

香山处理器的访存流水实现

王华强

中科院计算所

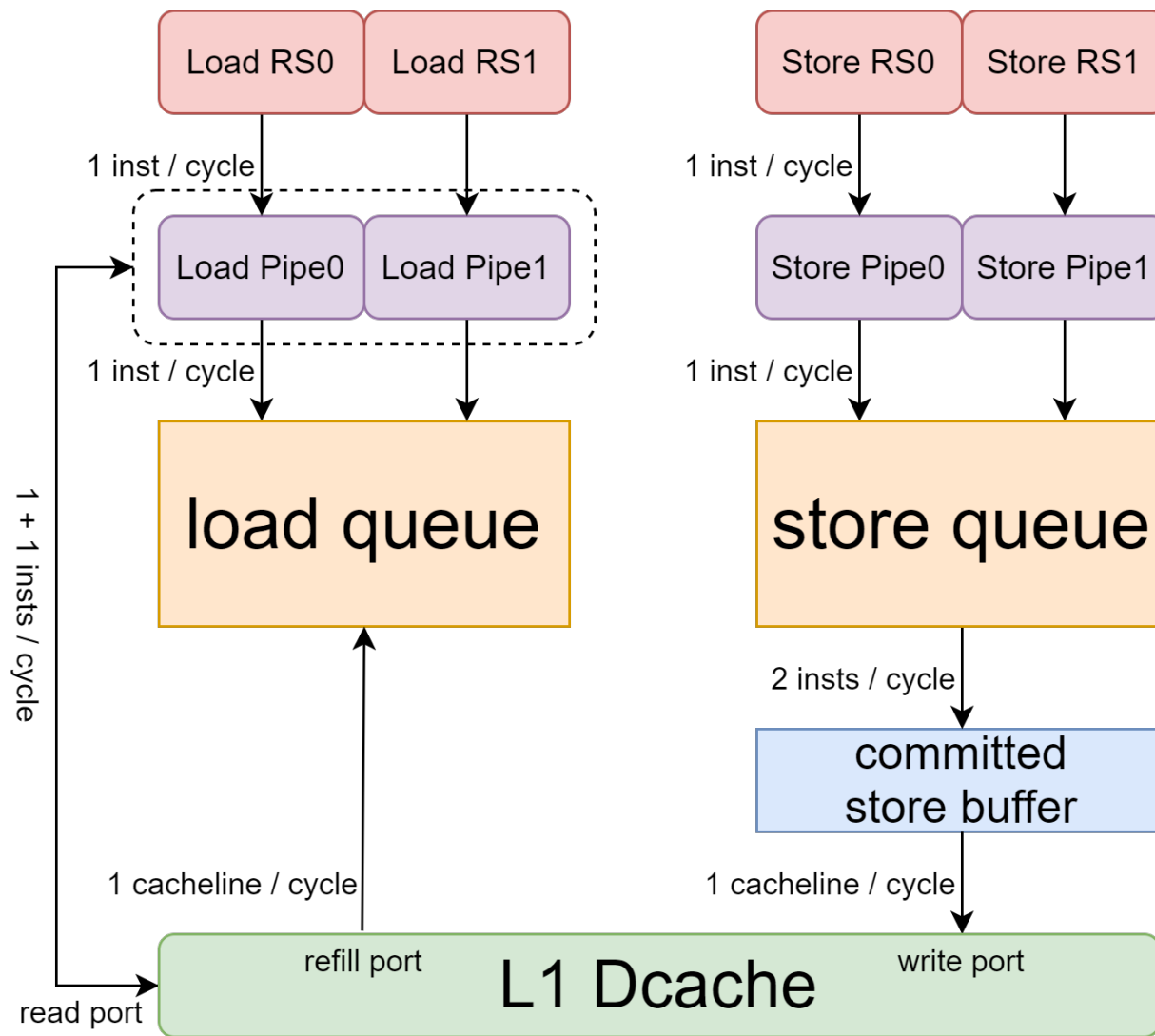
2021年6月25日

核内访存

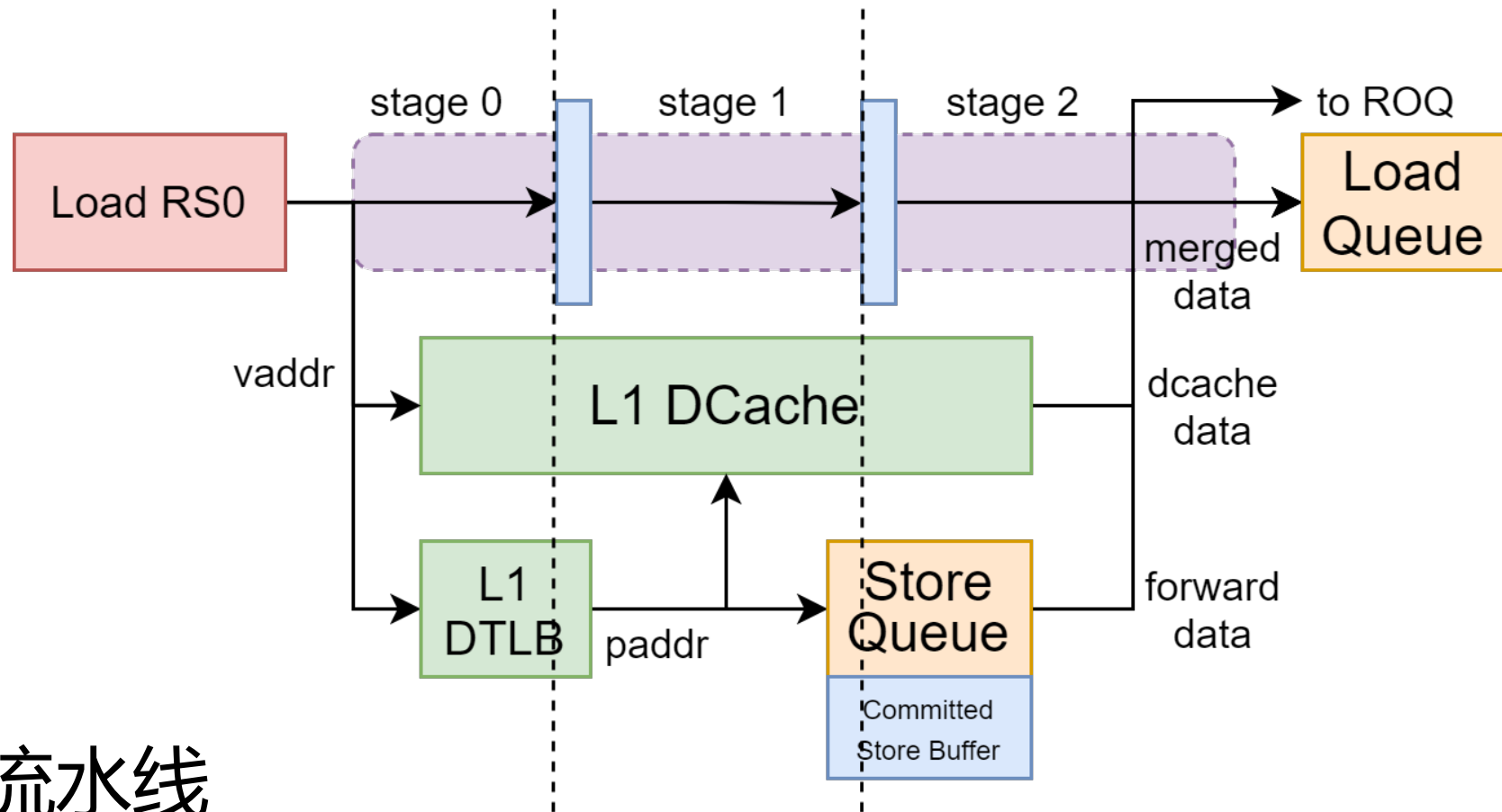
- 访存单元整体架构
- load 流水线划分
- store 流水线划分
- 核内访存队列与缓冲
- 关键访存机制的实现

核内整体访存架构

- **2**条 load 流水线，**3**级流水
- **2**条 store 流水线，**2+2**级流水
- **64**项 load queue
- **48**项 store queue
- **16**项 committed store buffer
- 配合香山 VIPT L1 cache

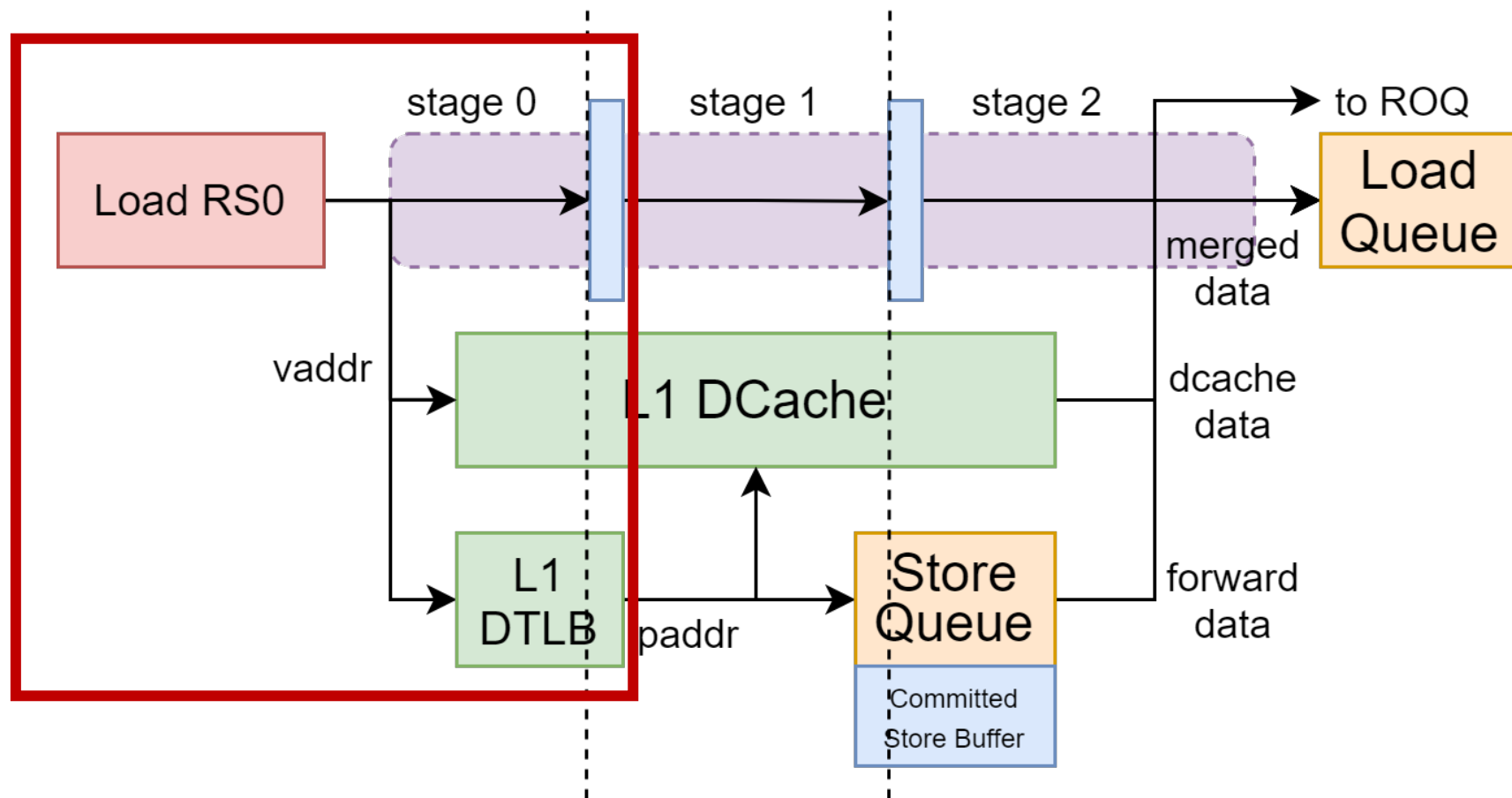


load 流水线划分



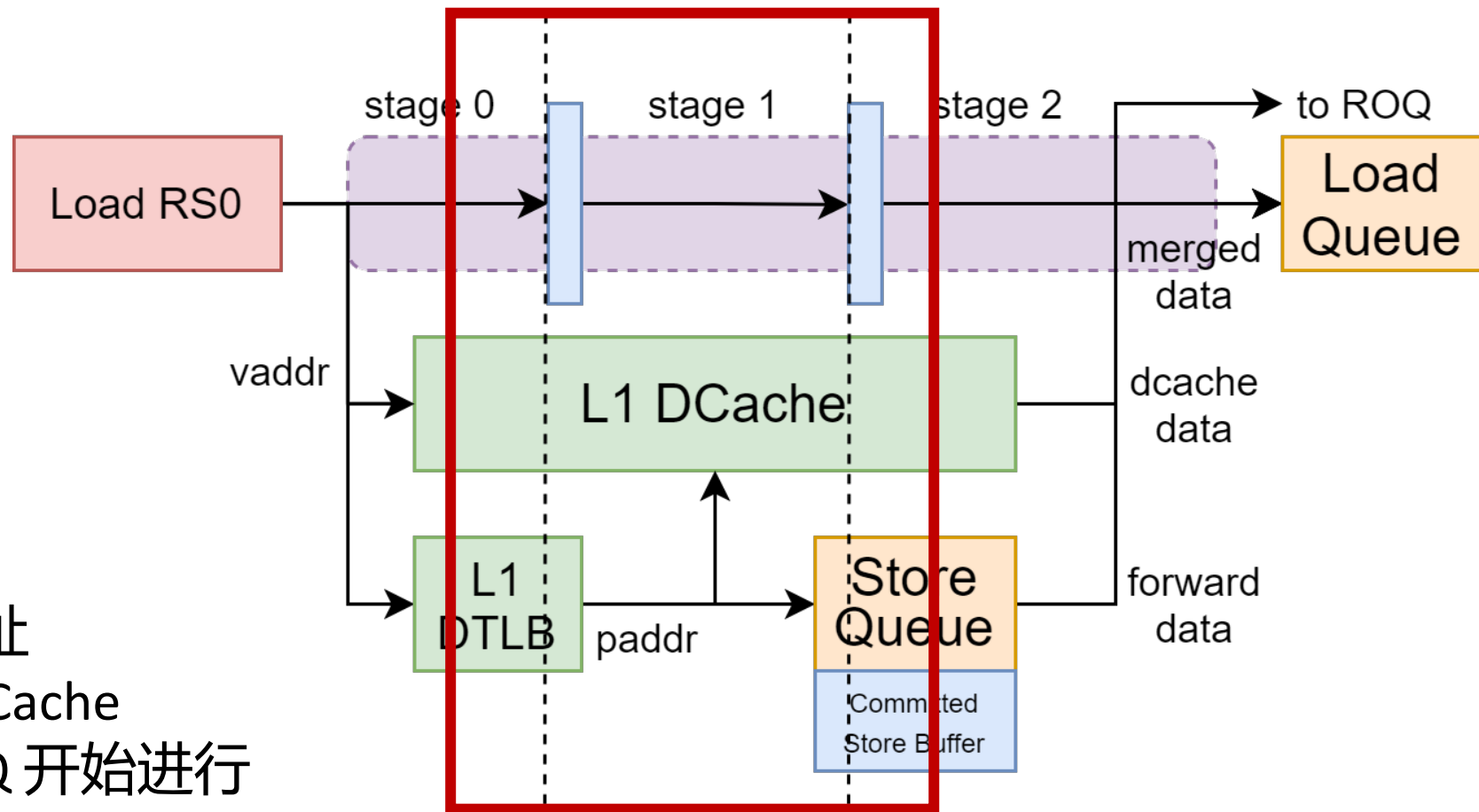
- **2条3级** load 流水线
- 配合 VIPT L1 DCache

load 流水线划分



- Stage 0
 - 计算虚拟地址送进 TLB , DCache

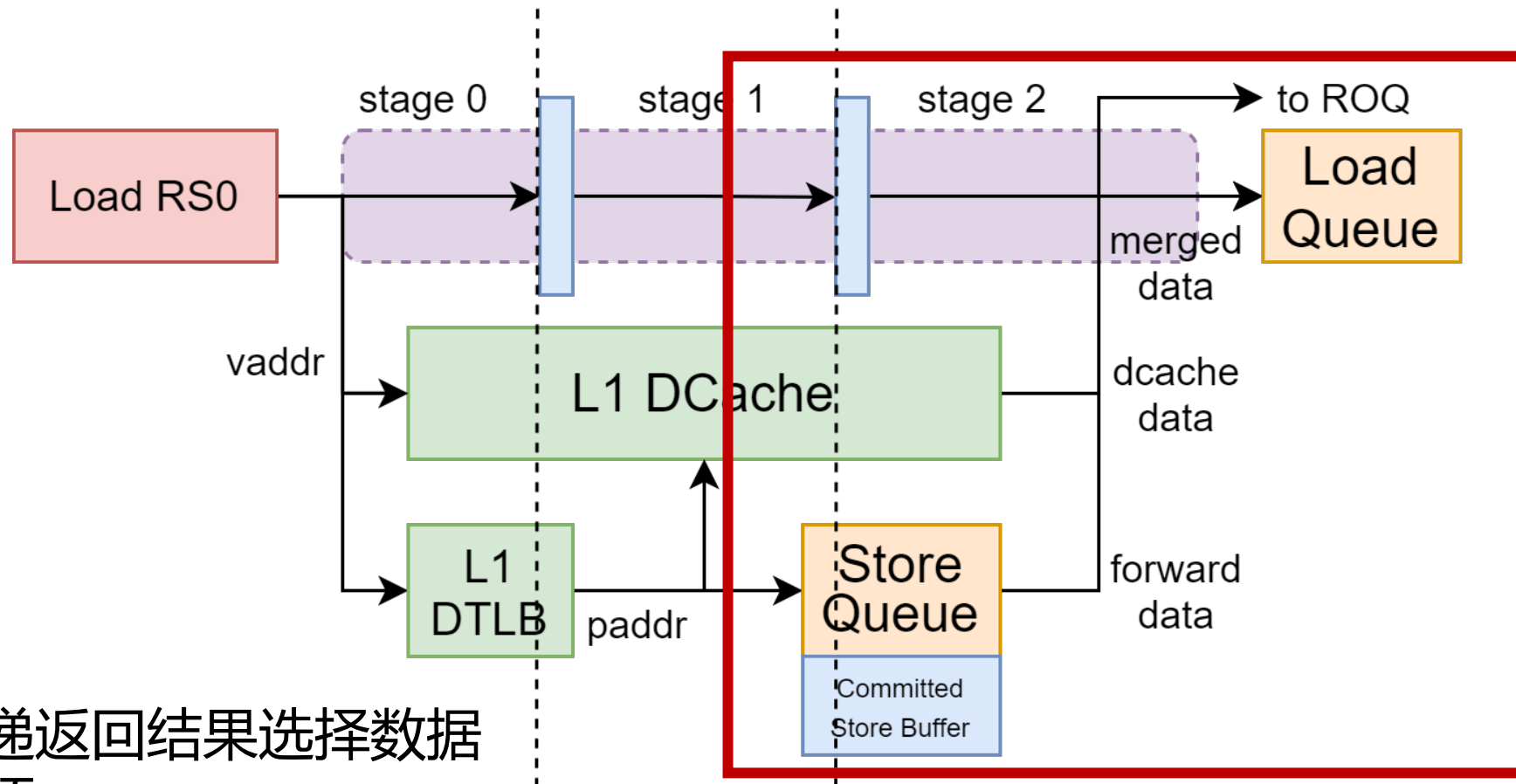
load 流水线划分



- Stage 1

- TLB 产生物理地址
- 物理地址送进 DCache
- 物理地址送进 SQ 开始进行 Store to Load Forward
- 根据 DCache 返回的命中向量准备唤醒后续指令

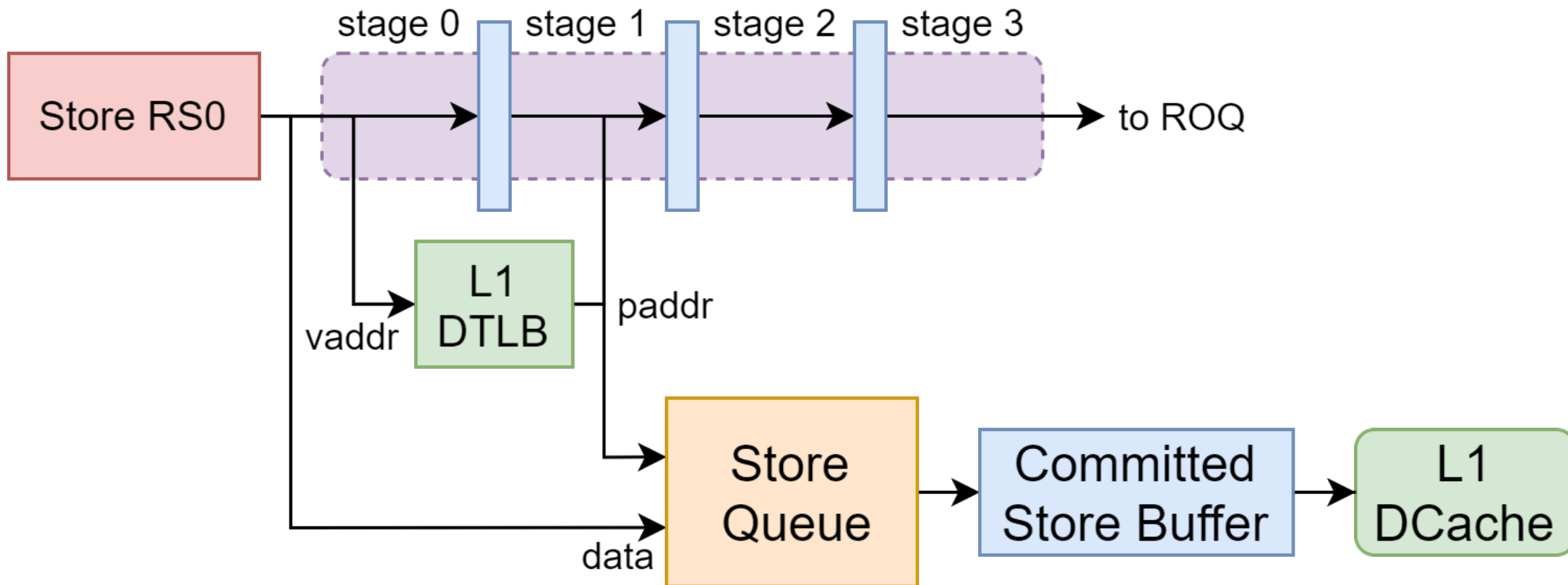
load 流水线划分



• Stage 2

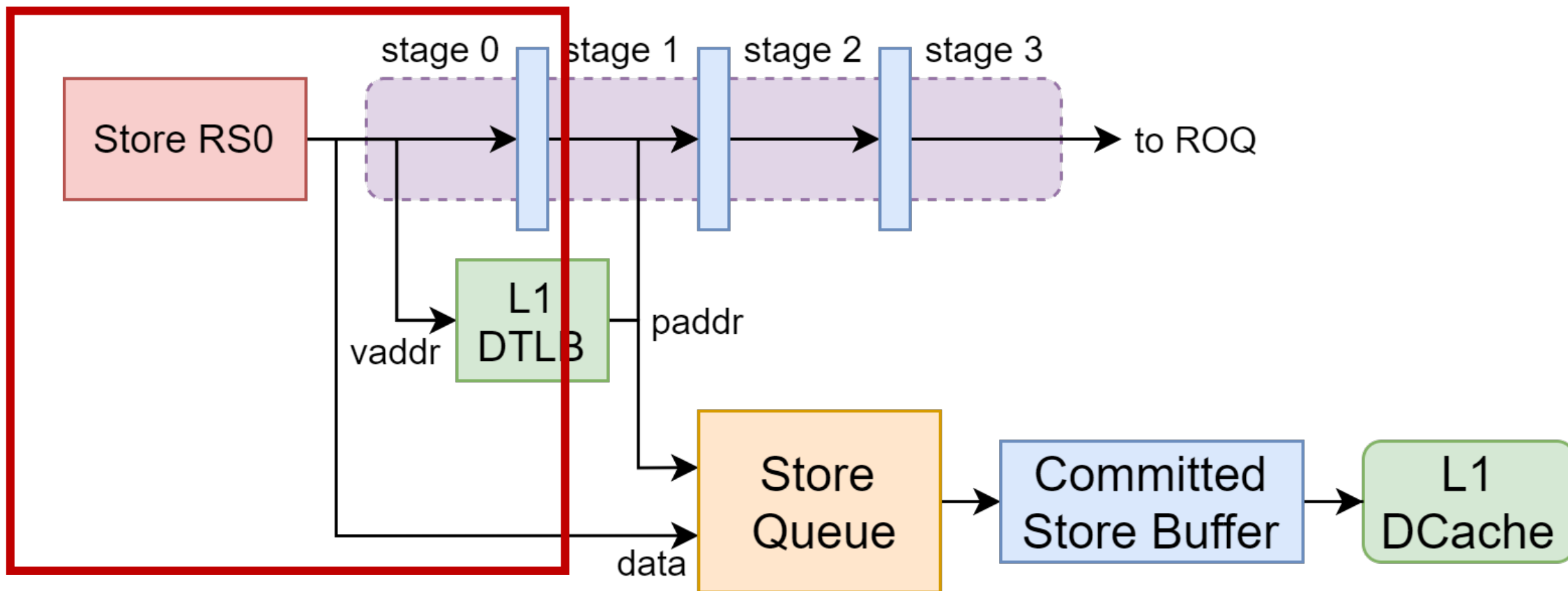
- 根据 DCache 及前递返回结果选择数据
- 更新 LQ 中的对应项
- 结果（整数）写回到公共数据总线
- 结果（浮点）送到浮点模块

store 流水线划分



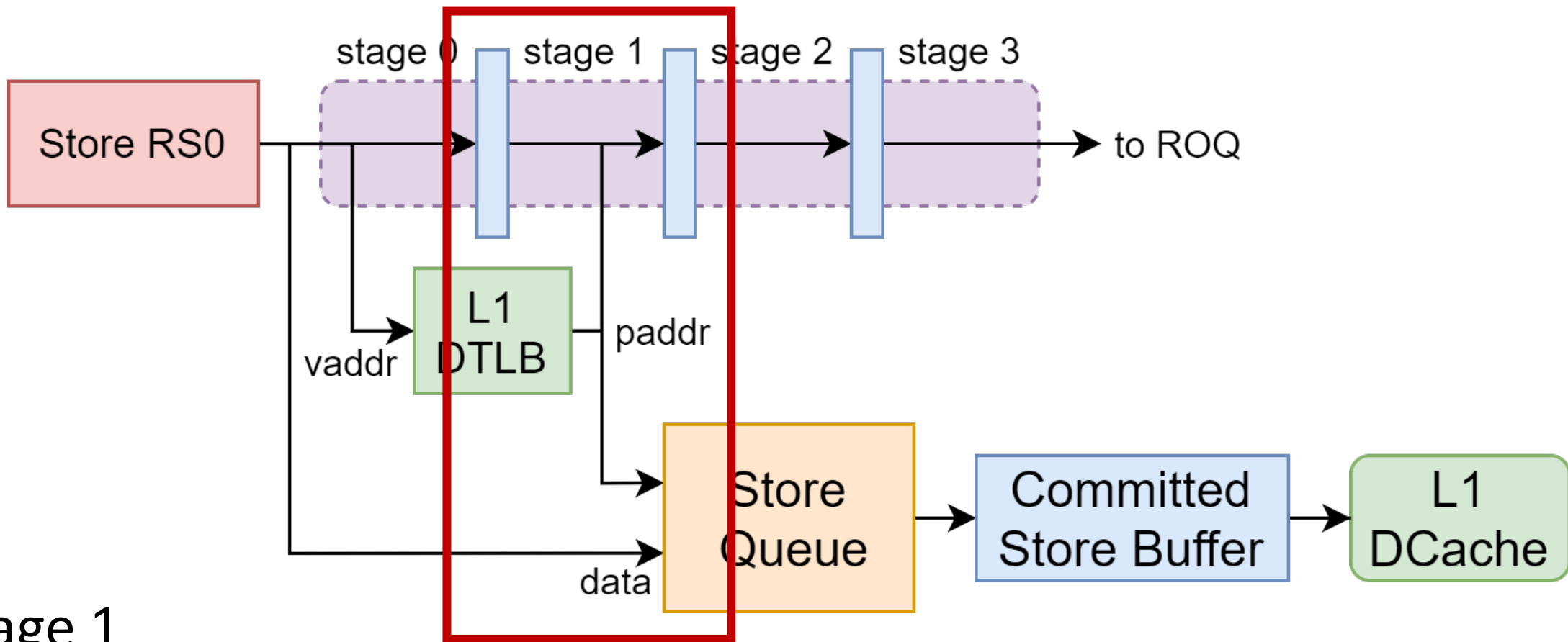
- 2 条 2+2 级 store 流水

store 流水线划分



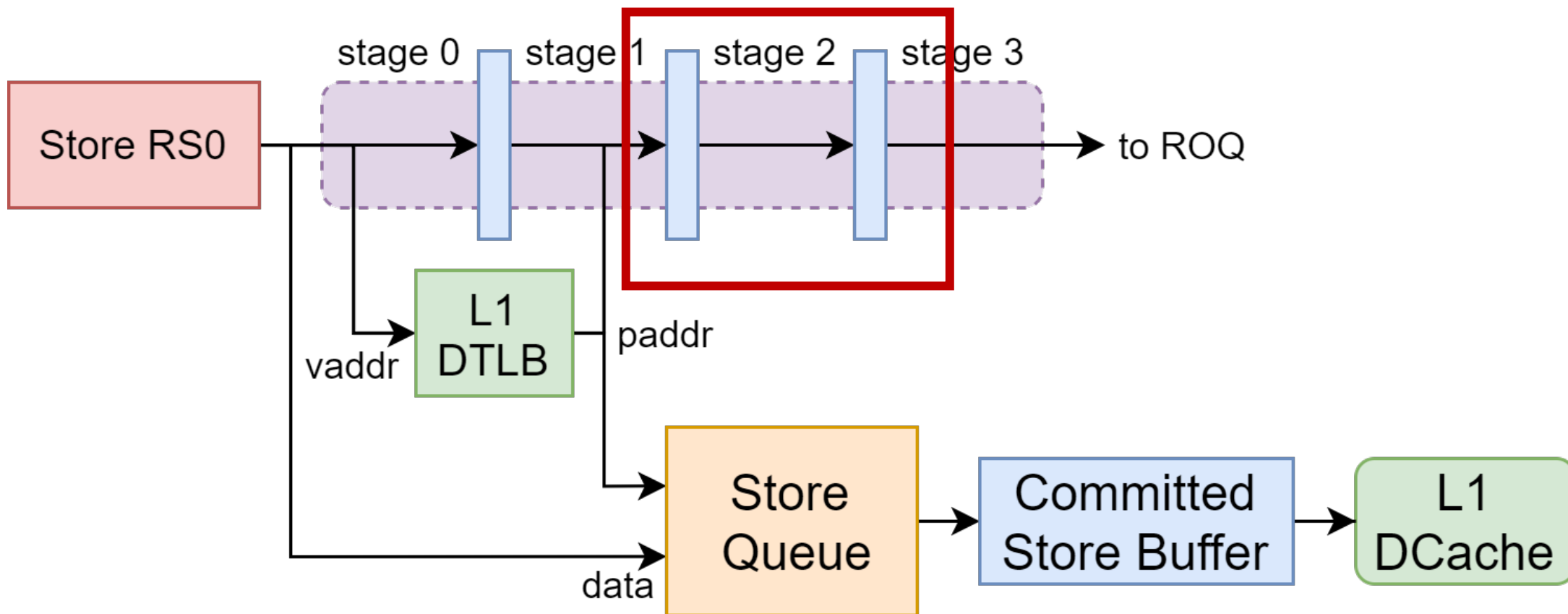
- Stage 0
 - 计算虚拟地址，送入 TLB

store 流水线划分



- Stage 1
 - TLB 生成物理地址，送入SQ
 - 开始进行访存依赖检查

store 流水线划分



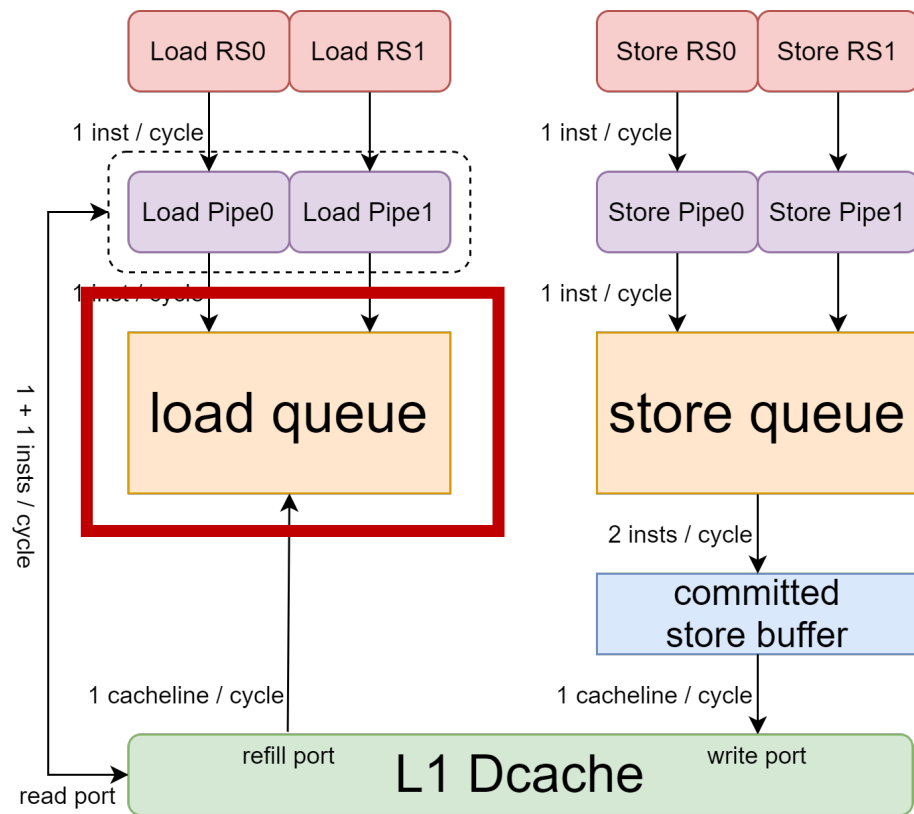
- Stage 2
 - 访存依赖检查



- 完成访存依赖检查

Load Queue

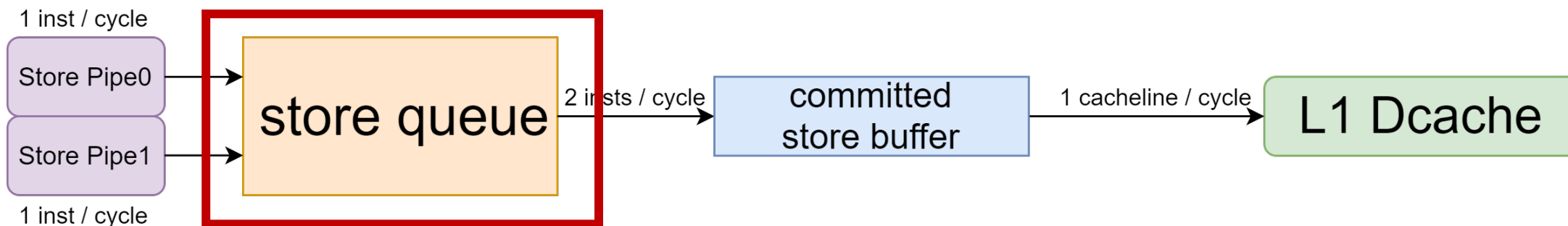
- **64**项循环队列
- 每周期至多：
 - 从 dispatch 处接收**6**条指令
 - 从 load 流水线接收**2**条指令的结果
 - 写回**2**条 miss 的 load 指令
 - 这些指令已经取得了 refill 的数据
 - 会与正常的访存流水线**争用**两个写回端口



- cache refill 时，会将这一 cacheline 的**所有数据**写到 LQ 中
 - 所有在LQ中等待这一 cacheline 数据的指令都会得到数据

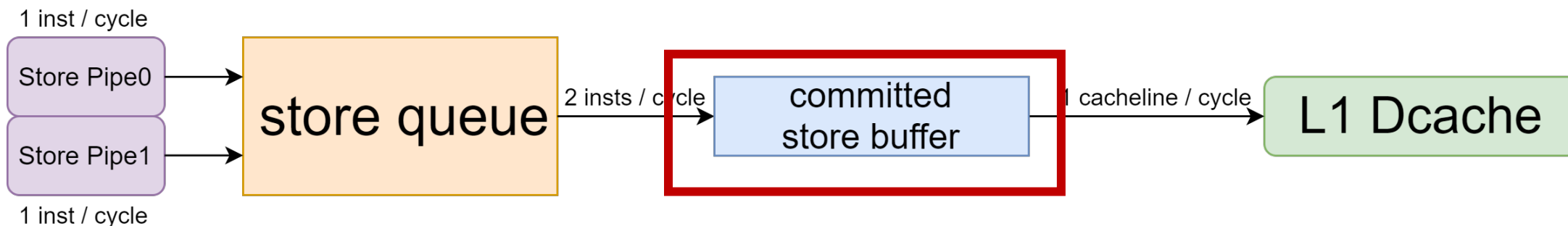
Store Queue

- **48**项循环队列
- 每周期至多：
 - 从 dispatch 处接收**6**条指令
 - 从 store 流水线接收**2**条指令的结果
 - 将**2**条指令的数据写入 committed store buffer
- 为**2**条load流水线提供 Store to Load Forward 结果



Committed Store Buffer

- 16项
- 每周期至多：
 - 接收2条 SQ 写入的 store 指令
 - 向 DCache 写一个 cacheline
- 以 cacheline 为粒度合并 store
- 当容量超过阈值时，执行换出操作
 - 使用 PLRU 选出要写入 cache 的 cacheline





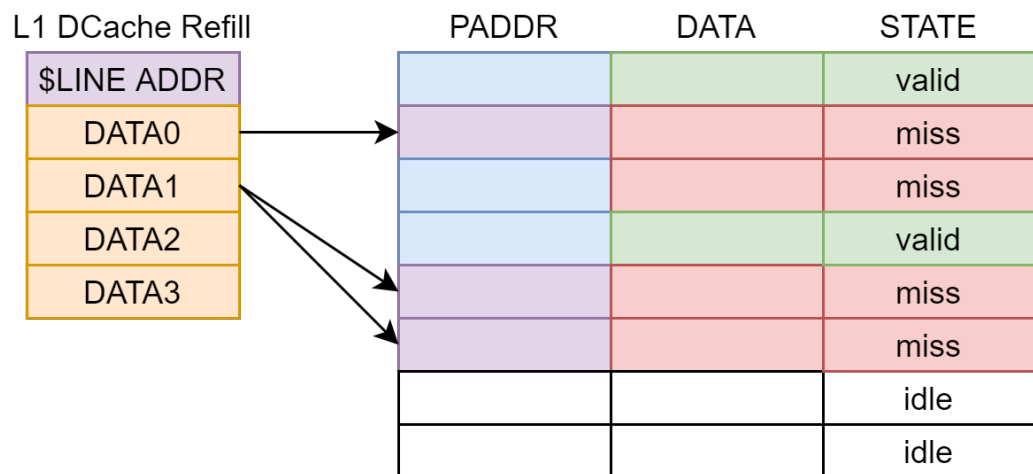
关键访存机制的实现

- DCache miss 处理
- TLB miss 处理
- Store to Load Forwarding
- Memory Dependency 的预测、检测及违例恢复

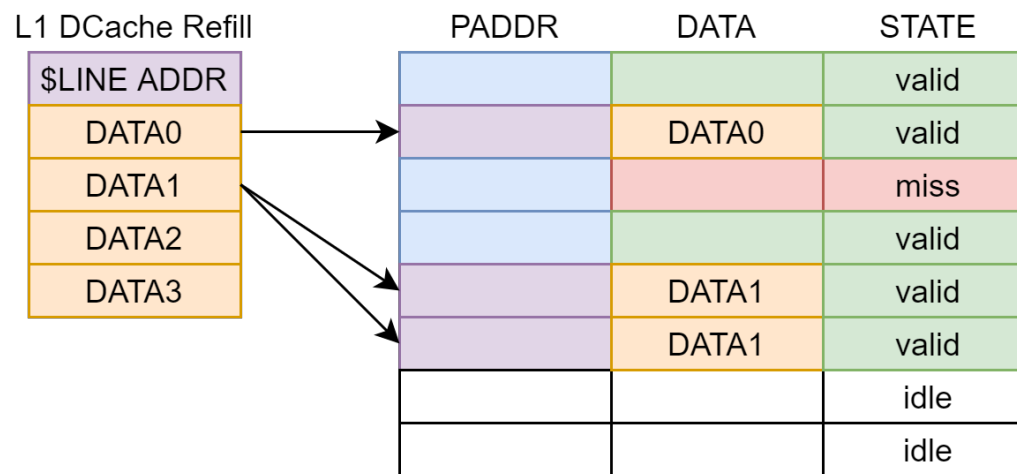


DCache Miss

- 如果 load miss, 在 load queue 中侦听 L1 DCache refill 结果



refill 前的 load queue



refill 后的 load queue

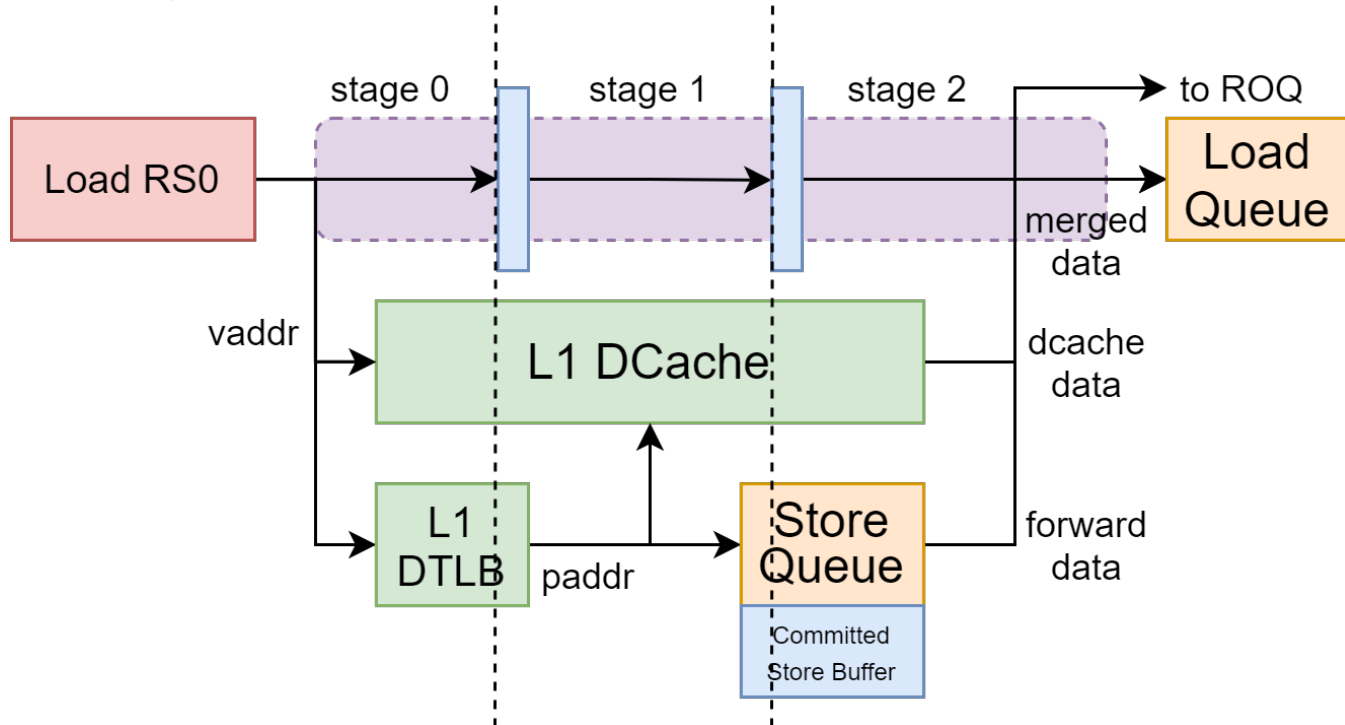
* 紫色方框代表位于相同的 cacheline

TLB miss 的处理：标记 replay

- 指令在以下条件下会被标记为需要 replay
 - TLB miss
 - L1 DCache MSHR full
 - 前递时发现地址匹配但数据未就绪 (Data invalid)
- 一条指令从访存 RS 中发射之后仍然**需要保留在 RS 中**
- 访存指令在离开流水线时向 RS **反馈**是否被标记为 replay
- 需要 replay 的指令会在 RS 中继续等待
 - 在一定时间间隔之后重新发射

Store To Load Forwarding

- 分配到三级流水
- 并行检查 committed store buffer 和 store queue
- 如果 DCache miss, 保留 forward 结果

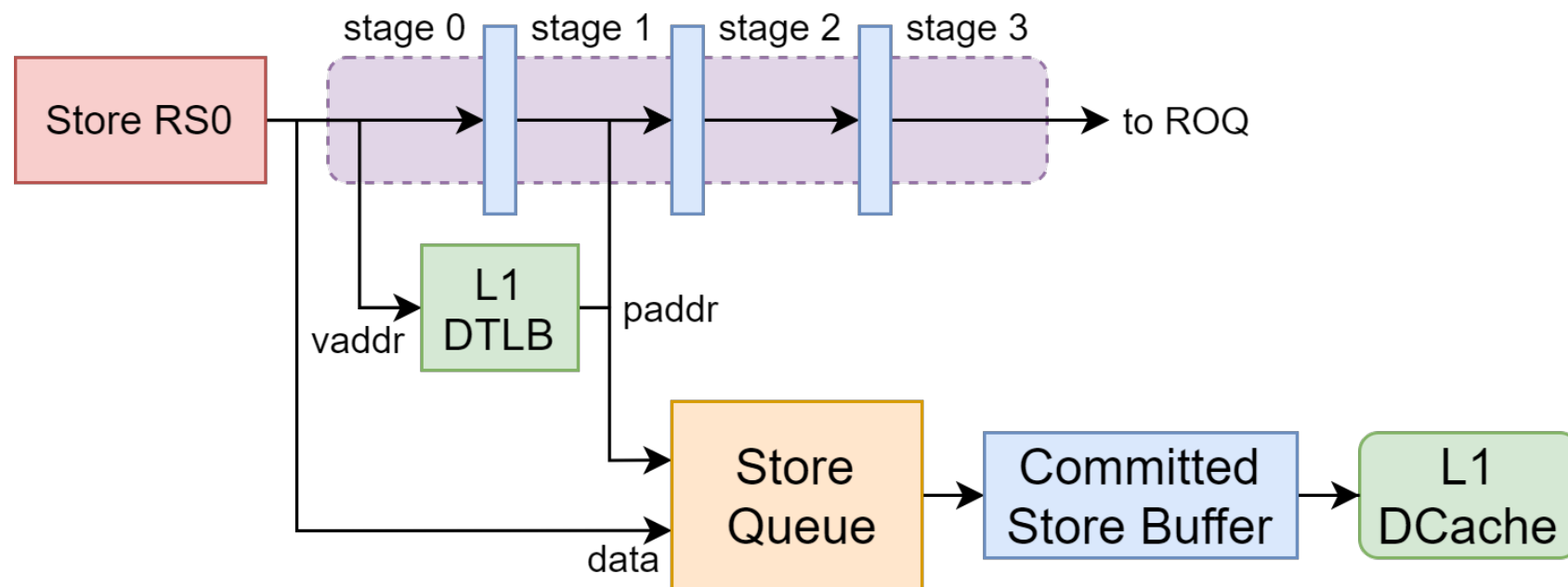




Load Violation

- 检查

- 在 store 指令到达 stage 1 时开始检查



- 恢复

- 检查出后**立即**触发回滚
- 无需等待指令提交



Memory Dependence Prediction

- 在 decode 附近根据 PC 预测 Memory Dependence
- 预测方式
 - Load Wait Table[1]
 - Store Sets[2]
- 如果预测到一条 **load** 指令可能发生 violation
 - load 指令在RS中等待到**之前的 store 都发射**之后才能发射

[1] Kessler R E . The Alpha 21264 Microprocessor[J]. IEEE Micro, 1999, 19(2):24-36.

[2] Chrysos G Z , Emer J S . Memory Dependence Prediction using Store Sets[J]. ACM SIGARCH Computer Architecture News, 2002, 26(3):142-153.

感谢 

北京微核芯科技有限公司
BEIJING VCORE TECHNOLOGY CO., LTD.

提供产业经验、联合完成结构设计及物理设计

招募香山处理器二期联合开发合作伙伴



北京微核芯科技有限公司
BEIJING VCORE TECHNOLOGY CO., LTD.



ESWIN

优矽科技

欢迎更多伙伴加入！

联系人：李迪 13811881360

**谢谢！
请批评指正**

以下内容备用



核内访存: 机制列表

- Load Queue / Store Queue / committed Store Buffer
- Store to Load Forwarding (STLF)
- Load violation check
- Load violation predict / Delay
- Uncached memory access
- Dcache Miss / TLB Miss
- Sleep
- Commit / Exception
- Committed Store buffer

Load Violation Check

- 在store指令到达S1时开始检查
- 三个来源
 - 两级Load流水线中无法被forward的指令
 - L1, L2
 - L0的指令可以被forward, 无需报violation
 - 已经完成的load: 检查LQ
- store流水线宽度为2: 一共有 $2 \times 3 = 6$ 个来源需要根据sqldx判断他们的先后

核内访存: Load violation check

- 需要找到最早违例的load
- 三级流水
 - S1: 在store写入到SQ的同时开始load violation check
 - Paddr CAM
 - 状态检查
 - S2
 - 针对LQ中的匹配结果做优先级选择
 - 合并L1/L2的检查结果
 - S3
 - 合并所有剩下的检查结果
 - 发信号给redirectGen模块

```
/**
 * Memory violation detection
 *
 * When store writes back, it searches LoadQueue for younger load instructions
 * with the same load physical address. They loaded wrong data and need re-execution.
 *
 * Cycle 0: Store Writeback
 *   Generate match vector for store address with rangeMask(stPtr, enqPtr).
 *   Besides, Load instructions in LoadUnit_S1 and S2 are also checked.
 * Cycle 1: Redirect Generation
 *   There're three possible types of violations, up to 6 possible redirect requests.
 *   Choose the oldest load (part 1). (4 + 2) -> (1 + 2)
 * Cycle 2: Redirect Fire
 *   Choose the oldest load (part 2). (3 -> 1)
 *   Prepare redirect request according to the detected violation.
 *   Fire redirect request (if valid)
 */

// stage 0:      lq l1 wb      l1 wb lq
//               | | |      | | | (paddr match)
// stage 1:      lq l1 wb      l1 wb lq
//               | | |      | | |
//               | |-----| |
//               |           |
// stage 2:      lq      l1wb      lq
//               |       |
//               -----
//               |
//               rollback req
```



访存违例预测: Load Wait Table (LWT)

- 21264[1]的设计
 - 1024x1 wait table
 - 在decode时根据PC查waittable, 获取loadWaitBit
 - 如果预测会发生访存违例, 则
 - 这些load需要在RS中等待到之前的所有store都从RS发射之后才能发射
 - 这些load在RS中, 但被置为不可选择的状态
 - 在SQ中维护已从store RS发射的sqldx指针
 - RS通过把这个sqldx与RS内部每条指令的sqldx比对, 将等待中的load指令转为可被选择的状态
 - 在访存违例发生时, 更新wait table的对应位
 - 每隔固定的周期重置waittable
 - 扩展: 使用2bit饱和计数器

访存违例预测: Store Set[2]

- 根据产生访存违例的store-load指令对 PC进行预测
- 在可能产生访存违例的store取指成功之后, 在表中进行标记. 如果后续有对应的load指令取指, 则预测这个load会访存违例
- 在store地址被计算出来之后, 取消标记, 随后的load不会被预测为访存违例

对应的load
不会被预测为违例

对应的load
会被预测为违例

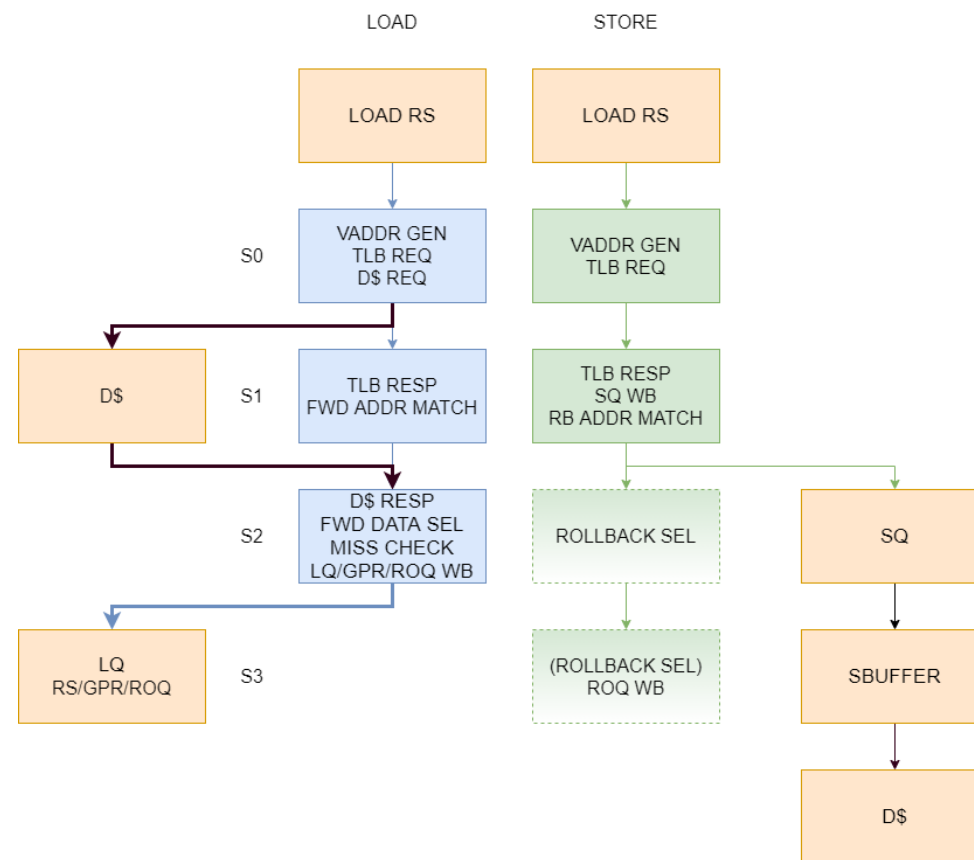
对应的load
会被预测为违例

store取指
经过访存违例预测器

store地址就绪
从RS中发射

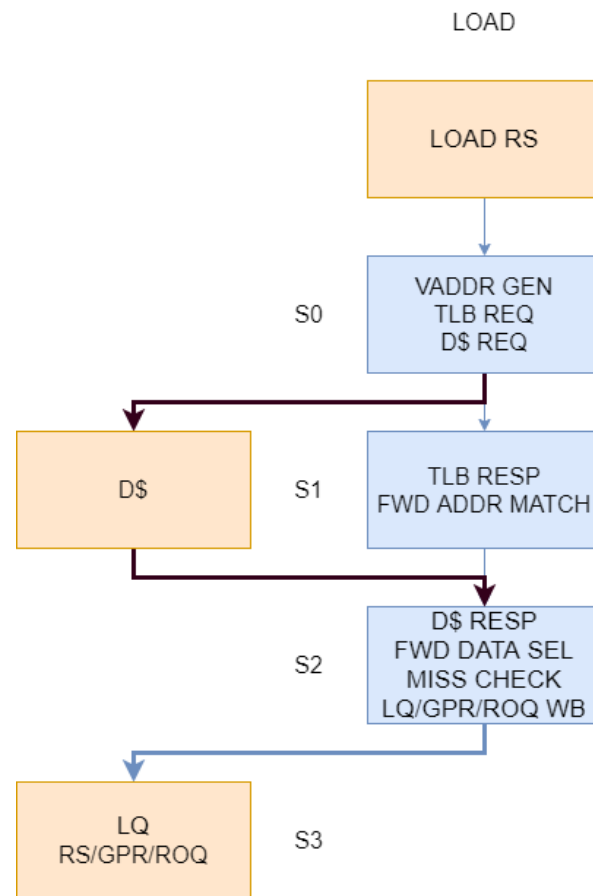
- 需要判断那些load-store是相互对应的

- # 核内整体访存架构
- **2**条 load 流水线，**3**级流水
 - **2**条 store 流水线，**2+2**级流水
 - 2级地址计算及转换
 - 2级异步的 load violation 检查
 - **64**项 load queue
 - **48**项 store queue
 - **16**项 committed store buffer
 - 合并已经提交的写请求
 - 以 cacheline 为单位将数据写入cache
 - 配合香山 non-blocking VIPT L1 cache



load流水线划分

- Load0:
 - 计算 vaddr 送进 TLB , Dcache
- Load1:
 - TLB 产生 paddr
 - paddr 送进 SQ 开始进行 store to load forward
 - paddr 送进 Dcache
 - 根据 Dcache 返回的hit向量准备唤醒后续指令
- Load2:
 - 根据 Dcache 及 forward 返回结果选择数据
 - 更新LQ中的对应项
 - 结果写回到公共数据总线/送到浮点模块执行后续操作



store流水线划分

- Store0:
 - 计算 vaddr , 送入TLB
- Store1:
 - TLB 生成 paddr , 送入SQ
 - 开始进行 load violation检查 (1/3)
- Store2:
 - load violation 检查 (2/3)
- Store3:
 - 完成 load violation 检查 (3/3)
 - 更新 SQ 中对应指令状态
 - 通知Roq store 指令可以合法提交

