实验2 简单组合电路设计

* 任务描述
* 相关知识
* 实验步骤
* 编程要求
* 测试说明
* 源码
* Testbench代码
* 控制台输出和波形图
* 遇到问题和解决方法
* 实验心得、意见和建议

## 任务描述

本关需要你根据所学的仿真测试的知识，完成选择器、译码器等组合电路的设计，对电路进行测试。熟悉vivado工具的操作；学习、掌握用Verilog语言设计组合逻辑电路的方法；掌握仿真测试方法，学习编写testbench并利用波形图进行测试。

## 相关知识

测试平台（Testbench）是用于测试和验证设计的正确性的程序。编写Testbench的主要目的是对使用硬件描述语言设计的电路进行仿真验证，测试设计电路的功能甚至部分性能是否与预期的目标相符。

测试一个实际功能电路需要用信号发生器来向电路输入测试信号、用示波器来观察电路的信号输出是否正确。一个待测的Verilog HDL模块就相当于一个功能电路，用Testbench对它进行仿真测试需要给待测模块输入激励、获取输出响应并作判断。Testbench需要完成以下工作：

（1）产生仿真激励（波形）；

（2）将激励施加到被测试模块端口并收集其输出响应；

（3）将输出响应与期望值进行比较，以判断是否符合预期目标。

典型的测试平台主要内容包括：

`timescale 1ns/100ps //这里可适当指定仿真的“时间单位/时间精度”

module XXX\_tb; //Testbench模块，通常没有输入和输出端口

//局部reg、wire变量声明

//用initial和always等语句产生激励（波形）

//实例引用被测试模块（籍以将激励自动施加其上）

//监视输出并与期望值做比较

//结束testbench程序的运行

endmodule

其中许多内容书写的先后顺序不拘。

假若被测模块定义为

module M1(in1, in2, out1); //in1、in2为input端口，out1为output端口

则用来测试M1模块的Testbench模块，习惯上命名为M1\_tb，无输入无输出。Testbench声明局部reg、wire变量时，应该包括（但不限于）一批与被测模块端口对应（不妨就同名）的变量，便于后面实例引用M1模块。并且与input端口、output端口对应的变量分别声明成reg型和wire型。

## 实验步骤

请同学们根据实验任务细化实验步骤。

1. 编写如图2.1的2选1选择器电路的结构描述模块，并生成类似图2.2的原理图（RTL Analysis->Elaborated Design->Schematic）。

  
图2.1 2选1选择器电路

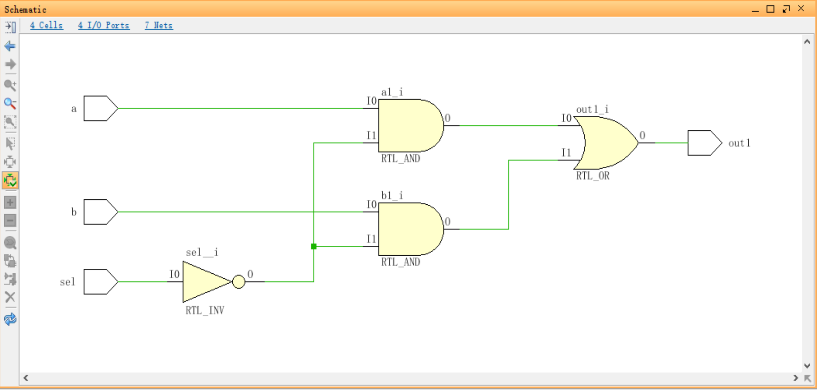


图2.2 2选1选择器Schematic

1. 编写2选1选择器电路的数据流描述模块，并生成Schematic。
2. 编写2选1选择器电路的行为描述模块，并生成Schematic。
3. 用2选1多路选择器构造3选1多路选择器。顶层模块有3个数据输入端口（u，v，w）、2个选择输入端口（s0，s1）和1个输出端口（m）。3选1多路选择器的电路和真值如图2-3所示。请编写模块，并生成Schematic。

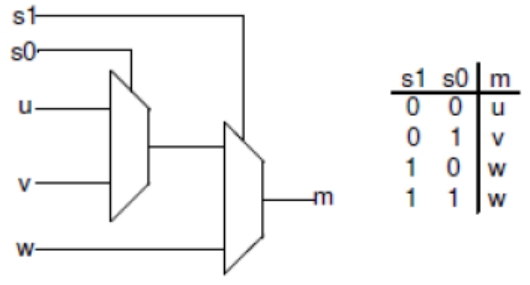


图2-3 3选1多路选择器电路和真值表

1. 设计一个3-8译码器模块，其真值表如表2.1所示。

模块请用以下格式：

module decoder\_38(F, CBA);

input [2:0] CBA;

output reg [7:0] F;

……

endmodule

表2.1 译码器真值表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | B | A | F7 | F6 | F5 | F4 | F3 | F2 | F1 | F0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

1. 为上述3-8译码器编写Testbench并进行测试。

## 编程要求

Testbench要能生成包含各种输入值和对应输出值的波形图，并在TCL控制台打印各种输入值和对应输出值。

## 测试说明

以下是测试样例。

【例】 五人投票表决器，过半数赞成则通过。仿真波形如图2.4所示。

module voter5(output pass, input vote);

wire [4:0] vote; //vote[i]表示第i人投票情况（1：赞成；0：反对）

reg pass; //最后结果（1：通过；0：不通过）

reg [2:0] count; //赞成票数

integer i;

always @(vote) begin

count = 0;

for (i = 0; i < 5; i = i+1) if (vote[i]) count = count + 1;

if (count >= 3) pass = 1; // 3人以上赞成，则 pass=1

else pass=0;

end

endmodule

//仿真测试Testbench模块

`timescale 1ns / 100ps

module voter5\_tb( );

wire pass;

reg [4:0] vote;

voter5 M(.pass(pass), .vote(vote));

initial begin

$display ("$time::[vote] [count] [pass]------");

$monitor ("%t::", $time, "[%b]\t[%d]\t[%b]", vote,M.count,pass);

end

initial begin

for (vote = 0; vote < 5'b11111; vote = vote + 1)

#2;

#2 $stop;

end

endmodule

//TCL控制台输出结果：

$time::[vote] [count] [pass]------

0::[00000] [0] [0]

2000::[00001] [1] [0]

4000::[00010] [1] [0]

6000::[00011] [2] [0]

8000::[00100] [1] [0]

10000::[00101] [2] [0]

12000::[00110] [2] [0]

14000::[00111] [3] [1]

16000::[01000] [1] [0]

18000::[01001] [2] [0]

20000::[01010] [2] [0]

22000::[01011] [3] [1]

24000::[01100] [2] [0]

26000::[01101] [3] [1]

28000::[01110] [3] [1]

30000::[01111] [4] [1]

32000::[10000] [1] [0]

34000::[10001] [2] [0]

36000::[10010] [2] [0]

38000::[10011] [3] [1]

40000::[10100] [2] [0]

42000::[10101] [3] [1]

44000::[10110] [3] [1]

46000::[10111] [4] [1]

48000::[11000] [2] [0]

50000::[11001] [3] [1]

52000::[11010] [3] [1]

54000::[11011] [4] [1]

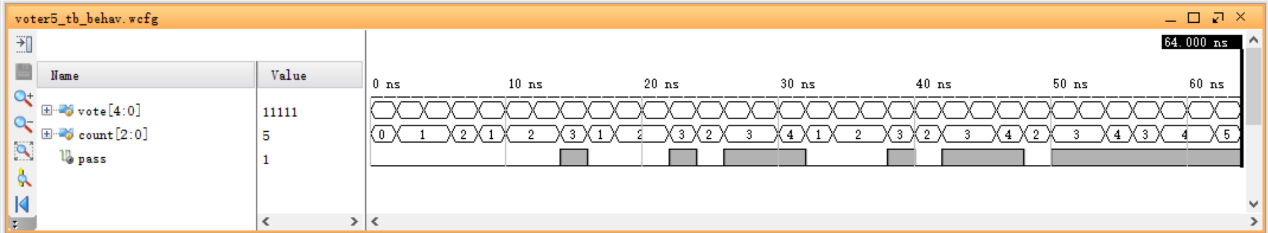
56000::[11100] [3] [1]

58000::[11101] [4] [1]

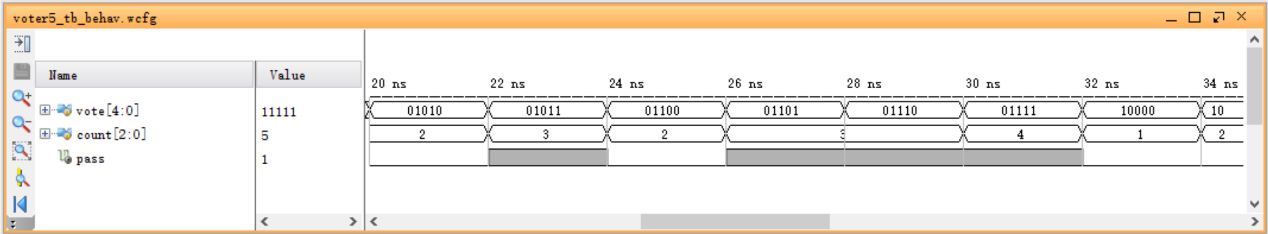
60000::[11110] [4] [1]

62000::[11111] [5] [1]

//波形图：



(a)波形



(b)波形（放大后）

图2.3 仿真波形图

## 源码

（要标明是什么具体任务对应的源码）

## Testbench代码

（要标明是什么具体任务、什么源码对应的Testbench代码）

## 控制台输出和波形图

（要标明是什么具体任务、Testbench代码对应的输出和波形，并作解读）

## 遇到问题和解决方法

(注：不限页数，可贴图，鼓励个性化描述，体现差异性。)

## 实验心得、意见和建议

(注：不限页数，鼓励对以上实验各部分内容提出自己的感想和建议。)