

第九届

全国大学生集成电路创新创业大赛

报告类型： 仿真报告

参赛杯赛： 竞业达杯

作品名称： 基于RISC-V指令集的CPU

队伍编号： CICC0903062

团队名称： 逆舟冰箱小队

**目录**

[快速预览简介 3](#_Toc18092)

[1． 仿真平台 4](#_Toc4693)

[1.1仿真工具 4](#_Toc30715)

[1.2测试环境说明 4](#_Toc19284)

[1.3编程语言 6](#_Toc27098)

[2． 仿真用例 6](#_Toc5191)

[2.1仿真测试用例 6](#_Toc21489)

[2.1.1 JYD2025\_Contest-Template 6](#_Toc9550)

[2.1.2 CoreMark 6](#_Toc22678)

[2.2 RV32I指令测试 7](#_Toc32596)

[2.3指令组合测试 7](#_Toc25929)

[2.3.1数据冒险 7](#_Toc4368)

[2.3.2控制冒险 8](#_Toc28723)

[2.3.3结构冒险 8](#_Toc30996)

[3． 仿真结果 9](#_Toc17030)

[3.1 RV32I指令测试 9](#_Toc29985)

[3.1.1 JYD2025\_Contest-Template 9](#_Toc16109)

[3.1.2 CoreMark 9](#_Toc20832)

[3.2指令组合测试 10](#_Toc19568)

[3.2.1数据冒险 10](#_Toc32310)

[3.2.2控制冒险 10](#_Toc4466)

# 快速预览简介

## 本报告详尽阐述了基于RISC-V指令集的CPU在第九届全国大学生集成电路创新创业大赛中的仿真验证工作。项目团队“逆舟冰箱小队”采用Vivado Simulator与Verilator双仿真工具策略，结合Verilog编程语言，对设计的CPU进行了全面测试。测试用例涵盖杯赛企业方提供的JYD2025\_Contest-Template及业界认可的CoreMark基准测试套件。

## 在JYD2025\_Contest-Template测试中，通过捕获LED接口与数码管数据，验证了CPU对RV32I指令集的全面支持，37条指令测试全部通过，程序执行时间为1255毫秒。CoreMark测试中，CPU成功完成1000次迭代，总运行时长为2795258毫秒，充分展示了其核心功能的稳定性与性能。

## 仿真平台方面，利用Vivado Simulator的强大波形查看与调试能力，以及Verilator的高效执行与脚本化特性，构建了高效的验证环境。同时，通过XDC约束文件确保RTL设计与实际硬件实现条件紧密结合，提高仿真结果的准确性。

## 此外，报告还详细描述了RV32I指令测试与指令组合测试（包括数据冒险与控制冒险）的具体实施与结果分析，进一步验证了CPU设计的正确性与高效性。

# 仿真平台

## 1.1仿真工具

## 本设计采用双仿真工具验证策略，结合商业工具和开源工具的优势，构建了完整的验证环境：

## Vivado Simulator：作为Xilinx官方提供的仿真工具，具有与Vivado设计套件无缝集成的优势。我们使用其强大的波形查看功能和调试能力，对设计进行行为级仿真和时序仿真。特别针对企业提供的JYD2025测试用例，Vivado Simulator能够准确模拟FPGA板级外设行为，如LED和数码管显示。

## Verilator：作为开源的Verilog仿真器，具有执行速度快、可脚本化等优点，显著提升了大规模测试（如CoreMark）的仿真效率。Verilator将Verilog代码转换为优化的C++模型，配合自定义的测试环境，实现了高效的周期精确仿真。

**1.2测试环境说明**

**测试环境 1：**

操作系统：**Windows 11**

仿真平台：**Xilinx Vivado 2023.2** 集成开发环境

仿真工具：**Vivado Simulator** (内置版本)

硬件描述：支持**SystemVerilog-2017**标准

平台设置：如图1所示

**测试环境2：**

操作系统：**Ubuntu LTS 22.04.5**

仿真工具：**Verilator 5.035** 开源工具套件

编译环境：**g++ 11.4.0 / GNU Make 4.3**

优化等级：**--O3** 编译优化

验证支持：**SystemVerilog 2009/2012**特性子集

平台设置：如图2所示

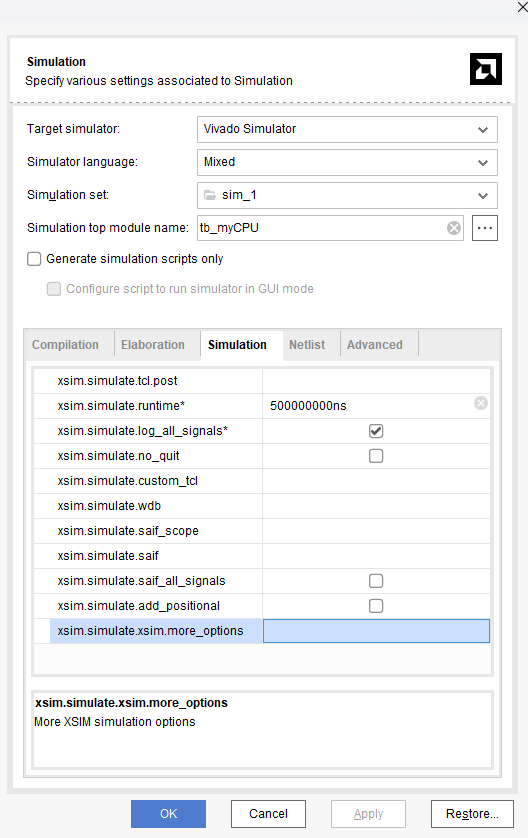


图 1

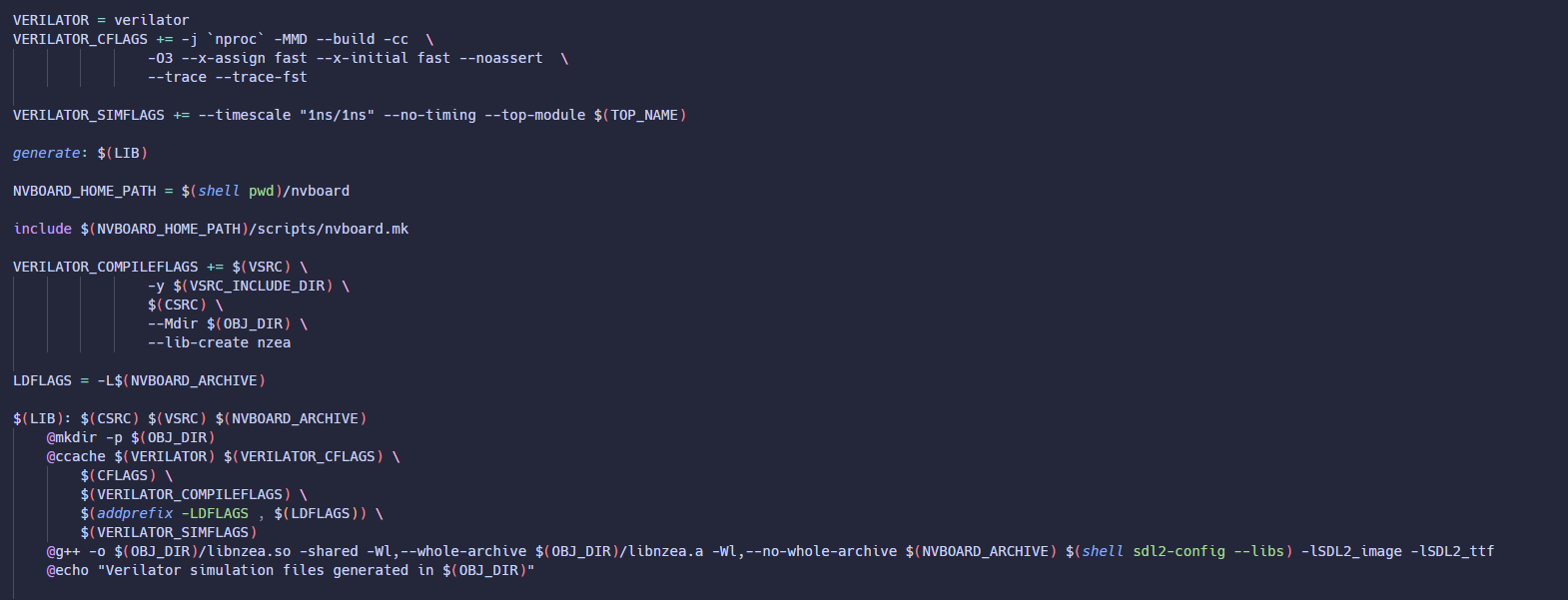


图 2

## 1.3编程语言

**Verilog、VHDL、SystemVerilog：**

在仿真验证中，我们使用Verilog作为核心设计语言，用于实现RTL级电路模型并生成可综合代码，其仿真结果直接反映硬件行为；VHDL主要用于集成第三方**IP**核及构建高精度参考模型，通过混合仿真验证设计功能正确性；SystemVerilog则在Verilog基础上扩展了高级验证功能，支持断言检查（SVA）等功能，显著提升仿真验证效率和完备性。三者协同工作，共同确保仿真验证的准确性和全面性。

**XDC：**

XDC约束文件在数字电路仿真验证中发挥着桥梁作用，我们通过其进行时序约束（如时钟定义、输入输出延迟）、物理约束（如管脚分配、I/O电平标准）和设计规则约束（如最大扇出、跨时钟域声明），将RTL设计与实际硬件实现条件紧密结合。这些约束不仅指导仿真工具进行更接近真实场景的时序分析和信号完整性验证，还能在早期发现潜在的时序违规、信号冲突等问题，确保仿真结果能够准确预测芯片在综合实现后的实际行为表现，为后续的综合、布局布线阶段提供可靠的参考依据，有效降低设计反复迭代的风险。

# 仿真用例

## 2.1仿真测试用例

### 2.1.1 JYD2025\_Contest-Template

这是由杯赛企业方提供的测试用例，为了更好的适配杯赛平台，我们使用Vivado Simulator对测试用例进行仿真。

### 2.1.2 CoreMark

CoreMark 是一个由嵌入式微处理器基准测试联盟（EEMBC）开发的处理器基准测试套件，旨在评估处理器核心功能的性能。它以其简单性和可移植性而闻名，被广泛应用于嵌入式系统和其他计算平台的性能测试。

我们选用CoreMark 作为仿真测试用例之一，主要基于以下原因：

1、专注核心功能: CoreMark 专注于测试处理器核心功能，如整数运算、控制流和内存访问等，而不依赖于特定操作系统或外部设备。这使得它能够更准确地反映处理器核心的性能。

2、简单性和可移植性: CoreMark 的代码结构简单，易于理解和移植到不同的平台。这使得我们能够方便地将其集成到我们的仿真环境中，并进行跨平台的性能比较。

3、广泛的认可度: CoreMark 已被广泛应用于嵌入式系统领域，并得到了业界的广泛认可。这使得 CoreMark 的测试结果具有更高的可信度和可比性。

## 2.2 RV32I指令测试

为了验证 CPU 是否正确实现了 RV32I 指令集，并确保指令执行的准确性，我们进行了 RV32I 指令测试。测试方法为执行包含所有 RV32I 指令的测试程序，并检查结果是否符合预期。测试程序包含 JYD2025\_Contest-Template和 CoreMark基准测试套件，两者均包含 37 条 RV32I 指令的测试用例。

## 2.3指令组合测试

### 2.3.1数据冒险

**识别依赖指令对：**在仿真过程中，识别出两条存在数据依赖的指令（例如，观察 IDU\_pc 为 80000004 的指令 I 和 IDU\_pc 为 80000008 的指令 II）。指令 I 对寄存器（如 sp）进行写入操作，而指令 II 读取并写入同一个寄存器 sp。

**触发冒险条件：**由于指令 II 依赖于指令 I 的结果，但指令 I 的结果尚未写入寄存器堆，构成了 RAW 数据冒险。

监测流水线控制信号：观察当指令 II 进入 IDU 时，PipelineCtrl 模块的行为。检查它是否检测到了该数据冲突。

**验证阻塞操作：**确认 PipelineCtrl 模块是否在检测到冲突时，正确地生成了阻塞信号（stall 被拉高）。

**观察流水线状态：**验证阻塞信号是否成功阻止了 IDU 及其之前的流水线阶段继续推进，同时允许指令 I 继续在流水线中执行（例如，观察到指令 I 进入 ALU 并完成计算）。

**确认结果写入与解除阻塞：**跟踪指令 I 的执行，直到其结果被正确写入寄存器堆（例如，观察到指令 I 进入 WBU 并完成对 sp 的写回）。验证在结果写入完成后，阻塞信号（stall）是否被拉低。

**评估结果：**确认流水线在数据依赖解决后是否能够恢复正常工作，指令 II 是否能在正确的数据基础上继续执行。

### 2.3.2控制冒险

**识别分支指令：**在仿真过程中，识别出特定的分支指令（例如，观察 IDU\_pc 为 80000010 时的 jal ra, 9160 指令）。

**触发冒险条件：**由于流水线特性，分支目标地址需在写回阶段才能确定，而在此之前，流水线会基于默认推测（如顺序执行）继续取指和执行后续指令。

监测分支结果：跟踪该分支指令在流水线中的执行过程。当分支指令到达能够确定目标地址的阶段（例如，进入 WBU 后计算出正确的下一条指令地址 0x800023d4）。

**验证控制逻辑：**观察 PipelineCtrl 模块的行为。检查它是否能够比较当前流水线前端正在执行的地址（例如，ALU\_pc 显示的 0x80000010）与分支指令计算出的正确目标地址（0x800023d4）。

**确认冲刷操作：**如果比较结果显示推测错误（地址不匹配），则验证 PipelineCtrl 模块是否正确地生成了冲刷信号（flush 被拉高）。

**评估结果：**确认冲刷信号是否成功清空了 WBU 之前的所有错误指令，从而保证了处理器能够从正确的地址继续执行，避免了因错误推测导致的执行错误。

### 2.3.3结构冒险

结构冒险，即取指与访存共用内存区域，在流水线中存在竞争导致的冒险，本来应该使用总线仲裁机制实现，不过目前官方要求使用的模板将IFU总线与LSU总线分离，因此不存在结构冒险，所以无需设计指令组合测试该冒险。

# 仿真结果

## 3.1 RV32I指令测试

通过以下测试中的关键波形抓取，以及测试结果可知37条指令均已测试通过。

### 3.1.1 JYD2025\_Contest-Template

通过抓取student\_top中的LED接口数据、以及perip\_bridge中的传给display\_seg的数码管数据，可以得到仿真测试用例通过。

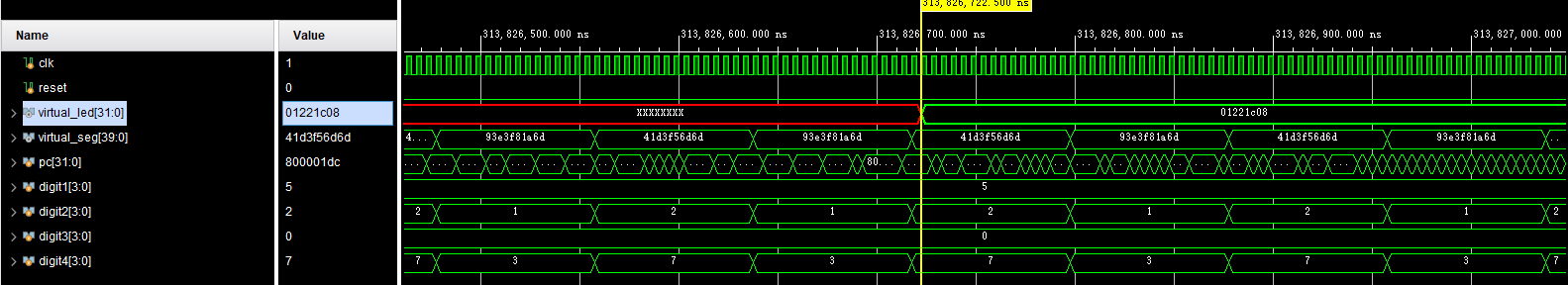


图3使用Vivado Simulator仿真下的官方测试用例仿真结果

这里LED信号映射到板子上对应的是一个正确的✔的形状，而且digit4、3、2、1分别对应八个数码管，可以映射到数码管上即为37\_00\_12\_55的显示，对应官方文档中的测试格式即为RV32I指令集测试的通过条数（37/37），测试程序执⾏的时间，单位为ms（用时1255ms，1255s）。

### 3.1.2 CoreMark

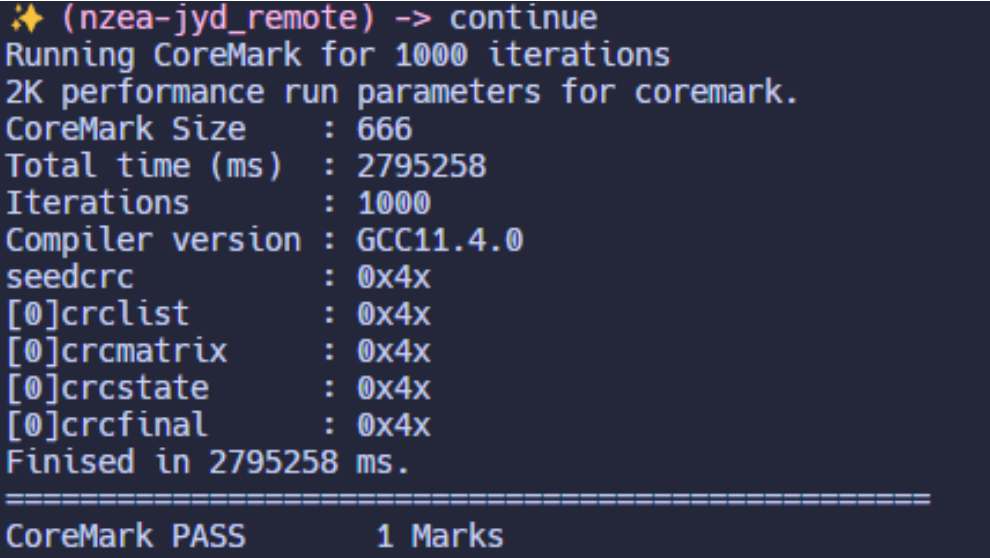


图4-使用Verilator仿真下CoreMark仿真结果

对Coremark进行1000次迭代，运行总时长为2795258ms，并且成功通过CoreMark测试。

## 3.2指令组合测试

通过抓取信号获得以下波形可以看出信号均符合预期。

### 3.2.1数据冒险

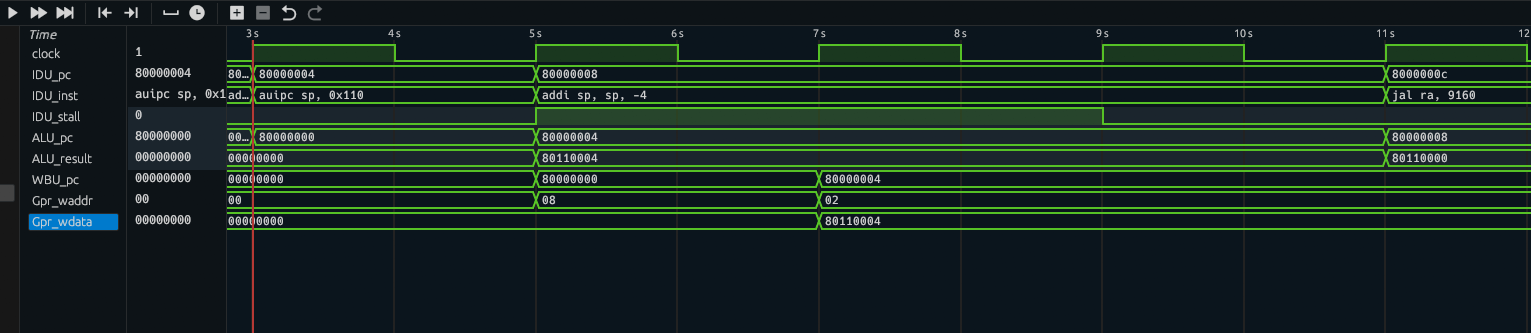


图5

### 3.2.2控制冒险



图6