姓名：epi　　实验日期： 2024 年 12 月 15 日

1、实验名称：实验五 数字系统综合设计

2、实验目的：

本实验的目的是综合运用本课程所学习的知识，设计并实现复杂的数字系统。

3、实验内容（二选一）：

(1) **实验 5.1——电子秤**

a) 简易电子秤设计；

b) UART 电子秤设计；

c) 帧解析电子秤；

d) 小数附加功能，整齐格式输出……

(2) **实验 5.2——简单处理器设计**

a) 基本简单处理器设计；

b) 乘法扩展；

c) 扩展功能——附加指令集；

d) 扩展功能——指令存储；

e) 扩展功能——UART 集成；

f) 使用 SRAM；

g) 其他附加（VGA 接口，蓝牙接口……）

实验步骤：

说明：总共完成的内容有：5.1 基础电子秤的步骤(a)和 5.2 简单 CPU 设计的(a),(b)两部分

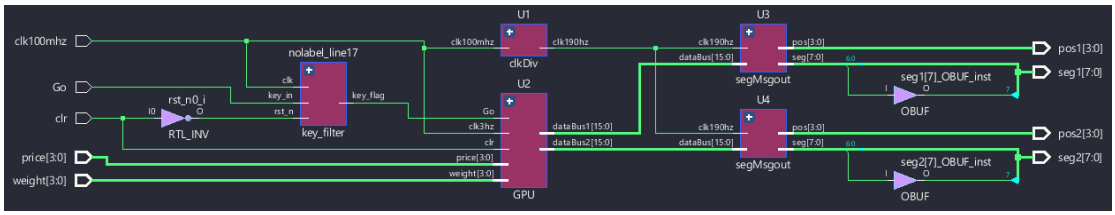
4、**实验 5.1 的实现及仿真验证**

(1) 描述你完成了哪些实验内容

完成了简易的电子秤，可以实现累计计数和单次计数的功能。

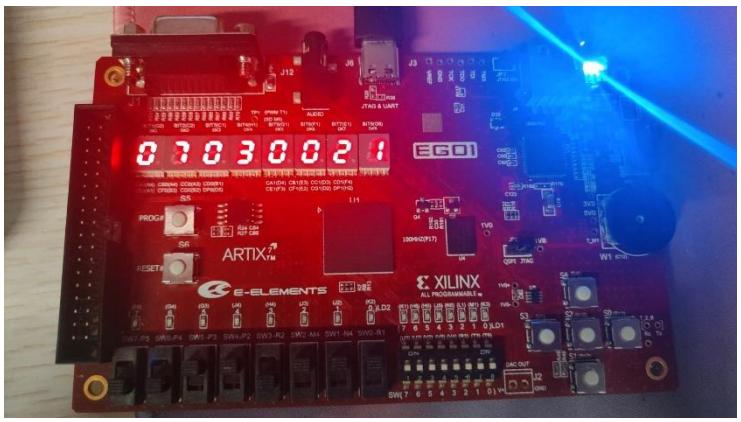
(2) 画出模块设计图，并给出解释

RTL 分析图如下



Key_filter 为消抖，clkDiv 为分频，GPU 处理数字计算，SegMsg 用于把得到的数据转化为段选与片选信号。

(3) 上板效果图，并给出解释

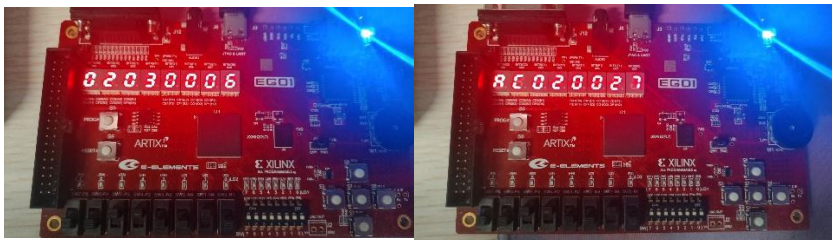


基础称重模式

基础称重模式：输入单价重量即可计算当前的总价

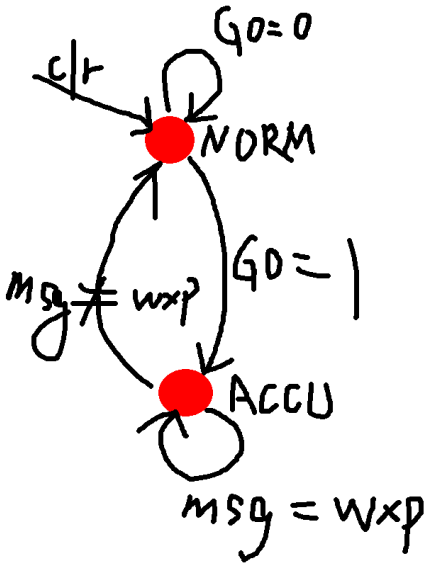
累计计价模式

累计计价模式：将当前的总价累加到寄存器中，显示当前的累计次数和累计价。



(4) 如果使用状态机完成该实验，你会设计几个状态，并给出状态转移图；如果不是使用状态机完成该实验，请详述你的设计方法

当然设计了状态机，分为 NORM 和 ACCU 两个状态。状态转移图如下



略微粗糙的状态转移图

(5) 其他的需要说明的内容

本实验比较复杂的部分是进制问题，若要始终保持十进制计算，则要同时考虑进制调整和时序搭配，代码比较繁杂，故考虑在运算时统一使用 16 进制计算，最后使用 decoder 模块实现十六进制的输出转换为十进制。Decoder 模块如下

```
module decoder_8(input [3:0] in, output [7:0] out);
    assign out[7:4] = in / 10;
    assign out[3:0] = in % 10;
endmodule

module decoder_16(input [7:0] in, output [15:0] out);
    assign out[15:12] = in / 1000;
    assign out[11:8]  = (in / 100) % 10;
    assign out[7:4]   = (in / 10) % 10;
    assign out[3:0]   = in % 10;
endmodule
```

~~虽然最后没拿到这部分的分就是了（~~

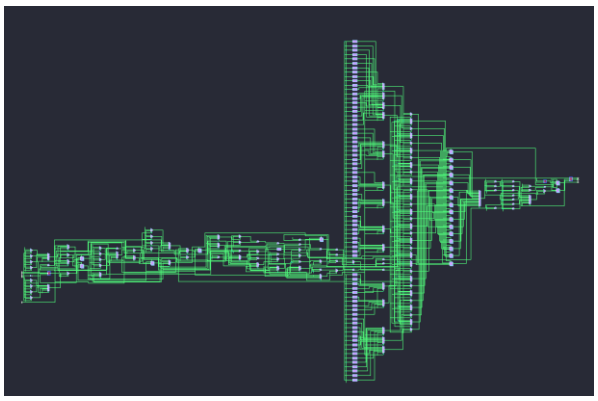
5、实验 5.2 的实现及仿真验证

(1) 描述你完成了哪些实验内容

完成了简易的电子秤，可以实现输入指令计算，将结果计入 result 寄存器，并可以使用 show Addr 读取相应的 result。

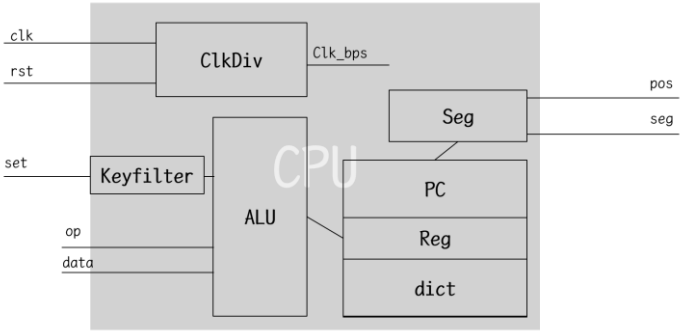
(2) 画出模块设计图，并给出解释

由于写的太赶了很多东西塞到了一起，RTL 分析图太乱了



十分粗糙的 RTL 分析图

下面是手动分析的模块图



CPU 模块设计图

关键点在于 ALU 对于相关寄存器的精确控制。

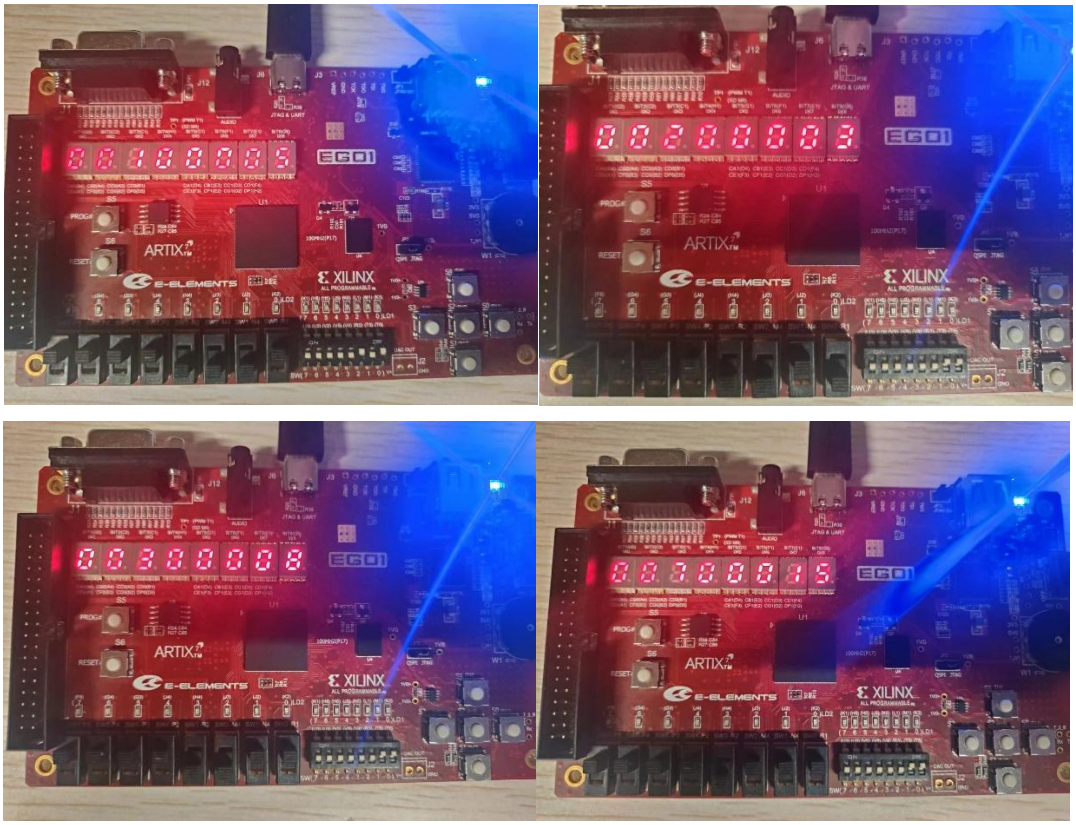
(3) 上板效果图，并给出解释

先简单设计一组用例如下

```
[op] rx ry data_bus
0000 01 00 00000101 // load  5 to r1; r1=5  show 5
0000 10 00 00000011 // load  3 to r2; r2=3  show 3
0010 01 10 00000000 // add  r1 && r2; r1=5+3=8 r2=3 show 8
<operations else>
<operations else>
0100 01 10 00000000 // mul  r1 && r2; r1=5*3=15 r2=3 mulreg=15
```

CPU 测试用例

上板图片如下（这里碍于篇幅仅显示 SHOW 指令，具体操作验收时已与助教详细展示过了）：

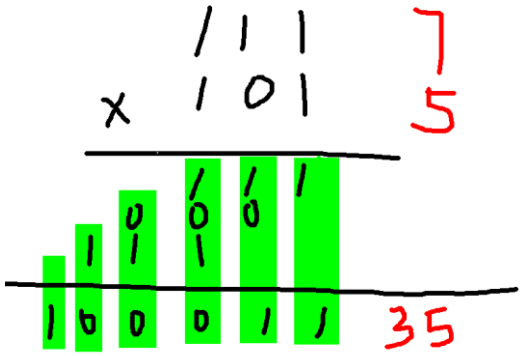


(4) 请详述 CPU 的设计方法

首先，由于是简化 CPU，所以省略去取址操作，故考虑设计指令集，控制信号的产生方式（由于指令集比较简单最后决定不单独设置模块，与 ALU 合并完成），为了减少繁琐的时序设计故设计单周期同步时序的 CPU，为了支持 SHOW 的操增加 dict 寄存器存储每次操作的值。于是划分出了 ALU，REG，SEG 三大模块，ALU 负责取数据，REG 负责数据的存储，SEG 负责数据的显示。并添加 ClkDiv，Keyfilter 等辅助模块

(5) 其他需要说明的内容

扩展中提到需要实现不是用乘法的乘法器，考虑乘法的竖式计算，不难发现乘法是若干次 n 位*1 位乘法和若干次错位加法得到的，值得注意，二进制中 n 位*1 位的乘法的结果不是 n 就是 0，这就将乘法转化成了移位和加法的操作。

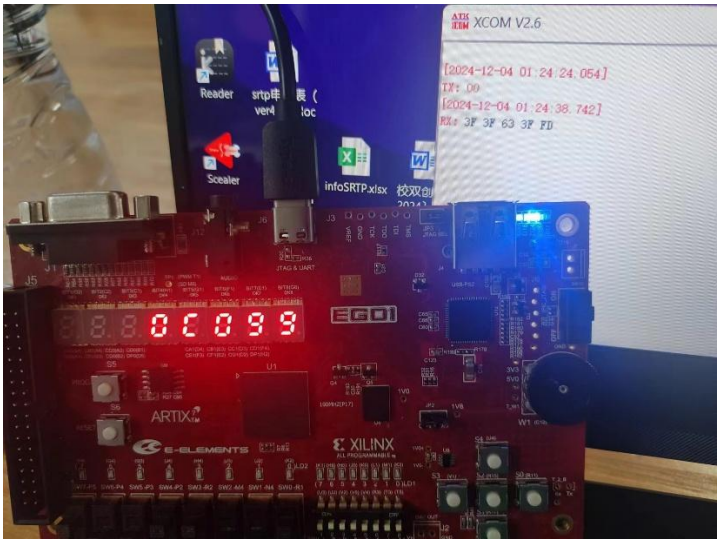


错位加法实现乘法原理图

6、实验中遇到的问题、现象及解决方法

问题 1：UART 接收和发送的数据不一致

现象：UART 接收和发送的数据不一致，会出现乱码情况，如下图所示：传 00 收到的是 C0，传 99 收到的却是 3F 3F 63 3F FD，奇偶校验，波特率，时钟频率，编码格式都检查过了没问题。



问题原因：未找到

解决方法：未找到

7、本次实验心得体会

做得很崩溃，花了很多很多时间和精力做实验也仅仅是做出了基础部分。

8、关于本次实验课程的改进建议

1. 首先，难度梯度不合理，本实验与前四次实验的难度跨度有点大，很难快速理解 UART，IP 核等知识（事实上也真的没了解明白）。

2. 其次，文档不清晰，包括数字逻辑网站，包括实验文档。（实验五的 ip 核设置的图片甚至是模糊的），这里贴一个 USTC 的同样教学内容的文档：https://soc.ustc.edu.cn/Digital/，

希望北科的文档也能好起来。

3. 再者，要求不明确，我已经听说在后面的计组（计算机组成原理）课我们还要做与数字逻辑类似的事情，这样的话数字逻辑我到底应该掌握到什么程度才算合格？并没有一个好的说明，这本应是在第一节课就该说的事情。

4. 最后，验收没能起到相应的检查是否成功理解代码并人工查重的作用。到了后面有的验收敷衍了事，甚至有了一些不太好的矛盾。这显然和初衷是不相吻合的。（不过我遇到的助教都很好，感谢助教，辛苦了！）

补充说明 1：说实话，做实验过程中确实有参考 AI 生成的代码，但是我提交过的代码绝对是自己亲手调过改过，讲的出来原理，说得出为什么的，我认为阅读 AI 的代码学习并不是一件可耻的事，相反能充分利用 AI 完成自己想实现的功能也是能力的一部分。

补充说明 2：这一份仓库是我的，本来都是在实验报告截止提交后上传的，但是忘记还有补交环节了。可能给 助教&&老师们 带来了麻烦，真的非常抱歉。

本文代码部分参考自：

https://blog.csdn.net/lsdqyz/article/details/122162873

https://blog.csdn.net/Accelerato/article/details/86546751