实验一 组合逻辑设计

**一、实验目的**

1 掌握Verilog语言和Vivado、Logisim开发平台的使用；

2 掌握基础组合逻辑电路的设计和测试方法。

**二、实验内容（用Logisim或Vivado实现）**

1 基础门电路（多输入门电路、复用器等）的设计和测试；

2 基础功能模块（编码器、译码器等）的设计与测试。

**三、实验要求**

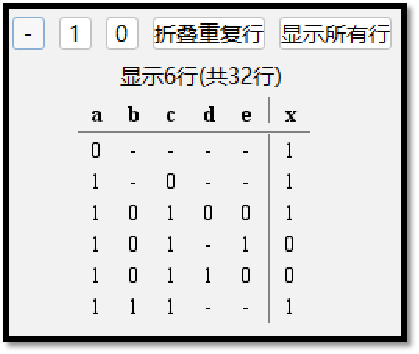
1 掌握Vivado或Logisim开发工具的使用，掌握以上电路的设计和测试方法；

2 记录设计和调试过程（Verilog代码/电路图/表达式/真值表，Vivado仿真结果，Logisim验证结果等）；

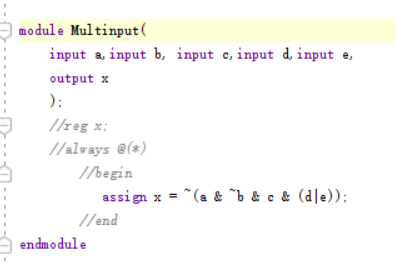
3 分析Vivado仿真波形/Logism验证结果，注重输入输出之间的对应关系。

**四、实验过程及分析**

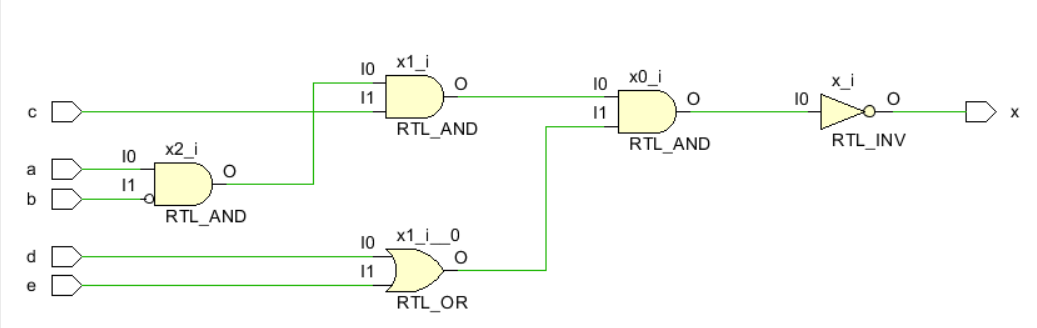
1 多输入门电路

实例的多输入门电路实现的组合逻辑表达式为x=~(a⋅~b⋅c⋅(d+e))，简化真值表为

设计实现代码

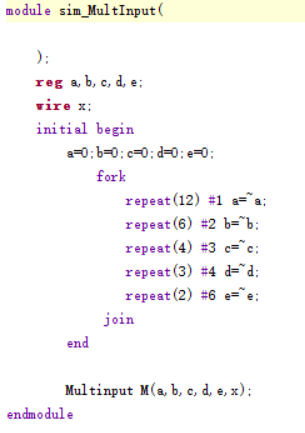
-->

RTL电路图

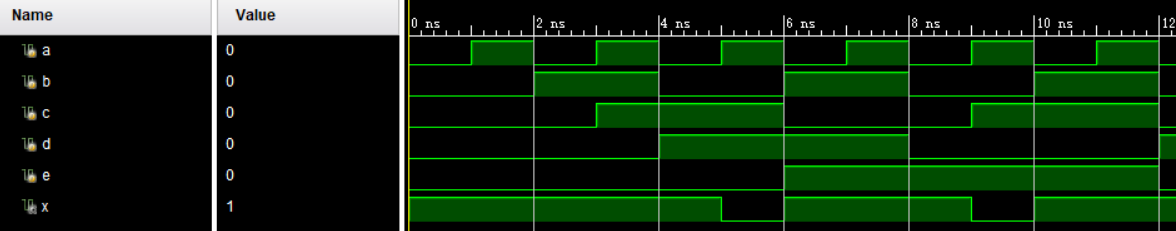
-->

各逻辑门线路连接与逻辑表达式相符。仿真的代码为：

-->



这里5个输入端周期性变化，仿真结果：

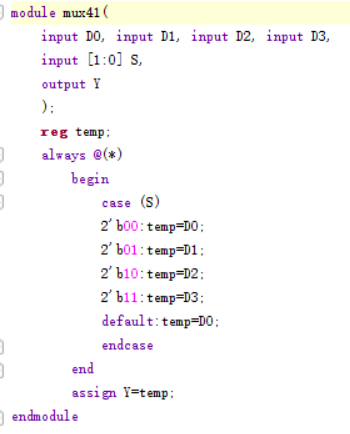


显然波形与上述真值表是吻合的。

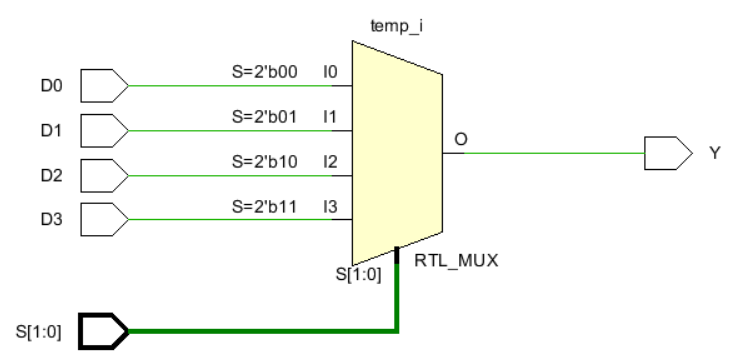
2 四路复用器

四路复用器有4个独立的输入信号以及一个2位的选择信号，根据选择信号分别为00，01，10，11在4个信号中选择对应的一个输出，其余被阻塞。用case语句实现：

-->

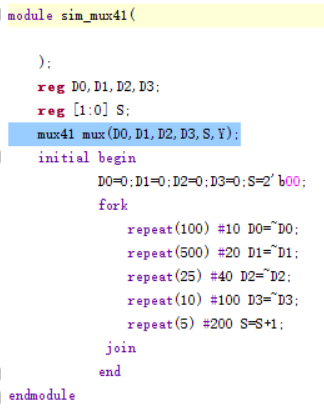


RTL电路图：

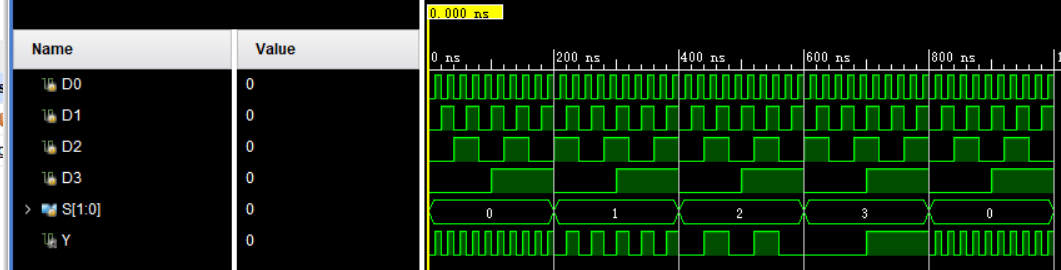


仿真代码：

-->



这里四个信号D0，D1，D2，D3是频率不同的周期性变换波形，选择信号周期性依此选择各路信号进行输出。仿真波形的结果是：

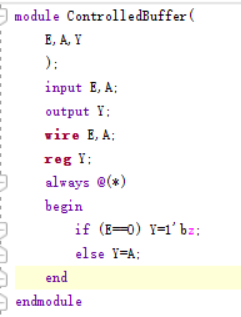


可以看出四路复用器实现了在思路信号中选择输出，结果符合设计预期。

也可以用三台缓冲器实现。三态缓冲器在使能信号E有效时将输入信号A输出；在E无效时，会输出一个高阻态z。这样，两个三态门分别接入两个输入信号，而两个使能端一个直接与选择信号sel连接，另一个通过反相器连接sel，就可以实现二路复用器。二路复用器级联实现四路复用器。

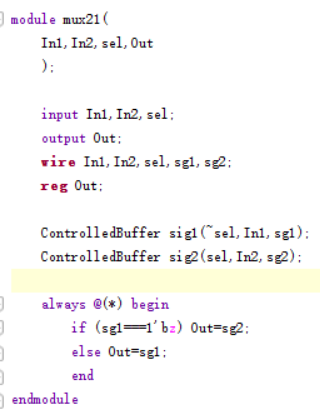
三态门设计代码：

-->



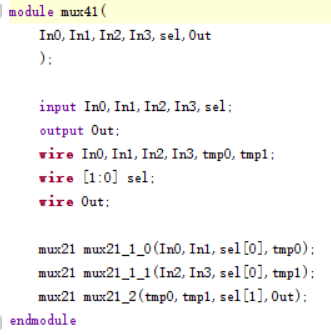
二路复用器：

-->

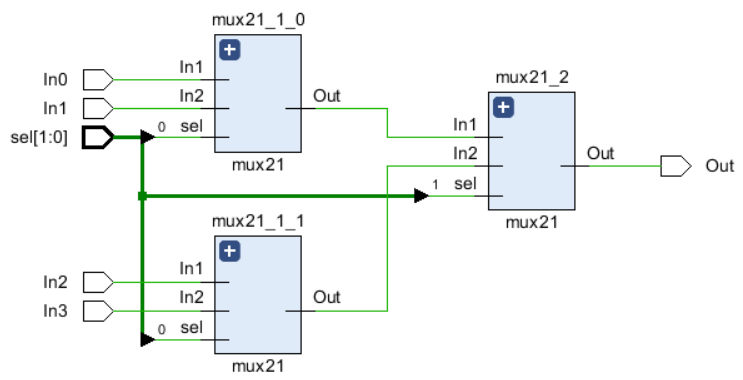


四路复用器：

-->

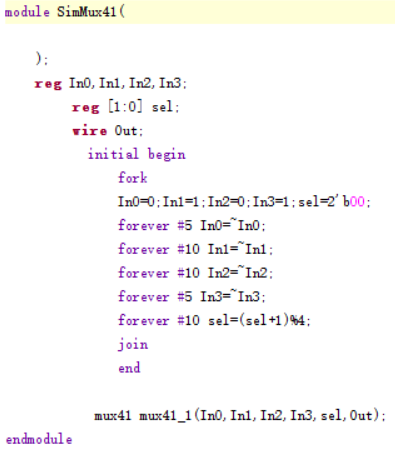


四路复用器的RTL电路图：

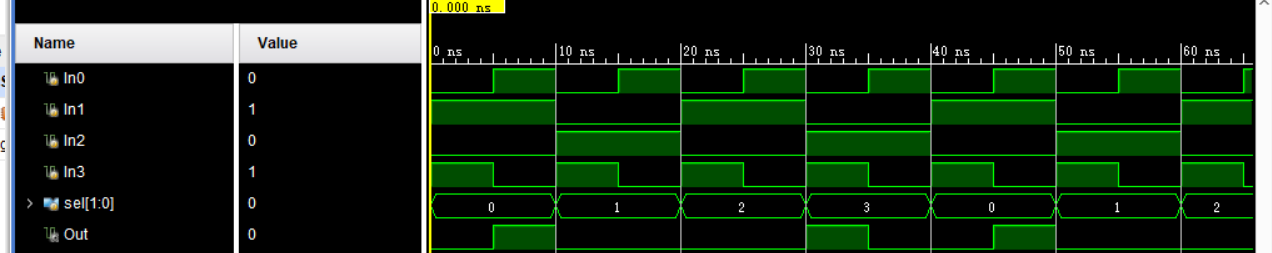


仿真代码：

-->



同样是在4个不同特性的信号中依此选择一个进行输出。仿真波形图：

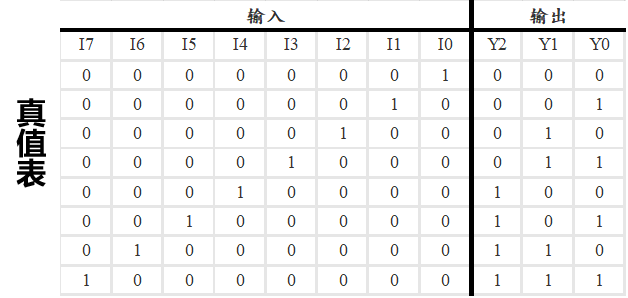


这一结果也是符合设计预期的。

3 编码器

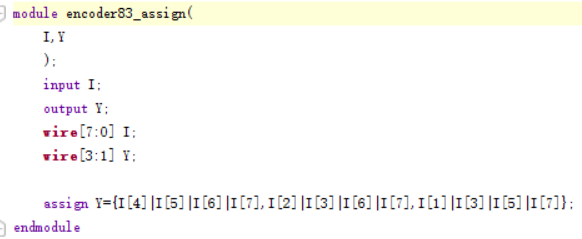
83编码器输入一个8位二进制数，其中仅有一位为1，输出一个3位二进制数，即1出现的位置用二进制表示。真值表：

-->

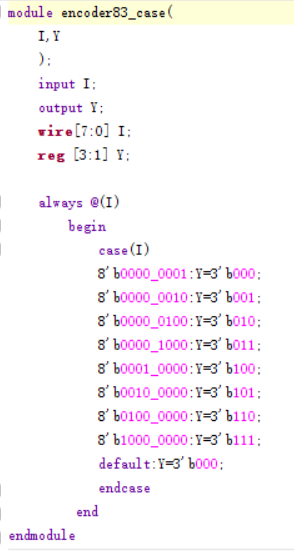


分别用asisign语句和case语句实现：

-->

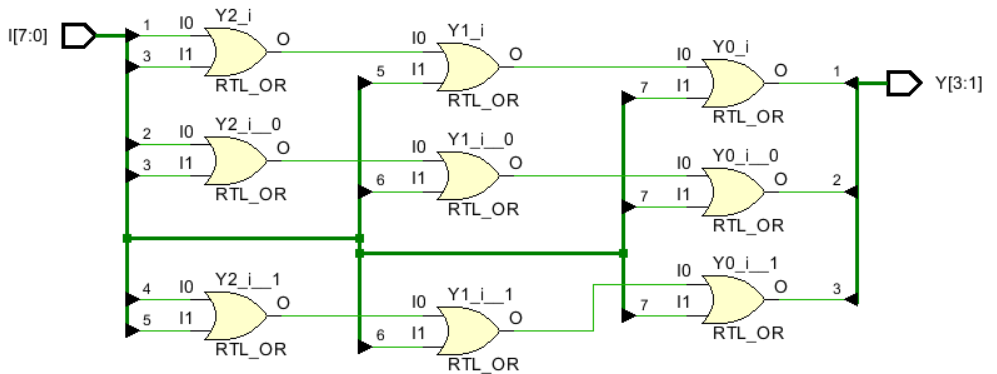


-->



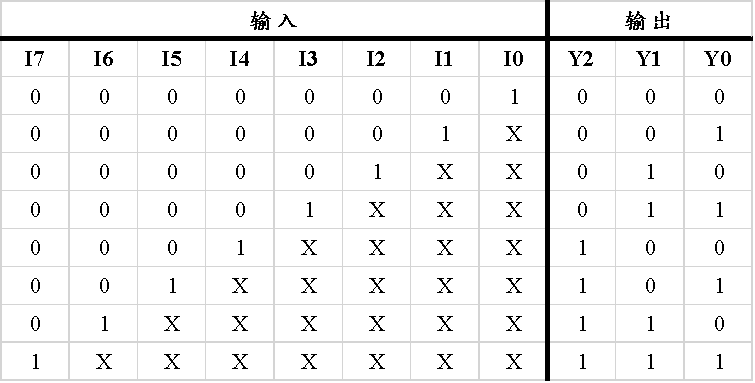
RTL电路图

-->



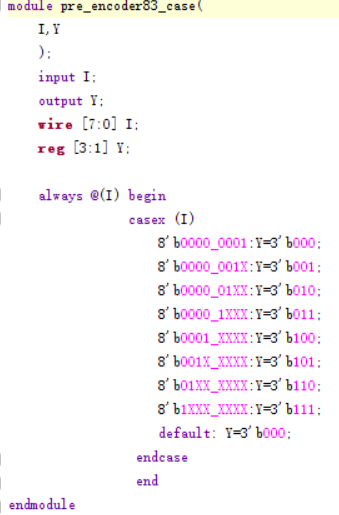
优先编码器在此基础上允许多位出现1，但只关心出现1的最高位，低位与输出结果无关。真值表：

-->

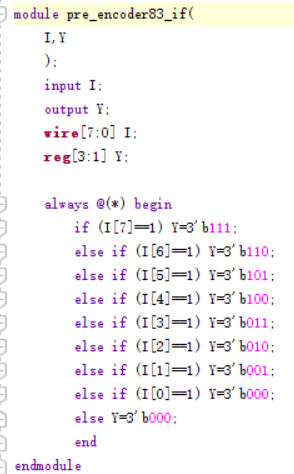


分别用case语句和if语句实现：

-->

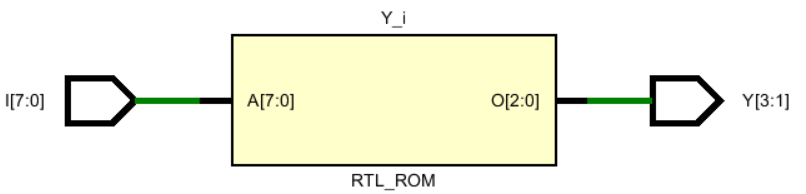


-->



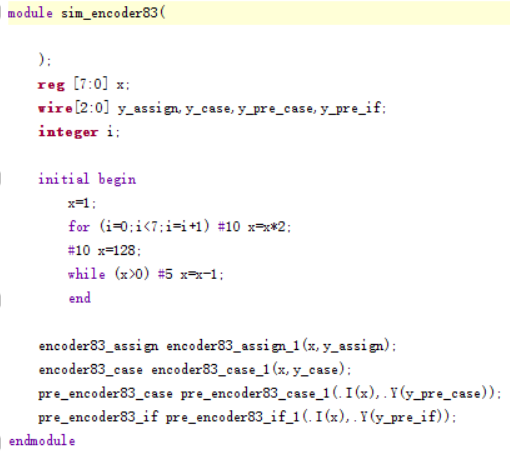
RTL电路图

-->

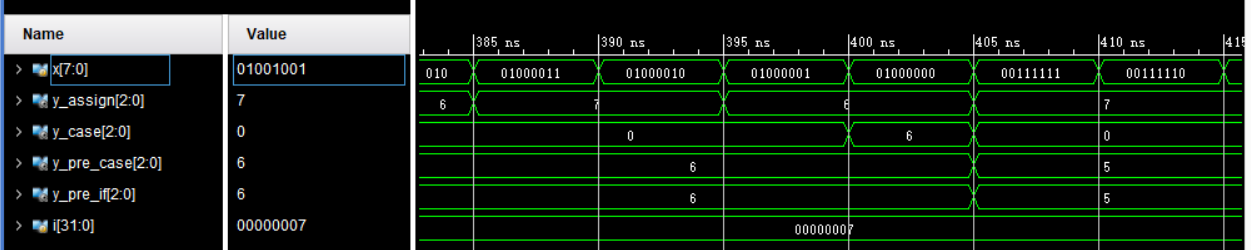
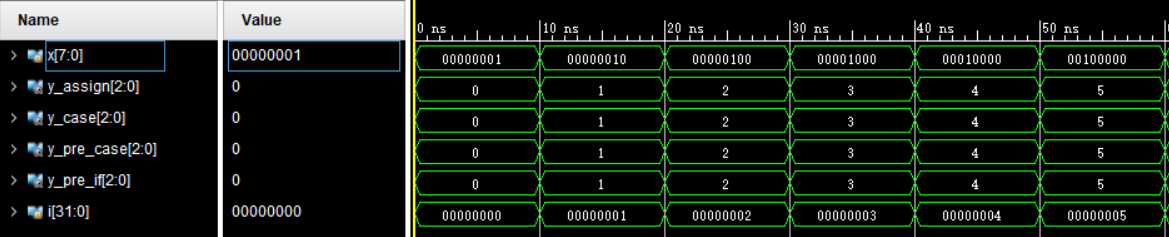


仿真代码

-



仿真波形结果

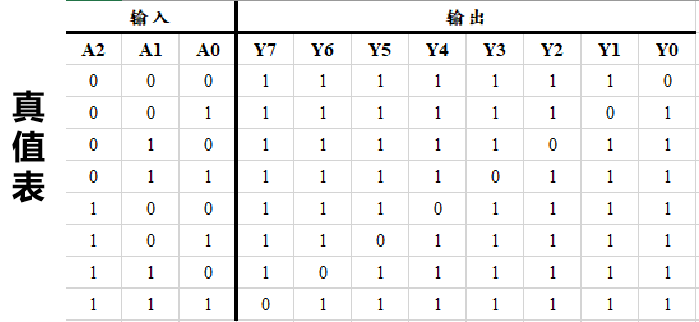


波形结果是符合设计预期的。在x只有一位是1时，4个编码器输出相同；在多位为1是，两个优先编码器都可以输出正确的结果，但普通的编码器输出结果是不正确的。

4 译码器

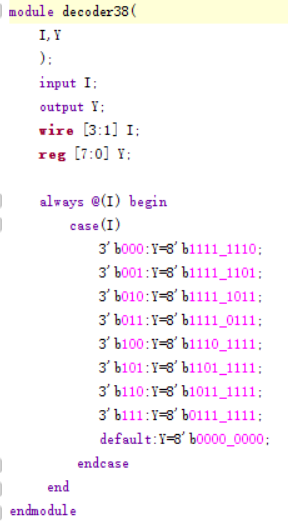
38译码器的工作原理与83编码器相反，输入3位二进制数，表示输出结果8位二进制数中0出现的位置，其他位置为0，真值表：

-->

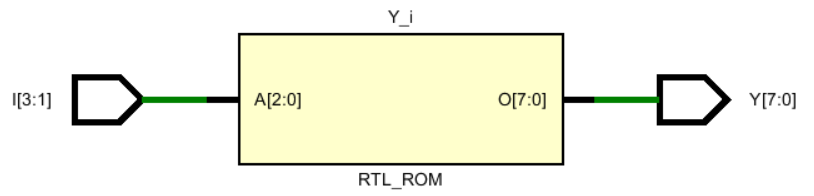


用case语句设计代码：

-->

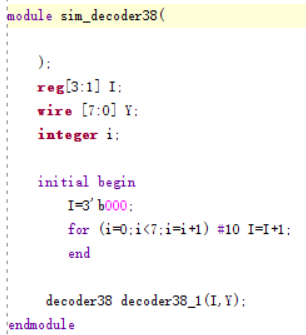


RTL电路图

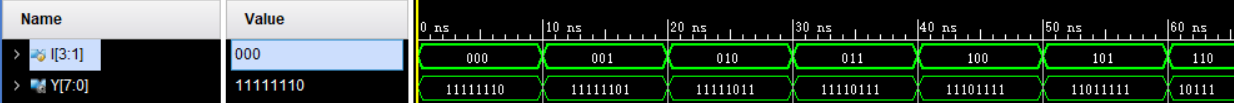


仿真代码

-->



输入信号从000增加到111，观察输出波形



输出波形符合设计预期。

**五、调试和心得体会**

1 在仿真代码设计输入信号时，要注意always代码块内是串行的，并行信号要用for…join的代码块结构。一个模块中每个代码块之间，以及调用模块都是并行的，这对控制信号的时序关系非常重要。

2 注意wire型数据和reg型数据的区别，在always代码块中进行赋值时，左边的变量必须是reg型，否则语法错误。