南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

计算机逻辑基础
实验报告

课程名称：	计算机逻辑基础
姓名 1：	张怀远
学号 1：	920000720124
姓名 2：	李易飞
学号 2：	920000720105
姓名 3：	昌久冬
学号 3：	920000720109
班级：	9200007201
任课老师：	代龙泉 马勇

2022 年 5 月

小组分工：

张怀远：实验仿真和线下操作

李易飞：实验仿真和报告撰写

昌九冬：实验仿真和线下操作

实验一
译码器的设计及应用实验

一、实验目的

学习组合逻辑点路译码器的设计方法及应用。

二、实验内容

1.问题 1 重述：

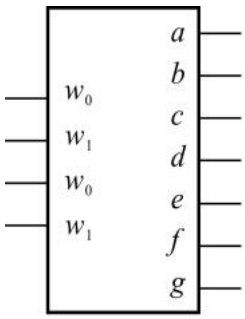
用 verilog 来实现 2-4 译码器或 3-8 译码器或七段译码器；**（完成者：李易飞）**

1.1.功能简述

使用 Vivado 软件以及 verilog 来实现 3-8 译码器。

3-8 译码器是一个具有 3 根输入线，一个使能端和八根输出线的逻辑电路。每种输入对应一种输出。

3-8 译码器的原理如下：



3-8 译码器逻辑符号

1.2.真值表

输入a[2:0]			输出y[7:0]							
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

3-8 译码器真值表

1.3.Verilog 程序

按照真值表，Verilog 语言描述如下;

仿真代码：

```
module exp1_1(
input  wire [2:0]    put_in,
output reg  [7:0] dout
    );
    always@(put_in)begin
        case(put_in)
            3'b000 : dout = 8'b1111_1110;
            3'b001 : dout = 8'b1111_1101;
            3'b010 : dout = 8'b1111_1011;
            3'b011 : dout = 8'b1111_0111;
            3'b100 : dout = 8'b1110_1111;
            3'b101 : dout = 8'b1101_1111;
            3'b110 : dout = 8'b1011_1111;
            3'b111 : dout = 8'b0111_1111;
            default: ;
        endcase
    end
endmodule
```

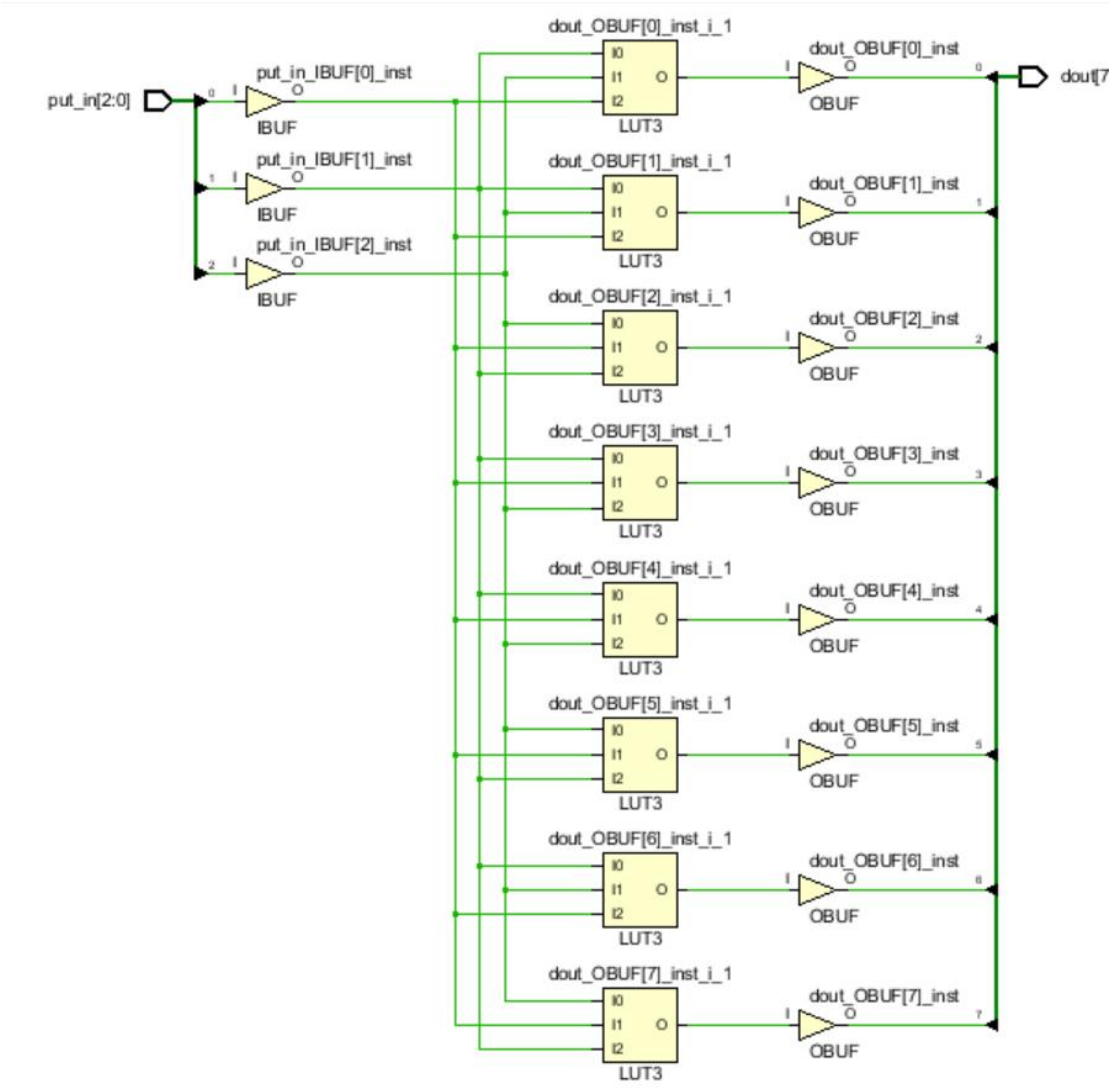
测试代码：

```
module exp1_1_tb(

    );
    reg    [2:0]  put_in;
    wire [7:0]    dout;
```

```
    exp1_1 u0(.put_in(put_in),.dout(dout));
    initial   begin
            put_in = 3'b000;
            #38;
            put_in = 3'b111;
            repeat(15)begin
                # 46;
                put_in = put_in + 1'b1;
            end
            #200;
            $stop(2);
        end
endmodule
```

1.4.电路原理图

RTL 原理图如下：



2.问题 2 重述：

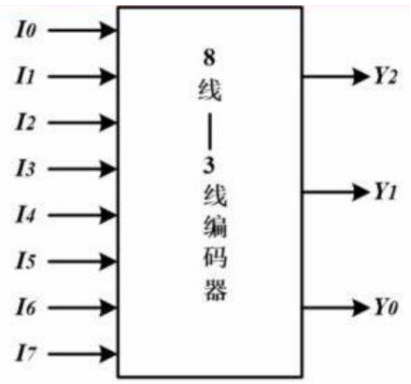
用 verilog 来实现 4-2 编码器或 8-3 编码器；**（完成者：李易飞）**

2.1.功能简述

使用 Vivado 软件以及 verilog 来实现 8-3 编码器。

8-3 编码器是一个它有 8 个信号输入端，3 个二进制码输出端。

8-3 编码器的原理如下：



8-3 编码器逻辑符号

2.2.真值表

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

8-3 编码器真值表

2.3.Verilog 程序

按照真值表，Verilog 语言描述如下：

仿真代码：

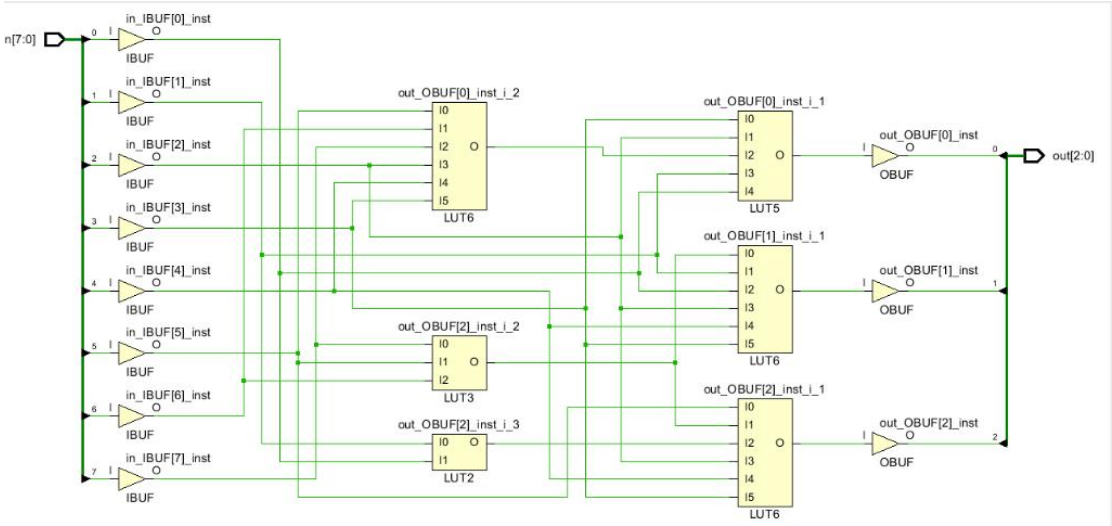
```
module seg7( input [3:0] din, output [7:0] dout );
reg [7:0] dout;
always @ (din)
begin
case(din)
4'b0000 : dout = 8'b00111111;
4'b0001 : dout = 8'b00000110;
```

```
4'b0010 : dout = 8'b01011011;
4'b0011 : dout = 8'b01001111;
4'b0100 : dout = 8'b01100110;
4'b0101 : dout = 8'b01101101;
4'b0110 : dout = 8'b01111101;
4'b0111 : dout = 8'b00000111;
4'b1000 : dout = 8'b01111111;
4'b1001 : dout = 8'b01101111;
default : dout = 8'b00000000;
endcase
end
endmodule
```

测试代码：

```
module encoder8_3(
    input [7:0] sw,
    output reg[2:0] led
);
    always @ (sw)  //当输入 sw 发生改变时，输出 led 也跟着改变
    begin
    case(sw)
    8'b0000_0001: led = 3'b000 ;
    8'b0000_0010: led = 3'b001 ;
    8'b0000_0100: led = 3'b010 ;
    8'b0000_1000: led = 3'b011 ;
    8'b0001_0000: led = 3'b100 ;
    8'b0010_0000: led = 3'b101 ;
    8'b0100_0000: led = 3'b110 ;
    8'b1000_0000: led = 3'b111 ;
    default: led = 3'b000 ;
    endcase
end
endmodule
```

2.4.电路原理图



三、简要操作步骤

1、根据 Vivado 的设计流程，建立本次实验项目的工程文件；

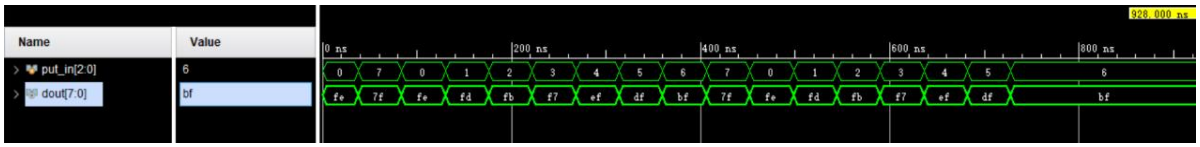
2、建立本次实验内容的 Verilog 功能模块，修改语法错误；

3、建立仿真文件，进行电路功能仿真；

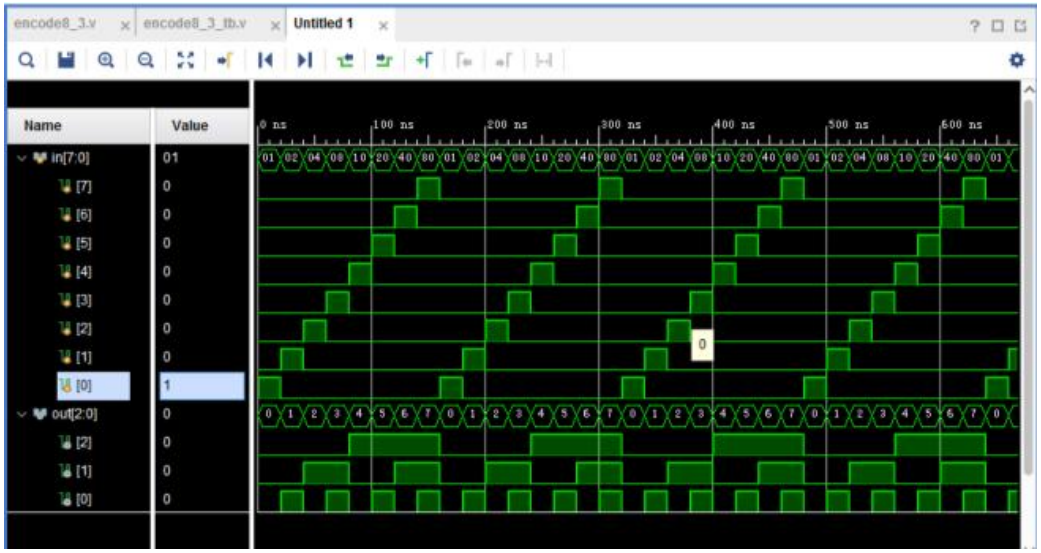
4、查看仿真结果，观察电路功能是否符合预期，不符合需对功能模块和仿真文件进行修改直到电路功能符合预期。

四、实验结果

问题 1 仿真波形如下：



问题 2 仿真波形如下：



五、线下实验现象记录（完成者：张怀远）

问题 2 8-3 编码器实验现象如下：

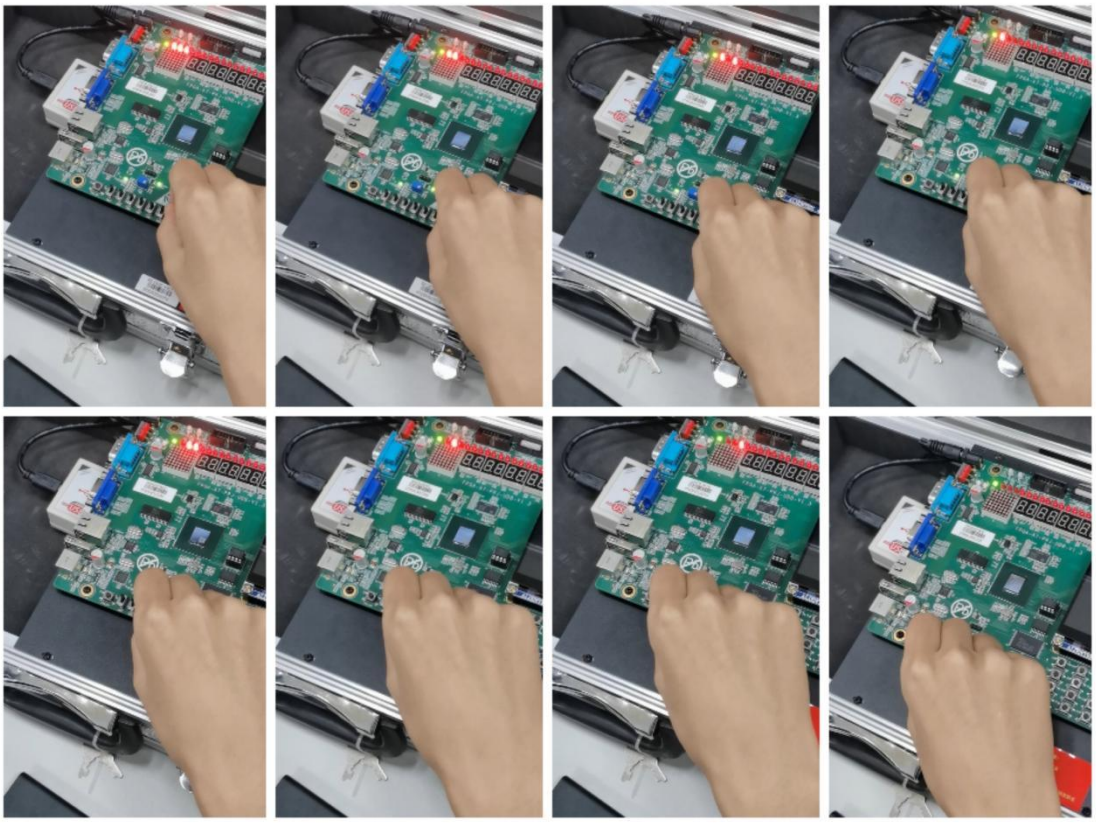
8 输入管脚 3 输出管脚设置情况如下图所示：

Project Summary | exp1_2.xdc

D:/Vivado data/exp1_2/exp1_2.srcs/constrs_1/new/exp1_2.xdc

```
set_property PACKAGE_PIN AC21 [get_ports {in[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in[7]}]
set_property PACKAGE_PIN AD24 [get_ports {in[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in[6]}]
set_property PACKAGE_PIN AC22 [get_ports {in[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in[5]}]
set_property PACKAGE_PIN AC23 [get_ports {in[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in[4]}]
set_property PACKAGE_PIN AB6 [get_ports {in[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in[3]}]
set_property PACKAGE_PIN W6 [get_ports {in[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in[2]}]
set_property PACKAGE_PIN AA7 [get_ports {in[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in[1]}]
set_property PACKAGE_PIN Y6 [get_ports {in[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {in[0]}]
set_property PACKAGE_PIN H7 [get_ports {out[2]}]
set_property PACKAGE_PIN D5 [get_ports {out[1]}]
set_property PACKAGE_PIN A3 [get_ports {out[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[1]}]
```

灯亮为 0，灯灭为 1，下图依次显示 8 个开关得到 0-7 的二进制信号：



实验二

数据通道选择器的设计及应用实验

一、实验目的

学习组合逻辑电路数据选择器的设计方法及应用。

二、实验内容

1.问题重述：

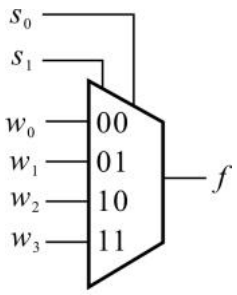
用 verilog 来实现 2 选 1 或 4 选 1 或 8 选 1 等数据选择器。**（完成者：李易飞）**

1.1.功能简述

使用 Vivado 软件以及 verilog 来实现 2 选 1 数据选择器。

2 选 1 数据选择器是一个有 2 路数据传送，能够根据需要将其中任意一路选出来送到输出的的电路。

2 选 1 数据选择器的原理如下：



(a) 逻辑符号

2 选 1 数据选择器逻辑符号

1.2.真值表

s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

(b) 真值表

2 选 1 数据选择器真值表

1.3.Verilog 程序

按照真值表，Verilog 语言描述如下;

仿真代码：

```
module data_select2_1(
input a, input b, input sel, output reg dout
    );
    always @ (a or b or sel)
```

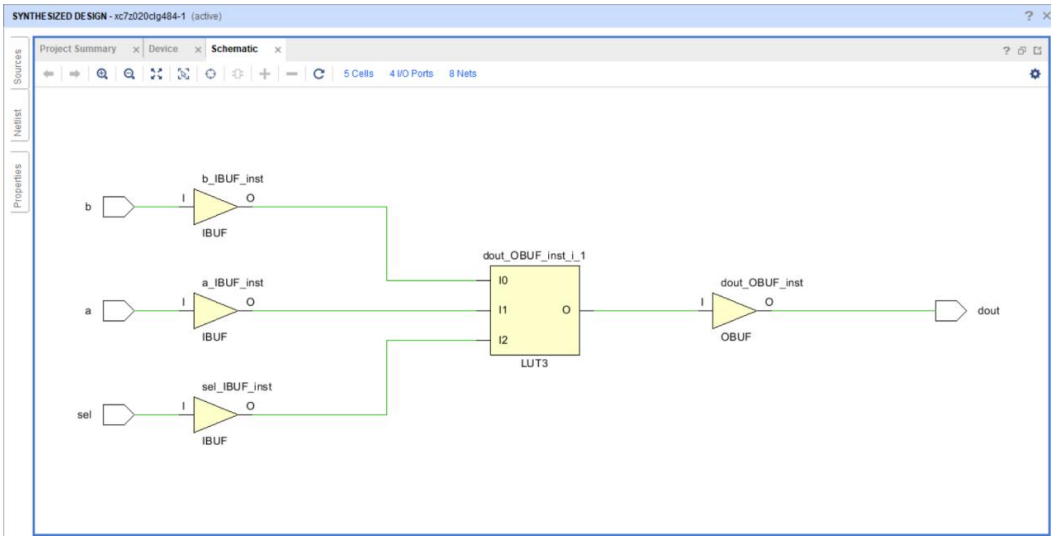
```
     begin
     case (sel)
     1'b0 : dout <= a;
     1'b1 : dout <= b;
     default : dout = 1'bx;
     endcase
     end
endmodule
```

测试代码：

```
module data_select2_1_tb(
    );
    reg a; reg b; reg sel;
     wire dout;
     data_select2_1 u0(.a(a), .b(b), .sel(sel), .dout(dout));
     initial begin //初始化输入变量
     a = 1'b0;
     b = 1'b0;
     sel = 1'b0; #10;
     end
     always @(a or b or sel) begin
     a = 1'b0; b = 1'b0; sel = 1'b0; #10; //每 10 个时间间隔变换输入变
量
     a = 1'b0; b = 1'b1; sel = 1'b0; #10;
     a = 1'b1; b = 1'b0; sel = 1'b0; #10;
     a = 1'b1; b = 1'b1; sel = 1'b0; #10;
     a = 1'b0; b = 1'b0; sel = 1'b1; #10;
     a = 1'b0; b = 1'b1; sel = 1'b1; #10;
     a = 1'b1; b = 1'b0; sel = 1'b1; #10;
     a = 1'b1; b = 1'b1; sel = 1'b1; #10;
     end
endmodule
```

1.4.电路原理图

RTL 原理图如下：



三、简要操作步骤

1、根据 Vivado 的设计流程，建立本次实验项目的工程文件；

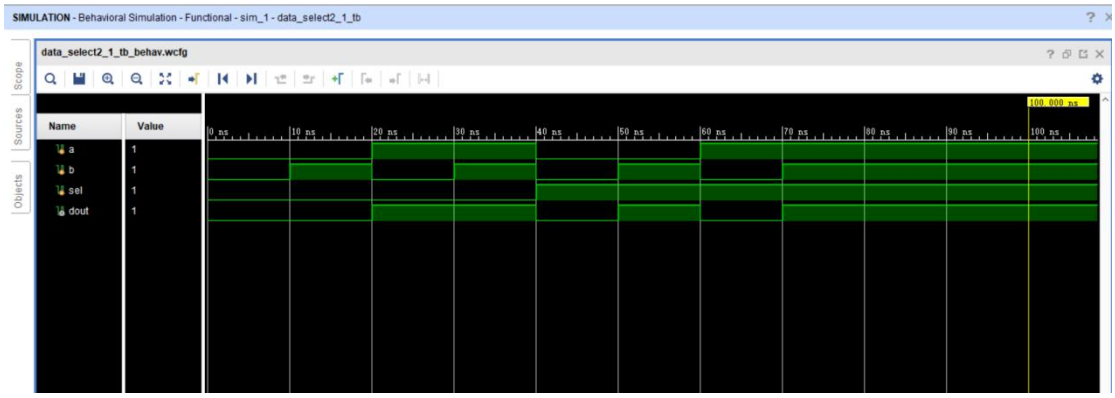
2、建立本次实验内容的 Verilog 功能模块，修改语法错误；

3、建立仿真文件，进行电路功能仿真；

4、查看仿真结果，观察电路功能是否符合预期，不符合需对功能模块和仿真文件进行修改直到电路功能符合预期。

四、实验结果

仿真波形如下：



五、线下实验现象记录（完成者：昌久冬）

2 选 1 数据选择器实验现象记录：

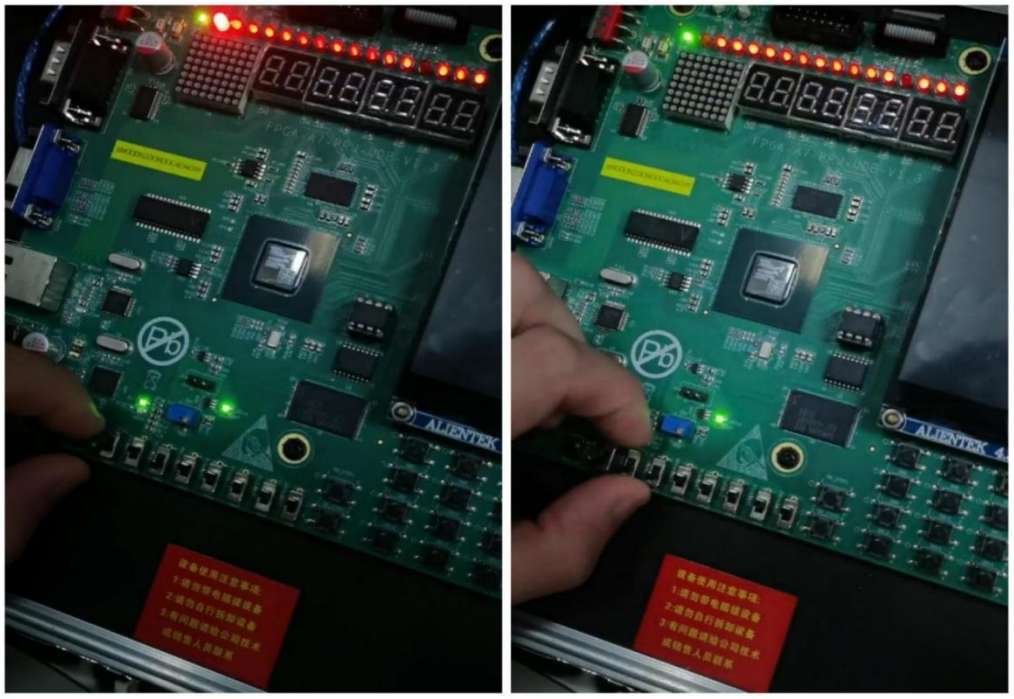
输入输出管脚接法如图所示：

```
set_property PACKAGE_PIN H7 [get_ports dout]
set_property PACKAGE_PIN AC21 [get_ports a]
set_property PACKAGE_PIN AD24 [get_ports b]
set_property PACKAGE_PIN AC22 [get_ports sel]
set_property IOSTANDARD LVCMOS33 [get_ports dout]
set_property IOSTANDARD LVCMOS33 [get_ports a]
set_property IOSTANDARD LVCMOS33 [get_ports b]
```

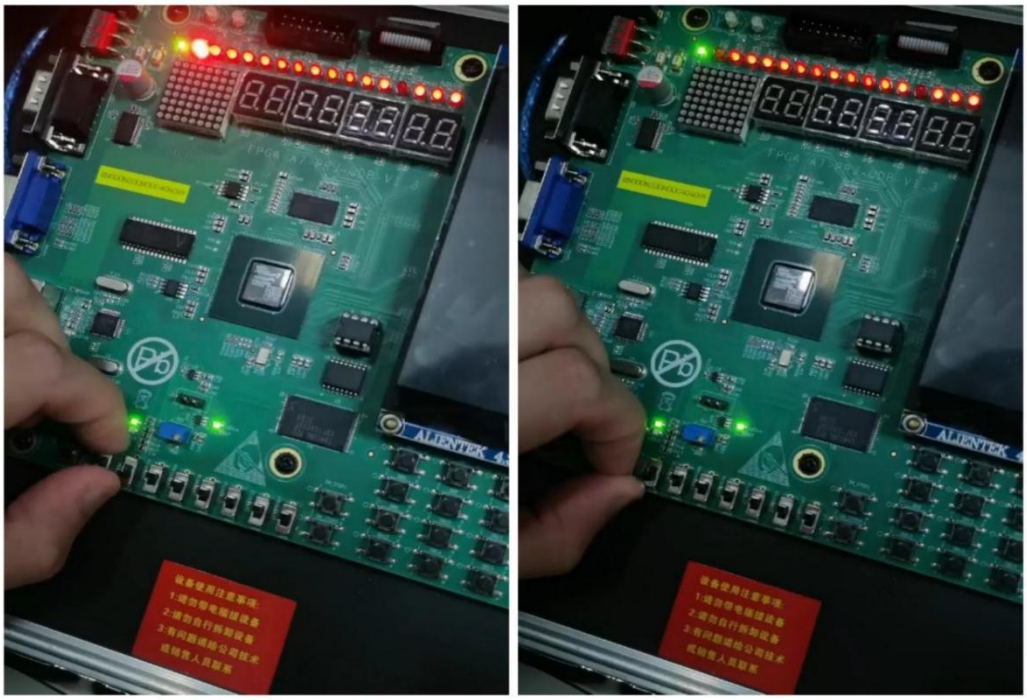
set_property IOSTANDARD LVCMOS33 [get_ports sel]

实验现象记录如下图：

select 管脚（从左向右数第三个）置 0 时选择 a（从左向右数第二个），输出状态由 a 决定，b 的改变不影响输出：



select 管脚置 1 时选择 b（从左向右数第一个），输出状态由 b 决定，a 的改变不影响输出：



实验三

计数器的设计及应用实验

一、实验目的

学习时序逻辑电路计数器的设计方法及应用。

二、实验内容

1.问题重述：

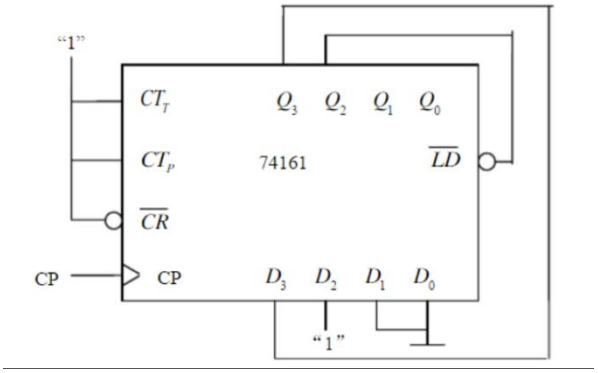
用 verilog 生成一个四位二进制计数器，同步清零；（**完成者：张怀远，李易飞**）

1.1.功能简述

使用 Vivado 软件以及 verilog 来实现四位二进制计数器。

四位二进制计数器是一个有 2 路数据传送，能够根据需要将其中任意一路选出来送到输出的的电路。

四位二进制计数器的原理如下：



四位二进制计数器逻辑符号

1.2.真值表

CLK	CLR	LD	EN	功能
↑	0	X	X	同步清零
↑	1	0	X	同步置数
X	1	1	0	保持
↑	1	1	1	加计数

四位二进制计数器真值表

1.3.Verilog 程序

按照真值表，Verilog 语言描述如下;

仿真代码：

```
module exp3_1(
    input clk,
```

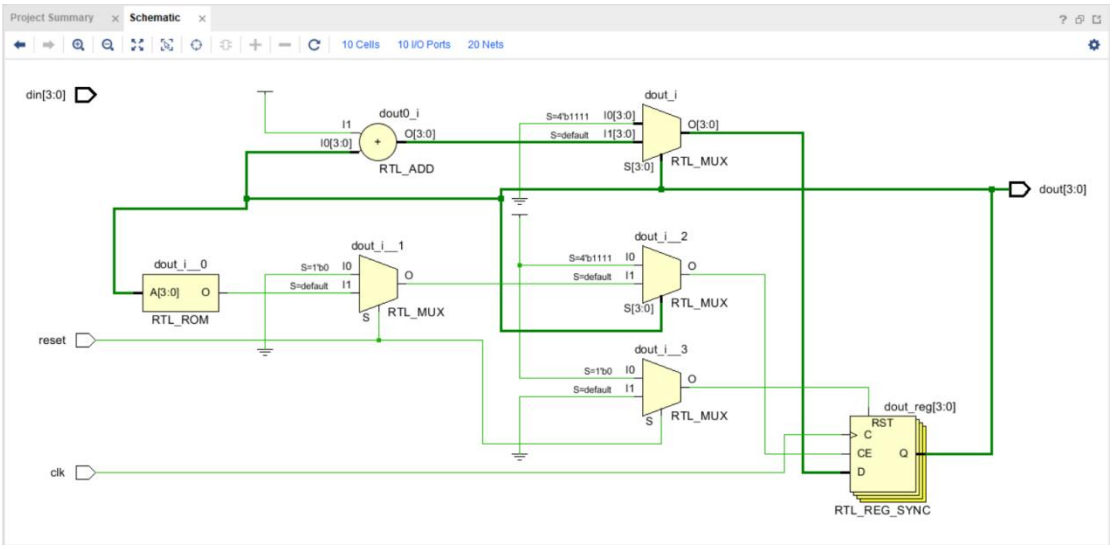
```
    input [3:0]din,
    input reset,
    output [3:0]dout
    );
    reg [3:0] dout;
    parameter max=4'b1111;
always @(posedge clk)
    begin
    if(!reset)
    dout<=4'b0000;
    else if(dout==max)
        dout<=4'b0000;
    else
        dout=dout+1;
    end
endmodule
```

测试代码：

```
module exp3_1_tb(

    );
    reg [3:0] din;
    reg clk,reset;
    wire [3:0] dout;
    exp3_1 u0(.clk(clk),.din(din),.reset(reset),.dout(dout));
    initial
        begin
        clk=0;
        #4 reset=1;
        #10 reset=0;
        #50 reset=1;
        din=4'b0000;
        end

    always begin
        #10 clk=~clk;
    end
endmodule
```

1.4.电路原理图

RTL 原理图如下：



2.问题重述：

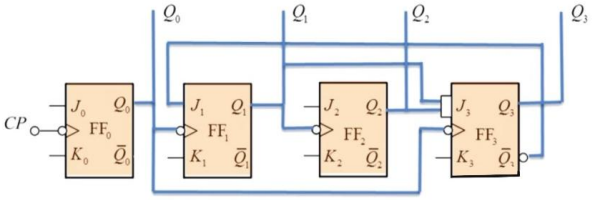
用 verilog 生成一个十进制计数器，异步清零；（**完成者：张怀远**）

2.1.功能简述

使用 Vivado 软件以及 verilog 来实现十进制计数器。

十进制计数器是指用二进制编码来表示十进制数字来实现计数过程的计数器。

十进制计数器的原理如下：



2.2.真值表

计数脉冲数	Q_3	Q_2	Q_1	Q_0	$J_0=1$	$K_0=1$	$J_1=\overline{Q}_3$	$K_1=1$	$J_2=1$	$K_2=1$	$J_3=Q_1Q_2$	$K_3=1$
0	0	0	0	0	1	1	1	1	1	1	0	1
1	0	0	0	1	1	1	1	1	1	1	0	1
2	0	0	1	0	1	1	1	1	1	1	0	1
3	0	0	1	1	1	1	1	1	1	1	0	1
4	0	1	0	0	1	1	1	1	1	1	0	1
5	0	1	0	1	1	1	1	1	1	1	0	1
6	0	1	1	0	1	1	1	1	1	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	1
8	1	0	0	0	1	1	0	1	1	1	0	1
9	1	0	0	1	1	1	0	1	1	1	0	1
10	0	0	0	0	1	1	1	1	1	1	0	1

十进制计数器真值表

2.3.Verilog 程序

按照真值表，Verilog 语言描述如下;

仿真代码：

```
module exp3_2(
    input clk,
```

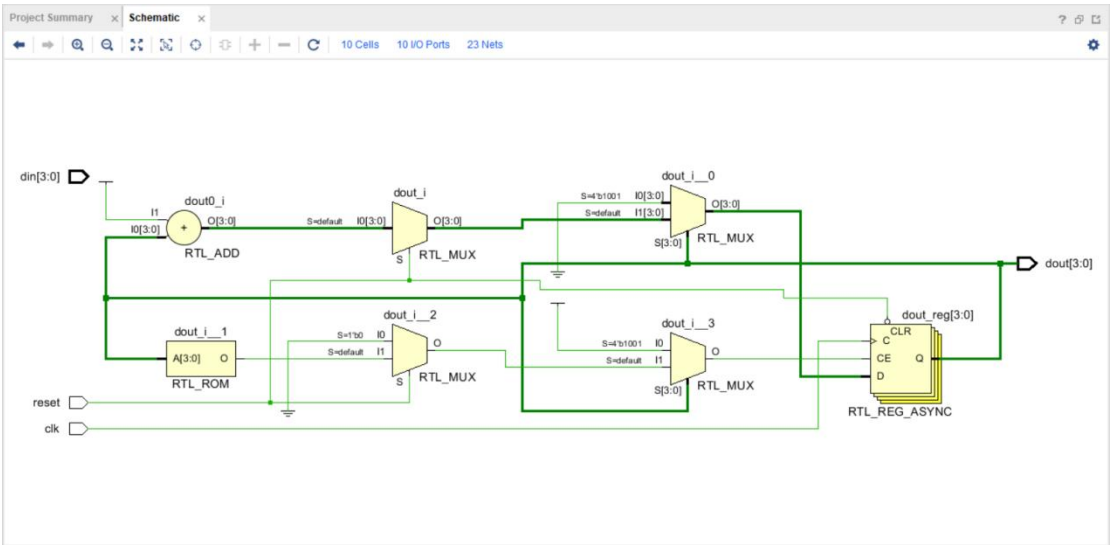
```
    input [3:0]din,
    input reset,
    output [3:0]dout
    );
    reg [3:0] dout;
    parameter max=4'b1001;
always @(posedge clk or negedge reset)
    begin
    if(!reset)
    dout<=4'b0000;
    else if(dout==max)
        dout<=4'b0000;
    else
        dout=dout+1;
    end
endmodule
```

测试代码：

```
module exp3_2_tb(
);
    reg [3:0] din;
    reg clk,reset;
    wire [3:0] dout;
    exp3_2 u0(.clk(clk),.din(din),.reset(reset),.dout(dout));
    initial
        begin
        clk=0;
        reset=1;
        #10 reset=0;
        #50 reset=1;
        #155 reset=0;
        #50 reset=1;
        din=4'b0000;
        end

    always begin
        #10 clk=~clk;
    end
endmodule
```

2.4.电路原理图

RTL 原理图如下：



3.问题重述：

用 verilog 生成一个异步可逆十进制计数器，具有同步置数功能；**（完成者：张怀远）**

3.1.功能简述

使用 Vivado 软件以及 verilog 来实现异步可逆十进制计数器。

异步可逆十进制计数器就是把加法计数器和减法计数器的作用合在一起，在逻辑线路上，对计数器的进位和借位脉冲进行适当的控制。即用一个与或门把进位和借位脉冲加以控制，便构成可逆计数器。

3.2.真值表

CLK	CLR	LD	EN	功能
X	0	X	X	异步清零
↑	1	0	X	同步置数
↑	1	1	0	减计数
↑	1	1	1	加计数

异步可逆十进制计数器真值表

3.3.Verilog 程序

按照真值表，Verilog 语言描述如下；

仿真代码：

```
module exp3_3(
    input clk,
    input [3:0]din,
    input reset,ld,en,
    output reg [3:0]dout
    );
    parameter max=4'b1001;
```

```
always @(posedge clk or negedge reset)
    begin
    if(!reset)
    dout<=4'b0000;
    else if(!ld)
    dout=din;
    else if(en)
        if(dout==max)
            dout<=4'b0000;
        else
            dout=dout+1;
    else
        if(dout==0)
            dout<=4'b1001;
        else
            dout=dout-1;
    end
endmodule
```

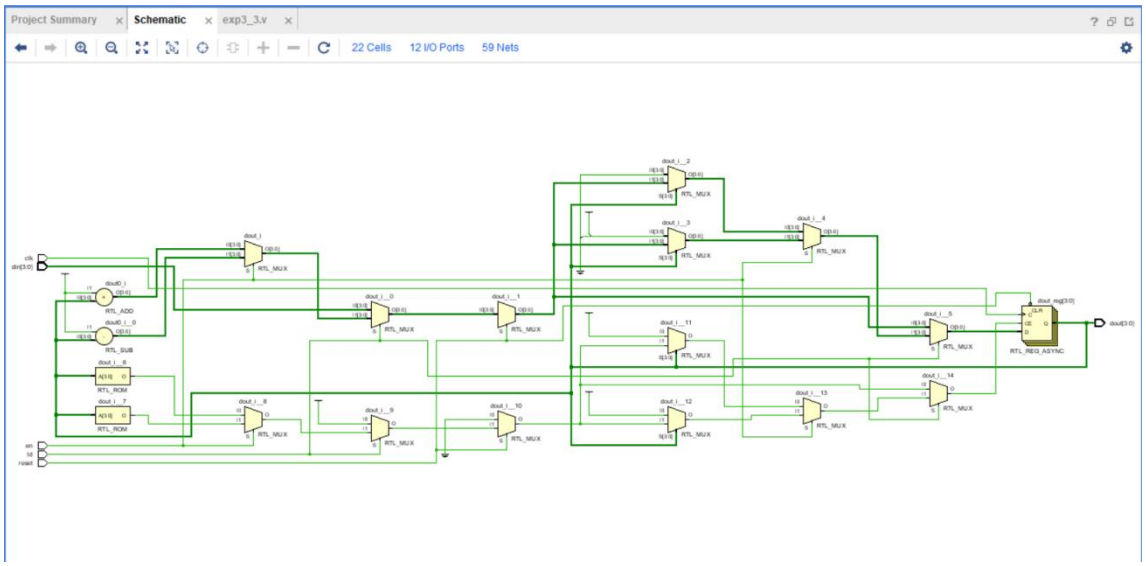
测试代码：

```
module exp3_3_tb(

    );
    reg [3:0] din;
        reg clk,ld,en,reset;
        wire [3:0] dout;
        exp3_3
u0(.clk(clk),.din(din),.reset(reset),.ld(ld),.en(en),.dout(dout));
        initial
            begin
            din=4'b0000;
            clk=0;
            ld=1;
            en=1;
            reset=1;
            #5 reset=0;
            #5 reset=1;
            #150 reset=0;
            #15 reset=1;
            #50 en=0;#200;en=1;
            #10 ld=0;#21;ld=1;
            end

        always begin
            #10 clk=~clk;
```

```
        end
endmodule
```

3.4.电路原理图

RTL 原理图如下：



4.问题重述：

用 verilog 生成一个模 100 的计数器，具有清零、预置功能；**（完成者：昌久冬）**

4.1.功能简述

使用 Vivado 软件以及 verilog 来实现模 100 的计数器。

模 100 的计数器具有记忆 100 个脉冲数的能力，计数器能表示 100 种状态。

4.2.Verilog 程序

Verilog 语言描述如下;

仿真代码：

```
module exp3_4(
    input clk,
    input [6:0]din,
    input reset,ld,en,
    output reg [6:0]dout
    );
    parameter max=7'b1100011;
always @(posedge clk or negedge reset)
    begin
    if(!reset)
    dout<=7'b0000000;
    else if(!ld)
    dout=din;
    else if(en)
        if(dout==max)
            dout<=7'b0000000;
        else
```

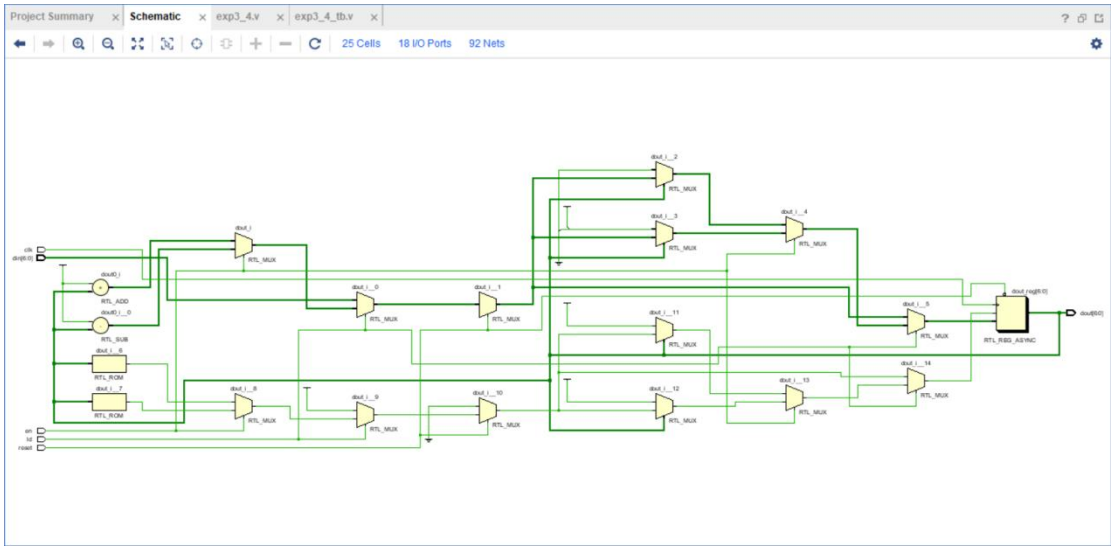
```
            dout=dout+1;
    else
        if(dout==0)
            dout<=7'b1100011;
        else
            dout=dout-1;
    end
endmodule
```

测试代码：

```
module exp3_4_tb(

    );
    reg [6:0] din;
        reg clk,ld,en,reset;
        wire [6:0] dout;
        exp3_4
u0(.clk(clk),.din(din),.reset(reset),.ld(ld),.en(en),.dout(dout));
        initial
            begin
            din=7'b0000000;
            clk=0;
            ld=1;
            en=1;
            reset=1;
            #5 reset=0;
            #5 reset=1;
            #150 reset=0;
            #15 reset=1;
            #50 en=0;#200;en=1;
            #10 ld=0;#21;ld=1;
            end

        always begin
            #10 clk=~clk;
        end
endmodule
```

4.3 电路原理图

RTL 原理图如下：



5. 问题重述：

用 Verilog 生成一个左移移位寄存器。**（完成者：张怀远）**

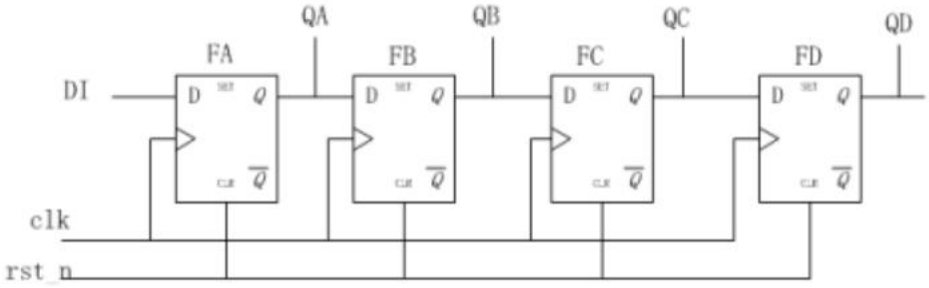
5.1. 功能简述

使用 Vivado 软件以及 verilog 来实现移位寄存器。

移位寄存器内的数据可以在移位脉冲(时钟信号)的作用下依次左移或右移。移位寄存器不仅可以存储数据，还可以用来实现数据的串并转换、分频，构成序列码发生器、序列码检测器，进行数值运算以及数据处理等。

右移位寄存器的右边寄存器的次态等于左边触发器的现态。串行输出数据从触发器 FD 的 QD 端输出，并行数据从个触发器的 QA~QD 端输出，两种输出方式都属于同向输出。各触发器都采用同一时钟信号，所以它们工作在同步状态。如果将 FD 的输出端 QD 接到 FA 的输入端 DI，则可以构成循环移位的右移位寄存器。

移位寄存器的原理如下：



5.2 Verilog 程序

按照真值表，Verilog 语言描述如下;

仿真代码：

```
module exp3_5(
  input  clk ,
  input  reset ,
  input  data_i ,
  output reg [3:0] result_o
    );
    reg [2:0] cnt;
    reg QA,QB,QC,QD;
```

```
    always @ (posedge clk or negedge reset)begin
      if(!reset) begin
        cnt <= 2'b0;
        QA <= 1'b0;
        QB <= 1'b0;
        QC <= 1'b0;
        QD <= 1'b0;
        result_o <= 4'b0;
      end
      else begin
        QA <= data_i;
        QB <= QA;
        QC <= QB;
        QD <= QC;
        result_o <= {QD,QC,QB,QA};
      end
    end

    endmodule
```

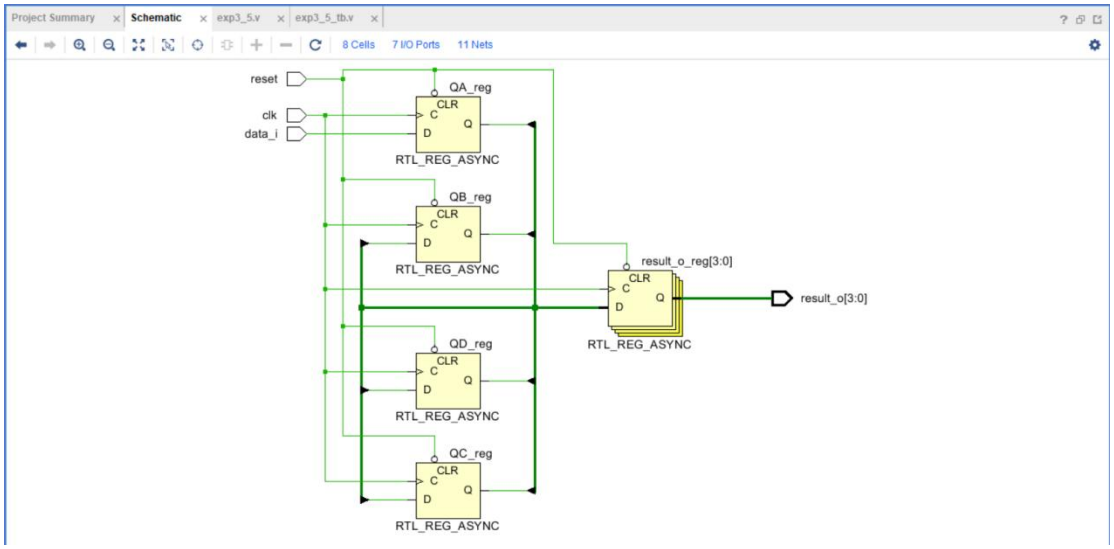
测试代码：

```
module exp3_5_tb(

    );
      reg clk;
      reg reset;
      reg data_i;
      wire [3:0] result_o;

    initial begin
        clk = 0;
        reset = 0;
        data_i = 0;
        #10 reset = 0;
        #10;
        reset = 1;
      end

      always #50  clk = ~clk;
      always begin
      #100 data_i = 1;
      #100 data_i = 1;
      #100 data_i = 0;
      #100 data_i = 1;
```

```
    end
exp3_5  exp3_5_tb(
                .clk(clk),
                .reset(reset),
                .data_i(data_i),
                .result_o(result_o)
              );
endmodule
```

5.3 电路原理图

RTL 原理图如下：



6. 问题重述：

用 Verilog 实现分频器（5 倍分频器）。**（完成者：昌久冬）**

6.1. 功能简述

使用 Vivado 软件以及 verilog 来实现分频器。

分频器通常是由计时器来实现的，将高频脉冲信号作为时钟信号送入计数器的时钟输入端进行计数，在计数器的输出端就可以得到频率低的信号。

6.2 Verilog 程序

按照真值表，Verilog 语言描述如下;

仿真代码：

```
module exp3_6(
    input clk,
    input rst_n,
    output reg clk_out
    );
    reg [3:0] cnt;
    always @(posedge clk or negedge rst_n)
    begin
        if(!rst_n)
        begin
            cnt <= 4'b0;
            clk_out <= 1'b0;
```

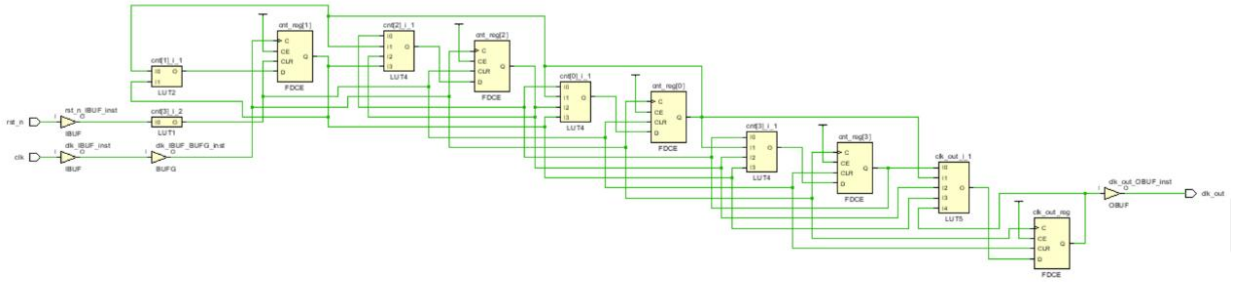
```
            end
            else
            begin
                if(cnt == 3'd4)
                begin
                    clk_out <= ~clk_out;
                    cnt <= 4'b0;
                end
                else
                begin
                    cnt <= cnt + 4'b1;
                end
            end
        end
endmodule
```

测试代码：

```
module exp3_6_tb(

    );
    reg clk,clr;
    wire clk_out;
    exp3_6 uo(.clk(clk),.rst_n(clr),.clk_out(clk_out));
    initial begin
        clk=0;
        clr=0;
     end
        always begin
        #10 clk=~clk;
        clr=1;
        end

endmodule
```

6.3 电路原理图

RTL 原理图如下：



三、简要操作步骤

1、根据 Vivado 的设计流程，建立本次实验项目的工程文件；

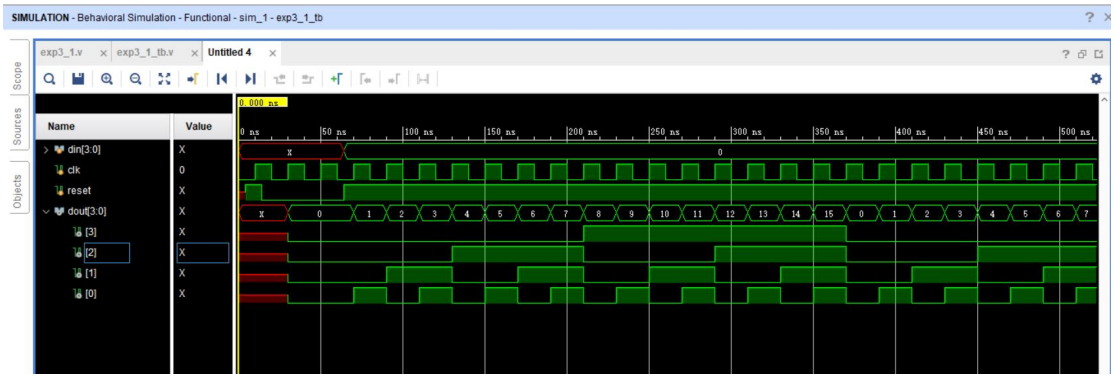
2、建立本次实验内容的 Verilog 功能模块，修改语法错误；

3、建立仿真文件，进行电路功能仿真；

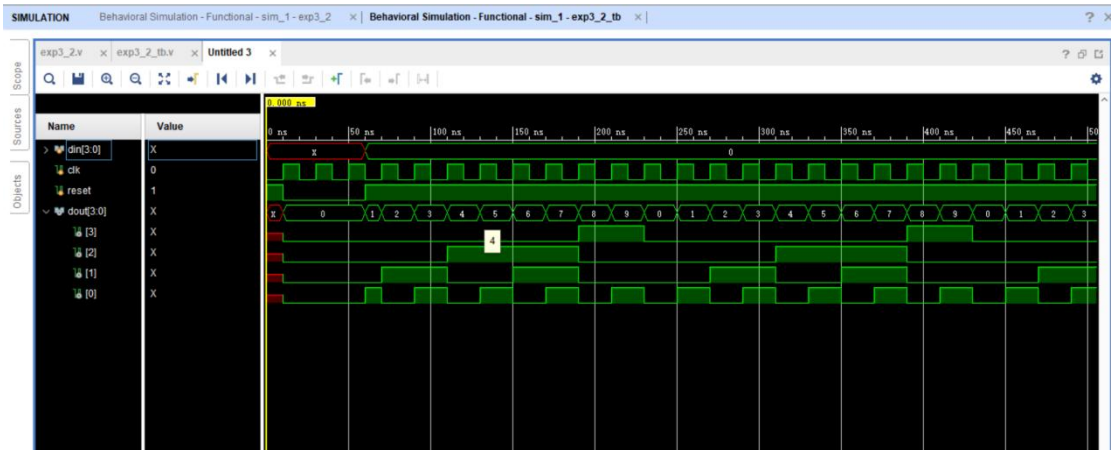
4、查看仿真结果，观察电路功能是否符合预期，不符合需对功能模块和仿真文件进行修改直到电路功能符合预期。

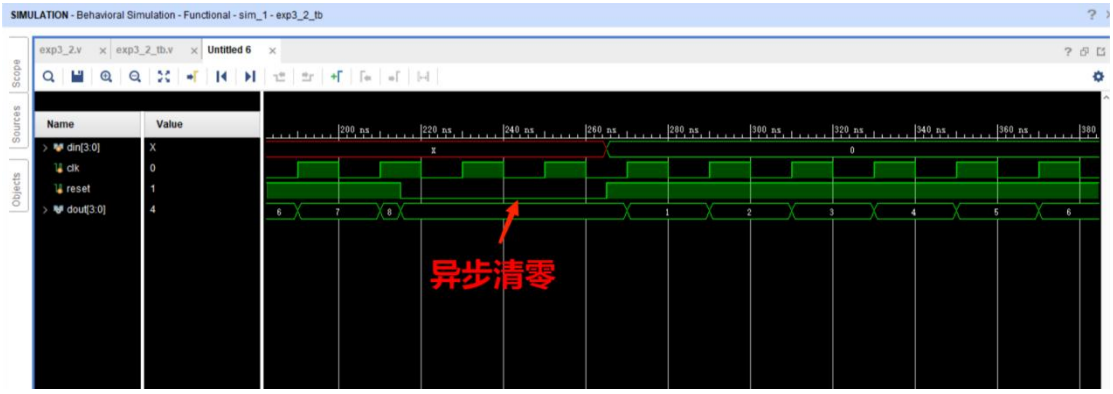
四、实验结果

问题 1 仿真波形如下：



问题 2 仿真波形如下：

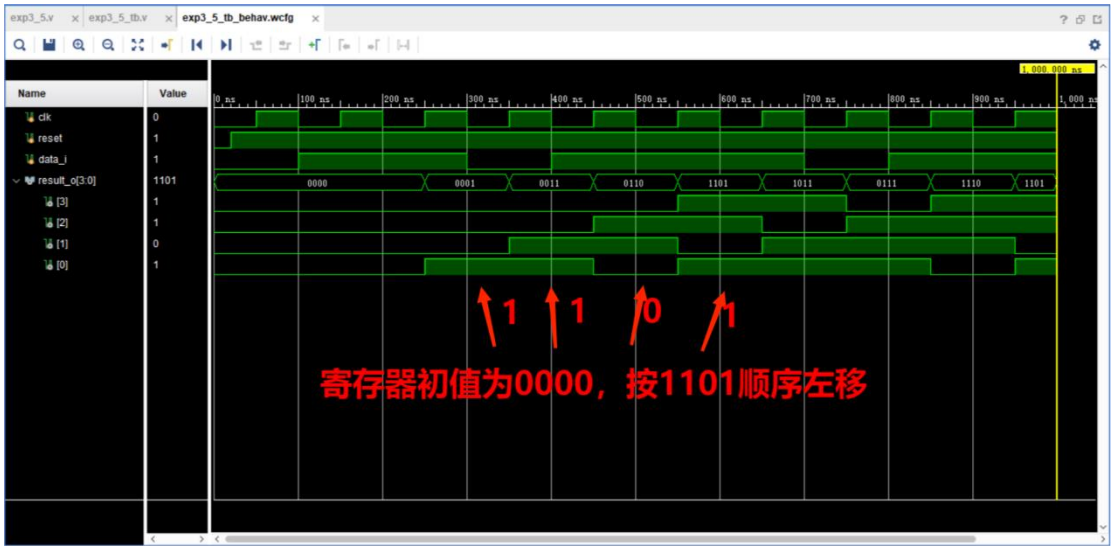




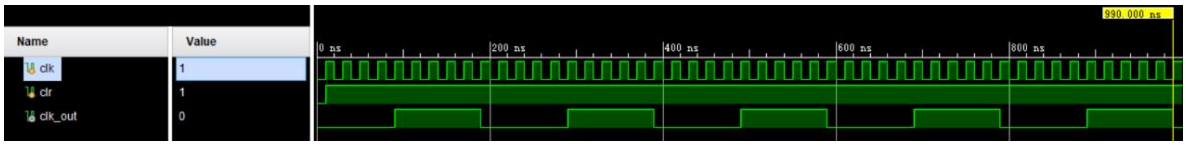
问题 3 仿真波形如下：

问题 4 仿真波形如下：

问题 5 仿真波形如下：



问题 6 仿真波形如下：



五、线下实验现象记录（完成者：张怀远）

问题 1：4 位二进制计数器的实验现象如下：

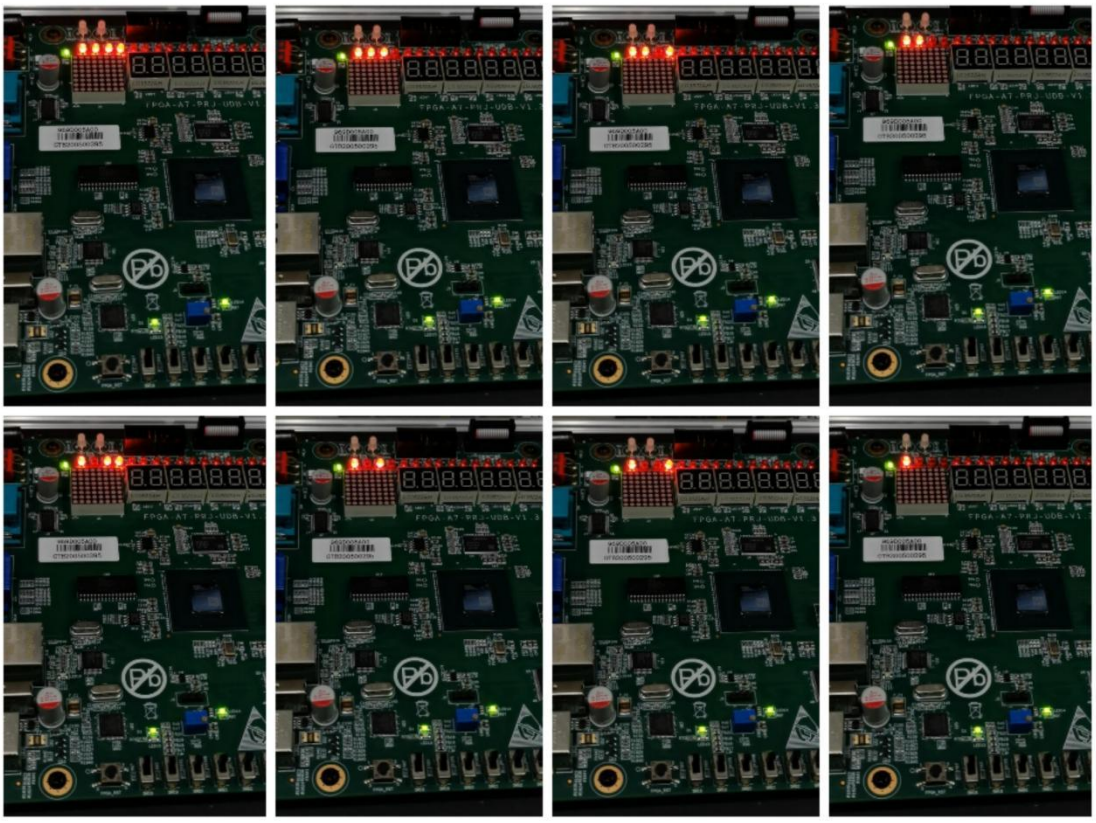
输入输出管脚图接法如图所示：

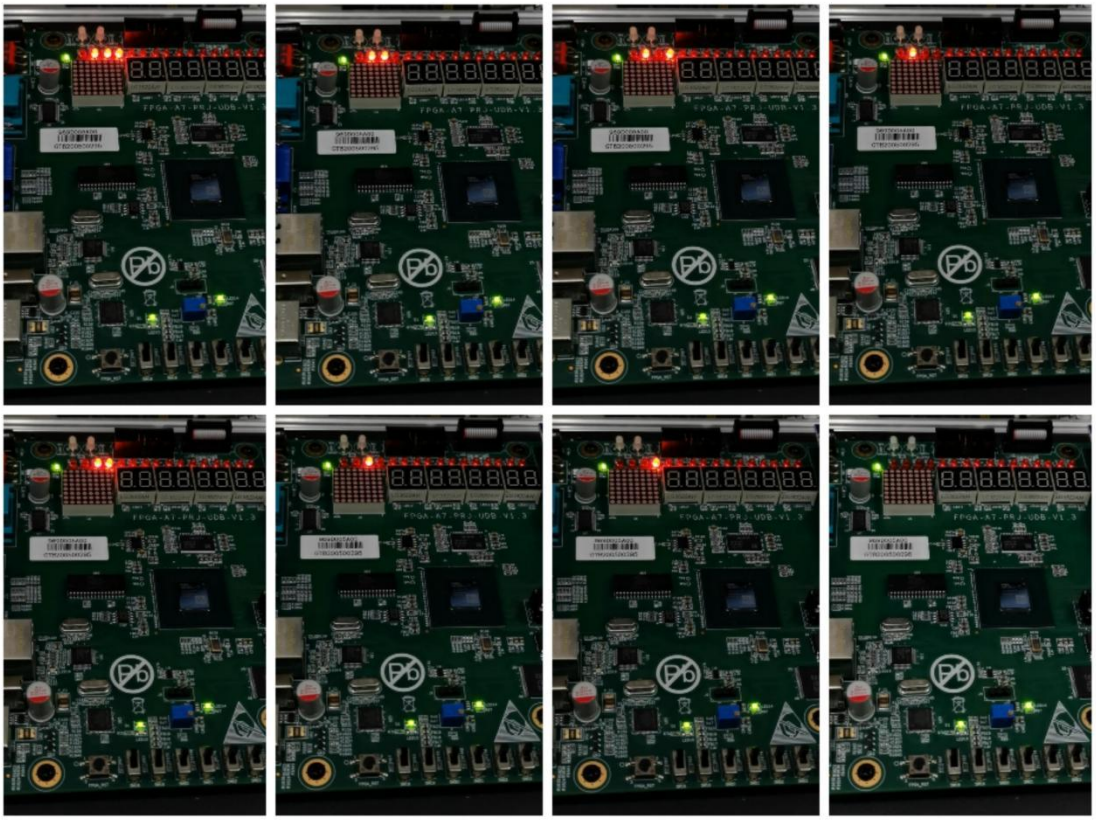
Project Summary | Schematic | exp3_1.xdc

D:/Vivado data/exp3_1/exp3_1.srcs/constrs_1/new/exp3_1.xdc

```
set_property IOSTANDARD LVCMOS33 [get_ports {din[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {din[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {din[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {din[0]}]
set_property PACKAGE_PIN AC21 [get_ports {din[3]}]
set_property PACKAGE_PIN AD24 [get_ports {din[2]}]
set_property PACKAGE_PIN AC22 [get_ports {din[1]}]
set_property PACKAGE_PIN AC23 [get_ports {din[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[0]}]
set_property PACKAGE_PIN D5 [get_ports {dout[2]}]
set_property PACKAGE_PIN H7 [get_ports {dout[3]}]
set_property PACKAGE_PIN A3 [get_ports {dout[1]}]
set_property PACKAGE_PIN A5 [get_ports {dout[0]}]
set_property PACKAGE_PIN AC19 [get_ports clk]
set_property PACKAGE_PIN Y3 [get_ports reset]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports reset]

```

灯亮表示 0，灯灭表示 1，由于 4 位二进制计数器模 16 进位，由此得到的 0-15 的计数如图所示：





实验四
状态机的设计及应用实验

一、实验目的

学习时序逻辑电路中 Mealy 和 Moore 状态机的设计方法及应用。

二、实验内容

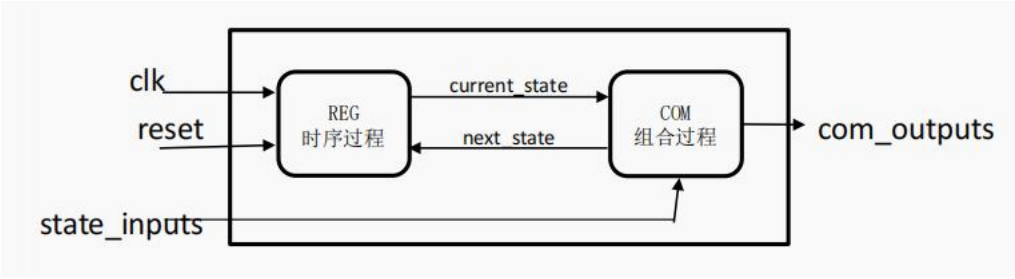
1.问题重述：

设计一个序列检测状态机，实现 Mealy 型或 Moore 型。**（完成者：昌久冬）**

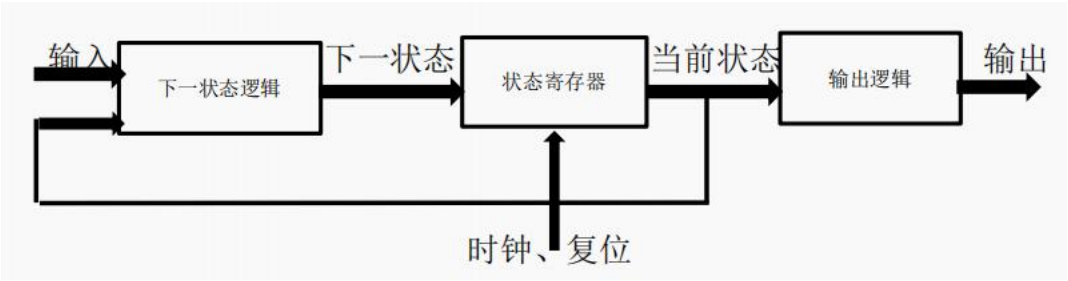
1.1.功能简述

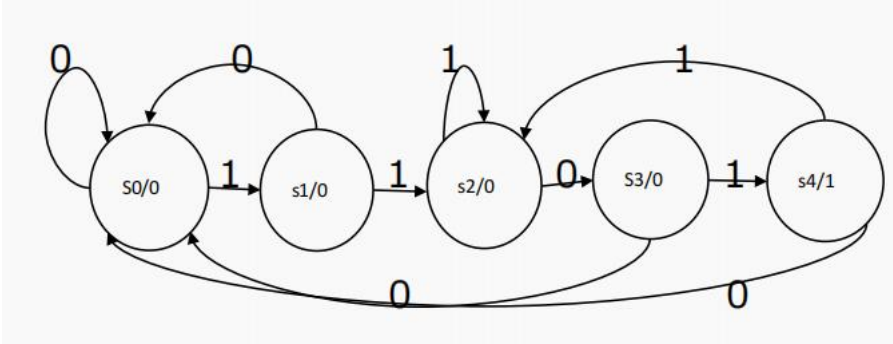
使用 Vivado 软件以及 verilog 来实现 Moore 型序列检测状态机。

有限自动状态机 FSM（Finate State Machine)是复杂数字系统设计中非常重要的一部分，是实现高效率高可靠性逻辑控制的重要途径。大部分数字系统都是由控制单元和数据单元 组成的。数据单元负责数据的处理和传输，而控制单元主要是控制数据单元的操作顺序。在数字系统中，控制单元往往通过使用有限状态机实现，有限状态机接受外部信号以及数 据单元产生的状态信号，从而产生控制信号序列。有限状态机的一般结构如下图：



Moore 型状态机的输出仅与状态机的状态有关，与状态机的输入无关。
Moore 型序列检测状态机的原理如下：





1.2.Verilog 程序

按照状态转换图，Verilog 语言描述如下；

```
module moore_1(
    input clk,
    input clr,
    input din,
    output dout
    );
    reg dout;
    reg[2:0] cs,ns;
    parameter s0=3'b000,
              s1=3'b001,
              s2=3'b010,
              s3=3'b011,
              s4=3'b100;
    always @(posedge clk or posedge clr)
    begin
    if(clr==1)cs<=s0;
```

```
        else       cs<=ns;
        end
        always @(cs or din)
        begin
            case(cs)
                s0: if(din==1) ns<=s1;else ns<=s0;
                s1: if(din==1) ns<=s2;else ns<=s0;
                s2: if(din==0) ns<=s3;else ns<=s2;
                s3: if(din==1) ns<=s4;else ns<=s0;
                s4: if(din==0) ns<=s0;else ns<=s2;
                default:ns<=s0;
            endcase
            if(ns==s4)dout=1;else dout=0;
        end
endmodule
```

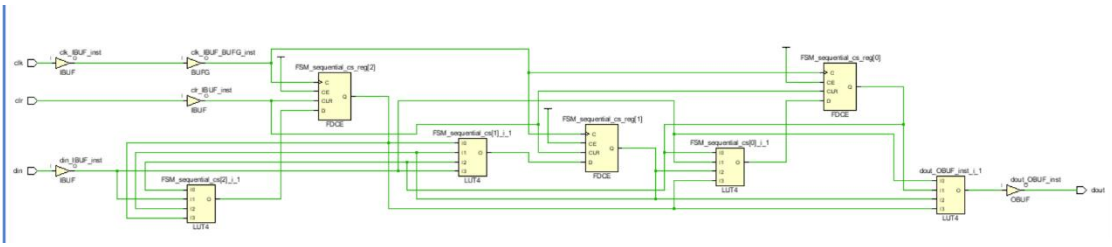
测试文件描述如下：

```
module moore_1_tb(
        );
        reg clk,clr;
        reg din;
        wire dout;
        moore_1 uo(.clk(clk),.clr(clr),.din(din),.dout(dout));
        initial begin
            clk=0;
            clr=0;
            din=0;
        end
        always begin
        #10 clk=~clk;
        end
        always begin
        #10 clk=1;
        #10 clr=0;
        #1000 clr=1;
        end
        always begin
        #20 din=1'b0;
        #20 din=1'b1;
        #20 din=1'b1;
        #20 din=1'b0;
        #20 din=1'b1;
        #20 din=1'b0;
        end
```

endmodule

1.3. 电路原理图

RTL 原理图如下：



2. 问题重述：

设计一个让发光灯显示不同状态的状态机。比如：

要求有 3 个显示状态， 且 3 个状态循环切换：a -> b -> c -> a

a. 选 2 个 led 灯按 1s 的间隔闪烁 5 次；

b. 选 3 个 led 灯按模 8 递增计数；

c. 选 4 个 led 灯按右移方式逐个点亮；**（完成者：昌久冬）**

2.1. 功能简述

使用 Vivado 软件以及 verilog 来实现让发光灯显示不同状态的状态机。

让发光灯显示不同状态的状态机的原理如下：

状态 a：计数器计数 10 次，两个 led 灯接口信号不断翻转，一个周期后，状态转换至 b，计数器清零，灯泡关闭。

状态 b：计数器计数 8 次，计数一个周期，器件信号 2，3，4 按模 8 递增计数，一个周期后，状态转换至 c，计数器清零，灯泡关闭。

状态 c：计数器计数 4 次，灯泡按右移方式逐个点亮，计数一个周期，状态转换至 a，计数器清 0，灯泡关闭。

2.2. Verilog 程序

按照状态变换图，Verilog 语言描述如下;

```
module exp_4_2(
        input clk,
        input clr,
        output dout,
        output d1,
        output d2,
        output d3,
        output d4
    );
        reg[1:0] dout;
        reg[1:0] cs,ns;
        reg[3:0] cnt1,cnt2,cnt3;
        reg d1;
        reg d2;
        reg d3;
        reg d4;
        parameter     a=3'b00,
```

```
                b=3'b01,
                c=3'b10;
always @(posedge clk or posedge clr)
    begin
    if(clr==1)
        begin
            ns<=a;
            cs<=a;
            d1<=1'b1;
            d2<=1'b1;
            d3<=1'b1;
            d4<=1'b1;
            cnt1<=4'd0;
            cnt2<=4'd0;
            cnt3<=4'd0;
            ns<=a;
        end
    else
    begin
    cs<=ns;
    if(cs==a&&ns==a)
    begin
            if(cnt1==4'd0)//计数开始，灯亮
            begin
              d1<=1'b0;
              d2<=1'b0;
              cnt1=cnt1+1'b1;
            end
            else if(cnt1==4'd9)//计数结束，灯灭
            begin
              d1<=1'b1;
              d2<=1'b1;
              cnt1=4'd0;
              ns<=b;
            end
            else//中间状态，灯闪烁
            begin
              cnt1=cnt1+1'b1;
              d1<=~d1;
              d2<=~d2;
            end
    end
    else cnt1<=4'd0;
    if(cs==c&&ns==c)
```

```
begin
    if(cnt2==4'd0)//计数开始，灯亮
    begin
        d1<=1'b0;
        d2<=1'b1;
        d3<=1'b1;
        d4<=1'b1;
        cnt2<=cnt2+1'b1;
    end
    else if(cnt2==4'd1)//计数开始，灯亮
    begin
        d1<=1'b1;
        d2<=1'b0;
        d3<=1'b1;
        d4<=1'b1;
        cnt2<=cnt2+1'b1;
     end
     else if(cnt2==4'd2)//计数开始，灯亮
     begin
         d1<=1'b1;
         d2<=1'b1;
         d3<=1'b0;
         d4<=1'b1;
         cnt2<=cnt2+1'b1;
     end
     else if(cnt2==4'd3)//计数开始，灯亮
     begin
         d1<=1'b1;
         d2<=1'b1;
         d3<=1'b1;
         d4<=1'b0;
         cnt2<=cnt2+1'b1;
     end
     else
     begin
         d1<=1'b1;
         d2<=1'b1;
         d3<=1'b1;
         d4<=1'b1;
         cnt2<=4'd0;
         ns<=a;
     end
end
else cnt2<=4'd0;
```

```
if(cs==b&&ns==b)
begin
    case(cnt3)
        4'd0:begin
            d2<=1'b1;
            d3<=1'b1;
            d4<=1'b1;
            cnt3<=cnt3+1'b1;
        end
        4'd1:begin
            d2<=1'b1;
            d3<=1'b1;
            d4<=1'b0;
        cnt3<=cnt3+1'b1;
        end
        4'd2:begin
            d2<=1'b1;
            d3<=1'b0;
            d4<=1'b1;
        cnt3<=cnt3+1'b1;
        end
        4'd3:begin
            d2<=1'b1;
            d3<=1'b0;
            d4<=1'b0;
        cnt3<=cnt3+1'b1;
        end
        4'd4:begin;
            d2<=1'b0;
            d3<=1'b1;
            d4<=1'b1;
        cnt3<=cnt3+1'b1;
        end
        4'd5:begin
            d2<=1'b0;
            d3<=1'b1;
            d4<=1'b0;
        cnt3<=cnt3+1'b1;
        end
        4'd6:begin
            d2<=1'b0;
            d3<=1'b0;
            d4<=1'b1;
        cnt3<=cnt3+1'b1;
```

```
                    end
                    4'd7:begin
                        d2<=1'b0;
                        d3<=1'b0;
                        d4<=1'b0;
                    cnt3<=cnt3+1'b1;
                    end
                    default:begin
                        d2<=1'b1;
                        d3<=1'b1;
                        d4<=1'b1;
                        cnt3<=4'd0;
                        ns<=c;
                    end
                endcase
            end
            else cnt3<=4'd0;
            dout=cs;
            end
          end
endmodule
```

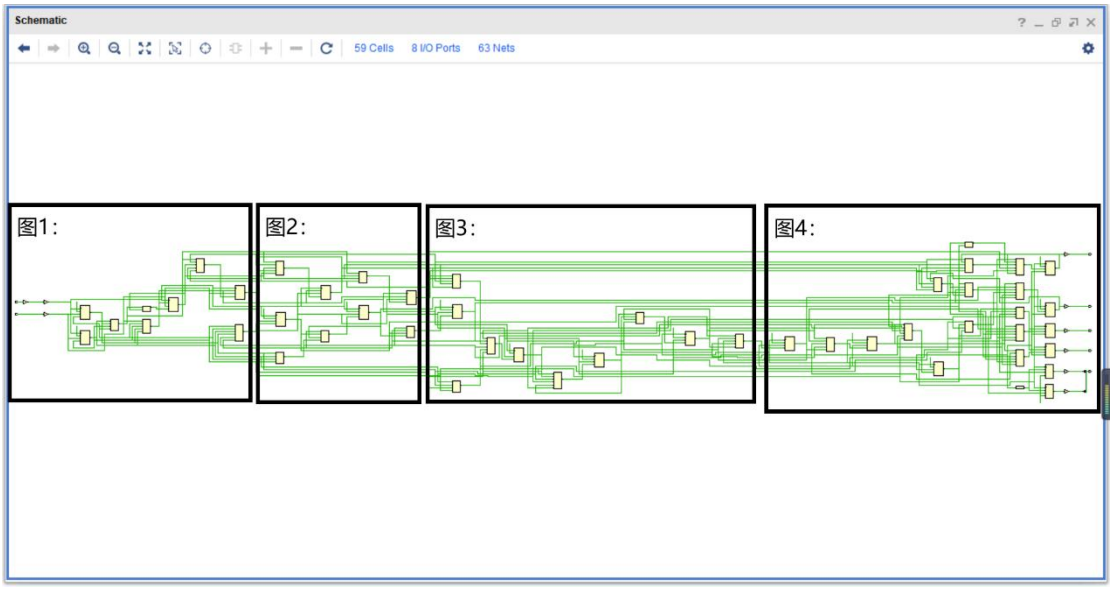
测试文件描述如下：

```
module exp_4_2_tb(
    );
    reg clk,clr;
    wire [1:0]dout;
    wire d1,d2,d3,d4;
    exp_4_2 uo(.clk(clk),.clr(clr),.dout(dout),.d1(d1),.d2(d2),.d3(d3),.d4(d4));
    initial begin
            clk=0;
            clr=1;
     end
     always begin
        #10 clk=~clk;
        clr=0;
        end
endmodule
```

2.4. 电路原理图

RTL 原理图如下：

由于电路较为复杂图片较大，整体原理图为：


Schematic
59 Cells
8 I/O Ports
63 Nets
图1:
图2:
图3:
图4:

各部分细节：

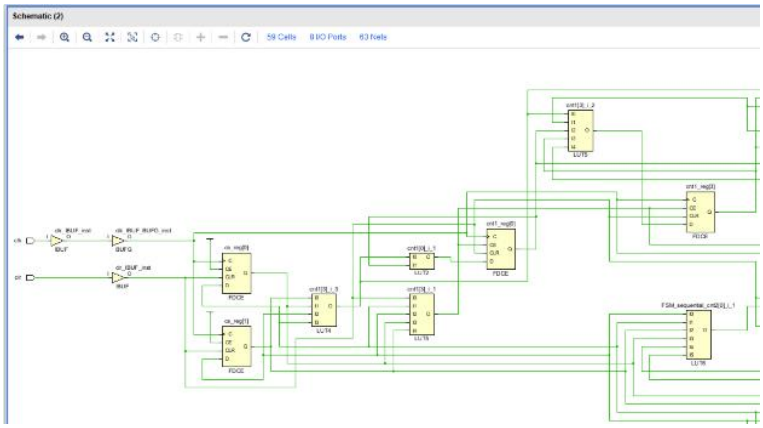

Schematic (2)
59 Cells
8 I/O Ports
63 Nets

图 1

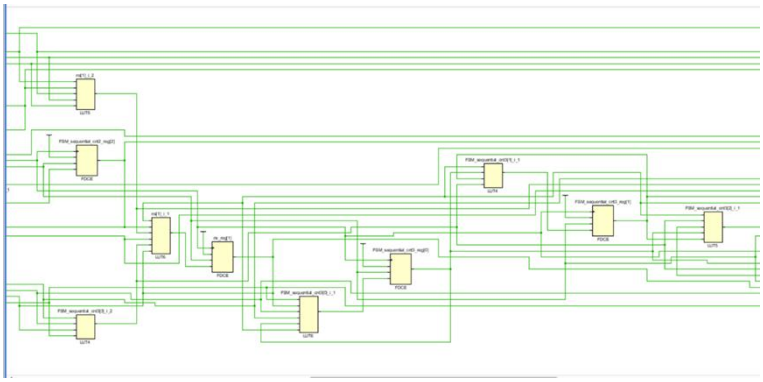


图 2

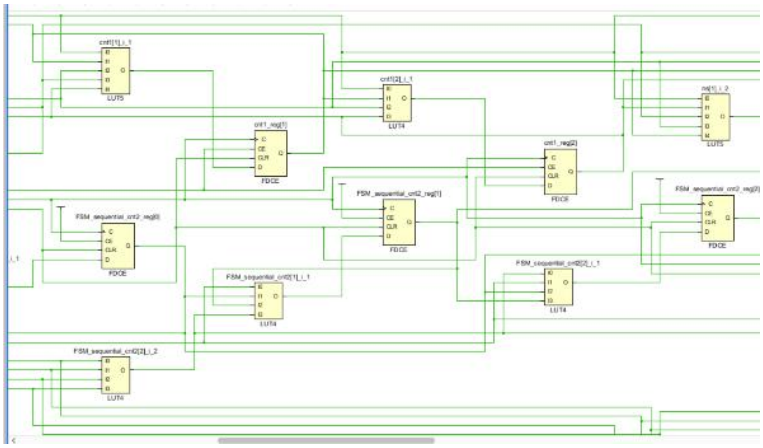


图 3

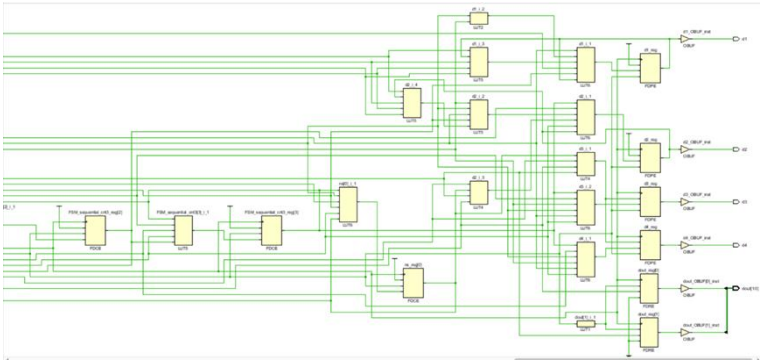


图 4

三、简要操作步骤

1、根据 Vivado 的设计流程，建立本次实验项目的工程文件；

2、建立本次实验内容的 Verilog 功能模块，修改语法错误；

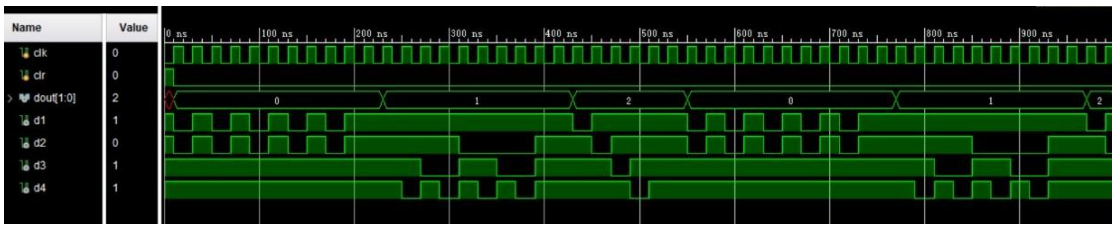
3、建立仿真文件，进行电路功能仿真；

4、查看仿真结果，观察电路功能是否符合预期，不符合需对功能模块和仿真文件进行修改直到电路功能符合预期。

四、实验结果

问题 1 仿真波形如下：

问题 2 仿真波形如下：



五、线下实验现象记录（完成者：昌久冬）

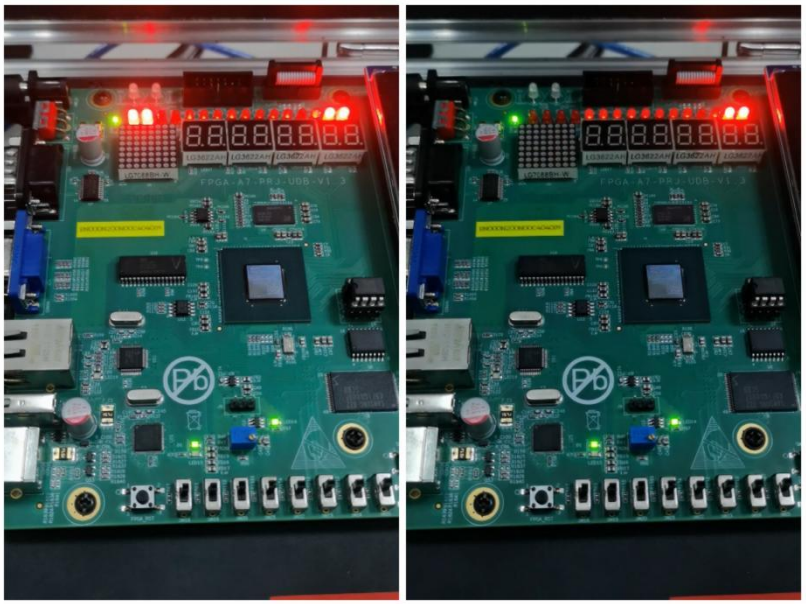
问题 2：让 LED 灯显示不同状态的状态机实验现象如下：

各输入输出接口如图所示：

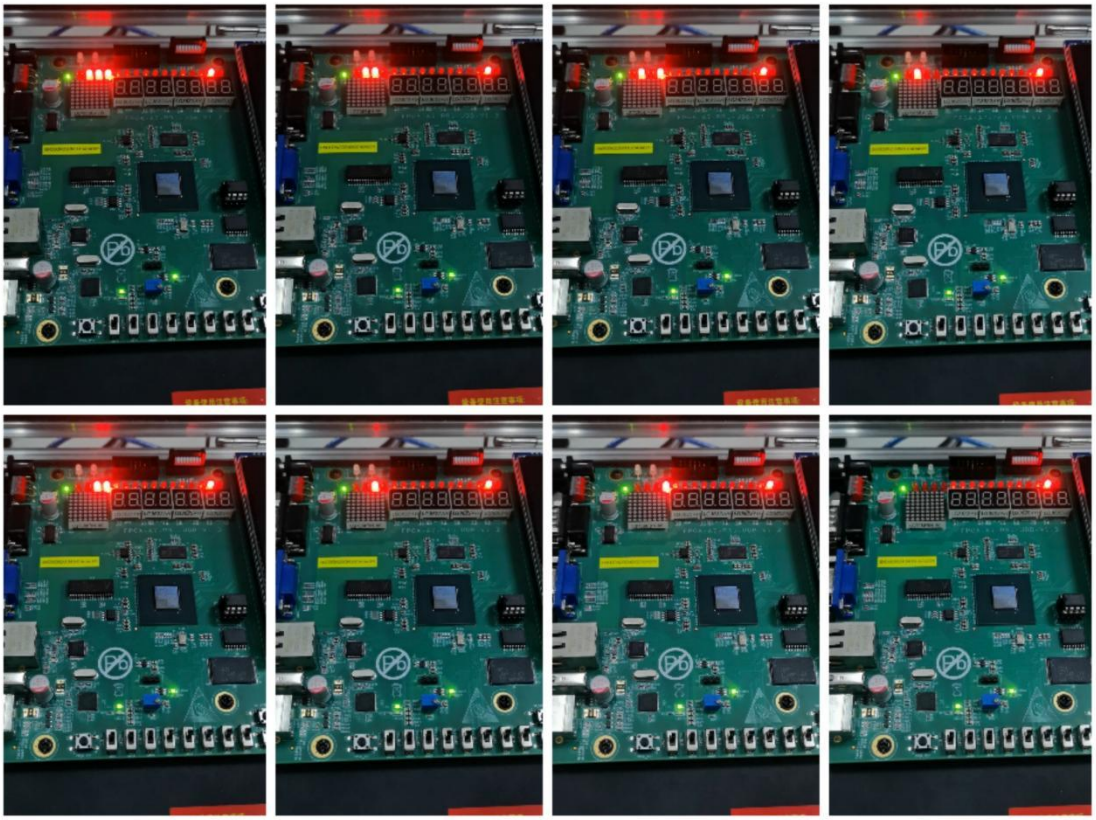
```
set_property PACKAGE_PIN J21 [get_ports {dout[1]}]
set_property PACKAGE_PIN AC19 [get_ports clr]
set_property PACKAGE_PIN K23 [get_ports {dout[0]}]
set_property PACKAGE_PIN H7 [get_ports d1]
set_property PACKAGE_PIN D5 [get_ports d2]
set_property PACKAGE_PIN Y3 [get_ports clk]
set_property PACKAGE_PIN A3 [get_ports d3]
set_property PACKAGE_PIN A5 [get_ports d4]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {dout[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports clr]
set_property IOSTANDARD LVCMOS33 [get_ports d4]
set_property IOSTANDARD LVCMOS33 [get_ports d3]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports d2]
set_property IOSTANDARD LVCMOS33 [get_ports d1]
```

灯亮表示 0，灯灭表示 1，各状态实验现象如下：

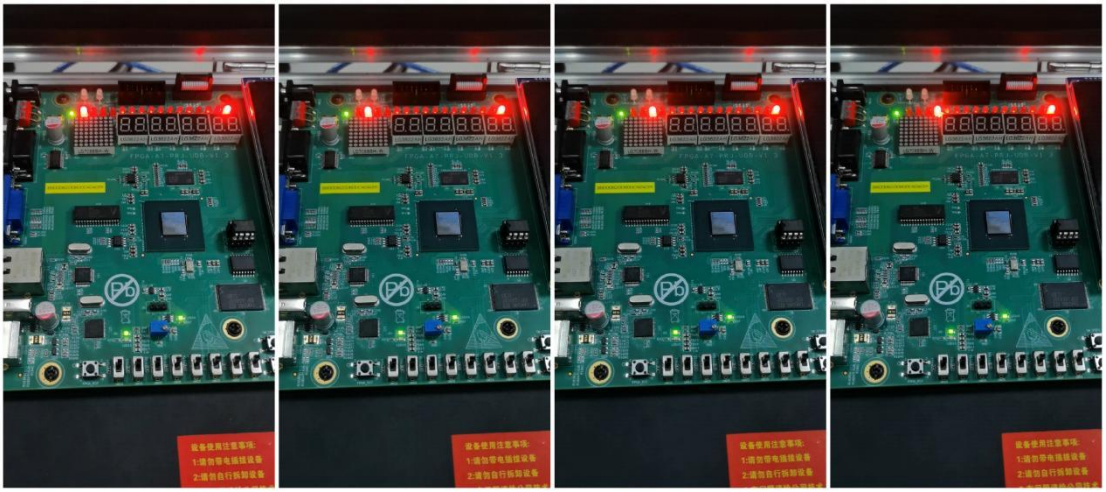
状态 a：2 个 led 灯按 1s 的间隔闪烁 5 次：



状态 b：3 个 led 灯按模 8 递增计数：



状态 c：4 个 led 灯按右移方式逐个点亮：



处于状态 a, b, c 时，右侧表示状态的两盏灯分别处于 00，01，10 态：