華中科技大學

课 程 实 验 报 告

课程名称： 计算机组成原理课程实验

专业班级： 自实 1901

学　　号： U201915560

姓　　名： 肖力文

报告日期： 2021 年 12 月 03 日

人工智能与自动化学院

目录

实验二 运算器组成实验

1 实验目的

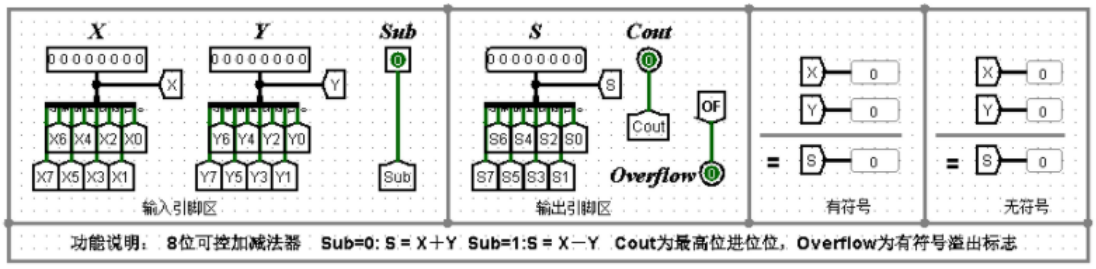
①熟悉 Logisim 软件平台。
②掌握运算器基本工作原理
③掌握运算溢出检测的原理和实现方法;
④理解有符号数和无符号数运算的区别;
⑤理解基于补码的加/减运算实现原理;
⑥理解阵列乘法器基本原理;
⑦熟悉运算器的数据传输通路。

2 实验环境

Logisim 是一款数字电路模拟的教育软件，用户都可以通过它来学习如何创建逻辑电路，方便简单。 它是一款基于 Java 的应用程序，可运行在任何支持 JAVA 环境的平台，方便学 生来学习设计和模仿数字逻辑电路。Logisim 中的主要组成部分之一就在于设计并以图示来 显示 CPU。当然 Logisim 中还有其他多种组合分析模型来对你进行帮助，如转换电路，表达 式，布尔型和真值表等等。同时还可以重新利用小规模的电路来作为大型电路的一部分。

3 实验内容

3.1 八位串行可控加减法电路设计

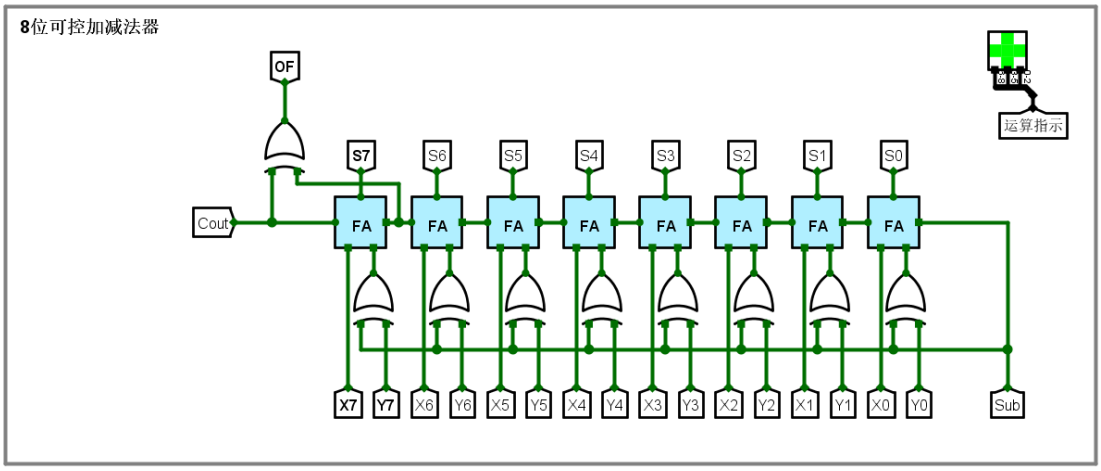


3.1.1 原理

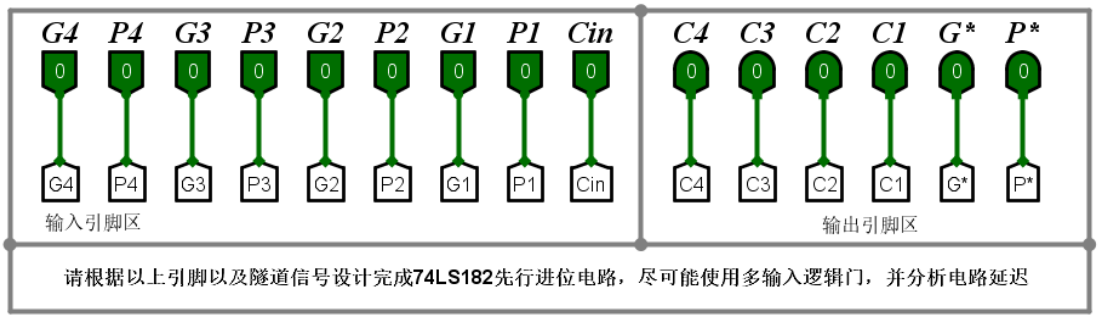
①用八个全加器若干门电路实现位串行可控加减法电路
②增加 Sub 位来控制运算器是执行加法运算还是减法运算
③对于有符号数的溢出检测信号 overflow 可以直接利用最高数值位进位和符号位进位异或得到。

④由于补码运算的特殊性质，减法可以通过加法实现，只需要将减数 Y 的补码再次求补后送入加法器即可实现减法运算，具体公式如下：[X] — [Y] = [X-Y]$_{补}$= [X]$_{补}$ +[-Y]$_{补}$。如图是多位可控加减法电路图，该电路在串行加法器中引入 Sub 控制信号，输入数 Y 的所有位 Y_i 均与 Sub 信号进行异或后送入多位串行加法器，当 Sub=0 时，送入加法器的是 Y 本身，当 Sub=I 时，送入加法器的是 Y 的反码，另外 Sub 连接到加法器的最低位的进位输入，实现了对 Y 操作数的逐位取反，末位加 1 的求补过程，从而完成减法操作，当 Sub=0 时，低位进位为零，不影响加法结果的正确性。对于减法的溢出检测，最直观的判断依据是正数减负数结果为负数，负数减正数结果为正数。

3.1.2 电路



3.2 四位先行进位电路



3.2.1 原理

①思路：用已知量代替过程量，得到进位信号，节约时间
②逻辑表达式：
进位生成函数 Gi = XiYi

进位传递函数 Pi = Xi ⊕ Yi

Cn = Gn + PnGn-1 + PnPn-1Gn-2 + … + PnPn-1Pn-2…P1C0

C1 = G1 + P1C0 = X1Y1 + X1⊕Y1

C2 = G2 + P2G1 + P2P1C0 = X2Y2 + (X2⊕Y2)(X1Y1) + (X2⊕X2)(X1⊕X1)C0

C3 = G3 + P3G2 + P3P2G1 + P3P2P1C0 = X3Y3 + (X3⊕Y3) (X2Y2) + (X3⊕Y3)(X2⊕X2) (X2Y2) + (X3⊕Y3) (X1Y1) + (X3⊕Y3)(X2⊕X2)(X1⊕X1) C0

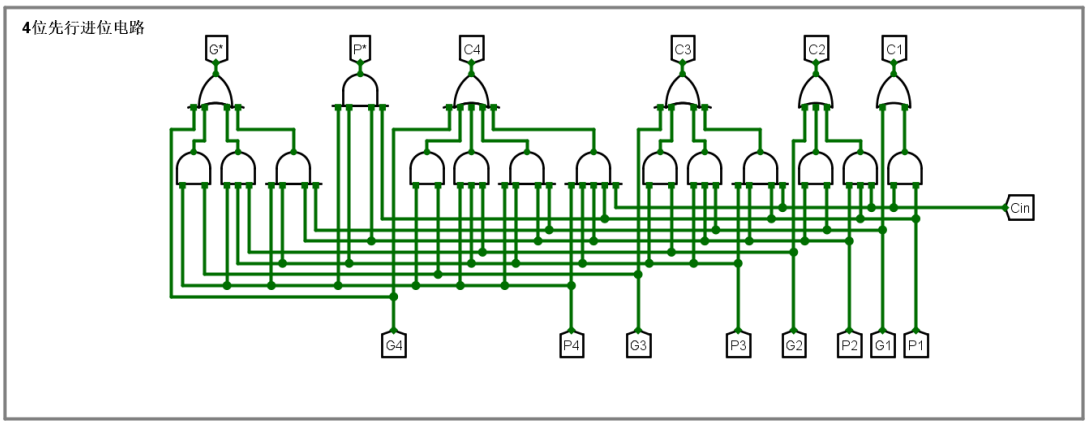
C4 = G4 + P4G3 + P4P3G2 + P4P3P2G1 + P4P3P2P1C0 = X4Y4 + (X4⊕Y4) (X3Y3) + (X4⊕Y4)(X3⊕X3) (X2Y2) + (X4⊕Y4) (X3⊕Y3) (X2⊕Y2) (X1Y1) + (X4⊕Y4)(X3⊕Y3)(X2⊕X2)(X1⊕X1) C0

令
G4* = G4 + P4G3 + P4P3G2 + P4P3P2G1

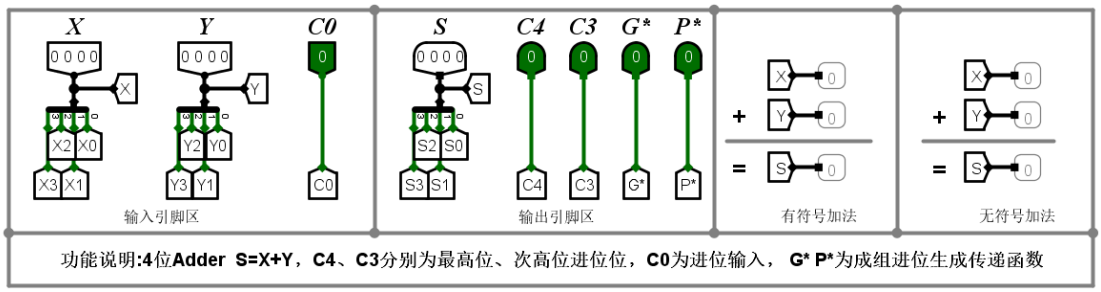
P4* = P4P3P2P1

得
C4 = G4* + P4* C0

3.2.2 电路



3.3 四位快速加法器设计



3.3.1 原理

①电路引脚如图所示

电路引脚

信号	输入/输出	位宽	说明
X	输入	4 位	加数
Y	输入	4 位	加数
C0	输入	1 位	进位输入
S	输出	4 位	运算和
C4	输出	1 位	最高位进位位
C3	输出	1 位	第 3 位进位位
G*	输出	1 位	成组生成函数
P*	输出	1 位	成组传递函数

②先将需要加和的数输入与门异或门电路，得到 P 和 Q

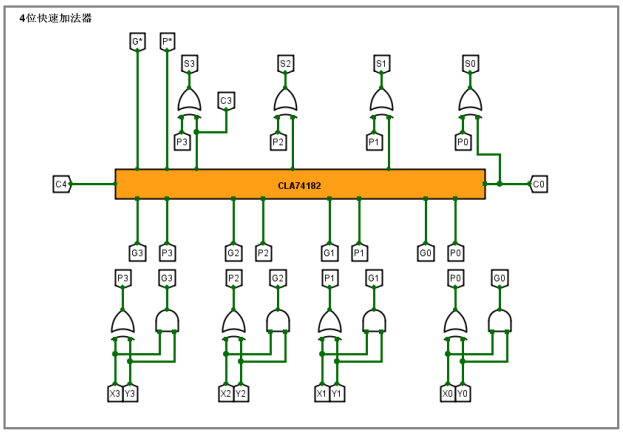
Pi=Xi⊕Yi

Gi=XiYi

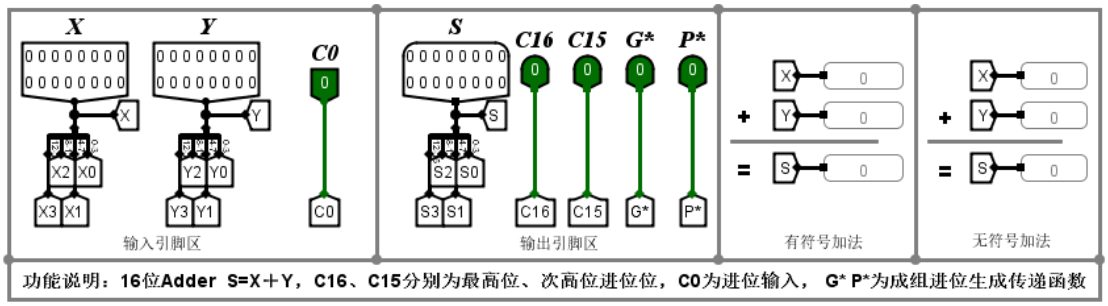
③将得到的 P 和 Q 输入四位先行进位电路，得到 C

④将得到的 C 与 P 异或，得到加和之后的值

3.3.2 电路



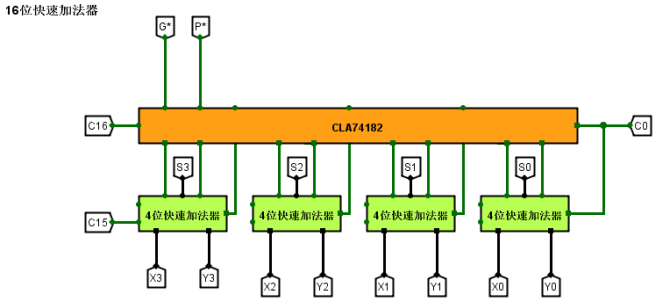
3.4 十六位快速加法器设计



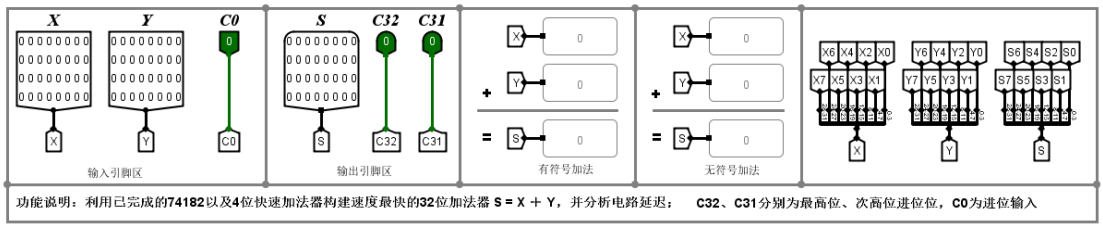
3.4.1 原理

①将 4 个 4 位快速加法器输出的成组进位生成、传递函数G^* 和P^* 及C_0 先连接到先行进位电路的输入端，即可先行产生 G4, G8, G12, G16 四个进位信号。再将对应信号连接到相应的快速加法器的进位输入端即可构成 16 位组内并行进位、组间并行进位的快速加法器。

3.4.2 电路



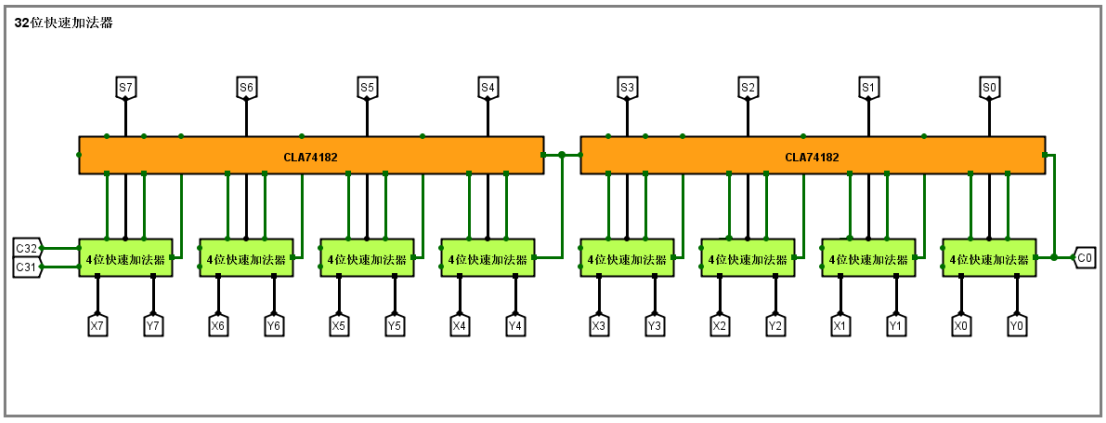
3.5 32 位快速加法器设计



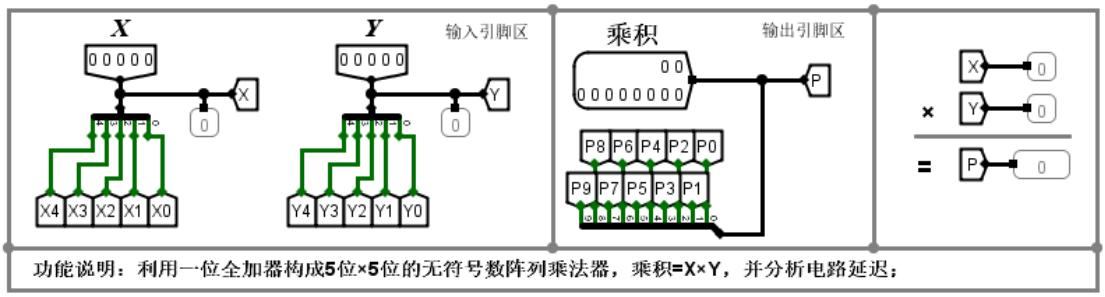
3.5.1 原理

①两个 16 位快速加法器直接串联，C16 进位信号采用上层的进位输出 C4

3.5.2 电路



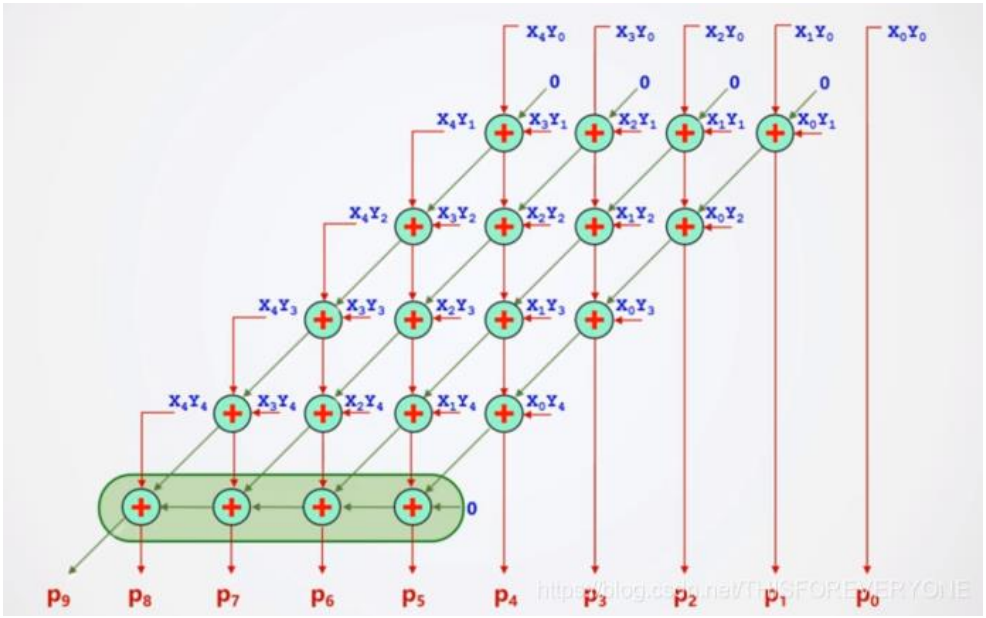
3.6 阵列乘法器设计



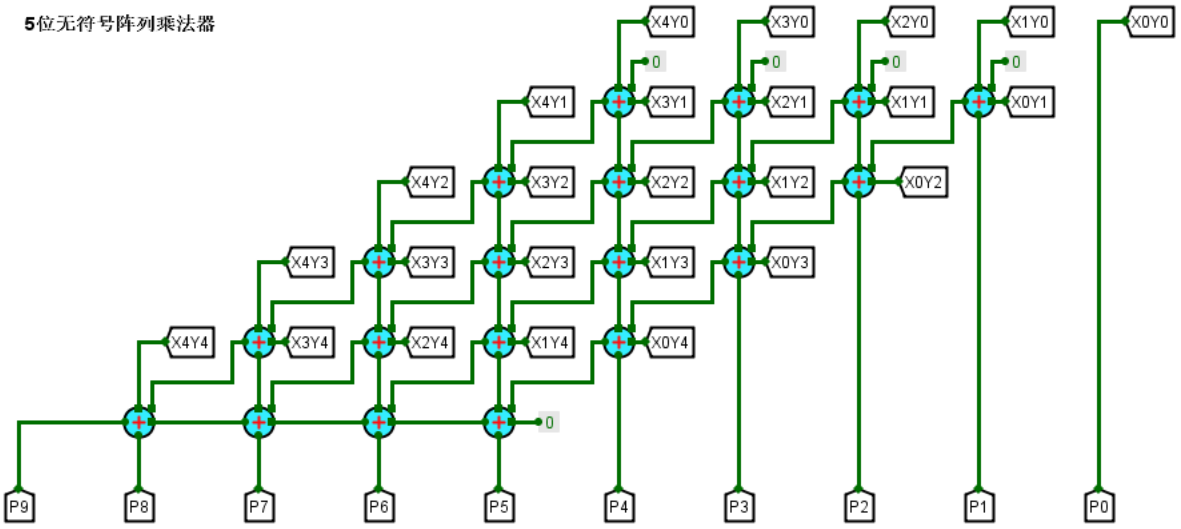
3.6.1 原理

①n 位求补器：根据输入的 6 位补码的符号位决定我们要通过求补码之后的 5 位还是没有求

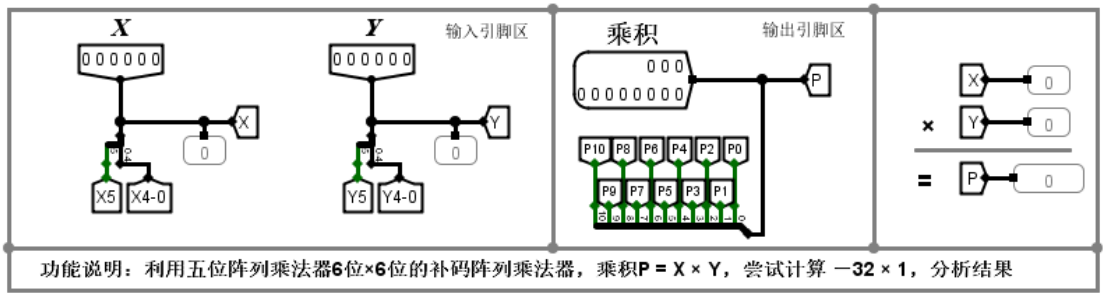
过补码的原来的数值位。这个就可以用 logisim 中的 5 位求补器来实现。
②与门阵列与阵列乘法器：无符号 5 * 5 阵列乘法器，输出是 10 位需要用相应的与门阵列构建
③具体原理如下：



3.6.2 电路



3.7 6 位补码阵列乘法器



3.7.1 原理

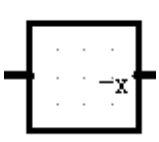
①因为是补码，符号位独立运算，数据位先转换为绝对值。
②由补码转换规则进行转换。
③XY 绝对值乘运算后，在转回补码。
④在 6 位补码阵列乘法器中利用 5 位阵列乘法器以及求补器等部件实现补码阵列乘法器
哇用到了分离器和其他几个运算器。有补码器，加法器，数据选择器。如下图。

1）数据选择器:

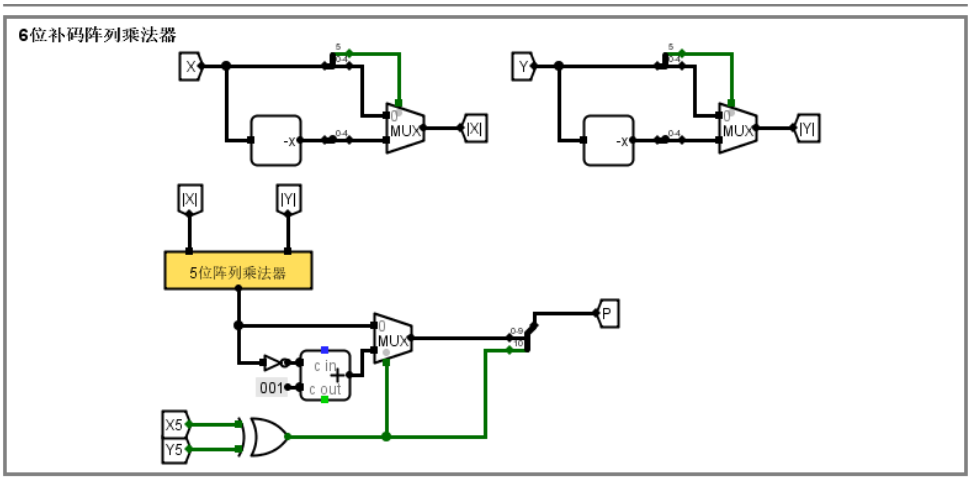
2）分离器:

选区：分离器(Splitter)	
朝向	东
输出	1
位宽	6
外观	中心式
位0	0（顶部）
位1	0（顶部）
位2	0（顶部）
位3	0（顶部）
位4	0（顶部）
位5	无

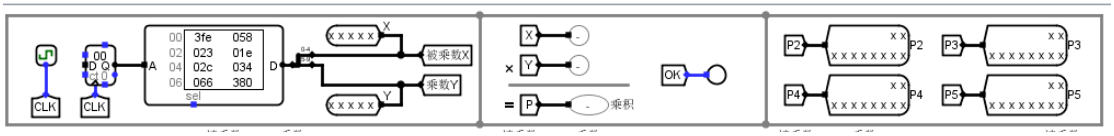
3）补码器:



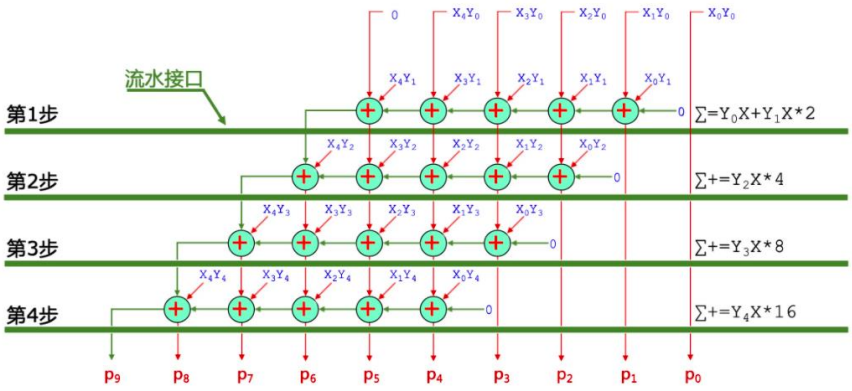
3.7.2 电路



3.8 五位无符号乘法流水线电路



3.8.1 原理

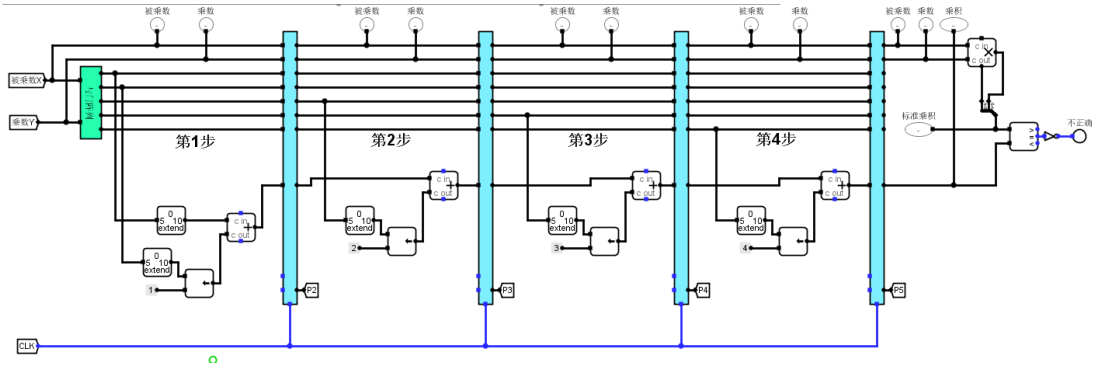


流水接口：寄存器，锁存加工结果 ∑，和后续步骤需要的数据 Y_iX

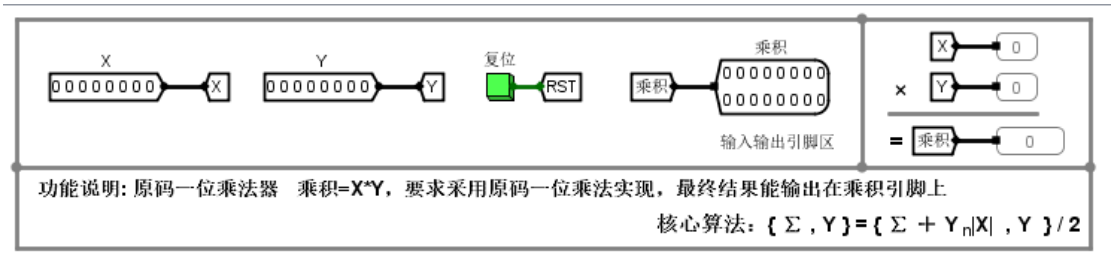
根据上图，我们可以得到流水线电路的设计思路。首先是被乘数 X 的每一位和乘数 Y 的最后一位相乘，得到一个部分积，由于结果是一个十位的二进制数，所以需要先将其扩展为十位，再送入下一级。第二级流水线的结果是被乘数 X 的每一位和乘数 Y 的倒数第二位相乘，实际结果应该是计算结果的两倍，也就是说第二级计算的结果比第一级高一位，并不能直接相加，所以需要先将第二级的结果左移一位再与第一级的结果相加。三四级的原理同上，最后我们便得到了计算的结果，再与自带的乘法器算得的结果进行比较，便得到了所有电路

的设计。

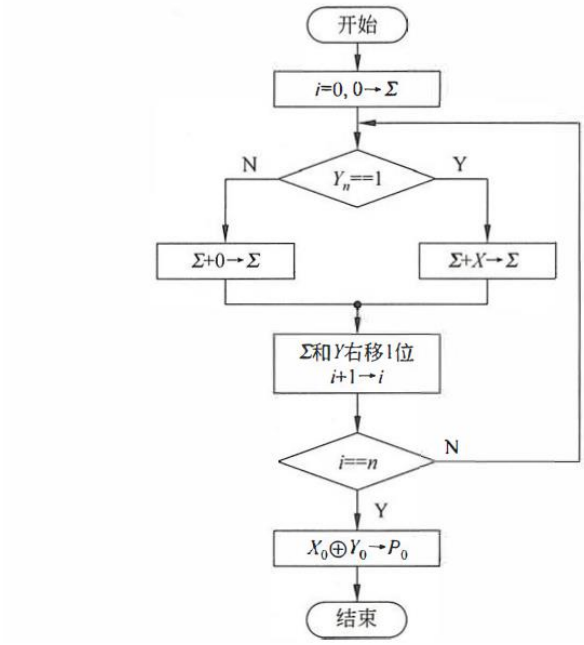
3.8.2 电路



3.9 原码一位乘法器



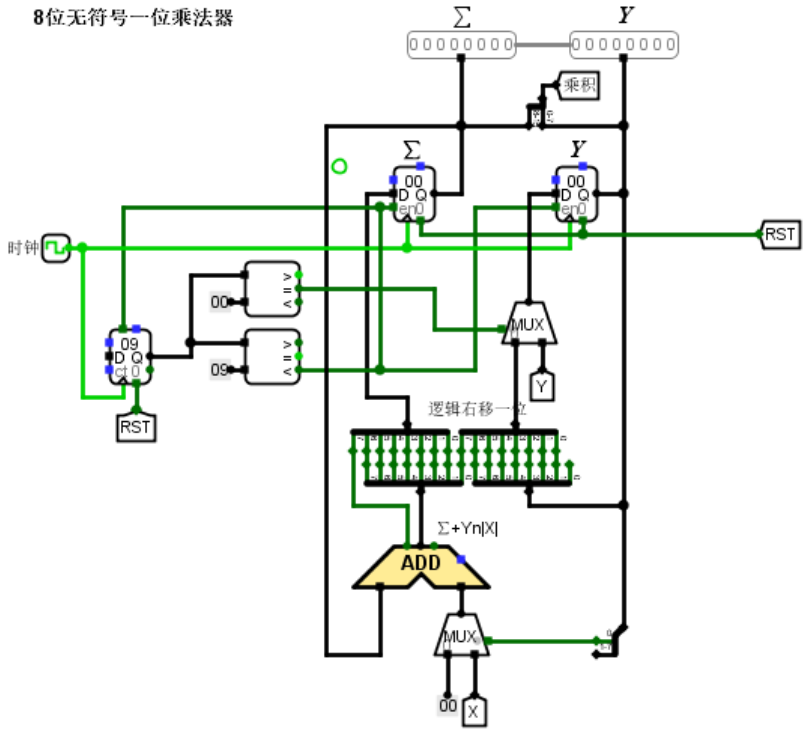
3.9.1 原理



上图是原码 1 位乘法的流程图，由于原码数值部分和真值相同，所以可以单独计算符号

位，利用∑ 存放部分积，i 为循环计数器，初始值均为零，根据乘数 Y 的最后一位的值，决定每次累加上 0，还是被乘数|X|。运算完毕后，部分积∑ 右移 1 位，Y 右移 1，然后继续累加运算，当乘数所有数值位均参与运算后，运算结束，得到数值部分的运算结果∑ ,最后单独计算符号位即可完成原码 1 位乘法运算。如果数值部分为 n 位，需要进行 n 次加法运算和移位操作。

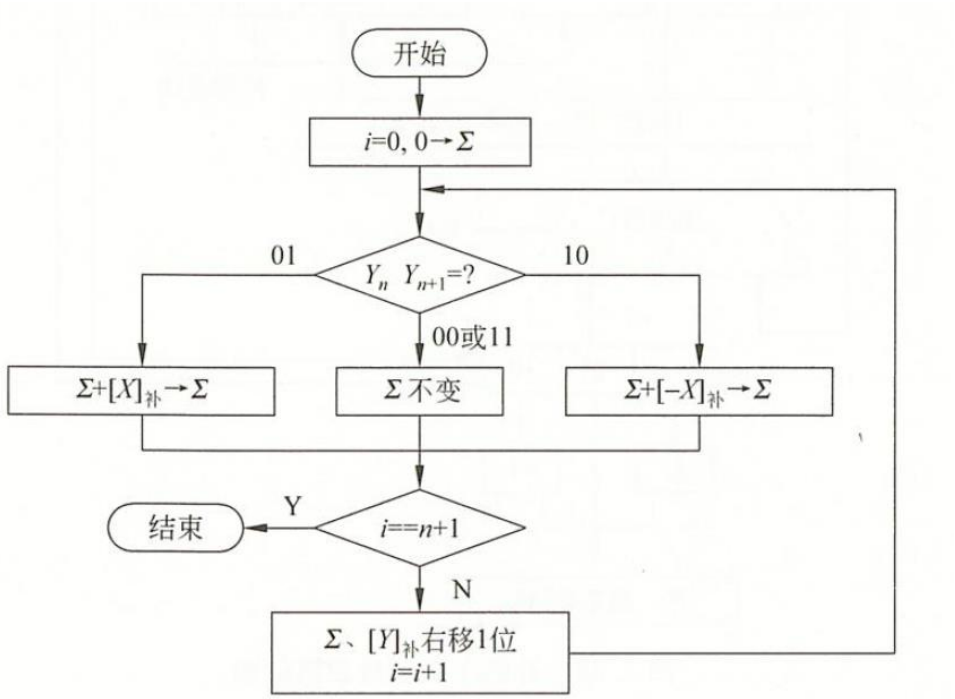
3.9.2 电路



后续所有实验中凡是涉及存储器件，如寄存器、计数器、RAM等，必须增加统一的复位信号RST，方便系统复位

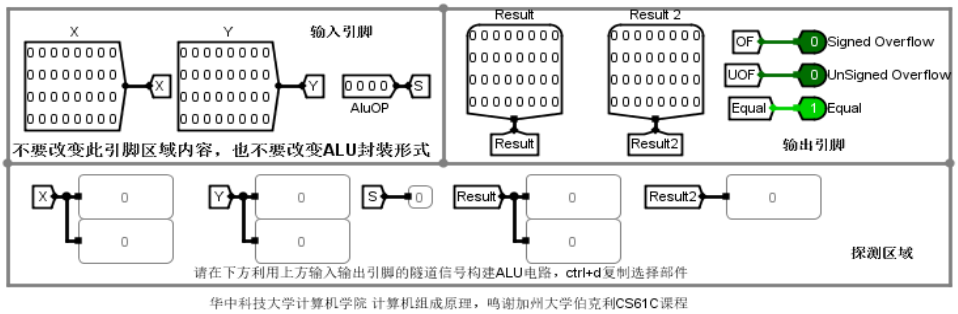
3.10 补码一位乘法器

3.10.1 原理



booth 1 位乘法中乘数采用双符号位参加运算 ，符号位也参与运算。利用 $\sum$ 存放部分积 ，i 为 循环计数器 ， 初始值为零，部分积 累加公式为 $\sum = \sum + (Y_{n+1} - Y_n)[X]_{补}$，根据 $Y_{n+1} - Y$ 决定累加运算的参数是 0 还是[X]补 或者是[—X]补。 运算完毕后，先判断循环次数是否达到 ，如未达到则部分 $\sum$ 右移 1 位，Y 右移 1 位，然后继续循环累加 ，当乘数符号位参与运算后，运算结束，得到的乘积存放在 $\sum$ 和 Y 中，无须单独计算符号位。如果数值部分为 n 位，需要进行 n+1 次加法运算和 n 次移位操作。

3.11 算术逻辑运算单元



3.11.1 原理

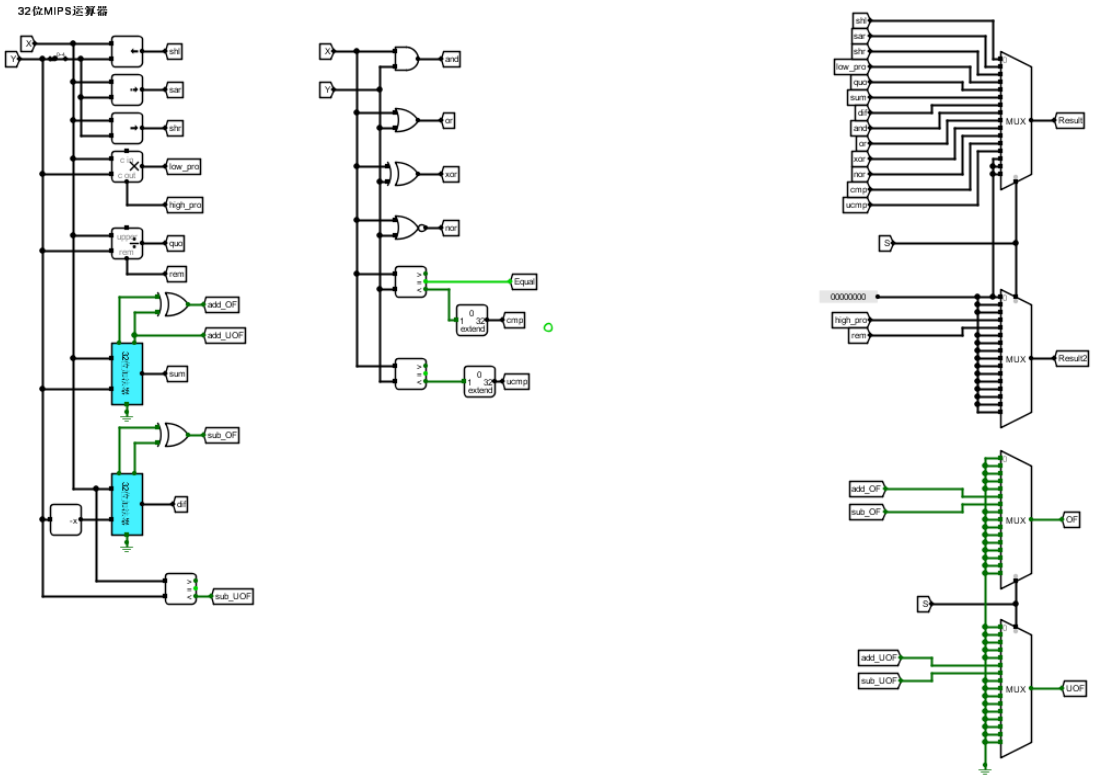
①功能如图所示：

ALU_OP	十进制	运算功能
0000	0	R = X << Y 逻辑左移 (Y取低五位) R_2=0
0001	1	R = X >> Y 算术右移 (Y取低五位) R_2=0
0010	2	R = X >> Y 逻辑右移 (Y取低五位) R_2=0
0011	3	R = $(X * Y)_{[31:0]}$ R_2 = $(X * Y)_{[63:32]}$ 无符号乘法
0100	4	R = X/Y R_2 = X%Y 无符号除法
0101	5	R = X + Y (Set OF/UOF)
0110	6	R = X - Y (Set OF/UOF)
0111	7	R = X & Y 按位与
1000	8	R = X | Y 按位或
1001	9	R = X⊕Y 按位异或
1010	10	R = ~(X |Y) 按位或非
1011	11	R = (X < Y) ? 1 : 0 有符号比较
1100	12	R = (X < Y) ? 1 : 0 无符号比较

②我们先构造出每一种功能的输出，再根据 OP 的值来选择输出
0 操作：逻辑左移，用一个移位器即可实现
1 操作：算术右移，同样用自带的移位器，更改属性为算术
2 操作：逻辑右移，同理
3 操作：无符号乘法，用自带的乘法器
4 操作：无符号除法，用自带的除法器
5 操作：加法，用构造好的 32 位加法器，add_OF 即判断最高位进位和符号位进位是否一致；add_UOF 即判断是否有进位。
6 操作：减法，用构造好的 32 位加法器。Y 各位取反，即可达到减法变加法。无符号数的减法溢出，带加减功能的 ALU 的进位取反后表示，有符号数的减法溢出，仍然用最高位和符号位是否相等来判断。
7—10 操作：直接用多位的与门/或门/异或门/同或门
11 操作： 比较器，补码型，结果 0 拓展
12. 比较器，无符号型，结果 0 拓展，再给出一个 equal 信号

③结构都构造好了之后，根据 OP 的值输出 Result，即可。

3.11.2 电路

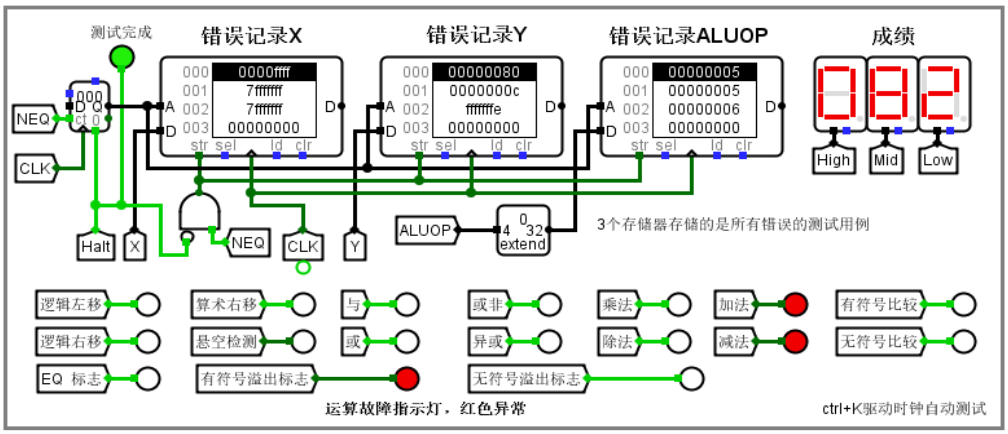


4 遇到的问题及解决方案

4.1 问题一

4.1.1 问题描述

在 ALU 自动测试实验中，第一次实验得到的结果如下所示

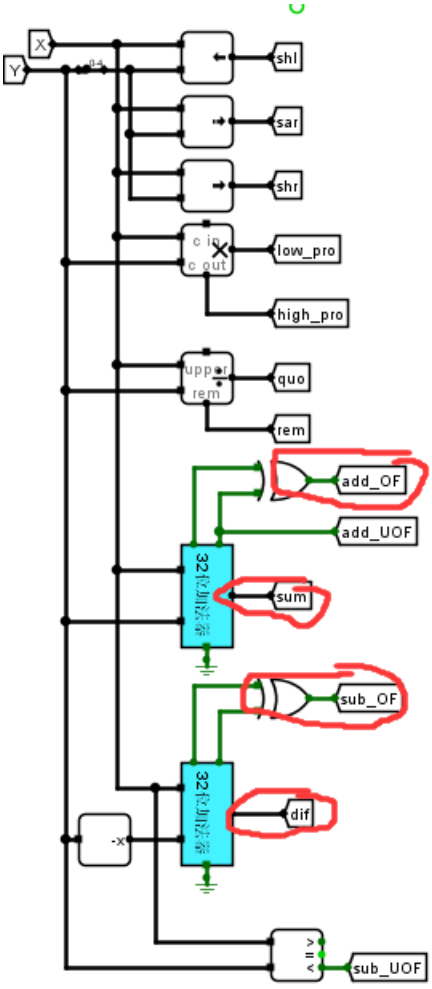


华中科技大学：计算机组成课程组

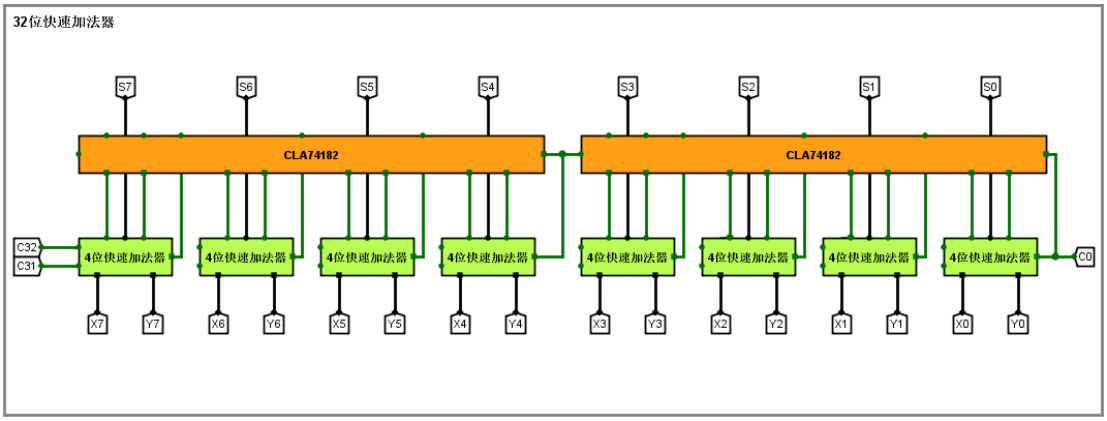
在加法、减法、有符号溢出标志处出现了错误.

4.1.2 解决思路

①回到 ALU 的电路图中去寻找可能出错的地方：（红色标记处即为出错处）

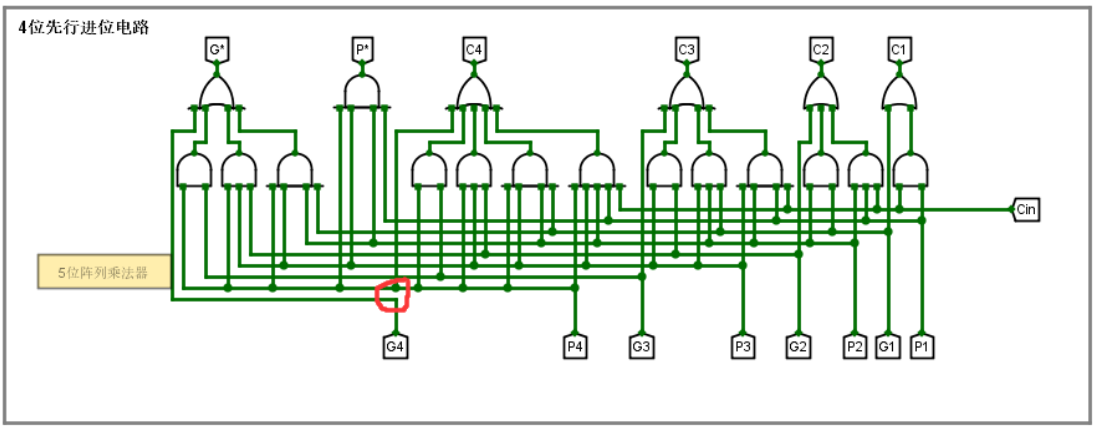


②根据错误提示，说明圈起来的地方出错了，也就是 32 位加法器内部出错了，于是我们进入 32 位加法器中去寻找错误

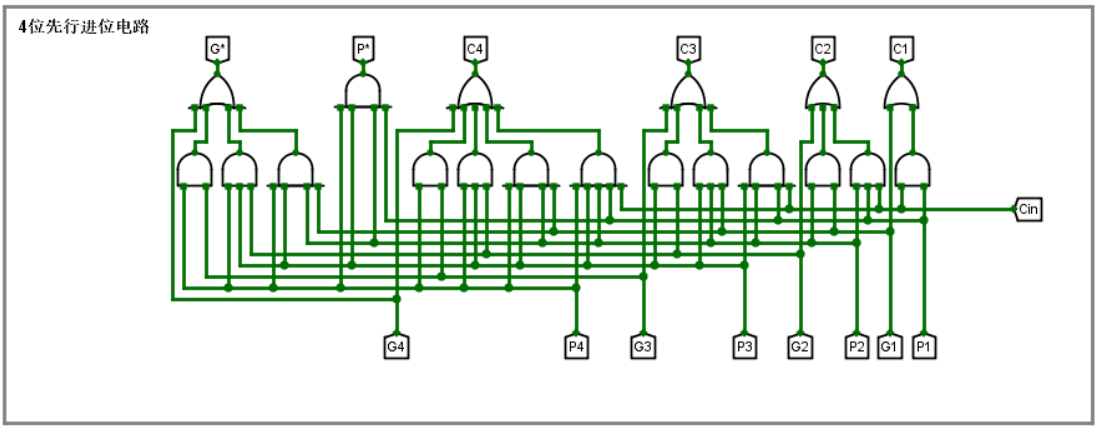


③32 位加法器的连接是没有错误的，并且显示是有符号溢出位出现错误，所以我估计问题

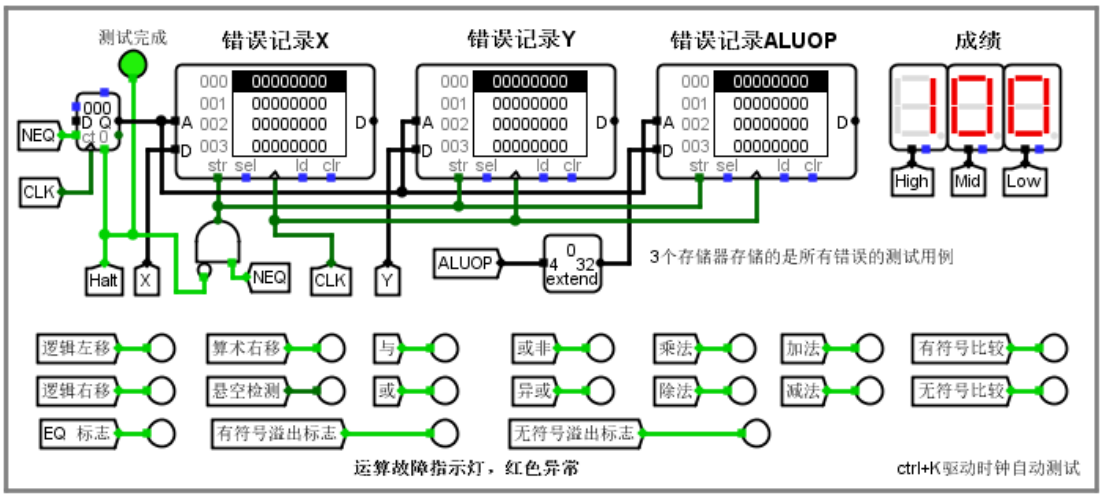
很有可能出现在 74182 中，并且溢出判断位很有可能出现错误，于是我们进入 74182 中寻找错误，特别注意溢出判断位的错误。
经过仔细地比对，发现 C4 有一处出现连接错误，如图：



修正后的连线为：



4.1.3 修正后结果



华中科技大学：计算机组成课程组

如图所示：修正后结果正确