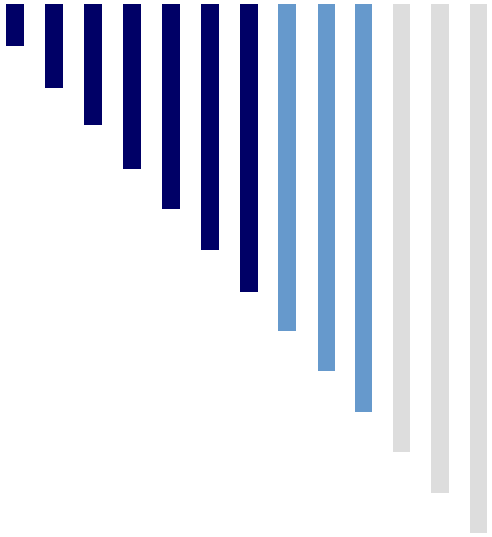

第7章

有限状态机设计技术

7.1 状态机的一般形式

7.1.1 状态机的基本结构

1. 状态机说明部分

```
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

7.1 状态机的一般形式

7.1.1 状态机的基本结构

3. 主控组合过程

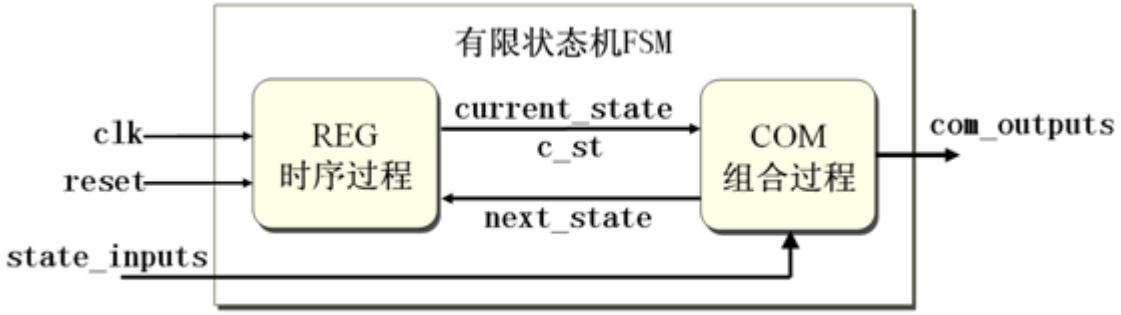

图 7-1　状态机一般结构示图

4. 辅助过程

【例 7-1】

```
module FSM_EXP (clk, reset, state_inputs, comb_outputs);
input clk,  reset;                                  // 状态机工作时钟和复位控制
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)
     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
```

接下页

7.1 状态机的一般形式

7.1.1 状态机的基本结构

```
        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
```

7.1 状态机的一般形式

7.1.1 状态机的基本结构

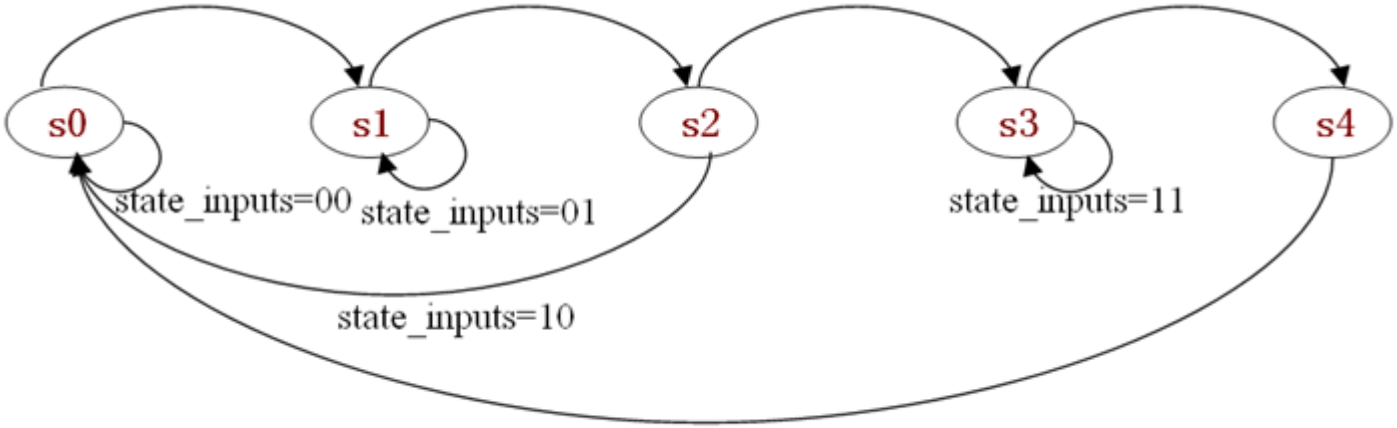

图 7-2　例 7-1 对应的状态图

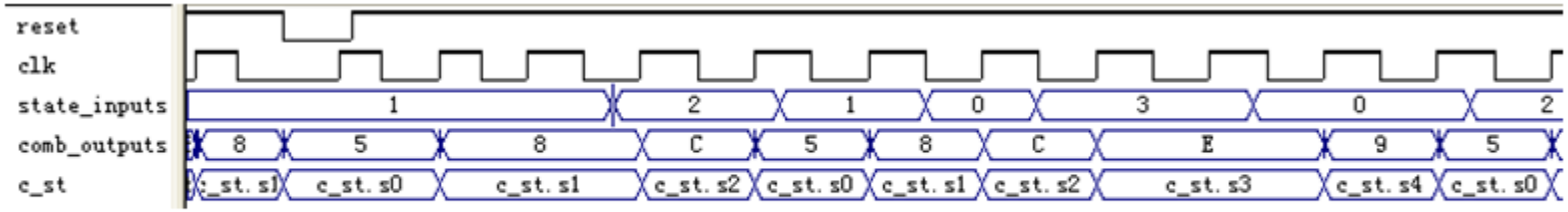

图 7-3　例 7-1 状态机的工作时序

7.1 状态机的一般形式

7.1.2 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。

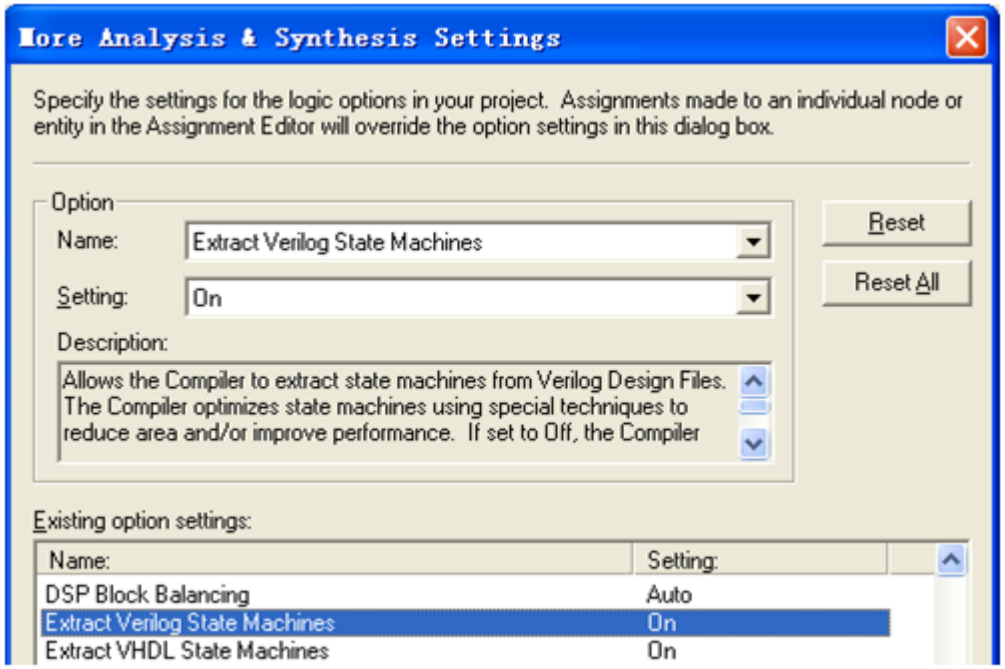

图 7-4　打开 Verilog 状态机萃取开关

7.2 Moore型有限状态机

7.2.1 实用状态机设计示例

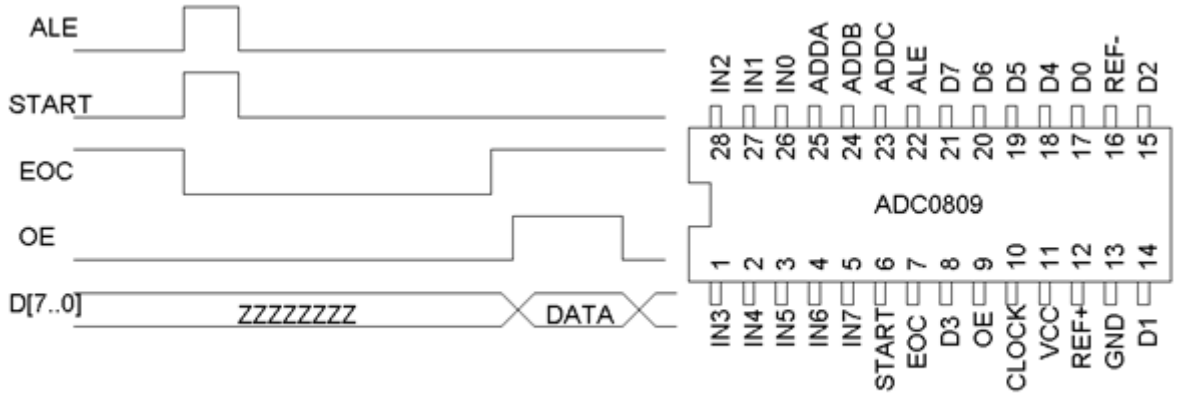

图 7-5　ADC0809 工作时序和芯片引脚图

7.2 Moore型有限状态机

7.2.1 实用状态机设计示例

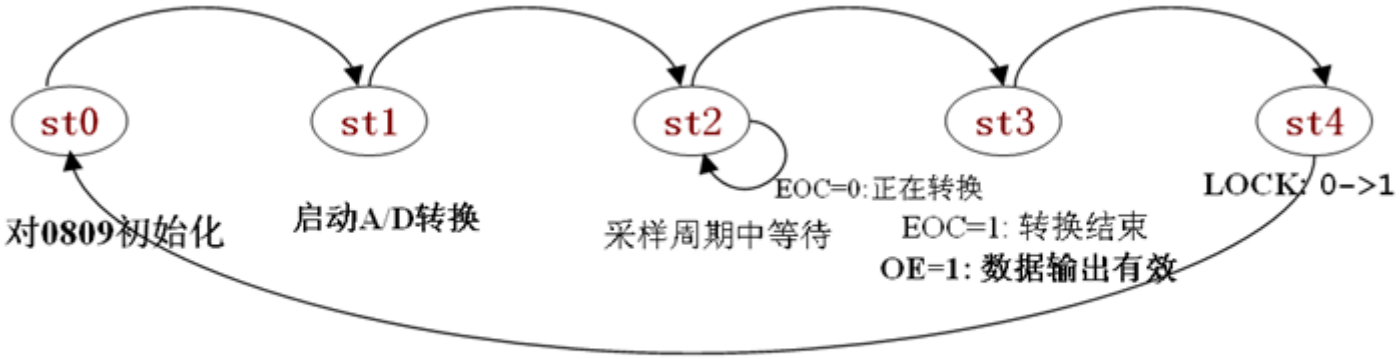

图 7-6 控制 ADC0809 采样状态图

7.2 Moore型有限状态机

7.2.1 实用状态机设计示例

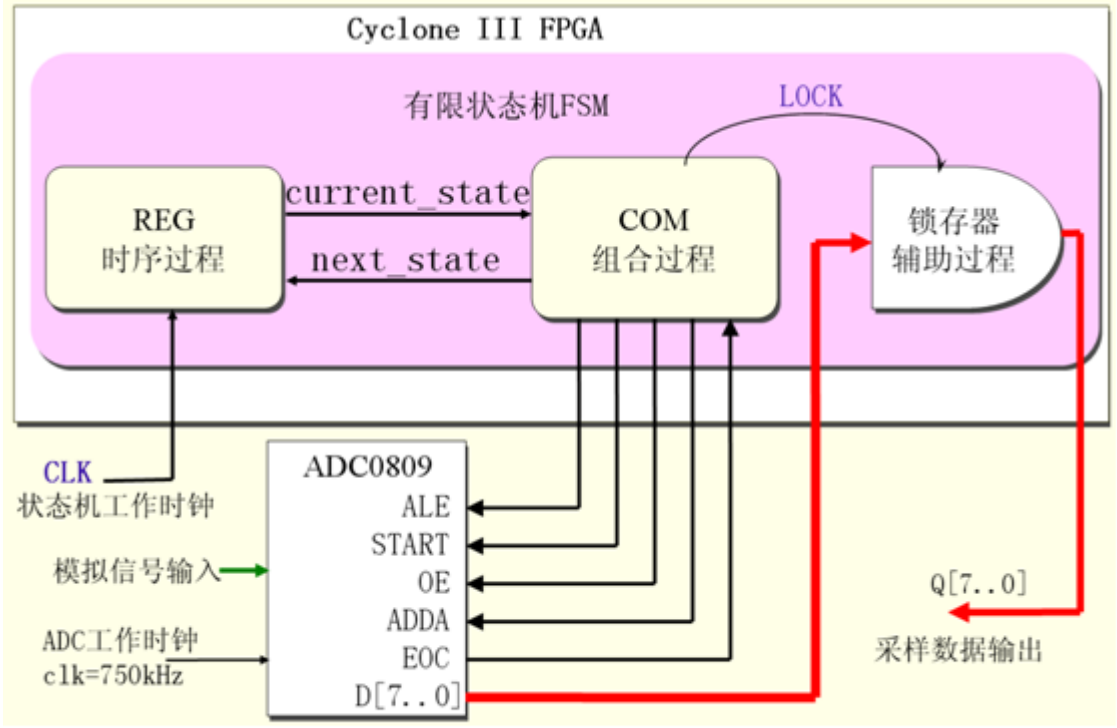

图 7-7 采样状态机结构框图

【例 7-2】

```
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
```

接下页

```
      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
```

7.2 Moore型有限状态机

7.2.1 实用状态机设计示例

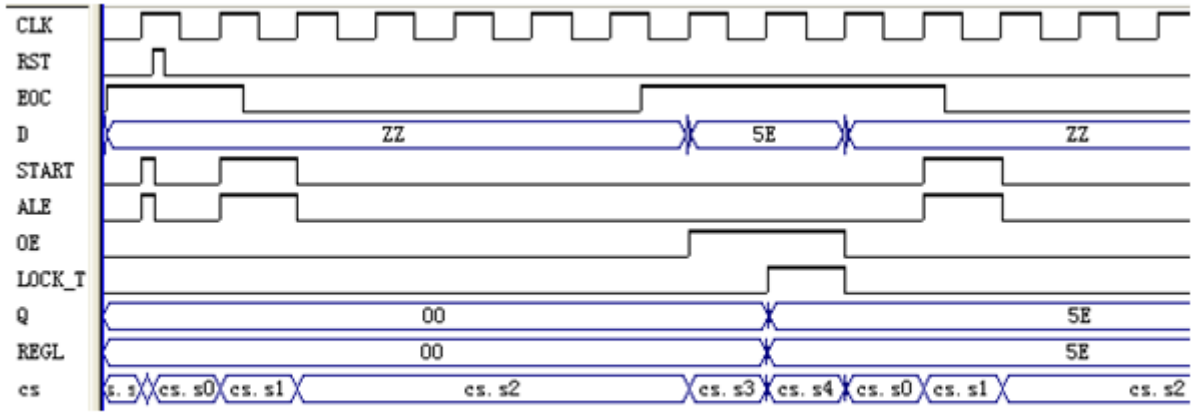

图 7-8　ADC0809 采样状态机工作时序

【例 7-3】

```
always @(cs or EOC)  begin    //状态转换过程
     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      //控制信号输出过程
     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
```

7.2.2 序列检测状态机设计

【例 7-4】

```
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
```

7.2 Moore型有限状态机

7.2.2 序列检测状态机设计

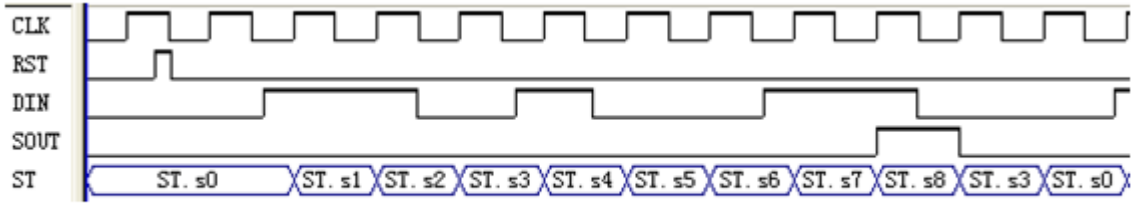

图 7-9　例 7-4 之序列检测器时序仿真波形

7.3 Mealy型状态机设计

【例 7-5】

```
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 ;   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
```

7.3 Mealy型状态机设计

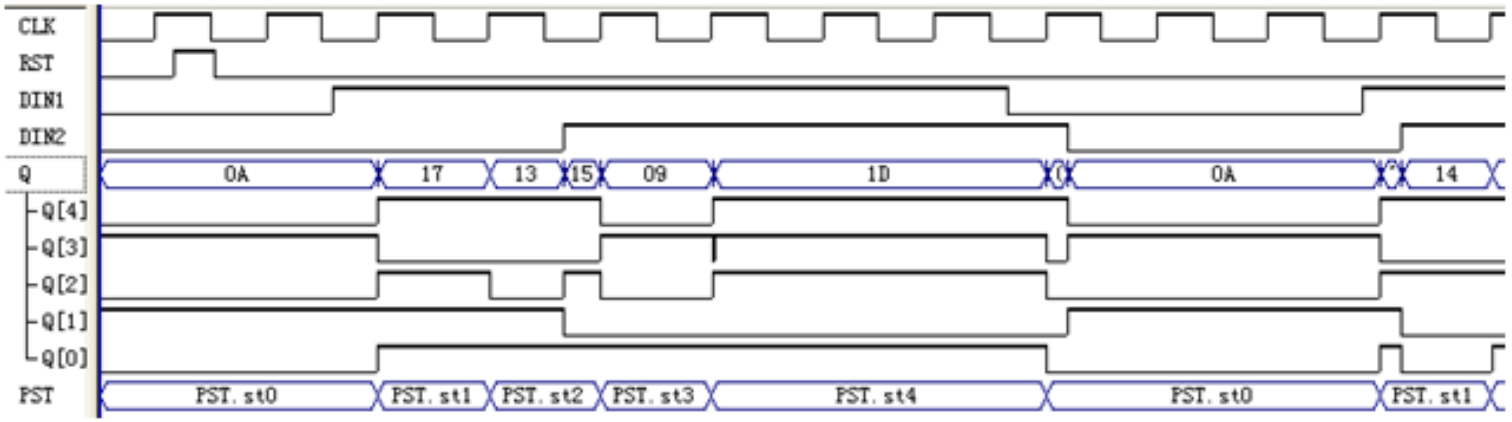

图 7-10 例 7-5 之双过程 Mealy 机仿真波形

7.3 Mealy型状态机设计

【例 7-6】

```
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
       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
endmodule
```

7.3 Mealy型状态机设计

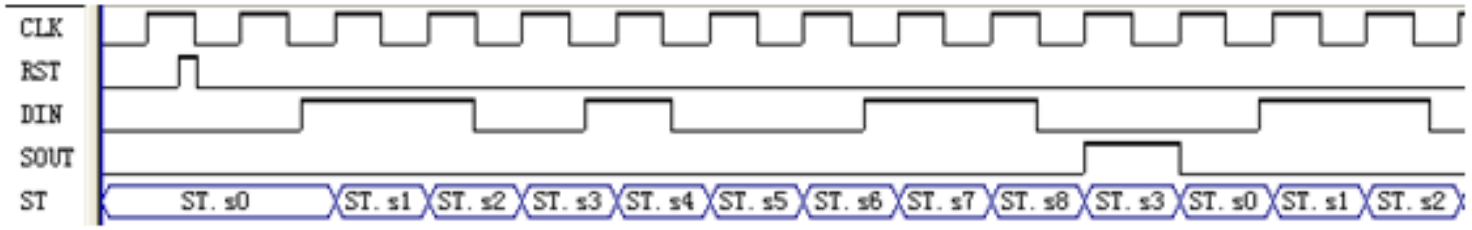

图 7-11 例 7-6 之单过程 Mealy 机仿真波形

7.4 不同编码类型状态机

7.4.1 直接输出型编码

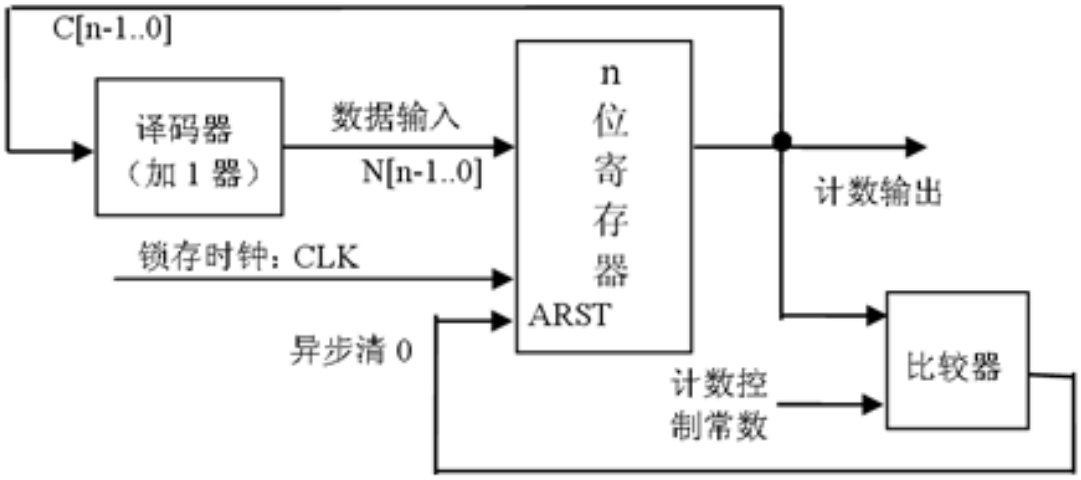

图 7-12 加法计数器一般模型

7.4 不同编码类型状态机

7.4.1 直接输出型编码

表 7-1 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 7-7】

```
module ADC0809  (D, CLK, EOC, RST, ALE, START, OE, ADDA, Q,LOCK_T);
   input[7:0] D;   input CLK,RST,EOC;    output[7:0] Q;
   output START,OE , ALE, ADDA,LOCK_T ;
   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
```

7.4 不同编码类型状态机

7.4.1 直接输出型编码

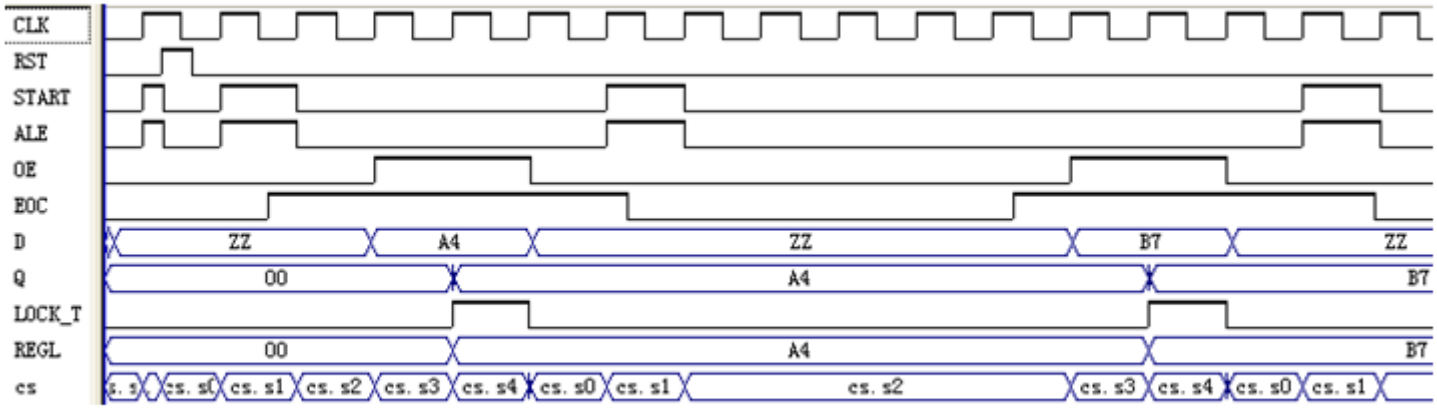

图 7-13 例 7-7 状态机工作时序图

■ 【例 7-8】

```
`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, 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
```

7.4 不同编码类型状态机

7.4.2 宏定义语句在状态编码定义中的用法

图 7-14 例 7-7 的状态编码波形图

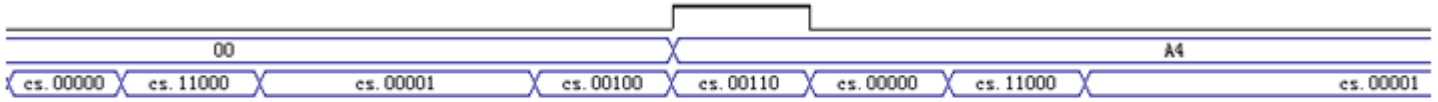

图 7-15 例 7-8 的状态编码波形图

7.4 不同编码类型状态机

7.4.3 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
```

7.4.4 顺序编码型状态机编码

表 7-2　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

7.4 不同编码类型状态机

7.4.5 一位热码编码

7.4.6 状态编码设置

1. 用户自定义方式

“User-Encoded”

2. 用属性定义语句设置

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

7.4 不同编码类型状态机

【例 7-9】

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

表 7-3　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

7.4 不同编码类型状态机

7.4.6 状态编码设置

3. 直接设置方法

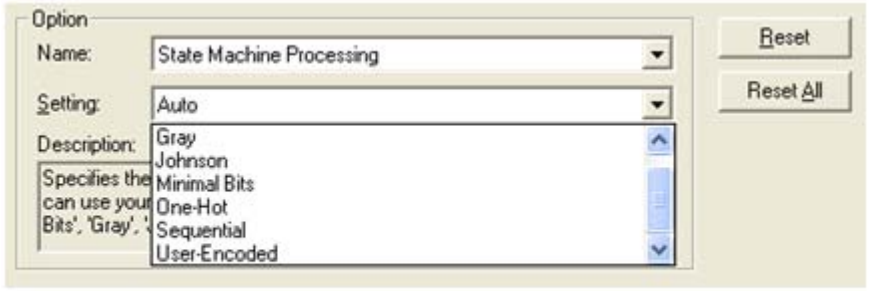

图 7-16 选择恰当的编码形式

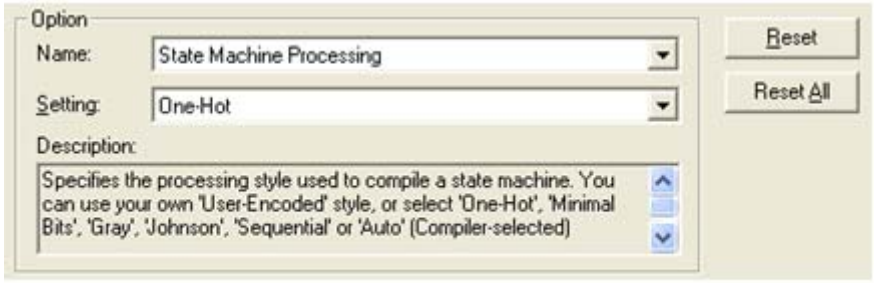

图 7-17 选择了一位热码编码形式

7.5 安全状态机设计

表 7-4　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

```
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 ;
```

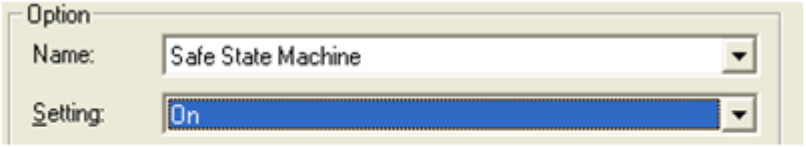

图 7-18 选择安全状态机设计

习　题

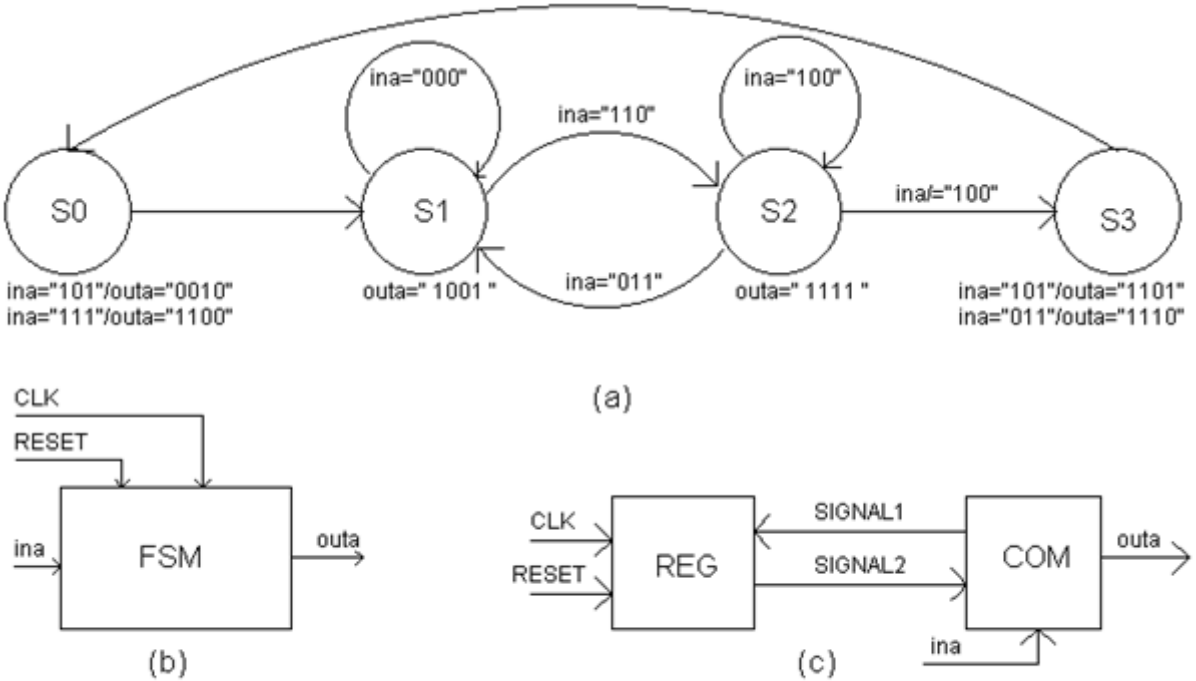

图 7-19　习题 7-7 状态图

EDA实验与创新实践

7-1 序列检测器设计

7-2 ADC采样控制电路设计

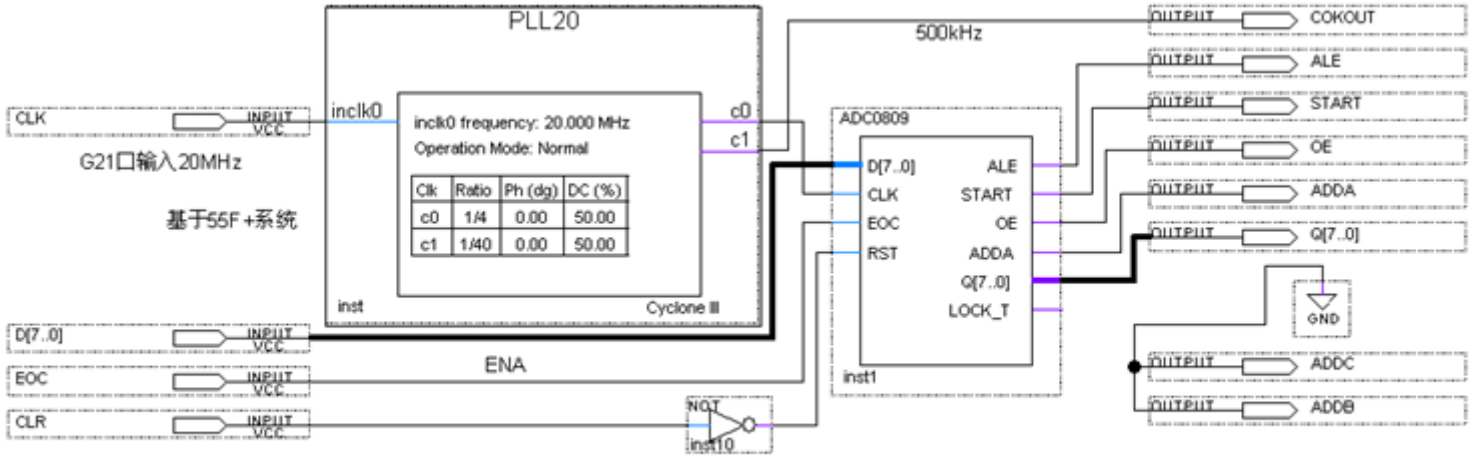

图 7-20 ADC0809 采样控制实验电路

EDA实验与创新实践

7-3 数据采集逻辑控制模块设计

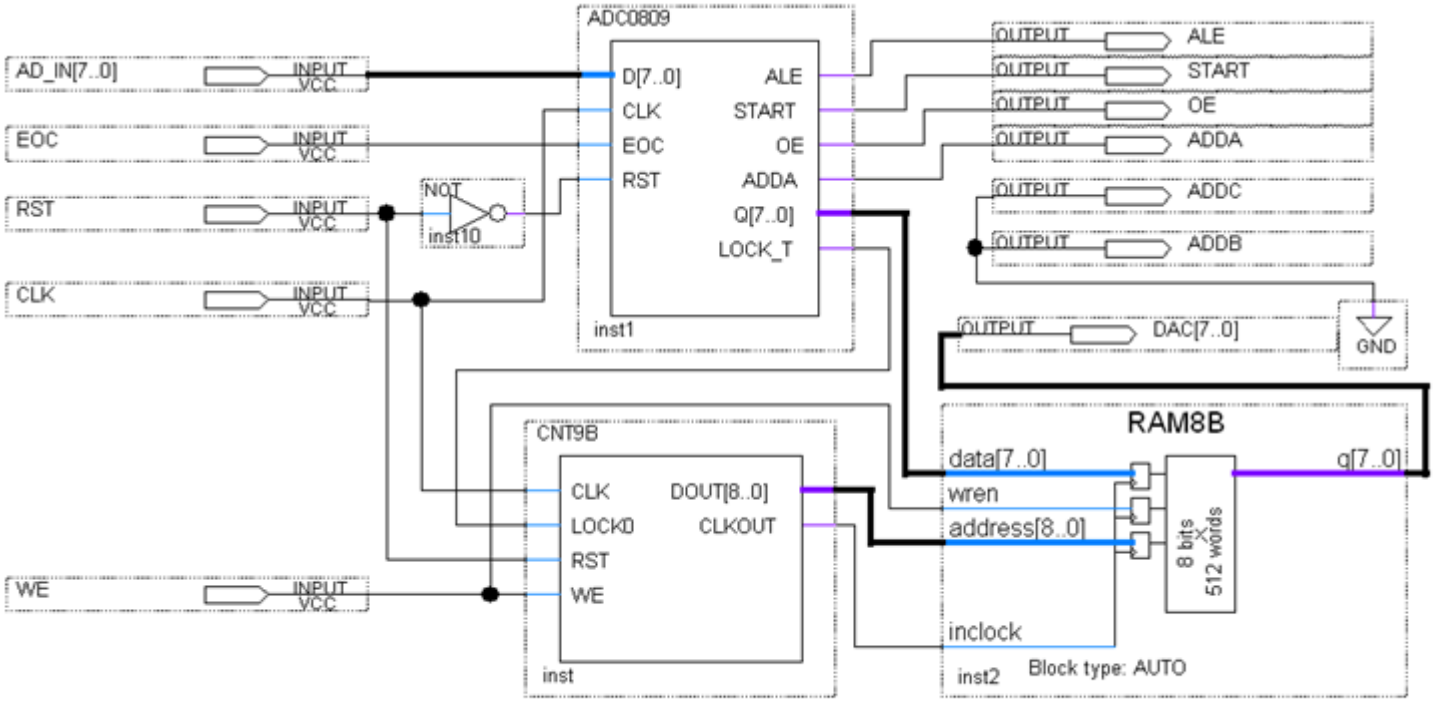

图 7-21　ADC0809 采样电路及简易存储示波器控制系统

EDA实验与创新实践

7-3 数据采集逻辑控制模块设计

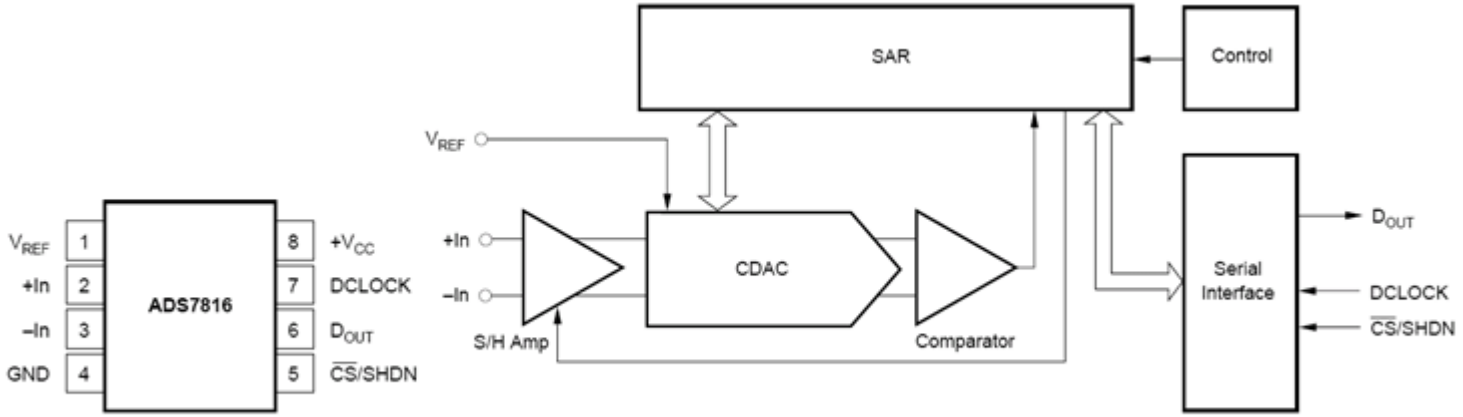

图 7-22　ADS7816 的引脚和结构

EDA实验与创新实践

7-3 数据采集逻辑控制模块设计

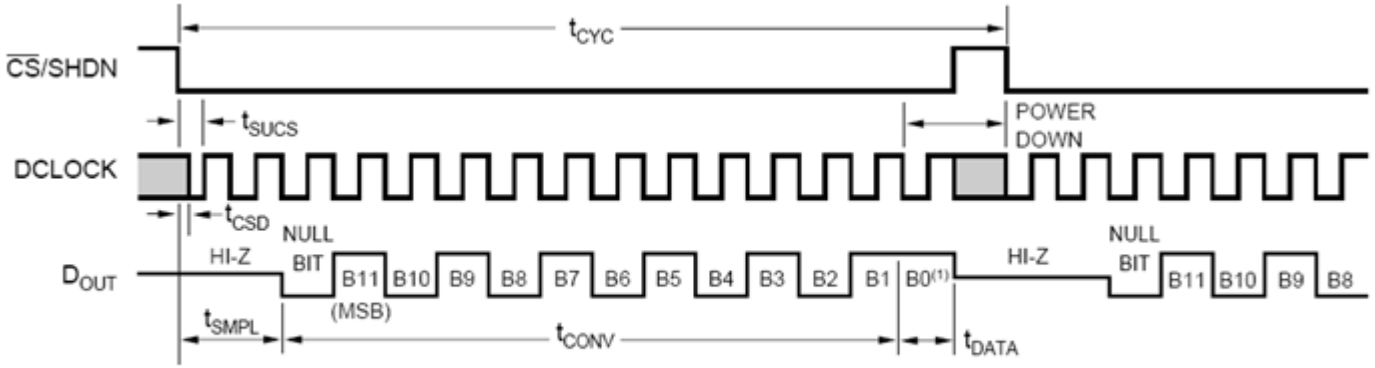

图7-23 ADS7816的工作时序

EDA实验与创新实践

7-4 五功能智能逻辑笔设计

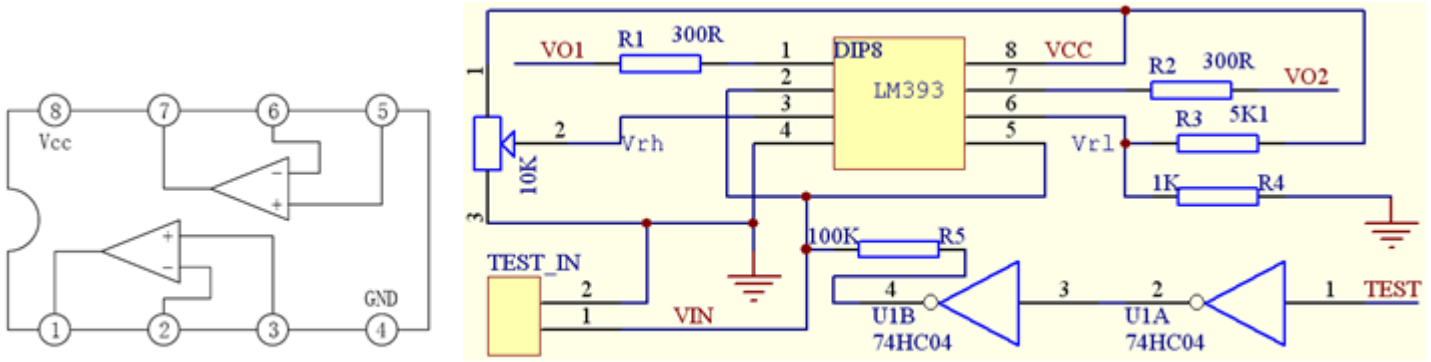

图 7-24 五功能智能逻辑笔电平信号采样电路，左图是 LM393 引脚图

【例 7-10】逻辑笔参考程序

```
module  LGC_PEN (CLK, VO,TEST,LED);
   input  CLK;          //状态机工作时钟
   input [2:1] VO;    //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 VO)   begin
      case (ST )
       s0: begin  TEST <=1'b1;  NST<=s1;  end
       s1: begin  TEST<=1'b1;  if (VO==2'b10)  NST<=s2; else NST<=s4; end
       s2: begin  TEST <=1'b0;   NST<=s3;  end
       s3: begin  TEST <=1'b0 ;
           begin if (VO==2'b01)  begin LED<=4'b1000; NST<=s0;  end
             else NST<=s4; end end
       s4: begin  if (VO==2'b01)  NST<=s5; else NST<=s7;   end
       s5: begin  if (VO==2'b01)  NST<=s6; else NST<=s7;   end
```

接下页

EDA实验与创新实践

7-4 五功能智能逻辑笔设计

```
s6: begin  if (VO==2'b01)  begin LED<=4'b0001; NST<=s0;  end
      else NST<=s7; end
s7: begin  if (VO==2'b10)  NST<=s8; else NST<=s10;   end
s8: begin  if (VO==2'b10)  NST<=s9; else NST<=s10;   end
s9: begin  if (VO==2'b10)  begin LED<=4'b0010; NST<=s0;  end
      else NST<=s10; end
s10: begin  if (VO==2'b11)  NST<=s11; else NST<=s13;   end
s11: begin  if (VO==2'b11)  NST<=s12; else NST<=s13;   end
s12: begin  if (VO==2'b11)  begin LED<=4'b0100; NST<=s0;  end
      else NST<=s13; end
s13: begin  LED<=4'b1111; NST<=s0;   end
 default :              NST<=s0;
endcase      end
endmodule
```

EDA实验与创新实践

7-5 VGA简单图像显示控制模块设计

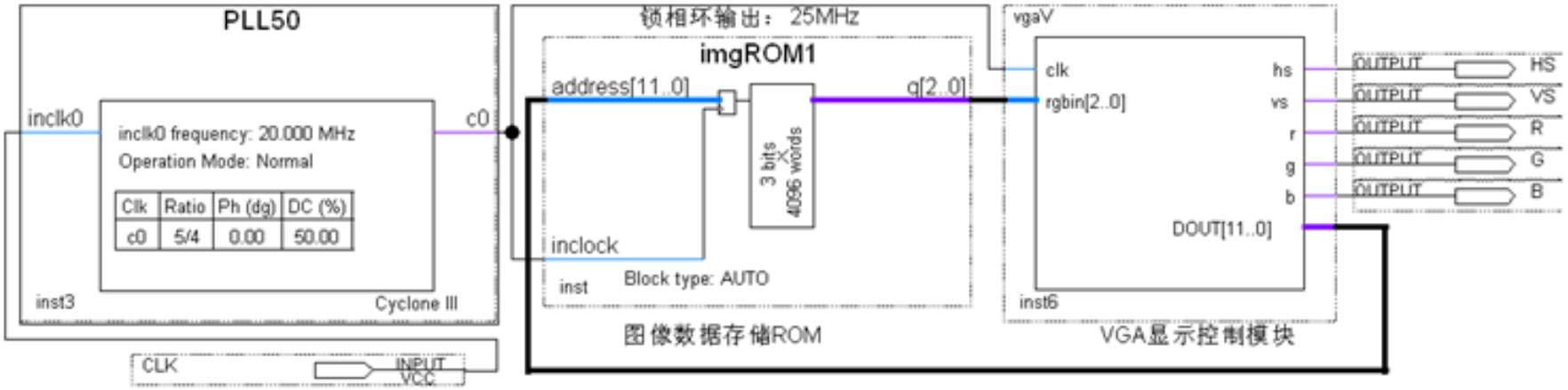

图 7-25 VGA 图像显示控制模块原理图

【例 7-11】

```
module vgaV (clk, hs, vs, r, g, b, rgbin, DOUT);
  input clk ;     //工作时钟 25MHz
   output hs,vs;   //场同步，行同步信号
  output r,g,b ;  // 红，绿，蓝信号，
  input[2:0] rgbin; //像素数据
  output[11:0] DOUT; //图像数据 ROM 的地址信号
  reg[9:0] hcnt, vcnt;   reg r,g,b;   reg hs,vs;
  assign DOUT = {vcnt[5:0], hcnt[5:0]} ;
  always @(posedge clk)   begin  //水平扫描计数器
     if (hcnt<800)   hcnt<=hcnt+1;   else   hcnt<={10{1'b0}};  end
  always @(posedge clk)    //垂直扫描计数器
     if (hcnt==640+8)    begin
       if (vcnt<525)  vcnt<=vcnt+1;  else  vcnt<={10{1'b0}};  end
 always @(posedge clk)   begin //场同步信号发生
     if ((hcnt>=640+8+8) & (hcnt<640+8+8+96))
       hs<=1'b0 ; else  hs<=1'b1 ;    end
 always @(vcnt)   begin   //行同步信号发生
     if ((vcnt>=480+8+2) & (vcnt<480+8+2+2))
        vs<=1'b0 ;  else  vs<=1'b1 ;  end
 always @(posedge clk)   begin
     if (hcnt<640 & vcnt<480)   //扫描终止
     begin  r<=rgbin[2] ;  g<=rgbin[1] ;  b<=rgbin[0];   end
     else begin  r<=1'b0; g<=1'b0;  b<=1'b0;  end
         end
 endmodule
```