RISC-V Tile Extension 扩展指令集（Deprecated）

1. 概要

本文档描述基于RISC-V的Tile扩展。Tile扩展的架构，选择了一套独立于RV Vector Extension的机制。这点不同于典型的NV/AMD的GPGPU路径。独立的Tile扩展架构，整体的地位等同于RVV，因此可认为完整的基于RISC-V的AI核包括三大部分：RISC-V scalar，RISC-V Vector Extension，RISC-V Tile Extension。

Tile扩展的主体是Matrix扩展，然后还包括对应的Tile vector ALU, Tile Load/Store, and misc instructions.

整篇文档学习RISC-V vector & matrix扩展规范。

Tile Core指令

指令类型：

1. 配置指令：配置数据类型，tile寄存器，tile大小，
2. 寄存器移动指令：Scalar，Vector，Tile寄存器之间的数据移动，数据广播指令
3. Load/Store指令：
   a. 将数据从**本地存储**（L2B SRAM/on-chip DRAM）load/store到Tile Register
   b. Remote atomic支持（RISC-V支持，应该）
   c. *Neighbour/remote load/store是否有必要？像卷积边缘等场景，有其他PE的访问能力会比较灵活，但确实增加了硬件复杂度，更长的L/S latency，更大的buffer等代价。
4. Img2Col/Col2Img的支持：
   a. RVV gather/scatter作为基本功能支持
   b. Tile core可对重要的pattern做直接支持，具体待分析后确定
5. MMA指令（Matrix）
   a. 支持B/C矩阵从L2B直接访问
6. SIMD指令（Elementwise/SFU），包括activation functions，量化等相关需求
7. 类型转换

其它根据情况再做判断添加。

PE间数据移动指令（DTE）

主要负责将数据在Host-PE L2B-3D DRAM与外部的数据移动，指令以做tensor级别的大块数据的移动。

其中，传输以3D block的方式时，Tensor需要通过Tensormap来描述。同时，传输的block通过坐标等信息指定。

另外也支持通过连续的，linear的方式来copy。

移动的模式，支持

- Copy
- Multicast

同步指令

RISC-V本身有fence指令。

1. 同thread的异步Engine间同步 （ per RV核 in SCTL单元）
2. Barrier： 类似threadblock sync，cluster sync，thread间在一个点做同步
3. Semaphore（counting semaphore，arrive/wait），用处比较多
    a. Produce-consumer
    b. Network transfer counting

2. 数据格式

- Tile core支持的数据类型：
    - Matrix
        - INT4，INT8，(W4A8)
        - FP8，FP16，BF16，TF32，
        - MXFP4，MXFP6，MXINT8 （TODO，下面的讨论先不包含这几个类型）
    - Vector额外支持类型：F16, BF16, FP32
- TELEN

    - The maximum size in bits of a tile core element，
    - 支持，4，8，16，32bit，
    - 对于MXFP（特别是MXFP6）需要特别考虑。
- TLEN
    - The number of bits in a single tile register,
    - 为了简单，先只支持8192bit，不支持像RVV一样过于灵活的组合

***TODO* MXFP数据格式，Sparse数据格式怎么表达?**

- 一个或两个data segment
- Bit length 非2的幂次

MMA data type

Data-type	Multiplicands (A or B)	Accumulator (D)
Floating Point	.f16	.f16, .f32
Alternate floating Point	.bf16	.bf16, .f32
Alternate floating Point	.tf32	.f32
Alternate floating Point	.fp8 （BF8: .e4m3, FP8: .e5m2）	.f16,.f32
8bit Integer	both .u8 or both .s8	.s32
4-bit integer	both .u4 or both .s4	.s32
MXINT8	b32/se8, INT8(.s8)	.bf16 / .fp32 (待定)
MXFP6	b32/se8, FP6(.e2m3,.e3m2)	.bf16 / .fp32 ((待定)
MXFP4	b32/se8, FP4(.e2m1)	.bf16 / .fp32 ((待定)

Table 29. Small Float Data Formats

FMT	Sign	Exp	Mant	Bias	+0	-0	Inf, -Inf	NaN, -NaN
FP16	1	5	10	15	0	0x8000	0x7c00, 0xfc00	normal
BF8 (E5M2)	1	5	2	16	0x00	0x00	0x80	0x80
FP8 (E4M3)	1	4	3	8	0x00	0x00	0x80	0x80

3. Tile Extension Programmer’ s Model

The tile extension adds 256vector registers, and several unprivileged CSRs to a base RISC-V ISA.

CSRs

TODO：add CSR register definitions

Tile CSRs

Address	Privilege	Name	Description
xxxx	URW	trstart	Tile register start row position
xxxx	URW	tcsr	Tile control and status register
xxxx	URO	tlenb	Tile register size in byte, mrows*trlenb, only 1024 for now
xxxx	URO	trlenb	RLEN (in bytes)
xxxx	URO	tisa	Tile instruction subset

对于Tile的数据类型以及作为matrix的时候的shape，比较倾向于跟随指令来动态指定，而不是使用CSRs。这样对于**混合精度，动态变换不同的shape**视角，比较有灵活方便，性能更好。

- Tile data type (指令encoding)
- **Tile matrix shape (CSR)**

ABI（Execution Context CSR）

在并行计算模型中，每个thread的执行有自己的context，这些context以CSR方式，允许shader instruction以及MMIO访问，包括

- thread idx，block idx，cluster idx，
- Thread dims, block dims, cluster dims, grid dims
- kernel resource
  - kerenl arg address, 指向user defined data
  - Shared memory base address
  - Private memory base address
  - Tile Register base index
  - Tile EMask?

注：

1. 有些context已经在RV CSR里包含了？比如，Start PC，VMID，等，这里是并行计算额外需要的CSR。
2. 有些context是指令不需要访问的，比如VF_ID/MIG_ID, logical Core ID，logical PE ID，logical Cluster ID。

TODO, add detailed ABI CSR into the table.

Address	Privilege	Name	Description
xxxx			
xxxx			
xxxx			
xxxx			
xxxx			

Tile Register

The Tile extension adds 256 architectural tile registers, tr0-tr255 to the base RISC-V ISA. Each tile register has a fixed TLEN bits of state.

Programmer view of tile register

Tile Register can be used for all Tile Core engines：tile matrix，tile vector(SFU & SIMD), and tile load/store.

Tile Register data can work in different views，根据所在的应用场景/指令。

- 矩阵：Matrix fragment with a certain shape，
- 向量：Vector of certain length，without shape info
- 一定长度的向量组：A group of vector elements，with certain vector length

Matrix view

RLEN (Row LENgth), is the length of each register row in bits, when the register is in MATRIX view.

Row number of the matrix is TLEN/RLEN.

Support shape of the matrix with one tile register:

	TLEN(bit)	RLEN(bit)	Matrix A		
			M	K (in ELEN of 32/16/8/4 bits)	
Matrix	8192	256	32	8/16/32/64	m32*n32 （32Byte，NV PTX）
	8192	512	16	16/32/64/128	**T-head支持 16*16 *(32bit)**
	8192	1024	8	32/64/128/256	

硬件实现每个cycle可以按照m8*k32(DP8/16/32/64)*n8的粒度来做指令内循环。

整体来说，我们的表达方式比较灵活，这样可以比较方便的在各种场景中选择高效的表达配置。

作为对比，T-head的matrix extension中，对于8192bit的TRLEN，只支持了M=16，K=16/32/64/128的一种模式。

Partial matrix representation

当我们需要表达的一个矩阵小于当前支持的matrix shape的时候，需要表达一个partial matrix。按照一些规则来表达：

- K（row）需要选择合适的shape，做align，对齐到某一个支持的RLEN
- M的选择也对应确定，可使用**EMASK（tile mask）**来指定特定的行做计算（P1优先级，具体使用待定）

Multiple register matrix

在很多的场景中，我们需要以多个寄存器来表达一个更大的Matrix。最常见的是M/K不变，但需要更大的ELEN的时候，整个matrix的bit位就超过了8192bit。因此就需要多个register来表示。

以上面标准矩阵32*32 ELEN=32bit的情况，需要4个Tile register来表示总共4KB的数据。

另外在动态matrix shape的支持中，我们也希望某一个matrix的长度是在一定范围内可变的，这样可减少使用MMA的指令条数。

TODO：Multiple Register的shape确定，为了减少硬件支持的难度，可做一定的限制。比如

1. **限制K维度保持不变。M可支持1-16.**

nreg[3:0]	register number
0	1
1	2
11	4
111	8
1111	16
others	reserved

如果matrix shape是 m32k32，matrix size(nreg)=16，则matrix register为m512k32。

🏆 指令在硬件的执行拆解举例，programmer不可见

```
for（int i=0, i<mat_b.nreg, i++)
{
    reg_a = read(mat_a);
    reg_b = read(mat_b.sub(i));
    o_index = 0;
```

```
        for (int m=0; m<reg_b.M/8; m++)
        {
            for (int n=0; n<reg_b.N/8; n++)
            {
                for (int k=0; k<reg_b.K/(256/TELEN); k++)
                {
                    reg_c[o_index]+=reg_a[m] DOT reg_b[n];
                }
                write(mat_c.sub(i), reg_c[o_index]);
                o_index++;
            }
        }
    }
```

Vector view

一个tile register可以当作一个vector，vector可以认为是一个matrix的特例，TLEN=RLEN=8192。

如果TELEN=16，则nELEM=8192/16=512

Vector view，可以认为是SIMD的工作模式。

Vector Group View

TBD

将一个tile register当作是一个等长vector的group。整体的概念比较类似Matrix。。。

比如，32个32Byte长的vector。

比如，8个128Byte长的vector。分别用index 0~7来取数。

可由tile mask来做条件执行。可以通过index来对其中的某个子vector做操作。

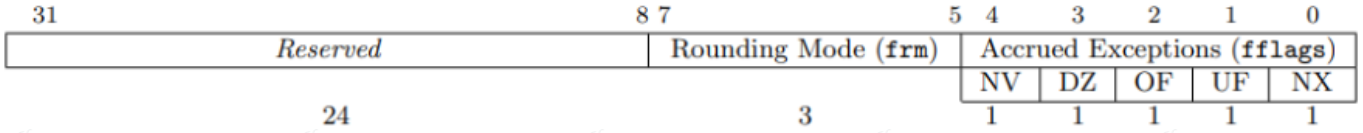
使用场景包括

- RVV register 与 Tile register之间move数据。指定vector register到tile register摸一个子tensor。

Rounding mode & saturation

- Floating-point rounding mode

The float matrix multiplication reuses the floating-point control and status register, **fcsr**, to select the dynamic rounding mode for floating-point arithmetic operations and hold the accrued exception flags.



rounding mode	Mnemonic	Meaning
000	RNE	Round to Nearest, ties to Even
001	RTZ	Round towards Zero
010	RDN	Round Down (towards -∞)
011	RUP	Round Up (towards +∞)
100	RMM	Round to Nearest, ties to Max Magnitude
101		Invalid. Reserved for future use
110		Invalid. Reserved for future use
111		Invalid in rounding mode register

- Fixed-point rounding mode and saturate flag

参考RVV 1.0定义。

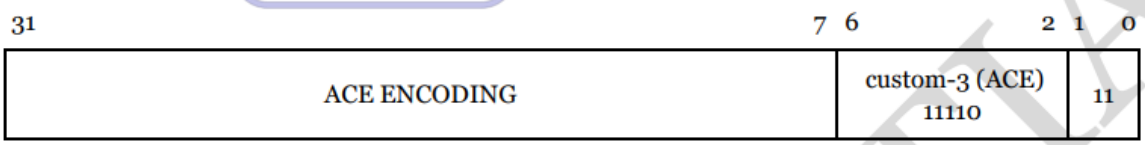
4. Tile Instruction Formats

指令格式

Andes ACE instruction

我们需要基于Andes ACE 来扩展指令，因此需要遵循ACE instruction encoding，如下：

In the AndeStar V5 architecture, an ACE instruction is a 32-bit instruction with the following format:



由于Tile register较多，并且Tile instruction比较复杂，需要的信息较多，我们决定都使用64bit的指令格式。因为需要使用Andes ACE instruction做扩展，64bit tile instruction由两个32bit ACE instruction拼成。Opcode都是1111011，不过，Andes的ACP，ASP等ACE扩展指令已经占用了一些指令encoding空间，比如，ace_bsync占用了b‘14~12=111的空间

因此，可能实际有用的bit是64-14-3=**47bit**，来表达每一条指令的实际信息。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Low																										1	1	1	1	0	1	1
High																		1	1	0						1	1	1	1	0	1	1

实际encoding空间还在研究，看怎么把Andes占用的空间尽量缩小。

注 是否指定Low 32bit地址在偶数DWORD地址上？

指令tuop总结

tuop: category of tile instruction op

- 000: TGMEM: MEMORY operations on global memory
- 001: TSMEM: MEMORY operations on shared memory
- 010: TCTRL: Conf registers & control
- 011: TMMA
- 100: TALU
- 101:
- **110: High 32bit**
- **111: ace_bsync/ace_nbsync/st_rd，以及ASP指令等**

指令操作数（Operand）

1. Scalar

a. Scalar Register
b. Immediate data

2. Vector (RVV vector register)
   a. 需确定哪些VGPR能够被tile core 访问
   b. LMUL (multiple VRF) 是否起作用
3. Tile (Tile register)

5. Configuration-setting Instructions

需要一些指令来操作tile CSR register。比如

- 使用上面的index，dimention，thread_id之类来计算地址，像将其读到GPR里。
- 一些可写的CSR，可以使用Tile CSR指令修改

Instruction Category		**TCTRL**
Instruction Name		**TCTRL.tcsr**
Field Name	**#bits**	**Description** **read or write Tile core CSR（from GPR）**
opcode	7	
tuop	3	b'010
ctrluop	3	control micro opcode **- tcsr:** - tsync: - twait - tsemaphore
csruop	2	register move micro opcode **- 0: scalar to tile csr** **- 1: imm to tile csr, source is imm, ignore num, use all left bits as imm dat** **- 2: tile csr to scalar**
rd	5	source or destination of CSR
rs0	5	destination or source GPR register
num	3	number of consecutive GPR(CSR) to read/write
imm	32	imm data to write to writable CSR

6. Tile Loads and Stores

Tile load直接从Global memory（TGMEM指令）或者shared memory（TSMEM指令）读数据，装置到Tile Register里。而Tile store将tile register的数据写到Global memory，或者shared memory。

根据load/store数据在memory的位置，Tile memory load store分为：

1. TGMEM，tile global memory load store
2. TSMEM，tile shared memory load store

指令\Scope	PE	Cluster	Chip	Multi-Chip	Host
SLSU-SMEM	M				
VLSU-SMEM	M				
TLSU-SMEM	M/A	M/A	M/A	A(MBAR)	
TACP-SMEM	M	M	M	M	
SLSU-GMEM	M		M		
VLSU-GMEM	M		M		
TLSU-GMEM	M/A		M/A	M*/A	M/A*
TACP-GMEM	M		M	M	M
M	memory access				
A	atomic				

根据load/store数据的layout，数据组成，可分为

1. bulk（TLEN长的连续地址的数据）
2. Row strided（TLEN长的，single strided的数据，stride来源于scalar register / GPR）
3. indexed（per row address offset with VREG(vector register)，可以读入无规则的离散的每行数据。）
4. ~~img2col（Tensor based img2col 支持，只支持从shared memory读写。~~

- Tile mask，指定哪些row需要读/写，
- Register number, 即 tile number，一次读写的tile数量，用来读写更大量的数据，暂可用1/2/4
- row-start？暂时不支持，应该可以用tile mask来简介达到目的
- Memory hint
  - Streaming？Not needed for now，unless we change the memory hierarchy

TGMEM

Instruction Category		TGMEM
Instruction Name		**TGMEM**
Field Name	**#bits**	**Description** **Memory operations on global memory, including** **load from/store to global memory space;**
opcode	7	
tuop	3	b'000
memuop	6	**memory micro opcode：** **bit 5~3: load, store,misc, atomic, etc.** **bit 2~0: formats**
lsuop	2	**for normal load/store micro opcode, when memuop=LD_B32/ST_B32** 0x0: unit stride 0x1: strided, 32B elements 0x2: indexed, 32B elements **for atomic micro opcode, only valid for memuop is an atomic op** **bit 0:** 0x0: scalar atomic operation, source & dst memory are all scalar, MAX size is 6 0x1: atomic on vector memory, source is a tile register (can be partial with tm **bit 1:** 0x0: no return 0x1: return **for block/tile load store micro opcode memuop=LD_BLK/ST_BLK** 0x0: non MX format (no header) 0x1: mx48 0x2: mx6 0x3: reserved, sparse
regdata	8	load destination or store source tile register, 如果.size>0，td是连续多个register的开始 atomic只支持size=0，rd是atomic load dst GPR for scalar CAS atomic op, additional rd+1用来作为交换值（new value）
sbase	5	rs1, base src address GPR index
		for normal load/store micro opcode, when memuop=LD_B32/ST_B32 addressing for different data pattern **0. unit stride，** for MX formats, use 5bit imm data as: （only for sparsity or compression) - bit 0：packed tile with header - bit 1~3：data body size of 256B*n(0~6) for other formats, it an imm offset of 32B elements, this is low 5 bits, imm_off **1. value in rs2 as the stride in 32B, if lsuop==strided** **2. index vector register, if lsuop==indexed, each as an address for 32B ele** **for atomic micro opcode,** 3. scalar value in rs2 for atomic 比较值source

opinfo(rs2)	5	**for block/tile load store micro opcode memuop=LD_BLK/ST_BLK** - bit 0-1: start sub-tile index (0/1/2/3) - bit 2: strided access enable, 0 - no strided, 1 - strided - bit 3: strided mode, 0, 512B stride, 1, 1024B strided
tilesize /atom	3	tilesize:: number of 2^n tile are loaded, used for - unit stride of LD_B32/ST_B32 - LD_BLK/ST_BLK, 只支持1, 2, 4个tile
	2	vec_size：x1, x2, x4, (x8 reserved) for scalar atomic (64bit)
	1	reuse: atm:use scalar 扩展给所有component做atomic
tmask	1	tile execution mask is enabled,only valid size==0, for partial tile, not valid for memuop=LD_BLK/ST_BLK
regoffseten	1	enable offset from scalar register
offset(rs3)	5	rs3: 1. for block load store(memuop=LD_BLK/ST_BLK), it is an offset in number of offsetAddr=offset*byte_of_a_block 2. for offset in bytes, directly add to base address; 3. Imm_offset[9:5], for LD_B32/ST_B32 & unit_stride & offseten=0;

Memuop definition

global						
	0	1	2	3	4	
	load	**store**	**misc**	**add**	**min**	**max**
0	LD_B32	ST_B32	COMMIT_MBAR?	ADD_U32	MIN_U32	MAX_U32
1	LD_BLK	ST_BLK	COMMIT_GROUP	ADD_I32	MIN_I32	MAX_I32
2			WBINV?	ADD_F32	MIN_F32	MAX_F32
3			TACP	ADD_BF16	MIN_BF16	MAX_BF16
4				ADD_FP16	MIN_FP16	MAX_FP16
5				ADD_U64	MIN_U64	MAX_U64
6				ADD_I64	MIN_I64	MAX_I64
7						

Alignment:

- LD/ST Address 32B alignment
- ATOM/MBAR, 8B alignment

Load/Store

1. LD_B32/ST_B32a

2. LD_BLK/ST_BLK

Atomic

TSMEM

Instruction Category		TSMEM
Instruction Name		**TSMEM**
Field Name	**#bits**	**Description** **Memory operations on shared memory, including load from/store to shar space, atomic, mbar**
opcode	7	
tuop	3	b'001
memuop	6	**memory micro opcode：** **bit 5~3: load, store,misc, atomic, etc.** **bit 2~0: formats**
lsuop	2	**for normal load/store micro opcode, when memuop=LD_B32/ST_B32** 0x0: unit stride 0x1: strided, 32B elements 0x2: indexed, 32B elements **for atomic micro opcode, only valid for memuop is an atomic op** **bit 0:** 0x0: scalar atomic operation, source & dst memory are all scalar, MAX size is 6 0x1: atomic on vector memory, source is a tile register (can be partial with tm **bit 1:** 0x0: no return 0x1: return **for block/tile load store micro opcode memuop=LD_BLK/ST_BLK** 0x0: non MX format (no header) 0x1: mx48 0x2: mx6 0x3: reserved, sparse
regdata	8	load destination or store source tile register, 如果.size>0，td是连续多个register的开始 atomic只支持size=0，rd是atomic load dst GPR for scalar CAS atomic op, additional rd+1用来作为交换值（new value）
sbase	5	rs1, base src address GPR index
		for normal load/store micro opcode, when memuop=LD_B32/ST_B32 addressing for different data pattern **0. unit stride，** for MX formats, use 5bit imm data as: （only for sparsity or compression) - bit 0：packed tile with header - bit 1~3：data body size of 256B*n(0~6) for other formats, it an imm offset of 32B elements, this is low 5 bits, imm_off **1. value in rs2 as the stride in 32B, if lsuop==strided** **2. index vector register, if lsuop==indexed, each as an address for 32B ele** **for atomic micro opcode,** 3. scalar value in rs2 for atomic 比较值source

opinfo(rs2)	5	**for block/tile load store micro opcode memuop=LD_BLK/ST_BLK** - bit 0-1: start sub-tile index (0/1/2/3) - bit 2: strided access enable, 0 - no strided, 1 - strided - bit 3: strided mode, 0, 512B stride, 1, 1024B strided
tilesize /atm	3	tilesize:: number of 2^n tile are loaded, used for - unit stride of LD_B32/ST_B32 - LD_BLK/ST_BLK, 只支持1, 2, 4个tile
	1	vec_size：x1, x2, x4, (x8 reserved) for scalar atomic (64bit)
	1	reuse: atm:use scalar 扩展给所有component做atomic
tmask	1	tile execution mask is enabled,only valid size==0, for partial tile, not valid for memuop=LD_BLK/ST_BLK
regoffseten	1	enable offset from scalar register
offset(rs3)	5	rs3: 1. for block load store(memuop=LD_BLK/ST_BLK), it is an offset in number of offsetAddr=offset*byte_of_a_block 2. for offset in bytes, directly add to base address; 3. Imm_offset[9:5], for LD_B32/ST_B32 & unit_stride & offseten=0;

Memuop

TSMEM								
	0	1	2	3	4	5	6	
	load	**store**	**misc**	**add**	**min**	**max**	**logic**	**logic**
0	LD_B32	ST_B32	MBAR_ARRIVE	ADD_U32	MIN_U32	MAX_U32	AND_B32	INC_U3
1	LD_BLK	ST_BLK	MBAR_ARRIVE_DROP	ADD_I32	MIN_I32	MAX_I32	OR_B32	DEC_U
2				ADD_F32	MIN_F32	MAX_F32	XOR_B32	SWAP_
3				ADD_BF16	MIN_BF16	MAX_BF16		CAS_B3
4				ADD_FP16	MIN_FP16	MAX_FP16		
5				ADD_U64	MIN_U64	MAX_U64		
6				ADD_I64	MIN_I64	MAX_I64		
7								

MBAR（MBAR_ARRIVE/MBAR_ARRIVE_DROP）

Memory Barrier is a 64bit counter in Shared Memory(L2B).

- 3bit phase
- 20bit: expected counter
- 20bit: pending counter

- 21bit: pending transaction counter

7. Tensor Async Copy Acceleration

Tensor memory async copy, 用于大块的tensor数据搬移。具体包括：

1.

- Load tensor memory from local or remote global memory to local shared memory
- Store tensor memory from local shared to global

2.

- Copy from local shared memory to remote shared memory

3. Device to device in the same cluster or from/to different cluster

- Copy from local DRAM to local or remote DRAM

Instruction Category		**TGMEM**
Instruction Name		**TACP**
Field Name	**#bits**	**Description:** **asynchronous copy data from host/dram/L2B to L2B/DRAM/Host, tensor相都在tensormap中，**
opcode	7	
tuop	3	b'000
memuop	6	同TGMEM
acpuop	2	TACP micro opcode: 0x0: Contiguous Linear copy, 不需要tensormap，只需要地址和联系copy的tens 数据可以是linear layout，也可以是tiled layout。(常用场景DRAM间搬大块连续数 移动数据) 0x1: tile conversation, tiled copy with linear tensormap, 一个tile一个tile的数据 据在tiled和linear之间做转换。（常用场景，host / device之间的数据搬移，DRAM linear数据格式转换和搬移) 0x2: tiled tensor map, tiled copy, with tiled tensormap, 一个tile一个tile的数据 Dst至少有一个是tiled tensormap，不是tensormap的将以linear的方式排列。 0x3: tensor map linear, linear copy with tensormap, source 和 destination 都是
dsttm	1	dst memory is described with tensor map
srctm	1	src memory is described with tensor map
dst	5	the GPR of addr of dest memory, or tensormap of dest tensor
src	5	the GPR of addr of src memory, or tensormap of src tensor
coord/size	5	vector register of src&dest coordinates or GPR
~~dim~~	~~2~~	~~2D~5D~~
dstloc	2	DST memory location - 0: shared::cta (local L2B) - 1:global（same or different cluster） - 2: shared::cluster(remote L2B)
srcloc	1	SRC memory location - 0: shared::cta (local L2B) - 1:global（same or different cluster）
completion	2	the completion mechanism： - 0：bulk - 1：mbar - 2：semaphore
multicastl2	1	multicast to L2B of same cluster. Only valid for global to cta and 4 PEs for now
completeobj	5	the GPR of addr of mbar or the index of semaphore（如果在TSYnc中加了sema

Tensormap description

最新参考 SiPU 1.0 DTE架构设计方案 v0.5

8. Matrix Multiply-Accumulate

完成矩阵乘加运算：

D = A * B + D

D = A * B, where the input from accumulator D is disabled.

Macuop definition draft（TBD）

Data-type	Multiplicands (A)	Multiplicands (B)	Accumulator (D)
Alternate floating Point	.tf32	.tf32	.f32
Alternate floating Point	.bf16	.bf16	.f32
Floating Point	.f16	.f16	.f32
Alternate floating Point	.fp8(.e4m3)	.fp8(.e4m3)	.f32
Alternate floating Point	.fp8(.e5m2)	.fp8(e5m2)	.f32
Floating Point	.f16	.f16	.f16
Alternate floating Point	.bf16	.bf16	.bf16
Alternate floating Point	.fp8(.e4m3,.e5m2)	.fp8(.e4m3,.e5m2)	.f16
Integer	both .u8 or both .s8	both .u8 or both .s8	.s32
4-bit integer	both .u4 or both .s4	both .u4 or both .s4	.s32
8/4-bit integer	both .u8 or both .s8	both .u4 or both .s4	.s32
MXINT8	MXINT8	MXINT8	.f32
MXINT4	MXINT4	MXINT4	.f32
MXINT8/4	MXINT8	MXINT4	.f32
MXFP6	MXFP6(.e3m2,.e2m3)	MXFP6(.e3m2,.e2m3)	.f32
MXFP4	MXFP4(e2m1)	MXFP4	.f32

支持的matrix shape

FP32/S32	A tile size	B tile size	D tile size		BF16/FP16	A tile size	B tile size	D tile size
32*32	mxk	kxn	mxn		32*32	mxk	kxn	mxn
	1x1	1x**1**	1x4			1x1	1x**1**	1x2
		1x**2**	1x8				1x**2**	1x4
		1x**4**	1x16				1x**4**	1x8
		1x**8**	1x32				1x**8**	1x16
	1x**1**	1x1	1x4			1x**1**	1x1	1x2
	1x**2**	2x1	1x4			1x**2**	2x1	1x2
	1x**4**	4x1	1x4			1x**4**	4x1	1x2
	1x**8**	8x1	1x4			1x**8**	8x1	1x2

FP32/S32	A tile size	B tile size	D tile size		BF16/FP16	A tile size	B tile size	D tile size
16*64	mxk	kxn	FP32/S32		16*64	mxk	kxn	BF16/FP16
	1x1	1x**1**	1x1			1x1	1x**2**	1x1
		1x**2**	1x2				1x**4**	1x2
		1x**4**	1x4				1x**8**	1x4
		1x**8**	1x8				1x**16**	1x8
	1x**1**	1x1	1x1			1x**1**	1x2	1x1
	1x**2**	2x1	1x1			1x**2**	2x2	1x1
	1x**4**	4x1	1x1			1x**4**	4x2	1x1
	1x**8**	8x1	1x1			1x**8**	8x2	1x1

FP32/S32	A tile size	B tile size	D tile		BF16/FP16	A tile size	B tile size	D tile
8*128	mxk	kxn	FP32/S32		8*128	mxk	kxn	BF16/FP16
	1x1	1x**4**	1x1			1x1	1x**8**	1x1
		1x**8**	1x2				1x**16**	1x2
		1x**16**	1x4				1x**32**	1x4
		1x**32**	1x8				1x**64**	1x8
	1x**1**	1x4	1x1			1x**1**	1x8	1x1
	1x**2**	2x4	1x1			1x**2**	2x8	1x1
	1x**4**	4x4	1x1			1x**4**	4x8	1x1
	1x**8**	8x4	1x1			1x**8**	8x8	1x1

TODO：特殊的format

1. MXFP6

SiPU MX Format端到端方案v0.5

特殊的format，tile shape 32*128。为了简单，MMA的shape数可以减少一些。

2. A8W4 （MXINT8 - MXINT4），对K方向对齐，就需要matrix A的tile size在K方向double。

TMMA

All operands are all read from the tile register.

Instruction Category		**TMMA**
Instruction Name		**TMMA**
Field Name	**#bits**	**Description：Matrix Multiply-Accumulate** **D = A * B + D** **D = A * B, where the input from accumulator D is disabled.**
opcode	7	
tuop	3	b'011
mmauop	5	mma micro opcode: different src/dst format refer to the mmauop table
dst	8	destination tile register
srca	8	tile reg of matrix A
srcb	8	tile reg of matrix B
nummma	3	number of 2^n of MMA micro op
multimode	1	b'0: A & B, output stationary, both extend in K direction (column) of matrix b'1: reserved
reusea	1	reuse tile reg srca,’ Reuse“是对下一条指令的优化提示，表明下一条和当前指令 存器复用当前这条指令。
reuseb	1	reuse tile reg srcb
reused	1	reuse td (td reused as ts3)
noacc	1	the input from accumulator D is disabled. 如果K方向多个tile（multitile=1, tilesize>1)，只有第一个不输入acc做累加。后续
transmata	1	Matrix A is transposed, only for 8bit/32*32
transmatb	1	Matrix B is transposed, only for 8bit/32*32
neg1	1	source 1 negative

TMMA mem（advanced feature）

Instruction Category		**TMMA**
Instruction Name		**TMMA_mem**
Field Name	**#bits**	**Description：Async Matrix Multiply-Accumulate with data from shared m D = A * B + D D = A * B, where the input from accumulator D is disabled. A和B至少一个会从Shared memory直接读取。当矩阵从shared memory读取的 的matrix descriptor来指定matrix的属性。**
opcode	7	
tuop	3	b'011
mmauop	5	mma micro opcode: different src/dst format refer to the mmauop table
dst	8	destination tile register
srca	8	tile reg of matrix A, or vector reg of matrix descriptor
srcb	8	tile reg of matrix B, or vector reg of matrix descriptor
nummma	3	number of 2^n tiles of matrix B(or A&B)
multimode	1	b'0: A & B, output stationary, both extend in K direction (column) of matrix b'1: reserved
reusea	1	reuse tile reg srca,’ Reuse“是对下一条指令的优化提示，表明下一条和当前指 dst 寄存器复用当前这条指令。
reuseb	1	reuse tile reg srcb
reused	1	reuse td (td reused as ts3)
noacc	1	the input from accumulator D is disabled. 如果K方向多个tile（multitile=1, tilesize>1)，只有第一个不输入acc做累加。后续
transmata	1	Matrix A is transposed, only for 8bit/32*32
transmatb	1	Matrix B is transposed, only for 8bit/32*32
neg1	1	source 1 negative

TODO: add matrix descriptor format

详细定义参见：

[legacy] MMA Async and Matrix Descriptor

9. Tile Vector Arithmetic

***draft* TO BE discussed**

Tile format conversion

数据格式转化指令，特别的，包含非MXformat转换成MXformat的quantization。

1. MXformat quantization, 多个tile转换成一个MX tile format
   - FP32，4个tile 转换
2.

Instruction Category		TALU_CVT
Instruction Name		**TCVT**
Field Name	**#bits**	**Description** **format conversion, including MXFormat conversion(quantization)**
opcode	7	
tuop	3	b'100
vecuop1	2	different vector micro opcode: **- talu_cvt (conversion/quantization)** - talu_1src （SFU） - talu_2src - talu_3src
vecuop2	6	src format & des format: 见下表
dst	8	destination tile register
src1	8	tile or scalar register of source 1
src1_type	2	source type - tile register - a integer GPR - a float register
rmode	2	rounding mode：reserved b'000: RNE b'001: RTZ b'010: RDN b'011: RUP
sat?	1	Saturation: b'00: sat b'01: relu b'10: satfinite
reuse1	1	reuse ts1
neg1	1	1 means to change source 1 negative, used only for FP data type
tmask	1	tile execution mask is enabled

vecuop2的定义：

vecuop2[5:2]	vecuop2[1:0]=b'00		vecuop2[1:0]=b'01		vecuop2[1:
b'0000	TCVT_F32_MXFP8_E4M3	4:1	TCVT_BF16_MXFP8_E4M3	2:1	TCVT_F16_MXFP8_
b'0001	TCVT_F32_MXFP8_E5M2	4:1	TCVT_BF16_MXFP8_E5M2	2:1	TCVT_F16_MXFP8_
b'0010	TCVT_F32_MXFP6_E3M2	16:3	TCVT_BF16_MXFP6_E3M2	8:3	TCVT_F16_MXFP6_
b'0011	TCVT_F32_MXFP6_E2M3	16:3	TCVT_BF16_MXFP6_E2M3	8:3	TCVT_F16_MXFP6_
b'0100	TCVT_F32_MXFP4	8:1	TCVT_BF16_MXFP4	4:1	TCVT_F16_MXFP4
b'0101	TCVT_F32_MXINT8	4:1	TCVT_BF16_MXINT8	2:1	TCVT_F16_MXINT8
b'0110	TCVT_F32_MXINT4	8:1	TCVT_BF16_MXINT4	4:1	TCVT_F16_MXINT4
b'0111	TCVT_F32_FP8_E4M3	4:1	TCVT_BF16_FP8_E4M3	2:1	TCVT_F16_FP8_E4
b'1000	TCVT_F32_FP8_E5M2	4:1	TCVT_BF16_FP8_E5M2	2:1	TCVT_F16_FP8_E5

1 Operand Vector ALU

主要是一个source operand，一个dst operand的指令类型，包括

- **Sfu special function**
- Matrix transpose

Instruction Category		**TALU_1OP**
Instruction Name		**Txxx**
Field Name	**#bits**	**Description** **1OP tile ALU operation, including special functions, mov, etc.**
opcode	7	
tuop	3	b'100
vecuop1	2	different vector micro opcode: - talu_cvt (conversion/quantization) **- talu_1src （SFU，transpose，etc）** - talu_2src - talu_3src
vecuop2	6	sub micro opcode，ALU op: **sigmoid/sin/cos/tanh/exp2/log2/rcp/sqrt/rsqrt/exp** - 11:abs, etc.
dst	8	destination tile register
src1	8	tile reg of source 1
src1_type	2	source type - tile register - a integer GPR - a float register - immediate
imm	32	additional bits for imm data(including 8bit of ts1)
reuse1	1	reuse ts1
neg1	1	1 means to change source 1 negative, used only for FP data type
tm	1	tile execution mask is enabled

vecuop2的定义：

vecuop2[5:2]	vecuop2[1:0]=b'00	vecuop2[1:0]=b'01	vecuop2[1:0]=b'10	vecuop2[1:0]=b'11
b'0000	TALU_SGMD_F32	TALU_SGMD_BF16	TALU_SGMD_F16	TALU_MAT_TRANS_E
b'0001	TALU_SIN_F32	TALU_SIN_BF16	TALU_SIN_F16	TALU_MAT_TRANS_E
b'0010	TALU_COS_F32	TALU_COS_BF16	TALU_COS_F16	TALU_MAT_TRANS_E
b'0011	TALU_EXP2_F32	TALU_EXP2_BF16	TALU_EXP2_F16	TALU_MAT_TRANS_E
b'0100	TALU_LOG2_F32	TALU_LOG2_BF16	TALU_LOG2_F16	TALU_MAT_TRANS_E
b'0101	TALU_EXP_F32	TALU_EXP_BF16	TALU_EXP_F16	TALU_MAT_TRANS_E
b'0110	TALU_RCP_F32	TALU_RCP_BF16	TALU_RCP_F16	
b'0111	TALU_SQRT_F32	TALU_SQRT_BF16	TALU_SQRT_F16	
b'1000	TALU_RSQRT_F32	TALU_RSQRT_BF16	TALU_RSQRT_F16	
b'1001	TALU_TANH_F32	TALU_TANH_BF16	TALU_TANH_F16	

关于transpose：暂时支持8-16-32 bit的数据格式，不包括MXFormat。

1. 8bit格式，支持M32N32转置成M32N32
2. 16bit格式，支持M16N32和M32N16之间的转置，也就是说，source/destination都是一个tile register，但tile mode不一样。
3. 32bit格式，支持M8N32和M32N8之间的转置，同样，source/destination都是一个tile register，但tile mode不一样。

Tile Move Instructions

TMV: Tile Move instructions, 完成数据在Scalar，Vector，Tile三个unit/寄存器之间的数据移动。

Instruction Category		TALU_2OP
Instruction Name		**TMV**
Field Name	**#bits**	**Description** **move data between GPR, VRF, TRF** **支持VMUL > 1**
opcode	7	
tuop	3	b'100
vecuop1	2	different vector micro opcode: - talu_cvt (conversion/quantization) - talu_1src （SFU，etc.） **- talu_2src (MV, ADD，MUL，…)** - talu_3src
vecuop2	6	sub micro opcode，ALU op: register move micro opcode **- 0: mv2t, move to tile register** **- 1: t2v, move from tile register** vector以B32的方式 scalar的方式，参考rvv的GPR-VRF间的方式做符号扩展。
tdreg	8	destination or source tile register
src1	5	source or destination of scalar or vector register
offset	5	(src2) index of the moved vector in tile register offset in tile register=index*VLEN/8，128B的粒度，offset=0~7
src1_type	2	source type - a integer GPR - a float register - source 1 is a vector register - immediate
offset_imm	1	offset is an imm
bc	1	broadcast to all row（rs2=0）

2 Operand Vector ALU

vecuop2[5:2]	vecuop2[1:0]=b'00	vecuop2[1:0]=b'01	vecuop2[1:0]=b'10	vecuop2[1:0]=b'11
b'0000	TALU_ADD_FP32	TALU_ADD_BF16	TALU_ADD_FP16	
b'0001	TALU_MUL_FP32	TALU_MUL_BF16	TALU_MUL_FP16	
b'0010				
b'0011				
b'0100				
b'0101				
b'0110				
b'0111				
b'1000	TALU_MIN_FP32	TALU_MIN_BF16	TALU_MIN_FP16	
b'1001	TALU_MAX_FP32	TALU_MAX_BF16	TALU_MAX_FP16	
b'1010				

Instruction Category		**TALU_2OP**
Instruction Name		**Txxx**
Field Name	**#bits**	**Description** **2OP tile ALU operation, including ADD/MUL/MIN/MAX etc.**
opcode	7	
tuop	3	b'100
vecuop1	2	different vector micro opcode: - talu_cvt (conversion/quantization) - talu_1src （SFU，etc.） **- talu_2src (MV, ADD，MUL，…)** - talu_3src
vecuop2	6	sub micro opcode, ALU op - ADD/MUL/MIN/MAX -
dst	8	destination tile register
src1	8	tile or scalar reg of source 1
src2	8	tile reg of source 2
src1_type	2	source 2 type - tile register - a integer GPR - a float register
reuse1	1	reuse ts1
reuse2	1	reuse ts2
neg1	1	source 1 negative
neg2	1	source 2 negative
tm	1	tile execution mask is enabled

3 Operand Vector ALU

Instruction Category		**TALU_3OP**
Instruction Name		**Txxx**
Field Name	**#bits**	**Description** **2OP tile ALU operation, including ADD/MUL/MIN/MAX etc.**
opcode	7	
tuop	3	b'100
vecuop1	2	different vector micro opcode: - talu_cvt (conversion/quantization) - talu_1src （SFU，etc.） - talu_2src (MV, ADD，MUL，…) **- talu_3src**
vecuop2	4	sub micro opcode, ALU op - FMA
dst	8	destination tile register
src1	8	tile or scalar reg of source 1
src2	8	tile reg of source 2
src3	8	tile reg of source 3
src1_type	2	source 2 type - tile register - a integer GPR - a float register
reuse1	1	reuse ts1
reuse2	1	reuse ts2
reuse3	1	reuse ts3
neg1	1	source 1 negative
neg2	1	source 2 negative
neg3	1	source 3 negative
tm	1	tile execution mask is enabled

vecuop2[5:2]	vecuop2[1:0]=b'00	vecuop2[1:0]=b'01	vecuop2[1:0]=b'10	vecuop2[1:0]=b'11
b'0000	TALU_FMA_FP32	TALU_FMA_BF16	TALU_FMA_FP16	
b'0001				
b'0010				
b'0011				
b'0100				
b'0101				
b'0110				
b'0111				
b'1000				
b'1001				
b'1010				

10. Synchronization

包括多种形式的同步功能。

t_wait

Andes ACE provide **ace_bsync & ace_nbsync** for data dependency between ACE and CPU (RV&RVV).:

- Blocking type (**ace_bsync**(SYNC_GROUP_NAME)): Instructions will stall the CPU pipeline, and no instructions will be issued until all specified target ACE instructions are finished.
- Non-blocking type (**ace_nbsync**(SYNC_GROUP_NAME): Instructions will stall the ACE pipeline, but not the CPU pipeline, from issuing subsequent instructions until **all specified target ACE instructions are finished.** As a result, CPU pipeline might be stalled later due to the back-pressure stall from the ACE pipeline.

这两条指令能满足一些基本的同步需求，但表达和效率可能不够好。我们需要做更精细的异步控制，类似nvida CUDA的Async-group机制。

Instruction Category		MISC
Instruction Name		**TCTRL**
Field Name	**#bits**	**Description** **Thread dependency check, including memory and other fences or operat**
opcode	7	
tuop	3	0x0
ctrluop	3	control micro opcode - tsync - **twait** - tsemaphore
waitop	2	0x0: TGMEM/TSMEM组合 0x1: TACP/commit group组合 0x2: wait one special operation, the operation type is specified by next 4 bits b'0000: b'0001:
tgld	1	
tgst	1	
tsld	1	
tsst	1	
tgld_cnt/ TACP/ async_mma	6	
tgst_cnt	6	
tsld_cnt	6	
tsld_cnt	6	

t_sync

Instruction Category		MISC
Instruction Name		**TCTRL**
Field Name	**#bits**	**Description** **Barrier synchronization within the scope of CTA or cluster. Support both n blocking and blocking barrier.**
opcode	7	
tuop	3	0x0
ctrluop	3	control micro opcode **- tsync:** - twait - tsemaphore
nonblk	1	0：blocking，(arrive&wait) 1：non-blocking（arrive)
scope	2	CTA，cluster
rs0	5	scalar register with sync id and number of sync threads
directid	1	0: use direct syncid
syncid	5	synchronizer id in tsync
imm	1	number of sync threads is an imm src0
immsrc0	8	src0, number of sync threads

t_fence