

**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

[1 实验目的 3](#_Toc89709496)

[2 实验环境 3](#_Toc89709497)

[3 实验内容 3](#_Toc89709498)

[3.1 八位串行可控加减法电路设计 3](#_Toc89709499)

[3.2 四位先行进位电路 4](#_Toc89709500)

[3.3四位快速加法器设计 6](#_Toc89709501)

[3.4 十六位快速加法器设计 7](#_Toc89709502)

[3.5 32位快速加法器设计 8](#_Toc89709503)

[3.6阵列乘法器设计 8](#_Toc89709504)

[3.7 6位补码阵列乘法器 10](#_Toc89709505)

[3.8 五位无符号乘法流水线电路 11](#_Toc89709506)

[3.9 原码一位乘法器 12](#_Toc89709507)

[3.10 补码一位乘法器 13](#_Toc89709508)

[3.11 算术逻辑运算单元 14](#_Toc89709509)

[4 遇到的问题及解决方案 16](#_Toc89709510)

[4.1 问题一 16](#_Toc89709511)

**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；

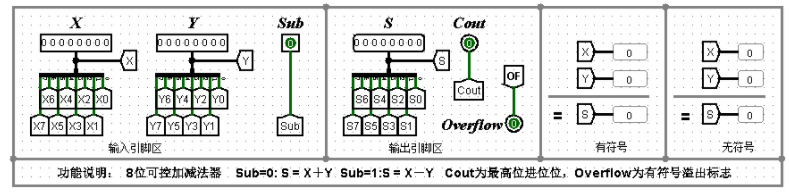
⑦熟悉运算器的数据传输通路。

## 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

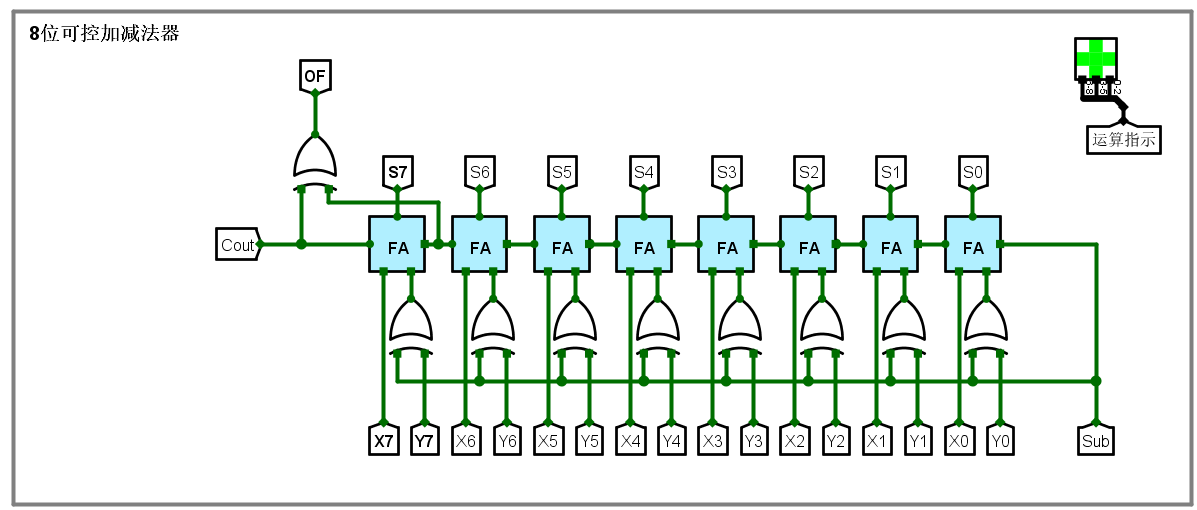
①用八个全加器若干门电路实现位串行可控加减法电路

②增加Sub位来控制运算器是执行加法运算还是减法运算

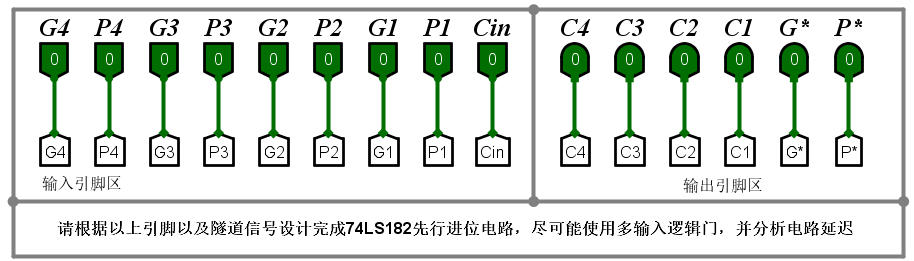
③对于有符号数的溢出检测信号overflow可以直接利用最高数值位进位和符号位进位异或得到。

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Yi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=l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

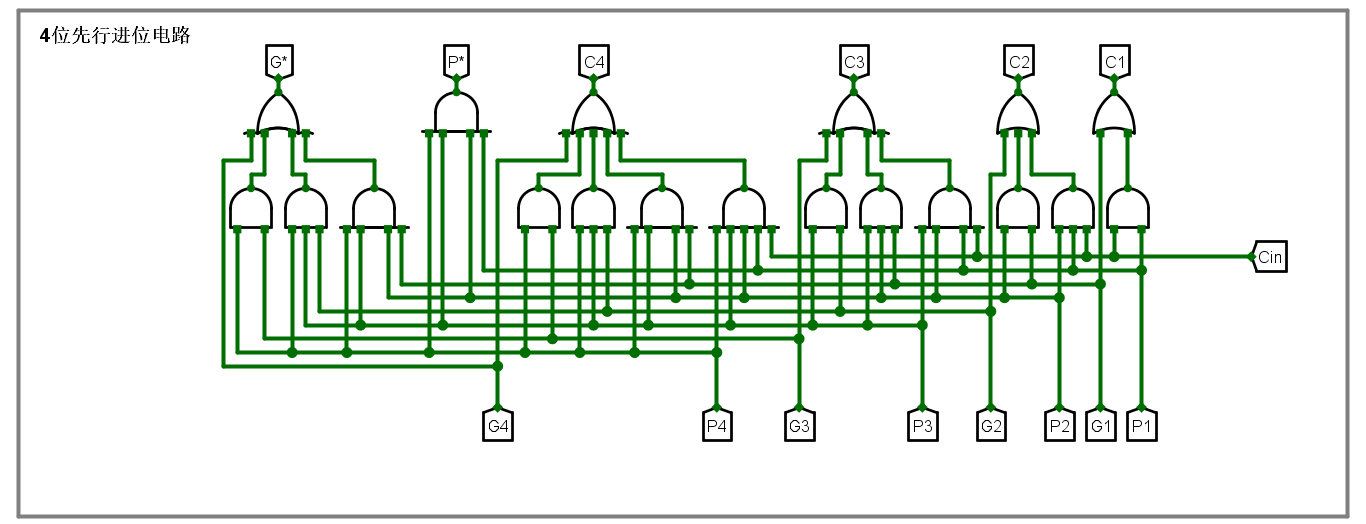
G4\* = G4 + P4G3 + P4P3G2 + P4P3P2G1

P4\* = P4P3P2P1

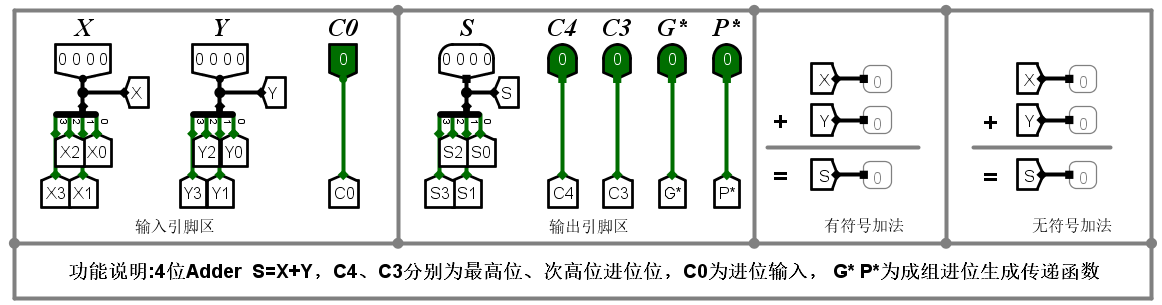
得

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



②先将需要加和的数输入与门异或门电路，得到P和Q

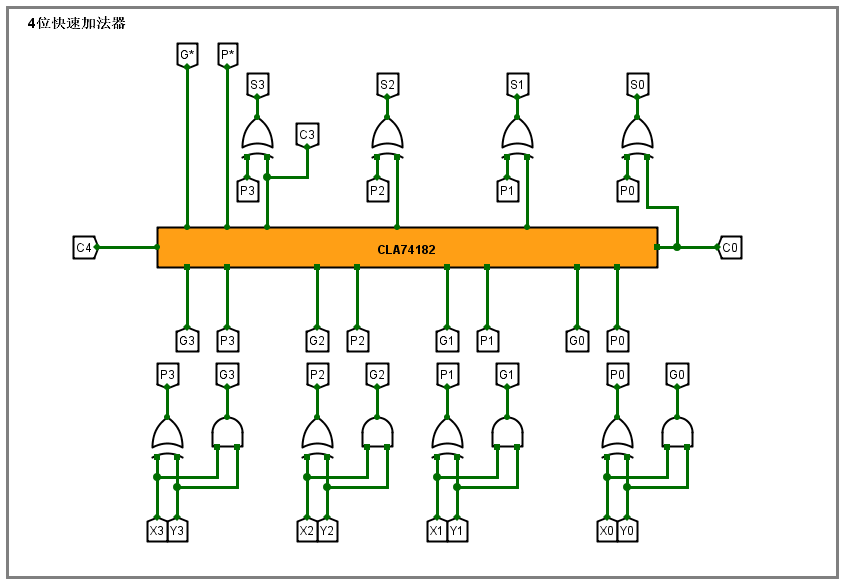
Pi=Xi⊕Yi

Gi=XiYi

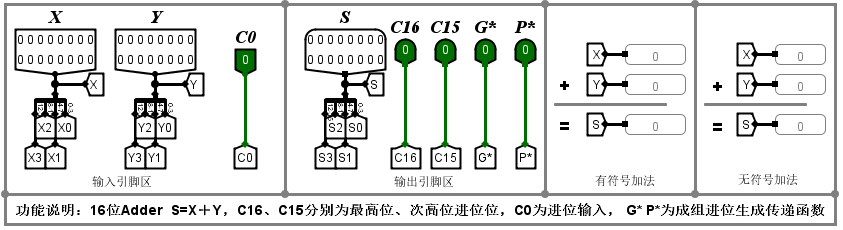
③将得到的P和Q输入四位先行进位电路，得到C

④将得到的C与P异或，得到加和之后的值

#### 3.3.2 电路



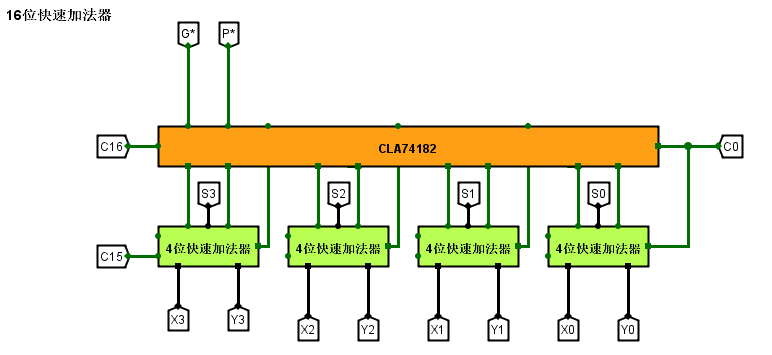
### 3.4 十六位快速加法器设计



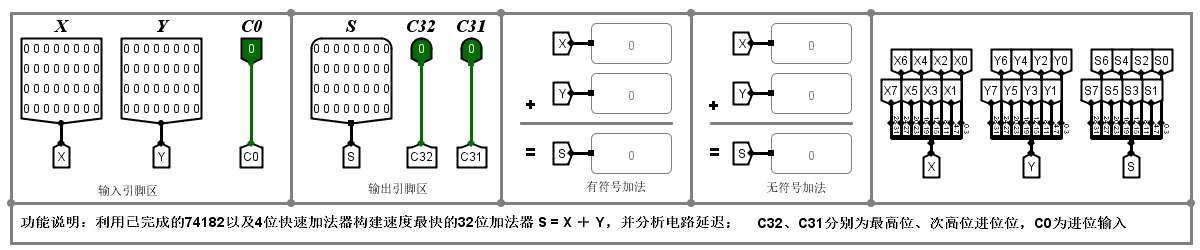
#### 3.4.1 原理

①将4个4位快速加法器输出的成组进位生成、传递函数 和 及 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



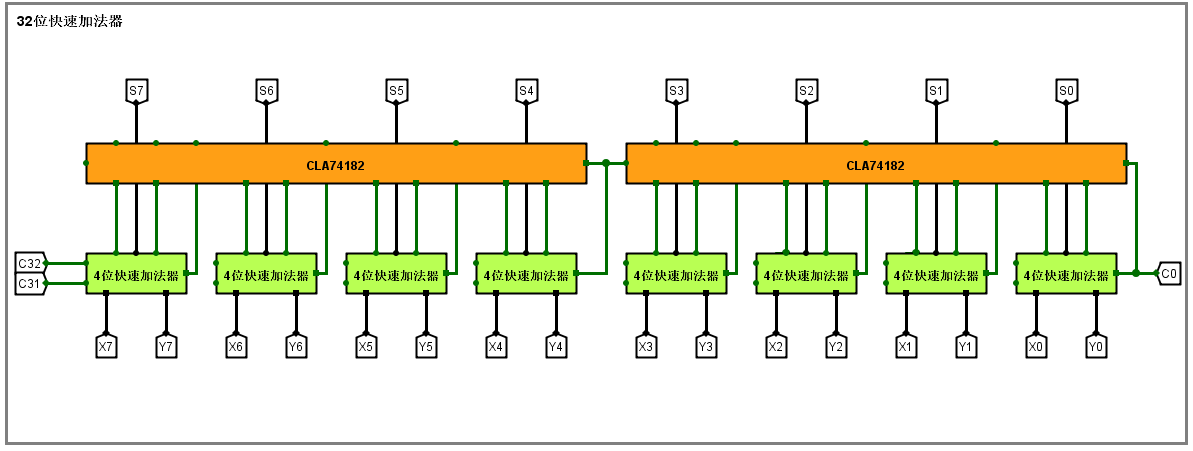
### 3.5 32位快速加法器设计



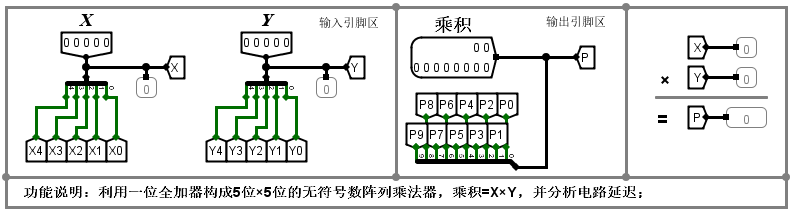
#### 3.5.1 原理

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

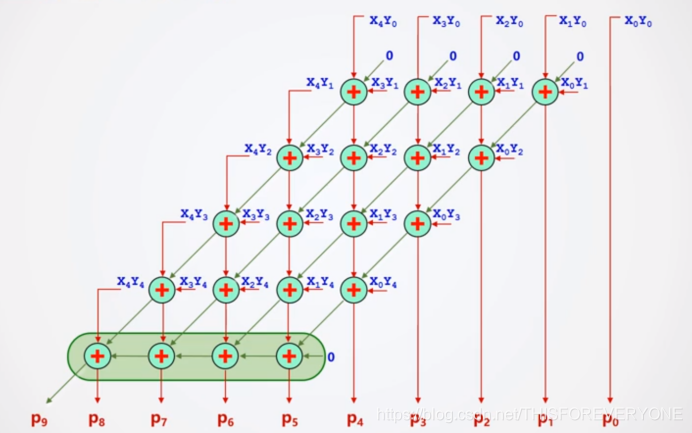


#### 3.6.1 原理

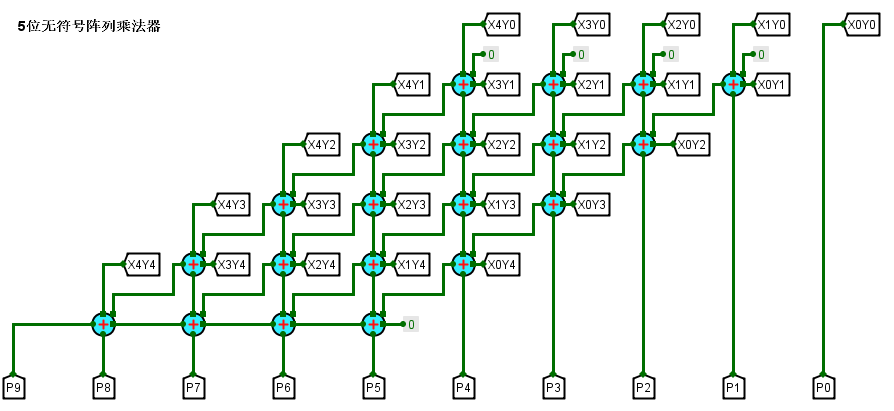
①n位求补器：根据输入的6位补码的符号位决定我们要通过求补码之后的5位还是没有求过补码的原来的数值位。这个就可以用logisim中的5位求补器来实现。

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

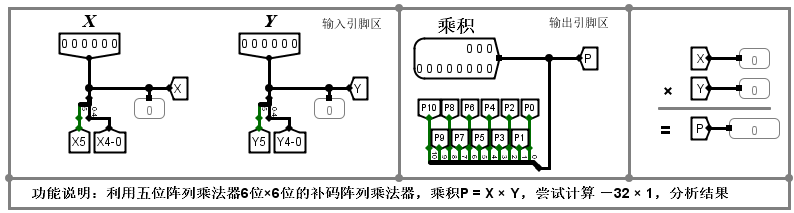
③具体原理如下：



#### 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。

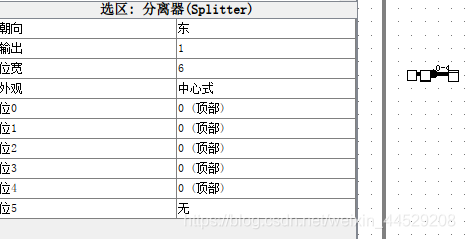
④在6位补码阵列乘法器中利用5位阵列乘法器以及求补器等部件实现补码阵列乘法器

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：



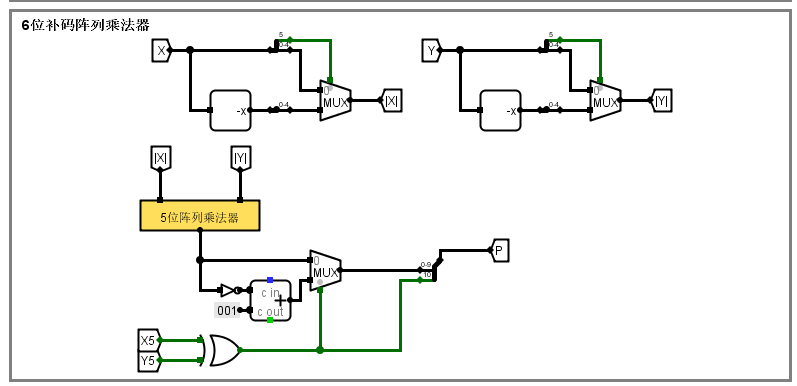
2）分离器：



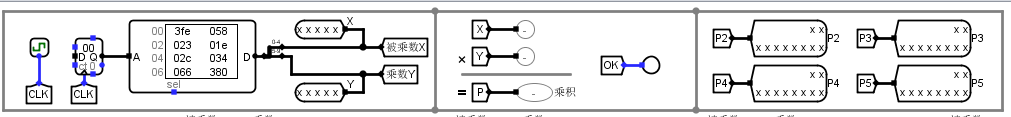
3）补码器：



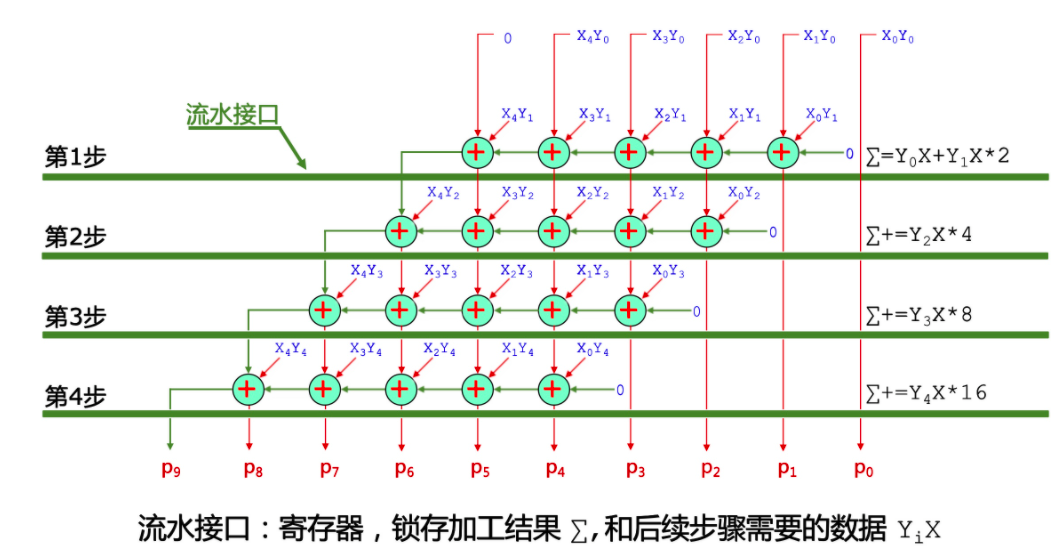
#### 3.7.2 电路



### 3.8 五位无符号乘法流水线电路

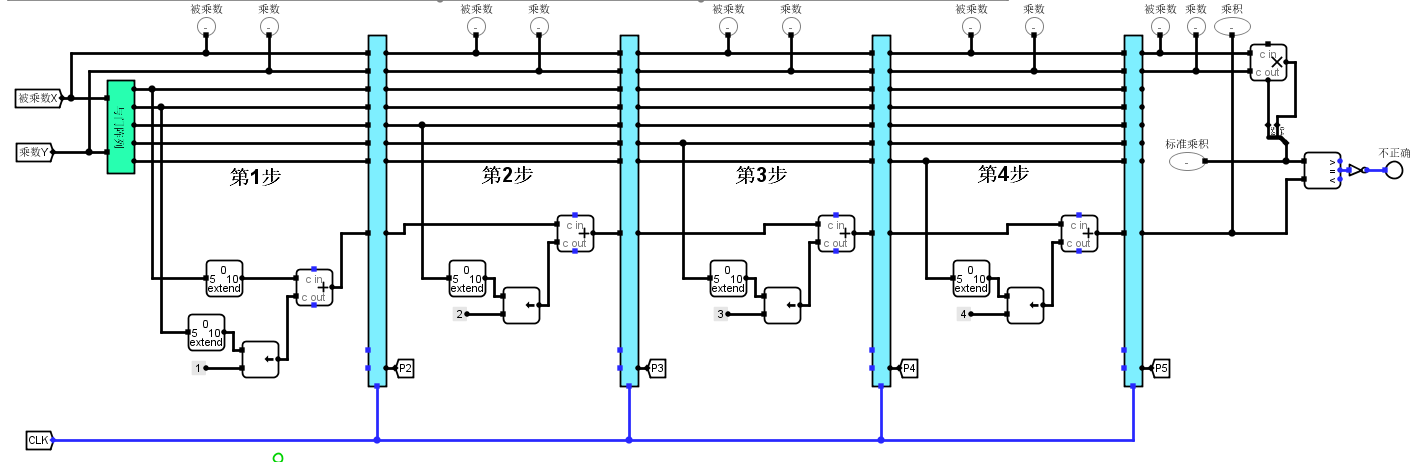


#### 3.8.1 原理

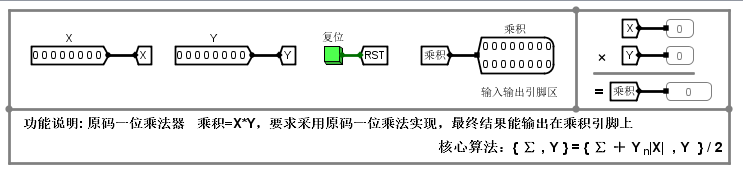


根据上图，我们可以得到流水线电路的设计思路。首先是被乘数X的每一位和乘数Y的最后一位相乘，得到一个部分积，由于结果是一个十位的二进制数，所以需要先将其扩展为十位，再送入下一级。第二级流水线的结果是被乘数X的每一位和乘数Y的倒数第二位相乘，实际结果应该是计算结果的两倍，也就是说第二级计算的结果比第一级高一位，并不能直接相加，所以需要先将第二级的结果左移一位再与第一级的结果相加。三四级的原理同上，最后我们便得到了计算的结果，再与自带的乘法器算得的结果进行比较，便得到了所有电路的设计。

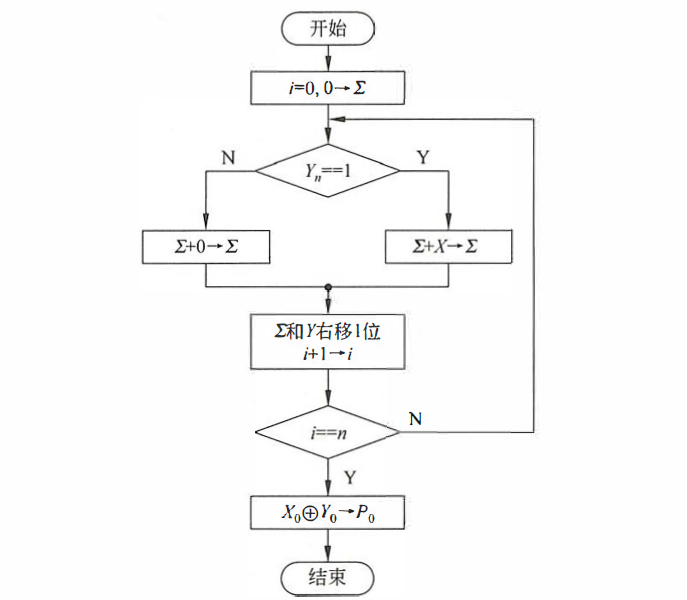
#### 3.8.2 电路



### 3.9 原码一位乘法器

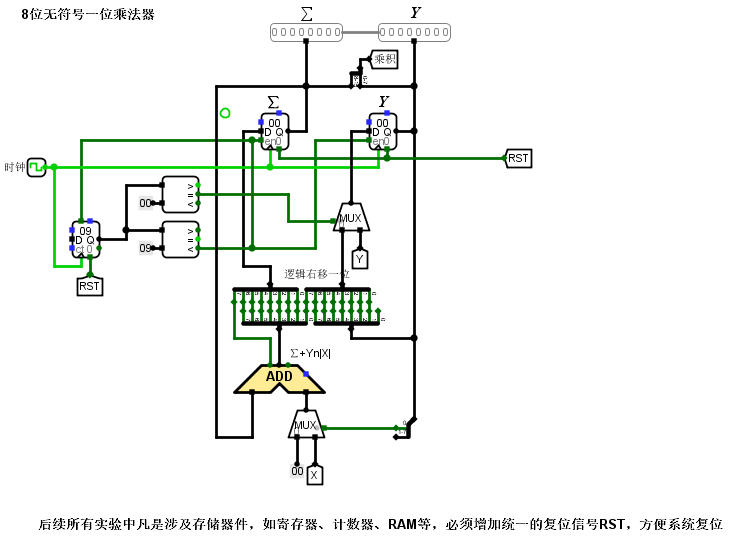


#### 3.9.1 原理

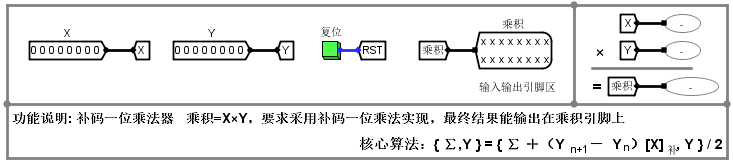


上图是原码1位乘法的流程图，由于原码数值部分和真值相同，所以可以单独计算符号位，利用存放部分积，i为循环计数器，初始值均为零，根据乘数Y的最后一位的值，决定每次累加上0，还是被乘数|X|。运算完毕后，部分积右移1位，Y右移1，然后继续累加运算，当乘数所有数值位均参与运算后，运算结束，得到数值部分的运算结果,最后单独计算符号位即可完成原码1位乘法运算。如果数值部分为n位，需要进行n次加法运算和移位操作。

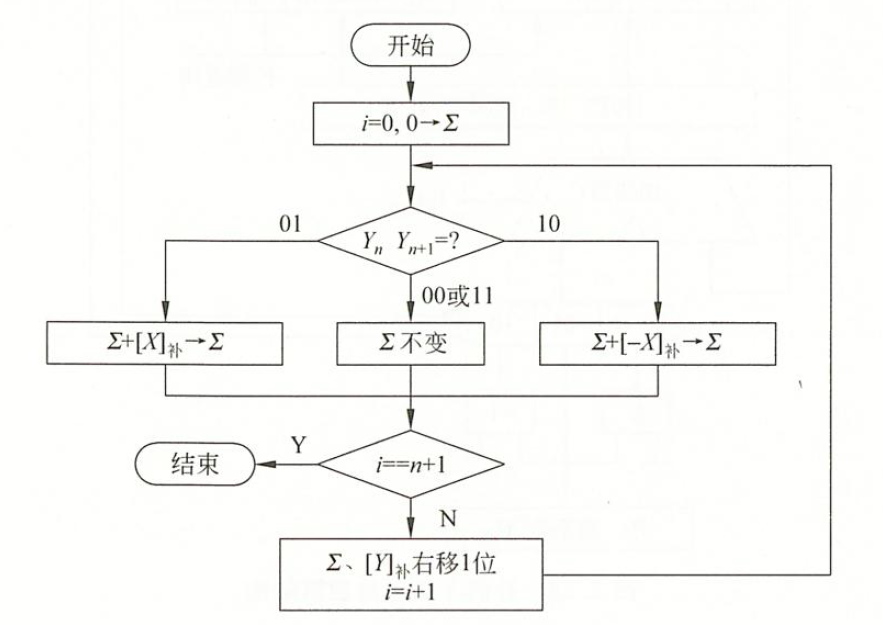
#### 3.9.2 电路



### 3.10 补码一位乘法器

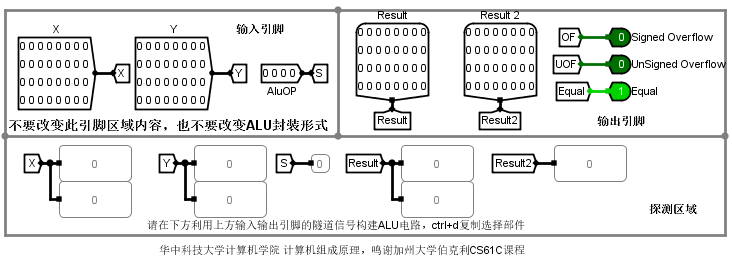


#### 3.10.1 原理



booth 1位乘法中乘数采用双符号位参加运箕 ，符号位也参与运算。 利用存放部分积 ，i为 循环计数器 ， 初始值为零，部分积 累加公式为，根据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：



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。

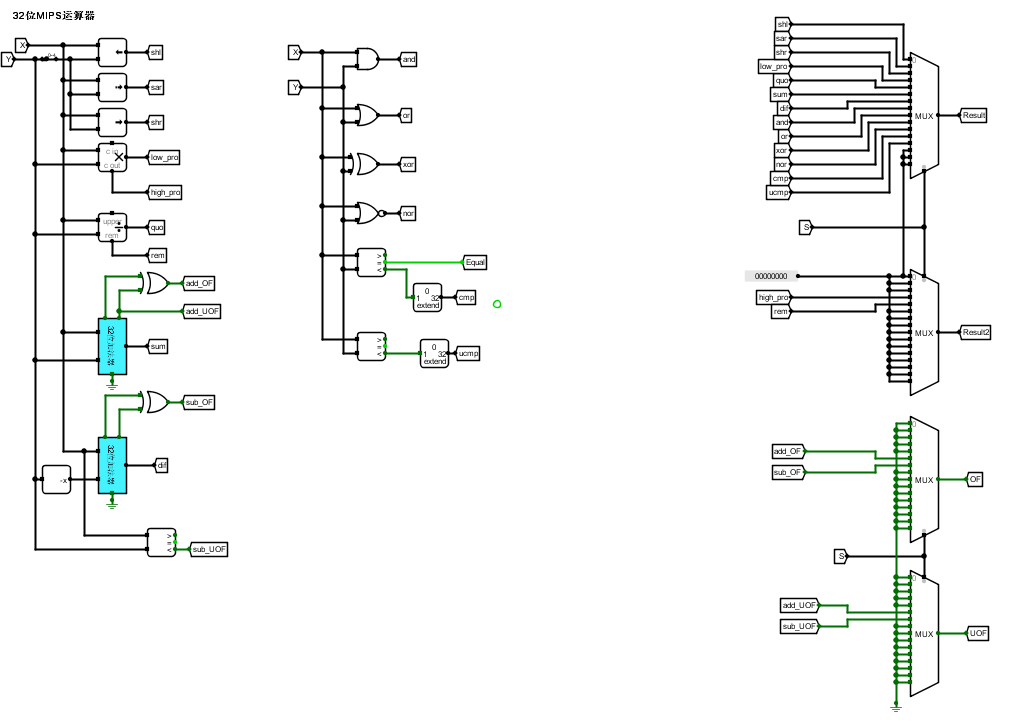
7—10操作：直接用多位的与门/或门/异或门/同或门

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

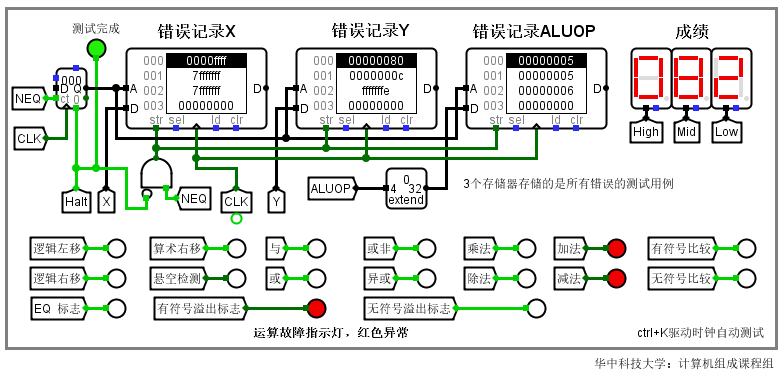


## 4 遇到的问题及解决方案

### 4.1 问题一

#### 4.1.1 问题描述

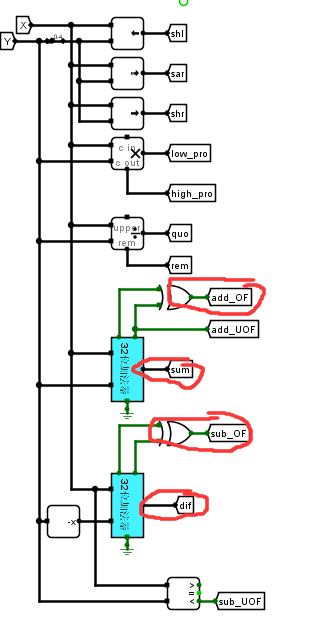
在ALU自动测试实验中，第一次实验得到的结果如下所示



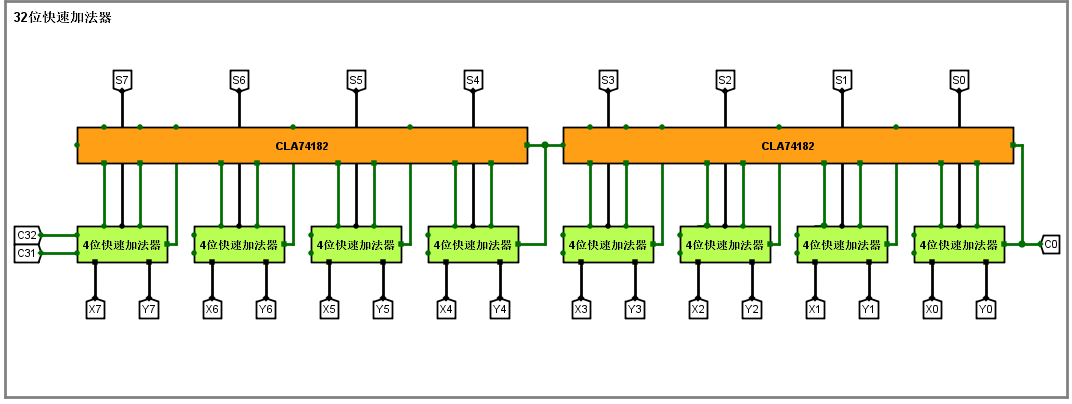
在加法、减法、有符号溢出标志处出现了错误.

#### 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）

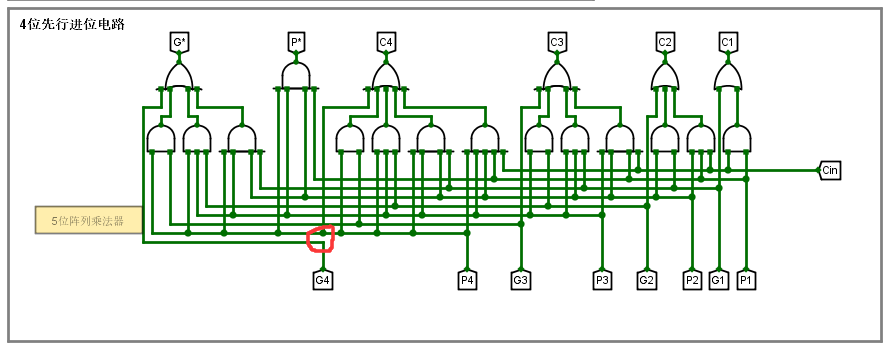


②根据错误提示，说明圈起来的地方出错了，也就是32位加法器内部出错了，于是我们进入32位加法器中去寻找错误

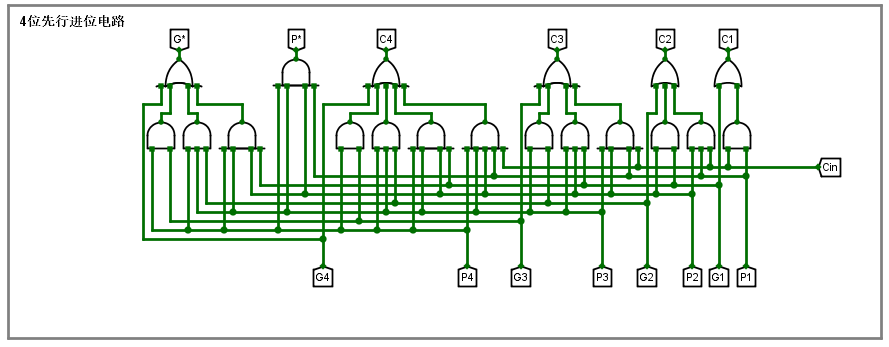


③32位加法器的连接是没有错误的，并且显示是有符号溢出位出现错误，所以我估计问题很有可能出现在74182中，并且溢出判断位很有可能出现错误，于是我们进入74182中寻找错误，特别注意溢出判断位的错误。

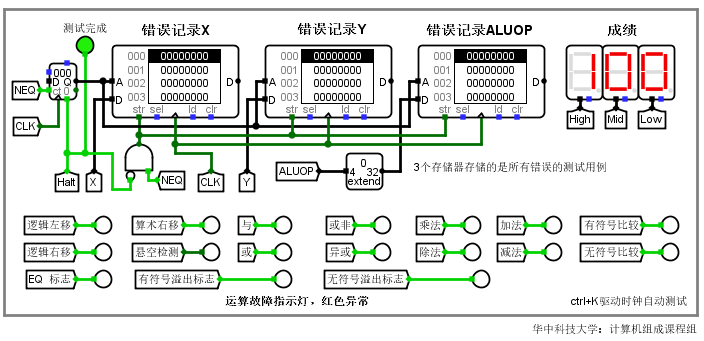
经过仔细地比对，发现C4有一处出现连接错误，如图：



修正后的连线为：



#### 4.1.3 修正后结果



如图所示：修正后结果正确