



香山昆明湖后端流水线的 设计演进

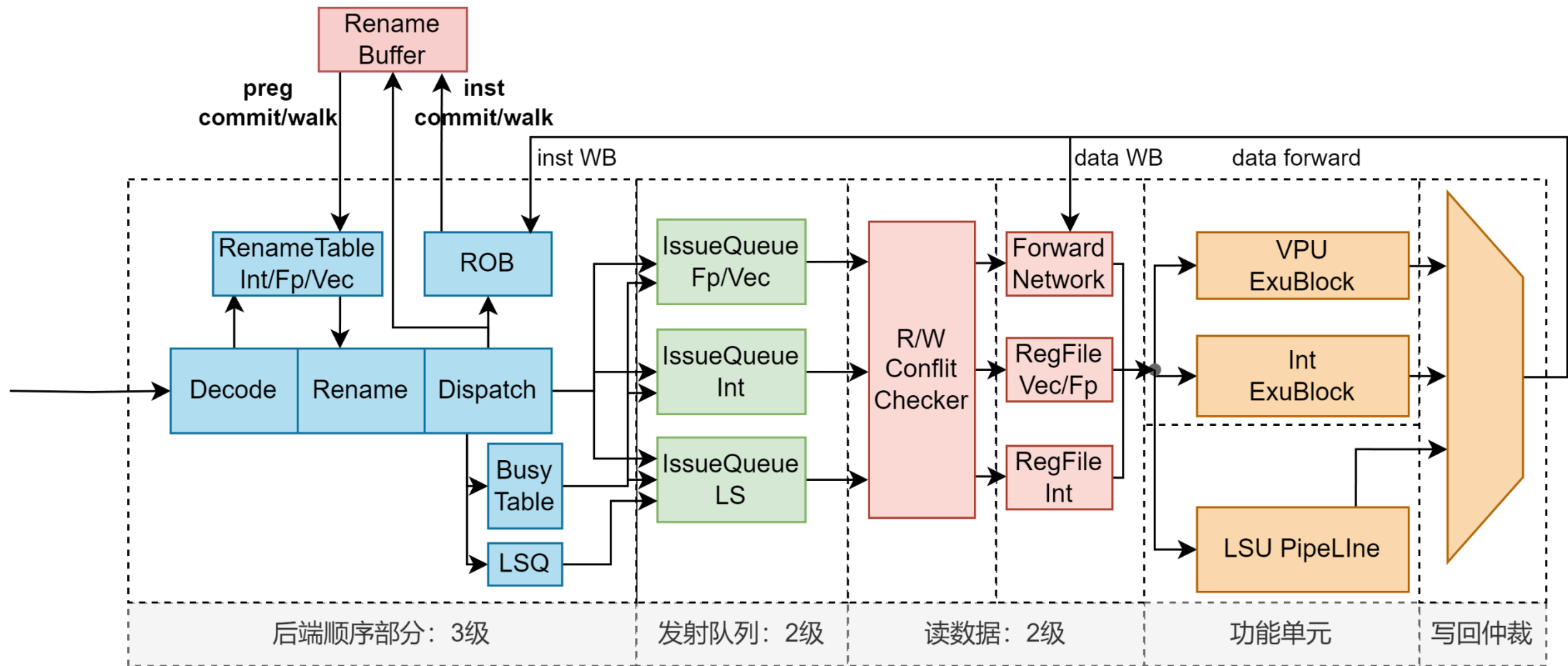
张紫飞¹ 胡轩¹ 唐浩晋¹ 何逸飞² 肖飞豹² 付丹阳³ 张娄峰⁴
张梓悦¹ 曹泽文⁵ 贾志杰¹ 刘泽昊¹

¹中国科学院计算技术研究所 ²北京开源芯片研究院

³大连理工大学 ⁴北京大学 ⁵中国科学院微电子研究所

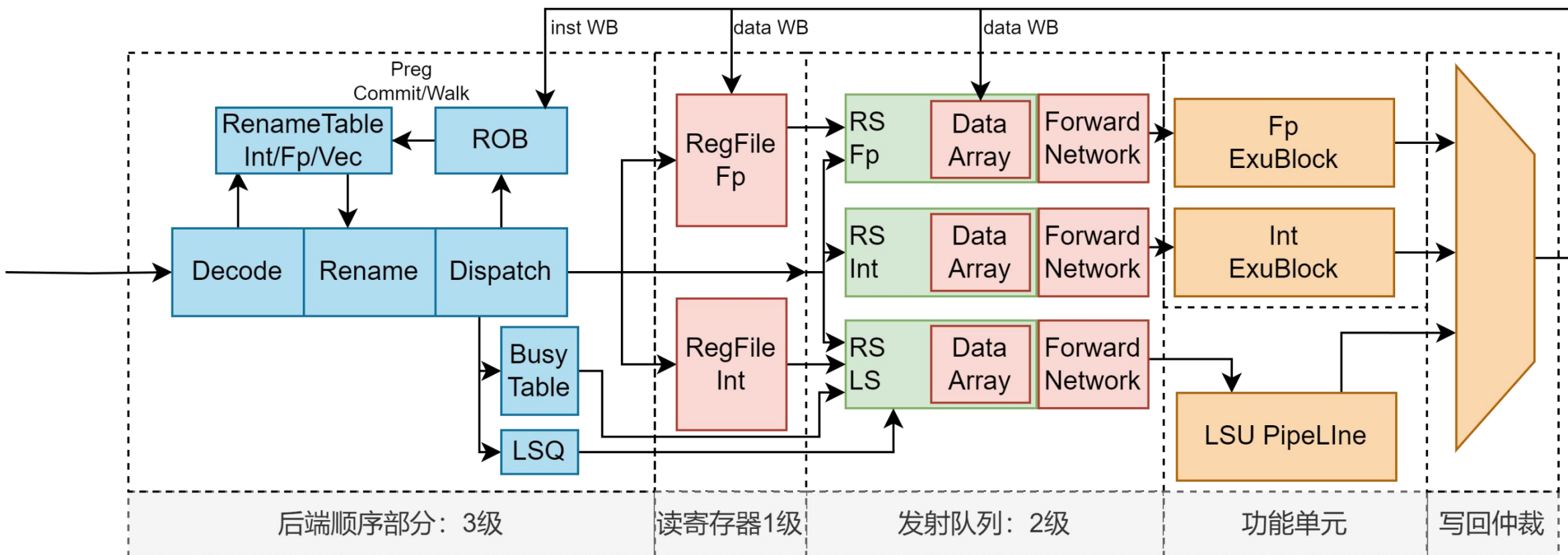
2023年8月24日@第三届 RISC-V 中国峰会

昆明湖后端架构总览



昆明湖后端流水级架构简图

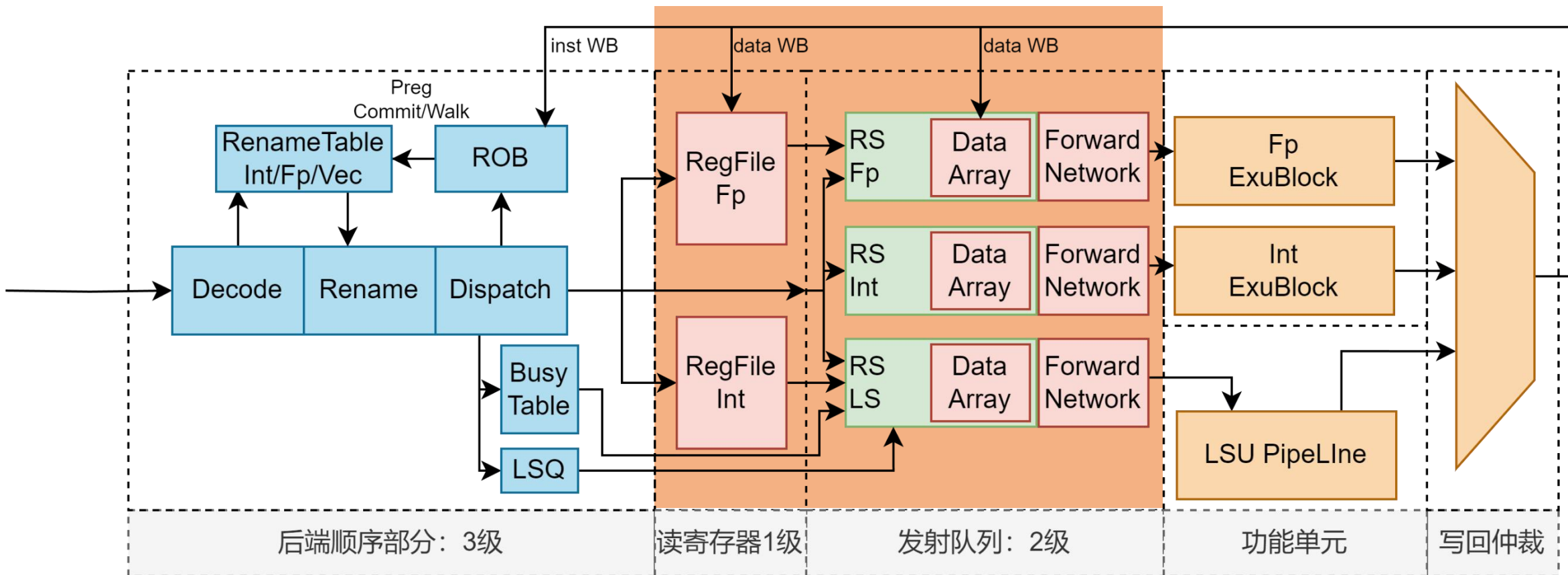
南湖后端架构总览



南湖后端流水级架构简图



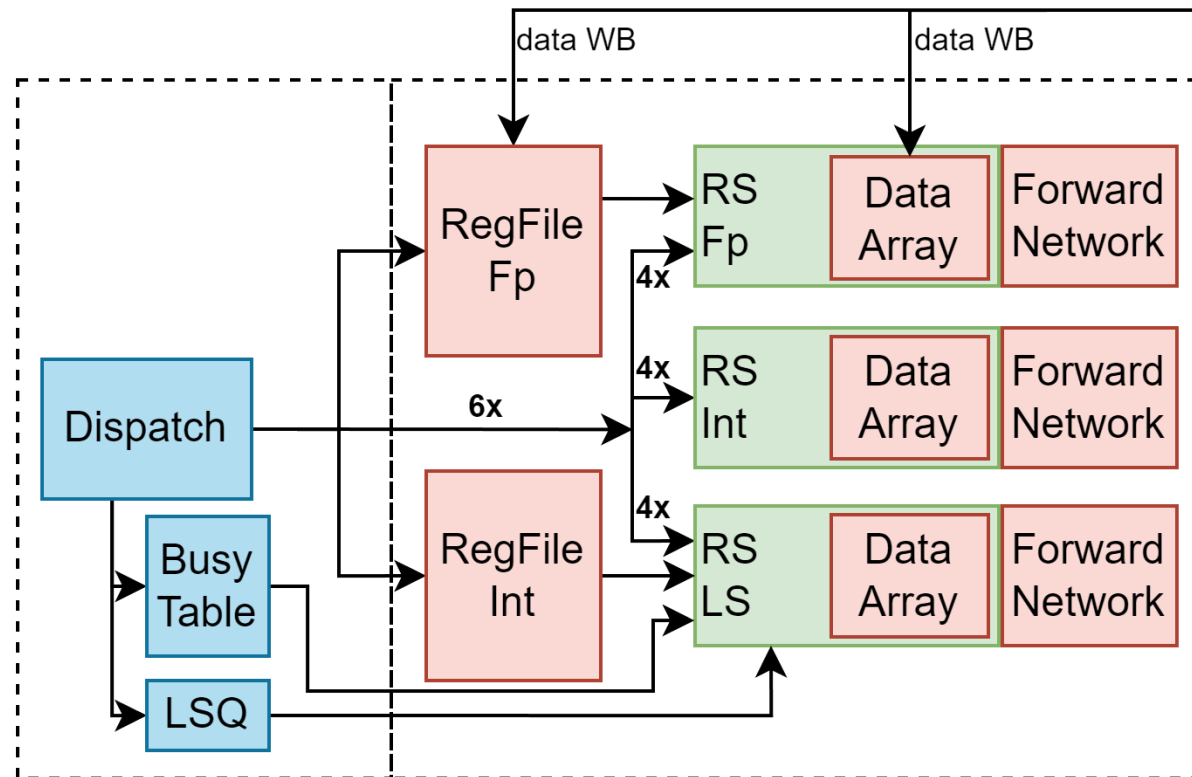
南湖后端流水线主要瓶颈



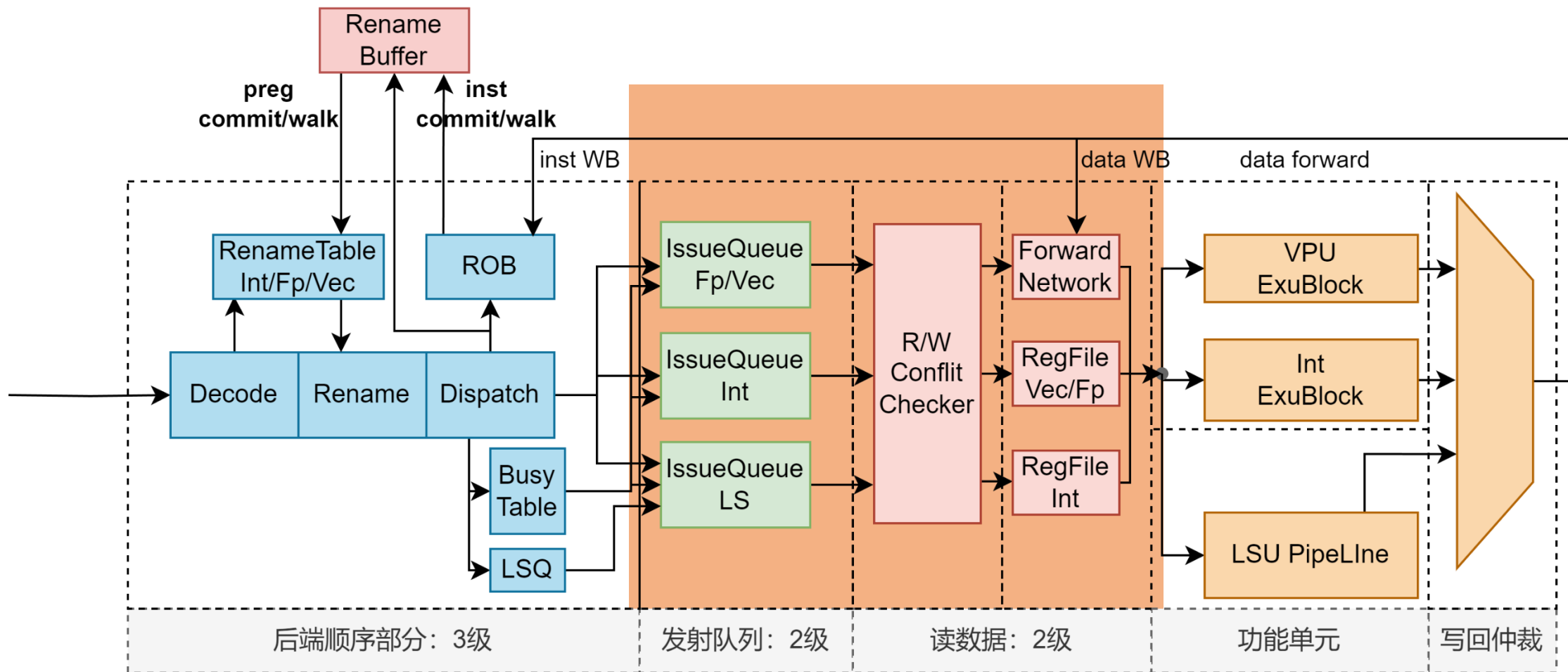
南湖后端流水线级架构简图

发射前读寄存器堆

- 寄存器读口数量限制 Dispatch 宽度
 - Decode, Rename, Commit 宽度为 6
 - Int/Fp/Mem Dispatch 宽度各为 4
 - **瓶颈无法突破，但可以后移**
- DataArray 面积较大
 - 以 32 项 AluRS 为例
 - 保存 4k bits 数据 (64x2x32)
 - 每个 DataArray 2 读 10 写
 - 时序较差



改进1：发射后读寄存器堆



昆明湖后端流水级架构简图

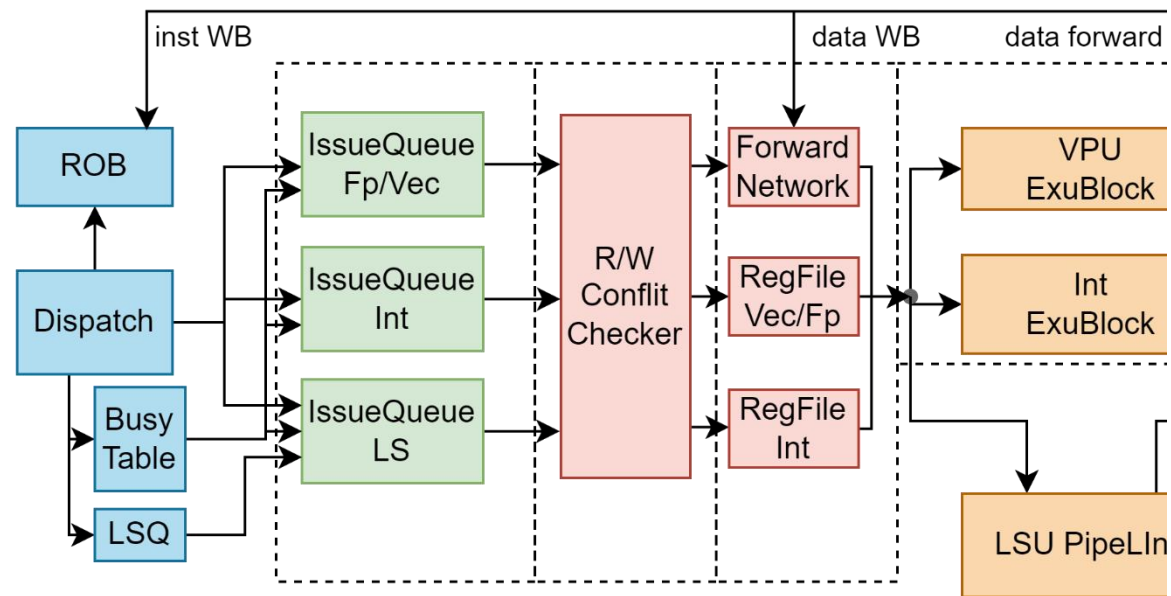
发射后读寄存器堆

• 优点

- Dispatch 阶段的**瓶颈后移**到指令发射后
- 降低保留站存储开销，**减小面积**
- 降低延迟，**提升发射队列容量**





• 新的问题

- 功能单元多导致**寄存器读口数量多**
- 发射队列到功能单元之间**流水级数增加**



昆明湖发射后读寄存器堆

发射后读的配套改进

- 组合功能单元  减小寄存器读口需求
- 推测唤醒及取消机制  降低流水级增加的代价
- 写回状态表预订写回端口  使推测唤醒更加准确
- 寄存器读口仲裁  解决寄存器读口冲突问题

发射后读的配套改进

- 组合功能单元



减小寄存器读口需求

- 推测唤醒及取消机制



降低流水级增加的代价

- 写回状态表预订写回端口



使推测唤醒更加准确

- 寄存器读口仲裁



解决寄存器读口冲突问题

改进1.1: 重新组合功能单元

- 弃用南湖的功能单元组合

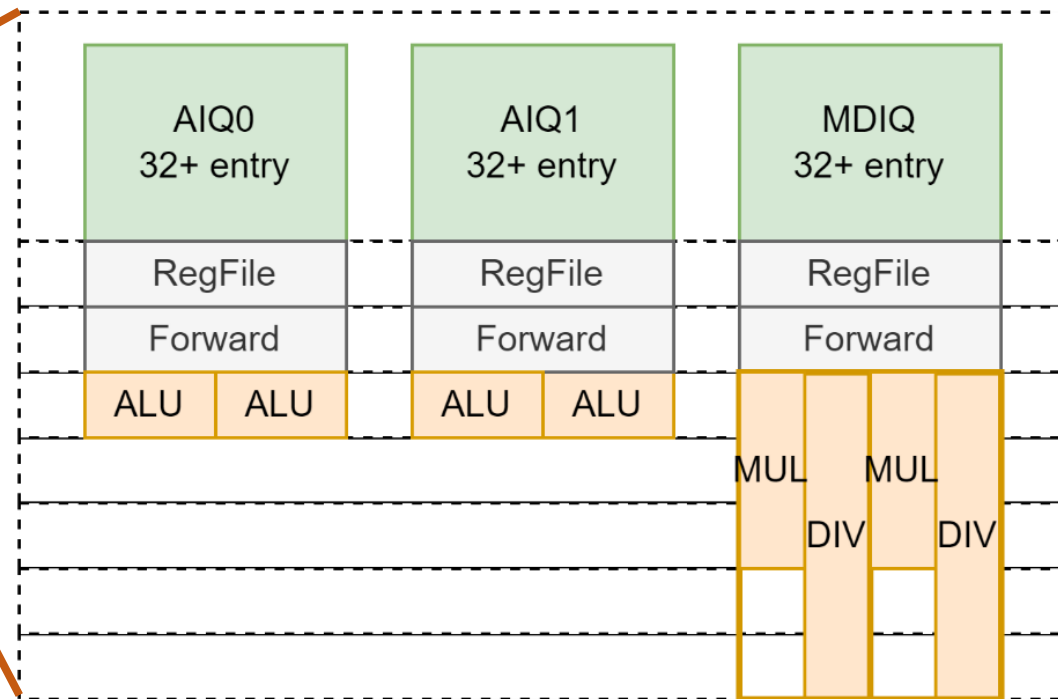
- $ALU * 4 + MDU * 2$, 12个读口

- 昆明湖重新规划功能单元分组

- $(ALU + MUL) * 2 + (ALU + DIV) * 2$, 8个读口

- 重新组合功能单元让读口需求下降

- 性能几乎不受影响

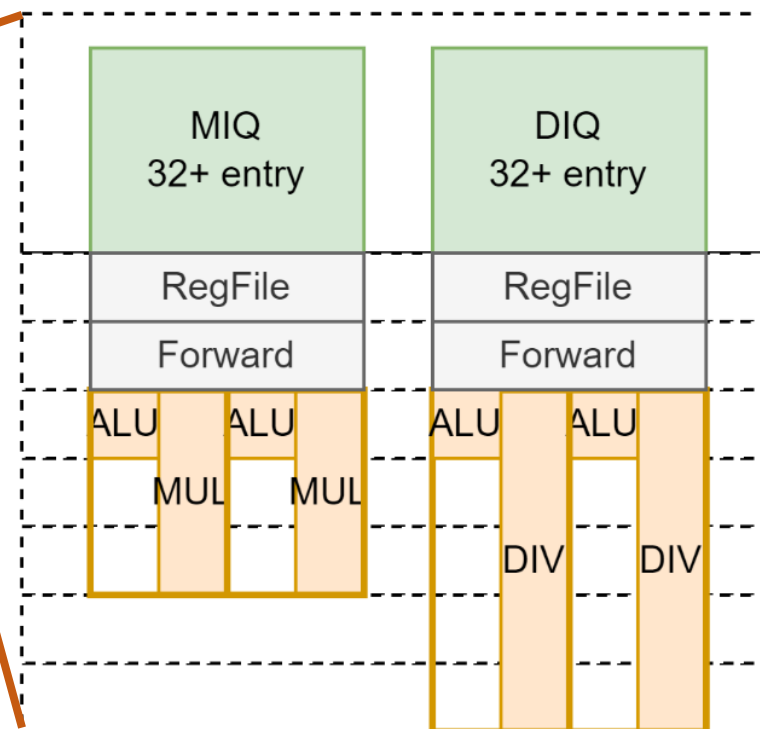


改进1.1: 重新组合功能单元

- 弃用南湖的功能单元组合
 - $ALU*4 + MDU*2$, 12个读口

• 昆明湖重新规划功能单元分组

- $(ALU+MUL)*2 + (ALU+DIV)*2$, 8个读口
- 重新组合功能单元让读口需求下降
 - 性能几乎不受影响



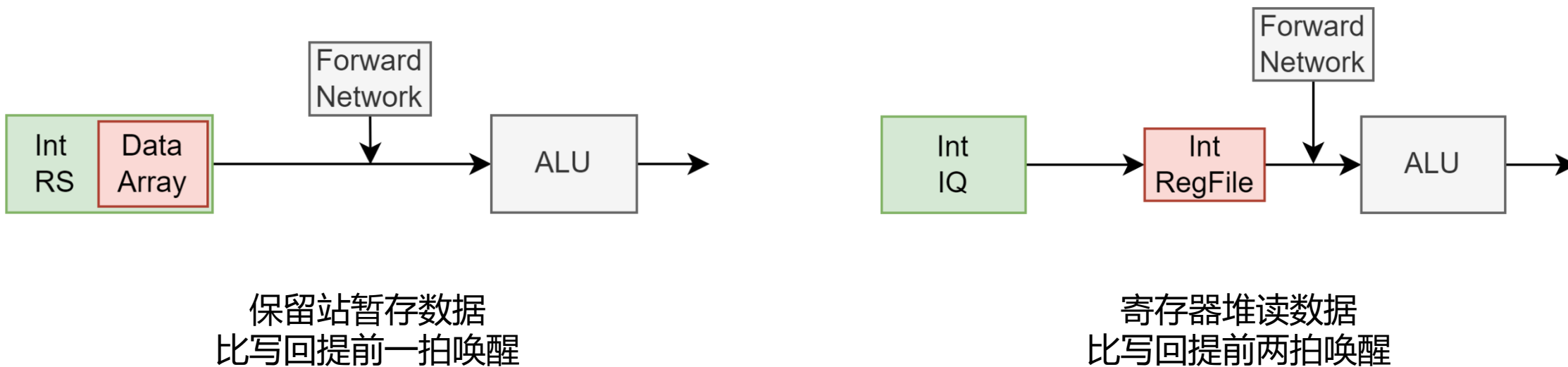
改进1.1: 重新组合功能单元

- 弃用南湖的功能单元组合
 - $ALU*4 + MDU*2$, 12个读口
- 昆明湖重新规划功能单元分组
 - $(ALU+MUL)*2 + (ALU+DIV)*2$, 8个读口
- 重新组合功能单元让读口需求下降
 - **性能几乎不受影响**

改进1.2: 推测唤醒和取消机制

- 需要推测唤醒的原因

- 发射后读寄存器堆没有暂存数据的位置
- 指令发射到执行多了一拍，需要比写回**提前两拍**唤醒下一条指令

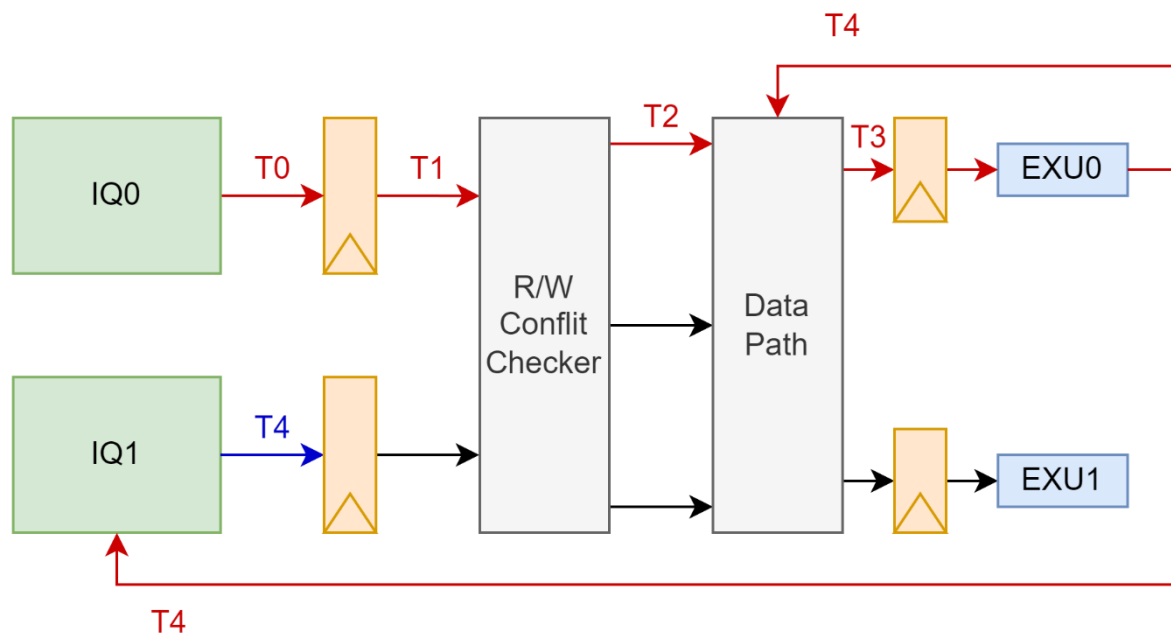


改进1.2: 推测唤醒和取消机制

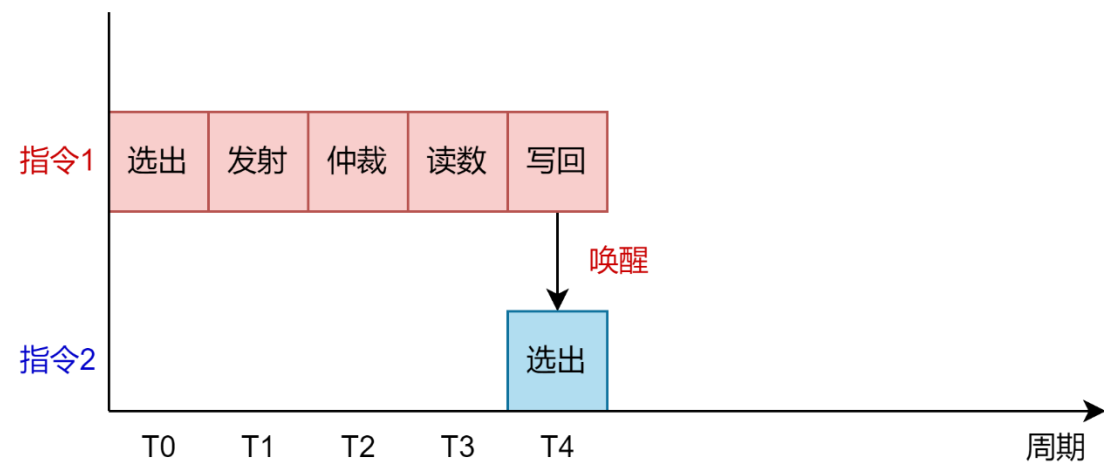
- 需要取消机制的原因
 - 唤醒源指令可能执行失败
 - 没抢到数据读端口
 - Load 指令 TLB miss 和 cache miss 等

没有推测唤醒

- 指令 1 写回时唤醒指令 2



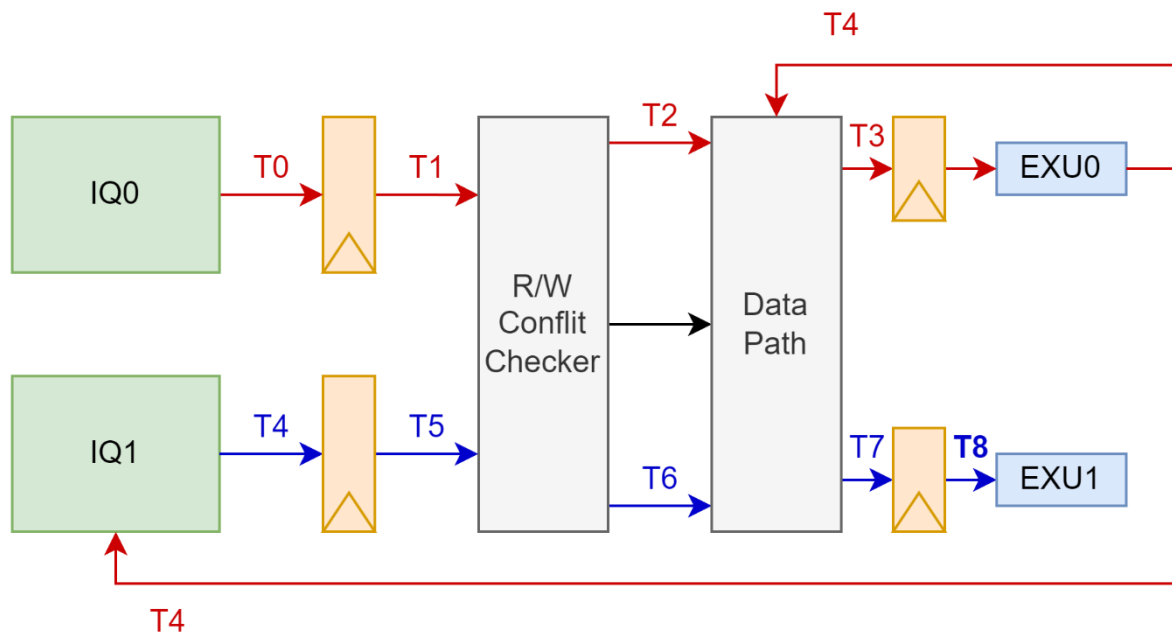
没有推测唤醒的示意图



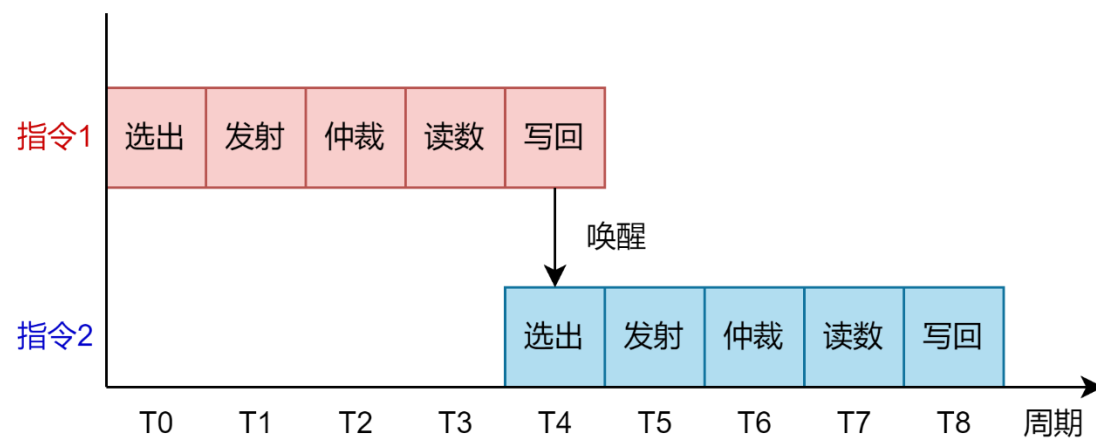
没有推测唤醒的时空图

没有推测唤醒

- 指令 1 写回时唤醒指令 2
- 流水线气泡很多



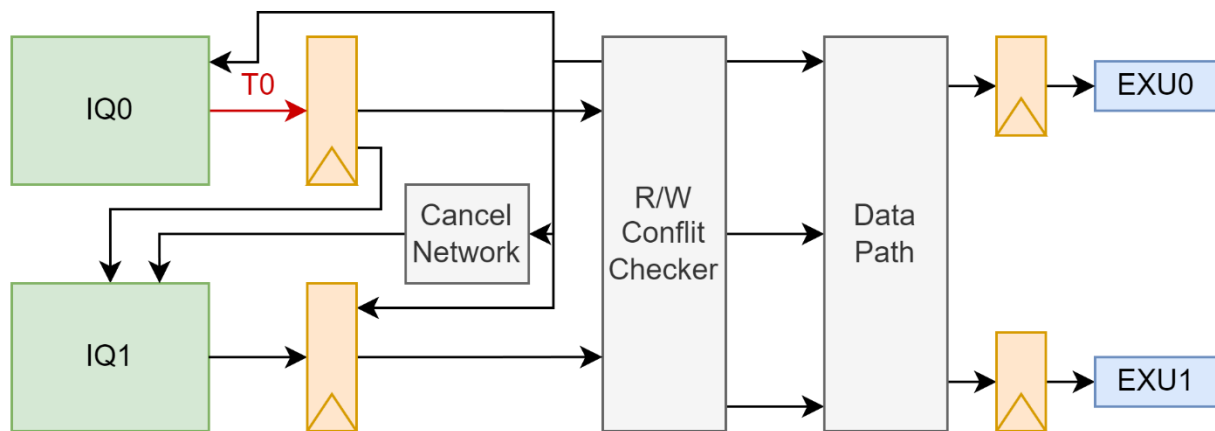
没有推测唤醒的示意图



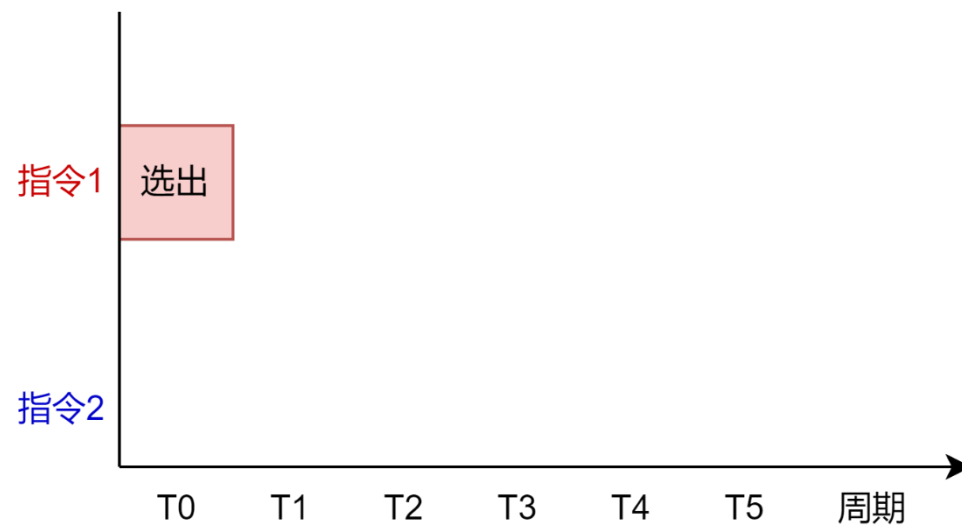
没有推测唤醒的时空图

实现推测唤醒

- 发射队列选出指令 1



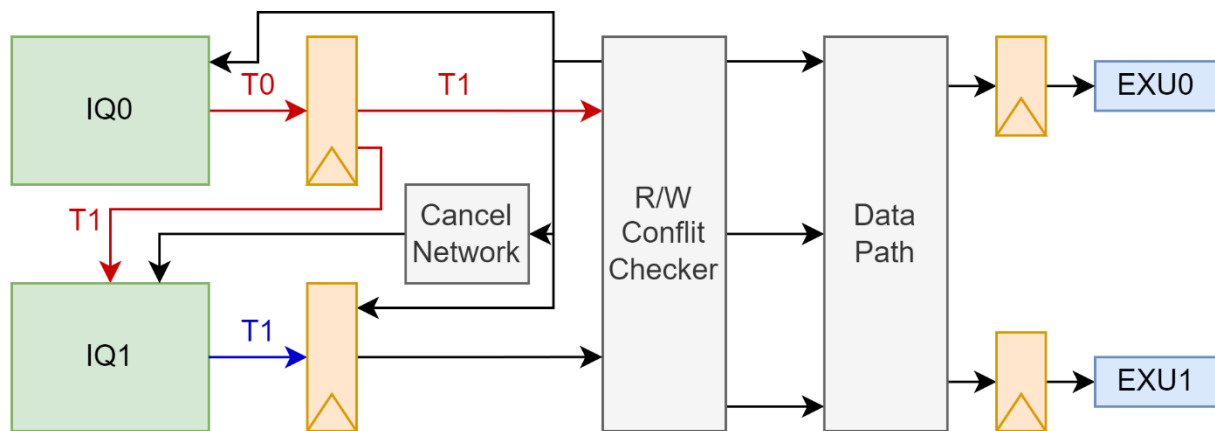
推测唤醒示意图



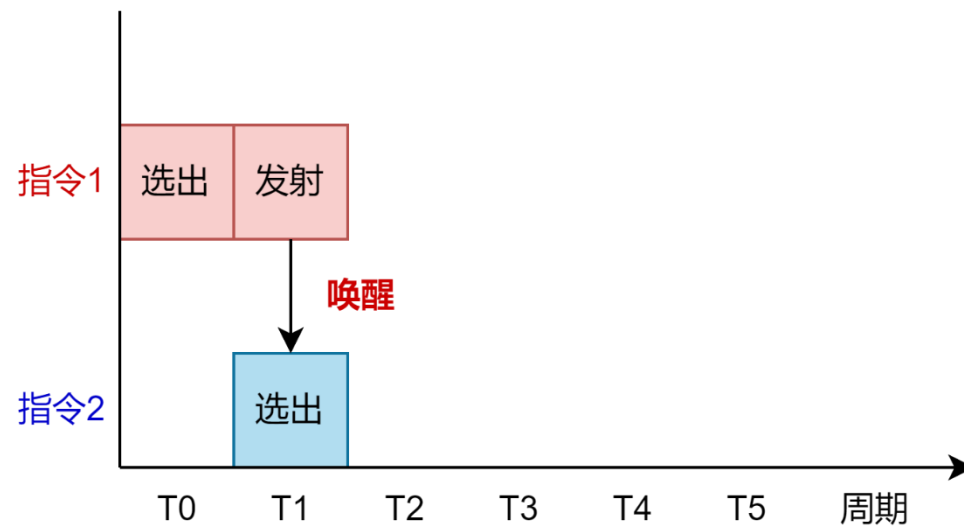
推测唤醒时空图

实现推测唤醒

- 指令 1 发射, 推测唤醒指令 2
- 发射队列选出 指令 2



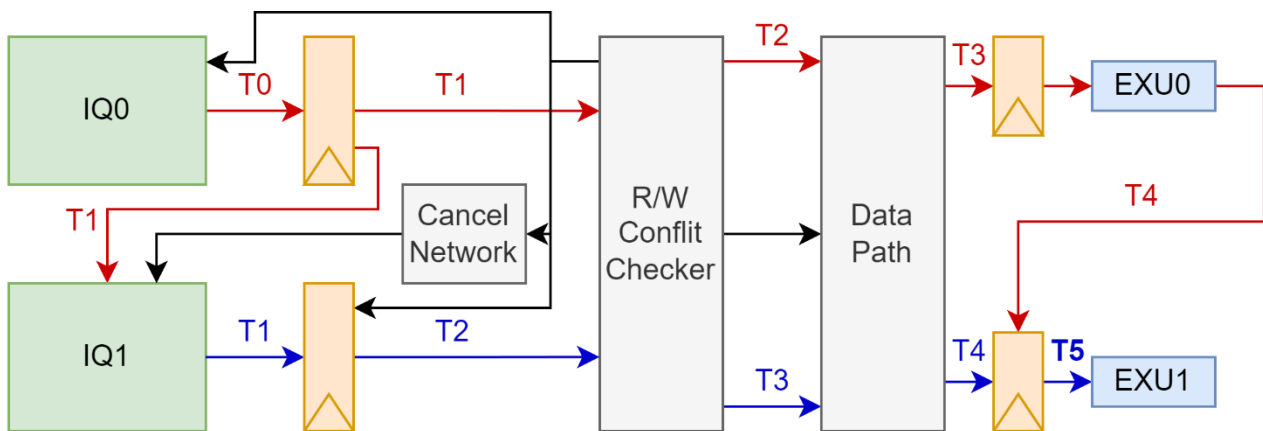
推测唤醒示意图



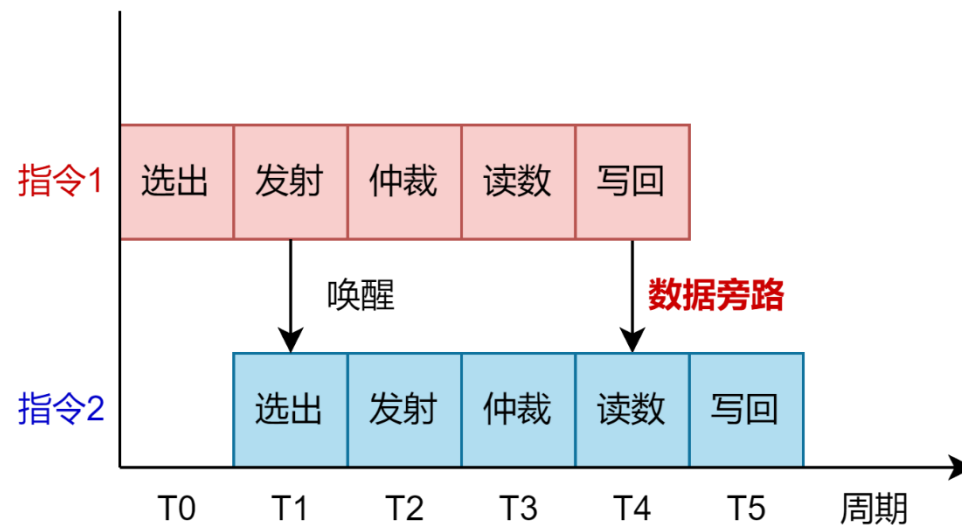
推测唤醒时空图

实现推测唤醒

- 指令 2 接收指令 1 的旁路数据
- 实现 back-to-back 唤醒，降低流水级增加的代价



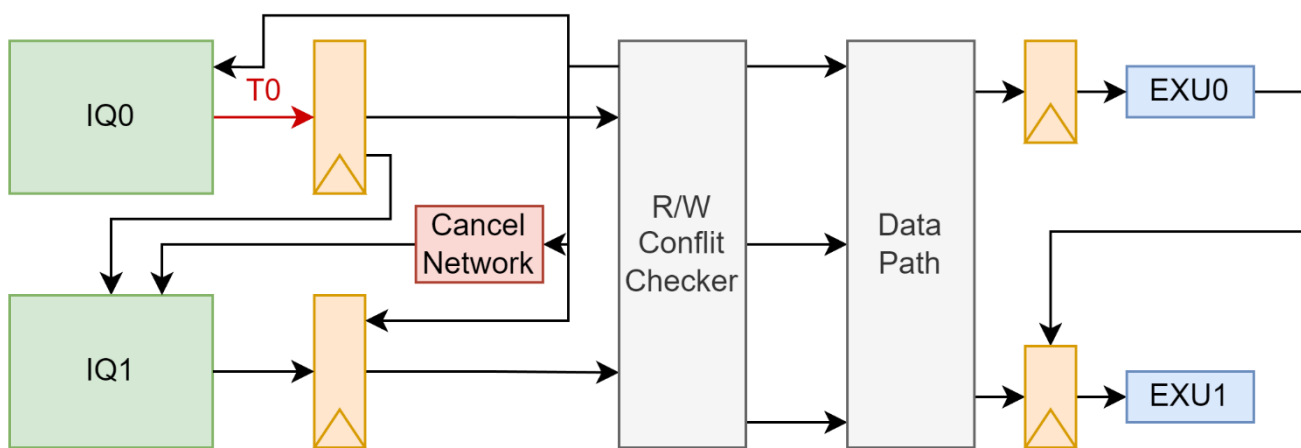
推测唤醒示意图



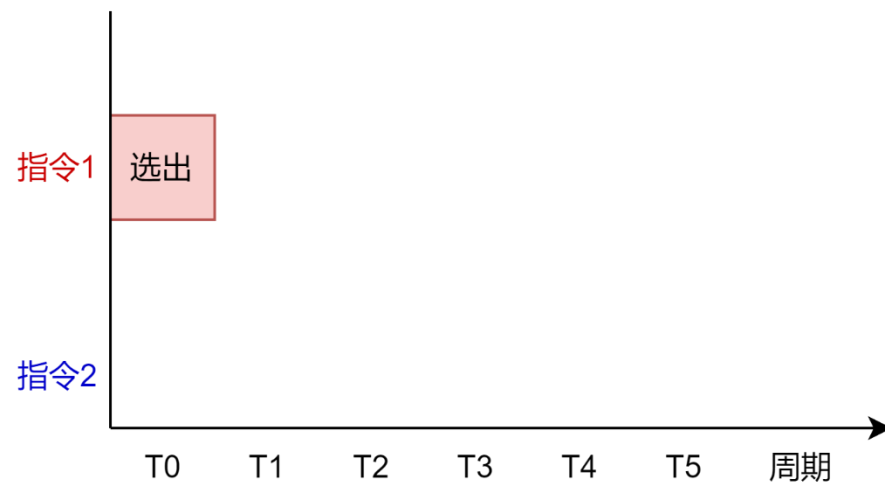
推测唤醒时空图

取消错误的推测唤醒

- 发射队列选出指令 1



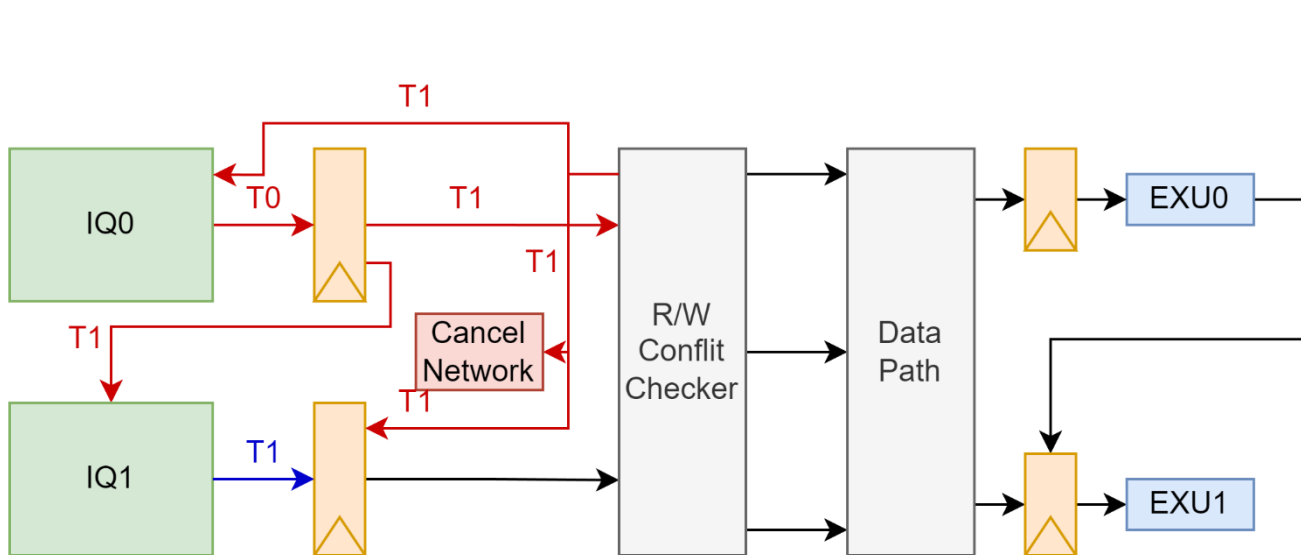
取消推测唤醒示意图



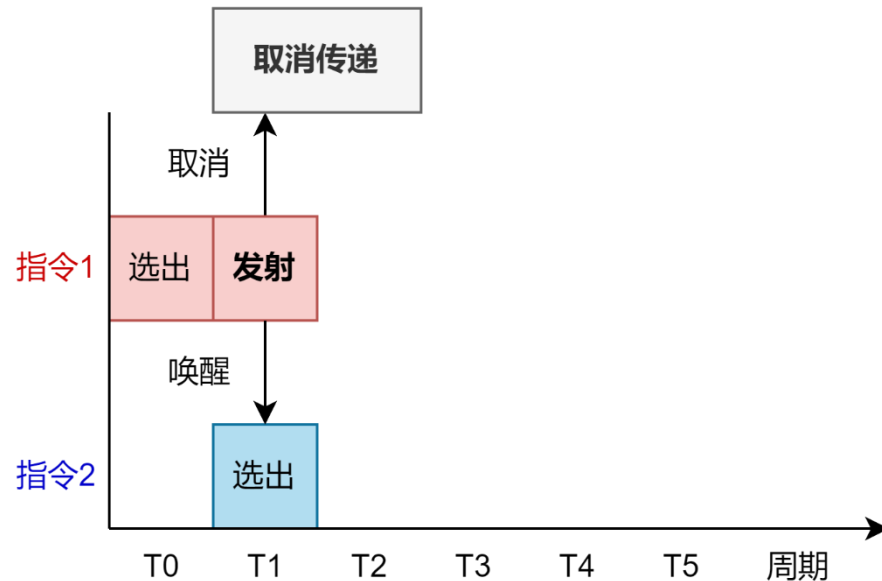
取消推测唤醒时空图

取消错误的推测唤醒

- 指令 1 发射，推测唤醒指令 2，选出指令 2
- 指令 1 仲裁失败，取消自身，取消信息进入取消传递网络



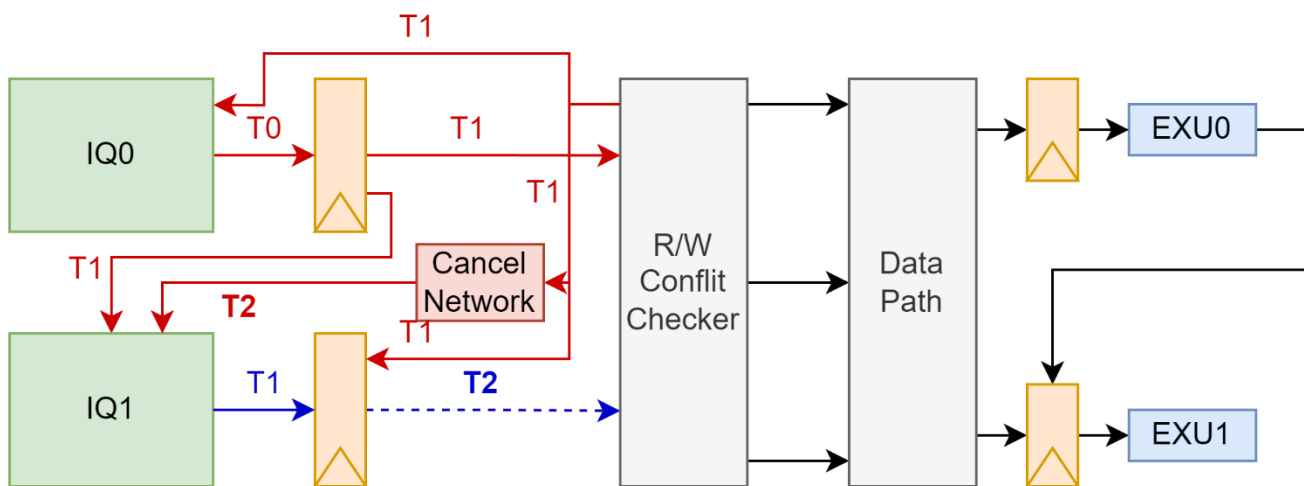
取消推测唤醒示意图



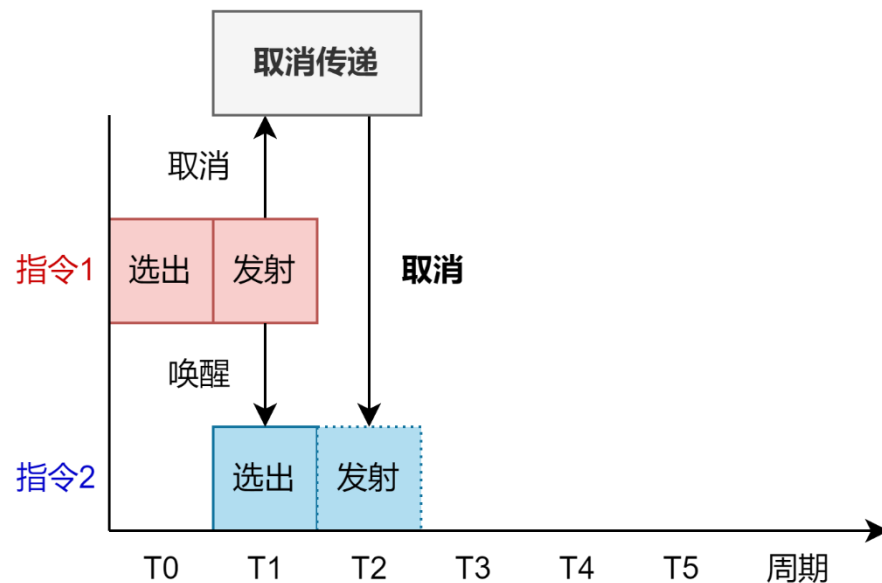
取消推测唤醒时空图

取消错误的推测唤醒

- 指令 2 不进入发射阶段
- 取消传递网络取消指令 2 的操作数的就绪状态

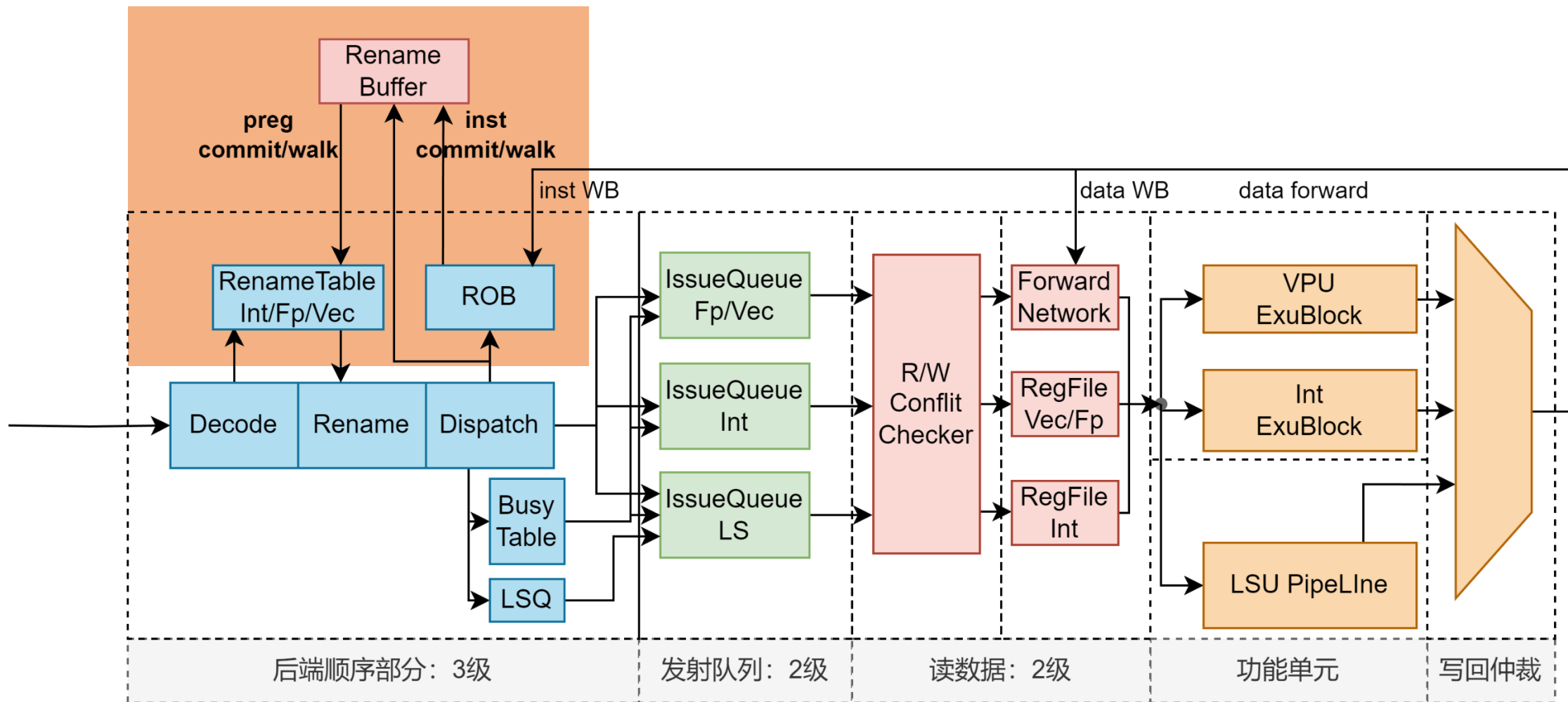


取消推测唤醒示意图



取消推测唤醒时空图

改进2：指令和寄存器提交解耦



昆明湖后端流水级架构简图

指令和寄存器提交解耦——动机

- 问题 1: ROB 中目的寄存器映射对**分支和Store指令是冗余的**
 - 冗余程度 19.6% ~ 38.4%*
 - 需求: 支持 ROB 每项对应 0 个寄存器映射来消除冗余
- 问题 2: 每条机器向量指令**写回至多 8 个寄存器**
 - 需求: 支持 ROB 每项对应 8 个寄存器映射
- 问题 3: 继续提升 ROB 容量来**增加乱序调度窗口**
 - 通过 ROB 压缩提升等效容量
 - 需求: 支持 ROB 每项对应多个运算指令的寄存器映射

* https://www.spec.org/workshops/2007/austin/papers/Performance_Characterization_SPEC_CPU_Benchmarks.pdf

指令和寄存器提交解耦——动机

- 问题 1: ROB 中目的寄存器映射对分支和Store指令是冗余的
 - 冗余程度 19.6% ~ 38.4%*
 - 需求: 支持 ROB 每项对应 0 个寄存器映射
- 问题 2: 每条机器向量指令写回至多 8 个寄存器
 - 需求: 支持 ROB 每项对应 8 个寄存器映射
- 问题 3: 继续提升 ROB 容量来增加乱序调度窗口
 - 通过 ROB 压缩提升等效容量
 - 需求: 支持 ROB 每项对应多个运算指令的寄存器映射

* https://www.spec.org/workshops/2007/austin/papers/Performance_Characterization_SPEC_CPU_Benchmarks.pdf

指令和寄存器提交解耦——动机

- 问题 1: ROB 中目的寄存器映射对分支和Store指令是冗余的
 - 冗余程度 19.6% ~ 38.4%*
 - 需求: 支持 ROB 每项对应 0 个寄存器映射
- 问题 2: 每条机器向量指令写回至多 8 个寄存器
 - 需求: 支持 ROB 每项对应 8 个寄存器映射
- 问题 3: 继续提升 ROB 容量来增加乱序调度窗口
 - 通过 ROB 压缩提升等效容量
 - 需求: 支持 ROB 每项对应多个运算指令的寄存器映射

将指令提交和寄存器提交解耦

* https://www.spec.org/workshops/2007/austin/papers/Performance_Characterization_SPEC_CPU_Benchmarks.pdf

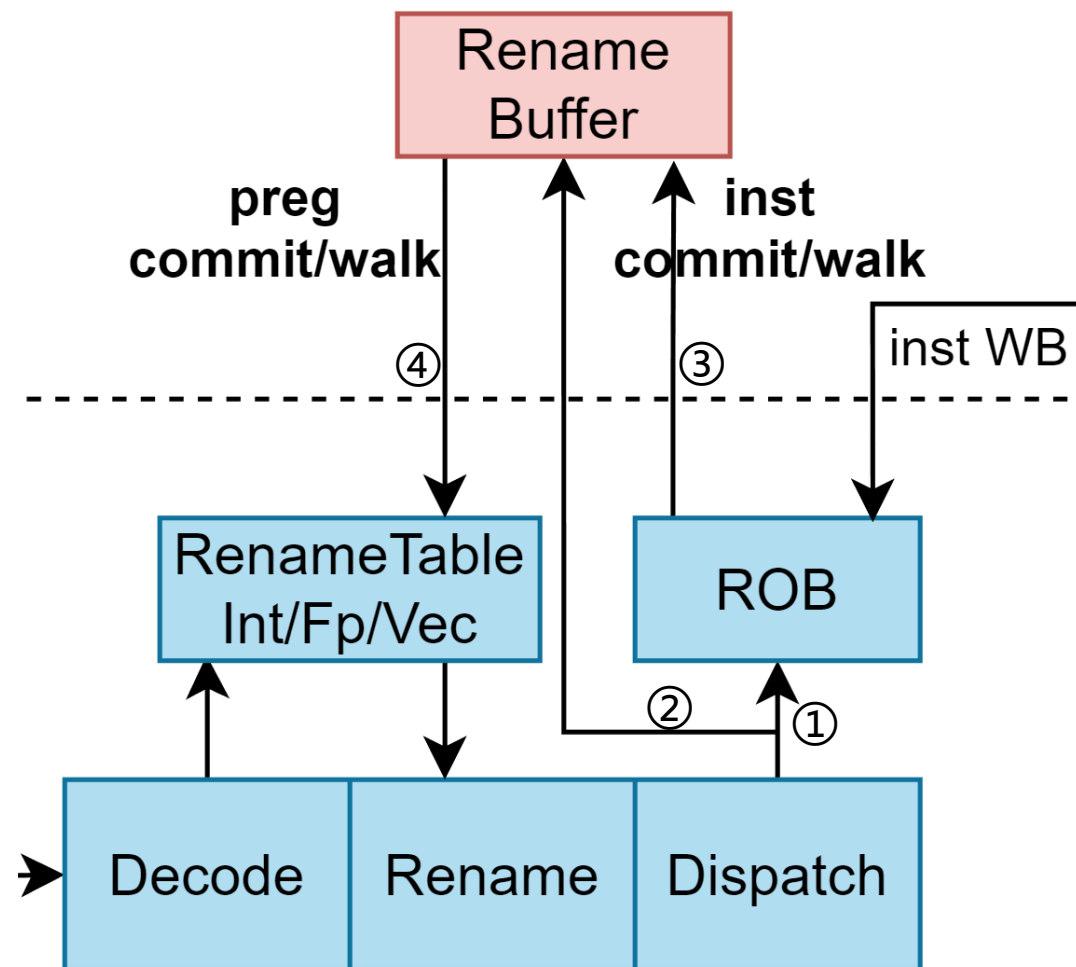
🌟 解耦指令和寄存器提交

• 流程

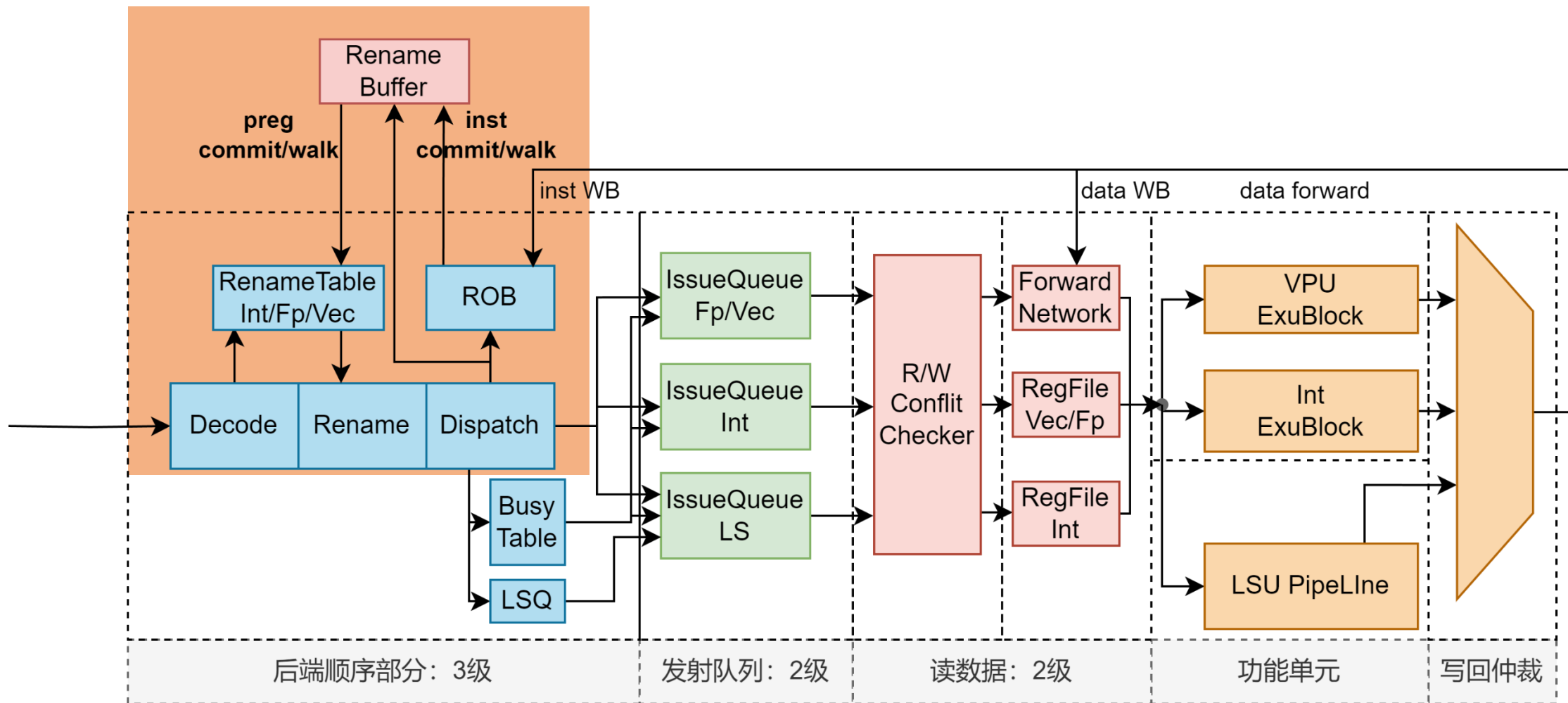
- ① 指令进入 ROB，记录对应的寄存器映射个数
- ② 寄存器映射信息进入 RenameBuffer (RAB)
- ③ 指令提交/回滚，给 RAB 传递寄存器数量信息
- ④ 寄存器提交/回滚，更新 RenameTable (RAT)

• 优点

- ROB entry 瘦身，**优化时序**
- **支持向量指令等的拆分**
- 支持 ROB 压缩，**提高 ROB 等效容量**



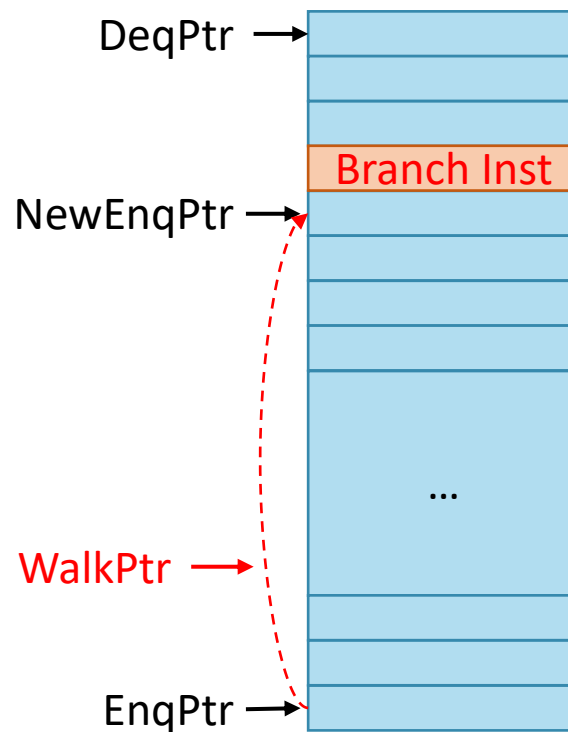
改进3：基于检查点的 RAT 恢复机制



昆明湖后端流水级架构简图

基于检查点的 RAT 恢复机制——动机

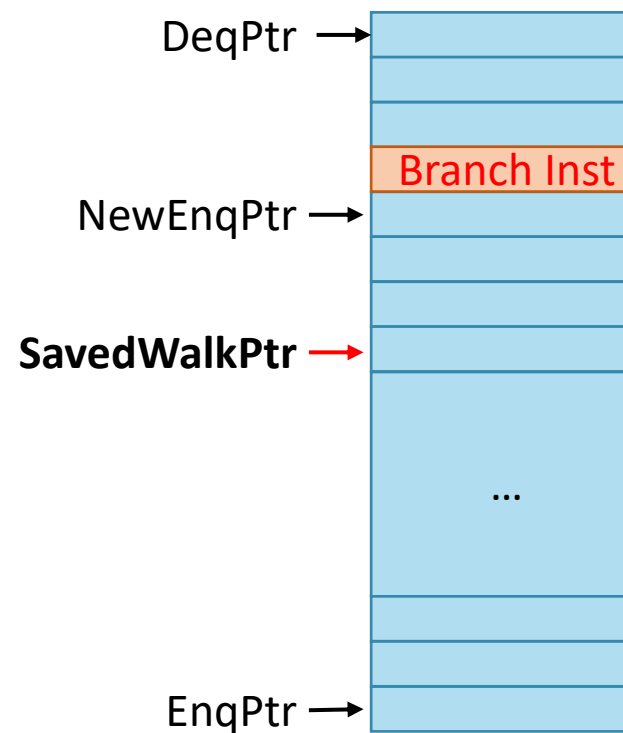
- 误预测和中断异常将导致流水线冲刷
- 恢复重命名表速度受 commit width 限制
- 重定向冲刷指令较多时，**回滚周期数太多**



ROB 重定向的 walk 过程

基于检查点的 RAT 恢复机制——动机

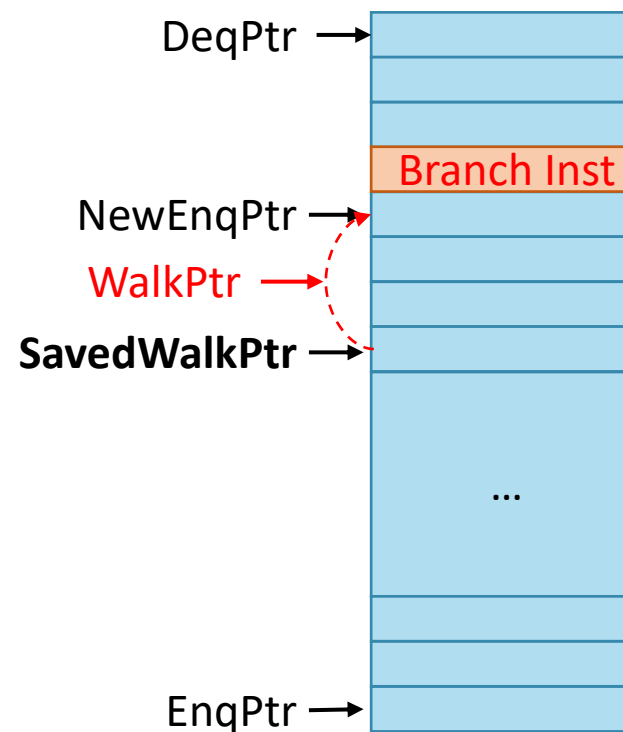
- 误预测和中断异常将导致流水线冲刷
- 恢复重命名表速度受 commit width 限制
- 重定向冲刷指令较多时，回滚周期数太多
- 假如重命名表可以**定期存档**？



ROB 重定向的 walk 过程

基于检查点的 RAT 恢复机制——动机

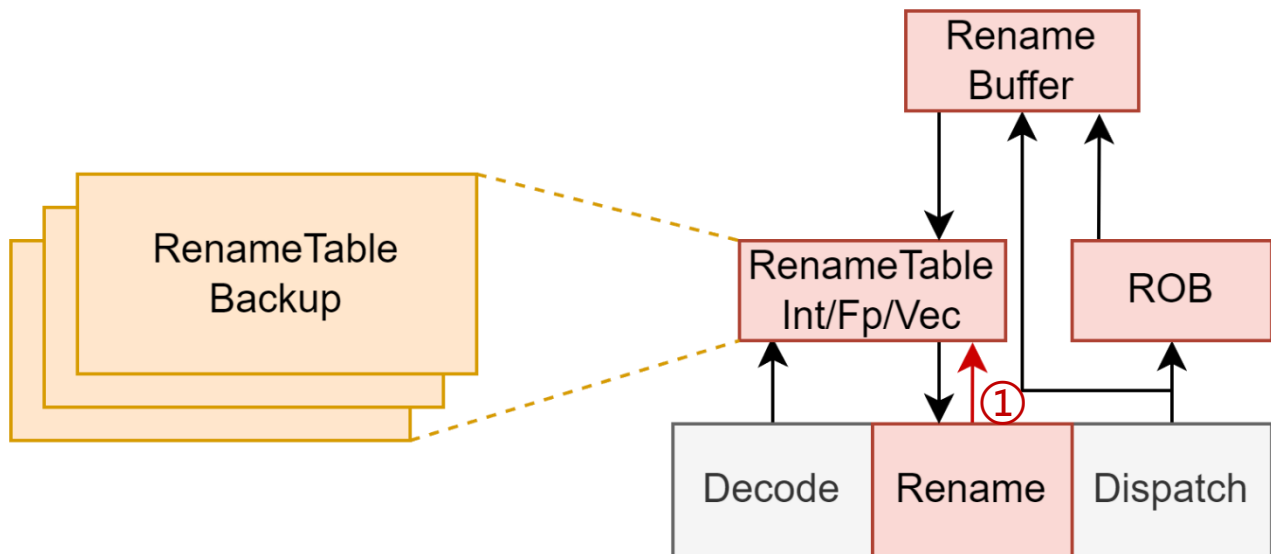
- 误预测和中断异常将导致流水线冲刷
- 恢复重命名表速度受 commit width 限制
- 重定向冲刷指令较多时，回滚周期数太多
- 假如重命名表可以定期存档？
- **从存档恢复就很快！**



ROB 重定向的 walk 过程

基于检查点的 RAT 恢复机制——创建

① Rename 阶段基于一定的规则决定是否创建检查点

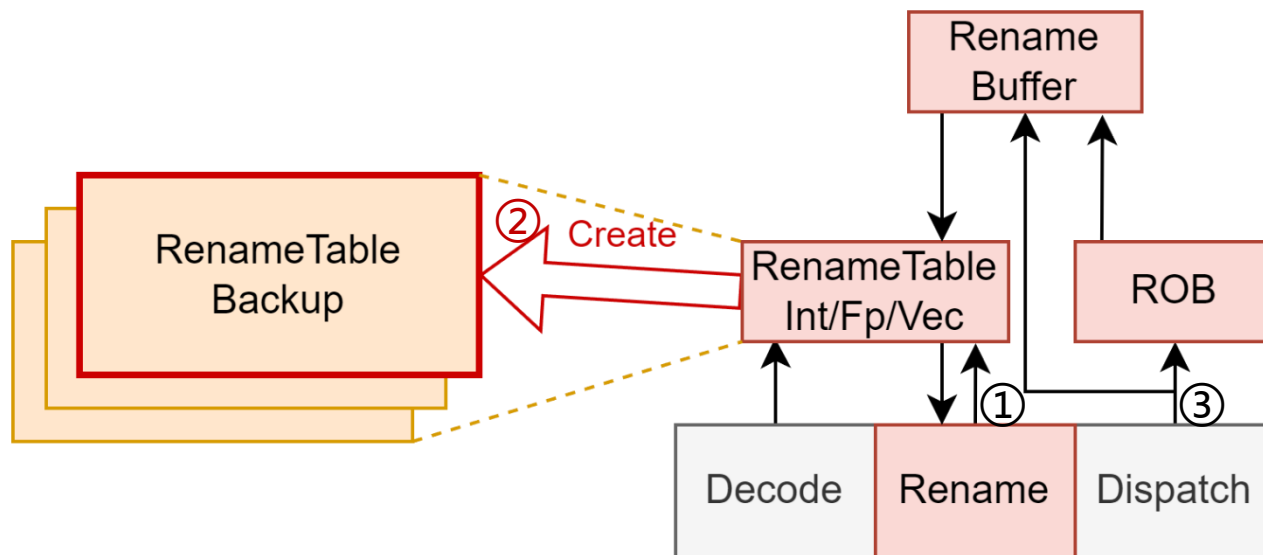


创建检查点示意图

基于检查点的 RAT 恢复机制——创建

① Rename 阶段基于一定的规则决定是否创建检查点

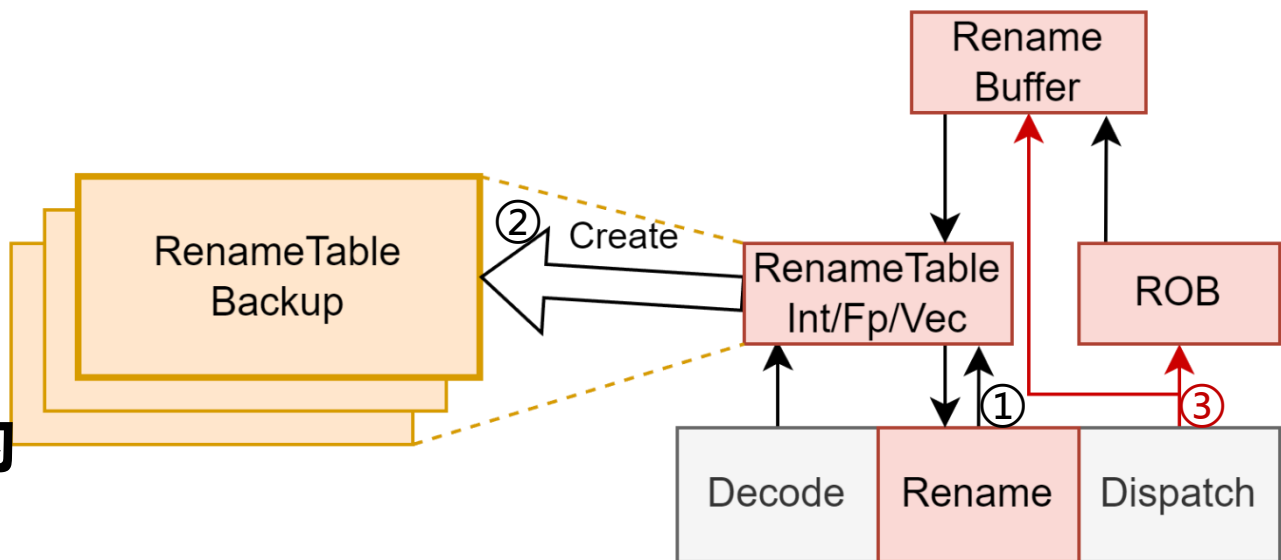
② RAT 创建当前的重命名状态备份



创建检查点示意图

基于检查点的 RAT 恢复机制——创建

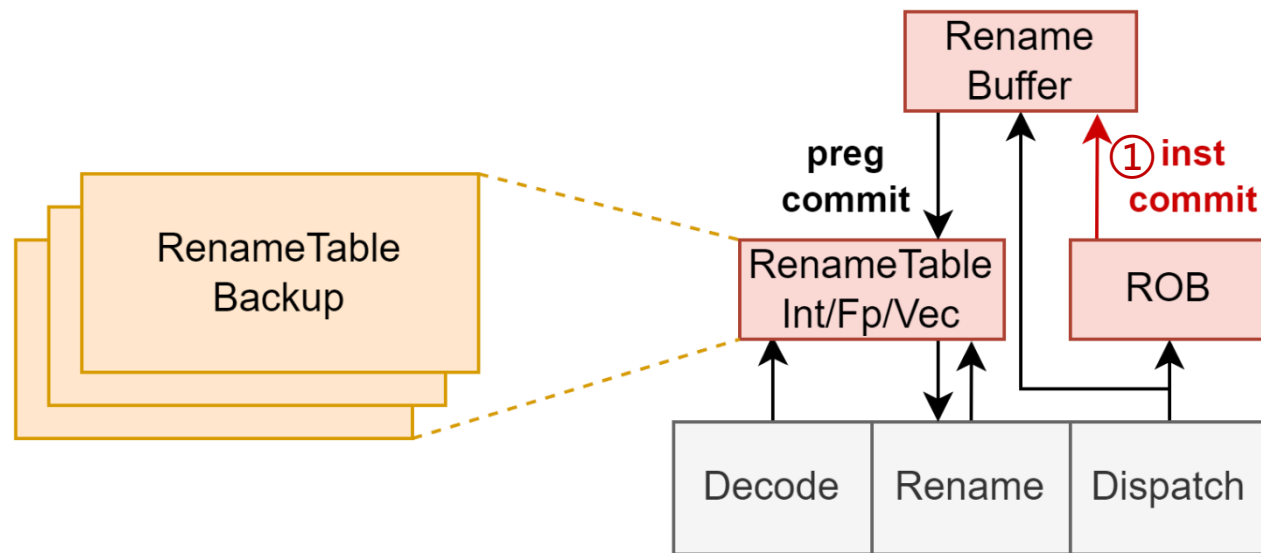
- ① Rename 阶段基于一定的规则决定是否创建检查点
- ② RAT 创建当前的重命名状态备份
- ③ Dispatch 阶段为 ROB 和 RAB 等的入队指针创建检查点



创建检查点示意图

基于检查点的 RAT 恢复机制——提交

① ROB 提交指令，将 entry 对应的寄存器映射数量提交给 RAB

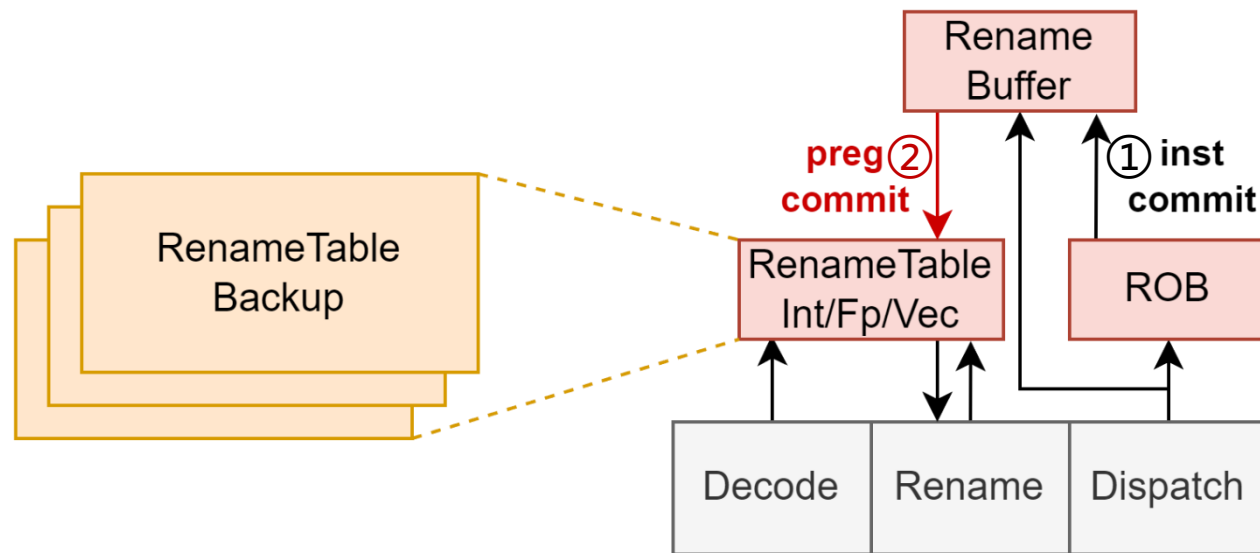


提交过程示意图

基于检查点的 RAT 恢复机制——提交

① ROB 提交指令，将 entry 对应的寄存器映射数量提交给 RAB

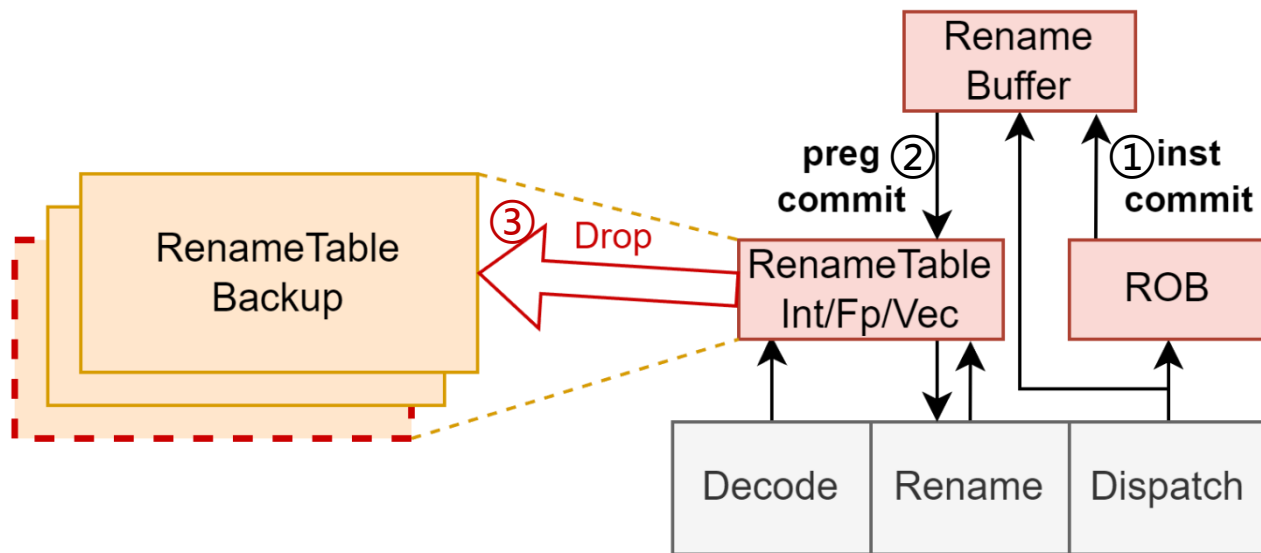
② RAB 提交寄存器映射给 RAT



提交过程示意图

基于检查点的 RAT 恢复机制——提交

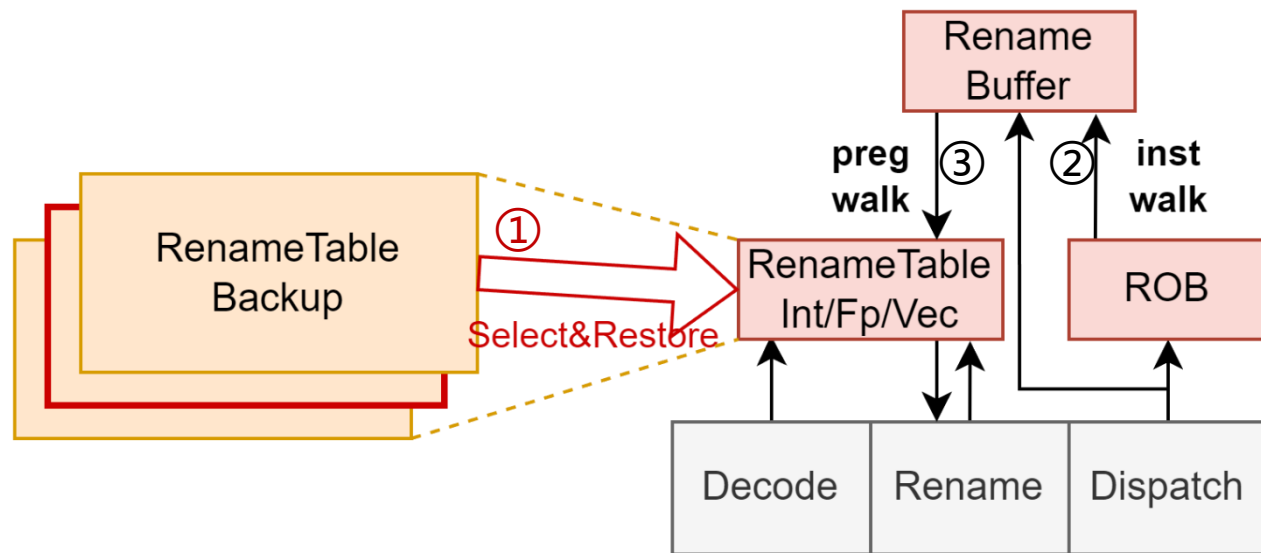
- ① ROB 提交指令，将 entry 对应的寄存器映射数量提交给 RAB
- ② RAB 提交寄存器映射给 RAT
- ③ RAT 丢弃过期的备份



提交过程示意图

基于检查点的 RAT 恢复机制——恢复

① RAT 寻找离回滚终点最近的备份并恢复

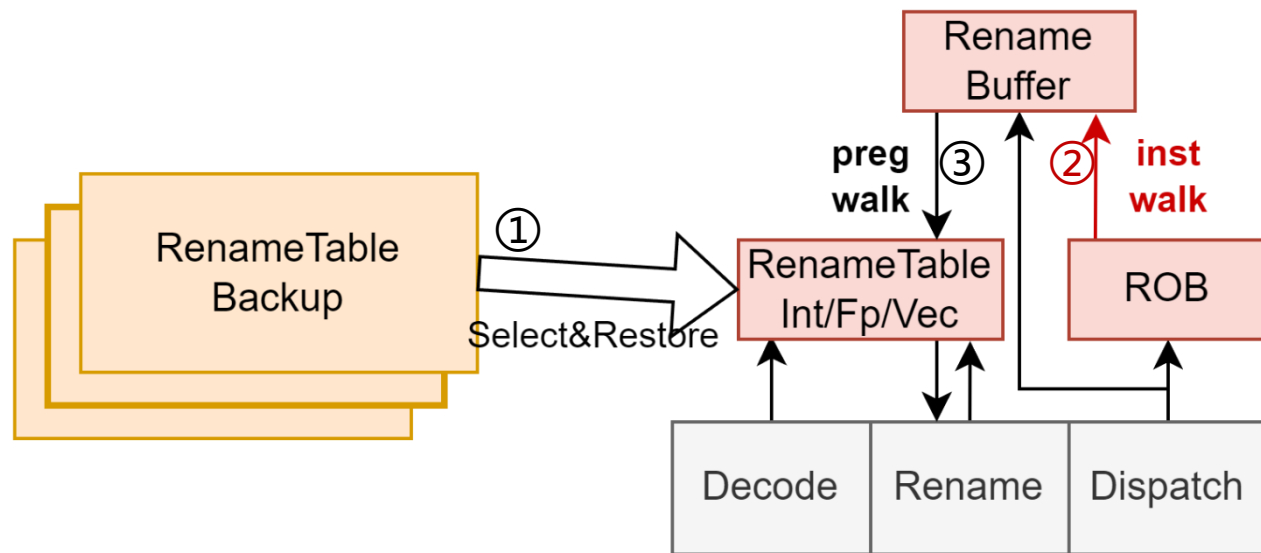


选择检查点、从检查点恢复的示意图

基于检查点的 RAT 恢复机制——恢复

① RAT 寻找离回滚终点最近的备份并恢复

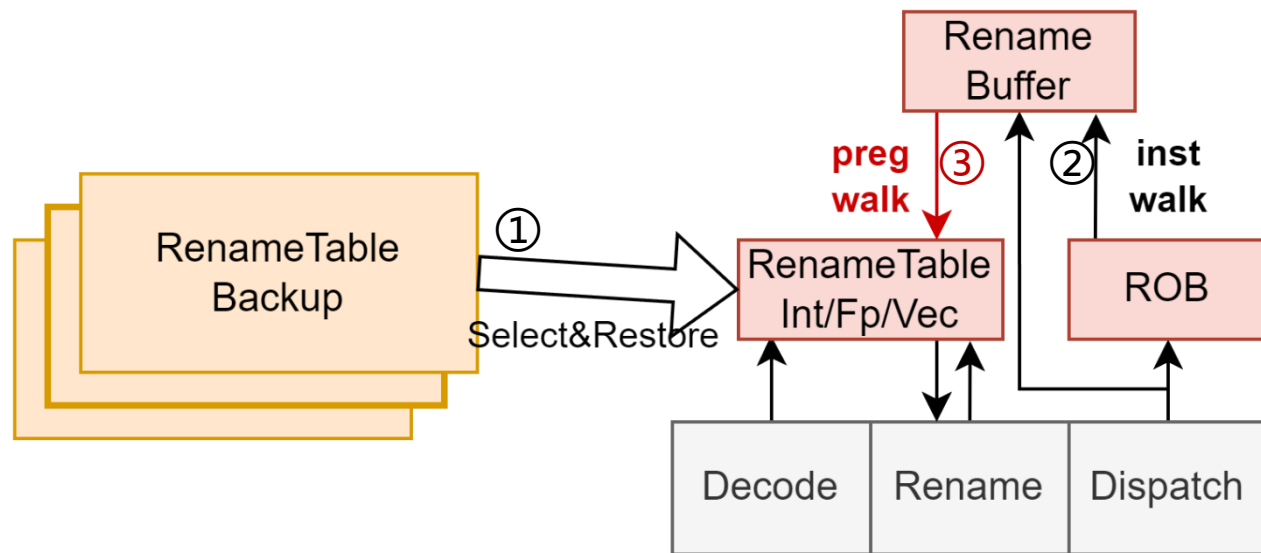
② ROB 从备份指针回滚指令，并向 RAB 传递对应的寄存器数量



选择检查点、从检查点恢复的示意图

基于检查点的 RAT 恢复机制——恢复

- ① RAT 寻找离回滚终点最近的备份并恢复
- ② ROB 从备份指针回滚指令，并向 RAB 传递对应的寄存器数量
- ③ RAB 从备份指针回滚寄存器映射给 RAT



选择检查点、从检查点恢复的示意图

后端流水线改进总结

- 发射后读寄存器堆
 - 重新组合功能单元
 - 推测唤醒和取消机制
- 指令和寄存器提交解耦
- 基于检查点的 RAT 恢复机制

昆明湖新需求-向量

- 昆明湖向量扩展规格
 - 兼容 RISC-V Vector 1.0 向量指令集扩展
 - VLEN: 128
 - 数据类型: INT8/INT16/INT32/INT64/FP16/FP32/FP64
 - 向量寄存器和浮点寄存器共用寄存器堆
 - 支持基于**指令拆分**和**重命名**的乱序调度
- 更多细节将在向量扩展报告中展示

昆明湖新需求-向量

- 昆明湖向量扩展规格
 - 兼容 RISC-V Vector 1.0 向量指令集扩展
 - VLEN: 128
 - 数据类型: INT8/INT16/INT32/INT64/FP16/FP32/FP64
 - 向量寄存器和浮点寄存器共用寄存器堆
 - 支持基于**指令拆分**和**重命名**的乱序调度
- 更多细节将在**向量扩展**报告中展示

8月24日 15:00
欢迎来三楼珍珠厅参加
《向量扩展设计和实现》报告

敬请批评指正!