计算机体系结构实验课程第二次实验报告

实验名称	流水线 CPU			班级	李雨森班
学生姓名	姚知言	学号	2211290	指导老师	董前琨
实验地点	实验楼 A306		实验时间	2024 年 9 月 23 日	

1、 实验目的

在多周期 CPU 实验完成的提前下，深入理解 CPU 流水线的概念。

熟悉并掌握流水线 CPU 的原理和设计。

最终检验运用 verilog 语言进行电路设计的能力。

通过亲自设计实现静态 5 级流水线 CPU，加深对计算机组成原理和体系结构理论知识的理解。

培养对 CPU 设计的兴趣，加深对 CPU 现有架构的理解和深思。

2、 实验内容说明

复习好上一次多周期 CPU 的实验，归纳常用的 MIPS 指令，确定自己准备实现的 MIPS 指令，对其进行分析；

认真学习流水线的概念，明白流水线的意义和架构，理解数据相关、控制相关和结构相关，尤其需要注意分支跳转指令及其延迟槽指令的处理；

依据自己设计中实现的指令，编写一段不少于 50 行的汇编程序，要求包含所有实现的指令。要求标注出指令间存在的相关，指出 CPU 可能存在的阻塞；

编写 verilog 代码，将表 9.2 中自己编写的汇编程序翻译为二进制，以 coe 文件的方式初始化到指令 ROM 中；

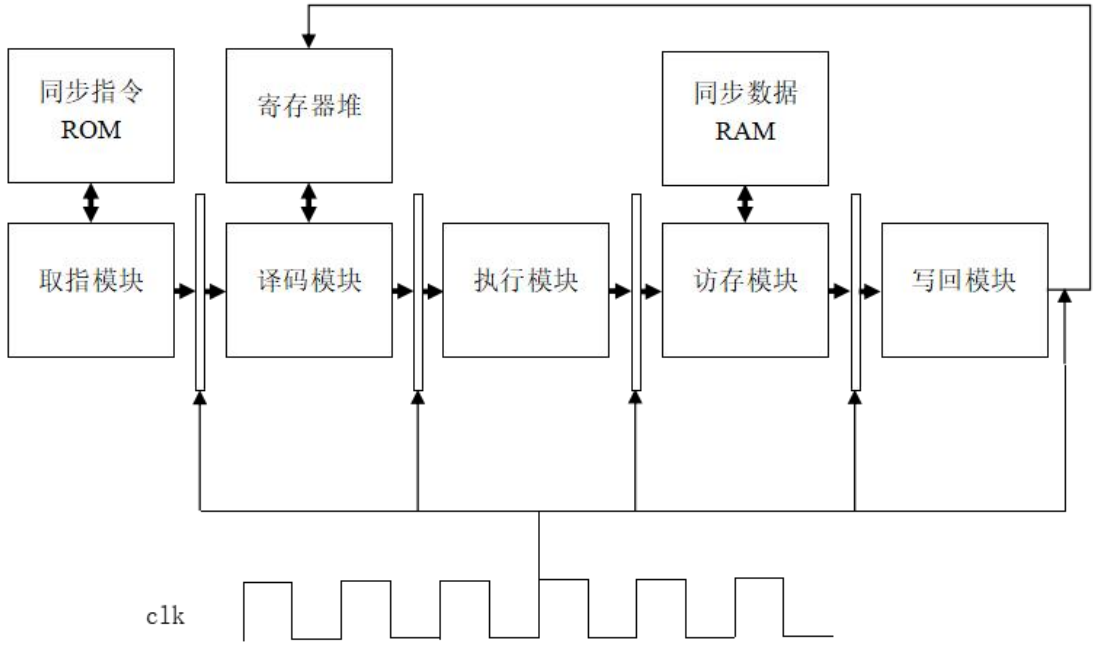
对该模块进行仿真，得出正确的波形，截图作为实验报告结果一项的材料，在仿真时需要将生成指令 ROM 时产生的.mif 文件拷贝到工程目录下，才能仿真成功；

完成调用流水 CPU 的外围模块的设计，并编写代码；

对代码进行综合布局布线下载到实验箱里 FPGA 板上，进行上板验证。

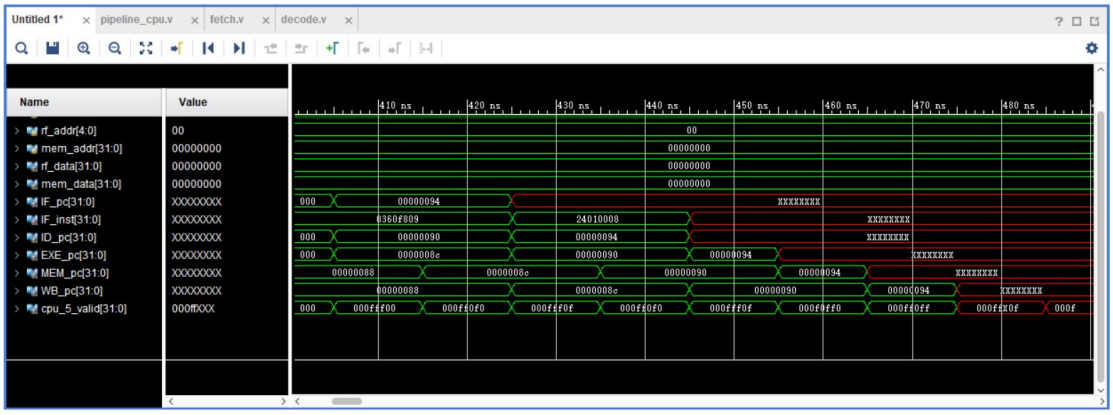
实验结束后，需按照规定的格式完成实验报告的撰写。

3、 实验原理图



4、 实验步骤

（1）分析执行过程，找到写后读问题



程序存在较为严重的写后读问题，在此举出较为容易发现的两例：

通过对结果的仿真，可以发现在 90H 执行（jalr $27，跳转到 50H 并为$31 赋值）后，进入 94H 执行阶段。随后程序出错。这说明在 90H 执行后，并没能顺利的取到跳转地址。

此外，在 48H 执行结束后，也先执行了 4CH，然后再执行 84H。但注意到 4CH 本应实现的（subu $5, $3,$4，为$5 赋值为 0DH）操作并没执行，即$5 的数据并没能得到定义（忘记截图了），起初我认为这是流水线 CPU 自动舍弃的方式，但经过接下来的实验验证我认为这一部分问题也来自于写后读数据访问的错误。

对于读后写问题，我认为在原始实验背景下并不存在。

（2）改进验证

经过分析，我尝试过修改 decode 中 wait 的判定，为 decode 阶段添加旁路等，但都未能成功。通过对波形图的分析，我发现 alu，mem，wb 的 wdest 更新会有延迟，因此我尝试在 decode 阶段预测后续要用到的 wdest，并增加 wait 判定。然而这种方法也未能顺利解决问题。

我也尝试了更改 IP 核的方法，更改 IP 核去掉 IP 核的延迟后可以顺利解决该问题，不过出于对可能引发其他 bug 的考量，我并没有最终选择该方法。

最终，为迎合 IP 核的延迟，我将 fetch 阶段的时钟周期延长一拍，保证了 fetch 能够顺利的取到跳转地址。

代码修改部分如下（其中修改部分标红表示）：

```
reg IF_mid;
always @(posedge clk)
begin
    if (!resetn || next_fetch)
    begin
        IF_over <= 1'b0;
        IF_mid <=1'b0;
    end
    else if(!IF_mid)
    begin
        IF_mid <= IF_valid;
    end
    else begin
        IF_over <= IF_mid;
```

```
        end
    end
```

（3）现有 CPU 的不足之处分析

1.当前流水线 CPU 对于数据相关/控制相关问题，采取等待的方法，这样对 CPU 的效率影响是比较大的，可以采用数据前递的方法进行优化。

2.可以采取分支预测的技术，减少在分支发生的时候额外的等待时间。

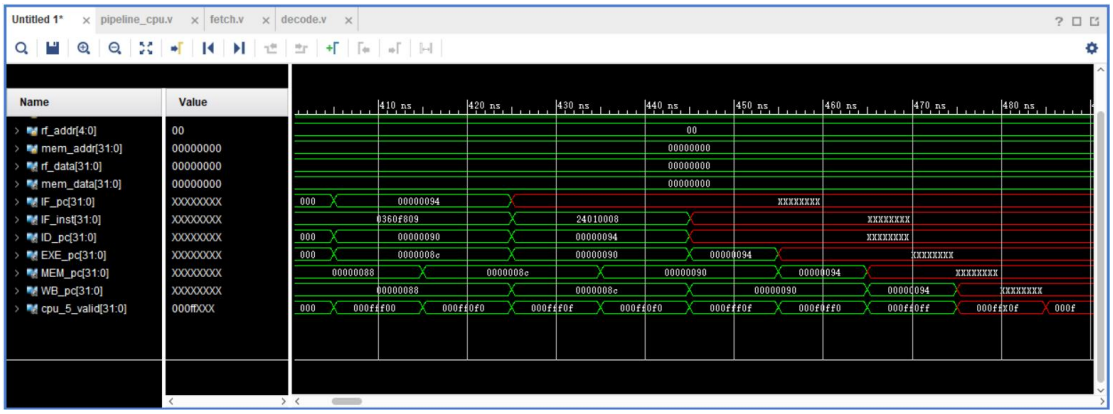
3.当前一个周期只会执行一条指令，可以考虑使用多发射技术来加快执行效率。

4.设计 cache，减少 mem 访存时间。

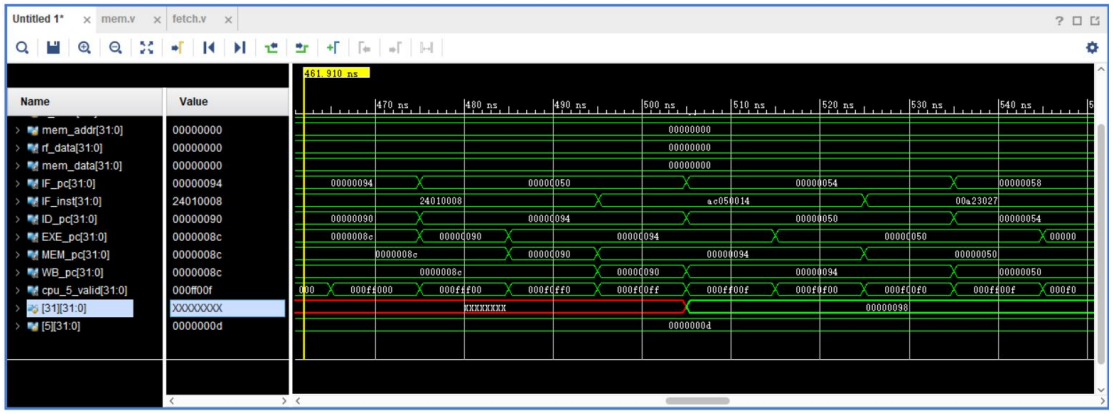
5.解决当前流水线 CPU 存在的问题，即在分支跳转之前会多执行跳转语句的后一条语句的问题。

5、 实验结果分析

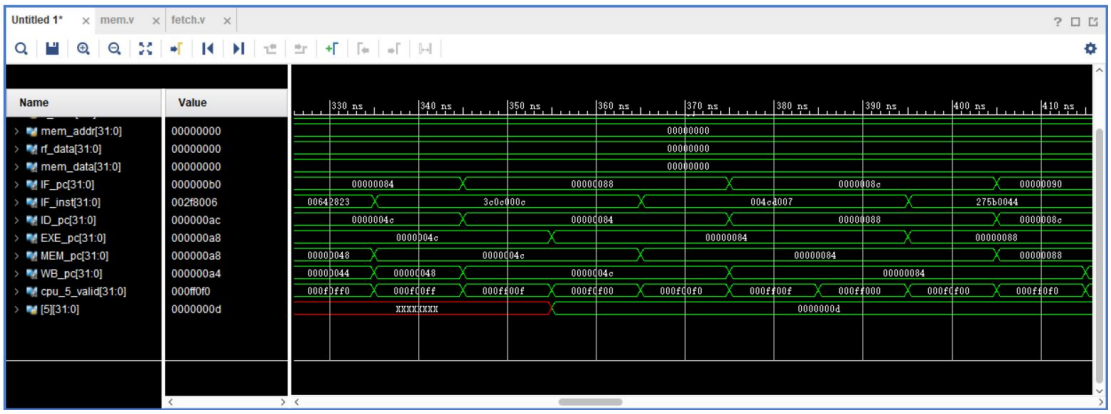
改进前的仿真结果，同上一部分图，在此重新展示以便对比。



改进后的仿真结果如下：



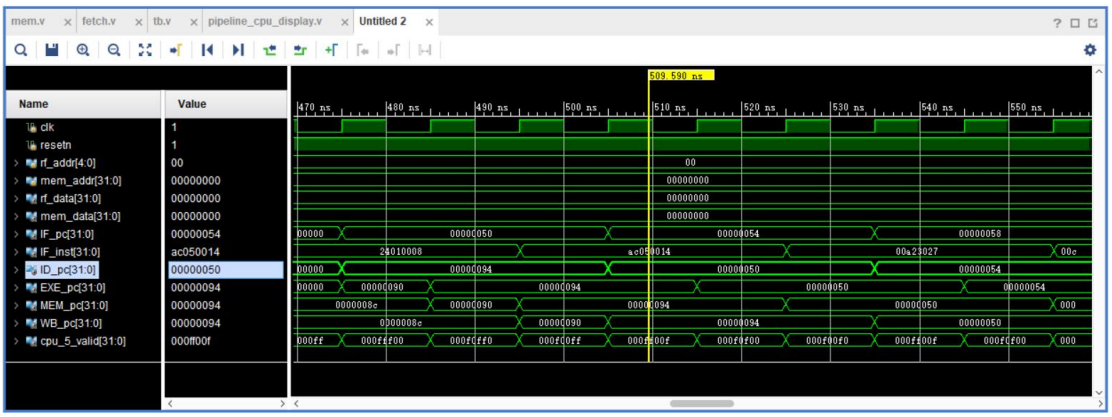
可以看到，成功跳转到 50H 的执行，并且$31 成功被赋值 98H，这说明 90H 的指令成功被执行，问题成功被解决。

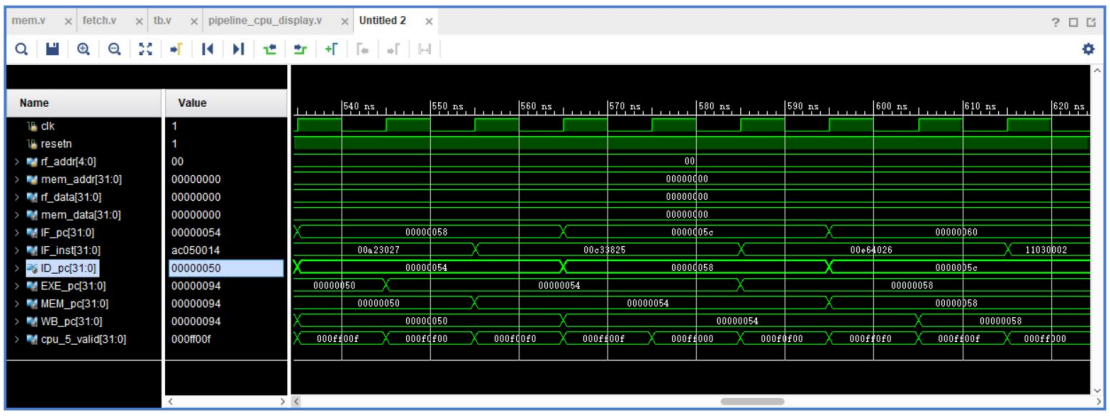


对 4CH 指令的补充说明部分，可以看到改进后 4CH 确确实实被多执行了，并且返回了正确的结果，对于这一问题，根据老师要求先不在本次实验中修改。

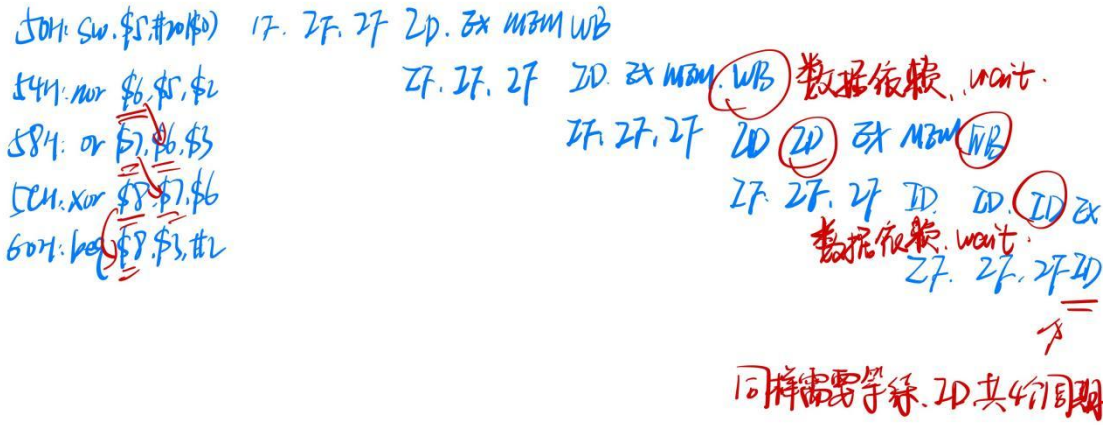
接下来，我对 50H-60H 这五条指令的流水线执行进行详细分析（选取原因：包含较为严重的数据依赖）。

仿真结果如下：





分析如下：



执行周期如图所示，可以看到 CPU 在严格数据依赖的情况下也能够正常生成结果，但也让我们看到在数据依赖较多的情况下浪费时间是较多的。

6、 总结感想

在这次实验中，我学习了流水线 CPU，并进行了 Bug 的修复以及存在的不足情况分析。此外，我对流水线 CPU 的执行周期进行了案例分析。这次实验让我深刻熟悉了流水线 CPU 的架构，也为后续性能优化打下良好的基础。

注意： 1、班级用任课老师姓名表示。

2、实验报告提交的文件名为“学号_姓名_组成原理第一次实验.pdf”，注意要导出成 pdf 文件。