中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

中国科学院大学
University of Chinese Academy of Sciences

2026 RISC-V 中国峰会

Wolvrix：面向芯片架构研究的 RTL 编译基础设施

高睿昊，唐浩晋，包云岗
中国科学院计算技术研究所

2026.10.20

内容提要

Why-How-What

- 为什么要新的 IR
- 如何实现新的 IR
- 案例研究进展

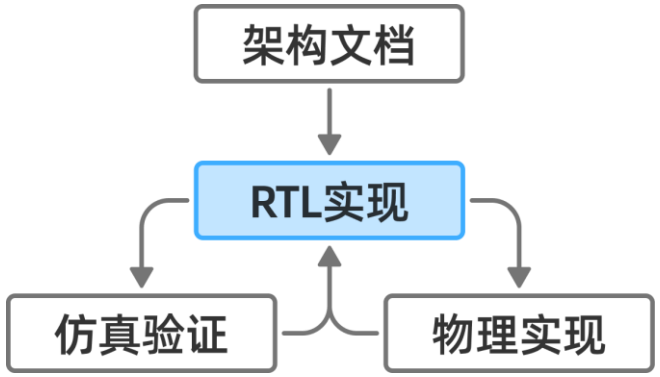
1.为什么需要一套新的 IR 和编译基础设施?

面向需求与问题

- 1.1 RTL 是芯片设计协作的事实接口
- 1.2 图结构是 RTL 分析的自然基础
- 1.3 现有图表示的两类问题

1.1 RTL 是芯片设计协作的事实接口

严格刻画工程细节，连接架构设想与物理实现的桥梁

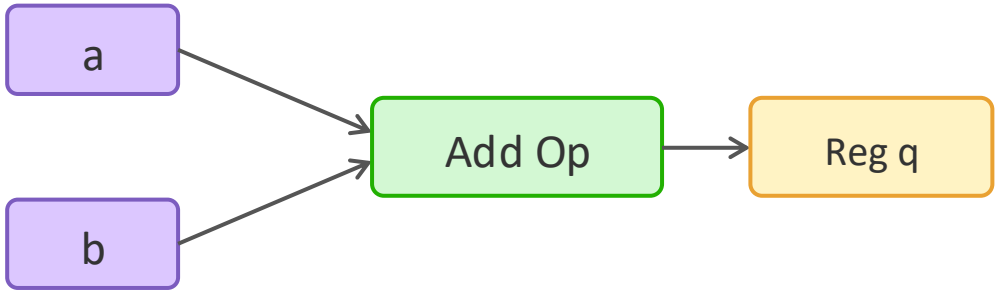


- 传统架构设计通过架构建模的方式，对架构进行迭代
- 以香山开源处理器实践为案例，RTL 是敏捷芯片设计协作的事实接口
- 我们希望在 RTL 层面开展分析变换，优化架构设计
  - 但是，语法特性丰富，处理复杂，分散架构研究的注意力

1.2 图结构是 RTL 分析的自然基础

将电路建模为图，符合分析处理 RTL 的直觉

- 节点：组合逻辑操作、寄存器等
- 边：数据流向关系



- 图结构把“谁依赖谁”“谁驱动谁”“边界在哪里”这些问题显式化
- 现有工具和研究已经反复实践
  - Verilator DFG/CFG，Yosys RTLIL
  - CIRCT IR，支持 Chisel、ESSENT/RepCut、GSIM 等一系列工作

1.3 现有图表示的两类问题

现有图结构为各自任务定制，通用性不足，存在共性短板

- 第一类问题：图结构的混合杂糅
  - 在图结构中嵌入非图成分，典型案例 CIRCT（基于 MLIR）中的 sv 方言

```
%9121 = comb.and bin %9074, %9077, %9081, %9083, %9085, %9089, %9091, %9094, %9006, %9022 : i1
sv.ifdef  @SYNTHESIS {
} else {
  sv.always posedge %8899 {
    %PRINTF_COND_ = sv.macro.ref.expr @PRINTF_COND_() : () -> i1
    %9434 = comb.and bin %PRINTF_COND_, %9023 : i1
    sv.if %9434 {
      sv.fwrite %c-2147483646_i32, "mmio invalid raddr: %x\0A"(%9393) : i32
    }
}
%9122 = comb.and bin %9405, %9416 : i1
%9123 = comb.mux bin %9122, %c2_i3_555, %mmio2Fstate : i3
```

- 第二类问题：表达能力不足
  - 研究场景假设过强：单时钟、不允许多读多写存储器、不支持调试操作（DPI）

Wolvrix 的目标

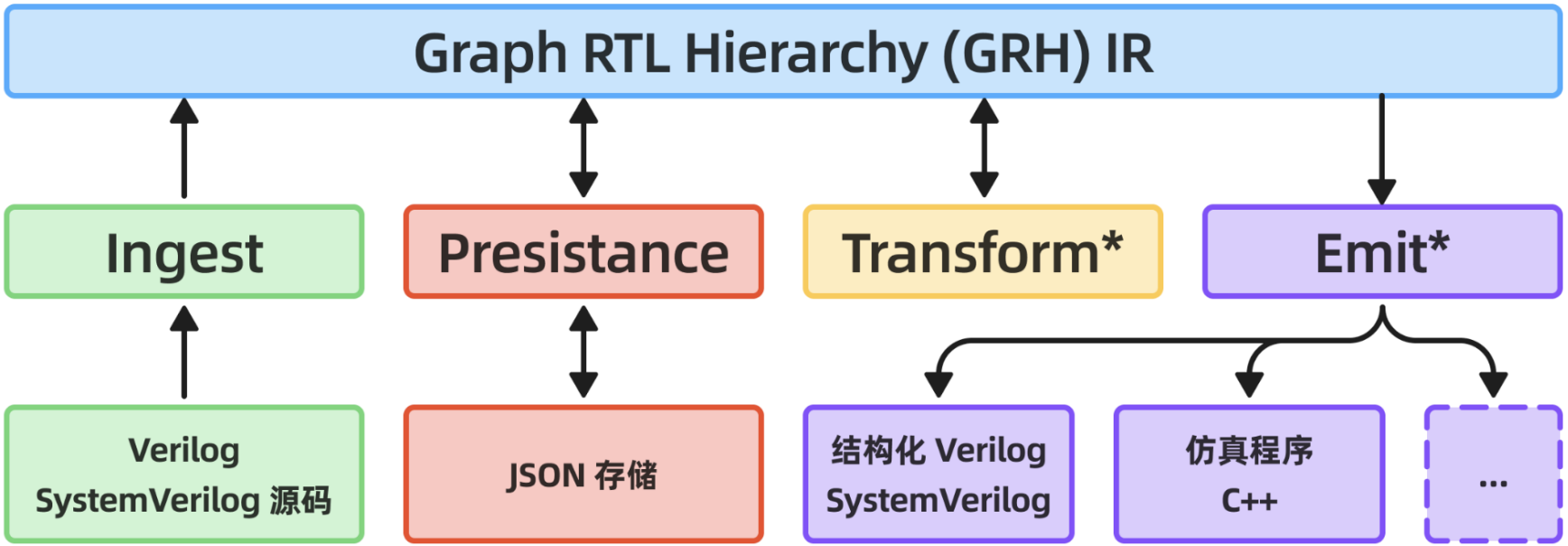
RTL 编译分析处理基础设施，定位类比 MLIR

- 通用 RTL 分析框架：分析、变换，关注电路架构，而非 RTL 语法处理
- 规整层次化图结构：明确定义，甚至标准化，作为统一抽象底座
- 对实际设计完备表达：多时钟域、多端口存储器、DPI、跨层次引用
- 大规模设计的高效编译：图构建、存储与遍历环节进行工程优化

2. Wolvrix 整体架构概览

五个核心组件

- 1：图结构核心库 GRH IR
- +4：建图、持久化、变换、发射输出



3. Graph RTL Hierarchy (GRH) IR

Design、Graph、Operation 和 Value 四类对象

- Design 对应一个电路设计，是 Graph 的集合
- Graph 语义等价于一个 Verilog 模块，定义端口和内部逻辑
- Operation 与 Value 形成 SSA 数据流

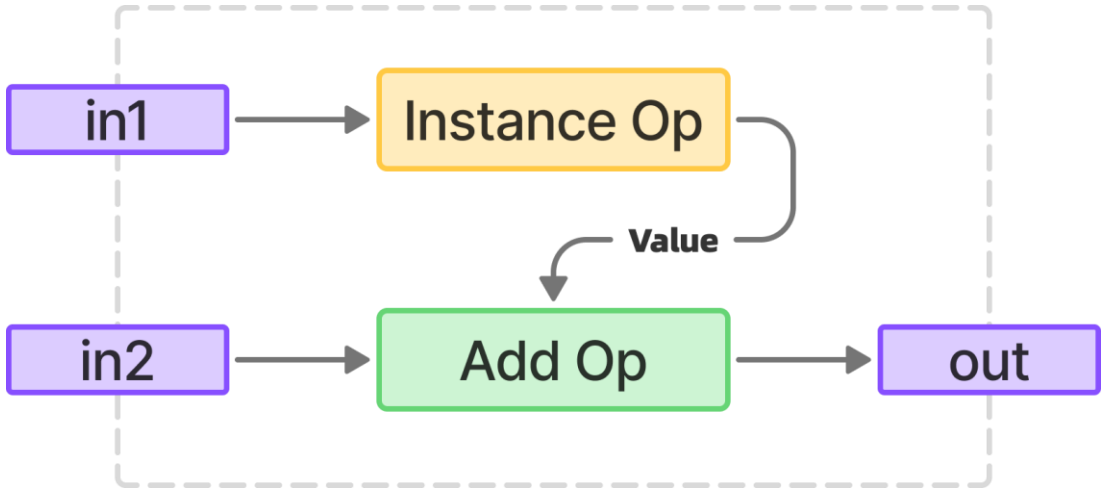


图1.Graph 内部包含 Operation 和 Value

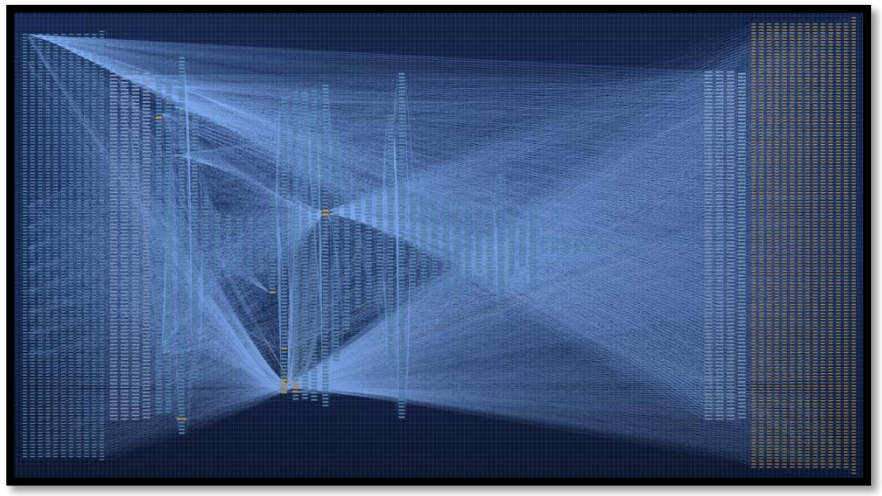


图2. GRH IR 建模的 XiangShan BPU

4. Transform 能力

一组对 GRH IR 进行变换的编译 Pass

- 如果你想要一张平整的图
  - xmr-resolve、hier-flatten、instance-inline
- 如果你想检查图的合法性
  - blackbox-guard、multidriven-guard
- 如果你想化简图结构
  - simplify (const-fold、redundant-elim、dead-code-elim……)
- 如果你想……可以开发自己的 Pass，接入Transform Pass框架

5. Python Binding 界面

在 Python 中编排 C++ 实现的高性能 GRH IR 操作

- 试过 Tcl，但还是 Python 吧，我们想让 AI 也能丝滑使用 Wolvrix
- 先 import wolvrix，然后：

```
with wolvrix.Session() as sess:
    sess.read_sv(path, out_design="design.main",slang_args=read_args,)

    sess.run_pass("xmr-resolve", design="design.main")
    sess.run_pass("simplify", design="design.main")
    sess.run_pass("stats", design="design.main")

    sess.store_json(design="design.main", output=json_out)
    sess.emit_sv(design="design.main", output=sv_out)
```

案例1：结构化 SystemVerilog 生成

验证 Wolvrix 的基本处理能力和正确性

- “读入 - 再生成”流程
  - 规模从小到大：HDLBits 模块、玄铁 C910、香山昆明湖 v3
  - 内部流程：SV 输入 → 建立 GRH → 变化规整 → JSON 输出读入 → Emit 生成 SV
  - 正确性保障：Verilator 仿真对齐测试
- 能够把复杂、工业级的 RTL 转成可分析、可改写、可复用的公共表示

案例2：香山处理器架构特征

验证 Wolvrix 的分析能力

- 针对大规模乱序超标量处理器香山昆明湖 v3 RTL 开展结构分析
  - 规模大： 5,778,793 Operation，5,134,133 Value，总 Value 位宽 87.97M bit
  - 位宽分布：99.171% 的逻辑结果不超过 64 bit，但存在最高 79,263 bit 的长尾
  - 逻辑分布：Mux、And 和 Or 占比超 50%
  - 状态分布：寄存器数量多、Memory 容量大
- 在进一步研究前，把握 RTL 的结构分布和潜在瓶颈/优化来源

案例3：RTL 仿真加速研究

GrhSIM 仿真模型助力学术论文的想法落地

- 仿真对象形式化为三元组 $Sim = \langle S, S_{edge}, G \rangle$

S：电路状态

$S = (S_{in}, S_{out}, S_{state})$

S_{in} - 输入端口值，eval 期间由调用方固定

S_{out} - 输出端口值，G 求值结果的一部分

S_{state} - 状态单元值，寄存器与存储器语义

S_{edge}：边缘跟踪存储

保存被跟踪信号**上一轮求值时的值**，供边缘检测与当前值比较

参与 G 的求值，**不参与 S' == S 不动点比较**

每轮结束同步为本轮起始值，总比被跟踪信号滞后一轮

G：仿真流图

$G = \langle V, E, H \rangle$

V - 操作顶点：相同输入产生相同输出

E - 数据流边：决定执行顺序，允许成环

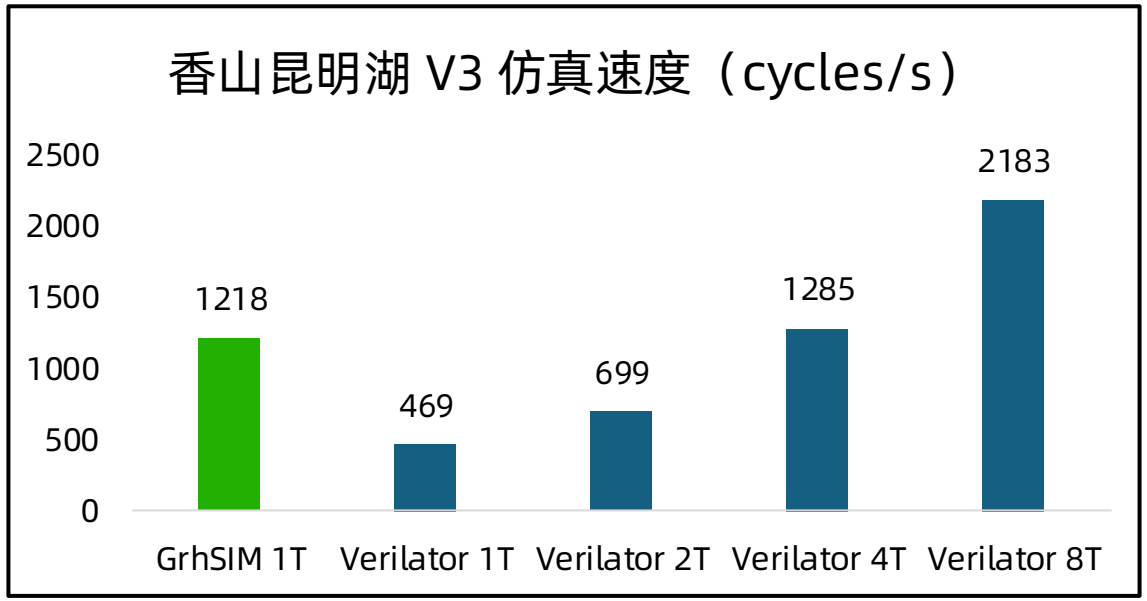
H - host 调用表：DPI、$time 等

eval 求解不动点：每次 eval 反复对整张流图求值，直到电路状态 S 收敛后返回

```
do { (S', S_edge') = G(S, S_edge) } while (S' != S)
```

案例3：RTL 仿真加速研究

- GrhSIM：移植 GSIM 到 GRH IR 的 RTL 仿真器
  - 基于超节点 DAG、Value fanout、状态读写和事件语义生成 C++ 仿真模型
  - DPI 等实际问题统一在 GRH IR 和 Wolvrix 框架下完成
  - 单线程仿真速度显著优于最流行的开源 RTL 仿真器 Verilator

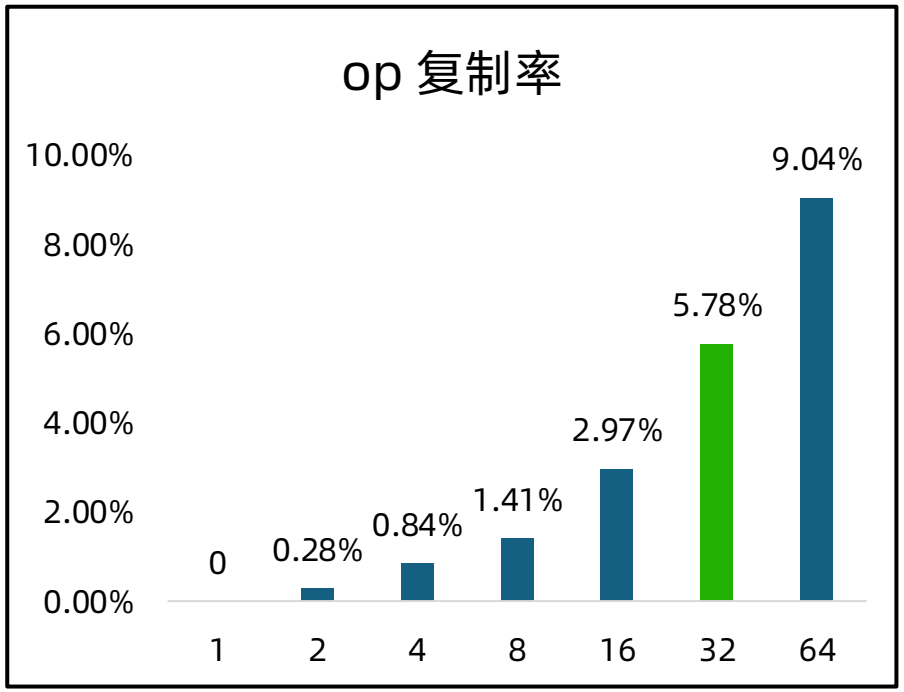


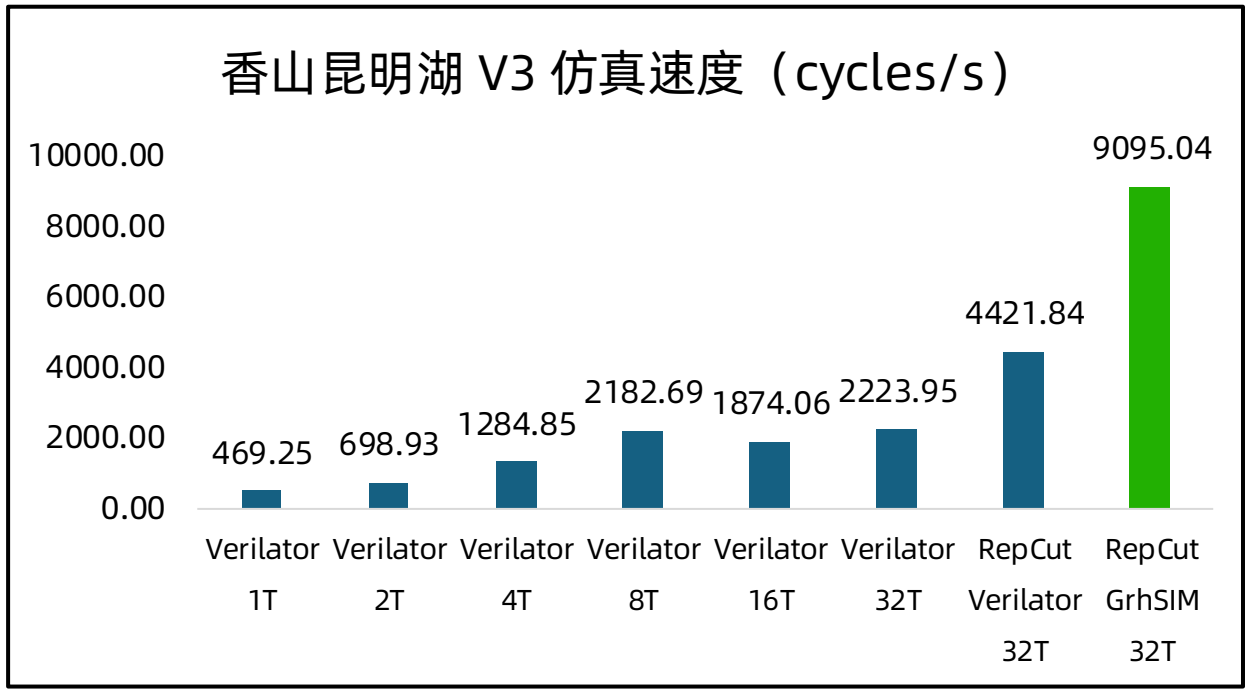
Wolvrix: RTL编译基础设施

案例3：RTL 仿真加速研究

RepCut：面向并行 RTL 仿真的图分区算法

- 通过重复部分仿真计算，以减少仿真线程间的数据同步和交换
- 基于 GRH IR 的状态边界、数据流边和层次信息，生成分区子图与顶层 wrapper
- 32 线程下，RepCut GrhSIM 约为原生 Verilator 仿真速度的 4 倍





未来规划

Wolvrix 持续演进中

- 🔥🔥🔥 RTL仿真加速支持，融合GSIM、RepCut等现有方案
- 🔥🔥与AI/LLM/Agent结合，辅助硬件开发
- 🔥与EDA工具链整合，实现高层次优化

也想听到你的需求和建议！

项目仓库

欢迎 Issue，欢迎 PR!

- wolvrix：完整实现
  - https://github.com/Wolf-Tungsten/wolvrix
- wolvrix-playground：用于开发 Wolvrix 的脚手架
  - https://github.com/Wolf-Tungsten/wolvrix-playground

wolvrix

wolvrix-playground

ECOS团队介绍

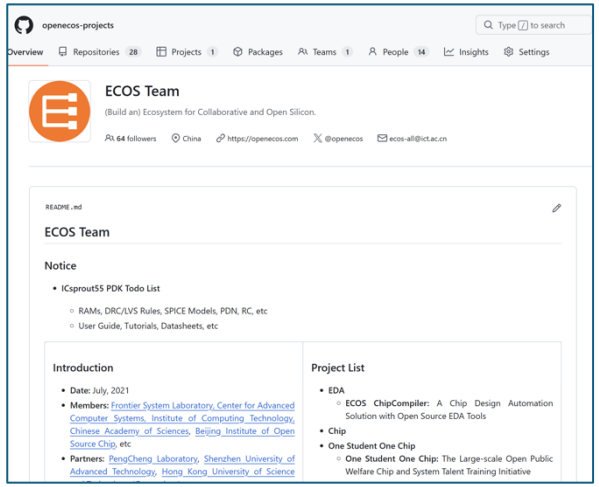
你也许听说过一生一芯

- EDA、Chip、One Student One Chip、System Solution
- 聚焦开源EDA+开源IP+开源PDK的全开源芯片设计解决方案，致力于用“**开源**”**理念革新芯片设计方法，实现“用开源降低芯片设计门槛，赋能千行万业”的终极愿景！**

公众号



官方网站（openecos.com）



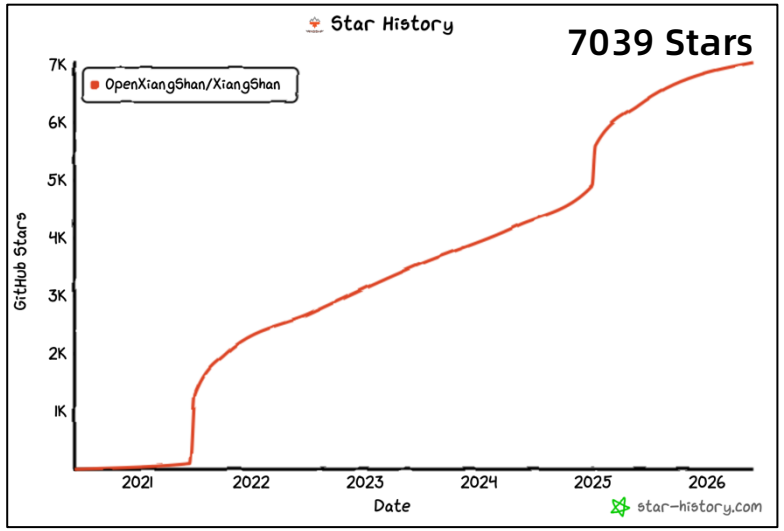
GitHub

Wolvrix: RTL编译基础设施

香山开源处理器团队介绍

携手社区，开发了目前世界上性能最高的工业级开源处理器

- 致力于打造世界顶尖的体系结构创新开源平台，服务于工业界、学术界、个人爱好者等的产品交付和学术研究需求，并形成一套高性能处理器的敏捷开发流程和基础设施。



香山技术讨论微信群　香山技术讨论 QQ 群　香山微信公众号

- 官方网站：xiangshan.cc
- 开源代码：github.com/OpenXiangShan

Wolvrix: RTL编译基础设施

感谢观看

恳请批评指正，多多交流！

- 高睿昊 中国科学院计算技术研究所 特别研究助理
  - gaoruihao@ict.ac.cn
- 唐浩晋 中国科学院计算技术研究所 博士研究生
  - tanghaojin22s@ict.ac.cn

- 开展领域特定硬件架构、RTL 仿真加速技术研究



