



北京大学  
PEKING UNIVERSITY  
1898

# 智能硬件体系结构

## 第十四讲：存算一体简介

主讲：陶耀宇、李萌

2024年秋季

# 注意事项

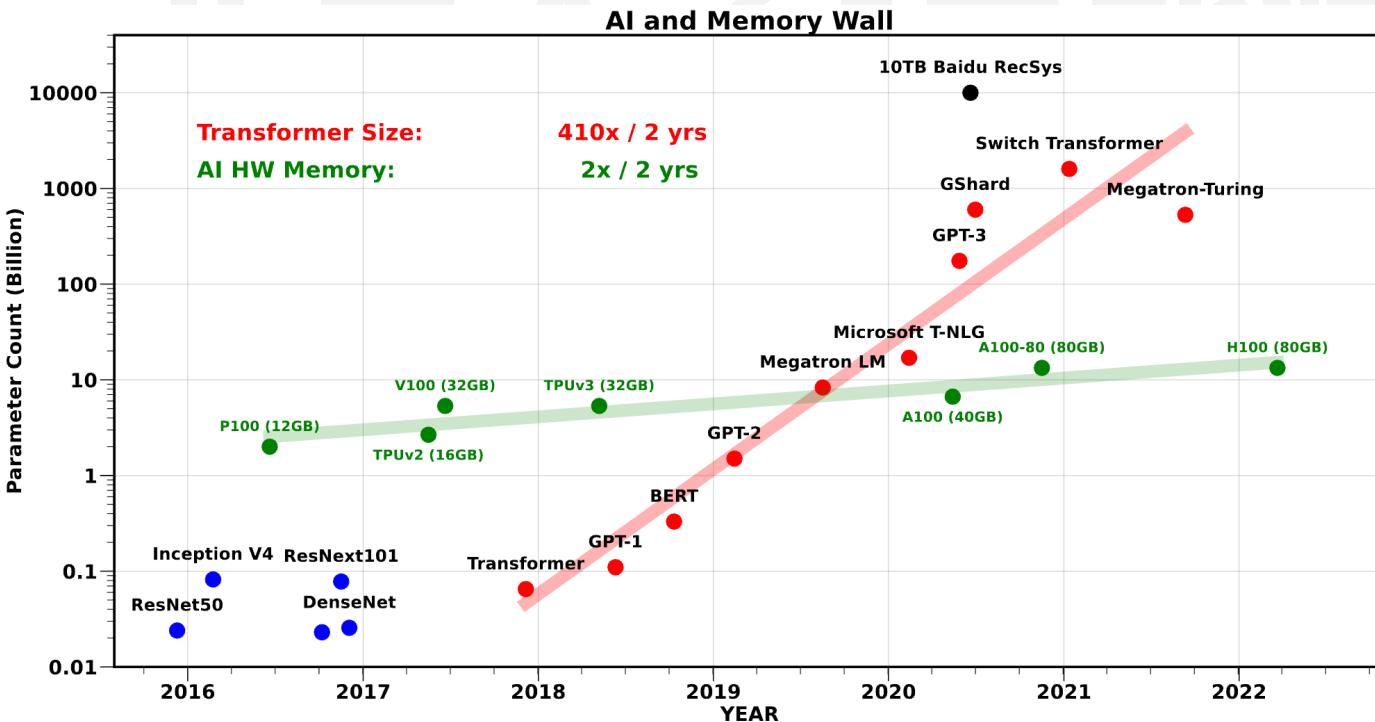
## • 课程作业情况

- 未来**1次作业 (12.15-12.30)**
  - AI芯片架构、软硬协协同优化
- **Paper Review (12月25号/27号1-3点，每个同学10分钟)**
  - 时间有疑义请尽快联系老师或助教提出更换
  - 请自行阅读近5年内的体系结构相关论文1-2篇
  - 做8页左右的PPT，**每个同学10分钟演讲 + 1~2分钟答疑**
- 第二次实验已经上传到课程网站，截止日期为**12月31号**

主讲：陶耀宇、李萌

# 人工智能模型的发展与挑战

- 随着人工智能的快速发展，神经网络模型规模指数级增长，对人工智能处理器的计算、存储和传输带宽需求也显著提升

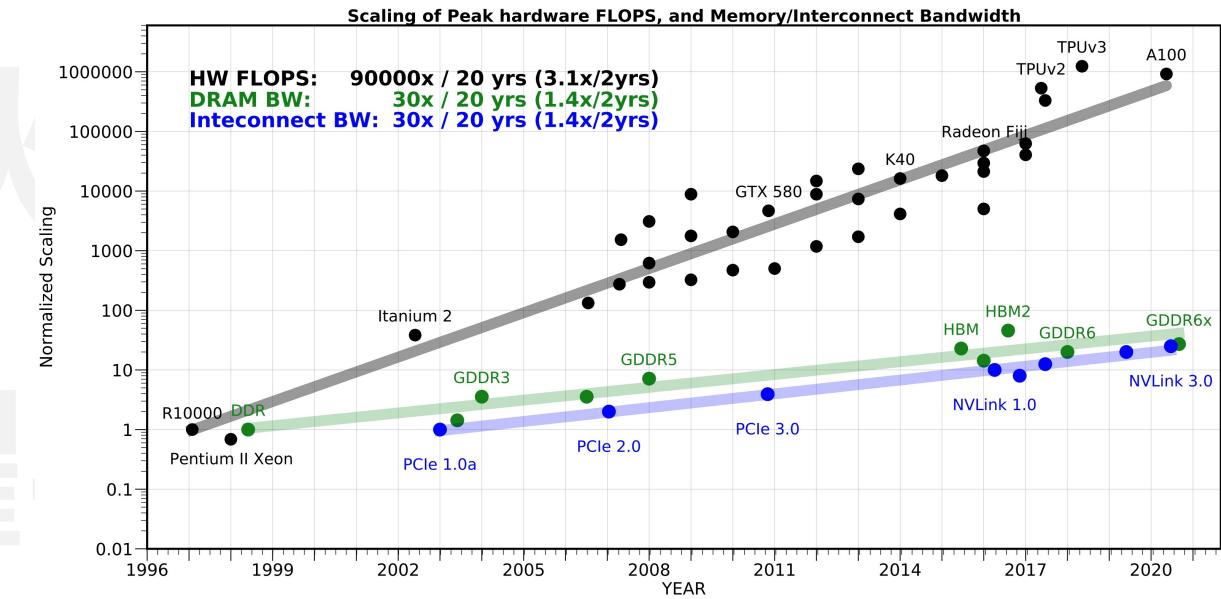
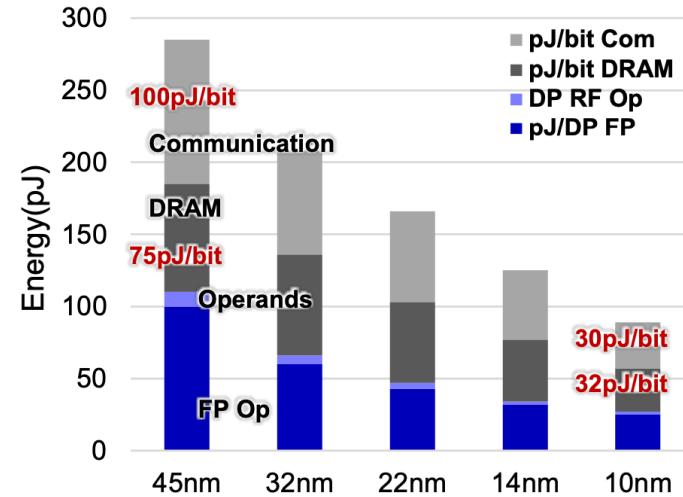
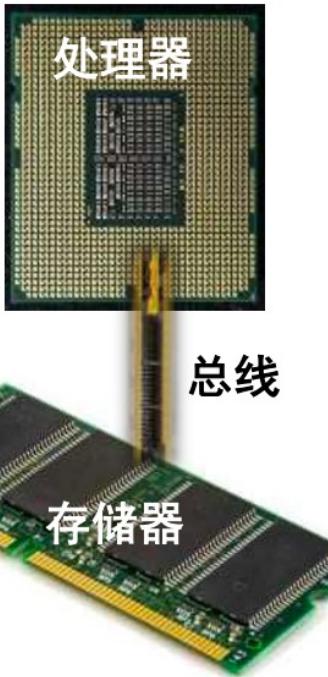


时间	模型	模型参数量 (GB)	模型算力需求 (TFLOPs-day)
2012	AlexNet	$10^{-2}$	$10^{-3}$
2015	ResNets	$10^{-1}$	$10^{-1}$
2017	Transformer	1	$10^1$
2020	GPT-3	$10^2$	$10^6$
2023	ChatGPT	$10^3$	$10^7$

[https://github.com/amirgholami/ai\\_and\\_memory\\_wall](https://github.com/amirgholami/ai_and_memory_wall)

# 人工智能模型的发展与挑战

- 然而，现有的冯诺依曼架构采用存储和计算分离的架构，面临“**存储墙**”和“**功耗墙**”瓶颈
  - 内存带宽/互连带宽增长与算力增长速度不一致（ $30\times$  vs  $90,000\times$ ），特别是对于先进节点
- 如何克服层“**存储墙**”瓶颈，进一步提升AI处理器算力和能效成为重要问题



[https://github.com/amirgholami/ai\\_and\\_memory\\_wall](https://github.com/amirgholami/ai_and_memory_wall)

- 回顾：什么样的算子更容易受到“存储墙”影响？

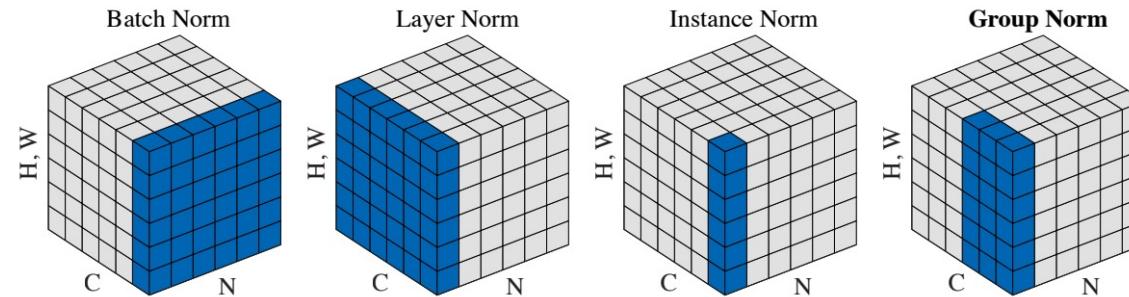
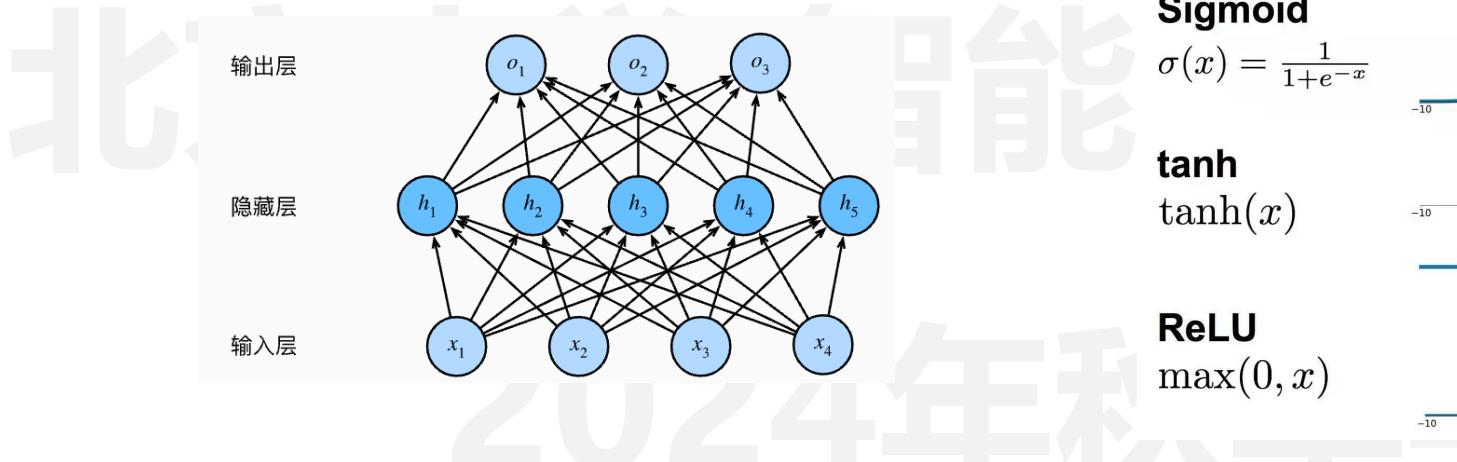
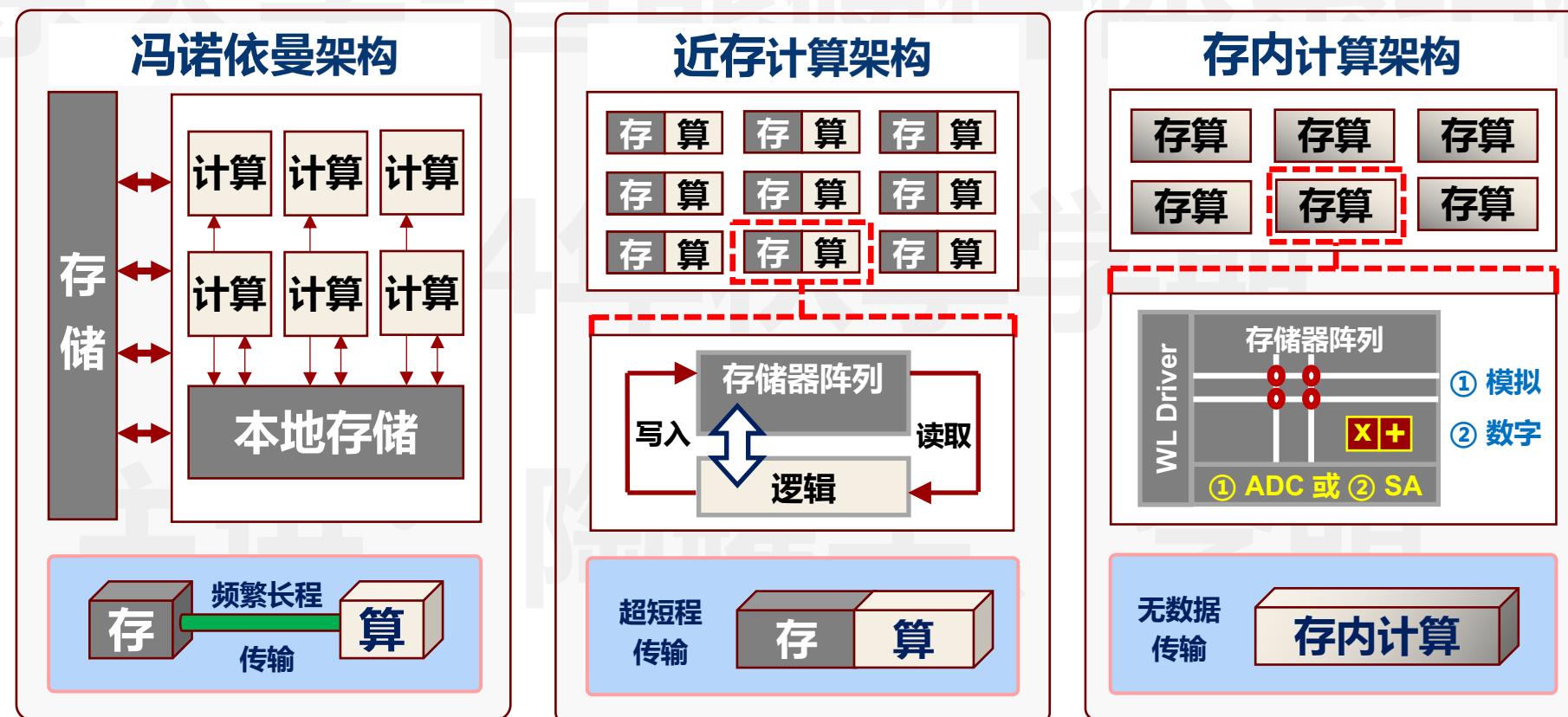


Figure 2. **Normalization methods.** Each subplot shows a feature map tensor, with  $N$  as the batch axis,  $C$  as the channel axis, and  $(H, W)$  as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

# 存算一体新架构

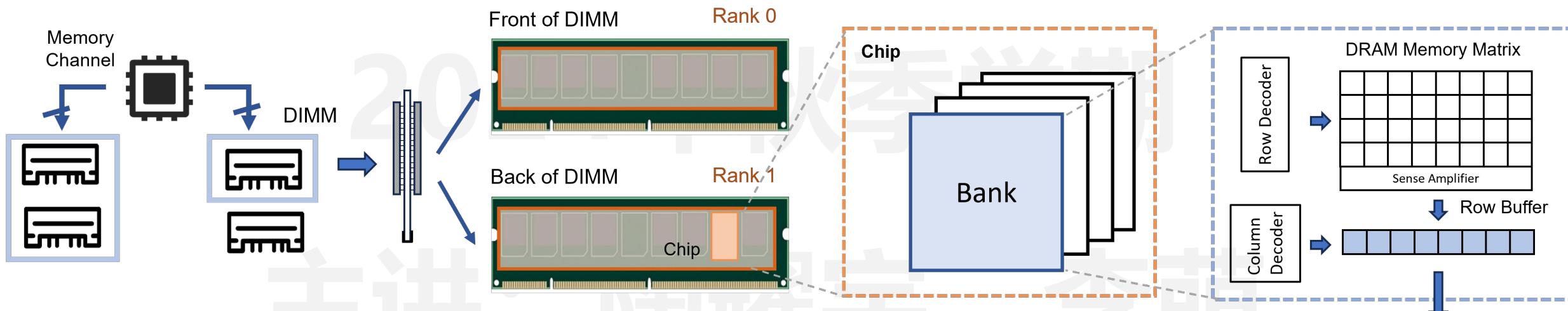
- 针对于访存瓶颈的算子，**存算一体技术**成为打破算力瓶颈的重要途径
- **近存计算 (NDP/PNM)**：存储阵列一般无需改动，只提供数据存储功能，计算模块安放在阵列附近
- **存内计算 (PIM)**：存储器件可以参与计算操作，这通常意味着存储阵列需要改动来支持计算





# DRAM内存架构简介

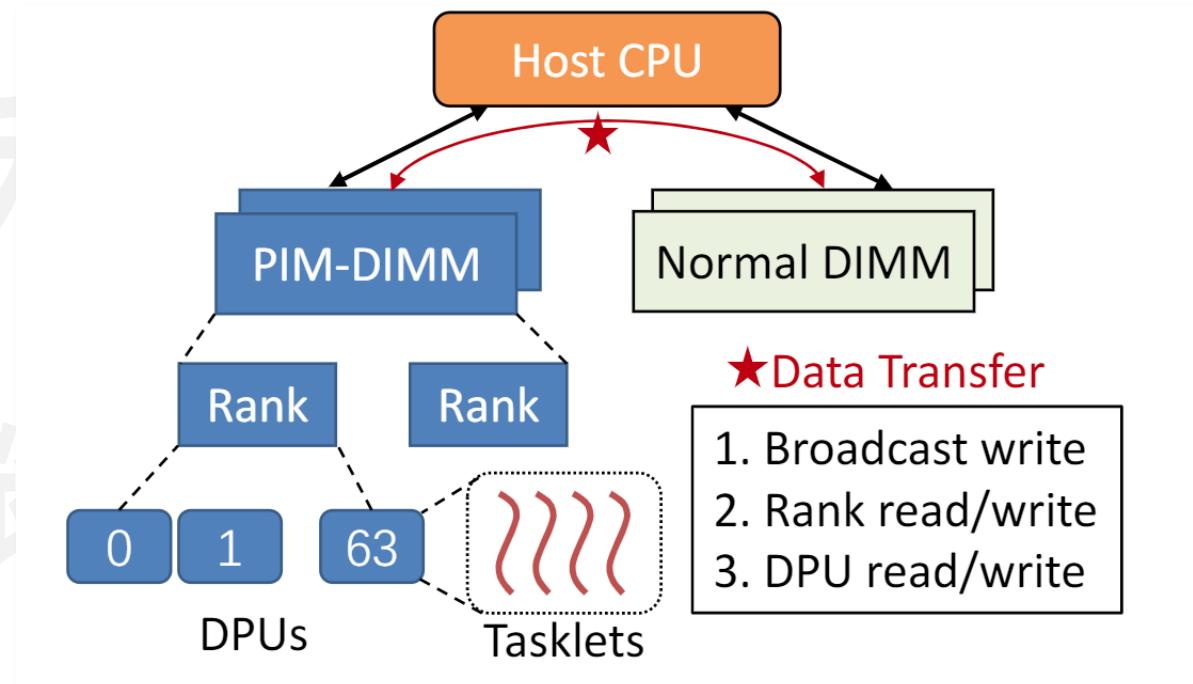
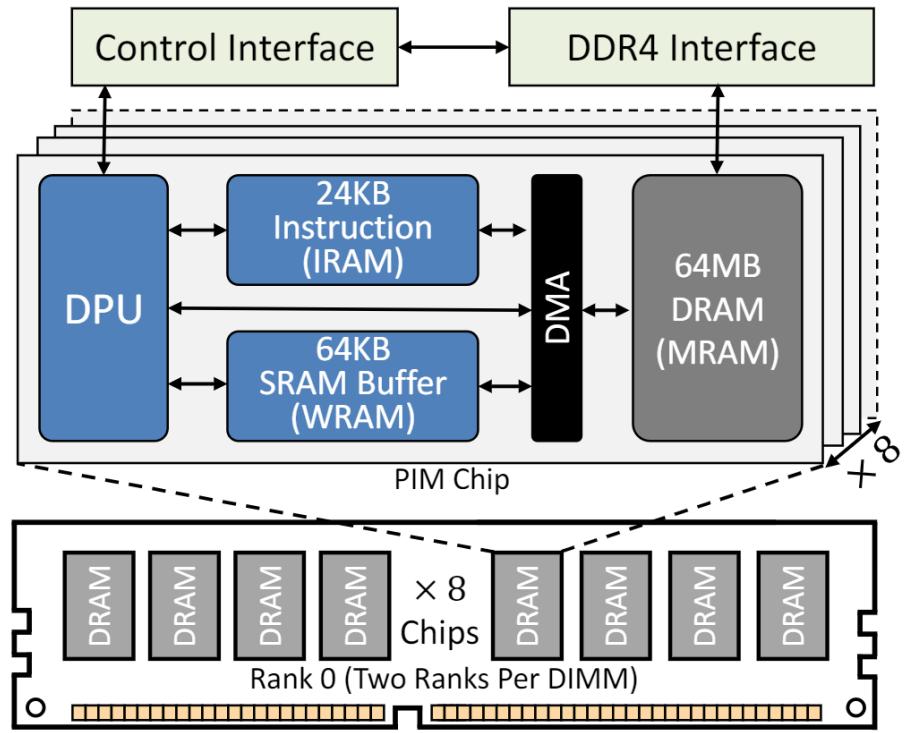
- DRAM的内存架构和组织形式
  - DRAM channel, DIMM, rank, chip, bank, column/row decoder
- 内部带宽 vs 外部带宽



# UPMEM PIM-DIMM

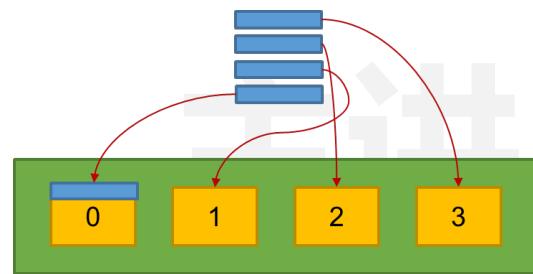
- UPMEM PIM-DIMM架构

- 单条8GB, 128个计算核心 (DPU) , 64KB便签存储器
- 内部带宽约为**100 GB/s** (DDR4带宽<20 GB/s)

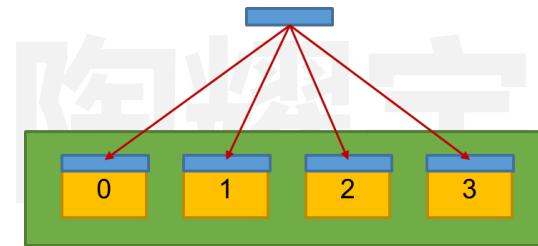


# UPMEM PIM-DIMM

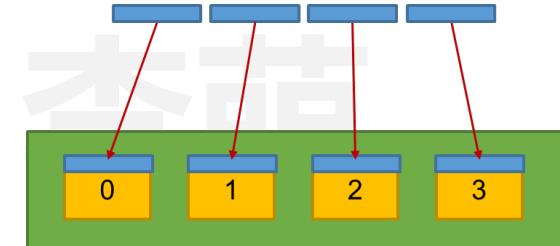
- UPMEM PIM-DIMM系统集成
  - 为PIM-DIMM分配了独立的物理地址空间
  - 数据传入包括serial、parallel和广播模式
- 资源分配/调度：
  - Tasklet (线程)：一个DPU 可以运行24个线程，有各自的stack
  - DPU：64MB Main RAM，DPU之间无法直接进行数据传递
  - Rank：64个DPU，可以进行并行CPU-DPU数据传输



CPU-DPU serial



CPU-DPU parallel



CPU-DPU broadcast

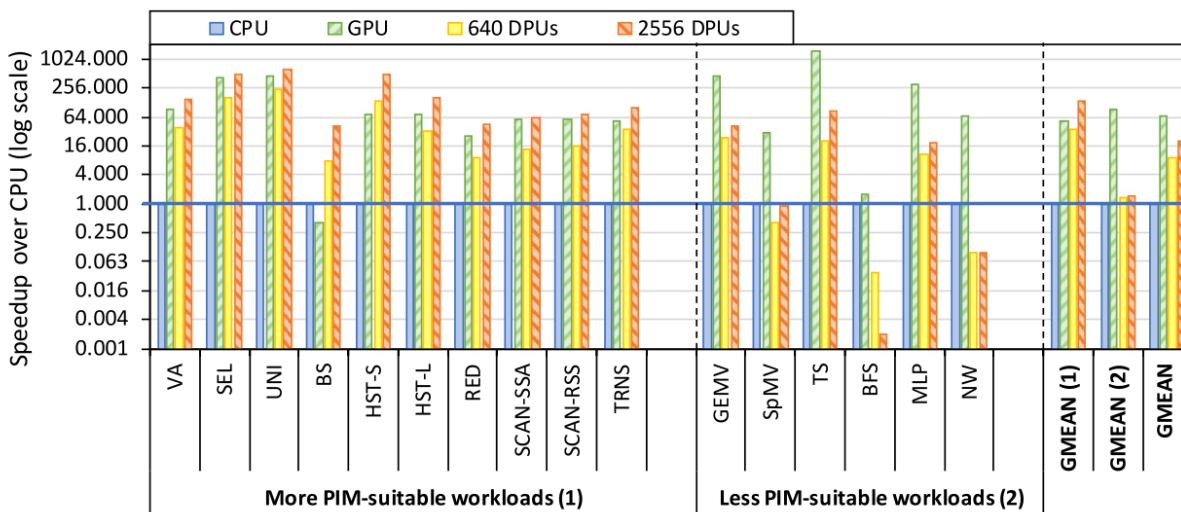
## • 性能测试

Table 4. Evaluated CPU, GPU, and UPMEM-based PIM Systems.

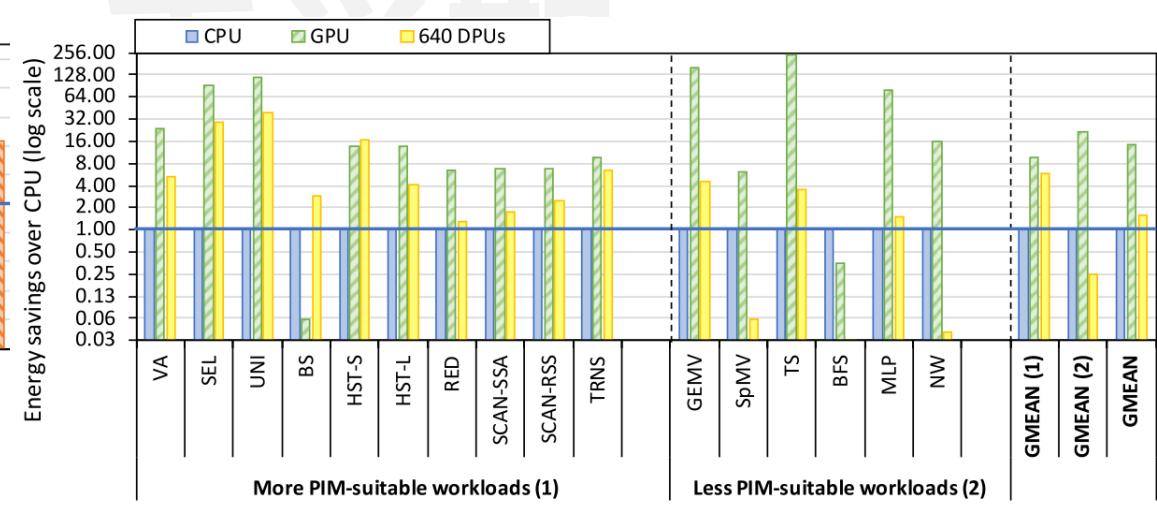
System	Process Node	Processor Cores			Memory		TDP
		Total Cores	Frequency	Peak Performance	Capacity	Total Bandwidth	
Intel Xeon E3-1225 v6 CPU [106]	14 nm	4 (8 threads)	3.3 GHz	26.4 GFLOPS*	32 GB	37.5 GB/s	73 W
NVIDIA Titan V GPU [192]	14 nm	80 (5,120 SIMD lanes)	1.2 GHz	12,288.0 GFLOPS	12 GB	652.8 GB/s	250 W
2,556-DPU PIM System	2x nm	2,556 <sup>9</sup>	350 MHz	894.6 GOPS	159.75 GB	1.7 TB/s	383 W <sup>†</sup>
640-DPU PIM System	2x nm	640	267 MHz	170.9 GOPS	40 GB	333.75 GB/s	96 W <sup>†</sup>

\*Estimated GFLOPS = 3.3 GHz × 4 cores × 2 instructions per cycle.

<sup>†</sup>Estimated TDP =  $\frac{\text{Total DPUs}}{\text{DPUs/chip}} \times 1.2 \text{ W/chip}$  [52].



性能对比



功耗对比

- 优势

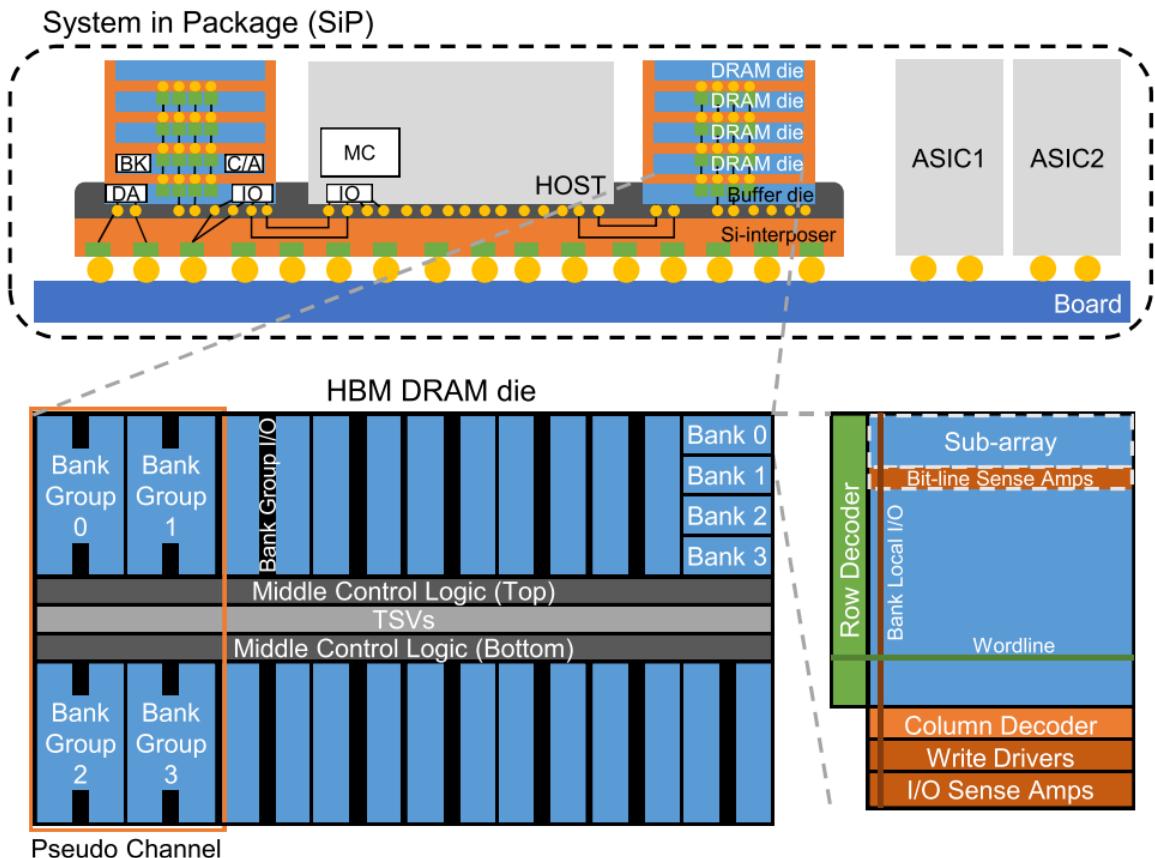
- Scalable memory bandwidth, 相对于HBM等容量更大 (同代对比)
- 通用编程模型, 且兼容目前的server系统

- 缺点

- 无法当作普通main-memory 使用: CPU 和Memory 都有内存控制器, 协调困难, 存在 **Memory interleaving**、**Cache coherence** 问题和系统支持问题 (虚拟内存等)
- 数据传输性能是瓶颈, 难以处理需要频繁同步的应用
- 应用切分困难: 总共包含 2048个DPU, 64MB local DRAM/DPU, DPU之间通信代价高
- 无SIMD / FP 单元, 浮点算力太低, 在深度学习应用场景受限

# Samsung's PIM-HBM

- HBM的带宽对于访存密集型深度学习算子仍然存在不足

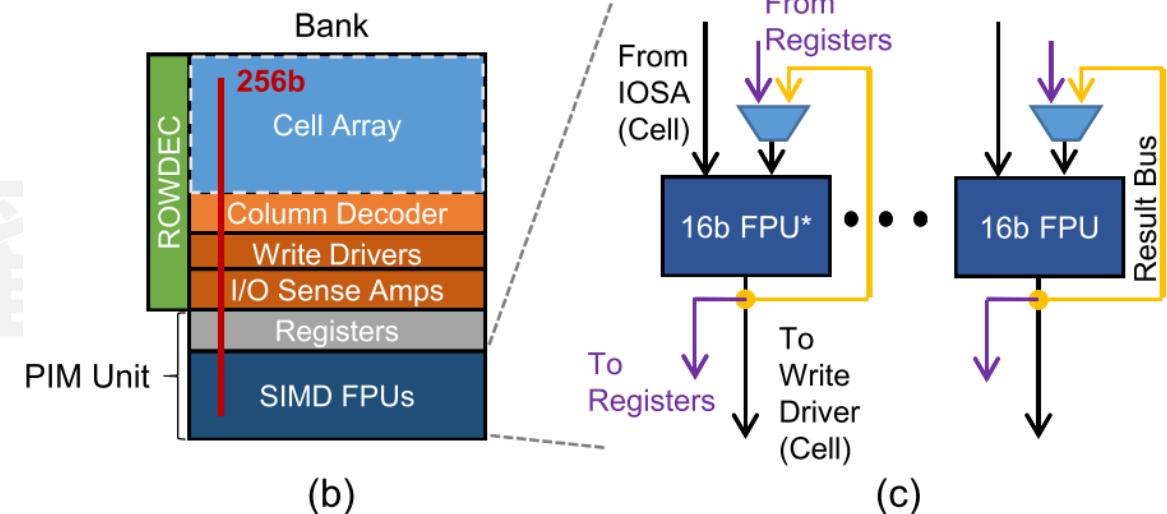
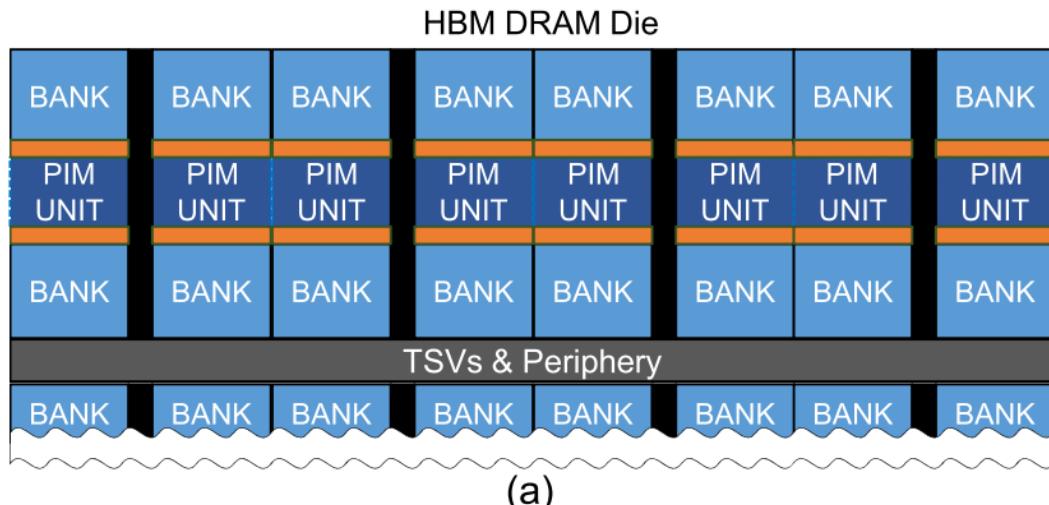


硬件体系结构

- **HBM2 架构:**
  - 4 pseudo channels (pCH) per Die
  - 4 bank group per pCH
  - 4 banks per bankgroup
  - 256-bit data block over 4 64-bit bursts over one pCH
  - 256GB/s (1GHz IO)

# Samsung's PIM-HBM

- **设计目标：**支持PIM和普通HBM模式，维持原本的DRAM bank和阵列设计
- 在Bank的I/O boundary处设置PIM单元
  - **Bank row数量减半**，两个bank共享PIM单元，每个PIM单元内包含 $16 \times 16$ bit的SIMD单元
- PIM单元由标准的DRAM column commands (RD、WD) 控制
  - 和UPMEM相似，指令提前load进memory，用RD指令操作数地址，一个channel所有bank可以并行读出4096个bit，RD指令同时激活PIM单元的计算



# Samsung's PIM-HBM

- PIM-HBM 三种操作模式

- Single Bank Mode

- 标准DRAM 模式

- 每次访问一个channel中的一个Bank

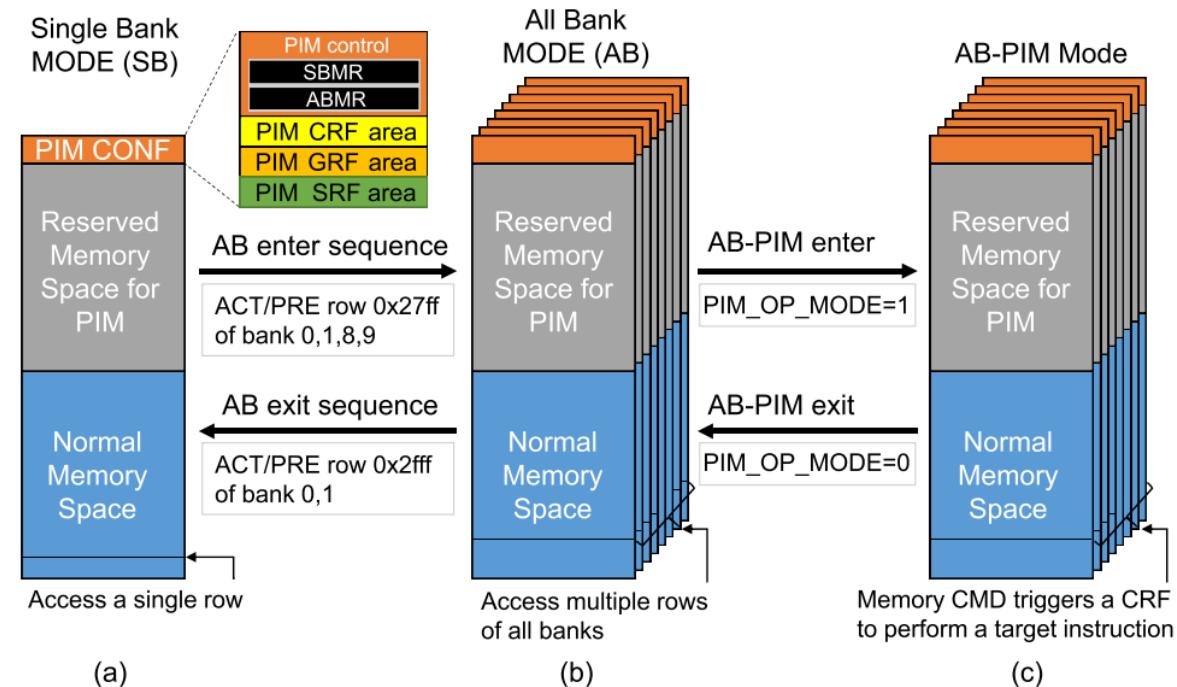
- All Bank Mode

- 一个DRAM 指令可以同时控制多个Bank

- 8x higher on-chip bandwidth

- All Bank PIM Mode

- 通过写high row地址以及PIM mode 寄存器来切换mode

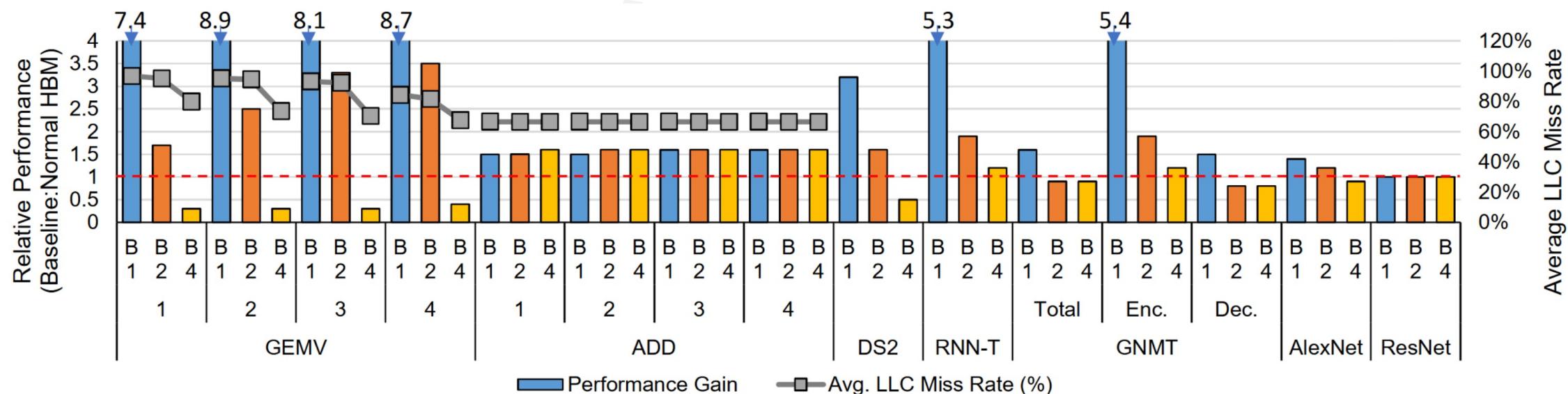
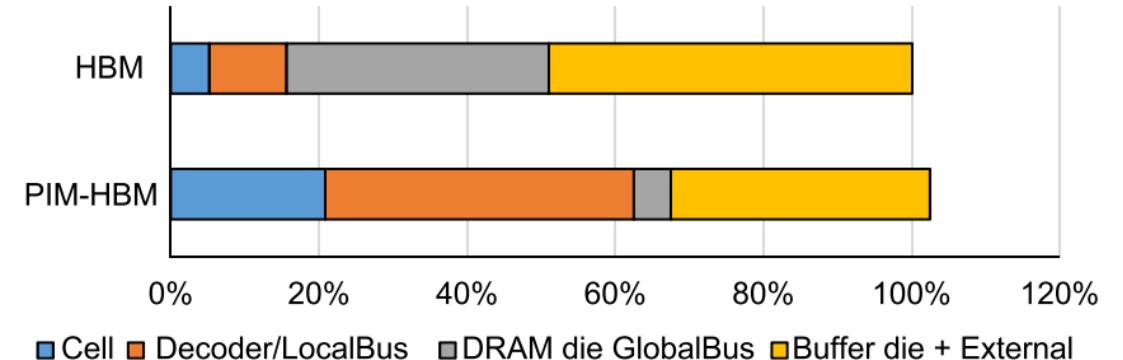


# Samsung's PIM-HBM

- 访存密集型应用上加速比明显

TABLE VI: Microbenchmark.

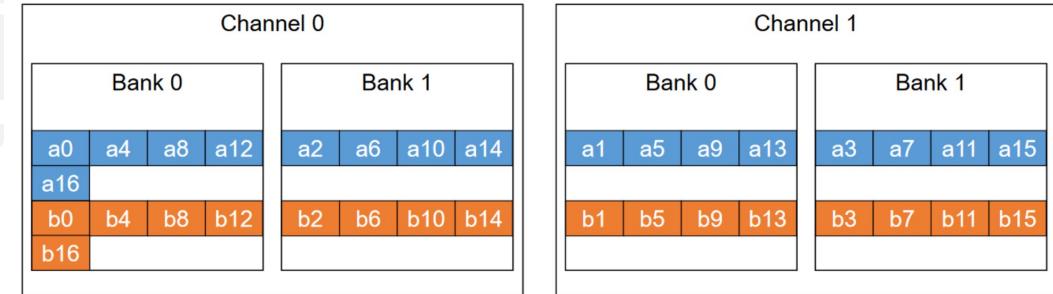
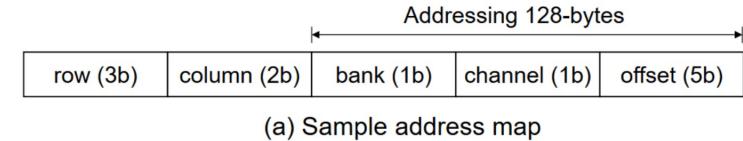
Name	GEMV Dim.	Name	ADD Dim.
GEMV1	$1k \times 4k$	ADD1	2M
GEMV2	$2k \times 4k$	ADD2	4M
GEMV3	$4k \times 8k$	ADD3	8M
GEMV4	$8k \times 8k$	ADD4	16M



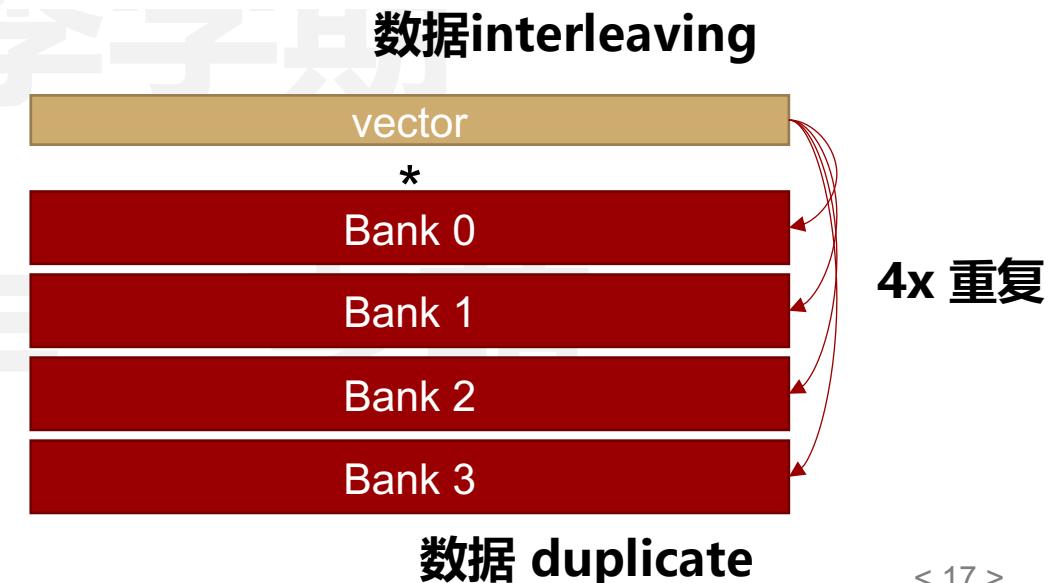
# Samsung's PIM-HBM

- **Data Interleaving** 问题:

- 对于elementwise 应用, interleaving 不影响计算结果
- 对于GEMV类应用, 权重矩阵需要软件层进行重排
- 控制粒度问题:
  - PIM计算和Host计算无法细粒度pipeline
  - 多cube后bank数太多, 小规模算子利用率低
- 数据Replica 问题
  - 以GEMV 为例, Host 需要向每个bank 写入输入
- PIM间没有直接的数据通路, 需要Host进行转发

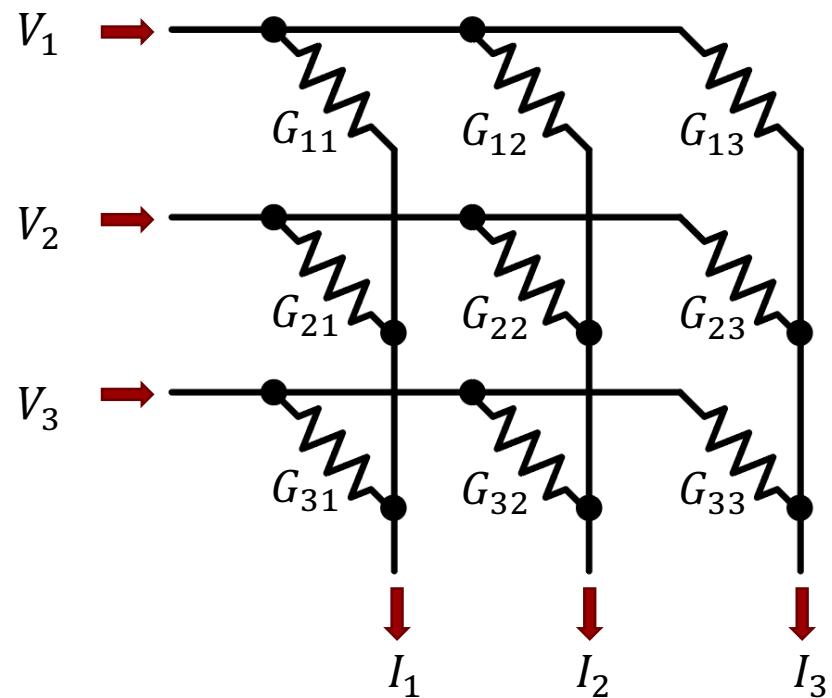


(b) Data placement of vector a and b (allocated to 128-bytes aligned address)



# 存内计算

- 存内计算可以分为**模拟**存内计算和**数字**存内计算
- 模拟存内计算利用基尔霍夫电流定律，可以高效实现矩阵向量乘法



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

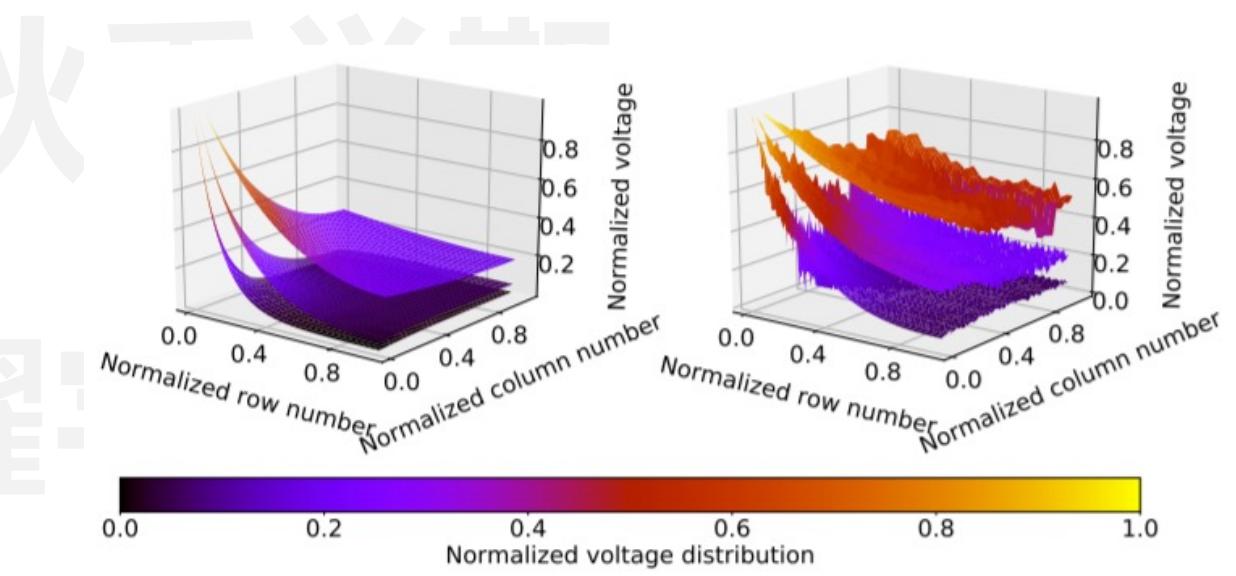
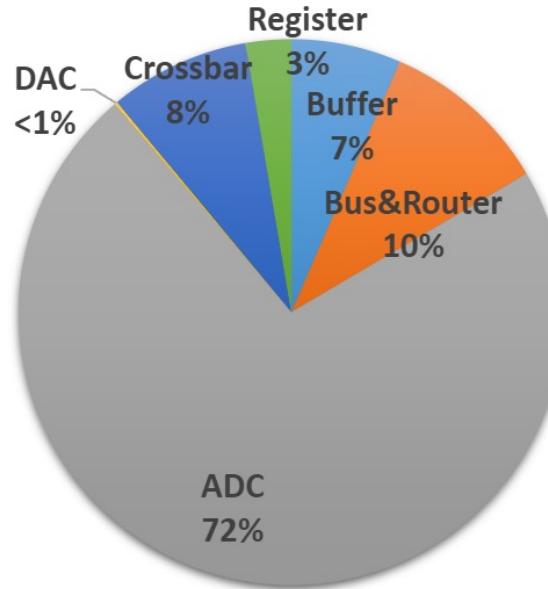
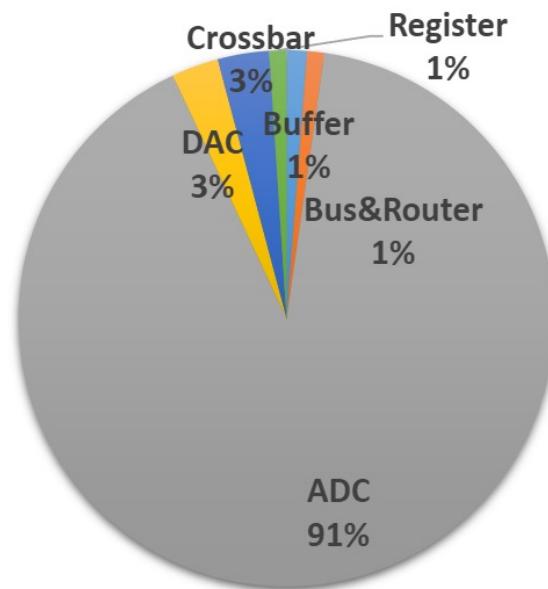
inputs                    weights                    outputs

- 存内计算介质可以采用SRAM、DRAM、Flash或者其他新型存储器

# 模拟存内计算面临的问题

- 模拟存内计算主要面临以下挑战：

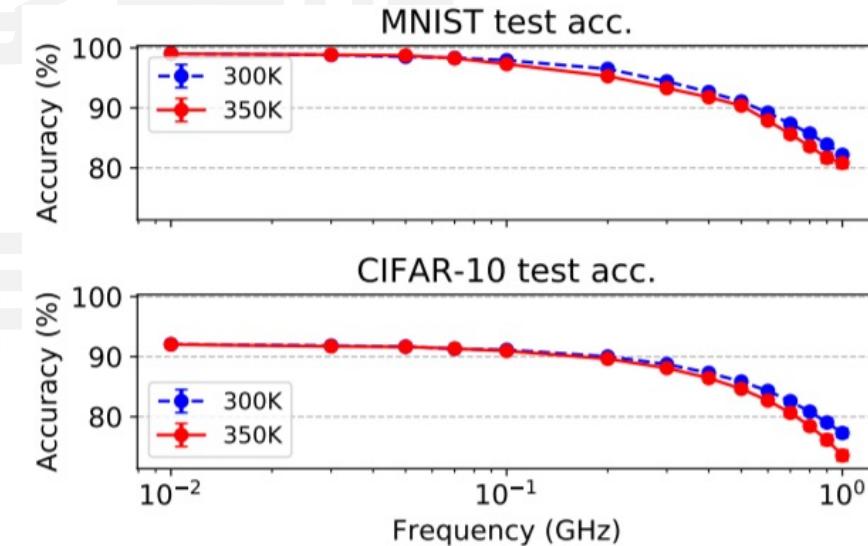
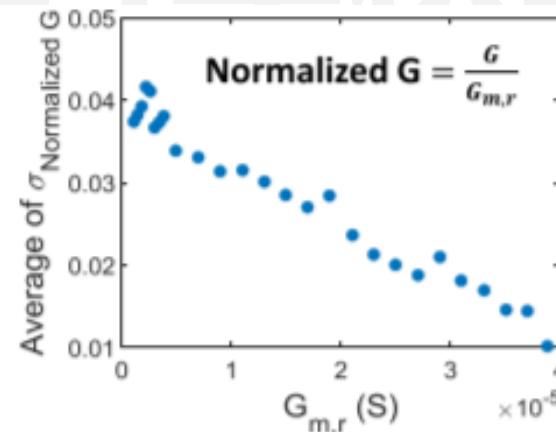
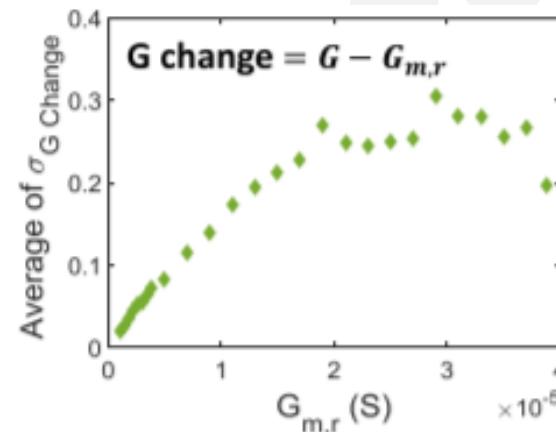
- ADC/DAC的开销很大
- 众多**非理想效应**，例如IR drop、stuck-at faults、高阻态和低阻态等



# 模拟存内计算面临的问题

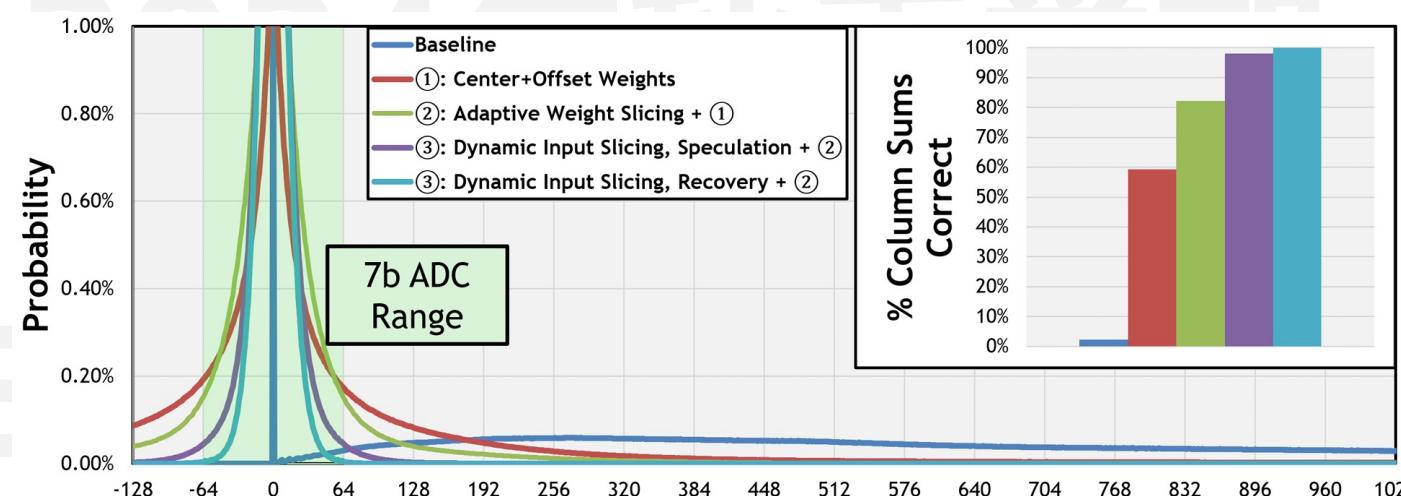
- 模拟存内计算主要面临以下挑战：

- ADC/DAC的开销很大
- 众多非理想效应，例如IR drop、stuck-at faults等
- 数据保持错误、编程错误等
- 本征噪声，例如热噪声、随机电报噪声等 → 如何避免噪声影响成为了重要的研究方向



# 代表性架构RAELLA

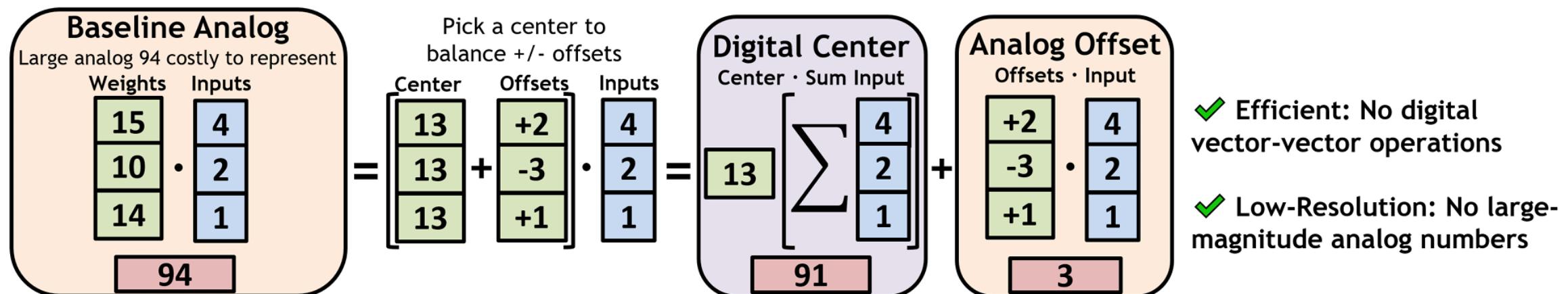
- MIT Vivienne Sze组发表于ISCA 2023
- 基于RRAM的存内计算架构，文章核心在于**不进行重训练**的前提下，降低ADC精度
  - 原有方法需要调整模型参数或者直接使用低精度ADC，往往需要重训练或造成计算误差
- 提出一系列方法，降低输出结果的范围，包括 (1)center + offset weights, (2) adaptive weight slicing, (3) dynamic input slicing



# 代表性架构RAELLA

- Center + offset weights
- 通过移动0点，均衡模型权重中的正负值，并通过支持有符号数计算，降低累加结果范围

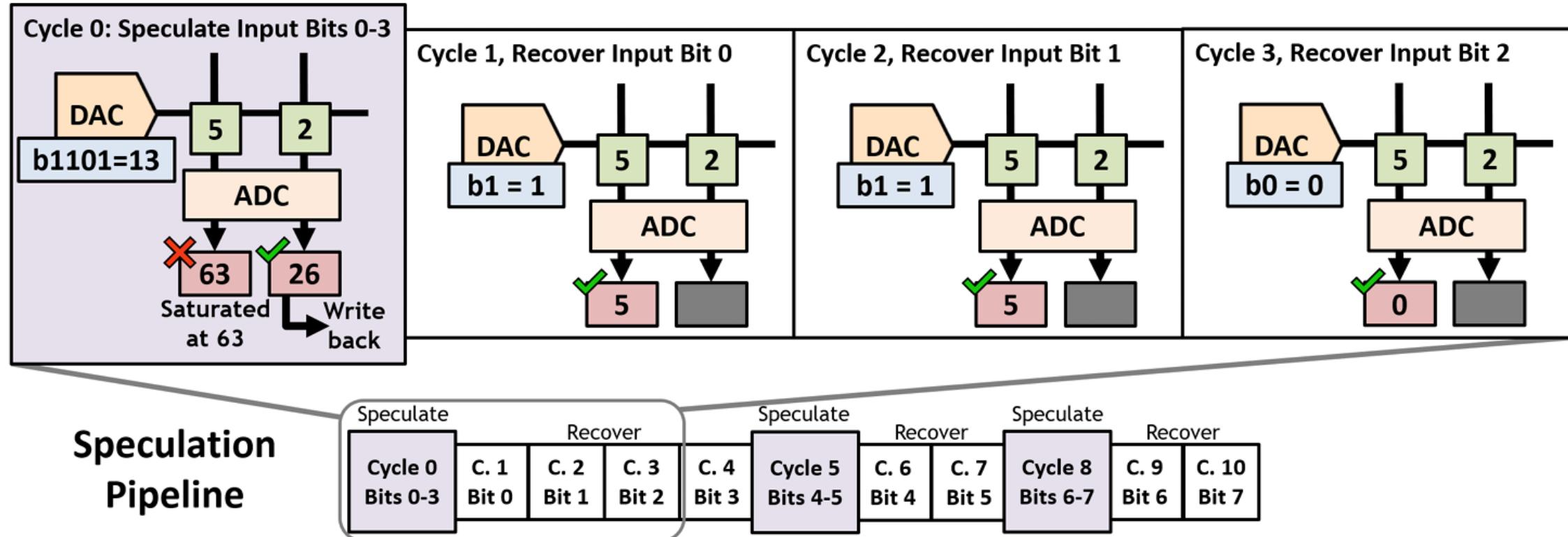
北京大学-智能硬件体系结构



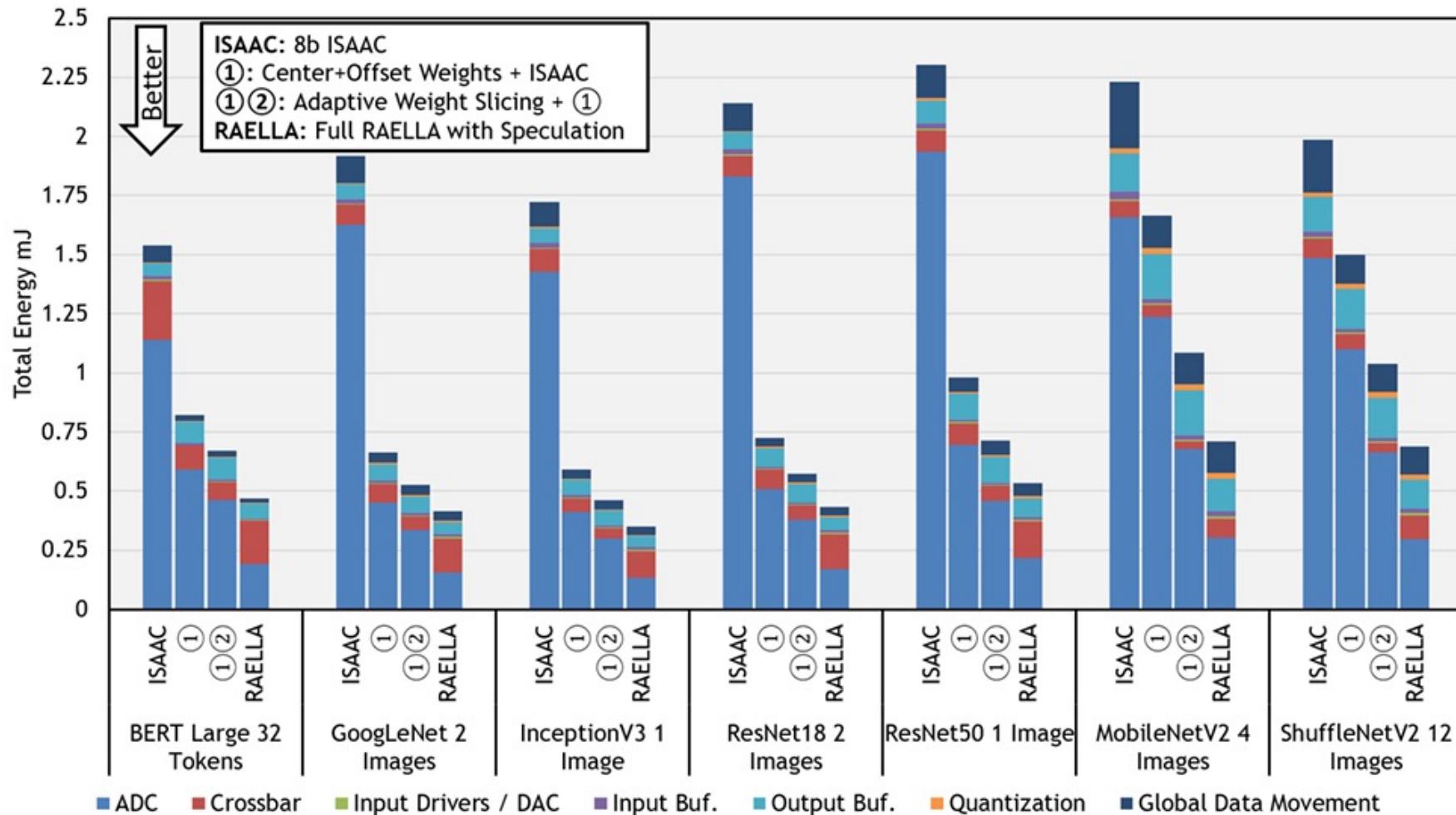
主讲：陶耀宇、李萌

# 代表性架构RAELLA

- Adaptive weight slicing + dynamic input slicing
- 通过控制权重和输入的slicing比特位数，平衡计算次数与单次计算的累加结果范围



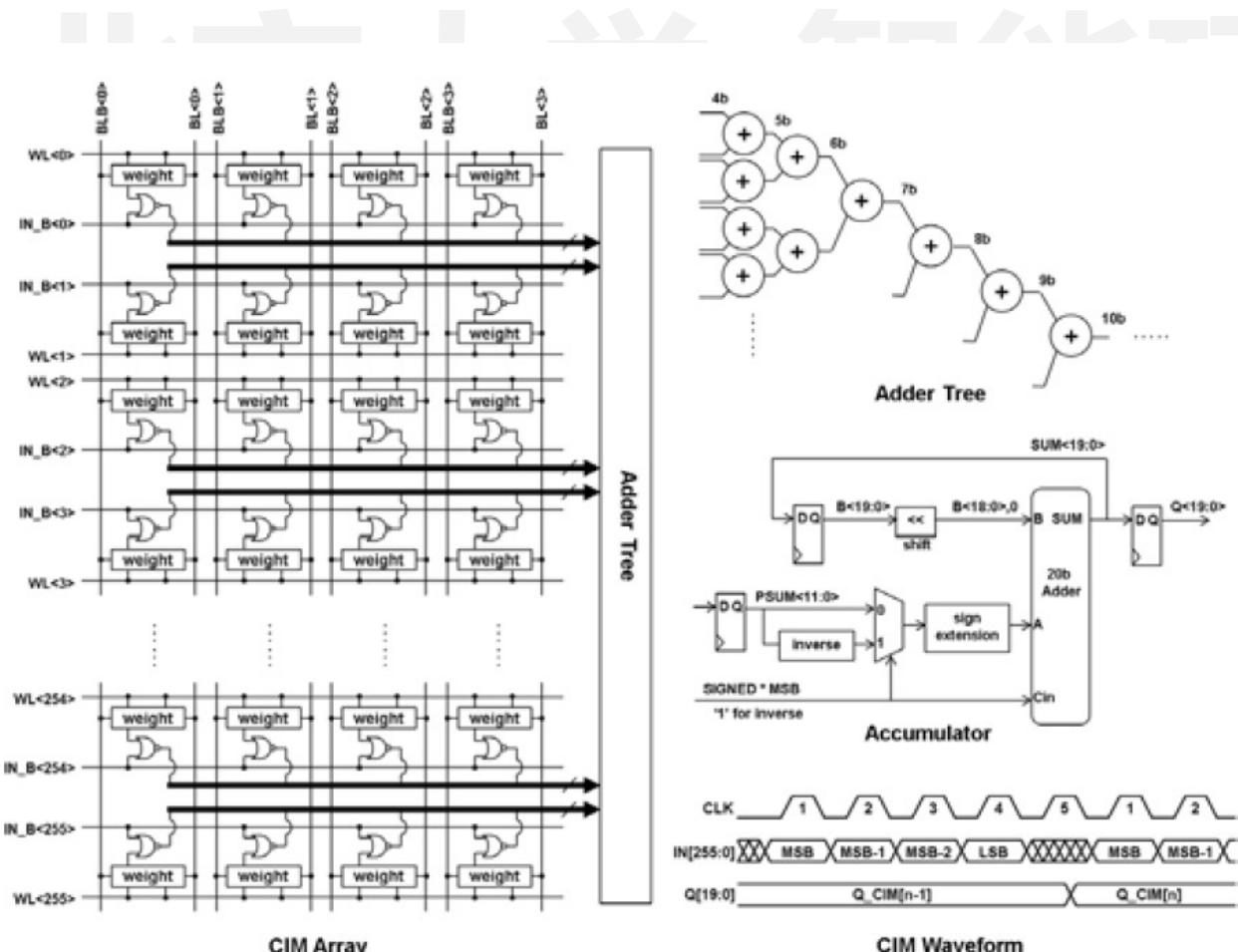
# 代表性架构RAELLA



Tanner Andrulis et al., *RAELLA: Reforming the Arithmetic for Efficient, Low-Resolution, and Low-Loss Analog PIM: No Retraining Required!*, ISCA 2023

## 数字存内计算

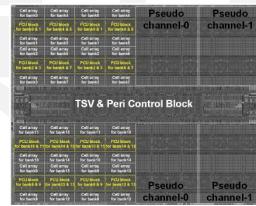
- 为了克服模拟存内计算的精度等挑战，**数字存内计算**被提出，代表性架构包括SRAM数字存算



- 显著降低噪声影响，准确率更高
- 但是计算并行度较低，累加树开销较大

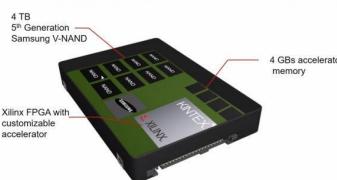
# 存算一体分类

**NDP+DRAM:**  
数字计算、通用  
利用bank级别并行  
神经网络、图计算



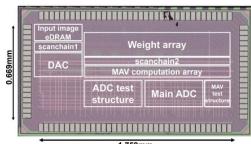
Samsung  
ISSCC 2021

**NDP+NAND Flash:**  
数字计算、通用  
SSD+计算单元  
数据搜索、分析等  
有商业化产品



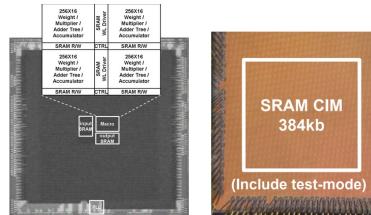
三星 SmartSSD CSD

**CIM+DRAM**  
低比特 (1bit/cell)  
模拟计算  
面向神经网络



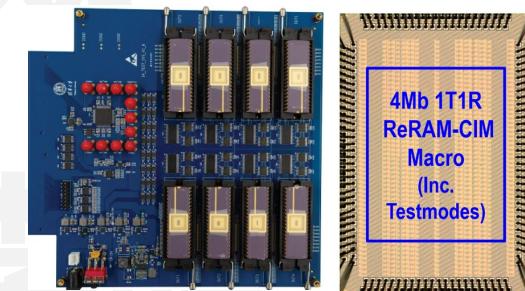
UT Austin, Intel  
ISSCC 2021

**CIM+SRAM**  
中-高比特  
数字、模拟计算  
工艺成熟、频率高  
各种规模神经网络



TSMC  
ISSCC 2021

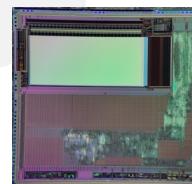
**CIM+新型NVM**  
低、中比特 (1~4bit/cell)  
数字、模拟计算  
工艺成熟度问题  
神经网络、通用逻辑



清华大学  
Nature 2020  
国立清华  
TSMC  
ISSCC 2021

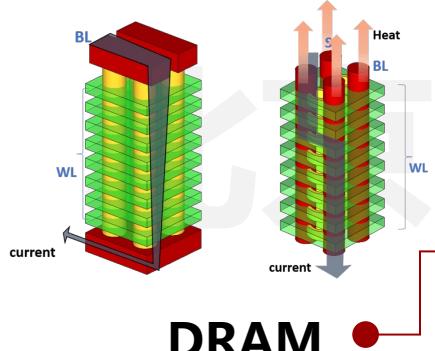
**Nand/Nor Flash**

**CIM+Nor Flash**  
高比特 (>7bit/cell)  
模拟计算、功耗低  
工艺成熟、节点受限  
低功耗AIoT场景



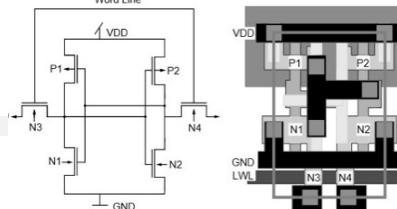
知存科技

# 存算一体分类



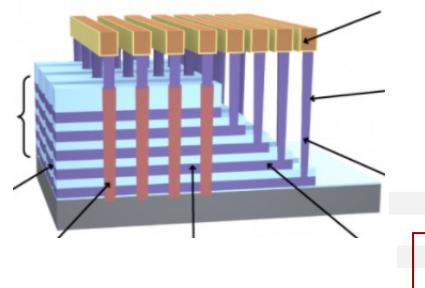
DRAM

**优点:** 工艺成熟、密度高  
**缺点:** 速度低、刷新、只近存  
**非易失性:** 否  
**适合场景:** 冯氏架构过渡



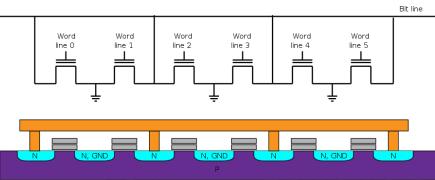
SRAM

**优点:** 工艺成熟、IP化应用  
**缺点:** 能效低、密度低  
**非易失性:** 否  
**适合场景:** 端侧、边缘中小算力



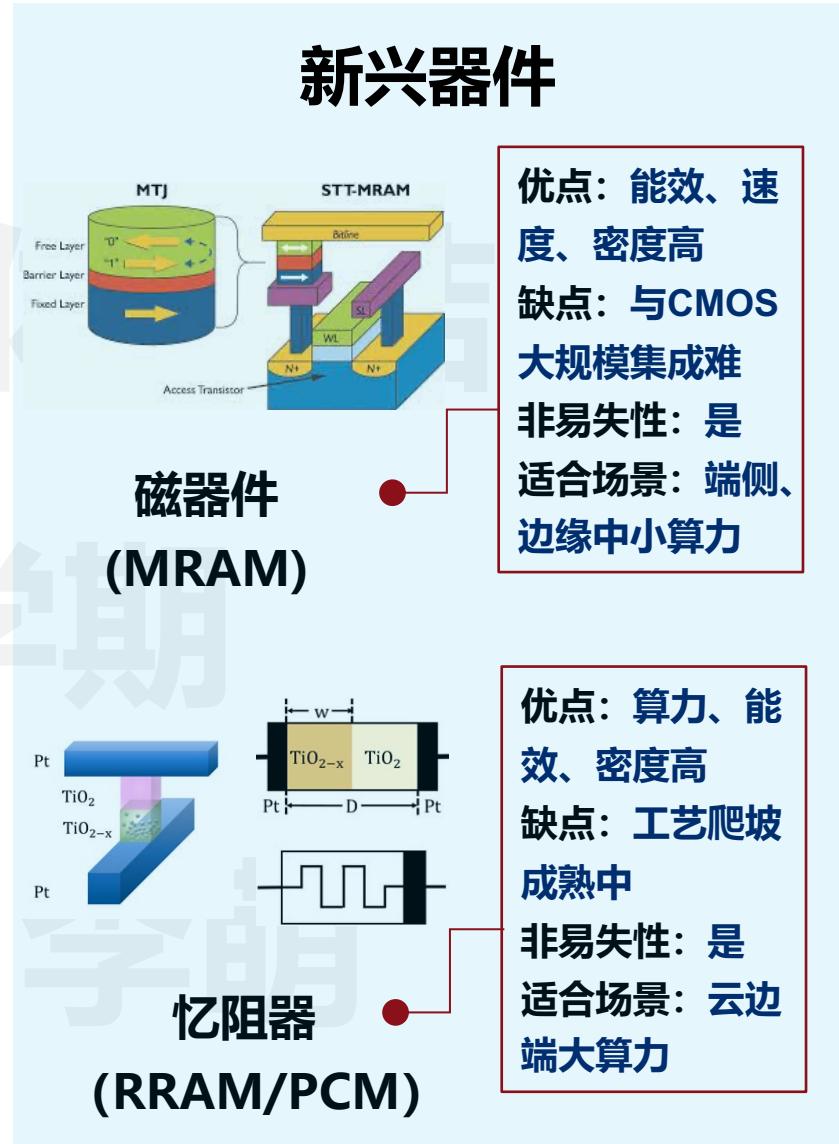
SSD/Nand Flash

**优点:** 工艺成熟、容量大、成本低  
**缺点:** 速度低、只能近存  
**非易失性:** 是  
**适合场景:** 云端大容量



Nor Flash

**优点:** 工艺成熟、密度高、成本低  
**缺点:** 对PVT变化敏感、能效低  
**非易失性:** 是  
**适合场景:** 端侧、边缘低成本



- 课程评估：扫右侧二维码或登录网上评估系统

([kcpg.pku.edu.cn](http://kcpg.pku.edu.cn))

- 希望大家后续科研学习一切顺利，也欢迎在课程结束后，同学们仍然能找我讨论有趣的问题

- 办公室：资源西楼2213A
- 邮箱：[meng.li@pku.edu.cn](mailto:meng.li@pku.edu.cn)
- 电话：18701103305 (微信同)



主讲：陶耀宇、李萌