



**《计算机组成原理》**

**课程实验报告**

姓名： 苏一涵

学院：信息学院

系：软件工程系

专业： 软件工程

学号：36720232204041

2025年3月12日

**第2次实验 运算方法与运算器**

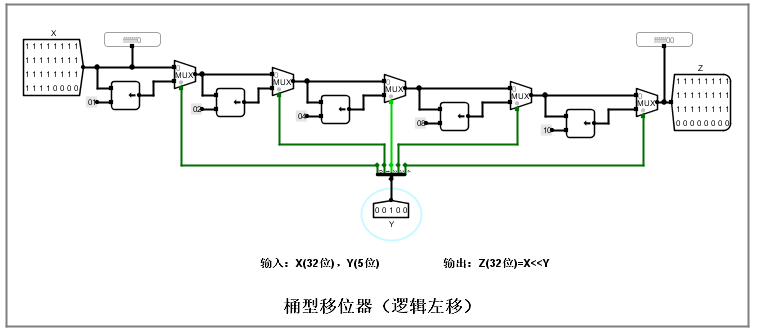
1. **实验环境**

**Windows系统下运行Logisim软件（需安装JDK）。**

1. **实验目的**
2. **验证桶型移位器、4位快速加法器、16位快速加法器（组内并行、组间串行）、5位无符号阵列乘法器（斜向、横向）、8位无符号一位乘法器、8位补码一位乘法器电路。**
3. **设计16位快速加法器（组内并行、组间并行）、32位快速加法器（组内并行、组间串行）、6位原码阵列乘法器、6位补码阵列乘法器、8位原码一位乘法器、8位补码一位乘法器（采用8位无符号一位乘法器实现）。**
4. **实验内容**
   1. **验证实验**
5. **桶型移位器：通过改变输入变量X和Y的不同取值，观察输出Z的变化以及Y是如何控制移位器的移动次数的。（结合具体实例分析截屏佐证）**

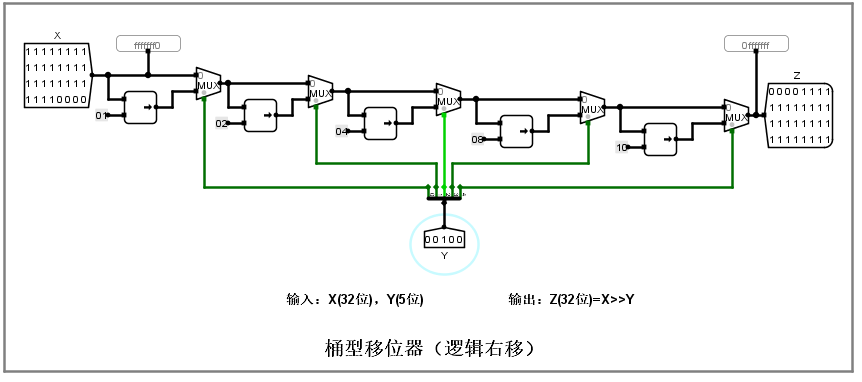
1.逻辑左移：Z=X<<Y

输入：X=FFFFFFF0，Y=00100输出：Z=FFFFFF00



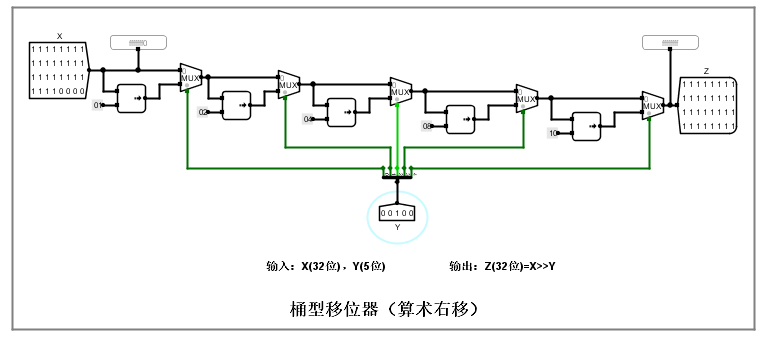
2.逻辑右移：Z=X>>Y

输入：X=FFFFFFF0，Y=00100输出：Z=0FFFFFFF

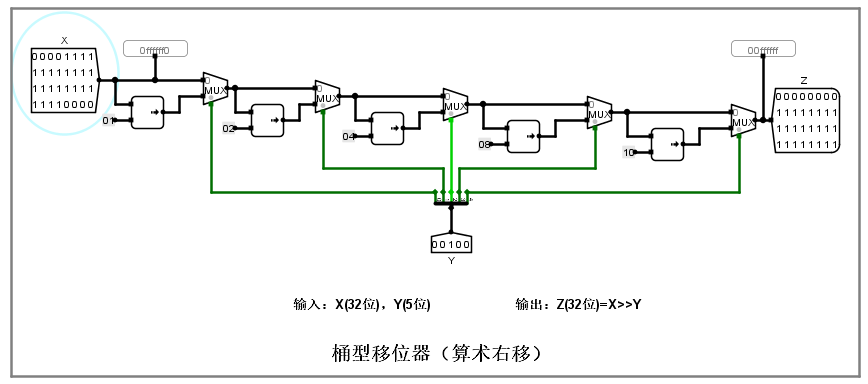


3．算术右移：Z=X>>Y

输入：X=FFFFFFF0，Y=00100输出：Z=FFFFFFFF（符号位保持不变）



输入：X=0FFFFFF0，Y=00100输出：Z=00FFFFFF（符号位保持不变）



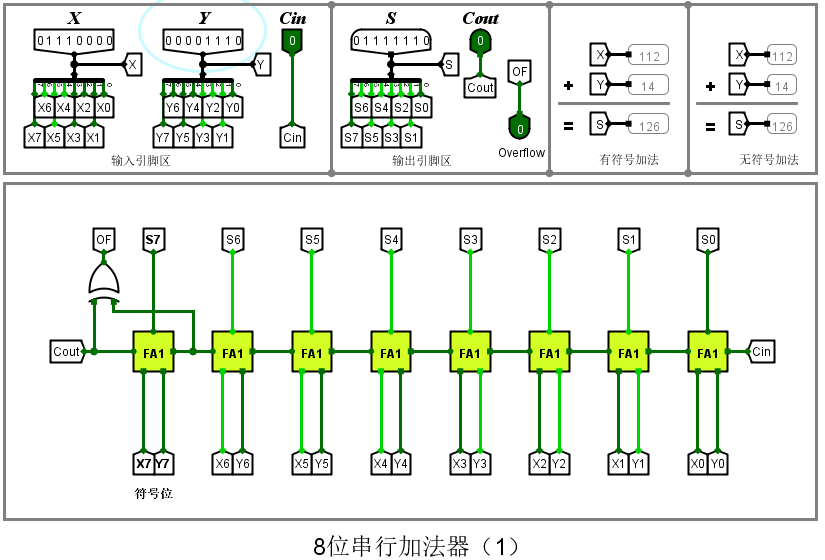
1. **8位串行加法器/8位串行可控加减法器**

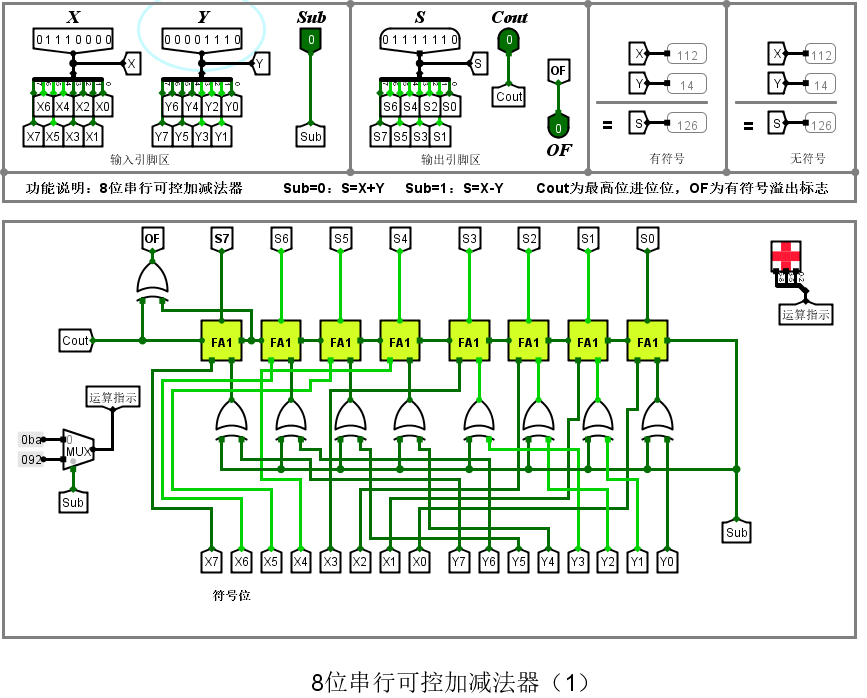
**具体要求：**

* **通过设置不同的X、 Y、 Cin值，对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）：**

**① 正数+正数=正数（不溢出）。**

**X=01110000,Y=00001110,CIN=0，结果如下：**

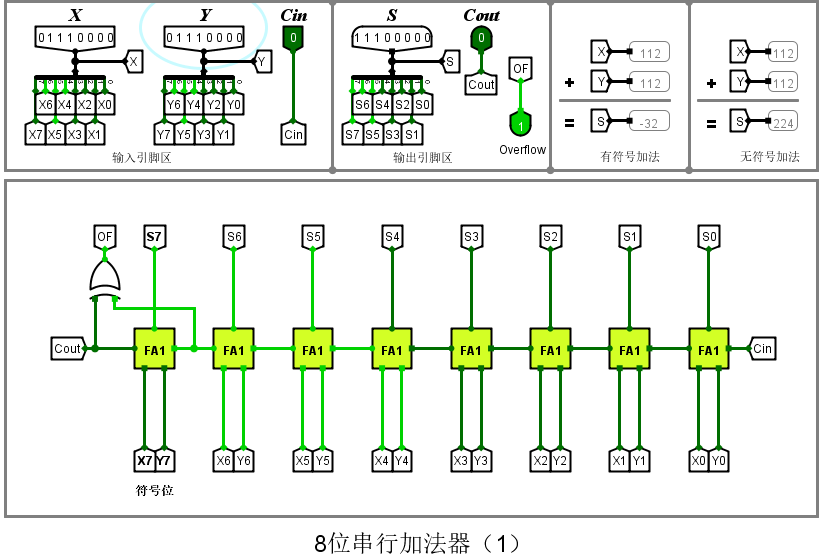
****

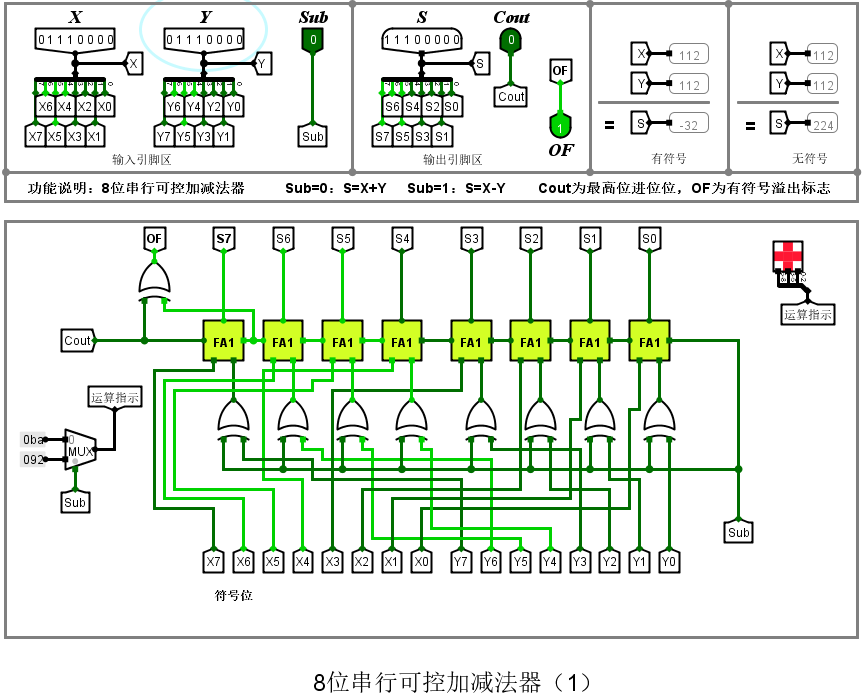
****

**S=01111110没有溢出。**

**② 正数+正数=负数（溢出）。**

**X=01110000,Y=01110000,CIN=0,结果如下图：**

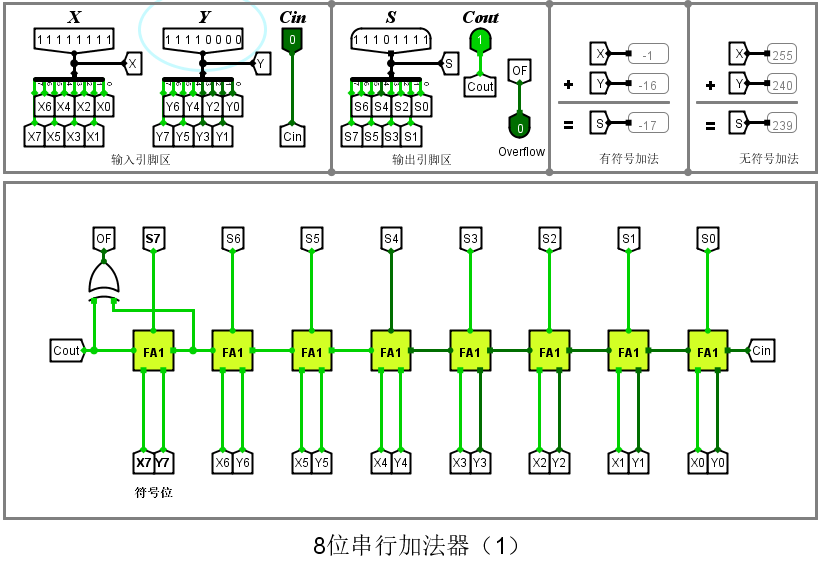
****

****

**S=11100000，符号位变为1，发生有符号溢出**

**③ 负数+负数=负数（不溢出）。**

**X=11111111,Y=11110000,CIN=0,结果如下：**

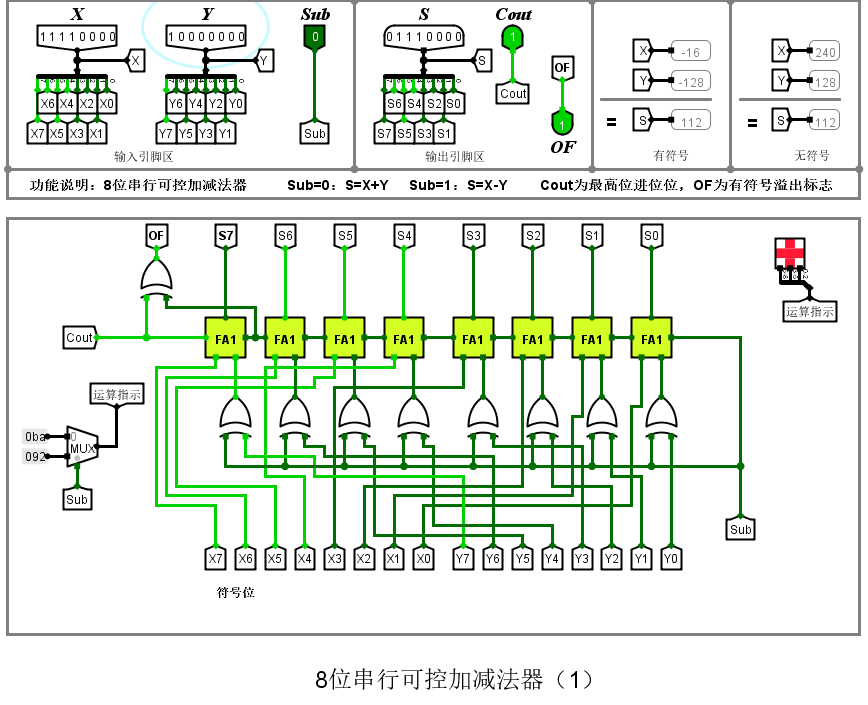
****

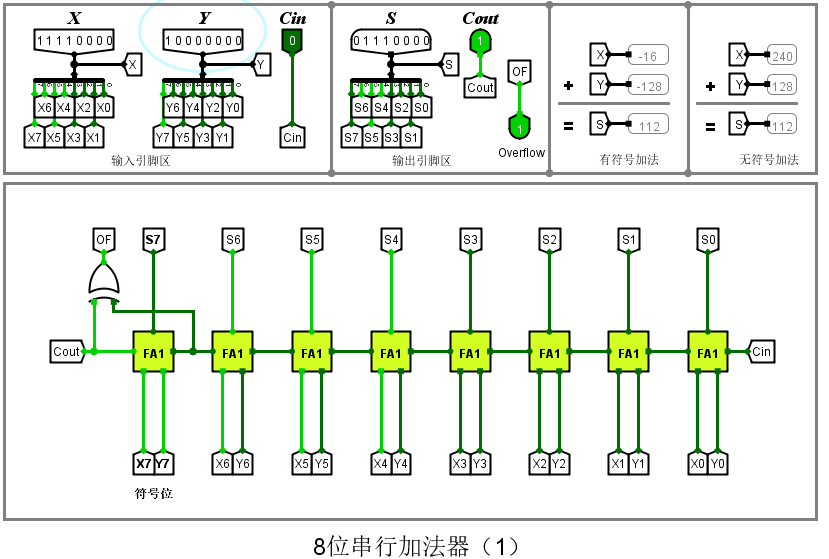
****

**S=11101111，没有溢出**

**④ 负数+负数=正数（溢出）。**

**X=11110000,Y=10000000,CIN=0结果如下：**

****

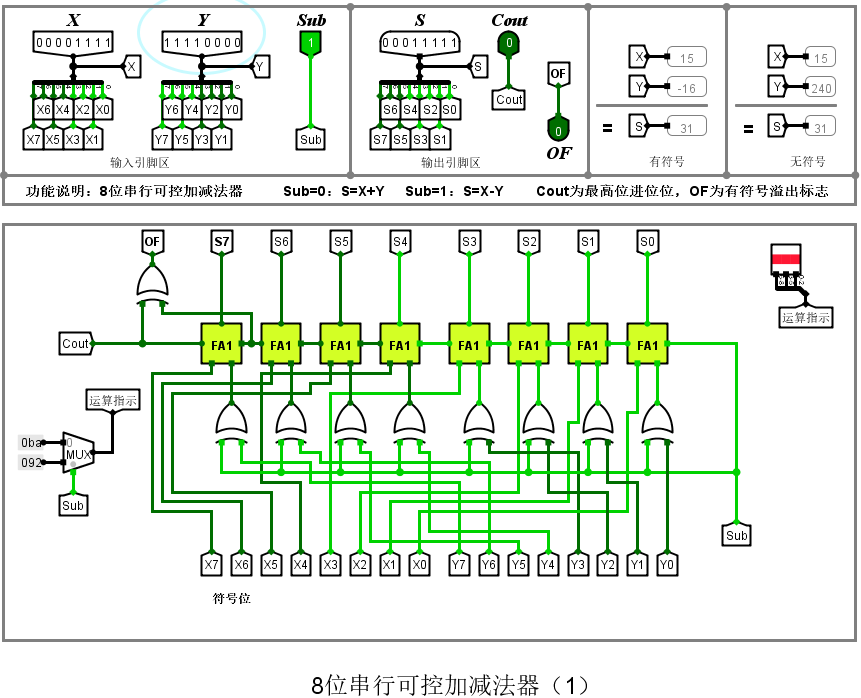
****

**S=01110000，发生溢出**

* **通过设置不同的X、 Y值，对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）：**

**① 正数-负数=正数（不溢出）。**

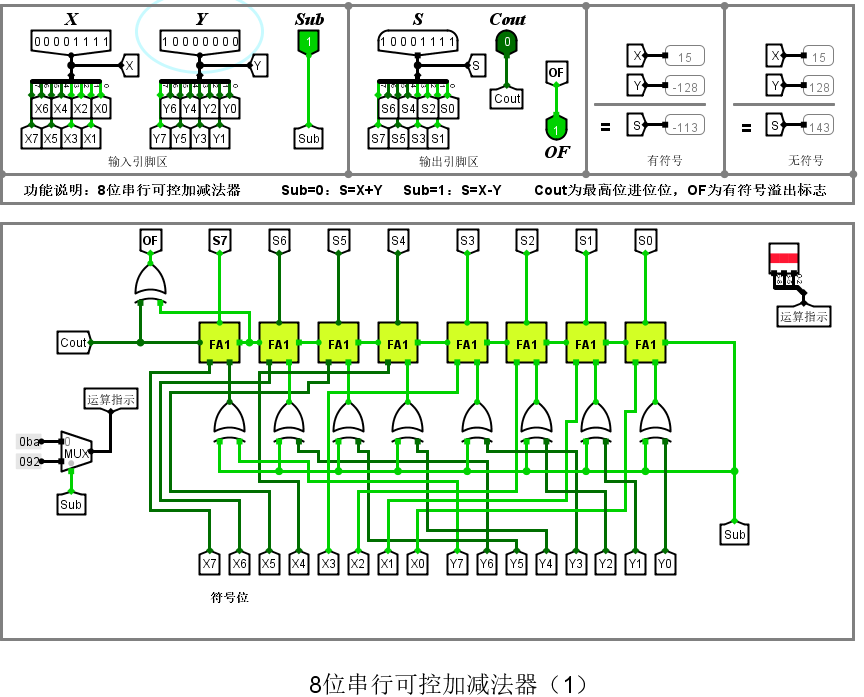
**X=00001111,Y=11110000,SUB=1，结果如下：**

****

**S=00011111，没有溢出**

**② 正数-负数=负数（溢出）。**

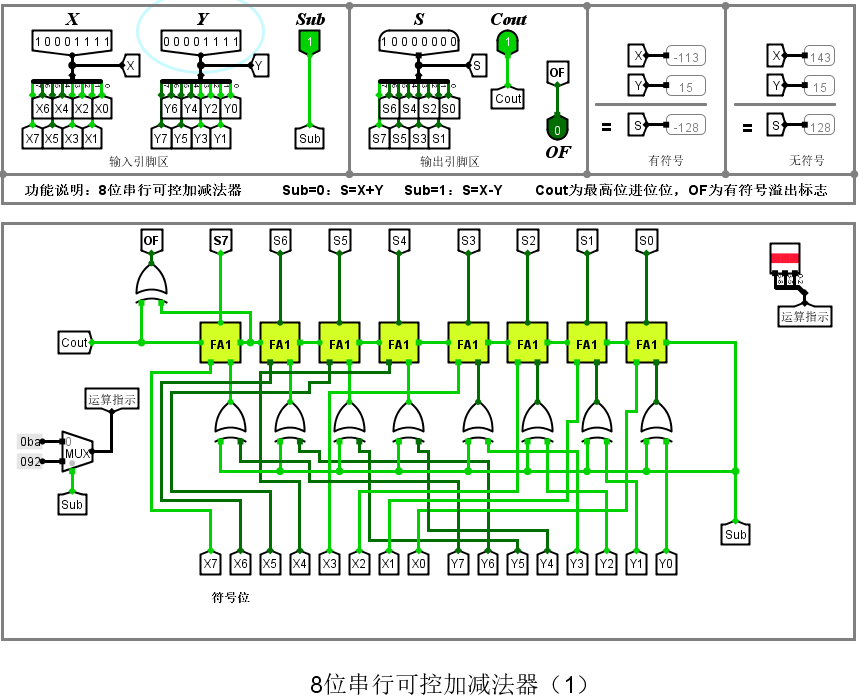
**X=00001111,Y=10000000,SUB=1，结果如下：**

****

**S=10001111，符号位变了，发生溢出**

**③ 负数-正数=负数（不溢出）。**

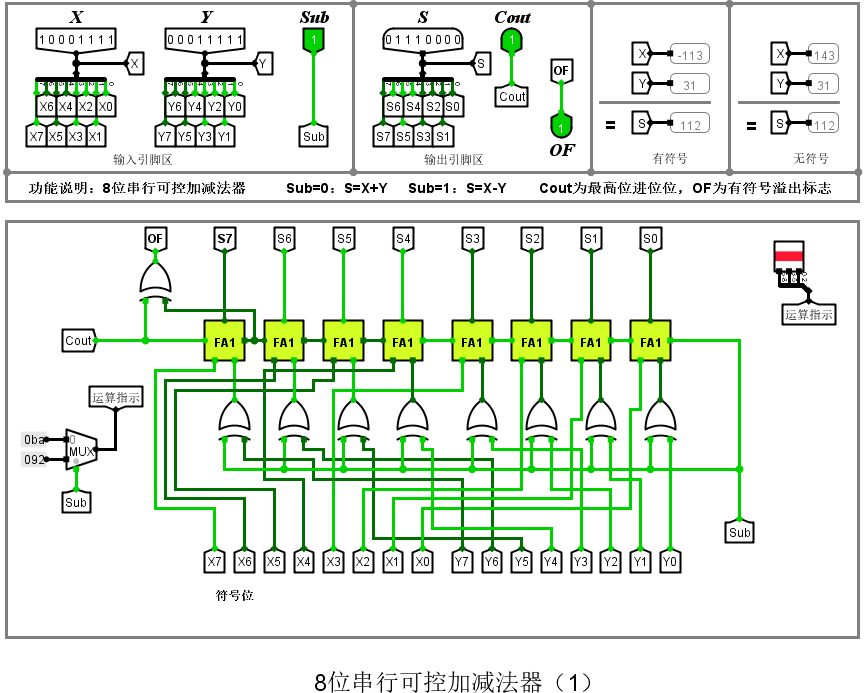
**X=10001111,Y=00001111,SUB=1，结果如下：**

****

**S=10000000，没有溢出**

**④ 负数-正数=正数（溢出）。**

**X=10001111,Y=00011111,SUB=1，结果如下：**

****

**S=01110000，发生溢出**

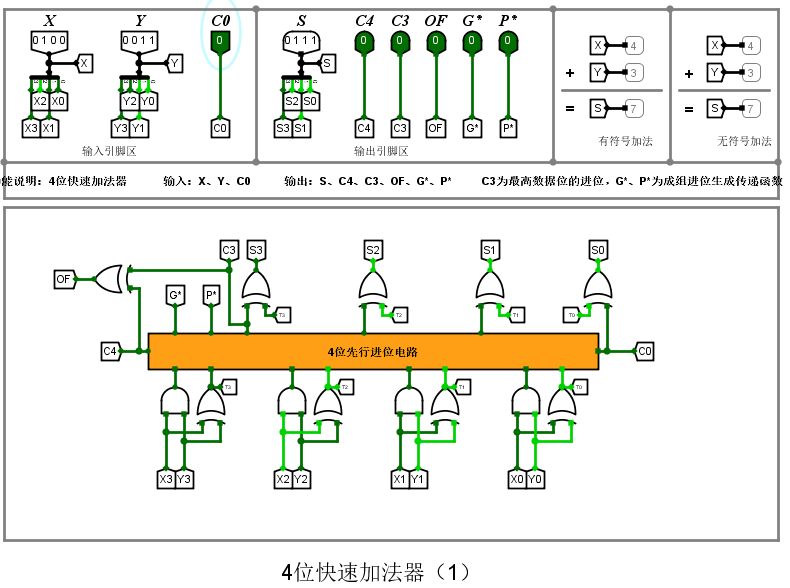
1. **4位快速加法器/16位快速加法器（组内并行、组间串行）**

**具体要求：通过设置不同的X、 Y、 C0值，对4位快速加法器、 16位快速加法器进行验证，并对结果进行分析（以下4种情况）：**

**4位快速加法器验证情况如下：**

**① 正数+正数=正数（不溢出）。**

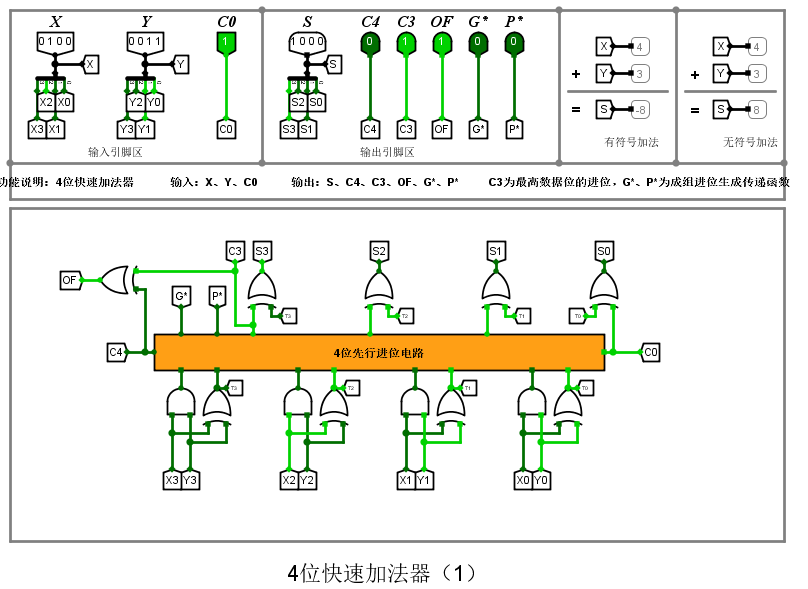
**X=0100,Y=0011,C0=0，结果如下：**

****

**S=0111，C4=0,C3=0,OF=0,G\*=0,P\*=0,没有溢出**

**② 正数+正数=负数（溢出）。**

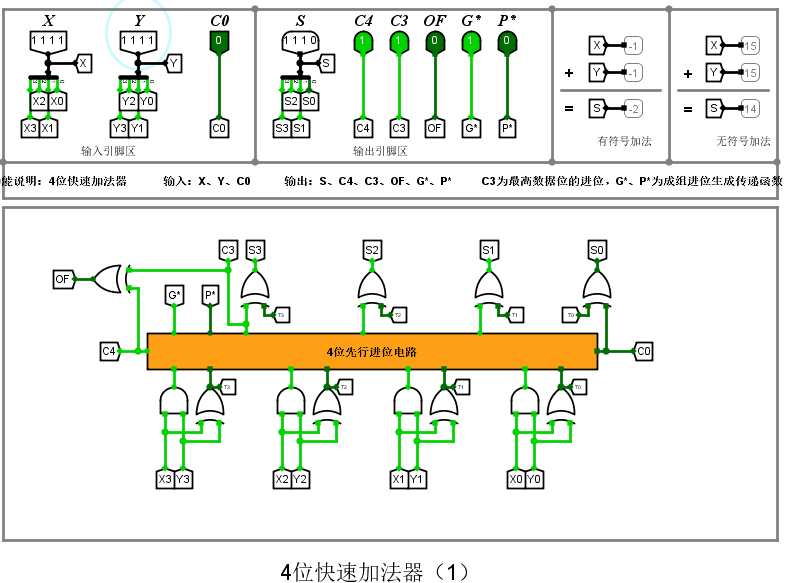
**X=0100，Y=0011，C0=1，结果如下：**

****

**S=1000，C4=0,C3=1,OF=1,G\*=0,P\*=0，溢出**

**③ 负数+负数=负数（不溢出）。**

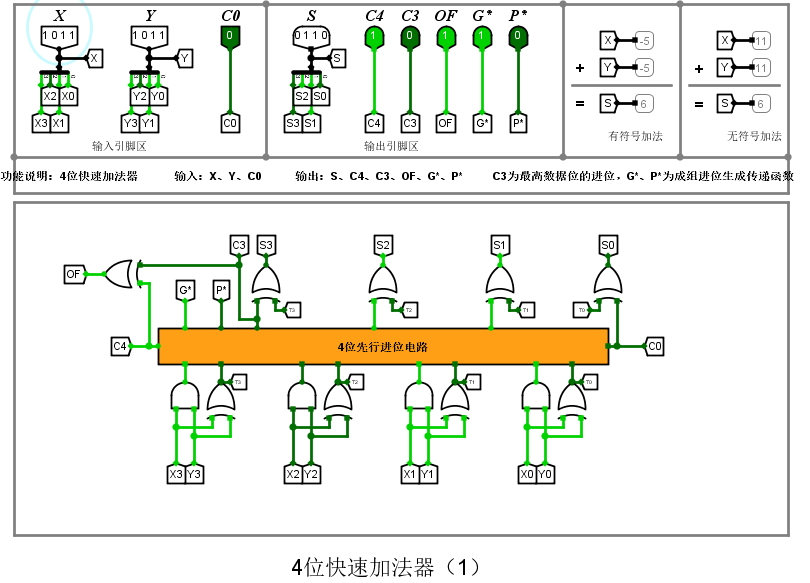
**X=1111,Y=1111,C0=0,结果如下：**

****

**S=1110，C4=1,C3=1,OF=0,G\*=1,P\*=0，没有溢出**

**④ 负数+负数=正数（溢出）。**

**X=1011,Y=1011,CIN=0,结果如下：**

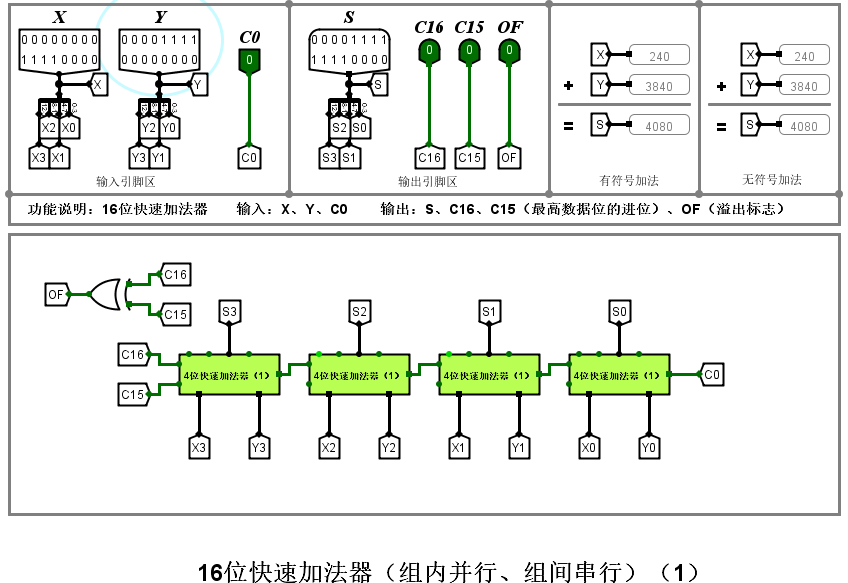
****

**S=0110，C4=1,C3=0,OF=1,G\*=1,P\*=0，发生溢出**

**16位快速加法器验证情况如下：**

**① 正数+正数=正数（不溢出）。**

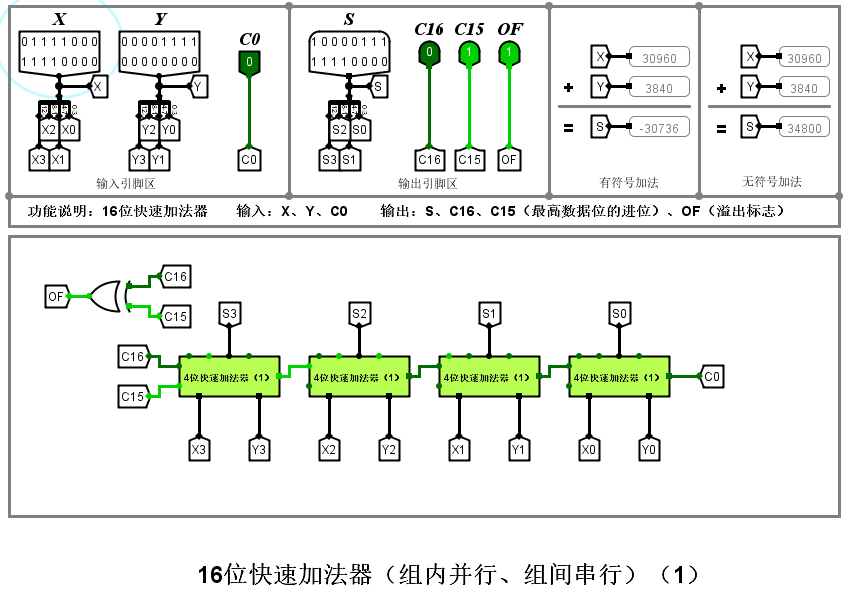
**X=0000000011110000,Y=0000111100000000,CIN=0**

****

**S=0000111111110000,OF=0,没有溢出**

**② 正数+正数=负数（溢出）。**

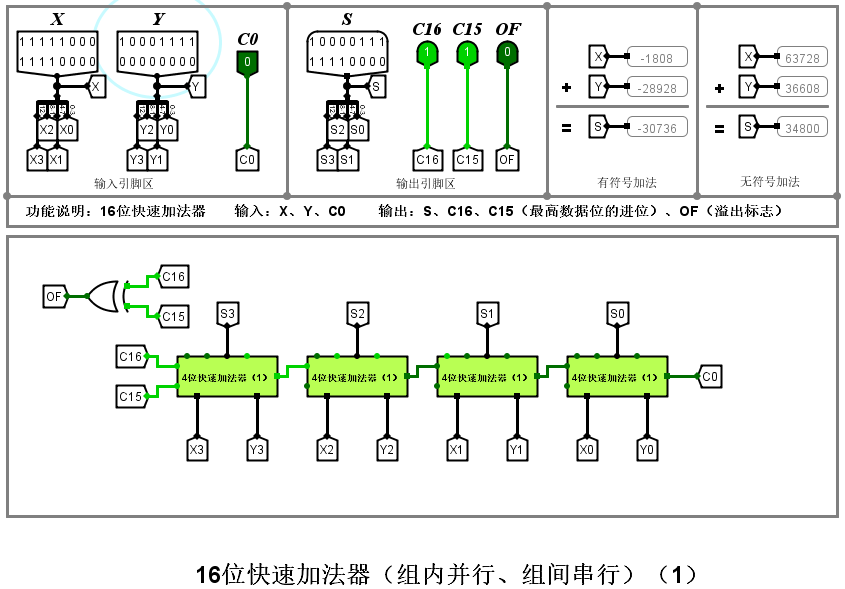
**X=0111100011110000,Y=0000111100000000,CIN=0**

****

**S=1000011111110000为负数,OF=1,发生溢出**

**③ 负数+负数=负数（不溢出）。**

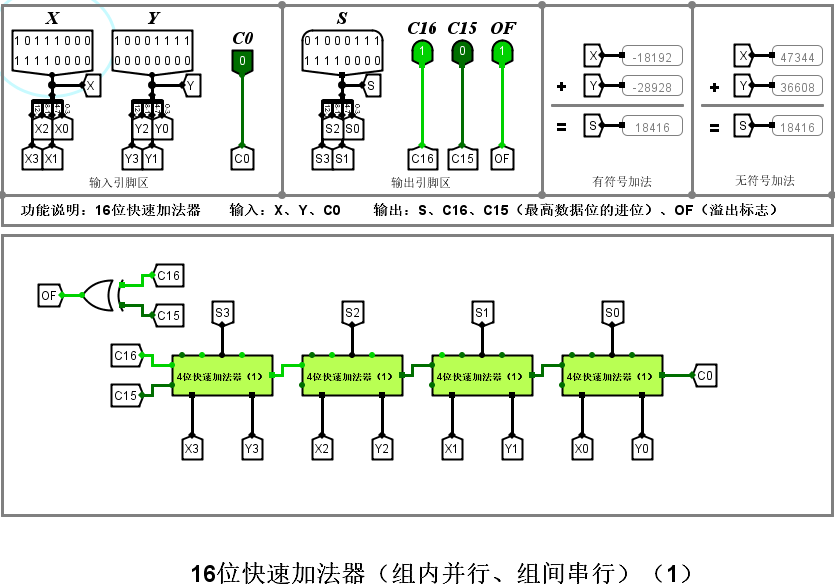
**X=1111100011110000,Y=1000111100000000,CIN=0**

****

**S=1000011111110000,OF=0,没有溢出**

**④ 负数+负数=正数（溢出）。**

**X=1011100011110000,Y=1000111100000000,CIN=0**

****

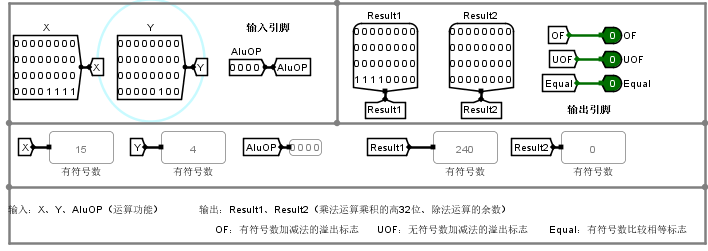
**S=0100011111110000,OF=1,发生溢出**

1. **算术逻辑单元（ALU）**

**具体要求：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：**

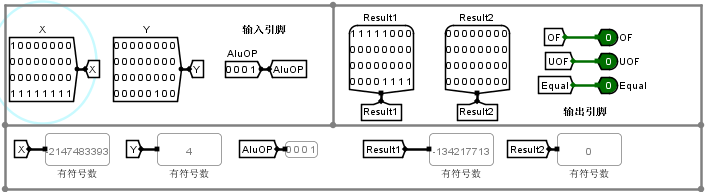
**① 逻辑左移**

**X=1111,Y=100,ALUOP=0000,即逻辑左移四位，结果如下：**

****

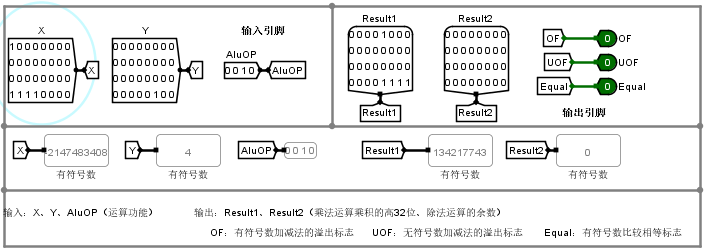
**② 算术右移**

**X=100……011111111,Y=100,ALUOP=0001,即算术右移四位，带符号位，结果如下：**

****

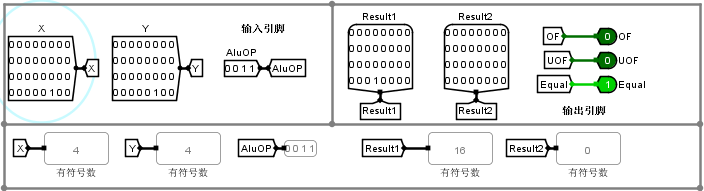
**③ 逻辑右移**

**X=100….11110000,Y=100,ALUOP=0010,即逻辑右移四位，不带符号位，结果如下：**

****

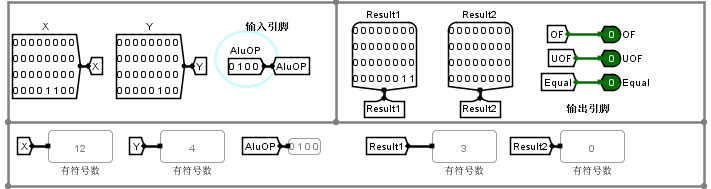
**④ 乘法运算**

**X=100,Y=100,ALUOP=0011,即乘法运算,结果如下：**

****

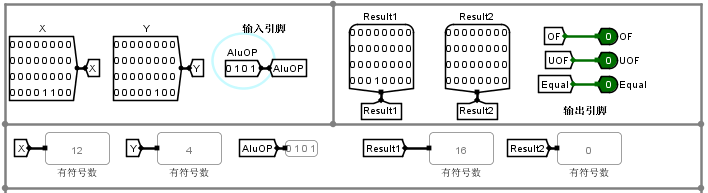
**⑤ 除法运算**

**X=1100,Y=100,ALUOP=0100,即除法运算,结果如下：**

****

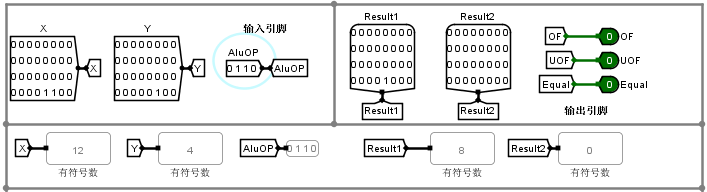
**⑥ 加法运算**

**X=1100,Y=100,ALUOP=0101,即加法运算,结果如下：**

****

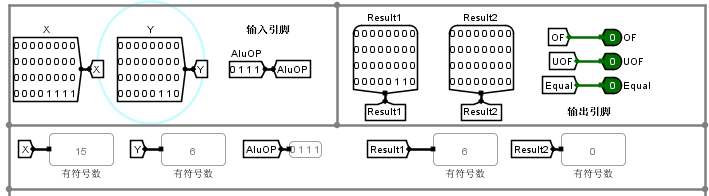
**⑦ 减法运算**

**X=1100,Y=100,ALUOP=0110,即减法运算,结果如下：**

****

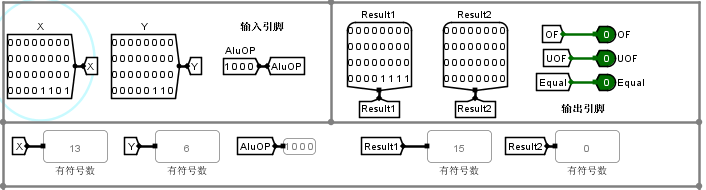
**⑧ 逻辑与运算**

**X=1111,Y=110,ALUOP=0111,即逻辑与运算,结果如下：**

****

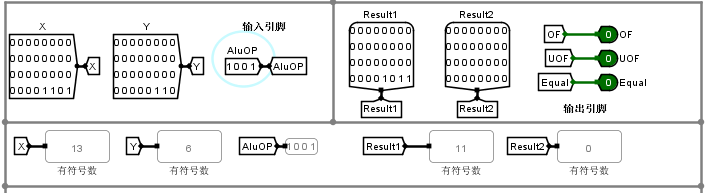
**⑨ 逻辑或运算**

**X=1101,Y=110,ALUOP=1000,即逻辑或运算,结果如下：**

****

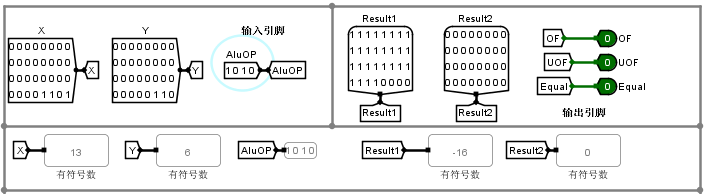
**⑩ 逻辑异或运算**

**X=1101,Y=110,ALUOP=1001,即逻辑异或运算,结果如下：**

****

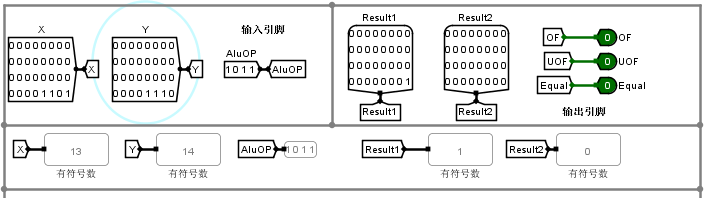
**⑪ 逻辑或非运算**

**X=1101,Y=110,ALUOP=1010,即逻辑或非运算,结果如下：**

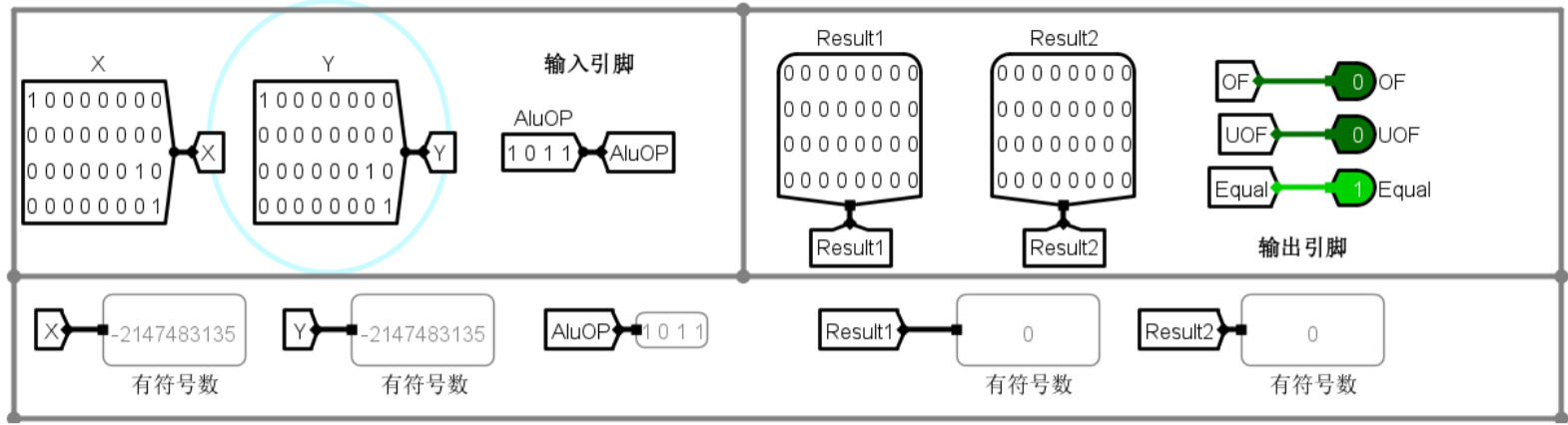
****

**⑫ 有符号数比较**

**X=1101,Y=1110,ALUOP=1011,即有符号数比较,** **X<Y 时，Result=1，否则 Result=0结果如下：**

****

**两数相等时，Equal=1，如图：**

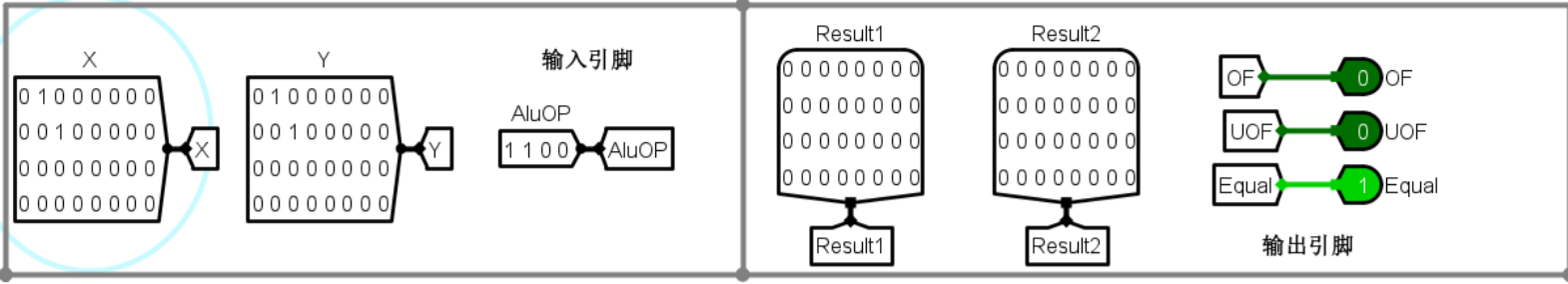


**⑬ 无符号数比较**

**X=100……01101,Y=1100……1110,ALUOP=1100,即无符号数比较,** **X<Y 时，Result=1，否则 Result=0结果如下：**

****

**两数相等时，Equal=1，结果如下：**

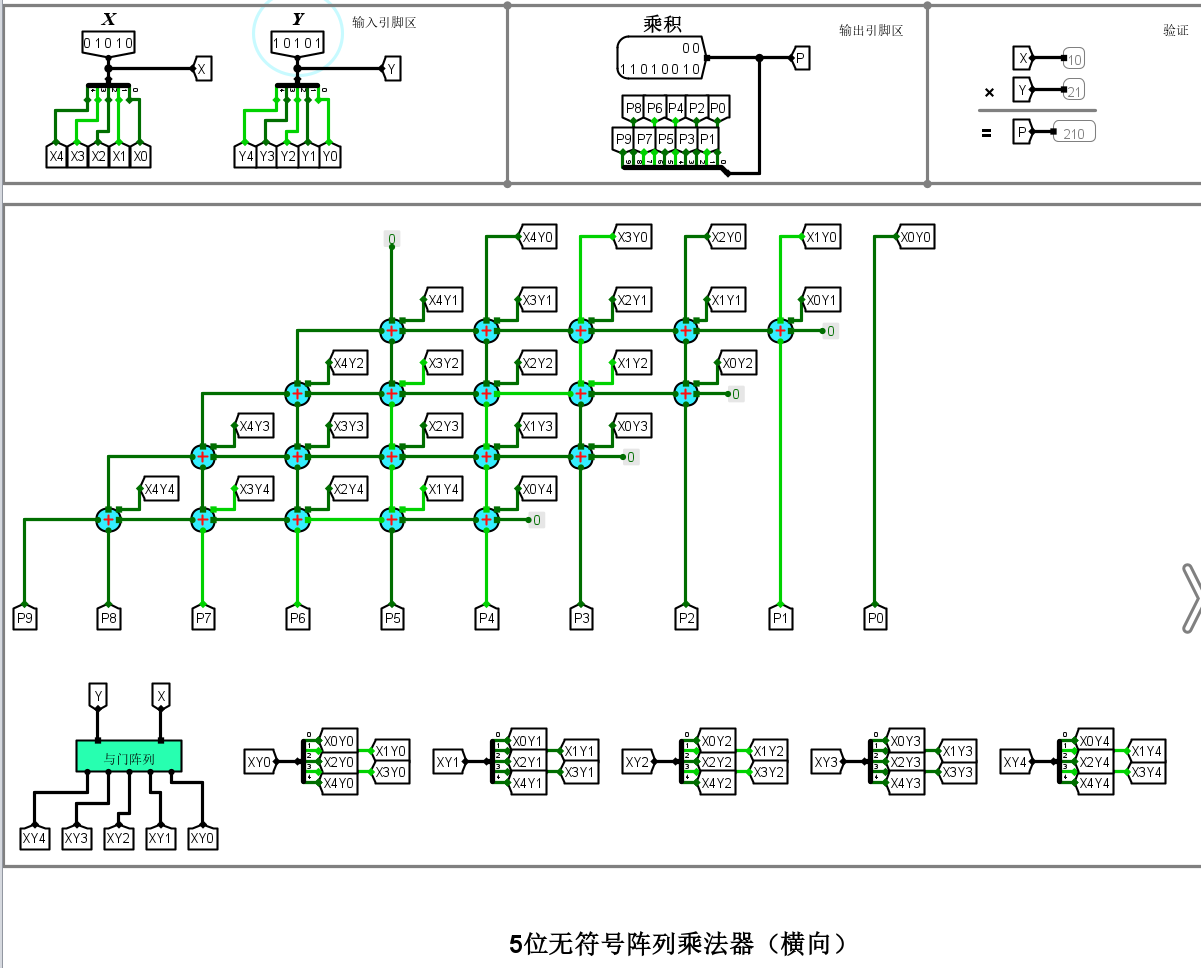


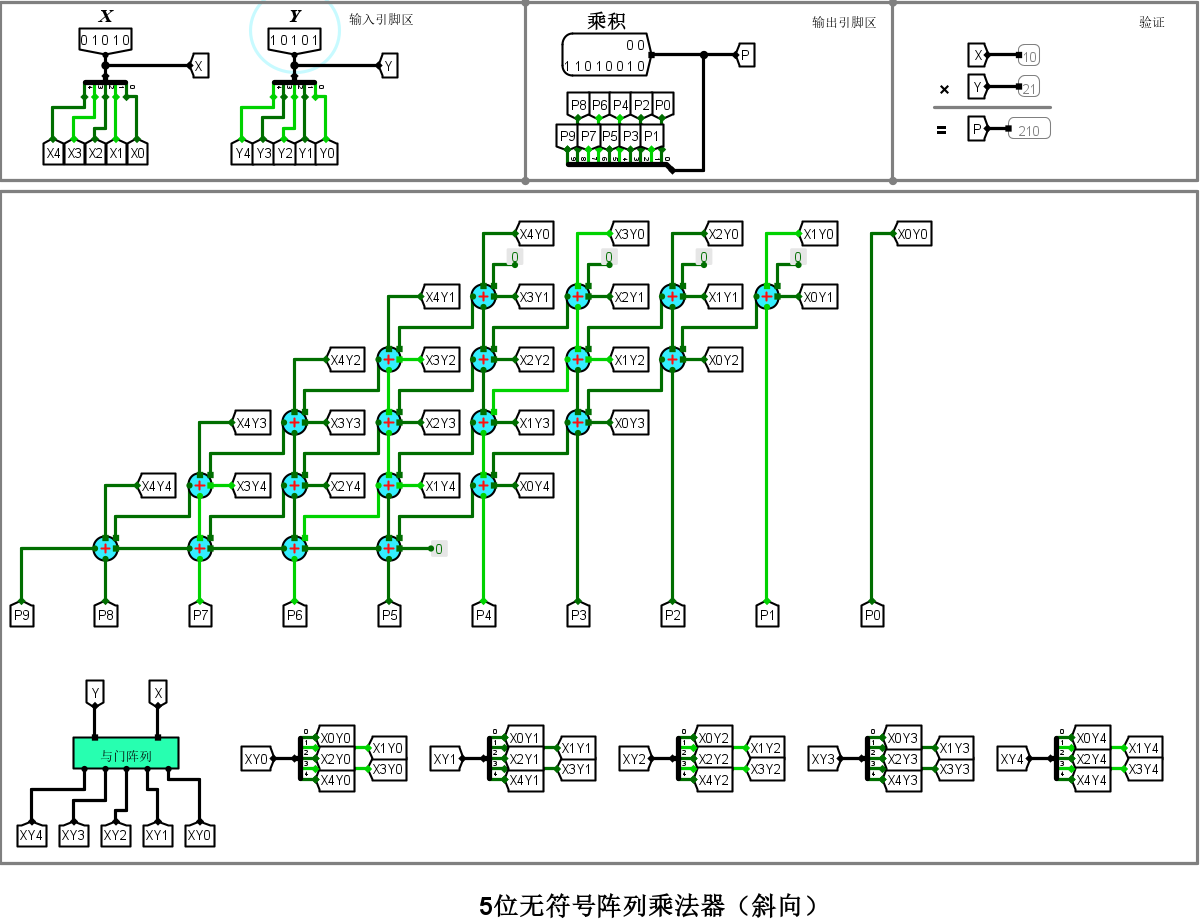
1. **5位无符号阵列乘法器（斜向/横向）**

**具体要求：按照下述要求，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：**

1. **X=非0， Y=非0， P=？**

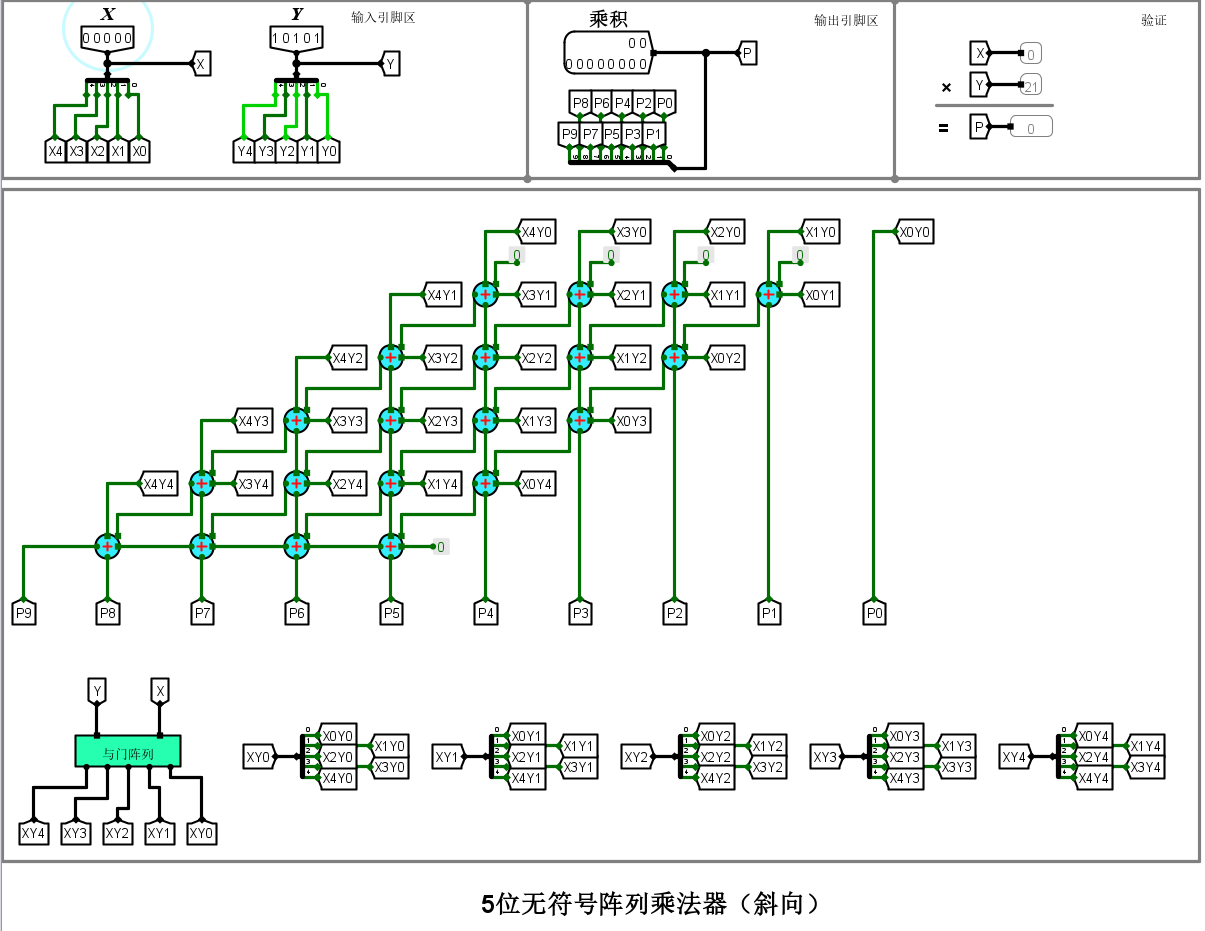
**X=1010，Y=10101，P=11010010**

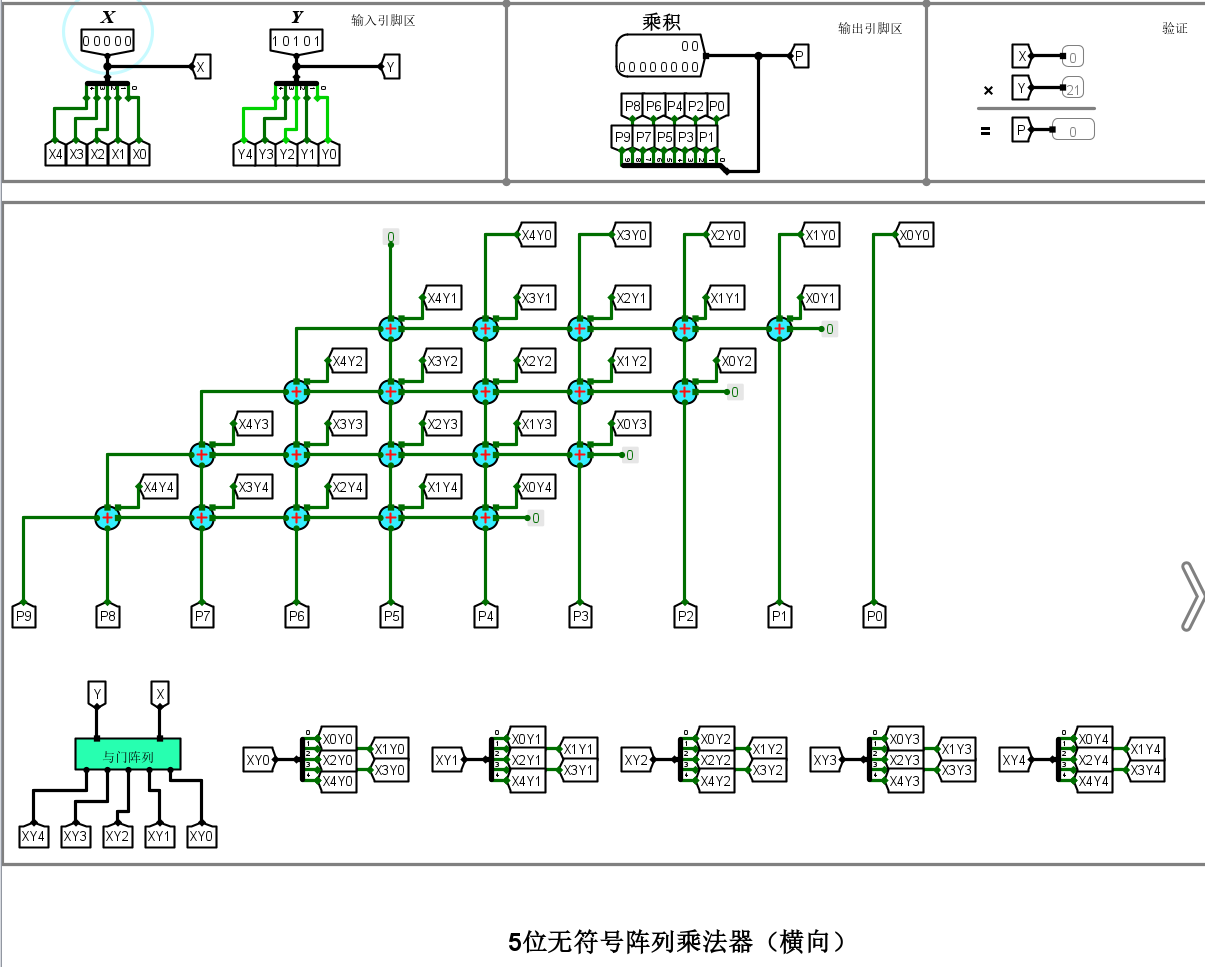




1. **X=0， Y=非0， P是不是=0？**

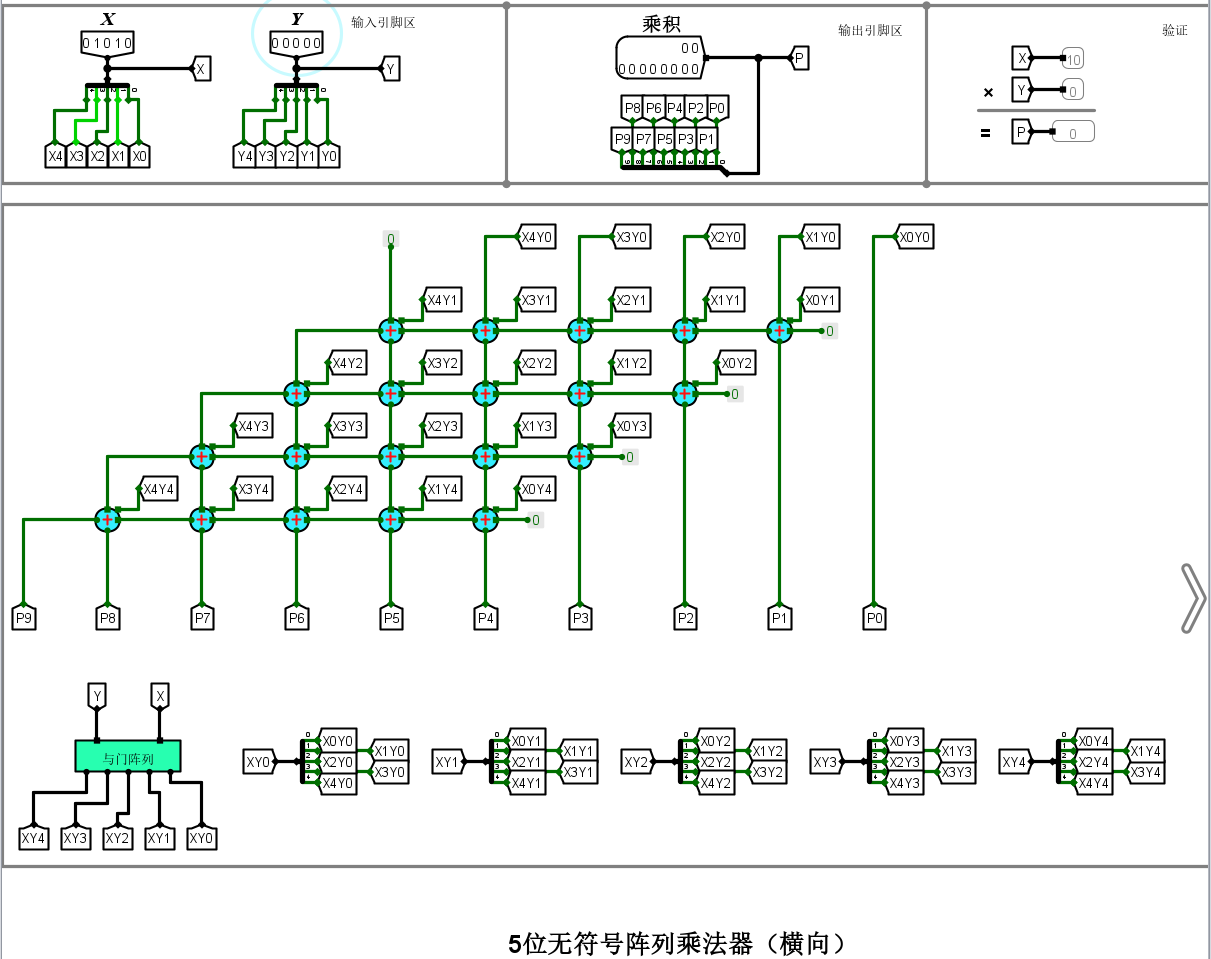
**X=0，Y=10101，P=0**





1. **X=非0， Y=0， P是不是=0？**

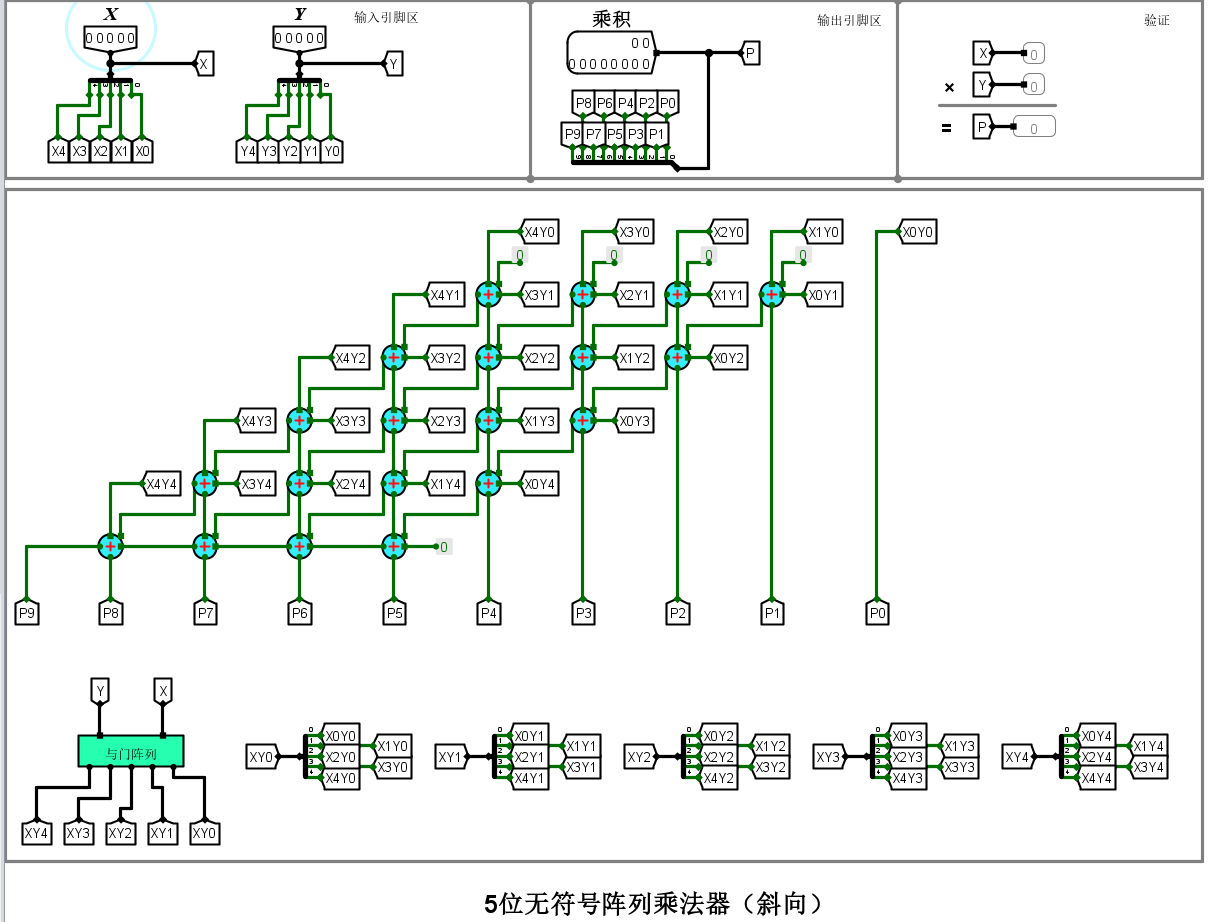
**X=1010，Y=0，P=0**

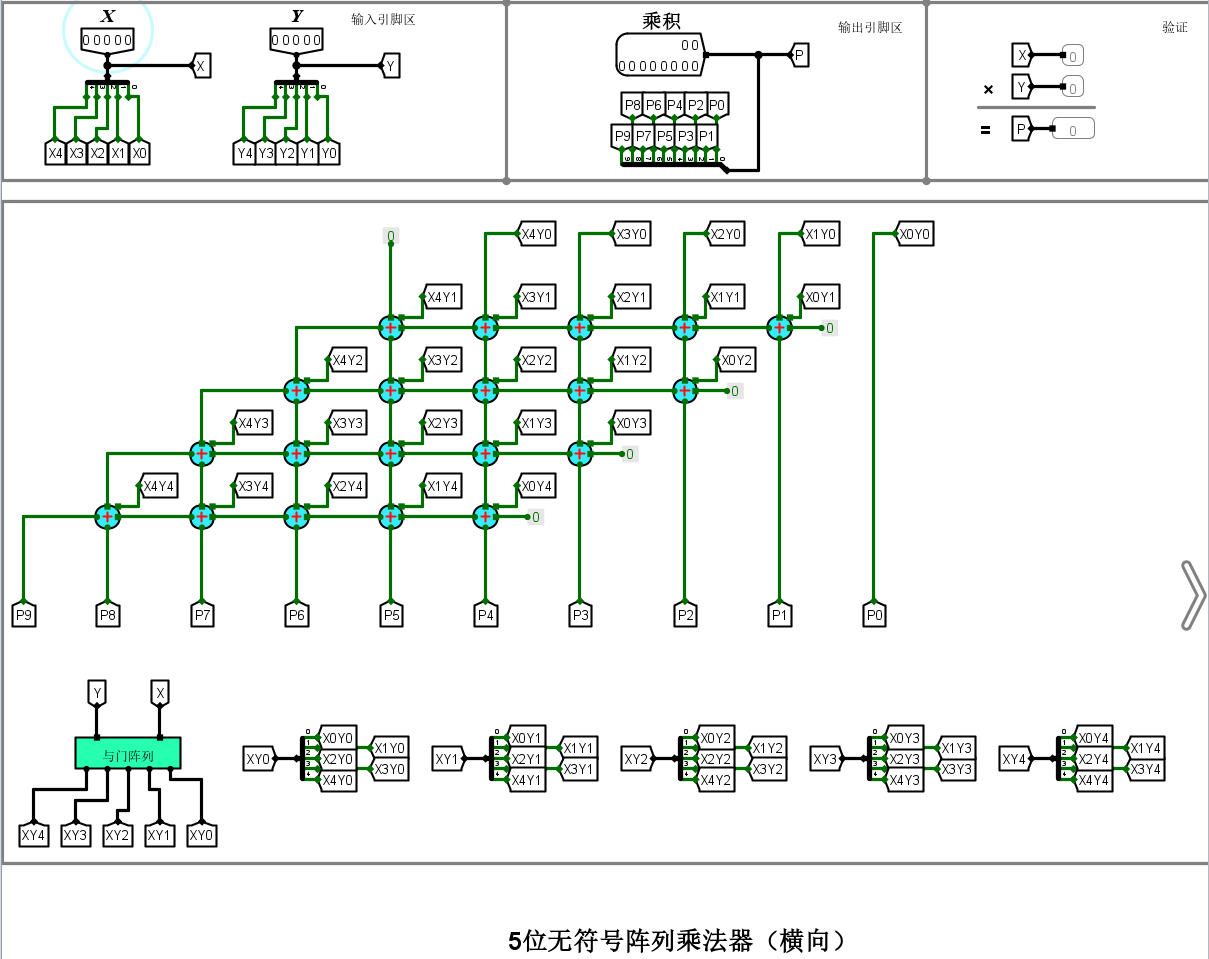




1. **X=0， Y=0， P是不是=0？**

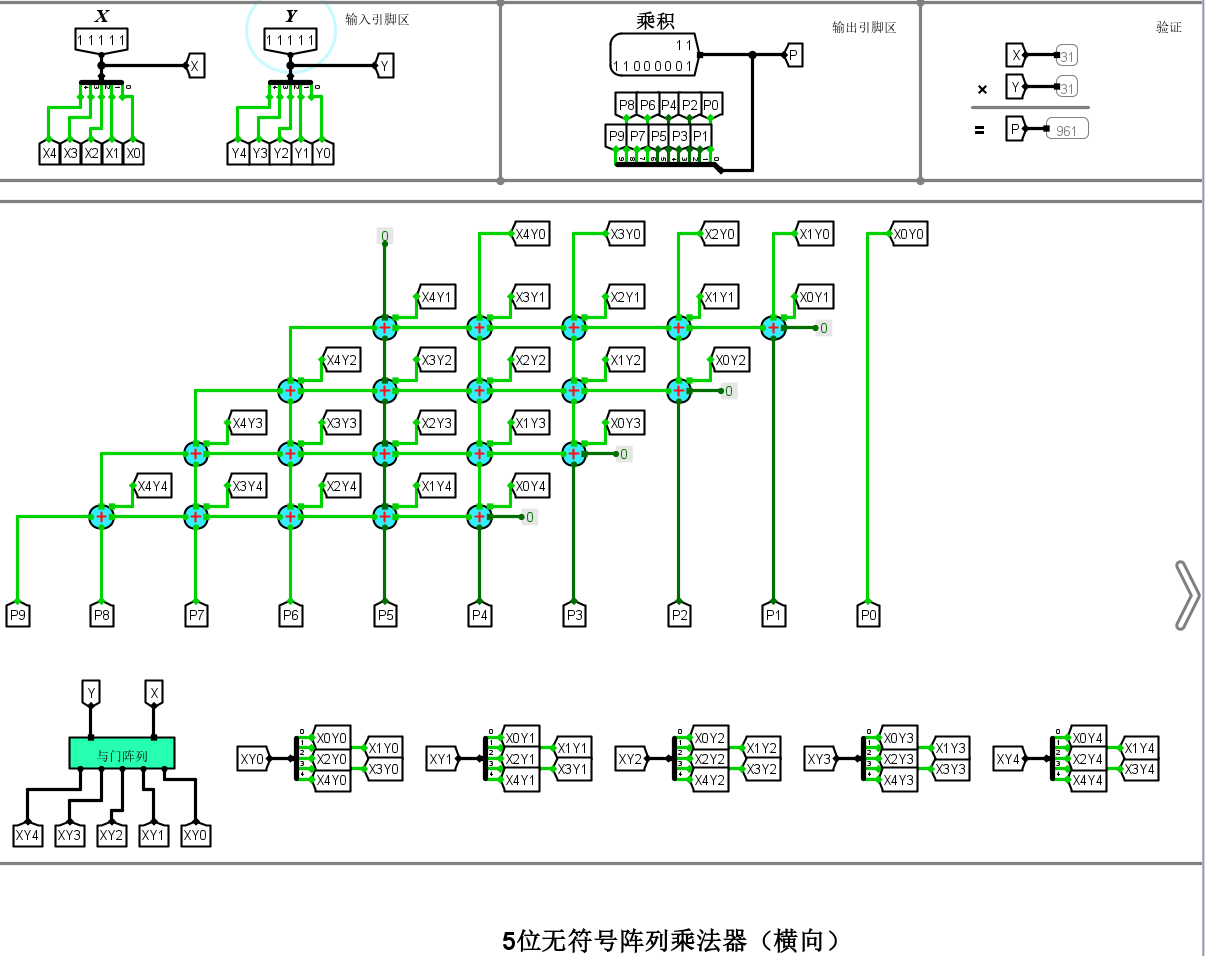
**X=0，Y=0，P=0**

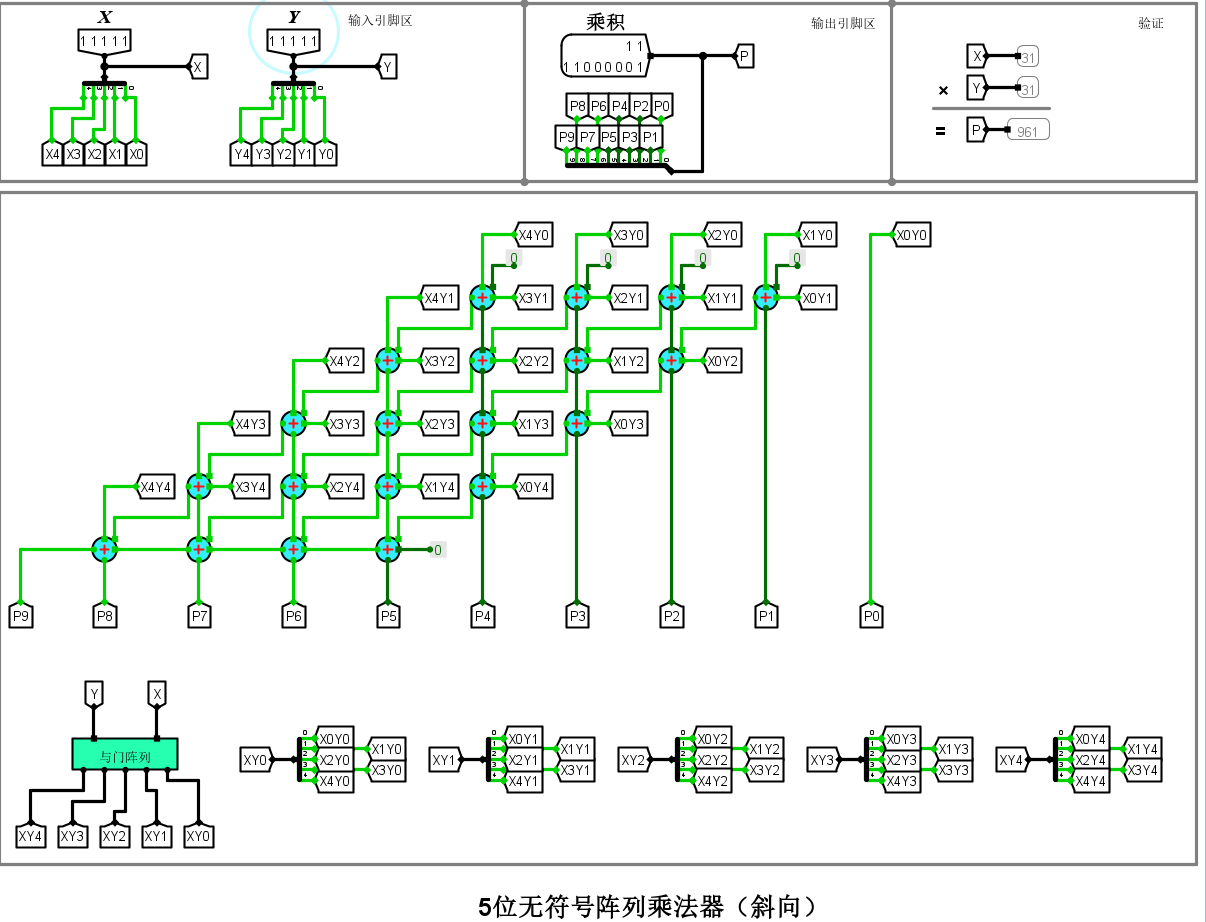




1. **X=31， Y=31， P是不是=961？**

**P=961**

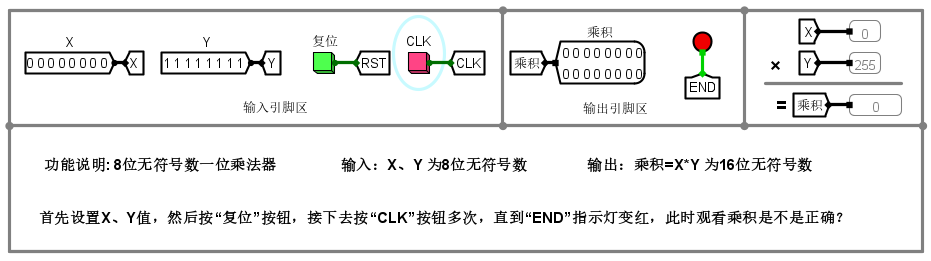




1. **8位无符号一位乘法器**

**具体要求：按照下述要求，设置不同的X,Y值（无符号数），完成验证工作：**

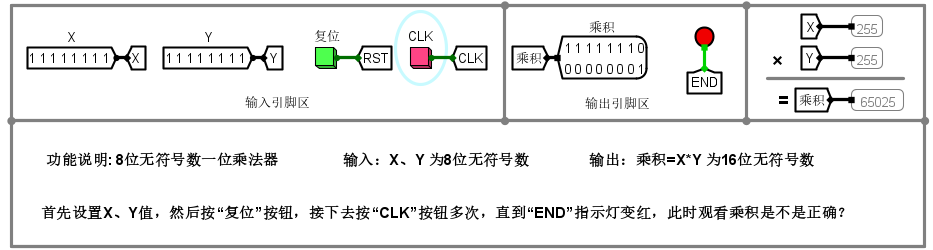
1. **X=0, Y任意，乘积是否为0？乘积为零**



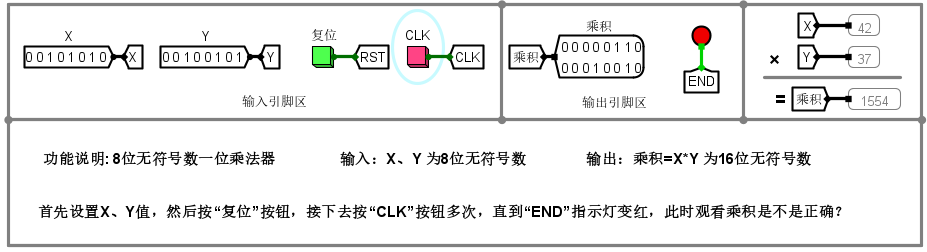
1. **X任意，Y=0，乘积是否为0？乘积为零**



1. **X=255,Y=255,乘积是否为65025？是**



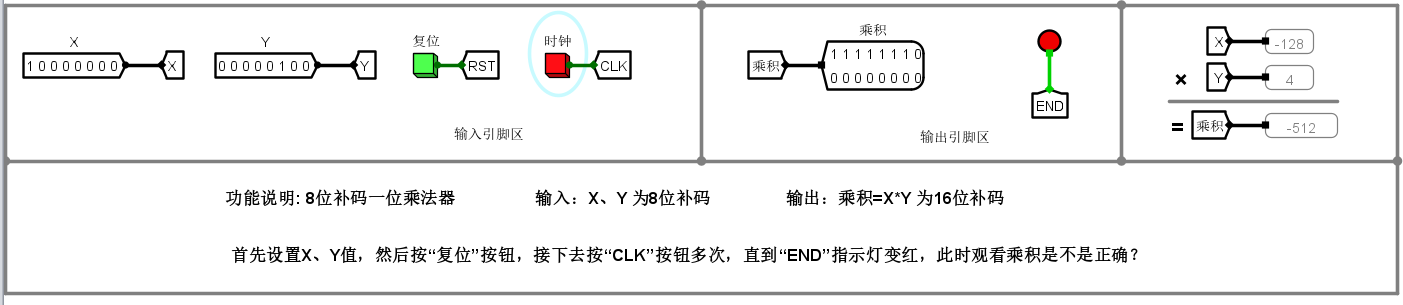
1. **X其他任意值，Y其他任意值，乘积是否正确？X=42，Y=37，乘积为1554正确**



1. **8位补码一位乘法器**

**具体要求：按照下述要求，设置不同的X、 Y值（补码），完成验证工作：**

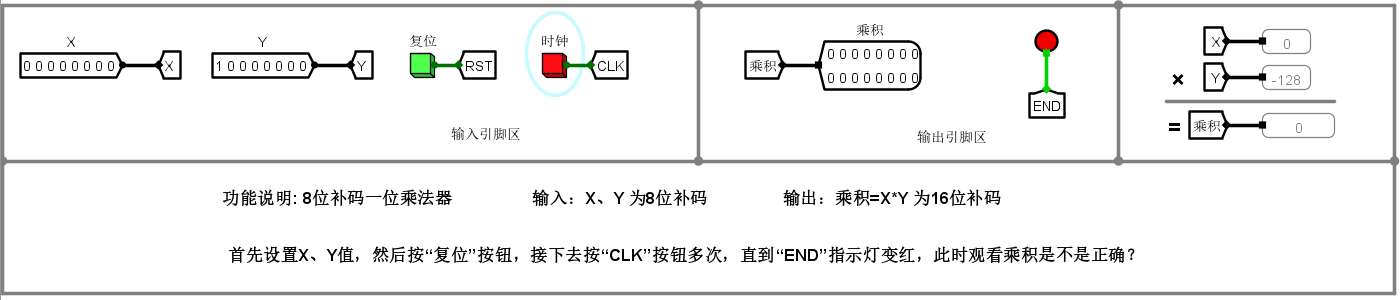
* 1. **X=-128， Y=其他， P是否正确？X=-128,Y=4,乘积为-512正确**



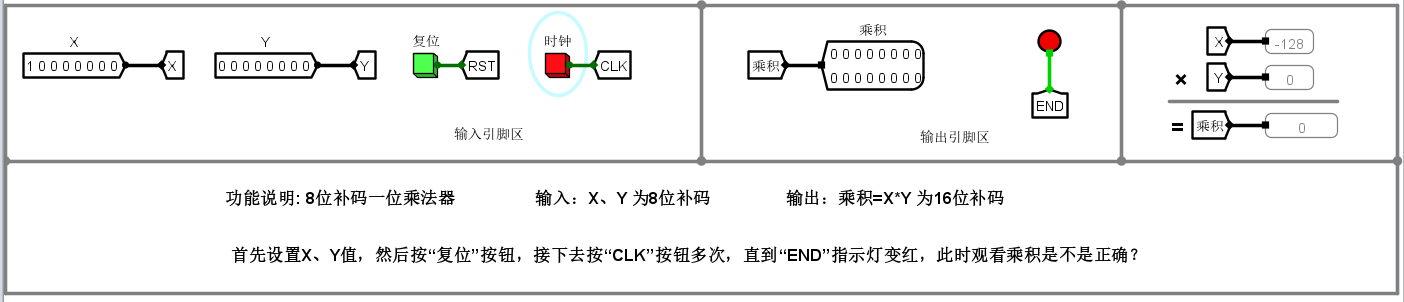
* 1. **X=其他， Y=-128， P是否正确？X=2,Y=-128,乘积为-256正确**



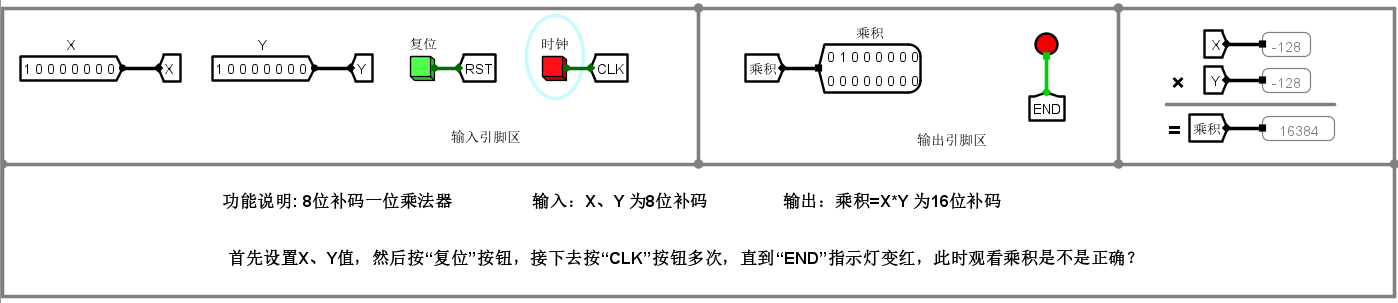
* 1. **X=0， Y=非0， P是否=0？X=0,Y=-128,乘积为0正确**



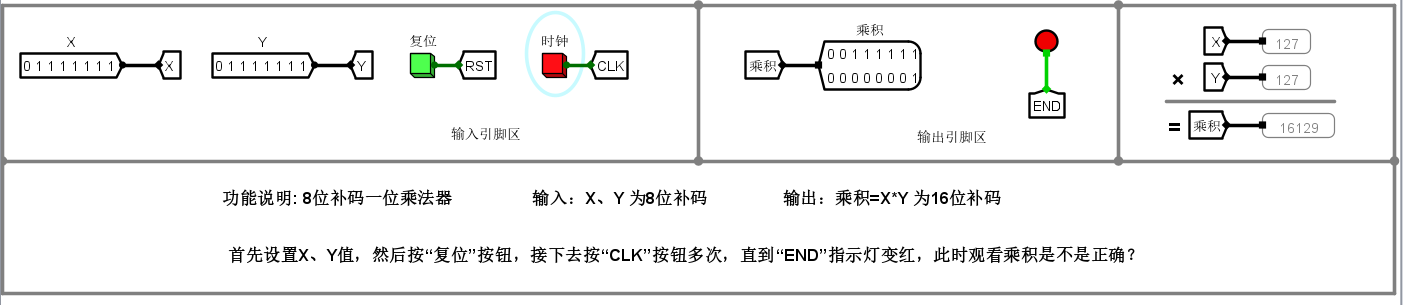
* 1. **X=非0， Y=0 ， P是否=0？X=-128,Y=0,乘积为0正确**



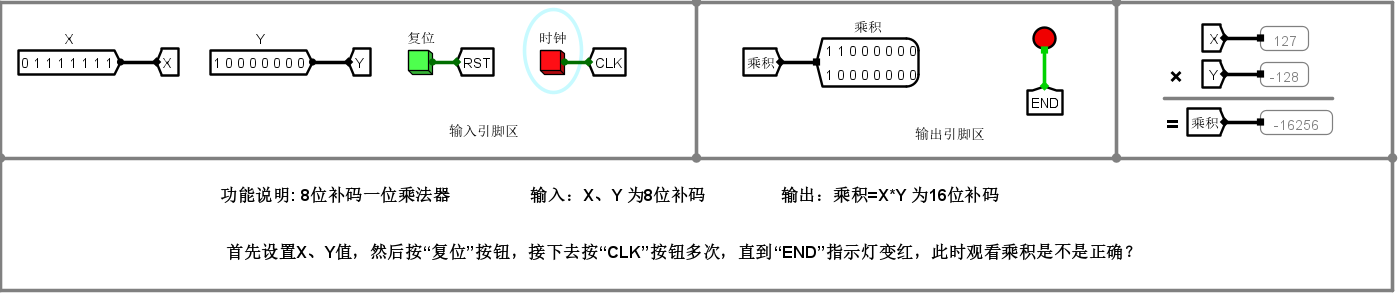
* 1. **X=-128， Y=-128， P是否=16384？是**



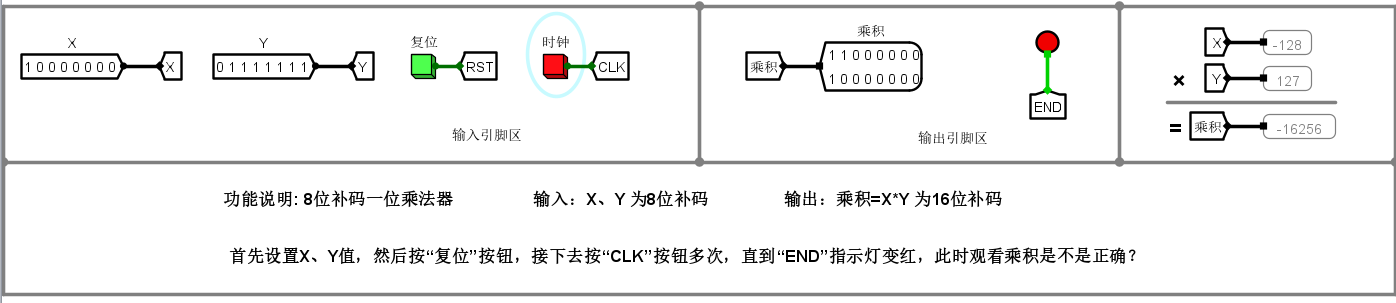
* 1. **X=127， Y=127， P是否=16129？是**



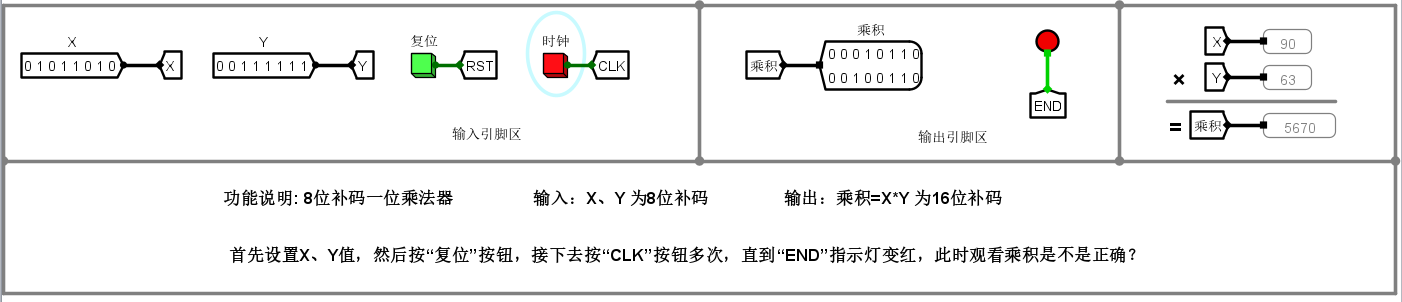
* 1. **X=127， Y==-128， P是否=-16256？是**



* 1. **X=-128， Y==127， P是否=-16256？是**



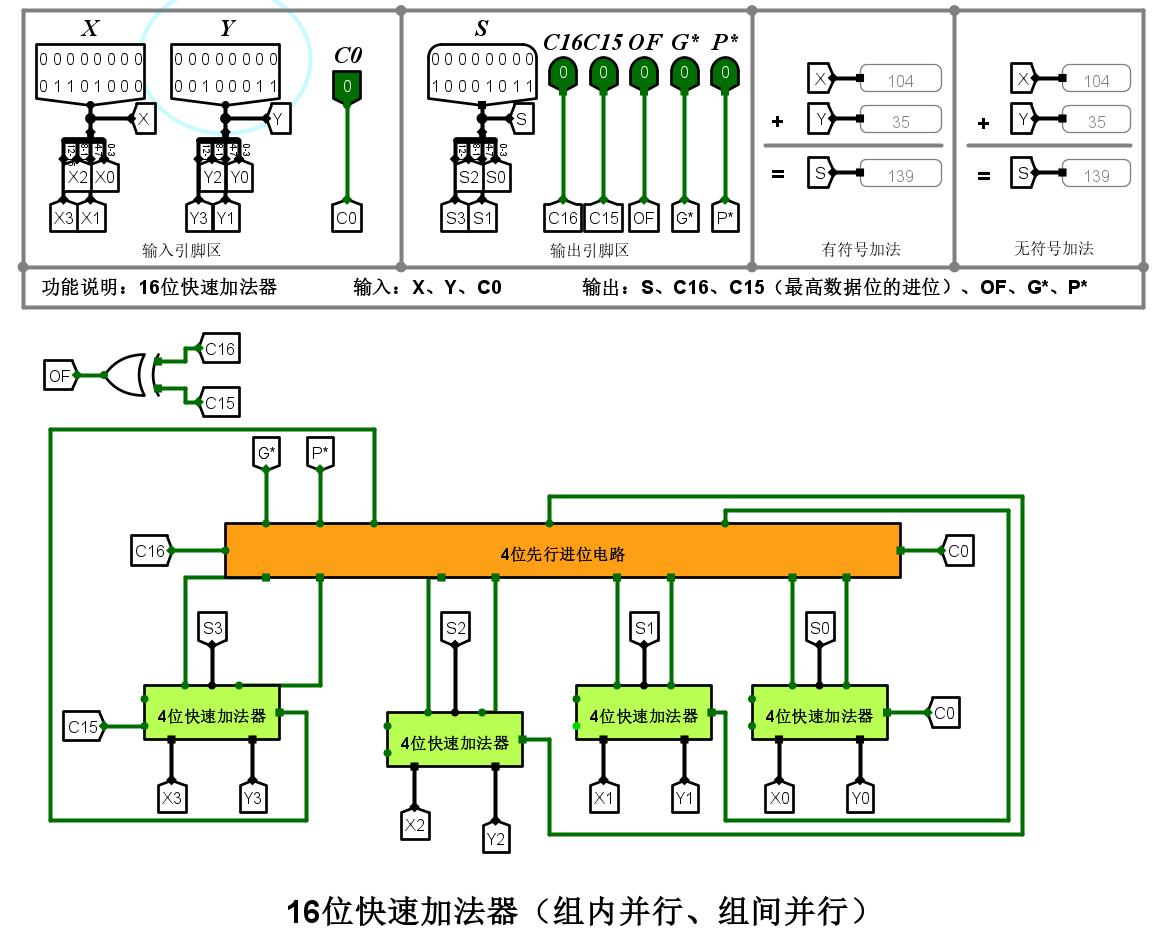
* 1. **X=其他， Y=其他， P是否正确？X=90,Y=63,乘积为5670，P正确**



* 1. **设计实验**

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**

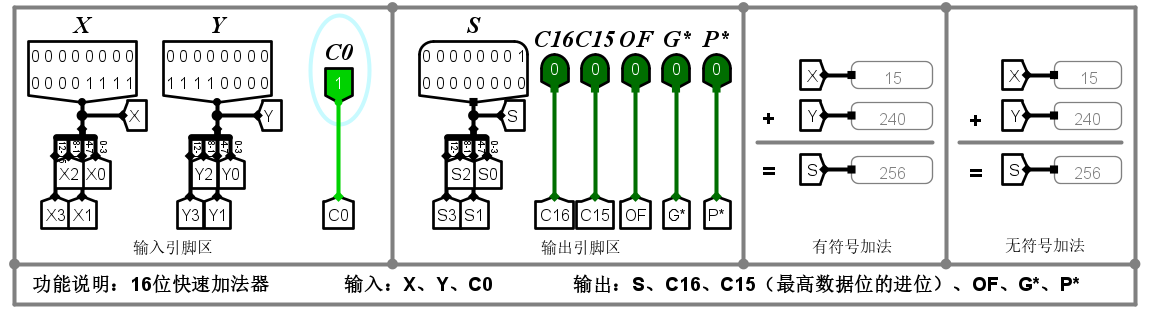
**思路：通过C0直接产生C4、C8、C12、C16的进位，电路如下：**



**具体要求：通过设置不同的X、 Y、 C0值，对电路进行验证，并对结果进行分析（以下4种情况）：**

**① 正数+正数=正数（不溢出）。**

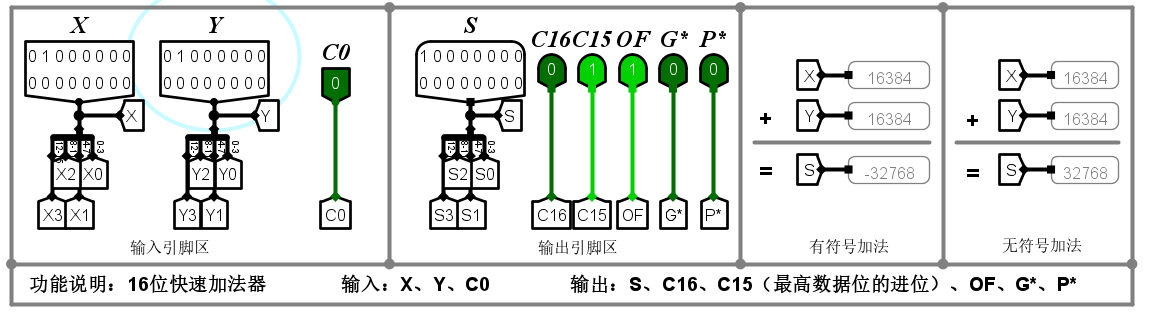
**设 X=1111，Y=11110000，C0=1，结果如下：**



**从 C15=0，C16=0，且异或后得0可知没有溢出**

**② 正数+正数=负数（溢出）。**

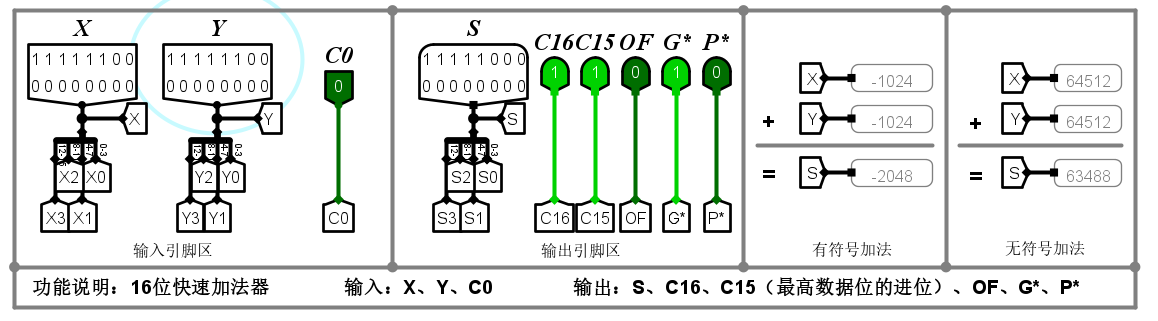
**设X=16384=Y，C0=0，结果如下：**



**16384+16384=32768，超出表示范围，故溢出，从C15=1，C16=0，且异或得1可知溢出**

**③ 负数+负数=负数（不溢出）。**

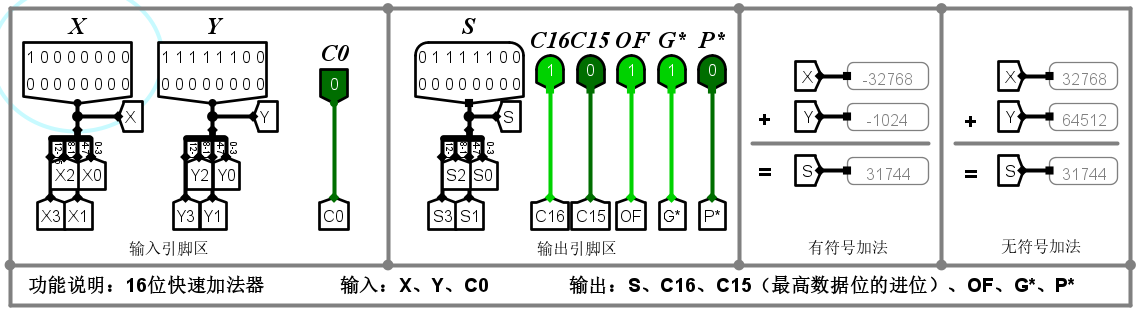
**设X=-1024=Y，C0=0，结果如下：**



**从C15=1，C16=1，且异或得0可知，没有溢出**

**④ 负数+负数=正数（溢出）。**

**设X=-32768，Y=-1024，C0=0，结果如下：**

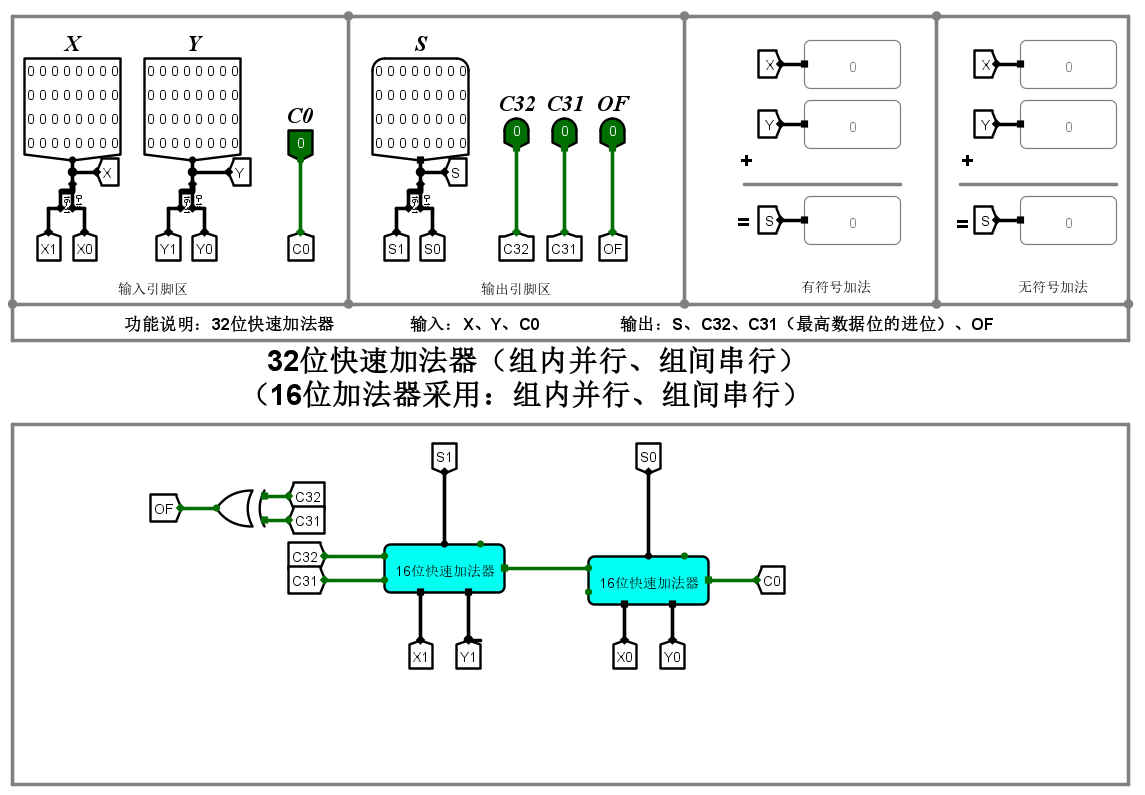


**由OF=1可知溢出**

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**

**通过将俩个十六位加法器串联即可得到三十二位加法器，两种电路如下：**

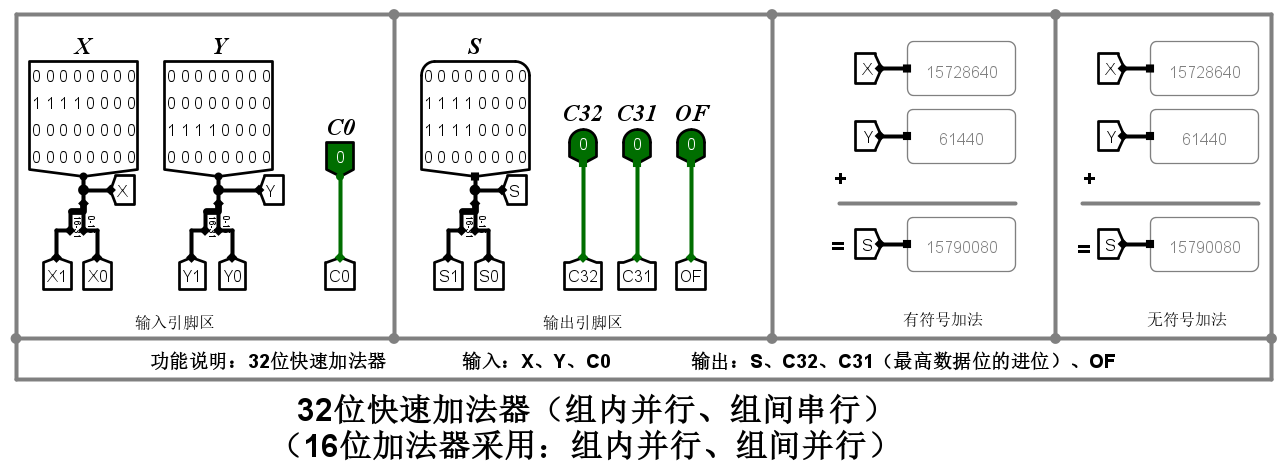


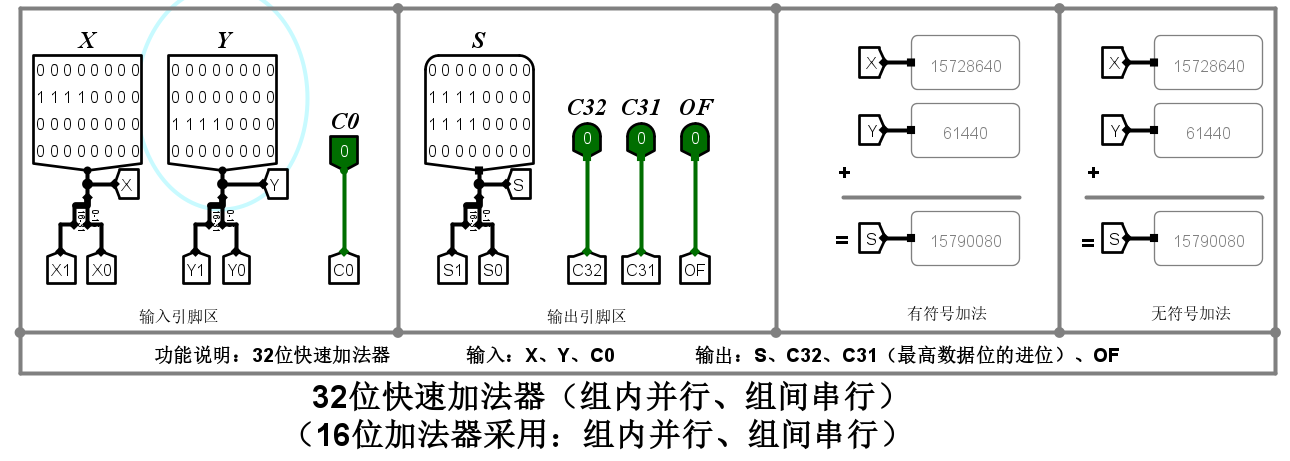


**具体要求：通过设置不同的X、 Y、 C0值，对电路进行验证，并对结果进行分析（以下4种情况）：**

**① 正数+正数=正数（不溢出）。**

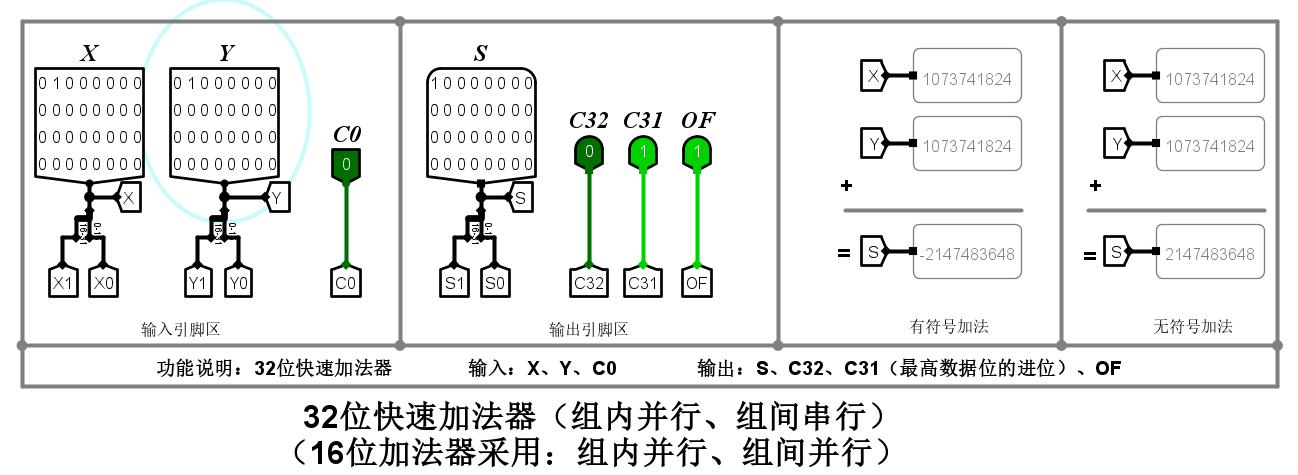
**设X=00F00000H，Y=000F0000H，C0=0，如图：**

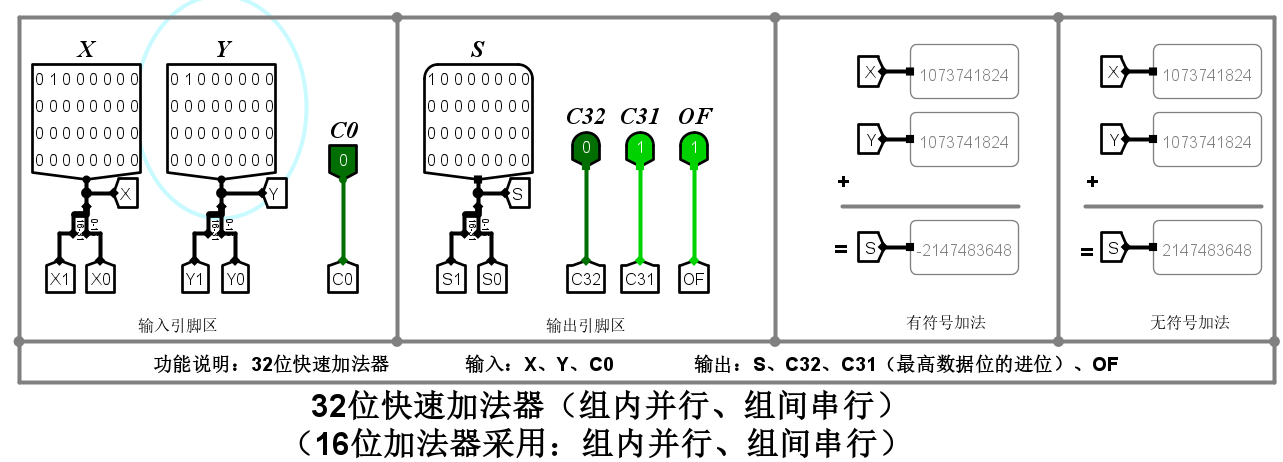




**② 正数+正数=负数（溢出）。**

**设X=** **1073741824，Y=** **1073741824，C0=0，结果如下：**

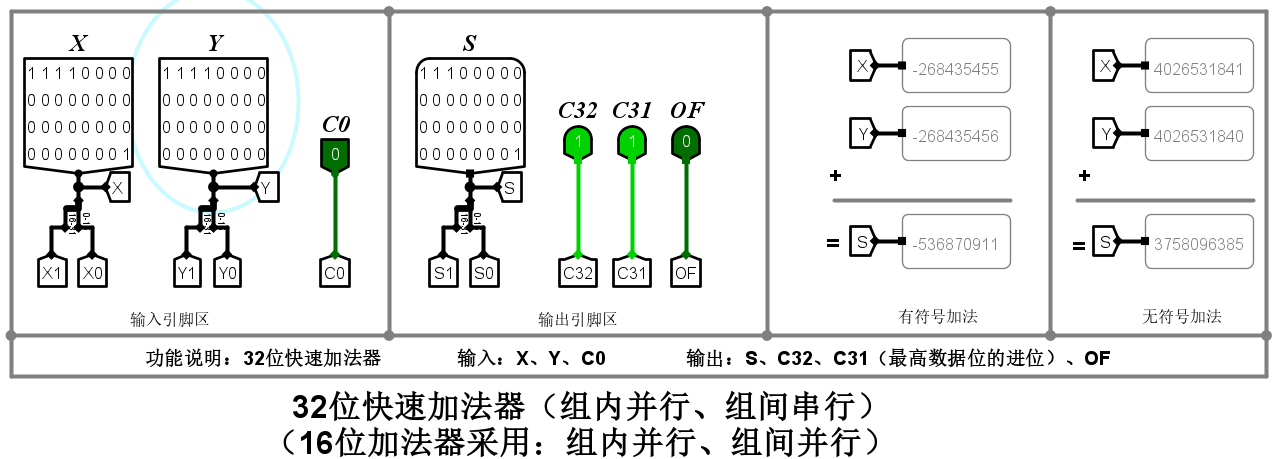


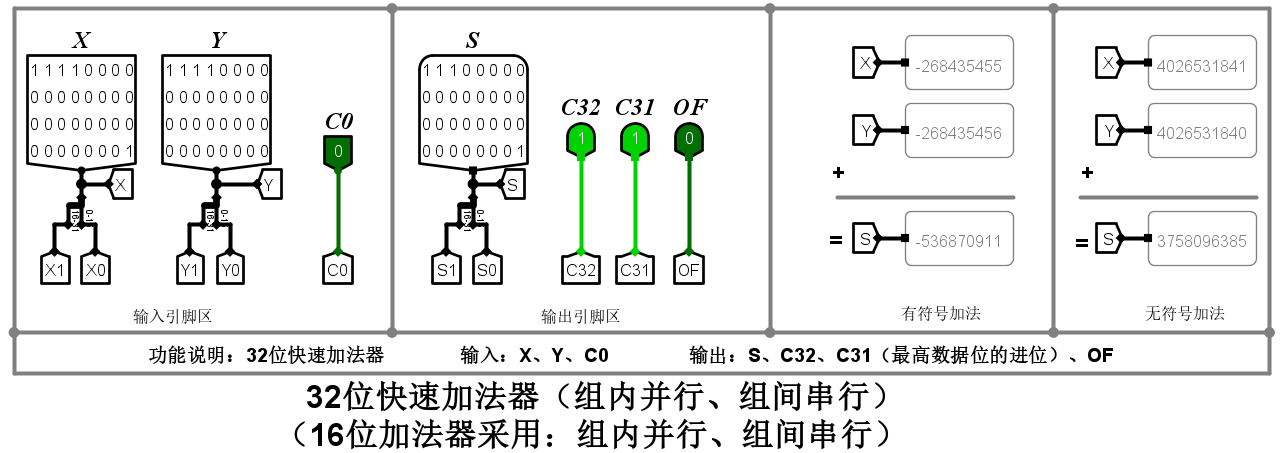


**由OF=1可知溢出**

**③ 负数+负数=负数（不溢出）。**

**设X=** **-268435455，Y=** **-268435456，C0=0，结果如下：**

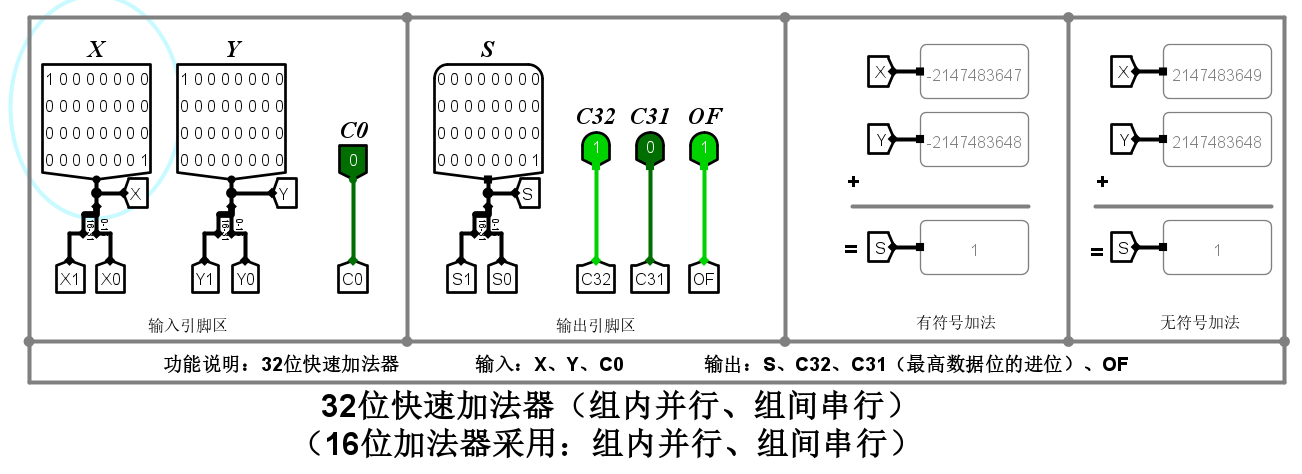


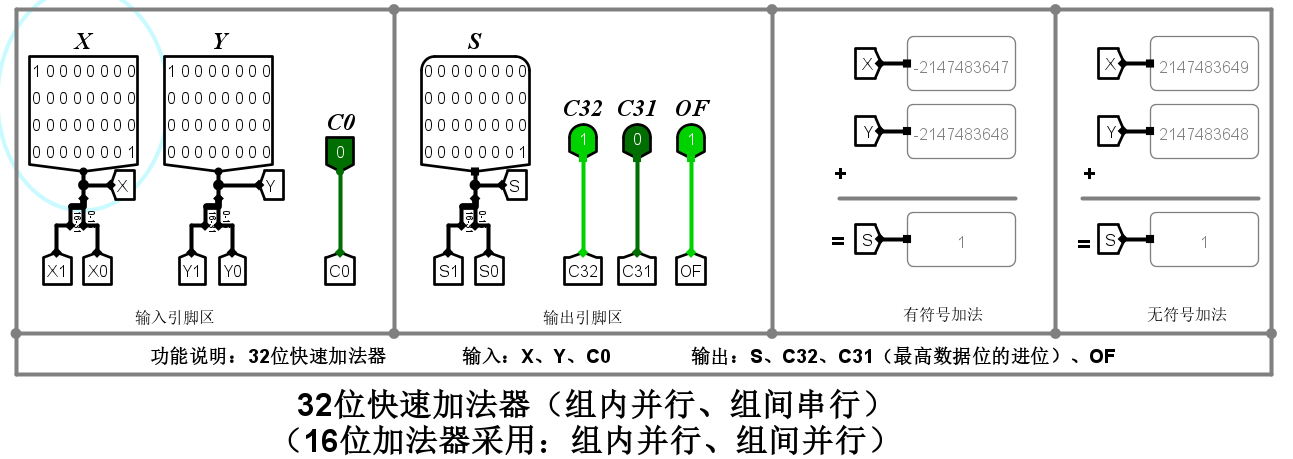


**S=** **-536870911为负数且OF=0，所以没有溢出**

**④ 负数+负数=正数（溢出）。**

**设X=** **-2147483648，Y=** **-2147483647，C0=0，结果如下：**

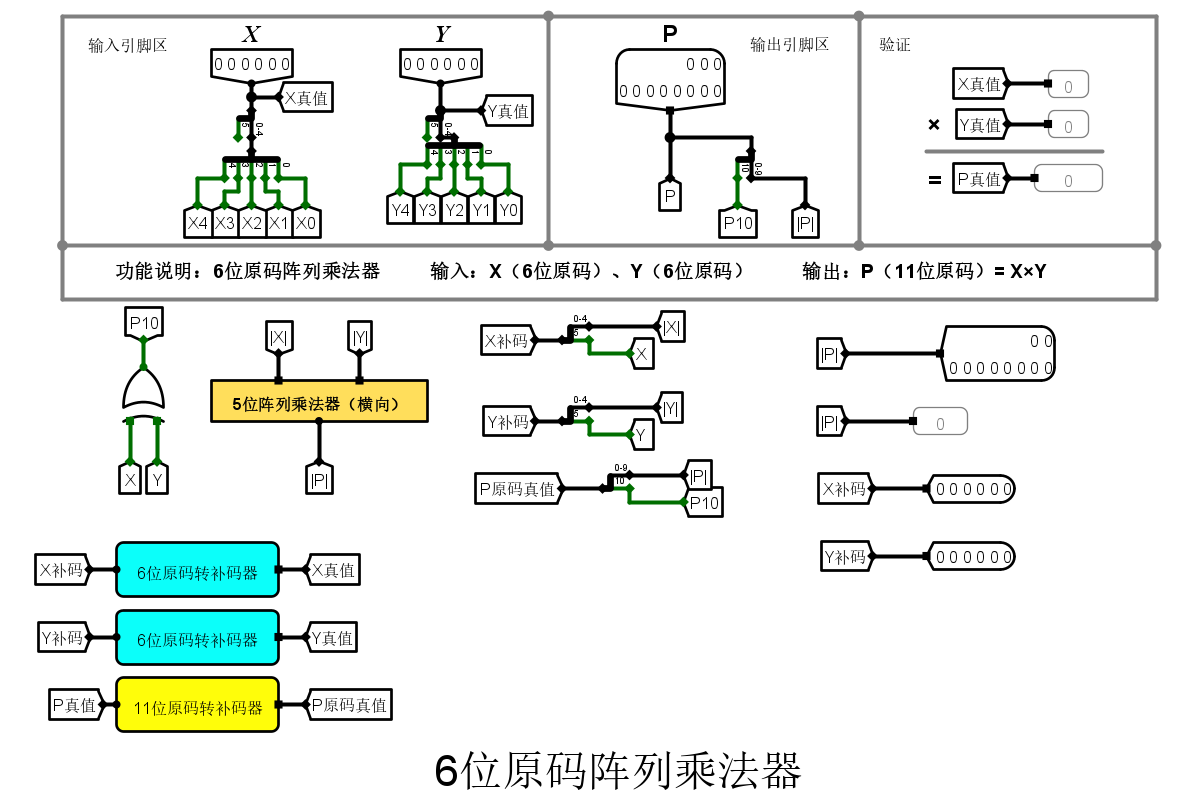




**S=1为正数且OF=1，所以溢出**

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**

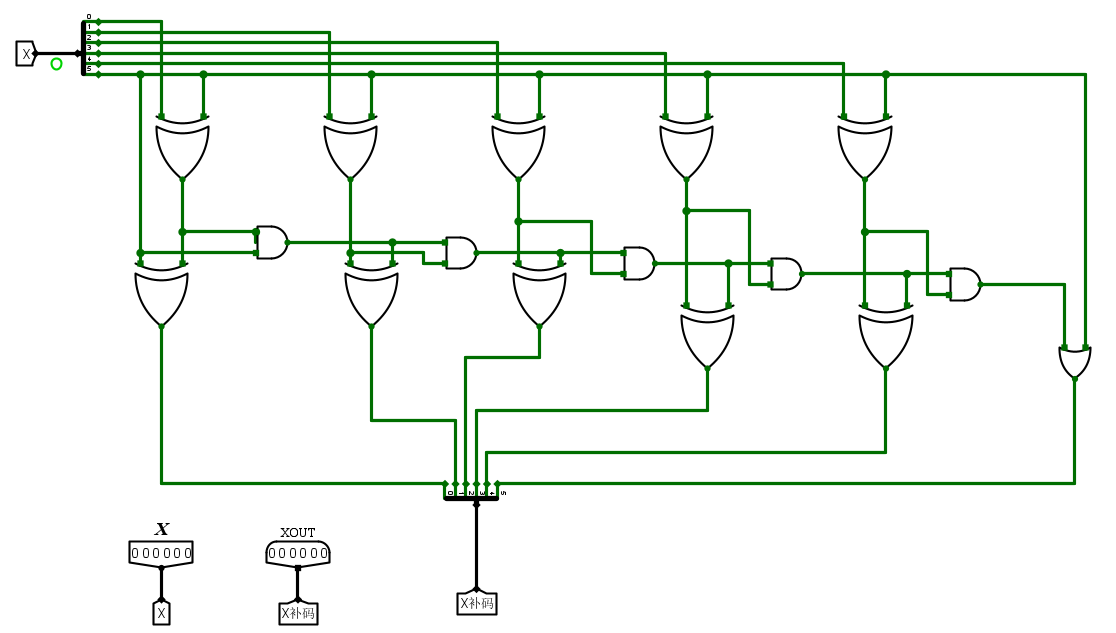
**设计电路如下：**



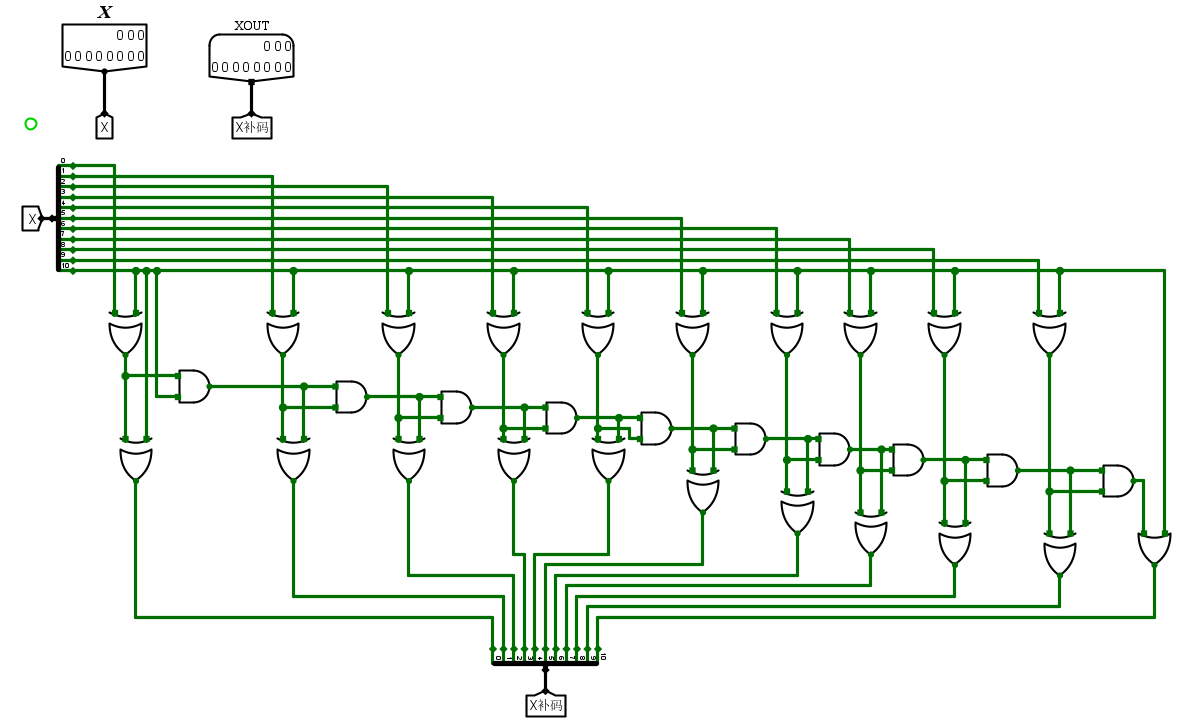
**在其中使用了5位阵列乘法器、设计的6位原码转补码电路以及11位原码转补码电路。**

**五位阵列乘法器电路与实验示例电路相同；**

**6位原码转补码电路如下：**



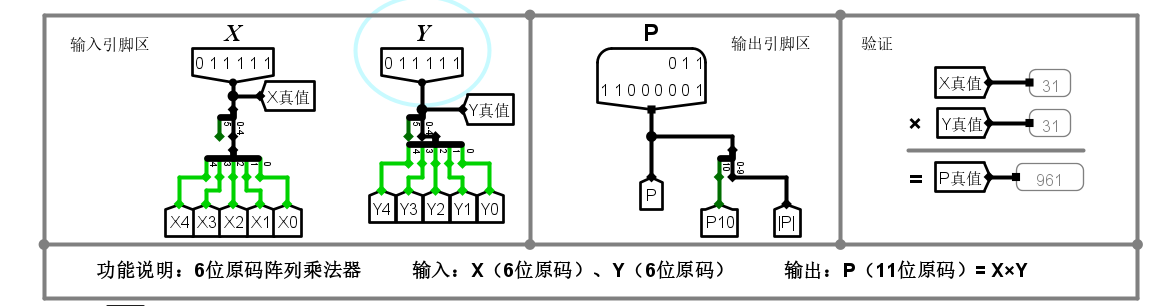
**11位原码转补码电路如下：**



**具体要求：通过改变X、 Y的值，对该乘法器进行验证：**

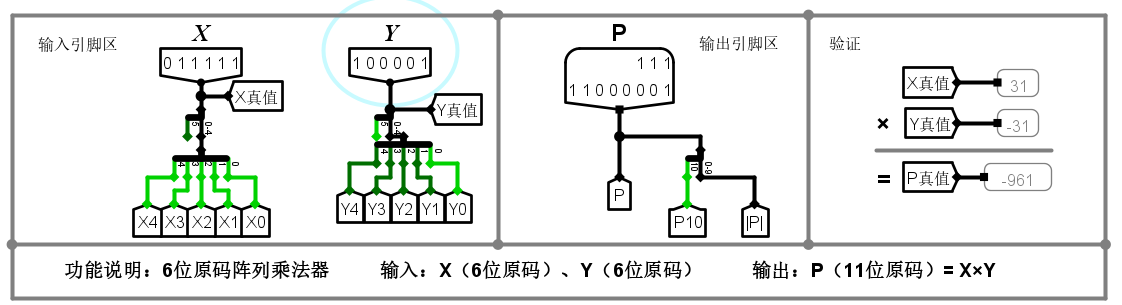
1. **X=正数， Y=正数， P=正数（例如： X=31、 Y=31、 P=961）。**

**设X=31,Y=31,则运算结果如下，P=961**



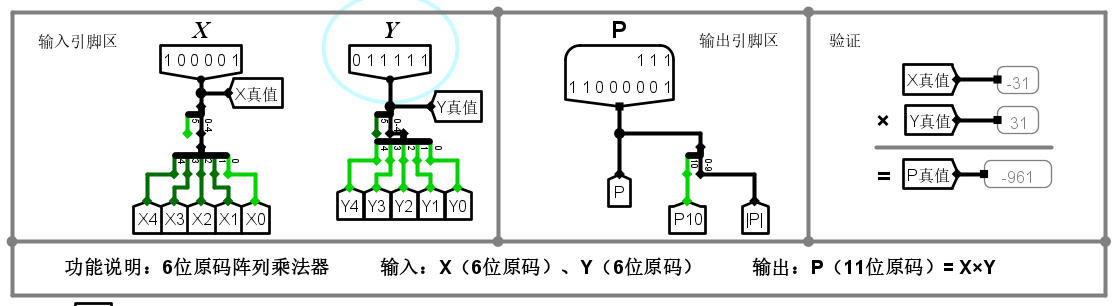
1. **X=正数， Y=负数， P=负数（例如： X=31、 Y=-31、 P=-961）。**

**设X=31,Y=-31则运算结果如下，P=-961**



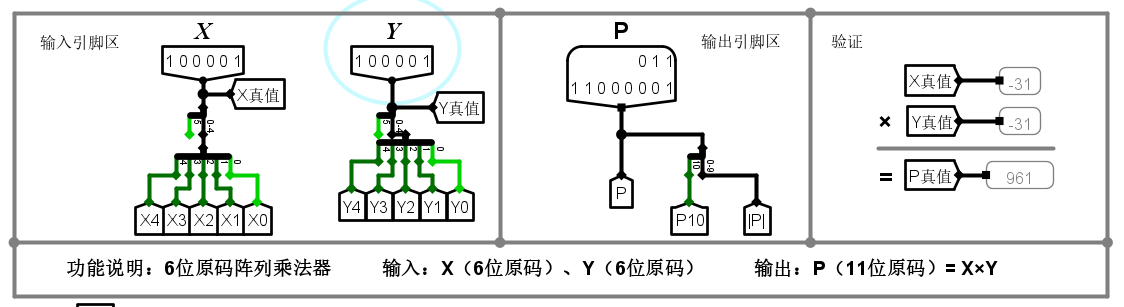
1. **X=负数， Y=正数， P=负数（例如： X=-31、 Y=31、 P=-961）。**

**设X=-31,Y=31,则运算结果如下，P=-961**



1. **X=负数， Y=负数， P=正数（例如： X=-31、 Y=-31、 P=961）。**

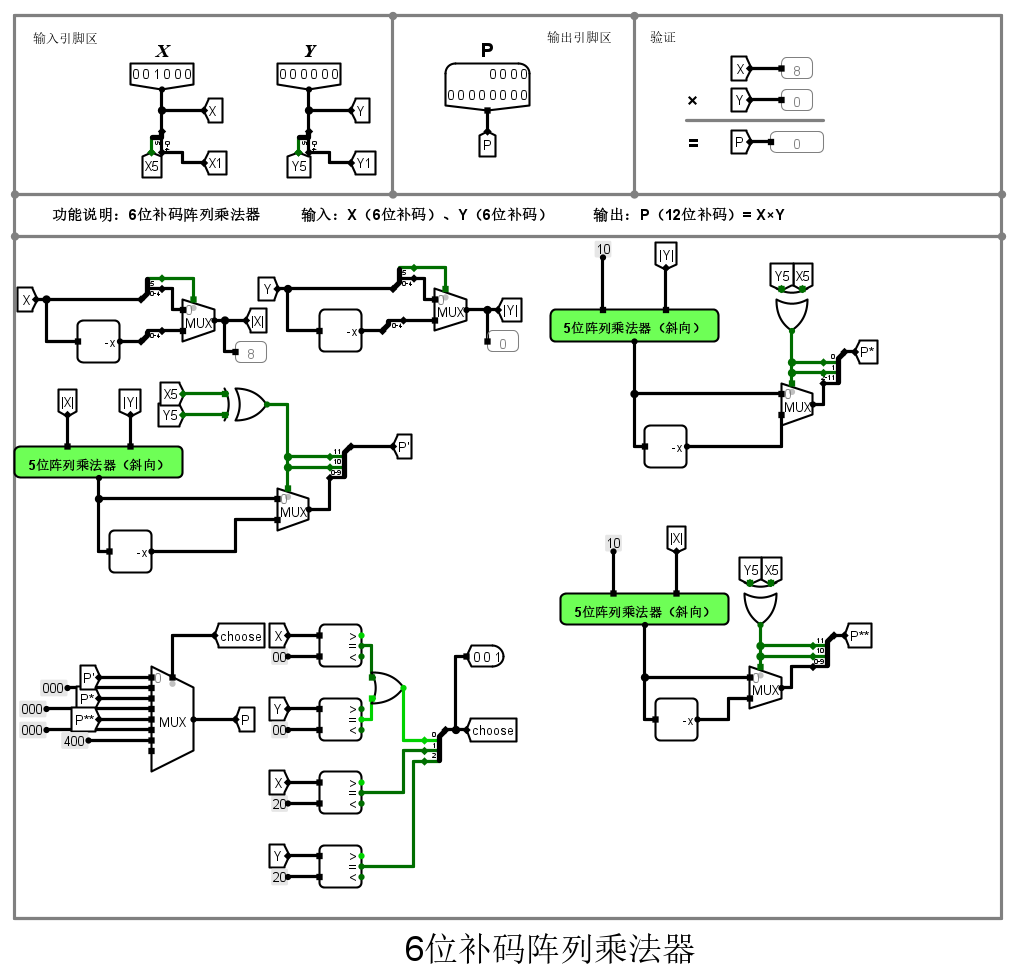
**设X=--31,Y=-31,则运算结果如下，P=961**



**• 注意：需要将X、 Y、 P的原码转换为真值进行显示**

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**

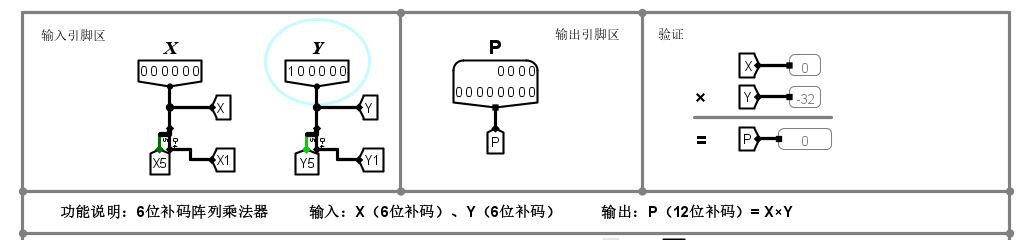
**电路如下：**



**具体要求： 通过改变X、 Y的值，对该乘法器进行验证：**

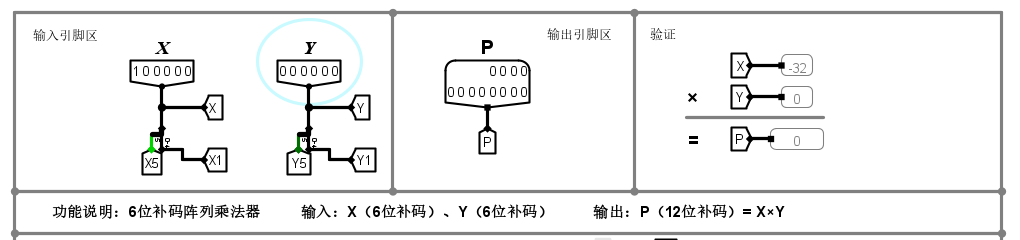
**① X=0， Y=负数； P=0。**

**例如X=0，Y=-32，结果如下图，P=0：**



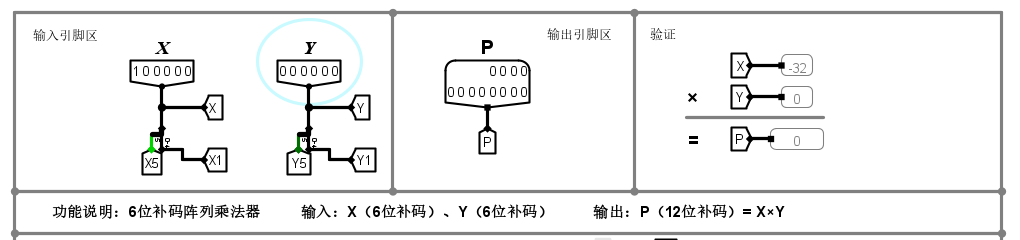
**② X=负数， Y=0； P=0。**

**例如X=-32，Y=0，结果如下图，P=0：**



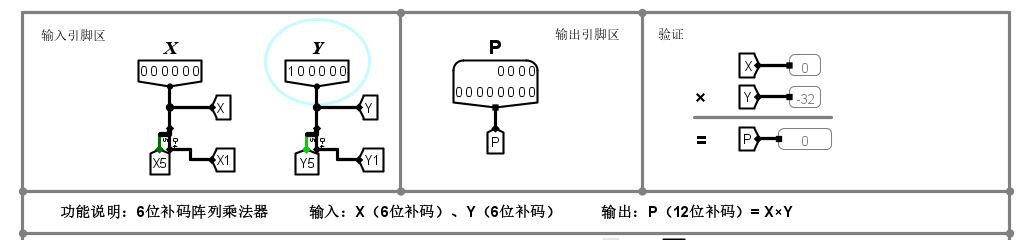
**③ X=-32， Y=0或正数； P=正确的数。**

**例如X=-32，Y=0，结果如下图，P=0：**



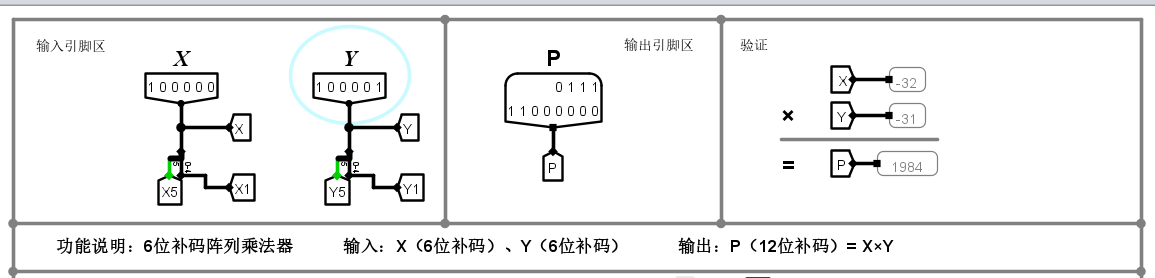
**④ X=0或正数， Y=-32； P=正确的数。**

**例如X=0，Y=-32，结果如下图，P=0：**



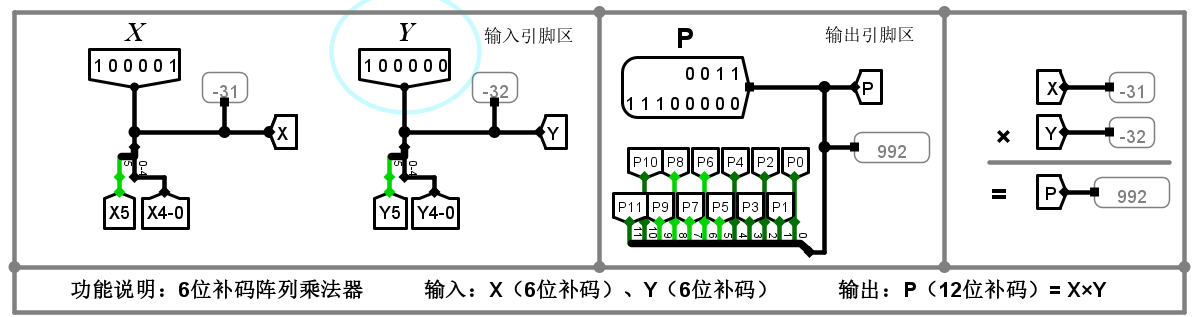
**⑤ X=-32， Y=负数； P=正确的数。**

**例如X=-32，Y=-31，结果如下图，P=1984：**



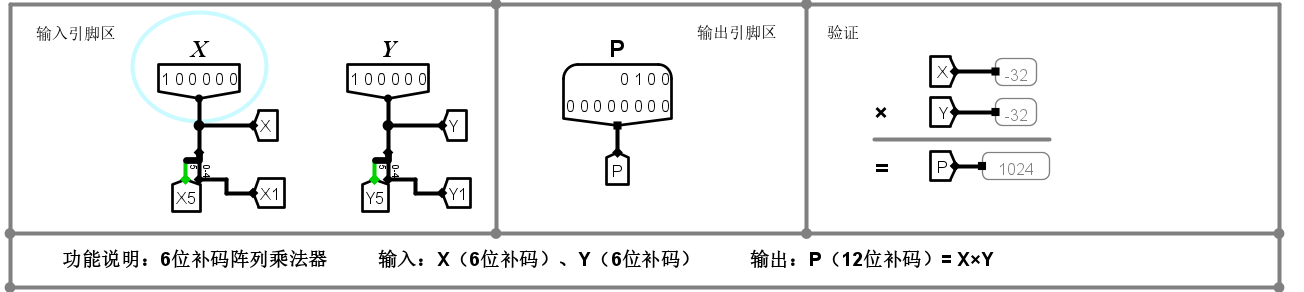
**⑥ X=负数， Y=-32； P=正确的数。**

**例如X=-31，Y=-32，结果如下图，P=992：**



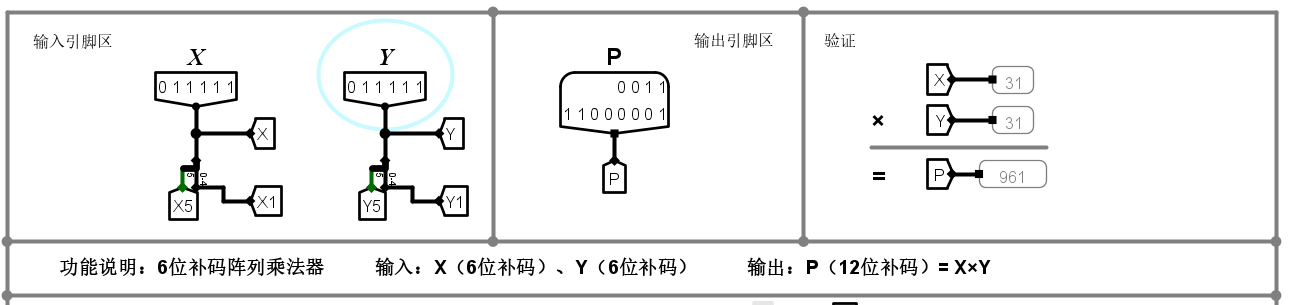
**⑦ X=-32， Y=-32， P=1024。**

**例如X=-32，Y=-32，结果如下图，P=1024：**



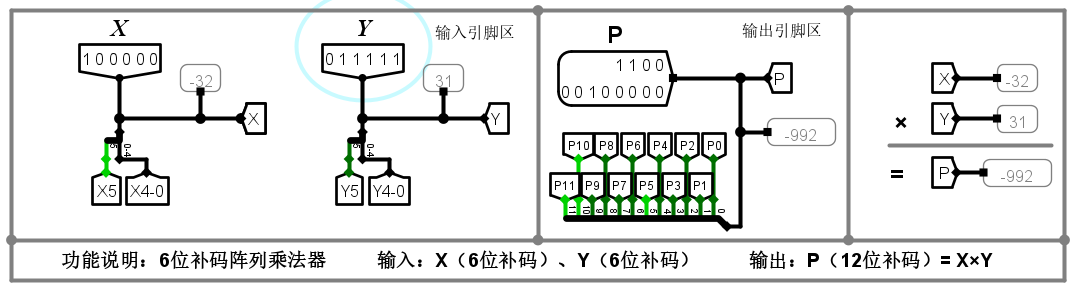
**⑧ X=31， Y=31， P=961。**

**例如X=-32，Y=31，结果如下图，P=961：**



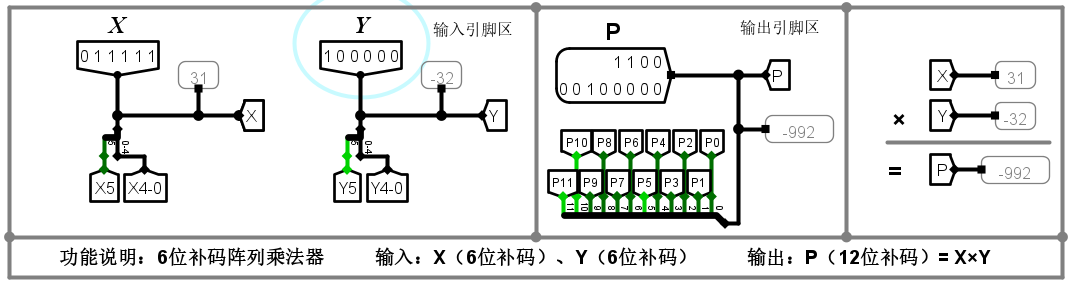
**⑨ X=-32， Y=31， P=-992。**

**例如X=-32，Y=31，结果如下图，P=-992：**



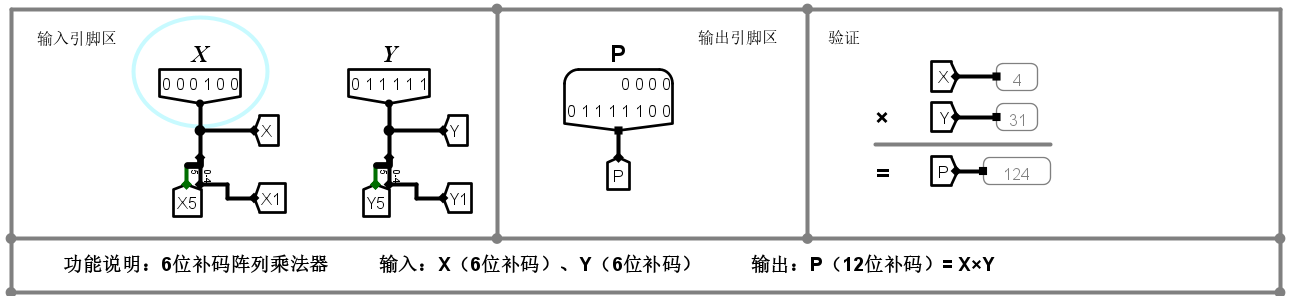
**⑩ X=31， Y=-32， P=-992。**

**例如X=31，Y=-32，结果如下图，P=-992：**



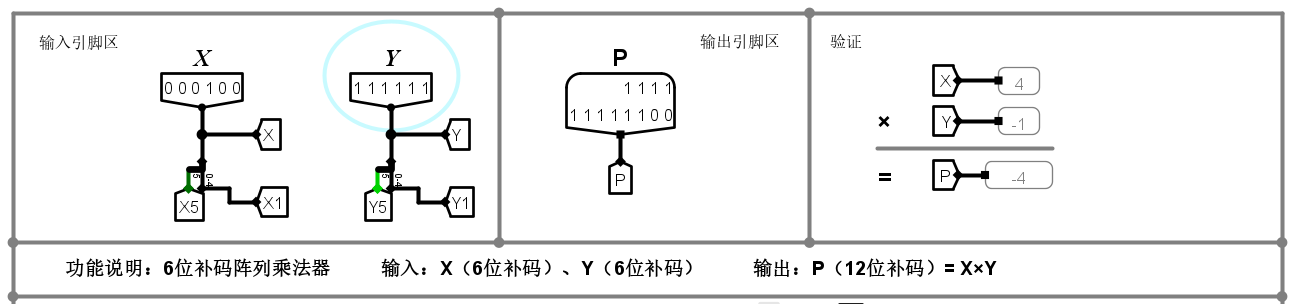
⑪ **X=正数， Y=正数； P=正确的数。**

**例如X=4，Y=31，结果如下图，P=124：**



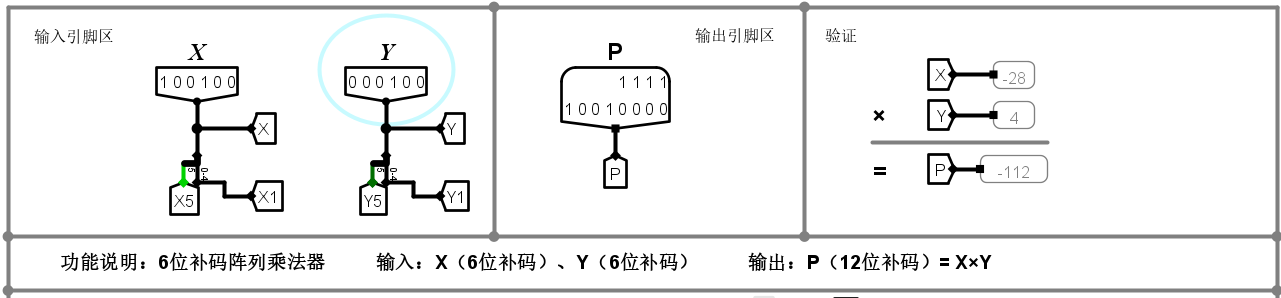
⑫ **X=正数， Y=负数； P=正确的数。**

**例如X=4，Y=-1，结果如下图，P=-4：**



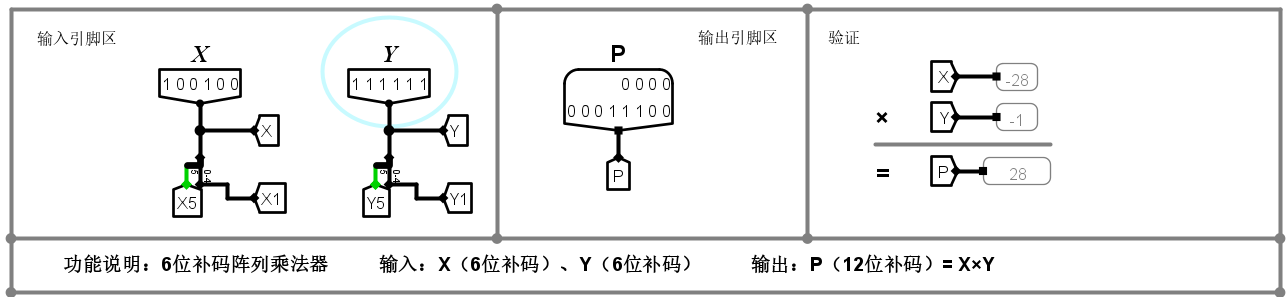
⑬ **X=负数， Y=正数； P=正确的数。**

**例如X=-28，Y=4，结果如下图，P=-112：**



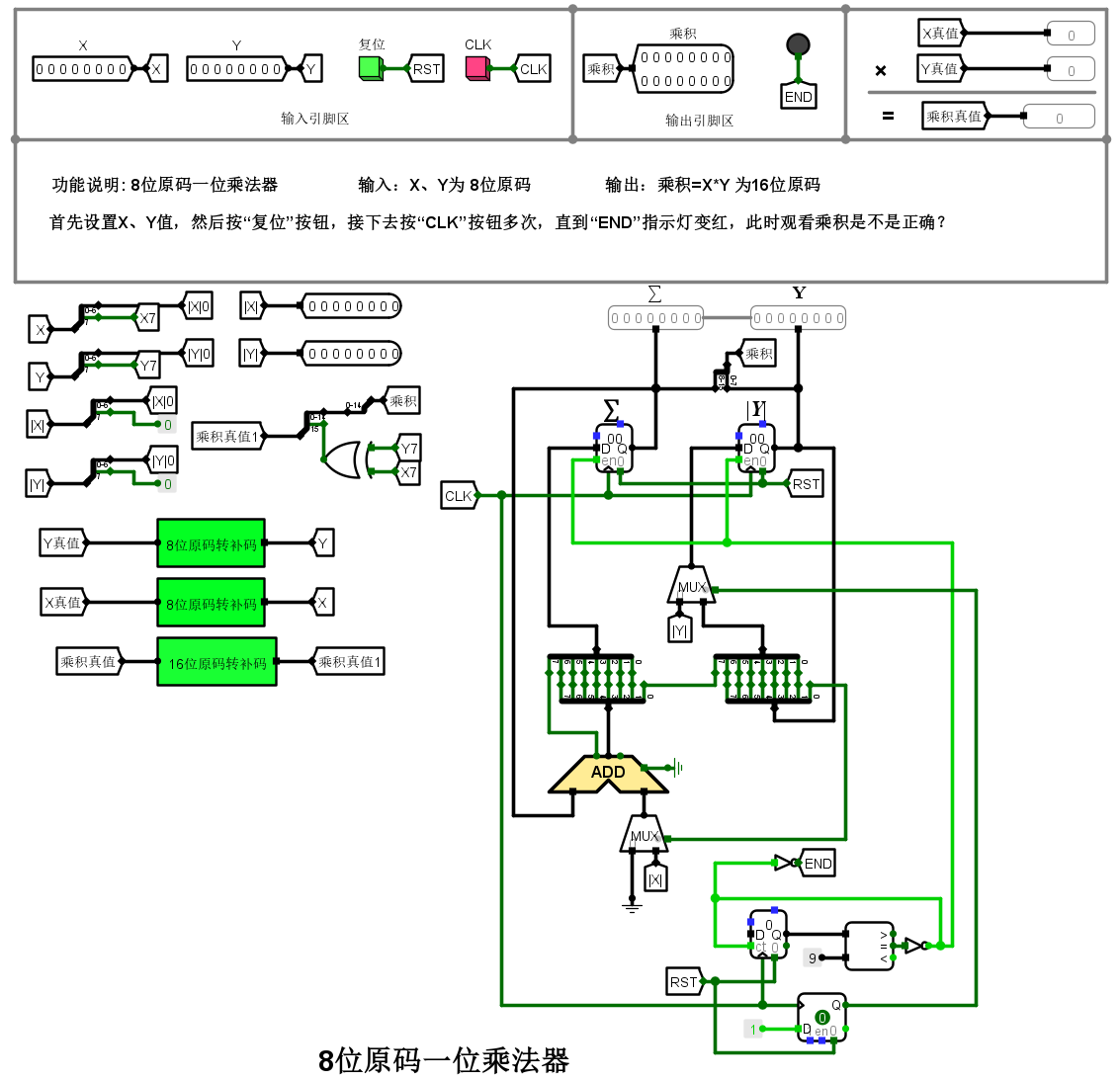
⑭ **X=负数， Y=负数； P=正确的数。**

**例如X=-28，Y=-1，结果如下图，P=28：**



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**

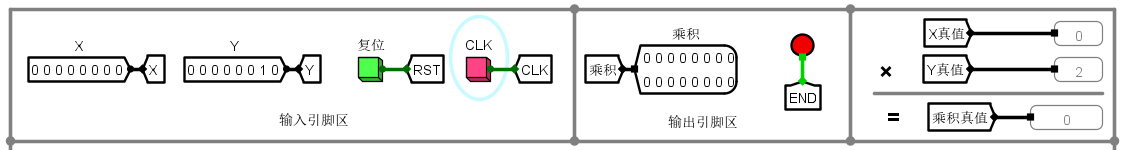
**电路设计如下：**



**具体要求：按照下述要求，设置不同的X、 Y值（原码），完成验证工作：**

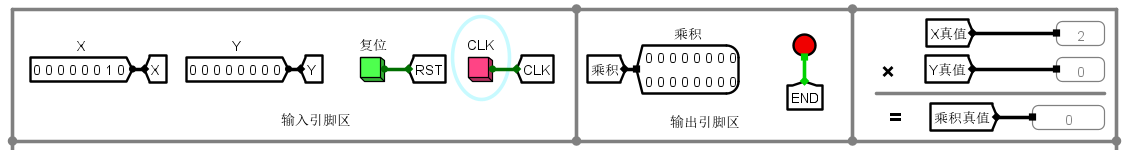
1. **X=0， Y=任意， P是否=0？**

**输入X=0，Y=2，得到结果P=0：**



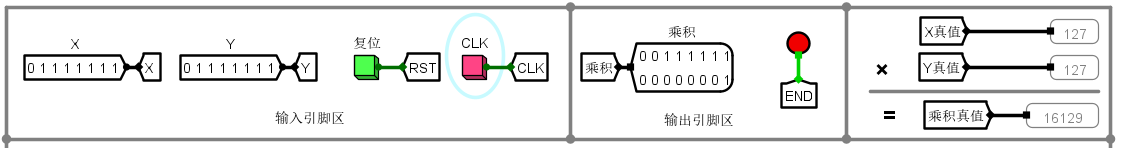
1. **X=任意， Y=0， P是否=0？**

**输入X=2，Y=0，得到结果P=0：**



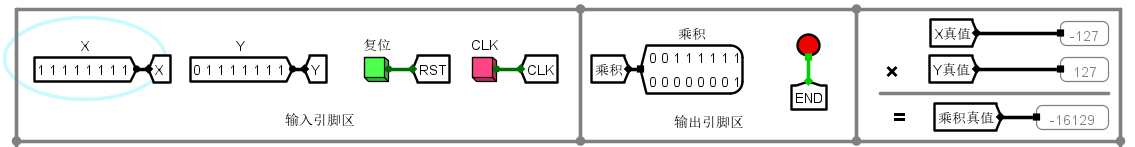
1. **X=127， Y=127， P是否=16129？**

**输入X=127，Y=127，得到结果P=16129：**



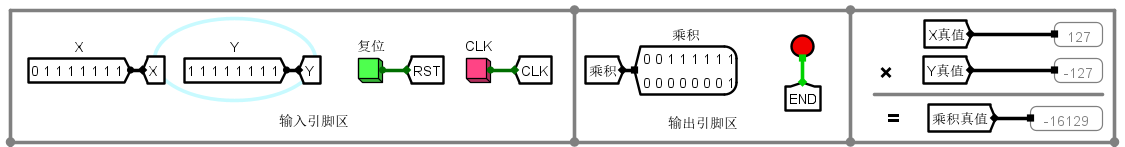
1. **X=-127， Y=127， P是否=-16129？**

**输入X=-127，Y=127，得到结果P=-16129：**



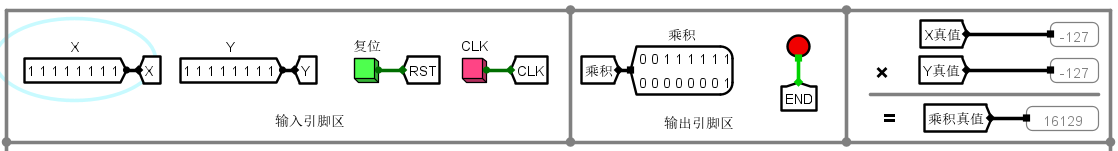
1. **X=127， Y=-127， P是否=-16129？**

**输入X=127，Y=-127，得到结果P=-16129：**



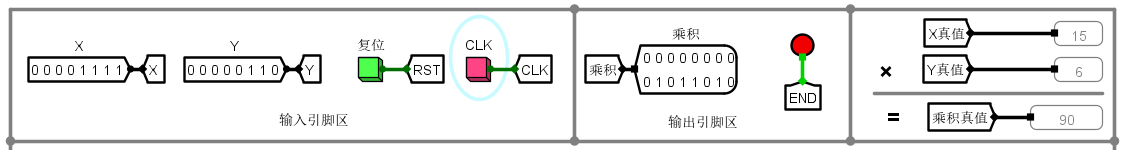
1. **X=-127， Y=-127， P是否=16129？**

**输入X=-127，Y=-127，得到结果P=16129：**



1. **X=其他， Y=其他， P是否正确？**

**输入X=8，Y=2，得到结果P=0：**



* 1. **挑战性实验**

**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；（注：完成挑战性实验的，本次实验成绩+10分。）**

**具体要求：按照下述要求，设置不同的X、 Y值（补码），完成验证：**

**（1） X=-128， Y=其他， P是否正确？**

**（2） X=其他， Y=-128， P是否正确？**

**（3） X=0， Y=非0， P是否=0？**

**（4） X=非0， Y=0 ， P是否=0？**

**（5） X=-128， Y=-128， P是否=16384？**

**（6） X=127， Y=127， P是否=16129？**

**（7） X=127， Y==-128， P是否=-16256？**

**（8） X=-128， Y==127， P是否=-16256？**

**（9） X=其他， Y=其他， P是否正确？**

1. **实验报告提交**
2. **实验报告命名为：学号+姓名+第2次实验报告.pdf。**
3. **将实验报告和设计文件打包为1个压缩文件，压缩文件命名为：学号+姓名+第2次实验.zip。**

**实验报告中应按照上述各项目中的“具体要求”进行数据验证，截屏以及文字等进行佐证。**

1. **将压缩文件上传到course.xmu.edu.cn，截止时间（2周内）：2025年3月26日晚上24点。**