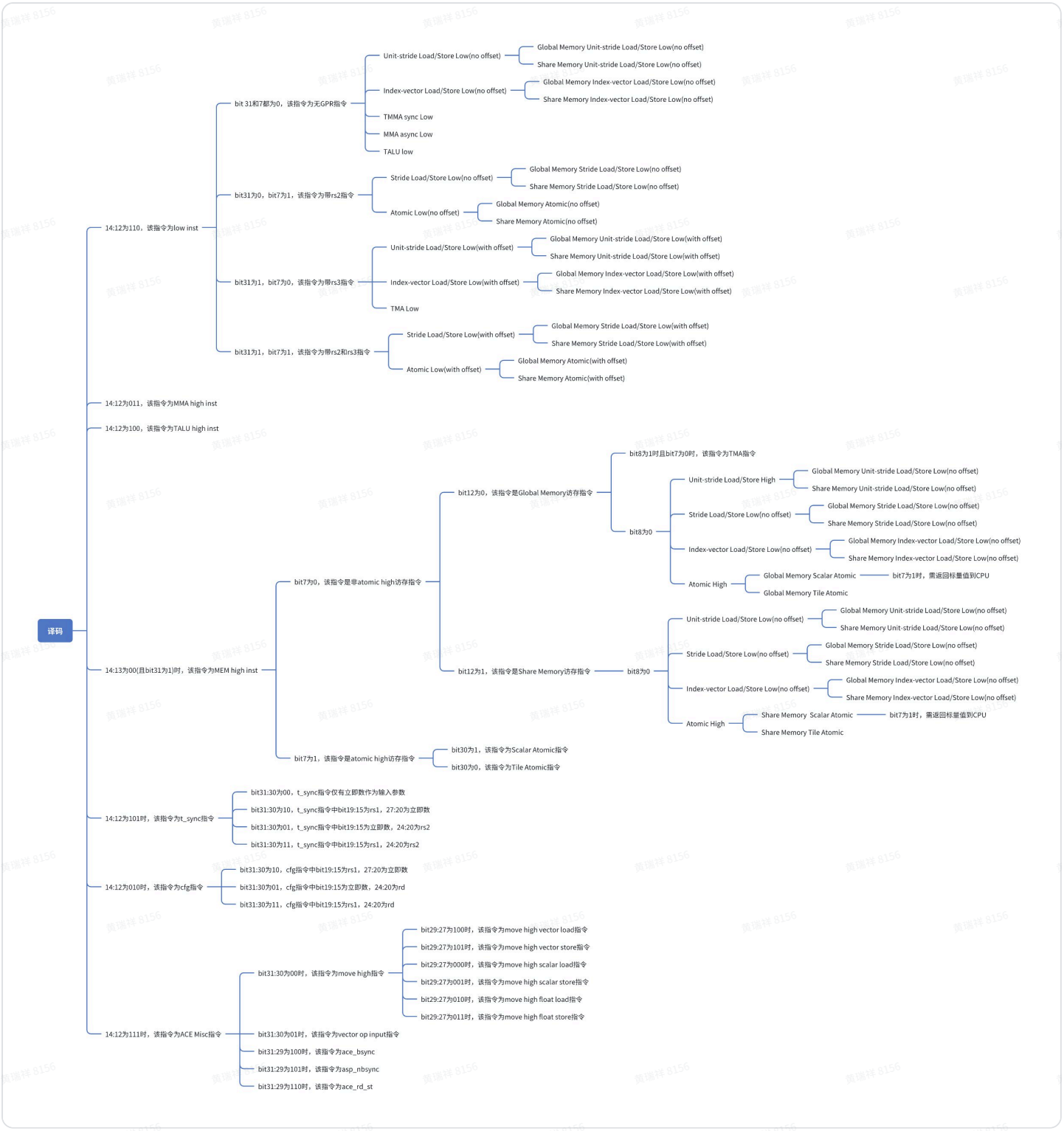
Tile Extension encoding 方案

> 🌟 http://isa.siorigin.com/这里有encoding的最新版本
>
> http://isa.siorigin.com/tile-extension/tile-extension.zip中可下载encoding文件

编码规则

> 🏕️ 根据是否有标量和向量寄存器的输入输出，确定指令种类。

该种类和 RISC-V Tile Extension 扩展指令集 不完全对应，存在一对多的情况。下方的思维导图中列出根据不同 tuop 使用标量寄存器输入输出的情况。向量寄存器在使用时会加入 ASP load/store 指令，仅将寄存器索引通过命令传输到 Tile Core。



TMA completion (lsuop:00/10) 建议 bulk和 semaphore 用这两个 encode，这样就能保证 completeobj用到mbar时，24:20 的 GPR 输入有效	memuop[5:3] 建议 4-7 表示所有的 atomic 指令，即 100-111 这样 bit4 就可以代表 atomic enable	Tuop 建议 TGMEM 和 TSMEM 的 14:12 encoding 改为 000 和 001，这样可以用两个 00 判定是否为访存指令 建议 Configuration Setting 的 14:12 encoding 改为 010。	Lsuop 00unit-stride 10 index-vector 01 stride

				11 atomic

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
inst_low_no_gpr	0														
inst_low_rs2	0														rs
inst_low_rs3	1									rs3					
inst_low_rs2_rs3	1									rs3					rs
inst_high_mma						td								ts1	
inst_high_talu_op	0					td								ts1	
inst_high_mem	1				ts1/td										rs
inst_high_mem_atomic_scalar_with_return	1						rd								rs
inst_high_mem_tma	1									rs2					rs
t_sync_no_gpr	0	0						imm2							imr
t_sync_rs1	1	0						imm2							rs
t_sync_rs2	0	1								rs2					imr
t_sync_rs1_rs2	1	1								rs2					rs
inst_cfg_rs1	1	0													rs
inst_cfg_rd	0	1													
inst_cfg_imm	0	0													
inst_high_asp_vec_load(move)	0	0	1	0	0										
inst_high_asp_vec_store(move)	0	0	1	0	1										
inst_high_asp_scalar_load(move)	0	0	0	0	0										
inst_high_asp_scalar_store(move)	0	0	0	0	1										
inst_high_asp_float_load(move)	0	0	0	1	0										
inst_high_asp_float_store(move)	0	0	0	1	1										
vec_in_store	0	1													
ace_bsync	1	0	0											rs1(syn	
ace_nbsync	1	0	1											rs1(syn	
ace_rd_st	1	1	0											rs1(syn	

Memory Operation

Unit-stride Load/Store

在 unit-stride 模式下，按一个 tile register 维度进行访存，rs1 是访问 base 地址，td 是 tile 寄存器。offseten 为 1 时 rs3 启用，可与 rs1 的值相加得访问地址，tilesize 是表示连续访问几个 tile reg（1、2、4、8），tmask 可按 32B 维度进行选择是否要进行操作。

Global Memory Unit-stride Load

ACE 指令编号							
1	**31**	30:23	22:20	19:15	14:12	**8**	**7**
	rs1_en:1	ts1	tilesize	rs1	tuop:000	**0**	**0**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**
	offseten	memuop:000000	rs3/imm2[4:0]	imm [9:5]	**tuop:110**	tmask	**lsuop:00**

Global Memory Unit-stride Store

ACE 指令编号							
1	**31**	30:23	22:20	19:15	14:12	**8**	**7**
	rs1_en:1	ts1	tilesize	rs1	tuop:000	**rs2_en:0**	**rd_en:0**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**
	offseten	memuop:001000	rs3/imm2[4:0]	imm [9:5]	**tuop:110**	tmask	**lsuop:00**

Share Memory Unit-stride Load

ACE 指令编号							
1	**31**	30:23	22:20	19:15	14:12	**8**	**7**

	rs1_en:1	ts1	tilesize	rs1	tuop:001	**rs2_en:0**	**rd_en:0**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**
	offseten	memuop:000000	rs3/imm2[4:0]	imm[9:5]	**tuop:110**	tmask	**lsuop:00**

Share Memory Unit-stride Store

ACE 指令编号							
1	**31**	30:23	22:20	19:15	14:12	**8**	**7**
	rs1_en:1	ts1	tilesize	rs1	tuop:001	**rs2_en:0**	**rd_en:0**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**
	offseten	memuop:001000	rs3/imm2[4:0]	imm[9:5]	**tuop:110**	tmask	**lsuop:00**

Stride Load/Store

在 stride 模式下，按 32B 进行访存，rs1 是访问 base 地址，td 是 tile 寄存器。offseten 为 1 时 rs3 启用，可与 rs1 的值相加得访问地址，rs2 代表以 32B 为维度的 stride，tilesize 是表示连续访问几个 tile reg（1、2、4、8），tmask 可按 32B 维度进行选择是否要进行操作。

Global Memory Stride Load

ACE 指令编号							
1	**31**	30:23	22:20	19:15	14:12	**8**	**7**
	rs1_en:1	td	tilesize	rs1	tuop:000	**rs2_en:0**	**rd_en:0**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**
	offseten	memuop:000000	rs3/imm2[4:0]	rs2	**tuop:110**	tmask	**lsuop:01**

Global Memory Stride Store

ACE 指令编号							
1	**31**	30:23	22:20	19:15	14:12	**8**	**7**
	rs1_en:1	ts1	tilesize	rs1	tuop:000	**rs2_en:0**	**rd_en:0**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**
	offseten	memuop:001000	rs3/imm2[4:0]	rs2	**tuop:110**	tmask	**lsuop:01**

Share Memory Stride Load

ACE 指令编号							
1	**31**	30:23	22:20	19:15	14:12	**8**	**7**
	rs1_en:1	td	tilesize	rs1	tuop:001	**rs2_en:0**	**rd_en:0**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**
	offseten	memuop:000000	rs3/imm2[4:0]	rs2	**tuop:110**	tmask	**lsuop:01**

Share Memory Stride Store

ACE 指令编号							
1	**31**	30:23	22:20	19:15	14:12	**8**	**7**
	rs1_en:1	ts1	tilesize	rs1	tuop:001	**rs2_en:0**	**rd_en:0**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**
	offseten	memuop:001000	rs3/imm2[4:0]	rs2	**tuop:110**	tmask	**lsuop:01**

Index Vector Load/Store

这类指令中用到 VS2 的 index vector 是通过额外加一条 ASP store 指令写入 tile core 执行的。

🥇 注：ASP_store指令仅为传输向量到tile core，无其他实际作用。

在 index-vector 模式下，按 32B 进行访存，rs1 是访问 base 地址，td 是 tile 寄存器。offseten 为 1 时 rs3 启用，可与 rs1 的值相加得访问地址，vs2 代表以 32B 为维度的 index vector，tilesize 是表示连续访问几个 tile reg（1、2、4、8），tmask 可按 32B 维度进行选择是否要进行操作。

Global Memory Index Vector Load

ACE 指令编号								
2(ASP_store)	31	30	29:28		26	25:20	14:12	11:7
	0	1(input data)	mode:00 (index_vector)		addrCtl	func6	tuop:111	vs2
1	**31**	30:23	22:20		19:15	14:12	**8**	**7**
	rs1_en:1	td	tilesize		rs1	tuop:000	**rs2_en:0**	**rd_en:0**
0	**31**	30:25	24:20		19:15	**14:12**	9	**8:7**
	offseten	memuop: 000000	rs3/imm2[4:0]		vs2	**tuop:110**	tmask	**lsuop:10**

Global Memory Index Vector Store

ACE 指令编号							
2(ASP_store)	31	30	29:28	26	25:20	14:12	11:7
	0	1(input data)	mode:00(index_vector)	addrCtl	func6	tuop:111	vs2
1	**31**	30:23	22:20	19:15	14:12	**8**	**7**
	rs1_en:1	ts1	tilesize	rs1	tuop:000	**rs2_en:0**	**rd_en:0**

0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**
	offseten	memuop:001000	rs3/imm2[4:0]	vs2	**tuop:110**	tmask	**lsuop:10**

Share Memory Index Vector Load

ACE 指令编号							
2(ASP_store)	31	30	29:28	26	25:20	14:12	11:7
	0	1(input data)	mode:00(index_vector)	addrCtl	func6	tuop:111	vs2
1	**31**	30:23	22:20	19:15	14:12	**8**	**7**
	rs1_en:1	td	tilesize	rs1	tuop:001	**rs2_en:0**	**rd_en:0**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**
	offseten	memuop:000000	rs3/imm2[4:0]	vs2	**tuop:110**	tmask	**lsuop:10**

Share Memory Index Vector Store

ACE 指令编号							
2(ASP_store)	31	30	29:28	26	25:20	14:12	11:7
	0	1(input data)	mode:00(index_vector)	addrCtl	func6	tuop:111	vs2
1	**31**	30:23	22:20	19:15	14:12	**8**	**7**
	rs1_en:1	ts1	tilesize	rs1	tuop:001	**rs2_en:0**	**rd_en:0**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**
	offseten	memuop:001000	rs3/imm2[4:0]	vs2	**tuop:110**	tmask	**lsuop:10**

Atomic

Scalar Atomic

在 scalar atomic 模式下，按64bit的数据宽度进行访存，rs1 是访问 base 地址。rd 是目的标量寄存器。rs2是atomic运算的新值。可与 rs1 的值相加得访问地址，rs3代表 CAS 的比较值。atm[2:1]表示x1/x2/x4，使用几个16b/32b/64b数。lsuop[0]为1代表rs2标量atomic。lsuop[1]为1表示是否要将 rs2 的值复制到 1024B 维度做 atomic 运算。cas_en确定是否为cas指令。

Global Memory Scalar Atomic

ACE 指令编号								
1	**31**	27:23	22:21	**20**	19:15	14:12	**8**	**7**
	rs1_en:1	rd	vec_size	**rtn**	rs1	tuop:000	**rs2_en:0**	**rd_en**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**	
	cas_en	memuop: 100xxx-111xxx	rs3/00000	rs2	**tuop:110**	tmask	**lsuop:01**	

ACE 指令编号								
1	**31**	30:23	22:21	**20**	19:15	14:12	**8**	**7**
	rs1_en:1		vec_size	**rtn**	rs1	tuop:000	**rs2_en:0**	**rd_en**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**	
	cas_en	memuop: 100xxx-111xxx	rs3/imm2[4:0]	rs2	**tuop:110**	tmask	**lsuop:11**	

Share Memory Scalar Atomic

在 scalar atomic 模式下，按64bit的数据宽度进行访存，rs1 是访问 base 地址。rd 是目的标量寄存器。rs2是atomic运算的新值。可与 rs1 的值相加得访问地址，rs3代表 CAS 的比较值。atm[2:1]表示x1/x2/x4，使用几个16b/32b/64b数。lsuop[0]为1代表rs2标量atomic。lsuop[1]为1表示是否要将 rs2 的值复制到 1024B 维度做 atomic 运算。cas_en确定是否为cas指令。

ACE 指令编号								
1	**31**	27:23	22:21	**20**	19:15	14:12	**8**	**7**
	rs1_en:1	rd	vec_size	**rtn:0**	rs1	tuop:001	**rs2_en:1**	**rd_en**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**	
	cas_en	memuop: 100xxx-111xxx	rs3/00000	rs2	**tuop:110**	tmask	**lsuop:01**	

ACE 指令编号								
1	**31**	30:23	22:21	**20**	19:15	14:12	**8**	**7**
	rs1_en:1		vec_size	**rtn:0**	rs1	tuop:001	**rs2_en:1**	**rd_en:0**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**	
	cas_en	memuop: 100xxx-111xxx	rs3/imm2[4:0]	rs2	**tuop:110**	tmask	**lsuop:11**	

Tile Atomic

在 Tile atomic 模式下，rs1 是访问 base 地址。td 是目的寄存器。offseten 为 1 时 rs3 启用，可与 rs1 的值相加得访问地址。。tmask 可按 32B 维度进行选择是否要进行操作。

Global Memory Tile Atomic

ACE 指令编号								
1	**31**	30:23	22:21	**20**	19:15	14:12	**8**	**7**
	rs1_en:1	ts2	vec_size	**rtn:0**	rs1	tuop:000	**rs2_en:0**	**rd_en**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**	
	offseten	memuop: 100xxx-111xxx	rs3/imm2[4:0]	imm	**tuop:110**	tmask	**lsuop:00**	

Share Memory Tile Atomic

ACE 指令编号								
1	**31**	30:23	22:21	**20**	19:15	14:12	**8**	**7**
	rs1_en:1	td	vec_size	**rtn:0**	rs1	tuop:001	**rs2_en:0**	**rd_en**
0	**31**	30:25	24:20	19:15	**14:12**	9	**8:7**	
	offseten	memuop: 100xxx-111xxx	rs3/imm2[4:0]	imm [9:5]	**tuop:110**	tmask	**lsuop:00**	

Tensor Async Copy

在 TMA 模式下，dst 和 src 都是标量寄存器输入，表示 tensor map 或地址。srctm 和 dsttm 确定 src 和 dst 是否为 tensormap。dstloc 和 srcloc 确定 dst 和 src 所处的 memory 区域。completion 确定 TMA 完成时的同步机制。dim 确定访存的维度。completeobj 确定 mbar 和 semaphore 的 index。coord/size 是一个坐标和数据量的描述符，用 vector 寄存器作为输入方式。tmask 可按 32B 维度进行选择是否要进行操作。multicastl2 代表在 cluster 内是否广播到 4 个 PE。tmauop 代表 linear 相关配置。

ACE 指令编号												
2(ASP_store)	**31**		30	29:28	26	25:20	**14:12**	11:7				
	0		1(input data)	mode:01(TMA)	addrCtl	func6	**tuop:111**	vs1				
1	**31**	30:29	28:27	26	25	24:20	19:15	14:12	11:10	9	**8**	**7**
	rs1_en:1	tmauop	dim	dsttm	srctm	dst	src	tuop:000	dstloc	srcloc	**rs2_en:1**	**rd_en:0**
0	**31**	30:25	24:20	19:15	14:12	10	9	**8:7**				
	offseten:1	memuop:	completob	coord/siz	tuop:110	multicastl	tmask	**completi**				

		011100	j(gpr)	e(vec reg)		2		**on** **(lsuop:00/10)** 建议bulk和semaphore用这两个encode，这样就能保证completeobj用到mbar时，24:20的GPR输入有效				

Tensor Async Copy Commit(TODO)

指令会在 Tile Pipeline 执行，将之前未完成的 CP指令设置 group_id。

ACE 指令编号										
1	31	30	29:26	24:20	19:15	14:12	11	10	9:8	7
	rs1_en:1	rd_en:0	memuop[5:2]:0100	dst	src	tuop:000	dsttm	srctm	dstloc	srcloc
0	31	30:29	28:27	26:25	24:20	19:15	14:12	10	9	8:7

	offsetesen:1	00		memuop[1:0]:01						

Configuration-setting Instructions

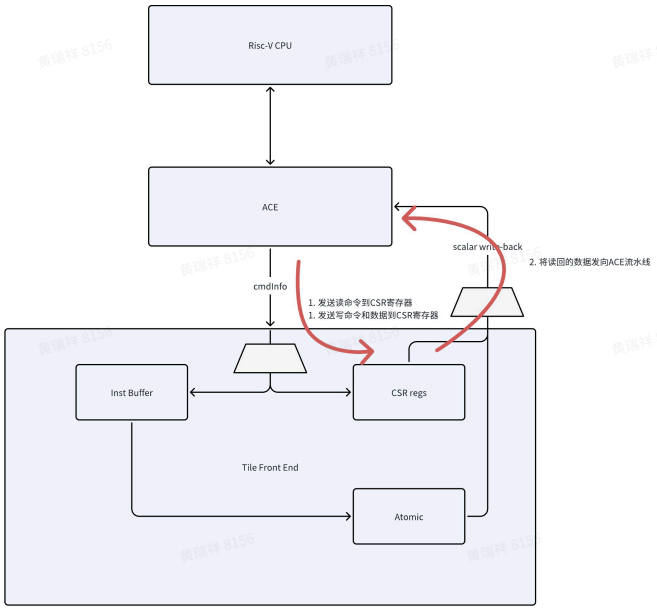
遵循以下的读写规则

Register operand				
Instruction	*rd* is `x0`	*rs1* is `x0`	Reads CSR	Writes CSR
CSRRW	Yes	-	No	Yes
CSRRW	No	-	Yes	Yes
CSRRS/CSRRC	-	Yes	Yes	No
CSRRS/CSRRC	-	No	Yes	Yes
Immediate operand				
Instruction	*rd* is `x0`	*uimm* = 0	Reads CSR	Writes CSR
CSRRWI	Yes	-	No	Yes
CSRRWI	No	-	Yes	Yes
CSRRSI/CSRRCI	-	Yes	Yes	No
CSRRSI/CSRRCI	-	No	Yes	Yes

Table 10. Conditions determining whether a CSR instruction reads or writes the specified CSR.

Table 10 summarizes the behavior of the CSR instructions with respect to whether they read and/or write the CSR.

微架构实现



如图所示，atomic读和CSR读都会写回ACE模块。这里为了减少Tile CSR读写的延迟，将CSR reg的位置放到Tile Front End最靠近ACE的位置。

读CSR：ACE将读命令在cycle 0发射到CSR regs后，cycle 1即可写回ACE，释放ACE读数据的功能单元。该指令的throughput为0.5，两个cycle可发射一个CSR读指令。

写CSR：ACE将写命令和数据在cycle 0发射到CSR regs后，释放ACE读数据的功能单元。该指令的throughput为1，一个cycle可发射一个CSR读指令。

TCSRR

该指令可将 CSR 原有的值读到 rd 指定的寄存器。

ACE 指令编号								
1	**31**	**30**	29:28	27:25	24:20	19:15	14:12	11:7
	rs1_en:0	**rd_en:1**	csr_uop:00	csr_idx[4:0]	csr_idx[9:5]		tuop:010	rd

TCSRW

或者将 rs1 的值写入到该寄存器

ACE 指令编号								
1	**31**	**30**	29:28	27:25	24:20	19:15	14:12	11:7

	rs1_en:1	**rd_en:0**	csr_uop:01	csr_idx[4:0]	csr_idx[9:5]	rs1	tuop:010	

TCSRWI

和 TCSRWI 相同，仅仅是未用到 rs1，用 5b 立即数代替。

ACE 指令编号								
1	**31**	**30**	29:28	27:25	24:20	19:15	14:12	11:7
	rs1_en:0	**rd_en:0**	csr_uop:10	csr_idx[4:0]	csr_idx[9:5]	uimm	tuop:010	

MMA

TMMA sync

同步矩阵乘累加指令

mmauop 代表矩阵乘的运算数据类型。td 代表输出矩阵，ts1/ts2 代表左右矩阵。tilesize 代表多寄存器运算。multiOprd 代表多寄存器运算时，A 矩阵是否是多寄存器。reuse1/reuse2/reused 代表左右和输出矩阵是否在下一条指令复用。transmata/transmatb 代表 A/B 矩阵是否要做 8bit 维度的转置。neg1 和 negd 代表左矩阵和输出矩阵是否要乘负一。

TMMA sync(ts1/ts2)

ACE 指令编号										
1	**31**	30:23	22:15	**14:12**	11	10	9	8	**7**	
	mem mata:0	td	ts1	**tuop:011**	transm ata	neg1	reuse1	reused	**mem matb:0**	
0	**31**	30:23	22:18	17:15	**14:12**	11	10	9	8	7

	0	ts2	mmauop	tilesize	**tuop:110**	transmatb	negd	reuse2	multitile	**0**

MMA async

异步矩阵乘累加指令。

mmauop 代表矩阵乘的运算数据类型。td 代表输出矩阵，ts1/ts2 或 vs1/vs2 代表左右矩阵的寄存器或向量代表的描述符。tilesize 代表多寄存器运算。multiOprd 代表多寄存器运算时，A 矩阵是否是多寄存器。reuse1/reuse2/reused 代表左右和输出矩阵是否要复用。transmata/transmatb 代表 A/B 矩阵是否要做 8bit 维度的转置。neg1 和 negd 代表左矩阵和输出矩阵是否要乘负一。

这类指令中用到 VS2 的 index vector 是通过额外加一条 ASP store 指令写入 tile core 执行的。

> 🥇 注：ASP_store指令仅为传输向量到tile core，无其他实际作用。

MMA async(ts1/vs2)

ACE 指令编号										
1	**31**	30:23	22:15	**14:12**	11	10	9	8	**7**	
	memmata:0	td	ts1	**tuop:011**	transmata	neg1	reuse1	reused	**memmatb:1**	
0	**31**	27:23	22:18	17:15	**14:12**	11	10	9	8	7
	0	rs2	mmauop	tilesize	**tuop:110**	transmatb	negd	reuse2	multitile	**0**

MMA async(ts2/vs1)

ACE 指令编号										
1	**31**	27:23	22:15	**14:12**	11	10	9	8	**7**	
	memmata:	td	rs1	**tuop:011**	transmata	neg1	reuse1	reused	**memmatb:**	

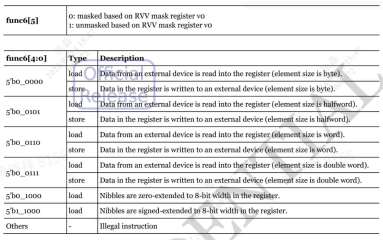
	1								**0**	
0	**31**	30:23	22:18	17:15	**14:12**	11	10	9	8	7
	0	ts2	mmau op	tilesize	**tuop:110**	transm atb	negd	reuse2	multiti le	**0**

MMA async

ACE 指令编号										
1	**31**	30:23	22:15	**14:12**	11	10	9	8	**7**	
	mem mata: 1	td	rs1	**tuop:011**	transm ata	neg1	reuse1	reused	**mem matb: 1**	
0	**31**	27:23	22:18	17:15	**14:12**	11	10	9	8	7
	0	rs2	mmau op	tilesize	**tuop:110**	transm atb	negd	reuse2	multiti le	**0**

Tile Move

输入输出操作使用 ASP 指令操作。



func6[5]	0: masked based on RVV mask register v0 1: unmasked based on RVV mask register v0

func6[4:0]	Type	Description
5'b0_0000	load	Data from an external device is read into the register (element size is byte).
	store	Data in the register is written to an external device (element size is byte).
5'b0_0101	load	Data from an external device is read into the register (element size is halfword).
	store	Data in the register is written to an external device (element size is halfword).
5'b0_0110	load	Data from an external device is read into the register (element size is word).
	store	Data in the register is written to an external device (element size is word).
5'b0_0111	load	Data from an external device is read into the register (element size is double word).
	store	Data in the register is written to an external device (element size is double word).
5'b0_1000	load	Nibbles are zero-extended to 8-bit width in the register.
5'b1_1000	load	Nibbles are signed-extended to 8-bit width in the register.
Others	-	Illegal instruction

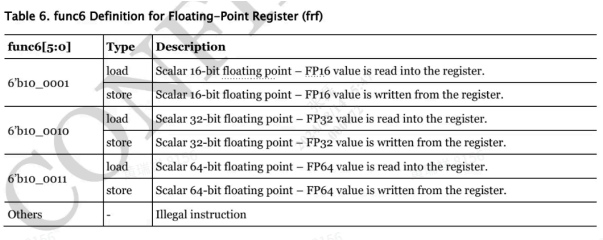


Table 6. func6 Definition for Floating-Point Register (frf)

func6[5:0]	Type	Description
6'b10_0001	load	Scalar 16-bit floating point – FP16 value is read into the register.
	store	Scalar 16-bit floating point – FP16 value is written from the register.
6'b10_0010	load	Scalar 32-bit floating point – FP32 value is read into the register.
	store	Scalar 32-bit floating point – FP32 value is written from the register.
6'b10_0011	load	Scalar 64-bit floating point – FP64 value is read into the register.
	store	Scalar 64-bit floating point – FP64 value is written from the register.
Others	-	Illegal instruction

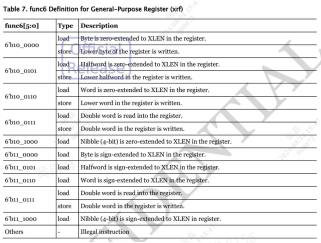


Table 7. func6 Definition for General-Purpose Register (xrf)

func6[5:0]	Type	Description
6'b10_0000	load	Byte is zero-extended to XLEN in the register.
	store	Lower byte of the register is written.
6'b10_0101	load	Halfword is zero-extended to XLEN in the register.
	store	Lower halfword of the register is written.
6'b10_0110	load	Word is zero-extended to XLEN in the register.
	store	Lower word in the register is written.
6'b10_0111	load	Double word is read into the register.
	store	Double word in the register is written.
6'b10_1000	load	Nibble (4-bit) is zero-extended to XLEN in the register.
6'b11_0000	load	Byte is sign-extended to XLEN in the register.
6'b11_0101	load	Halfword is sign-extended to XLEN in the register.
6'b11_0110	load	Word is sign-extended to XLEN in the register.
6'b11_0111	load	Double word is read into the register.
	store	Double word in the register is written.
6'b11_1000	load	Nibble (4-bit) is sign-extended to XLEN in register.
Others	-	Illegal instruction

vs1/vd/rs1/rd 表示 move 的 dst 或者 src，另一个 src 或者 dst 是 td。rs2 代表寄存器中的 index。bc 代表向量或标量广播到整个 Tile Reg。

Tile Move(Vector)

ACE 指令编号								
2(ASP_store)	31	30	29:27	26	25:20	19	14:12	11:7

	0	0(move)	mvop:101	addrCtl:0	func6	bc	tuop:111	vs1
0	**31**	30:23	19:15	14:12	**7**			
	0	td		tuop:110	**0**			

ACE 指令编号								
2(ASP_load)	31	30	29:27	26	25:20	19	14:12	11:7
	0	0(move)	mvop:100	addrCtl	func6	bc	tuop:111	vd
0	**31**	30:23	19:15	14:12	**7**			
	0	ts1		tuop:110	**0**			

Tile Move(integer scalar)

ACE 指令编号								
2(ASP_store)	31	30	29:27	26	25:20	19	14:12	11:7
	0	0(move)	mvop:001	addrCtl	func6	bc	tuop:111	rs1
0	**31**	30:23	19:15	14:12	**7**			
	0	td		tuop:110	**0**			

ACE 指令编号								
2(ASP_load)	31	30	29:27	26	25:20	19	14:12	11:7
	0	0(move)	mvop:000	addrCtl	func6	bc	tuop:111	rd
0	**31**	30:23	19:15	14:12	**7**			
	0	ts1		tuop:110	**0**			

Tile Move(float scalar)

ACE 指令编号								
2(ASP_store)	31	30	29:27	26	25:20	19	14:12	11:7
	0	0(move)	mvop:011	addrCtl	func6	bc	tuop:111	frs1
0	31	30:23	19:15	14:12	7			
	0	td	rs2/imm	tuop:110	**rs2_en**			

ACE 指令编号								
2(ASP_load)	31	30	29:27	26	25:20	19	14:12	11:7
	0	0(move)	mvop:010	addrCtl	func6	bc	tuop:111	frd
0	31	30:23	19:15	14:12	7			
	0	ts1	rs2/imm	tuop:110	**rs2_en**			

TALU

1op ALU, SFU，transpose and Conversion

ts1 表示运算的输入，td 表示运算输出。vecuop2 代表操作类型。neg1 代表在输入运算时要乘负一。tmask 表示要根据 mask 选择执行。reuse1 表示是否复用输入数据。format 代表数据类型。

ACE 指令编号								
1	**31**	30:23	22:15	**14:12**	11:8	**7**		
	0	td	ts1	**tuop:100**	vecuop2[5:2]	**0**		
0	**31**	30:29		**14:12**	11	10:9	**8**	**7**
	0				tmask	vecuop1		**0**

	vecuop2[1:0]		**tuop:110**			**asp_en:0**			

ACE 指令编号									
1	**31**	30:23	19:15	**14:12**	11:8	**7**			
	0	td	rs1	**tuop:011**	vecuop2[5:2]	**1**			
0	**31**	30:29		**14:12**	11	10:9	**8**	**7**	
	0	vecuop2[1:0]		**tuop:110**	tmask	vecuop1	**asp_en:0**	**0**	

ACE 指令编号												
2(ASP_store)	31	30	29:28	27	26	25:20	14:12	11:7				
	0	1(input data)	mode:11(ALU)	vec_en:0	addrCtl	func6	tuop:111	frs1				
1	**31**	30:23	19:15	**14:12**	11:8	**7**						
	0	td	frs1	**tuop:011**	vecuop2[5:2]	**0**						
0	**31**	30:29					**20**	**14:12**	11	10:9	**8**	**7**
	0	vecuop2[1:0]					**src_type:float0**	**tuop:110**	tmask	vecuop1	**asp_en:1**	**0**

ACE 指令编号												

2(ASP_store)	31	30	29:28	27	26	25:20	14:12	11:7				
	0	1(input data)	mode:11(ALU)	vec_en:1	addrCtl	func6	tuop:111	vs1				
1	**31**	30:23	22:15	**14:12**	11:8	**7**						
	0	td	vs1	**tuop:011**	vecuop2[5:2]	**0**						
0	**31**	30:29	28	27	26	25	**20**	**14:12**	11	10:9	**8**	**7**
	0	vecuop2[1:0]	neg2	neg1	reuse2	reuse1	**src_type: vec 1**	**tuop:110**	tmask	vecuop1	**asp_en:1**	**0**

2op ALU, MIN/MAX, Compare

ts1、ts2 表示运算的输入，td 表示运算输出。vecuop2 代表操作类型。neg1、neg2 代表在输入运算时要乘负一。tmask 表示要根据 mask 选择执行。reuse1、reuse2 表示是否复用输入数据。format 代表数据类型。

ACE 指令编号												
1	**31**	30:23	22:15	**14:12**	11:8	**7**						
	0	td	ts1	**tuop:011**	vecuop2[5:2]	**0**						
0	**31**	30:29	28	27	26	25	22:15	**14:12**	11	10:9	**8**	**7**
	0	vecuop2[1:0]	neg2	neg1	reuse2	reuse1	ts2	**tuop:110**	tmask	vecuop1	**asp_en:0**	**0**

ACE指令编号												
1	**31**	30:23	22:15	**14:12**	11:8	**7**						
	0	td	ts1	**tuop:011**	vecuop2[5:2]	**0**						
0	**31**	30:29	28	27	26	25	19:15	**14:12**	11	10:9	**8**	**7**
	0	vecuop2[1:0]	neg2	neg1	reuse2	reuse1	rs2	**tuop:110**	tmask	vecuop1	**asp_en:0**	**1**

ACE指令编号												
2(ASP_store)	31	30	29:28	27	26	25:20	14:12	11:7				
	0	1(input data)	mode:11(ALU)	vec_en:0	addrCtl	func6	tuop:111	frs2				
1	**31**	30:23	22:15	**14:12**	11:8	**7**						
	0	td	ts1	**tuop:011**	vecuop2[5:2]	**0**						
0	**31**	30:29	28	27	26	25	**20**	19:15	**14:12**	11	10:9	**8**	**7**
	0	vecuop2[1:0]	neg2	neg1	reuse2	reuse1	**src_type:float**	frs2	**tuop:110**	tmask	vecuop1	**asp_en:1**	**0**

ACE 指令编号													
2(ASP_store)	31	30	29:28	27	26	25:20	14:12	11:7					
	0	1(input data)	mode:11(ALU)	vec_en:1	addrCtl	func6	tuop:111	vs2					
1	**31**	30:23	22:15	**14:12**	11:8	**7**							
	0	td	ts1	**tuop:011**	vecuop2[5:2]	**0**							
0	**31**	30:29	28	27	26	25	**20**	19:15	**14:12**	11	10:9	**8**	**7**
	0	vecuop2[1:0]	neg2	neg1	reuse2	reuse1	**src_type:vec**	vs2	**tuop:110**	tmask	vecuop1	**asp_en:1**	**0**

3op ALU, MAD and Quantization

ts1、ts2、ts3 表示运算的输入，td 表示运算输出。vecuop2 代表操作类型。neg1、neg2、neg3 代表在输入运算时要乘负一。tmask 表示要根据 mask 选择执行。reuse1、reuse2、reuse3 表示是否复用输入数据。format 代表数据类型。

ACE 指令编号								
1	**31**	30:23	22:15	**14:12**	11:8	**7**		
	0	td	ts1	**tuop:011**	vecuop2[3:0]	**0**		
0	**31**	30:23	22:15	**14:12**	11	10:9	**8**	**7**
	0	ts3	ts2	**tuop:110**	tmask	vecuop1	**asp_en:0**	**0**

ACE 指令编号									
1	**31**	30:23	22:15	**14:12**	11:8	**7**			
	0	td	ts1	**tuop:011**	vecuop2[3:0]	**0**			
0	**31**	30:23		19:15	**14:12**	11	10:9	**8**	**7**
	0	ts3		rs2	**tuop:110**	tmask	vecuop1	**asp_en:0**	**1**

ACE 指令编号									
2(ASP_store)	31	30	29:28	27	26	25:20	14:12	11:7	
	0	1(input data)	mode:11(ALU)	vec_en:1	addrCtl	func6	tuop:111	vs2	
1	**31**	30:23	22:15	**14:12**	11:8	**7**			
	0	td	ts1	**tuop:011**	vecuop2[3:0]	**0**			
0	**31**	30:23	**20**	19:15	**14:12**	11	10:9	**8**	**7**
	0	ts3	**src_type:float0**	vs2	**tuop:110**	tmask	vecuop1	**asp_en:1**	**0**

ACE 指令编号									
2(ASP_store)	31	30	29:28	27	26	25:20	14:12	11:7	
	0	1(input data)	mode:11(ALU)	vec_en:0	addrCtl	func6	tuop:111	frs2	
1	**31**	30:23	22:15	**14:12**	11:8	**7**			
	0	td	ts1	**tuop:011**	vecuop2[3:0]	**0**			
0	**31**	30:23	**20**	19:15	**14:12**	11	10:9	**8**	**7**

	0	ts3	**src_type:vec 1**	frs2	**tuop:110**	tmask	vecuop 1	**asp_en:1**	**0**

Synchronization

	thread内Tile Core指令同步	thread内CPU和Tile Core同步	PE内多threads同步	cluster内同步	chip内同步	多chip间同步	描述
t_wait	☑	☑	☐	☐	☐	☐	各个硬件单元的同步控制，RV Core可根据该信号等待特定模块的执行结束
t_sync	☐	☐	☑	☑	☐	☐	多个thread间的同步控制，不可监控tile core的各个模块的运行状态
DTE group wait	☑	☐	☐	☐	☐	☐	DTE模块按group将DTE指令分组后，该指令一直等待特定的group指令完成后再允许后续指令执行
mbarrier	☑	☐	☑	☑	☑	☐	细粒度的访存和thread的同步，仅能写到L2B
semaphore	☑	☐	☑	☑	☑	☑	TODO：待确认是否和上述功能重叠
atomic	☐	☐	☑	☑	☑	☑	全范围的同步机制

t_sync(reg)

多 thread 间进行同步。nonblk 为 0 代表不进行当前 thread 的阻塞，仅通知其他 thread（arrive）。sync_id 代表同步序号，scope 代表是 CTA 还是 cluster 间同步。imm 代表要同步的 thread 的 mask。

ACE 指令编号									

	31	30	29	28	27:20	19:15	14:12	11:10	9:7
1	rs0_en:1	0		nonblk		sync_id (reg/imm)	tuop:101	scope	ctrluop:000

t_sync(imm)

多 thread 间进行同步。nonblk 为 0 代表不进行当前 thread 的阻塞，仅通知其他 thread（arrive）。sync_id 代表同步序号，scope 代表是 CTA 还是 cluster 间同步。imm 代表要同步的 thread 的 mask。

ACE 指令编号									
	31	30	29	28	27:20	19:15	14:12	11:10	9:7
1	rs0_en:0	0	rs2_en	nonblk	imm	sync_id (reg/imm)	tuop:101	scope	ctrluop:000

t_wait(memory)

st_cnt/ld_cnt 代表 global memory 的 store 和 load 指令执行完成多少数量后才能让 t_wait 指令提交。st_en 和 ld_en 是选择 store/load 的 enable。global/share 用于选择是 global memory 还是 share memory。

该指令会在 ACE Pipeline 执行，在 tgld_cnt/tgst_cnt/tsld_cnt/tsst_cnt 小于等于指令中的st_cnt或者ld_cnt时提交，允许后续指令执行。

ACE 指令编号											
	31	30	29	28:26	25	24:20	19:15	14:12			9:7
1	rs0_en:0	0	rs2_en:0	waitop:000	global/share	st_cnt	ld_cnt	tuop:101			ctrluop:001

t_wait(TMA_wait) (TODO)

指令会在 ACE Pipeline 执行，在指定的 tma group_id 的指令全部完成后，tma_done 信号拉高时提交，允许后续指令执行。

ACE 指令编号								
1	31	30	29	28:26	25	19:15	14:12	9:7
	rs0_en:0	0	rs2_en:0	waitop:001	commit/wait:1	tma_group_id	tuop:101	ctrluop:001

t_wait(MMA)

指令会在 ACE Pipeline 执行，tma_done 信号拉高时提交，允许后续指令执行。tma_done 会在当前 thread 的所有 MMA 指令执行完成后拉高。

ACE 指令编号								
1	31	30	29	28:26	25	19:15	14:12	9:7
	rs0_en:0	0	rs2_en:0	waitop:010			tuop:101	ctrluop:001

t_wait(ALU/SFU)

指令会在 ACE Pipeline 执行，alu_done 信号拉高时提交，允许后续指令执行。alu_done 会在当前 thread 的所有 ALU 指令执行完成后拉高。

ACE 指令编号								
1	31	30	29	28:26	25	19:15	14:12	9:7
	rs0_en:0	0	rs2_en:0	waitop:100			tuop:101	ctrluop:001

t_wait(All)

指令会在 ACE Pipeline 执行，等待之前所有的信号全部拉高才可以提交，允许后续指令执行

ACE 指令编号								
1	31	30	29	28:26	25	19:15	14:12	9:7

	rs0_en:0	0	rs2_en:0	waitop:111			tuop:101	ctrluop:001