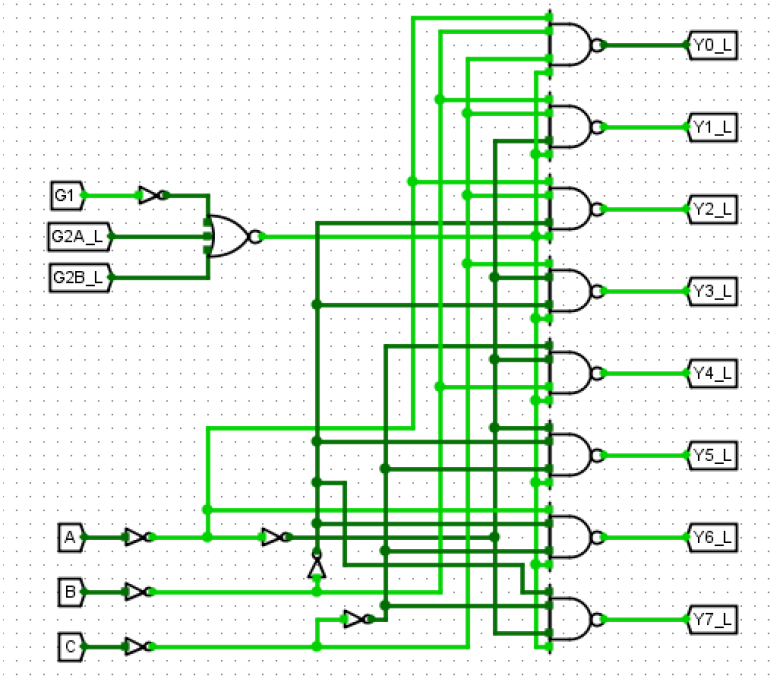
目录

1． 译码器实验

1.1 分析需求

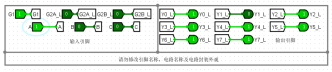
根据图示的 74x138 的电路图和所给的原件设计一个反相逻辑门电路构成的 3-8 译码器。

1.2 Logisim，启动!

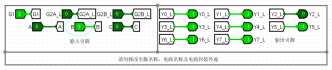


1.3 测试

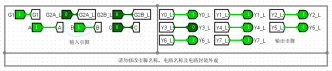
通过仿真测试，观察输出，发现均满足要求



(a) 图 1



(b) 图 2



(c) 图 3

图 1: 测试结果

1.4 功能表

表 12.1 74X138 功能表

输入						输出							
G1	G2A_L	G2B_L	C	B	A	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	1	X	X	X	1	1	1	1	1	1	1	1
0	X	X	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

1.5 总结与心得

本实验的线路较多，要注意小心错误连接和眼花，可以按一定的顺序依次连接不容易出错。

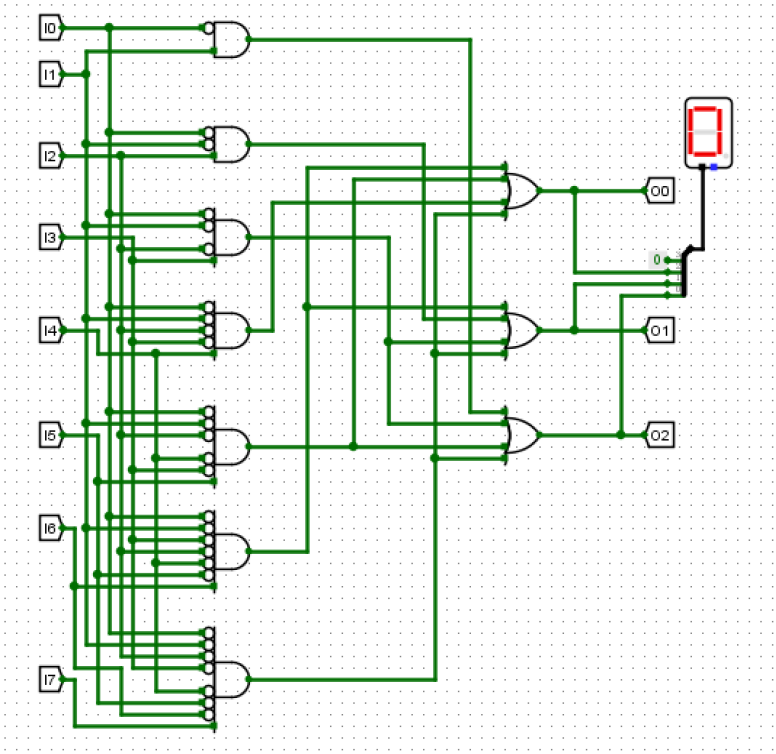
2． 编码器实验

2.1 分析需求

根据实验要求给出的 8-3 优先级编码器原理图，设计一个由逻辑门电路构成的 8-3 优先级编码器。

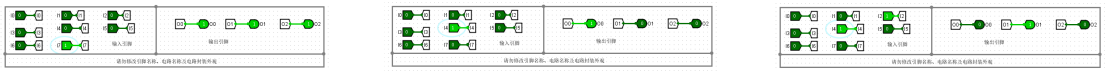
2.2 Logisim，启动!

、按要求连接好电路



2.3 仿真测试

测试结果符合优先级编码器的要求



(a) 图 1　(b) 图 2　(c) 图 3

图 2: 测试结果

2.4 功能表

表 2.2　8-3 优先级编码器功能表

输　入								输　出			
I0	I1	I2	I3	I4	I5	I6	I7	O0	O1	O2	Hex 显示
1	X	X	X	X	X	X	X	0	0	0	0
0	1	X	X	X	X	X	X	0	0	1	1
0	0	1	X	X	X	X	X	0	1	0	2
0	0	0	1	X	X	X	X	0	1	1	3
0	0	0	0	1	X	X	X	1	0	0	4
0	0	0	0	0	1	X	X	1	0	1	5
0	0	0	0	0	0	1	X	1	1	0	6
0	0	0	0	0	0	0	1	1	1	1	7

2.5 总结与心得

注意在给 Hex 设置时需要在最高位设置常量 0。

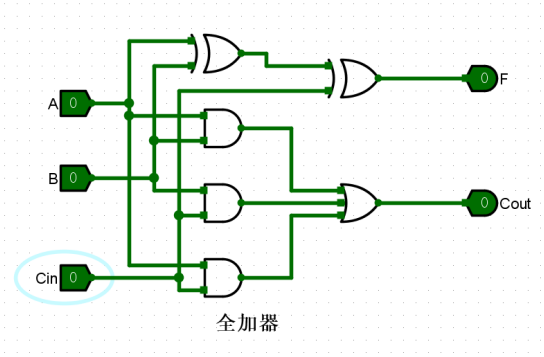
3． 加减法器实验

3.1 分析需求

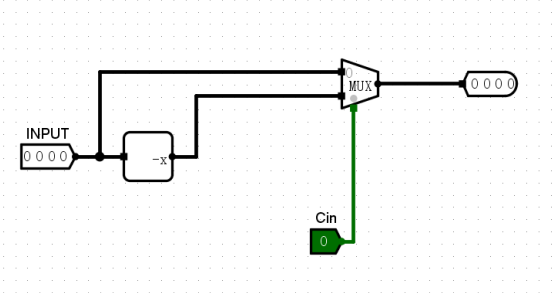
4 位二进制数补码减法运算实验。根据 Cin 输入值区分加减法运算，当 Cin=0 时，执行补码加法运算 F=X+Y；当 Cin=1 时，执行补码减法运算 F=X-Y。

3.2 Logisim，启动！

全加器的实现

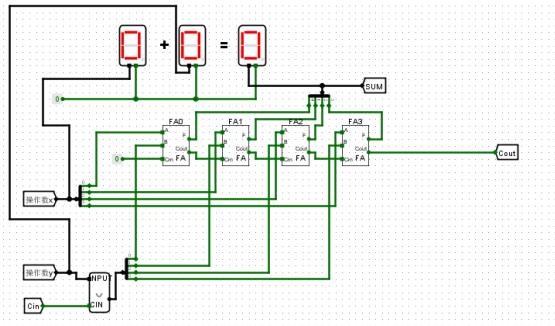


补码运算器的实现



四位加减法器的实现

这里我们使用了补码运算器和一个二路选择器对 Y 的输入进行了处理，若 CIN = 1，则使用 Y 的补码进行运算（相当于做减法），否则使用原码。



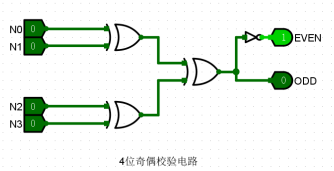
4． 汉明码校验电路

4.1 汉明码相关知识

校验位为 2 的 k 次幂,k 为正整数。通过校验位来计算出是否发生错误以及错误码的位数并进行纠错。

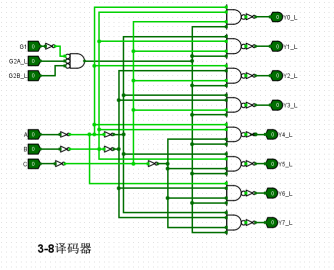
4.2 Logisim，启动!

奇偶校验器的实现

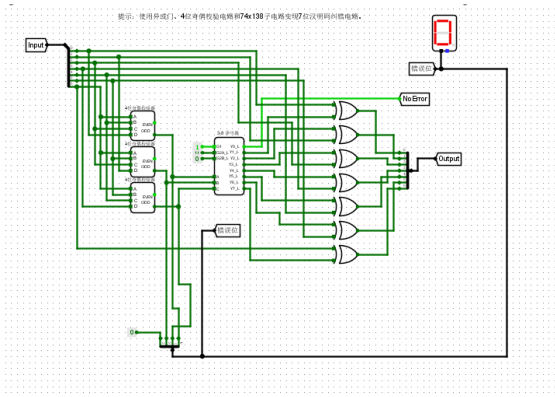


4位奇偶校验电路

3-8 译码器的实现


3-8译码器

完整汉明码校验器的实现



4.3 总结与心得

在花一定的时间了解汉明码的原理和工作规则之后对理解这个实验非常有帮助。注意这里的 3-8 译码器不是前面的 74X138 译码器，需要在输出的时候取反，否则会出现输出结果刚好相反的情况。

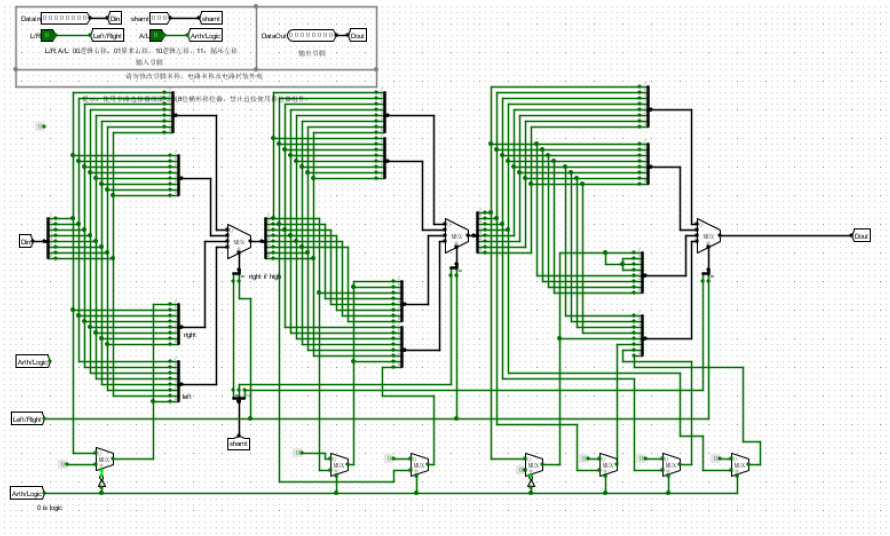
5． 桶形移位器

5.1 关于题目中提到的不同的移位方式

算数左移：空出来的低位补 0; 循环左移：空出来的低位补高位移动溢出的部分; 算数右移：高位补和最高位一样的; 逻辑右移：高位补 0

5.2 Logisim，启动!

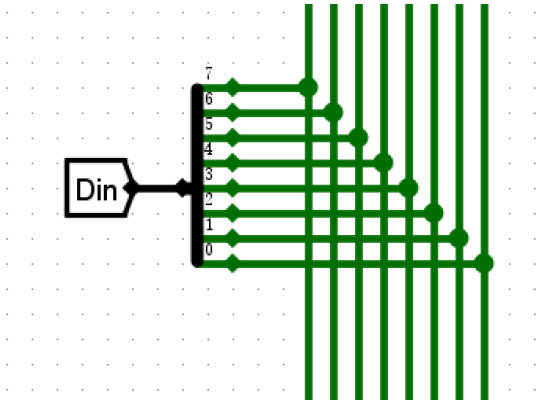
电路的完整实现



5.3 几个我认为有意义的实现细节

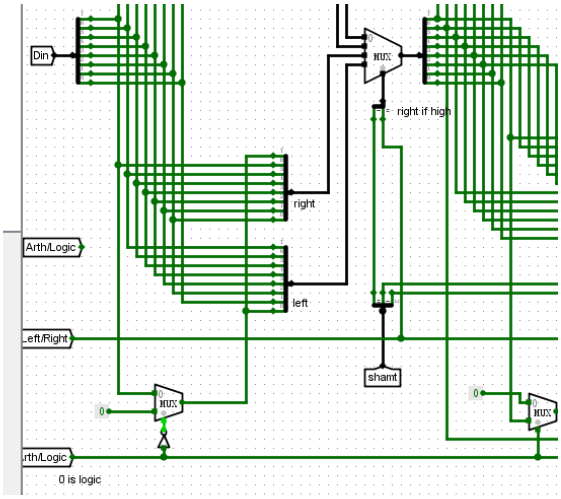
7 位是最高位，0 位是最低位

谁敢相信这个可怜的小女孩因为这个 debug 了一小时呢，怀疑了系统怀疑了测评甚至怀疑了天气偏偏没怀疑自己 TT



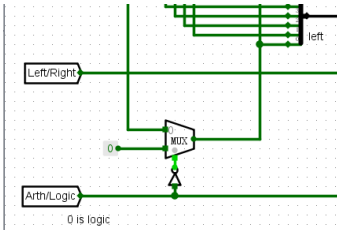
左/右移和移动位数的控制

在四路选择器中，高位连接 shamt，若 shamt 对应位为 1，表示在这个选择器中发生了对应位数的移动；低位连接 L/R，用于控制是左移还是右移。



A/L 的控制

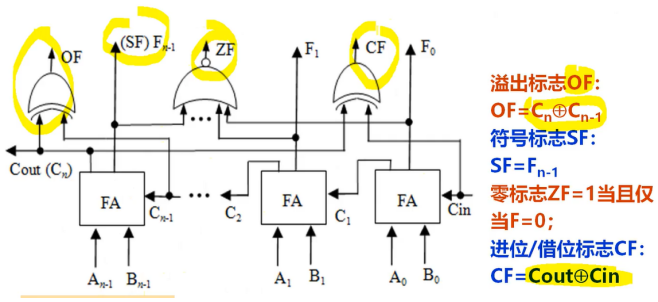
通过二路选择器来选择在对应位上补的数。



6． 思考题

6.1 修改实验中的加法器电路，生成进位标志 CF、溢出标志 OF、符号标志 SF 和结果为零标志位 ZF

从 PPT 中我们可以知道原理图



即符号标志（无符号没有意义）

$$SF = F_{n-1}$$

溢出标志（无符号没有意义）

$$OF = C_{n-1} \oplus C_n$$

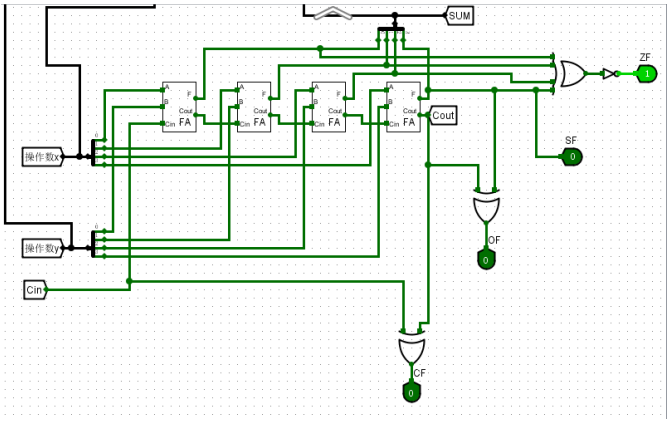
进位标志（带符号没有意义）

$$CF = Cout \oplus Cin$$

零标志

$$ZF = !(F_0 \cdot F_1 \cdot \ldots \cdot F_{n-1}$$

电路实现



6.2 在执行比较指令时，通常使用减法运算后，判断标志位的方式来实现，试通过上述加法器实验举例说明判别的方法

判别的方法：若

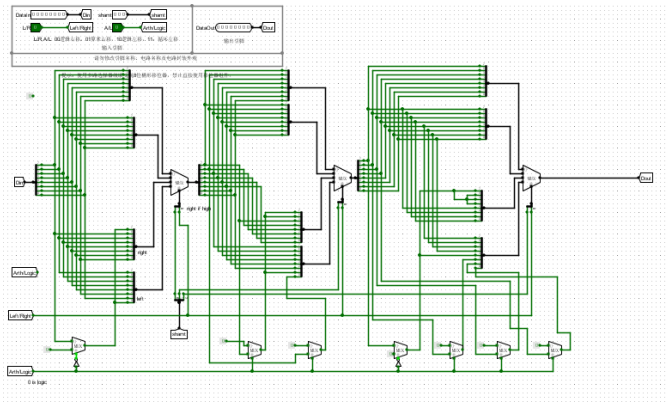
$$ZF == 0$$

则可判断被减数等于减数，这种方法对于有符号数和无符号数都适用。对于有符号数，ZF 不等于 0 时候，若 OF=SF，那么被减数大于减数，否则小于。对于无符号数，CF = 0 表示被减数大于减数，否则小于。

6.3 如何使用逻辑门电路实现 4 位无符号二进制数比较器，并扩展到带符号数的比较，输出相等、大于和小于三个结果。

只需要根据 6.2 中提到的判断方法，对 ZF，OF，SF，CF 进行逻辑运算即可。由于实现较为简单这里不再赘述。

6.4 32 位桶形移位器



我们希望在 8 位桶形移位器的基础上实现 32 位桶形移位器，只需要在 8 位的基础上再做修改修改部分：

- 在第三个 4-1MUX 后方继续增加两个串行的 4-1MUX，并且分别由 shamt 的 3，4 位控制（移动 8 位，移动 16 位）
- 依据 8 位移位器同样的原理，分别用 8 个，16 个 2-1MUX 控制是 A/L
- 将 Cin 和 Cout 都改为 32 位，shamt 改为 5 位