

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：傅申_____

学生学号：PB20000051_____

完成日期：2021 年 10 月 31 日

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

【实验环境】

- Windows PC 一台: CPU 为 Intel i5-1035G1
- Logisim 仿真工具
- Xilinx Vitis and Vivado ML Edition 2021.1 与 Micorsoft Visual Studio Code

【实验过程】

Step 1: 用真值表自动生成电路

在 Logisim 中新建电路 Step.1. , 然后在画布上放置 4 个输入引脚和 4 个输出引脚, 分别标上标号 I1 - I4, O1 - O4, 并按高低位顺序排列. 点击菜单栏的 “Project” → “Analyze Circuit” 选项, 在弹出的窗口中选择 “Table” 选项, 点击相应的位置修改输出值 (如图 1(a)), 点击 “Build Circuit” 即可生成电路, 如图 1(b).

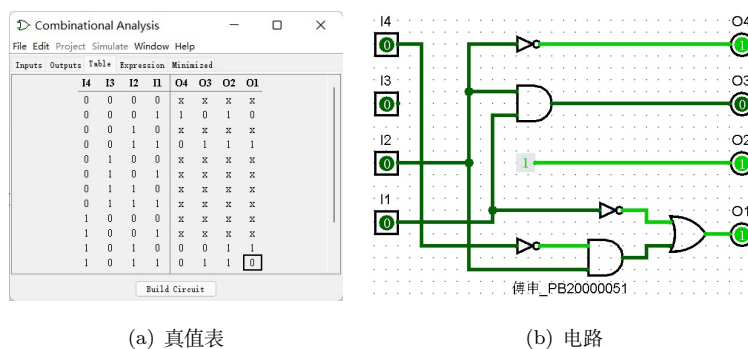


图 1: 用真值表自动生成电路

Step 2: 用表达式生成电路图

当输入信号比较多时, 编辑真值表也是比较繁重, 这时我们可以通过表达式生成电路. 比如要生成一个 8 位的优先编码选择器, 其真值表有 256 项之多, 但我们可以很快的

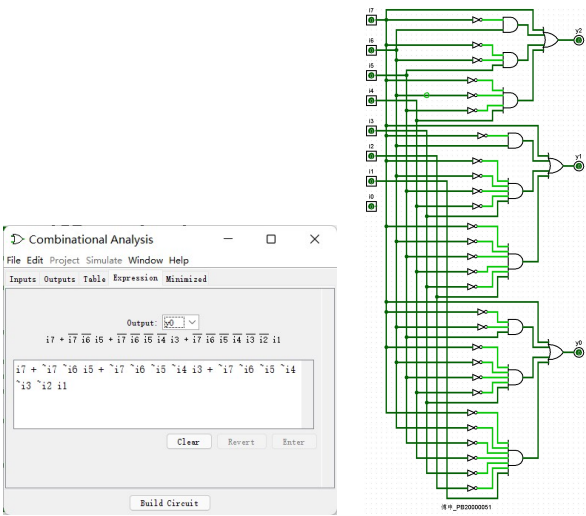
写出其表达式:

$$\begin{aligned}y_2 &= i_7 + \overline{i_7} \cdot i_6 + \overline{i_7} \cdot \overline{i_6} i_5 + \overline{i_7} \cdot \overline{i_6} \cdot \overline{i_5} i_4 \\y_1 &= i_7 + \overline{i_7} \cdot i_6 + \overline{i_7} \cdot \overline{i_6} \cdot \overline{i_5} \cdot \overline{i_4} \cdot i_3 + \overline{i_7} \cdot \overline{i_6} \cdot \overline{i_5} \cdot \overline{i_4} \cdot \overline{i_3} \cdot i_2 \\y_0 &= i_7 + \overline{i_7} \cdot i_6 + \overline{i_7} \cdot \overline{i_6} \cdot \overline{i_5} \cdot \overline{i_4} \cdot i_3 + \overline{i_7} \cdot \overline{i_6} \cdot \overline{i_5} \cdot \overline{i_4} \cdot \overline{i_3} \cdot \overline{i_2} \cdot i_1\end{aligned}$$

在 Logisim 的画布上放置相应的输入和输出引脚, 然后点击菜单栏的 “Project” → “Analyze Circuit” 选项, 在弹出的窗口中选择 “Expression” 选项, 填入每个输出信号的表达式如下:

$$\begin{aligned}y_2 &= i_7 + \sim i_7 i_6 + \sim i_7 \sim i_6 i_5 + \sim i_7 \sim i_6 \sim i_5 i_4 \\y_1 &= i_7 + \sim i_7 i_6 + \sim i_7 \sim i_6 \sim i_5 \sim i_4 i_3 + \sim i_7 \sim i_6 \sim i_5 \sim i_4 \sim i_3 i_2 \\y_0 &= i_7 + \sim i_7 \sim i_6 i_5 + \sim i_7 \sim i_6 \sim i_5 \sim i_4 i_3 + \sim i_7 \sim i_6 \sim i_5 \sim i_4 \sim i_3 \sim i_2 i_1\end{aligned}$$

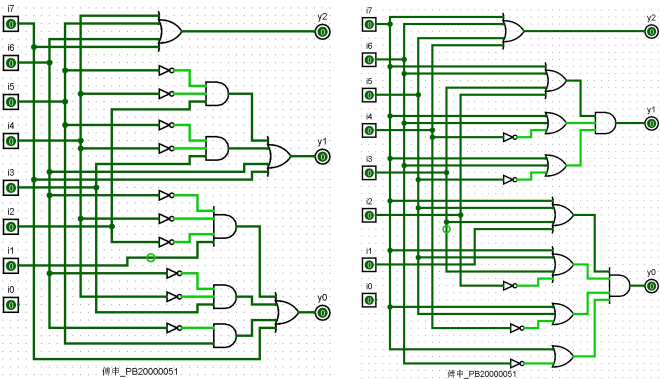
最后点击 “Build Circuit” 生成电路, “Expression” 选项卡和生成的电路如下图 2.



(a) “Expression” 选项卡 (b) 生成的电路

图 2: 用表达式生成电路图

可以看出该电路占用的逻辑门较多, 这时可以借助窗口中的 “Minimized” 的选项卡来对表达式和电路进行简化, 如图 3 中的化简后的与或式和或与式结构.



(a) 与或式 (b) 或与式

图 3: 简化后的电路

Step 3: Verilog HDL 语法入门

Verilog 模块的最基本结构如下代码 1.

Verilog 代码 1: Verilog 模块基本结构

```
1  module 模块名(  
2      输入端口声明,  
3      输出端口声明  
4  );  
5      内部信号声明 <可选>;  
6  
7      逻辑描述 (模块主体)  
8  endmodule  
9  
10 // 单行注释  
11 /*  
12     多行注释  
13 */
```

比如我们要实现一个半加器, 其 Verilog 代码就如下代码 2, 其中 add 模块从行为级上进行描述, 而 add1 模块则从门电路层次进行描述.

Verilog 代码 2: 半加器模块

```
1      // 从行为级上描述  
2  module add(  
3      input a, b,  
4      output sum, cout  
5  );  
6      assign {cout,sum} = a + b; // 位拼接  
7  endmodule  
8  
9  // 从电路级上描述  
10 module add1(  
11     input a,b,  
12     output sum,cout  
13 );  
14     // 两个 assign 是位置无关的  
15     assign cout = a & b;  
16     assign sum  = a ^ b;  
17 endmodule
```

而要使用已经设计了的模块, 我们可以在代码中调用该模块. 比如我们要使用半加器来构造一个全加器, 其 Verilog 代码就如代码 3.

Verilog 代码 3: 全加器

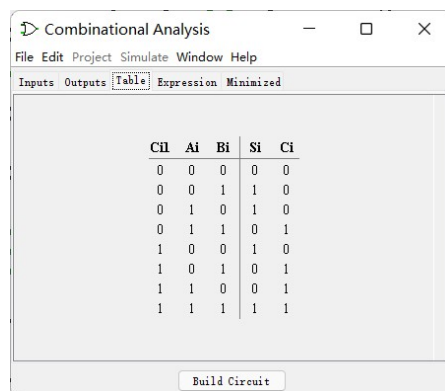
```

1 module full_add(
2     input a,b,cin,
3     output sum,cout
4 );
5 wire s,carry1,carry2; // 内部信号声明
6 // 调用半加器
7 add add_inst1(
8     .a (a ),
9     .b (b ),
10    .sum (s ),
11    .cout (carry1)
12 );
13 add add_inst2(
14     .a (s ),
15     .b (cin ),
16     .sum (sum ),
17     .cout (carry2)
18 );
19 assign cout = carry1 | carry2;
20 endmodule

```

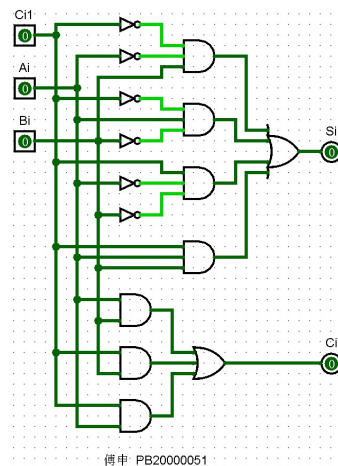
【实验练习】

题目 1: 在 Logisim 中新建电路, 在画布上放置 3 个输入引脚和 2 个输出引脚, 并标上标号, 打开 “table” 选项卡, 编辑真值表如下图 4(a), 点击 “Build Circuit”, 生成的电路如下图 4(b).



| C1i | A1 | B1 | S1 | C1 |
|-----|----|----|----|----|
| 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 |

(a) 真值表



(b) 电路

图 4: 题目 1 中的真值表和电路

题目 2: 分析下面的真值表, 列出下面的逻辑表达式:

$$Y_7 = \overline{G_1} + G_2 + G_3 + \overline{A_2} + \overline{A_1} + \overline{A_0}$$

$$Y_6 = \overline{G_1} + G_2 + G_3 + \overline{A_2} + \overline{A_1} + A_0$$

$$Y_5 = \overline{G_1} + G_2 + G_3 + \overline{A_2} + A_1 + \overline{A_0}$$

$$Y_4 = \overline{G_1} + G_2 + G_3 + \overline{A_2} + A_1 + A_0$$

$$Y_3 = \overline{G_1} + G_2 + G_3 + A_2 + \overline{A_1} + \overline{A_0}$$

$$Y_2 = \overline{G_1} + G_2 + G_3 + A_2 + \overline{A_1} + A_0$$

$$Y_1 = \overline{G_1} + G_2 + G_3 + A_2 + A_1 + \overline{A_0}$$

$$Y_0 = \overline{G_1} + G_2 + G_3 + A_2 + A_1 + A_0$$

在 Logisim 中新建电路, 在画布上放置 6 个输入引脚和 8 个输出引脚, 并标上标号, 打开 “Expression” 标签卡, 填入每个输出信号的表达式如下:

$$Y_7 = \sim G_1 + G_2 + G_3 + \sim A_2 + \sim A_1 + \sim A_0$$

$$Y_6 = \sim G_1 + G_2 + G_3 + \sim A_2 + \sim A_1 + A_0$$

$$Y_5 = \sim G_1 + G_2 + G_3 + \sim A_2 + A_1 + \sim A_0$$

$$Y_4 = \sim G_1 + G_2 + G_3 + \sim A_2 + A_1 + A_0$$

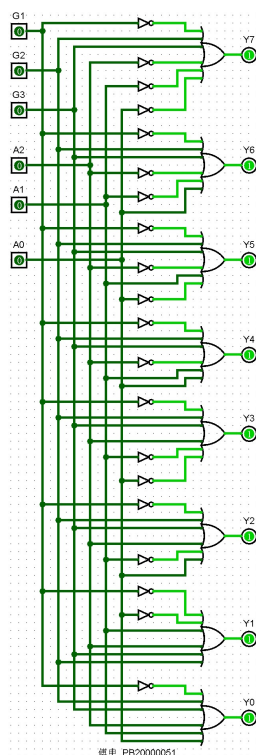
$$Y_3 = \sim G_1 + G_2 + G_3 + A_2 + \sim A_1 + \sim A_0$$

$$Y_2 = \sim G_1 + G_2 + G_3 + A_2 + \sim A_1 + A_0$$

$$Y_1 = \sim G_1 + G_2 + G_3 + A_2 + A_1 + \sim A_0$$

$$Y_0 = \sim G_1 + G_2 + G_3 + A_2 + A_1 + A_0$$

点击 “Build Circuit” 按钮, 生成的电路如图 5(a), 转到 “Table” 选项卡, Logisim 分析出的部分电路真值表如图 5(b).



(a) 生成的电路

| Combinational Analysis | | | | | | | | | | | | | |
|--|---------|-------|---|------------|---|-----------|---|---|---|---|---|---|---|
| File Edit Project Simulate Window Help | | | | | | | | | | | | | |
| Inputs | Outputs | Table | | Expression | | Minimized | | | | | | | |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 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 |

(b) 部分真值表

图 5: 题目 2 中的电路

题目 3: 使用 Logisim 绘制 1 bit 位宽的二选一选择器电路图如图，根据该电路图，编写的 Verilog 代码如下代码 4.

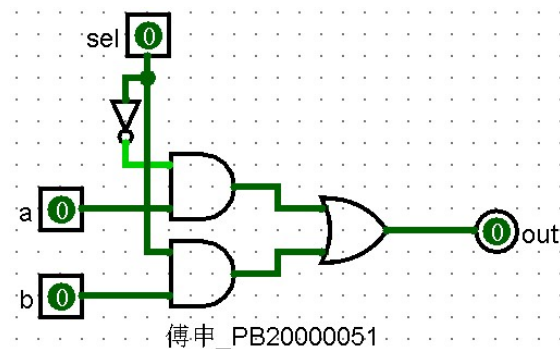


图 6: 1 bit 位宽的二选一选择器电路

Verilog 代码 4: 1 bit 位宽的二选一选择器

```
1 module MUX_1bit_2to1 (  
2     input i0, i1, sel,  
3     output out  
4 );  
5     assign out = (~sel & i0) | (sel & i1);  
6 endmodule
```

题目 4: 通过例化题目 3 中的二选一选择器，在四选一选择器中调用该模块，先对高位进行选择，在对低位进行选择，如代码 5.

Verilog 代码 5: 1 bit 位宽的四选一选择器

```
1 module MUX_1bit_4to1 (  
2     input a, b, c, d, sel1, sel0,  
3     output out  
4 );  
5     wire low_bit_0, low_bit_1;  
6  
7     MUX_1bit_2to1 Mux0(  
8         .i0 (a),  
9         .i1 (c),  
10        .sel(sel1),  
11        .out(low_bit_0)  
12    );  
13    MUX_1bit_2to1 Mux1(  
14        .i0 (b),  
15        .i1 (d),  
16        .sel(sel1),  
17        .out(low_bit_1)
```

```

18     );
19
20     MUX_1bit_2to1 Mux2(
21         .i0 (low_bit_0),
22         .i1 (low_bit_1),
23         .sel(sel0),
24         .out(out)
25     );
26 endmodule

```

对应的电路图如图.

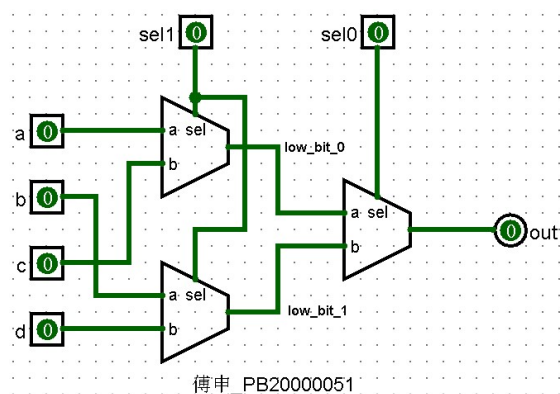


图 7: 1 bit 位宽的四选一选择器电路

题目 5: 使用化简后的逻辑表达式:

$$y_2 = i_7 + i_6 + i_5 + i_4$$

$$y_1 = i_7 + i_6 + \overline{i_5} \cdot \overline{i_4} \cdot i_3 + \overline{i_5} \cdot \overline{i_4} \cdot i_2$$

$$y_0 = i_7 + \overline{i_6} \cdot i_5 + \overline{i_6} \cdot \overline{i_4} \cdot i_3 + \overline{i_6} \cdot \overline{i_4} \cdot \overline{i_2} \cdot i_1$$

可以编写出八位优先编码器的 Verilog 代码如下代码 6.

Verilog 代码 6: 八位优先编码器

```

1 module priority_encoder_8bit (
2     input i7, i6, i5, i4, i3, i2, i1, i0,
3     output y2, y1, y0
4 );
5     assign y2 = i7 | i6 | i5 | i4;
6     assign y1 = i7 | i6 | (~i5 & ~i4 & i3) | (~i5 & ~i4 & i2);
7     assign y0 = i7 | (~i6 & i5) | (~i6 & ~i4 & i3) |
8         (~i6 & ~i4 & ~i2 & i1);
9 endmodule

```


题目 6: 分析 Verilog 代码, 可以知道 s_1, s_2 的表达式为:

$$s_1 = \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot \bar{c} + a \cdot b \cdot c = (a \oplus b) \oplus c$$

$$s_2 = \bar{a} \cdot b \cdot c + a \cdot \bar{b} \cdot c + a \cdot b \cdot \bar{c} + \bar{a} \cdot \bar{b} \cdot \bar{c} = \overline{(a \oplus b) \oplus c}$$

得出真值表如下表. 可以看出, 当 a, b, c 中有奇数个 1 时, s_1 为 1, s_2 为 0; 当 a, b, c

表 1: Verilog 代码真值表

| a | b | c | s_1 | s_2 |
|-----|-----|-----|-------|-------|
| 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 |

中有偶数个 1 时 (或全为 0 时), s_1 为 0, s_2 为 1. 因此, 该 Verilog 代码的功能是统计 a, b, c 中 1 的个数是奇数还是偶数. 对应的电路图如图 .

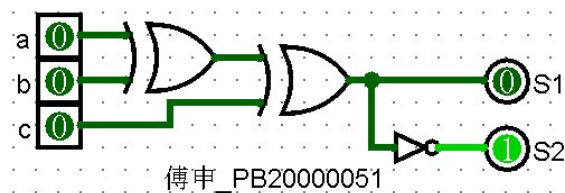


图 8: Verilog 代码对应的电路图

【总结与思考】

收获 了解了在 Logisim 中如何通过真值表和表达式生成组合逻辑电路以及如何简化电路, 入门了 Verilog HDL 语言.

难易程度与任务量 较轻松.

建议 暂无.