



清华大学  
Tsinghua University

# Ventus OpenGPGPU 项目

DSPLAB

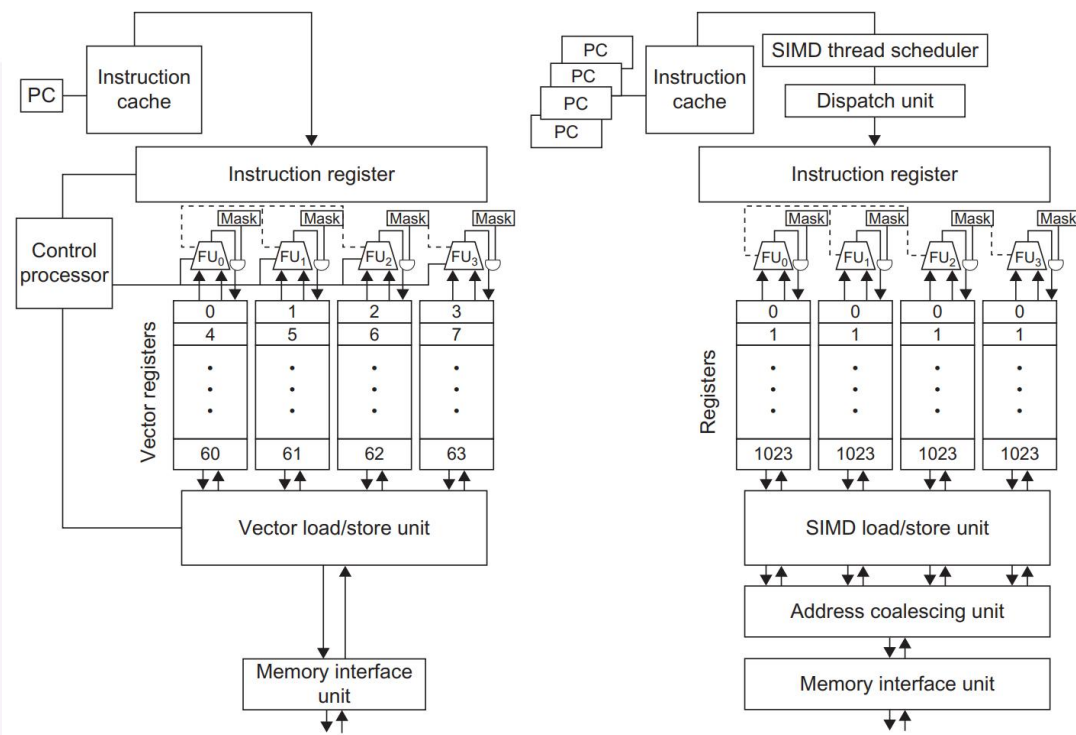
清华大学集成电路学院

**MICRO 2025**

58th IEEE/ACM International Symposium on Microarchitecture®

# 向量处理器与 GPGPU

- 向量处理器在计算、内存访问、分支、任务控制和寄存器堆栈方面与 SIMD 架构类似。
- 动机：与 SIMD 相比，GPGPU 提供了更好的可扩展性和编译器简洁性。
- 将向量通道映射到 GPGPU 线程

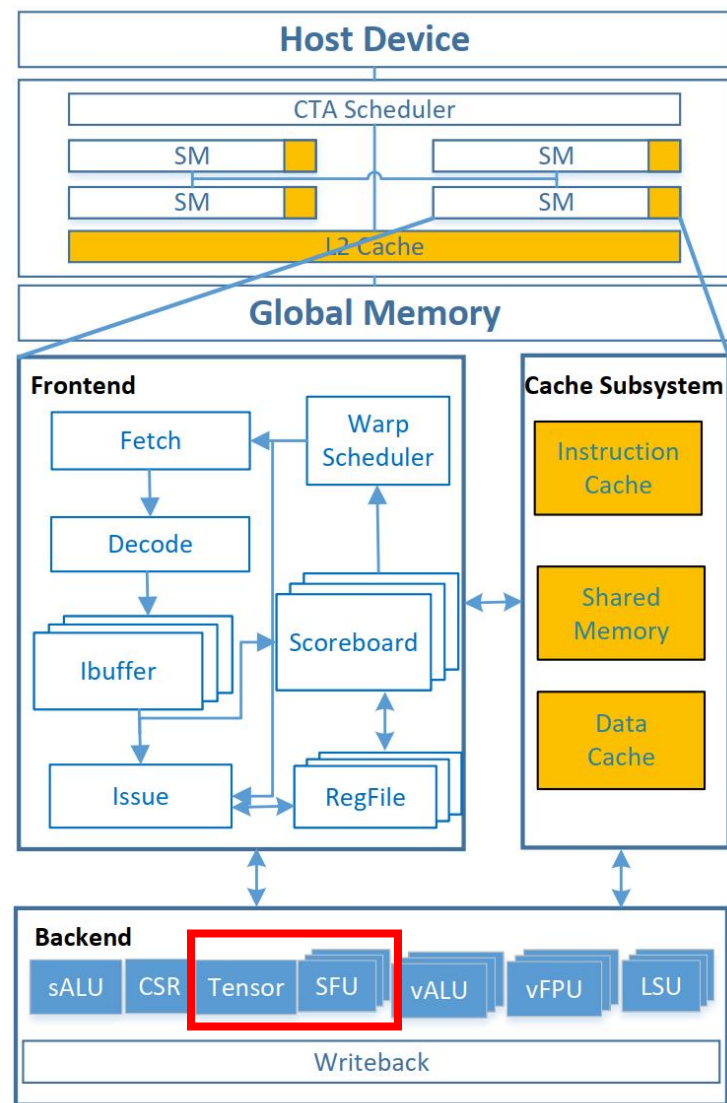


	Computation	Memory Access	Branch	Task control	Register	Architecture
Vector	Vector Lane	Gather/Scatter	Mask Registers	Control Processor	Vector Registers	Vector Processor
GPGPU	SIMD Lane	Global load/store	Predicate Registers	Thread Block Scheduler	SIMD Lane Registers	Multithreaded SIMD Processor

\* 《计算机体系结构：量化方法》

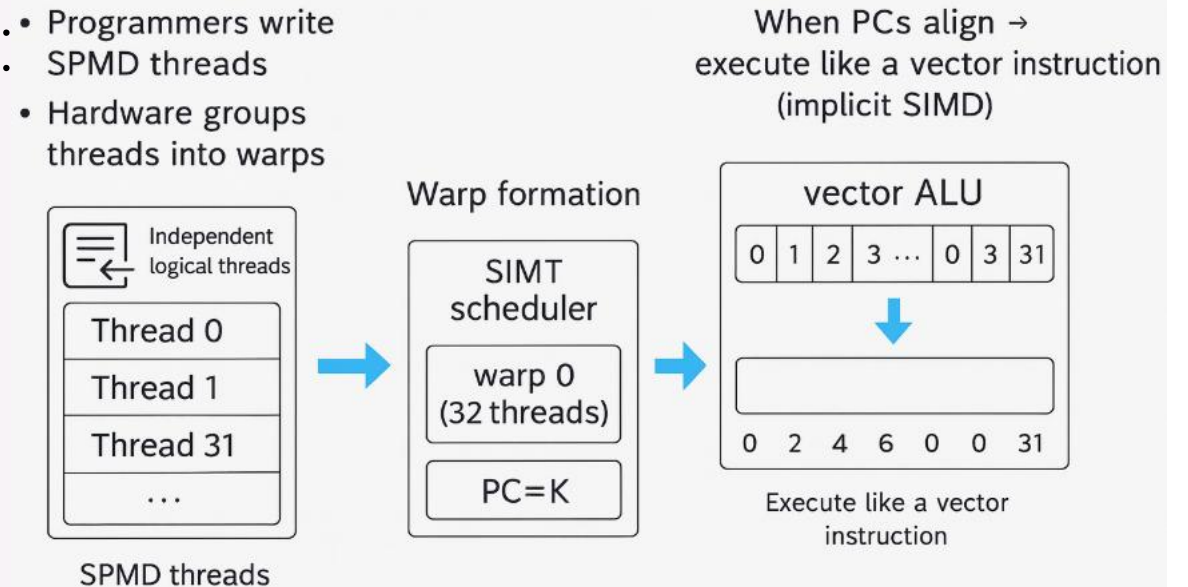
# Ventus GPGPU 设计

- Ventus GPGPU 选择 RISC-V 标量及其向量扩展作为基本 ISA。
- 将向量处理器的 ISA 转换为高性能 GPGPU 的理念源于向量处理器的 SIMD 通道与 GPGPU 数据路径之间的架构相似性。
- 基于 Chisel HDL 的开源高性能 GPGPU 硬件。
- 从 OpenCL 到 Ventus GPGPU ISA 的完整软件工具链。



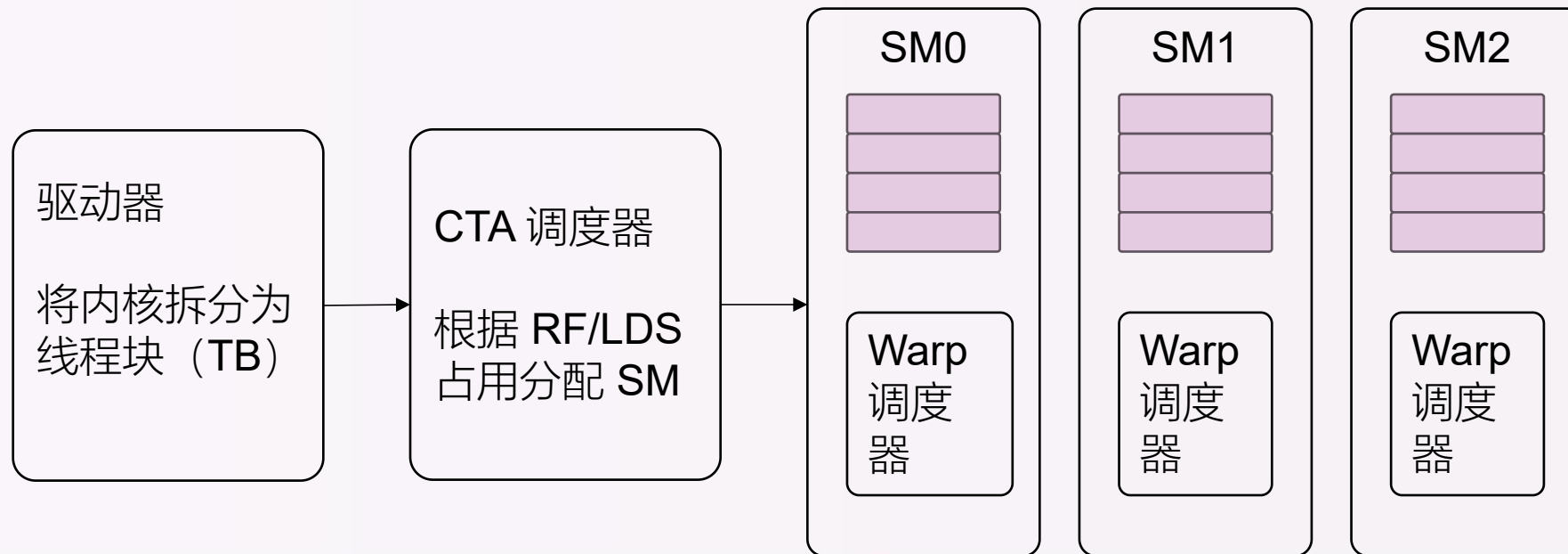
# 从 SIMD 到 SIMT (隐式 SIMD)

- 程序员编写 SPMD 线程
- 硬件将线程分组成 warp
- 当程序计数器对齐时 → 像向量指令一样执行 (隐式 SIMD)
- 将 Warp 视为 RISC-V V 程序
- 选择 Zve32f; 固定 VL=32, SEW=32



## 两级调度：线程块与 Warp

- 驱动器将内核拆分为线程块（TB）
- 线程块调度器根据 RF/LDS 占用分配 SM
- Warp 调度器处理启动、顺序、同步、退出以及 SIMT 栈



# Ventus GPGPU 指令集

EXT	Inst Type	1.0 Ver	2.0 Ver	3.0 Ver
V	Configuration-Setting	partially supported	partially supported	partially supported
	Loads and Stores	supported	supported	supported
	Integer Arithmetic	supported	supported	supported
	Floating-Point	supported	supported	supported
	Reduction	×	added per requirements	added per requirements
	Mask	partially supported	added per requirements	added per requirements
I		partially supported	supported	supported
M		supported	supported	supported
F		supported	supported	supported
D		×	×	supported
A		×	supported	supported
RV64		×	×	Supported via transformation

type	instruction name	usage
kernel response	endprg	endprg x0,x0,x0
synchronization	barrier, barriersub	barrier x0,x0,imm barriersub x0,x0,imm
branch control	vbeq, vbne, vbld, vbge, vbldu, vbgeu	vbeq vs2, vs1, offset vbne vs2, vs1, offset
branch control	join, setrpc	join v0, v0, 0 setrpc rd, rs1, offset
register index extension	regext, regexti	regext x0, x0, imm regexti x0, x0, imm
register pair	regpair, regpairi	regpair x0, x0, imm regpairi x0, x0, imm
memory access	vlw12.v, vlh(u)12.v, vlb(u)12.v vsw12.v, vsh12.v, vsb12.v	vlw12.v vd,offset(vs1) vsw12.v vd,offset(vs2)
memory access	vlw12d.v, vlh(u)12d.v, vlb(u)12d.v vsw12d.v, vsh12d.v, vsb12d.v	vlw12d.v vd,offset(vs1) vsw12d.v vd,offset(vs2)
async memory access	cp_dma, cp_dma_bulk, cp_dma_tensor, cp_dma_mbarrier	cp_dma cpsize cp_dma_bulk srcsize, src, dst
prefix	pre_default, pre_defaulti	pre_default imm, pair, abs, neg pre_defaulti imm, pair, abs, neg
prefix memory	pre_m (size), pre_m_global (size), pre_m_private (size), pre_m_local (size)	pre_m_32 ch, imm, pair pre_m_global_64 ch, imm, pair pre_m_local_128 ch, imm, pair
calculate	vadd12.vi	vadd12.vi vd,vs1,imm
tensor	vfexp.v, vftta.v, mma	vfexp vd,v2,v0.mask mma_8x8x8_FP32_FP32 vd, vs2, vs1

## 自定义指令

分支、同步和 Warp 控制 — SIMT

寄存器/立即数扩展 — 减少寄存器溢出

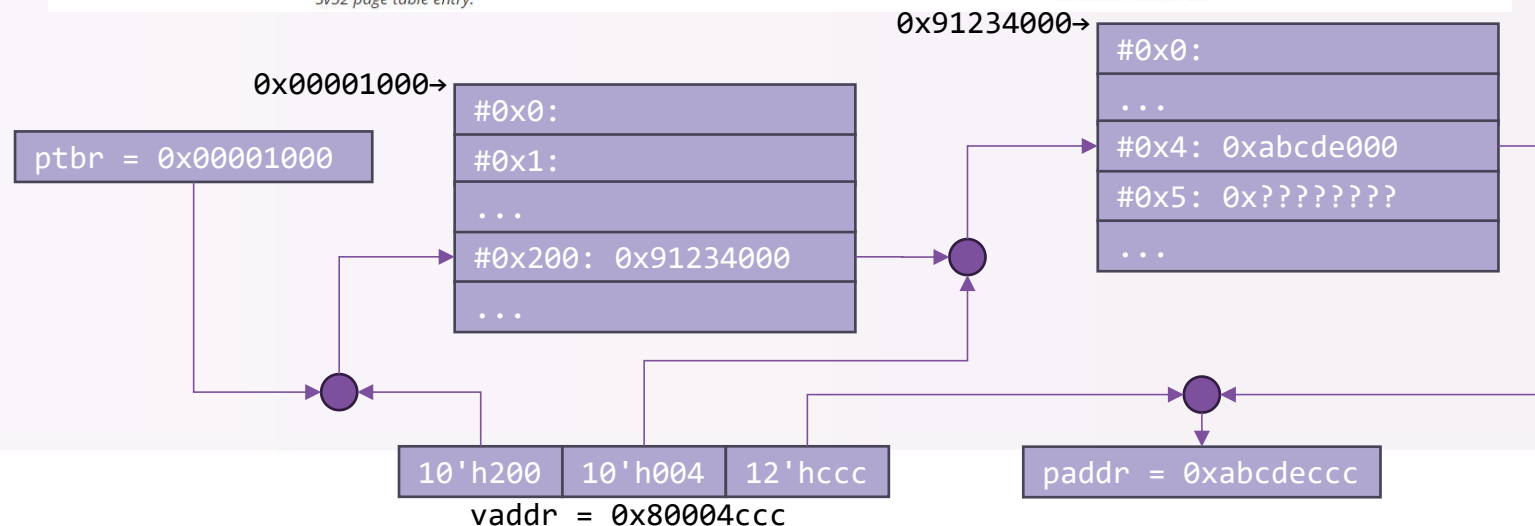
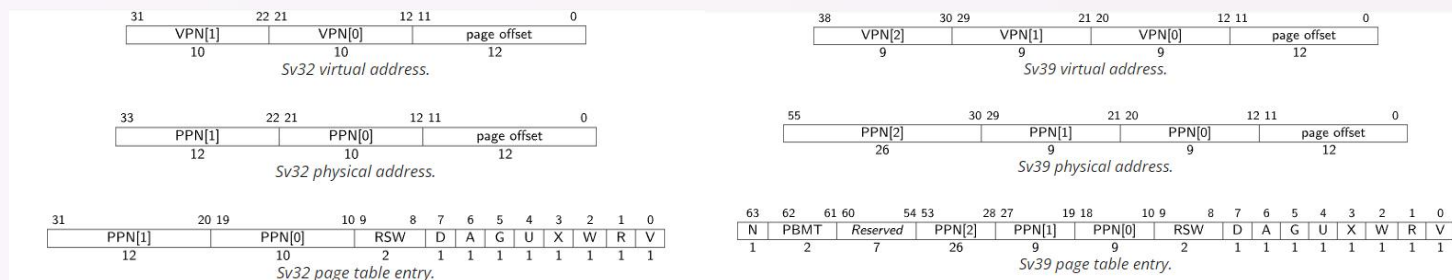
寄存器对拼接 — 实现 64 位运算

自定义内存访问指令 — 满足编译器需求

张量运算和指数运算 — 由 DSA 支持

# 虚拟内存：Sv32/Sv39 规范

- RISC-V 定义了多种不同长度的虚拟地址和不同的页表系统
- Sv32：虚拟地址 32 位 / 物理地址 34 位，两级页表（10+10）
- Sv39：虚拟地址 39 位 / 物理地址 56 位，三级页表（9+9+9）



# Ventus GPGPU 缓存一致性

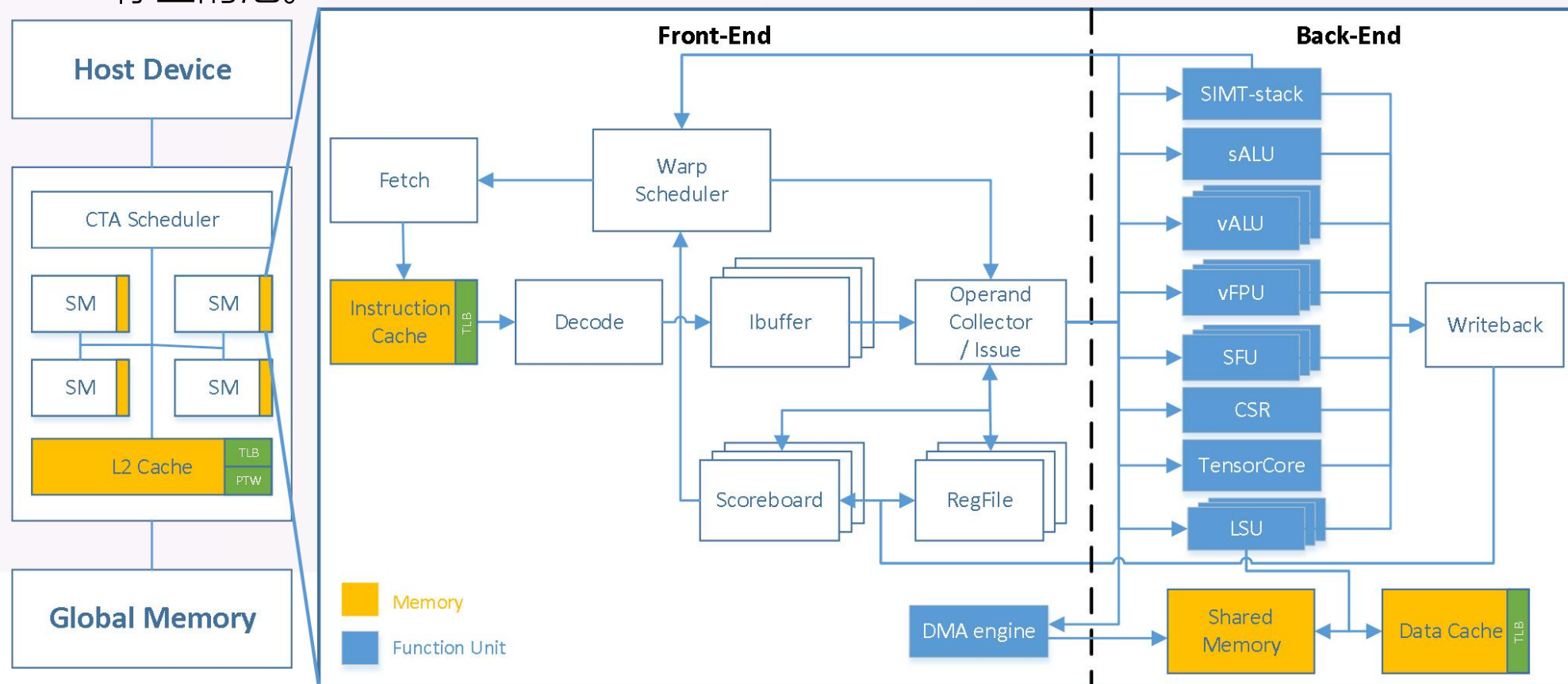
- 基于 RISC-V 弱内存模型 (RVWMO) , Ventus 采用释放一致性指导的缓存一致性 (RCC) 。这使得 SM (流处理器) 私有缓存之间保持一致, 同时避免了完整硬件一致性协议的高硬件复杂性和运行时带宽开销。原子指令按每线程行为定义。



Microarchite ctural action	(4) global invalidation	(3) global invalidation	(4) global invalidation
RVWMO marker	.aq qualifier	.rl qualifier	FENCE
RCC coherence operation	“acquire” and “release”	“release”	“acquire” and “release”

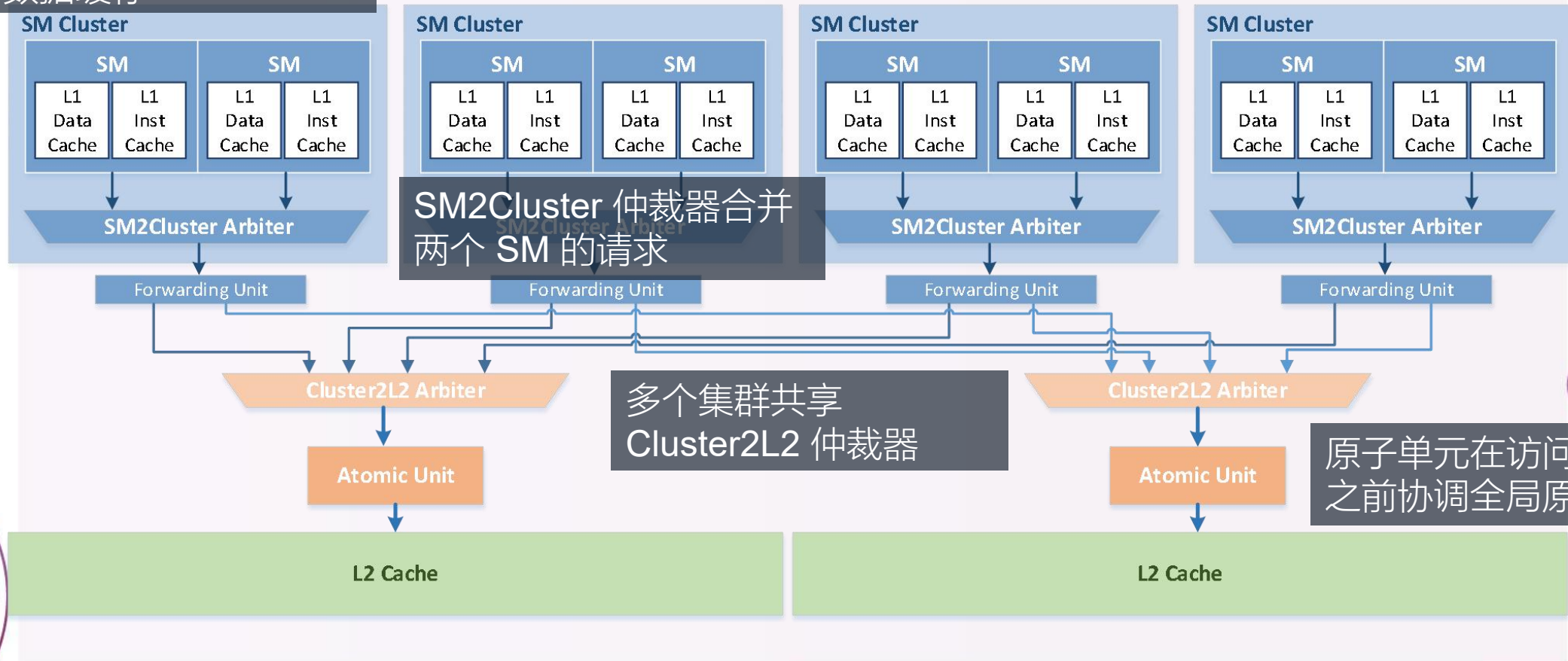
# SIMT 流水线概述

- 经典 SM 流程：取指 → 解码 → 指令缓冲 → 记分板 → 调度 → 操作数收集 → 发射 → 执行 → 内存 → 回写。
- 每个 inflight 操作都带有 WID 标签；前端具备 SIMT 感知；后端将操作视为带有 WID 标签的池。



# Ventus 缓存子系统概述

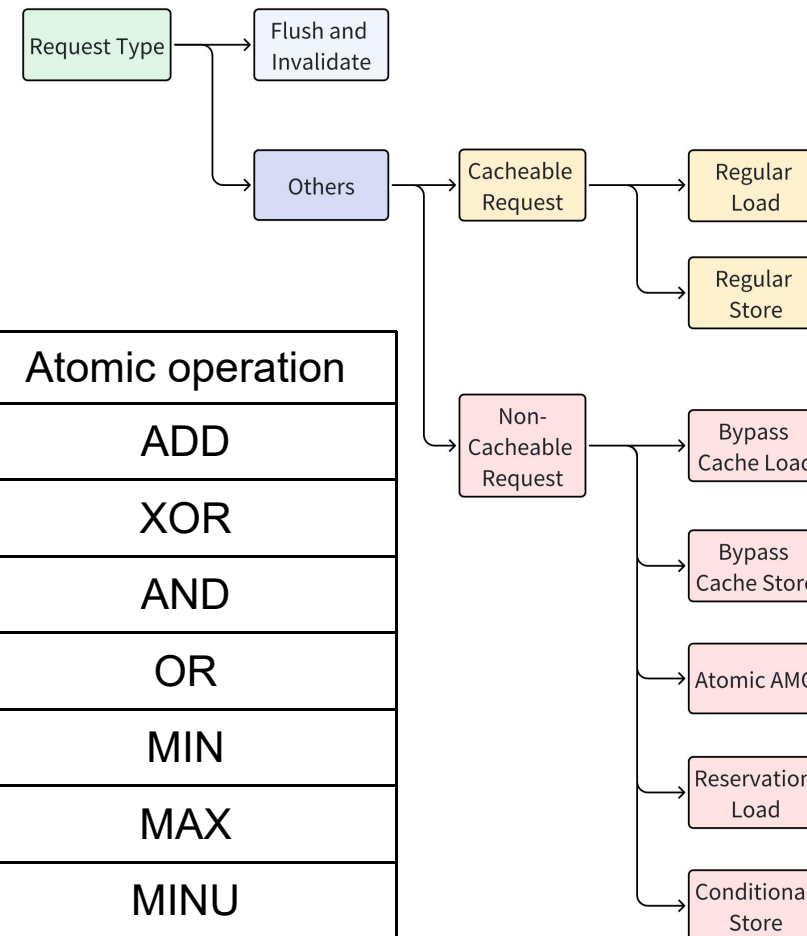
每个 SM 拥有私有 L1 指令/数据缓存



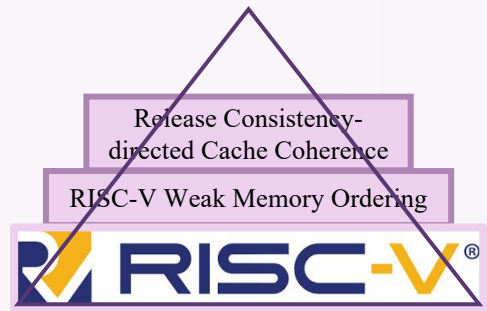
# L1 数据缓存支持的操作

Opcode	Parameter	Operation
0	0	Normal load
0	1	Load-Reserved (LR)
0	2	Uncached load (bypass L1)
1	0	Normal store
1	1	Store-Conditional (SC)
1	2	Uncached store (bypass L1)
2	0-7	Atomic AMO
3	0	Global invalidate (L2)
3	1	Global flush (L2)
3	2	Wait until MSHR empty
3	3	L1 invalidate
3	4	L1 flush

Parameter	Atomic operation
0	ADD
1	XOR
2	AND
3	OR
4	MIN
5	MAX
6	MINU
7	MAXU



# 在 Ventus 中实现 PPO



RVWMO

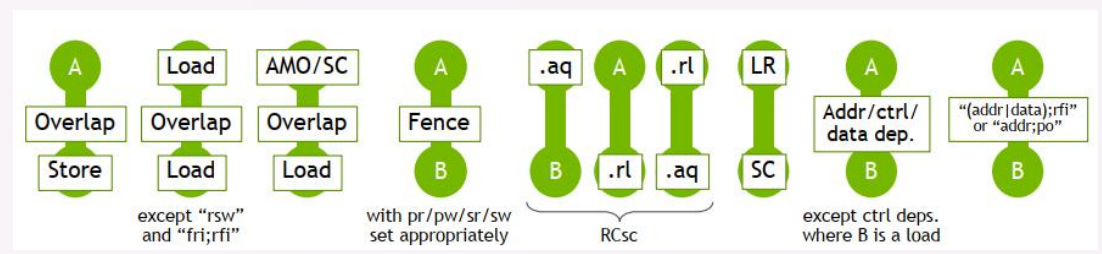
RV32I FENCE

31	28	27	26	25	24	23	22	21	20	19	15	14	12	11	7	6	0
fm	PI	PO	PR	PW	SI	SO	SR	SW	rs1	funct3	rd	opcode					
4	1	1	1	1	1	1	1	1	5	3	5	7					
FM predecessor successor									0	FENCE	0	MISC-MEM					

RV32A AMO

31	27	26	25	24	20	19	15	14	12	11	7	6	0
funct5	aq	rl	rs2	rs1	funct3	rd	opcode						
5	1	1	5	5	3	5	7						
AMOSWAP.W/D	ordering	src	addr	width	dest	AMO							
AMOADD.W/D	ordering	src	addr	width	dest	AMO							
AMOAND.W/D	ordering	src	addr	width	dest	AMO							
AMOOR.W/D	ordering	src	addr	width	dest	AMO							
AMOXOR.W/D	ordering	src	addr	width	dest	AMO							
AMOMAX[U].W/D	ordering	src	addr	width	dest	AMO							
AMOMIN[U].W/D	ordering	src	addr	width	dest	AMO							

RISC-V 中用于内存一致性和同步的指令编码

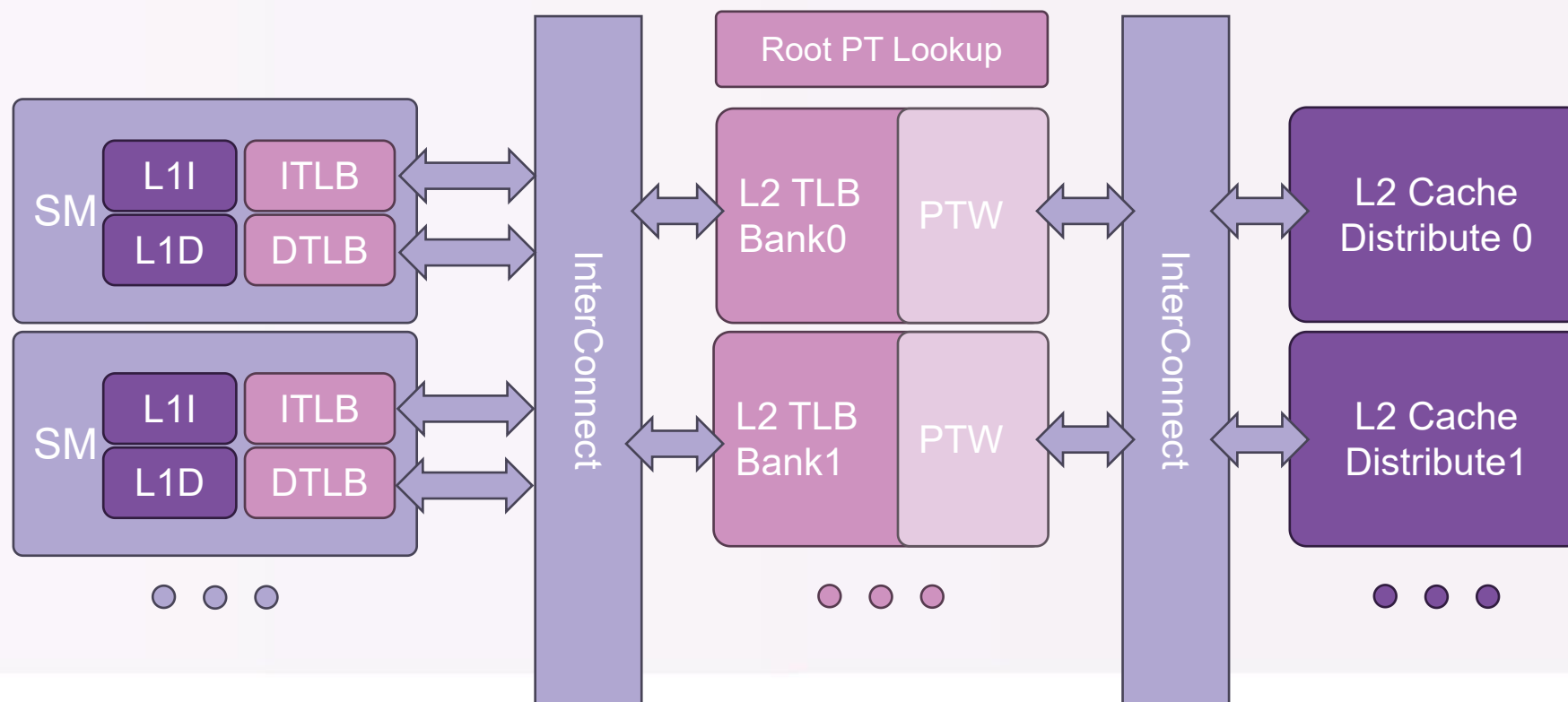


PPO 规则的可视化解释

- 缓存微架构设计
- 使用缺失状态保持寄存器（MSHR）和写入状态保持寄存器（WSHR）来解决 PPO1 和 PPO2
- PPO4–7 对应于诸如 Fence 和 acquire/release 等一致性操作，这些通过诸如清空 MSHR、全局刷新和全局失效等微架构机制支持。
- PPO9–13 已在流水线中解决，而 PPO2 和 PPO8 通过在缓存流水线中插入额外检查来处理

# 虚拟内存：架构设计

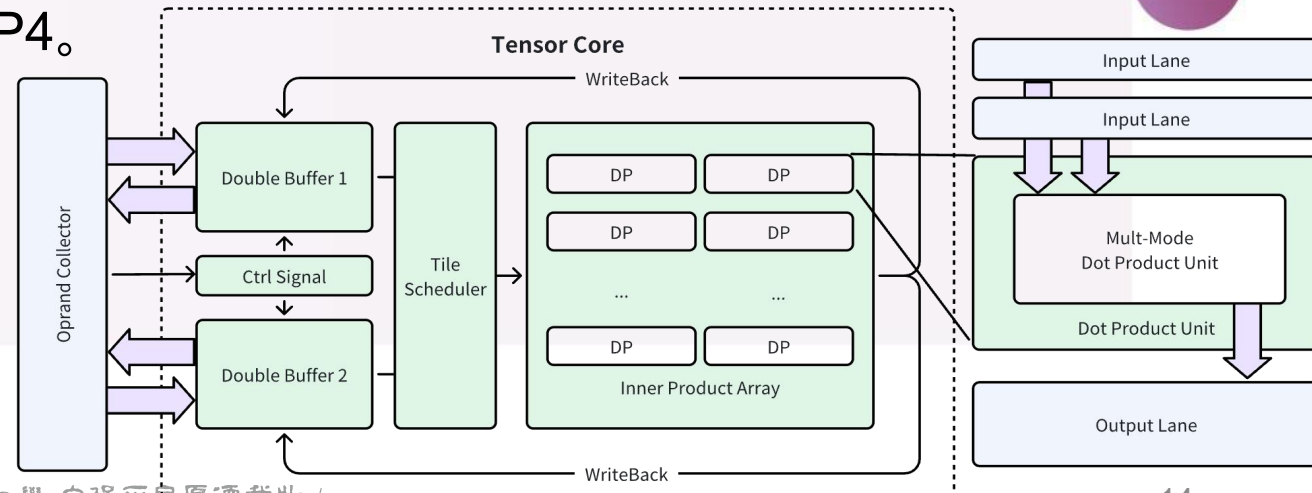
- L1 缓存采用虚拟索引|虚拟标签（VIVT）；L2 保持物理索引|物理标签（PIPT）
- 采用分层交叉开关互连进行访问



# Ventus 张量核概述

新版本的 Ventus 具备多精度张量核特性，具体如下：

- 计算规模扩展：
  - FP16 矩阵乘法计算规模扩大到  $16 \times 16 \times 16$ 。
  - 低位精度计算规模随 k 维度自然扩展。
- 计算资源复用：
  - FP16/INT8/INT4 精度点乘单元复用硬件资源。
- 计算精度提升：
  - 细粒度量化：支持 OCP MX-FP8/FP6/FP4。
  - 低位浮点支持：支持 FP8/FP6/FP4。
- 支持稀疏计算



# 多精度

- 支持多精度矩阵乘累加（MMA）计算：
- FP16：支持混合精度并在 FP16 精度下累加。
- INT8/INT4：与 FP16 共享点乘单元，在 INT32 精度下累加。
- FP8/FP6/FP4：支持在 FP32 精度下累加。
- MX FP8/FP6/FP4：支持符合 OCP MX 标准的细粒度量化因子。

[1] "IEEE Standard for Floating-Point Arithmetic," in IEEE Std 754-2019 (Revision of IEEE 754-2008) , vol., no., pp.1-84, 22 July 2019.

[2] Google, “The Bfloat16 Numerical Format”, <https://cloud.google.com/tpu/docs/bfloat16>.

[3] Micikevicius, Paulius, et al., “OCP 8-bit Floating-Point Specification”, June 2023

[4] Bitar Darvish Rouhani, et al., “OCP Microscaling Formats (MX) Specification”, Sep 2023

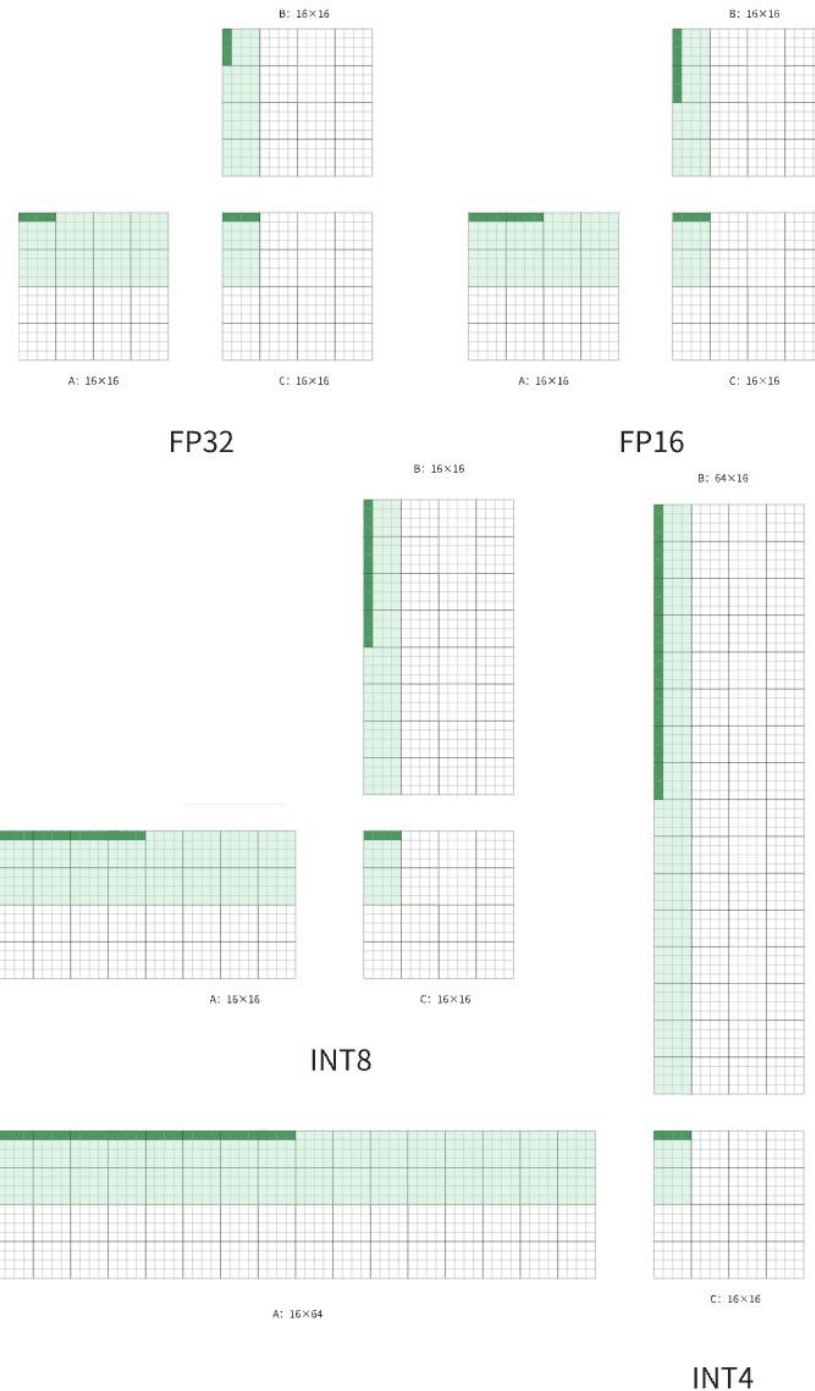
Type	Sig n	expore nt	mati ssa	Total	Description
FP16	1	5	10	16	From IEEE 754 standard.
BF16	1	8	7	16	A new data type introduced by Google in TensorFlow, commonly used in deep learning training. It better matches the distribution of neural network weights and addresses the issue of FP16's limited expressiveness range during model training.
FP8 E4M3	1	4	3	8	Defined in September 2022 by NVIDIA, Arm, and Intel, FP8 offers a more nonlinear representation range compared to INT8.
FP8 E5M2	1	5	2	8	
INT8	1	N/A	7	8	Quantization precision commonly used in the industry
FP6 E3M2	1	3	2	6	Based on the OCP MX standard
FP6 E2M3	1	2	3	6	
FP4 E2M1	1	2	1	4	
INT4	1	N/A	3	4	Quantization precision.

# 多尺寸

- 支持多尺寸矩阵乘累加（MMA）计算：
- FP32/FP16/FP16 混合精度/TF32/BF16/INT8/INT4：支持 m16n16k16、m32n8k16、m8n32k16。
- INT8/FP8：支持 m16n16k32、m32n8k32、m8n32k32。
- INT4：支持 m16n16k64、m32n8k64、m8n32k64。
- 对于 INT8/INT4/FP8 等精度，阵列支持在 k 维度上的形状扩展。
- NVIDIA 2:4 稀疏加速支持使 k 维度进一步扩展，理论峰值性能翻倍。

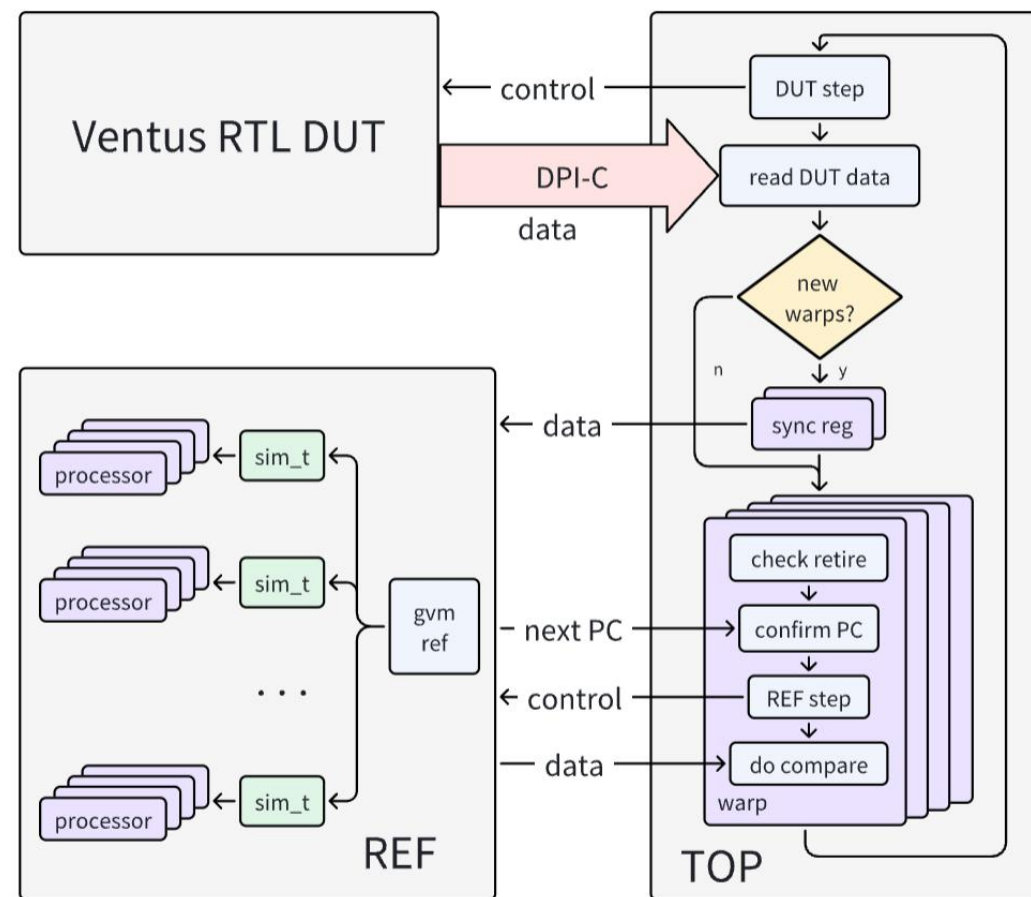
[1] A. Mishra et al., “Accelerating Sparse Deep Neural Networks,” Apr. 16, 2021, arXiv: arXiv:2104.08378. doi: 10.48550/arXiv.2104.08378.

[2] C. Zhang et al., “DSTC: Dual-Side Sparse Tensor Core for DNNs Acceleration on Modern GPU Architectures,” IEEE Trans. Comput., pp. 1–14, 2024, doi: 10.1109/TC.2024.3475814.



# GPU 验证模型 (GVM)

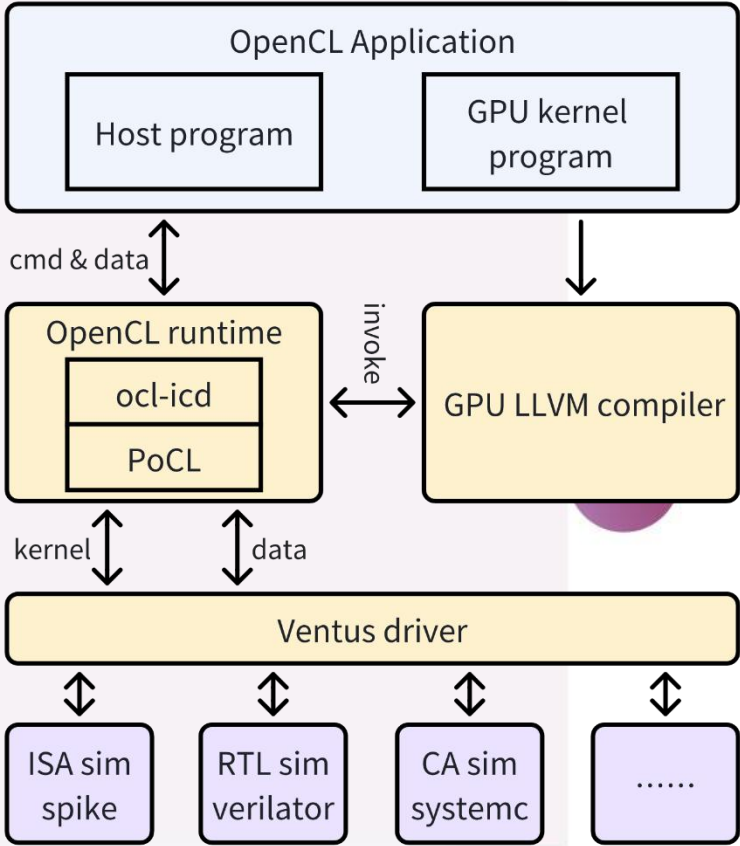
- GVM 是一种针对 Chisel 或 Verilog 开发的 GPGPU 的类 UVM 验证模型
- 设备每个周期运行一步，由顶层仿真程序捕获指令派发、完成等事件。
- 为了解决 GPGPU 缺乏硬件 ROB（与 CPU 不同）并使 DUT 与参考模型的同步复杂的问题，构建了一个 Warp 级软件 ROB。
- GVM 帮助用户高效调试软件和硬件。



# 仿真框架更新和工具链集成

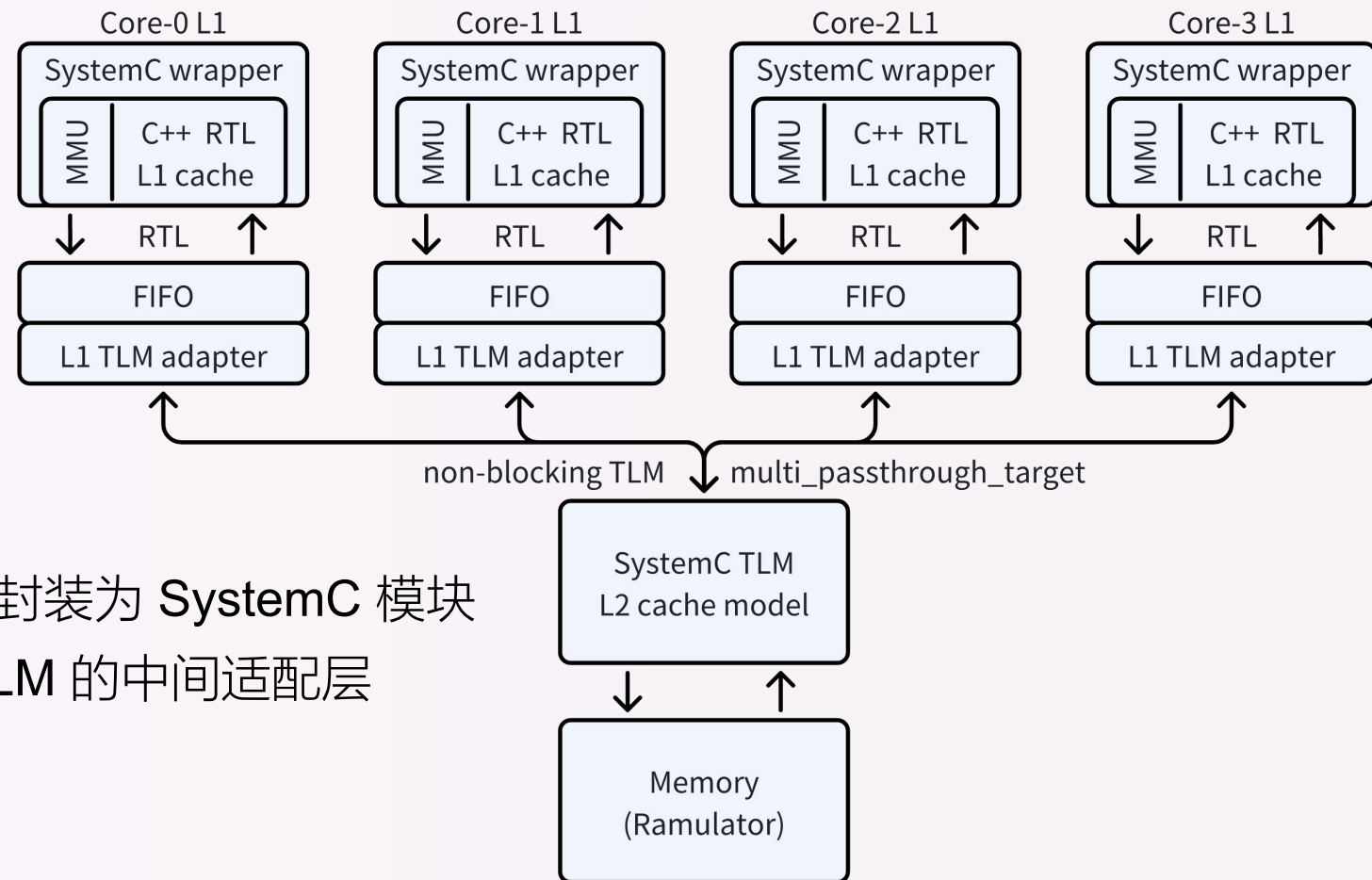
- 在 Ventus 工具链中集成 Chisel RTL Verilator 仿真框架和周期级模拟器
  - 定义仿真框架 API 并打包动态库
  - 虚拟内存管理驱动：SV32/SV39
  - 多个仿真后端采用统一调用接口，方便切换
- 
- 优势：
  - 无需手动生成测试用例中间文件，简化仿真流程
  - 能够验证 OpenCL 程序的结果正确性
  - 提供一键部署脚本 + 回归测试脚本
  - 相比 chiseltest 方案（RTL），显著提高仿真效率
  - 避免重复编译，允许多线程并行仿真

```
VENTUS_BACKEND=spike      ./a.out # Using the Spike
Instruction-Level Simulator
VENTUS_BACKEND=rtlsim     ./a.out # Using Chisel RTL
Simulation
VENTUS_BACKEND=cyclesim   ./a.out # Using cycle-level
Simulator
```



# 基于 SystemC 的周期级模拟器

- 缓存系统建模
- 原子操作支持
- 命中和延迟统计



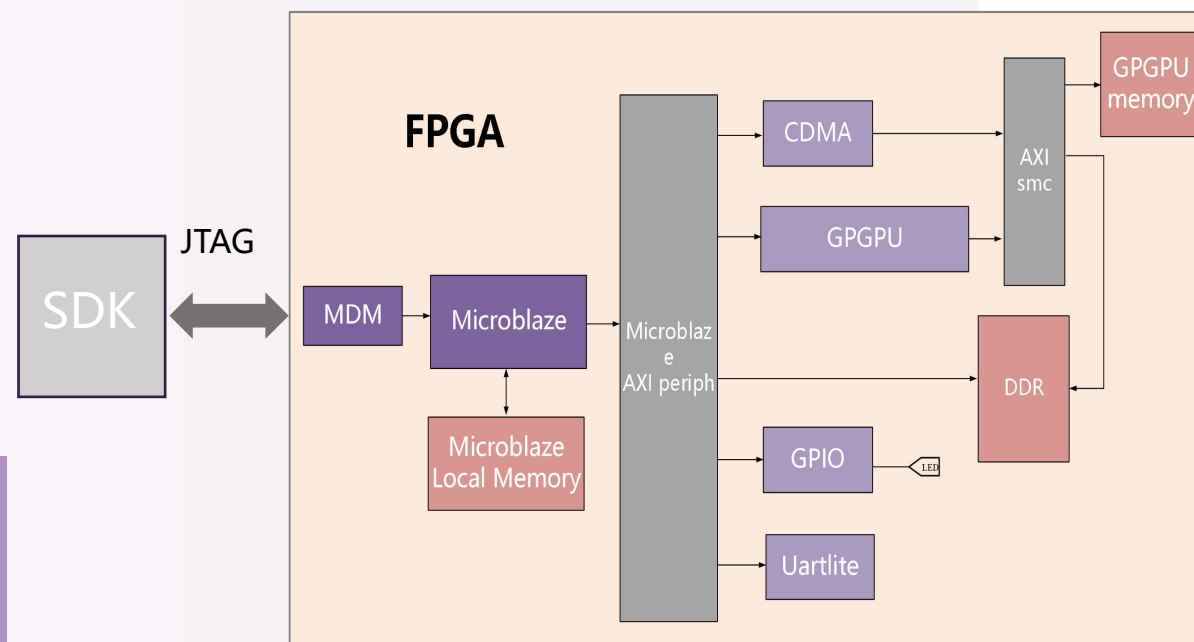
- L1: C++ RTL 建模，封装为 SystemC 模块
- 适配器：从 RTL 到 TLM 的中间适配层
- L2: 事务队列建模
- MMU: 功能级建模

# Ventus-Community: FPGA 测试架构

- 控制核心：运行测试逻辑的 MicroBlaze 处理器
- 调试与交互接口：通过 SDK 将 .elf 文件上传到 MicroBlaze 内存以控制外围设备
- 内存子系统：AXI 接口 DDR，用于大容量数据存储

该平台采用 MicroBlaze 软核处理器管理外围设备（GPGPU、DDR），提供 JTAG 调试和 AXI 总线通信，便于硬件加速任务的开发和测试。

## Virtex Ultrascale + HBM VCU128



<https://opengpgpu.org.cn/>



代码仓库

<https://github.com/THU-DSP-LAB>

<https://www.gitlink.org.cn/THU-DSP-LAB>



GitHub



GitLink

MICRO 2025

58th IEEE/ACM International Symposium on Microarchitecture®

# THANK YOU



OpenGPU  
乘影



清华大学教育基金会  
TSINGHUA UNIVERSITY EDUCATION FOUNDATION