



# 香山处理器 Tutorial

---

陈国凯 陈熙 胡轩 李昕 张梓悦

中科院计算所

2022.08.25



# 目录

- 一. 香山是什么
- 二. 如何使用香山
- 三. 如何仿真运行香山
- 四. 香山仿真框架
- 五. 香山开发工具




# 目录

- 一. 香山是什么
- 二. 如何使用香山
- 三. 如何仿真运行香山
- 四. 香山仿真框架
- 五. 香山开发工具


# 香山处理器概述

- 一款高性能处理器，目前已迭代两代
  - 第一代雁栖湖架构支持 RV64GC 指令集
  - 第二代南湖架构支持 RV64GCBK 指令集
- 使用 Chisel 硬件设计语言实现，参数化设计
- 包含差分测试框架、仿真快照、检查点等开发工具
- 建立了包含设计、实现、验证在内全开源工具敏捷开发流程




**XiangShan**  
Open-source high-performance RISC-V processor  
✉ xiangshan-all@ict.ac.cn  
<https://github.com/OpenXiangShan>

 Overview

 Repositories 42













 Projects

 Packages



# 项目结构

- debug: 一些跑测试常用命令的脚本
- scripts: 仿真使用的脚本, 如编译控制、SRAM 替换
- src: 香山 RTL 代码 (不含FPU、L2/L3\$)
  - main/scala
    - xiangshan: 香山设计
    - bus & utils .....: 配合核心设计的辅助代码
  - Test/scala: 除法器测试及仿真顶层
- fudian: 浮点运算单元结构与测试代码
- huancun: L2/L3 缓存结构与测试代码
- ready-to-run: 预编译的 nemu-so, 以及部分 workload
- other submodules
  - difftest: 差分测试
  - .....

	.github
	debug
	<a href="#">difftest @ bafef94</a>
	<a href="#">fudian @ 3dd05b0</a>
	<a href="#">huancun @ a47ce2c</a>
	images
	project
	<a href="#">ready-to-run @ ec61625</a>
	<a href="#">rocket-chip @ 254ebf7</a>
	scripts
	src
	tools/readmemh



# 逻辑结构

## • 前端

- BPU, FTQ, IFU, I\$, IBuffer

## • 后端

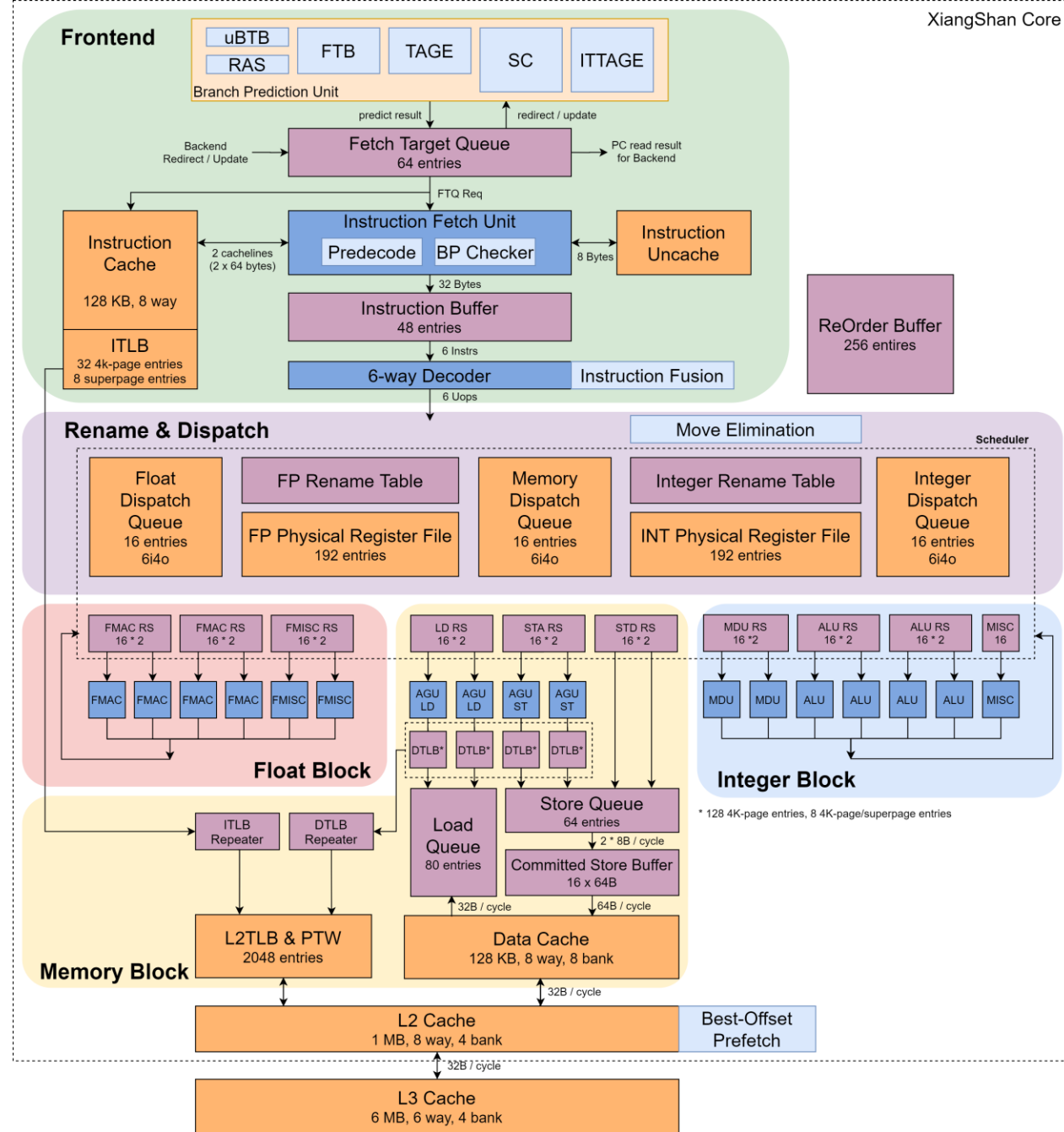
- CtrlBlock, IntBlock, FloatBlock

## • 访存

- Memblock, MMU, D\$

## • 缓存

- L2\$, L3\$

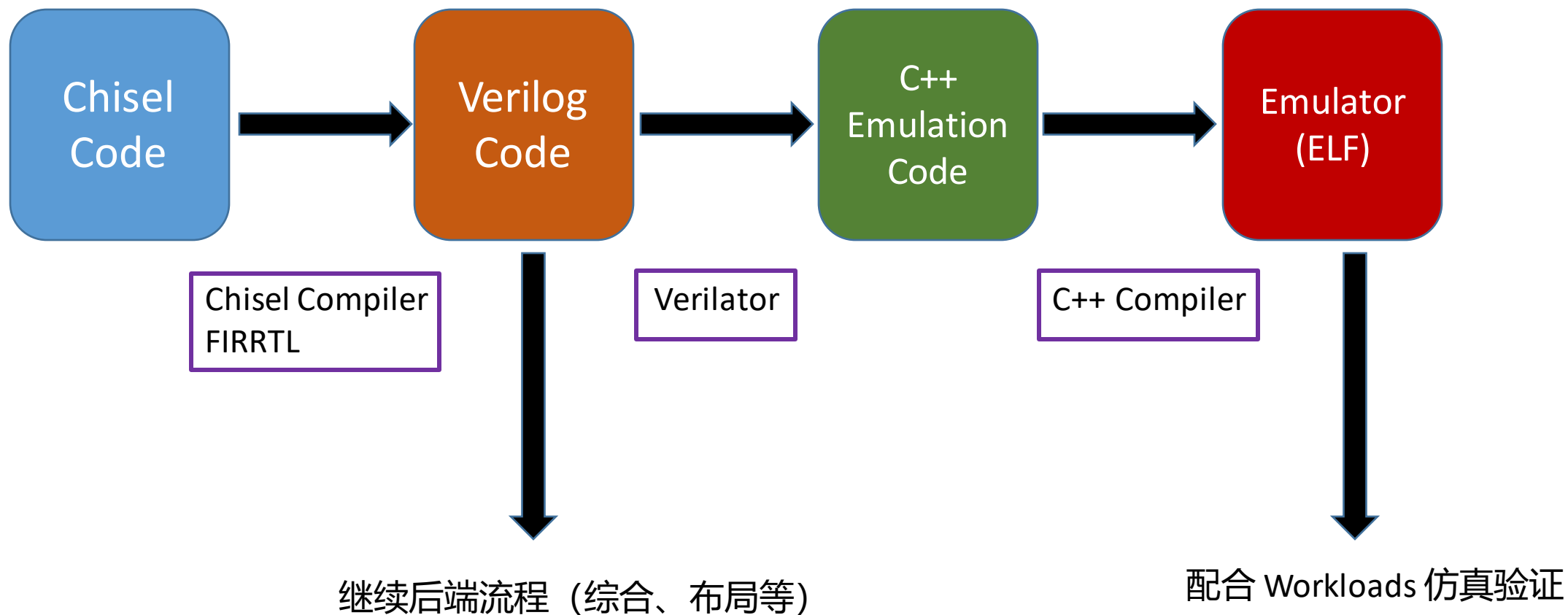




# 目录

- 一. 香山是什么
- 二. 如何使用香山
- 三. 如何仿真运行香山
- 四. 香山仿真框架
- 五. 香山开发工具

# Chisel 工作流程





# 参数系统

- Config:
  - `src/main/scala/xiangshan/Parameters.scala`
    - 隐式传参
    - 控制核内的大部分配置：
      - 分支预测器的规模和组合
      - 各种队列的大小，如重定序缓冲，保留站，Store Buffer，Cache 和 TLB 的大小
  - `src/main/scala/top/Configs.scala`
    - default config: 默认配置，编译时间较长 ~ 1h
    - minimal config: 迷你配置，编译时间较短 ~ 20min
    - make emu **CONFIG=MinimalConfig**



# 目录

- 一. 香山是什么
- 二. 如何使用香山
- 三. 如何仿真运行香山
- 四. 香山仿真框架
- 五. 香山开发工具

# 香山编译

- export NOOP\_HOME=/PATH/TO/XiangShan
- export NEMU\_HOME=/PATH/TO/NEMU
- make init # 初始化子模块
- make verilog # 生成香山的verilog文件 NUM\_CORES=2 可生成双核版
- make emu # 生成香山仿真的emu文件 NUM\_CORES=2 可生成双核版
- make emu CONFIG=MinimalConfig EMU\_THREADS=8 EMU\_TRACE=1  
# MinimalConfig, 8线程仿真 允许导出波形

- AM (Abstract Machine) 向程序提供了裸机运行时环境
- 提供最简单的运行时环境，以及printf, memcpy等常用库函数
- 支持逻辑环境下运行coremark, microbench等工作load
- 可编译自定义裸机程序
- 编译和运行
  - `make ARCH=riscv64-xs -j`
  - `path/to/emu -i coremark.bin`



- [illegible]

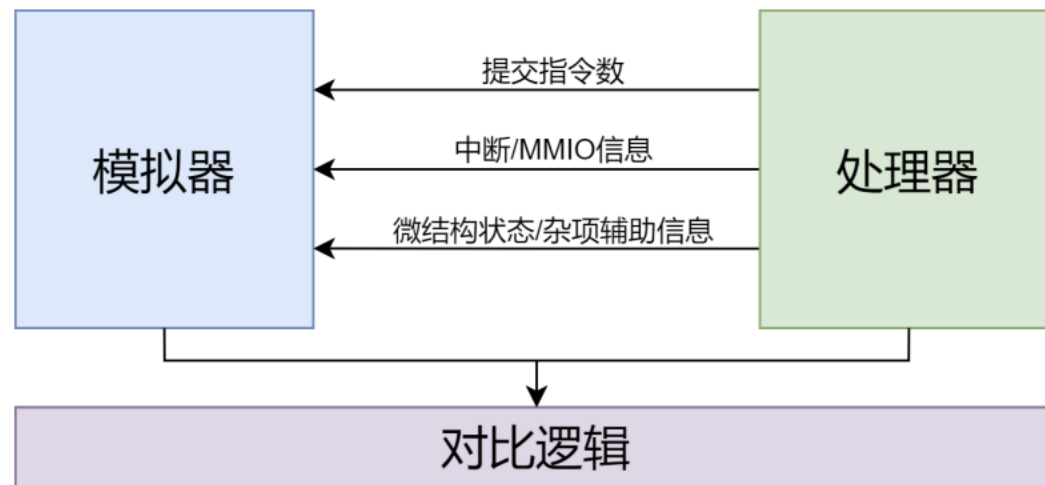


# 目录

- 一. 香山是什么
- 二. 如何使用香山
- 三. 如何仿真运行香山
- 四. 香山仿真框架
- 五. 香山开发工具

# DiffTest: 在线比对框架

- 仿真工具: Verilator
- 差分测试框架:
  - 与 NEMU 模拟器做运行时比对
  - 当处理器和模拟器行为不一致时, 仿真程序将会停止



```
while (1) {  
    icnt = cpu_step();  
    nemu_step(icnt);  
    r1s = cpu_getregs();  
    r2s = nemu_getregs();  
    if (r1s != r2s) { abort(); }  
}
```

在线验证机制

- NEMU (NJU Emulator) 是一款执行速度媲美QEMU的解释器
- 两种模式： 执行模式和Difftest模式
- 执行模式
  - `make riscv64-xs_defconfig -j`
- Difftest模式
  - `make riscv64-xs-ref_defconfig -j`
- 使用 `make menuconfig` 自定义设置





# 目录

- 一. 香山是什么
- 二. 如何使用香山
- 三. 如何仿真运行香山
- 四. 香山仿真框架
- 五. 香山开发工具

# 生成 Checkpoint

- Uniform Checkpoint
  - 每隔N条指令打一个点

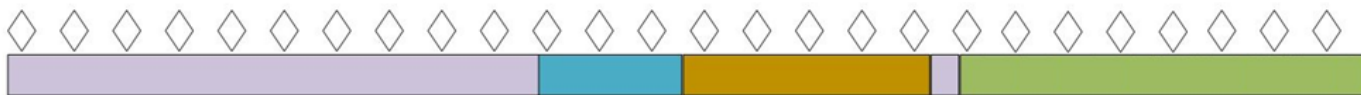
- Simpoint Checkpoint
  - 根据程序特性选点
  - 分为三步：

- Profiling：执行一轮 workload，收集程序行为信息
- Clustering：进行聚类，得到权重最高的多个程序片段
- Checkpointing：再执行一轮 workload，根据聚类的结果生成对应的点

- 生成的 Checkpoint 可供 NEMU 和香山核仿真运行
- 可在Linux Kernel上运行SPEC生成Simpoint，快速评测性能

基于快进的采样 (典型：SMARTS)

- 均一采样
- 每个采样点较小 (5k~50k)



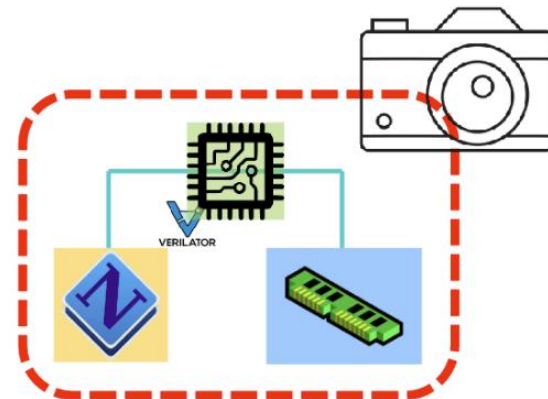
基于检查点的采样 (典型：SimPoint)

- 选择性、带权重采样
- 每个采样点较大 (50M~200M)



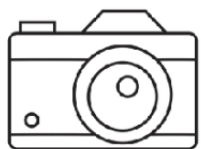
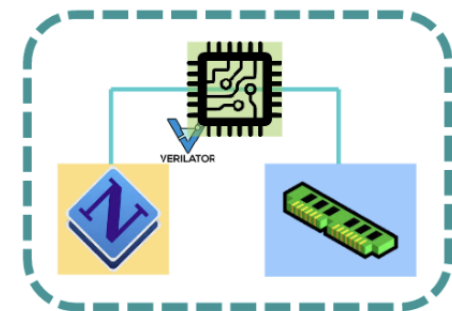
# 🔥 LightSSS: 一种基于内存的轻量级仿真快照

- Light-weight Simulation SnapShot
- 核心思想:
  - 从进程抽象视角看仿真状态，每隔一段时间用 fork 对进程做快照
  - 仿真出错时，从离错误点最近的快照开始恢复执行，并打印波形
- 优点
  - 可扩展性好
  - 效率高

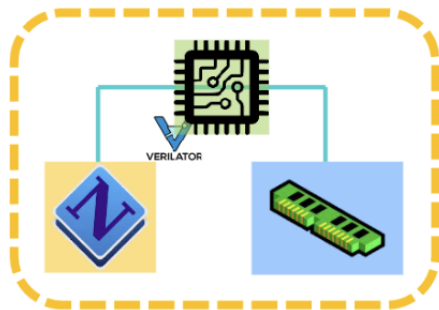


# LightSSS: 一种基于内存的轻量级仿真快照

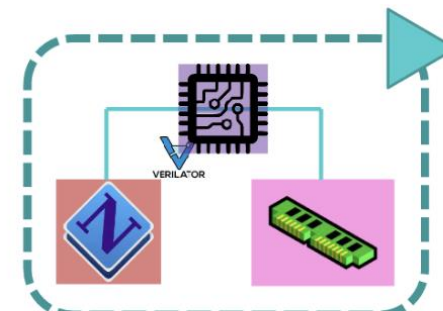
(1)



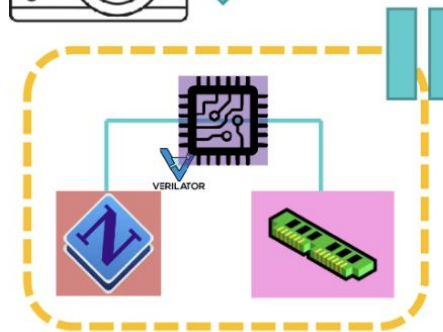
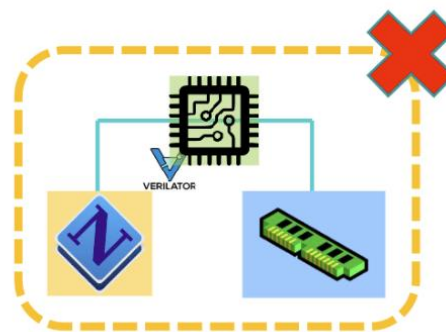
fork = 把进程完全复制一份



(2)

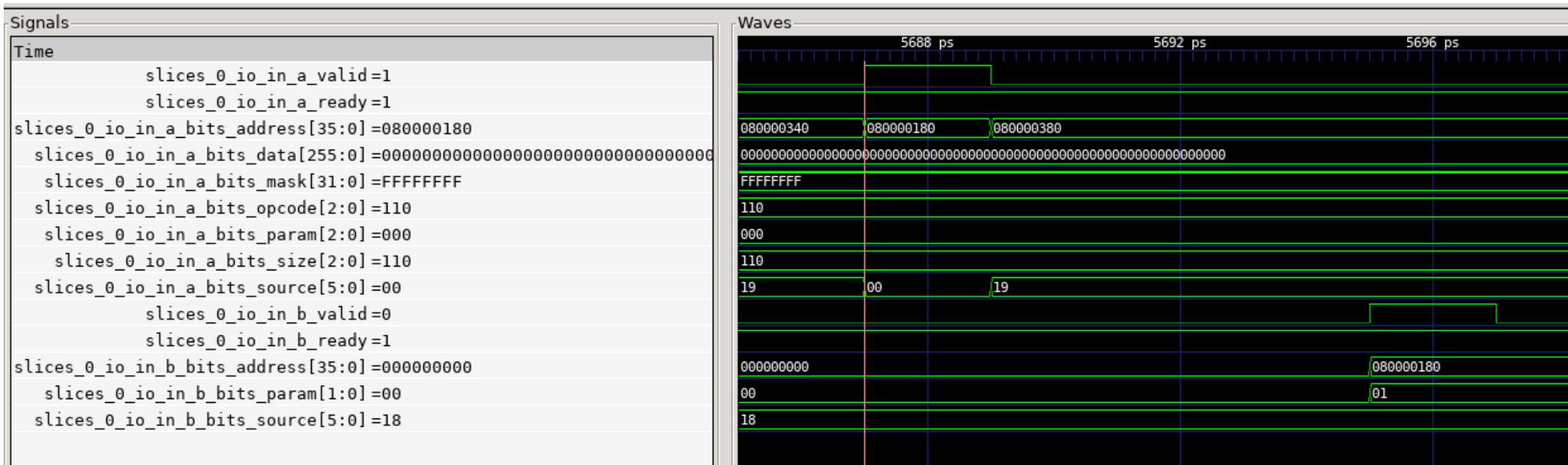


fork



# ChiselDB 简介

- Chisel Database
- 目的
  - 解决波形存储**结构化数据**效率低且调试不便的问题
- 使用方法
  - 在仿真时，利用 DPI-C 调用 C++ 函数，将结构化数据保存在数据库中
  - 仿真结束后，查询数据库获得结构化数据



- 结构化数据仅在部分条件有意义, 不方便分析
- 难以筛选分类

# ChiselDB 查看结构化数据

2022-08-24@16:00:16.db X

XiangShan > build > 2022-08-24@16:00:16.db

Search tables... Reset Filters Records: 853

Tables (3)	NAME	CHANNEL	OPCODE	PARAM	SOURCE	SINK	ADDRESS	DATA_0	STAMP
> TL_LOG	Search column	Search column...	Search column...	Search column...	Search column...	Search column	Search column...	Search column...	Search column...
> sqlite_sequence									
> DIR_LOG									
1	L2_L1I_0	0	6	0	0	0	2147483648	0	285
2	L2_L1I_0	0	5	0	2	0	2147483840	0	291
3	L2_L1I_0	0	5	0	2	0	2147483904	0	293
4	L3_L2_0	0	6	0	0	0	2147483840	0	1032
5	L3_L2_0	0	6	0	48	0	2147483648	0	1033
6	L3_L2_0	0	6	0	49	0	2147483904	0	1034
7	MEM_L3	0	6	0	0	0	2147483840	0	8198
8	MEM_L3	0	6	0	0	0	2147483648	0	8198
9	MEM_L3	0	6	0	1	0	2147483904	0	8200
10	MEM_L3	3	5	0	0	0	2147483840	11124977857725126	8215
11	MEM_L3	3	5	0	0	0	2147483840	69457674904127770	8216
12	MEM_L3	3	5	0	0	0	2147483648	317857644676115	8218

- 数据库里的仿真数据都是有意义的
- 轻松筛选需要的数据



# ChiselDB 数据转义

MEM_L3	0	6	0	0	0	2147483840	0	8198
MEM_L3	0	6	0	0	0	2147483648	0	8198
MEM_L3	0	6	0	1	0	2147483904	0	8200
MEM_L3	3	5	0	0	0	2147483840	11124977857725126	8215
MEM_L3	3	5	0	0	0	2147483840	69457674904127770	8216
MEM_L3	3	5	0	0	0	2147483648	317857644676115	8218

时间戳	名称	通道	命令	源	目的	地址	数据
8198	MEM_L3	A	AcquireBlock	0	0	800000c0	0
8198	MEM_L3	A	AcquireBlock	0	0	80000000	0
8200	MEM_L3	A	AcquireBlock	1	0	80000100	0
8215	MEM_L3	D	GrantData	0	0	800000c0	0027861bc15802c6
8216	MEM_L3	D	GrantData	0	0	800000c0	00f6c3632795c918
8218	MEM_L3	D	GrantData	0	0	80000000	1211700000413

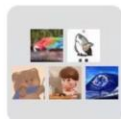
- 数据转义后，增强了可读性



# **Demonstration Time!**

# 互助交流

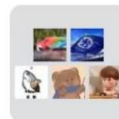
- 香山开发环境文档



2022香山Tutorial交流  
群1



该二维码7天内(8月31日前)有效, 重新进  
入将更新



2022香山Tutorial交流  
群2



该二维码7天内(8月31日前)有效, 重新进  
入将更新

敬请批评指正