数字系统

实验二— MIPS单周期微处理器设计

专业：电子信息与通信学院

班级：电信1703班

姓名：刘智超

学号：U201713314

时间：2019.4.3

指导教师：罗杰

一．实验任务

利用HDL语言，基于Xilinx FPGA nexys4实验 平台，设计一个能够执行以下MIPS指令集的 单周期类MIPS处理器，要求完成所有支持指 令的功能仿真，验证指令执行的正确性，要求 编写汇编程序将本人学号的ASCII码存入 RAM的连续内存区域

– 支持基本的内存操作如lw，sw指令

– 支持基本的算术逻辑运算如add，sub，and， or，slt，andi指令

– 支持基本的程序控制如beq，j指令

二．实验目的

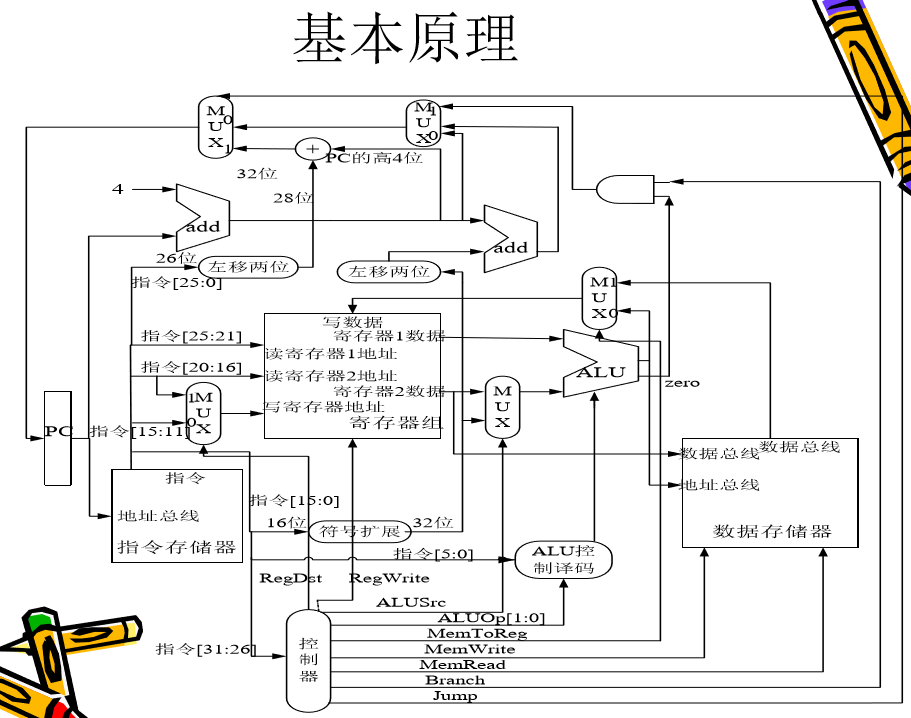
• 了解微处理器的基本结构

• 掌握哈佛结构的计算机工作原理

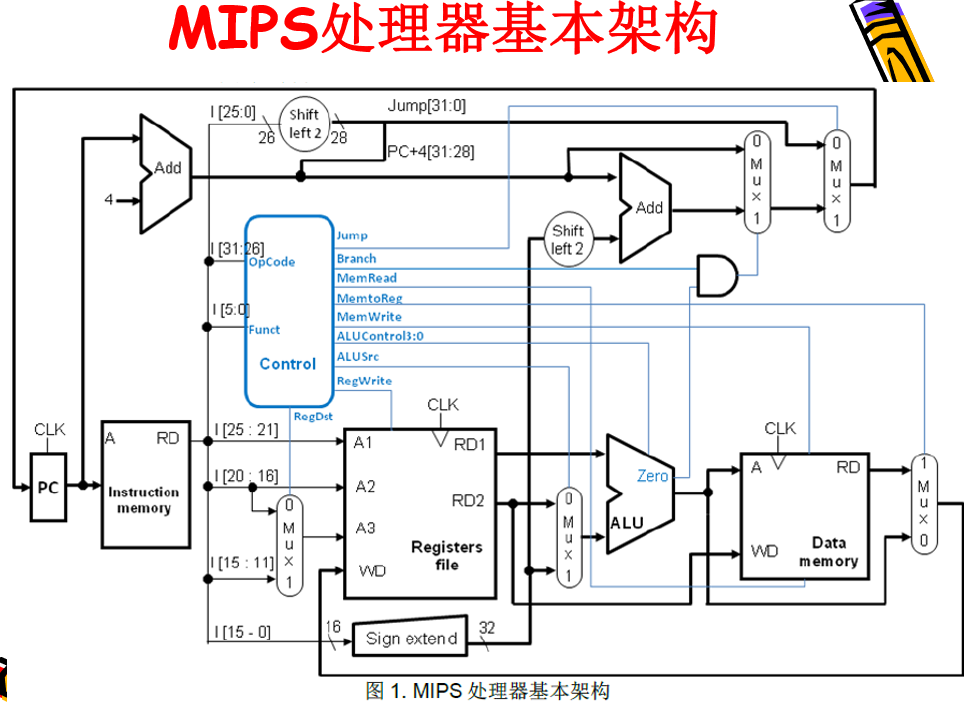
• 学会设计简单的微处理器

• 了解软件控制硬件工作的基本原理

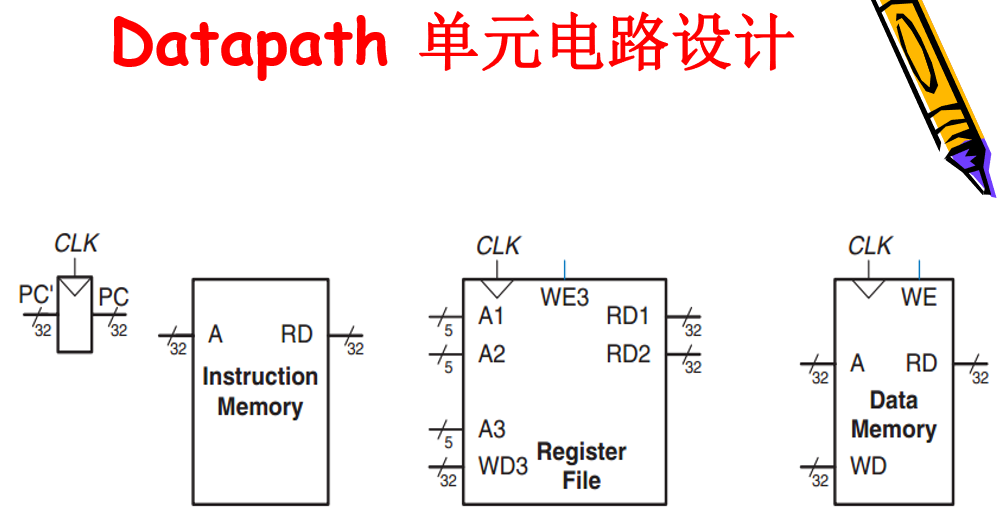
三．各模块硬件设计原理，verilog代码

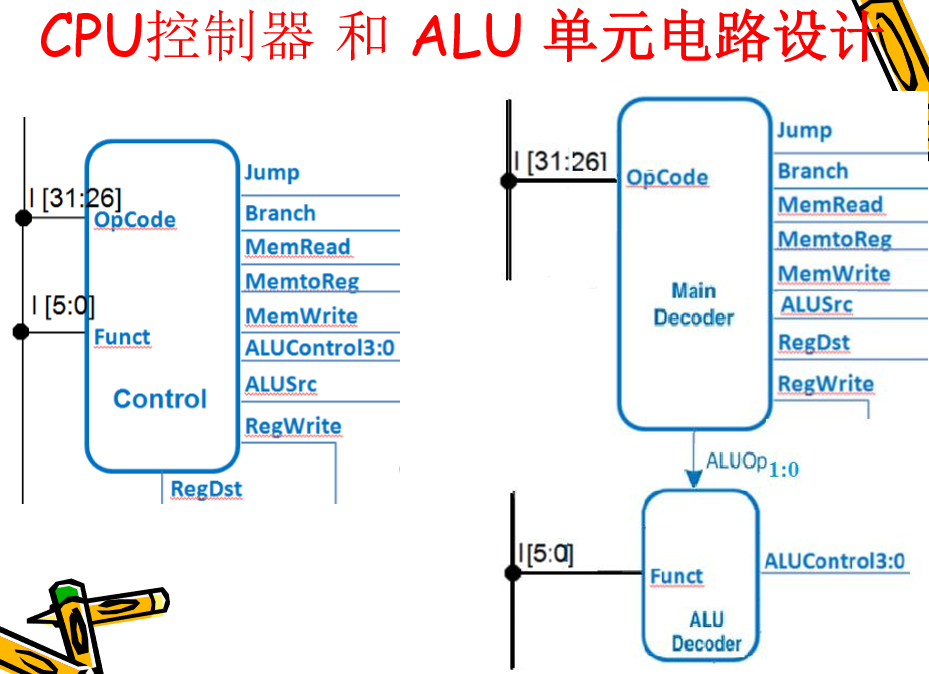


下面是MIPS处理器的基本架构

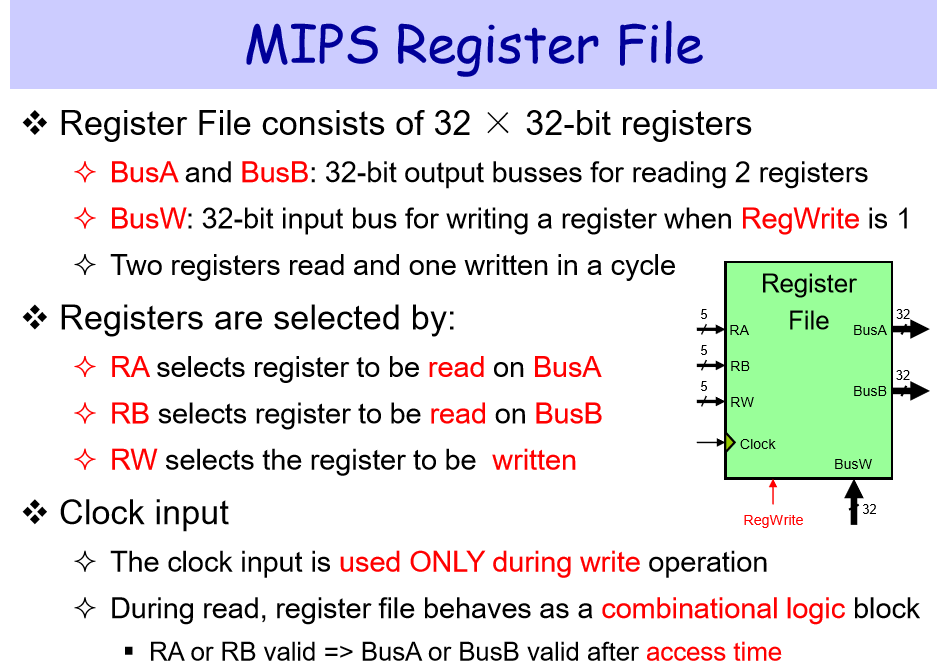


接下来看看各个模块具体设计原理与代码

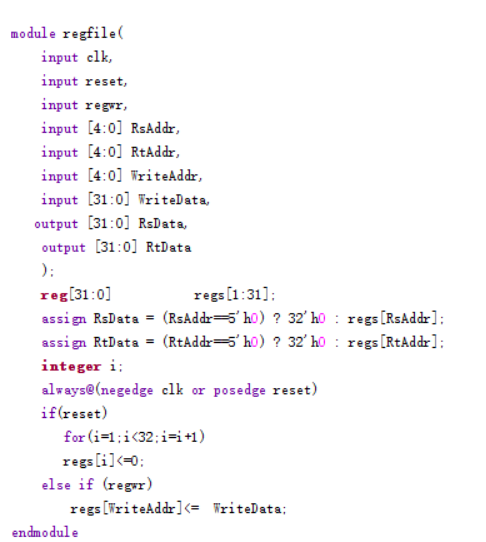




1. MIPS Register File

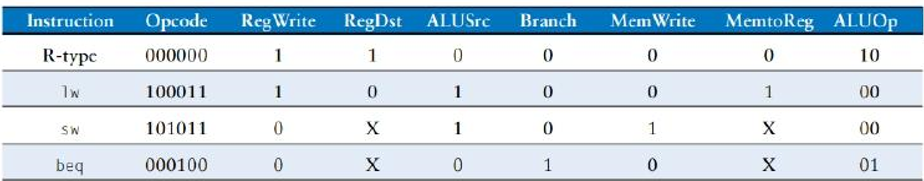


主要完成的操作是读寄存器和写寄存器，即根据指令集的32位指令来判断是读寄存器还是写寄存器，具体代码如下

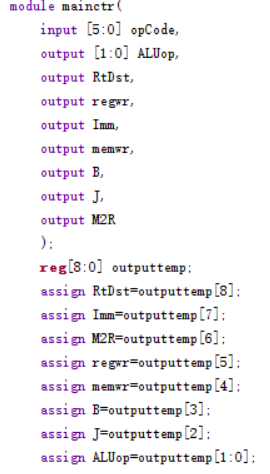
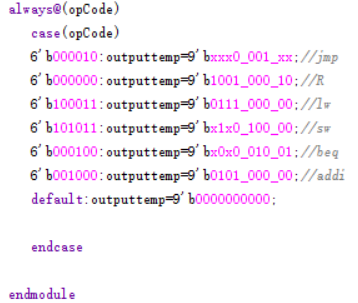


1. 主控制器mainctr模块

这个模块主要就是输入32位指令前6位的Opcpde,输出10位相应的Datapath的控制信号，相应的输入输出可根据如下真值表判断

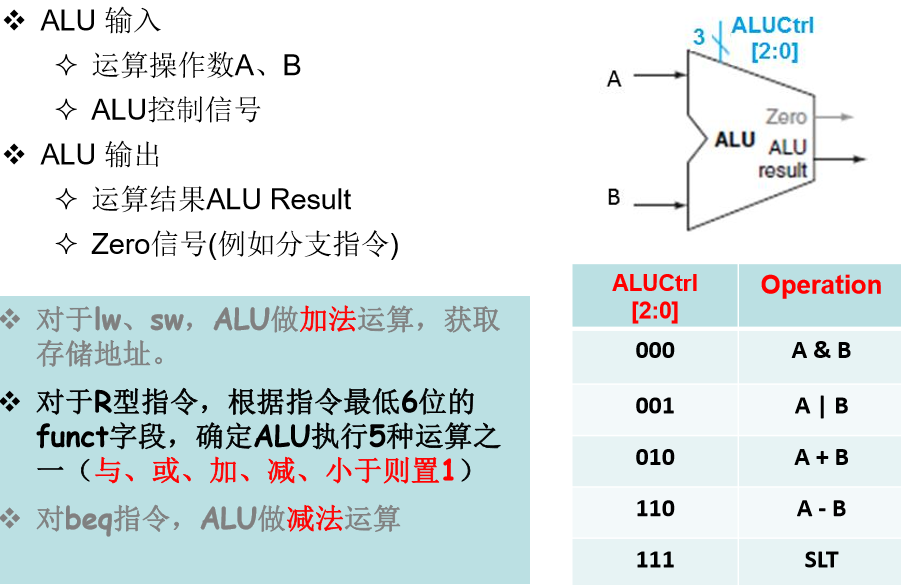


具体代码如下:

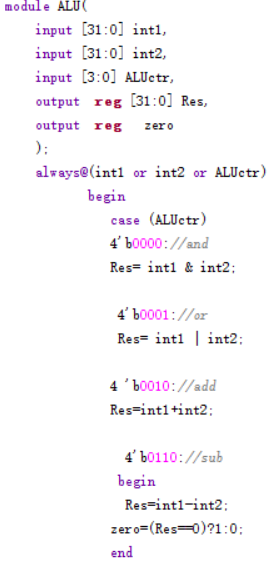
 

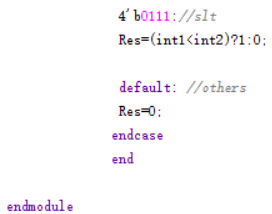
1. ALU模块

这个模块主要是完成对输入的2个运算操作数A,B进行相应的运算，并输出运算结果和Zero信号，如下图



具体代码如下:



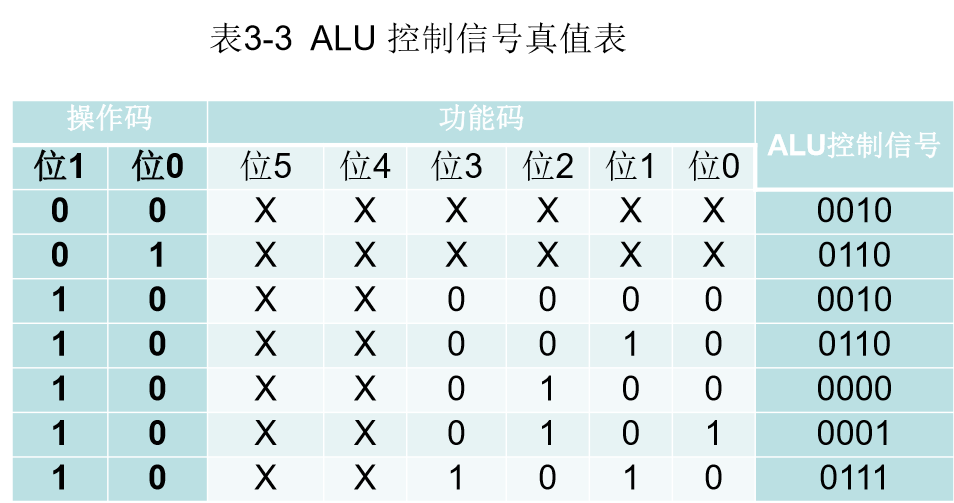


1. ALUctr模块

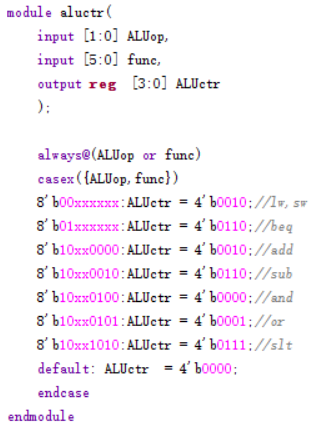
这个模块主要是根据32位指令中前6位Opcode和后面的6位funct来输出4位ALUCtrl信号，具体原理如下



相应的真值表

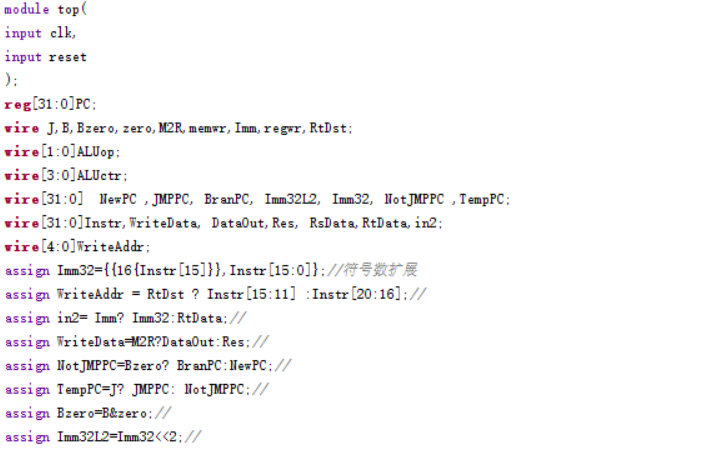


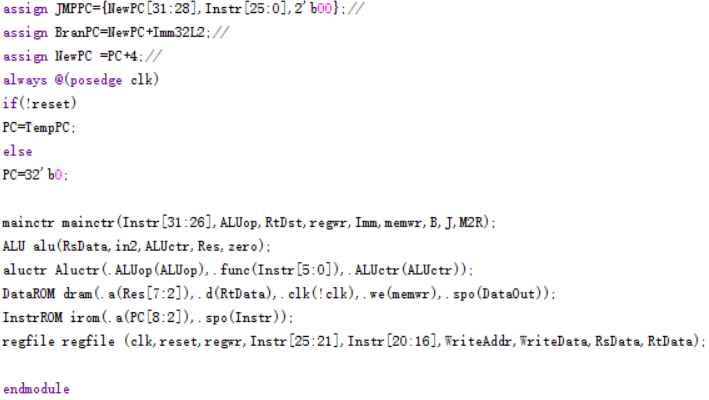
代码如下:



1. 顶层top模块

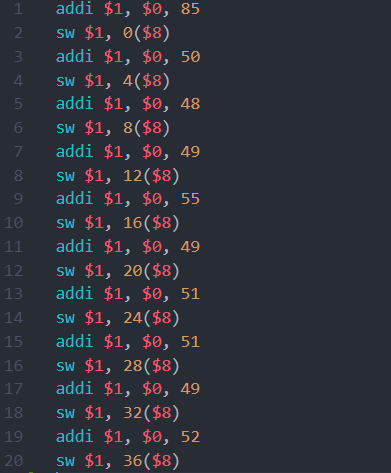
主要实现各个模块之间的连接并实例化各个模块,具体代码如下,符号扩展也包含在里面:



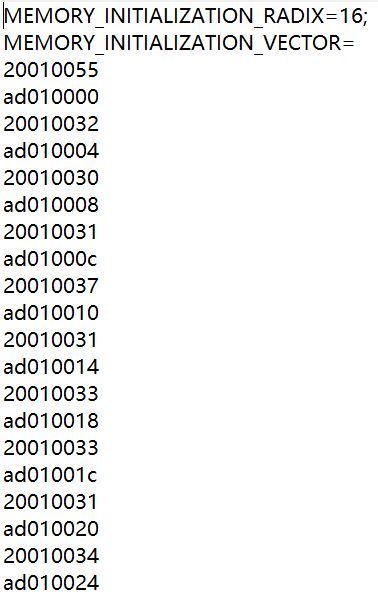


四．Rom汇编程序设计、代码

主要是运用addi和sw命令将自己的学号存在内存当中连续的一段，具体代码如下:



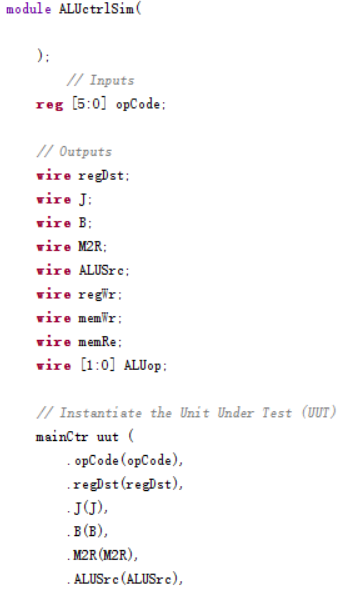
相应的coe文件如下

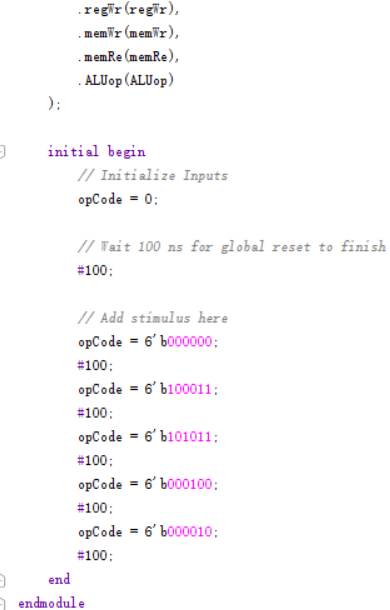


五. 各个模块的仿真激励代码、仿真结果截图

1.ALUctrl模块仿真

代码如下:



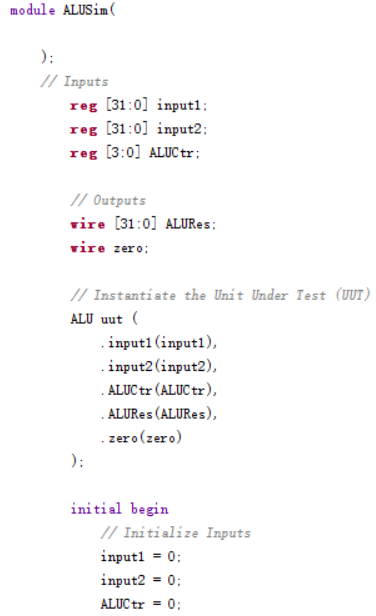


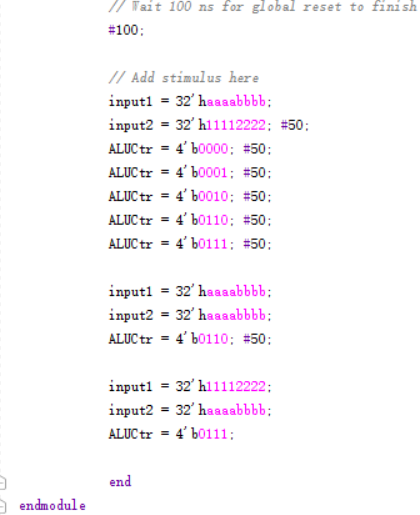
仿真结果如下:



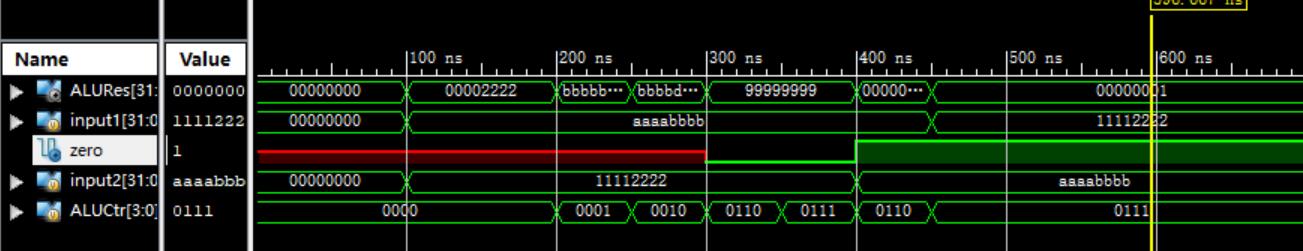
2.ALU模块仿真

代码如下:



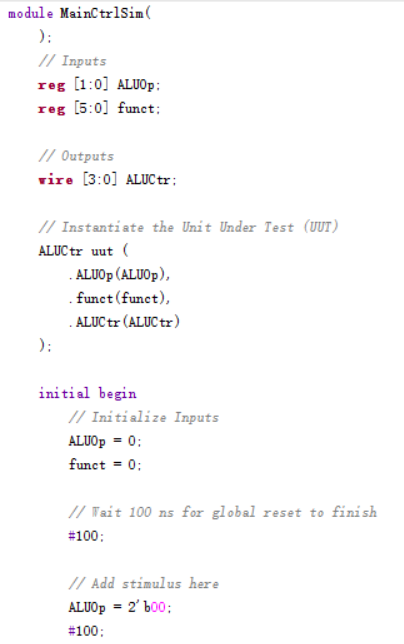


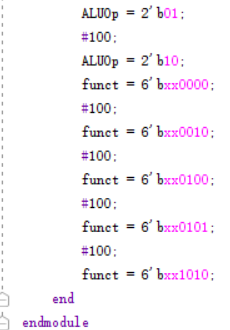
仿真结果如下:



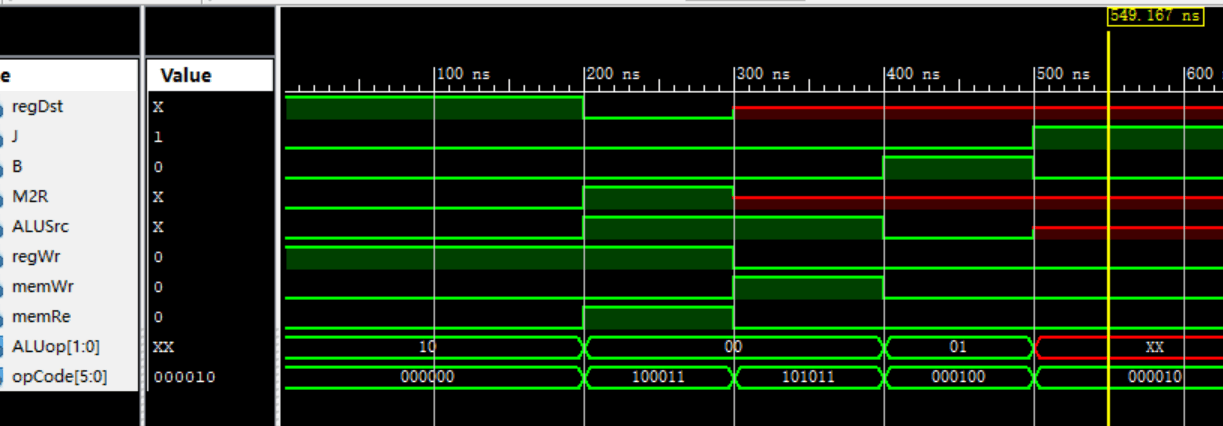
3.Mainctr仿真

代码如下:





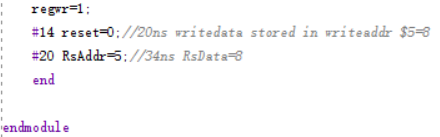
仿真结果如下:



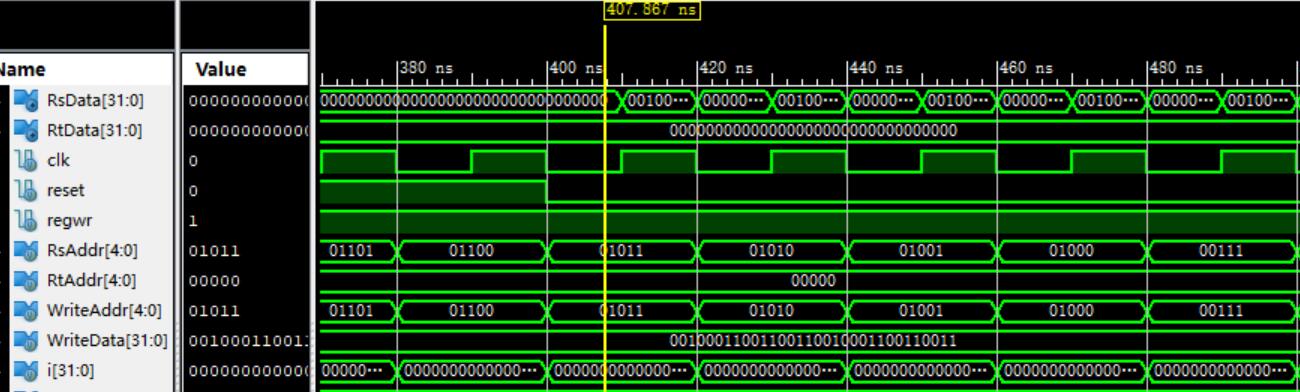
4.RegSim

代码如下:



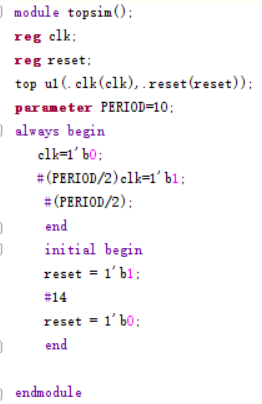


仿真结果如下:



5.top模块仿真

代码如下:



仿真结果如下:



六.心得，体会与建议

这次实验是第一次用vivado来做一个大型项目，而且运用了模块化思想，根据微机原理课上的内容，具体实践起来，虽然也遇到了不少的问题，但感觉自己确实是得到了锻炼，学会了如何用理论指导实践。这次的心得主要就是如何从零开始，一个一个地增加相应的模块来实现想要完成的功能，最终制作一个MIPS的CPU的任务。

再就是明白了仿真的重要性，很多时候自认为写的代码是正确的，逻辑上检查都没有任何问题，但总是达不到自己想要的结果，百思不得其解。只有在添加了仿真文件之后，仿真发现结果和预期不一样的时候，才会发现是代码的问题。这一点真的很重要，因为很多时候代码思路确实没问题，主要问题就出在误输入上。

还有就是对于整个CPU的各个组件更为了解了，比如在写汇编代码输入学号时，相应的机器指令到底被传到哪里去了，结合整体框图，就很清楚的直到是先存到InstrRom当中，之后一条条执行，先将要写入内存的数据存入寄存器中，再从寄存器写到DataRom,这个过程不断循环，直到把学号每一位都存入了内存。也正因为写入内存是用WriteData模块实现的，所以在仿真时，学号既可以从内存的相应单元看，也可以从WriteData模块来看，结果都是一样的。还有就是如果自己没有额外实现addi指令的话，在内存中也看不到自己的学号，因为你根本就没实现这个功能。还有就是在进行各个子模块仿真时，要先将相应仿真模块代码set as top,这样仿真时才会只显示需要的内容，否则就按默认的top模块来仿真。这就是我在这次实验中主要碰到的几个问题，踩了不少坑，也学到了不少。

最后就是觉得这种实验确实很锻炼人，一定要自己不断尝试，不断碰到问题，解决问题，这样才能学到真东西。