Implementation Framework And Present Support Analysis Of Gem5

gem5 实现框架分析和 RISC-V 支持现状

主讲人: 卢睿博

PLCT 实验室

January 20, 2021

目录

目录

gem5 介绍

what is gem5?

The gem5 simulator is a modular platform for computer-system architecture research, encompassing system-level architecture as well as processor microarchitecture.

gem5 的优势

- 支持功能和时序仿真
- 拥有两种仿真模式：
  - 全系统级仿真和系统调用级的仿真
- 支持多种的指令架构
- 模块化管理系统模型
  - Processor Cores + Memory Hierarchy + I/O Systems
- 用 C++ 组织后端，Python 组织前端，开源

gem5 的组件

Michigan m5+Wisconsin GEMS = gem5

Key Features

- **Execution modes**：System-call Emulation(**SE**), Full System(**FS**)
- **Available binaries**:gem5.debug/opt/prof/perf/fast
- **ISAs**:ALPHA,ARM,MIPS,POWER,RISC-V,SPARC,x86,and NULL
- **CPU models**: AtomicSimple, TimingSimple, InOrder, and O3
- **Memory models**: Classic,Ruby
- **Interconnection networks**: Simple, Garnet

目录

Simulation Flow

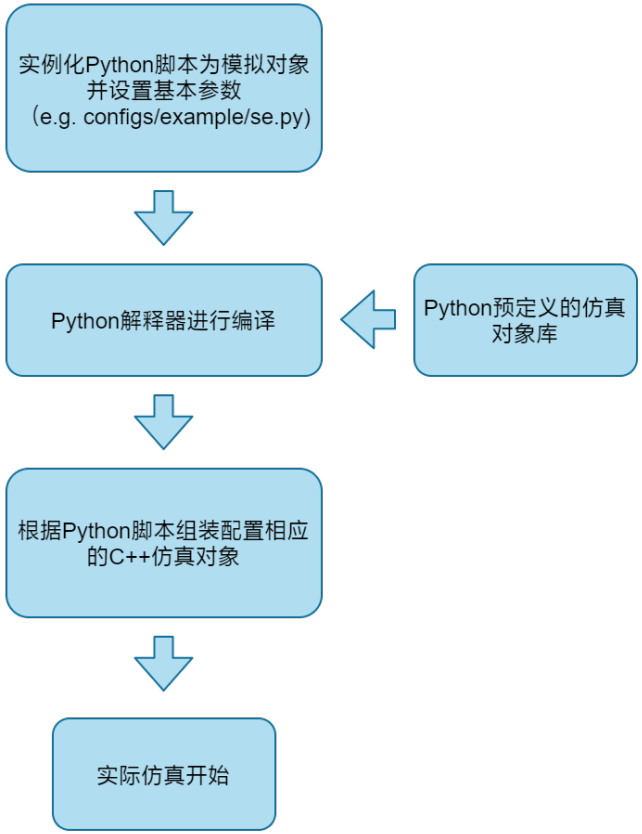


Figure: Python Simulation Flow

Call Sequence Hello

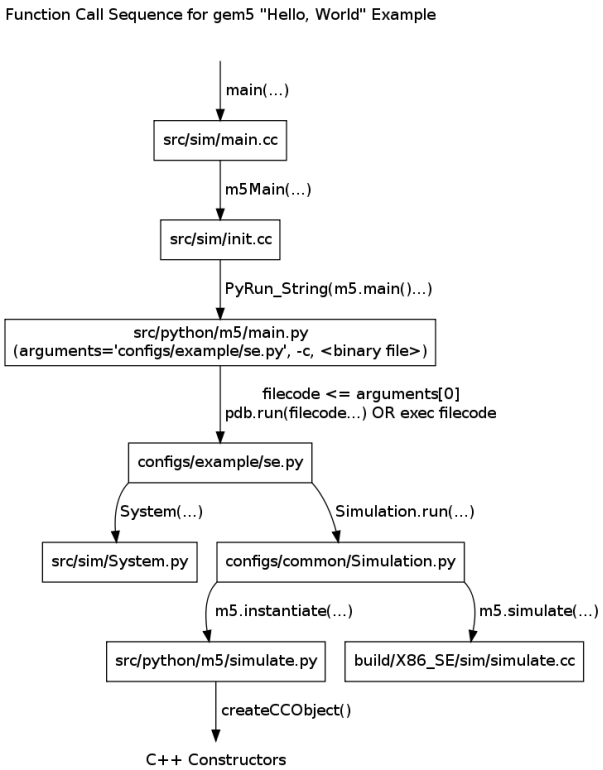


Figure: Python Simulation Flow

目录

工作模式

Full system(FS)

- 用于引导操作系统
- 裸机模型，包括各种设备
- 中断，异常，特权指令，错误处理程序
- 模拟 UART 协议串口输出
- 模拟帧缓冲区输出

Syscall emulation(SE)

- 用于运行独立的应用，或不同平台的指令集
- ISA + 系统调用模型
- 通过调用主机模拟系统调用
- 无调度的简单的地址转换模型

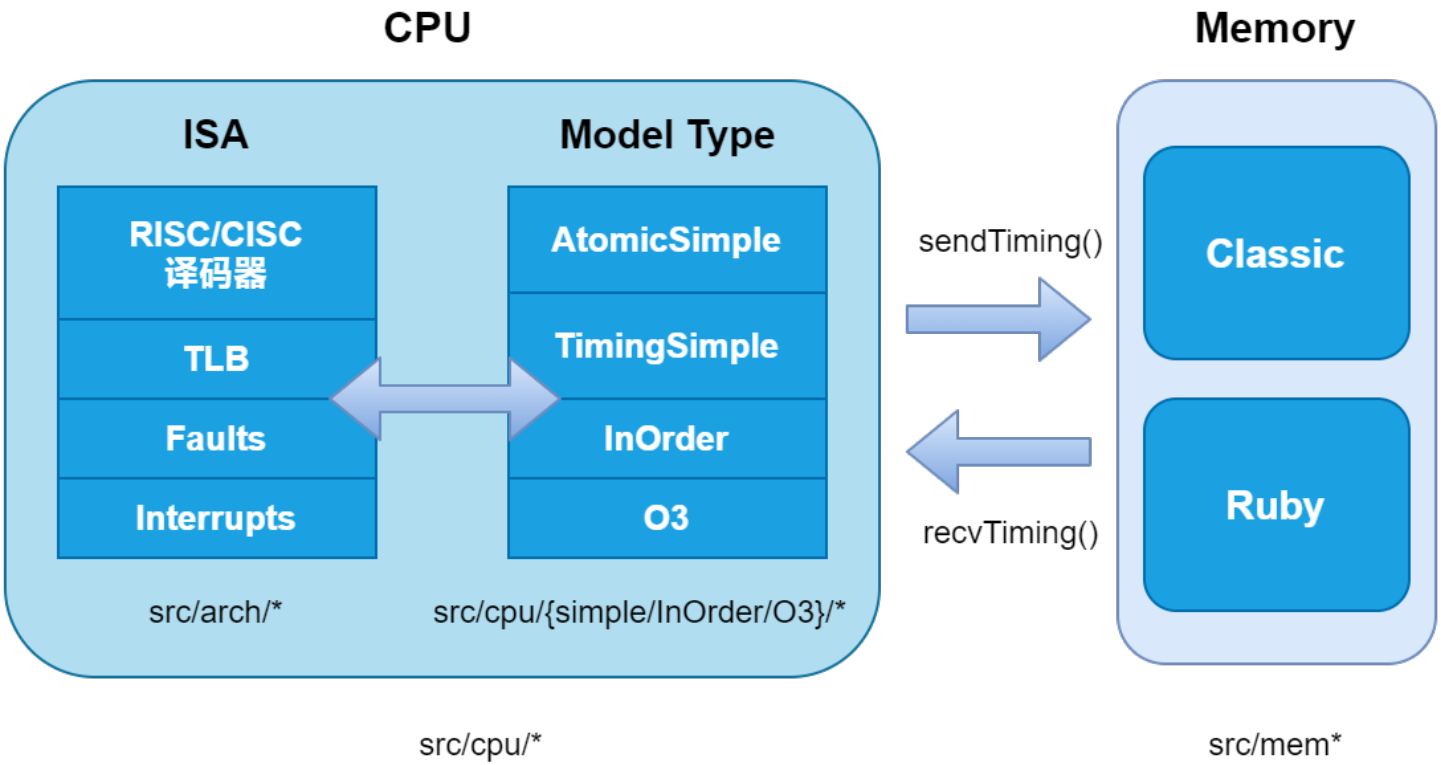
目录

Overview

- 支持的 CPU 模型
  - AtomicSimple CPU
  - TimingSimple CPU
  - InOrder CPU
  - O3 CPU
- CPU 模型的内部结构
  - 基本参数
  - 时间缓冲
  - 关键结构

CPU 模型-系统级别视图 (System Level View)

CPU 模型被设计为可以和任意的指令集和存储系统“热插拔”



Supported CPU 模型

Simple CPUs

- 功能性的，顺序性的
- AtomicSimple and TimingSimple CPU
- 快速的功能性的模拟：**twolf** 基准测试上 2.9M and 1.2M ips
- 适用于不需要细节 CPU 模型模拟的情形：
  - 缓存预热期，驱动主机的客户端程序，测试程序

Detailed CPUs

- 时序的，参数化的流水线模型
- Minor,Trace,O3
- 比上一个慢，**twolf** 基准测试 200K ips
  - 模拟了每个流水线阶段的时序
  - 同时强调模拟的时序和指令的执行以保证准确性
  - 对于缓存一致性，I/O，多处理器等的研究十分重要

AtomicSimpleCPU src/cpu/simple/atomic.cc,hh

- 每个 CPU **tick()** 执行一个指令的所有必要的操作
- 访存是原子的- **atomic**
- 最快的功能性的模拟

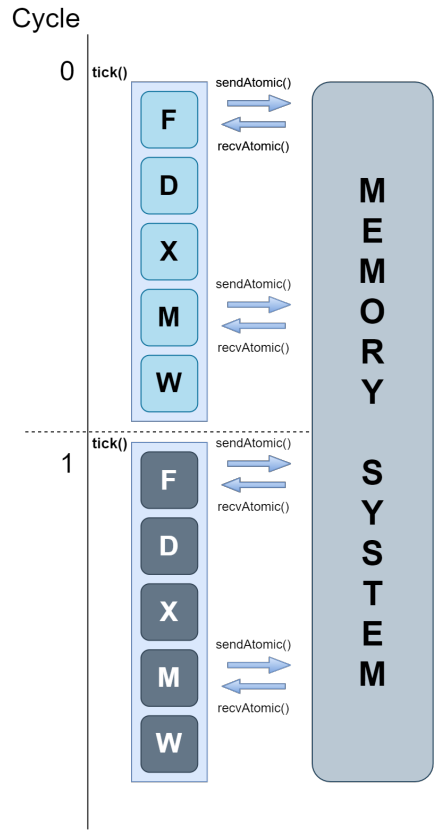


Figure: AtomicSimpleCpu

TimingSimpleCPU src/cpu/simple/timing.cc,hh

- CPU 在内存返回前都是等待状态
- 访存是时序的-**Timing**
- 快速的，提供不同时序级别的模拟

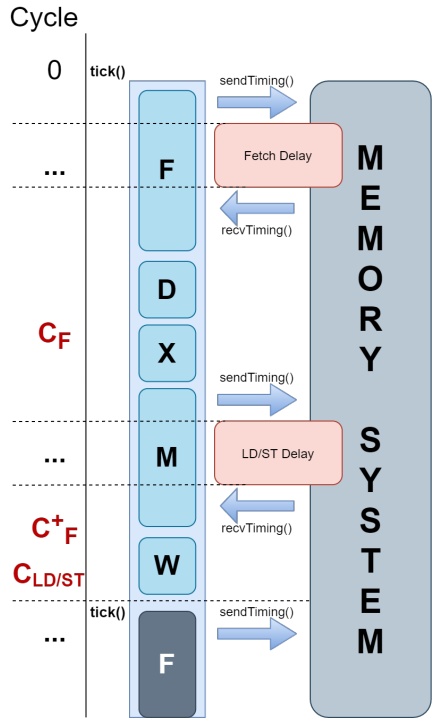


Figure: TimingSimpleCpu

MinorCPU src/cpu/minor/*.cc,hh

- 顺序执行的固定流水线的 CPU 模型
- 5 级流水线
  - Fetch1 : 取到缓存行
  - Fetch2 : 将行分解为指令
  - Decode : 将指令分解为微操作
  - Execute: 指令执行和数据操作

Figure: Stage Of MinorCpu

O3 CPU src/cpu/o3/*.cc,hh

- 细致的乱序 CPU 模型
- Fetch,Decode,Rename,IEW,Commit
- **IEW**:Issue,Execute 和 Writeback
- **Execute in Execute**
- **Template Policies**
- **ISA** 独立性
- 分支和内存依赖的预测

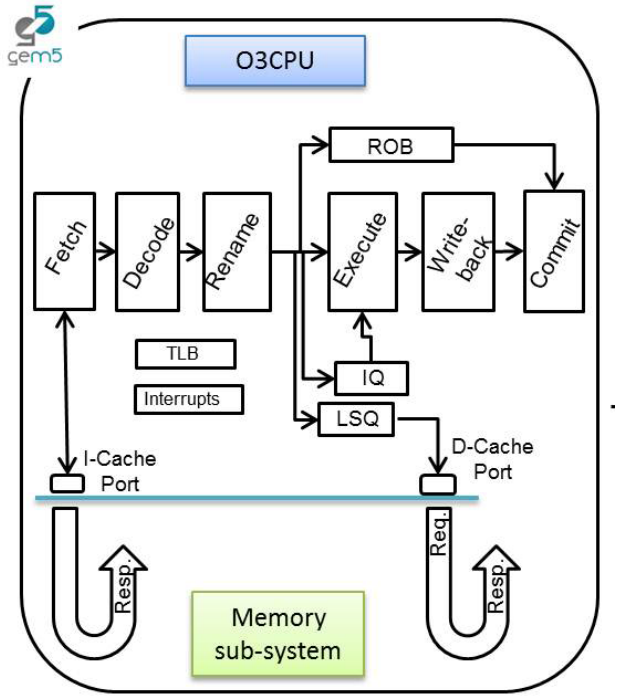


Figure: O3Cpu

目录

Overview

- 通用存储系统
  - Ports,Connections
  - Packets,Requests
  - Access Types: Atomic, Timing, Functional
- 两种存储系统模型
  - Classic
  - Ruby

Classic Memory System

- (几乎) 适用于任何配置，牺牲了小部分真实性，可配置性
- 利用 **MOESI** 协议保证一致性
- 使用 express snoops 帧防止泛洪

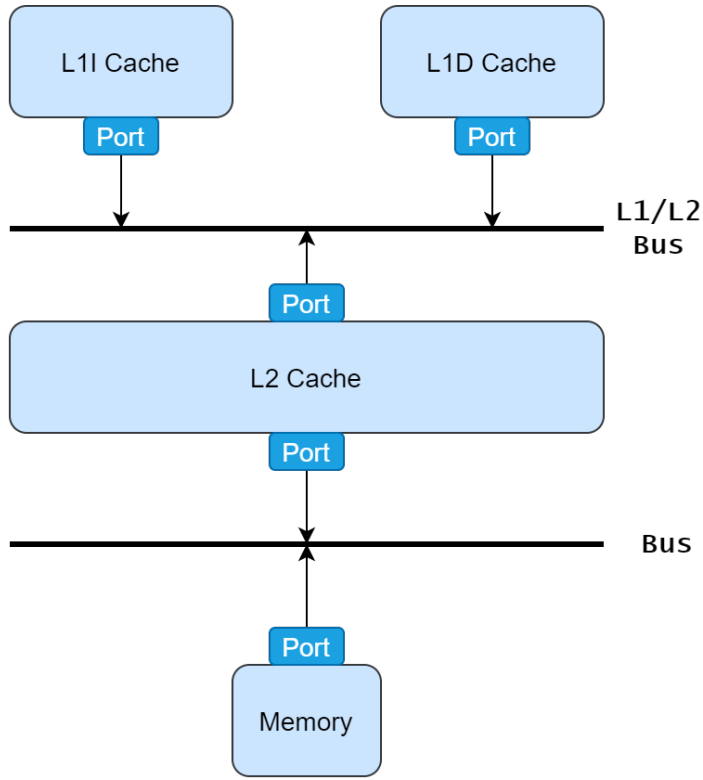
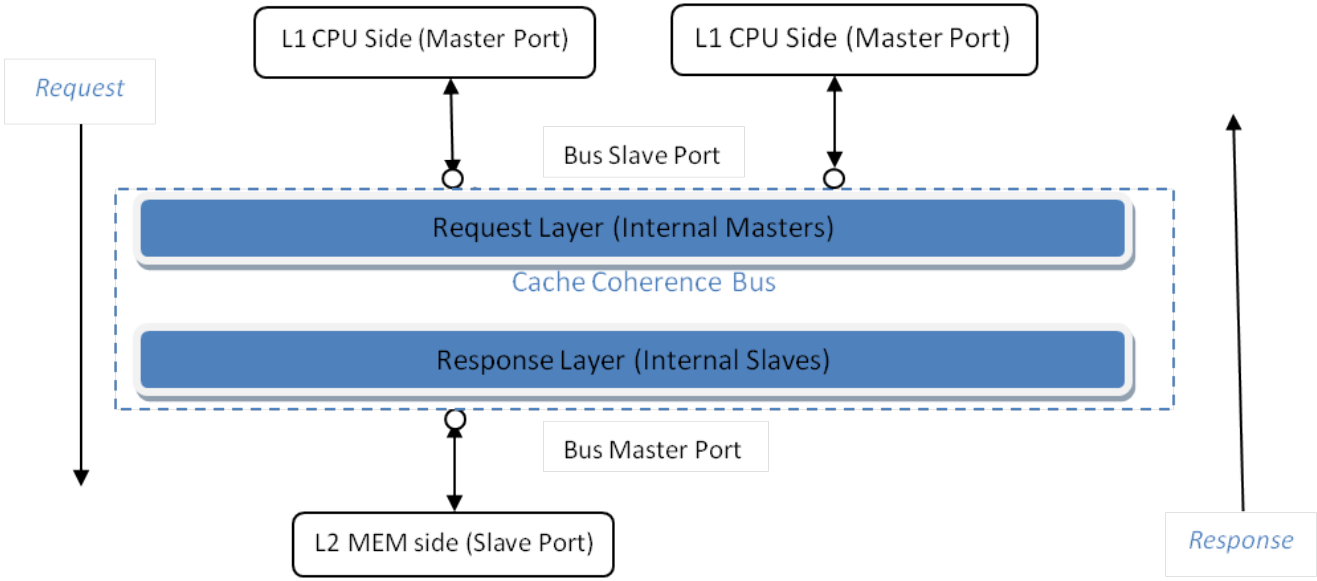


Figure: Classic Memory System

Classic Caches

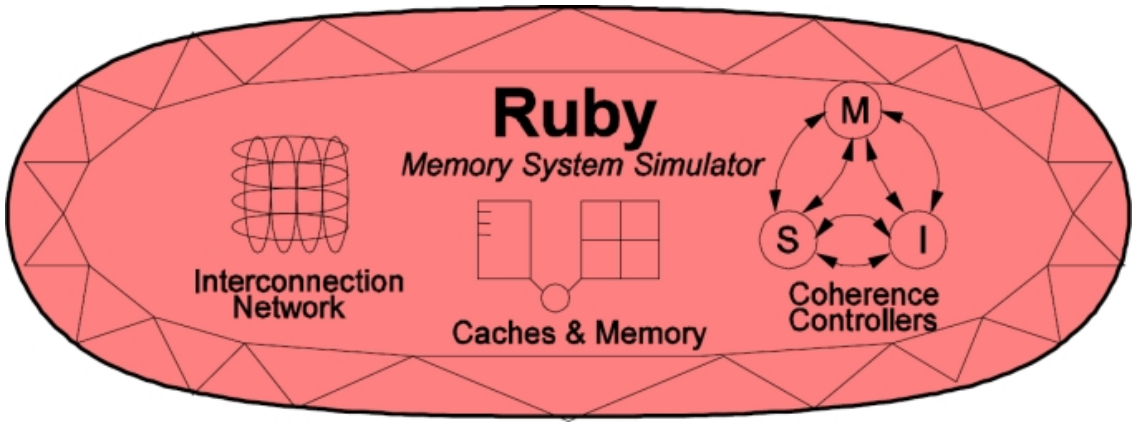
- 两种包：内存映射包和侦听包
- 两种形式：Request 和 Response



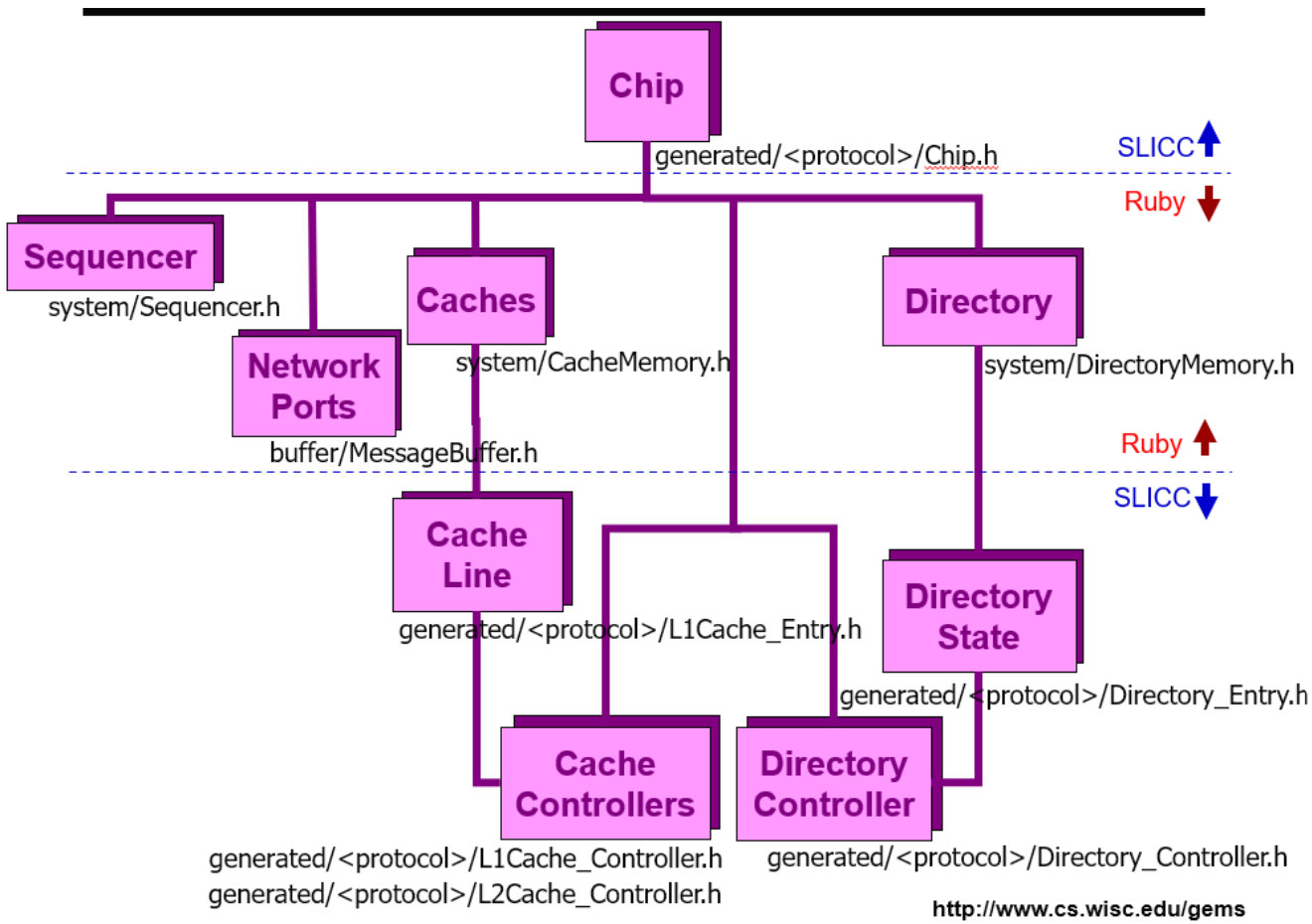
Overview

Ruby 为内存子系统实现了详细的仿真模型。实现了各种替换策略，一致性协议实现，互连网络，DMA 和内存控制器，请求和处理响应的定序器等。这些模型是模块化，灵活且高度可配置的。模型设计的主要思想是：

- 关注点分离-例如网络拓扑与实现分开指定
- 高可配置性-几乎可以控制存储器的任何方面。
- 快速原型化-定义了 **SLICC** 用于指定各种控制器的功能。



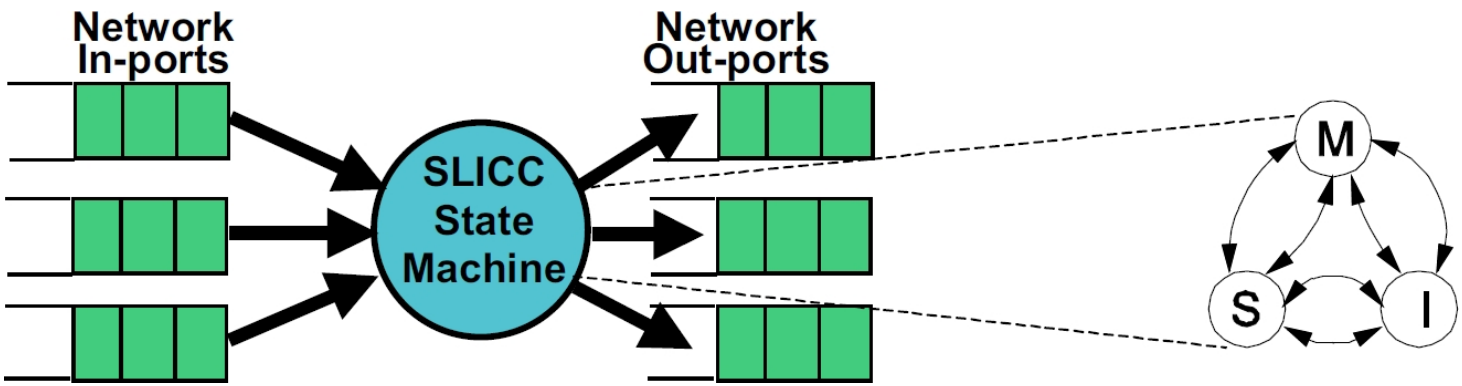
Ruby Software Structure



SLICC

SLICC-Specification Language for Implementing Cache Coherence

- 支持多种缓存一致性协议
- 指定状态机的行为
- 组合内存模型的某些组件



小组件

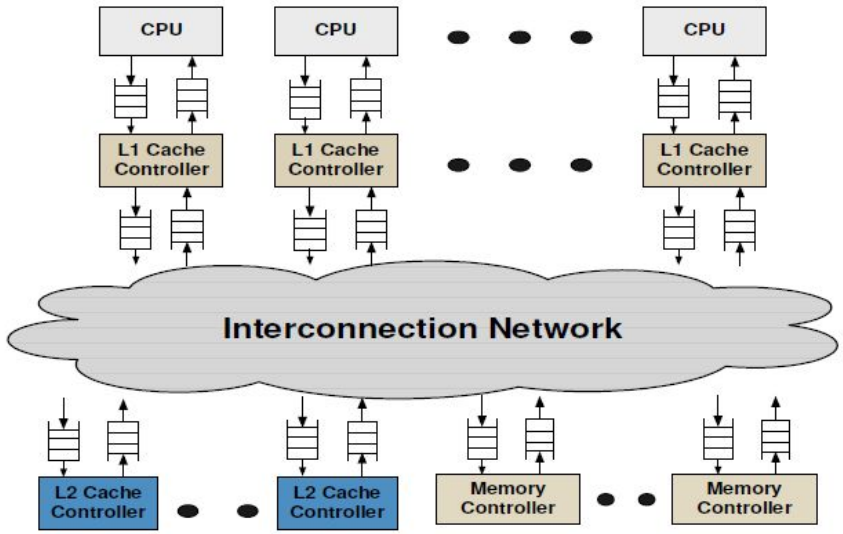
- Sequencer : 定序器
  发送内存请求并返回内存相应，每个硬件线程 (或内核) 都有一个
- Cache Memory: 高速缓存
  可设置大小，关联性，替换策略, 实例化为 L1,L2,L3
- Replacement Policies: 替换策略
  模块化，不同的高速缓存实例可以使用其选择的不同替换策略
- Memory Controller: 存储控制器
  负责模拟和处理未命中所有缓存的请求

InterConnection Network

互连网络将内存系统的各个组件（缓存，内存，DMA）连接在一起

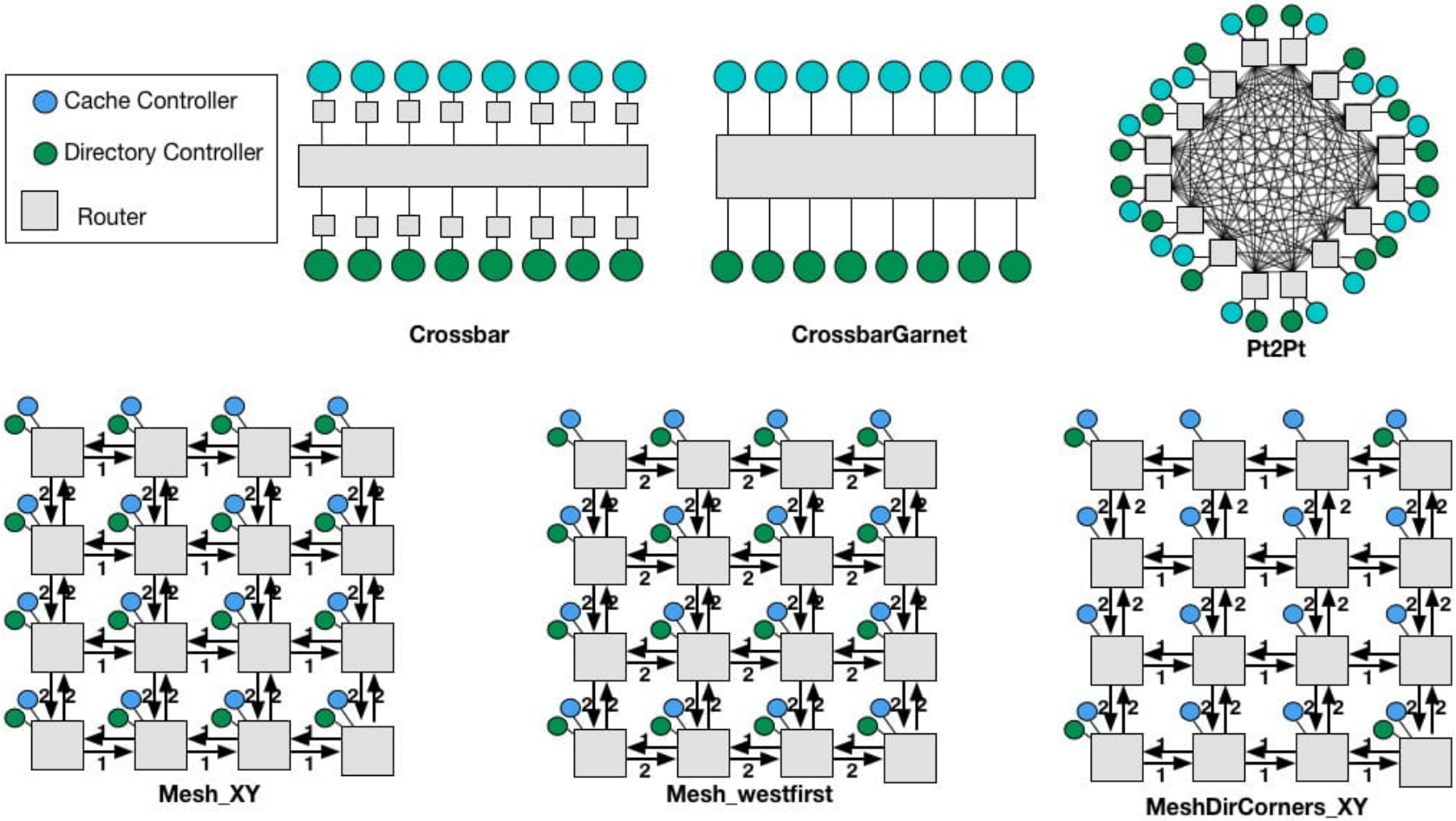
关键组件：

- Topology：拓扑结构
- Routing: 路由协议
- Flow Control: 流量控制
- Router Microarchitecture: 处理器微架构



可以用外部模拟器 **TOPAZ** 代替互连网络。

Topologies 拓扑结构



目录结构

src/mem

- protocols : SLICC 的协议说明
- slicc : SLICC 解析器和代码生成器的实现
- ruby
  - common: 常用的数据结构，例如地址，直方图，数据块
  - filters: 各种 Bloom 过滤器（GEMS 中的过期代码）
  - network: 互连实现，拓扑规范，功率计算，消息缓冲区
  - profiler: 分析缓存事件，内存控制器事件
  - recorder: 缓存预热和访问跟踪记录
  - slicc-interface: 消息数据结构，各种映射（例如，地址到目录节点），实用程序功能（例如，address 和 int 之间的转换
  - structures: 协议无关组件-CacheMemory，DirectoryMemory
  - system: 连接组件-Sequencer，RubyPort，RubySystem

目录

RISC-V in gem5

- RISC-V 初始支持 [A. Roelke, CARRV 2017]
  RV64GC，单核系统，Systemcall emulation (SE) mode
- 多核系统支持 [CARRV 2018]
- 基础 Baremetal Fullsystem 及示例配置
- 不支持 Linux，Android 等操作系统的 Fullsystem mode
- 基于 riscv-tests 的指令测试集

RISC-V Resources

Table: Risc-v resources sources for gem5 20.1

Resource	Compiled/Built Resource
riscv-tests	dhryston.riscv, median.riscv, mm.riscv, mt-matmul.riscv, mt-vvadd.riscv, multiply.riscv, pmp.riscv, qsort.riscv, rsort.riscv, spmv.riscv, towers.riscv, vvadd.riscv
insttests	insttest-rv64a, insttest-rv64c, insttest-rv64d, insttest-rv64f, insttest-rv64i, insttest-rv64m
simple/pthread (riscv64)	test-pthread-create-seq, test-pthread-create-para, test-pthread-mutex, test-atomic, test-pthread-cond, test-std-thread, test-std-mutex, test-std-condition-variable
simple/hello	riscv-executable

目录

References

- gem5 documentation
  http://www.gem5.org/documentation
- CO-EN4730 - lecture08
  http://www.dejazzer.com/coen4730/doc/lecture08_supplemental_gem5.pdf
- RISC-V Vector 及定制指令 Gem5 实现
  http://crva.ict.ac.cn/crvs2020/index/slides/1-4.pdf
- Improving Support for RISC-V in gem5
  https://content.riscv.org/wp-content/uploads/2017/12/Wed1648_gem5_Roelke.pdf
- Implementing the RISC-V ISA in gem5
  https://carrv.github.io/2017/slides/roelke-risc5-carrv2017-slides.pdf
- Mutifacet GEMS
  https://research.cs.wisc.edu/gems/isca_tutorial.ppt

Thank you

Thank you for listening!

Q&A

Questions?