**体系结构实验课程第 二 次实报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 五级流水线改进 | | | 班级 | 李雨森老师班 |
| 学生姓名 | 秦泽斌 | 学号 | 2212005 | 指导老师 | 董前琨 |
| 实验地点 | 实验A304 | | 实验时间 | 2024.10.12 | |

1. **实验目的**

1. 在多周期CPU实验完成的提前下，深入理解CPU流水线的概念。

2. 熟悉并掌握流水线CPU的原理和设计。

3. 最终检验运用verilog语言进行电路设计的能力。

4. 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。

5. 培养对CPU设计的兴趣，加深对CPU现有架构的理解和深思。

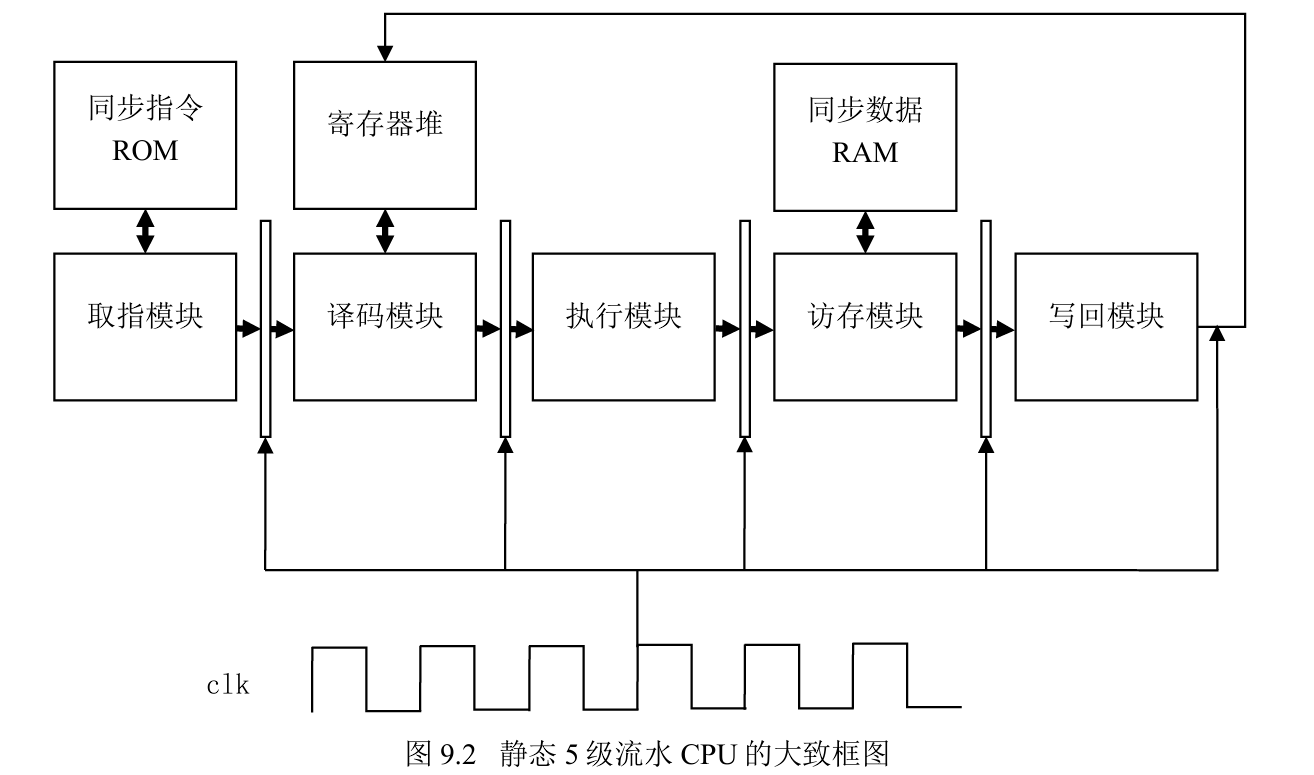
1. **实验内容说明**

1、分析现有指令的执行过程，找到存在的读后写/写后读问题，并进行改进验证（推荐自己编写指令rom中的指令，针对存在的问题进行测试改进）然后写入实验报告。

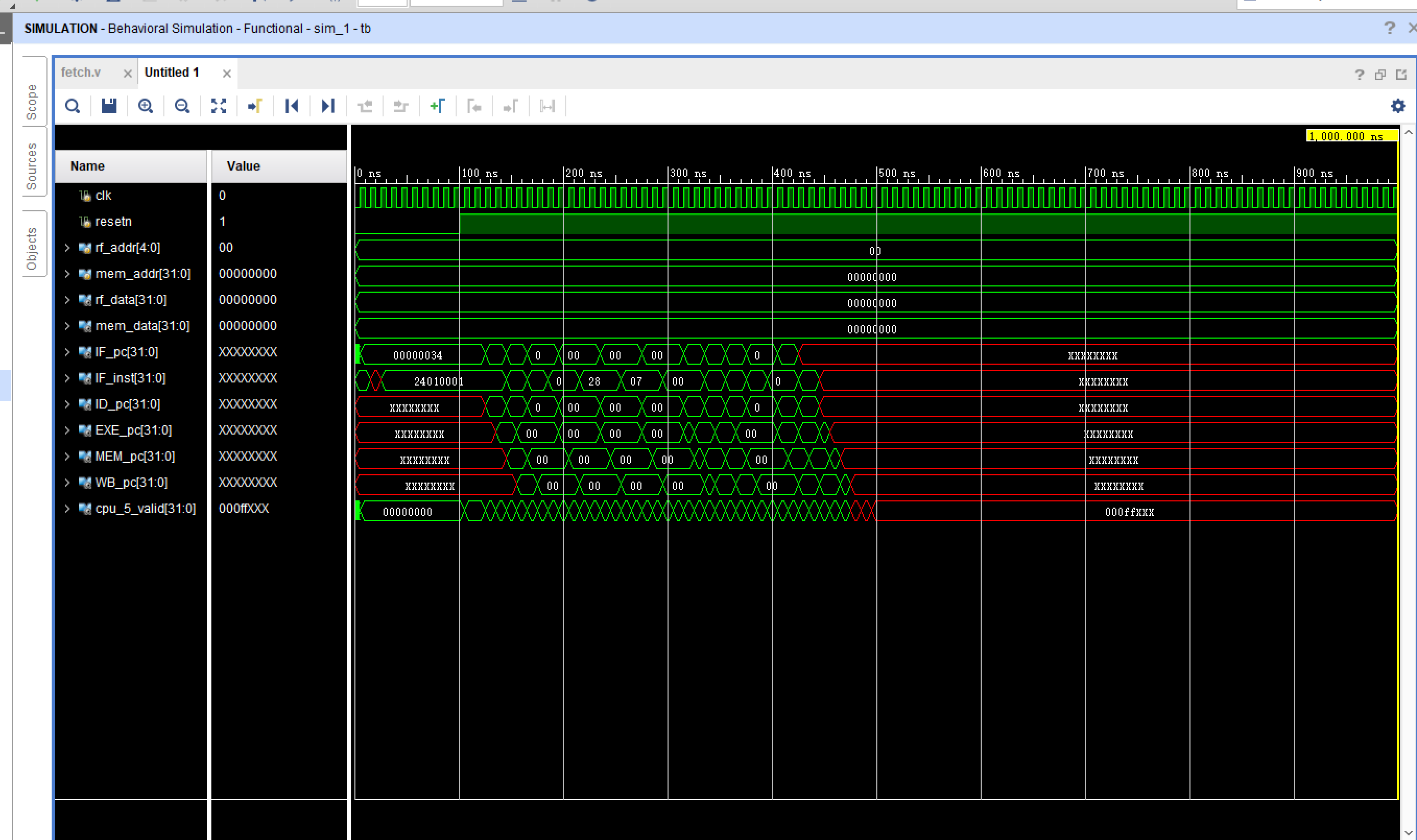
2、在做1的过程中，寻找现有CPU的不足之处，提出一些自己的改进方案和想法，包括但不限于实验指导手册第十章中的优化部分，并初步讨论一下可行性，为最后一次实验做准备。

3、实验报告中的原理图放图9.2即可。注意要在实验报告中详细说明改进过程，验证时先描述改进前的情况，再说明改进后的情况，进行具体对比。

1. **实验原理图**



1. **实验步骤**
2. 分析现有五级流水线架构并进行模拟仿真，发现存在读后写/写后读的问题，cpu不能正常运行



1. 通过分析代码可以得知问题出现在fetch.v中

//-----{IF执行完成}begin

//由于指令rom为同步读写的,

//取数据时，有一拍延时

//即发地址的下一拍时钟才能得到对应的指令

//故取指模块需要两拍时间

//故每次PC刷新，IF\_over都要置0

//然后将IF\_valid锁存一拍即是IF\_over信号

always @(posedge clk)

begin

if (!resetn || next\_fetch)

begin

IF\_over <= 1'b0;

end

else

begin

IF\_over <= IF\_valid;

end

end

//如果指令rom为异步读的，则IF\_valid即是IF\_over信号，

//即取指一拍完成

//-----{IF执行完成}end

1. 通过增加一拍延迟可以解决相关问题

reg temp;

always @(posedge clk)

begin

if (!resetn || next\_fetch)

begin

IF\_over <= 1'b0;

temp<=1'b0;

end

else if(!temp)

begin

temp<=1'b1;

end

else

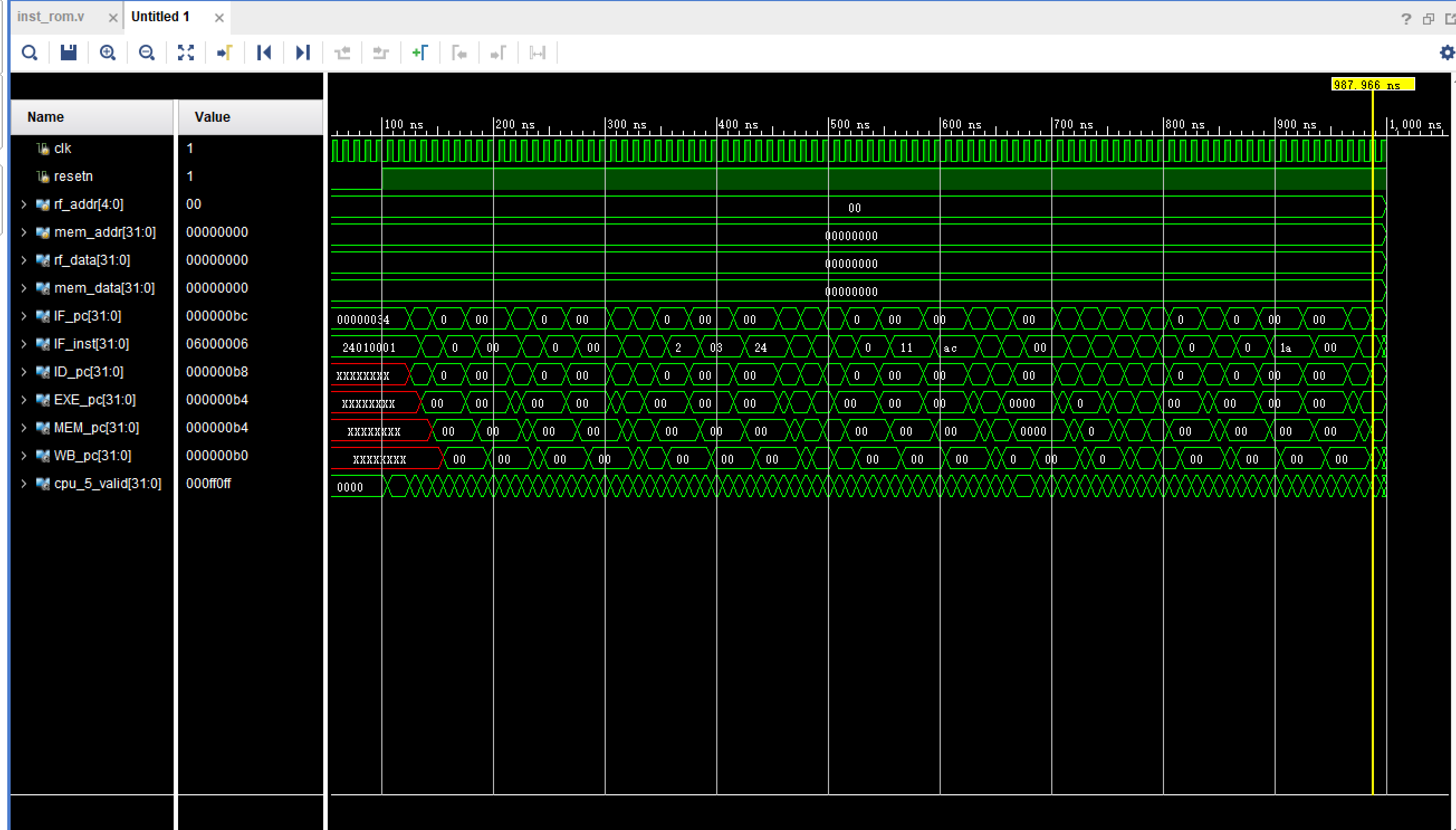
begin

IF\_over <= IF\_valid;

end

end

1. **实验结果分析**

1、改进五级流水线

CPU正常运行，问题解决。

2、寻找现有CPU的不足之处，提出一些自己的改进方案和想法

（1）、控制冒险问题：当遇到分支跳转指令时，由于无法在流水线早期阶段确定是否需要跳转，可能导致流水线错误地取指令，产生控制冒险。

改进方案：

* 分支预测（Branch Prediction）： 引入简单的静态分支预测机制，例如：总是预测“无跳转”或根据历史判断跳转。如果误预测，则撤回指令并重新取指。
* 动态分支预测： 可以使用两位饱和计数器来追踪分支指令的执行情况，动态预测是否跳转。动态预测能够根据历史执行情况调整预测，减少错误预测的概率。

（2）、结构冒险问题：由于只有一个寄存器堆或者内存模块，可能导致多个指令同时需要访问同一个资源而产生冲突。

改进方案：

* 增加多端口寄存器堆： 使用双端口寄存器堆，使得同时的读/写操作不再冲突。例如在写回阶段写寄存器时，译码阶段仍可以读取寄存器而不发生冲突。
* 拆分数据路径： 将数据存储和指令存储分开，避免指令和数据存储的冲突，尤其是在访存和取指操作需要同时进行时。

等等

1. **总结感想**

通过本次实验，我加深了对静态五级流水线CPU的认识以及对RAW&WAR等bug的分析，增强了自己发现问题的能力，同时也提高了解决问题的能力。