计算机组成原理课程设计实验一

多周期CPU基于硬件开发板的测试

学院：计算机学院

班级：XXX

组员：

# 实验目的

1、了解现代计算机硬件设计的基本流程和方法。

2、了解典型的RISC 处理器MIPS 的体系结构。

3、了解汇编语言到机器语言到计算机执行软硬的逻辑关系。

4、掌握处理器的设计原理和方法。

5、培养寄存器级硬件故障的检错和排错能力。

# 实验分工

## 2.1 分工

A：

B：

…

## 2.2 组长联系方式

A：189XXXX

# 实验环境

硬件环境：Xilinx

软件环境：Vivado

系统环境：Windows10

# 实验内容及原理

## 4.1 实验内容

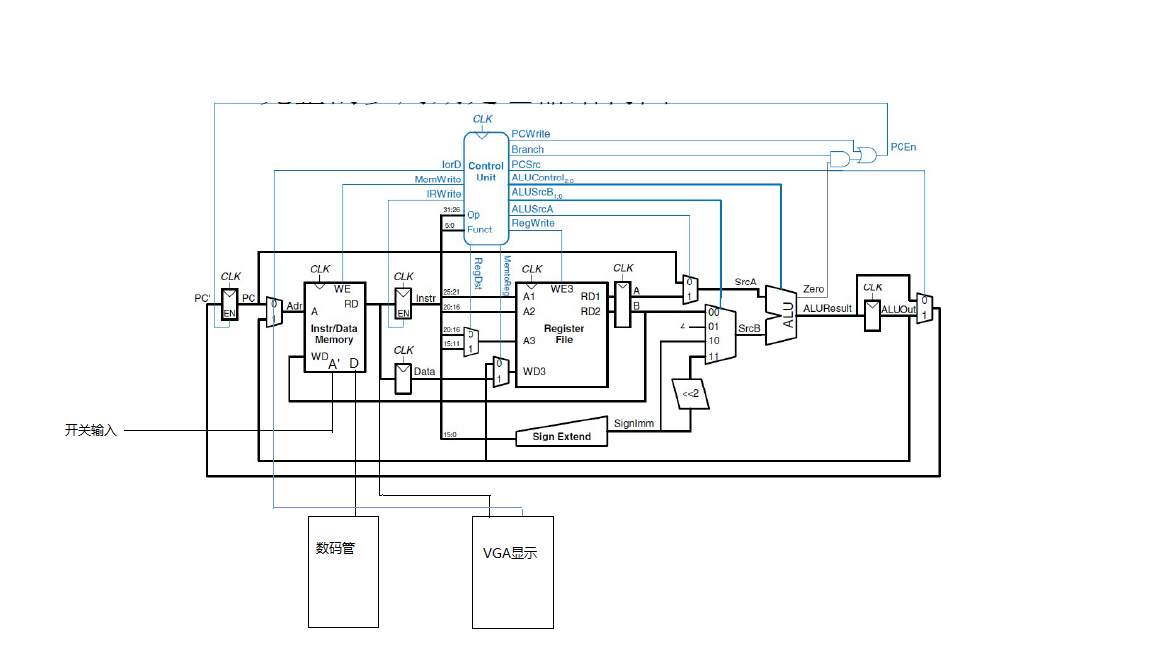
本模块的任务的主要任务在前两个模块的基础上，结合特点的开发板，整合板上接口资源，将设计下载到开发板上验证。

## 4.2 实验设计与原理

我们在实现的过程中，有关多周期CPU内容完全延续实验二中的设计，在此不赘述。

实验中，我们对当前执行的指令以及下一条的机器码在VGA上显示。同时，在数码管上对寄存器的内容进行显示，寄存器的选择由五个开关决定。因此需要修改之前的数据通路。以及模块设计。

修改后的数据通路如下：



由图可以看出，在Memory中增加了由开关输入的地址，并将数据传入数码管。并将Memory中传出的数据以及IorD传入VGA显示，以显示指令。在实际实现的时候与数据通路中稍有区别。

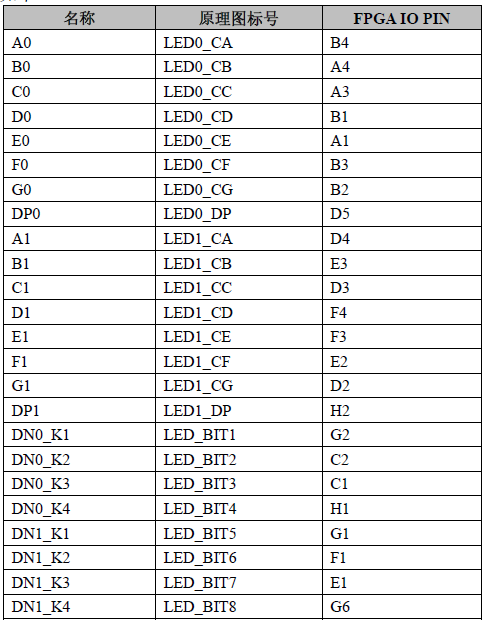
在实验中我们使用的信号配置如下图：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 信号 | 配置设备管脚 | 功能说明 |
| 输入信号 | Rst | 1个按钮(R17) | 作为系统清零，重置输入 |
| Clk | 一个按钮（R15） | 作为时钟输入信号 |
| 寄存器选择信号 | 五个开关 | 作为寄存器的地址输入，选择给定的寄存器的内容输出 |
| 输出信号 | 七段数码管 | 8个 | 显示32位数据 |
| VGA | Vga RGB | 显示正在执行的机器码及下一条机器码 |

## 4.3 数码管显示原理

在EGO1用户手册中，有对数码管的管脚使用的描述。在该开发板中，数码管为共阴极数码管，即公共极输入低电平。共阴极由三极管驱动，FPGA需要提供正向信号。同时段选端连接高电平，数码管上的对应位置才可以被点亮。因此，FPGA输出有效的片选信号和段选信号都应该是高电平。

数码管的管脚配置如下：



由管脚配置可以看出，LED\_BIT为片选，被选择的段接高电平即写1，其余写0，其对应的数码管被点亮。而LED0与LED1为片选，分别控制数码管的前4个与后4个的内容。

因此，在数码管的显示中，只需要利用循环，对段选和片选分别赋值，使得数码管依次被点亮即可。需要注意的是，数码管所使用的时钟频率不能过高，在实验中，我们使用的是1KHz的频率。

# 实验步骤

## 5.1 多周期CPU移植

由于实验三的内容是基于实验二实现的，因此，主要部分的代码都与上相同，主要修改的是RF模块以及IM\_DM模块。

对于RF模块，由于需要数码管进行显示，因此，RF模块的修改为如下：



其中，dis\_addr为输入的需要显示的寄存器的地址，dis\_data为输出的该地址的寄存器的内容。

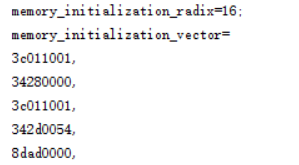
内容赋值语句修改如下：



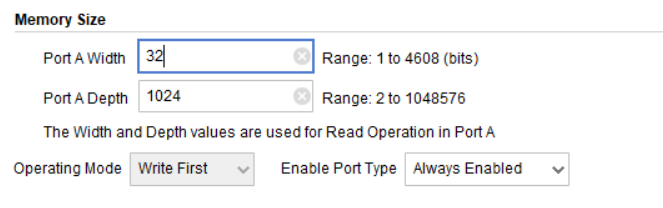
其余内容与实验二一致。

对于IM\_DM模块，由于下板的缘故，不能像在modelsim之中一样对电脑上的文件进行读写，所以需要创建一个RAM的IP，并在初始化的COE文件中写入机器码。

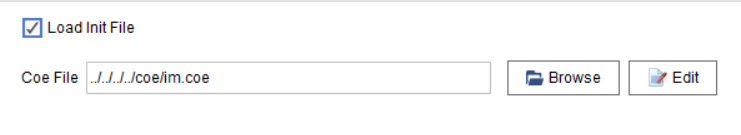
首先需要写COE文件，在实验中，我们使用的是斐波那契数列的机器码，写为COE文件如下：



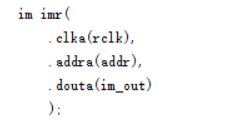
之后，创建一个RAM的IP，IP的设置如下：



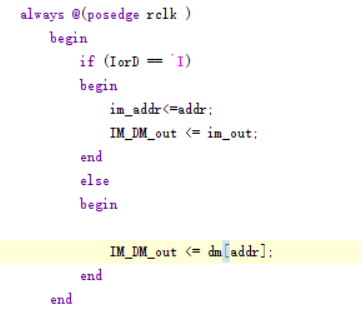
由于IM的指令是32位长，因此Width 为32位，并设置大小的1024。初始化的COE文件如下：



在IM\_DM模块中，添加IP实例化的代码如下：

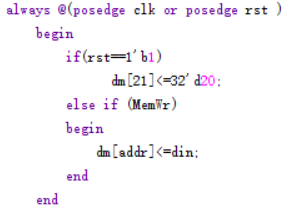


内存模块的读改为如下：



在设计中，IM\_DM的读是不由时钟控制的，但是对于RAM的读必须需要时钟，并且，在设计中，控制CPU的时钟是由用户输入的，此时钟较慢，而控制IM\_DM的读所用的时钟是板载时钟，此时钟较快，因此这样的实现依旧能够满足需求。

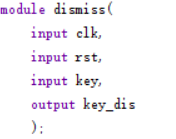
由于在斐波那契数列中的代码需要对DM进行初始化，因此，DM的读修改如下：



## 5.2 新时钟生成

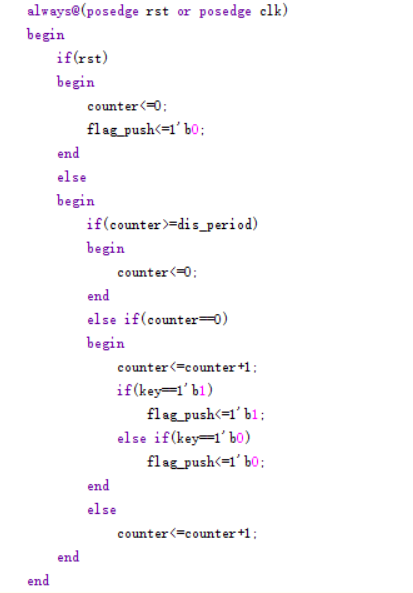
新时钟的控制模块为nclk\_gen。由于新的时钟是由用户按键输入，因此首先需要按键的消抖。

按键消抖的模块定义如下：

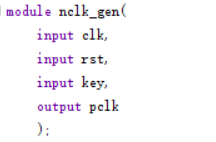


其中，clk为输入的板载时钟，rst为重置键，key为输入的按键，key\_dis为消抖过的按键输出。

实验中，我们实现的按键消抖的方式与分频类似，利用循环，在每次计数为0的时候，判断按键是否按下，若按下，则赋值为1，若没有按下，赋值为0。通过控制计数的频率，可以达到消除毛刺的效果。实现代码如下：



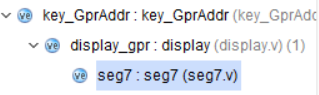
而新时钟的生成模块定义如下：



其中key为输入的按键，pclk为新时钟输出。需要在其中调用按键消抖模块，并将按键消抖的结果赋值给新时钟，具体代码较为简单，不在这里详细描述。

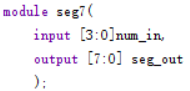
## 5.3 数码管显示

实验中，数码管用于显示所需要的寄存器的内容。所对应的文件层次如下：



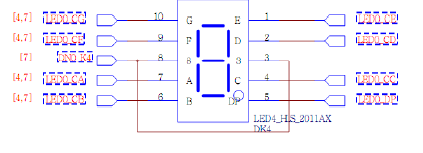
### 5.3.1 seg7

Seg7部分主要用于完成段选的工作，模块的定义如下：



其中num\_in为需要显示的数字的十六进制，seg\_out为输出的段选信号。在代码中，需要对不同的数字，对应到数码管上的每一段，有效则为1。

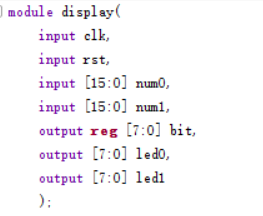
对应如下：



例如0的段选为3f，以此类推，在这里不详细给出所有代码。

### 5.3.2 display\_gpr

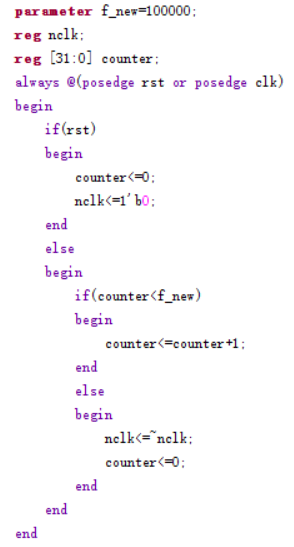
Display\_gpr用于完成片选的工作，模块的定义如下：



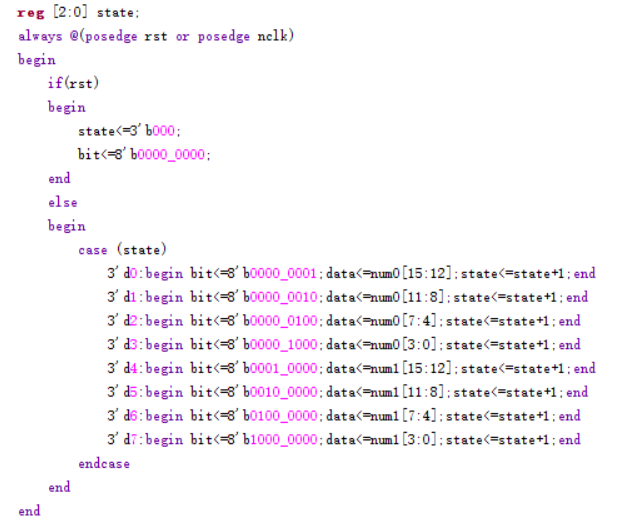
其中num0为数码管的前4位，num1为需要显示的数码管的后4位。Bit为片选，led0为前四位的段选，led1为后4位的段选。

由于数码管需要使用更低的频率，因此要对100MHz的时钟进行分频。分频的做法是通过循环计数，在计数值到达规定值时，对新的时钟进行取反。数码管使用的时钟是1KHz，因此计数值最大为100M/1K=100000。

分频的代码如下：

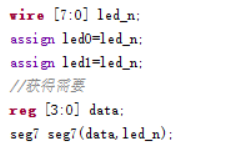


之后需要利用新的时钟对数码管进行循环显示。实验中，我们利用一个state值，使其每次加一，并利用case进行状态变换来实现循环。在每个状态，将需要显示的位置赋值给位选bit，将需要显示的数据赋值给data。代码如下：



但data是十六进制的数，因此需要利用seg7模块将其转换为段选，赋值给led0与led1。

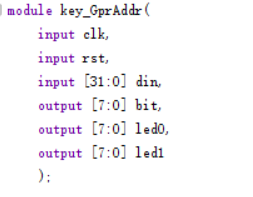
代码如下：



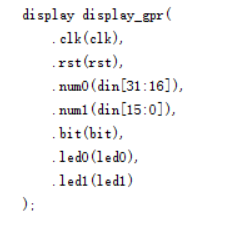
### 5.3.3 key\_GprAddr

此模块的实现十分简单，即调用display模块，显示需要的32位数据。

模块的定义如下：



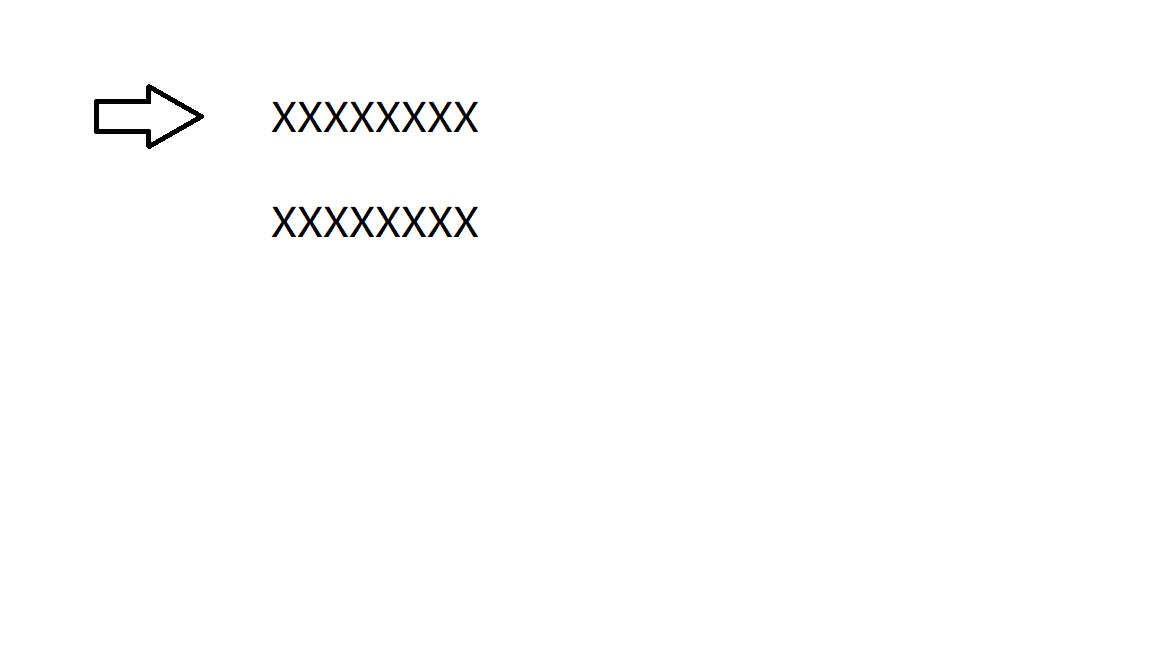
除了din为需要显示的寄存器内容外，其余与display相同。之后只需要将din的前16位赋值给num0，后16位给num1。



## 5.4 VGA显示

### 5.4.1 综述

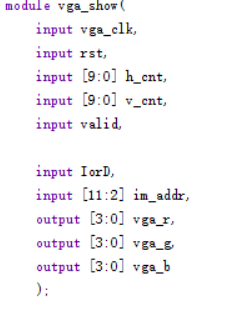
实验中VGA用于显示当前执行的指令及其下一条，并用箭头指示当前执行的指令，基本的显示如下：



VGA的显示主要由两个部分组成，一个是VGA显示器的时序控制，一个是VGA的显示部分。在老师所给的VGA样例中给出了VGA时序的代码，在此不详细描述。只给出显示部分的代码。

### 5.4.2 VGA内容显示

VGA内容显示模块由vga\_show实现，vga\_show模块定义如下：

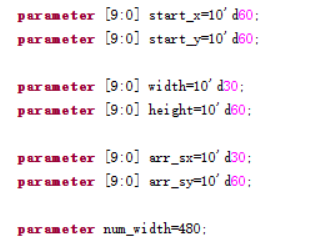


其中，vga\_clk为VGA的时钟；h\_cnt为扫描到的行上的坐标；v\_cnt为扫描到的列上的坐标；Valid为是否为屏幕上的点的标志位；IorD为数据通路中的标志位；标志输入的地址为指令地址还是数据地址；im\_addr为输入的地址；vga\_r、vga\_g、vga\_b为输出的当前点的VGA显示信息。

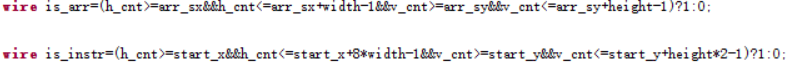
### 1、参数确定

首先要确定显示图片的大小，以及显示图片的位置。

在实验中，我们设置箭头以及数字的大小都为30\*60。指令的起始位置为（60，60），箭头的起始位置为（30，60）。因此参数的设置如下：



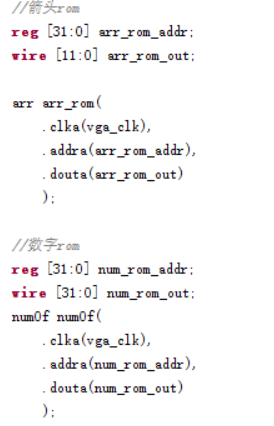
此外还需要确定显示的区域，只需要判断（h\_cnt，v\_cnt）表示的点是否在图片中即可。代码如下：



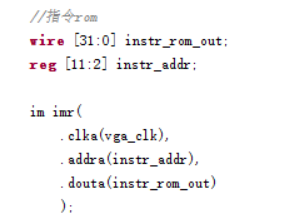
其中is\_arr表示箭头的显示区域,is\_instr表示指令的显示区域。

### 2、rom

由于需要显示数字和箭头，因此需要对其的rom核进行实例化。代码如下：



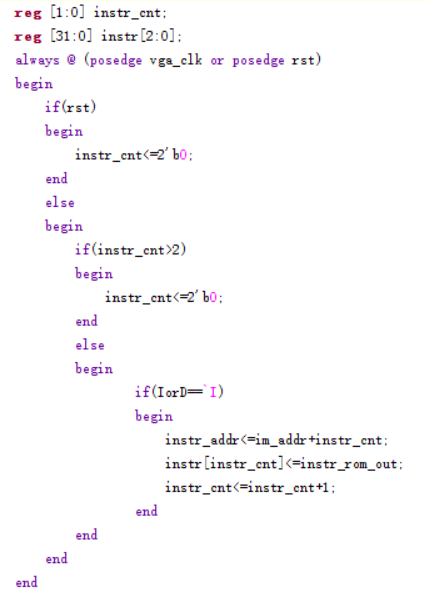
并且由于显示的为指令，因此在这个模块中，再次实例化了im的IP核。代码如下：



### 3、读取指令

该模块的输入为正在执行的指令的地址，而需要显示的是当前执行的指令及其下一条，因此循环读取指令。

代码如下：

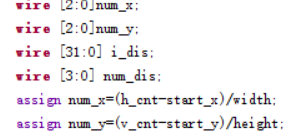


其中instr\_cnt为循环的计数器，instr为指令的寄存器。在每一个时钟到达时，并且IorD为I，即传入的地址为指令地址时，将rom地址赋值，并将结果赋值给instr即可。

### 4、确定显示字符

当确定了指令的显示位置之后，需要确定某个位置显示的字符是什么。

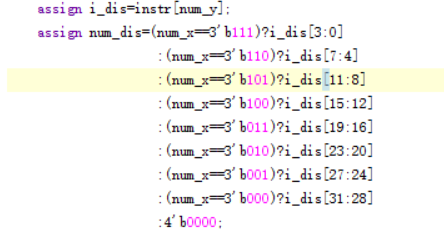
首先定义变量如下：



其中，num\_x表示显示的为第几个字符，num\_y表示显示的第几个指令，这两个变量都可以由h\_cnt以及v\_cnt确定。如下：



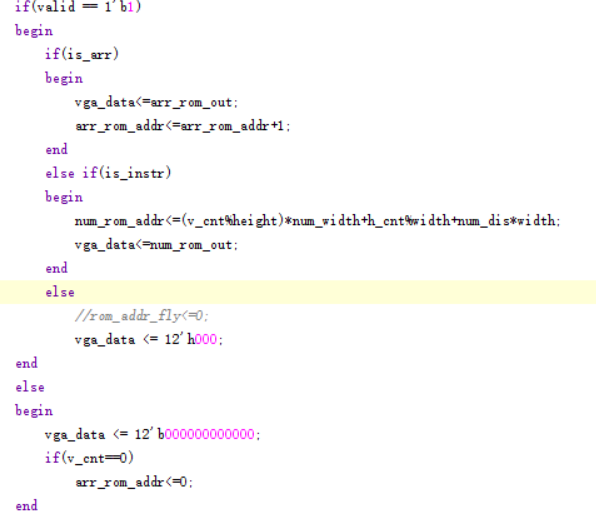
i\_dis表示当前点显示的指令内容，num\_dis为当前点显示的数字内容，这两个变量可以由num\_x以及num\_y确定。如下：



### 5、显示

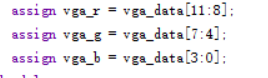
显示部分较为简单，与老师所给的样例类似，根据设定的标志位可以判断出是否应该显示，显示的是什么。当valid为1 时，表示在屏幕显示区域内，此时，当is\_arr为真时，表示显示的箭头，可以将rom读取的内容赋值给vga\_data，并将rom的地址+1。当is\_instr为真时，显示的为字符，此时应该由显示的字符以及h\_cnt以及v\_cnt确定该点在rom中的位置，并将读取的内容赋值给vga\_data。

主要代码如下：



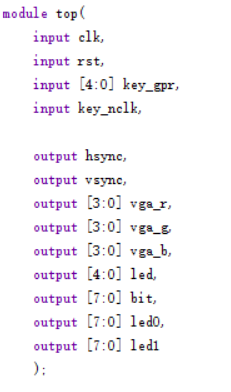
### 6、赋值RGB

目前得到的信息为vga\_data，但是VGA的管脚信息为RGB，因此需要对其进行赋值，代码如下：



## 5.5顶层文件

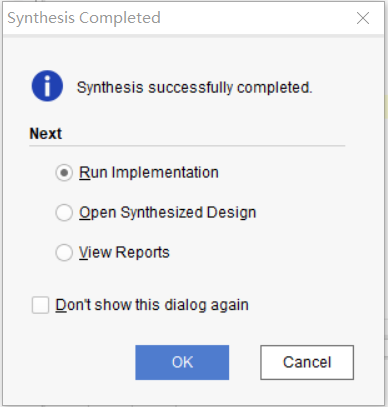
由于是在第二个实验的基础上完成的，因此顶层文件中需要添加实例化数码管显示以及VGA显示，在此不详细描述。顶层文件的定义如下：



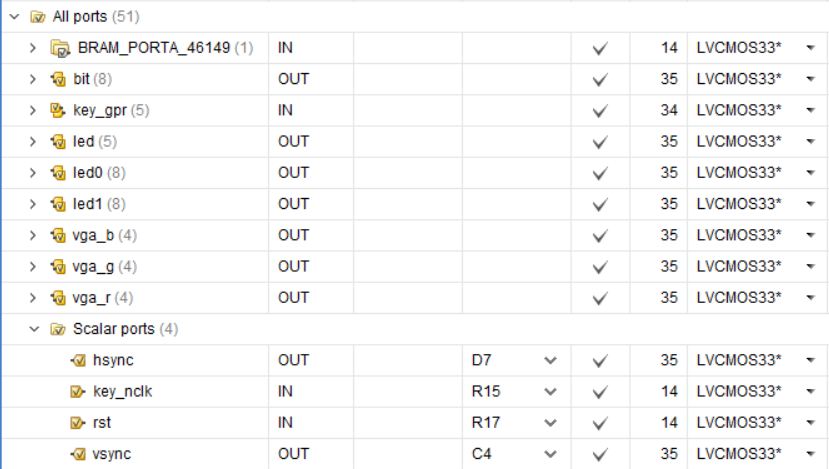
其中clk为时钟输入，rst为重置键，key\_gpr为寄存器地址输入，key\_nclk为CPU运行时钟输入，hsync、vsync为VGA的行场同步信号，vga\_r、vga\_g、vga\_b为VGA输出的RGB信号，led为寄存器地址（按键）对应的信号灯输出，bit为数码管的位选，led0与led1为数码管的片选。

## 5.6 编译运行

点击Run Synthesis进行综合，综合完毕之后弹窗如下，点击OK。



点击Open Elaborated Design，按照原理部分所给，进行管脚信息的配置。



配置完成之后点击保存，之后点击Run Implemetation、Gnerate Bitstream、Open Hardware Manager，并将代码烧入到板子中即可。

这时点击板子的按键，观察指令以及寄存器内容是否与Modelsim一致。由于我们的结果已经有视频拍摄部分，因此在此不贴出图片。

# 遇到的问题

由于实验部分比较简单，因此没有遇到什么大的问题，除了在代码中将vga\_data赋值为ROM的地址导致VGA的显示混乱之外，主要的文图存在于按键的消抖。

由于按键的消抖需要注意消抖的时间间隔，而一开始我设置的时间间隔过短，导致了很多的错误，后来将时间间隔调整至10ms后这个错误消失了。

但是实验中，我们仍然存在着很多的问题，例如没有对寄存器以及内存进行初始化，导致重置时，寄存器的显示依旧有值等等。我们尝试将寄存器的初始化加入，但是会导致寄存器的显示一直为0（不明原因），因此将其删去。

# 实验心得

### 7.1 A

这个实验还是较为简单的，实现了CPU之后，第三个实验只是将CPU进行下板的操作，由于我们在VGA中已经做过数码管的相关工作，因此代码的编写较为简单，出现的问题也不难解决，唯一感受比较心累的就是代码的编译过程，不知道为什么，代码的编译会变得越来越慢，导致整个实验很大部分的时间都花在了编译的过程中。但是实验也是非常有趣的，在modelsim上的实现与下板并不相同，下板的过程中可以使其分布执行，并看到指令以及寄存器的结果还是十分有趣的。希望以后有机会能够实现更加有趣的CPU。

### 7.2 B

第三个实验是在第二个实验的基础上进行下板，做实验三的时候我们实验二的程序还有一点点小问题，所以第三个实验过程中我主要的任务还是在调试第二个程序以确保移植成功。因为之前已经写过vga和数码管的显示代码，所以第三个实验的工作量不是很大，但是效果还是很明显的额，能够在显示器上显示当前执行的指令的机器码，在数码管上显示选择的寄存器的值。

总的来说这次实验让我对硬件语言也有了更深刻的理解，也加深了我对硬件的兴趣，希望以后有机会还能接触到硬件的开发。

### 7.3 C

在之前的处理器设计中，主要了解了设计方法以及其相关逻辑在整个设计中的功能和作用，在基础硬件开发板的测试实验中，再次熟悉了Verilog语言，在硬件版上实现，进一步验证了多周期CPU的相关内容。在熟悉硬件逻辑以及相关设计之后，这次实验更对硬件开发板有了一定的了解，懂得了如何用开发板对之前的逻辑进行测试的过程，受益匪浅。

### 7.4 D

这次实验中，我们主要是把之前的多周期处理器代码下载到板子上。除了本来的多周期cpu我们为了方便演示添加了一些vga显示。这次实验还是相对简单的，在这次实践过程中，我学习到了多周期处理器从设计到实现的全过程，收获很大，希望以后还有机会设计指令更加多样化，功能更强的cpu。

### 7.5 E

这次实验比较简单，在第二实验的基础上，把已经做好的的多周期处理器代码下载到板子上即可。为了演示的直观，我们结合了汇编实验中的VGA操作，将结果显示在了VGA中。这次实验相对简单，通过本次实验并结合之前的实验，我学习到了如何设计并实现一个多周期处理器，通过小组协作，能够细分任务并注意到许多细节，本以为下板比较复杂，通过本次试验简单了解了下板的操作和如何实践，令我受益匪浅，希望以后有机会可以再多接触硬件开发。

### 7.6 F

在小学期我们学习了计算机组成原理课程设计这门课程，它是一门综合应用计算机组成原理课程的知识和内容来解决问题并实现功能的课程，

通过学习这门课的知识并完成实验的过程，了解了现代计算机硬件设计的基本流程和方法。了解了典型的RISC 处理器MIPS 的体系结构。了解了汇编语言到机器语言到计算机执行软硬的逻辑关系。掌握了处理器的设计原理和方法。培养了寄存器级硬件故障的检错和排错能力

本模块的任务的主要任务在前两个模块的基础上，结合特点的开发板，整合板上接口资源，将设计下载到开发板上验证。为源程序增加输入输出信号，调用板上外设，完成板上验证工作。

课程知识的实用性很强，因此实验就显得非常重要，刚开始做实验的时候，由于自己的理论知识基础不好，在实验过程遇到了许多的难题，也使我感到理论知识的重要性。但是我并没有气垒，在实验中发现问题，独立思考，团队合作，最终解决问题，从而也就加深我对理论知识的理解。

# 参考文献

[1] mips汇编与数据通路.pdf

[2] 单周期与多周期处理器