

SiFive InclusiveCache 在香山处理器中的应用与调整

王海喆¹, 刘志刚¹, 谭佳展²

¹中科院计算所

²北京大学

2021年6月25日

■ 大纲

- SiFive InclusiveCache 在香山的应用
- SiFive InclusiveCache 功能调整
- SiFive InclusiveCache 时序调整

■ SiFive InclusiveCache 基本介绍

- SiFive 开源的 TileLink 协议 Cache 实现
 - <https://github.com/sifive/block-inclusivecache-sifive>
- 目标应用场景: **System Cache / LLC**
- 支持上游一致性协议
 - 可向上游发 probe
- 不支持下游一致性协议
 - 不处理下游的 probe 请求
 - 假设自身为末级 cache

■ SiFive InclusiveCache 在香山的应用

- 为什么使用 SiFive InclusiveCache
 - 第一版香山（雁栖湖）专注 core 设计
 - 考虑多核一致性支持
 - 后续将重新编写 L2/L3 cache
- 规划应用
 - 私有 L2 (512KB)
 - 共享 L3 (4MB)
 - 目标频率：28nm 1.5 GHz (core 同频)
 - 仅针对 cache 模块优化内部寄存器间路径
- 实际应用
 - 单核私有 L2 (1MB)
 - 28nm 1.3 GHz (core 同频)
 - 流片成本和工期考量

■ 大纲

- SiFive InclusiveCache 在香山的应用
- SiFive InclusiveCache 功能调整
- SiFive InclusiveCache 时序调整

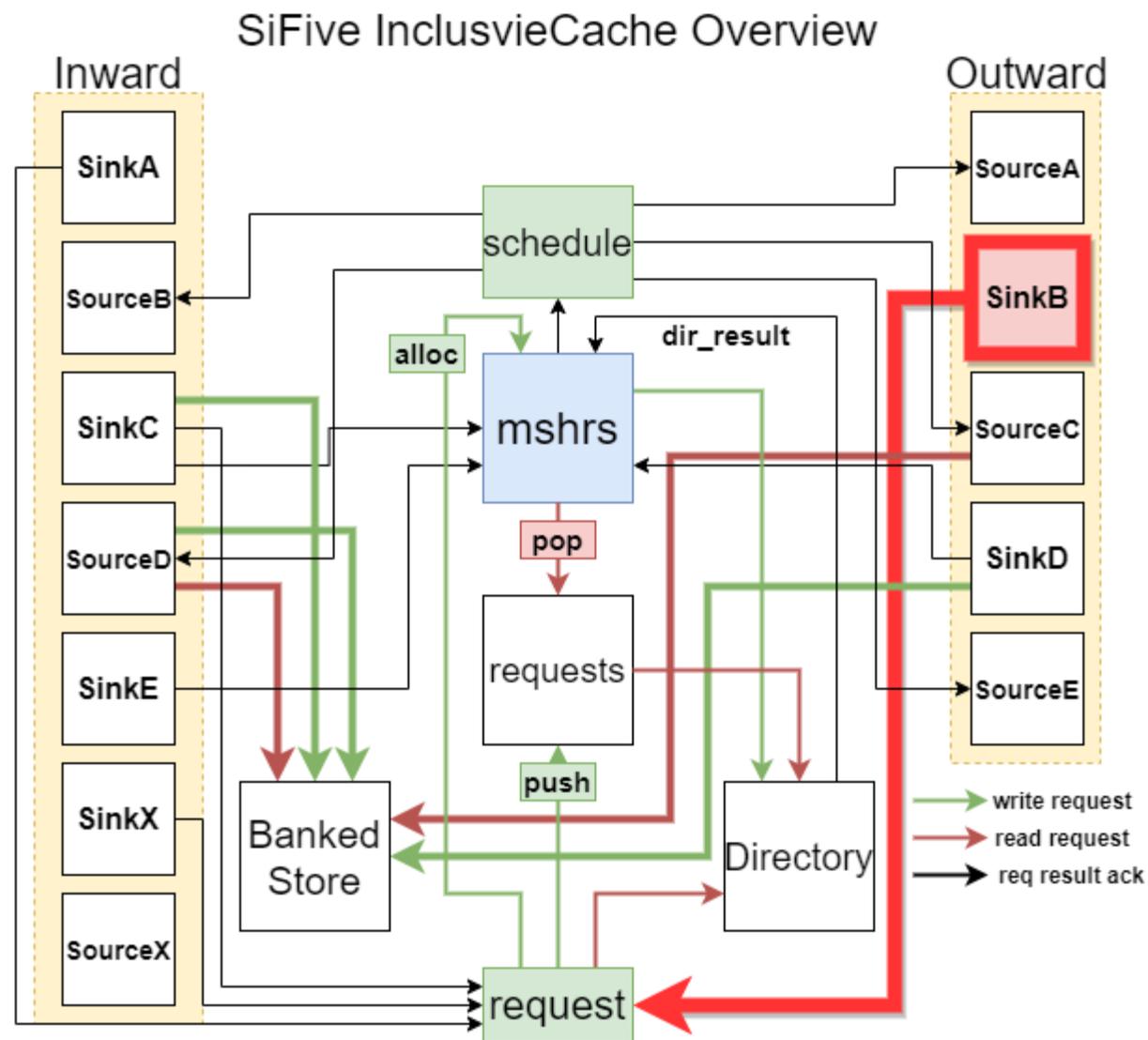
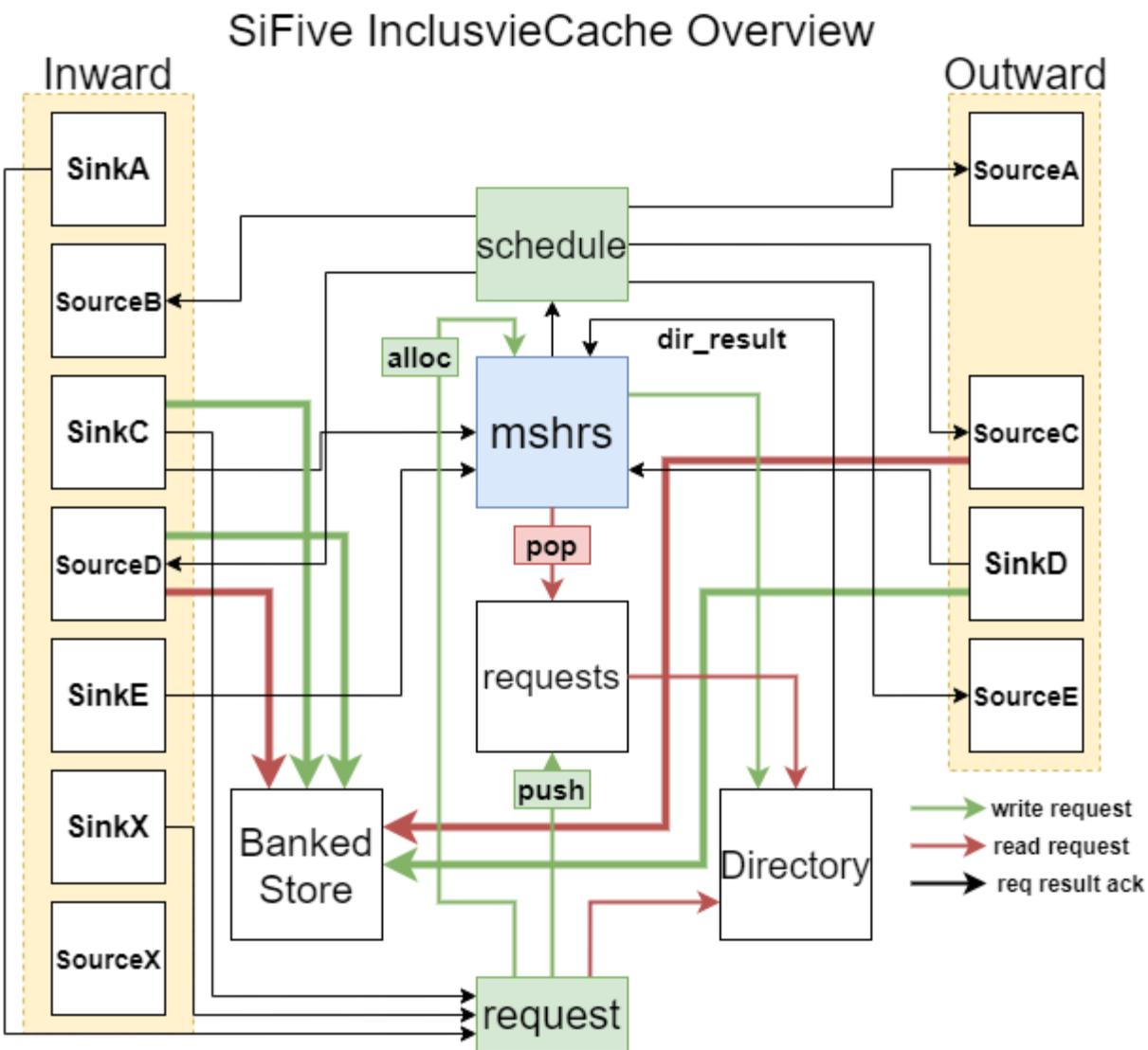
■ SiFive InclusiveCache 功能调整 (概览)

- 支持下游一致性协议
- Uncached Get 不缓存
- 替换算法更改：随机 → 伪LRU

■ 支持下游一致性协议 (概览)

1. 添加 SinkB 接收下游 Probe
2. 增加 MSHR 状态路径
3. 修改 SourceC 支持 ProbeAck

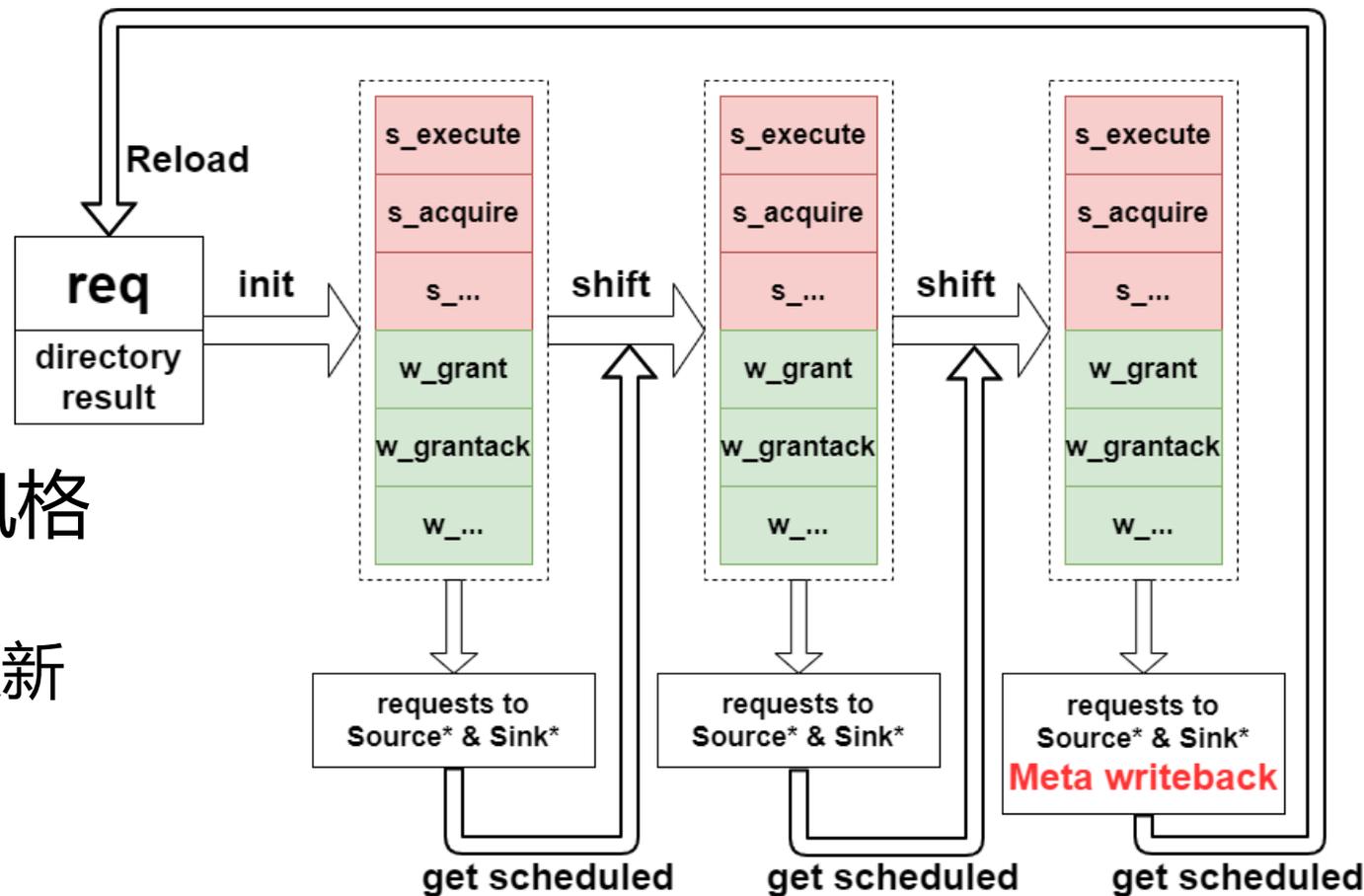
支持下游一致性协议：添加 SinkB



■ 支持下游一致性协议：增加 MSHR 状态路径

• InclusiveCache 状态机设计风格

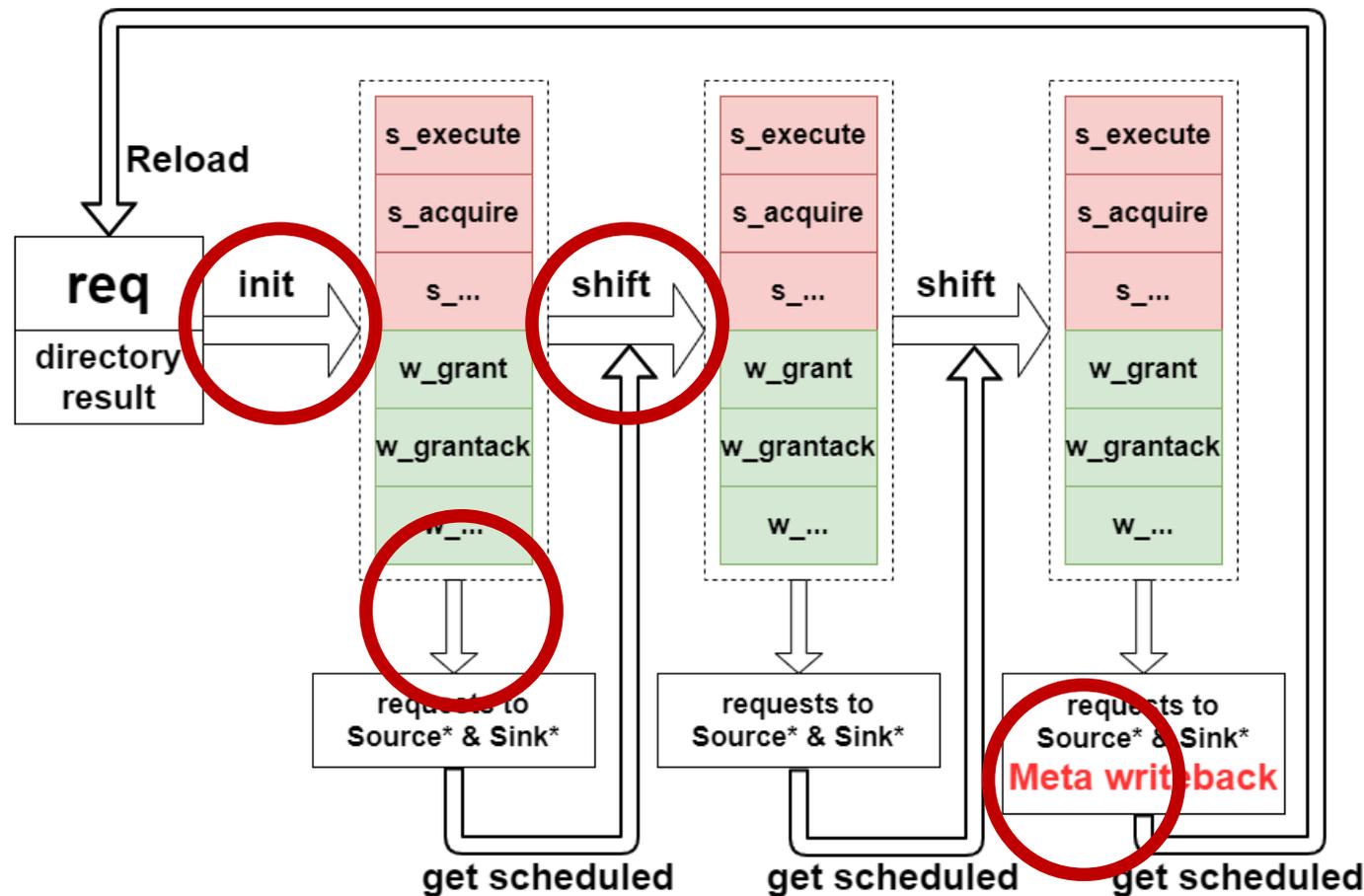
- $s_$ 系列寄存器: 调度时更新
- $w_$ 系列寄存器: 收到 ACK 时更新
- $req = func(s_*, w_*)$
- $state = func(s_*, w_*)$



■ 支持下游一致性协议：增加 MSHR 状态路径

• 修改点

- MSHR 初始化
 - SinkB request case
 - Directory hit/miss
- Miss (None) & Hit (TIP)
 - 请求 SourceC 发 ProbeAck
- Hit (Branch / Trunk)
 - 先请求 SourceB 向上游发 Probe
 - Inclusive
 - 等待 SinkC 收到 ProbeAck
 - 请求 SourceC 发 ProbeAck
- Meta
 - 根据 Probe 类型和缓存情况更新 cache block permission



■ Uncached Get 不缓存

- Uncached Get = miss 的 Get 请求
 - L2/L3 收到的 Get 仅由 L1 指令 cache 发出
 - MMIO 走其他路径
- 不缓存 Uncached Get
 - L1 指令 cache 容量较大
 - L1plus, 自带 L2 指令 cache
 - 不用为指令专门在 L2 分配缓存
 - 软件可通过 fence.i 指令进行同步
 - 提升 L1 数据 cache 对 L2 的利用率

■ 大纲

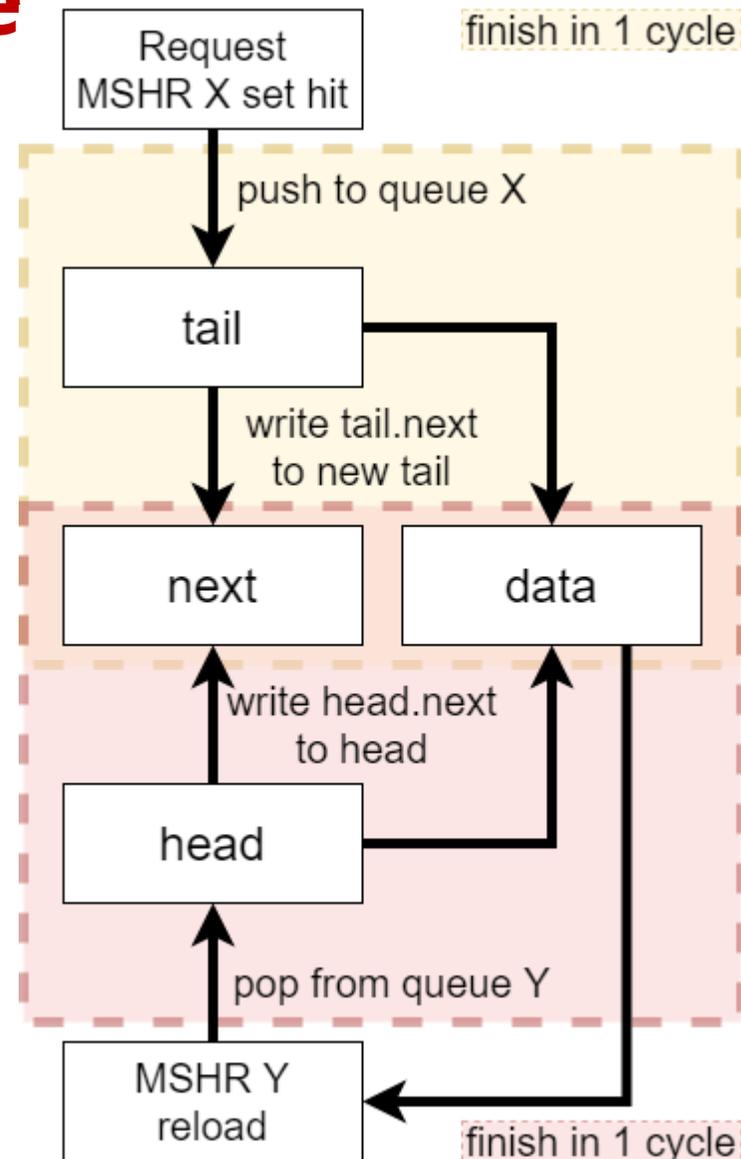
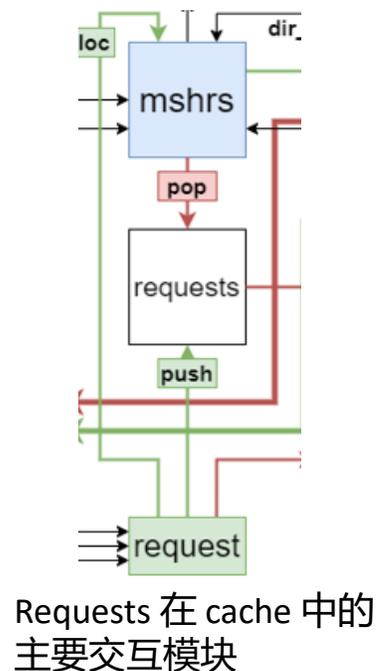
- SiFive InclusiveCache 在香山的应用
- SiFive InclusiveCache 功能调整
- SiFive InclusiveCache 时序调整

■ SiFive InclusiveCache 时序调整

- 移除 requests 的 ListBuffer
- 先比较后选择
- 切分调度逻辑
- 其他修改
 - 启用 dirReg 配置
 - 缩减 MSHR 数量
 - 移除 atomics 模块
 - 切分 grant (refill) 路径

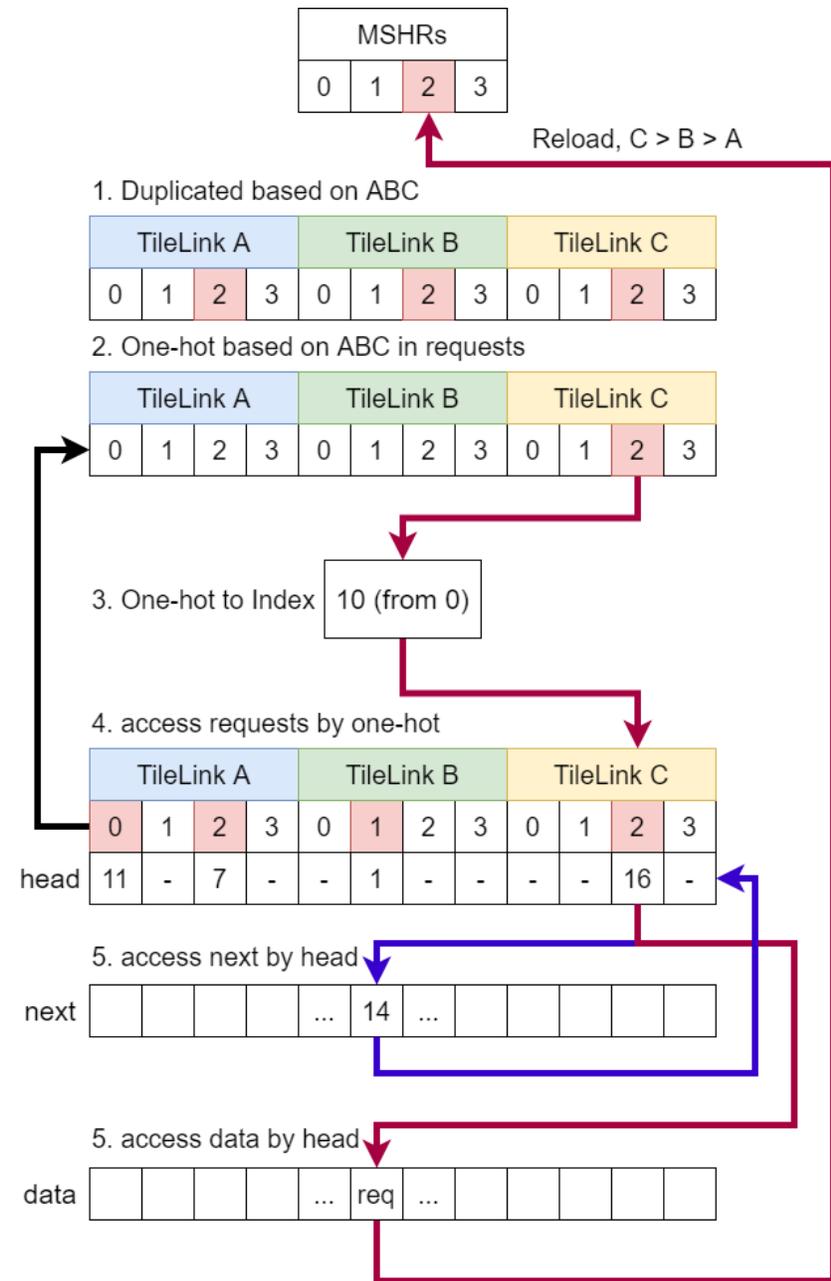
ListBuffer in SiFive InclusiveCache

- 共享结点池动态链表 buffer
 - 每项 MSHR 对应一个 queue
 - 每个 queue 缓冲同 set 请求
 - 动态分配缓冲资源，避免端口阻塞
- 实现特征
 - 4 RAMs: head, tail, next, data
 - pop/push 时需要连续 ram 访问
 - 用 head/tail 的结果作为地址访问 next,data
 - head.read → next.read → head.write
 - 当周期完成



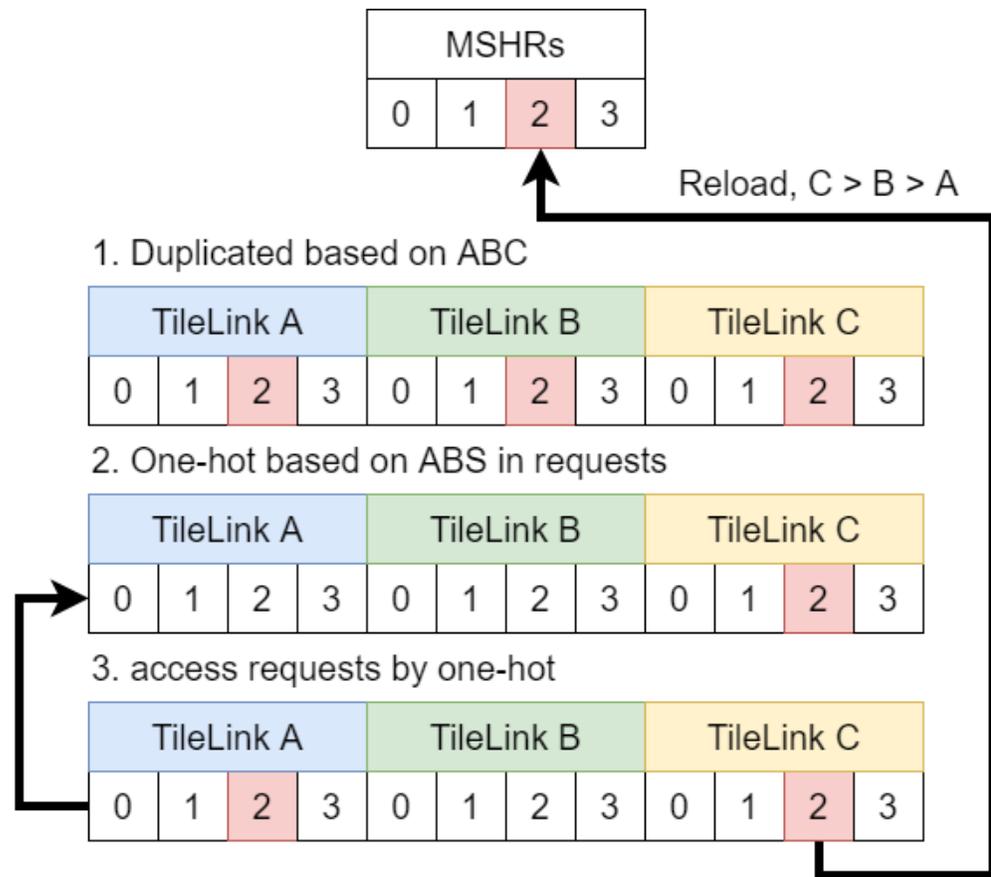
■ 移除 ListBuffer

- ListBuffer: 单周期内级联 RAM 访问
 - 时序难以满足
 - 到中间计算 index 步骤几乎已时序违例
- 关键组合路径
 - MSHR selection+ OneHot + OHToUInt +
 - Pop 更新: head.read → next.read → head.write
 - MSHR alloc: head.read → data.read → ...



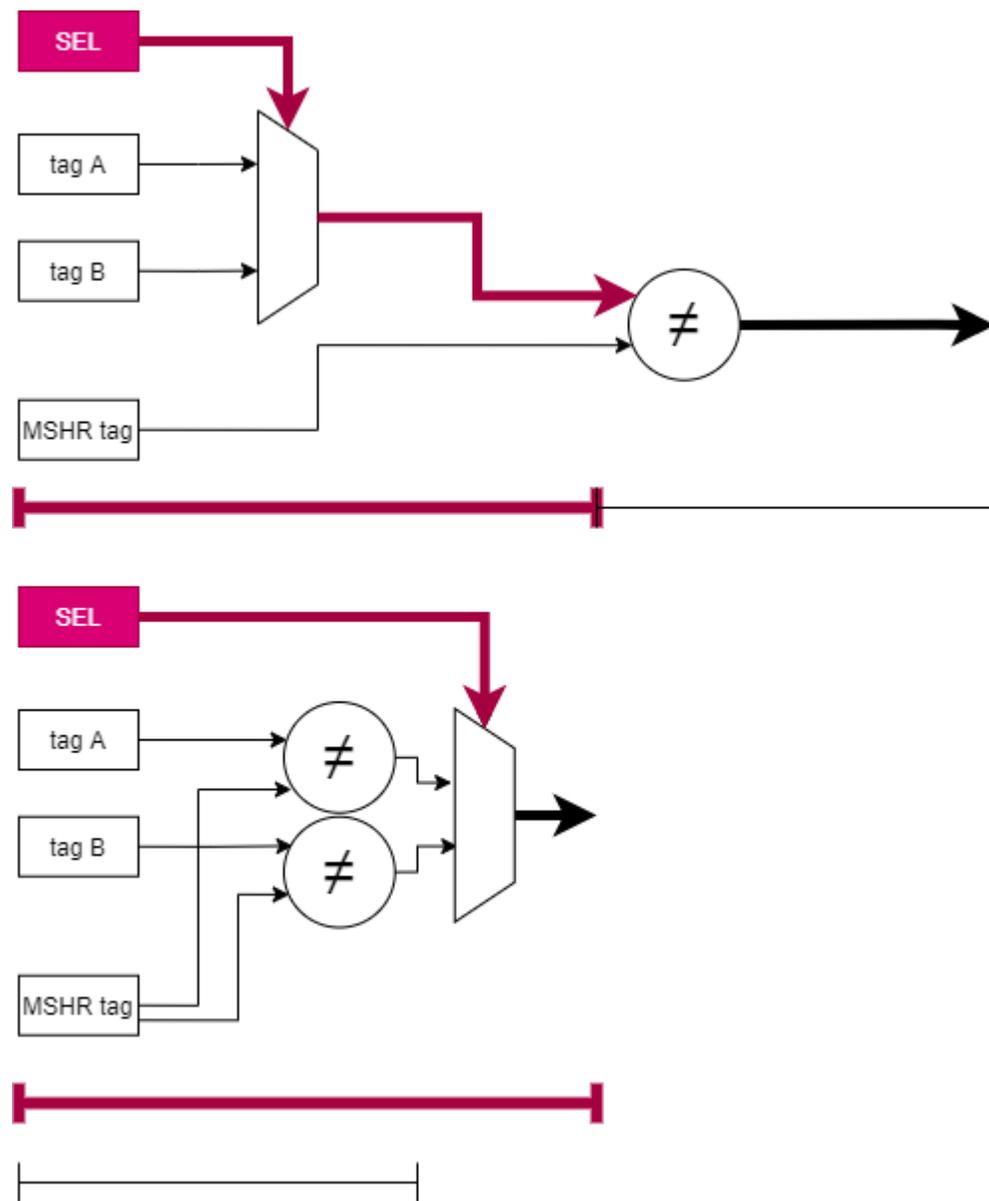
■ 移除 ListBuffer

- 替换为 reg array
 - 每个 MSHR 每个通道仅缓冲 1 项 request
 - 牺牲缓冲性能
- 索引从 index 改为 onehot selection
 - 原本输入的 index 即来自 onehot selection
 - 减少信号转换, 节省不必要的组合路径
 - 依赖于修改后的扁平存储结构



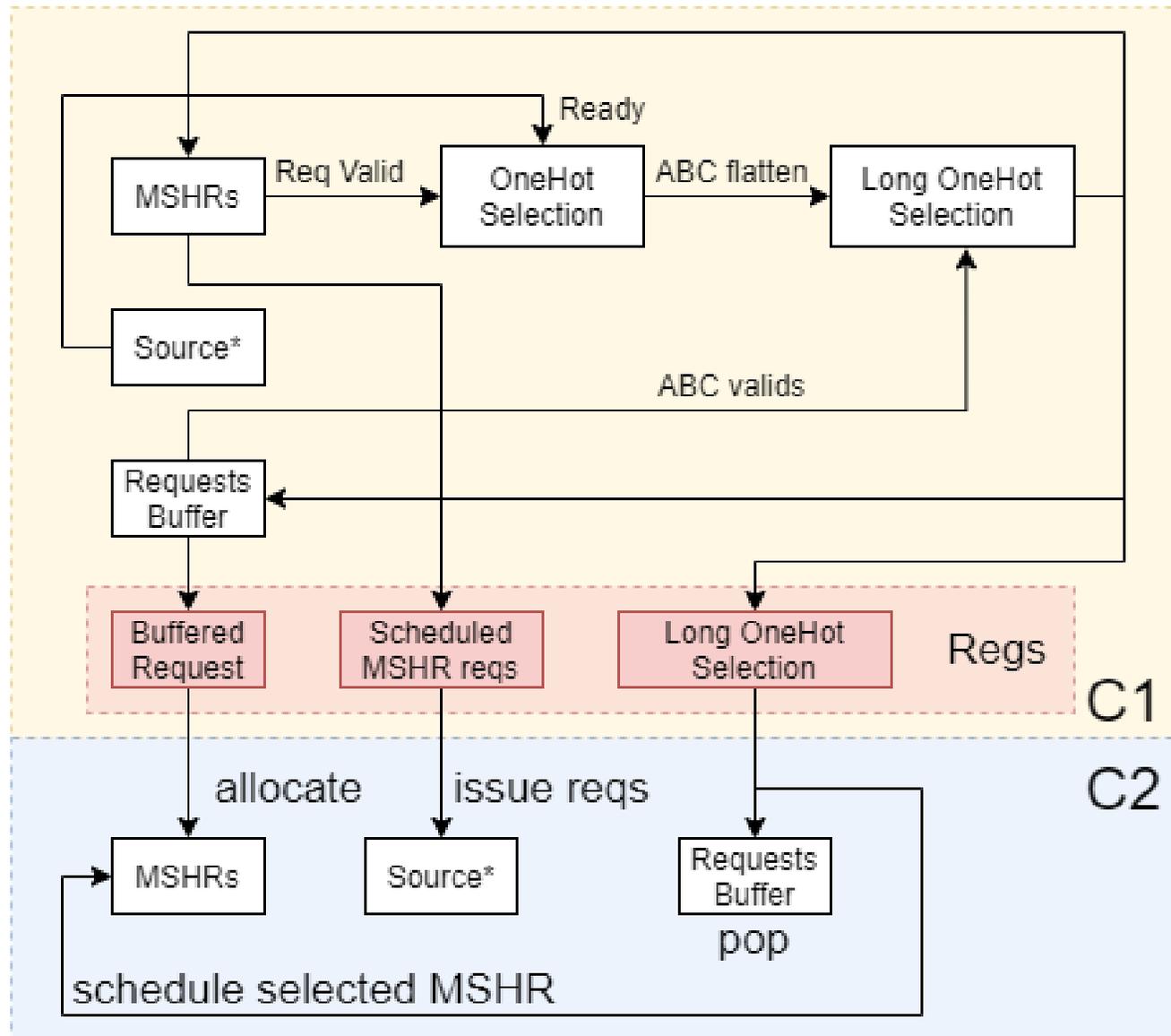
■ 先比较后选择

- 多处需要比较 request 的 set 或 tag
 - 检查冲突与碰撞
 - 比较使用的 XOR 逻辑门延迟大
- 参与比较的 request 经过多层选择
 - Selection 经过一系列比较与运算
 - 路径长，延迟大
 - 用于比较的 XOR 位于关键路径
- 先比较后选择
 - XOR 与 selection 延迟重叠
 - 依赖于扁平的缓冲存储结构调整
 - 空间换时间



■ 切分调度逻辑

- 原版调度逻辑组合路径最长
 - 前述 ListBuffer 路径为其一部分
 - e.g.
 - MSHR selection
 - + ListBuffer pop
 - + MSHR allocate/init
- 切分为两个周期
 - 先锁存选择与索引结果
 - 下周期再发送给各目标模块
 - 非流水（工期原因）



切分后的调度逻辑在两个周期中的信号传输

■ 其余修改

- 启用 dirReg 配置
 - SiFive InclusiveCache 预留配置项
 - 将 Directory 结果锁存 1 拍
- 缩减 MSHR 数量
 - 调度逻辑 fanout 随项数增大, 插入的 clkbuf 增加时延
 - 最终定为 15 项 (与 L1 基本相等)
- 移除 atomics 模块
 - 组合逻辑 Atomics 模块直连 BankedStore RAM 写口, 延迟极大
 - 第一版香山仅在 L1 处理 atomics
- Grant (refill) 路径切分
 - sinkD.src --> way/set --/--> grant_safe --> BankedStore
- 删除 MSHR repeat 机制
 - Repeat 要求 tag 比较, 增加关键路径延迟

■ 总结

• 功能方面

- FPGA 单核 XS+L2+L3 可运行 Linux (ramfs) + SPEC 2006
- FPGA 单核 XS+L2 (流片版本) 可运行 Debian, 含 SDMMC、GMAC 外设

• 时序方面

- 原始版本在 1.5GHz 时序下最严重时违例 0.49ns
 - 1.5GHz 下一周期为 0.67ns
- 经功能与设计裁剪、结构调整, cache 内寄存器间路径满足 1.5GHz 时序
- 最终因其他模块(间)的时序问题和后端优化工期, 频率定位 1.3GHz

```
Startpoint: mods_0/bc_mshr/request_set_reg_3_
             (rising edge-triggered flip-flop clocked by core_clk)
Endpoint:   mods_0/sourceD/s3_bypass_data_REG_1_reg_116_
             (rising edge-triggered flip-flop clocked by core_clk)
Path Group: PG_CORE
Path Type:  max
-----
data required time                                0.28800
data arrival time                                -0.77733
-----
slack (VIOLATED)                                 -0.48933
```

违例路径
时序报告摘录

谢谢