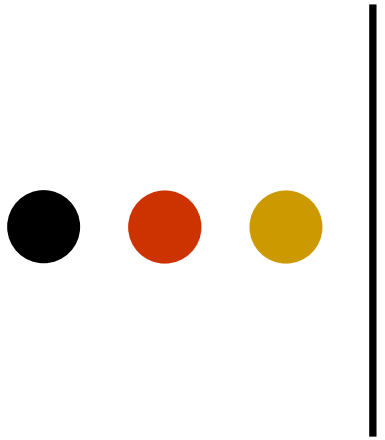


第6章

实用状态机设计技术

6.1 Verilog状态机的一般形式

6.1.1 状态机的特点与优势

（1）高效的顺序控制模型。

（2）容易利用现成的EDA工具进行优化设计。

（3）系统性能稳定。

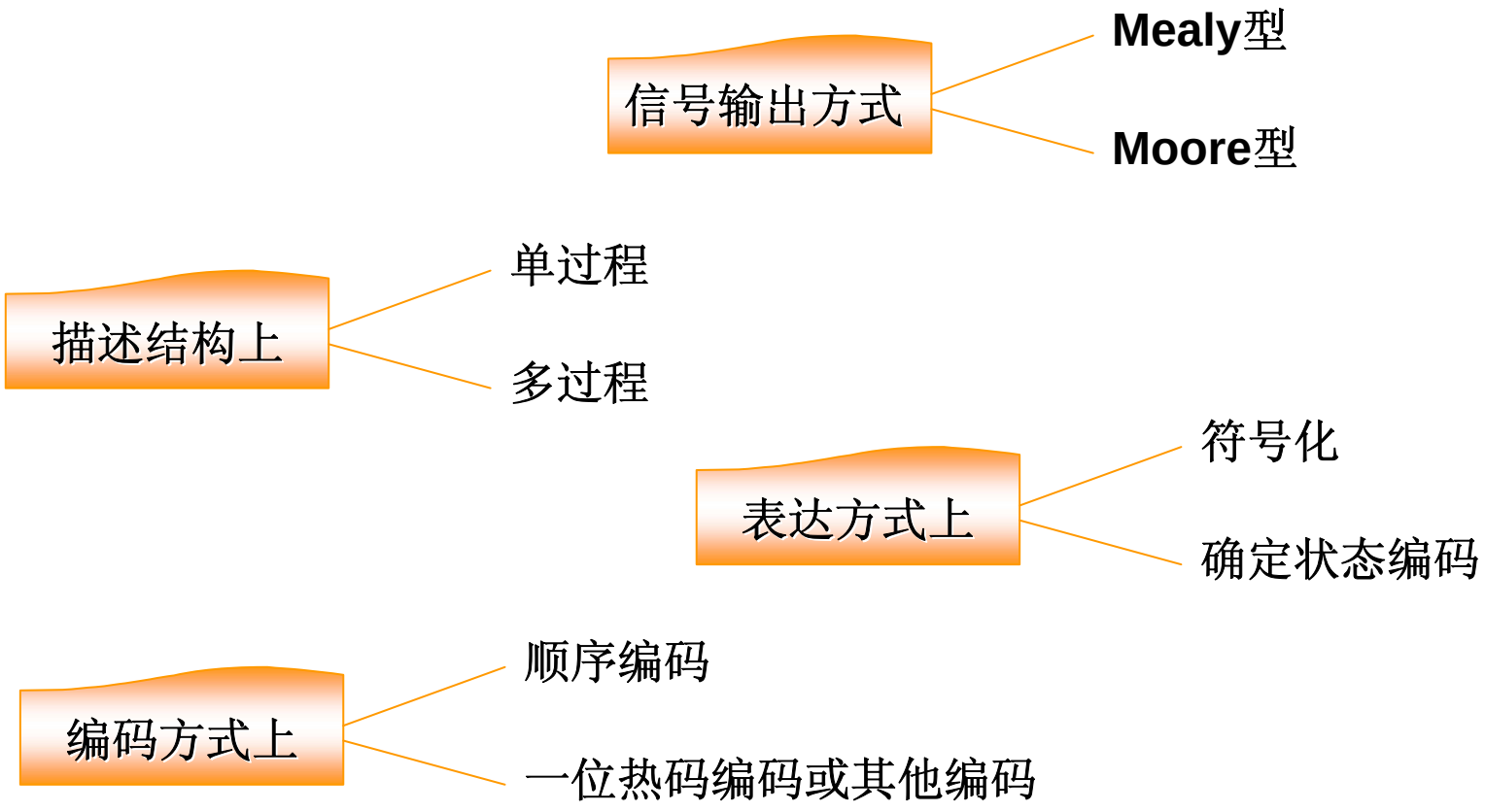
（4）设计实现效率高。

（5）高速性能。

（6）高可靠性能。

6.1 Verilog状态机的一般形式

6.1.2 Verilog状态机的一般结构



6.1 Verilog状态机的一般形式

6.1.2 Verilog状态机的一般结构

1. 说明部分

```
parameter[2:0]  s0=0, s1=1, s2=2 , s3=3, s4=4;
reg[2:0] current_state, next_state;
```

2. 主控时序过程

6.1 Verilog状态机的一般形式

6.1.2 Verilog状态机的一般结构

3. 主控组合过程

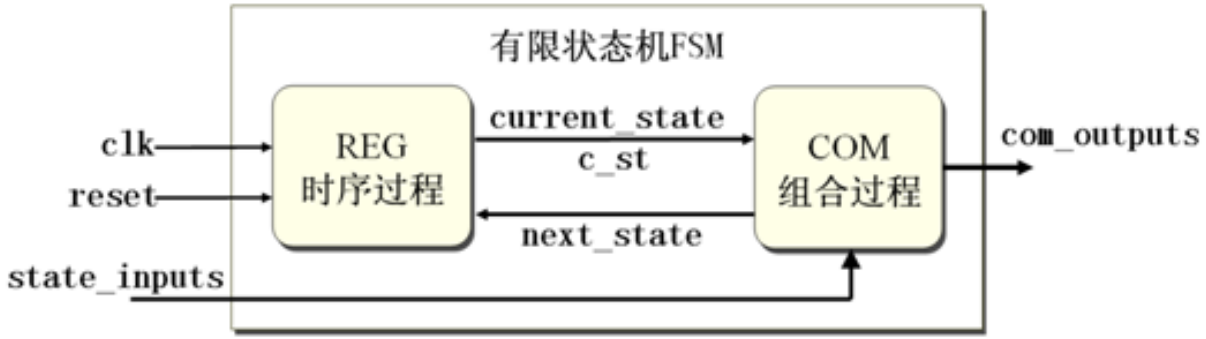


图 6-1　状态机一般结构示图

4. 辅助过程

【例 6-1】

```
module FSM_EXP (clk, reset, state_inputs, comb_outputs);
input clk; input reset;                  // 状态机工作时钟和状态机复位控制
input [0:1] state_inputs;                // 来自外部的状态机控制信号
output [3:0] comb_outputs;               // 状态机对外部发出的控制信号输出
reg [3:0] comb_outputs;
parameter  s0=0,s1=1,s2=2,s3=3,s4=4 ;       // 定义状态参数
reg [4:0]  c_st, next_state;             // 定义现态和次态的状态变量
  always @(posedge clk or negedge reset)
     begin                               // 主控时序过程
    if (!reset)  c_st<=s0;               // 复位有效时，下一状态进入初态s0
         else    c_st<=next_state ;  end
  always @(c_st or state_inputs) begin          // 主控组合过程
    case (c_st)   //为了在仿真波形中容易看清，将current_state简为c_st
     s0 : begin  comb_outputs<=5 ;    //进入状态s0时，输出控制码5
        if (state_inputs==2'b00)  next_state<=s0; //条件满足，回初态s0
             else  next_state<=s1;  end     //条件不满足，到下一状态s1
     s1 : begin  comb_outputs<=8 ;          //进入状态s1时，输出控制码8
        if (state_inputs==2'b01)  next_state<=s1;
             else  next_state<=s2 ;  end
     s2 :  begin  comb_outputs<=12 ;
        if (state_inputs==2'b10)  next_state<=s0;
             else  next_state<=s3 ;  end
     s3 :  begin  comb_outputs<=14 ;
        if (state_inputs==2'b11)  next_state<=s3;
            else   next_state<=s4 ;  end
     s4 :   begin  comb_outputs<=9 ; next_state<=s0 ;  end
        default : next_state<=s0 ; //现态若未出现以上各态，返回初态s0
     endcase    end
endmodule
```

6.1 Verilog状态机的一般形式

6.1.2 Verilog状态机的一般结构

4. 辅助过程

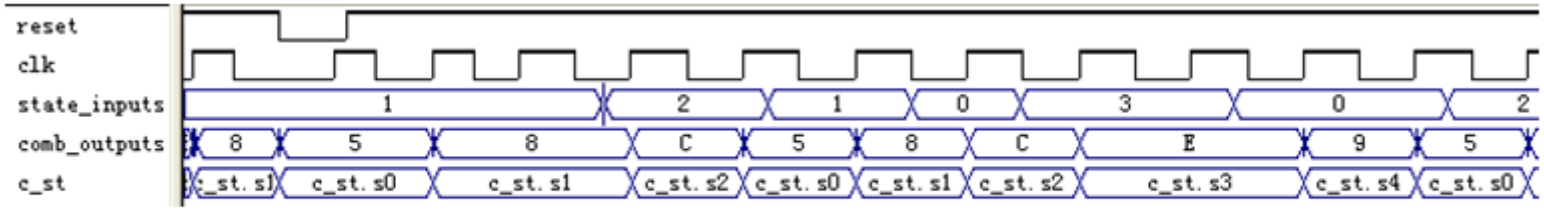


图6-2　例6-1状态机的工作时序

6.1 Verilog状态机的一般形式

6.1.3 初始控制与表述

（1）打开“状态机萃取”开关。

（2）关于参数定义表述。

（3）状态变量定义表述。

6.2 Moore型状态机设计

6.2.1 多过程结构型状态机

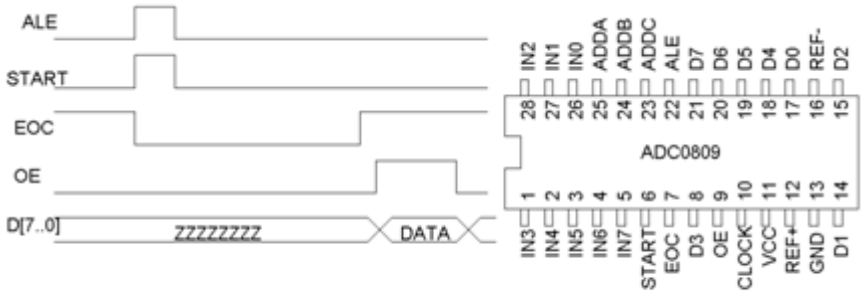


图 6-3 ADC0809 工作时序和芯片引脚图

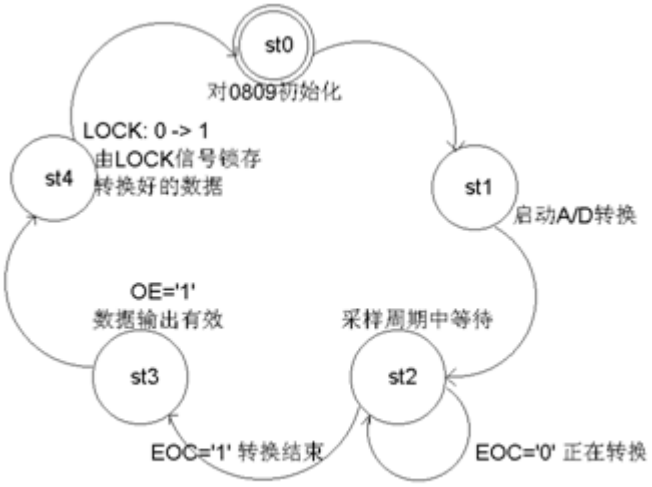


图 6-4 控制 ADC0809 采样状态图

6.2 Moore型状态机设计

6.2.1 多过程结构型状态机

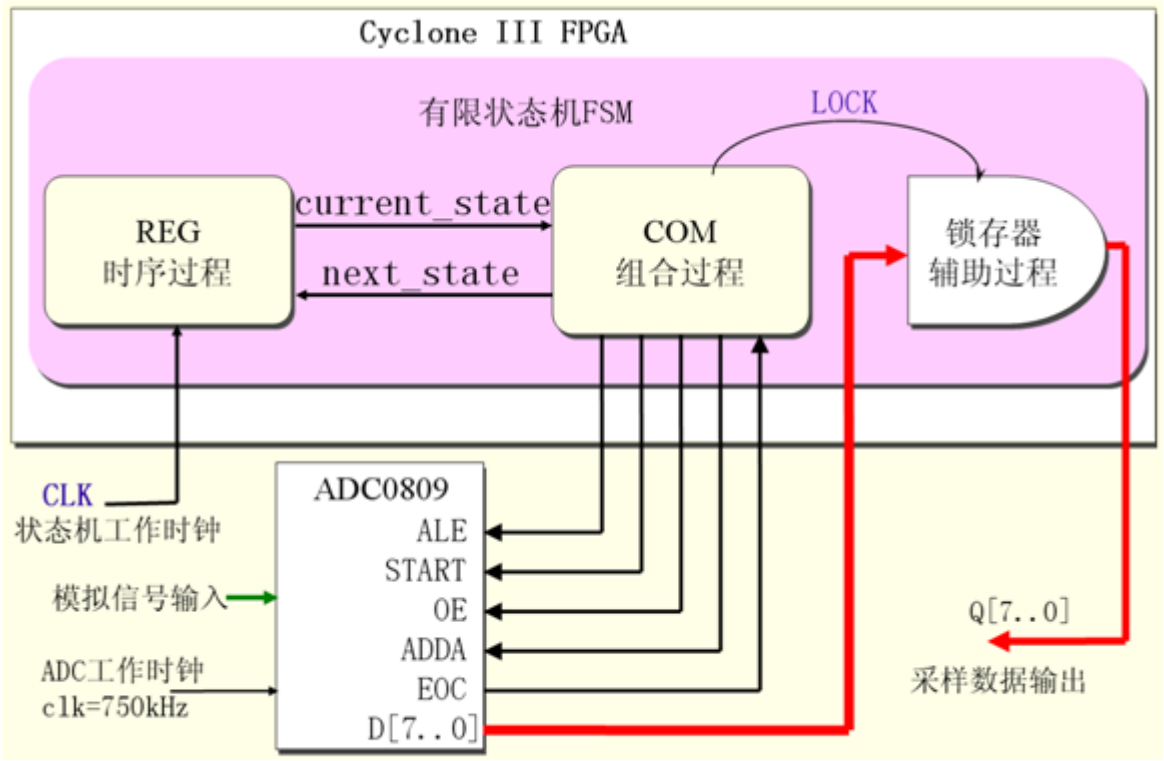


图 6-5 采样状态机结构框图

【例6-2】为了仿真方便观察，输出口增加了内部锁存信号LOCK_T

```
module ADC0809 (D, CLK, EOC, RST, ALE, START, OE, ADDA, Q,LOCK_T);
  input[7:0] D;        //来自0809转换好的8位数据
  input CLK,RST,EOC; //状态机工作时钟、系统复位和转换状态指示(低电平表示正在转换)
  output ALE;          //8个模拟信号通道地址锁存信号
  output START,OE ;      //转换启动信号，和数据输出三态控制信号
  output ADDA,LOCK_T ;   //信号通道控制信号和锁存测试信号
  output[7:0] Q;      reg ALE, START, OE;
  parameter s0=0,s1=1,s2=2,s3=3,s4=4;  //定义各状态子类型
  reg[4:0] cs , next_state ;          //为了便于仿真显示，现态名简为cs
  reg[7:0] REGL;  reg LOCK;            // 转换后数据输出锁存时钟信号
  always @(cs or EOC)  begin          // 组合过程，规定各状态转换方式
    case (cs)
      s0 : begin  ALE=0 ; START=0 ; OE=0 ; LOCK=0 ;
                  next_state <= s1 ;   end          //0809初始化
      s1 : begin  ALE=1 ; START=1 ; OE=0 ; LOCK=0 ;
                  next_state <= s2 ;    end          //启动采样信号START
      s2 : begin  ALE=0 ; START=0 ; OE=0 ; LOCK=0 ;
          if (EOC==1'b1)  next_state = s3 ;         //EOC=0表明转换结束
            else  next_state = s2 ;  end             //转换未结束，继续等待
      s3 : begin ALE=0 ; START=0 ; OE=1;  LOCK=0; //开启OE，打开AD数据口。
                  next_state = s4 ;     end          //下一状态无条件转向s4
      s4 : begin ALE=0 ;  START=0 ; OE=1;  LOCK=1; //开启数据锁存信号
                  next_state <= s0 ;     end
      default : begin  ALE=0 ; START=0 ; OE=0 ; LOCK=0 ;
                  next_state = s0 ;     end
    endcase    end
  always @(posedge CLK or  posedge RST)  begin  //时序过程
    if (RST) cs <= s0 ;
      else  cs <= next_state ; end // 由现态变量cs将当前状态值带出过程
  always @(posedge LOCK)                    //寄存器过程
    if (LOCK)   REGL <= D ; // 此过程中，在LOCK的上升沿将转换好的数据锁入
  assign ADDA =0 ; assign Q = REGL ; //选择模拟信号进入通道IN0
  assign LOCK_T = LOCK ;  //将测试信号输出
endmodule
```

6.2 Moore型状态机设计

6.2.1 多过程结构型状态机

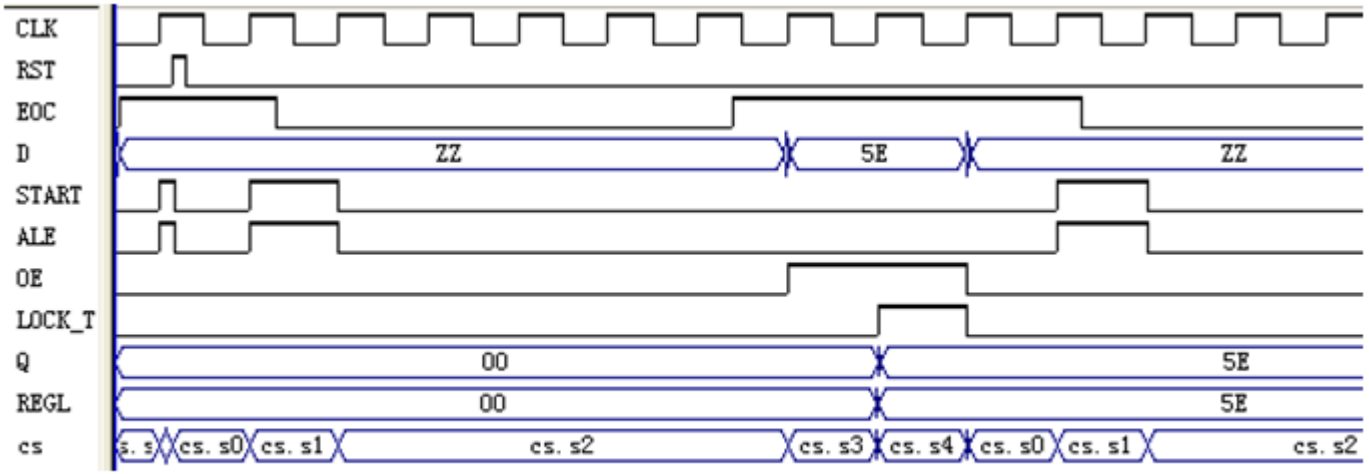


图 6-6 ADC0809 采样状态机工作时序

6.2 Moore型状态机设计

【例 6-3】

```
always @(cs or EOC)  begin
     case (cs)
        s0 : next_state <= s1 ;
        s1 : next_state <= s2 ;
        s2 : if (EOC==1'b1)  next_state=s3 ; else  next_state=s2 ;
        s3 : next_state = s4 ;
        s4 : next_state <= s0 ;
        default : next_state = s0 ;
    endcase   end
always @(cs)  begin
     case (cs)
        s0 : begin  ALE=0 ; START=0 ; OE=0 ; LOCK=0 ;  end
        s1 : begin  ALE=1 ; START=1 ; OE=0 ; LOCK=0 ;  end
        s2 : begin  ALE=0 ; START=0 ; OE=0 ; LOCK=0 ;  end
        s3 : begin  ALE=0 ; START=0 ; OE=1 ; LOCK=0 ;  end
        s4 : begin  ALE=0 ; START=0 ; OE=1 ; LOCK=1 ;  end
  default : begin  ALE=0 ; START=0 ; OE=0 ; LOCK=0 ;  end
     endcase    end
```

6.2.2 序列检测器及其状态机设计

【例 6-4】

```
module  SCHK (CLK, DIN, RST,SOUT);      //11010011高位在前。
   input  CLK, DIN, RST;                 //时钟信号，输入数据，和复位信号
   output SOUT;                          //检测结果输出
   parameter   s0=40, s1=41, s2=42, s3=43, s4=44,
               s5=45, s6=46, s7=47,s8=48 ; // 设定9个状态参数
   reg[8:0] ST,NST ;                     //设定现态变量和次态变量
   always @(posedge CLK or posedge RST)
      if (RST)  ST<=s0 ; else  ST<=NST ;
   always @(ST or DIN)   begin
      case (ST )
          s0 :  if (DIN==1'b1)  NST<=s1; else NST<=s0;
          s1 :  if (DIN==1'b1)  NST<=s2; else NST<=s0;
          s2 :  if (DIN==1'b0)  NST<=s3; else NST<=s0;
          s3 :  if (DIN==1'b1)  NST<=s4; else NST<=s0;
          s4 :  if (DIN==1'b0)  NST<=s5; else NST<=s0;
          s5 :  if (DIN==1'b0)  NST<=s6; else NST<=s0;
          s6 :  if (DIN==1'b1)  NST<=s7; else NST<=s0;
          s7 :  if (DIN==1'b1)  NST<=s8; else NST<=s0;
          s8 :  if (DIN==1'b0)  NST<=s3; else NST<=s0;
       default :                NST<=s0;
      endcase          end
    assign  SOUT = (ST==s8) ;
endmodule
```

6.2 Moore型状态机设计

6.2.2 序列检测器及其状态机设计

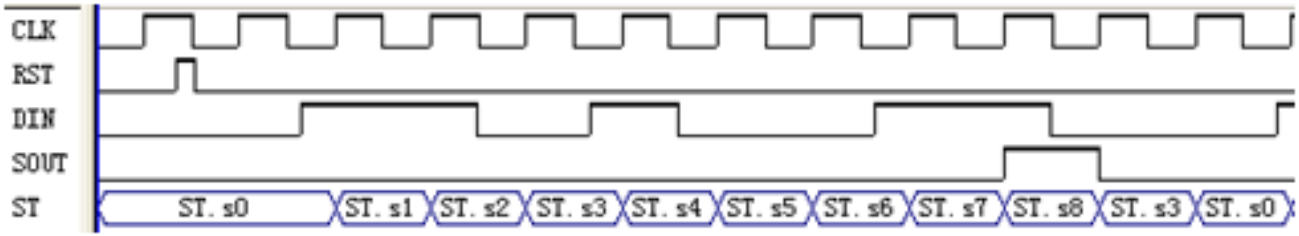


图 6-7　例 6-4 之序列检测器时序仿真波形

6.3 Mealy型状态机设计

【例 6-5】

```
module MEALY1 (CLK, DIN1,DIN2, RST, Q);
input CLK, DIN1,DIN2, RST;   output[4:0] Q;
reg[4:0] Q;   reg[4:0] PST;
parameter st0=0, st1=1, st2=2, st3=3, st4=4;
always @(posedge CLK or posedge RST)   begin :  REG
   if (RST) PST <= st0 ; //现态变量定义
      else   begin
       case (PST)
         st0 :  if (DIN1==1'b1)  PST<=st1 ; else PST<=st0 ;
         st1 :  if (DIN1==1'b1)  PST<=st2 ; else PST<=st1 ;
         st2 :  if (DIN1==1'b1)  PST<=st3 ; else PST<=st2 ;
         st3 :  if (DIN1==1'b1)  PST<=st4 ; else PST<=st3 ;
         st4 :  if (DIN1==1'b0)  PST<=st0 ; else PST<=st4 ;
         default :               PST<=st0 ;
       endcase    end    end
always @(PST or DIN2)   begin :  COM       //输出控制信号的过程
    case (PST)
      st0 :  if (DIN2==1'b1)  Q=5'H10 ; else   Q=5'H0A ;
      st1 :  if (DIN2==1'b0)  Q=5'H17 ; else   Q=5'H14 ;
      st2 :  if (DIN2==1'b1)  Q=5'H15 ; else   Q=5'H13 ;
      st3 :  if (DIN2==1'b0)  Q=5'H1B ; else   Q=5'H09 ;
      st4 :  if (DIN2==1'b1)  Q=5'H1D ; else   Q=5'H0D ;
      default :               Q=5'b00000 ;
     endcase    end
endmodule
```

6.3 Mealy型状态机设计

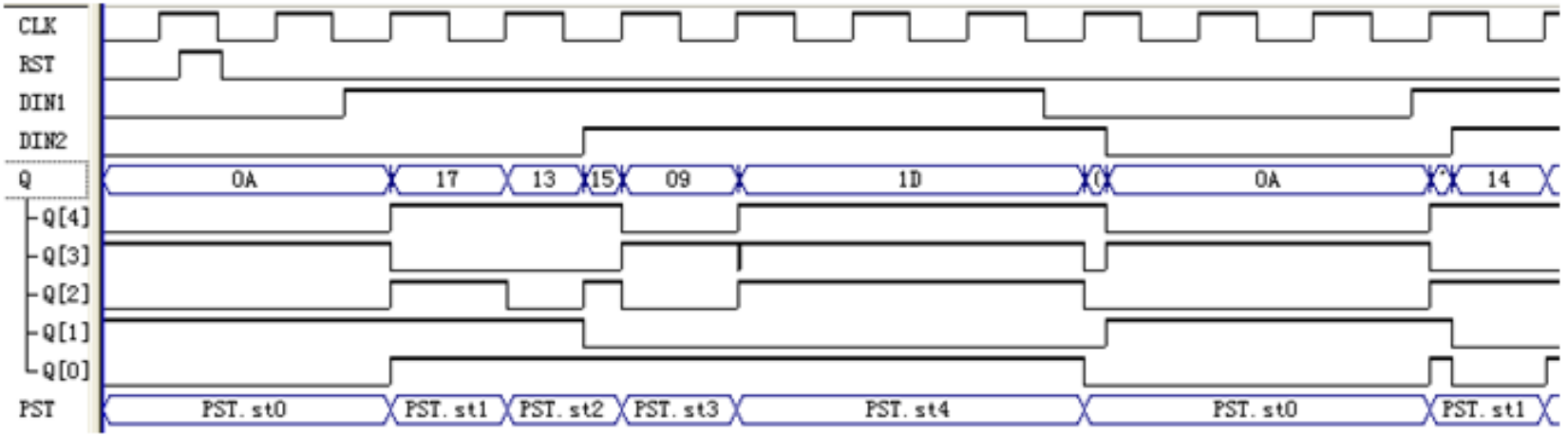


图 6-8　例 6-5 之双过程 Mealy 机仿真波形

6.3 Mealy型状态机设计

【例 6-6】

```
module MEALY2 (CLK, DIN1,DIN2, RST, Q);
  input CLK, DIN1,DIN2, RST;    output[4:0] Q;
  parameter  st0=0, st1=1, st2=2, st3=3, st4=4;
  reg[4:0] PST;  reg[4:0] Q;
  always @(posedge CLK or posedge RST)  begin
    if (RST)  PST <= st0 ;    else  begin
      case (PST)
        st0 :  begin
               begin if (DIN2==1'b1)  Q=5'H10  ; else Q=5'H0A;  end
               begin if (DIN1==1'b1)  PST<=st1 ; else PST<=st0; end
              end
        st1 :  begin
               begin if (DIN2==1'b0)  Q=5'H17  ; else Q=5'H14 ; end
               begin if (DIN1==1'b1)  PST<=st2 ; else PST<=st1; end
              end
        st2 :  begin
               begin if (DIN2==1'b1)  Q=5'H15 ;  else Q=5'H13;  end
               begin if (DIN1==1'b1)  PST<=st3 ; else PST<=st2; end
              end
        st3 :  begin
               begin if (DIN2==1'b0)  Q=5'H1B ;  else Q=5'H09;  end
               begin if (DIN1==1'b1)  PST<=st4 ; else PST<=st3; end
              end
        st4 :  begin
               begin if (DIN2==1'b1)  Q=5'H1D ;  else Q=5'H0D ; end
               begin if (DIN1==1'b0)  PST<=st0 ; else PST<=st4; end
              end
         default :     begin  PST<=st0 ; Q=5'b00000 ; end
       endcase      end      end
endmodule
```

6.3 Mealy型状态机设计

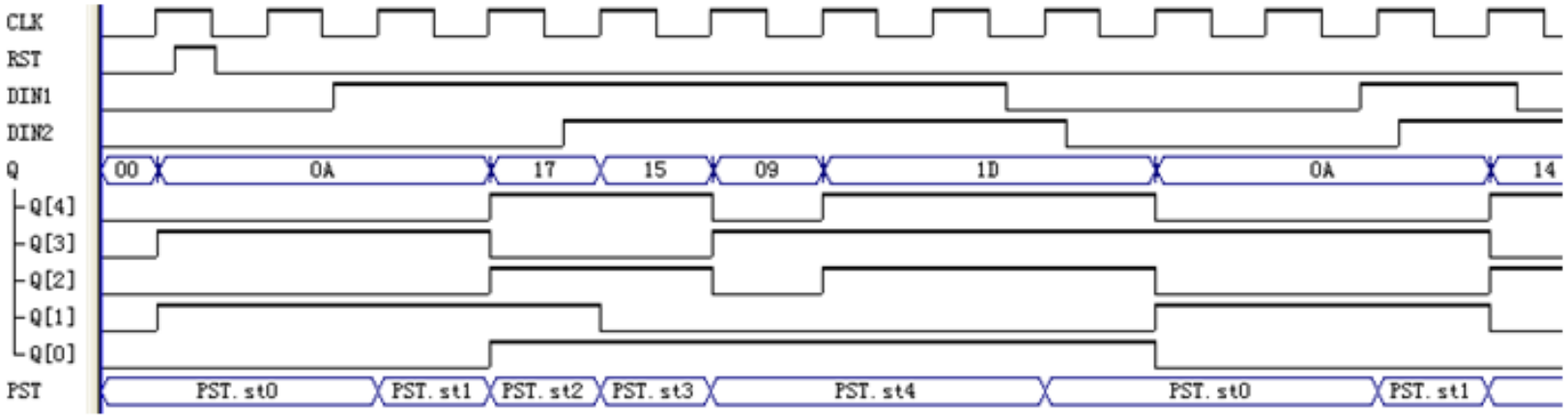


图6-9　例6-6之单过程Mealy机仿真波形

6.3 Mealy型状态机设计

【例 6-7】

```
module SCHK  (CLK, DIN, RST,SOUT);//检测11010011
   input  CLK, DIN, RST;    output SOUT;
   parameter s0=0, s1=1, s2=2, s3=3, s4=4, s5=5, s6=6, s7=7, s8=8;
   reg[8:0] ST ;   reg SOUT;
   always @(posedge CLK)   begin
      SOUT=0;
      if (RST)  ST<=s0 ;   else   begin
         casex (ST )
            s0 :  if (DIN==1'b1)  ST<=s1; else  ST<=s0;
            s1 :  if (DIN==1'b1)  ST<=s2; else  ST<=s0;
            s2 :  if (DIN==1'b0)  ST<=s3; else  ST<=s0;
            s3 :  if (DIN==1'b1)  ST<=s4; else  ST<=s0;
            s4 :  if (DIN==1'b0)  ST<=s5; else  ST<=s0;
            s5 :  if (DIN==1'b0)  ST<=s6; else  ST<=s0;
            s6 :  if (DIN==1'b1)  ST<=s7; else  ST<=s0;
            s7 :  if (DIN==1'b1)  ST<=s8; else  ST<=s0;
            s8 :  begin SOUT=1 ;
                  if (DIN==1'b0)  ST<=s3; else  ST<=s0;  end
          default :  ST<=s0;
        endcase     end     end
endmodule
```

6.3 Mealy型状态机设计

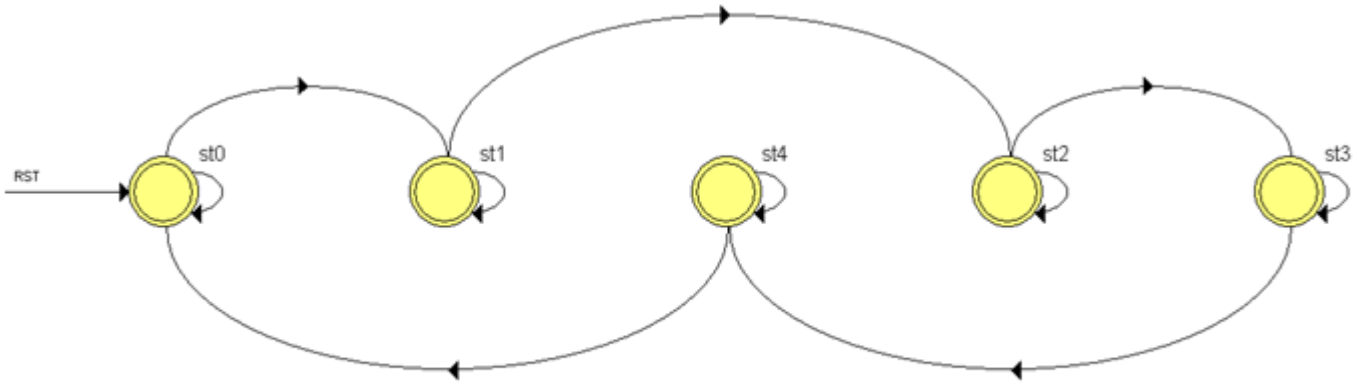


图 6-10　例 6-6，6-5 的状态图（QuartusII 的 State Machine Viewer 状态图）

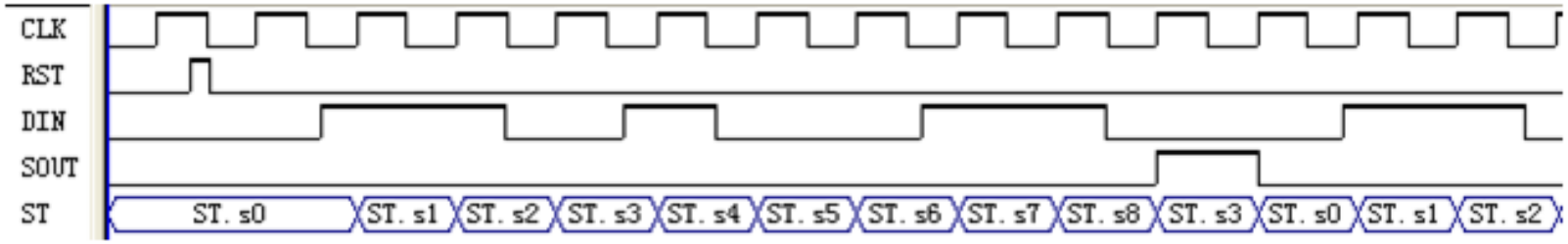


图 6-11　例 6-7 之单过程 Mealy 机仿真波形

6.4 不同编码类型的状态机

6.4.1 直接输出型编码

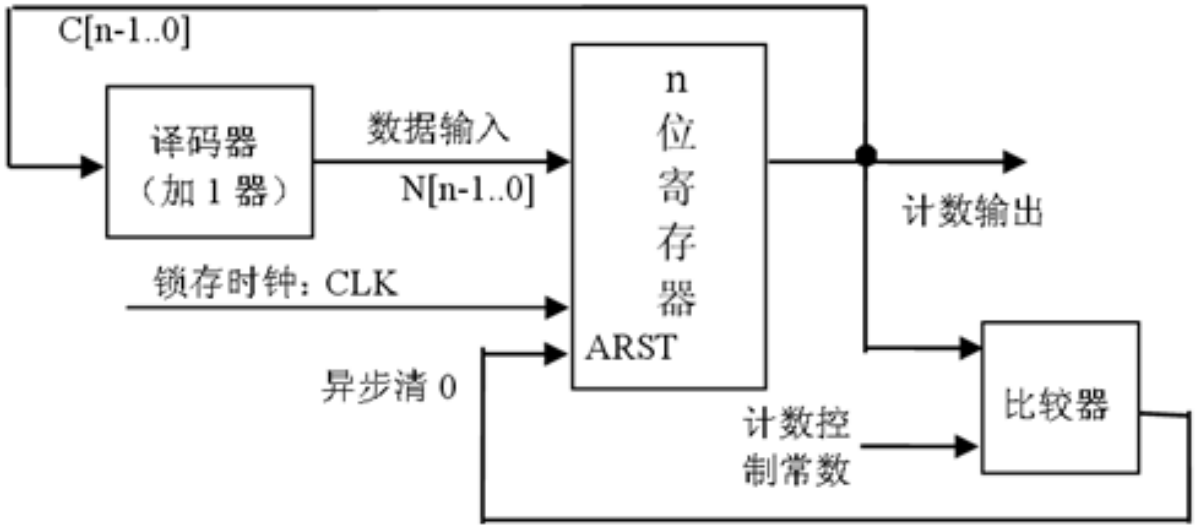


图 6-12 加法计数器一般模型

6.4 不同编码类型的状态机

6.4.1 直接输出型编码

表 6-1　控制信号状态编码表

状态	状态编码					
	START	ALE	OE	LOCK	B	功能说明
s0	0	0	0	0	0	初始态
s1	1	1	0	0	0	启动转换
s2	0	0	0	0	1	若测得 EOC=1 时，转下一状态 ST3
s3	0	0	1	0	0	输出转换好的数据
s4	0	0	1	1	0	利用 LOCK 的上升沿将转换好的数据锁存

6.4 不同编码类型的状态机

6.4.1 直接输出型编码

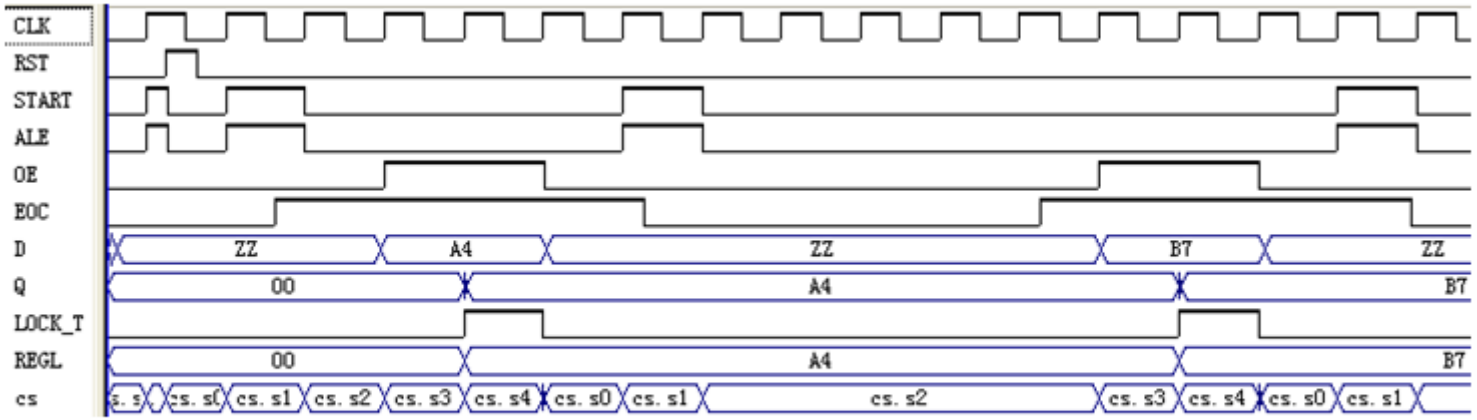


图 6-13 例 6-8 状态机工作时序图

【例 6-8】此程序的硬件实测方法可参考实训项目 6-2

```
module ADC0809  (D, CLK, EOC, RST, ALE, START, OE, ADDA, Q,LOCK_T);
  input[7:0] D;   input CLK,RST,EOC;
  output START,OE , ALE, ADDA,LOCK_T ;   output[7:0] Q;
 parameter s0=5'B00000,s1=5'B11000,s2=5'B00001,s3=5'B00100,s4=5'B00110;
  reg[4:0] cs ,SOUT, next_state ;    reg[7:0] REGL;  reg LOCK;
  always@ (cs or EOC)  begin
   case (cs)
     s0 : begin  next_state<=s1 ;  SOUT=s0 ;   end
     s1 : begin  next_state<=s2 ;  SOUT=s1 ;   end
     s2 : begin  SOUT=s2 ;
          if (EOC==1'b1)  next_state=s3 ;
         else  next_state = s2 ;    end
     s3 : begin  SOUT=s3 ;  next_state = s4 ;   end
     s4 : begin  SOUT=s4 ;  next_state = s0 ;   end
     default : begin  next_state=s0 ;  SOUT=s0; end
   endcase
   end
 always @ (posedge CLK or  posedge RST)  begin  //时序过程
    if (RST) cs <= s0 ;  else  cs<=next_state ; end
 always @ (posedge SOUT[1] )                  //寄存器过程
   if (SOUT[1])   REGL <= D ;
 assign ADDA =0 ;           assign Q = REGL ;
 assign LOCK_T = SOUT[1] ; assign     OE = SOUT[2] ;
 assign    ALE = SOUT[3] ; assign   START = SOUT[4] ;
endmodule
```

6.4.2 用宏定义语句定义状态编码

【例 6-9】

```
`define s0  5'B00000    //定义状态码
`define s1  5'B11000
`define s2  5'B00001
`define s3  5'B00100
`define s4  5'B00110
module ADC0809 (D, CLK, EOC, RST, ALE, START, OE, ADDA, Q,LOCK_T);
   input[7:0] D;    input CLK,RST,EOC;
   output START,OE , ALE, ADDA,LOCK_T ;   output[7:0] Q;
   reg[4:0] cs ;    reg[4:0] SOUT, next_state ;
reg[7:0] REGL;  reg LOCK;
always @ (cs or EOC)  begin
    case (cs)
      `s0 : begin  next_state<=`s1 ;  SOUT=`s0 ;   end
      `s1 : begin  next_state<=`s2 ;  SOUT=`s1 ;   end
      `s2 : begin  SOUT=`s2 ;
             if (EOC==1'b1)  next_state=`s3 ;
          else  next_state = `s2 ;    end
      `s3 : begin  SOUT=`s3 ;  next_state = `s4 ;   end
      `s4 : begin  SOUT=`s4 ;  next_state = `s0 ;   end
     default : begin  next_state=`s0 ;  SOUT=`s0; end
    endcase    end
  always @ (posedge CLK or  posedge RST)  begin  //时序过程
     if (RST) cs <= `s0 ;  else  cs<=next_state ; end
  always @ (posedge SOUT[1] )                  //寄存器过程
    if (SOUT[1])   REGL <= D ;
  assign ADDA =0 ;   assign Q = REGL ;
  assign LOCK_T = SOUT[1] ;
  assign  START = SOUT[4] ;
  assign    ALE = SOUT[3] ;
  assign     OE = SOUT[2] ;
endmodule
```

6.4 不同编码类型的状态机

6.4.2 用宏定义语句定义状态编码

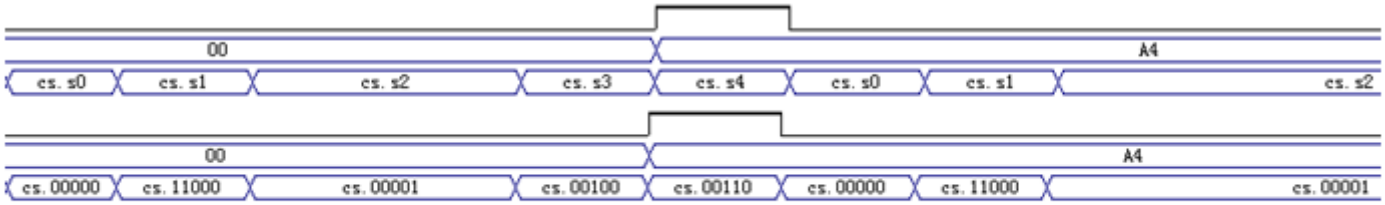


图 6-14　用`define（下）或 paramete（上）来定义状态元素的状态机仿真波形

6.4 不同编码类型的状态机

6.4.3 宏定义命令语句

```
`define  宏名（标志符）  宏内容（字符串）
```

```
`define  s   A+B+C+D
```

```
assign   DOUT = A+B+C+D+E;
```

6.4 不同编码类型的状态机

6.4.4 顺序编码

表 6-2 编码方式

状　　态	顺 序 编 码	一位热码编码	约翰逊码编码
States	Sequential-Encoded	One-Hot-Encoded	Johnson-Encoded
State0	000	100000	0000
State1	001	010000	1000
State2	010	001000	1100
State3	011	000100	1110
State4	100	000010	1111
State5	101	000001	0111

6.4 不同编码类型的状态机

6.4.5 一位热码编码

表 6-2 编码方式

状　　态	顺 序 编 码	一位热码编码	约翰逊码编码
States	Sequential-Encoded	One-Hot-Encoded	Johnson-Encoded
State0	000	100000	0000
State1	001	010000	1000
State2	010	001000	1100
State3	011	000100	1110
State4	100	000010	1111
State5	101	000001	0111

6.4 不同编码类型的状态机

6.4.6 状态编码设置

1. 用户自定义方式

2. 用属性定义语句设置

```
(* syn_encoding = "one-hot" *)
```

【例 6-10】

```
module SCHK  (CLK, DIN, RST,SOUT);
input  CLK, DIN, RST;    output SOUT;
parameter s0=0, s1=1, s2=2, s3=3, s4=4, s5=5, s6=6, s7=7,s8=8 ;
 (*syn_encoding="one-hot"*) reg[8:0] ST ;
reg SOUT;
always @(posedge CLK)   begin
 . . . . . .
```

6.4 不同编码类型的状态机

2. 用属性定义语句设置

表 6-3 **编码方式属性定义及资源耗用参考**

编码方式	编码方式属性定义	逻辑宏单元数 LCs	触发器数 REGs
一位热码	(* syn_encoding = "one-hot" *)	13	10
用户自定义码	(* syn_encoding = "user" *)	12	5
格雷码	(* syn_encoding = "gray" *)	8	5
顺序码	(* syn_encoding = "sequential" *)	10	5
约翰逊码	(* syn_encoding = "johnson" *)	23	6
默认编码	(* syn_encoding = "default" *)	13	10
最简码	(* syn_encoding = "compact" *)	9	5
安全一位热码	(* syn_encoding = "safe, one-hot" *)	21	10

6.4 不同编码类型的状态机

3. 直接设置方法

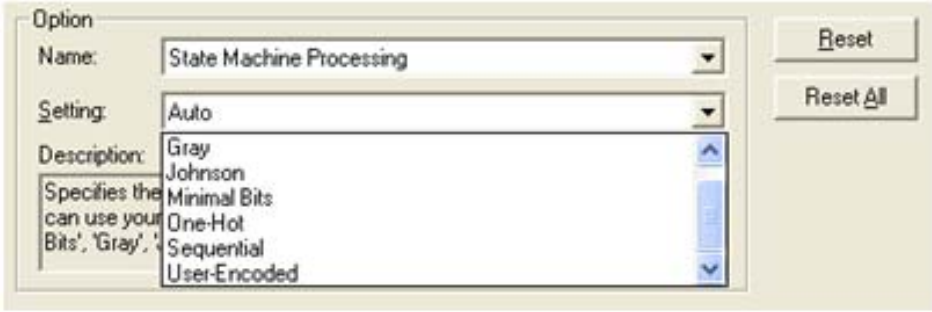


图 6-15 选择恰当的编码形式

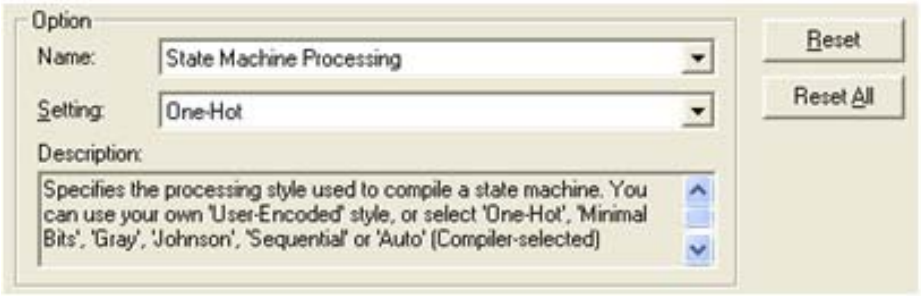


图 6-16 选择了一位热码编码形式

6.5 状态机容错技术

6.5.1 状态导引法

```
parameter s0=0,s1=1,s2=2,s3=3,s4=4, s5=5, s6=6,s7=7;
    ...
 S5 :   next_state = s0 ;
 S6 :   next_state = s0 ;
 S7 :   next_state = s0 ;
 default : begin  next_state=s0 ;
```

表 6-4 剩余状态

状 态	顺序编码
s0	000
s1	001
s2	010
s3	011
s4	100
s5	101
s6	110
s7	111

6.5 状态机容错技术

6.5.2 状态编码监测法

6.5.3 借助EDA工具自动生成安全状态机

图 6-17 选择安全状态机设计

6.6 硬件数字技术排除毛刺

6.6.1 延时方式去毛刺

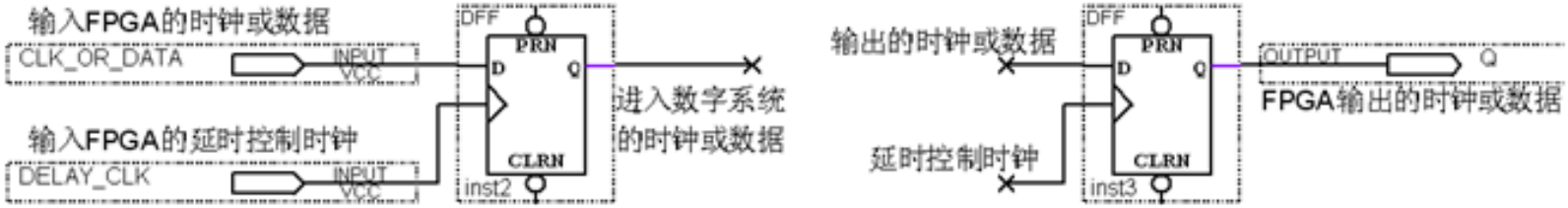


图 6-18 单触发器输入(左图)或输出(右图)延时电路

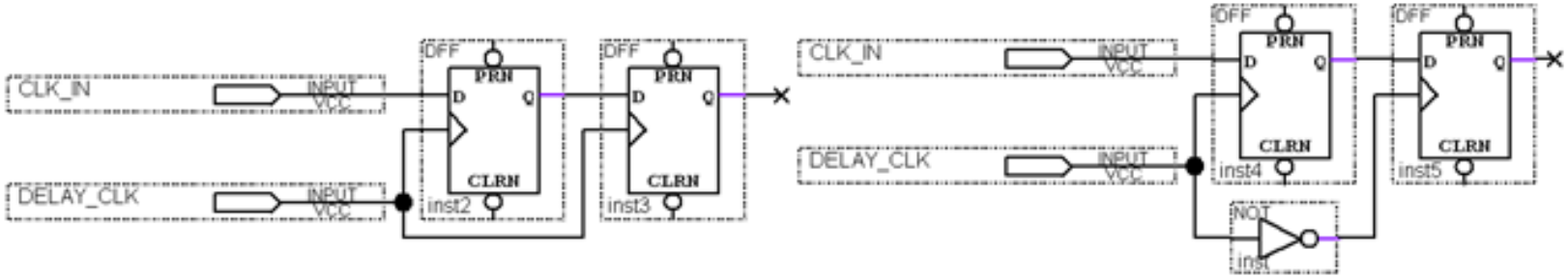


图 6-19 双触发器延时电路

6.6 硬件数字技术排除毛刺

6.6.1 延时方式去毛刺

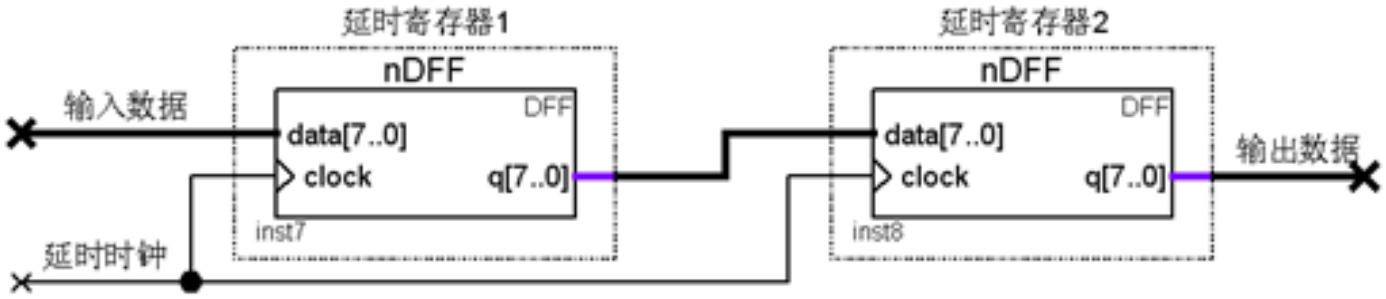


图 6-20 双寄存器数据延时电路

6.6 硬件数字技术排除毛刺

6.6.2 逻辑方式去毛刺

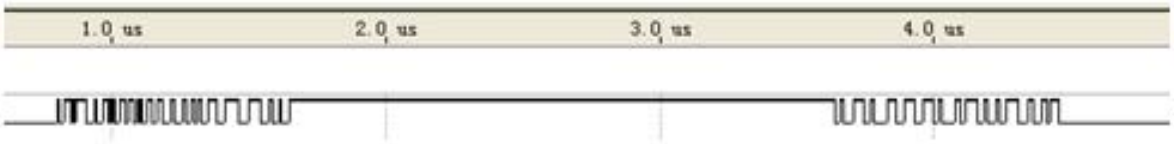


图 6-21 在信号上升与下降都含随机干扰抖动信号的时钟信号

6.6 硬件数字技术排除毛刺

6.6.2 逻辑方式去毛刺

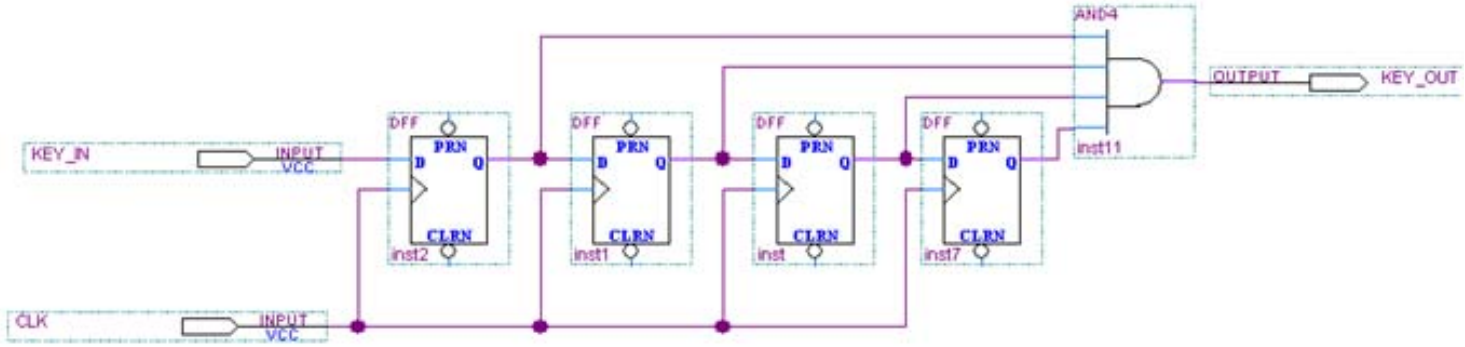


图 6-22 消抖动电路

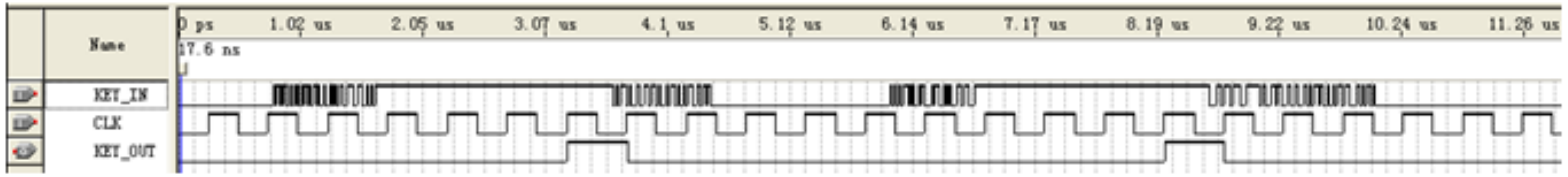


图 6-23 消抖动电路仿真波形

6.6 硬件数字技术排除毛刺

6.6.3 定时方式去毛刺

【例 6-11】

```
module ERZP  (CLK, KIN,KOUT);
   input   CLK, KIN;                    //工作时钟和输入信号
   output  KOUT;     reg KOUT;
   reg [3:0] KH,KL;                     //定义对高电平和低电平脉宽计数之寄存器。
   always @(posedge CLK)  begin
      if (!KIN)  KL<=KL+1 ;             //对键输入的低电平脉宽计数
        else  KL<=4'b0000;    end       //若出现高电平，则计数器清 0
   always @(posedge CLK)  begin
       if (KIN)  KH<= KH+1;             //同时对键输入的高电平脉宽计数
         else KH<=4'b0000;   end        //若出现高电平，则计数器清 0
   always @(posedge CLK)  begin
       if (KH > 4'b1100) KOUT<=1'B1;//对高电平脉宽计数一旦大于 12，则输出 1
         else if (KL > 4'b0111)
    KOUT<=1'B0;          end           //对低电平脉宽计数若大于 7，则输出 0
endmodule
```

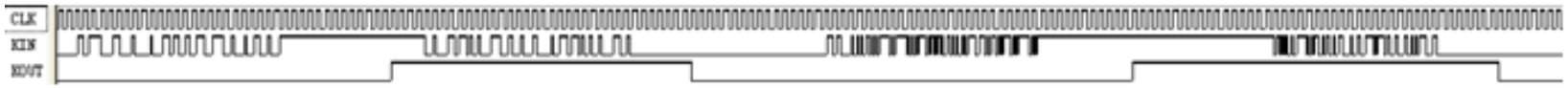


图 6-24 例 6-11 消抖动电路仿真波形

习　题

6-1

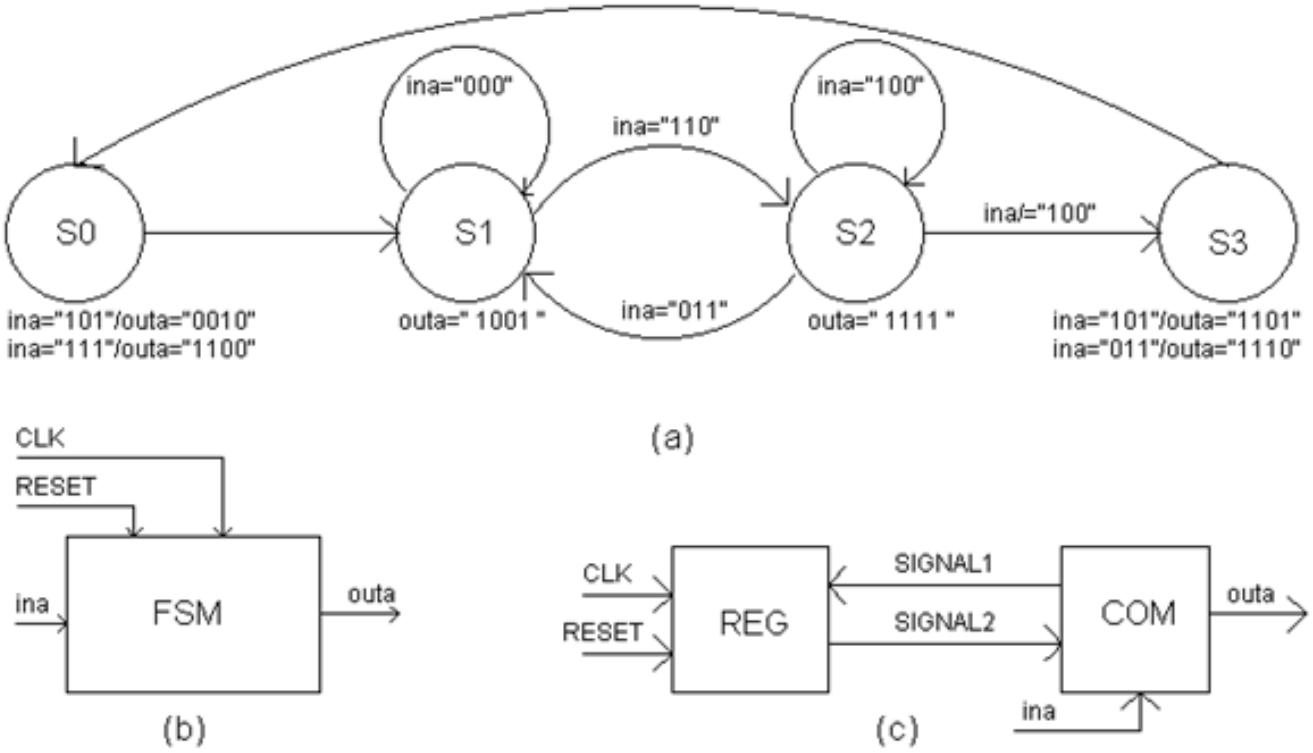


图 6-25　习题 6-1 状态图

实训项目

6-1 序列检测器设计

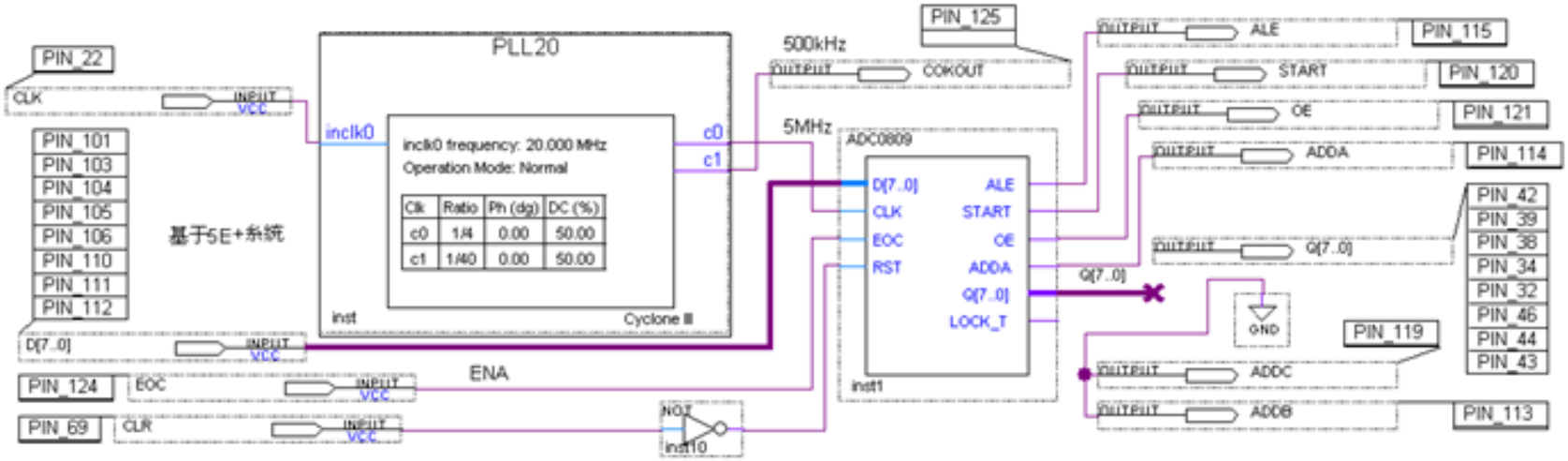


图 6-26　ADC0809 采样控制实验电路与引脚锁定指示

实训项目

6-3 五功能智能逻辑笔设计

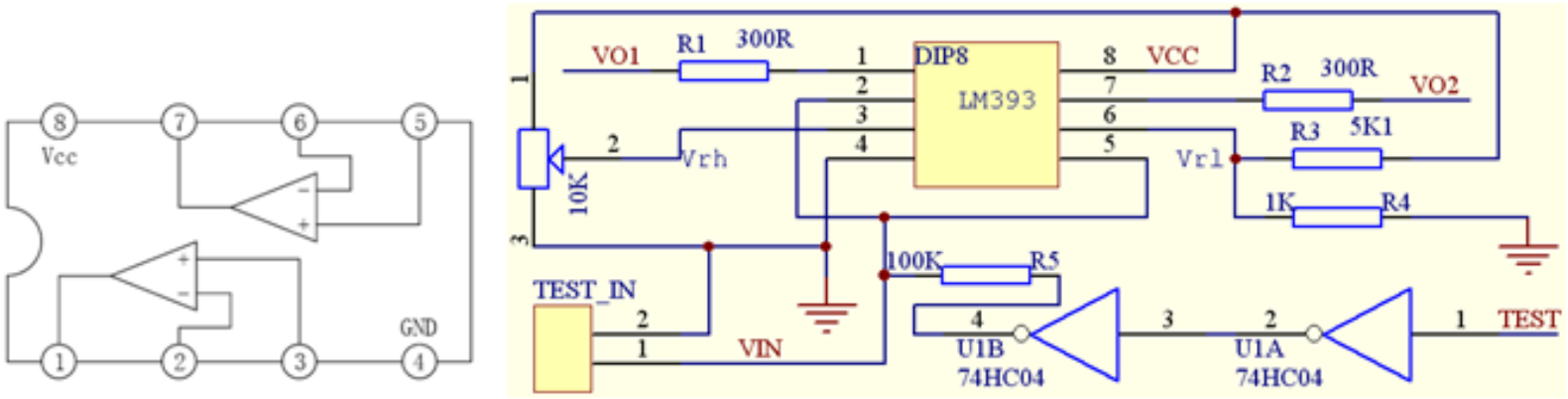


图 6-27 五功能智能逻辑笔电平信号采样电路，左图是 LM393 引脚图

实训项目

6-4 点阵型与字符型液晶显示器驱动控制电路设计

6-5 硬件消抖动电路设计

6-6 数字彩色液晶显示控制电路设计

实训项目

6-7 PS2键盘控制模型电子琴电路设计

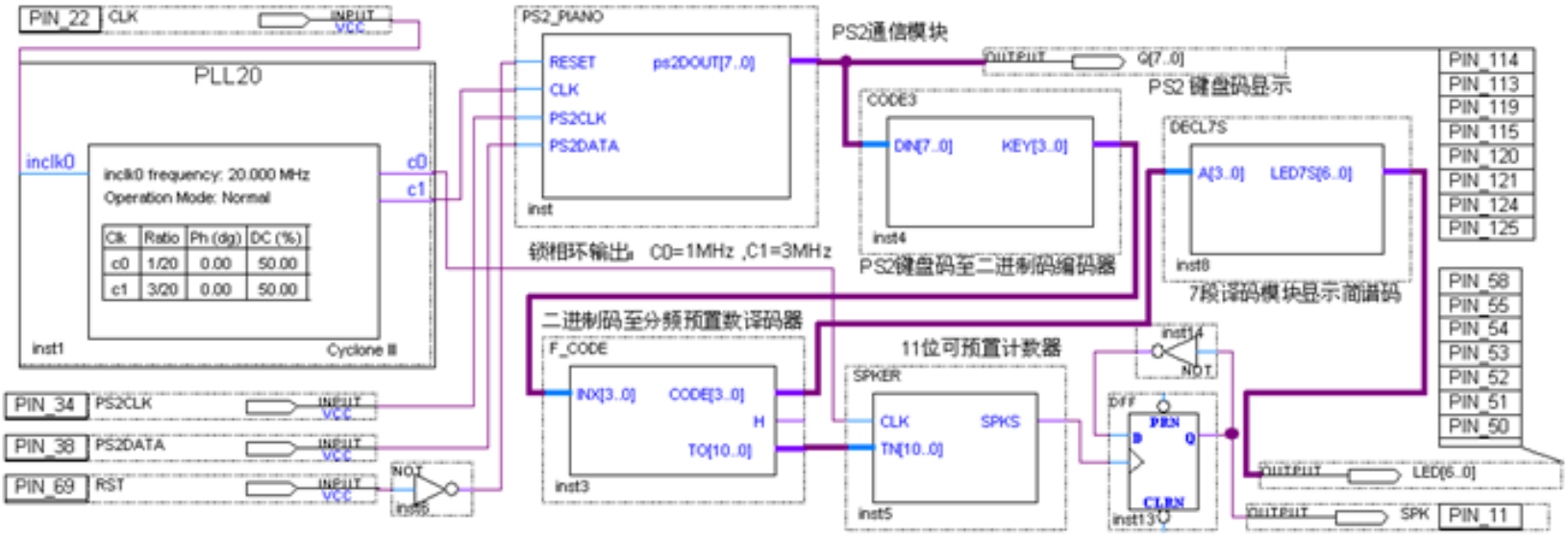


图 6-28　PS2 键盘控制模型电子琴电路顶层设计

实训项目

6-7 PS2键盘控制模型电子琴电路设计

表 6-5 PS2 键盘键控与输出码对照表

Key	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Data	1C	32	21	23	24	2B	34	33	43	3B	42	4B	3A	31	44
Key	P	Q	R	S	T	U	V	W	X	Y	Z	0	1	2	3
Data	4D	15	2D	1B	2C	3C	2A	1D	22	35	1A	45	16	1E	26
Key	4	5	6	7	8	9	`	-	=	\	]	;	’	,	.
Data	25	2E	36	3D	3E	46	0E	4E	55	5D	5B	4C	52	41	49
Key	/	[	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	KP0
Data	4A	54	05	06	04	0C	03	0B	83	0A	01	09	78	07	70
Key	KP1	KP2	KP3	KP4	KP5	KP6	KP7	KP8	KP9	KP.	KP-	KP+	KP/	KP*	END
Data	69	72	7A	6B	73	74	6C	75	7D	71	7B	79	4A	7C	69
Key	BKSP	SPACE	TAB	CAPS	L SHFT	L CTRL	L CUI	L ALT	R SHFT	R CTRL	R CUI				
Data	66	29	0D	58	12	14	1F	11	59	14	27				
Key	R ALT	APPS	ENTER	ESC	INSERT	HOME	PG UP	DELETE	PG DN	NUM					
Data	11	2F	5A	76	70	6C	7D	71	7A	77					
Key	U ARROW	L ARROW	D ARROW	R ARROW	KP EN	SCROLL	PRNT SCRN	PAUSE							
Data	75	6B	72	74	5A	7E	12 7C	14							

实训项目

6-8 状态机控制串/并转换8数码静态显示

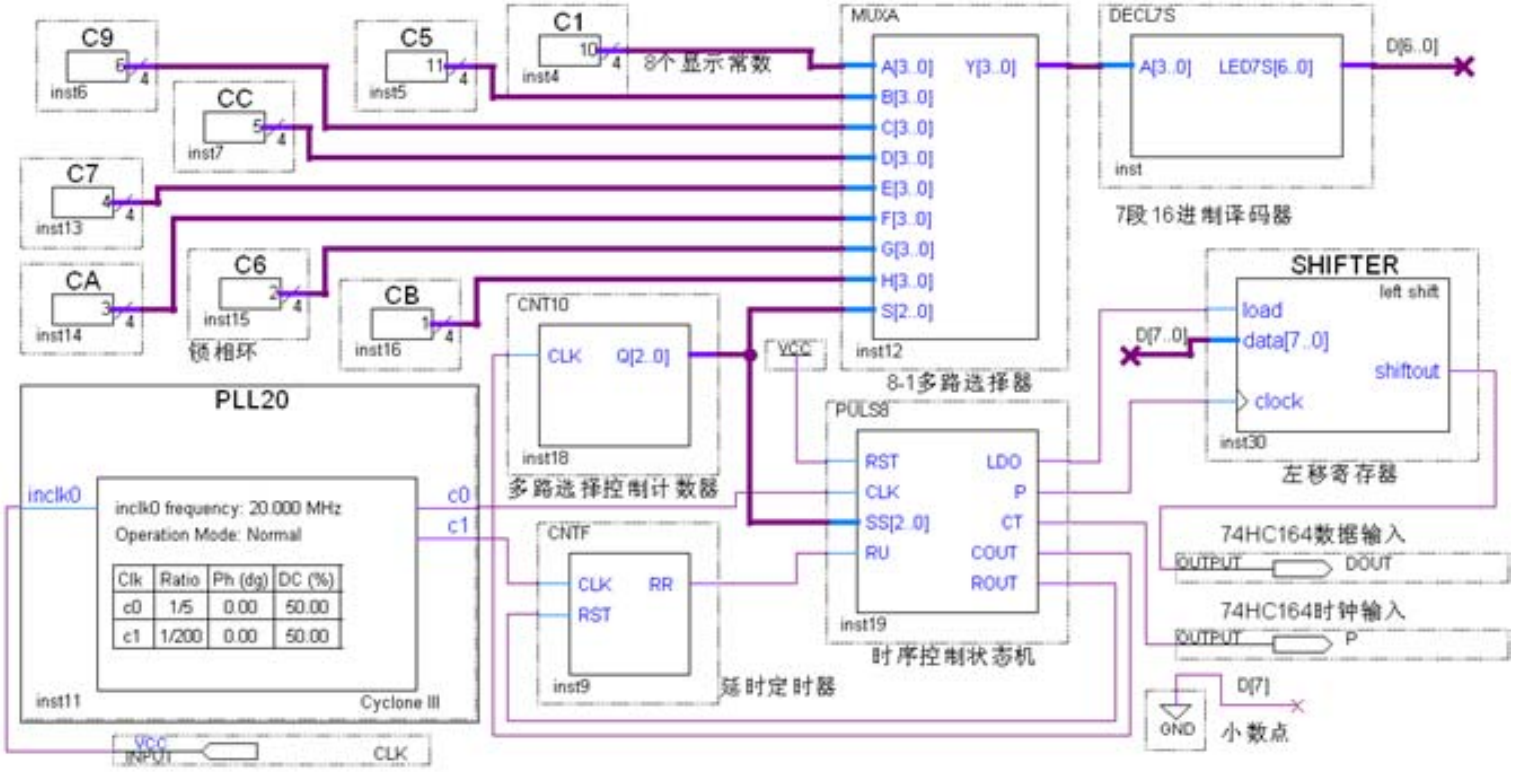


图 6-29　8位串行静态显示状态机控制电路