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：王章瀚

学生学号：PB18111697

完成日期：2019/10/8

计算机实验教学中心制

2019 年 09 月

1 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

2 实验环境

PC 一台
Windows 或 Linux 操作系统
Java 运行环境（jre）
Logisim 仿真工具
vlab.ustc.edu.cn (jre 和 Logisim 工具都可在此网站获取)

3 实验过程

3.1 用真值表自动生成电路

由于上一次实验对 Logisim 的操作需要大量的拖拽、布局、连线等操作，费时费力。故这次实验，学习了通过真值表生成电路的方法。
实验指导书中给出了按以下真值表完成本项实验的步骤
。

输入	输出
0001	1010
0011	0111
1010	0011
1011	0110
1111	0101

图 1: 真值表

按照指导书的步骤，填写好真值表：

a	b	c	d	x	y	z	u
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
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	0	1	0	1

图 2: Logisim 中填写的真值表

得到电路图如下：

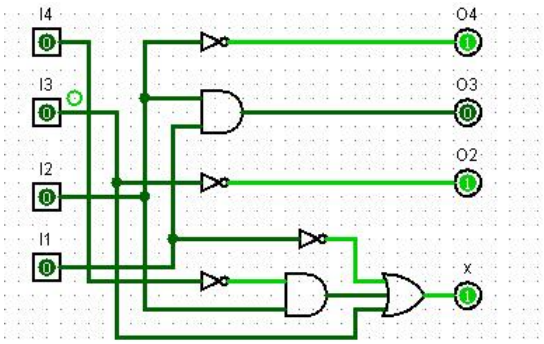


图 3: 通过真值表生成的电路

3.2 用表达式生成电路图

用真值表生成电路图虽然能减少工作，但是一旦输入变量变多，真值表的编辑也将成为十分麻烦的事情。但如果我们可以通过表达式来生成电路，将会使事情变得十分简单。

我们可以在同样的地方选择 Expression 选项，填写每个输出信号的表达式，以此生成电路。

我们甚至可以用 Minimized 选项卡对表达式进行化简，从而减少逻辑门数量。

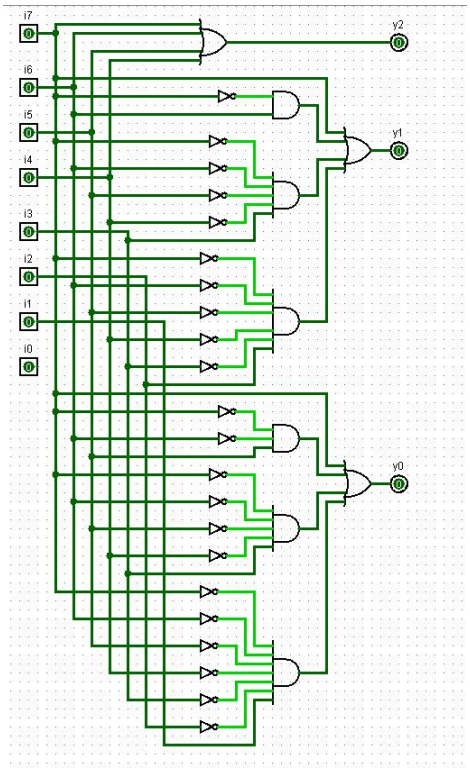


图 4: 未经简化的电路图

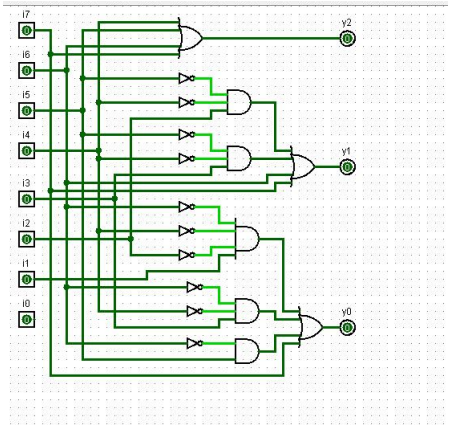


图 5: 经简化的电路图

此外，我们还可以获取电路图的信息。

Logisim: 表达式生成电路-简化 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

图 6: 电路图信息

Logisim 的自动生成电路功能十分方便，但也有一点不足，就是它的输

入输出信号必须是单 bit 位宽的。

3.3 Verilog HDL 语法入门

下面我们通过对照 Logisim 中设计的简单电路来学习 Verilog 语法

例 1.

如下图所示，左侧为电路结构，右侧为电路封装图，可以看到，该电路包含一个输入，取名为 in 信号（信号可自行命名），两个输出，out 信号与输入直连，out_n 信号为输入信号取反，该电路模块我们命名为 test。

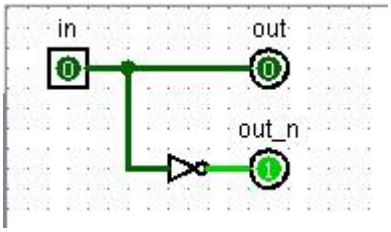


图 7: test 的电路图

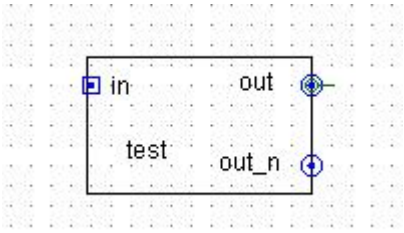


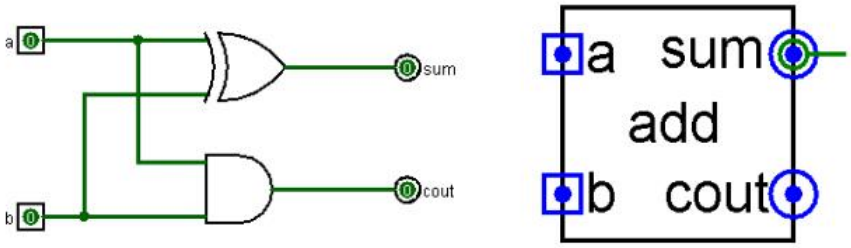
图 8: test 的封装

按此图，写出 Verilog 模块代码如下：

```
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);
//可在此声明内部变量
/*******以下为逻辑描述部分******/
    assign out = in;
    assign out_n = ~in;
/*******逻辑描述部分结束******/
endmodule    //模块名结束关键词
```

例 2.

下图左侧为半加器电路结构图，右侧为电路封装图



按此图，写出 Verilog 模块代码如下：

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

```

这里使用了位拼接功能。其中 a+b 使得 a 和 b 在一起组成了一个两 bit 位宽的信号。而 cout 和 sum 通过位拼接符 {} 将两个单 bit 信号拼接成 2bit 信号，用于接收结果。

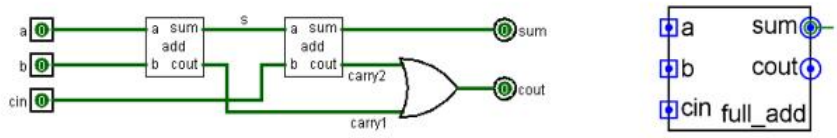
上述代码是从行为级上描述的，效果与下面这段一样。两条赋值语句的位置是不影响结果的。

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

```

例 3.

利用前面例子中所设计的半加器，构造一个全加器，其电路结 构图和电路封装图如下所示



按此图，写出 Verilog 模块代码如下：

```
module full_add(
    input a,b,cin,
    output sum,cout);
    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

```

本例中，关键字 wire 表明声明的信号为线网类型，可以通过 assign 赋值的都是这种类型的信号。凡是没有明确声明类型的变量都被当作为线网类型 wire 来处理。

此外，还利用到了模块调用的功能。

4 实验练习

4.1 题目 1

4.1.1 题目

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

4.1.2 实验结果

实验结果如下图所示：

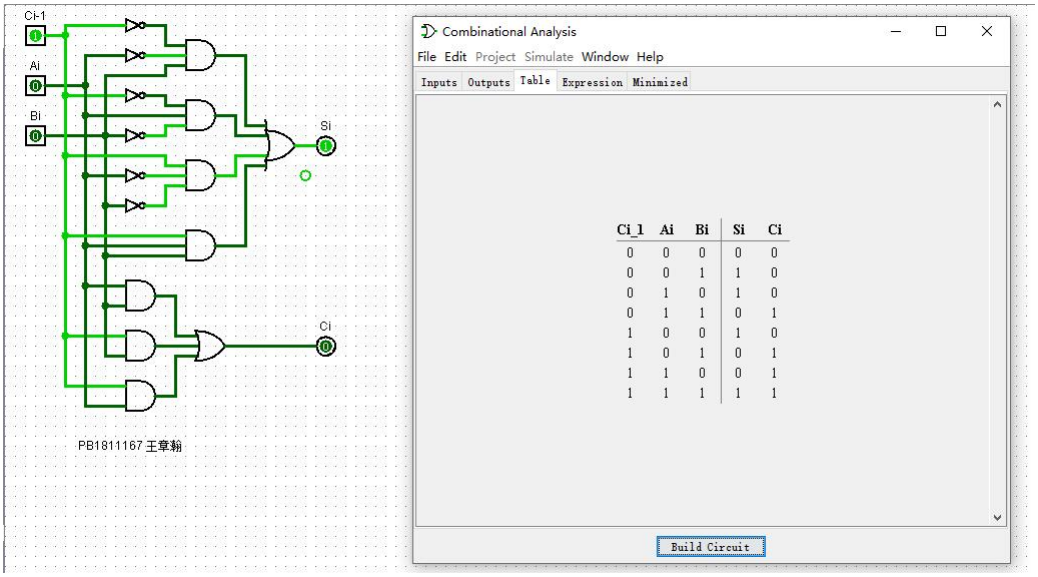


图 9: 题目 1 的实验结果

4.2 题目 2

4.2.1 题目

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

4.2.2 实验结果

实验结果如下图所示：

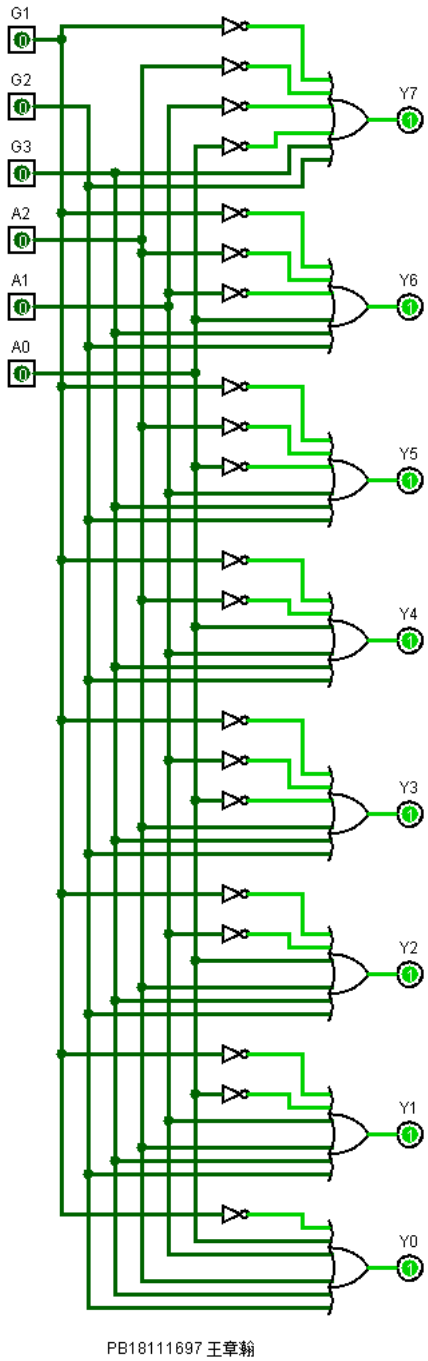


图 10: 题目 2 的实验结果

4.3 题目 3

4.3.1 题目

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。

4.3.2 实验结果

使用 Logisim 绘制的 1bit 位宽二选一选择器电路图如下：

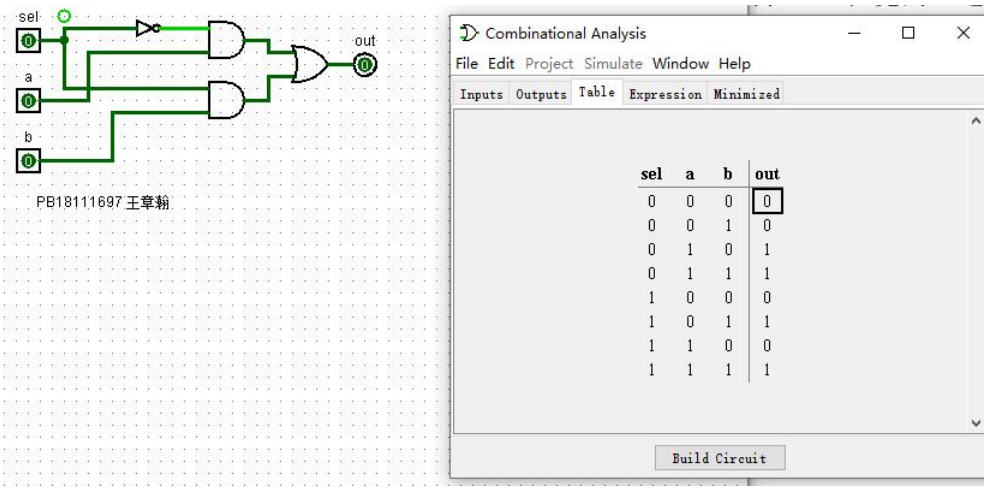


图 11: 题目 3 的二选一选择器电路图

编写的 Verilog 代码如下：

```
module mux2_1bit(
    input a, b, sel,
    output out);
    assign out = (~sel&a)|(sel&b);
endmodule

```

4.4 题目 4

4.4.1 题目

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。

4.4.2 实验结果

编写的 Verilog 代码如下：

```
module mux4_1bit(
    input a,
    input b,
    input c,
    input d,
    input sel1,
    input sel0,
    output out
    );
    wire temp0, temp1;
    mux2_1bit mux2_inst1(
        .a(a),
        .b(b),
        .sel(sel0),
        .out(temp0));
    mux2_1bit mux2_inst2(
        .a(c),
        .b(d),
        .sel(sel0),
        .out(temp1));
    mux2_1bit mux2_inst3(
        .a(temp0),
        .b(temp1),
        .sel(sel1),
        .out(out));
endmodule

```

利用 Vivado 的仿真功能枚举所有情况，得到如下图：

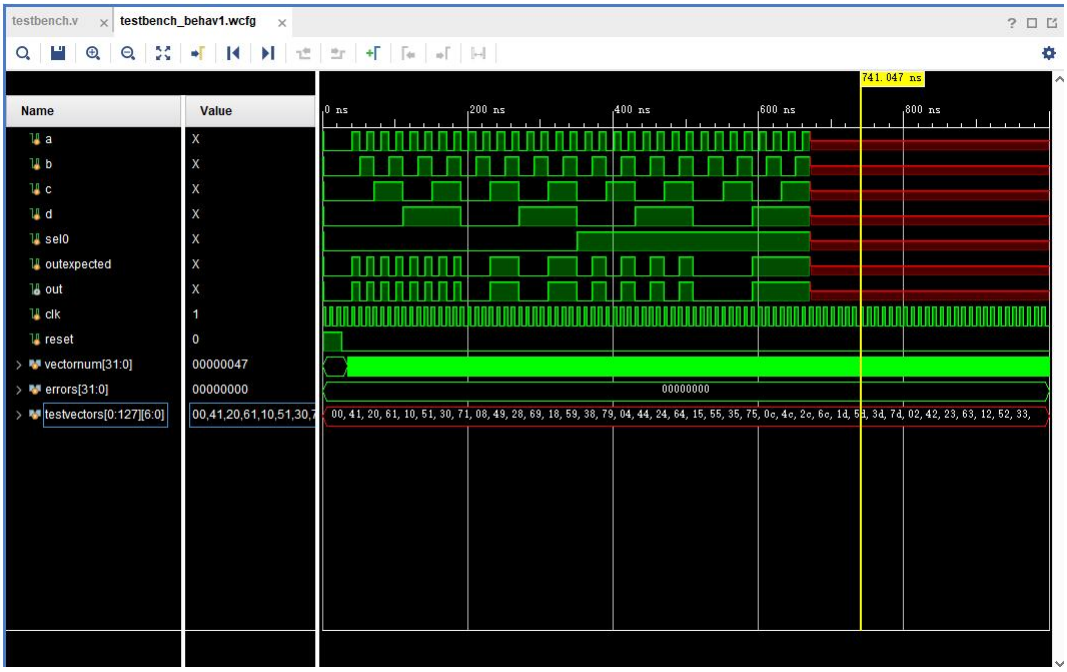


图 12: 题目 4 的 Vivado 仿真结果

这说明该代码是正确的。

此外，其相应的电路图如下：

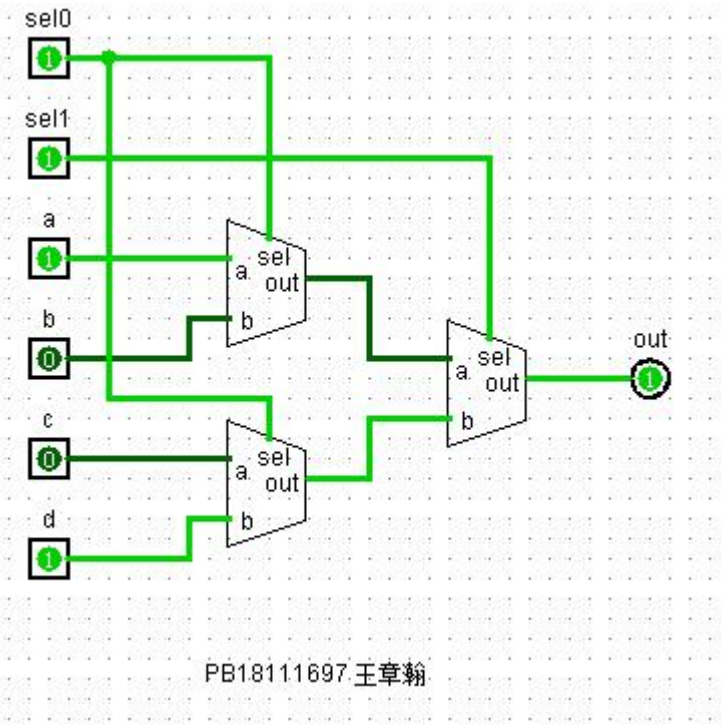


图 13: 题目 4 的四选一选择器电路图

4.5 题目 5

4.5.1 题目

根据前面用到的八位优先编码器真值表，编写 verilog 代码。

4.5.2 实验结果

经过 Logisim 辅助进行简化，得到如下表达式：

$$y0 = \overline{i6}\ \overline{i4}\ \overline{i2}\ i1\ +\ \overline{i6}\ \overline{i4}\ i3\ +\ \overline{i6}\ i5\ +\ i7$$
$$y1 = \overline{i5}\ \overline{i4}\ i2\ +\ \overline{i5}\ \overline{i4}\ i3\ +\ i6\ +\ i7$$
$$y2 = i4\ +\ i5\ +\ i6\ +\ i7$$

据此编写 Verilog 代码如下：

```
1  module encoder(
2      input [7:0] i,
```

```
        output [3:0] y
        );
        assign y[0] = (~i[6] & ~i[4] & ~i[2] & i[1])
            | (~i[6] & ~i[4] & i[3])
            | (~i[6] & i[5])
            | i[7];
        assign y[1] = (~i[5] & ~i[4] & i[2])
            | (~i[5] & ~i[4] & i[3])
            | i[6]
            | i[7];
        assign y[2] = i[4] | i[5] | i[6] | i[7];
    endmodule

```

4.6 题目 6

4.6.1 题目

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

```

4.6.2 实验结果

可以看出来，s1 和 s2 的表达式如下：

$$s1 = \overline{a}\ \overline{b}\ c \ \oplus\ \overline{a}\ b\ \overline{c} \ \oplus\ a\ \overline{b}\ \overline{c} \ \oplus\ a\ b\ c$$
$$s1 = \overline{a}\ b\ c \ \oplus\ a\ \overline{b}\ c \ \oplus\ a\ b\ \overline{c} \ \oplus\ \overline{a}\ \overline{b}\ \overline{c}$$

其电路图如下：

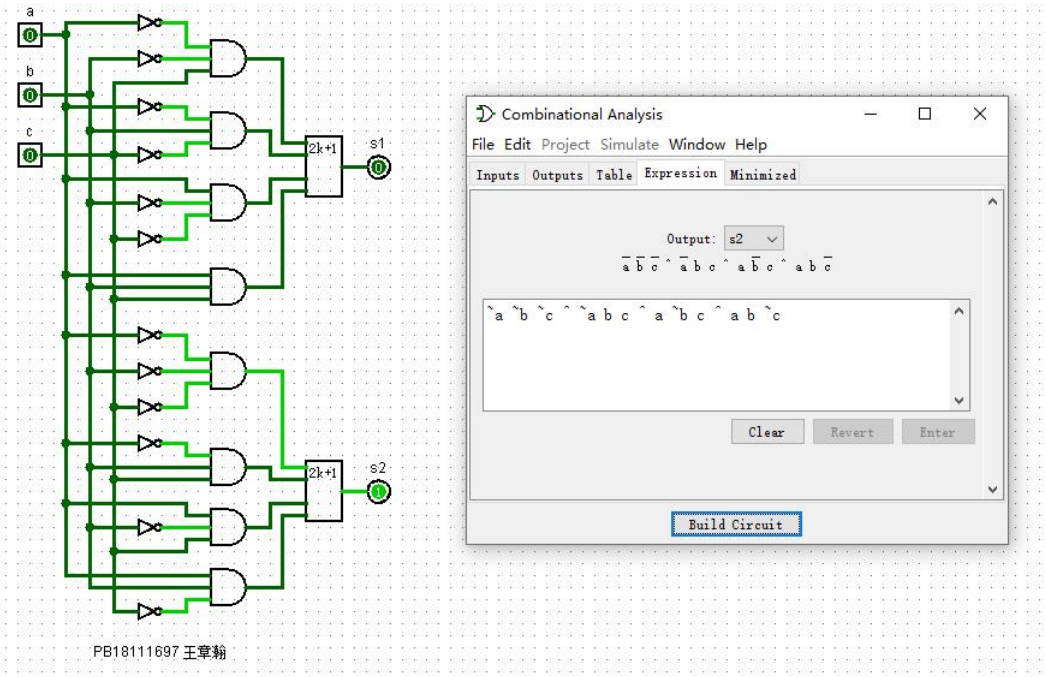


图 14: 题目 6 未简化电路图

借助 Logisim 的表达式化简功能，可以知道上式实际上如下：

$$s1 = \overline{a}\ \overline{b}\ c\ +\ \overline{a}\ b\ \overline{c}\ +\ a\ \overline{b}\ \overline{c}\ +\ a\ b\ c$$
$$s1 = \overline{a}\ b\ c\ +\ a\ \overline{b}\ c\ +\ a\ b\ \overline{c}\ +\ \overline{a}\ \overline{b}\ \overline{c}$$

可以画出电路图：

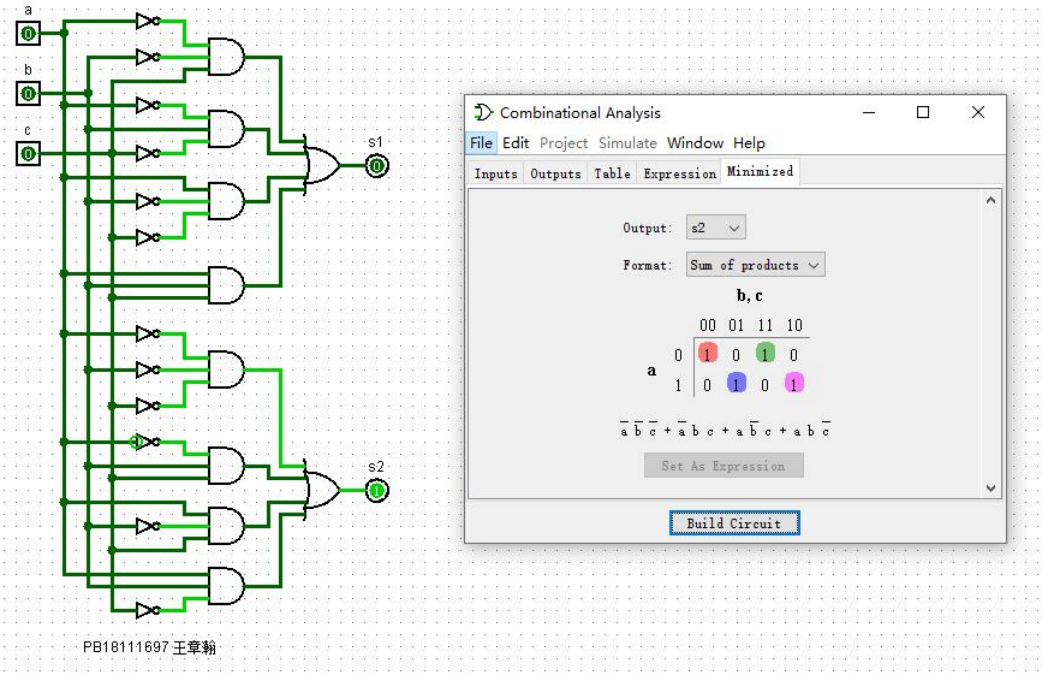


图 15: 题目 6 简化后电路图

由此容易看出来，题述 Verilog 代码实现的功能是统计输入 a,b,c 值为 1 的个数的奇偶性。若为个数为奇数，则 s1=1,s2=0；若为偶数，则 s1=0,s2=1

5 总结与思考

5.1 本次实验的收获

在本次实验中，我进一步了解了 Logisim 的高阶操作。能够使用 Logisim 中利用真值表，表达式来画电路图，并利用相关功能化简表达式，使得利用 Logisim 的效率更高。此外，也初步接触了 Verilog 代码的使用，了解了 Verilog 模块的基本结构与基本使用方法，包括 input,output 的使用，wire 的含义，assign 的使用等。

5.2 评价本次实验的难易程度

本次实验内容难度适中，基本上是可以自主完成的。

5.3 评价本次实验的任务量

本次实验任务量较大，需要课后花费五、六个小时才能完成。但这可能是刚开始接触相关实验还不能较好适应导致的。

5.4 为本次实验提供改进建议

本次实验的实验指导书中有涉及到表达式的内容。其中，模数课程上表达式使用的或所用符号是 +，而 Verilog 中的 + 表示的是加法，对于单 bit 数据来说，相当于一个异或的操作。这一点容易引起不清，还望详述。此外，如能将 Verilog 的介绍放在 Vivado 使用的解说之后，可能效果更好。