# 香山开源处理器
# 用户手册

适用于 昆明湖 V2R2

Kunminghu-V2R2-alpha1
2025 年 1 月 16 日

香山开源处理器社区
XiangShan Open-Source Processor Community

# 目录

# 前言

## 版本信息

| 日期 | 版本 | 说明 |
| --- | --- | --- |
| | | 第一次发布 |

## 声明

# 1 概述

本章是昆明湖 V2R2 的概述。昆明湖 V2R2 是北京开源芯片研究院香山处理器团队（以下简称香山团队）研发的第三代微架构——昆明湖——的 V2R2 版本。

昆明湖的目标是面向服务器和高性能嵌入式场景的通用 CPU，计划通过三次迭代完成这一目标。

- 昆明湖 V1：昆明湖 V1 是昆明湖架构探索阶段，在原有南湖架构上做了大量重构，SPECCPU 2006 得分从 10 分提升到 15 分
- 昆明湖 V2：昆明湖 V2 的目标依照最新 RISC-V 规范完善功能，具体的规范来自 RVA23 profile 和 server SOC spec
- 昆明湖 V3：昆明湖 V3 的目标是优化单 die 32-64 核的多核性能，同时支持多计算 die 的功能

当前版本为昆明湖 V2R2 ，计划通过 2-3 个 Release 版本完成上述目标，昆明湖整体特性见本章 特性 小节，具体规范和指令集支持情况见 § 3 指令集。

为了支持昆明湖的研发和在目标场景落地，香山团队持续开发迭代其它相关组件，包括性能模拟器 xs-gem5，指令集仿真器 NEMU，在线比较框架 difftest 等。本文是昆明湖处理器和 CPU 核相关 IP 的说明，其它组件见相应文档。

## 1.1 简介

如上所述，昆明湖 V2R2 是香山的重要组成部分。与昆明湖 V1 和昆明湖 V2R1 相比，增加了大量符合 RISC-V 规范的指令和 IP；与昆明湖 V2R1 相比，是香山系列 IP 中首个支持 CHI 协议的 IP。

昆明湖 V2R2 系列 IP 包括 CPU Core（含 L2），核内中断控制器（AIA IMSIC），Timer 和 Debug 等模块。

## 1.2 特性

### 1.2.1 处理器核特性

- 支持 RV64 及其扩建指令集
- 支持 RVV 1.0，VLEN 128bit x 2
- 支持 Cacheable 空间的非对齐访问
- 支持内存管理单元（MMU）
- 最大支持 48 位物理地址，支持 39 位和 48 位虚拟地址
- 支持 timer 中断，支持 RVA23-Sstc 特性

### 1.2.2 Cache 和 TLB 特性

- ICache 64KB，支持 Parity

- DCache，最高 64KB，支持 ECC
- Unified L2，最高 1MB，支持 ECC
- L2 作为昆明湖 V2R2 的总线出口，不支持关闭
- 支持一级和二级 TLB

### 1.2.3 总线接口

- 支持 TileLink v1 总线
- 支持 CHI Issue B 和 CHI Issue E.b 的子集，事务和 flit 字段详见总线接口章节；CHI 版本配置方法见总线接口章节

### 1.2.4 中断

- 符合 AIA 1.0 的 CSR
- 符合 AIA 1.0 的 IMSIC (注 1)
- 符合 RISC-V priviledge 的 NMI，提供单独的 NMI 信号，允许用于自行选择连接，详见中断章节

### 1.2.5 Debug 特性

- 支持 DebugModule spec 0.13；
- 支持 HPM；
- 支持 E-trace（注 2）
- 在线正确性比较（Difftest）；
- 支持 CHI 最小单核验证环境；

注 1：AIA aplic 暂不在昆明湖 V2R2 开源清单中，如需要可与我们联系；
注 2：E-trace 核外 trace IP 暂不在昆明湖 V2R2 开源清单中，如需了解可与我们联系。

# 2 处理器简介

## 2.1 结构框图

昆明湖 V2R2 的结构框图如图 2.1 所示。

## 2.2 核内子系统

昆明湖 V2R2 的核内子系统主要包括：取指令单元（IFU）、指令译码单元（IDU）、重命名单元（Rename）、乱序调度单元（Dispatch and Issue）、整数运算单元（IntExu）、浮点运算单元（FPExu）、向量运算单元（VecExu）、存储单元（LSU）、重排序缓存（ROB）、内存管理单元（MMU）、物理内存保护单元（PMP/PMA）和二级高速缓存（L2 Cache）。

### 2.2.1 取指令单元

取指令单元负责从存储中提取指令并对指令进行初步处理，每周期最多可处理 32 字节的指令。为了提高取指令的效率，昆明湖 V2R2 实现了诸如 ICache、FDIP、BPU 等多种结构，具有低功耗、高分支预测准确率和高指令预取准确率的特点。

### 2.2.2 指令译码单元

指令译码单元接收来自取指令单元的指令并进行译码。指令译码单元每周期最多可以处理六条简单指令，并支持指令融合技术。对于诸如多数向量运算指令这样的复杂指令，指令译码单元会将其拆分为数条微指令以供后续单元执行处理。

### 2.2.3 重命名单元

重命名单元将经过译码的指令中的逻辑寄存器重命名为物理寄存器。为了进一步扩宽乱序调度窗口，重命名单元还实现了 ROB 压缩功能。此外，重命名单元也实现了 move 指令消除的功能，通过旁路操作数的方式消除了 move 指令的执行时间。

### 2.2.4 乱序调度单元

乱序调度单元分为分派（Dispatch）单元和发射（Issue）单元两部分。分派单元根据指令的类型，将指令分派到不同的运算发射队列中。发射单元则负责根据指令的就绪情况，将指令发射到执行单元中，并在发射后读取寄存器堆的值。

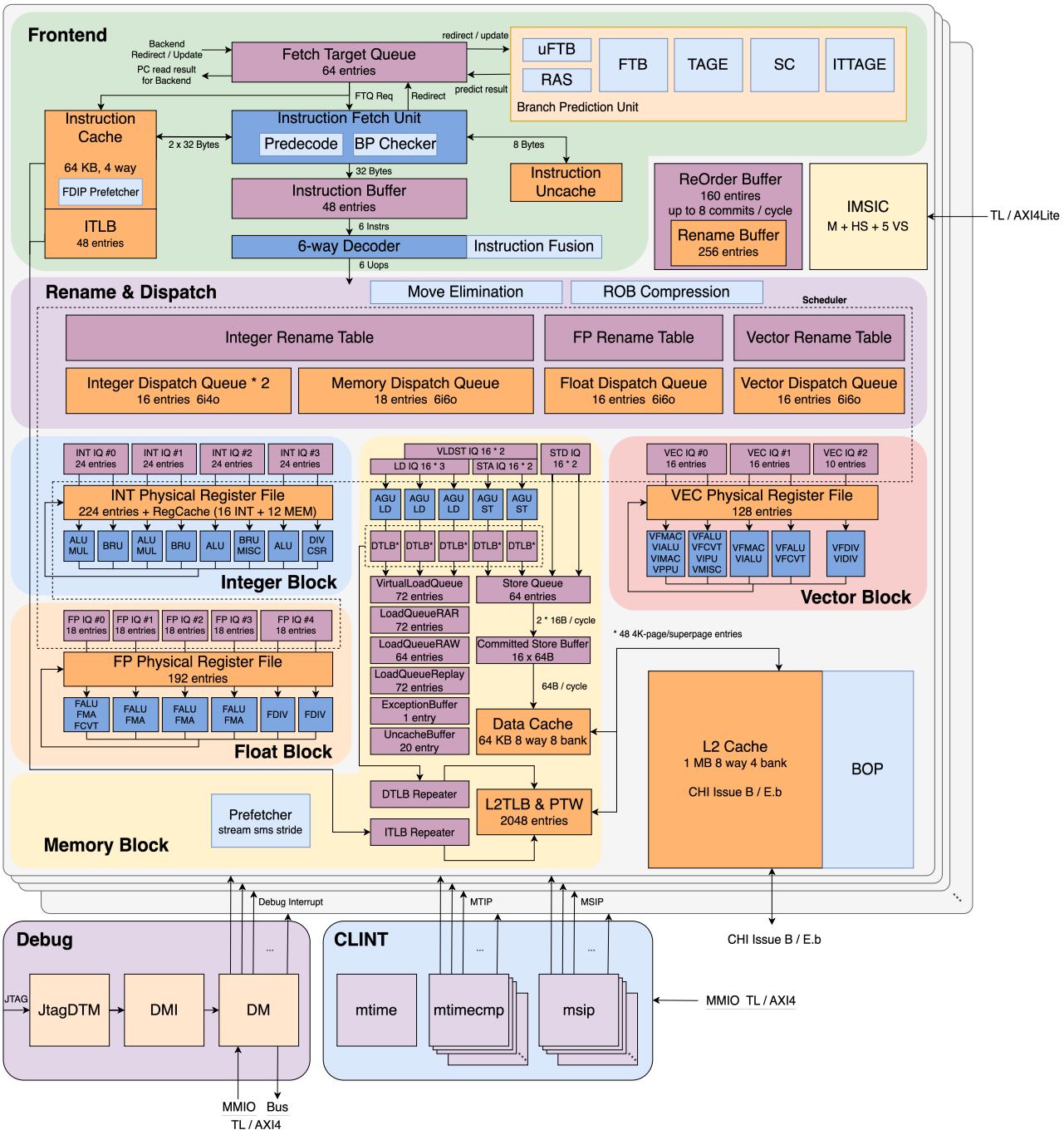


图 2.1: 昆明湖 V2R2 微架构框图

### 2.2.5 运算单元

运算单元包括整数运算单元、浮点运算单元和向量运算单元。

整数运算单元包括算数逻辑单元（ALU）、乘法单元（MUL）、除法单元（DIV）、分支跳转单元（BJU）和控制状态单元（CSR）等。ALU 执行 64 位整数操作。MUL 执行整数乘法运算。DIV 的设计采用了基 16 的 SRT 算法，执行周期数根据操作数的不同而变化。BJU 可以在一个周期内计算出跳转地址，并判断分支预测是否正确。CSR 可以处理控制状态寄存器读写指令，并针对部分读 CSR 指令进行了可流水优化。

浮点运算单元包括浮点算数逻辑单元（FALU）、浮点乘累加单元（FMA）、浮点除法单元（FDIV）和浮点转换单元（FCVT）。FALU 负责加减、比较、符号注入、分类等操作。FMA 负责普通乘法、融合乘累加等操作。FDIV 负责浮点除法等操作。FCVT 负责浮点转换等操作。

向量运算单元总体可分为向量整数运算单元和向量浮点运算单元，其中的整数和浮点单元又被进一步细分为算数逻辑单元、乘累加单元、浮点单元和转换单元等。

### 2.2.6 存储单元

存储单元每周期支持最多三条整数 Load 指令的发射执行、两条整数 Store 指令的发射执行和两条向量访存微指令的发射执行，并支持 DCache 的非阻塞式访问。支持字节、半字、字、双字和四字的存储/载入指令，并支持字节和半字的载入指令的符号位和零扩展。访存指令可以流水执行，使得单条访存流水线中的数据吞吐量可以达到一个周期存取一个数据。支持多种预取技术，可以减少访存时的 DCache 缺失率，提高访存效率。当 DCache 缺失后，支持总线的并行访问。

### 2.2.7 重排序缓存

重排序缓存负责指令的乱序写回和顺序退休。通过支持指令的并行写回与快速退休，提高了指令的退休效率。通过重命名缓存（RAB），实现了指令提交和寄存器提交的解耦，在进一步提高指令退休效率的同时降低了重排序缓存的开销。重排序缓存每周期最多可退休八条指令，支持精确异常。

### 2.2.8 内存管理单元

内存管理单元（MMU）支持 Sv39 和 Sv48，可以将 39 位或 48 位虚拟地址转换为 48 位物理地址。支持 H 拓展及两阶段地址翻译，支持 Sv39x4 和 Sv48x4。也支持 PBMT 扩展。

具体内容详见 § 7 内存模型。

### 2.2.9 物理内存保护单元

物理内存保护单元包括物理内存保护（PMP）和物理内存属性（PMA）。其中，PMP 的实现遵循 RISC-V 手册规定，默认支持 16 项。

PMA 的实现采用了类 PMP 的方式，利用了 PMP Configure 寄存器的两个保留位，设为 atomic 和 cachable，分别为是否支持原子操作和是否可缓存。默认支持 16 项。

PMP 和 PMA 的最小粒度为 4KB，因此不支持 NA4 模式。

具体内容详见 § 7 内存模型。

### 2.2.10 二级高速缓存

二级高速缓存采用分 bank 的流水线架构，每个周期可并行处理 64 个可高速缓存地址空间的访存请求（包括 DCache、ICache、PTW 发送的请求，预取请求，snoop 请求等）。默认配置下，L2 Cache 大小

为 1MB 分为 4 个 bank，采用 8 路组相联的组织方式，并采用了 inclusive 的包含策略。

二级高速缓存对外接口可选 CHI Issue B 和 CHI Issue E.b，并进行了跨时钟域和电压域处理。

接口信息详见 § 11 总线接口。

## 2.3 多核子系统

昆明湖 V2R2 的多核子系统包括中断控制器、计时器和调试系统。

### 2.3.1 中断控制器

中断控制器分为传入消息信号中断控制器（IMSIC）和处理器核局部中断控制器（CLINT）。IMSIC 默认支持 7 个 interrupt file（M + S + 5 VS），并默认支持 254 个有效中断号。CLINT 用于处理软件中断和计时器中断。

具体内容详见 § 10 中断控制器。

### 2.3.2 计时器

计时器复用了 CLINT 中的 mtime 寄存器，计时器将计时器的值广播到各个核内子系统，以支持核内子系统读取 time 寄存器和 Sstc 扩展等功能。

具体内容详见 § 10 中断控制器。

### 2.3.3 调试系统

昆明湖调试系统兼容 RISC-V Debug V0.13 手册标准，对外调试接口支持 JTAG。通过一个共享的 JTAG 接口对不同的核内子系统进行调试。

具体内容详见 § 12 调试。

# 3 指令集

## 3.1 支持的指令集扩展

i, m, a, f, d, c, b, v, h, sdtrig, sha, shcounterenw, shgatpa, shlcofideleg, shtvala, shvsatpa, shvstvala, shvstvecd, smaia, smcsrind, smdbltrp, smmpm, smnpm, smrnmi, smstateen, ss1p13, ssaia, ssccptr, sscofpmf, sscounterenw, sscsrind, ssdbltrp, ssnpm, sspm, ssstateen, ssstrict, sstc, sstvala, sstvecd, ssu64xl, supm, sv39, sv48, svade, svbare, svinval, svnapot, svpbmt, za64rs, zacas, zba, zbb, zbc, zbkb, zbkc, zbkx, zbs, zcb, zcmop, zfa, zfh, zfhmin, zic64b, zicbom, zicbop, zicboz, ziccif, zicclsm, ziccrse, zicntr, zicond, zicsr, zifencei, zihintpause, zihpm, zimop, zkn, zknd, zkne, zknh, zksed, zksh, zkt, zvbb, zvfh, zvfhmin, zvkt, zvl128b, zvl32b, zvl64b

# 4 数据寄存器

## 4.1 通用寄存器

昆明湖 V2R2 具有 32 个 64 位的通用寄存器，其功能可以参照 RISC-V 的手册定义，如下表

表 4.1: 通用寄存器

| 寄存器 | ABI 名称 | 描述 | 保存者 |
|---|---|---|---|
| x0 | zero | 硬件连线 0 | \ |
| x1 | ra | 返回地址 | 调用者 |
| x2 | sp | 堆栈指针 | 被调用者 |
| x3 | gp | 全局指针 | \ |
| x4 | tp | 线程指针 | \ |
| x5 | t0 | 临时/备用链接寄存器 | 调用者 |
| x6-7 | t1-2 | 临时寄存器 | 调用者 |
| x8 | s0/fp | 保留寄存器/帧指针 | 被调用者 |
| x9 | s1 | 保留寄存器 | 被调用者 |
| x10-11 | a0-1 | 函数参数/返回值 | 调用者 |
| x12-17 | a2-7 | 函数参数 | 调用者 |
| x18-27 | s2-11 | 保留寄存器 | 被调用者 |
| x28-31 | t3-6 | 临时寄存器 | 调用者 |

## 4.2 浮点寄存器

昆明湖 V2R2 支持 RV64 的 F 扩展与 D 扩展，具有 32 个 64 位的浮点寄存器，其功能可以参照 RISC-V 的手册定义，如下表

表 4.2: 浮点寄存器

| 寄存器 | ABI 名称 | 描述 | 保存者 |
|---|---|---|---|
| f0-7 | ft0-7 | 浮点临时寄存器 | 调用者 |
| f8-9 | fs0-1 | 浮点保留寄存器 | 被调用者 |
| f10-11 | fa0-1 | 浮点参数/返回值 | 调用者 |
| f12-17 | fa2-7 | 浮点参数 | 调用者 |
| f18-27 | fs2-11 | 浮点保留寄存器 | 被调用者 |

| 寄存器 | ABI 名称 | 描述 | 保存者 |
|---|---|---|---|
| f28-31 | ft8-11 | 浮点临时寄存器 | 调用者 |

昆明湖 V2R2 同时支持单精度浮点运算与双精度浮点运算，在进行单精度浮点运算时，只使用浮点寄存器的低 32 位。昆明湖 V2R2 还支持 zfa 扩展中的半精度运算，它们仅仅使用浮点寄存器的低 16 位。

### 4.2.1 浮点寄存器与通用寄存器间传输数据

通用寄存器与浮点寄存器之间可以进行数据传输，精度为单精度，该功能通过传输指令实现。
通用寄存器到浮点寄存器：

1. FMV.W.X
2. FCVT.S.W
3. FCVT.S.WU
4. FCVT.S.L
5. FCVT.S.LU

浮点寄存器到通用寄存器：

1. FMV.X.W
2. FCVT.W.S
3. FCVT.WU.S
4. FCVT.L.S
5. FCVT.LU.S

## 4.3 向量寄存器

昆明湖 V2R2 支持 RV64 的 V 扩展，具有 32 个 128 位的向量架构寄存器。

### 4.3.1 向量寄存器与通用寄存器间传输数据

向量寄存器与通用寄存器间可以进行数据传输，该功能由 VMV 指令实现，如下所示：

1. VMV.V.X 将通用寄存器的值传输给向量寄存器
2. VMV.S.X 将通用寄存器的值传输给向量寄存器首元素
3. VMV.X.S 将向量寄存器首元素传输给通用寄存器

### 4.3.2 向量寄存器与浮点寄存器间传输数据

向量寄存器与浮点寄存器间可以进行数据传输，该功能由 VFMV 指令实现，如下所示：

1. VFMV.V.F 将浮点寄存器的值传输给向量寄存器
2. VFMV.S.F 将浮点寄存器的值传输给向量寄存器首元素
3. VFMV.F.S 将向量寄存器首元素传输给浮点寄存器

# 5 特权模式与控制状态寄存器

## 5.1 处理器模式

昆明湖 V2R2 支持 RISC-V 特权架构手册规定的以下 6 种特权模式。

表 5.1: 昆明湖 V2R2 支持的特权模式列表

| 名称 | 缩写 | PRV | V |
|---|---|---|---|
| 机器模式（Machine mode） | M | 3 | 0 |
| 监管模式（Supervisor mode） | HS/S | 1 | 0 |
| 用户模式（User mode） | U | 0 | 0 |
| 虚拟监管模式（Virtual supervisor mode） | VS | 1 | 1 |
| 虚拟用户模式（Virtual user mode） | VU | 0 | 1 |
| 调试模式（Debug mode） | D | | |

昆明湖 V2R2 初始化时处在 M 模式。对于一般场景，各模式权限高低为 M > S > U；对于虚拟化场景，各模式权限高低为 M > HS > VS > VU。

### 5.1.1 机器模式

机器模式（Machine mode，M 模式）由机器级 ISA 规定，具有最高的权限。M 模式通常用于机器固件，具有以下特性：

- M 模式下可以访问全部的 M、S、H、VS、U CSR，但不可访问部分调试模式 CSR。
- M 模式通常以物理地址取指、访存，不进行虚拟地址翻译，但以下情况除外：
  - `mstatus` .MPRV = 1 时，加载（Load）和存储（Store）操作按 MPP 字段指定的模式进行虚拟地址翻译。
  - 使用 HLV、HLVX、HSV 等虚拟机加载存储指令时，按照 `hstatus` 字段的 SPVP 字段指定的虚拟模式（VS 或 VU）进行两阶段地址翻译。
- M 模式取指、访存通常不进行 PMP 检查，默认情况下具有权限，但以下情况除外：
  - 加载、存储操作按照上述条件以其他特权模式执行时，按照其特权模式进行 PMP 检查。
  - 某个 PMP 项被锁定时，取指、访存操作均需检查此 PMP 项权限。

### 5.1.2 监管模式

监管模式（Supervisor mode，S 模式）由监管级 ISA 规定，虚拟化扩展将之扩充为为虚拟机管理扩展的监管模式（Hypervisor-extended supervisor mode，HS 模式）。S/HS 模式通常用于操作系统和虚拟机

管理程序，其具有以下特性：

- S/HS 模式下可访问 S、H、VS、U CSR，不可访问 M CSR 和调试模式 CSR。
- S 模式的取指、访存通常需要根据 satp 寄存器进行虚拟地址翻译，但以下情况除外：
  - 使用 HLV、HLVX、HSV 等虚拟机加载存储指令时，按照 `hstatus` .SPVP 字段指定的虚拟模式（VS 或 VU）进行两阶段地址翻译。
- S 模式总是需要进行 PMP 检查。
- S 模式不可执行 M 模式特权指令。

### 5.1.3 用户模式

用户模式（User mode，U 模式）具有以下特性：

- U 模式仅可访问非特权 CSR，主要包括浮点、向量和非特权计数器 CSR。
- U 模式的取指、访存通常需要根据 satp 寄存器进行虚拟地址翻译，但以下情况除外：
  - 当 `hstatus` .HU = 1 时，使用 HLV、HLVX、HSV 等虚拟机加载存储指令时，按照 `hstatus` .SPVP 字段指定的虚拟模式（VS 或 VU）进行两阶段地址翻译。
- U 模式总是需要进行 PMP 检查。
- U 模式通常不可执行特权指令，部分情况下存在例外。

### 5.1.4 虚拟监管模式

虚拟监管模式（Virtual supervisor mode，VS 模式）由虚拟化扩展引入，具有以下特性：

- VS 模式下可访问 S、U CSR，但不可访问 M、H、VS CSR。
- VS 模式下访问特定的 S 模式寄存器，会被重定向到对应的 VS 寄存器。
- VS 模式下的取指、访存通常需要根据 hgatp、vsatp 寄存器进行两阶段地址翻译。
- VS 模式总是需要进行 PMP 检查。
- VS 模式不可执行 M 模式特权指令，亦不可执行 H 特权指令。

### 5.1.5 虚拟用户模式

虚拟用户模式（Virtual user mode，VU 模式）由虚拟化扩展引入，具有以下特性：

- VU 模式仅可访问非特权 CSR，主要包括浮点、向量和非特权计数器 CSR。
- VU 模式下的取指、访存通常需要根据 hgatp、vsatp 寄存器进行两阶段地址翻译。
- VU 模式总是需要进行 PMP 检查。
- VU 模式通常不可执行特权指令。

### 5.1.6 调试模式

调试模式（Debug mode，Debug 模式）由调试扩展（Debug 扩展）引入，其特性和细节请参考 § 12 调试。

## 5.2 控制和状态寄存器（Control and Status Registers）

我们将以权限的不同对控制和状态寄存器（CSRs）进行分组介绍。

### 5.2.1 用户模式可读写的 CSRs

昆明湖 V2R2 中实现的权限为用户模式可读写（URW）的 RISC-V 的 CSRs 如下表所示：

表 5.2: 昆明湖 V2R2 支持的 URW 的 CSRs 列表

| 名称 | 特权级 | 编号 | 描述 | 组别 |
|---|---|---|---|---|
| fflags | U | 0x001 | 浮点异常累积状态寄存器 | 非特权浮点 |
| frm | U | 0x002 | 浮点动态舍入模式寄存器 | 非特权浮点 |
| fcsr | U | 0x003 | 浮点控制和状态寄存器 | 非特权浮点 |
| vstart | U | 0x008 | 向量起始位置寄存器 | 非特权向量 |
| vxsat | U | 0x009 | 定点溢出标志位寄存器 | 非特权向量 |
| vxrm | U | 0x00A | 定点舍入模式寄存器 | 非特权向量 |
| vcsr | U | 0x00F | 向量控制和状态寄存器 | 非特权向量 |
| vl | U | 0xC20 | 向量长度寄存器 | 非特权向量 |
| vtype | U | 0xC21 | 向量数据类型寄存器 | 非特权向量 |
| vlenb | U | 0xC22 | 向量寄存器字节数寄存器 | 非特权向量 |

### 5.2.2 用户模式只读的 CSRs

昆明湖 V2R2 中实现的权限为用户模式只读（URO）的 RISC-V 的 CSRs 如下表所示：

表 5.3: 昆明湖 V2R2 支持的 URO 的 CSRs 列表

| 名称 | 特权级 | 编号 | 描述 | 组别 |
|---|---|---|---|---|
| cycle | U | 0xC00 | 用户模式周期计数器 | 用户模式计数器 |
| time | U | 0xC01 | 用户模式时间计数器 | 用户模式计数器 |
| instret | U | 0xC02 | 用户模式退休指令计数器 | 用户模式计数器 |
| hpmcounter3 | U | 0xC03 | 用户模式计数器 3 | 用户模式计数器 |
| ... | ... | ... | ... | ... |
| hpmcounter31 | U | 0xC1F | 用户模式计数器 31 | 用户模式计数器 |

### 5.2.3 监管模式可读写的 CSRs

昆明湖 V2R2 中实现的权限为监管模式可读写（SRW）的 RISC-V 的 CSRs 如下表所示：

表 5.4: 昆明湖 V2R2 支持的 SRW 的 CSRs 列表

| 名称 | 特权级 | 编号 | 描述 | 组别 |
|---|---|---|---|---|
| sstatus | S | 0x100 | 监管模式处理器状态寄存器 | 监管陷入设置 |
| sie | S | 0x104 | 监管模式中断使能控制寄存器 | 监管陷入设置 |

| 名称 | 特权级 | 编号 | 描述 | 组别 |
|---|---|---|---|---|
| stvec | S | 0x105 | 监管模式陷入向量基址寄存器 | 监管陷入设置 |
| scounteren | S | 0x106 | 监管模式计数器使能控制寄存器 | 监管陷入设置 |
| senvcfg | S | 0x10A | 监管模式环境配置寄存器 | 监管环境配置 |
| sscratch | S | 0x140 | 监管模式陷入临时数据备份寄存器 | 监管陷入处理 |
| sepc | S | 0x141 | 监管模式陷入保留程序计数器 | 监管陷入处理 |
| scause | S | 0x142 | 监管模式陷入事件原因寄存器 | 监管陷入处理 |
| stval | S | 0x143 | 监管模式陷入事件向量寄存器 | 监管陷入处理 |
| sip | S | 0x144 | 监管模式陷入事件等待状态寄存器 | 监管陷入处理 |
| stimecmp | S | 0x14D | 监管模式计时器中断比较值寄存器 | 监管陷入设置 |
| siselect | S | 0x150 | 监管模式间接寄存器选择信号寄存器 | 监管间接访问寄存器 |
| sireg | S | 0x151 | 监管模式间接寄存器别名寄存器 | 监管间接访问寄存器 |
| stopei | S | 0x15C | 监管模式顶部外部中断寄存器 | 监管中断 |
| satp | S | 0x180 | 监管模式虚拟地址转换和保护寄存器 | 监管保护和转换 |
| scontext | S/Debug | 0x5A8 | 监管模式上下文寄存器 | 调试寄存器 |
| sbpctl | S | 0x5C0 | 推测状态分支预测控制寄存器 | 推测状态分支预测控制 |
| spfctl | S | 0x5C1 | 推测状态预取控制寄存器 | 推测状态分支预测控制 |
| slvpredctl | S | 0x5C2 | 推测状态 LOAD 违例预测控制寄存器 | 推测状态分支预测控制 |
| smblockctl | S | 0x5C3 | 推测状态内存阻塞控制寄存器 | 推测状态分支预测控制 |
| srnctl | S | 0x5C4 | 推测状态运行时控制寄存器 | 推测状态运行时控制 |

### 5.2.4 监管模式只读的 CSRs

昆明湖 V2R2 中实现的权限为监管模式只读（SRO）的 RISC-V 的 CSRs 如下表所示：

表 5.5: 昆明湖 V2R2 支持的 SRO 的 CSRs 列表

| 名称 | 特权级 | 编号 | 描述 | 组别 |
|---|---|---|---|---|
| stopi | S | 0xDB0 | 监管模式顶层中断 | 监管中断 |

### 5.2.5 虚拟监管模式可读写的 CSRs

昆明湖 V2R2 中实现的权限为虚拟监管模式可读写（HRW）的 RISC-V 的 CSRs 如下表所示：

表 5.6: 昆明湖 V2R2 支持的 HRW 的 CSRs 列表

| 名称 | 特权级 | 编号 | 描述 | 组别 |
|---|---|---|---|---|
| vsstatus | VS | 0x200 | 虚拟监管模式处理器状态寄存器 | 虚拟监管 |
| vsie | VS | 0x204 | 虚拟监管模式中断使能控制寄存器 | 虚拟监管 |
| vstvec | VS | 0x205 | 虚拟监管模式陷入向量基址寄存器 | 虚拟监管 |
| vsscratch | VS | 0x240 | 虚拟监管模式陷入临时数据备份寄存器 | 虚拟监管 |
| vsepc | VS | 0x241 | 虚拟监管模式陷入保留程序计数器 | 虚拟监管 |
| vscause | VS | 0x242 | 虚拟监管模式陷入事件原因寄存器 | 虚拟监管 |

| 名称 | 特权级 | 编号 | 描述 | 组别 |
|---|---|---|---|---|
| vstval | VS | 0x243 | 虚拟监管模式陷入事件向量寄存器 | 虚拟监管 |
| vsip | VS | 0x244 | 虚拟监管模式陷入事件等待状态寄存器 | 虚拟监管 |
| vstimecmp | VS | 0x24D | 虚拟监管模式计时器中断比较值寄存器 | 虚拟监管 |
| vsiselect | VS | 0x250 | 虚拟监管模式间接寄存器选择信号寄存器 | 虚拟监管间接访问寄存器 |
| vsireg | VS | 0x251 | 虚拟监管模式间接寄存器别名寄存器 | 虚拟监管间接访问寄存器 |
| vstopei | VS | 0x25C | 虚拟监管模式顶部外部中断寄存器 | 虚拟监管中断 |
| vsatp | VS | 0x280 | 虚拟监管模式虚拟地址转换和保护寄存器 | 虚拟监管 |
| hstatus | HS | 0x600 | 虚拟机模式处理器状态寄存器 | 虚拟机陷入设置 |
| hedeleg | HS | 0x602 | 虚拟机模式陷入降级控制寄存器 | 虚拟机陷入设置 |
| hideleg | HS | 0x603 | 虚拟机模式中断降级控制寄存器 | 虚拟机陷入设置 |
| hie | HS | 0x604 | 虚拟机模式中断使能寄存器 | 虚拟机陷入设置 |
| htimedelta | HS | 0x605 | 虚拟机模式虚拟化计时器 | 虚拟机计时器 |
| hcounteren | HS | 0x606 | 虚拟机模式计数器使能寄存器 | 虚拟机陷入设置 |
| hgeie | HS | 0x607 | 虚拟机模式客户机外部中断启用寄存器 | 虚拟机陷入设置 |
| hvien | HS | 0x608 | 虚拟机模式虚拟中断使能寄存器 | 虚拟机陷入设置 |
| hvictl | HS | 0x609 | 虚拟机模式虚拟中断控制寄存器 | 虚拟机陷入设置 |
| henvcfg | HS | 0x60A | 虚拟机模式环境配置寄存器 | 虚拟机配置 |
| htval | HS | 0x643 | 虚拟机模式陷入事件向量寄存器 | 虚拟机陷入处理 |
| hip | HS | 0x644 | 虚拟机模式陷入事件等待状态寄存器 | 虚拟机陷入处理 |
| hvip | HS | 0x645 | 虚拟机模式虚拟中断挂起寄存器 | 虚拟机陷入处理 |
| hviprio1 | HS | 0x646 | 虚拟机模式 VS-Level 中断优先级寄存器 1 | 虚拟机陷入处理 |
| hviprio2 | HS | 0x647 | 虚拟机模式 VS-Level 中断优先级寄存器 2 | 虚拟机陷入处理 |
| htinst | HS | 0x64A | 虚拟机模式陷入指令寄存器 | 虚拟机陷入处理 |
| hgatp | HS | 0x680 | 虚拟机模式客户地址转换和保护寄存器 | 虚拟机保护和转换 |
| hcontext | HS/Debug | 0x6A8 | 虚拟机模式上下文寄存器 | 调试寄存器 |

### 5.2.6 虚拟监管模式只读的 CSRs

昆明湖 V2R2 中实现的权限为虚拟监管模式只读（HRO）的 RISC-V 的 CSRs 如下表所示：

表 5.7: 昆明湖 V2R2 支持的 HRO 的 CSRs 列表

| 名称 | 特权级 | 编号 | 描述 | 组别 |
|---|---|---|---|---|
| hgeip | HS | 0xE12 | 虚拟机模式客户机外部中断挂起寄存器 | 虚拟机陷入处理 |
| vstopi | VS | 0xEB0 | 虚拟监管模式顶层中断 | 虚拟监管级中断 |

### 5.2.7 机器模式可读写的 CSRs

昆明湖 V2R2 中实现的权限为机器模式可读写（MRW）的 RISC-V 的 CSRs 如下表所示：

表 5.8: 昆明湖 V2R2 支持的 MRW 的 CSRs 列表

| 名称 | 特权级 | 编号 | 描述 | 组别 |
|---|---|---|---|---|
| mstatus | M | 0x300 | 机器模式处理器状态寄存器 | 机器陷入设置 |
| misa | M | 0x301 | 机器模式处理器指令集寄存器 | 机器陷入设置 |
| medeleg | M | 0x302 | 机器模式陷入降级控制寄存器 | 机器陷入设置 |
| mideleg | M | 0x303 | 机器模式中断降级控制寄存器 | 机器陷入设置 |
| mie | M | 0x304 | 机器模式中断使能寄存器 | 机器陷入设置 |
| mtvec | M | 0x305 | 机器模式陷入向量基址寄存器 | 机器陷入设置 |
| mcounteren | M | 0x306 | 机器模式计数器使能寄存器 | 机器陷入设置 |
| mvien | M | 0x308 | 机器模式虚拟中断使能寄存器 | 机器陷入设置 |
| mvip | M | 0x309 | 机器模式虚拟中断挂起寄存器 | 机器陷入设置 |
| menvcfg | M | 0x30A | 机器模式环境配置寄存器 | 机器配置 |
| mstateen0 | M | 0x30C | 机器模式状态使能寄存器 | 机器状态使能扩展 |
| mcountinhibit | M | 0x320 | 机器模式计数禁止寄存器 | 机器计数器配置 |
| mhpmevent3 | M | 0x323 | 机器模式性能监测事件选择寄存器 3 | 机器计数器配置 |
| ... | ... | ... | ... | ... |
| mhpmevent31 | M | 0x33F | 机器模式性能监测事件选择寄存器 31 | 机器计数器配置 |
| mscratch | M | 0x340 | 机器模式陷入临时数据备份寄存器 | 机器陷入处理 |
| mepc | M | 0x341 | 机器模式陷入保留程序计数器 | 机器陷入处理 |
| mcause | M | 0x342 | 机器模式陷入事件原因寄存器 | 机器陷入处理 |
| mtval | M | 0x343 | 机器模式陷入事件向量寄存器 | 机器陷入处理 |
| mip | M | 0x344 | 机器模式陷入事件等待状态寄存器 | 机器陷入处理 |
| mtinst | M | 0x34A | 机器模式陷入指令寄存器 | 机器陷入处理 |
| mtval2 | M | 0x34B | 机器模式陷入事件向量寄存器 2 | 机器陷入处理 |
| miselect | M | 0x350 | 机器模式间接寄存器选择信号寄存器 | 机器间接访问寄存器 |
| mireg | M | 0x351 | 机器模式间接寄存器别名寄存器 | 机器间接访问寄存器 |
| mtopei | M | 0x35C | 机器模式顶部外部中断寄存器 | 机器中断 |
| pmpcfg0 | M | 0x3A0 | 物理内存保护配置寄存器 0 | 机器内存保护 |
| pmpcfg2 | M | 0x3A2 | 物理内存保护配置寄存器 2 | 机器内存保护 |
| ... | ... | ... | ... | ... |
| pmpcfg14 | M | 0x3AE | 物理内存保护配置寄存器 14 | 机器内存保护 |
| pmpaddr0 | M | 0x3B0 | 物理内存保护基址寄存器 0 | 机器内存保护 |
| ... | ... | ... | ... | ... |
| pmpaddr63 | M | 0x3EF | 物理内存保护基址寄存器 63 | 机器内存保护 |
| mnscratch | M | 0x740 | 机器模式不可屏蔽中断临时数据备份寄存器 | 机器不可屏蔽中断处理 |
| mnepc | M | 0x741 | 机器模式不可屏蔽中断保留程序计数器 | 机器不可屏蔽中断处理 |
| mncause | M | 0x742 | 机器模式不可屏蔽中断事件原因寄存器 | 机器不可屏蔽中断处理 |

| 名称 | 特权级 | 编号 | 描述 | 组别 |
|---|---|---|---|---|
| mnstatus | M | 0x744 | 机器模式不可屏蔽处理器状态寄存器 | 机器不可屏蔽中断处理 |
| mseccfg | M | 0x747 | 机器模式安全配置寄存器 | 机器配置 |
| tselect | M/Debug | 0x7A0 | Debug/Trace 触发器选择寄存器 | 调试寄存器 |
| tdata1 | M/Debug | 0x7A1 | 第一个 Debug/Trace 触发器数据寄存器 | 调试寄存器 |
| tdata2 | M/Debug | 0x7A2 | 第二个 Debug/Trace 触发器数据寄存器 | 调试寄存器 |
| tdata3 | M/Debug | 0x7A3 | 第三个 Debug/Trace 触发器数据寄存器 | 调试寄存器 |
| tinfo | M/Debug | 0x7A4 | Debug/Trace 触发器信息寄存器 | 调试寄存器 |
| tcontrol | M/Debug | 0x7A2 | 机器模式触发器使能寄存器 | 调试寄存器 |
| mcontext | M/Debug | 0x7A3 | 机器模式上下文寄存器 | 调试寄存器 |
| pmacfg0 | M | 0x7C0 | PMA 配置寄存器 0 | PMA 配置 |
| pmacfg2 | M | 0x7C2 | PMA 配置寄存器 2 | PMA 配置 |
| pmaaddr0 | M | 0x7C8 | PMA 地址寄存器 0 | PMA 地址 |
| … | … | … | … | … |
| pmaaddr15 | M | 0x7D7 | PMA 地址寄存器 15 | PMA 地址 |
| mcycle | M | 0xB00 | 机器模式周期计数器 | 机器计数器 |
| minstret | M | 0xB02 | 机器模式退役指令计数器 | 机器计数器 |
| mhpmcounter3 | M | 0xB03 | 机器模式性能监测计数器 3 | 机器计数器 |
| … | … | … | … | … |
| mhpmcounter31 | M | 0xB1F | 机器模式性能监测计数器 31 | 机器计数器 |

### 5.2.8 机器模式只读的 CSRs

昆明湖 V2R2 中实现的权限为机器模式只读（MRO）的 RISC-V 的 CSRs 如下表所示：

表 5.9: 昆明湖 V2R2 支持的 MRO 的 CSRs 列表

| 名称 | 特权级 | 编号 | 描述 | 组别 |
|---|---|---|---|---|
| mvendorid | M | 0xF11 | 供应商编号寄存器 | 机器信息 |
| marchid | M | 0xF12 | 架构编号寄存器 | 机器信息 |
| mimpid | M | 0xF13 | 机器模式硬件实现编号寄存器 | 机器信息 |
| mhartid | M | 0xF14 | 机器模式逻辑内核编号寄存器 | 机器信息 |
| mconfigptr | M | 0xF15 | 配置数据结构指针 | 机器信息 |
| mtopi | M | 0xFB0 | 机器模式顶层中断 | 机器陷入设置 |

### 5.2.9 调试模式可读写的 CSRs

昆明湖 V2R2 中实现的权限为调试模式可读写（DRW）的 RISC-V 的 CSRs 如下表所示：

表 5.10: 昆明湖 V2R2 支持的 DRW 的 CSRs 列表

| 名称 | 特权级 | 编号 | 描述 | 组别 |
|---|---|---|---|---|
| dcsr | Debug | 0x7B0 | 调试模式控制与状态寄存器 | 调试模式寄存器 |
| dpc | Debug | 0x7B1 | 调试模式程序计数器 | 调试模式寄存器 |
| dscratch0 | Debug | 0x7B2 | 调试模式暂存寄存器 0 | 调试模式寄存器 |
| dscratch1 | Debug | 0x7B3 | 调试模式暂存寄存器 1 | 调试模式寄存器 |

## 5.3 自定义 CSRs（Custom CSRs）

在上述实现的控制状态寄存器表中可以发现，昆明湖 V2R2 扩展了 RISC-V 的 CSRs，实现了一些 RISC-V 手册中未定义的 CSR，下面将对这些 CSR 进行介绍。

### 5.3.1 sbpctl

sbpctl 的地址为 0x5C0，其初始化值为下表的默认值，其每一 bit 的功能如下表所示

表 5.11: sbpctl 的 bit 功能

| 位 | 功能 | 默认值 |
|---|---|---|
| 0 | UBTB_ENABLE 设 1 代表开启 uftb | 1 |
| 1 | BTB_ENABLE 设 1 代表开启主 ftb | 1 |
| 2 | BIM_ENABLE 设 1 代表开启 bim 预测器[1] | 1 |
| 3 | TAGE_ENABLE 设 1 代表开启 TAGE 预测器 | 1 |
| 4 | SC_ENABLE 设 1 代表开启 SC 预测器 | 1 |
| 5 | RAS_ENABLE 设 1 代表开启 RAS 预测器 | 1 |
| 6 | LOOP_ENABLE 设 1 代表开启 loop 预测器[2] | 1 |
| 7-31 | WARL | 0 |

### 5.3.2 spfctl

spfctl 的地址为 0x5C1，其初始化值为下表的默认值，其每一 bit 的功能如下表所示:

表 5.12: spfctl 的 bit 功能

| 位 | 功能 | 默认值 |
|---|---|---|
| 0 | 控制 L1 指令预取器，设 1 代表开启预取 | 1 |
| 1 | 控制 L2 预取器，设 1 代表开启预取 | 1 |
| 2 | 控制 SMS 预取器，设 1 代表开启预取 | 1 |
| 3 | 控制 SMS 预取器是否在 hit 时接受训练：设 1 代表 hit 也会接受训练，设 0 代表只有 miss 才会训练 | 0 |

[1]当前 bim 已被移除，该位无实际功能。

[2]当前 Loop 预测器尚未合入主线，该位无实际功能

| 位 | 功能 | 默认值 |
|---|---|---|
| 4 | 控制 SMS 预取器的 agt 表，设 1 代表开启 agt 表 | 1 |
| 5 | 控制 SMS 预取器的 pht 表，设 1 代表开启 pht 表 | 1 |
| 6-9 | 控制 SMS 预取器的 active page 阈值 | 12 |
| 10-15 | 控制 SMS 预取器的 active page 跨度 | 30 |
| 16 | 无功能 | 1 |
| 17 | 控制 L2 预取器是否只对 store 预取，设 1 代表是 | 0 |
| 高位 | 目前高位无实际功能 | 0 |

### 5.3.3 slvpredctl

slvpredctl 的地址为 0x5C2，其初始化值为下表的默认值，其每一 bit 的功能如下表所示

表 5.13: slvpredctl 的 bit 功能

| 位 | 功能 | 默认值 |
|---|---|---|
| 0 | 控制访存违例预测器是否禁用，设 1 代表禁用 | 0 |
| 1 | 控制访存违例预测器是否禁止 load 指令推测执行，设 1 代表禁止 | 0 |
| 2 | 控制访存违例预测器是否会阻塞 store 指令，设 1 代表会阻塞 | 0 |
| 4-8 | 访存违例预测器的 reset 间隔，设该位域的值为 x，则间隔为 2^(10+x) | 6 |
| 其余位 | 目前其余位无功能 | 0 |

### 5.3.4 smblockctl

smblockctl 的地址为 0x5C3，其初始化值为下表的默认值，其每一 bit 的功能如下表所示

表 5.14: smblockctl 的 bit 功能

| 位 | 功能 | 默认值 |
|---|---|---|
| 0-3 | 控制 sbuffer 的 flush 阈值 | 7 |
| 4 | 控制是否开启 ld-ld 违例检查，设 1 代表开启 | 1 |
| 5 | 控制是否开启 soft prefetch，设 1 代表开启 | 1 |
| 6 | 控制是否上报 cache 发生的 ecc 错误，设 1 代表开启 | 1 |
| 7 | 控制是否支持 uncache 的 outstanding 访问，设 1 代表开启 | 0 |
| 其余位 | 目前其余位无功能 | 0 |

### 5.3.5 srnctl

srnctl 的地址为 0x5C4，其初始化值为下表的默认值，其每一 bit 的功能如下表所示

表 5.15: srnctl 的 bit 功能

| 位 | 功能 | 默认值 |
|---|---|---|
| 0 | fusion decoder 是否开启，1 开启 | 1 |
| 1 | speculative virtual address inv 是否开启 | 1 |
| 2 | wfi 指令是否开启 | 1 |
| 其余位 | 目前其余位无功能 | 0 |

# 6 异常与中断

## 6.1 概述

异常和中断是处理器从正常程序执行流跳转到特权模式的一种机制，用于应对各种意外或预期的事件。这些机制帮助处理器解决执行过程中遇到的问题，维护系统的稳定性，处理外部事件的响应。

- 异常（Exception）是指处理器在执行当前指令时遇到无法继续执行的情况，需要立即暂停正常的程序流，并进入异常处理程序。异常通常是同步的，即它们与当前正在执行的指令直接相关。常见的异常类型包括：

  1. 非法指令：当处理器遇到无法识别或执行的指令时，会产生非法指令异常。
  2. 地址未对齐异常：当执行加载或存储操作时，访问的内存地址未对齐，处理器会引发异常。
  3. 页面错误：在虚拟内存系统中，当页表缺失或内存访问权限不足时，会发生页错误异常。
  4. 环境调用（系统调用）：应用程序通过特定指令请求内核服务时，也会触发异常。

- 中断（Interrupt）是由外部硬件设备（如计时器、外设等）触发的信号，要求处理器暂停当前执行的程序，去处理这些外部事件。中断是异步的，即它们与当前指令执行无关，可能在任何时刻发生。根据中断来源的不同，可以分为以下几类：

  1. 硬件中断：由外部硬件设备产生，如外设的输入输出请求、定时器中断等。
  2. 软件中断：由软件触发，用于与操作系统交互或执行某些系统级别的任务。

- 陷入（Trap）是一种处理器从用户模式或较低特权级别切换到特权模式（如内核模式）处理事件的机制。处理器通过陷入机制捕获异常或中断，保存上下文信息，并跳转到相应的处理程序。处理完成后，处理器通过返回指令（如 mret 或 sret）恢复之前的执行状态，继续执行原来的程序。

## 6.2 异常

昆明湖 V2R2 支持多种异常，如下表所示：

表 6.1: 昆明湖 V2R2 支持的异常列表

| 异常编号 | trap 原因 | tval 更新值 | tval2 更新值 |
|---|---|---|---|
| 0 | 指令地址非对齐 | 0 | 0 |
| 1 | 取指令权限异常（非跨页） | 指令起始地址 | 0 |
| 1 | 取指令权限异常（跨页） | 下一页首地址 | 0 |
| 2 | 非法指令 | 非法指令编码 | 0 |

| 异常编号 | trap 原因 | tval 更新值 | tval2 更新值 |
|---|---|---|---|
| 3 | 断点 | 异常指令地址 | 0 |
| 3 | EBREAK 指令 | 0 | 0 |
| 4 | LOAD 地址非对齐 | 访存的起始地址 | 0 |
| 5 | LOAD 权限异常 | 实际 fault 的起始地址 | 0 |
| 6 | STORE/AMO 地址非对齐 | 访存的起始地址 | 0 |
| 7 | STORE/AMO 权限异常 | 实际 fault 的起始地址 | 0 |
| 8 | U-ECALL | 0 | 0 |
| 9 | S-ECALL | 0 | 0 |
| 10 | VS-ECALL | 0 | 0 |
| 11 | M-ECALL | 0 | 0 |
| 12 | 指令页表异常（非跨页） | 指令起始地址 | 0 |
| 12 | 指令页表异常（跨页） | 下一页首地址 | 0 |
| 13 | LOAD 页表异常（非跨页） | 访存的起始地址 | 0 |
| 13 | LOAD 页表异常（跨页） | 实际 fault 的起始地址 | 0 |
| 15 | STORE/AMO 页表异常（非跨页） | 访存的起始地址 | 0 |
| 15 | STORE/AMO 页表异常（跨页） | 实际 fault 的起始地址 | 0 |
| 20 | 客户机指令页表异常（非跨页） | 指令起始地址 | 对应的 GPA |
| 20 | 客户机指令页表异常（跨页） | 下一页首地址 | 实际 fault 的 GPA |
| 21 | 客户机 LOAD 页表异常（非跨页） | 访存的起始地址 | 对应的 GPA |
| 21 | 客户机 LOAD 页表异常（跨页） | 实际 fault 的起始地址 | 实际 fault 的 GPA |
| 22 | 虚拟化指令异常 | 非法指令编码 | 0 |
| 23 | 客户机 STORE/AMO 页表异常（非跨页） | 访存的起始地址 | 对应的 GPA |
| 23 | 客户机 STORE/AMO 页表异常（跨页） | 实际 fault 的起始地址 | 实际 fault 的 GPA |

注意：香山实际上不存在指令地址非对齐异常。

我们以机器模式为例，介绍异常处理的步骤。

### 6.2.1 异常响应

第一步：处理器保存发生异常的 PC 到 mepc 中。

第二步：将 mcause 的中断标志位置为 0 ，同时将异常编号写入 mcause ，并修改对应的 mtval。

第三步：将 mstatus 的 MIE 位保存到 MPIE 位，然后将 MIE 位清零。将 mstatus 的 MPV 位与 MPP 位分别置为当前的 Virtual Mode 和 Privilege Mode，并根据条件修改对应的 GVA 位。

第四步：根据 mtvec 取对应的指令并执行。

### 6.2.2 异常返回

异常返回通过 mret 指令实现，该指令执行下列操作。

- 根据 mepc 恢复 PC。
- 根据 mstatus 的 MPIE 位恢复 mstatus 的 MIE 位。
- 将当前的 Privilege Mode 改为 mstatus 的 MPP 位，且如果 MPP 位为机器模式，则将 Virtual Mode 置为 0，否则置为 mstatus 的 MPV 位。

## 6.3 中断

昆明湖 V2R2 支持多种中断，如下表所示：

表 6.2: 昆明湖 V2R2 支持的中断列表

| 中断编号 | trap 原因 |
|---|---|
| 1 | 监管模式软件中断（SSI） |
| 2 | 虚拟监管模式软件中断（VSSI） |
| 3 | 机器模式软件中断（MSI） |
| 5 | 监管模式定时器中断（STI） |
| 6 | 虚拟监管模式定时器中断（VSTI） |
| 7 | 机器模式定时器中断（MTI） |
| 9 | 监管模式外部中断（SEI） |
| 10 | 虚拟监管模式外部中断（VSEI） |
| 11 | 机器模式外部中断（MEI） |
| 12 | 监管模式客户机外部中断（SGEI） |
| 13 | 本地计数器溢出中断（LCOFIP） |
| 16-23 | 标准本地中断 |
| 24-31 | 自定义中断 |
| 32-47 | 标准本地中断 |
| 48-63 | 自定义中断 |

我们以机器模式为例，介绍中断处理的步骤

### 6.3.1 中断优先级

昆明湖 V2R2 的中断优先级，如下表所示：

表 6.3: 昆明湖 V2R2 中断优先级

| 中断优先级 | 组别 | 编号 | 描述 |
|---|---|---|---|
| 最高 | Custom Group 0 | 63, 31, 62 | 最高优先级 Custom 中断 |
| | | 61, 30, 60 | |
| | Local Group 0 | 47, 23, 46 | 高优先级 Local 中断 |
| | | 45, 22, 44 | |
| | | 43, 21, 42 | |
| | | 41, 20, 40 | |
| | Custom Group 1 | 59, 29, 58 | 次高优先级 Custom 中断 |
| | | 57, 28, 56 | |
| | TSEO Group | 11, 3, 7 | 机器模式中断：外部、软件、时钟 |
| | | 9, 1, 5 | 监管模式中断：外部、软件、时钟 |
| | | 12 | 监管模式客户机外部中断 |
| | | 10, 2, 6 | 虚拟监管模式中断：外部、软件、时钟 |

| 中断优先级 | 组别 | 编号 | 描述 |
|---|---|---|---|
| | | 13 | 本地计数器溢出中断 |
| | Custom Group 2 | 55, 27, 54 | 中等优先级 Custom 中断 |
| | | 53, 26, 52 | |
| | Local Group 1 | 39, 19, 38 | 低优先级 Local 中断 |
| | | 37, 18, 36 | |
| | | 35, 17, 34 | |
| | | 33, 16, 32 | |
| | Custom Group 3 | 51, 25, 50 | 最低优先级 Custom 中断 |
| 最低 | | 49, 24, 48 | |

### 6.3.2 中断响应

第一步：处理器执行完当前指令后，将下一条指令的 PC 存入 mepc。

第二步：将 mcause 的中断标志位置为 1 ，同时将异常编号写入 mcause ，并将 mtval 置为 0。

第三步：将 mstatus 的 MIE 位保存到 MPIE 位，然后将 MIE 位清零。将 mstatus 的 MPV 位与 MPP 位分别置为当前的 Virtual Mode 和 Privilege Mode，并根据条件修改对应的 GVA 位。

第四步：根据 mtvec 取对应的指令并执行。

### 6.3.3 中断返回

- 根据 mepc 恢复 PC。
- 根据 mstatus 的 MPIE 位恢复 mstatus 的 MIE 位。
- 将当前的 Privilege Mode 改为 mstatus 的 MPP 位，且如果 MPP 位为机器模式，则将 Virtual Mode 置为 0，否则置为 mstatus 的 MPV 位。

# 7 内存模型

## 7.1 内存模型概述

### 7.1.1 内存属性

昆明湖 V2R2 支持三种内存类型，分别是可高速缓存内存、不可高速缓存内存和不可缓存外设。

地址的物理内存属性（Physical Memory Attributes）由硬件规定，其支持 RISC-V 手册中所规定的属性：

- R（Read Permission）：可读权限
- W（Write Permission）：可写权限
- X（Execute Permission）：可执行权限
- A（Address matching mode）：地址匹配模式
- L（Lock Bit）：锁定状态

同时昆明湖 V2R2 还支持 atomic（支持原子指令）属性和 c（支持高速缓存）属性。

### 7.1.2 内存一致性模型

可高速缓存内存采用 RVWMO（RISC-V Weak Memory Ordering）内存模型。在该模型下，多核之间内存实际的读写顺序和程序给定的访问顺序会不一致。因此 RISC-V 的指令集架构中提供了 Fence 指令来保证内存访问的同步。同时 RISC-V 的 A 扩展中还提供了 LR/SC 指令和 AMO 指令来进行加锁和原子操作。

不可高速缓存内存和不可缓存外设两种内存类型都是强有序（strongly ordered）的。

## 7.2 虚拟内存管理

### 7.2.1 MMU 概述

昆明湖 V2R2 MMU（Memory Management Unit）支持 RISC-V SV48 和 SV39 分页机制。其作用主要有：

- 地址转换：将虚拟地址（48 或 39 位）转换为物理地址（48 位）；
- 页面保护：对页面的访问者进行读/写/执行权限的检查；
- 虚拟化支持：支持虚拟化环境下通过两阶段地址翻译将客户机虚拟地址转换为主机物理地址；
- 异常处理：各模块根据请求来源返回异常。

### 7.2.2 TLB 组织形式

MMU 主要利用 TLB（Translation Look-aside Buffer）来实现地址转换等功能。TLB 将 CPU 访存使用多个虚拟地址作为输入，转换前检查 TLB 的页面属性，再输出该虚拟地址对应的物理地址。

昆明湖 V2R2 MMU 采用两级 TLB 的组织结构，第一级为 L1 TLB，分别为指令 ITLB 与数据 DTLB；第二级为 L2 TLB。

L1 ITLB 的项配置如下表

| 项名 | 项数 | 组织结构 | 替换算法 | 存储内容 |
|---|---|---|---|---|
| Page | 48 | 全相联 | PLRU | 全部大小页 |

L1 DTLB 的项配置如下表

| 项名 | 项数 | 组织结构 | 替换算法 | 存储内容 |
|---|---|---|---|---|
| Page | 48 | 全相联 | PLRU | 全部大小页 |

取指请求、加载与存储请求在访问 ITLB 与 DTLB 时，如果命中，在下一拍可以得到物理地址与对应权限属性。

L2 TLB 为指令和数据共用，L2 TLB 包含六个主要单元：

1. Page Cache：缓存页表，Sv48 的 4 层页表全部且分开缓存，可以一拍完成 4 层信息的查询。另外，对于虚拟化的请求，Page Cache 会分开存储第一阶段（va -> gpa）和第二阶段（gpa -> hpa）的请求，不会直接存储 va -> hpa 的映射。
2. Page Table Walker：查询内存中的前两级页表；
3. Last Level Page Table Walker：查询内存中的最后一级页表；
4. Hypervisor Page Table Walker：负责查询第二阶段的页表转换
5. Miss Queue：缓存查询 Page Cache 和 Last Level Page Walker 的 miss 请求；
6. Prefetcher：预取器。

### 7.2.3 地址转换流程

MMU 负责将虚拟地址翻译成物理地址，并用翻译得到的物理地址进行访存。昆明湖 V2R2 支持 Sv39/Sv48 分页机制，虚拟地址长度为 39/48 位，低 12 位是页内偏移，支持 Sv39 时高 27 位分为三段（每段 9 位），也就是三级页表，支持 Sv48 时高 36 位分为四段（每段 9 位），也就是四级页表。

昆明湖 V2R2 的物理地址为 48 位，虚拟地址和物理地址的结构如图所示。遍历页表需要进行四次内存访问，需要通过 TLB 对页表做缓存。

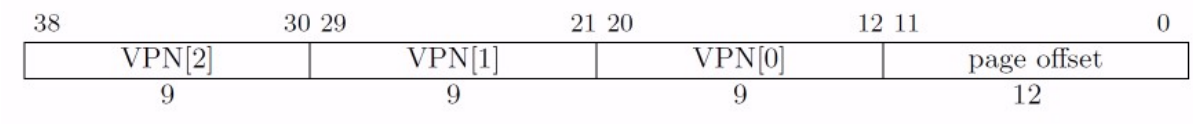


图 7.1: Sv38 虚拟地址结构

在进行地址翻译时，前端取指通过 ITLB 进行地址翻译，后端访存通过 DTLB 进行地址翻译。ITLB 和 DTLB 如果 miss，会通过 Repeater 向 L2 TLB 发送请求。前端取指和后端访存对 TLB 均采用非阻塞式访问，如果 TLB 返回缺失，可以重新调度查询其他请求。

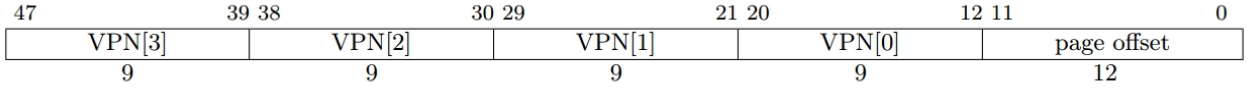


图 7.2: Sv48 虚拟地址结构

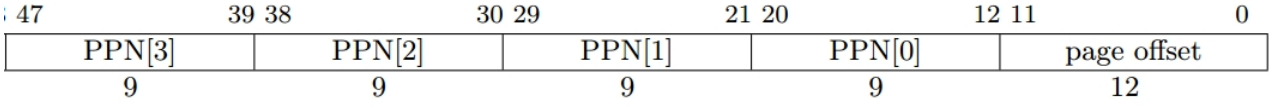


图 7.3: 物理地址结构

昆明湖 V2R2 页表用于存储下级页表的入口地址或者最终页表的物理信息，页表结构如图 6.3 所示：

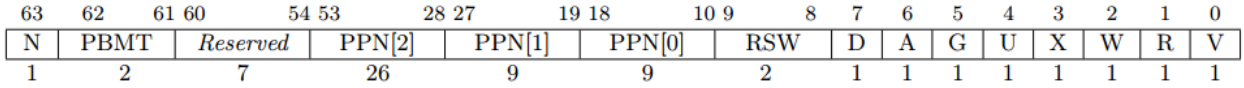


图 7.4: 页表结构

昆明湖 V2R2 页表结构各 bit 属性：

**PBMT：Page-Based Memory Types**

PBMT 位表示虚拟地址的 PMA 属性，具体定义按场景划分，如图所示。(注：此功能需要开启 menvcfg 寄存器的 PBMTE 位)

- PMA 模式兼容原来未定义虚拟地址 PMA 时的系统属性设定，遵循原物理地址 PMA 属性，即 sysmap 属性；
- NC 模式下，强制覆盖不可缓存、幂等、弱排序属性，一般用于主存；
- IO 模式下，强制覆盖不可缓存、不幂等、强排序属性，一般用于外设；
- PBMT 位为 3 时的情况为保留模式，未来进行定义。
- SVPBMT 中未涉及到的 PMA 属性，以原来物理地址的 PMA 为准，强制覆盖的属性，以 PBMT 为准。

**RSW：Reserved for use by Supervisor softWare**

给特权态软件保留的页表属性位，硬件不做处理。

**D：Dirty bit ：** 表示页面自上次清除 D 位以来是否被写入。

**A：Access bit:** 表示页面自上次清除 A 位以来是否被访问（包括访存和取指）。

**G：Global bit:** 表示页面作用于全局。

**U：User bit:** 表示页面为 User 态专用。

**X：Executable bit:** 表示页面可执行。

**W：Writeable bit:** 表示页面可写。

**R：Readable bit:** 表示页面可读。

**V：Valid bit:** 表示页面有效。

昆明湖 V2R2 支持 Svade 拓展，如果页面在没有设置 A 位的情况下被访问（访存或取指），或者页面在没有设置 D 位的情况下被写入，都会触发 page fault。不支持 Svadu 拓展，不会硬件更新 A/D 位。

地址转换的详细流程描述如下，此处以 Sv39 为例：

| Mode | Value | Requested Memory Attributes |
|---|---|---|
| PMA | 0 | None |
| NC | 1 | Non-cacheable, idempotent, weakly-ordered (RVWMO or RVTSO), main memory |
| IO | 2 | Non-cacheable, non-idempotent, strongly-ordered (I/O ordering), I/O |
| - | 3 | Reserved for future standard use |

图 7.5: PMBT 的定义

CPU 要访问某个虚拟地址，若 TLB 命中，则从 TLB 中直接获取物理地址及相关属性。若 TLB 缺失，则地址的转换具体步骤为：

1. 根据 SATP.PPN 和 VPN[2] 得到一级页表访存地址 {SATP.PPN, VPN[2], 3'b0}，使用该地址访问 L2 Cache，得到 64-bit 一级页表 PTE;
2. 检查 PTE 是否符合 PMP 权限，若不符合则产生相应 access fault 异常；若符合则根据规则判断 X/W/R 位是否符合叶子页表条件，若符合叶子页表条件则说明已经找到最终物理地址，到第 3 步；若不符合则回到第 1 步，使用 pte.ppn 代替 satp.ppn，vpn 换为下一级 vpn，再拼接 3'b0 继续第一步流程；
3. 找到了叶子页表，结合 PMP 中的 X/W/R/L 位和 PTE 中的 X/W/R 位得到两者的最小权限进行权限检查，并将 PTE 的内容回填到 L2 TLB 中；
4. 在任何一步的 PMP 检查中，如果有权限违反，则根据访问类型产生对应的 access fault 异常；
5. 若得到叶子页表，但：访问类型违反 A/D/X/W/R/U-bit 的设置，产生对应的 page fault 异常；若三次访问结束仍未得到叶子页表，则产生对应的 page fault 异常；若访问 L2 Cache 过程中得到 access fault 响应，产生 page fault 异常；
6. 若得到叶子页表，但访问次数少于 3 次，则说明得到了大页表。检查大页表的 PPN 是否按照页表尺寸对齐，若未对齐，则产生 page fault 异常。

这里额外说明，{{processor_name}} 不允许访问建立在 MMIO 地址空间的页表。如果在页表遍历过程中，查询得到某级页表的地址位于 MMIO 空间中，会上报 access fault 异常。

### 7.2.4 虚拟化两阶段地址转换

昆明湖 V2R2 支持 H 拓展，在非虚拟化模式且未执行虚拟化访存指令时，地址翻译过程与未加入 H 拓展时基本一致，在虚拟化模式或者执行虚拟化访存指令时，需要判断是否开启两阶段地址翻译。

CPU 根据 vsatp 与 hgatp 开启两阶段翻译，vsatp 结构如下图所示（昆明湖 V2R2 中 SXLEN 固定为 64）

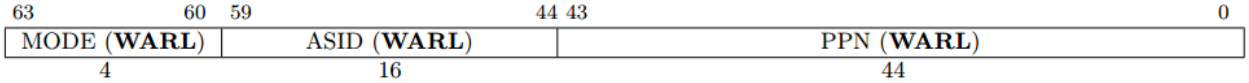


图 7.6: vsatp 结构

vsatp 控制 VS-stage 阶段翻译，一般默认 vsatp 保持开启，即在虚拟化模式或者执行虚拟化访存指令时，默认开启 VS-stage 阶段地址翻译。

hgatp 控制 G-stage 阶段翻译，hgatp 结构与翻译模式如下图所示

| 位 | 域 | 描述 |
|---|---|---|
| [63:60] | MODE | 表示地址转换的模式。该域为 0 时为 Bare mode，不开启地址翻译或地址保护，该域为 8/9 时表示 Sv39x4/Sv48x4 地址转换模式，如果该域为其他值会上报 illegal instruction fault |
| [57:44] | VMID | 虚拟机标识符。对于昆明湖 V2R2 采用的 Sv39x4/Sv48x4 地址转换模式，VMID 长度最大值都为 14 |
| [43:0] | PPN | 表示第二阶段翻译的根页表的物理页号，由物理地址右移 12 位得到 |

昆明湖 V2R2 使用 Sv39x4/Sv48x4 分页机制，两机制的虚拟地址结构如下图所示

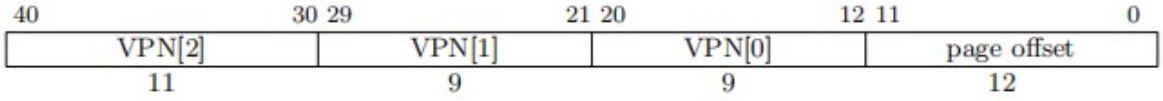


图 7.7: sv39x4 虚拟地址结构

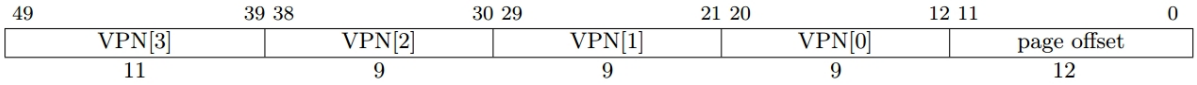


图 7.8: sv48x4 虚拟地址结构

VS-stage 负责将客户机虚拟地址转换成客户机物理地址，G-stage 负责将客户机的物理地址转换成主机的物理地址。第一阶段的翻译和非虚拟化的翻译过程基本一致，第二阶段的翻译在 PTW 与 LLPTW 模块中进行，查询逻辑为：首先在 Page Cache 中查找，如果找到则返回给 PTW 或者 LLPTW，如果没找到就进入 HPTW 进行翻译，由 HPTW 返回并填入 Page Cache。

在两阶段地址翻译中，第一阶段翻译得到的地址（包括翻译过程中计算得到的页表地址）均为客户机的物理地址，需要进行第二阶段翻译得到真实的物理地址后才能进行访存读取页表。逻辑上的 Sv39x4 与 Sv48x4 翻译过程如下图所示。

### 7.2.5 系统控制寄存器

**MMU 地址转换寄存器（SATP）**

昆明湖 V2R2 架构支持长度为 16 的 ASID（地址空间标识符），在 SATP 寄存器中保存。SATP 寄存器的格式如表所示。

| 位 | 域 | 描述 |
|---|---|---|
| [63:60] | MODE | 表示地址转换的模式。该域为 0 时为 Bare mode，不开启地址翻译或地址保护，该域为 8/9 时表示 Sv39/Sv48 地址转换模式，如果该域为其他值会上报 illegal instruction fault |
| [59:44] | ASID | 地址空间标识符。ASID 的长度可参数化配置，对于昆明湖 V2R2 采用的 Sv39/Sv48 地址转换模式，ASID 长度最大值为 16 |
| [43:0] | PPN | 表示根页表的物理页号，由物理地址右移 12 位得到 |

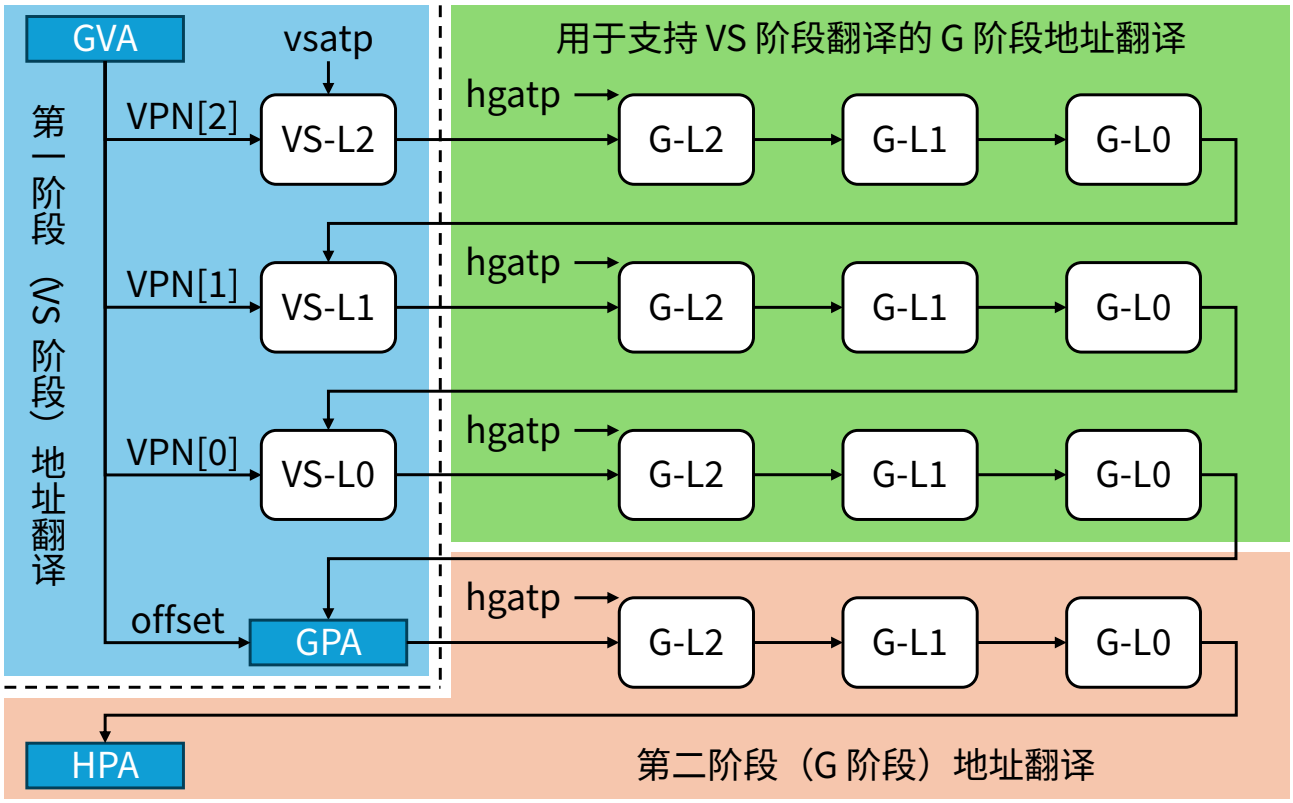


图 7.9: VS 阶段为 Sv39、G 阶段为 Sv39x4 的两阶段地址翻译

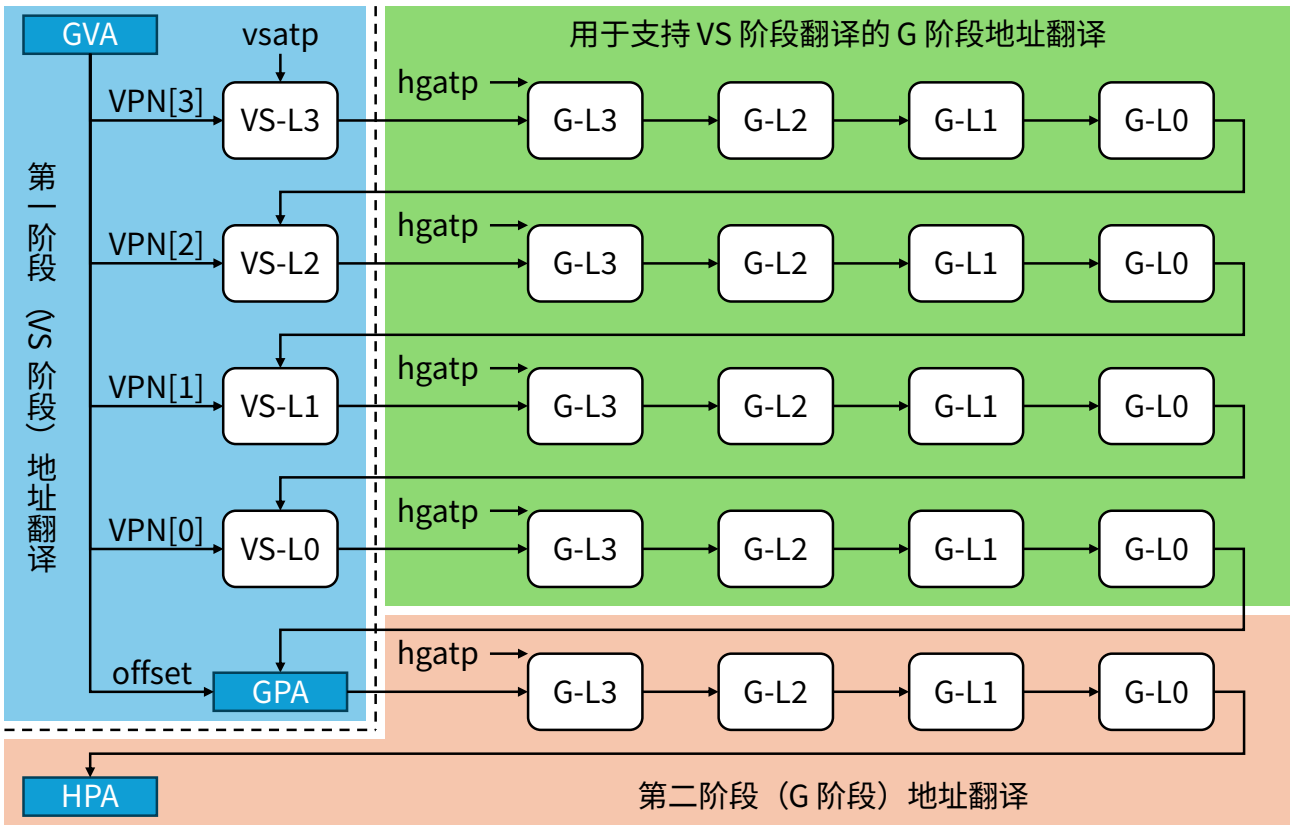


图 7.10: VS 阶段为 Sv48、G 阶段为 Sv48x4 的两阶段地址翻译

在虚拟化模式下，SATP 将被 VSATP 寄存器代替，并且其中的 PPN 为客户机根页表的客户机物理页号，而非真实的物理地址，需要进行第二阶段翻译才能得到真实物理地址。

## 7.3 物理内存保护 & 物理地址属性

### 7.3.1 PMP 概述

昆明湖 V2R2 遵从 RISC-V 标准。PMP（Physical Memory Protection）单元负责对物理地址的访问权限进行检查，判定当前工作模式下 CPU 是否具备对该地址的读/写/执行权限。

PMA（Physical Memory Attributes）单元的作用是定义物理内存的属性。它主要负责指定内存区域的类型，定义内存访问特性与控制内存访问的行为。

PMA 与 PMP 配合使用，共同管理和控制物理内存的访问和属性，确保系统的安全性和性能。

昆明湖 V2R2 PMP&PMA 单元的设计规格有：

- 支持物理地址保护；
- 支持物理地址属性；
- 支持 PMP&PMA 并行执行检查；
- 支持分布式 PMP 与分布式 PMA；
- 支持异常处理。

### 7.3.2 PMP 控制寄存器

昆明湖 V2R2 默认 PMP 为 16 项，可以通过参数化修改，采用分布复制式实现方法。PMP 表项主要由一个 8bit 的配置寄存器与一个 64bit 的地址寄存器构成，默认没有初始值。

PMP 寄存器的地址空间如下表

| PMP 寄存器名 | 地址 |
|---|---|
| pmpcfg0 | 0x3A0 |
| pmpcfg2 | 0x3A2 |
| pmpaddr0 | 0x3B0 |
| pmpaddr1 | 0x3B1 |
| pmpaddr2 | 0x3B2 |
| pmpaddr3 | 0x3B3 |
| pmpaddr4 | 0x3B4 |
| pmpaddr5 | 0x3B5 |
| pmpaddr6 | 0x3B6 |
| pmpaddr7 | 0x3B7 |
| pmpaddr8 | 0x3B8 |
| pmpaddr9 | 0x3B9 |
| pmpaddr10 | 0x3BA |
| pmpaddr11 | 0x3BB |
| pmpaddr12 | 0x3BC |
| pmpaddr13 | 0x3BD |
| pmpaddr14 | 0x3BE |

| PMP 寄存器名 | 地址 |
|---|---|
| pmpaddr15 | 0x3BF |

其中，PMP 配置寄存器 pmpcfg0，pmpcfg2 的布局如图所示

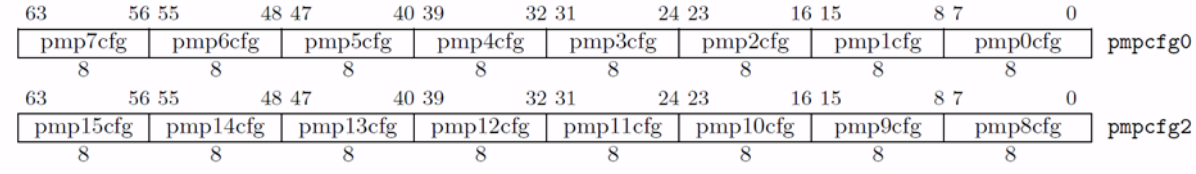


图 7.11: PMP 地址寄存器

PMP 配置寄存器的格式如下表

| 位 | 域 | 描述 |
|---|---|---|
| 7 | L | 表示该 PMP 表项是否被锁定。该域默认为 0，表示未被锁定；为 1 时表示被锁定，需复位才可解除 |
| [6:5] | 保留 | 保留位，默认为 0 |
| [4:3] | A | 表示该 PMP 表项的地址匹配模式。该域默认为 0，表示 PMP 表项被禁用且不匹配任何地址。该域为 1、2、3 时分别表示 TOR、NA4、NAPOT 地址匹配模式 |
| 2 | X | 表示该 PMP 表项配置的地址是否支持指令执行。该域为 1 时表示支持指令执行，为 0 时不支持指令执行 |
| 1 | W | 表示该 PMP 表项配置的地址是否支持写。该域为 1 时表示支持写，为 0 时不支持写 |
| 0 | R | 表示该 PMP 表项配置的地址是否支持读。该域为 1 时表示支持读，为 0 时不支持读 |

PMP 地址寄存器的格式如下表

| 位 | 域 | 描述 |
|---|---|---|
| [63:34] | 0 | PMP 地址寄存器的高 30 位为 0 |
| [33:0] | address | 存储 PMP 表项配置的地址的 [35:2] 位 |

### 7.3.3 PMA 属性寄存器

PMA 的实现采用类似 PMP 的方式，PMA 寄存器默认拥有初始值，需手动设置与平台地址属性一致。PMA 寄存器利用了 M 态 CSR 的保留寄存器地址空间，默认为 16 项。

PMA 寄存器的地址空间如下表

| PMA 寄存器名 | 地址 | 复位值 |
|---|---|---|
| pmacfg0 | 0x7C0 | 64'h80b080d08000000 |
| pmacfg2 | 0x7C2 | 64'h6f0b080b080f080b |

| PMA 寄存器名 | 地址 | 复位值 |
|---|---|---|
| pmaaddr0 | 0x7C8 | 64'h0 |
| pmaaddr1 | 0x7C9 | 64'h0 |
| pmaaddr2 | 0x7CA | 64'h0 |
| pmaaddr3 | 0x7CB | 64'h4000000 |
| pmaaddr4 | 0x7CC | 64'h8000000 |
| pmaaddr5 | 0x7CD | 64'hc000000 |
| pmaaddr6 | 0x7CE | 64'hc4c4000 |
| pmaaddr7 | 0x7CF | 64'he000000 |
| pmaaddr8 | 0x7D0 | 64'he004000 |
| pmaaddr9 | 0x7D1 | 64'he008000 |
| pmaaddr10 | 0x7D2 | 64'he008400 |
| pmaaddr11 | 0x7D3 | 64'he400000 |
| pmaaddr12 | 0x7D4 | 64'he400800 |
| pmaaddr13 | 0x7D5 | 64'hf000000 |
| pmaaddr14 | 0x7D6 | 64'h20000000 |
| pmaaddr15 | 0x7D7 | 64'h120000000 |

其中，pmacfg0、pmacfg2 的布局如图所示

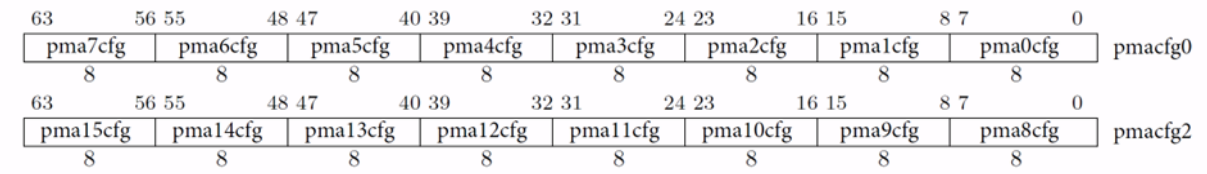


图 7.12: PMA 地址寄存器

PMA 配置寄存器的格式如下表

| 位 | 域 | 描述 |
|---|---|---|
| 7 | L | 表示该 PMA 表项是否被锁定。该域默认为 0，表示未被锁定；为 1 时表示被锁定 |
| 6 | C | 表示该 PMA 表项配置的地址是否可缓存。该域为 1 时表示可缓存，为 0 时表示该地址属于 mmio 空间，不可缓存 |
| 5 | Atomic | 表示该 PMA 表项配置的地址是否允许原子访问。该域为 1 时表示可原子访问，为 1 时表示不可原子访问 |
| [4:3] | A | 表示该 PMA 表项的地址匹配模式。该域默认为 0，表示 PMA 表项被禁用且不匹配任何地址。该域为 1、2、3 时分别表示 TOR、NA4、NAPOT 地址匹配模式 |
| 2 | X | 表示该 PMA 表项配置的地址是否支持指令执行。该域为 1 时表示支持指令执行，为 0 时不支持指令执行 |
| 1 | W | 表示该 PMA 表项配置的地址是否支持写。该域为 1 时表示支持写，为 0 时不支持写 |
| 0 | R | 表示该 PMA 表项配置的地址是否支持读。该域为 1 时表示支持读，为 0 时不支持读 |

PMA 地址寄存器的格式如下表

| 位 | 域 | 描述 |
|---|---|---|
| [63:34] | 0 | PMA 地址寄存器的高 30 位为 0 |
| [33:0] | address | 存储 PMA 表项配置的地址的 [35:2] 位 |

PMA 寄存器描述的地址空间及属性如下表

| 地址下界 | 地址上界 | 描述 | 属性 |
|---|---|---|---|
| 0x00_0000_0000 | 0x00_0FFF_FFFF | Reserved | |
| 0x00_1000_0000 | 0x00_1FFF_FFFF | QSPI Flash | RX |
| 0x00_2000_0000 | 0x00_2FFF_FFFF | Reserved | |
| 0x00_3000_0000 | 0x00_3000_FFFF | GPU(V550) | RW |
| 0x00_3001_0000 | 0x00_3001_FFFF | G71 | RW |
| 0x00_3002_0000 | 0x00_3003_FFFF | Reserved | |
| 0x00_3004_0000 | 0x00_3004_FFFF | DMA | RW |
| 0x00_3005_0000 | 0x00_3005_FFFF | SDMMC | RW |
| 0x00_3006_0000 | 0x00_3015_FFFF | USB | RW |
| 0x00_3016_0000 | 0x00_3025_FFFF | DATA_CPU_BRIDGE | RW |
| 0x00_3026_0000 | 0x00_30FF_FFFF | Reserved | |
| 0x00_3100_0000 | 0x00_3100_FFFF | QSPI | RW |
| 0x00_3101_0000 | 0x00_3101_FFFF | GMAC | RW |
| 0x00_3102_0000 | 0x00_3102_FFFF | HDMI | RW |
| 0x00_3103_0000 | 0x00_3103_FFFF | HDMI_PHY | RW |
| 0x00_3104_0000 | 0x00_3105_FFFF | DP | RW |
| 0x00_3106_0000 | 0x00_3106_FFFF | DDR0 | RW |
| 0x00_3107_0000 | 0x00_3107_FFFF | DDR0_PHY | RW |
| 0x00_3108_0000 | 0x00_3108_FFFF | DDR1 | RW |
| 0x00_3109_0000 | 0x00_3109_FFFF | DDR1_PHY | RW |
| 0x00_310A_0000 | 0x00_310A_FFFF | IIS | RW |
| 0x00_310B_0000 | 0x00_310B_FFFF | UART0 | RW |
| 0x00_310C_0000 | 0x00_310C_FFFF | UART1 | RW |
| 0x00_310D_0000 | 0x00_310D_FFFF | UART2 | RW |
| 0x00_310E_0000 | 0x00_310E_FFFF | IIC0 | RW |
| 0x00_310F_0000 | 0x00_310F_FFFF | IIC1 | RW |
| 0x00_3110_0000 | 0x00_3110_FFFF | IIC2 | RW |
| 0x00_3111_0000 | 0x00_3111_FFFF | GPIO | RW |
| 0x00_3112_0000 | 0x00_3112_FFFF | CRU | RW |
| 0x00_3113_0000 | 0x00_3113_FFFF | WDT | RW |
| 0x00_3114_0000 | 0x00_3114_FFFF | USB2_PHY0 | RW |
| 0x00_3115_0000 | 0x00_3115_FFFF | USB2_PHY1 | RW |
| 0x00_3116_0000 | 0x00_3116_FFFF | USB2_PHY2 | RW |

| 地址下界 | 地址上界 | 描述 | 属性 |
|---|---|---|---|
| 0x00_3117_0000 | 0x00_3117_FFFF | USB2_PHY3 | RW |
| 0x00_3118_0000 | 0x00_3118_FFFF | USB3_PHY0 | RW |
| 0x00_3119_0000 | 0x00_3119_FFFF | USB3_PHY1 | RW |
| 0x00_311a_0000 | 0x00_311a_FFFF | USB3_PHY2 | RW |
| 0x00_311b_0000 | 0x00_311b_FFFF | USB3_PHY3 | RW |
| 0x00_311c_0000 | 0x00_311c_FFFF | PCIE0_CFG | RW |
| 0x00_311d_0000 | 0x00_311d_FFFF | PCIE1_CFG | RW |
| 0x00_311e_0000 | 0x00_311e_FFFF | PCIE2_CFG | RW |
| 0x00_311f_0000 | 0x00_311f_FFFF | PCIE3_CFG | RW |
| 0x00_3120_0000 | 0x00_3120_FFFF | SYSCFG | RW |
| 0x00_3121_0000 | 0x00_3130_FFFF | DATA_CPU_BRIDGE | RW |
| 0x00_3131_0000 | 0x00_37FF_FFFF | Reserved | |
| 0x00_3800_0000 | 0x00_3800_FFFF | CLINT (In cpu) | RW |
| 0x00_3801_0000 | 0x00_3801_FFFF | Reserved | |
| 0x00_3802_0000 | 0x00_3802_0FFF | Debug (In cpu) | RW |
| 0x00_3802_1000 | 0x00_38FF_FFFF | Reserved | |
| 0x00_3900_0000 | 0x00_3900_0FFF | CacheCtrl | RW |
| 0x00_3900_1000 | 0x00_3900_1FFF | Core Reset | RW |
| 0x00_3900_2000 | 0x00_3BFF_FFFF | Reserved | |
| 0x00_3C00_0000 | 0x00_3FFF_FFFF | PLIC (In cpu) | RW |
| 0x00_4000_0000 | 0x00_4FFF_FFFF | PCIe0 | RW |
| 0x00_5000_0000 | 0x00_5FFF_FFFF | PCIe1 | RW |
| 0x00_6000_0000 | 0x00_6FFF_FFFF | PCIe2 | RW |
| 0x00_7000_0000 | 0x00_7FFF_FFFF | PCIe3 | RW |
| 0x00_8000_0000 | 0x1F_FFFF_FFFF | DDR | RWXIDSA |

PMA 寄存器的条目信息如下表

| 地址 | c | atomic | a | x | w | r |
|---|---|---|---|---|---|---|
| 0x0 | false | false | 0 | false | false | false |
| 0x0 | false | false | 0 | false | false | false |
| 0x0 | false | false | 0 | false | false | false |
| 0x10000000 | false | false | 1 | false | false | false |
| 0x20000000 | false | false | 1 | true | false | true |
| 0x30000000 | false | false | 1 | false | false | false |
| 0x31310000 | false | false | 1 | false | true | true |
| 0x38000000 | false | false | 1 | false | false | false |
| 0x38010000 | false | false | 1 | false | true | true |
| 0x38020000 | false | false | 1 | false | false | false |
| 0x38021000 | false | false | 1 | true | true | true |
| 0x39000000 | false | false | 1 | false | false | false |

| 地址 | c | atomic | a | x | w | r |
|---|---|---|---|---|---|---|
| 0x39002000 | false | false | 1 | false | true | true |
| 0x3c000000 | false | false | 1 | false | false | false |
| 0x80000000 | false | false | 1 | false | true | true |
| 0x480000000 | true | true | 1 | true | true | true |

## 7.4 异常处理机制

昆明湖 V2R2 MMU 模块可能产生 4 种异常，包括 guest page fault、page fault、access fault 以及 L2 TLB page cache 的 ECC 校验出错。根据请求来源分别交付给对应模块处理。

MMU 可能产生的异常以及处理流程如下表

| 模块 | 可能产生的异常 | 处理流程 |
|---|---|---|
| ITLB | | |
| | 产生 inst page fault | 根据请求来源，分别交付给 Icache 或 IFU 处理 |
| | 产生 inst guest page fault | 根据请求来源，分别交付给 Icache 或 IFU 处理 |
| | 产生 inst access fault | 根据请求来源，分别交付给 Icache 或 IFU 处理 |
| DTLB | | |
| | 产生 load page fault | 交付给 LoadUnits 进行处理 |
| | 产生 load guest page fault | 交付给 LoadUnits 进行处理 |
| | 产生 store page fault | 根据请求来源，分别交付给 StoreUnits 或 AtomicsUnit 处理 |
| | 产生 store guest page fault | 根据请求来源，分别交付给 StoreUnits 或 AtomicsUnit 处理 |
| | 产生 load access fault | 交付给 LoadUnits 进行处理 |
| | 产生 store access fault | 根据请求来源，分别交付给 StoreUnits 或 AtomicsUnit 处理 |
| L2 TLB | | |
| | 产生 guest page fault | 交付给 L1 TLB，L1 TLB 根据请求来源交付处理 |
| | 产生 page fault | 交付给 L1 TLB，L1 TLB 根据请求来源交付处理 |
| | 产生 access fault | 交付给 L1 TLB，L1 TLB 根据请求来源交付处理 |
| | ecc 校验出错 | 无效掉当前项，返回 miss 结果并重新进行 Page Walk |

另外，根据 RISC-V 手册，Page Fault 的优先级高于 Access Fault，但是如果 Page Table Walk 过程中，出现了 PMP 检查或 PMA 检查的 Access Fault，此时页表项为非法，会发生 Page Fault 和 Access Fault 一起出现的特殊情况，昆明湖 V2R2 选择报 Access Fault。其余情况下均满足 Page Fault 的优先级高于 Access Fault。

## 7.5 内存访问顺序

在不同的场景下，昆明湖 V2R2 对地址空间的访问过程不同，简要归纳如下：

场景 1：CPU 不进行 VA-PA 转换

- CPU 要访问 PA；
- 查找地址属性；

- PMP 检查，确认地址读写执行权限；
- 执行地址访问。

场景 2：非虚拟化环境下进行 VA-PA 转换

- CPU 要访问 VA；
- 通过 MMU 进行地址翻译，得到页表项；
- 根据 PTE 查找地址、属性与权限信息；
- PMP 检查，确认地址读写执行权限；
- 执行地址访问。

场景 3：虚拟化环境下进行 VA-PA 转换

- CPU 要访问客户机 VA；
- 查看虚拟化寄存器确认两阶段地址翻译是否开启；
- MMU 进行两阶段地址翻译，得到页表项；
- 根据 PTE 查找地址、属性与权限信息；
- PMP 检查，确认地址读写执行权限；
- 执行地址访问。

# 8 内存子系统

## 8.1 L1 指令 Cache

### 8.1.1 概述

L1 指令高速缓存的主要特征如下：

- 指令高速缓存大小可配置，支持 16K/32K/64K
- 4 路组相联，缓存行大小为 64B
- 支持刷新
- 采用 PLRU 替换算法
- 支持提供两个连续的缓存行
- 支持指令预取
- 采用可配置的 MSHR 管理取指与预取请求
- 支持路查找
- 支持 ECC 校验
- 支持检查地址翻译错误、物理内存保护错误

### 8.1.2 路查找队列

昆明湖 V2R2 为提高缓存的命中率与访问速度，实现了高速指令缓存的路查找队列（WayLookUp）。路查找队列为先进先出结构，暂存缓存命中信息和指令地址翻译缓存（ITLB）返回的结果，同时监听 MSHR 写入 SRAM 的缓存行，对命中信息进行更新。用户可以通过配置 `nWayLookupSize` 来改变路查找队列的深度，同时可以反压限制预取最大距离。

### 8.1.3 MSHR 管理取指与预取请求

MSHR（Miss Status Holding Register）是一种缓存管理结构，用于跟踪和处理缓存缺失（cache miss）事件。当处理器请求的数据在缓存中未找到时，MSHR 会记录缺失的状态并管理随后的数据请求。昆明湖 V2R2 中，取指请求和预取请求均通过 MSHR 进行管理，所有 MSHR 公用一组数据寄存器以减少面积。用户可配置 MSRH 取指数量寄存器 `nFetchMshr` 与 MSHR 预取数量寄存器 `nPrefetchMshr`，来分别改变取指与预取的 MSHR 寄存器数量。

### 8.1.4 ECC 校验

昆明湖 V2R2 支持对指令高速缓存进行 ECC 校验。用户可通过配置 `DataCodeUnit` 来改变校验单元大小，单位为 bit 。每 64 位数据对应 1 位校验位。

### 8.1.5 分支预测单元

分支预测单元（BPU）内的预测器包括分支目标缓冲（uFTB）、取指目标缓冲（FTB）、条件分支预测器（TAGE-SC）、间接分支预测器（ITTAGE）和返回地址预测器（RAS）。

- uFTB 负责快速预测条件分支是否跳转
- FTB 负责维护预测块的起始地址，终止地址，所含分支指令 PC 地址、分支指令类型、基础的方向结果
- TAGE-SC 是条件分支指令的主预测器
- ITTAGE 负责预测间接跳转指令
- RAS 负责预测 return 类型的间接跳转指令跳转地址

### 8.1.6 分支目标缓冲

为了加快连续跳转时取指单元的取指效率，昆明湖 V2R2 在分支预测单元的第一级用分支目标缓冲（uFTB）为处理器作出无空泡的基础预测，其主要功能如下：

- 缓存 FTB 项，生成一周期预测结果。uFTB 中维护了一个小型的 FTB 项缓存，拿到 PC 之后，会在一周期之内读出 PC 所对应的 FTB 项，并从 FTB 项生成一阶段预测结果。
- 维护两比特饱和计数器，提供基础条件分支结果。uFTB 中为 FTB 项缓存的每一行都维护了对应的两比特饱和计数器，其方向预测结果会反映在 uFTB 的预测结果输出中。
- 根据更新请求更新 FTB 缓存和两比特饱和计数器

uFTB 进行预测的分支指令包括：

- BEQ、BNE、BLT、BLTU、BGE、BGEU、C.BEQZ、C.BNEZ
- JAL、JALR、C.J、C.JAL、C.JR、C.JALR

### 8.1.7 条件分支预测器

昆明湖 V2R2 使用 TAGE-SC 作为条件分支的主预测器。TAGE-SC 可以看作两个功能相对独立的组件：预测部分 TAGE 和校验部分 SC。

- 标记几何历史长度预测器 TAGE（Tagged Geometric History Length Predictor）利用历史长度不同的多个预测表，可以挖掘极长的分支历史信息。TAGE 功能是预测一个跳转指令是否跳转，它由一个基预测表和多个历史表组成，首先通过多个历史表进行分支预测，如果没有预测结果，则再采用基础预测表的预测结果。
- SC (Statistical Corrector) 是统计校正器。当 SC 会参考 TAGE 的预测结果和统计偏向的结果。并根据这两个结果，矫正最终的预测结果。

在昆明湖 V2R2 中，由于每个预测块最多可以有 2 条跳转指令，因此 TAGE 在每次预测最多同时预测 2 条条件分支指令。在访问 TAGE 的各个历史表时，用预测块的起始地址作为 PC，同时取出两个预测结果，并基于相同的全局历史进行预测。

TAGE-SC 进行预测的分支指令包括：

- BEQ、BNE、BLT、BLTU、BGE、BGEU、C.BEQZ、C.BNEZ

### 8.1.8 取指目标缓冲

取指目标缓冲（FTB）暂存 FTB 项，为后续的 TAGE、ITTAGE、SC、RAS 等高级预测器提供更为精确的分支指令位置、类型、目标地址等分支预测块关键信息，也为总是跳转的分支指令提供基础的方向预测。FTB 模块内有一 FTBBank 模块负责 FTB 项的实际存储，模块内使用了一块多路 SRAM 作为存储器，采用 4 路组相联进行存储，共 2048 项，每项最多存储 2 条分支，最多可存储 4096 条分支。

FTB 进行预测的分支指令包括：

- BEQ、BNE、BLT、BLTU、BGE、BGEU、C.BEQZ、C.BNEZ
- JAL、JALR、C.J、C.JAL、C.JR、C.JALR

### 8.1.9 间接分支预测器

昆明湖 V2R2 使用 ITTAGE 负责对间接分支的目标地址进行预测。ITTAGE 的每个表项在 TAGE 表项的基础上加入了所预测的跳转地址项，最后输出结果为选出的命中预测跳转地址而非选出的跳转方向。由于每个 FTB 项仅存储至多一条间接跳转指令信息，ITTAGE 预测器每周期也最多预测一条间接跳转指令的目标地址。

ITTAGE 进行预测的分支指令包括：

- JALR：源寄存器为 X1、X5 除外
- C.JALR：源寄存器为 X5 除外
- C.JR：源寄存器为 X1、X5 除外

### 8.1.10 返回地址预测器

返回地址预测器（RAS）用于函数调用结束时，返回地址的快速准确预测。当预测块被 RAS 的前级预测器预测为函数调用指令时，RAS 将该函数调用指令的后续指令地址入栈；当预测块被 RAS 的前级预测器预测为函数返回指令时，则从 RAS 弹栈。为了最大限度地减少执行过程中错误猜测导致的数据污染，昆明湖 V2R2 使用了基于持久化栈的返回地址预测器。RAS 分为提交栈和推测栈两个部分，推测栈利用了分支预测单元的预测结果来进行预测，并根据后端返回的信息将数据压入提交栈中。

- 函数调用指令包括：JAL、JALR、C.JALR
- 函数返回指令包括：JALR、C.JR、C.JALR

## 8.2 L1 数据 Cache

### 8.2.1 概述

L1 数据高速缓存的主要特征如下：

- 数据高速缓存大小为 64KB
- 8 路组相联，缓存行大小为 64B
- 虚拟地址索引，物理地址标记（VIPT）
- 支持至多 3 个并行的 64 / 128 bits 读操作
- 支持至多 1 个 512 bits 读操作
- 支持至多 1 个 512 bits 写操作
- 写策略采用写回-写分配模式

- 支持向 L2 Cache 请求缺失数据并重填
- 支持 Probe 请求的处理及替换数据块写回
- 支持原子请求的处理
- 支持 Random、LRU、PLRU 替换算法，默认 PLRU 算法
- 支持与 L2 Cache 配合处理缓存别名问题
- 支持 SMS、Stride 和 Stream 硬件预取

### 8.2.2 L1 DCache 一致性

L1 数据高速缓存采用 TileLink 协议维护多个处理器核心数据高速缓存的一致性，该协议要求在响应内存访问操作时进行提权和降权操作，同时定义了缓存行在数据高速缓存上的 4 个状态，分别是：

- Nothing：当前没有缓存数据副本的节点，没有读写权限
- Trunk：具有写权限的干净数据副本
- Dirty：具有写权限的脏数据副本
- Branch：具有只读数据缓存副本

### 8.2.3 独占式访问

L1 数据高速缓存支持 LR/SC 指令和 AMO 指令。用户可以使用 LR/SC 指令构成原子锁等同步原语实现同一个核不同进程之间或者不同核之间的同步，也可以使用 AMO 指令直接进行一些简单的原子运算操作。

MemBlock 中设置了 s_normal 和 s_atomics 两种状态，在执行原子指令时，状态被设置为 s_atomics，此时执行流进入原子指令处理单元，开始独占式访问。对于 LR/SC 指令对，L1 数据高速缓存会在执行 LR 指令时注册 reservation set，并独占该 reservation set 一定周期（默认 64 个时钟周期），独占期间 DCache 会阻塞其他核心的访问，防止多核场景下 LR/SC 指令循环陷入死锁；对于 AMO 指令，L1 数据高速缓存会判断是否其是否命中缓存，决定该条 AMO 请求是否需要进入 MissQueue。如果当前的 AMO 请求未命中缓存，则将这次请求信息写入 MissQueue；若本次 AMO 请求命中，则读取 data 的结果，然后执行 AMO 指令的运算操作，最后向原子指令处理单元返回响应。MemBlock 中发现原子指令处理单元完成执行，则将状态修改为 s_normal 继续执行。

### 8.2.4 替换与写回

L1 数据高速缓存采用写回（write-back）和写分配（write-allocate）的写策略，采用可配置 Random、LRU、PLRU 替换策略，默认选择使用 PLRU 策略；选出替换块后将其放入写回队列中，向 L2 Cache 发出 Release 请求。

## 8.3 L2 Cache

### 8.3.1 概述

L2 高速缓存的主要特征如下：

- 高速缓存大小为 1MB
- 8 路组相联，缓存行大小为 64B
- L2 Cache 与 L1 DCache 是严格包含关系，与 L1 ICache、PTW 是非严格包含关系

- 物理地址索引，物理地址标记（PIPT）
- 每次访问最大宽度为 64B
- 写策略采用写回-写分配模式
- 采用 DRRIP 替换算法
- 支持指令预取、TLB 预取、数据预取机制
- 支持 Best-Offset Prefetch（BOP）
- 采用非阻塞流水线结构

### 8.3.2 Cache 一致性

昆明湖 V2R2 二级高速缓存采用类 MESI 协议维护多个处理器核心数据高速缓存的一致性，包括如下 5 个 cache 状态：

- Invalid：表示缓存行不在该数据高速缓存中
- BRANCH：表示缓存行可能位于多个数据高速缓存中，该拷贝有可读权限
- TRUNK Clean：表示缓存行仅位于此核心的数据高速缓存中，该拷贝是干净块且没有读写权限，但是上游的数据缓存有读和写权限
- TRUNK Dirty：表示缓存行仅位于此核心的数据高速缓存中，该拷贝是脏块且没有读写权限，但是上游的数据缓存有读和写权限
- TIP Clean：表示缓存行仅位于此核心的数据高速缓存中，该拷贝是干净块且有读写权限，上游的数据或者有读权限、或者没有任何权限
- TIP Dirty：表示缓存行仅位于此核心的数据高速缓存中，该拷贝是脏块且有读写权限，上游的数据或者有读权限、或者没有任何权限

其他可能影响总线事务的标志位：

- `clients`：标志 L1 DCache 有无数据拷贝，用于辅助判断在 BRANCH 或 TIP 状态下 L1 DCache 有无可读权限的拷贝
- `alias`：别名位，用于硬件处理 L1 DCache（VIPT）的别名问题，只在 L1 DCache 有数据拷贝时有效，记录当前物理地址在 L1 DCache 中的虚拟地址索引超出物理页偏移的部分
- `prefetch`：标志该数据拷贝是否是预取块

### 8.3.3 组织形式

昆明湖 V2R2 L2 Cache 采用了分块（Slice）的流水线结构，默认为 4 个 Slice，将访问地址按照 PA[7:6] 离散在 4 个不同的 Slice 中，允许多个访问的并行处理，从而调高访问效率。

# 9 向量

## 9.1 版本支持

兼容 RISC-V “V” Vector Extension, Version 1.0 。

## 9.2 向量编程模型

向量扩展支持以下特性：

- 拥有 32 个独立的向量架构寄存器 v0-v31。向量寄存器的位宽为 128 位。
- 对于向量浮点指令，支持的元素类型为 FP32、FP64（即 SEW = 16/32）。
- 对于向量整型指令，支持的元素类型为 INT8、INT16、INT32 和 INT64（即 SEW = 8/16/32/64）。
- 支持向量寄存器的分组，以提高向量运算的效率。支持 4 种分组方式：每组包含 1/2/4/8 个向量寄存器，分成 32/16/8/4 个组。

## 9.3 向量控制寄存器

有 7 个非特权 CSR：

- vstart

  向量起始位置寄存器指定了执行向量指令时起始元素位置。每条向量指令执行后 vstart 都会被清零。在大多数情况下，软件不需要改动 vstart。只有向量存储指令支持非 0 的 vstart；所有的向量运算指令都要求 vstart = 0，否则会产生非法指令异常。

- vxsat

  定点溢出标志位寄存器，只有 bit0 有效，表示是否有定点指令产生溢出结果。

- vxrm

  定点舍入模式寄存器，支持 4 种舍入模式：向大数舍入、向偶数舍入、向零舍入和向奇数舍入。

- vcsr

  向量控制状态寄存器。

- vl

  向量长度寄存器指定了向量指令更新目的寄存器的元素范围。一般情况下，向量指令更新目的寄存器中元素序号小于 vl 的元素，元素序号大于等于 vl 的元素根据 vta 的值写全 1 或者保留原值。

- vtype

  向量数据类型寄存器，设定向量计算的基本数据属性，包括：vill、vsew、vlmul、vta 和 vma。

- vlenb

  向量位宽寄存器, 以字节为单位表示向量位宽。

- 此外还支持向量状态维护功能，在 mstatus[10:9] 处定义了 VS 位，可以用于判断上下文切换时，是否需要保存向量相关的寄存器。

## 9.4 向量相关异常

向量指令可以分为三大类:

- 向量运算
- 向量 load
- 向量 store

向量运算不会触发异常。
向量 Load 与向量 Store 统称为向量访存。
向量访存可以引起手册规定的异常。
向量 Load 会引发:

- 3 BreakPoint
- 4 Load address misaligned
- 5 Load access fault
- 13 Load page fault
- 21 Load guest page fault

向量 Store 会引发:

- 3 BreakPoint
- 6 Store address misaligned
- 7 Store access fault
- 15 Store page fault
- 23 Store guest page fault

实现中，向量访存不允许访存 MMIO，对于 MMIO 的向量访存会引发 Load/Store access fault 异常。
在向量访存中触发异常，根据手册规定实现如下:

1. 非 fault-only-first 指令，将会设置 vstart 寄存器为触发异常的元素位置。触发异常的元素保留原值，异常元素之后的元素会根据 Tail 与 Mask 的配置来进行处理。
2. fault-only-first 指令，若异常的元素为第一个元素，则会触发异常。否则将会设置 vl 寄存器为异常的元素位置，且不触发异常。触发异常的元素保留原值，异常元素之后的元素会根据 Tail 与 Mask 的配置来进行处理。
3. segment 指令以 segment 为单位引发异常，在 segment 中引发异常之后，将会设置 vstart 寄存器为触发异常的 segment 位置。segment 中触发异常的元素之前的元素将会正常的被访问执行。

# 10 中断控制器

昆明湖 V2R2 中，中断控制器包括 IMSIC 外部中断控制器，CLINT 本地中断控制器，下面进行详细说明。

## 10.1 CLINT 中断控制器

### 10.1.1 概要

CLINT 为 HART 提供 M 特权级下的软件中断，以及 M 特权级下的 time 定时中断，以及 64bit time 计时器。

### 10.1.2 寄存器映射

表 10.1: CLINT 的寄存器排布

| offset address | Width | attribute | Description |
|---|---|---|---|
| 0x0000_0000 | 4B | RW | HART M 软中断配置寄存器 |
| 0x0000_0004 | | | |
| ... | | | Reserved |
| 0x0000_3FFF | | | |
| 0x0000_4000 | 8B | RW | MTIMECMP 寄存器 |
| 0x0000_4008 | | | |
| ... | | | Reserved |
| 0x0000_BFF7 | | | |
| 0x0000_BFF8 | 8B | RW | MTIME 寄存器 |

## 10.2 IMSIC 中断控制器

### 10.2.1 概要

IMSIC 作为 RISCV 的外部中断控制器之一，负责 MSI 中断的接收与传递，涵盖 M, S, VS 特权级下的中断上报。每种特权级下的中断配置通过 IMSIC interrupt file MMIO 空间实现，默认支持 interrupt file 数目 7 个：M, S,5 个 VS interrupt file. 默认支持有效中断号：1-255.

### 10.2.2 寄存器映射

DEVICE 通过发送中断 ID 到 IMSIC 内部 interrupt file MMIO 空间，从而实现 MSI 的发送。RISCV AIA SPEC 明确规定，多 interrupt files 场景下, Supervisor-level 只能访问 all Supervisor-level and guest interrupt files, 不能访问 Machine-level interrupt files. 因此, 在地址排布上, 所有的 Machine-level interrupt file 集中连续分配, Supervisor-level and guest interrupt files 集中连续分配, 这样仅用一个 PMP table entry 就能保证 Supervisor-level 没有 S/VS interrupt files 以外的访问权限。硬件实现上，M, S/VS interrupt file 寄存器空间，拥有各自的基地址。如下对两个特权级下的中断寄存器访问进行说明。

表 10.2: M interrupt file

| 寄存器 | 地址偏移 | 位宽 | 属性 | 复位值 | 描述 |
|---|---|---|---|---|---|
| setipnum_le | 0x0000 | 32 | WO | 32'h0 | interrupt file 访问寄存器。写入数据为 MSI 中断 ID，读取值为 0. 默认支持最高 8bit 中断 ID 写入。 |

表 10.3: S/VS interrupt file

| 寄存器 | 地址偏移 | 位宽 | 属性 | 复位值 | 描述 |
|---|---|---|---|---|---|
| setipnum_le | 0x0000 | 32 | WO | 32'h0 | interrupt file 访问寄存器。写入数据为 MSI 中断 ID，读取值为 0. 默认支持最高 8bit 中断 ID 写入。 |
| setipnum_le_s | 0x0000 | 32 | WO | 32'h0 | interrupt file 访问寄存器。写入数据为 MSI 中断 ID，读取值为 0. 默认支持最高 8bit 中断 ID 写入，MSI ID 超过 8bit 访问，硬件自动截断低 8bit. |
| setipnum_le_vs1 | 0x1000 | 32 | WO | 32'h0 | VS 1 interrupt file 访问寄存器。写入数据为 MSI 中断 ID，读取值为 0. 默认支持最高 8bit 中断 ID 写入，MSI ID 超过 8bit 访问，硬件自动截断低 8bit. |
| setipnum_le_vs2 | 0x2000 | 32 | WO | 32'h0 | VS 2 interrupt file 访问寄存器。写入数据为 MSI 中断 ID，读取值为 0. 默认支持最高 8bit 中断 ID 写入，MSI ID 超过 8bit 访问，硬件自动截断低 8bit. |
| setipnum_le_vs3 | 0x3000 | 32 | WO | 32'h0 | VS 3 interrupt file 访问寄存器。写入数据为 MSI 中断 ID，读取值为 0. 默认支持最高 8bit 中断 ID 写入，MSI ID 超过 8bit 访问，硬件自动截断低 8bit. |
| setipnum_le_vs4 | 0x4000 | 32 | WO | 32'h0 | VS 4 interrupt file 访问寄存器。写入数据为 MSI 中断 ID，读取值为 0. 默认支持最高 8bit 中断 ID 写入，MSI ID 超过 8bit 访问，硬件自动截断低 8bit. |

| 寄存器 | 地址偏移 | 位宽 | 属性 | 复位值 | 描述 |
| --- | --- | --- | --- | --- | --- |
| setipnum_le_vs5 | 0x5000 | 32 | WO | 32'h0 | VS 5 interrupt file 访问寄存器。写入数据为 MSI 中断 ID，读取值为 0. 默认支持最高 8bit 中断 ID 写入，MSI ID 超过 8bit 访问，硬件自动截断低 8bit. |

## 10.3 核间中断

多核间通信可以通过核间中断来完成，核间中断有两种方式可以实现。

- 通过配置 CLINT 软件中断，可实现 M 特权级中断上报。
- 通过配置 IMSIC interrupt file，可实现 M, S, VS 特权级下的中断上报。

# 11 总线接口

昆明湖 V2R2 的总线接口具有 256 位宽度，支持 CHI Issue B 或 Issue E.b 的子集。关于该协议的详细内容，请参考 AMBA® CHI Architecture Specification。

## 11.1 支持的响应类型

CHI 协议的 RespErr 可以表示响应正常或是错误。昆明湖 V2R2 支持的响应类型如下。

| RespErr 的值 | 响应类型 |
|---|---|
| 0b00 | Normal Okay |
| 0b01 | Exclusive Okay |
| 0x11 | Non-data Error，即 NDERR |

由于昆明湖 V2R2 不具有数据校验码，因此不支持 DERR。

## 11.2 不同总线响应下的行为

- Normal Okay：普通传输访问成功，或 exclusive 传输访问失败；读传输 exclusive 访问失败代表总线不支持 exclusive 传输，产生访问错误异常，写传输 exclusive 访问失败仅代表抢锁失败，不会返回异常。
- Exclusive Okay：exclusive 访问成功。
- NDERR：访问出错，读传输产生访问错误异常，写传输忽略此错误。

## 11.3 接口信号

CHI 使用不同的通道传输不同的消息，传输的消息包括：

- 数据（DAT）
- 请求（REQ）
- 响应（RSP）
- 监听（SNP）

以 TX 字母为前缀的通道用于发送消息，以 RX 字母为前缀的通道用于接收消息。昆明湖 V2R2 一共有 6 个通道：

- RXDAT

- RXRSP
- RXSNP
- TXDAT
- TXREQ
- TXRSP

后文将会列出这些通道包含的信号。除了这些通道外，总线接口还包含以下的信号。

| 信号名 | I/O | 功能描述 |
|---|---|---|
| chi_rx_linkactiveack | O | 决定 RX 的状态。 |
| chi_rx_linkactivereq | I | 决定 RX 的状态。 |
| chi_tx_linkactiveact | I | 决定 TX 的状态。 |
| chi_tx_linkactivereq | O | 决定 TX 的状态。 |
| chi_rxsactive | I | 表示 RX 有正在进行的事务。 |
| chi_txsactive | O | 表示 TX 有正在进行的事务。 |

RX 的 linkactiveack 和 linkactivereq 决定了 RX 的状态；TX 的 linkactiveack 和 linkactivereq 决定了 TX 的状态。

| 状态 | linkactivatereq | linkactivateack |
|---|---|---|
| STOP | 0 | 0 |
| ACTIVATE | 1 | 0 |
| RUN | 1 | 1 |
| DEACTIVATE | 0 | 1 |

### 11.3.1 通道信号

表 11.4: RXDAT 通道信号

| 信号名 | I/O | 功能描述 |
|---|---|---|
| chi_rx_dat_flitv | I | flit 的有效信号，高电平表示 flit 有效 |
| chi_rx_dat_lcrdv | O | L-Credit 有效信号 |
| chi_rx_dat_flit | I | RXDAT 通道的 flit |
| chi_rx_dat_flitpend | I | flit 的 pending 信号，表示接下来会传输一个 flit |

表 11.5: RXRSP 通道信号

| 信号名 | I/O | 功能描述 |
|---|---|---|
| chi_rx_rsp_flitv | I | flit 的有效信号，高电平表示 flit 有效 |
| chi_rx_rsp_lcrdv | O | L-Credit 的有效信号 |
| chi_rx_rsp_flit | I | RXRSP 通道的 flit |

| 信号名 | I/O | 功能描述 |
|---|---|---|
| chi_rx_rsp_flitpend | I | flit 的 pending 信号，表示接下来会传输一个 flit |

表 11.6: RXSNP 通道信号

| 信号名 | I/O | 功能描述 |
|---|---|---|
| chi_rx_snp_flitv | I | flit 的有效信号，高电平表示 flit 有效 |
| chi_rx_snp_lcrdv | O | L-Credit 的有效信号 |
| chi_rx_snp_flit | I | RXSNP 通道的 flit |
| chi_rx_snp_flitpend | I | flit 的 pending 信号，表示接下来会传输一个 flit |

表 11.7: TXDAT 通道信号

| 信号名 | I/O | 功能描述 |
|---|---|---|
| chi_tx_dat_flitv | O | flit 的有效信号，高电平表示 flit 有效 |
| chi_tx_dat_lcrdv | I | L-Credit 的有效信号 |
| chi_tx_dat_flit | O | TXDAT 通道的 flit |
| chi_tx_dat_flitpend | O | flit 的 pending 信号，表示接下来会传输一个 flit |

表 11.8: TXREQ 通道信号

| 信号名 | I/O | 功能描述 |
|---|---|---|
| chi_tx_req_flitv | O | flit 的有效信号，高电平表示 flit 有效 |
| chi_tx_req_lcrdv | I | L-Credit 的有效信号 |
| chi_tx_req_flit | O | TXREQ 通道的 flit |
| chi_tx_req_flitpend | O | flit 的 pending 信号，表示接下来会传输一个 flit |

表 11.9: TXRSP 通道信号

| 信号名 | I/O | 功能描述 |
|---|---|---|
| chi_tx_rsp_flitv | O | flit 的有效信号，高电平表示 flit 有效 |
| chi_tx_rsp_lcrdv | I | L-Credit 的有效信号 |
| chi_tx_rsp_flit | O | TXRSP 通道的 flit |
| chi_tx_rsp_flitpend | O | flit 的 pending 信号，表示接下来会传输一个 flit |

### 11.3.2 flit 格式

位宽为空，代表此信号与上一行的信号是共用的。信号名后面标注 *，代表此信号仅适用于 CHI Issue E.b。位宽后面标注 *，代表此信号在 CHI Issue B 和 Issue E.b 中具有不同的位宽，两个位宽中标注 * 的是 E.b 中的位宽。

表 11.10: Data flit

| 信号名 | 位宽 | 功能描述 |
|---|---|---|
| QoS | 4 | Quality of Service，数值越大优先级越高。 |
| TgtID | id_width | 目标 ID。 |
| SrcID | id_width | 来源 ID。 |
| TxnID | 8/12* | 事务 ID。 |
| HomeNID | id_width | Home 节点 ID，请求者在发送 CompAck 时把这个 ID 作为 TgtID。 |
| Opcode | 3/4* | 操作码。 |
| RespErr | 2 | 相应错误码。 |
| Resp | 3 | 响应状态。 |
| DataSource | 3/4* | 数据来源。 |
| {1'b0，FwdState[2:0]}* | | 指示从监听者发送到请求者的 CompData 中的状态。 |
| {1'b0, DataPull[2:0]}* | | |
| CBusy | 3 | 完成者的繁忙程度，其编码由具体的实现决定。 |
| DBID | 8/12* | 数据缓冲区 ID，用于请求方的 TxnID。 |
| CCID | 2 | 关键数据块的 ID。 |
| DataID | 2 | 正在被传输的数据块的 ID。0b00 表示 [255:0]，0b10 表示 [511:256]。 |
| TagOp | 2 | 表示要对 Tag 执行的操作。 |
| Tag | 8 | n 组 4 位 tag，每个 tag 绑定对应顺序的 16B 数据，地址对齐。 |
| TU | 2 | 指示要更新的 tag。 |
| TraceTag | 1 | 标记，用于跟踪。 |
| RSVDC | 4 | 保留给用户使用，其含义由具体的实现决定。 |
| BE | 32 | 字节使能。表示每个字节是否有效。 |
| Data | 256 | 数据。 |

表 11.11: Request flit

| 信号名 | 位宽 | 功能描述 |
|---|---|---|
| QoS | 4 | Quality of Service，数值越大优先级越高。 |
| TgtID | id_width | 目标 ID。 |
| SrcID | id_width | 来源 ID。 |
| TxnID | 8/12* | 事务 ID。 |
| ReturnNID | id_width | 需要回复的节点 ID。 |
| StashNID | | Stash 请求的目标 ID。 |
| {4'b0, SLCRepHint[6:0]}* | | SLCRepHint：SLC 替换提示。 |
| StashNIDValid | 1 | 用于 Stash 事务，表示 StashNID 是否有效。 |
| Endian | | 用于原子事务，0 表示小端，1 表示大端。 |
| Deep | | 在响应之前是否必须先写入最终目的地。 |

| 信号名 | 位宽 | 功能描述 |
| --- | --- | --- |
| ReturnTxnID | 8/12* | 用于 DMT。 |
| {6'b0，StashLPIDValid，StashLPID[4:0]}* | | StashLPIDValid：用于 stash 事务，StashLPID 的有效信号。StashLPID：用于 Stash 事务。 |
| Opcode | 6/7* | 操作码。 |
| Size | 3 | 数据大小。 |
| Addr | RAW | 地址。 |
| NS | 1 | 用于指示物理地址空间。 |
| LikelyShared | 1 | 表示请求的数据是否可能与另一个请求节点共享。 |
| AllowRetry | 1 | 是否允许重试。 |
| Order | 2 | 用于指定事务的顺序要求。 |
| PCrdType | 4 | Credit 类型。 |
| MemAttr | 4 | 本次事务的属性。 |
| SnpAttr | 1 | Snoop 属性。 |
| DoDWT | | 执行 DWT, 影响 DBIDResp 的 TgtID 和 TxnID 取值。 |
| PGroupID | 5/8* | 用于 PCMO 事务。 |
| StashGroupID | | 用于 StashOnceSep 事务。 |
| TagGroupID | | 用于标记。 |
| {3'b0，LPID[4:0]}* | | 逻辑处理器 ID，用于一个请求者包含多个逻辑处理器的情况。 |
| Excl | 1 | 用于 exclusive 事务。 |
| SnoopMe | | 用于原子事务，指定是否必须向请求者发送 Snoop。 |
| ExpCompAck | 1 | 表示事务是否包含了一个 CompAck 响应。 |
| TagOp | 2 | 表示要对 Tag 执行的操作。 |
| TraceTag | 1 | 标记，用于跟踪。 |
| RSVDC | 4 | 保留给用户使用，其含义由具体的实现决定。可以是任何值。 |

表 11.12: Response flit

| 信号名 | 位宽 | 功能描述 |
| --- | --- | --- |
| QoS | 4 | Quality of Service，数值越大优先级越高。 |
| TgtID | id_width | 目标 ID。 |
| SrcID | id_width | 来源 ID。 |
| TxnID | 8/12* | 事务 ID。 |
| Opcode | 4/5* | 操作码。 |
| RespErr | 2 | 响应错误码。 |
| Resp | 3 | 响应状态。 |
| FwdState | 3 | 用于 DCT，指示从监听者发送到请求者的 CompData 中的状态。 |
| {2'b0，DataPull} | | 用于 Stash 事务，指示 Snoop 响应是否需要 Data Pull。 |

| 信号名 | 位宽 | 功能描述 |
|---|---|---|
| CBusy | 3 | 完成者的繁忙程度，其编码由具体的实现决定。 |
| DBID | 8/12* | 数据缓冲区 ID，用于请求方的 TxnID。 |
| {4'b0，PGroupID[7:0]} | | 用于 Persistent CMO 事务。 |
| {4'b0，StashGroupID[7:0]} | | 用于 Stash 事务。 |
| {4'b0，TagGroupID[7:0]} | | 用于标记。 |
| PCrdType | 4 | Credit 类型。 |
| TagOp | 2 | 表示要对 Tag 执行的操作。 |
| TraceTag | 1 | 标记，用于跟踪。 |

表 11.13: Snoop flit

| 信号名 | 位宽 | 功能描述 |
|---|---|---|
| QoS | 4 | Quality of Service，数值越大优先级越高。 |
| SrcID | id_width | 来源 ID。 |
| TxnID | 8/12* | 事务 ID。 |
| FwdNID | id_width | 指示 CompData 响应可以转发到哪个请求者。 |
| FwdTxnID | 8/12* | 用于 DCT。 |
| {6'b0，<br>StashLPIDValid[0:0]，<br>StashLPID[4:0]}* | | StashLPIDValid：用于 Stash 事务，StashLPID 的有效位。<br>StashLPID：用于 Stash 事务。 |
| {4'b0，VMIDExt[7:0]}* | | VMIDExt：用于 DVM 事务。 |
| Opcode | 5 | 操作码。 |
| Addr | SAW | 地址。 |
| NS | 1 | 用于指示物理地址空间。 |
| DoNotGoToSD | 1 | 指示是否要求 Snoopee 不转换到 SD 状态。 |
| RetToSrc | 1 | 该字段请求 Snoopee 将缓存行的副本返回给 Home。 |
| TraceTag | 1 | 标记，用于跟踪。 |

## 11.4 支持的 Coherency Transaction 类型

- SnpShared
- SnpClean
- SnpOnce
- SnpNotSharedDirty
- SnpUniqueStash
- SnpMakeInvalidStash
- SnpUnique
- SnpCleanShared
- SnpCleanInvalid
- SnpMakeInvalid

- SnpStashUnique
- SnpStashShared
- SnpSharedFwd
- SnpCleanFwd
- SnpOnceFwd
- SnpNotSharedDirtyFwd
- SnpUniqueFwd

# 12 调试

本章是昆明湖的调试模块的设计文档。昆明湖 debug 兼容 RISC-V Debug V0.13 手册标准。对外调试接口支持 JTAG。调试接口是软件与处理器交互的通道。用户可以通过调试接口获取 CPU 的状态，包括寄存器和存储器内容，以及其他片上设备的信息。支持程序下载。

## 12.1 Debug Module

如下图所示。昆明湖的 debug 工作是由调试软件（GDB）、调试代理服务程序 (openocd)、调试器 (debug module wrapper) 等组件一起配合完成的，其中调试器包括 JtagDTM，DMI，DM。调试接口在整个 CPU 调试环境中的位置如下图所示。其中，调试软件和调试代理服务程序通过网络互联，调试代理服务程序与调试器通过 Jtag 仿真器连接，调试器与 Jtag 仿真器的调试接口以 JTAG 模式通信。

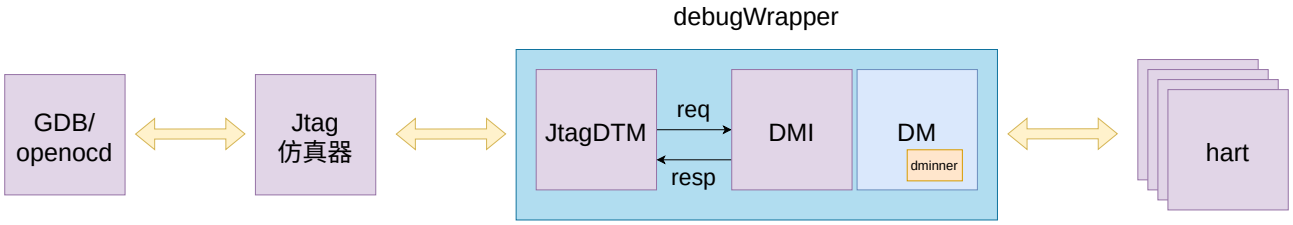


图 12.1: debug module

调试器与 hart 之间的连接以及时钟域关系如下图所示：
当前昆明湖 debug module 实现的情况如下：

- 支持从第一条指令开始的调试，在 cpu 复位之后进入调试模式。
- 支持单步调试。
- 支持软断点 (ebreak 指令)、硬断点（trigger）和内存断点（trigger）。
- 支持读写 CSR 和内存，支持 progbuf 和 sysbus 两种访存方式。

## 12.2 Trigger Module

当前昆明湖 trigger module 的实现情况如下：

- 昆明湖 trigger module 当前实现的 debug 相关的 CSR 如下表所示。
- trigger 的默认配置数量是 4 (支持用户自定义配置)。
- 支持 mcontrol6 类型的指令、访存的 trigger。
- match 支持相等，大于等于，小于三种类型（向量访存当前只支持相等类型匹配）。
- 仅支持 address 匹配，不支持 data 匹配。

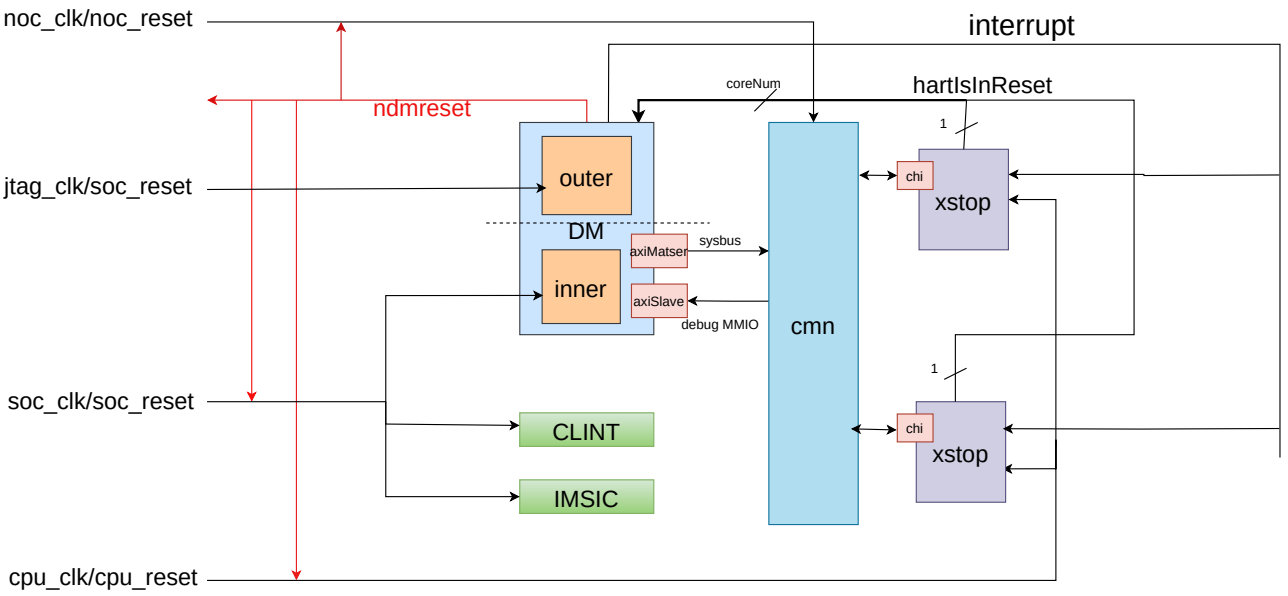


图 12.2: debug2harts

- 仅支持 timing = before。
- 支持一对 trigger 的 chain。
- 为了防止 trigger 的二次产生 breakpoint 的异常，支持通过 xSTATUS.xIE 控制。
- 支持 H 扩展的软硬件断点，watchpoint 调试手段。
- 支持原子指令的访存 trigger。

表 12.1: 昆明湖实现的 debug 相关的 csr

| 名称 | 地址 | 读写 | 介绍 | 复位值 |
|---|---|---|---|---|
| Tselect | 0x7A0 | RW | trigger 选择寄存器 | 0X0 |
| Tdata1(Mcontrol6) | 0x7A1 | RW | trigger data1 | 0xF0000000000000000 |
| Tdata2 | 0x7A2 | RW | trigger data2 | 0x0 |
| Tinfo | 0x7A4 | RO | trigger info | 0x40 |
| Dcsr | 0x7B0 | RW | Debug Control and Status | 0x40000003 |
| Dpc | 0x7B1 | RW | Debug PC | 0x0 |
| Dscratch0 | 0x7B2 | RW | Debug Scratch Register 0 | - |
| Dscratch1 | 0x7B3 | RW | Debug Scratch Register 1 | - |

# 13 性能监测单元

昆明湖 V2R2 性能监测单元（PMU）根据 RISC-V 特权手册实现了基本的硬件性能监测功能，并额外支持 sstc 以及 sscofpmf 拓展，用于统计处理器运行中的部分硬件信息和线程信息，供软件开发人员进行程序优化。

性能监测单元统计的软硬件信息主要分为以下几种：

- 硬件线程执行的时钟周期数 (cycle)
- 硬件线程已提交的指令数 (minstret)
- 硬件定时器 (time)
- 处理器关键部件性能事件统计 (hpmcounter3 - hpmcouonter31，countovf)

## 13.1 PMU 的编程模型

### 13.1.1 PMU 的基本用法

PMU 的基本用法如下：

- 通过 mcountinhibit 寄存器关闭所有性能事件监测。
- 初始化各个监测单元性能事件计数器，包括：mcycle, minstret, mhpmcounter3 - mhpmcounter31。
- 配置各个监测单元性能事件选择器，包括: mhpmcounter3 - mhpmcounter31。昆明湖 V2R2 对每个事件选择器可以配置最多四种事件组合，将事件索引值、事件组合方法、采样特权级写入事件选择器后，即可在规定的采样特权级下对配置的事件正常计数，并根据组合后结果累加到事件计数器中。
- 配置 xcounteren 进行访问权限授权
- 通过 mcountinhibit 寄存器开启所有性能事件监测，开始计数。

### 13.1.2 PMU 事件溢出中断

昆明湖 V2R2 性能监测单元发起的溢出中断 LCOFIP，统一中断向量号为 12，中断的使能以及处理过程与普通私有中断一致，详见异常与中断

## 13.2 PMU 相关的控制寄存器

### 13.2.1 机器模式性能事件计数禁止寄存器 (MCOUNTINHIBIT)

机器模式性能事件计数禁止寄存器 (mcountinhibit)，是 32 位 WARL 寄存器，主要用与控制硬件性能监测计数器是否计数。在不需要性能分析的场景下，可以关闭计数器，以降低处理器功耗。

表 13.1: 机器模式性能事件计数禁止寄存器说明

| 名称 | 位域 | 读写 | 行为 | 复位值 |
|---|---|---|---|---|
| HPMx | 31:4 | RW | mhpmcounterx 寄存器禁止计数位:<br>0: 正常计数<br>1: 禁止计数 | 0 |
| IR | 3 | RW | minstret 寄存器禁止计数位:<br>0: 正常计数<br>1: 禁止计数 | 0 |
| – | 2 | RO 0 | 保留位 | 0 |
| CY | 1 | RW | mcycle 寄存器禁止计数位:<br>0: 正常计数<br>1: 禁止计数 | 0 |

### 13.2.2 机器模式性能事件计数器访问授权寄存器 (MCOUNTEREN)

机器模式性能事件计数器访问授权寄存器 (mcounteren)，是 32 位 WARL 寄存器，主要用于控制用户态性能监测计数器在机器模式以下特权级模式 (HS-mode/VS-mode/HU-mode/VU-mode) 中的访问权限。

表 13.2: 机器模式性能事件计数器访问授权寄存器说明

| 名称 | 位域 | 读写 | 行为 | 复位值 |
|---|---|---|---|---|
| HPMx | 31:4 | RW | hpmcounterenx 寄存器 M-mode 以下访问权限位:<br>0: 访问 hpmcounterx 报非法指令异常<br>1: 允许正常访问 hpmcounterx | 0 |
| IR | 3 | RW | instret 寄存器 M-mode 以下访问权限位:<br>0: 访问 instret 报非法指令异常<br>1: 允许正常访问 | 0 |
| TM | 2 | RW | time/stimecmp 寄存器 M-mode 以下访问权限位:<br>0: 访问 time 报非法指令异常<br>1: 允许正常访问 | 0 |
| CY | 1 | RW | cycle 寄存器 M-mode 以下访问权限位:<br>0: 访问 cycle 报非法指令异常<br>1: 允许正常访问 | 0 |

### 13.2.3 监督模式性能事件计数器访问授权寄存器 (SCOUNTEREN)

监督模式性能事件计数器访问授权寄存器 (scounteren)，是 32 位 WARL 寄存器，主要用于控制用户态性能监测计数器在用户模式 (HU-mode/VU-mode) 中的访问权限。

表 13.3: 监督模式性能事件计数器访问授权寄存器说明

| 名称 | 位域 | 读写 | 行为 | 复位值 |
|---|---|---|---|---|
| HPMx | 31:4 | RW | hpmcounterenx 寄存器用户模式访问权限位:<br>0: 访问 hpmcounterx 报非法指令异常<br>1: 允许正常访问 hpmcounterx | 0 |
| IR | 3 | RW | instret 寄存器用户模式访问权限位:<br>0: 访问 instret 报非法指令异常<br>1: 允许正常访问 | 0 |
| TM | 2 | RW | time 寄存器用户模式访问权限位:<br>0: 访问 time 报非法指令异常<br>1: 允许正常访问 | 0 |
| CY | 1 | RW | cycle 寄存器用户模式访问权限位:<br>0: 访问 cycle 报非法指令异常<br>1: 允许正常访问 | 0 |

### 13.2.4 虚拟化模式性能事件计数器访问授权寄存器 (HCOUNTEREN)

虚拟化模式性能事件计数器访问授权寄存器 (hcounteren)，是 32 位 WARL 寄存器，主要用于控制用户态性能监测计数器在客户虚拟机 (VS-mode/VU-mode) 中的访问权限。

表 13.4: 监督模式性能事件计数器访问授权寄存器说明

| 名称 | 位域 | 读写 | 行为 | 复位值 |
|---|---|---|---|---|
| HPMx | 31:4 | RW | hpmcounterenx 寄存器客户虚拟机访问权限位:<br>0: 访问 hpmcounterx 报非法指令异常<br>1: 允许正常访问 hpmcounterx | 0 |
| IR | 3 | RW | instret 寄存器客户虚拟机访问权限位:<br>0: 访问 instret 报非法指令异常<br>1: 允许正常访问 | 0 |
| TM | 2 | RW | time/vstimecmp(via stimecmp) 寄存器客户虚拟机访问权限位:<br>0: 访问 time 报非法指令异常<br>1: 允许正常访问 | 0 |
| CY | 1 | RW | cycle 寄存器客户虚拟机访问权限位:<br>0: 访问 cycle 报非法指令异常<br>1: 允许正常访问 | 0 |

### 13.2.5 监督模式时间比较寄存器 (STIMECMP)

监督模式时间比较寄存器 (stimecmp)，是 64 位 WARL 寄存器，主要用于管理监督模式下的定时器中断 (STIP)。

STIMECMP 寄存器行为说明：

- 复位值为 64 位无符号数 64'hffff_ffff_ffff_ffff。
- 在 menvcfg.STCE 为 0 且当前特权级低于 M-mode (HS-mode/VS-mode/HU-mode/VU-mode) 时，访问 stimecmp 寄存器产生非法指令异常，且不产生 STIP 中断。
- stimecmp 寄存器是 STIP 中断产生源头：在进行无符号整数比较 time  stimecmp 时，拉高 STIP 中断等待信号。
- 监督模式软件可以通过写 stimecmp 控制定时器中断的产生。

### 13.2.6 客户虚拟机监督模式时间比较寄存器 (VSTIMECMP)

客户虚拟机监督模式时间比较寄存器 (vstimecmp)，是 64 位 WARL 寄存器，主要用于管理客户虚拟机监督模式下的定时器中断 (STIP)。

VSTIMECMP 寄存器行为说明：

- 复位值为 64 位无符号数 64'hffff_ffff_ffff_ffff。
- 在 henvcfg.STCE 为 0 或者 hcounteren.TM 时，通过 stimecmp 寄存器访问 vstimecmp 寄存器产生虚拟非法指令异常，且不产生 VSTIP 中断。
- vstimecmp 寄存器是 VSTIP 中断产生源头：在进行无符号整数比较 time + htimedelta  vstimecmp 时，拉高 VSTIP 中断等待信号。
- 客户虚拟机监督模式软件可以通过写 vstimecmp 控制 VS-mode 下定时器中断的产生。

## 13.3 PMU 相关的性能事件选择器

机器模式性能事件选择器 (mhpmevent3 - 31)，是 64 为 WARL 寄存器，用于选择每个性能事件计数器对应的性能事件。在昆明湖 V2R2 中，每个计数器可以配置最多四个性能事件进行组合计数。用户将事件索引值、事件组合方法、采样特权级写入指定事件选择器后，该事件选择器所匹配的事件计数器开始正常计数。

表 13.5: 机器模式性能事件选择器说明

| 名称 | 位域 | 读写 | 行为 | 复位值 |
|---|---|---|---|---|
| OF | 63 | RW | 性能计数上溢标志位:<br>0: 对应性能计数器溢出时置 1，产生溢出中断<br>1: 对应性能计数器溢出时值不变，不产生溢出中断 | 0 |
| MINH | 62 | RW | 置 1 时，禁止 M 模式采样计数 | 0 |
| SINH | 61 | RW | 置 1 时，禁止 S 模式采样计数 | 0 |
| UINH | 60 | RW | 置 1 时，禁止 U 模式采样计数 | 0 |
| VSINH | 59 | RW | 置 1 时，禁止 VS 模式采样计数 | 0 |
| VUINH | 58 | RW | 置 1 时，禁止 VU 模式采样计数 | 0 |
| – | 57:55 | RW | – | 0 |

| 名称 | 位域 | 读写 | 行为 | 复位值 |
|---|---|---|---|---|
| OP_TYPE2 | 54:50 | RW | 计数器事件组合方法控制位: | 0 |
| OP_TYPE1 | 49:45 | | 5'b00000: 采用 or 操作组合 | |
| OP_TYPE0 | 44:40 | | 5'b00001: 采用 and 操作组合 | |
| | | | 5'b00010: 采用 xor 操作组合 | |
| | | | 5'b00100: 采用 add 操作组合 | |
| EVENT3 | 39:30 | RW | 计数器性能事件索引值: | – |
| EVENT2 | 29:20 | | 0: 对应的事件计数器不计数 | |
| EVENT1 | 19:10 9:0 | | 1: 对应的事件计数器对事件计数 | |
| EVENT0 | | | | |

其中，计数器事件的组合方法为：

- EVENT0 和 EVENT1 事件计数采用 OP_TYPE0 操作组合为 RESULT0。
- EVENT2 和 EVENT3 事件计数采用 OP_TYPE1 操作组合为 RESULT1。
- RESULT0 和 RESULT1 组合记过采用 OP_TYPE2 操作组合为 RESULT2。
- RESULT2 累加到对应事件计数器。

对性能事件选择器中事件索引值部分复位值规定如下：

- 由于目前昆明湖 V2R2 定义的各个性能事件集合大小不超过 150，因此规定 EVENTx 高两位复位固定值：
- 对于 mhpmevent 3-10: 40'h0000000000
- 对于 mhpmevent11-18: 40'h4010040100
- 对于 mhpmevent19-26: 40'h8020080200
- 对于 mhpmevent27-31: 40'hc0300c0300

昆明湖 V2R2 将提供的性能事件根据来源分为四类，包括：前端，后端，访存，缓存，同时将计数器分为四部分，分别记录来自上述四个源头的性能事件：

- 前端：mhpmevent 3-10
- 后端：mhpmevent11-18
- 访存：mhpmevent19-26
- 缓存：mhpmevent27-31

表 13.6: 昆明湖 V2R2 前端性能事件索引表

| 索引 | 事件 |
|---|---|
| 0 | noEvent |
| 1 | frontendFlush |
| 2 | ifu_req |
| 3 | ifu_miss |
| 4 | ifu_req_cacheline_0 |
| 5 | ifu_req_cacheline_1 |

| 索引 | 事件 |
|---|---|
| 6 | ifu_req_cacheline_0_hit |
| 7 | ifu_req_cacheline_1_hit |
| 8 | only_0_hit |
| 9 | only_0_miss |
| 10 | hit_0_hit_1 |
| 11 | hit_0_miss_1 |
| 12 | miss_0_hit_1 |
| 13 | miss_0_miss_1 |
| 14 | IBuffer_Flushed |
| 15 | IBuffer_hungry |
| 16 | IBuffer_1_4_valid |
| 17 | IBuffer_2_4_valid |
| 18 | IBuffer_3_4_valid |
| 19 | IBuffer_4_4_valid |
| 20 | IBuffer_full |
| 21 | Front_Bubble |
| 22 | icache_miss_cnt |
| 23 | icache_miss_penalty |
| 24 | bpu_s2_redirect |
| 25 | bpu_s3_redirect |
| 26 | bpu_to_ftq_stall |
| 27 | mispredictRedirect |
| 28 | replayRedirect |
| 29 | predecodeRedirect |
| 30 | to_ifu_bubble |
| 31 | from_bpu_real_bubble |
| 32 | BpInstr |
| 33 | BpBInstr |
| 34 | BpRight |
| 35 | BpWrong |
| 36 | BpBRight |
| 37 | BpBWrong |
| 38 | BpJRight |
| 39 | BpJWrong |
| 40 | BpIRight |
| 41 | BpIWrong |
| 42 | BpCRight |
| 43 | BpCWrong |
| 44 | BpRRight |
| 45 | BpRWrong |
| 46 | ftb_false_hit |

| 索引 | 事件 |
|---|---|
| 47 | ftb_hit |
| 48 | fauftb_commit_hit |
| 49 | fauftb_commit_miss |
| 50 | tage_tht_hit |
| 51 | sc_update_on_mispred |
| 52 | sc_update_on_unconf |
| 53 | ftb_commit_hits |
| 54 | ftb_commit_misses |

表 13.7: 昆明湖 V2R2 后端性能事件索引表

| 索引 | 事件 |
|---|---|
| 0 | noEvent |
| 1 | decoder_fused_instr |
| 2 | decoder_waitInstr |
| 3 | decoder_stall_cycle |
| 4 | decoder_utilization |
| 5 | rename_in |
| 6 | rename_waitinstr |
| 7 | rename_stall |
| 8 | rename_stall_cycle_walk |
| 9 | rename_stall_cycle_dispatch |
| 10 | rename_stall_cycle_int |
| 11 | rename_stall_cycle_fp |
| 12 | rename_stall_cycle_vec |
| 13 | rename_stall_cycle_v0 |
| 14 | rename_stall_cycle_vl |
| 15 | me_freelist_1_4_valid |
| 16 | me_freelist_2_4_valid |
| 17 | me_freelist_3_4_valid |
| 18 | me_freelist_4_4_valid |
| 19 | std_freelist_1_4_valid |
| 20 | std_freelist_2_4_valid |
| 21 | std_freelist_3_4_valid |
| 22 | std_freelist_4_4_valid |
| 23 | std_freelist_1_4_valid |
| 24 | std_freelist_2_4_valid |
| 25 | std_freelist_3_4_valid |
| 26 | std_freelist_4_4_valid |
| 27 | std_freelist_1_4_valid |

| 索引 | 事件 |
|---|---|
| 28 | std_freelist_2_4_valid |
| 29 | std_freelist_3_4_valid |
| 30 | std_freelist_4_4_valid |
| 31 | std_freelist_1_4_valid |
| 32 | std_freelist_2_4_valid |
| 33 | std_freelist_3_4_valid |
| 34 | std_freelist_4_4_valid |
| 35 | dispatch_in |
| 36 | dispatch_empty |
| 37 | dispatch_utili |
| 38 | dispatch_waitinstr |
| 39 | dispatch_stall_cycle_lsq |
| 40 | dispatch_stall_cycle_rob |
| 41 | dispatch_stall_cycle_int_dq |
| 42 | dispatch_stall_cycle_fp_dq |
| 43 | dispatch_stall_cycle_ls_dq |
| 44 | dispatchq1_in |
| 45 | dispatchq1_out |
| 46 | dispatchq1_out_try |
| 47 | dispatchq1_fake_block |
| 48 | dispatchq1_1_4_valid |
| 49 | dispatchq1_2_4_valid |
| 50 | dispatchq1_3_4_valid |
| 51 | dispatchq1_4_4_valid |
| 52 | dispatchq2_in |
| 53 | dispatchq2_out |
| 54 | dispatchq2_out_try |
| 55 | dispatchq2_fake_block |
| 56 | dispatchq2_1_4_valid |
| 57 | dispatchq2_2_4_valid |
| 58 | dispatchq2_3_4_valid |
| 59 | dispatchq2_4_4_valid |
| 60 | dispatchq3_in |
| 61 | dispatchq3_out |
| 62 | dispatchq3_out_try |
| 63 | dispatchq3_fake_block |
| 64 | dispatchq3_1_4_valid |
| 65 | dispatchq3_2_4_valid |
| 66 | dispatchq3_3_4_valid |
| 67 | dispatchq3_4_4_valid |
| 68 | dispatchq4_in |

| 索引 | 事件 |
|---|---|
| 69 | dispatchq4_out |
| 70 | dispatchq4_out_try |
| 71 | dispatchq4_fake_block |
| 72 | dispatchq4_1_4_valid |
| 73 | dispatchq4_2_4_valid |
| 74 | dispatchq4_3_4_valid |
| 75 | dispatchq4_4_4_valid |
| 76 | rob_interrupt_num |
| 77 | rob_exception_num |
| 78 | rob_flush_pipe_num |
| 79 | rob_replay_inst_num |
| 80 | rob_commitUop |
| 81 | rob_commitInstr |
| 82 | rob_commitInstrMove |
| 83 | rob_commitInstrFused |
| 84 | rob_commitInstrLoad |
| 85 | rob_commitInstrBranch |
| 86 | rob_commitInstrLoadWait |
| 87 | rob_commitInstrStore |
| 88 | rob_walkInstr |
| 89 | rob_walkCycle |
| 90 | rob_1_4_valid |
| 91 | rob_2_4_valid |
| 92 | rob_3_4_valid |
| 93 | rob_4_4_valid |
| 94 | dispatch2Iq1_out_fire_cnt |
| 95 | issueQueue_enq_fire_cnt |
| 96 | IssueQueueAluMulBkuBrhJmp_full |
| 97 | IssueQueueAluMulBkuBrhJmp_full |
| 98 | IssueQueueAluBrhJmpI2fVsetriwiVsetriwvf_full |
| 99 | IssueQueueAluCsrFenceDiv_full |
| 100 | dispatch2Iq2_out_fire_cnt |
| 101 | issueQueue_enq_fire_cnt |
| 102 | IssueQueueFaluFcvtF2vFmac_full |
| 103 | IssueQueueFaluFmac_full |
| 104 | IssueQueueFaluFmac_full |
| 105 | IssueQueueFaluFmac_full |
| 106 | IssueQueueFdiv_full |
| 107 | dispatch2Iq3_out_fire_cnt |
| 108 | issueQueue_enq_fire_cnt |
| 109 | IssueQueueVfmaVialuFixVimacVppuVfaluVfcvtVipuVsetrvfwvf_full |

| 索引 | 事件 |
|---|---|
| 110 | IssueQueueVfmaVialuFixVfaluVfcvt_full |
| 111 | IssueQueueVfdivVidiv_full |
| 112 | dispatch2Iq4_out_fire_cnt |
| 113 | issueQueue_enq_fire_cnt |
| 114 | IssueQueueStaMou_full |
| 115 | IssueQueueStaMou_full |
| 116 | IssueQueueLdu_full |
| 117 | IssueQueueLdu_full |
| 118 | IssueQueueLdu_full |
| 119 | IssueQueueVlduVstuVseglduVsegstu_full |
| 120 | IssueQueueVlduVstu_full |
| 121 | IssueQueueStdMoud_full |
| 122 | IssueQueueStdMoud_full |
| 123 | bt_std_freelist_1_4_valid |
| 124 | bt_std_freelist_2_4_valid |
| 125 | bt_std_freelist_3_4_valid |
| 126 | bt_std_freelist_4_4_valid |
| 127 | bt_std_freelist_1_4_valid |
| 128 | bt_std_freelist_2_4_valid |
| 129 | bt_std_freelist_3_4_valid |
| 130 | bt_std_freelist_4_4_valid |
| 131 | bt_std_freelist_1_4_valid |
| 132 | bt_std_freelist_2_4_valid |
| 133 | bt_std_freelist_3_4_valid |
| 134 | bt_std_freelist_4_4_valid |
| 135 | bt_std_freelist_1_4_valid |
| 136 | bt_std_freelist_2_4_valid |
| 137 | bt_std_freelist_3_4_valid |
| 138 | bt_std_freelist_4_4_valid |
| 139 | bt_std_freelist_1_4_valid |
| 140 | bt_std_freelist_2_4_valid |
| 141 | bt_std_freelist_3_4_valid |
| 142 | bt_std_freelist_4_4_valid |

表 13.8: 昆明湖 V2R2 访存性能事件索引表

| 索引 | 事件 |
|---|---|
| 0 | noEvent |
| 1 | load_s0_in_fire |
| 2 | load_to_load_forward |

| 索引 | 事件 |
|---|---|
| 3 | stall_dcache |
| 4 | load_s1_in_fire |
| 5 | load_s1_tlb_miss |
| 6 | load_s2_in_fire |
| 7 | load_s2_dcache_miss |
| 8 | load_s0_in_fire (LoadUnit_1) |
| 9 | load_to_load_forward (LoadUnit_1) |
| 10 | stall_dcache (LoadUnit_1) |
| 11 | load_s1_in_fire (LoadUnit_1) |
| 12 | load_s1_tlb_miss (LoadUnit_1) |
| 13 | load_s2_in_fire (LoadUnit_1) |
| 14 | load_s2_dcache_miss (LoadUnit_1) |
| 15 | load_s0_in_fire (LoadUnit_2) |
| 16 | load_to_load_forward (LoadUnit_2) |
| 17 | stall_dcache (LoadUnit_2) |
| 18 | load_s1_in_fire (LoadUnit_2) |
| 19 | load_s1_tlb_miss (LoadUnit_2) |
| 20 | load_s2_in_fire (LoadUnit_2) |
| 21 | load_s2_dcache_miss (LoadUnit_2) |
| 22 | sbuffer_req_valid |
| 23 | sbuffer_req_fire |
| 24 | sbuffer_merge |
| 25 | sbuffer_newline |
| 26 | dcache_req_valid |
| 27 | dcache_req_fire |
| 28 | sbuffer_idle |
| 29 | sbuffer_flush |
| 30 | sbuffer_replace |
| 31 | mpipe_resp_valid |
| 32 | replay_resp_valid |
| 33 | coh_timeout |
| 34 | sbuffer_1_4_valid |
| 35 | sbuffer_2_4_valid |
| 36 | sbuffer_3_4_valid |
| 37 | sbuffer_full_valid |
| 38 | enq (LsqWrapper) |
| 39 | ld_ld_violation (LsqWrapper) |
| 40 | enq (LsqWrapper) |
| 41 | stld_rollback (LsqWrapper) |
| 42 | enq (LsqWrapper) |
| 43 | deq (LsqWrapper) |

| 索引 | 事件 |
|---|---|
| 44 | deq_block (LsqWrapper) |
| 45 | replay_full (LsqWrapper) |
| 46 | replay_rar_nack (LsqWrapper) |
| 47 | replay_raw_nack (LsqWrapper) |
| 48 | replay_nuke (LsqWrapper) |
| 49 | replay_mem_amb (LsqWrapper) |
| 50 | replay_tlb_miss (LsqWrapper) |
| 51 | replay_bank_conflict (LsqWrapper) |
| 52 | replay_dcache_replay (LsqWrapper) |
| 53 | replay_forward_fail (LsqWrapper) |
| 54 | replay_dcache_miss (LsqWrapper) |
| 55 | full_mask_000 (LsqWrapper) |
| 56 | full_mask_001 (LsqWrapper) |
| 57 | full_mask_010 (LsqWrapper) |
| 58 | full_mask_011 (LsqWrapper) |
| 59 | full_mask_100 (LsqWrapper) |
| 60 | full_mask_101 (LsqWrapper) |
| 61 | full_mask_110 (LsqWrapper) |
| 62 | full_mask_111 (LsqWrapper) |
| 63 | nuke_rollback (LsqWrapper) |
| 64 | nack_rollback (LsqWrapper) |
| 65 | mmioCycle (LsqWrapper) |
| 66 | mmioCnt (LsqWrapper) |
| 67 | mmio_wb_success (LsqWrapper) |
| 68 | mmio_wb_blocked (LsqWrapper) |
| 69 | stq_1_4_valid (LsqWrapper) |
| 70 | stq_2_4_valid (LsqWrapper) |
| 71 | stq_3_4_valid (LsqWrapper) |
| 72 | stq_4_4_valid (LsqWrapper) |
| 73 | dcache_wbq_req |
| 74 | dcache_wbq_1_4_valid |
| 75 | dcache_wbq_2_4_valid |
| 76 | dcache_wbq_3_4_valid |
| 77 | dcache_wbq_4_4_valid |
| 78 | dcache_mp_req |
| 79 | dcache_mp_total_penalty |
| 80 | dcache_missq_req |
| 81 | dcache_missq_1_4_valid |
| 82 | dcache_missq_2_4_valid |
| 83 | dcache_missq_3_4_valid |
| 84 | dcache_missq_4_4_valid |

| 索引 | 事件 |
|---|---|
| 85 | dcache_probq_req |
| 86 | dcache_probq_1_4_valid |
| 87 | dcache_probq_2_4_valid |
| 88 | dcache_probq_3_4_valid |
| 89 | dcache_probq_4_4_valid |
| 90 | load_req |
| 91 | load_replay |
| 92 | load_replay_for_data_nack |
| 93 | load_replay_for_no_mshr |
| 94 | load_replay_for_conflict |
| 95 | load_req |
| 96 | load_replay |
| 97 | load_replay_for_data_nack |
| 98 | load_replay_for_no_mshr |
| 99 | load_replay_for_conflict |
| 100 | load_req |
| 101 | load_replay |
| 102 | load_replay_for_data_nack |
| 103 | load_replay_for_no_mshr |
| 104 | load_replay_for_conflict |
| 105 | tlbllptw_incount |
| 106 | tlbllptw_inblock |
| 107 | tlbllptw_memcount |
| 108 | tlbllptw_memcycle |
| 109 | pagetablecache_access |
| 110 | pagetablecache_l2_hit |
| 111 | pagetablecache_l1_hit |
| 112 | pagetablecache_l0_hit |
| 113 | pagetablecache_sp_hit |
| 114 | pagetablecache_pte_hit |
| 115 | pagetablecache_rwHarzad |
| 116 | pagetablecache_out_blocked |
| 117 | fsm_count |
| 118 | fsm_busy |
| 119 | fsm_idle |
| 120 | resp_blocked |
| 121 | mem_count |
| 122 | mem_cycle |
| 123 | out_blocked |
| 124 | ldDeqCount (MemBlockInlined) |
| 125 | stDeqCount (MemBlockInlined) |

表 13.9: 昆明湖 V2R2 缓存性能事件索引表

| 索引 | 事件 |
|---|---|
| 0 | noEvent |
| 1 | req_buffer_merge |
| 2 | req_buffer_flow |
| 3 | req_buffer_alloc |
| 4 | req_buffer_full |
| 5 | recv_prefetch |
| 6 | recv_normal |
| 7 | nrWorkingABCmshr |
| 8 | nrWorkingBmshr |
| 9 | nrWorkingCmshr |
| 10 | conflictA |
| 11 | conflictByPrefetch |
| 12 | conflictB |
| 13 | conflictC |
| 14 | client_dir_conflict |
| 15 | selfdir_A_req |
| 16 | selfdir_A_hit |
| 17 | selfdir_B_req |
| 18 | selfdir_B_hit |
| 19 | selfdir_C_req |
| 20 | selfdir_C_hit |
| 21 | selfdir_dirty |
| 22 | selfdir_TIP |
| 23 | selfdir_BRANCH |
| 24 | selfdir_TRUNK |
| 25 | selfdir_INVALID |

## 13.4 PMU 相关的性能事件计数器

昆明湖 V2R2 的性能事件计数器共分为两组，分别是：机器模式事件计数器、监督模式事件计数器、用户模式事件计数器

表 13.10: 机器模式事件计数器列表

| 名称 | 索引 | 读写 | 介绍 | 复位值 |
|---|---|---|---|---|
| MCYCLE | 0xB00 | RW | 机器模式时钟周期计数器 | - |
| MINSTRET | 0xB02 | RW | 机器模式退休指令计数器 | - |
| MHPMCOUNTER3-31 | 0XB03-0XB1F | RW | 机器模式性能事件计数器 | 0 |

其中 MHPMCOUNTERx 计数器相应由 MHPMEVENTx 控制，指定计数相应的性能事件。

监督模式事件计数器包括监督模式计数器上溢中断标志寄存器 (SCOUNTOVF)

表 13.11: 监督模式计数器上溢中断标志寄存器 (SCOUNTOVF) 说明

| 名称 | 位域 | 读写 | 行为 | 复位值 |
|---|---|---|---|---|
| OFVEC | 31:3 | RO | mhpmcounterx 寄存器上溢标志位:<br>1：发生上溢<br>0：没有发生上溢 | 0 |
| – | 2:0 | RO 0 | – | 0 |

scountovf 作为 mhpmcounter 寄存器 OF 位的只读映射，受 xcounteren 控制:

- M-mode 访问 scountovf 可读正确值。
- HS-mode 访问 scountovf： mcounteren.HPMx 为 1 时，对应 OFVECx 可读正确值；否则只读 0。
- VS-mode 访问 scountovf：mcounteren.HPMx 以及 hcounteren.HPMx 均为 1 时，对应 OFVECx 可读正确值；否则只读 0。

表 13.12: 用户模式事件计数器列表

| 名称 | 索引 | 读写 | 介绍 | 复位值 |
|---|---|---|---|---|
| CYCLE | 0xC00 | RO | mcycle 寄存器用户模式只读副本 | - |
| TIME | 0xC01 | RO | 内存映射寄存器 mtime 用户模式只读副本 | - |
| INSTRET | 0xC02 | RO | minstret 寄存器用户模式只读副本 | - |
| HPMCOUNTER3-31 | 0XC03-0XC1F | RO | mhpmcounter3-31 寄存器用户模式只读副本 | 0 |