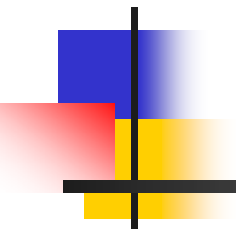


第8章

有限状态机设计技术

8.1 Verilog状态机的一般形式

8.1.1 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。

8.1 Verilog状态机的一般形式

8.1.2 状态机的一般结构

1. 说明部分

```
TYPE FSM_ST IS (s0,s1,s2,s3,s4);
SIGNAL current_state, next_state : FSM_ST;

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

typedef enum {s0, s1, s2, s3, s4} type_user ;
type_user     current_state, next_state ;
```

8.1 Verilog状态机的一般形式

8.1.2 状态机的一般结构

2. 主控时序过程

3. 主控组合过程

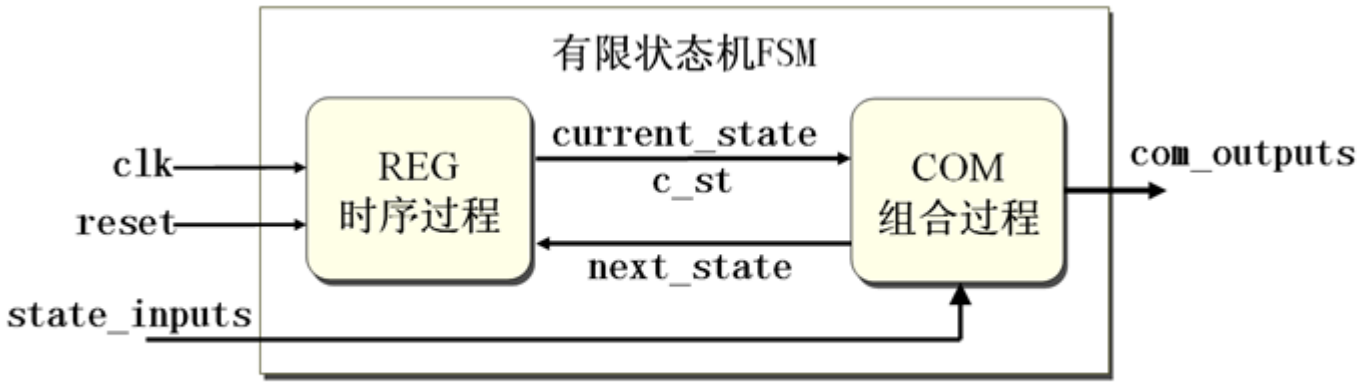


图 8-1　状态机一般结构示图

8.1 Verilog状态机的一般形式

8.1.2 状态机的一般结构

4. 辅助过程

【例 8-1】

```
module FSM_EXP (clk, reset, state_inputs, comb_outputs);
input clk;                                          // 状态机工作时钟
input reset;                                        // 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
```

接下页

接上页

```
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
```

8.1 Verilog状态机的一般形式

8.1.2 状态机的一般结构

4. 辅助过程

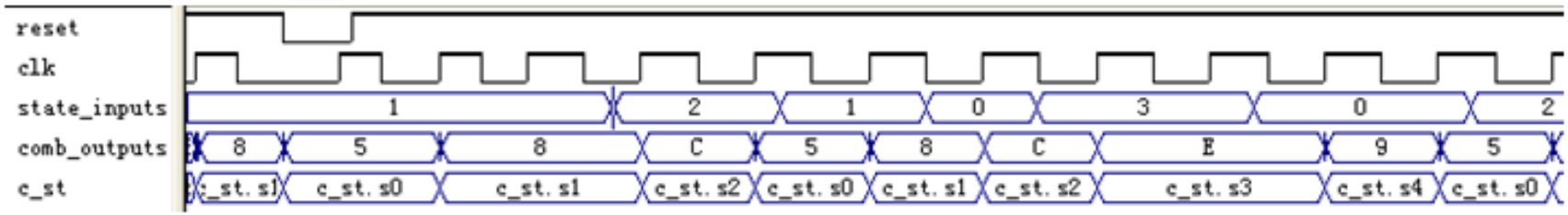


图 8-2　例 8-1 状态机的工作时序

8.1 Verilog状态机的一般形式

8.1.3 初始控制与表述

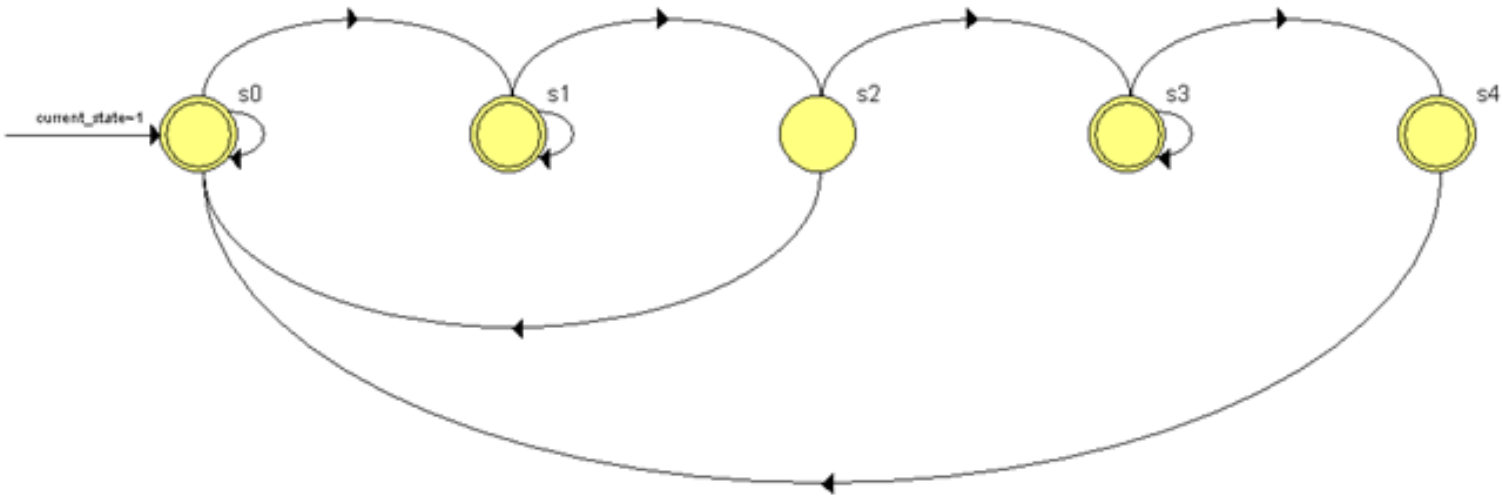


图 8-4　例 8-1 的状态图

8.2 Moore型状态机及其设计

8.2.1 多过程结构型状态机

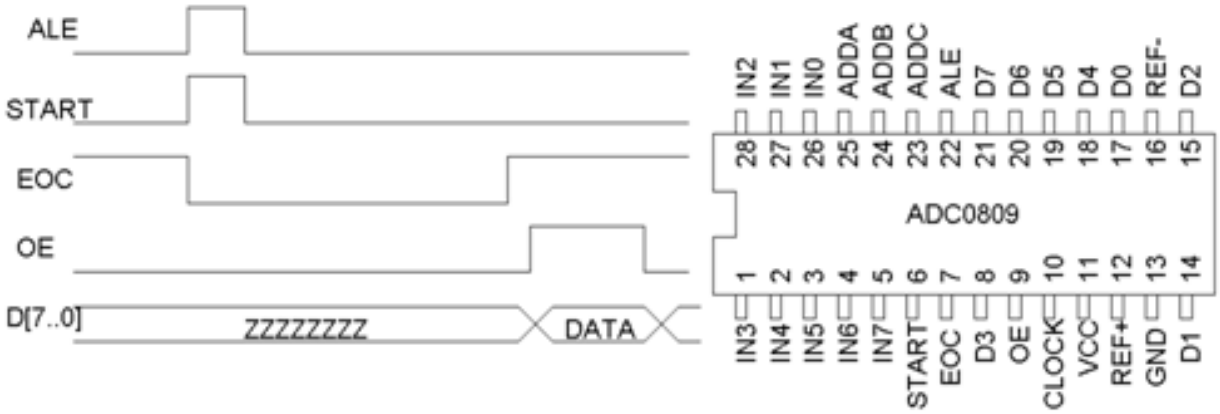


图 8-5　ADC0809 工作时序和芯片引脚图

8.2 Moore型状态机及其设计

8.2.1 多过程结构型状态机

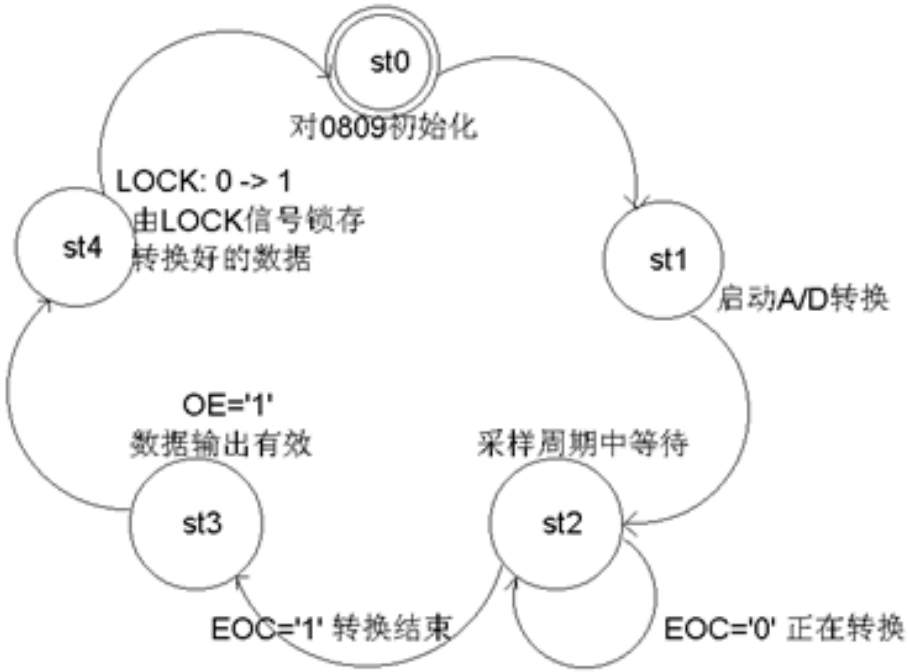


图 8-6 控制 ADC0809 采样状态图

8.2 Moore型状态机及其设计

8.2.1 多过程结构型状态机

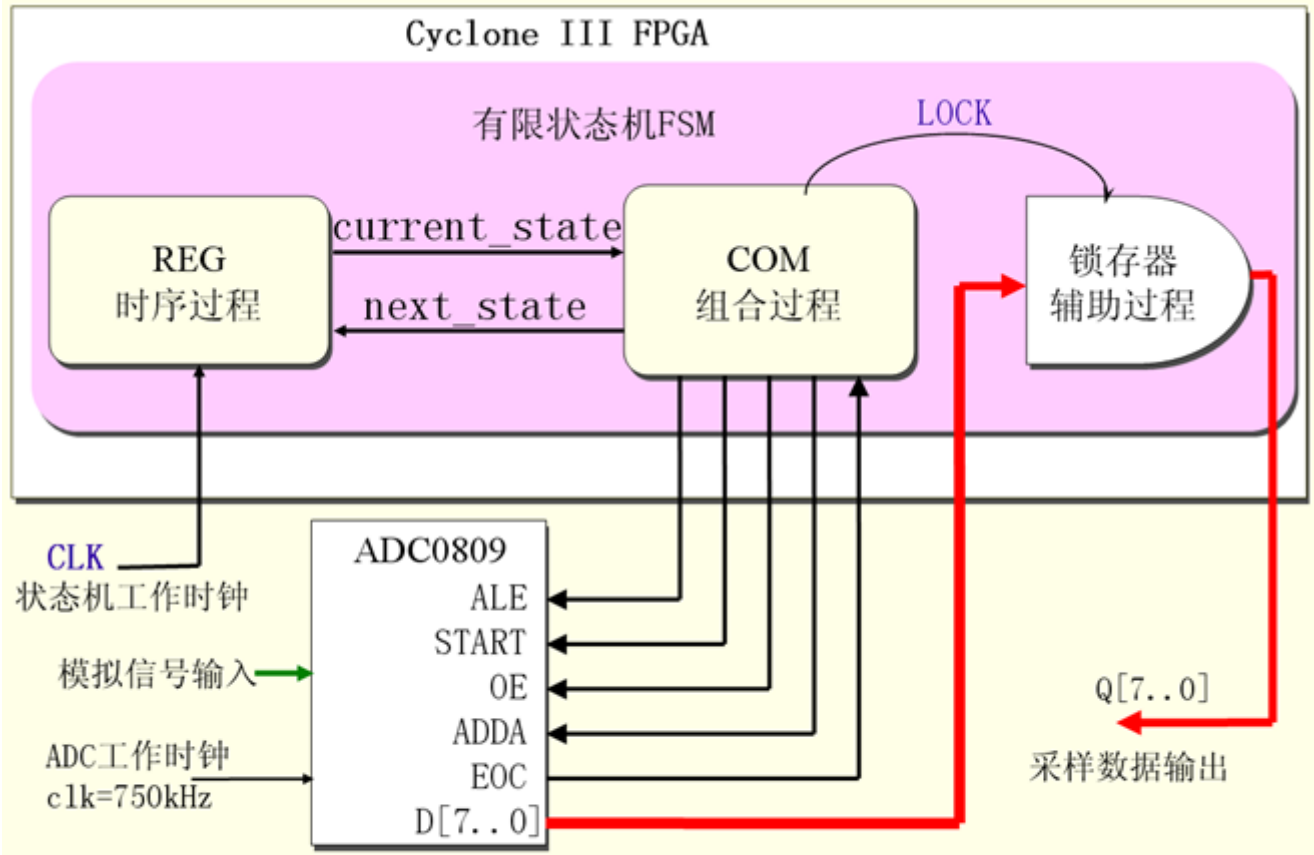


图 8-7 采样状态机结构框图

【例 8-2】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;                 //状态机工作时钟,和系统复位控制
  input EOC;                     //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 ;
```

接下页

接上页

```
                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
```

8.2 Moore型状态机及其设计

8.2.1 多过程结构型状态机

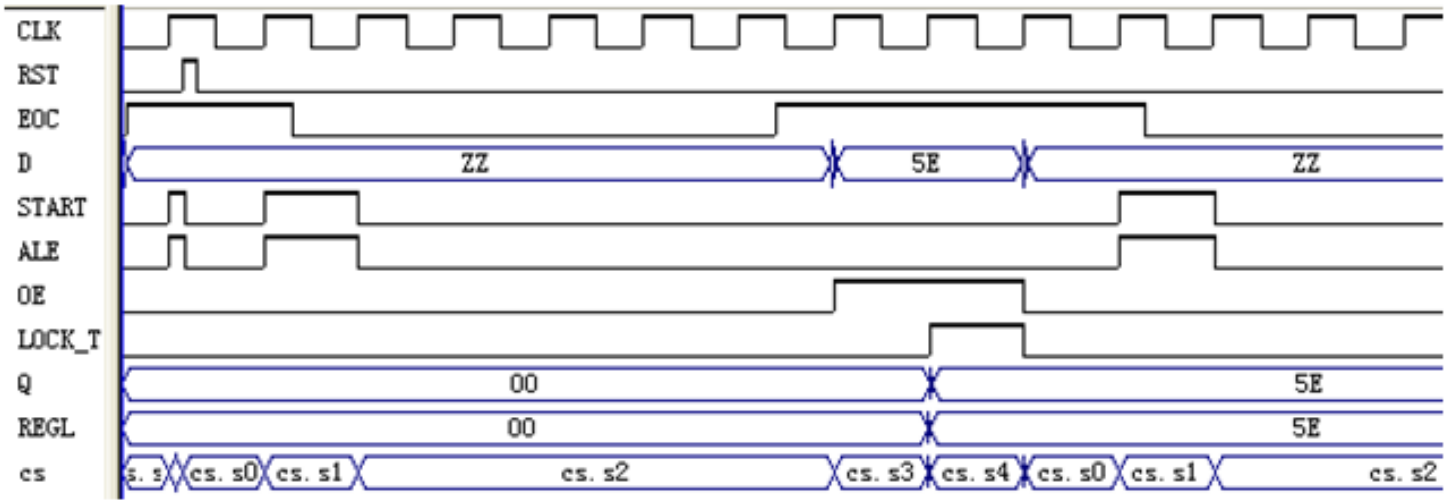


图 8-8　ADC0809 采样状态机工作时序

【例 8-3】

```
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
```

8.2.2 序列检测器及其状态机设计

【例 8-4】

```
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
```

8.2 Moore型状态机及其设计

8.2.2 序列检测器及其状态机设计

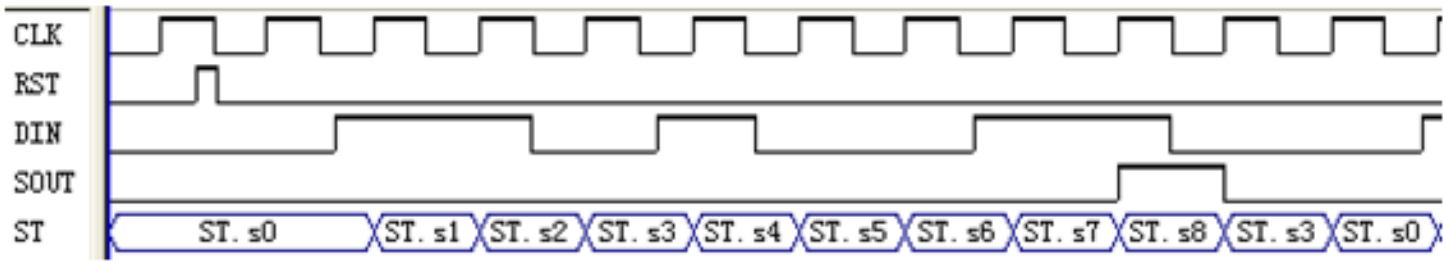


图 8-9　例 8-4 之序列检测器时序仿真波形

8.3 Mealy型状态机设计

【例 8-5】

```
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
```

接下页

8.3 Mealy型状态机设计

接上页

```
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
```

8.3 Mealy型状态机设计

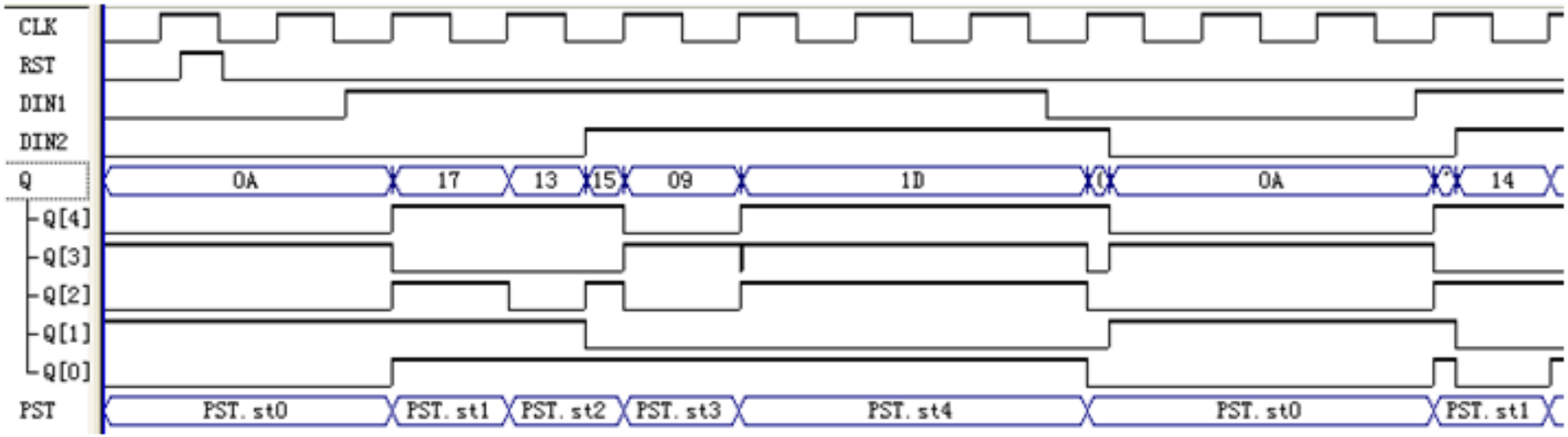


图 8-10　例 8-5 之双过程 Mealy 机仿真波形

8.3 Mealy型状态机设计

【例 8-6】

```
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
```

接下页

8.3 Mealy型状态机设计

接上页

```
            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
```

8.3 Mealy型状态机设计

```
always @ (posedge CLK )   begin  : COM
```

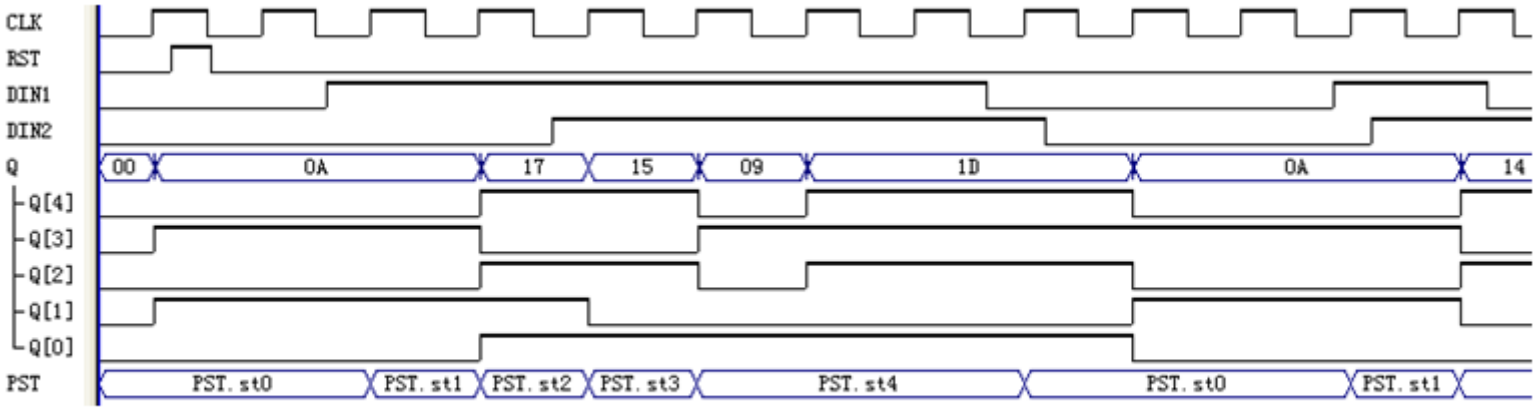


图 8-11　例 8-6 之单过程 Mealy 机仿真波形

8.3 Mealy型状态机设计

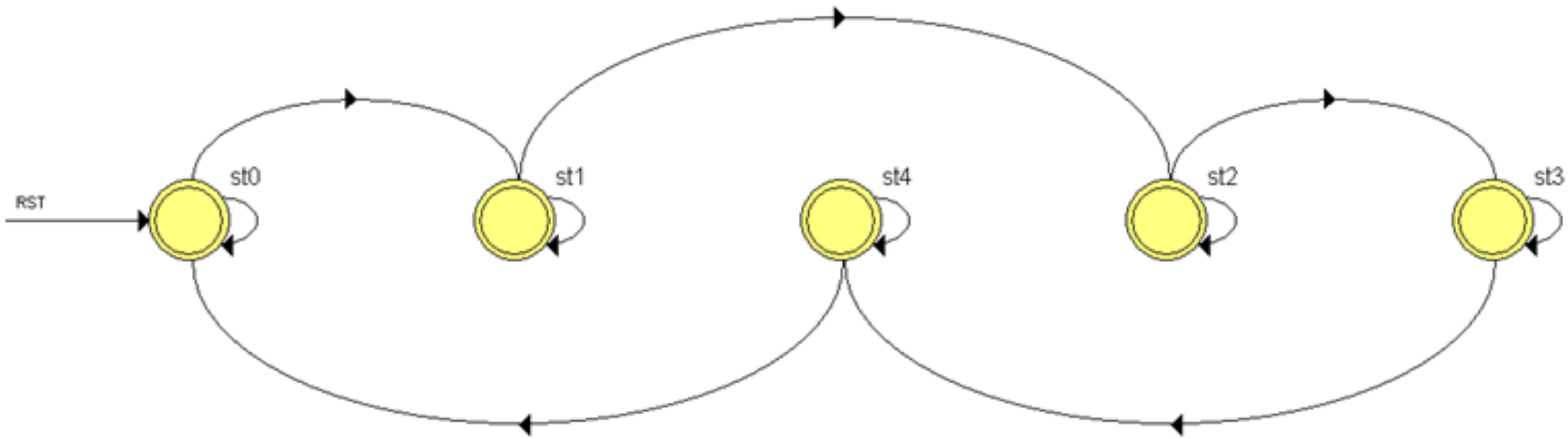


图 8-12　例 8-6，8-5 的状态图

【例 8-7】

```
module SCHK  (CLK, DIN, RST,SOUT);//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
```

8.3 Mealy型状态机设计

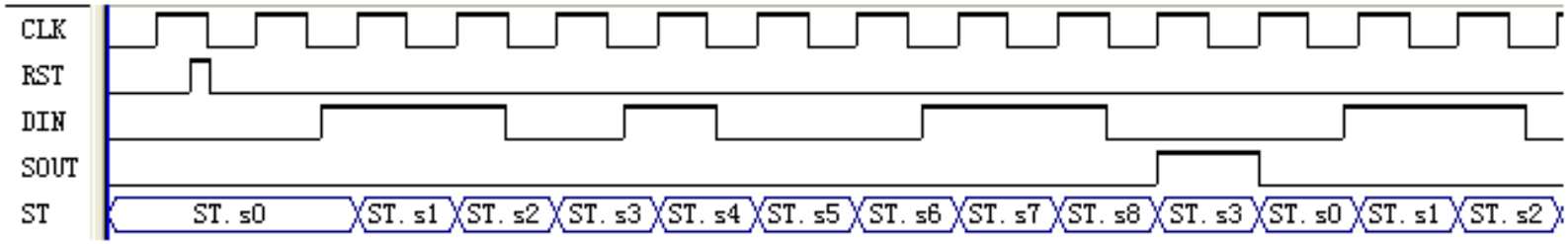


图 8-13　例 8-7 之单过程 Mealy 机仿真波形

8.4 SystemVerilog的枚举类型应用

```
module ADC  (CLK, DIN, RST,SOUT);//11010011高位在前。
   input  CLK, DIN, RST;    output SOUT;
   typedef  enum {s0,s1,s2,s3,s4,s5,s6,s7,s8} type_u;
   type_u   ST,NST  ;
   always @(posedge CLK or posedge RST)
```

8.5 状态机图形编辑设计

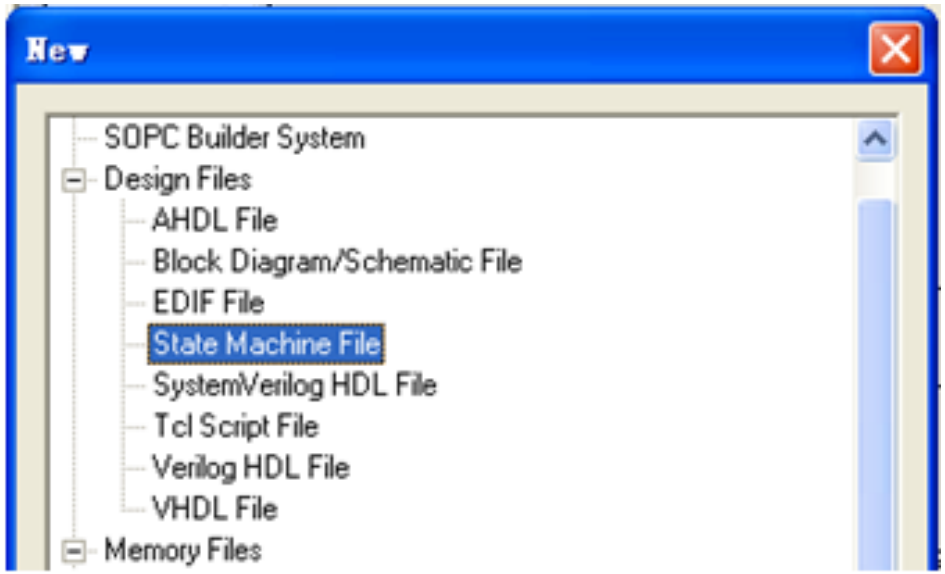


图 8-14 打开状态机图形编辑窗

8.5 状态机图形编辑设计

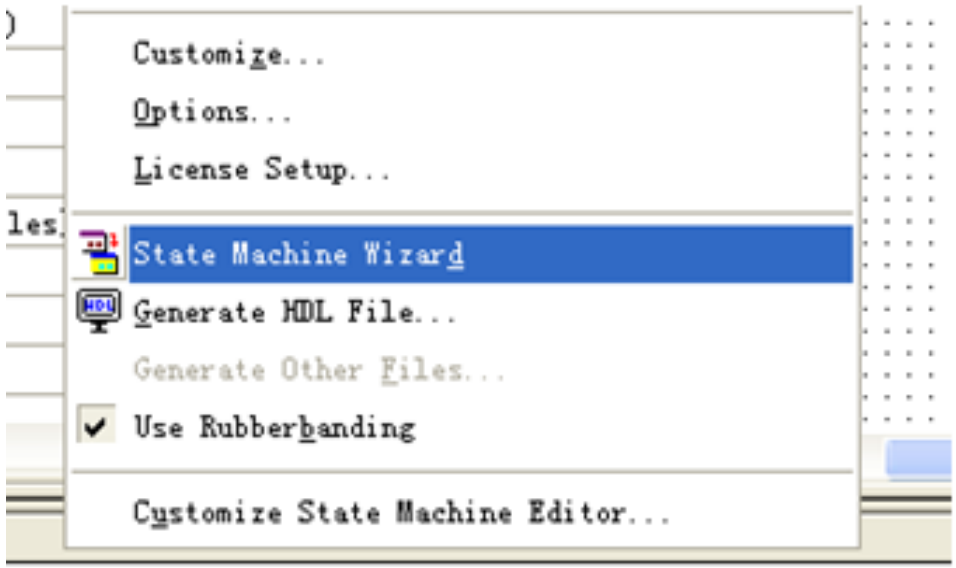


图 8-15 打开状态机编辑器

8.5 状态机图形编辑设计

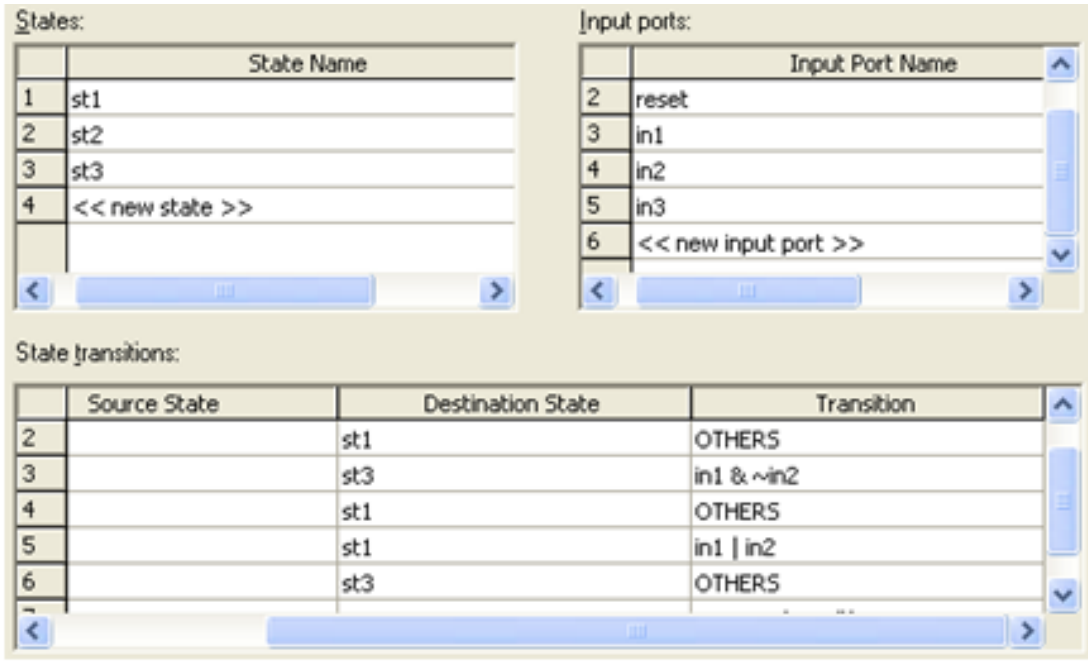


图 8-16 设置状态变量和状态转换条件

8.5 状态机图形编辑设计

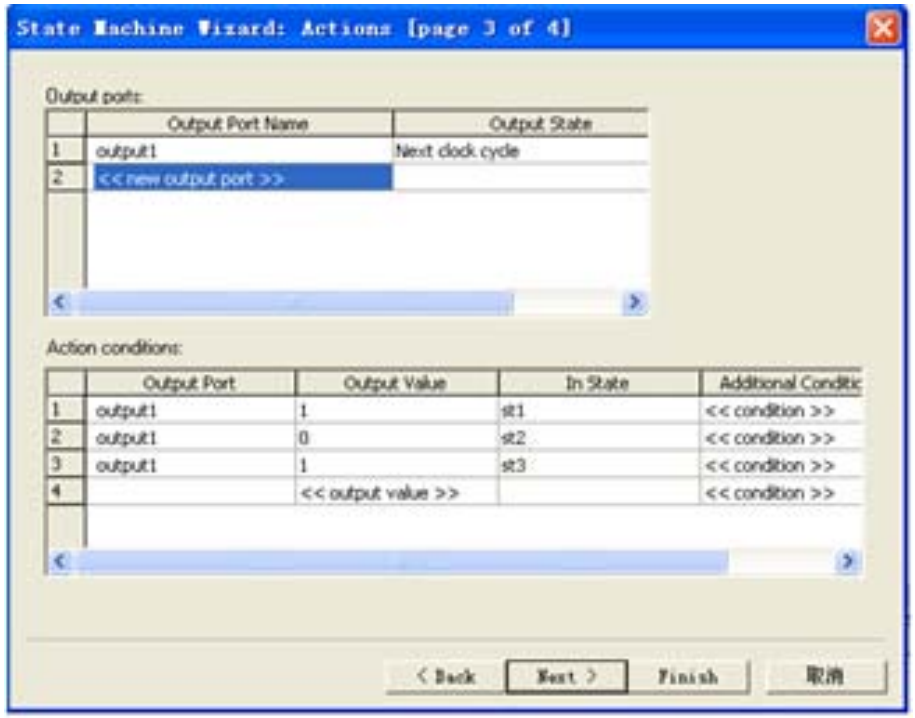


图 8-17 设置状态机输出

8.5 状态机图形编辑设计

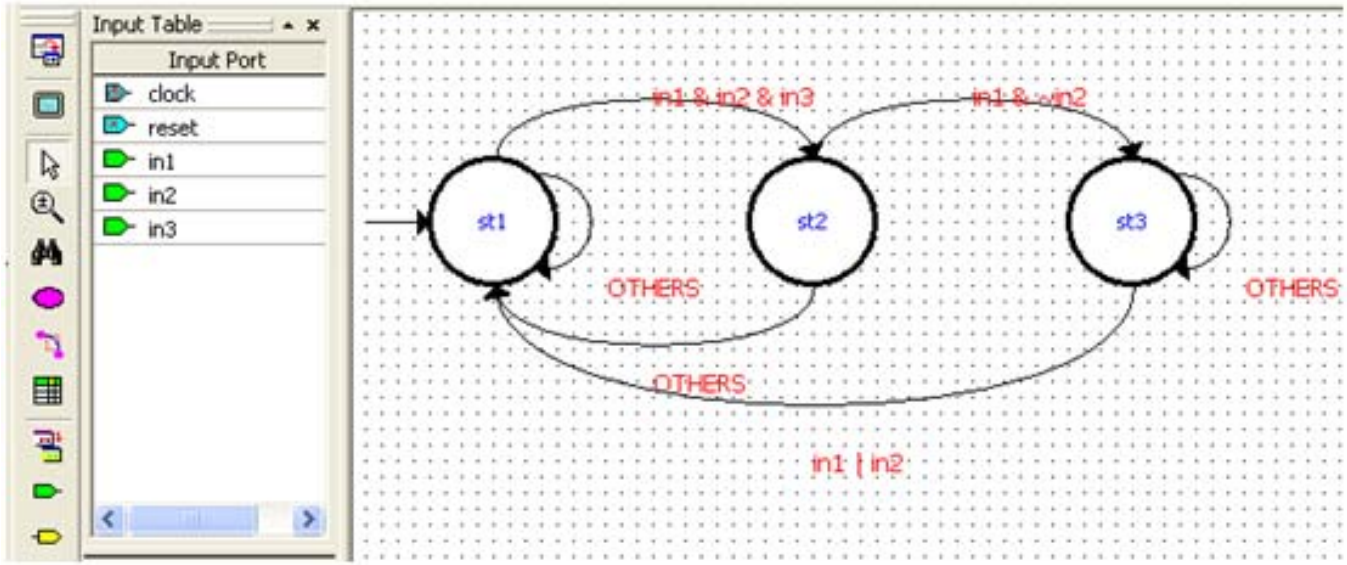


图 8-18 状态机图形编辑器上的状态图

8.5 状态机图形编辑设计

图 8-19 状态机转变成语言

【例 8-8】 Verilog HDL 程序：SM1.v

```
`timescale 1ns/1ns
module SM1 ( clock,reset,in1,in2,in3, output1);
   input clock, reset,in1,in2, in3;
   tri0 reset; tri0 in1; tri0 in2;  tri0 in3;
   output output1;   reg output1, reg_output1;
   reg [2:0] fstate, reg_fstate;
   parameter st1=0, st2=1, st3=2;
   initial
   begin reg_output1 <= 1'b0; end
   always @(posedge clock or posedge reset)  begin
       if (reset)  begin  fstate <= st1; output1 <= 1'b0; end
       else begin  fstate <= reg_fstate; output1 <= reg_output1; end
    end
   always @(fstate or in1 or in2 or in3)  begin
       reg_output1 <= 1'b0;
       case (fstate)
          st1: begin  if (((in1 & in2) & in3)) reg_fstate <= st2;
              else  reg_fstate <= st1;  reg_output1 <= 1'b1;
          end
        . . .
          default: begin reg_output1 <= 1'bx;
              $display ("Reach undefined state");
          end    endcase     end
endmodule
```

8.5 状态机图形编辑设计

【例 8-9】 System Verilog 程序：SM1.sv

```
`timescale 1ns/1ns
module SM1 ( input clock, input reset, input in1,
             input in2, input in3, output output1);
   reg reg_output1;
   enum int unsigned { st1=0, st2=1, st3=2 } fstate, reg_fstate;
   always_ff @(posedge clock or posedge reset)
    begin  if (reset) begin  fstate <= st1; output1 <= 1'b0; end
           else  begin fstate <= reg_fstate;  output1 <= reg_output1;
           end      end
   always_comb  begin
       . . .
end
endmodule
```

8.6 不同编码类型状态机

8.6.1 直接输出型编码

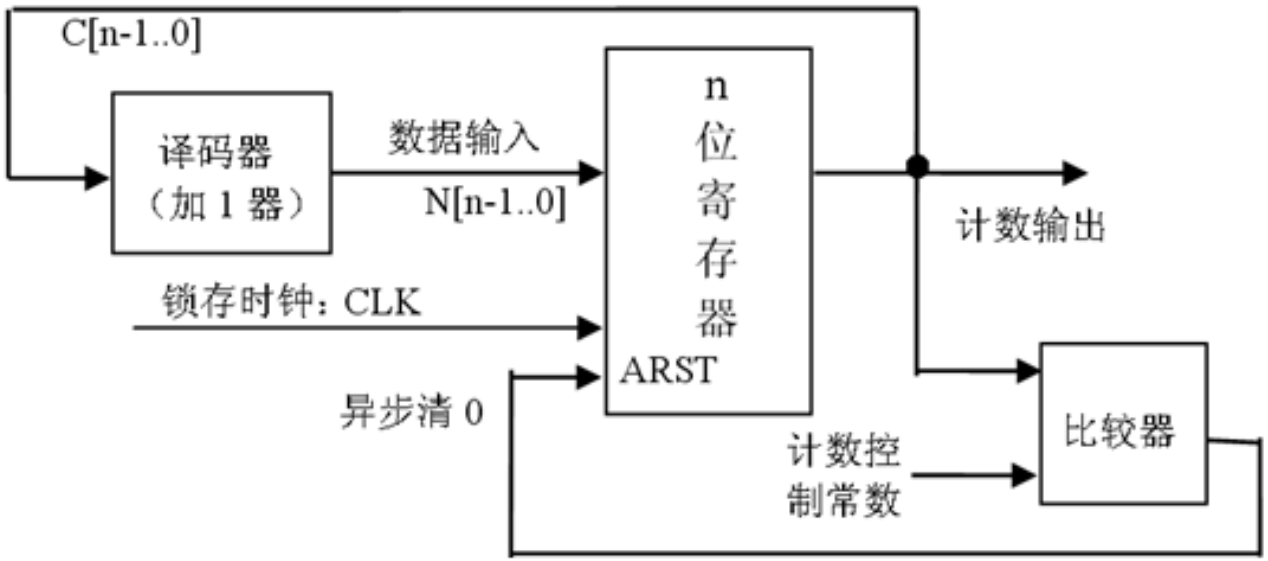


图 8-20 加法计数器一般模型

8.6 不同编码类型状态机

8.6.1 直接输出型编码

表 8-1 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

8.6 不同编码类型状态机

8.6.1 直接输出型编码

【例 8-10】

```
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 ;
```

接下页

8.6 不同编码类型状态机

接上页

```
            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
```

8.6 不同编码类型状态机

8.6.1 直接输出型编码

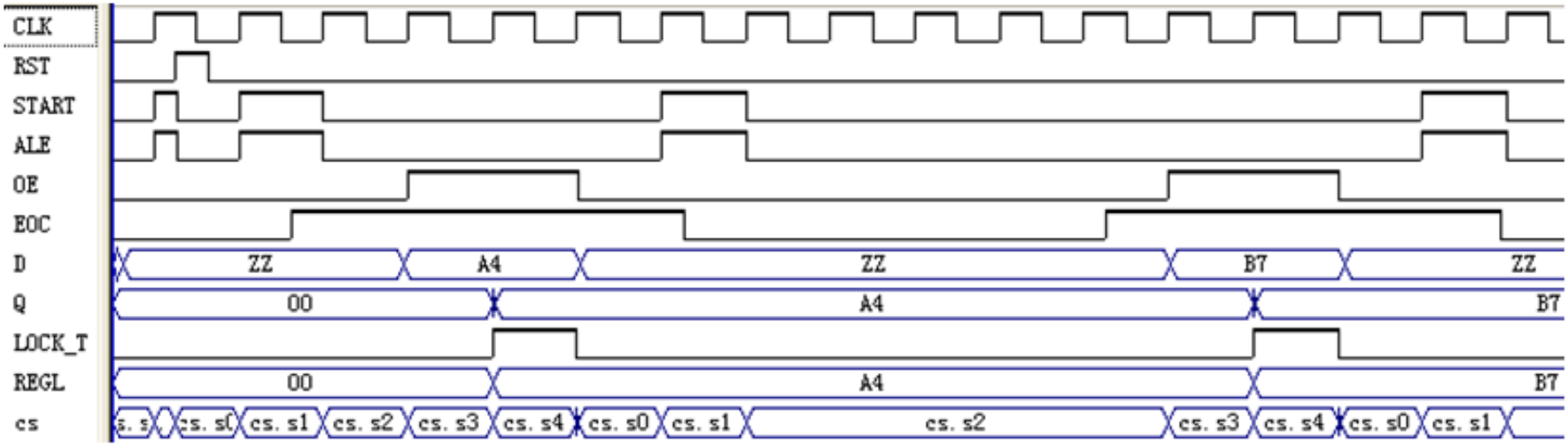


图 8-21 例 8-10 状态机工作时序图

8.6 不同编码类型状态机

8.6.2 用宏定义语句定义状态编码

【例 8-11】

```
`define s0  5'B00000
`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)
```

接下页

8.6 不同编码类型状态机

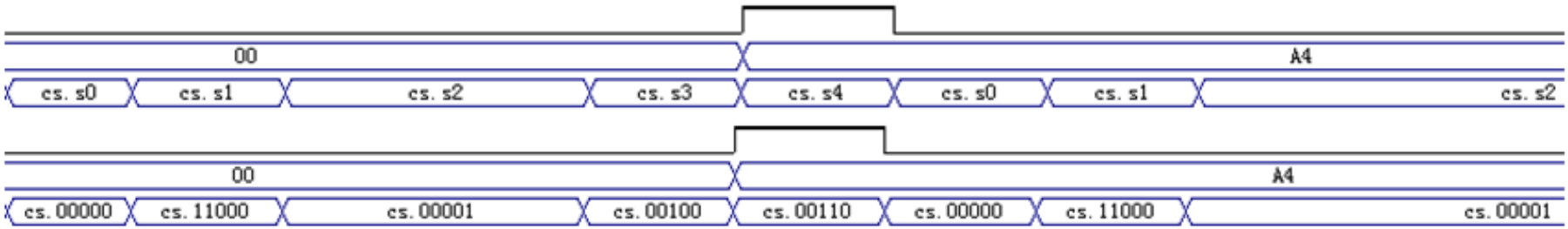
接上页

```
        `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
```

8.6 不同编码类型状态机

8.6.2 用宏定义语句定义状态编码



8.6 不同编码类型状态机

8.6.3 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;
```

8.6 不同编码类型状态机

8.6.4 顺序编码

表 8-2 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

8.6 不同编码类型状态机

8.6.5 一位热码编码

8.6.6 状态编码设置

1. 用户自定义方式

8.6 不同编码类型状态机

2. 用属性定义语句设置

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

【例 8-12】

```
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
 . . . . . .
```

8.6 不同编码类型状态机

2. 用属性定义语句设置

表 8-3　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

8.6 不同编码类型状态机

3. 直接设置方法

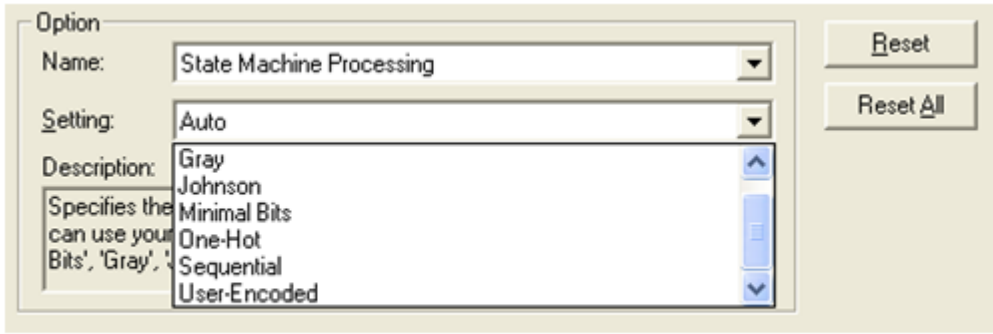


图 8-22 选择恰当的编码形式

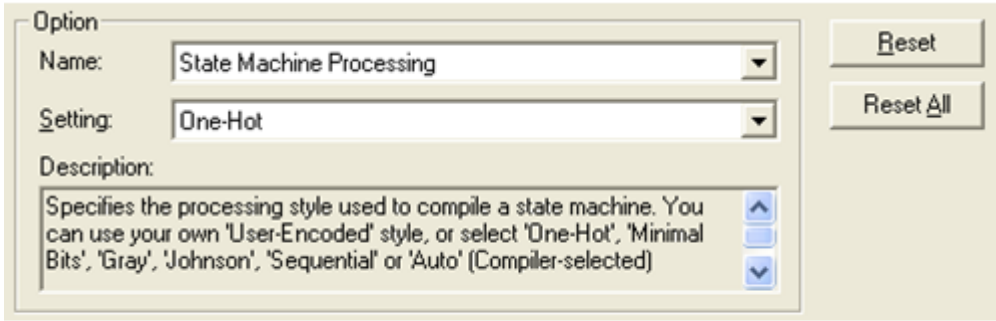


图 8-23 选择了一位热码编码形式

8.7 安全状态机设计

表 8-4　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

8.7 安全状态机设计

8.7.1 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 ;
```

8.7 安全状态机设计

8.7.2 状态编码监测法

8.7.3 借助EDA工具自动生成安全状态机

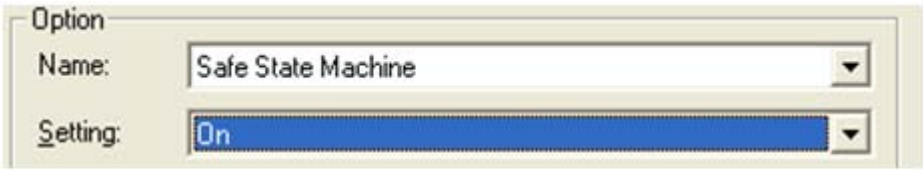


图 8-24 选择安全状态机设计

8.8 硬件数字技术排除毛刺

8.8.1 延时方式去毛刺

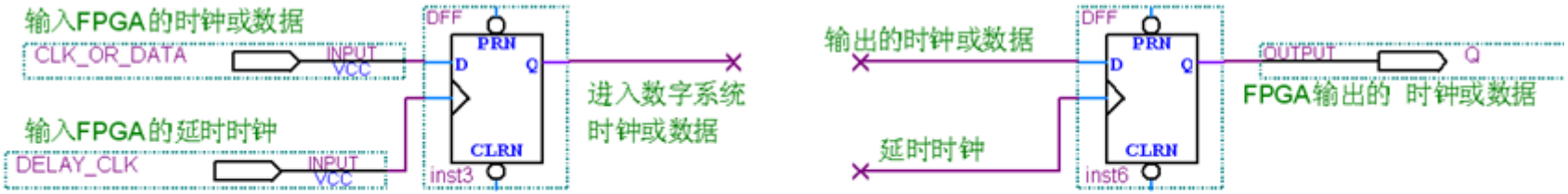


图 8-25 单触发器输入(左图)或输出(右图)延时电路

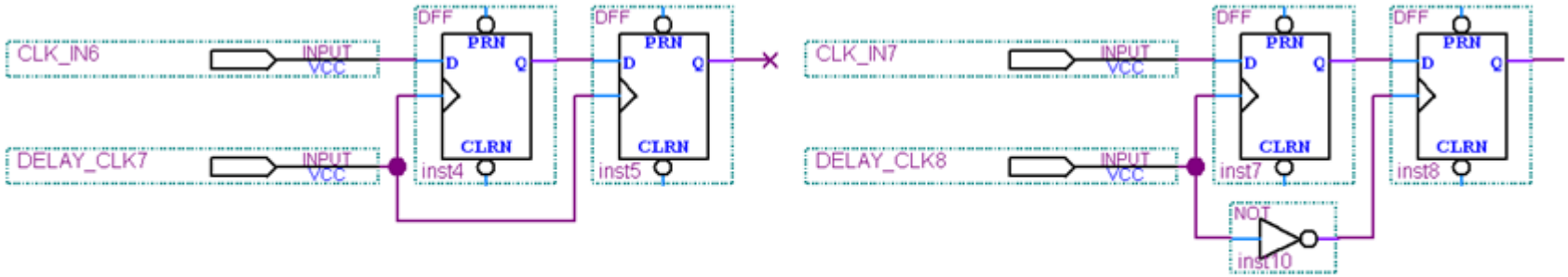


图 8-26 双触发器延时电路

8.8 硬件数字技术排除毛刺

8.8.1 延时方式去毛刺

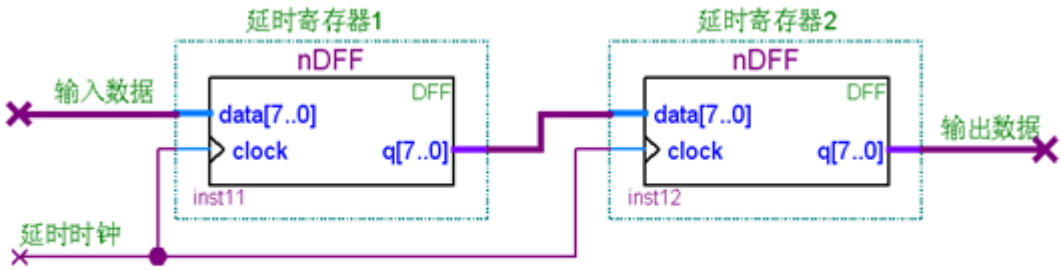


图 8-27 双寄存器数据延时电路

8.8 硬件数字技术排除毛刺

8.8.2 逻辑方式去毛刺

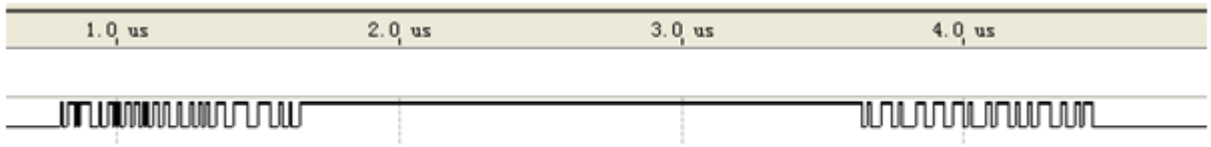


图 8-28 在信号上升与下降都含随机干扰抖动信号的时钟信号

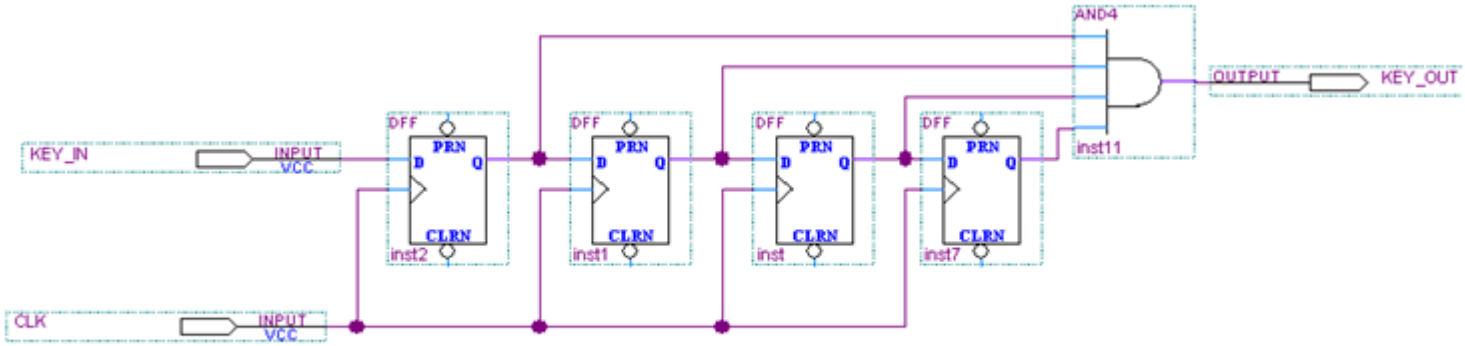


图 8-29 消抖动电路

8.8 硬件数字技术排除毛刺

8.8.2 逻辑方式去毛刺

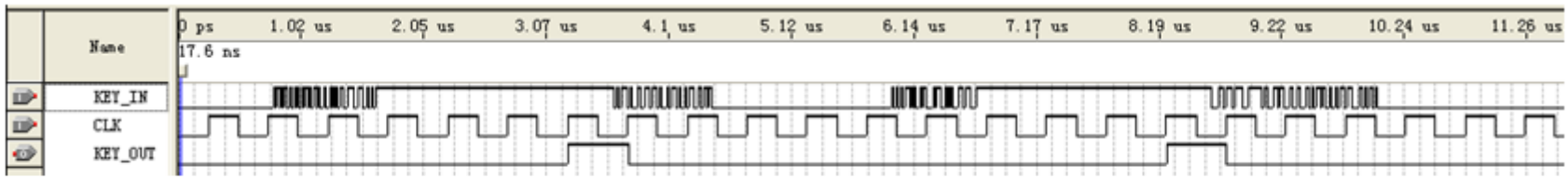


图 8-30 消抖动电路仿真波形

8.8 硬件数字技术排除毛刺

8.8.3 定时方式去毛刺

【例 8-13】

```
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;                         //对低电平脉宽计数若大于 7，则输出 0
    end
endmodule
```

8.8 硬件数字技术排除毛刺

8.8.3 定时方式去毛刺

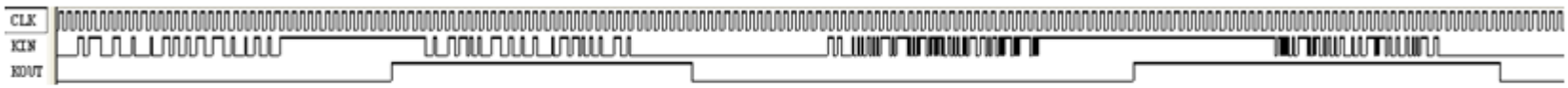


图 8-31 例 8-13 消抖动电路仿真波形

习　题

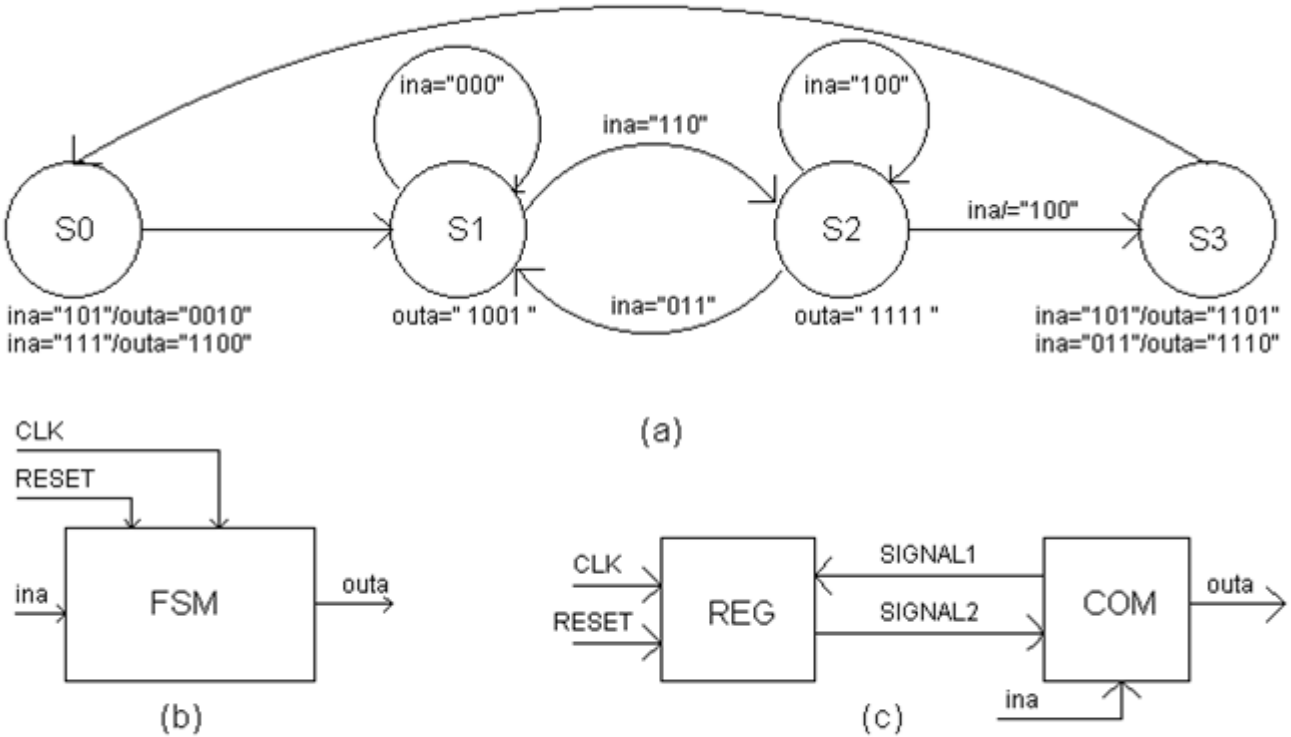


图 8-32　习题 8-3 状态图

实验与设计

8-1 序列检测器设计

8-2 ADC采样控制电路设计

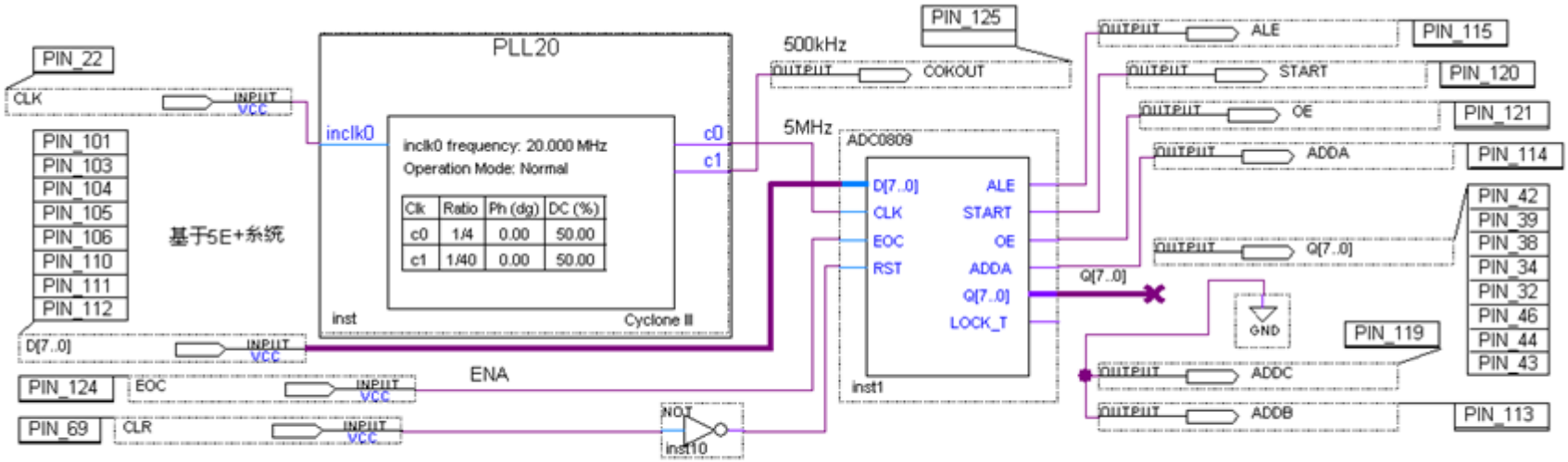


图 8-33 ADC0809 采样控制实验电路与引脚锁定指示

实验与设计

8-3 数据采集模块设计

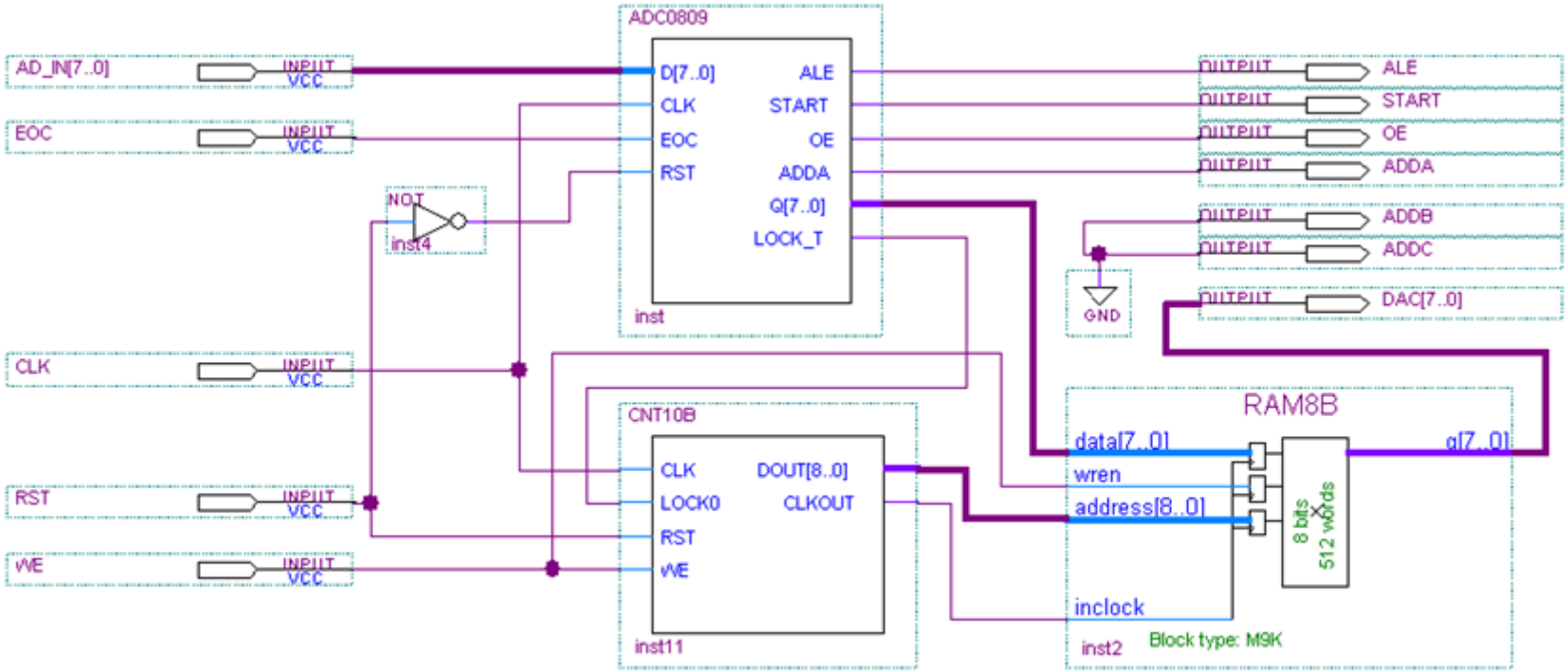


图 8-34　ADC0809 采样电路及简易存储示波器控制系统

实验与设计

8-4 五功能智能逻辑笔设计

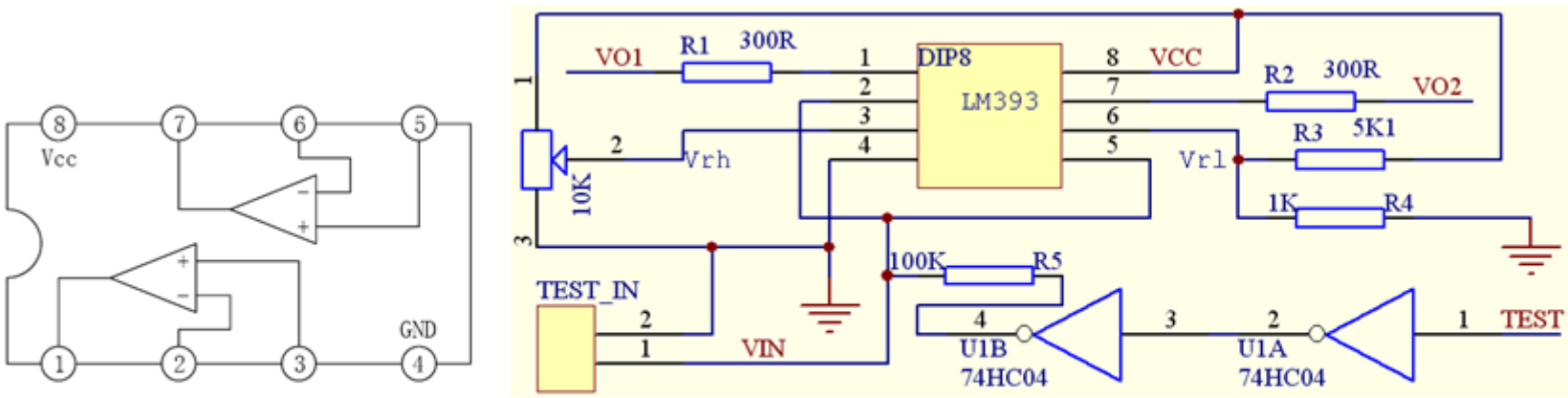


图 8-35 五功能智能逻辑笔电平信号采样电路，左图是 LM393 引脚图

实验与设计

8-5 比较器加DAC器件实现ADC转换功能电路设计

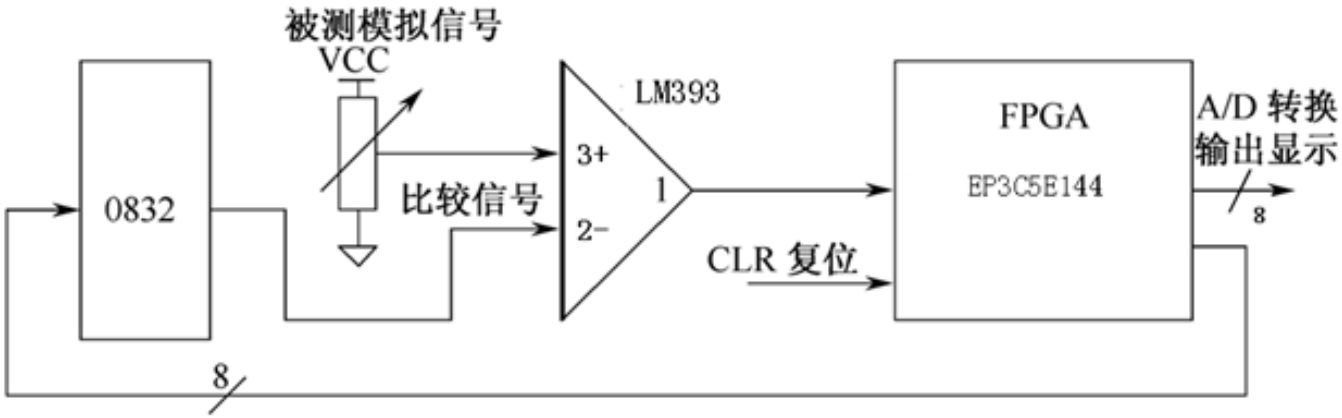


图 8-36　比较器和 D/A 构成 A/D 电路框图

实验与设计

8-6 通用异步收发器UART设计

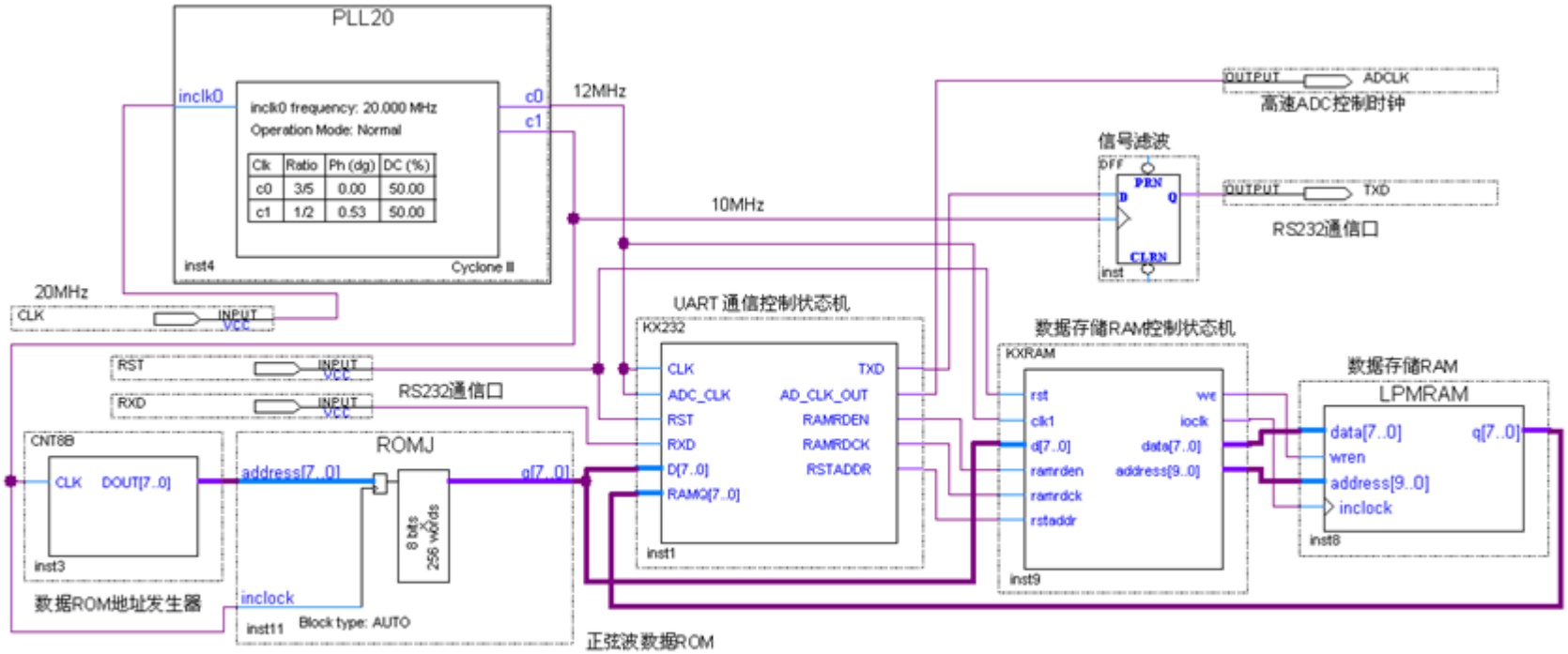


图 8-37 UART 通信设计顶层电路

实验与设计

8-6 通用异步收发器UART设计

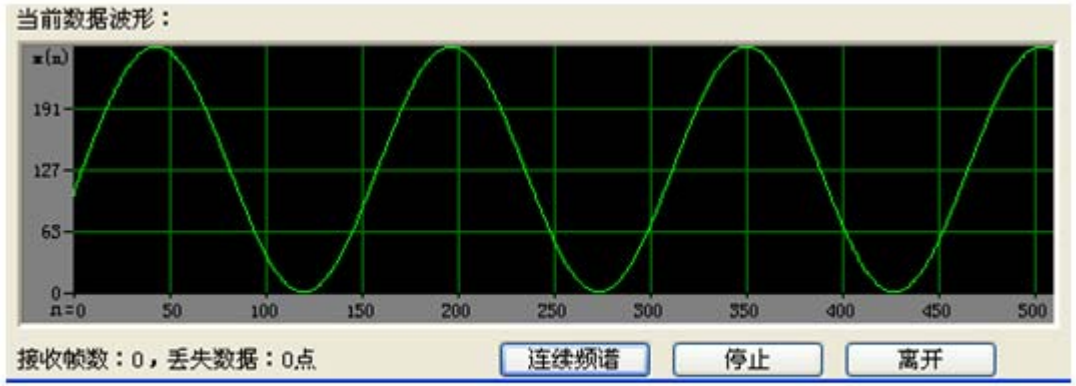


图 8-38　通过 RS232 通信来自 5E+系统的正弦波信号

实验与设计

8-7 点阵型与字符型液晶显示器驱动控制电路设计

8-8 串行**ADC/DAC**控制电路设计

8-9 硬件消抖动电路设计

8-10 数字彩色液晶显示控制电路设计

实验与设计

8-11 状态机控制串/并转换8数码静态显示

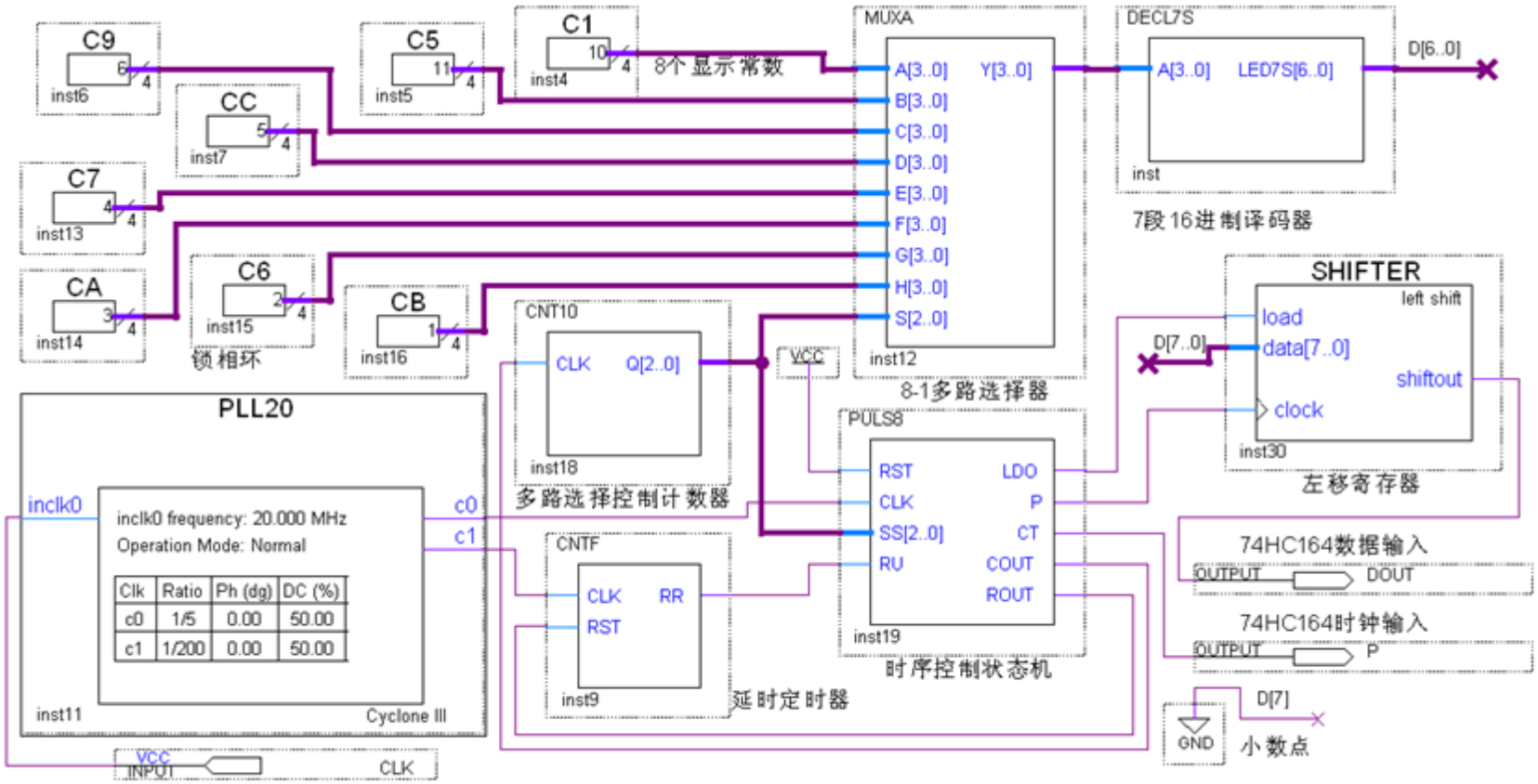
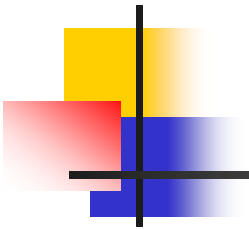


图 8-39　8 位串行静态显示状态机控制电路



实验与设计

8-12 基于CPLD的FPGA PS模式编程配置控制电路设计