2023年春季

智能芯片设计

第3章 智能计算特性与加速原理

华中科技大学 人工智能与自动化学院
多谱信息智能处理技术全国重点实验室

本章内容

3.1 典型CNN/RNN的计算模型

3.2 网络层的计算原理及特点

3.3 智能计算的挑战分析

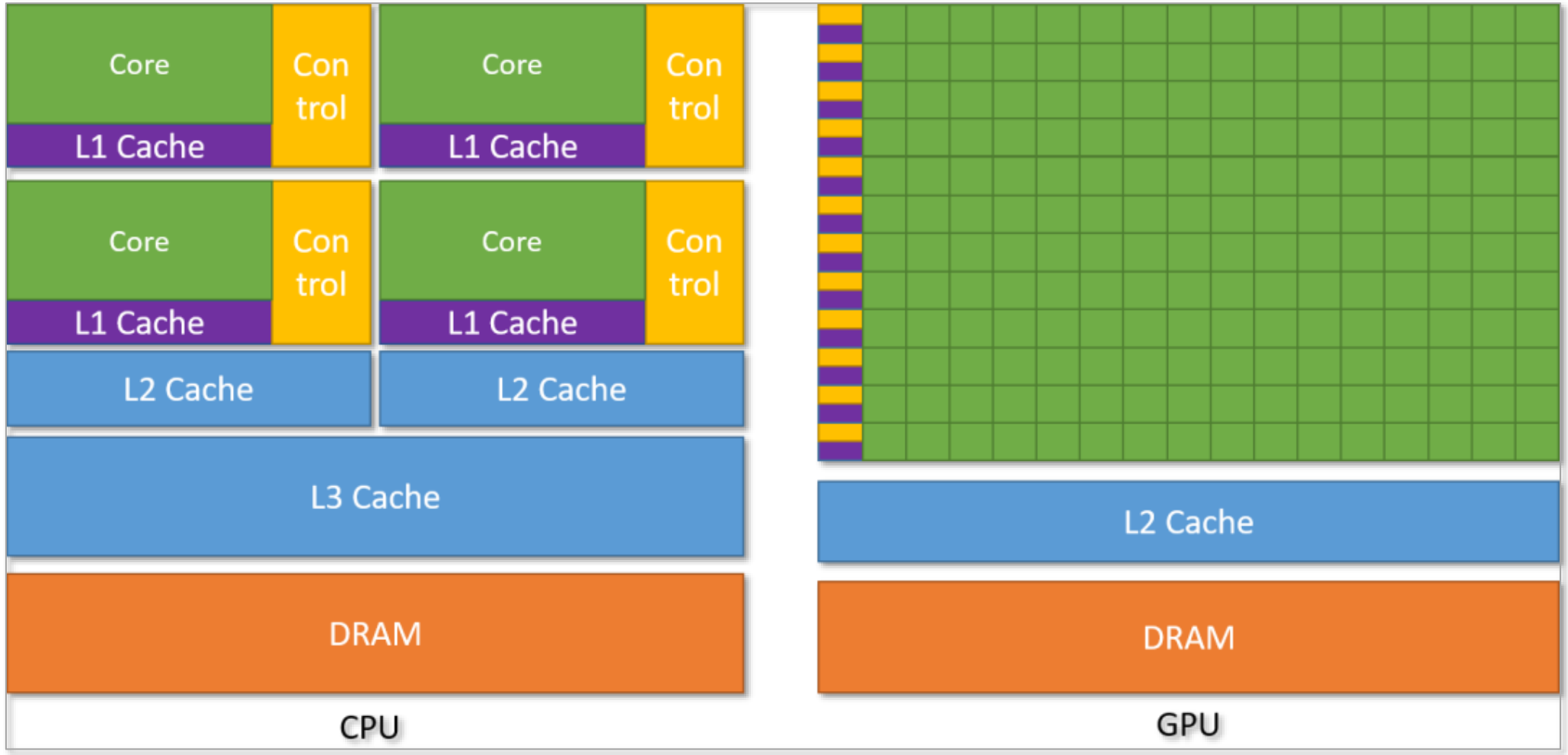
3.4 卷积层并行计算原理

3.5 数据复用技术

3.6 GPU/TPU网络计算加速原理

3.6.1 GPU网络计算加速原理

口现代GPU架构



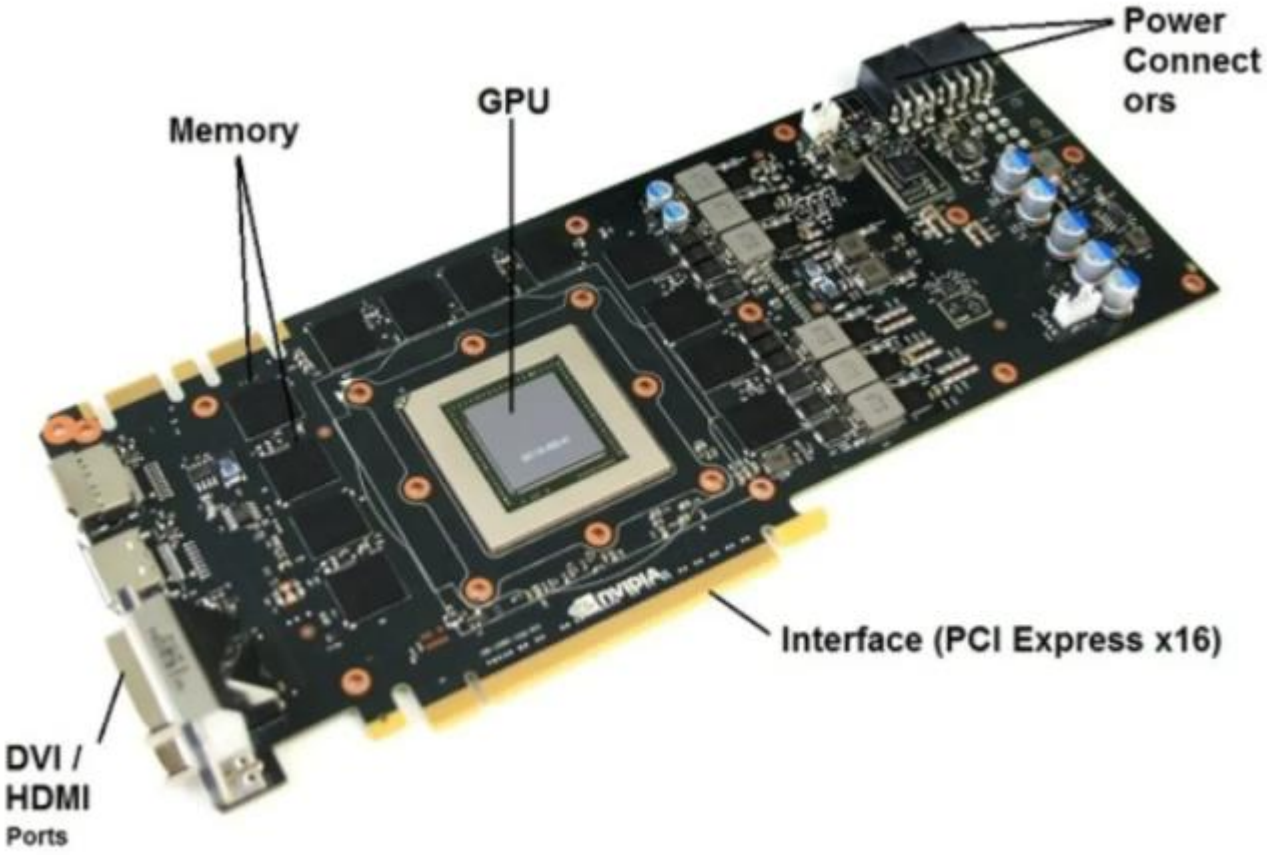
3.6.1 GPU网络计算加速原理

现代GPU架构: NVIDIA GPU

- Tesla（特斯拉）2008年，应用于早期的 CUDA 系列显卡
- Fermi（费米）2010年，第一个完整的 GPU 计算架构
- Kepler（开普勒）2012年，Fermi 的优化版
- Maxwell（麦克斯韦）2014年，首次支持实时的动态全局光照效果
- Pascal（帕斯卡）2016年，GPU 将处理器和数据集成在同一个程序包内，实现更高的计算效率
- Volta（伏打）2017年，首次将一个 CUDA 内核拆分为FP32 和 INT32 两部分，首次支持混合精度运算，提高了5倍于 Pascal 计算速度，还增加了专用于深度学习的 Tensor Core 张量单元
- Turing（图灵）2018年，增加了 RT Core 专用光线追踪处理器，去掉了对 FP64 计算的支持
- Ampere（安培）2020年，重新支持 FP64，新增 BF16 数据类型，专为深度学习优化

3.6.1 GPU网络计算加速原理

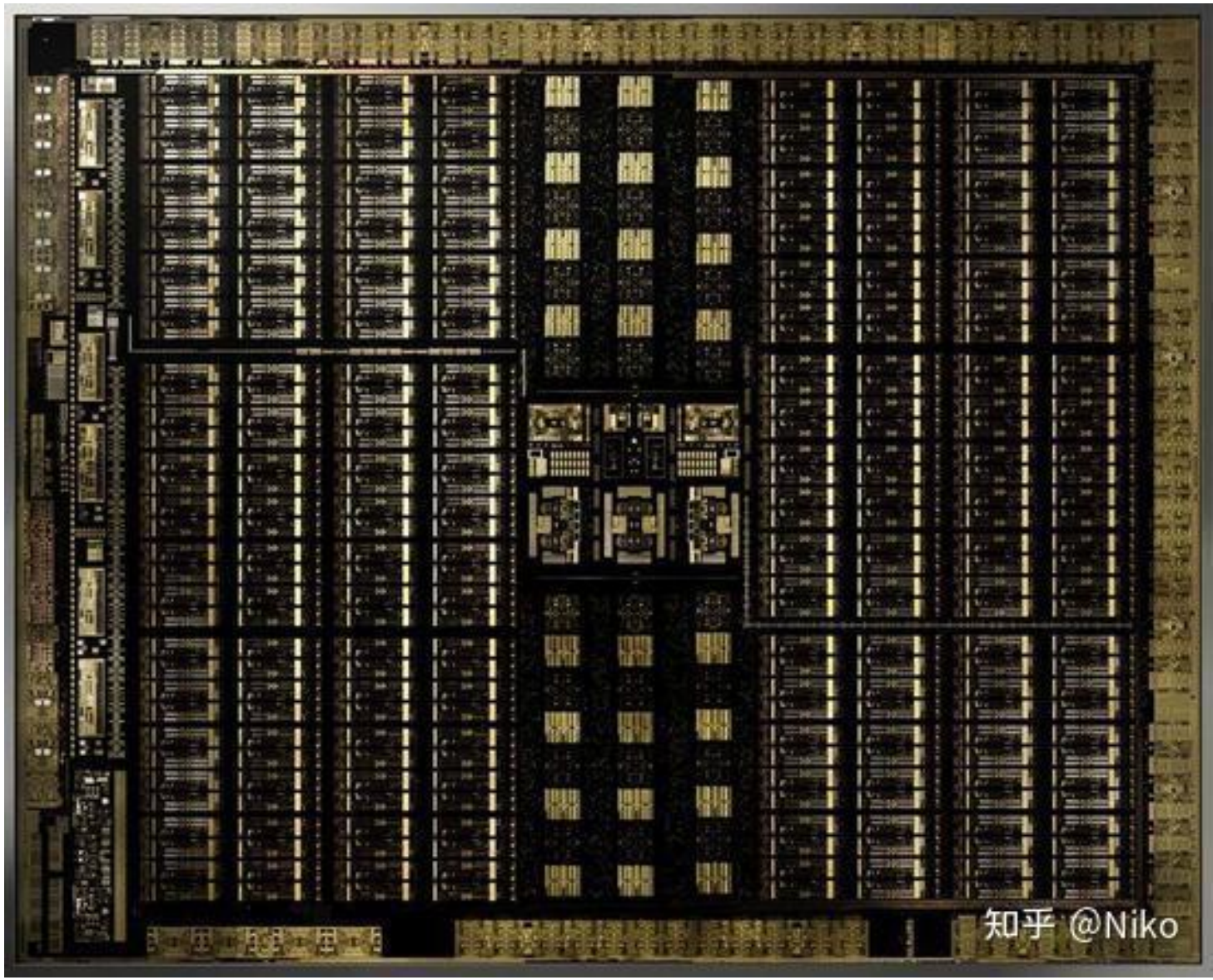
- RTX2080Ti



3.6.1 GPU网络计算加速原理

- 基于TU102的RTX2080Ti

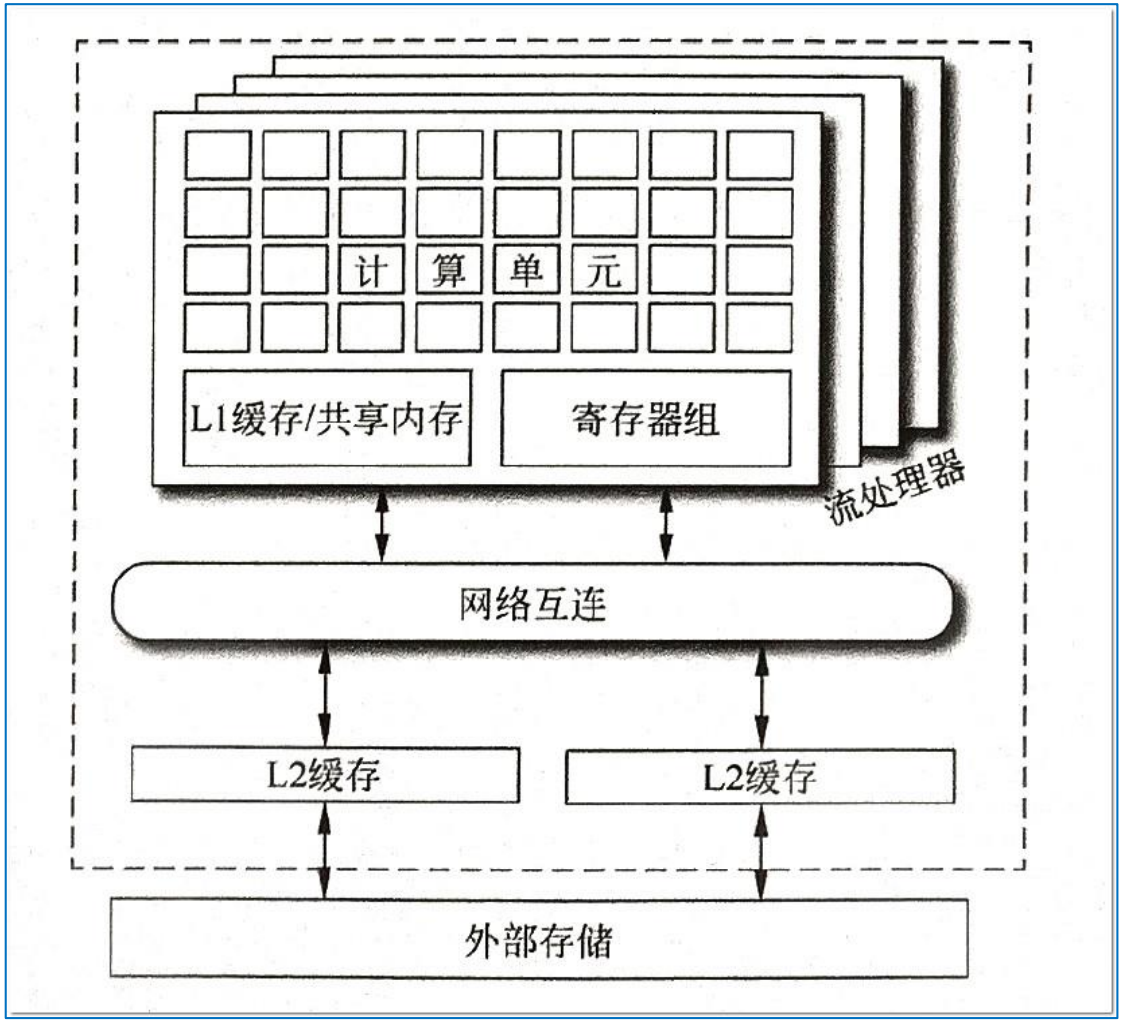
TU102旗舰级核心:RTX2080Ti
TU104高端核心:RTX2080
TU106中端核心:RTX2070



3.6.1 GPU网络计算加速原理

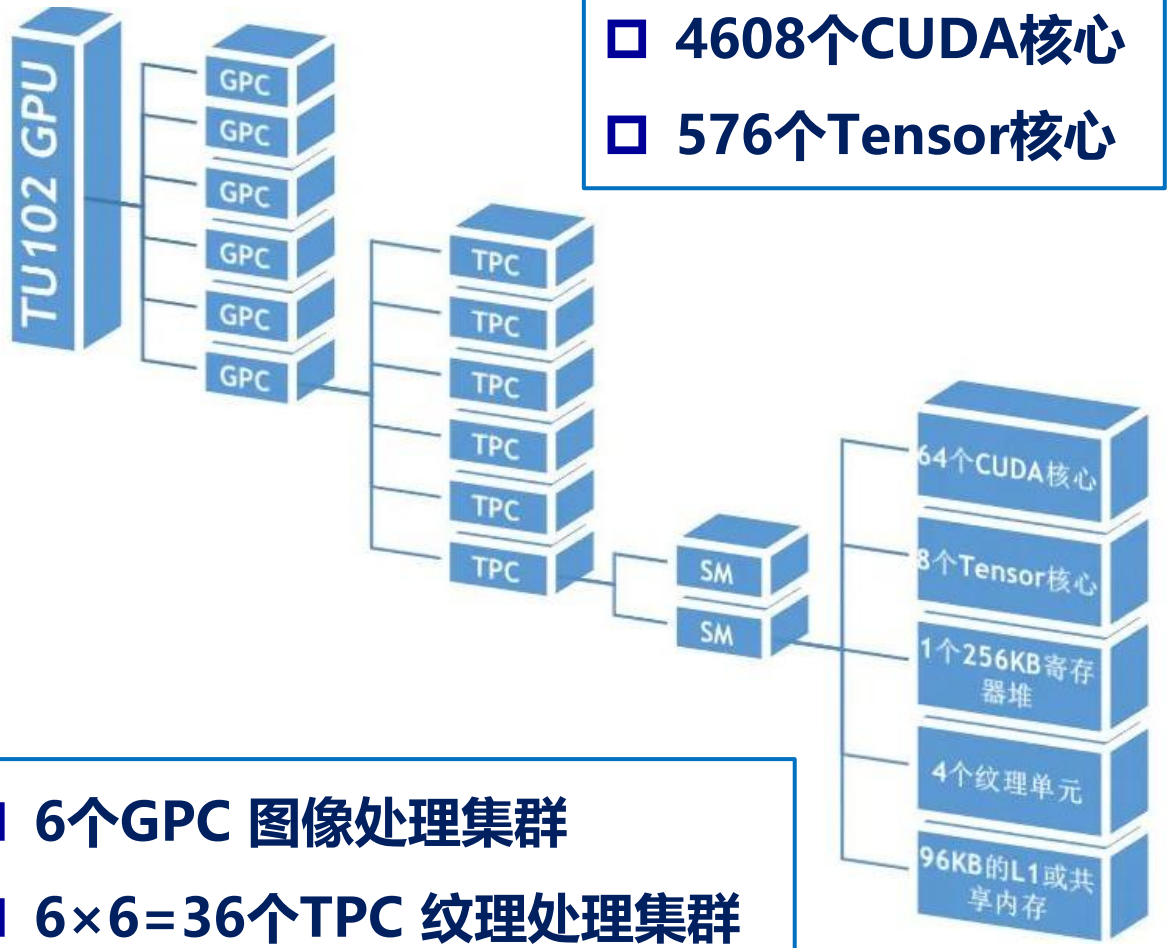
现代GPU架构: NVIDIA GPU

- 流处理器
- 多层级片上存储器
- 网络互连结构
- Turing架构TU102
  - 72个流处理器
  - L1缓存+共享内存+寄存器组+L2缓存+外部存储



3.6.1 GPU网络计算加速原理

- 基于TU102的RTX2080Ti

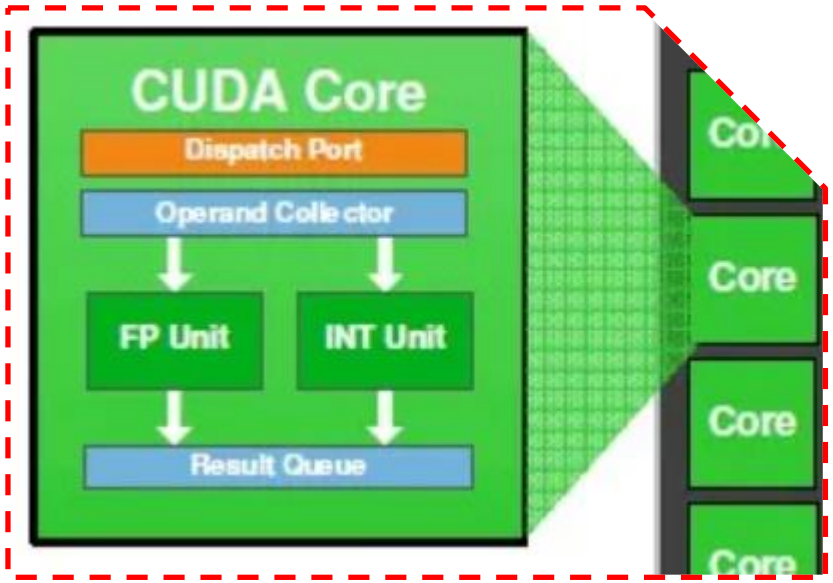


- 4608个CUDA核心
- 576个Tensor核心

- 6个GPC 图像处理集群
- 6×6=36个TPC 纹理处理集群
- 36×2=72个SM 流处理器

3.6.1 GPU网络计算加速原理

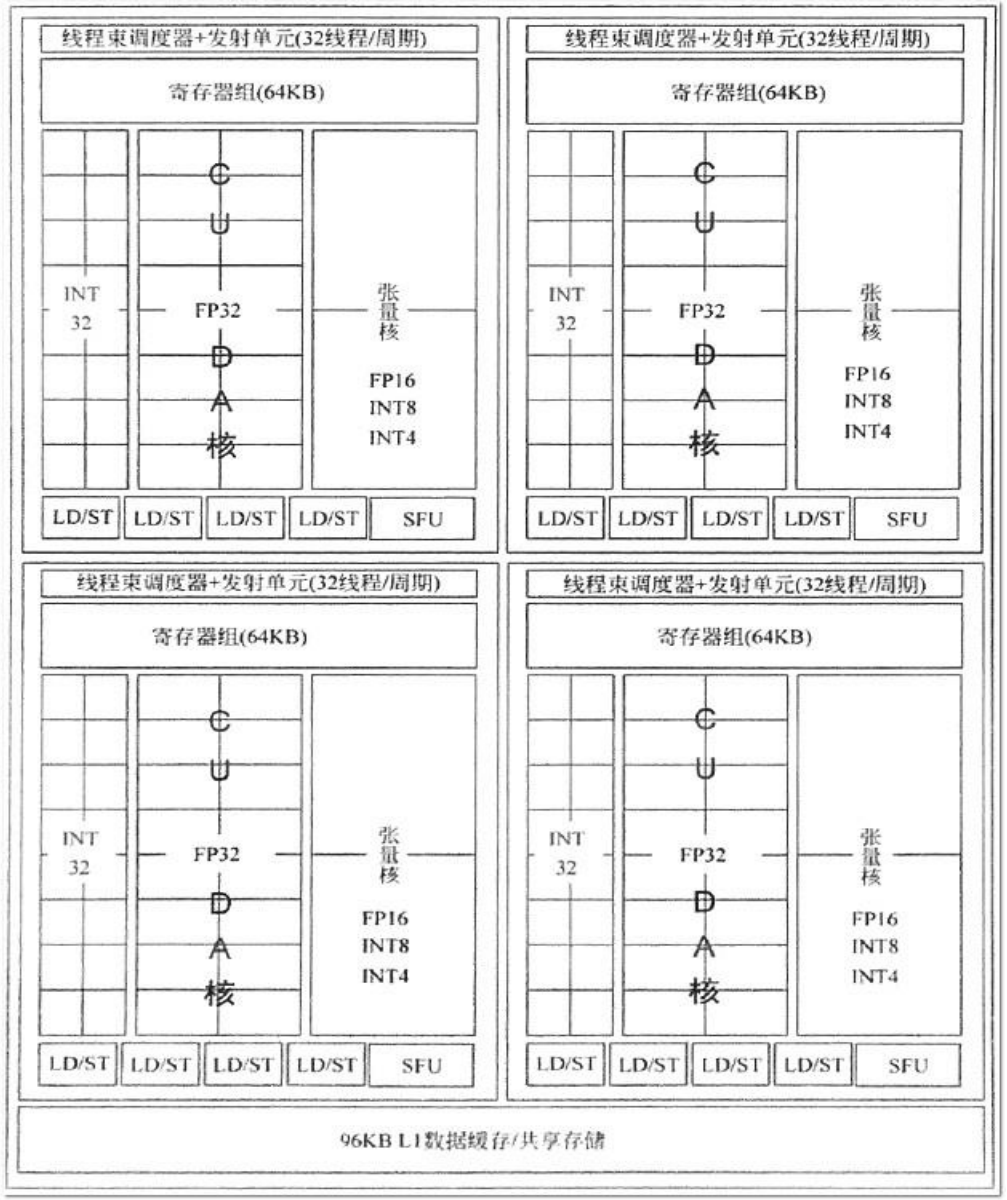
- 基于TU102的RTX2080Ti



3.6.1 GPU网络计算加速原理

Turing流处理器

- 每个含64个CUDA核，负责FP32计算
- SIMT方式在CUDA框架下调用
- 8个张量核，支持FP16、INT8和INT4等多精度计算
- 通过WMMA指令在CUDA框架下调用张量核
- 256KB寄存器组、96KB的L1缓存存放数据
  - 每个线程分配一定数量的寄存器存放程序变量
  - L1缓存可配置为共享内存
  - L1缓存执行机制由内部控制



3.6.1 GPU网络计算加速原理

□ CUDA 编程模型—两层 API 来调用底层 GPU 硬件

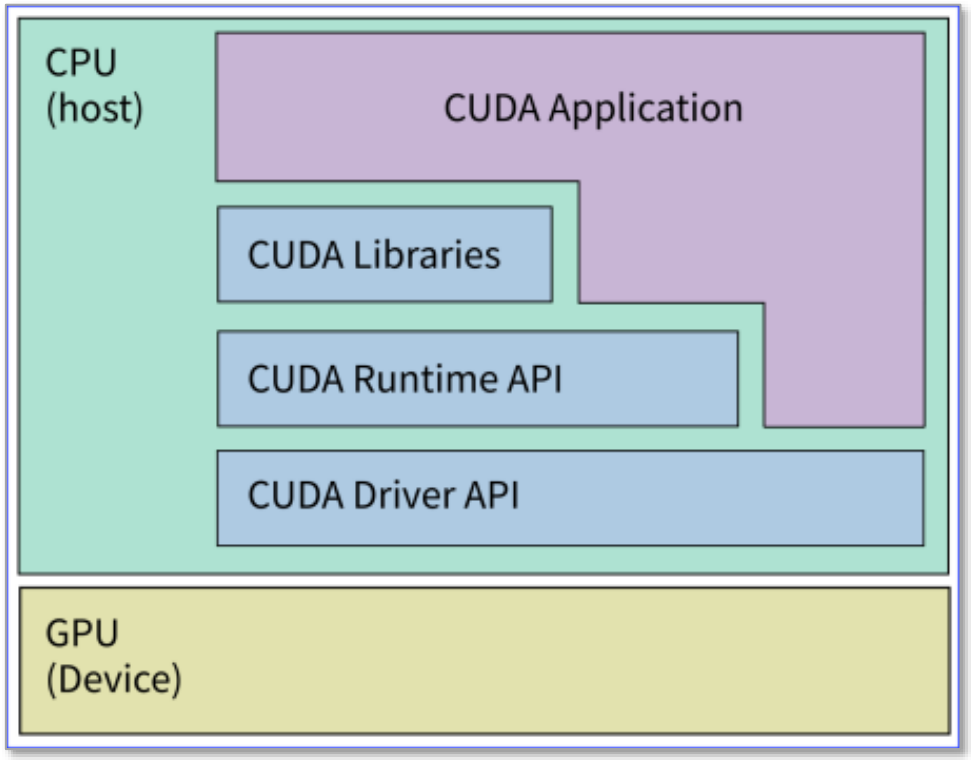
- **CUDA 驱动 API (CUDA Driver API)**

  基于句柄的底层接口，大多数对象通过句柄被引用，可以通过直接操作硬件执行一些复杂的功能

- **CUDA运行时 API (CUDA Runtime API)**

  对 Driver API 进行了封装，隐藏了部分实现细节，更多使用的是 Runtime API

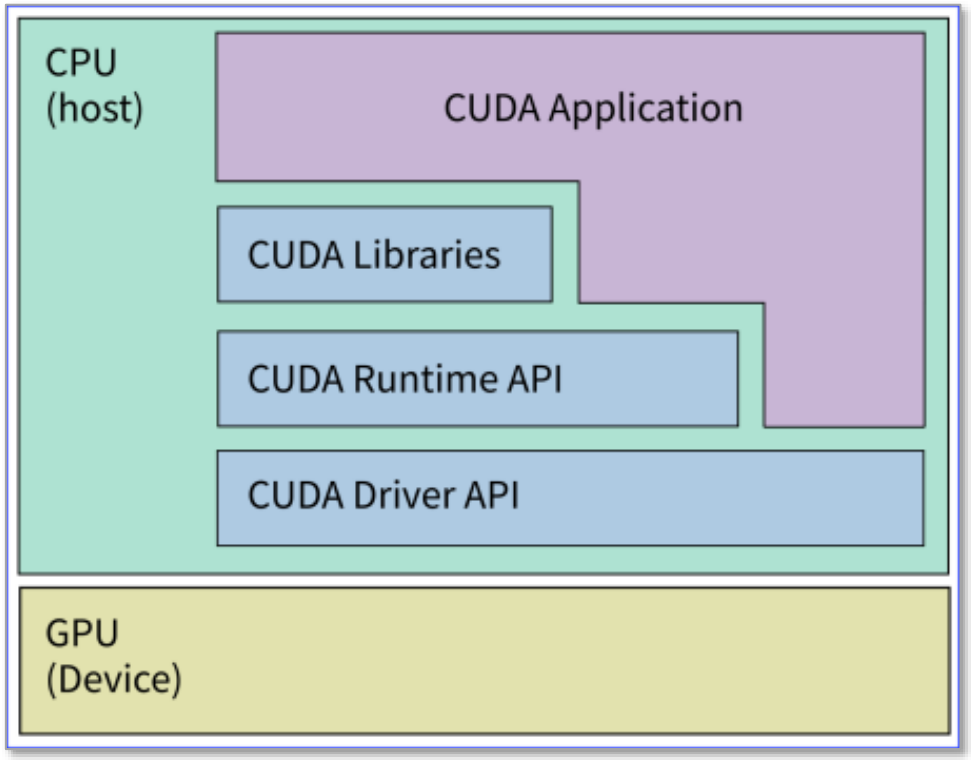
- **Runtime API 和 Driver API 之间没有明显的性能差距，两种不能混合使用，单独使用其一**



3.6.1 GPU网络计算加速原理

□ CUDA 编程模型—CUDA 函数库 (CUDA Libraries)

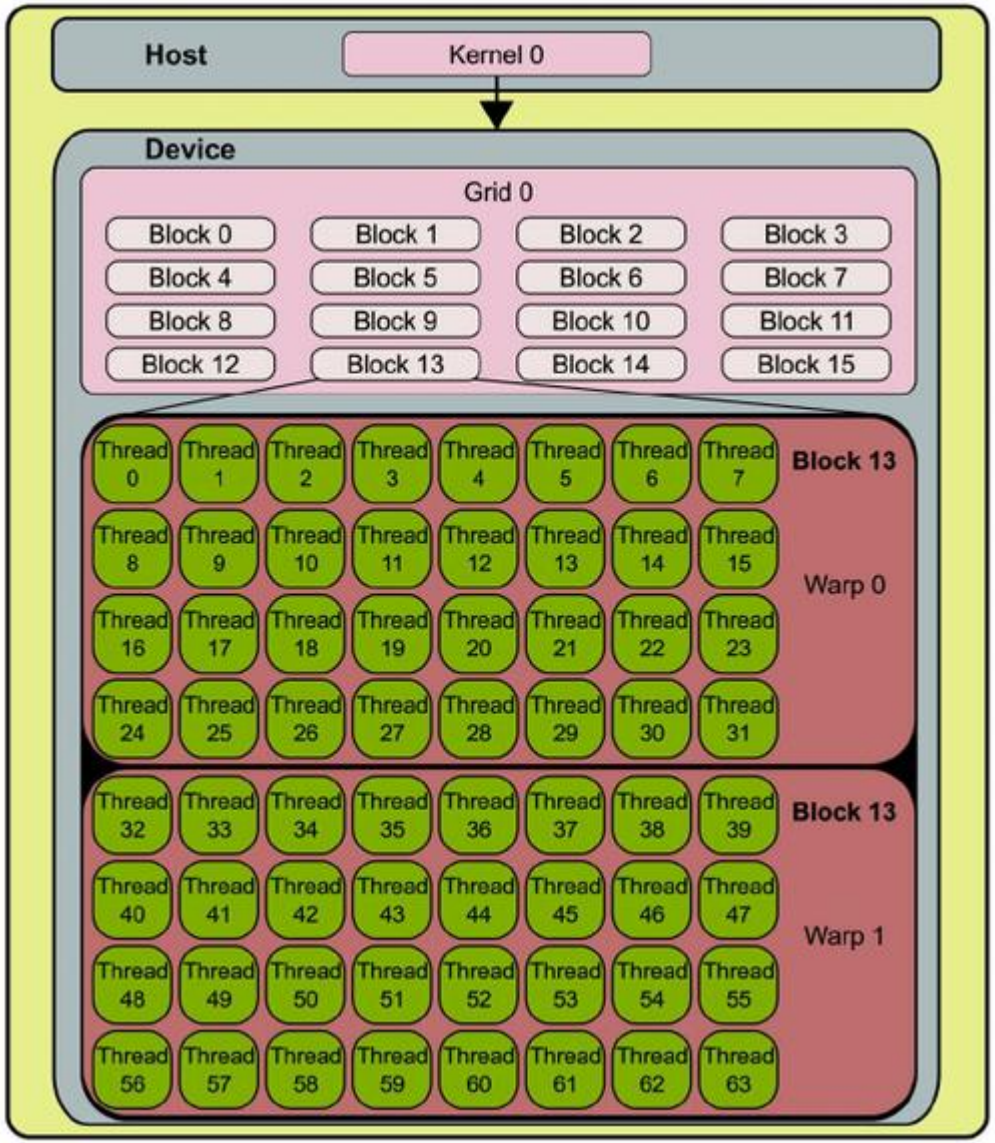
- ■ CUDA 提供了成熟的高效函数库
  - ◆ CUFFT：利用 CUDA 进行傅立叶变换的函数库
  - ◆ CUBLAS：利用 CUDA 进行加速的完整标准矩阵与向量的运算库
  - ◆ CUDPP：并行操作函数库
  - ◆ CUDNN：利用 CUDA 进行加速的深度卷积神经网络的运算库



3.6.1 GPU网络计算加速原理

□ CUDA 线程模型：线程是程序执行的最基本单元

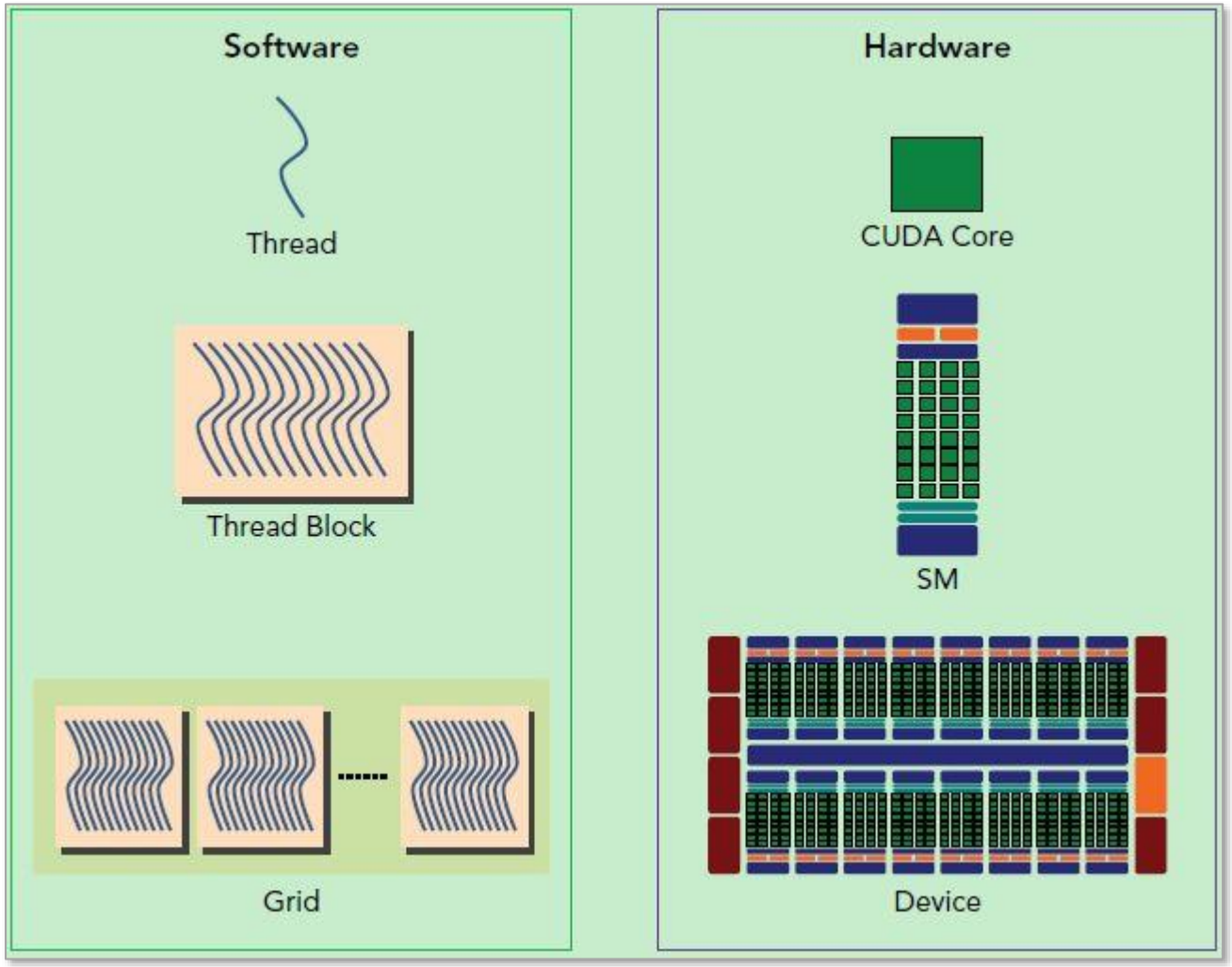
- Thread：线程，并行的基本单位
- Block：线程块，互相合作的线程组
  - 以1维、2维或3维组织
  - 允许彼此同步
  - 可以通过共享内存快速交换数据
- Grid，网格，由一组 Block 组成
  - 共享全局内存
  - 以1维、2维组织
- warp：GPU执行程序时的调度单位，一个warp包含32个并行thread，对不同数据资源执行相同的指令,即SIMT



3.6.1 GPU网络计算加速原理

□ CUDA 软件和硬件对应结构

软件	硬件	描述
Thread	SP	每个线程由每个线程处理器（SP）执行
Block	SM	线程块由多核处理器（SM）执行
Grid	Device	一个 kernel 由一个 grid 来执行，一次只能在一个 GPU 上执行



3.6.1 GPU网络计算加速原理

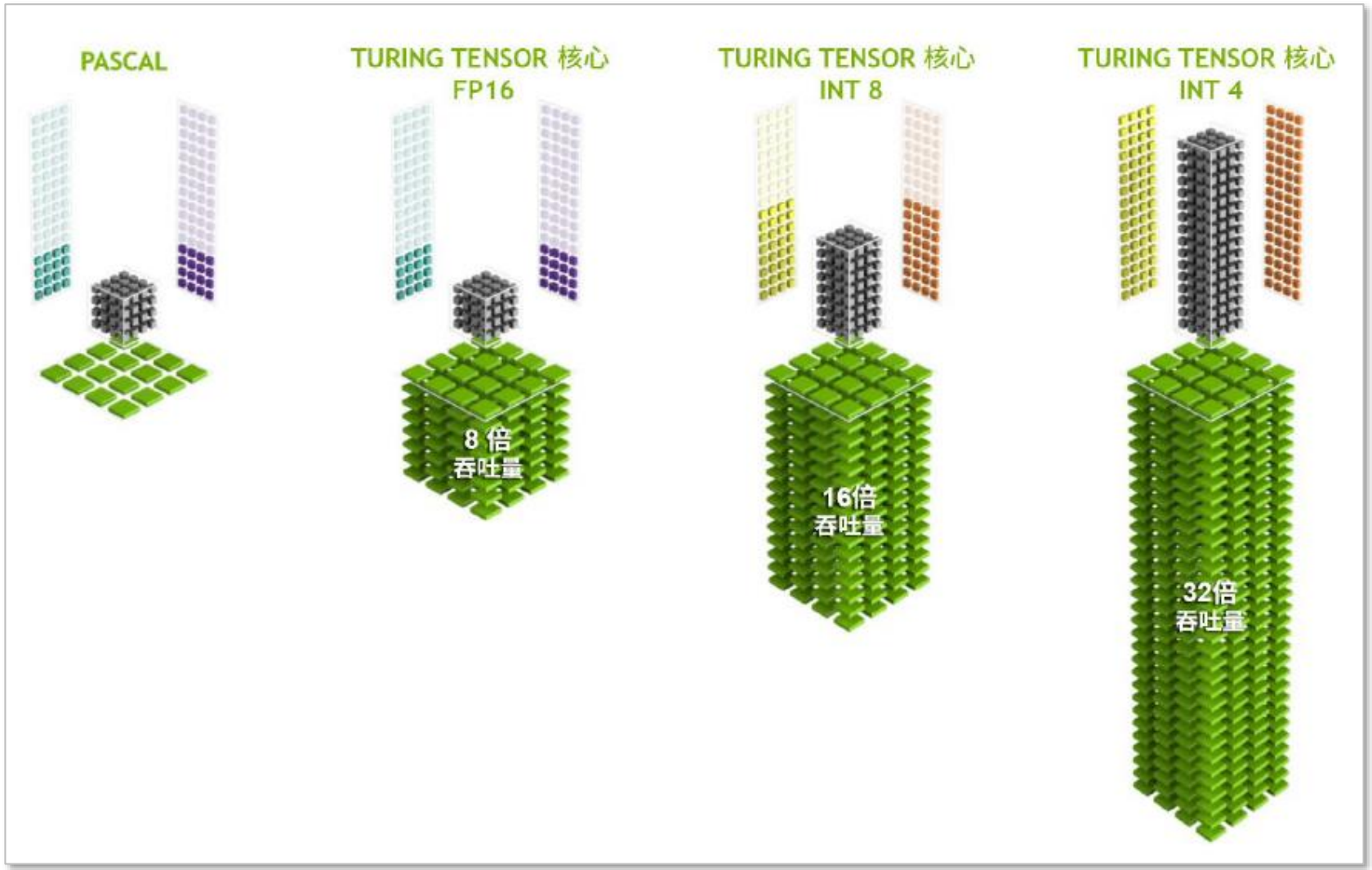
张量核Tensor Core

- 每个TC单周期内可执行64个FP16融合乘加运算(FMA)
- 每个流处理器单周期可实现512个FP16融合乘加运算(FMA)
- 每个流处理器单周期可实现2048个INT8融合乘加运算(FMA)

$$\begin{pmatrix} D_{00} & D_{01} & D_{02} & D_{03} \\ D_{10} & D_{11} & D_{12} & D_{13} \\ D_{20} & D_{21} & D_{22} & D_{23} \\ D_{30} & D_{31} & D_{32} & D_{33} \end{pmatrix} = \begin{pmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & A_{22} & A_{23} \\ A_{30} & A_{31} & A_{32} & A_{33} \end{pmatrix} \times \begin{pmatrix} B_{00} & B_{01} & B_{02} & B_{03} \\ B_{10} & B_{11} & B_{12} & B_{13} \\ B_{20} & B_{21} & B_{22} & B_{23} \\ B_{30} & B_{31} & B_{32} & B_{33} \end{pmatrix} + \begin{pmatrix} C_{00} & C_{01} & C_{02} & C_{03} \\ C_{10} & C_{11} & C_{12} & C_{13} \\ C_{20} & C_{21} & C_{22} & C_{23} \\ C_{30} & C_{31} & C_{32} & C_{33} \end{pmatrix}$$

3.6.1 GPU网络计算加速原理

□ 张量核Tensor Core

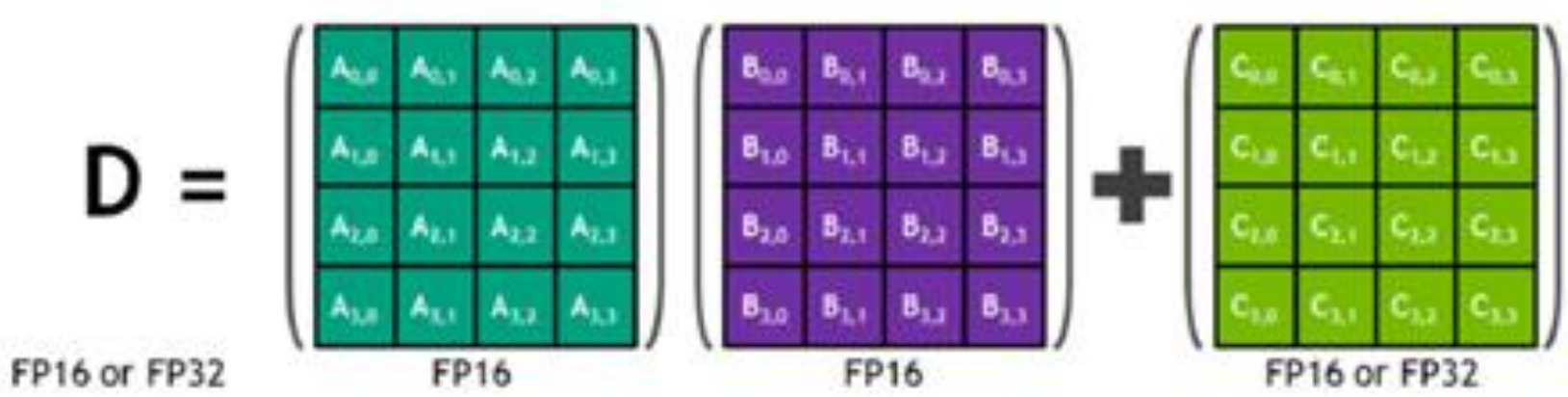


3.6.1 GPU网络计算加速原理

□ 张量核Tensor Core

- 调用WMMA(Warp-level Matrix Multiply and Accumulate)指令编程TC
- 指令进入TC后分解为细粒度HMMA(Half Precision Matrix Multiply Accumulate)指令，控制流处理器中的线程，每个线程在一个时钟周期内完成1次4个数的点积运算(FEDPs)
- 每个TC在一个时钟周期内完成16次FEDPs，即每个TC包含16个线程
- 每个TC在**一个时钟周期内**完成2个4×4矩阵相乘并累加一个4×4矩阵的计算

$$D = A \times B + C$$

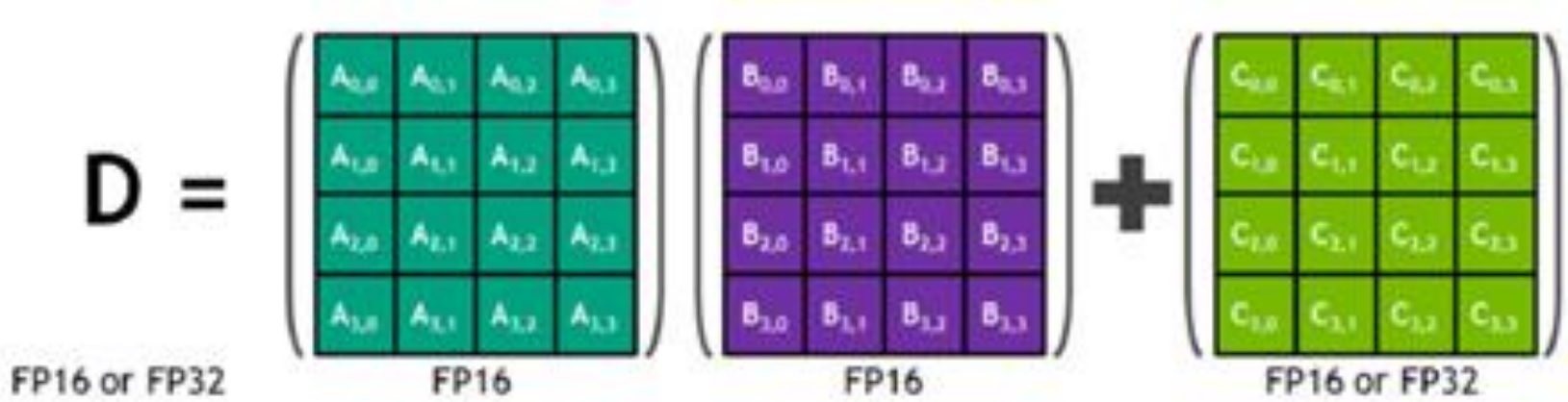


3.6.1 GPU网络计算加速原理

口 张量核Tensor Core

- 每个TC在**一个时钟周期内**完成2个4×4矩阵相乘并累加一个4×4矩阵的计算

$$D = A \times B + C$$

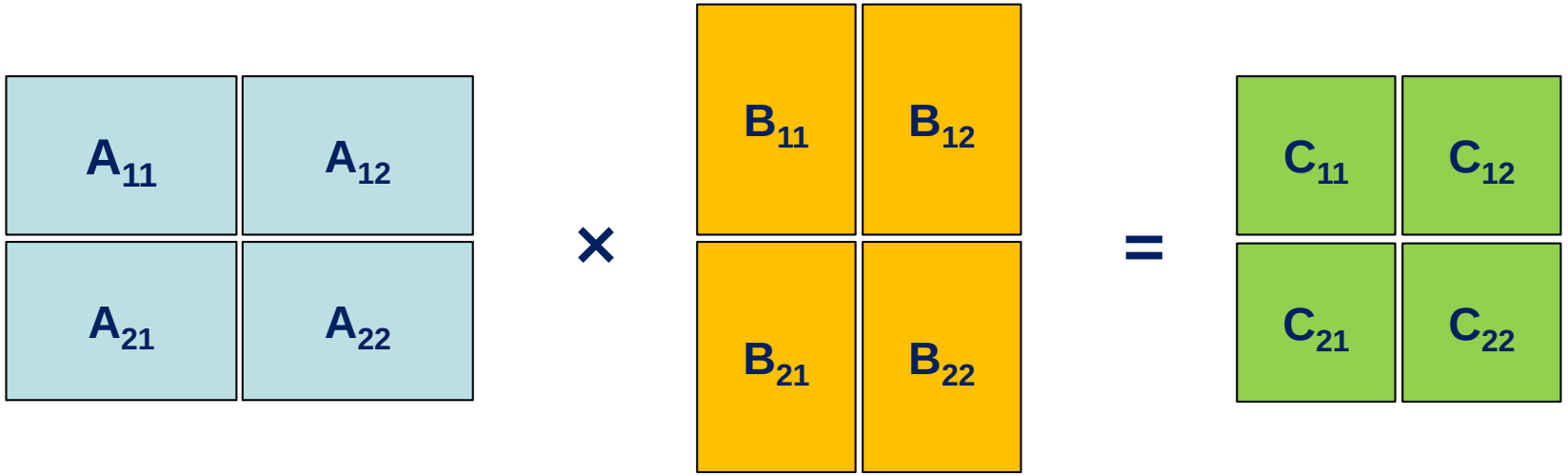


- 上述计算需进行 4×4×4 = 64 次乘加计算
- 每个流处理器中的8个TC可以独立并行进行上述计算
- RTX2080Ti中的72个流处理器中的TC可以完成72*8*64=36864次乘加

3.6.1 GPU网络计算加速原理

口 张量核Tensor Core

- 对于更大的矩阵，先将大矩阵切分成小矩阵，并分配在多个TC中分别独立计算，再累加合并得到大矩阵计算的结果



$$C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21} \qquad C_{12} = A_{11} \times B_{12} + A_{12} \times B_{22}$$

$$C_{21} = A_{21} \times B_{11} + A_{22} \times B_{21} \qquad C_{22} = A_{21} \times B_{12} + A_{22} \times B_{22}$$

3.6.1 GPU网络计算加速原理

神经网络加速计算基本原理

- 利用神经网络计算的**数据独立性**和**可并行性**
- GPU加速关键方式：**并行化**、**矢量化**，通用矩阵相乘**GEMM**
- 矩阵相乘，采用单指令多线程(SIMT)处理模式
- 矩阵乘法实现卷积计算

3.6.1 GPU网络计算加速原理

口 计算示例

F
特征图像

通道1		
F_{00}	F_{01}	F_{02}
F_{03}	F_{04}	F_{05}
F_{06}	F_{07}	F_{08}

通道2		
F_{10}	F_{11}	F_{12}
F_{13}	F_{14}	F_{15}
F_{16}	F_{17}	F_{18}

W
卷积核1

W_{00}	W_{01}	W_{10}	W_{11}
W_{02}	W_{03}	W_{12}	W_{13}

G
卷积核2

G_{00}	G_{01}	G_{10}	G_{11}
G_{02}	G_{03}	G_{12}	G_{13}

M
卷积核3

M_{00}	M_{01}	M_{10}	M_{11}
M_{02}	M_{03}	M_{12}	M_{13}

N
卷积核4

N_{00}	N_{01}	N_{10}	N_{11}
N_{02}	N_{03}	N_{12}	N_{13}

3.6.1 GPU网络计算加速原理

口 计算示例

W_{00}	W_{01}	W_{02}	W_{03}	W_{10}	W_{11}	W_{12}	W_{13}
G_{00}	G_{01}	G_{02}	G_{03}	G_{10}	G_{11}	G_{12}	G_{13}
M_{00}	M_{01}	M_{02}	M_{03}	M_{10}	M_{11}	M_{12}	M_{13}
N_{00}	N_{01}	N_{02}	N_{03}	N_{10}	N_{11}	N_{12}	N_{13}

×

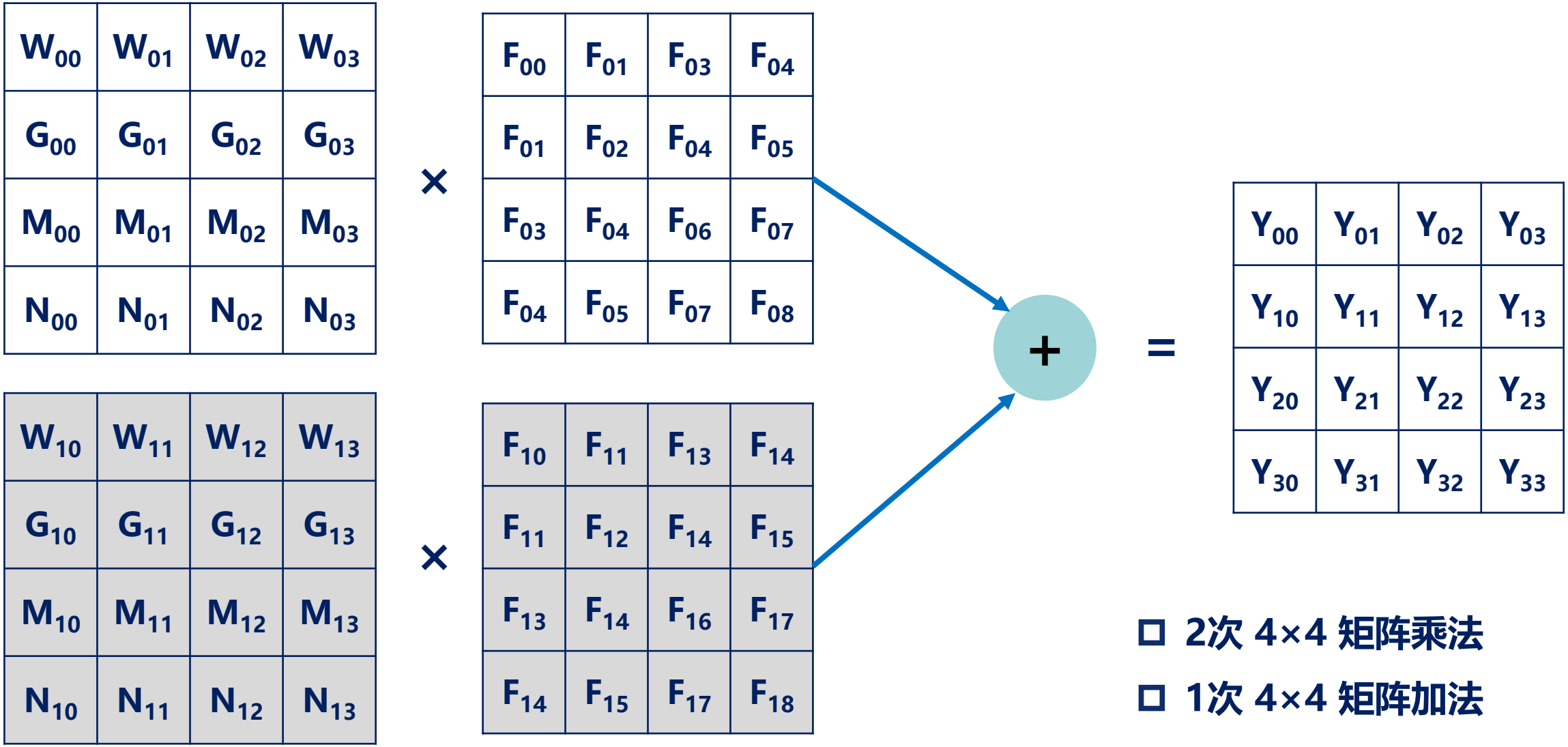
F_{00}	F_{01}	F_{03}	F_{04}
F_{01}	F_{02}	F_{04}	F_{05}
F_{03}	F_{04}	F_{06}	F_{07}
F_{04}	F_{05}	F_{07}	F_{08}
F_{10}	F_{11}	F_{13}	F_{14}
F_{11}	F_{12}	F_{14}	F_{15}
F_{13}	F_{14}	F_{16}	F_{17}
F_{14}	F_{15}	F_{17}	F_{18}

=

Y_{00}	Y_{01}	Y_{02}	Y_{03}
Y_{10}	Y_{11}	Y_{12}	Y_{13}
Y_{20}	Y_{21}	Y_{22}	Y_{23}
Y_{30}	Y_{31}	Y_{32}	Y_{33}

3.6.1 GPU网络计算加速原理

口 计算示例



- 口 2次 4×4 矩阵乘法
- 口 1次 4×4 矩阵加法

3.6.1 GPU网络计算加速原理

GPU加速原理总结

- 卷积转换为矩阵乘加运算GEMM
- 优化的专用张量处理单元阵列进行并行加速计算
- 通过专用指令控制张量单元的矩阵加速

大规模张量单元并行化 + 卷积GEMM矢量化 → 加速网络卷积计算

3.6.2 TPU网络计算加速原理

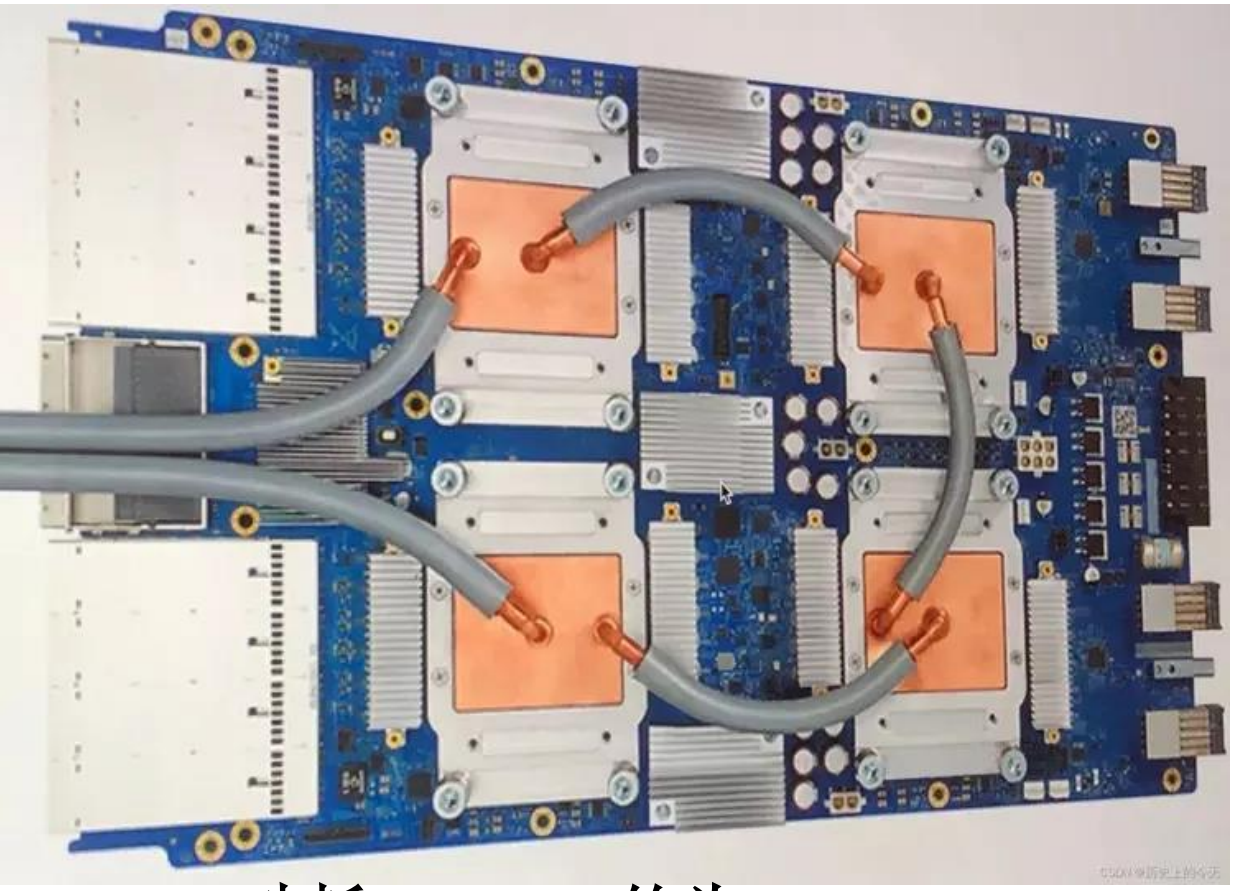
□ TPU的发展历程

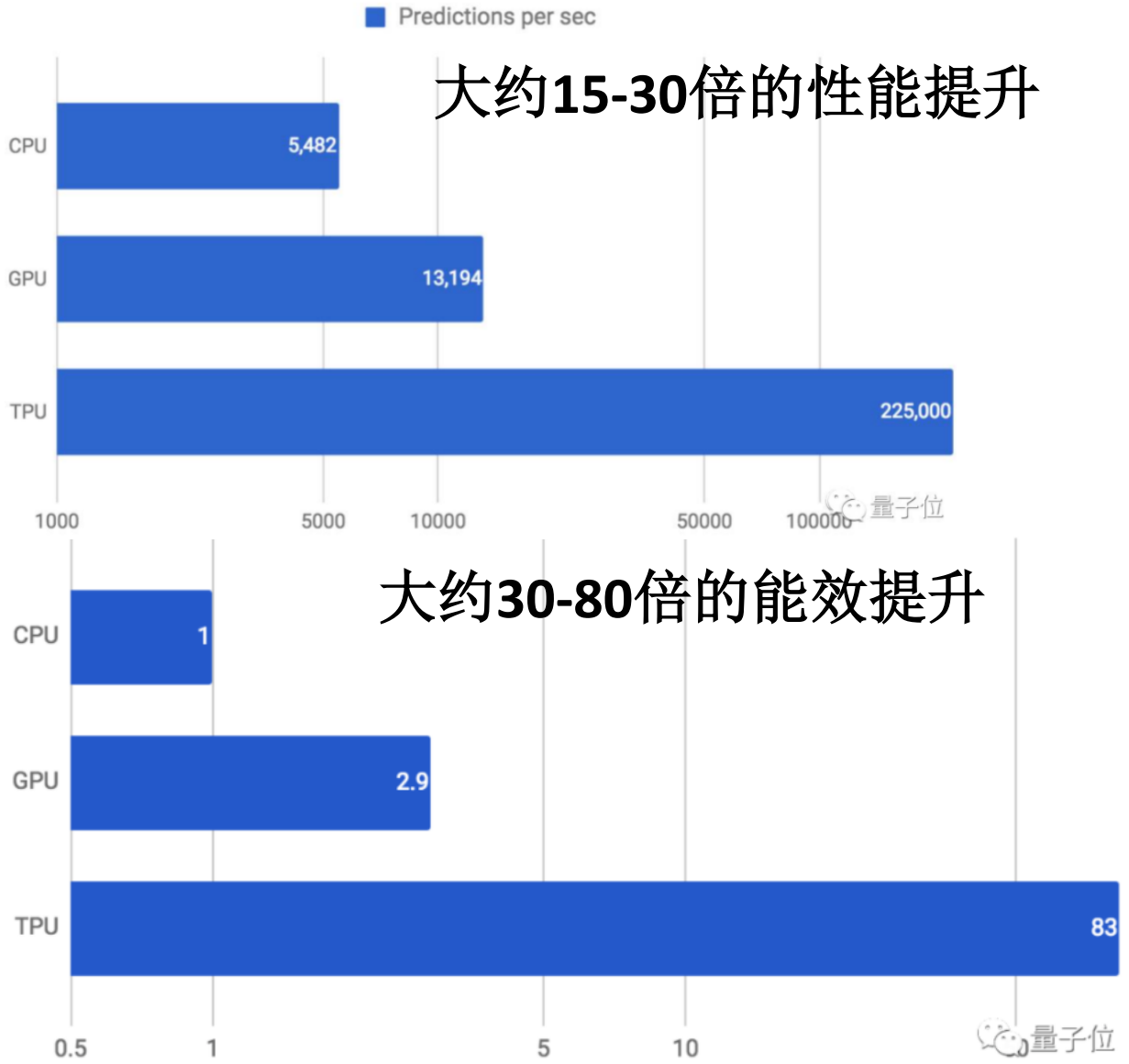
名称	时间	性能	应用
TPUv1	2016年	92Tops	数据中心推理应用
TPUv2	2017年	180TFlops（集成4块芯片），64GB（HBM）	数据中心训练和推理，引入到Google Cloud中
TPUv3	2018年	420TFlops，128GB（HBM）	数据中心的训练和推理（进一步扩展）
Edge TPU	2018年	可处理高吞吐量的流式数据	IoT设备
TPUv2 pod	2019年	11.5千万亿次浮点运算/s，4TB（HBM），二维环面网状网络	深度学习领域
TPUv3 pod	2019年	>100千万亿次浮点运算/s，32TB（HBM），二维环面网状网络	深度学习领域

3.6.2 TPU网络计算加速原理

□ TPU（v1）实物

它与同期的CPU和GPU相比（英特尔至强E5-2699 v3与Tesla K80 GPU），





大约15-30倍的性能提升

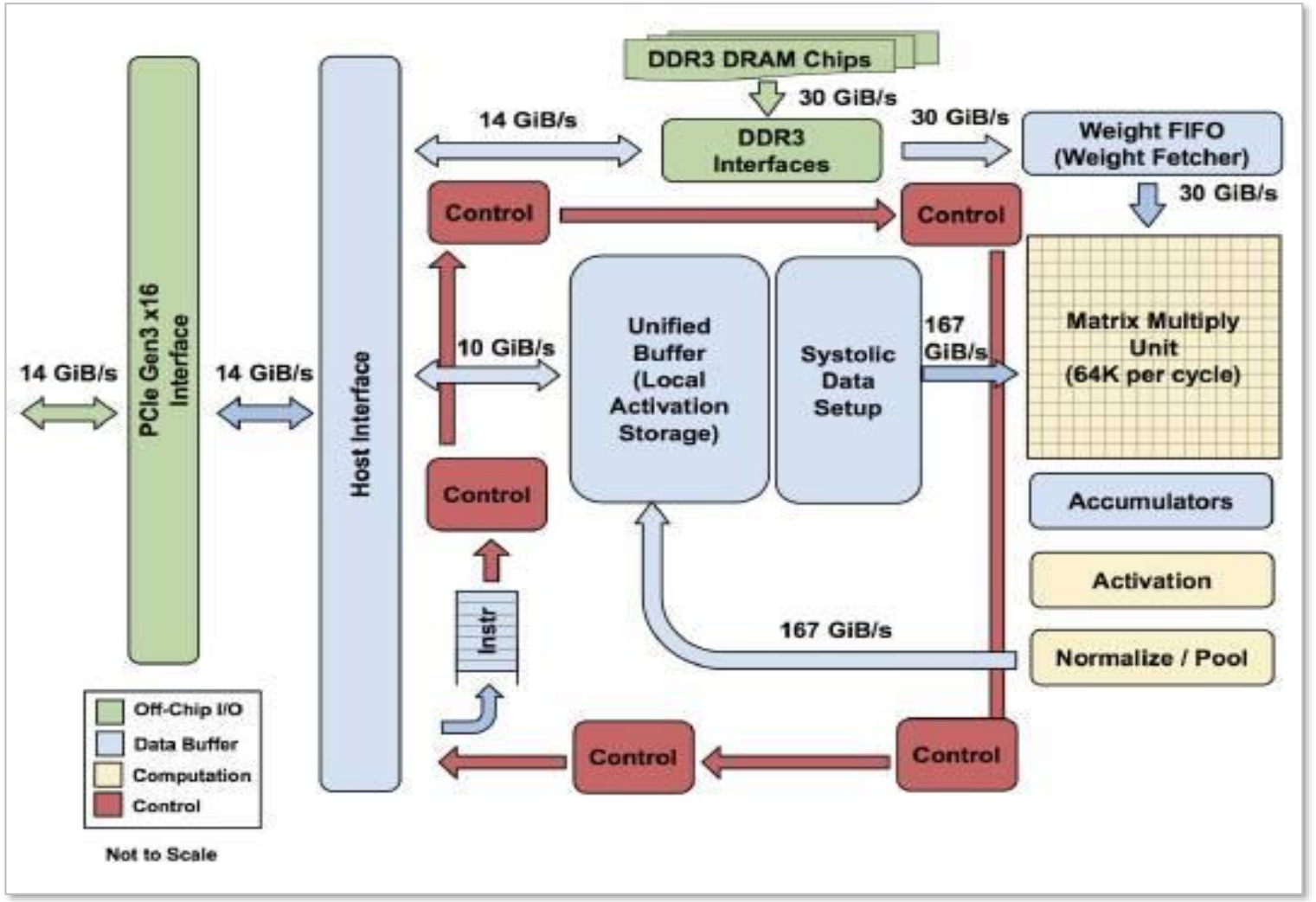
大约30-80倍的能效提升

功耗：TPU v1约为40W，TPU v4-175W，而同时期的A100已达400W

3.6.2 TPU网络计算加速原理

TPU（v1）架构

- 脉动阵列矩阵乘单元
- 矢量计算单元
- 主接口模块
- 控制队列模块
- 统一缓冲区
- DMA控制模块
- DDR3控制器 8GB

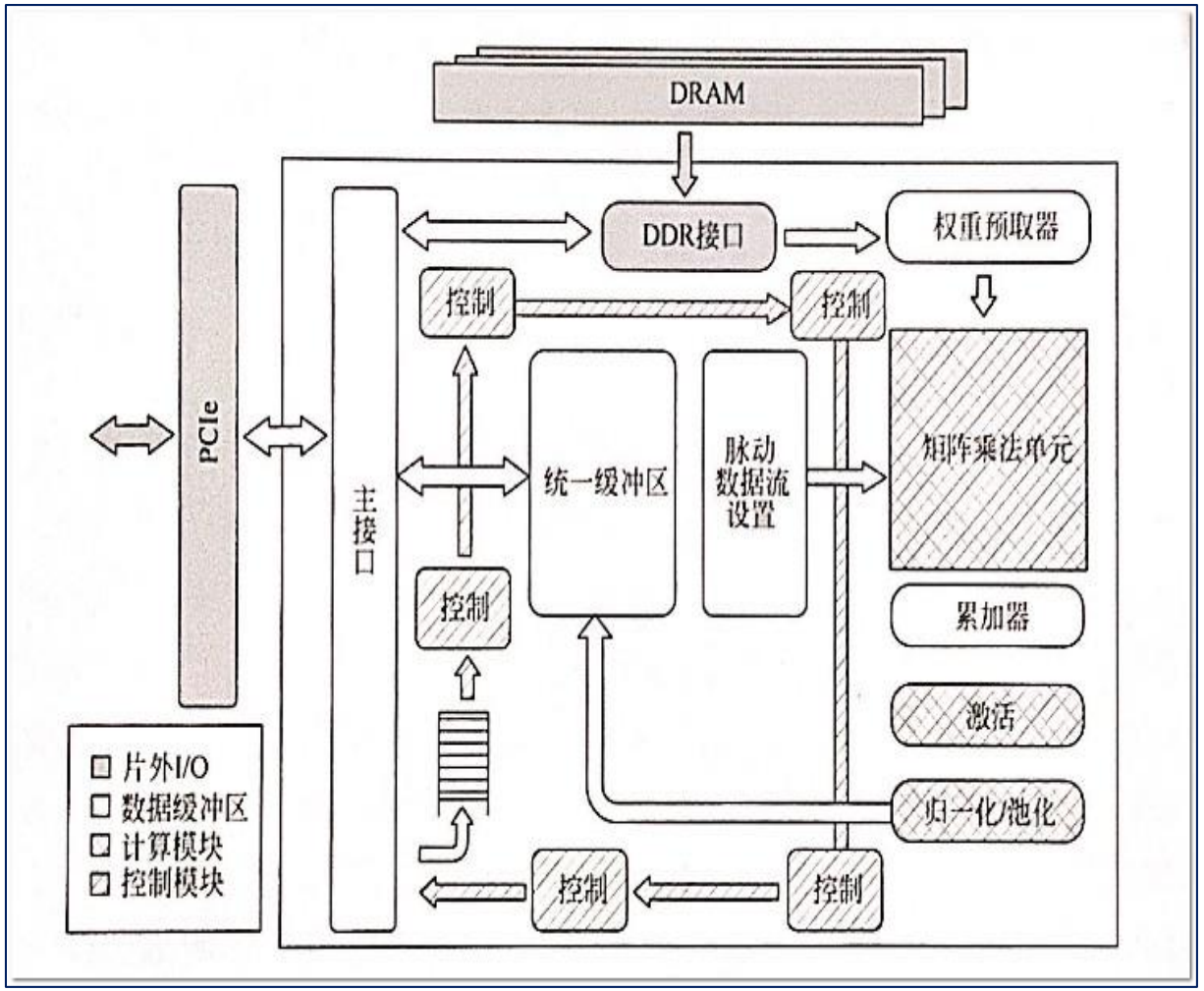


ISCA2017 In-Datacenter Performance Analysis of a Tensor Processor Unit

3.2 谷歌TPU架构

- 脉动阵列/矢量计算单元/主接口模块/队列模块/统一缓冲区/DMA控制模块

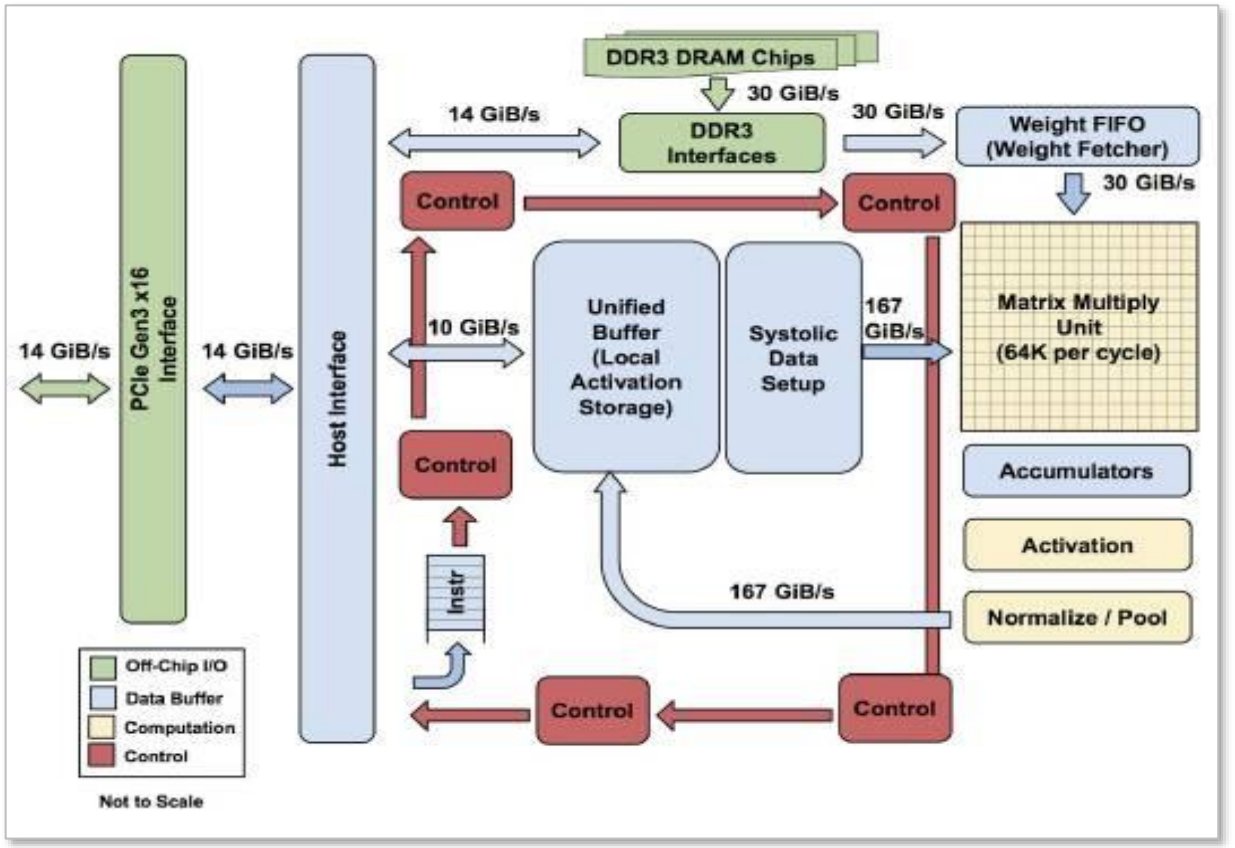
- 主接口：获取参数和配置，发送指令队列
- DMA控制模块：读取IFM和权重数据至片上缓存中
- 控制队列模块：
  - 控制计算过程与方式，如IFM和权重数据如何进入、如何分块累加
  - 发送控制信号至其它模块
- 统一缓存区：
  - 存储输入/中间/输出结果
  - 中间结果发送至脉动阵列计算



3.6.2 TPU网络计算加速原理

□ TPU（v1）架构

- Matrix Multiply unit(MXU)：矩阵乘积计算单元
- Accumulators 结果累加单元 4MB
- activation 硬件神经元激活函数
  sigmoid(s形函数), tanh(双曲正切函数)
- Unified Buffer: 24MB SRAM 存储器
- Sytolic data setup : 脉动数据生成器
- Normalize/Pool 结果归一化单元



ISCA2017 In-Datacenter Performance Analysis of a Tensor Processor Unit

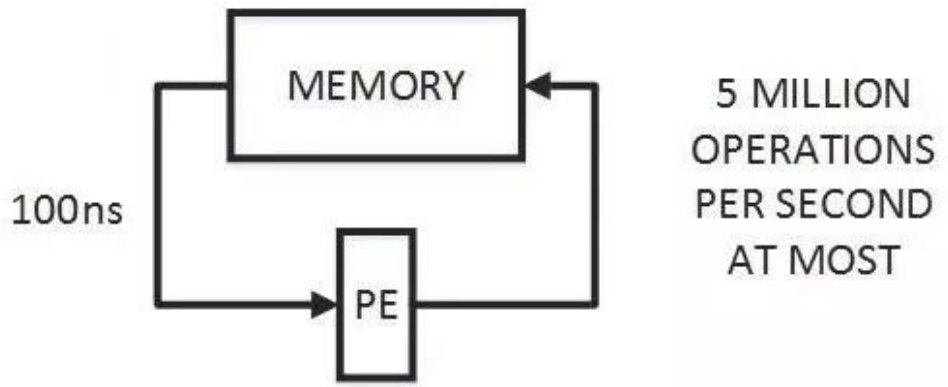
3.6.2 TPU网络计算加速原理

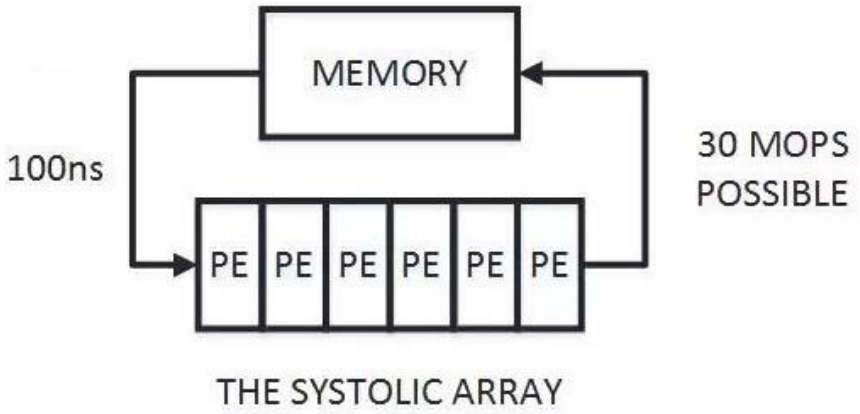
- 脉动阵列 systolic array
  - Why systolic architectures?
    - Simple and regular design
    - Concurrency and communication
    - Balancing computation with I/O
  - 脉动阵列的特征
    - 由多个同构的PE构成，可以是一维或二维，串行、阵列或树的结构
    - PE功能相对简单，通过大量PE并行来提高运算效率
    - PE只能向相邻的PE发送数据，数据向“下游”流动，直到流出最末PE
  - 脉动架构灵活性、扩展性较差，被发明之后并没有得到广泛应用
  - DNN大量使用卷积运算和矩阵运算，这正好也是是脉动架构的优势

3.6.2 TPU网络计算加速原理

□ 脉动阵列 systolic array

- 让数据尽量在处理单元中多流动一会儿
- 多次重用了输入数据，消耗较小的访存带宽情况下实现较高的运算吞吐率

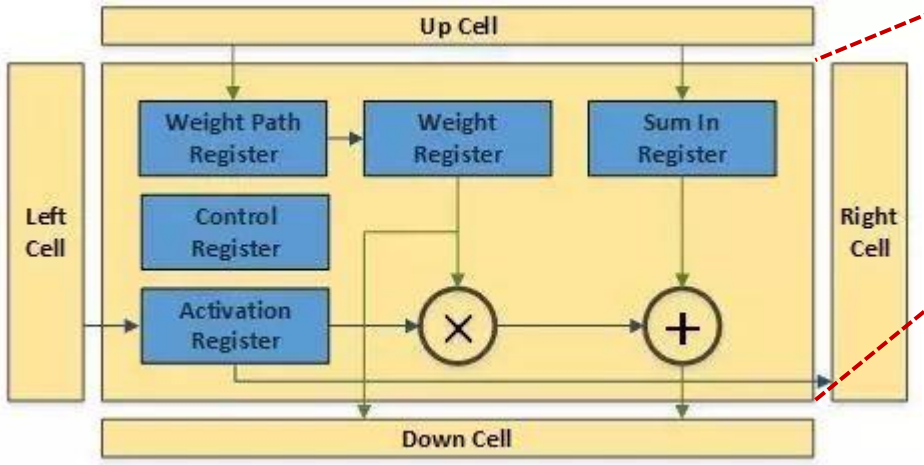


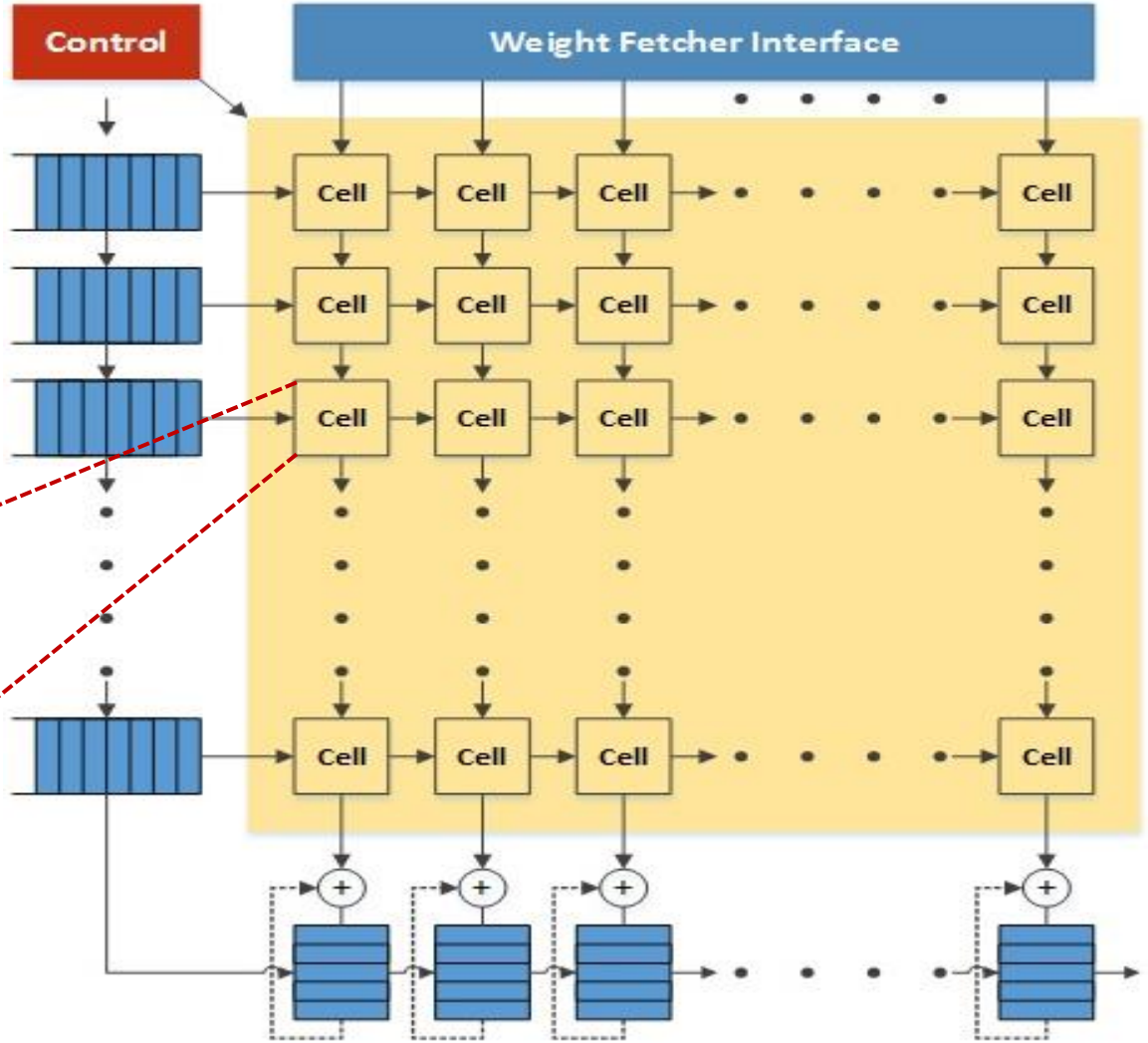


3.6.2 TPU网络计算加速原理

□ 脉动阵列 systolic array

- 让数据尽量在处理单元中多流动
- 多次重用了输入数据，消耗较小的访存带宽情况下实现较高的运算吞吐率

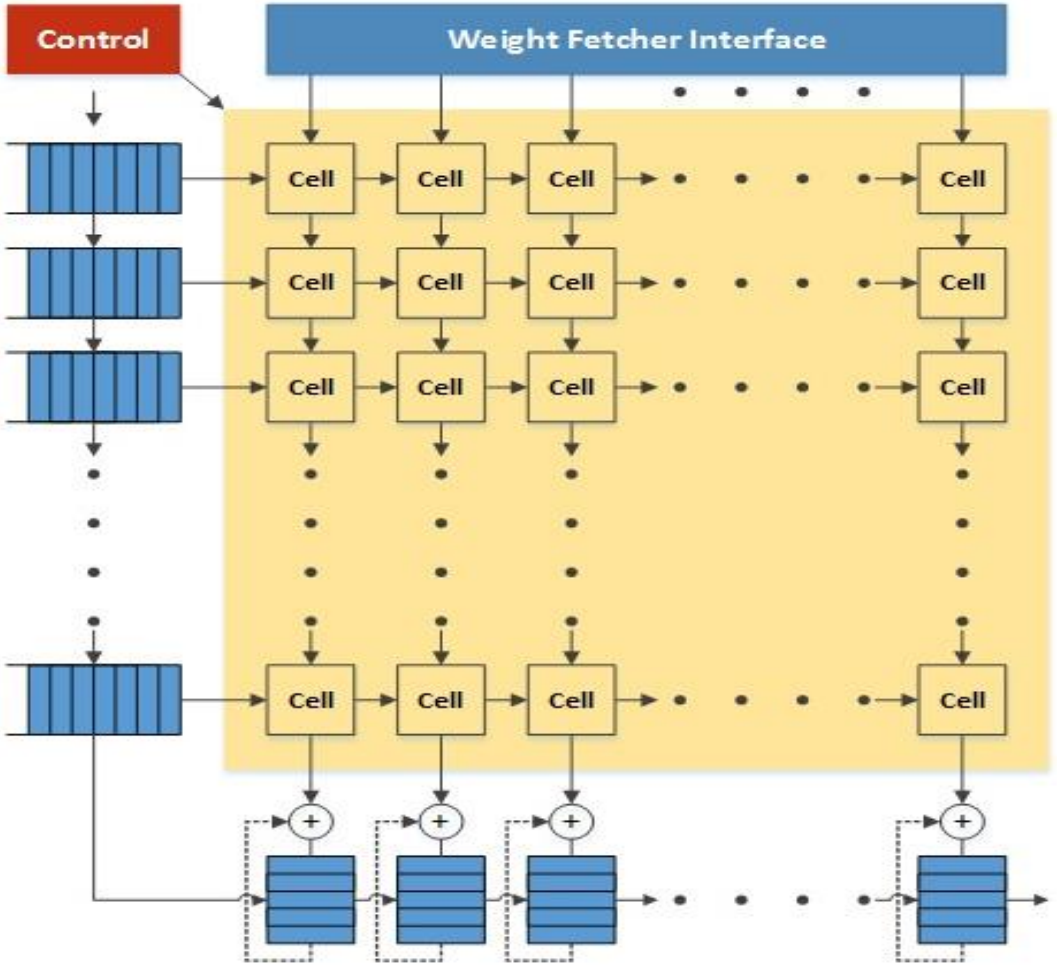




3.6.2 TPU网络计算加速原理

□ 脉动阵列 systolic array

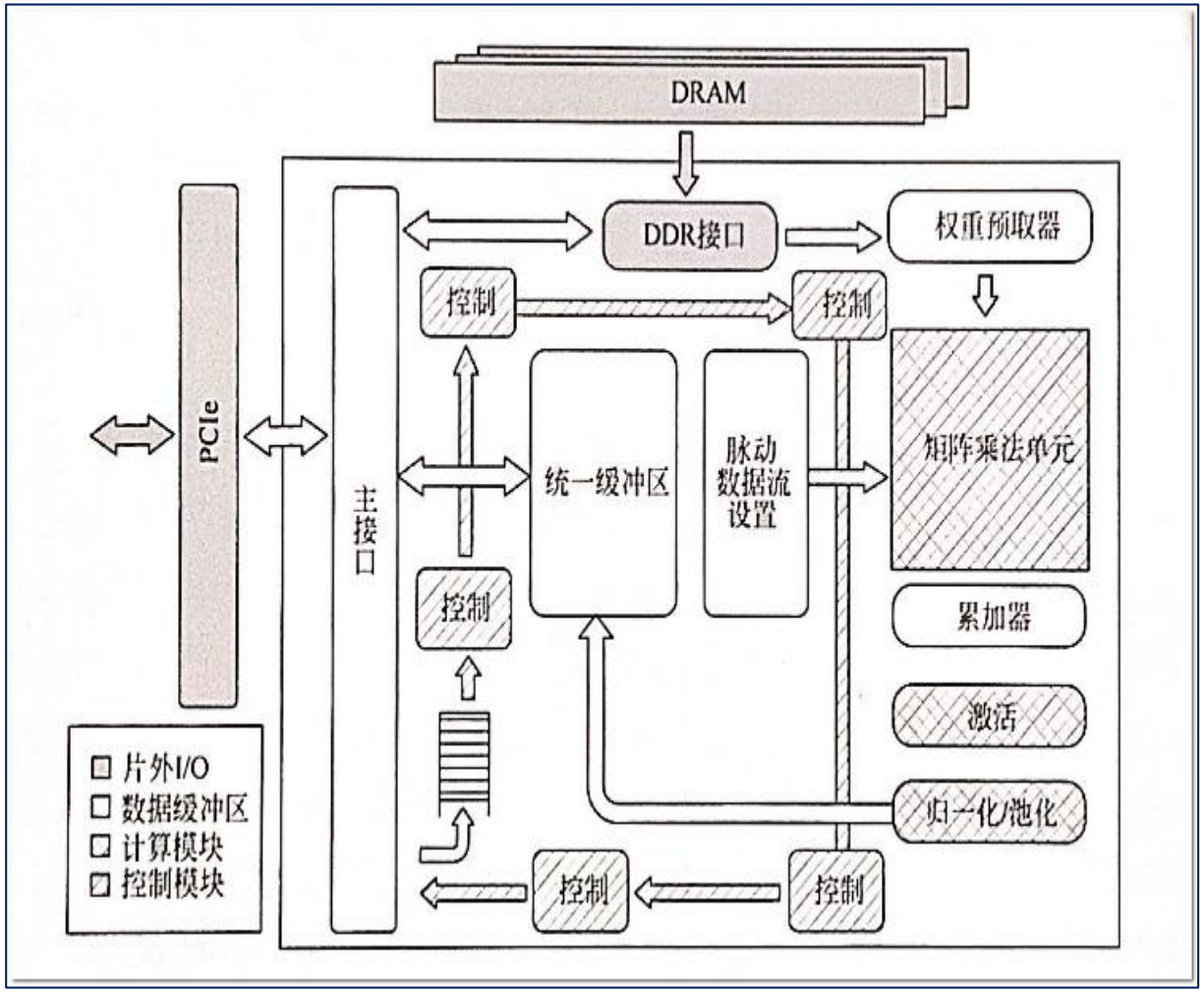
- 256x256个MACS
- 8位有符号/无符号整型MAC能力
- 算力峰值65536×700MHZ = 65536 × 700 000 000 × 2 ≈ 92TOPS
- 每个周期产生的结果暂存到大小为4M的Accumulators(累加器)中



3.6.2 TPU网络计算加速原理

□ TPU网络计算流程

- 按照每层网络顺序，在简单的指令和硬件状态机的统一控制下执行
- 从片外获取IFM，从DDR中获取权重数据，送入脉动阵列完成卷积计算
- 每个周期产生的结果暂存到累加器中
- 矢量单元进行非线性激活/池化操作
- 每层输出结果暂存到统一缓存区中，准备作为下一层的输入



3.6.2 TPU网络计算加速原理

TPU指令集

- 属于CISC指令集，平均每个指令的时钟周期是10~20，指令通过PCIe总线进入
- 多数是宏指令，实质是硬件控制状态机，大幅降低指令译码和存储的开销
- 神经计算的指令有5条
  - 数据读指令 Read_Host_Memory
  - 权重读指令 Read_Weight
  - 矩阵运算指令 MatrixMultiply/Convolve, Activate
  - 数据写回指令 Write_Host_Memory
- 指令格式占12位：统一缓冲区地址3位、累加器缓冲区地址2位，操作数长度4位，操作码/标志3位

3.2 TPU神经网络计算指令

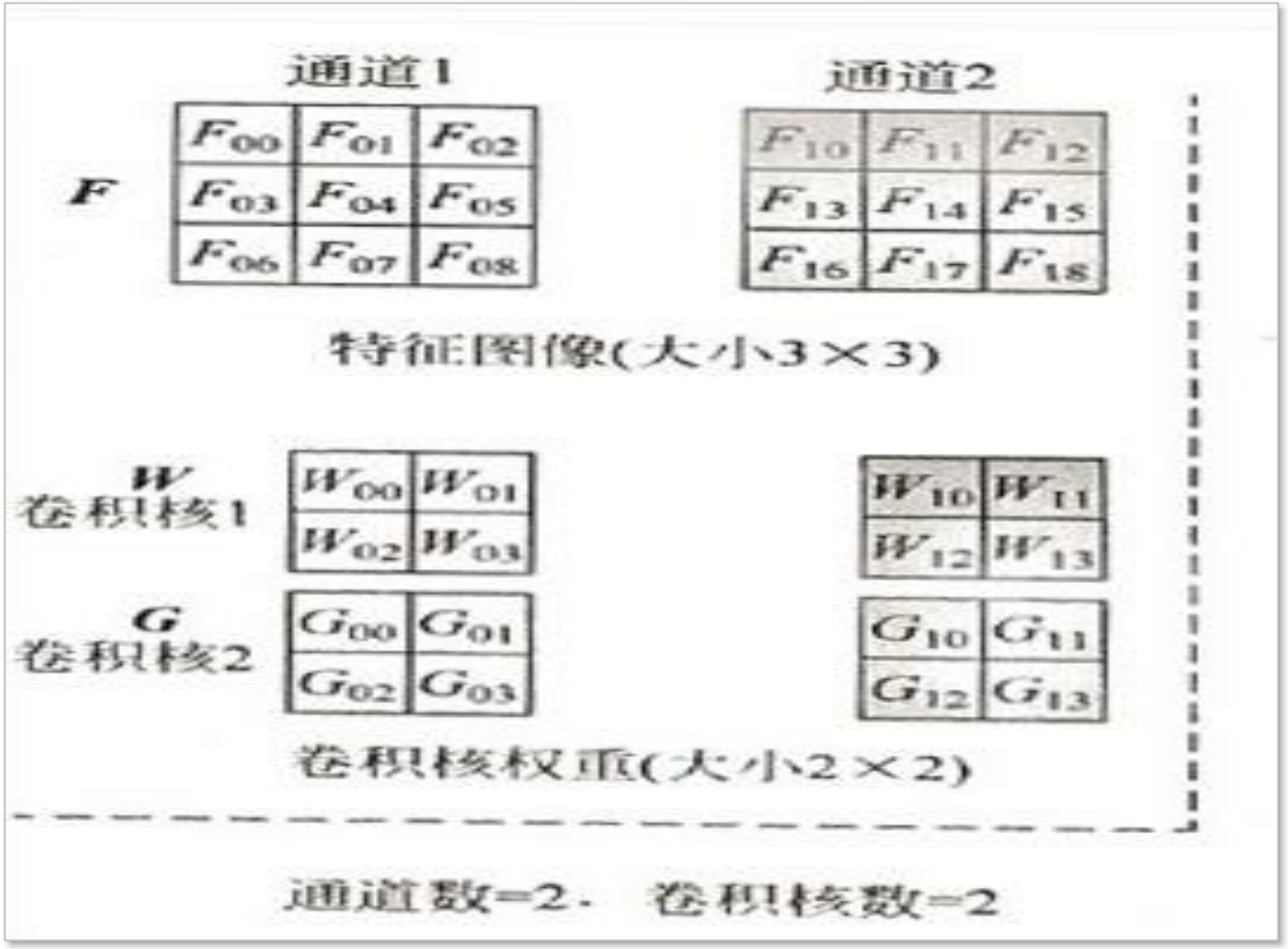
TPU指令执行流程

- Read_Host_Memory将IFM数据从主存读取到统一缓冲区
- Read_Weight指令从内存中提取卷积核权重并固定在脉动阵列中
- MatrixMultiply/Convolve指令将统一缓冲区中的输入数据送人脉动阵列运算后再载入到累加器中累加
- Activate指令运行激活函数、池化操作运算再将结果存入统一缓冲区
- Write_Host_Memory指令将统一缓冲区的最终运算结果写回系统内存

TPU 指令	功能
Read_Host_Memory	从内存中读取数据
Read_Weights	从内存中读取权值
MatrixMultiply/Convolve	数据和权重相乘或卷积，累加结果
Activate	应用激活函数
Write_host_Memory	将结果写入内

3.6.2 TPU网络计算加速原理

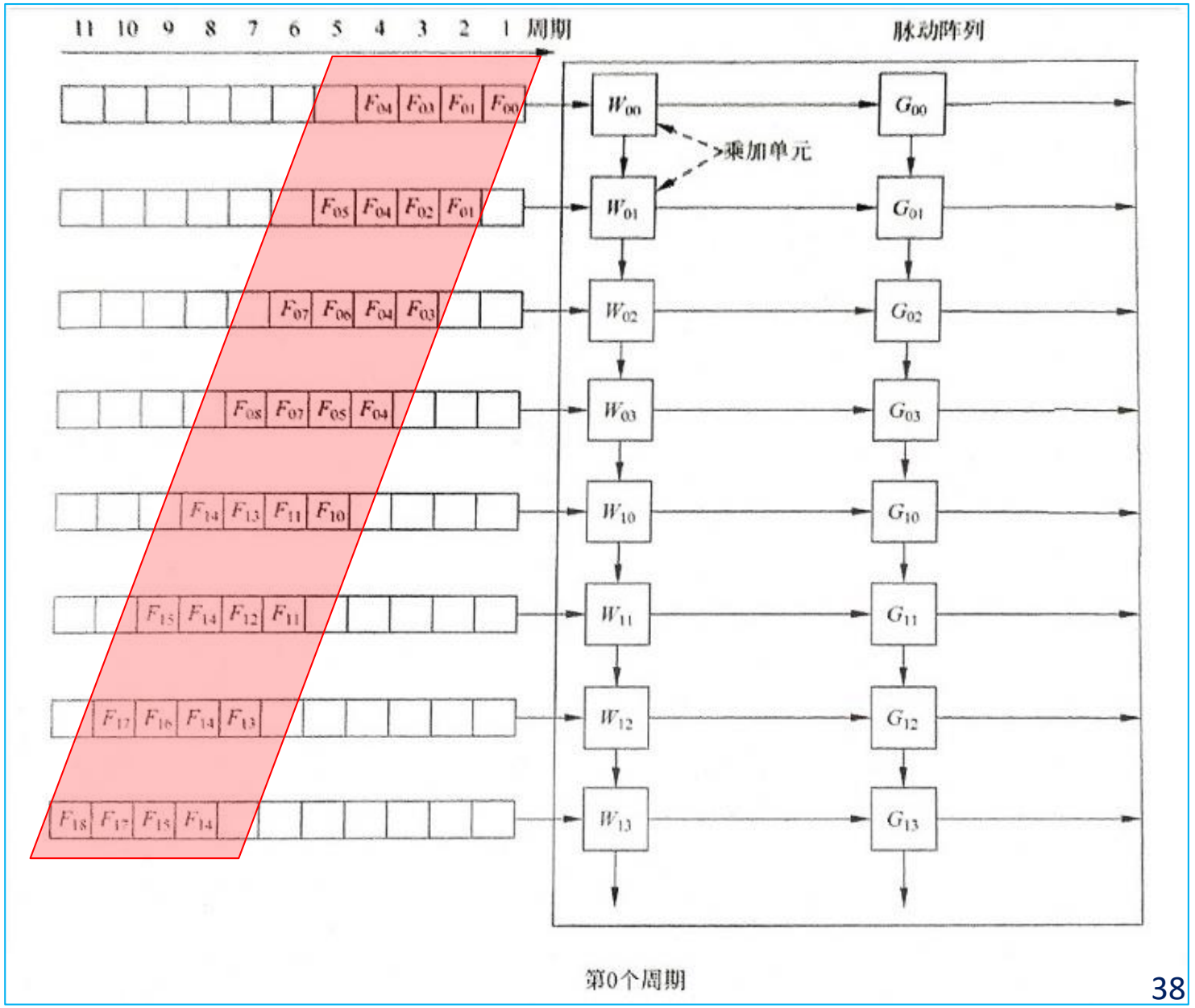
□ 计算示例



3.6.2 TPU网络计算加速-脉动阵列技术

- 脉动阵列的主体是二维滑动阵列，每个节点是一个脉动乘加计算单元
- 行列之间通过横向、纵向数据通路实现数据向右、向下滑动传递

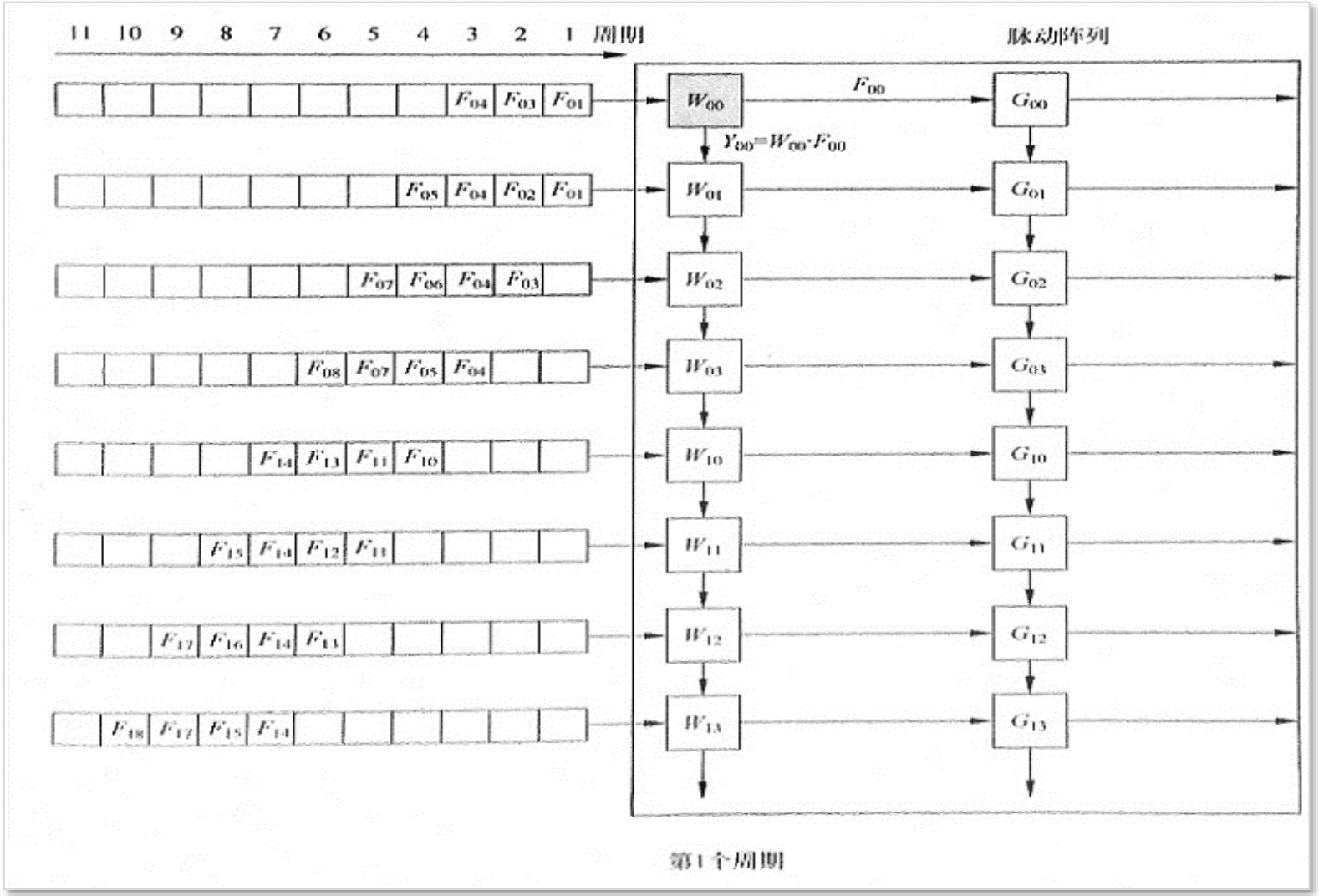
- 固定卷积核权重
- 横向脉动输入数据
- 纵向脉动MAC计算



3.6.2 TPU网络计算加速-脉动阵列技术

脉动阵列第1个周期计算状态

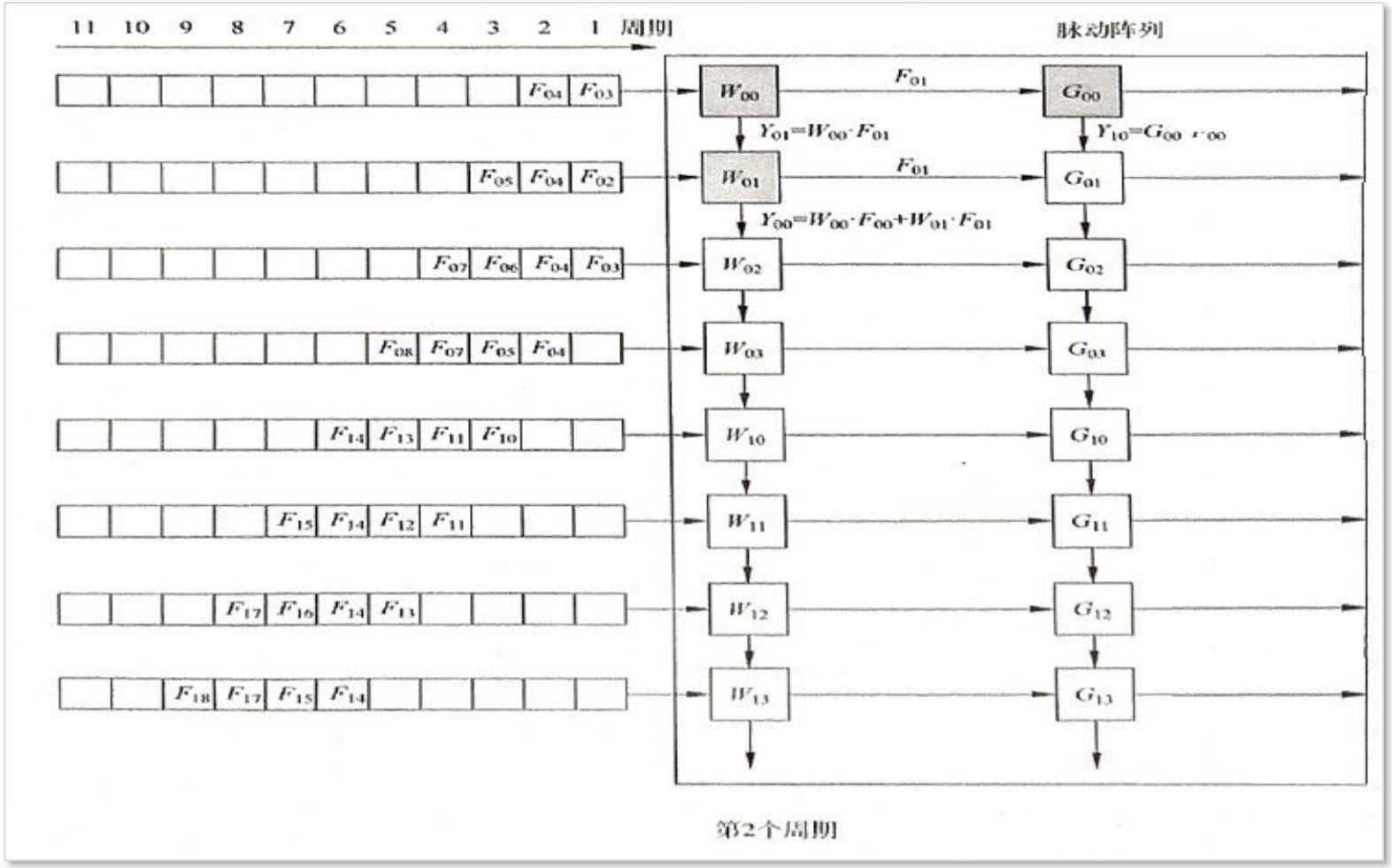
- F_{00}进入W_{00}乘加单元与W_{00}权重乘加产生Y_{00}的部分和
- F_{00}转移到G_{00}乘加单元入口



3.6.2 TPU网络计算加速-脉动阵列技术

脉动阵列第2个周期计算状态

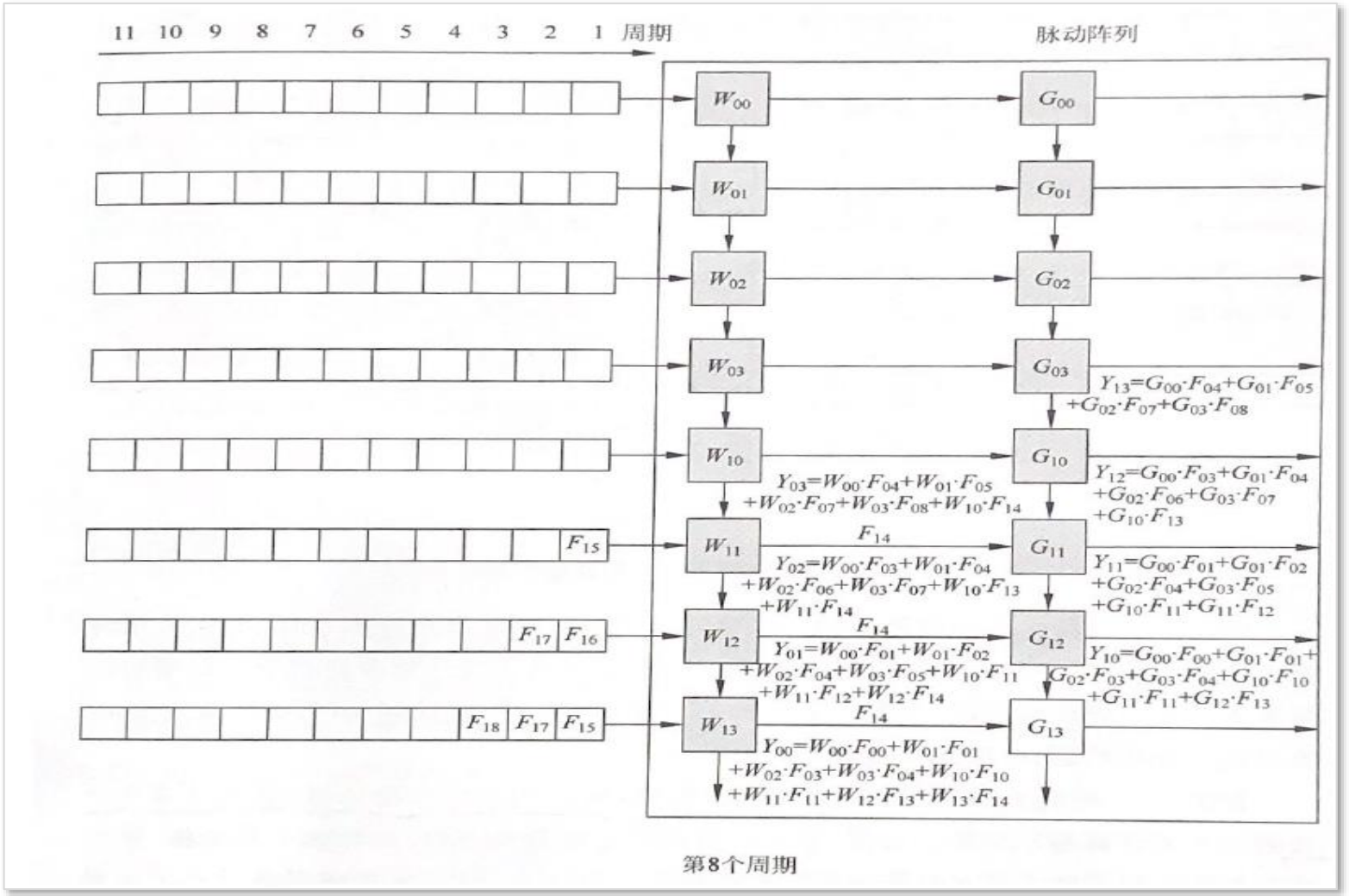
- 第2行输入F_{01}与W_{01}乘加产生的部分和与第一行传递下来的Y_{00}的部分和相加
- F_{01}转移到G_{01}乘加单元入口
- F_{01}与G_{00}乘加产生Y_{10}的部分和



3.6.2 TPU网络计算加速-脉动阵列技术

脉动阵列第8个周期计算状态

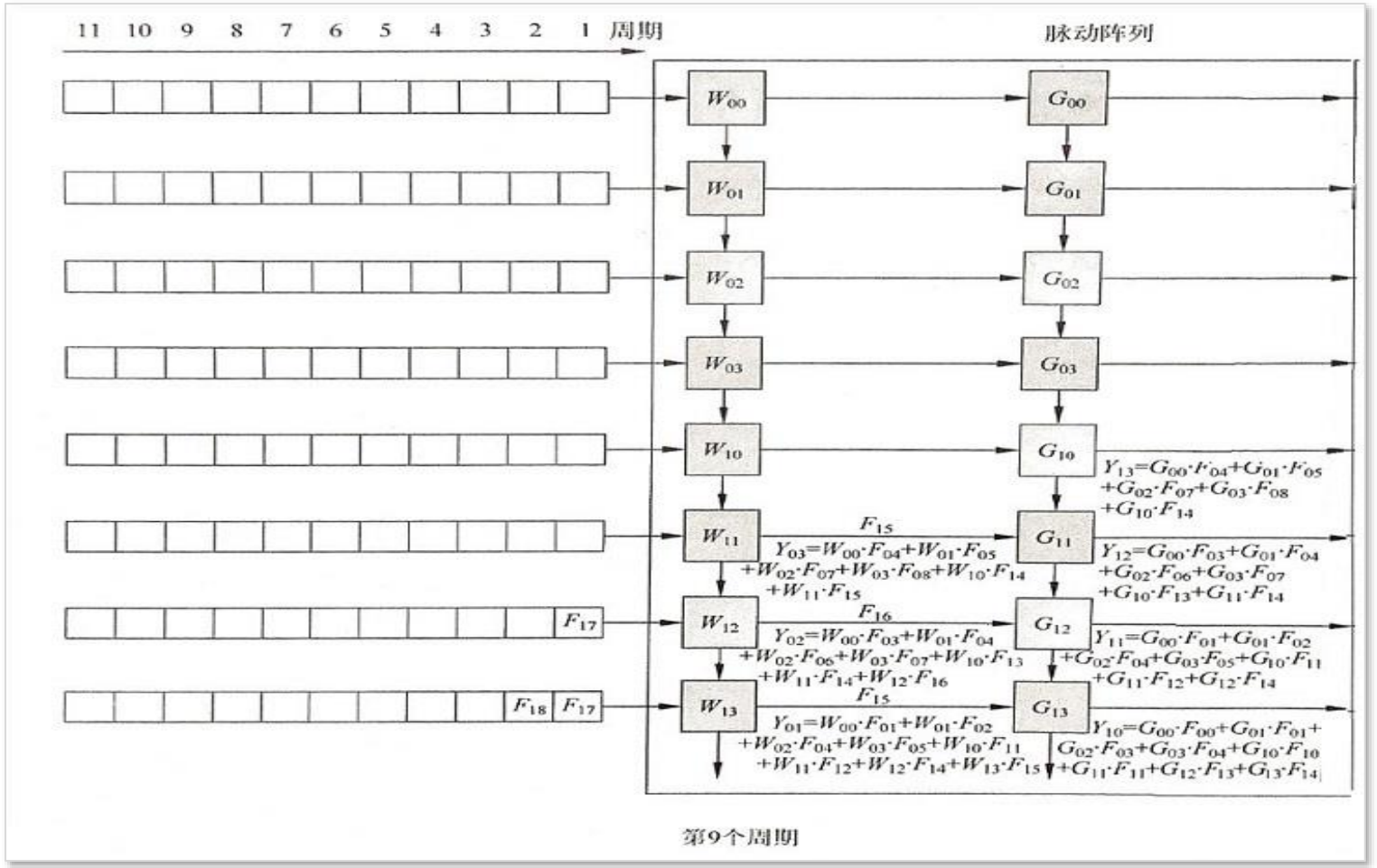
- 得到第一个卷积结果Y_{00}



3.6.2 TPU网络计算加速-脉动阵列技术

脉动阵列第9个周期计算状态

- 得到两个卷积结果，此后每次得到两个卷积结果
- 直到所有IFM数据都被卷积完毕



3.6.2 TPU网络计算加速-脉动阵列技术

- 脉动阵列数据必须事先排列好的顺序依次进入
- 填满阵列需要启动时间：行数+列数-1
- 卷积核过大或通道太多时，脉动阵列过于庞大，采用分割计算、末端累加
- 多通道权重分割为几个部分，依次对每个部分权重进行计算
- 结果临时存放在底部加法累加器中，换入另一组权重块后结果被累加
- 直到所有权重块都遍历完成，累加器累加所有结果输出最终值

3.6.2 TPU网络计算加速原理

TPU加速原理总结

- 片上高达24MB内存、4MB累加器缓存，减少片外访存
- 脉动阵列优化卷积运算，充分利用数据局部性，脉动传递数据，降低访存次数
- 优化的整体架构和数据提供方式，让计算单元满载运行，实现极高的吞吐率
- 运算过程采用多级流水线，通过重叠执行多条MAC指令来隐藏时延

大规模脉动阵列 + 大容量片上存储 → 加速网络卷积计算