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

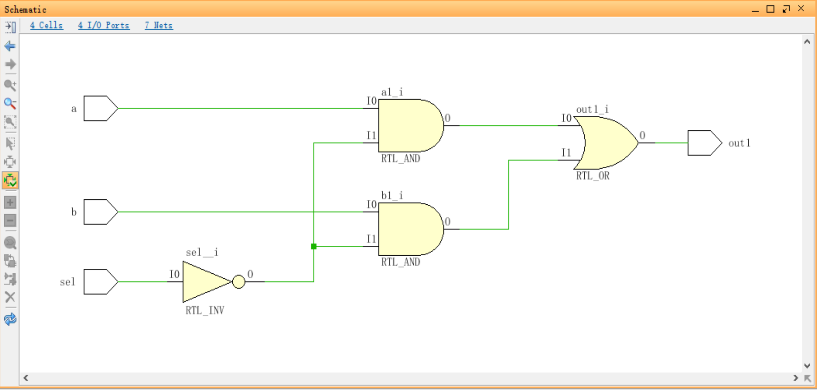


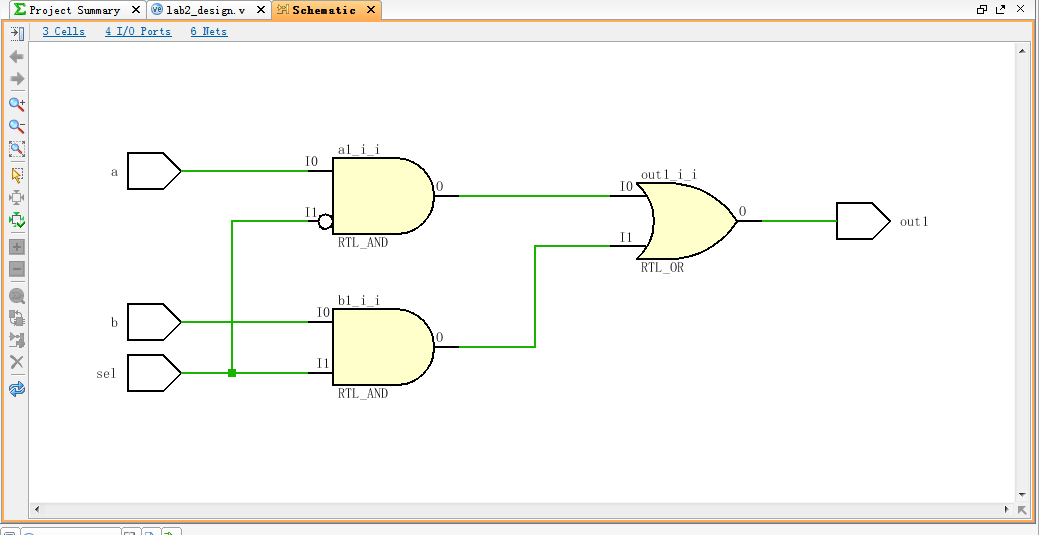
图2.2 2选1选择器Schematic

首先发现报告上的Schematic图给错了。

我的实验操作步骤：

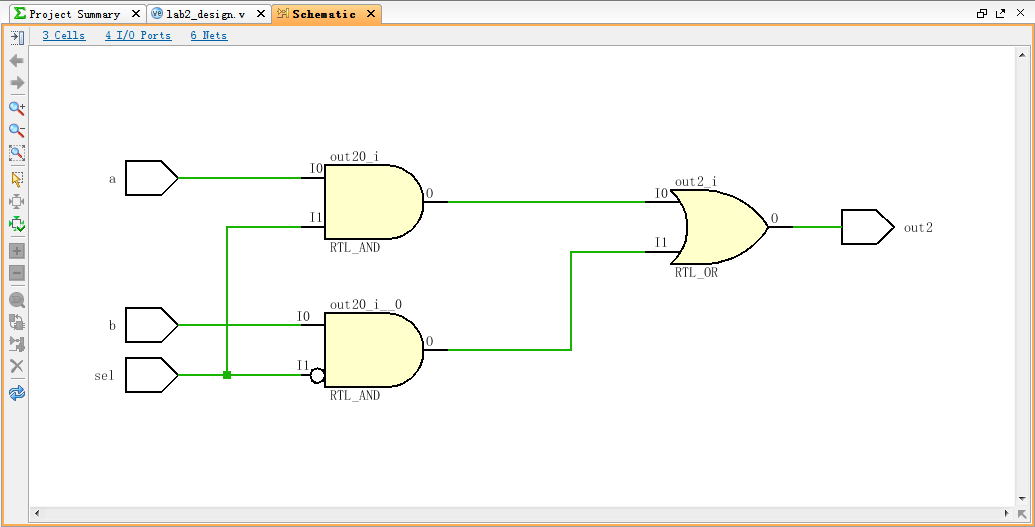
首先在vivado中创建一个空的工程，添加一个空的源文件lab2\_design，这里我还是选择的是和第一次试验相同的**XC7A100TCSG324-1**。

根据图2-1编写出结构描述模块，生成的Schematic图如下：



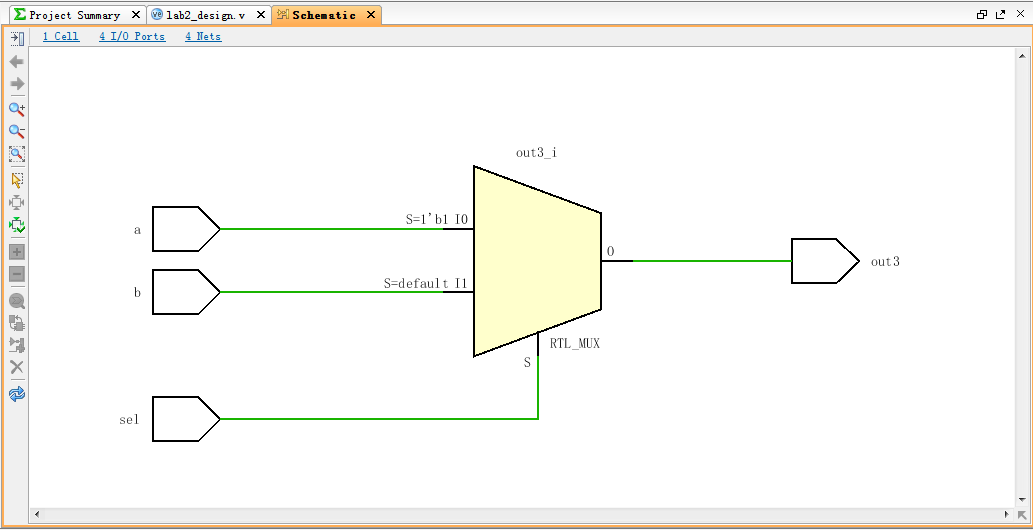
1. 编写2选1选择器电路的数据流描述模块，并生成Schematic。

根据图2-1编写出数据流描述模块，生成的Schematic图如下：



1. 编写2选1选择器电路的行为描述模块，并生成Schematic。

根据图2-1编写出行为描述模块，生成的Schematic图如下：



1. 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。

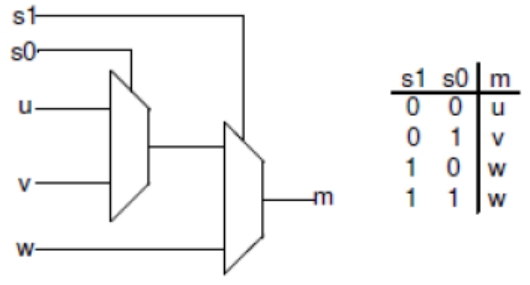
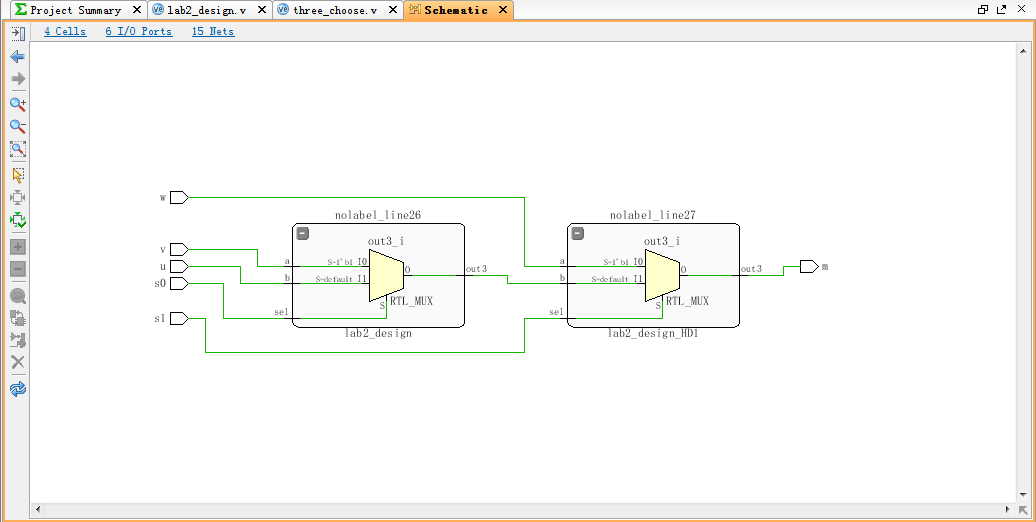


图2-3 3选1多路选择器电路和真值表

根据图2-3编写三选一多路选择器，这里我们是把两个文件放在同一个工程中，第二个便可以调用第一个模块。

生成的Schematic图如下：



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;

……

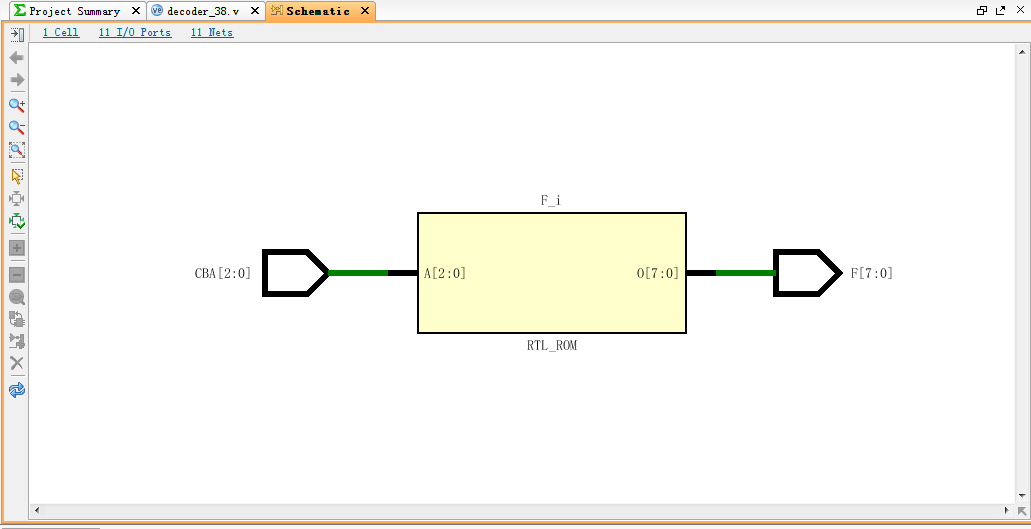
endmodule

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 |

根据所给提示要求生成的38译码器，源代码如下：

生成的Schematic图如下：



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

3-8译码器模块，这里使用行为描述通过循环来给输入赋值。

控制台输出：

$time::[in] [out]------

0::[000] [11111110]

20::[001] [11111101]

40::[010] [11111011]

60::[011] [11110111]

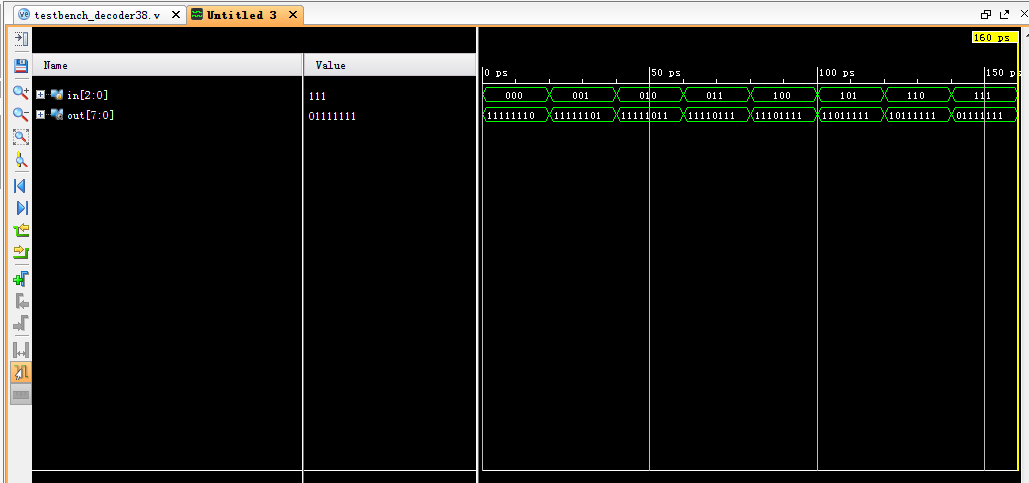
80::[100] [11101111]

100::[101] [11011111]

120::[110] [10111111]

140::[111] [01111111]

波形图：放大到比例适合：



## 源码

编写2选1选择器电路的结构描述模块

`timescale 1ns / 1ps

module lab2\_design(a,b,sel,out1);

input a,b,sel;

output out1;

wire nsel,a1,b1; //定义中间变量

not sel\_i(nsel,sel);

and a1\_i(a1,a,nsel);

and b1\_i(b1,b,sel);

or out1\_i(out1,a1,b1);

endmodule

编写2选1选择器电路的数据流描述模块：

`timescale 1ns / 1ps

module lab2\_design(a,b,sel,out2);

input a,b,sel;

output out2;

assign out2 = (a&sel) |(b&(~sel));

endmodule

编写2选1选择器电路的行为描述模块

`timescale 1ns / 1ps

module lab2\_design(a,b,sel,out3);

input a,b,sel;

output out3;

reg out3;

always @ (sel)

begin

if(sel == 1'b1)

out3 = a;

else

out3 = b;

end

endmodule

编写三选一多路选择器

`timescale 1ns / 1ps

module three\_choose(u,v,w,s0,s1,m);

input u,v,w,s0,s1;

output m;

lab2\_design(v,u,s0,temp1);

lab2\_design(w,temp1,s1,m);

endmodule

设计一个3-8译码器模块

`timescale 1ns / 1ps

module decoder\_38decoder\_38(F, CBA);

input [2:0] CBA;

output reg [7:0] F;

always @(CBA)

begin

case(CBA)

3'd0: F = 8'b1111\_1110;

3'd1: F = 8'b1111\_1101;

3'd2: F = 8'b1111\_1011;

3'd3: F = 8'b1111\_0111;

3'd4: F = 8'b1110\_1111;

3'd5: F = 8'b1101\_1111;

3'd6: F = 8'b1011\_1111;

3'd7: F = 8'b0111\_1111;

endcase

end

endmodule

（要标明是什么具体任务对应的源码）

## Testbench代码

这里是对于38译码器而写的testbench代码：

`timescale 10ps / 1ps

module testbench\_decoder38;

reg [2:0] in;

wire [7:0] out;

decoder\_38 test(.F(out), .CBA(in));

initial

begin

$display ("$time::[in] [out]------");

$monitor ("%t::", $time, "[%b]\t[%b]", in, out);

end

initial begin

for (in = 0; in <= 3'b111; in = in + 1)

#2;

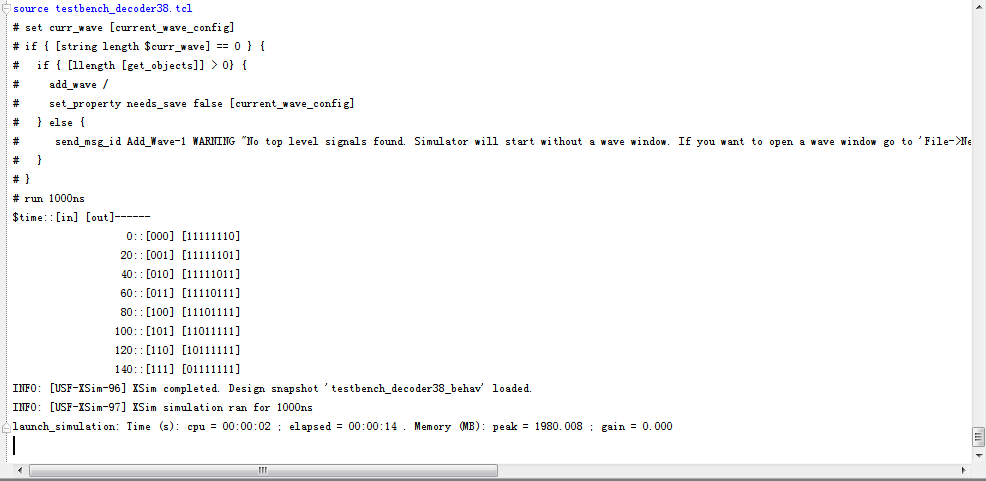
#2 $stop;

end

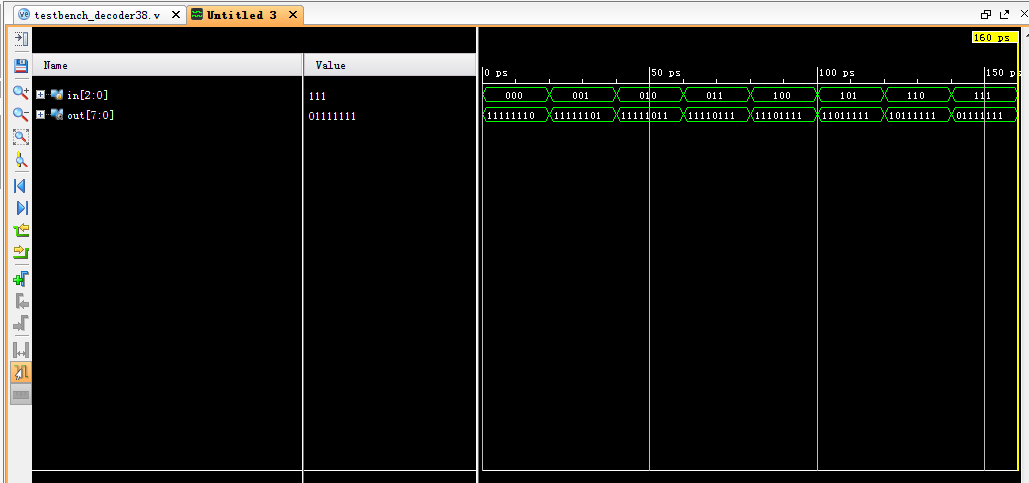
endmodule

## 控制台输出和波形图

这是控制台输出：



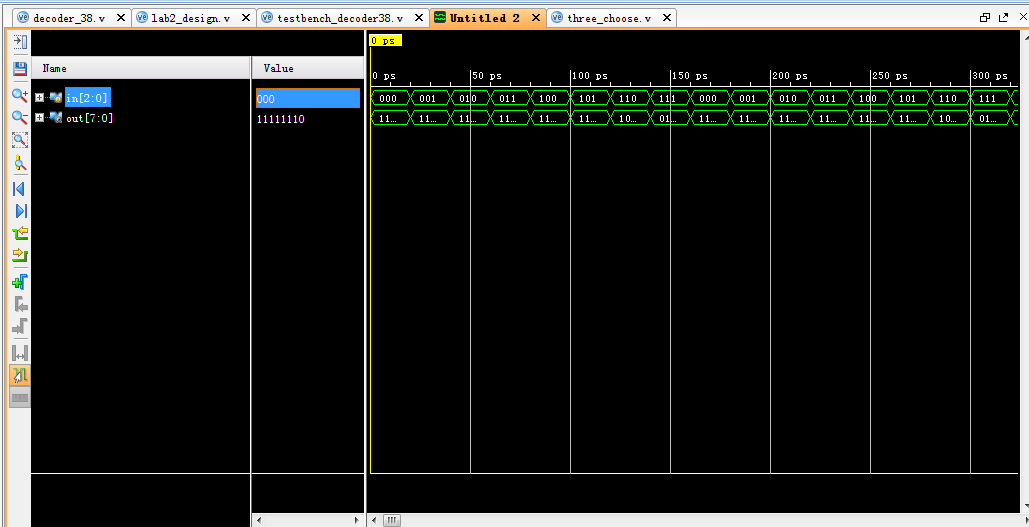
这是decoder\_38的输出波形：



修改循环条件后改为无限循环后的控制台输出：



修改条件后改为无限循环后的输出波形：



## 遇到问题和解决方法

这次实验，基本上除了前面四个模块是自己写的之外，testbench部分是使用老师的代码修改而得到的，因此遇到了报错，但是vivado有的报错会显示行数，但是有些报错就仅仅是报错，并不知道自己错在哪里，最后我只能重写了一遍。

编写前面几个的模块时候，在用多种方法描述的时候，我发现只能够把语句写成最简单的形式，例如a&b这种形式，系统才会使用与门这种基本电路来描述，否则直接就变成了个选择模块，并不知道内部的具体结构。

在生成Schematic图的时候，我发现必须把该文件set to top才能够生成该文件的Schematic图，此外，在一个模块中调用了该工程中的另外一个模块，会自动在source文件目录下看到生成了调用模块的子目录，方便我观察自己调用了哪些模块。

在第三次实验课上，老师单独对我的实验二进行了检查，并且在testbench文件中发现了我的问题，之前我设计for循环的时候，把判断条件直接设置成了3’b111，但是老师说希望我把循环条件改为<=3’b111这样的话就可以不断地重复输出译码值，修改后可以更加全面的看出来这是一个3-8译码器，进一步说明了这个系统在所有的输入可能情况下都可以正常工作。

这次试验的调试我开始也不是特别的熟练，在老师的指导下才知道看message框来寻找错误，但是有的时候报错只会在控制台上显示error，因此对于查看报错我才知道有两个地方可以看。

在一个工程中编写多个文件的时候，如果要生成一个设计文件的Schematic图，需要先把该文件set as top之后再查看原理图。在生成仿真的时候也是一样，需要把tb文件set as top。

## 实验心得、意见和建议

这次试验，编写的程序都是十分简单的，但是这也是我第一次动手编写verilog程序，对于一些语法还不是特别的熟悉，希望在之后的实验中能够逐渐熟悉起来。对于一些不是特别明确的报错还是尽量避免，对于仿真部分，下次可以考虑的更加详细，最好能找到自己程序可能产生的漏洞，比如万一发生了错误输入或是错误输出，最好有一个default的步骤，也可以在实际中避免一些考虑不周到的地方。