**实验10**

**组合逻辑电路（上）**

# **实验介绍**

在之前的实验中，我们已经完成了对于实验语言（Verilog）以及实验环境（Vivado + FPGA）的学习。从本次实验开始，我们将正式进入数字电路实验的设计部分。在数字电路中，根据逻辑功能与电路结构的不同，我们可以将数字电路分成两大类：组合逻辑电路与时序逻辑电路。Lab4 将主要介绍组合逻辑电路的概念、原理、设计与应用。

组合逻辑电路是由多个逻辑门（例如与门、或门、非门等）组成的电路。从逻辑功能上说，其任意时刻的输出仅仅取决于该时刻的输入，而与电路曾经的状态无关。换而言之，对于某一输入，电路会给出固定的输出，且输入变化后输出内容也会发生变化（不考虑门电路的延迟）。因此，组合逻辑电路可以看作是纯粹的逻辑运算器，输出可以被由输入组成的逻辑表达式确定。

组合逻辑的特点可以总结如下：

* 只由逻辑门和电线构成，没有存储器件；
* 输出仅与输入有关，不存在任何时序关系；
* 逻辑门的输出可以直接连接到其他门的输入端；
* 不会产生任何形式的反馈，每个输出信号只由与之直接相连的输入决定。

组合逻辑电路在数字电路中应用极为广泛，常见的组合逻辑电路包括：逻辑门、选择器、加法器、编码器、译码器等。本次实验里，我们将带大家系统学习组合逻辑电路的相关知识，掌握组合电路设计的相关技巧。

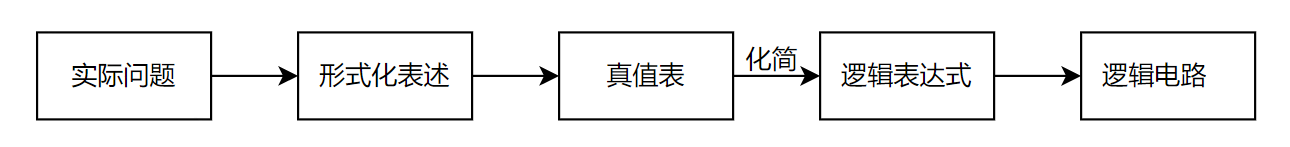
# **实验目标**

1. 通过进一步学习模块化的思想，能够将一些常见的组合逻辑电路用 Verilog 硬件描述语言实现
2. 学习组合逻辑电路分析的基本技巧
3. 学习组合逻辑电路设计的基本技巧

# **实验内容**

## 1 传统设计流程

首先，我们将介绍组合逻辑电路的传统设计流程，即从实际问题出发，如何设计并得到最终可以解决问题的电路结构。一般而言，针对特定的问题，设计组合逻辑电路的最基本的流程如图所示：



流程图中展示的方法一般适用于简单的、单模块的组合逻辑电路设计。随着 Verilog 编程能力的提升，我们往往会使用数据流或行为级描述直接实现模块的功能，而不会完整地执行上述的流程。但这需要我们对于 Verilog 语言对应的电路结构了如指掌。作为一名初学者，我们依然有必要掌握最为基础的设计流程。

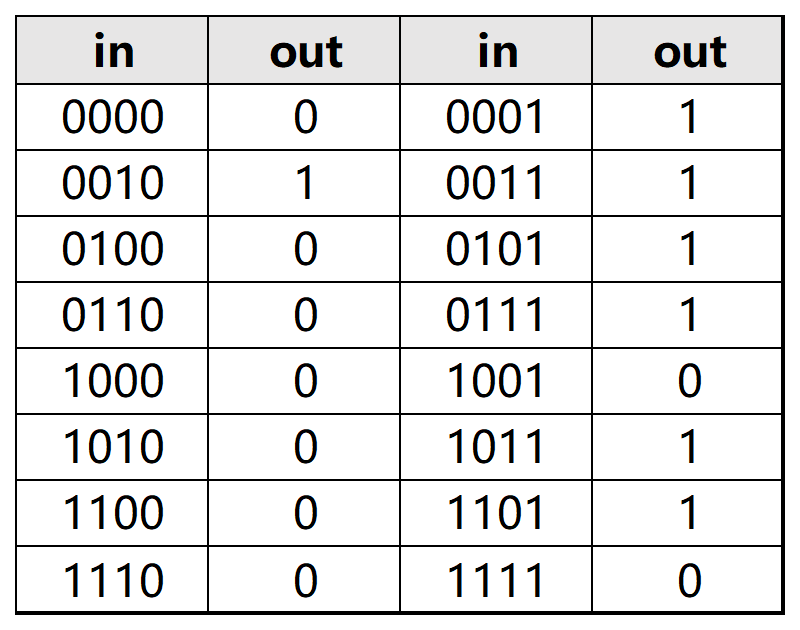
在得到逻辑表达式后，我们便可以开始进行 Verilog 代码的编写工作。接下来，我们将从一个实际问题入手，带大家体验组合逻辑电路的设计过程。

某同学现在希望你能够编写一个模块，用于检测输入的数据是否为素数。简单起见，我们假定待测结果位宽为 4，以二进制编码格式输入。现在需要设计一个硬件模块，对于给定的输入 in，输出 out 表示其是否为素数。特别地，该同学要求你「将 1 也视为素数」。一个可行的模块端口定义如下：



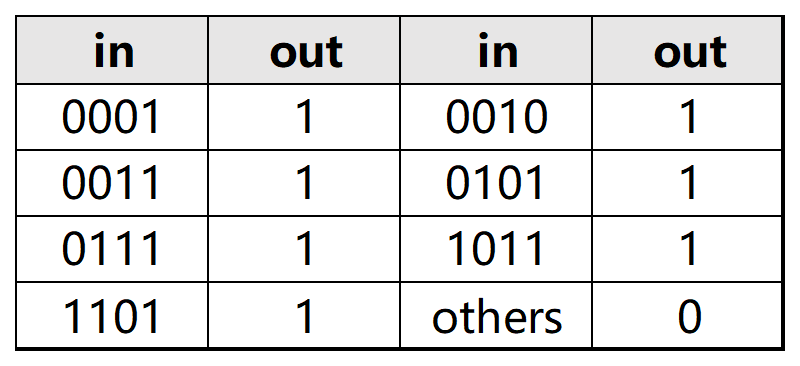
### ****1.1 真值表****

针对上面已经形式化表述好的问题，我们首先尝试列出其对应的真值表。真值表记录了每个输入组合对应的输出结果，适用于输入情况有限的问题，其特点是直观、全面。上面这个例子的真值表如下：

[](https://soc.ustc.edu.cn/Digital/lab4/figs/combinational/prime_tf.png)

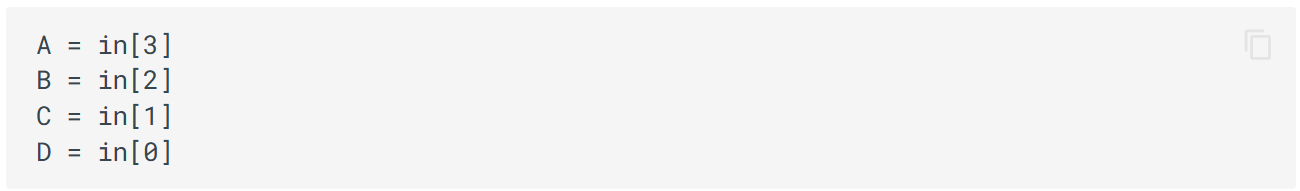
输入 in 为待测数据的 4 bits 二进制原码，输出 out 为 1 时代表输入数据为素数或 1。

这张表包含了所有的输出结果，而我们更感兴趣 out 为 1 的表项，因此我们可以将其单独提出，得到一个简化版的真值表：

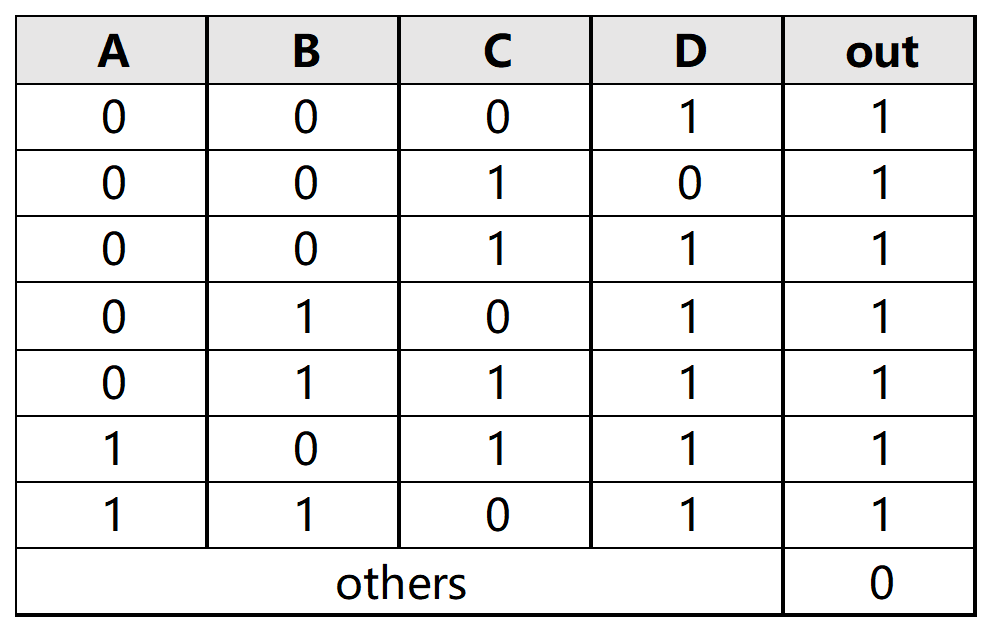
[](https://soc.ustc.edu.cn/Digital/lab4/figs/combinational/prime_tf_2.png)

### ****1.2 逻辑表达式****

根据真值表，我们就可以得到该电路的逻辑表达式了，只需要将为 1 的项使用逻辑运算符表示出来即可。为了便于区分，我们记



上面的真值表就可以改写为：

[](https://soc.ustc.edu.cn/Digital/lab4/figs/combinational/prime_tf_3.png)

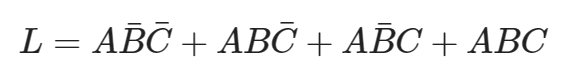
这样，我们就可以根据每一行，列出下面的逻辑表达式



这个式子一共有七项，每一项对应着真值表中为 1 的一行。不难验证，任何ABCD 的组合输入均可以得到 out 的正确结果。

### ****1.3 卡诺图****

通过真值表得到的逻辑表达式是正确的，但不一定是最简的。例如：



实际上等价于 L=A。如何尽可能地化简得到的逻辑表达式呢？这就需要借助卡诺图了。

卡诺图是一种图形表示法，由美国工程师卡诺所发明。相比真值表，卡诺图更加简洁直观、灵活方便，但依然只适用于变量少的情况。

下面是我们之前素数真值表对应的卡诺图。直观来看，卡诺图是一张二维图表，横纵坐标均为输入变量的不同取值。

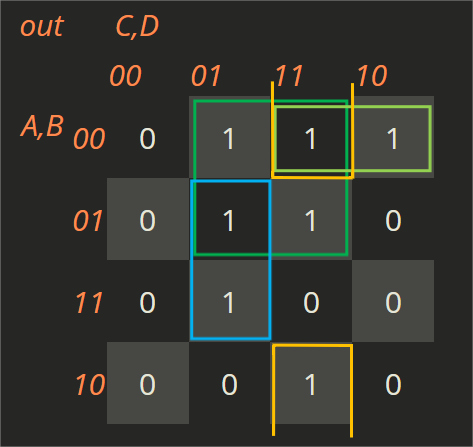
[](https://soc.ustc.edu.cn/Digital/lab4/figs/combinational/k1.png)

在这张图里，纵坐标为 AB、横坐标为 CD，对应的数值为 00、01、11、10。这样的排列保证了任何相邻的两个小方格在 ABCD 的坐标值上只有一位不同。例如，绿色框代表的输入为 ABCD=0101，蓝色框代表的输入为 ABCD=0111，仅有 C 一位不同。每一个小方格都与一种可能的输入对应。我们在每一个小方格中填入该输入对应的 out 值，也就得到了上图中的结果。

接下来是卡诺图最重要的一步：画圈。画圈需要满足下面的准则：

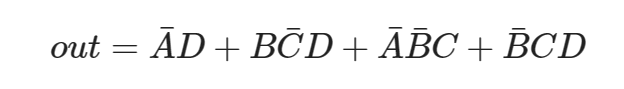
1. 需要包含所有为 1 的方格，不能包含为 0 的方格；
2. 圈可以重叠，但必须为矩形或正方形，且大小为 2 的幂；
3. 每个圈需要尽可能大。换而言之，圈的数目应当尽可能少。

按照上面的准则画出的圈就可以得到简化的真值表。例如，素数电路的卡诺图画圈之后就如下图所示：

[](https://soc.ustc.edu.cn/Digital/lab4/figs/combinational/k2.png)

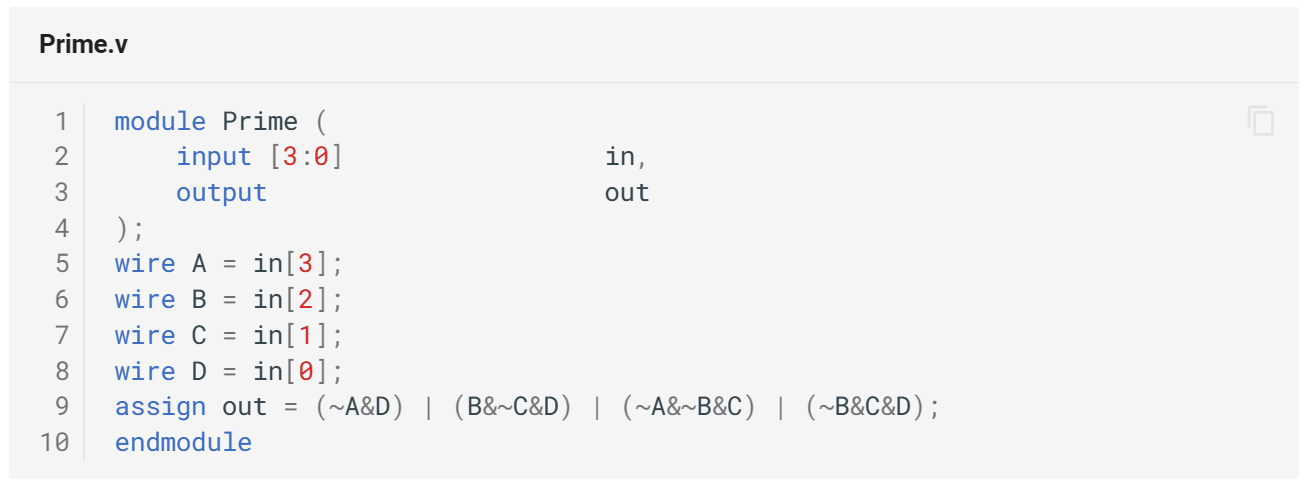
其中包含 0001、0011、0101、0111 的为 2 × 2 大小的圈，对应的输入情况为 AD = 01（B、C 的输入不影响结果）；包含 0101、1101 的圈对应的输入情况为 BCD = 101；包含 0011、0010 的圈对应的输入情况为 ABC = 001；包含 0011、1011 的圈对应的输入情况为 BCD = 011。

根据所画的圈，我们可以得到简化的逻辑表达式。



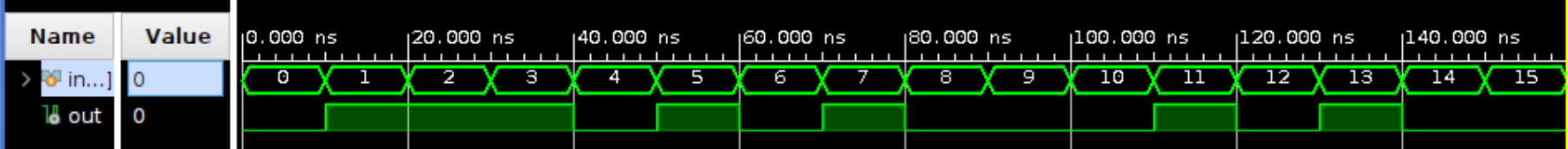
### ****1.4 实现****

根据简化的逻辑表达式，我们可以编写如下的数据流级 Verilog 描述代码：



运行仿真的结果如下：



[](https://soc.ustc.edu.cn/Digital/lab4/figs/combinational/prime.png)

从波形上看，其已经实现了正确的结果。

事实上，更复杂的情况下，我们往往难以列出完整的真值表，而是可以将代表特定功能的组合逻辑模块进一步组合以实现高级的功能，这时的程序设计体验就与高级语言更加接近了。例如，当我们将全加器组合成32位加法器时，我们是在从计算层面的「进位」去考虑连接方式，而非给32位加法结果的每一位去设定真值表进行连接。

## 2 Logisim自动电路搭建

在先前的实验中，我们可以使用 Logisim 管理窗中的各种组件以及自己设计的子电路，设计出各种功能的组合逻辑电路，但是在搭建电路时大量的拖拽、布局、连线非常费时费力。好消息是，Logisim 提供了两种自动化生成电路的方式：通过真值表生成、通过表达式生成。

### ****2.1通过真值表****

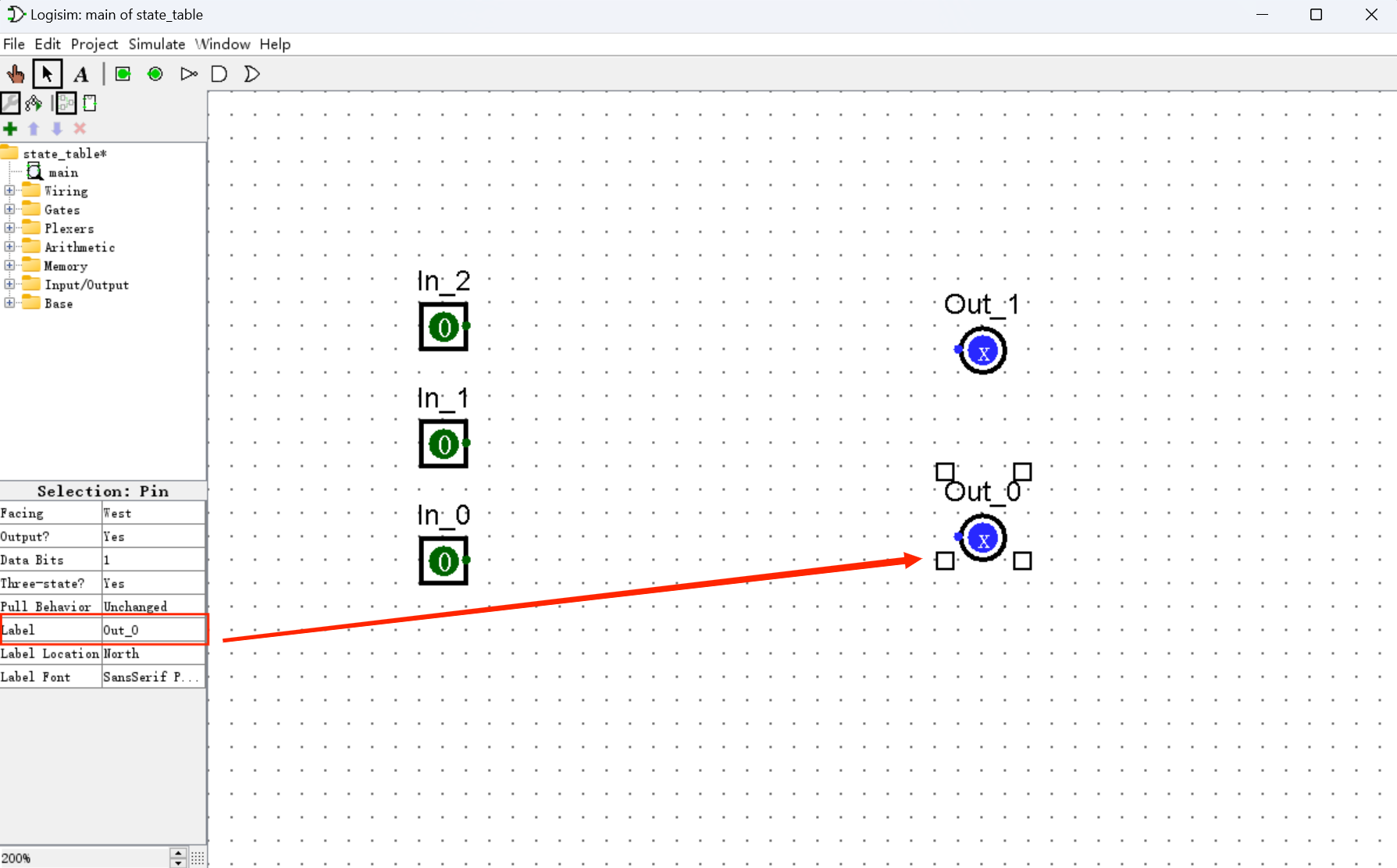
考虑一个 3bits 优先编码器，输入为 3bits，输出为 2bits。当输入均为 0 时，输出为 11，代表无输入；否则输出为当前的优先译码结果。

我们要设计该电路，一般的做法是：

* 根据真值表画出各输出项的卡诺图；
* 通过卡诺图写出各输出项的逻辑表达式；
* 根据逻辑表达式画出电路图，完成电路设计。

一般真值表中没有列出的输入项，对应的输出视为 x（即 0、1 均可）。Logisim 能够帮我们完成上述步骤中大部分的工作。

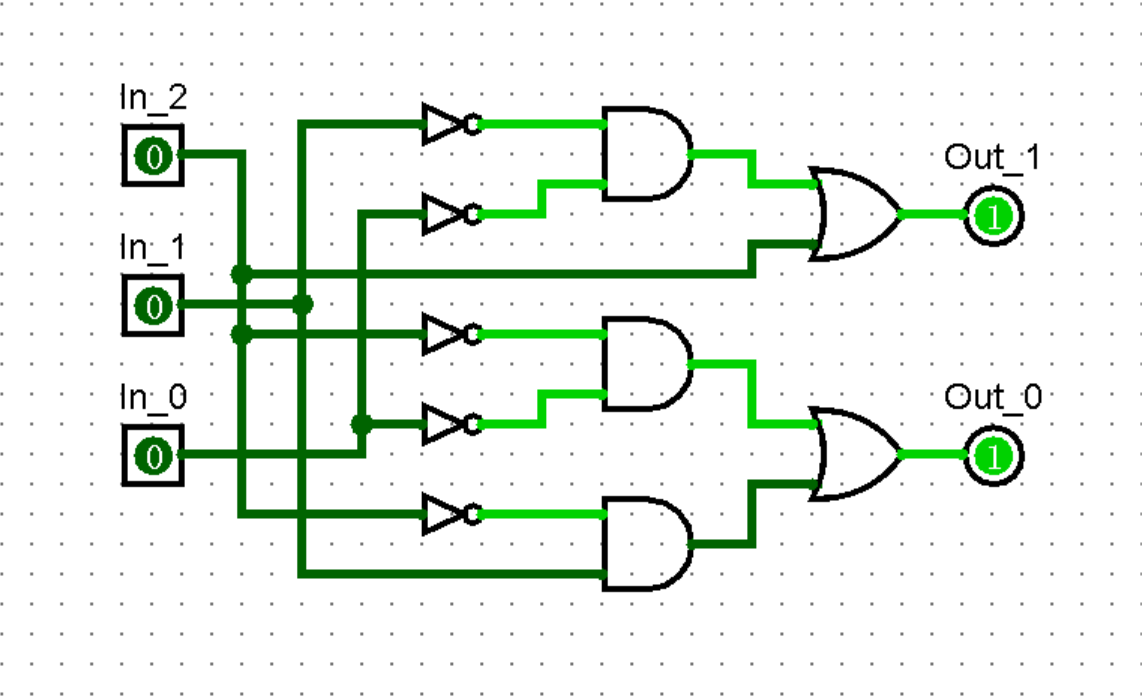
首先在 Logisim 中新建一个电路图（电路名称自拟），在电路图中放置输入引脚，有几个输入就放几个引脚，按同样的方式放置输出引脚。放置完毕后，给所有引脚标上标号，并按高低位顺序排列。对于表 1，我们的输入引脚为三个，输出引脚为两个。完成后的效果如下图所示。

[](https://soc.ustc.edu.cn/Digital/lab4/figs/logisim/log_example1.png)

接下来，在菜单栏的 Project 选项卡中找到 Analyze Circuit 选项并选中。在弹出的窗口中选择 Table 选项，按照前面的真值表修改输出值（鼠标点击输出信号对应的叉号就可修改），最后点击 Build Circuit 便可生成电路（弹出的对话框都选择是）

|  |  |
| --- | --- |
| [IMG_257](https://soc.ustc.edu.cn/Digital/lab4/figs/logisim/log_example2.png)打开 Analyze Circuit 窗口 | [IMG_258](https://soc.ustc.edu.cn/Digital/lab4/figs/logisim/log_example3.png)修改输出值 |

生成的电路如下图所示。

[](https://soc.ustc.edu.cn/Digital/lab4/figs/logisim/log_example4.png)

### ****2.2 通过表达式****

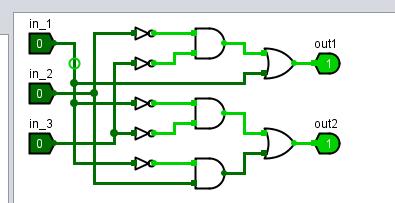
通过真值表生成电路确实能为我们减少工作，但是也存在不足之处，我们知道，真值表条目数与输入项个数呈指数相关，当输入信号数量较多时，编辑真值表也是一项非常繁重的工作。例如：一个 8 位的优先编码器电路，其完整的真值表有 256 项之多。此时，我们可以选择通过表达式生成电路。

回到 3bits 优先编码器的例子。首先在 Logisim 中新建一个电路图（电路名称自拟），在电路图中放置输入引脚，有几个输入就放几个引脚，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。完成后的效果如图所示。

[](https://soc.ustc.edu.cn/Digital/lab4/figs/logisim/yinjiao.png)

接下来，在菜单栏的 Project 选项卡中找到 Analyze Circuit 选项并选中。在弹出的窗口中选择 Expression 选项，按照相应的接口填入各输出项的逻辑表达式，最后点击 Build Circuit 生成电路：

|  |  |
| --- | --- |
| [IMG_257](https://soc.ustc.edu.cn/Digital/lab4/figs/logisim/expression.png)打开 Analyze Circuit 窗口 | [IMG_258](https://soc.ustc.edu.cn/Digital/lab4/figs/logisim/expression_1.png)修改输出值 |

[](https://soc.ustc.edu.cn/Digital/lab4/figs/logisim/expression_done.png)

# 思考与练习

1. 请使用行为级描述的方式重新编程实现 Prime 模块。
2. 请设计一个组合逻辑电路，用于将二进制数字转换为对应的 8421BCD 编码。