**东南大学电工电子实验中心**

**实 验 报 告**

**课程名称： 数字与逻辑电路实验A**

**第五次实验**

实验名称： FPGA 小系统设计

院 （系）： 自动化 专 业： 自动化

姓 名： 邹滨阳 学 号： 08022305

实 验 室: 金智楼电子技术4室105 实验组别： 无

同组人员： 无 实验时间：2023年12月7日

评定成绩： 审阅教师：

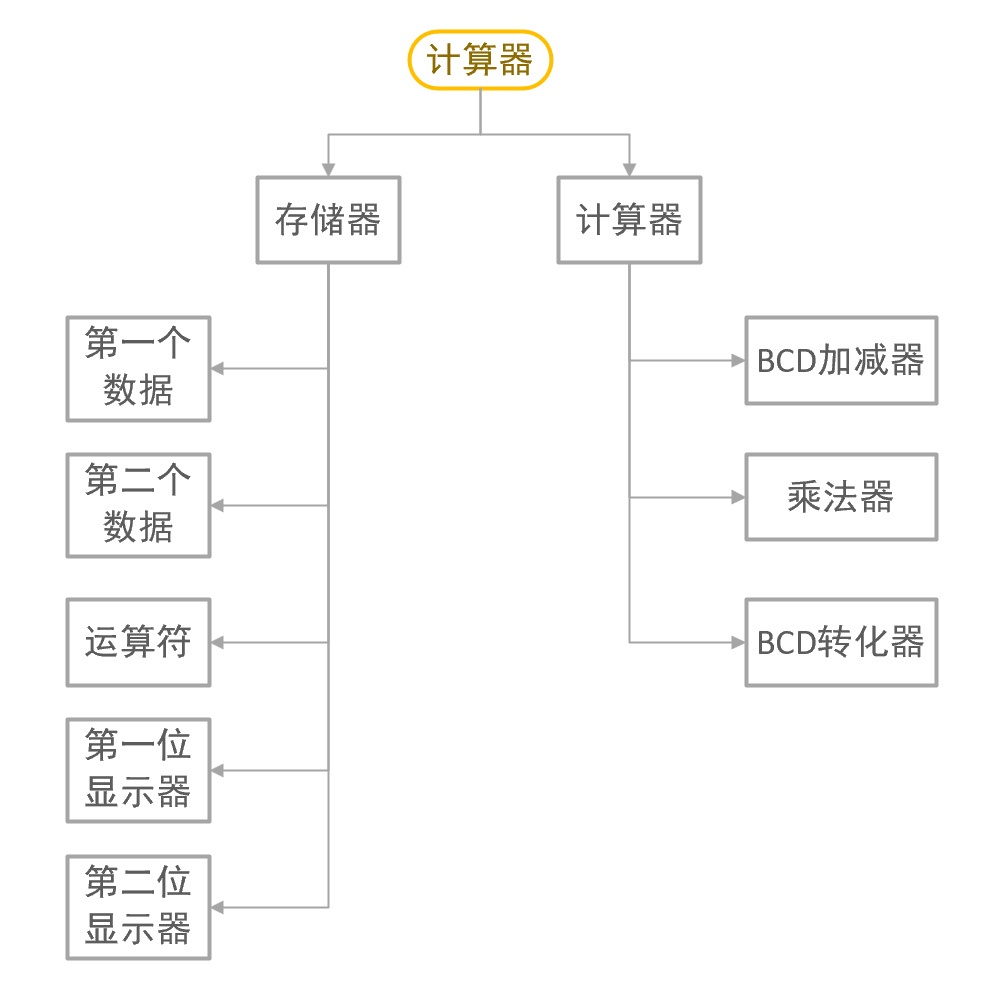
**一、实验目的**

1. 综合前面所学的各项内容

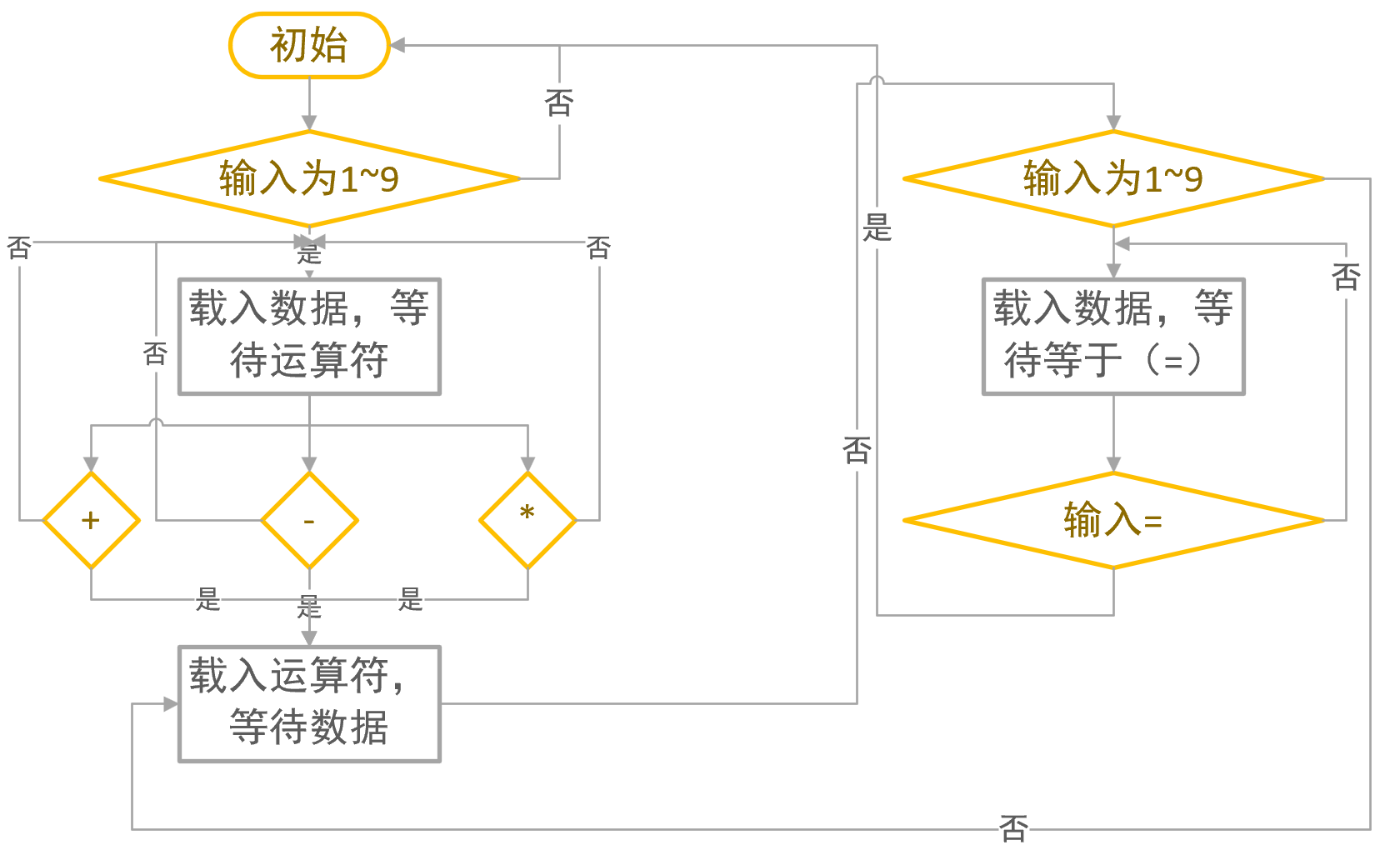
2. 了解掌握数字系统设计的流程和方法

3. 掌握复杂电路连接和调试技能

**二、实验原理（预习报告内容，如无，则简述相关的理论知识点。）**

大致为把该数字系统进行分类：  


绘制流程图如下：



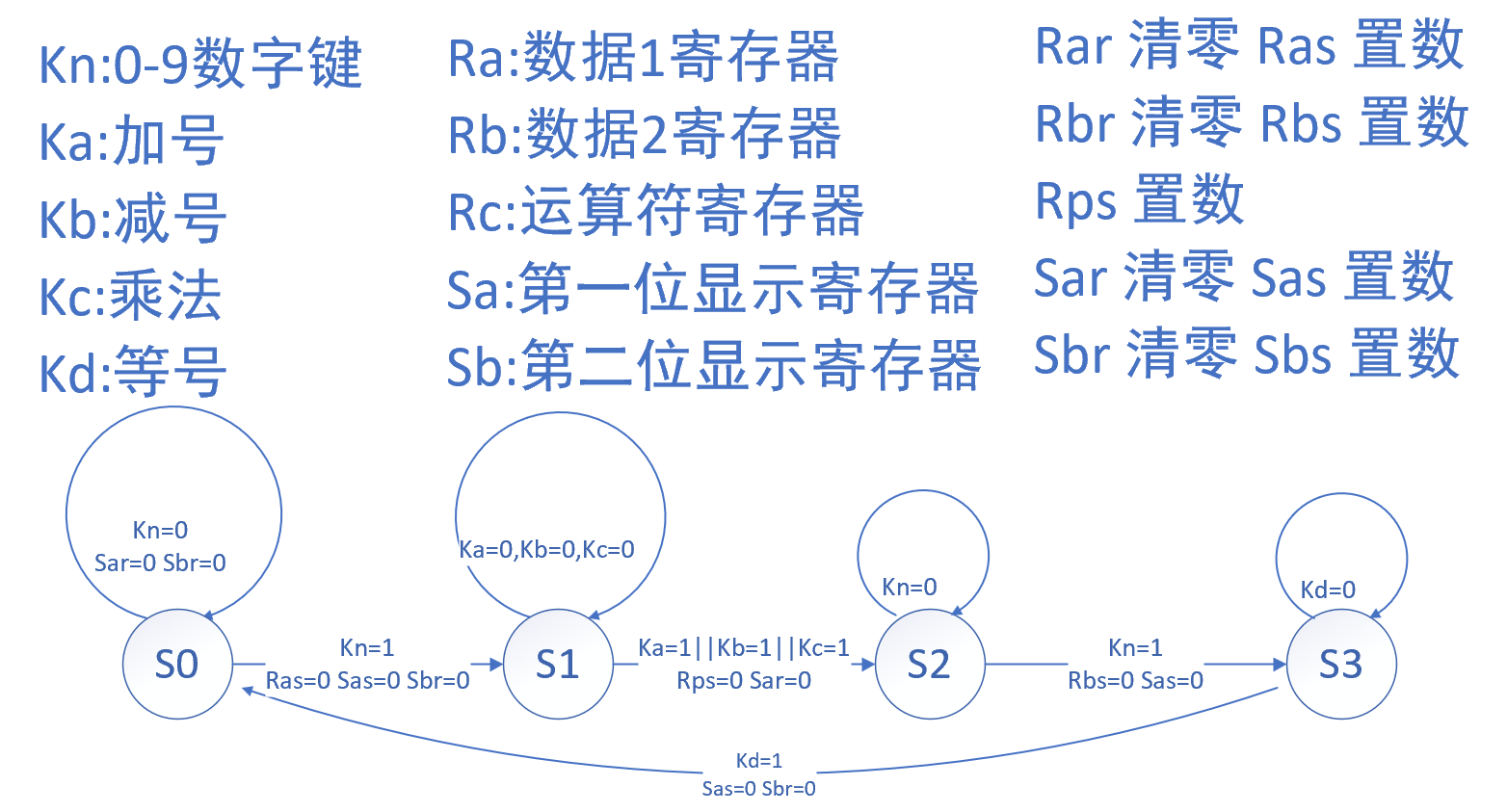
根据流程我们可以划分出以下几个状态和对应编号

|  |  |  |
| --- | --- | --- |
| 状态 | 编号 | 功能 |
| S0 | 000 | 初始状态（显示0） |
| S1 | 001 | 输入一个数据后的状态（显示载入的数据） |
| S2 | 010 | 输入运算后的状态（显示0） |
| S3 | 011 | 输入下一个数据后的状态（显示载入的数据） |
| S4 | 100 | 输入等于=后的状态（显示答案） |

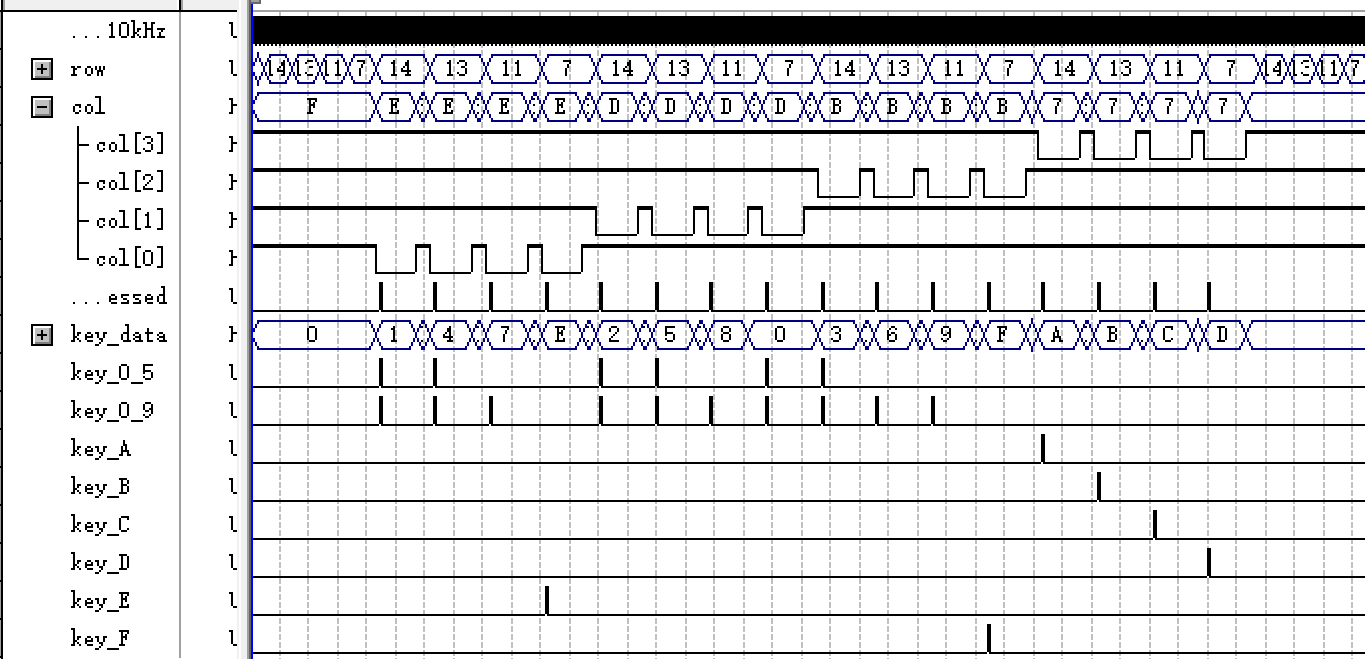
其实可以把S4和S0合并

|  |  |  |
| --- | --- | --- |
| 状态 | 编号 | 功能 |
| S0 | 00 | 初始状态（显示上一次的答案（没有则为0）） |
| S1 | 01 | 输入一个数据后的状态（显示载入的数据） |
| S2 | 10 | 输入运算后的状态（显示0） |
| S3 | 11 | 输入下一个数据后的状态（显示载入的数据） |

所以可以画出状态转移图：



这里打错了应该是Sbs=0



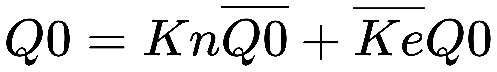
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 状态 | | 输入 | | | | | 状态编码 | | | | 输出 | | | | | | | | |
| 现态 | 次态 | Ka + | Kb - | Kc \* | Kd = | Kn | 现态 | 次态 | EN | LD | Sbs | Sbr | Sas | Sar | Rps | Rbs | Rbr | Ras | Rar |
| S0 | S0 | Φ | Φ | Φ | Φ | 0 | 00 | 00 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| S1 | 0 | 0 | 0 | 0 | 1 | 01 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| S1 | S1 | 0 | 0 | 0 | Φ | Φ | 01 | 01 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| S2 | 1 | 0 | 0 | 0 | 0 | 10 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| S2 | S2 | Φ | Φ | Φ | Φ | 0 | 10 | 10 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| S3 | 0 | 0 | 0 | 0 | 1 | 11 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| S3 | S3 | Φ | Φ | Φ | 0 | Φ | 11 | 11 | Φ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| S0 | 0 | 0 | 0 | 1 | 0 | 00 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
|  |  |  |  |  |  |  |  |  | 计数 | 置数 | 置数 | 清零 | 置数 | 清零 | 置数 | 置数 | 清零 | 置数 | 清零 |
|  |  |  |  |  |  |  |  |  |  |  | 高位显示 | | 低位显示 | | 符号 | 数据二 | | 数据一 | |

利用卡诺图计算相关的数据

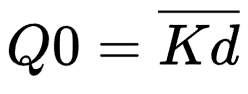
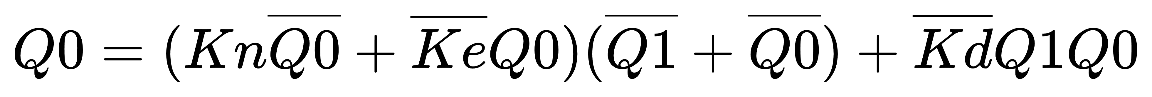
（括号内是错误）（令Ke=Ka+Kb+Kc

Q0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| KeKn\Q1Q0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | Φ | 0 |
| 01 | 1 | 1 | Φ | 1 |
| 11 | 1 | 0 | Φ | 1 |
| 10 | 0 | 0 | Φ | 0 |

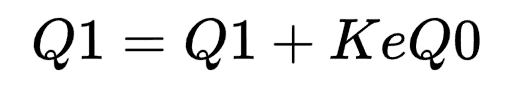


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kd\Q1Q0 | 00 | 01 | 11 | 10 |
| 0 | Φ | Φ | 1 | Φ |
| 1 | Φ | Φ | 0 | Φ |

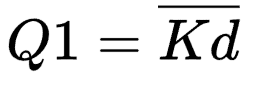
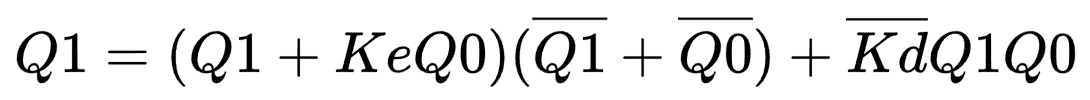
 

Q1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| KeKn\Q1Q0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | Φ | 1 |
| 01 | 0 | 0 | Φ | 1 |
| 11 | 0 | 1 | Φ | 1 |
| 10 | 0 | 1 | Φ | 1 |

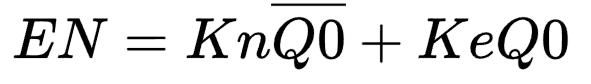


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kd\Q1Q0 | 00 | 01 | 11 | 10 |
| 0 | Φ | Φ | 1 | Φ |
| 1 | Φ | Φ | 0 | Φ |

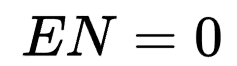
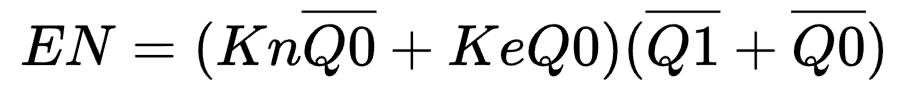
 

EN：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| KeKn\Q1Q0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | Φ | 0 |
| 01 | 1 | 0 | Φ | 1 |
| 11 | 1 | 1 | Φ | 1 |
| 10 | 0 | 1 | Φ | 0 |

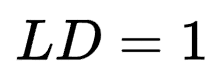


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kd\Q1Q0 | 00 | 01 | 11 | 10 |
| 0 | Φ | Φ | 0 | Φ |
| 1 | Φ | Φ | Φ | Φ |

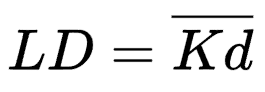
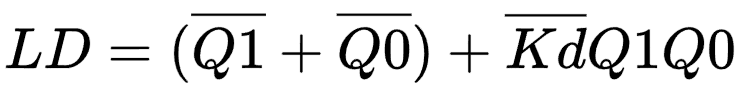
 

LN：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| KeKn\Q1Q0 | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | Φ | 1 |
| 01 | 1 | 1 | Φ | 1 |
| 11 | 1 | 1 | Φ | 1 |
| 10 | 1 | 1 | Φ | 1 |



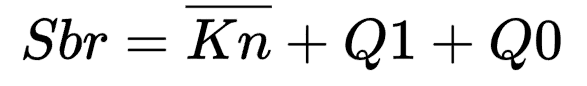
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kd\Q1Q0 | 00 | 01 | 11 | 10 |
| 0 | Φ | Φ | 1 | Φ |
| 1 | Φ | Φ | 0 | Φ |

Sbs：与LD一致

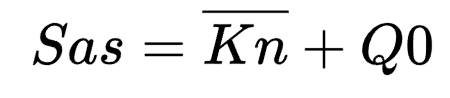
Sbr：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| KeKn\Q1Q0 | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 1 | 1 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 0 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

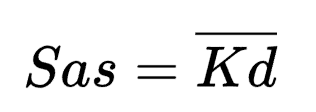
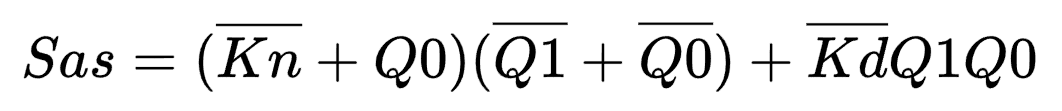


Sas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| KeKn\Q1Q0 | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | Φ | 1 |
| 01 | 0 | 1 | Φ | 0 |
| 11 | 0 | 1 | Φ | 0 |
| 10 | 1 | 1 | Φ | 1 |

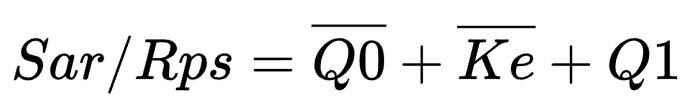


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kd\Q1Q0 | 00 | 01 | 11 | 10 |
| 0 | Φ | Φ | 1 | Φ |
| 1 | Φ | Φ | 0 | Φ |

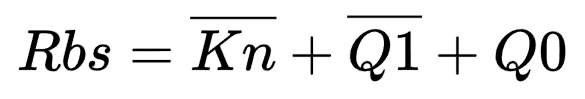
Sar与Rps:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| KeKn\Q1Q0 | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 1 | 0 | 1 | 1 |
| 10 | 1 | 0 | 1 | 1 |



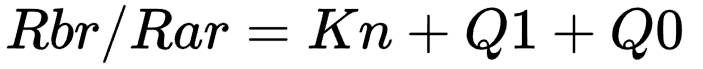
Rbs：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| KeKn\Q1Q0 | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 1 | 1 | 0 |
| 11 | 1 | 1 | 1 | 0 |
| 10 | 1 | 1 | 1 | 1 |

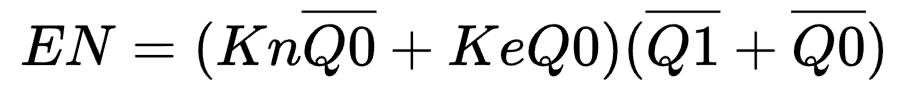


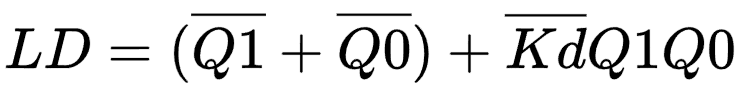
Rbr/Rar:

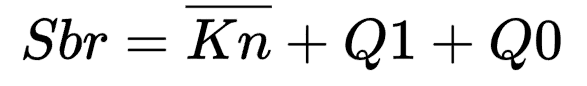
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| KeKn\Q1Q0 | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 1 | 1 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 1 | 1 | 1 |

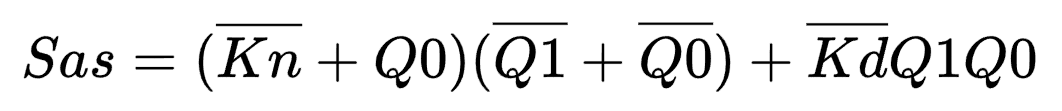


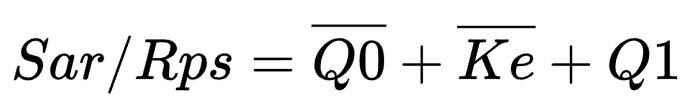
Ras:与Sbr相同

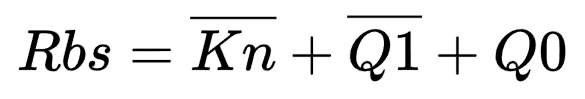
总结：  


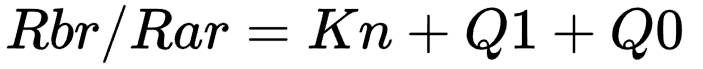
Sbs/LD: 

Sbr/Ras: 

Sas: 

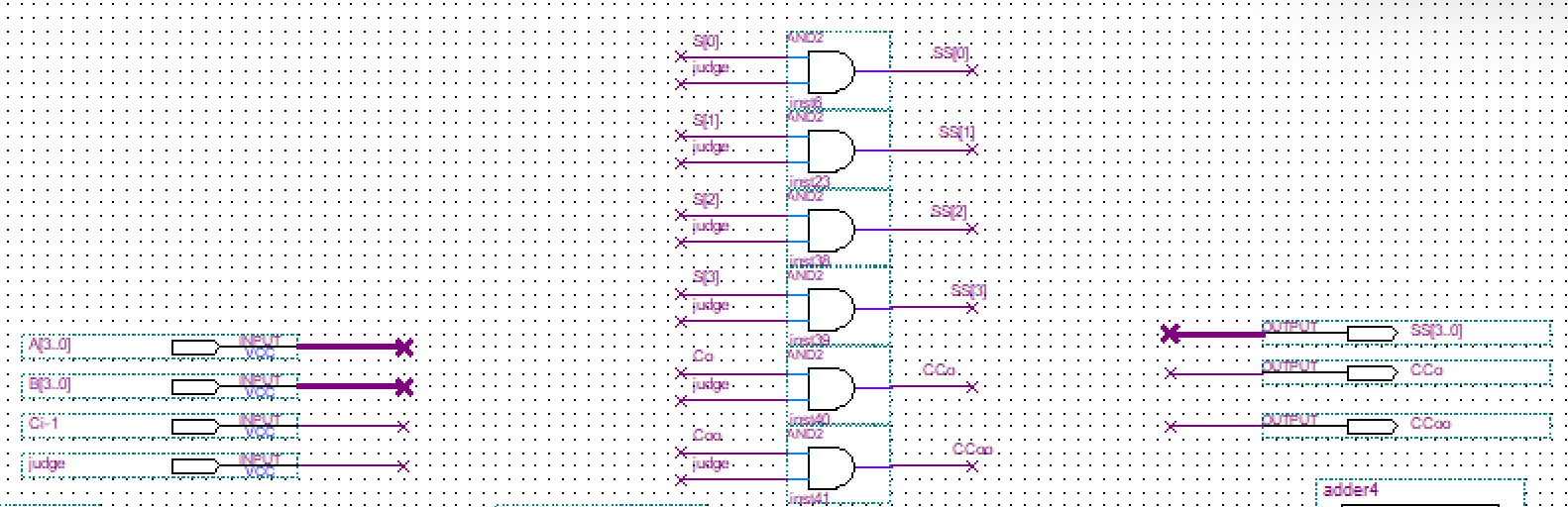
Sar/Rps: 

Rbs: 

Rbr/Rar: 

为了使原本的BCD加减器能根据实际情况启用或者禁用，所以增加

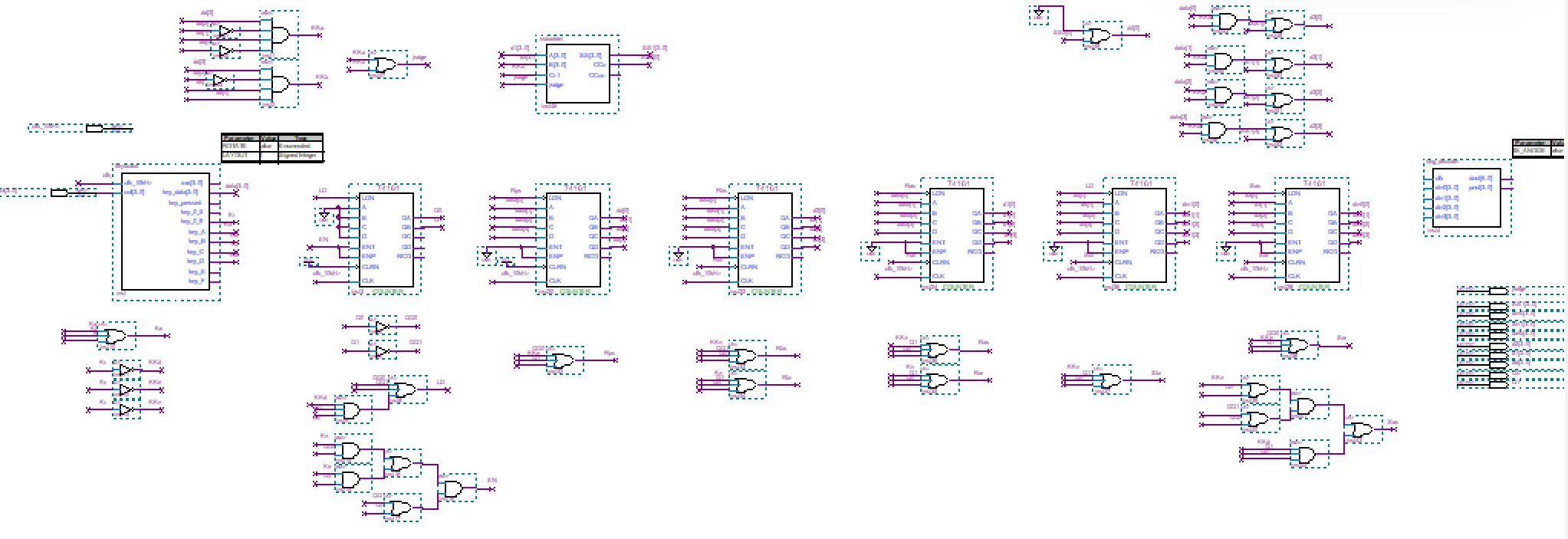
Judge为1 的时候启用，为0的时候禁用



这样修改后如果judge为0，则输出全是0

+：A 1010 -：B1011 \*：C1100 =：D1101

最终电路图如下



但是这个并没有区分模块，将在后面进行修改

**三、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：

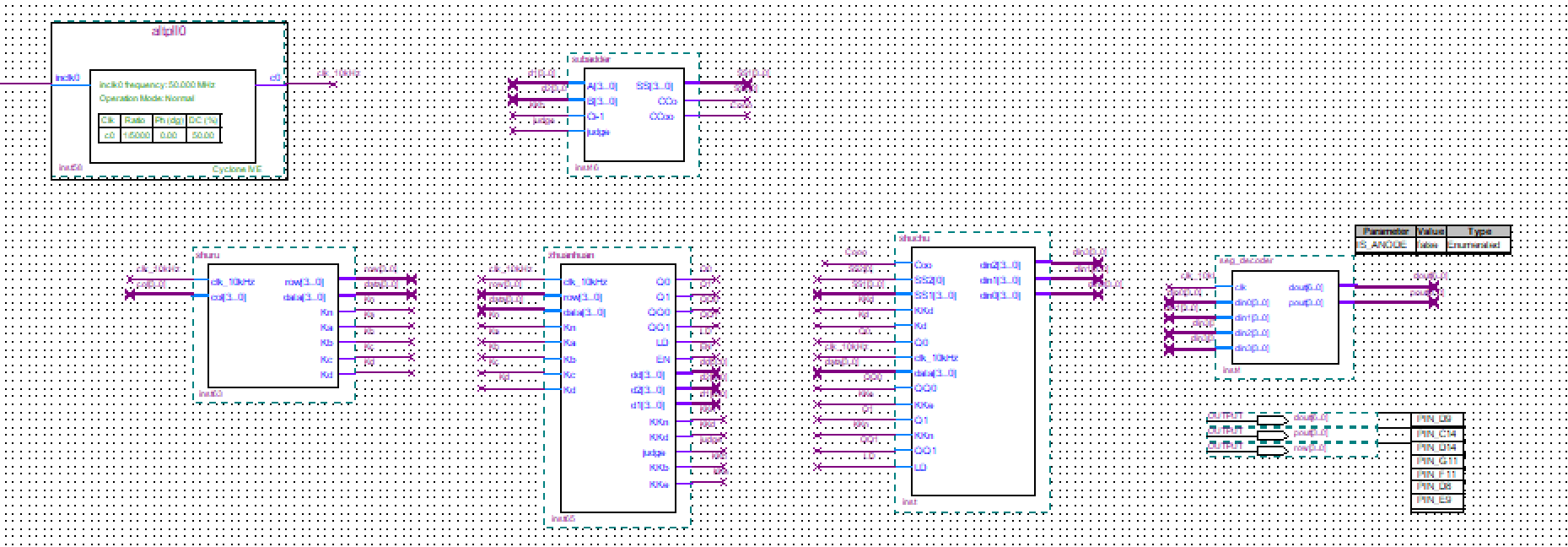
(1) 一位十进制数的相加；

(2) 一位十进制数的相减；

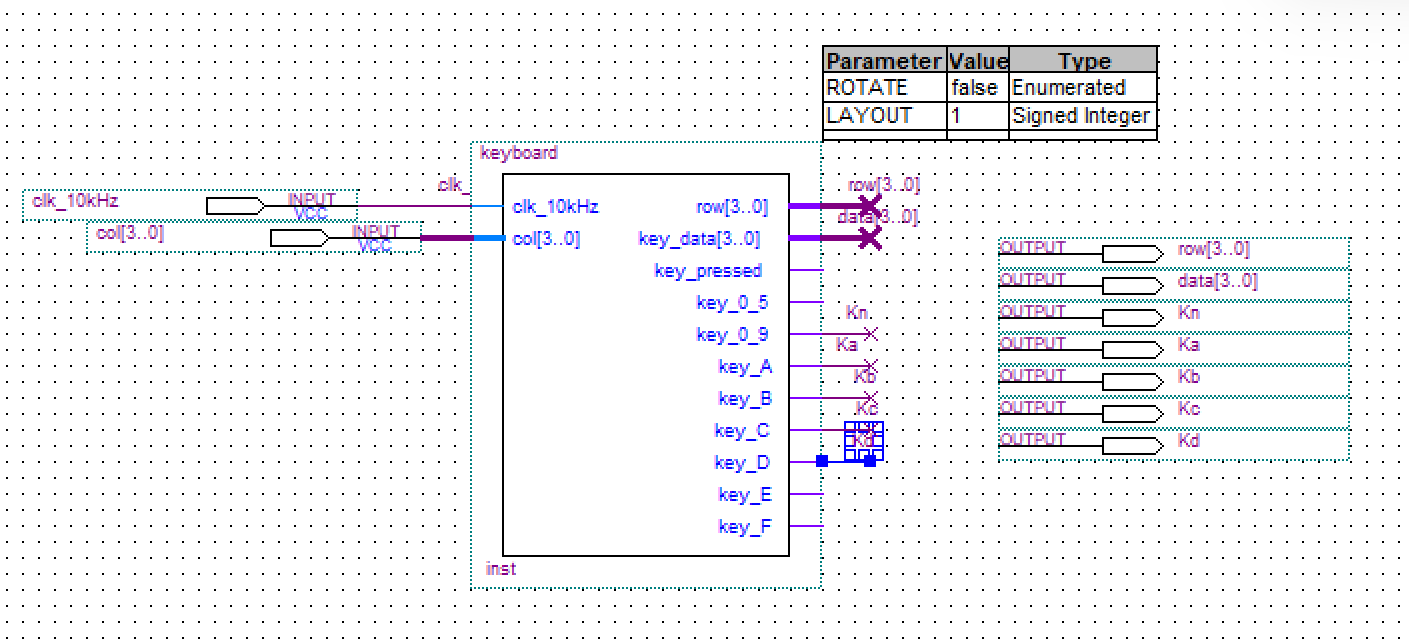
(3) 数值和运算符用4×4键盘输入（实验室提供接口程序），其中A为“+”，B为“-”，C为“×”，E为“=”

(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

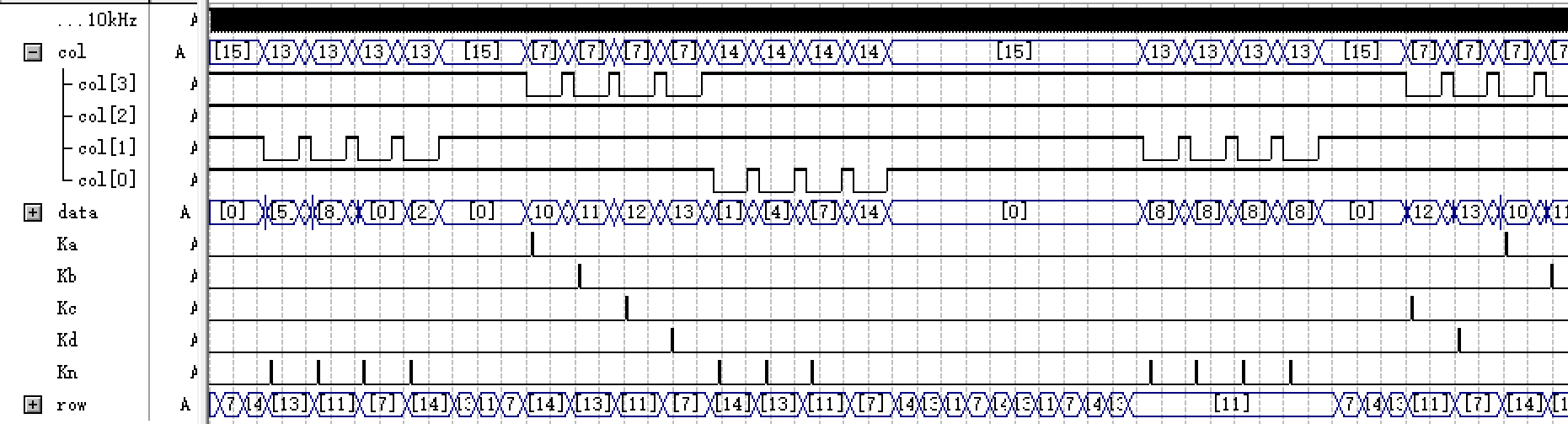
但是由于原本的设计的电路过于冗杂，所以把它拆开成六个部分，分别是输入部分（左下），脉冲部分（右上），状态转化部分（左二），BCD加减器部分（中上），显示存储部分（右二）和显示部分（右一）



这个是输入部分，主要通过读入键盘数据，转化为数值和对各个键位是否按下的判断

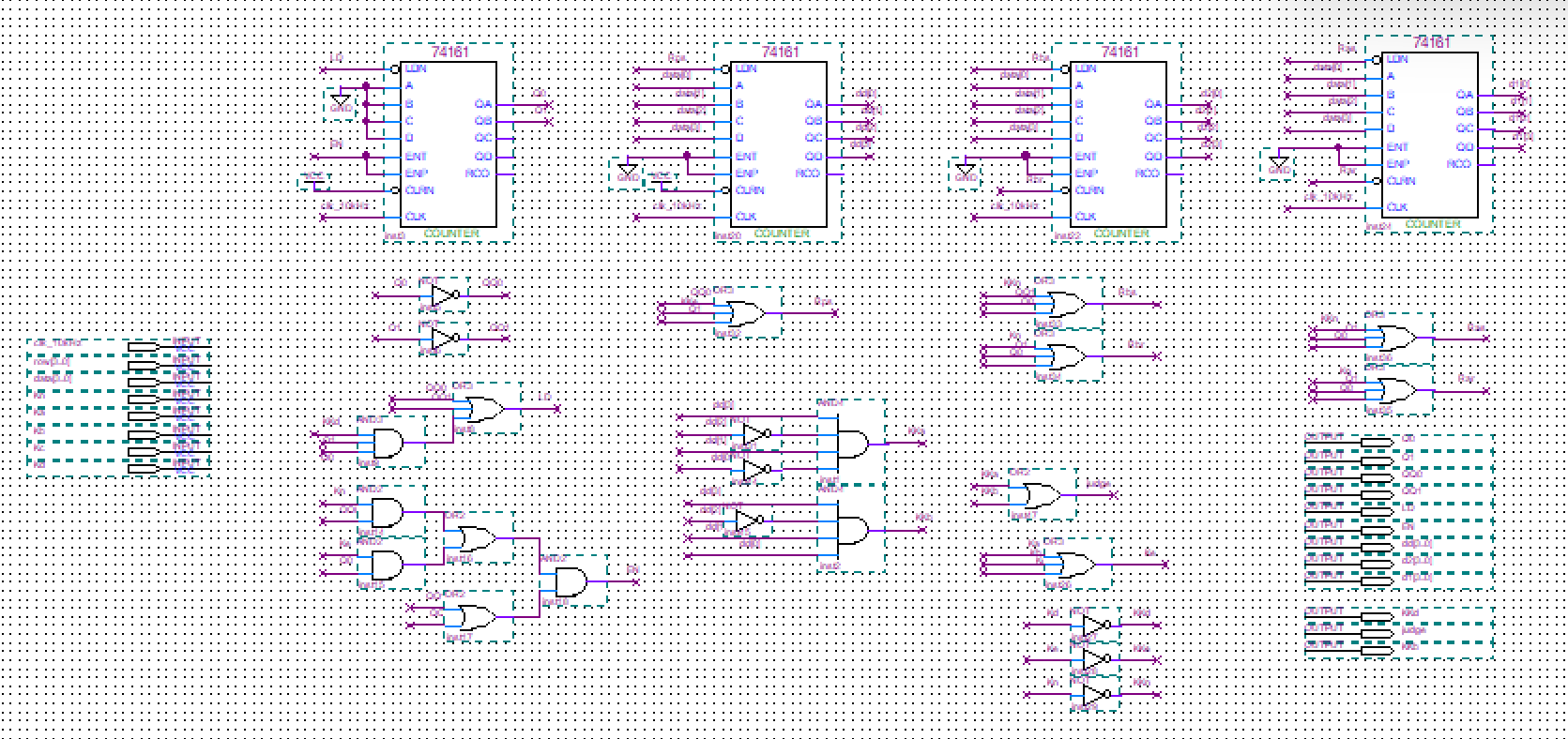


以下是仿真的案列。

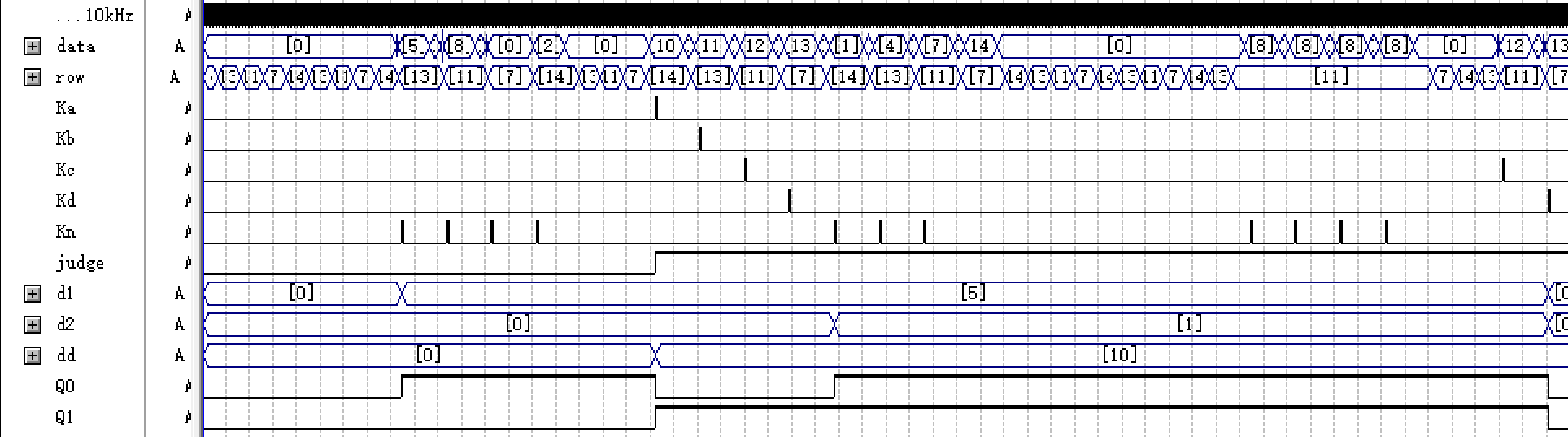


可以看出依次按下了4次数字键，1次加法，1次减法键，1次乘法键，1次等号键，7次数字键，1次乘法键，1次等号键，1次加法键，1次减法键。将这个结果导入到状态的仿真当中

这是状态记录部分，主要负责记录现在计算器处于的状态，同时负责记录运算符，和两个输入的数据，所以利用了4个74161来进行记录

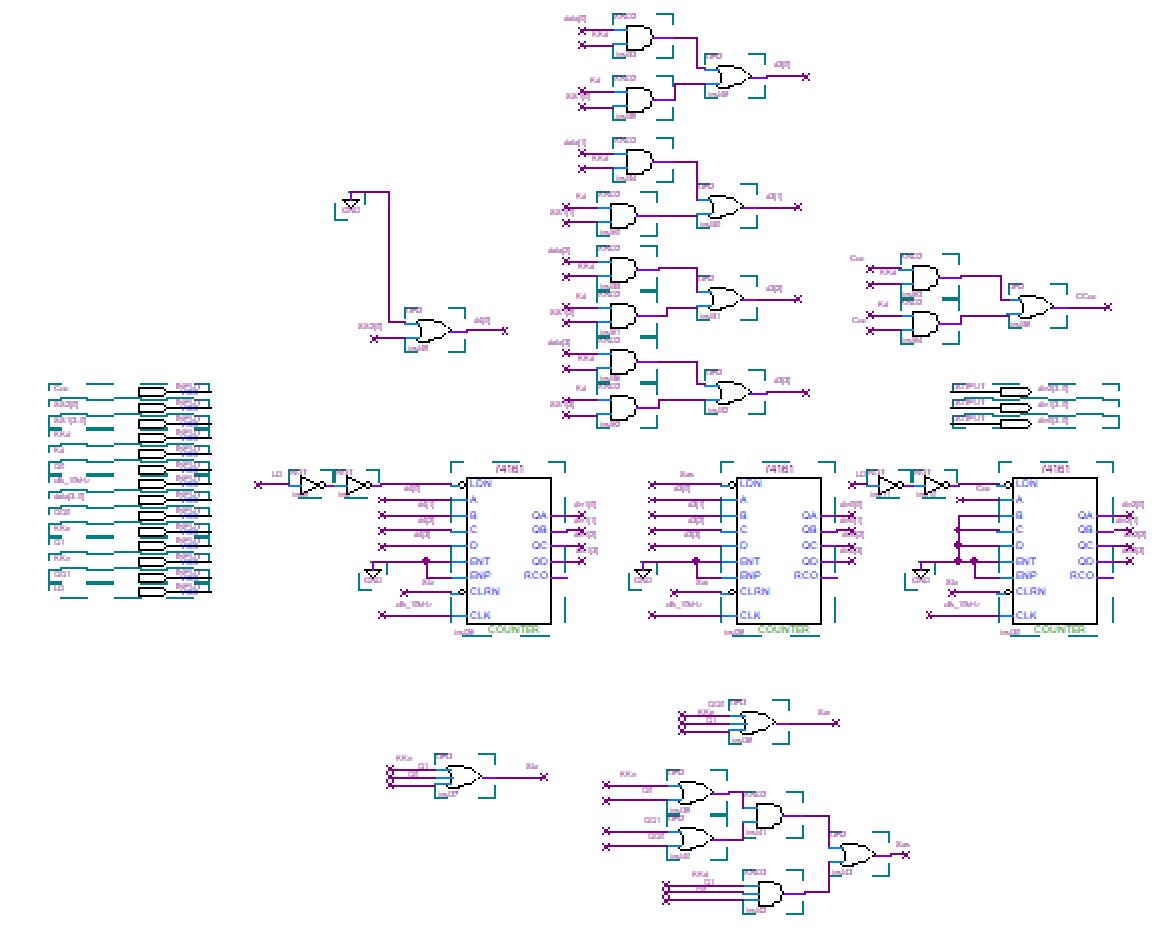


以下是把输入导入到状态转换部分仿真的结果



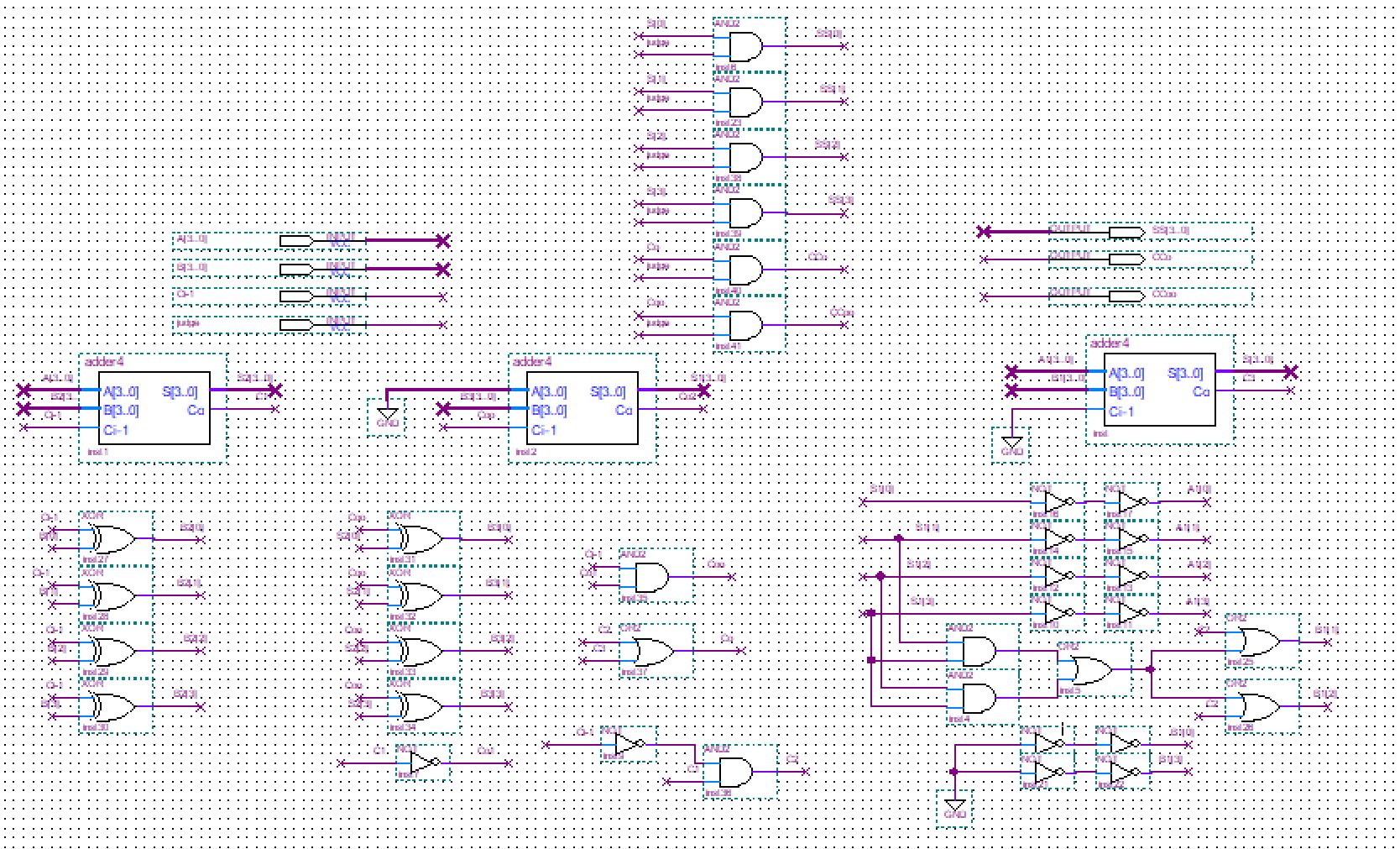
可以看见当第一次按下数字后，Q1Q0从00进入01状态，同时d1存入第一个数据5，然后可以看见之后按下的3次数字都没有造成影响。但是当按下加号键后Q1Q0从01进入10状态，同时记录的符号dd变成了10，同时judge也等于1，可以看见之后按下的减法，乘法，等号都没有产生影响。而按下下一个数字1后，Q1Q0从10进入11状态，d2记录下下一个数字1。最后按下等号键后，Q1Q0从11变成00，开始下一轮计算。

这一部分是显示存储部分，负责存储要显示的数据，分别是个位存储，十位存储和符号位存储，所以利用了3个74161进行存储。

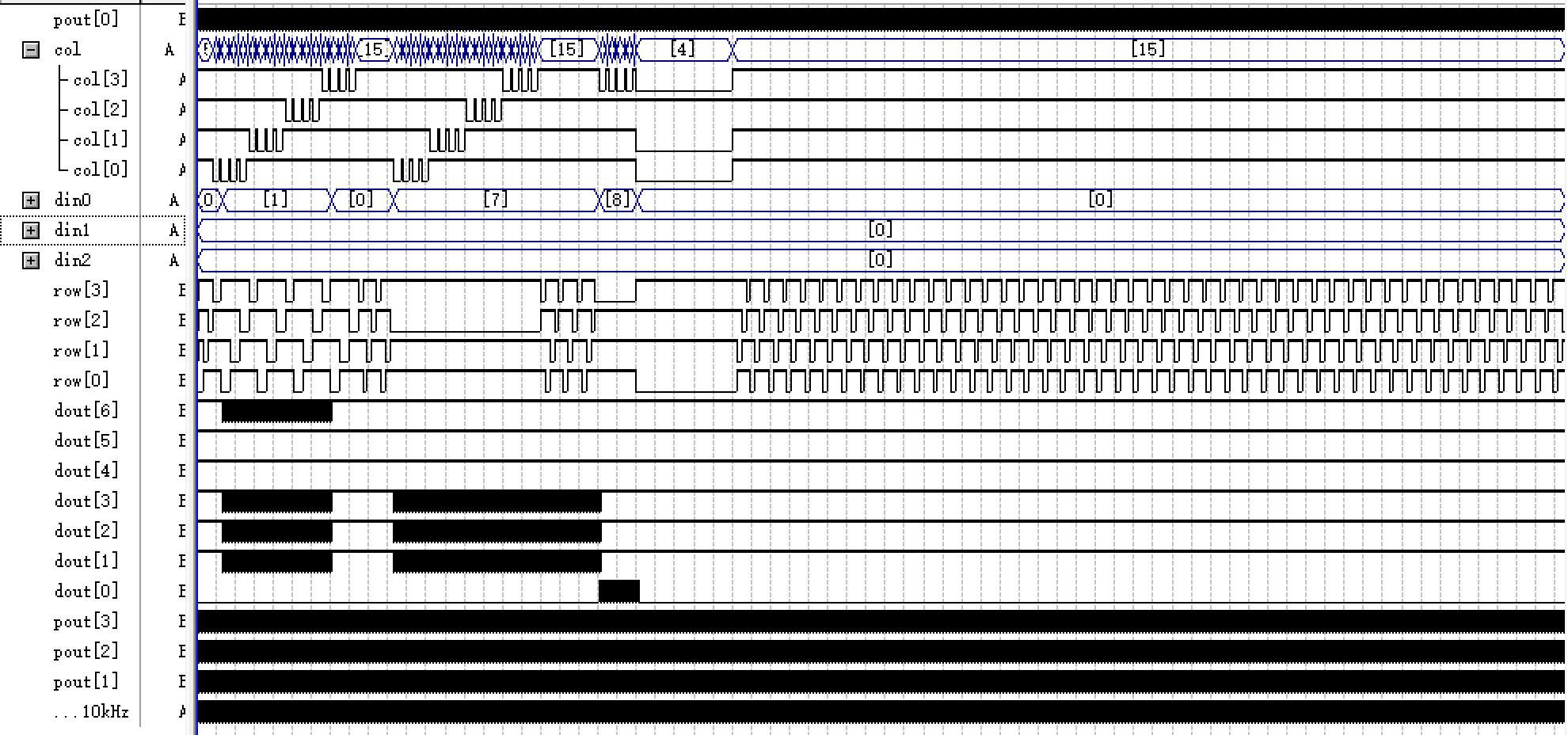


这个部分由于输入的内容过多（实际上是因为是对大电路进行拆分，导致很多数据交错，没法比较简化的完成一个板块的输入输出），因此不方便进行仿真，所以最终采用了整体仿真来解决。

下一张图是之前完成过的BCD加减器，这里就不再赘述，还有就是脉冲产生部分和输出部分，因为采用的是学校提供的文件，这里也不再赘述



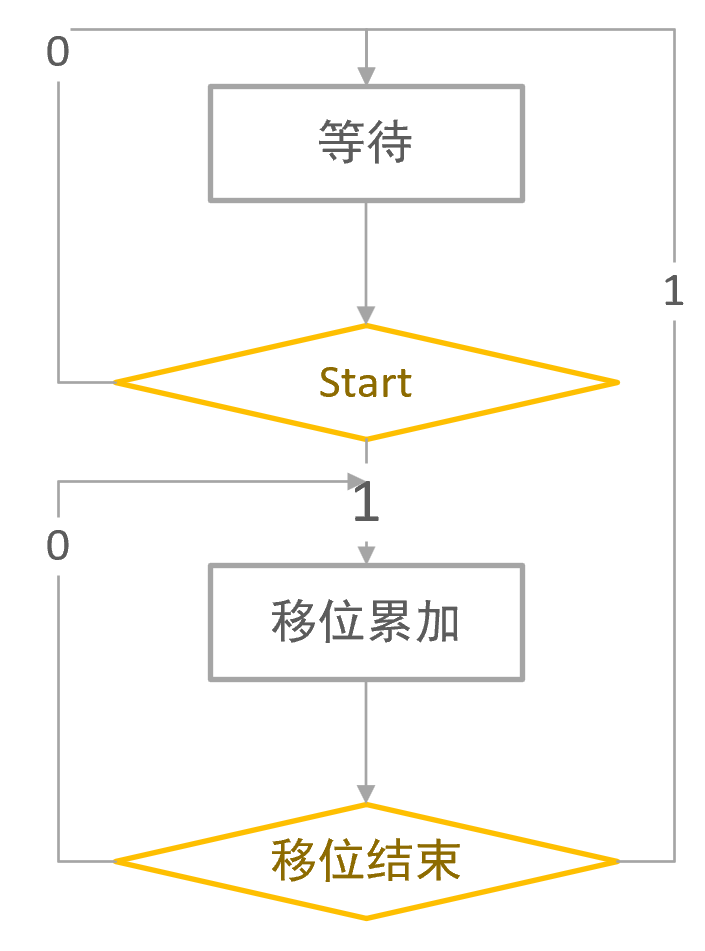
以下为整体仿真结果，最终完成了1+7=8的实现

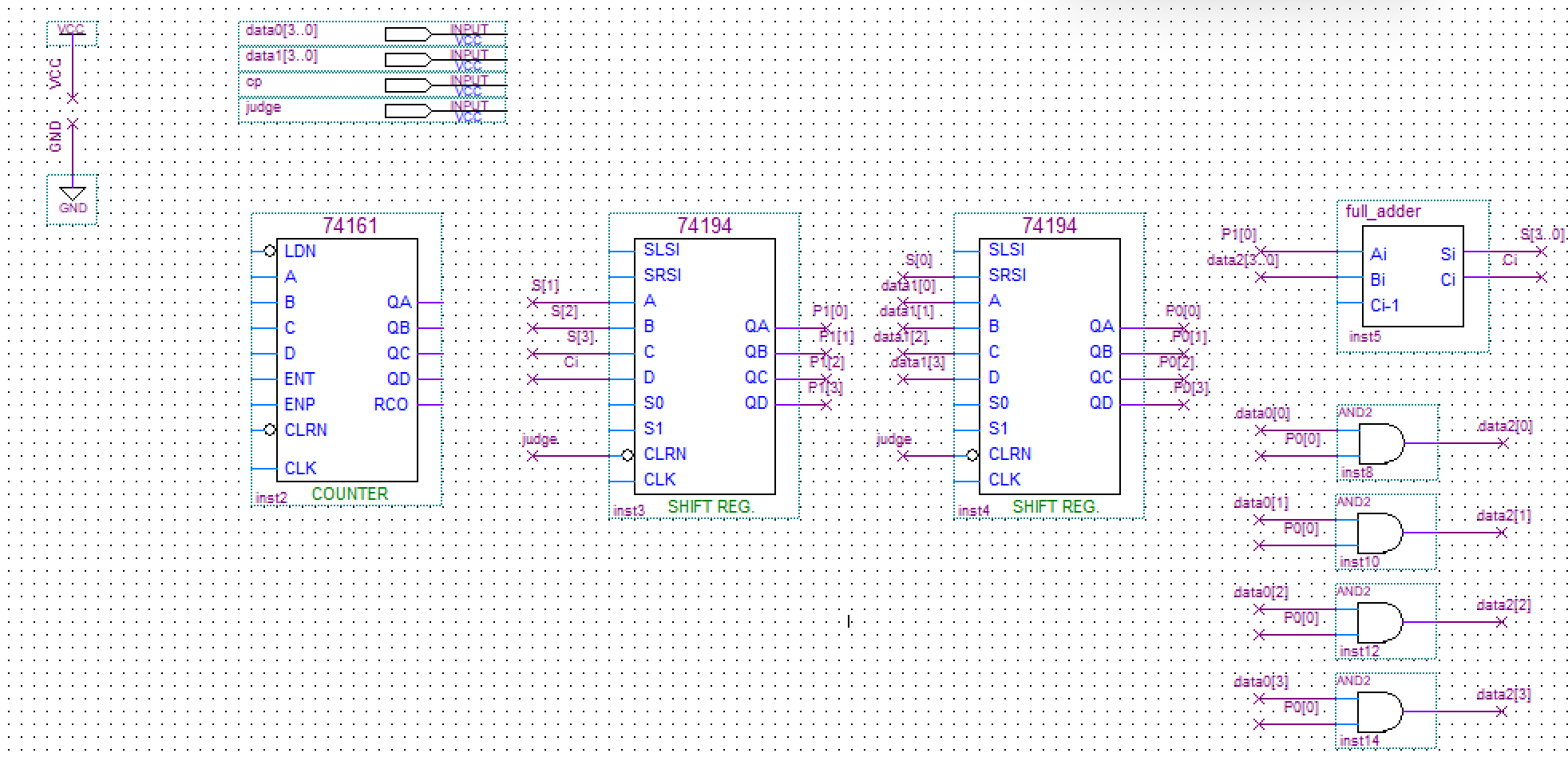


课堂上验收成功。

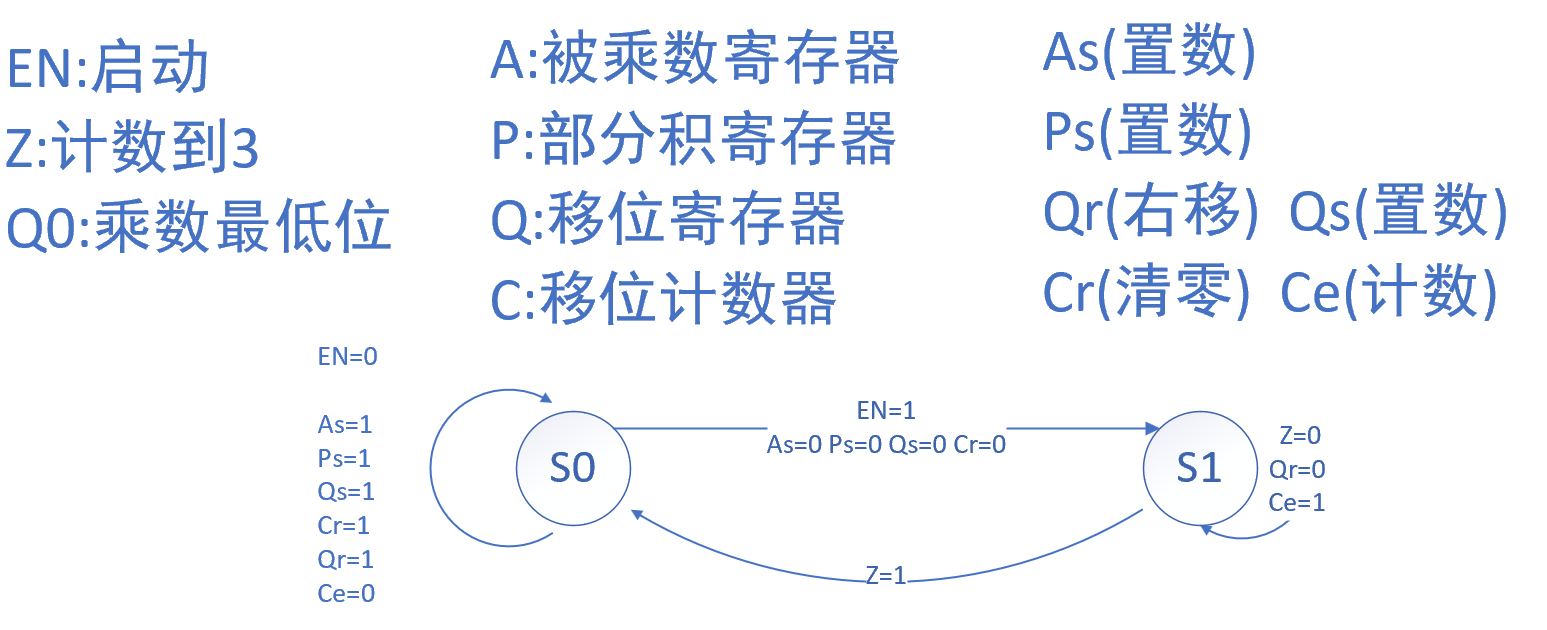
2. 实验扩展部分(30%)

(1) 一位十进制数的相乘，必须采用串行乘法实现；画出流程图



根据串行乘法算法，可以得到去除控制电路的的原理图如下：  


以此为基础画出状态转换图



要点在于完成一次计算后保持不会重新进入循环

根据实际情况对状态图进行修改，由000进入001时，将进行第一次计算，而从001到010，010到011，011到100，100到101将进行四次位移和四次重载，所以修改后的状态表如下

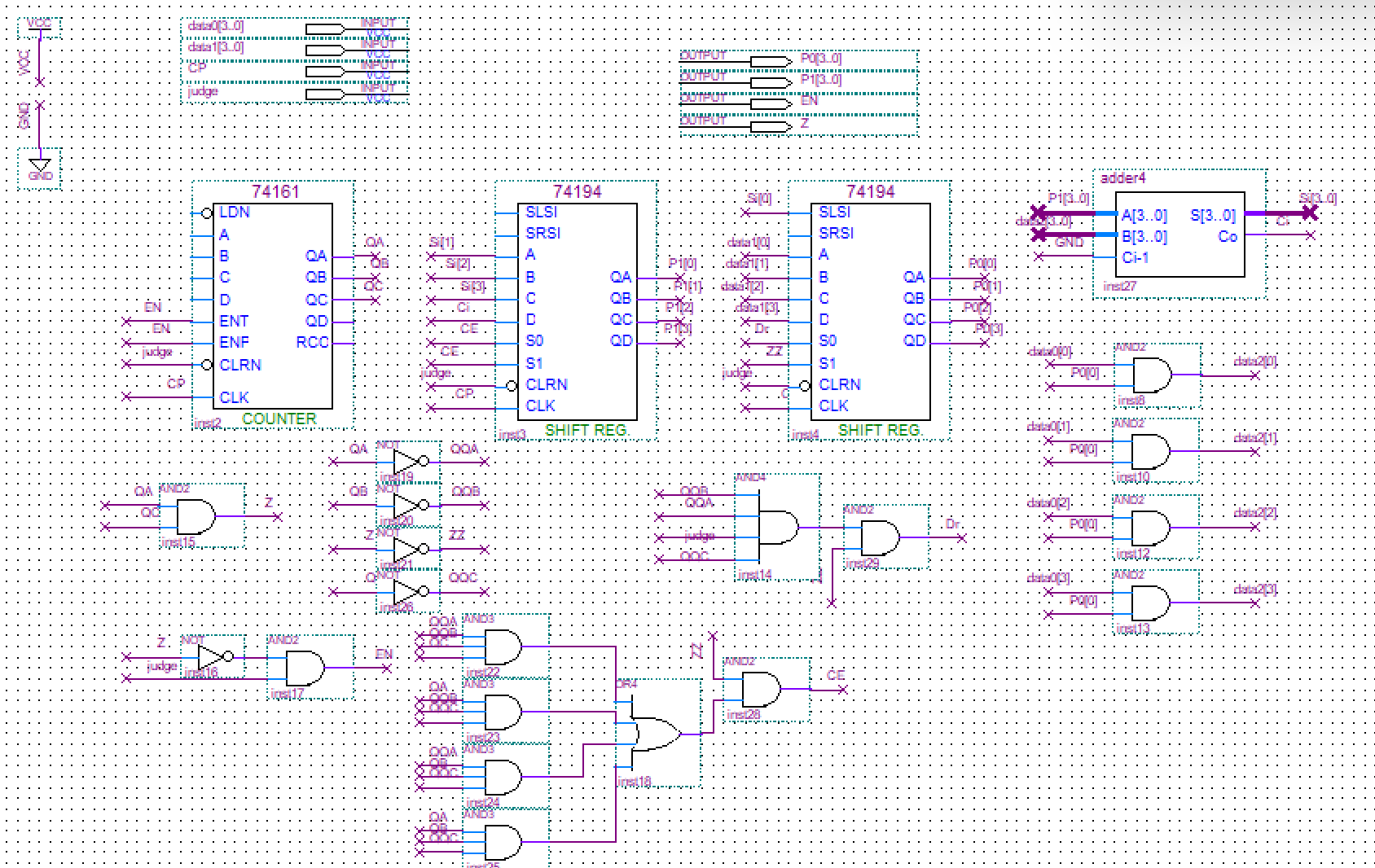
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EN | Z现态 | Z次态 | As | Ps | Qr | Qs | Cr | Ce |
| 0 | 0(000) | 0(000) | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0(000) | 0(001) | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0(001) | 0(010) | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0(010) | 1(011) | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0(011) | 1(100) | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0(100) | 1(101) | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1(101) | 0(000) | 1 | 1 | 1 | 1 | 1 | 0 |

Z=QAQC

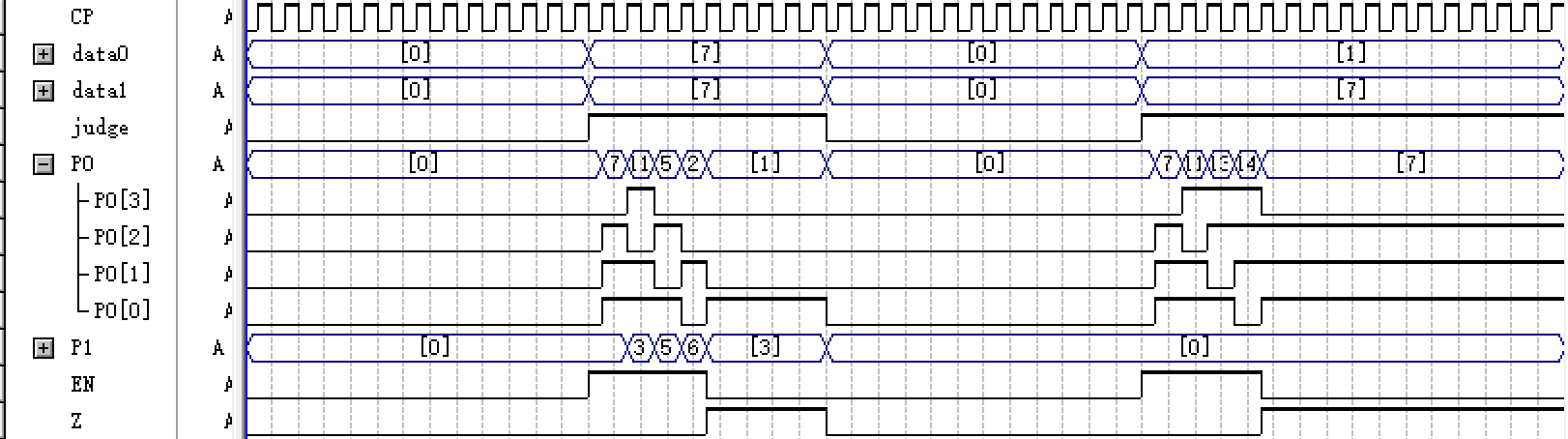
EN=(Z非)judge

DR(QS)= (Z非) (QA非)(QB非)(QC非)judge

其实原本可以去掉101直接从100连接到000，但是出于保存数据的想法，还是增加了这一过程，可以让这个数据输出的更加持久

****

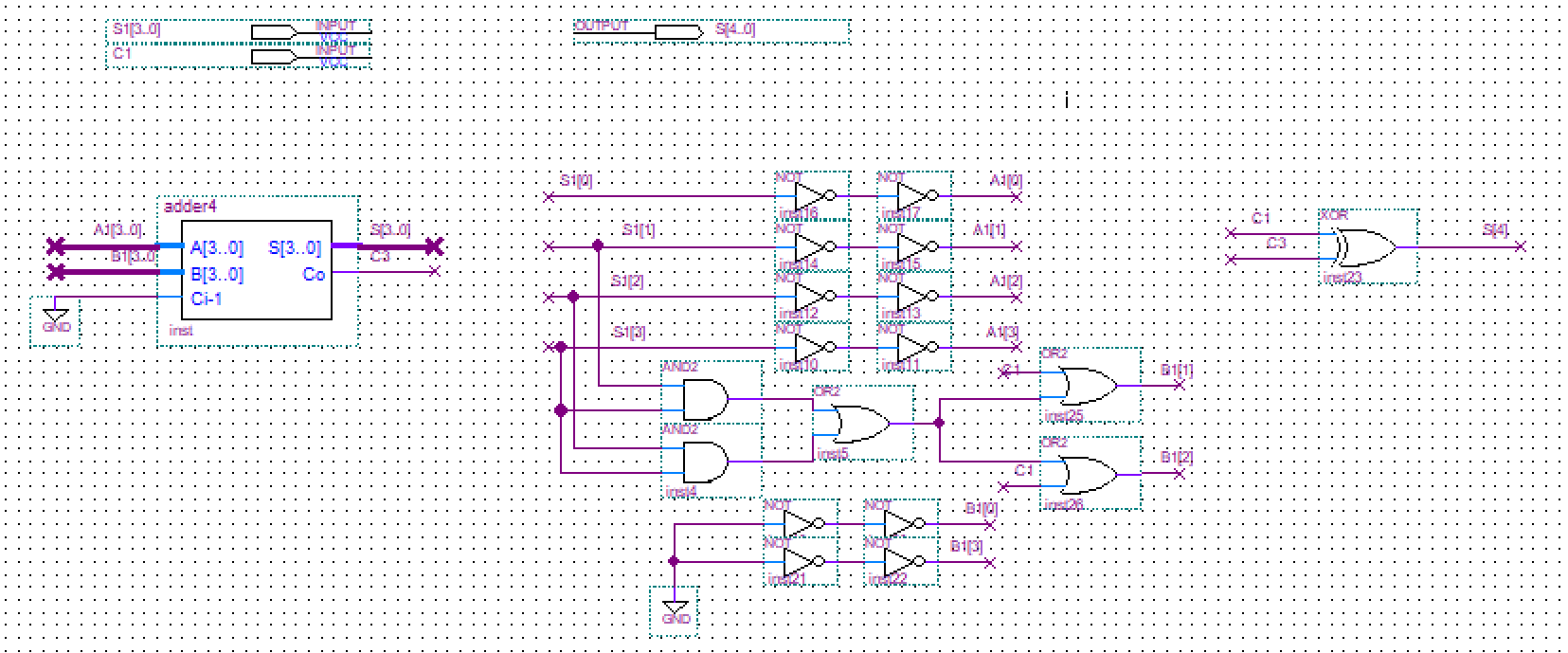
最终仿真结果如下：



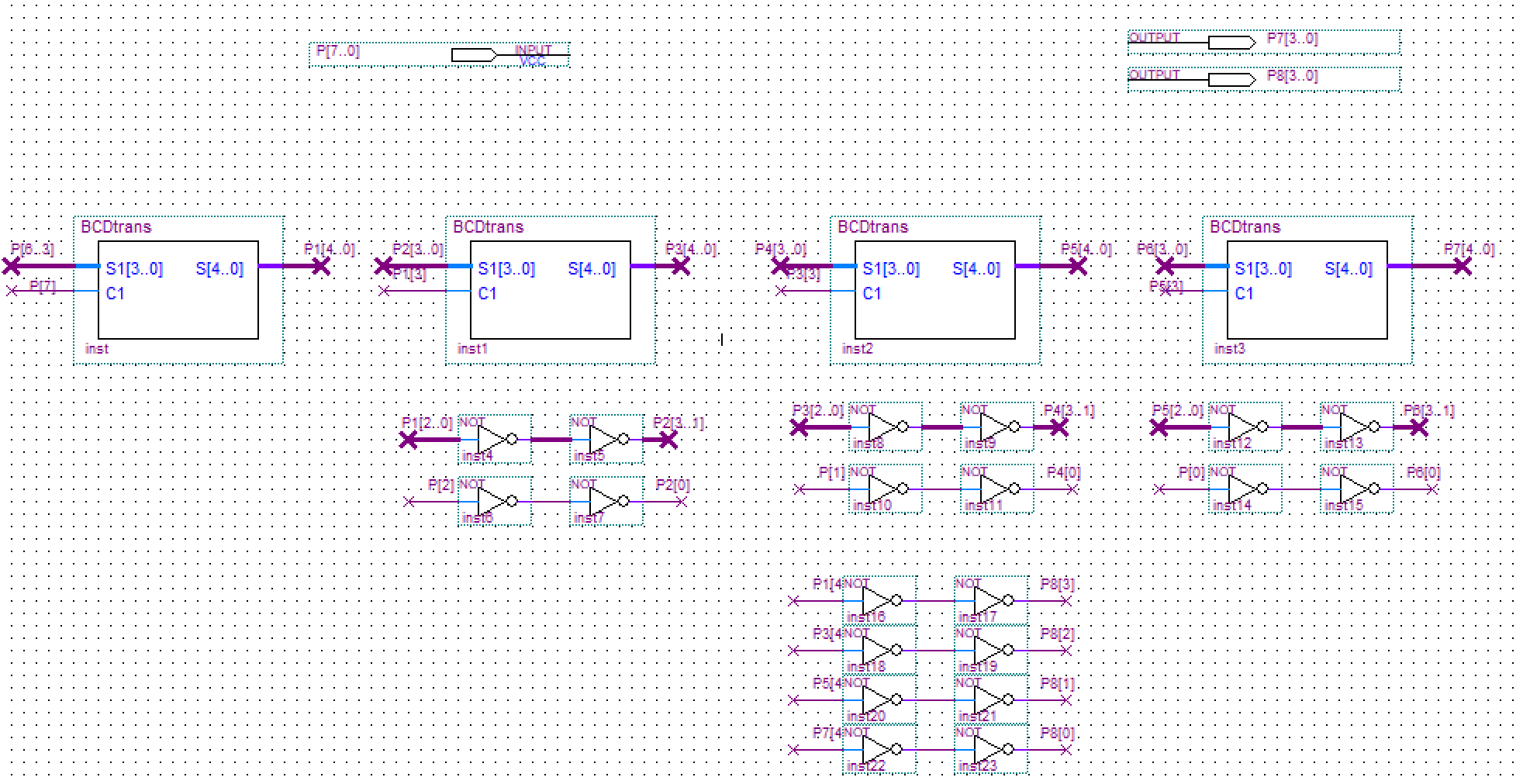
首先是按下计算7\*7，可以看到P0也就是移位寄存器位移了4次，移位计数器计数了4次，最终得到了结果110001，正确。然后是计算1\*7，也可以看到P0位移了4次，最终得到了0111，正确。

接着使其转化成BCD码

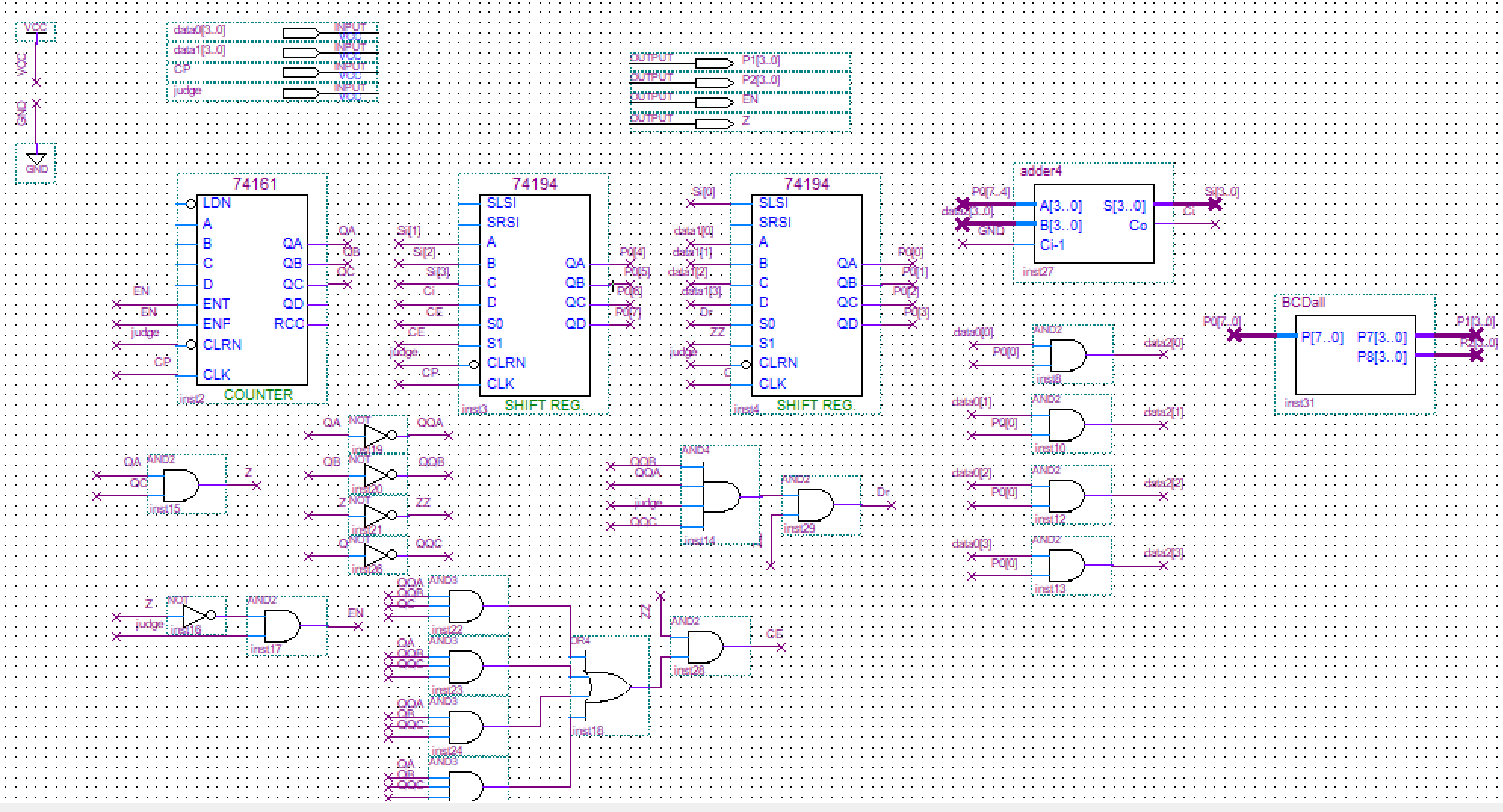
改进原有的BCD加法器，让其把输入的五位数转化成BCD码

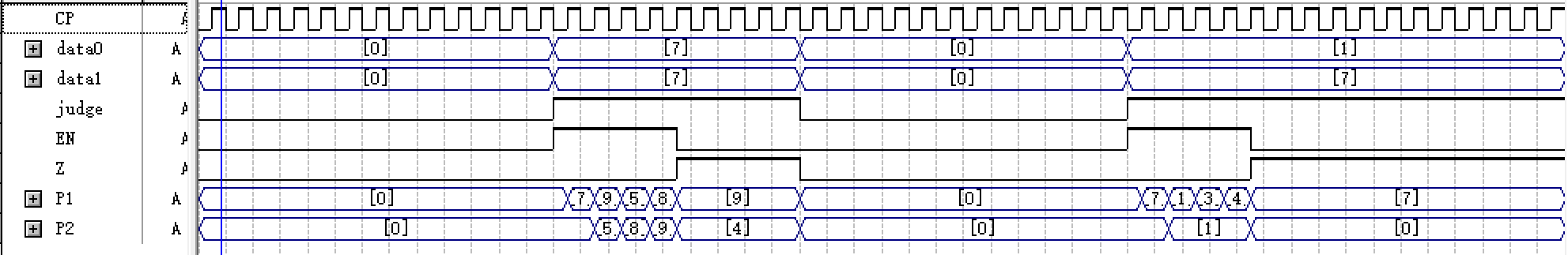


接着把其导出为原件建立7位二进制bcd转换器



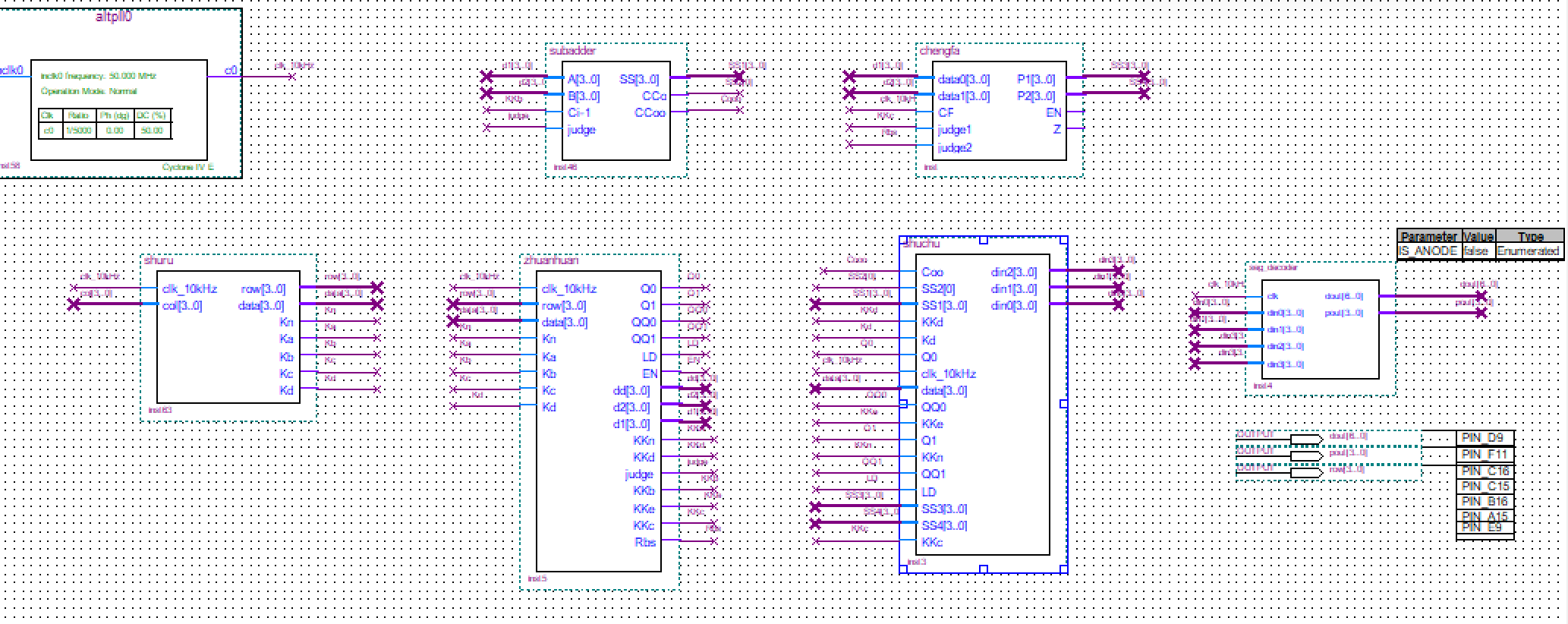
最终导入到乘法器验证



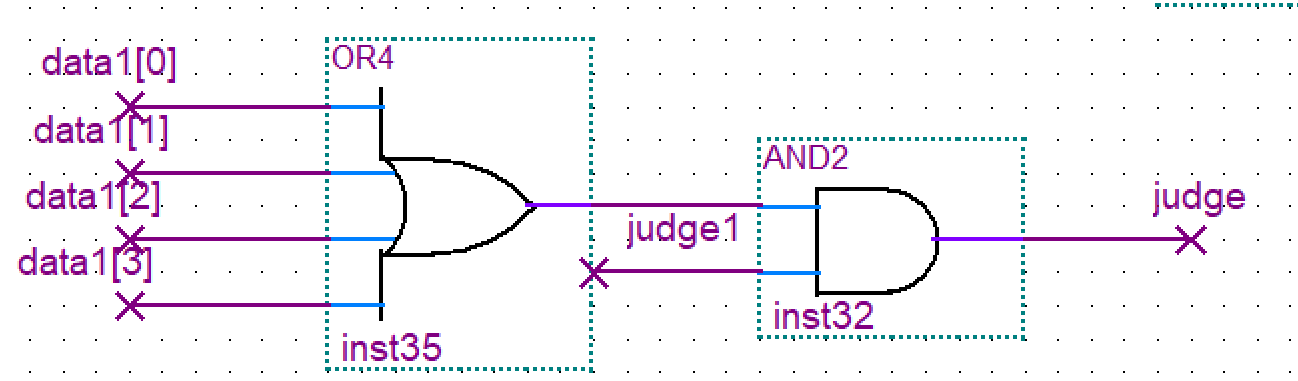


结果分别是49和7符合要求

将其与原本的计数器整合



验证时发现不管怎么算乘法始终为0，经过排查发现是因为judge会在输入✖时候变为1，这时候会自动计算，但此时另一个乘数始终为0，这就会导致乘法结果为0，所以说增加判定条件



如果另一个乘数始终为0，那自然结果为0，可以干脆不用计算，但是如果另一个乘数有值了，那自然和运算结果为1，这时候就可以开始工作了。

最终验证成果

**四、实验使用仪器设备（名称、型号、规格、编号、使用状况）**

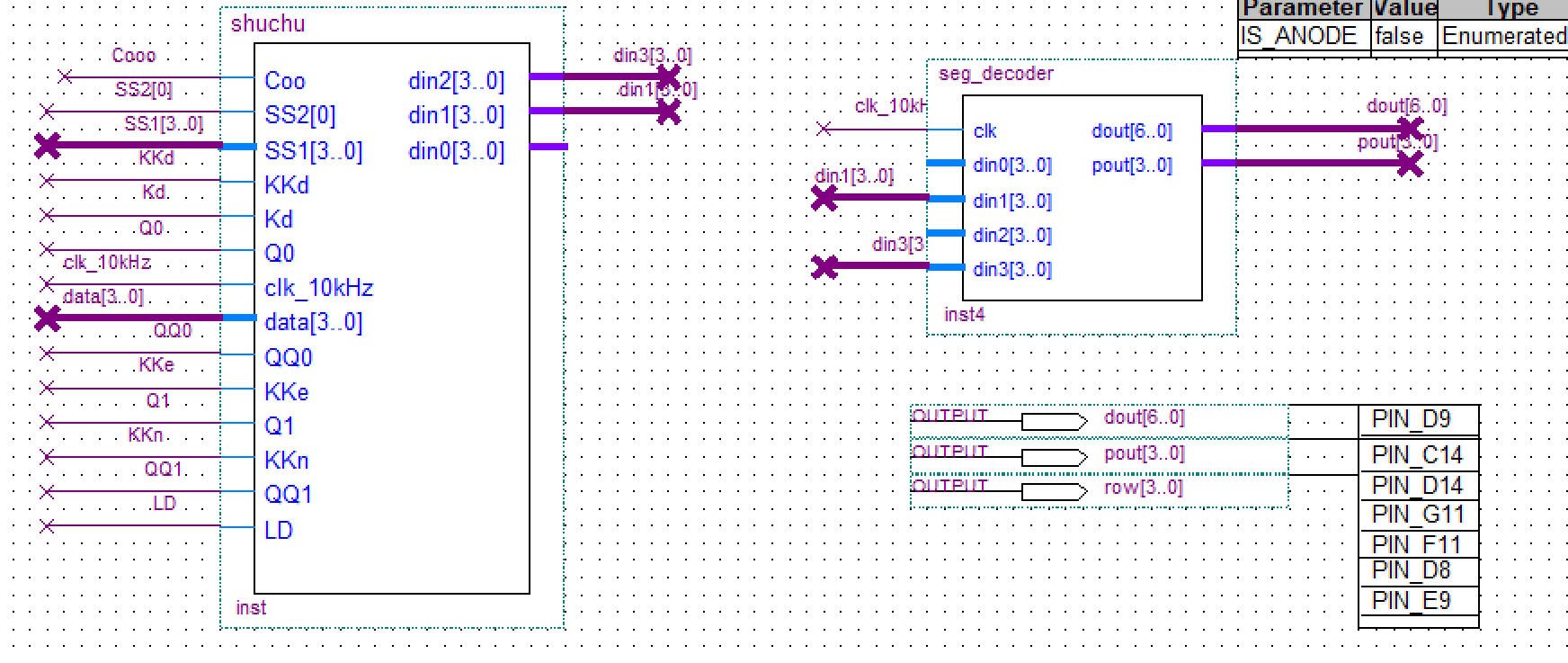
FPGA板EP4CE6E22C8N

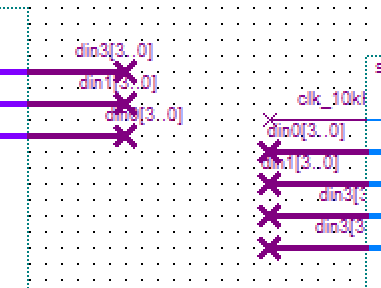
Quartus II

**五、实验总结**

本次实验遇到的一个令人印象深刻的问题是在数码管的输出部分，数码管能正常输出1位，2位数字，但是如果想输出3位数字就会显示错误，出现始终无法更正的现象，最终我采用了固定变量法和替换法，先是验证了3位数字，即符号位，十位数字和个位数字始终没有出错，再两两固定，发现均能正常显示，最终选择用两位数码管同时输出，最终解决了这个问题。这提醒我们不要太拘泥于自己的错误，要从另外的角度来思考。

错误验证方案（输出符号位和十位）：



最终解决方案：  


本次实验的目的是综合前面所学的各项内容，了解掌握数字系统设计的流程和方法，掌握复杂电路连接和调试技能。实验的内容是设计一个简易计算器，能够实现一位十进制数的加、减、乘运算，并用数码管显示结果。实验的原理是利用有限状态机的思想，将计算器的工作过程划分为四个状态，分别是初始状态、输入第一个数据后的状态、输入运算符后的状态、输入第二个数据后的状态。根据状态之间的转移条件和输出函数，设计了状态转换电路、显示存储电路、BCD加减器和串行乘法器等模块，并用FPGA板进行了仿真和验证。

通过本次实验，我收获了以下几点：

掌握了串行乘法器的原理和设计方法，学会了用移位寄存器等基本元件构建复杂的运算电路。掌握数码管显示的原理和设计方法，学会了用译码器计数器等基本元件实现数码管的驱动和显示。掌握了更深层次FPGA板的使用方法，学会了用Quartus II软件进行电路的绘制、编译、仿真和下载，用4×4键盘输入数据，用数码管显示结果，用示波器观察信号波形。

本次实验中，我也遇到了一些问题和困难，主要有以下几点：

在设计状态转换电路时，我没有考虑到输入等号后的状态应该返回初始状态，而不是保持在输出结果的状态，导致无法进行下一次计算。后来我发现了这个逻辑错误，并及时修改了电路。

在设计串行乘法器时，我没有注意到输入乘号后的状态应该判断另一个乘数是否为零，如果为零则不进行计算，否则会导致乘法结果为零。后来我发现了这个问题，并增加了一个判定条件，解决了这个问题。

总的来说，本次实验是一次非常有意义的学习经历，让我深刻理解了数字系统设计的原理和方法，提高了我的电路设计和调试能力，也激发了我的创新思维和动手能力。我希望今后能够继续学习和探索更多的数字系统设计的知识和技术，设计出更加优秀的数字系统。

**六、参考资料（预习、实验中参考阅读的资料）**

**《数字集成电路数据手册》，国防工业出版社**

**《数字逻辑与数字系统》，王银城等编著，清华大学出版社**

**《数字逻辑与数字系统设计》，陈宏等编著，高等教育出版社**