中国科学院计算技术研究所 INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES
中国开放指令生态（RISC-V）联盟 China RISC-V Alliance
北京开源芯片研究院 BEIJING INSTITUTE OF OPEN SOURCE CHIP

香山昆明湖后端流水线的设计演进

刘泽昊[3,1] 张紫飞[1] 胡轩[1] 张梓悦[1] 唐浩晋[1] 贾志杰[1] 林志达[1]
吉骏雄[1] 曾锦鸿[1] 肖飞豹[2] 程光辉[2] 游朝阳[2] 张铭旭[4]

[1]中国科学院计算技术研究所 [2]北京开源芯片研究院

[3]中国科学院大学 [4]西安交通大学

2025 年 7 月 16 日@第五届 RISC-V 中国峰会

目录

- 昆明湖 V2 后端功能设计演进
  - RVA23 Profile 的特权部分实现
  - RISC-V 高级中断架构
  - RISC-V E-Trace 拓展
- 昆明湖 V2 后端性能设计演进
  - 分派阶段优化
  - 寄存器堆缓存
- 昆明湖 V3 下阶段优化点

目录

- **昆明湖 V2 后端功能设计演进**
  - **RVA23 Profile 的特权部分实现**
  - RISC-V 高级中断架构
  - RISC-V E-Trace 拓展
- 昆明湖 V2 后端性能设计演进
  - 分派阶段优化
  - 寄存器堆缓存
- 昆明湖 V3 下阶段优化点

RVA23 Profile 概览（后端部分）

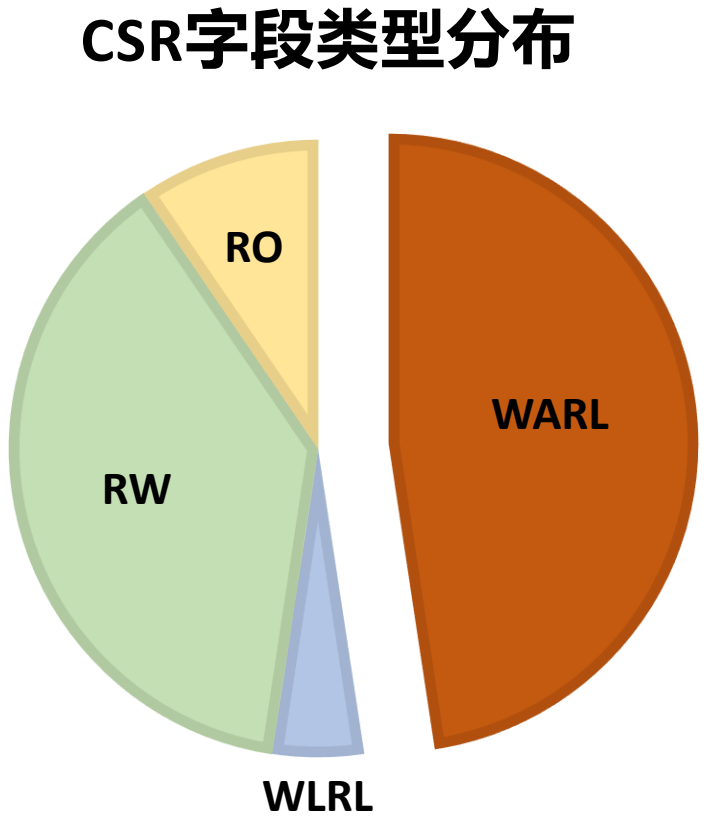
- 非特权级
  - 运算和访存：V, Zfh, Zvfh, Zimop, Zacas
  - 其它：Zihintpause
- 特权级
  - 异常：Sstvala, Shtvala, Shvstvala, Smdbltrp, Ssdbltrp
  - 中断：Smaia, Ssaia, Sstc, Smrnmi
  - H扩展：H, Shgatpa, Shvstvecd
  - 性能计数器：Sscofpmf, Sscounterenw, Shcounterenw, Smcsrind
  - 状态控制：Smstateen, Ssstateen
  - Debug：Sdtrig

RVA23 Profile 概览（后端部分）

- 非特权级
  - 运算和访存：V, Zfh, Zvfh, Zimop, Zacas
  - 其它：Zihintpause
- 特权级
  - 异常：Sstvala, Shtvala, Shvstvala, Smdbltrp, Ssdbltrp
  - 中断：Smaia, Ssaia, Sstc, Smrnmi
  - H扩展：H, Shgatpa, Shvstvecd
  - 性能计数器：Sscofpmf, Sscounterenw, Shcounterenw, Smcsrind
  - 状态控制：Smstateen, Ssstateen
  - Debug：Sdtrig

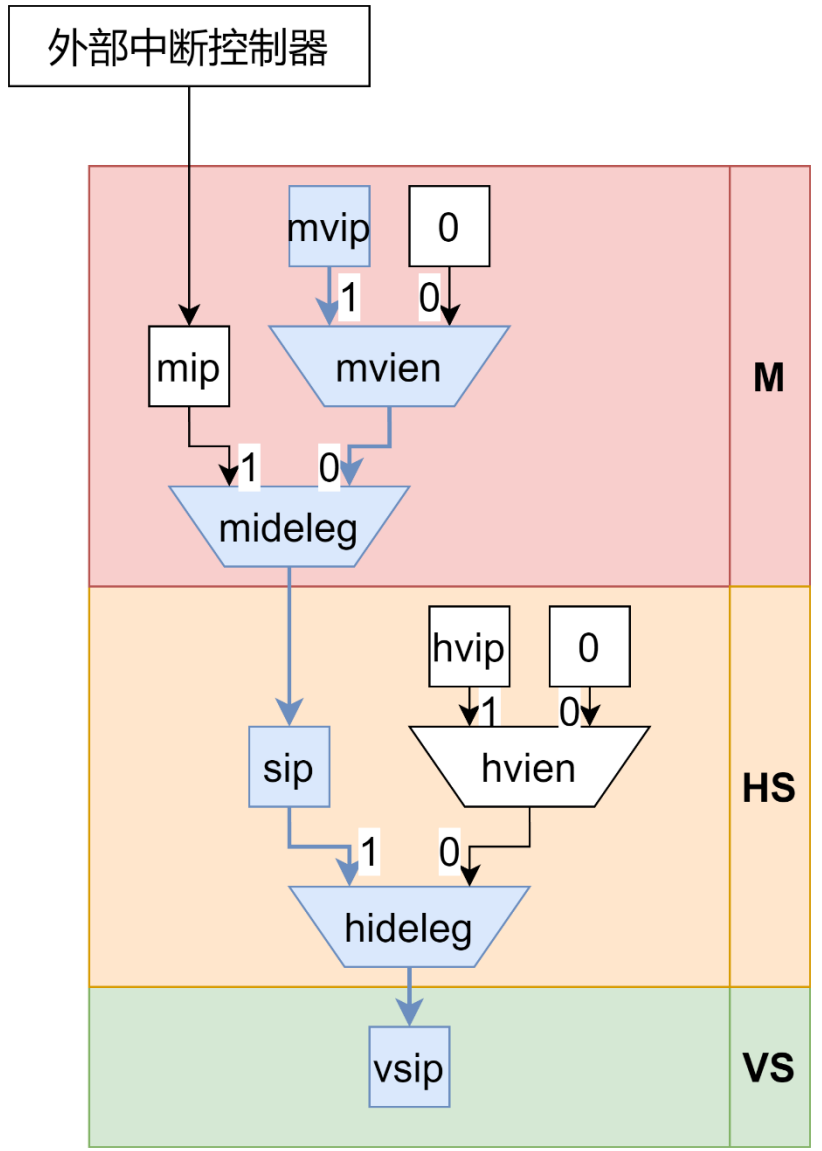
特权级规范的核心是 CSR

- 每个 CSR 接口由多个字段组成
  - mstatus 包含 FS、VS、MIP 等字段
- 规范约定了每个字段的基础属性
  - RW、RO、WARL 和 WLRL
  - 但不够准确，绝大多数字段都是 WARL
  - **别名关系**和**硬件可配置选项**让 WARL 多样且复杂
- H + AIA 组合让特权模式更加复杂！
- 需要更精确字段描述！



新版 CSR 设计原则 1

- 提供更准确的字段描述
  - 别名：sstatus.VS 是 mstatus.VS 的别名（读写）
  - 视图：sip.SEIP 是 mip.SEIP 的视图（只读）
  - 条件别名：
    当 mvien=1 且 mideleg=0 且 hideleg=1时，
    vsip 是 mvip 的别名（或视图）
  - 实现基于别名和视图的**跨特权级信息穿透机制**
- 抽象出统一接口
  - 统一读写接口定义，防止实现混乱
  - 统一事件更新 CSR 字段接口，处理特权事件



新版 CSR 设计原则 2

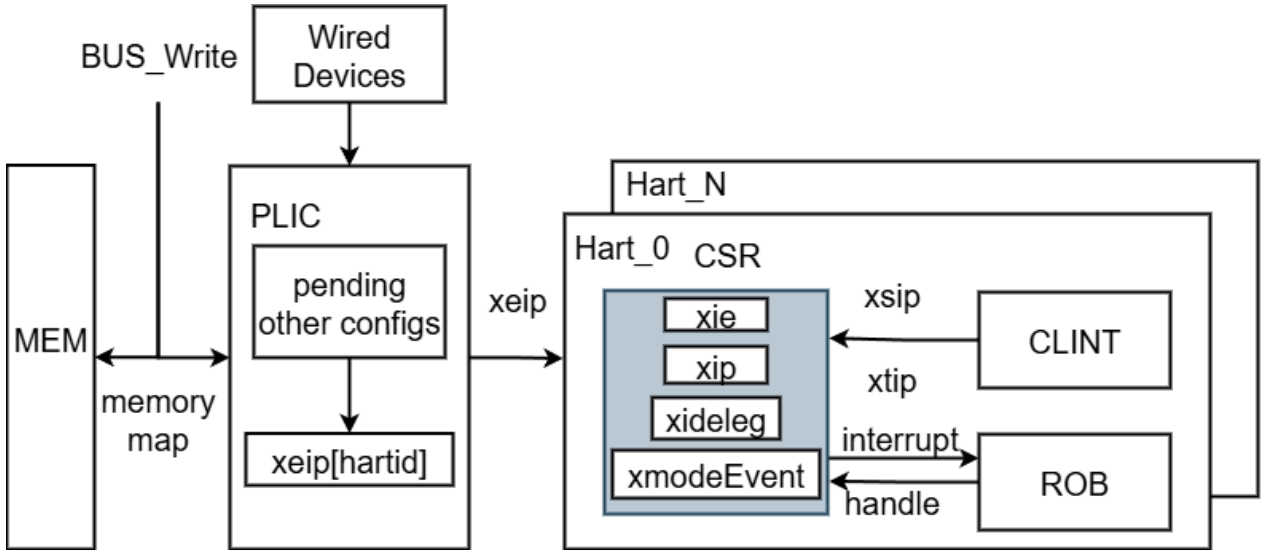
- CSR 处理 Trap 事件模块化
  - 五种特权级结合Trap进入和返回事件
    - Debug、MN、M、S、VS
    - Trap 进入和 Trap 返回
  - MachineLevelEntryEvent 处理进入 M 特权级的全部事件
  - 事件更新 CSR 字段接口，直接更新字段
- 精细化权限检查单元
  - CSR 访问权限检查
    - 特权级检查、直接和间接访问权限限制检查
  - 指令执行检查
    - 特权级、条件执行权限

目录

- **昆明湖 V2 后端功能设计演进**
  - RVA23 Profile 的特权部分实现
  - **RISC-V 高级中断架构**
  - RISC-V E-Trace 拓展
- 昆明湖 V2 后端性能设计演进
  - 分派阶段优化
  - 寄存器堆缓存
- 昆明湖 V3 下阶段优化点

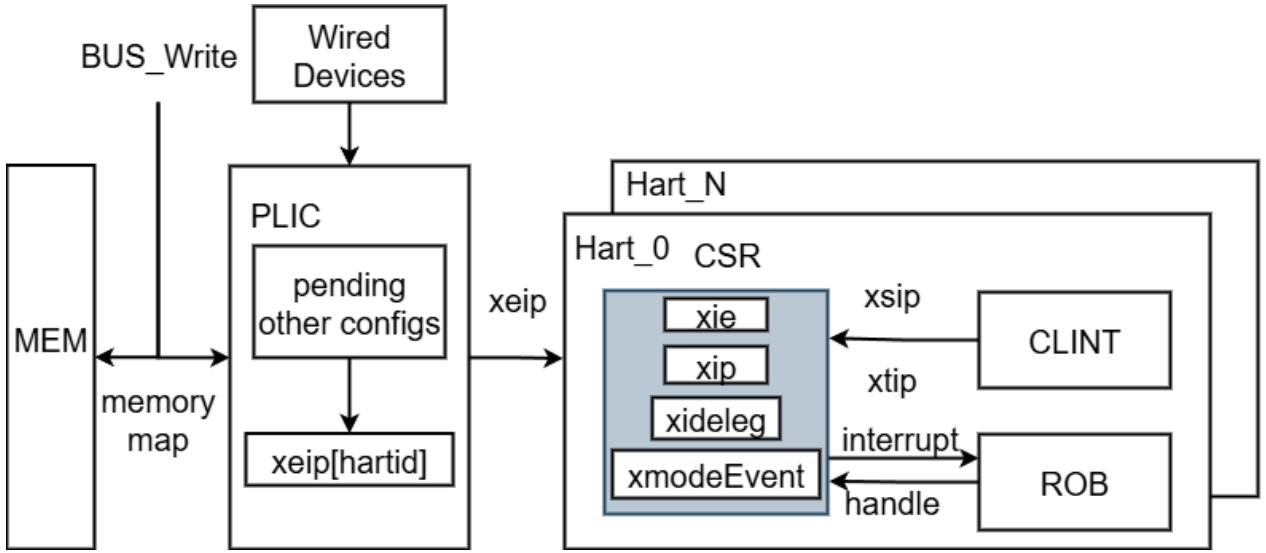
传统中断控制架构

- PLIC(平台级中断控制器)--外部中断
  - 中断配置
    - 内存映射寄存器
  - 中断产生
    - 仲裁最高优先级中断
  - 核内中断处理
    - 根据线中断信号拉高对应pending
    - 将中断代理至对应特权级处理
  - 软件中断处理
    - claim 请求原子清除 pending
- CLINT(核本地中断控制器)—软件/定时器中断



传统中断控制架构

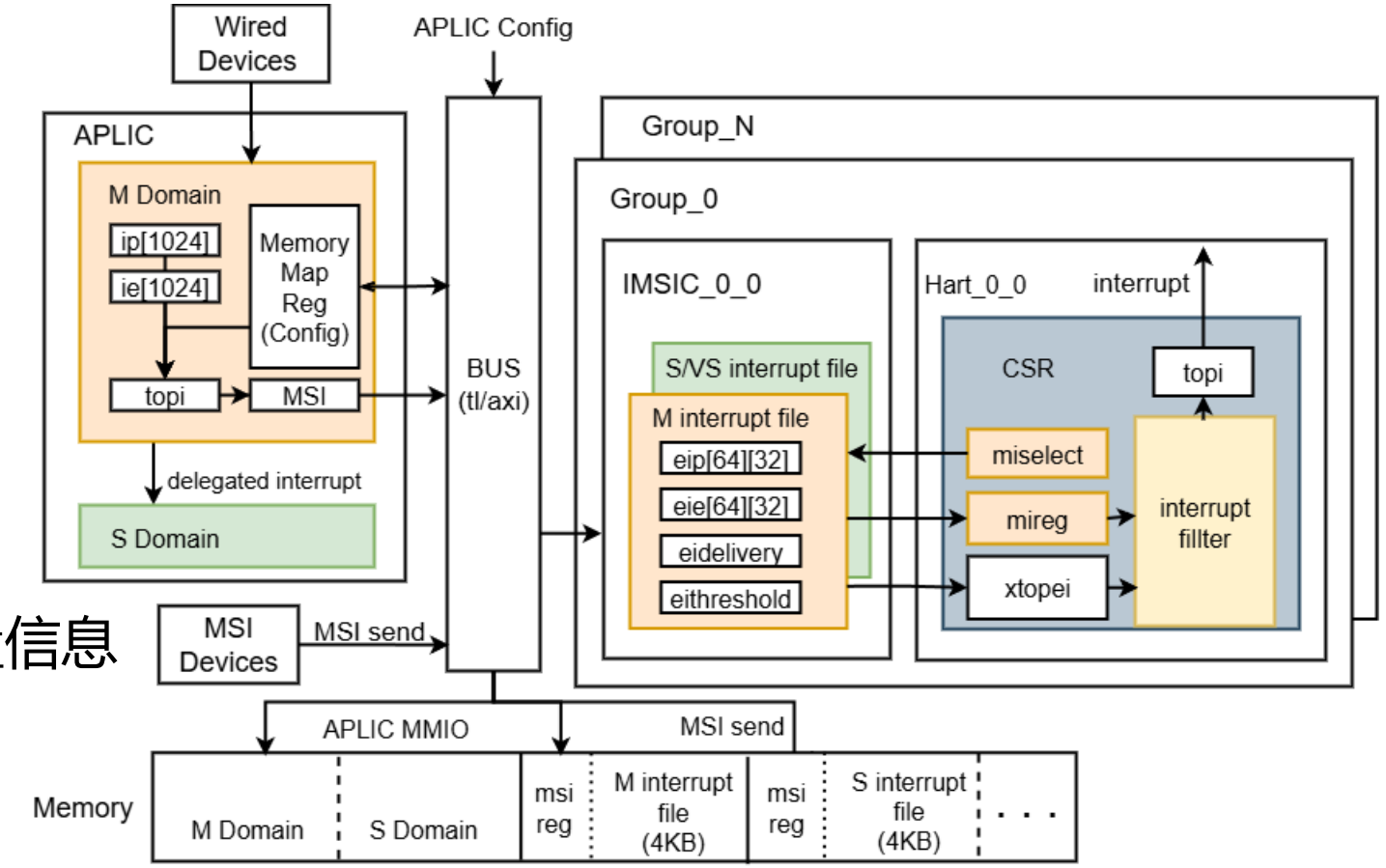
- 传统中断控制架构缺陷
  - 可拓展性差
    - 在核数、中断源数规模较大时，仲裁延迟大
  - 缺乏虚拟化中断控制
    - 仅能通过软件模拟注入虚拟化中断（延迟高）
  - 仅支持外设线级中断
  - 可配置性差
    - 仅支持外部中断内部优先级配置



- RISC-V 高级中断架构（AIA）

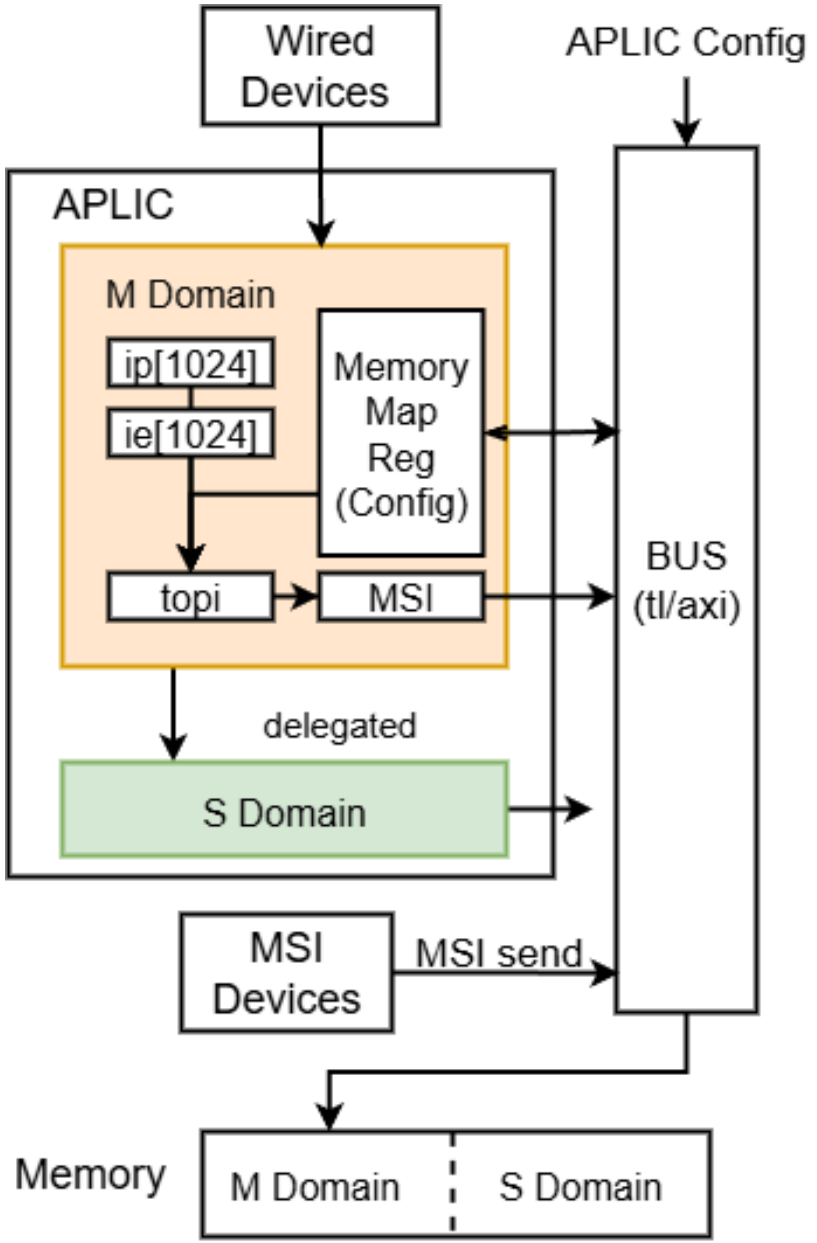
RISC-V 高级中断控制架构

- 高级平台中断控制器（APLIC）
  - 线中断转化为消息中断
  - 分层中断域管理
- 消息中断控制器（IMSIC）
  - 分布式架构 (提高拓展性)
  - 支持虚拟化场景
  - 接收消息中断
  - 支持通过间接访问 CSR 更改配置信息
- AIA 拓展 CSR
  - xtopi/topei 最高优先级中断编号
  - xiselect/xireg 间接访问寄存器



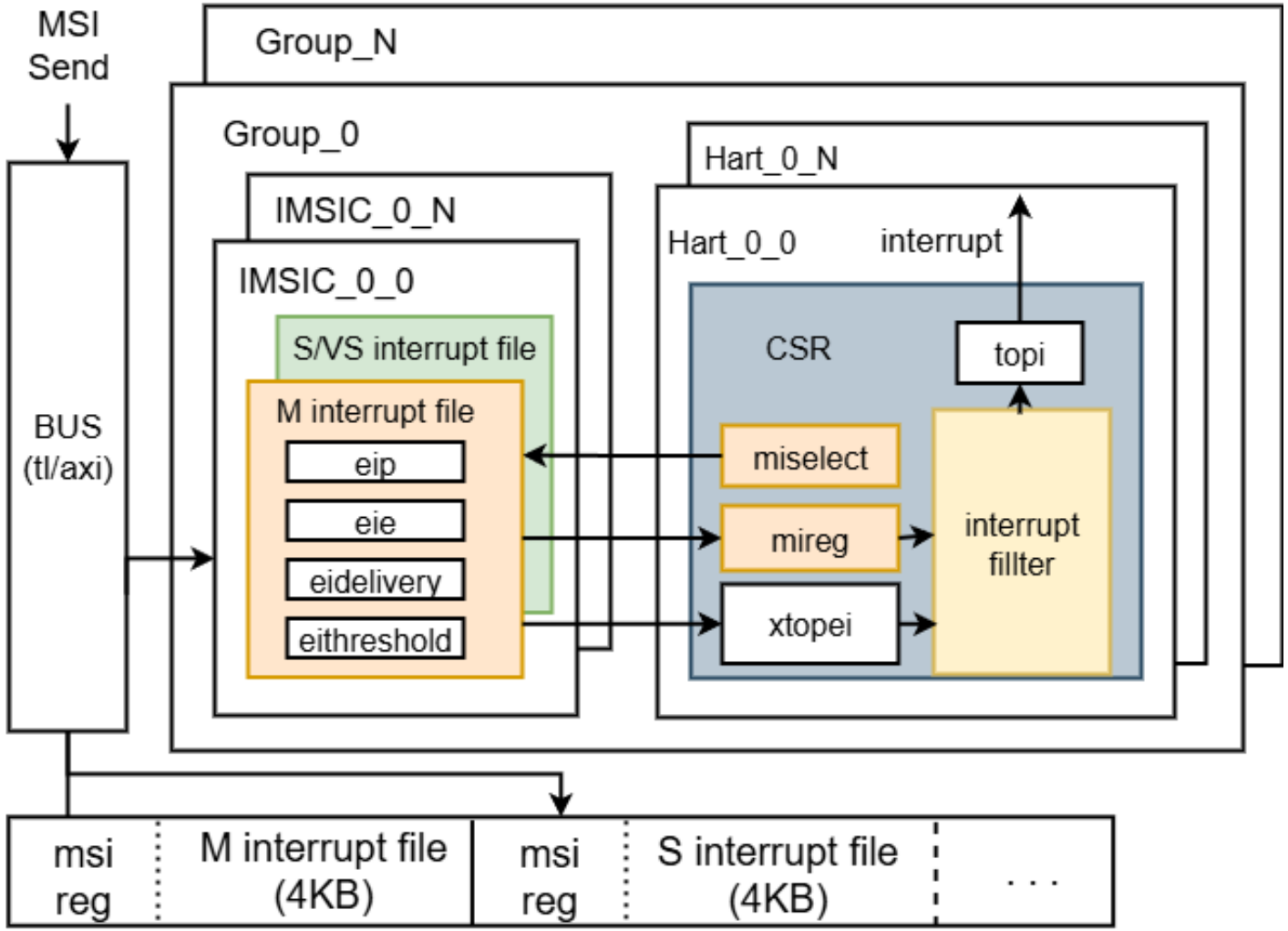
RISC-V 高级中断控制架构

- 高级平台中断控制器（APLIC）
  - 可配置的消息/线中断生成
  - 分层中断域管理
- 中断配置（内存映射寄存器）
  - 软件访问主 domain 配置中断域信息
  - 低特权级软件访问各自 domain 独立配置中断信息
- 中断产生
  - 主 domain 接收外设线中断并处理或委托
  - 各自 domain仲裁选出最高优先级中断



RISC-V 高级中断控制架构

- 消息中断控制器（IMSIC）
  - 支持外设写入消息直接触发中断
  - 分层分组管理的中断架构
    - 多核心分组
    - 特权级分层
  - 支持虚拟化中断控制
- 中断配置
  - 寄存器间接访问
  - 优先级直接通过编号确定
    - 消息中断原生可配置优先级
    - 线中断必须通过外设软件额外配置



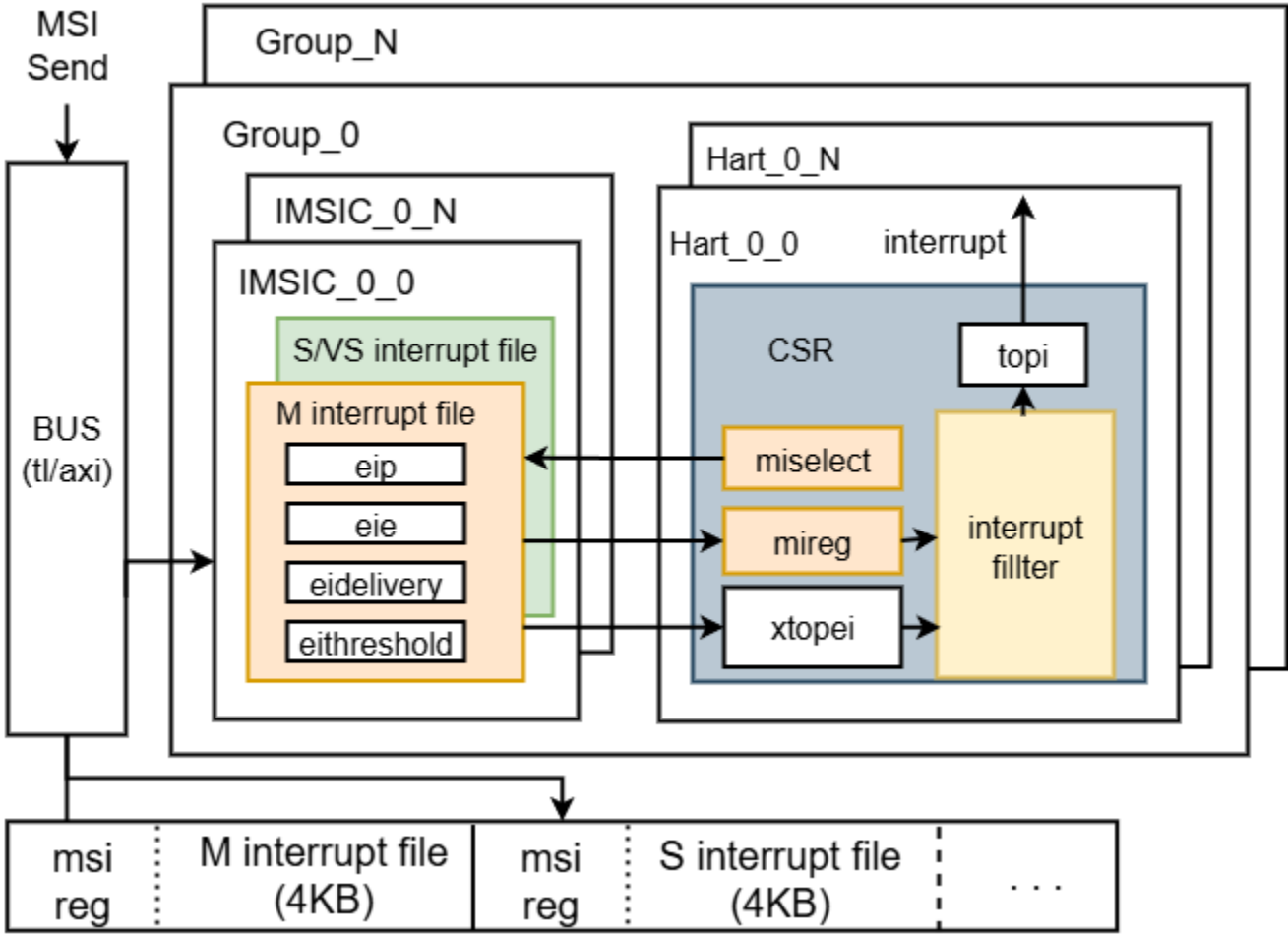
RISC-V 高级中断控制架构

- 中断产生
  - 总线消息请求映射到 IMSIC 的中断文件

Base	PrivilegeLevel	GroupID	HartID	GuestID	...

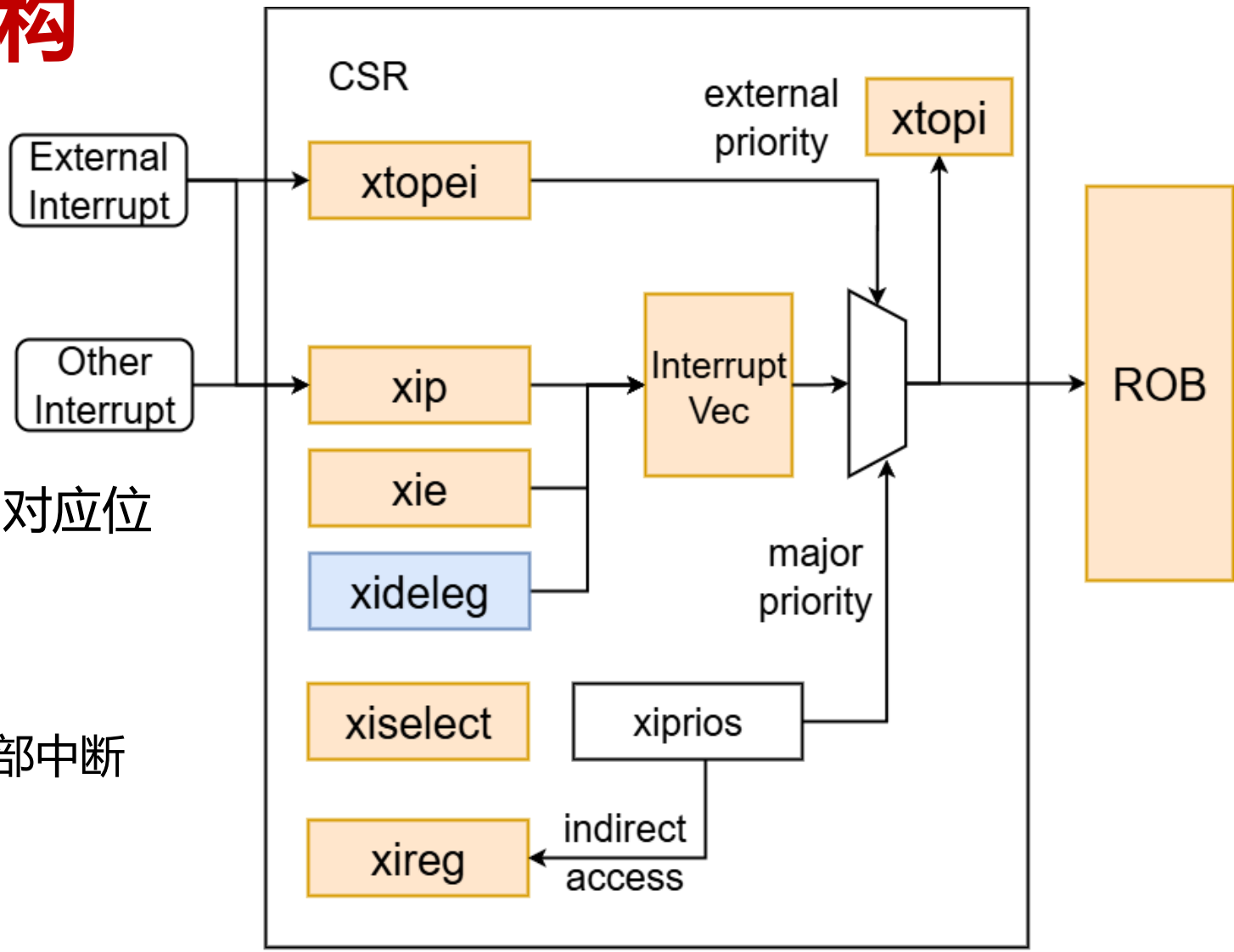
  - 各中断文件仲裁最高优先级中断
- 软件中断处理
  - 通过读写 xtopei 发送 claim请求
  - Claim 请求原子性清除对应

```
call handler major(hard)
num = CSRRW *topei
num >>16(minor)
call handler for ei num(soft call func)
func ret(soft)
*ret(hard)
```



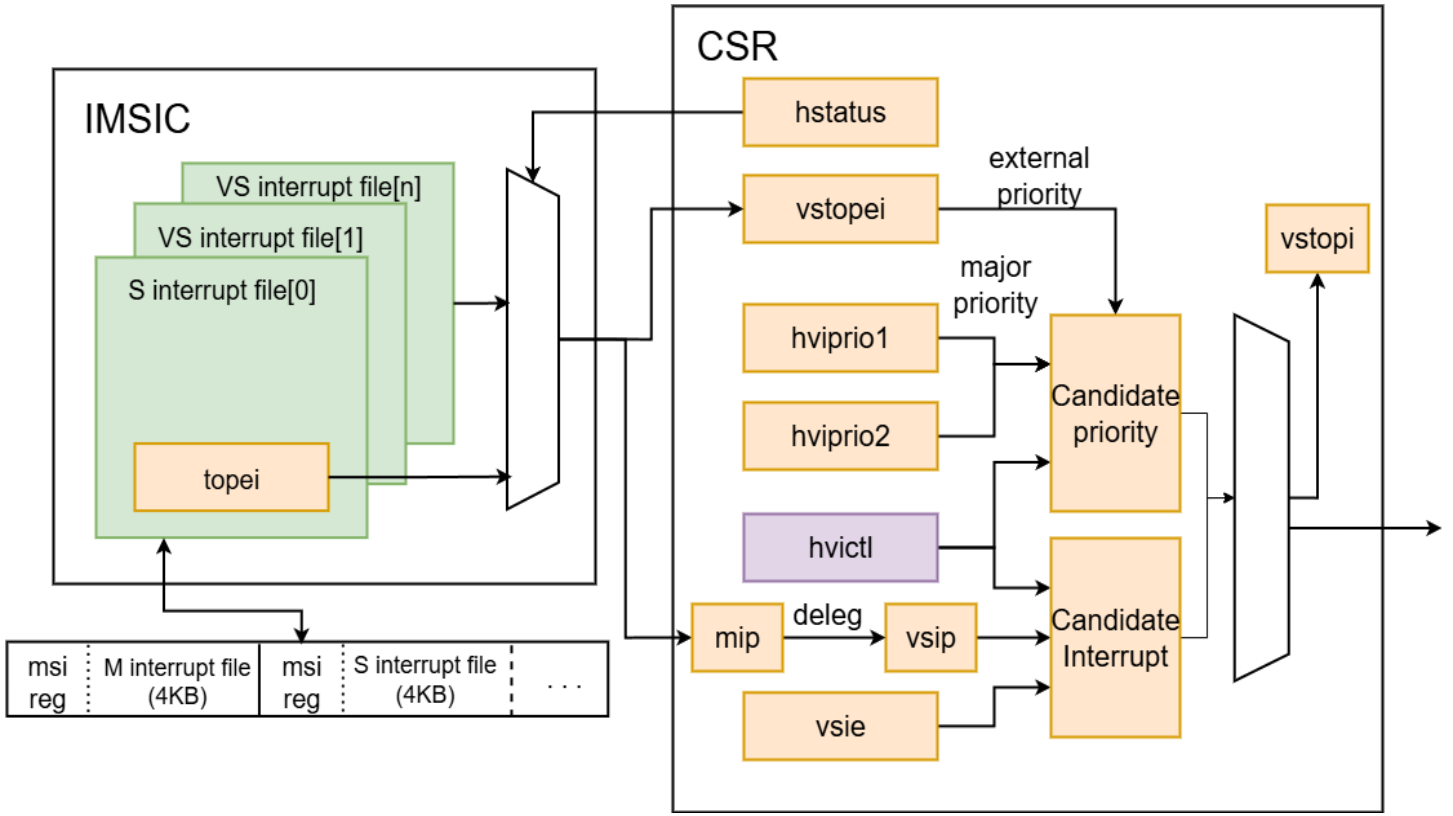
RISC-V 高级中断控制架构

- 主中断优先级配置
  - 寄存器间接访问xiprio
- 核内中断处理(M/S中断场景)
  - 接收中断信号，拉高对应 xip 位
  - 额外接收外部中断 ID，拉高 xtopei 对应位
  - xip, xie 共同决定候选中断向量
  - 仲裁选出最高优先级中断
    - 主中断优先级(time, software) + 外部中断优先级(xtopei)
  - 向 ROB 发出中断处理请求
  - 根据 代理信息，在特定特权级处理中断



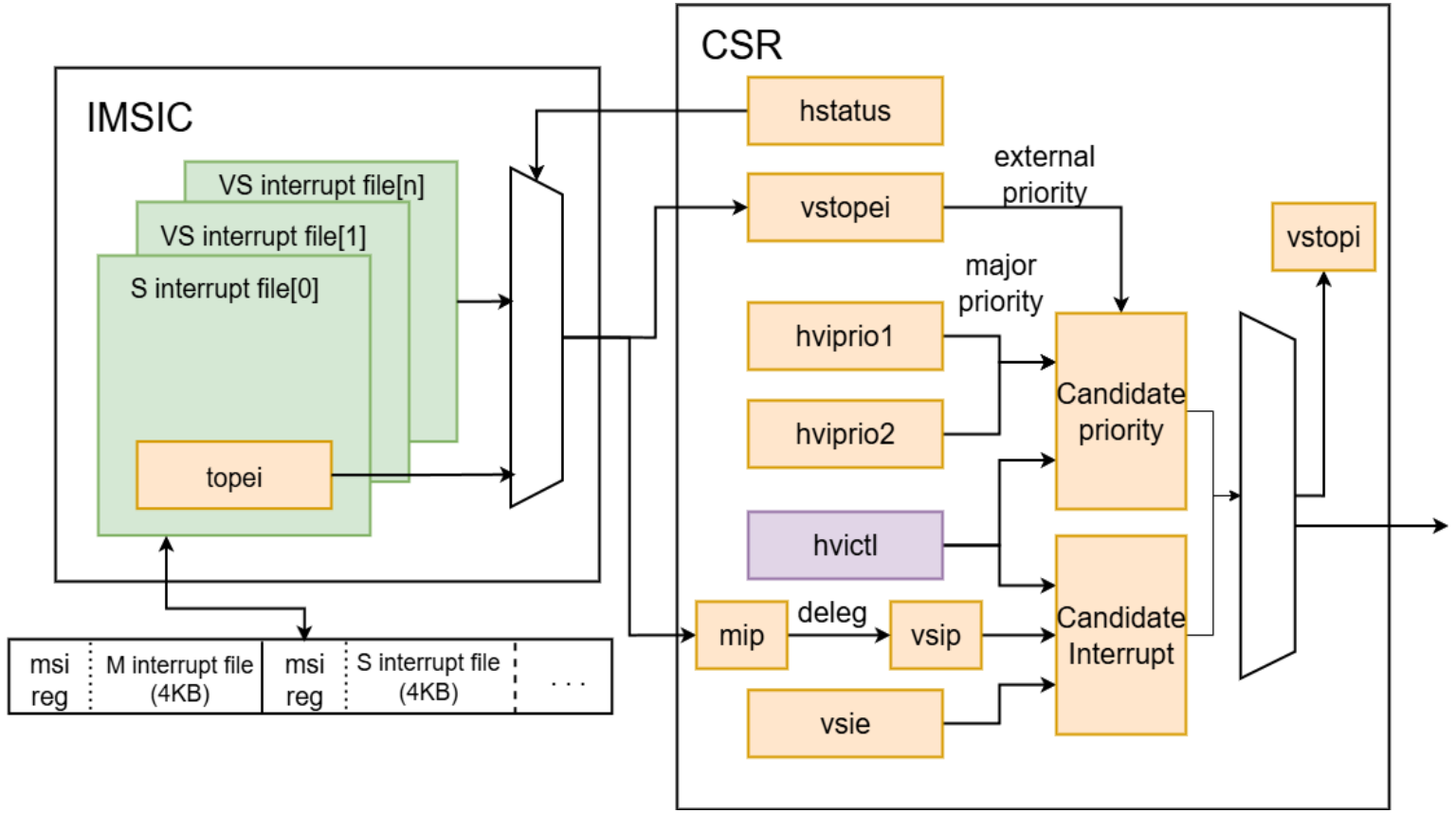
RISC-V 高级中断控制架构

- 传统虚拟外部中断(无AIA)
  - 委托操作系统模拟注入(延迟高)
- **虚拟中断直通**
  - iommu 构建 interrupt file 映射
  - 通过 memory-map 经映射直接触发虚拟中断(延迟较低)
- 虚拟中断注入
  - 宿主机通过写 hvictl 向客户机注入中断
  - 无须高延迟 mmio 访问，**配置灵活**



RISC-V 高级中断控制架构

- 虚拟中断核内处理(支持直通 + hvictl 注入)
  - 两套中断来源
    - 来自 ip, ie 即虚拟中断直通
    - 来自 hvictl.IID 直接注入
  - 两套主中断优先级
    - 来自 hviprio1, hviprio2配置
    - 来自 hvictl.iprio 注入
  - 两套外部中断的主中断优先级
    - 来自 IMISC 选出的 topei 编号
    - 来自 hvictl.iprio
  - **混合仲裁**得到最高优先级中断



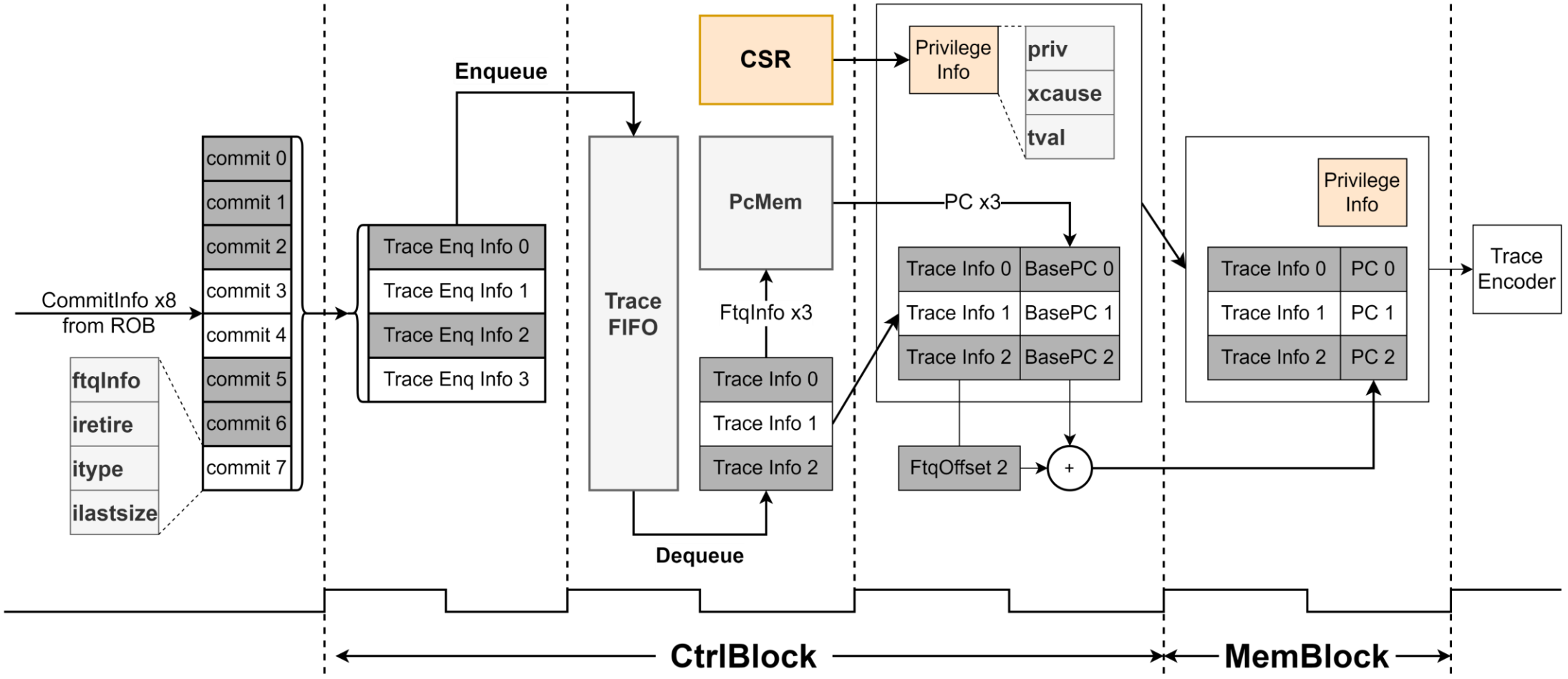
目录

- **昆明湖 V2 后端功能设计演进**
  - RVA23 Profile 的特权部分实现
  - RISC-V 高级中断架构
  - **RISC-V E-Trace 拓展**
- 昆明湖 V2 后端性能设计演进
  - 分派阶段优化
  - 寄存器堆缓存
- 昆明湖 V3 下阶段优化点

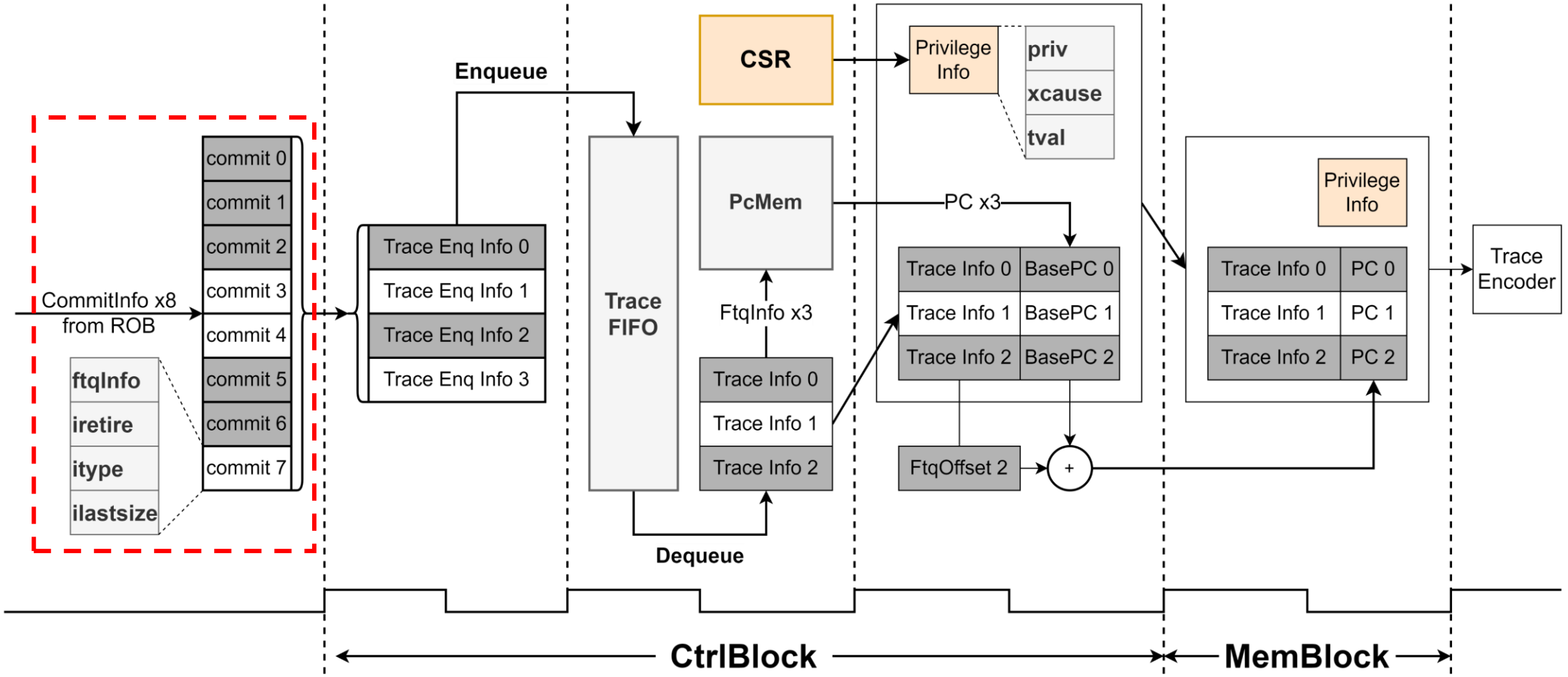
RISC-V Trace 规范概览

- Trace 拓展用途
  - 记录处理器运行过程中**关键行为的数据**，还原现场
    - 指令执行、访存操作、控制流变化等
  - 用于测试：指令级重放，CFI 验证等
  - 用于性能分析：热点片段
- 两个 Trace 规范
  - E-Trace 拓展
    - 仅记录程序流变化行为，利用静态代码结构重构完成路径
  - N-Trace 拓展
    - 记录每条指令精确行为：PC、指令、访存信息、寄存器读写等

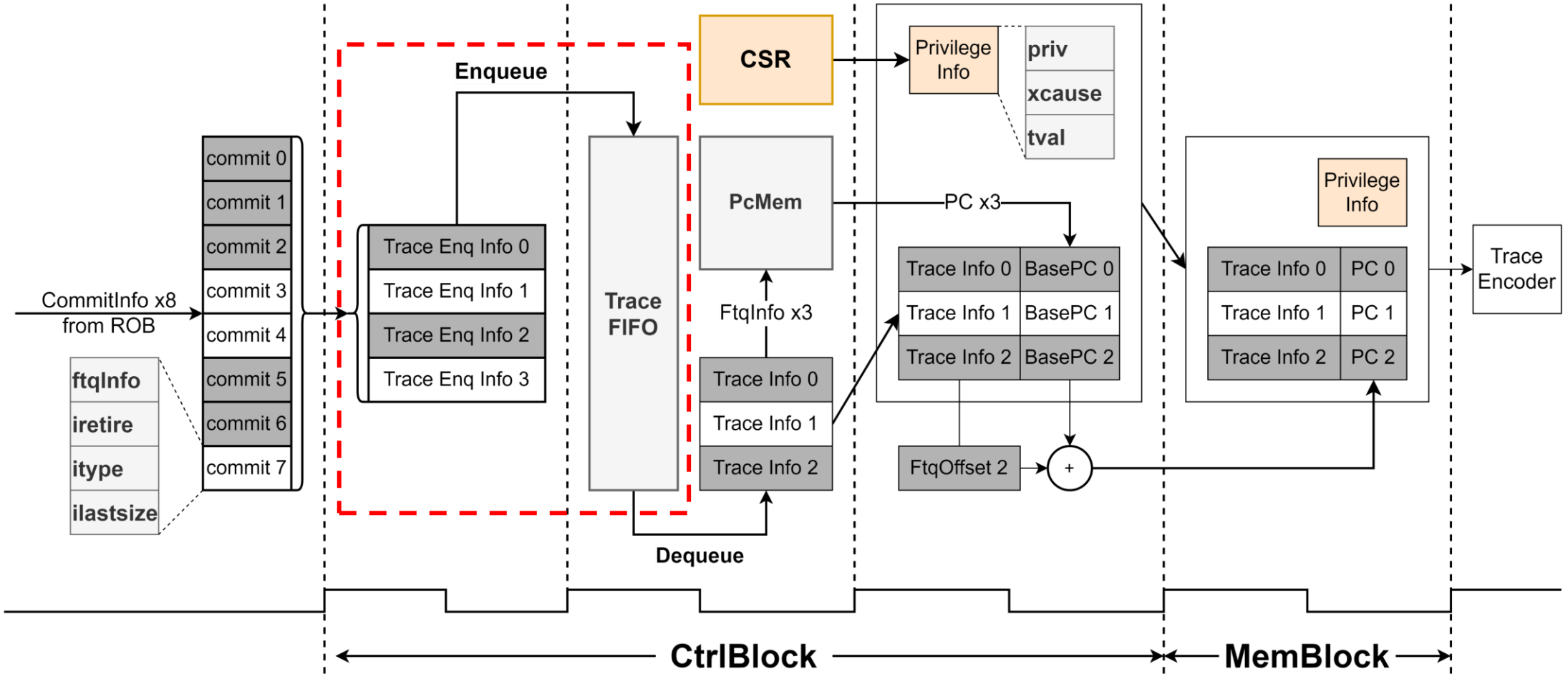
E-Trace 核内实现概览



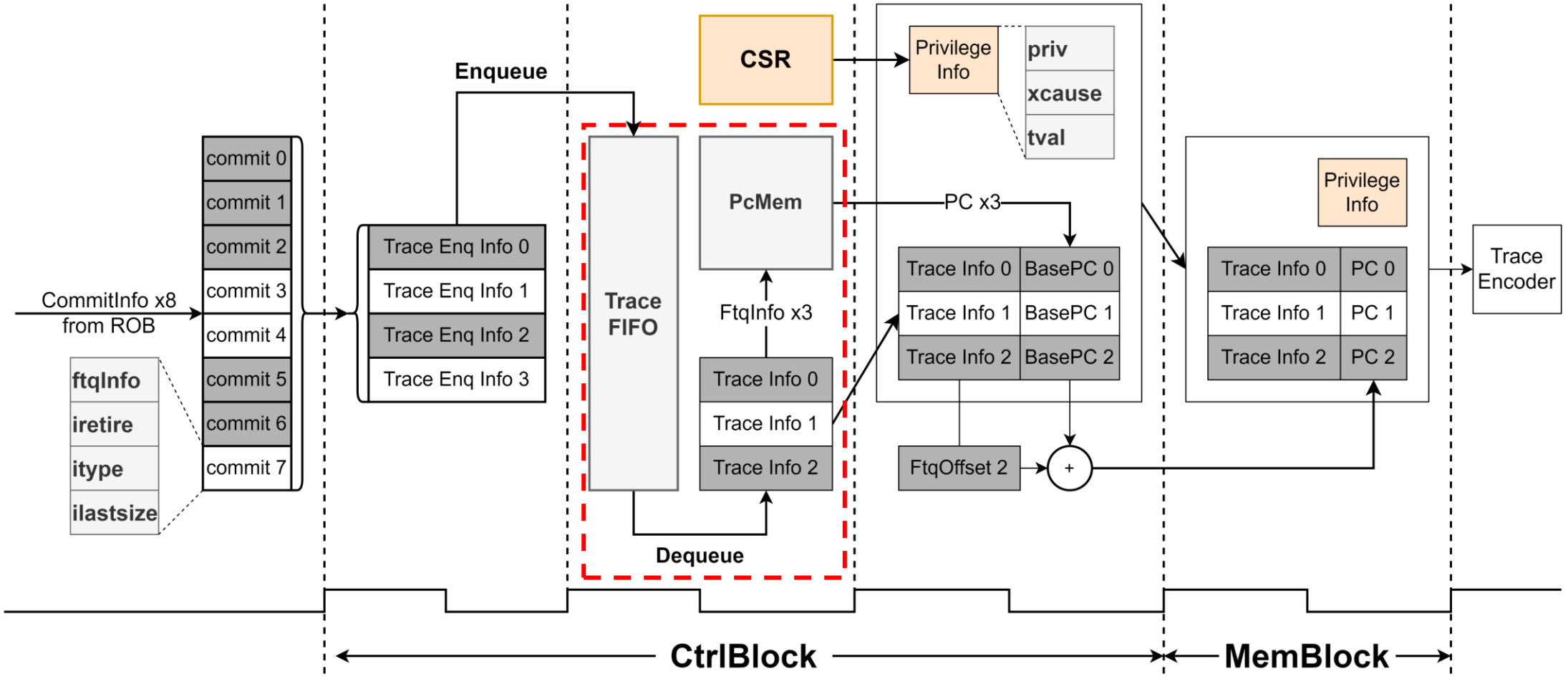
记录指令流变化


CommitInfo x8
from ROB
commit 0
commit 1
commit 2
commit 3
commit 4
commit 5
commit 6
commit 7
ftqInfo
iretire
itype
ilastsize
Enqueue
Trace Enq Info 0
Trace Enq Info 1
Trace Enq Info 2
Trace Enq Info 3
Trace FIFO
CSR
PcMem
FtqInfo x3
Trace Info 0
Trace Info 1
Trace Info 2
Dequeue
Privilege Info
priv
xcause
tval
PC x3
Trace Info 0
BasePC 0
Trace Info 1
BasePC 1
Trace Info 2
BasePC 2
FtqOffset 2
+
Privilege Info
Trace Info 0
PC 0
Trace Info 1
PC 1
Trace Info 2
PC 2
Trace Encoder
CtrlBlock
MemBlock

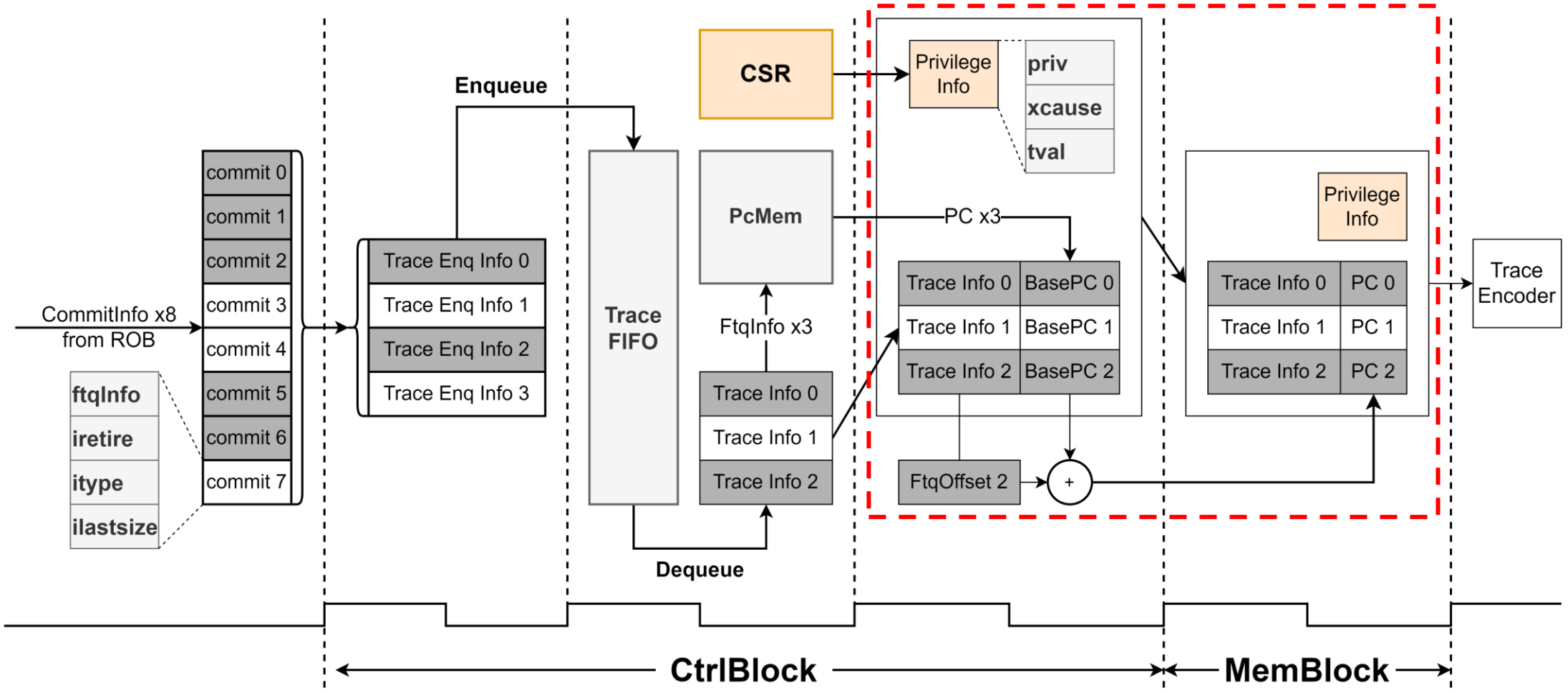
压缩提交指令 & 入队


CSR
Privilege Info
priv
xcause
tval
Enqueue
commit 0
commit 1
commit 2
commit 3
commit 4
commit 5
commit 6
commit 7
CommitInfo x8 from ROB
ftqInfo
iretire
itype
ilastsize
Trace Enq Info 0
Trace Enq Info 1
Trace Enq Info 2
Trace Enq Info 3
Trace FIFO
PcMem
PC x3
FtqInfo x3
Trace Info 0
Trace Info 1
Trace Info 2
Trace Info 0 BasePC 0
Trace Info 1 BasePC 1
Trace Info 2 BasePC 2
FtqOffset 2
+
Privilege Info
Trace Info 0 PC 0
Trace Info 1 PC 1
Trace Info 2 PC 2
Trace Encoder
Dequeue
CtrlBlock
MemBlock

出队 & 读PCMem


CSR
Enqueue
Privilege Info
priv
xcause
tval
commit 0
commit 1
commit 2
commit 3
commit 4
commit 5
commit 6
commit 7
CommitInfo x8 from ROB
ftqInfo
iretire
itype
ilastsize
Trace Enq Info 0
Trace Enq Info 1
Trace Enq Info 2
Trace Enq Info 3
Trace FIFO
PcMem
FtqInfo x3
Trace Info 0
Trace Info 1
Trace Info 2
PC x3
Trace Info 0
BasePC 0
Trace Info 1
BasePC 1
Trace Info 2
BasePC 2
FtqOffset 2
+
Privilege Info
Trace Info 0
PC 0
Trace Info 1
PC 1
Trace Info 2
PC 2
Trace Encoder
Dequeue
CtrlBlock
MemBlock

计算 PC & 获取特权信息

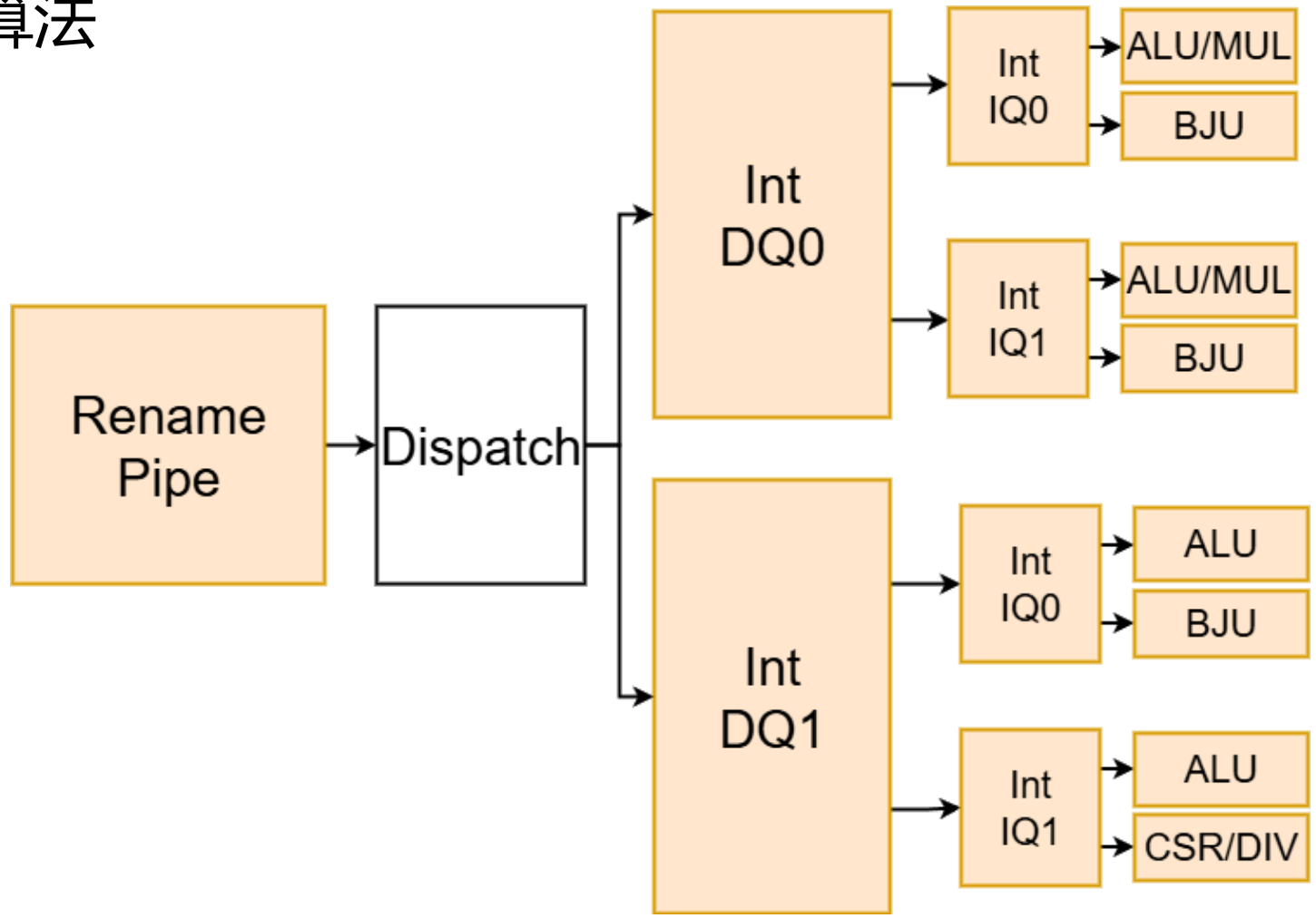

CSR
Privilege Info
priv
xcause
tval
Enqueue
commit 0
commit 1
commit 2
commit 3
commit 4
commit 5
commit 6
commit 7
CommitInfo x8 from ROB
ftqInfo
iretire
itype
ilastsize
Trace Enq Info 0
Trace Enq Info 1
Trace Enq Info 2
Trace Enq Info 3
Trace FIFO
PcMem
FtqInfo x3
Trace Info 0
Trace Info 1
Trace Info 2
Dequeue
PC x3
Trace Info 0
BasePC 0
Trace Info 1
BasePC 1
Trace Info 2
BasePC 2
FtqOffset 2
+
Privilege Info
Trace Info 0
PC 0
Trace Info 1
PC 1
Trace Info 2
PC 2
Trace Encoder
CtrlBlock
MemBlock

目录

- 昆明湖 V2 后端功能设计演进
  - RVA23 Profile
  - RISC-V 高级中断架构
  - RISC-V E-Trace 拓展
- **昆明湖 V2 后端性能设计演进**
  - **分派阶段优化**
  - 寄存器堆缓存
- 昆明湖 V3 下阶段优化点

昆明湖 V1 分派算法

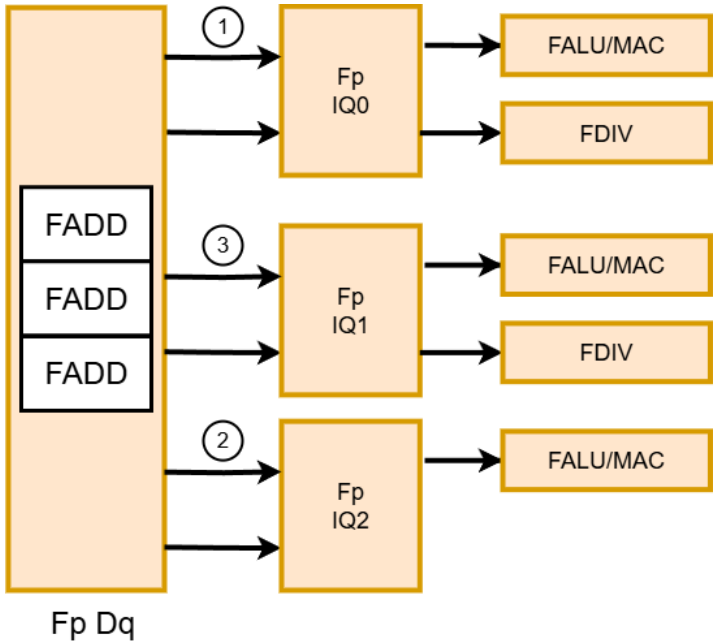
- 分布式发射队列带来复杂分派算法
  - 指令**均衡分配**
  - 指令应分尽分
  - 相邻指令分派不同发射队列
- **发射队列缓存形式**的分派算法
  - 将分派阶段划分为两周期
  - 按**指令类型**增加发射队列缓存
    - 降低分派算法复杂度
    - 一定程度提高乱序窗口
  - 发射队列缓存出队均衡分派
    - 类 RoundRobin 轮询



昆明湖 v2 分派算法 —— 动机

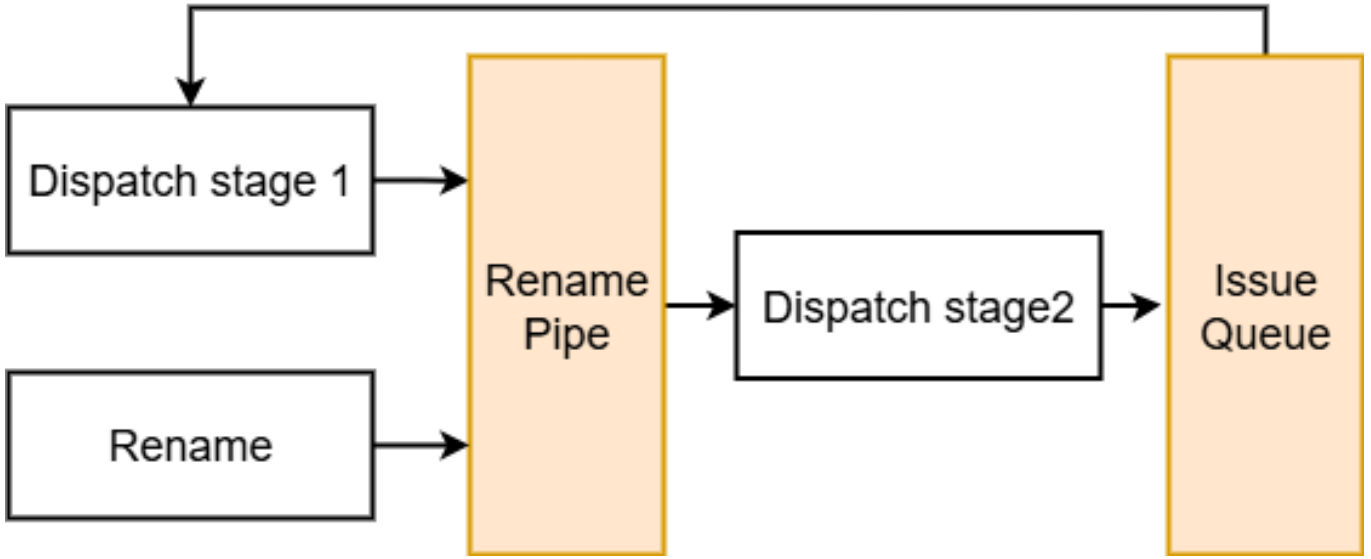
- 现有算法缺陷：
  - 发射队列缓存增大电路面积
    - 总计 82 项 (robidx + uopinfo(rename/decode) + val
  - 大队列出入队增大电路时序与功耗
  - 两周期分派加深流水线级数
  - 轮询无法保证 IQ 容量均衡
- 基于如下观察：
  - 分派两周期用于分派算法的逻辑并不多
    - 大队列出入队（Dq/Rob）
  - 昆明湖 v2 配置下，发射队列缓存收益不大

	IntDq	FpDq	VecDq	LsDq
数量	2	1	1	1
项数	16	16	16	18
入队数	6	6	6	6
出队数	4	6	6	6



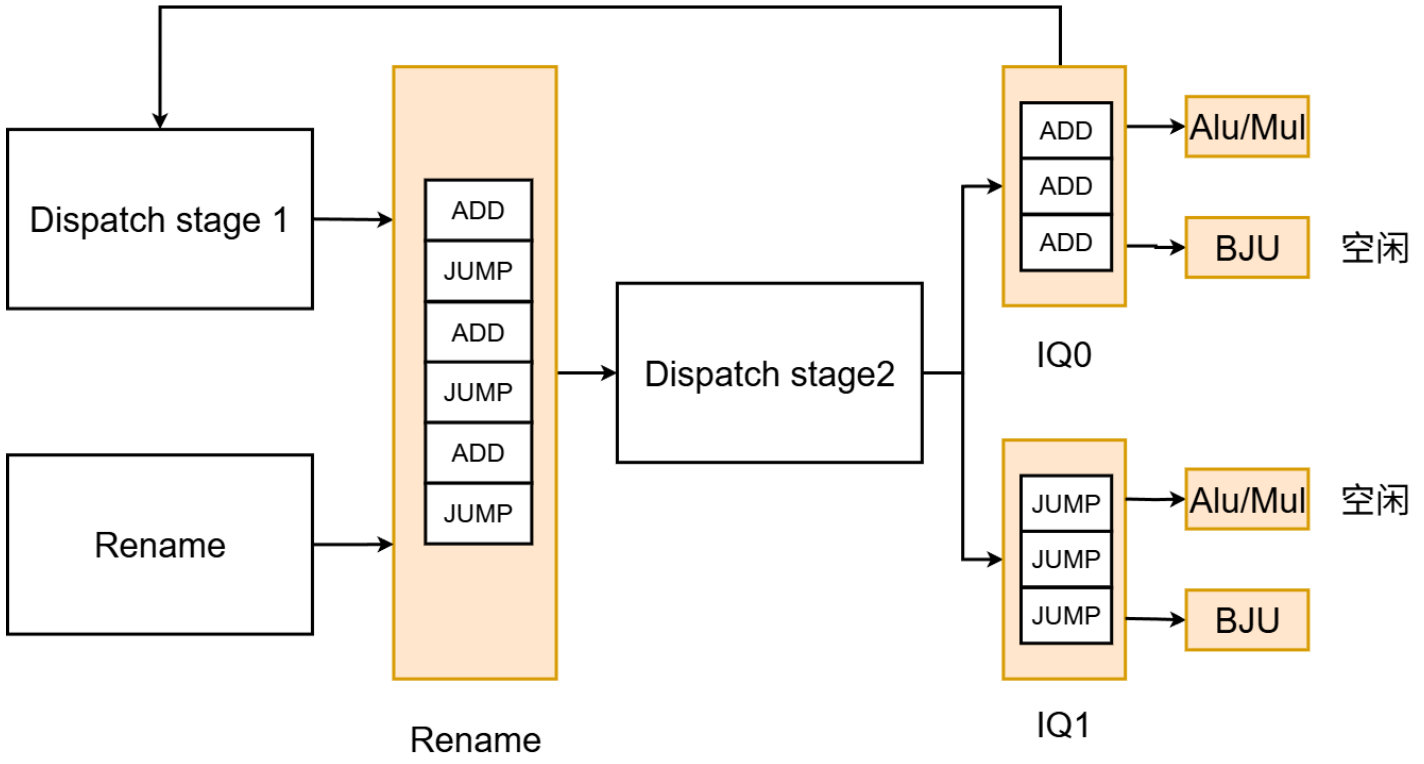
昆明湖 v2 分派算法 —— 实现

- 目标
  - 删除发射队列缓存，重命名结果经分派逻辑直通发射队列
  - 在主流水线**减少一拍**
  - 实现**更均衡的**分派算法
- 技术方案
  - 重构分派算法
    - 发射队列容量 + 轮询
  - 重组分派流水
    - 前移至与Rename并行



昆明湖 v2 分派算法 —— 实现

- 统计发射队列容量
  - 仅统计发射队列总体指令数，可能造成功能单元流水线空闲
  - 按**出队类型**统计指令数



昆明湖 v2 分派算法 —— stage 1

- 对发射队列负载全排序
  - 按指令类型生成比较矩阵
  - 规约求和得到全排序
- 按全排序结果生成待选分派目的序列
  - 轮询
- 统计各类型指令待分派序列

	IQ0_port0	IQ1_port0	IQ2_port0	IQ3_port0
IQ0_port0	0	**deq0<deq1**	deq0<deq2	deq0<deq3
IQ1_port0	~(0,1)	**0**	deq1<deq2	deq1<deq3
IQ2_port0	~(0,2)	**~(1,2)**	0	deq2<deq3
IQ3_port0	~(0,3)	**~(1,3)**	~(2,3)	0

按列规约求和

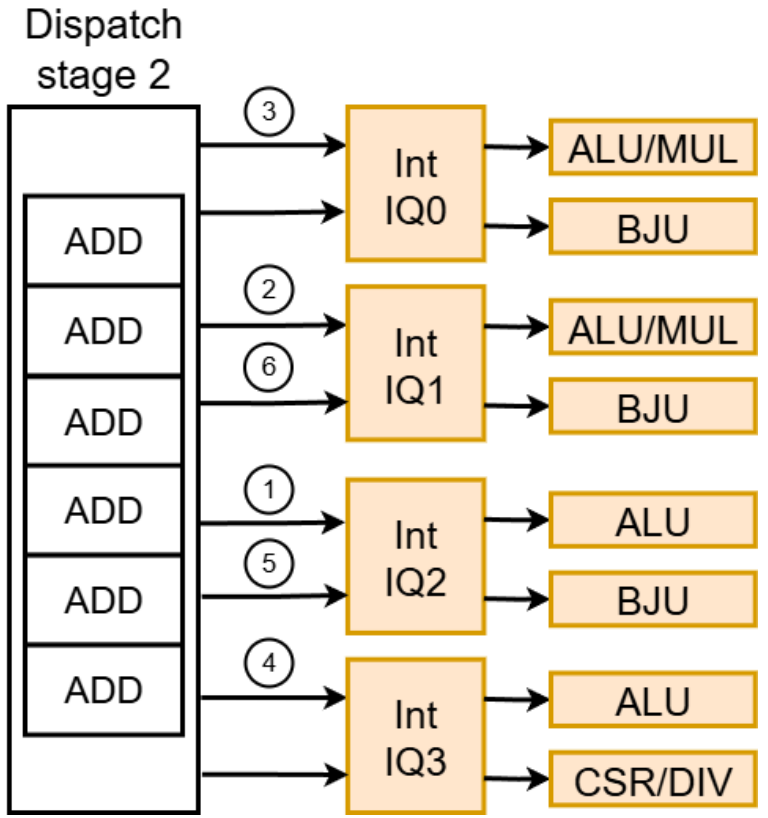
priority	2	1	0	3

容量排序

同类型指令	目的IQ			
0	0	0	1	0
1	0	1	0	0
2	1	0	0	0
3	0	0	0	1

昆明湖 v2 分派算法 —— stage 2

- 按优先级序列轮询展开
  - 展开至发射宽度
  - 待选指令根据优先级编码器选择 IQ
- 边界情况处理
  - 待选指令数超过目的IQ数
    - 使用第二个IQ入口
  - IQ被选数超过入队端口数
    - 能发先发，其余精准阻塞
  - 被选IQ满
    - 能发先发，其余精准阻塞



同类型指令	IQ0	IQ1	IQ2	IQ3
0	0	0	1	0
1	0	1	0	0
2	1	0	0	0
3	0	0	0	1
4	0	0	1	0
5	0	1	0	0

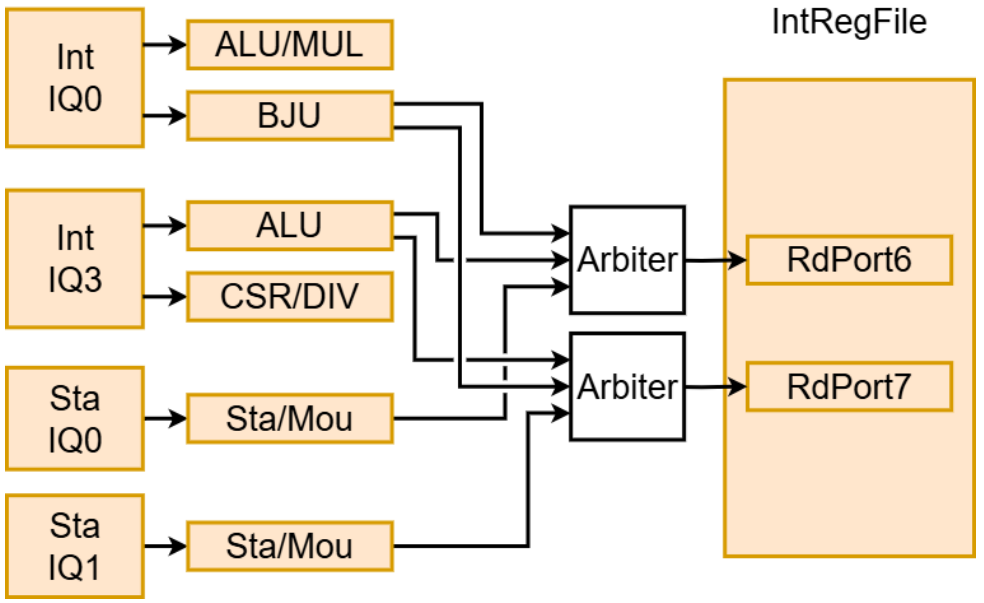
目录

- 昆明湖 V2 后端功能设计演进
  - RVA23 Profile
  - RISC-V 高级中断架构
  - RISC-V E-Trace 拓展
- **昆明湖 V2 后端性能设计演进**
  - 分派阶段优化
  - **寄存器堆缓存**
- 昆明湖 V3 下阶段优化点

寄存器堆缓存 —— 动机

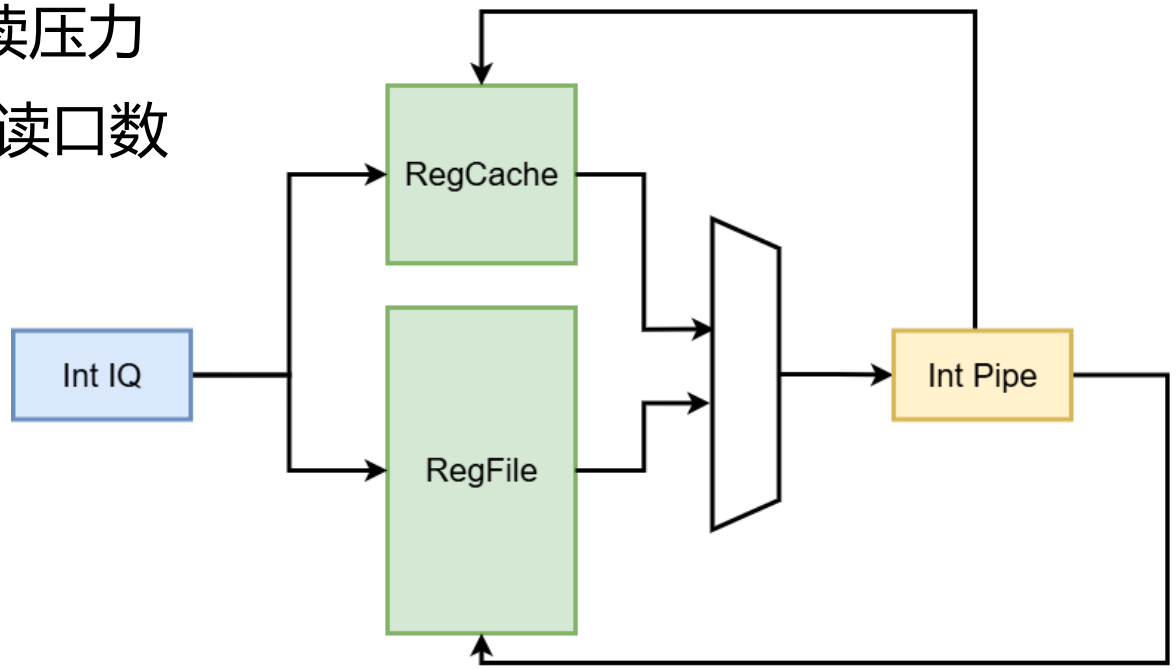
- 寄存器堆并发读口需求量大
  - 功能流水线吞吐量增大读口需求
  - 过多的读口限制唤醒取消链时序
- 现有策略
  - 重新组合功能单元，降低读口需求
  - 功能单元跨 IQ 共用读口
    - 冲突时优先级仲裁
- 如何进一步降低寄存器堆读口数
  - 寄存器堆缓存

RegFile	Int RegFile	Fp RegFile	Vec RegFile
项数	224	192	128
读口	15	11	12
写口	5	5	4



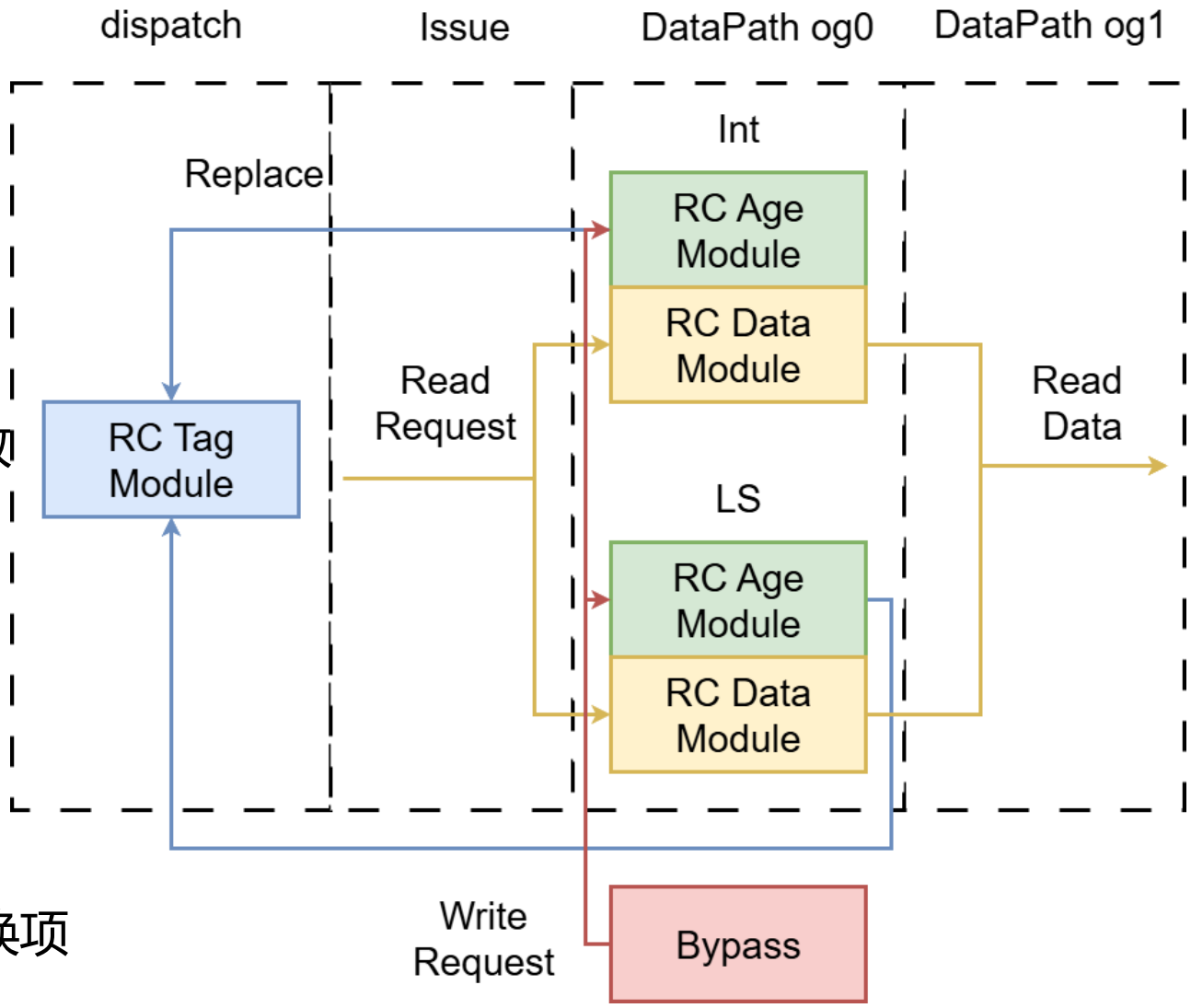
寄存器堆缓存 —— 实现

- 目的
  - 作为寄存器堆子集，减轻 Reg File 并发读压力
  - 在不影响性能的情况下，降低寄存器堆读口数
- 基本功能
  - 存储近期部分功能单元写回结果
  - 承担部分查询寄存器堆请求
  - 目前仅实现整数寄存器堆缓存



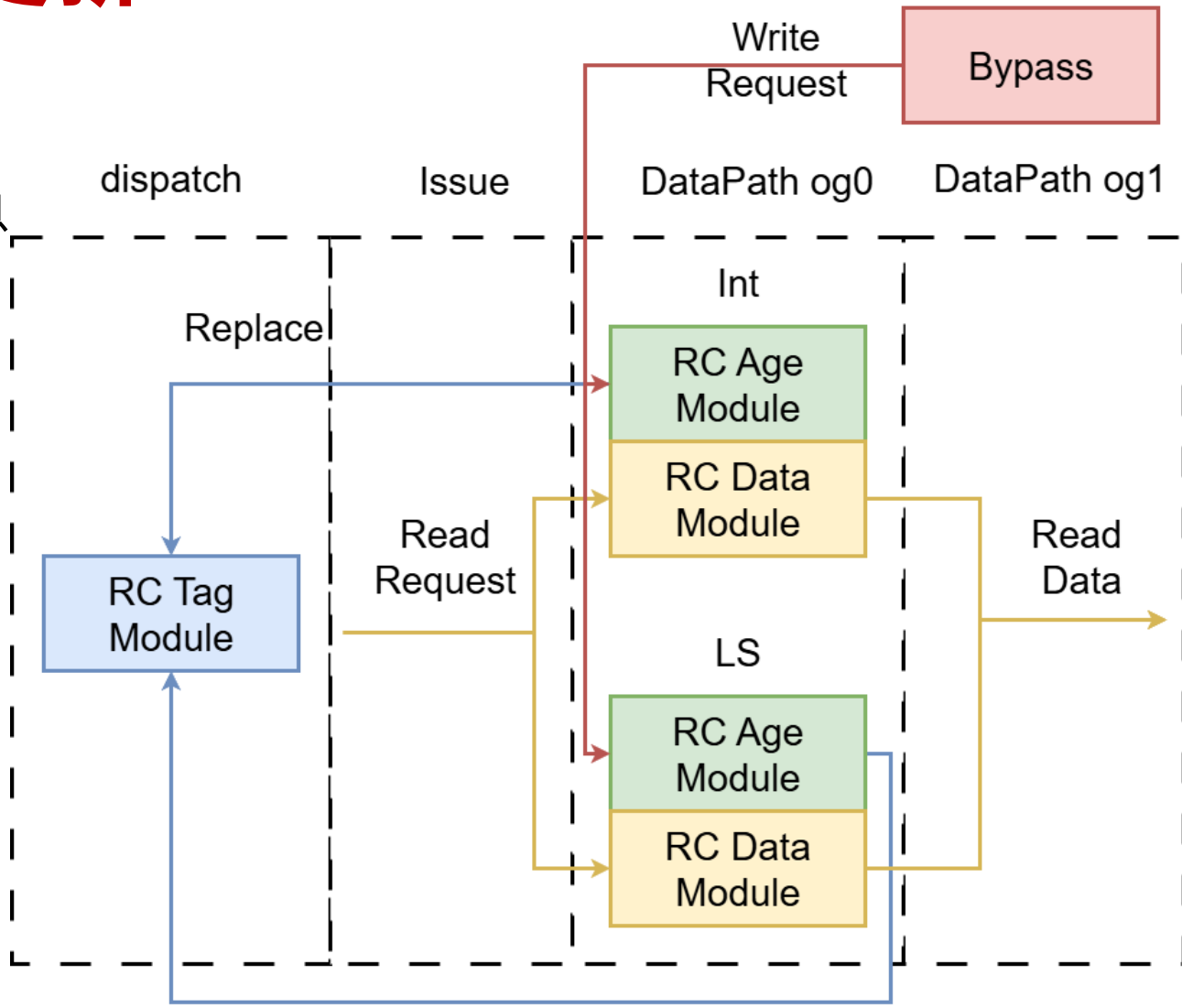
寄存器堆缓存 —— 组织形式

- 数据部分（Data/tag）：
  - 分为整数（16项）、访存（12项）两部分
    - Data 存储数据，tag 存储物理寄存器号
  - 5 bit 全相连索引
  - 共有 整数 16 个读口，访存 7 个读口
  - DATA 位于 DataPath 流水级，与Reg File 一致
  - Tag 位于 Dispatch流水级
  - 写数据来自旁路唤醒信号
- 年龄部分：
  - 每项分别维护 2bit 年龄计数器
  - 每部分维护年龄矩阵
  - 根据年龄矩阵信息，仲裁选出4、3条待替换项



寄存器堆缓存 —— 读取和更新

- 读取流程
  - Dispatch 阶段 根据源操作数物理寄存器信息匹配 Tag，并生成 RC 读地址
  - Datapath og0 阶段 根据读地址发出读请求
  - Datapath og1 阶段 返回读数据
- 更新流程
  - Tag 更新
    - IQ 选出发射后，根据推测唤醒信息替换被年龄矩阵选出的项
  - Data更新
    - 唤醒发出与唤醒数据存在3拍间隔
    - 根据唤醒旁路信息更新



寄存器堆缓存 —— 收益

- 在原寄存器堆读口配置下，添加寄存器堆缓存
  - SPEC 2006 : INT +2.48%
  - SPEC 2006: FP +0.39%

- 在不影响性能前提下，成功将整数寄存器堆读口缩减

RegFile	Int RegFile	Fp RegFile	Vec RegFile
项数	224	192	128
读口	15 -> 11	11	12
写口	5	5	4

目录

- 昆明湖 V2 后端功能设计演进
  - RVA23 Profile
  - RISC-V 高级中断架构
  - RISC-V E-Trace 拓展
- 昆明湖 V2 后端性能设计演进
  - 分派阶段优化
  - 寄存器堆缓存
- 昆明湖 V3 下阶段优化点

昆明湖 V3 标量优化点

- ROB 压缩优化
  - 提高 ROB 等效容量
  - Tradeoff: ROB 表项大小与 ROB表项数目
- 唤醒取消链优化
  - 推测唤醒取代写回唤醒
- 功能单元重构
- 寄存器提前释放
  - 提高物理寄存器利用率
- 指令融合分析
  - 分析热点融合 pattern

……(to be continued)

昆明湖 V3 向量优化点

- 复杂向量指令采用专用功能单元取代复杂译码拆分
  - 新版 gather 单元可将吞吐 x16, 延迟 /8
  - 移除临时逻辑寄存器，提升向量调度窗口
- 优化向量 uop 拆分
  - 降低 uop 拆分的最大数量
  - 降低非向量资源占用
- 添加 uop 间的推测唤醒
  - 对非谓词 uop 间做 back-to-back 唤醒
- 优化向量连续访存指令
- 优化向量译码
  - 减少一级向量译码流水线

……(to be continued)

敬请批评指正！