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：Ouedraogo Ezekiel B.

学生学号：PL19215001

完成日期：2020/11/03

计算机实验教学中心制

2020 年 10 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】

PC 一台：Windows 操作系统

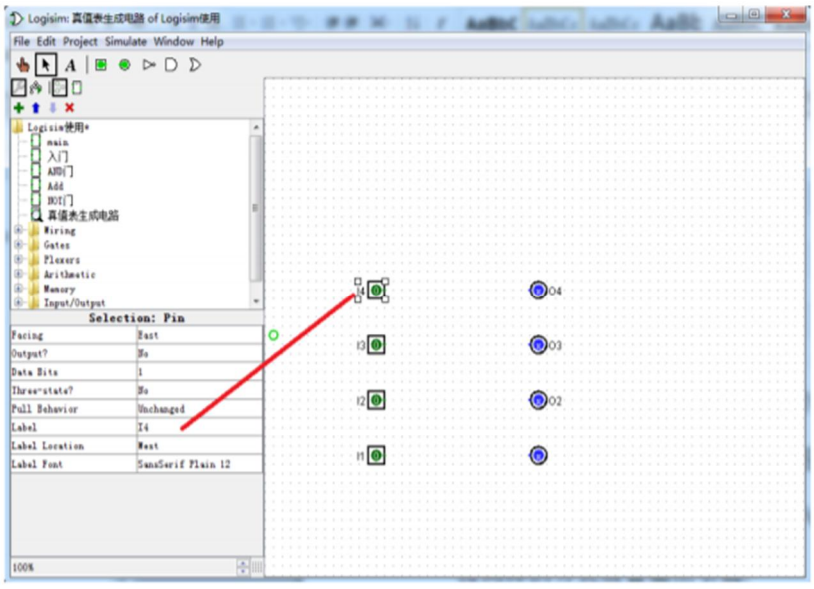
Logisim 仿真工具。

Notepad++ 文本编辑器

【实验过程】

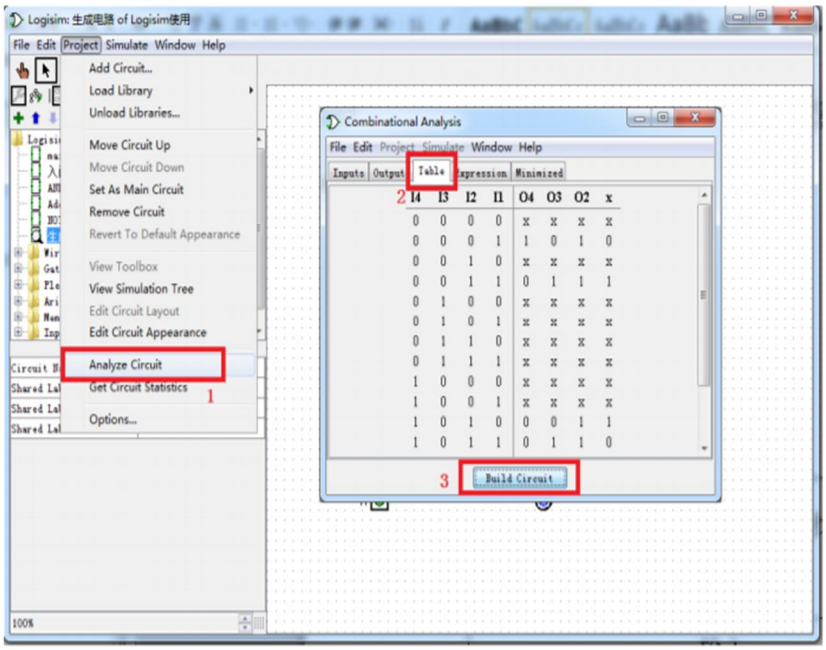
1. 用真值表自动生成电路

首先在电路图中放置输入引脚，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。



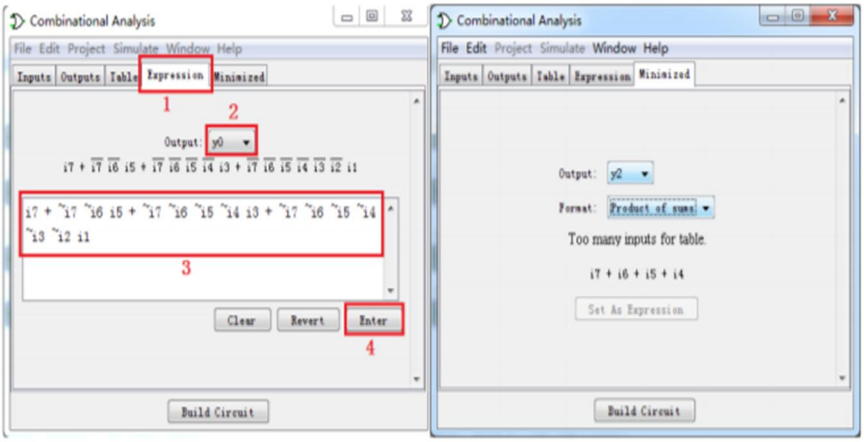
然后在菜单栏的“Project”选项卡中选“Analyze Circuit”。在弹出的窗口中选择“Table”选项，修改输出值，

最后点击“Build Circuit”便可生成电路。



2. 用表达式生成电路图

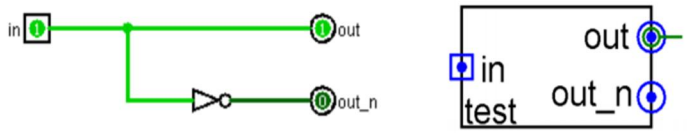
在 Logisim 中直接输入表达式生成电路，在“Project” --> “Analyze Circuit”的弹出窗口中选择“Expression”选项， 填入每个输出信号的表达式，再借助“Minimized”选项卡对表达式进行简 化，进而减少电路使用的逻辑门数量。最后点击“Build Circuit”生成电路。



3. Verilog HDL 语法入门

通过对照 Logisim 中设计的简单电路学习 Verilog 语 法。

1） test 模块



下面以此为例介绍 Verilog 的写法。编写 Verilog 代码前首先新建一个文本文件，修改后缀并重命名为 test.v，使用 Notepad++文本编辑器进行编辑。代码内容为:

```
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 out_n = ~in;
/*******逻辑描述部分结束******/
endmodule //模块名结束关键词
```

Verilog 模块的最基本结构。

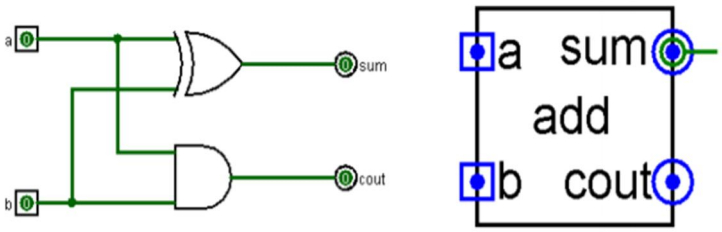
```
module  模块名(
    输入端口声明,
    输出端口声明);
    内部信号声明<可选>;

    逻辑描述（模块主体）
endmodule
```

上述代码包含了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。

代码中也用到了一个非 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。

单行注 释以“//”开始，多行注释则使用“/* 注释内容 */”。

2） 半加器



其 Verilog 代码为：

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

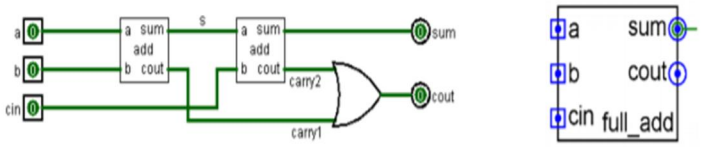
代码中因为输入 a, b 都是单 bit 信号， 但他们相加的结果需要一个两 bit 位宽的信号才能保存，而 cout 和 sum 都是单 bit 的信号，通过使用{}（位拼接符号）将两个单 bit 信号拼接成了一个 2bit 信号，用于接收相加的结果。

此外此电路从行为级上进行了描述，直接描述了两个输入信号相 加这一行为，实际上下面这段代码与前面所描述的电路是完全一样的， 因为它们的行为特性一样。

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

在上面这段代码中有两条连续赋值语句，它们的顺序交换并不会 对电路产生影响。

3） 全加器



其 Verilog 代码如下：

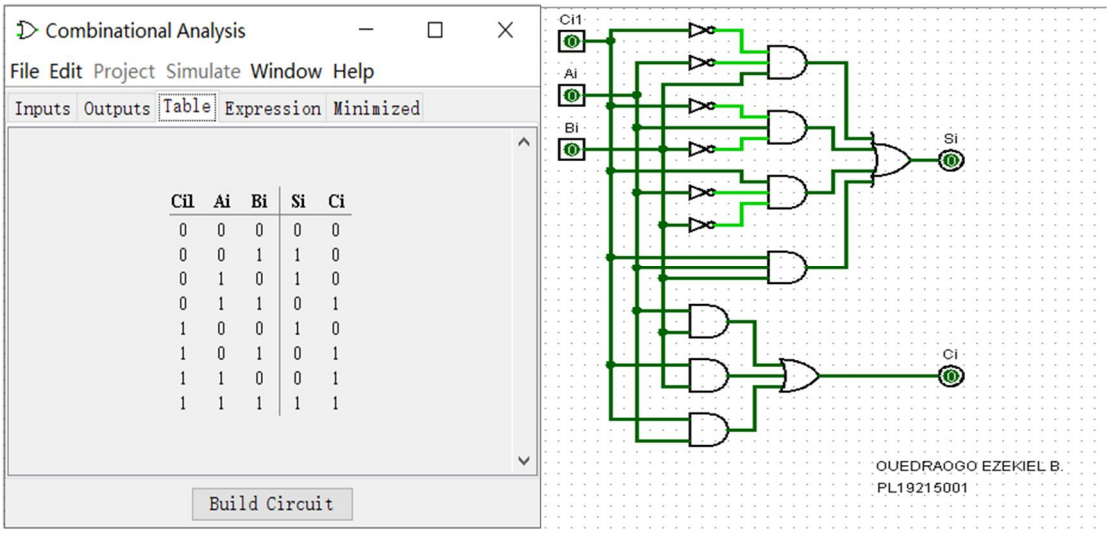
```
module full_add(
input  a,b,cin,
output sum,cout);
wire s,carry1,carry2;
add   add_inst1(
.a     (a    ),
.b     (b    ),
.sum   (s    ),
.cout  (carry1));
add add_inst2(
.a     (s    ),
.b     (cin  ),
.sum   (sum  ),
.cout  (carry2));
assign cout = carry1 | carry2;
endmodule
```

该代码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。wire 类型是 verilog 中的默认类型。

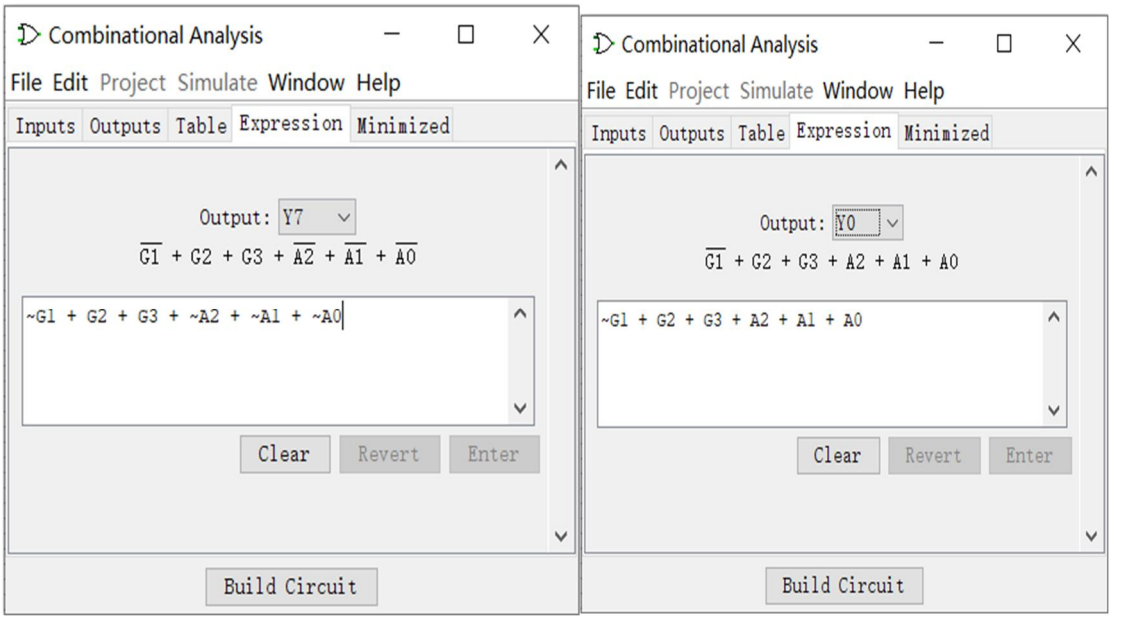
模块调用：在本例中，我 们调用了两个半加器，以实现全加器的功能，其中 add 为被调用模块 的模块名，add_inst1、add_inst2 为实例化名称。

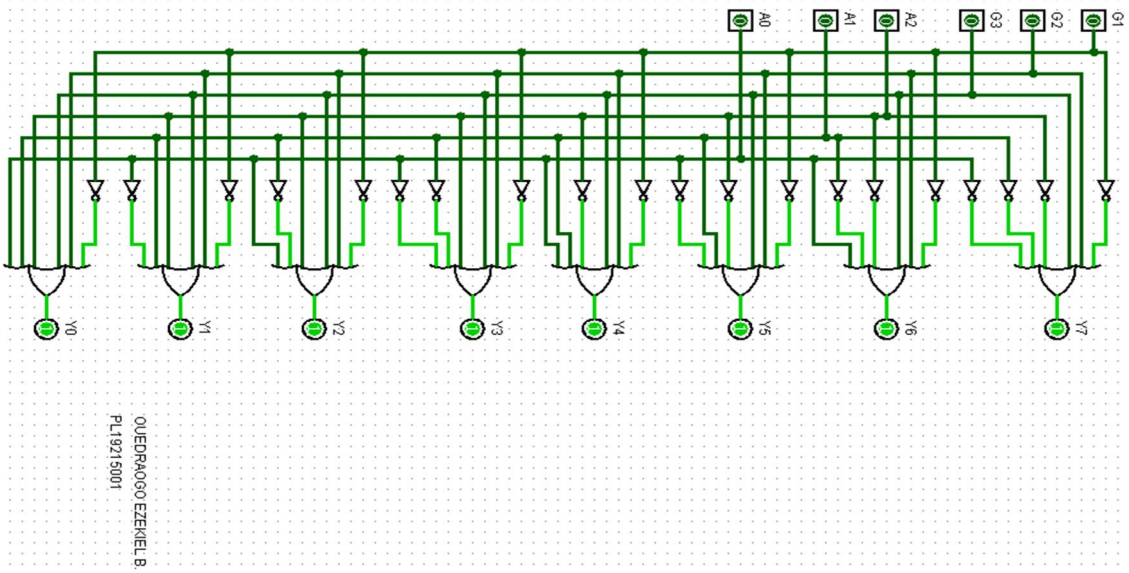
【实验练习】

1. 通过 Logisim 编辑真值表功能，完成电路 设计。



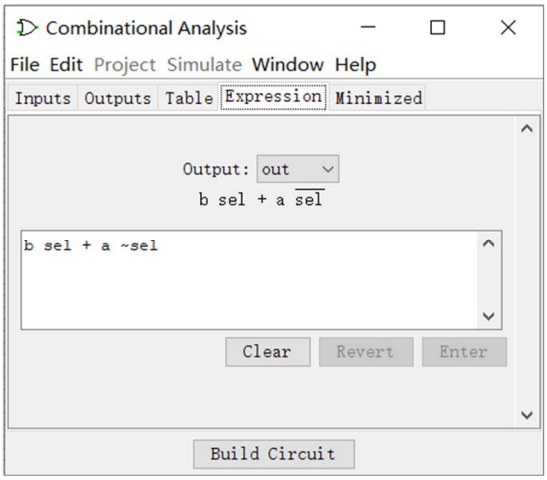
2. 通过 Logisim 的编辑表达式功能完成电路 设计

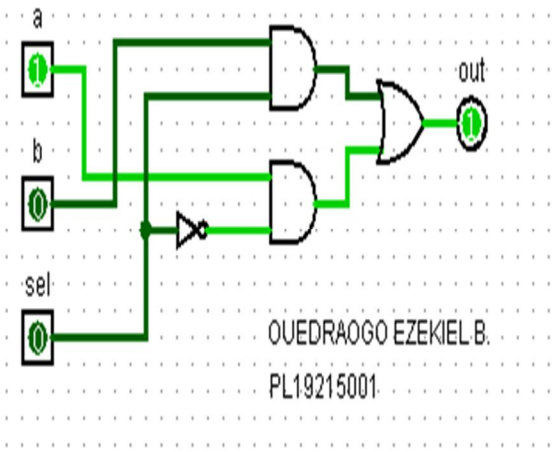




3. 二选一选择器

使用 Logisim 绘制 1bit 位宽的二选一选择器电路图。输入为a, b, sel，输出为out，sel 为 0 时选通 a 信号。通过 Logisim 编辑表达式功能设计电路 $out = a \cdot \overline{sel} + b \cdot sel$





据生成的电路图编写 Verilog 代码

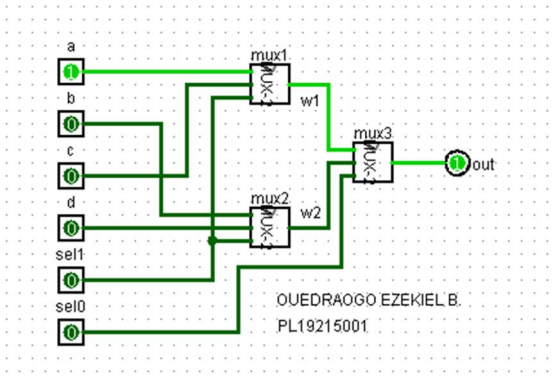
```
module Mux_2 (
    input a,
    input b,
    input sel,
    output out
);

    assign out = (sel & b) | (~sel & a);

endmodule
```

4. 四 选一选择器

通过上面的二选一选择器，用 Verilog 实现一个四 选一选择器。输入信号为 a,b,c,d,sel1,sel0，sel1 和 sel0 都为 0 时选中 a 信号。

```
12  module Mux_4(
13      input a, b, c, d,
14      input sel0, sel1,
15      output out );
16
17      wire w1,w2;
18      Mux_2 mux1(.a(a),.b(c),.sel(sel1),.out(w1));
19      Mux_2 mux2(.a(b),.b(d),.sel(sel1),.out(w2));
20      Mux_2 mux3(.a(w1),.b(w2),.sel(sel0),.out(out));
21
22  endmodule
```

对应的电路图



5. 优先编码器

根据前八位优先编码器真值表，编写 verilog 代码。

真值表

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

代码

```
module top_module(
    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
```

6．描述代码功能

代码

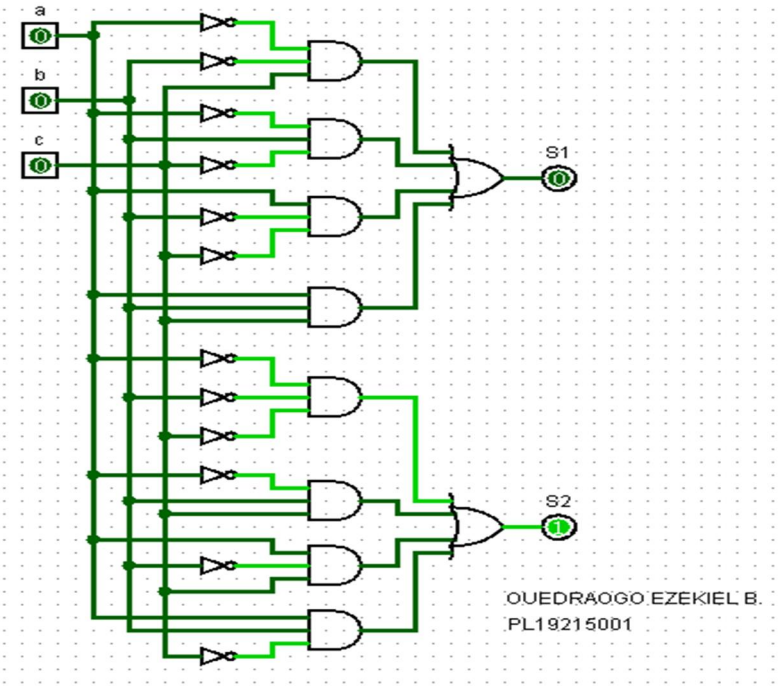
```
module test(
input a,b,c,
output s1,s2);
assign s1= ~a &~b & c | ~a & b &~c | a &~b &~c | a & b & c;
assign s2= ~a & b & c | a &~b & c | a & b &~c | ~a &~b &~c;
endmodule
```

为了研究代码的功能，首先建其真值表

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

已看出a,b,c中有奇数个 1 时$S1$为 1,S2 为 0，有偶数个 1 时 S1 为 0，S2 为 1。因此该代码判断输入中 1 个数的奇偶性。

电路图



【总结与思考】

通过本次实验咱们进一步了解了 Logism 的更多的功能如自动生成电路功能，学到了 Verilog 的基本语法。