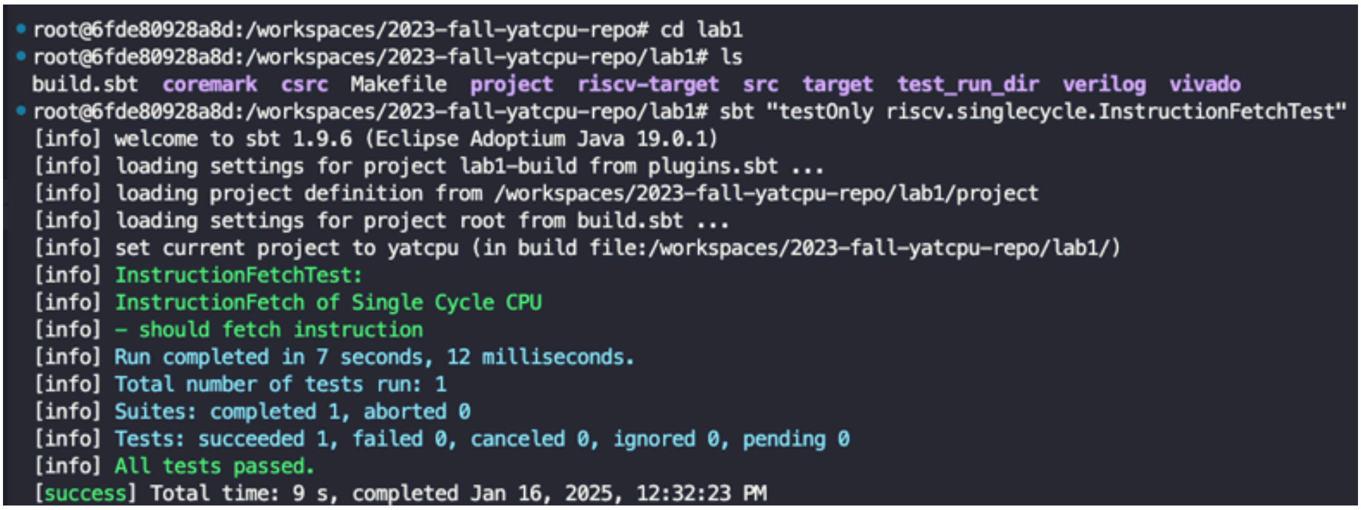
实验一： 单周期 CPU 实验报告

23336004 陈政宇

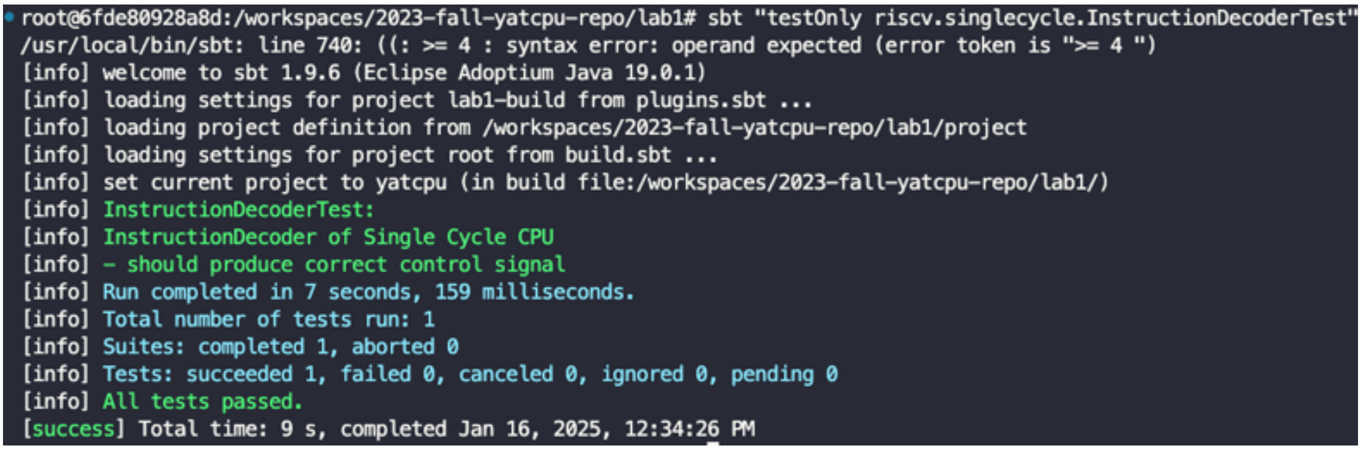
1.测试用例概述

1. **InstructionFetchTest**
   - **测试层面**:
     验证基本指令执行功能模拟了 CPU 执行连续的 100 条随机指令时 `InstructionFetch` 模块能否正确顺序执行指令或跳转
   - **加载方法**:
     在仿真环境中以汇编或机器码方式加载
   - **执行结果**:
     通过



```
• root@6fde80928a8d:/workspaces/2023-fall-yatcpu-repo# cd lab1
• root@6fde80928a8d:/workspaces/2023-fall-yatcpu-repo/lab1# ls
  build.sbt  coremark  csrc  Makefile  project  riscv-target  src  target  test_run_dir  verilog  vivado
• root@6fde80928a8d:/workspaces/2023-fall-yatcpu-repo/lab1# sbt "testOnly riscv.singlecycle.InstructionFetchTest"
  [info] welcome to sbt 1.9.6 (Eclipse Adoptium Java 19.0.1)
  [info] loading settings for project lab1-build from plugins.sbt ...
  [info] loading project definition from /workspaces/2023-fall-yatcpu-repo/lab1/project
  [info] loading settings for project root from build.sbt ...
  [info] set current project to yatcpu (in build file:/workspaces/2023-fall-yatcpu-repo/lab1/)
  [info] InstructionFetchTest:
  [info] InstructionFetch of Single Cycle CPU
  [info] - should fetch instruction
  [info] Run completed in 7 seconds, 12 milliseconds.
  [info] Total number of tests run: 1
  [info] Suites: completed 1, aborted 0
  [info] Tests: succeeded 1, failed 0, canceled 0, ignored 0, pending 0
  [info] All tests passed.
  [success] Total time: 9 s, completed Jan 16, 2025, 12:32:23 PM
```

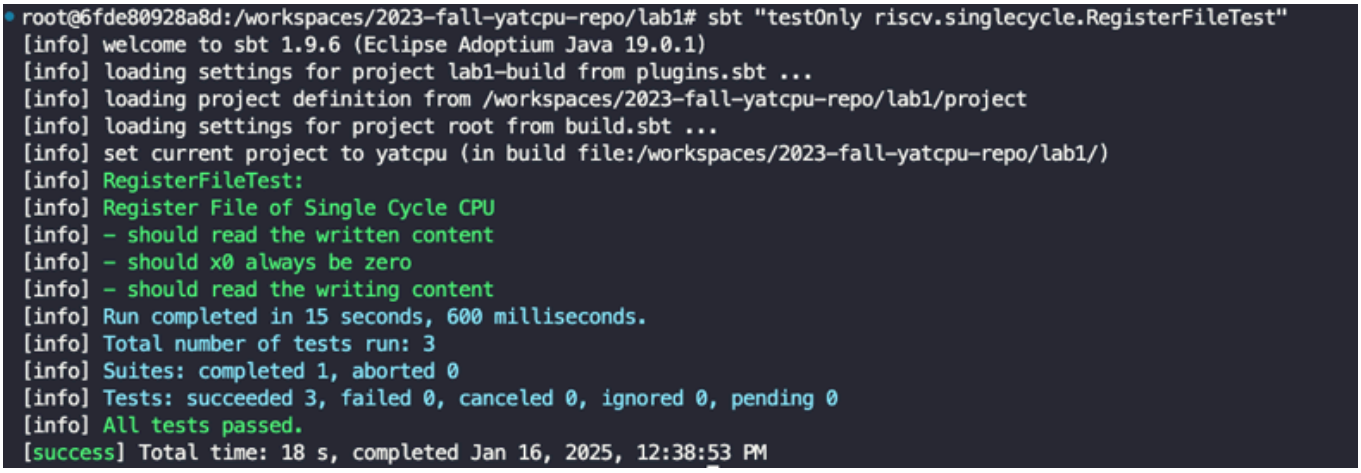
2. **InstructionDecoderTest**
   - **测试层面**:
     测试了解码过程能否正确识别指令。
   - **加载方法**:
     同上。
   - **执行结果**
     通过



```
root@6fde80928a8d:/workspaces/2023-fall-yatcpu-repo/lab1# sbt "testOnly riscv.singlecycle.InstructionDecoderTest"
/usr/local/bin/sbt: line 740: ((: >= 4 : syntax error: operand expected (error token is ">= 4 ")
[info] welcome to sbt 1.9.6 (Eclipse Adoptium Java 19.0.1)
[info] loading settings for project lab1-build from plugins.sbt ...
[info] loading project definition from /workspaces/2023-fall-yatcpu-repo/lab1/project
[info] loading settings for project root from build.sbt ...
[info] set current project to yatcpu (in build file:/workspaces/2023-fall-yatcpu-repo/lab1/)
[info] InstructionDecoderTest:
[info] InstructionDecoder of Single Cycle CPU
[info] - should produce correct control signal
[info] Run completed in 7 seconds, 159 milliseconds.
[info] Total number of tests run: 1
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 1, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[success] Total time: 9 s, completed Jan 16, 2025, 12:34:26 PM
```

3. **RegisterFileTest**
    - **测试层面**

      测试了写寄存器功能的正确性
    - **加载方法** 与 **执行结果** 同上。



```
root@6fde80928a8d:/workspaces/2023-fall-yatcpu-repo/lab1# sbt "testOnly riscv.singlecycle.RegisterFileTest"
[info] welcome to sbt 1.9.6 (Eclipse Adoptium Java 19.0.1)
[info] loading settings for project lab1-build from plugins.sbt ...
[info] loading project definition from /workspaces/2023-fall-yatcpu-repo/lab1/project
[info] loading settings for project root from build.sbt ...
[info] set current project to yatcpu (in build file:/workspaces/2023-fall-yatcpu-repo/lab1/)
[info] RegisterFileTest:
[info] Register File of Single Cycle CPU
[info] - should read the written content
[info] - should x0 always be zero
[info] - should read the writing content
[info] Run completed in 15 seconds, 600 milliseconds.
[info] Total number of tests run: 3
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 3, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[success] Total time: 18 s, completed Jan 16, 2025, 12:38:53 PM
```

4. **FibnacciTest**, **QuickSortTest**等
    - **测试层面**:

      完整执行一个程序的层面测试 CPU， 除了中断处理、与外设通信以外的所有 CPU 功能
    - **加载方法** 与 **执行结果** 同上。

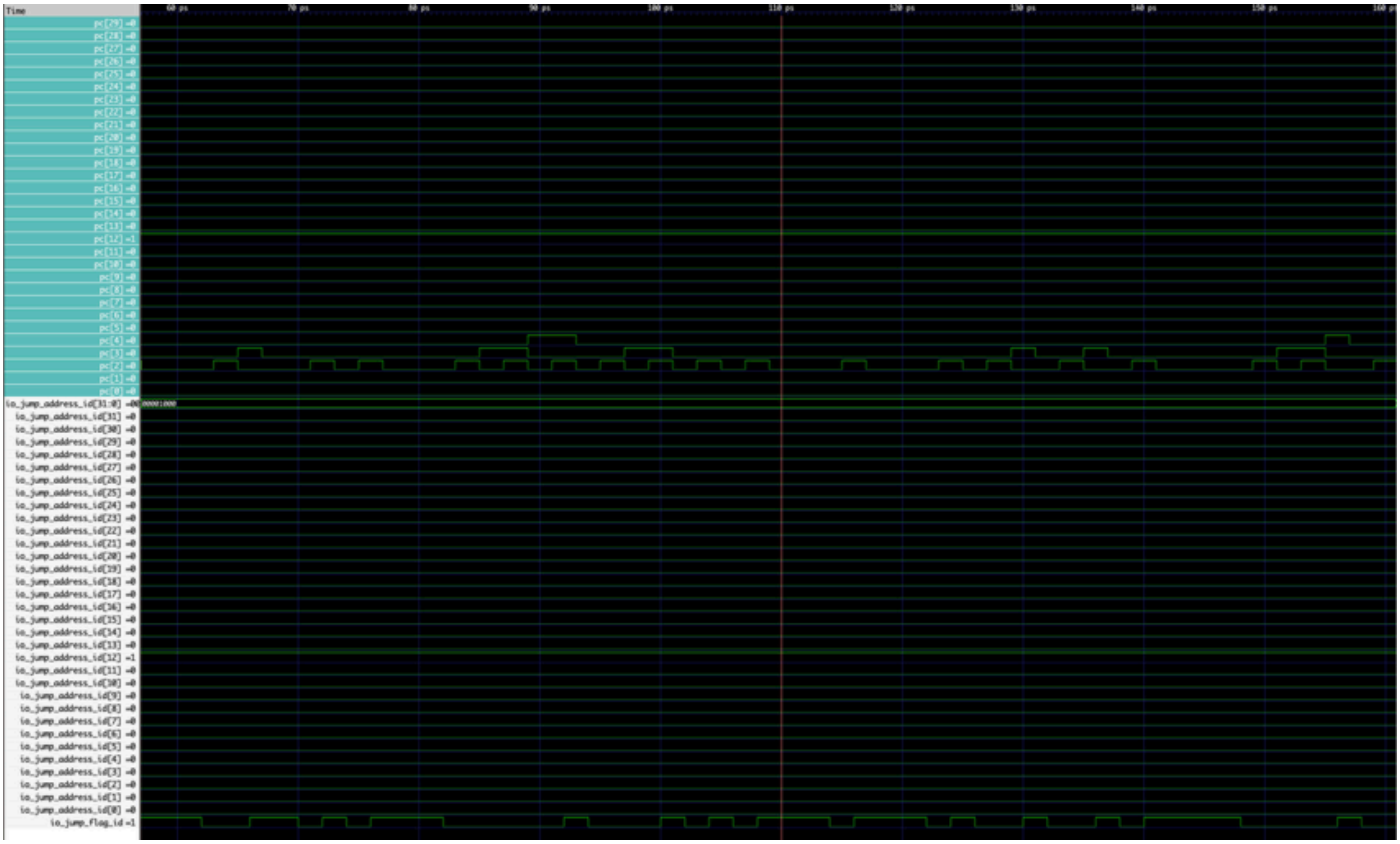
2.信号波形说明

- **加载指令**: 地址总线、控制信号变化
- **执行指令**: ALU输入输出、寄存器读写信号变化
- **存储指令**: 内存读写信号及数据总线变化

在测试框架中输出波形截图，从指令开始到结束依次查看核心控制信号（如PC、寄存器读写、ALU运算结果）的变化趋势。

(1) 取指

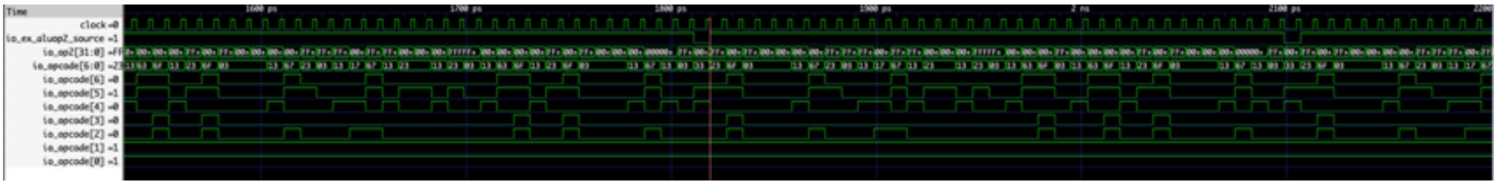
波形图



可以观察到当 `io_jump_flag_id = 1` 时， `pc` 会变为 `io_jump_address` ; 否则 `pc = pc + 4` 。

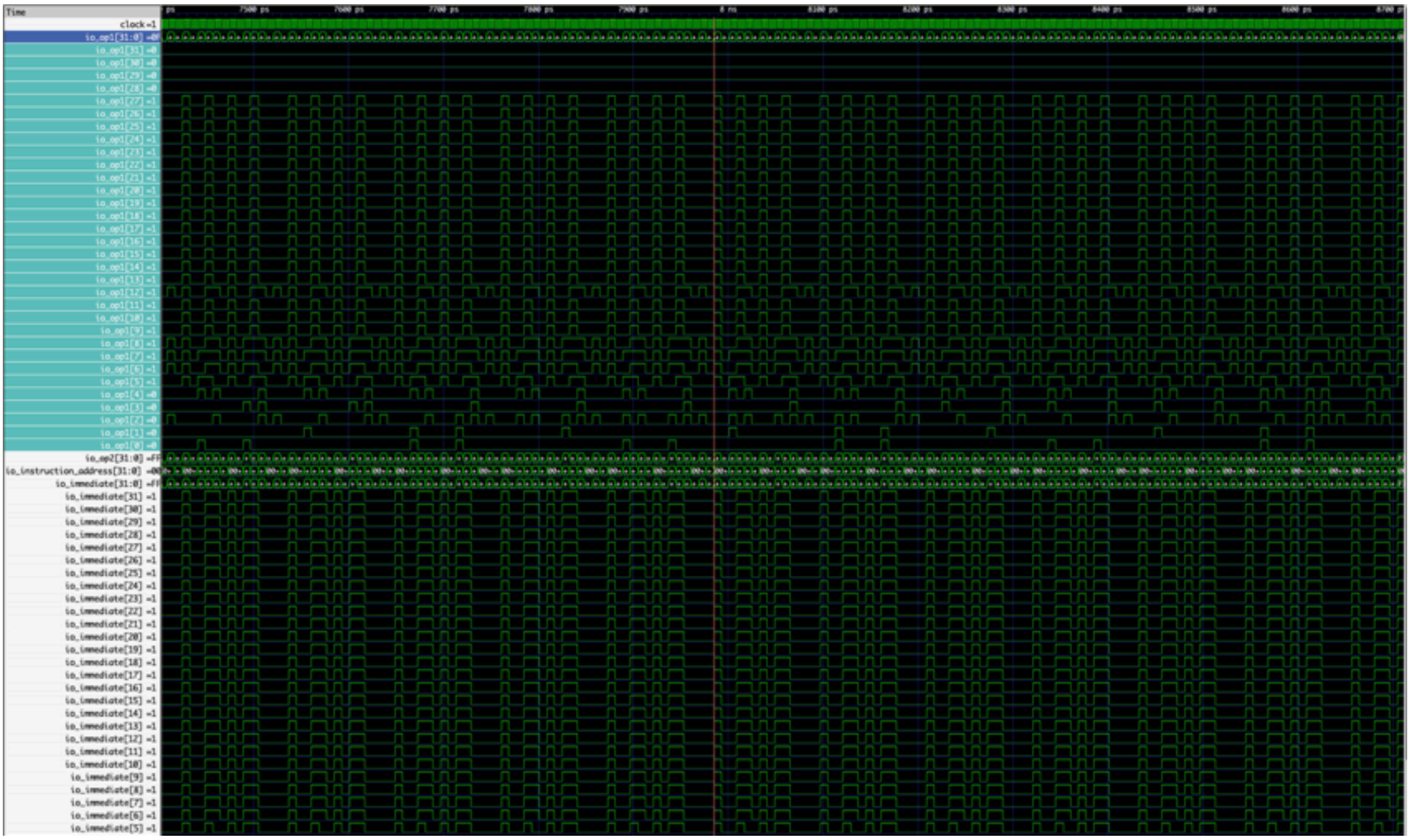
(2) 解码

波形图



`ex_aluop2_source` 的波形与 `opcode` 的类型相吻合。

(3) 执行



`alu_io_op1` 根据 `io_aluop_source1` 的取值选择接收指令地址或寄存器的数据。

`alu_io_op2` 根据 `io_aluop_source1` 的取值选择接收立即数或是寄存器数据。

`alu_io_funct` 接收来自控制通路的信号 `alu_ctrl.io.alu_funct` 以决定 `alu` 执行的运算。

波形结果符合预期。

4.外设输出验证

通过串口检测软件 `MobaXterm` 检测到串口输出，说明程序在实验板上正确运行