北京大学
PEKING UNIVERSITY

AI ASIC: Design and Practice (ADaP)

Fall 2023

Processing-In-Memory

燕博南

Outline

- Motivation
- Basic Structure of PIM
- PIM/CIM Behavior Model

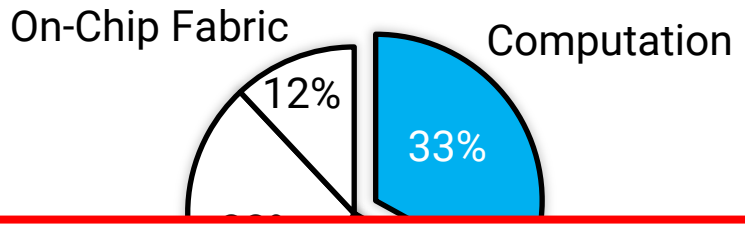
北京大学
PEKING UNIVERSITY

Why Fused Memory & Processing Units?

Why Does Processing-In-Memory (PIM) Rise?

PEKING UNIVERSITY

Power

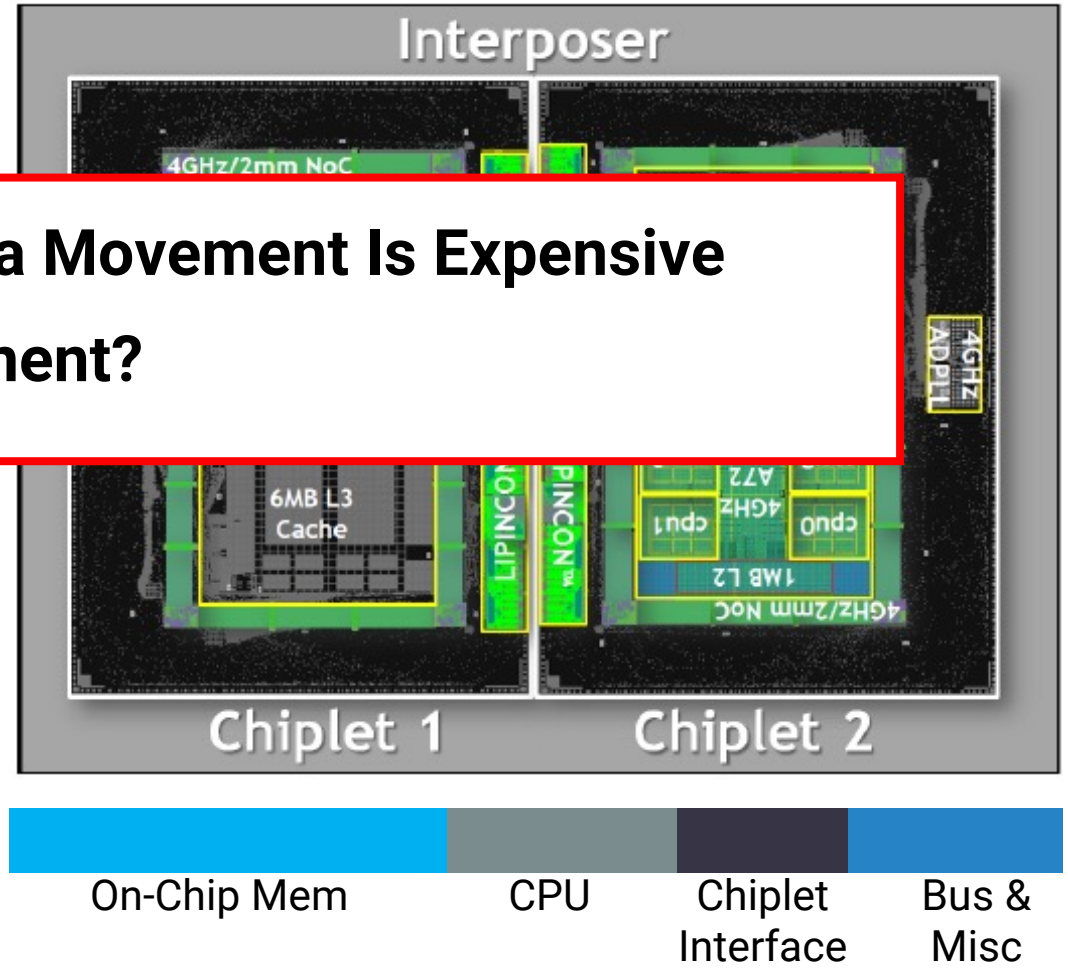


- **Memory is the Bottleneck; Data Movement Is Expensive**
- **How to reduce the data movement?**

Energy

Operations (32bit)	Energy (pJ)
Int ADD	0.1
Float ADD	0.9
Register File	1
Int Multiply	3.1
Float Multiply	3.7
SRAM Cache	5
DRAM Memory	640

Area



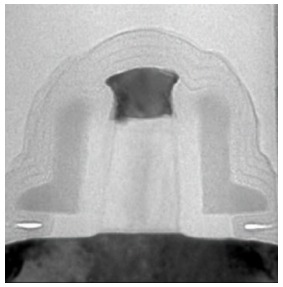
Source: AMD, Intel, [Mu-Shan Lin, et al, JSSC 2019]

Specialized Hardware Enhances Efficiency

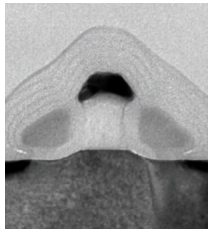
$$P = \alpha C V_{DD}^2 f$$

P: Power Dissipation
α : Activity Factor
C: Load Capacitance
V_{DD}: Power Supply
f : Clock Frequency

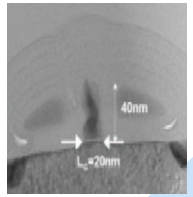
50nm L_G



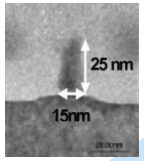
30nm L_G



20nm L_G



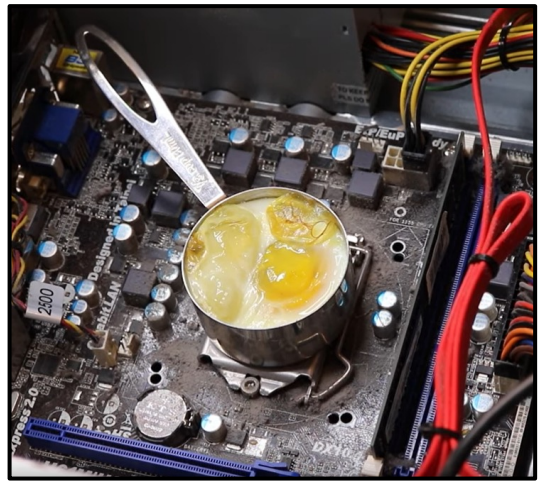
15nm L_G



MOSFET Scaling

Transistor Size & Clock Frequency

Multicore to Manycore



Domain-Specific

groq

Cambricon 寒武纪

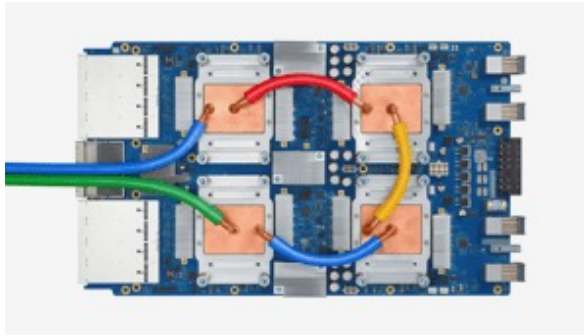
MYTHIC

habana An Intel Company

GRAPHCORE

arm

NVIDIA



Google TPU v3
~200Watts
(4 TPU to run AlphaGo, 300x less power consumption)

...

Source: Vasileska et al., JTCN, 2008, Google, https://www.youtube.com/watch?v=xL40UN5p6IY&t=673s

Uniqueness of Neural Network Execution

Multiply-Accumulate (MAC):

inputs

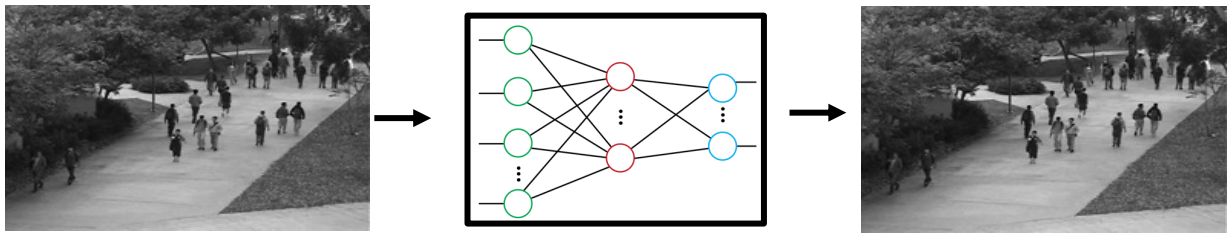
weights

outputs

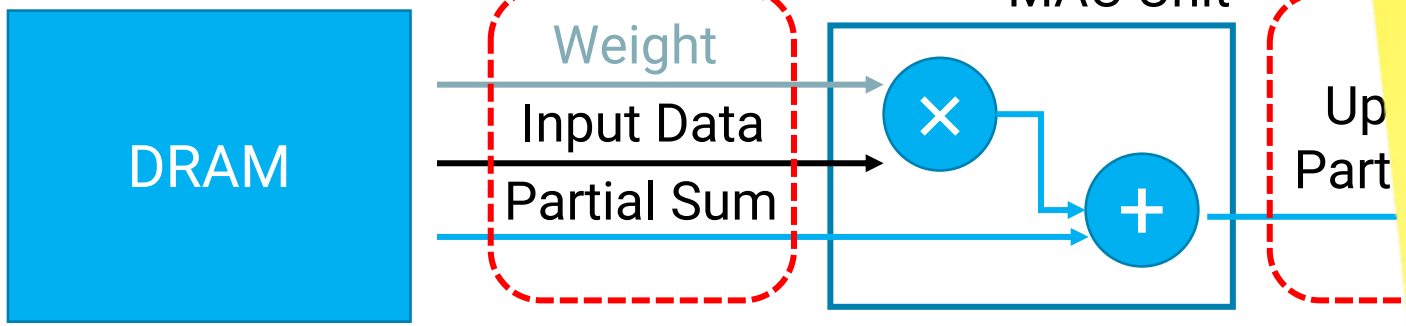
streaming

stationary

$$[x_1\ x_2\ \dots\ x_m]\begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & & \vdots \\ w_{m1} & \cdots & w_{mn} \end{bmatrix} = [y_1\ y_2\ \dots\ y_n] \quad \Rightarrow \quad y_j = \sum_{i=1}^{m} x_i \cdot w_{ji}$$

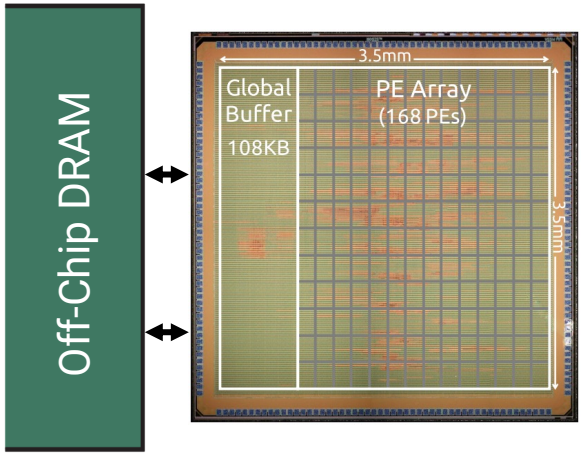
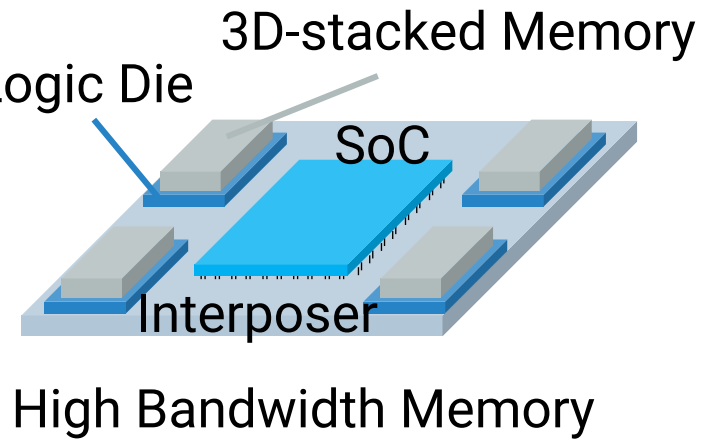
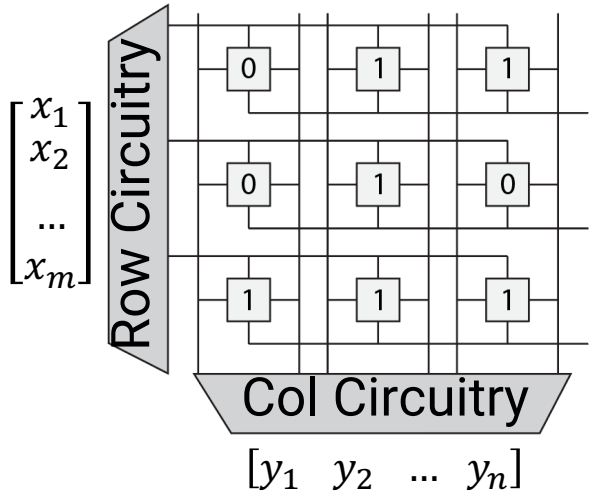


Execution:



Inference:
Inputs change;
weights stay.
How to fix weights to where MAC Units without moving?

Make Memory Access Less Expensive

	CMOS Accelerator (Planar Integration)	Near-Memory Computing	In-Memory Computing
Design Approach			
Energy to Access Weights	200×~6× of MAC Energy	55×~6× of MAC Energy	**0**

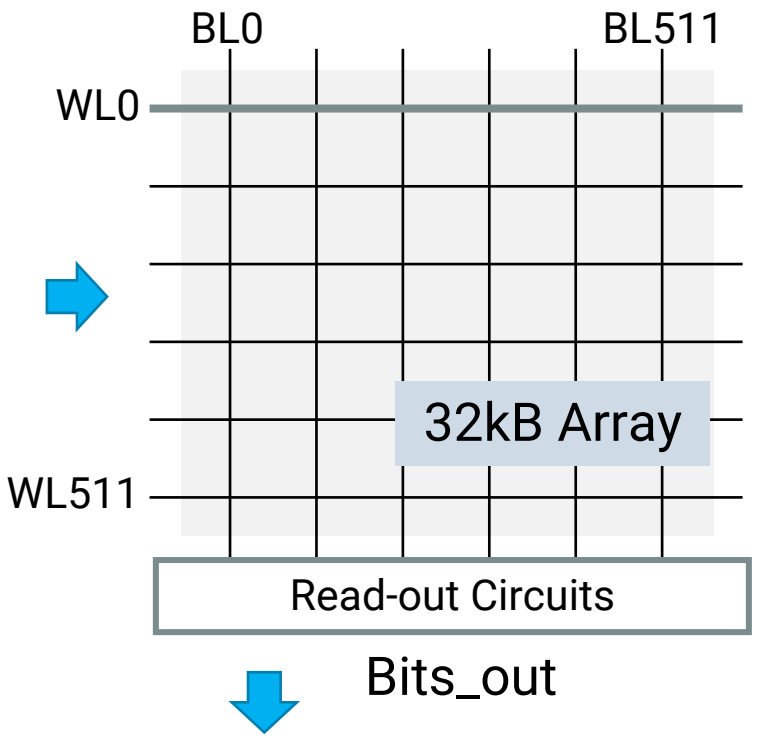
DRAM On-Chip Buffer

Source: AMD, Y.-H. Chen, JSSCC,2017

From Memory to In-Memory Computing

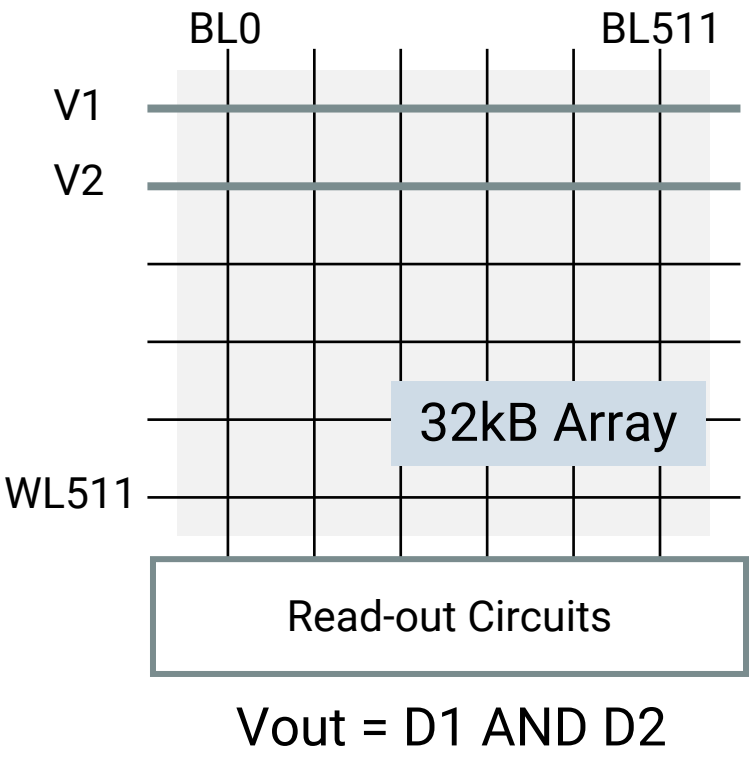
北京大学 PEKING UNIVERSITY

Typical Memory Access



Bits accessed: 1*512=512

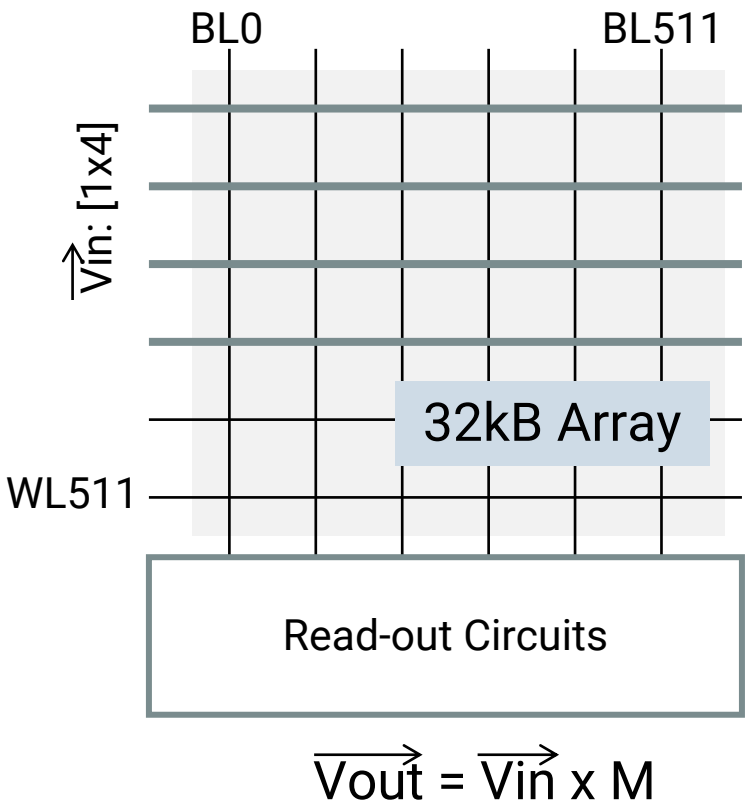
In-Memory Logic



Vout = D1 AND D2

Bits accessed: 2*512=1k

In-Memory General Matrix Multiplication (GEMM)



$\overrightarrow{Vout} = \overrightarrow{Vin} \times M$

Bits accessed: 4*512=2k

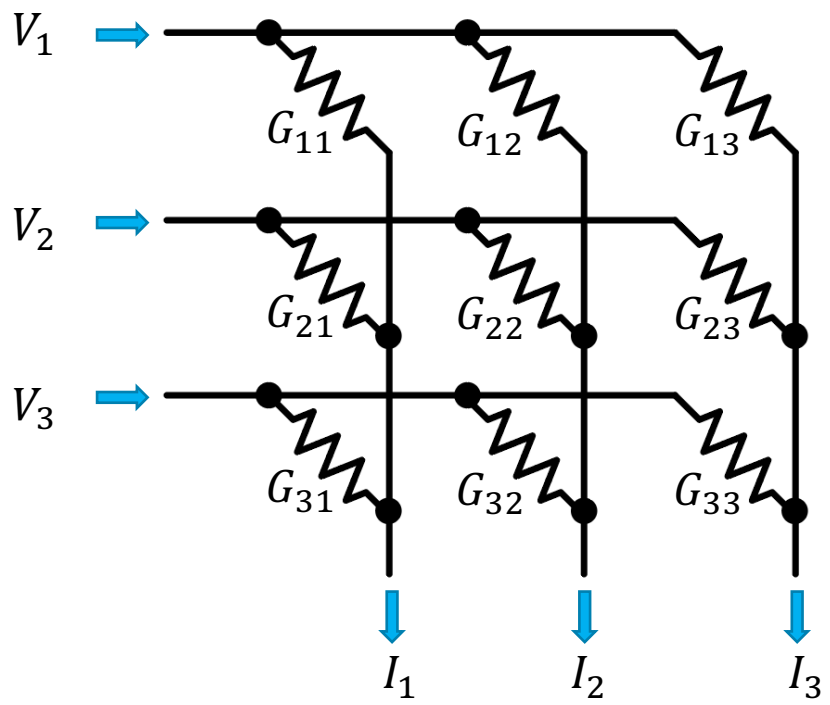
B. Yan, Adv. Intell. Sys., 2019

Basic Idea: Memristor PIM

Basic Idea: Resistivity Memory PIM

inputs weights outputs

$$[V_1 \quad V_2 \quad V_3]\begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} = [I_1 \quad I_2 \quad I_3]$$

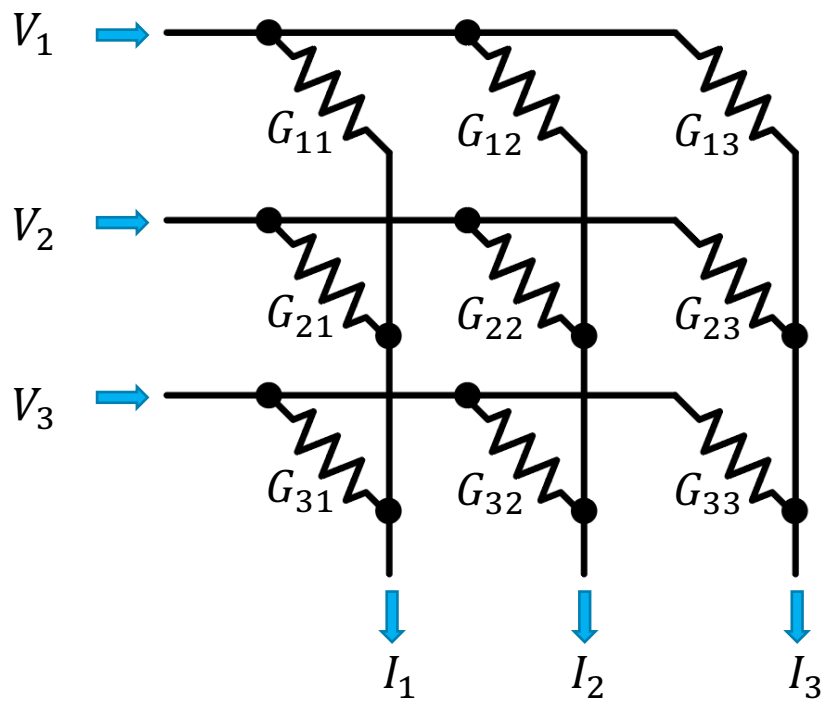


- Weight Matrix Stored as Conductance $\boldsymbol{G}$
- Rely on Analog Computation (Kirchhoff's Current Law) for "almost free"
  - Multiplication: $\mathrm{I} = V \cdot G$
  - Addition: $I^{column} = I_1^{row} + I_2^{row} + I_2^{row}$
- Ideal Nanoscale Devices for $\boldsymbol{G}$:
  - Programmable Conductance
  - Multi-Level Cell
  - Small Footprint/High Density
  - Compatible with Existing CMOS Process

Basic Idea: Resistivity Memory PIM

inputs weights outputs

$$[V_1 \quad V_2 \quad V_3]\begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} = [I_1 \quad I_2 \quad I_3]$$



- Q: Can we use currents as the inputs?

$$[I_1 \quad I_2 \quad I_3]\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} = [V_1 \quad V_2 \quad V_3]$$

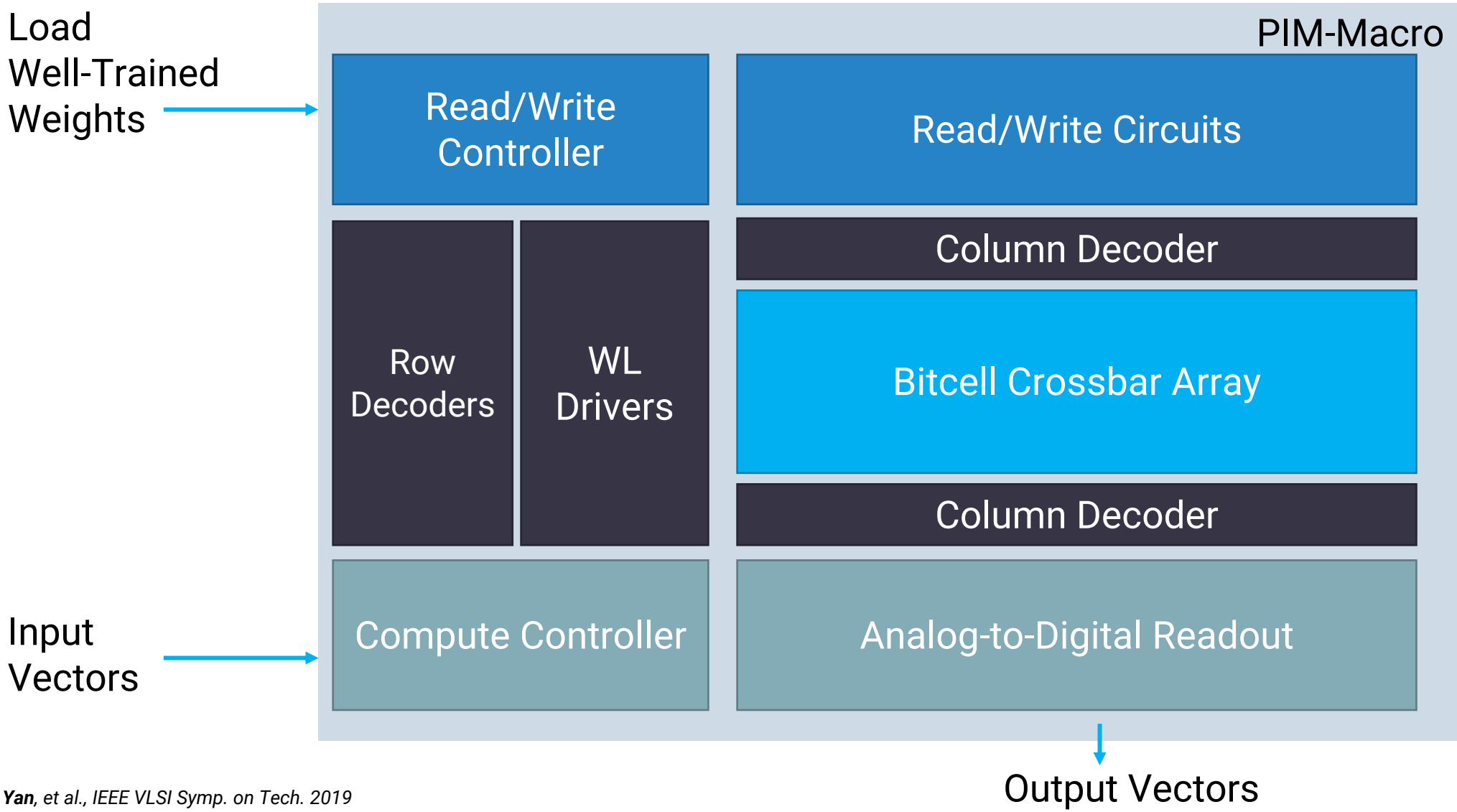
- Q: What circuits do we need? Why?

How A SoC Designer Sees PIM/CIM?

PEKING UNIVERSITY

- CIM Behavior Model

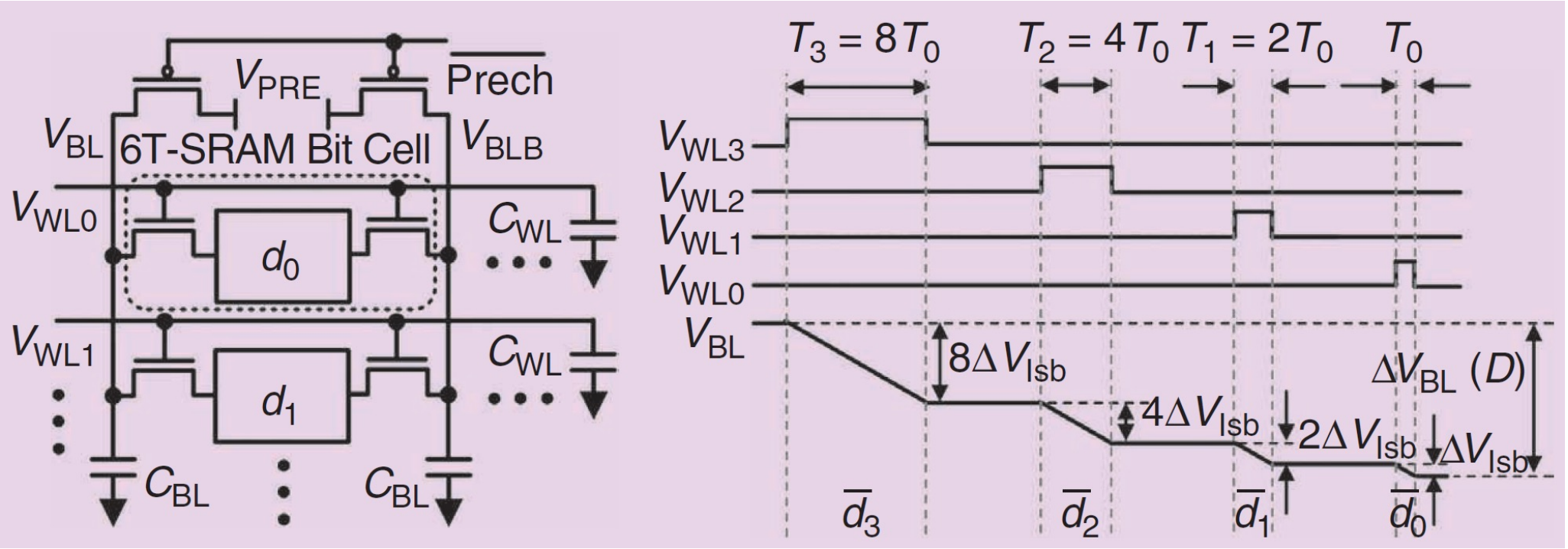
In-Memory Computing Solution Stack-Macro



***B. Yan**, et al., IEEE VLSI Symp. on Tech. 2019*

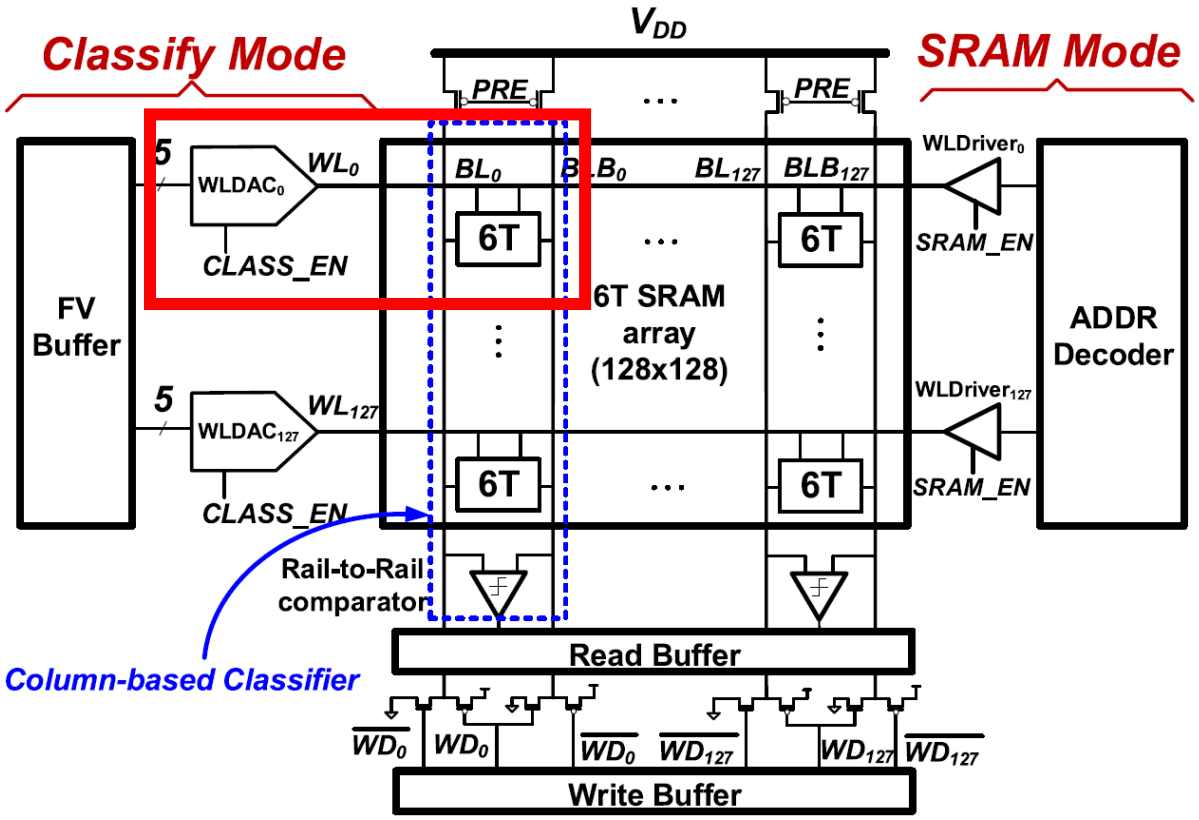
Basic Idea: SRAM PIM

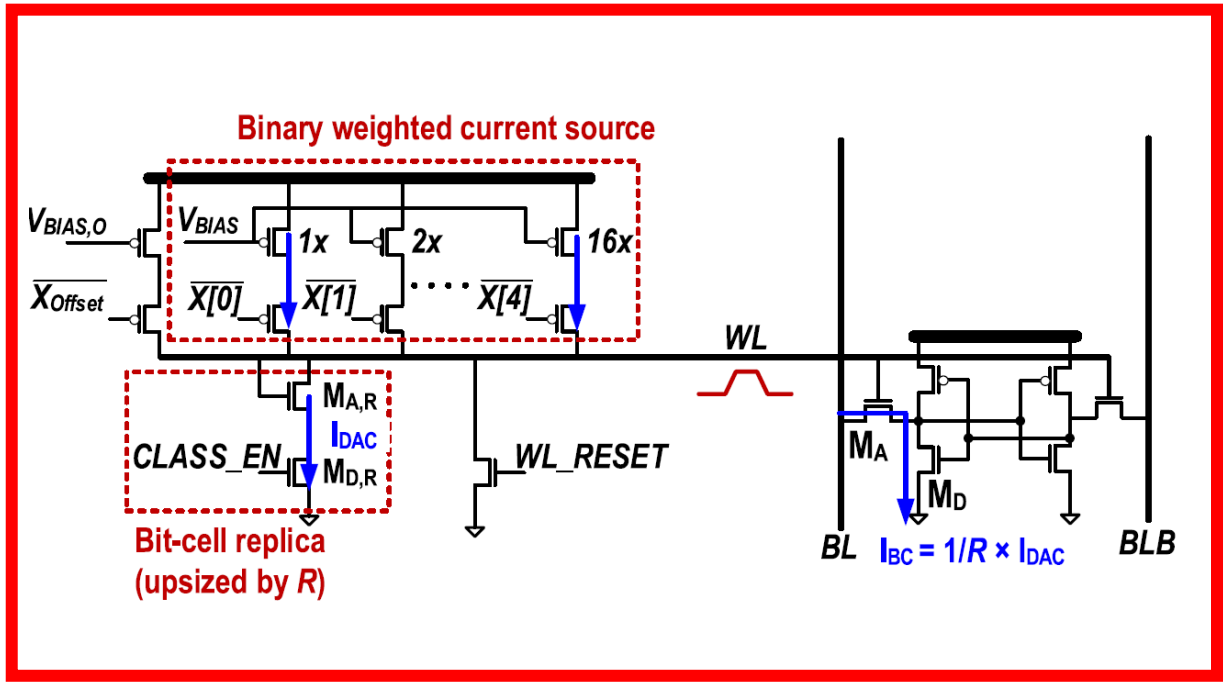
Basic Idea: SRAM PIM



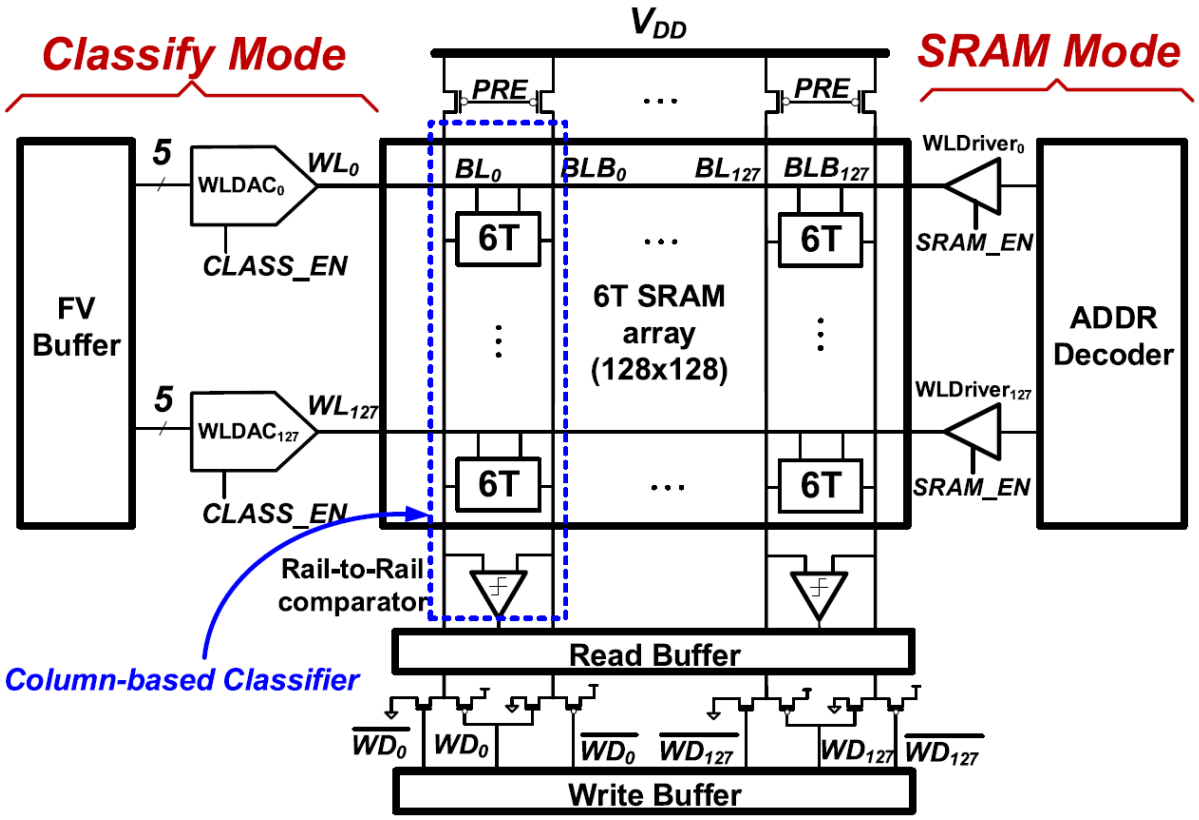
- Precharge BL
- Control the opening time windows of each WL

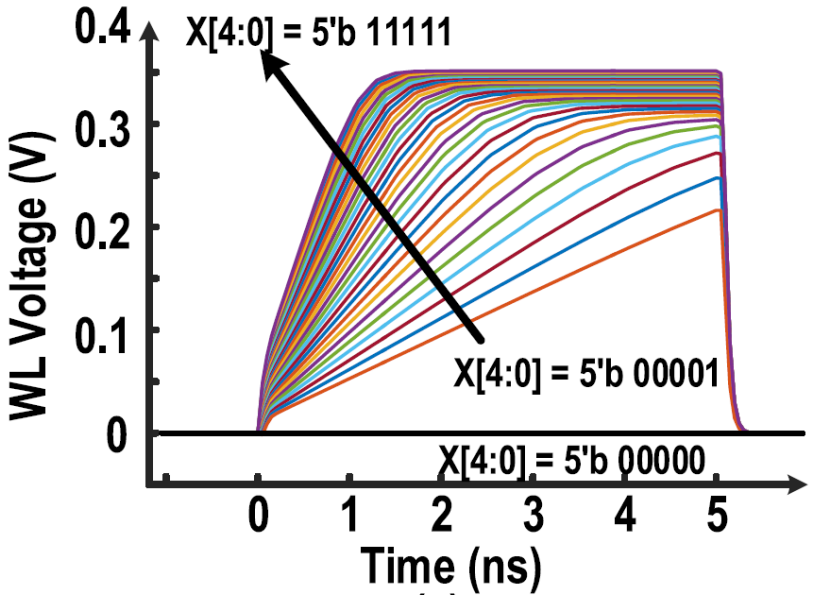
Basic idea: SRAM PIM

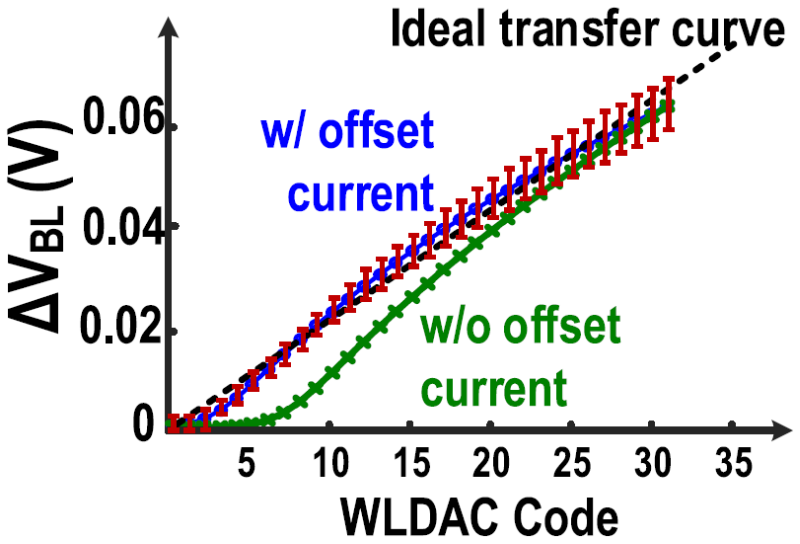




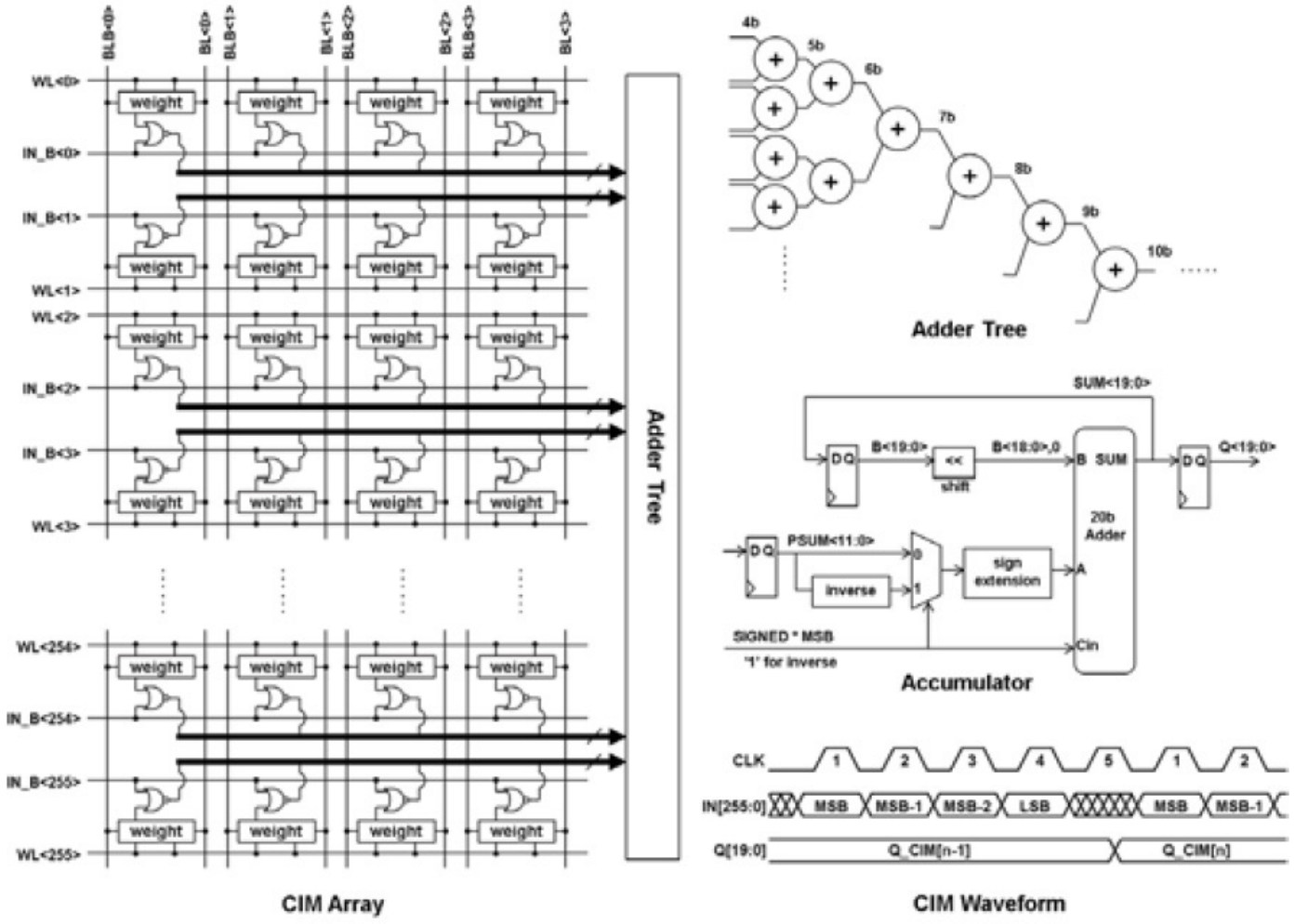
Basic idea: SRAM PIM







Basic Idea: Digital SRAM PIM

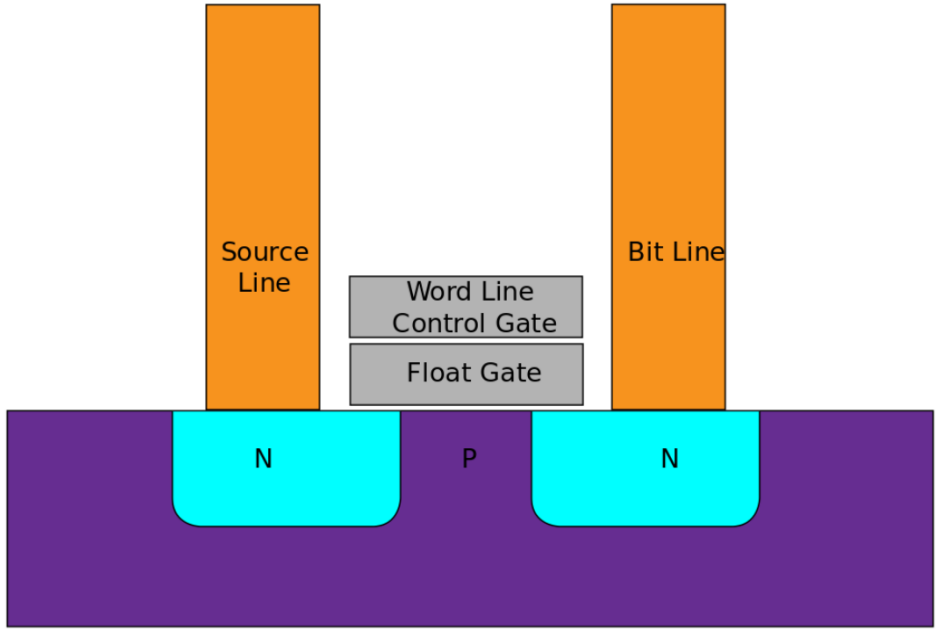


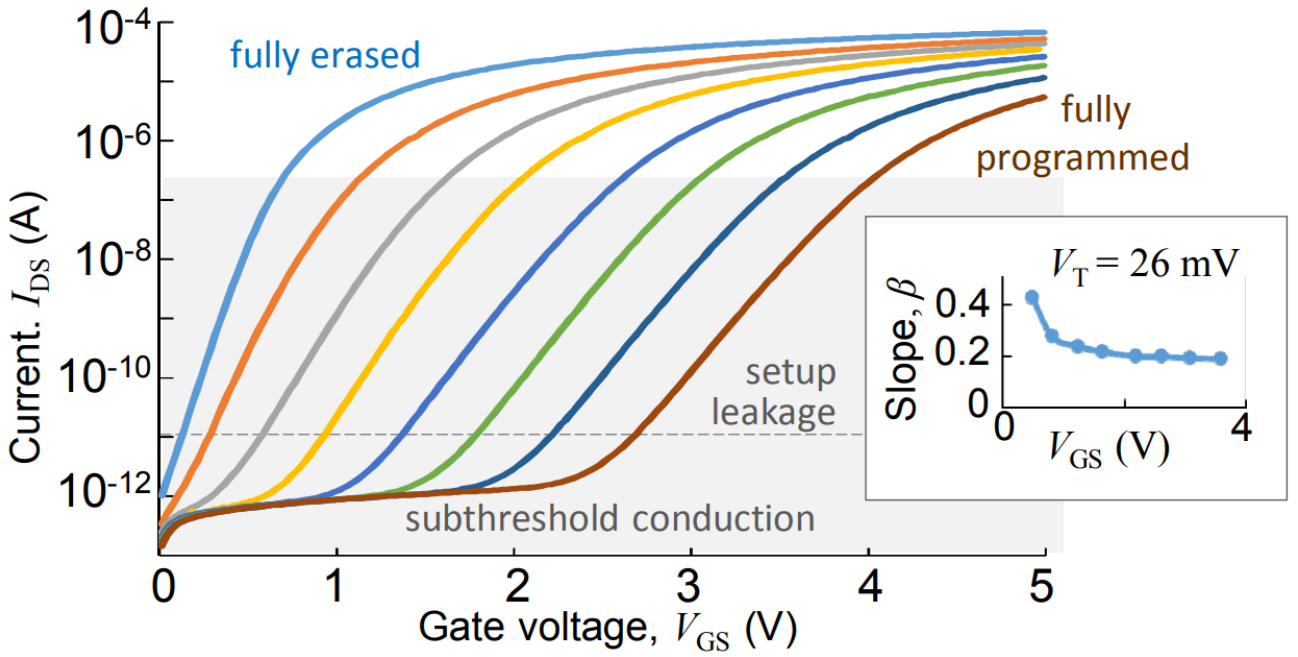
Why Digital PIM?

- Strength: High Noise Margin
- Weakness: Low parallelism

Basic Idea: Flash PIM

Basic Idea: Flash Device

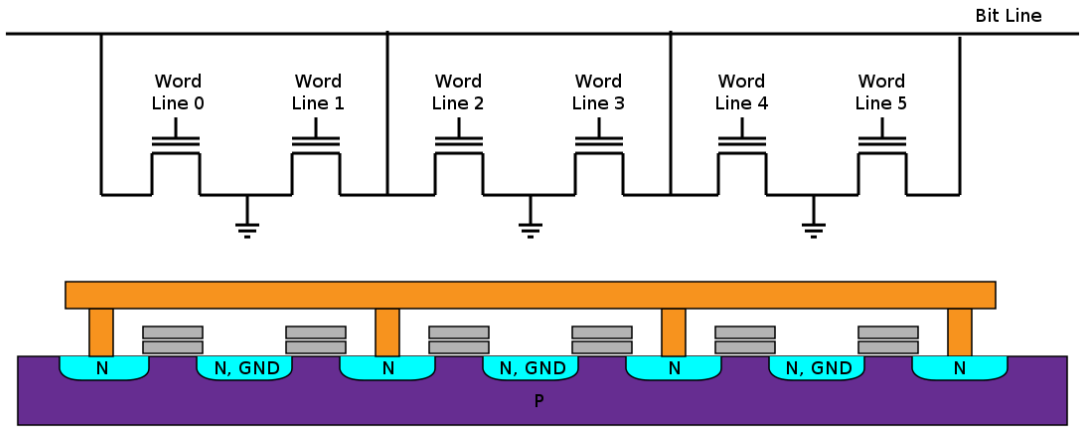




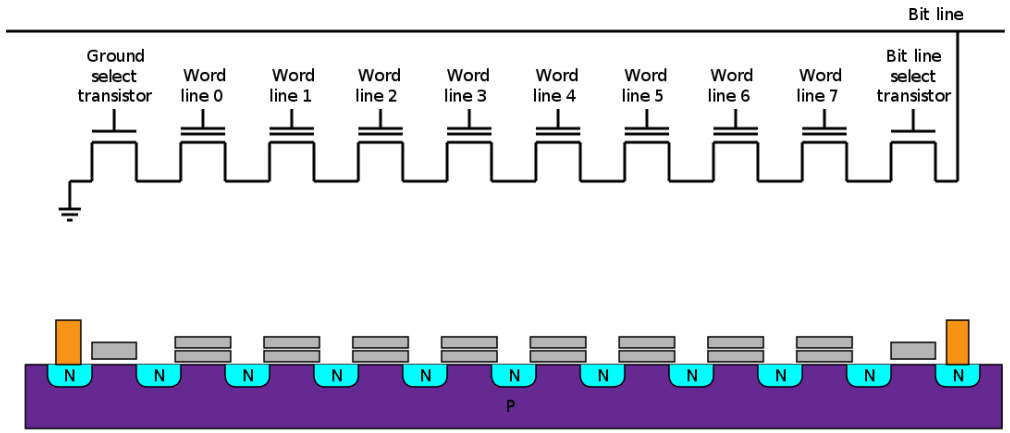
Basic Idea: Flash PIM: 2 Structures

北京大学 PEKING UNIVERSITY

NOR Flash

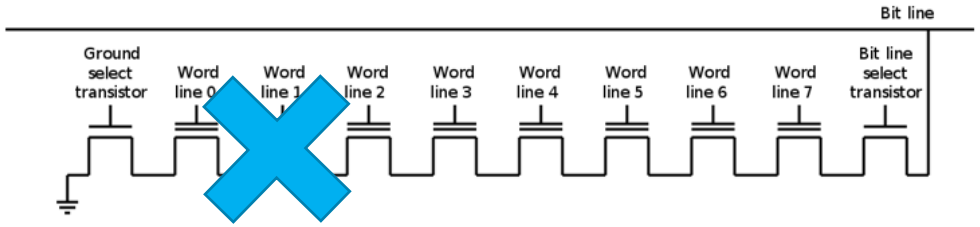


NAND Flash

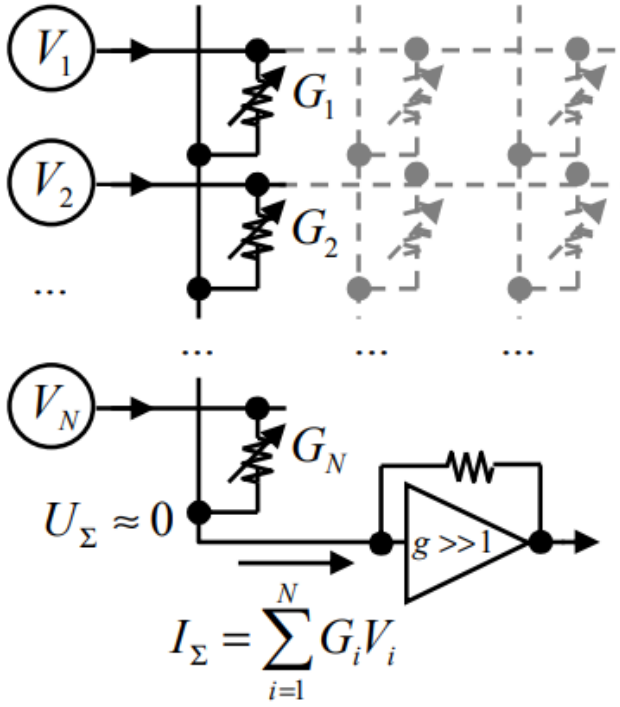


Q: Which one can be used for PIM-VMM?

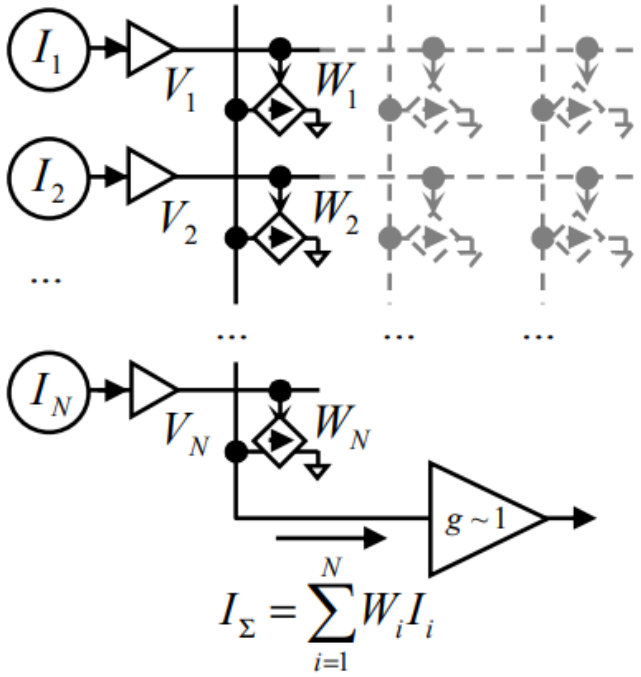
A: NOR Flash



Basic Idea: Flash PIM



Memristor PIM

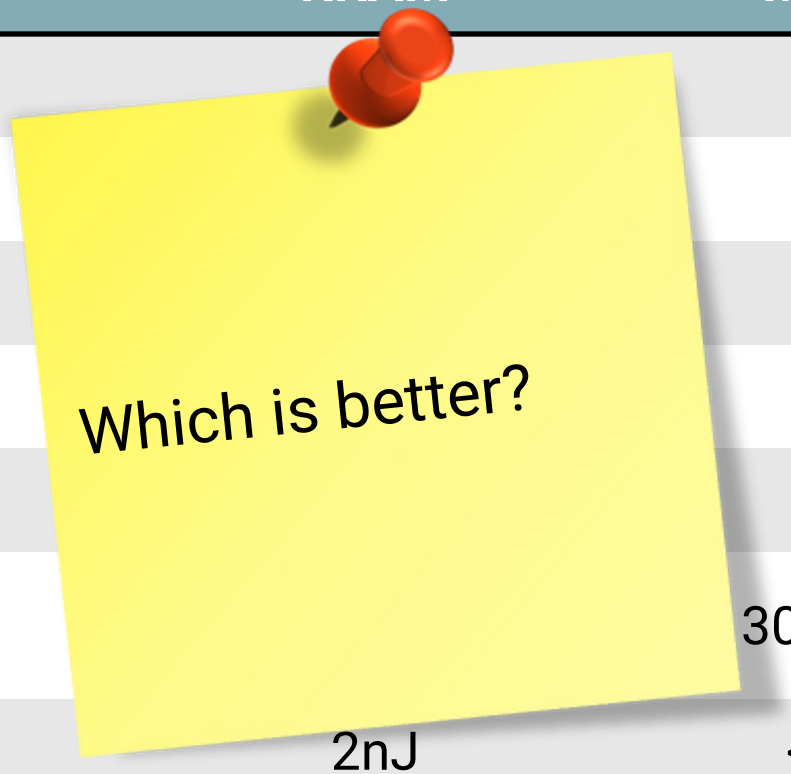


Flash PIM

Memory Technologies

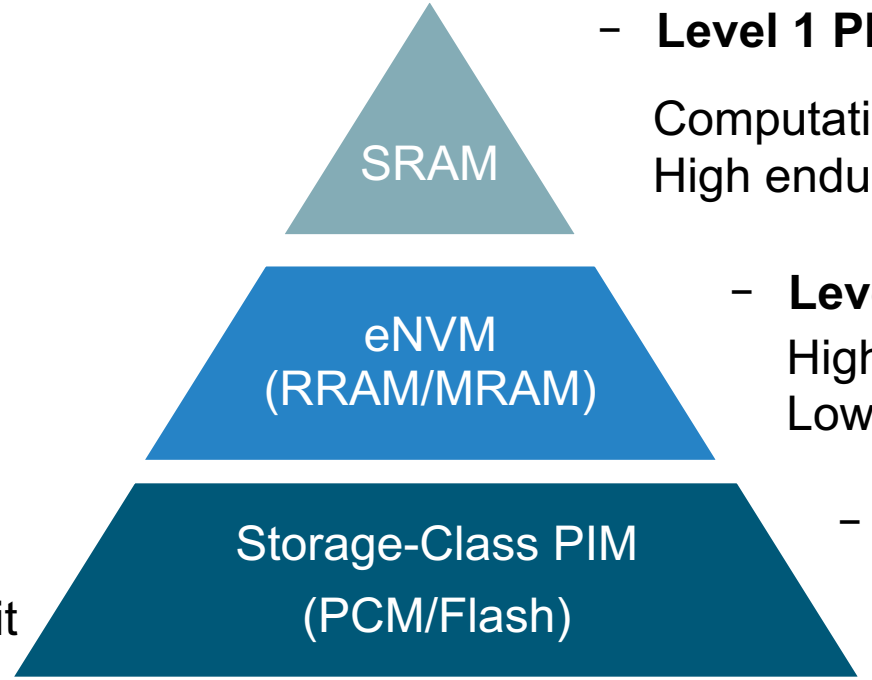
北京大学 PEKING UNIVERSITY

Property	PCM	RRAM	MRAM	CMOS-SRAM
Multilevel Cell	Yes		No	No
Storage Density	High		High	Low
R_{OFF}/R_{ON}	High		Low	High
Nonvolatility	Yes		Yes	No
Leakage	Low		Low	High
Cell Area	$16F^2$		$30\sim80F^2$	$160F^2$ (6T) $231F^2$ (8T)
Write Energy	6nJ	2nJ	<1nJ	<0.1nJ
Write Latency	150ns	100ns	10ns	<1ns
Endurance	10^7cycles	10^6cycles	10^{15}cycles	10^{16}cycles



Future PIM Memory Hierarchy

R/W Speed	*RW*	*Capacity*	*GEMM*
↑	<1ns	1Mb	~1ns@4~8bit
	<100ns	10Mb	~10ns@4~8bit
Density ↓	<10us	10Gb	~100ns@4~8bit

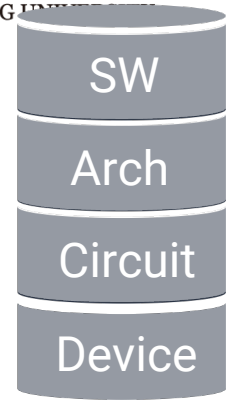


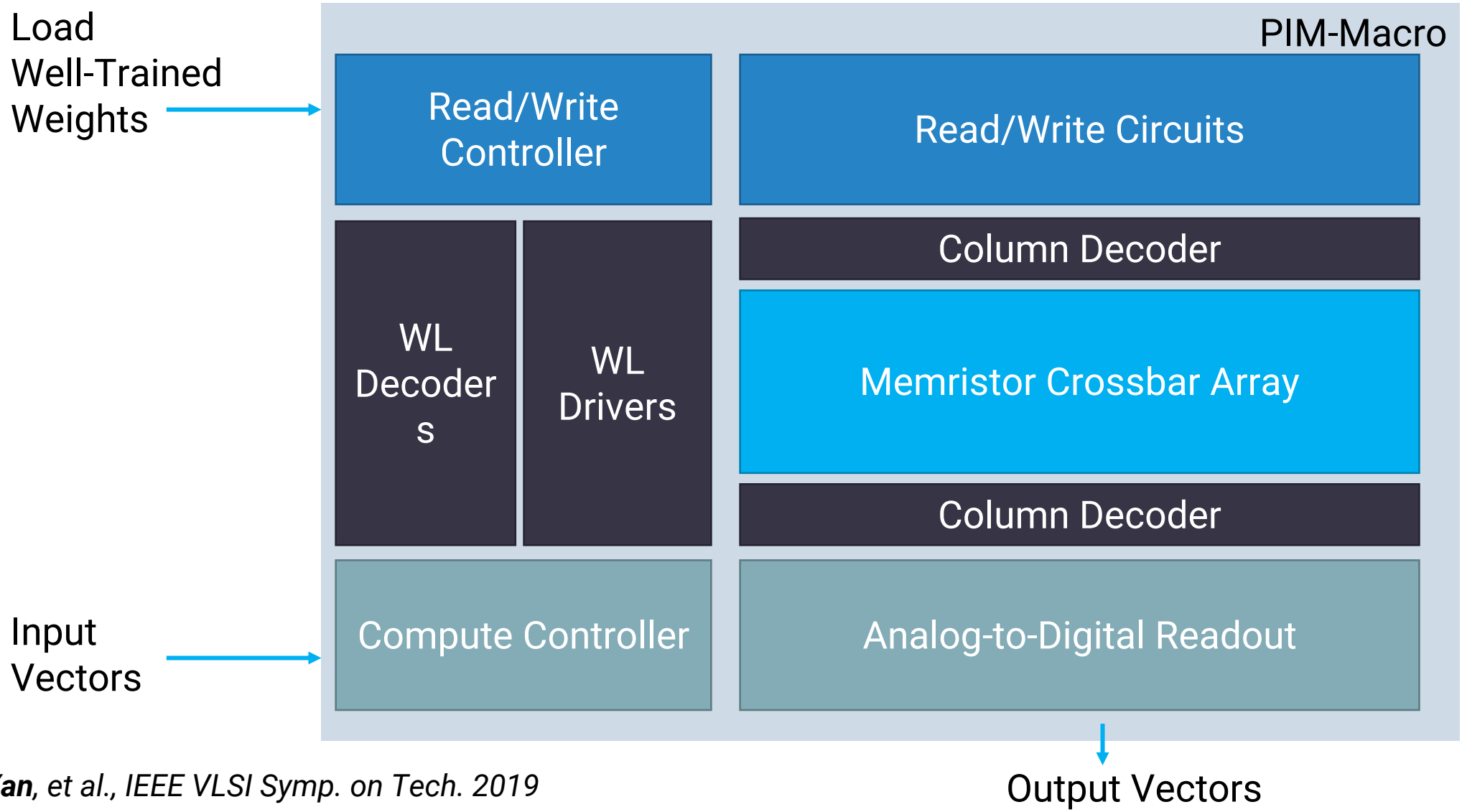
- **Level 1 PIM:**
  Computationally Intense
  High endurance
- **Level 2 PIM:**
  High density
  Low power
- **Level 3 PIM:**
  Large capacity
  OTP Weights

(numbers only indicates order of magnitudes)

Place PIM into System

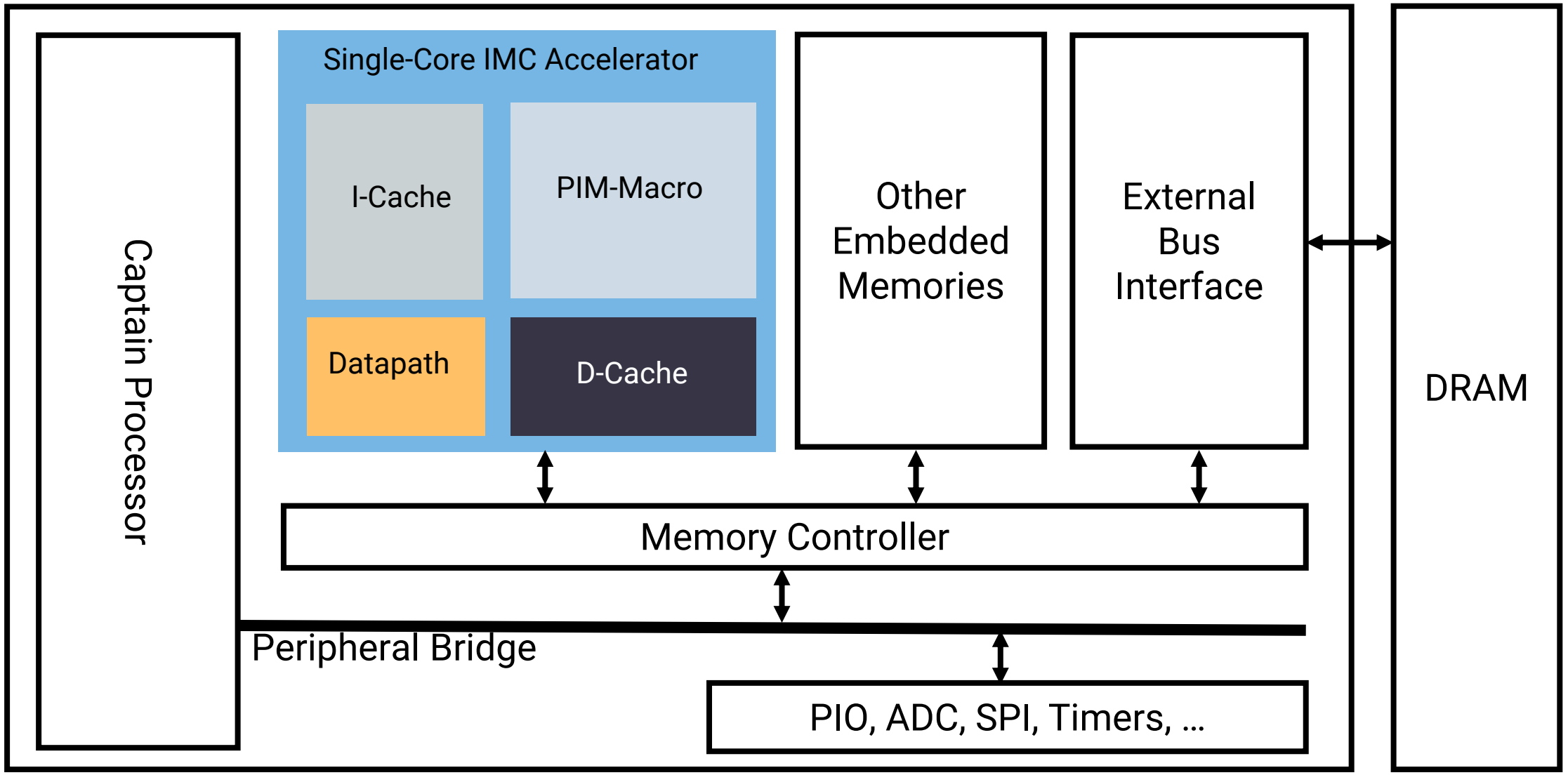
In-Memory Computing Solution Stack-Macro

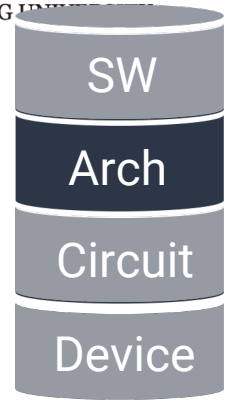




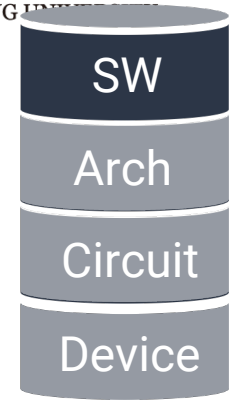
. ***Yan***, *et al., IEEE VLSI Symp. on Tech. 2019*

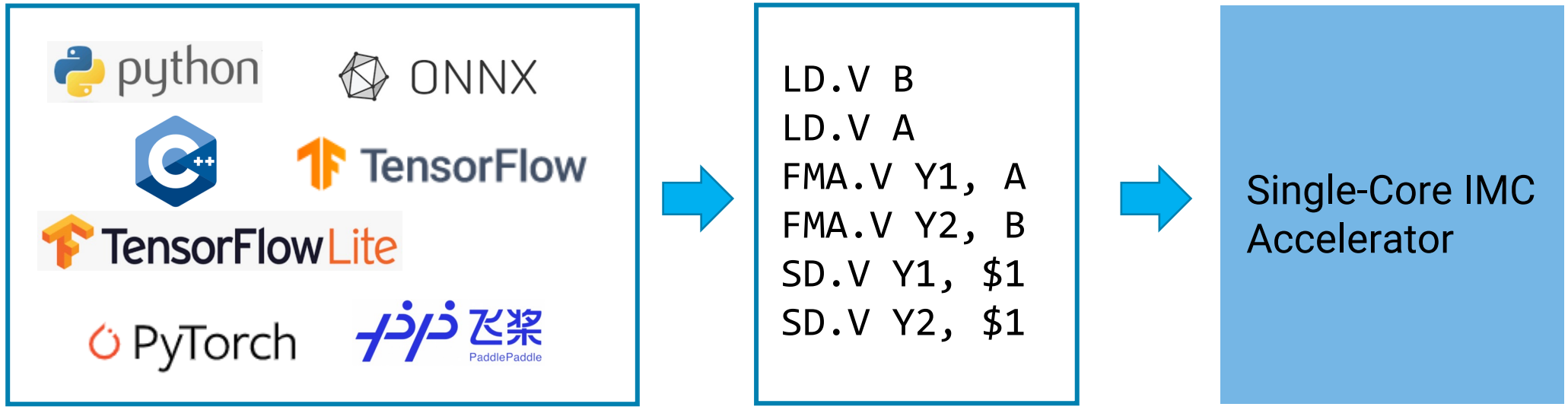
In-Memory Computing Solution Stack-SoC





In-Memory Computing Solution Stack-Software





Hardware-Software codesign is required for best use of ASICs:

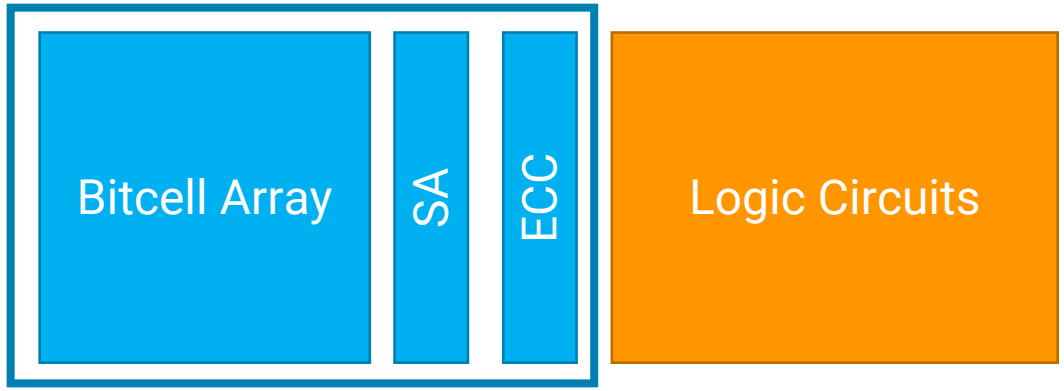
- Various ASICs need specialized compilers
- Various ML Models need specially optimized scheduling/dataflow

Not Only PIM

Compromise of PIM: Processing-Near Memory

Challenges:
A/D converter based compute readout circuits are

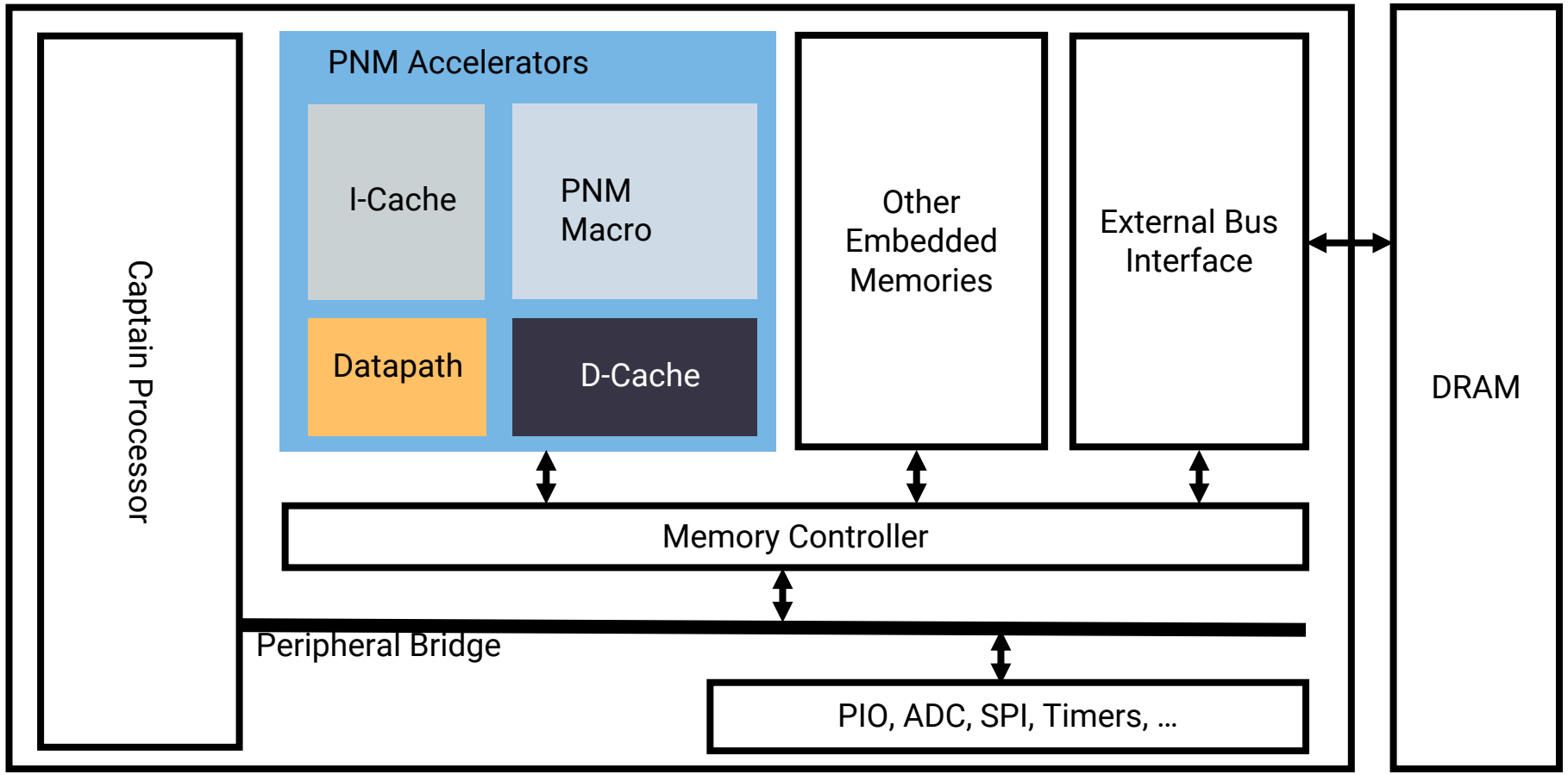
- sooooo hard to design
- sooooo large
- sooooo prune to noise/process variations



Normal Memory Macro

Processing-Near Memory Merging into Systems

- At what level does PNM shares behaves the same as PIM?

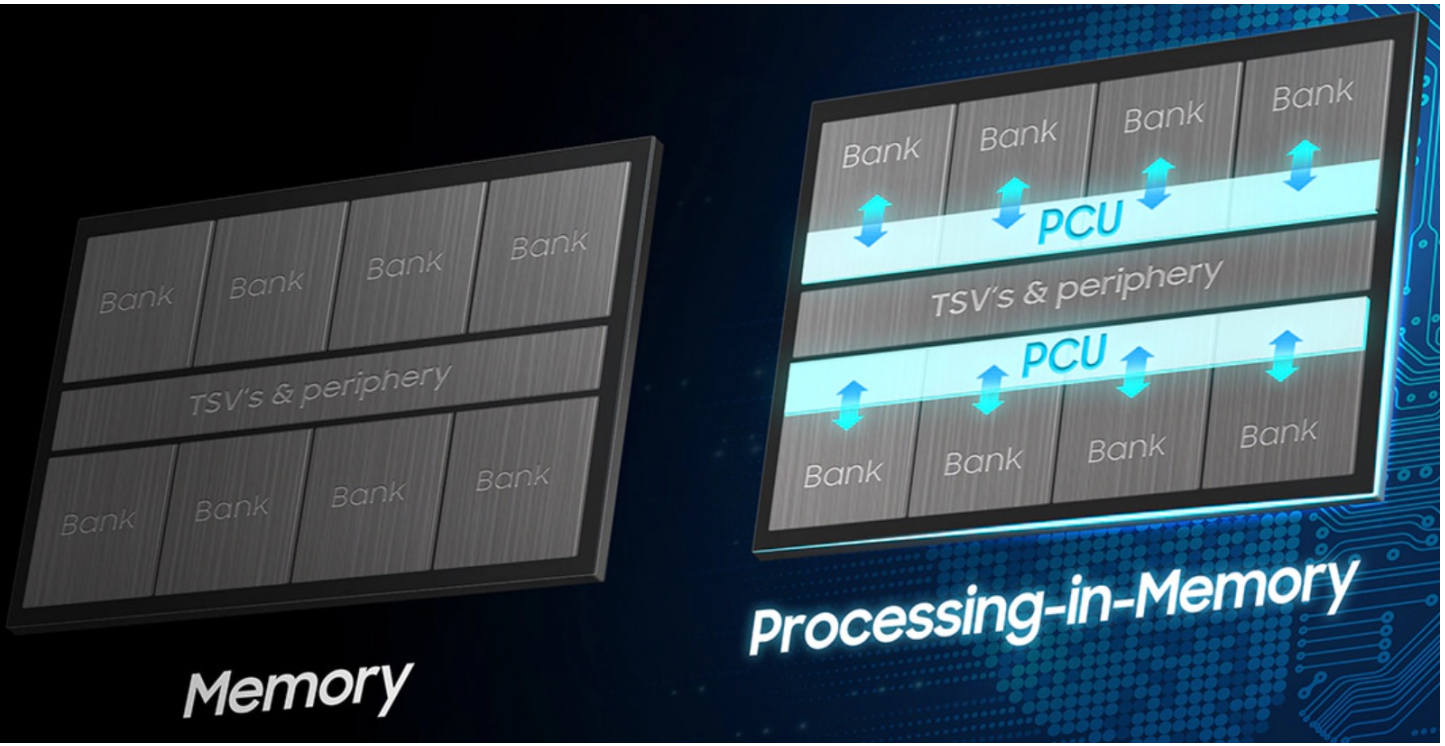


Different Level of Memory-Compute Fusion

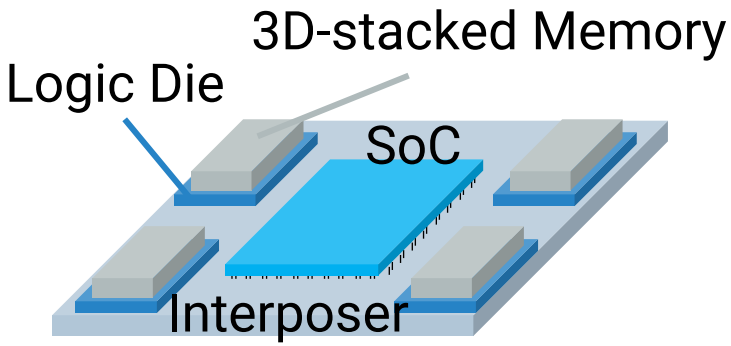
PEKING UNIVERSITY

- In-Memory Computing
  - Memristor PIM
  - SRAM PIM
  - Flash PIM
- Near-Memory Computing
  - HBM-PIM
  - Embedded MRAM+SA+Computation
- In-Storage Computing / Computational Storage
  - Solid-State Drive (SSD) + ARM CPU = In-Storage Computing Card

Examples: HBM-PIM … PNM



Near-Memory Computing



High Bandwidth Memory

For a system designer, the processing is actually happening in the “memory” chips

Examples: In-Storage Computing

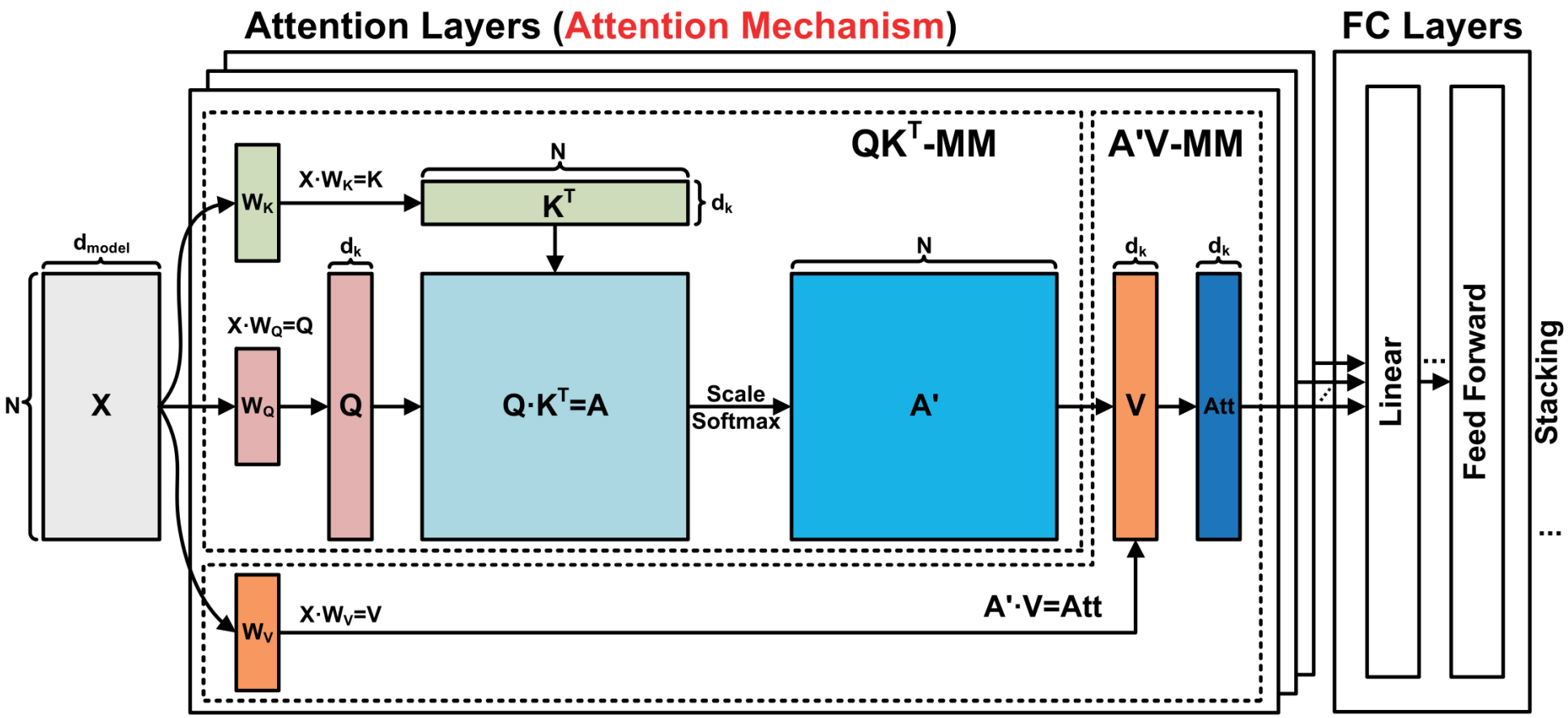
SmartSSD® CSD

- SSD + FPGA
- SSD + CPU
- Major purpose: storage

End

- PISLIB： PIM macro model

Accelerator Computing Task



Courtesy: TranCim Paper