

【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】

PC 一台

macOS 操作系统

Java 运行环境(jre)

Logisim 仿真工具

vlab.ustc.edu.cn

【实验过程】

Step1:

用真值表生成电路:

例题题干中给出的真值表为：

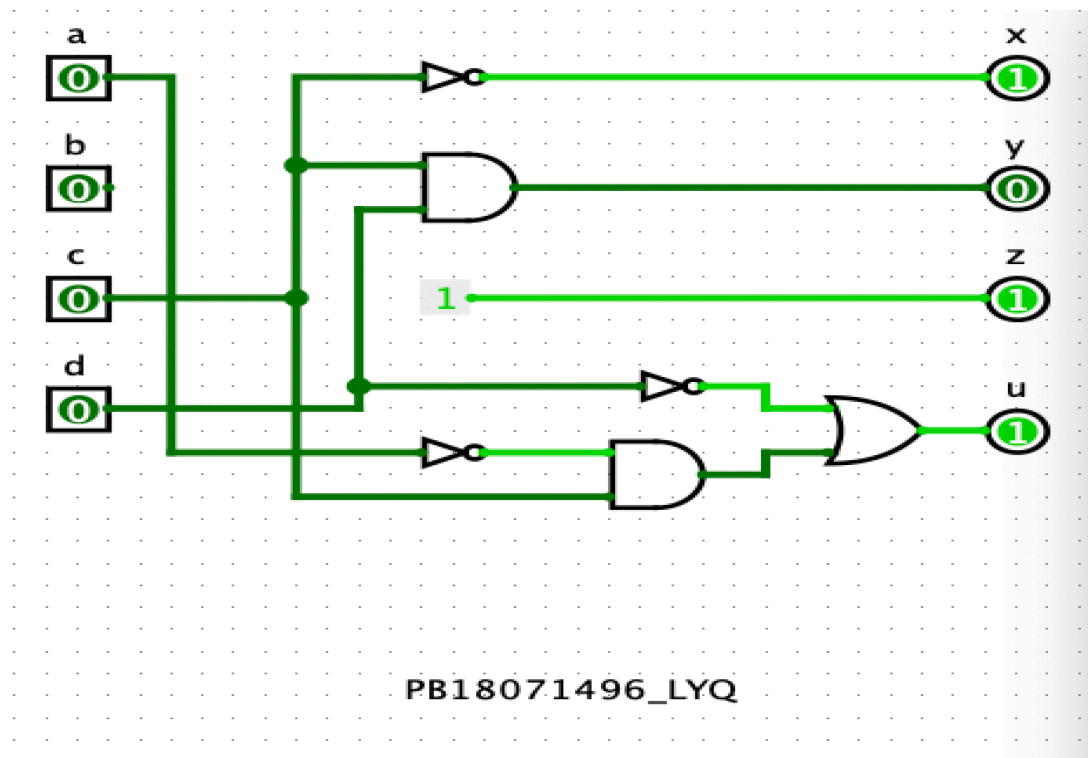
输入	输出
0001	1010
0011	0111
1010	0011
1011	0110
1111	0101

放置4个输入，4个输出：

Combinational Analysis							
Inputs		Outputs		Table	Expression	Minimized	
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	x	x	x	x

Build Circuit

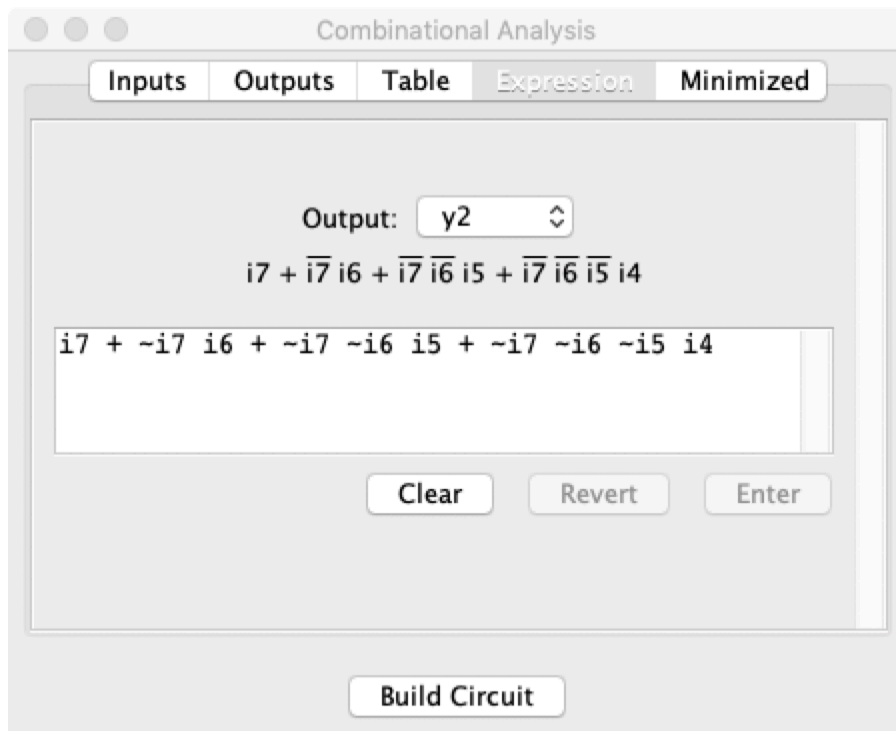
Build Circuit 后，得到下图中的电路：



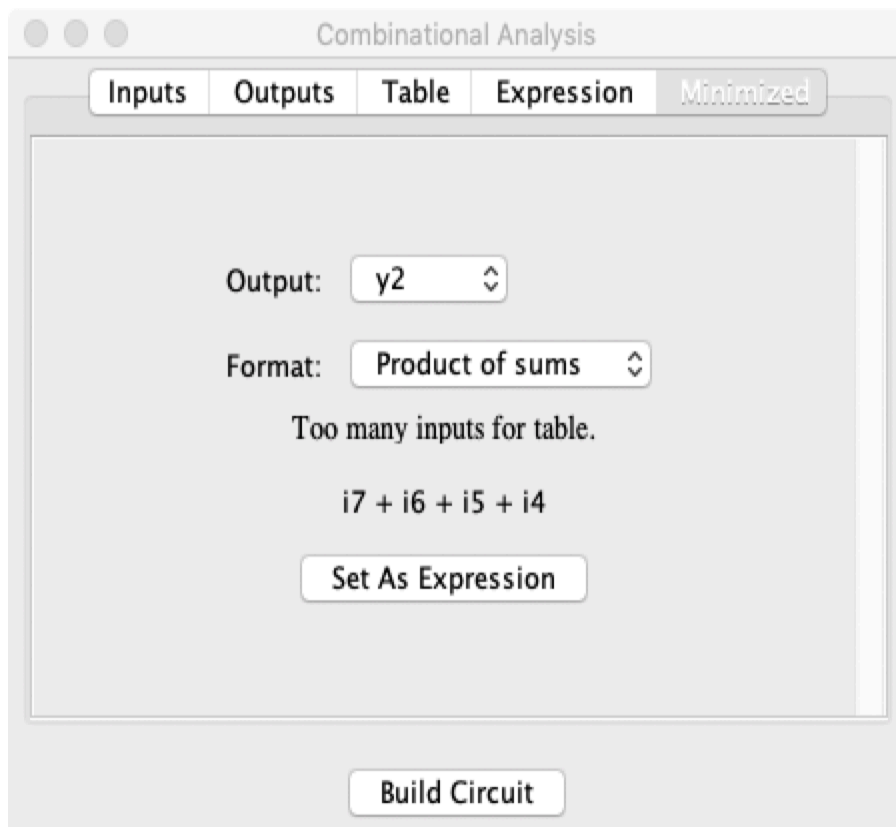
完毕。

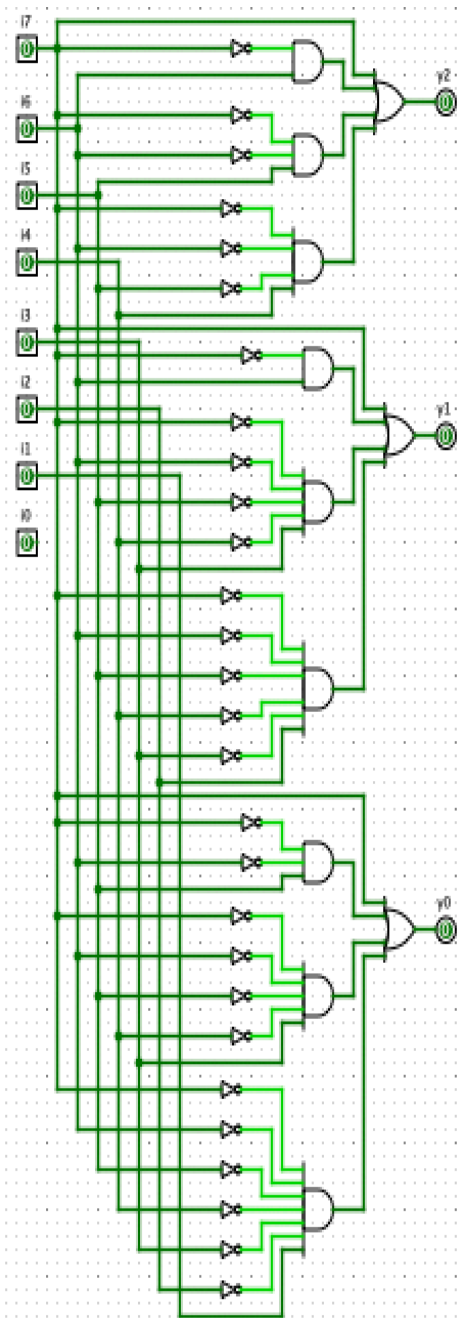
Step2:

用表达式生成电路：（以 y2 为例）

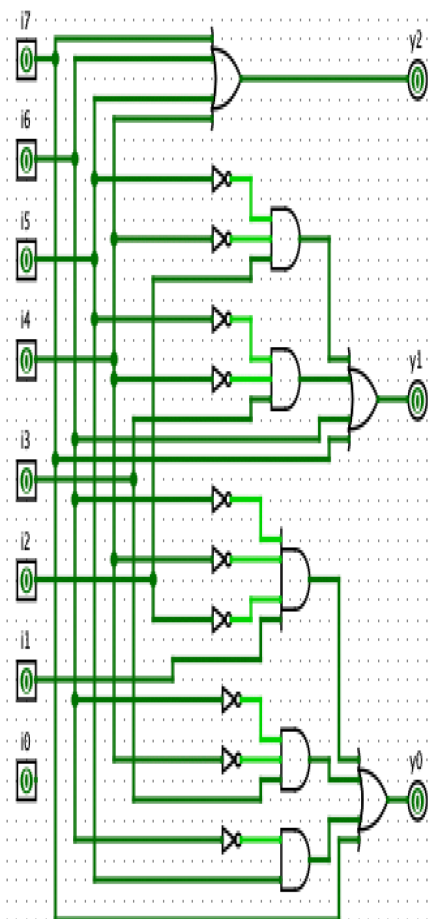


之后还可设置表达式的不同形式，得到的电路结构也不同





PB18071496_李昱祁



PB18071496

但它们的逻辑功能是一样的。

Logisim: step2 Statistics				
Component	Library	Simple	Unique	Recur...
Pin	Wiring	11	11	11
NOT Gate	Gates	10	10	10
AND Gate	Gates	5	5	5
OR Gate	Gates	3	3	3
Label	Base	1	1	1
TOTAL (without project's sub...		30	30	30
TOTAL (with subcircuits)		30	30	30

Close

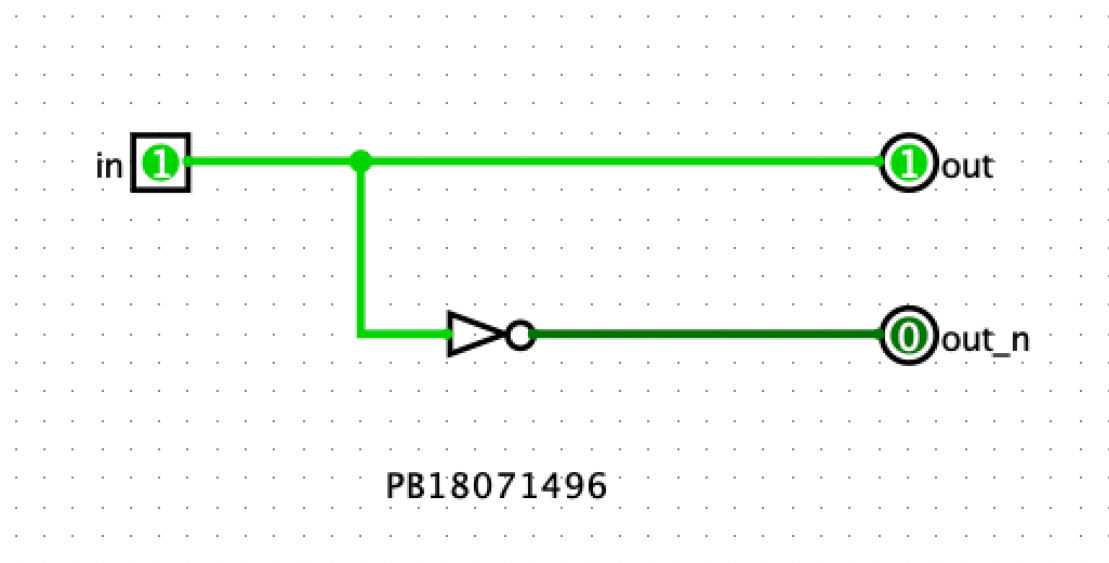
之后，根据讲义中的指导统计了电路中各种门的数量（上图）

Step3:

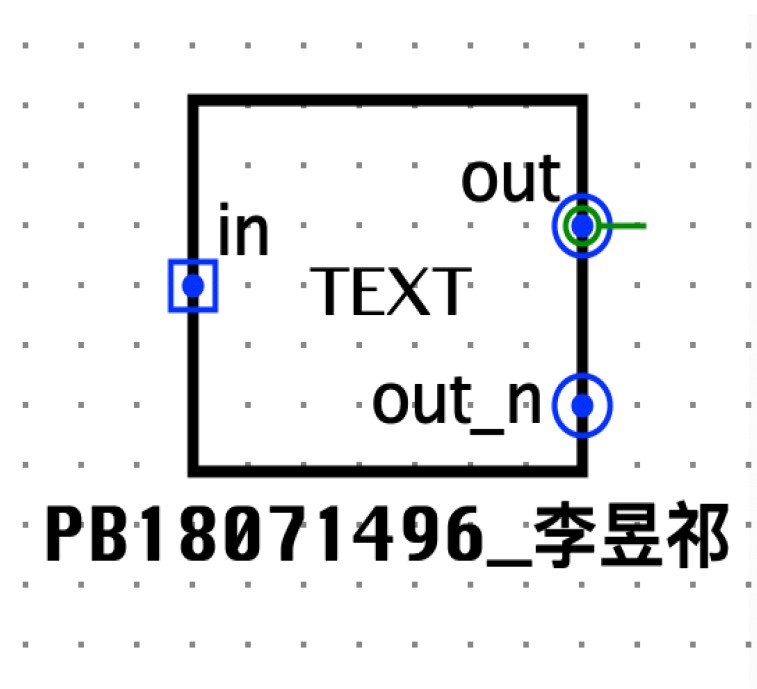
Verilog HDL 语法入门:

例 1:

按照讲义要求先画出电路:



之后进行封装：



在 Visual Studio code 编辑器中写出该电路的代码：

```

🔧 step3.v
1  module test( //模块名称
2  input in, //输入信号声明
3  output out, //输出信号声明
4  output out_n); //如需要, 可在此处声明内部变量
5  /*****以下为逻辑描述部分*****/
6      assign out = in;
7      assign out_n = ~in;
8  /*****逻辑描述部分结束*****/
9  endmodule //模块名结束关键词
10

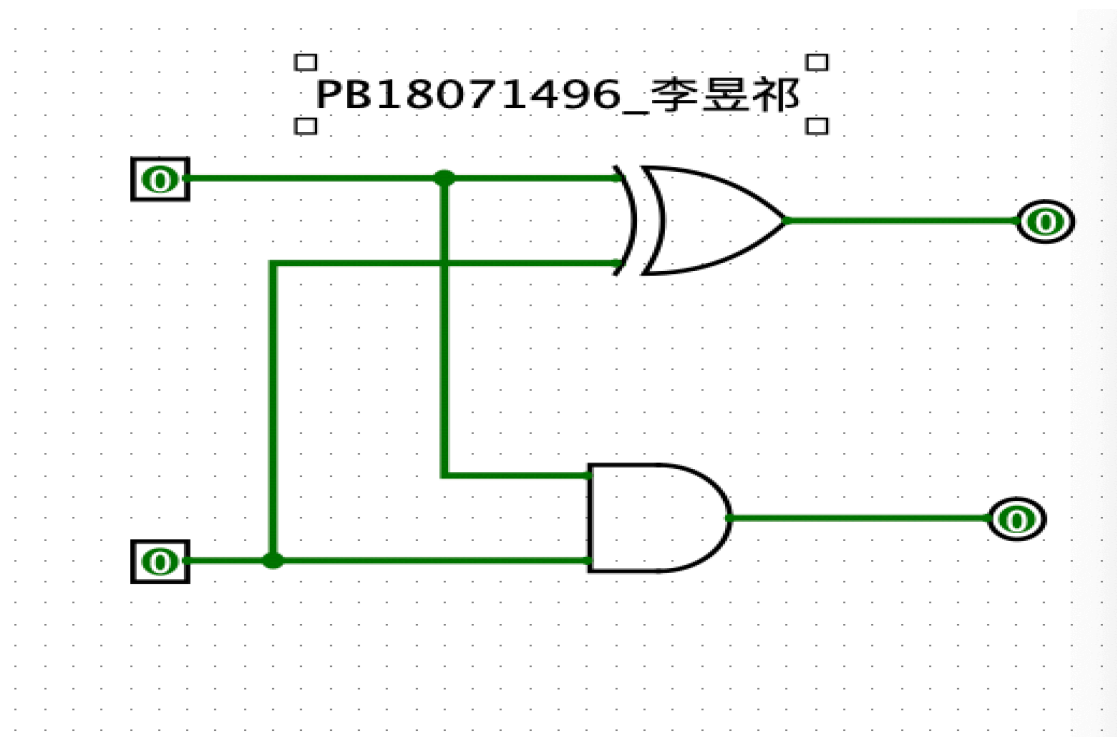
```

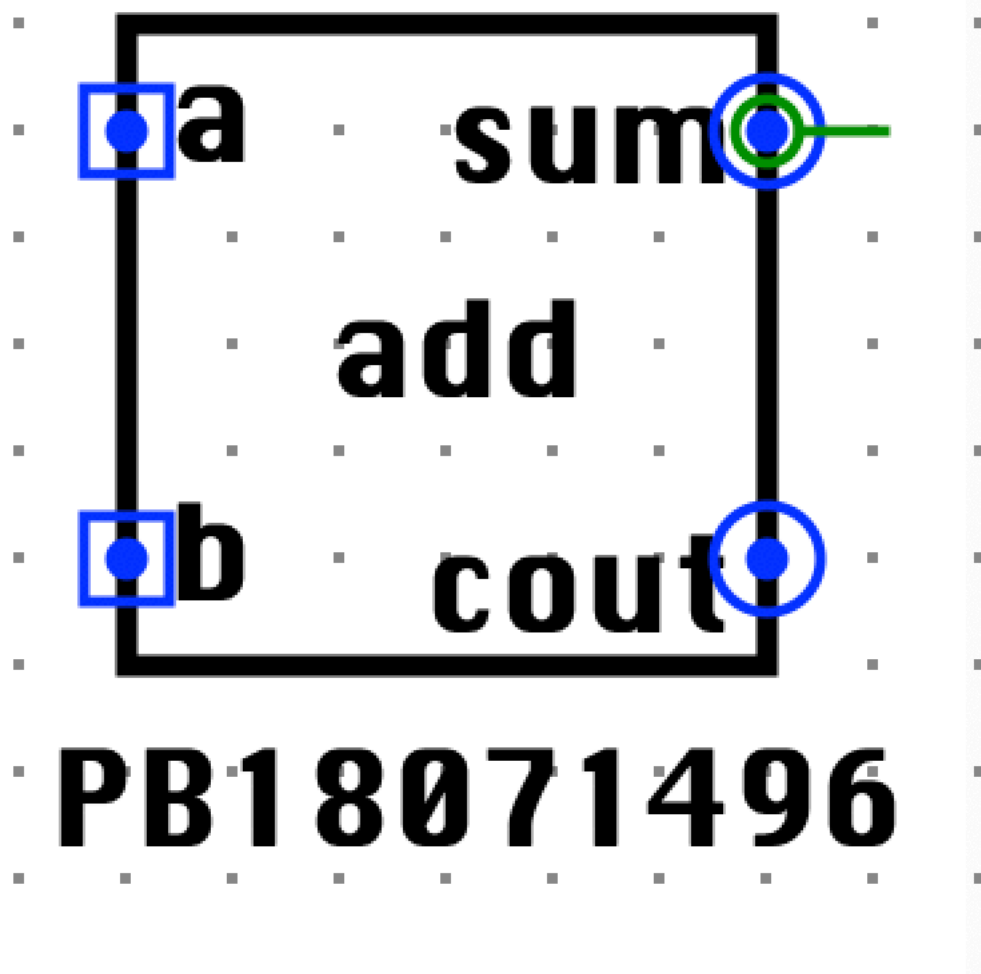
(此主题下 verilog 代码的关键字是紫色的)

verilog 支持 C 及 C++风格的注释

例 2:

同样的, 先画出电路并进行封装





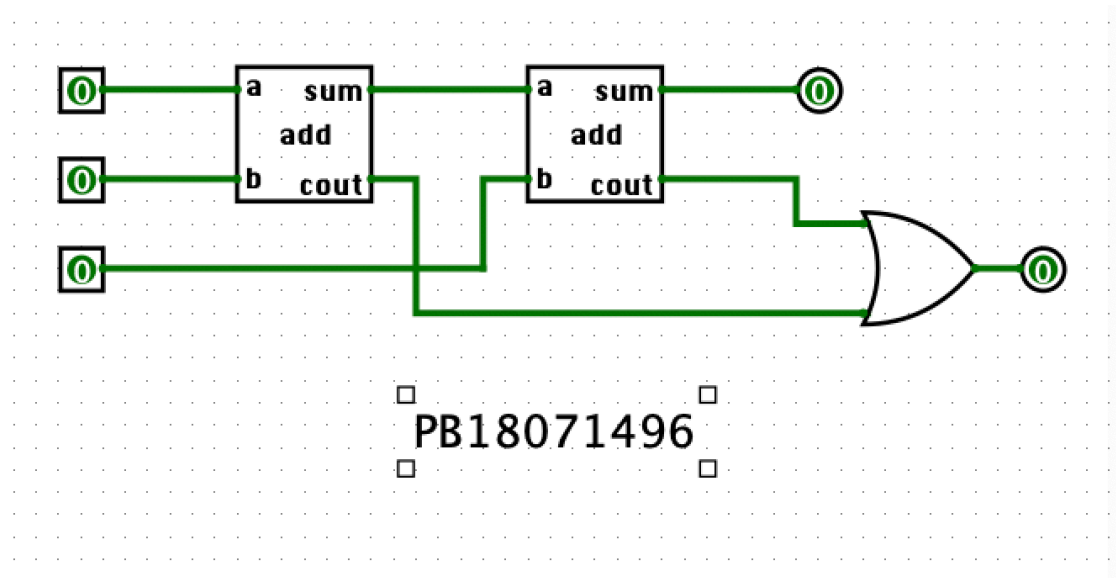
写出对应的 verilog 代码:

```
step3.v
1  module add( //半加器
2      input a,b,
3      output sum,cout);
4      assign {cout,sum} = a + b; //位拼接功能
5  endmodule
```

实现的功能为半加器

例 3:

使用上一例中构造完毕位的半加器进行新电路的设计:



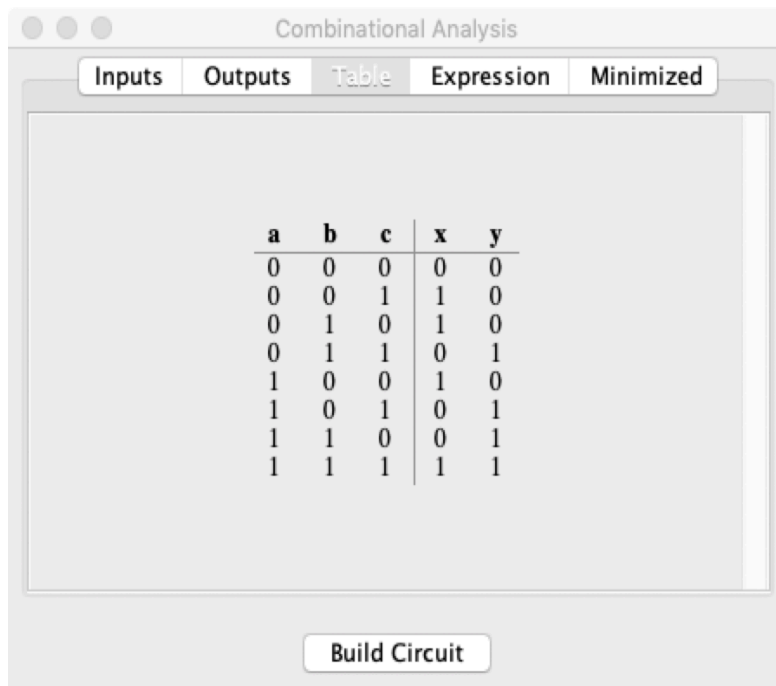
Verilog 代码为:

```
step3_v
1  module full_add(
2      input  a,b,cin,
3      output sum,cout);
4      wire s,carry1,carry2;
5      add  add_inst1(
6          .a    (a    ),
7          .b    (b    ),
8          .sum   (s    ),
9          .cout  (carry1));
10     add add_inst2(
11         .a    (s    ),
12         .b    (cin  ),
13         .sum   (sum  ),
14         .cout  (carry2));
15     assign cout = carry1 | carry2;
16     endmodule
```

【实验练习】

题目 1:

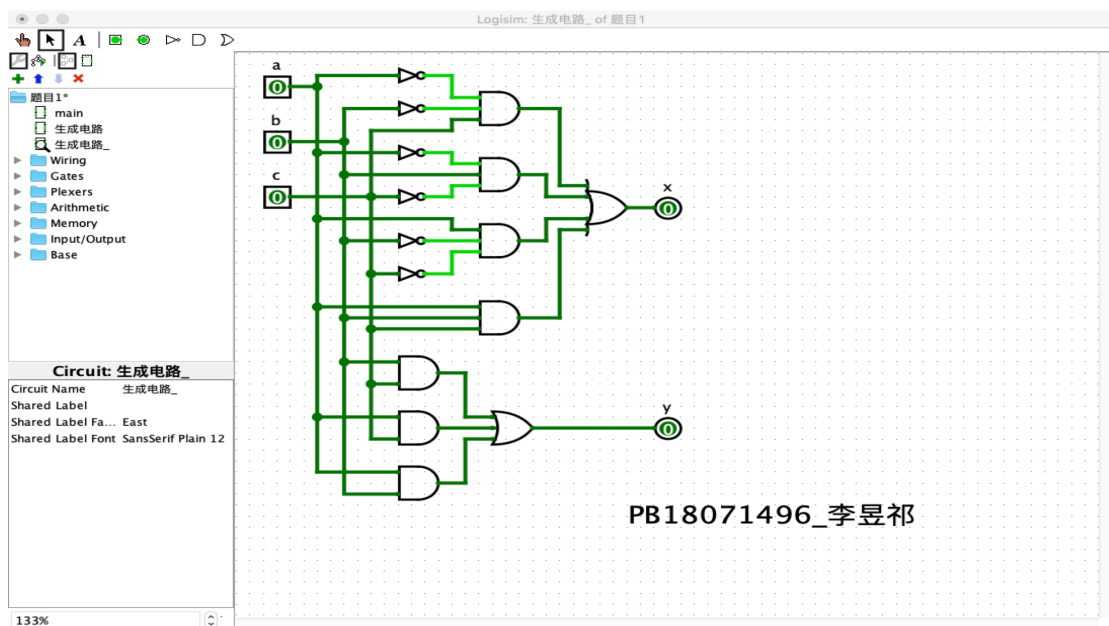
仿照讲义例题中的内容，直接在“Project”->“Analyze circuit”中输入真值表：



a	b	c	x	y
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

之后为了直观上简化电路，不对使用的门进行限制。

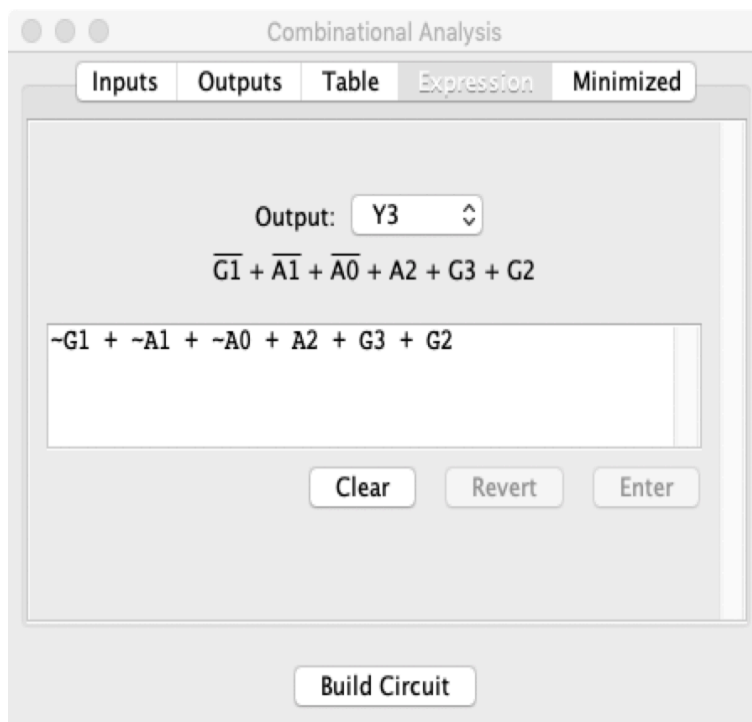
得到如下电路：



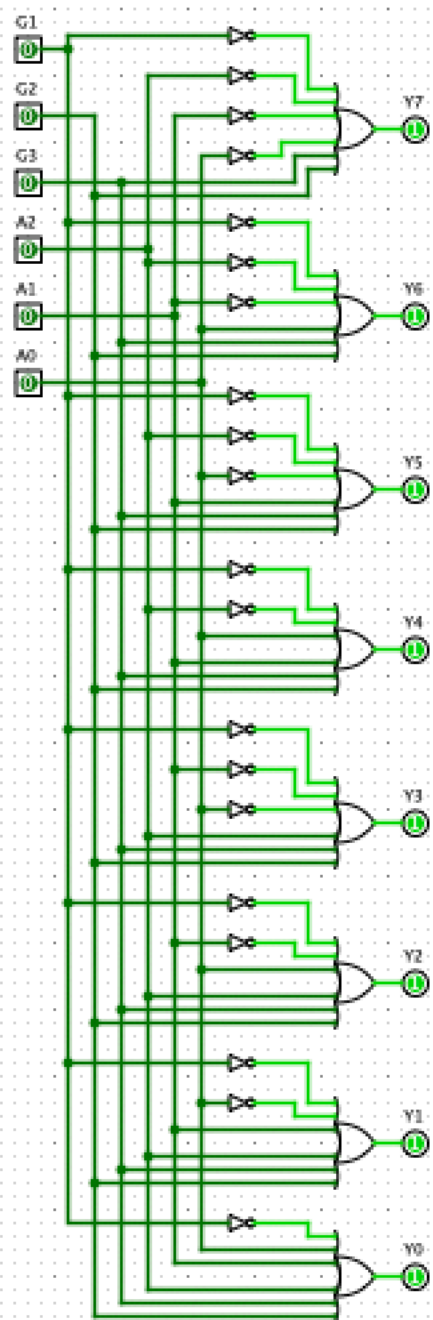
完毕。

题目 2:

根据题干中所给表达式可以写出 Y7~Y0 的逻辑表达式，并且输入至 Analyze circuit 面板的 Expression 选项，下面截取 Y3 为例：



之后即可进行 Build Circuit 操作，生成如下电路：

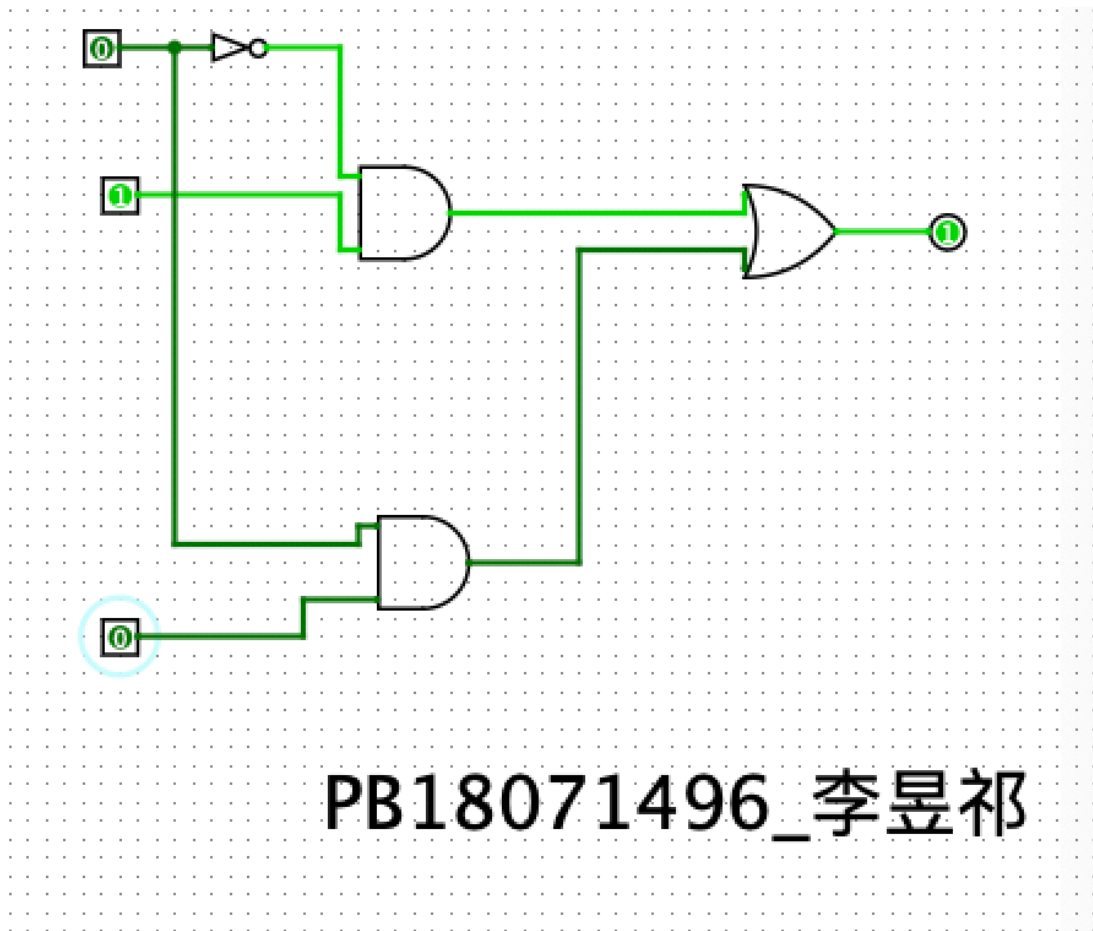


PB18071496_李昱祁

完毕

题目 3:

先搭建基本的 1-bit 二选一电路:

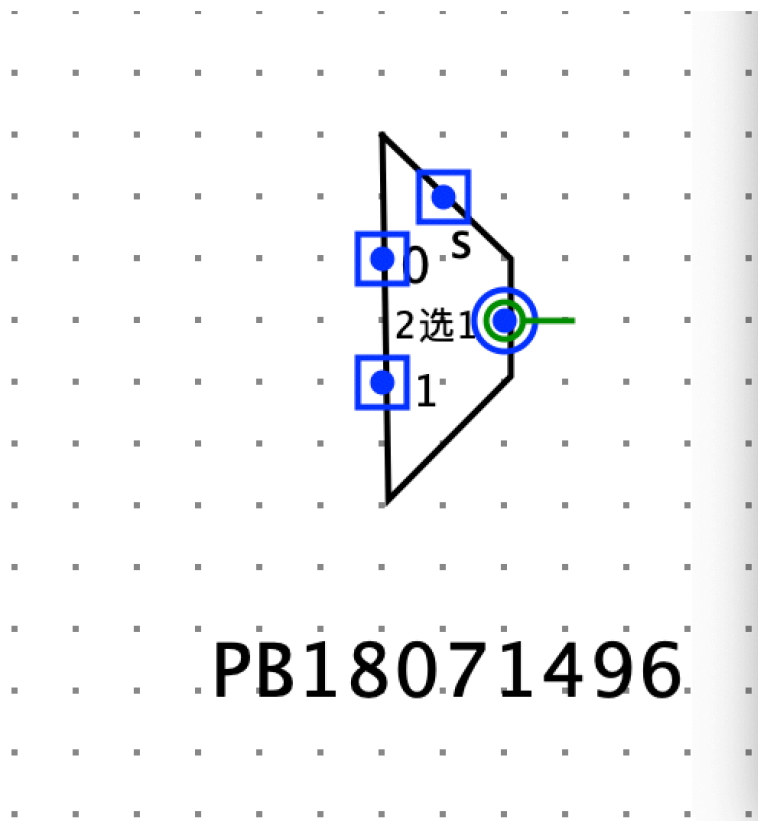


对应的 verilog 代码如下：

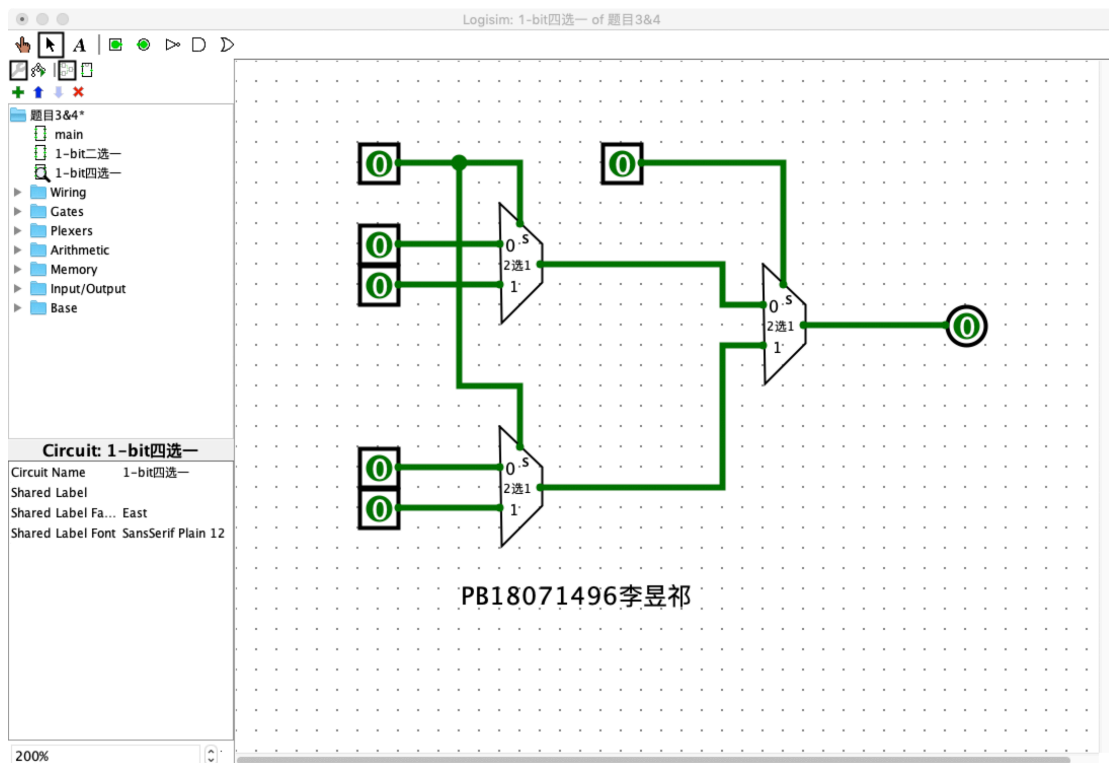
```
🔧 题目3_verilog代码.v
1  module exm3(input sel,
2      input a,b,
3      output out);
4      assign out = (~sel&a)|(sel&b);
5  endmodule
```

题目 4：

先将题目 3 中画出的 2 选 1 电路封装：



用 1-bit 来搭建 2-bit:



对应 verilog 代码:

```
题目4_verilog代码.v
1  module exm4(input a,b,c,d
2      input sel0,sel1
3      output out);
4      wire E1,E2;
5      exm3 _mux1(sel1,a,b,E1);//exm3是题目3中已封装好的电路
6      exm3 _mux2(sel1,c,d,E2);
7      exm3 _mux3(sel0,E1,E2,out);
8  endmodule
```

题目 5:

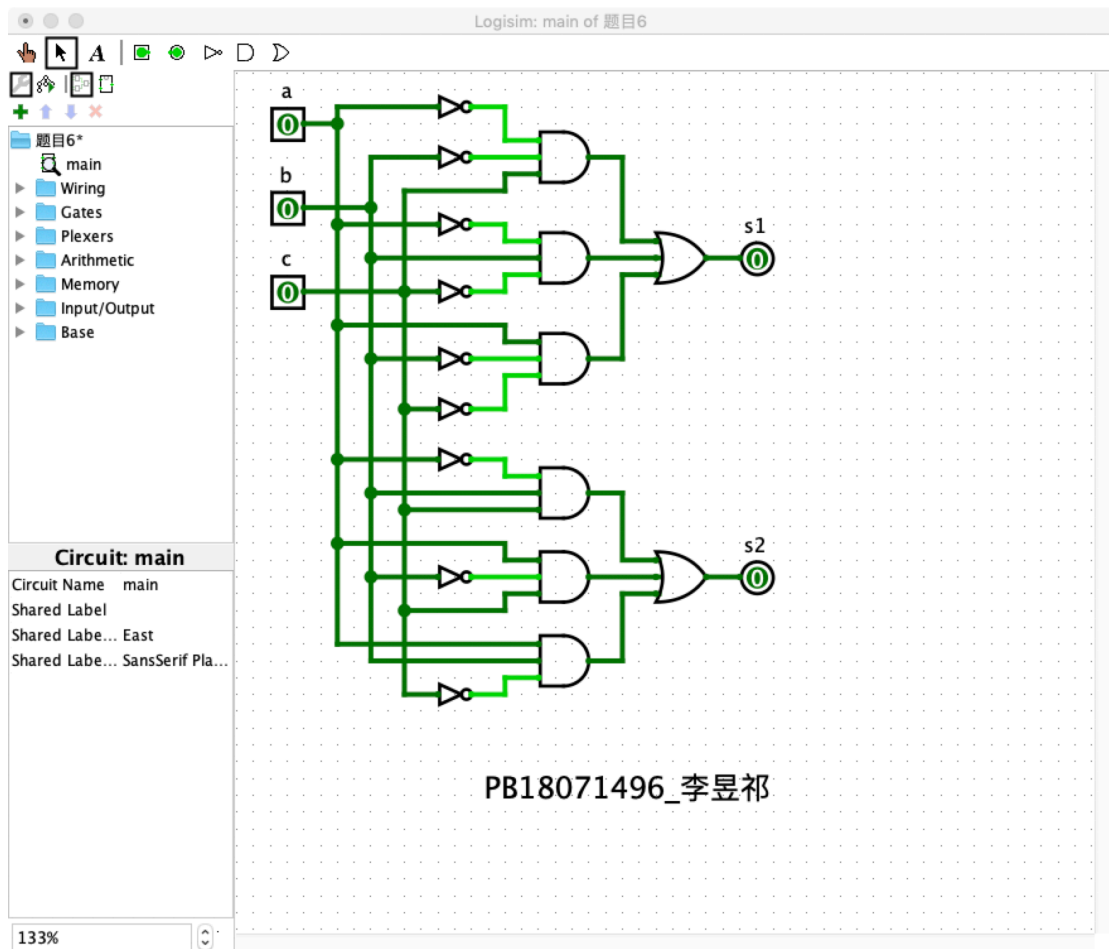
讲义例题中已给出 $Y_2 \sim Y_0$ 的逻辑表达式,直接使用 verilog 的赋值语句即可:

```
题目5_verilog代码.v
1  module (input I7,I6,I5,I4,I3,I2,I1,I0
2      output Y0,Y1,Y2);
3      assign Y2 = I7 + ~I7&I6 + ~I7&~I6&I5 + ~I7&~I6&~I5;
4      Y1 = I7 + ~I7&I6 + ~I7&~I6&~I5&~I4&I3 + ~I7&~I6&~I5&~I4&~I3&I2;
5      Y0 = I7 + ~I7&~I6&I5 + ~I7&~I6&~I5&~I4&I3 + ~I7&~I6&~I5&~I4&~I3&I2;
6
7  endmodule
```

题目 6:

不难读出该段代码蕴含的 input、output 之间的逻辑关系。通过之前的练习,可以通过输入 Expression 逻辑表达式的方式快速生成电路。

结果如下:



【总结与思考】

1. 掌握了 logisim 软件通过真值表、逻辑表达式生成电路的方法；

初步了解 verilog 语言编程, 并且自己在 vscode 编辑器上编写了一些简单的 verilog 程序。

2. 较低
3. 略少
4. 第一次实验也让同学们自行搭建了 2 选 1, 4 选 1 数据选择器, 这次的题目中又出现了, 有些重复。