**实验 4：运算部件设计**

221275027 喻思文

**4.1 4位先行进位加法器 CLA**

1）实验原理

· 先行进位部件CLU

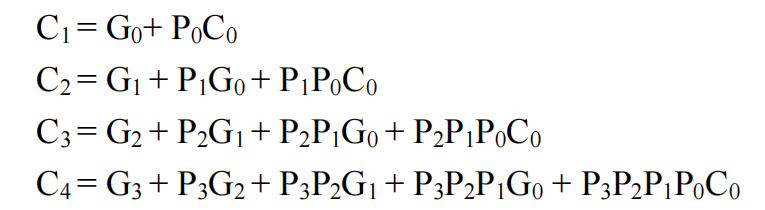
手册中已给出逻辑表达式：

进位传递函数：

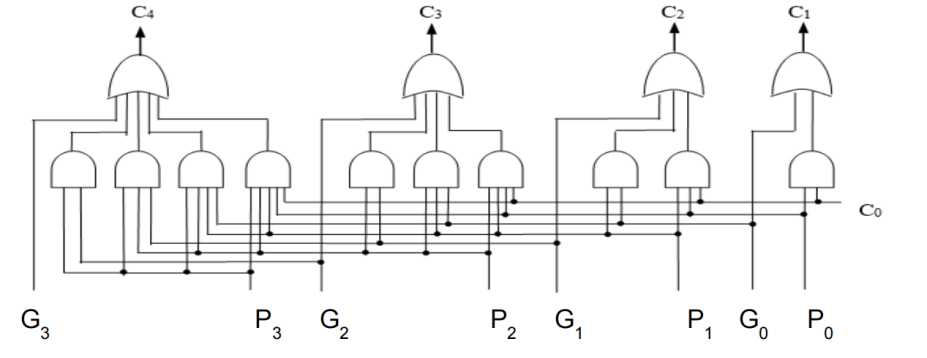
进位生成函数：

此部分并入在全加器中。

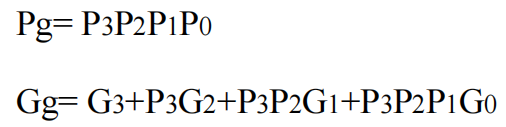
列入到4位二进制数加法器进位的逻辑表达式中，可以得到以下 4 个先行进位的逻辑表达式，下面表达式可以看出，仅与和有关，相互间的进位没有依赖关系。只要和同时到达，就可几乎同时形成，并同时生成各个数位的和。



因此原理图如下：



采用组内和组间都并行的进位方式。为了实现组间并行，需要在先行进位部件中输出组间进位生成函数Gg和组间进位传递函数Pg，表达式如下：

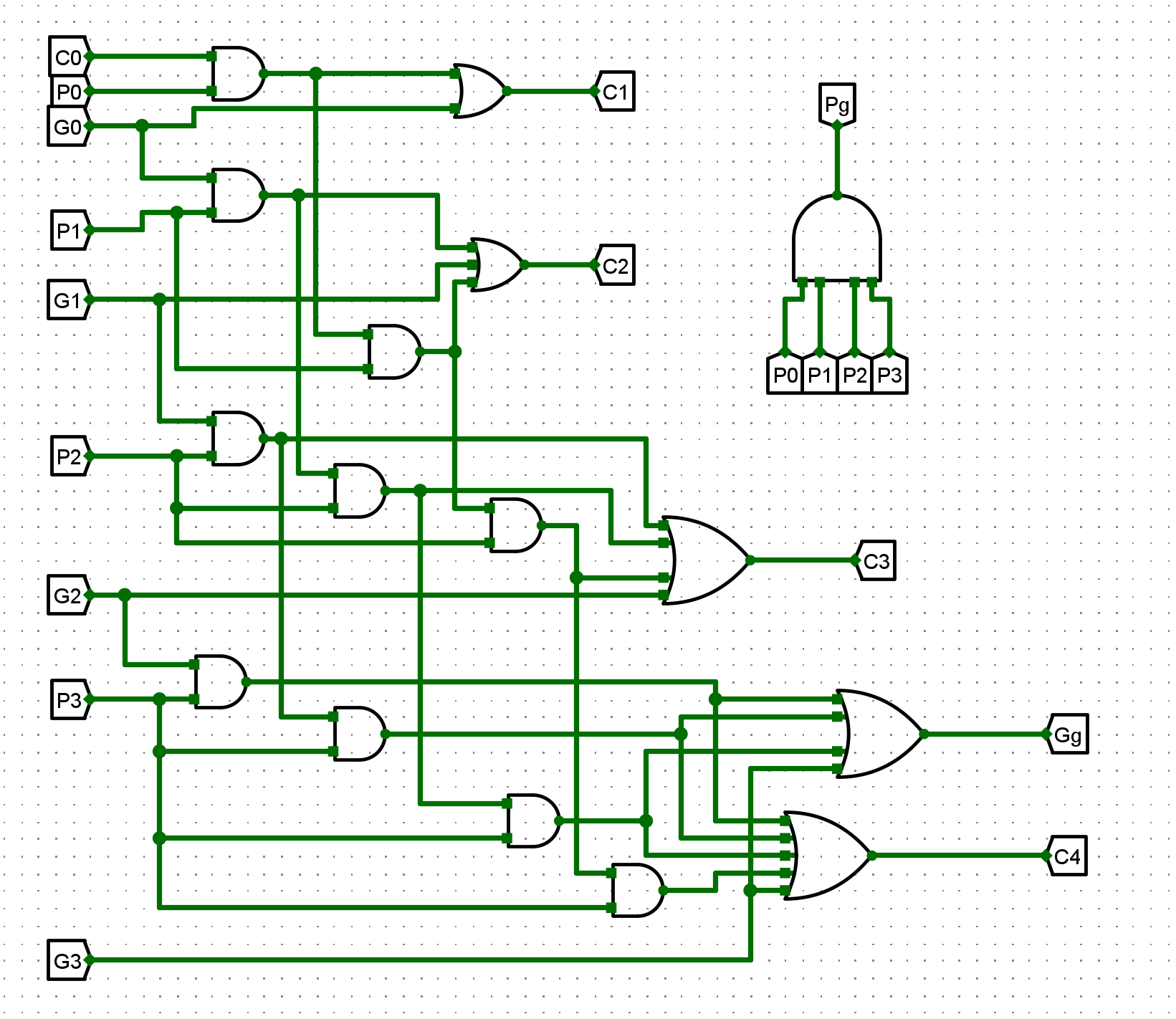


· 加法器CLA

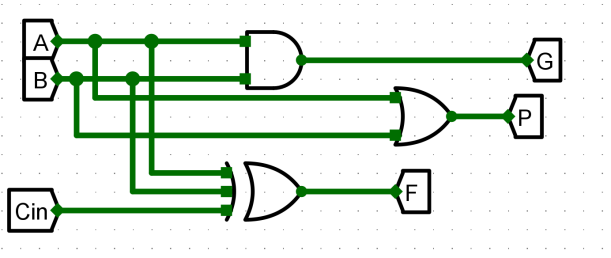
先行进位部件接受低位的全加器得到的进位传递和生成位，得到进位C传递给下一位全加器，作为新的Cin，如此下去，每一个全加器的F为输出位。

1. Logisim电路图

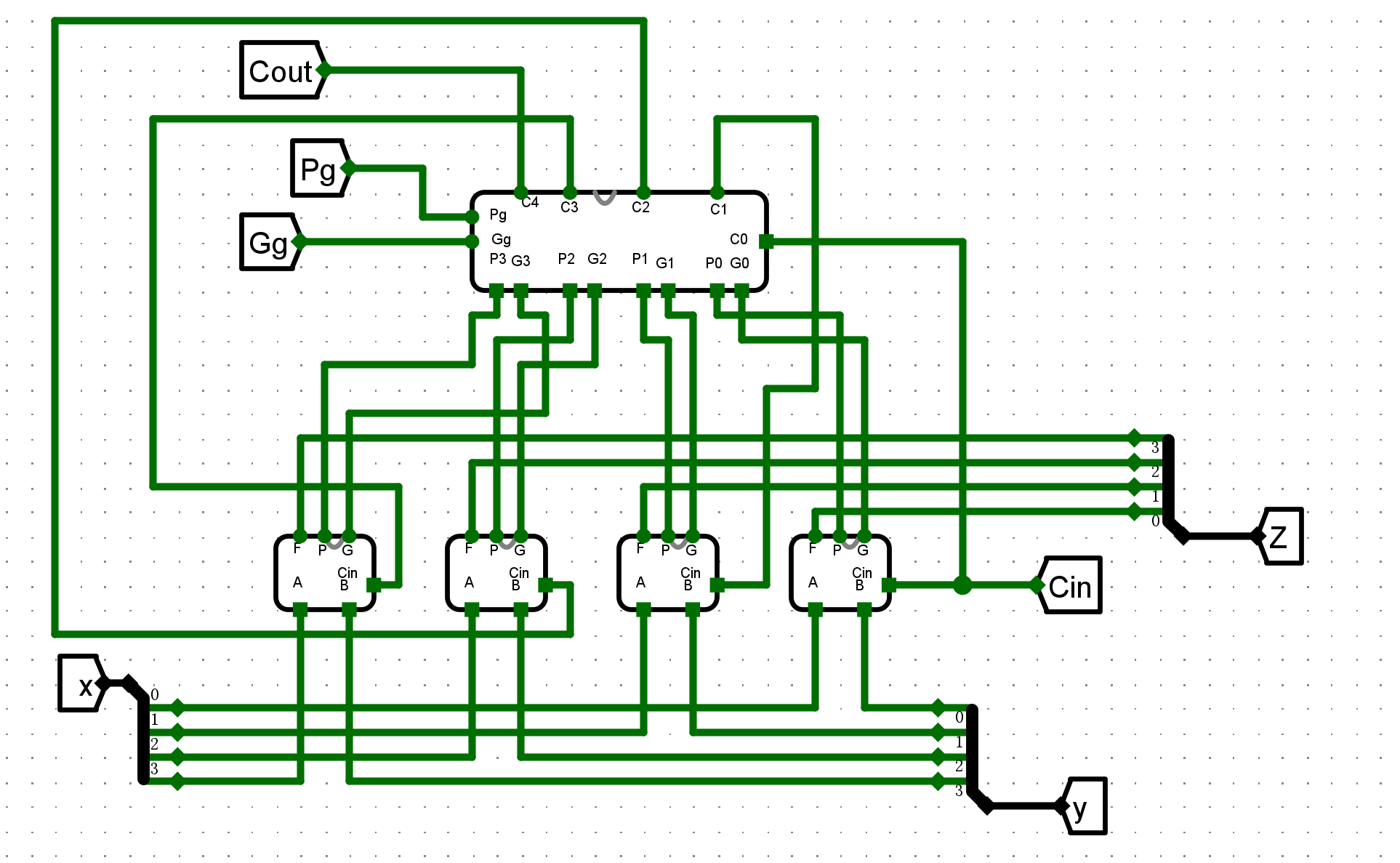
根据原理图和表达式，Logisim电路图基本与原理图相同，CLU如下：



加入进位传递和生成的全加器如下：

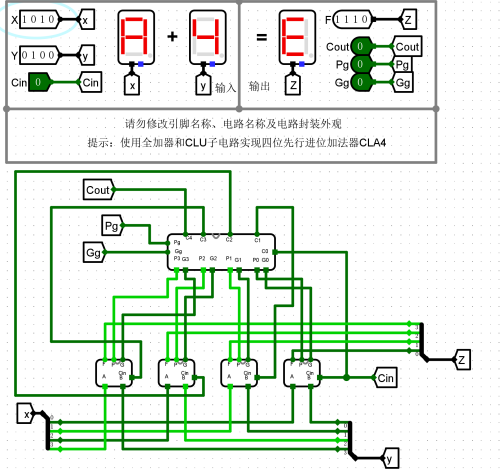
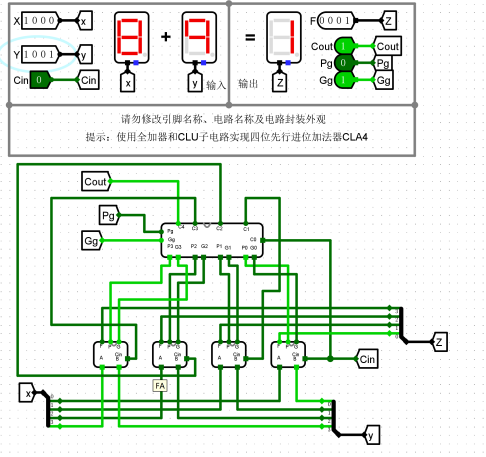


最终CLA加法器如下：



1. 实验验证

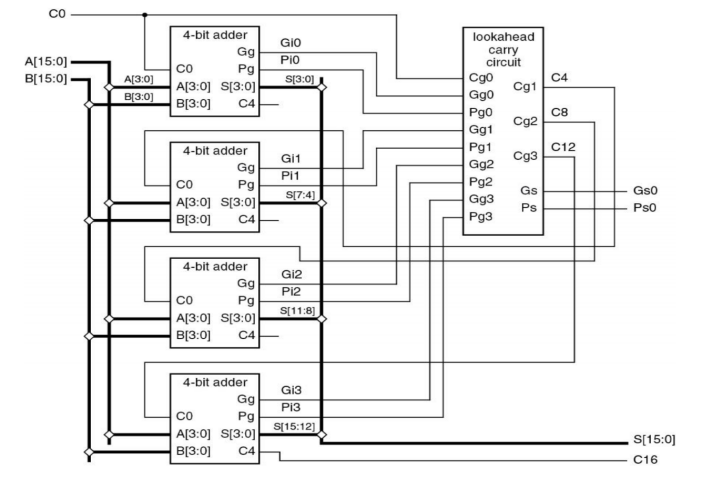
如下两图所示结果正确，发生进位情况也能正确表示：

**4.2 16位两级先行进位加法器实验**

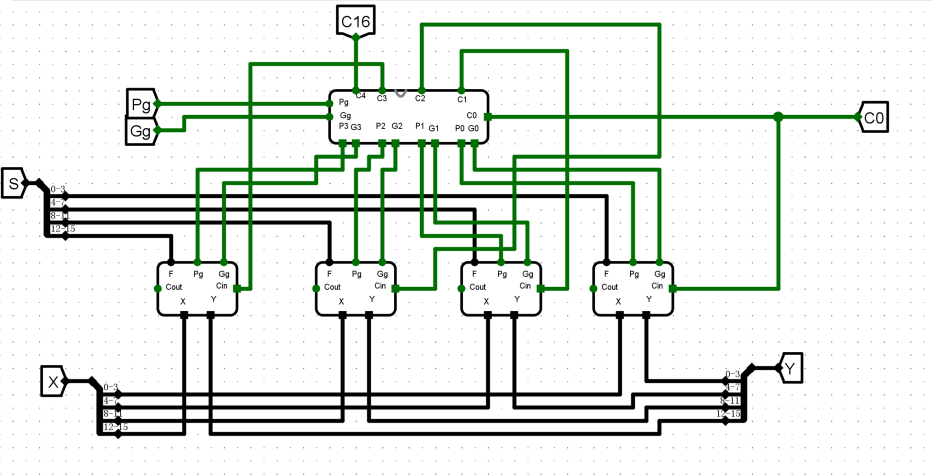
1）实验原理

对于一个16位加法器，可以分成4组，每组用一个4位先行进位加法器CLA实现。实际上CLA和第一关的全加器作用基本相近，故电路整体涉及与第一关基本一致。下面是原理图：



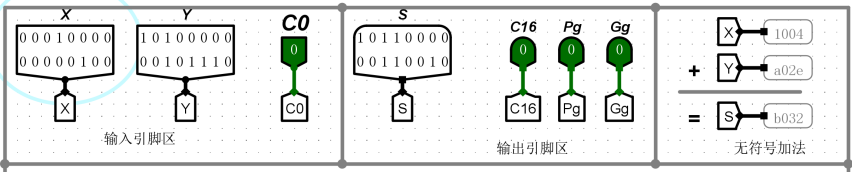
1. Logisim电路图

按照原理图，电路图如下：

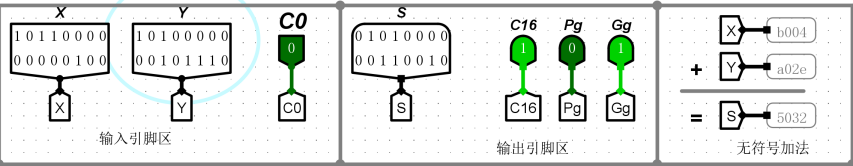


1. 实验验证

常规加法：



高位进位：

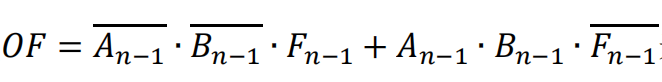


**4.3 32位快速加法器构建实验**

1）实验原理

电路构建级联难度不大，通过将两个 16 位两级先行进位加法器串行级联构建一个 32 位加法器即可。

这次需要输出一系列标志位：

溢出标志：

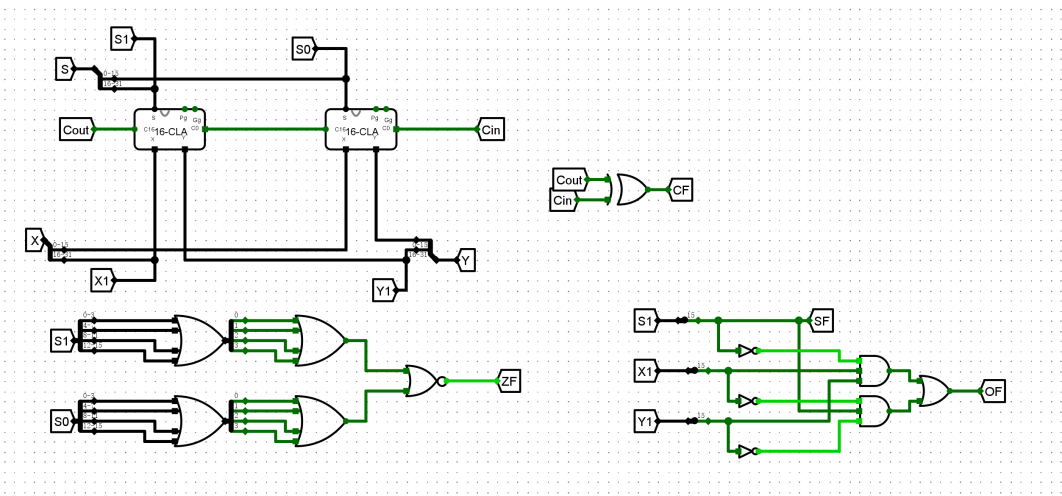
结果为0标志位ZF的可采用分组分级进行或运算的方式获取，避免扇入输入数量过多的问题，因此分成2个16位，每组4位4位或，最后输出或运算结果。

进位标志CF=Cin⊕Cout

符号标志SF=结果最高位

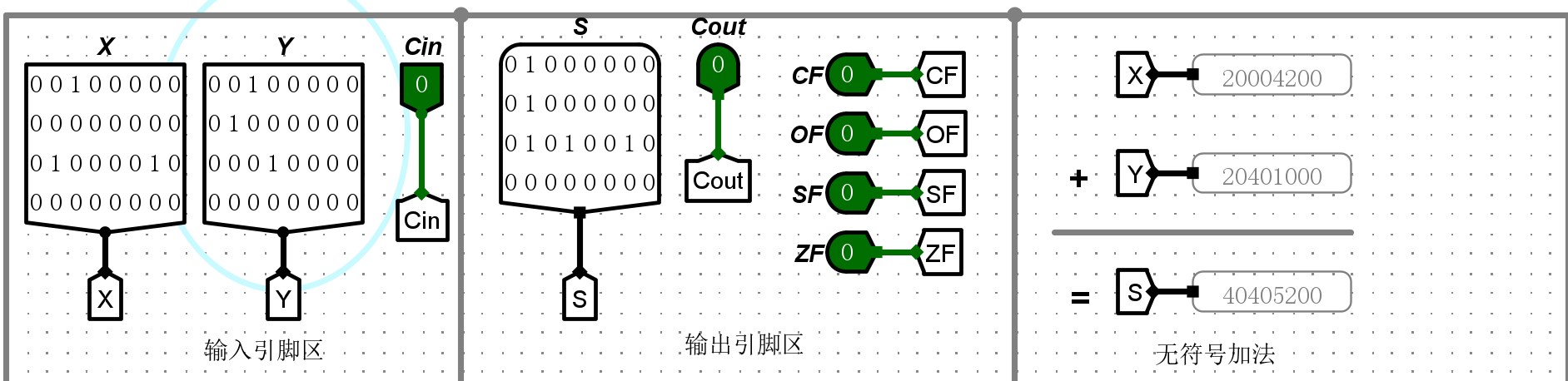
1. Logisim电路图

因此，按照上面原理，电路图如下：

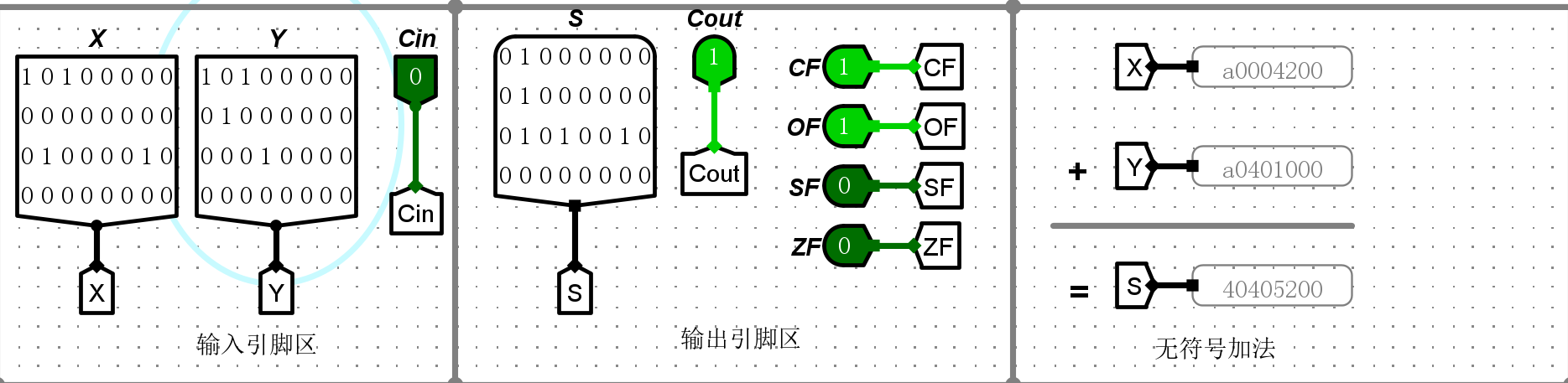


1. 实验验证

常规加法：



高位进位（溢出）：



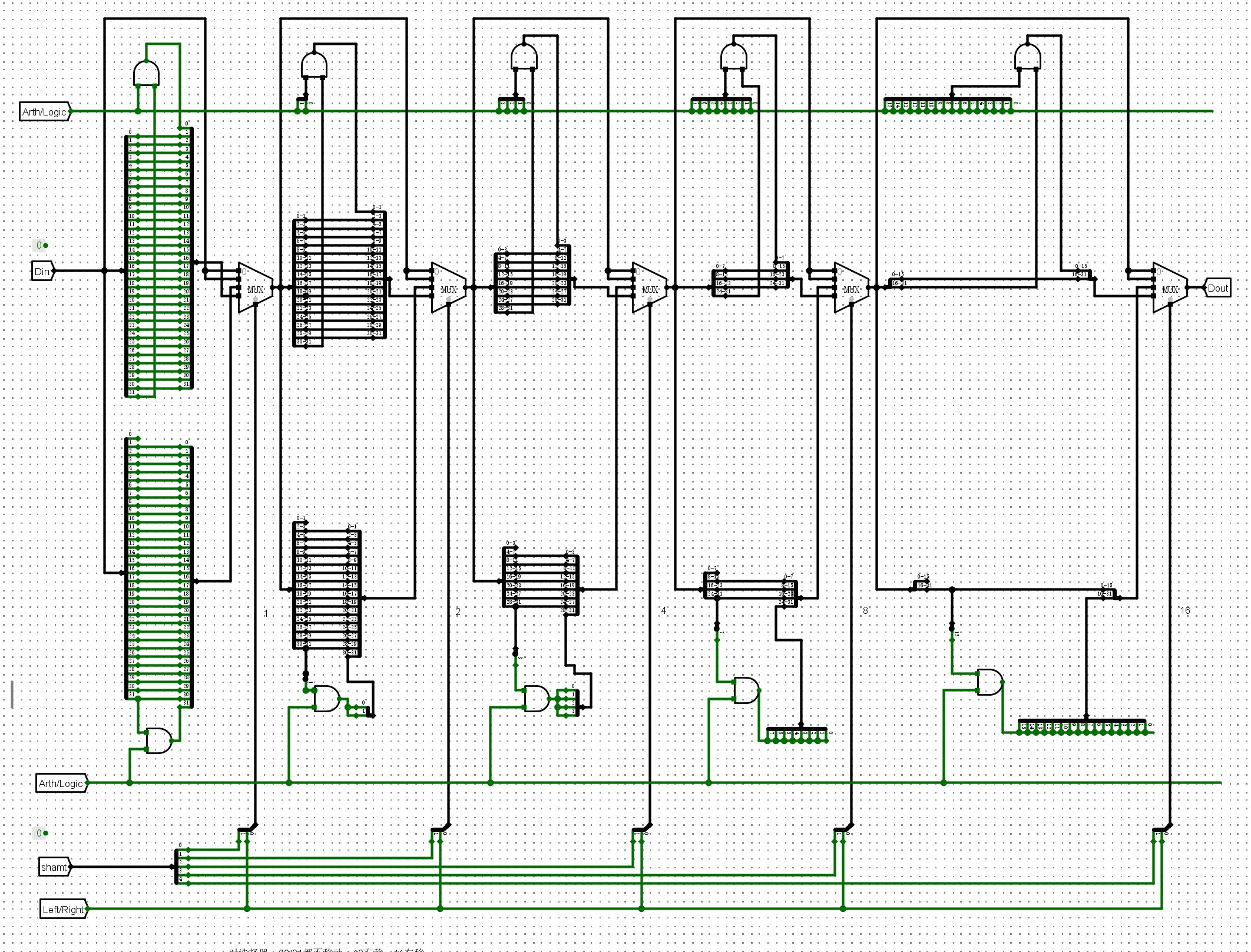
**4.4 32位桶形移位器设计**

1）实验原理

与8位的基本类似，也在实验2的思考题中提及，用分线器移位。

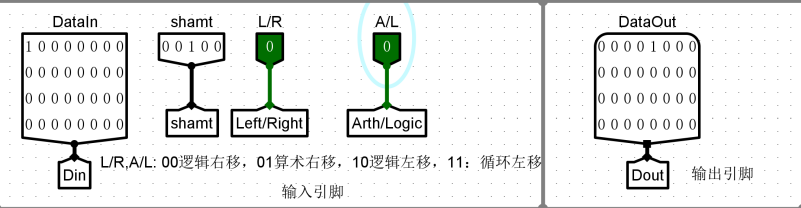
1. Logisim电路图

下面是实现电路：

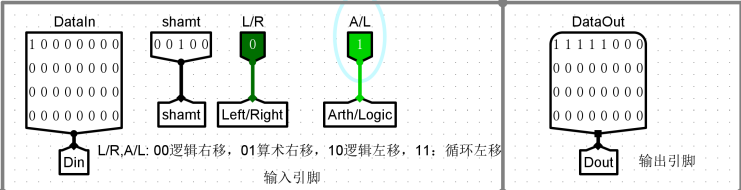


3）实验验证

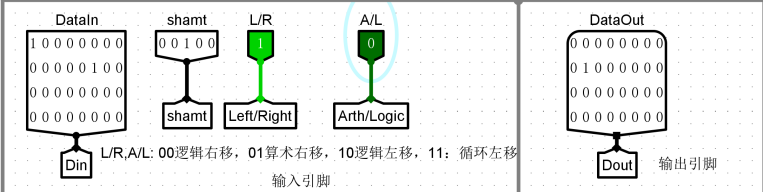
逻辑右移：



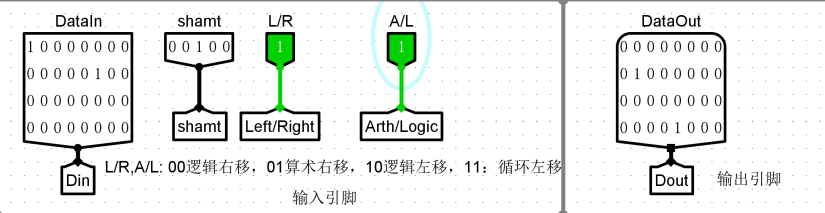
算术右移：



逻辑左移：



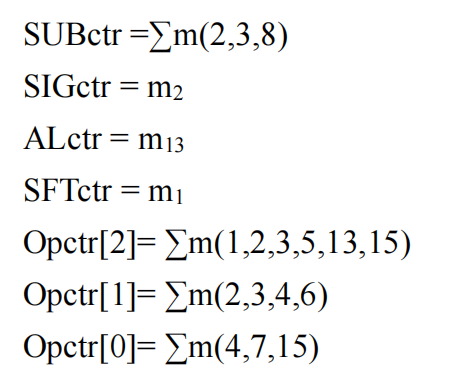
循环左移：



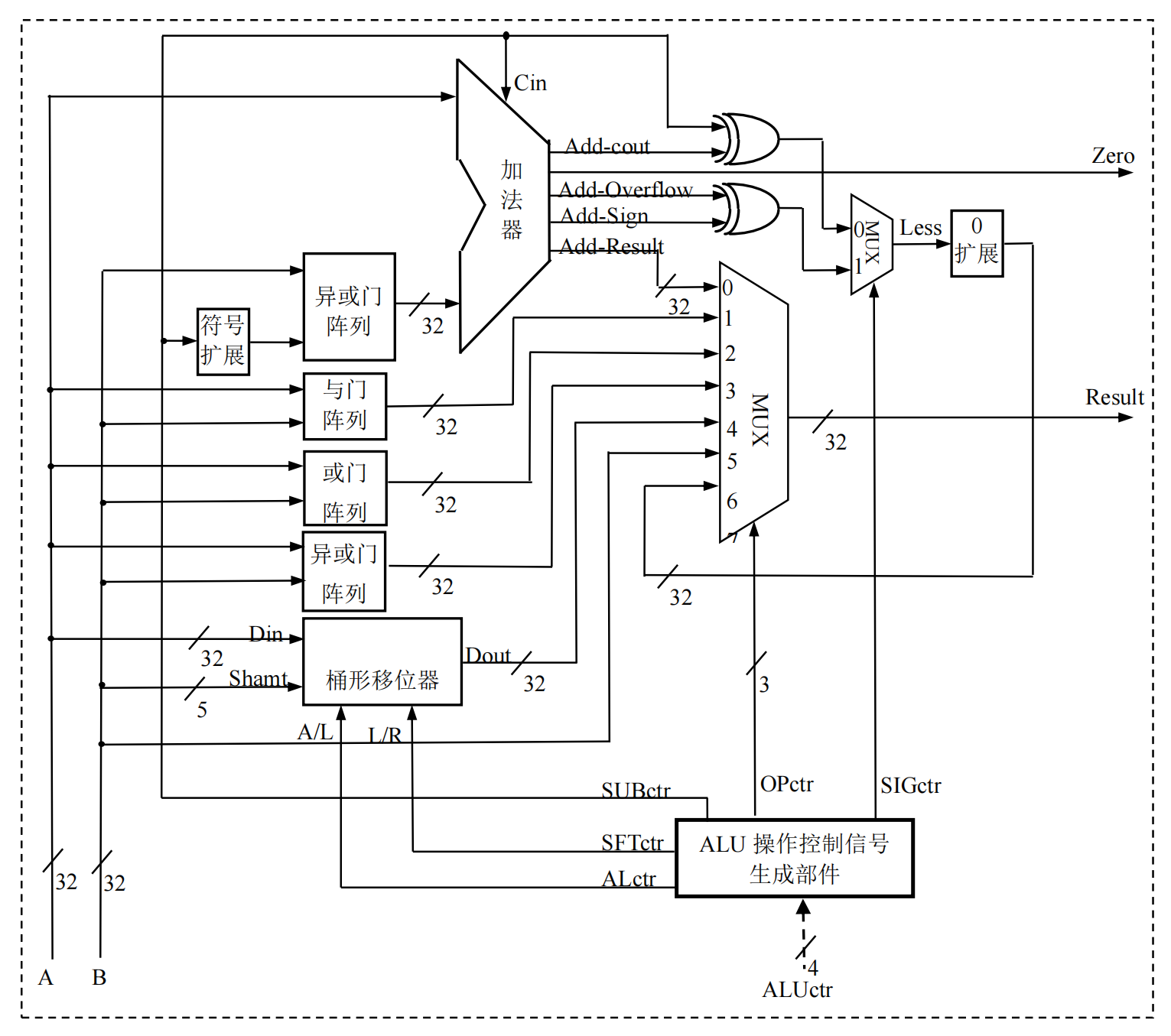
**4.5 ALU设计实验**

1）实验原理

事实上，ALU实验是集合前面几次实验的元件，用控制码控制运行的一个运算器，控制码最小项逻辑表达式如下：

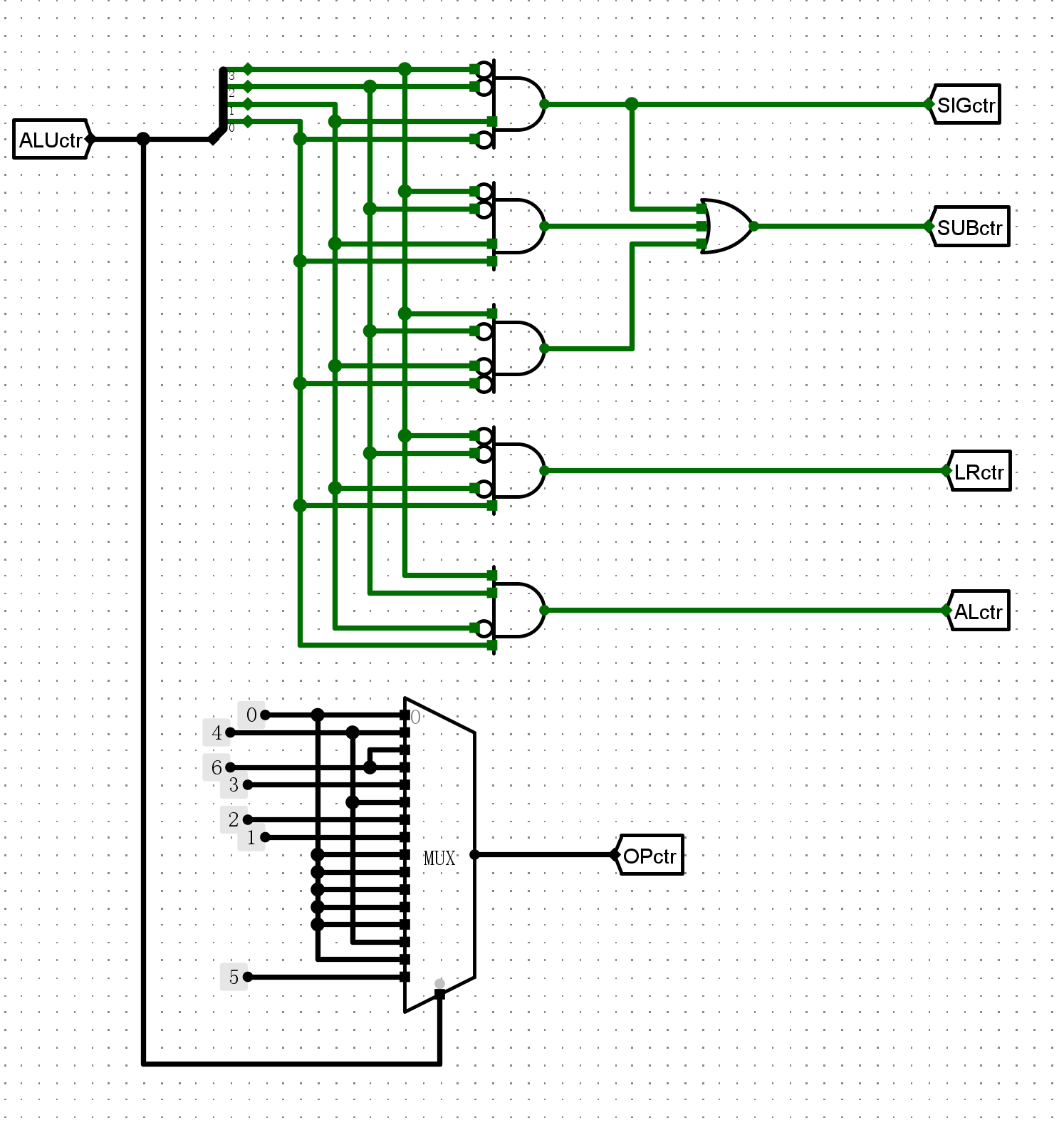


而其他的各个功能与前面实验一样。下面是集合电路原理图：

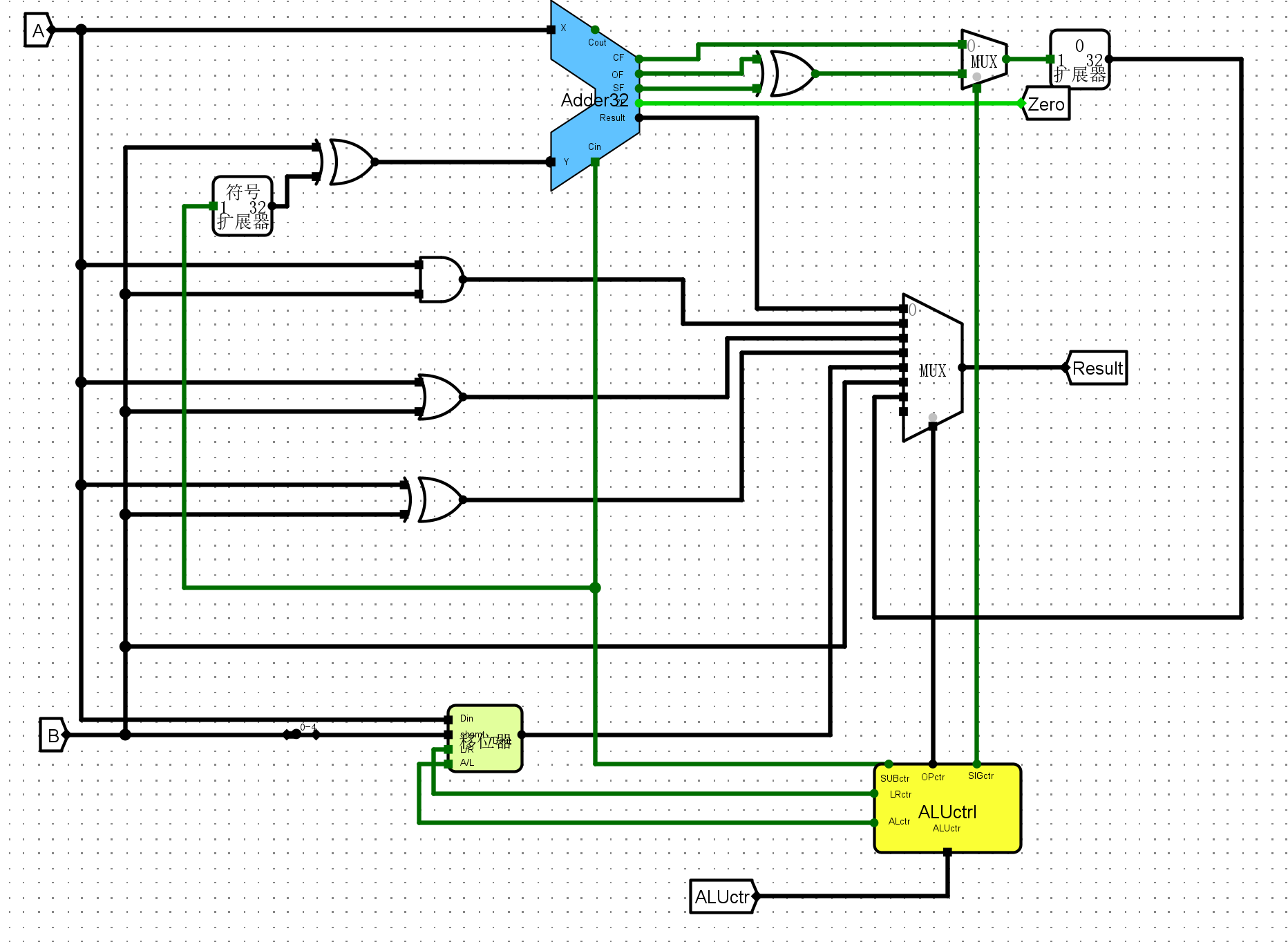


1. Logisim电路图

ALUctr控制转化电路：

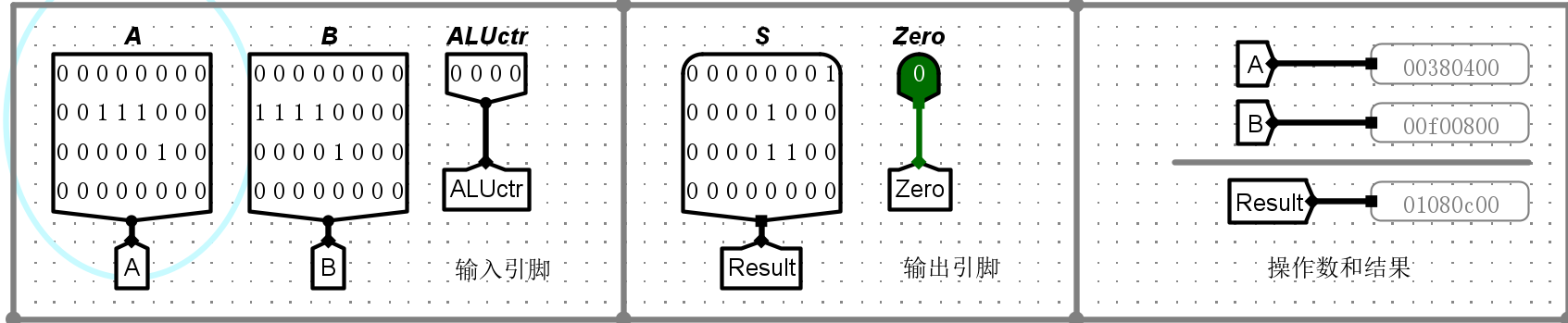


ALU电路如下，内部电路可以参考前面已通过的实验，这里篇幅所限不再展示：

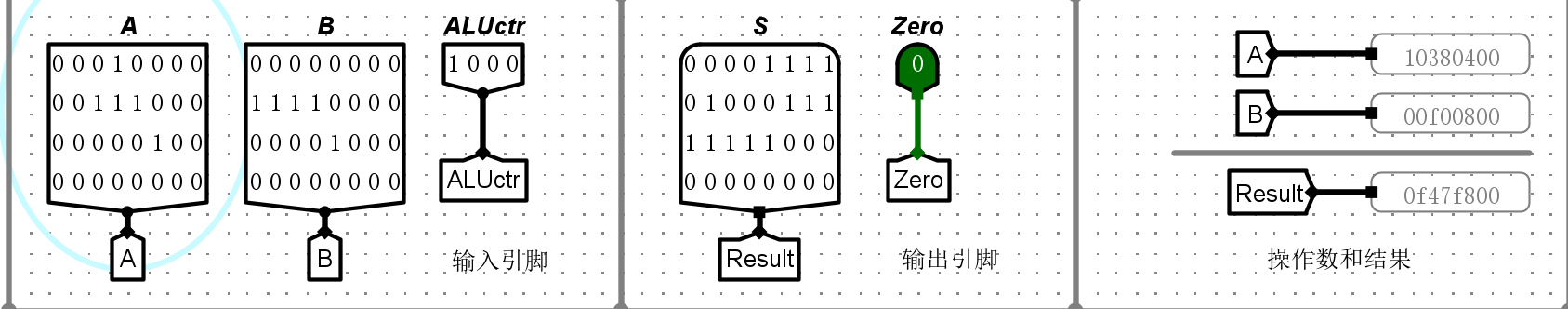


3）实验验证

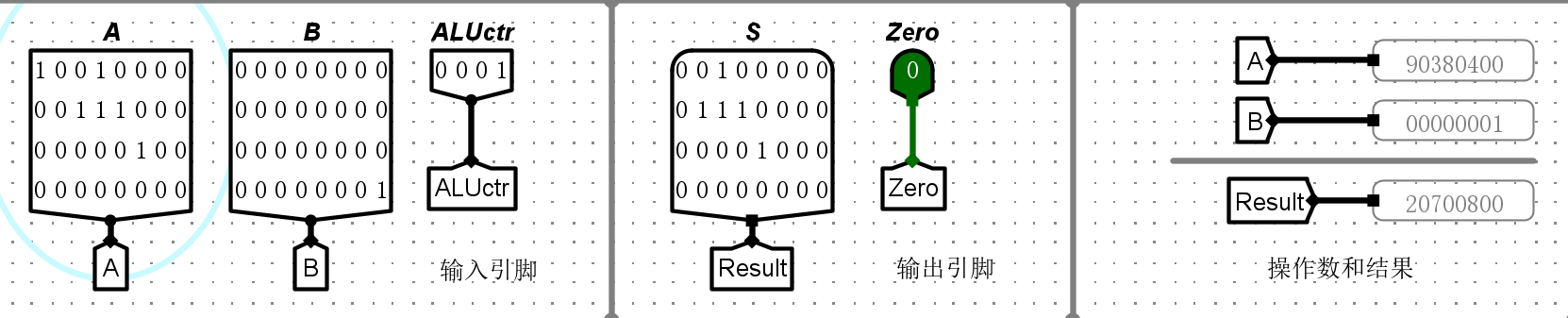
· 加法



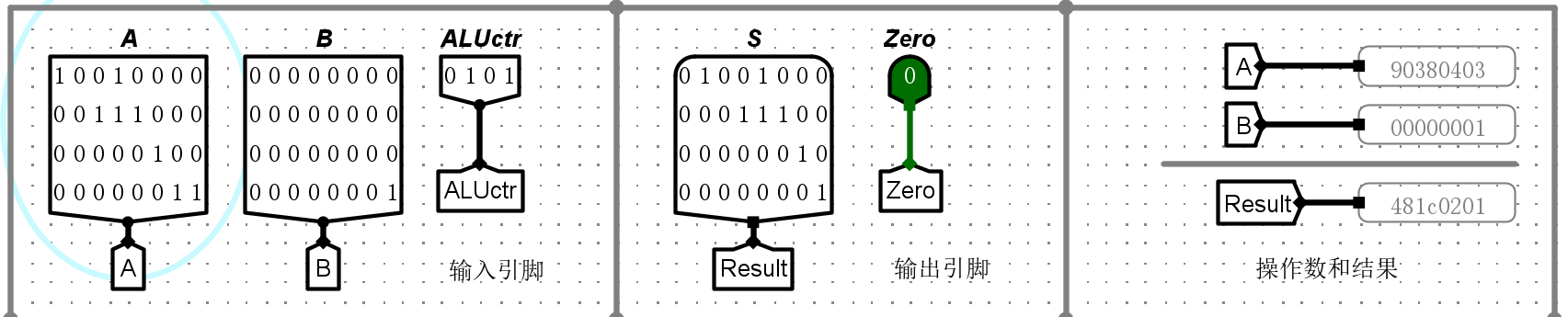
· 减法



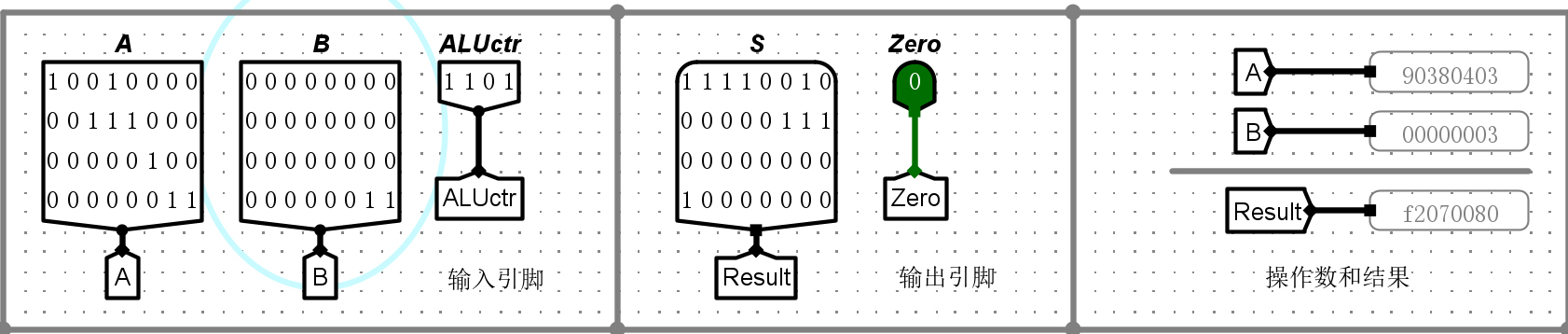
· 逻辑左移



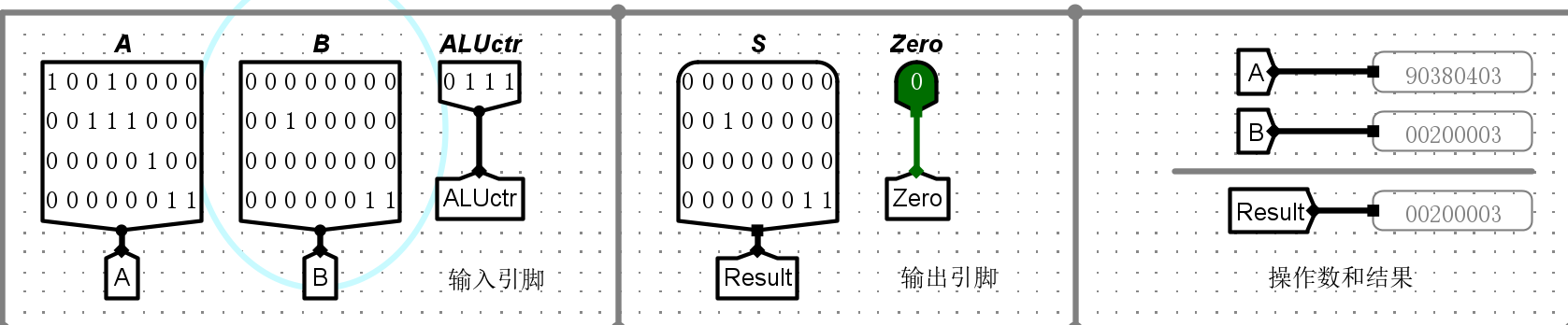
· 逻辑右移



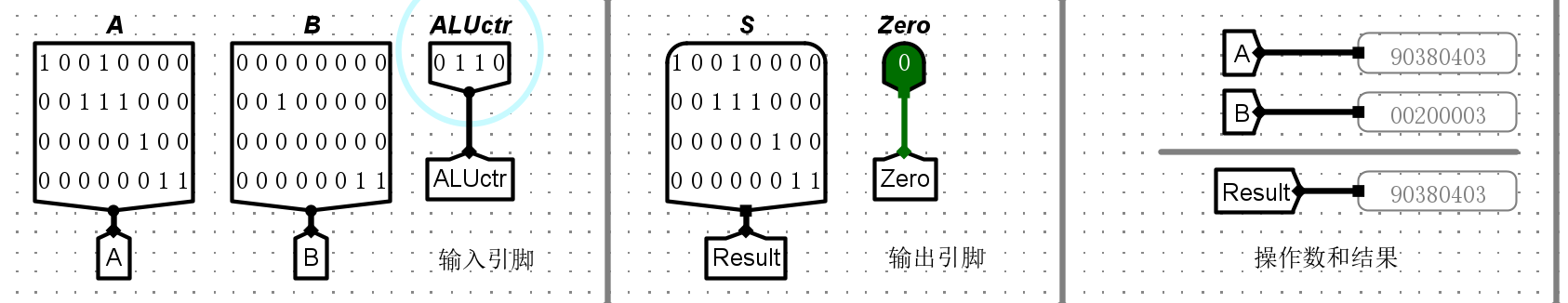
· 算术右移



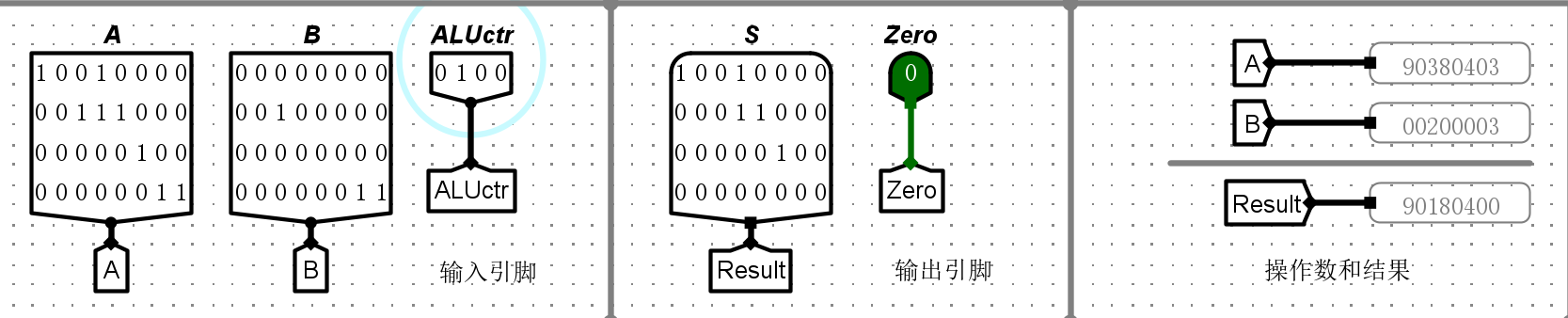
· 与



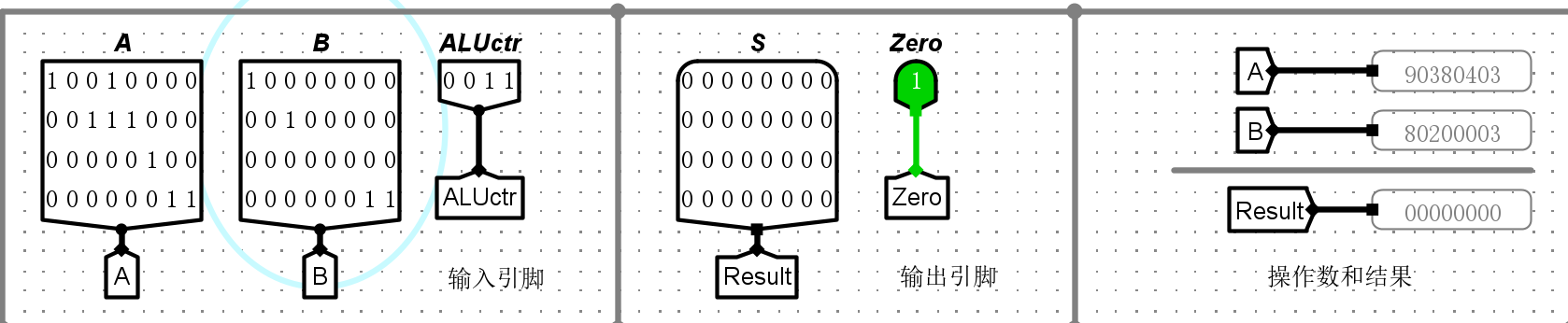
· 或



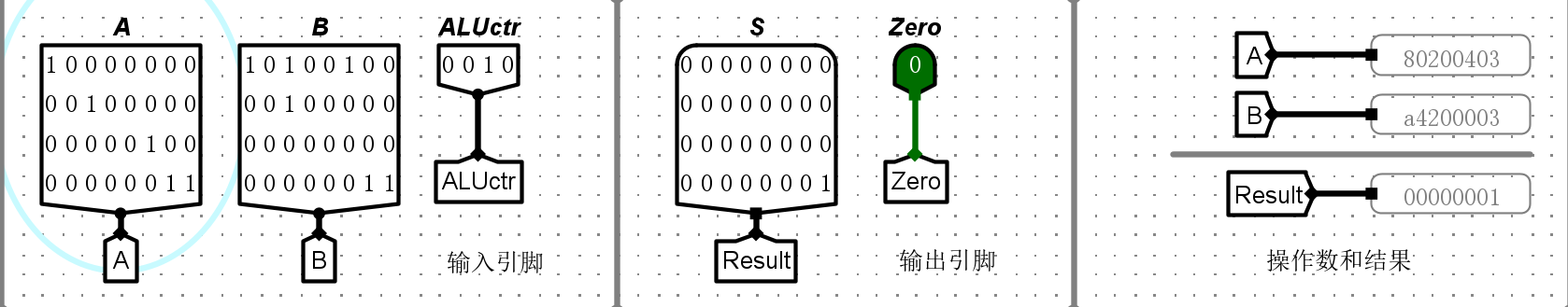
· 异或



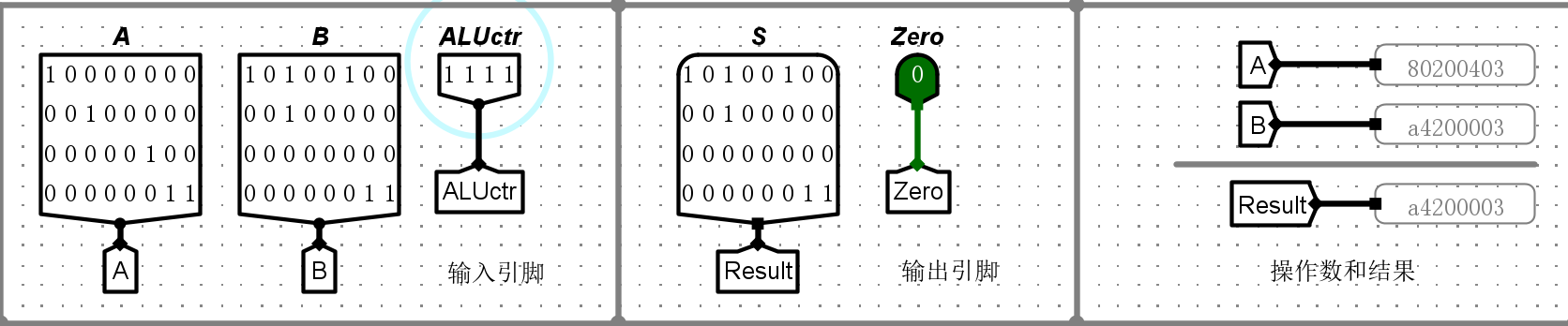
· 无符号小于比较



· 有符号小于比较



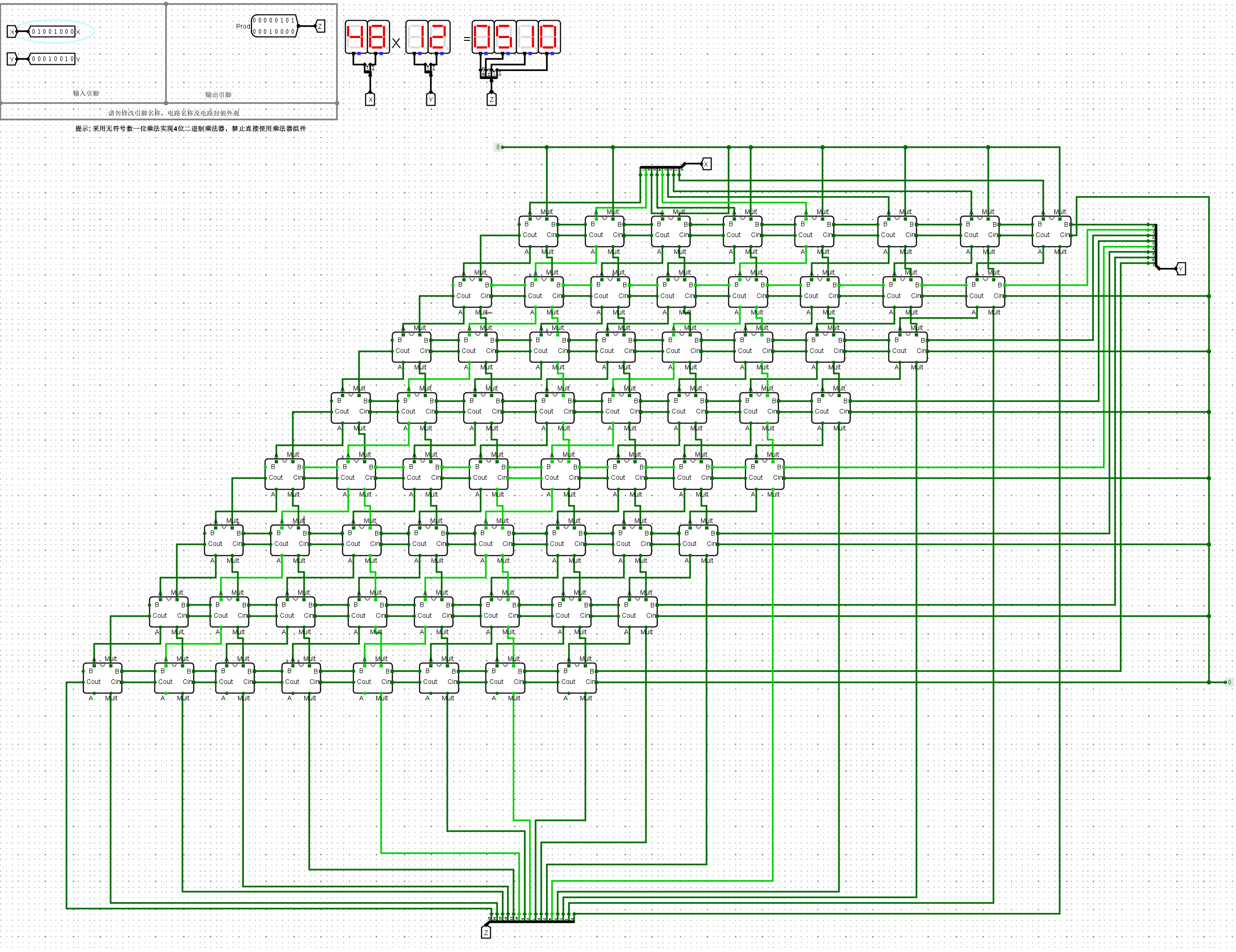
· 取操作数B



**4.6 思考题**

1. 将实验3中的快速乘法器设计电路扩展到32位无符号数相乘，并探讨如何将该乘法器融合到实验中的ALU电路来实现乘法运算。

仍然按照实验3的子细胞阵列思路扩展成32×32的阵列（电路实在太麻烦了，就不连了QAQ（再放一下实验三的电路图，按照这个方式拓展就行了）

融合到ALU中：1001-1100，1110选一个作为ALUctr，输出的OPctr为111（实验中剩余的空位），即可输出所需结果

1. 在RV32I中新增一条指令，然后在ALU中新增一个新运算，并通过测试数据进行验证。

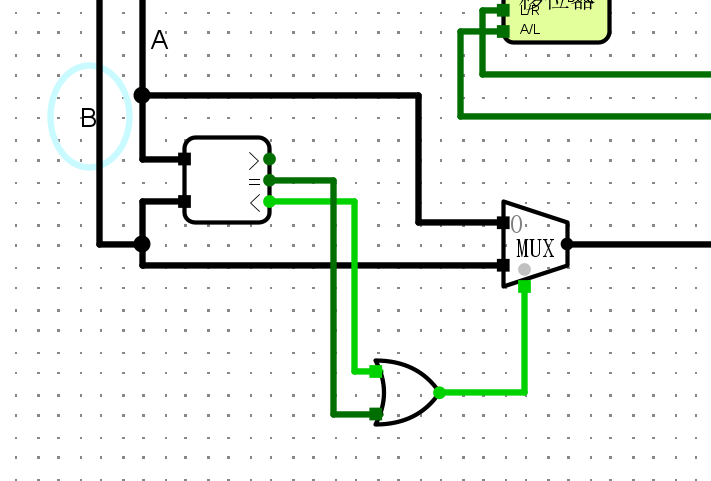
增加的指令：max rd, rs,1 rs2（选择两个有符号数中较大值）

ALUctr=1110, OPctr=111

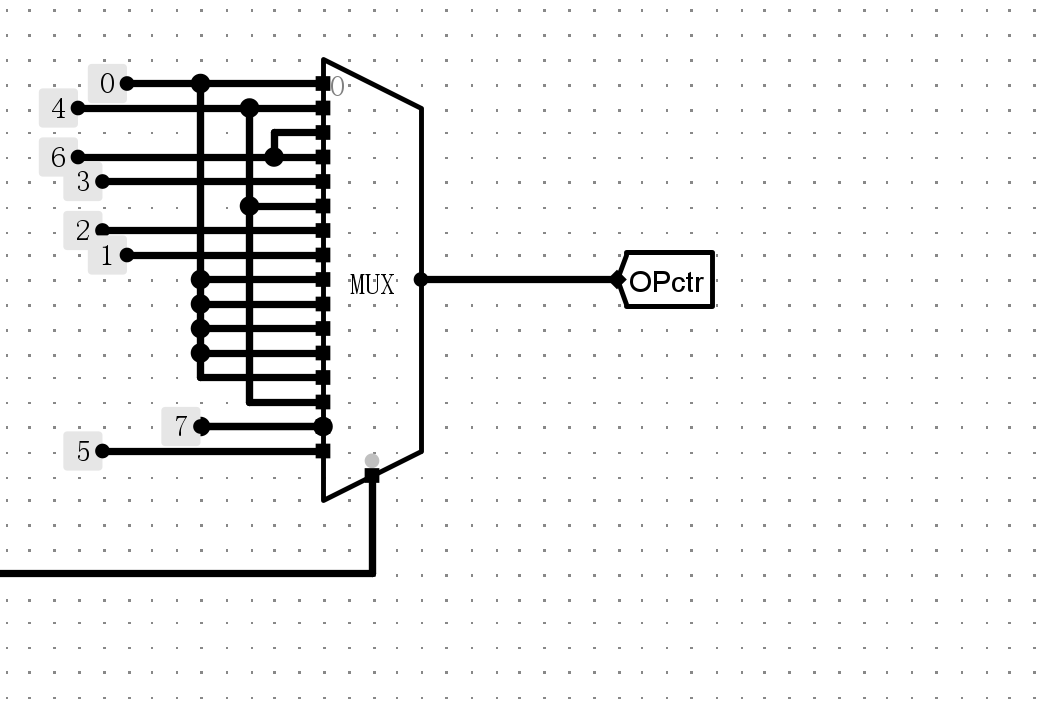
A<=B选A，否则选B

修改电路如下（详见lab4.5.circ中的ALU\_new电路和ALUctr\_new电路）：

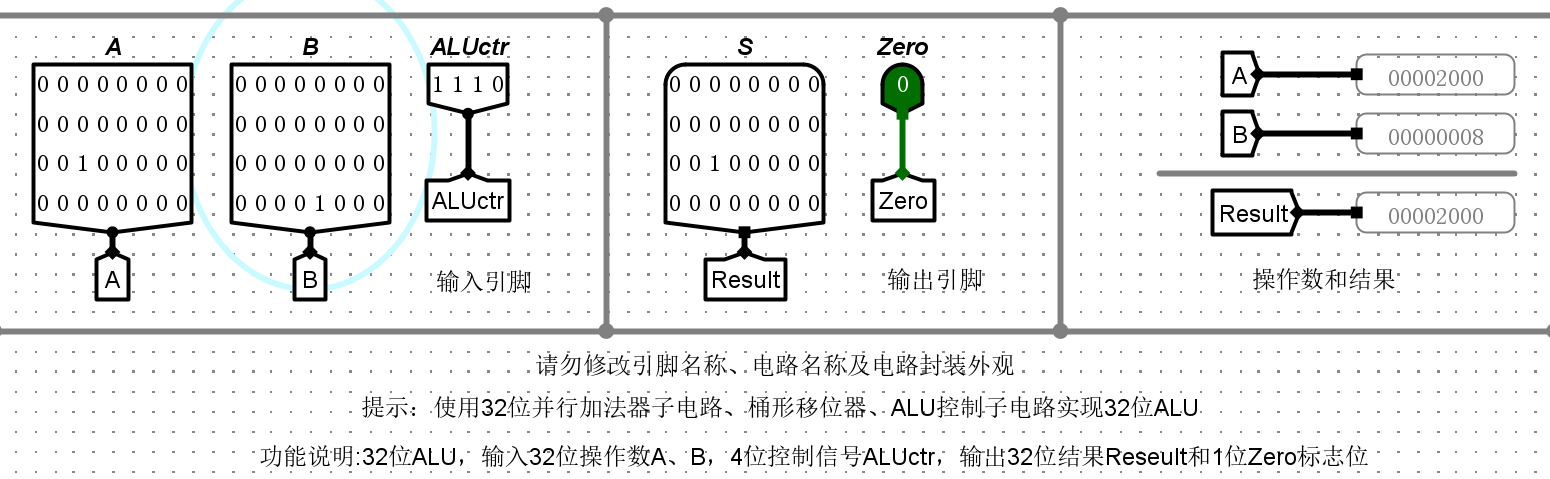
在ALU中加入比较器：

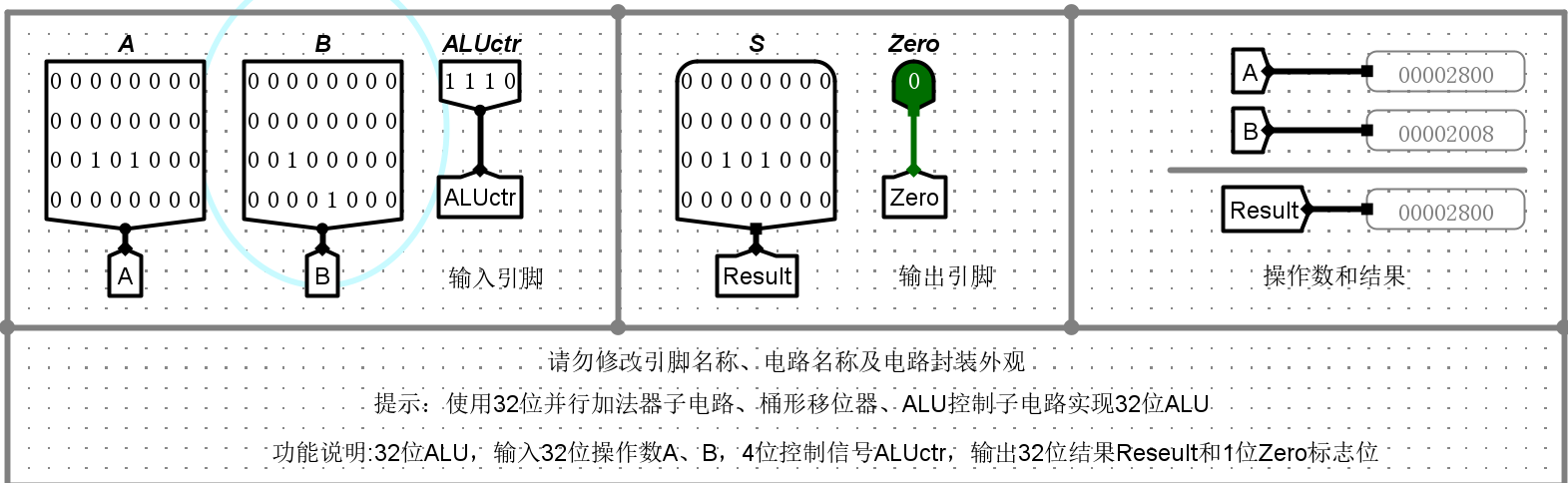


输出新的OPctr电路如下，增加7（111）的输出：



实验验证如下：





两侧均通过。

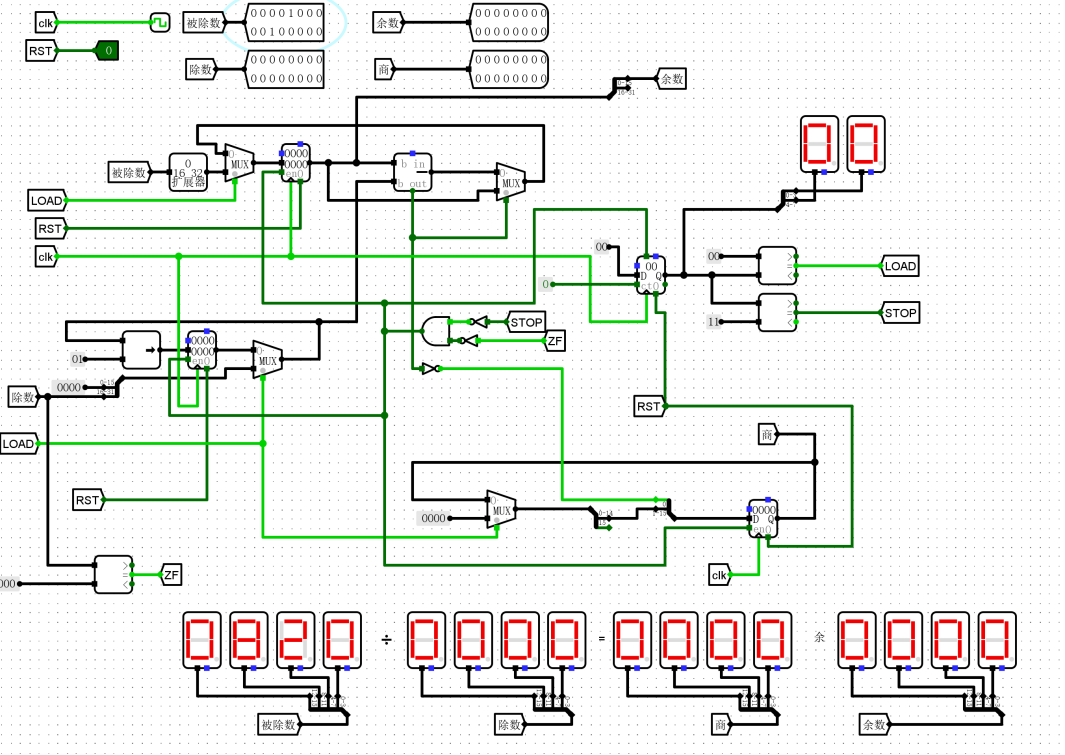
1. 如何实现32位无符号数除法器？

本人实现基本按照手算除法过程，ppt中，被除数位数扩展二倍，前面加上32位0。除数末尾加上32个0，这样在移位器中不用考虑其他问题直接移动33次就好。被除数-移位后的除数，如果够减，商的低位加入一位1，否则加入一位0，减完之后除数移动一位，做差结果作为新的被除数。不够减则上一个被除数仍作为下次被除数。

除数为0的处理：为了防止溢出，除数与0作比较，信号输出ZF，ZF=1则确认除数为0，时钟启动也不进行运算。

由于Logisim最大到32位，如果32位进行扩展则达到64位，无法进行连接电路，故下面电路做的16位无符号数的除法（详见32-division.circ的main电路，test电路失败）。如果32位除法，部分元件应该是64位，停止的计数次数为33（0x21）。下面验证过程已包含电路图。

除数为0：



常规除法：

