TensorCore1 设计规范（Design Spec）

- 0 设计需求
- 1 全局参数及模块列表
  - 1.1 全局参数
  - 1.2 模块列表
- 2 顶层
  - 2.1 顶层架构
  - 2.2 tensor_core 顶层模块
- 3 格式转换模块
  - 3.1 to_fp9_con 模块
  - 3.2 to_next_con 模块
- 4 张量计算模块
  - 4.1 mm_mul_add 模块
  - 4.3 mv_mul 模块
  - 4.4 tc_mul 模块
  - 4.5 tc_add 模块
- 5 数据通道模块
  - 5.1 tc_mul 模块
  - 5.2 tc_add 模块
  - 5.3 tc_mm_add 模块
- 6 基本计算模块
  - 6.1 fmul_s1 模块
  - 6.2 naivemultiplier 模块
  - 6.3 fmul_s2 模块
  - 6.4 fmul_s3 模块
  - 6.5 fadd_s1 模块
  - 6.6 fadd_s2 模块
  - 6.7 far_path 模块
  - 6.8 near_path 模块
- 7 舍入模块
- 8 辅助计算模块
  - 8.1 shift_right_jam 右移模块
  - 8.2 lza 前导 0 预测模块
  - 8.3 lzc 前导 0 计数模块
- 9 存储模块 SRAM
  - 9.1 SRAM(FP9)模块
  - 9.2 SRAM(FP4-FP22)模块
  - 9.3 SRAM(L1)模块

修改建议

1. 有些模块用了 axistream 接口，有些没用。这个是有问题的，至少要保证在两级 FIFO 或 SRAM 之间的模块接口保持一致。因为 stream 和非 stream 接口之间的转换必须借助 FIFO 才能实现。
2. tc_mul 是 A 的多个列（行）同时对应 B 的一行（列），这里就存在 A 需要多次读取的问题，那么多次读取前面的 SRAM 就不能是 FIFO，因为它需要一次写入，多次读取。这里需要有一个专用的类似多维 DMA 的电路来实现。接口可以设计成流式，但是这一级的 SRAM 不是 FIFO。
3. 文档里的 6 的基本计算模块都是组合逻辑么？

4. 2.2 主要更改点

in_valid 由 a/b/c 三个 valid 和三个 last 相与，out_ready 由三个 ready 相与（按照 AXI Stream 协议处理）

在顶层不能保证 a，b，c 数据同时到达，现实中是无法做到在顶层模块做到的。在 tensorcore 外没法同步 a,b,c 他们应该是串行的。这个同步应该在内部的第一级 RAM/FIFO 输出做。

本文按照 tensorcore 自顶向下的次序，对设计要求进行阐述

0 设计需求

设计需求如下：

1）兼容现有 gpgpu-sim 里的函数接口模式，方便编译器开发者与开源社区为该 TensorCore 适配 CUDA 编译器。

2）在兼容 gpgpu-sim 的前提下支持 FP8 和 FP4，方便性能提升。

3）未来能整合入 vortex。

合作方需求：MAC 或 TensorCore 本体，标准化，方便嵌入各种设计。

说明：不包括 TensorCore 外的处理器核。

Tensor Core 是一种专为加速矩阵乘法和累加操作设计的硬件单元，其核心目标是通过并行化和流水线技术加速矩阵乘法和累加操作（GEMM），广泛应用于人工智能（AI）与高性能计算（HPC)任务。

本规格说明书描述了一种支持 **FP16/BF16（BF16 后面支持）,FP8(E5M2 和 E4M3),FP4** 输入，FP16/**FP32** 累加的 Tensor Core 设计。

核心特性

- **混合精度支持：**
  - 输入数据格式：FP16/BF16（**BF16 后面支持**）、FP8(E5M2 和 E4M3)、FP4。（均转换为 FP9 再进入 TensorCore）
  - 内部累加精度：~~FP16~~、**FP22（原为 FP32）**。（建议只选一个作为内部累加精度，方便设计）
  - 输出数据格式：FP8/FP16/**FP32**
- **高吞吐量：**
  - 针对密集矩阵运算（如 GEMM：通用矩阵乘法）进行优化。
  - 支持数据搬运与计算并行，性能 x2，寄存器开销 x2（选配）（如果不是 gpgpu-sim 已经支持的，建议去掉，否则编译器工作量太大）
- **灵活的数据类型：**
  - 支持多种输入和输出数据类型。
- **内存高效性：**
  - 利用共享内存和寄存器实现低延迟数据访问。

精度模式

Tensor Core 支持以下精度模式：

FP16/BF16（**BF16 后面支持**）	FP32	**FP32**	深度学习训练
FP8	FP32	FP8/FP16/**FP32**	深度学习推理
FP4	FP32	FP8/FP16/**FP32**	低比特神经网络

输入精度	累加精度	输出精度	主要应用场景
FP16/BF16（BF16 后面支持）	FP22	**FP32**	深度学习训练
FP8	FP22	FP8/FP16/**FP32**	深度学习推理
FP4	FP22	FP8/FP16/**FP32**	低比特神经网络

优先支持粗体标注精度

注意：输入数据均转为 FP8 再进入 TensorCore 参与计算。

矩阵运算

Tensor Core 执行矩阵乘法和累加操作，形式如下：

D=A·B+C （矩阵内积再加）

其中：

- A 和 B 是输入矩阵。
- C 是累加矩阵。

支持的输入矩阵形状

硬件级支持：16x16x16

CUDA 编译级支持：

- FP16/BF16: **m16n16k16**/m32n8k16/m8n32k16
- FP8 : **m16n16k16**/m32n8k16/m8n32k16
- FP4 : **m16n16k16**/m32n8k16/m8n32k16

优先支持粗体标注形状

Dimensions m,n,k: m x k matrix_a; k x n matrix_b; m x n accumulator

数据流

Tensor Core 内部有高达 4Kbyte 的寄存器用于缓存输入矩阵，A,B 占 512byte, C,D 占 1Kbyte。每个块(tile)分为 32 个 threads，每个 thread 包含 4 个 32bit 的寄存器，以兼容 CUDA wmma API。（大小待确认）

1. **输入矩阵：**
   - 从共享内存或者全局内存加载，并做 0 延迟的矩阵转置运算后存入寄存器 Tile，（共享内存和全局内存最好写为 share memory 与 global memory，避免误解）
   - 在寄存器中暂存以实现低延迟访问。A,C 矩阵在寄存器中为行优先排序，B 矩阵为列优先排序。
2. **计算：**
   - 矩阵乘法和累加操作在多个线程中并行执行

3. **输出矩阵：**
    - 结果存储在输出寄存器 Tile 中，D 矩阵在寄存器中为行优先排序。
    - 将寄存器中的输出结果做 0 延迟转置后，写回共享内存或全局内存。

性能指标

- **计算单元：**
    - 一个 TensorCore 包含 32 个并行计算线程(兼容 CUDA)（数量待确认）
    - 每个线程包含 1-8 个(可配置）Mac 运算模块
    - 每个 Mac 运算模块每个周期处理 1 个 32 位寄存器中的 A，B 输入累加。
- **吞吐量：**
    - FP16/BF16：128/1028　　FOPS/cycle（数量待确认）
    - FP8：　256/2056　　FOPS/cycle（数量待确认）
    - FP4：　512/4096　　FOPS/cycle（数量待确认）
- **延迟：**
    - 10-40 个 cycle(受数据总线宽度与 Mac 配置数量，数据类型以及矩阵形状影响）
- **能效：**
    - 设计目标为高能效显著大于 10TOPS/W

1 全局参数及模块列表

1.1 全局参数

参数名	缺省值	描述	备注
SHAPE_K		tc_mul 需要的乘法器个数，即 A 的列数（=B 的行数）	
SHAPE_M		每个 mv_mul 中的 tc_mul 个数（=A 的行数）	
SHAPE_N		mv_mul 的个数（=B 的列数）	
EXPWIDTH		指数位宽	
PRECISION		精度，通常也叫尾数位宽，包括隐含位	

OUTPC		输出精度，用于 fadd_s1 模块	
LATENCY		流水线延迟	
MATRIX_BUS_WIDTH	512 （注意这里与前一版不同）	A, B 矩阵的总线位宽	must be 2^n
MAC_PER_THREAD	2	每个 THREAD 的 MAC 数量	
PARALLEL_DATA_MOVER	0	0：不支持并行数据搬运 1：支持并行数据搬运	暂不支持并行数据搬运
TRANSPOSE	0	0：不支持矩阵转置 1：支持矩阵转置	当设为 1 时 transpose_en 端口无效 暂不支持矩阵转置

参数名称	缺省值	描述	备注
MATRIX_BUS_WIDTH	1024	A, B 矩阵的总线位宽	must be 2^n
MAC_PER_THREAD	2	每个 THREAD 的 MAC 数量	
PARALLEL_DATA_MOVER	0	0：不支持并行数据搬运 1：支持并行数据搬运	暂不支持并行数据搬运
TRANSPOSE	0	0：不支持矩阵转置 1：支持矩阵转置	当设为 1 时 transpose_en 端口无效 暂不支持矩阵转置

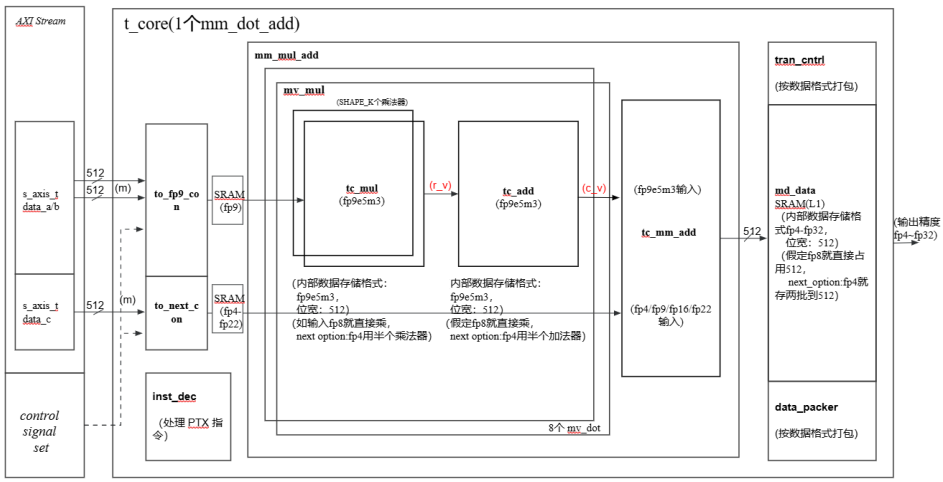
1.2 模块列表

分类	模块	描述
顶层	tensor_cor_e.v	张量核心顶层设计模块
	tensor_cor_e_exev.v	张量核心顶层设计模块(最顶层)
格式转换	to_fp8_con.v	FP8 格式转换器（顶层接口）
	to_fp8_core.v	FP8 格式转换核心逻辑模块(最后是转化为 FP9)
	fp22_to_fp8_con.v	FP22 格式到 FP8 格式的专用转换器
张量计算	tc_dot_product.v	点积计算单元（核心计算模块）
	tc_mul_pipe.v	乘法流水线模块
	tc_add_pipe.v	加法流水线模块
	naivemultiplier.v	基础乘法器实现
浮点运算单元	fadd_s1.v	浮点加法器第一阶段
	fadd_s2.v	浮点加法器第二阶段
	fmul_s1.v	浮点乘法器第一阶段
	fmul_s2.v	浮点乘法器第二阶段
	fmul_s3.v	浮点乘法器第三阶段
辅助计算	cf_math_pkg.sv	数学函数和常数定义包
	norm_div_sqrt_mvp.sv	支持乘法的归一化/除法/平方根计算模块
存储单元	singleport_SRAM.v	单端口 SRAM 存储器模型
配置与控制	config_registers.v	配置寄存器模块
项目支撑	define.sv	全局宏定义文件
	filelist.f	项目文件列表（用于编译/仿真）
	run.tcl	自动化运行脚本
日志文件	command.log	命令执行历史日志
	scl.log	脚本运行日志
其他文件	default.svf	默认配置文件
	qrd_mvp.pvk	未知

2 顶层

2.1 顶层架构

位于网盘/MB1_RDPrjIntern/prj25_TensorCore1/框图/TensoreCore 架构图.oslides



m：矩阵，r_v：行向量，c_v：列向量，ABC 矩阵维数均为 8*8，（默认）精度相同，8 个 mv_mul 模块，每个 mv_mul 模块中有 8 个 tc_mul 模块

1. 总线与 tensor_core 模块之间：
    1. 来自总线上的数据以矩阵形式送入 tensor_core 模块，其中 A/C 元素均为行主存，C 元素为列主存（这里应该有误，B 元素为列主存吧？），位宽都是 512
2. 格式转换器和 sram：
    1. to_fp9_con 将所有精度的数据转换为 fp9（e5m3），to_next_con 根据 tc_mm_add 的输入精度选择不同的格式将输入转换为 fp4-fp22
    2. 插入 SRAM 用于存储计算结果，fp9 sram 的位宽为 512/8*9，若总线数据精度为 fp4，则需要并串转换存 2 次，若数据精度为 fp16，则计算 2 个周期存 1 次
3. mm_mul_add 模块和最后一级 SRAM：
    1. 把 gemm 算法分解为 gemv 算法，让 A 矩阵分别乘 B 矩阵的每个列向量，结果列向量组以 fp（9e5m3）输入到 tc_mm_add 模块构成中间矩阵，与来自 SRAM（fp4-fp22）的矩阵元素做累加
    2. 输出位宽为 512，根据 sram 内部数据存储格式调整输出次数
    3. 最后一级 SRAM 用于暂存计算结果，输入位宽为 512，若数据精度为 fp8，则直接占用 512，若数据精度为 fp4 则存两批到 512，若数据精度为 fp16，则计算 2 个周期存 1 次，若数据精度为 fp32，则计算 4 个周期存 1 次
4. mv_mul 模块：

    1. 对于一个 mv_mul 而言，输入数据为 A 矩阵和 B 矩阵的一个列向量
    2. 将 A 矩阵分解为若干个行向量，分发到 8 个 tc_mul 组成的阵列在其内部做向量的元素乘法，输出的结果为 A 的第 n 行向量与 B 的第 n 行标量相乘的行向量
    3. 上一步得到的行向量组送入 tc_add 加法树得到一个列向量
5. tc_mm_add 模块：
    1. 若干个列向量拼接组成中间矩阵和来自 SRAM（fp4-fp22）的矩阵元素做累加
6. tensor_core 模块输出：
    1. 输出精度为 fp4-fp32，和 SRAM 存储数据的精度保持一致

2.2 tensor_core 顶层模块

主要更改点：

1. 去掉 parameter EXPWIDTH 和 PRECISION 而在 mv_mul 模块中具体配置。因为乘法器和累加器精度不同，可以先不改
2. 由 SHAPE_[M|N|K]控制矩阵维度，去掉 DIM_[M|N|K]
3. in_valid 由 a/b/c 三个 valid 和三个 last 相与，out_ready 由三个 ready 相与（按照 AXI Stream 协议处理）
4. 除 clk 和 rst_n 外，输入输出后缀分别为_i 和_o
5. 将位宽配置与线程信息解耦

功能解析详见配置类模块分析

该模块的配置为

- 矩阵总线宽度（MATRIX_BUS_WIDTH）为 512bit
- 每个线程具有 2 个 MAC 单元，暂定由 1 个 mm_mul_add 模块实现
- 不支持并行数据搬运和转置

接口时序说明

- **配置：**
    - rst_n 解除后，将 en_i 置低。
    - 当 busy_o 信号拉低后，对配置接口按需求进行配置。
    - 配置接口必须在 busy 为高时保持不变，否则行为未定义。也就是说配置必须在 busy_o 为 0 时进行
- **启动计算：**
    - 配置完成后，将 en_i 信号置高，等待输入数据，当收到第一个输入流数据时，busy_o 拉高表示 tensorCore 开始运算

- **计算过程：**
  - 在计算过程中，busy _o 信号保持高电平。
- **计算完成：**
  - 当计算完成时，最后一个输出数据完成后，busy _o 信号置低，irq _o 信号置高。表示一个周期完成，等待下一个周期的数据输入

引脚名称	方向	位宽	说明
clk	input	1	全局时钟，上升沿触发
rst_n	input	1	异步复位信号，低电平有效。复位时，所有内部状态和寄存器清零
en_i	input	1	控制信号，用于使能 tensor_core 1：启动　　0：停止
busy_i（似乎应为 busy_o）	output	1	状态信号，表示 tc 数据输入或输出尚未完成 状态信号，高电平有效。当 Tensor Core 正在计算时，busy 为高
irq_o	output	8	状态信号，高 6 位保留，用于生成计算完成中断和配置错误中断 中断信号，高电平有效。 bit 0 ： 当计算完成时置高，通知外部模块。 bit 1 ： 当内部出错时置高 其他：预留
irq_mask_i	input	8	控制信号，irq 中断掩码 中断使能信号，高电平有效 控制 irq 对应 bit
矩阵类型控制信号			

引脚名称	方向	位宽	说明
屏蔽 layout_i	input	4	控制信号，用于布局 bit 0: A 矩阵布局 0: 行优先 1: 列优先（预留） bit 1: B 矩阵布局 0: 行优先(预留) 1: 列优先 bit 2: C 矩阵布局 0: 行优先 1: 列优先 （预留） bit 3: D 矩阵布局 0: 行优先 1: 列优先 （预留）
transpose_en_i（可能不必须）	input	1	控制信号，用于转置使能 0: 禁用转置功能 1：启用转置功能 （预留）
type_ab_i	input	5	控制信号，矩阵 A/B 元素的数据格式 0x2 : fp8 0x6 : fp4 0xA : fp16 其他：为将来扩展预留，预计增加 INT4

引脚名称	方向	位宽	说明
type_ab_sub_i	input	3	控制信号，矩阵 A/B 元素的数据子格式 当 type_ab=2（fp8）时有效 0: E5M2　　1: E4M3
type_cd_i	input	5	控制信号，矩阵 C/D 元素的数据格式 4/8 0xA : fp16 （预留） 0xE : fp32 其他：预留
shape_m_i	input	4	大矩阵 A 的行数，用于计算 mm_mul_add 模块的处理次数 矩阵块形状 M 维度 0:4　1: 8　2: **16**　3: 32　4: 64　5: 128 6: 256 7: 512 **仅支持本 IP 支持的形状维度(16)，其余**预留
shape_n_i	input	4	大矩阵 B 的列数，用于计算 mm_mul_add 模块的处理次数 矩阵块形状 N 维度 0:4　1: **8**　2: **16**　3: 32　4: 64　5: 128 6: 256 7: 512 **仅支持本 IP 支持的形状维度(8,16)　其余**预留

引脚名称	方向	位宽	说明
shape_k_i	input	4	大矩阵 A 的列数或 B 的行数，用于计算 mm_mul_add 模块的处理次数 矩阵块形状 K 维度 0:4　1: 8　2: **16**　3: **32**　4: **64**　5: 128 6: 256 7: 512 **仅支持本 IP 支持的形状维度 (16,32,64),其余**预留
与 AXI Stream 总线的交互信号 Tensor Core 模块包含三个 AXI Stream 输入接口，分别用于输入矩阵 A、B 与累加矩阵 C			
s_axis_tdata_[a|b|c]_i	input	MATRIX_BUS_WIDTH	数据信号，矩阵 A/B/C 的数据
s_axis_tvalid_[a|b|c]_i	input	1	控制信号，矩阵 A/B/C 的数据是否有效 输入数据有效信号，高电平有效
s_axis_tready_[a|b|c]_o	output	1	状态信号，矩阵 A/B/C 的数据是否就绪 输入数据接收就绪信号，高电平有效
s_axis_tlast_[a|b|c]_i	input	1	控制信号，矩阵 A/B/C 的数据是否传输完毕 输入数据包结束信号，高电平有效
m_axis_tdata_d_o	output	MATRIX_BUS_WIDTH	数据信号，矩阵 D 的数据
m_axis_tvalid_d_o	output	1	状态信号，矩阵 D 的数据是否有效 输出数据有效信号，高电平有效
m_axis_tready_d_i	input	1	控制信号，矩阵 D 的数据是否就绪 输出数据接收就绪信号，高电平有效

引脚名称	方向	位宽	说明
m_axis_tlast_d_o	output	1	状态信号，矩阵 D 的数据是否传输完毕 输出数据包结束信号，高电平有效
舍入控制			
rm_i	input	3	控制信号，舍入模式
写回目的寄存器索引控制信号			
ctrl_reg_idxw_i	input	8	控制信号，写回目的寄存器索引（包括扩展寄存器）
reg_idxw_o	output	8	数据信号，目的寄存器索引
线程束 ID			
ctrl_wid_i	input	DEPTH_WARP	控制信号，Warp ID
wid_o	output	DEPTH_WARP	数据信号，线程束 id
输出数据写回信号			
wb_wvd_rd_o	output	MATRIX_BUS_WIDTH	数据信号，流水线响应接口中的写回向量目的寄存器读响应，XLEN=22
wvd_mask_o	output	NUM_THREAD	数据信号，写回向量目的寄存器掩码
wvd_o	output	1	状态信号，表示需要写回

3 格式转换模块

3.1 to_fp9_con 模块

接收 ab 数据，根据数据精度选择不同格式转换通路，最终转换为 fp9（e5m3）格式存到 sram 模块

若总线数据精度为 fp4，则需要并串转换存 2 次，若数据精度为 fp16，则计算 2 个周期存 1 次

引脚名称	方向	位宽	说明
格式控制信号			

引脚名称	方向	位宽	说明
type_ab_i	input	5	输入矩阵 A/B 元素的数据格式
type_ab_sub_i	input	3	输入矩阵 A/B 元素的数据子格式
数据信号			
[a|b]_i	input	512	矩阵 A/B 数据输入
[a|b]_o	output	512/8*9	转换后的 B 数据，fp9（e5m3）格式
特殊情况与异常输出			
握手信号？			

3.2 to_next_con 模块

引脚名称	方向	位宽	说明
格式控制信号			
type_cd_i	input	5	输入矩阵 C 元素的数据格式（暂定和 type_ab_i 相同）
数据信号			
c_i	input	512	数据信号，输入为来自总线的矩阵 C 元素
c_o	output	512/8*22	数据信号，输出为 mm_mul_add 所需的累加输入
特殊情况与异常输出			
握手信号？			

4 张量计算模块

4.1 mm_mul_add 模块

该模块例化了 8 个 mv_mul 模块和一个矩阵累加模块 tc_mm_add，将 A 矩阵所有数据和 B 矩阵列向量数据依次发送到 mv_mul 模块，最后把中间矩阵和外部累加矩阵求和输出，是 gemm 分解为 gemv 算法的实现

引脚名称	方向	位宽	说明
输入数据信号			
[a|b]_i	input	512/8*9	矩阵 A/B 输入数据，线程数*元素位宽
c_i	input	512/8*22	矩阵 C 输入数据
只读控制信号			
rm_i	input	VL*3	线程束舍入模式配置寄存器
ctrl_reg_idxw_i	input	8	控制信号，寄存器写回索引
ctrl_warpid_i	input	DEPTH_WARP	控制信号，Warp ID
握手信号			
in_valid_i	input	1	握手信号，输入数据是否有效
out_ready_i	input	1	握手信号，下游是否准备好接收结果
in_ready_o	output	1	状态信号，本模块是否准备好接收数据
out_valid_o	output	1	状态信号，输出结果是否有效
输出数据信号			
result_o	output	512	输出计算结果（矩阵形式）
fflags_o	output	VL*5	线程异常标志寄存器输出（如 overflow）

4.3 mv_mul 模块

该模块实现了 gemv 的功能，大致流程如下：

1. 例化 shape_m 个 tc_mul 模块，分发 A 矩阵的每行数据到 tc_mul 中，用于 A 的一行和 B 的一列相乘
2. 例化 tc_add 模块，实现输入行向量组的对行求和，此时输出为 shape_n*1 的列向量

引脚名称	方向	位宽	说明
输入数据信号			
a_i	input	512/8*9	数据信号，矩阵 A 元素输入
b_i	input	8*9	数据信号，矩阵 B 的列向量元素输入

引脚名称	方向	位宽	说明
只读控制信号			
rm_i	input	3	控制信号，舍入模式
ctrl_reg_idxw_i	input	8	控制信号，写回目的寄存器索引（包括扩展寄存器）
ctrl_wid_i	input	DEPTH_WARP	控制信号，Warp ID
握手信号			
in_valid_i	input	1	握手信号，输入数据是否有效
out_ready_i	input	1	握手信号，下游是否准备好接收结果
in_ready_o	output	1	状态信号，本模块是否准备好接收数据
out_valid_o	output	1	状态信号，输出结果是否有效
输出数据信号			
result_o	output	8*9	数据信号，A 矩阵乘 B 矩阵列向量得到的列向量
fflags_o	output	5	异常标志寄存器输出（如 overflow）

4.4 tc_mul 模块

该模块实现了 A 的每一行与 B 的每一列相乘，可以例化 SHAPE_K=8 个 ventus 的 tc_mul_pipe 模块，并对位宽适当调整来实现

引脚名称	方向	位宽	说明
输入数据信号			
a|b_i	input	8*9	输入 A/B 矩阵元素数据
只读控制信号			
rm_i	input	3	舍入模式（IEEE 754）
ctrl_c_i	input	4/8/16	控制信号，累加项 C
ctrl_rm_i	input	3	控制信号，舍入模式
ctrl_reg_idxw_i	input	8	控制信号，目标寄存器索引
ctrl_warpid_i	input	DEPTH_WARP	控制信号，Warp ID

引脚名称	方向	位宽	说明
握手信号			
in_valid_i	input	1	握手信号，输入数据是否有效
out_ready_i	input	1	握手信号，下游模块是否准备好接收输出
in_ready_o	output	1	握手信号，本模块是否准备好接收数据
out_valid_o	output	1	握手信号，输出数据是否有效
输出数据信号			
result_o	output	8*9	矩阵 A 的行向量元素乘 B 的列向量元素组成的行向量
fflags_o	output	5	浮点异常标志（如 overflow, underflow 等）
传递控制信号			
ctrl_c_o	output	4/8/16	传递控制信号，累加项 C
ctrl_rm_o	output	3	传递控制信号，舍入模式
ctrl_reg_idxw_o	output	8	传递控制信号，目标寄存器索引
ctrl_warpid_o	output	DEPTH_WARP	传递控制信号，Warp ID

4.5 tc_add 模块

该模块实现了对 tc_mul 阵列输出的行向量组的行累加，可以通过例化 SHAPE_N=8 个加法树实现，加法树的构造参考 ventus 的 tc_add_pipe，同样也只是位宽的调整

引脚名称	方向	位宽	说明
输入数据信号			
r_v_i	input	8*8*9	输入行向量组元素数据
只读控制信号			
rm_i	input	3	舍入模式（IEEE 754）
ctrl_c_i	input	4/8/16	控制信号，累加项 C
ctrl_rm_i	input	3	控制信号，舍入模式
ctrl_reg_idxw_i	input	8	控制信号，目标寄存器索引

引脚名称	方向	位宽	说明
ctrl_warpid_i	input	DEPTH_WARP	控制信号，Warp ID
握手信号			
in_valid_i	input	1	握手信号，输入数据是否有效
out_ready_i	input	1	握手信号，下游模块是否准备好接收输出
in_ready_o	output	1	握手信号，本模块是否准备好接收数据
out_valid_o	output	1	握手信号，输出数据是否有效
输出数据信号			
result_o	output	8*9	对行向量组沿行求和的列向量
fflags_o	output	5	浮点异常标志（如 overflow, underflow 等）
传递控制信号			
ctrl_c_o	output	4/8/16	传递控制信号，累加项 C
ctrl_rm_o	output	3	传递控制信号，舍入模式
ctrl_reg_idxw_o	output	8	传递控制信号，目标寄存器索引
ctrl_warpid_o	output	DEPTH_WARP	传递控制信号，Warp ID

5 数据通道模块

5.1 tc_mul 模块

该模块用于对单个 A 矩阵元素和 B 矩阵元素相乘，是构成点积模块的基本单元之一。

乘法流水线分 3 级实现，用反压信号控制流水线寄存器更新：

1. 预处理阶段（fmul_s1）和尾数相乘（naivemultiplier）
2. 第一级输出整合阶段（fmul_s2）
3. 舍入与规格化阶段（fmul_s3）
4. 扩位至 fp22

引脚名称	方向	位宽	说明
输入数据信号			

引脚名称	方向	位宽	说明
a|b_i	input	5+4	输入 A/B 矩阵元素数据
只读控制信号			
rm_i	input	3	舍入模式（IEEE 754）
ctrl_c_i	input	4/8/16	控制信号，累加项 C
ctrl_rm_i	input	3	控制信号，舍入模式
ctrl_reg_idxw_i	input	8	控制信号，目标寄存器索引
ctrl_warpid_i	input	DEPTH_WARP	控制信号，Warp ID
握手信号			
in_valid_i	input	1	握手信号，输入数据是否有效
out_ready_i	input	1	握手信号，下游模块是否准备好接收输出
in_ready_o	output	1	握手信号，本模块是否准备好接收数据
out_valid_o	output	1	握手信号，输出数据是否有效
输出数据信号			
result_o	output	5+4	浮点乘法结果（带累加）
fflags_o	output	5	浮点异常标志（如 overflow, underflow 等）
传递控制信号			
ctrl_c_o	output	4/8/16	传递控制信号，累加项 C
ctrl_rm_o	output	3	传递控制信号，舍入模式
ctrl_reg_idxw_o	output	8	传递控制信号，目标寄存器索引
ctrl_warpid_o	output	DEPTH_WARP	传递控制信号，Warp ID

5.2 tc_add 模块

该模块用于实现加法树和与 C 矩阵元素相加，是构成点积模块的基本单元之一

加法流水线分 2 级实现，用反压信号控制流水线寄存器更新：

1. fadd_s1 对两个浮点数分类，选择计算路径相加
2. fadd_s2 处理舍入，溢出和规格化

pinlist 和 tc_mul_pipe 模块相同，位宽略有差异

5.3 tc_mm_add 模块

在 tc_mm_add 模块例化时需注意：

1. 对前级模块 mv_mul 的输出进行整合：前级输出为列向量组，进行位拼接形成矩阵
2. 混合精度相加时，需要适当扩位

引脚名称	方向	位宽	说明
输入数据信号			
c_v_i	input	8*8*9	输入列向量组元素数据
只读控制信号			
rm_i	input	3	舍入模式（IEEE 754）
ctrl_c_i	input	4/8/16	控制信号，累加项 C
ctrl_rm_i	input	3	控制信号，舍入模式
ctrl_reg_idxw_i	input	8	控制信号，目标寄存器索引
ctrl_warpid_i	input	DEPTH_WARP	控制信号，Warp ID
握手信号			
in_valid_i	input	1	握手信号，输入数据是否有效
out_ready_i	input	1	握手信号，下游模块是否准备好接收输出
in_ready_o	output	1	握手信号，本模块是否准备好接收数据
out_valid_o	output	1	握手信号，输出数据是否有效
输出数据信号			
result_o	output	512	行主序输出的结果矩阵
fflags_o	output	5	浮点异常标志（如 overflow, underflow 等）
传递控制信号			
ctrl_c_o	output	4/8/16	传递控制信号，累加项 C
ctrl_rm_o	output	3	传递控制信号，舍入模式
ctrl_reg_idxw_o	output	8	传递控制信号，目标寄存器索引

引脚名称	方向	位宽	说明
ctrl_warpid_o	output	DEPTH_WARP	传递控制信号，Warp ID

6 基本计算模块

6.1 fmul_s1 模块

该模块具有以下功能：

1. 对输入浮点数进行分类（规格化/非规格化/无穷/NaN 等）
2. 计算初步乘法结果（符号/指数/尾数）
3. 处理特殊值情况（NaN/Inf/零等）
4. 为第二阶段准备规格化参数

引脚名称	方向	位宽	说明
输入数据信号			
s_axis_tdata_a	input	5+4	矩阵 A 元素输入
s_axis_tdata_b	input	5+4	矩阵 B 元素输入
只读控制信号			
rm_i	input	3	舍入模式（IEEE 754）
特殊情况与异常输出			
out_special_case_valid_o	output	1	特殊值标志是否有效
out_special_case_nan_o	output	1	结果是否为 NaN
out_special_case_inf_o	output	1	结果是否为 ±Inf
out_special_case_inv_o	output	1	是否为无效操作（invalid）
out_special_case_haszero_o	output	1	是否存在零操作数
out_early_overflow_o	output	1	是否发生早期溢出
out_may_be_subnormal_o	output	1	可能为非规格化数
输出数据信号			
out_prod_sign_o	output	1	乘积符号位

引脚名称	方向	位宽	说明
out_shift_amt_o	output	5+1	最终移位量
out_exp_shifted_o	output	5+1	移位后指数
传递控制信号			
out_rm_o	output	3	传递舍入模式

6.2 naivemultiplier 模块

该模块用于尾数相乘，LEN=4

引脚名称	方向	位宽	说明
regenable	input	1	控制信号，寄存器写使能
s_axis_tdata_a	input	LEN	矩阵 A 元素尾数输入
s_axis_tdata_b	input	LEN	矩阵 B 元素尾数输入
result	output	LEN * 2	数据输出，尾数相乘结果

6.3 fmul_s2 模块

该模块用于传递第一阶段处理结果，EXPWIDTH=5，PRECISION=4

引脚名称	方向	位宽	说明
特殊情况、异常与舍入控制信号			
in_special_case_valid_i	inout	1	特殊路径有效标志
in_special_case_nan_i	inout	1	NaN 结果标志
in_special_case_inf_i	inout	1	无穷标志
in_special_case_inv_i	inout	1	无效操作标志
in_special_case_haszero_i	inout	1	存在零操作数
in_early_overflow_i	inout	1	早期溢出标志
in_may_be_subnormal_i	inout	1	可能为非规格化数
in_rm_i	inout	3	舍入模式

引脚名称	方向	位宽	说明
输出数据信号			
in_prod_sign_i	inout	1	乘积符号位
in_shift_amt_i	inout	EXPWIDTH+1	尾数移位量
in_exp_shifted_i	inout	EXPWIDTH+1	调整后指数
prod_i	inout	PRECISION*2	尾数乘积

6.4 fmul_s3 模块

EXPWIDTH=5, PRECISION=4

该模块的功能为：

1. 对乘法结果进行最终舍入处理
2. 处理溢出情况
3. 生成规格化结果和异常标志
4. 输出到加法树的中间结果

引脚名称	方向	位宽	说明
输入数据信号			
in_prod_i	input	PRECISION * 2	数据信号，乘积中间结果
in_prod_sign_i	input	1	乘积符号位
in_shift_amt_i	input	EXPWIDTH + 1	数据信号，移位量
in_exp_shifted_i	input	EXPWIDTH + 1	数据信号，移位后指数
特殊情况与异常输入			
in_special_case_valid_i	input	1	特殊路径有效标志
in_special_case_nan_i	input	1	NaN 结果标志
in_special_case_inf_i	input	1	无穷标志
in_special_case_inv_i	input	1	无效操作标志
in_special_case_haszero_i	input	1	存在零操作数标志
in_early_overflow_i	input	1	早期溢出标志

引脚名称	方向	位宽	说明
in_may_be_subnormal_i	input	1	可能为非规格化数
只读控制信号			
in_rm_i	input	3	舍入模式（IEEE 754）
输出数据信号			
result_o	output	EXPWIDTH + PRECISION	数据信号，最终规格化结果
fflags_o	output	5	异常标志（NV, OF, UF, DZ, NX）
输出到加法树的数据与标志信号			
to_fadd_fp_prod_sign_o	output	1	输出给加法器的符号位
to_fadd_fp_prod_exp_o	output	EXPWIDTH	数据信号，输出给加法器的指数
to_fadd_fp_prod_sig_o	output	2 * PRECISION - 1	数据信号，输出给加法器的尾数（怎么扩位？）
to_fadd_is_nan_o	output	1	输出 NaN 标志
to_fadd_is_inf_o	output	1	输出无穷标志
to_fadd_is_inv_o	output	1	输出无效操作标志
to_fadd_overflow_o	output	1	输出溢出标志

6.5 fadd_s1 模块

- 当作为加法树时，EXPWIDTH=5, PRECISION=8, OUTPC=4
- 当作为累加器时，EXPWIDTH=8, PRECISION=28, OUTPC=14

引脚名称	方向	位宽	说明
输入数据信号			
a_i	input	EXPWIDTH + PRECISION	操作数 a
b_i	input	EXPWIDTH + PRECISION	操作数 b

引脚名称	方向	位宽	说明
用于 fma 指令的标志信号，根据需要选择性保留			
b_inter_valid_i	input	1	中间结果是否有效
b_inter_flags_is_nan_i	input	1	中间结果为 NaN 标志
b_inter_flags_is_inf_i	input	1	中间结果为无穷标志
b_inter_flags_is_inv_i	input	1	无效操作标志
b_inter_flags_overflow_i	input	1	溢出标志
舍入模式控制信号			
rm_i	input	3	舍入模式（IEEE 754）
out_rm_o	output	3	传递的舍入模式
输出给下一级的数据信号			
out_far_sign_o	output	1	far path 符号位
out_far_exp_o	output	EXPWIDTH	far path 指数
out_far_sig_o	output	OUTPC + 3	far path 尾数（带保护位）
out_near_sign_o	output	1	near path 符号位
out_near_exp_o	output	EXPWIDTH	near path 指数
out_near_sig_o	output	OUTPC + 3	near path 尾数（带保护位）
特殊情况与异常输出			
out_special_case_nan_o	output	1	结果为 NaN 标志
out_special_case_inf_sign_o	output	1	无穷结果的符号位
out_small_add_o	output	1	小加法标志（表示非规格数加法）
out_far_mul_of_o	output	1	far path 乘法溢出标志
out_near_sig_is_zero_o	output	1	near path 尾数为零标志
路径选择信号			

引脚名称	方向	位宽	说明
out_sel_far_path_o	output	1	路径选择信号（1 表示选择 far path）

6.6 fadd_s2 模块

- 当作为加法树时，EXPWIDTH=5, PRECISION=4
- 当作为累加器时，EXPWIDTH=8, PRECISION=14

该模块的功能为：

1. 对第一阶段结果进行舍入处理
2. 处理溢出情况
3. 生成最终规格化结果
4. 输出异常标志

引脚名称	方向	位宽	说明
输入数据信号			
in_far_sign_i	input	1	far path 符号位
in_far_exp_i	input	EXPWIDTH	far path 指数
in_far_sig_i	input	PRECISION + 3	far path 尾数（含保护位）
in_near_sign_i	input	1	near path 符号位
in_near_exp_i	input	EXPWIDTH	near path 指数
in_near_sig_i	input	PRECISION + 3	near path 尾数
路径选择信号			
in_sel_far_path_i	input	1	路径选择信号（1 表示 far path）
特殊情况与异常输入			
rm_i	input	3	舍入模式（IEEE 754）
in_far_mul_of_i	input	1	far path 乘法溢出标志
in_near_sig_is_zero_i	input	1	near path 尾数为零
in_special_case_valid_i	input	1	特殊路径有效标志

引脚名称	方向	位宽	说明
in_special_case_iv_i	input	1	无效操作标志（Invalid）
in_special_case_nan_i	input	1	NaN 结果标志
输出数据和标志信号			
out_result_o	output	EXPWIDTH + PRECISION	最终计算结果
out_fflags_o	output	5	浮点异常标志（NV, OF, UF, DZ, NX）
out_far_uf_o	output	1	far path 下溢标志
out_near_of_o	output	1	near path 溢出标志

6.7 far_path 模块

- 当作为加法树时，EXPWIDTH=5, PRECISION=8, OUTPC=4
- 当作为累加器时，EXPWIDTH=8, PRECISION=28, OUTPC=14

该模块实现的是浮点加法/减法中的 Far Path 加法路径，即当两个浮点数指数差 expdiff_i 大于等于 2 时使用的路径。

其主要功能包括：

1. 对阶并对尾数对齐：
   使用 shift_right_jam 模块将较小数尾数右移（带粘性位 sticky）对齐。
2. 尾数相加/相减：
   根据 effsub_i 控制是否为减法（即两数符号不同）。采用补码加法方式处理带符号的相加减。
3. 规格化：
   - 根据加法结果 addr_result 的高位判断是否需要规格化：是否有进位（cout）、是否可以保持原指数（keep）、是否需要减小指数（cancellation）。
   - 最终尾数取出 OUTPC 位有效位并添加保护位、粘性位等。
4. 输出最终结果：
   - 指数调整后输出为 result_exp_o；
   - 尾数截取为 result_sig_o；
   - 符号位保持为较大数（此处为 A）的符号。

名称	方向	位宽	说明
a_sign_i	input	1	A 操作数的符号位
a_exp_i	input	[EXPWIDTH-1:0]	A 操作数的指数位
a_sig_i	input	[PRECISION-1:0]	A 操作数的尾数
b_sig_i	input	[PRECISION-1:0]	B 操作数的尾数（已确认指数更小）
expdiff_i	input	[EXPWIDTH-1:0]	指数差 a_exp_i - b_exp_i
effsub_i	input	1	有效减法控制位（不同符号）
small_add_i	input	1	特殊小加法标志（用于规约控制）
result_sign_o	output	1	结果符号位（继承自 A）
result_exp_o	output	[EXPWIDTH-1:0]	结果指数
result_sig_o	output	[OUTPC+2:0]	规格化尾数（包含保留位和粘性位）

6.8 near_path 模块

- 当作为加法树时，EXPWIDTH=5, PRECISION=8, OUTPC=4
- 当作为累加器时，EXPWIDTH=8, PRECISION=28, OUTPC=14

near_path 模块实现了浮点减法中指数差值较小（通常 < 2） 的近路径计算逻辑

名称	方向	位宽	说明
a_sign_i	input	1	A 操作数的符号位
a_exp_i	input	[EXPWIDTH-1:0]	A 操作数的指数
a_sig_i	input	[PRECISION-1:0]	A 操作数的尾数
b_sign_i	input	1	B 操作数的符号位
b_sig_i	input	[PRECISION-1:0]	B 操作数的尾数
need_shift_b_i	input	1	是否需要对 B 的尾数进行右移
result_sign_o	output	1	近路径计算结果的符号位
result_exp_o	output	[EXPWIDTH-1:0]	近路径计算结果的指数
result_sig_o	output	[OUTPC+2:0]	近路径计算结果的规格化尾数
sig_is_zero_o	output	1	尾数是否为全 0

名称	方向	位宽	说明
a_lt_b_o	output	1	是否 A 的尾数小于 B（用于符号判断）

7 舍入模块

该模块用于根据舍入模式处理输入浮点数，WIDTH 为父模块 PRECISION-1

引脚名称	方向	位宽	说明
in	input	WIDTH	需舍入的原始尾数
sign	input	1	数值符号位，用于判断向上/向下舍入
roundin	input	1	舍入位（Round bit）
stickyin	input	1	黏着位（Sticky bit）
rm	input	3	舍入模式（RNE/RTZ/RUP/RDN/RMM）
out	output	WIDTH	尾数舍入结果
inexact	output	1	是否存在精度丢失（即需舍入）
cout	output	1	进位标志（舍入产生进位）
r_up	output	1	是否发生了向上舍入

8 辅助计算模块

由于 fp8 的尾数较少，因此可能会有比 ventus 长尾数操作逻辑门更少、延迟更小的算法，具有一定的优化空间

8.1 shift_right_jam 右移模块

实现右移并检测黏着位是否为 1

引脚名称	方向	位宽	含义
in	input	[LEN-1:0]	原始输入数据
shamt	input	[EXP-1:0]	右移的位数（移位量）
out	output	[LEN-1:0]	右移后的输出结果

引脚名称	方向	位宽	含义
sticky	output	1	粘滞位，表示移出位是否包含 1

8.2 lza 前导 0 预测模块

该模块用于提前预测运算结果中前导零的位置，加快归一化过程。

引脚名称	方向	位宽	含义
a	input	[LEN-1:0]	输入操作数 A
b	input	[LEN-1:0]	输入操作数 B
c	output	[LEN-1:0]	LZA 输出结果，供前导零检测使用

8.3 lzc 前导 0 计数模块

位于文件 ventus-gpgpu-verilog/src/gpgpu_top/sm/pipeline/sfu_v2/float_div_mvp/lzc.sv 中，这里用到了 cf_math_pkg 包中的 idx_width 函数，详见 cf_math_pkg::idx_width 函数

归约二叉树实现的尾零计数器 / 首零计数器模块：

当 MODE 设为 0 时，模块作为尾零计数器使用，cnt_o 表示从最低有效位（LSB）开始的连续 0 的数量；
当 MODE 设为 1 时，模块作为首零计数器使用，cnt_o 表示从最高有效位（MSB）开始的连续 0 的数量。

如果输入中全为 0（不包含任何 1），则 empty_o 置为 1，同时 cnt_o 的值为最大可能的零数量减 1。
例如（MODE = 0）：

- 输入 in_i = 000_0000，则 empty_o = 1，cnt_o = 6
- 输入 in_i = 000_0001，则 empty_o = 0，cnt_o = 0
- 输入 in_i = 000_1000，则 empty_o = 0，cnt_o = 3

参数名	类型	默认值	说明
WIDTH	int unsigned	2	输入向量的位宽
MODE	bit	1'b0	模式选择：0 → 尾零计数，1 → 首零计数

参数名	类型	默认值	说明
CNT_WIDTH	int unsigned	自动计算	输出计数结果的位宽（根据 WIDTH 推导）

信号名	方向	位宽	说明
in_i	input	WIDTH	输入向量
cnt_o	output	CNT_WIDTH	首/尾部 0 的数量
empty_o	output	1	若输入全为 0，该信号为 1，表示计数器“空”

9 存储模块 SRAM

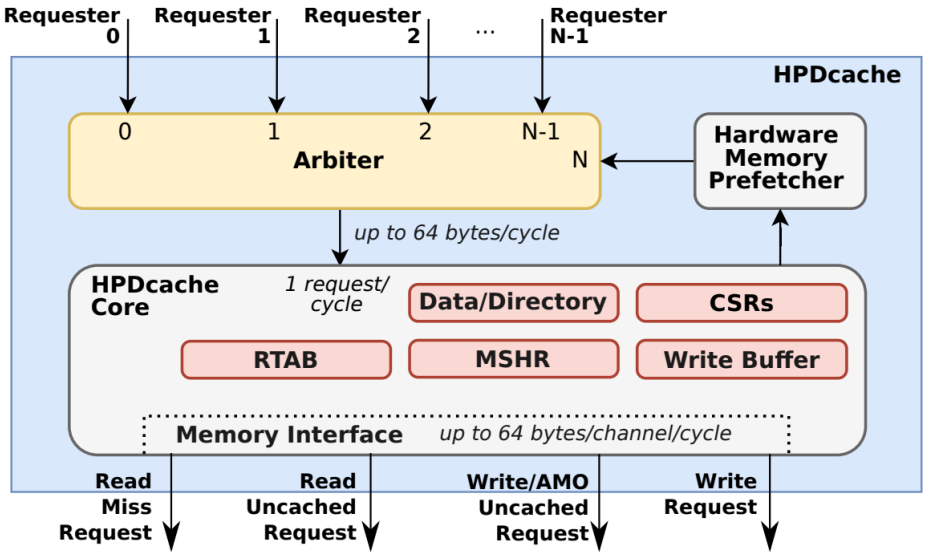


Figure 1: HPDcache Micro-architecture Overview

9.1 SRAM(FP9)模块

该模块用于存储经过 to_fp9_con 转换后的计算结果（fp9 格式的 A/B 数据）

9.2 SRAM(FP4-FP22)模块

该模块用于存储经过 to_next_con 转换后的计算结果（fp4-fp22 格式的 C 数据）

9.3 SRAM(L1)模块

该模块用于存储经过 mm_mul_add 模块计算后的计算结果，输入为中间矩阵和 C 矩阵元素累加后的结果，存储数据精度为 fp4-fp22。