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与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

1. design 代码

```
module test1_1(
    a, b, c, d, e, x
    );
    input a, b, c, d, e;
    output x;
    assign x=~(a&(~b)&c&(d|e));//~(a?~b?c?(d+e))
endmodule
```

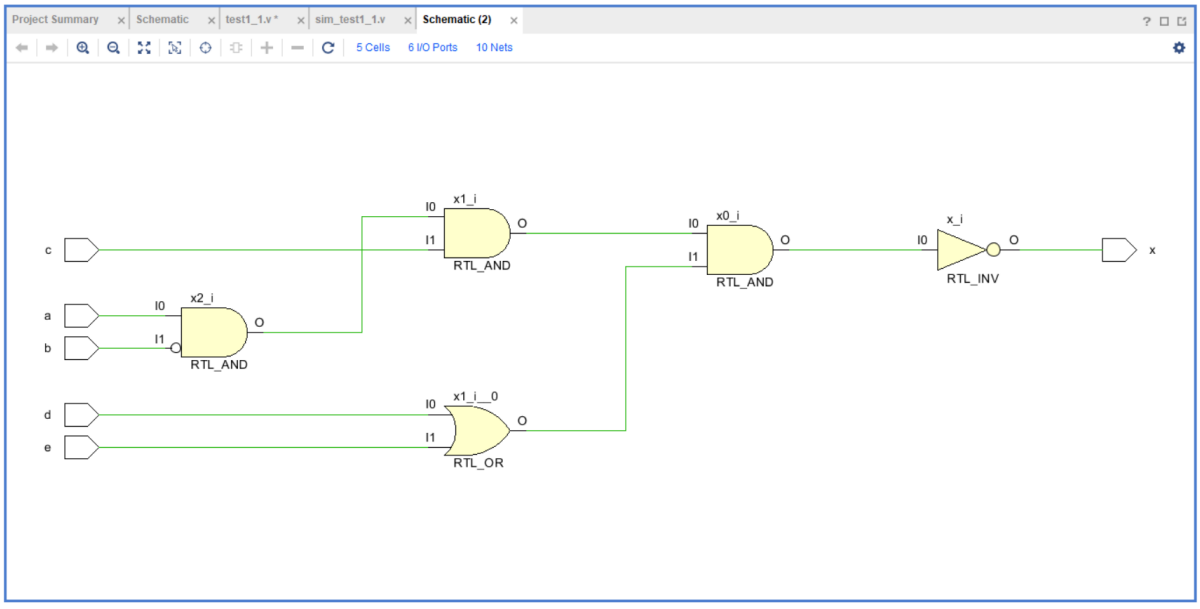
设置5个输入量 a,b,c,d,e ,1个输出量 x ,将输出 x 按照表达式 ~(a·~b·c·(d+e)) 设置为 x=~(a&(~b)&c&(d|e));

2. simulator 代码

```
22  `timescale 1ns/1ps
23  module sim_test1_1();
24      reg a,b,c,d,e;
25      wire x;
26      test1_1 test1_1(a,b,c,d,e,x);
27      initial
28      begin
29          a=0;b=0;c=0;d=0;e=0;
30          fork
31              repeat(100) #10 a=~a;
32              repeat(50) #20 b=~b;
33              repeat(25) #40 c=~c;
34              repeat(10) #100 d=~d;
35              repeat(5) #200 e=~e;
36          join
37      end
38  endmodule
39
```

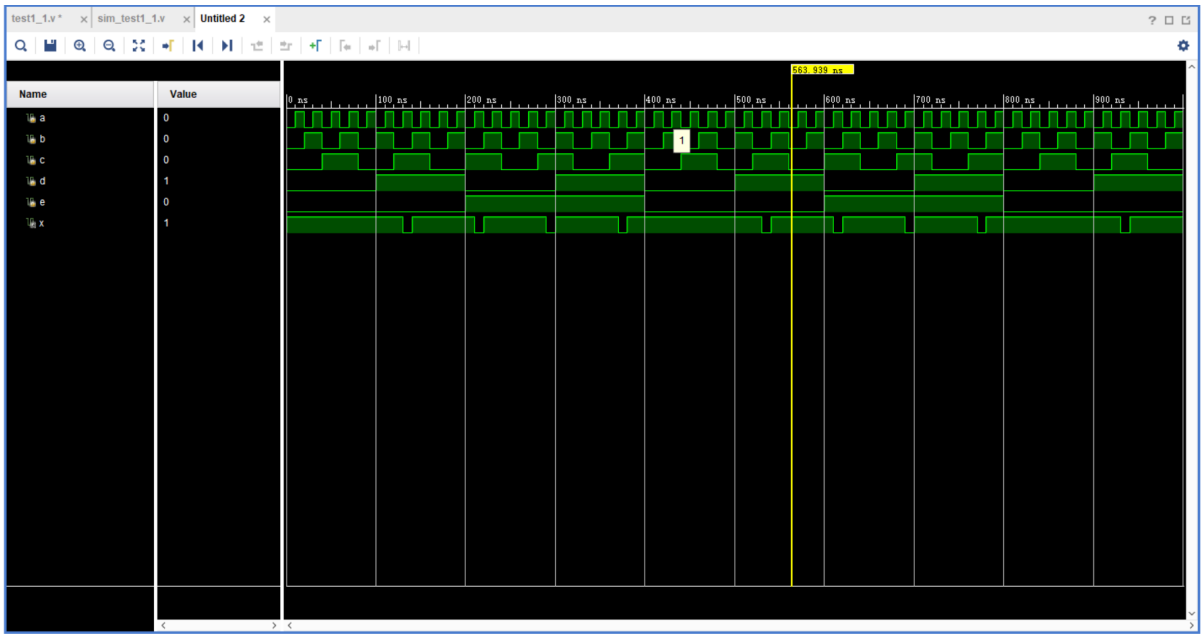
实例化design代码，利用fork并行块，设置5个输入信号初始为0，分别10，20，40，100，200ns取反一次

3. RTL 图



用两个与非门实现（a·~b·c），一个或非门实现（d+e），再通过一个与非门和非门实现 ~(a·~b·c·(d+e))

4. simulation图



5个输入信号初始为0，分别10，20，40，100，200ns取反一次，当 a=1,b=0,c=1,d+e>=1 时，输出信号x为高电平，与逻辑表达式匹配

2. 复用器

1. design 代码

```
module test1_2(
    D0,D1,D2,D3,S,Y
    );
    input D0,D1,D2,D3;
    input [1:0] S;
    output Y;
    reg temp;
    always@(*)
    begin
        case(S)
            2'b00: temp=D0;
            2'b01:temp=D1;
            2'b10:temp=D2;
            2'b11:temp=D3;
            default:temp=D0;
        endcase
    end
    assign Y=temp;
endmodule
```

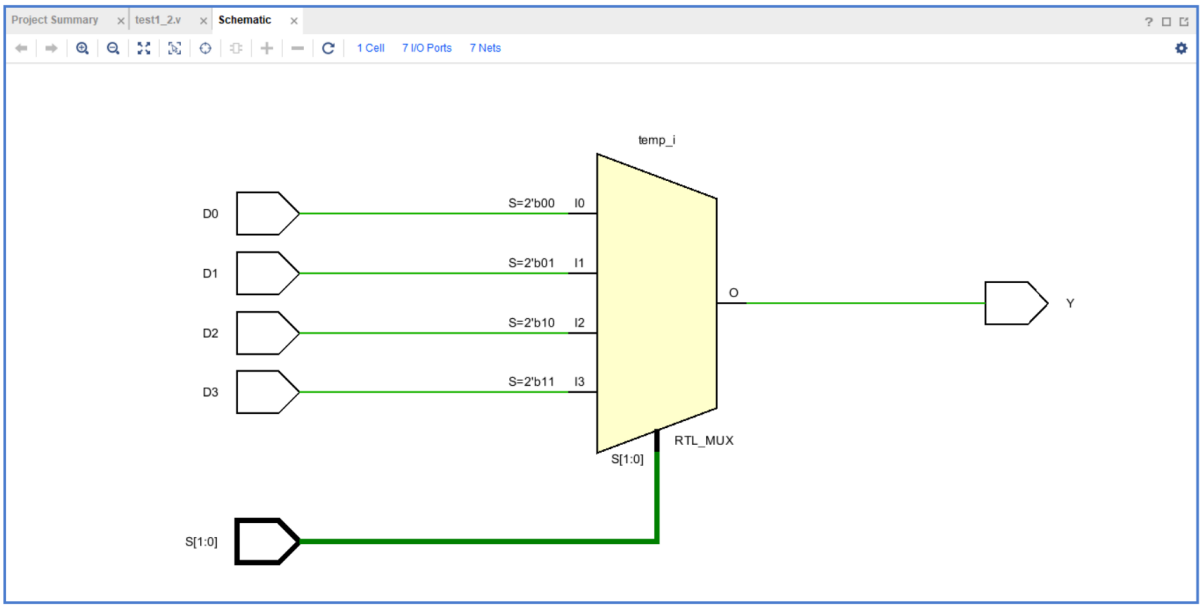
段Verilog代码表示一个2:1多路选择器。根据输入信号S的值，它选择其中一个数据输入（D0、D1、D2、D3），将其传递到输出Y。如果S的值不匹配任何情况，将默认选择D0。这个多路选择器的操作通过case语句和一个临时寄存器temp实现，最后将temp的值分配给输出Y。

2. simulator 代码

```
22
23  module sim_test1_2();
24      reg D1,D2,D3,D0;
25      reg [1:0] S;
26      wire Y;
27      test1_2 test1_2(D0,D1,D2,D3,S,Y);
28      initial
29      begin
30         D0=0;D1=0;D2=0;D3=0;S=2'b00;
31          fork
32             repeat(100) #10 D0=~D0;
33             repeat(50) #20 D1=~D1;
34             repeat(25) #40 D2=~D2;
35             repeat(10) #100 D3=~D3;
36             repeat(5) #200 S=S+1;
37          join
38      end
39  endmodule
```

实例化design代码，利用fork并行块，设置4个输入信号（D0、D1、D2、D3）初始为0，分别10，20，40，100ns取反一次,设置S每200ns自加1

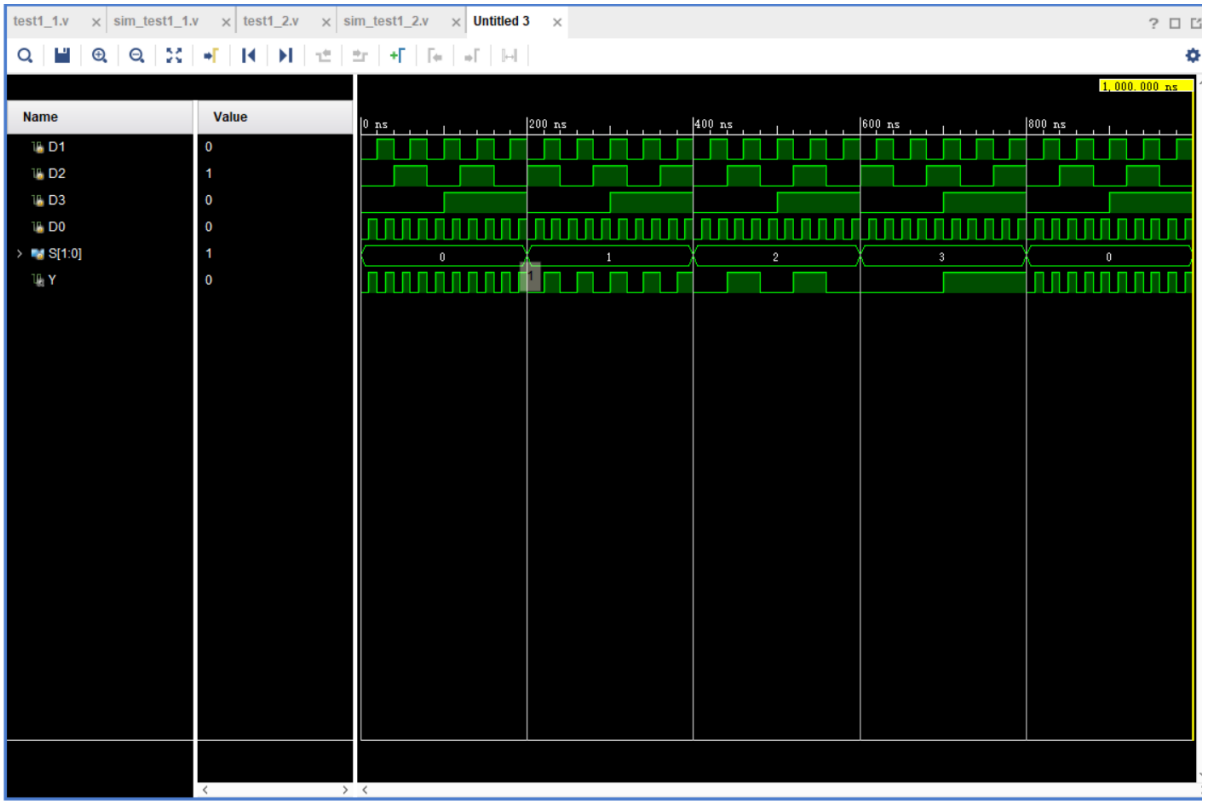
3. RTL 图



仿真电路图为一个多路复用器，输入信号为（D0、D1、D2、D3），S为选择信号，按照下面的代码选择输出

```
case(S)
        2'b00: temp=D0;
        2'b01:temp=D1;
        2'b10:temp=D2;
        2'b11:temp=D3;
```

4. simulation图



当S为 `00,01,10,11` 时，波形分别与 `D0,D1,D2,D3` 相同，与设计相符合，说明设计正确。

3. 优先编码器

1. design 代码

```
module test1_3(I,Y);
    input I;
    output Y;
    wire [7:0] I;
    reg [3:1] Y;
    always @(I) begin
        casex(I)
            8'b0000_0001:Y=3'b000;
            8'b0000_001x:Y=3'b001;
            8'b0000_01xx:Y=3'b010;
            8'b0000_1xxx:Y=3'b011;
            8'b0001_xxxx:Y=3'b100;
            8'b001x_xxxx:Y=3'b101;
            8'b01xx_xxxx:Y=3'b110;
            8'b1xxx_xxxx:Y=3'b111;
            default:Y=3'b000;
        endcase
    end

endmodule
```

设置了一个8位的输入信号 `I` 和一个3位的输出信号 `Y` 。该模块使用 `casex` 语句来匹配输入信号 `I` 的非0的最高位，并根据匹配结果将输出信号 `Y` 设置为对应的值。

```
module test1_3_2(I,Y);
    input I;
    output Y;
    wire [7:0] I;
    reg [3:1] Y;
    always @(I) begin
        if (I[7]==1) Y=3'b111;
        else if (I[6]==1) Y=3'b110;
        else if (I[5]==1) Y=3'b101;
        else if (I[4]==1) Y=3'b100;
        else if (I[3]==1) Y=3'b011;
        else if (I[2]==1) Y=3'b010;
        else if (I[1]==1) Y=3'b001;
        else if (I[0]==1) Y=3'b000;
        else Y=3'b000;
    end
endmodule
```

设置了一个8位的输入信号 `I` 和一个3位的输出信号 `Y` 。该模块使用 `if` 语句，按顺序从高位到低位检测是否为一,一旦检测到一，将输出信号 `Y` 为对应值。

2. simulator 代码

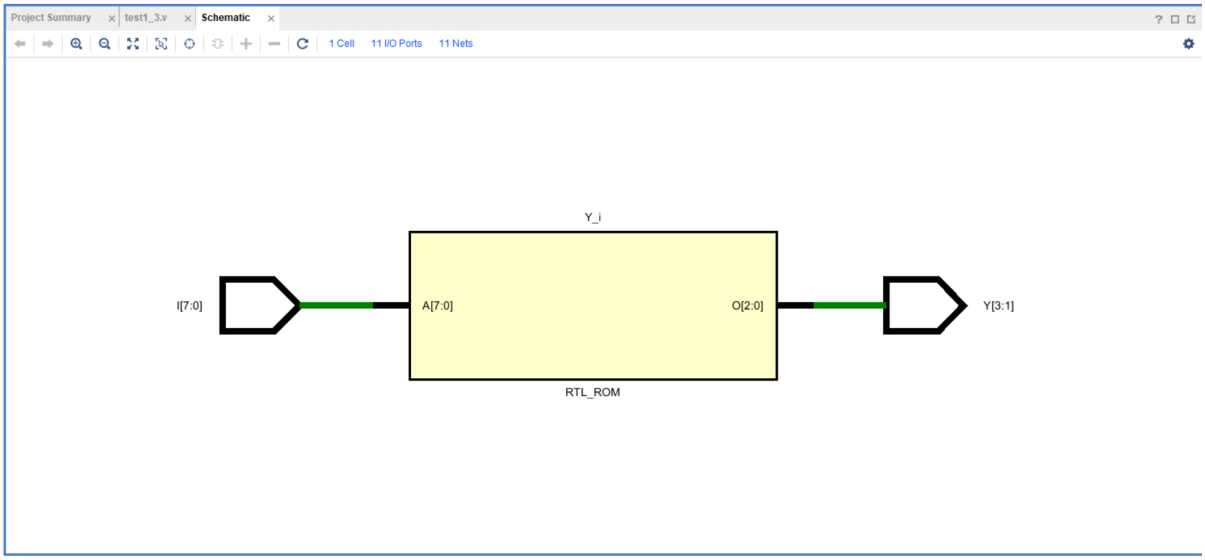
```
module sim_test1_3();
    reg [7:0] x;
    wire [2:0] y_pre_case,y_pre_if;
    integer i;

    initial begin
        x=1;
        for (i=0;i<7;i=i+1) #10 x=x*2;
        #10 x=128;
        while (x>0) #5 x=x-1;
    end
    test1_3 test1_3(x,y_pre_case);
    test1_3_2 test1_3_2(x,y_pre_if);
endmodule
```

设置输入信号从1开始，每10ns `*2` ,到128后，每5秒减1，分别实例化design1(casex方法)和

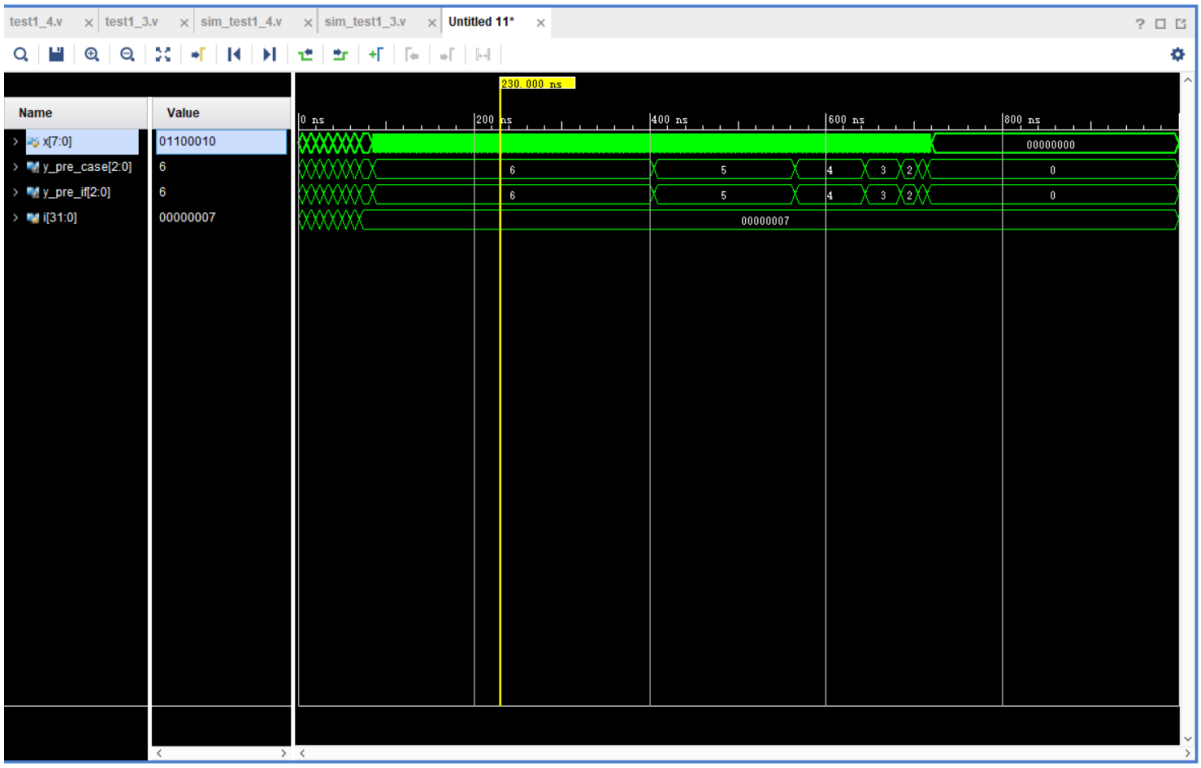
design2(if方法)。

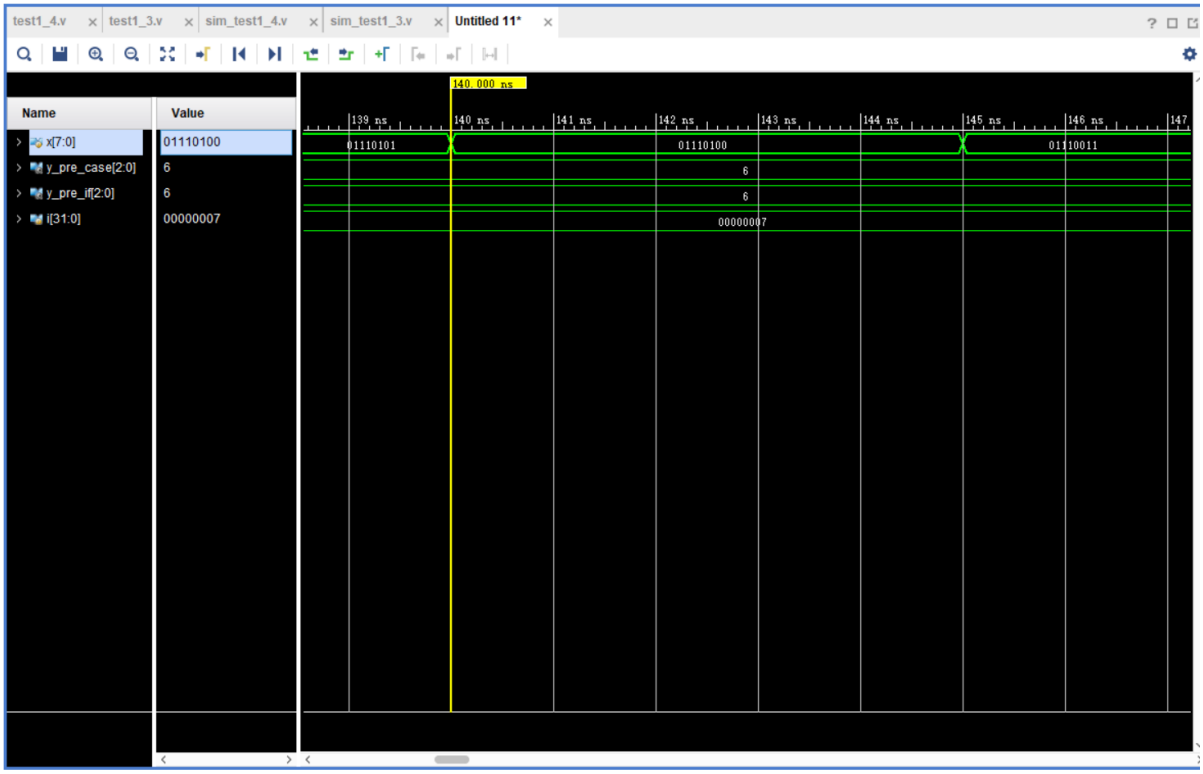
3. RTL 图



一个8位的输入信号 `I` 经过一个编码器，形成一个3位的输出信号 `Y`

4. simulation图





当输入信号非0的最高位为 `i` 时，输出信号 `y_pre_casex` 和 `y_pre_if` 相同且为 `i`，与真值表匹配。

4. 3-8译码器

1. design 代码

```
module test1_4(A,Y);
    input A;
    output Y;
    wire [2:0] A;
    reg [7:0] Y;
    always @(A) begin
        Y[7]=(~A[2])|(~A[1])|(~A[0]);
        Y[6]=(~A[2])|(~A[1])|(A[0]);
        Y[5]=(~A[2])|(A[1])|(~A[0]);
        Y[4]=(~A[2])|(A[1])|(A[0]);
        Y[3]=(A[2])|(~A[1])|(~A[0]);
        Y[2]=(A[2])|(~A[1])|(A[0]);
        Y[1]=(A[2])|(A[1])|(~A[0]);
        Y[0]=(A[2])|(A[1])|(A[0]);
    end
endmodule
```

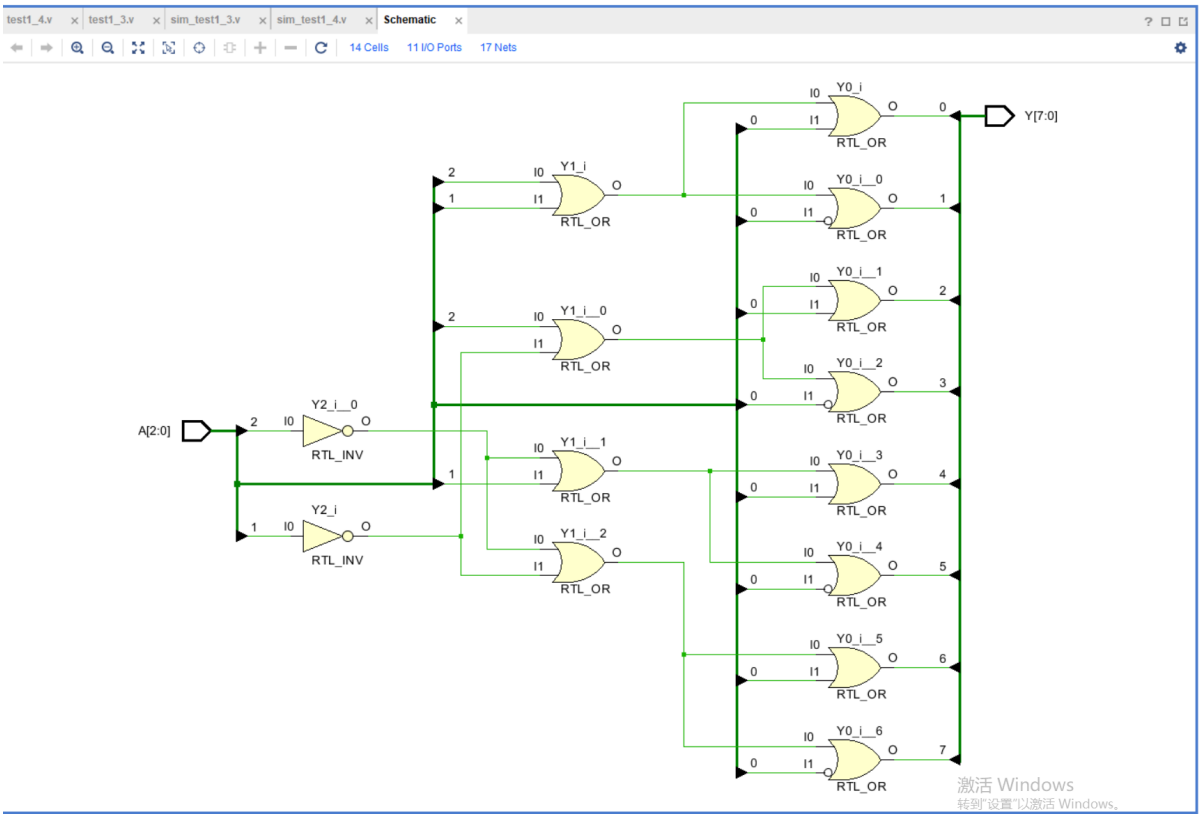
设置3位输入A和8位输入Y，按照逻辑表达式，分别设置Y的每一位

2. simulator 代码

```
module sim_test1_4();
    reg [2:0] x;
    wire [7:0] y;
    integer i;
    initial begin
        x=0;
            for (i=0;i<7;i=i+1) #50 x=x+1;
            #50 x=7;
            while (x>0) #50 x=x-1;
     end
     test1_4 test1_4(x,y);
endmodule
```

设置3位输入信号x，从 `0` 开始每50ns加一，到 `7` 后，每50ns减 `1` 到 `0` ,用于验证从 `000` 到 `111` ，能否正确转化为对应的8位输出。

3. RTL 图



通过表达式可以发现操作主要是与和非，RTL显示的也是用与非门来是实现design功能

4. simulation图

当输入信号从 `000b` 到 `111b`， 对应的十进制为 `0` 到 `7`， 输出信号对应的第i位为 `0` ,其他为都为 `1`，

与真值表匹配。

五 调试和心得体会

1. 相较于上学期的数电实验，这次比较系统的学习了verilog的语法，例如：
    i. `always@(敏感信号)`，敏感信号可为 `*`，会自动匹配所有输入信号
    ii. `fork` ,生成同步块
    iii. `#i` ,暂停 `i` ns,相当于 `sleep(i)`
    iv. `casex(s)` ,可以实现用x作为通配符，提供了更方便的操作
2. 此外，本学期的实验的挑战性增加了，因为有些实验没有提供代码，需要我们自行设计和实现数字电路。这锻炼了我们的创造力和问题解决能力，因为我们需要独立思考如何实现特定的功能，然后将其转化为Verilog代码。这种挑战性的实验让我更深入地理解了计算机组成原理，尤其是如何将理论知识应用于实际项目中。