

# 中国科学技术大学计算机学院

## 《数字电路实验》报告



实验题目：简单组合逻辑电路

学生姓名：舒文炫\_\_\_\_\_

学生学号：PB18000029\_\_\_\_\_

完成日期：2021.10.31\_\_\_\_\_

计算机实验教学中心制

2020 年 09 月

## 【实验题目】

简单组合逻辑电路

## 【实验目的】

- 熟练掌握 Logisim 的基本用法
- 进一步熟悉 Logisim 更多功能
- 用 Logisim 设计组合逻辑电路并进行仿真
- 初步学习 Verilog 语法

## 【实验环境】

- Win10 操作系统
- logisim 仿真工具

## 【实验过程】

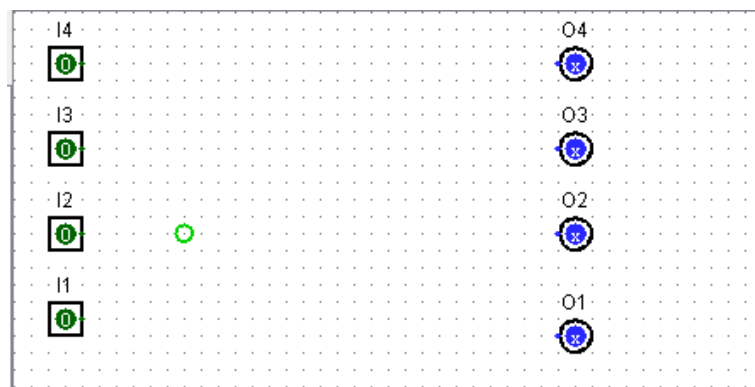
### 0.1 Step1: 用真值表自动生成电路

通过上次实验我们学习了使用 Logisim 管理窗中的各种组件以及自己设计的子电路，设计出各种功能的组合逻辑电路，但是在搭建电路时大量的拖拽、布局、连线非常费时费力，这里学习另一种搭建电路的方法：真值表生成电路。正常步骤如下


- 根据真值表画出各输出项的卡诺图
- 通过卡诺图写出各输出项的逻辑表达式
- 根据逻辑表达式画出电路图，完成电路设计

不过 logisim 工具可以直接帮助我们完成上面大部分工作，下面我按照指导书的步骤一步一步来

首先放置 4 个输入引脚和 4 个输出引脚，分别标号



按照指导书上的真值表输入


Combinational Analysis

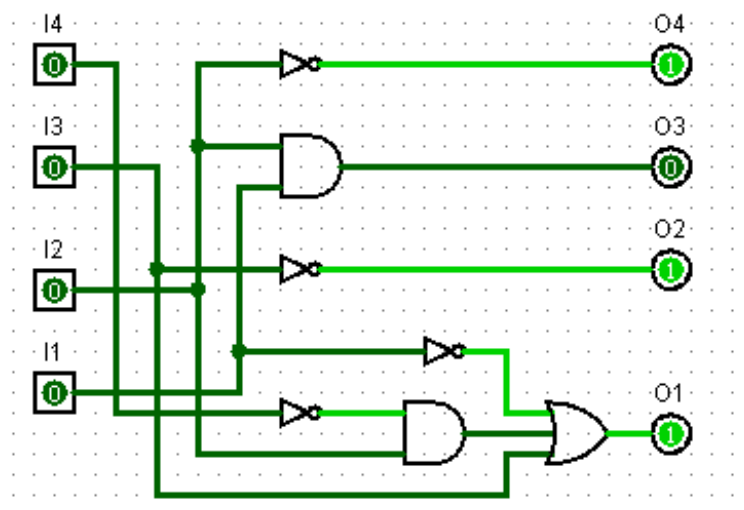
File Edit Project Simulate Window Help

Inputs	Outputs	Table	Expression	Minimized
--------	---------	-------	------------	-----------

I4	I3	I2	I1	O4	O3	O2	O1
0	0	0	0	x	x	x	x
0	0	0	1	1	0	1	0
0	0	1	0	x	x	x	x
0	0	1	1	0	1	1	1
0	1	0	0	x	x	x	x
0	1	0	1	x	x	x	x
0	1	1	0	x	x	x	x
0	1	1	1	x	x	x	x
1	0	0	0	x	x	x	x
1	0	0	1	x	x	x	x
1	0	1	0	0	0	1	1
1	0	1	1	0	1	1	0

Build Circuit

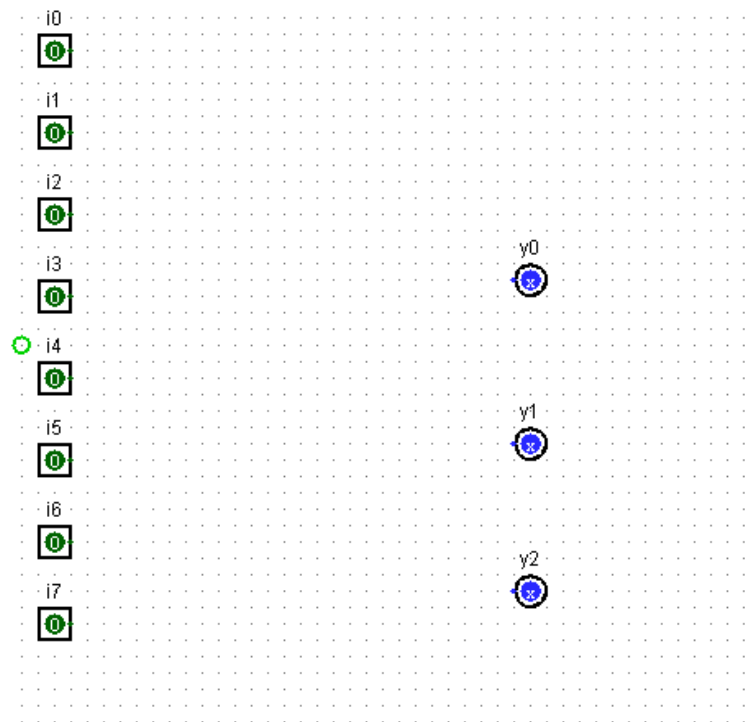
点击 build circuit 然后可以看到电路如图所示



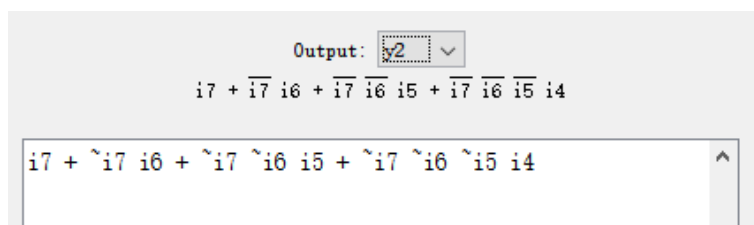
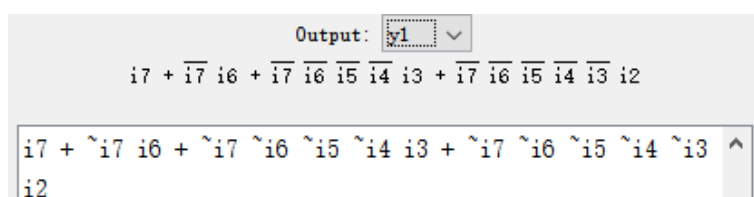
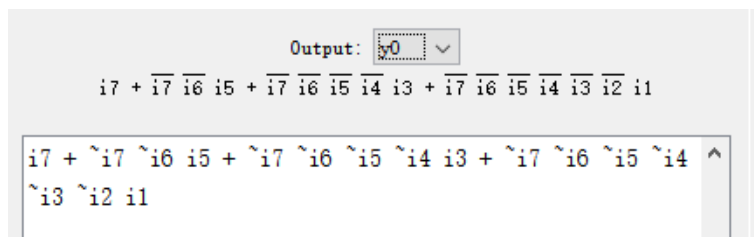
这便是用真值表生成的一个简单组合逻辑电路。

## 0.2 Step2: 用表达式生成电路图

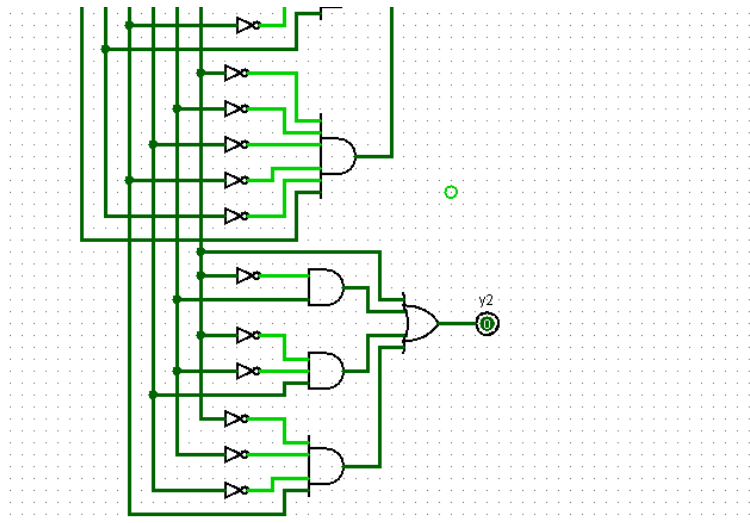
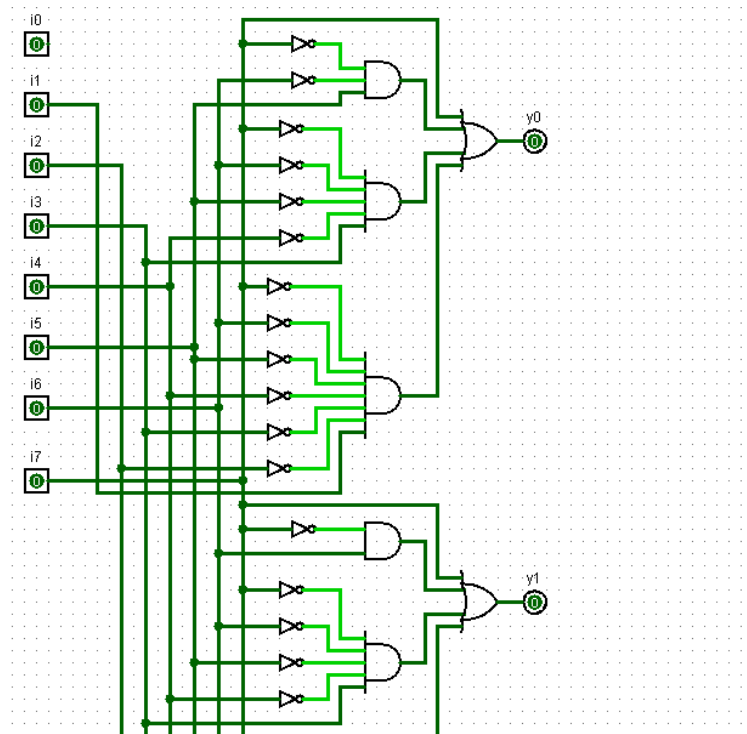
真值表的一个缺陷是真值表条目数与输入项个数呈指数相关,当输入信号数量较多时,编辑真值表也是一项非常繁重的工作,此时如果直接使用表达式来生成会简单很多下面用八位优先编码器为例,有 8 个输入,3 个输出



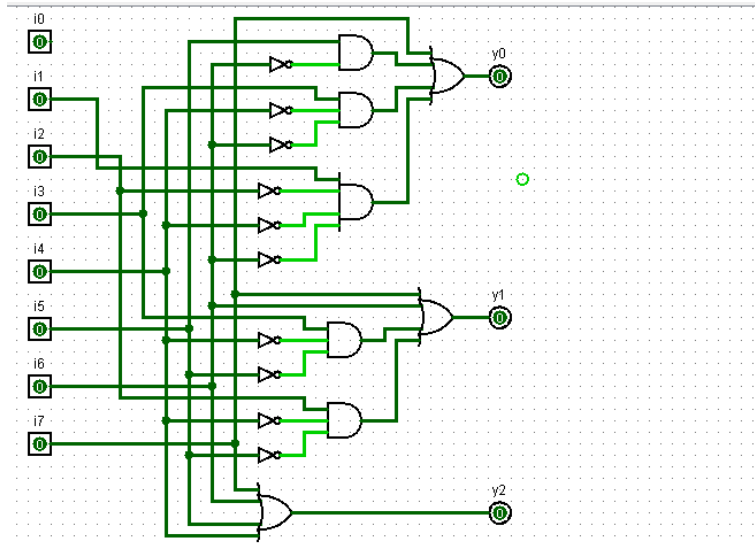
根据真值表，可以写出表达式，logisim 的表达式生成电路，在 analysis circuit 的 expression 选项里面。



然后按下 build circuit 按钮，生成电路如下，这里直接按照表达式生成的电路比较长，分在两个图里面



如果使用了简化电路这个选项，我们可以得到



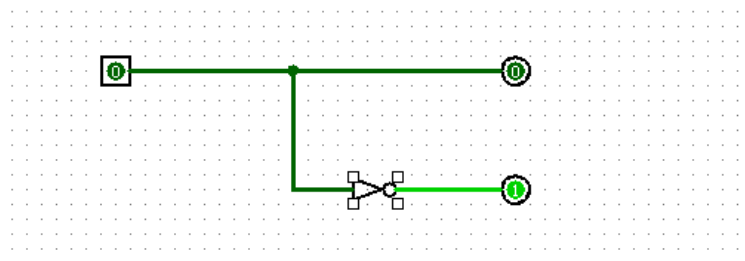
可以发现这里电路变得十分简单。然后 logisim 还有一个很方便的统计电路的功能，可以统计电路中使用的各种门的数量，下面是简化后的电路的统计信息

Logisim: main Statistics				
Component	Library	Simple	Unique	Recu...
Pin	Wiring	11	11	11
NOT Gate	Gates	10	10	10
AND Gate	Gates	5	5	5
OR Gate	Gates	3	3	3
TOTAL (without project's su...		29	29	29
TOTAL (with subcircuits)		29	29	29

Logisim 的自动生成电路功能，能为用户带来便利，节省大量时间，但也有一点小小的不足，其输入输出信号必须是单 bit 位宽，对于多 bit 位宽的输入信号并不支持，需要将其拆分成多个单 bit 信号才可以。

### 0.3 Step3: Verilog HDL 语法入门

这一块进行 Verilog 语法的学习，下图是一个简单的电路



该电路对于的 Verilog 代码如下

## test.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

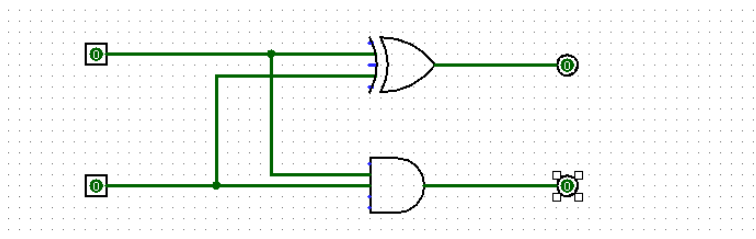
```
module test(  
input in,  
output out,  
output out_n);  
assign out=in;  
assign out_n = ~in;  
endmodule
```

Verilog 最基本的结构是模块，每个模块都是以关键字 module 开头，以 endmodule 结束。module 后面是模块名，括号内是输入输出信号的声明（任何一个有实际功能的电路都应该有输入输出。如果模块功能较复杂的话，可能会用到一些中间信号，那就要在模块内部声明

在本例中用到了一个非常重要的关键字 “assign”，该关键字放在逻辑表达式之前，用于表明后面是一条连续赋值语句

此外，同很多编程语言一样，Verilog 中也可以有注释，单行注释以 “//” 开始，多行注释则使用 “/\* 注释内容 \*/”，注释内容仅仅是为了增加代码可读性，不会对代码功能产生影响

下面这个例子稍微复杂一点，电路图如下，是一个一位半加器



其对应的 Verilog 代码如下

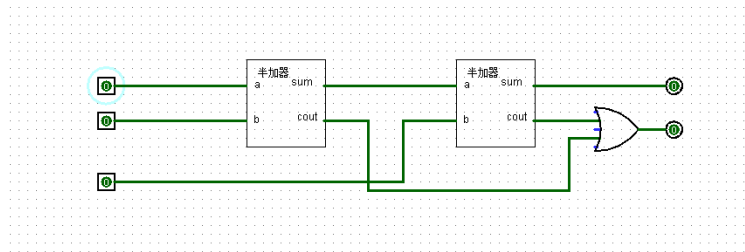
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
module add(  
input a,b,  
output sum,cout);  
assign {cout,sum}=a+b;  
endmodule
```

可以看到这里没有任何和门有关的符号，这里使用了位拼接功能，将  $a+b$  的输出拆成  $sum$  和  $cout$ ，即和数和进位数，这即是电路的行为级描述。这里当然也能直接用 `assign` 描述如下


```
module add(  
  input a,b,  
  output sum,cout);  
  assign cout = a & b;  
  assign sum = a ^ b;  
endmodule
```

下面一个例子涉及模块的调用下图是全加器的电路图，这里使用了前面构造的半加器



对于的 verilog 代码如下



 fulladd.v - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)  

```
module full_add(  
input a,b,cin,  
output sum,count);  
wire s,carry1,carry2;  
add add_inst1(  
.a (a ),  
.b (b ),  
.sum (s ),  
.cout (carry1));  
add add_inst2(  
.a (s ),  
.b (cin ),  
.sum (sum ),  
.cout (carry2));  
assign cout=carry1||carry2;  
endmodule
```

这里就全加器即为 full\_add 模块，里面调用了前面构建的 add 模块。这里 add 模块被实例化了两次，那在最终的电路中就会实实在在的出现两个半加器，它们的行为特性完全一样，只不过各自的输入输出信号不同，工作时也相互独立，互不影响。同时还用到了内部信号声明，关键字 wire 表明声明的信号为线网类型，对于这种信号类型，可以简单的理解为电路中的导线，可以通过 assign 关键字进行赋值的信号都是这种类型，wire 类型是 verilog 中的默认类型，凡是没有明确声明类型的信号，都被当作 wire 类型处理

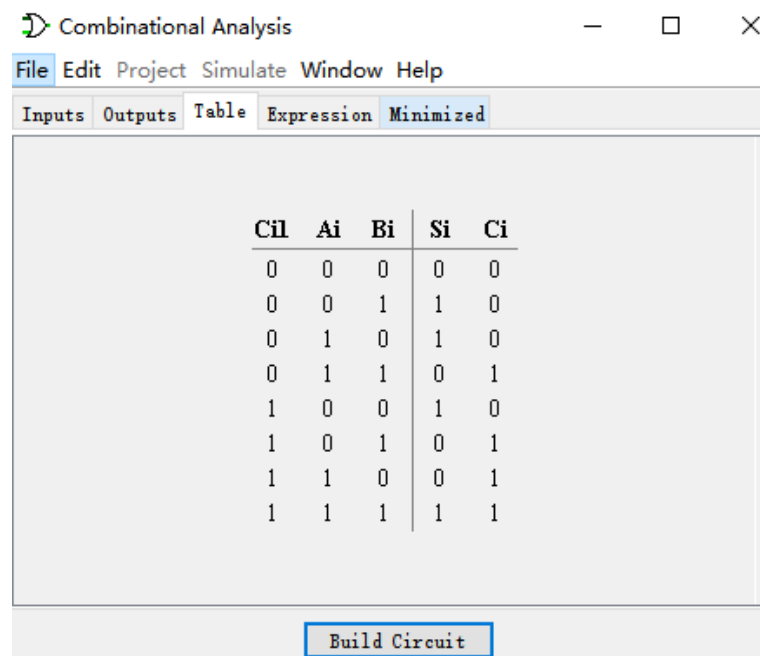
## 【实验练习】

### 0.4 题目 1

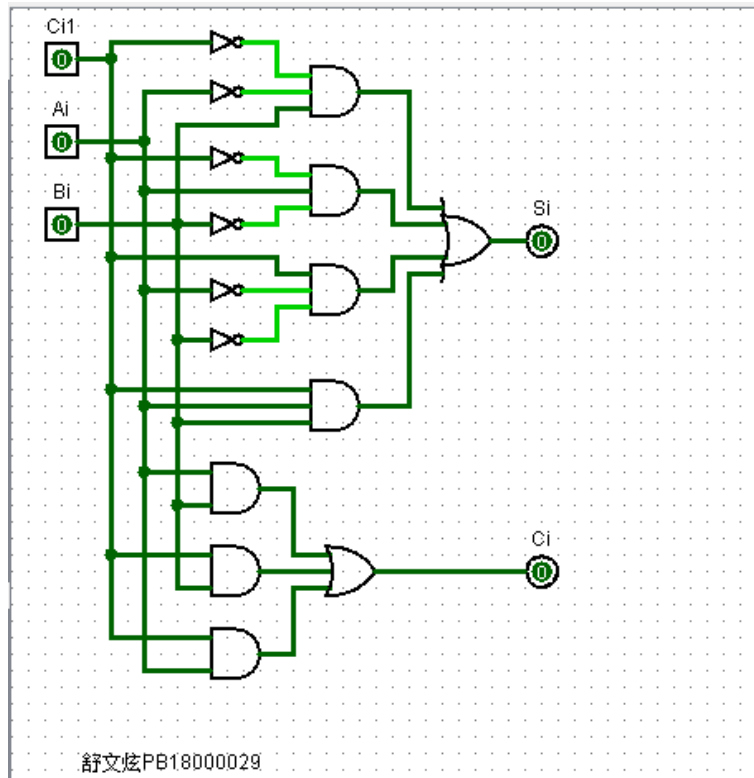
依据如下真值表，通过 Logisim 编辑真值表功能，完成电路设计。电路下方需标注姓名学号。

输入			输出	
Ci-1	Ai	Bi	Si	Ci
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

先放置 3 个输入编号 Ci1, Ai,Bi, 两个输出 Si,Ci 将真值表输入到 logisim 中，如下图



点击 build circuit 可以得到电路图如下



## 0.5 题目 2

根据下列真值表，通过 Logisim 的编辑表达式功能完成电路设计，电路下方需标注姓名学号。

输入						输出							
G1	G2	G3	A2	A1	A0	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

先放置 6 个输入 G1 G3,A2 A0, 8 个输出 Y7 Y0 根据真值表可以得到表达式，我将其整理放在了 txt 文件里面，这里是直接得到的表达式，所以看起来很复杂，截图如下

\*新建文本文档.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

Y0=G2+G3+~G1+G1 ~G2 ~G3 ~A2 ~A1 A0+G1 ~G2 ~G3 ~A2 A1 ~A0+G1 ~G2  
 ~G3 ~A2 A1 A0+G1 ~G2 ~G3 A2 ~A1 ~A0+G1 ~G2 ~G3 A2 ~A1 A0+G1 ~G2 ~G3  
 A2 A1 ~A0+G1 ~G2 ~G3 A2 A1 A0

Y1=G2+G3+~G1+G1 ~G2 ~G3 ~A2 ~A1 ~A0+G1 ~G2 ~G3 ~A2 A1 ~A0+G1 ~G2  
 ~G3 ~A2 A1 A0+G1 ~G2 ~G3 A2 ~A1 ~A0+G1 ~G2 ~G3 A2 ~A1 A0+G1 ~G2 ~G3  
 A2 A1 ~A0+G1 ~G2 ~G3 A2 A1 A0

Y2=G2+G3+~G1+G1 ~G2 ~G3 ~A2 ~A1 ~A0+G1 ~G2 ~G3 ~A2 ~A1 A0+G1 ~G2  
 ~G3 ~A2 A1 A0+G1 ~G2 ~G3 A2 ~A1 ~A0+G1 ~G2 ~G3 A2 ~A1 A0+G1 ~G2 ~G3  
 A2 A1 ~A0+G1 ~G2 ~G3 A2 A1 A0

Y3=G2+G3+~G1+G1 ~G2 ~G3 ~A2 ~A1 ~A0+G1 ~G2 ~G3 ~A2 ~A1 A0+G1 ~G2  
 ~G3 ~A2 A1 ~A0+G1 ~G2 ~G3 A2 ~A1 ~A0+G1 ~G2 ~G3 A2 ~A1 A0+G1 ~G2 ~G3  
 A2 A1 ~A0+G1 ~G2 ~G3 A2 A1 A0

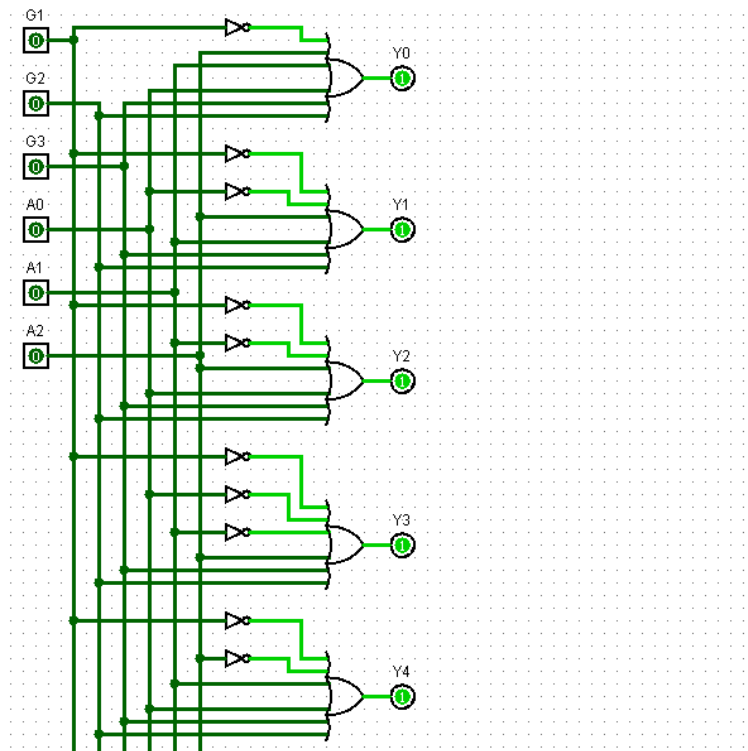
Y4=G2+G3+~G1+G1 ~G2 ~G3 ~A2 ~A1 ~A0+G1 ~G2 ~G3 ~A2 ~A1 A0+G1 ~G2  
 ~G3 ~A2 A1 ~A0+G1 ~G2 ~G3 ~A2 A1 A0+G1 ~G2 ~G3 A2 ~A1 A0+G1 ~G2 ~G3  
 A2 A1 ~A0+G1 ~G2 ~G3 A2 A1 A0

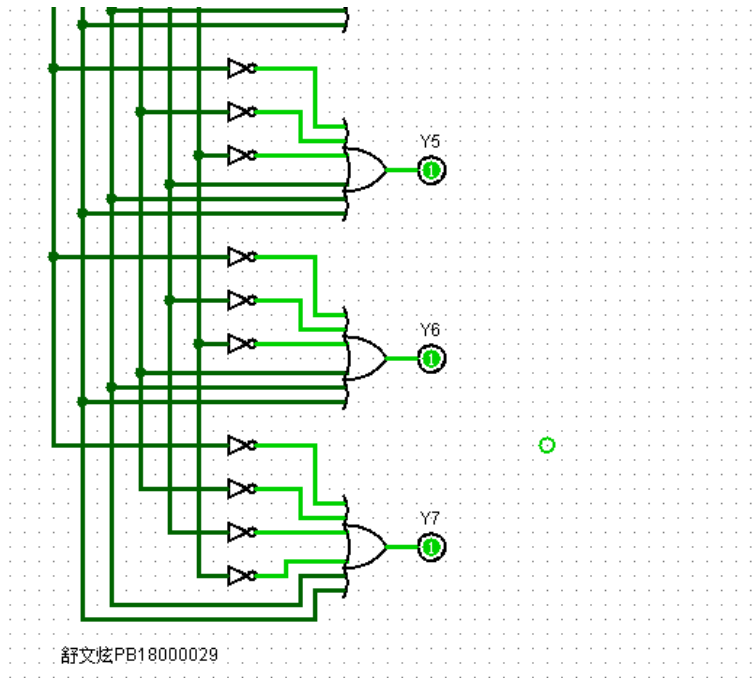
Y5=G2+G3+~G1+G1 ~G2 ~G3 ~A2 ~A1 ~A0+G1 ~G2 ~G3 ~A2 ~A1 A0+G1 ~G2  
 ~G3 ~A2 A1 ~A0+G1 ~G2 ~G3 ~A2 A1 A0+G1 ~G2 ~G3 A2 ~A1 ~A0+G1 ~G2 ~G3  
 A2 A1 ~A0+G1 ~G2 ~G3 A2 A1 A0

Y6=G2+G3+~G1+G1 ~G2 ~G3 ~A2 ~A1 ~A0+G1 ~G2 ~G3 ~A2 ~A1 A0+G1 ~G2  
 ~G3 ~A2 A1 ~A0+G1 ~G2 ~G3 ~A2 A1 A0+G1 ~G2 ~G3 A2 ~A1 ~A0+G1 ~G2 ~G3  
 A2 ~A1 A0+G1 ~G2 ~G3 A2 A1 A0

Y7=G2+G3+~G1+G1 ~G2 ~G3 ~A2 ~A1 ~A0+G1 ~G2 ~G3 ~A2 ~A1 A0+G1 ~G2  
 ~G3 ~A2 A1 ~A0+G1 ~G2 ~G3 ~A2 A1 A0+G1 ~G2 ~G3 A2 ~A1 ~A0+G1 ~G2 ~G3  
 A2 ~A1 A0+G1 ~G2 ~G3 A2 A1 ~A0

将表达式依次输入点击 build circuit 可以得到电路图，这里如果不简化会得到一个巨长的电路图，无法忍受，这里我将其简化，下面是简化后的电路图

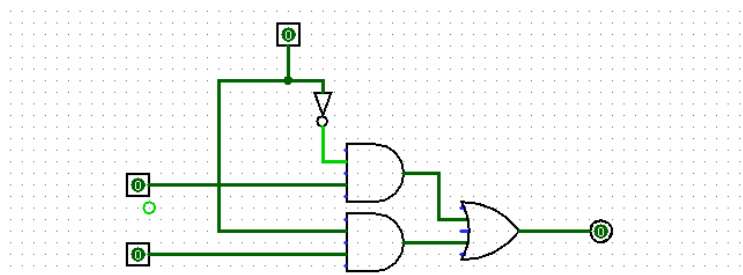




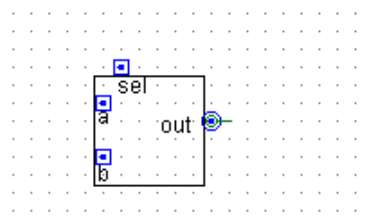
### 0.6 题目 3

使用 Logisim 绘制 1bit 位宽的二选一选择器电路图，并根据生成的电路图编写 Verilog 代码。输入信号为 a,b,sel，输出信号为 out,sel 为 0 时选通 a 信号。

电路图绘制如下



对应封装为



根据该电路图编写 verilog 代码如下

```

module _2sel1(
input a,b,sel,
output out);
assign out=a&&~sel||b&&sel;
endmodule

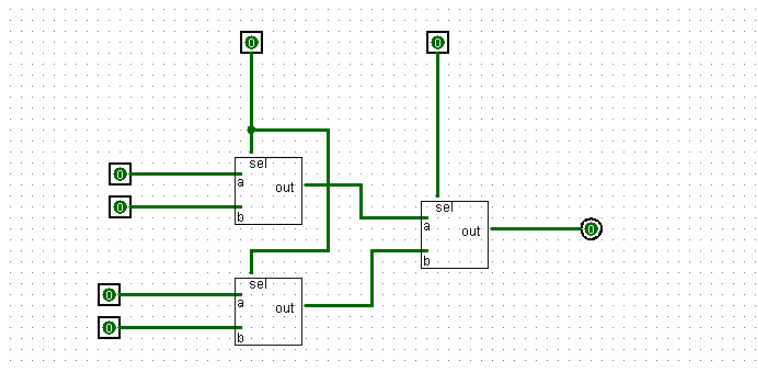
```

我将该模块命名为 \_2sel1

## 0.7 题目 4

通过例化题目 3 中的二选一选择器，用 Verilog 实现一个四选一选择器，并画出对应的电路图。输入信号为 a,b,c,d,sel1,sel0,out，sel1 和 sel0 都为 0 时选中 a 信号。

这里电路图如下



这里调用了前面封装的二选一选择器，根据该电路编写的 verilog 代码如下

#### 4选1选择器.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
module _4sel1(
input a,b,c,d,sel1,sel0,
output out);
wire ab,cd;
  _2sel1 _2sel_inst1(
.a (a ),
.b (b ),
.sel (sel0 ),
.out (ab ));
  _2sel1 _2sel_inst2(
.a (c ),
.b (d ),
.sel (sel0 ),
.out (cd ));
  _2sel1 _2sel_inst3(
.a (ab ),
.b (cd ),
.sel (sel1 ),
.out (out ));
endmodule
```

这里四选一选择器命名为 \_4sel1, 然后里面将 \_2sel1 模块实例化了三次, 前两个实例使用内部变量 ab, cd 表示输出, 选择 a,b 中的一个以及 c,d 中的一个后面将 ab, cd 表示输入, 用内部变量 abcd 表示输出, 选择 ab, cd 中的一个。这样就实现了四选一的功能。

## 0.8 题目 5

根据前面用到的八位优先编码器真值表, 编写 verilog 代码。

这里由于前面以及给出了表达式, 我直接考虑使用 assign 来连续赋值, verilog 代码如下

#### 8位优先编码器.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
module bianma(
input i0,i1,i2,i3,i4,i5,i6,i7,
output y0,y1,y2);
assign y0=i7||~i7&&~i6&&i5||~i7&&~i6&&~i5&&~i4&&i3||
~i7&&~i6&&~i5&&~i4&&~i3&&~i2&&i1;
assign y1=i7 || ~i7&&i6 || ~i7&&~i6&&~i5&&~i4&&i3 || ~i7&&~i6&&~i5&&
~i4&&~i3&&i2;
assign y2=i7 || ~i7&&i6 || ~i7&&~i6&&i5 || ~i7&&~i6&&~i5&&i4;
endmodule
```

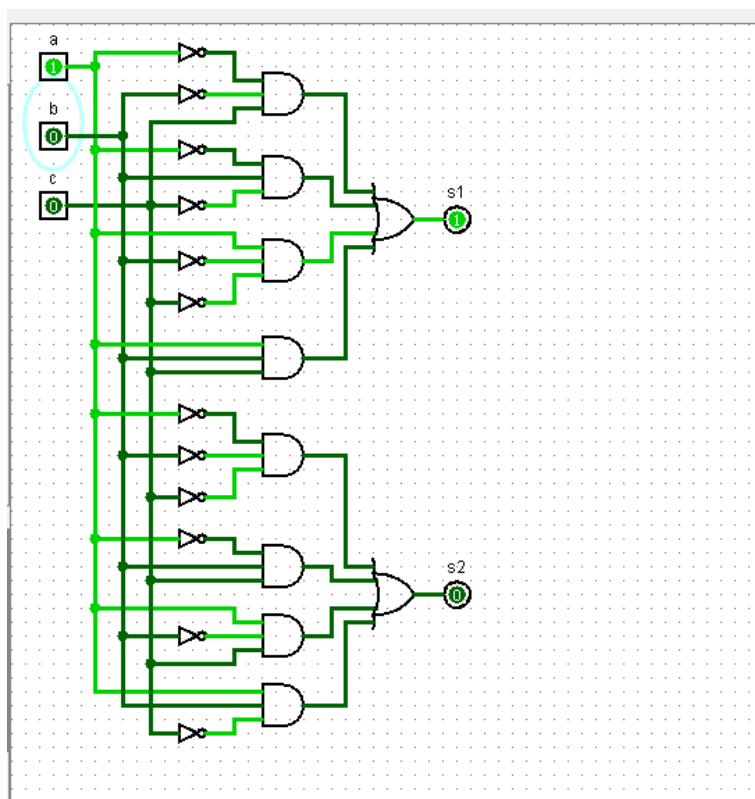
这里注意一下由于输入信号是 1bit 位宽的所以使用按位与 & 和逻辑与 && 都是一样的。

## 0.9 题目 6

阅读如下 Verilog 代码，描述其功能，并画出其对应的电路图。

```
module test(  
    input a, b, c,  
    output s1, s2);  
    assign s1= ~a & ~b & c / ~a & b & ~c / a & ~b & ~c / a & b & c;  
    assign s2= ~a & b & c / a & ~b & c / a & b & ~c / ~a & ~b & ~c;  
endmodule
```

可以观察发现有 s1 在奇数个 1 的时候输出 1，s2 在有偶数个 1 的时候输出 1，该电路可以进行奇偶性检测。电路图如下，这里直接使用表达式来生成电路了



## 【总结与思考】

1. 请总结本次实验的收获

本次实验学习了各种 logisim 生成电路的方法，以及简单的 verilog 语法，然后如何在电路图和 verilog 代码之间转化，收获很大



2. 请评价本次实验的难易程度

本次实验难度适中，就是比较繁琐，题目 2 用表达式生成电路，表达式每个都巨长无比，写起来稍不小心出错还挺麻烦

3. 请评价本次实验的任务量

任务量不是很大，画画电路图倒还行，输入表达式，极大锻炼耐心

4. 请为本次实验提供改进建议

建议不要搞太复杂的表达式，没有任何意义，徒增任务量罢了。