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

KAUSHIK ROY

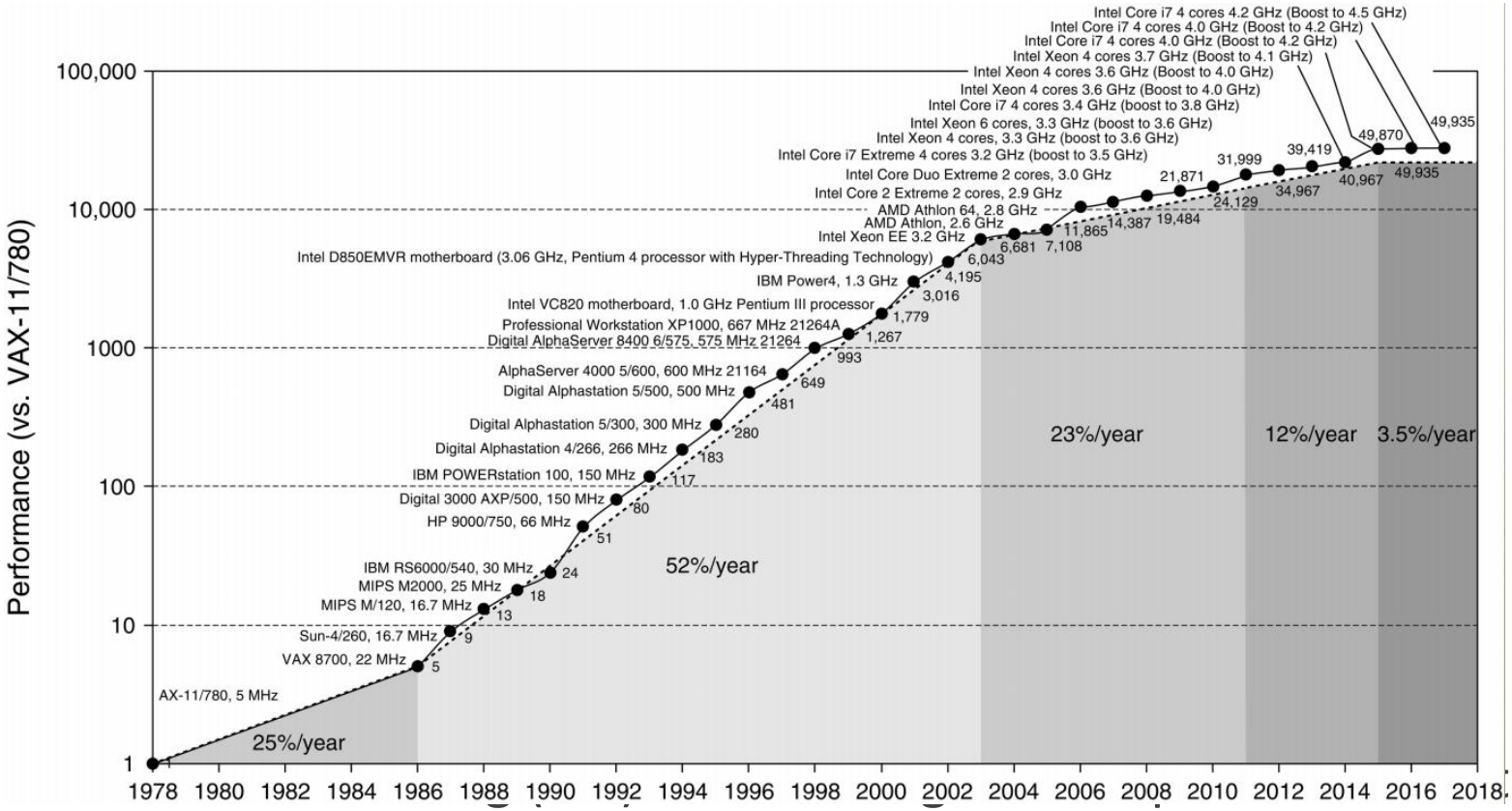
PURDUE UNIVERSITY

KAUSHIK@PURDUE.EDU

SRC JUMP

C-BRIC

Challenge: Computational Demands and Technology Scaling

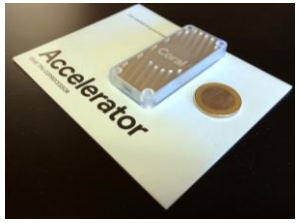


- Dennard scaling has ended -> General-purpose systems have hit performance/energy wall
- Domain specific acceleration (hardware/software) is the key
  - Industry examples - Google TPU, Microsoft Brainwave, Nvidia Tensor Core

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

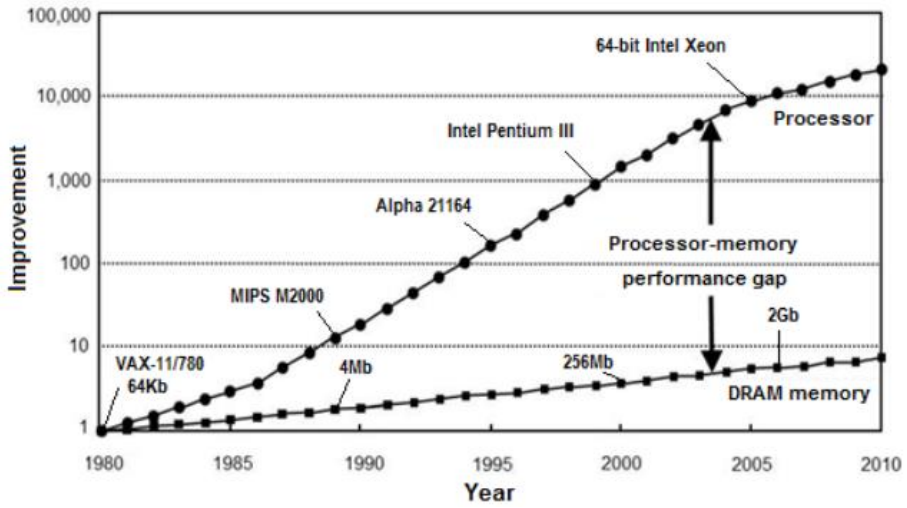
Where do the in-efficiencies come from?

Algorithms — Hardware Architecture — Circuits and Devices

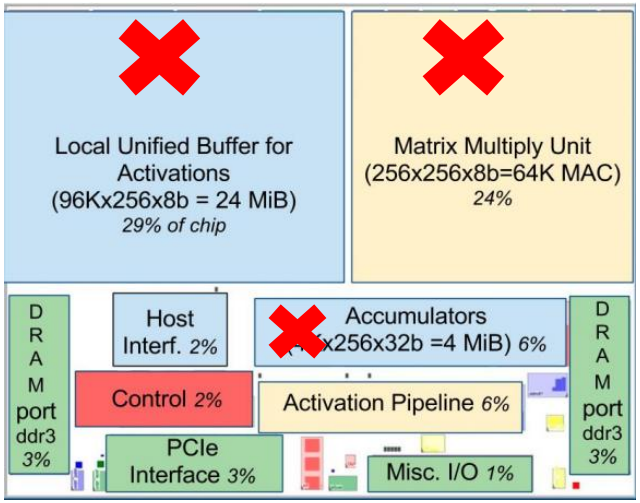
Ref: Venkataramani, S., Roy, K. and Raghunathan, A. "Efficient embedded learning for IoT devices." In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 308-311. IEEE.

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

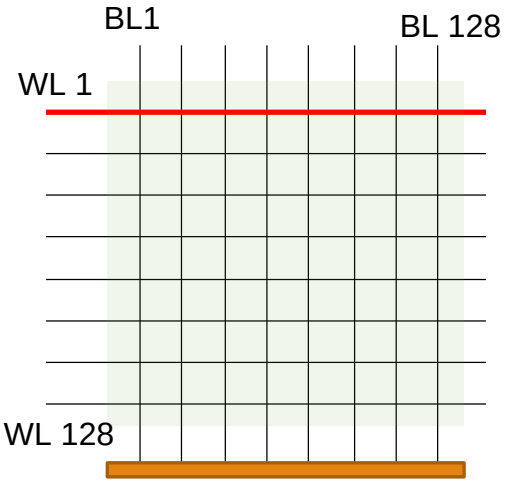


TPU Floorplan, ISCA 2017

In-Memory Computing for ML

2KB Memory

Typical Memory Access



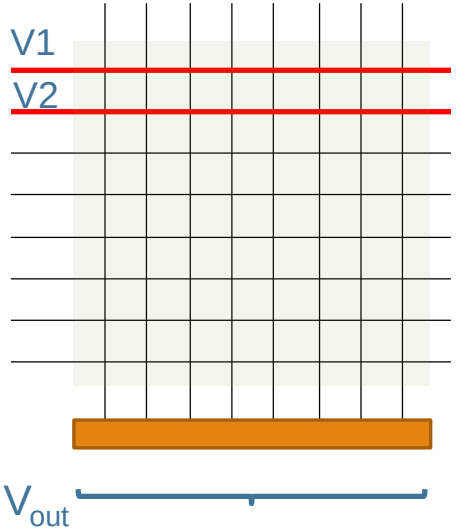
Bits Accessed = 128

Operation = Read/Write

2KB Memory

In-Memory Computing

Digital



Bits Accessed = 2*128

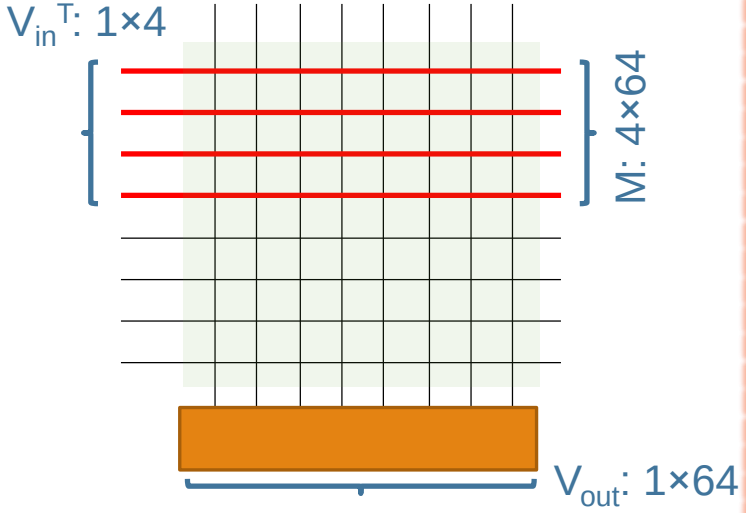
Operation = Bit-wise AND

V_{out} = V1 AND V2

2KB Memory

In-Memory Computing

Analog



Bits Accessed = 4*128

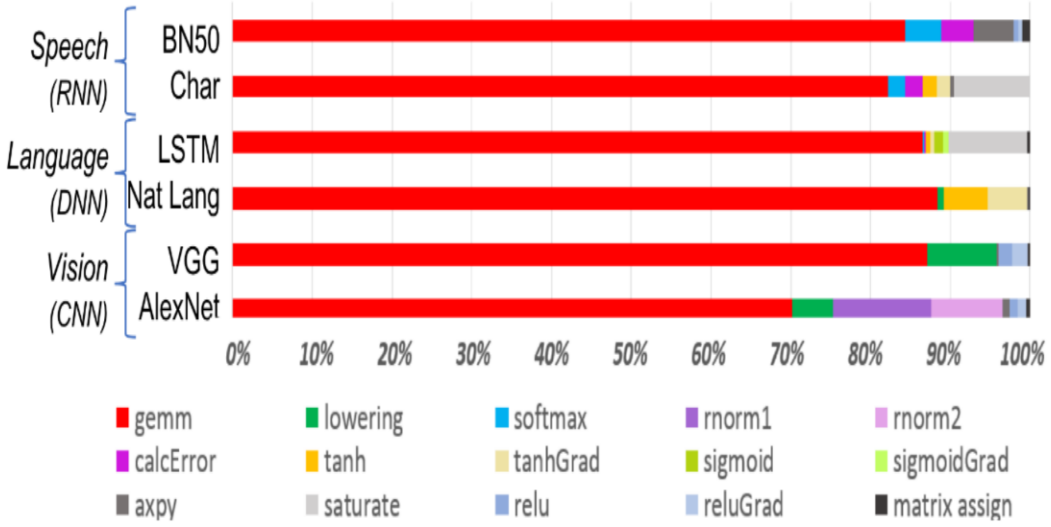
Operation = Matrix Vector Multiply

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)

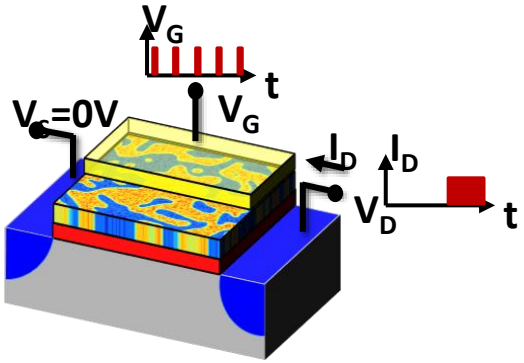
- Deep Learning needs – lots of matrix multiplications

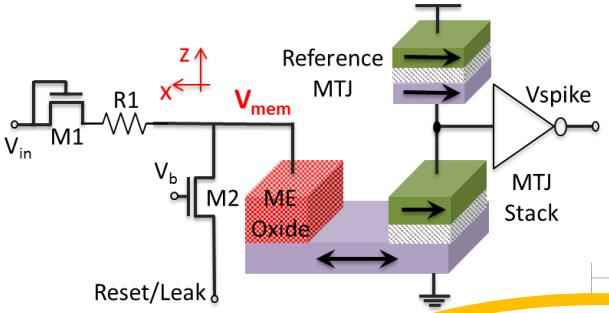


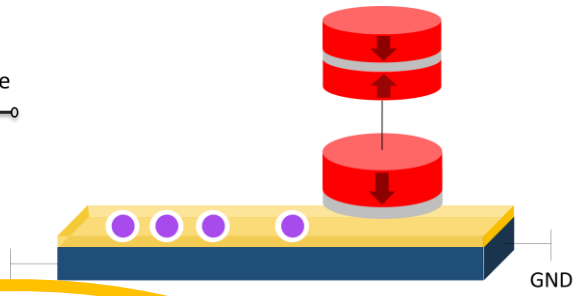
Bruce Fleischer et al, IBM Research, 2018

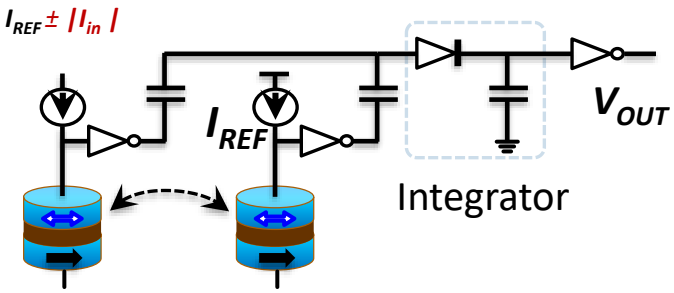
- Challenge: sustaining deep learning's insatiable compute demands

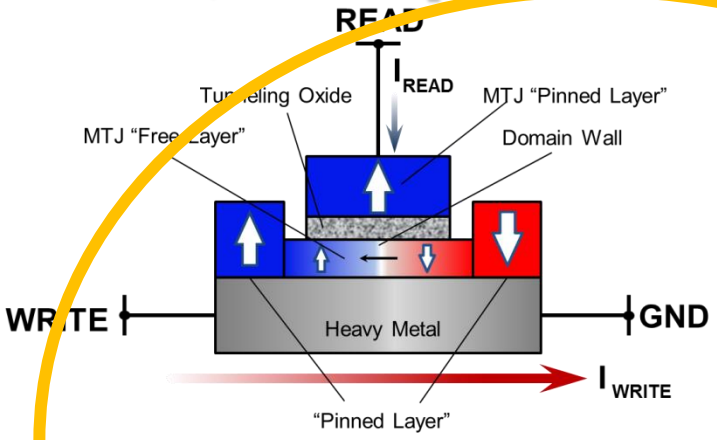
Technology: Non-Volatile Memories

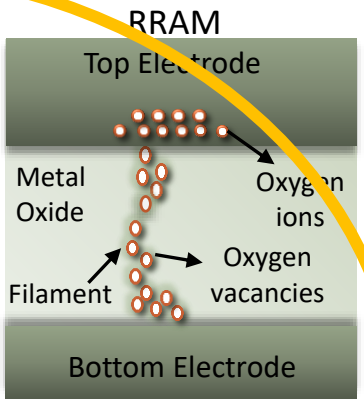


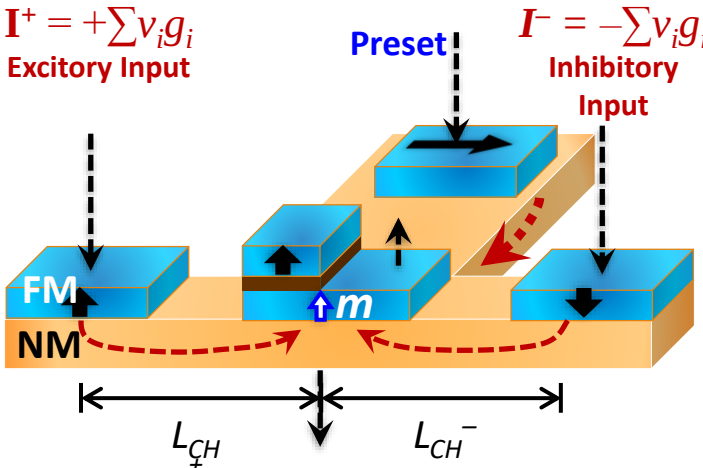


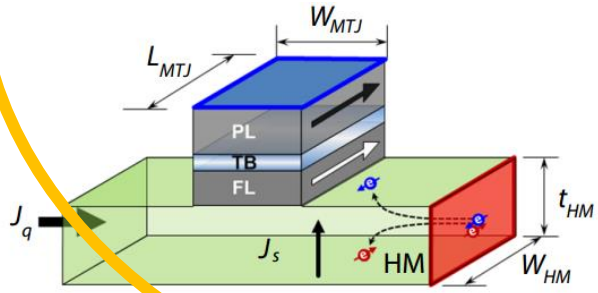


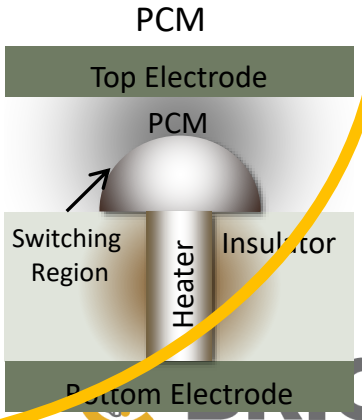




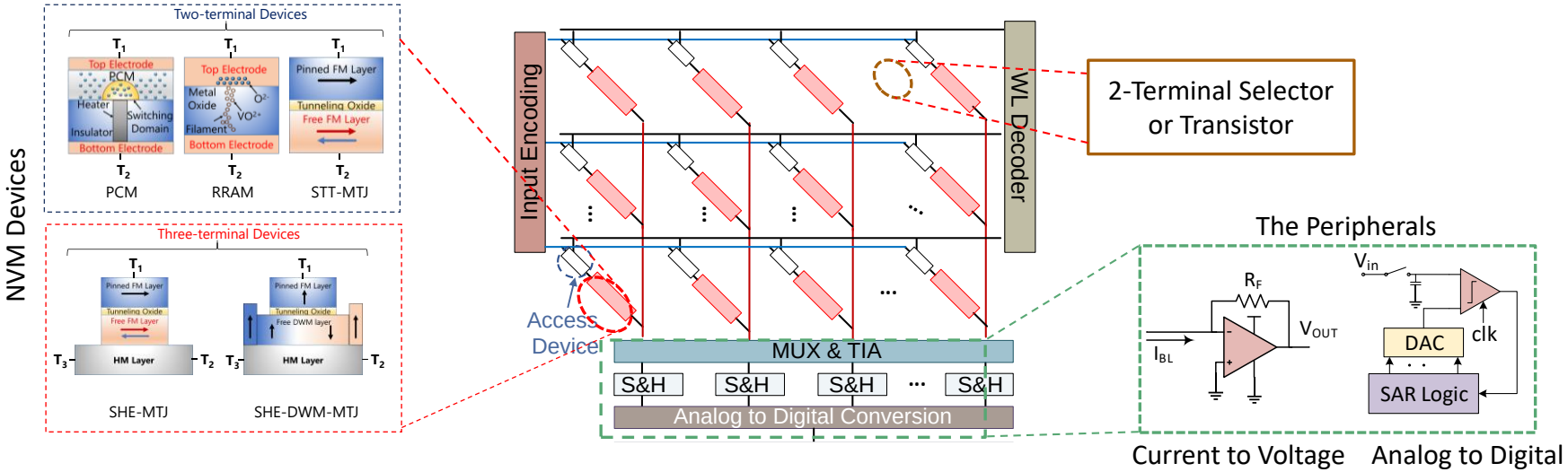


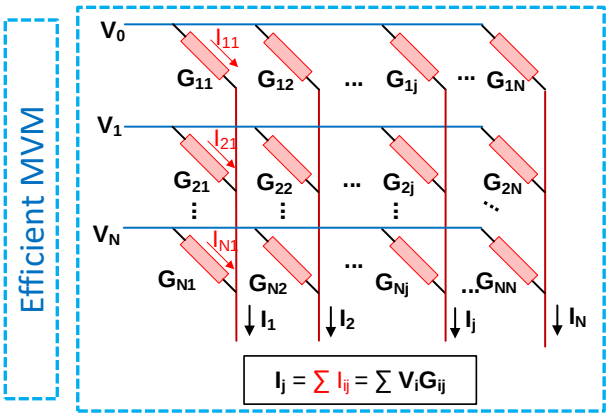


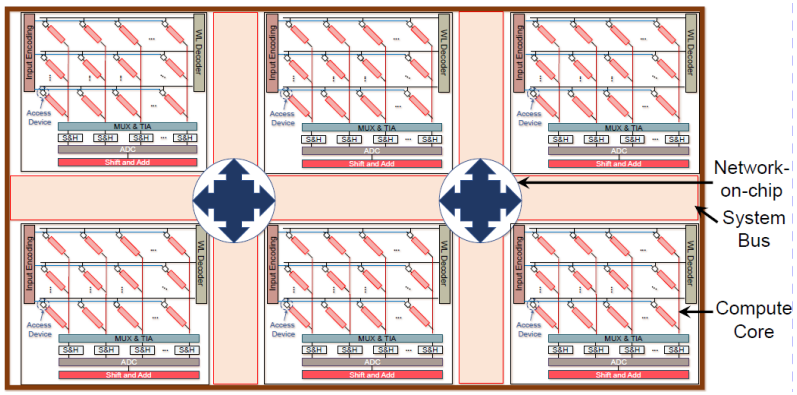




Non-volatile Memory Crossbars

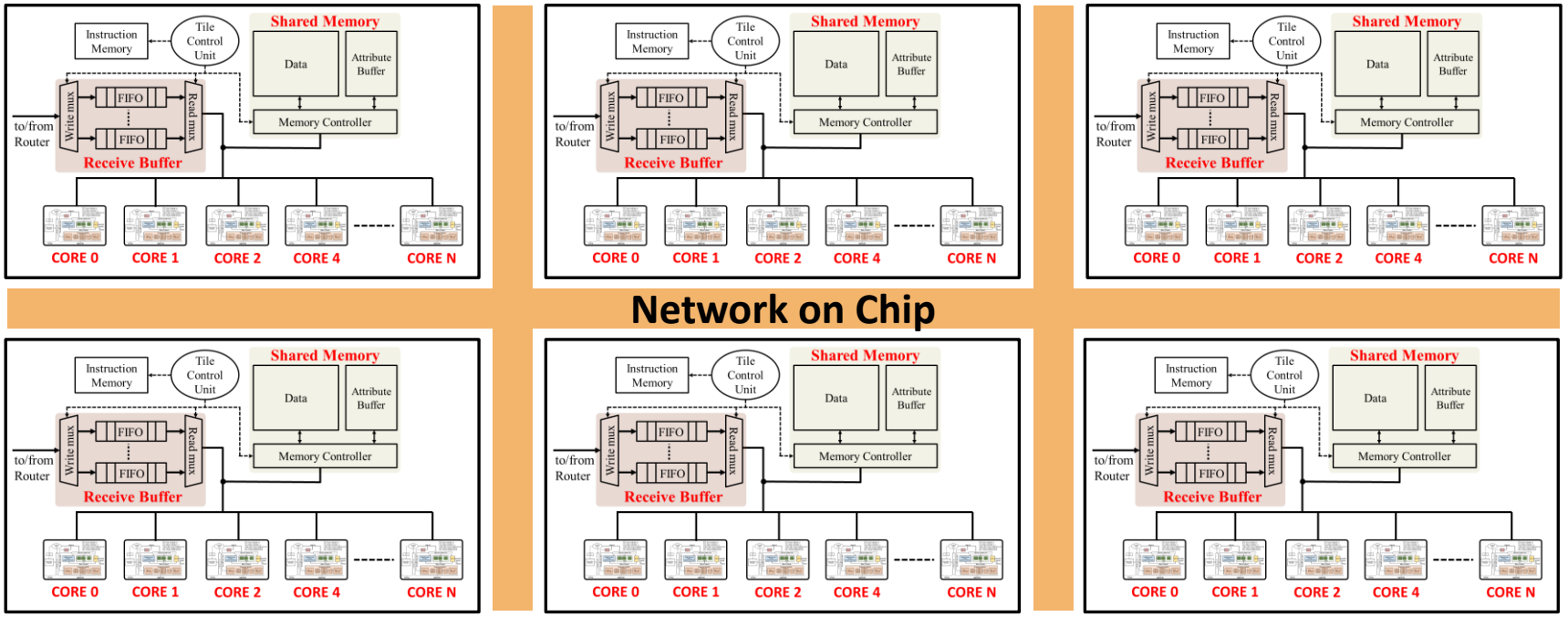






High on-chip storage

Architecture: *Spatial scalability*

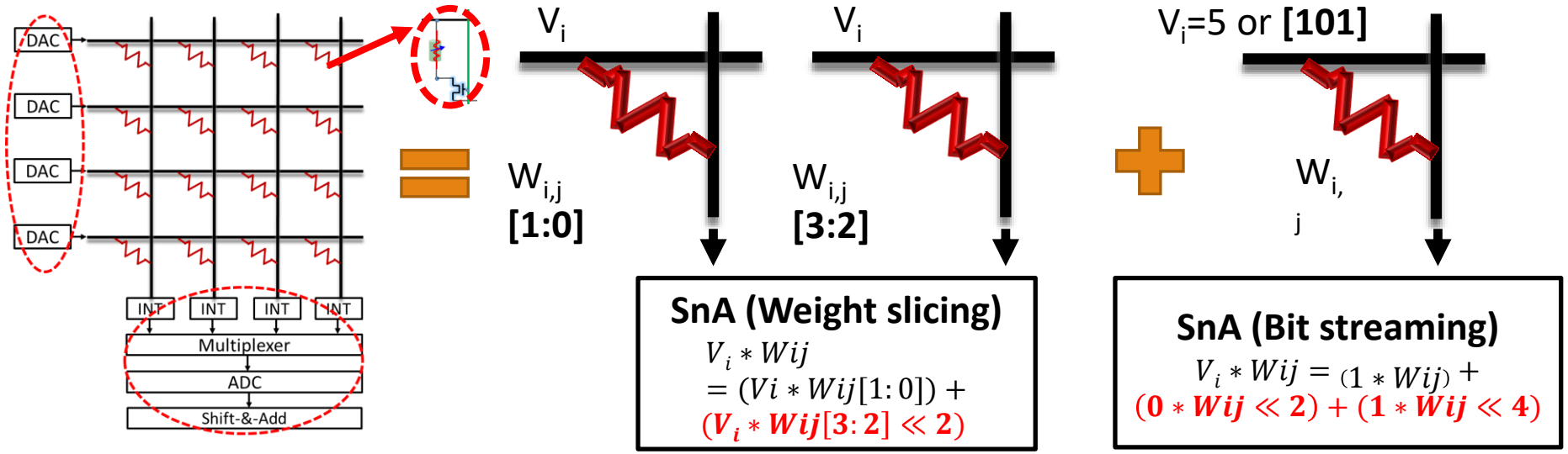


Massively parallel accelerator –> Amenable to Data-Level Parallelism -> Highly efficient ML inference

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

C-BRIC

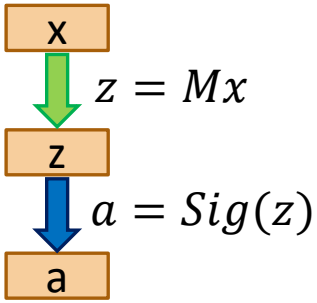
Bit-slicing (weights and inputs)



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

Architecture – *Heterogeneous core*

1 Layer MLP



MLP maps on one MVMU

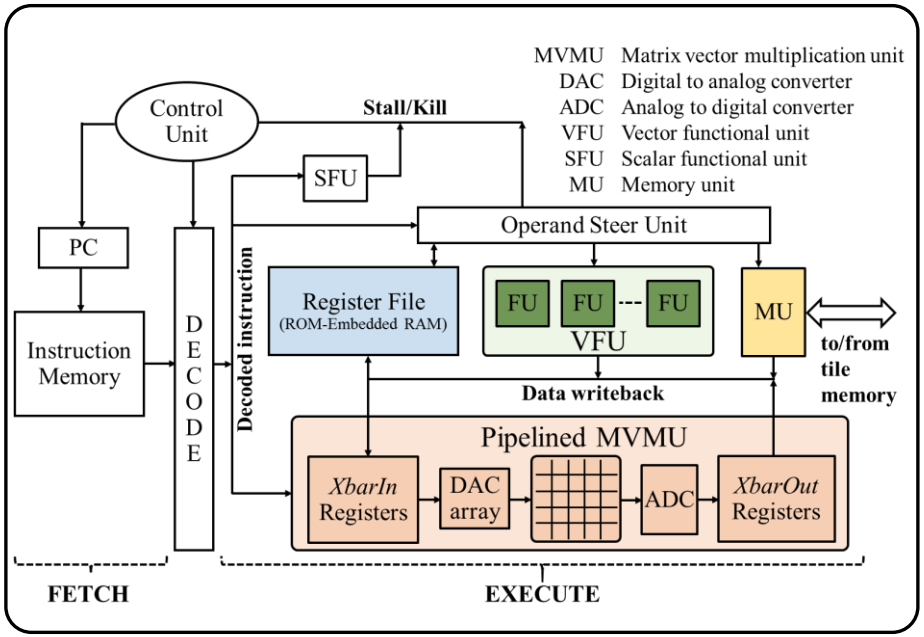
```
LD $m_i1, x, n
MVM 0b01
ALU_SIG $1, $m_o1, n
ST $1, a
```

MLP maps on multiple MVMUs/Cores

```
MVM 0b01

// for all partial sums
LD xxx // bring partial sums from other cores
ALU_ADD xxx // accumulate partial sums

ALU_SIG $1, $m_o1, n
```



*MVMU is implemented using bit-slicing as described in PUMA (*Ankit et al, ASPLOS 2019*)

Larger matrices require more data movements to compute effective MVM.

Wish List….

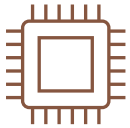
Device

High ON/OFF Ratio

Device Linearity

High Endurance

Multi-level cells



Macro

Large Crossbar Size

Low Peripheral Overhead

Good Selector Device

Source/Sink Resistance

Line Resistance

Architecture/Algo

Minimal data movement

MVMs, Vector Operations on NVM

Training/Mapping on Architecture

Sparsity

C-BRIC

Challenges: NVM devices

Device

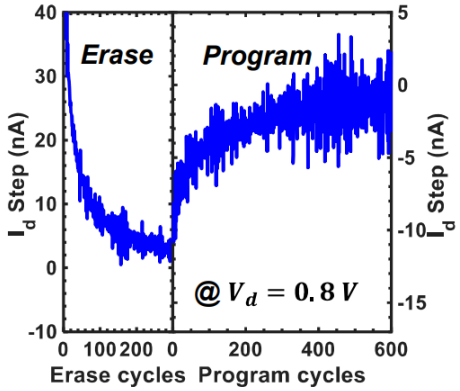
High ON/OFF Ratio

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

Challenges: NVM Compute Macro

Macro

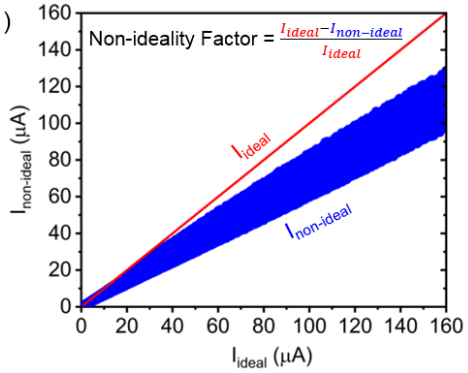
Large Crossbar Size

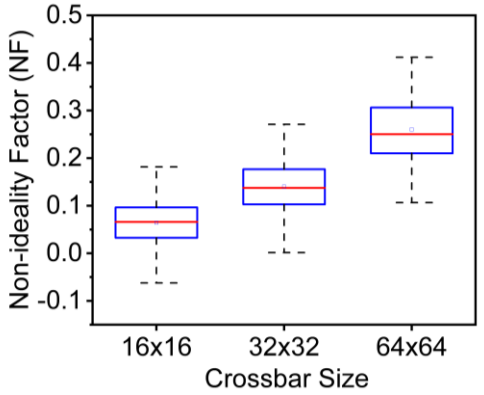
Low cost peripheral overhead

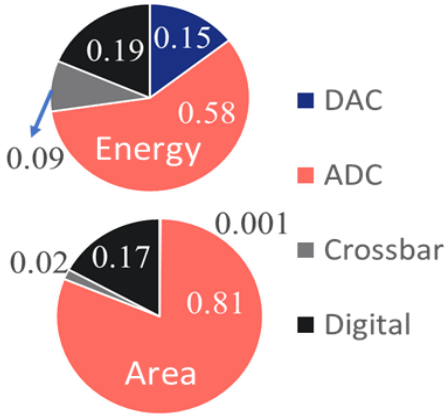
Good selector device

Source/Sink/Line resistances

- 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

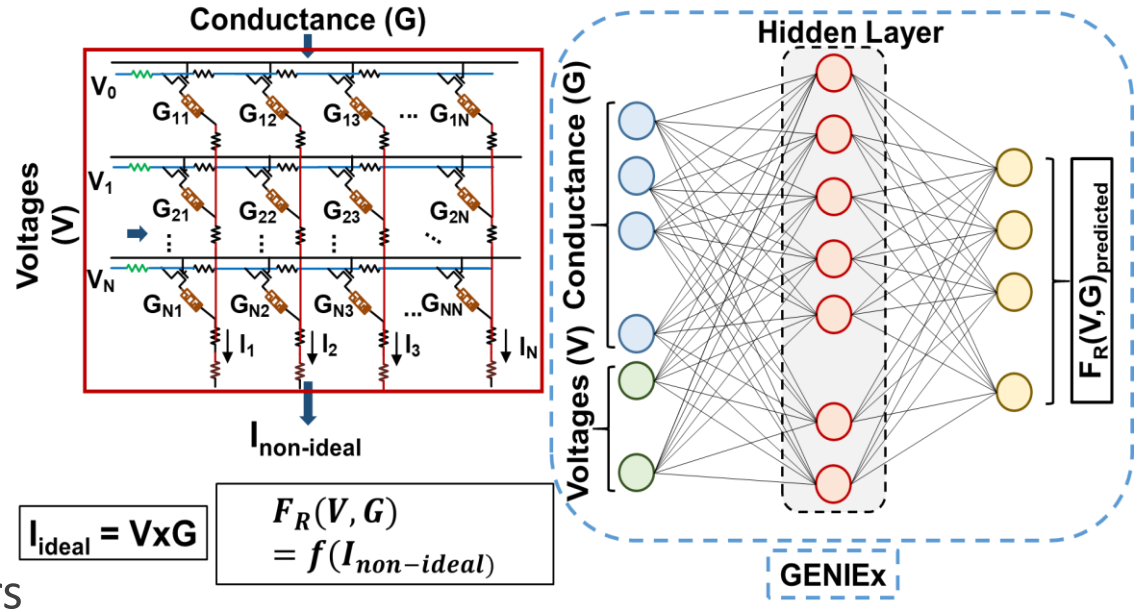
Source: Ankit et al, ASPLOS 2019

C-BRIC

Modeling Cross-bars – GENIEx, A Generalized Approach to Emulating Non-ideality in Memristive Crossbars

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

- 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



$I_{ideal} = VxG$

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

GENIEx provides modeling capability for different types of non-idealities

Chakraborty, Ali, Ankit, Roy, "GENIEx: A Generalized Approach to Emulating Non-Ideality in Memristive Xbars using Neural Networks", DAC 2020

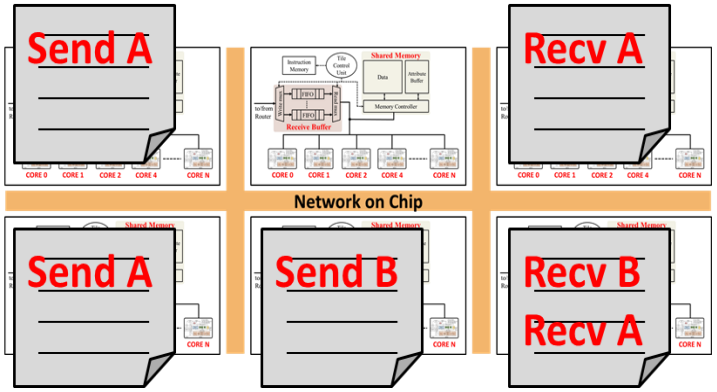
C-BRIC

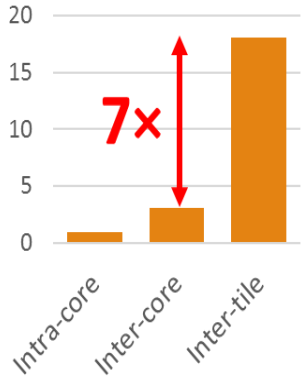
Challenges: Architectures

Architecture

Minimal data movement

Wide variety of operations with NVM

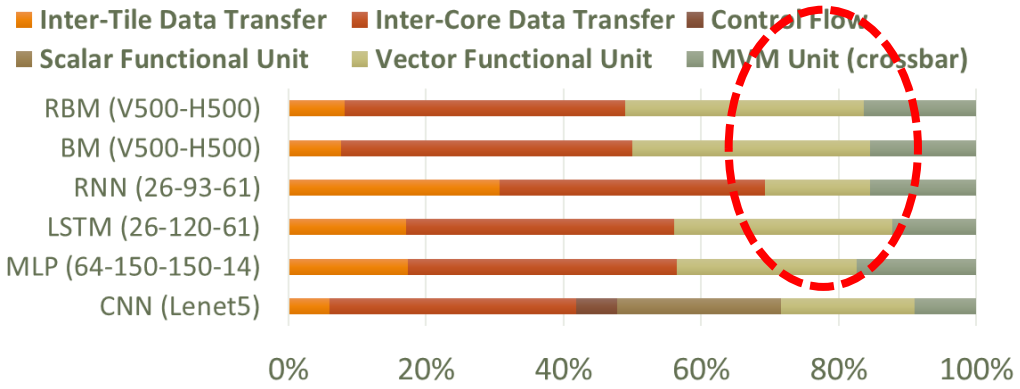




Data movement between MVMUs – partial sum, activations is frequent

- Cost of data movement increases across the hierarchy – core, tile, chip

Mapping of workloads to Instructions



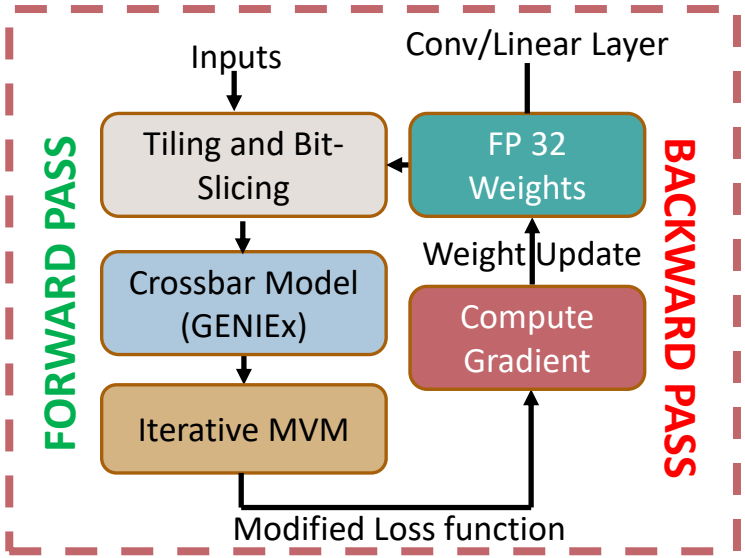
MVMU alone cannot execute ML apps

- Vector operations: require previous writes to crossbar
- Transcendental operations: MVMU (even with writes) can do add/multiply only

C-BRIC

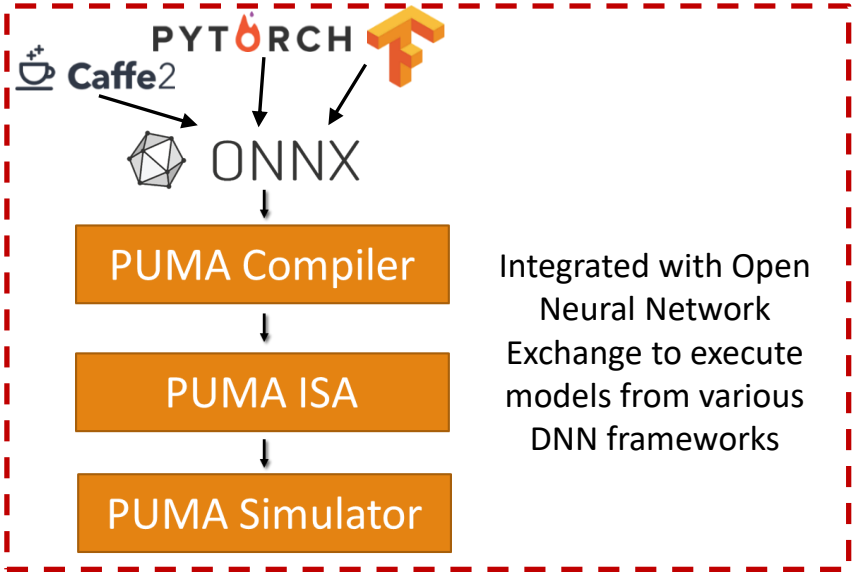
Potential Solutions

Improving functional accuracy



- **Modifying the training algorithm can recover functional inaccuracies due to crossbar non-idealities**

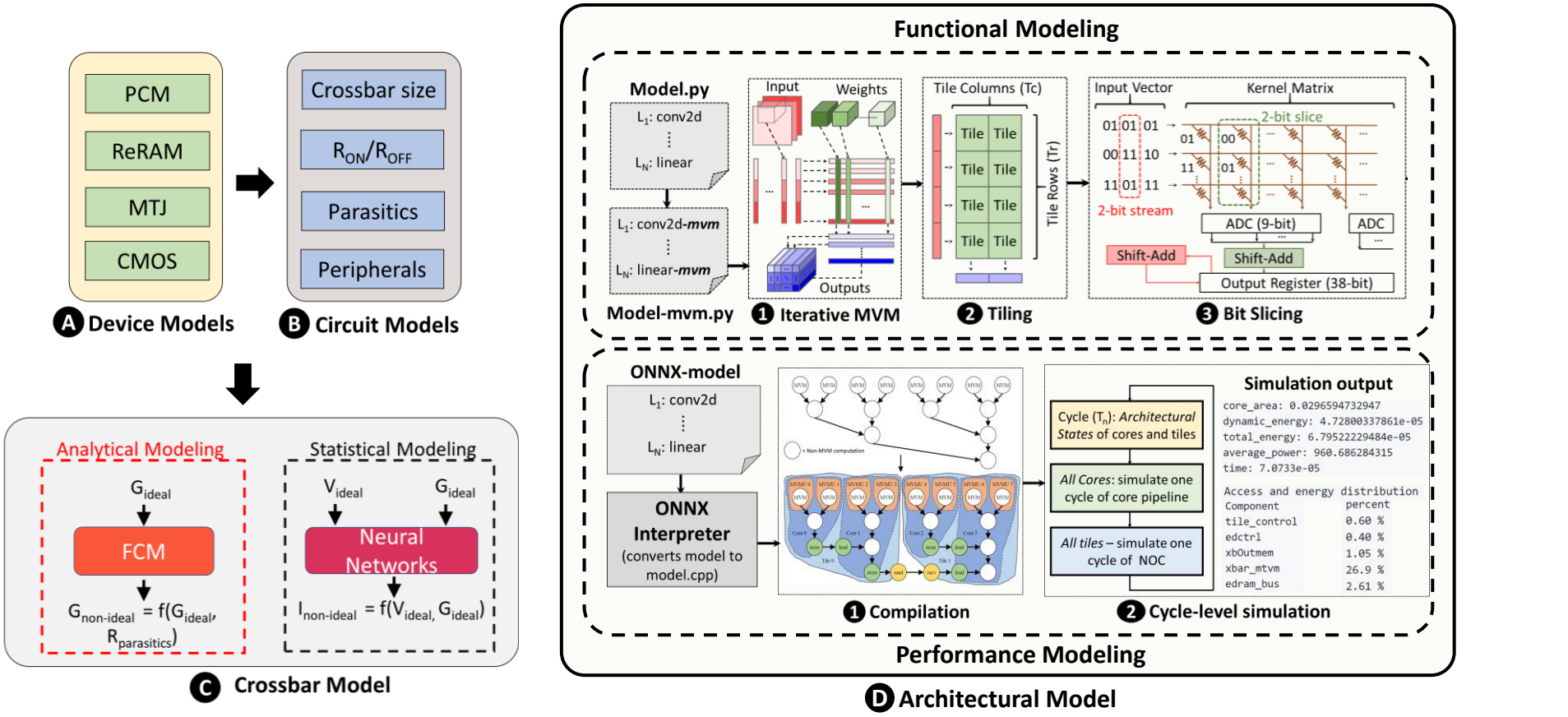
Improving hardware efficiency



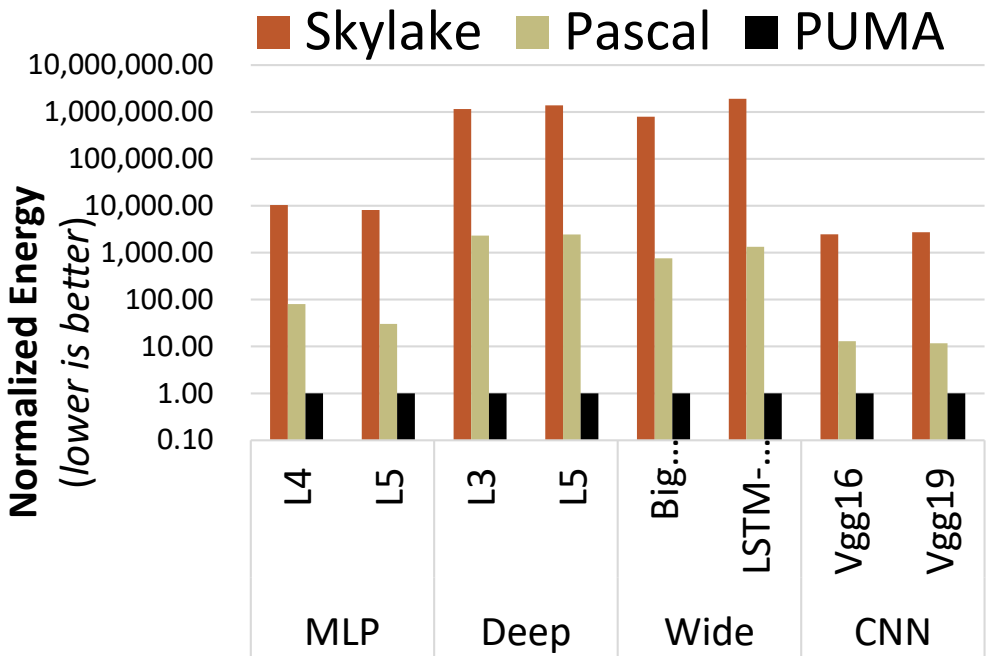
- **Hardware-software codesign to optimize data reuse, support different operations/models can lead to high efficiency – low energy, latency and high utilization.**

C-BRIC

Modeling and Simulation Ecosystem



Results – Inference Energy (Batch size 1)

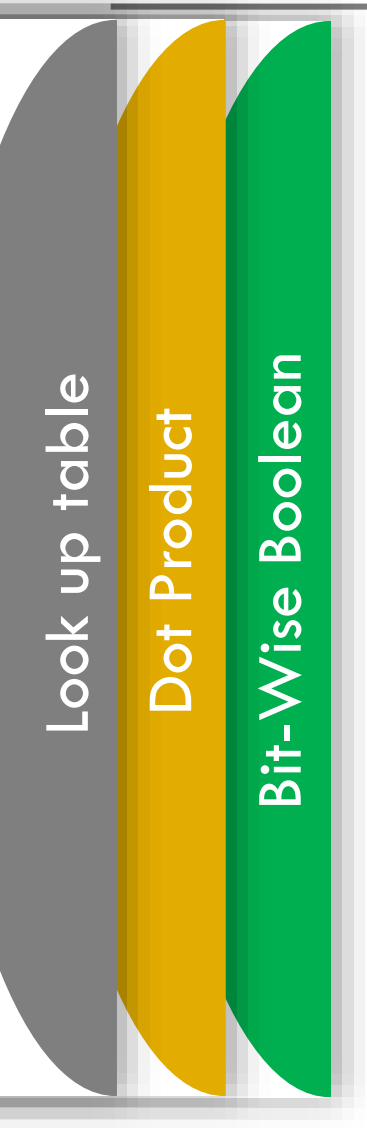


- CNNs show upto 13.0× reduction (least).
  - **High weight reuse, even at batch-size 1.**
- MLPs show upto 80.1× reduction.
  - **No weight reuse, small models.**
- LSTMs show upto 2446× reduction.
  - **Little Weight reuse, large models (billions of parameters).**

- **PUMA achieves massive reduction in inference energy across all platforms for all benchmarks.**
- Similar trends in inference latency reductions are obtained across all benchmarks.

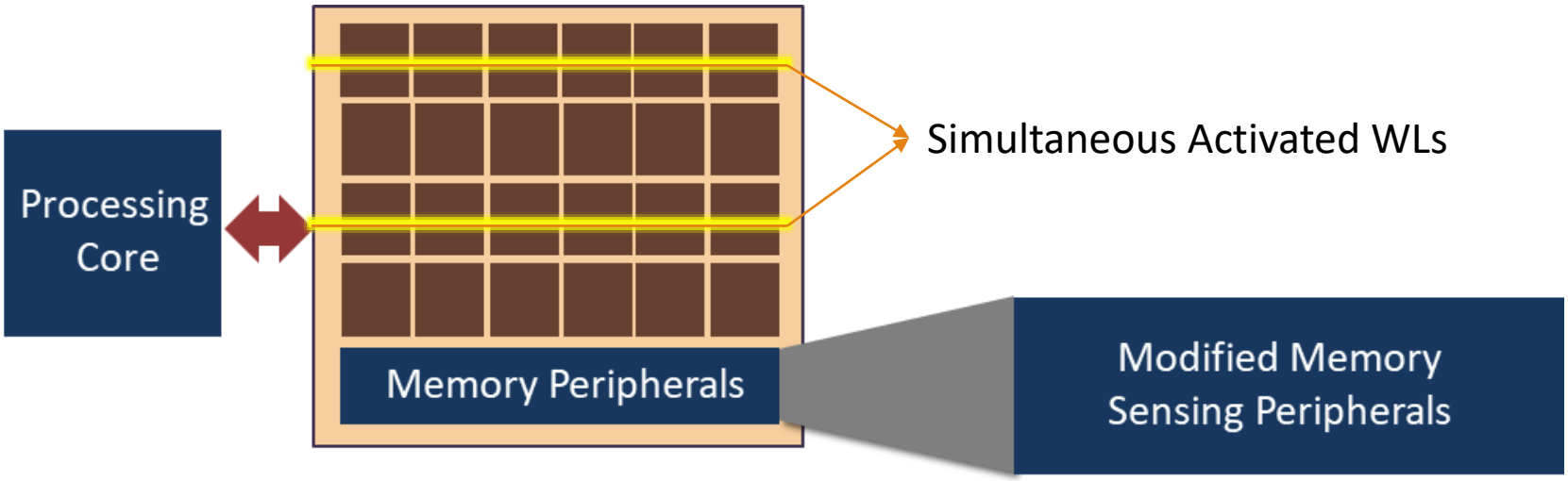
C-BRIC

In-Memory Computing…SRAMs, DRAMs..



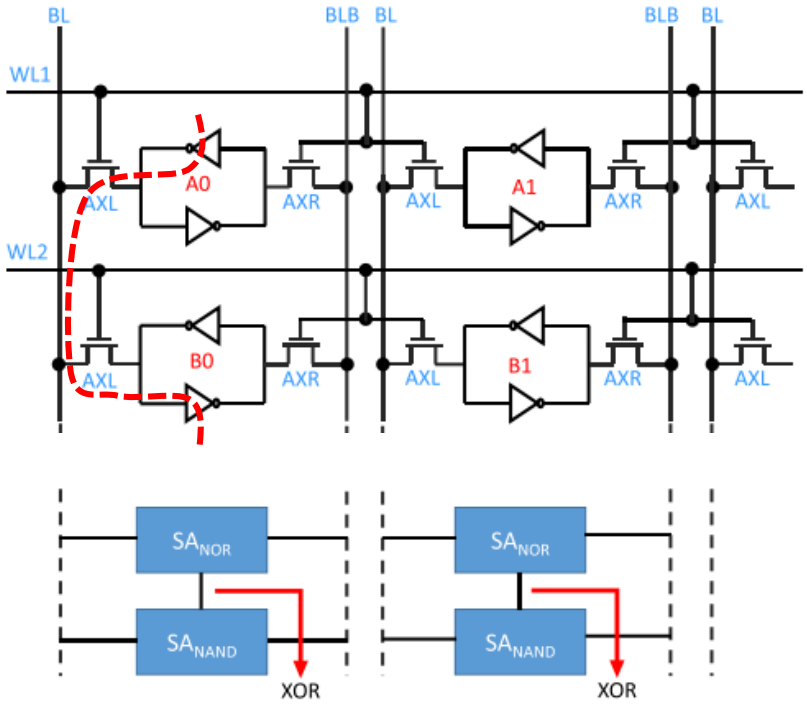
In-Memory Bit-wise Vector Boolean Operations

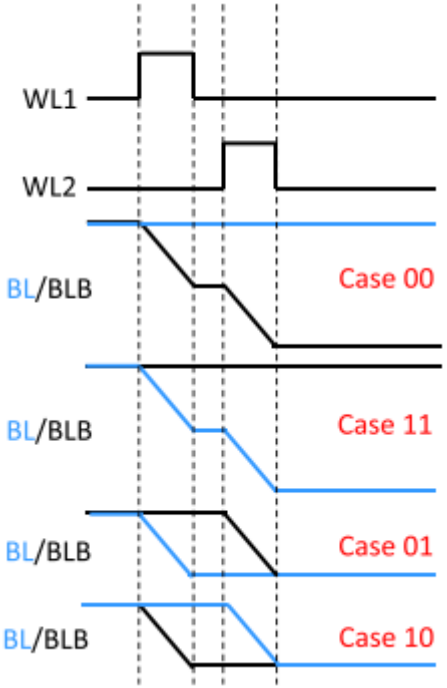
Modifying the Peripheral Sensing Circuits to Read-out A Vector Boolean Function



C-BRIC

SRAM: In-Memory Bit-wise Boolean Operations



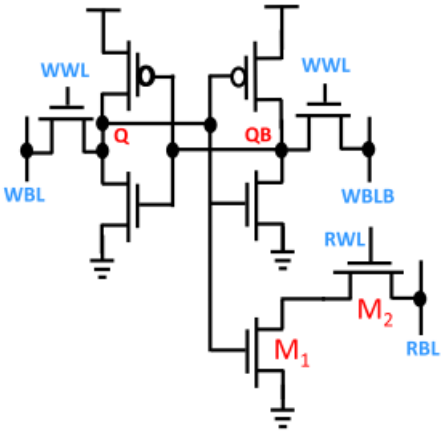


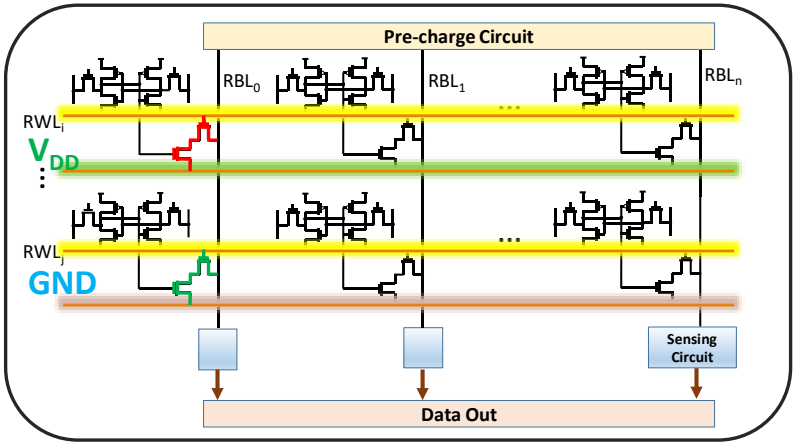
Agrawal, A et al., 2018. X-sram,. IEEE TCAS-I

6T-SRAM

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

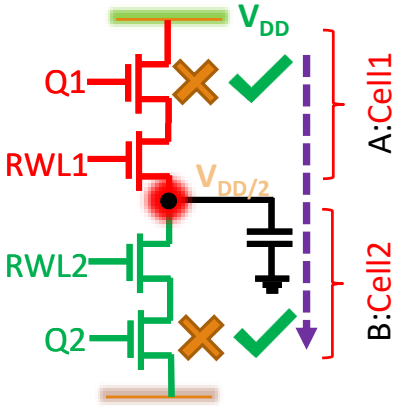
X-SRAM: Bit-Wise Vector Boolean Operations

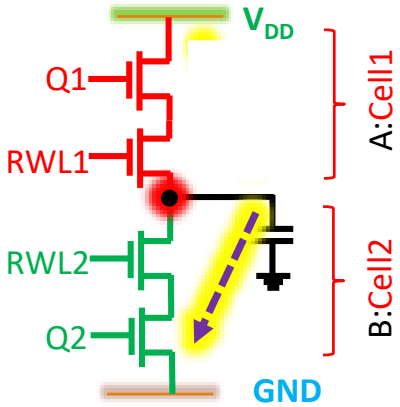


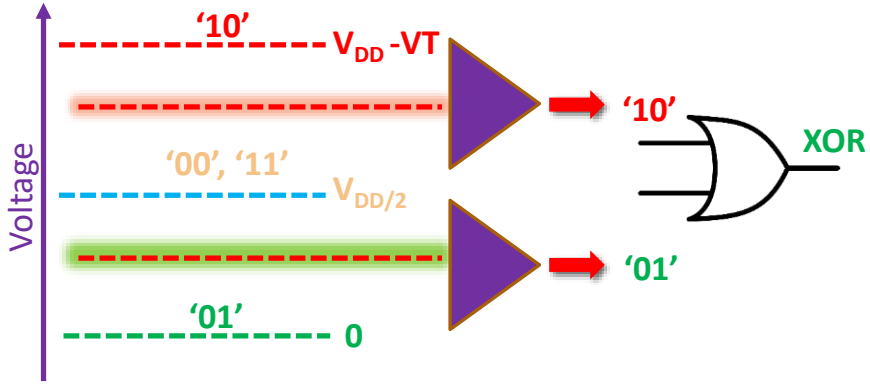


8T-SRAM

- Decoupled read/write: simultaneous WL activation.
- 8T cells are wire-NORed. Easy to sense bitwise NOR.
- Single ended read inverters for sensing.

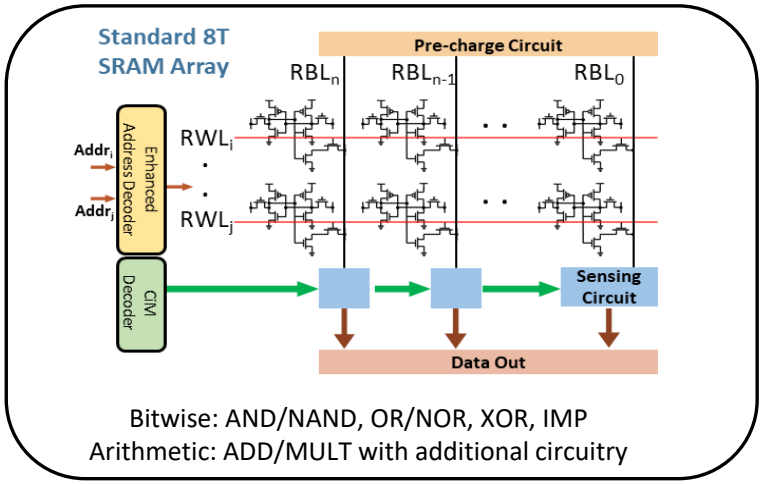




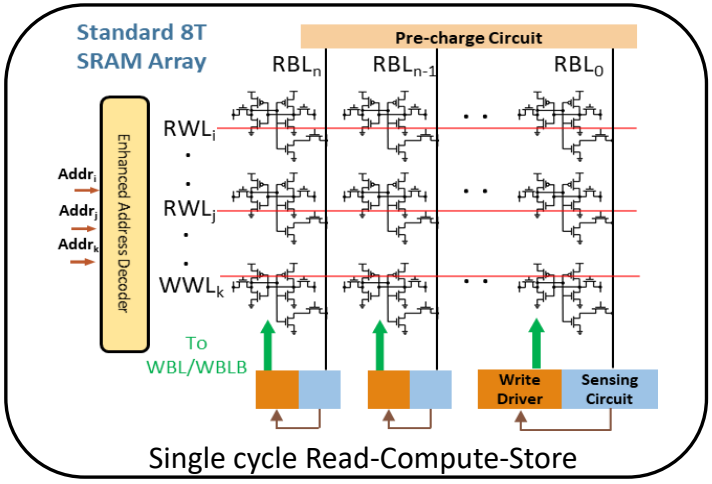


Agrawal, A et al., 2018. X-sram,. IEEE TCAS-I

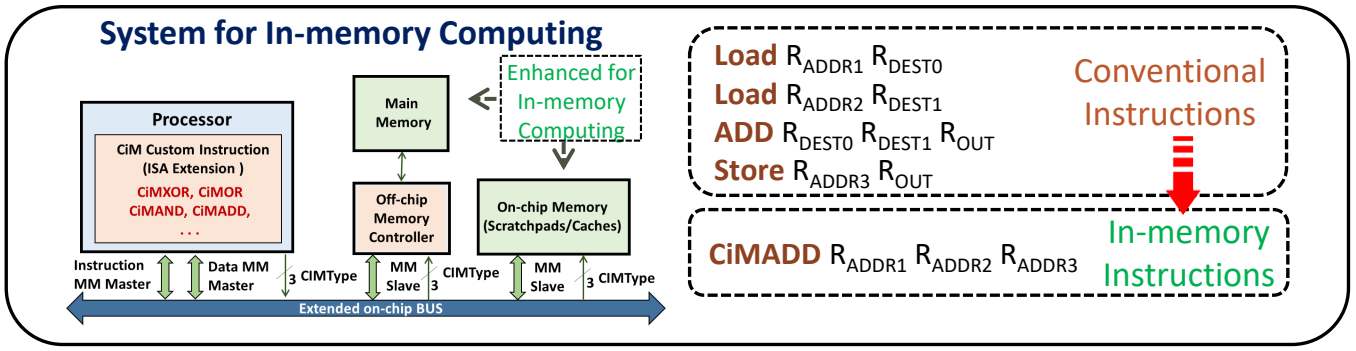
X-SRAM: Bit-Wise Vector Boolean Operations



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



Single cycle Read-Compute-Store

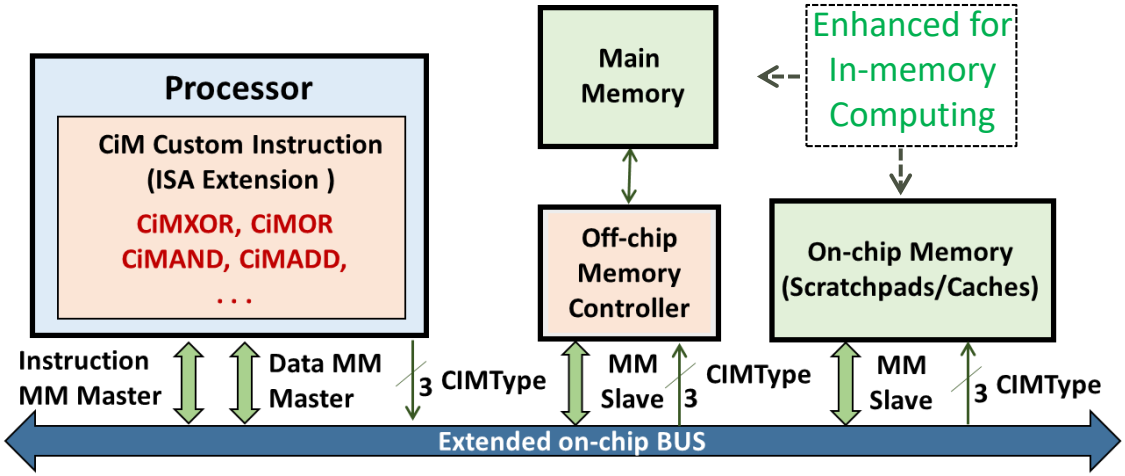


Agrawal, A et al., 2018. X-sram,. IEEE TCAS-I

C-BRIC

X-SRAM: Bit-Wise Vector Boolean Operations

System for In-memory Computing



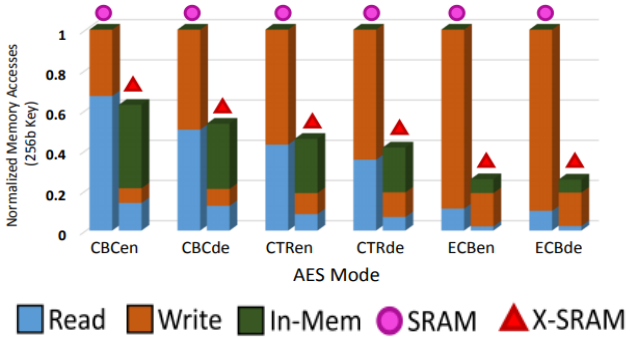
Conventional Instructions

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

In-memory Instructions

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

Upto 2x energy benefits, and 8x latency benefits for Binary/XNOR-Net Image classification

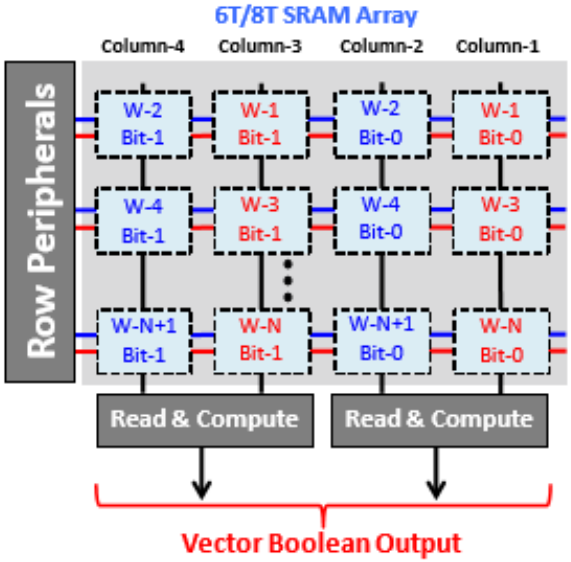


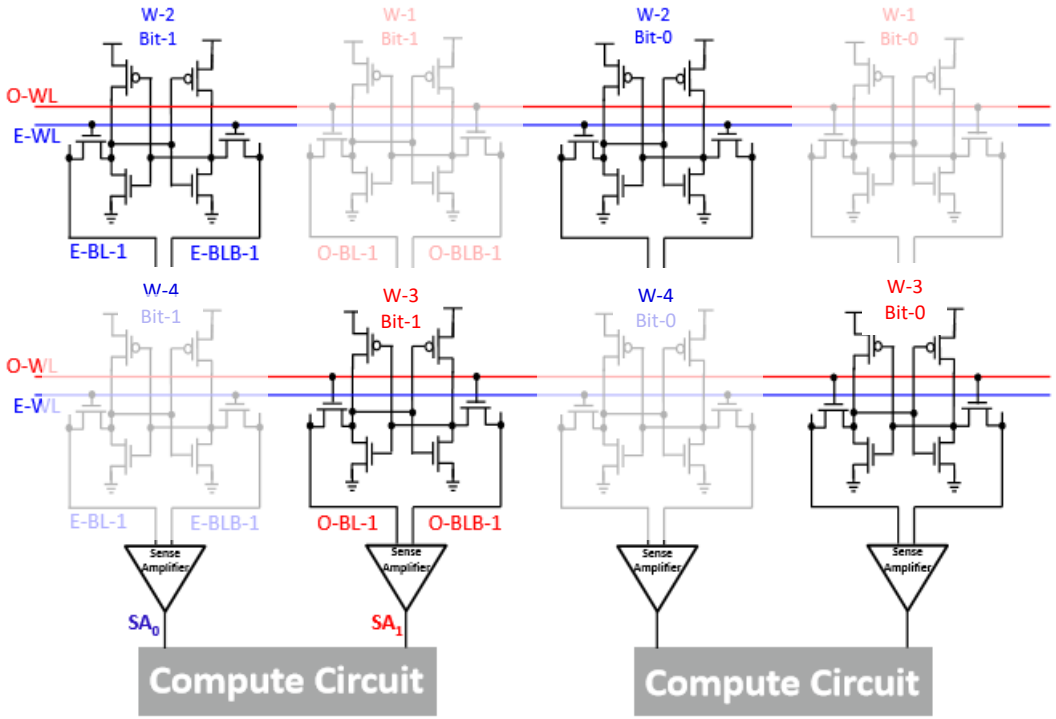
Upto 75% lower memory accesses on AES encryption application

Agrawal,Jaiswal,Roy TCAS-I, 2018

C-BRIC

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





8T-SRAM

- 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).

C-BRIC

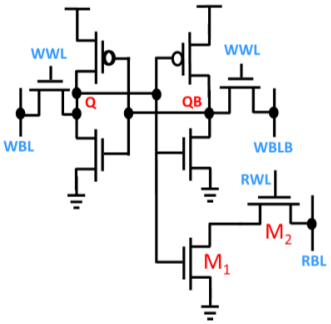
In-Memory Computing…SRAMs, DRAMs..

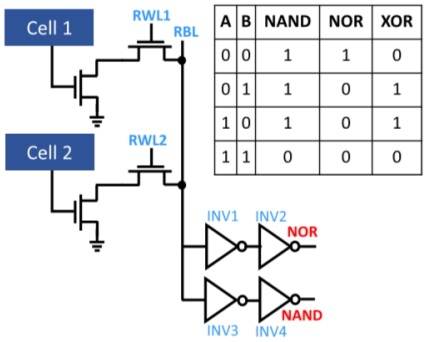
Look up table

Dot Product

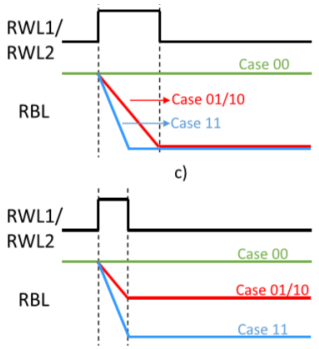
Bit-Wise Boolean

X-SRAM





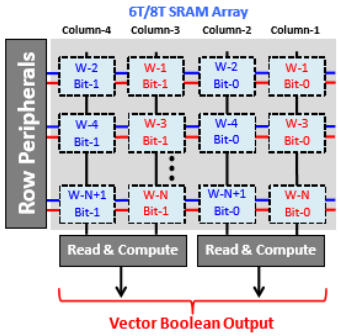
A	B	NAND	NOR	XOR
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	0	0

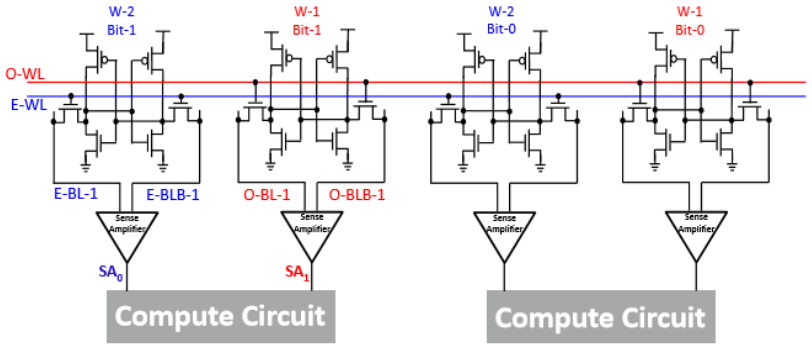


- 2 Wordlines Activated
- Bitwise NAND/NOR/XOR

Agrawal, Amogh, et al. "X-sram: Enabling in-memory boolean computations in cmos static random access memories." *IEEE Transactions on Circuits and Systems I: Regular Papers* 65.12 (2018): 4219-4232.

i-SRAM



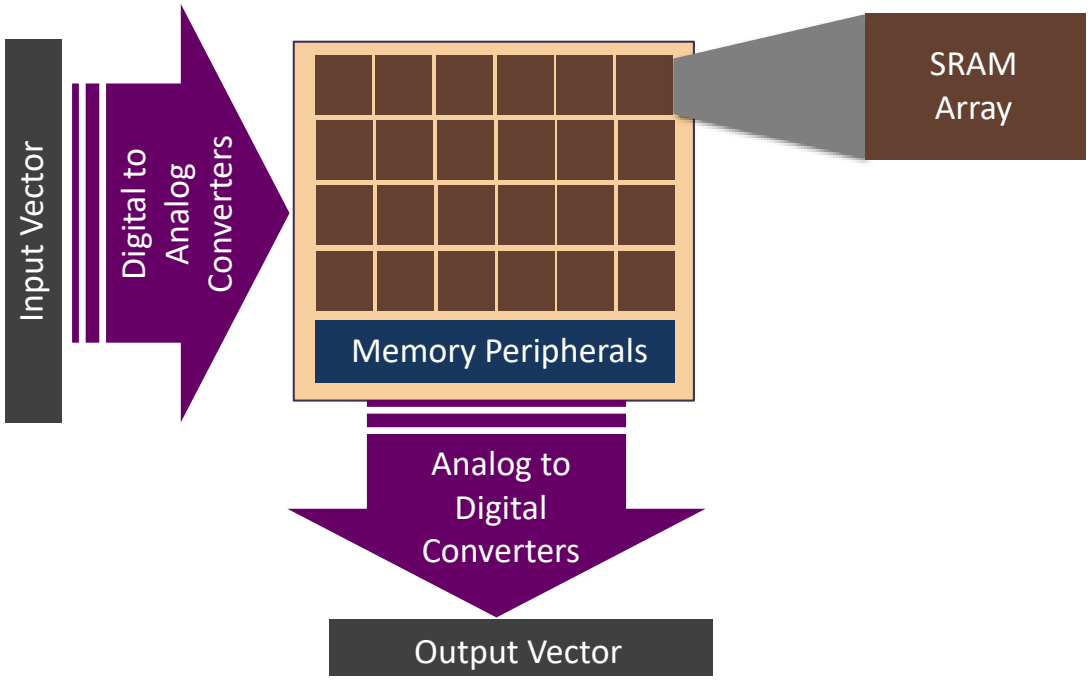


- Interleaved Wordlines
- Bitwise NAND/NOR/XOR

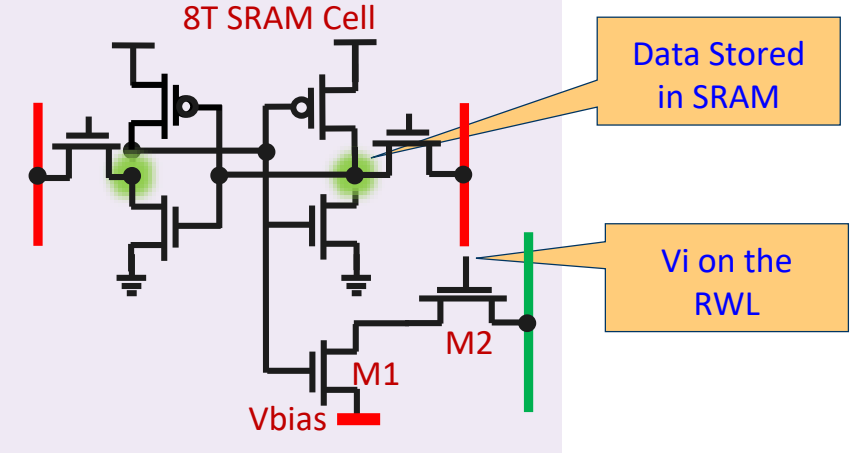
C-BRIC

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

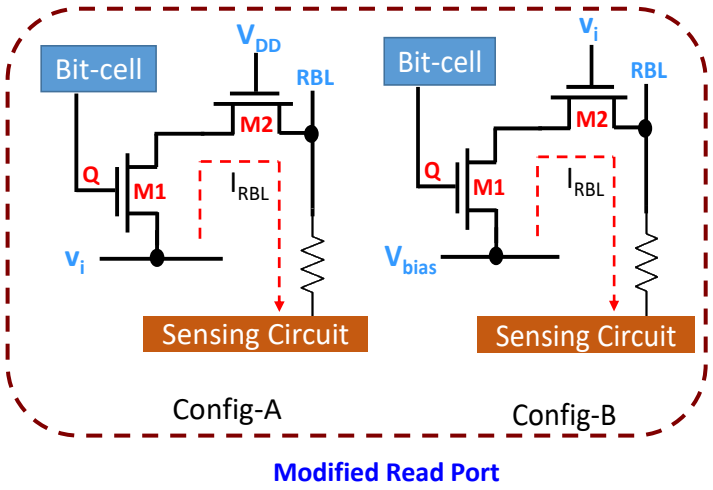


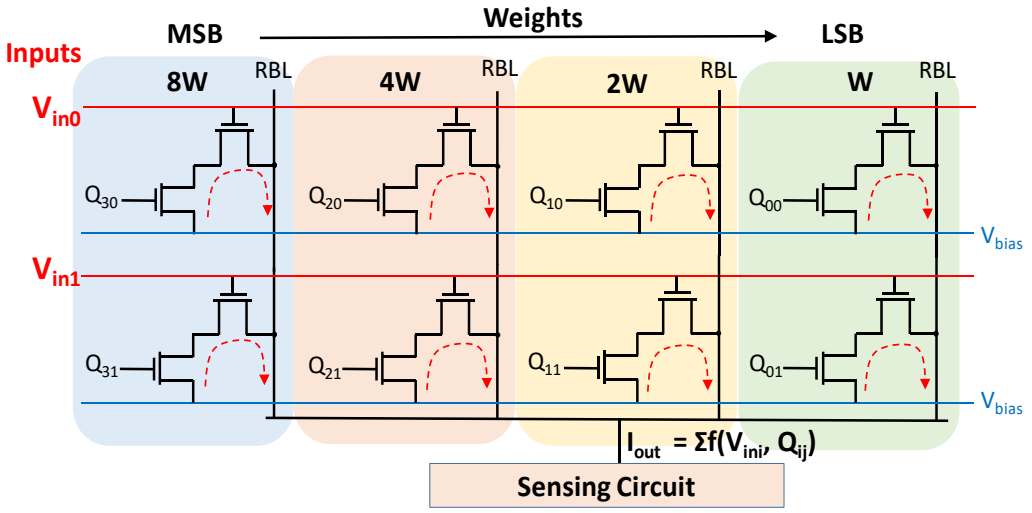
The Dot Product = $\Sigma V_i.W_i$

Vi: Analog Voltages on RWL or SL

Wi: Data Stored in SRAM

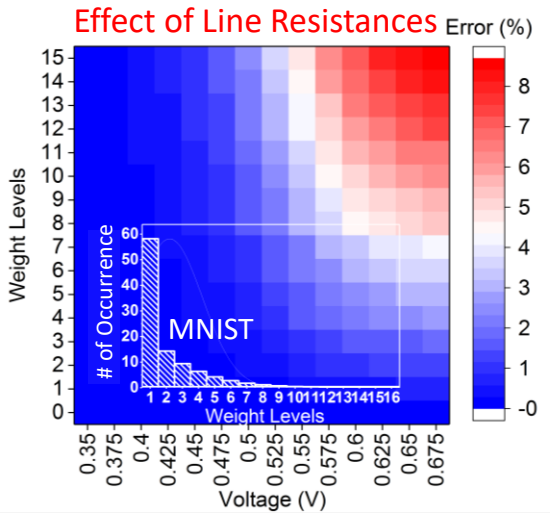
Summation: KCL addition

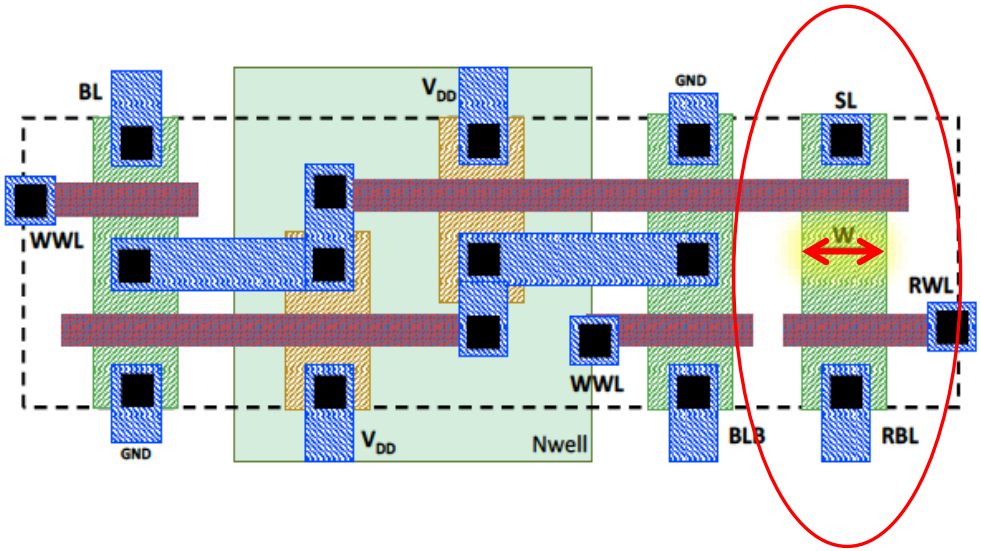




A. Jaiswal, I. Chakraborty et al , under review in TVLSI, 2018

8T SRAM as a Multi-Bit Dot Product Engine





- Profiling the neural network reveals most weight levels at 0.
- This means current levels are low, mostly in the BLUE region in the error plot.
- However, there is a ~30% increase in bit-cell area.
- Multi-VT design can be exploited to reduce area.

A. Jaiswal, I. Chakraborty et al , TVLSI, 2019

C-BRIC

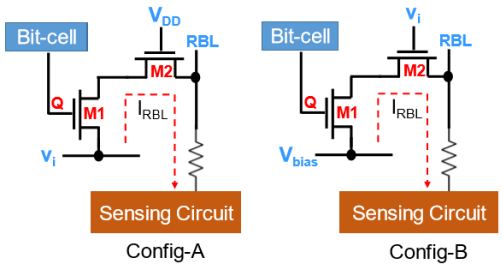
In-Memory Computing…SRAMs, DRAMs..

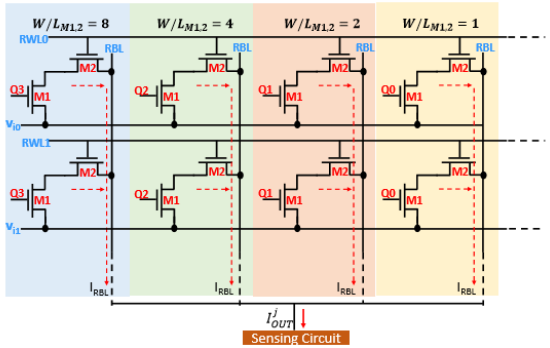
Look up table

Dot Product

Bit-Wise Boolean

8T Dot Product Engine

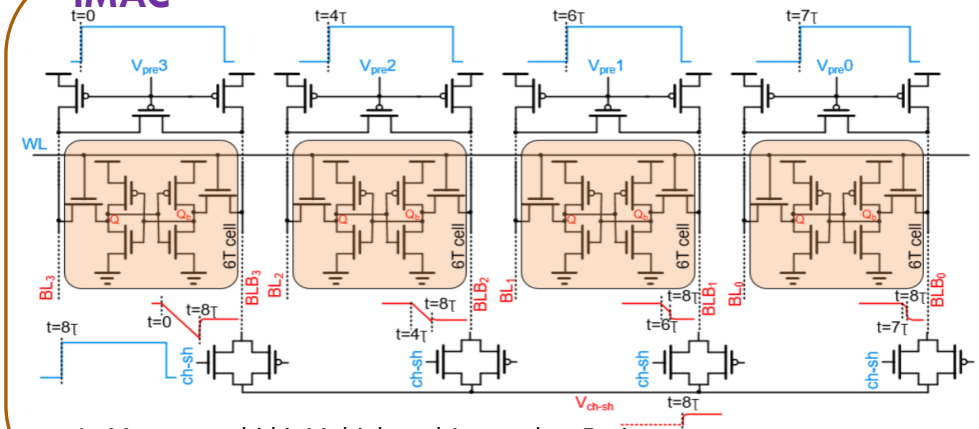




- Multi-bit Dot Products
- Highly Parallel Operations

Akhilesh Jaiswal, Indranil Chakraborty, Amogh Agrawal, Kaushik Roy, "**8T SRAM Cell as a Multi-bit Dot Product Engine for Beyond von-Neumann Computing**"arXiv:1802.08601v2 [cs.ET]

IMAC

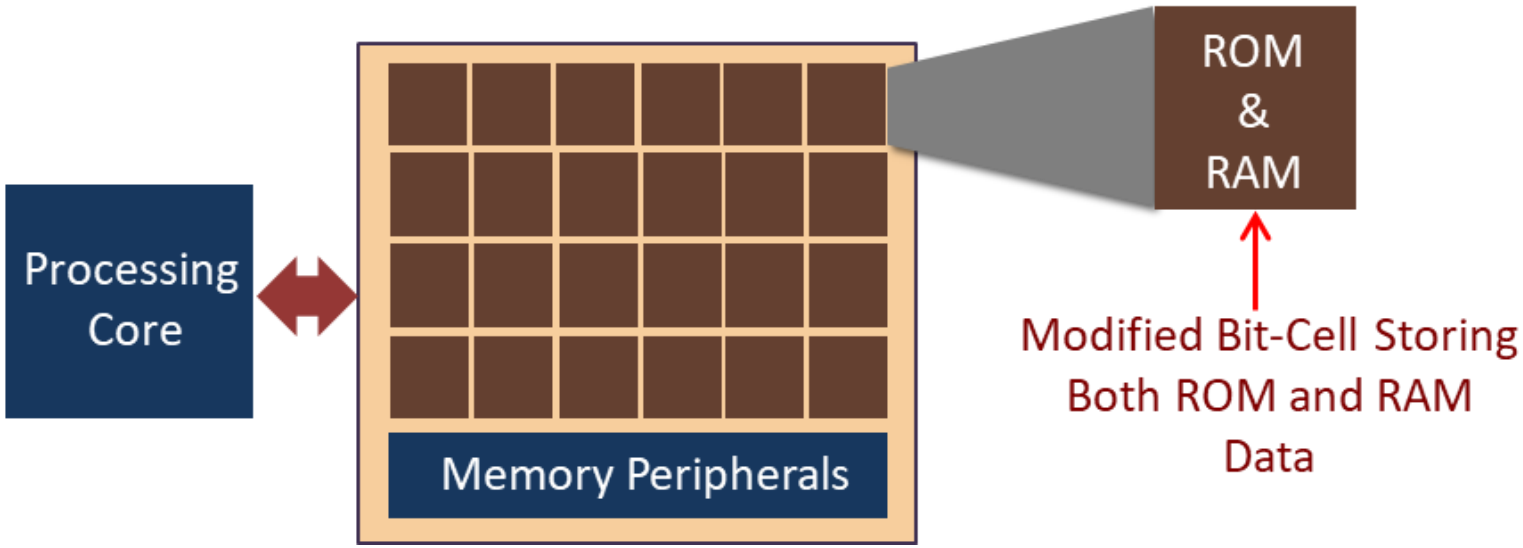


- In-Memory multi-bit Multiply and Accumulate Engine
- Analog Multiplication in Functional read followed by analog accumulation

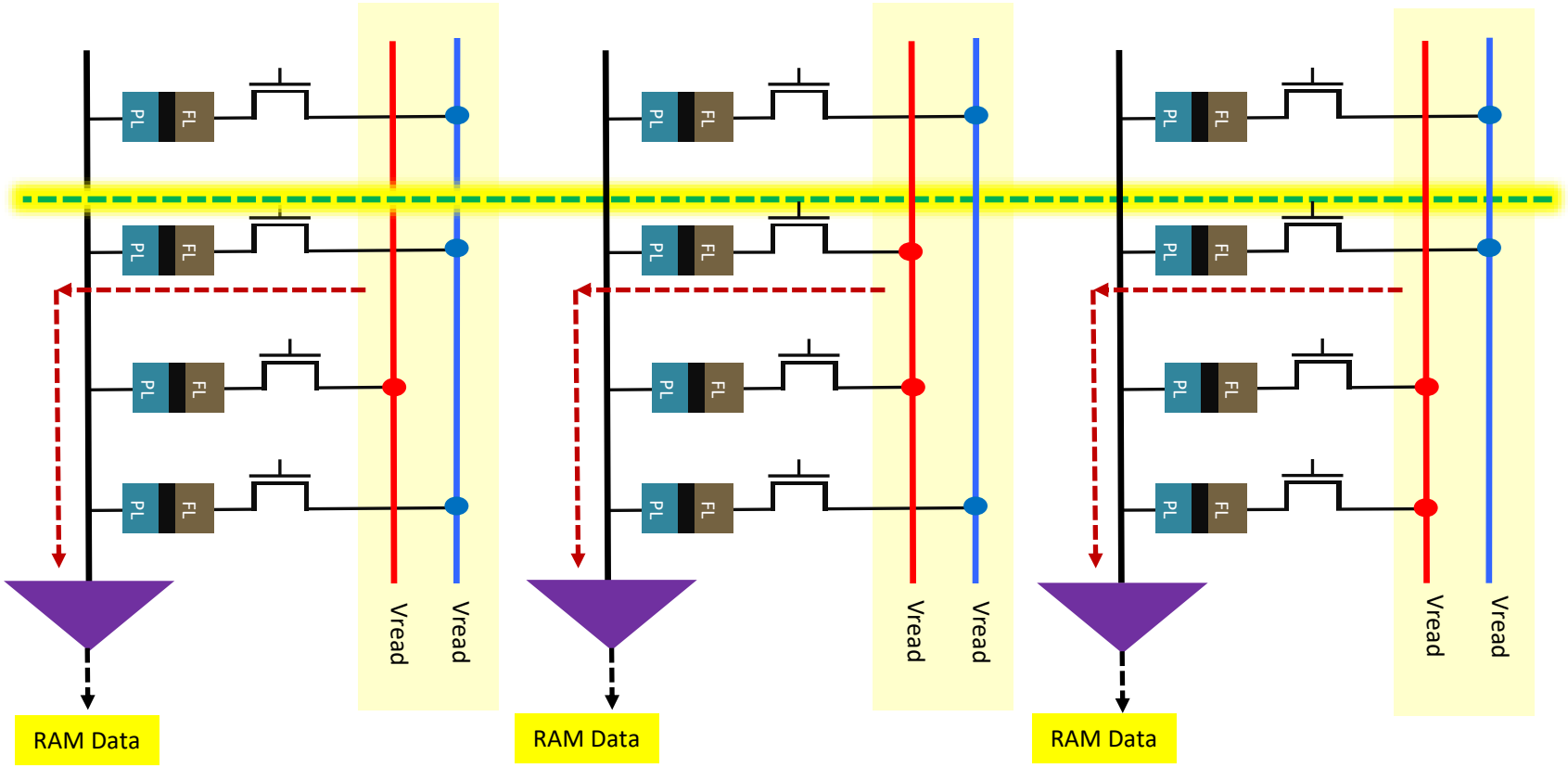
C-BRIC

ROM Embedded RAM

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



Embedding ROMs in RAMs (NVMs)

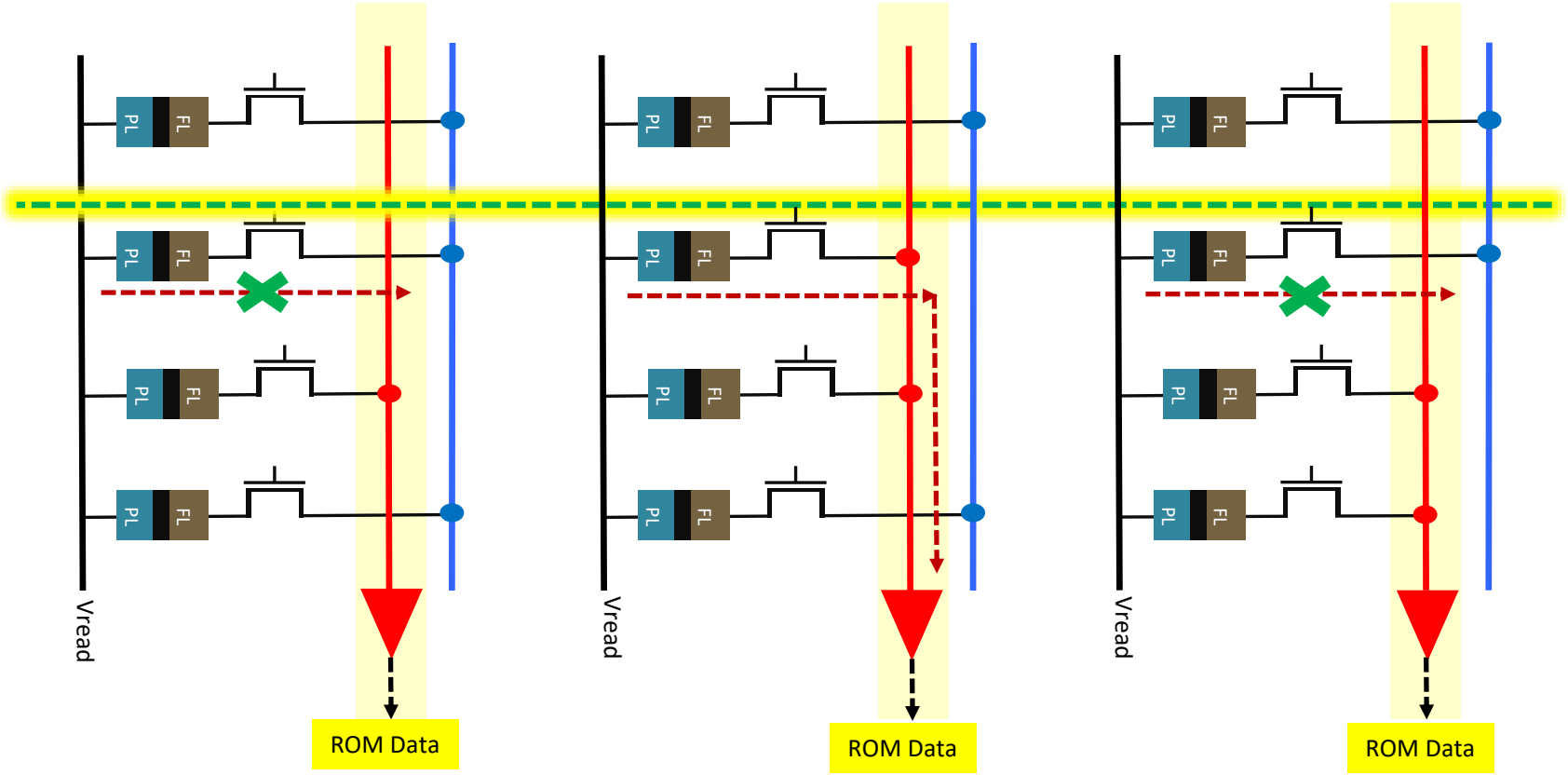


Both ROM and RAM data are stored in the same bit-cell. The read cycle determines whether the ROM or the RAM data is being read.

D. Lee, K. Roy, IEEE EDL 2013

C-BRIC

Embedding ROMs in RAMs (STT-MRAM)



D. Lee, K. Roy, IEEE EDL 2013

C-BRIC

Computing in DRAM

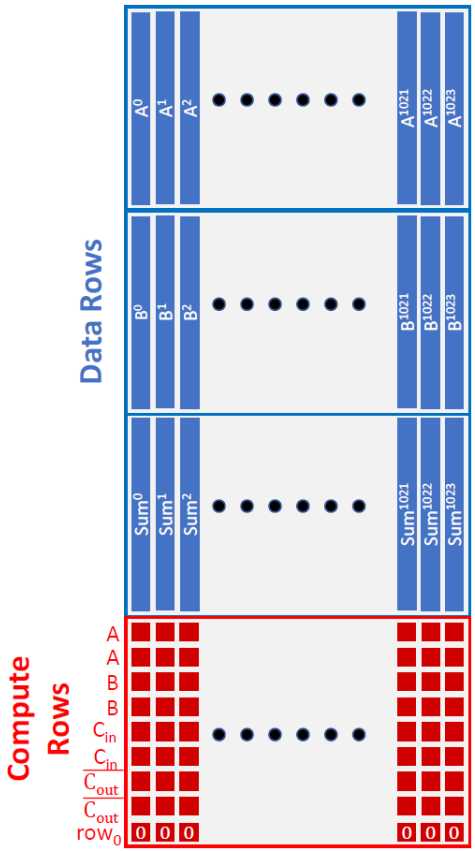
In-DRAM Low-cost Bit-serial Addition

- **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)**



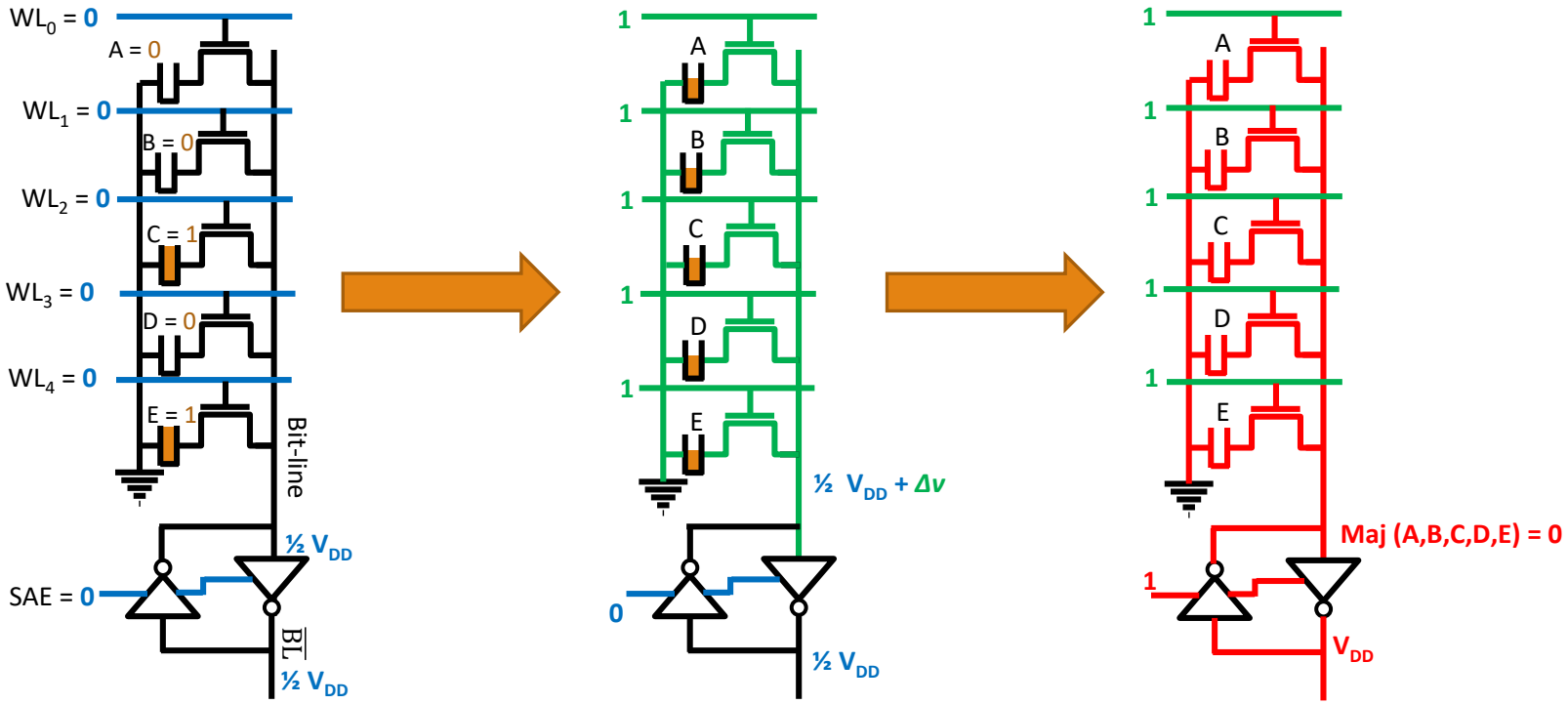
- **The result of A+B addition is stored in the same column**
- **Massively-parallel vector additions in bit-serial mode**
- **No need for carry shifts across bitlines!**

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

C-BRIC

In-DRAM Low-cost Bit-serial Addition

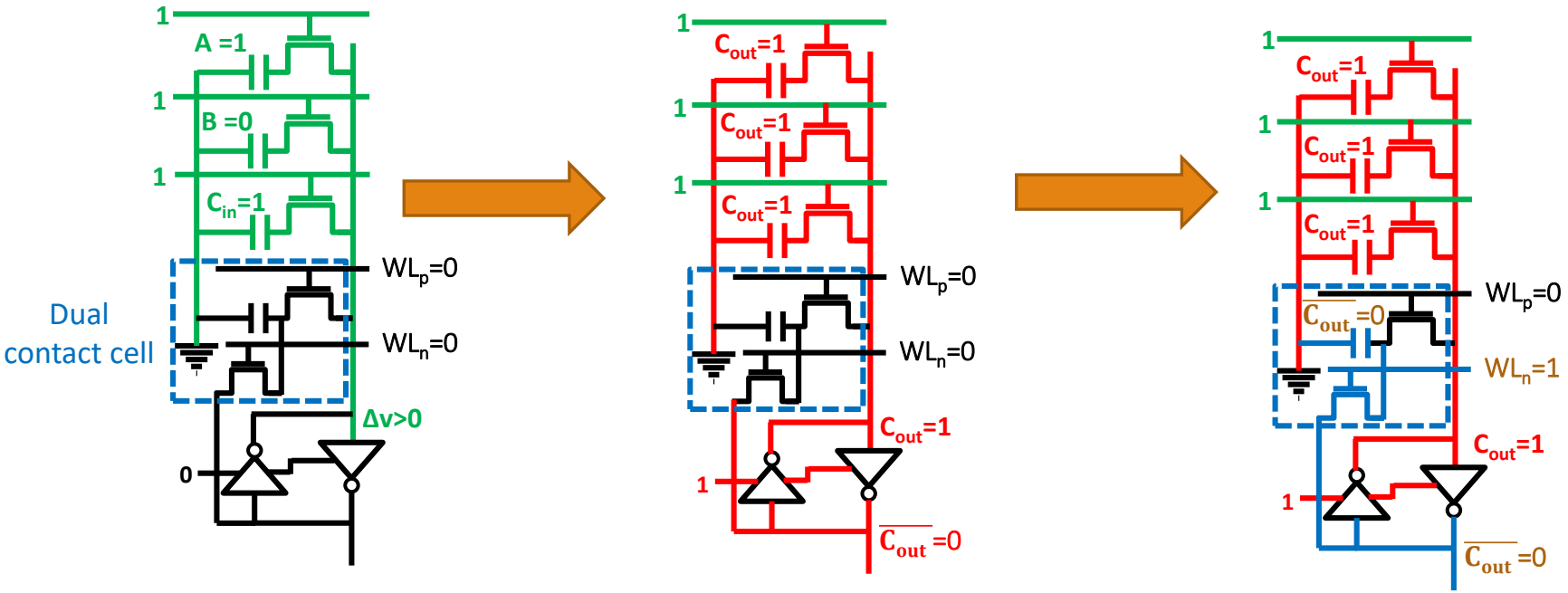
Multiple row activation to calculate Maj(A,B,C,D,E)



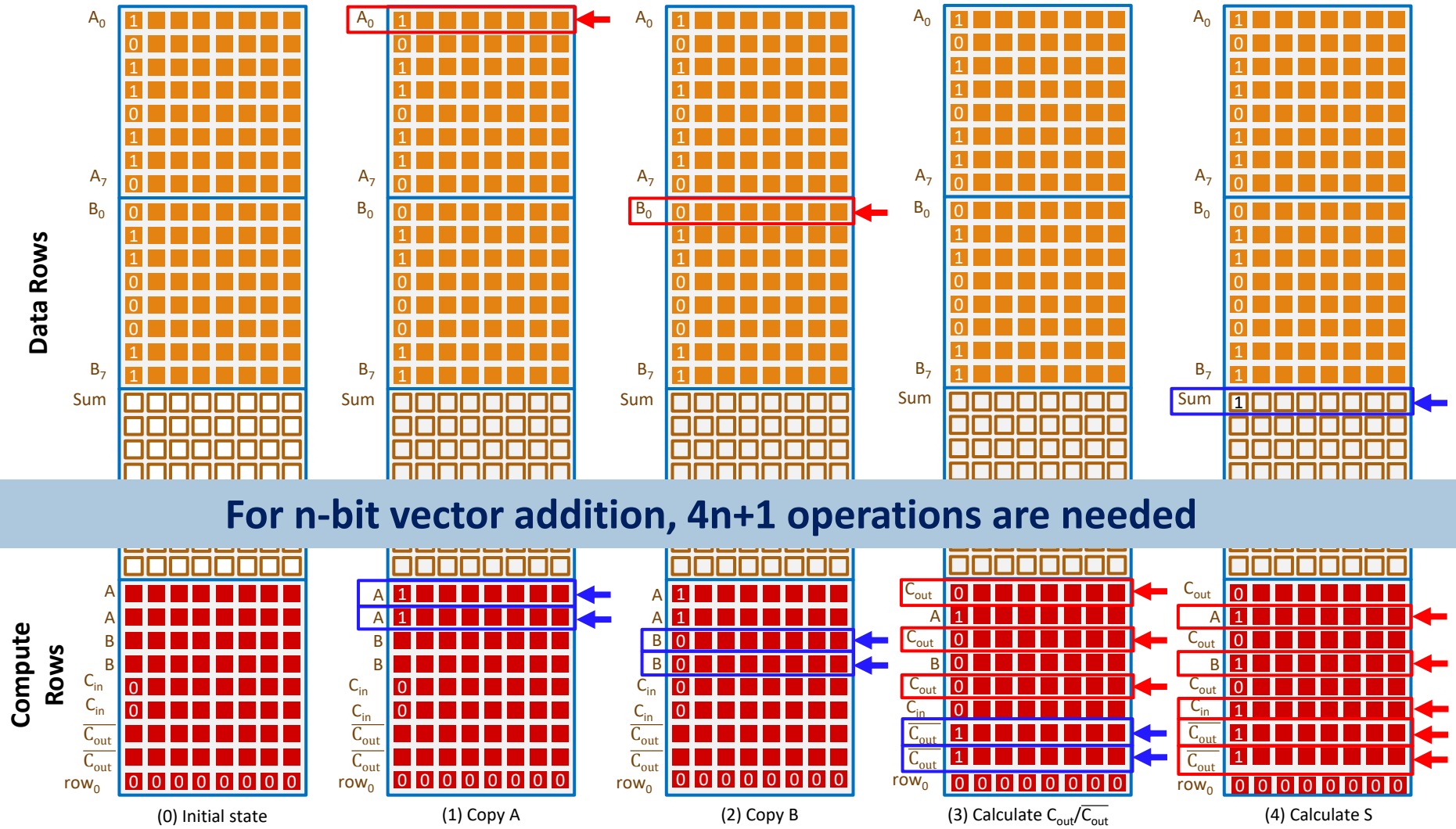
In-DRAM Low-cost Bit-serial Addition

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

Calculate C_{out} and , $\overline{C_{out}}$ in one step



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

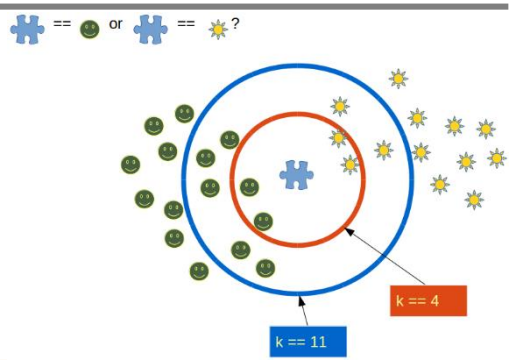


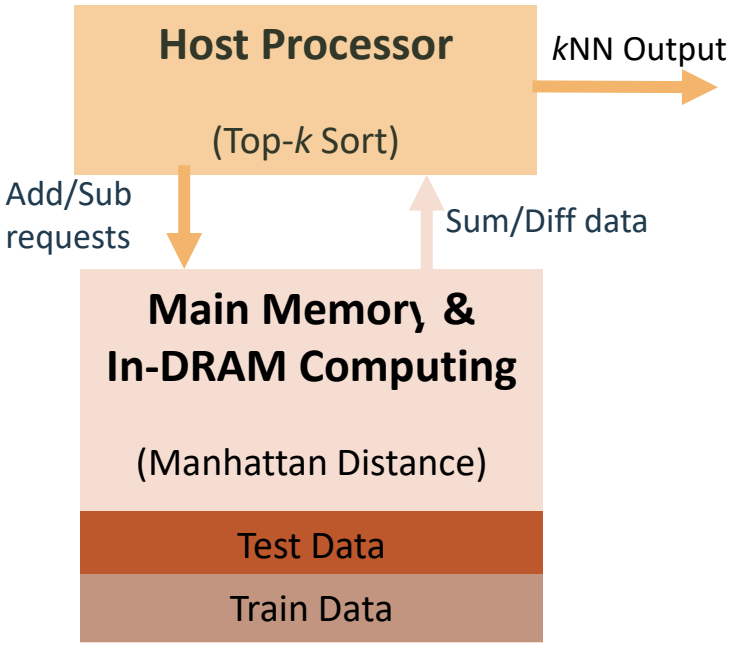
For n-bit vector addition, 4n+1 operations are needed

C-BRIC

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

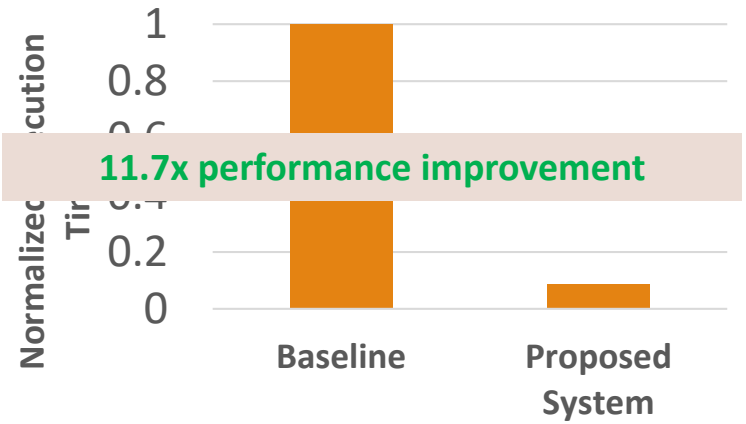
- Checks *k* closest samples, Query vector assigned to the group that holds majority among those k samples
- Mainly consists of two computation stages: Distance computation and Global top-k sort
- The k value: The number of closest samples to be checked
- One-to-one distance computation: a large # of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





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*



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

C-BRIC

Conclusions

- While possibilities of achieving large improvements in inference latency and energy is possible…
- There are several challenges…
  - 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

C-BRIC

Neuro-electronics Research Laboratory

C-BRIC