一. 课设成果名称

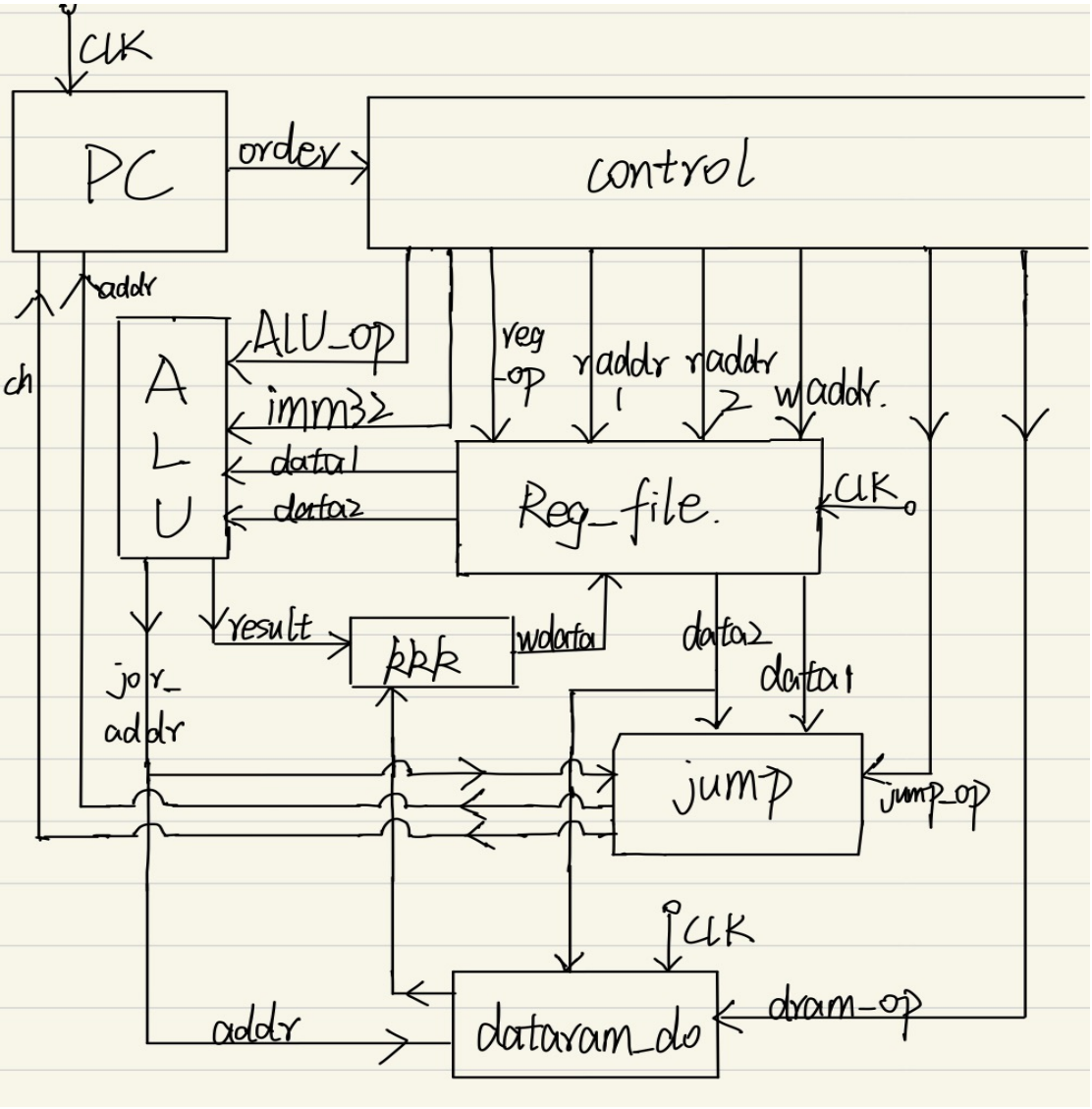
实现 20 条指令的单周期 CPU

二. 总体设计思路

CPU 设计的重要思想在于由顶向下,代码模块化.

通过 CPU 设计实战 该书附录的 MIPS 指令系统规范来熟悉要实现 20 条指令,了解指令的特性和细节,大约看了三四遍,然后进行了指令的分类,分类标准有很多,比如是否访问寄存器,是否访问内存,是否跳转(PC 改变),然后将实现类似的指令放在一起实现,减少工作量。

熟悉指令后然后通过指令之间的相似执行过程,确定 cpu 的模块和数据通路和控制信号,顶层模块设计草图如下:



后期代码实现时对 cpu 的模块有更改,添加了与 PC 相关的模块,来控制输入指令,执行指令状态。

大体模块分为 PC，control，ALU，寄存器堆，跳转模块，数据 ram 读写等模块，使用了两个 RAM IP 核，分别做指令 RAM 和数据 RAM，还有一个乘法 IP 核实现乘法指令。模块之间的数据通路做了明确的规定和数据的特定含义,实现了通路复用。

本次实现的 20 条指令如下:

ADDU,ADDIU,SUBU,MUL,SLT,SLTU 算术运算

SLL,SRL,SRA 移位指令

AND,OR,XOR,NOR 按位逻辑运算

BEQ,BNE 分支指令

JAL,JR 跳转指令

LW,SW 访存指令

特殊的 LUI 指令

三.运行测试展示

cpu_tb.v

cpu 激励文件如上，20 条指令全部测试完毕，结果符合预期。

本单周期 cpu 设计较特殊，wea = 1'b1 时输入指令状态，PC 读入所有指令存到指令 RAM 中，然后 wea = 1'b0 后 cpu 进入执行指令阶段 PC 遂周期逐条读出执行.所以激励文件较特殊。

这次单周期 cpu 遇到了许多问题，譬如模块执行的延迟和上升沿的问题，寄存器堆数据冲突竞争的问题，都一一克服了，收获了很多，感谢学长的无私的帮助和宝贵的建议

此为最后的仿真波形图的截图：

本想下板，但学长反馈说这种板子不合适。

四. 课设总结体会

关于 cpu 设计感觉不仅需要大局感，还要从小处着手，仿佛心中有线路有数据在流动，这样才能更全面。（需要仔细和认真）。

这个顶层的数据通路要根据一条条指令特点来设计，那些通路可以复用，那些不行，那些可能冲突，设计完了就已经把每条指令执行过程中数据随时间在线路中流动的线路已经确定了。

对 verilog 这个语言“几乎没有纠错能力”深有体会。

------摘自与学长分享感悟的聊天记录

五. 课设代码文件

工程文件发布到 GitHub 上, 网址如下

https://github.com/ywlcode/singlecpuywl