

# 计算机组成实验五

## 简单的类MIPS单周期处理器实现—整体调试

### 5140219355

### 朱耀明

[计算机组成实验五 1](#_Toc451247282)

[简单的类MIPS单周期处理器实现—整体调试 1](#_Toc451247283)

[5140219355 1](#_Toc451247284)

[朱耀明 1](#_Toc451247285)

[1.总述 2](#_Toc451247286)

[1.1 实验名称 2](#_Toc451247287)

[1.2 实验目的 2](#_Toc451247288)

[1.3 实验范围 2](#_Toc451247289)

[2.单周期处理器设计 4](#_Toc451247290)

[2.1工程文件配置 4](#_Toc451247291)

[2.2单周期处理器Top模块 4](#_Toc451247292)

[2.3 指令内存的实现 5](#_Toc451247293)

[2.4 控制信号的实现 6](#_Toc451247294)

[2.5 PC及INST的实现 7](#_Toc451247295)

[2.6 reset信号 7](#_Toc451247296)

[2.7 各个模块实例化 7](#_Toc451247297)

[3.仿真 9](#_Toc451247298)

[3.1MIPS测试代码 9](#_Toc451247299)

[3.2 寄存器，内存初始化 9](#_Toc451247300)

[3.3仿真激励文件编写与仿真 9](#_Toc451247301)

[4.上板实验 10](#_Toc451247302)

[4.1分频器 10](#_Toc451247303)

[4.2 寄存器与内存初始化 11](#_Toc451247304)

[4.3 LED与开关的设计 11](#_Toc451247305)

[4.4 其他部分代码的修改 12](#_Toc451247306)

[4.5 管脚约束 12](#_Toc451247307)

[5.小结 14](#_Toc451247308)

[5.1心得 14](#_Toc451247309)

[5.2实验细节 14](#_Toc451247310)

[5.3建议 14](#_Toc451247311)

# 1.总述

## 1.1 实验名称

简单的类MIPS单周期处理器实现—整体调试

## 1.2 实验目的

完成单周期的类MIPS处理器

## 1.3 实验范围

1. ISE的使用

2. Xilinx spartan3E 实验板的使用

3. 使用VerilogHDL进行逻辑设计

4. 仿真测试，下载验证

# 2.单周期处理器设计

## 2.1工程文件配置

最前面的工程文件配置等等基本与lab3,lab4相同，之后需要添加已有模块。

Device Properties中各属性填写如下

Product Category: ALL

Family: Spartan3E

Device: XC3S500E

Package: FG320

Speed: -4

Synthesis Tool: XST(VHDL/Verilog)

Simulator: ISim(VHDL/Verilog)

Preferred Language: Verilog

在lab5实现的单周期处理器中，会用到之前实验实现的各个模块。具体添加方法如下：

右键点击 Hierarchy 窗口，选择 Add Copy of Source ，添加已有的模块。 将此前两次实验中的模块文件（ \*.v）拷贝到 lab5 工程目录下。Adding Source Files… 中，选中全部要添加的文件， 保持默认项，点 OK。见Fig.1所示。

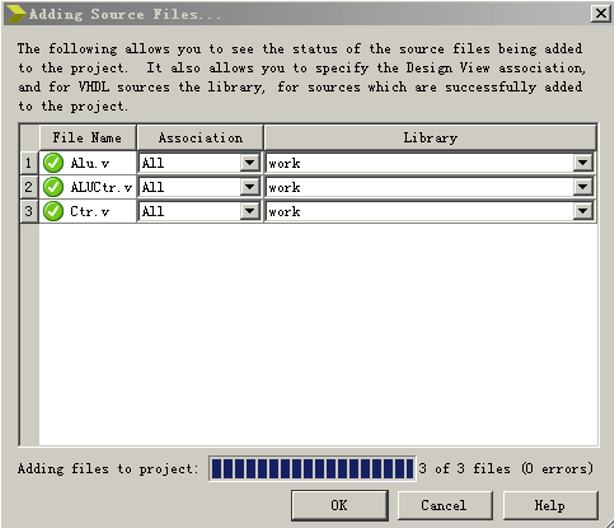


Figure 1

实验的顶层模块Top需要重新定义

## 2.2单周期处理器Top模块

2.2.1模块描述

Top模块的在MIPS处理器中设计如Fig.2

注：图上的**棕色**字是我自己添加的，这些控制信号的名称会在之后的实现代码中使用。对于实验指导书上以命名的变量，为了命名方式的统一，都改为了全字母大写，以下划线区分单词的命名方式,Zero 为了提高辨识度，改为ALU\_OUT\_ZERO

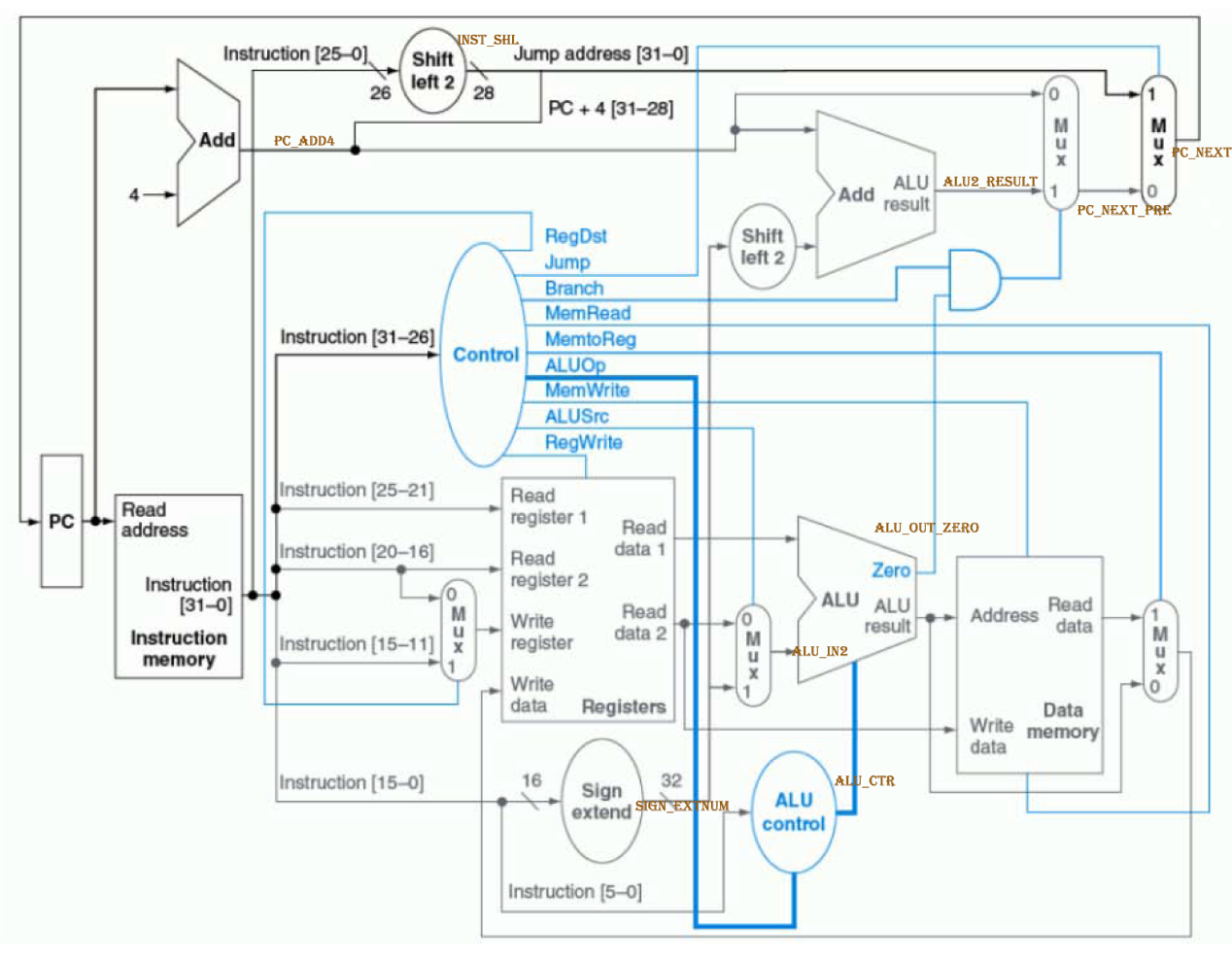


Figure 2

2.2.2 输入输出定义

在前面的实验中已经叙述过，实验实现的处理器是哈佛架构。因为指令、数据内存会单独初始化，这里的输入只用考虑时钟和重置信号。如Fig.3所示。



Figure 3

## 2.3 指令内存的实现

lab4中的data\_memory模块仅实现了数据内存。在此次实验的Top模块中，还需要对指令内存进行实现。见Fig.4

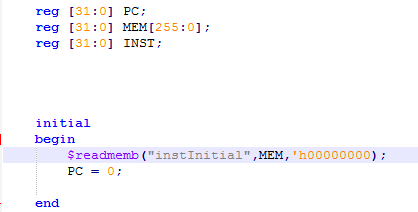


Figure 4

其中PC为程序计数器，INST用于储存当前指令。MEM储存指令。

和前面的数据内存一样，为了方便起见，指令内存MEM也是使用32bit作为单位而非常用的8bit，仅支持字对齐模式。

在前面的实验已经分析过，MEM最大可以为4GB，实验中不必实现这么多的指令，因此这里将MEM设为了255\*4B大小。

指令初始化使用了实验指导书上的$readmemb指令读取二进制文本文件。指令初始化的细节会在之后的仿真部分阐述。

## 2.4 控制信号的实现

2.4.1控制信号

控制信号参见Fig.2, 声明见Fig.5

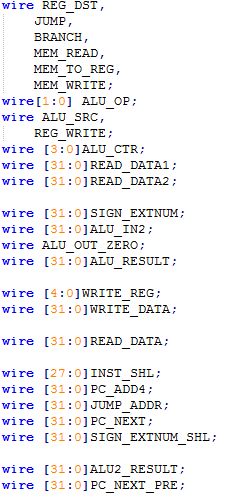


Figure 5

控制变量的实现见之后实例化的部分。

2.4.2 选择器的实现

因为Verilog语言提供了冒号问号的三目运算符，该运算符可以很好的模拟MUX，因此实验中我没有再额外撰写MUX模块。所有与MUX相关的赋值见Fig.6

使用wire类型变量与assign关键字，再加上三目运算符，实现了组合逻辑。

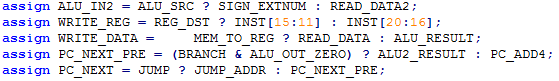


Figure 6

## 2.5 PC及INST的实现

PC寄存器与指令寄存器是MIPS处理器中非常重要的寄存器。

2.5.1 PC寄存器

PC寄存器定义为reg型变量。理论上32bit可以访问4G的指令内存，如前所述，实验不需要实现那么多的指令，但为了与理论统一，仍采用了32bit的PC，这可能会带来一些问题，具体会在之后的仿真和上板部分叙述。

在每个时钟周期的上升沿，PC寄存器会进行更新；此外，我采用了同步重置的方法设定reset：当且仅当时钟上升沿时检测到reset信号时，PC重置，见Fig.7

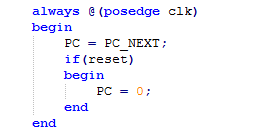


Figure 7

2.5.2 INST寄存器

INST寄存器储存当前指令。MIPS处理器的指令为定长的32bit，故INST寄存器应定义为32位的reg变量。

前面以及提到实验中是以32bit而非常用的8bit作为存储器的单位，而在实现控制信号时，为了与规范统一，仍然采用了PC\_ADD4 = PC + 4的赋值方式读取下一跳指令（非跳转时）。为了解决这样做产生的冲突，我对INST与PC的关系进行了如下处理。见Fig.8

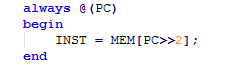


Figure 8

这样就保证了32bit的内存模式下，即使是PC每次自增4（非跳转）也能使每条指令得以执行。

## 2.6 reset信号

lab5需要我们实现reset信号重置的效果。这样需要寄存器模块也能接受reset信号并完成清零操作。这在lab4中是没有实现的。修改并不难，加入Fig.9所示部分就能解决

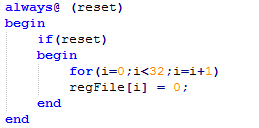


Figure 9

## 2.7 各个模块实例化

实例化前两次实验中编写的模块， 实例化的过程中需要连接模块的端口。我采用了如下的端口实例化方式：

模块 模块名(.端口 1 名(信号 1), .端口 2 名(信号 2) )

具体见Fig.10，此部分代码较多，只摘选了一部分模块。详细见工程文件

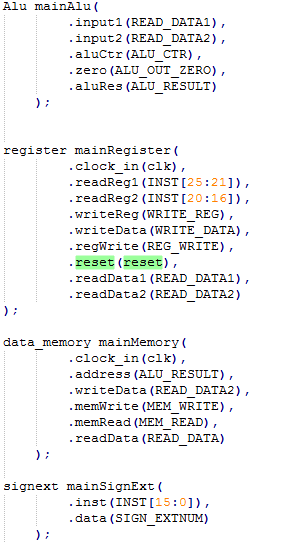


Figure 10

此外，有一部分控制变量是在Top模块中完成的，见Fig.11

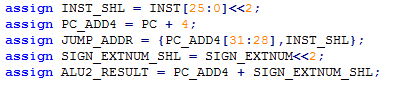


Figure 11

至此，单周期处理器的设计完成。接下来照例是仿真环节。

# 3.仿真

## 3.1MIPS测试代码

编写MIPS机器码是整套实验中最麻烦的环节之一了。起初我准备人工将MIPS指令翻译成MIPS处理器的机器码，后来发现这样太容易出错，也很费时。后来我使用了《计算机组成与设计》一书作者Patterson为MIPS指令集编写的开源IDE——MARS，使用MARS可以将MIPS指令翻译为十六进制码文本文件，如Fig.12(也是本次仿真的测试码),再使用MATLAB可以将其轻松转化为二进制码。这样节约了许多时间。

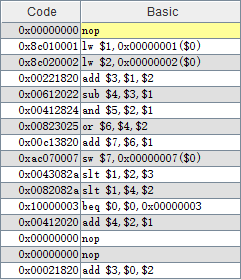


Figure 12

将得到的二进制文件放入instInitial中，与工程文件放入同一文件夹。初始化方式见Fig.13，二进制文件见附件的工程文件，这里不再赘述。

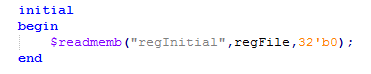


Figure 13

## 3.2 寄存器，内存初始化

寄存器与内存的初始化方法与之前的类似，使用$readmemb，在本次实验中，为了方便之后的波形观察，我将数据内存全部初始化为00000000000000000000000000000001。

## 3.3仿真激励文件编写与仿真

之前的步骤中已经完成了指令内存初始化，因此在仿真文件中除了时钟外不再需要添加其他激励。此时需要把reset也置零。如Fig.14

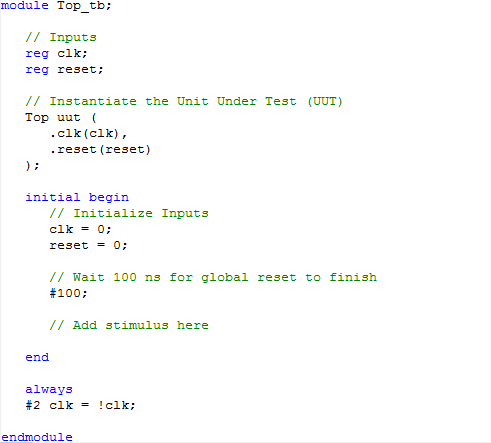


Figure 14

运行simulator behave module，得到的波形如Fig.15

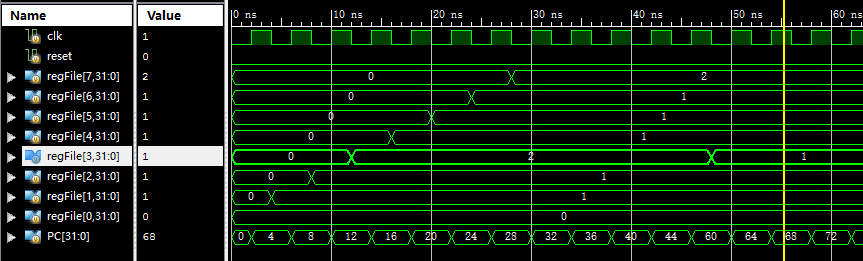


Figure 15

对照Fig.12中的指令，观察0-7号寄存器与PC寄存器的值，可见之前编写的代码实现了单周期MIPS处理器的要求。（使用的MIPS指令对内存地址为7的单元也有修改，但是若直接进行波形观察同样会出现lab4遇到的问题，解决方法在后续的实验细节会阐述）接下来是上板验证。

# 4.上板实验

## 4.1分频器

实验使用的Xilinx spartan3E 实验板是以C9管脚接受的晶振作为时钟输入源，C9晶振的频率很高，对于人工观察验证，必须要对其进行分频处理。该这次实验我直接照搬了lab2中实现的分频器。如Fig.16 所示。

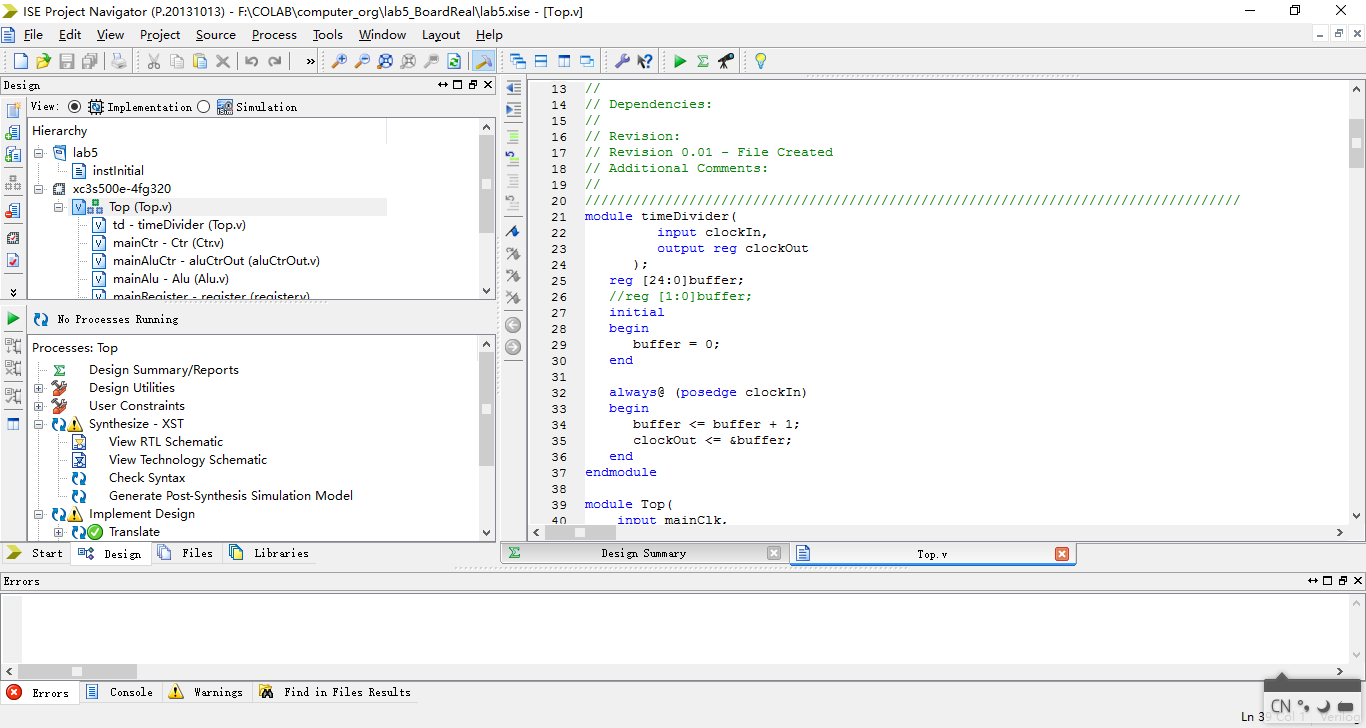


Figure 16

为了不大面积重命名变量，我将Top模块的输入时钟重命名为mainClk, 重新定义wire型变量clk，之后如Fig.17实例化分频器。这样不会影响其他的模块。

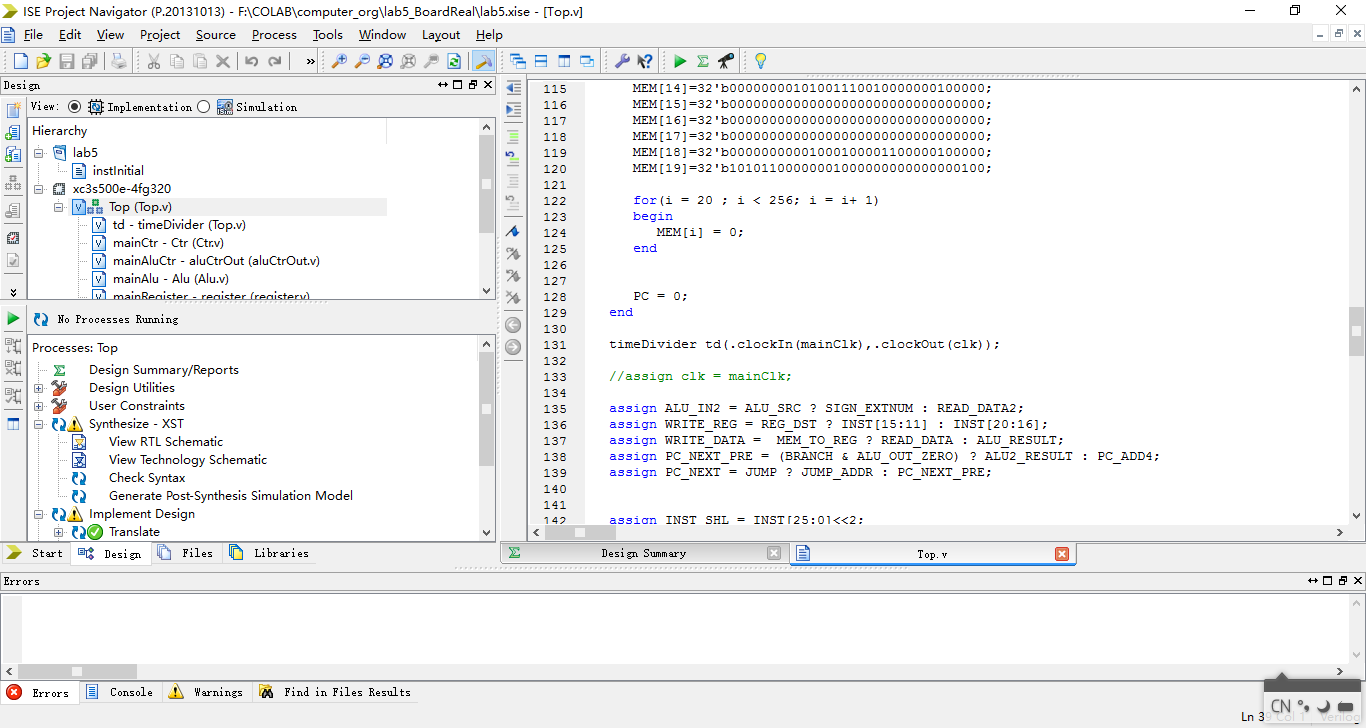


Figure 17

## 4.2 寄存器与内存初始化

因为编写的文本文件不能被直接烧录到实验板上，因此初始化不能使用之前的方法。对此我改变了赋值方法。Fig.18给出了指令内存的赋值方式。寄存器则是内存使用for语句赋值，不再赘述。

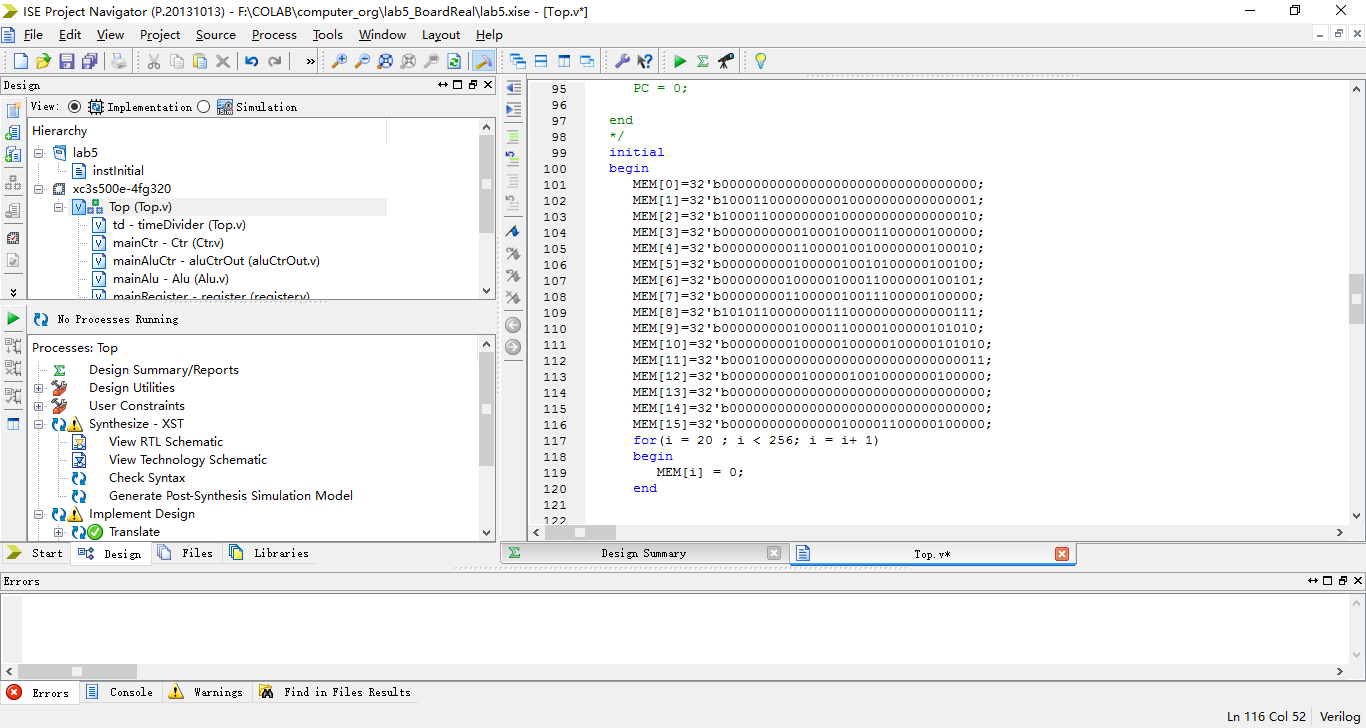


Figure 18

## 4.3 LED与开关的设计

实验板没有提供足够的硬件来观察所有的寄存器、内存情况。实验板有5 个 pushbutton， 4 个switch， 8 个 led。对此，我进行了如下的设计：

1.使用一个按动开关(H13)作为showPC，当按下时，LED显示当前PC后8bit；否则显示选中的寄存器后8bit

2.使用一个拨动开关(N17)作为reset，打开为1，重置数据

3.使用三个拨动开关(L13,L14,H18)作为寄存器选择器。可以选择0-7号一共8个寄存器，正好覆盖了自己撰写的MIPS代码中的所有被访问的寄存器。

## 4.4 其他部分代码的修改

相应地，需要对代码进行一些修改。模块输入输出与新增变量定义见Fig.19，其中LED为所选定寄存器后8bit。

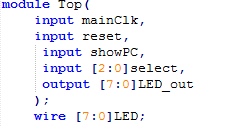


Figure 19

因为LED需要记录所选寄存器的低8bit的值，所以对register模块需要再进行修改以支持LED的输出。见Fig.20与Fig.21

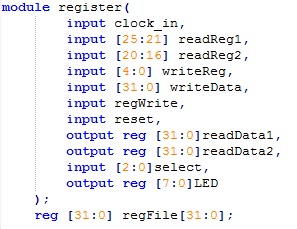


Figure 20

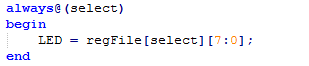


Figure 21

因为设计中使用了showPC来决定显示PC还是寄存器，因此还需添加一个MUX，如Fig.22



Figure 22

## 4.5 管脚约束

管脚的设计已在前面阐述，这里直接给出输入输出信号与具体管脚对应关系Fig.23

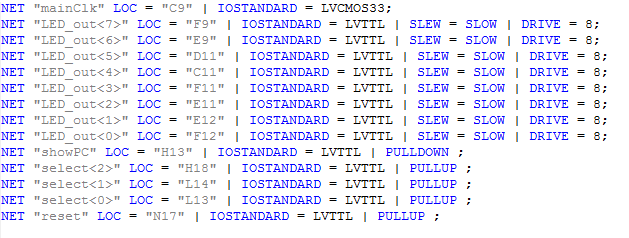


Figure 23

接下来进行烧录上板操作。这其实早在lab1,lab2就进行过了，具体操作按照实验指导书，实验报告便不再重复。

之后改变开关状态，观察LED，上课时已验证LED状态与仿真波形同步。上板步骤完成！

# 5.小结

## 5.1心得

之前的两个实验都是对具体的模块进行设计，对MIPS处理器的理解要求不高，只要掌握了Verilog语言的设计原则，还是比较简单的。这个实验则是设计MIPS单周期处理器，需要对MIPS处理器各个模块间的联系有比较深入的认识。

在这次的实验设计中，我也反复查阅了《计算机组成与设计》一书。现在我是使用硬件描述语言，按照前人已经实现过的思路来设计处理器，都非常费时；而前辈们不仅是自己规划、设计硬件与指令集，在当年硬件描述语言不太成熟的时代，还要用组合逻辑电路图来完成自己的设计，其工作量之大也是难以想象的吧。

这次实验的控制变量较多，需要命名规范。lab6中出现了更多的变量，对命名的讨论会在lab6里阐述。

## 5.2实验细节

这次实验我在上板阶段遇到过不少问题

我在上板前往往是使用ISE提供的运行simulator behave module来进行仿真调试。这次我发现上板问题后将分频器的倍数调小为2后，又运行了仿真工具，发现仿真结果是正确的，这让我非常困惑。后来与同学交流后知道了自己撰写的二进制机器码文件是不能被写入实验板的。于是我重新调整了代码，直接在.v文件里进行初始化赋值，这才解决了问题。

我最先设计上板时的想法是使用按动开关来作为reset，但实践起来发现按动后LED都不再重新点亮。具体原因我想不太清楚，于是改变了设计思路，reset使用拨动开关实现，按动开关实现showPC，这样可以实现。

还有一个细节是按动开关的管脚约束ucf文件中最后要写PULLDOWN，否则无法实现功能。

此外，我和同学交流是听说initial语句对硬件上板不能起到初始化的作用，但在我使用了该语句也能成功对实验板进行初始化。个人猜测可能是ISE版本不同造成的。

前面还提到了仿真时观察数据内存的波形还是会出现强制退出的可能性。对此我还是没有特别好的解决方案，只能直接地如Fig.24这样来观察内存。



Figure 24

## 5.3建议

因为波形仿真不足以排除所有错误(初始化、管脚约束等)，上板时可能遭遇许多难以预计的问题，建议老师可以增加实验室开放时间。

实验室可以提供类似MARS这样的MIPS开发工具。(lab6的实验指导书似乎有提到MIPS汇编器，但没有在实验室电脑上找到)