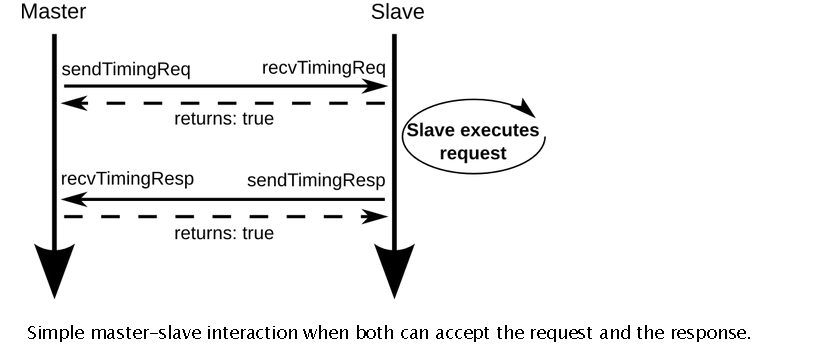
Gem5 Simple\_cache的详细介绍

Simple\_cache的主要组成是：Cache实体CpusidePort，MemsidePort，Stats

前言：

1. Pkt（Packet）分为Timing、Fuctional、Atomic三种



Timing：（retry由slave方发送，retry response由master方发送）

Fuctional：

Atomic：

2.Cache使用unordered\_map,全相连，随机替换策略，阻塞Cache，没有实现一致性

**方法：**

一、一般方法:

1.**get\_port**();

Port &SimpleCache::getPort(const std::string &if\_name, PortID idx)

功能：通过传递的name和port id获得port的别名

2.Simple\_cache构造函数

二、Simple\_cache：：Cpusideport的方法（Cpusideport由ResponsePort集成而来）

3.**sendPacket**()

Void SimpleCache::CPUSidePort::sendPacket(PacketPtr pkt)

功能：发送Timing的response包，如果失败blockedPacket = pkt；

1. **getAddrRanges**()

AddrRangeList SimpleCache::CPUSidePort::getAddrRanges() const

功能：从这个内存部件里面获得地址范围AddrRangeList（list of AddrRange），然后这个函数调用Cache的mem\_port的getAddrRanges()，其实就是mem处获取地址的范围

5.**trySendRetry**()

void SimpleCache::CPUSidePort::trySendRetry();

功能：在blockedPacket 为空且needRetry为true时，向上层发送Retry，needRetry至为false

6.**recvFunctional**()（fuctional pkt不需要response 以及retry）

void SimpleCache::CPUSidePort::recvFunctional(PacketPtr pkt)

功能：接受Functional\_pkt的接口，转去调用handleFunctional方法

7.**recvTimingReq**()

boolSimpleCache::CPUSidePort::recvTimingReq(PacketPtr pkt)

功能：接受Timing\_pkt的接口，blockedPacke||needRetry其一为真时候，不会调用handleRequest，，转去调用handleRequest方法

8.**recvRespRetry**（）

void SimpleCache::CPUSidePort::recvRespRetry()

功能：接受Timing\_pkt\_retry response的接口,将之前的blockedPacket 重新发送出去（通过**sendPacket**方法），如果失败在发送一次retry。

1. Simple\_cache：：Memsideport的方法（Memsideport由RequestPort集成而来）

9.**sendPacket**()

void SimpleCache::MemSidePort::sendPacket(PacketPtr pkt)

功能：向memside方向发送pkt，如果失败blockedPacket = pkt；

10.**recv****TimingResp**()

bool SimpleCache::MemSidePort::recvTimingResp(PacketPtr pkt)

功能：接受Timing\_pkt response的接口，转去执行**handleResponse**方法。

11.**recvReqRetry()**

void SimpleCache::MemSidePort::recvReqRetry()

功能：接受来自memside的Timing\_pkt requestre try的接口，将blockedPacket （之前阻塞的request pkt）再发送一次。

12.**recvRangeChange**()

void SimpleCache::MemSidePort::recvRangeChange()

功能：调用Cache的cpu\_port的

四、SimpleCache的方法

13.**handleRequest()**

Bool SimpleCache::handleRequest(PacketPtr pkt, int port\_id)

功能：处理来自cpu\_side的request请求，如果阻塞（blocked == true）则返回false，将Cache设定为blocked== true状态。调度访问事件**accessTiming**(pkt)。

14.**handleResponse()**

Bool SimpleCache::handleResponse(PacketPtr pkt)

功能：处理来自cpu\_side的response应答，通过pkt的信息插入更新Cache内容（insert(pkt);

），统计miss\_latency,如果我们必须将请求pkt升级到一个完整的Cacheline，现在我们可以使用那个pkt来构造Response。向cpuside发送Response pkt。

15.**sendResponse()**

void SimpleCache::sendResponse(PacketPtr pkt)

功能：向cpuside发送Response pkt。如果需要发送retry（needRetry== true）就发送retry

16.**handleFunctional()**

Void SimpleCache::handleFunctional(PacketPtr pkt)

功能：处理Functional pkt。调用accessFunctional(),如果其为真就（pkt->makeResponse();

），否则向mem\_side发送Functional pkt

17.**accessTiming**()

Void SimpleCache::accessTiming(PacketPtr pkt)

功能：调用**accessFunctional，**如果hit（统计命中数，重构pkt，向cpuside发送response）

，如果miss（统计miss次数，保留miss当前时间，重构pkt，确保块地址对齐的情况下向memside发送request pkt）

18.**accessFunctional**()

Bool SimpleCache::accessFunctional(PacketPtr pkt)

功能：访问cache，判断是否命中，如果命中，根据pkt是read或者是write，使用pkt的方法对Cache空间进行写或读。

1. **Insert**()

Void SimpleCache::insert(PacketPtr pkt)

功能：在接受来自memside 的response时调用，是将mem中的块写入Cache，若Cache满了，则使用随机算法，将任意块替换出去。