运算器实验报告书

学　　院：人工智能与自动化学院

班　　级：人工智能 2004 班

姓　　名：陈乃睿

学　　号：U202012593

实验时间：2022 年 5 月 27 日

目录

一、实验名称

运算器实验
1. 加法器实验
2. 阵列乘法器实验
3. 一位乘法器实验
4. ALU 实验

二、实验目的

1. 加法器实验:

(1) 掌握 1 位全加器的实现逻辑，掌握多位可控加减法电路的实现逻辑，熟悉 Logisim 平台基本功能，能在 Logisim 中实现多位可控加减法电路。

(2) 掌握快速加法器中先行进位的原理，能利用相关只是设计 4 位先行进位电路，并利用设计的 4 位先行进位电路构造 4 位快速加法器，能分析对应电路的时间延迟。

(3) 理解成组进位产生函数、成组进位传递函数的概念，熟悉 Logisim 平台子电路的概念，能利用前述实验封装好的 4 位先行进位电路以及 4 位快速加法器子电路构建 16 位、32 位、64 位快速加法器，并能利用相关知识分析对应电路的时间延迟，理解电路并行的概念。

2. 阵列乘法器实验：

掌握阵列乘法器的实现原理，能够分析阵列乘法器的性能，能在 Logisim 中绘制阵列乘法器电路。

3. 一位乘法器实验：

(1) 掌握原码 1 位乘法运算的基本原理，熟练掌握 Logisim 中寄存器组件的使用，能在 Logisim 平台中设计实现一个 8 位×8 位的无符号数乘法器。

(2) 掌握补码 1 位乘法运算的基本原理，熟练掌握 Logisim 寄存器组件的使用，能在 Logisim 平台中设计实现一个 8 位×8 位的定点数补码 1 位乘法器。

4. ALU 实验：

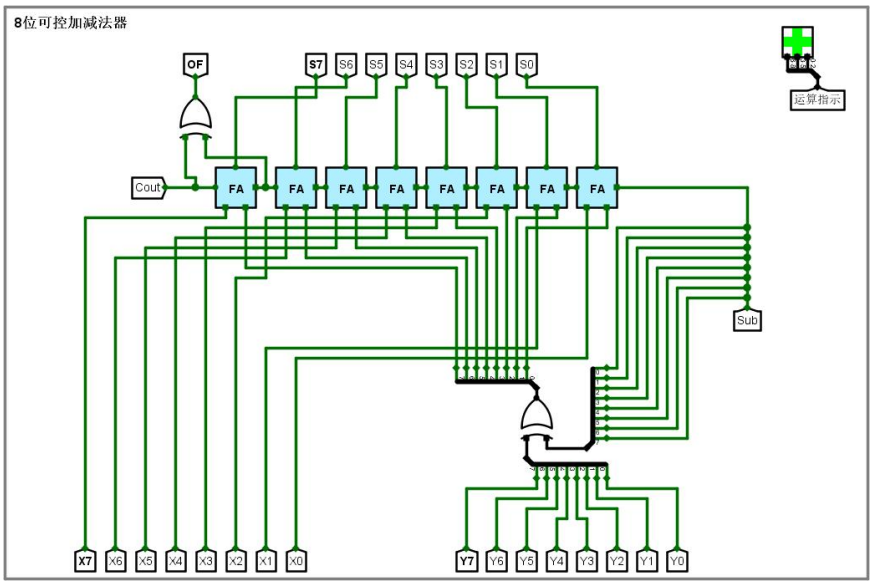
理解算术逻辑运算单元（ALU）的基本构成，掌握 Logisim 中各种运算组件的使用方法，熟悉多路选择器的使用，能利用前述实验完成的 32 位加法器和 Logisim 中的运算组件构造指定规格的 ALU 单元。

三、实验内容

1. 加法器实验：
   (1) 在 Logisim 模拟器中打开 alu.circ 文件，在对应的子电路中利用已经封装好的全加器设计 8 位串行可控加减法电路。
   (2) 在 Logisim 中打开 alu.circ 文件，在对应子电路中实现可级联的 4 位先行进位电路，再利用 4 位先行进位电路构造 4 位快速加法器。
   (3) 在 Logisim 中打开 alu.circ 文件，在对应的子电路中利用 4 位先行进位电路和 4 位快速加法器构造 16 位组间先行进位、组内先行进位快速加法器，并验证其功能是否正常。
   (4) 用两个 16 位加法器直接串联，成为 32 位快速加法器。
2. 阵列乘法器实验：
   在 Logisim 中打开 alu.circ 文件，在 5 位阵列乘法器中实现斜向进位的阵列乘法器。
3. 一位乘法器实验：
   (1) 在给出的文件中增加控制电路和数据通路使得该电路能自动完成 8 位无符号的 1 位乘法运算，首先设置引脚初始值，然后驱动时钟自动仿真，电路可自动完成运算，运算结束结果传输到输出引脚，运算结束时电路应该自动停止。
   (2) 在给出的文件中增加控制电路和数据通路，使得该电路能自动完成 8 位补码 1 位乘法运算。首先设置引脚初始值，然后驱动时钟仿真，电路自动完成运算，运算结束则结果传输到输出引脚，运算结束时电路应该自动停止。
4. ALU 实验：
   利用前面实验封装好的 32 位加法器以及 Logisim 平台中现有运算组件构建一个 32 位算术逻辑运算单元（禁用 Logisim 系统自带的加法器，减法器），可支持算术加、减、乘、除、逻辑与、或、非、异或、逻辑左移、逻辑右移，支持常用程序状态标志（有符号溢出 OF、无符号溢出 UOF、结果相等 Equal）

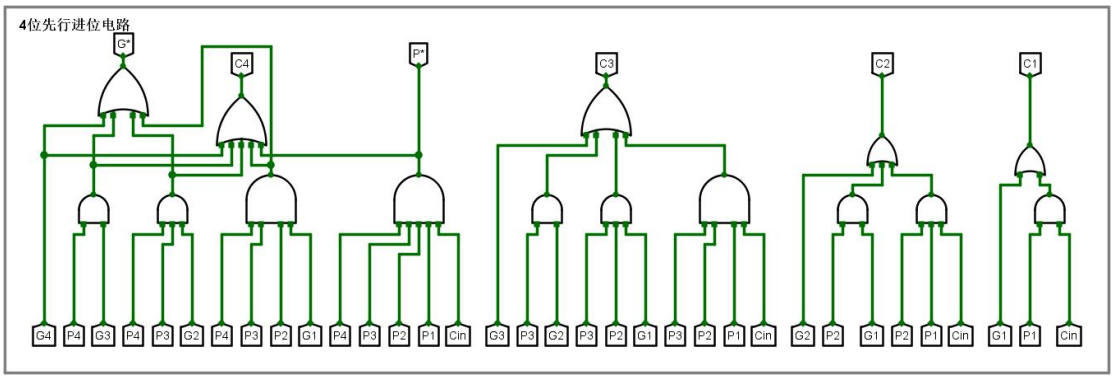
四、实验电路与设计思路

1. 加法器实验
   (1) 八位可控加减法器



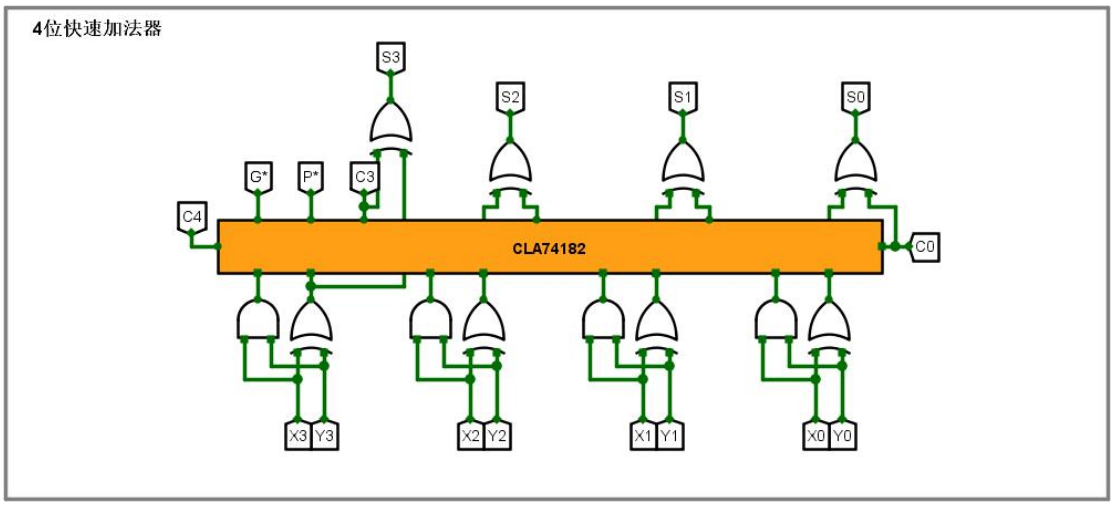
sub 标志着该运算器是运算加还是减。若 sub=0，则为加法，将 x(i)与 y(i)分别相加，输出加完后的数据，并将进位位传入 x(i+1)和 y(i+1)的加法中；若 sub=1，则为减法，此时 Y 应变为相反数，所有位数应当取反，并在最后加上 1，该“1”由 sub 数据直接传入 x(0)与 y(0)加法器的进位位中即可。

(2) 四位先行进位 74182：



按照书上推导的公式，用 P、Q 来计算每个位数的进位，以方便后续四位数加法的直接计算。

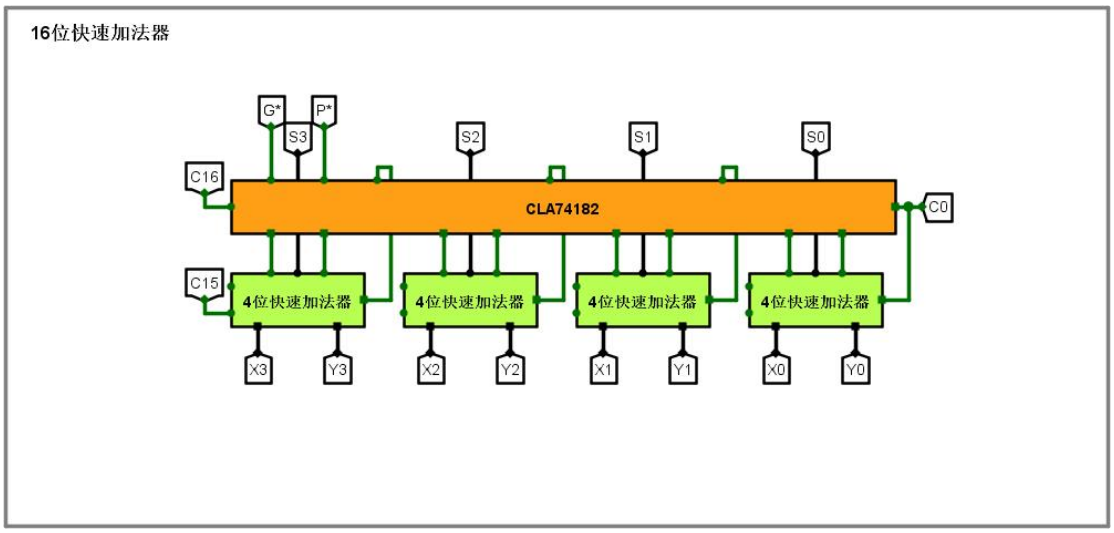
(3) 四位快速加法器：



使用公式$S_i = \mathrm{C_i} \oplus \mathrm{P_i}$计算出加法运算后的每一位，并传出$P^*$与$Q^*$以方便

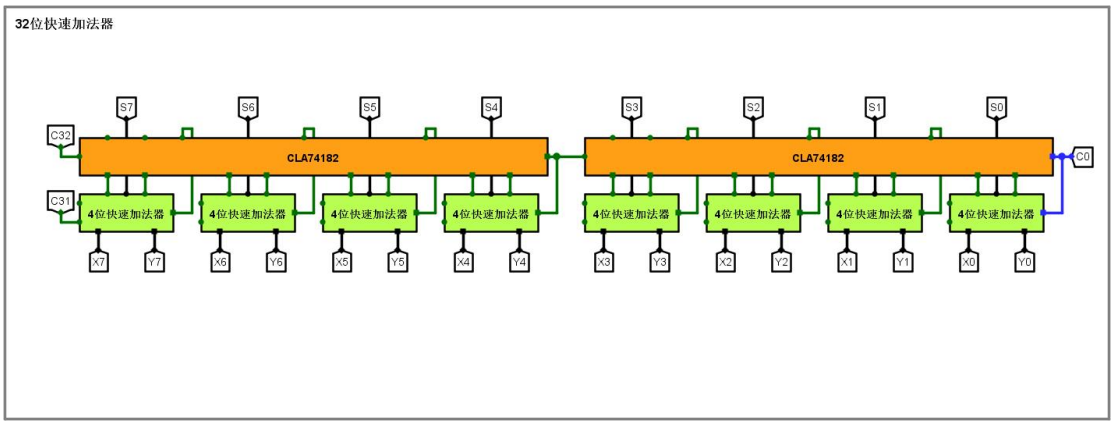
后续 16 位加法运算的进位计算。

(4) 16 位快速加法器：



仿照 4 位快速加法器，不同的是加法运算后的每一个单位（每 4 位数为一个单位）由 4 位快速加法器得出，且进位位由P^*与Q^*计算得出。

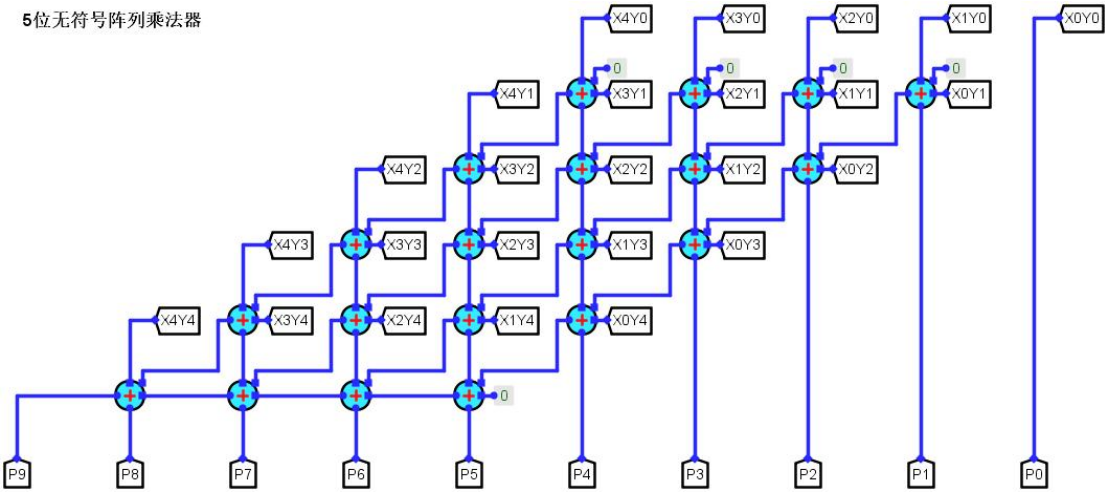
(5) 32 位快速加法器：



用两个 16 位快速加法器相连，其中要将低位的 16 位加法器的溢出（进位）信息传递到高位的 16 位加法器中，应传入到快速进位器的初始进位位以及第一个 4 位快速进位器的进位位。最后引出引脚 C31 与 C32，用于后续实验电路中溢出的判断。

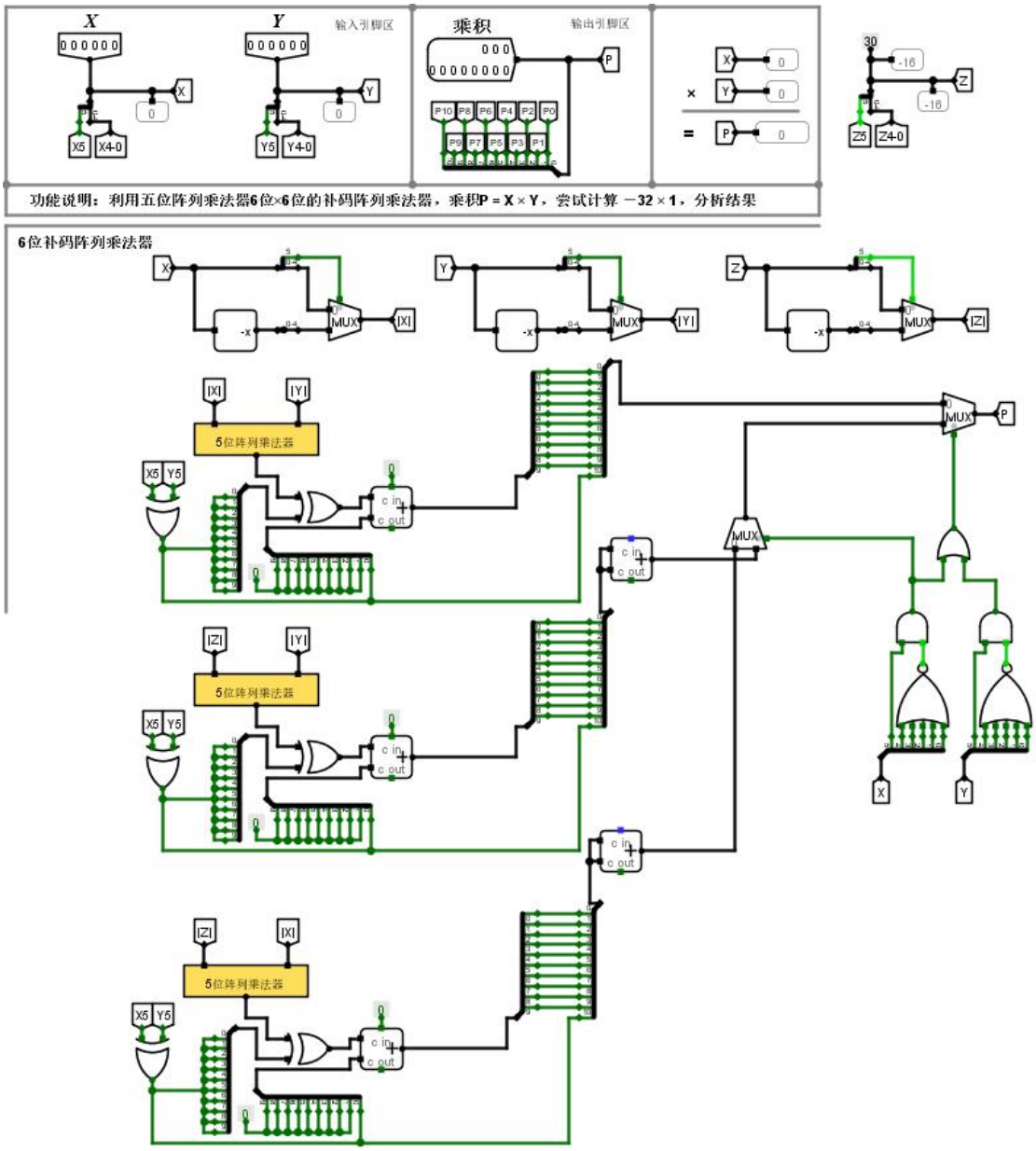
2. 阵列乘法器实验

(1) 5 位阵列乘法器：



该乘法器的原理和列竖式计算乘法的原理一模一样，将各行错开相加，将进位位传入更高位的列进行计算，最后得出结果。

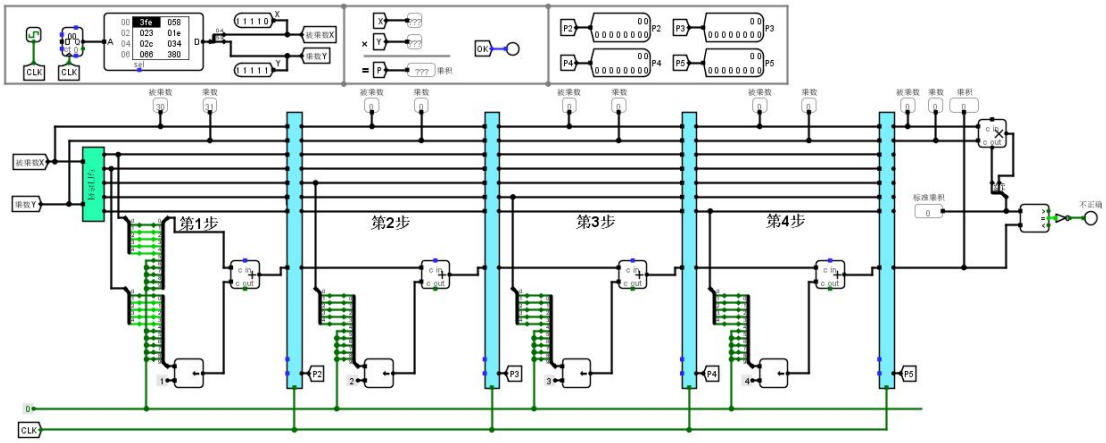
(2) 6 位补码阵列乘法器：



将两乘数 X、Y 分为符号位与数据位，符号位（X5、Y5）用于确认结果的符号，数据位用于计算结果的数值。若结果符号为正，则 X5X6 的符号相同，异或值为 0，直接传入 10 位结果与 1 位符号即可；若结果符号为负，则 X5X6 符号不同，异或值为 1，数据位应当取反后+1，即：乘法器出来后的数据位每一位与 1 异或后总体数值+1 得到最终的数据位，1 传入到符号位。

其中，当 X、Y 其中有一个补码为“100000”（-32）时，结果会发生错误。此时，可将-32 拆成-16-16 来进行计算，结果乘 2 后传出即可。

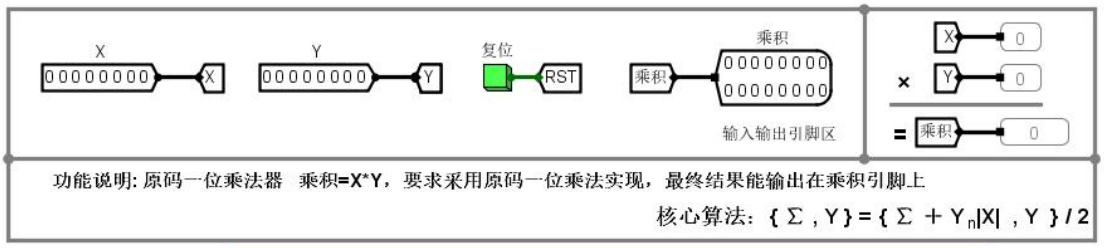
(3) 5 位无符号乘法流水线：

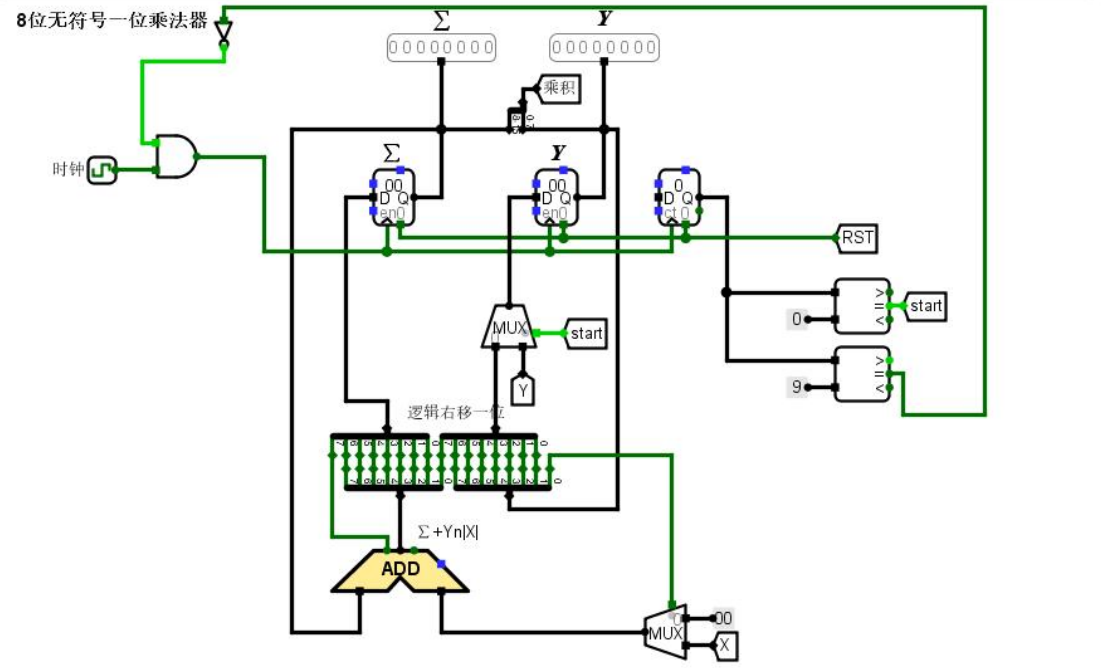


该电路分为上下两部分，上半部分传输正确的乘法计算式，下半部分用5位无符号乘法器的方法自己连接乘法电路，最终在最右边进行判断。

3. 一位乘法器实验

(1) 原码一位乘法器：



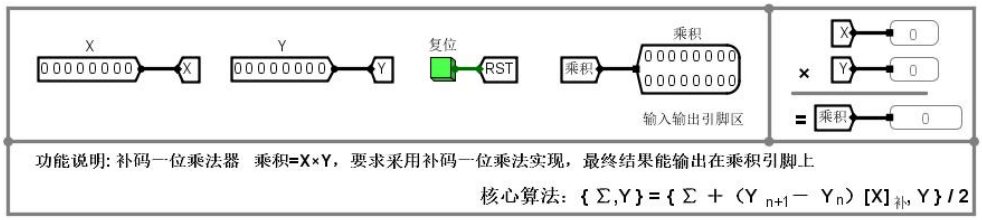


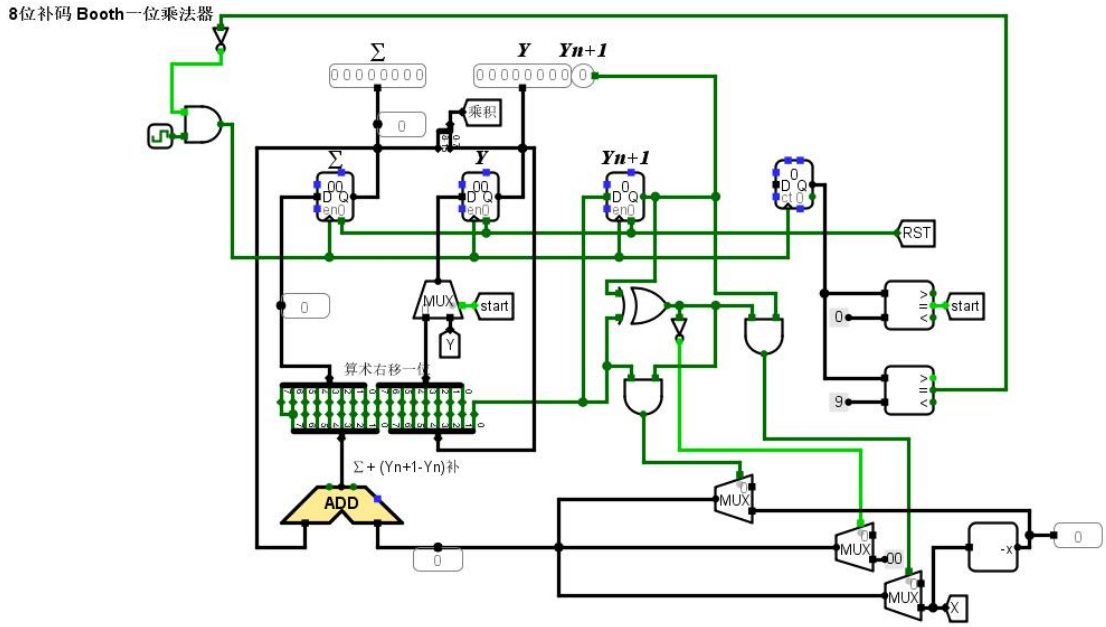
后续所有实验中凡是涉及存储器件，如寄存器、计数器、RAM等，必须增加统一的复位信号RST，方便系统复位

原码一位乘法器的运行原理和5位数阵列乘法器相似，不同的是该乘法器是一行一行按顺序加出结果。初始累加和为0，每次循环进行一次判断：

该次计算时 Y 的最后一位为 0 还是 1？若为 0，则累加和加上 0；若为 1，则累加和加上 X。此后，累加和与 Y 均右移一位，累加和右移是等价于列竖式时每一行左移；Y 右移是为了方便取出 Y 的下一位（变为最后一位）。其中，累加和右移后，最高位应补齐加法器的溢出位。第一次时钟周期时，应将 Y 的初始值送入循环；当再进行八次时钟周期后，即 9 次时钟周期后，加法已经完成，此时应将时钟信号屏蔽，防止乘法器继续变化。

(2) 补码一位乘法器：

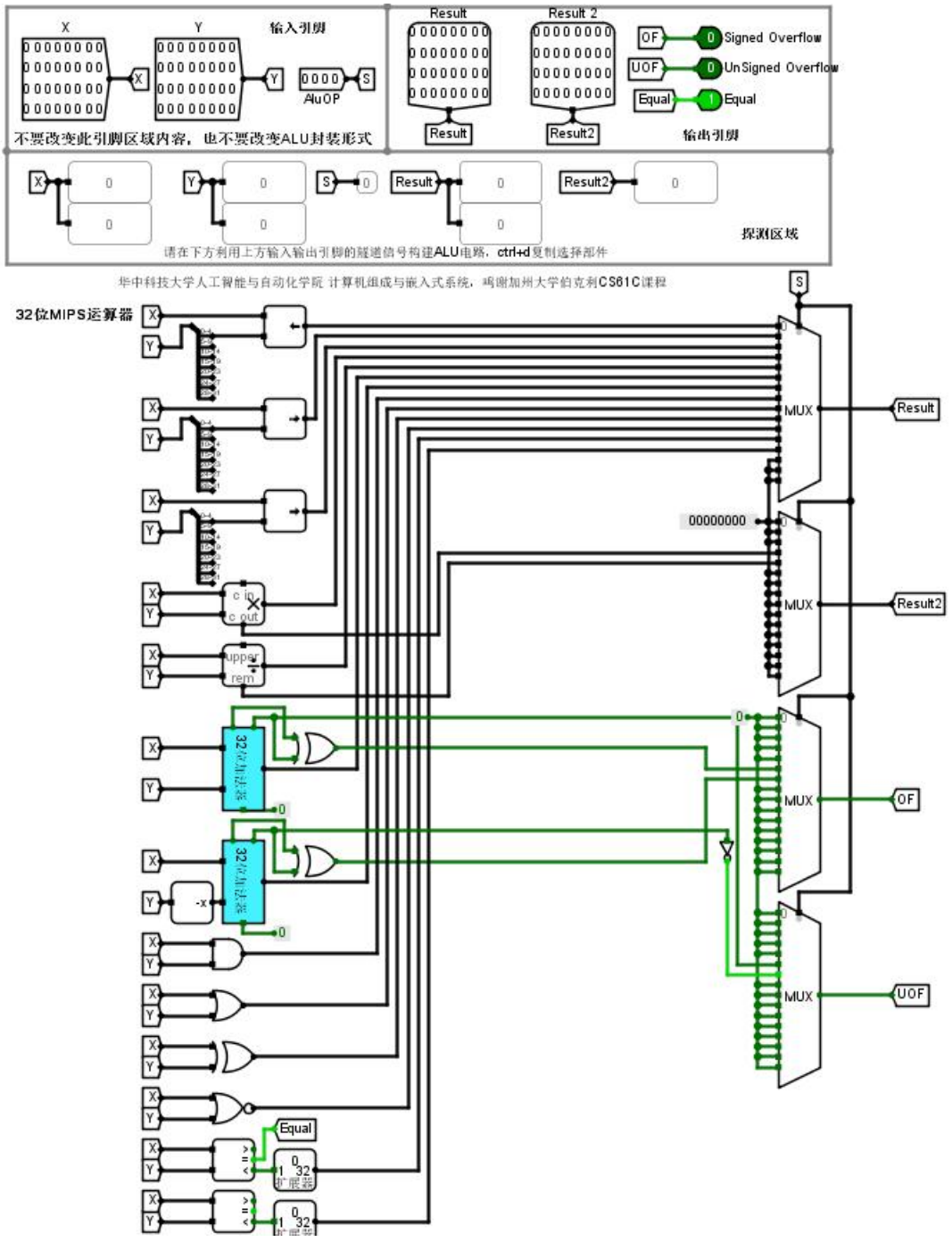




补码一位乘法器的思想和原码一位乘法器的思想大同小异，主要不同是补位以及加数的不同。当 Yn+1 与 Yn 相同时，加数为 0；当不同时，若 Yn+1 为 1，则加数应为 X 的补码；若 Yn 为 1，则加数应为-X 的补码。当累积和右移时，最高位补充的不再是加法器的溢出位，而是补平移前的符号位。其余线路与原码一位乘法器相同。

4. ALU 实验

算术逻辑运算单元 ALU：

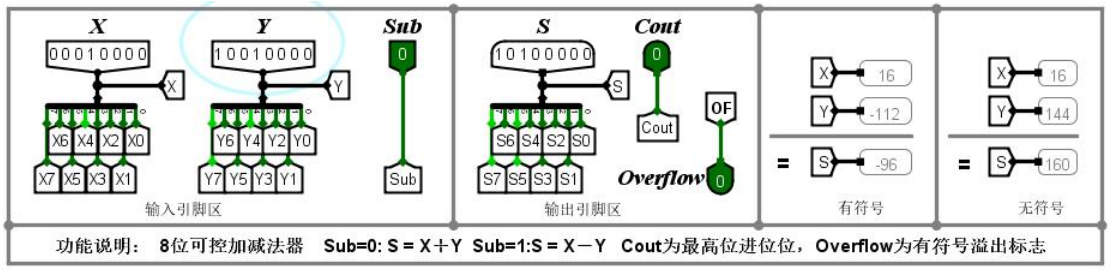


该实验要求连成一个能实现多个功能的运算器，只需分别完成各个功能的电路后，用译码器选择正确的通道输出即可。

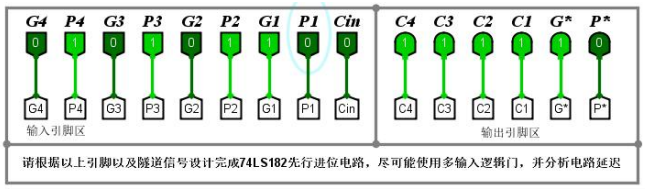
五、实验结果

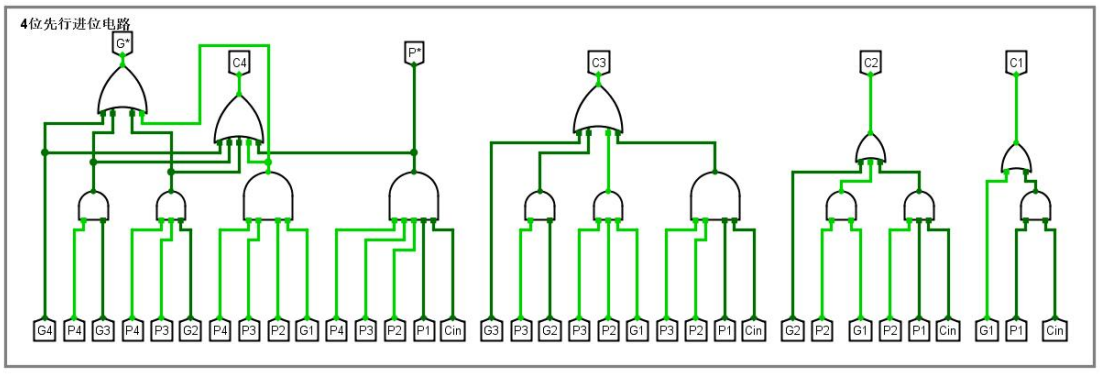
1. 加法器实验

(1) 八位可控加减器：

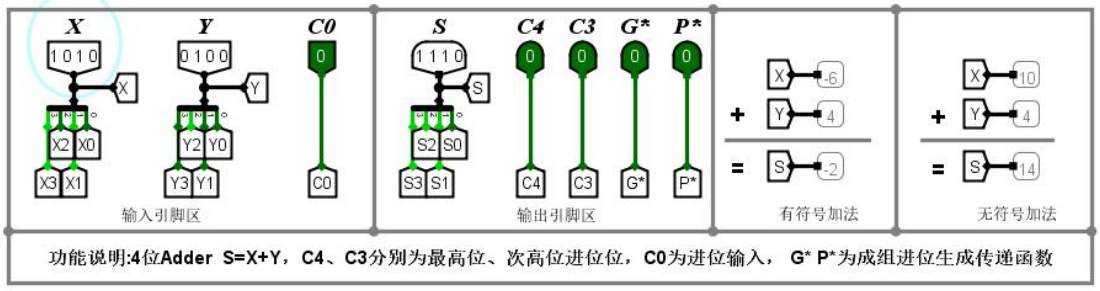


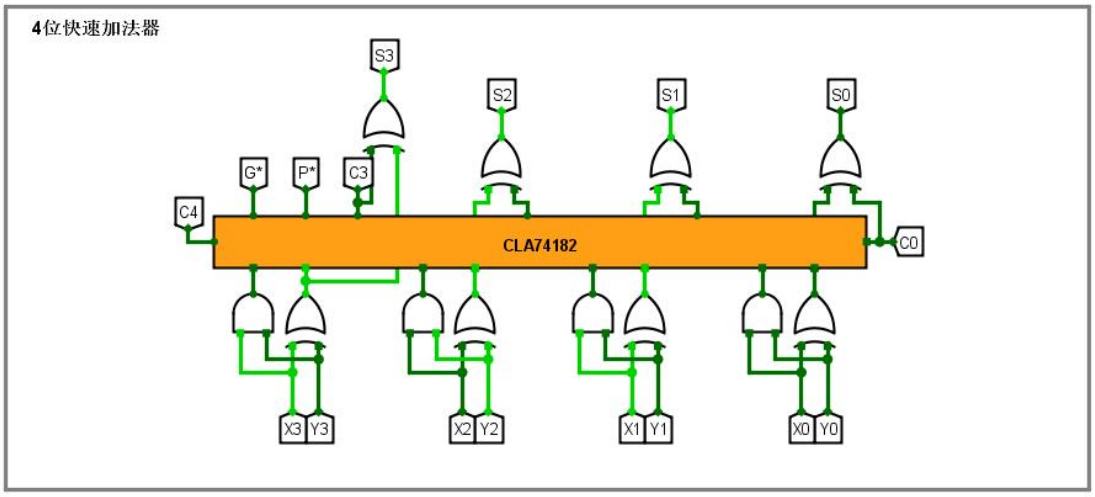
(2) 4 位先行进位 74182：



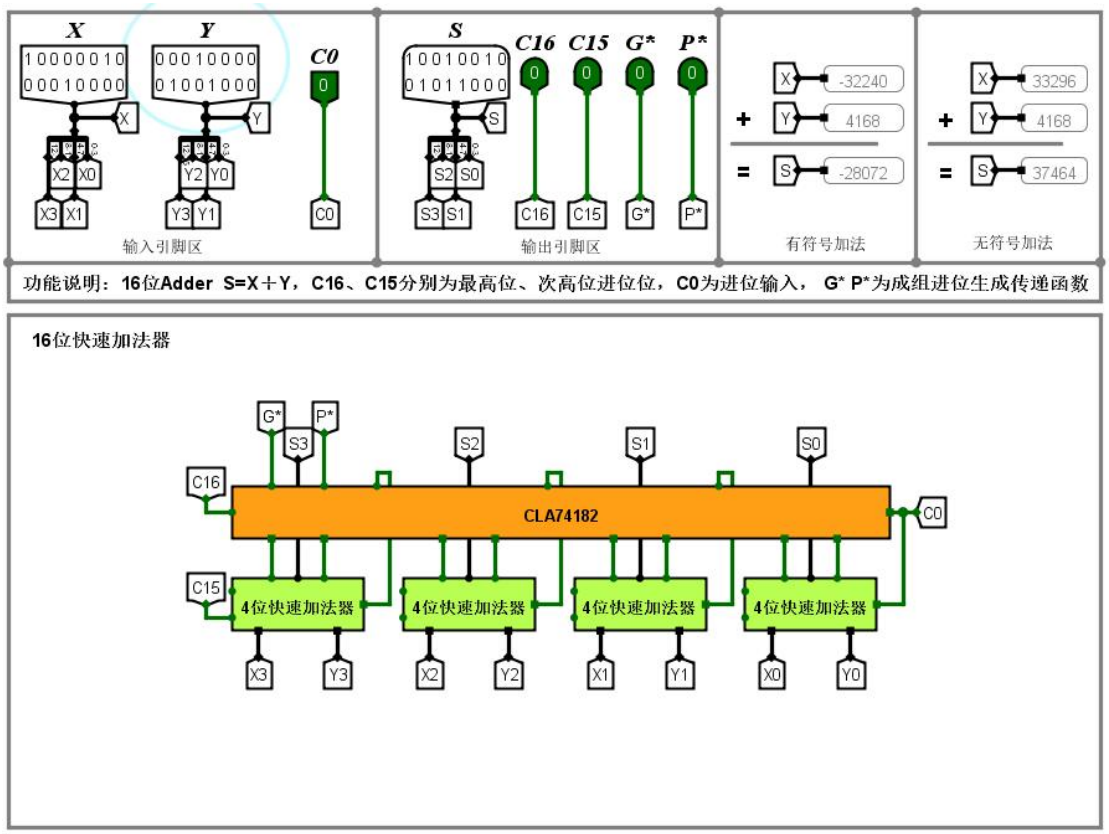


(3) 4 位快速加法器：

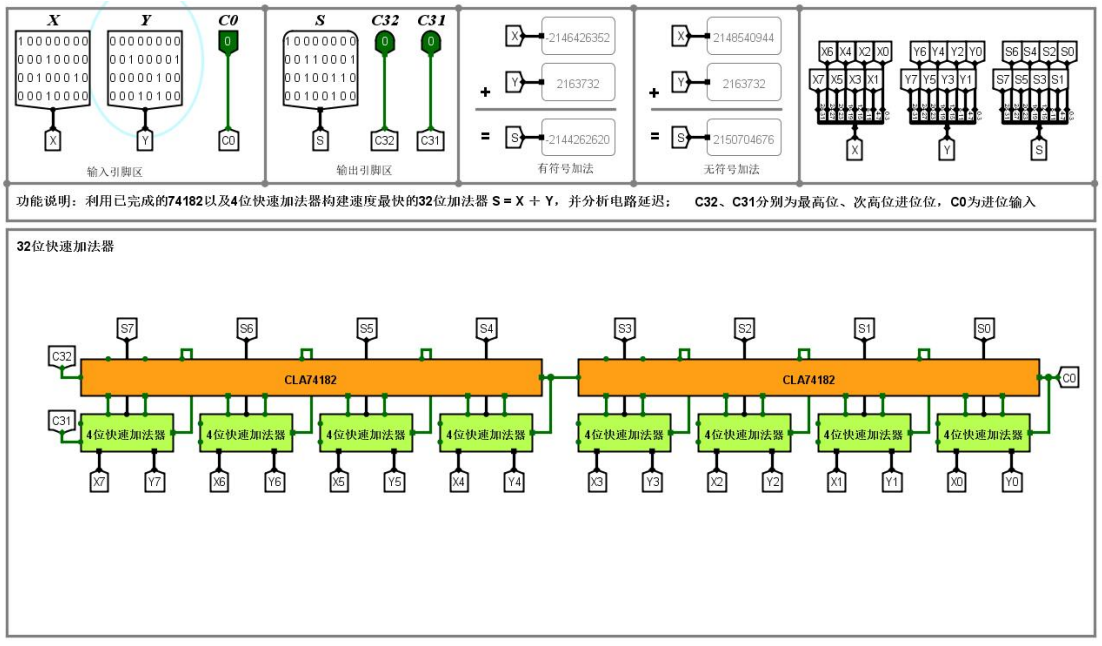




(4) 16 位快速加法器：

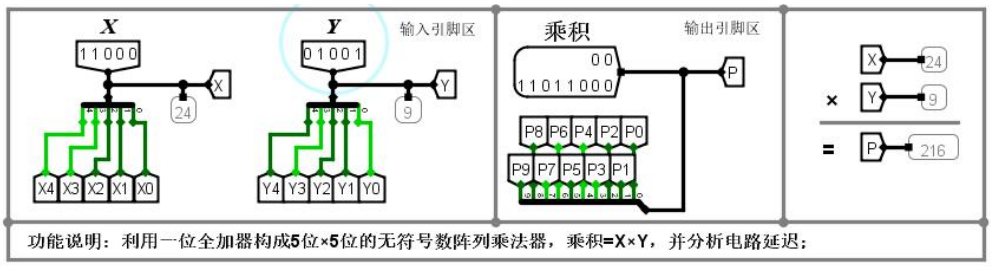


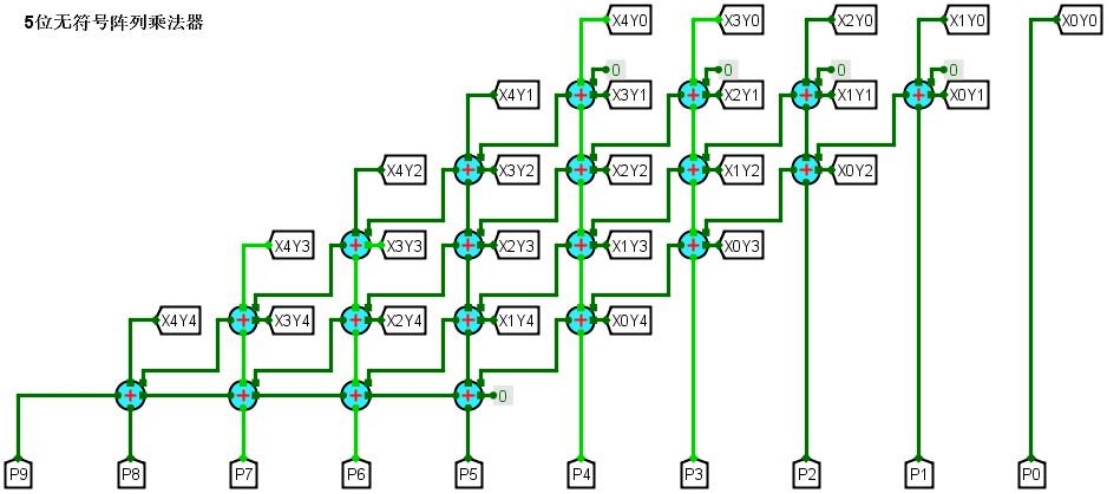
(5) 32 位快速加法器：



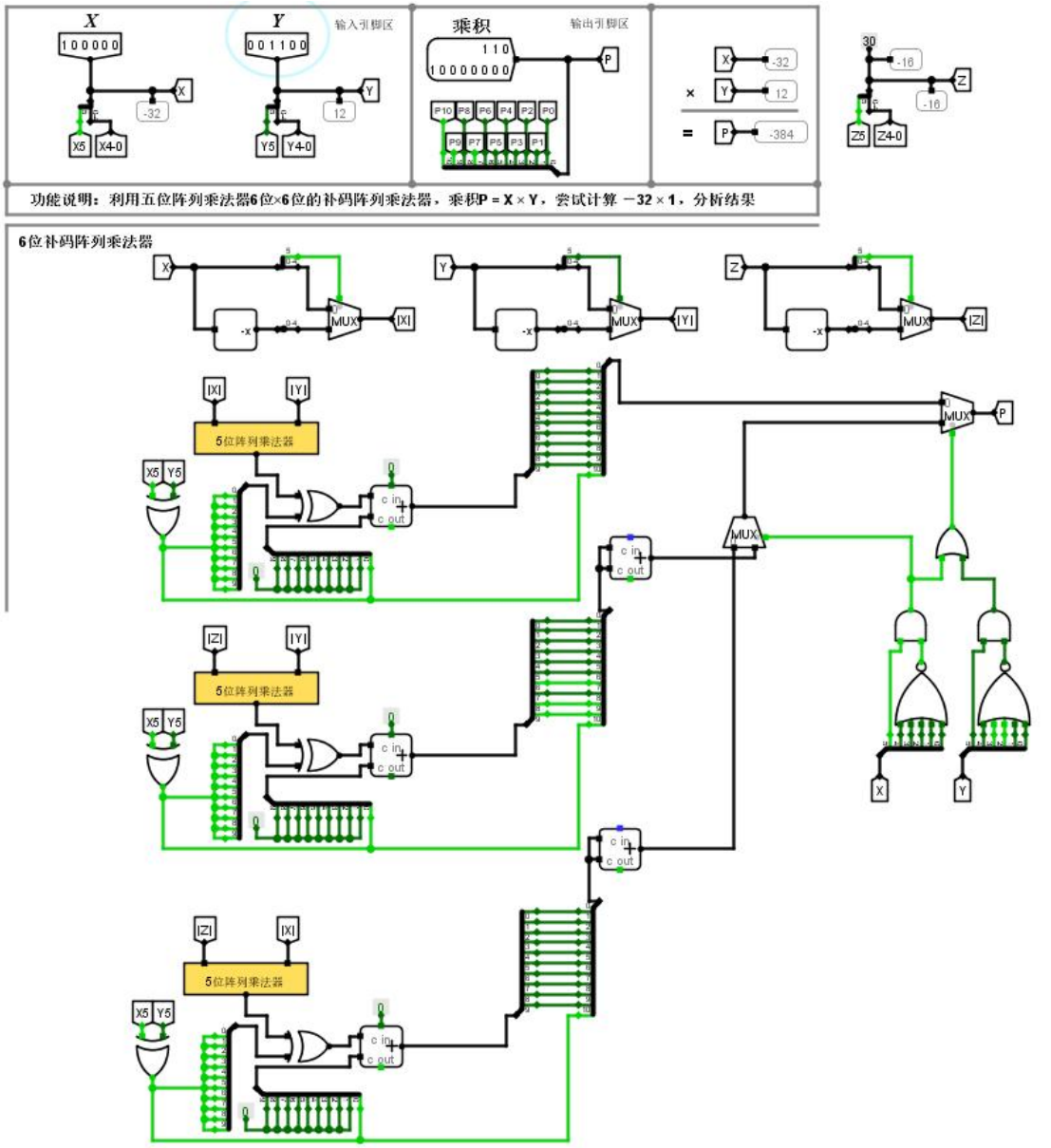
2. 阵列乘法器实验

(1) 5 位阵列乘法器：

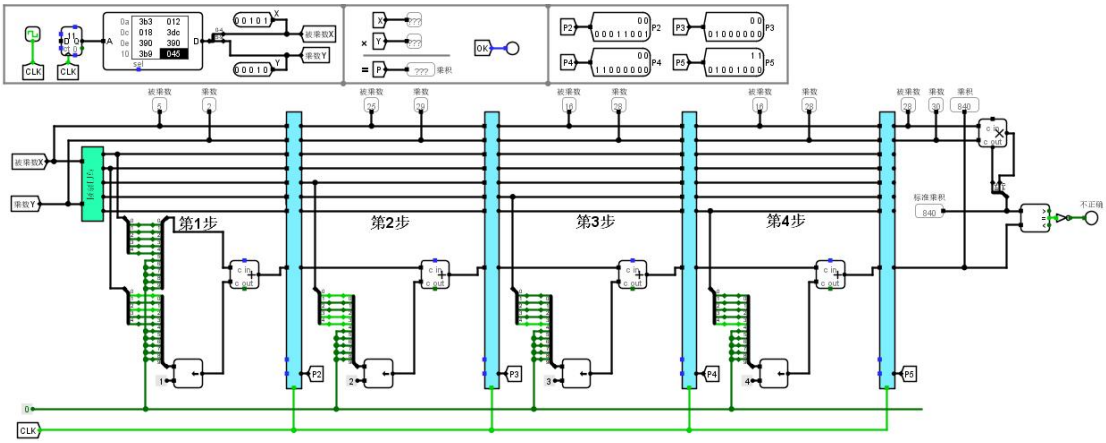

X
11000
X
24
Y
01001
Y
9
输入引脚区
乘积
00
11011000
P
输出引脚区
P8 P6 P4 P2 P0
P9 P7 P5 P3 P1
X4 X3 X2 X1 X0
Y4 Y3 Y2 Y1 Y0
×
=
216
功能说明：利用一位全加器构成5位×5位的无符号数阵列乘法器，乘积=X×Y，并分析电路延迟；


5位无符号阵列乘法器
X4Y0
X3Y0
X2Y0
X1Y0
X0Y0
X4Y1
X3Y1
X2Y1
X1Y1
X0Y1
X4Y2
X3Y2
X2Y2
X1Y2
X0Y2
X4Y3
X3Y3
X2Y3
X1Y3
X0Y3
X4Y4
X3Y4
X2Y4
X1Y4
X0Y4
0
P9
P8
P7
P6
P5
P4
P3
P2
P1
P0

(2) 6 位补码阵列乘法器：

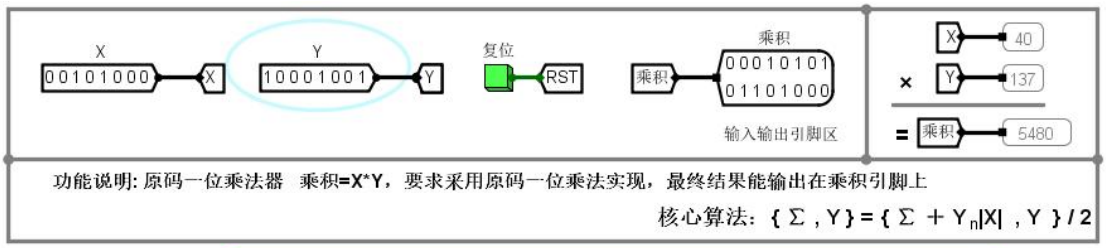


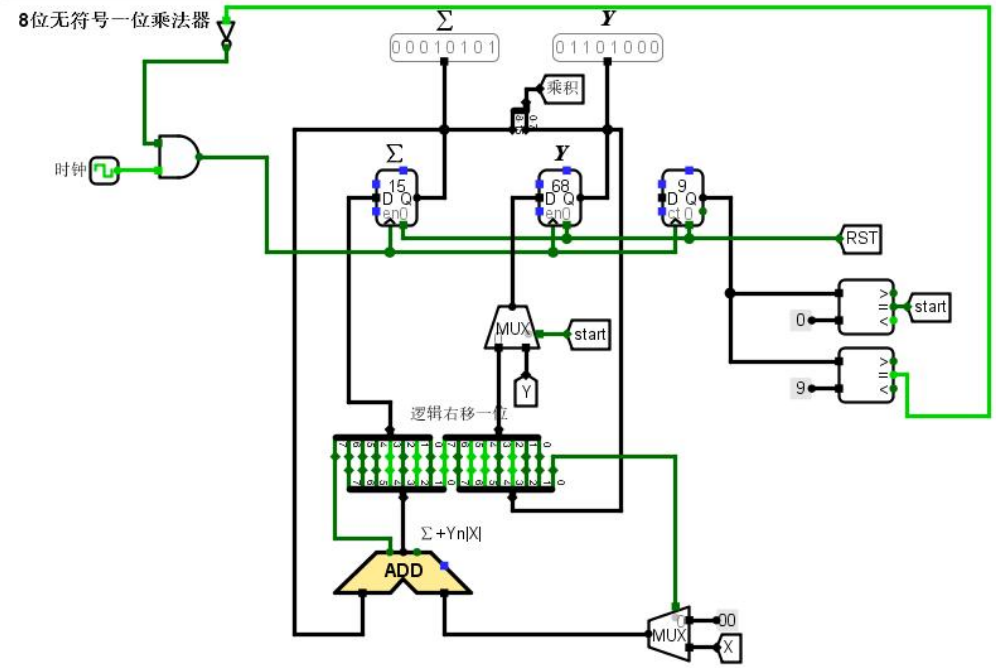
(3) 5 位无符号乘法流水线：



3. 一位乘法器实验

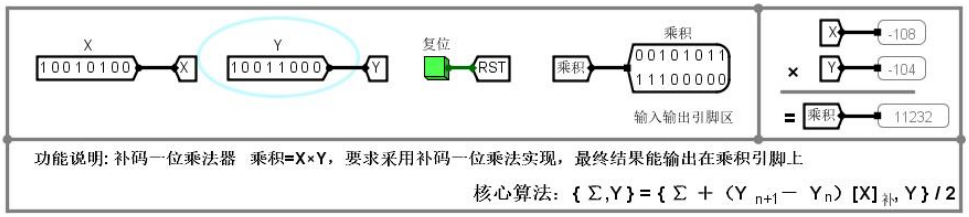
(1) 原码一位乘法器：

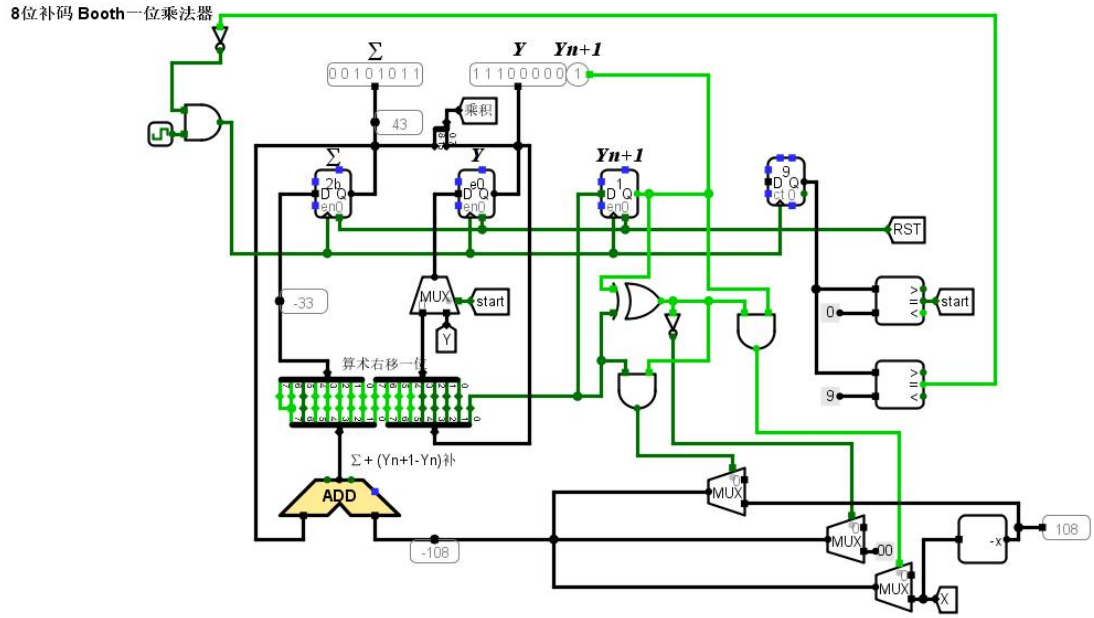




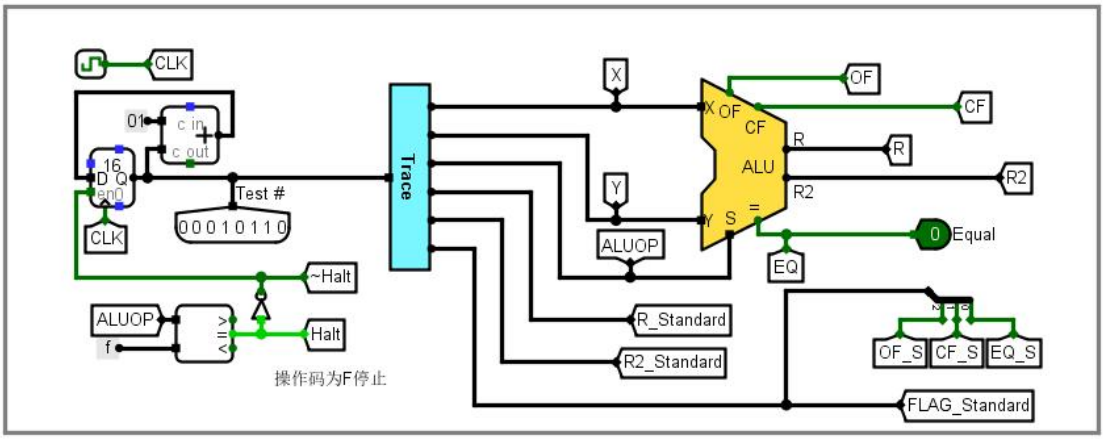
后续所有实验中凡是涉及存储器件，如寄存器、计数器、RAM等，必须增加统一的复位信号RST，方便系统复位

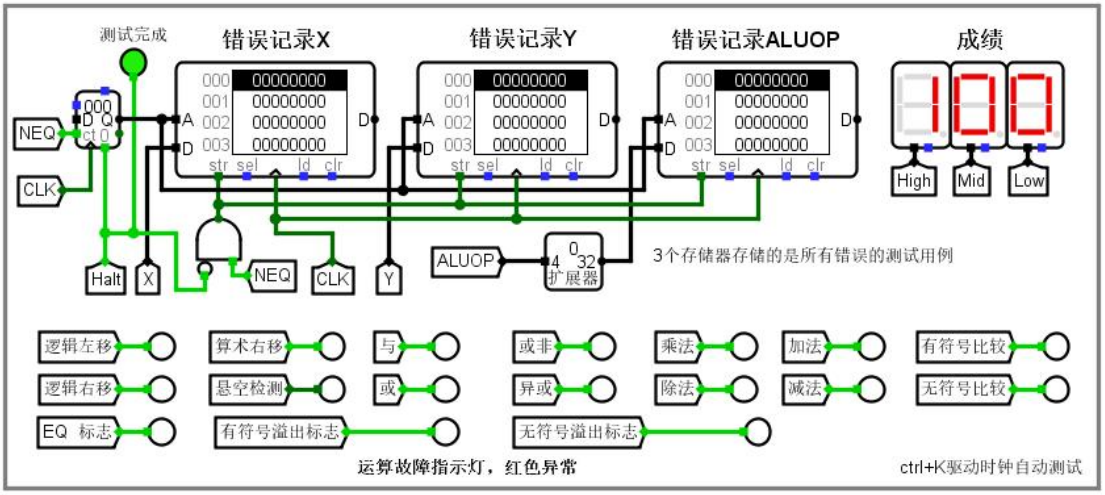
(2) 补码一位乘法器：


X
10010100
X
Y
10011000
Y
复位
RST
乘积
乘积
00101011
11100000
输入输出引脚区
X
-108
×
Y
-104
=
乘积
11232
功能说明: 补码一位乘法器 乘积=X×Y，要求采用补码一位乘法实现，最终结果能输出在乘积引脚上
核心算法：{ Σ,Y } = { Σ ＋（Y n+1 － Y n）[X] 补, Y } / 2


8位补码 Booth一位乘法器
Σ
00101011
43
Y
Yn+1
11100000
1
乘积
RST
MUX
start
-33
Y
算术右移一位
Σ + (Yn+1-Yn)补
ADD
-108
0
9
108
-x
X
00

4. ALU 实验





六、实验总结

这次实验所连电路数量较多，但较为简单，在连接过程中一共出现了三次错误：一次是在 32 位快速加法运算器中，低位的 16 位加法器的溢出位仅仅只传入了高位的 16 位加法器的先行进位中，未传入 4 位加法器的进位位中；第二个错误是补码一位乘法器中，累积和右移时将加法器的进位位补充至最高位，而不是将符号位补充至最高位；第三个错误是理解错误题意，在 ALU 电路中隧道“UOF”的意思是无符号的溢出位判断，而我理解成了无溢出的判断。综上，连接线路不仅是一门技术活，还是一门耐心活。必须要有足够的耐心与一定的头脑，才能成功连接线路。期待自己下一次能有更好的表现。