

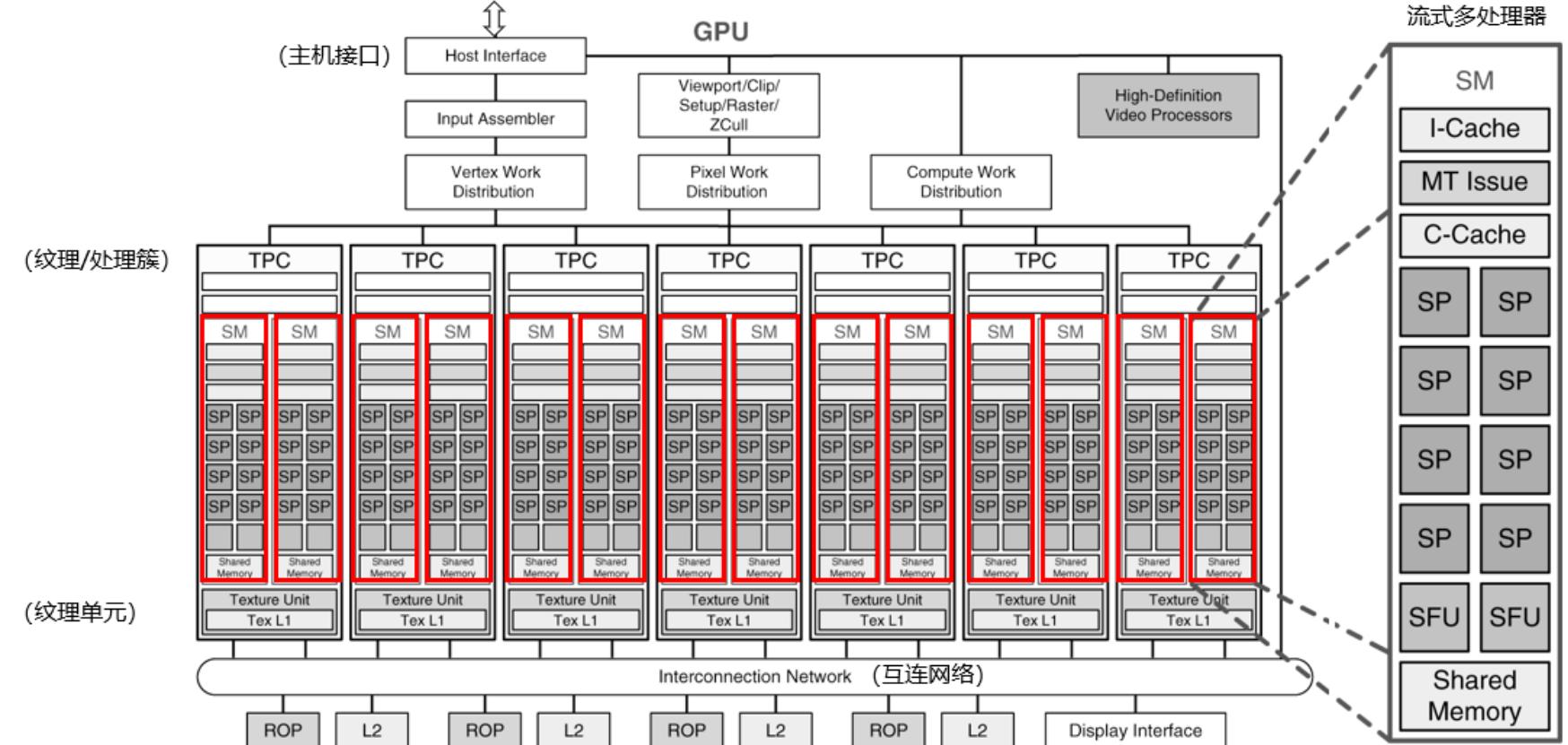
Lecture 04 GPU Architecture and Open-Source RISC-V GPU

GPU架构与开源RISC-V GPU

2025-10

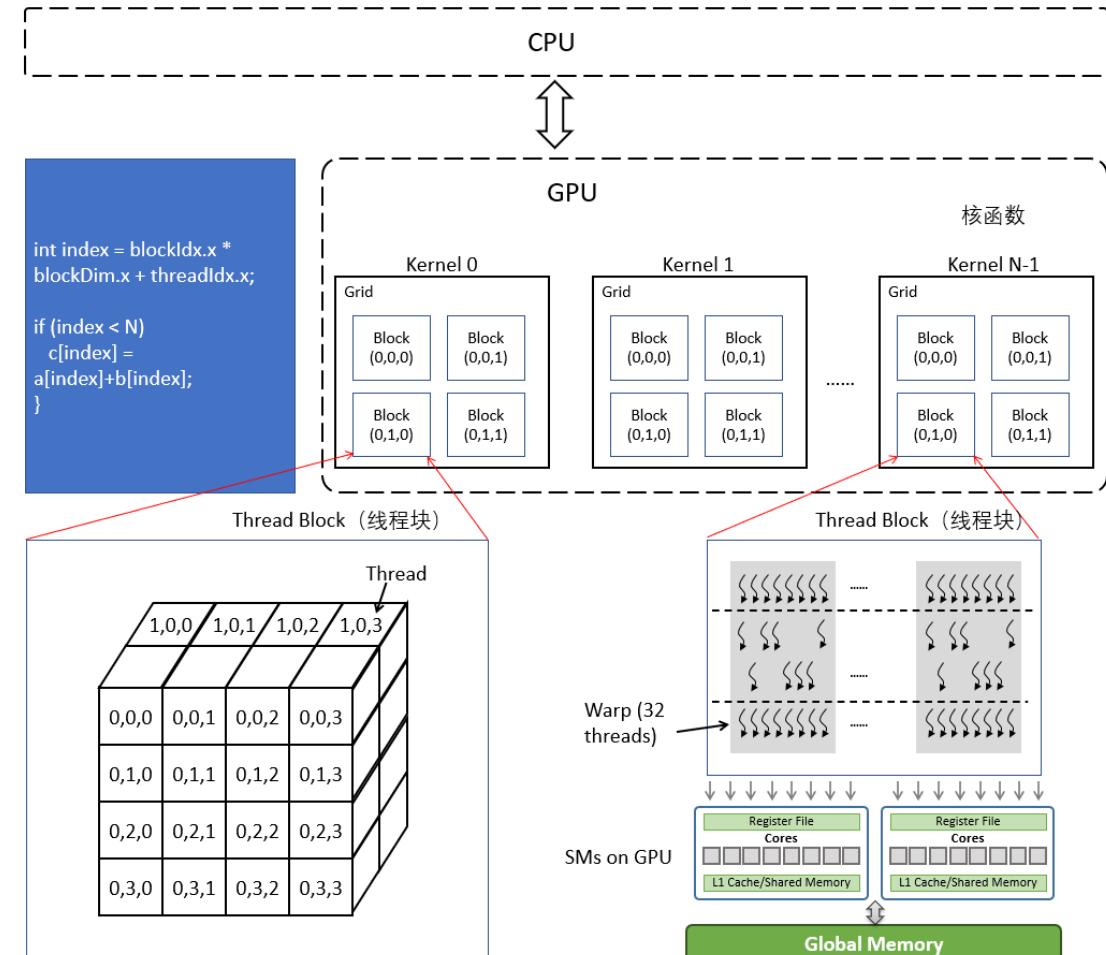
经典GPU架构

- **流式多处理器** (Stream Multi-processor, SM) 是构建整个 GPU 的核心模块 (执行整个 Kernel Grid)，一个流式多处理器上一般同时运行多个线程块
- 每个流式多处理器可以视为具有较小结构的CPU，支持**指令并行** (多发射)
- Register和shared memory是SM的稀缺资源。CUDA将这些资源分配给所有驻留在SM中的 threads
- 流式多处理器是线程块的运行载体，但一般不支持乱序执行。每个流式多处理器上的单个Warp以SIMD方式执行相同指令

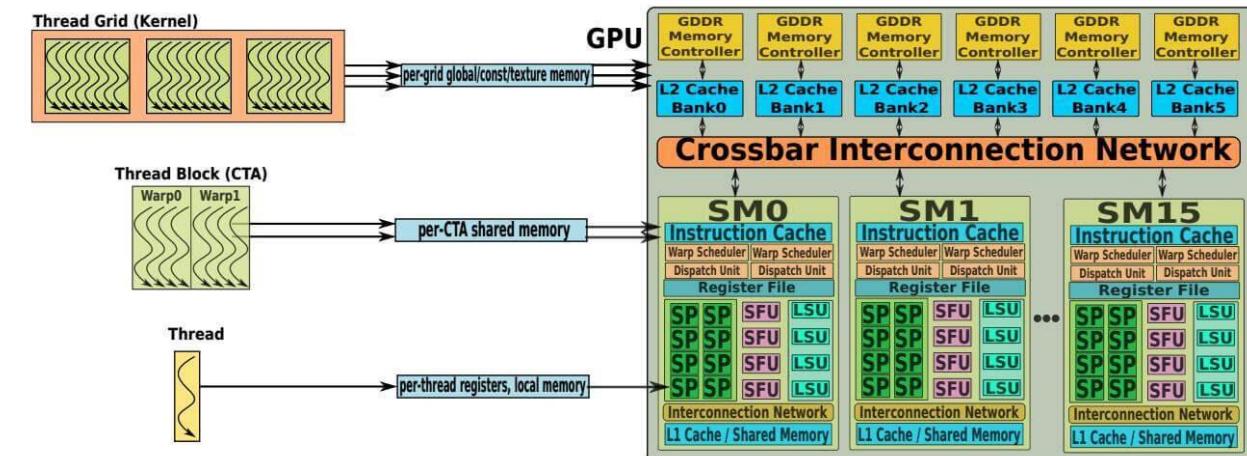
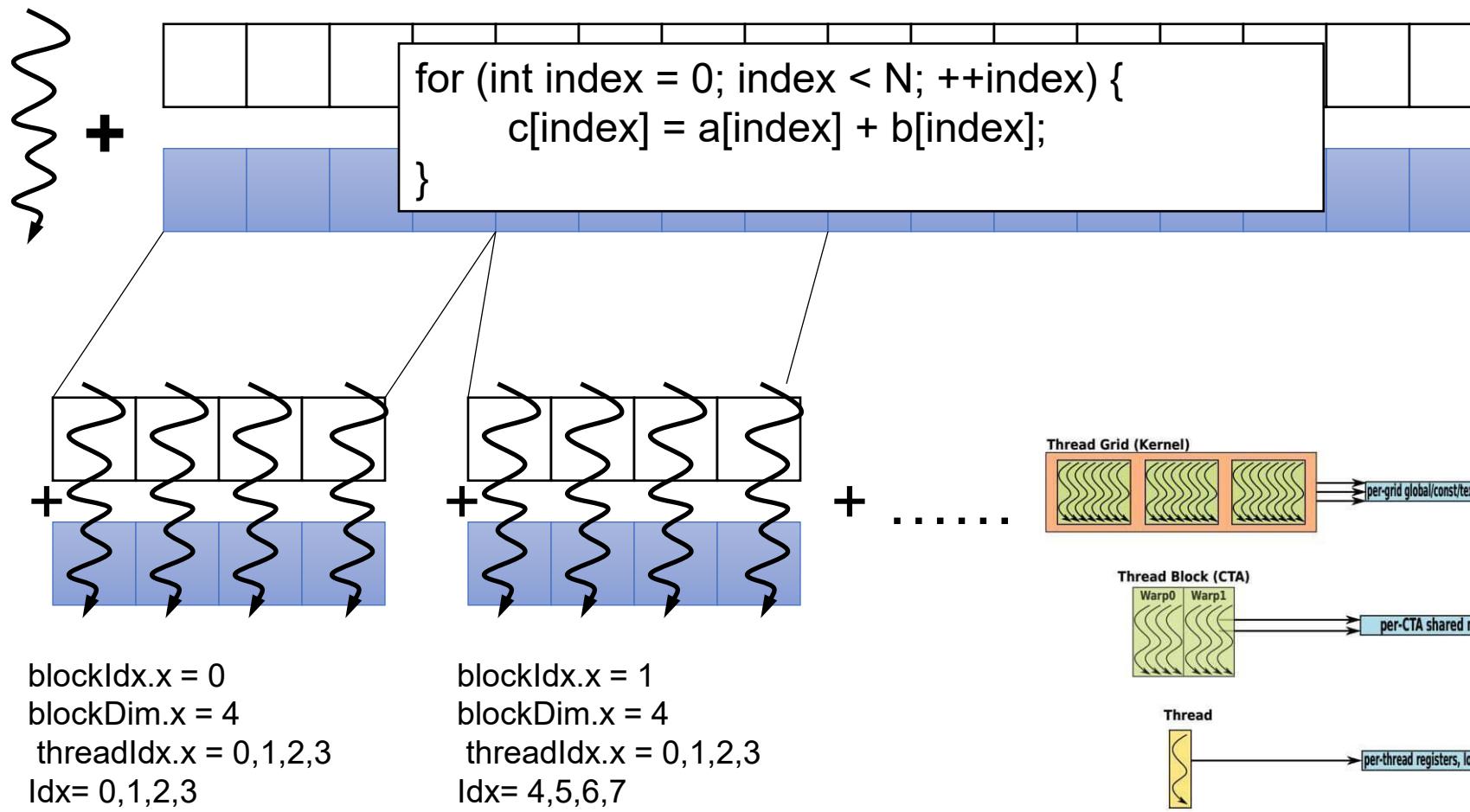


软件线程结构

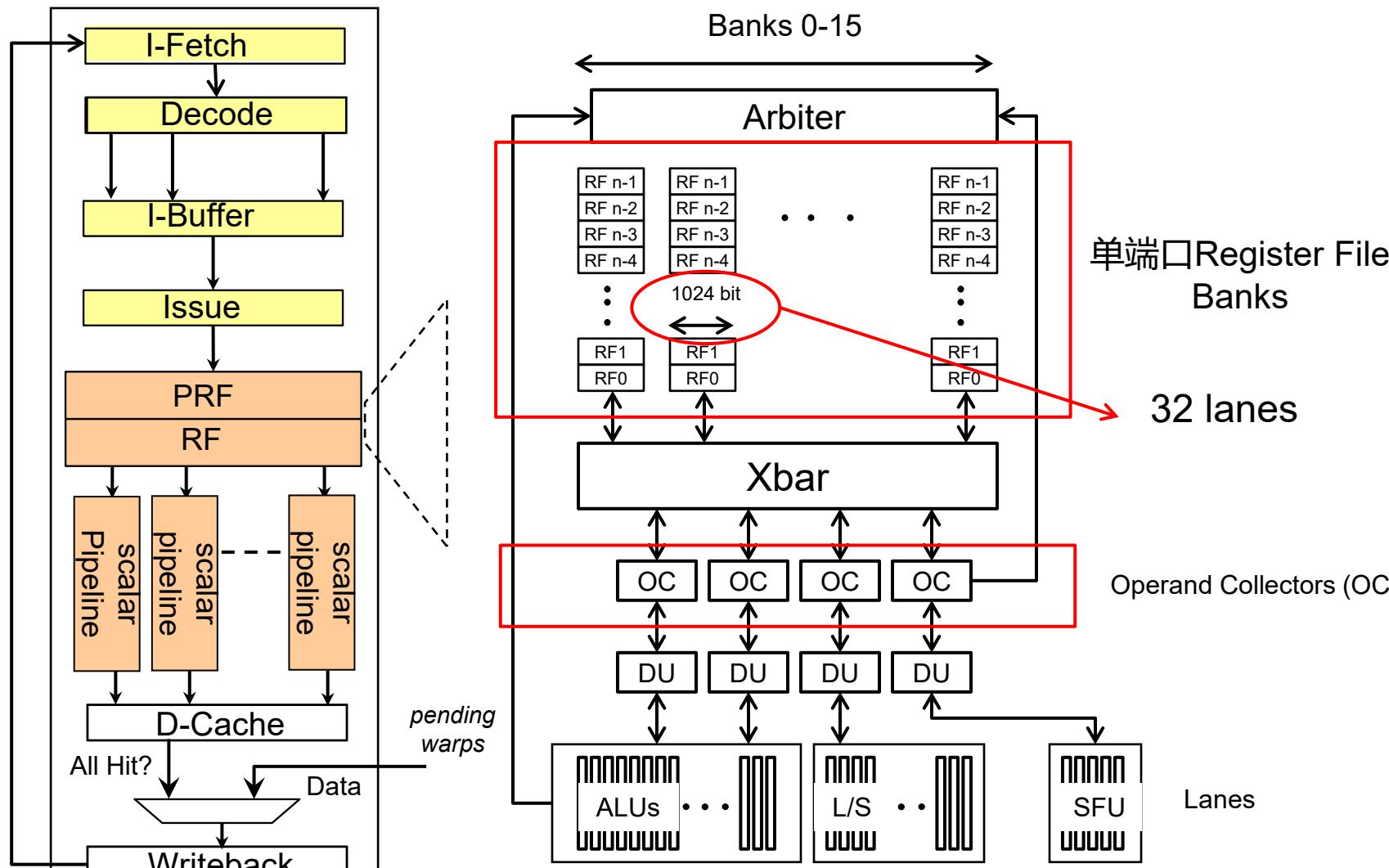
- 主机上调用了核函数之后，程序执行进入GPU，生成大量的线程。线程执行前，核函数执行配置，确定线程块数和每个线程块中的线程数以及共享内存大小
- Kernel -> grid -> block -> warp -> thread
- 线程块 (Block)** 是一组并发线程，可以通过屏蔽同步和共享内存来相互协作，可以是多维的。而网格则是一组线程块，每个线程块可独立执行
- 每个核函数 (Kernel)** 对应一组网格，而每个网格中包括多个线程块
- 线程通过以下坐标变量来区标识：
 - blockIdx，网格内的块标识
 - threadIdx，块内的线程标识



Block执行



典型GPU存储架构：从全局内存到Register File（寄存器堆）

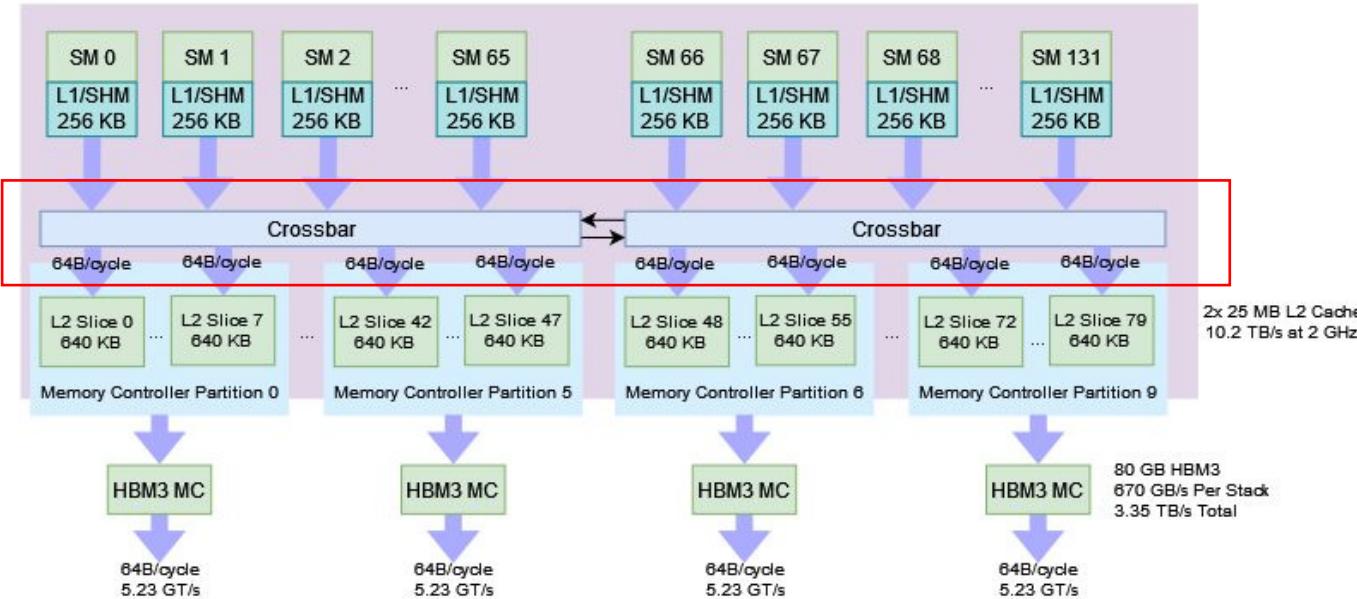


- 全局内存(Global Memory): 全局内存是GPU的主要存储空间，用于存储程序代码、数据和中间结果。全局内存的带宽和延迟对GPU的性能有着重要影响
- 纹理内存(Texture Memory): 纹理内存是一种只读内存，专门用于存储纹理数据。纹理内存具有缓存机制，能够高效地处理纹理采样操作
- 常量内存(Constant Memory): 常量内存是一种只读内存，用于存储程序中不会改变的常量数据。常量内存具有缓存机制，能够提供高速的常量访问
- 共享内存(Shared Memory): 共享内存是SM内部的高速缓存，由同一个SM内的所有线程共享。共享内存的访问速度比全局内存快得多，可以用于存储线程间需要共享的数据
- 寄存器堆(Register File): 寄存器堆是SM内部的高速存储单元，用于存储线程的局部变量和中间结果。寄存器堆的访问速度最快，但其容量有限。

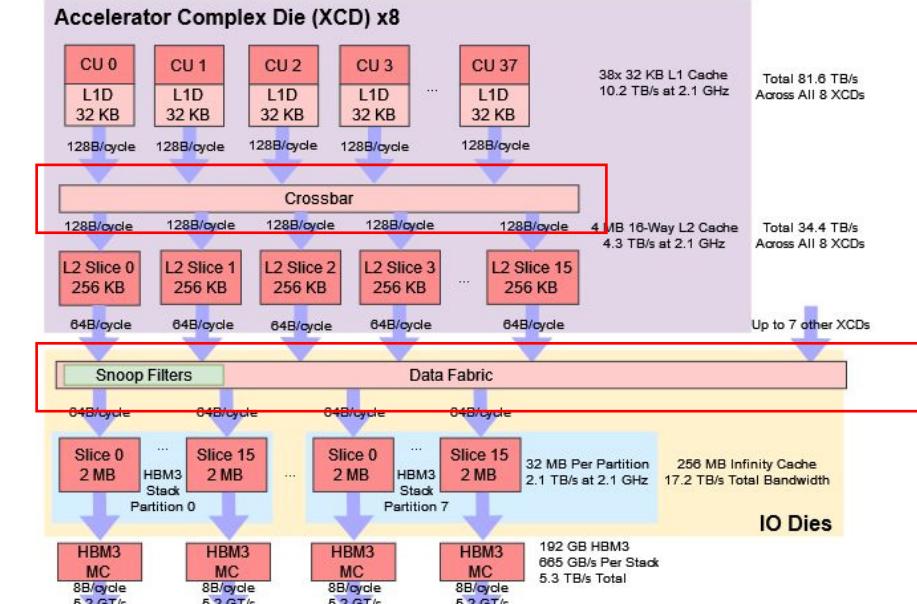
现代GPU的存储架构

- H100: 由132个流式多处理器 (SM) 组成，并将它们呈现为一个大型统一 GPU。L2缓存拆分为两个实例。单个 SM可以使用全部50MB的L2缓存，但访问超过25MB的缓存将导致性能下降
- 优点：可有效的利用缓存容量
- SM和L2 Bank之间经Crossbar片上网络互连，允许多个 SM 和 L2 Bank 之间同时通信，从而提供可观的NoC (Network on a chip, 片上网络) 吞吐量。一个典型的Crossbar片上网络封装了一个地址总线和两个数据总线。地址总线从SM 到 L2 Bank是单向的；而两条数据总线在SM和L2组之间形成双向通道，进行点对点通信
- MI300: 计算能力分离到加速器复合芯片 (Accelerator Complex Dies, 简称 XCD) 上。每个 XCD 包含一组核心和一个共享缓存
- MI300X 的 XCD 不会使用其他 XCD 的 L2 缓存容量进行缓存

Nvidia Hopper (H100 SXM5)

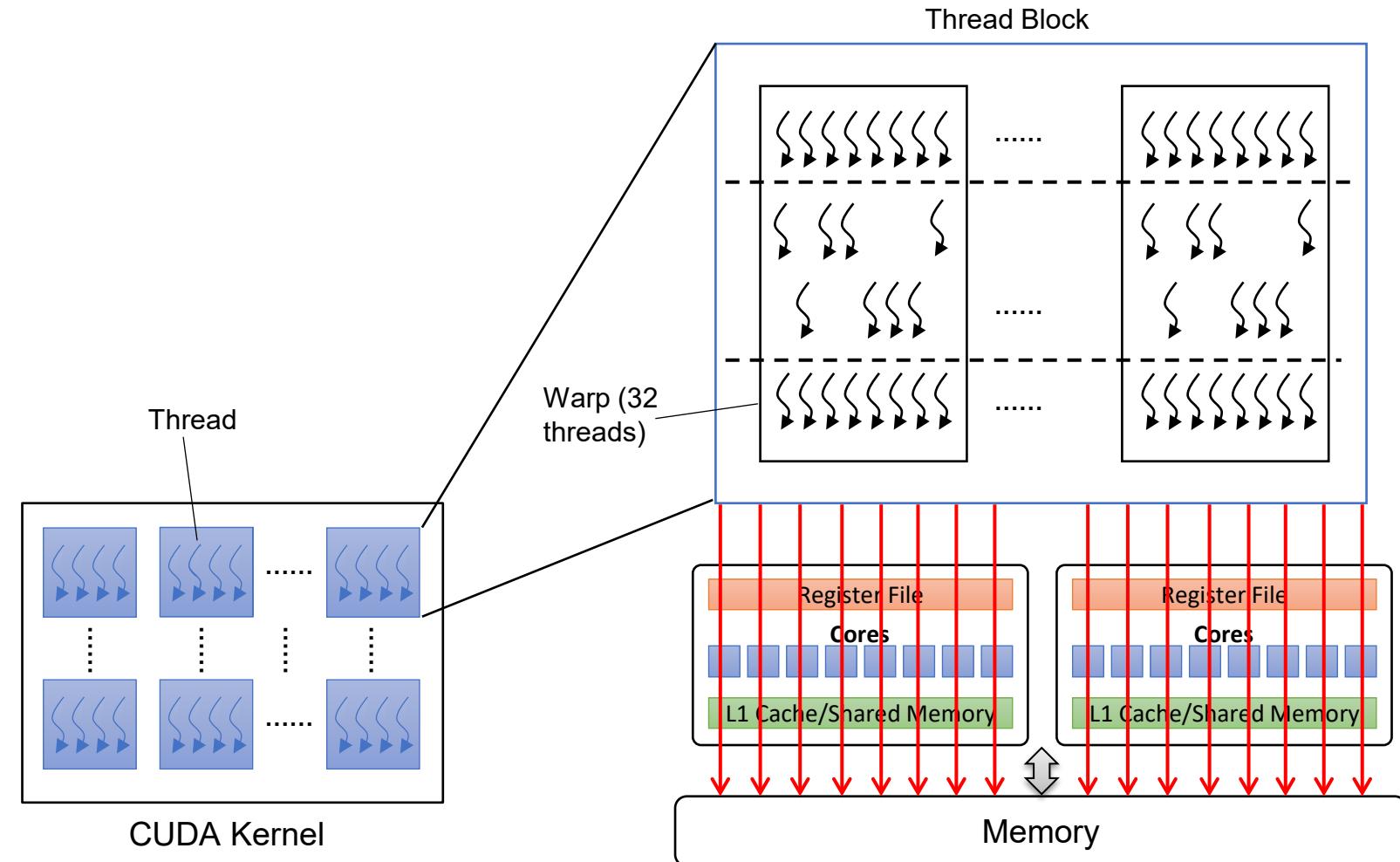


AMD CDNA3 (MI300X)



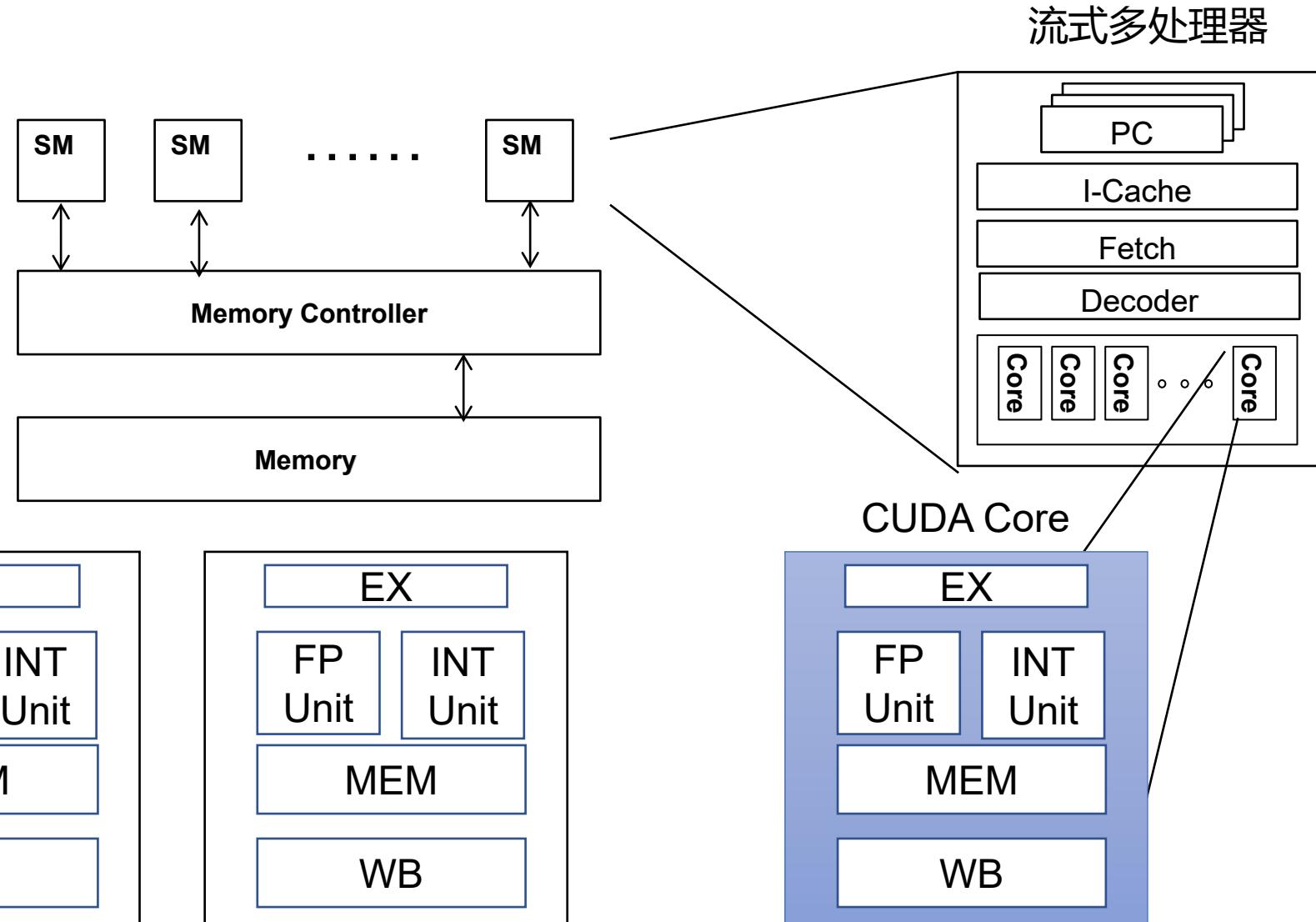
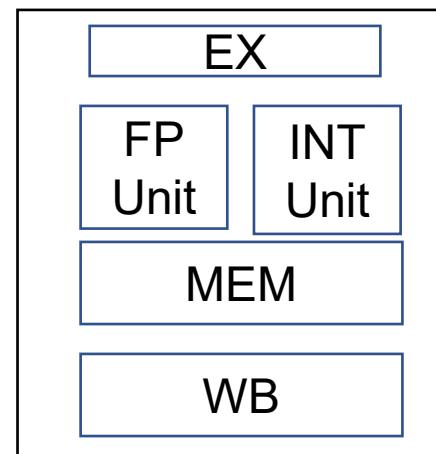
Kernel -> Grid -> Block -> Warp -> Thread

- Kernel: 用户编写的并行计算函数
- Grid: 由一个单独的kernel启动的所有线程组成一个grid。Grid中所有线程共享global memory
- (Thread) Block: 将数据分块处理。Block内部的多个线程可以同步(synchronize) , 可访问共享内存(share memory)
- Warp是GPU调度和运行的基本单元，一般由32个并行执行的线程组成，采用SIMT模式 (单指令多数据流) 。Warp内的线程必须执行相同指令，避免因分支导致性能下降。Warp由SM的硬件warp scheduler负责调度，一个SM同一个时刻可以执行多个warp
- Thread是GPU执行的基本单位，每个CUDA Core对应一个线程



SIMT核心/CUDA核心

- 一个Warp一次执行一条通用指令Thread (Warp中) 映射到 CUDA 核心
- 同一个warp中的thread可以以任意顺序执行
- 每个SM有一个32位register集合放在register file中，还有固定数量的shared memory

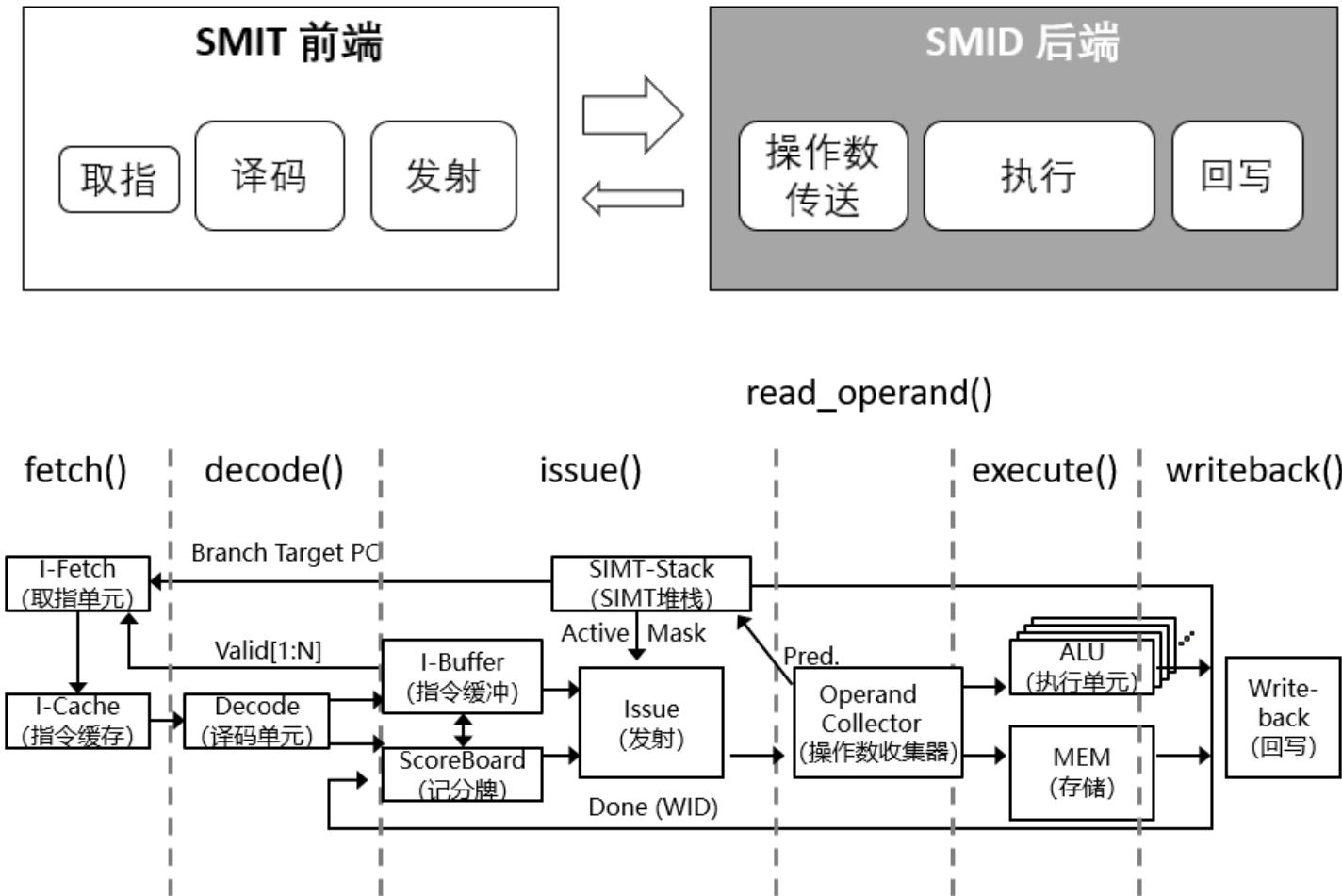


流式处理器基本结构

流式多处理器按照流水线可以分为**SIMT前端**和**SIMD后端**。整个流水线处理划分为六个阶段，包括取指、译码、发射、操作数传送、执行与写回

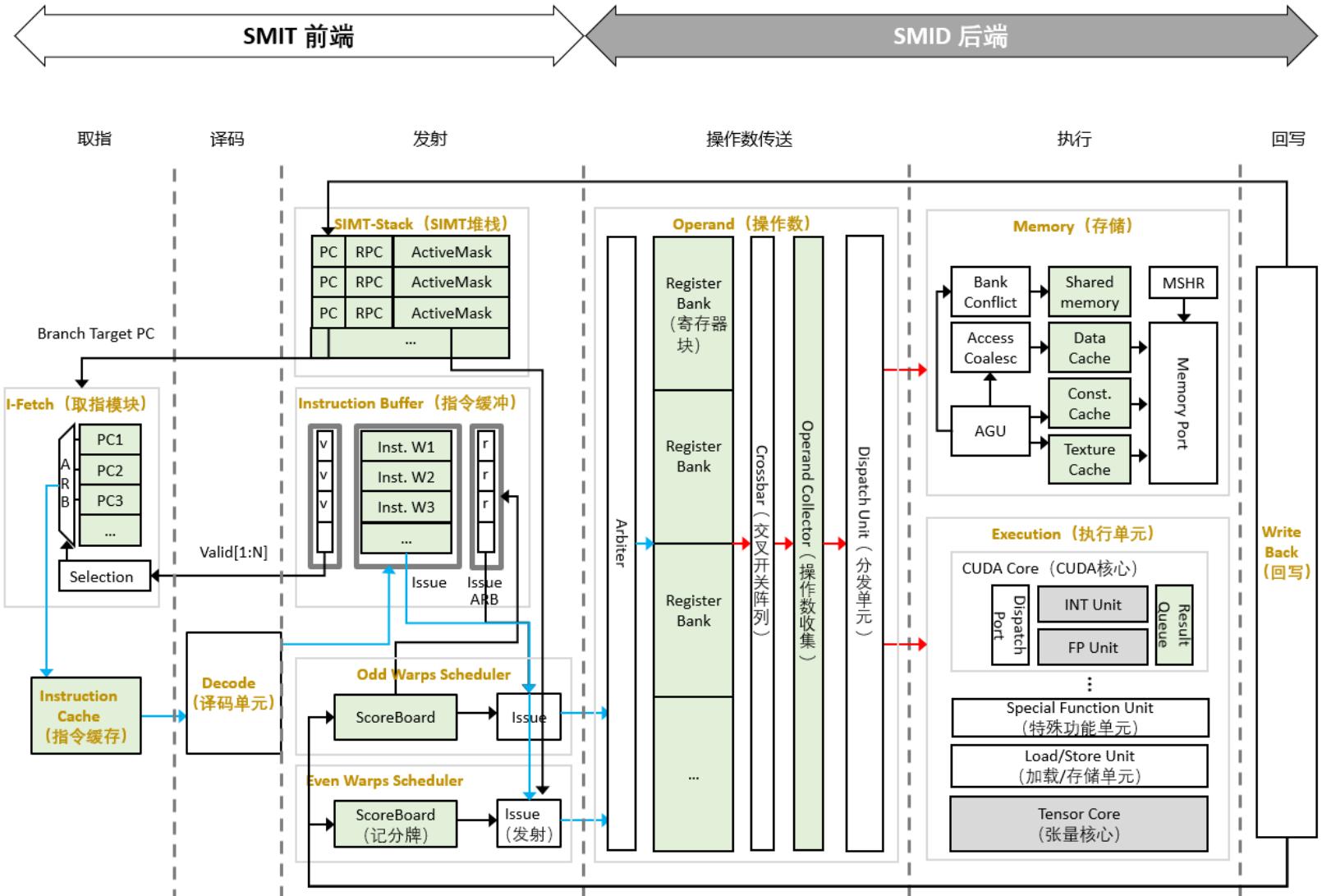
按照软件级别，**SIMT层面**，流式多处理器由线程块组成，每个线程块由多个线程束组；**SIMD层面**，每个线程束内部在同一时间执行相同指令，对应不同数据，由统一的**线程束调度器**（Warp scheduler）调度

线程块调度程序将线程块分派给SIMT前端，线程在流式多处理器上以Warp为单位并行执行



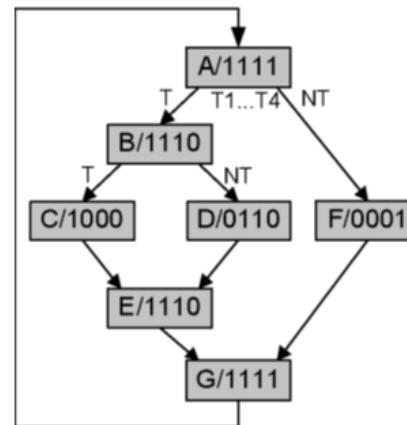
流式处理器完整架构

- 取指单元 (I-Fetch)**：负责将指令请求发送到指令缓存。并将程序计数器 (PC) 指向下一条指令
- 指令缓存 (I-Cache)**：如来自取指单元的请求在指令缓存中被命中，则将指令传送给译码单元，否则把请求保存在未命中状态保持寄存器 (MSHR) 中
- 译码单元 (Decode)**：将指令解码并转发至 I-Buffer。该单元还将源和目标寄存器信息转发到记分牌，并将指令类型、目标地址（用于分支）和其他控制流相关信息转发到 SIMT 堆栈
- SIMT 堆栈 (SIMT Stack)**：SIMT 堆栈负责管理控制流相关的指令和提供下一程序计数器相关的信息
- 记分牌 (Scoreboard)**：用于支持指令级并行。并行执行多条独立指令时，由记分牌跟踪挂起的寄存器写入状态避免重复写入
- 指令缓冲 (I-Buffer)**：保存所有 Warp 中解码后的指令信息。Warp 的循环调度策略决定了指令发射到执行和写回阶段的顺序
- 后端执行单元**：后端执行单元包括 CUDA 核心（相当于 ALU）、特殊功能函数、LD/ST 单元、张量核心 (Tensor core)。特殊功能单元的数量通常比较少，计算相对复杂且执行速度较慢。（例如，正弦、余弦、倒数、平方根）
- 共享存储**：除了寄存器文件，流式多处理器也有共享存储，用于保存线程块不同线程经常使用的公共数据，以减少对全局内存的访问频率



SIMT掩码

- SIMT堆栈中使用了SIMT掩码 (SIMT Mask) 来处理线程束分化问题
- 通过串行执行完毕分支之后，线程在 Reconverge Point (重合点) 又重新聚合在一起以便最大提高其并行能力
- 当出现分支时按照减少重合堆栈 (Reconvergence stack) 的深度的方法，先执行活跃线程最多的分支，然后再执行活跃少的分支
- 死锁问题：如果出现分支就表明每个分支的指令和处理是不一致的，容易使一些共享数据失去一致性。如果在同一个Warp内如果存在分支，则线程之间可能不能够交互或者进行数据交换，在一些实际算法中可能使用锁定 (Lock) 机制来进行数据交换。但掩码恰恰可能因为调度失衡，造成锁定一直不能被解除，造成死锁问题



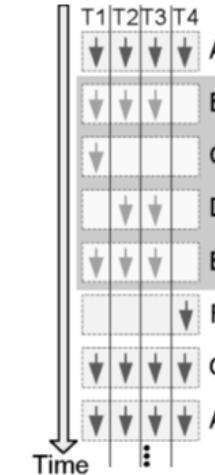
(a) Control Flow Graph of an Example Program

Next PC	Active Mask	Ret./Reconv. PC
G	1111	-
F	0001	G
B	1110	G

(c) Stack based Reconvergence: Initial State

Next PC	Active Mask	Ret./Reconv. PC
G	1111	-
F	0001	G
E	1110	G
D	0110	E
C	1000	E

(d) Stack based Reconvergence: After Divergent Branch



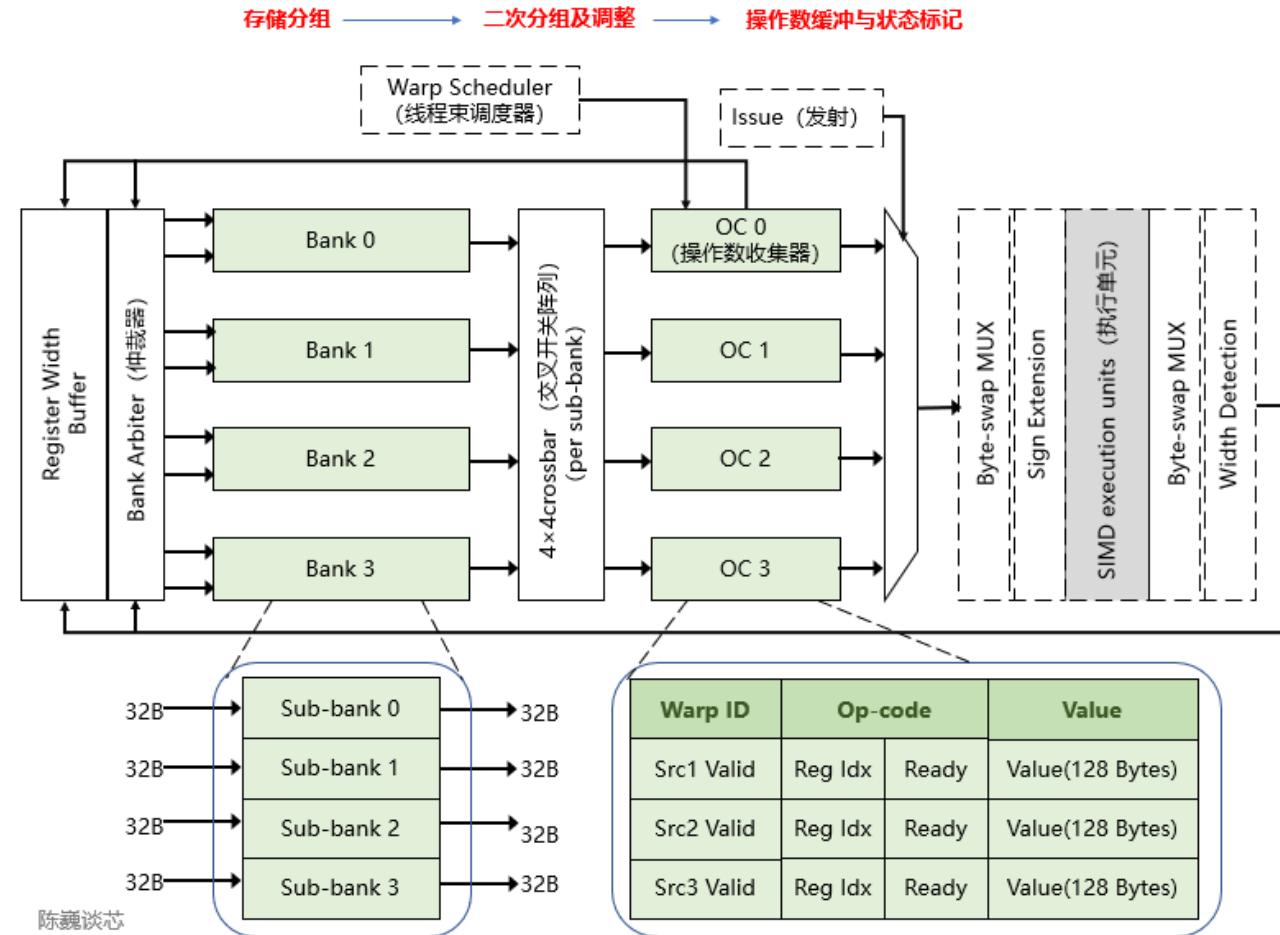
(b) Resource Utilization vs. Time for Reconvergence at Immediate Postdominator of B

Next PC	Active Mask	Ret./Reconv. PC
G	1111	-
F	0001	G
E	1110	G

(e) Stack based Reconvergence: After Reconvergence

操作数传送与访问冲突的处理

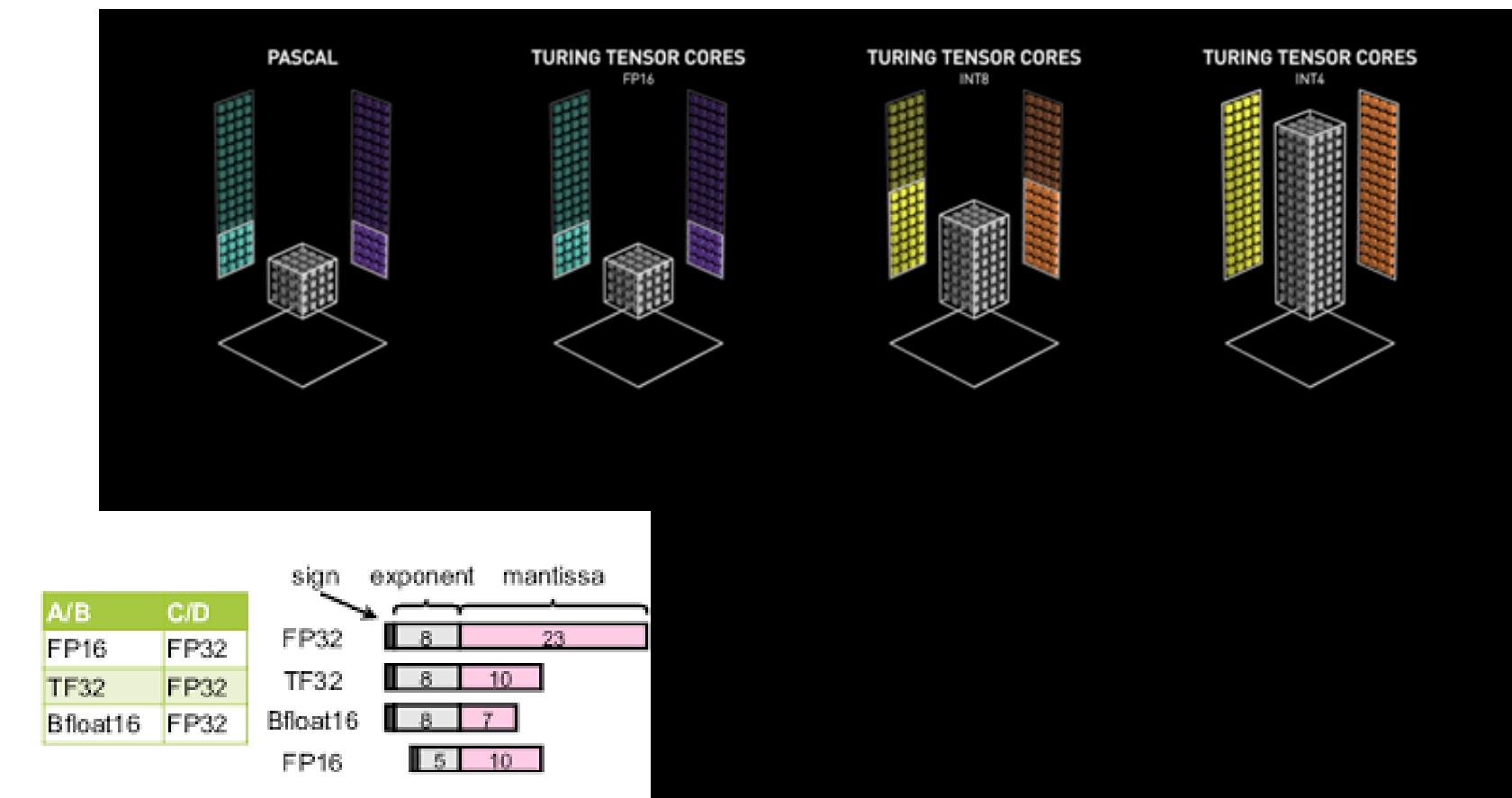
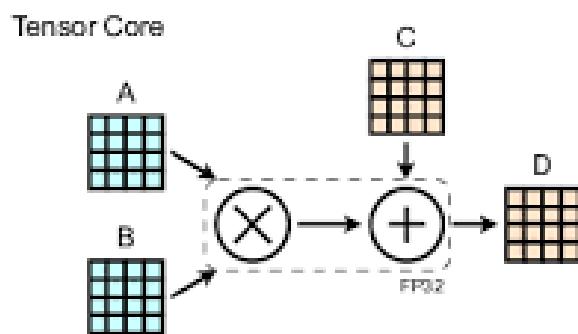
- 通过在Warp内不断切换线程来隐藏内存访问延迟是可能的，但是切换线程意味着需要大量寄存器堆来保存上下文信息。在实际设计时，一般采用序列化的寄存器堆访问来避免多端口寄存器的高成本
- 1) 使用单端口SRAM代替常规寄存器，提升存储密度。与常规双端口8T SRAM相比，单端口6T SRAM具有显著的面积优势
- 2) 进行寄存器合并/打包，将来自同一指令的多个寄存器读取合并到单个物理寄存器的单次或集中读取，降低平均读取功耗
- 3) 寄存器堆分块（几组较小的SRAM），保障足够的操作数同时进入到执行模块中，减少了执行单元的等待



陈巍谈芯

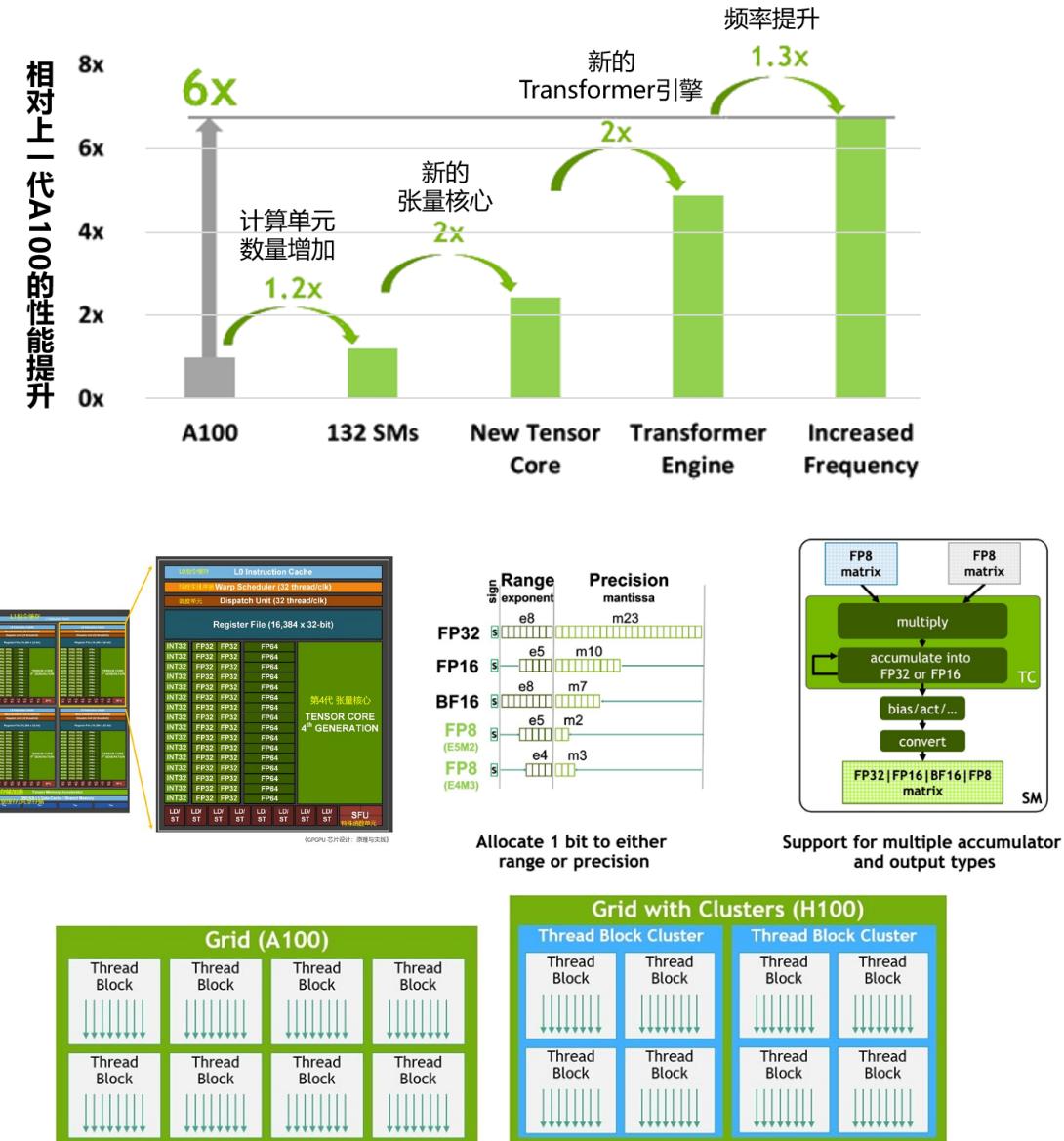
张量核心

- TensorCore 是从Nvidia Volta 架构 GPU开始支持的重要特性，使CUDA 开发者能够使用混合精度来获得更高的吞吐量
- GPU中的张量核心是一种固定模式的矩阵计算核心，支持混合精度计算，动态调整计算精度以加快计算，同时保持合适的准确性
- 具体分析见后继课程



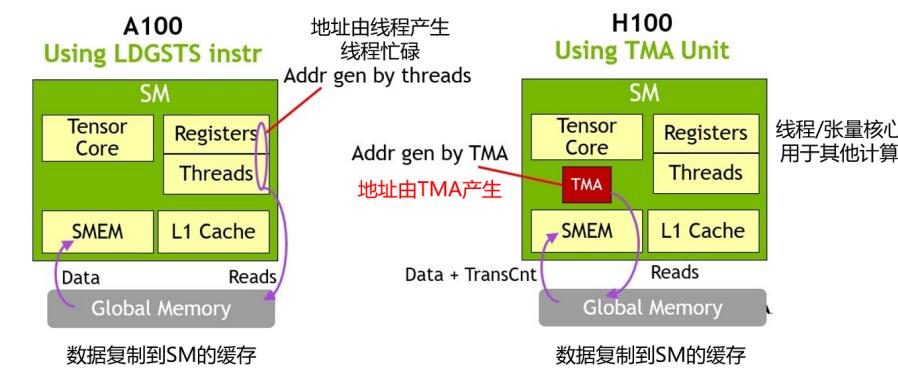
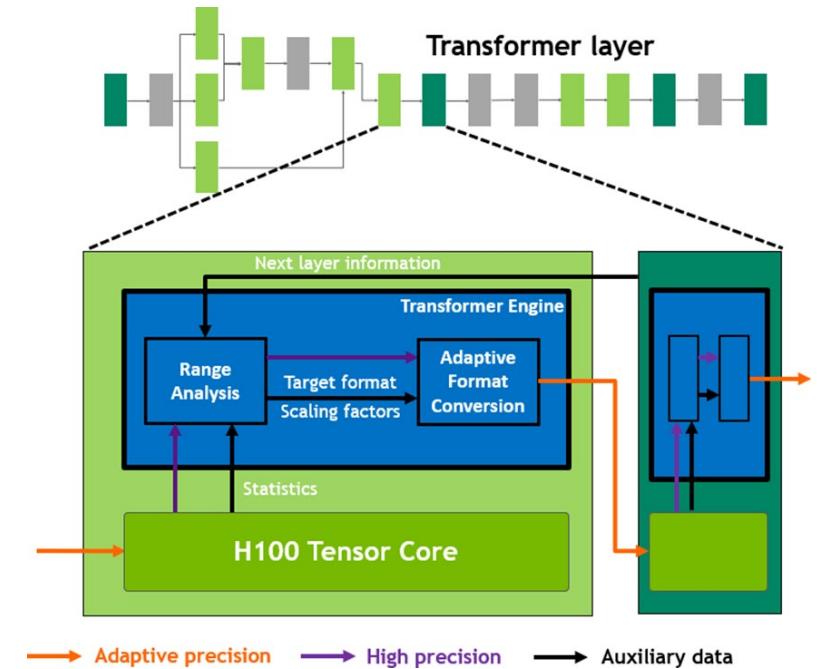
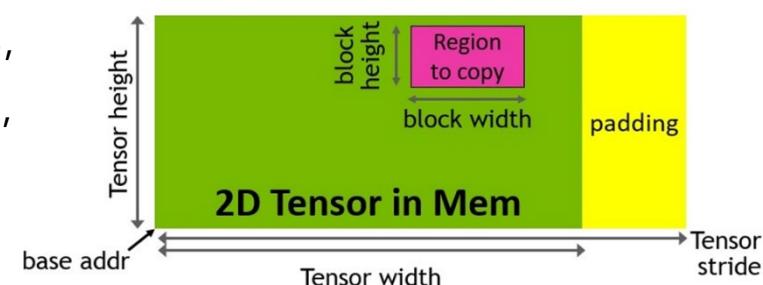
现代GPU架构 (H100/Hopper架构)

- Hopper架构GPU 由8个图形处理集群 (Graphics Processing Cluster, GPC) “拼接” 组成。外周与多组HBM3封装在一起 (Chiplet技术)
- Hopper架构的性能提升和主要变化体现在新型线程块集群技术和新一代的流式多处理器
- Hopper架构中引入了一种新的线程块集群机制，该机制可以跨SM单元进行协同计算。H100 中的线程块集群可在同一GPC内的大量SM并发运行，这样对较大的模型具有更好的加速能力
- Hopper架构的新一代流式多处理器中，引入了 **FP8张量核心** (Tensor Core) 来加速 AI 训练和推理



针对大模型/Transformer的优化

- 张量存储加速器 (Tensor Memory Accelerator, TMA) , 以提高张量核心与全局存储和共享存储的数据交换效率
- TMA (异步操作, 多线程排序完成数据传输) 使用张量维度和块坐标指定数据传输, 而不是简单的按数据地址直接寻址。TMA 通过支持不同的张量布局 (1D-5D 张量) 、不同的存储访问模式、卸载线程的数据复制任务, 显著降低了寻址开销并提高了效率
- 新的Transformer引擎中使用了混合精度, 在计算过程中智能地管理计算精度, 在 Transformer计算的每一层, 根据下一层神经网络层及所需的精度, 在FP8和其他浮点格式中进行动态格式转换, 充分运用张量核心的算力

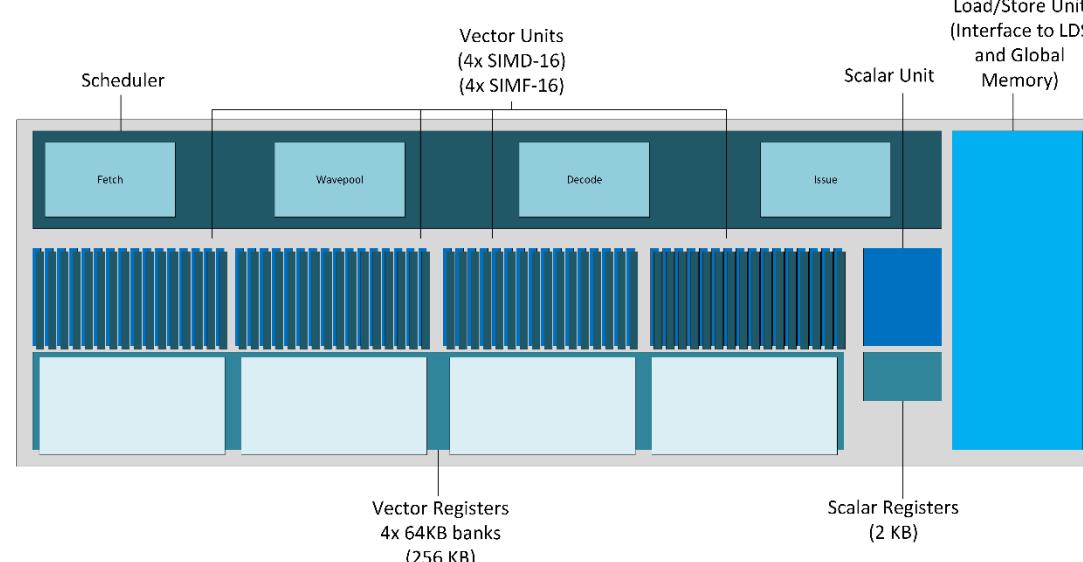
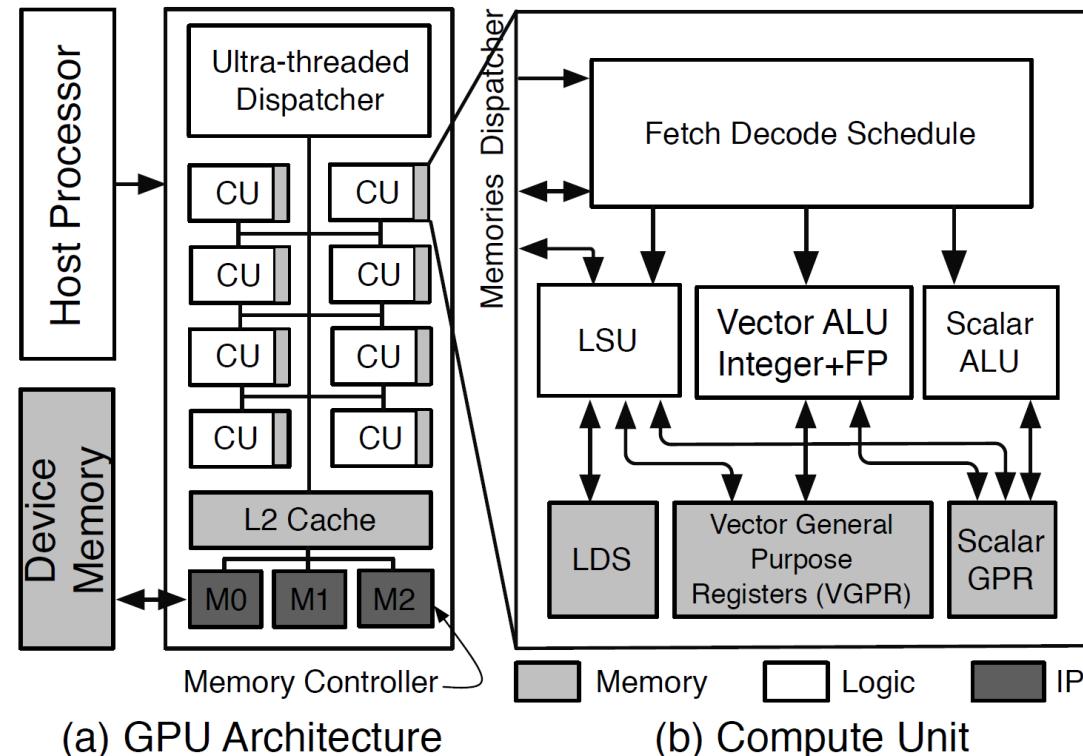


开源GPU蓬勃发展

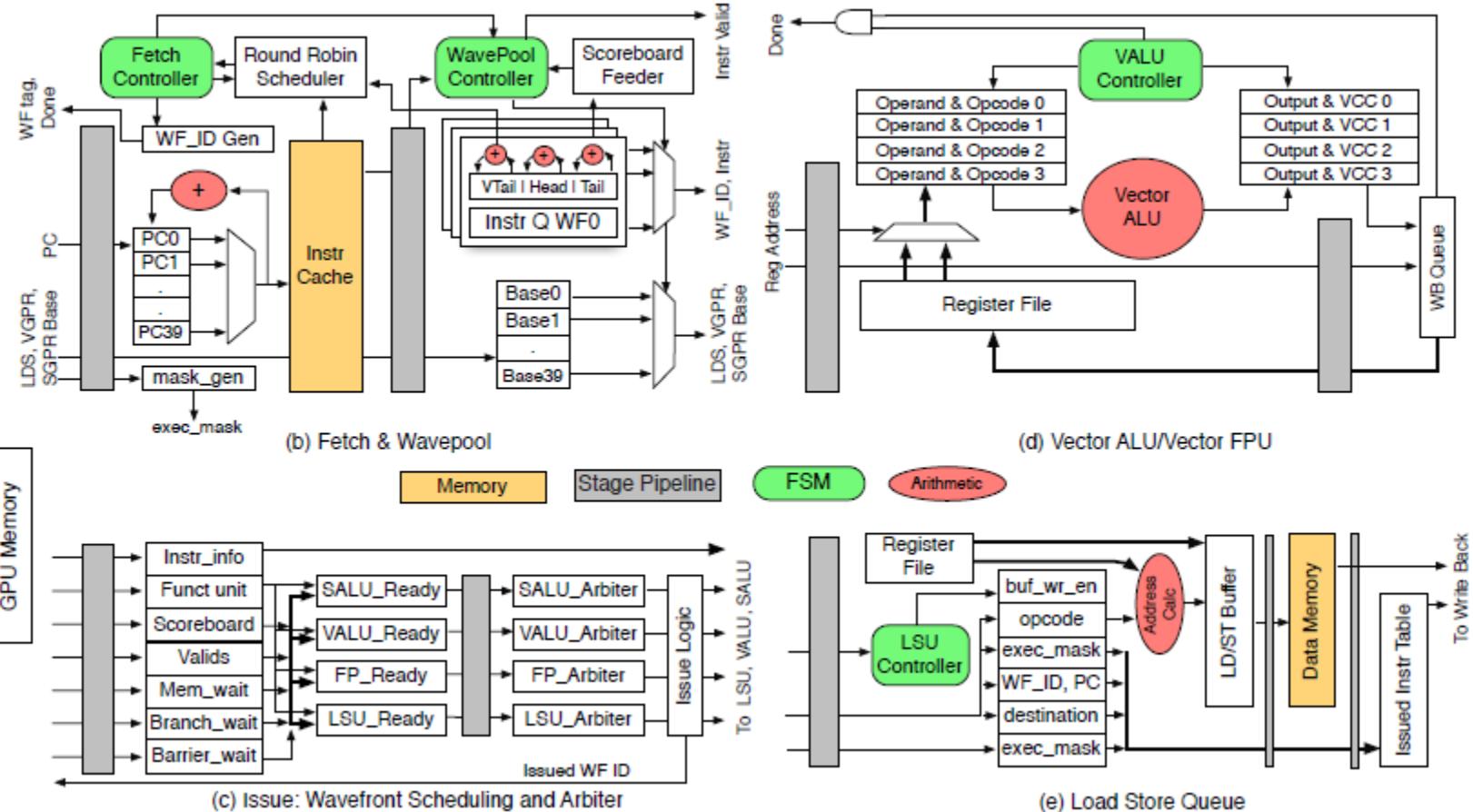
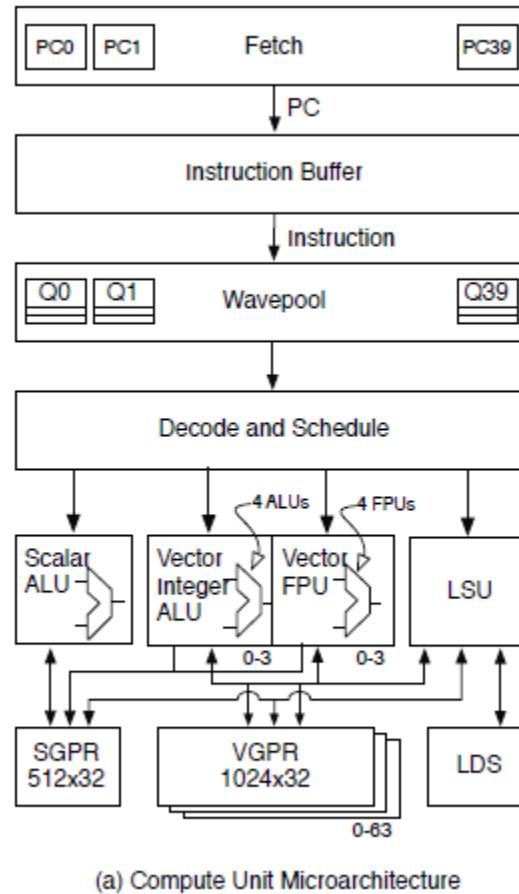
序号	GPU	发布时间	指令集	模式	图形支持能力	软件栈支持	特点
1	e-GPU	2025年5月	RISC-V	SIMT	主要专注于TinyAI应用，无明确3D图形渲染支持	有软件栈，支持Tiny-OpenCL轻量级编程框架	开源可配置RISC-V GPU平台；针对TinyAI设备；与X-HEEP集成；TSMC 16nm实现，300MHz@0.8V；多配置支持，最高16线程，15.1x加速，3.1x能效提升
2	FuryGPU	2024年3月	自定义	SIMT	支持1990年代高端图形功能；运行Quake在720p下达60fps	完整现代Windows软件驱动栈	基于Xilinx Zynq UltraScale+ FPGA的硬件GPU；自定义PCB和PCIe连接；开源硬件和软件项目，专注于实时游戏渲染
3	Ventus (乘影)	2024年	RISC-V (RV32IMAFD/RV64IMAFD扩展，带向量扩展)	SIMT	支持3D图形和OpenGL渲染	有软件栈，支持OpenCL 2.0；完整软件栈包括LLVM-based compiler, PoCL-based runtime, kernel driver	高性能开源GPGPU；可配置微架构；支持tensor operations；Chisel HDL开发；FPGA实现；扩展RVV为SIMT架构；支持生成式AI和LLM应用
4	rvgpu	2023年	RISC-V (RISCV64GC)	SMIT	无明确图形渲染支持	支持CUDA (含Runtime)、OpenCL、Vulkan	模拟器基于qemu，底层编译由llvm支持、具备cmodel和驱动，文档很少
5	Skybox	2023年5月	RISC-V	SMIT	支持Vulkan图形API	全栈开源，包括软件、编译器、硬件和模拟环境	开源图形渲染在可编程RISC-V GPU上；混合微架构加速软件/硬件渲染；端到端GPU研究；Altera Stratix 10 FPGA上32核心（512线程），3.7 GPixels@230MHz
6	Simty	未明确 (约2023年)	RISC-V	SIMT	未明确，主要为GPGPU	有软件栈，支持类似CUDA的编程 (NoCL库)	Synthesizable SIMT-style RISC-V GPGPU核心；动态标量化；并行标量/向量管道；寄存器文件和缓存压缩；CHERI内存安全；支持CUDA-like C++库和基准测试
7	tiny-gpu	未明确	自定义11指令ISA (包括BRnzp、CMP、ADD、SUB、MUL、DIV、LDR、STR、CONST、RET)	SIMT (通过SIMD 实现，线程同步执行)	无 (专注于GPGPU计算；未来计划添加基本图形内核)	支持HIP API，与PyTorch兼容；开放支持SYCL或CUDA；单源C++编译和运行时	最小Verilog GPU设计，用于学习GPU基础；<15个文件，全文档；支持线程块调度、异步内存请求；证明概念内核如矩阵运算；未来添加缓存、分支分歧等
8	VeriGPU	2022年4月	基于RISC-V的松散兼容ISA (在与GPU设计冲突时偏离RISC-V)	SIMT (隐含于GPU设计，用于ML工作负载)	有限，主要针对机器学习而非图形渲染	针对ASIC tape-out的开源GPU，专注于机器学习；使用BF16浮点以降低核心面积；仅实现ML关键浮点操作 (如exp, log, tanh, sqrt)；模拟支持和验证流程	针对ASIC tape-out的开源GPU，专注于机器学习；使用BF16浮点以降低核心面积；仅实现ML关键浮点操作 (如exp, log, tanh, sqrt)；模拟支持和验证流程
9	Vortex	2021年10月	RISC-V (RV32IMF和RV64IMAFD扩展)	SIMT	支持3D图形、OpenGL和Vulkan渲染	有软件栈，支持OpenCL 1.2和CUDA；完整软件和硬件栈，包括内核运行时、主机驱动和模拟器	全栈开源RISC-V GPGPU；可配置微架构 (核心、warp、线程数)；支持本地内存、L1/L2/L3缓存；FPGA实现，可扩展至32核心；用于ML、图形和图分析；PCIe-based soft GPU
10	RV64X	2021年1月	RISC-V (带向量扩展，支持8/16/32位定/浮点标量和向量)	Vector (带SIMD-like并行集群)	计划支持Vulkan、OpenGL/DX；使用Zink/ANGLE/Wine	Vulkan起点；Mesa兼容OpenGL；ANGLE for OpenGL ES；Wine for Direct3D	免费开源RISC-V GPU；支持通用计算 (ML/AI用INT8)；与CPU共享内存模型；适合Android
11	Scratch	2017年10月	基于AMD Southern Islands ISA (扩展自MIAOW)	SIMT	无明确图形渲染支持	有软件栈，支持端到端工具链，包括应用感知修剪工具；OpenCL-based	端到端应用感知soft-GPGPU架构和修剪工具；自动识别内核需求；基于FPGA；扩展MIAOW以优化资源使用
12	FGPU	2018年	自定义ISA，支持OpenCL内核	SIMT	无明确图形渲染支持	有软件栈，支持OpenCL；通过Python API控制	开源RTL实现，用于FPGA上的通用计算；高度优化构建块；可编程在OpenCL中；聚焦高性能并行计算
13	Theia	未明确 (软件项目，非硬件GPU；约2018年)	未适用 (软件光栅器)		支持软件渲染3D图形	有软件栈，使用SDL2和Eigen3；C++实现	软件光栅器，非硬件GPU；用于理解GPU管道；不适合作为硬件开源GPU对比
14	NyuziRaster	2016年	自定义GPGPU ISA (基于Nyuzi处理器)	SIMT	支持图形渲染，包括3D模型	有软件栈，包括LLVM-based C/C++编译器；指令集仿真器	开源soft GPU，支持图形渲染；集成多线程处理器；用于微架构实验FPGA实现
15	FlexiGrip	2015年 (约)	基于NVIDIA G80 (Tesla微架构)	SIMT	无明确图形渲染支持	有软件栈，支持类似CUDA的编程	开源VHDL GPGPU模型；完全微架构实现；用于FPGA；FlexGripPlus版本扩展功能
16	HWACHA	2015年	RISC-V (带非标准扩展Xhwacha)	Vector (SIMT-like with micro-ops)	支持3D模型渲染示例	有软件栈，支持OpenCL编译器 (进行中)；LLVM-based C/C++编译器；指令集仿真器	Decoupled vector-fetch accelerator；RoCC接口集成；支持loop-based auto-vectorizer；FPGA实现；用于IoT和DSP应用
17	MIAOW	2014年10月	AMD Southern Islands ISA	SIMT	无图形渲染支持	有软件栈，支持OpenCL子集	开源RTL实现AMD GPU ISA；用于GPGPU工作负载架构分析；Verilog设计；FPGA原型；运行OpenCL程序
18	GPLGPU (kickstarter)	2014年8月	未指定 (Verilog图形引擎，兼容Number Nine GPU)	未指定 (固定功能 GPU)	支持2D/3D图形加速；VGA控制器	无明确软件栈，Verilog实现；PCI总线接GPL v3许可的2D/3D图形引擎；类似于1990年代末GPU；纯固定功能，无顶点变换；开源硬件设计	无明确软件栈，Verilog实现；PCI总线接GPL v3许可的2D/3D图形引擎；类似于1990年代末GPU；纯固定功能，无顶点变换；开源硬件设计

MIAOW整体架构

- MIAOW (发音为 “me-ow”) 是由威斯康星大学麦迪逊分校研究小组开发的开源GPU
- MIAOW基于AMD公开发布的Southern Islands指令集架构 (SI ISA) , 实现了一个适用于执行AMD Graphics Core Next (GCN) 微架构分析和GPGPU工作负载实验的计算单元
- 计算单元 (CU) 由执行标量和向量运算所需的模块组成。取指单元作为 CU 和调度器之间的接口, 接收执行wavefront所需的信息。wavepool作为所有已取指指令的队列, 能够同时跟踪和支持多达 40 个不同的波前。解码单元负责指令解码和 64 位指令的整理

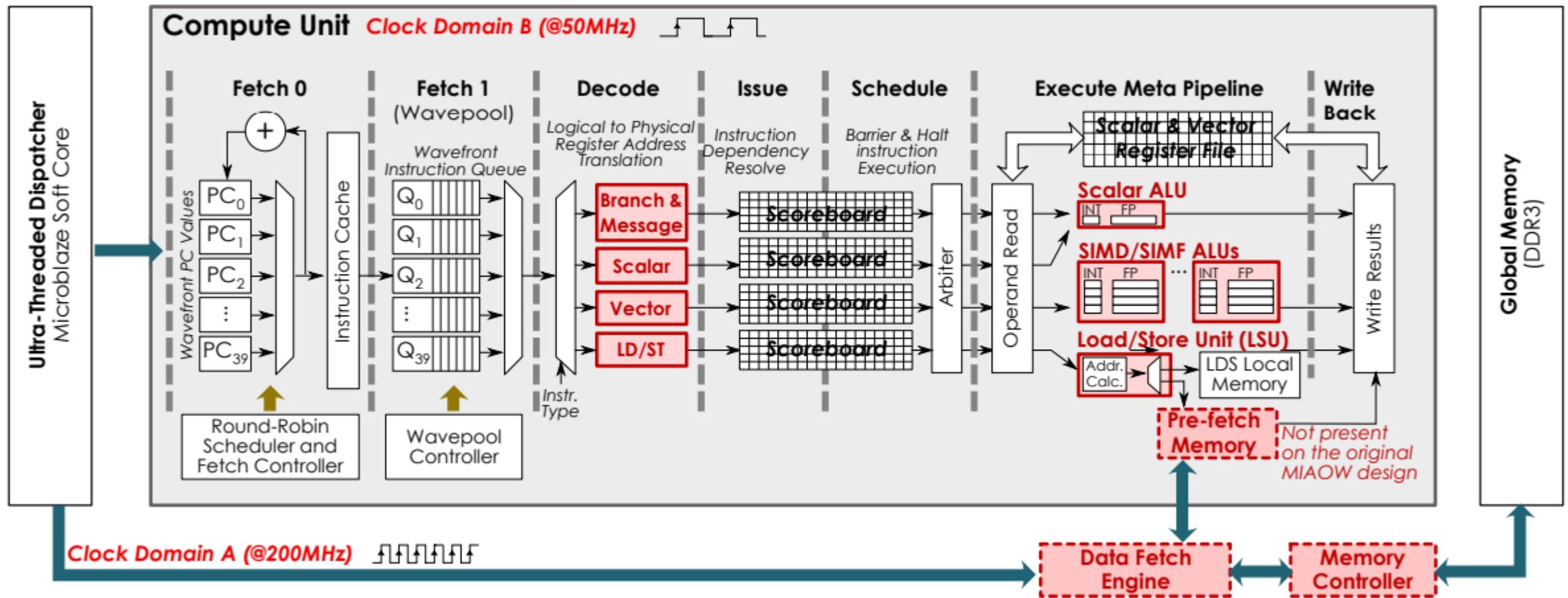


MIAOW微架构



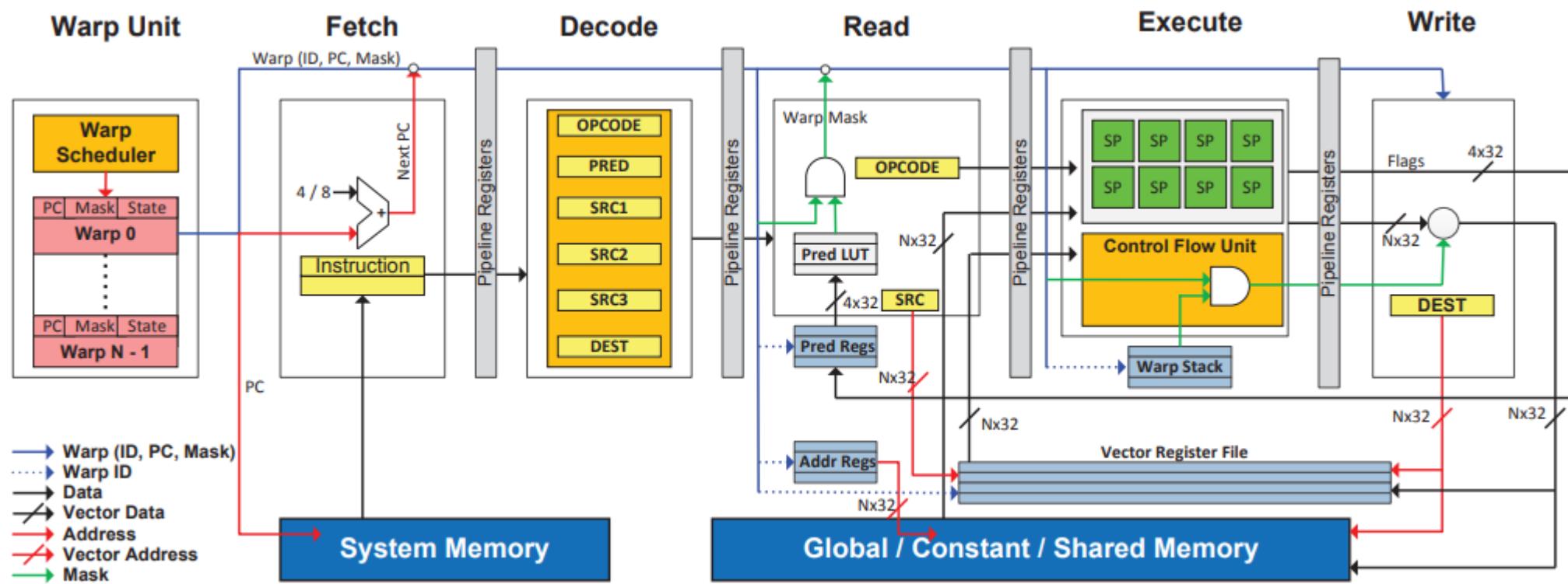
MIAOW2-SCRATCH

- 基于原始 MIAOW 系统（此处命名为 MIAOW2.0）的改进版本，此处扩展为支持一组 156 条指令，并增强以提供快速预取存储器系统和双时钟域。与没有修剪的优化版本相比，加速比提高了 2.4 倍，能效提高了 2.1 倍
- 引入了CU架构改进，即：一个单独的时钟域来解耦计算单元的关键路径：超线程调度程序和预取内存系统，以减少原设计的数据访问延迟



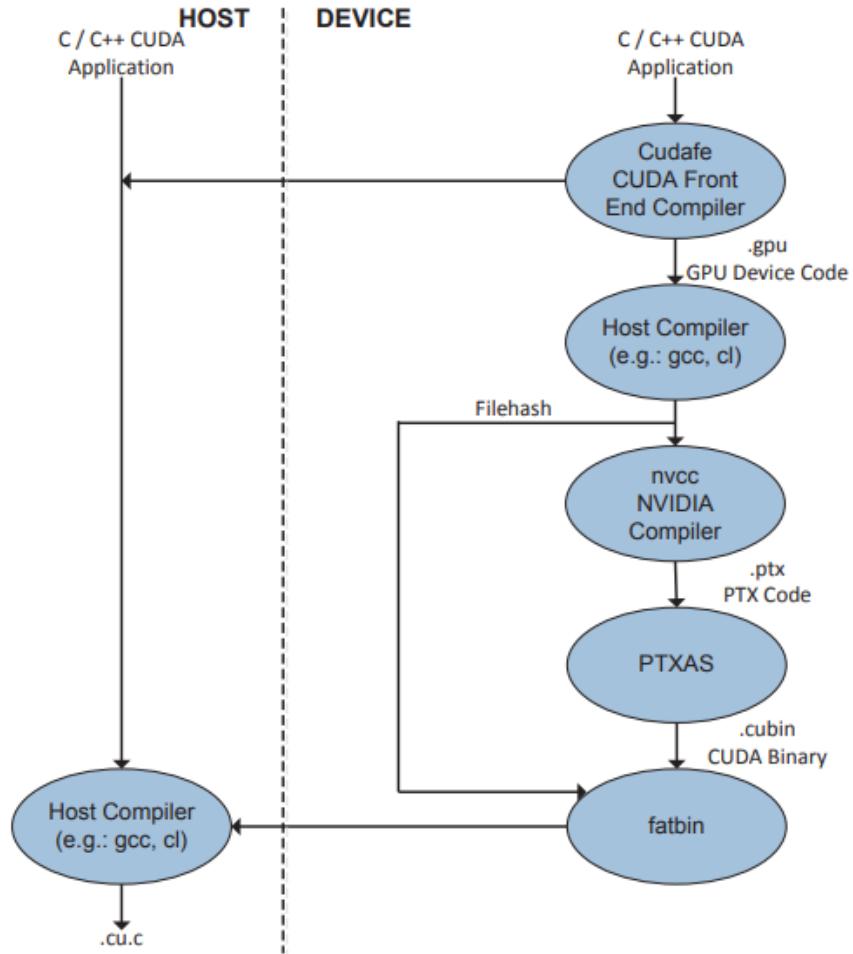
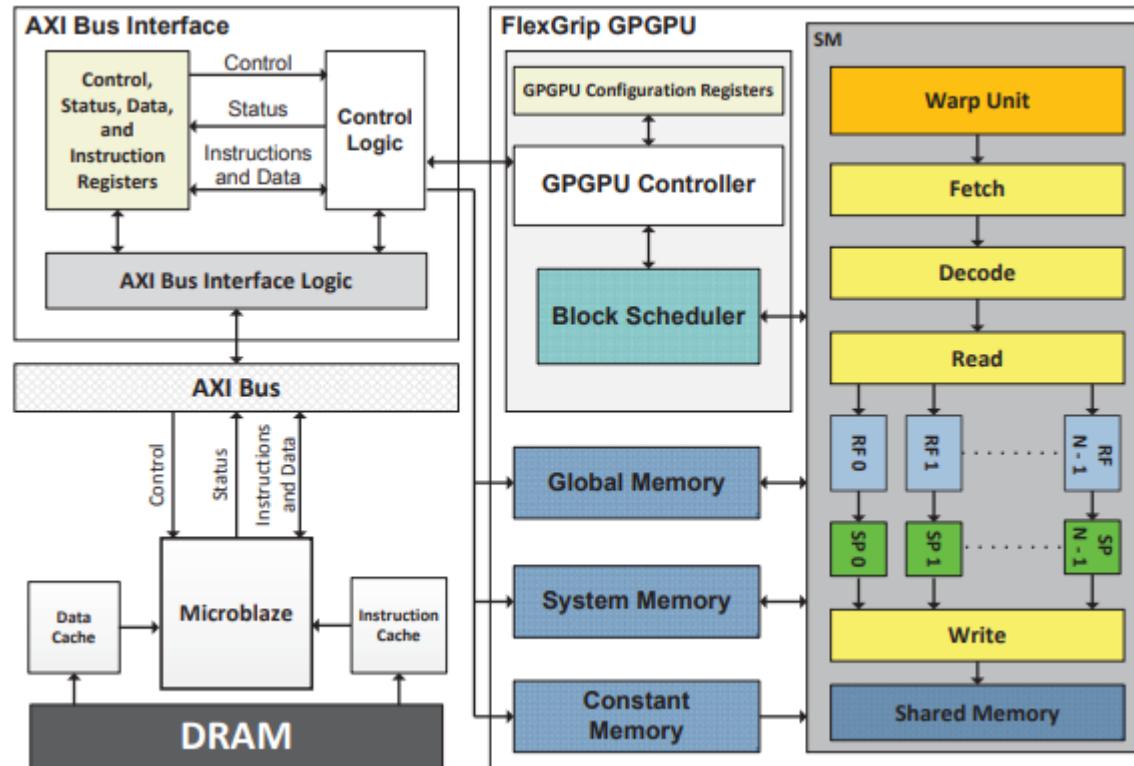
FlexGripPlus流式多处理器

- FlexGripPlus是一款完全用 VHDL 语言描述的开源GPU，并实现了NVIDIA的G80微架构。最初由 University of Massachusetts 开发，并于 2010 年发布
- 改进后的版本完全兼容SM_1.0兼容的 CUDA 编程环境，支持 28 条汇编 (SASS) 指令



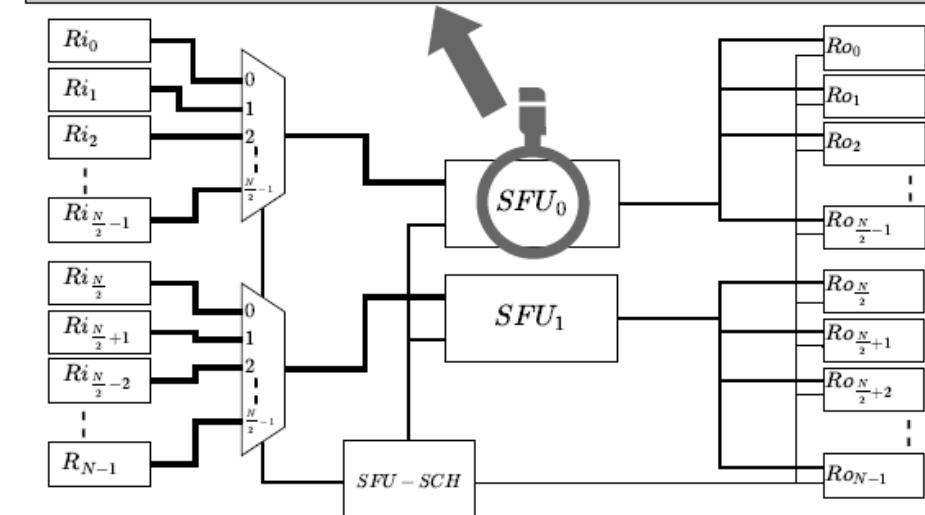
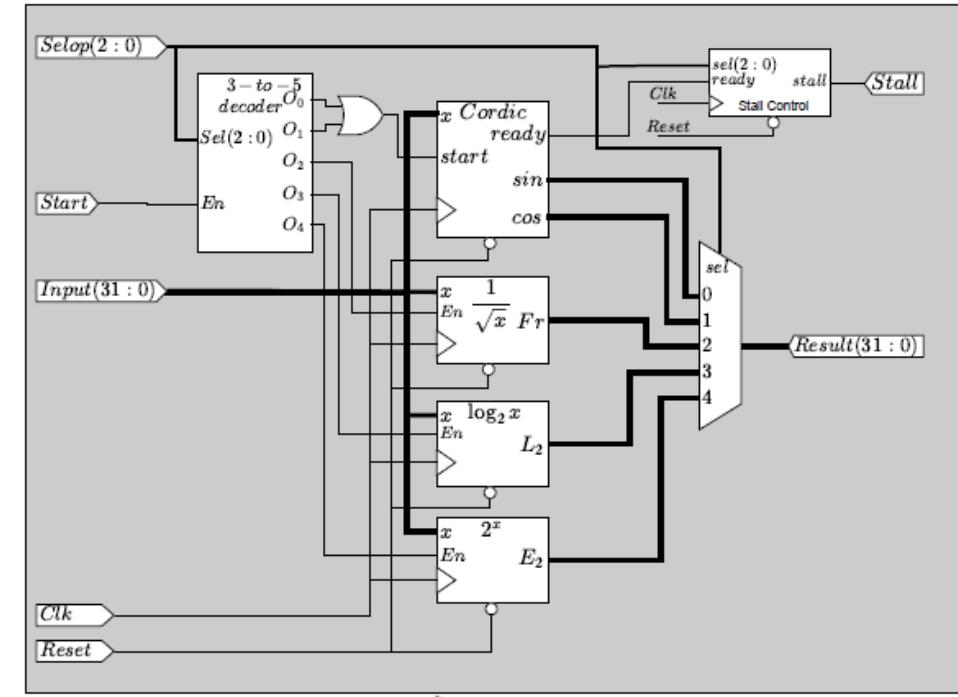
FlexGripPlus整体架构与软件栈

- 该架构支持将 CUDA 直接编译为二进制文件，该二进制文件可在基于 FPGA 的 GPGPU 上执行，无需硬件重新编译



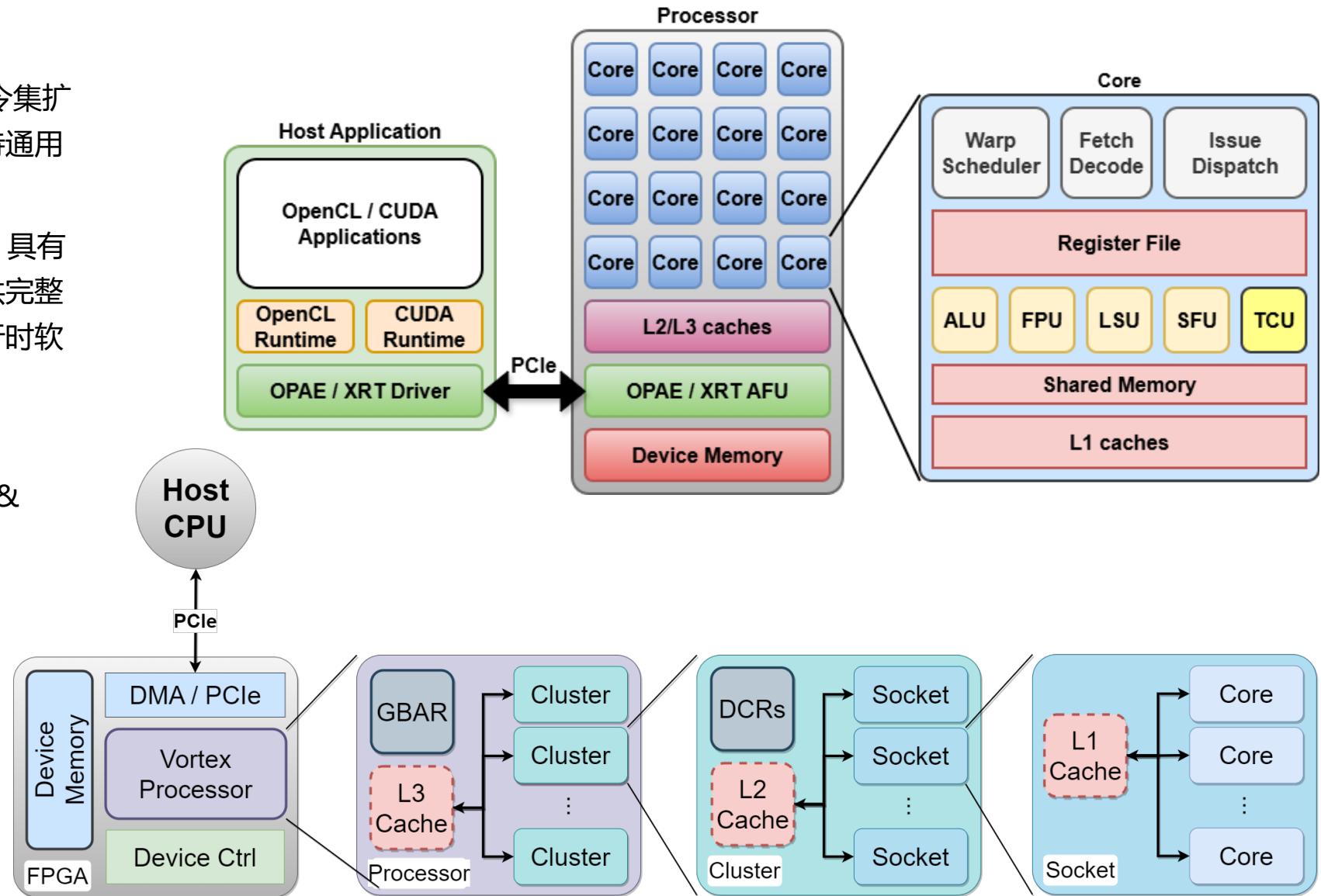
特殊功能单元 (SFU)

- Special Function Unit (SFU) 是 GPU (图形处理单元) 中的专用功能单元，主要用于执行特定数学运算，如三角函数、指数运算、对数运算等复杂数学处理
- SFU 通过硬件加速实现数学运算
 - 三角函数计算：支持正弦 (sin)、余弦 (cos)、正切 (tan) 等运算，广泛应用于图形渲染中的光照计算和物体表面反射
 - 指数与对数运算：用于物理模拟、复杂算法中的科学计算硬件架构
 - SFU 支持超越函数和平面属性插值的计算
- SFU 通常集成在 GPU 核心中，与流处理器 (SP) 协同工作，通过并行计算加速大规模数据处理



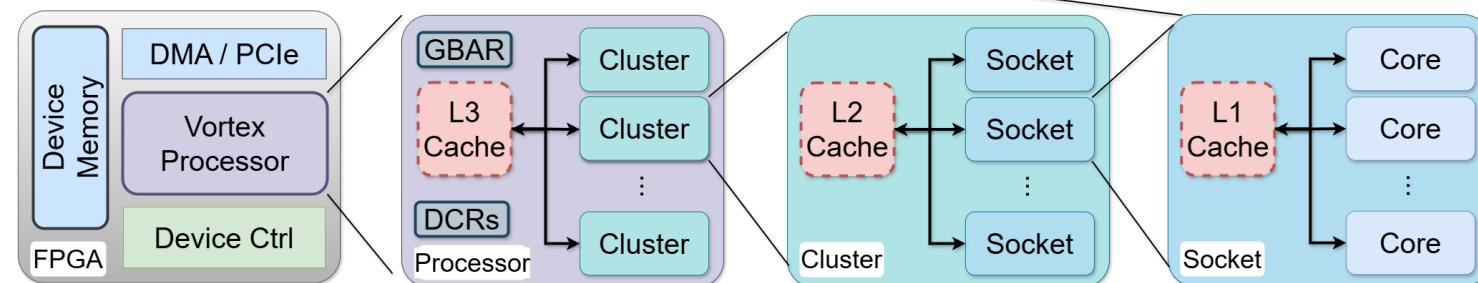
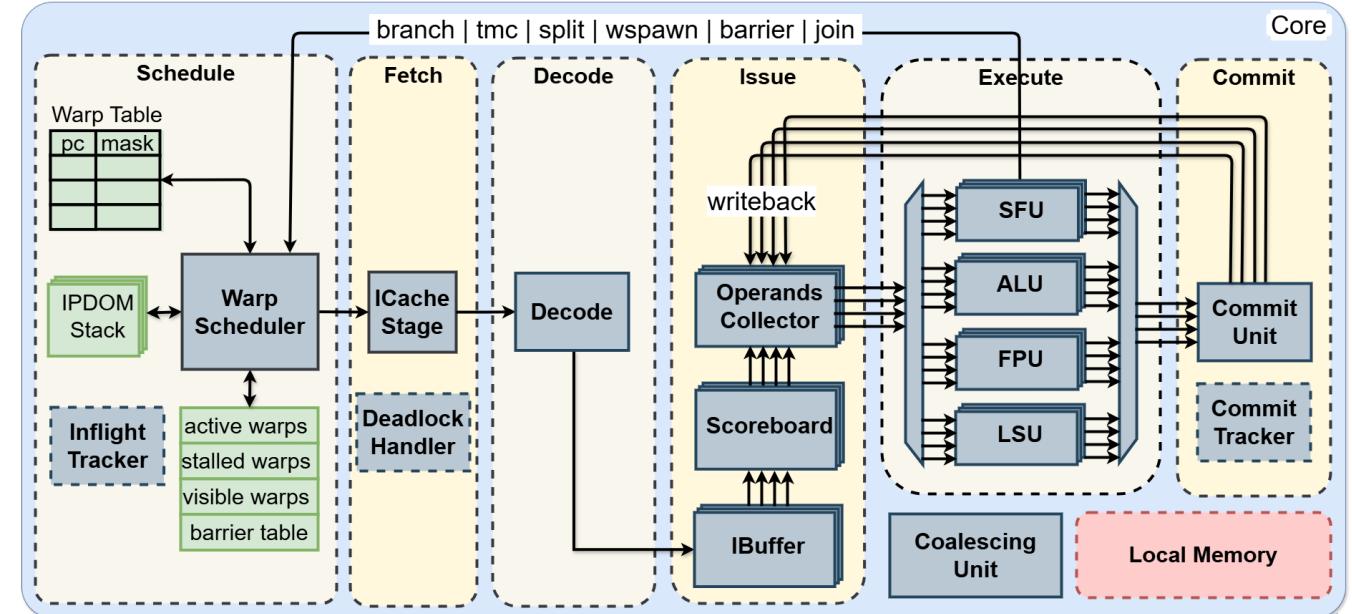
Vortex整体架构

- Vortex 是一个基于 RISC-V 指令集扩展的开源软硬件项目，旨在支持通用图形处理器 (GPGPU)
- Vortex 支持 OpenCL/ CUDA，具有高度可定制性和可扩展性，提供完整的开源编译器、驱动程序和运行时软件栈
- PCIe host-device interface
- Floating-point Units, caches & shared Memory
- 32 and 64-bit ISA
- LLVM: Compiler
- POCL: OpenCL
- CuPBoP: CUDA



Vortex流式处理器

- Core
 - 6级RISC-V架构顺序发射OOO (乱序) 执行
- Socket (Cores)
 - Multiple cores
 - Optional L1 cache
- Cluster
 - Multiple sockets
 - Optional L2 cache
- Processor
 - Global barrier
 - Optional L3 cache



Vortex执行级

1) Dispatch Queues

- Flow control
- Per execute unit

2) Dispatch Unit

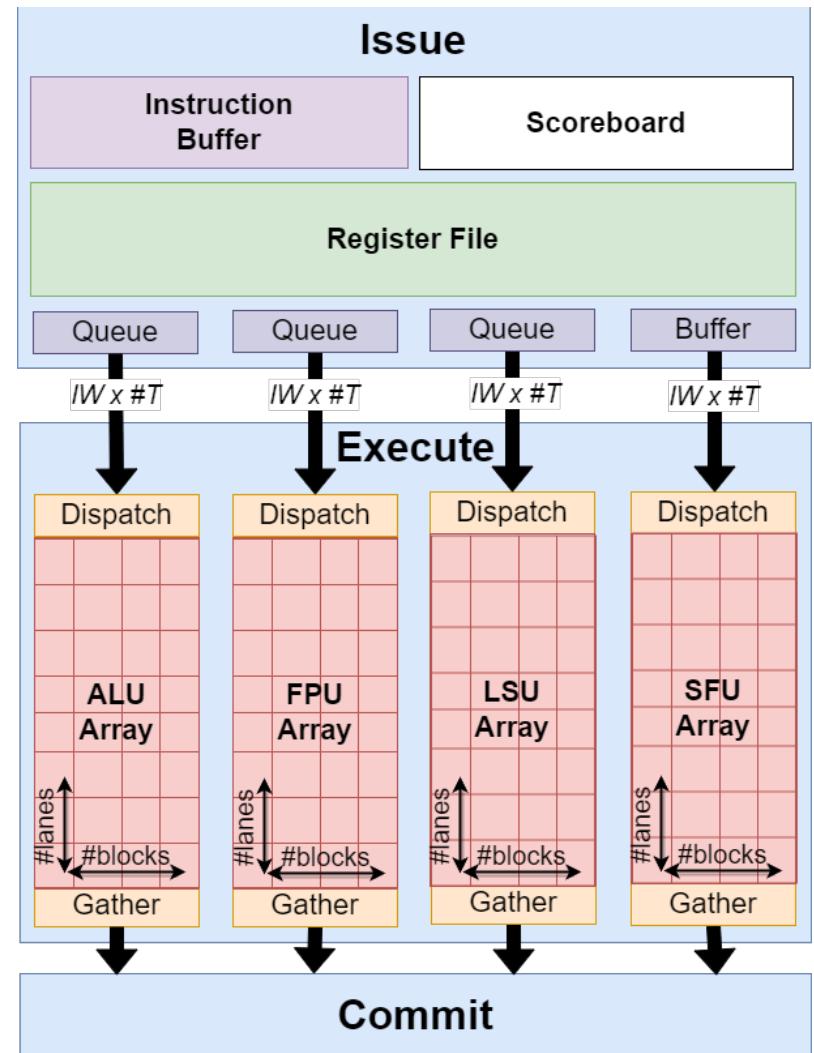
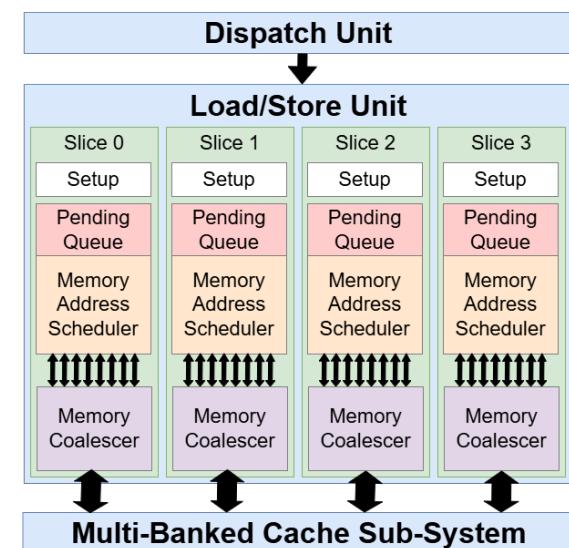
- Read $IW \times \#T$ data from queue
- Send $\#Block \times \#lanes$ data to array
- Per execute unit

3) Function Unit Array

- ALU/FPU/LSU/SFU

4) Gather Unit

- Merge outputs from array
- Return $IW \times \#T$ results
- Per execute unit



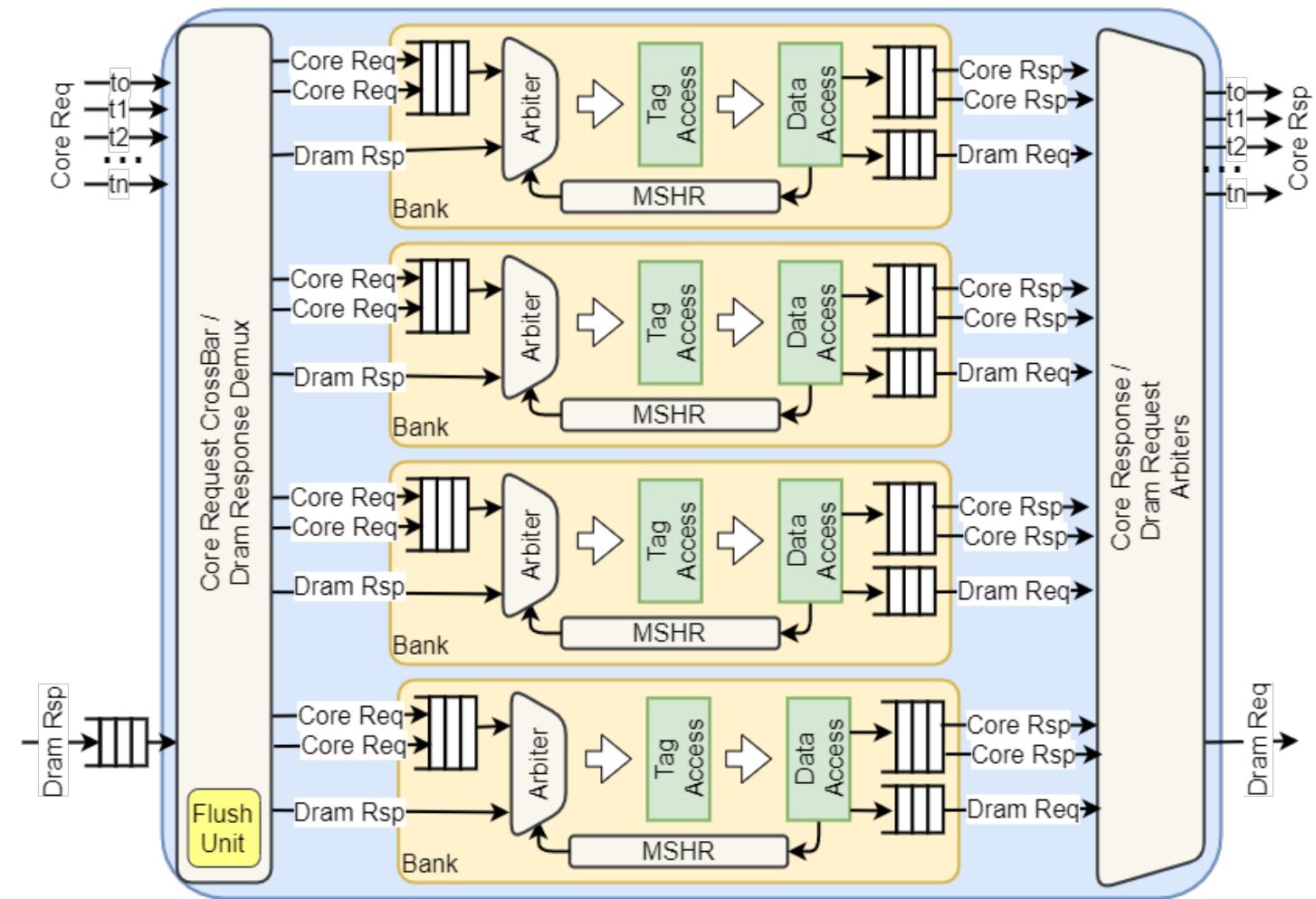
Vortex高速缓存

High-Performance

- Multi-banked
- Non-blocking operation

Highly Configurable

- Banking
- Associativity
- Write-back / Write-through
- Dirty bytes
- Eviction policy (内存淘汰机制)
- Hierarchical flush
- Pipeline queues



Vortex Tensor Core

乘加在人工智能算法中占据主要位置

Tensor Unit

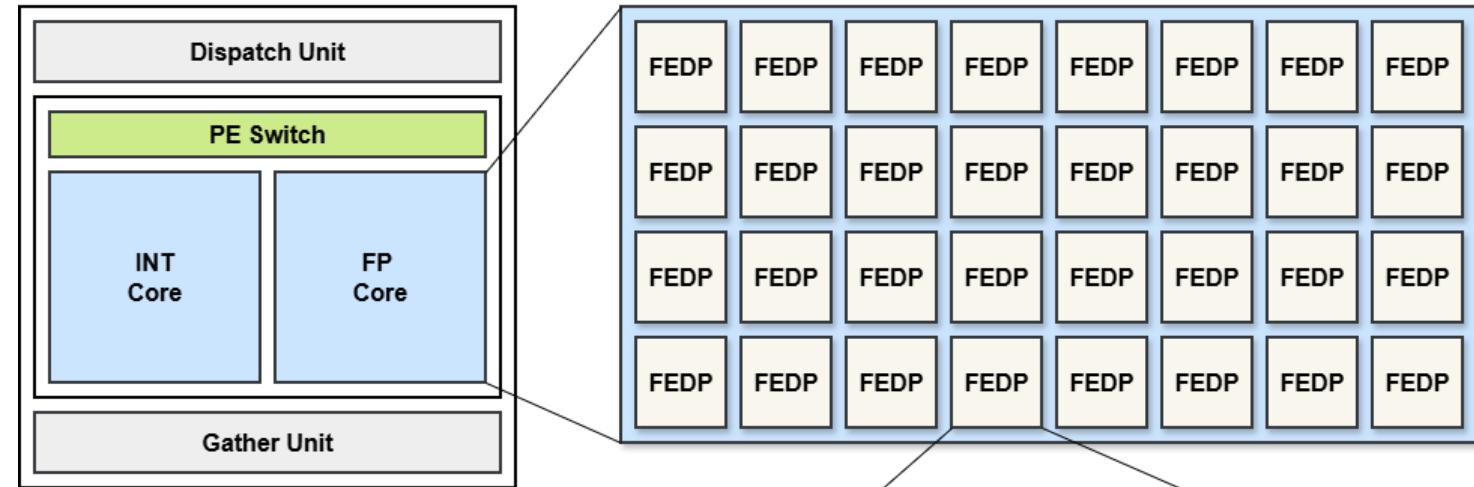
- Dispatch unit
- PE switch
- Integer Core
- Floating Point Core
- Gather unit

Fused-Element Dot Product

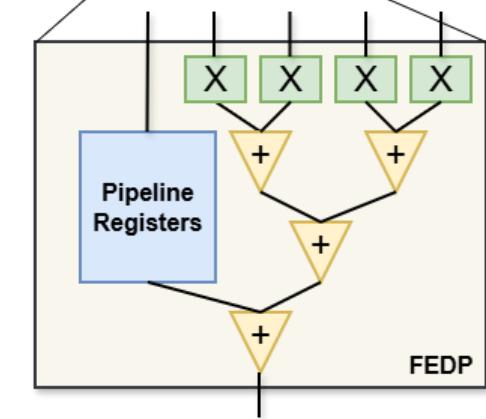
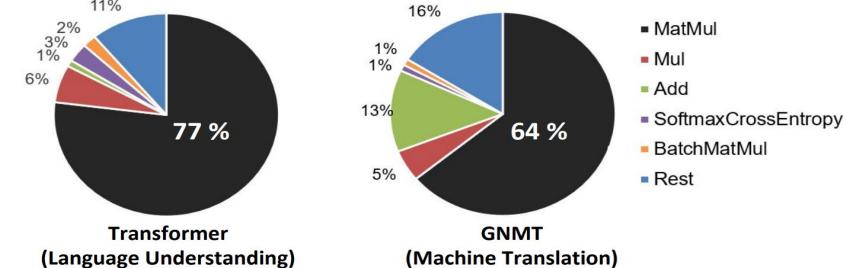
- Multipliers
- Adder Tree
- 4-cycle Pipeline (DPI, INT)

Mixed-precision formats

- TF32, Fp16, Int8, Int4
- 尚不支持FP8/NVFP4等关键格式

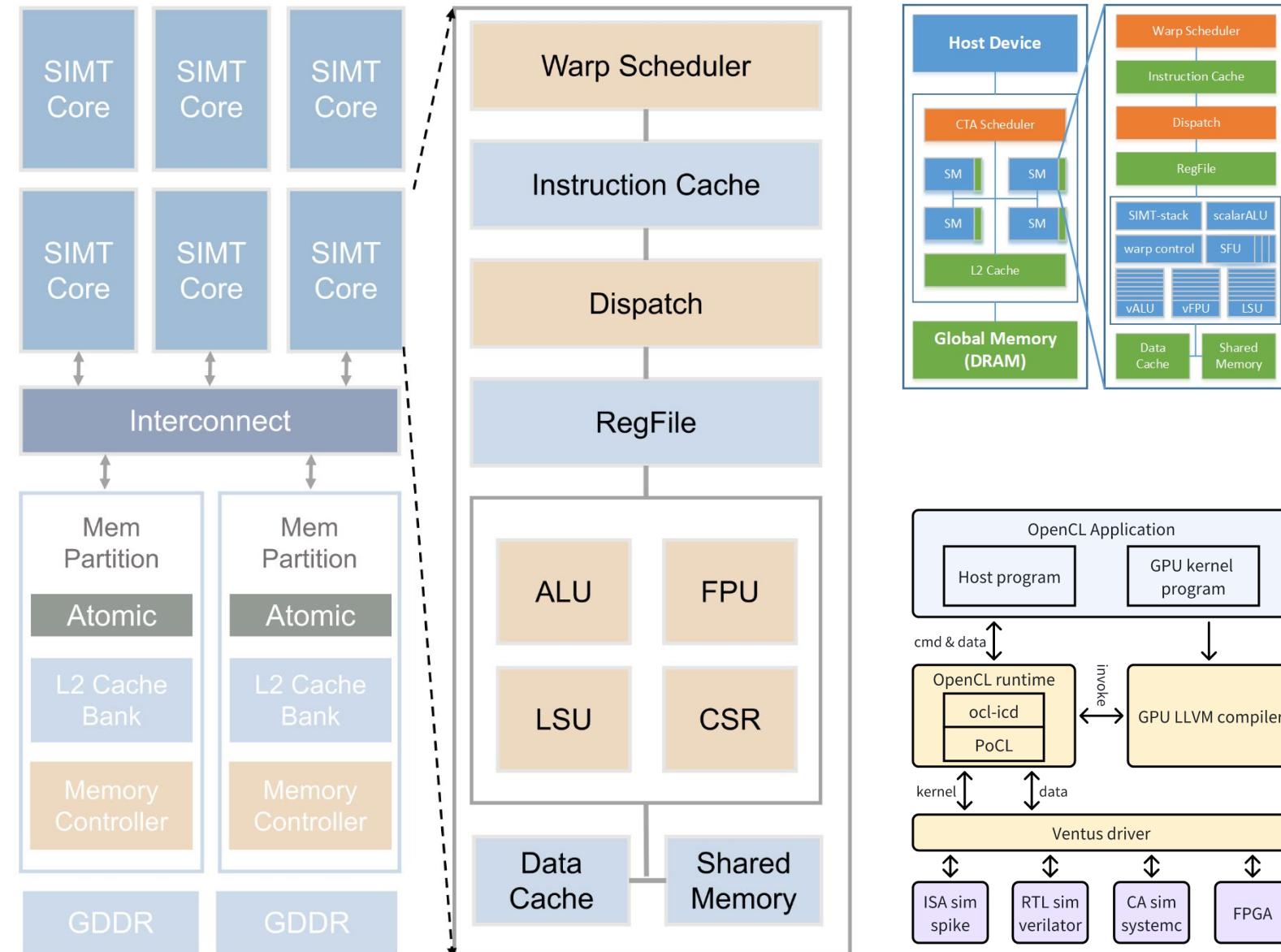


Runtime breakdown on V100 GPU



Ventus (乘影) 整体架构

- 乘影 GPGPU 指令集以 RISC-V 向量扩展（后文简称为 RVV）为核心设计 GPGPU，相比 RISC-V 标量指令，具有更丰富的表达含义，可以实现访存特性表征、区分 workgroup 和 thread 操作等功能。核心思想是在编译器层面以 v 指令作为 thread 的行为描述，并将 thread->warp/workgroup 的公共数据合并为标量指令
- 硬件上一个 warp 就是一个 RVV 程序，通常向量元素长度为 num_thread，同时又将 workgroup 中统一执行的公共地址计算、跳转等作为标量指令执行，即 Vector-Thread 架构
- 更详细内容见嘉宾分享



课程基本信息

- 课程名称：基于RISC-V的开源GPU架构与设计探索
- 授课地点：深圳市学苑大道南山智园C3栋20层2005（清华大学深圳国际研究生院）
- 授课时间：十周（每周六，9:00-12:00 14:00-18:00）
- 本课程聚焦RISC-V开源架构与GPU设计的**热门交叉前沿**，构建从指令集到众核计算架构的**探索性知识体系**，以最**短路径**实现RISC-V GPU体系架构入门，并探讨3D/3.5D Chiplet、开源EDA等技术在开源GPU中的应用潜力
- 课程核心内容涵盖：RISC-V指令集架构（ISA）与GPU并行计算架构的融合设计，开源GPU核心模块（如流处理器、存储架构、TensorCore），核心编译器的原理与探索实现。通过C-Model仿真与FPGA实现，学生将掌握从架构建模、功能验证到部署实现的精简芯片设计流程，并探索开源RISC-V GPU的大模型（Transformer）应用，为进入AI芯片/GPU编程等前沿领域的研究生学习或职业发展奠定**核心竞争力与先发优势**
- <https://github.com/chenweiphd/OpenRVGPUCourse>

OpenRVGPU Course

为培养下一代图灵奖获得者传递火种



清华大学深圳国际研究生院
Tsinghua Shenzhen International Graduate School

