体系结构实验课程第二次实报告

实验名称	多周期 CPU 实现			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 A306		实验时间	2024.9.21	

1、 实验目的

（1）. 在单周期 CPU 实验完成的提前下，理解多周期的概念。

（2）. 熟悉并掌握多周期 CPU 的原理和设计。

（3）. 进一步提升运用 verilog 语言进行电路设计的能力。

（4）. 为后续实现流水线 cpu 的课程设计打下基础。

2、 实验内容说明

请同学们结合实验指导手册的多周期实验内容，实现多周期 CPU 功能并完成如下内容：

1、在给出代码的基础上，完成多周期 CPU 的功能仿真，分析在指令 rom 中的指令的执行过程。

2、找到程序执行过程中存在的 bug 并尝试修复（修复方式不唯一，建议多做尝试）。

3、完成实验报告，并在报告中介绍自己的分析、bug 追踪核调试过程，并做好自己的总结，为后续实验做准备。（本次实验着重看仿真结果，可以不上箱验证）

3、 实验原理图

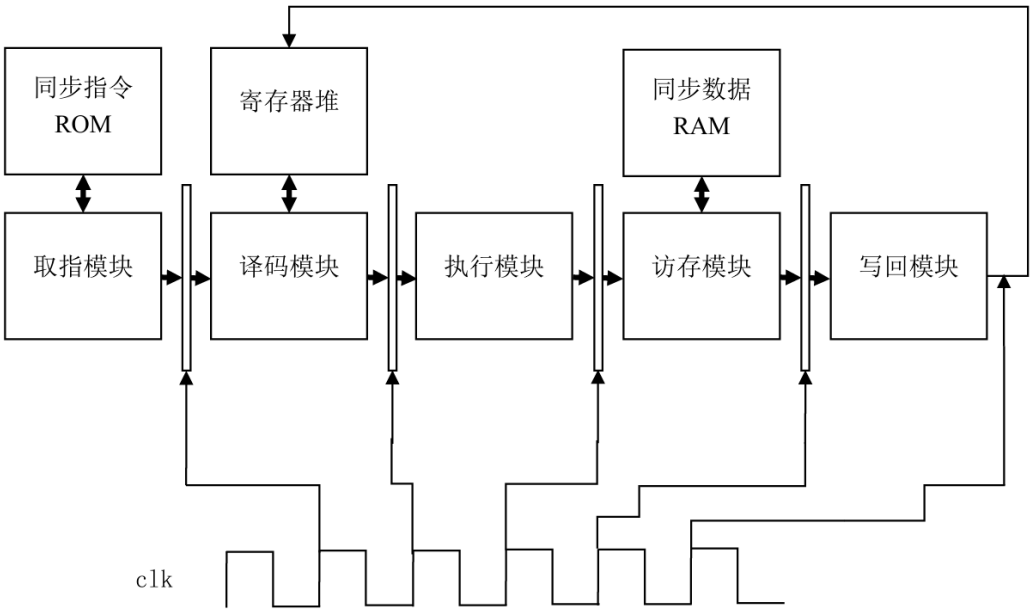


图 8.1 多周期 CPU 的大致框图

多周期 CPU 设计在单周期 CPU 基础上，主要做两部分改进。第一部分是控制单元，增加控制电路使每一个时钟只有一个阶段的电路产生的结果有效，并锁存上一阶段的结果用于后续阶段的运行；第二部分是数据通路，增加实现新增指令的电路。

4、 实验步骤

（1） 分析在指令 rom 中的指令的执行过程。

在多周期 CPU 设计中，指令的执行过程经过了多个阶段，包括取指（Fetch）、解码（Decode）、执行（Execute）、访存（Memory Access）、写回（Write Back）。每个阶段都涉及不同的模块，逐步处理从指令 ROM 中取出的指令。

1. 指令取指阶段（Fetch） `fetch.v`

在取指阶段，CPU 从指令 ROM 中读取指令，并将其传递给下一阶段进行处理。这个阶段的关键是：

PC（Program Counter）：程序计数器决定了下一条要取的指令地址。

指令 ROM：指令 ROM 根据 PC 的值提供对应的指令。

在`fetch.v`文件中：

`inst_addr` 表示指令的地址，通过 PC 值来确定当前取的指令。

`inst_rdata` 表示从指令 ROM 中读取的指令。

当有效信号 `F_valid` 为真时，指令从指令 ROM 中取出，传递到解码阶段。PC 的更新可以是顺序的（即 PC+4）或者跳转（由分支指令或者跳转指令决定）。

2. 指令解码阶段（Decode） `decode.v`

解码阶段负责解析从取指阶段传递下来的指令，提取操作码（opcode），寄存器操作数等信息。这个阶段的任务是：

提取操作数：根据指令的格式，解码操作数、立即数、跳转地址等。

控制信号生成：根据操作码生成控制信号，指导后续阶段的执行。

在`decode.v`文件中：

`rf_rdata1` 和 `rf_rdata2` 是从寄存器堆中读取的操作数。

`ID_EXE_bus` 包含了解码后的指令和操作数，并传递到执行阶段。

3. 执行阶段（Execute） `exe.v`

执行阶段通过 ALU（算术逻辑单元）执行算术运算、逻辑运算、跳转等操作。在这个阶段，关键是：

ALU 操作：根据解码阶段的控制信号，执行对应的运算。

跳转与分支：如果是跳转指令，ALU 将计算出新的 PC 值。

在`exe.v`文件中：

`alu_result` 是 ALU 运算的结果。

`EXE_MEM_bus` 将执行结果（包括 ALU 结果、存储控制信号、写回信息等）传递到访存阶段。

4. 访存阶段（Memory Access） `mem.v`

访存阶段负责处理 Load 和 Store 指令，从数据存储器读取或向其写入数据。这个阶段的任务是：

读写数据存储器：根据指令类型（Load/Store）和地址（ALU 结果），从数据存储器中读取数据或写入数据。

在`mem.v`文件中：

`dm_addr` 是访存地址，由 ALU 的运算结果决定。

`dm_wen` 和 `dm_wdata` 控制数据写入操作。

`load_result` 是访存读出的数据，如果是`Load`指令则传递给写回阶段。

5. 写回阶段（Write Back） `wb.v`

写回阶段负责将运算结果（或者访存读取的数据）写回寄存器堆。该阶段的任务是：

写回寄存器堆：将 ALU 的结果或者从存储器加载的值写回目的寄存器。

在`wb.v`文件中：

`rf_wen` 表示寄存器堆的写使能信号。

`rf_wdata` 是写回寄存器的数据。

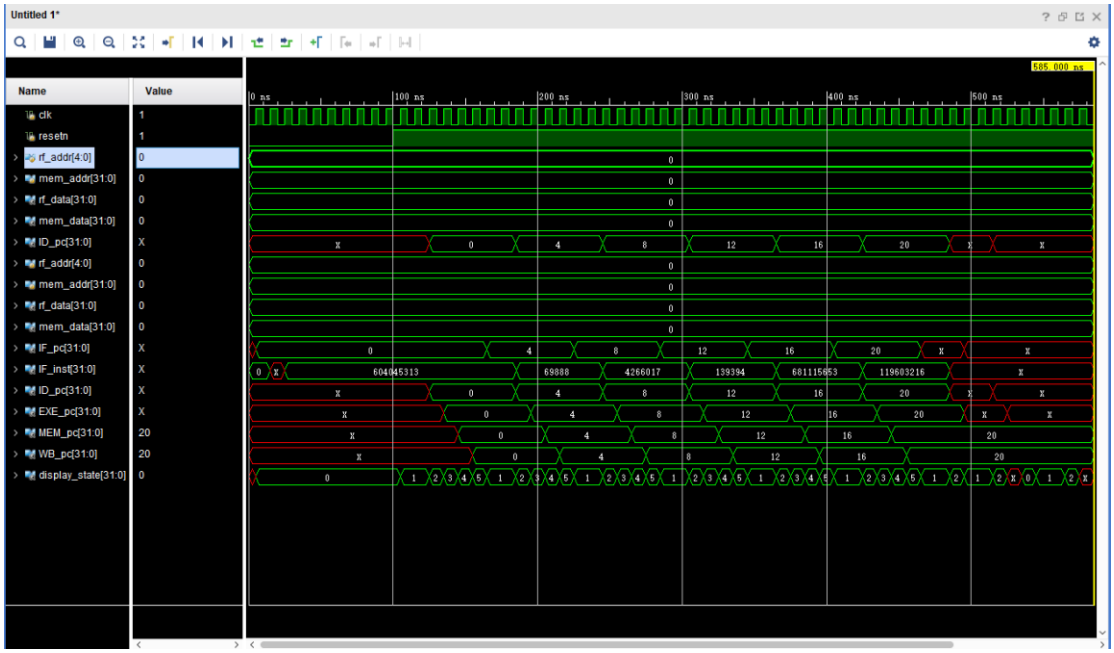
（2） 找到程序执行过程中存在的 bug 并尝试修复

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。所以我们尝试将 fetch 的触发时间往后延迟一拍，这里采取的操作是将 fetch 的锁存信号改为在时钟的下沿触发。

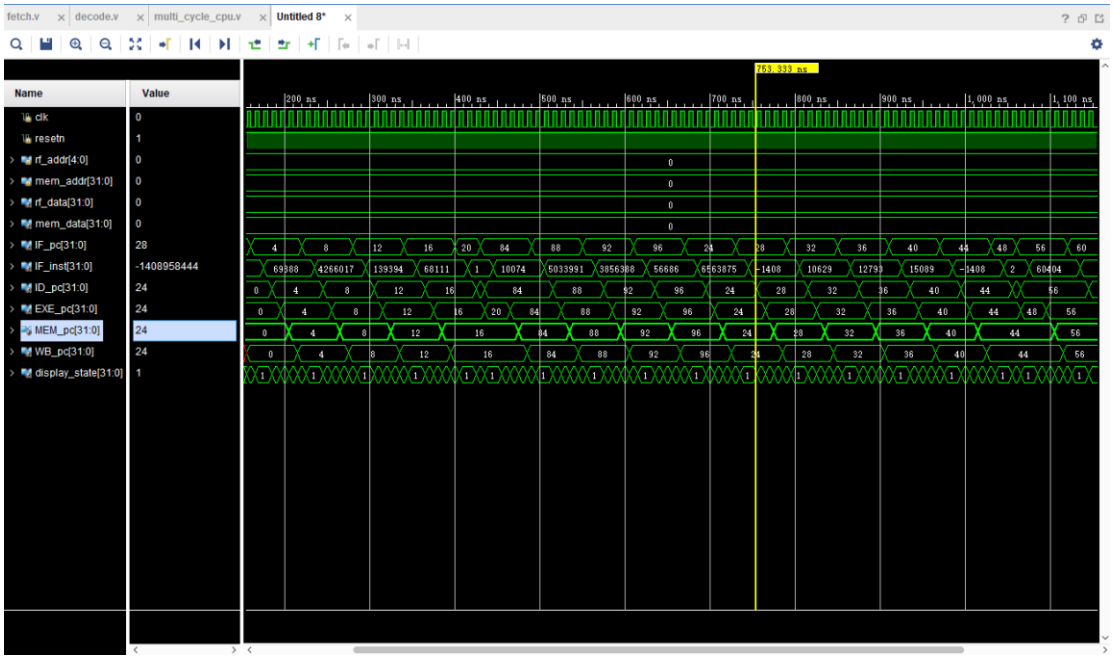
```
//IF到ID的锁存信号
always @(negedge clk)
begin
    if(IF_over)
    begin
        IF_ID_bus_r <= IF_ID_bus;
    end
end
```

5、 实验结果分析

源码仿真结果：一些信号在特定的时间内呢会出现红色 x 的不稳定状态



修改 bug 后的仿真结果：红色状态消失，所有信号正常。



6、 总结感想

通过本次实验，我更加熟悉了解了多周期 CPU 的实现原理，同时也提高了我的代码调试能力等。