



**《计算机组成原理》**

**课程实验报告**

姓名：陈澄

学院：信息学院

系：软件工程

专业：软件工程

学号：32420212202930

2023年3月23日

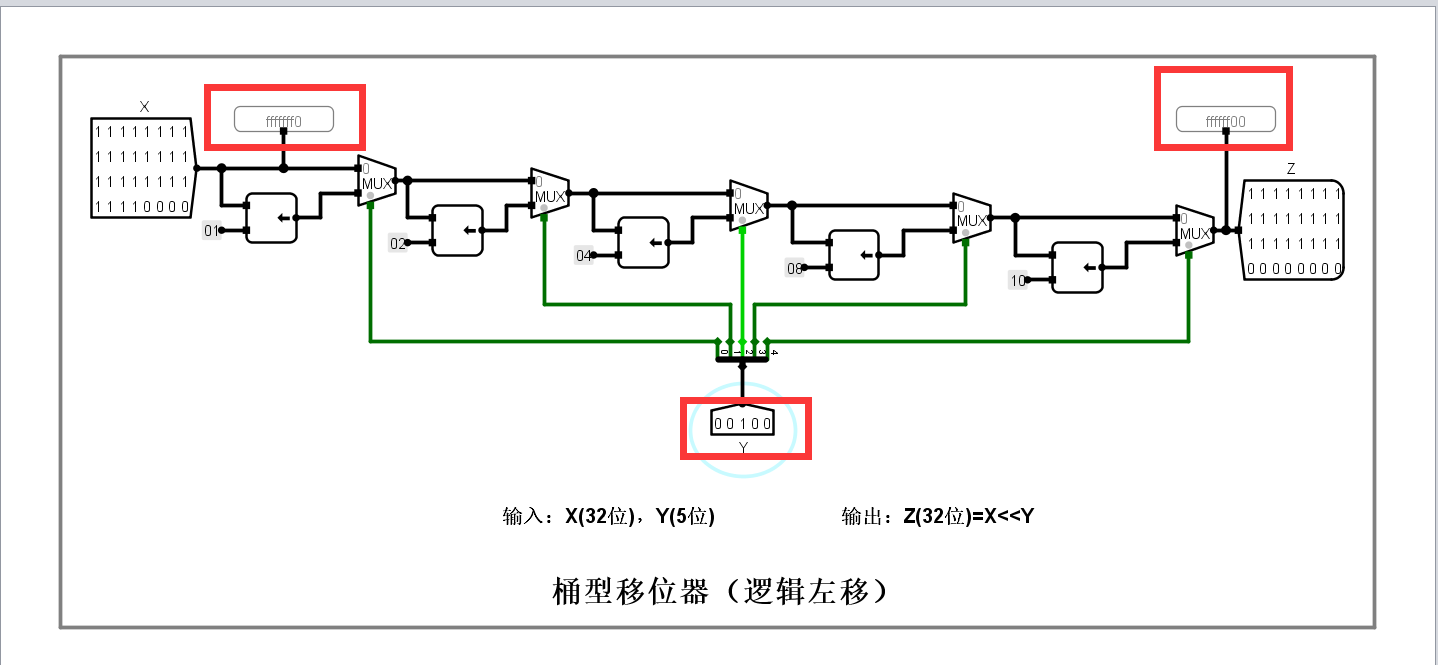
**第2次实验 运算方法与运算器**

1. **实验环境**
2. **Windows系统下运行Dev-C++（或其他C语言开发环境）。**
3. **Windows系统下运行Logisim软件（需安装JDK）。**
4. **实验内容**
   1. **课堂完成部分（验证实验的内容）**
5. **请按照实验课件规定的内容，逐个完成实验。**
6. **采用屏幕拷贝的方式记录实验结果，黏贴到实验报告中，并给出相关的文字说明。**

一、桶型移位器

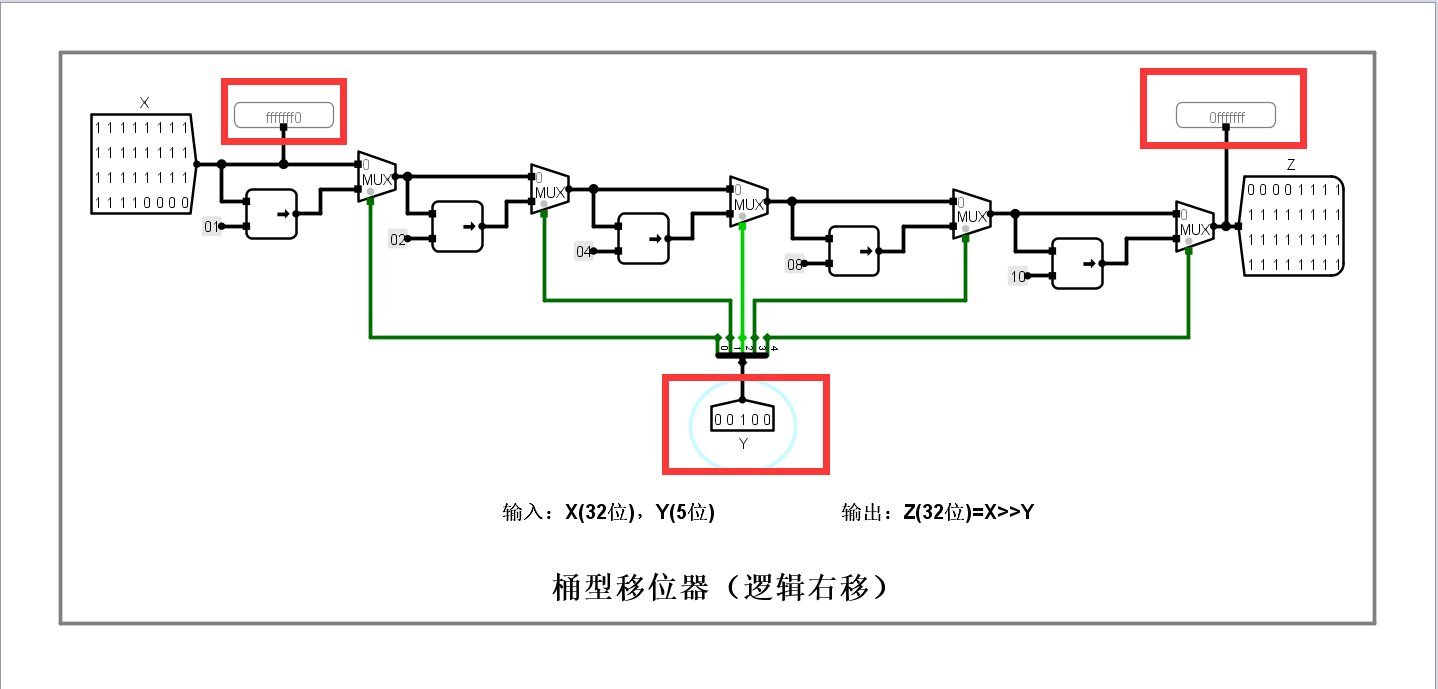
1、逻辑左移：Z=X<<Y

请验证：X=FFFFFFF0，Y=00100，Z=FFFFFF00



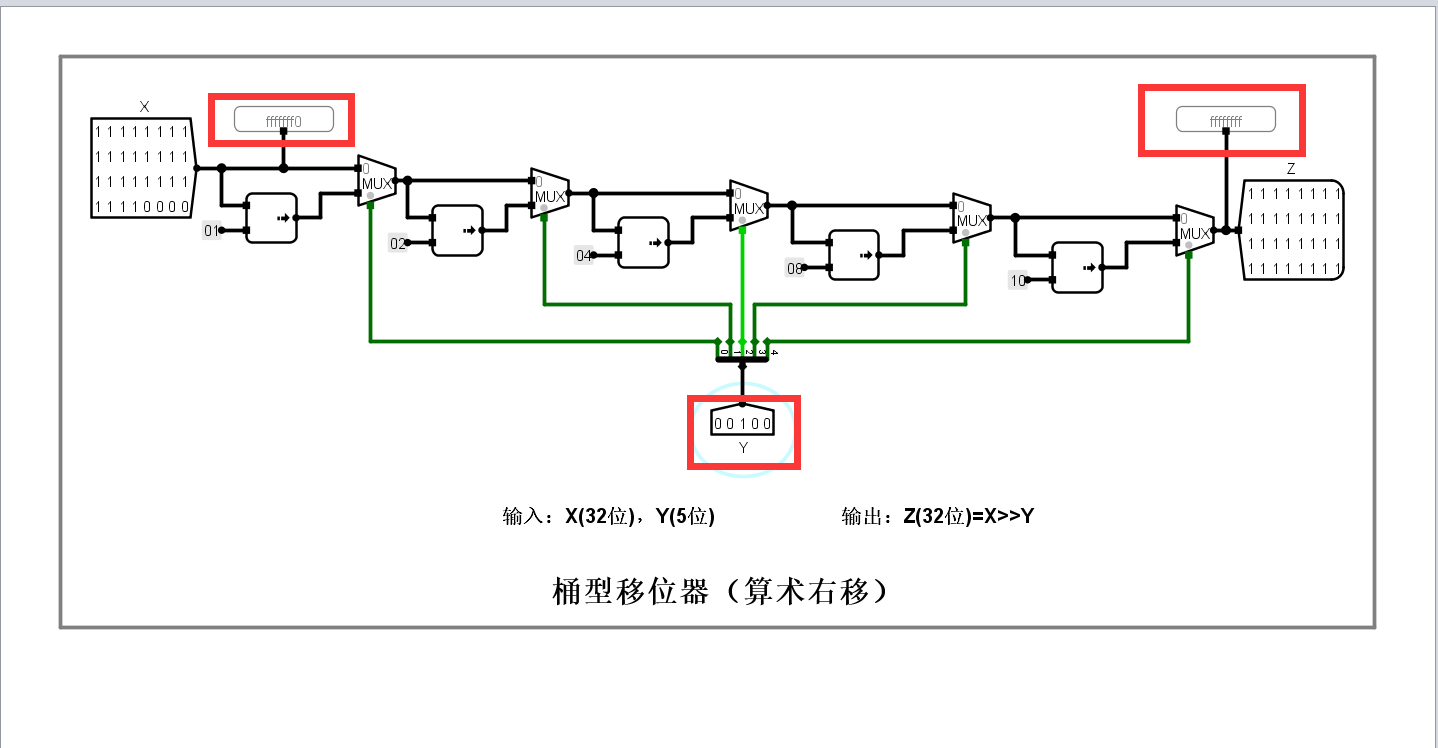
2、逻辑右移：Z=X>>Y

请验证：X=FFFFFFF0，Y=00100，Z=0FFFFFFF

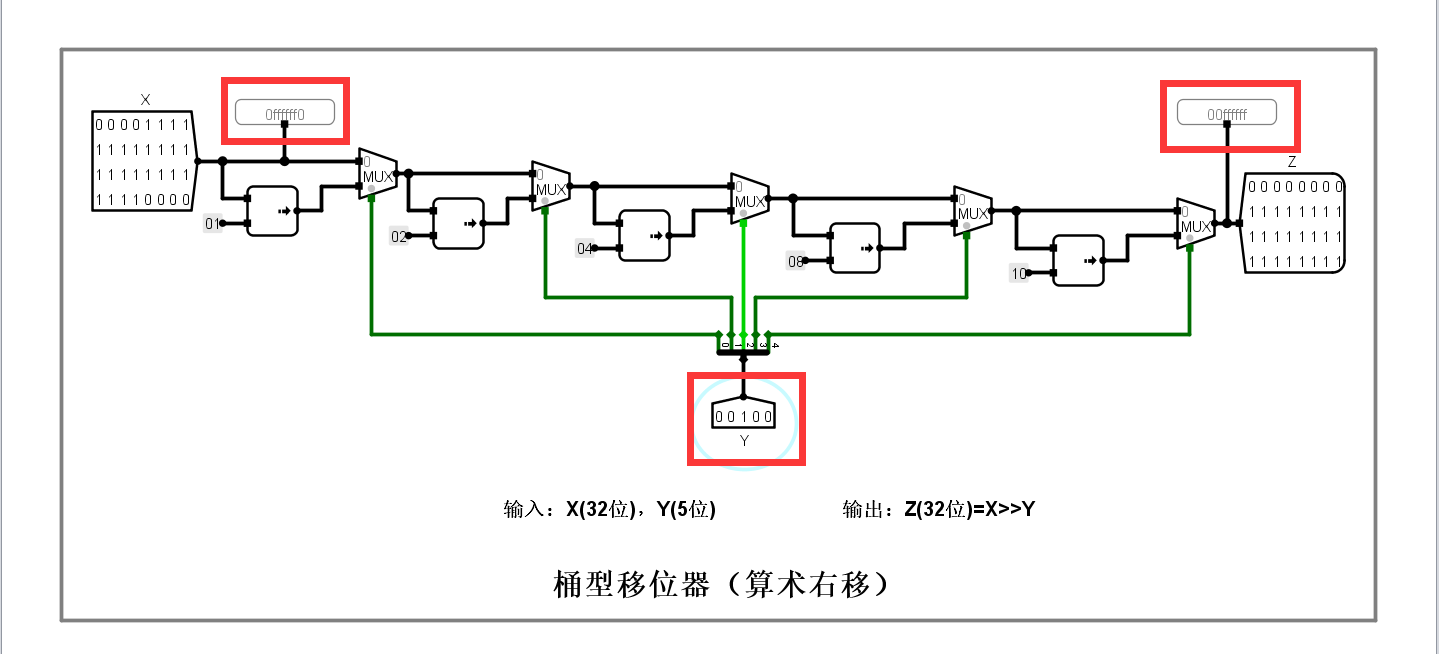


3、算术右移：Z=X>>Y

请验证：X=FFFFFFF0，Y=00100，Z=FFFFFFFF



请验证：X=0FFFFFF0，Y=00100，Z=00FFFFFF



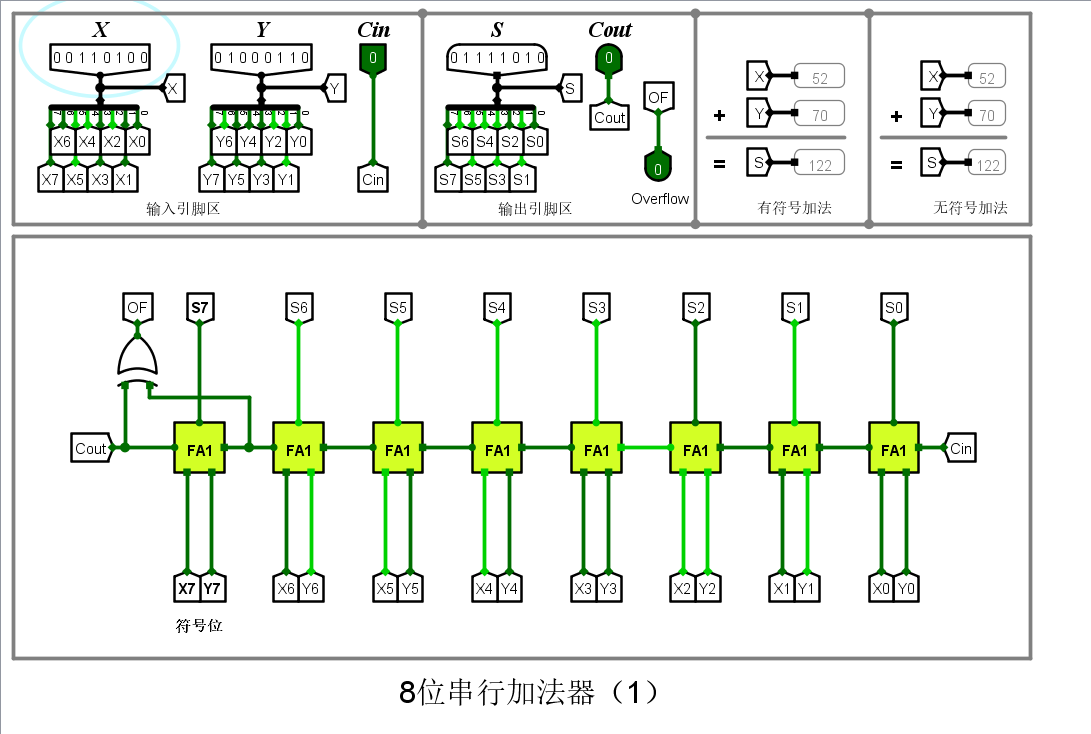
二、串行加法器

1、8位串行加法器

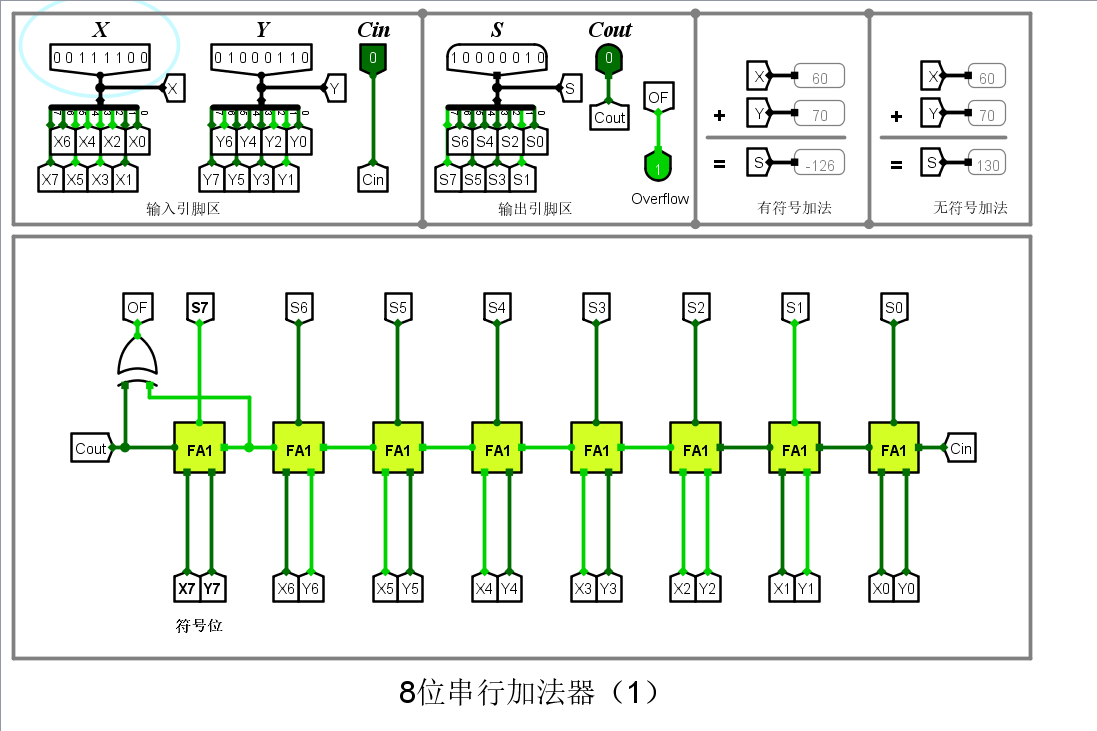
请同学们通过设置不同的X、Y、Cin值，对上述8位串行加法器、8位串行可控加减法器

（Sub=0）进行验证，并对结果进行分析（以下4种情况）：

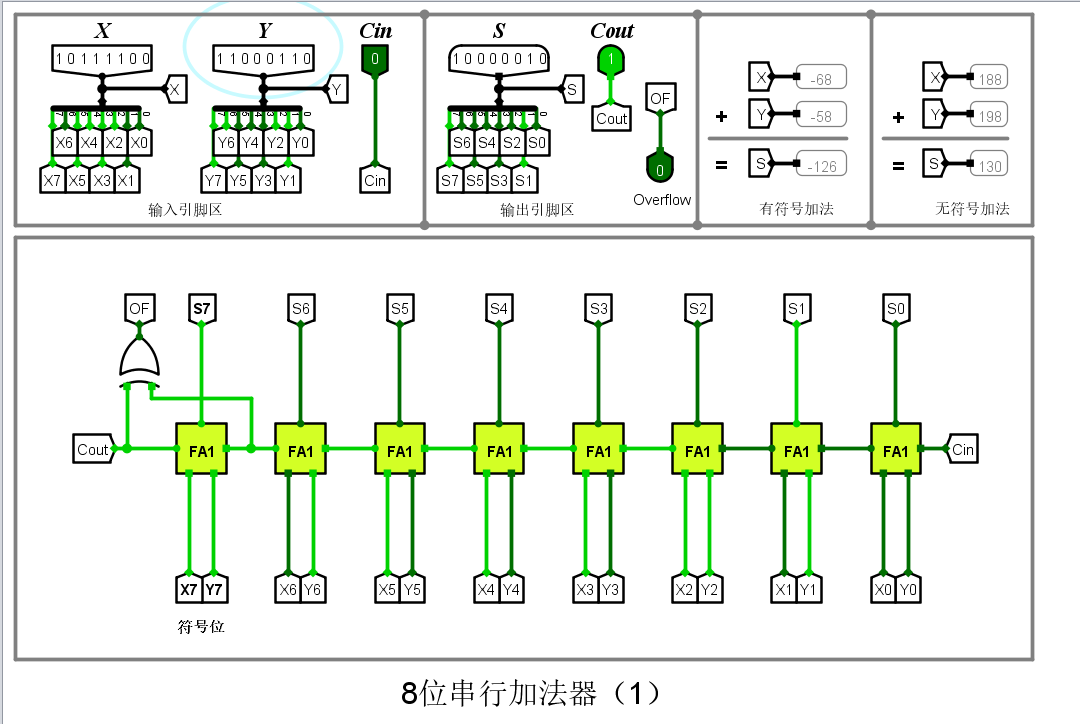
正数+正数=正数（不溢出）



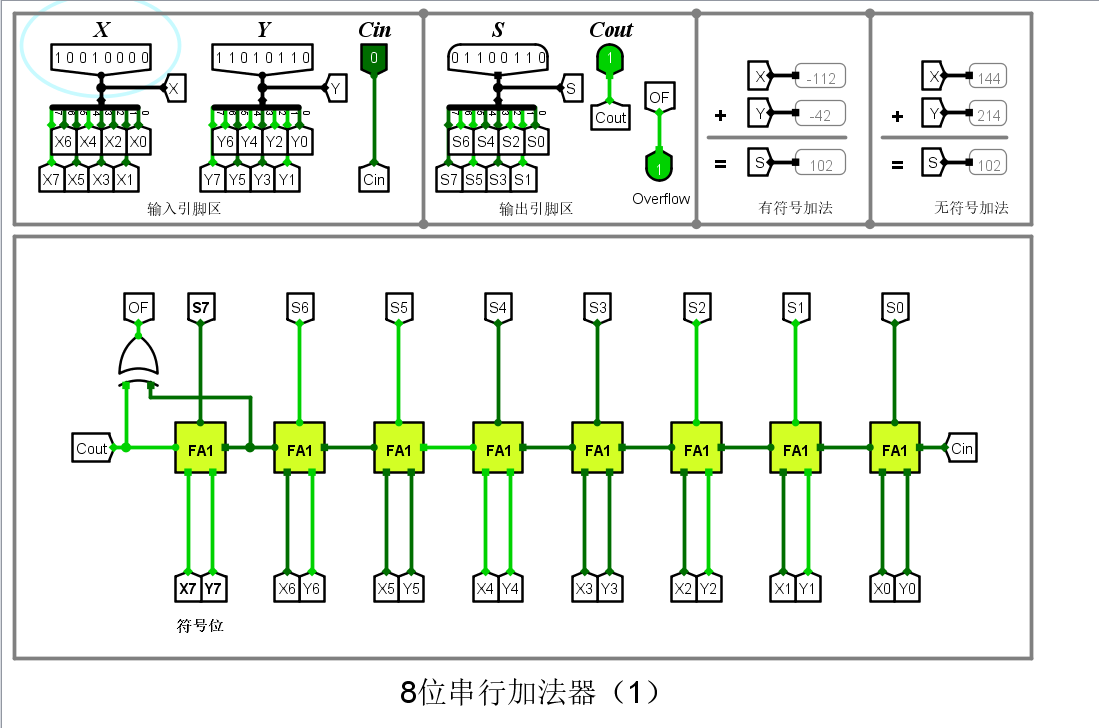
正数+正数=负数（溢出）



负数+负数=负数（不溢出）



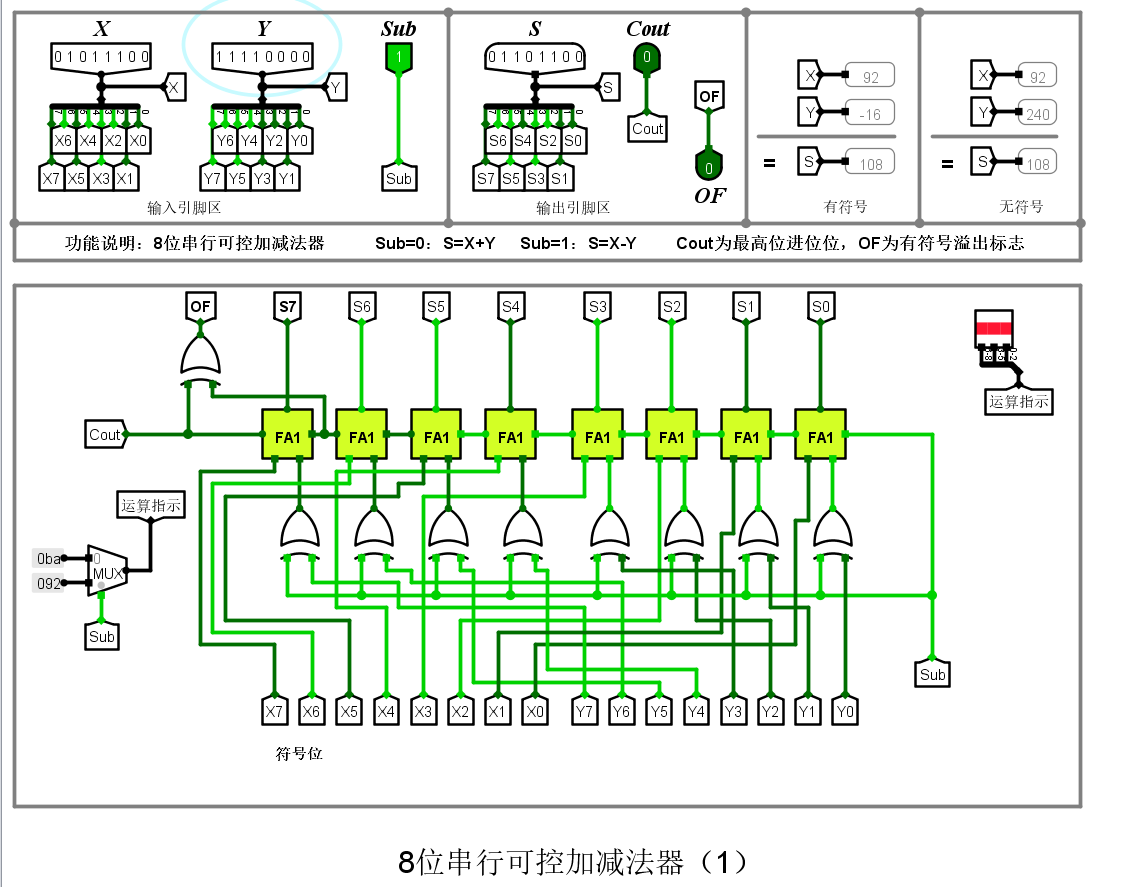
负数+负数=正数（溢出）



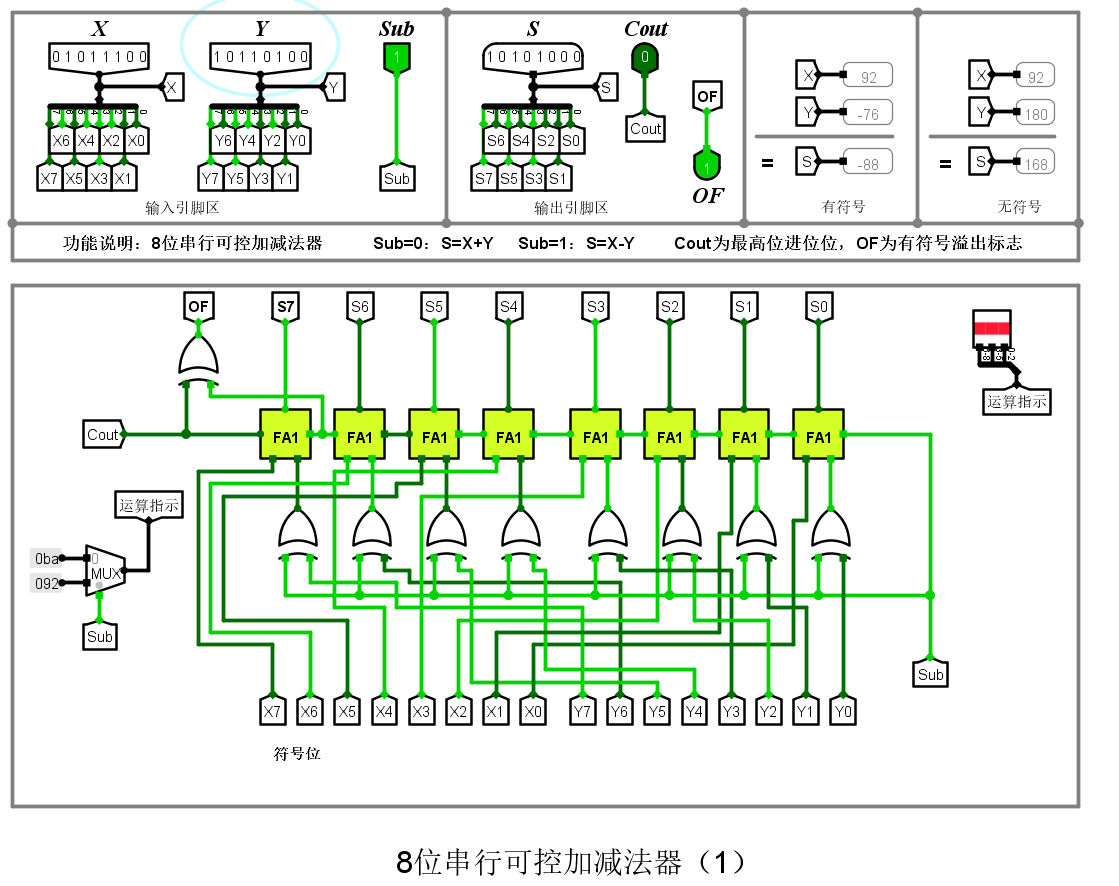
请同学们通过设置不同的X、Y值，对上述8位串行可控加减法器（Sub=1）进行验证，并对

结果进行分析（以下4种情况）：

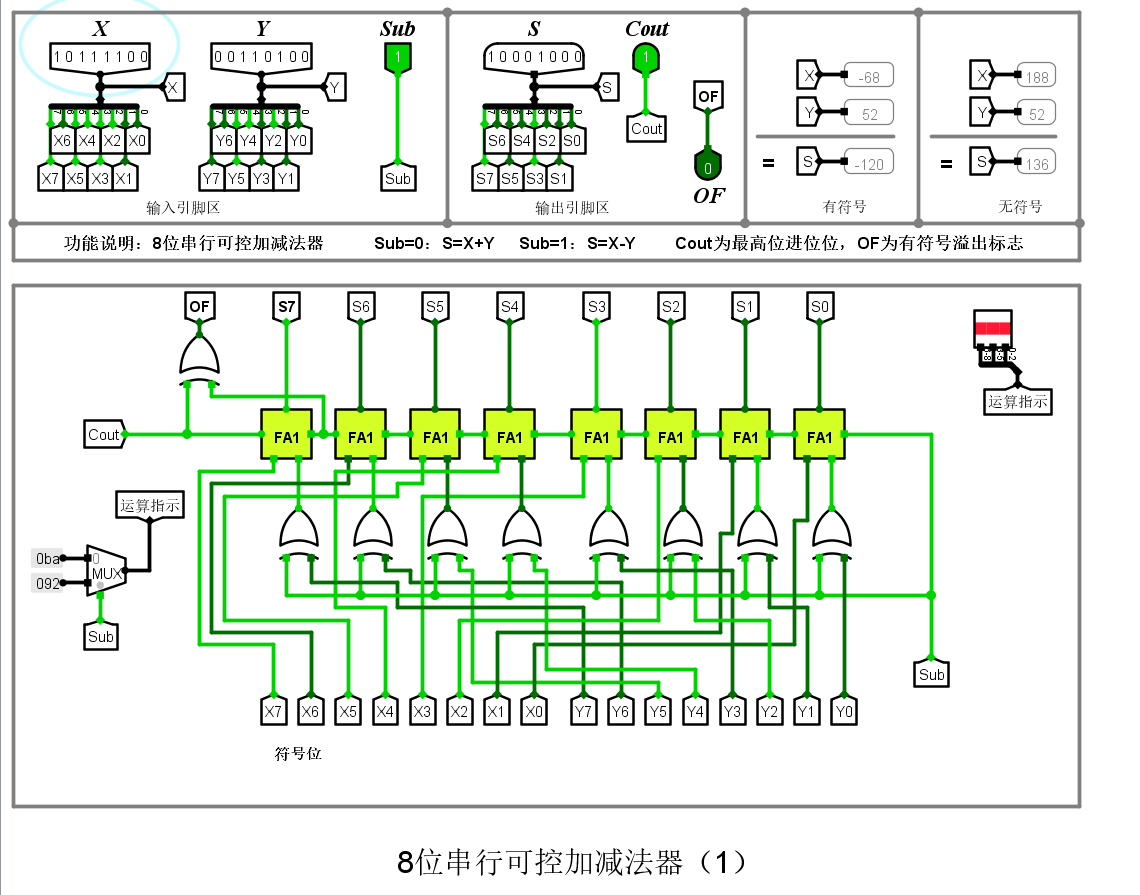
正数-负数=正数（不溢出）



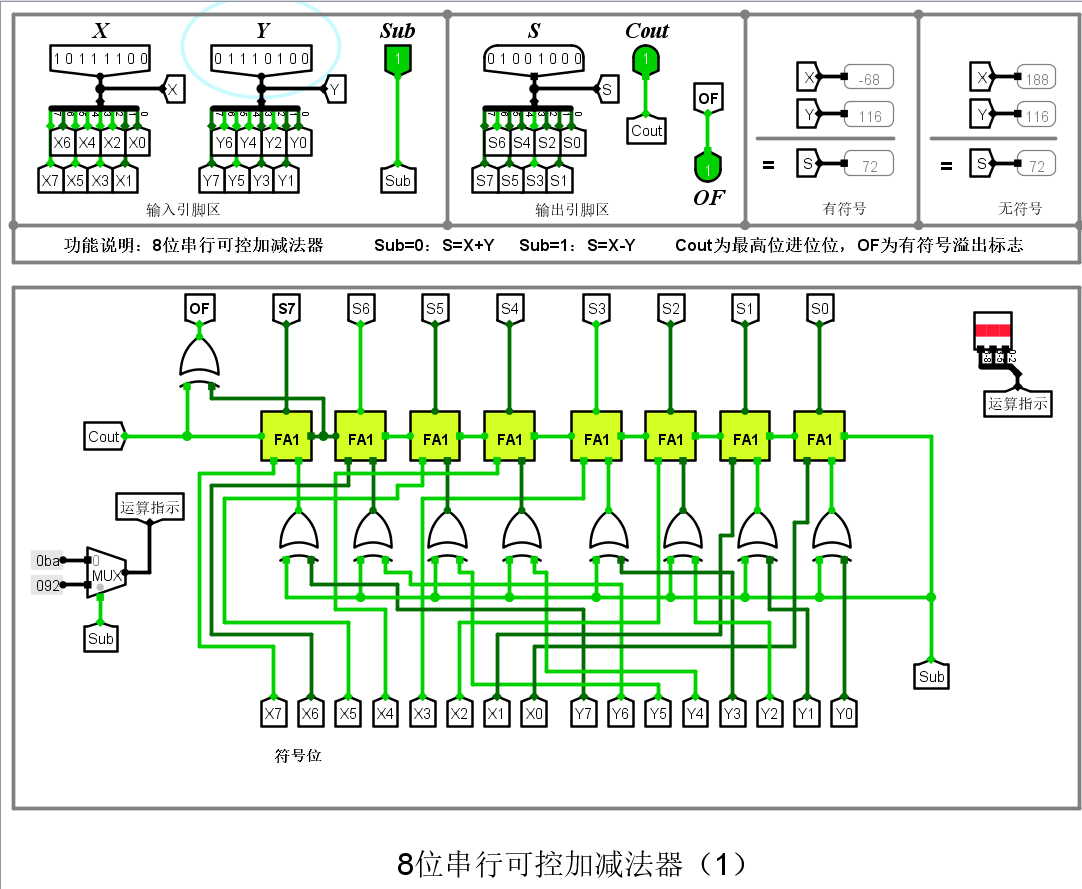
正数-负数=负数（溢出）



负数-正数=负数（不溢出）



负数-正数=正数（溢出）

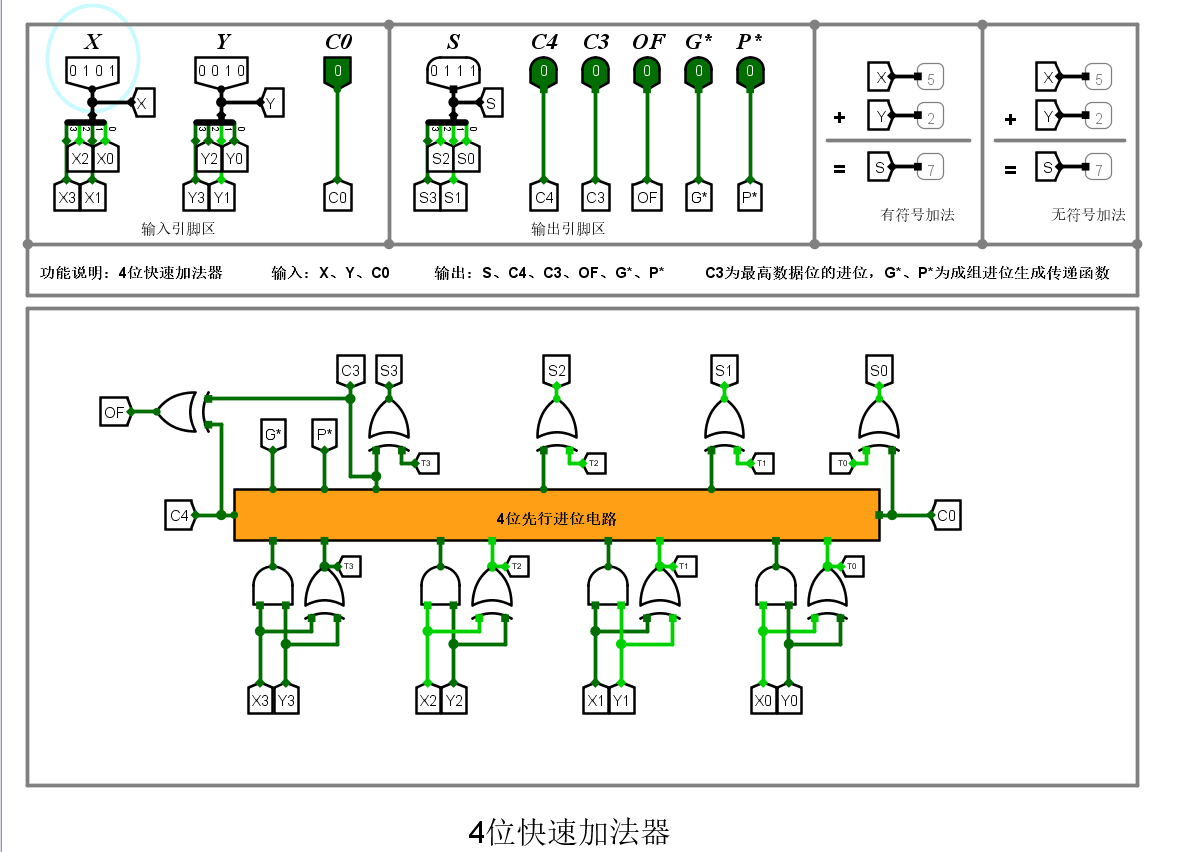


1. 先行进位加法器

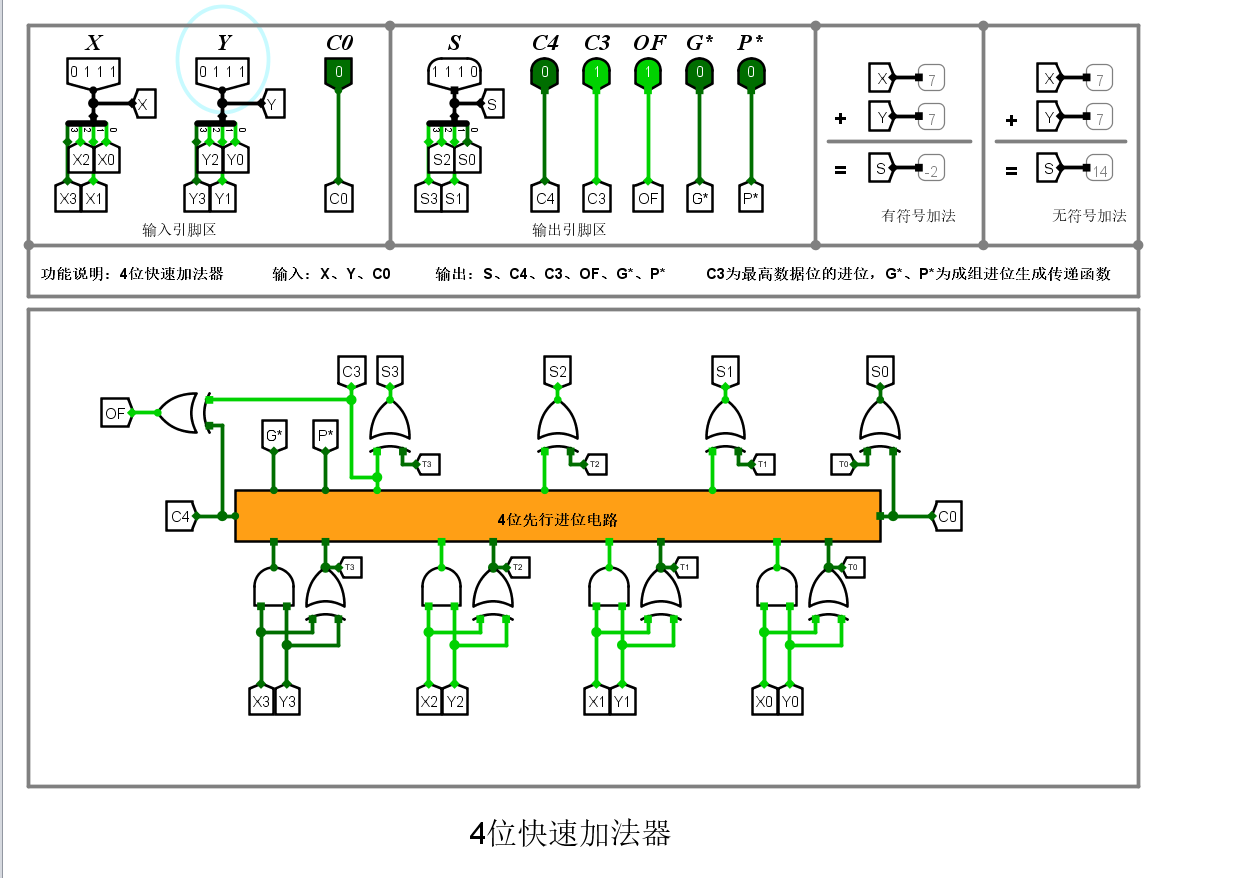
请同学们通过设置不同的X、Y、C0值，对上述4位快速加法器、16位快速加法器（两种方式）、32位快速加法器（两种电路）进行验证，并对结果进行分析（以下4种情况）：

4位快速加法器

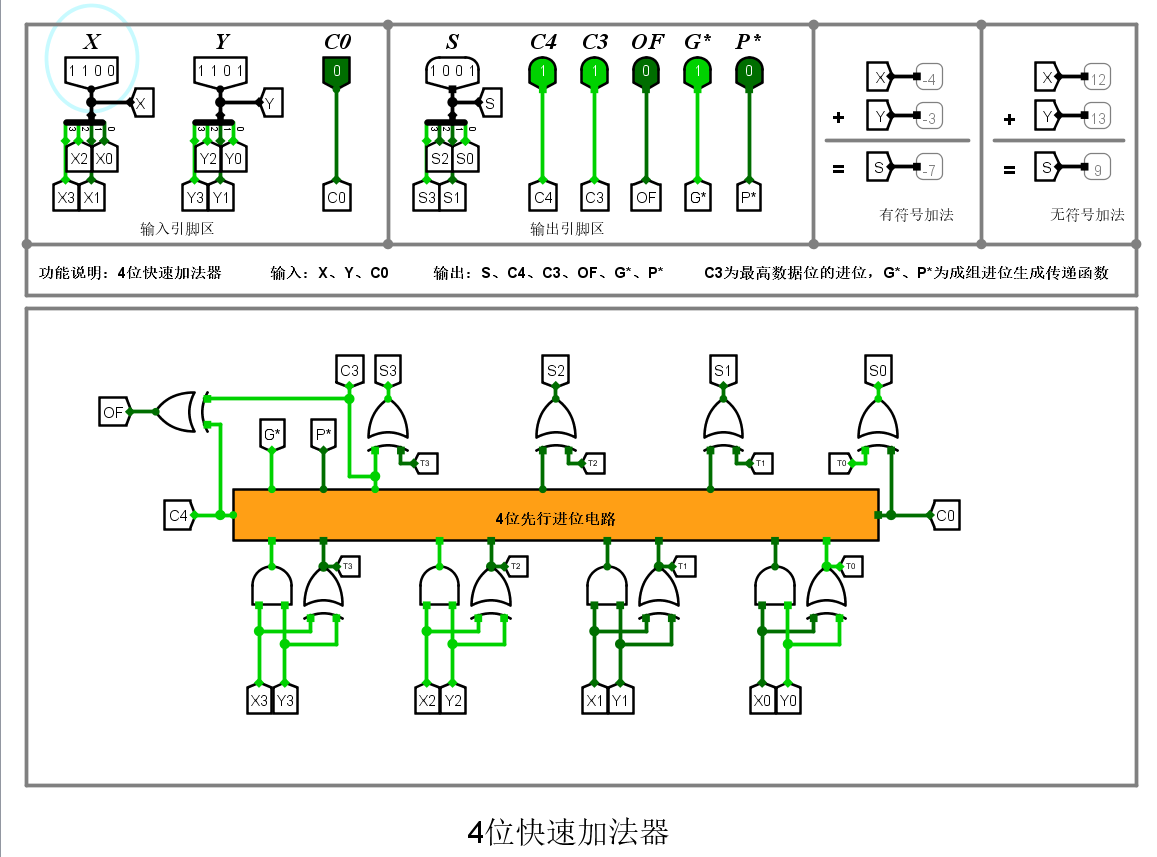
正数+正数=正数（不溢出）



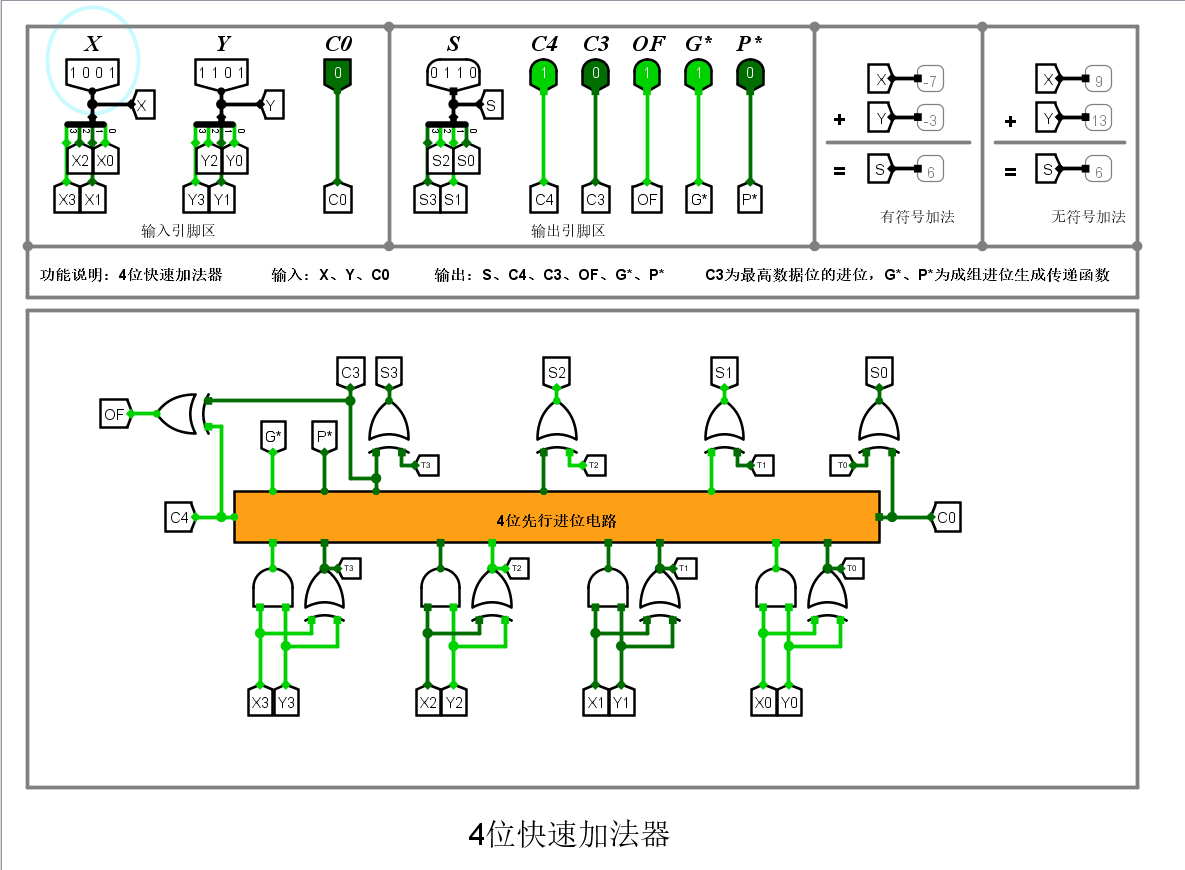
正数+正数=负数（溢出）



负数+负数=负数（不溢出）



负数+负数=正数（溢出）

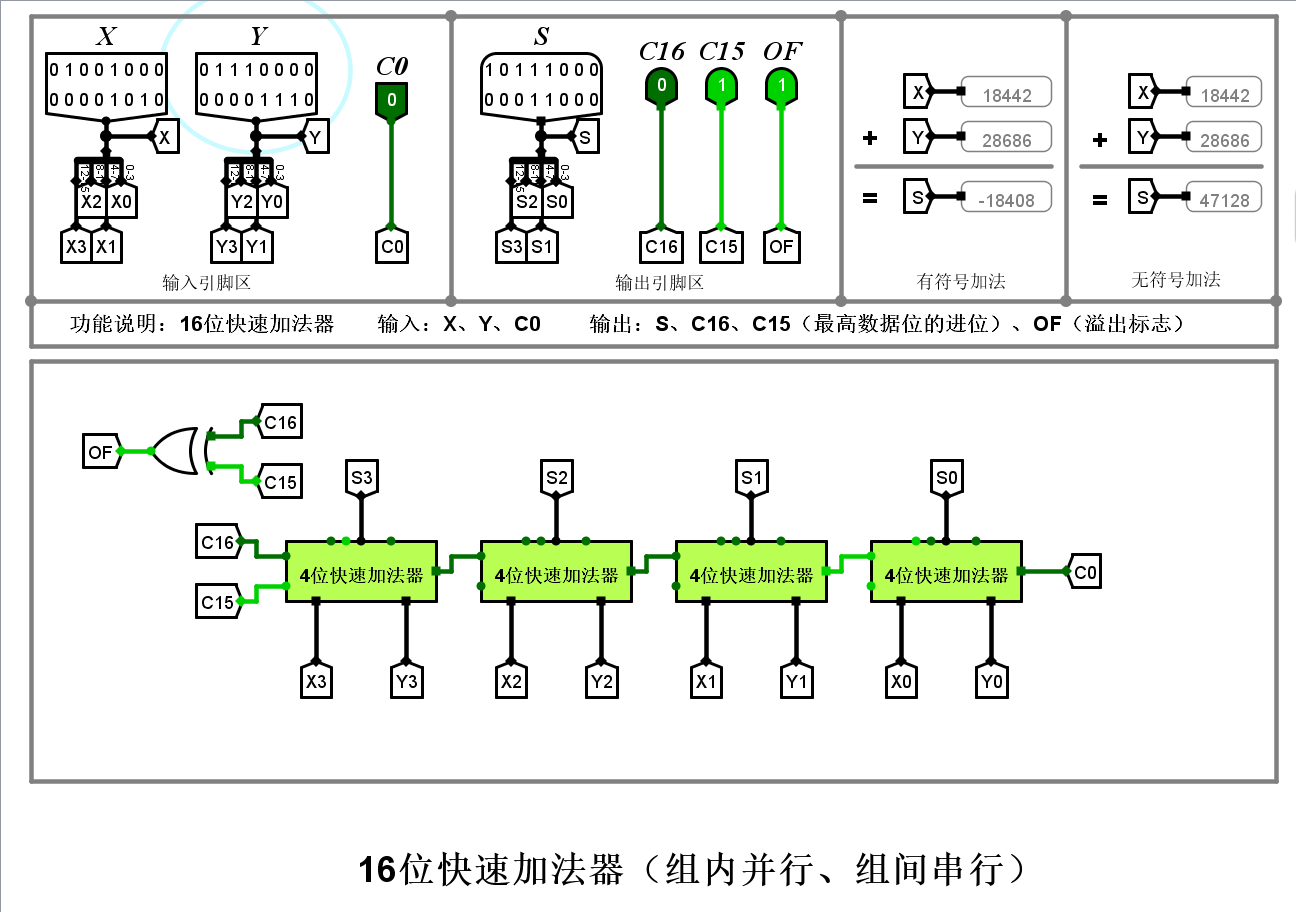


16位快速加法器（组内并行、组间串行）

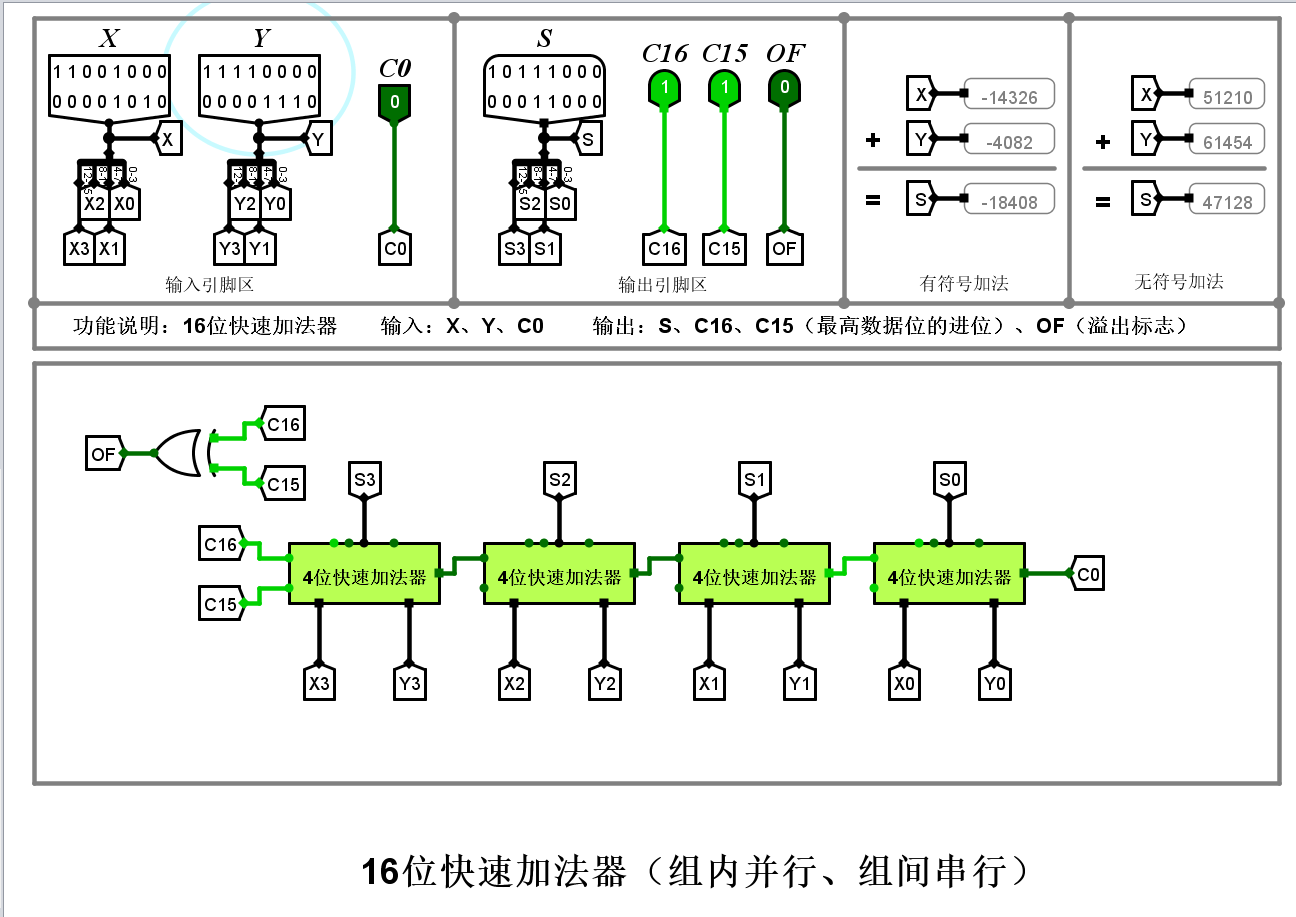
正数+正数=正数（不溢出）



正数+正数=负数（溢出）



负数+负数=负数（不溢出）

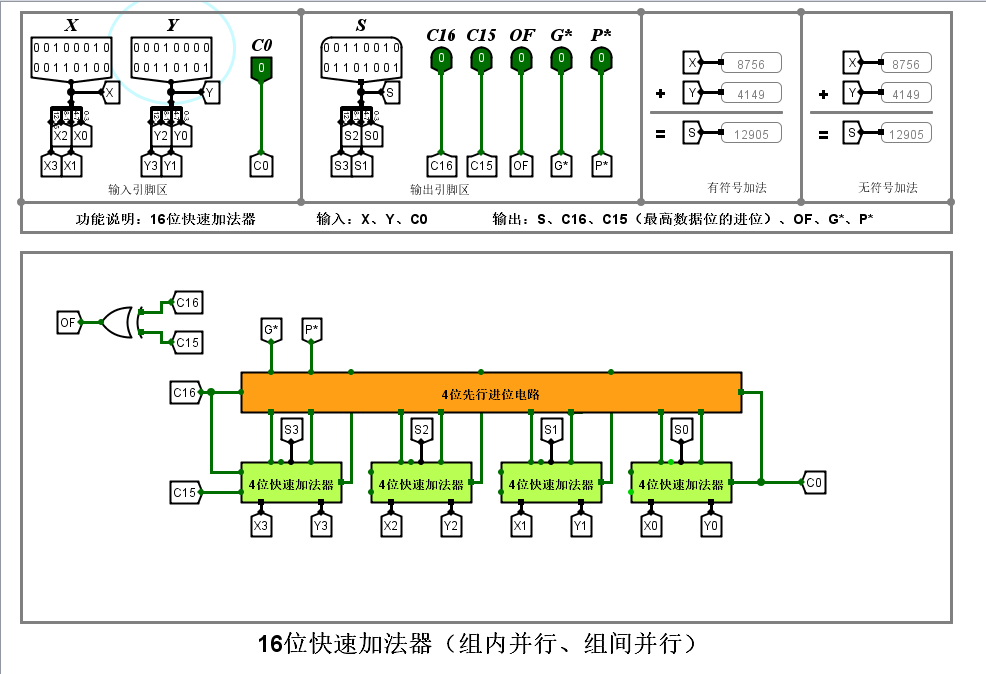


负数+负数=正数（溢出）

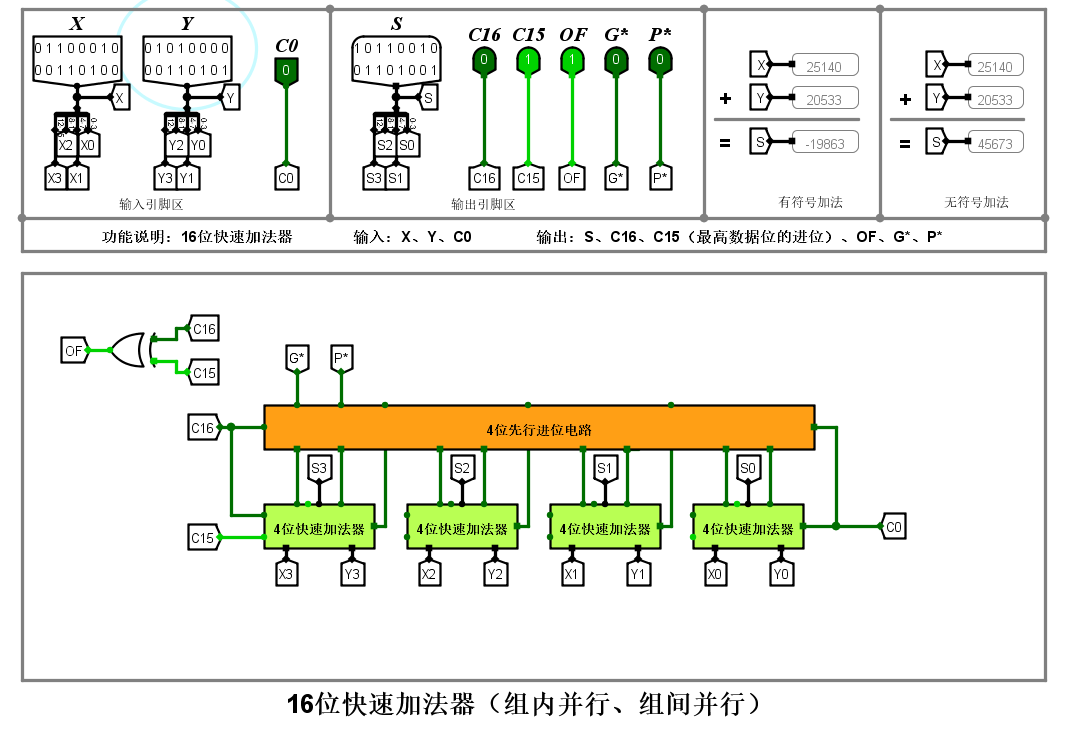


16位快速加法器（组内并行、组间并行）

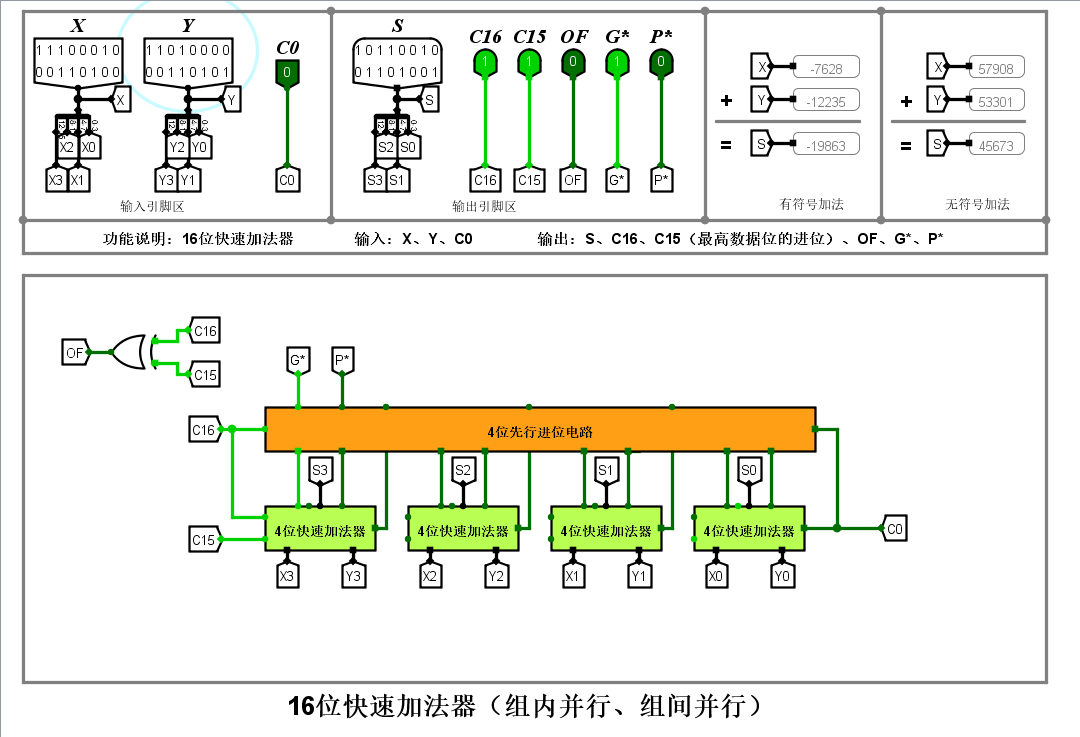
正数+正数=正数（不溢出）



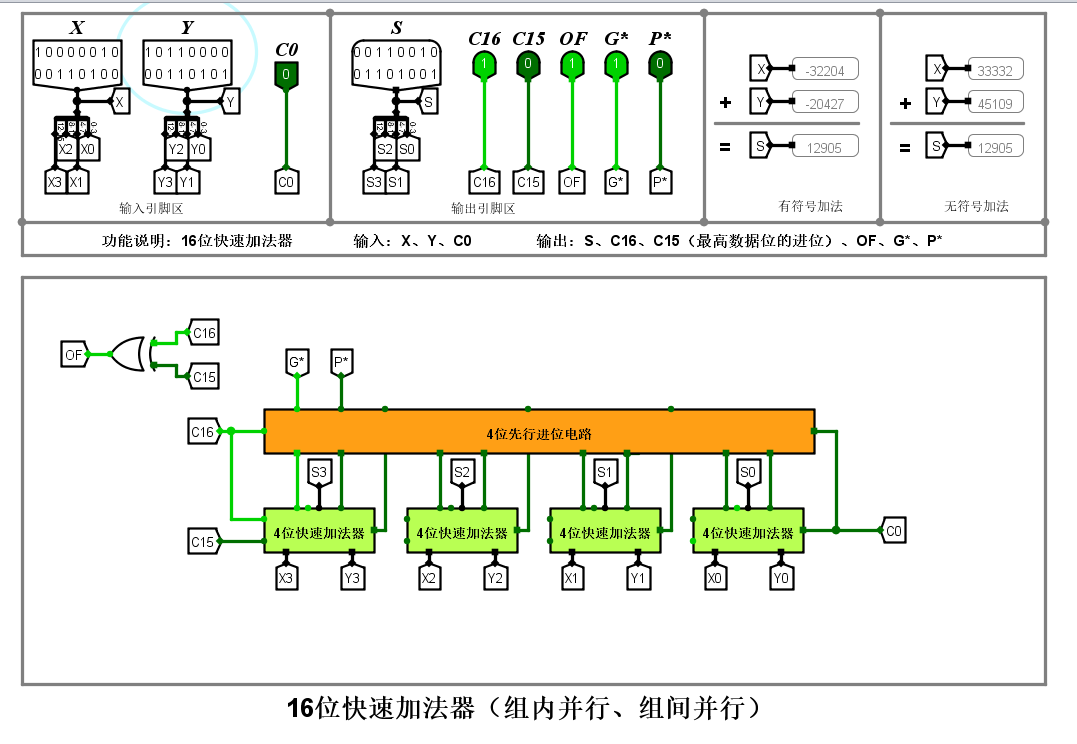
正数+正数=负数（溢出）



负数+负数=负数（不溢出）

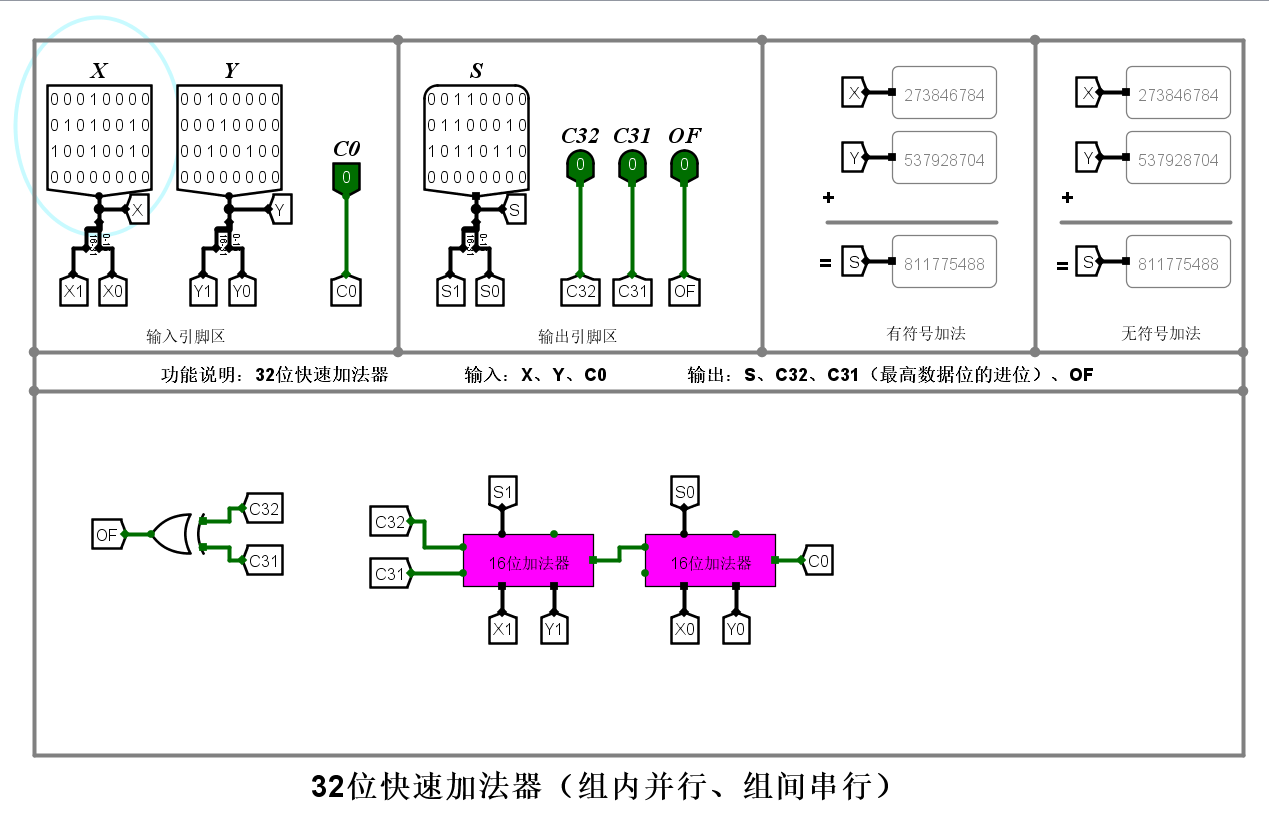


负数+负数=正数（溢出）

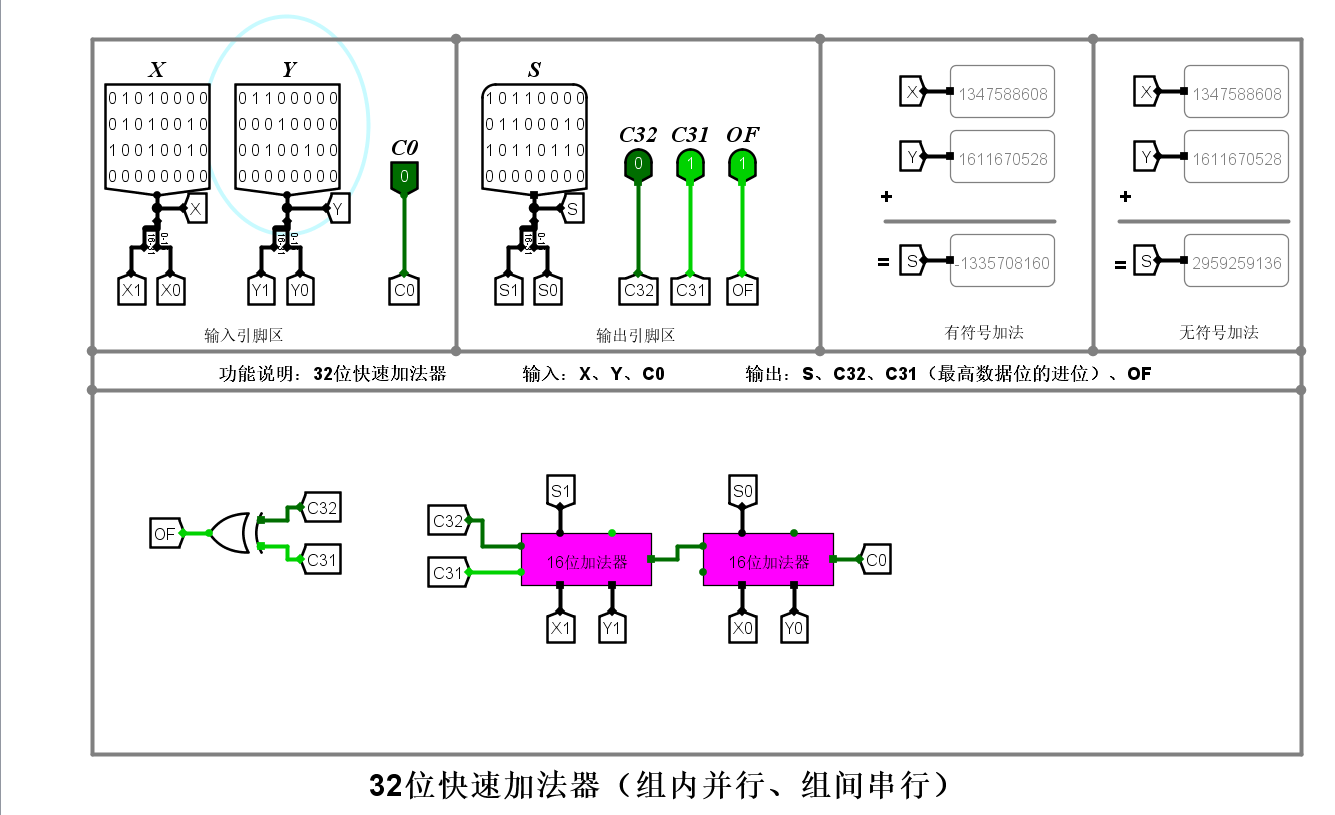


32位快速加法器（组内并行，组间串行）（1）

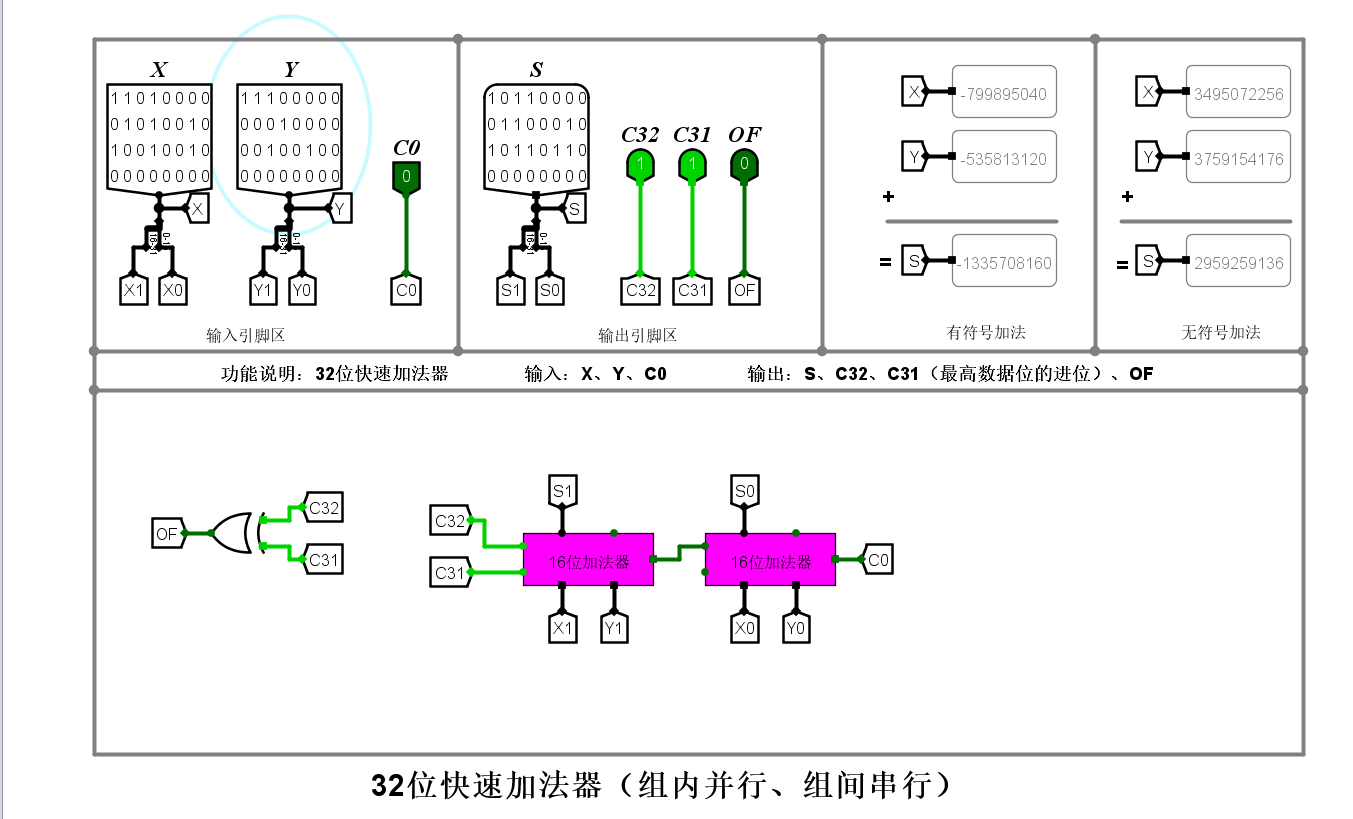
正数+正数=正数（不溢出）



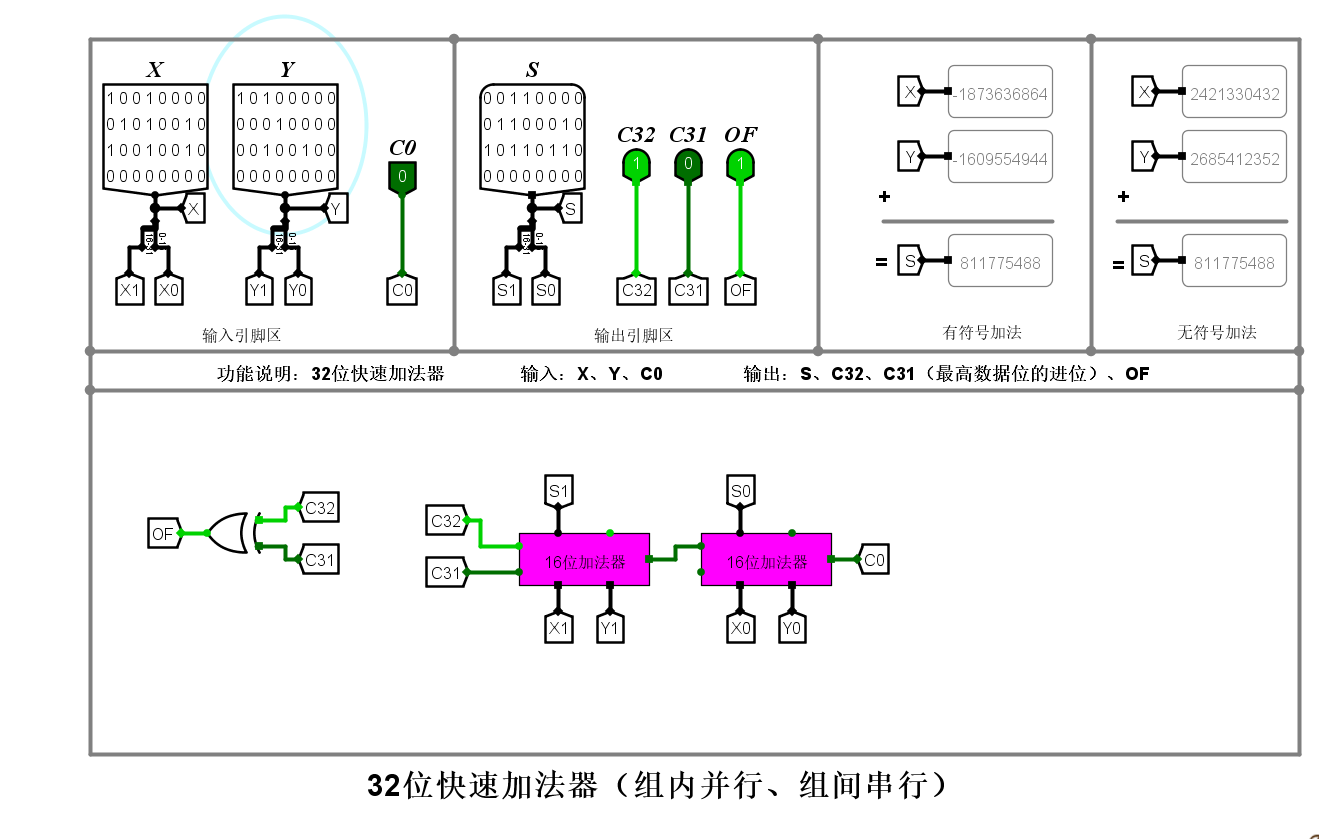
正数+正数=负数（溢出）



负数+负数=负数（不溢出）

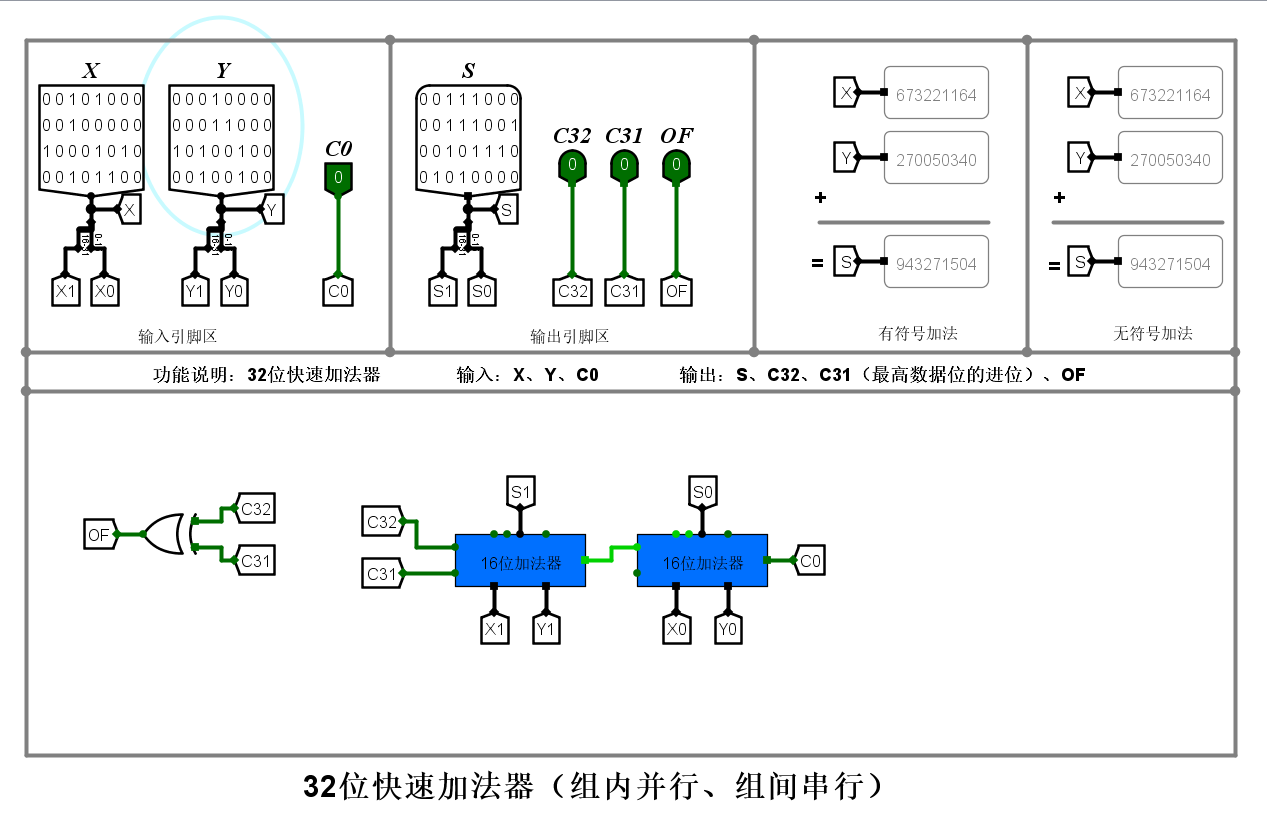


负数+负数=正数（溢出）

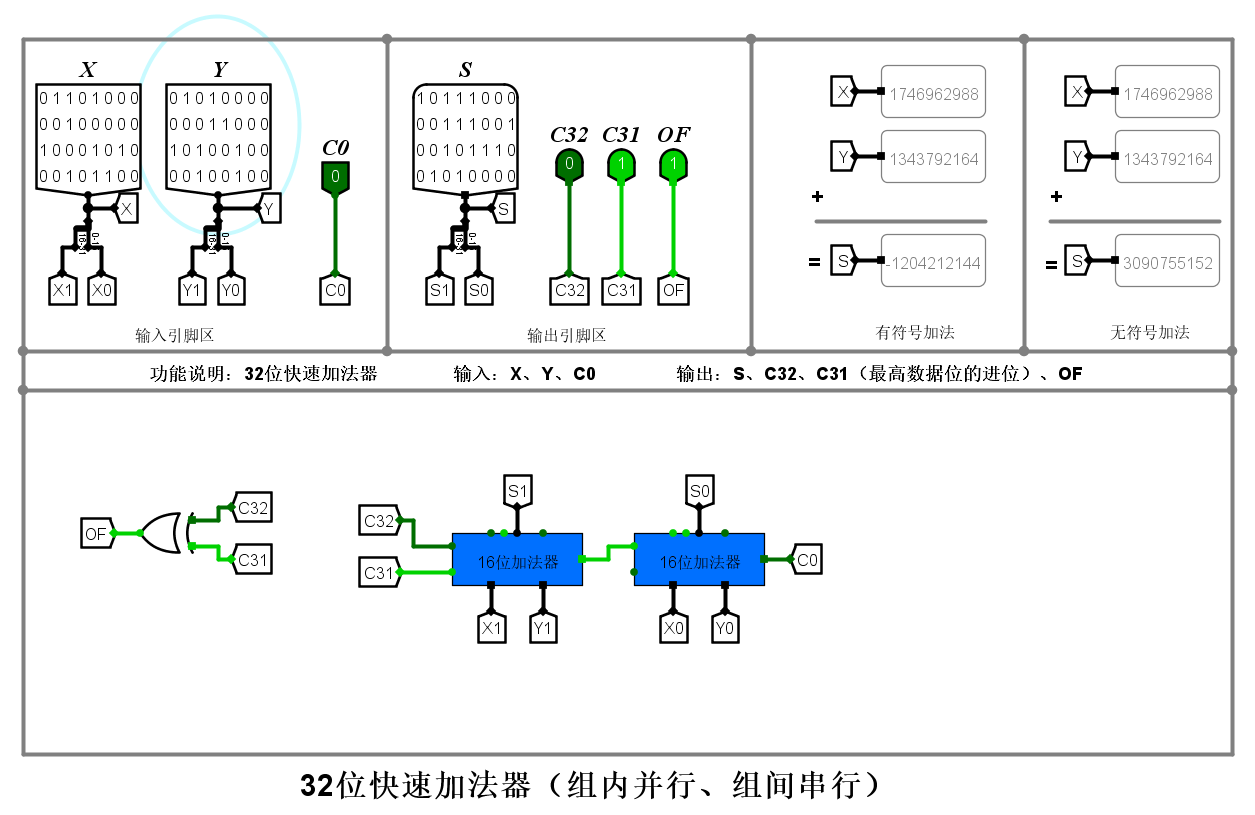


32位快速加法器（组内并行，组间串行）（2）

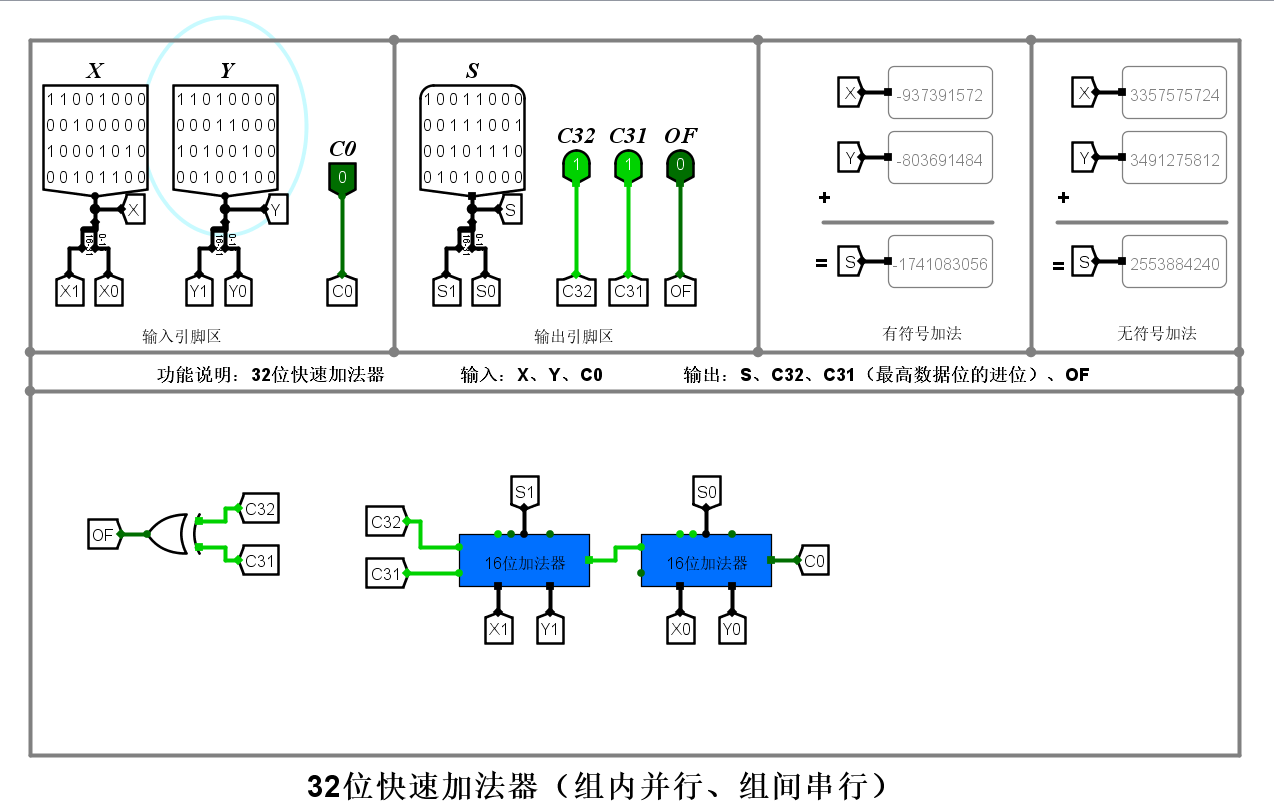
正数+正数=正数（不溢出）



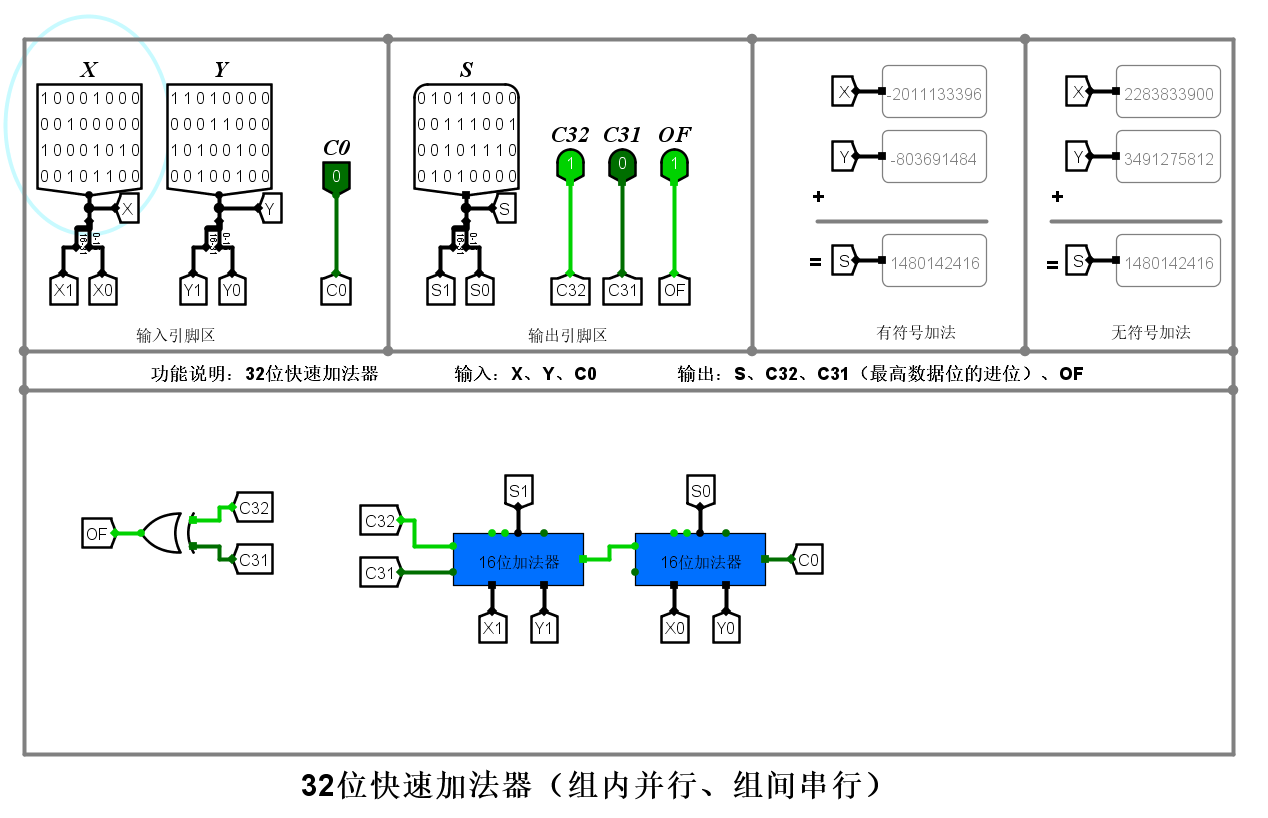
正数+正数=负数（溢出）



负数+负数=负数（不溢出）



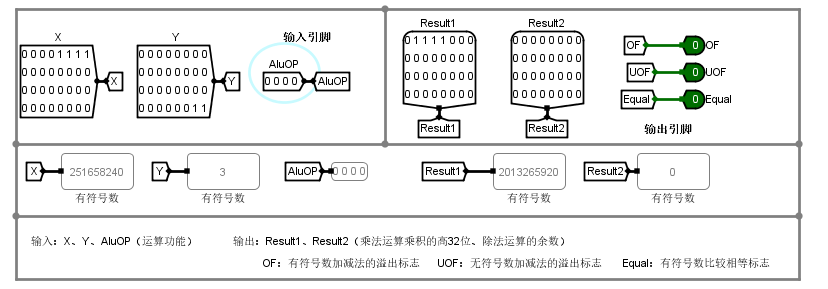
负数+负数=正数（溢出）



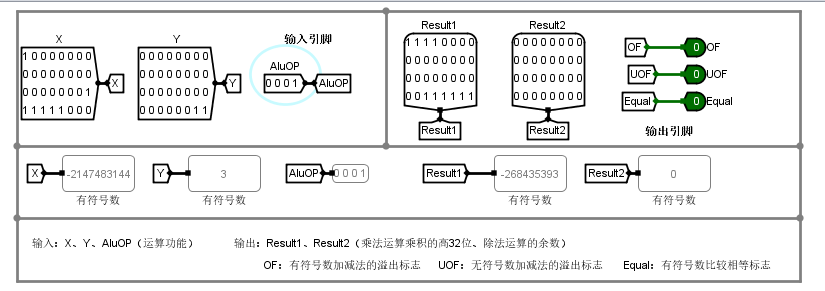
四、算术逻辑单元（ALU）

请同学们在上述32位ALU中，通过改变不同的X、Y、AluOP值，完成下列运算功能：

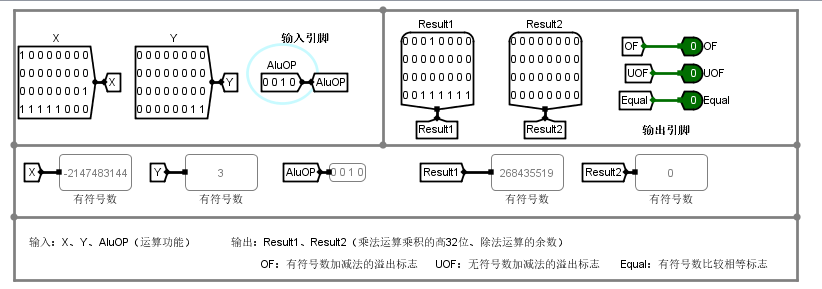
① 逻辑左移



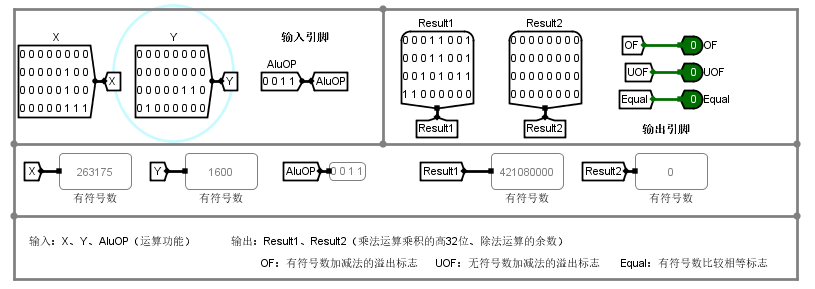
② 算术右移



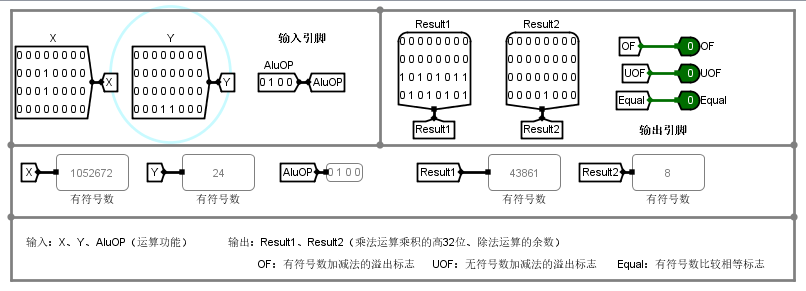
③ 逻辑右移



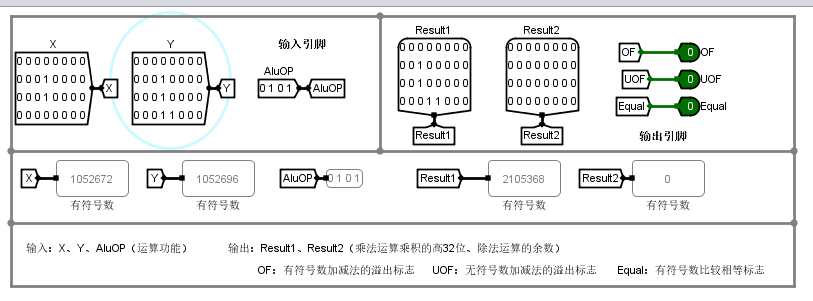
④ 乘法运算



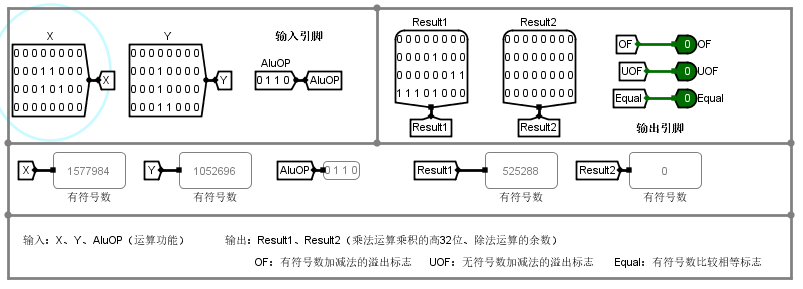
⑤ 除法运算



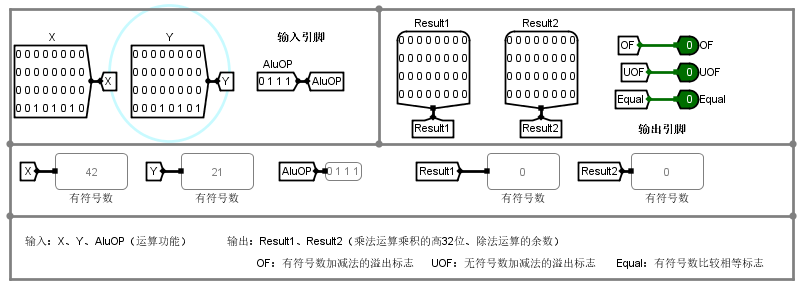
⑥ 加法运算



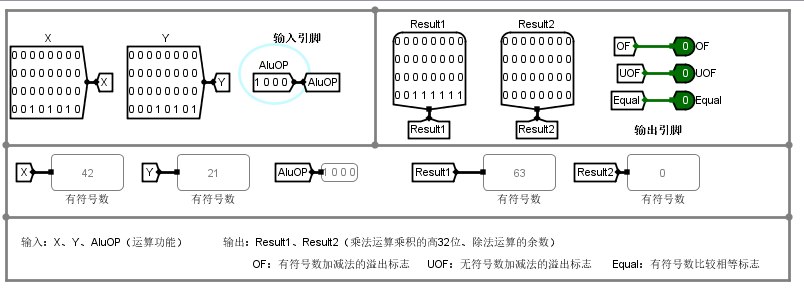
⑦ 减法运算



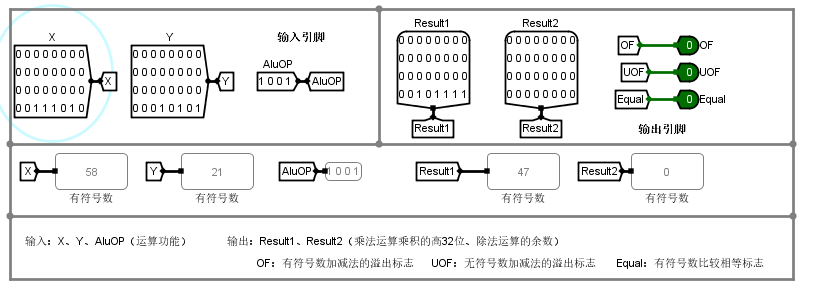
⑧ 逻辑与运算



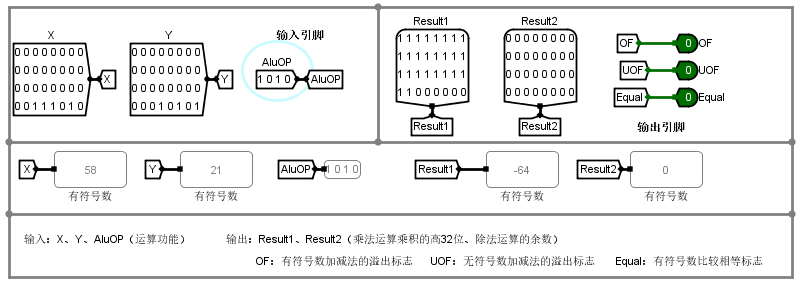
⑨ 逻辑或运算



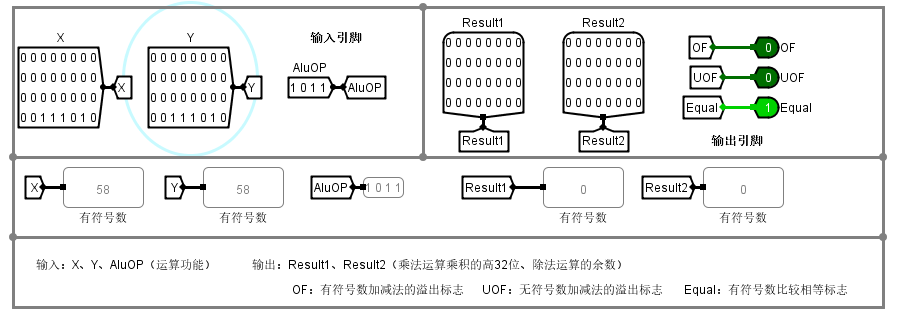
⑩ 逻辑异或运算



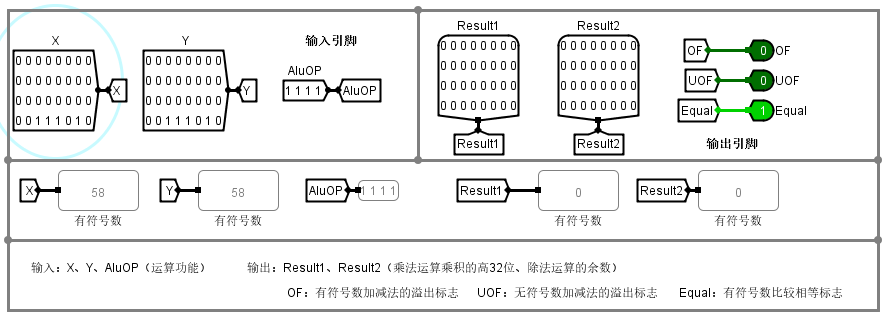
⑪ 逻辑或非运算



⑫ 有符号数比较



⑬ 无符号数比较

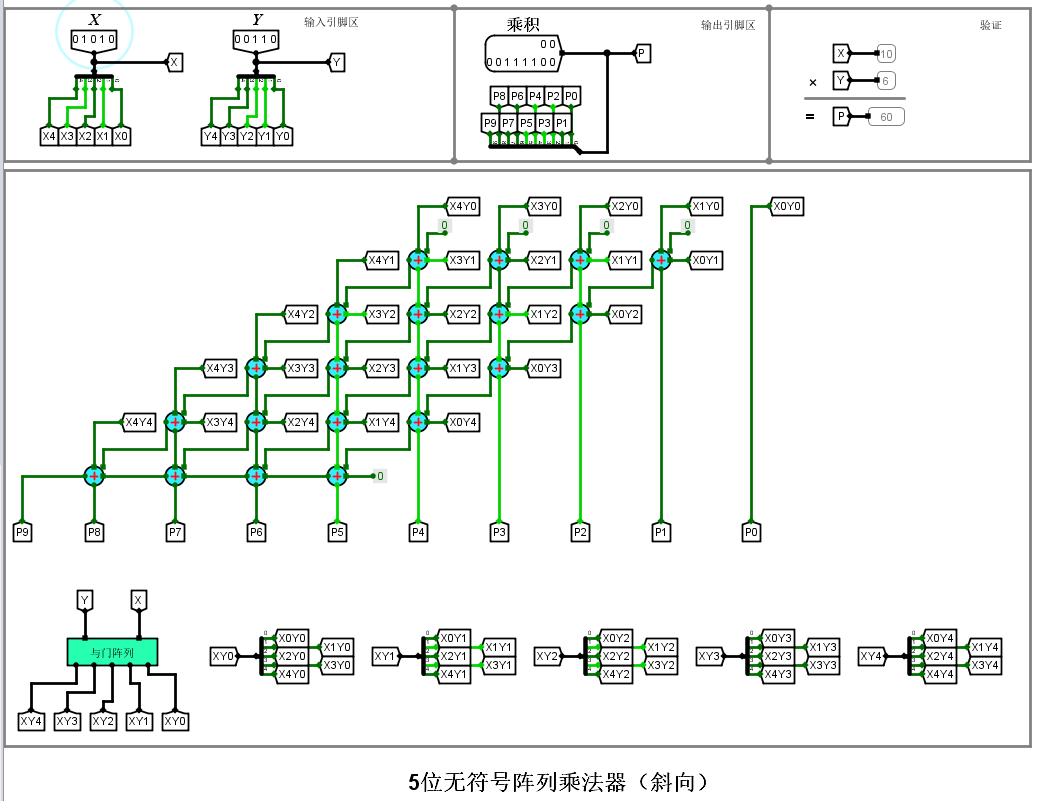


五、阵列乘法器

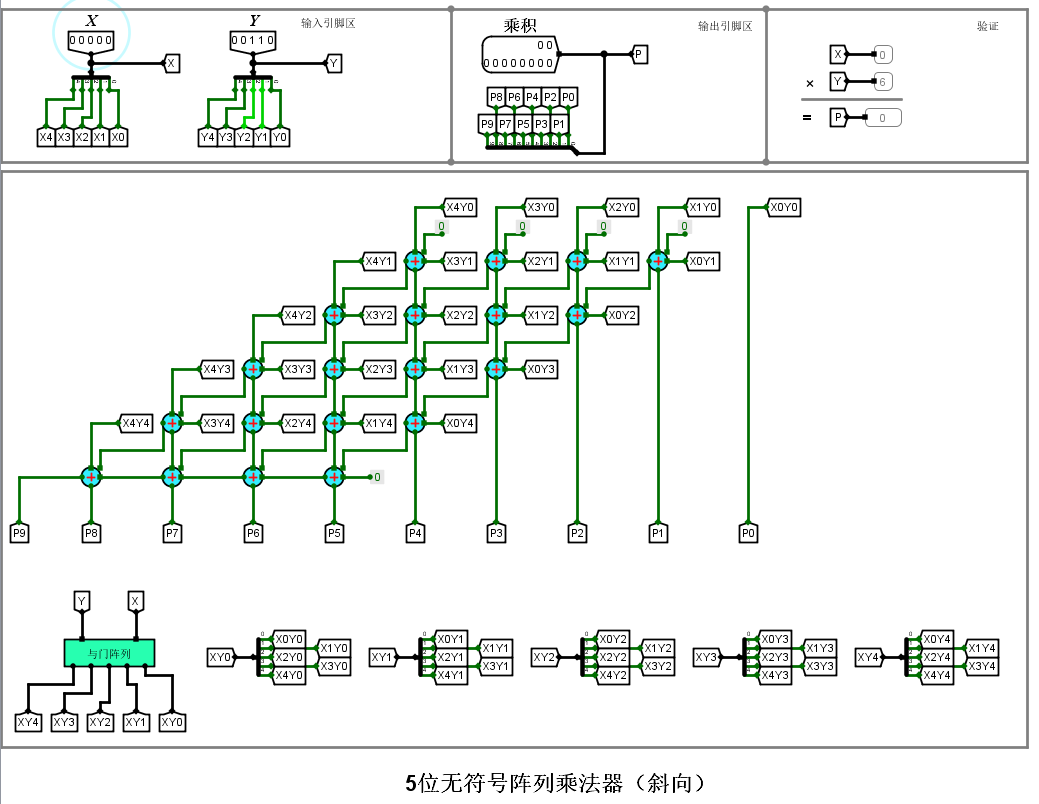
请同学们通过改变X、Y的值，对上述5位无符号阵列乘法器（斜向、横向）进行验证：

斜向：

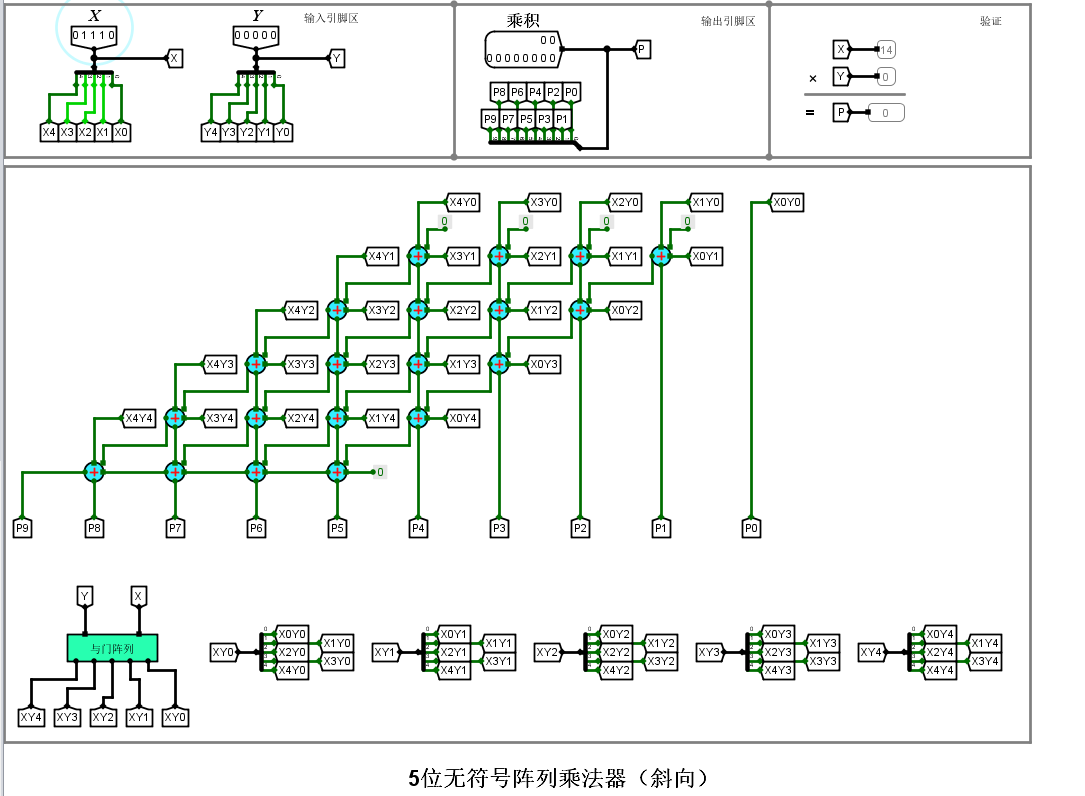
X=非0，Y=非0，P=？P=X\*Y



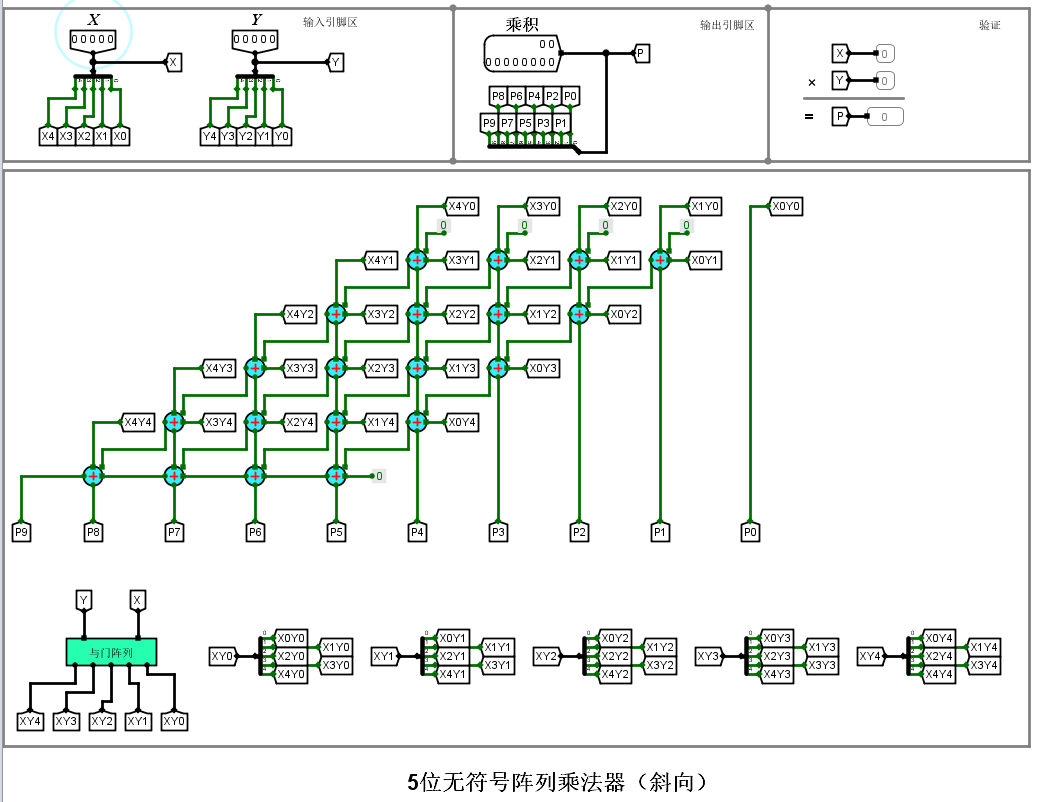
X=0，Y=非0，P是不是=0？是



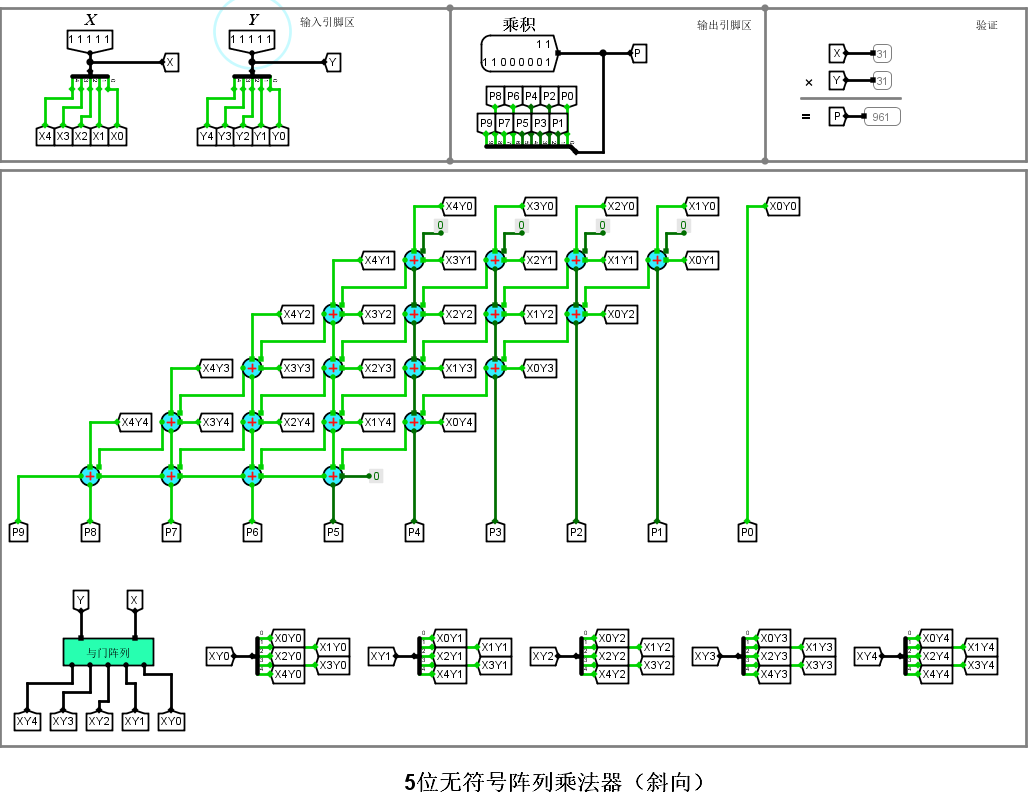
X=非0，Y=0，P是不是=0？是



X=0，Y=0，P是不是=0？是

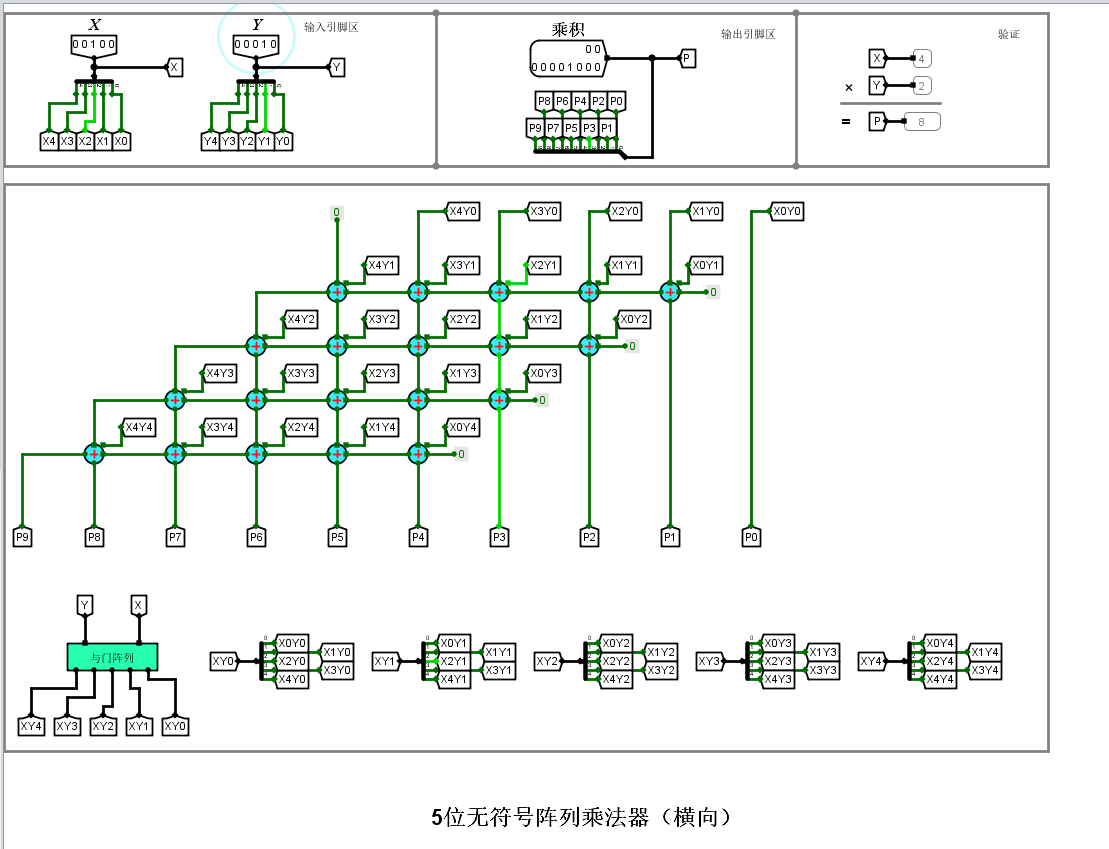


X=31，Y=31，P是不是=961？是

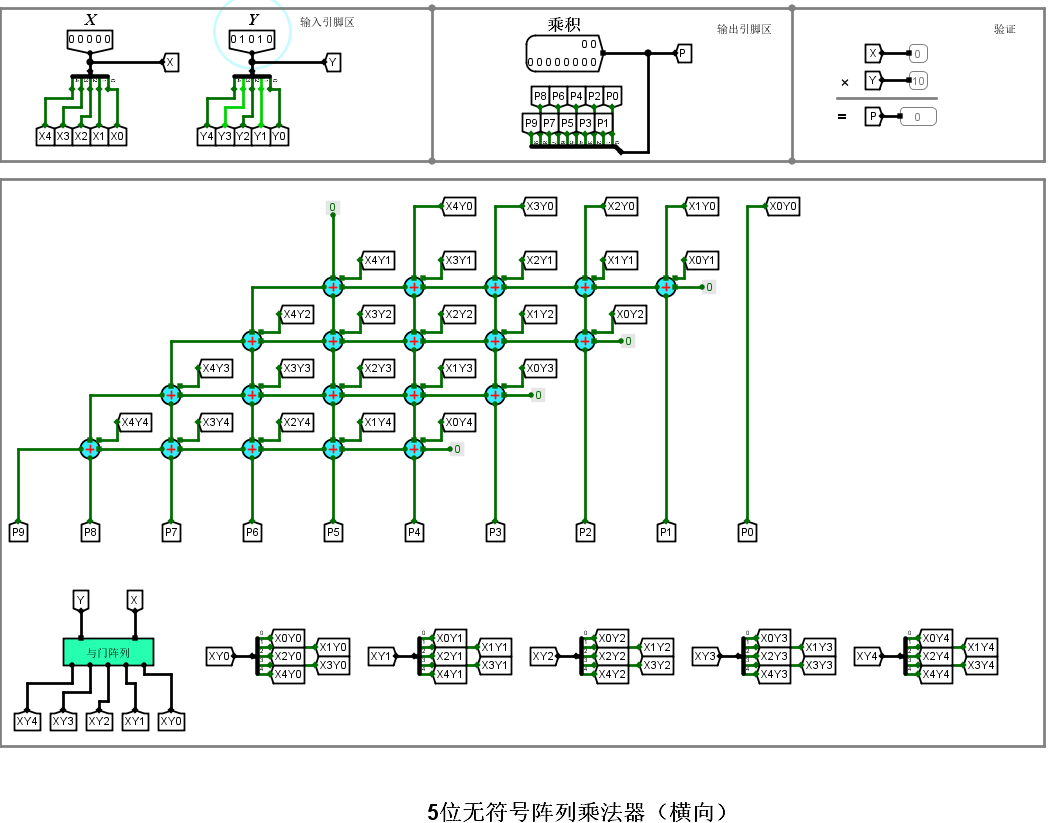


横向：

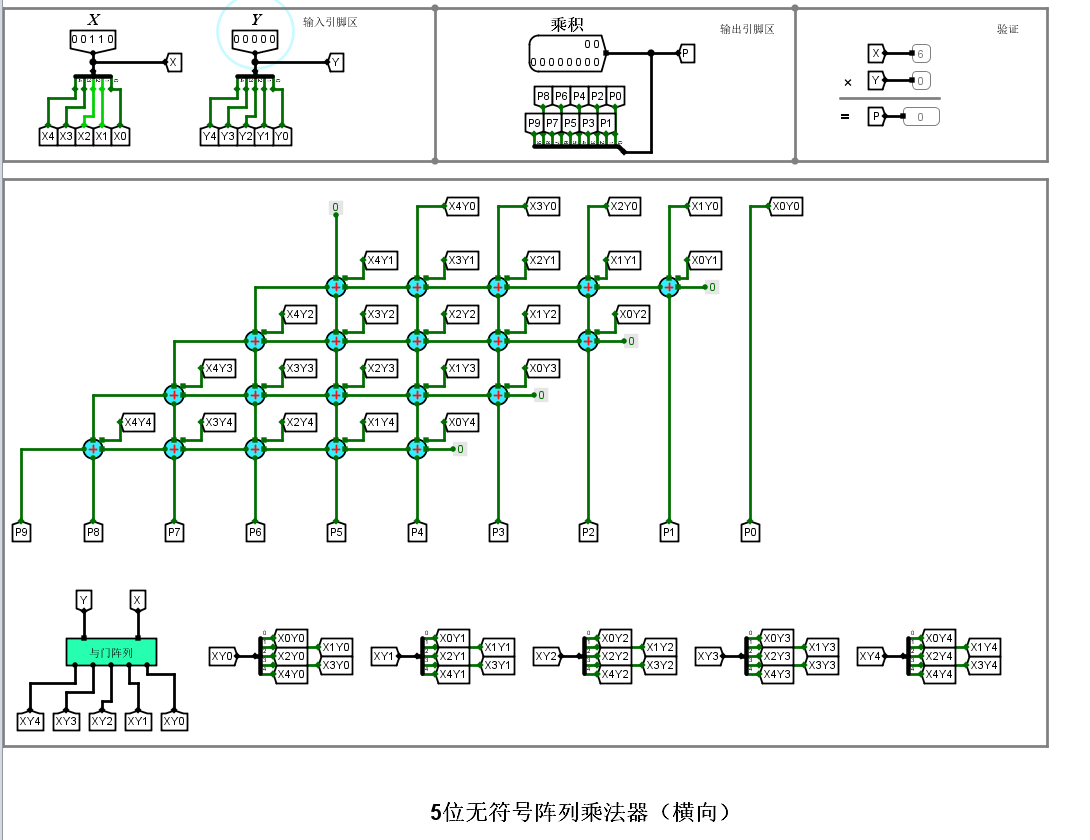
X=非0，Y=非0，P=？P=X\*Y



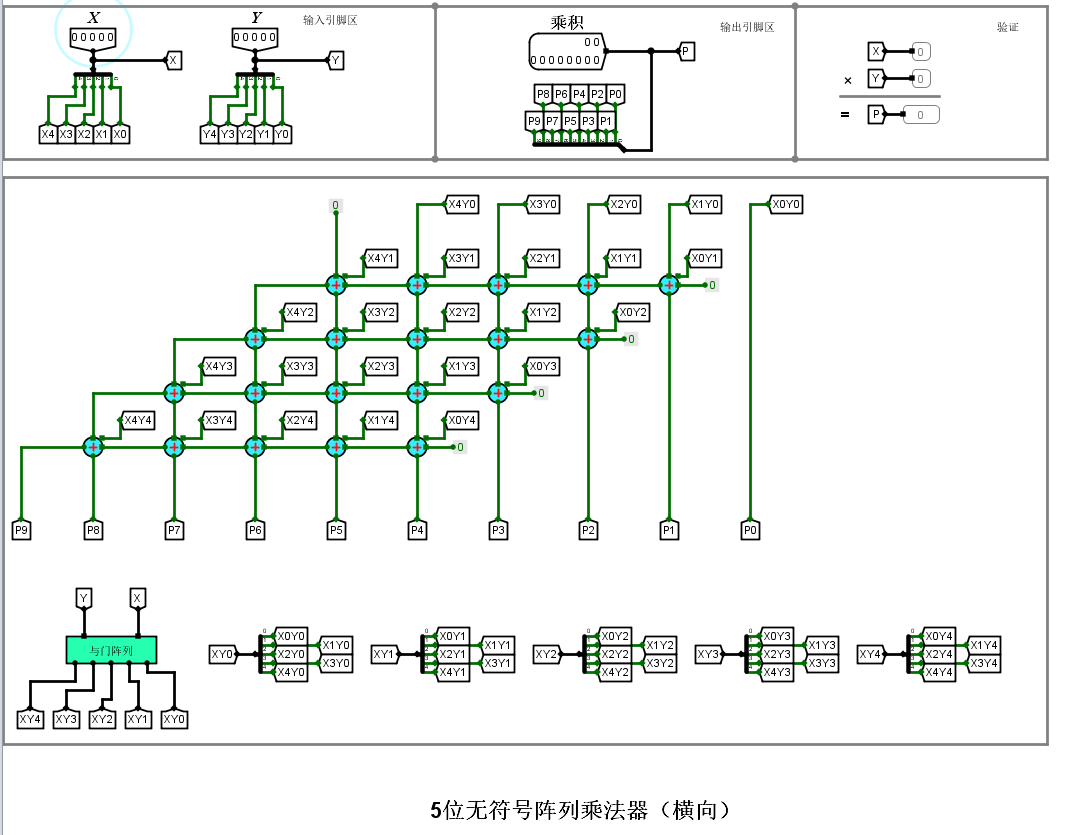
X=0，Y=非0，P是不是=0？是



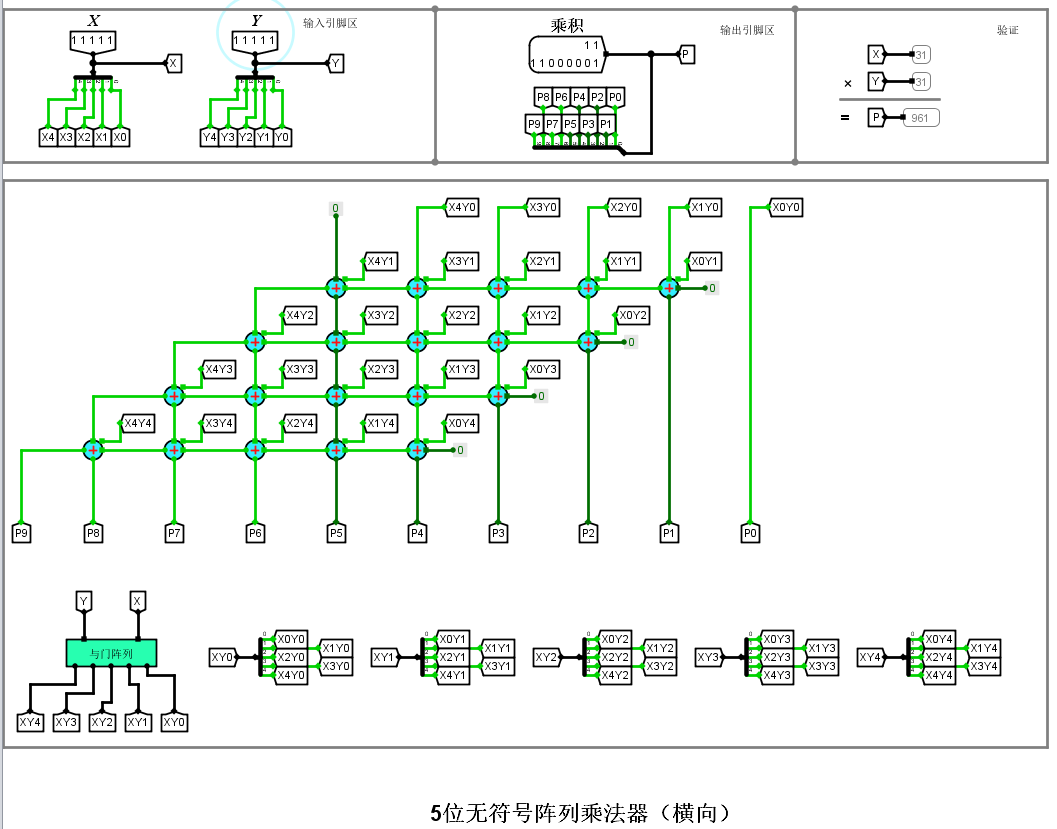
X=非0，Y=0，P是不是=0？是



X=0，Y=0，P是不是=0？是



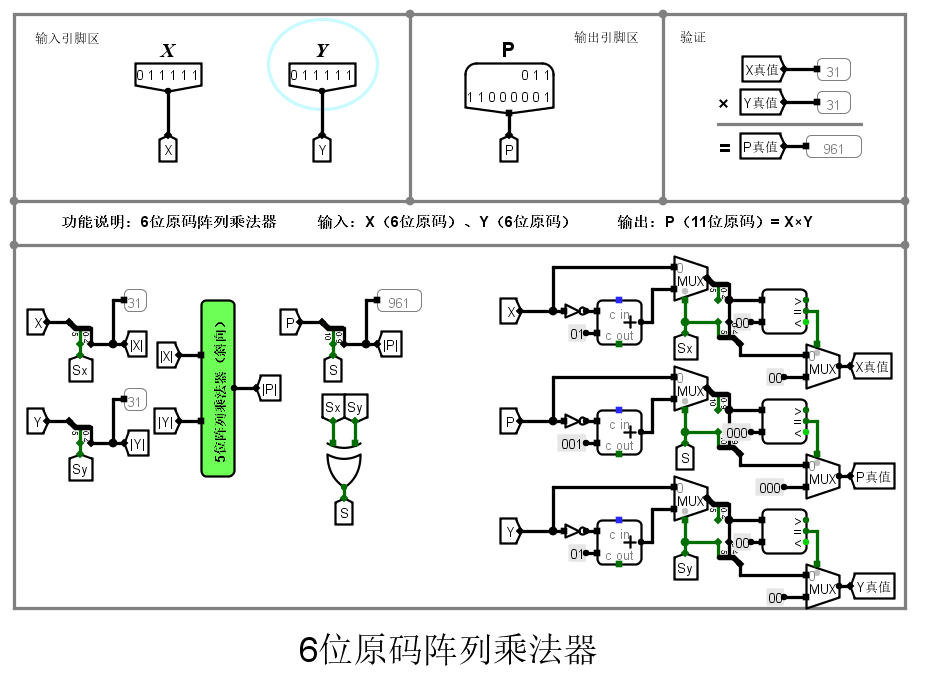
X=31，Y=31，P是不是=961？是



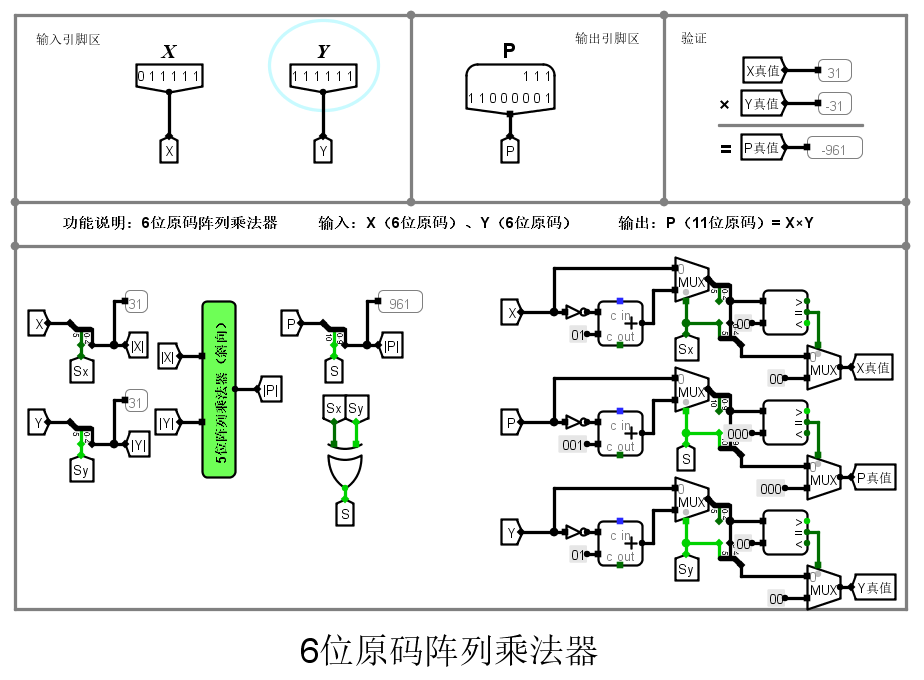
3、6位原码阵列乘法器（设计实验）

请同学们通过改变X、Y的值，对实现的6位原码阵列乘法器进行验证：

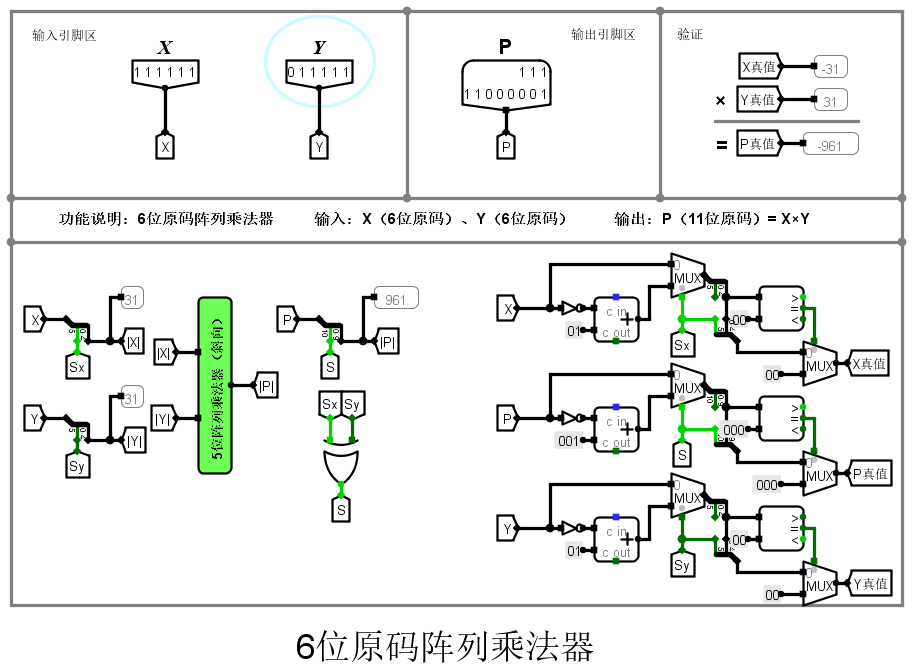
X=正数，Y=正数，P=正数（例如：X=31、Y=31、P=961）



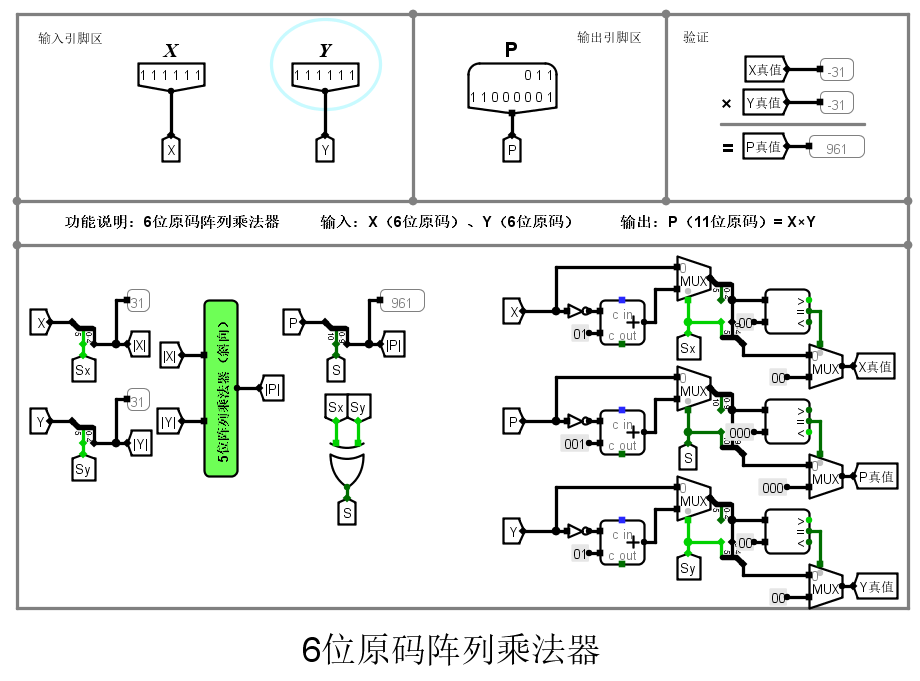
X=正数，Y=负数，P=负数（例如：X=31、Y=-31、P=-961）



X=负数，Y=正数，P=负数（例如：X=-31、Y=31、P=-961）



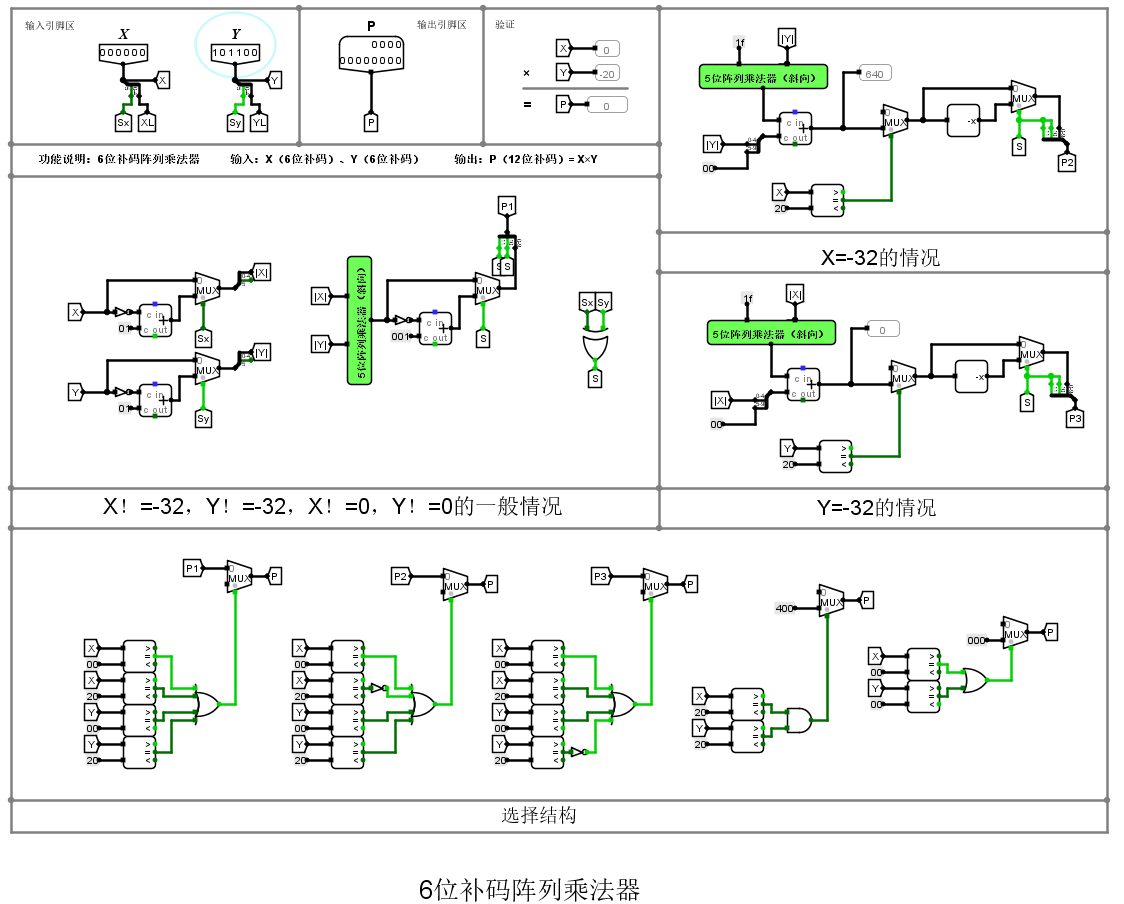
X=负数，Y=负数，P=正数（例如：X=-31、Y=-31、P=961）



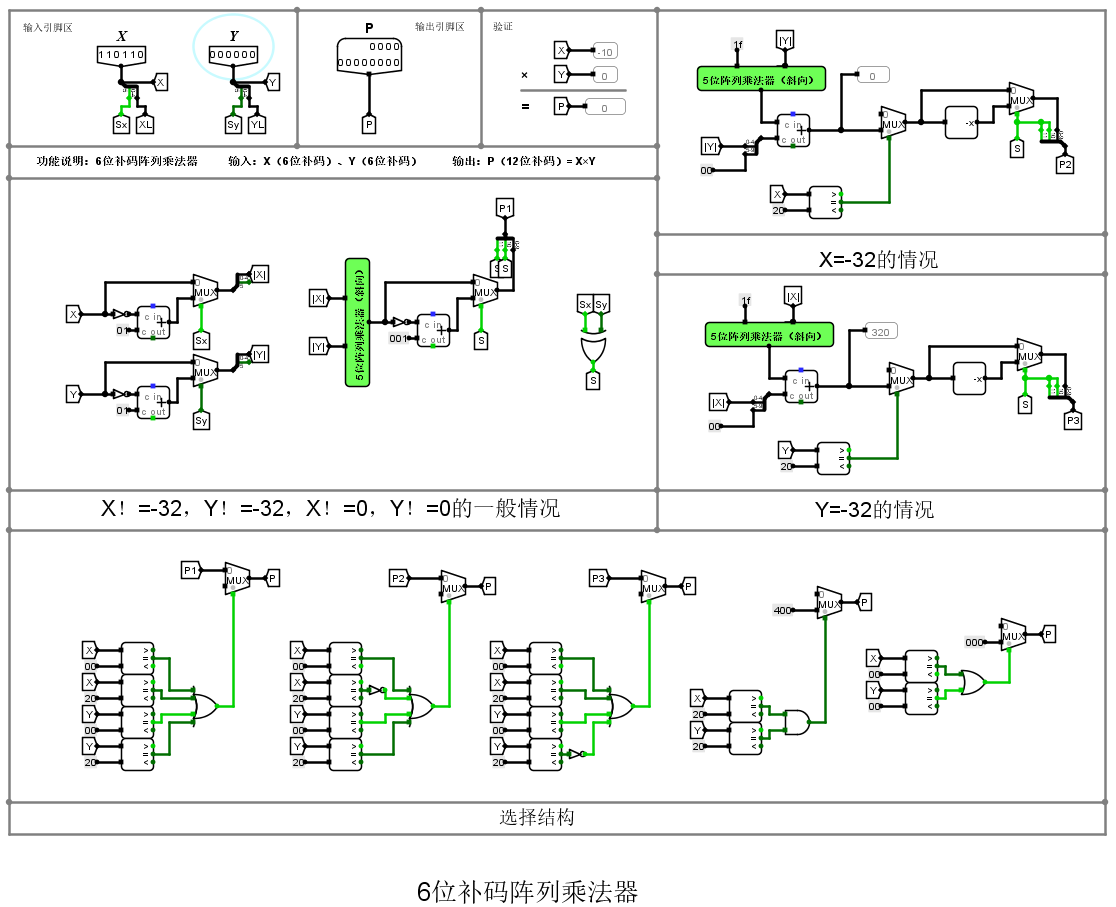
4、6位补码阵列乘法器（设计实验）

请同学们通过改变X、Y的值，对上述6位补码阵列乘法器进行验证：

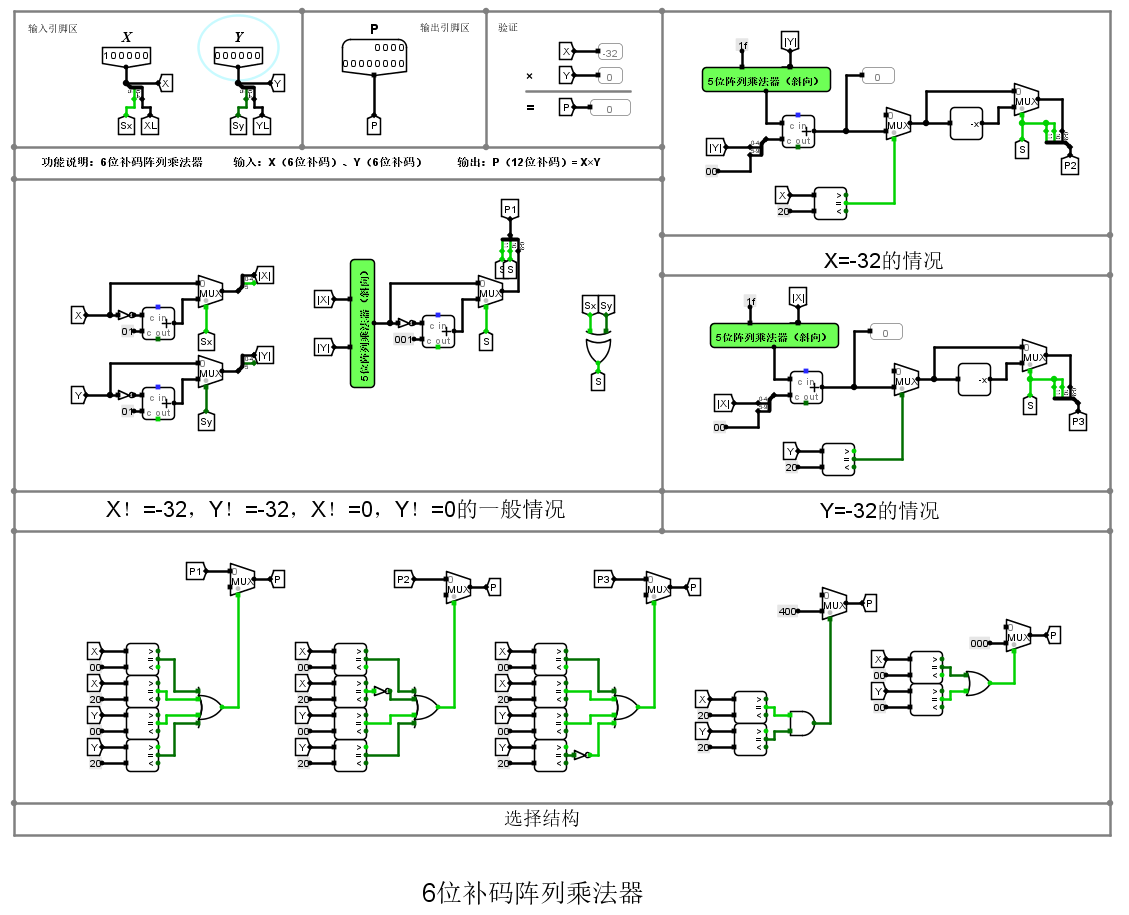
① X=0，Y=负数；P=0

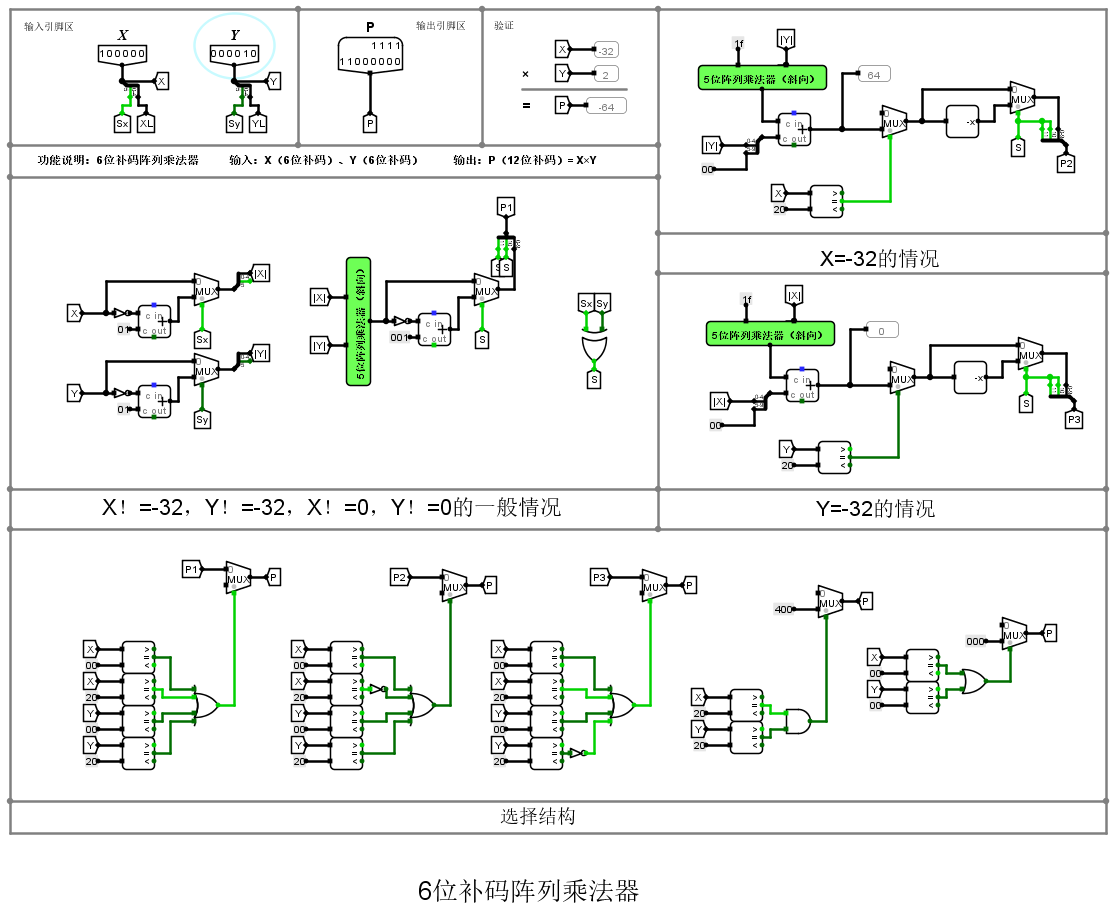


② X=负数，Y=0；P=0

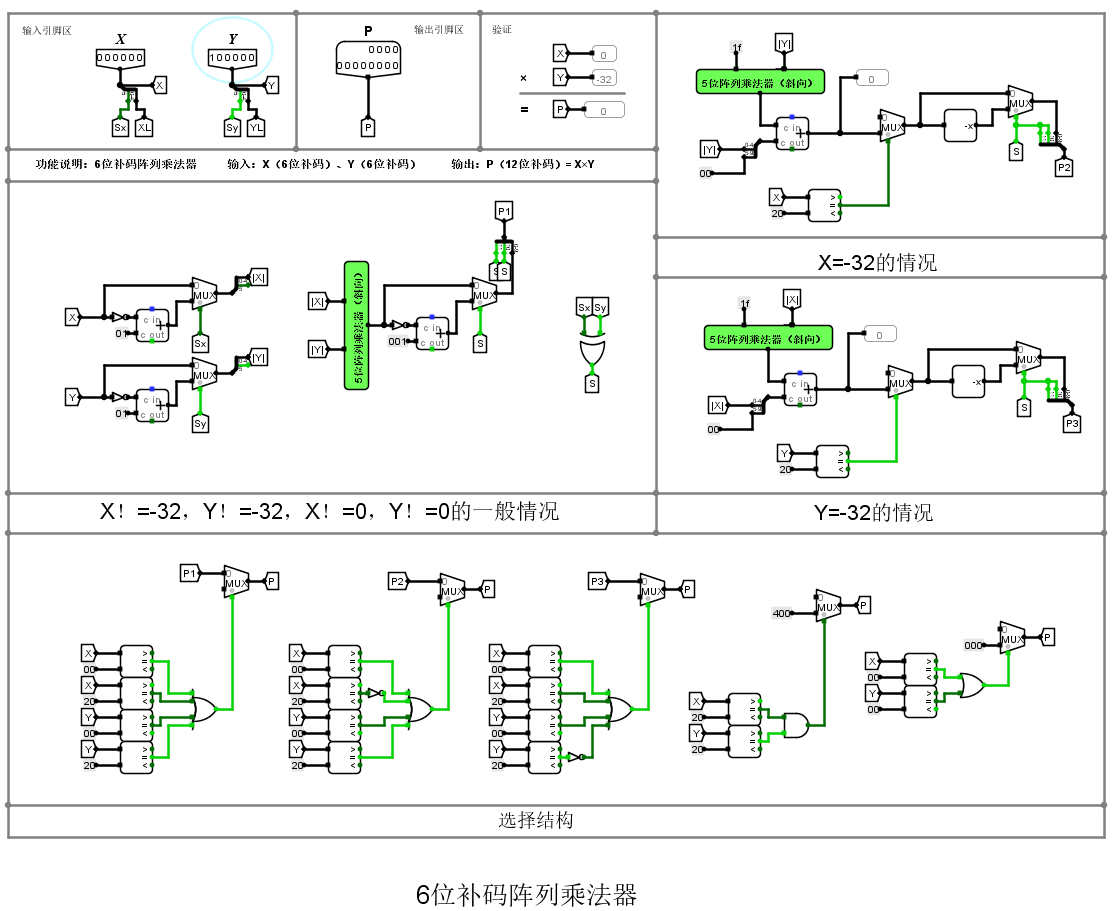


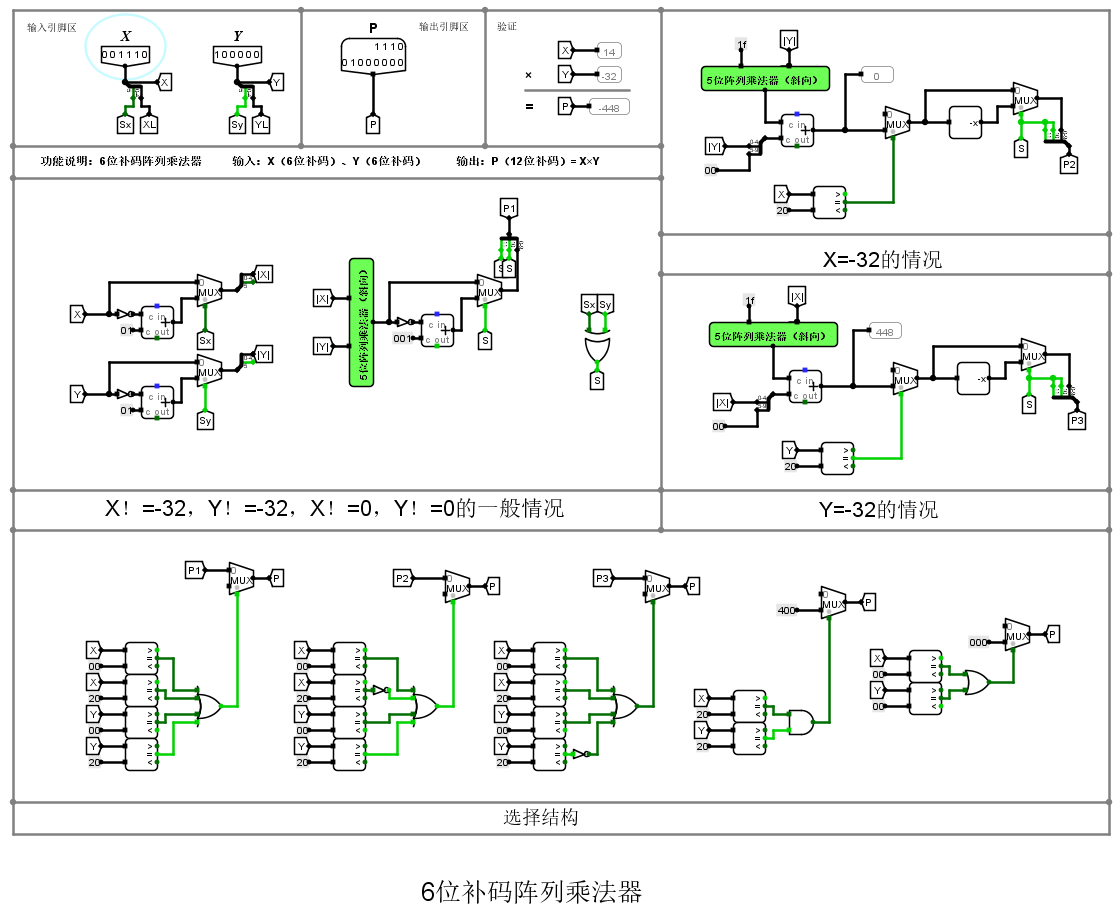
③ X=-32，Y=0或正数；P=正确的数



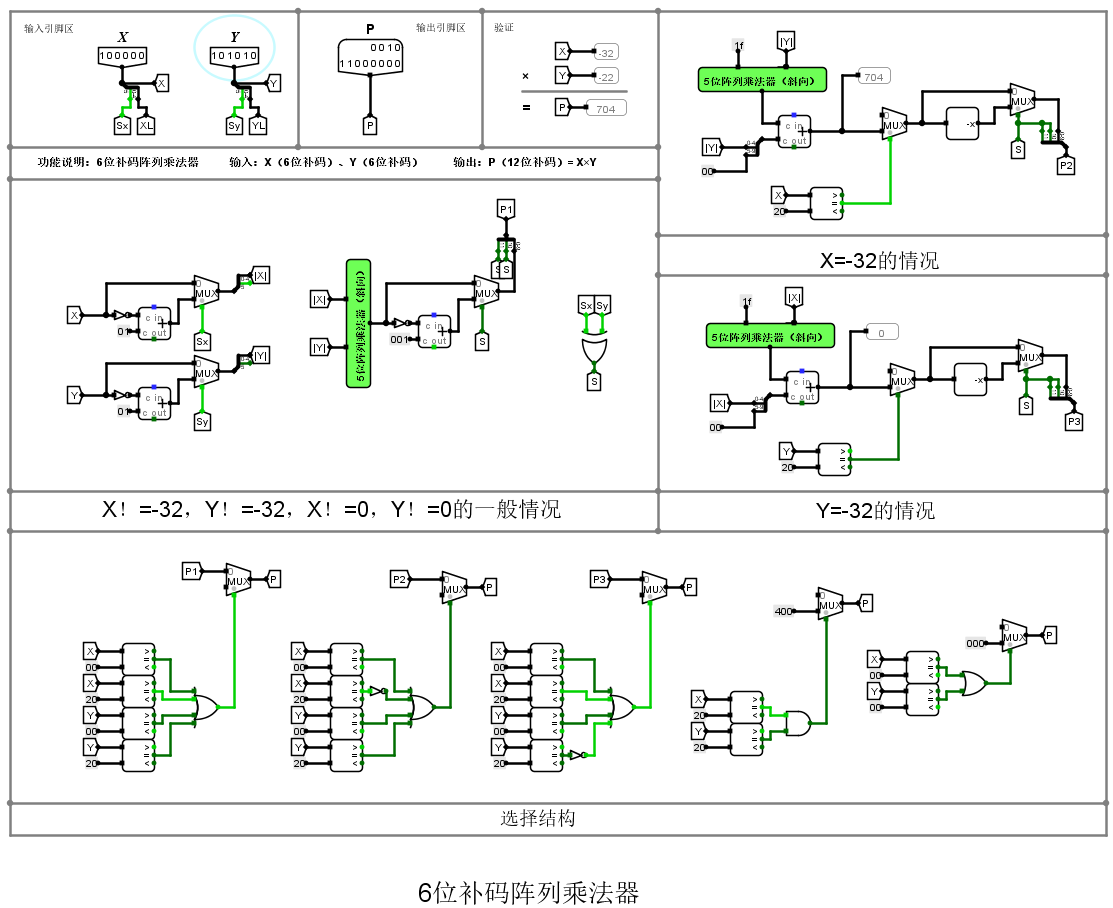


④ X=0或正数，Y=-32；P=正确的数

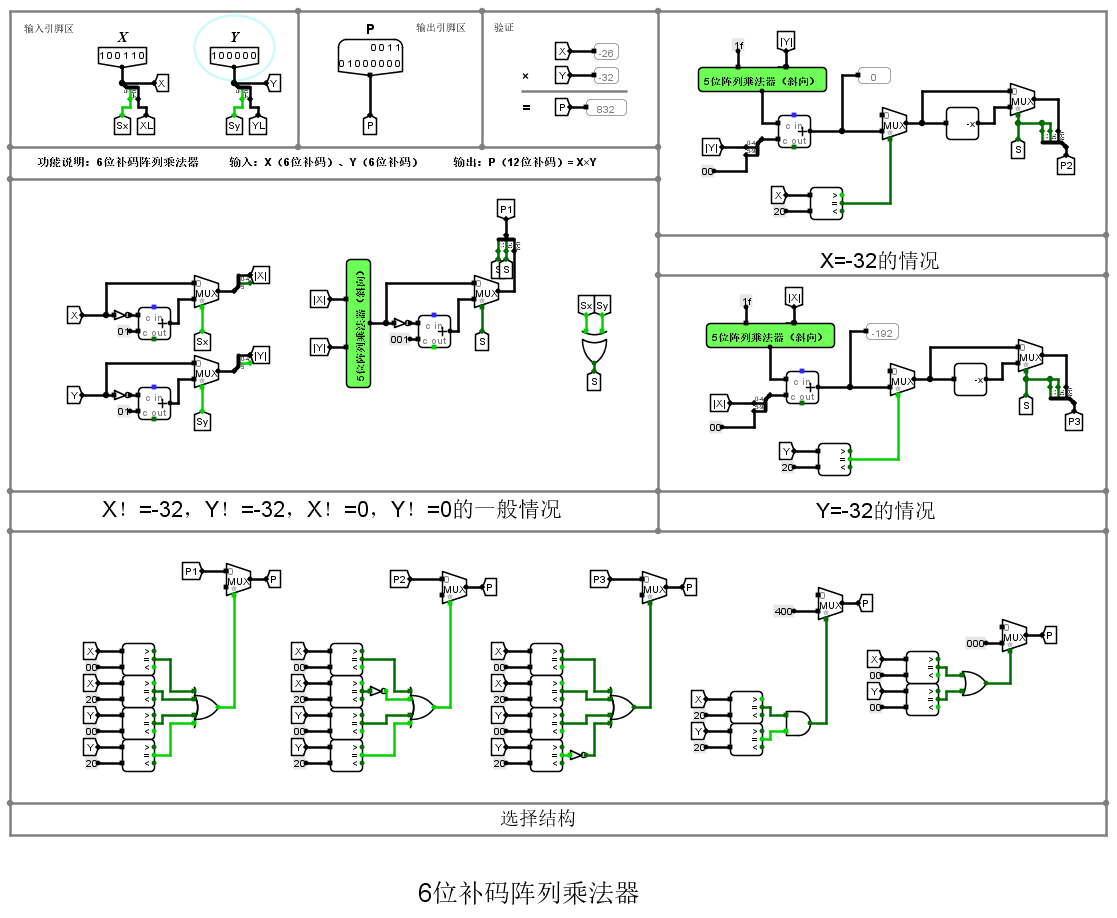




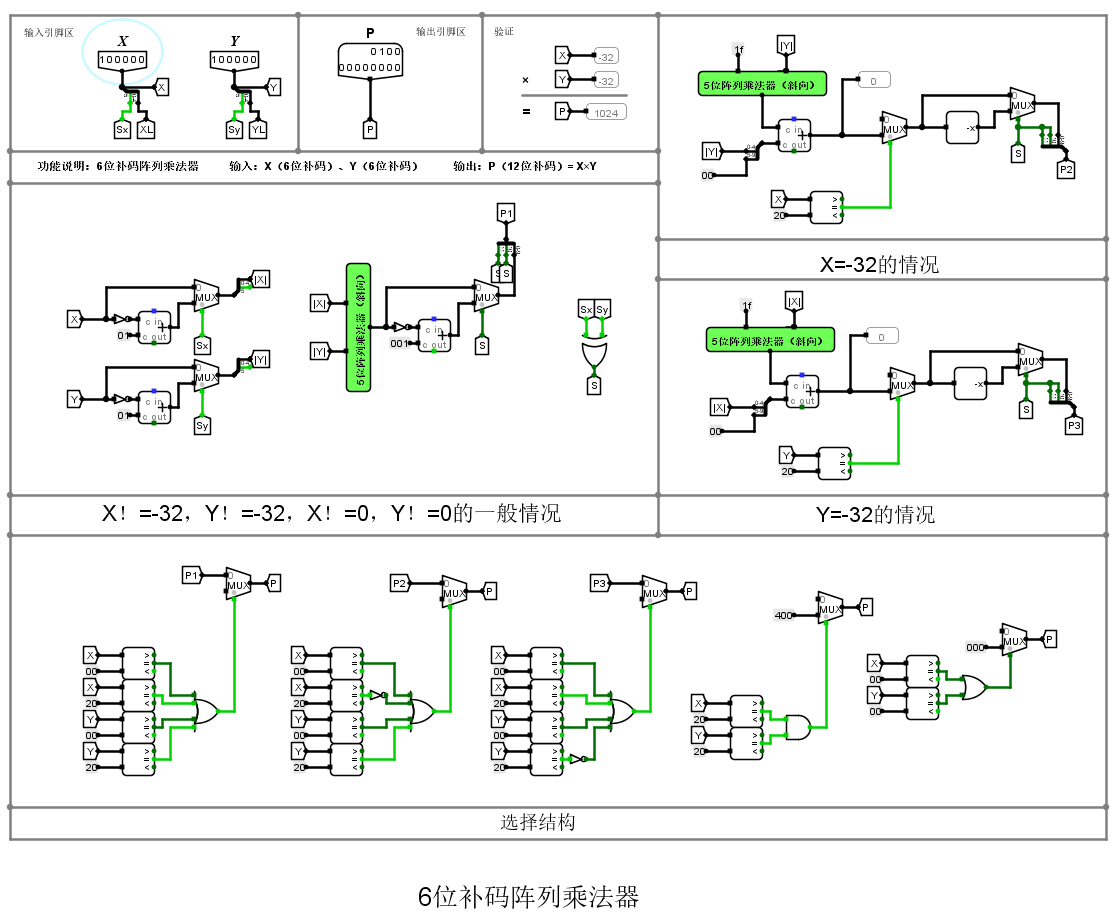
⑤ X=-32，Y=负数；P=正确的数



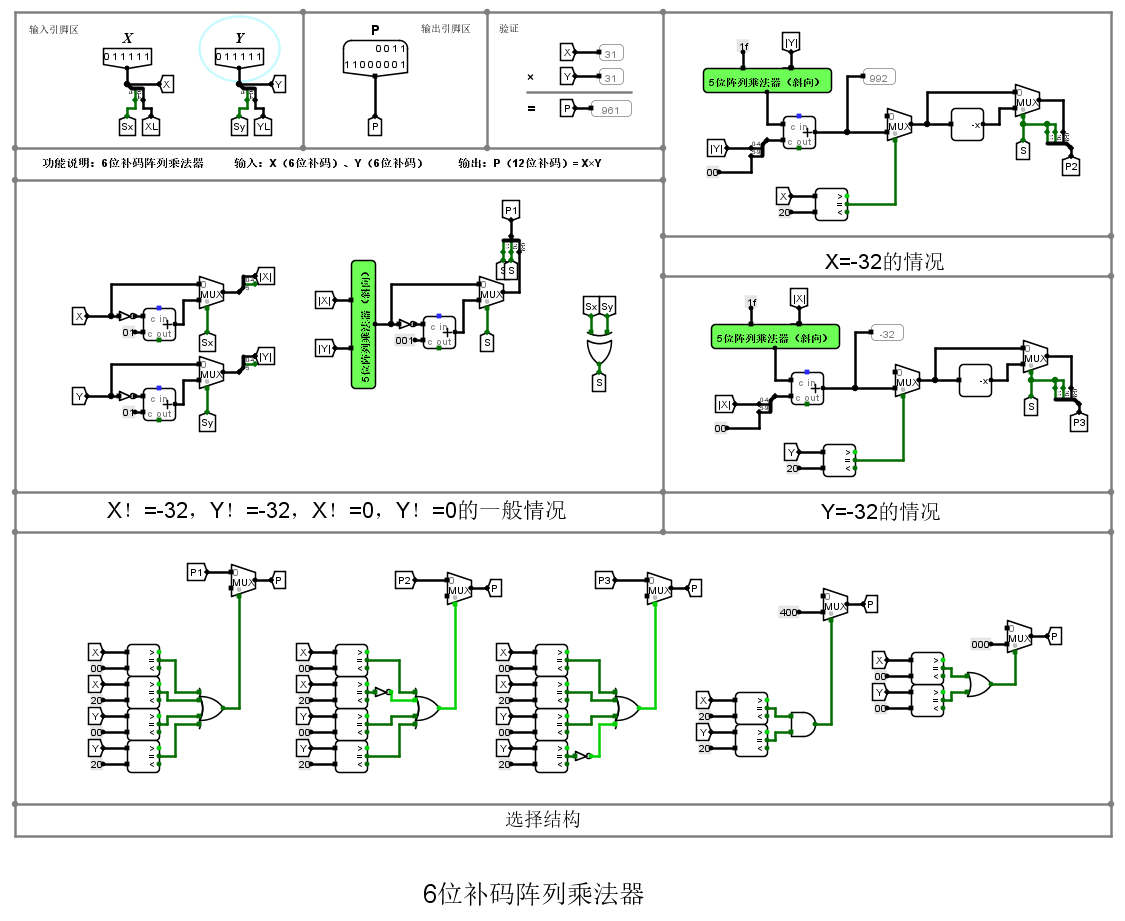
⑥ X=负数，Y=-32；P=正确的数



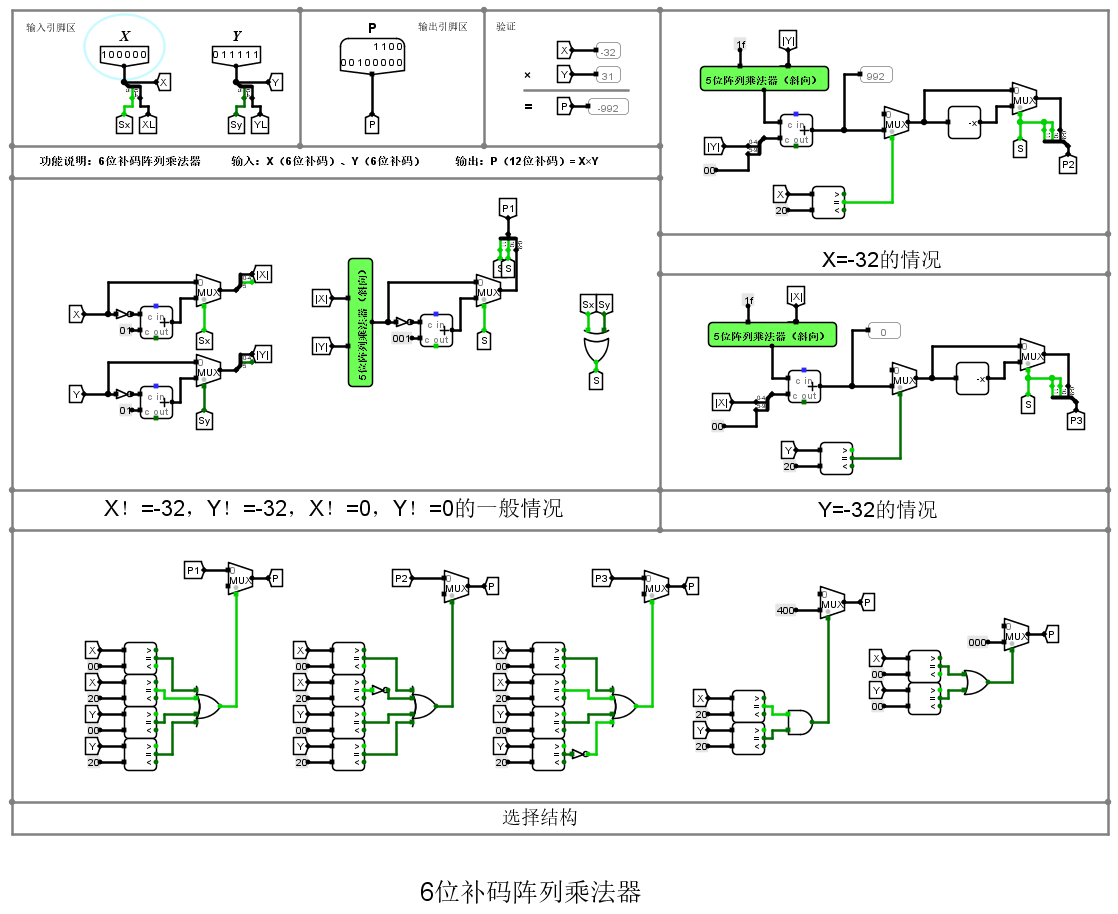
⑦ X=-32，Y=-32，P=1024



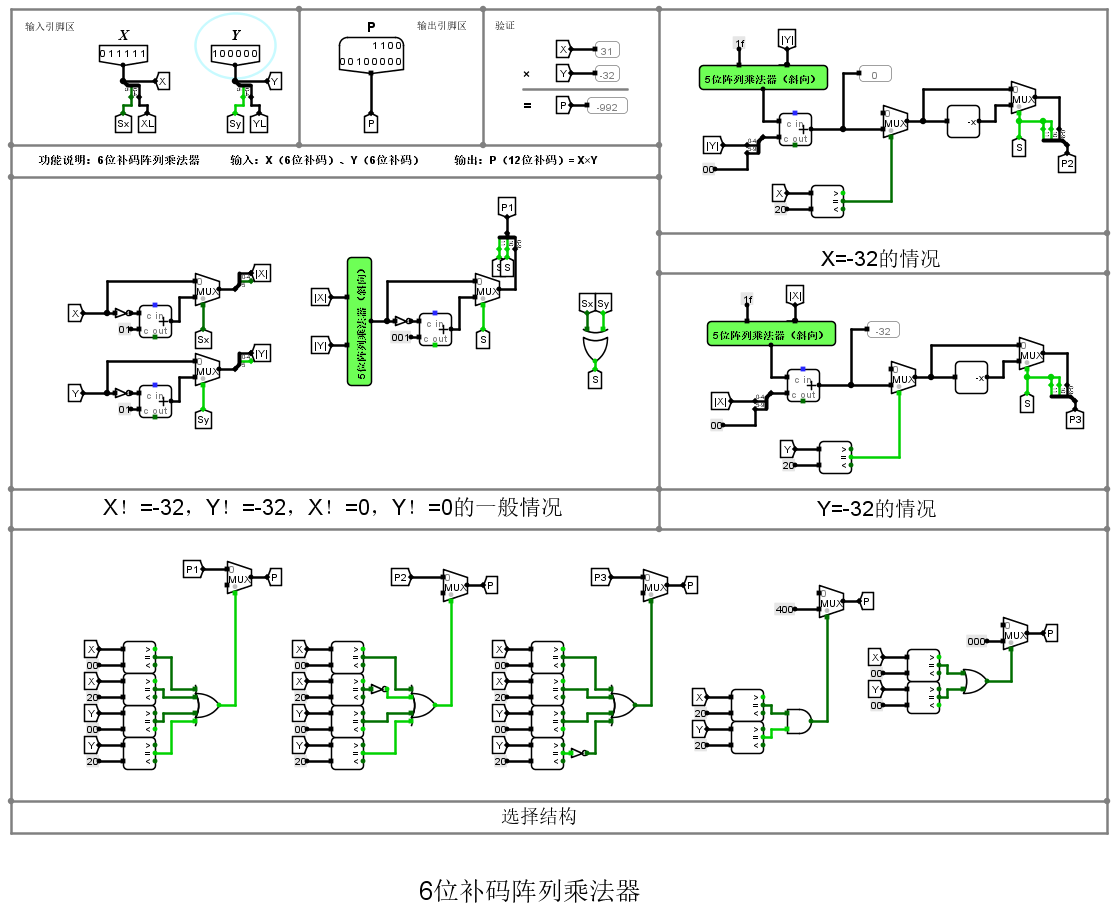
⑧ X=31，Y=31，P=961



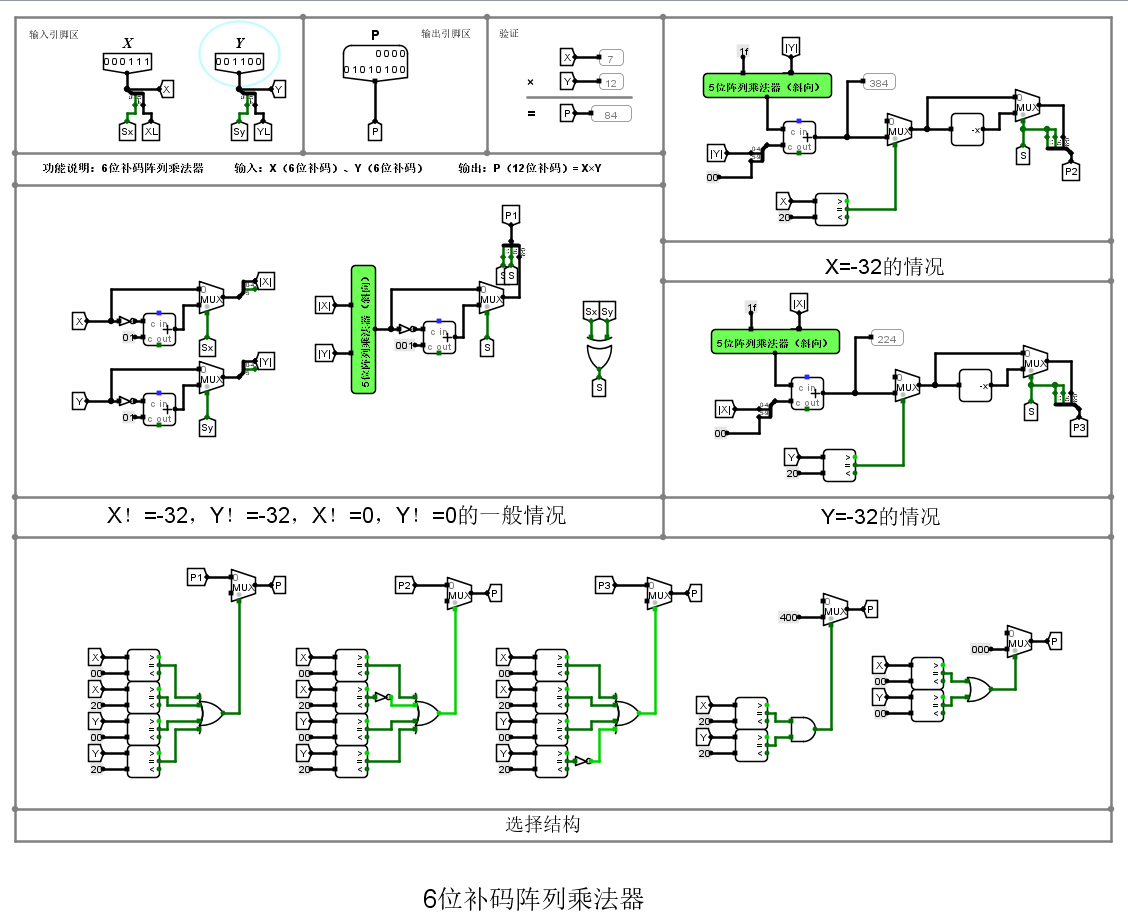
⑨ X=-32，Y=31，P=-992



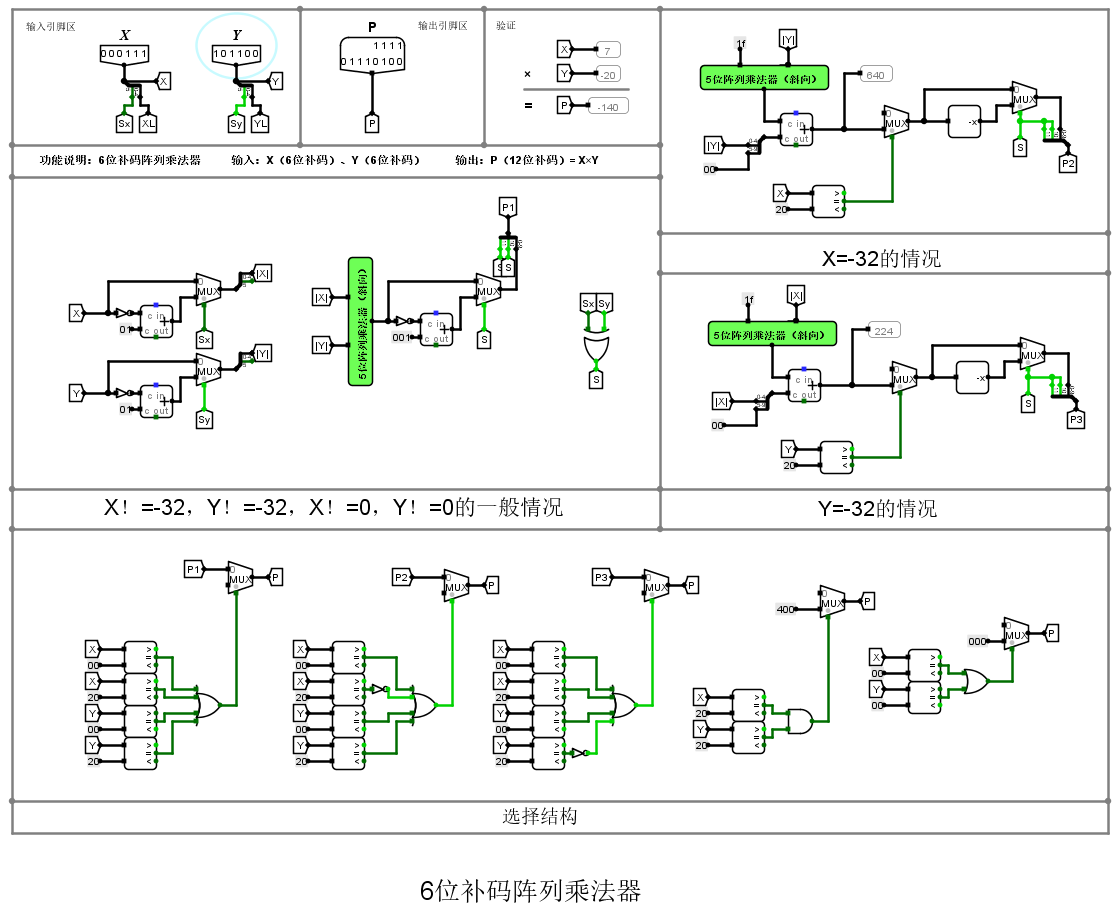
⑩ X=31，Y=-32，P=-992



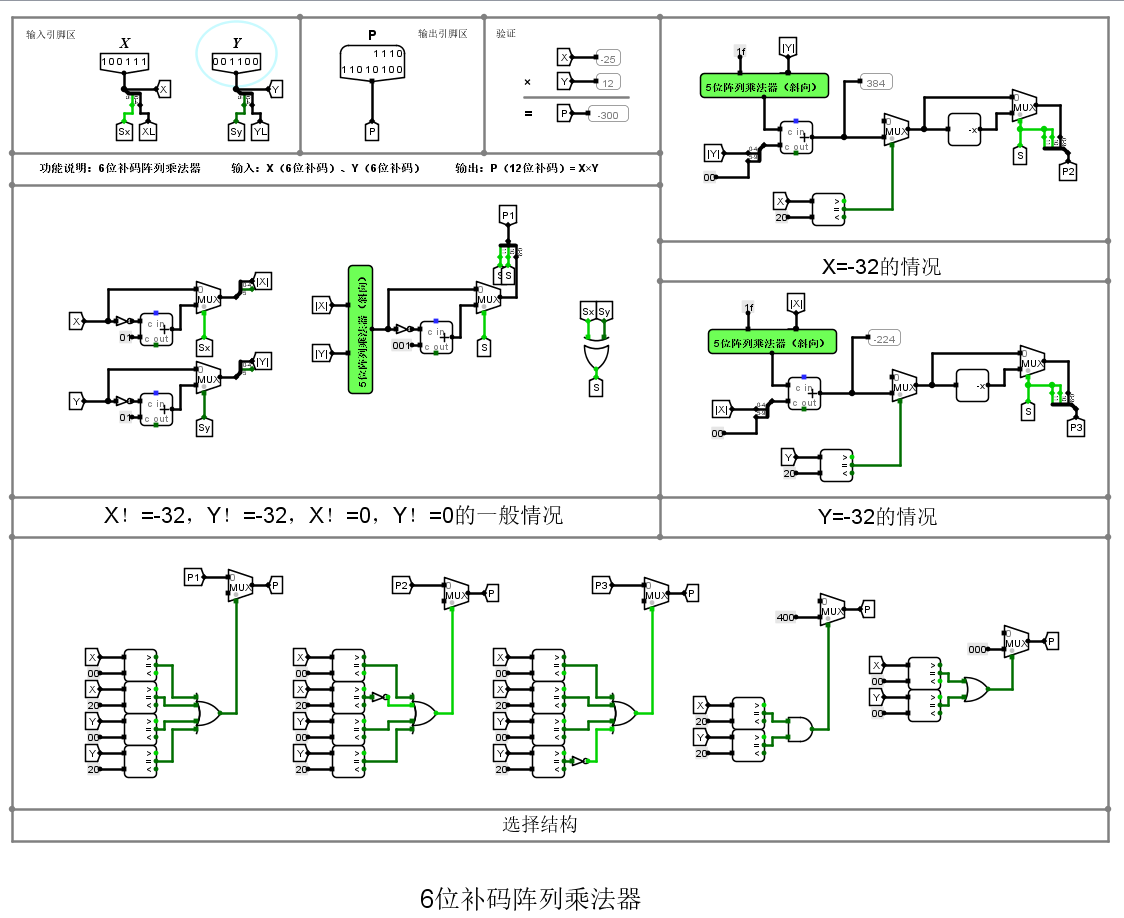
⑪ X=正数，Y=正数；P=正确的数



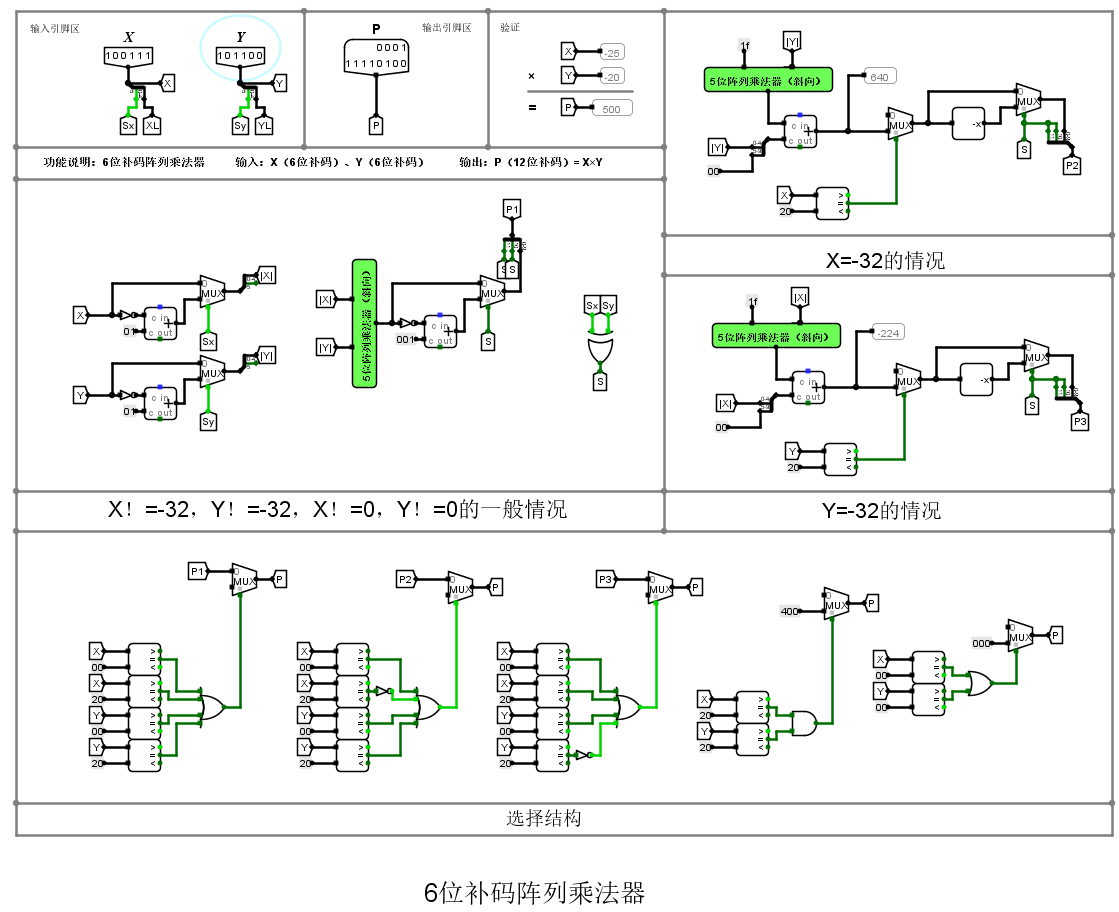
⑫ X=正数，Y=负数；P=正确的数



⑬ X=负数，Y=正数；P=正确的数



⑭ X=负数，Y=负数；P=正确的数

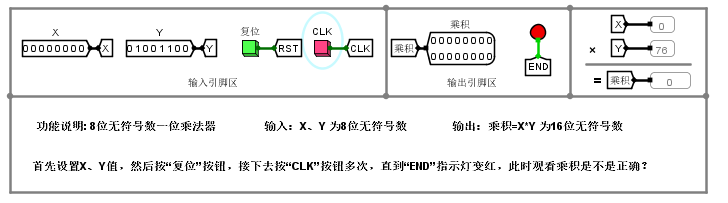


六、原码和补码一位乘法器

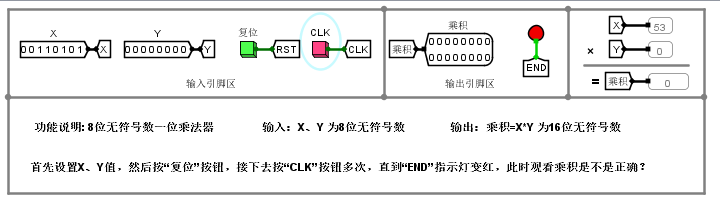
1、8位无符号数一位乘法器

请同学们设置不同的X、Y值（无符号数）

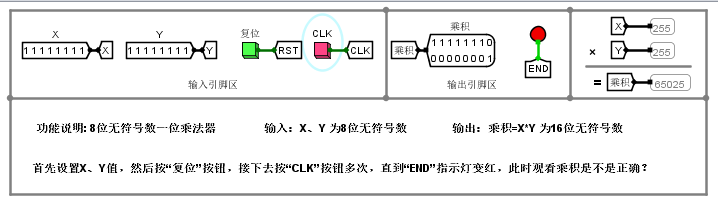
X=0，Y=任意，乘积是否=0是



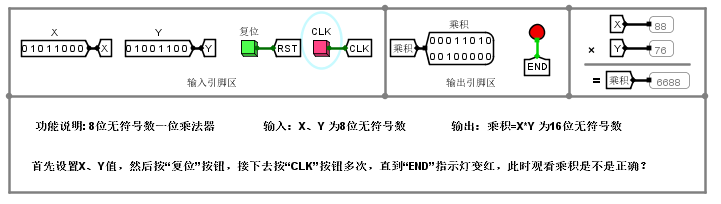
X=任意，Y=0，乘积是否=0是



X=255，Y=255，乘积是否=65025是



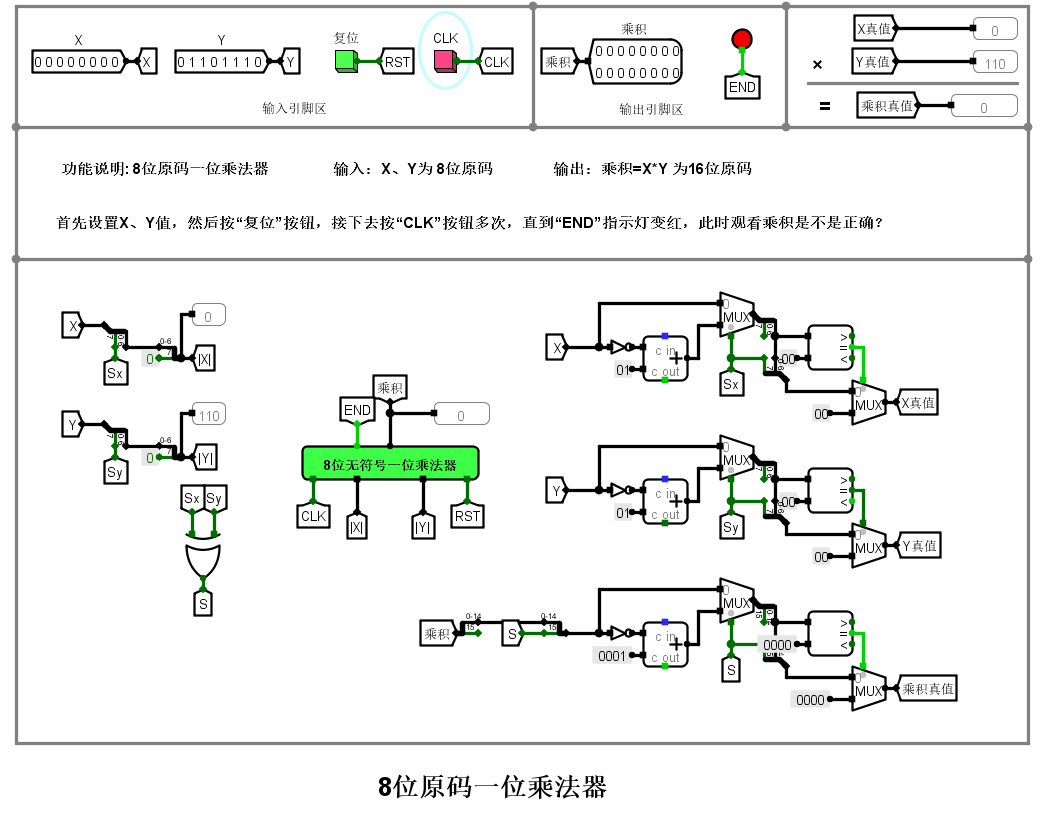
X=其他，Y=其他，乘积是否正确？是



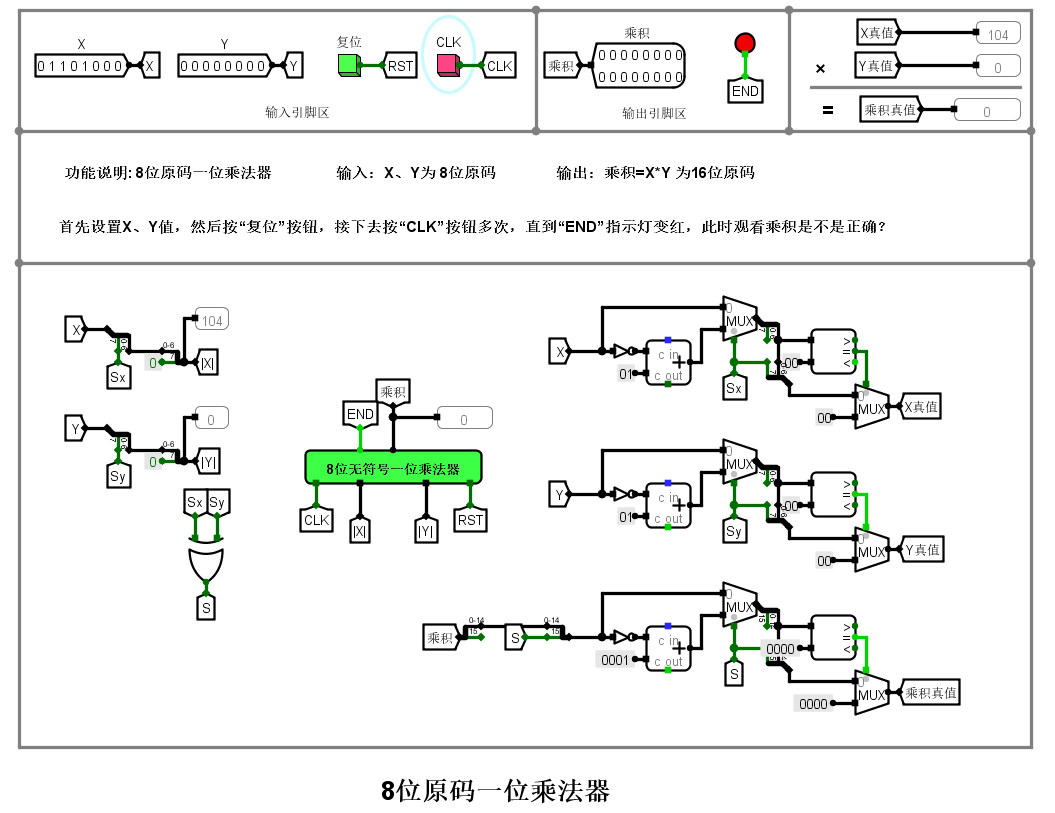
2、8位原码一位乘法器（设计实验）

请同学们设置不同的X、Y值（原码）

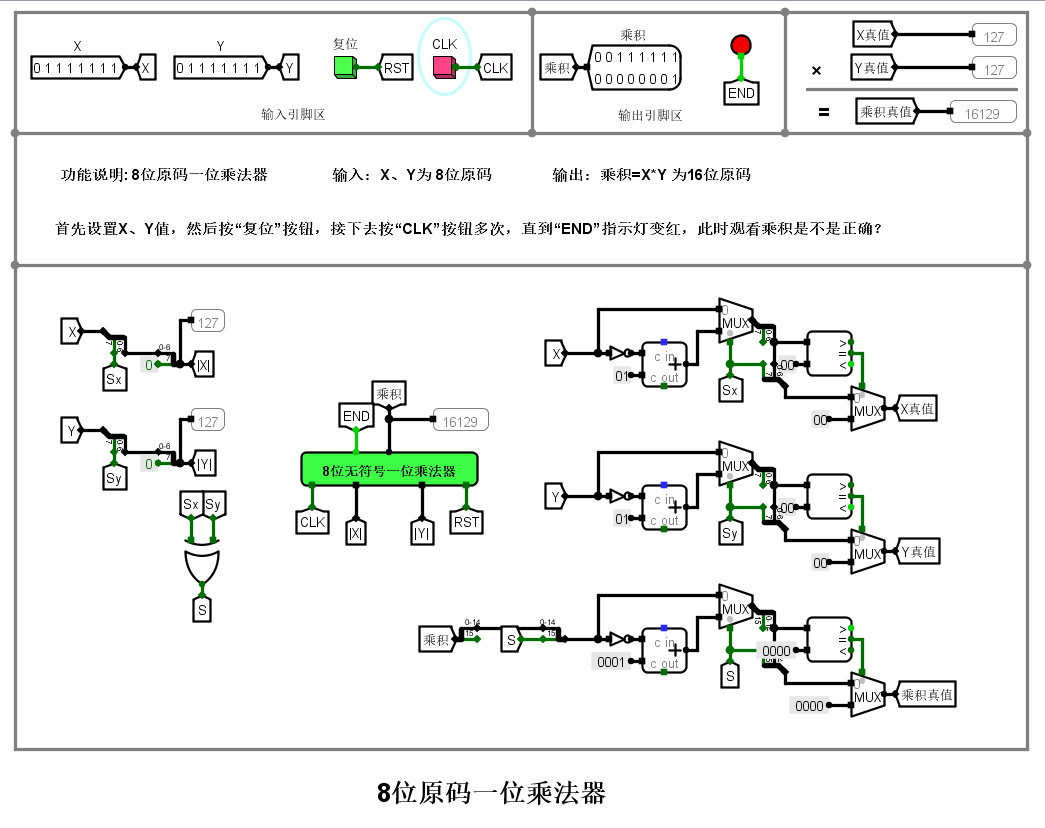
X=0，Y=任意，P是否=0是



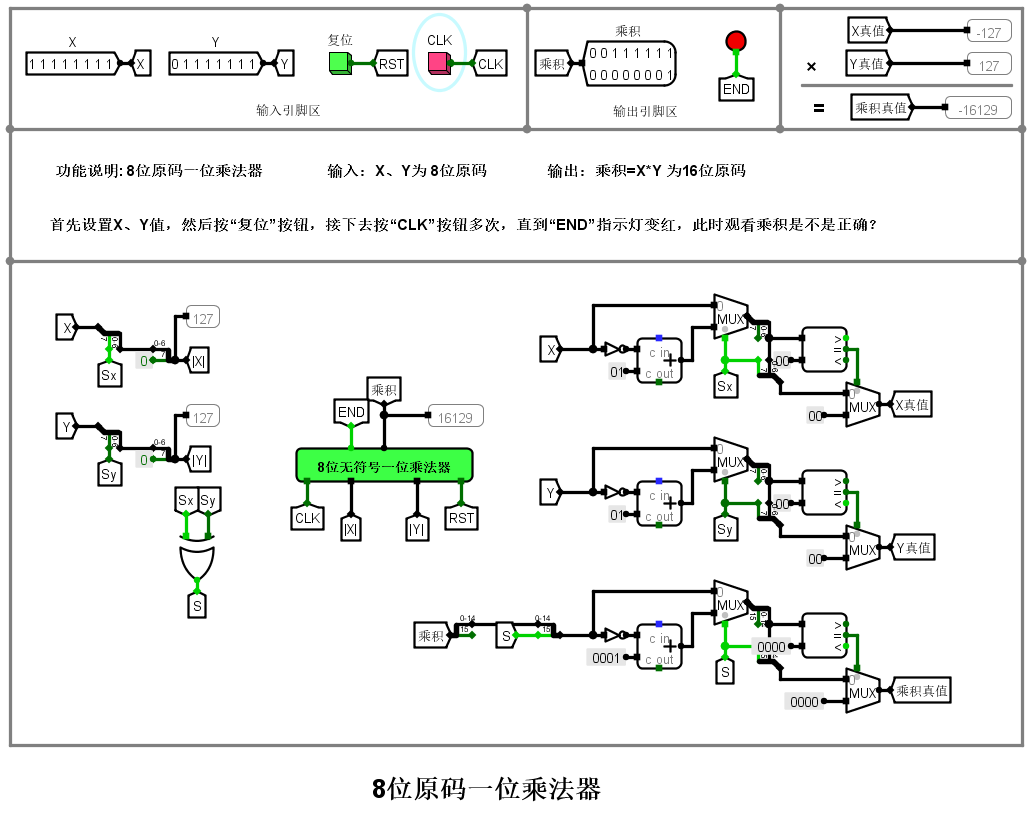
X=任意，Y=0，P是否=0



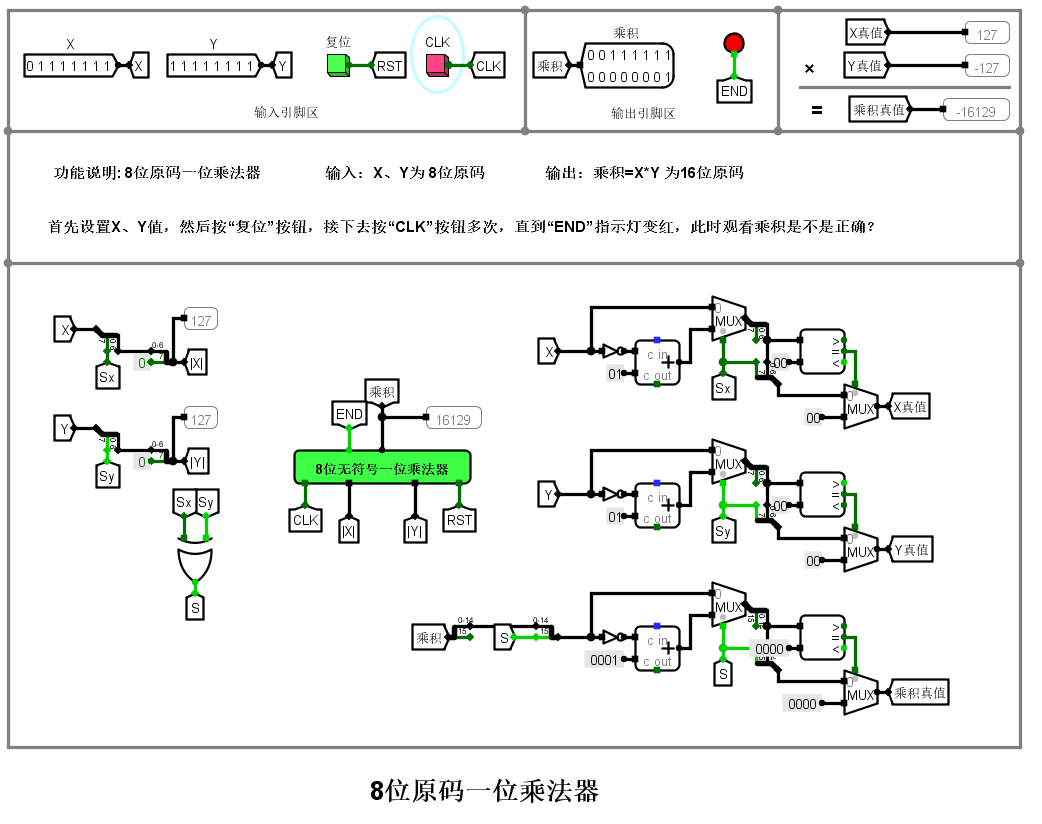
X=127，Y=127，P是否=16129是



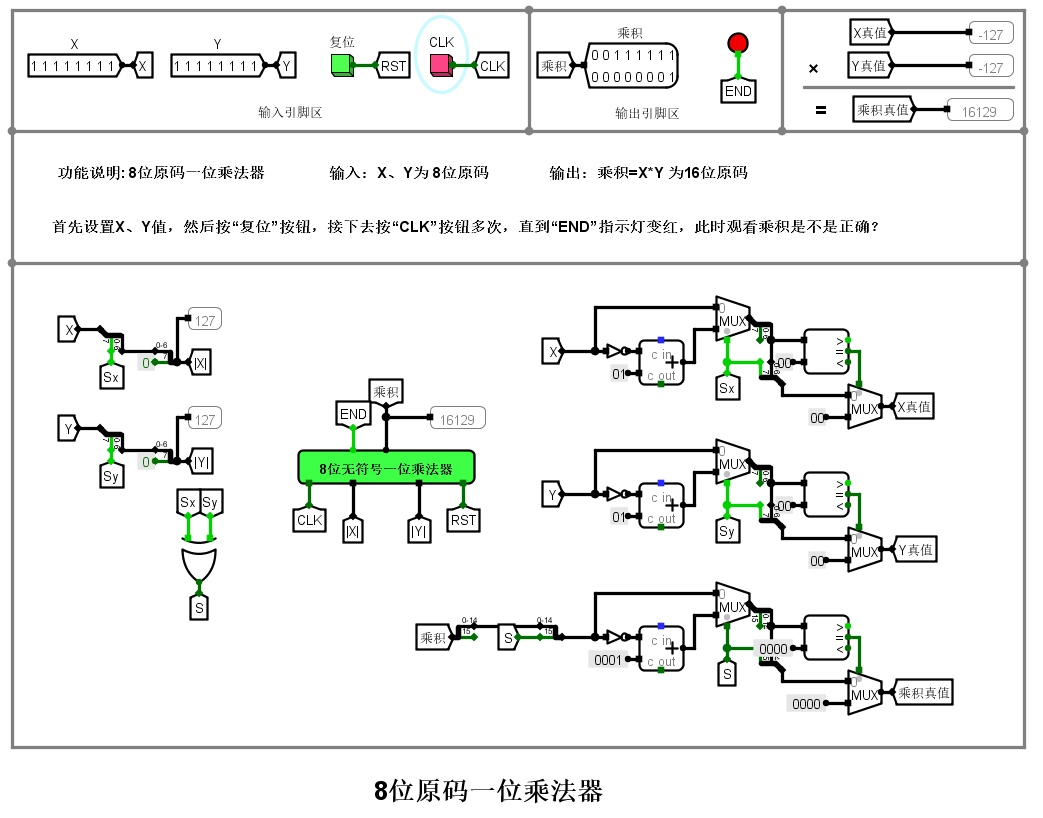
X=-127，Y=127，P是否=-16129是



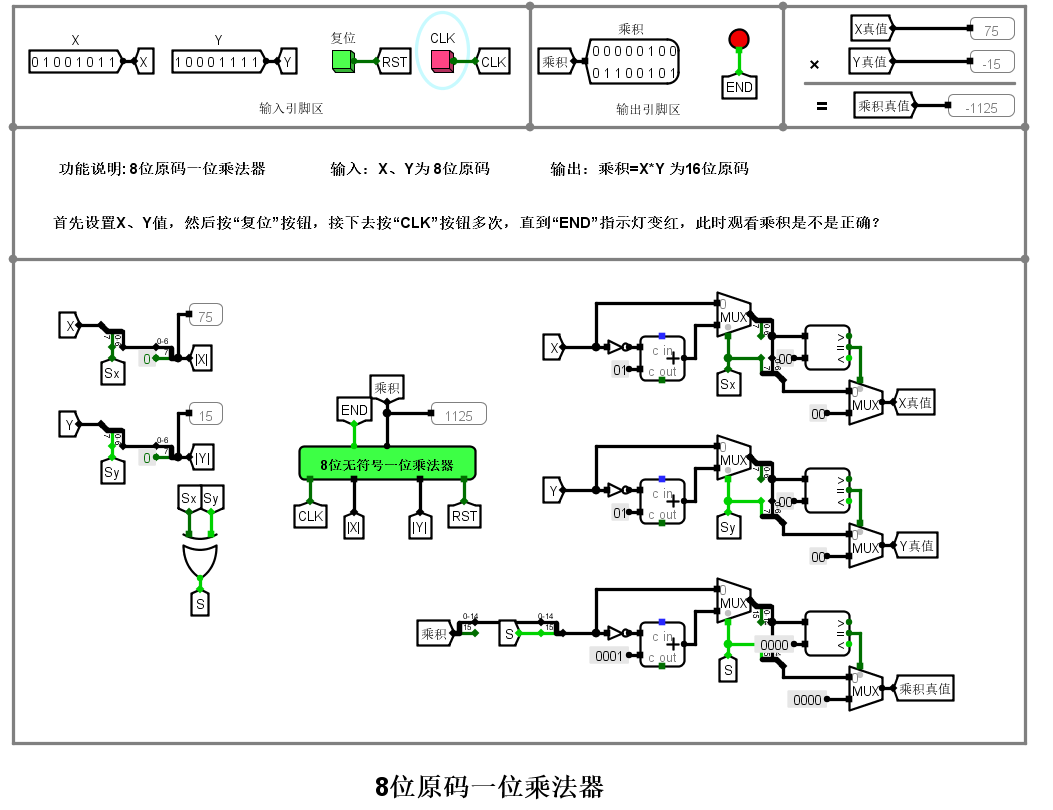
X=127，Y=-127，P是否=-16129



X=-127，Y=-127，P是否=16129是



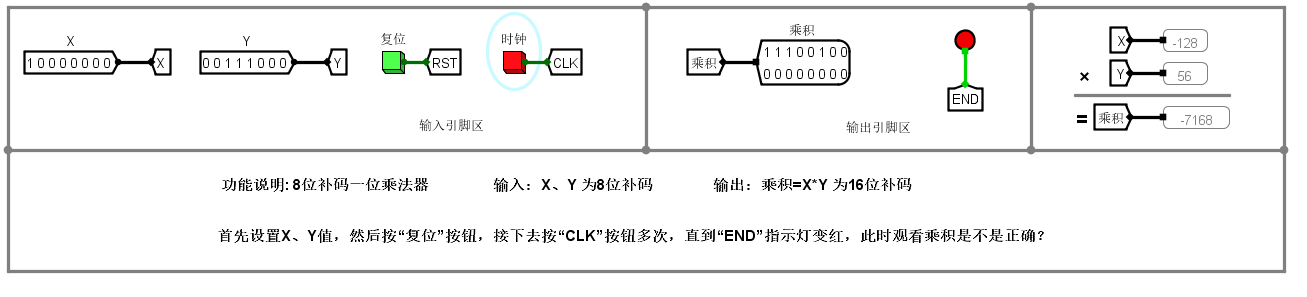
X=其他，Y=其他，P是否正确？是



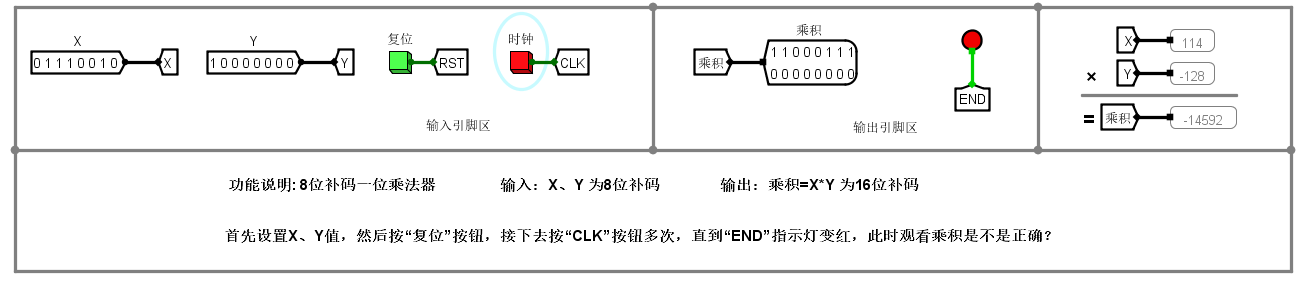
3、8位补码一位乘法器

请同学们设置不同的X、Y值（补码）

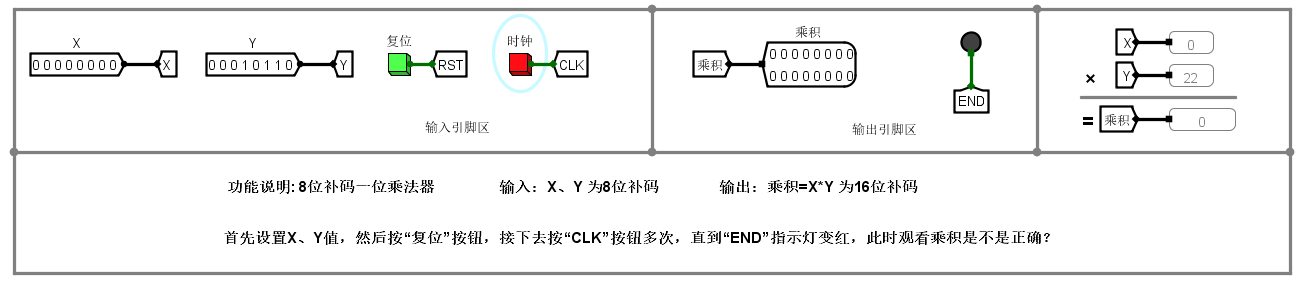
X=-128，Y=其他，P是否正确？是



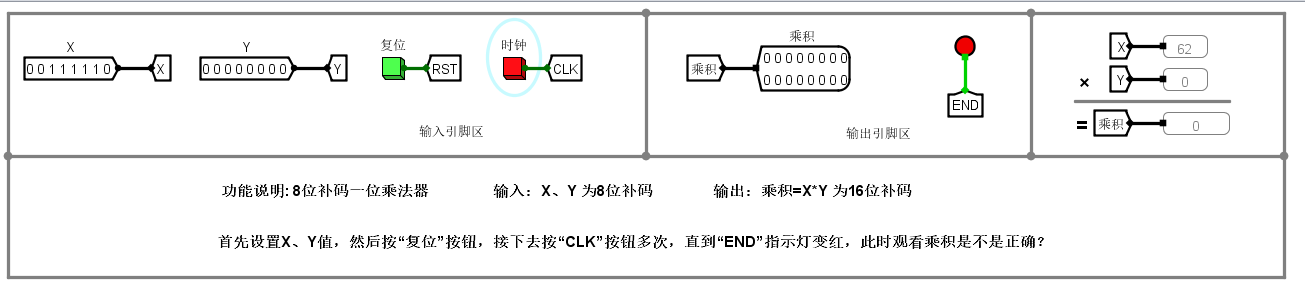
X=其他，Y=-128，P是否正确？是



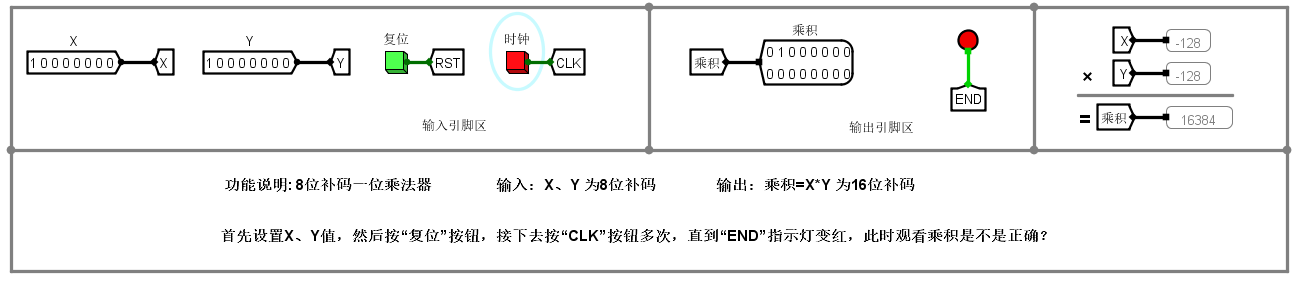
X=0，Y=非0，P是否=0？是



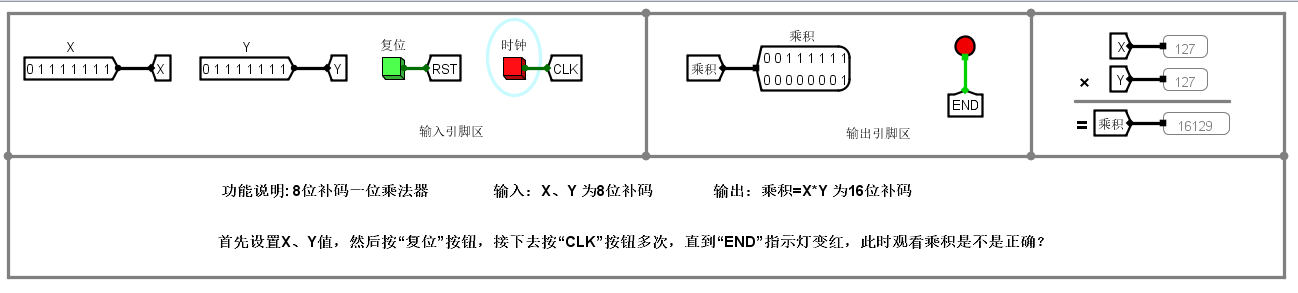
X=非0，Y=0 ，P是否=0？



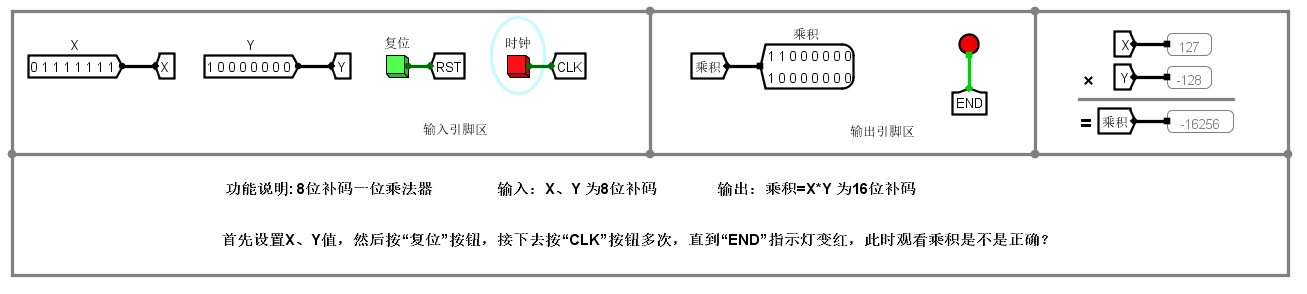
X=-128，Y=-128，P是否=16384？是



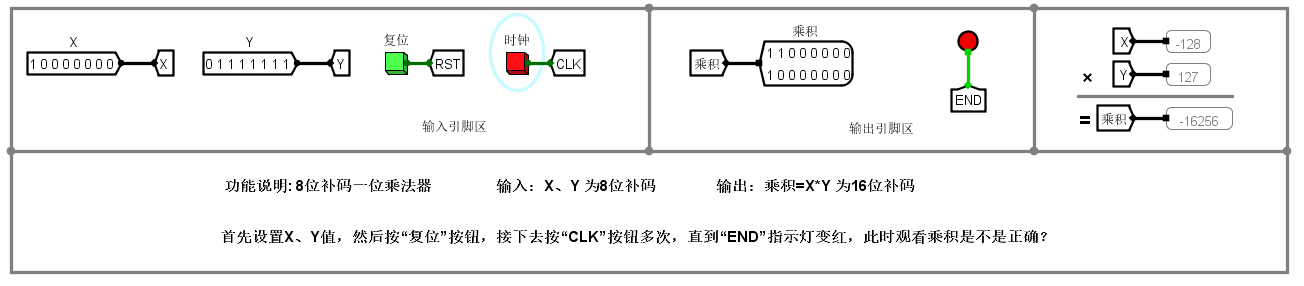
X=127，Y=127，P是否=16129？是



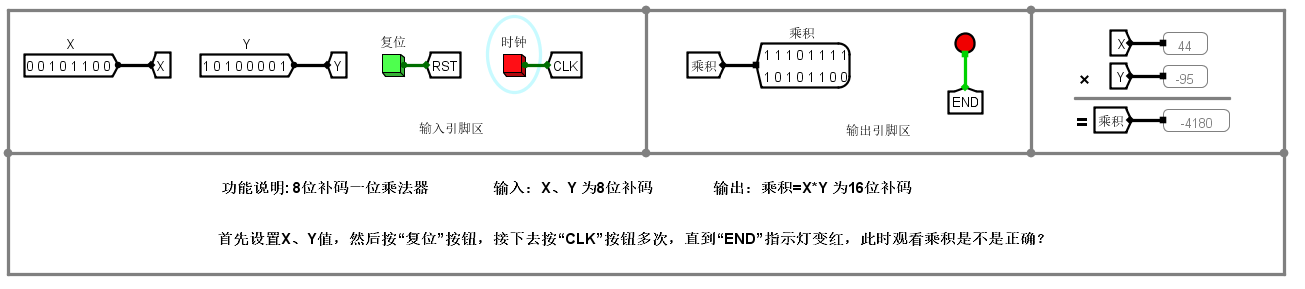
X=127，Y==-128，P是否=-16256？是



X=-128，Y==127，P是否=-16256？是



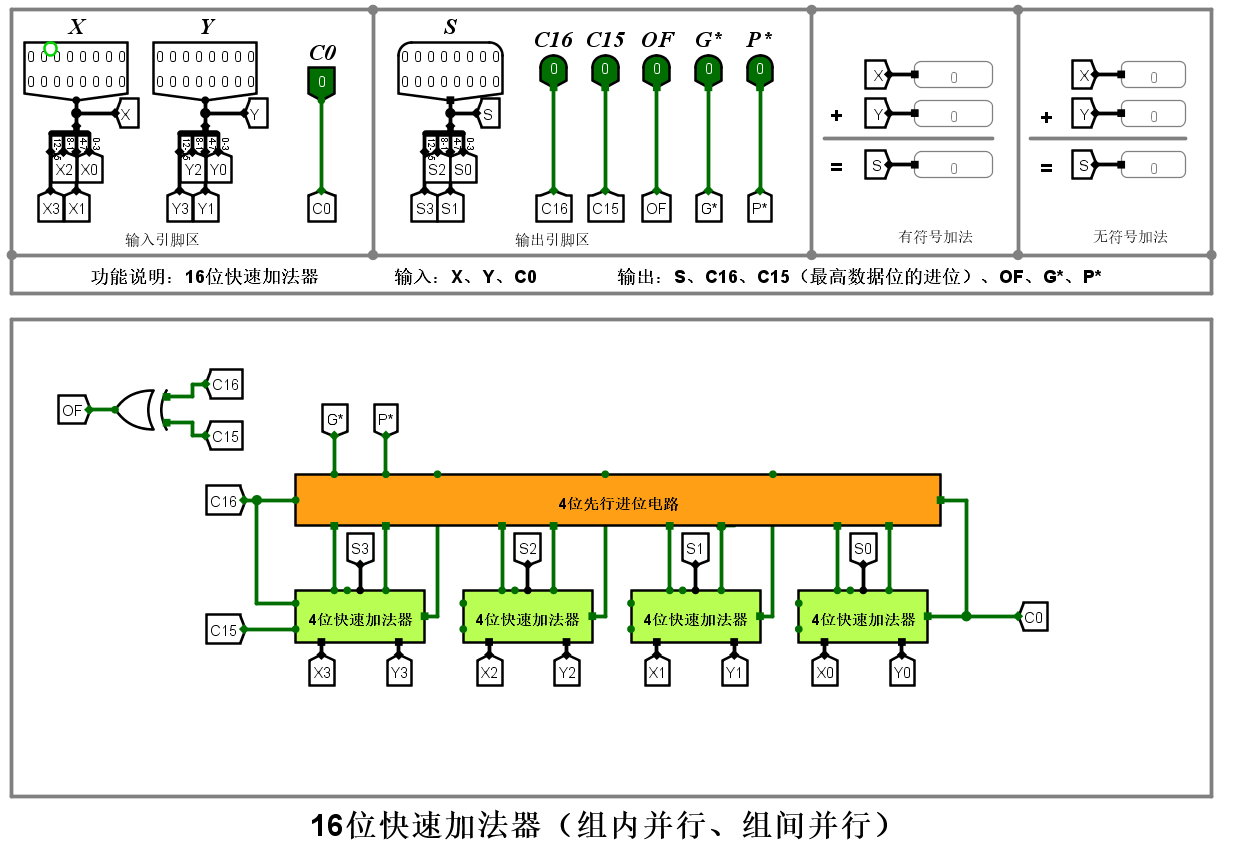
X=其他，Y=其他，P是否正确？是



* 1. **课后完成部分（设计实验的内容）**

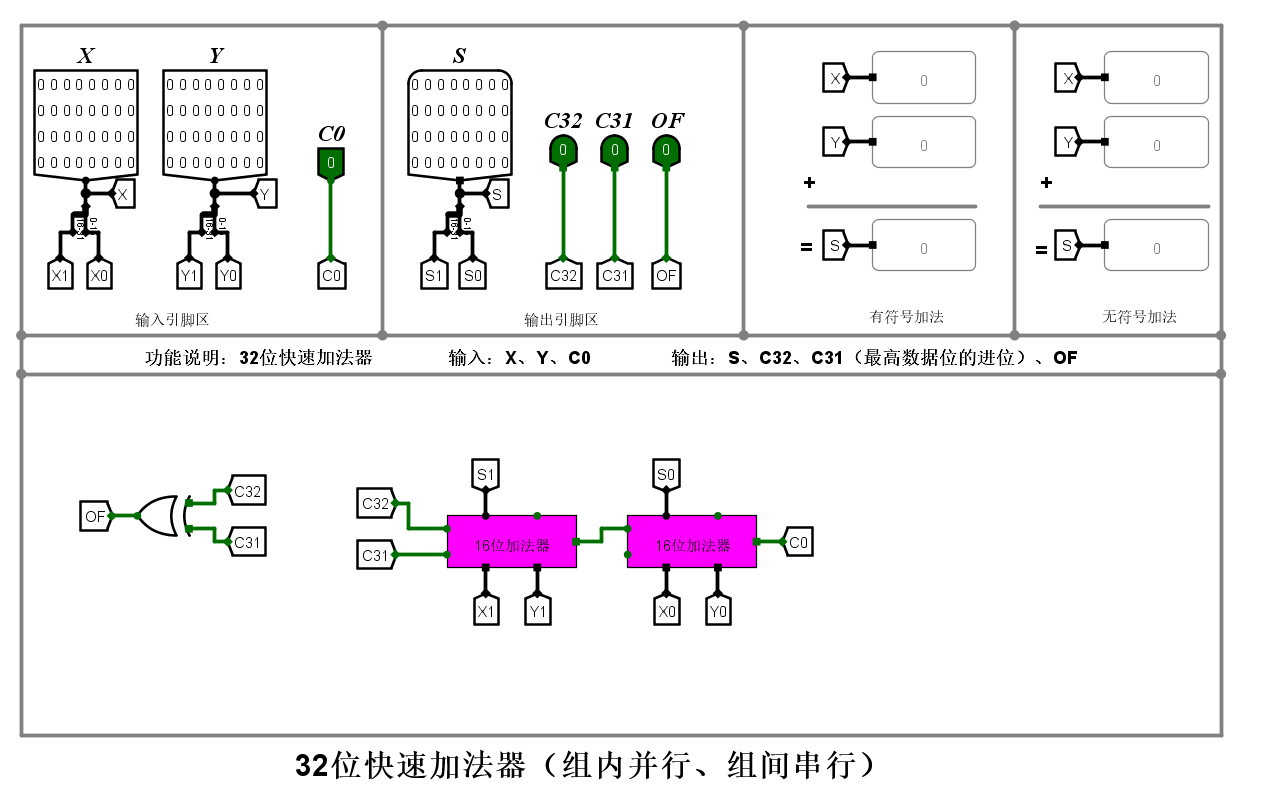
1. **16位快速加法器（组内并行、组间并行）（设计实验）——文件名： 16位快速加法器（组内并行、组间并行）.circ**

**主要实现思路：将四个四位快速加法器并联，四位快速加法器向四位先行进位电路提供进位生成传递函数P,G，由四位先行进位电路计算并获得每个加法器的C0并输入，最后将C15与C16异或得到溢出标志OF。**

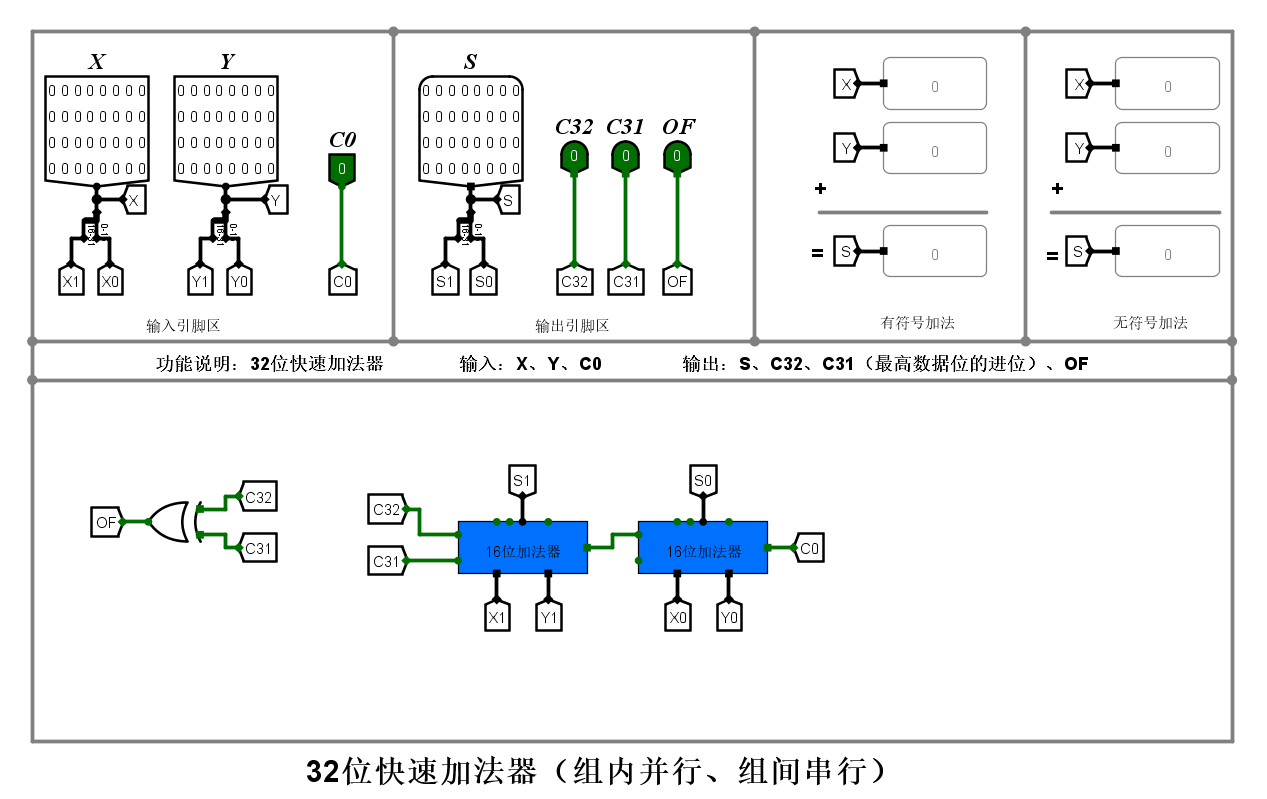


1. **32位快速加法器（组内并行、组间串行）（设计实验）——文件名： 32位快速加法器（组内并行、组间串行）.circ**

**主要实现思路：将两个16位快速加法器串联，将C31与C32异或得到OF溢出标志即可**

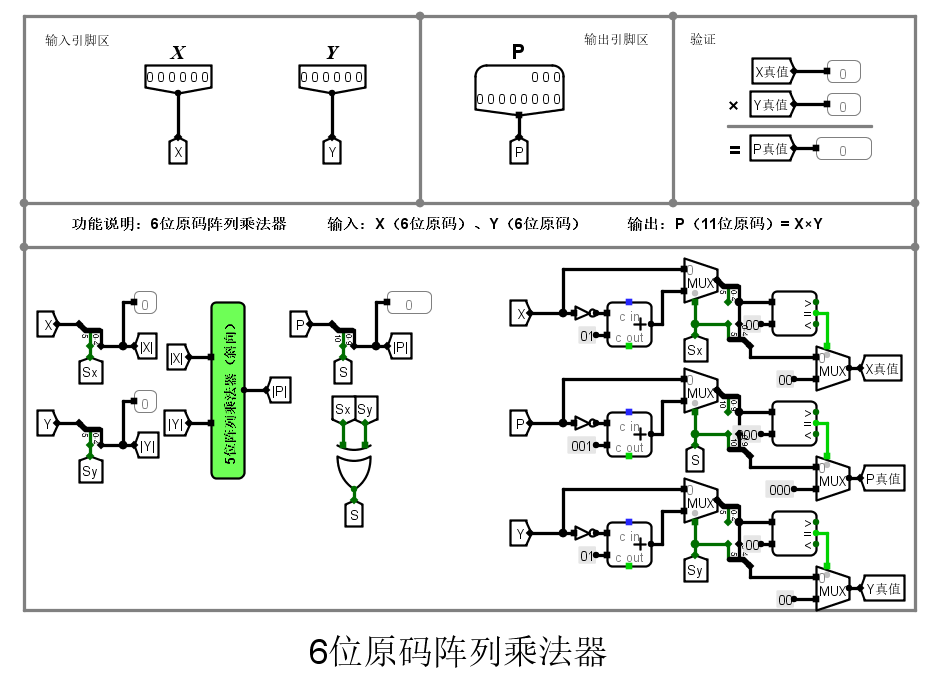


**主要实现思路：将上个电路的16位加法器换为组内并行组间并行的即可**



1. **6位原码阵列乘法器（设计实验）——文件名：6位原码阵列乘法器.circ**

**主要实现思路：由左到右分别是：取X,Y的符号位并将剩余5位作为|X|,|Y|；将|X|,|Y|输入5位阵列乘法器计算得到P的数据位|P|，将X,Y的符号位Sx,Sy异或得到P的符号位S，将|P|与其符号位|S|结合得到P，右侧电路分别是将X,Y,P转化为真值的电路。**



1. **6位补码阵列乘法器（设计实验）——文件名：6位补码阵列乘法器.circ**

**主要实现思路：将情况分为5种：**

**X!=0,Y!=0,X!=-32,Y!=-32的一般情况：该情况与上个电路类似，增加一个补码与真值的转化即可。**

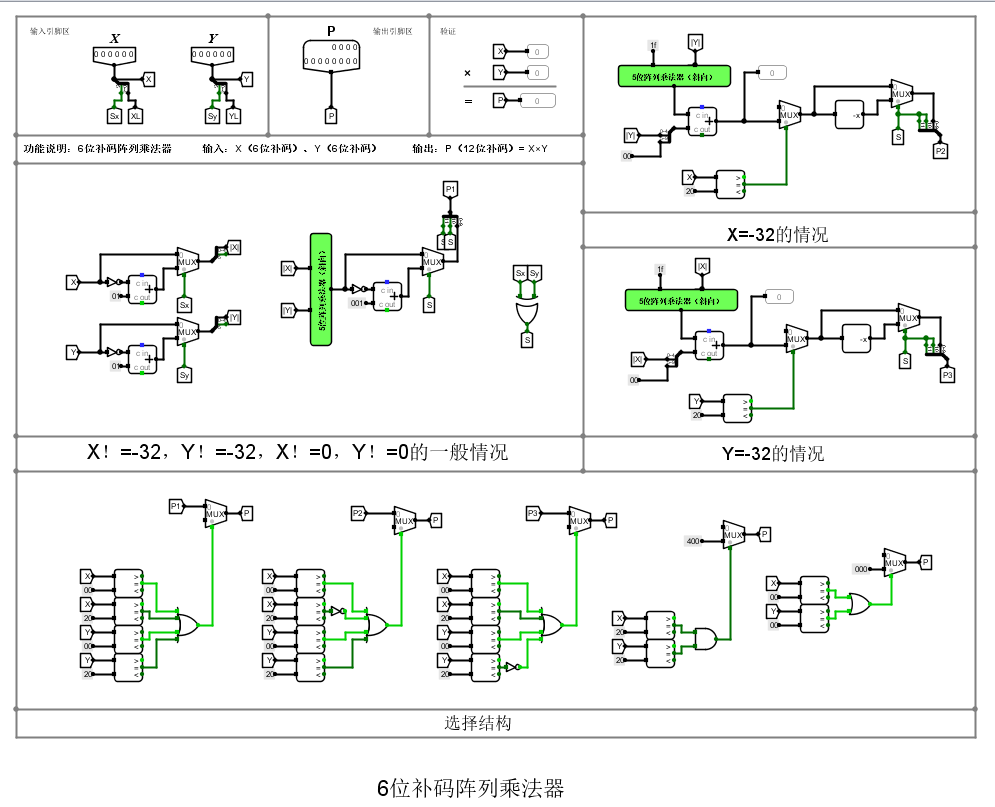
**X=-32,Y!=-32,X!=0,Y!=0的情况：直接使|Y|与31相乘再加上|Y|即可实现|Y|\*32，加上P的符号位S即可。**

**X!=-32,Y=-32,X!=0,Y!=0的情况：与上情况类似。**

**X!=-32,Y!=-32,X=0或Y=0的情况：直接令P=0即可。**

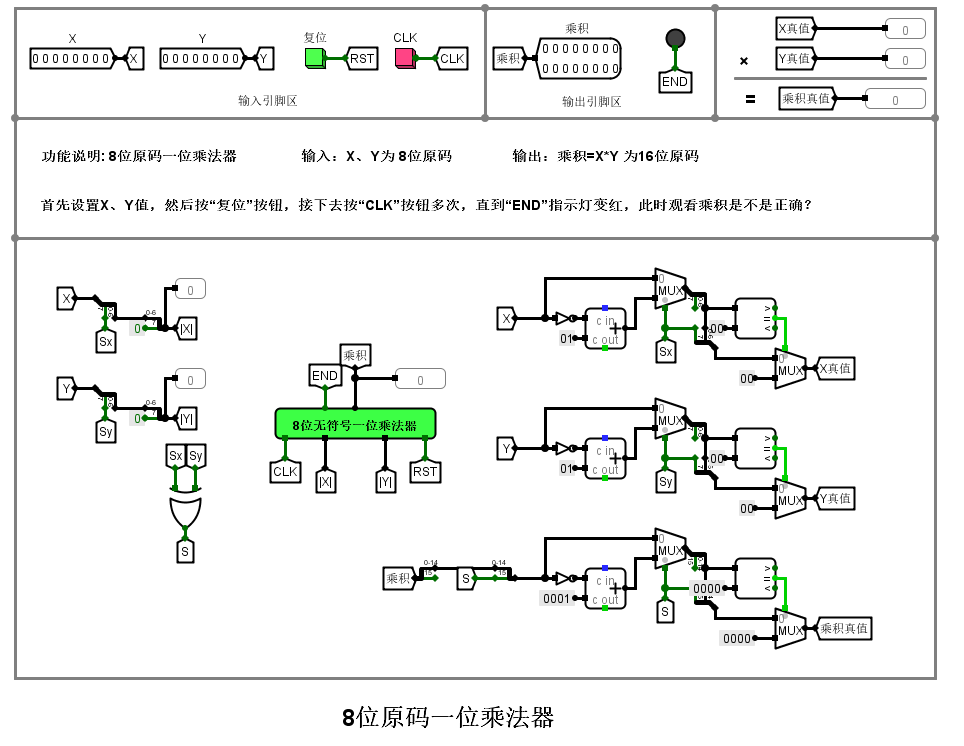
**X=-32,Y=-32的情况：直接令P=1024即可。**

**将上述5种情况通过下方的选择电路实现不同情况加载不同电路（具体通过比较器与MUX实现）。**



1. **8位原码一位乘法器（设计实验）——文件名：8位原码一位乘法器.circ**

**主要实现思路：取得X,Y的后7位并将首位补0得到|X|,|Y|，|X|,|Y|通过8位无符号一位乘法器计算得到乘积，X,Y的符号位异或得到乘积符号位，右侧电路将X,Y，乘积转化为真值。**



1. **8位补码一位乘法器（采用8位无符号一位乘法器实现）（挑战性实验）——文件名：8位补码一位乘法器（采用8位无符号一位乘法器实现）.circ**
2. **请将上述设计实验的运行结果，以屏幕拷贝形式保存，黏贴在实验报告中，并给出相关的文字说明。**