南京理工大学计算机科学与工程学院计算机逻辑基础 实验报告

班　　级 9191062302

学生姓名 况宇

学　　号 919107820406

教　　师 余立功

教　　师 潘志兰

实验一 8-3 编码器实现

1.实验目的

1. 熟悉常用编码器，译码器的功能逻辑。
2. 学习组合逻辑电路译码器或编码器的设计方法及应用。
3. 熟悉Verilog 代码的编写方法。
4. 掌握复杂编码译码器的设计方法。

2.实验内容

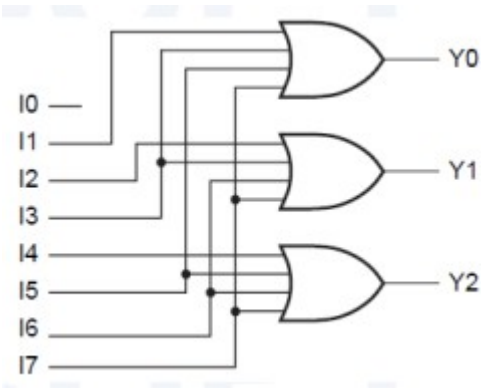
主要内容：用Verilog 来实现8-3 编码器并实现 8-3 编码器的仿真。

实验原理：

编码器是指能将每一组输入信息或数目变换为相应二进码输出，即能完成特定编码功能的逻辑电路。常用的编码器有二进制编码器、二-十进制编码器、优先编码器等。

用n 位二进制代码对 2^n 个信号进行编码的电路，称为二进制编码器。常用的二进制编码器有 4 线-2 线、 8 线-3 线和 16 线-4 线等。由于编码器各个输出信号逻辑表达式的基本形式是有关输入信号的或运算，所以其逻辑电路是由或门组成的阵列，这也是编码器基本电路结构的一个显著特点。

8-3 编码器原理图：



8-3 编码器真值表：

输入din[7:0]									输出dout[2:0]		
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	0	1	1	1
其他									0	0	0

3.实验步骤

（1） 根据Vivado的设计流程，先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，进行电路的功能仿真；
（4） 编译综合，直到没有错误，查看综合结果（原理图及资源利用报告）；

4.实验程序

源文件：

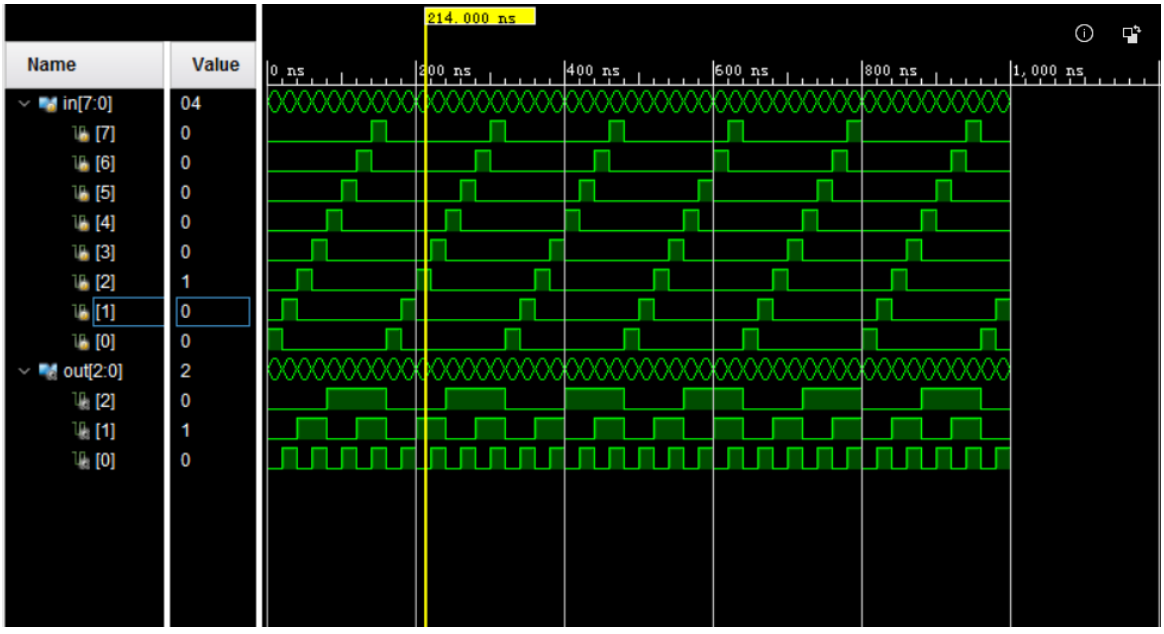
```
module encode8_3(
    input [7:0] in,
    output [2:0] out   );
    reg [2:0] out;
    always @( in )
    begin
        if ( in[0] )  out = 3'b000;
       else if ( in[1] )  out = 3'b001;
       else if ( in[2] )  out = 3'b010;
       else if ( in[3] )  out = 3'b011;
       else if ( in[4] )  out = 3'b100;
       else if ( in[5] )  out = 3'b101;
       else if ( in[6] )  out = 3'b110;
       else if ( in[7] )  out = 3'b111;
       else out = 3'bxxx;
   end
endmodule
```

仿真文件：

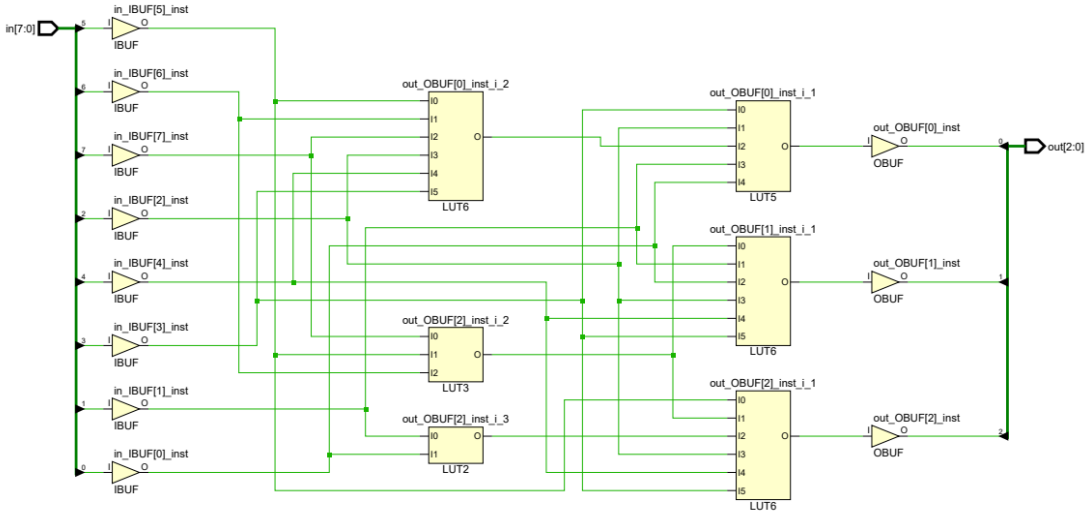
```
module encode8_3_tb(    );
  reg [7:0] in;
  wire [2:0] out;
  encode8_3 u0 ( .in(in) , .out(out) );

  always begin
     in = 8'b00000001 ; # 20;
     in = 8'b00000010 ; # 20;
     in = 8'b00000100 ; # 20;
     in = 8'b00001000 ; # 20;
     in = 8'b00010000 ; # 20;
     in = 8'b00100000 ; # 20;
     in = 8'b01000000 ; # 20;
     in = 8'b10000000 ; # 20;
   end
endmodule
```

5.实验结果

波形图：



原理图：


in[7:0]
in_IBUF[5]_inst
IBUF
in_IBUF[6]_inst
IBUF
in_IBUF[7]_inst
IBUF
in_IBUF[2]_inst
IBUF
in_IBUF[4]_inst
IBUF
in_IBUF[3]_inst
IBUF
in_IBUF[1]_inst
IBUF
in_IBUF[0]_inst
IBUF
out_OBUF[0]_inst_i_2
LUT6
out_OBUF[2]_inst_i_2
LUT3
out_OBUF[2]_inst_i_3
LUT2
out_OBUF[0]_inst_i_1
LUT5
out_OBUF[1]_inst_i_1
LUT6
out_OBUF[2]_inst_i_1
LUT6
out_OBUF[0]_inst
OBUF
out_OBUF[1]_inst
OBUF
out_OBUF[2]_inst
OBUF
out[2:0]

实验二 ：数据通道选择器的设计及应用实验

1．实验目的：

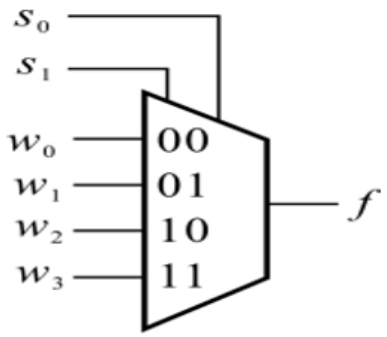
学习组合逻辑电路数据选择器的设计方法及应用。

2．实验内容：

用verilog来实现2选1数据选择器。

实验原理：

在数字系统中，经常需要把多个不同通道的信号发送到公共的信号通道上，通过多路选择器可以完成这一功能。在数字系统设计中，常用CASE语句和IF语句描述多路选择器。多路选择器是由几路数据输入、一位或多位的选择控制，和一路数据输出所组成的，如2选1、4选1、8选1等多路选择器，它们的符号和真值表如下所示（例）：



（a） 逻辑符号

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

3.实验步骤

1. 根据Vivado 的设计流程，先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，进行电路的功能仿真；
4. 编译综合，直到没有错误，查看综合结果。

4. 实验程序

源文件：

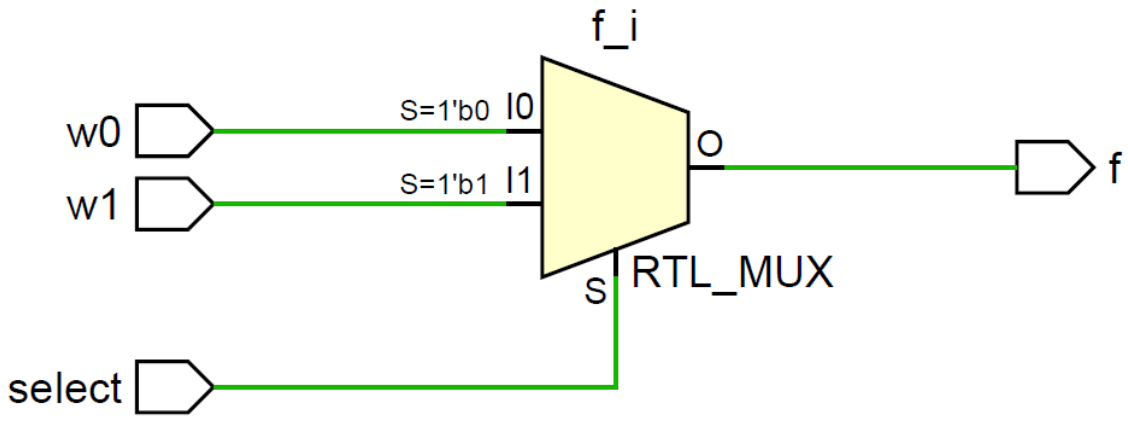
```
module mux2(
    select,w0,w1,f
    );
    input select;
    input w0;
    input w1;
    output reg f;
always @(*)
    begin
        case(select)
        2'b0: f = w0;
        2'b1: f = w1;
        default: f = 1'bx;
        endcase
    end
endmodule
```

仿真文件：

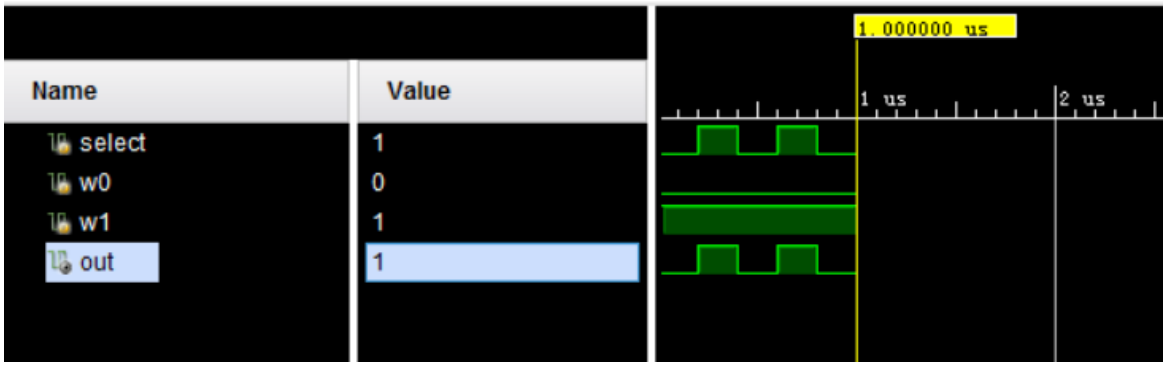
```
module mux2_tb(
    );
    reg select;
    reg w0;
    reg w1;
    wire out;
    mux2 u0(.select(select),.w0(w0),.w1(w1),.f(out));
    always begin
        select = 2'b0;w0 = 1'b0;w1 = 2'b1; # 200;
        select = 2'b1;w0 = 1'b0;w1 = 2'b1; # 200;
    end
endmodule
```

5. 实验结果：

原理图：



波形图：



实验三 ：计数器(或移位寄存器)的设计及应用实验

1．实验目的:

学习时序逻辑电路计数器的设计方法及应用。

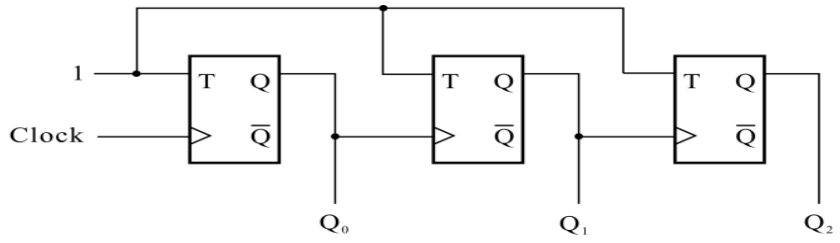
2．实验内容:

（1）用verilog生成一个四位二进制计数器，同步清零；
（2）用verilog生成一个十进制计数器，异步清零；
（3）用verilog生成一个异步可逆十进制计数器，具有同步置数功能；
（4）用verilog生成一个模100的计数器，具有清零、预置功能；
（5）用Verilog生成一个移位寄存器（左移或右移）。
（6）用Verilog实现分频器。

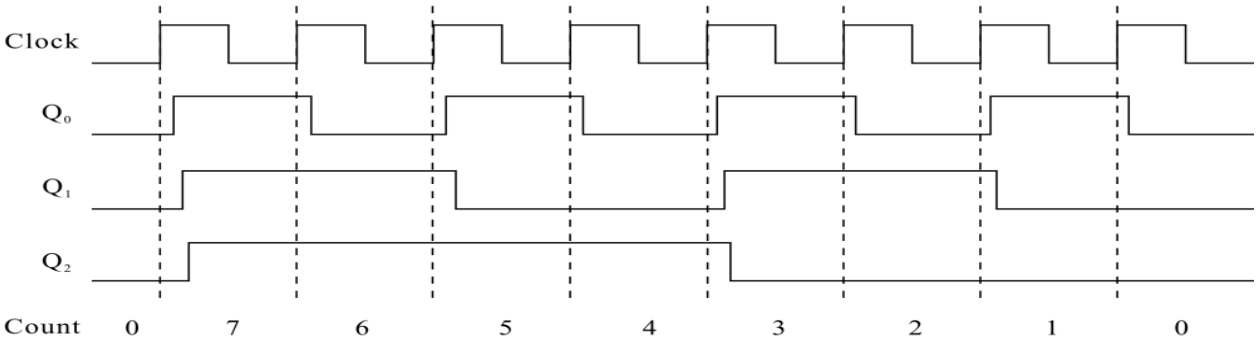
实验原理:

在数字系统设计中，计数器和寄存器是最常用的两类功能器件，也是最常见的时序逻辑元件。计数器是一种能统计输入脉冲个数的时序电路，它可以记录特定事件的发生次数，产生控制系统中不同任务的时间间隔，记录特定事件之间的时间间隔等，它可以用于定时器、分频器、程序控制器、信号发生器等多种数字设备中。计数器按脉冲的作用方式分类，可分为同步计数器和异步计数器。

在同步计数器中，各个触发器的时钟输入端和同一个时钟脉冲源相连，因而所有触发器状态（即计数器状态）的改变都与时钟脉冲同步。而在异步计数器中，有的触发器直接受输入计数脉冲控制，有的是利用其他触发器输出作为时钟输入信号，因此所有触发器状态的改变有先有后，是异步的，如模8计数器，如下图：



(a) 电路



(b) 波形图

3.实验步骤

（1） 根据Vivado的设计流程，先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，进行电路的功能仿真；

（4） 编译综合，直到没有错误，查看综合结果（原理图及资源

利用

报告）；

（5） 建立约束文件xdc，生成bit文件；

（6） 正确连接实验箱，下载bit文件，记录实验现象及结果。

4.实验程序

源文件：

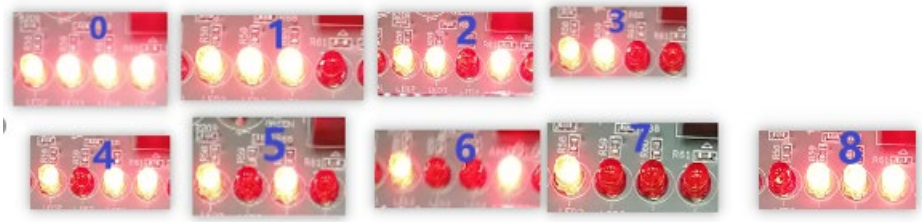
```
module counter10(CLK,RST,COUT,DOUT);
    input CLK,RST;
    output[4:0] DOUT;
    output reg COUT;
    reg [3:0] Q1;
    always@(posedge CLK or negedge RST)
        begin
            if(!RST)
                Q1 <= 0;
            else if(Q1 < 9)
                Q1=Q1+1;
            else
                Q1=4'b0000;
        end
    assign DOUT=Q1;
    always@(Q1)
        begin
            if(Q1==4'h9)
                COUT=1'b1;
            else
                COUT=1'b0;
        end
endmodule
```

140%

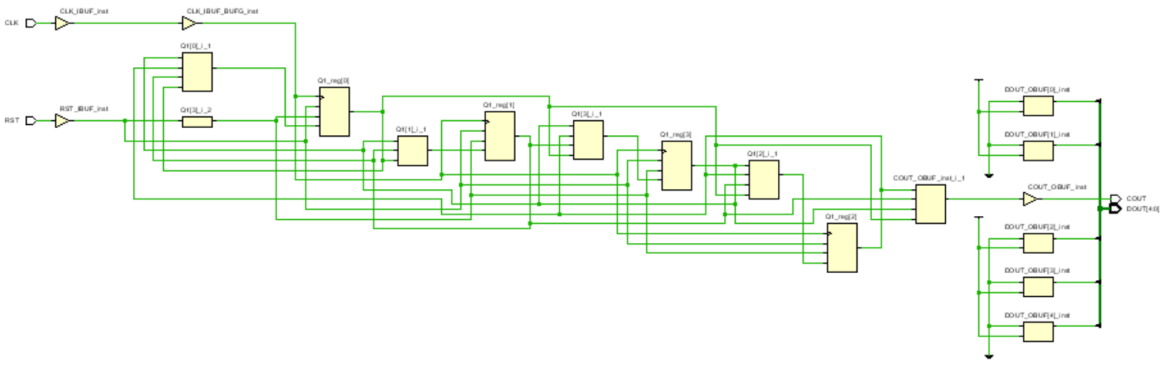
引脚：

```
set_property IOSTANDARD LVCMOS33 [get_ports {DOUT[4]}]
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 IOSTANDARD LVCMOS33 [get_ports COUT]
set_property IOSTANDARD LVCMOS33 [get_ports CLK]
set_property IOSTANDARD LVCMOS33 [get_ports RST]
set_property PACKAGE_PIN H7 [get_ports {DOUT[4]}]
set_property PACKAGE_PIN D5 [get_ports {DOUT[3]}]
set_property PACKAGE_PIN A3 [get_ports {DOUT[2]}]
set_property PACKAGE_PIN A5 [get_ports {DOUT[1]}]
set_property PACKAGE_PIN A4 [get_ports {DOUT[0]}]
set_property PACKAGE_PIN AD24 [get_ports CLK]
set_property PACKAGE_PIN J25 [get_ports COUT]
set_property PACKAGE_PIN AC21 [get_ports RST]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets CLK_IBUF]
```

5.实验结果



原理图：



实验四 ： 状态机的设计及应用实验

1.实验目的

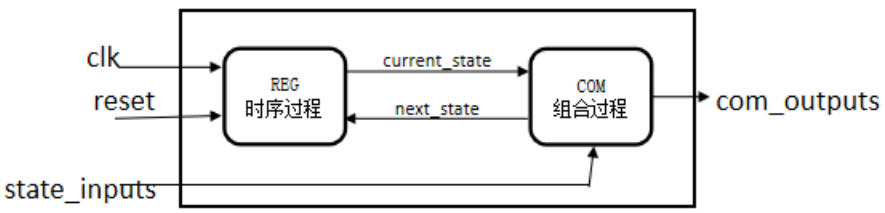
学习时序逻辑电路中Mealy和Moore状态机的设计方法及应用。

2.实验内容

设计一个序列检测状态机，实现Mealy型或Moore型。

实验原理

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:



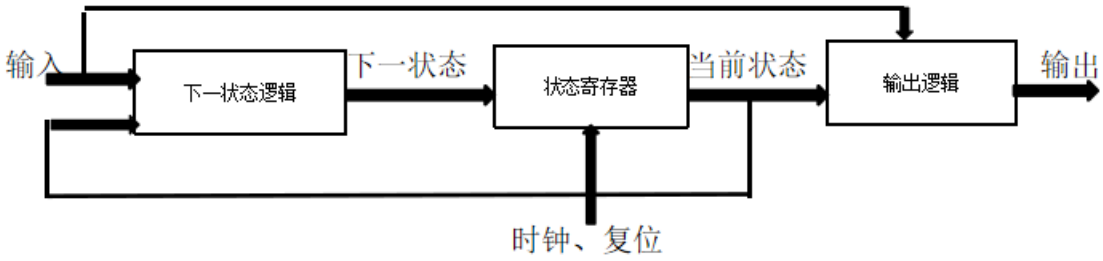
① 从状态机的信号输出方式上分，有Mealy型和Moore型两种状态机。

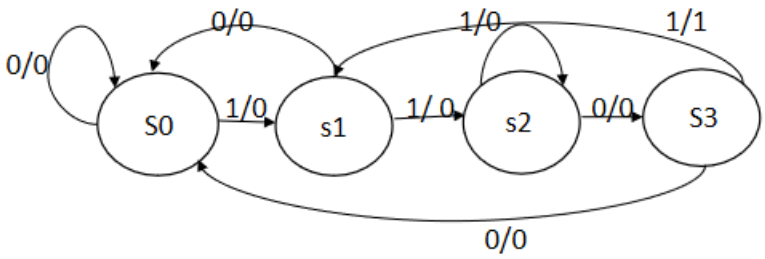
②从状态机的描述结构上分，有单过程状态机多过程状态机。

③从状态表达方式上分，有符号化状态机和确定状态编码的状态机。

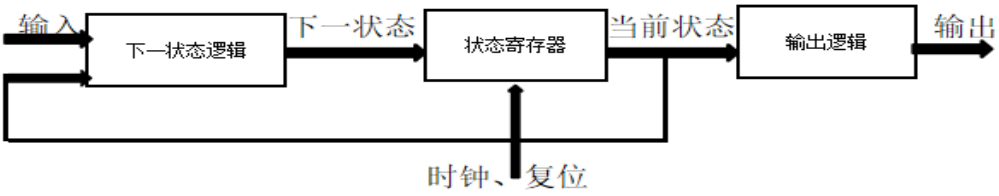
④从状态机编码方式上分，有顺序编码状态机、一位热码编码状态机或其他编码方式状态机。

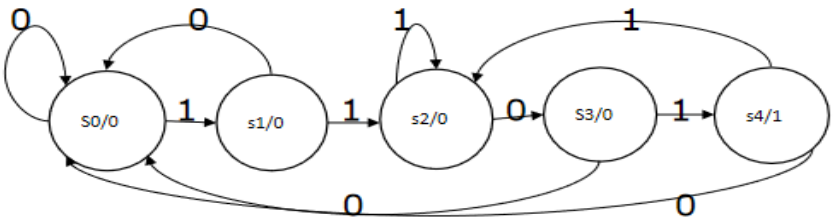
Mealy型状态机如下图 所示，它的输出由状态机的输入和状态机的状态共同决定。





Moore型状态机结构如下图所示，与Mealy型状态机区别在于，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。





3.实验步骤

（1） 根据Vivado的设计流程，先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，进行电路的功能仿真；

（4） 编译综合，直到没有错误，查看综合结果（原理图及资源利用报告）；

（5） 建立约束文件xdc，生成bit文件；

（6） 正确连接实验箱，下载bit文件，记录实验现象及结果。

4.实验程序

源文件：

```
module xulie406(
    input clk,
    input clr,
    input din,
    output reg dout
    );
    reg [3:0] cs;
    reg [3:0] ns;
    parameter s0=4'b1101, s1=4'b1010, s2=4'b0101, s3=4'b1011, s4=4'b0110, s5=4'b1100, s6=4'b1001, s7=4'b0011, s8=4'b0111, s9=4'b1110;
    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==0) ns<=s1;else ns<=s0;
            s1:if(din==1) ns<=s2;else ns<=s1;
            s2:if(din==1) ns<=s3;else ns<=s1;
            s3:if(din==0) ns<=s4;else ns<=s0;
            s4:if(din==0) ns<=s5;else ns<=s2;
            s5:if(din==1) ns<=s6;else ns<=s2;
            s6:if(din==1) ns<=s7;else ns<=s1;
            s7:if(din==1) ns<=s8;else ns<=s4;
            s8:if(din==0) ns<=s9;else ns<=s1;
            s9:if(din==0) ns<=s1;else ns<=s0;
            default:ns<=s0;
         endcase
    end
always@(cs)
begin
     if(cs==s9)
        dout=1;
     else
        dout=0;
end
endmodule
```

仿真文件：

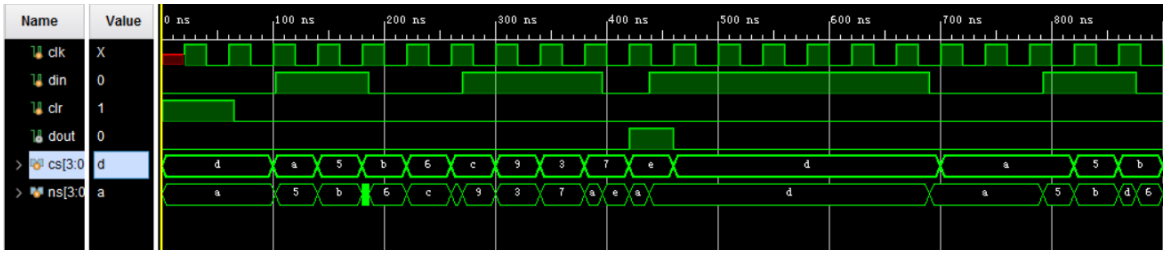
仿真的预置序列为：0110011100

```
module xulie406_tb(

    );
reg clk;
reg din;
wire dout;
reg clr;
xulie u0(.clk(clk),.clr(clr),.din(din),.dout(dout));
initial begin
clr=1; #20;
clk=0; #20;
end
always
    begin
    clk=~clk; # 20;
    end
always
    begin
    clr=1; # 65;
    clr=0; # 1000;
    end
always begin
    din=0;#60;
    din=0;#42;
    din=1;#42;
    din=1;#42;
    din=0;#42;
    din=0;#42;
    din=1;#42;
    din=1;#42;
    din=1;#42;
    din=0;#42;
    din=1;#42;
    din=1;#42;
    din=1;#42;
    din=1;#42;
    din=1;#42;
    din=1;#42;
end
endmodule
```

5.实验结果

仿真的预置序列为：0110011100

波形图：



原理图：