计算机设计与实践 HITSZ - miniRVCPU

项目结构

```
├─ single-cycle 单周期 CPU 代码
│  ├─ trace trace 比对代码
│  │   ├─ in IROM 和 DRAM 在 CPU 里面
│  │   ├─ out IROM 和 DRAM 在 CPU 外面
│  │   └─ wrong-example 错误示例代码
│  └─ onBoard 开发板运行代码
│      ├─ test1 测试 1 上板代码（实验性质）
│      └─ test2 测试 2 上板代码（通用）
├─ pipeline 流水线 CPU 代码
│  ├─ trace trace 比对代码
│  └─ onBoard 开发板运行代码（通用）
├─ report 实验报告
│  ├─ report.md 总报告
│  ├─ 0.3.md 资源使用情况、功耗数据截图（实现后）
│  ├─ 1.2.md 单周期 CPU 模块详细设计
│  ├─ 1.3.md 单周期 CPU 仿真及结果分析
│  ├─ 2.3.md 流水线 CPU 模块详细设计
│  ├─ 2.4.md 流水线 CPU 仿真及结果分析
│  ├─ 3.1.md ALU 模块设计的细微差别
│  ├─ 3.2.md 检测数据冒险的判断条件
│  └─ 4.md 总结
├─ _srcs 资源文件
│  └─ ... （清单不在此列出）
└─ _images 图片仓库
   └─ ... （清单不在此列出）
```

参考

- https://hitsz-cslab.gitee.io/cpu
    - (指导书年年在变)
- https://github.com/HITSZ-CDP/cdp-tests
    - (测试框架年年在变)
- https://github.com/xyfJASON/HITSZ-miniRV-1
- https://github.com/Yikai-coder/HITsz_CPU_design
- https://github.com/FinCreWorld/miniRV-1

实现

- 实现了单周期 CPU 以及流水线 CPU，涉及 24 条基本指令。

- 分支预测功能：采用静态分支预测，默认不进行跳转。
- 实现了结构清晰、可高度复用的总线外设。

Tips

在使用我的代码之前建议先完整的看完 README 和实验报告，至少要有一个印象；

工程的代码不一定能直接跑 trace 或者上板，后面复用的前面的代码也不一定是完全复用，或多或少进行了修改，所以直接抄我代码的人可能会一头雾水。

我自信我丰富的注释可以解决大部分疑惑，故而只要你认真看过一遍代码，想清楚原理，那么一定会对你大有帮助。

工程

各个工程在时间上大概是递进的，因此后面的工程或多或少写的比前面的更合理、更简洁，但我仍然认为展示思考的过程也是很重要的；实际上，每个工程也会有微妙的差别，因此上一个工程的代码不能完美的复用到下一个步骤，所以建议比对一下文件的差别，看看相较于之前修改了什么。

- single_cycle
    - trace
        - in：IROM 和 DRAM 在 CPU 里面。
        - out：IROM 和 DRAM 在 CPU 外面。
        - wrong_example：基于 out 修改产生的错误版本。
    - onBoard
        - test1：运行给定的 IROM 的指令，在数码管上显示 2500_0018。
            - 在 out 的基础上，使用老师提供的 ip 核，增添了数码管的外设（**做法是偷懒直接显示 x8!**），**删除了 debug 的输出。**
        - test2：将自己的汇编指令导入 IROM，实现计算器的功能。
            - 在 test1 的基础上，增添了拨码开关和 led 灯的外设（正式的外设），提供了另一种数码管的 DISPLAY 方法（在注释中）。
        - ~~final：为测试 CPU 最大频率的版本。在 test2 的基础上，将绝大多数寄存器变量修改为线网，减少寄存器延迟。~~
        - test2 设计好了外设，因此上板的两个实验（trace 上板、实现计算器）都可以使用 test2 的代码，只不过需要修改 IROM 和 DRAM 的 IP 核。
- pipeline
    - trace
        - 和单周期区别较大。
    - onBoard
        - 删除了 debug 的输出，复用单周期总线外设代码，可用于 test1、test2 两个实验。

single_cycle/trace/in

把 IROM 和 DRAM 的实例化延迟到阶段中。

- top
    - IF

        - PC
        - NPC
        - IROM
    - ID/WB
        - SEXT
        - RF
        - MUX4_1
    - EX
        - ALU
        - MUX2_1
    - MEM
        - DRAM

single_cycle/trace/out

在 top 里实例化 IROM 和 DRAM，**便于后续上板。**

- top
    - miniCPU
        - IF
            - PC
            - NPC
        - ID/WB
            - SEXT
            - RF
            - MUX4_1
        - EX
            - ALU
            - MUX2_1
    - IROM
    - DRAM

single_cycle/trace/wrong_example

结构同 out，修改为大多数寄存器改线网，使用大量 assign 和三目运算符，错误原因请看报告。

single_cycle/onBoard/test1

- top
    - clk_div：选择 PLL，选择 global buffer，不需要 reset 和 lock；初期尝试可使用 10MHz 分频。
    - miniCPU：内部结构略。
    - IROM：使用老师提供的 IROM 的 IP 核。
    - DRAM：使用老师提供的 DRAM 的 IP 核。
    - DISPLAY：数码管的显示部件，*十六进制数字的显示方法可能各异，请注意*。

single_cycle/onBoard/test2

- top

    - clk_div：设置略，可以尝试使用更高的频率，如 25 MHz。
    - prgrom：建一个空的 IROM 的 IP 核，**导入自己写的计算器的机器码。**
    - miniCPU：将部分输出端口名字修改的符合规范，更适合阅读。
    - BUS：总线桥，模仿实验一 SOC 设计。
    - MEM：对 dram 的包装，将输入输出端口名字修改的符合规范，更适合阅读。
        - dram：建一个空的 DRAM 的 IP 核。
    - SwitchDriver：开关外设。
    - LedDriver：LED 外设。
    - DigitDriver：七段数码管外设。
        - DISPLAY：数码管的显示部件。

pipeline/trace

- top
    - miniCPU
        - HAZARD_DETECTION
        - IF
            - PC
            - NPC
        - REG_IF_ID
        - ID/WB
            - CONTROLLER
            - SEXT
            - RF
        - REG_ID_EX
        - EX
            - ALU_MUX
            - ALU
            - NPC_CONTROL
            - WD_MUX1
        - REG_EX_MEM
        - MEM
            - WD_MUX2
        - REG_MEM_WB
    - IROM
    - DRAM

pipeline/onBoard/test1、test2

miniCPU 外部同 single_cycle/onBoard/test2 的结构。