**实验2：组合逻辑部件设计实验报告**

221275027 喻思文

* 1. **译码器实验**

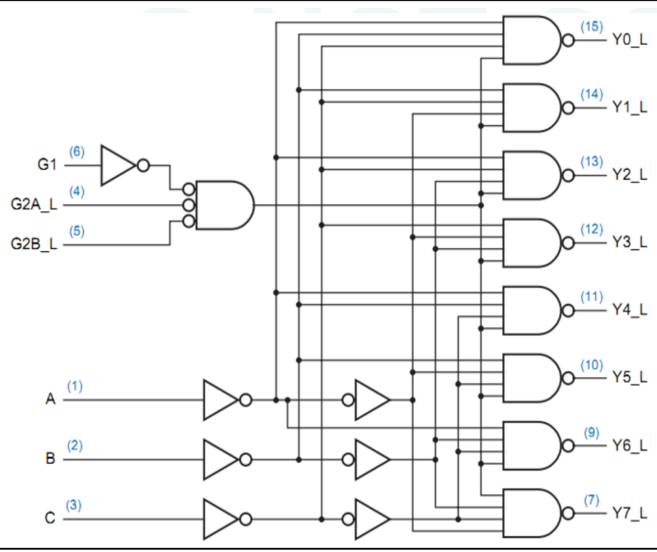
1. 设计原理

G1、G2A\_L、G2B\_L为使能端EN，G1为1，G2A\_L、G2B\_L为0有效；输入A、B、C为二进制编码，最高位为C、最低位为A；输出Y0\_L至Y7\_L可以视为数字0-7的指示位。比如ABC为011，EN=1时则对应Y3\_L=0，其他为1。

真值表如下：

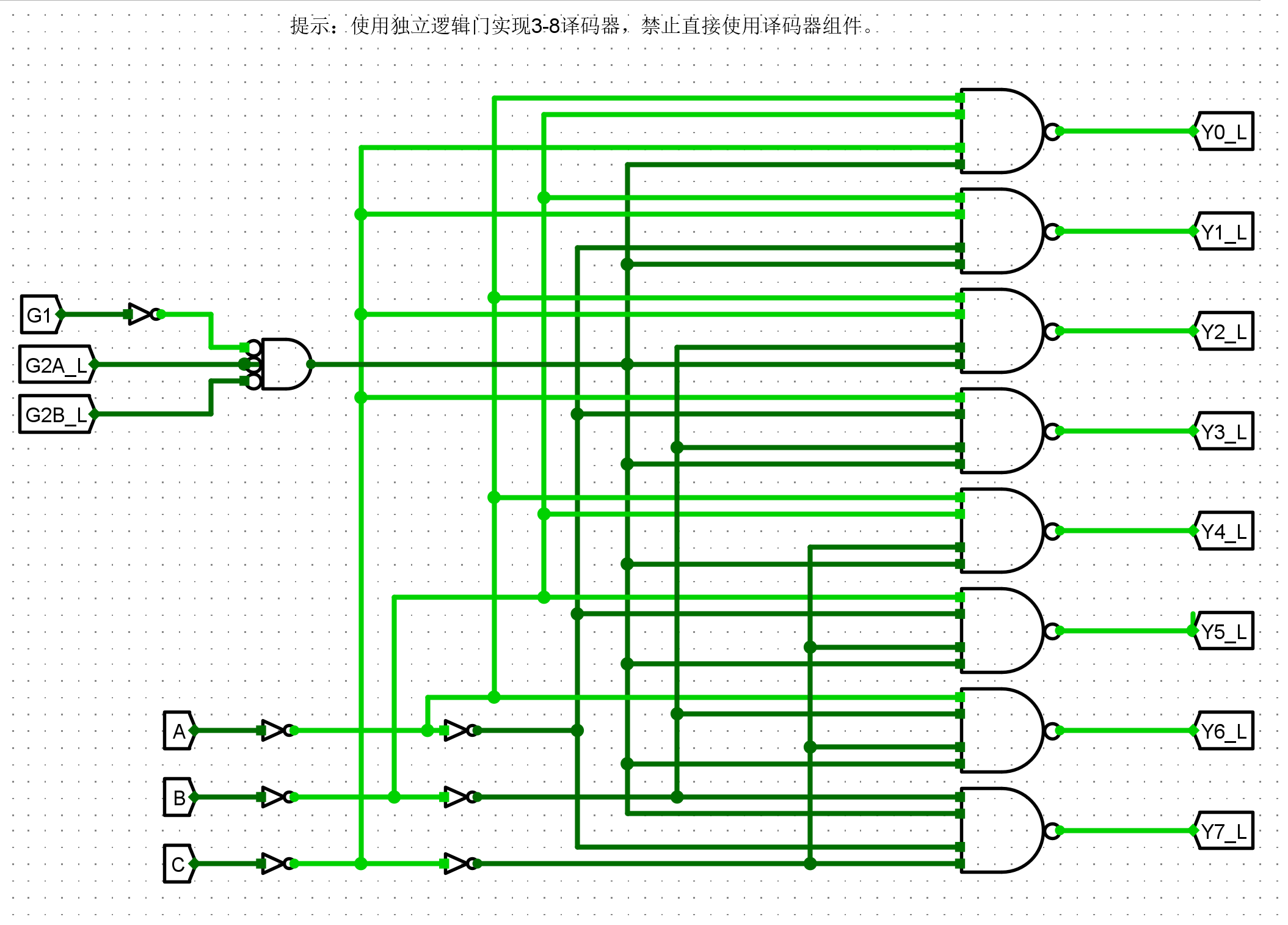
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| G1 | G2A\_L | G2B\_L | C | B | A | Y0\_L | Y1\_L | Y2\_L | Y3\_L | Y4\_L | Y5\_L | Y6\_L | Y7\_L |
| 0 | x | x | x | x | x | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| x | 1 | x | x | x | x | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| x | x | 1 | x | x | x | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

因此原理图如下：

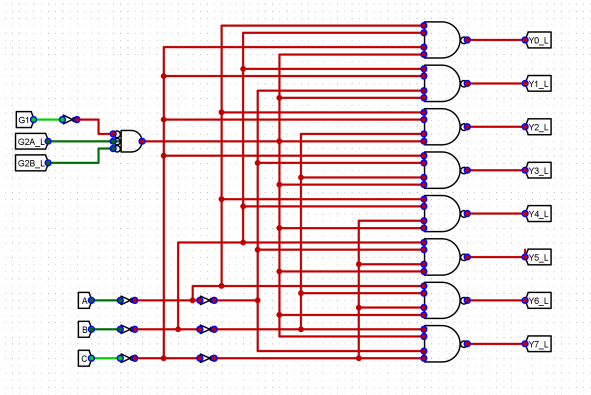
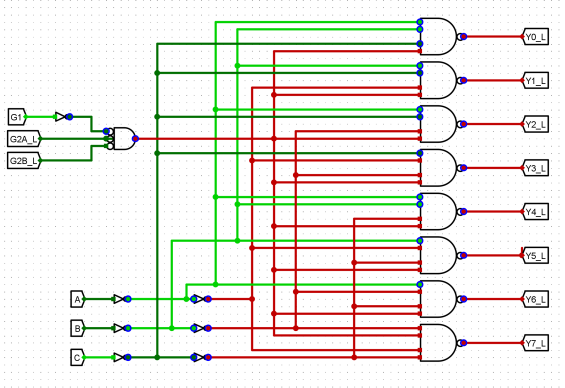


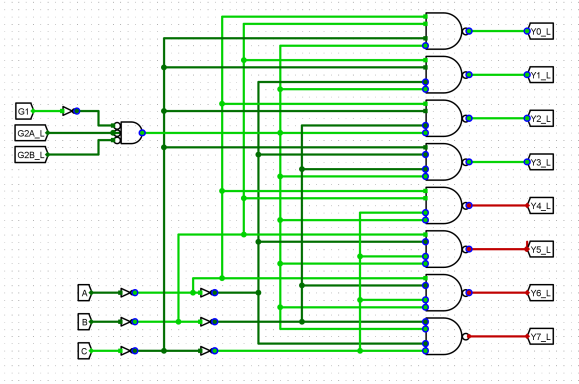
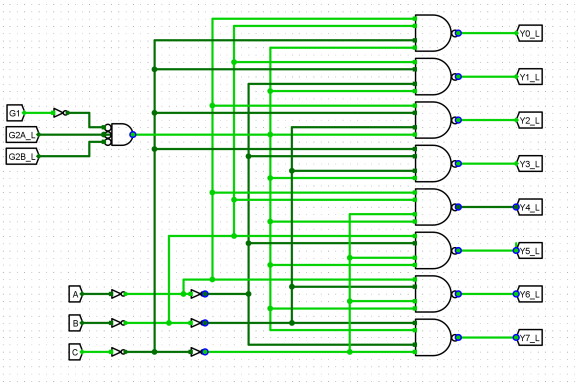
1. Logisim电路图

根据上面原理图，使用9个与门，6个输入端，8个输出端，得到下面电路图：



1. 单步测试

① ②

③ ④

输出正确，通过测试集，第一关结束。

* 1. **编码器实验**

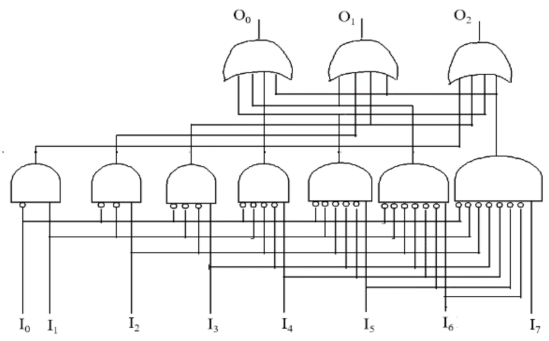
1. 设计原理

8-3优先权编码器是一个能够将8个输入信号进行编码，并根据优先级生成3位输出的电路。它可以根据输入信号的优先级，将其中优先级最高的输入转换成对应的三位编码。如果同时有多个输入信号被激活，那么只有优先级最高的信号会被编码器编码。也就是按照第一个1出现的输入端输出数位（优先级由高到低I\_0 - I\_7）。输出位由高到低是O\_0、O\_1、O\_2。

因此真值表如下：

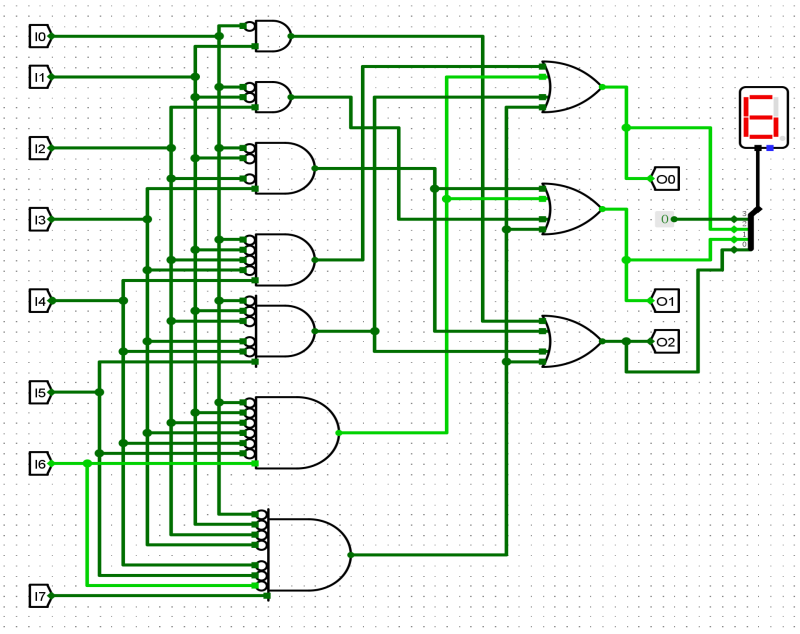
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| I\_0 | I\_1 | I\_2 | I\_3 | I\_4 | I\_5 | I\_6 | I\_7 | O\_0 | O\_1 | O\_2 |
| 1 | x | x | x | x | x | x | x | 0 | 0 | 0 |
| 0 | 1 | x | x | x | x | x | x | 0 | 0 | 1 |
| 0 | 0 | 1 | x | x | x | x | x | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | x | x | x | x | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | x | x | x | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | x | x | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | x | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

根据真值表给出原理图如下：



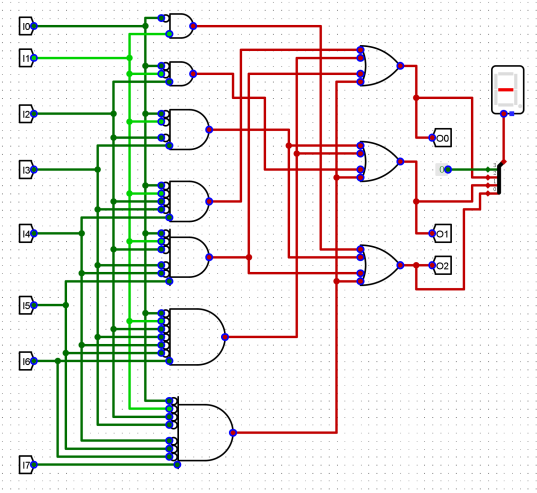
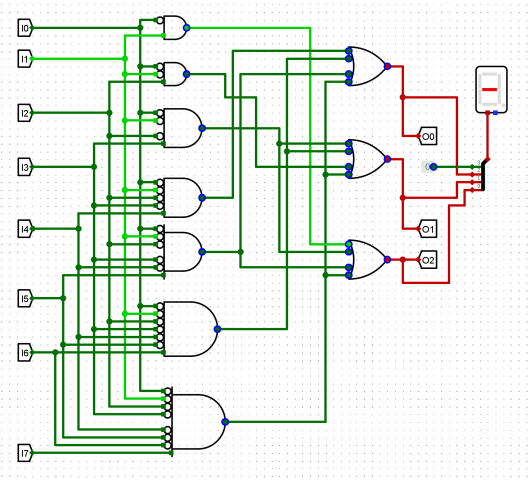
1. Logisim电路图

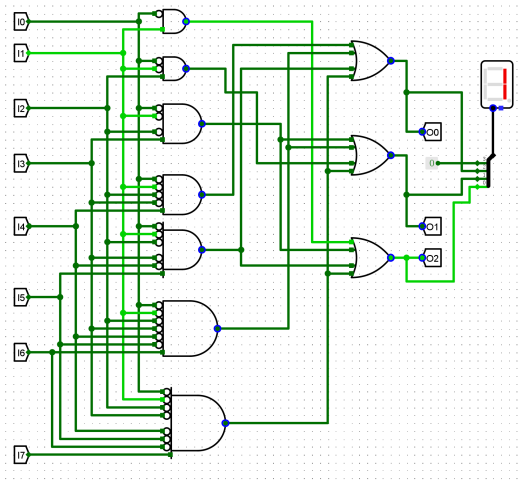
根据原理图，使用8个输入端，3个输出端，7个与门和3个或门，电路图如下：



1. 单步测试

如下：

① ②

③通过测试样例，第二关结束。

* 1. **4位加法器实验**

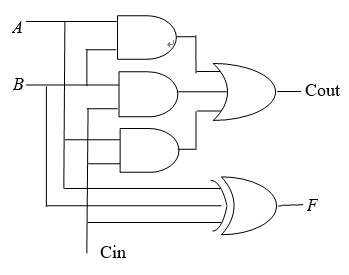
1. 设计原理

输入为加数、被加数和低位进位Cin，输出为和F、进位Cout，真值表如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Cin | F | Cout |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

根据真值表，有以下表达式：

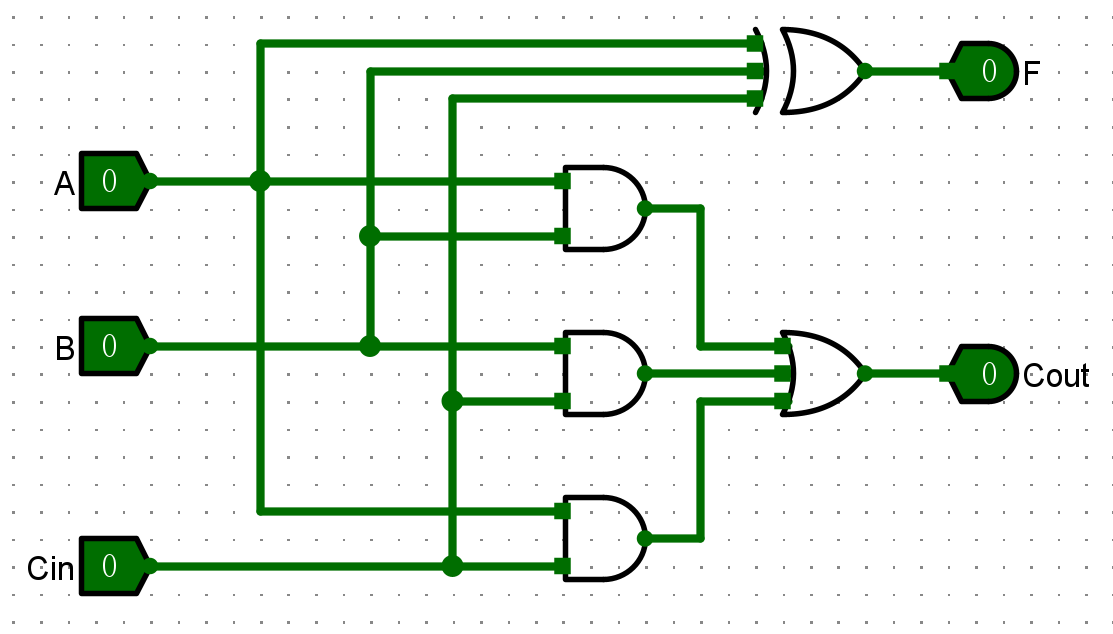
 因此，全加器原理图如下：



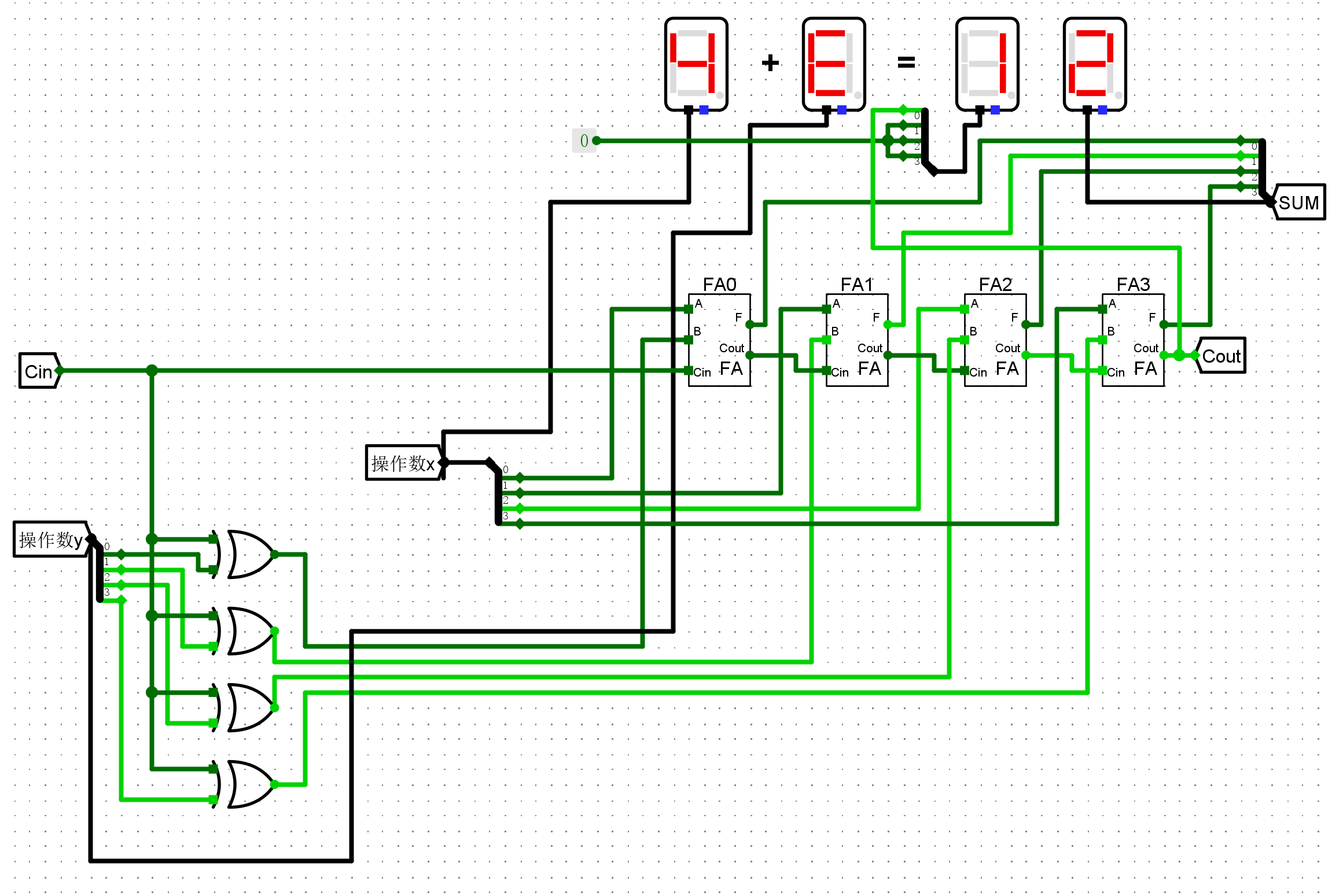
根据Cin输入值区分加减法运算，当Cin=0时，执行补码加法运算F=X+Y；当Cin=1时，执行补码减法运算F=X-Y。在减法中，补码是各位取反，最低位加1，这⾥我们就先各位取反，然后最低位加的1可以直接用Cin在最低位作为初始“进位”正常参与补码的加法就可以了，而Cin=0不取反，Cin=1取反Y这样的过程正如异或过程。因此对Y的每一位和Cin异或之后与X进行串行进位加法。整个加法过程，每一个加法器的Cout作为下一个的Cin判断进位，输出的F作为结果的一位。

1. Logisim电路图

全加器电路图如下：

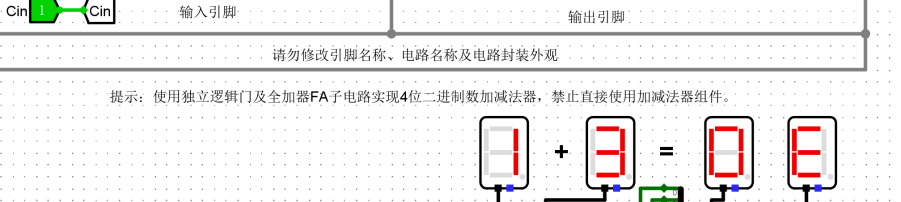
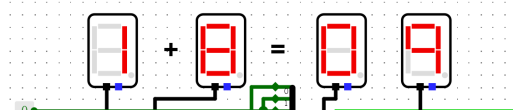


加法器如下：



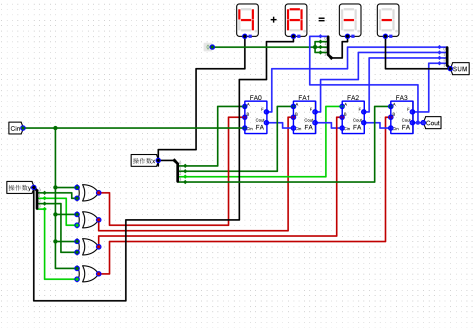
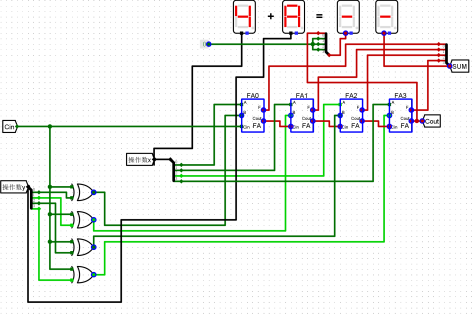
1. 结果展示与单步测试

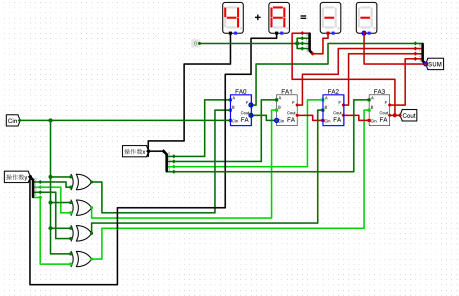
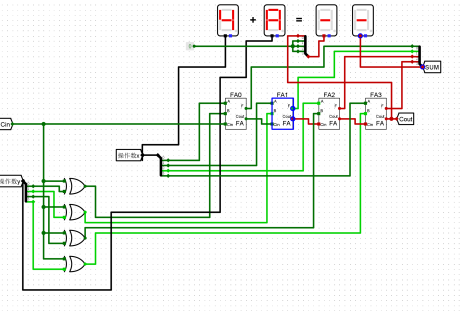
电路图已展示进位加法，下面分别展示非进位加法，结果为正和负的减法：

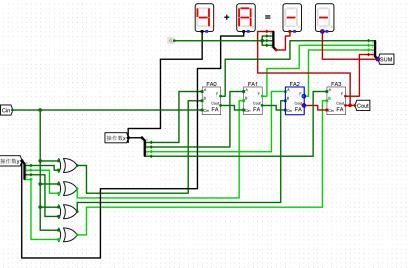
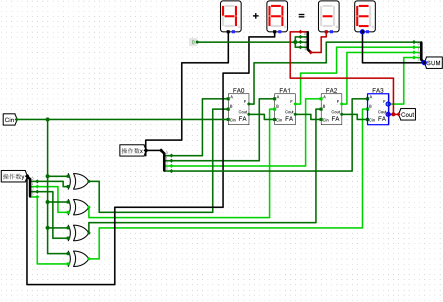


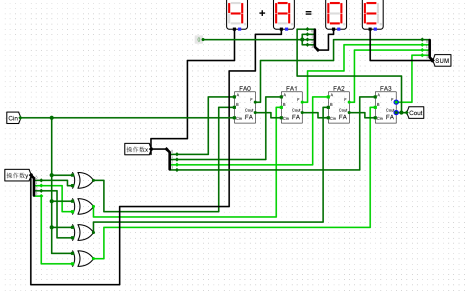
其中，1-3=-2，结果位第一位是cout，后面是2的补码E；5-3=2没有变负数，结果位第二个是2，第一个cout为1。其他标志位将在思考题处展示。

单步展示如下，省略部分加法器内部的过程单步。

① ②

③ ④

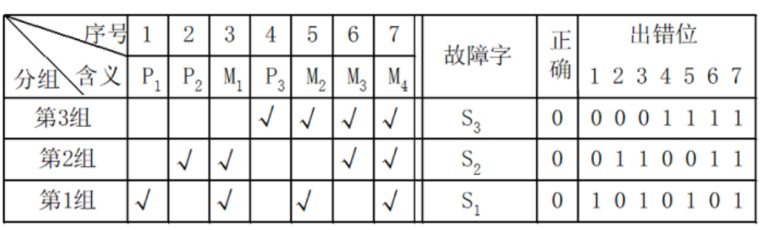
⑤ ⑥

⑦已通过所有测试点，第三关结束。

* 1. **汉明码纠错实验**

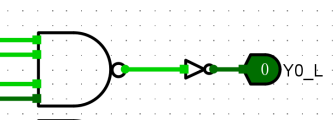
1. 设计原理

汉明校验的主要思想为：将需要进行检验或者纠错的数据分组，每组设置1位校验位P，校验位与组内的数据进行奇偶校验运算生成故障字S。若故障字为0，表示无错；否则故障字的数值就是出错位在码字中的位置编号（本实验4位数据，3位校验码，**用偶校验**）。除去 0 的情况，故障字的编码即可确定哪一位出现错误，例如，P1校验失败，P2、P3成功P3P2P1=001则可以看出出错在第1位。如下所示7位汉明码校验的例子：

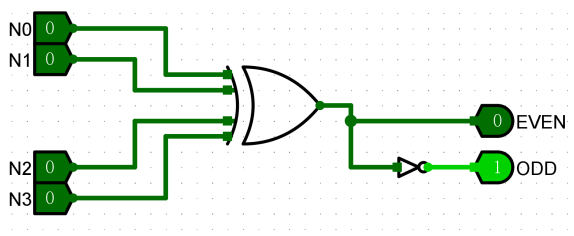


1. Logisim电路图

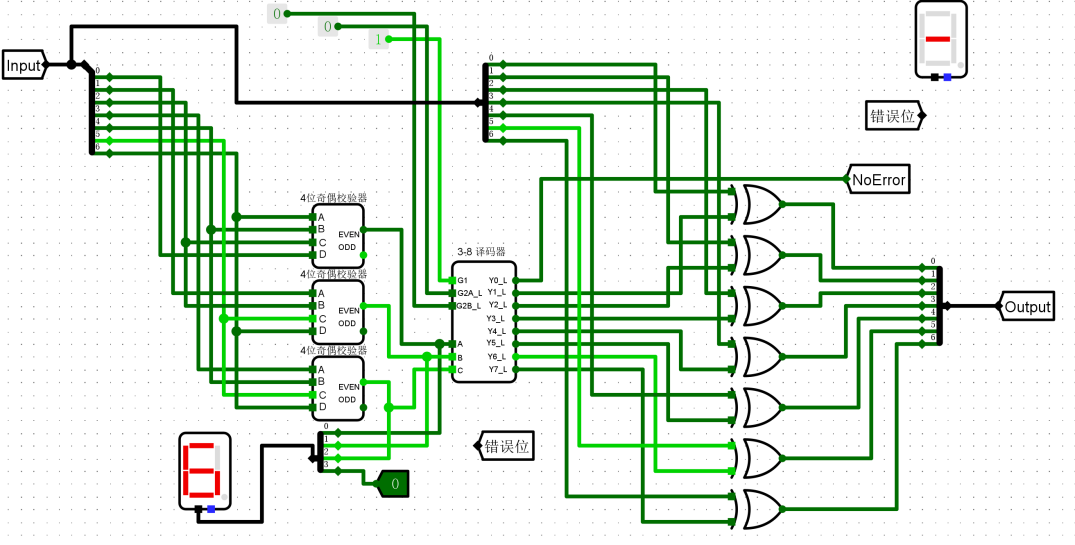
· 3-8译码器：与第一关基本一样，但是输出端再加一个取反（或者输出不取反）。**注意手册上给的是Y0，原始电路是Y0\_L，不要忘记取反。**



· 4位奇偶校验器：同实验1思考题，如下：



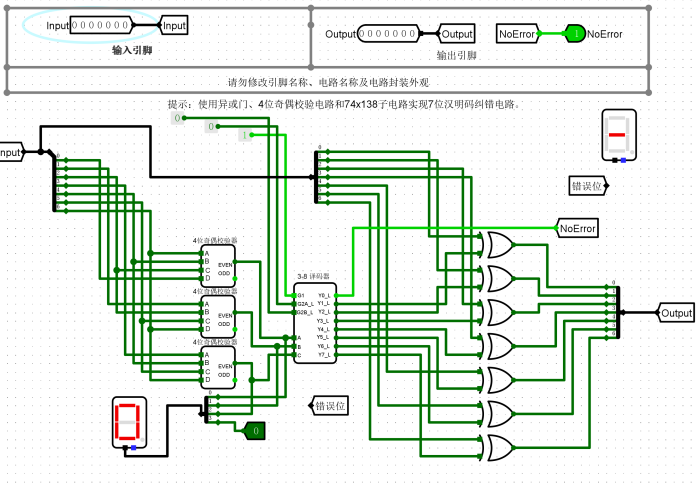
· 汉明码检验纠错电路如下，**注意手册中校验器odd端输出问题**：



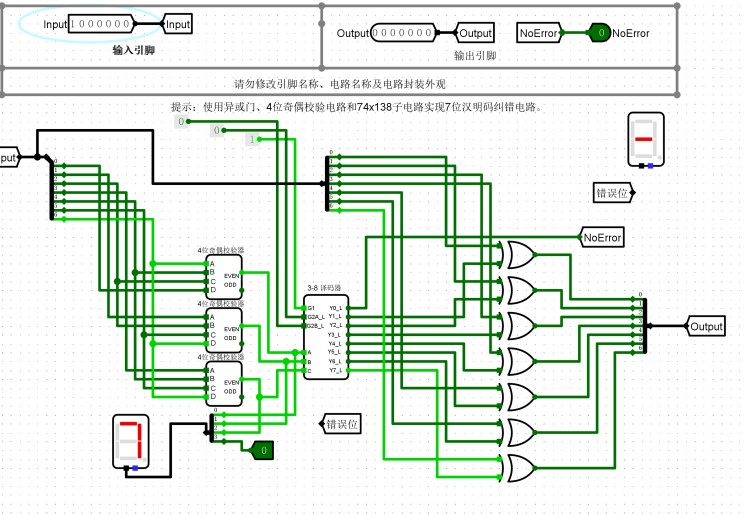
纠错过程就是出问题的位根据3-8译码器指出的错误标识1取反即可，否则不用变，异或运算。

1. 结果测试

初始状态输入0000000无错误：



改变一位，指出错误位并修正（错误标号为从右到左记）：



通过评测，第四关结束。

* 1. **8位桶形移位器实验**

1. 设计原理

· 注意到只给了三个多路选择器，然后8位数，因此可以考虑进行1位或2位或4位移动，因为任何0-7的数都能用3位二进制表示。移动位数shamt也是3位二进制，因此只需要在每个过程根据位判断是否进行该处的移动即可。

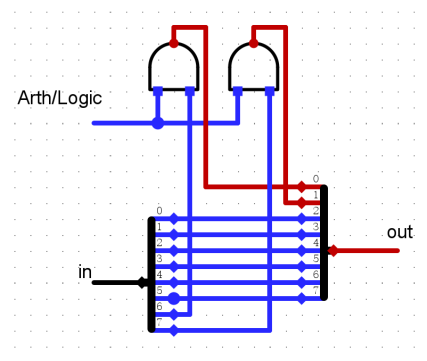
· 移动原理：

① 逻辑左移：高位移除，低位补0，A/L=0。

② 循环左移：高位移除，补到被移动的低位，A/L=1。

可以看出，为了补0或保留高位补上，高位与A/L做逻辑与即可。

左移2位的原理图如下，注意0-7是从低位到高位（也是左移方向）：

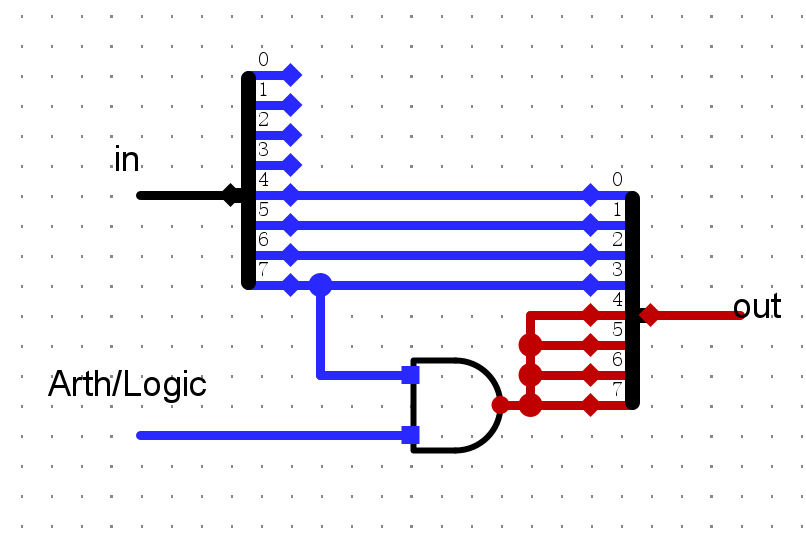


③ 逻辑右移：低位去掉，高位补0，A/L=0

④ 算术右移：低位去掉，高位补原最高位（符号位）的值，A/L=1

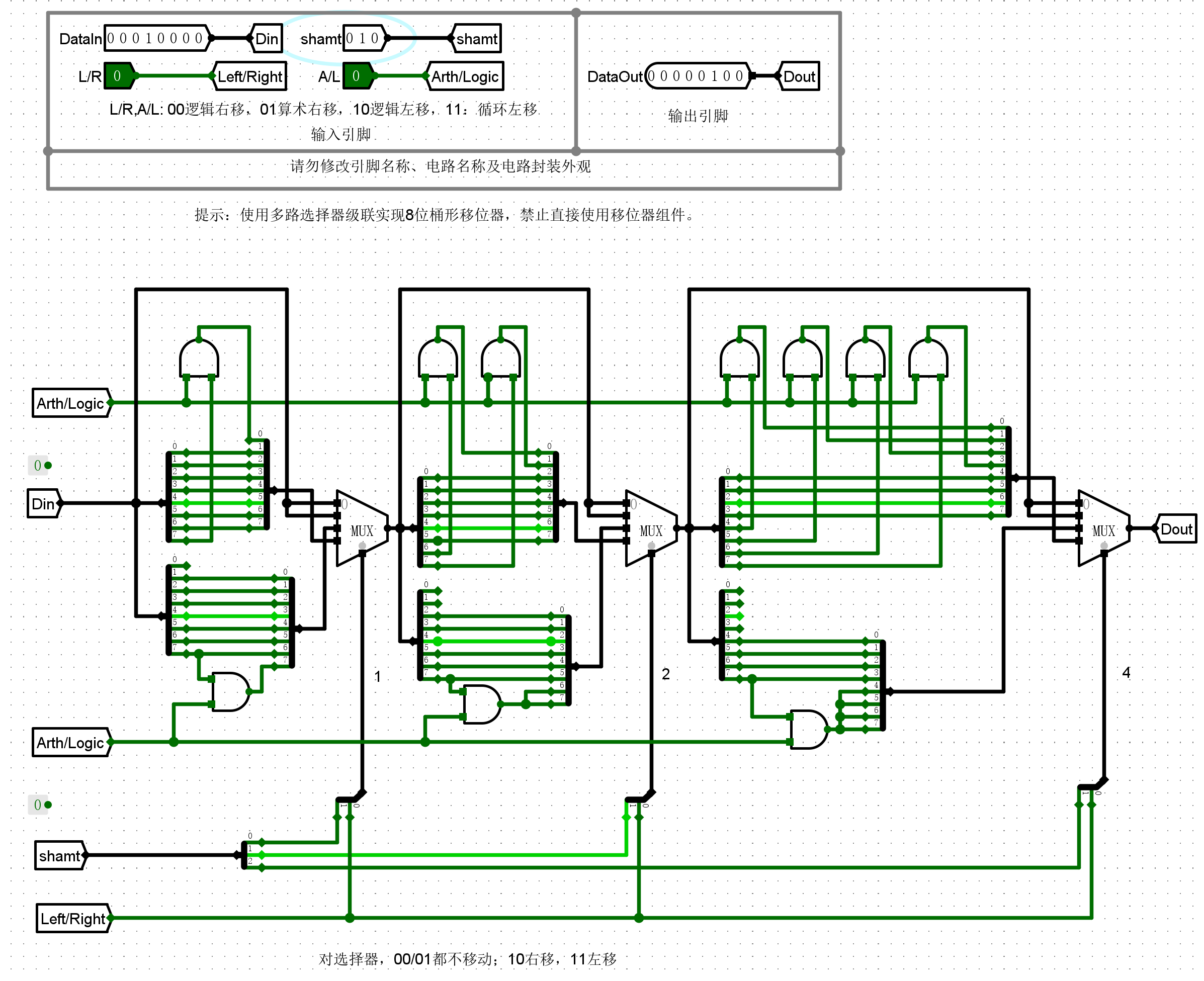
因此，还是高位与A/L值做与运算即可。

右移4位原理图如下，注意右移是高到低方向：



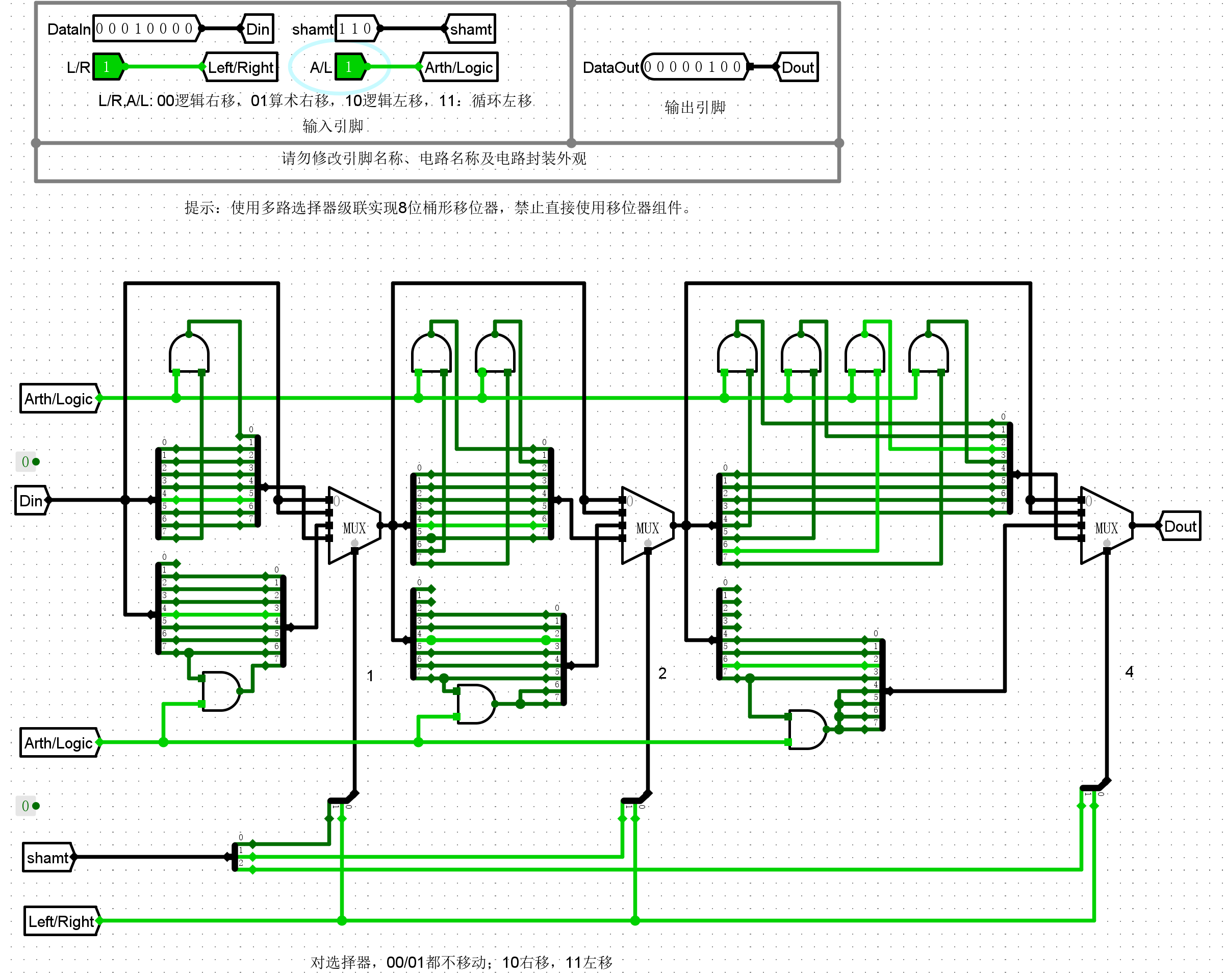
1. Logisim电路图

结合上面原理，电路图如下：

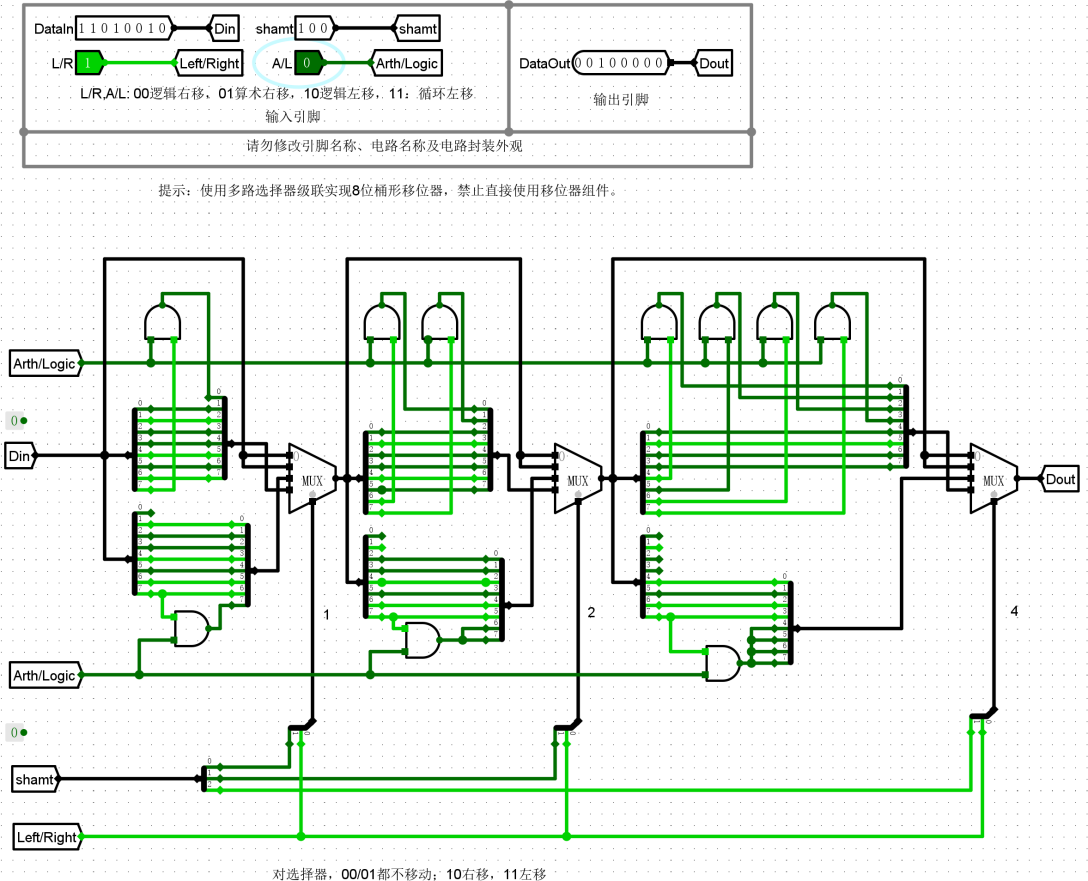


1. 结果展示测试

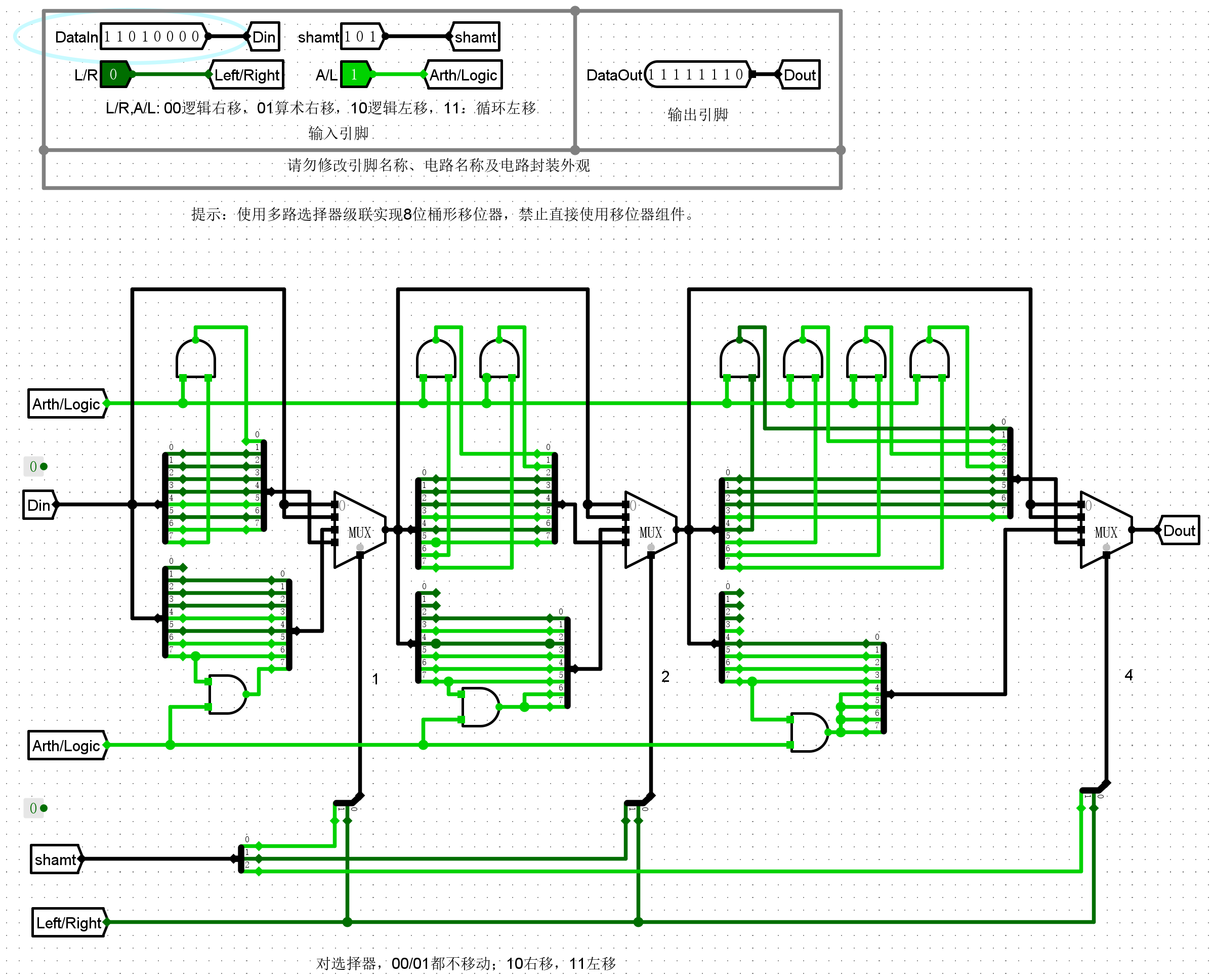
· 循环左移6位



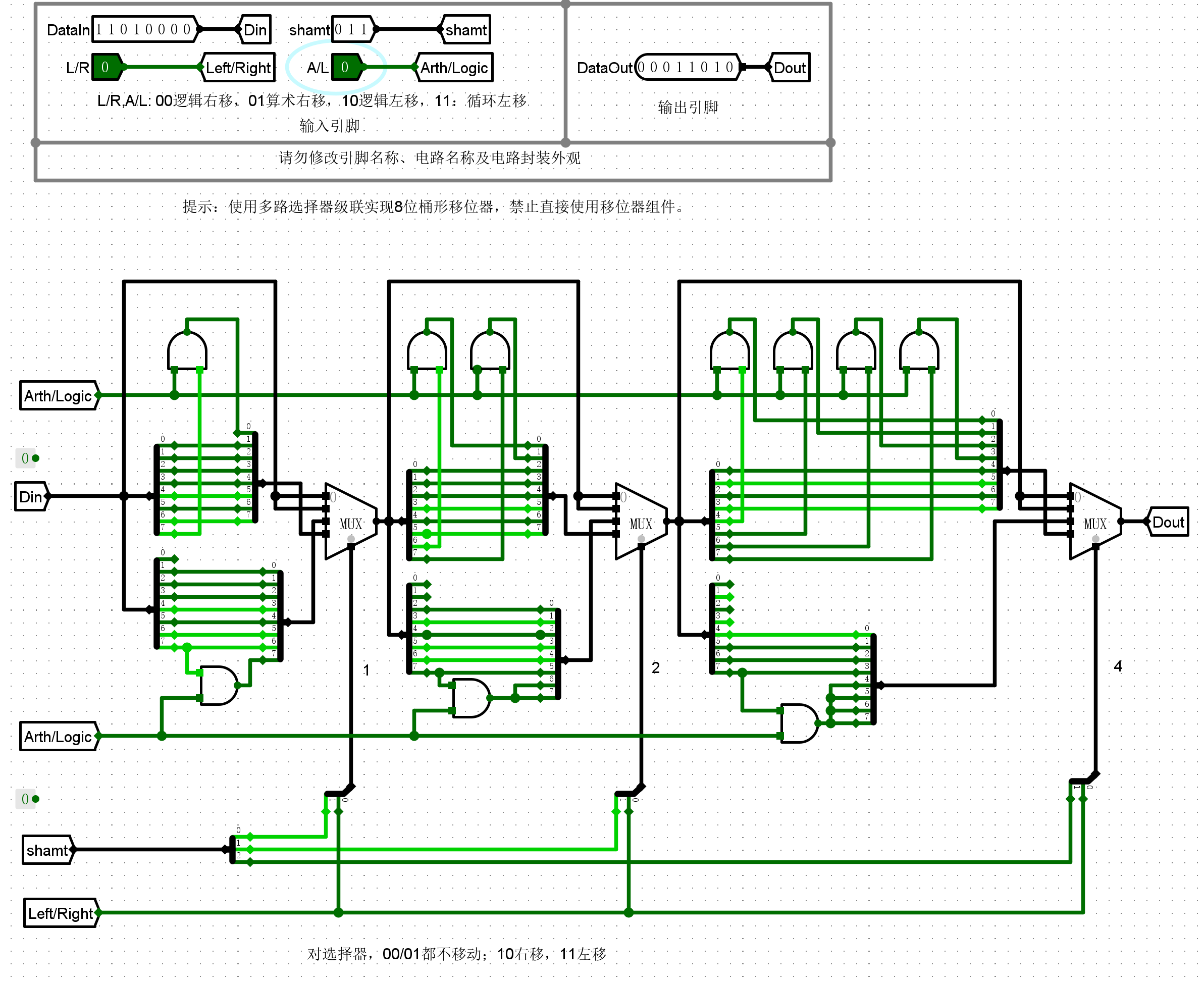
· 逻辑左移4位



· 算数右移5位



· 逻辑右移3位



通过评测，第五关结束。

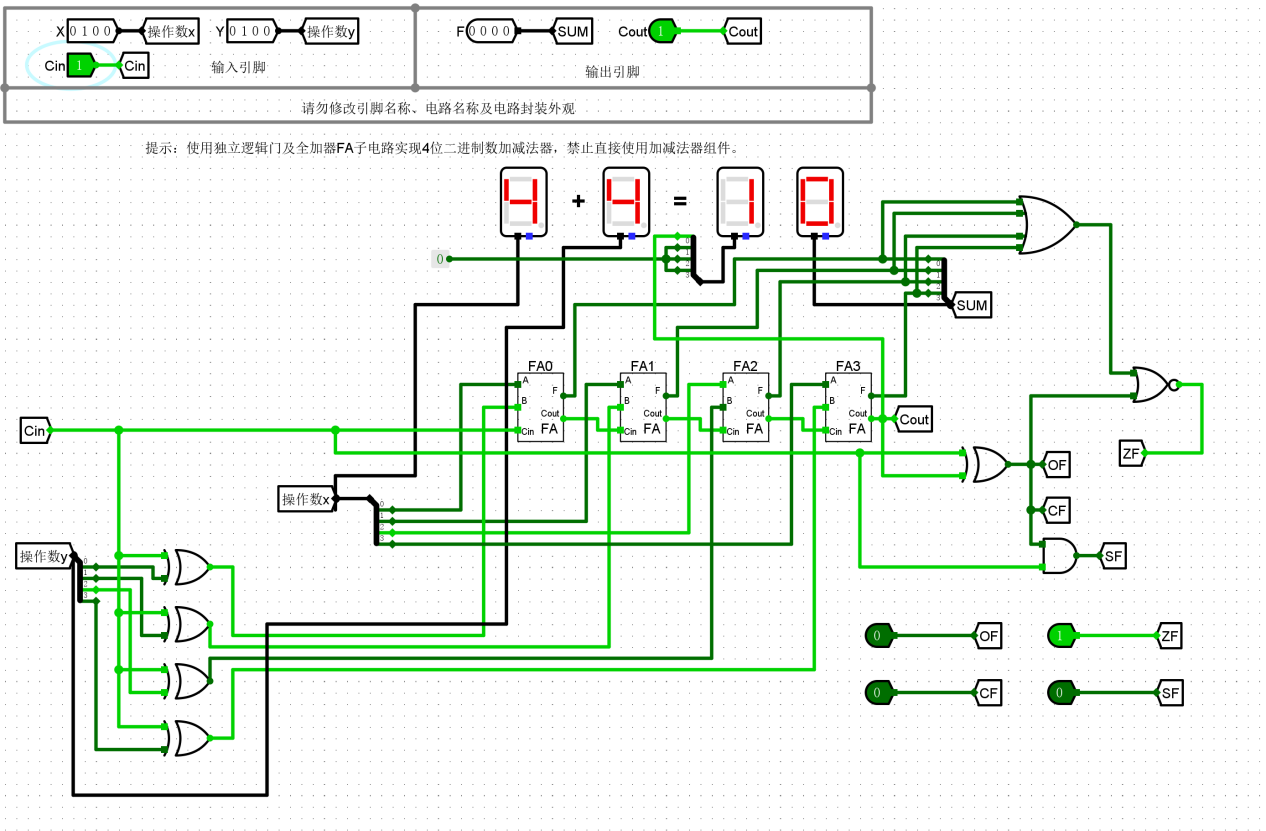
* 1. **思考题**

1. 修改实验中的加法器电路，生成进位标志CF、溢出标志OF、符号标志SF和结果为零标志位ZF。

在本次实验4位加法器中，Cout表示加减法器的进位/借位输出。

注意到测试点减法Cin=1时，结果为负数时Cout=0，否则Cout=1。加法下结果超过15，Cout=1。可以看作0-15的范围，超过这个范围均视作溢出。因此，CF按照教材，因为这里含有溢出，ZF要在F=0基础上CF也要为0。OF直接看CF，SF输出1为负数，此时Cin=1，Cout=0（但还要保证其他均为0，注意到OF一定为1，而且在这个电路中减法不溢出，一定是正数）。因此，各标志计算公式如下：

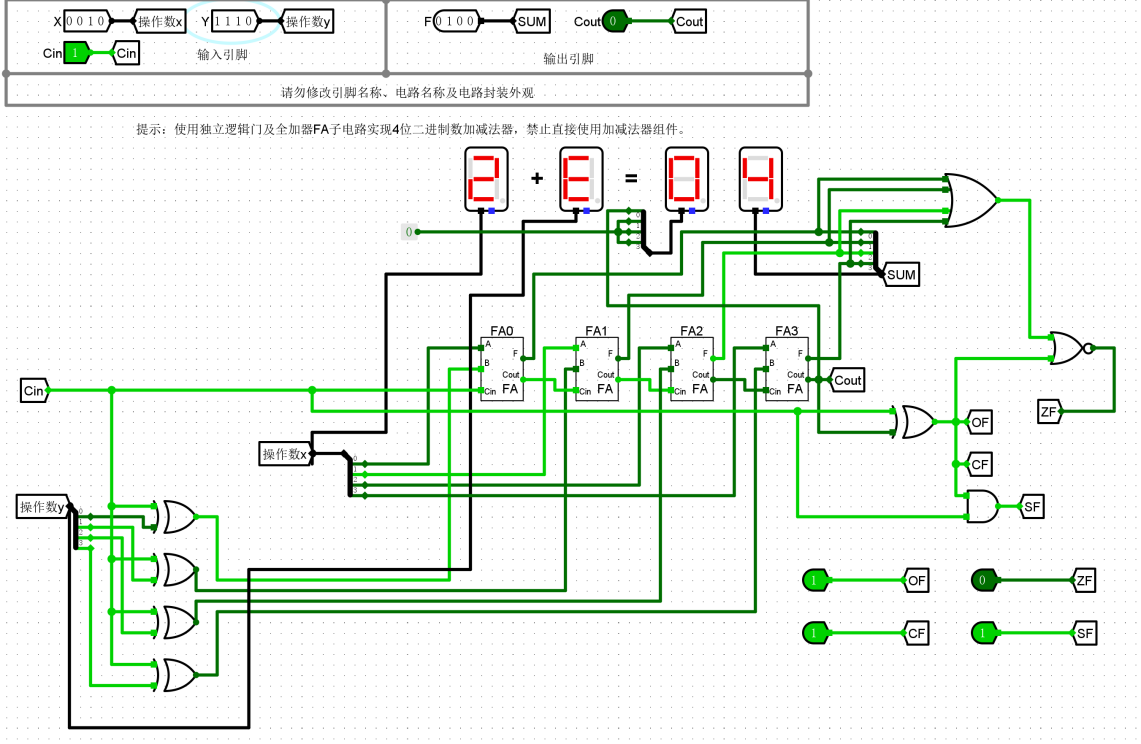
 因此电路设计如下：



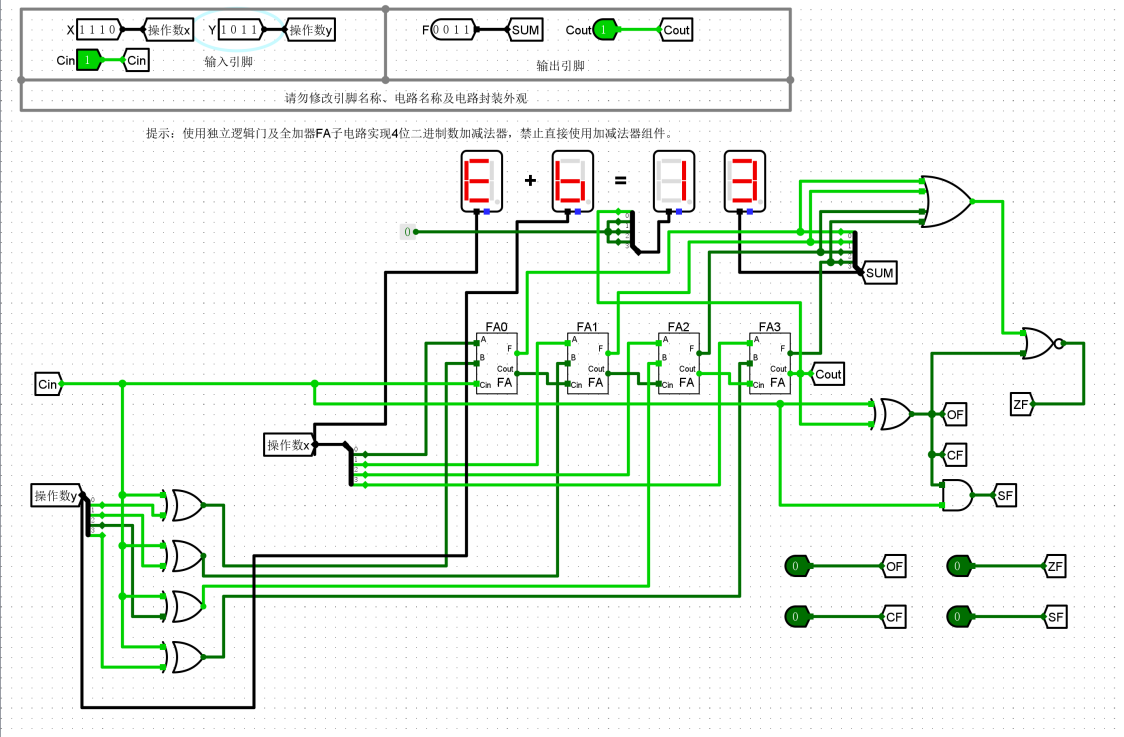
1. 在执行比较指令时，通常使用减法运算后，判断标志位的方式来实现，试通过上述加法器实验举例说明判别的方法。

直接对XY做减法，F=X-Y，若标志符号SF=1，说明X<Y，反之X>Y。

实验如下，这里X=2，Y=14，可以看出，SF=1，为负数，X<Y。



再来一个X=14，Y=11，这时SF=0，正数，X>Y。



显然，ZF=1说明X=Y

1. 如何使用8位桶形移位器扩展到32位桶形移位器

在8位桶形移位器基础上，修改输入、输出、多路选择器位数到32，shamt位数为5，增加两个多路选择器。为了方便连线，对分线器以及与门进行数据位宽加倍，基本思路与8位的一样，除了位宽处理（事实上，8位的在移动2、4位处也可以增加数据位宽简化连线）。如下图所示（电路在lab2.5.circ中作为副电路了，命名为32-buck）：

