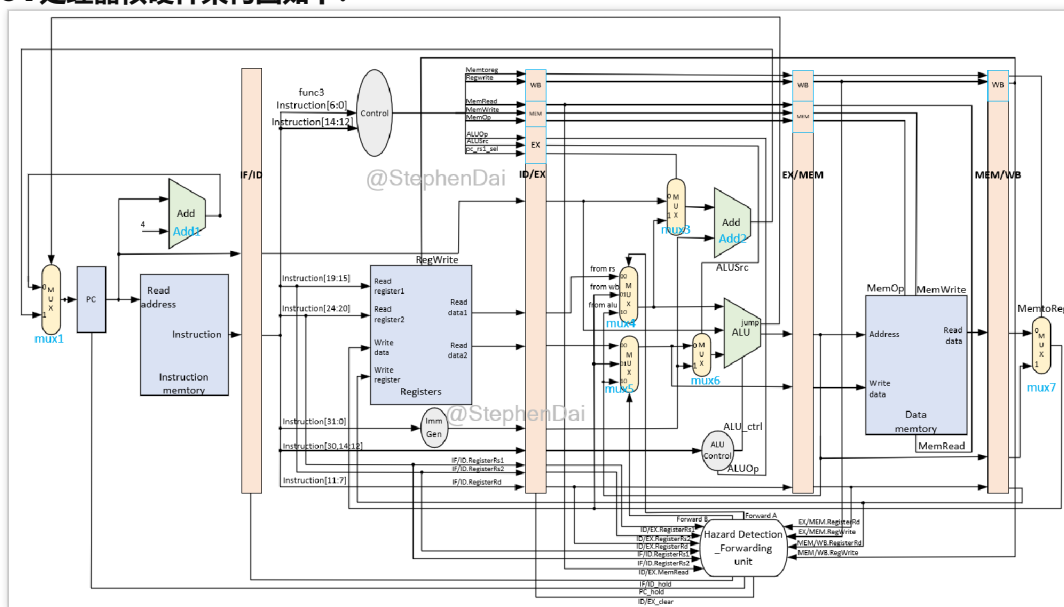


# 1. 概述

- 本项目用verilog语言实现了一个具有五级流水线的简单RISCV处理器核，实现RV32I指令集（37条）以及扩展指令集M（8条），并验证了每条指令实现的正确性。项目文件里面包含所有的设计和测试文件以及完整的makefile脚本，可以“开箱即用”。
- 本项目RISCV处理器的设计参考《计算机组成与设计：硬件软件接口 RISC-V版》一书，如果时间充足的话非常建议先阅读该书前四章再来学习；测试部分使用官方的指令测试文件测试。
- 在看代码之前，推荐先看笔者写的教程专栏：[RISCV处理器设计-CSDN博客](#)，搭配食用效果更佳。

实现的RISCV处理器核硬件架构图如下：



项目文件层次结构如下：

- **doc** 文件夹里面是riscv spec手册；
- **pic** 文件夹里面是硬件架构图；
- **tb** 里面的：
  - **generated** 文件夹包含每条指令测试文件的各种形式：
    - .dump文件是反汇编后的指令，可以查看各指令测试文件执行了什么命令
    - .txt文件是十六进制的机器码，是用于在tb里面的测试文件
    - 其余文件可以不看
  - **my\_inst\_txt** 文件夹包含一些我自己的写的测试指令，如几种冒险及一些基本跳转指令，你也可以运行一下去分析；
  - **inst\_txt** 是 **generated** 文件夹中的.txt格式测试文件，单独放一个文件夹
- **RISCV\_vivado\_xxx** 文件夹为vivado工程文件夹，里面已经包含了设计、验证、测试文件，直接打开里面的vivado工程就好了

## 2. 使用说明

本文件夹是基于vivado实现的，我用的vivado版本是2019.1

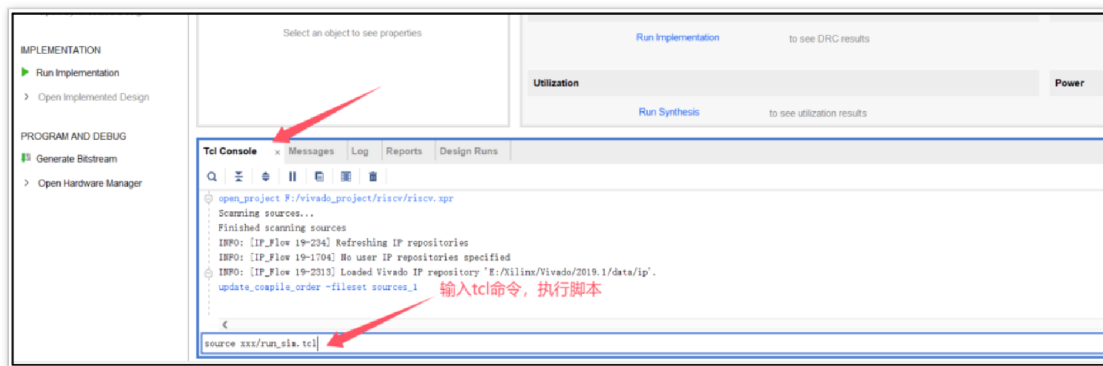
## 3. 如何跑起来

进入vivado工程文件夹，双击打开工程文件

.Xil	2025/4/3 15:36	文件夹	
riscv.cache	2025/4/3 15:36	文件夹	
riscv.hw	2025/4/3 15:36	文件夹	
riscv.ip_user_files	2025/4/3 15:36	文件夹	
riscv.sim	2025/4/3 15:36	文件夹	
riscv.srds	2025/4/3 15:36	文件夹	
sim_logs	2025/4/3 11:52	文件夹	
batch_run.tcl	2025/4/3 15:37	TCL 文件	2 KB
riscv.xpr	2025/4/3 15:32	Vivado Project F...	35 KB
run_sim.tcl	2025/4/3 15:33	TCL 文件	1 KB

双击打开工程文件

在 Tcl Console 中输入 tcl 命令，执行脚本。其中 xxx 为脚本文件的路径：



脚本说明：

- **run\_sim.tcl**：执行单条指令的测试程序

```
close_sim -quiet

set instr_file "rv32ui-p-add.txt"
set_property verilog_define "INSTR_FILE=\"${instr_file}\"" [get_filesets sim_1]

launch_simulation
# 执行仿真直到结束 (vivado启动仿真默认仿真1000ns，所有测试中最长测试时间为10150ns)
run 10000ns
```

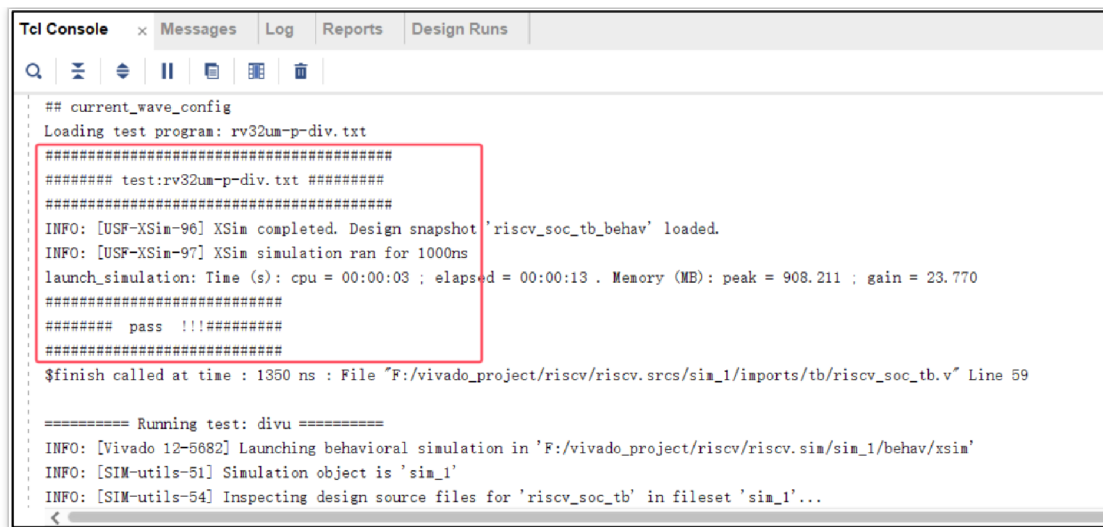
此处更改要执行的指令测试程序

- **batch\_run.tcl**：批量执行所有指令的测试程序

```
# rv32ui 指令测试列表
set ui_test_list {
    "add" "sub" "sll" "slt" "sltu" "xor" "srl" "sra" "or" "and" "lui" "auipc" "jal"
    "jalr" "beq" "bne" "blt" "bge" "bltu" "bgeu" "lb" "lh" "lw" "lbu" "lhu" "sb"
    "sh" "sw" "addi" "slli" "sltiu" "xori" "ori" "andi" "slli" "srli" "srai"
}

# rv32um 指令测试列表
set um_test_list {"div" "divu" "mul" "mulh" "mulhsu" "mulhu" "rem" "remu"}
```

输出打印信息中可以看每条指令测试程序的执行情况：



```
Tcl Console x Messages Log Reports Design Runs
[Icons]
## current_wave_config
Loading test program: rv32ui-p-div.txt
#####
##### test:rv32ui-p-div.txt #####
#####
INFO: [USF-XSim-96] XSim completed. Design snapshot 'riscv_soc_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:13 . Memory (MB): peak = 908.211 ; gain = 23.770
#####
##### pass !!!#####
#####
$finish called at time : 1350 ns : File "F:/vivado_project/riscv/riscv.srcs/sim_1/imports/tb/riscv_soc_tb.v" Line 59

===== Running test: divu =====
INFO: [Vivado 12-5682] Launching behavioral simulation in 'F:/vivado_project/riscv/riscv.sim/sim_1/behav/xsim'
INFO: [SIM-utils-51] Simulation object is 'sim_1'
INFO: [SIM-utils-54] Inspecting design source files for 'riscv_soc_tb' in fileset 'sim_1'...
```

## 4. 参考资料

《计算机组成与设计：硬件软件接口 RISC-V版》

## 5. 跟新日志

2023.08.31 : RISC-V基础版

2024.03.24 : 修正了一些bug、增加了一些脚本命令

2024.05.01 : 增加cache

2024.12.21 : 修正立即数扩展单元bug, 支持扩展指令集M

## 6. 后续计划

后续有时间考虑更新:

- 增加中断
- 实现分支预测
- 增加cache缓存 (已填坑)
- 实现扩展指令集M (已填坑)
- 完善处理器的测试, 跑个回归测试...

## 7. 注

- 指令测试文件rv32ui-p-sh.dump及对应txt文件修改地方: 地址0x60处的sh指令修改为sw指令
- 指令测试文件rv32ui-p-sb.dump及对应txt文件修改地方: 地址0x58处的sb指令修改为sh指令
- Load型指令测试文件ram有初始值无法验证, 但Store型指令测试文件中已经包含了Load型指令的验证