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：_____谭骏飞_____

学生学号：_____PB20061276_____

完成日期：_____2021.10.28_____

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。通过实验进一步巩固数电中通过真值表画逻辑电路和通过卡诺图化简电路的知识,加深对 Verilog 和逻辑电路对应关系的理解，提升编写 Verilog 程序的能力。

【实验环境】

PC 一台,能流畅的连接校园网

Logisim 仿真工具

vlab.ustc.edu.cn

vscode + Verilog 相关插件

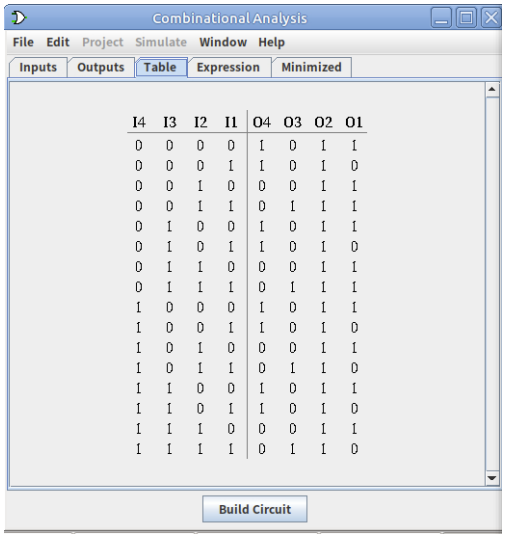
【实验过程】

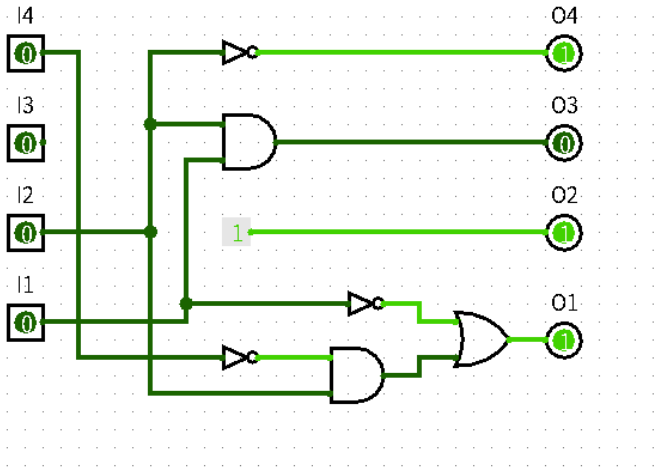
Step1：用真值表自动生成电路

在 Logisim 中新建一个电路图，然后再电路图中放置输入引脚，有几个输入就放几个 引脚，按同样的方式放置输出引脚。放置完毕后，给所有引脚标上标 号，并按高低位顺序排列。在菜单栏的“Project”选项卡中找到“Analyze Circuit”选项， 并选中。在弹出的窗口中选择“Table”选项，按照前面的真值表修 改输出值（鼠标点击输出信号对应的叉号就可修改），最后点击“Build Circuit”便可

生成电路（弹出的对话框都选择“是”）。

实际操作过程中输入的真值表和产生的逻辑电路如下图。

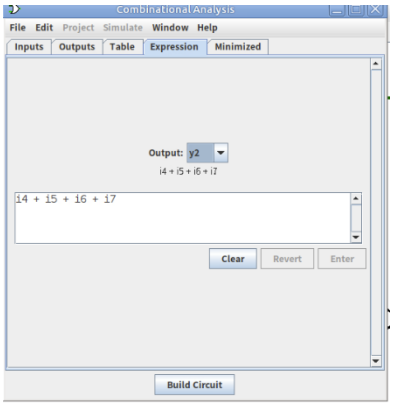


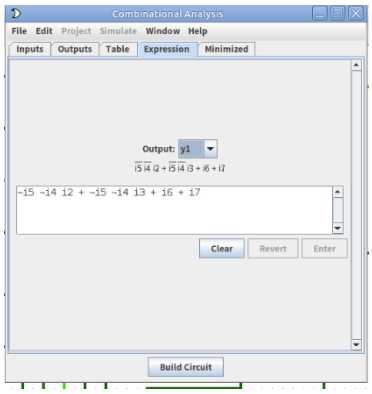


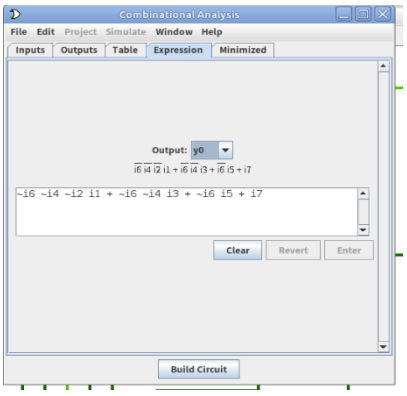
Step2：用表达式生成电路图

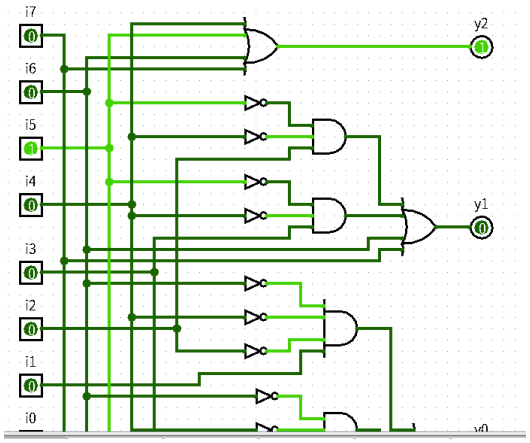
除了通过真值表生成电路，为了避免过长的真值表带来的巨大工作量，Logisim 中也可以通过给出各输出信号的表达式来生成逻辑电路。我们可以在 Logisim 中直接输入表达式生成电路，在“Project” --> “Analyze Circuit”的弹出窗口中选择“Expression”选项， 填入每个输出信号的表达式。最后点击“Build Circuit”生成电路。 有时候手动输入的表达式并不是最简形式，最终生成的电路也会占用 较多的逻辑门，我们可以借助“Minimized”选项卡对表达式进行简 化，进而减少电路使用的逻辑门数量，电路输入信号不多的情况下， 该窗口还能显示卡诺图。

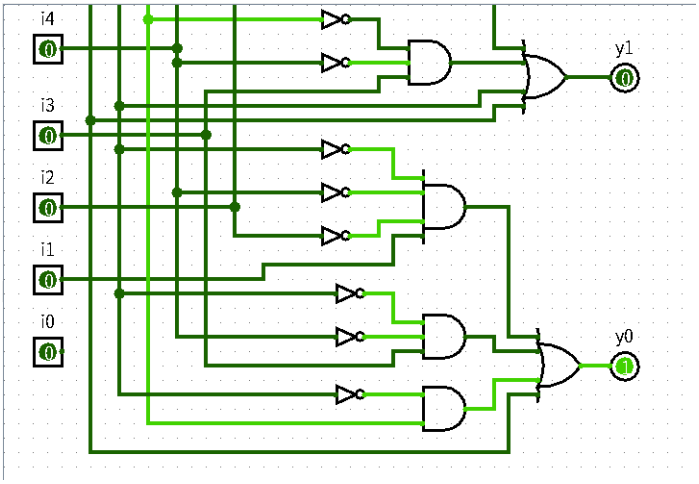
实际操作过程中的表达式和生成的电路如下。



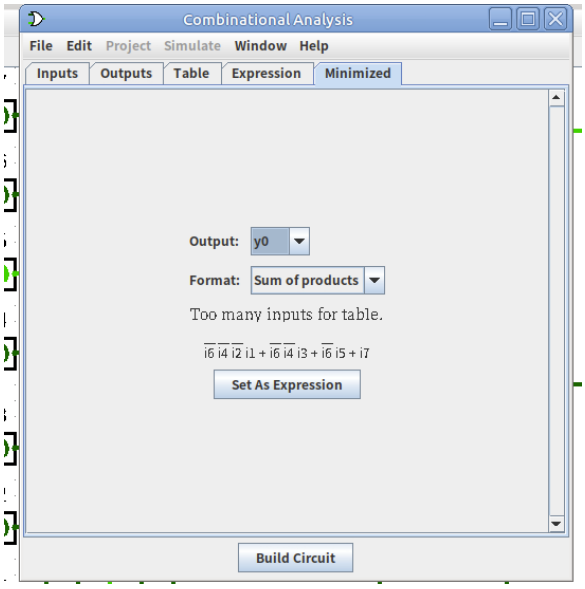






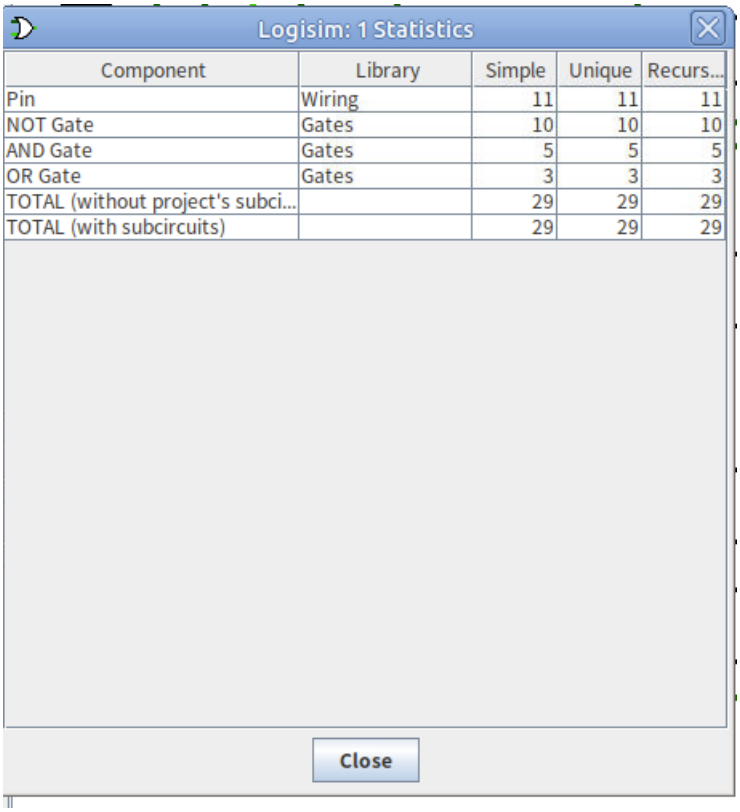


“Minimized”选项卡界面如下。



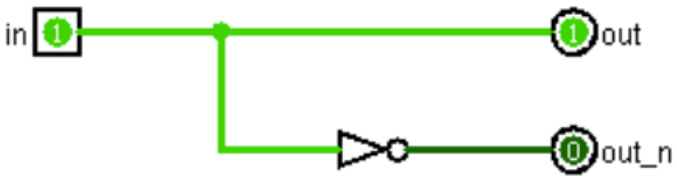
我们还可以通过“Project”--> “Get Circuit Statistics”选 项

统计电路的基本信息。上述图中的电路数据如下。



Step3：Verilog HDL 语法入门

例 1.利用 Verilog 实现下图电路。

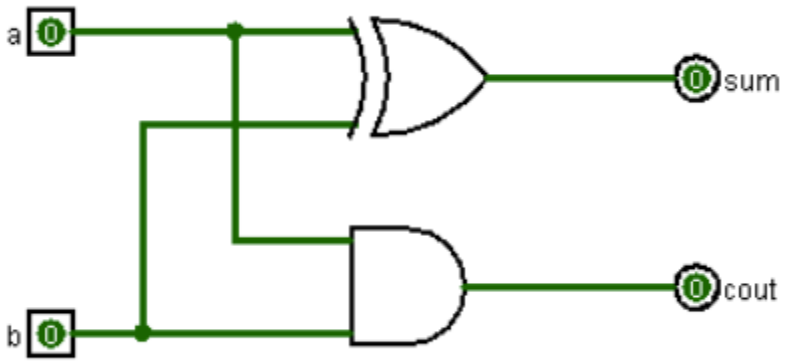


编写的 Veilog 代码如下。（这里用到了关键字“assign”，该关键字放在逻辑表达式之前，用于表明后面是一条连续赋值语句，一般来说，对组合逻辑的赋值都可以使用该关键字实现）

```
module test ( //模块名称
    input in,   //输入信号声明
    output out, //输出信号声明
    output out_n
);
//如需要，可在此处声明内部变量
/*****以下为逻辑描述部分*****/
    assign out = in;
    assign ou_n = ~in;

endmodule
```

例 2.利用 Verilog 实现半加器电路结构。

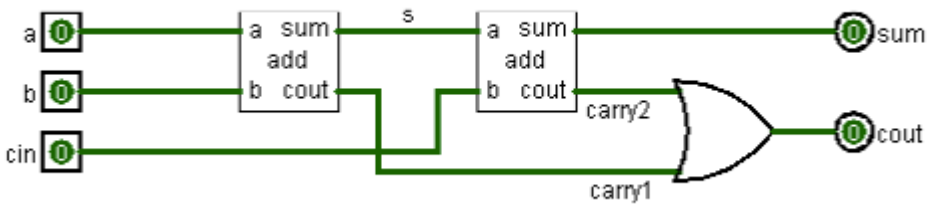


编写的代码如下。（通过使用{}（位拼接符号）将两个单 bit 信号拼接成了一个 2bit 信号，用于接收相加的结果，也可以通过两条连续赋值语句 assign cout = a&b; assign sum = a^b;来实现相同的效果）

```
module add (
    input a,b,
    output sum,cout//cout为向后一位的进位
);
assign {cout,sum}=a+b//拼接，将两个单bit信号拼成了一个2bit信号

endmodule
```

例 3. 利用前面例子中所设计的半加器，构造一个全加器



编写的代码如下。

```
module full_add (
    input a,b,cin,
    output sum,cout
);
wire s,carry1,carry2;
add add_inst1(  //基于名字的端口映射方式
    .a (a ),
    .b (b ),
    .sum (s ),
    .cout(carry1));
add add_inst2(
    .a (s ),
    .b (cin ),
    .sum (sum ),
    .cout(carry2));
assign cout = carry1|carry2;
endmodule
```

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。

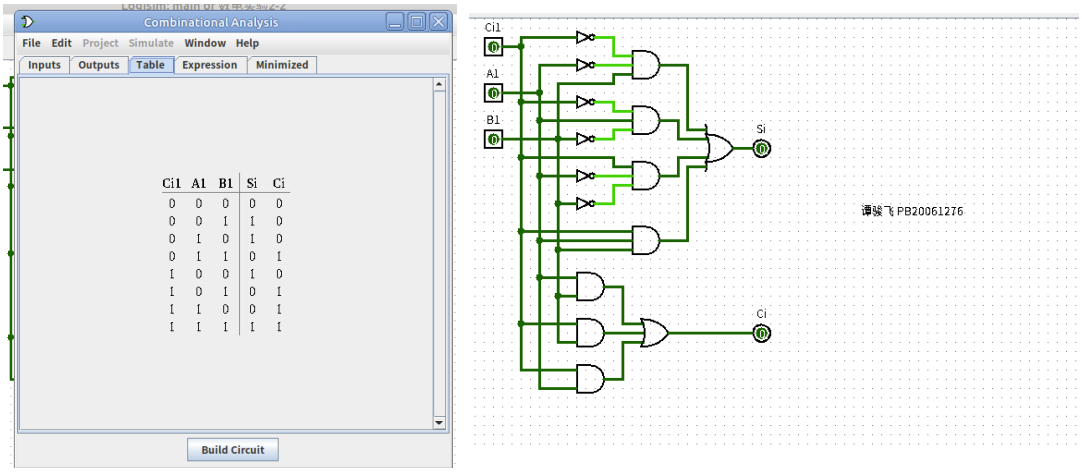
在本例中调用了两个半加器，以实现全加器的功能，其中 add 为被调用模块的模块名，此名称不可随意改动，必须与被调用模块的名称完全一致， add_inst1、add_inst2 为实例化名称，该名称可自行指定。add 模块被实例化了两次，那在最终的电路中就会实实在在的出现两个半加器，它们的行为特性完全一样，只不过各自的输入输出信号不同，工作时也相互独立，互不影响。

【实验练习】

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

通过 Logisim 中“Project->Analyze Circuit->Table”选项生成如下电路。



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

通过 Logisim 中“Project->Analyze Circuit->Expression”选项并利用 Minimized 选项化简后得到的表达式和电路如下。

Output: Y7

$\overline{G1} + \overline{A2} + \overline{A1} + \overline{A0} + G3 + G2$

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

Clear Revert Enter

Output: Y6

$\overline{G1} + \overline{A2} + \overline{A1} + A0 + G3 + G2$

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

Clear Revert Enter

Output: Y5

$\overline{G1} + \overline{A2} + \overline{A0} + A1 + G3 + G2$

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

Clear Revert Enter

Output: Y4

$\overline{G1} + \overline{A2} + A0 + A1 + G3 + G2$

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

Clear Revert Enter

Output: Y3

$\overline{G1} + \overline{A1} + \overline{A0} + A2 + G3 + G2$

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

Clear Revert Enter

Output: Y2

$\overline{G1} + \overline{A1} + A0 + A2 + G3 + G2$

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

Clear Revert Enter

Output: Y1

$\overline{G1} + \overline{A0} + A1 + A2 + G3 + G2$

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

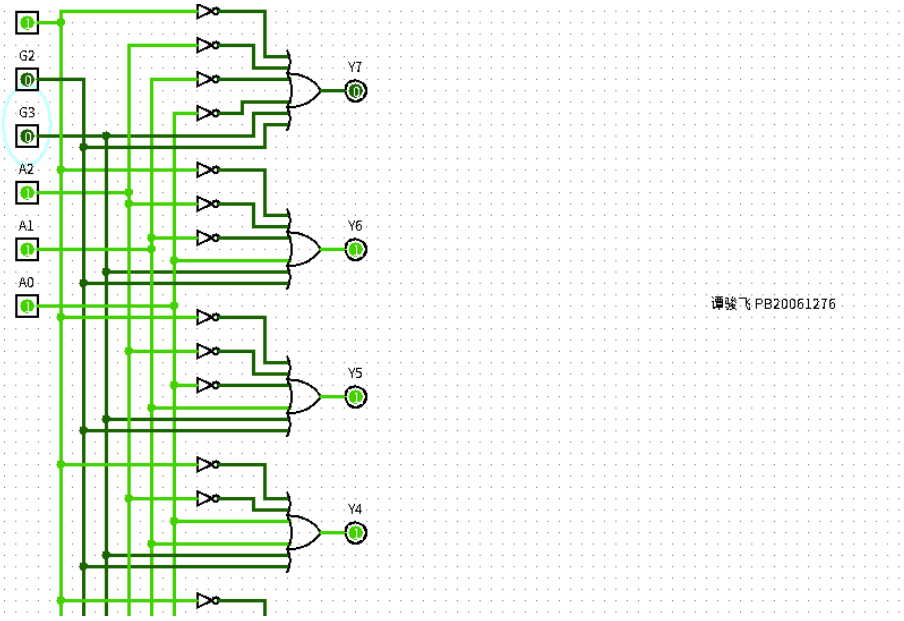
Clear Revert Enter

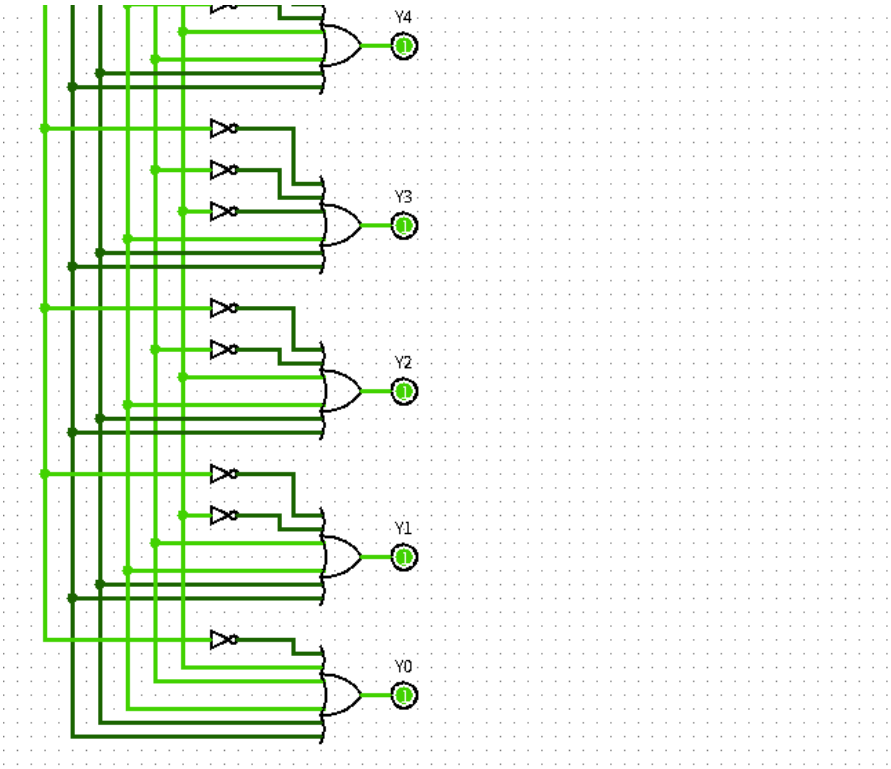
Output: Y0

$\overline{G1} + A0 + A1 + A2 + G3 + G2$

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

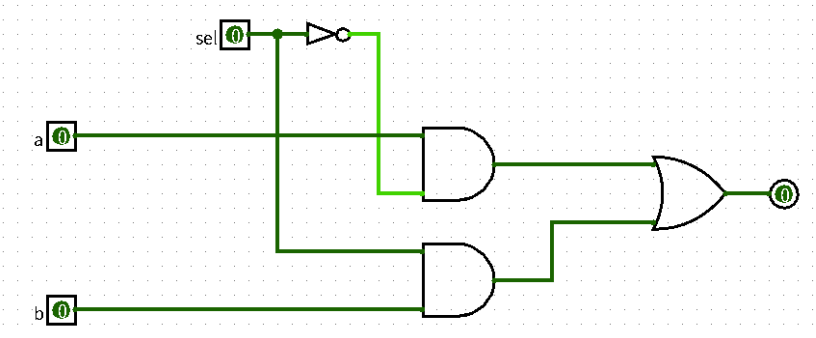
Clear Revert Enter





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。

Logisim 电路图如下。



Verilog 代码如下。

```
module selector (
    input a,b,sel
    output c
);
    assign c = ~sel&a|sel&b;
endmodule
```

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。

对应的代码如下。

```
//4选1数据选择器
module selector4 (
    input a,b,c,d,sel0,sel1;
    output out
);
    wire f1,f2;
    selector sel_inst1(
        .a(a),
        .b(b),
        .sel(sel0)
        .c(f1)
    );
    selector sel_inst2(
        .a(c),
        .b(d),
        .sel(sel0)
        .c(f2)
    );
    selector sel_inst3(
        .a(f1),
        .b(f2),
        .sel(sel1)
        .c(out)
    );
endmodule
```

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。

输入								输出		
i7	i6	i5	i4	i3	i2	i1	i0	y2	y1	y0
1	x	x	x	x	x	x	x	1	1	1
0	1	x	x	x	x	x	x	1	1	0
0	0	1	x	x	x	x	x	1	0	1
0	0	0	1	x	x	x	x	1	0	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	0	0	1	0	0	0

通过真值表计算 y2,y1,y0 和八个输入之间的函数关系，写出了下图中的 Verilog 代码。

数电实验2-3.v

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

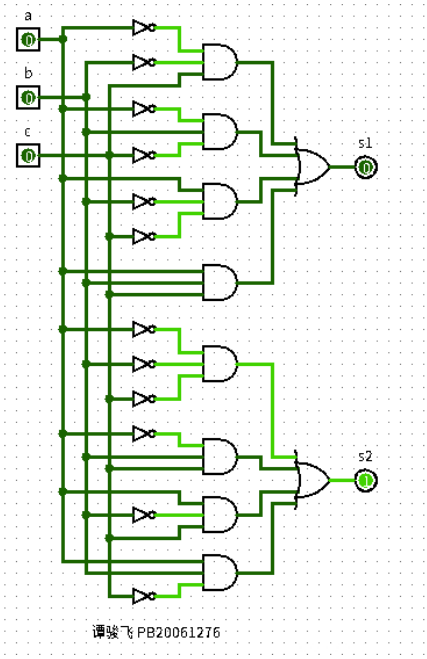
题目 6：阅读如下 Verilog 代码，描述其功能，并画出其对应的电路图。

```
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
```

写真值表利用 Logisim 中“Project->Analyze Circuit->Table”

选项得到的电路如下。（左边为对应的真值表）

a	b	c	s1	s2
0	0	0	0	1
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	0



该电路的功能为：检测 abc 三个输入变量中 0 的个数，若三个输入变量中检测到偶数个 0，则 s1=1，s2=0；若三个输入变量中检测到奇数个 0，则 s1=0，s2=1。

【总结与思考】

本次实验中利用真值表和输入输出的表达式关系在 Logisim 里画电路图的方法大大减少了构建复杂电路时的工作量，提升了画电路图的效率。同时本实验中多次将 Logisim 中的电路图、Verilog 代码和真值表相结合，让原本抽象的代码更加形象具体，便于理解，有助于我更好地掌握 Verilog 语言的语法与实现部分功能的程序编写，提高了我编写 Veriog 程序的能力。