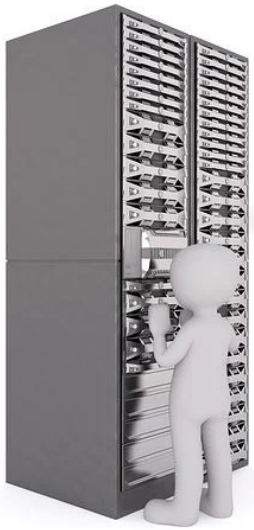
香山性能瓶颈分析

高泽宇 唐浩晋

2022 年 8 月 25 日

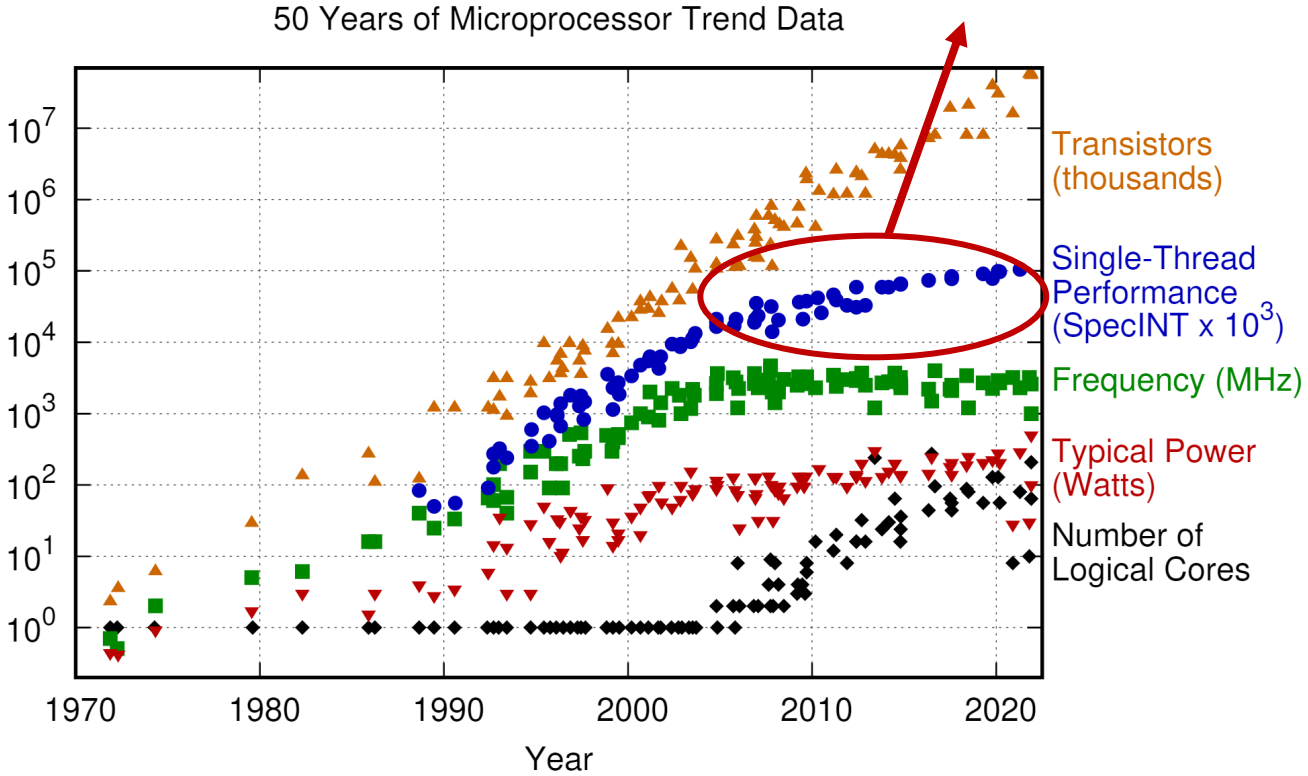
背景：高性能处理器

对高性能需求越来越大



性能提升越来越难

性能增长速率变缓



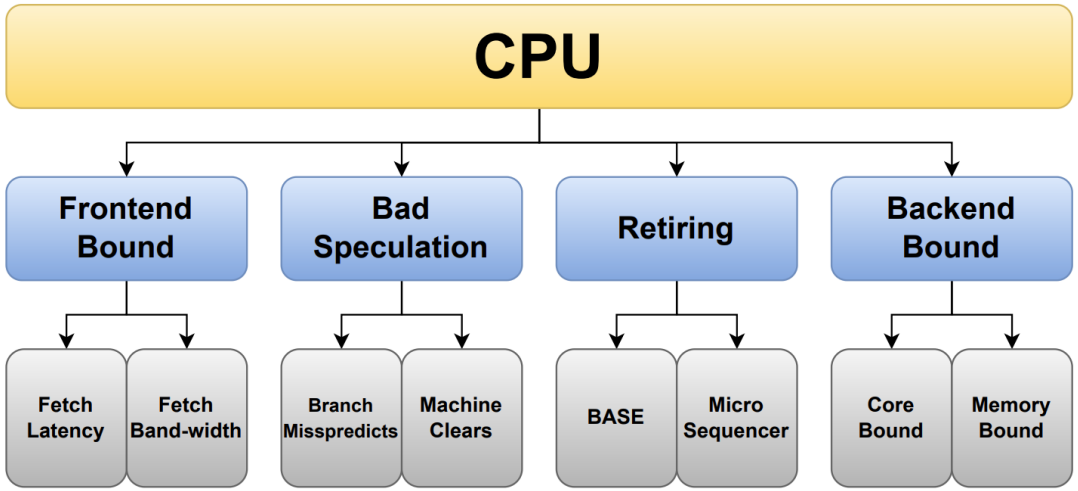
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

- https://pixabay.com/zh/
- https://github.com/karlrupp/microprocessor-trend-data/blob/master/50yrs/50-years-processor-trend.pdf

背景：如何实现性能分析？

- **已有工作**
  - 性能建模：IBM[MICRO1999]
    - 速度快，但是准确性不高
  - 朴素性能分析方法： WISC[VLDB1999]
    - 速度慢，准确性高，但是乱序处理器结果存在重叠
  - 区间性能分析方法：GHENT[SIGPLAN2006]
    - 没有重叠，但是只记录未命中事件
  - ★ Top-Down 性能分析方法：Intel[ISPASS2014]
    - 自顶向下，精确到发射槽粒度，分层细分
    - 不仅记录未命中事件
    - 也记录乱序超标量处理器的其他特点

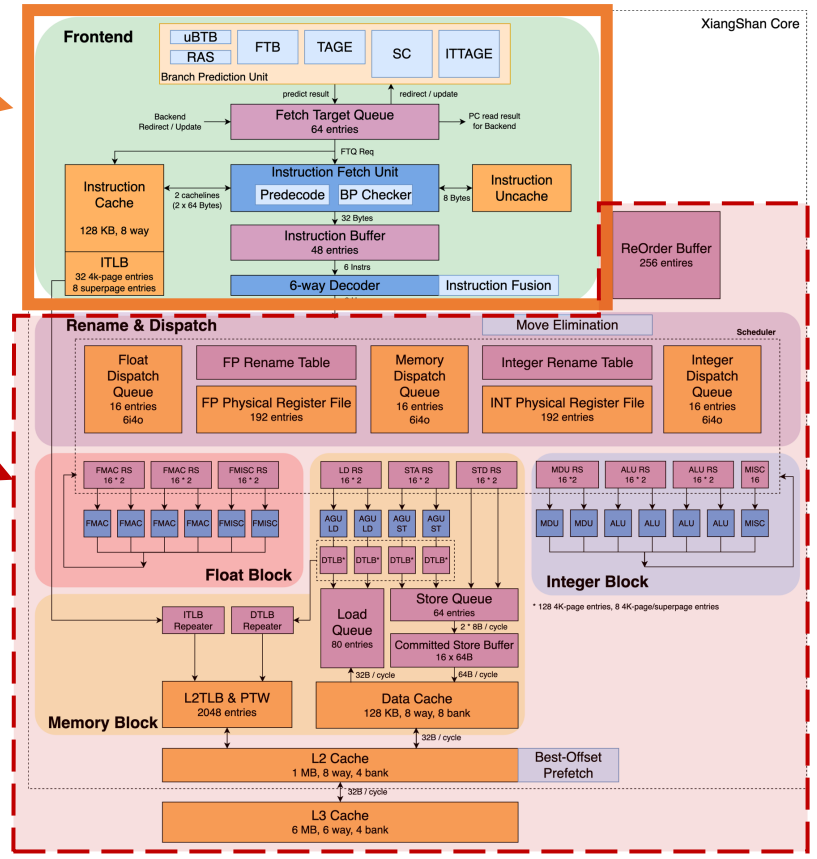
Top-Down方法中层次化的结构设计



针对香山处理器的性能分析模型

- **香山前端：负责处理器的指令供给**
  - 包含分支预测、取指单元、译码单元等
- **香山后端：负责指令的具体执行**
  - 包含指令调度、派遣、重命名、重排序、执行、访存等
- **本工作将 Top-Down 方法应用于香山处理器**
  - 为香山处理器的性能优化、下一版迭代提供精准数据支撑

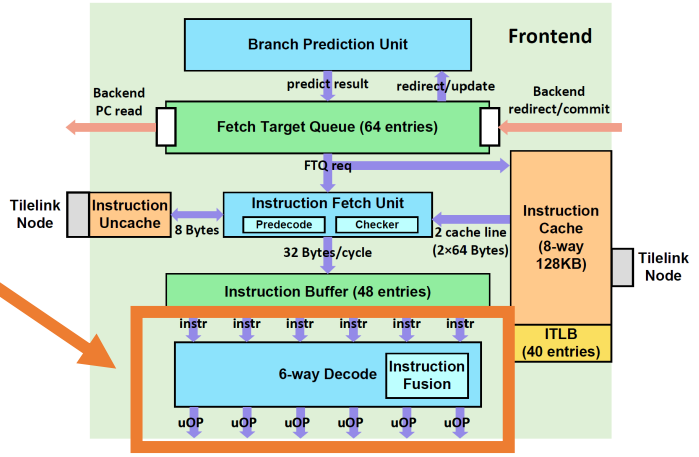
处理器前端



- https://xiangshan-doc.readthedocs.io/

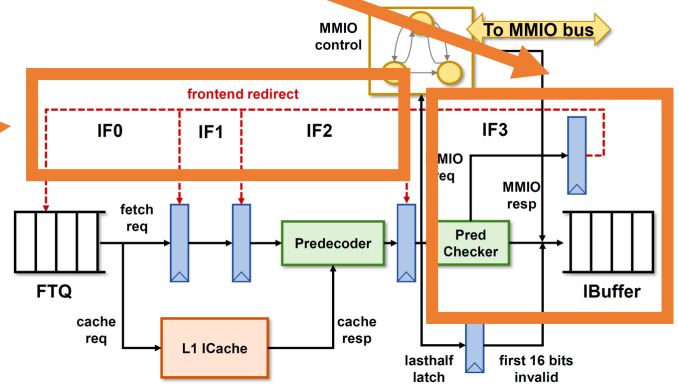
在香山上实现 Top-Down 的难点

- **难点①：指令集差异大，不兼容 RISC-V 架构**
  - 现有 Top-Down 工作基于 x86 架构完成，有不同的特征
  - 例：取指带宽计算方法不同



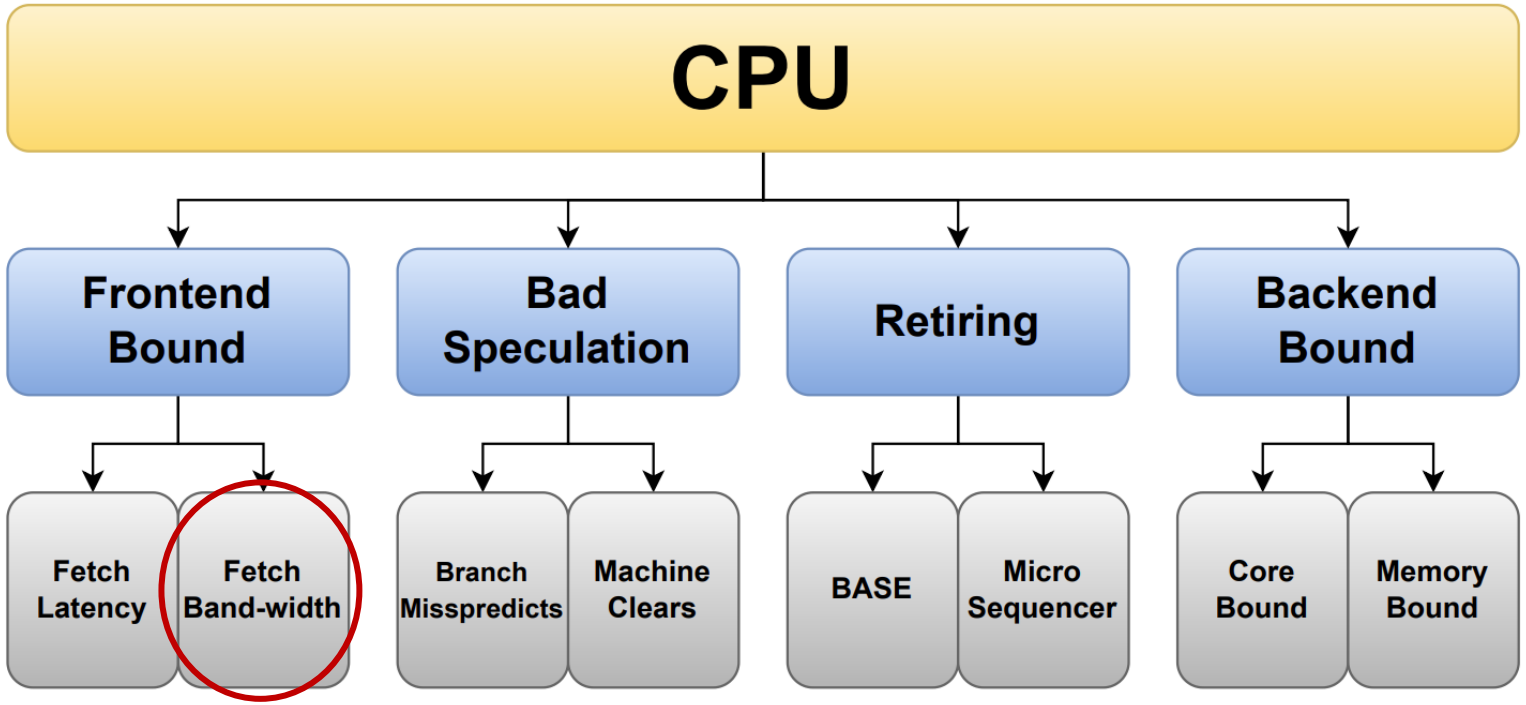
- **难点②：微结构差异大，无法适配香山结构特征**
  - 现有工作支持 Intel 的处理器模型，与香山微结构不同
  - 例：前端的模块划分不同

- **难点③：微结构复杂度高，难以定位性能瓶颈**
  - 现有 Top-Down 模型的分解粒度较粗
  - 例：仅有三个层次，未进一步细化



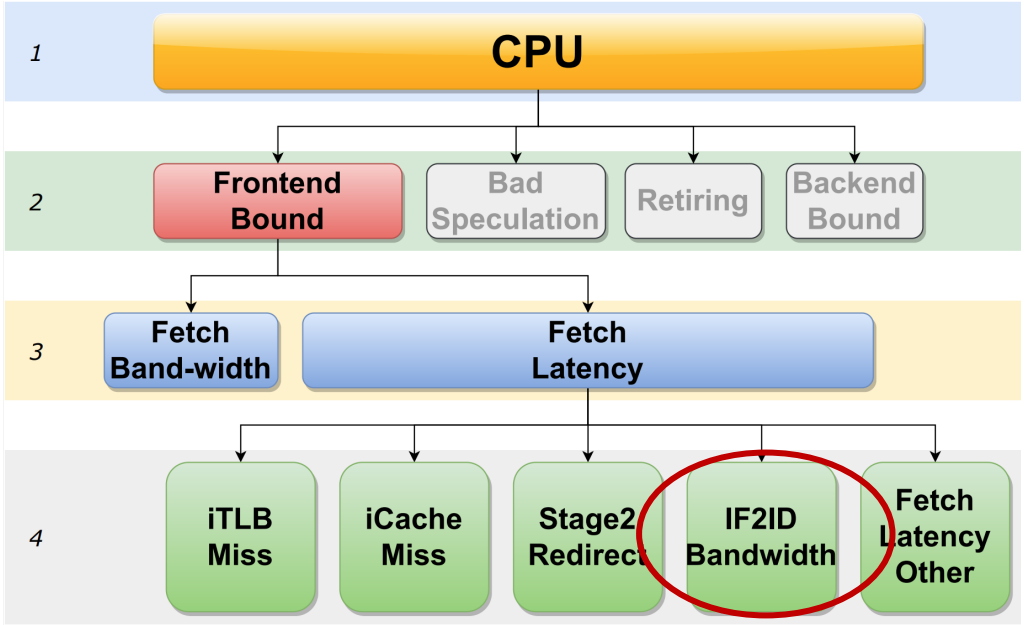
难点①：指令集差异大

- **针对 RISC-V 指令集完成优化适配**
  - 例：取指带宽 = 取指源 1 + 取指源 2
    → 取指带宽 = 指令融合



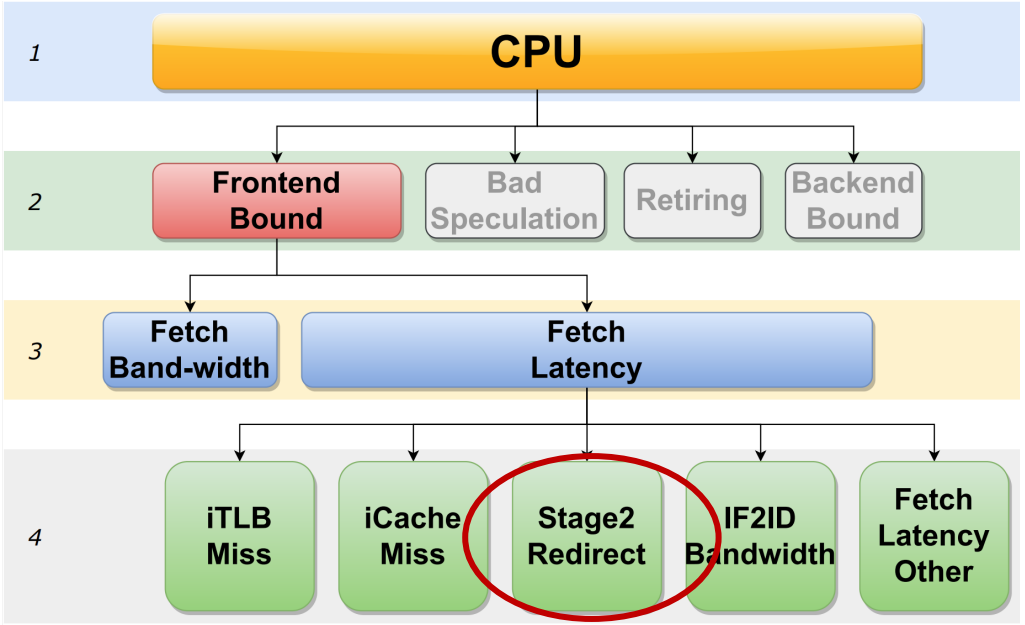
难点②：微结构差异大

- **针对香山处理器微结构调整 Top-Down 设计**
  - 照搬 Top-Down 的模块划分
    - Fetch Latency: iTLB Miss、 iCache Miss、 Redirect、 Other
    - 问题存在于 Other 模块
    - 继续拆分 Other 模块，实现 IF2ID Bandwidth 模块
  - →不适配香山的微结构
  - 完善：
    - 先统计主要模块的端口利用率
    - 根据性能损失比重定位遗漏情况所在模块
    - 利用端口利用率主动暴露遗漏瓶颈

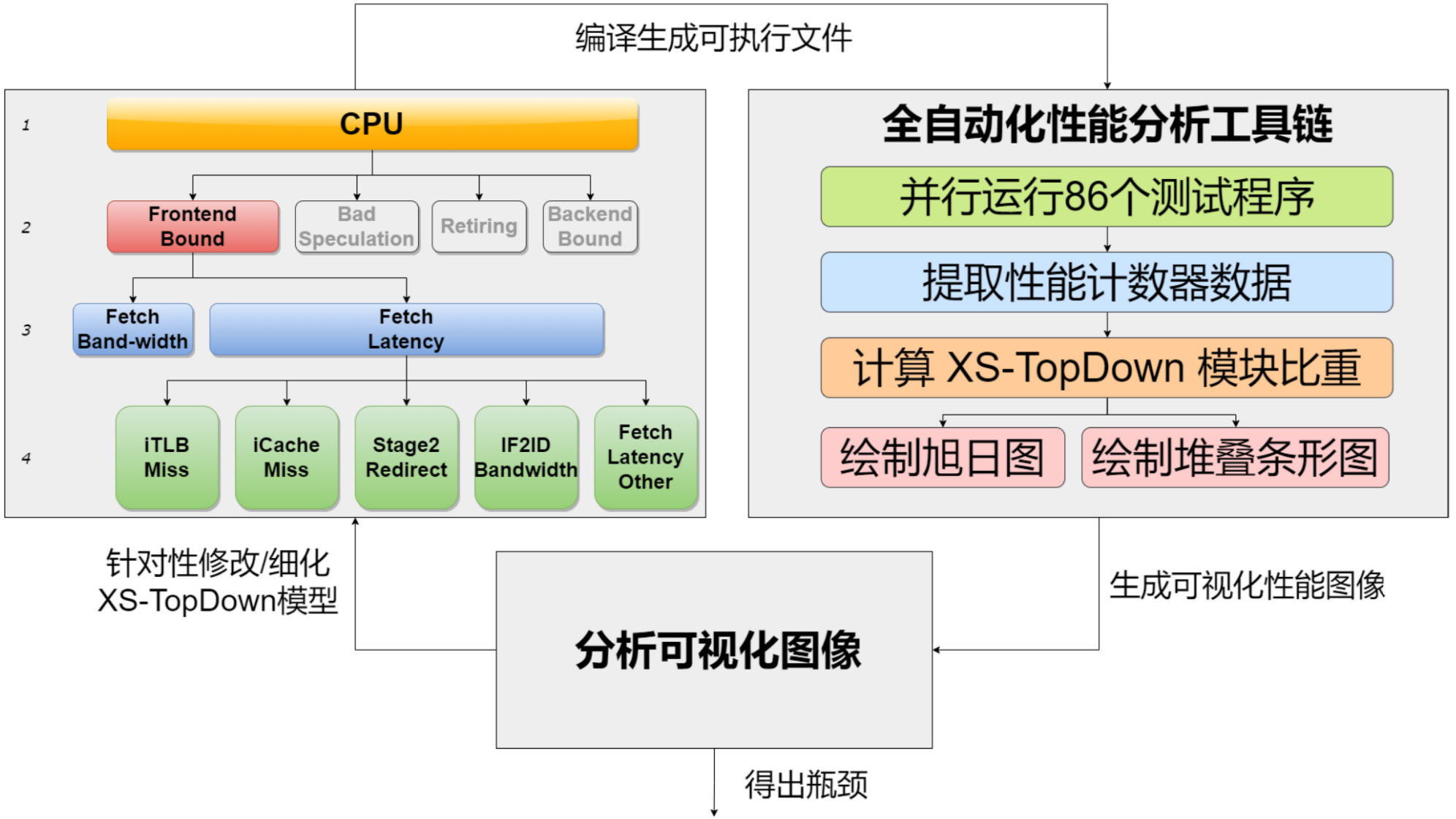


难点③：微结构复杂度高

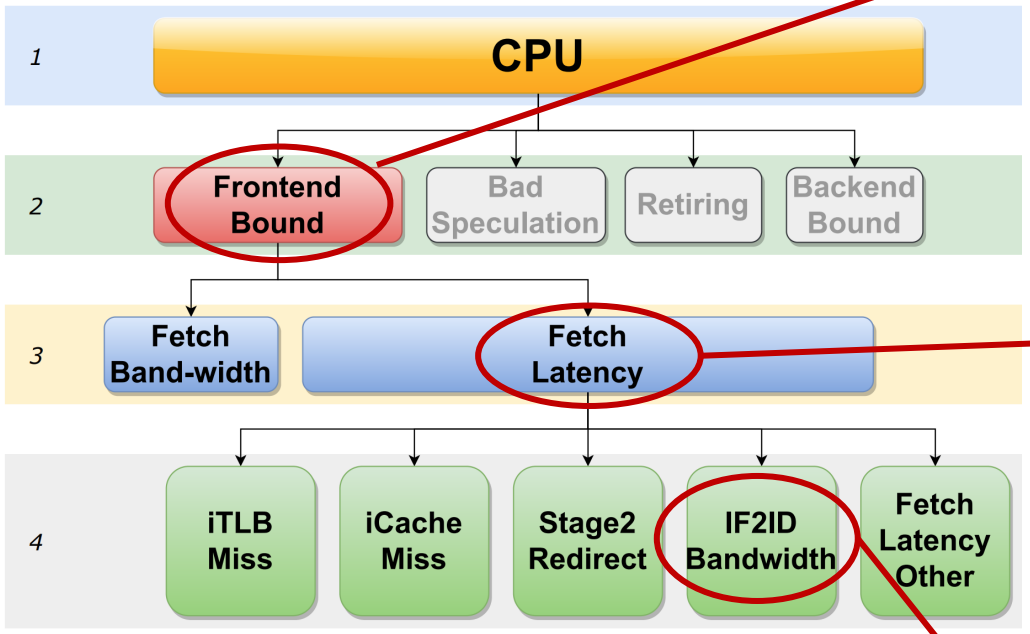
- **细化模型的分层设计**
- 增加一层计数器层次
  - 对第四层进一步分层
  - 例：重定向阻塞
    - 分支预测重定向阻塞
    - ROB 冲刷阻塞
    - Load/Store 重定向阻塞
    - 预取器冲刷阻塞
    - ……
  - 目的：
    - 缩小性能瓶颈范围
    - 验证性能分析模型的准确性

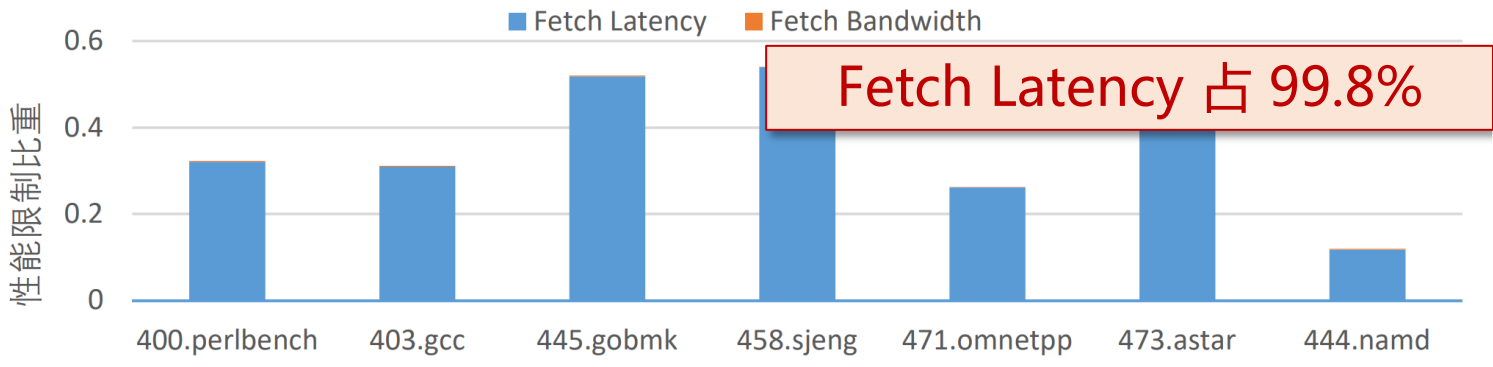


工程实现

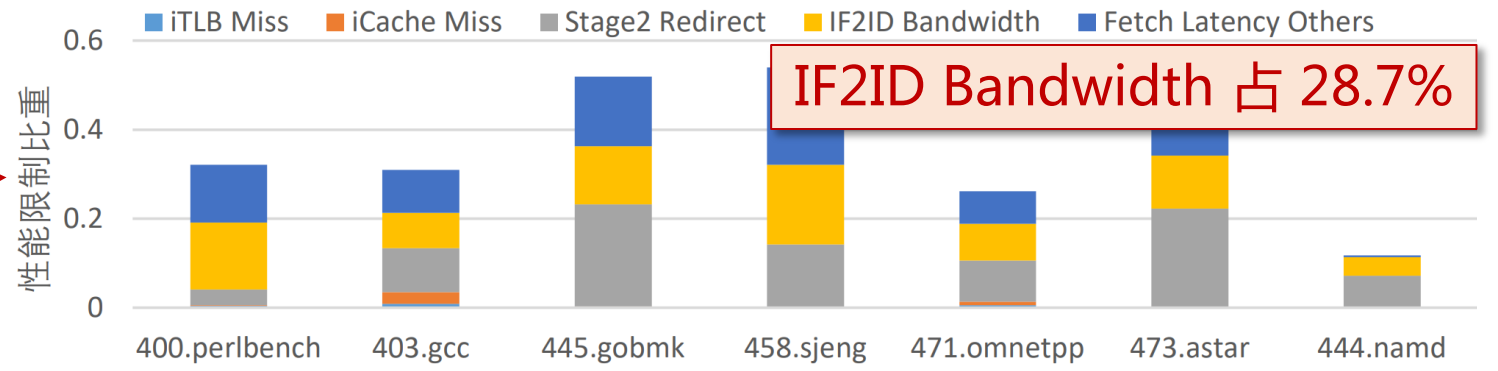

编译生成可执行文件
1
CPU
2
Frontend Bound
Bad Speculation
Retiring
Backend Bound
3
Fetch Band-width
Fetch Latency
4
iTLB Miss
iCache Miss
Stage2 Redirect
IF2ID Bandwidth
Fetch Latency Other
全自动化性能分析工具链
并行运行86个测试程序
提取性能计数器数据
计算 XS-TopDown 模块比重
绘制旭日图
绘制堆叠条形图
针对性修改/细化
XS-TopDown模型
生成可视化性能图像
分析可视化图像
得出瓶颈

前端：Top-Down 分析

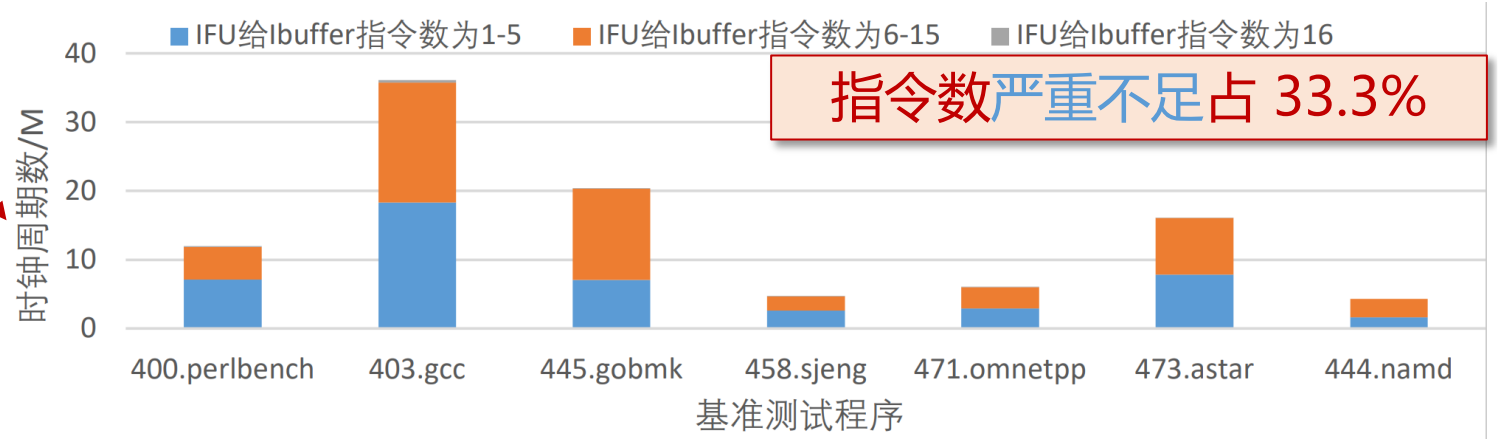




Fetch Latency 占 99.8%

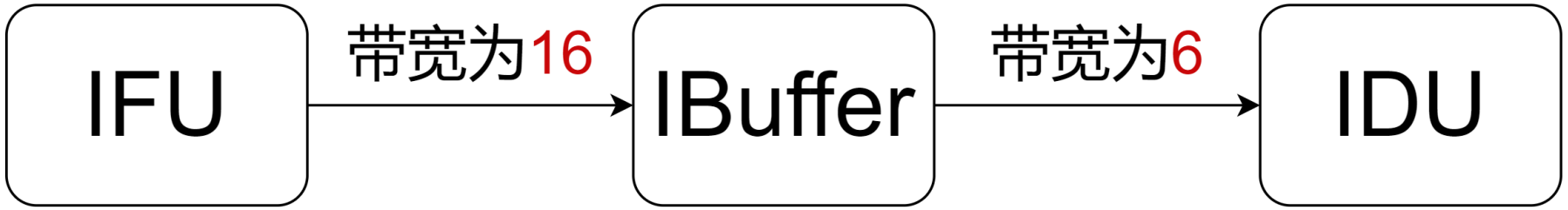


IF2ID Bandwidth 占 28.7%



指令数严重不足占 33.3%

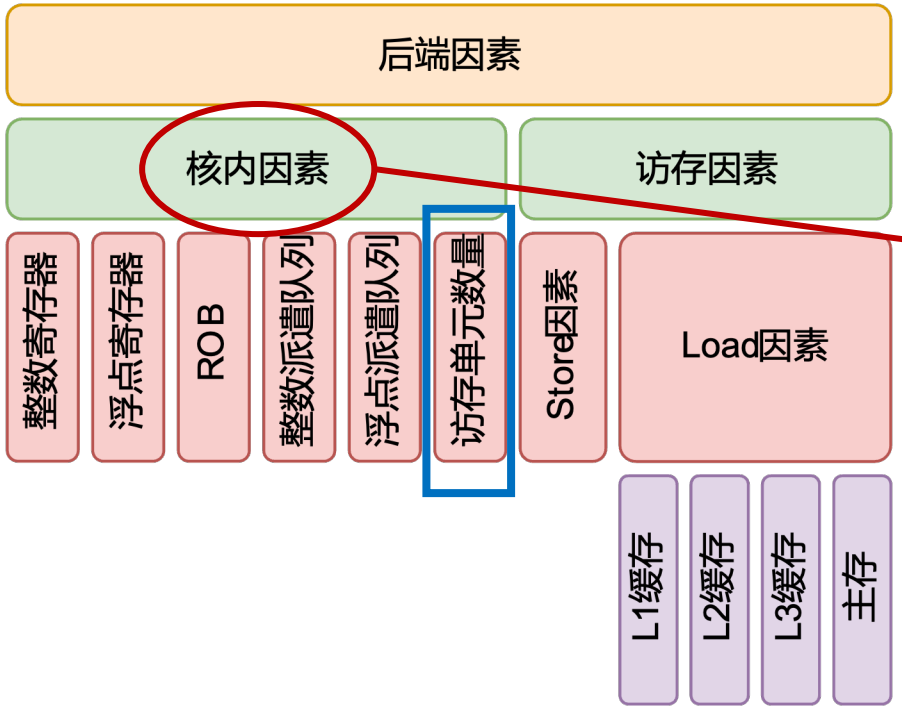
前端：结论

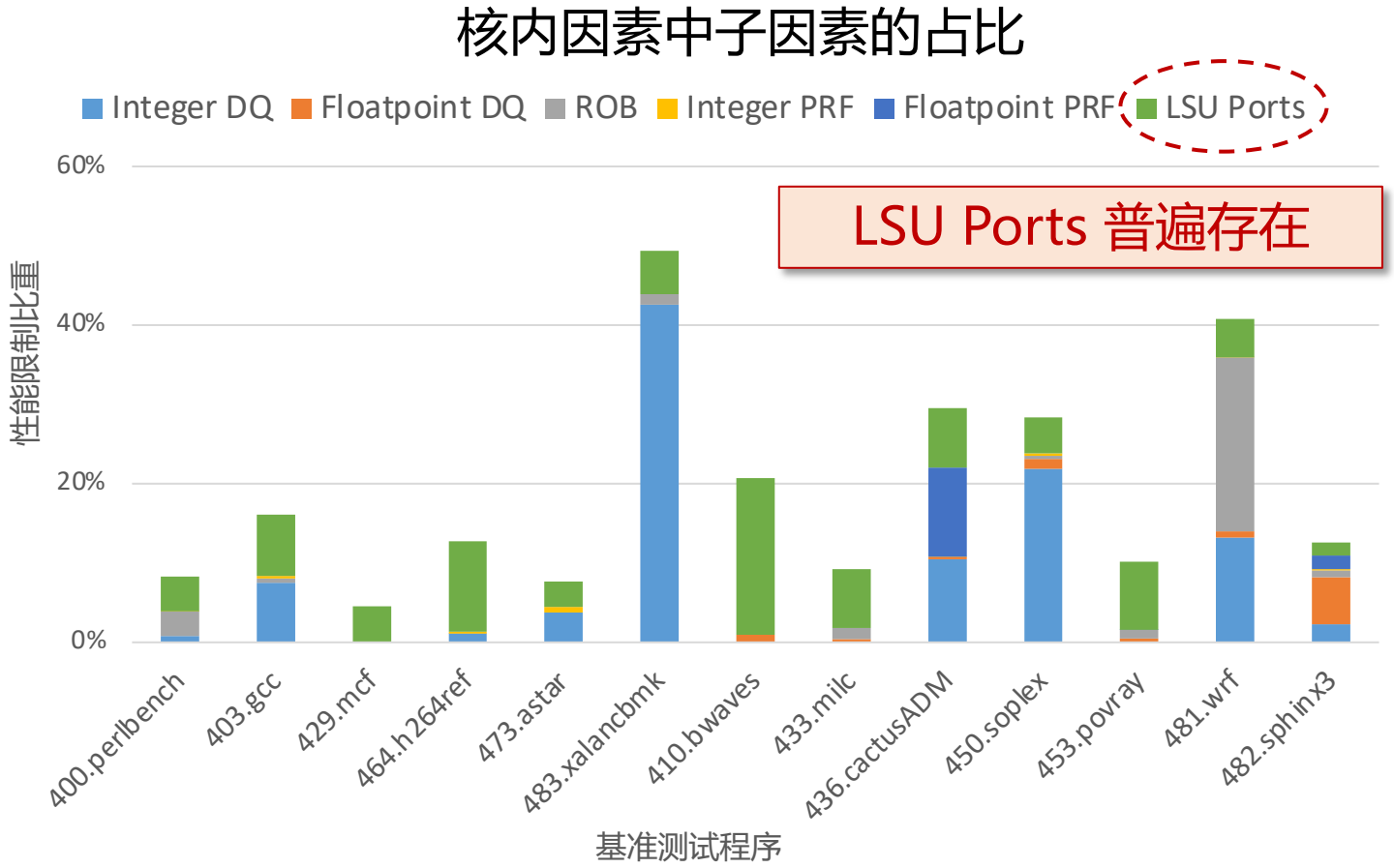


- 分析：
  - 香山处理器更完整的性能计数器
    → 密集跳转分支指令
    → IFU 暂停取指
  - 最终导致 IFU 取指数经常 < 16，甚至 < 6
  - IDU 长时间处于饥饿状态

- 瓶颈所在：
  - 香山处理器前端针对密集跳转分支指令取指能力不足

后端：Top-Down 分析





后端：增加 LSU 数量

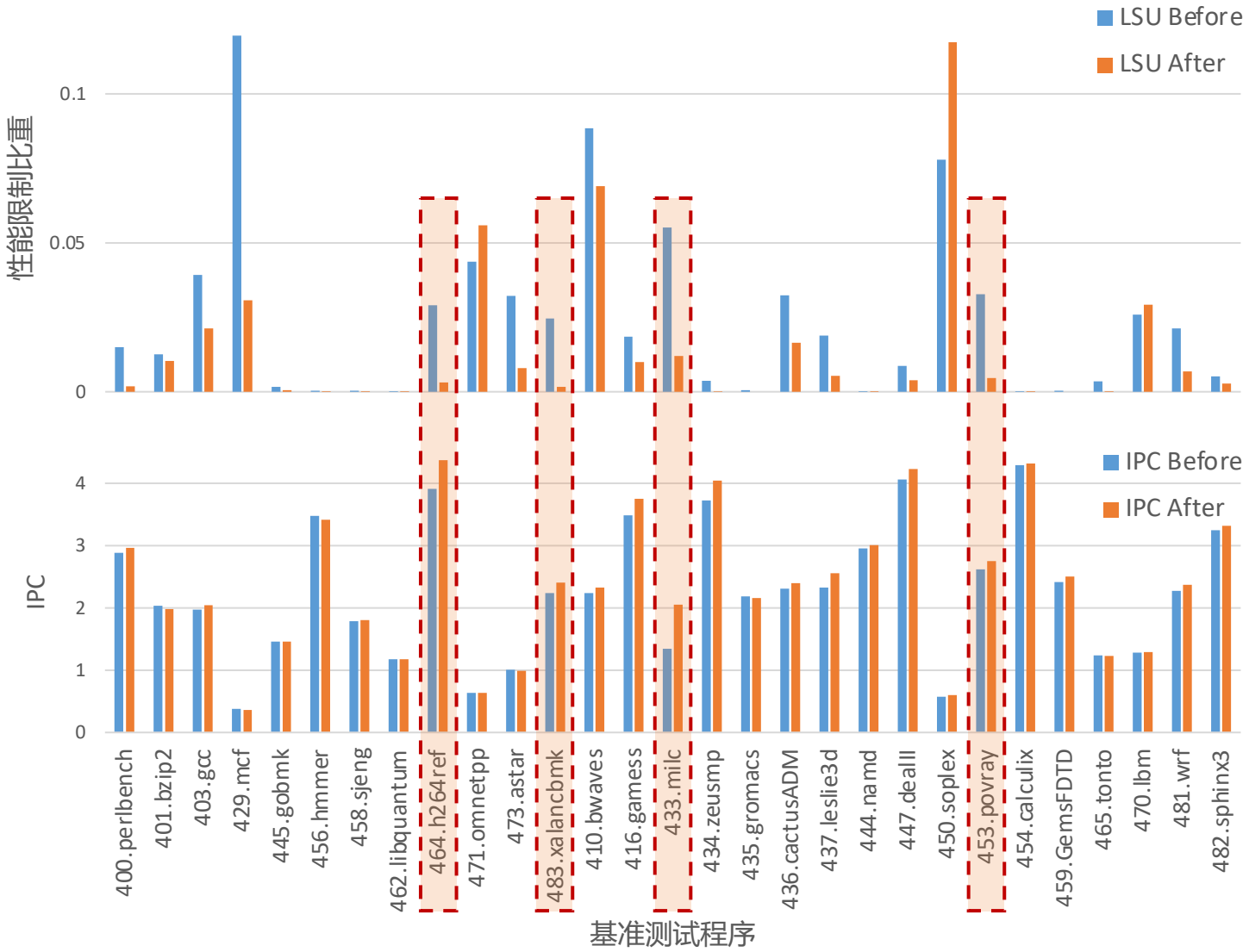
- 根据 Top-Down 分析的结果，增加 LSU 数量
  - 修改处理器 RTL 设计，参数化访存流水线
  - 增多访存流水线数目

与此前相比变化的参数

参数	含义	原配置	本实验采用的配置
LduCnt	读内存执行单元数量	2	4
StuCnt	写内存执行单元数量	2	4
LoadPipelineWidth	读内存流水线宽度	2	4
StorePipelineWidth	写内存流水线宽度	2	4

实例：后端 - 结果对比

增加 LSU 前后的 LSU Ports 部分占比及 IPC 变化



LSU Ports 所占百分比

464.h264ref:	下降了89.27%
483.xalancbmk：	下降了93.59%
433.milc：	下降了78.17%
453.povray：	下降了85.81%

IPC 均有可观的提升

IPC 提升百分比

464.h264ref:	提升了11.66%
483.xalancbmk：	提升了 7.37%
433.milc：	提升了52.20%
453.povray：	提升了 5.09%

谢谢！