AI 芯片 – AI 计算体系

矩阵运算&比特位

ZOMI

Talk Overview

1. AI 计算体系

- 深度学习计算模式
- 计算体系与矩阵运算

2. AI 芯片基础

- 通用处理器 CPU
- 从数据看 CPU 计算
- 通用图形处理器 GPU
- AI专用处理器 NPU/TPU
- 计算体系架构的黄金10年

Talk Overview

I. AI 计算体系与矩阵运算

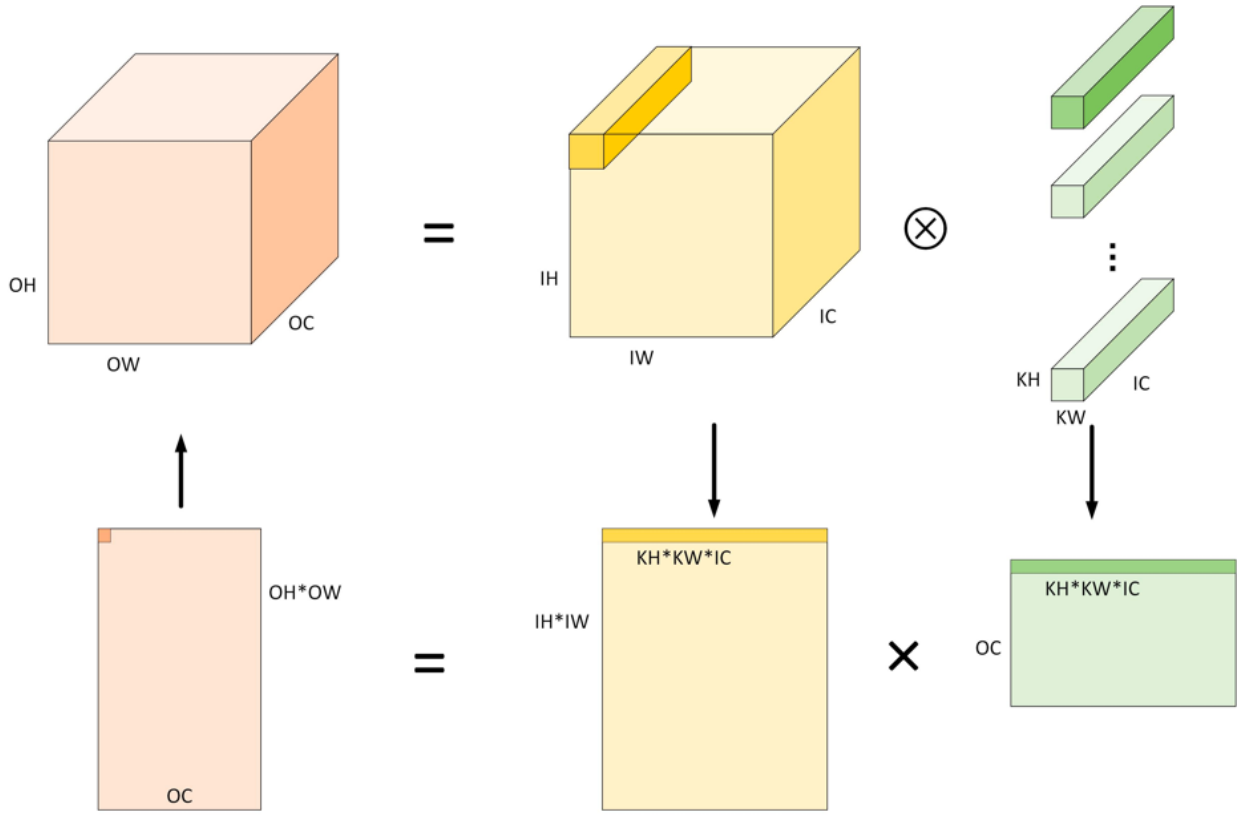
- Key Metrics – AI芯片关键指标
- Matrix Multiplication – 矩阵运算
- Bit Width – 比特位数
- Specialized Hardware – 专用硬件

Matrix Multiplication

矩阵运算

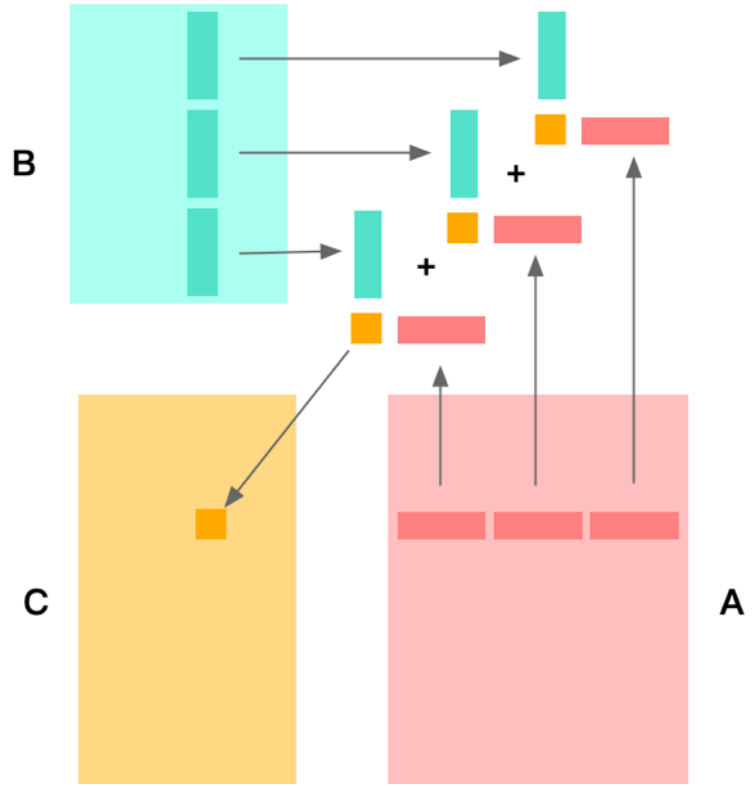
从卷积 Conv 到矩阵乘 MM

- 通过数据重排，完成 Im2col 的操作之后会得到一个输入矩阵，卷积的 Weights 也可以转换为一个矩阵，卷积的计算就可以转换为两个矩阵相乘的求解，得到最终的卷积计算结果。

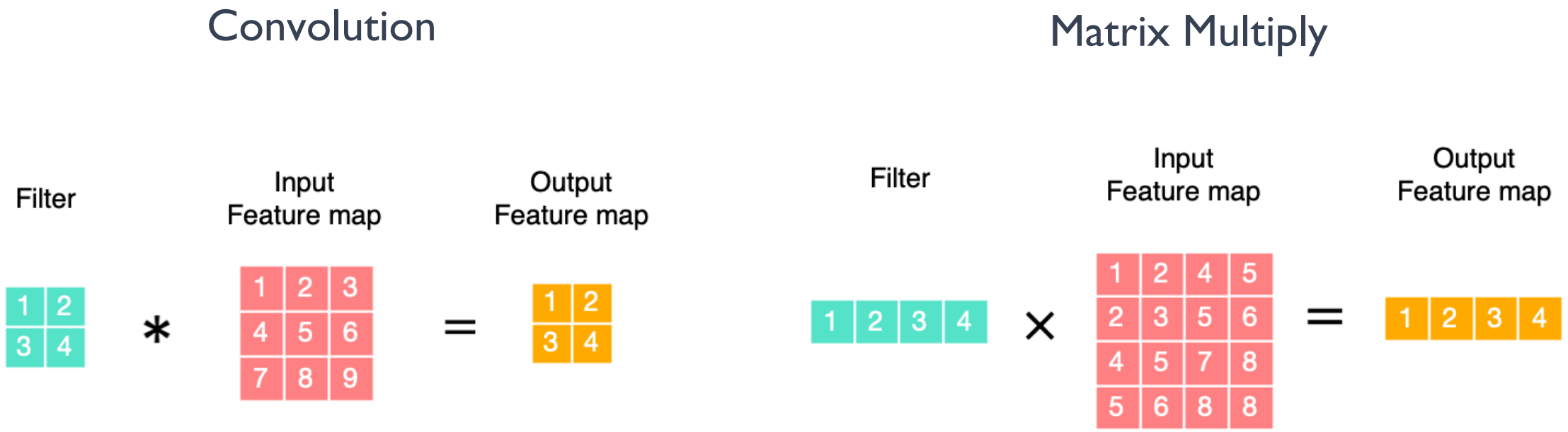


从卷积 Conv 到矩阵乘 MM

- 在计算 $MR \times NR$ 小块时，传统的方法是在 K 维度上拆分，在一次计算 Kernel 处理中，仅计算 K 维的局部。那么在每次计算 Kernel 的处理中，都会发生对输出的加载和存储。

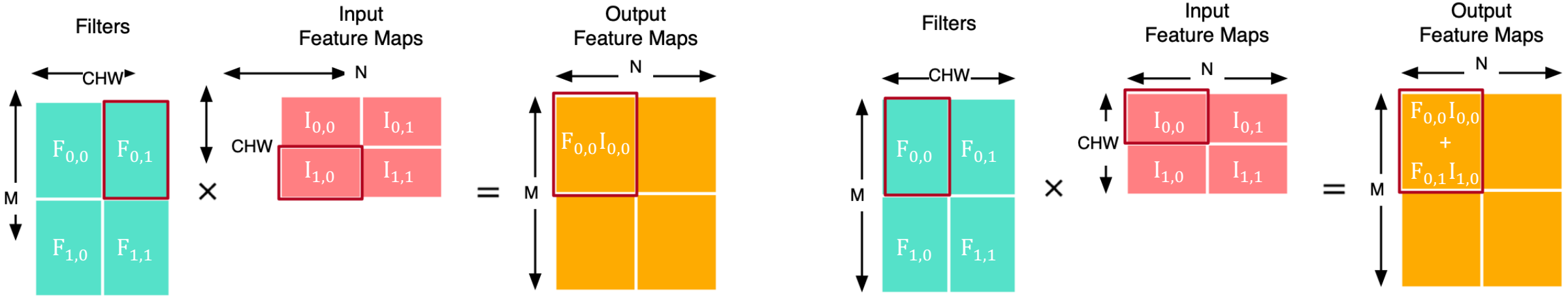


从卷积 Conv 到矩阵乘 MM



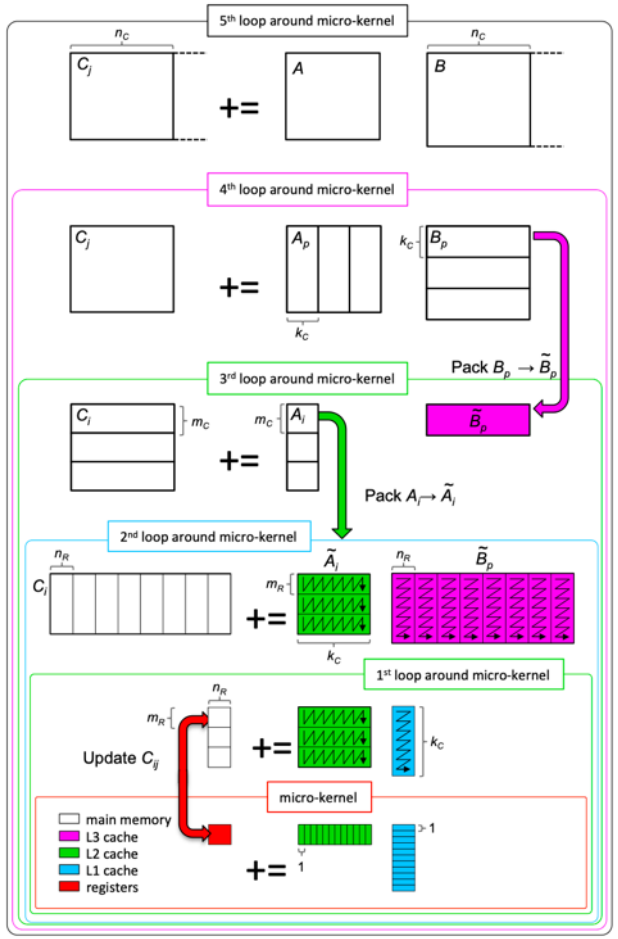
矩阵乘分块 Tiling

- 根据 Cache 大小来对矩阵进行分块 Tiling，最大程度重用数据和利用空间换时间



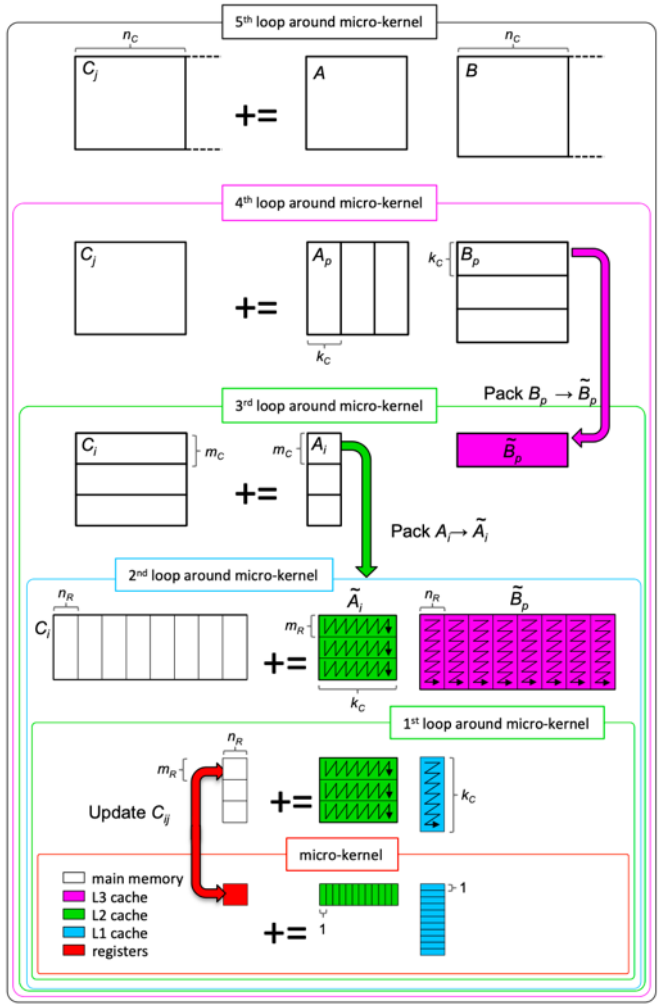
CPU/GPU 支持矩阵乘的库

- **实现逻辑：**
  - Lib 感知相乘矩阵的 Shape
  - 选择最优的 Kernel 实现来执行
- **实现方法：**
  - Loop 循环优化 (Loop tiling)
  - 多级缓存 (memory hierarchy)
- **现有库：**Matrix Multiplication (GEMM)
  - CPU: OpenBLAS, Intel MKL, etc.
  - GPU: cuBLAS, cuDNN, etc.



CPU/GPU 支持矩阵乘的库

Loop 5 **for** $j_c = 0 : n-1$ **steps of** n_c
$\mathcal{J}_c = j_c : j_c + n_c - 1$
Loop 4 **for** $p_c = 0 : k-1$ **steps of** k_c
$\mathcal{P}_c = p_c : p_c + k_c - 1$
$B(\mathcal{P}_c, \mathcal{J}_c) \rightarrow B_c$ // Pack into B_c
Loop 3 **for** $i_c = 0 : m-1$ **steps of** m_c
$\mathcal{I}_c = i_c : i_c + m_c - 1$
$A(\mathcal{I}_c, \mathcal{P}_c) \rightarrow A_c$ // Pack into A_c
// Macro-kernel
Loop 2 **for** $j_r = 0 : n_c - 1$ **steps of** n_r
$\mathcal{J}_r = j_r : j_r + n_r - 1$
Loop 1 **for** $i_r = 0 : m_c - 1$ **steps of** m_r
$\mathcal{I}_r = i_r : i_r + m_r - 1$
// Micro-kernel
Loop 0 **for** $k_r = 0 : k_c - 1$
$C_c(\mathcal{I}_r, \mathcal{J}_r)$
$\mathrel{+}= A_c(\mathcal{I}_r, k_r) \;\; B_c(k_r, \mathcal{J}_r)$
endfor
endfor
endfor
endfor



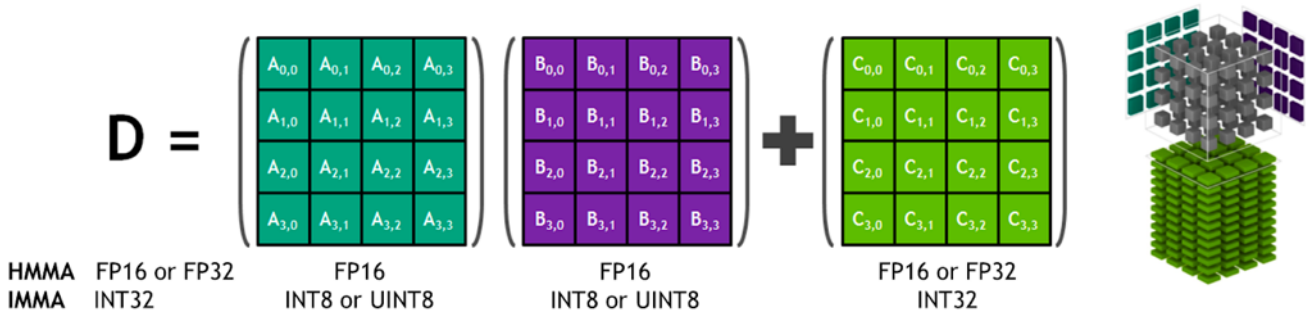
卷积代替算法

- **Fast Fourier Transform** [mathieu, ICLR 2014]
  - Pro：计算量从 $O(N_0^2 N_f^2)$ to $O(N_0^2 \text{long}_2 N_0)$
  - Con：增加内存存储空间
- **Strassen** [Cong, ICANN 2014]
  - Pro：计算量从 $O(N^3)$ to $O(N^{2.807})$
  - Con：数值稳定性问题
- **Winograd** [Lavin, CVPR 2016]
  - Pro：3x3等小矩阵由2.X倍加速
  - Con：额外辅助矩阵，辅助矩阵依赖于卷积核大小

减少指令开销

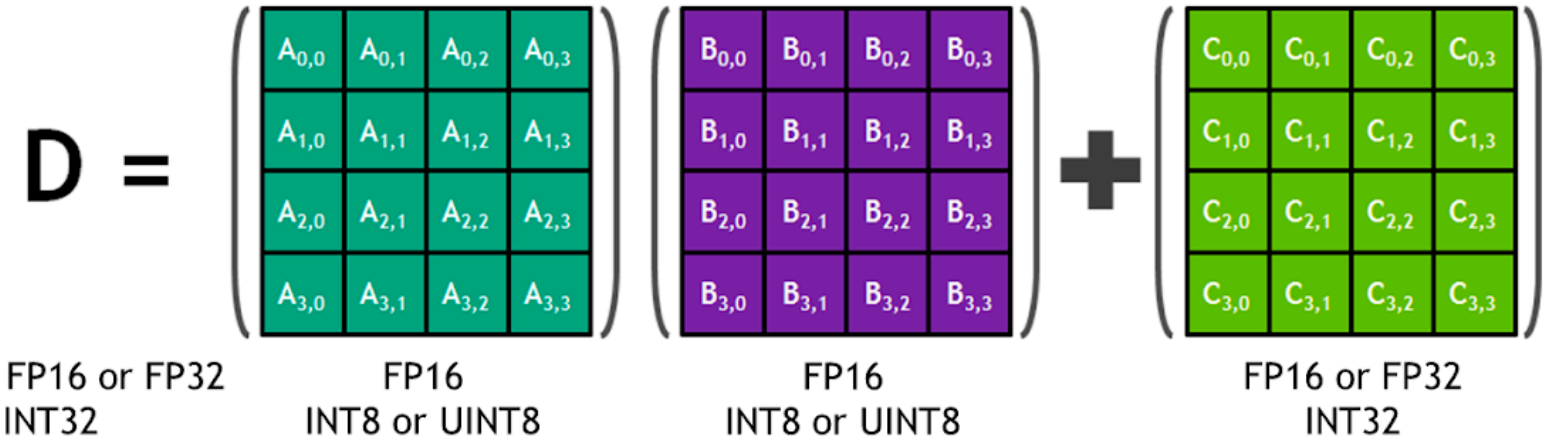
- 每个指令执行更多的 MACs 计算
  - CPU：SIMD / Vector 指令
  - GPU：SIMT / Tensor 指令
  - NPU：SIMD / Tensor 、 Vector 指令

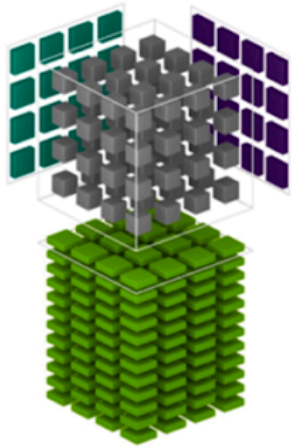
- 在不增加内存带宽的前提下，单时钟周期内执行更多的 MACs 计算
  - 支持低比特计算 (e.g., 512bits per cycle, perform **64** 8-bit MACs vs **16** 32-bit MACs)



减少指令开销

- 在不增加内存带宽的前提下，单时钟周期内执行更多的 MACs 计算
  - 支持低比特计算 (e.g., 512bits per cycle, perform **64** 8-bit MACs vs **16** 32-bit MACs)





对于矩阵运算 计算体系思考 I

I. 软件 Software

- 减少没有必要的 MACs：使用其他代替算法
- 增加 PE 利用率：对kernel实现进行Loop优化和Memory优化

对于矩阵运算 计算体系思考 II

I. 硬件 Hardware

- **减少每次 MAC 计算的时间**
  - 增加 PE 单元计算能力
- **增加 MACs 并行计算能力**
  - 增加片内 PE 数量
  - 支持低bits数PE计算
- **增加 PE 利用率**
  - 增加片内 Cache
  - 额外的内存带宽