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：实验 02_简单组合逻辑电路

学生姓名：张展翔

学生学号：PB20111669

完成日期：2021.11.2

计算机实验教学中心制

2020 年 09 月

【实验题目】

实验 02_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

利用 Verilog 来写一些基本的代码

【实验环境】

PC 一台,

能流畅的连接校园网 Logisim 仿真工具

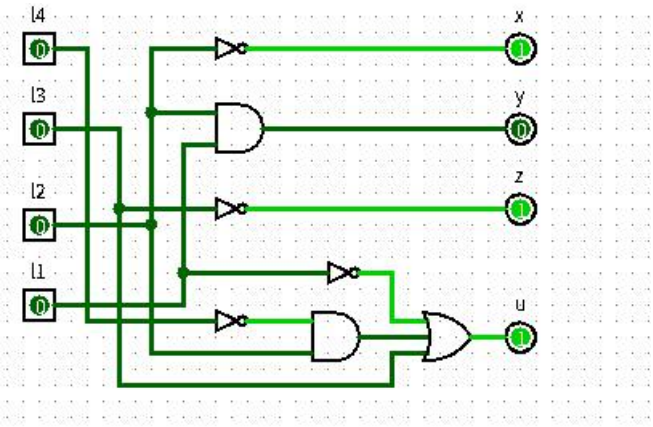
vlab.ustc.edu.cn（jre、Logisim 工具以及 Verilog 语法介绍都可在此网站获取）

【实验过程】

Step1：用真值表自动生成电路

通过 Logisim 的 Analyze Circuit 功能来实现绘制电路图

先分别摆放出 4 个输入和输出并进行编号排列.在利用 Analyze Circuit 中填写真值表,然后即可自动生成如下图所示的电路图



Step2：用表达式生成电路图

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

根据上图的真值表，我们可以很快得出每个输出的表达式

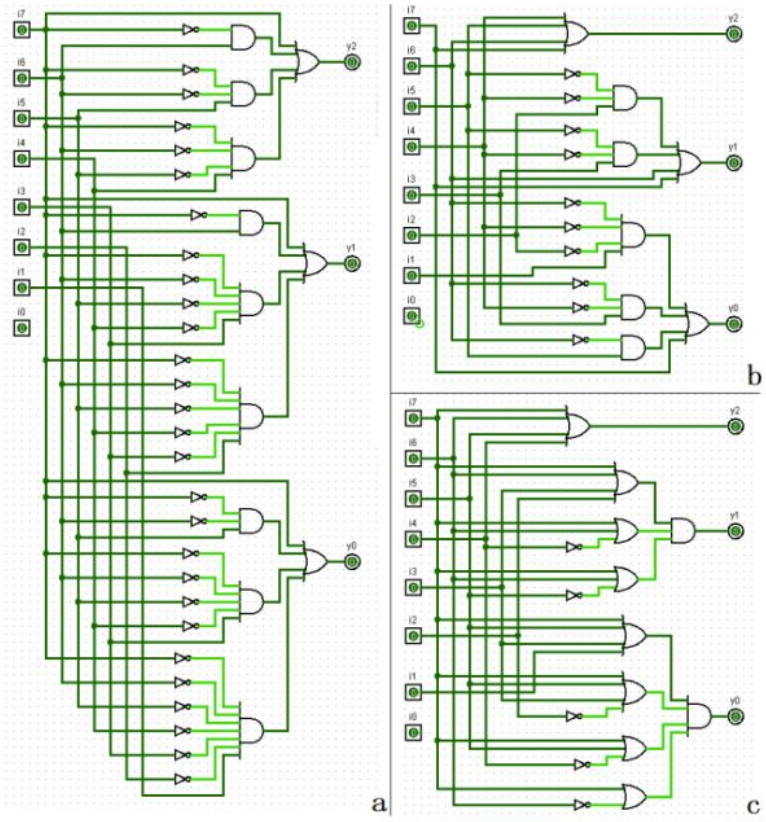
y2 = i7 + ~i7 i6 + ~i7 ~i6 i5 + ~i7 ~i6 ~i5 i4

y1 = i7 + ~i7 i6 + ~i7 ~i6 ~i5 ~i4 i3 + ~i7 ~i6 ~i5 ~i4 ~i3 i2

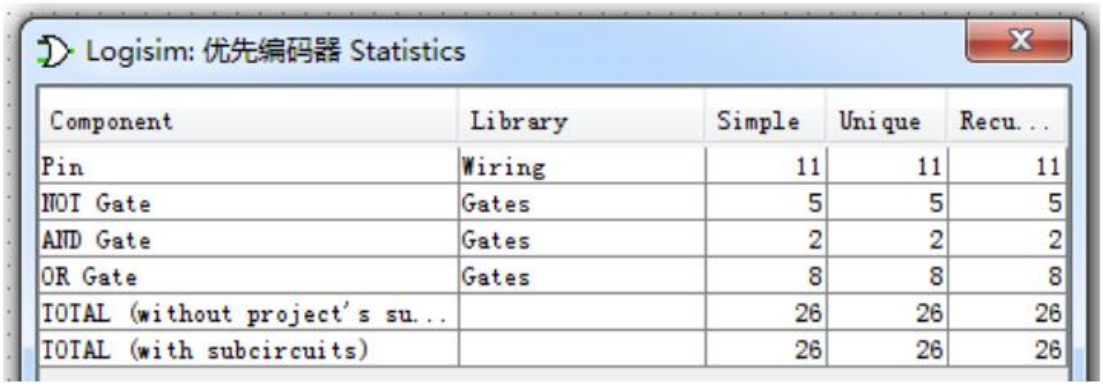
y0 = i7 + ~i7 ~i6 i5 + ~i7 ~i6 ~i5 ~i4 i3 + ~i7 ~i6 ~i5 ~i4 ~i3 ~i2 i1

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。下图对比了同一功能电路未化简的电路结构 和化简后的与或式和或与式结构，可以看出占用逻辑门数量有明显差异。



此外，还可以利用“Project”--> “Get Circuit Statistics”选项统计电路的基本信息。


Logisim: 优先编码器 Statistics

Component	Library	Simple	Unique	Recu...
Pin	Wiring	11	11	11
NOT Gate	Gates	5	5	5
AND Gate	Gates	2	2	2
OR Gate	Gates	8	8	8
TOTAL (without project's su...		26	26	26
TOTAL (with subcircuits)		26	26	26

Step3：Verilog HDL 语法入门

Verilog 模块的最基本结构

module 模块名(

输入端口声明,

输出端口声明);

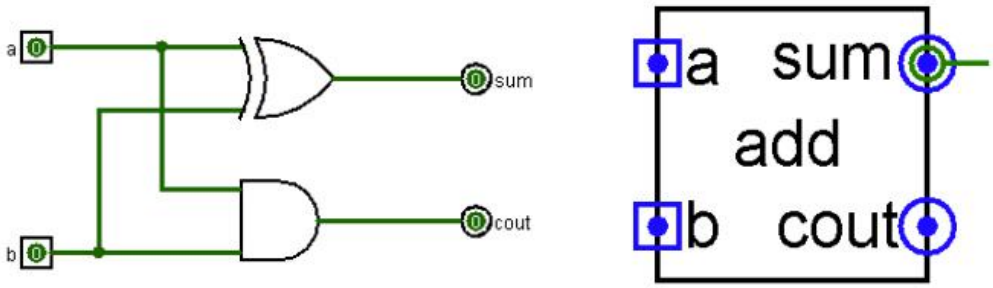
内部信号声明;

逻辑描述（模块主体）

Endmodule

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，但也存在例外，比如后续讲到对电路进行仿真时，其仿真激励文件一般就没有输入输出信号）。 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，此例中没有用到，所以没有声明。 逻辑描述部分是每个模块的主体，用于描述该电路的行为特性， 其语法还算简单，相信读者可以很容易就能看懂。这里用到了一个非 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，一般来说，对组合逻辑的赋值都可以使 用该关键字实现。 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。

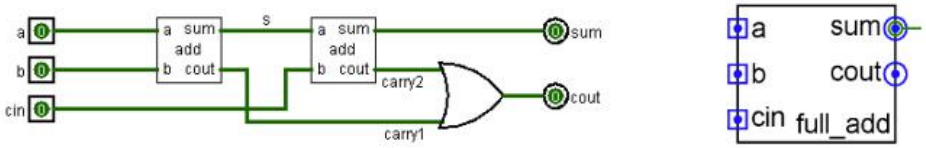
例 2. 下图左侧为半加器电路结构图，右侧为电路封装图



其 Verilog 代码为：

```
module add(
input a,b,
output sum,cout);
assign {cout,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



其 Verilog 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 ),
```

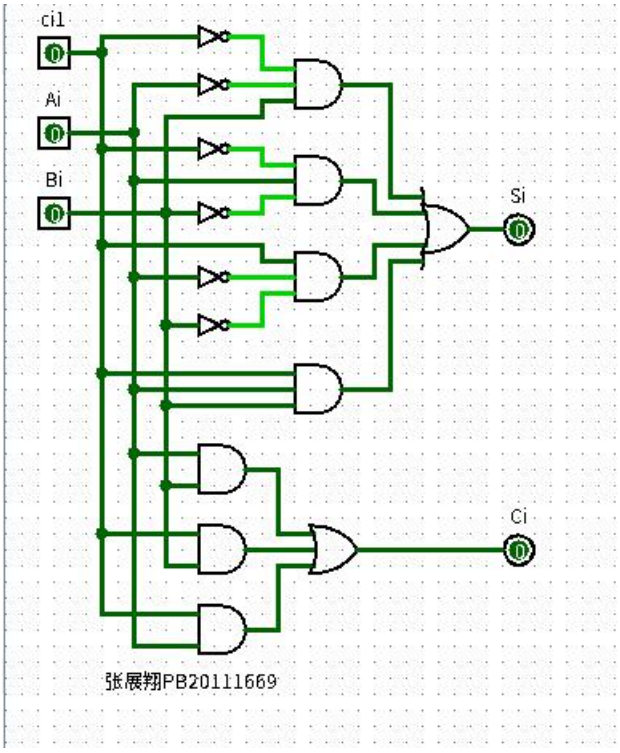
```
 .cout (carry1));
 add add_inst2(
.a (s ),
.b (cin ),
 .sum (sum ),
 .cout (carry2));
assign cout = carry1 | carry2;
 endmodule
```

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。 模块调用在 Verilog 中也非常重要，在电路较复杂时，我们需要 将其分解成若干个子电路，最后再将子电路整合，或者复用第三方以 及自己之前设计的功能模块时，都需要用到模块调用。在本例中，我 们调用了两个半加器，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】

1.

利用 Logisim 的真值表生成功能生成了以下电路图



2. 由下图的真值表可以写出每一个输出的表达式,并填入 Logisim 中,可以转换为最大项或最小项表达式来进行生成.

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

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

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

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

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

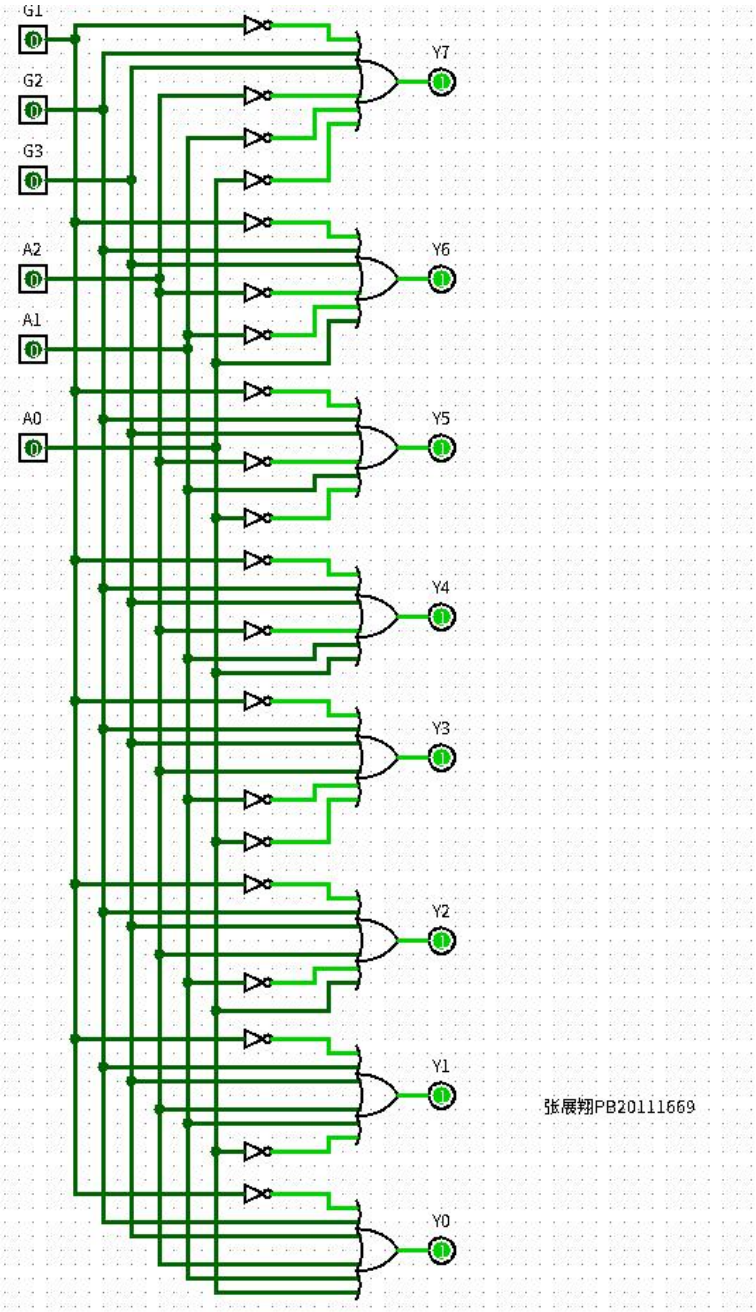
Y3=~G1 + ~A1 + ~A0 + A2 + G3 + G2

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

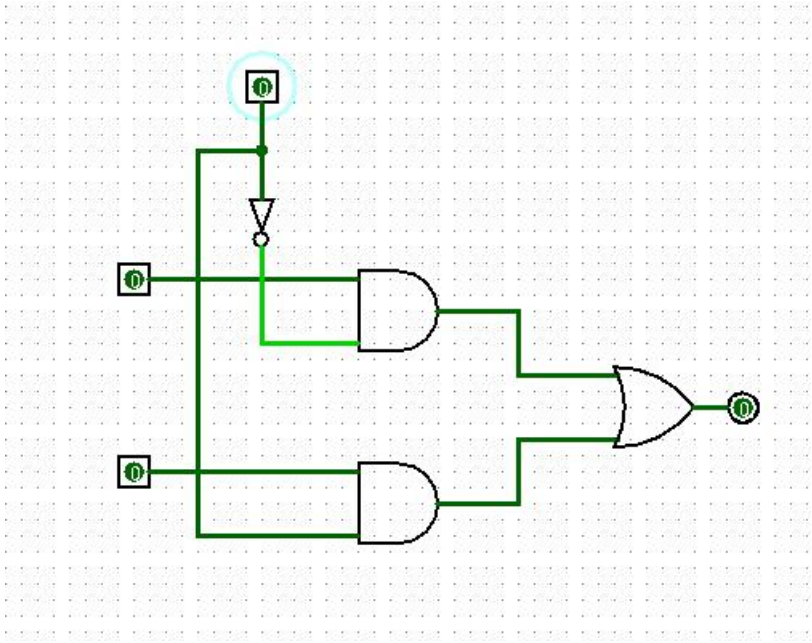
Y1=~G1 + ~A0 + A1 + A2 + G3 + G2

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

将上述表达式填入 Logisim 的每一个输出所对应的表达式栏目中,再点击生成,即可得到以下电路图:

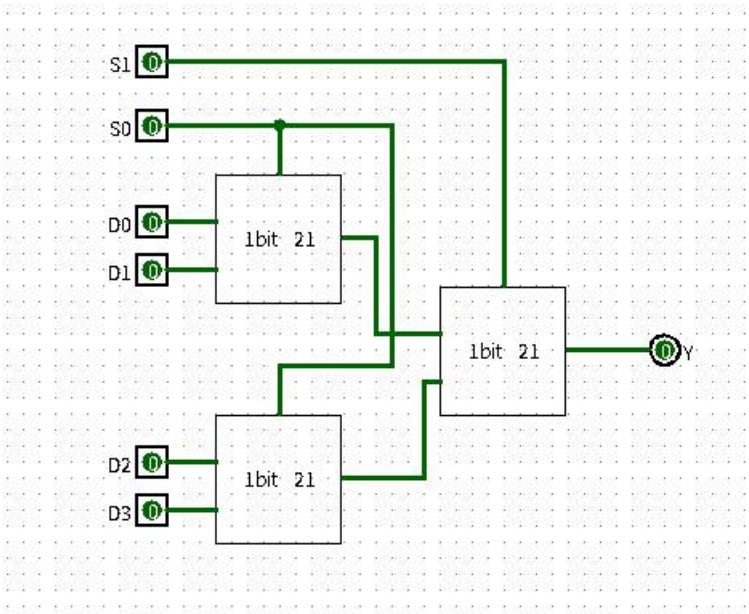


3.通过与.或.非三种门来实现了 1-bit 的二选一选择器,其 Verilog 代码如下所示:



```
module sel_21(input a,b,sel,output out);
    assign out=(a&(~sel))|(b&sel);
    endmodule;
```

4.通过二选一选择器所实现的四选一选择器如下所示:



Verilog 代码如下:

```
module sel_41(input a,b,c,d,sel0,sel1,output out);
    wire o1,o2;
```

```
    sel_21 s1(
    .D0(D0  ),
    .D1(D1  ),
    .S0(S0  ),
    .o1(o1  ));
    sel_21 s2(
    .D2(D2  ),
    .D3(D3  ),
    .S0(S0  ),
    .o2(o2  ));
    sel_21 s1(
    .o1(o1  ),
    .o2(o2  ),
    .S1(S1  ),
    .out(out  ));
    endmodule;
```

5.由下面的真值表可知，该电路图为八位优先译码器，故其Verilog代码如下：

```
    module encode8(input i0,i1,i2,i3,i4,i5,i6,i7, output y0,y1,y2);
        assign y2 = i4 | i5 | i6 | i7;
        assign y1 = ~i5 & ~i4 & i2 | ~i5 & ~i4 & i3 | i6 | i7;
```

assign y0 = ~i6 & ~i4 & ~i2 & i1 | ~i6 & ~i4 & i3 | ~i6 & i5 | i7;

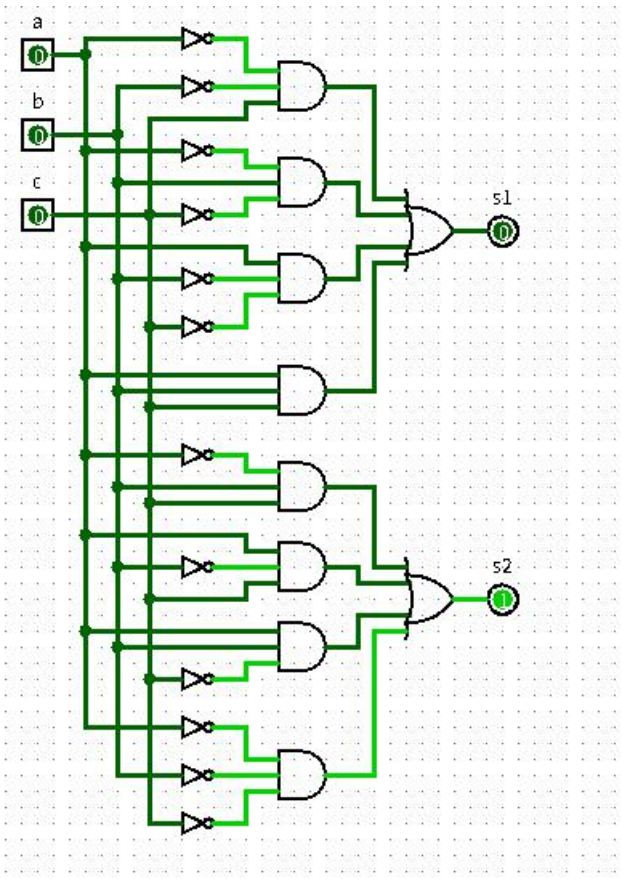
endmodul

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

6.

```
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;
```

上段 Verilog 代码所实现的电路可以由 Logisim 中的 Analyze Circurt 功能来绘制出此电路图如下所示



功能：判断输入端口 1 的个数，若为奇数个，则输出 10，反之则输出 01

【总结与思考】

本次实验任务量适中，进一步的加强了对 Logisim 仿真软件的使用技巧，学会了利用 Analyze Circuit 功能来进行自动绘制模拟电路，初步了解了 Verilog HDL 语言的语法和使用，收获较大。