network topology
- Information representations
- Learning models

Mathematical Foundations

- Theory of learning
- Network optimization
- Distributed learning
- Safety & robustness analysis

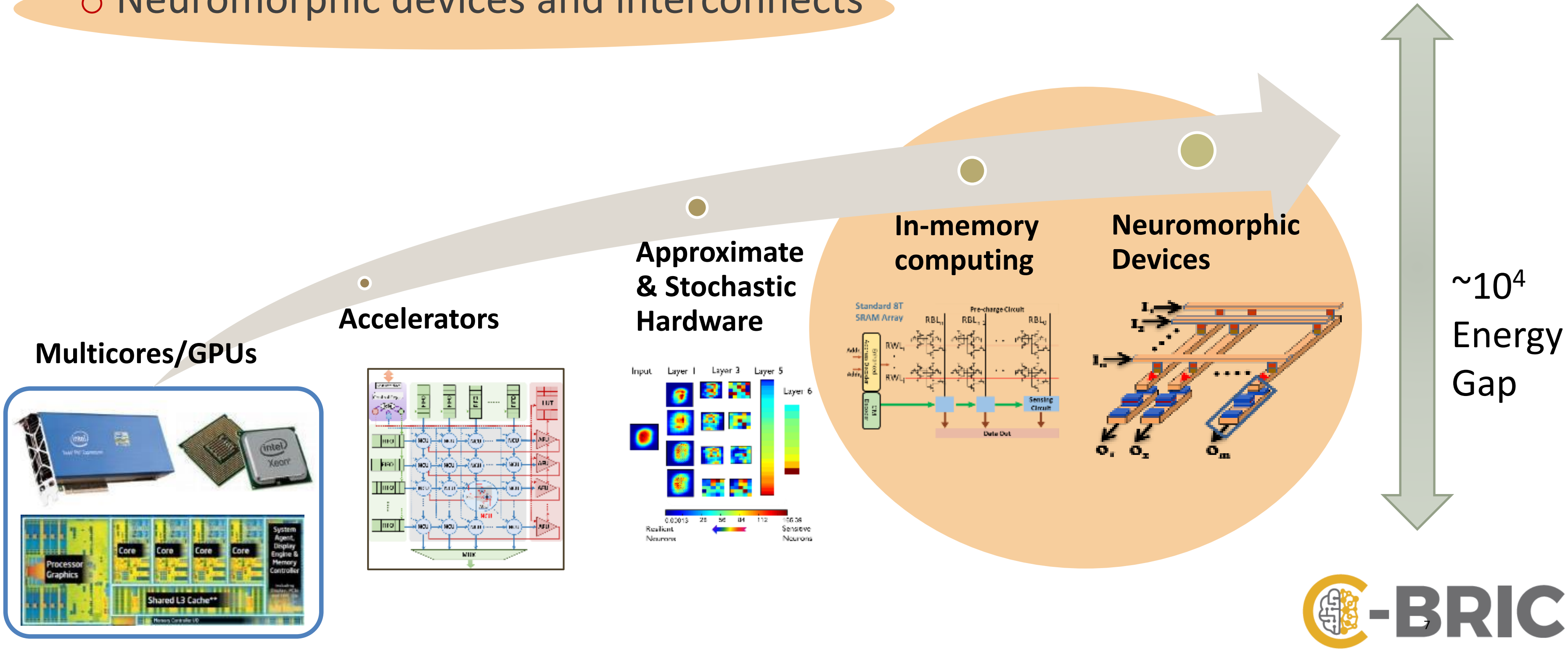
Hardware Design

- In-memory compute fabrics
- Stochastic & approximate hw
- Algorithm/Hw codesign

C-BRIC

AI Hardware Architecture: Circuits & Devices

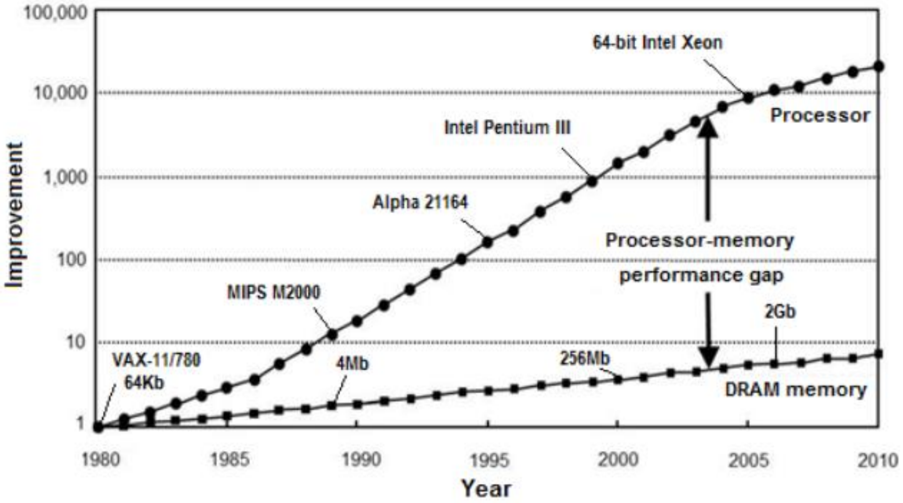
- Circuits and architectures that can efficiently implement the algorithms (possibly embody computing principles from the brain)
  - Near-/In-Memory Computing
  - Approximate and stochastic hardware
  - Neuromorphic devices and interconnects



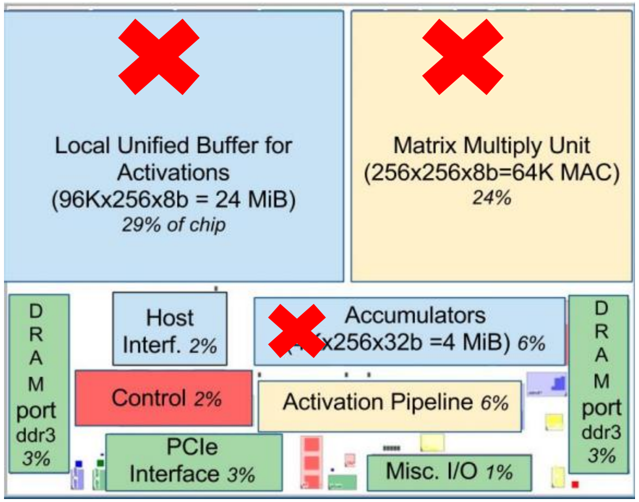
C-BRIC

Background – In-Memory Computing

- **Definition:** Design approach that performs computation close to memory to overcome memory bottlenecks – bandwidth, energy
- Effective for simple arithmetic - bit-wise operations; fixed-point add, multiply, Truth-tables (ROMs/RAMs)
- Typical systems have much higher compute throughput than memory bandwidth(s)
- Lots of chip area are memory components (>=50% in TPU)
  - Caches (L1, L2, …), Register File, Scratchpad, Buffers



Computer Architecture: A Quantitative Approach

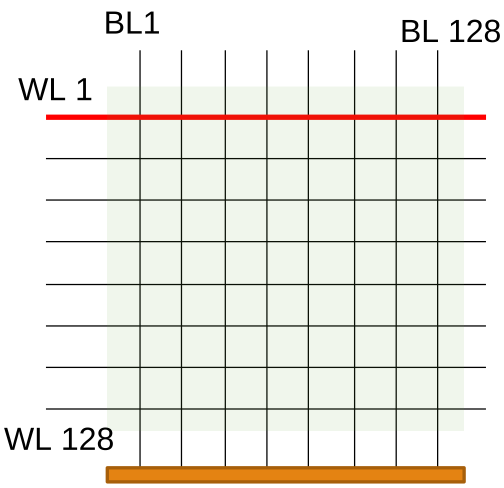


TPU Floorplan, ISCA 2017

In-Memory Computing for ML

2KB Memory

Typical Memory Access



Peripherals

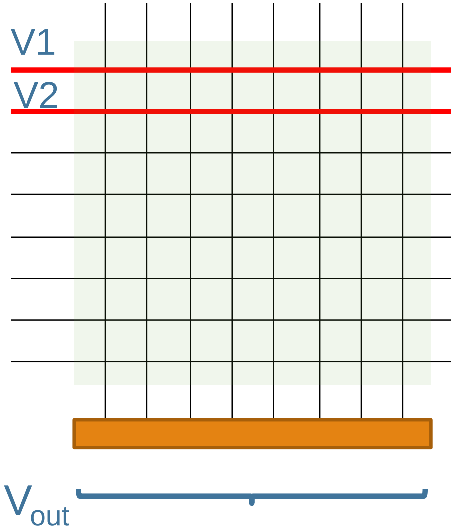
Bits Accessed = 128

Operation = Read/Write

2KB Memory

In-Memory Computing

Digital



Bits Accessed = 2*128

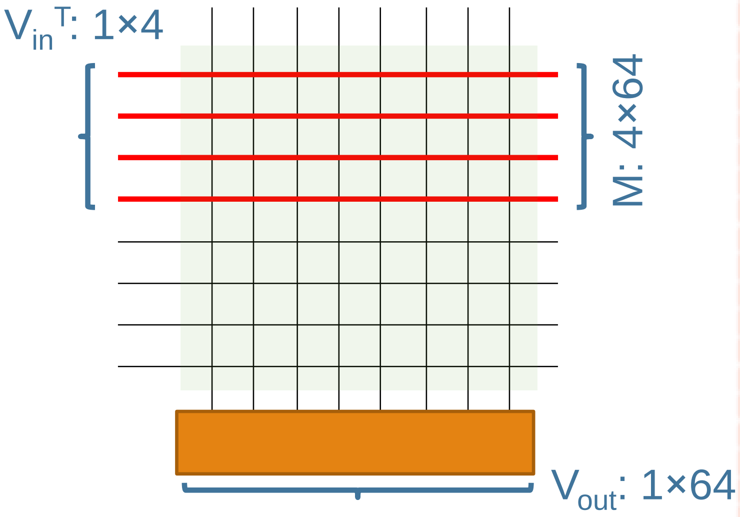
Operation = Bit-wise AND

V_{out} = V1 AND V2

2KB Memory

In-Memory Computing

Analog



Bits Accessed = 4*128

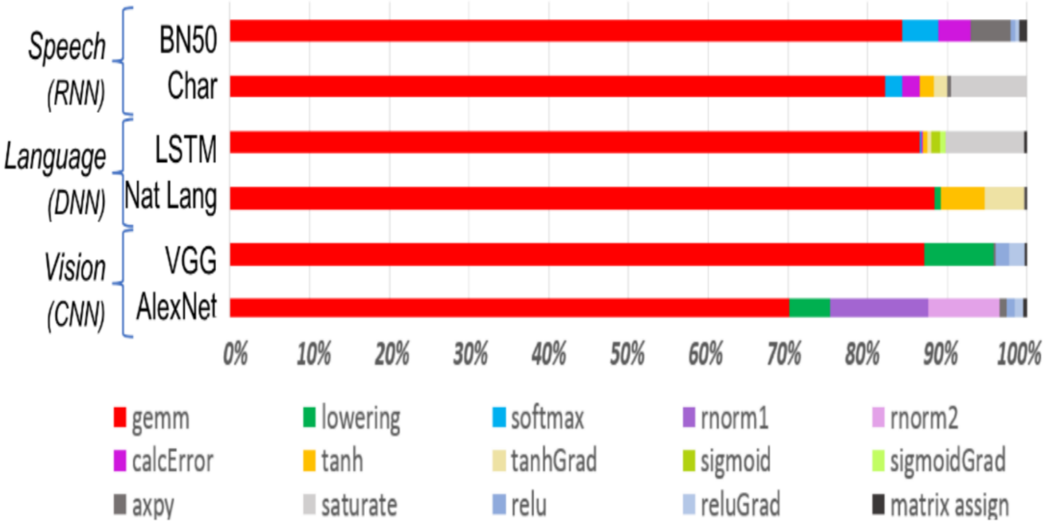
Operation = Matrix Vector Multiply

V_{out} = M * V_{in}

Non Volatile Memory	Jain et al. TVLSI’17, Abbrogio et al. Nature’18, Cai et al. Nature Elec.’19, Xue et al. ISSCC’20, Liu et al., ISSCC’20
SRAM	Biswas et al. ISSCC’18, Valavi et al. JSSC’19, Si et al. ISSCC’19, Jaiswal et al. TVLSI’20, Dong et al. ISSCC’20 (TSMC – 7nm)
ROM/RAM	Lee et. al. EDL 2013, Lee et. al. TVLSI 2013,

Machine Learning (Deep Learning)

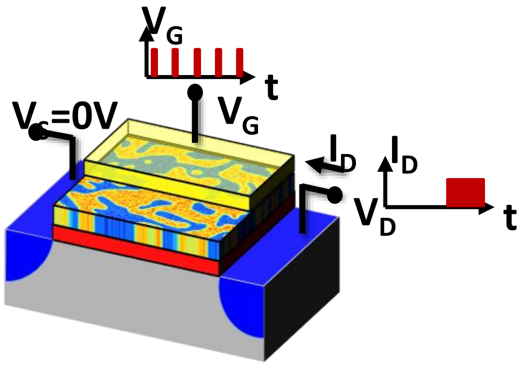
- Deep Learning needs – lots of matrix multiplications

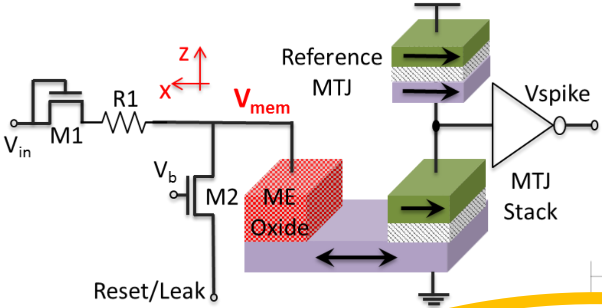


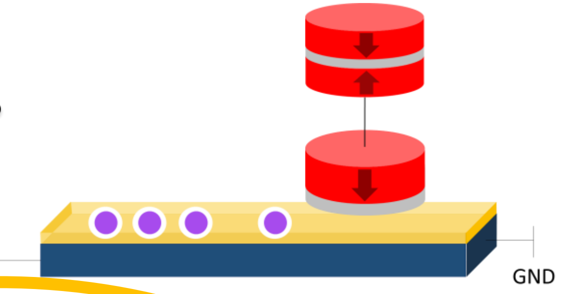
Bruce Fleischer et al, IBM Research, 2018

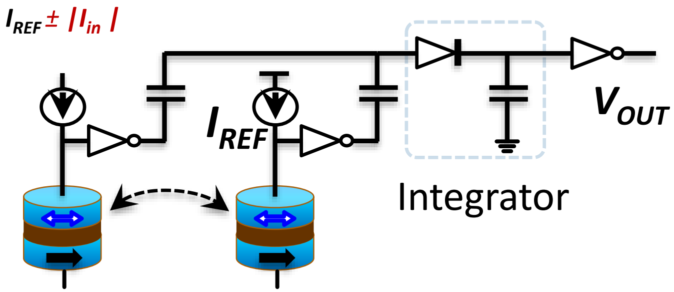
- Challenge: sustaining deep learning's insatiable compute demands

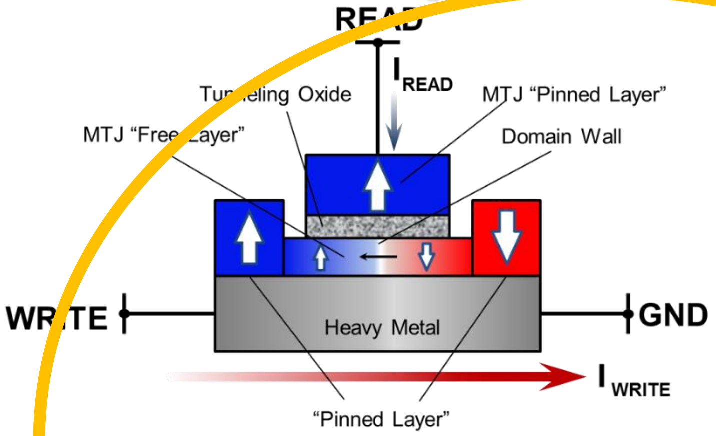
Technology: Non-Volatile Memories

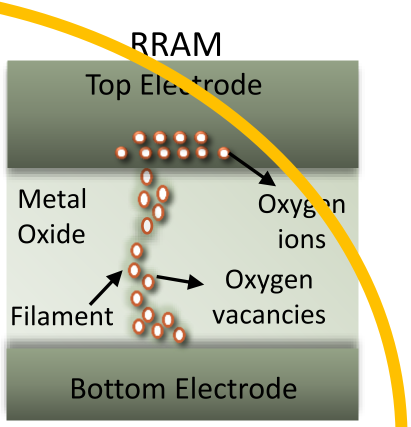


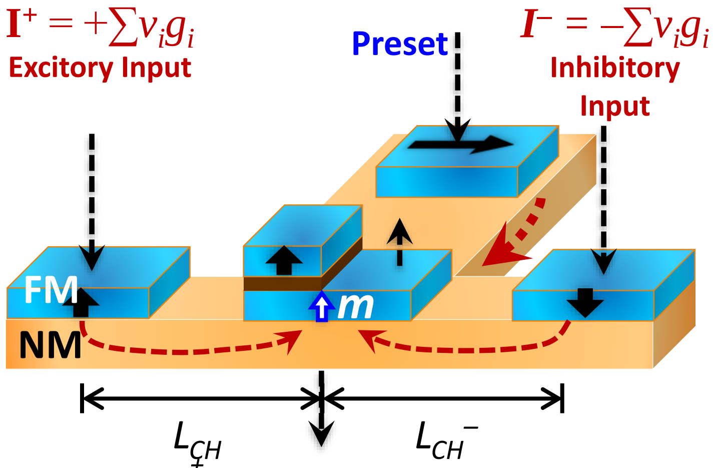


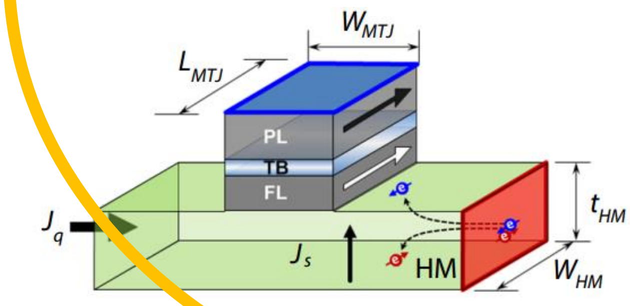


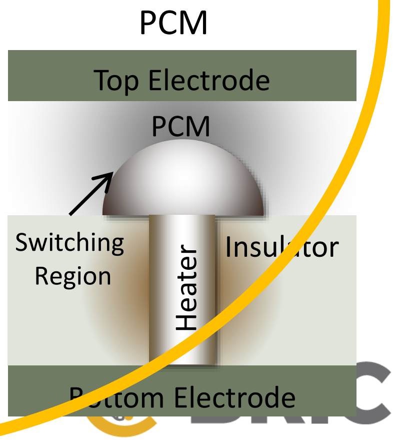










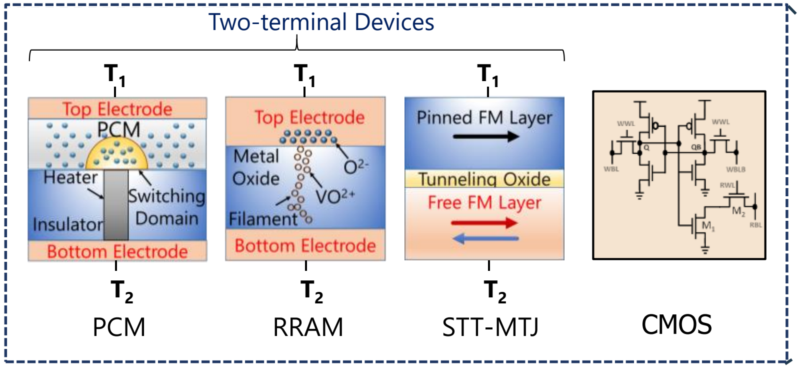


CMOS SRAM and Non-volatile Memories

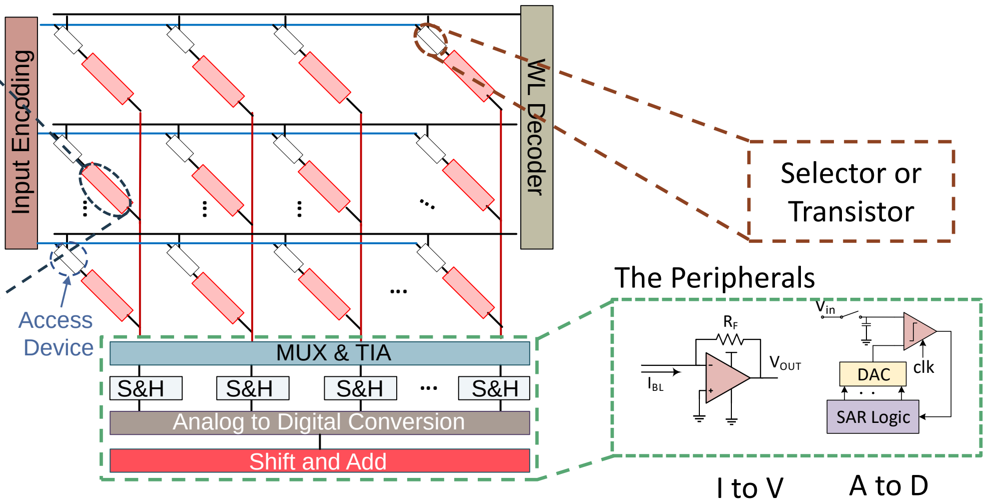
Property	PCM	RRAM	MTJ	CMOS (SRAM)
Multi-level cell	Yes	Yes	No	No
Storage Density	High	High	High	Low
R_{ON}/R_{OFF}	High	High	Low	High
Non-volatility	Yes	Yes	Yes	No
Leakage	Low	Low	Low	High
Cell Area	$16F^2$	$16F^2$	$30\text{-}80F^2$	$160F^2$ (6T), $231F^2$ (8T)
Write Energy	6 nJ	2 nJ	< 1 nJ	< 0.1 nJ
Write Latency	150 ns	100 ns	10 ns	< 1ns
Endurance	10^7 cycles	10^5 cycles	10^{15} cycles	> 10^{16} cycles

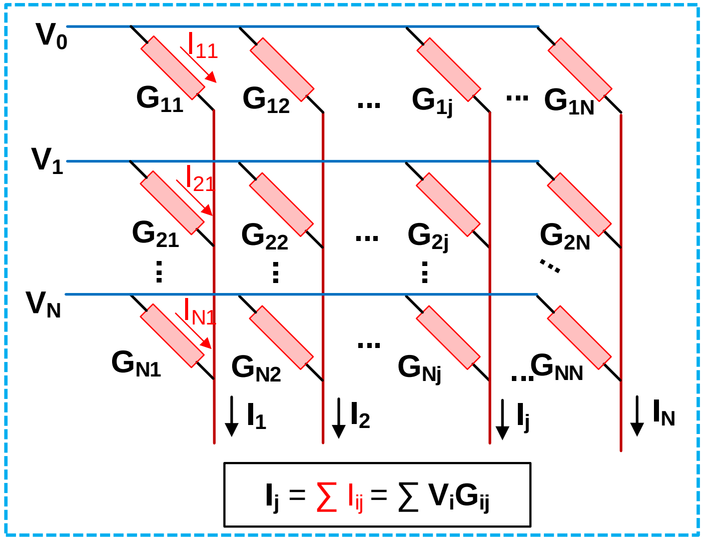
Fundamental building blocks of in-memory computing

In-Memory Computing Memory Devices

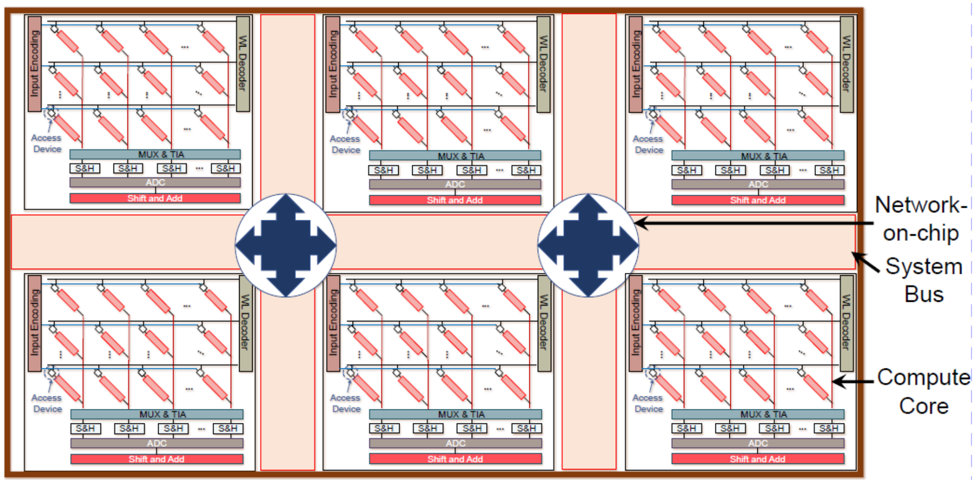


Chakraborty et. al. Resistive Crossbars as Approximate Hardware Building Blocks for Machine Learning: Opportunities and Challenges, Proc. of IEEE, 2020





$$I_j = \sum I_{ij} = \sum V_i G_{ij}$$

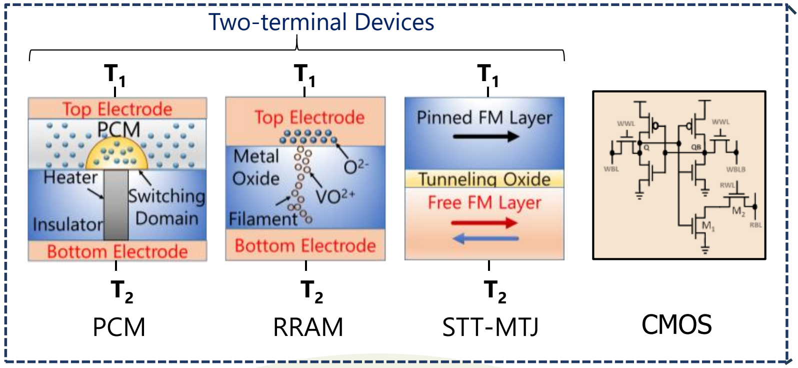


Efficient MVM

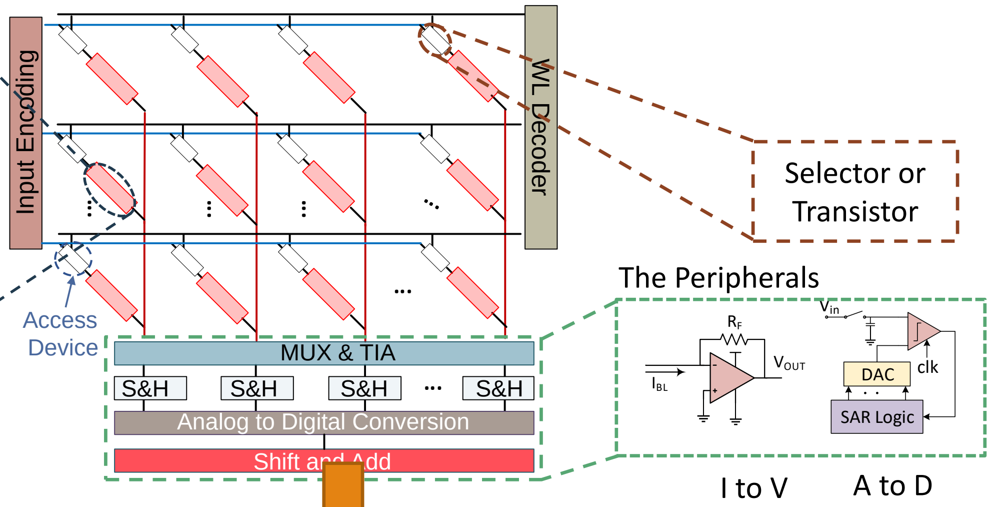
Spatially Distributed Cores

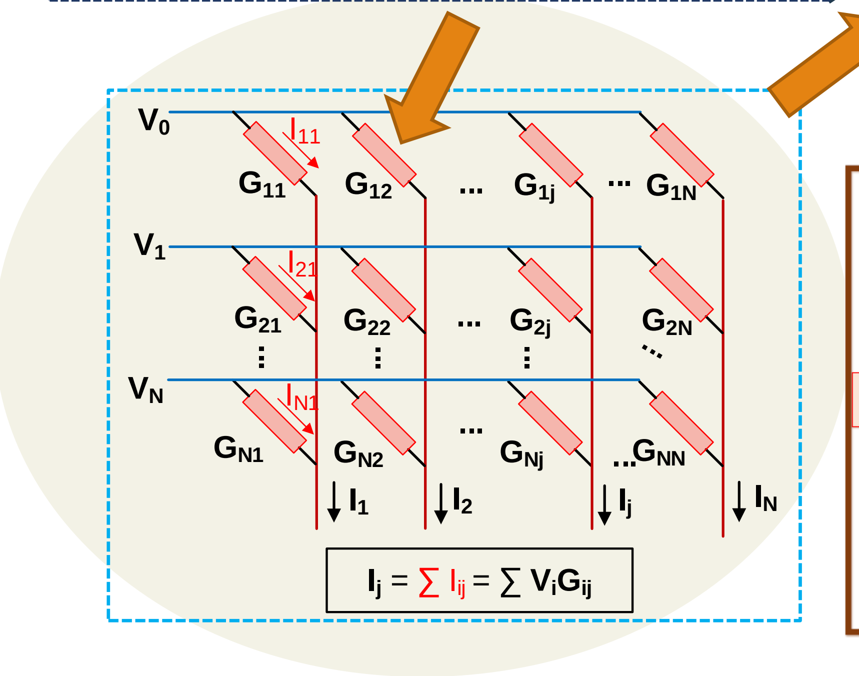
Efficient Hardware Architecture: CiM

In-Memory Computing Memory Devices



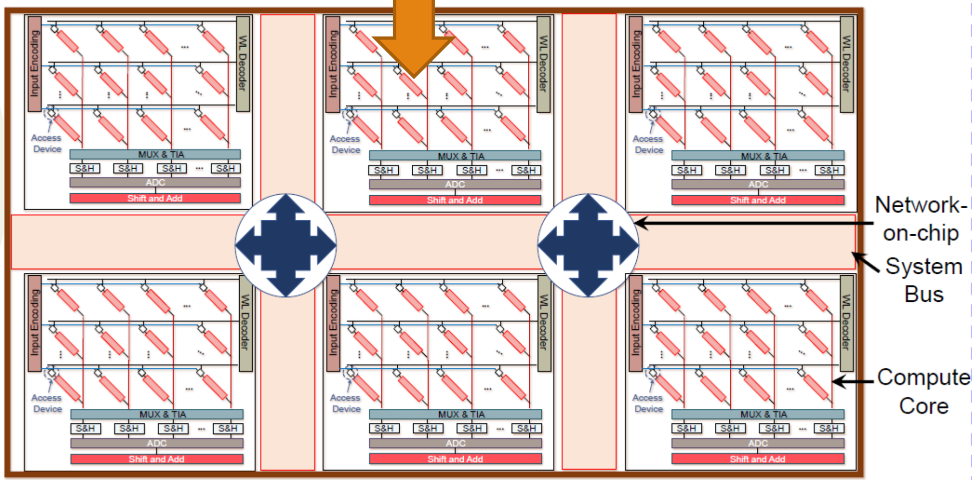
Chakraborty et. al. Resistive Crossbars as Approximate Hardware Building Blocks for Machine Learning: Opportunities and Challenges, Proc. of IEEE, 2020





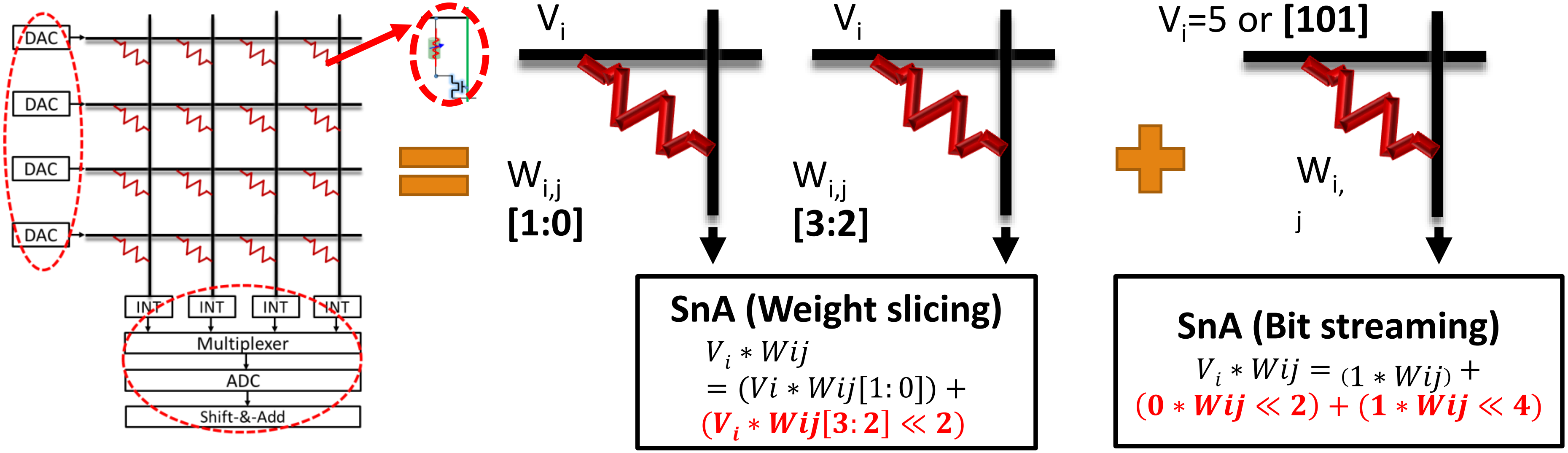
$$I_j = \sum I_{ij} = \sum V_i G_{ij}$$

Efficient MVM



Spatially Distributed Cores

Bit-slicing (weights and inputs)



SnA (Weight slicing)

$V_i * Wij$
$= (Vi * Wij[1:0]) +$
$(\boldsymbol{V_i} * \boldsymbol{Wij}[3:2] \ll 2)$

SnA (Bit streaming)

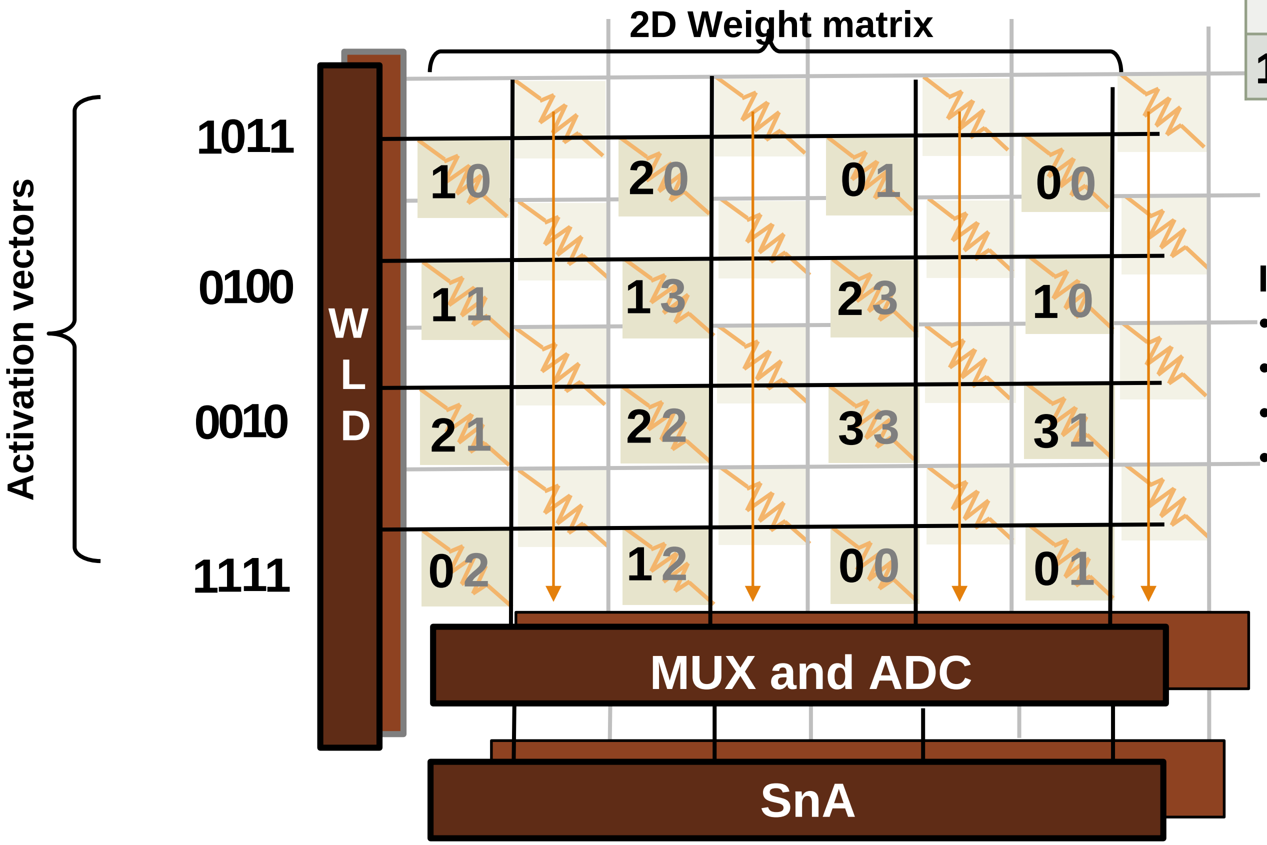
$V_i * Wij = (1 * Wij) +$
$(\mathbf{0} * \boldsymbol{Wij} \ll \mathbf{2}) + (\mathbf{1} * \boldsymbol{Wij} \ll \mathbf{4})$

- Bit-slicing: weight slicing and input streaming enable using **low precision crossbars** and **low precision DACs** to compose **high precision MVMU**

Analog CiM : Implementation details

- Parallel and efficient GEMV implementation:

$$\begin{bmatrix}11\\4\\2\\15\end{bmatrix} \times \begin{bmatrix}4 & 8 & 1 & 0\\5 & 7 & 11 & 4\\9 & 10 & 15 & 13\\2 & 6 & 0 & 1\end{bmatrix}$$



Important design points:

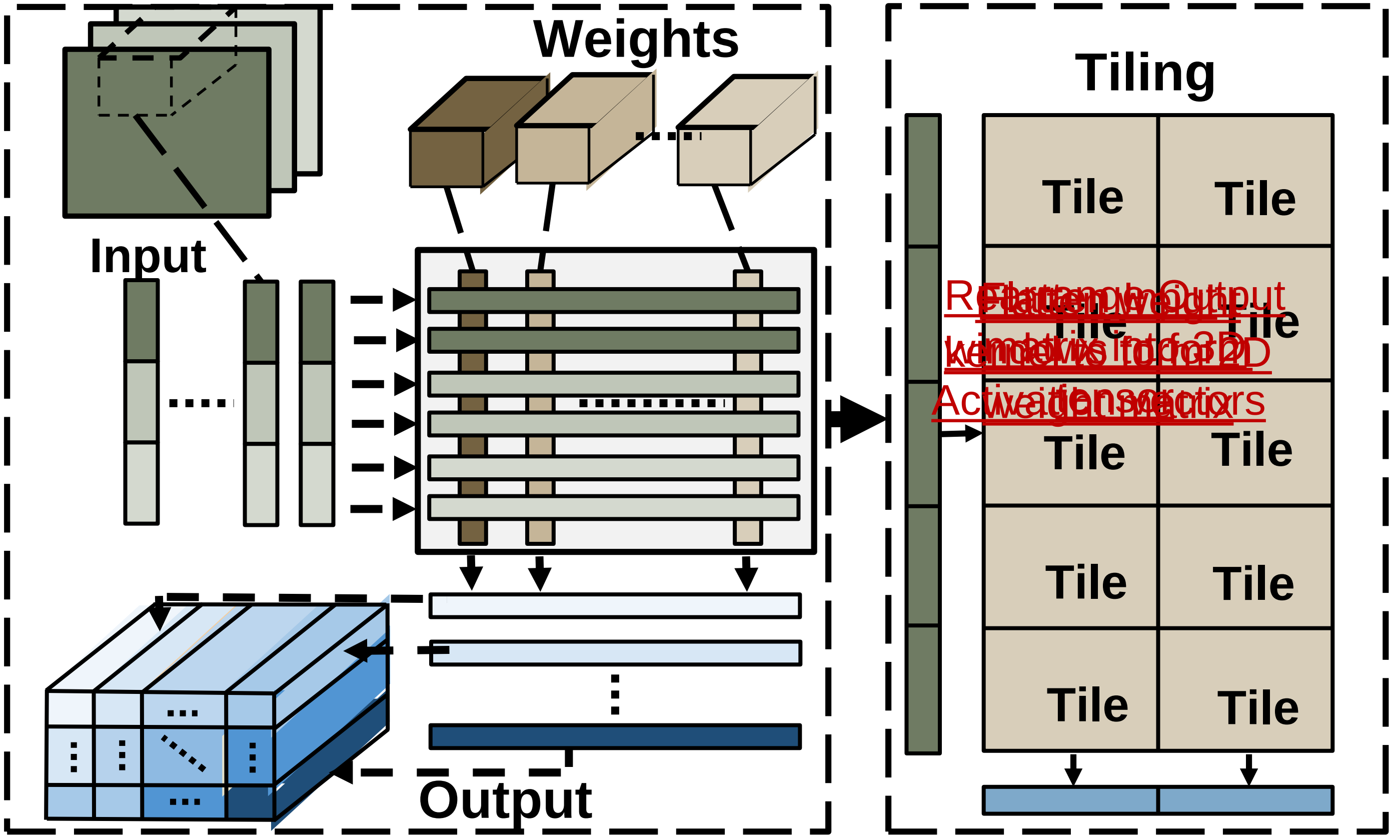
- Bit slice of weights
- Bit stream of activations
- Memory array size
- ADC precision

~~Divide high [illegible] memory devices~~

In Memory MVM

CiM processing details(1)

- Workload Mapping



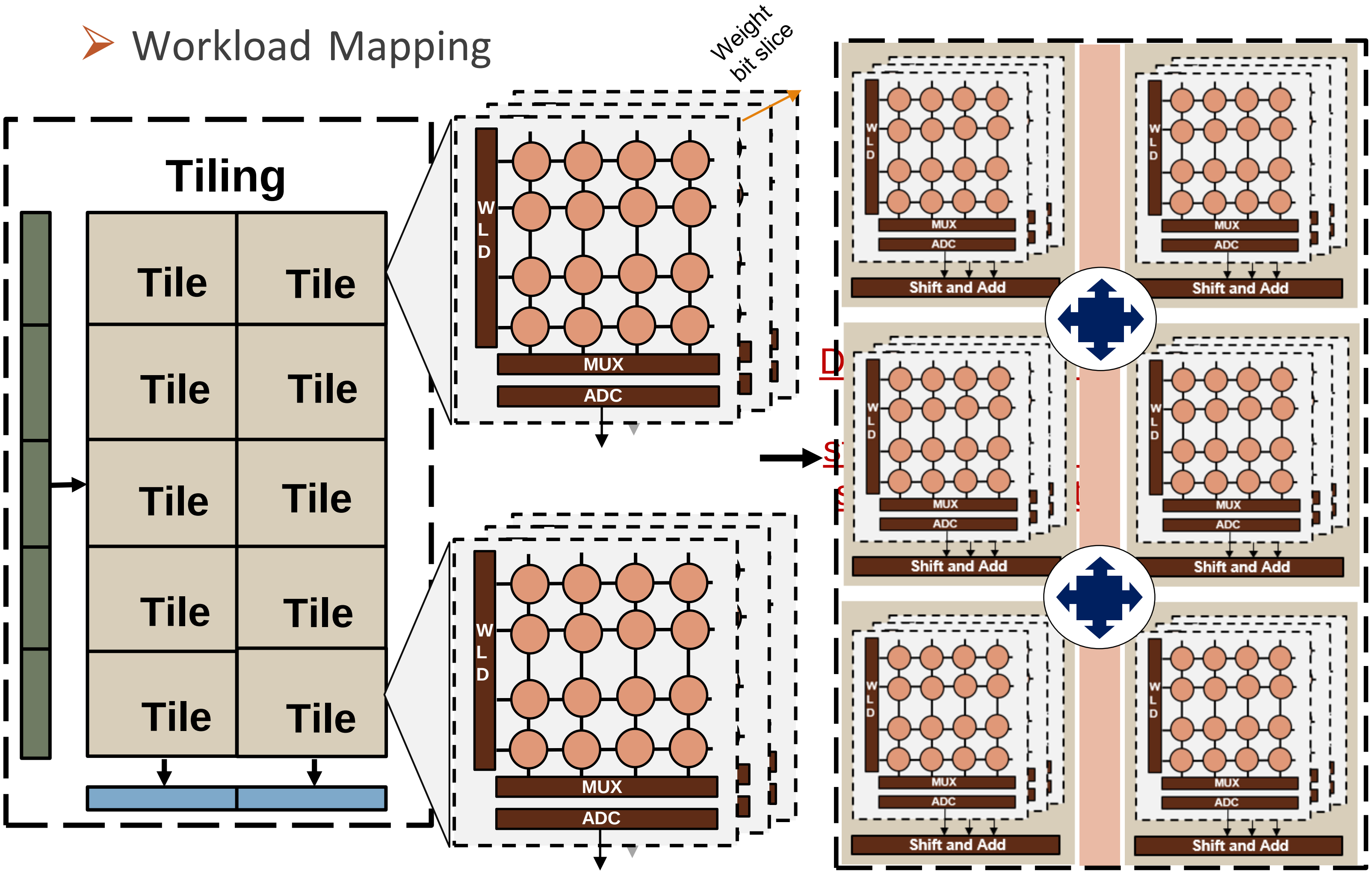
Rearrange Output vectors to 3D Activation tensors

Flatten Weight kernels to form 2D Weight matrix

Divide Weight matrix into tiles based on memory array rows

CiM processing details(2)

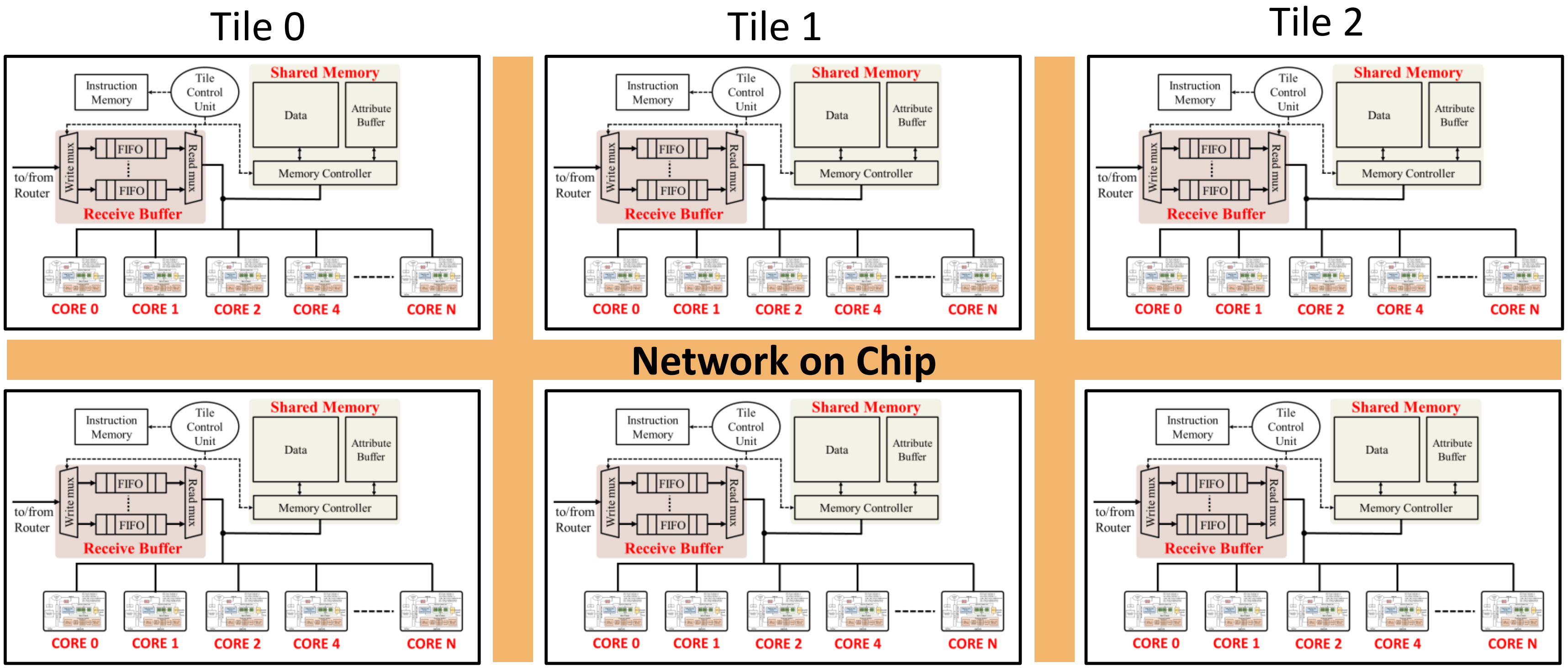
- Workload Mapping



Intra Tile Accumulation of partial sums using Shift and Add block.

Inter tile Accumulation using NoC

Architecture: *Spatial scalability*



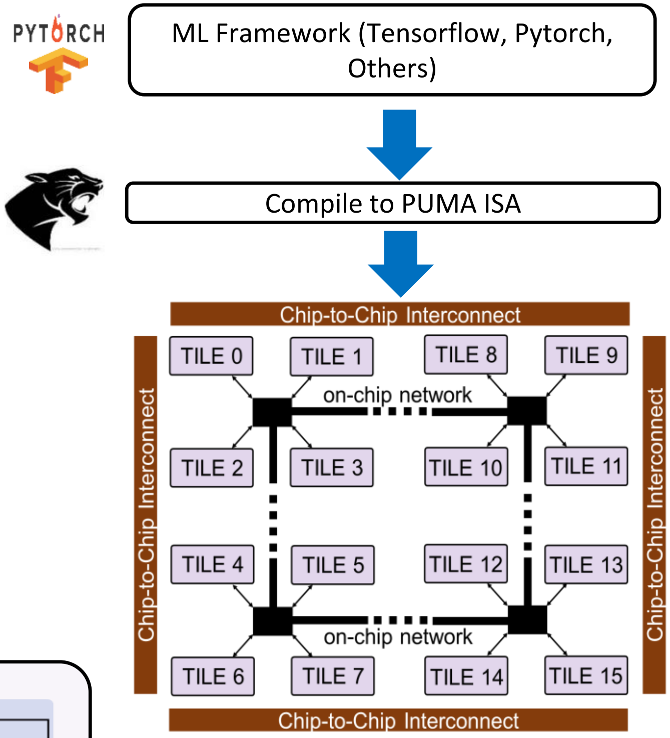
Massively parallel accelerator –> Amenable to Data-Level Parallelism -> Highly efficient ML inference

Ankit, Roy, et. Al., “PUMA: A Programmable Ultra-efficient Memristor-based Accelerator for Machine Learning Inference”, ASPLOS 2019.

PUMA: Resistive Crossbar based Programmable Architecture

➢ Features

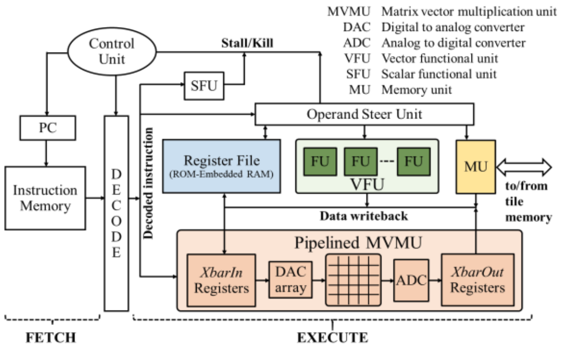
- ○ Analyze the memory-compute characteristics of ML applications
- ○ An ISA-programmable accelerator built with hybrid CMOS-NVM technology



PUMA Chip

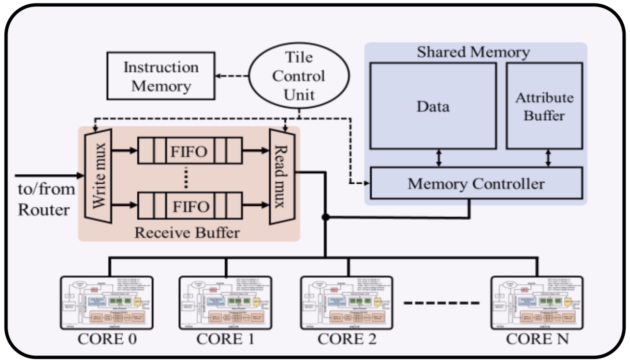
(dataflow architecture)

A. Ankit et al, ASPLOS, 2019



PUMA Core

(NVM Crossbar + Digital CMOS)

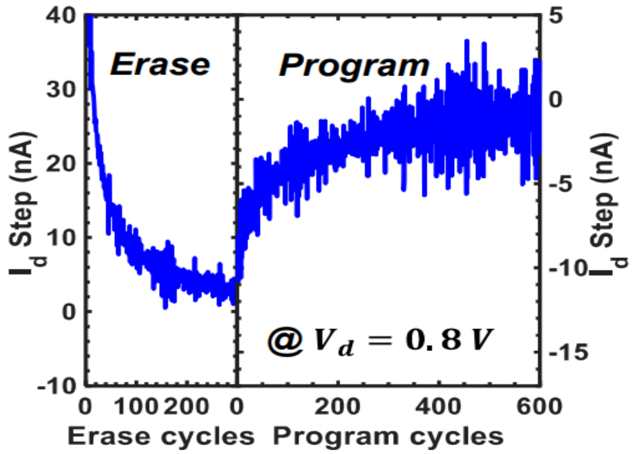


PUMA Tile (multi-core)

Challenges: NVM devices

Device
High ON/OFF Ratio
Device Linearity
High Endurance
Multi-level cells

- Compared to CMOS:
  - ✓ Non-volatility
  - ✓ High density
  - ✓ Low leakage
  - ✓ Capable of in-memory compute
  - ×Write energy/latency
- Current devices are highly non-linear
  - Expensive write operations and peripheral circuitry
- R_{ON}/R_{OFF} ratios are limited to ~10×
- RRAM has poor endurance.
- More than 4-bits/cell is not reliable yet.



[1] IEDM, 2019

Property	PCM	RRAM	MTJ	CMOS
Multi-level cell	Yes	Yes	No	No
Storage Density	High	High	High	Low
R_{ON}/R_{OFF}	High	High	Low	High
Non-volatility	Yes	Yes	Yes	No
Leakage	Low	Low	Low	High
Write Energy	6 nJ	2 nJ	< 1 nJ	< 0.1 nJ
Write Latency	150 ns	100 ns	10 ns	< 1ns
Endurance	10^7 cycles	10^5 cycles	10^{15} cycles	> 10^{16} cycles

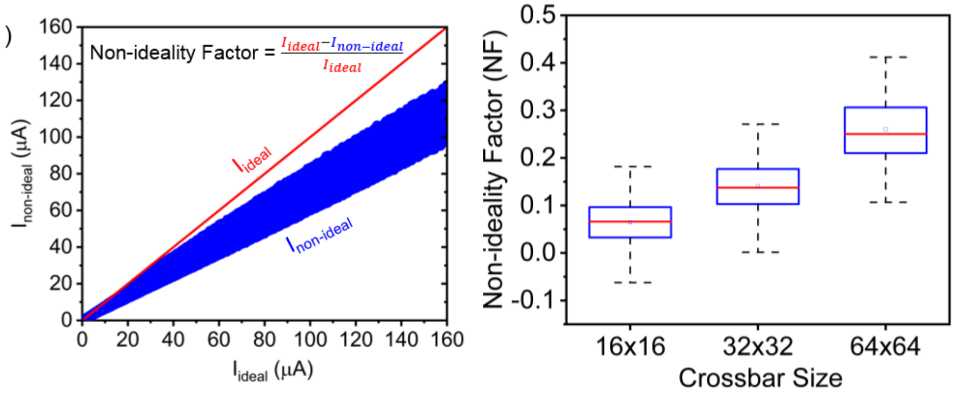
C-BRIC

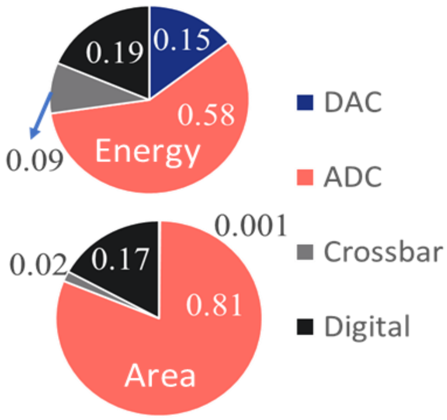
Macro

Large Crossbar Size
Low cost peripheral overhead
Good selector device
Source/Sink/Line resistances

Challenges: NVM Compute Macro

- NVM crossbars can have various non-idealities (parasitics, non-ideal devices)
- Such non-idealities can introduce varying amounts of functional errors based on different voltage and conductance
- Errors increase with higher crossbar sizes
- ADCs consume 58% and >80% of the total energy and area, respectively





Analog Operations	Energy (pJ)	Dig. Operations	Energy (pJ)
MVM Energy	3.84	ALU	25.6
ADC Energy	128	Access (FMA)	480
Other peripherals	12.8		
Total	144.6	Total	505.6

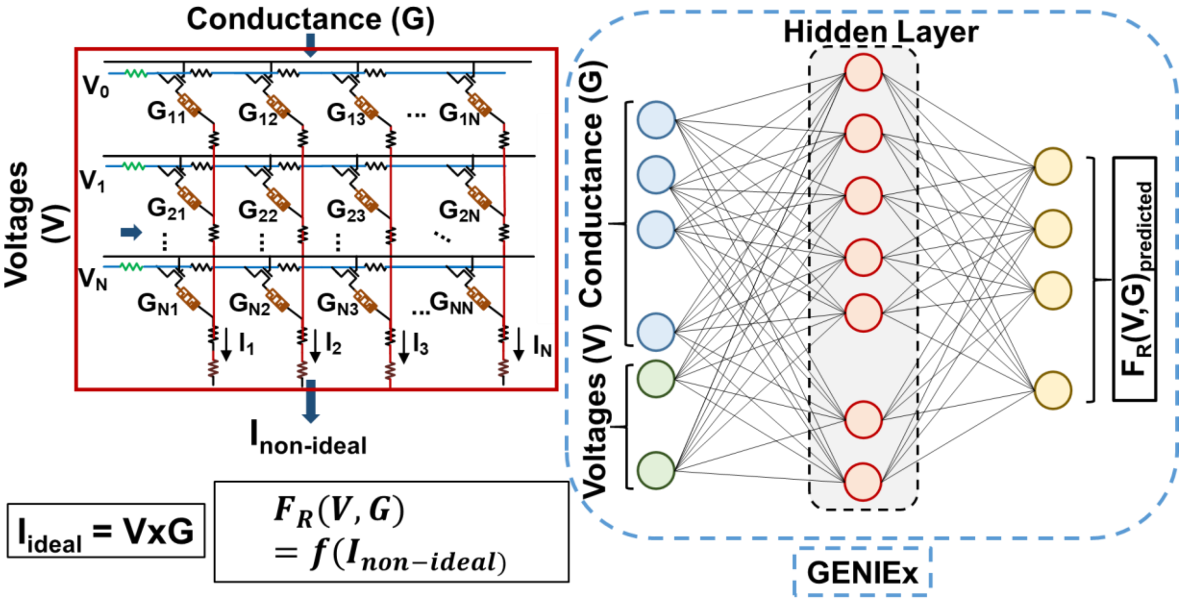
8-bit MVM

Source: Ankit et al, ASPLOS 2019

GENIEx: A Generalized Approach to Emulating Non-Ideality in Memristive X-bars

$$I_{column} = f(V_i, G_{ij}(V), R_{source}, R_{sink}, R_{wire})$$

- f is a data-dependent non-linear function.
- Neural networks are efficient tools for capturing the close inter-dependence of its inputs.
- Neural network to model the behavior of non-ideal crossbars



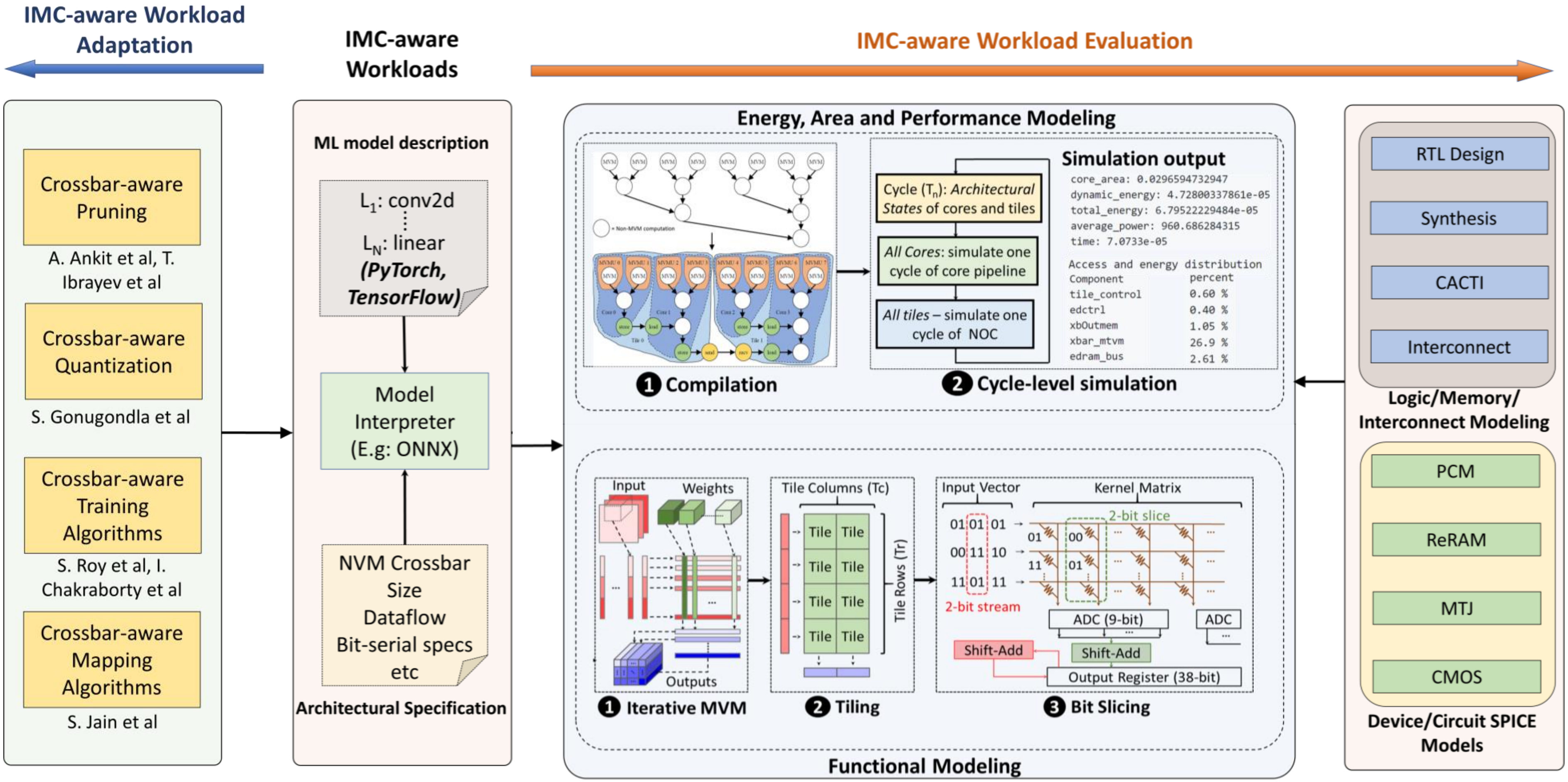
I_{ideal} = VxG

$$F_R(V,G) = f(I_{non-ideal})$$

Chakraborty et al, DAC 2020, https://arxiv.org/pdf/2003.06902.pdf

- GENIEx provides modeling capability for different non-idealities

Resistive Crossbar Based Accelerator Design Flow



IMC-aware Workload Adaptation

IMC-aware Workloads

IMC-aware Workload Evaluation

Crossbar-aware Pruning

A. Ankit et al, T. Ibrayev et al

Crossbar-aware Quantization

S. Gonugondla et al

Crossbar-aware Training Algorithms

S. Roy et al, I. Chakraborty et al

Crossbar-aware Mapping Algorithms

S. Jain et al

ML model description

L_1: conv2d
⋮
L_N: linear
(PyTorch, TensorFlow)

Model Interpreter (E.g: ONNX)

NVM Crossbar Size
Dataflow
Bit-serial specs
etc

Architectural Specification

Energy, Area and Performance Modeling

1 Compilation

Cycle (T_n): *Architectural States* of cores and tiles

All Cores: simulate one cycle of core pipeline

All tiles – simulate one cycle of NOC

Simulation output

```
core_area: 0.0296594732947
dynamic_energy: 4.72800337861e-05
total_energy: 6.79522229484e-05
average_power: 960.686284315
time: 7.0733e-05

Access and energy distribution
Component                percent
tile_control             0.60 %
edctrl                   0.40 %
xbOutmem                 1.05 %
xbar_mtvm                26.9 %
edram_bus                2.61 %
```

2 Cycle-level simulation

1 Iterative MVM

2 Tiling

3 Bit Slicing

Functional Modeling

RTL Design

Synthesis

CACTI

Interconnect

Logic/Memory/ Interconnect Modeling

PCM

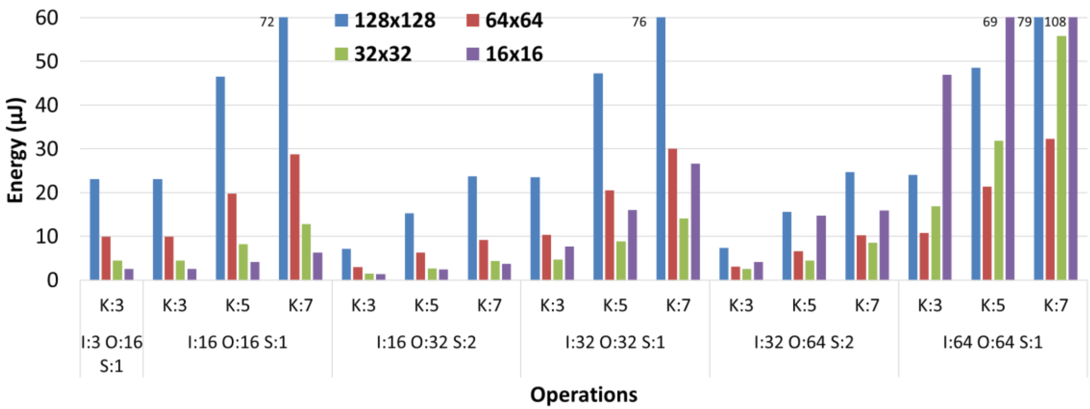
ReRAM

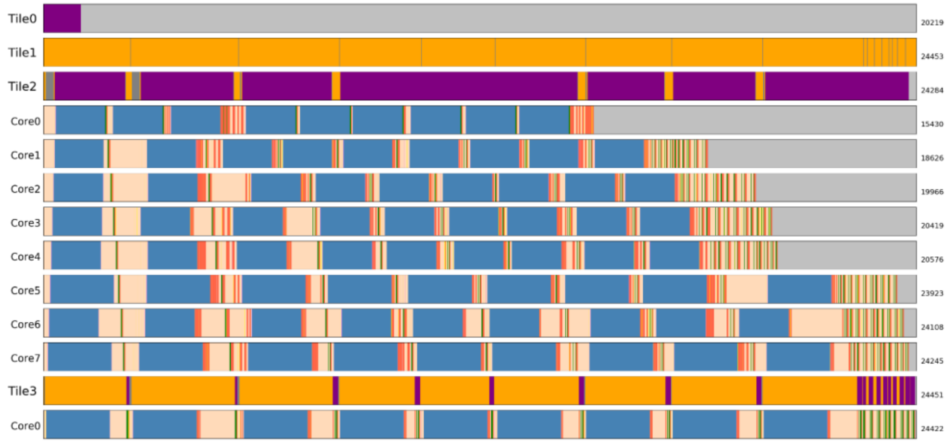
MTJ

CMOS

Device/Circuit SPICE Models

Performance Simulator – Scope

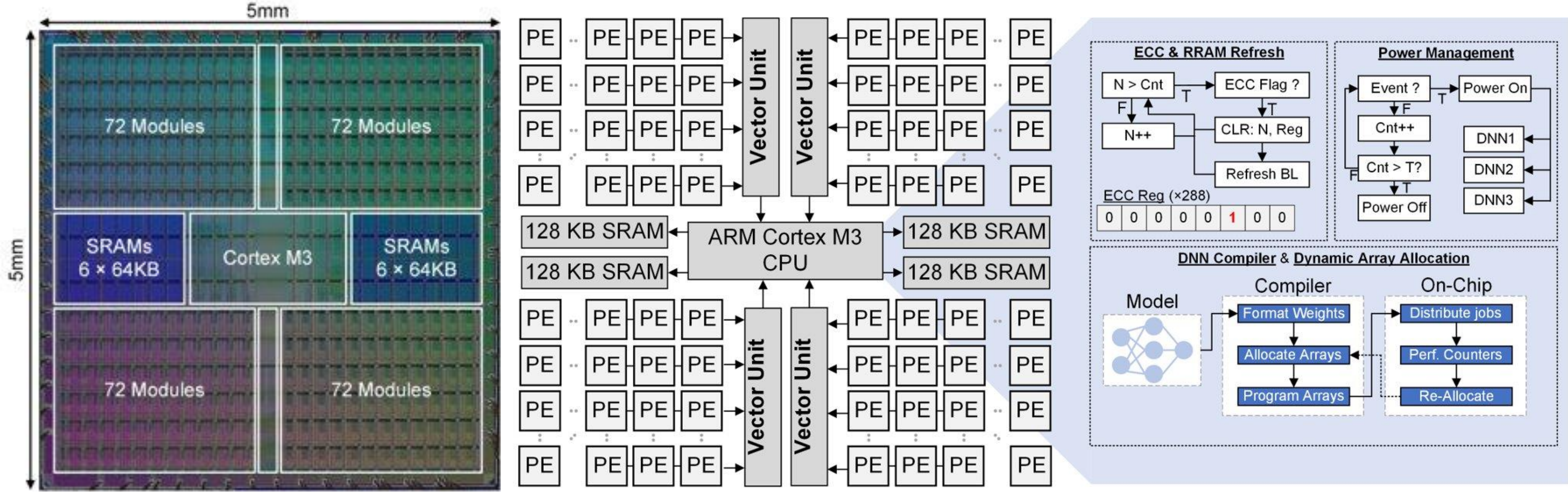




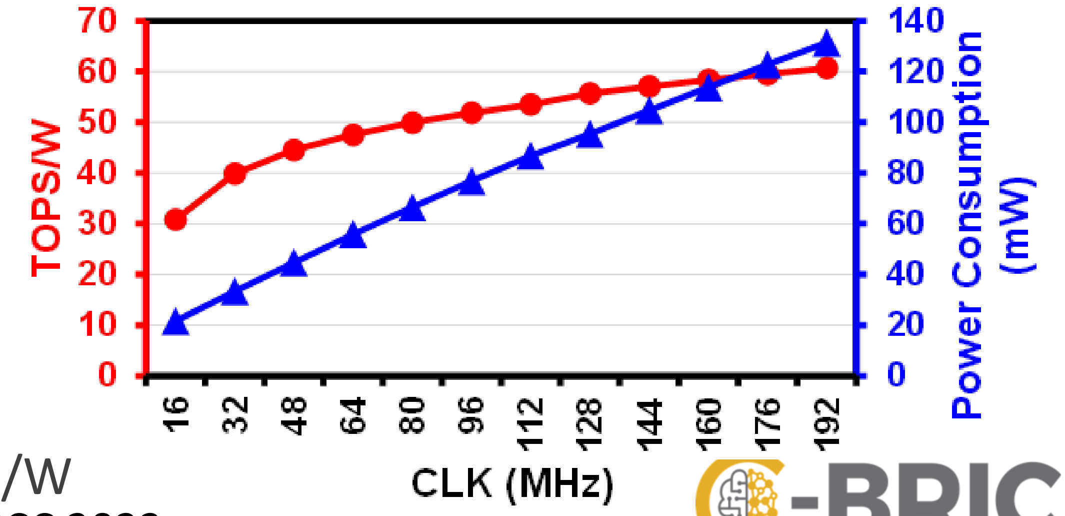
- **Design space exploration of ML kernels**
- Efficiency depends on multiple parameters
  - Workload properties
  - Architecture configuration
  - Runtime Utilization
- **Performance Bottleneck Analysis**
- Runtime characteristics has complex dependency of workload and hardware properties

- CNNs show upto 13.0× reduction (least). **High weight reuse, even at batch-size 1.**
- MLPs show upto 80.1× reduction. **No weight reuse, small models.**
- LSTMs show upto 2446× reduction. **Little Weight reuse, large models (billions of parameters).**

Gen2 N40 RRAM CIM with Embedded Processor



Technology: TSMC 40nm RRAM



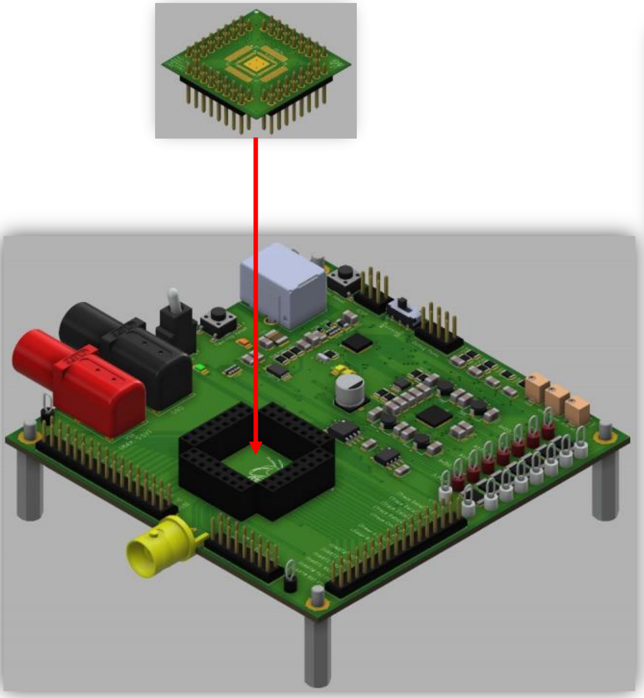
Key Innovation

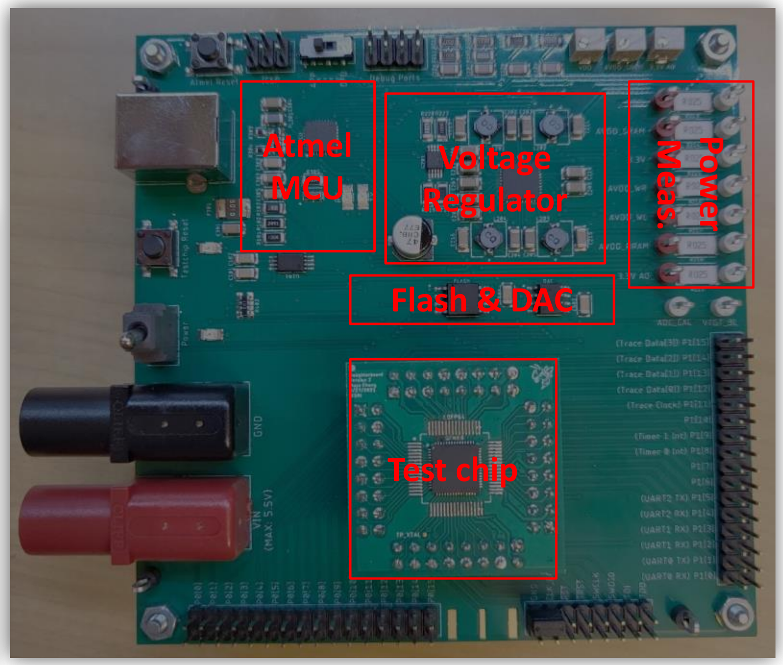
- Full system demonstration with Embedded Cortex M3 processor
- Highest effective RRAM density with >3X improvement of array density w.r.t. SOTA and >50 TOPS/W

Raychowdhury, GaTech, ISSCC 2022

C-BRIC

Evaluation Board



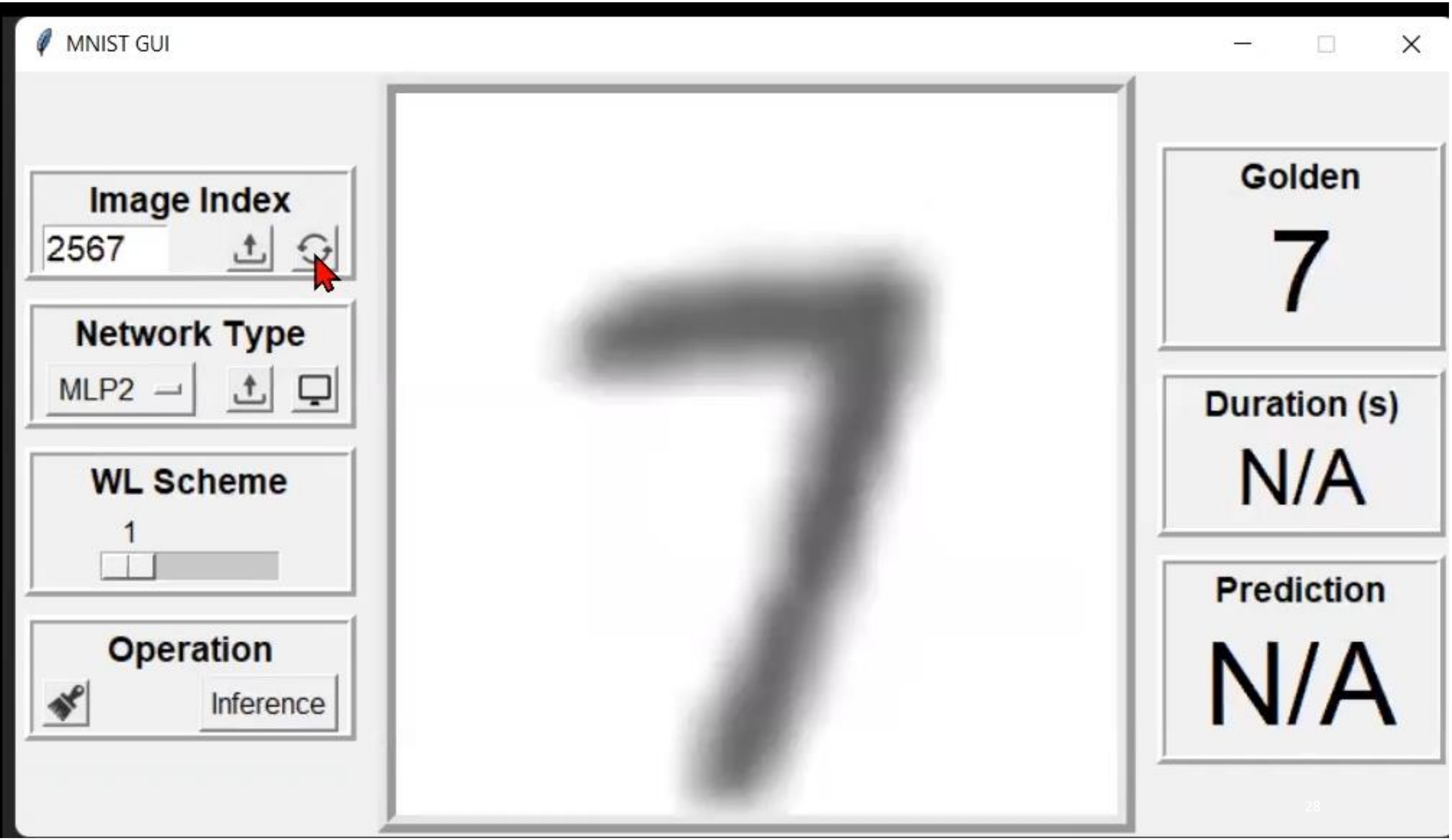


Block	Model
MCU	Atmel Atmega32u2
Voltage Regulator	Analog Device LTC3676
External Flash	Adesto Tech. AT45DB321E
DAC	Burr-Brown DAC7612

https://muyachang.github.io/rram-pyterminal/

- Full python programmability and OS support.
- Currently being used as a test-vehicle for both research and undergraduate teaching
- Planning to share the evaluation board with CBRIC PIs so that we can use this as a test-bed for algorithmic and embedded system research

System Demonstration



MNIST GUI

Image Index

2567

Network Type

MLP2

WL Scheme

1

Golden

7

Duration (s)

N/A

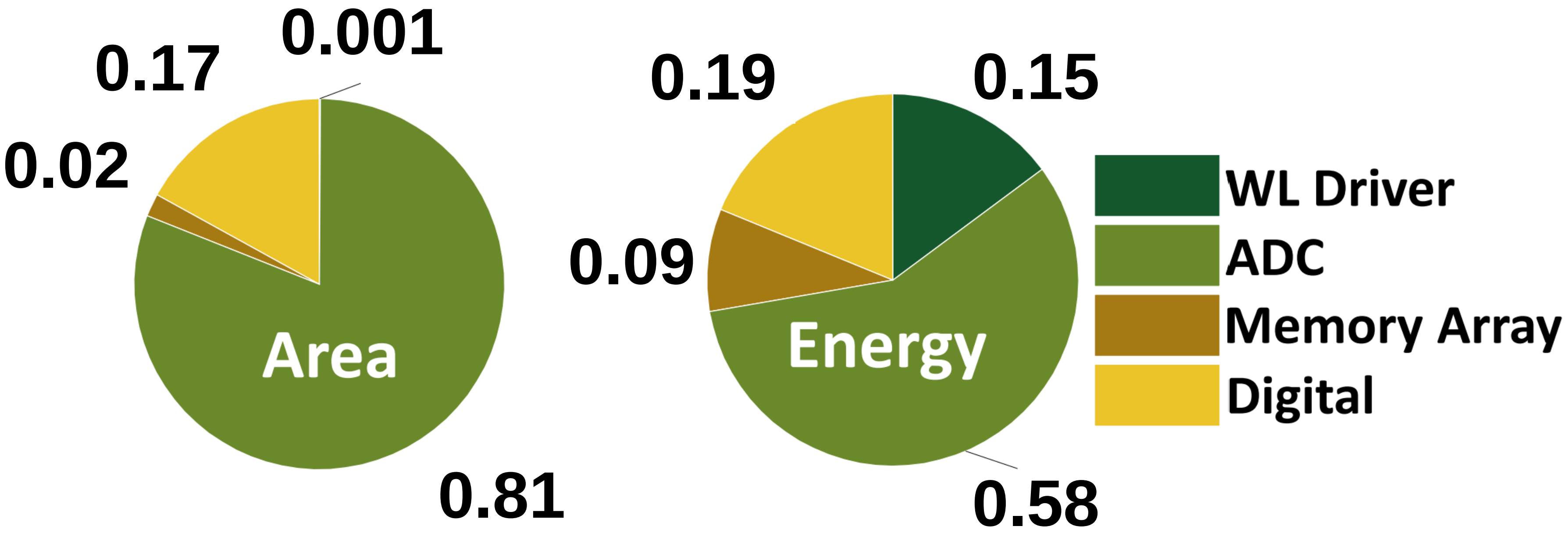
Prediction

N/A

Revisit ADC: Near ADC-less CiM

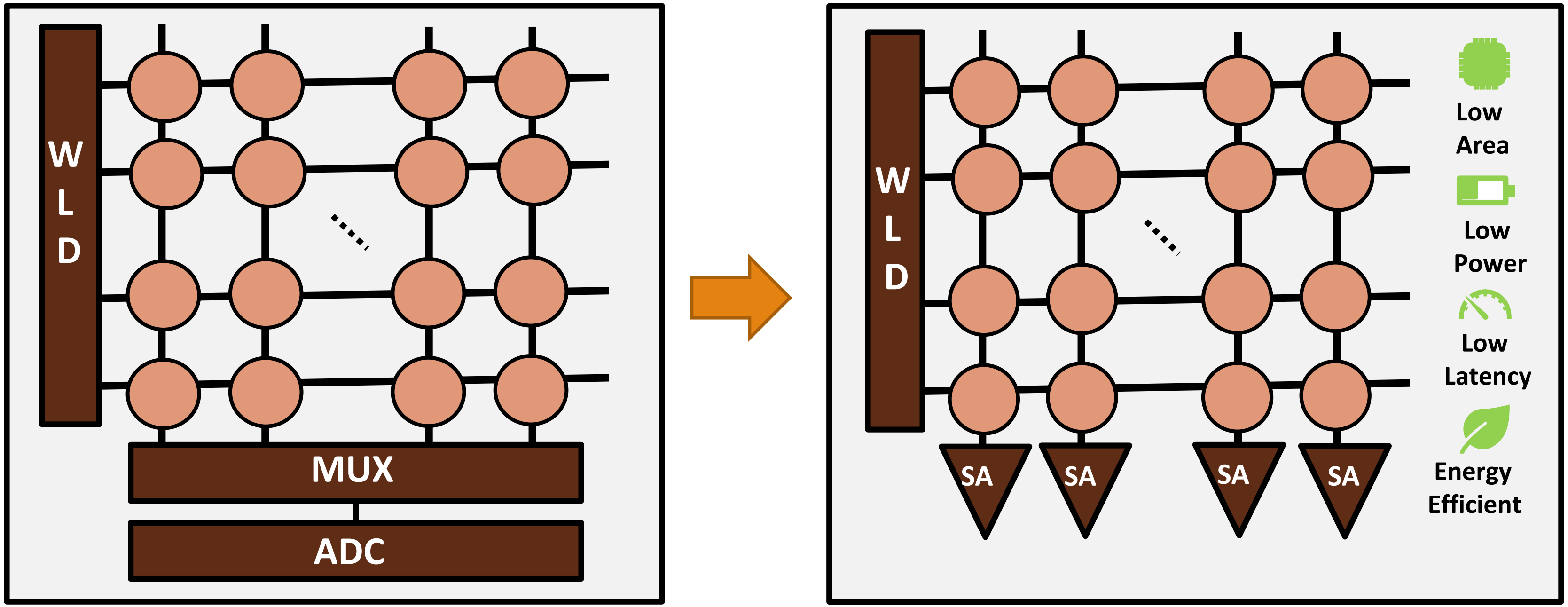
ADC overhead in CiM accelerators

- Large percentage of area and energy profile dominated by ADCs.



Mitigating Overhead with ADC-Less Design

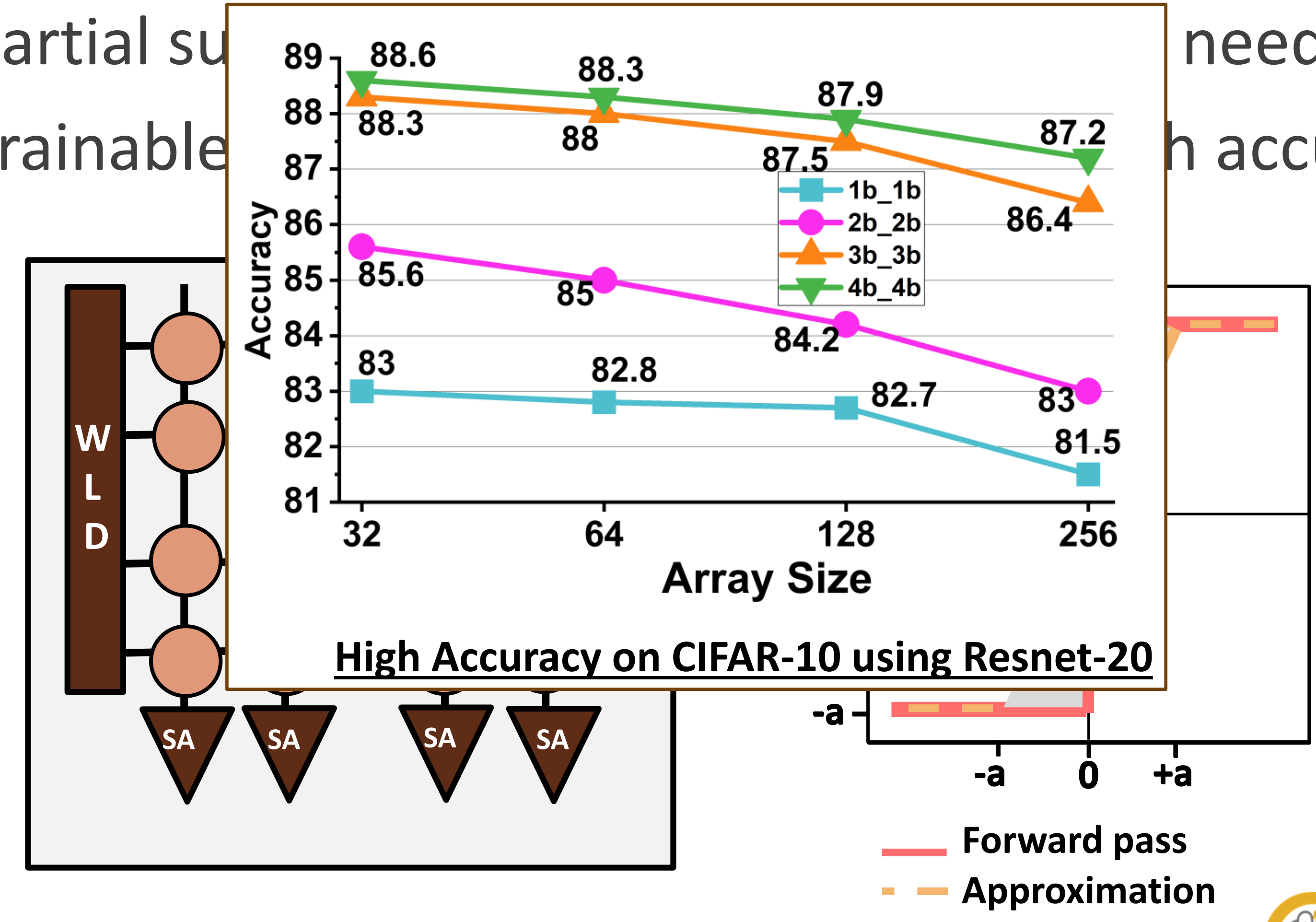
- Area and energy profile dominated by ADCs in CiM accelerators.
- ADC-Less Design: Use Sense Amplifiers for Analog to Digital conversion of Array output



Saxena, Chakraborty, Roy, Towards ADC-Less Compute-In-Memory Accelerators for Energy Efficient Deep Learning, DATE 2022

SW co-design for ADC-Less CiM Accelerators

- ADC-Less CiM accelerators have 1b partial sums.
- Partial su needed.
- Trainable h accuracy.



High Accuracy on CIFAR-10 using Resnet-20

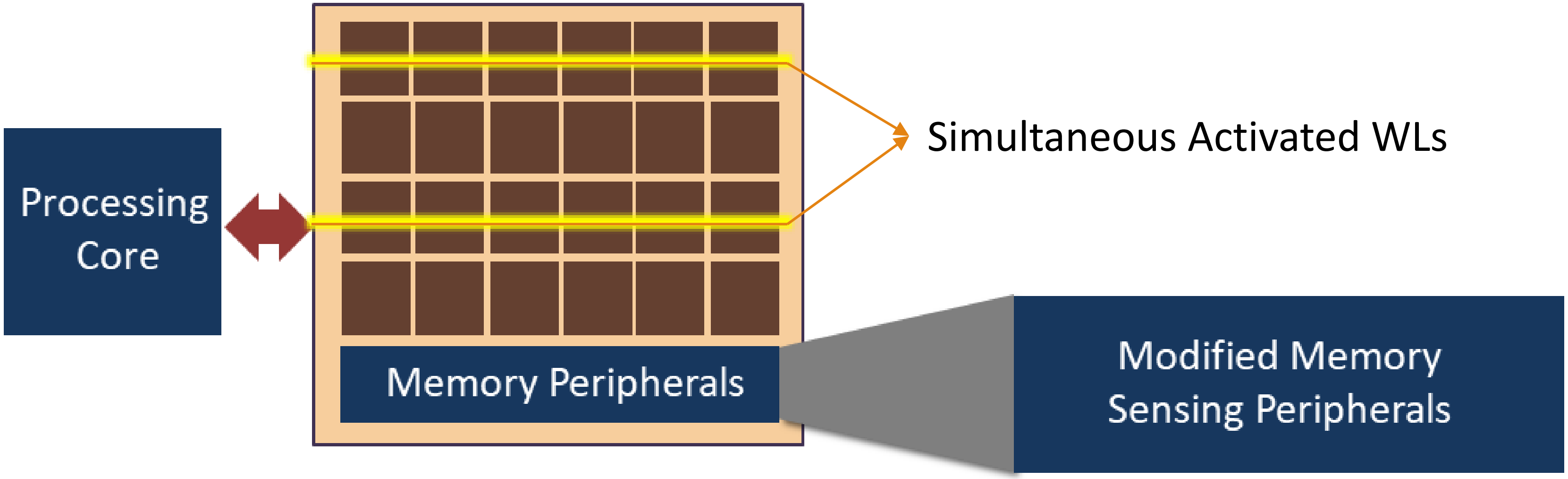
Scale Factor ‘a’ is trained

CMOS SRAM and Non-volatile Memories

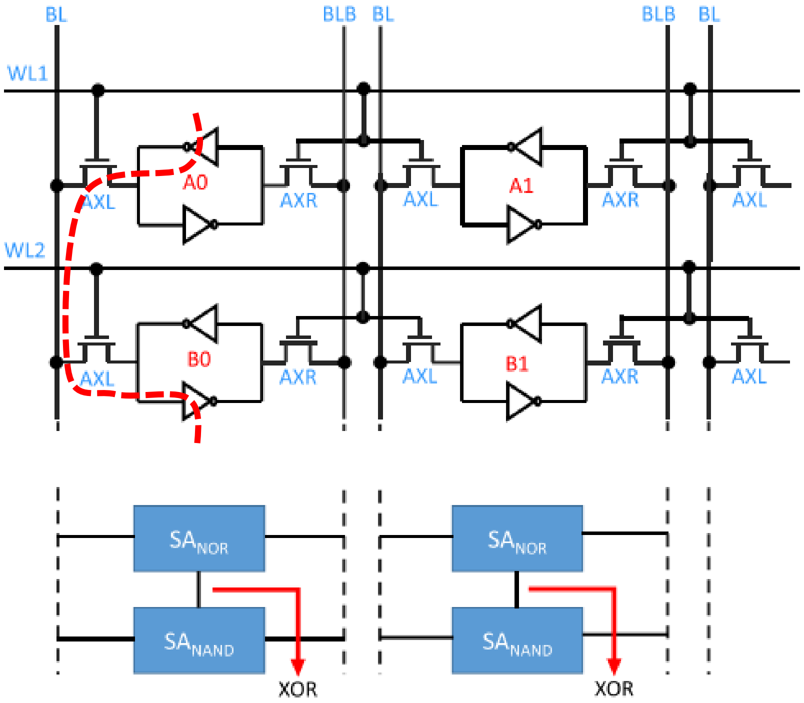
Property	PCM	RRAM	MTJ	CMOS (SRAM)
Multi-level cell	Yes	Yes	No	No
Storage Density	High	High	High	Low
R_{ON}/R_{OFF}	High	High	Low	High
Non-volatility	Yes	Yes	Yes	No
Leakage	Low	Low	Low	High
Cell Area	$16F^2$	$16F^2$	$30\text{-}80F^2$	$160F^2$ (6T), $231F^2$ (8T)
Write Energy	6 nJ	2 nJ	< 1 nJ	< 0.1 nJ
Write Latency	150 ns	100 ns	10 ns	< 1ns
Endurance	10^7 cycles	10^5 cycles	10^{15} cycles	> 10^{16} cycles

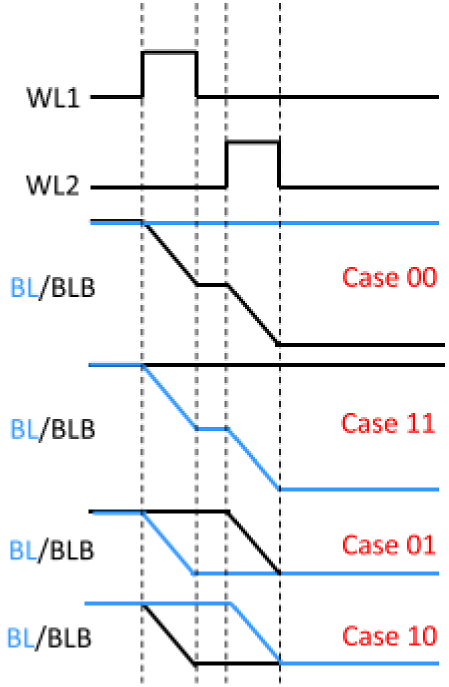
In-Memory Bit-wise Vector Boolean Operations

[Modifying the Peripheral Sensing Circuits to Read-out A Vector Boolean Function]


Processing Core
Memory Peripherals
Simultaneous Activated WLs
Modified Memory Sensing Peripherals

SRAM: In-Memory Bit-wise Boolean Operations



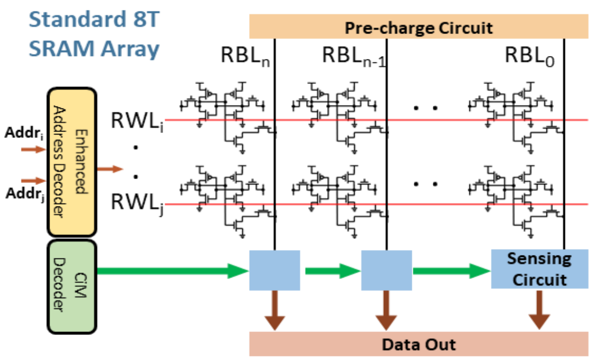


Shanbhag et. al.

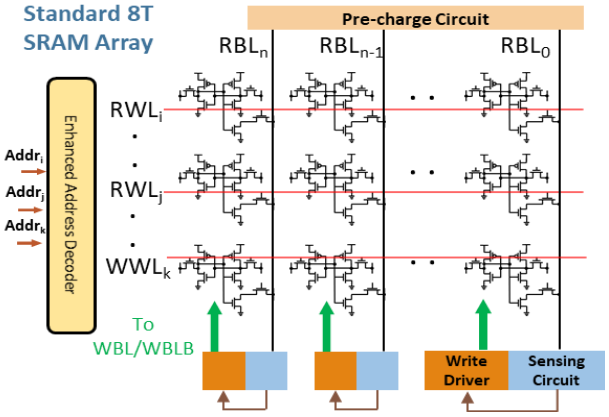
6T-SRAM

- Staggered WL activation to avoid short circuits between cells.
- Asymmetric SAs help detect bitwise NAND/NOR/XORs

X-SRAM: Bit-Wise Vector Boolean Operations

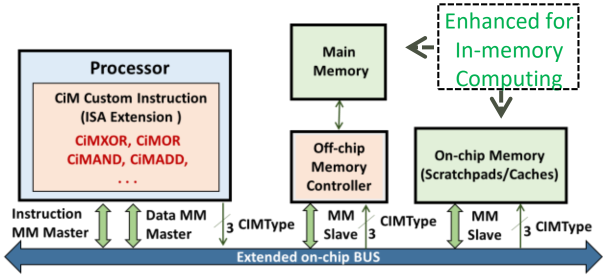


Bitwise: AND/NAND, OR/NOR, XOR, IMP
Arithmetic: ADD/MULT with additional circuitry



Single cycle Read-Compute-Store

System for In-memory Computing



Load R_{ADDR1} R_{DEST0}
Load R_{ADDR2} R_{DEST1}
ADD R_{DEST0} R_{DEST1} R_{OUT}
Store R_{ADDR3} R_{OUT}

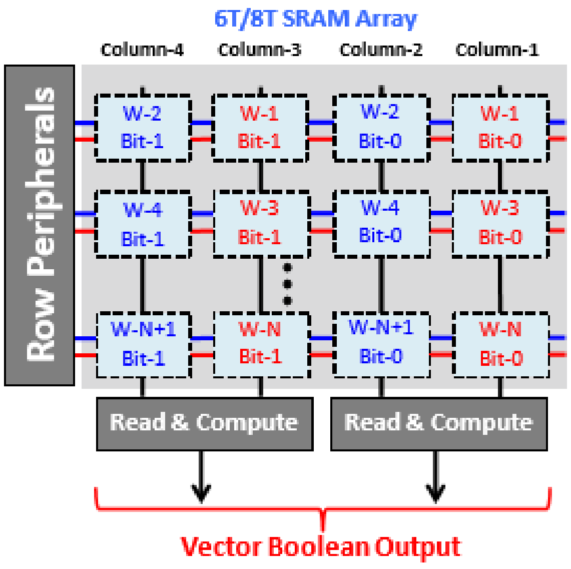
Conventional Instructions

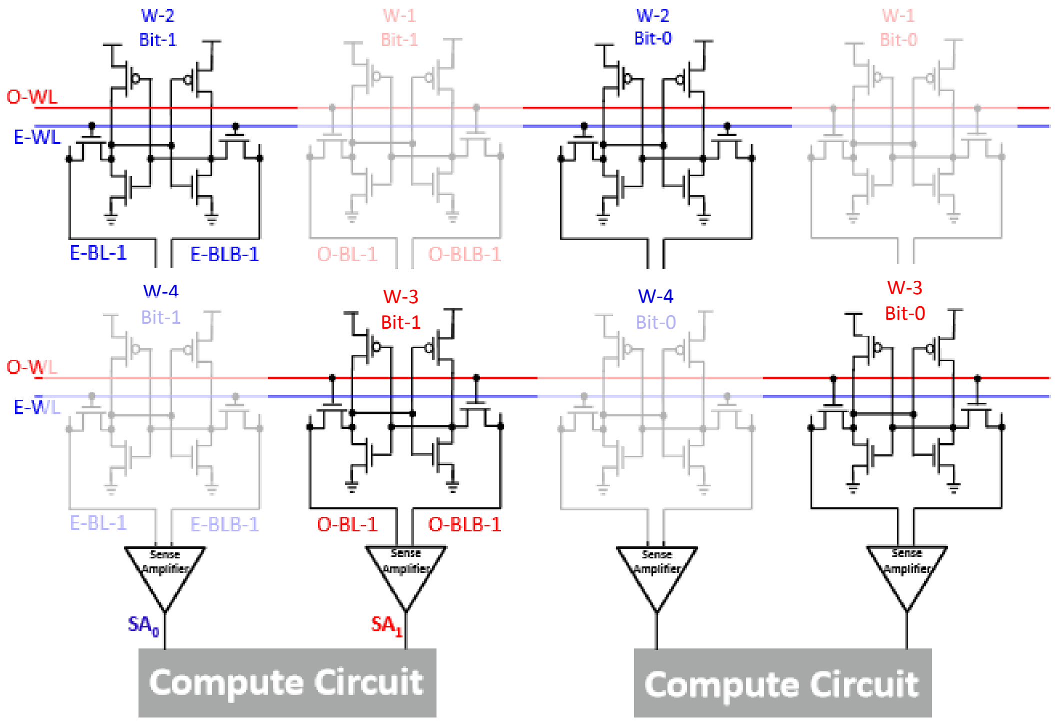
CiMADD R_{ADDR1} R_{ADDR2} R_{ADDR3}

In-memory Instructions

Agrawal, A et al., 2018. X-sram,. IEEE TCAS-I

i-SRAM: Interleaved Wordlines for In-Memory Vector Boolean Operations





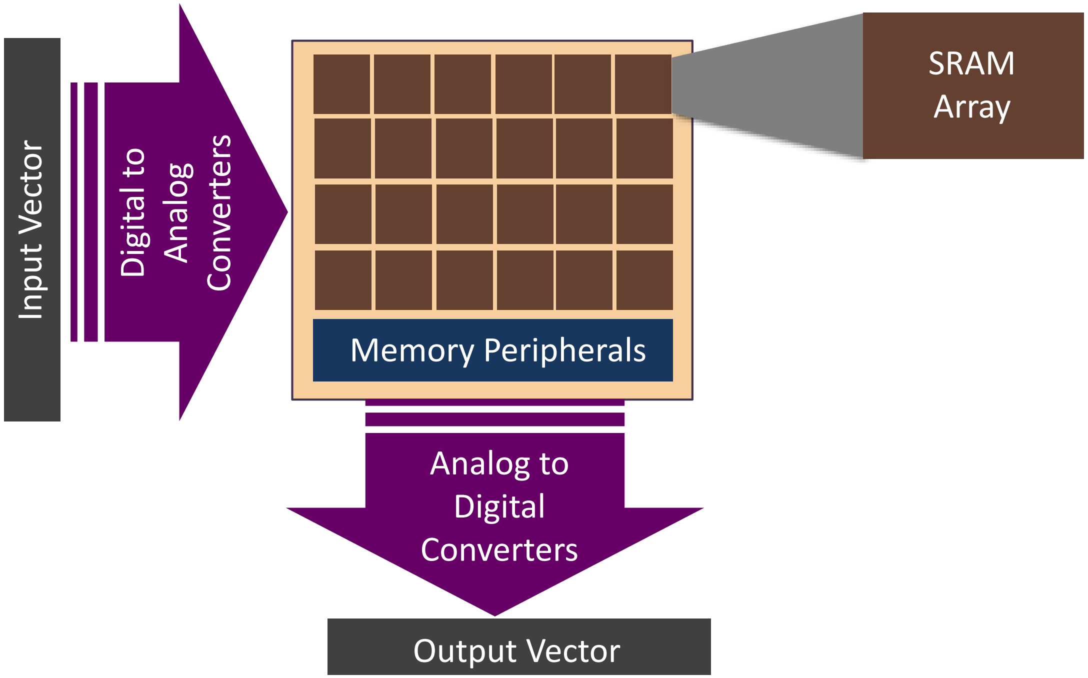
<u>8T-SRAM</u>

- Each row has two read word-lines, and bit-lines are connected to read and compute blocks
- Interleaved 6T cells with bit-lines connected to sense-amplifiers then compute circuits.
- The circuit schematic for NAND(AND)
- The circuit scheme for NOR(OR)

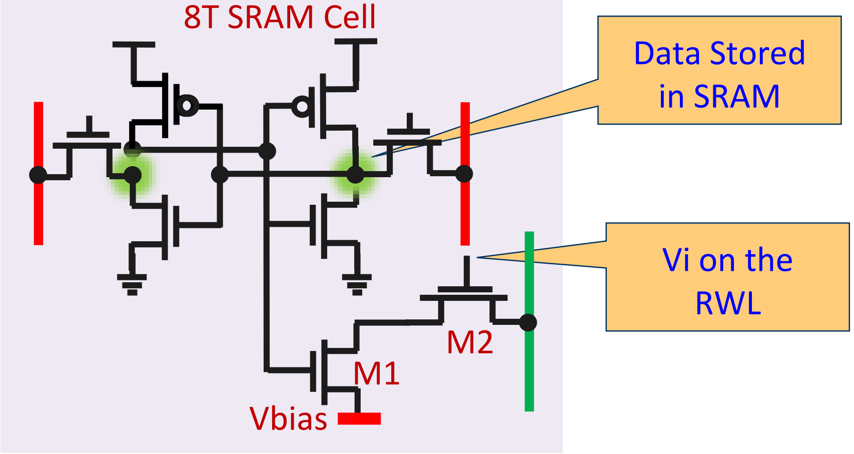
Jaiswal, et al. "i-SRAM: Interleaved Wordlines for Vector Boolean Operations Using SRAMs." *IEEE Trans. CAS I:* (2020).

In-memory Dot Product Computations

[In-Memory Dot Product Acceleration by use of Current-Mode Computations in SRAMs]



8T SRAM as a Multi-Bit Dot Product Engine



Data Stored in SRAM

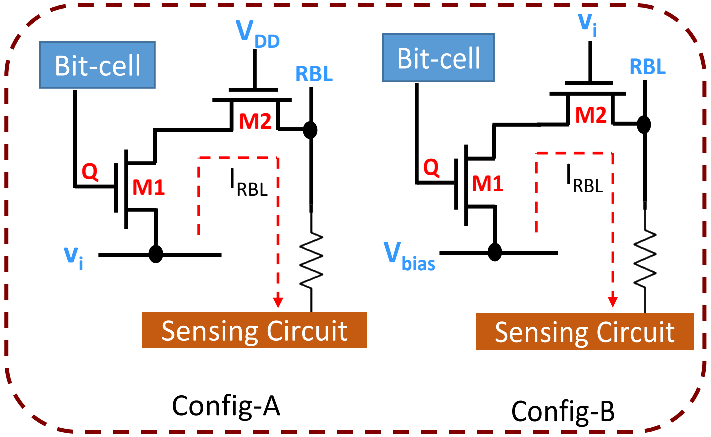
Vi on the RWL

The Dot Product = $\Sigma V_i . W_i$

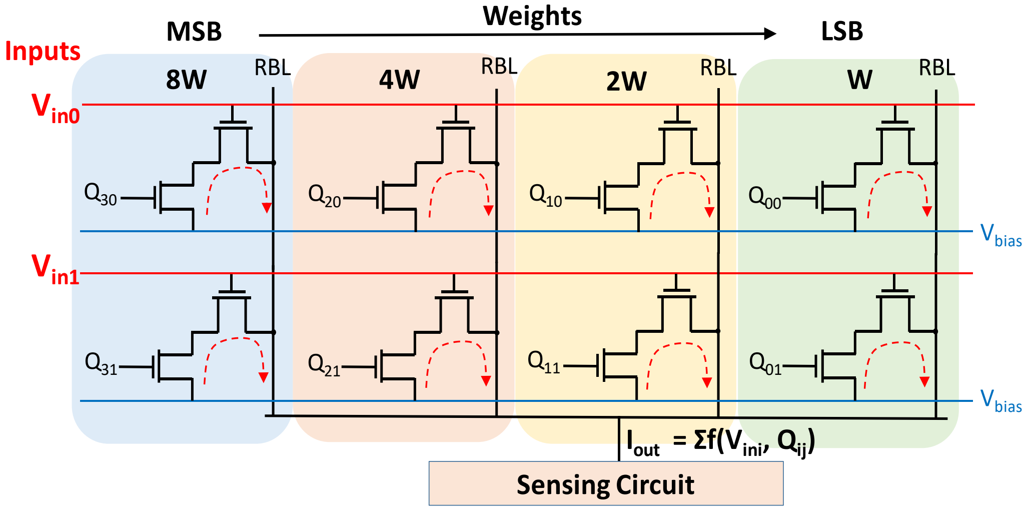
Vi: Analog Voltages on RWL or SL

Wi: Data Stored in SRAM

Summation: KCL addition

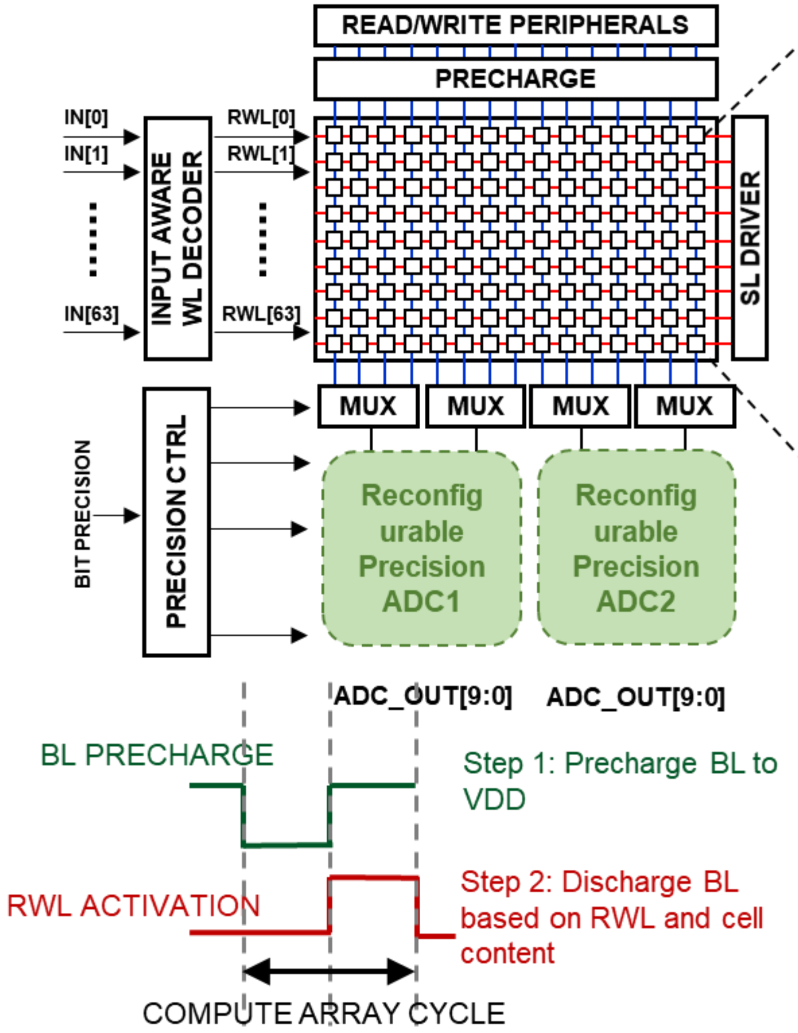


Modified Read Port



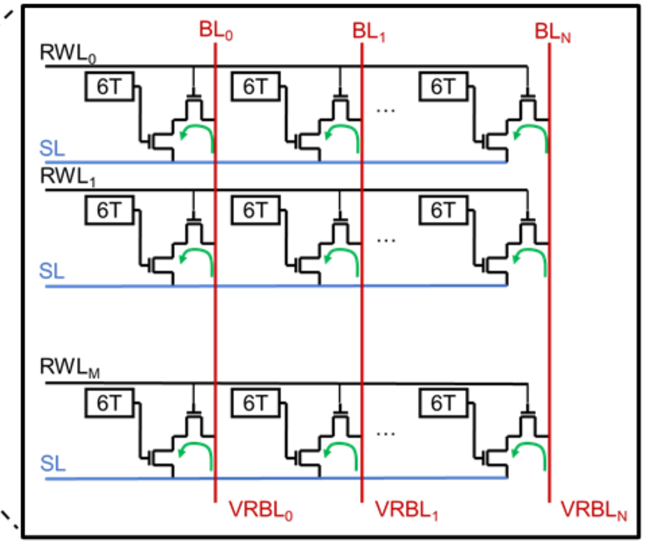
A. Jaiswal, I. Chakraborty et al , under review in TVLSI, 2018

CiM Macro



Step 1: Precharge BL to VDD

Step 2: Discharge BL based on RWL and cell content

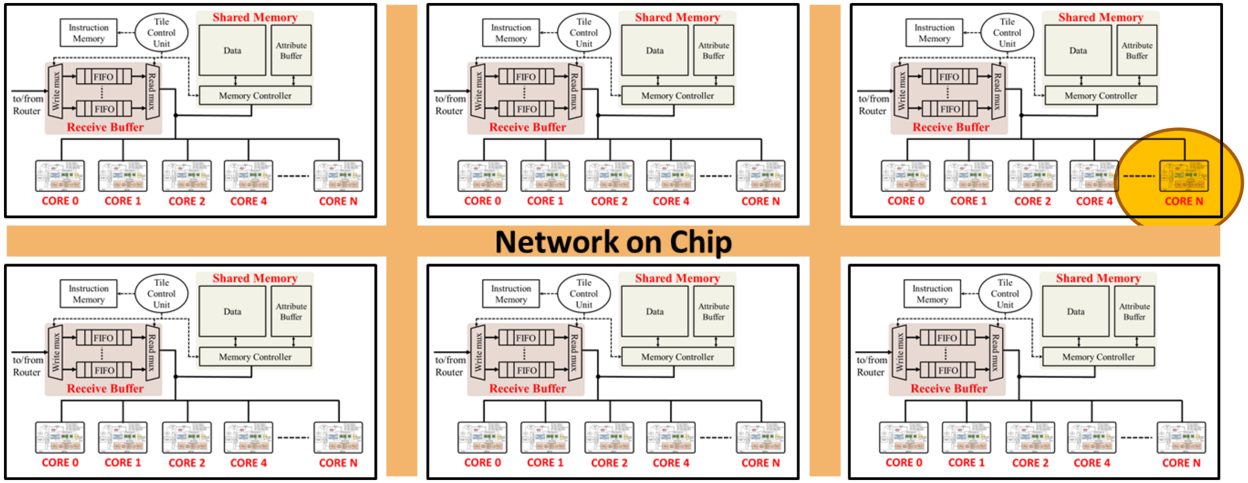


1b IN -1b W compute logic

RWL	Q	Current
0	0	No BL Discharge
0	1	No BL Discharge
1	0	No BL Discharge
1	1	BL Discharge

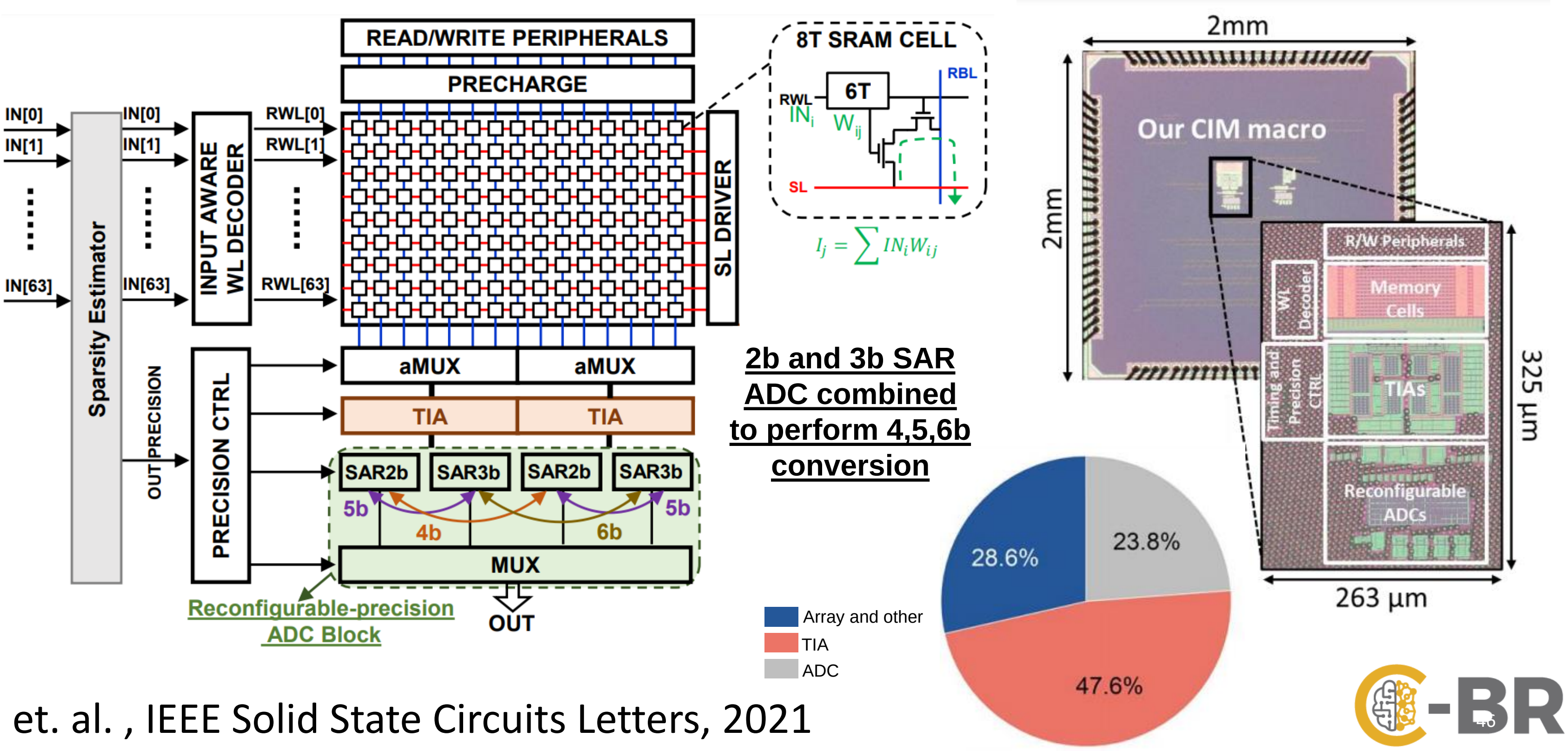
- More the number of ones in input or weights in a column, more discharge of BL.
- V_{RBL} proportional to MAC output between inputs and weights.

IMC CMOS Cores (ANNs)



CMOS Array: Sparsity Aware CiM Macro design

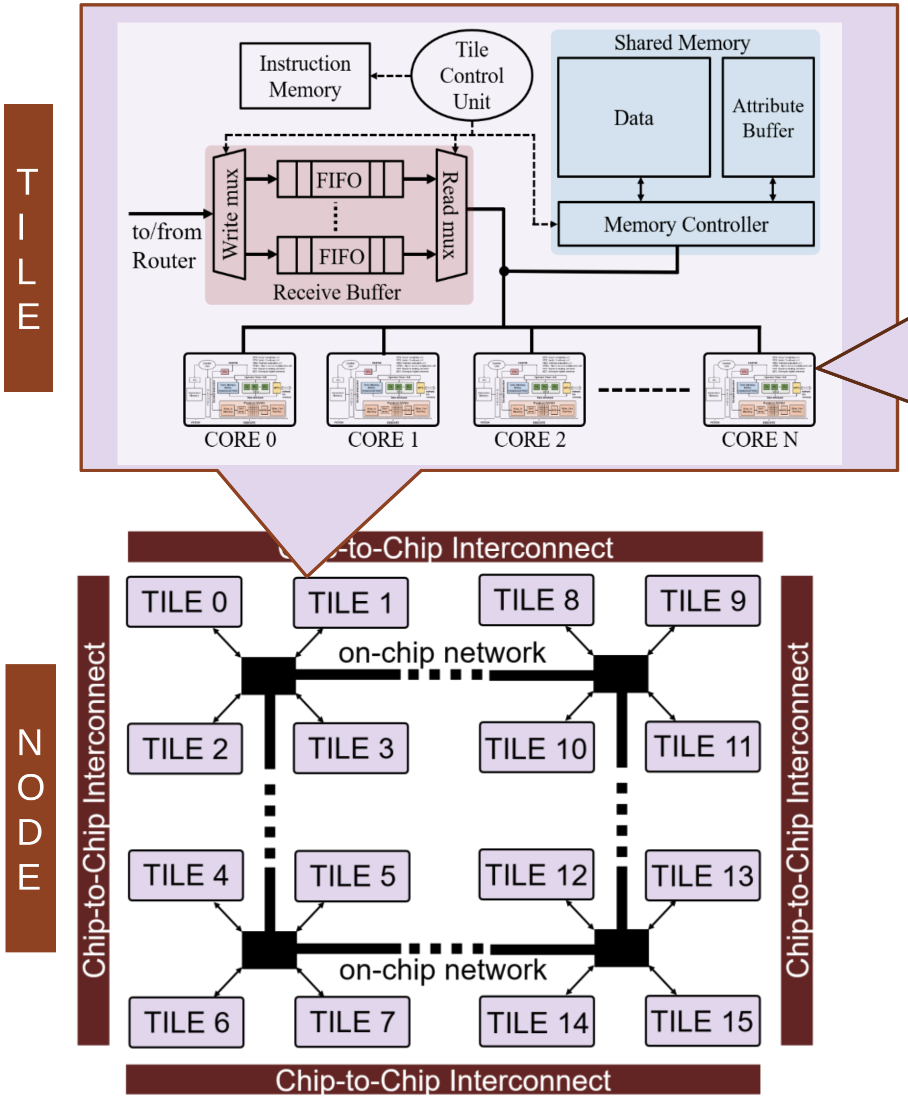
- Bit serial CiM acceleration provides opportunities to leverage abundant bit level sparsity.
- Different levels of input sparsity can be leveraged by dynamically reconfiguring ADC precision (2b-6b).

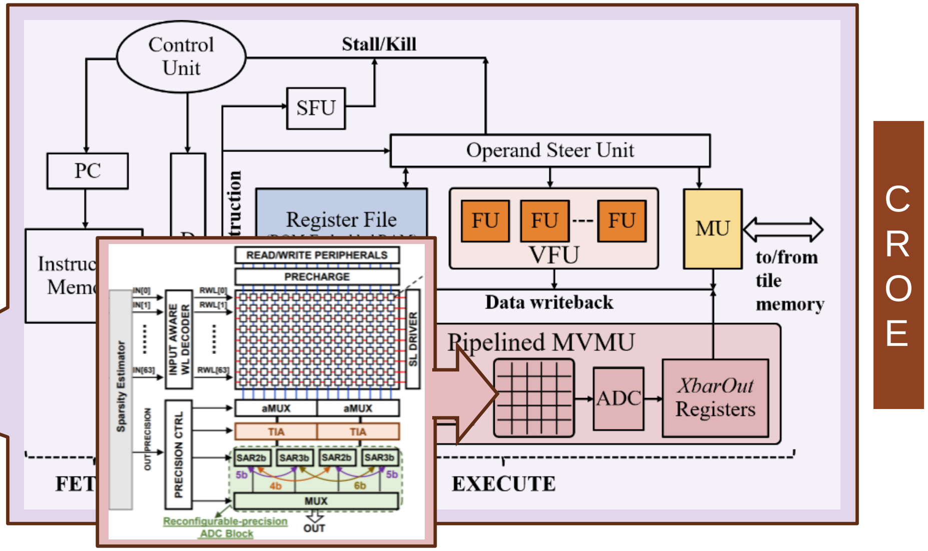


2b and 3b SAR ADC combined to perform 4,5,6b conversion

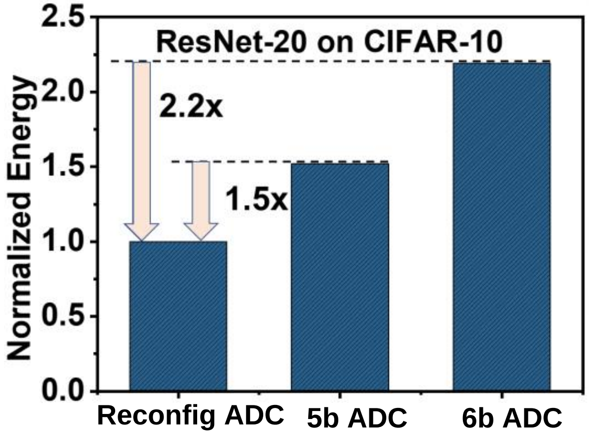
A. Ali et. al. , IEEE Solid State Circuits Letters, 2021

System with sparsity aware Core



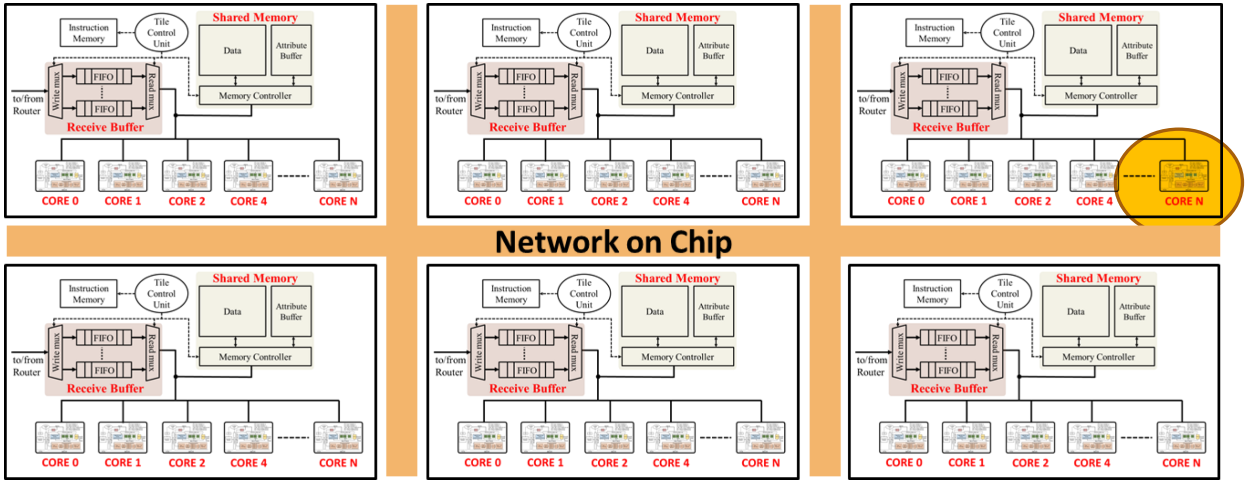


Sparsity aware macro



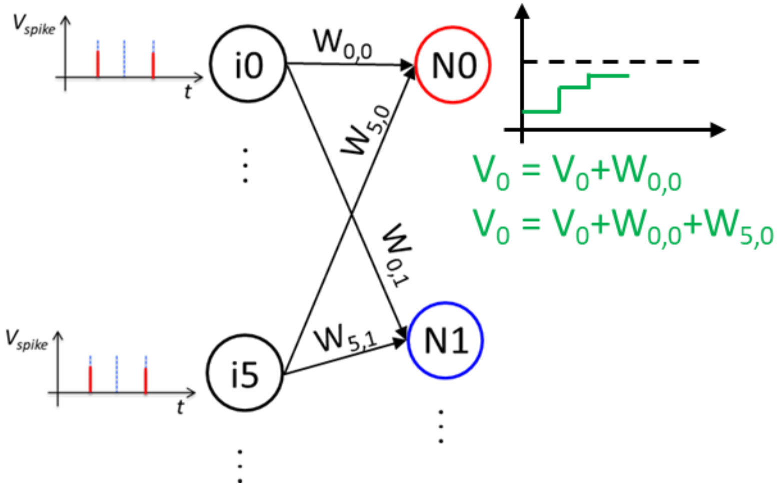
2.2X improvement in inference energy

IMC CMOS Cores (SNNs)



SNNs

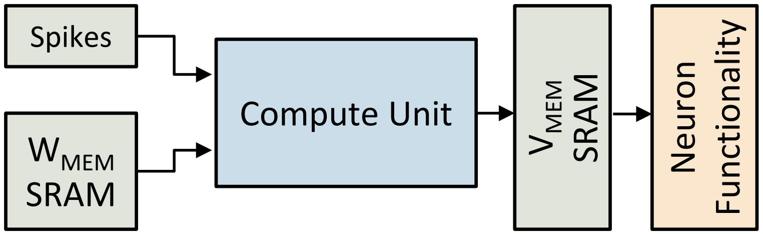
- Challenges in current digital SNN hardware:
  - Data transfer bottleneck to/from memory.
  - Additional SNN-specific data movements for processing V_{MEM} for multiple timesteps.
  - Limited functionality due to area and power expensive custom neuron circuitry.
- Approach:
  - ✓ Fused W_{MEM} and V_{MEM} CIM Array integrating all processing modes required for SNN inference – accumulate, threshold, reset etc.



$V_0 = V_0+W_{0,0}$

$V_0 = V_0+W_{0,0}+W_{5,0}$

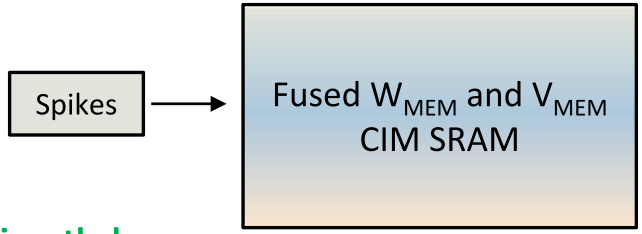
Conventional Approach:



Data transfer bottleneck

Limited functionality due to custom neuron circuitry

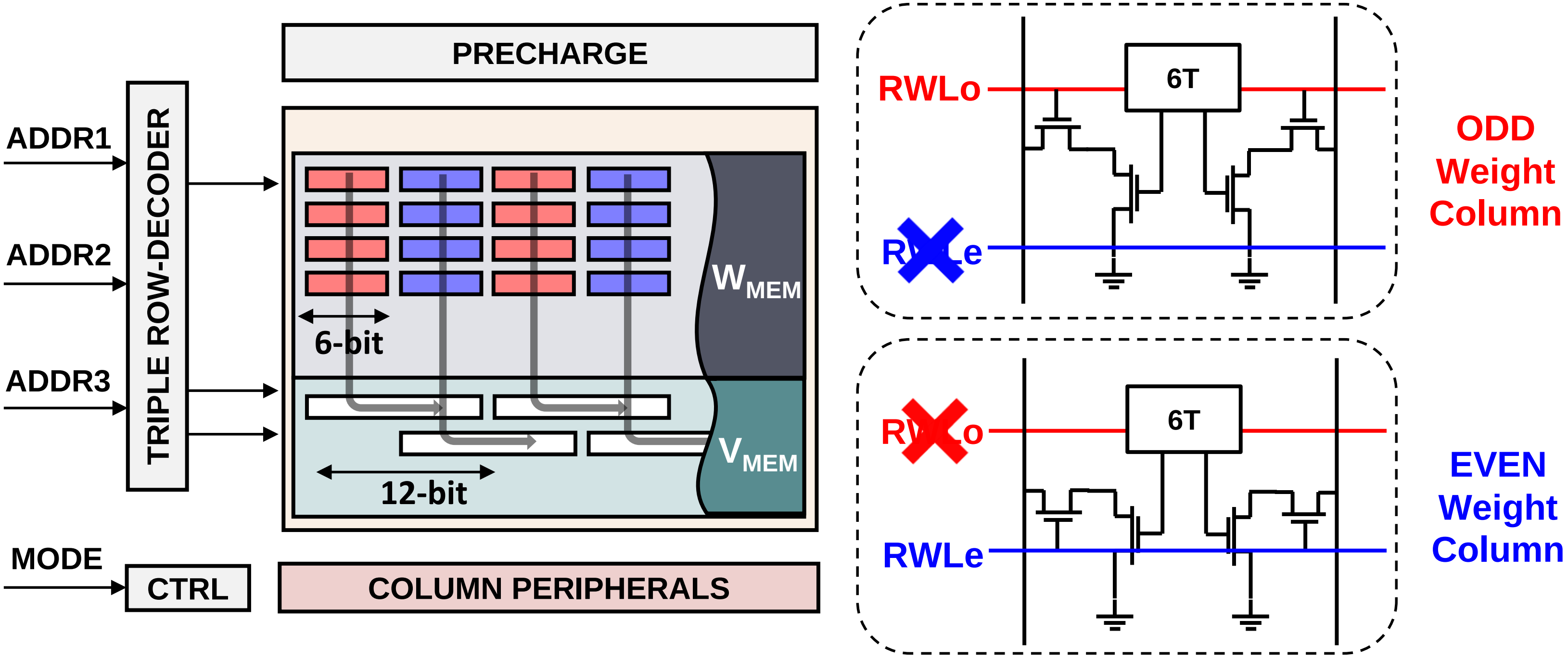
Proposed Approach:



Significantly lower data transfer due to CIM

Embedded neuron functionality

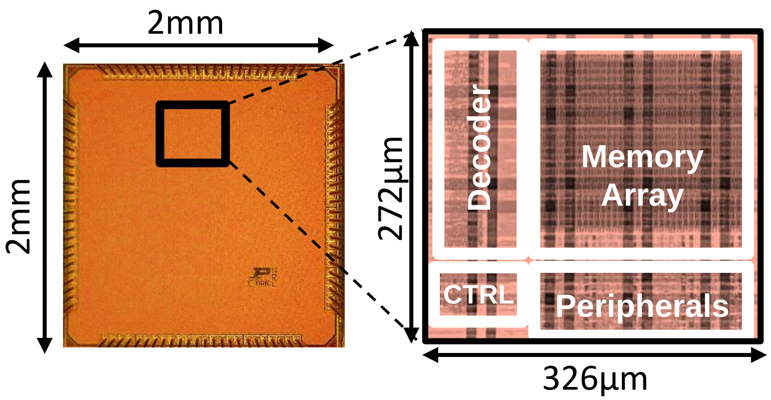
Organization of the Fused W_{MEM}/V_{MEM} Array



- Different bit-precision requirements for weights and V_{MEM}.
- We propose a mapping strategy to efficiently use the SRAM area with minimal peripheral complexity:
  - Fit more weights in each row using odd/even RWLs.
  - Staggered alignment of corresponding V_{MEM} data.
  - Same peripherals are used by reconfiguring them in odd/even cycles.

In-memory Computing Macro for SNNs: Chip Summary

- We propose a 10T SRAM based CIM macro for SNN inference.
- The macro consists of a fused W_{MEM} and V_{MEM} and supports all processing modes required for SNN inference - accumulate, threshold, spike-check, reset etc.
- The macro also supports multiple neuron functionalities through various instruction sequences.
- The macro also leverages sparsity in the input spikes for energy-efficiency.
- The prototype chip was fabricated in 65nm LP CMOS process, achieves an energy-efficiency of 0.99 TOPS/W @ 0.85V, 200MHz, for signed 11-bit operations.
- We demonstrate sentiment classification using the intrinsic dynamics of SNNs achieving competitive accuracies with LSTMs.



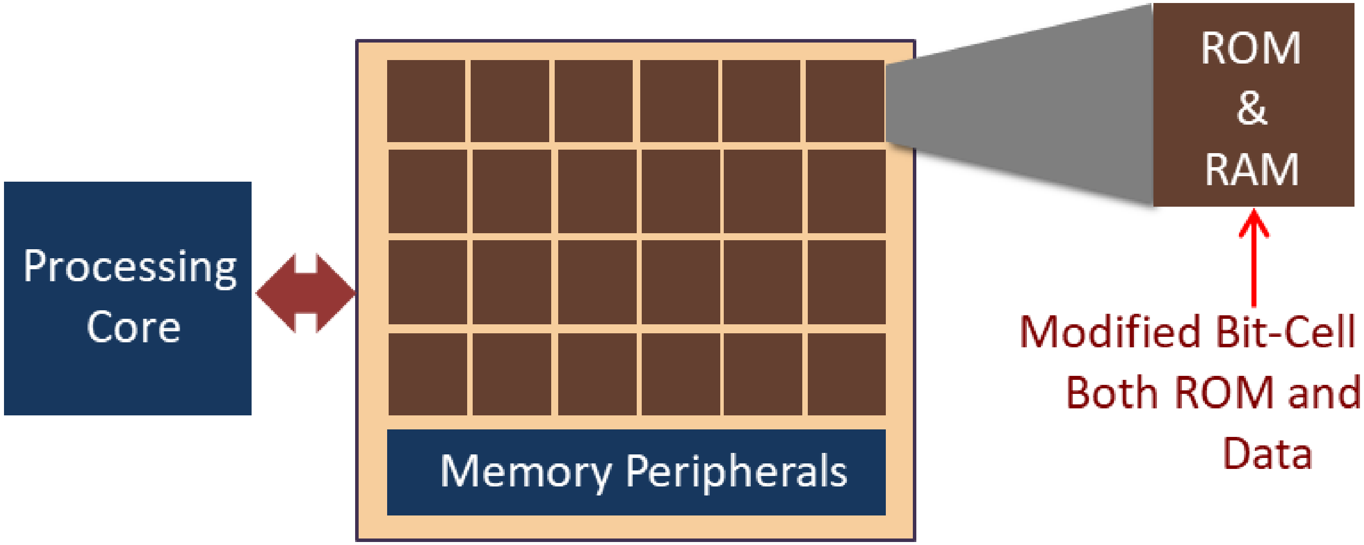
Technology	**65nm**
Macro Area	**0.089 mm^2**
Cell Type	**10T**
Memory	**W_{MEM} : 9kb** **V_{MEM} : 2.25kb**
Weight/Vmem bits	**6-bit/11-bit**
Supply Voltage	**0.7 ~ 1.2 V**
Max. Frequency	**500 MHz**
Energy Efficiency	**0.99 (TOPS/W)** @200MHz, 0.85V (signed 11-bit op)

A. Agrawal et. al. , “IMPULSE….”, IEEE Solid State Circuits Letters, 2021

Embedding ROM in a RAM

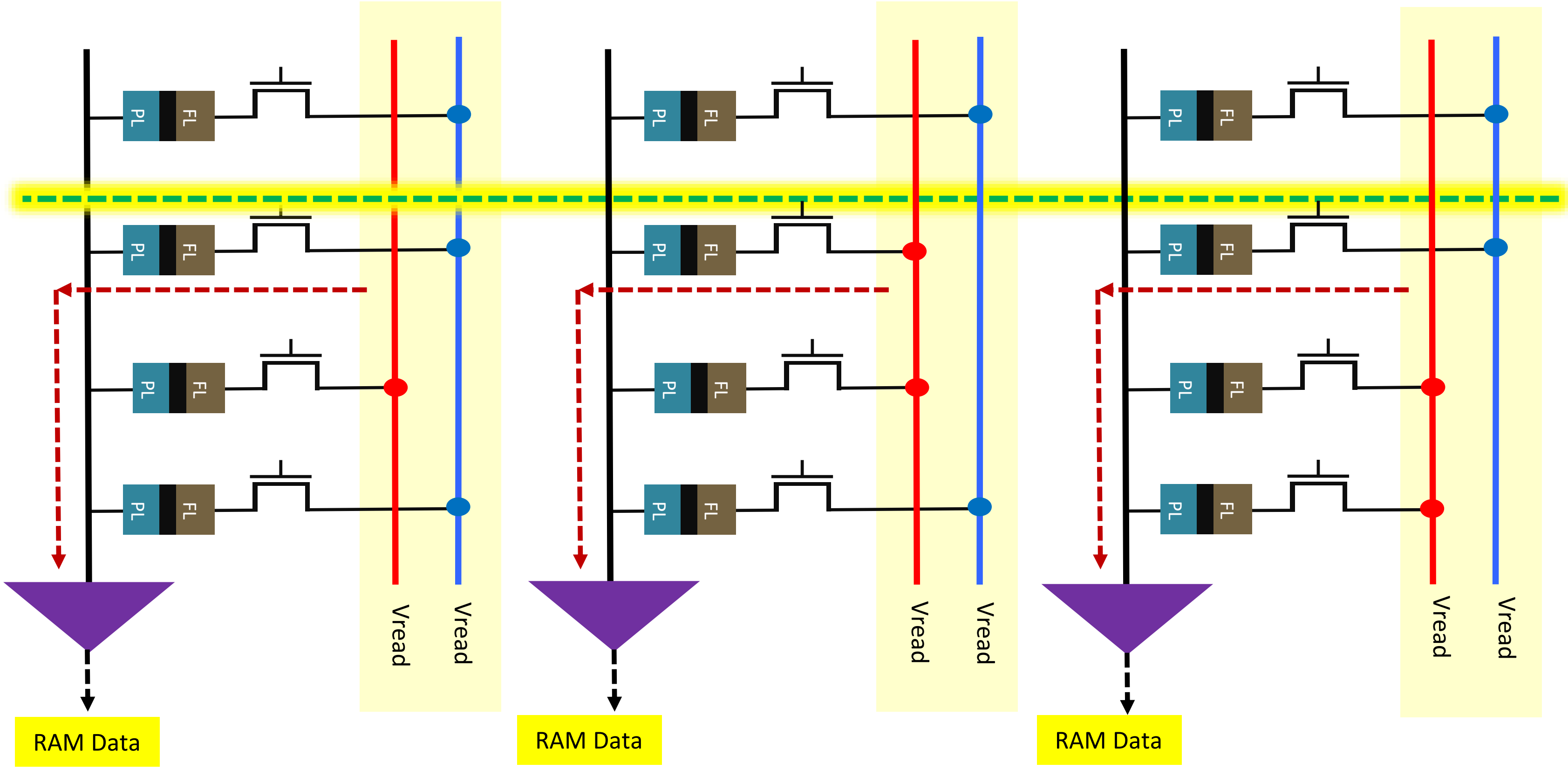
ROM Embedded RAM

[Embedding ROM in CMOS and 1T-1R Arrays Enabling Near-Memory Computing through Lookup Tables]



Modified Bit-Cell Storing Both ROM and RAM Data

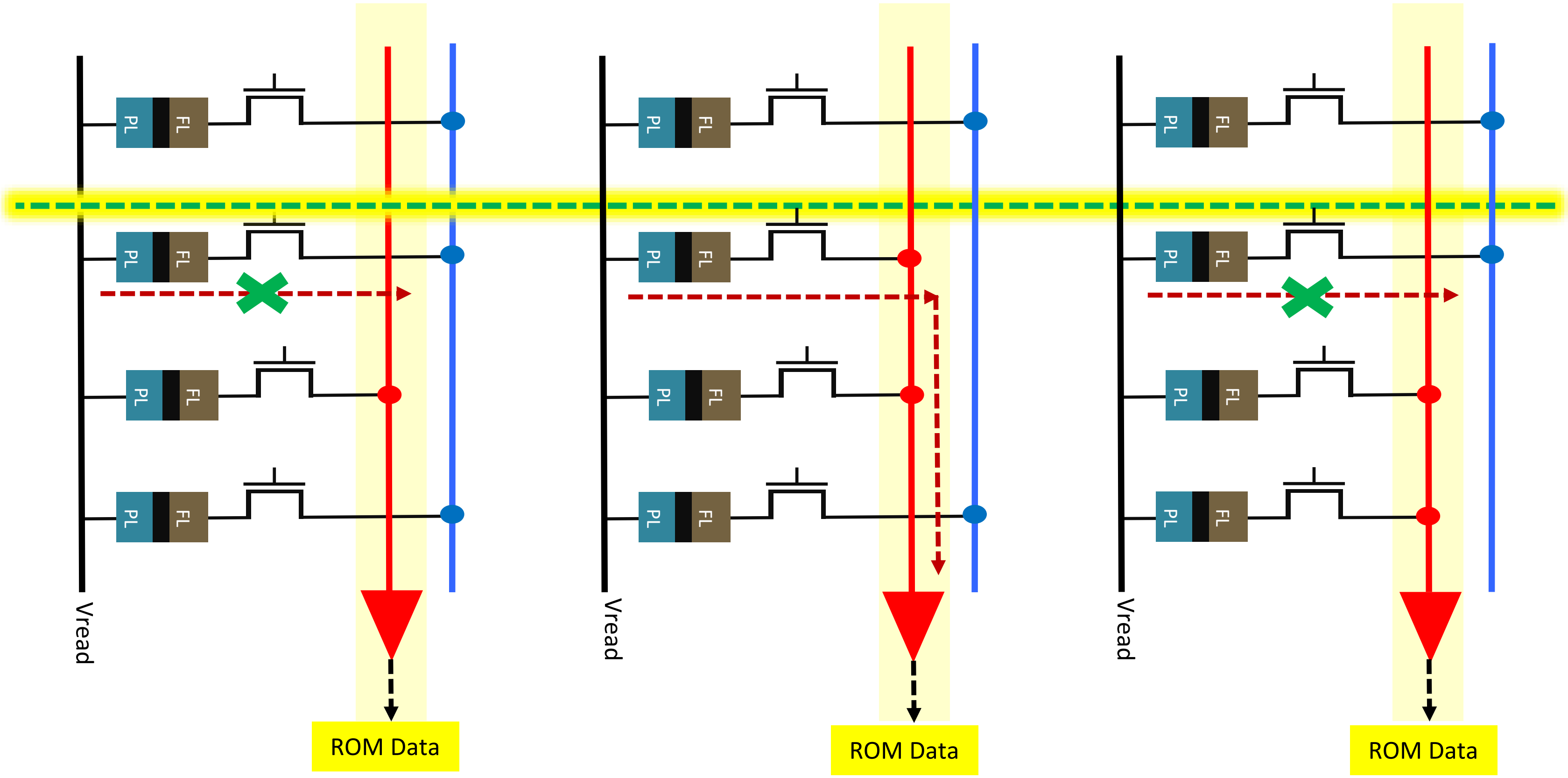
Embedding ROMs in RAMs (NVMs)



Both ROM and RAM data are stored in the same bit-cell. The read cycle determines whether the ROM or the RAM data is being read.

D. Lee, K. Roy, IEEE EDL 2013

Embedding ROMs in RAMs (STT-MRAM)



D. Lee, K. Roy, IEEE EDL 2013

Computing-In-DRAM

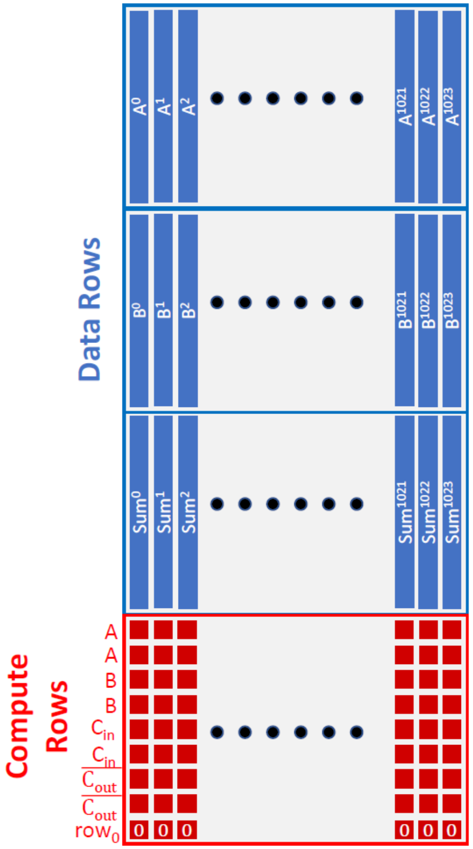
In-DRAM Low-cost Bit-serial Addition

- **Majority-based vector addition**

$$C_{out} = Maj(A, B, C_{in})$$

$$S = Maj(A, B, C_{in}, \overline{C_{out}}, \overline{C_{out}})$$

- **Store data vectors in column-based fashion**
- **Same subarray peripheral circuits**
- **Add 9 reserved rows for compute (<1% area overhead)**

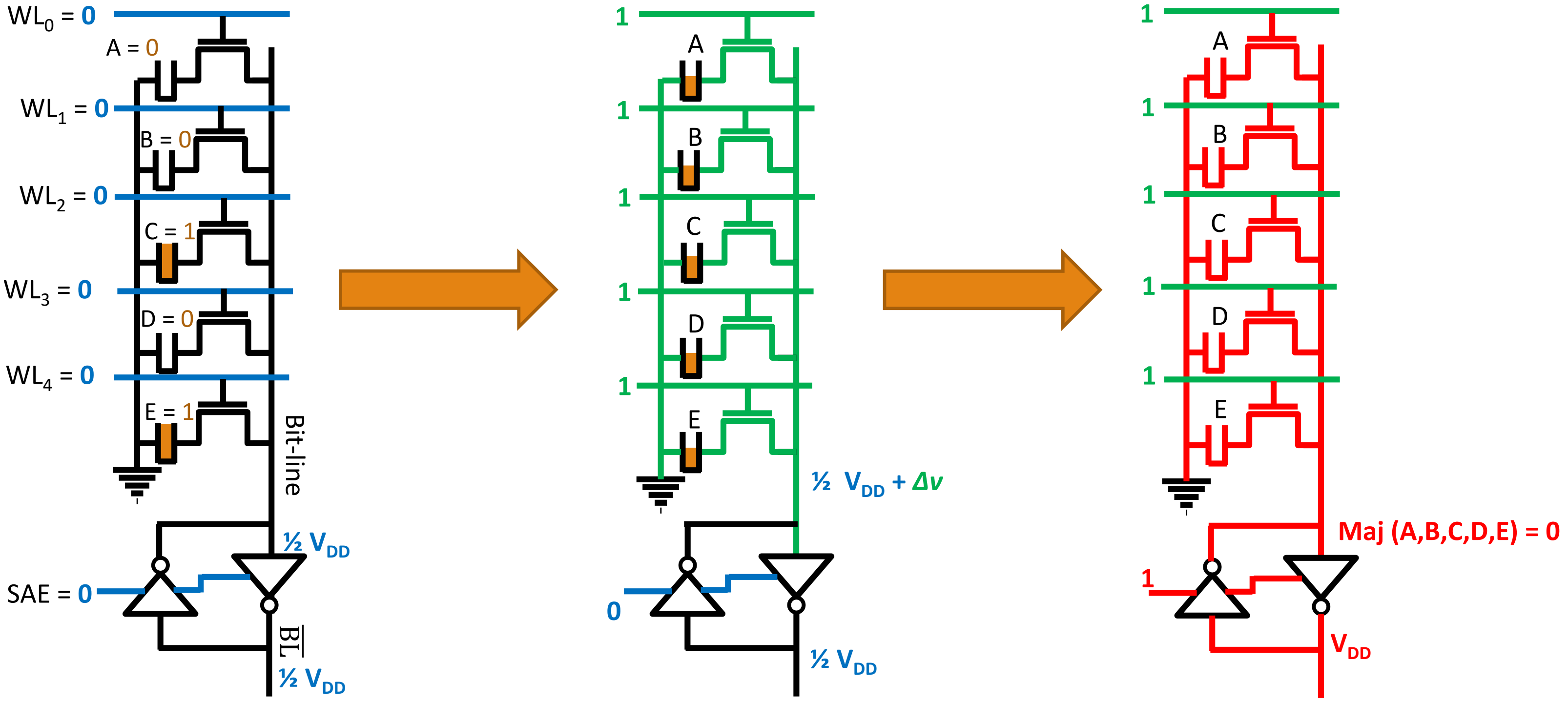


- **The result of A+B addition is stored in the same column**
- **Massively-parallel vector additions in bit-serial mode**
- **No need for carry shifts across bitlines!**

Ali, Jaiswal, Roy, “In-Memory Low-Cost Bit-Serial Addition Using Commodity DRAM Technology,” TCAS-I, 2019

In-DRAM Low-cost Bit-serial Addition

Multiple row activation to calculate Maj(A,B,C,D,E)

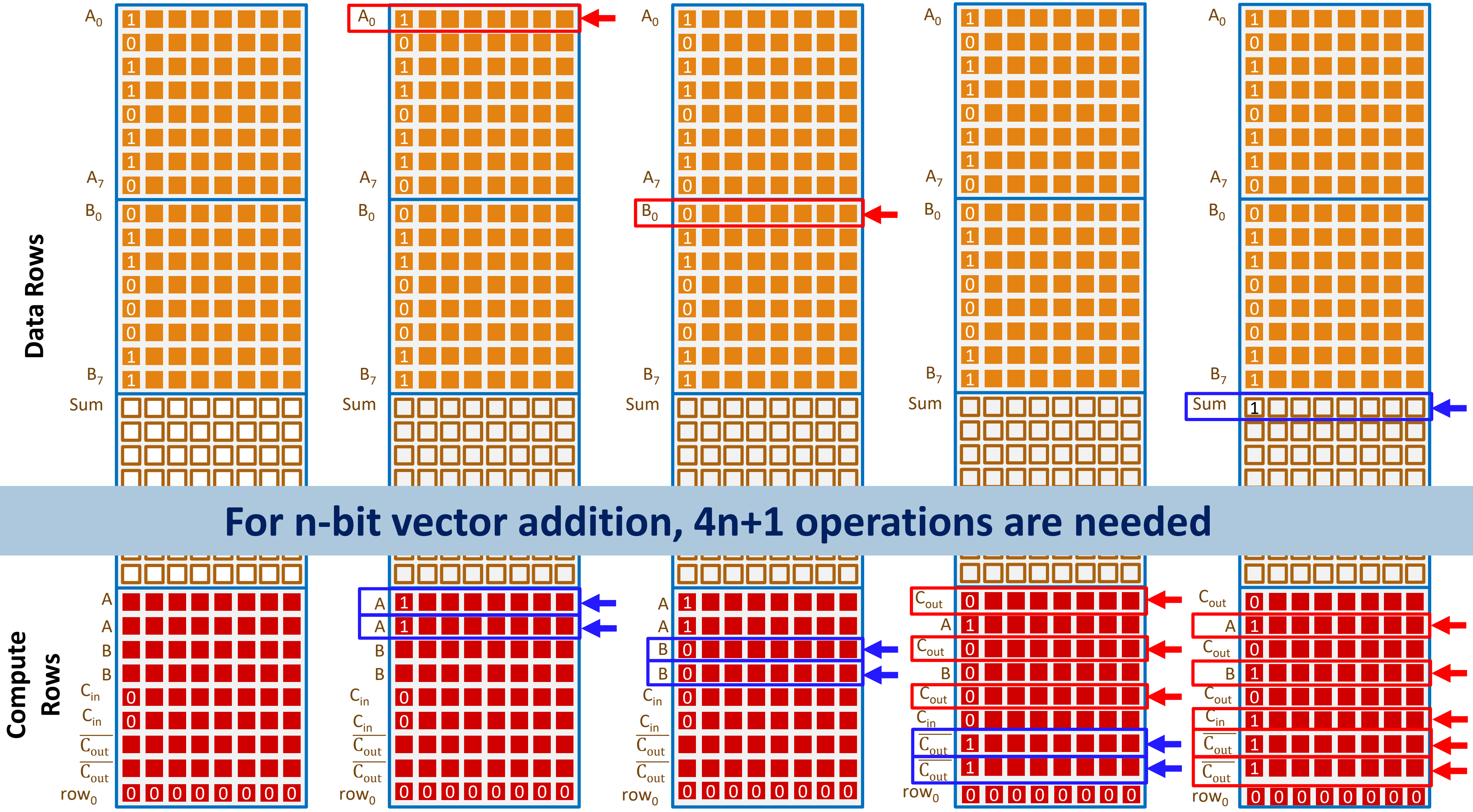


(1) Initial state

(2) Enable WLs

(3) Enable Sense amp

An example of 1-bit addition of two vectors A and B



(0) Initial state

(1) Copy A

(2) Copy B

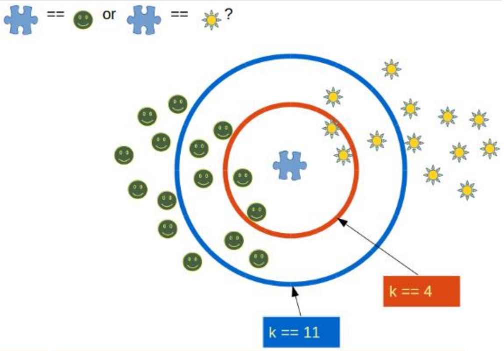
(3) Calculate C_{out}/$\overline{C_{out}}$

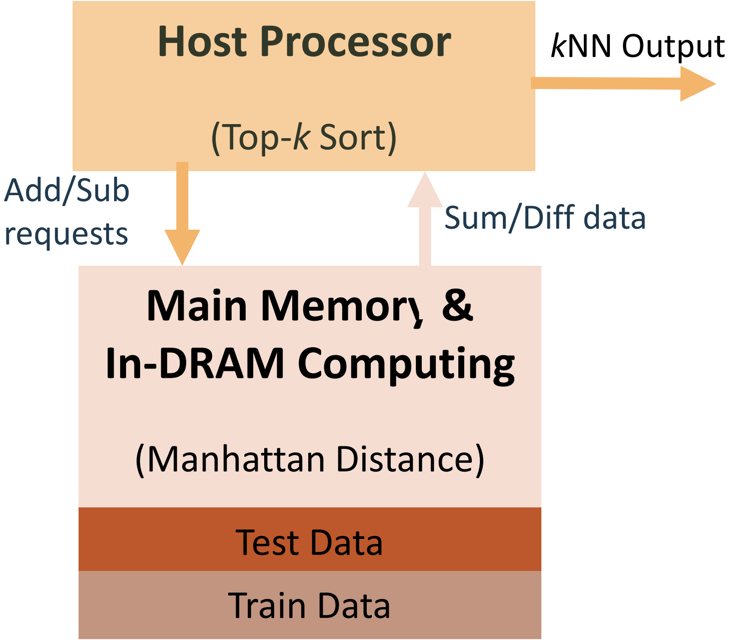
(4) Calculate S

For n-bit vector addition, 4n+1 operations are needed

Case Study: Compute-in-DRAM based k-NN Acceleration

- Checks *k* closest samples, Query vector assigned to the group that holds majority among those k samples
- Mainly consists of two computation stages: Distance computation and Global top-k sort
- The k value: The number of closest samples to be checked
- One-to-one distance computation: a large # of memory accesses




Host Processor

(Top-*k* Sort)

*k*NN Output

Add/Sub requests

Sum/Diff data

Main Memory & In-DRAM Computing

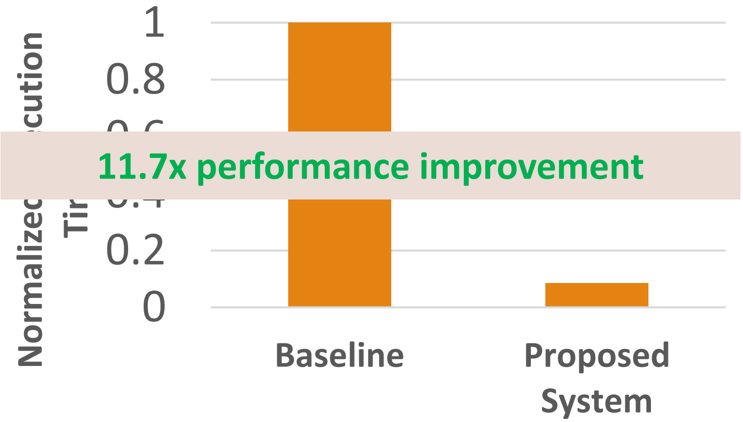
(Manhattan Distance)

Test Data

Train Data

Processor	X86, 2GHz
L1 Cache	**32KB I- and D-Cache**
L2 Cache	**2 MB**
Main memory	**1024 MB, DDR3-1600, 1 channel, 1 rank, 8 banks**

Using modified gem5 simulator from *S.Xu et. al,ICAL'19*



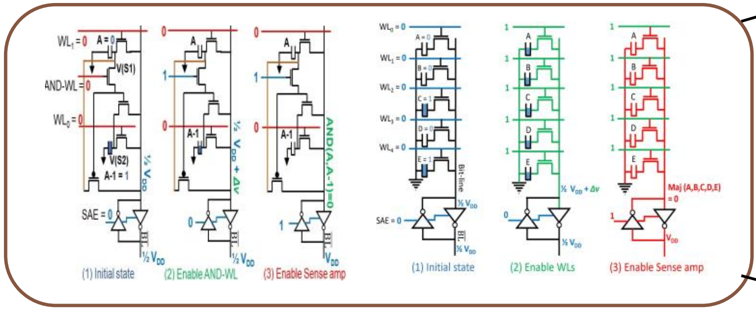
11.7x performance improvement

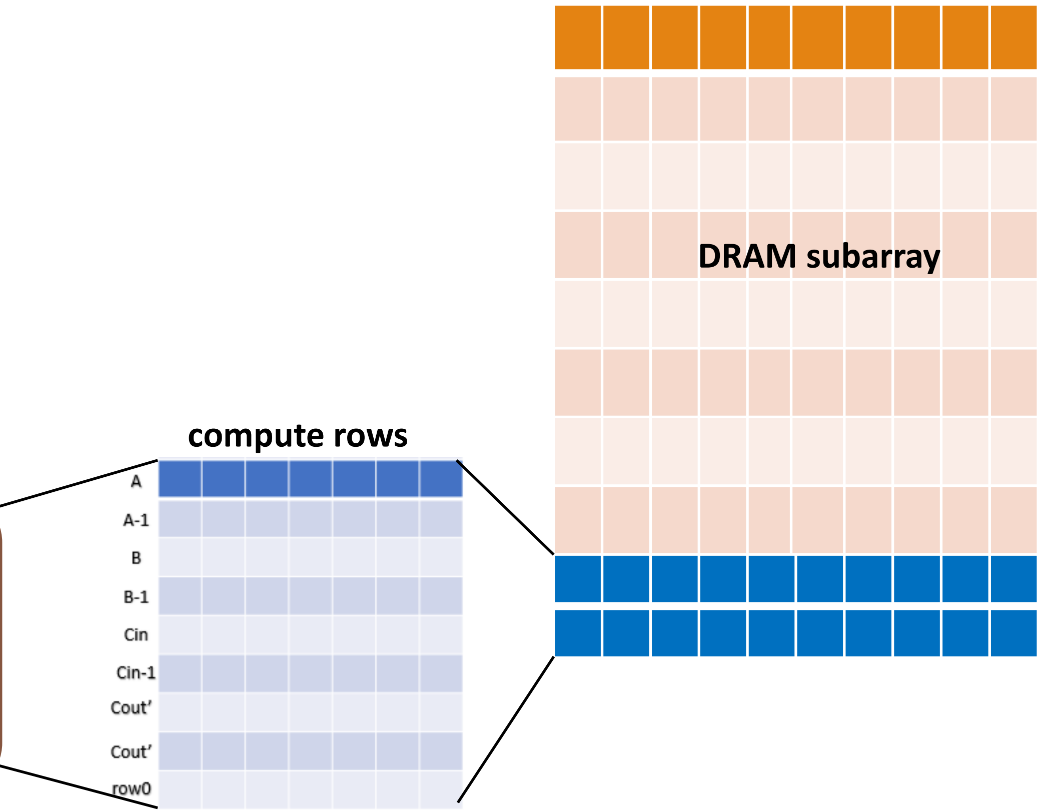
Ali, Jaiswal, Roy, "In-Memory Low-Cost Bit-Serial Addition Using Commodity DRAM Technology," TCAS-I, 2019

Multiplication Primitive

				A_3	A_2	A_1	A_0
				B_3	B_2	B_1	B_0
Carry out	A_3B_3	A_3B_2 A_2B_3	A_3B_1 A_2B_2 A_1B_3	A_3B_0 A_2B_1 A_1B_2 A_0B_3	A_2B_0 A_1B_1 A_0B_2	A_1B_0 A_0B_1	A_0B_0
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

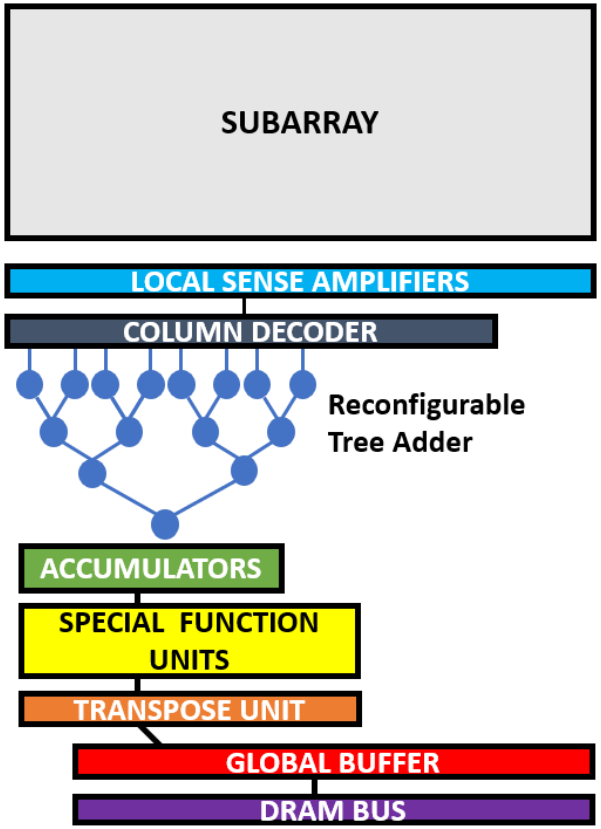
AND + **ADD**





- Multiplication can be broken down in terms of AND and ADD operations.
- Multiplication operation reserves 9 compute rows in the DRAM subarray.

PIM-DRAM Architecture



SUBARRAY

LOCAL SENSE AMPLIFIERS

COLUMN DECODER

Reconfigurable Tree Adder

ACCUMULATORS

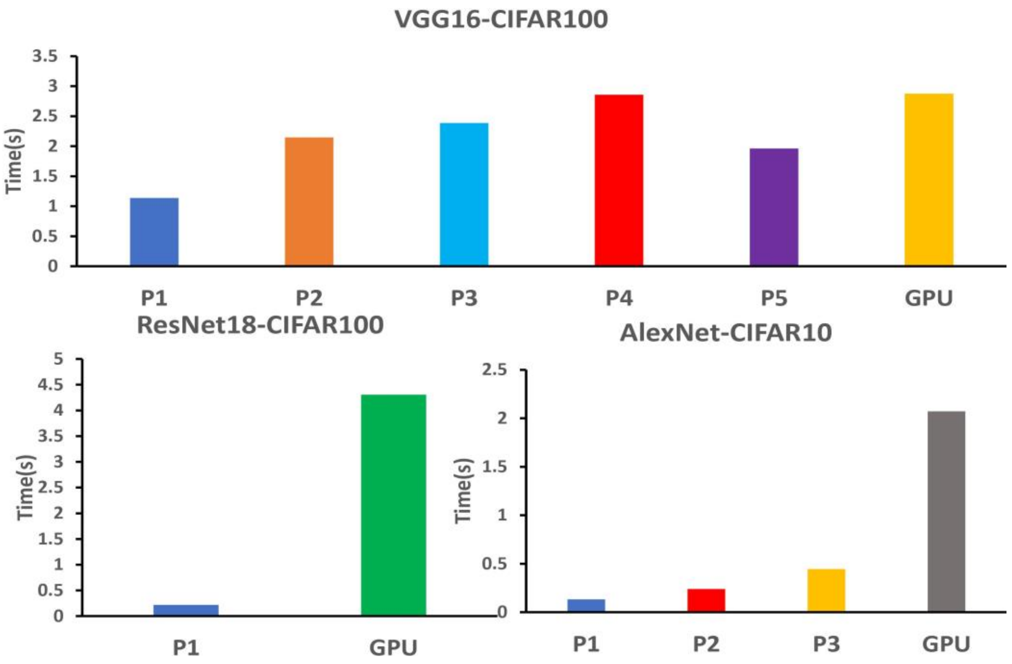
SPECIAL FUNCTION UNITS

TRANSPOSE UNIT

GLOBAL BUFFER

DRAM BUS

Roy, Ali, Raghunathan, PIM-DRAM: Accelerating Machine Learning Workloads using Processing in Commodity DRAM, IEEE JETCAS, 2021

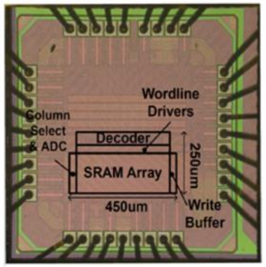


4-bit act. and W

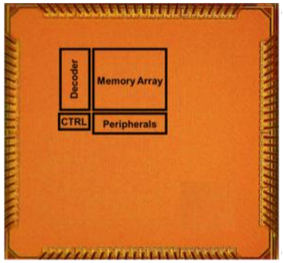
- ➢ Special Function Units consist of ReLu, Pooling, Batch normalization and Quantize units.
- ➢ The Multiplication operation happens in parallel across different DRAM subarrays for data in transposed layout.
- ➢ Every DRAM bank is allocated to a layer in the Neural Network.
- ➢ P1-P5 represents the degree of parallelism while mapping DNN layers on DRAM banks.

C-BRIC Artifacts: Chip Gallery #1
In-Memory Computing, Digital DNN Accelerators

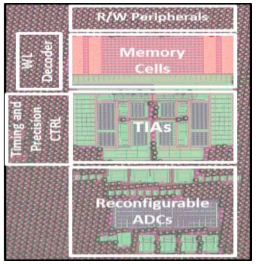
SRAM IMC



65nm: XNOR-SRAM
S. VLSI'18, JSSC'20

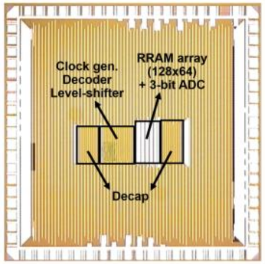


65nm: IMPULSE
SSCL'21

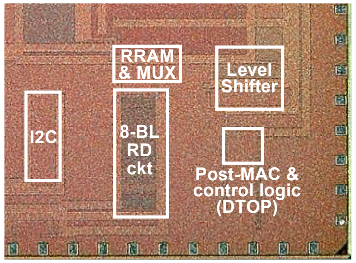


65nm: Dyn. Sparsity
SSCL'21

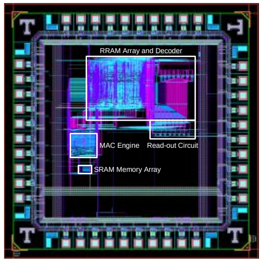
RRAM IMC



Winbond 90nm:
XNOR-RRAM, 2-bit-per-cell
Micro'19, TED'20, SSCL'20



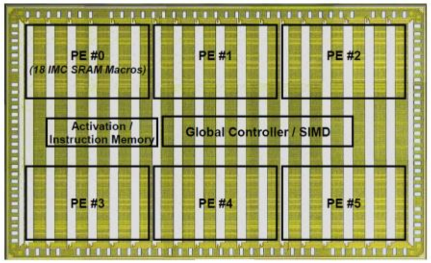
TSMC 40nm: Binary RRAM,
multi-bit encoding
ISSCC'21, CICC'21, JSSC'22



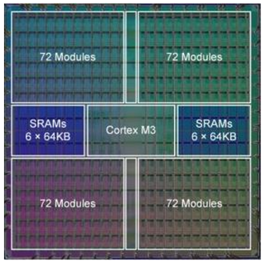
SUNY 65nm:
RRAM/SRAM
based hybrid IMC

IMC Macros

IMC Systems

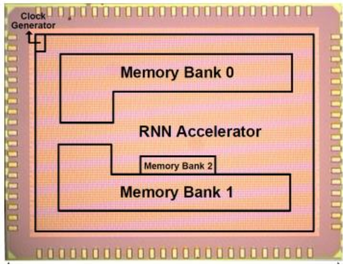


28nm: Prog. IMC accelerator
S. VLSI'21

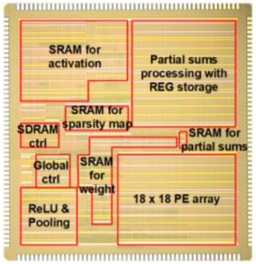


TSMC 40nm: RRAM IMC System
with embedded processor

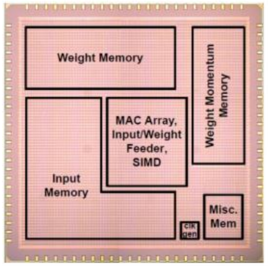
Digital Accelerators



65nm: LSTM inference w/
hierarchical structured sparsity
ESSCIRC'19, JSSC'20



40nm: CNN inference w/
conditional computing
CICC'20, JSSC'21

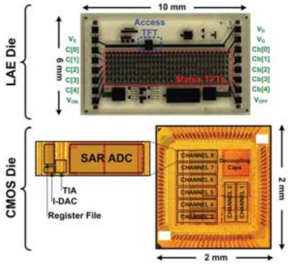


65nm: 16-bit fixed-point
CNN training accelerator
SSCL'20

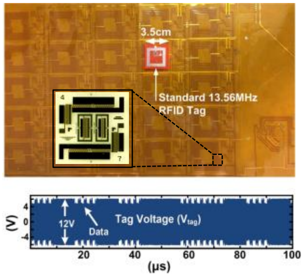
ASU
Purdue
Georgia Tech
(w/ ASCENT,
Samsung, TSMC)

C-BRIC

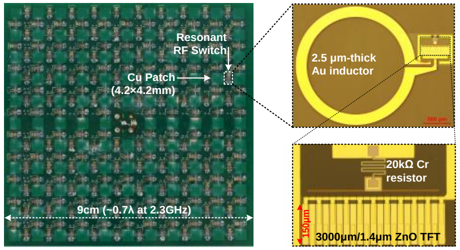
C-BRIC Artifacts: Chip Gallery #2
Large-Area Sensing, RL, SNN, Optimization, Robotics



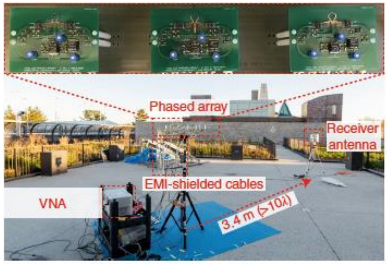
Hybrid 130nm CMOS / LAE tactile sensing array ISSCC'19



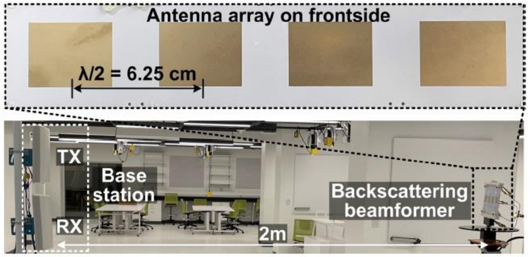
LAE 13.56 MHz RFID reader array SSCL'18



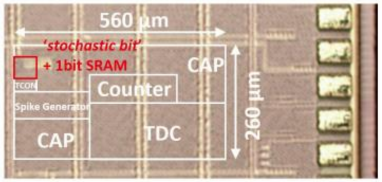
LAE 2.4 GHz monolithically-integrable reconfigurable antenna IEDM'20



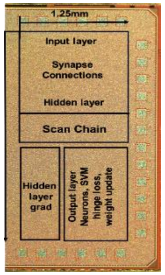
LAE 1 GHz Phased Array Nature Electronics'21



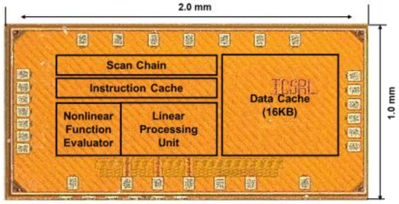
LAE 2.4 GHz passive backscattering beamformer for event-driven sensing



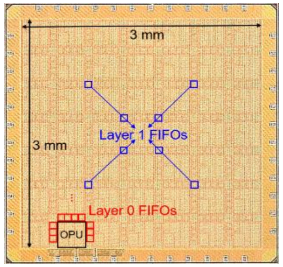
Stochastic binary SNN 90nm: TCAS-I'20



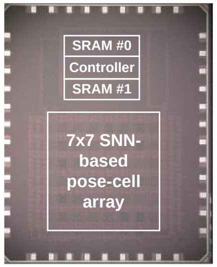
Tiny-RL, analog compute 130nm: JSSC'18



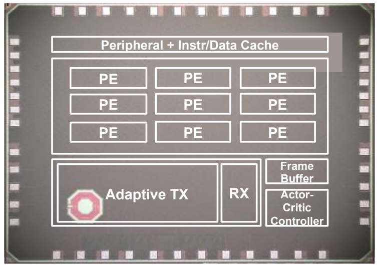
Edge AI for swarm robotics 130nm: ISSCC'18, JSSC'19



65nm: OPTIMO 49-core optimization processor CICC'19, JSSC'19



65nm: NeuroSLAM SNN-based visual SLAM acc. ISSCC'20, JSSC'21



65nm: Edge SoC with edge-cloud load balancing S' VLSI'20, JSSC (review)

Princeton
Georgia Tech
Purdue (w/ Intel)

Parting Thoughts…

- While possibilities of achieving large improvements in inference latency and energy is possible…
- There are several challenges…
  - Array efficiency
  - Cross-bar non-idealities: Device non-linearity, access transistor/selector device, circuit non-idealities (line resistance, source/sink resistance), process variability
  - Reliability and endurance of non-volatile devices
  - High write cost for NVMs
  - A/D and D/A converters
  - Data movements from partial sums
  - Training/mapping to the hardware – degradation over time for some NVM technologies
  - Need for vector operations and floating point operations
  - SRAMs are large compared to emerging NVMs

Neuro-electronics Research Laboratory

C-BRIC

Acknowledgement

Vannevar Bush Faculty Fellow