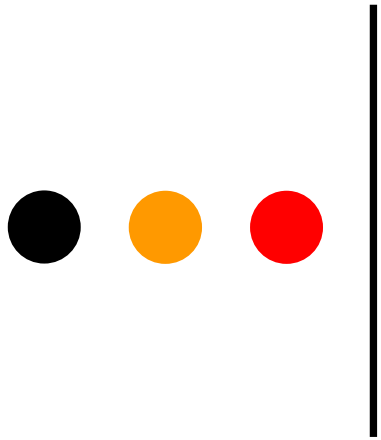

第10章

Verilog状态机设计技术

10.1 Verilog状态机的一般形式

10.1.1 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）性能稳定。

（4）高速性能。

（5）高可靠性能。

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

1. 状态机说明部分

```
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 ;
```

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

2. 主控时序过程

3. 主控组合过程

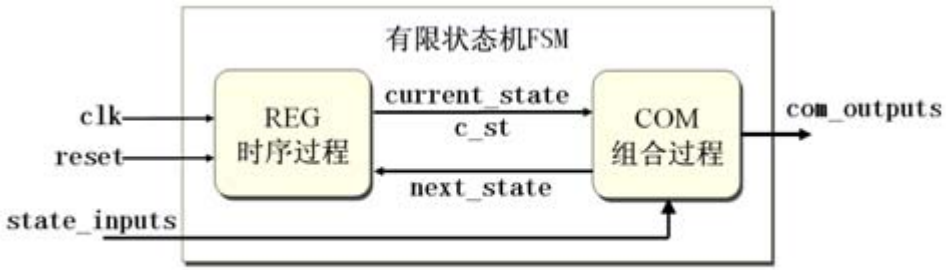

图 10-1 状态机一般结构示图

4. 辅助过程

【例 10-1】

```
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
  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
```

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

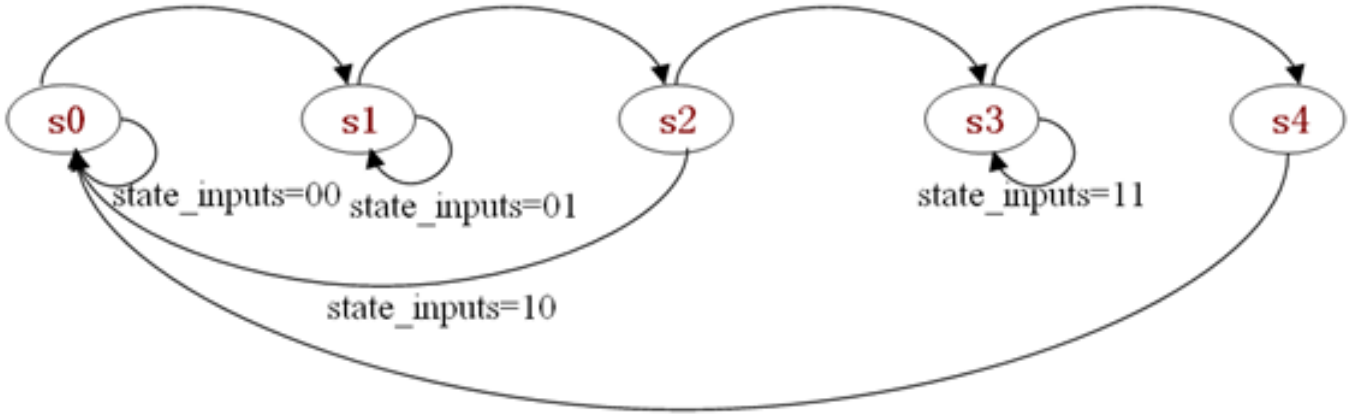

图 10-2　例 10-1 状态机的工作时序

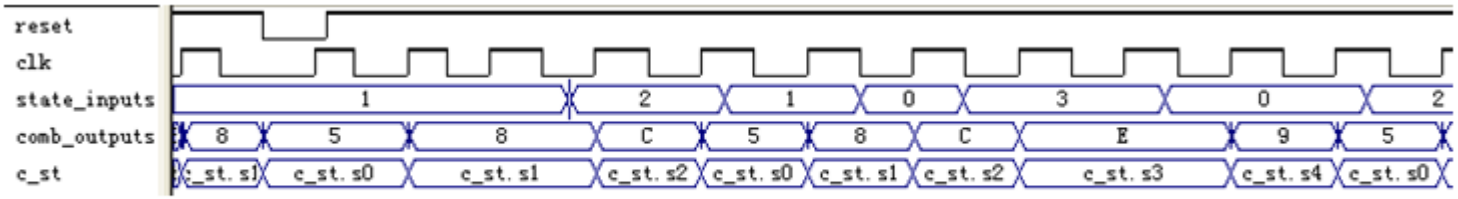

图 10-3　例 10-1 状态机的工作时序

10.1 Verilog状态机的一般形式

10.1.3 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。

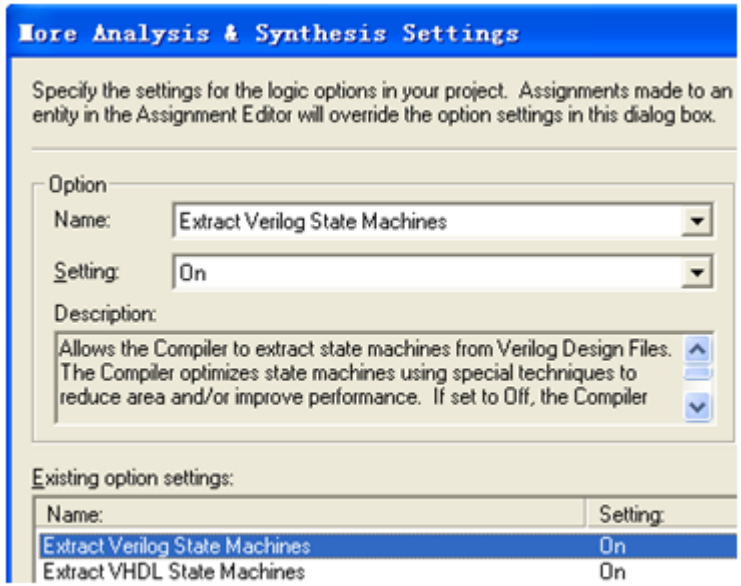

图 10-4 打开 Verilog 状态机萃取开关

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

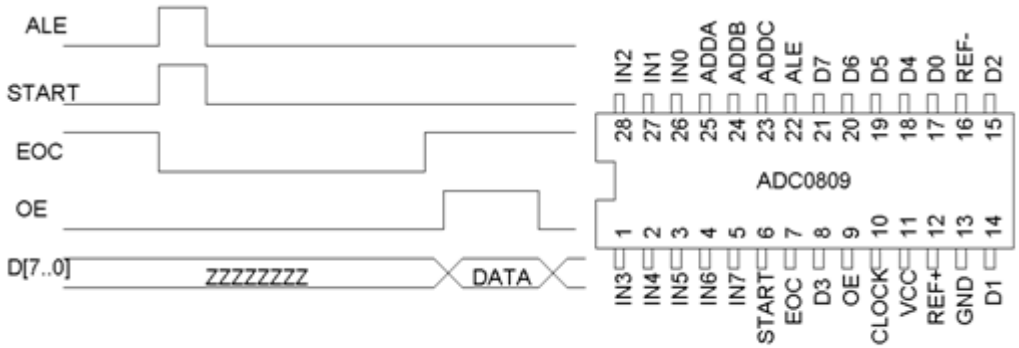

图 10-5　ADC0809 工作时序和芯片引脚图

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

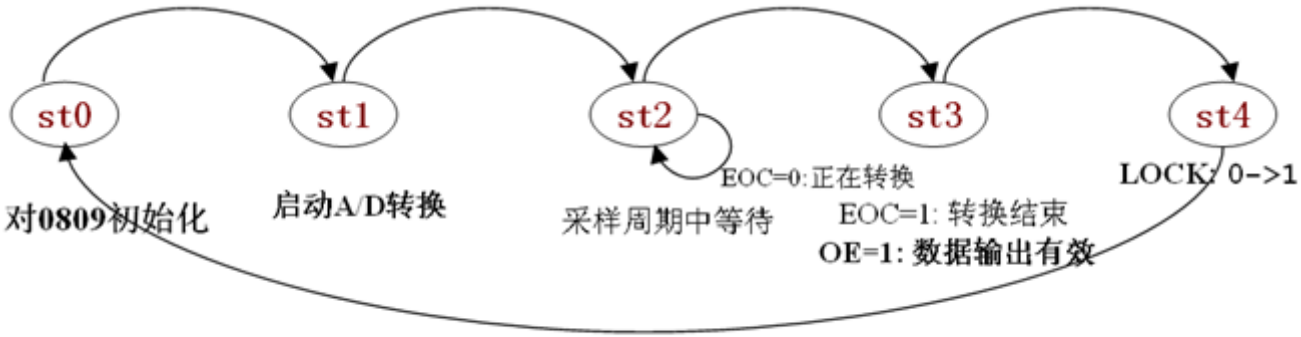

图 10-6 控制 ADC0809 采样状态图

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

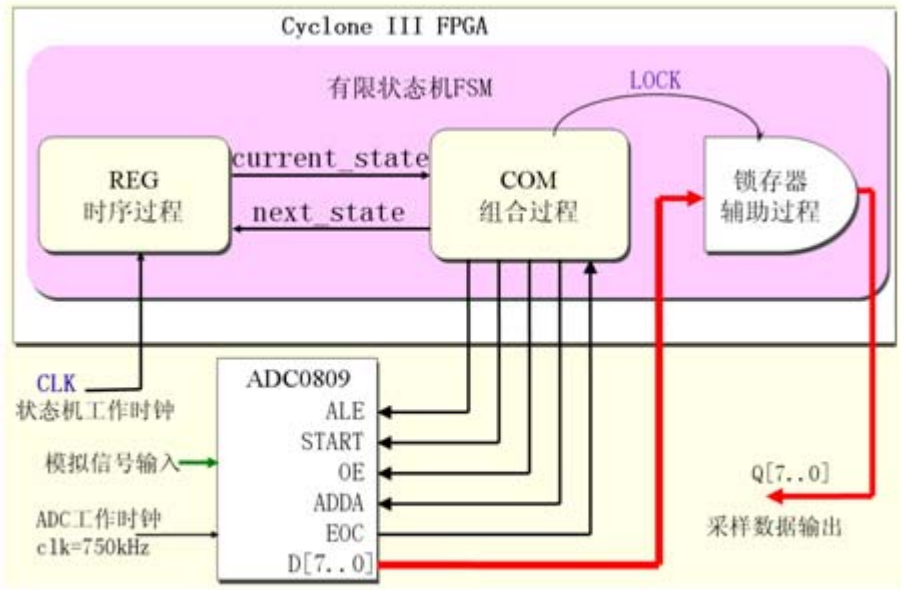

图 10-7 采样状态机结构框图

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

(* synthesis，probe_port，keep *)

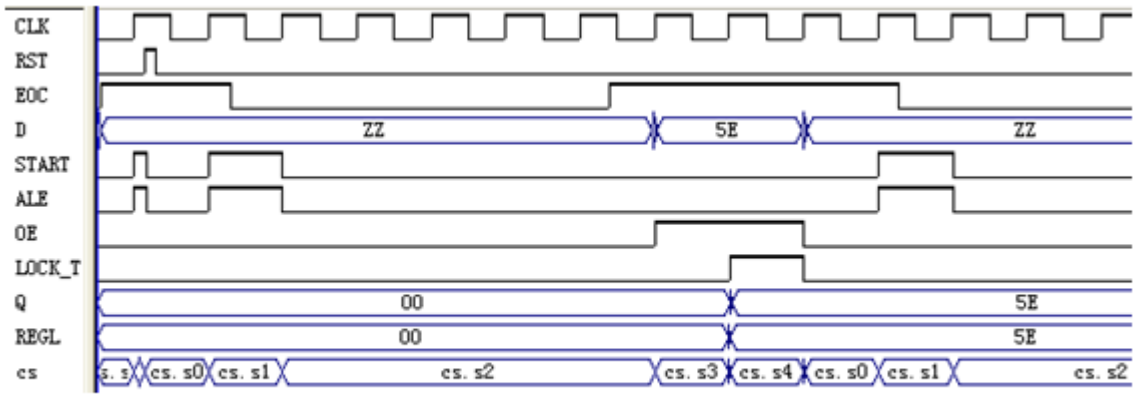

图 10-8　ADC0809 采样状态机工作时序

【例 10-2】

```
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 ;
                  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。
```

接下页

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

```
                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
  always @(posedge LOCK)  if(LOCK)   REGL<=D;//在LOCK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
```

10.2 Moore型状态机及其设计

【例 10-3】

```
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
```

10.2 Moore型状态机及其设计

10.2.2 序列检测器及其状态机设计

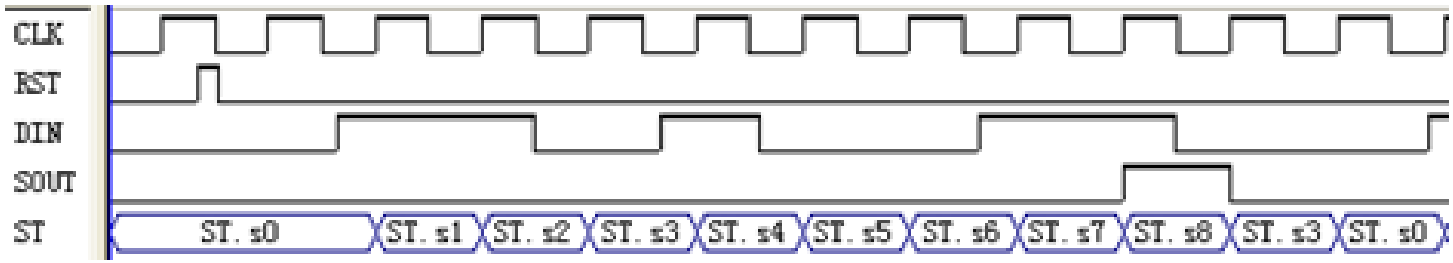

图 10-9　例 10-4 之序列检测器时序仿真波形

10.2 Moore型状态机及其设计

【例 10-4】

```
module  SCHK (input  CLK, DIN, RST,  output SOUT);
   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   //11010011 串行输入，高位在前。
      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
```

10.3 Mealy型状态机设计

【例 10-5】

```
module MEALY1 (input CLK, DIN1,DIN2, RST, output 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 ;      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
```

10.3 Mealy型状态机设计

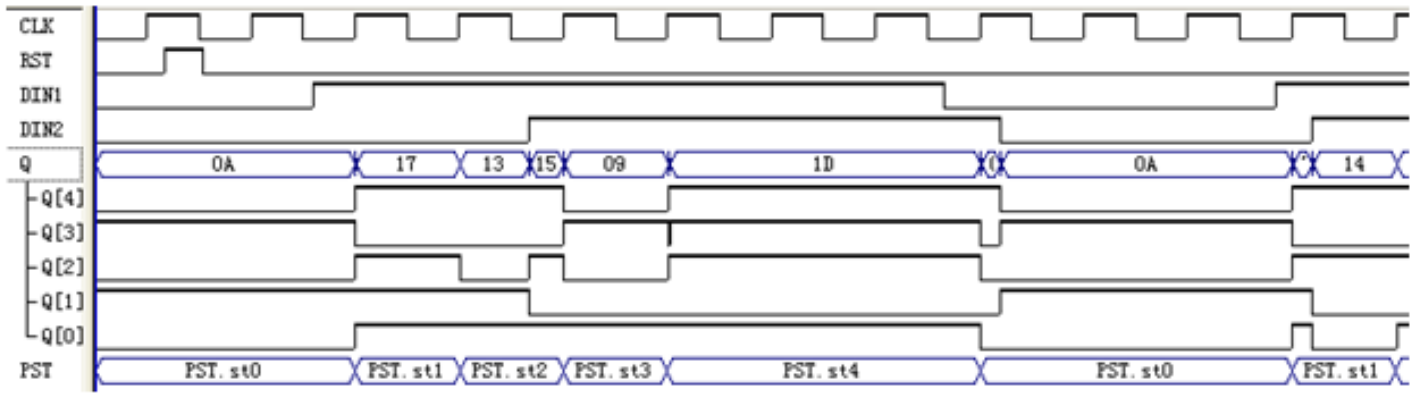

图 10-10 例 10-5 之双过程 Mealy 机仿真波形

10.3 Mealy型状态机设计

【例 10-6】

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

10.3 Mealy型状态机设计

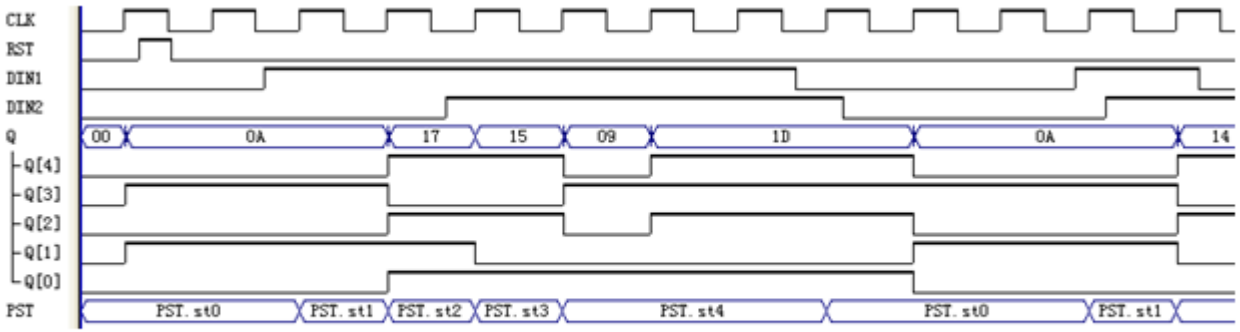

图 10-11　例 10-6 之单过程 Mealy 机仿真波形

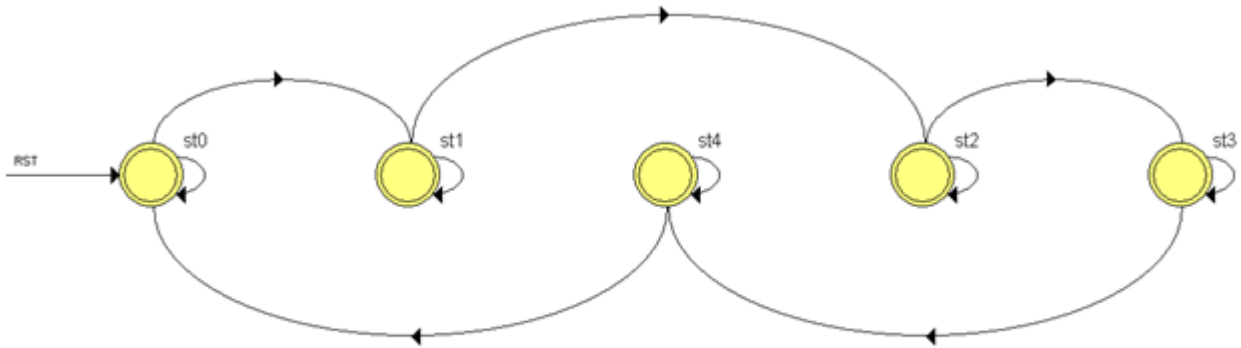

图 10-12　例 10-6，10-5 的状态图

10.3 Mealy型状态机设计

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

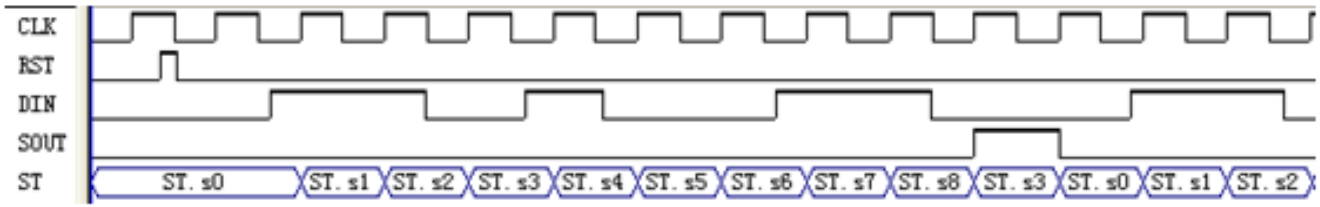

图 10-13　例 10-7 之单过程 Mealy 机仿真波形

10.3 Mealy型状态机设计

【例 10-7】

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

10.5 状态机图形编辑设计

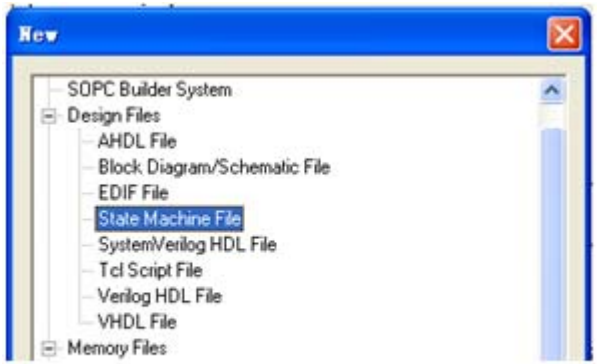

图 10-14 打开状态机图形编辑窗

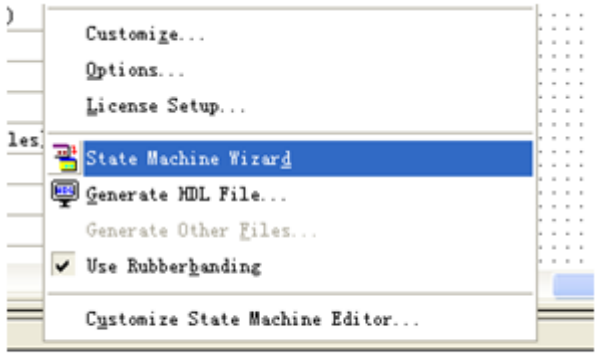

图 10-15 打开状态机编辑器

10.5 状态机图形编辑设计

States:

	State Name
1	st1
2	st2
3	st3
4	<< new state >>

Input ports:

	Input Port Name
2	reset
3	in1
4	in2
5	in3
6	<< new input port >>

State transitions:

	Source State	Destination State	Transition
2		st1	OTHERS
3		st3	in1 & ~in2
4		st1	OTHERS
5		st1	in1 | in2
6		st3	OTHERS

图 10-16 设置状态变量和状态转换条件

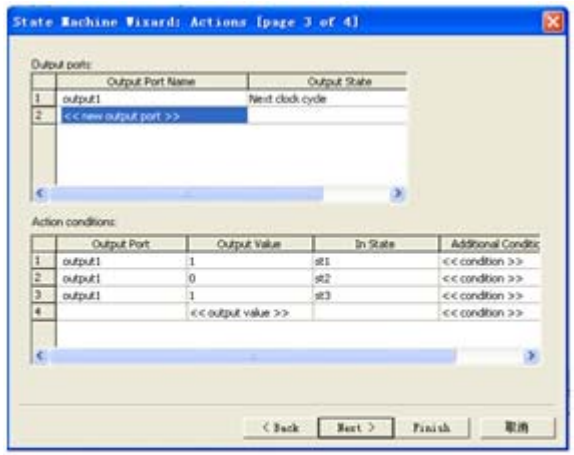

图 10-17 设置状态机输出

10.5 状态机图形编辑设计

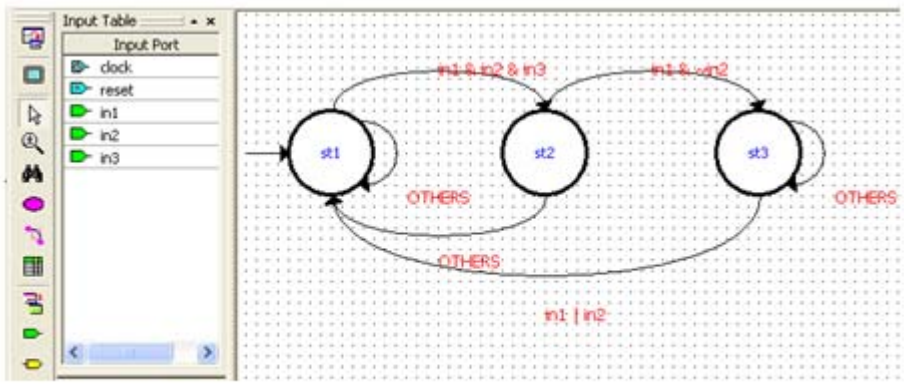

图 10-18 状态机图形编辑器上的状态图

图 10-19 状态机转变成语言

10.6 不同编码类型状态机

10.6.1 直接输出型编码

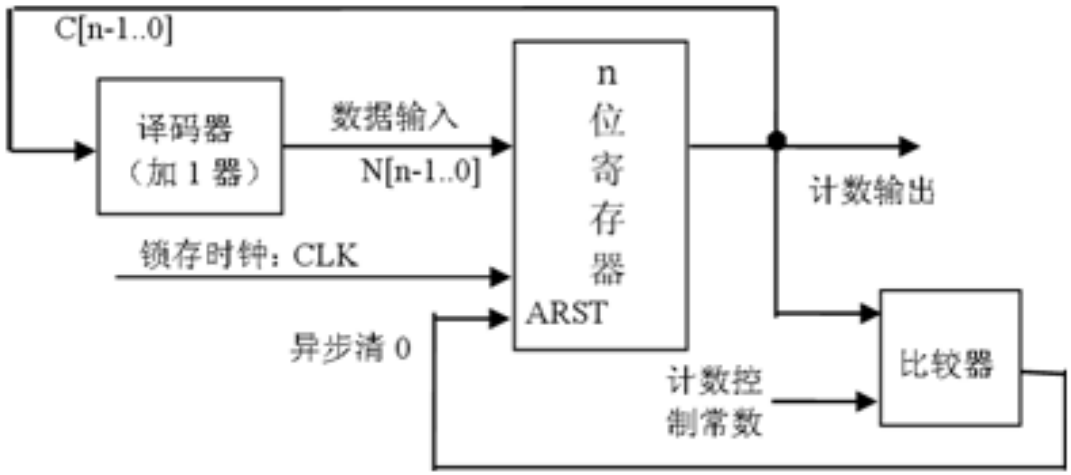

图 10-20 加法计数器一般模型

10.6 不同编码类型状态机

10.6.1 直接输出型编码

表 10-1 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

【例 10-8】此程序的硬件实测方法可参考实验 10-2

```
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
```

10.6 不同编码类型状态机

10.6.1 直接输出型编码

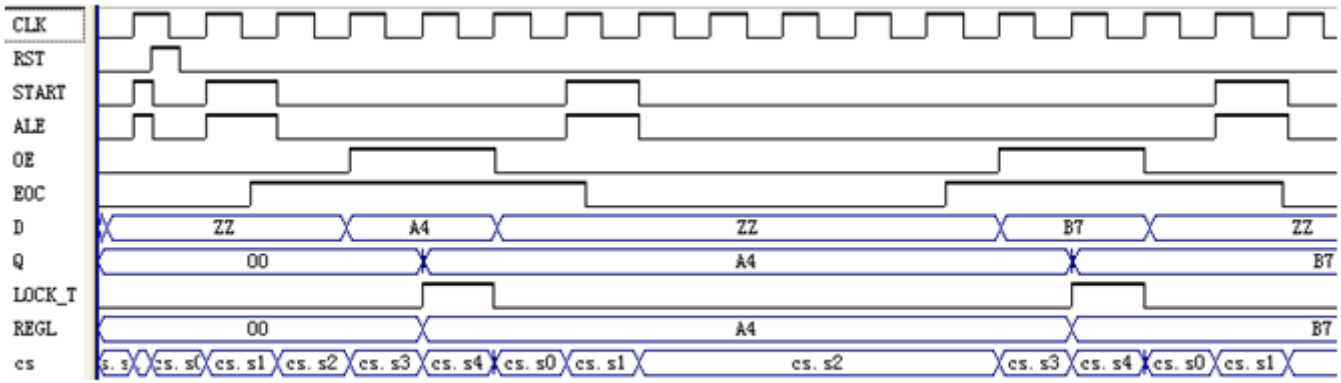

图 10-21 例 10-10 状态机工作时序图

10.6 不同编码类型状态机

10.6.2 用宏定义语句定义状态编码

【例 10-9】

```
`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)
      `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
```

接下页

10.6 不同编码类型状态机

10.6.2 用宏定义语句定义状态编码

```
  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
```

图 10-22 不同状态元素定义状态机的仿真波形

10.6 不同编码类型状态机

10.6.3 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
```

10.6 不同编码类型状态机

10.6.4 顺序编码

表 10-2　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

10.6 不同编码类型状态机

10.6.5 一位热码编码

10.6.6 状态编码设置

1. 用户自定义方式

2. 用属性定义语句设置

【例 10-10】

```
module SCHK  (input  CLK, DIN, RST, output reg 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 ;
always @(posedge CLK)   begin
```

10.6 不同编码类型状态机

10.6.6 状态编码设置

2. 用属性定义语句设置

表 10-3 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

10.6 不同编码类型状态机

10.6.6 状态编码设置

3. 直接设置方法

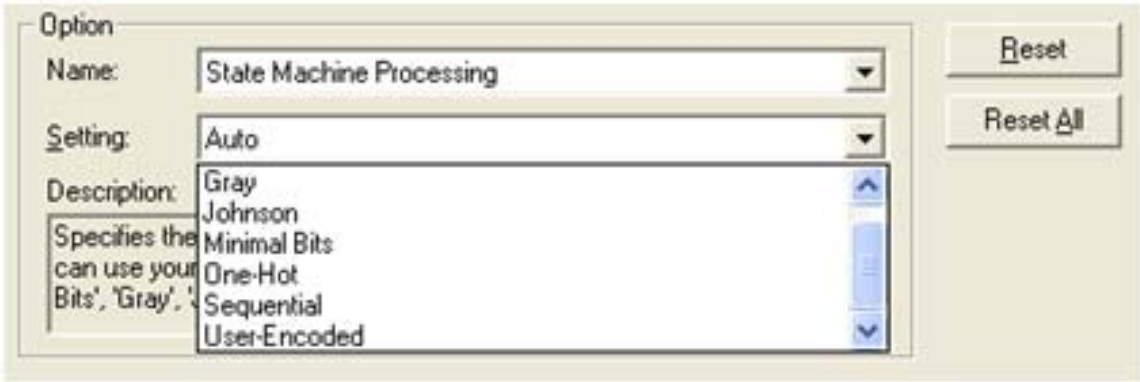

图 10-23 选择恰当的编码形式

10.7 异步有限状态机设计

1、异步状态机设计示例1

【例 10-11】

```
module ASM_WAVE1 (input CLK,RST,  output W1,W2);
    (* synthesis, keep *)  reg [1:0] CS; //为在波形图中了解 CS 的情况
    reg [1:0] NS;    reg W1,W2;   parameter  S0=1,S1=3,S2=2,S3=0;
  always @ (RST or CS)  //注意其状态编码形式，这将影响状态机功能
    begin  if (RST)  CS<=S0; else CS<=NS; end
  always @(CS)    begin
     if (CS==S0) begin W1=1'b0; W2=1'b0; end
     if (CS==S1) begin W1=1'b1; W2=1'b0; end
     if (CS==S2) begin W1=1'b1; W2=1'b1; end
     if (CS==S3) begin W1=1'b0; W2=1'b1; end     end
  always @ (CLK)    begin
    case(CS)
      S0 : if (CLK)  NS = S1; else NS = S0;
      S1 : if (~CLK) NS = S2; else NS = S1;
      S2 : if (CLK)  NS = S3; else NS = S2;
      S3 : if (~CLK) NS = S0; else NS = S3;
      default: NS=S0;
    endcase    end
 endmodule
```

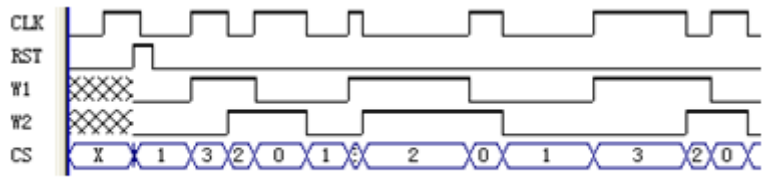

图 10-24　例 10-11 的工作时序图

10.7 异步有限状态机设计

2、异步状态机设计示例2

【例 10-12】

```
module ASM_WAVE2 (CLK1,CLK2,RST,W) ;
  input CLK1,CLK2,RST;    output W;   reg [1:0] NS;    reg W, Z;
    (* synthesis, keep *)  reg [1:0] CS;    //为在波形图中了解 CS 的情况
  parameter  S0=2'b00,S1=2'b01,S2=2'b11,S3=2'b10;//注意其状态编码形式
  always @ (RST or CS)    begin  if (RST)  CS<=S0; else CS<=NS; end
  always @ (CS or W)   begin
      case(CS)
        S0 : if (W==1) Z=0 ; else Z=1;
        S1 : if (W==1) Z=0 ; else Z=1;
        S2 : if (W==1) Z=0 ; else Z=1;
    endcase     end
 always @ (CS or CLK1 or CLK2)   begin
     case(CS)
       S0 : if ({CLK1,CLK2}==2'b00)       begin NS <= S0; W<=W; end
            else if ({CLK1,CLK2}==2'b01)  NS <= S1;
            else if ({CLK1,CLK2}==2'b10)  NS <= S2;
            else if ({CLK1,CLK2}==2'b11)  NS <= S3;
            else NS <= S0;
       S1 : if ({CLK1,CLK2}==2'b00)       NS <= S0;
            else if ({CLK1,CLK2}==2'b01)  begin NS <= S1; W<=1'b0;end
```

接下页

10.7 异步有限状态机设计

2、异步状态机设计示例2

```
        else if ({CLK1,CLK2}==2'b10)  NS <= S2;
        else if ({CLK1,CLK2}==2'b11)  NS <= S3;
        else NS <= S0;
    S2 : if ({CLK1,CLK2}==2'b00)      NS <= S0;
        else if ({CLK1,CLK2}==2'b01)  NS <= S1;
        else if ({CLK1,CLK2}==2'b10)  begin NS <= S2; W<=1'b1; end
        else if ({CLK1,CLK2}==2'b11)  NS <= S3;
        else NS <= S0;
    S3 : if ({CLK1,CLK2}==2'b00)      NS <= S0;
        else if ({CLK1,CLK2}==2'b01)  NS <= S1;
        else if ({CLK1,CLK2}==2'b10)  NS <= S2;
        else if ({CLK1,CLK2}==2'b11)  begin NS <= S3;W<=Z;end
        else NS <= S0;
    endcase     end
endmodule
```

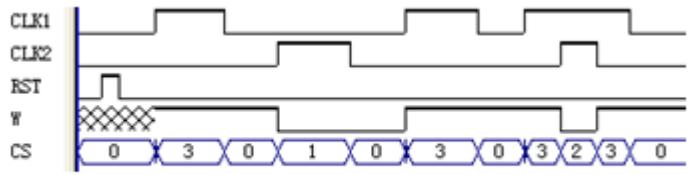

图 10-25　例 10-12 的工作时序图

10.7 异步有限状态机设计

【例 10-13】

```
module ASM3 (input CLK1,CLK2,RST, output reg W);
  (* synthesis, keep *)  reg [2:0] CS;
  reg [2:0] NS; wire CLK; //注意其状态编码形式
    parameter  S0=3'b000,S1=3'b001,S2=3'b011,S3=3'b010,
               S4=3'b110,S5=3'b111,S6=3'b101,S7=3'b100;
  assign CLK=CLK1 & CLK2;
  always @ (RST or CS)  if (RST)  CS<=S0; else CS<=NS;
  always @ (CS or CLK)   begin
     case(CS)
       S0 : if (~CLK)  NS = S1; else begin NS = S0; W=1'b0; end
       S1 : if (CLK)   NS = S2; else begin NS = S1; W=1'b0; end
       S2 : if (~CLK)  NS = S3; else begin NS = S2; W=1'b0; end
       S3 : if (CLK)   NS = S4; else begin NS = S3; W=1'b0; end
       S4 : if (~CLK)  NS = S5; else begin NS = S4; W=1'b0; end
       S5 : if (CLK)   NS = S6; else begin NS = S5; W=1'b0; end
       S6 : if (~CLK)  NS = S7; else begin NS = S6; W=1'b1; end
       S7 : NS = S0;
    default: begin NS = S0;W=1'b0; end
     endcase   end
endmodule
```

10.7 异步有限状态机设计

2、异步状态机设计示例2

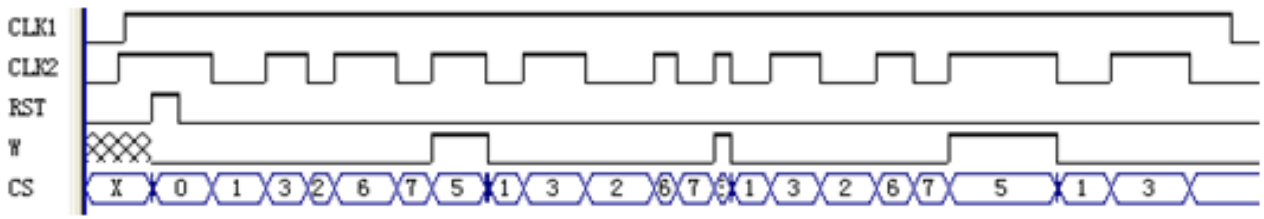

图 10-26 例 10-13 的工作时序图

10.8 安全状态机设计

表 10-4 **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

10.8 安全状态机设计

10.8.1 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 ;
```

10.8 安全状态机设计

10.8.2 状态编码监测法

10.8.3 借助EDA工具自动生成安全状态机

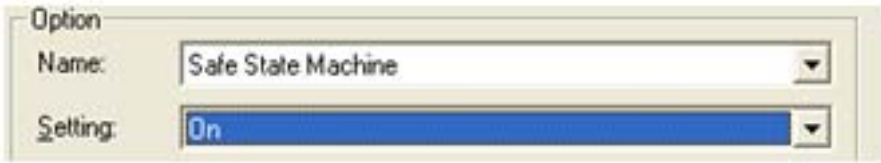

图 10-27 选择安全状态机设计

10.9 硬件数字技术排除毛刺

10.9.1 延时方式去毛刺

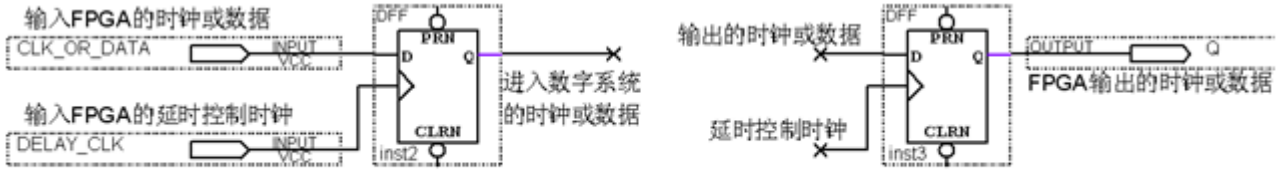

图 10-28 单触发器输入(左图)或输出(右图)延时电路

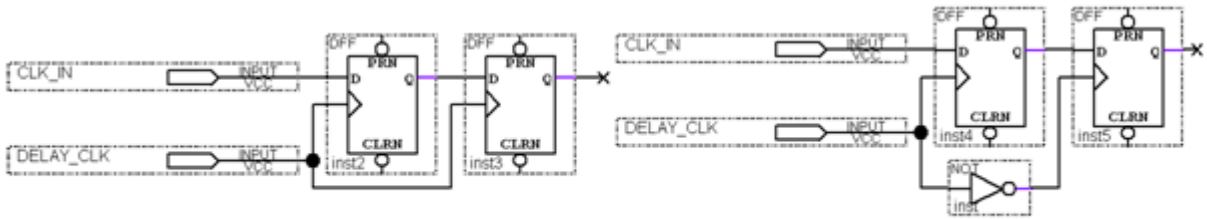

图 10-29 双触发器延时电路

10.9 硬件数字技术排除毛刺

10.9.1 延时方式去毛刺

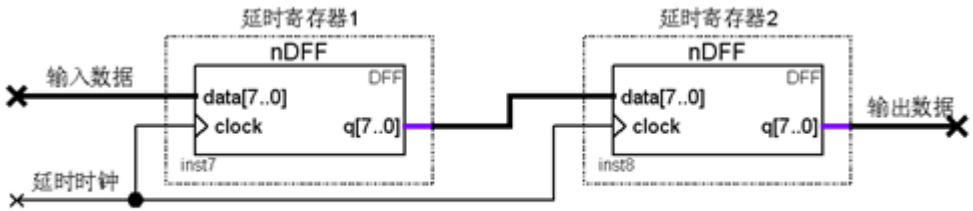

图 10-30 双寄存器数据延时电路

10.9 硬件数字技术排除毛刺

10.9.2 逻辑方式去毛刺

图 10-31 信号上升与下降沿都含随机干扰抖动信号

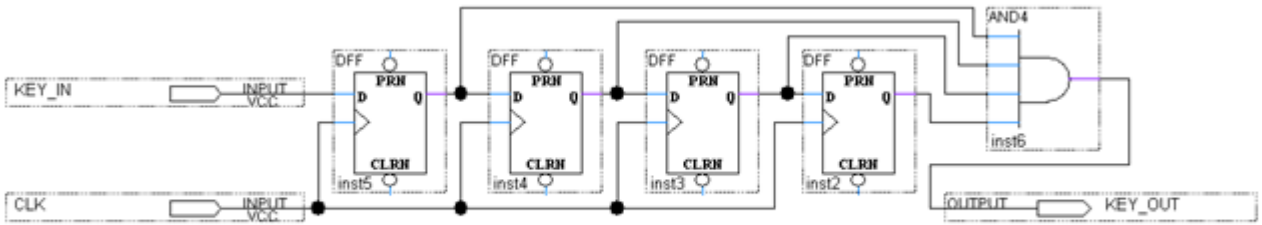

图 10-32 消抖动电路

10.9 硬件数字技术排除毛刺

10.9.2 逻辑方式去毛刺

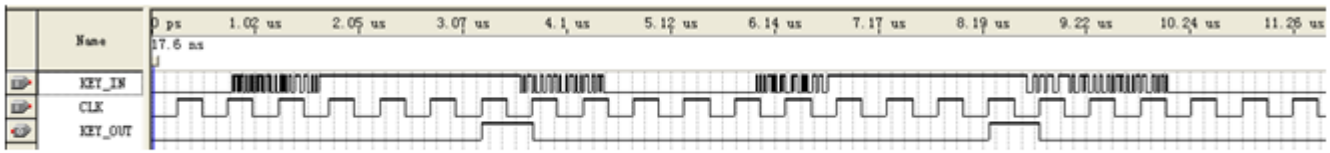

图 10-33 消抖动电路仿真波形

实验与设计

10-1 序列检测器设计

10-2 **ADC采样控制电路设计**

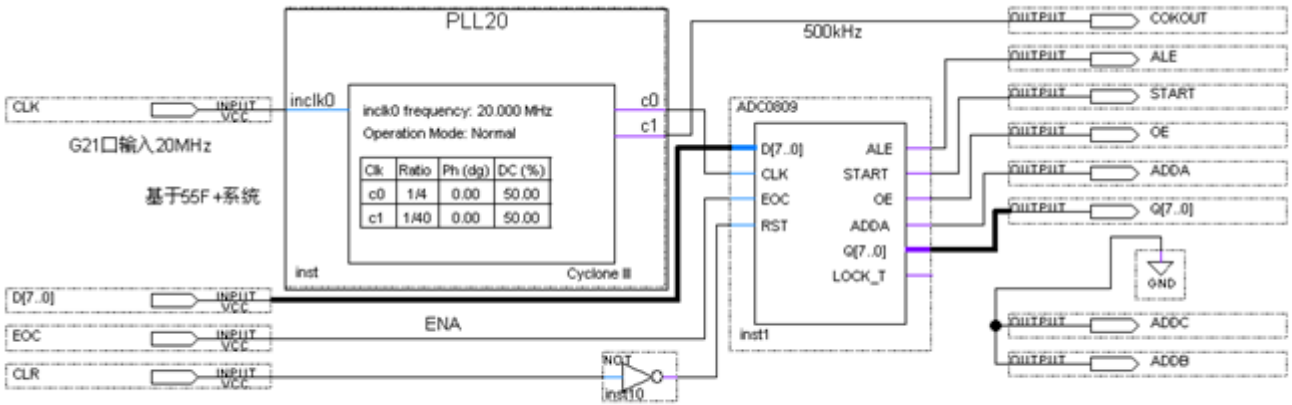

图 10-34　ADC0809 采样控制实验电路

实验与设计

10-3 数据采集模块设计

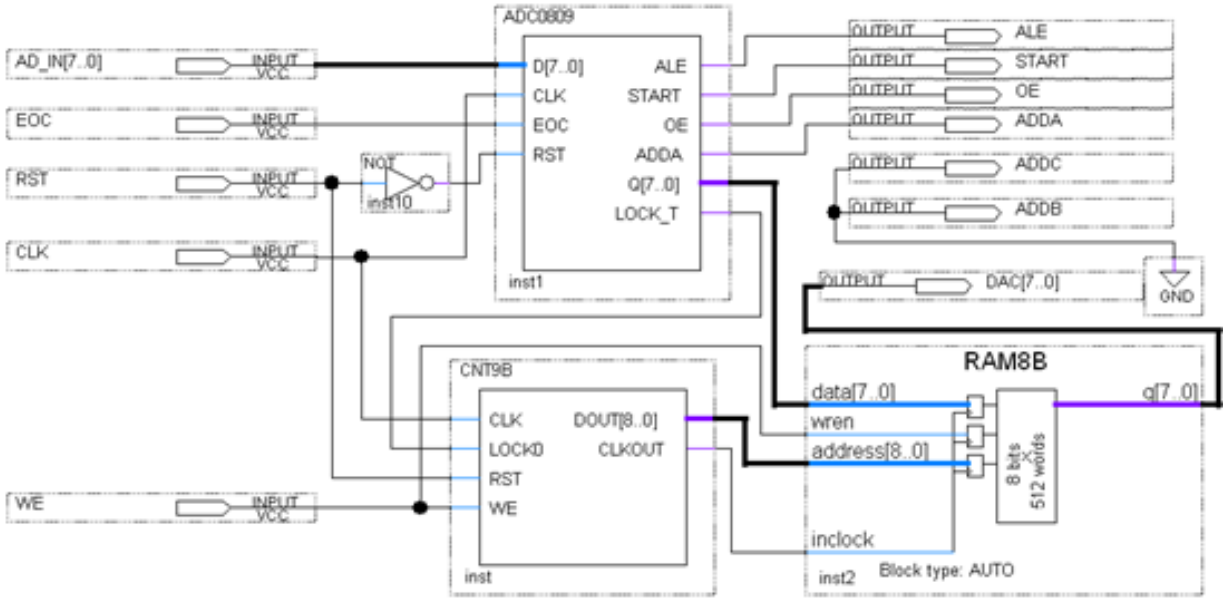

图 10-35　ADC0809 采样电路及简易存储示波器控制系统

实验与设计

10-3 数据采集模块设计

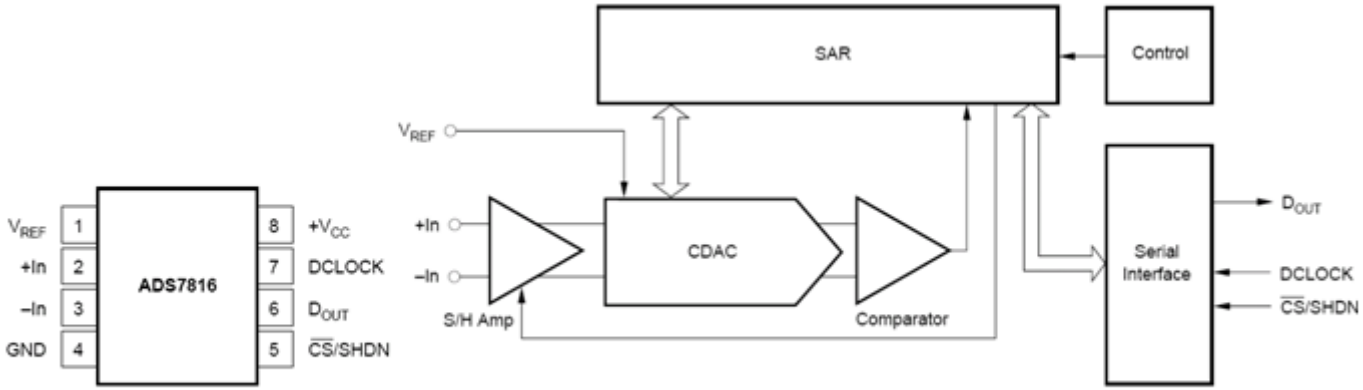

图10-36 ADS7816的引脚和结构

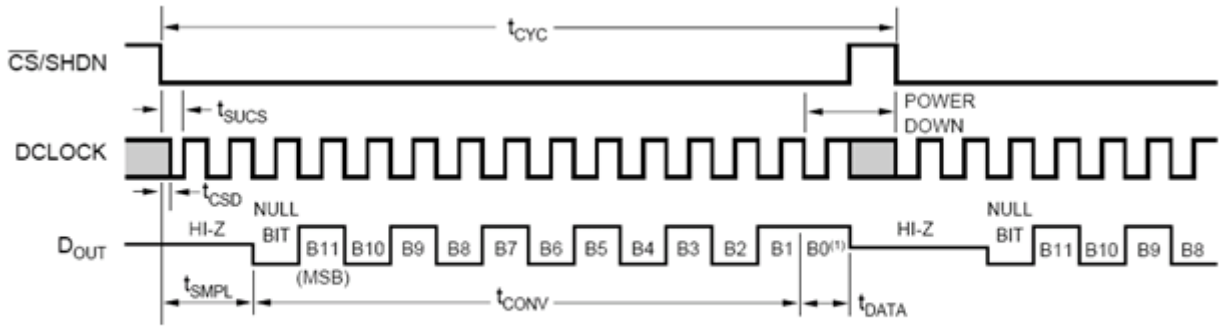

图10-37 ADS7816的工作时序

实验与设计

10-4 五功能智能逻辑笔设计

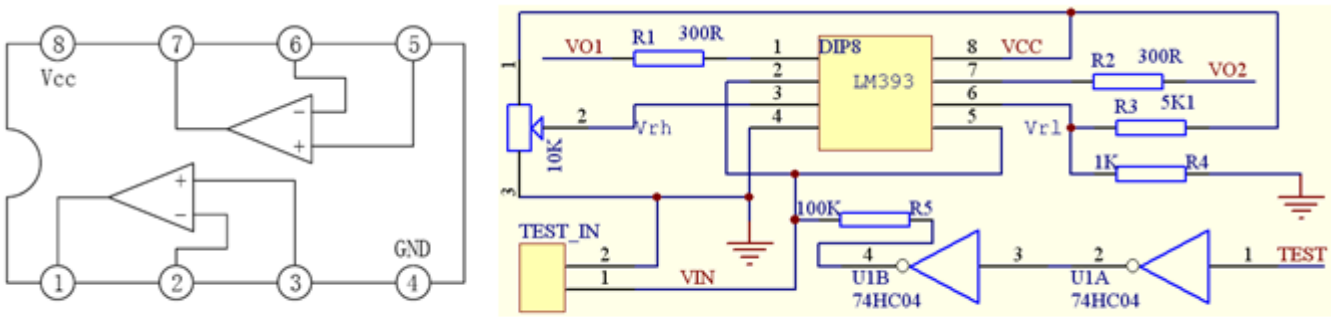

图 10-38 五功能智能逻辑笔电平信号采样电路，左图是 LM393 引脚图

【例 10-14 16】五功能智能逻辑笔参考程序

```
module  LGC_PEN (CLK, V0,TEST,LED);
   input  CLK;        //状态机工作时钟
   input [2:1] V0;   //输入来自 LM393 中两个比较器的输出信号
   output[4:0] LED; output TEST; //外接 5 个指示发光管以及测试高阻态信号 TEST
   parameter   s0=0, s1=1, s2=2, s3=3, s4=4, s5=5, s6=6, s7=7,
               s8=8,s9=9, s10=10, s11=11, s12=12, s13=13 ;
   reg[4:0] ST,NST ; reg TEST;  reg[3:0] LED ;
   always@(posedge CLK )    ST<=NST ;
   always @(ST or V0)   begin
     case (ST )
     s0: begin  TEST <=1'b1;  NST<=s1;  end
     s1: begin  TEST<=1'b1;  if (V0==2'b10)  NST<=s2; else NST<=s4; end
     s2: begin  TEST <=1'b0;   NST<=s3;  end
     s3: begin  TEST <=1'b0 ;
          begin if (V0==2'b01)  begin LED<=4'b1000; NST<=s0;  end
            else NST<=s4; end end
     s4: if(V0==2'b01)  NST<=s5; else NST<=s7;
     s5: if(V0==2'b01)  NST<=s6; else NST<=s7;
     s6: if(V0==2'b01)  begin LED<=4'b0001; NST<=s0; end  else NST<=s7;
     s7: if(V0==2'b10)  NST<=s8; else NST<=s10;
     s8: if(V0==2'b10)  NST<=s9; else NST<=s10;
     s9: if(V0==2'b10)  begin LED<=4'b0010; NST<=s0; end  else NST<=s10;
    s10: if(V0==2'b11)  NST<=s11; else NST<=s13;
    s11: if(V0==2'b11)  NST<=s12; else NST<=s13;
    s12: if(V0==2'b11)  begin LED<=4'b0100; NST<=s0; end  else NST<=s13;
    s13: begin  LED<=4'b1111; NST<=s0;   end
       default :   NST<=s0;
      endcase       end
endmodule
```