**东南大学电工电子实验中心**

**实 验 报 告**

**课程名称： 数字电路实验**

**第 5 次实验**

实验名称： FPGA时序逻辑设计

院 （系）：电气工程学院专 业：电气工程及其自动化

姓 名： 王皓冬 学 号： 16022627

实 验 室: 401 实验组别：

同组人员： 实验时间： 12月26日

评定成绩： 审阅教师：

**一、实验目的**

1. 综合前面所学的各项内容

2. 了解掌握数字系统设计的流程和方法

3. 掌握复杂电路连接和调试技能

**二、实验原理**

**0 输入信号与输出信号编码**

输入信号

用COL[3..0]作为输入，COL[i]代表模拟键盘第i个列信号。

输出信号

用pout[3..0]、dout[6..0]和ROW[3..0]作为输出。其中，pout[i]标识第i个七段数码管；dout[j]标识七段数码管的第j个灯管，低电平时灯亮（由seg-decoder代码推导出，在此仅提及）；ROW[k]标识模拟键盘的第k个行信号。COL与ROW共同组合为键盘功能。

其他标识说明

1. (A3 A2 A1 A0)2、(B3 B2 B1 B0)2分别代表A、B对应的四位二进制数，A代表被操作数，B代表操作数，如减法计算表达式为A - B。“1”代表该为二进制数为1。number代表数据，输出“1”代表有数据0~9输入。

2. ADD、SUB、MUL分别是addition、subtraction、multiplication的缩写，分别代表加法、减法、乘法。“1”代表选中该计算符，“0”代表未选中。operator代表计算符，输出“1”代表+、-、x号被输入。

3. equal代表等号，输出“1”时代表等号符被输入。

4. 角标i代表输入，o代表输出，分别是in、out的缩写。

5. ones-dig代表BCD码个位，tens-dig代表BCD码十位，dig是digit的缩写。

6. Y[i]代表总状态机的第i个状态，具体状态说明见主控电路State的状态转换图。“1”时代表处于该状态，“0”代表未处于。

7. CLK代表时钟信号，c0代表经转换为10kHz的时钟信号。

8. clean代表清零，输出“1”代表执行清零功能，处于“0”时不作用。CLRN代表经取反后的clean信号，是因为某些芯片清零引脚以低电平有效，为了统一clean信号高电平有效而引入的。

**实验内容：**

**1. 实验基础部分(70%) 设计一个简易计算器，它具有下列运算功能：**

(1) 一位十进制数的相加；

(2) 一位十进制数的相减；

(3) 数值和运算符用4×4键盘输入（实验室提供接口程序），其中A为“+”，B为“-”，C为“×”，E为“=”

(4) 数值用数码管以十进制形式显示，以加法为例，初始时显示全“0”，先输入被加数，再输入运算符，按下运算符键后，数码管显示全“0”，再输入加数，方法和前面一样，最后按下“=”，数码管显示运算结果

**2. 实验扩展部分(30%)**

(1) 一位十进制数的相乘，必须采用串行乘法实现；

(2) 其他自选功能

**3. 实验要求：**

(1) 根据设计要求划分设计层次、单元模块和接口信号，在预习报告上记录设计过程，绘制系统框图，每个模块的状态转移图或ASM图，并设计验证方案。

(2) 用原理图输入法设计所有单元模块并编译，分析编译时产生的错误和警告信息

(3) 对所有的单元模块进行功能仿真，并记录和分析全部仿真结果

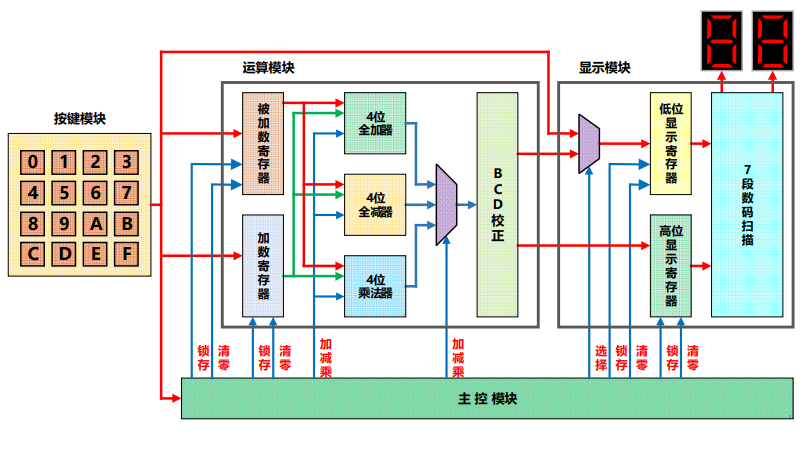
(4) 在顶层文件中连接全部单元模块并编译、综合、分配管脚和适配。

(5) 对整个系统进行时序仿真，并记录和分析仿真结果。

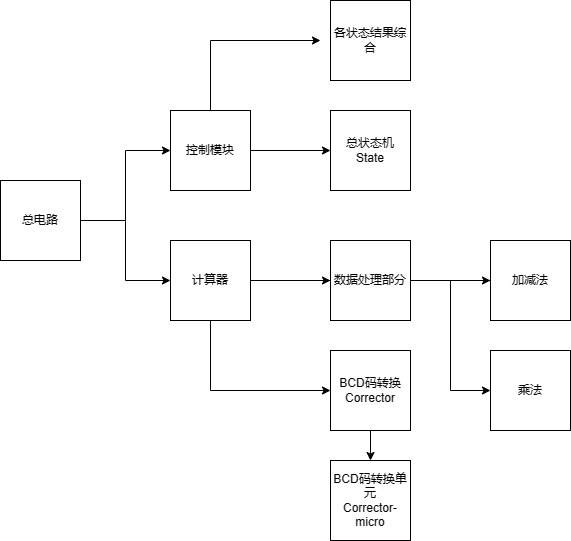
(6) 将仿真正确的设计下载到实验箱上，连接输入输出设备和示波器进行板级验证

**1. 逻辑模块**

该计算器的逻辑模块可按如下树状图表示：

****

粗略分为控制电路与数据处理电路两模块。控制电路包括了整个流程的状态机控制功能，数据处理模块包括了计算器的功能系统与校正系统。具体模块细分如下图。



在进行本实验时，采取了由下至上的设计思路，从模块单元逐一验证，最终组装为顶层电路。

**2. 模块设计与状态图**

**加减法功能：**

加减功能直接采用了实验二中设计的BCD加减器的设计，其真值表也引用了该实验实验报告的对应部分真值表。如下：

表 1 加法功能

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **输入** | | | | | | | | | | **输出** | | | | | |
| **A3** | **A2** | **A1** | **A0** | **B3** | **B2** | **B1** | **B0** | **CI** | **ASi** | **ASo** | **S4** | **S3** | **S2** | **S1** | **S0** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |

表 2 减法功能

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **输入** | | | | | | | | | | **输出** | | | | | |
| **A3** | **A2** | **A1** | **A0** | **B3** | **B2** | **B1** | **B0** | **CI** | **ASi** | **ASo** | **S4** | **S3** | **S2** | **S1** | **S0** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| … | … | … | … | … | … | … | … | … |  | … | … | … | … | … | … |

该模块无状态机。

**乘法功能（Mul4）：**

是通过串联乘法实现的。设计思路同教材。

在设计之初，采用的设计是4个7位二进制加法器串行实现，其中7位二进制加法器可以通过4位全加器得到。完成设计后反思时，我对比了自己的思路与教材思路，得到：

1. 该思路***难以实现清零功能***。虽然可以input一个clean信号，并接过一个反相器后与所有结果作与运算，使得clean输入为高电平时能将输出强行置零，但其清零时长难以控制。

2. ***无寄存功能***。这是最主要的一个区别，反思我自己的设计思路时，发现如果输入信号临时改变，会导致输出对应改变。

乘法功能，或计算器功能的***清零条件***将在下文***寄存功能***处具体分析。

**主控电路（State）：**

设计时，对于“达到条件才进入下一个状态”有多种思路：

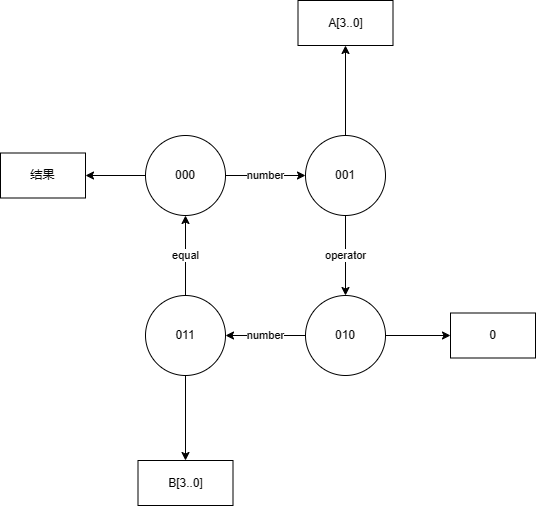
1. 可以采用74161，在使能端附加门电路组实现。由于过于复杂，未采用。由此引出一个话题：所有有转移条件的计数器，我们都可以采用在计数器***使能端附加电路***的方式实现功能。下文的思路也是基于此的。

2. 可以采用上一个实验中的One-Hot码或二进制码序列检测器的设计。同样，该设计较为复杂，故也未采用。

3. 可以利用数选器74151实现不同状态判断不同条件。记74151的各输入数据下标为i，将接入状态i对应跳转条件（由State给出），如接入number。将计数器的二进制计数反接到74151的地址端，实现选择条件的功能；再将151的正输出Y接到161的使能端，控制是否计数到下一个状态（地址），便可实现151与161的***互锁控制***。

这种方法较为简单，实际设计的时候采用的是这种方法。

对应状态转换图如图所示。



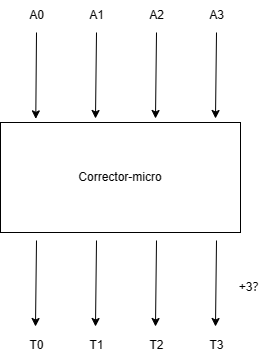
其中方框中内容为数码管应当显示的内容。

**8位二进制码转BCD功能（Corrector-micro，Corrector）：**

是利用校正单元（Corrector-micro）错位串联实现的，而校正单元采用的原理是“输入的4位二进制码大于等于5则加3”。这种方法的本质原理是，5（0101）左移一位，相当于乘2，得到的BCD码应是10（1000），但实际是1010（二进制码的10）。因此对0101加3得到1000，转换为BCD码。这其实是余三码的类似思路。

4个转换单元错位串联，即可实现移位校正。

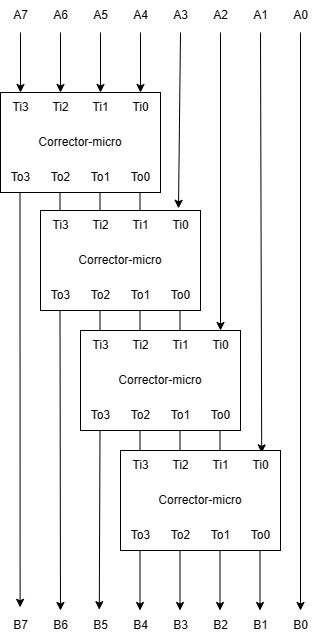
逻辑图如下。



移位是在将Corrector-micro级联时完成的。该电路不需要状态转换机，可直接得出结果。这是因为乘法模块***已有寄存功能***，该模块只需要完成将乘法的二进制输出转换为BCD码输出的功能。

而判断“大于等于5”可以通过译码的思路实现，译出的十进制数端口中的端口输出有效电平即代表输入的二进制数大于等于5。级联两个74138，得到4线-16线译码器。将输入的4位二进制数接入译码器端口，得到译码结果。将低位138的结果，以及高位138的结果作与非运算，即可判断“大于等于5”。这里没有考虑10~15的数据，因为这些输入属于***无效输入***，按键只会出现0~9的数据，不会出现这种情况。

将校正器单元错位组合为总校正器逻辑图如下。



**数码管显示：**

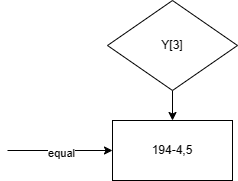
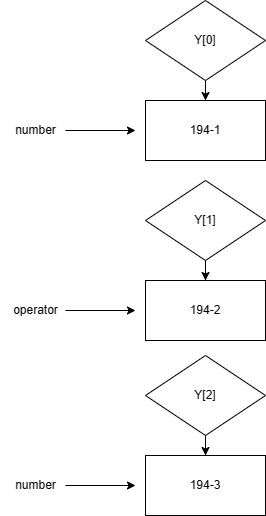
由于计算器输出仅为计算结果，而数码管应实时显示输入数A、B和结果。因此，需要设计一个模块用于在不同State状态下正确输出对应的数值。

该模块的思路很简单，只需要将对应State状态的函数Y[i]与数值作与运算，再将所有状态的结果作或运算即可。

同样，由于已有寄存功能，该模块无状态机。

**寄存功能：**

利用74194寄存。采用5个74194，分别存放A、符号、B、BCD输出。其状态转换图如下。



其中，菱形框图代表执行判断。

这里的清零功能需要特别判断。由于该模块只进行数据存储，不参与运算，我们采用假设法分析：假设没有条件清零功能，我们先后输入两个计算式：

那么第二个计算式刚输入时，计算器模块会首先计算第一个算式的数据,与第二个计算式的已输入数据，得到的结果是

这时计算器结果是错误的，输入到符号时同理。而当且仅当状态3，即已输入时，这时计算器的输出是正确的，而寄存器194-4，5中存放的是。因此，只需要在状态3时将乘法功能的数据清零，使其重新运算即可（此时的运算结果恰是正确的）。因此，寄存部分***不需要清零功能***。

该分析也补全了外电路中***计算器模块的清零条件***。

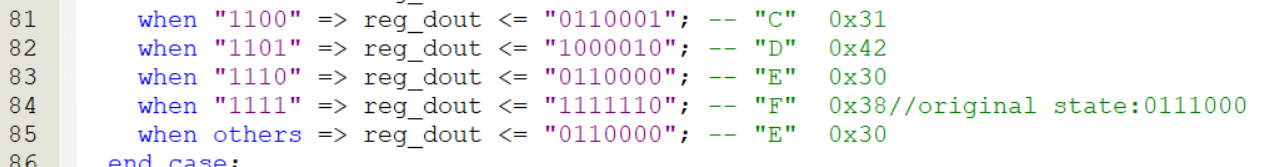
**减法“负号”的处理：**

由于实验二中设计的BCD加减器符号位的负号输出是1，这样就会得到“0 – 1 = 11”的奇怪结果。因此，负号需要特殊处理。

最开始采用了全置1的处理方式。由于减法功能结果范围为[-9 , 9]，且BCD码单位的范围为[0 , 9]，因此将BCD码的十位全置1，即1111便不会输出数值。本实验的seg\_decoder模块是根据16进制设计的，1111对应的输出是F。因此，这种设计思路下，输出的负号用F代替，“0 - 1”输出结果为“F1”。

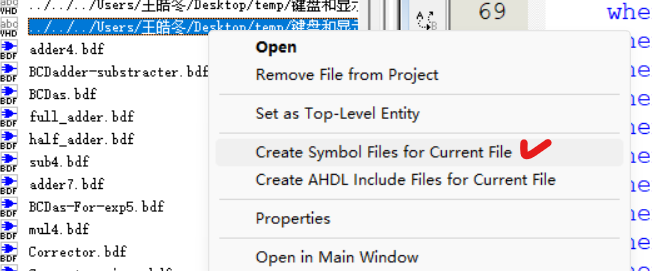
然而，这种结果仍有悖常识，人机交互性不强，仍需要特殊说明。因此，试探性对其优化，使得负数结果能直接输出负号。

实验中提供的seg\_decoder是.vhd文件，是代码。翻译代码，代码中7段译码管“1111”对应的代码是011100（F），“0000”对应的代码是“000001”，则说明“0”代表数码管灭。因此“-”对应的代码应是“111110”（0和-的数码管灯恰好是相反的）。作更改如下。



其中，origin state：0111000是为了便于改回原代码添加的注释。

通过.vhd文件创造芯片：



即可实现正确输出负号。

**3. 逻辑函数式**

由于电路主要为中规模逻辑电路组合，因此在这里仅列出部分引脚的逻辑函数式。

**总电路状态机（State）：**

计数器两使能端记为ENT与ENP，151输出为Y，则

151的地址端应有

**寄存模块：**

用于储存第i个状态信息的194两使能端记为、，则应有

**串行乘法器（Mul4）：**

储存低位数据的194数据输入端记为LOW[3..0]，其使能端记为、，输出记为P[3..0]，右移端记为SRlow；高位对应记为HIGH[3..0]与、，输出记为T[3..0]。

加法器输出记为S[3..0]与Co，两加数分别为Ai[3..0]、Bi[3..0]。

乘法结果输出为M[7..0]。

则194数据输入端应有

使能端应有

这里Y[i]代表计数器-译码器组合的第i个译码输出，Y[5]的下标5的原因将在电路原理图部分结合电路图样阐释。

整体输出应有

加法器输入应有

上两式中等号右侧逻辑表达式可对换位置。

最后，两194间的移位：

**校正器单元（Corrector-micro）：**

记4线-16线译码器的第i个译码输出为YiN，判断数据大于等于5的信号为judge，则

加法器的+3输入（0011）记为Three[3..0]，则该输入端逻辑变量为

**BCD校正器（Corrector）：**

已在设计思路部分给出了具体组合逻辑。

**BCD计算器（Calculator）：**

合并三种计算结果的思路同BCD加减器，即将功能输入与上数值再作并运算。记BCD加减器输出为SAB[7..0]，BCD乘法器输出为SM[7..0]，则个位输出有

十位由于对负号进行了特殊处理，记全高的四位二进制数为high[3..0]，有

**显示模块：**

记状态Y[i]对应显示的内容应为Disp[i][3..0]，总显示为Disp[3..0]（个位为例，十位同理），则

其中i=2是输入计算符跳转得到的状态，应显示0，因此i=2时的Disp[2][3..0]是用四根接地线组成的Zero[3..0]（0000）。

**顶层（cal-UL）：**

按各模块功能设计接线即可。其中，如设计思路中所述，乘法器（也就是封装完成的计算器）的清零端输入记为clean-2，应使其在“完成一次计算后再次输入计算式的加数时”清零。因此得到

**4. 电路原理图**

BCD加减器的电路在实验二中已经说明，且较为简易，这里不再赘述。

**总电路状态机（State）：**

**寄存模块：**

****

**串行乘法器（Mul4）：**

****

其中，由于CLK控制计数器间接控制74194使能端，与CLK直接接入74194之间存在延迟关系；且加法器的输入有赖于高位74194的输出数据变化。因而实际计数归零、开始计数变为1的瞬间，74194仍然未变化。所以上图设计时，存在两个CLK的延迟，计数器的结束位为5。

**校正器单元（Corrector-micro）：**



**8位BCD校正器（Corrector8）：**



该校正器是在Corrector-micro的基础上完成的，即校正器单元的封装芯片。

**BCD乘法器（BCDmul）：**

两模块简单顺联即可。

**BCD计算器（calculator）：**

如上。其中，为了便于扩展功能，对8位输出的高四位与低四位及符号位分别输出。

**显示模块：**



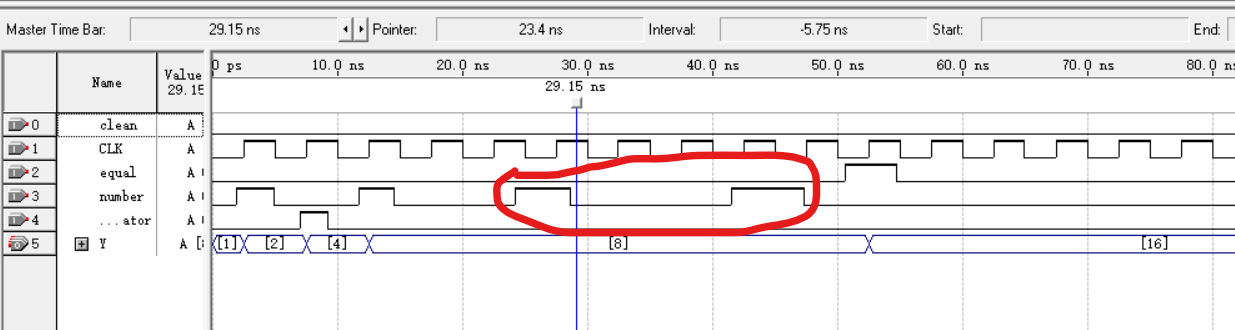
**顶层（cal-UL）：**



**5. 仿真**

部分模块，如寄存器模块过于简易，验证通过即可，不需要仿真，因此没有记录仿真。

**总电路状态机(State)：**

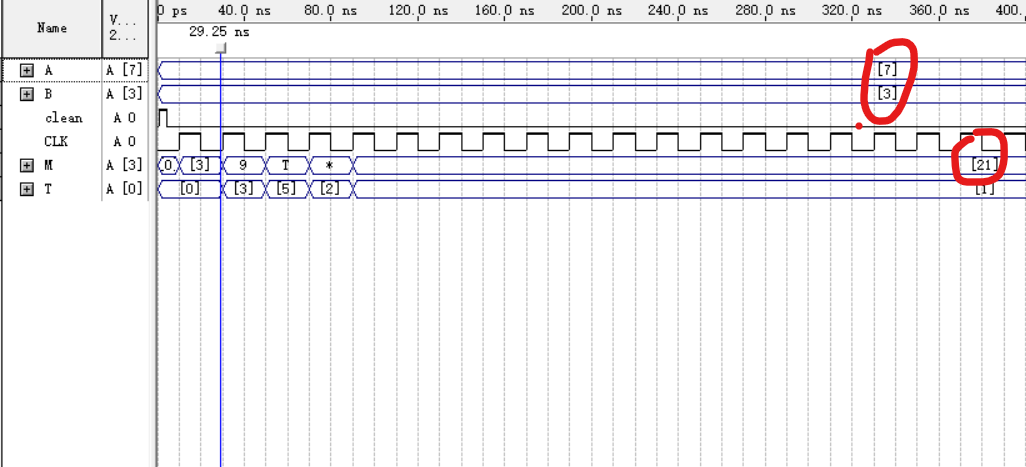
****

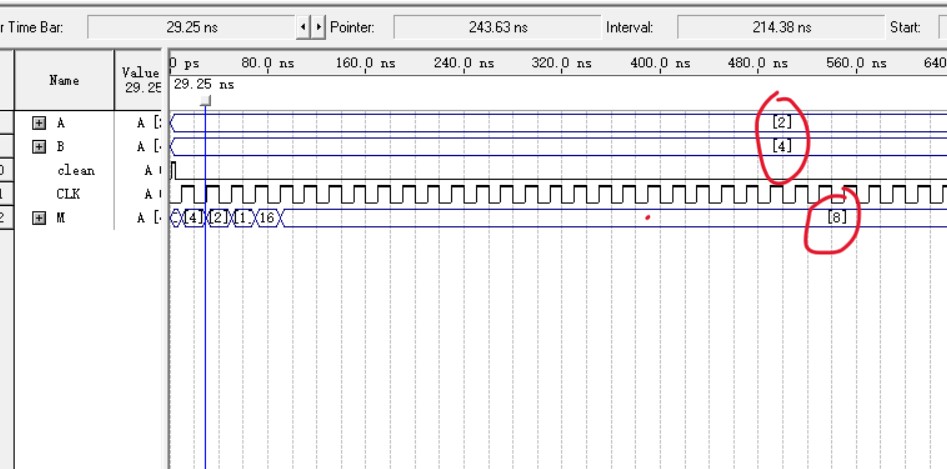
可以看到，状态机正确进行了状态转换。其中，Y初始状态为1的原因是Y是利用One-Hot码表征电路状态的，初始状态是Y[0]输出为1，Y[5..0]=00001，故初状态输出1。

红色部分为验证错误输入的部分。可以看到，应当输入等号（equal）时我们进行错误输入（这里为number），状态机不会进入下一个状态。

**串行乘法器(Mul4)：**

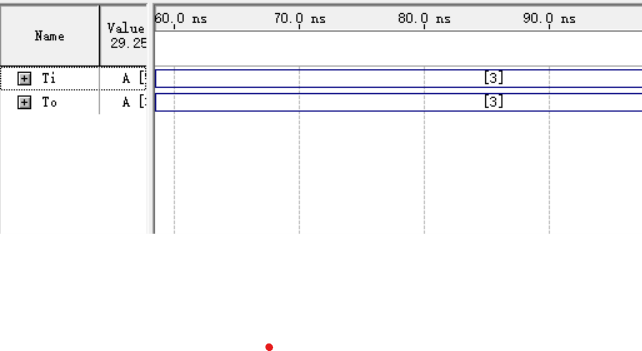
仿真所用计算式为7 x 3 = 21与2 x 4 = 8。M为输出，T为高位寄存器的输出端数值，是便于验证引入的。

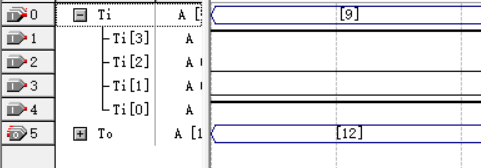


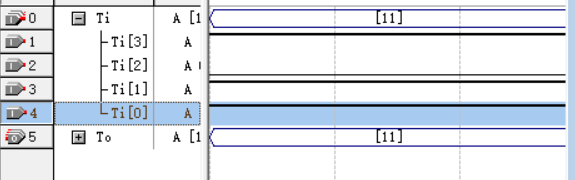


可以看到，在几次移位后，乘法器正确输出了结果的二进制表示，并能够保持。为方便验证功能，在t=0时为clean添加了一个高电平脉冲。

**校正器单元(Corrector-micro)：**



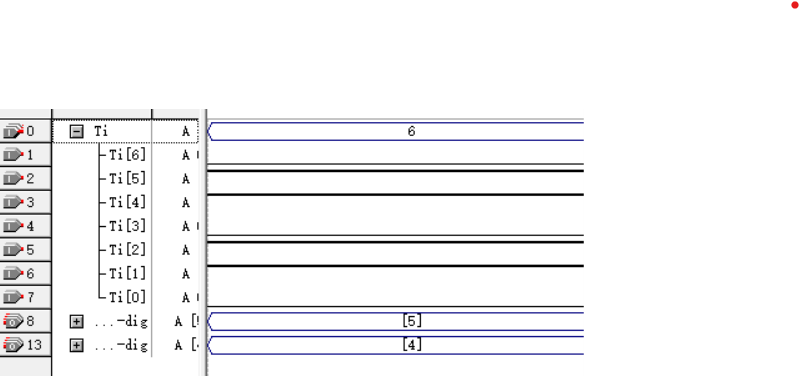




可以看到，数据为3时不校正；数据为9时校正为12；数据为11时超出校正范围，不校正。

**8位BCD校正器(Corrector8)：**

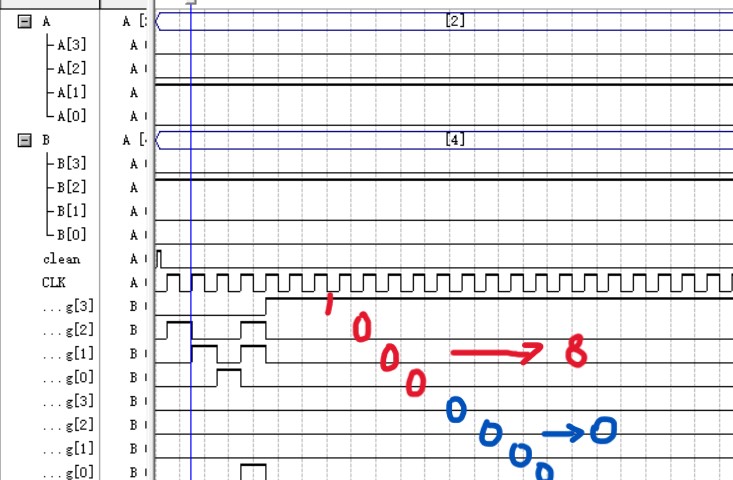
以54（0011 0110）为例。图中显示的Ti为“6”是错误的，因此截取了完整的波形表征是二进制数据54。



可以看到，十位和各位分别输出了5和4，结果正确。

**BCD乘法器(BCDmul)：**

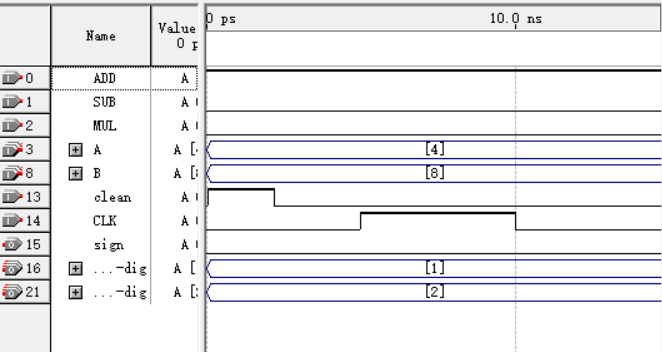
以2 x 4 = 8为例。

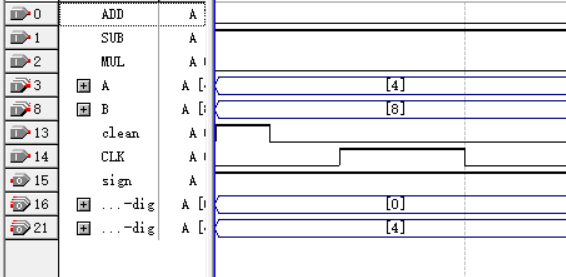


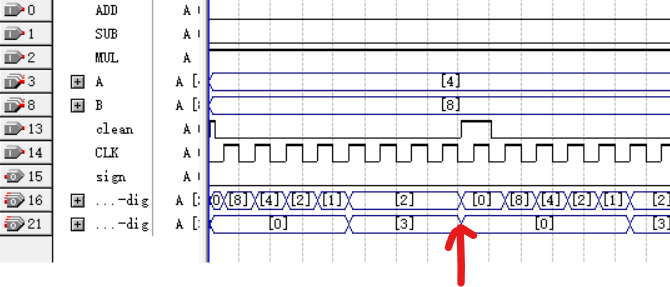
可以看到，**十位（蓝色）**与**个位（红色）**输出为0、8，正确输出了BCD码结果。

**BCD计算器(calculator)：**

以A=4、B=8为例。即计算式分别为4 + 8、4 – 8、4 x 8。从上至下分别为加法、减法、乘法。

（加法：4+8）

（减法：4-8）

（乘法：4x8）

可以看到，对于输入A=4、B=8，各功能均正确输出了BCD码表示的计算结果。其中，减法功能的sign（符号）也正确输出了“1”，即负号。

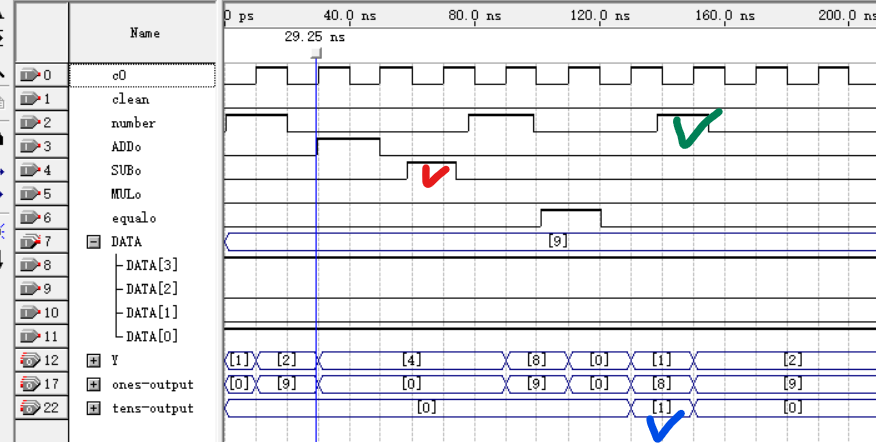
同时，对于乘法功能，若在时钟的低电平时期加入一个clean的高电平信号，观察到结果立刻被清零。这表明clean的异步清零功能有效。

**顶端（cal-UL）：**

为便于总和验证，对文件做更改如下。



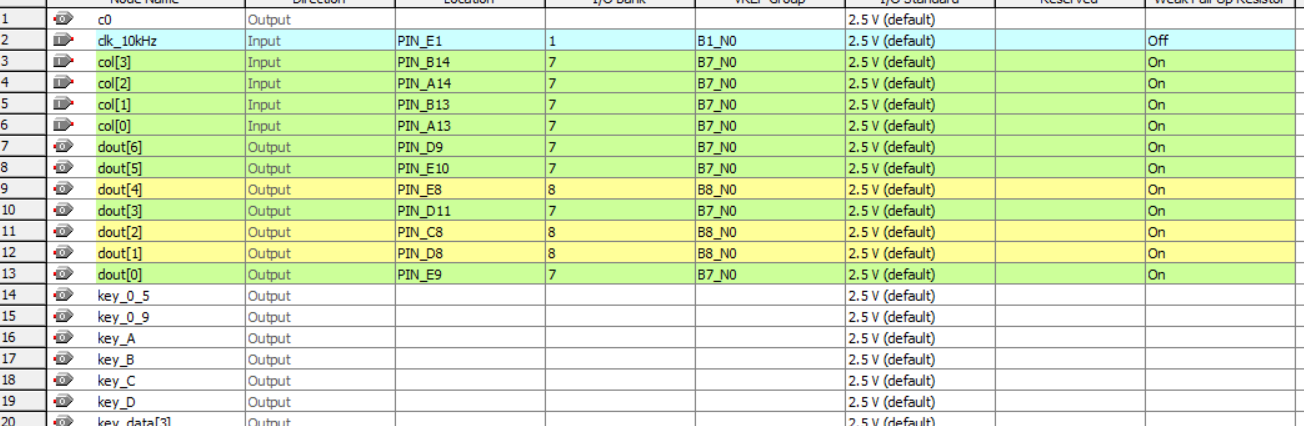
删去了用于人机交互的部分。

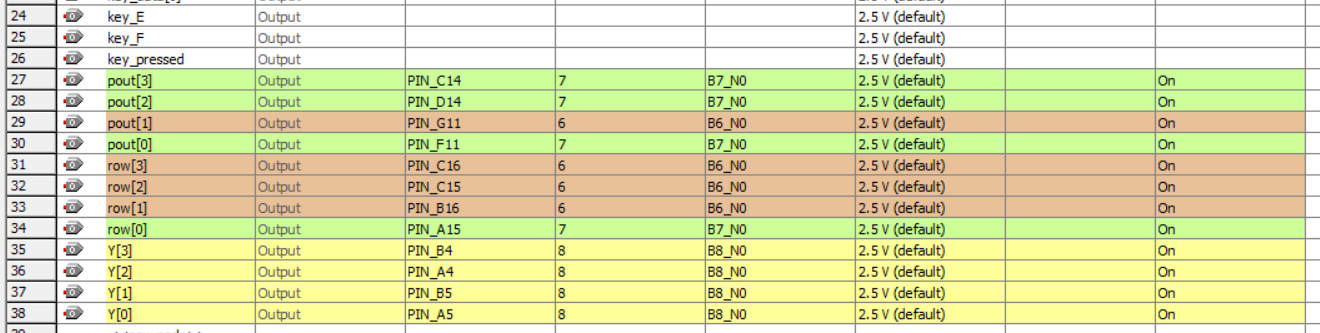
****

这里以计算式9 + 9为例。可以看到，在依次输入number信号后，正确实现了状态转换。如果错误输入了信号（红色勾部分），则状态不会跳转。最后，正确输出了计算结果18（蓝色勾部分）。计算完成后，若再次给进number信号，会重新开始计算（绿色勾部分）。

各分部计算式已在上文验证，因此在此仅验证总和功能无误即可。

**6. 管脚分配**





**三、实验记录**（记录实验具体步骤、原始数据、实验过程、实验中遇到的故障现象、排除故障的过

程和方法等）

**实验步骤：**将程序下载到FPGA板，并连接键盘。

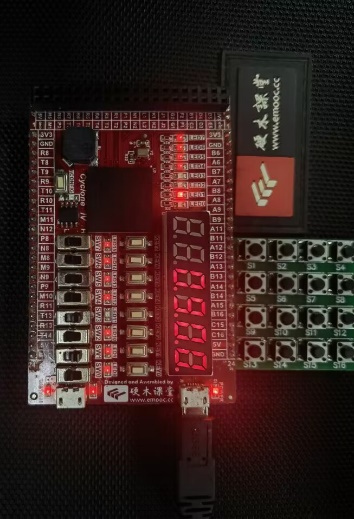
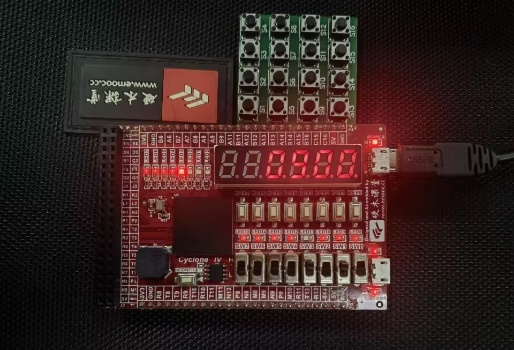
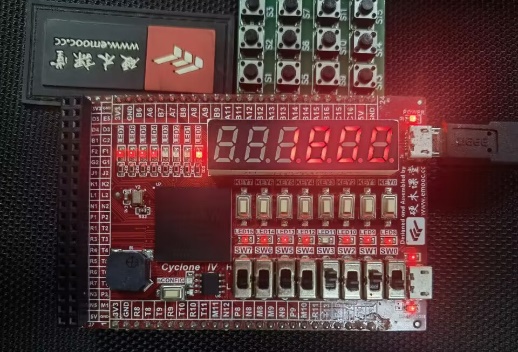
1.计算功能：分别输入加法、减法、乘法计算式，每种功能分别验证是否需要进位的两个计算式。观察结果是否正确。

2.错误输入：在应当输入数字时输入符号或等号，观察是否会扰乱输入顺序。

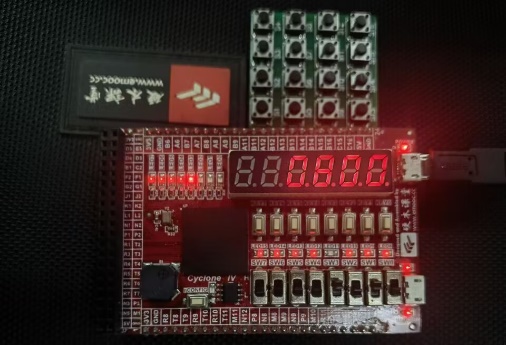
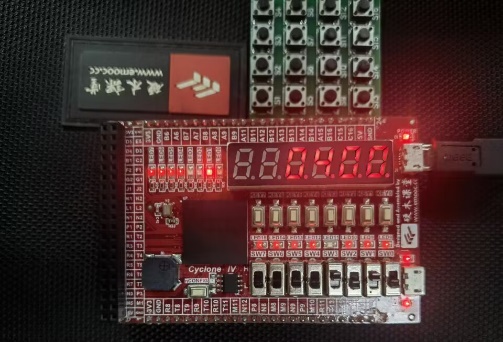
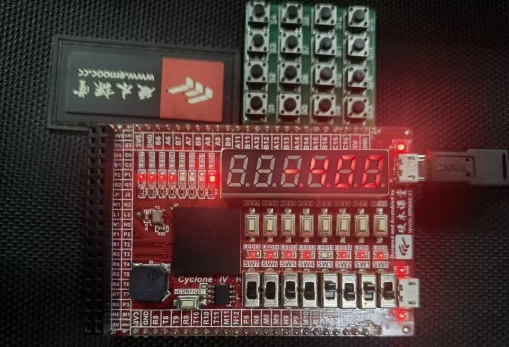
3.清零功能：设计时加入了手动清零功能，对应F按键。在有数据输入时按下F按键，观察是否能实现清零功能。

实验发现，上述功能均能实现。在此各记录一例。

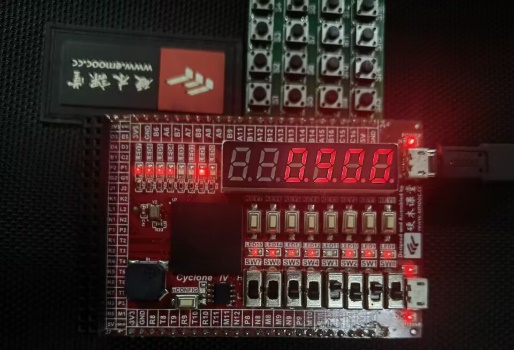
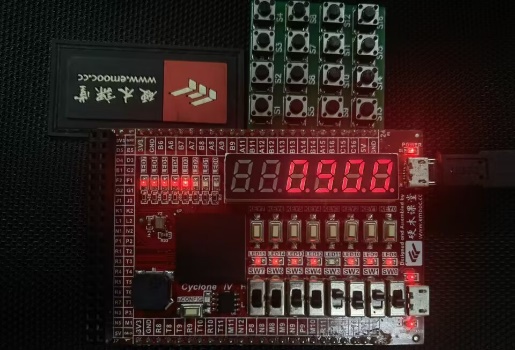
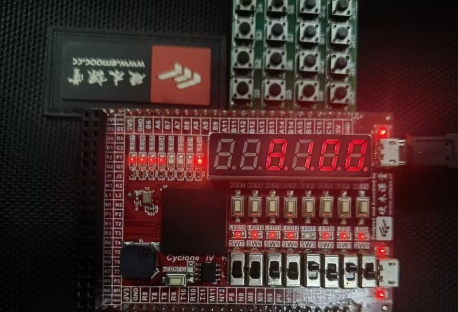
加法：9 + 9 = 18

减法：4 - 8= -4

乘法：9 x 9 = 81

**遇到的问题及解决方法：**

1.最开始连接到键盘上时，输入现象为：有时按键能改变数码管显示，有时不能；数码管显示的内容不定，有时是C，有时是9。

在这种情况下，根据以往经验，首先考虑到了接出状态机。将状态机利用LED0~3显示，对应状态即为对应LED角标。检测结果发现，状态机在1~3间顺序循环，没有达到状态0。因此，有两种推测：跳过了状态0，或总控电路的计数有问题。由于总控电路State仿真结果正确，再次检查无误，判断为跳过了状态0。

2.判断为“跳过状态0”后，由于State没有问题，检查芯片的外电路接线逻辑。推演发现，外电路中寄存芯片clean（所有芯片的清零输入）输入实现的逻辑是：当输入为number且为状态Y[1]时即清零并寄存。由于寄存后会一直保持数据，且此时设计的State是通过寄存器控制的，因此直接跳过了状态0。修改后状态机无误。

**四、实验仪器**

FPGA板

**五、实验小结**（总结实验完成情况，对设计方案和实验结果做必要的讨论，简述实验收获和体会）

本次实验是大作业，也理所当然地是做得最长的一个实验，共计3周，比别人多出来的一周是序列检测器割下来的。看了眼自己的报告，嗯，21页，可能写完这坨就22页了，好像第一次报告加上修改的三个实验我也才写了30页。虽然有一大坨都是图，但这次显然是也是打字最多的一次。我怎么知道的呢？因为手疼

这次实验可谓倾尽了我学期所学，从自己突发奇想地乱摸索源代码最后真的捣鼓出一个非常好看的负号，到报告的撰写，是本学期各方面的集大成者。为什么提到了写报告呢？因为这次报告我自我感觉良好，嘿嘿。其实是因为这次报告我采用了一些相对来说标准化的方式去写，比如标准化命名（这是在上一个实验里为了打字省事学会的，像写论文一样来一个ASPRG）、好看的状态图（刚学会用draw.io这个网站，真的好好看！）、公式的1.5倍行距（因为也在写大物实验论文，自行推导的公式有几十个，密密麻麻很丑，发现改成1.5倍行距就很好看了）等等等等，所以真的可以说是本学期各方面，不仅仅是数电实验的集大成者。

也发现了，其实很多任务都是进行人机交互。这次改负号其实不仅仅是想到这一层就去试了，还有一层原因是，我在自行实验时思考过实验二与本实验的区别。其实不是功能实现难度的提升，主要的难度提升在于人机交互。暑期学校的C++大作业也如出一辙，程序主体是早就编写好的功能，但为功能加上人机交互的部分，难度甚至比实现基本功能还大。所以我想，既然已经加了那么多人机交互的难度，最后负号还要用F表示，好像有点亏。所以我才实践了负号的修改。

**六、参考资料**

《数字逻辑设计实践2023年教学计划A\_V1.0》