# EDA技术实用教程

# 第7章

# VHDL有限状态机设计

# 7.1 VHDL状态机的一般形式

## 7.1.1 为什么要使用状态机

（1）高效的顺序控制模型。

（2）容易利用现成的EDA优化工具。

（3）性能稳定。

（4）设计实现效率高。

（5）高速性能。

（6）高可靠性能。

# 7.1 VHDL状态机的一般形式

## 7.1.2 一般有限状态机的结构

### 1. 说明部分

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

### 2. 主控时序进程

# 7.1 VHDL状态机的一般形式

## 7.1.2 一般有限状态机的结构

### 3. 主控组合进程

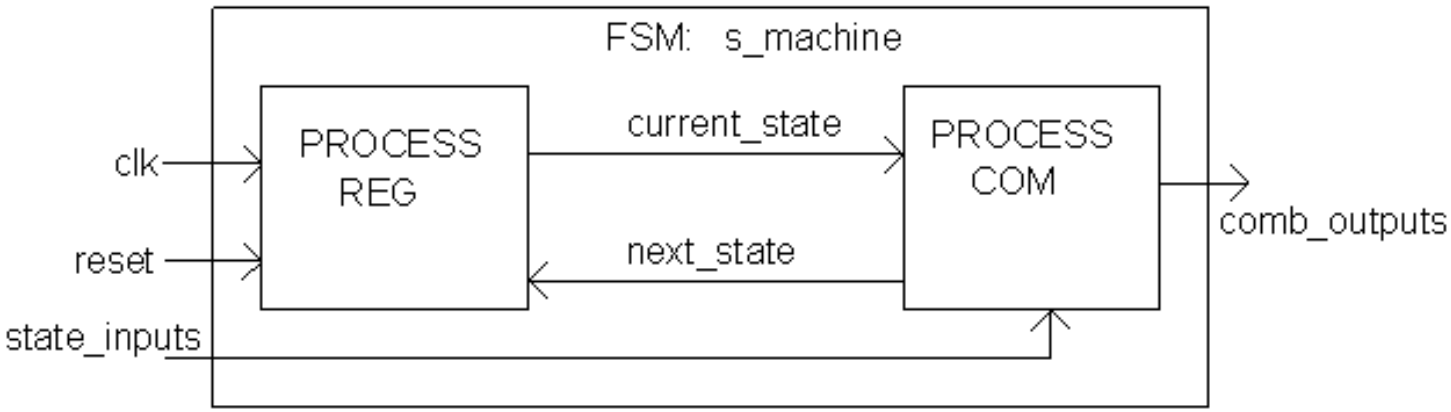


图 7-1 状态机一般结构示图

# 7.1 VHDL状态机的一般形式

**4.** 辅助进程

【例 7-1】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY FSM_EXP IS
PORT ( clk,reset    : IN STD_LOGIC;                    --状态机工作时钟和复位信号
 state_inputs : IN STD_LOGIC_VECTOR(0 TO 1); -- 来自外部的状态机控制信号
 comb_outputs : OUT INTEGER RANGE 0 TO 15 ); --状态机对外部发出的控制信号
END FSM_EXP;
ARCHITECTURE behv OF FSM_EXP IS
TYPE FSM_ST IS (s0, s1, s2, s3, S4);  --数据类型定义，定义状态符号
SIGNAL c_st, next_state: FSM_ST;--将现态和次态定义为新的数据类型 FSM_ST
BEGIN
 REG: PROCESS (reset,clk)   BEGIN        --主控时序进程
   IF reset='0' THEN   c_st<=s0;--检测异步复位信号，复位信号后回到初态 s0
    ELSIF clk='1' AND clk'EVENT THEN  c_st <= next_state;  END IF;
 END PROCESS REG;
```

接下页

接上页

```
COM:PROCESS(c_st, state_Inputs)   BEGIN  --主控组合进程
    CASE c_st IS
      WHEN s0 => comb_outputs<= 5;  --进入状态 s0 后输出 5
        IF state_inputs="00" THEN  next_state<=s0;
          ELSE  next_state<=s1;  END IF;
      WHEN s1 =>  comb_outputs<= 8;
        IF state_inputs="01" THEN  next_state<=s1;
          ELSE  next_state<=s2;   END IF;
      WHEN s2 =>   comb_outputs<= 12;
        IF state_inputs="10" THEN  next_state <= s0;
          ELSE  next_state <= s3;  END IF;
      WHEN s3 =>  comb_outputs <= 14;
        IF state_inputs="11" THEN  next_state <= s3;
          ELSE  next_state<=s4;     END IF;
      WHEN s4 =>  comb_outputs <= 9; next_state <= s0;
      WHEN OTHERS =>  next_state <= s0 ;
    END case;
  END PROCESS COM;
END behv;
```

# 7.1 VHDL状态机的一般形式

## 7.1.2 一般有限状态机的结构

### 4. 辅助进程

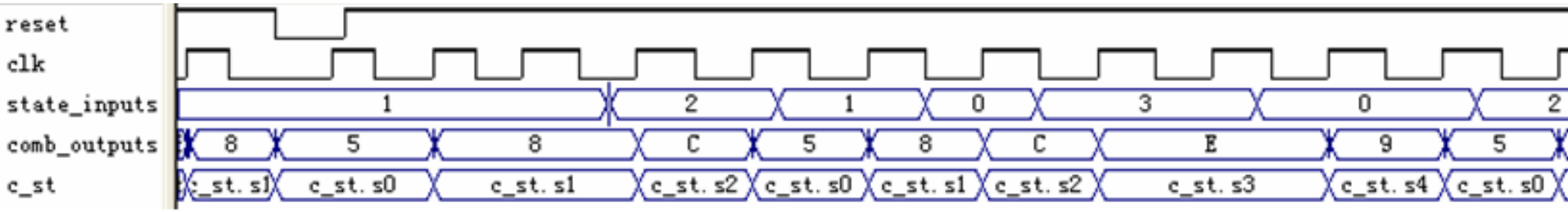


图 7-2　例 7-1 状态机的工作时序

# 7.1 VHDL状态机的一般形式

## 7.1.3 状态机设计初始控制与表述

（1）打开“状态机萃取”开关

（2）状态图观察

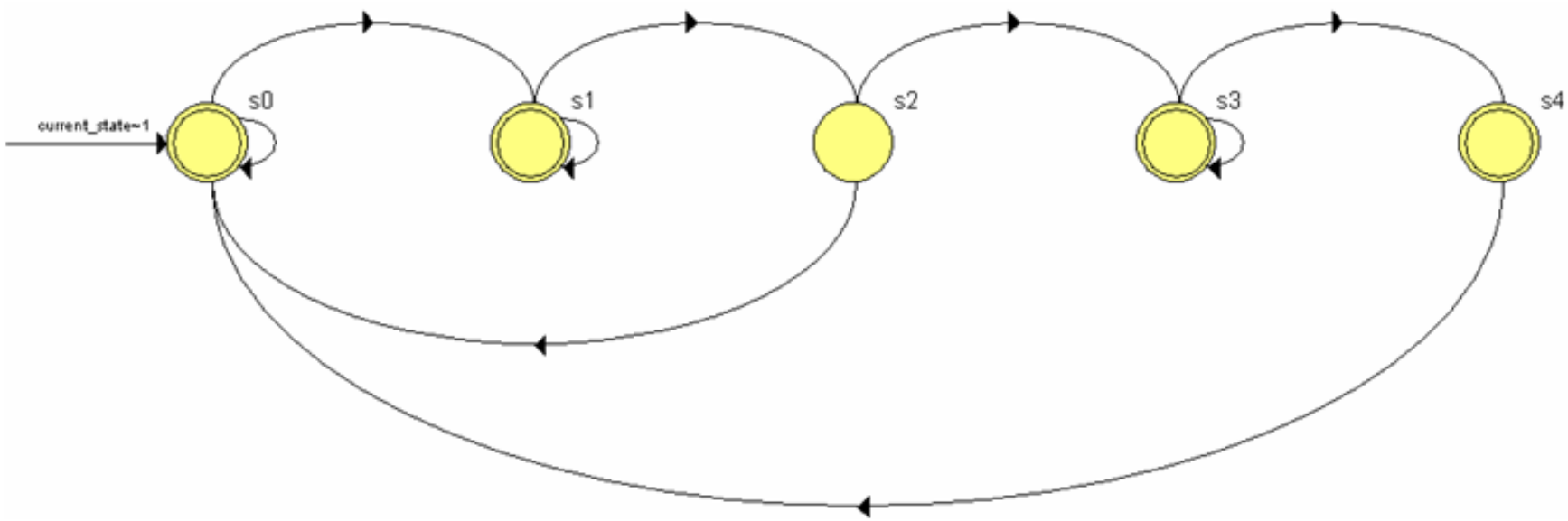


图 7-3　例 7-1 的状态图

# 7.2 Moore型有限状态机的设计

## 7.2.1 ADC采样控制设计及多进程结构状态机

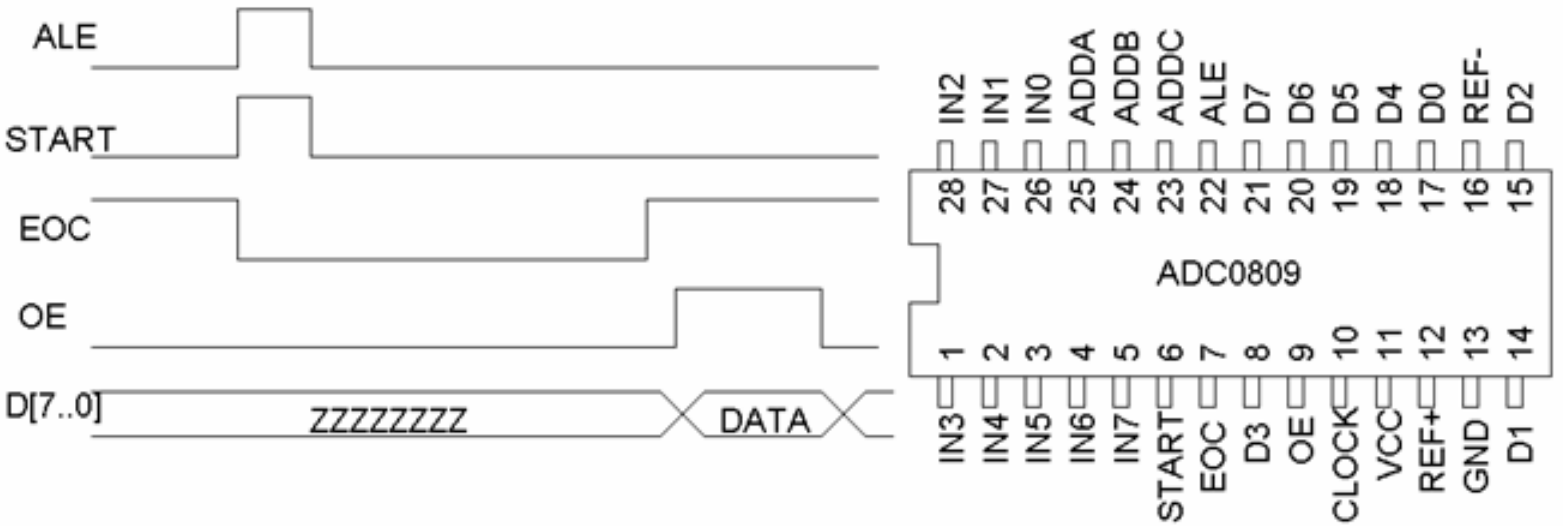


图 7-4　ADC0809 工作时序和芯片引脚图

# 7.2 Moore型有限状态机的设计

## 7.2.1 ADC采样控制设计及多进程结构状态机

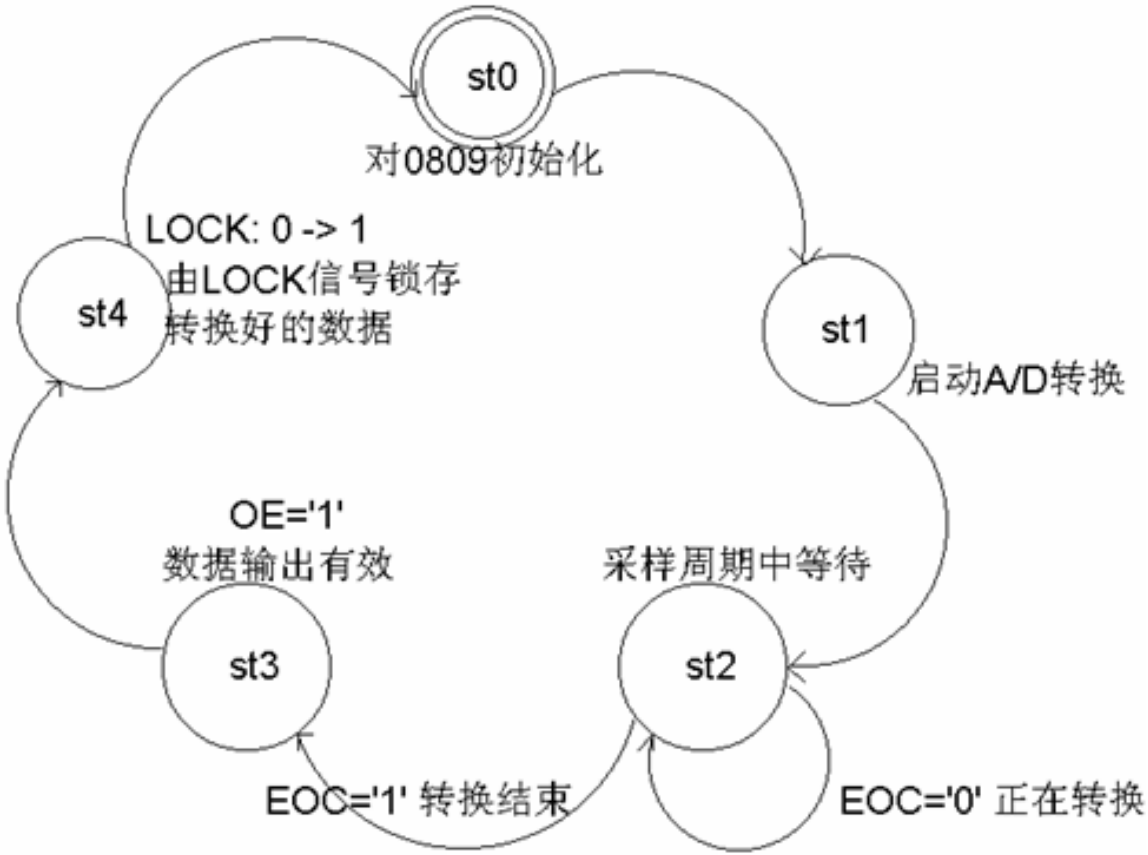


图 7-5 控制 ADC0809 采样状态图

# 7.2 Moore型有限状态机的设计

## 7.2.1 ADC采样控制设计及多进程结构状态机

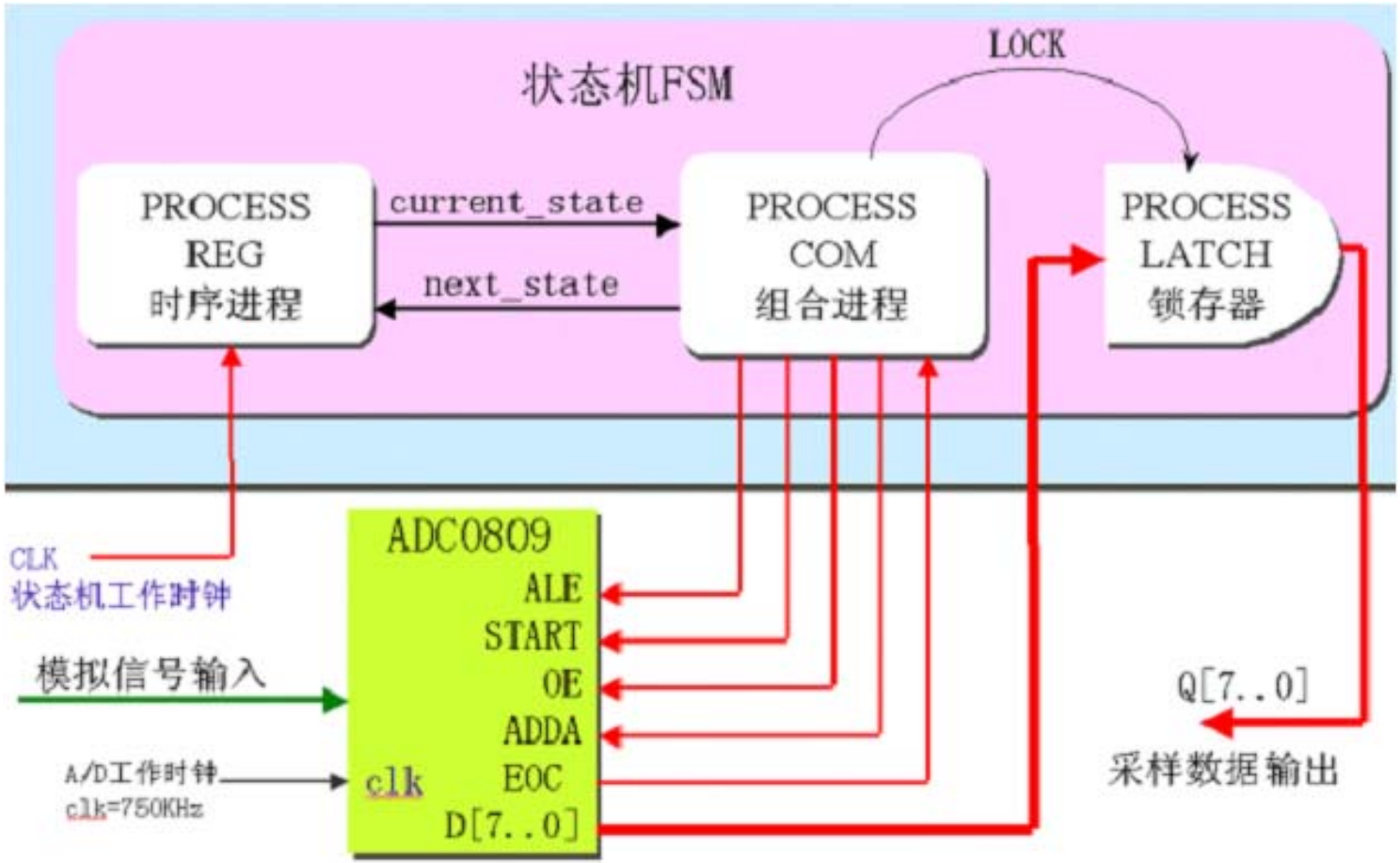


图 7-6 采样状态机结构框图

【例 7-2】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ADC0809 IS
    PORT (D  : IN STD_LOGIC_VECTOR(7 DOWNTO 0); --来自0809转换好的8位数据
          CLK ,RST : IN STD_LOGIC;    --状态机工作时钟和系统复位控制
EOC : IN STD_LOGIC;              --转换状态指示，低电平表示正在转换
ALE : OUT STD_LOGIC; --8个模拟信号通道地址锁存信号
START, OE  : OUT STD_LOGIC; --转换启动信号和数据输出三态控制信号
         ADDA, LOCK_T   : OUT STD_LOGIC;--信号通道控制信号和锁存测试信号
         Q  : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)  );
END ADC0809;
ARCHITECTURE behav OF ADC0809 IS
TYPE states IS (s0, s1, s2, s3,s4) ; --定义各状态
  SIGNAL cs, next_state: states :=s0 ;
  SIGNAL REGL      : STD_LOGIC_VECTOR(7 DOWNTO 0);
  SIGNAL LOCK      : STD_LOGIC;
 BEGIN
    ADDA <= '0';  LOCK_T<=LOCK;  --地址ADDA置0
```

接下页

接上页

```
COM:  PROCESS(cs,EOC)  BEGIN  --组合进程，规定各状态转换方式
  CASE cs IS
  WHEN s0 => ALE<='0';START<='0';OE<='0';LOCK<='0';next_state <= s1;
  WHEN s1=> ALE<='1';START<='1';OE<='0';LOCK<='0';next_state <= s2;
  WHEN s2=> ALE<='0';START<='0';OE<='0';LOCK<='0';
    IF (EOC='1') THEN next_state <= s3; --EOC=0表明转换结束
   ELSE next_state <= s2;  END IF ;              --转换结束未结束，继续等待
   WHEN s3=> ALE<='0';START<='0';OE<='1';LOCK<='0';next_state <= s4;
  WHEN s4=> ALE<='0';START<='0';OE<='1';LOCK<='1';next_state <= s0;
  WHEN OTHERS => ALE<='0';START<='0';OE<='0';LOCK<='0';next_state <= s0;
  END CASE ;
 END PROCESS COM ;
REG: PROCESS (CLK,RST)   BEGIN --时序进程
       IF RST='1'  THEN  cs <= s0;
       ELSIF   CLK'EVENT AND CLK='1'  THEN  cs <= next_state;   END IF;
  END PROCESS REG;
 LATCH1:  PROCESS (LOCK)  BEGIN --锁存器进程
       IF LOCK='1' AND LOCK'EVENT THEN   REGL <= D ;   END IF;
      END PROCESS LATCH1;
          Q <= REGL;
END behav;
```

# 7.2 Moore型有限状态机的设计

## 7.2.1 ADC采样控制设计及多进程结构状态机

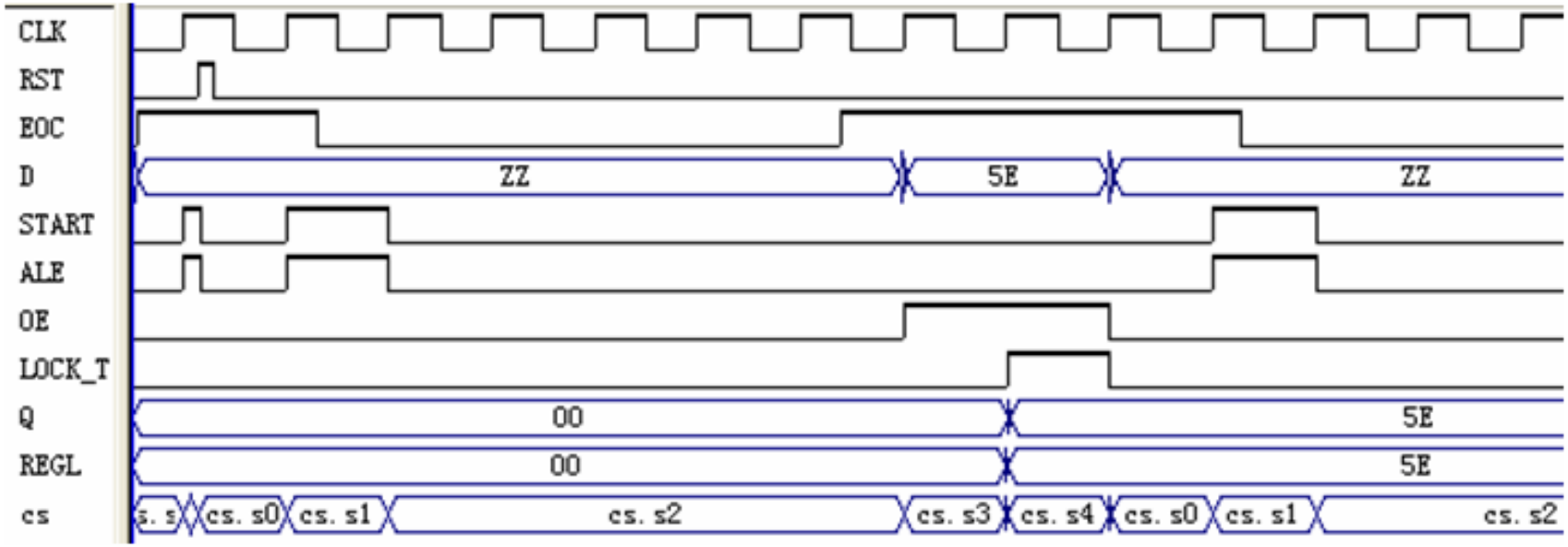


图 7-7　ADC0809 采样状态机工作时序

【例 7-3】

```
COM1: PROCESS(cs,EOC) BEGIN
  CASE cs IS
  WHEN s0=> next_state <= s1;
  WHEN s1=> next_state <= s2;
  WHEN s2=> IF (EOC='1')THEN next_state <= s3;
            ELSE next_state <= s2; END IF ;
  WHEN s3=> next_state <= s4;--开启 OE
  WHEN s4=> next_state <= s0;
  WHEN OTHERS => next_state <= s0;
  END CASE ;
END PROCESS COM1 ;
COM2: PROCESS (cs) BEGIN
  CASE cs IS
  WHEN s0=>ALE<='0';START<='0';LOCK<='0';OE<='0' ;
  WHEN s1=>ALE<='1';START<='1';LOCK<='0';OE<='0' ;
  WHEN s2=>ALE<='0';START<='0';LOCK<='0';OE<='0' ;
  WHEN s3=>ALE<='0';START<='0';LOCK<='0';OE<='1' ;
  WHEN s4=>ALE<='0';START<='0';LOCK<='1';OE<='1' ;
  WHEN OTHERS => ALE<='0';START<='0';LOCK<='0';
  END CASE ;
END PROCESS COM2 ;
```

# 7.2 Moore型有限状态机的设计

## 7.2.2 序列检测器之状态机设计

【例 7-4】检测数据 11010011，高位在前。

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY  SCHK IS
  PORT(DIN,CLK, RST  : IN STD_LOGIC; --串行输入数据位/工作时钟/复位信号
       SOUT : OUT STD_LOGIC); --检测结果输出
END SCHK;
ARCHITECTURE behav OF SCHK IS
TYPE states IS (s0, s1, s2, s3,s4, s5, s6, s7, s8) ; --定义各状态
      SIGNAL ST, NST: states :=s0 ; --设定现态变量和次态变量
     BEGIN
  COM:  PROCESS(ST, DIN)    BEGIN      --组合进程，规定各状态转换方式
CASE ST  IS  --11010011
```

接下页

# 7.2 Moore型有限状态机的设计

接上页

```
  WHEN s0=>  IF DIN = '1' THEN NST <= s1 ; ELSE NST<=s0 ; END IF ;
  WHEN s1=>  IF DIN = '1' THEN NST <= s2 ; ELSE NST<=s0 ; END IF ;
  WHEN s2=>  IF DIN = '0' THEN NST <= s3 ; ELSE NST<=s0 ; END IF ;
  WHEN s3=>  IF DIN = '1' THEN NST <= s4 ; ELSE NST<=s0 ; END IF ;
  WHEN s4=>  IF DIN = '0' THEN NST <= s5 ; ELSE NST<=s0 ; END IF ;
  WHEN s5=>  IF DIN = '0' THEN NST <= s6 ; ELSE NST<=s0 ; END IF ;
  WHEN s6=>  IF DIN = '1' THEN NST <= s7 ; ELSE NST<=s0 ; END IF ;
  WHEN s7=>  IF DIN = '1' THEN NST <= s8 ; ELSE NST<=s0 ; END IF ;
  WHEN s8=>  IF DIN = '0' THEN NST <= s3 ; ELSE NST<=s0 ; END IF ;
  WHEN OTHERS =>  NST<=s0 ;
  END CASE ;
  END PROCESS ;
  REG: PROCESS (CLK,RST)    BEGIN   --时序进程
    IF RST='1'  THEN  ST <= s0;
       ELSIF   CLK'EVENT AND CLK='1'  THEN  ST <= NST;   END IF;
     END PROCESS REG;
        SOUT  <=  '1' WHEN ST=s8 ELSE  '0' ;
END behav ;
```

# 7.2 Moore型有限状态机的设计

## 7.2.2 序列检测器之状态机设计

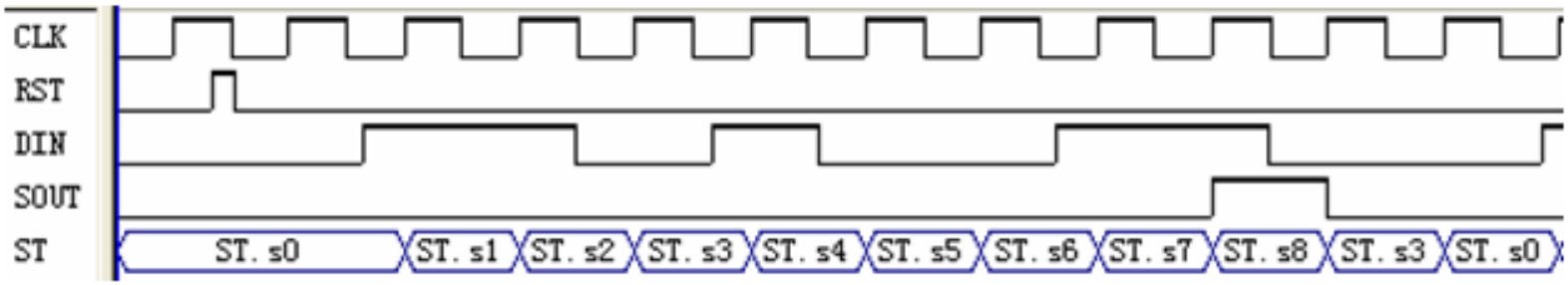


图 7-8　例 7-4 之序列检测器时序仿真波形

# 7.3 Mealy型有限状态机的设计

【例 7-5】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MEALY1 IS
PORT(CLK, DIN1,DIN2,RST : IN STD_LOGIC; --串行输入数据位/工作时钟/复位信号
          Q : OUT STD_LOGIC_VECTOR(4 DOWNTO 0)); --检测结果输出
END MEALY1;
ARCHITECTURE behav OF MEALY1 IS
TYPE states IS (st0, st1, st2, st3,st4) ; --定义各状态
SIGNAL PST : states ;
BEGIN
REGCOM: PROCESS(CLK,RST,PST, DIN1)    BEGIN
IF RST='1' THEN PST <=st0;   ELSIF  RISING_EDGE(CLK)  THEN
CASE PST  IS
```

接下页

# 7.3 Mealy型有限状态机的设计

接上页

```
    WHEN st0=>  IF DIN1='1' THEN PST<=st1 ; ELSE PST<=st0 ; END IF ;
    WHEN st1=>  IF DIN1='1' THEN PST<=st2 ; ELSE PST<=st1 ; END IF ;
    WHEN st2=>  IF DIN1='1' THEN PST<=st3 ; ELSE PST<=st2 ; END IF ;
    WHEN st3=>  IF DIN1='1' THEN PST<=st4 ; ELSE PST<=st3 ; END IF ;
    WHEN st4=>  IF DIN1='0' THEN PST<=st0 ; ELSE PST<=st4 ; END IF ;
    WHEN OTHERS =>  PST<=st0 ;
    END CASE ; END IF;
    END PROCESS REGCOM:;
  COM: PROCESS (PST,DIN2)   BEGIN
    CASE PST IS
    WHEN st0=>  IF DIN2='1' THEN Q<="10000" ; ELSE Q<="01010"; END IF ;
    WHEN st1=>  IF DIN2='0' THEN Q<="10111" ; ELSE Q<="10100"; END IF ;
    WHEN st2=>  IF DIN2='1' THEN Q<="10101" ; ELSE Q<="10011"; END IF ;
    WHEN st3=>  IF DIN2='0' THEN Q<="11011" ; ELSE Q<="01001"; END IF ;
    WHEN st4=>  IF DIN2='1' THEN Q<="11101" ; ELSE Q<="01101"; END IF ;
    WHEN OTHERS =>  Q<="00000" ;
    END CASE ;
    END PROCESS COM;
  END;
```

# 7.3 Mealy型有限状态机的设计

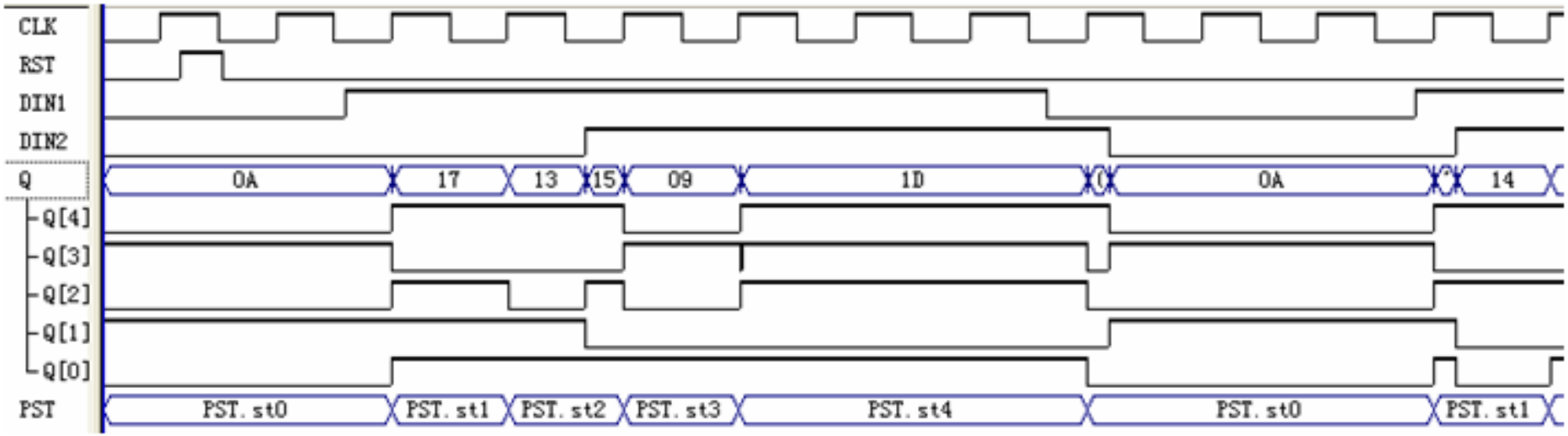


图 7-9　例 7-5 之双过程 Mealy 机仿真波形

# 7.3 Mealy型有限状态机的设计

【例 7-6】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MEALY2 IS
  PORT(CLK, DIN1,DIN2, RST : IN STD_LOGIC; --串行输入数据位/工作时钟/复位信号
       Q : OUT STD_LOGIC_VECTOR(4 DOWNTO 0)); --检测结果输出
END MEALY2;
ARCHITECTURE behav OF MEALY2 IS
TYPE states IS (st0, st1, st2, st3,st4) ; --定义各状态
SIGNAL PST : states ;
BEGIN
PROCESS(CLK,RST,PST, DIN1,DIN2)    BEGIN
IF RST='1' THEN PST <=st0;  ELSIF RISING_EDGE(CLK)   THEN
CASE PST  IS
  WHEN st0=>  IF DIN1='1' THEN PST <= st1 ; ELSE PST<=st0 ; END IF ;
        IF DIN2='1' THEN Q <= "10000" ; ELSE Q<="01010" ; END IF ;
  WHEN st1=>  IF DIN1='1' THEN PST <= st2 ; ELSE PST<=st1 ; END IF ;
        IF DIN2='0' THEN Q <= "10111" ; ELSE Q<="10100" ; END IF ;
```

接下页

# 7.3 Mealy型有限状态机的设计

接上页

```
  WHEN st2=>  IF DIN1='1' THEN PST <= st3 ; ELSE PST<=st2 ; END IF ;
        IF DIN2='1' THEN Q <= "10101" ; ELSE Q<="10011" ; END IF ;
  WHEN st3=>  IF DIN1='1' THEN PST <= st4 ; ELSE PST<=st3 ; END IF ;
        IF DIN2='0' THEN Q <= "11011" ; ELSE Q<="01001" ; END IF ;
  WHEN st4=>  IF DIN1='0' THEN PST <= st0 ; ELSE PST<=st4 ; END IF ;
        IF DIN2='1' THEN Q <= "11101" ; ELSE Q<="01101" ; END IF ;
  WHEN OTHERS =>  PST<=st0 ; Q<="00000" ;
  END CASE ;
 END IF;
  END PROCESS ;
END;
```

# 7.3 Mealy型有限状态机的设计

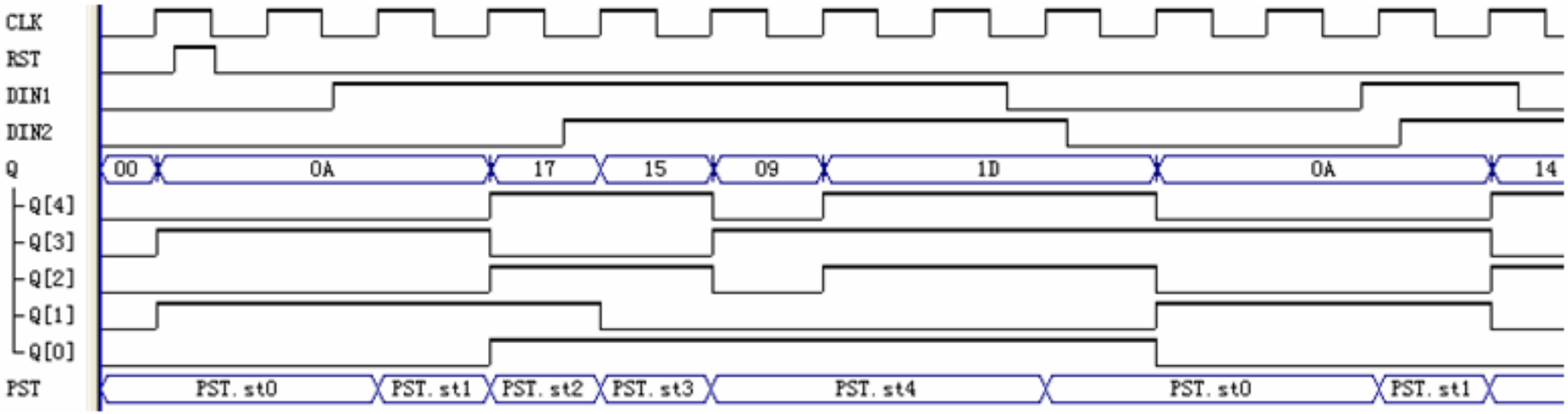


图 7-10　例 7-6 之单过程 Mealy 机仿真波形

# 7.3 Mealy型有限状态机的设计

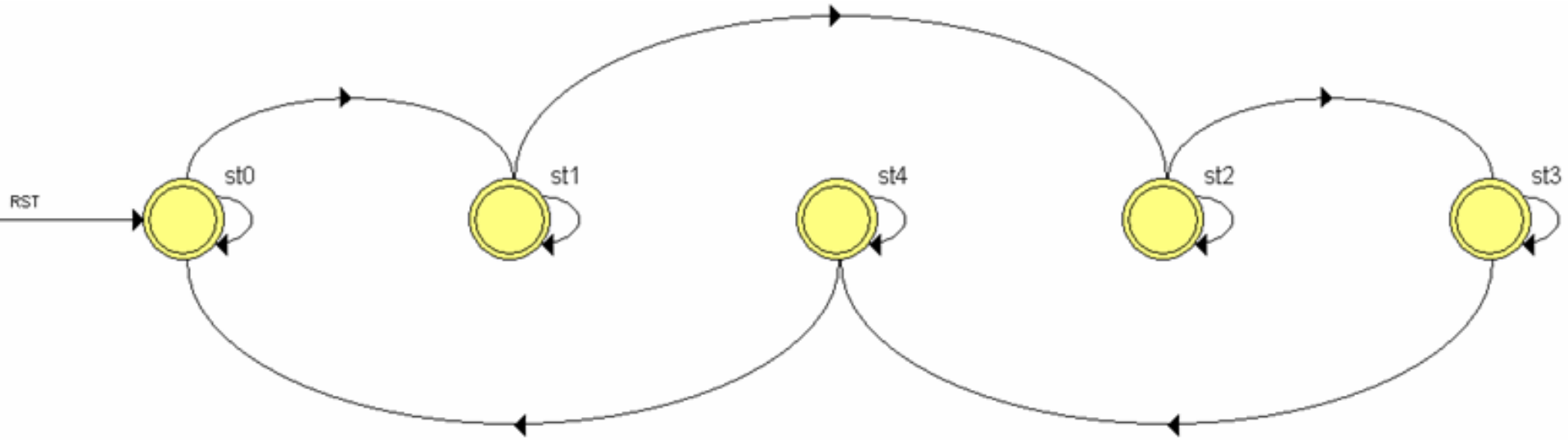


图 7-11　例 7-6，7-5 的状态图

# 7.3 Mealy型有限状态机的设计

【例 7-7】

```
LIBRARY IEEE ;
LIBRARY IEEE ; --11010011
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY  SCHK IS
  PORT(DIN,CLK, RST  : IN STD_LOGIC; --串行输入数据位/工作时钟/复位信号
         SOUT : OUT STD_LOGIC); --检测结果输出
END SCHK;
ARCHITECTURE behav OF SCHK IS
TYPE states IS (s0, s1, s2, s3,s4, s5, s6, s7, s8) ; --定义各状态
 SIGNAL ST : states :=s0 ;
      BEGIN
PROCESS(CLK,RST,ST, DIN)    BEGIN
 IF RST='1'  THEN  ST <= s0; ELSIF   CLK'EVENT AND CLK='1'  THEN
CASE ST  IS
  WHEN s0=>  IF DIN = '1' THEN ST <= s1 ; ELSE ST<=s0 ; END IF ;
  WHEN s1=>  IF DIN = '1' THEN ST <= s2 ; ELSE ST<=s0 ; END IF ;
  WHEN s2=>  IF DIN = '0' THEN ST <= s3 ; ELSE ST<=s0 ; END IF ;
```

接下页

# 7.3 Mealy型有限状态机的设计

接上页

```
  WHEN s3=>  IF DIN = '1' THEN ST <= s4 ; ELSE ST<=s0 ; END IF ;
  WHEN s4=>  IF DIN = '0' THEN ST <= s5 ; ELSE ST<=s0 ; END IF ;
  WHEN s5=>  IF DIN = '0' THEN ST <= s6 ; ELSE ST<=s0 ; END IF ;
  WHEN s6=>  IF DIN = '1' THEN ST <= s7 ; ELSE ST<=s0 ; END IF ;
  WHEN s7=>  IF DIN = '1' THEN ST <= s8 ; ELSE ST<=s0 ; END IF ;
  WHEN s8=>  IF DIN = '0' THEN ST <= s3 ; ELSE ST<=s0 ; END IF ;
  WHEN OTHERS =>  ST<=s0 ;
  END CASE ;
  IF (ST=s8)  THEN  SOUT<='1'; ELSE  SOUT<='0';   END IF;  END IF;
 END PROCESS;
END behav ;
```

# 7.3 Mealy型有限状态机的设计

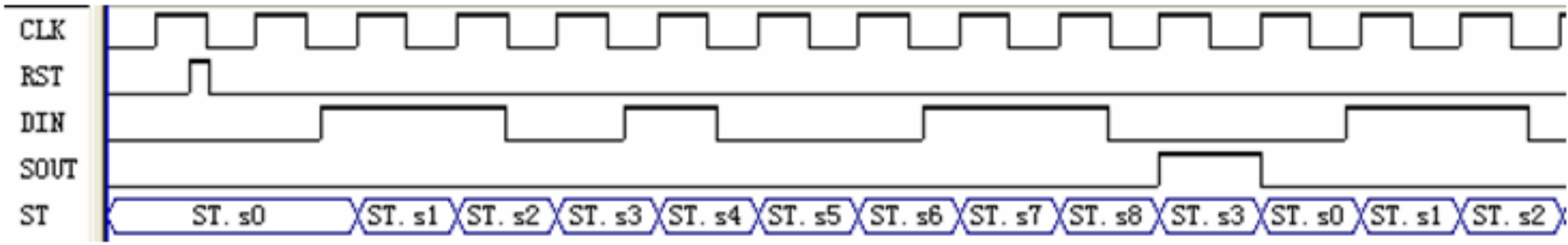


图 7-12 例 7-7 之单进程 Mealy 机仿真波形

# 7.4 状态机图形编辑设计方法

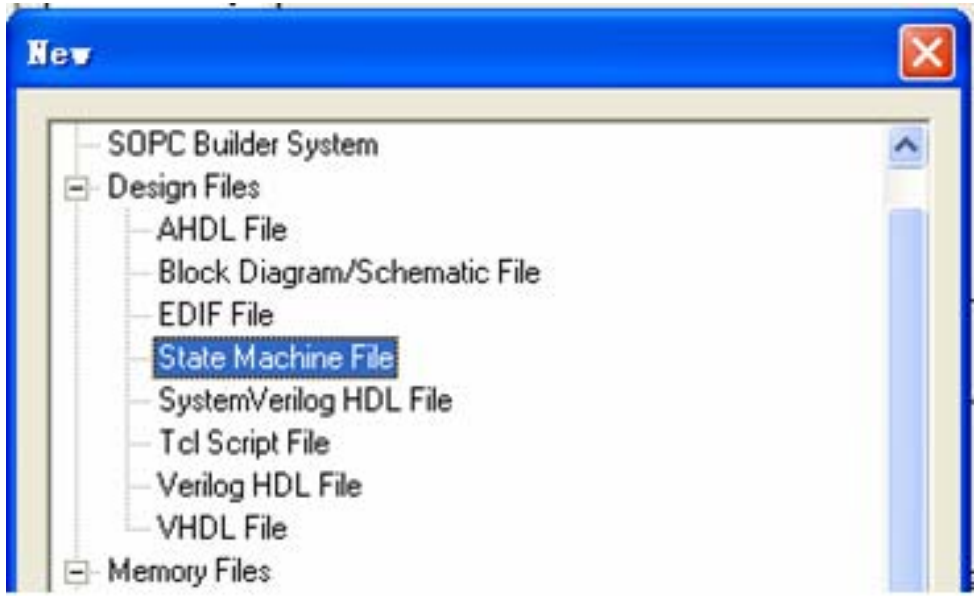


图 7-13 打开状态机图形编辑窗

# 7.4 状态机图形编辑设计方法

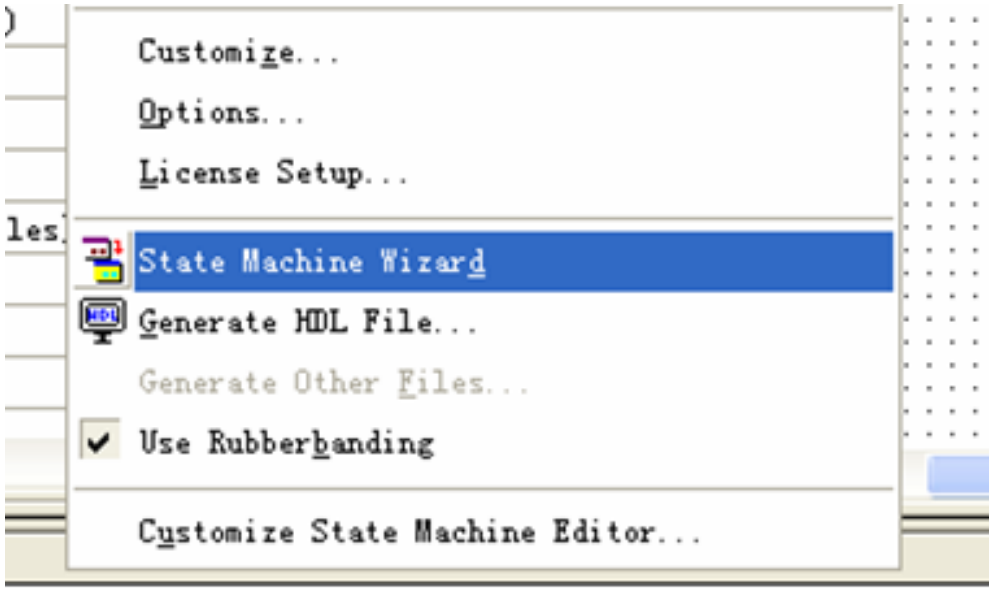


图 7-14 打开状态机编辑器

# 7.4 状态机图形编辑设计方法

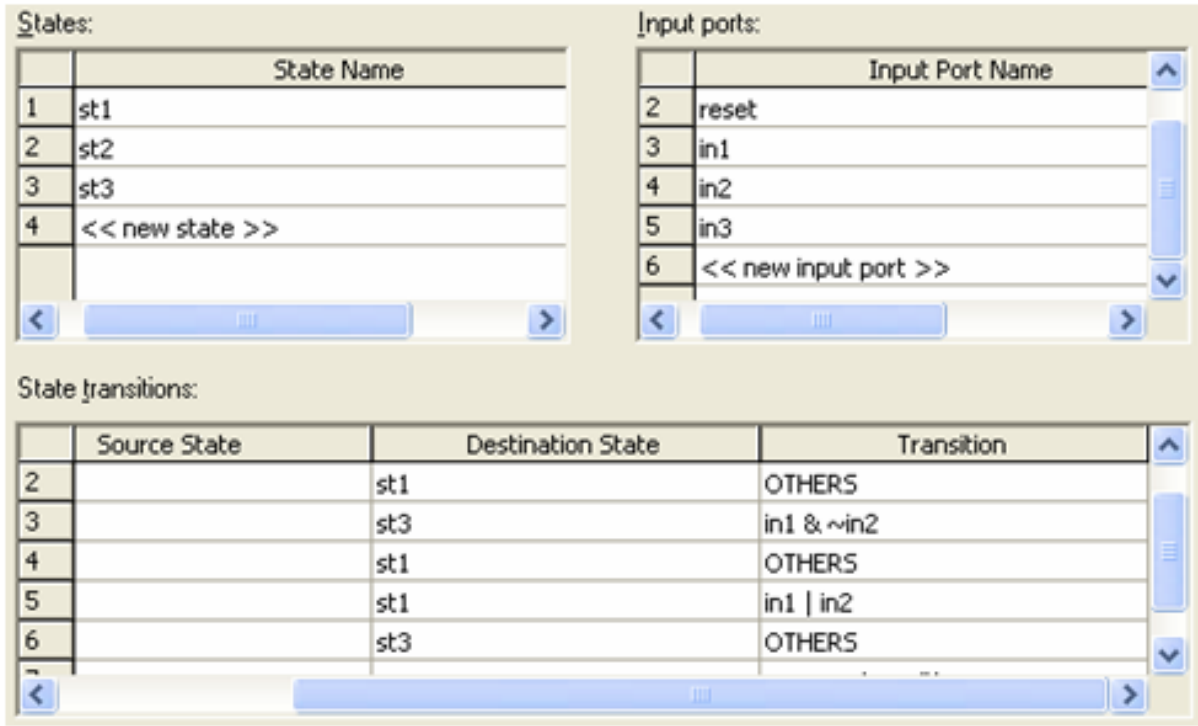


图 7-15 设置状态变量和状态转换条件

# 7.4 状态机图形编辑设计方法

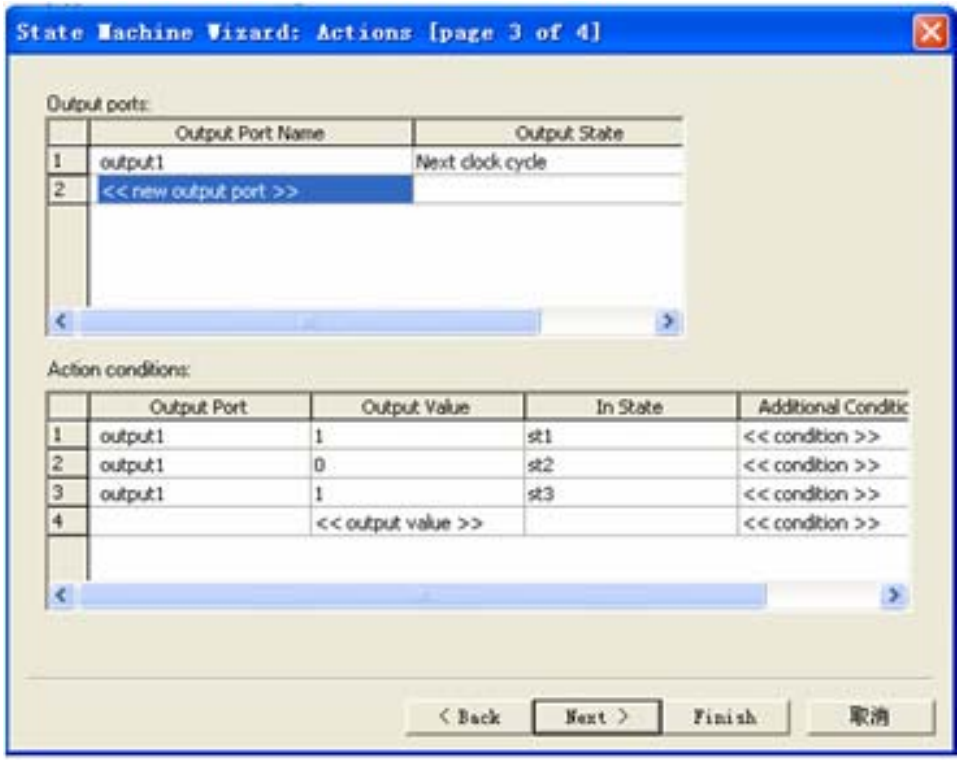


图 7-16 设置状态机输出

# 7.4 状态机图形编辑设计方法

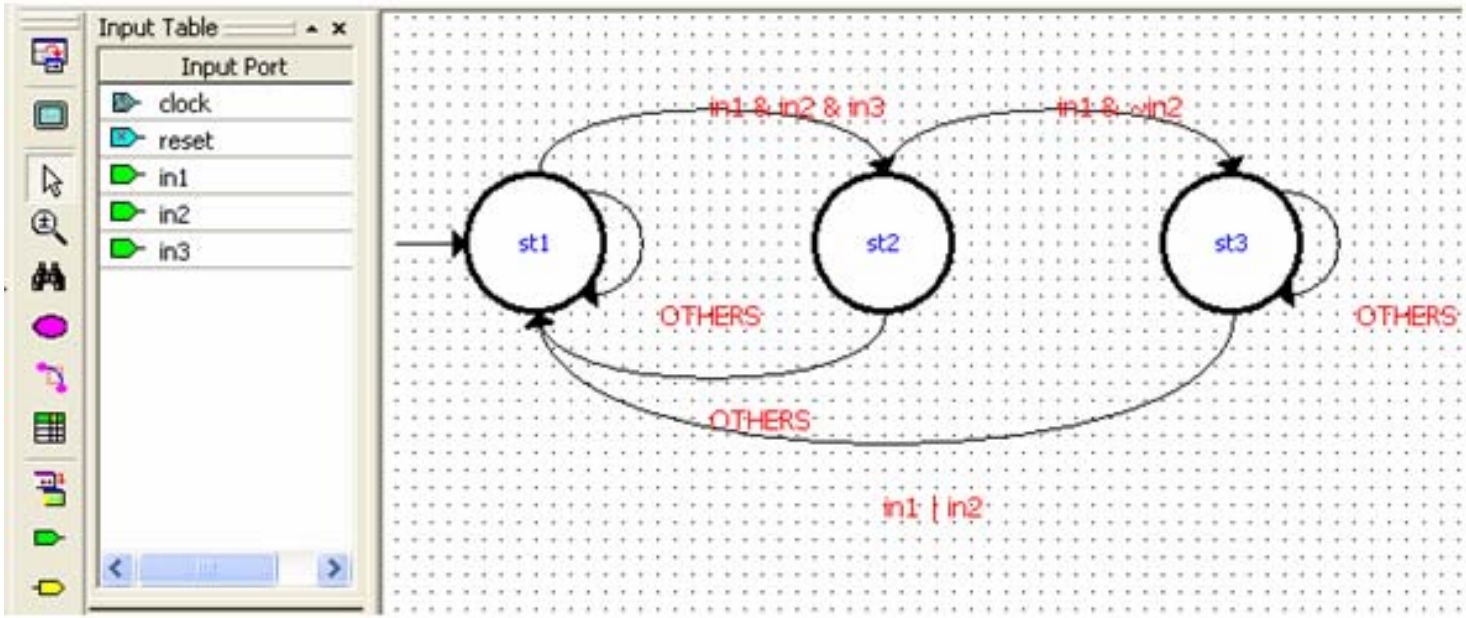


图 7-17 状态机图形编辑器上的状态图

# 7.4 状态机图形编辑设计方法

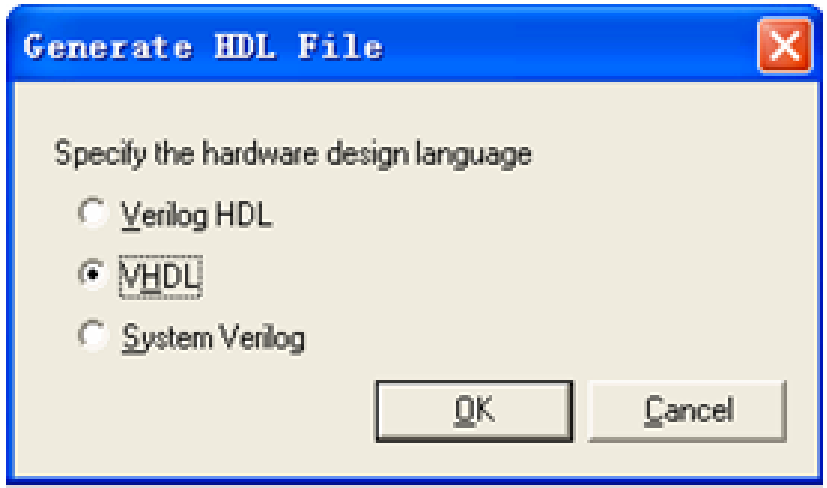


图 7-18 状态机转变成语言

# 7.5 状 态 编 码

## 7.5.1 直接输出型编码

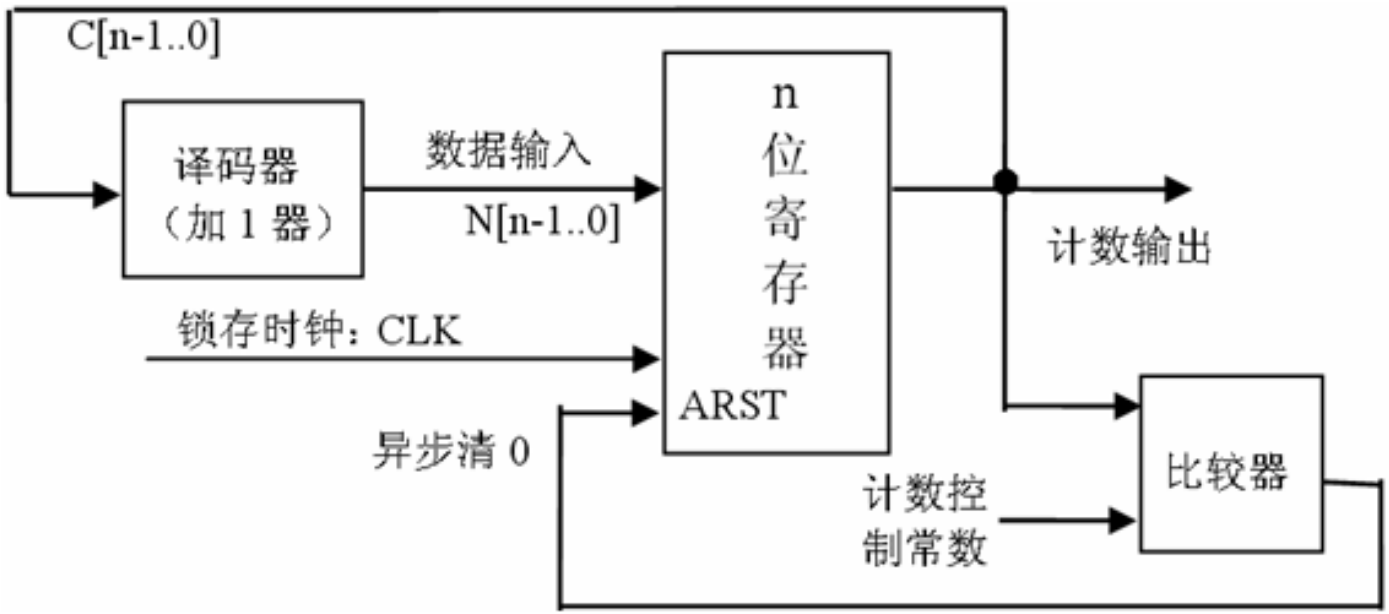


图 7-19 加法计数器一般模型

# 7.5 状 态 编 码

## 7.5.1 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 7-8】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ADC0809 IS
    PORT (D  : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
          CLK ,RST,EOC  : IN STD_LOGIC;
       ALE,START, OE ,ADDA, LOCK_T  : OUT STD_LOGIC;
         Q  : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)  );
END ADC0809;
ARCHITECTURE behav OF ADC0809 IS
 SIGNAL cs ,SOUT: STD_LOGIC_VECTOR(4 DOWNTO 0);
  CONSTANT s0 : STD_LOGIC_VECTOR(4 DOWNTO 0):= "00000" ;
  CONSTANT s1 : STD_LOGIC_VECTOR(4 DOWNTO 0):= "11000" ;
  CONSTANT s2 : STD_LOGIC_VECTOR(4 DOWNTO 0):= "00001" ;
  CONSTANT s3 : STD_LOGIC_VECTOR(4 DOWNTO 0):= "00100" ;
  CONSTANT s4 : STD_LOGIC_VECTOR(4 DOWNTO 0):= "00110" ;
  SIGNAL   REGL  : STD_LOGIC_VECTOR(7 DOWNTO 0);
  BEGIN
     Q <= REGL;  ADDA <= '0';
```

接下页

接上页

```
PROCESS(cs,EOC)
  BEGIN
    IF RST='1'  THEN  cs<=s0;  ELSIF  CLK'EVENT AND CLK='1'   THEN
   CASE cs IS
   WHEN s0 => cs <= s1 ;  SOUT<=s0;
   WHEN s1 => cs <= s2 ;  SOUT<=s1 ;
   WHEN s2 => SOUT<=s2 ; IF (EOC='1') THEN cs <= s3;
      ELSE cs <= s2;  END IF ;
   WHEN s3 => cs <= s4 ;  SOUT<=s3;
   WHEN s4 => cs <= s0 ;  SOUT<=s4;
   WHEN OTHERS => cs <= s0; SOUT<=s0;
   END CASE ; END IF;
 END PROCESS ;
  LATCH1:  PROCESS (SOUT(1),D) BEGIN
       IF SOUT(1)='1' AND SOUT(1)'EVENT THEN REGL<=D;  END IF;
   END PROCESS LATCH1;
 LOCK_T<=SOUT(1);  START<=SOUT(4); ALE<=SOUT(3);  OE<=SOUT(2) ;
END behav;
```

# 7.5 状态编码

## 7.5.1 直接输出型编码

【例 7-9】

```
ARCHITECTURE behav OF ADC0809 IS
type STAT is (s0,s1,s2,s3,s4);
attribute enum_encoding : string;
attribute enum_encoding of STAT : type is "00000 11000 00001 00100 00110";
  SIGNAL cs, next_state: STAT ;
```

# 7.5 状 态 编 码

## 7.5.2 顺序编码

## 7.5.3 一位热码状态编码

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 |

# 7.5 状 态 编 码

## 7.5.4 状态编码设置

**1.** 用户自定义方式

**2.** 直接设置方法

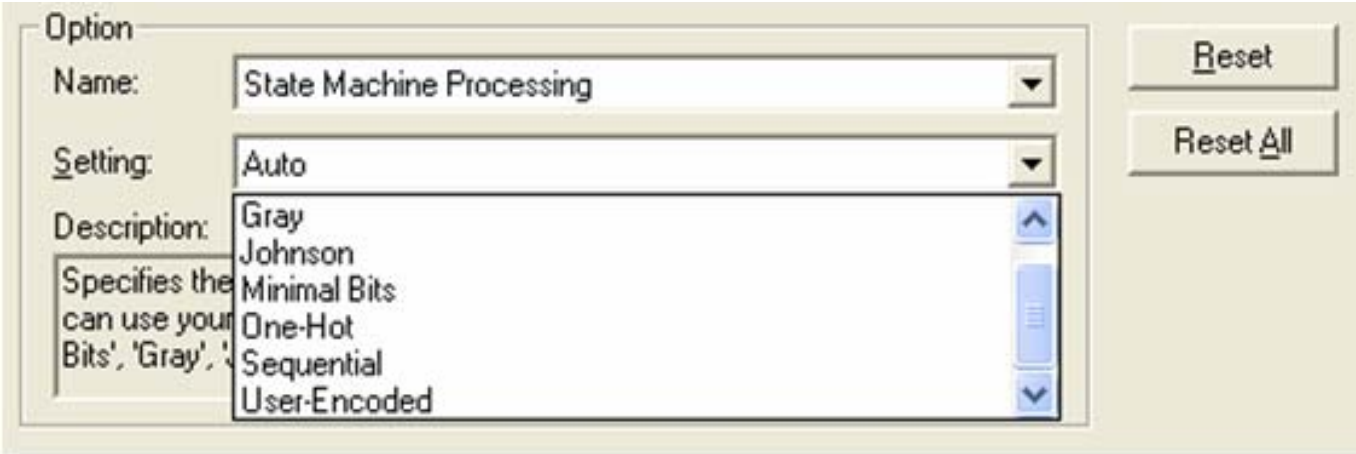


图 7-20 选择恰当的编码形式

# 7.5 状 态 编 码

## 7.5.4 状态编码设置

### 2. 直接设置方法

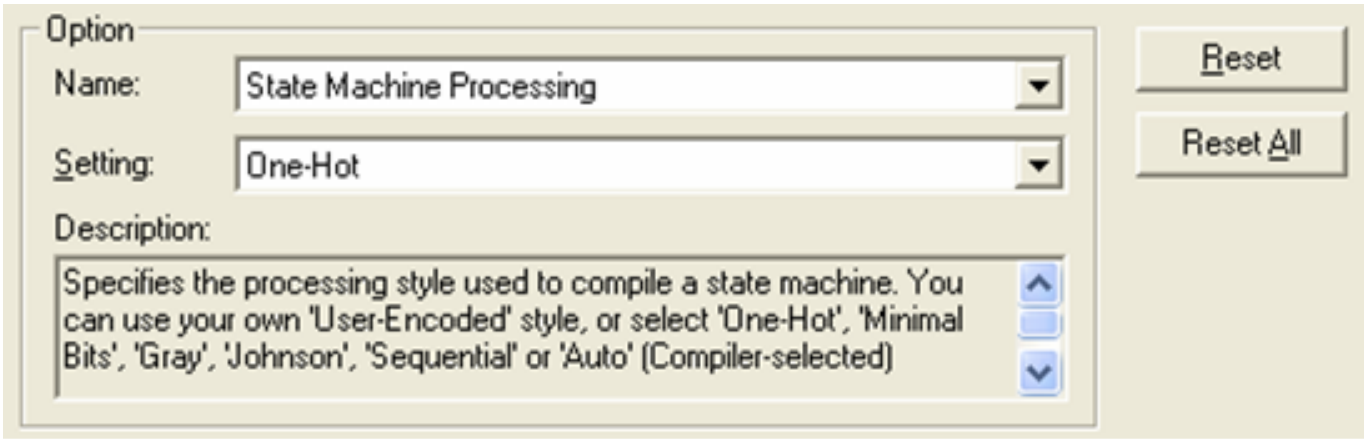


图 7-21 选择了一位热码编码形式

# 7.5 状 态 编 码

## 7.5.4 状态编码设置

### 3. 用属性定义语句设置

【例 7-10】

```
ARCHITECTURE behav OF SCHK IS
 TYPE states IS (s0, s1, s2, s3,s4, s5, s6, s7, s8) ;
 attribute enum_encoding : string;
 attribute enum_encoding of states : type is "one-hot";
 SIGNAL ST : states :=s0 ;
 BEGIN
```

# 7.5 状 态 编 码

## 7.5.4 状态编码设置

### 3. 用属性定义语句设置

**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 | type is "one-hot" | 11 | 10 |
| 用户自定义码 | type is "user" | 12 | 5 |
| 格雷码 | type is "gray" | 8 | 5 |
| 顺序码 | type is "sequential" | 10 | 5 |
| 约翰逊码 | type is "johnson" | 23 | 6 |
| 默认编码 | type is "default" | 11 | 10 |
| 最简码 | type is "compact" | 9 | 5 |
| 安全一位热码 | type is "safe, one-hot" | 18 | 10 |

# 7.6 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 |

# 7.6 安全状态机设计

## 7.6.1 程序直接导引法

```
TYPE states IS (s0, s1, s2, s3,s4, s5, s6, s7) ; --定义所有状态
    ...
 WHEN s5 =>   next_state<=s0;
 WHEN s6 =>   next_state<=s0;
 WHEN s7 =>   next_state<=s0;
 WHEN OTHERS =>   next_state<=s0;
```

# 7.6 安全状态机设计

## 7.6.2 状态编码监测法

alarm <= (st0 AND (st1 OR st2 OR st3 OR st4 OR st5)) OR (st1 AND (st0 OR st2 OR...

## 7.6.3 借助EDA优化控制工具生成安全状态机

```
attribute enum_encoding of states : type is "safe,one-hot";
```

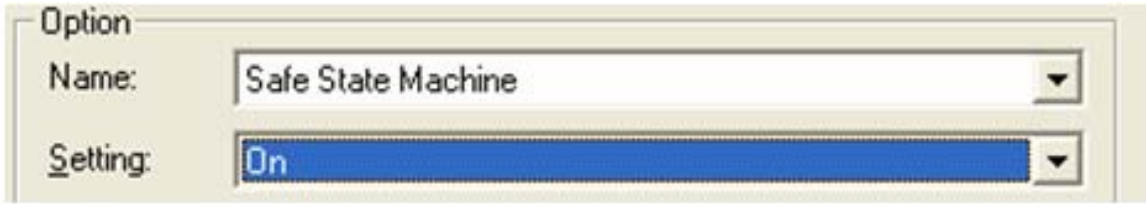


图 7-22 选择安全状态机设计

# 7.7 硬件数字技术排除毛刺

## 7.7.1 延时方式去毛刺

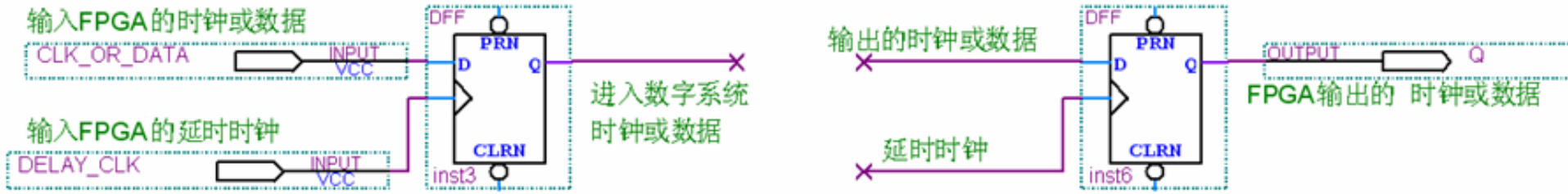


图 7-23 单触发器输入(左图)或输出(右图)延时电路

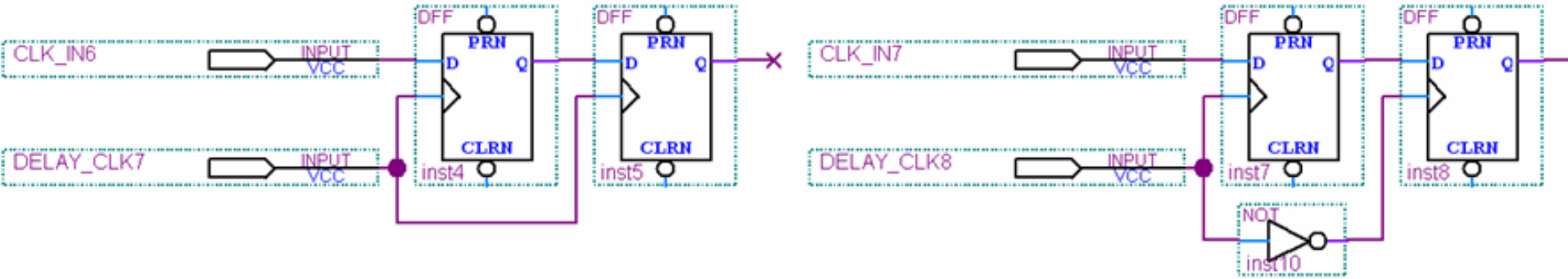


图 7-24 双触发器延时电路

# 7.7 硬件数字技术排除毛刺

## 7.7.1 延时方式去毛刺

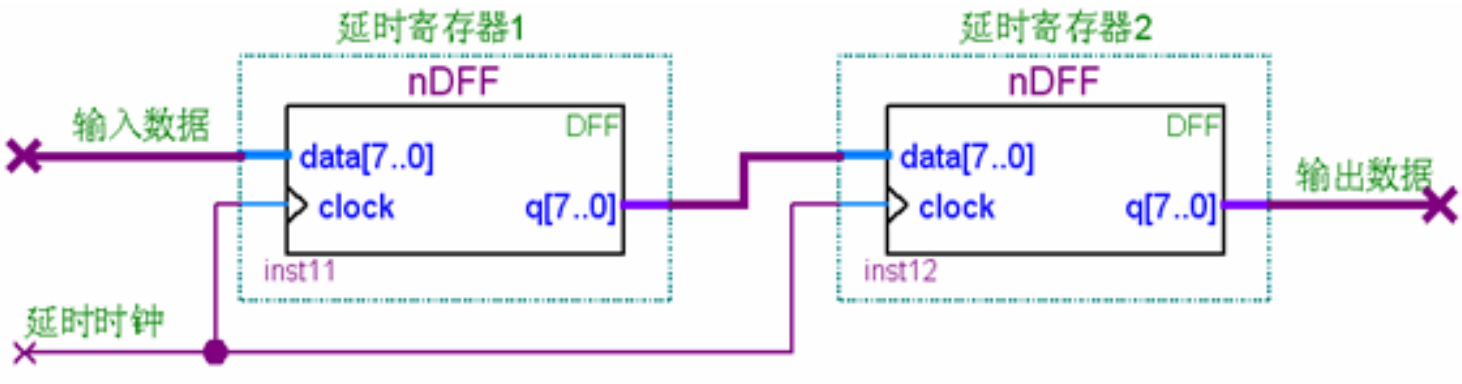


图 7-25 双寄存器数据延时电路

# 7.7 硬件数字技术排除毛刺

## 7.7.2 逻辑方式去毛刺


图 7-26 在信号上升与下降都含随机干扰抖动信号的时钟信号

# 7.7 硬件数字技术排除毛刺

## 7.7.2 逻辑方式去毛刺

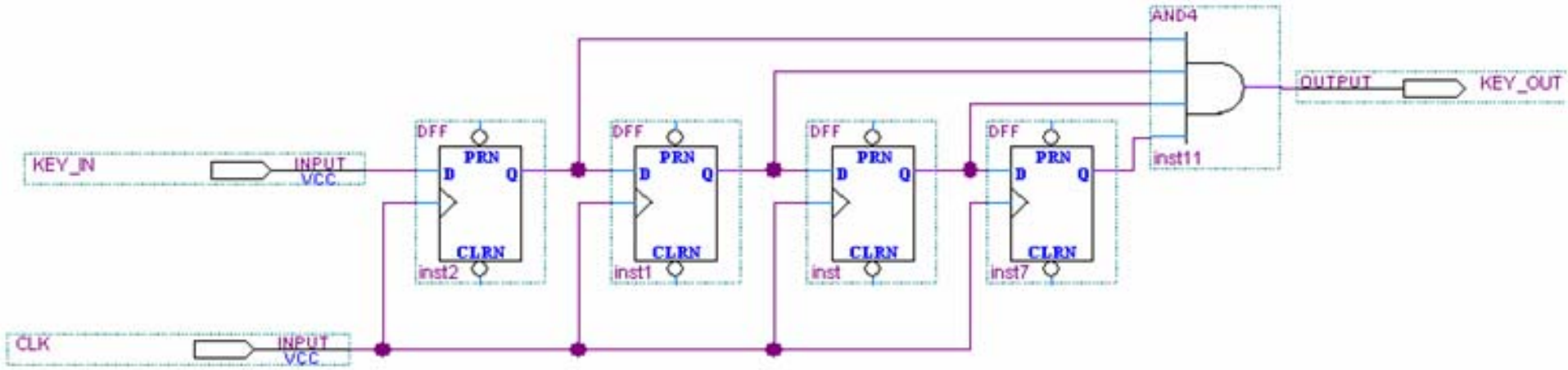


图 7-27 消抖动电路

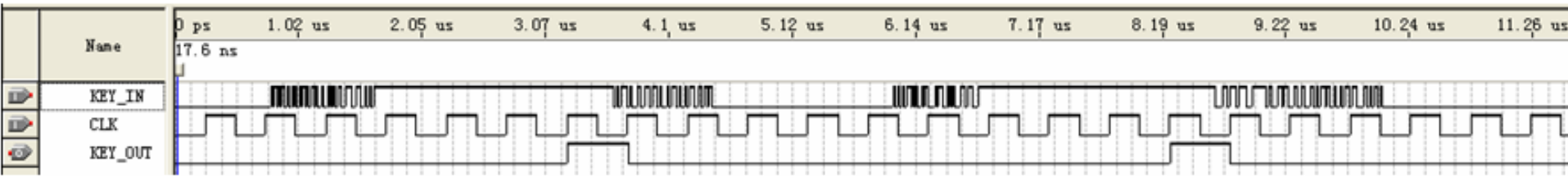


图 7-28 消抖动电路仿真波形

# 7.7 硬件数字技术排除毛刺

## 7.7.3 定时方式去毛刺

【例 7-11】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY  TRM0 IS
  PORT(CLK, KIN  : IN STD_LOGIC; --工作时钟和输入信号
           KOUT : OUT STD_LOGIC);
END TRM0;
ARCHITECTURE behav OF TRM0 IS --定义对高电平和低电平脉宽计数之寄存器。
SIGNAL KH,KL : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
PROCESS(CLK,KIN)   BEGIN
  IF RISING_EDGE(CLK) THEN
     IF (KIN='0') THEN  KL<=KL+1 ;--对键输入的低电平脉宽计数
        ELSE  KL<="0000";  END IF;  END IF;--若出现高电平，则计数器清 0
END PROCESS;
```

接下页

# 7.7 硬件数字技术排除毛刺

## 7.7.3 定时方式去毛刺

接上页

```
PROCESS(CLK,KIN)   BEGIN
  IF RISING_EDGE(CLK) THEN
     IF (KIN='1') THEN  KH<=KH+1 ;--同时对键输入的高电平脉宽计数
        ELSE  KH<="0000";  END IF;  END IF; --若出现高电平，则计数器清 0
END PROCESS;
PROCESS(CLK,KH,KL)   BEGIN
  IF RISING_EDGE(CLK) THEN
     IF (KH>"1100") THEN  KOUT<='1' ;--对高电平脉宽计数一旦大于 12，则输出 1
      ELSIF  (KL>"0111") THEN KOUT<='0';--对低电平脉宽计数若大于 7，则输出 0
  END IF;    END IF;
END PROCESS;
END;
```

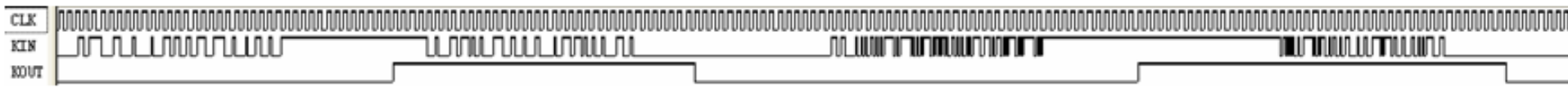


图 7-29 例 7-11 消抖动电路仿真波形

# 习　题

**7-1** 根据图**7-30**（**a**）所示的状态图，分别按照图**7-30**（**b**）和图**7-30**（**c**）写出对应结构的**VHDL**状态机。并根据表**7-2**，分别用**3**中不同编码方式实现二状态机，并讨论他们的容错措施。

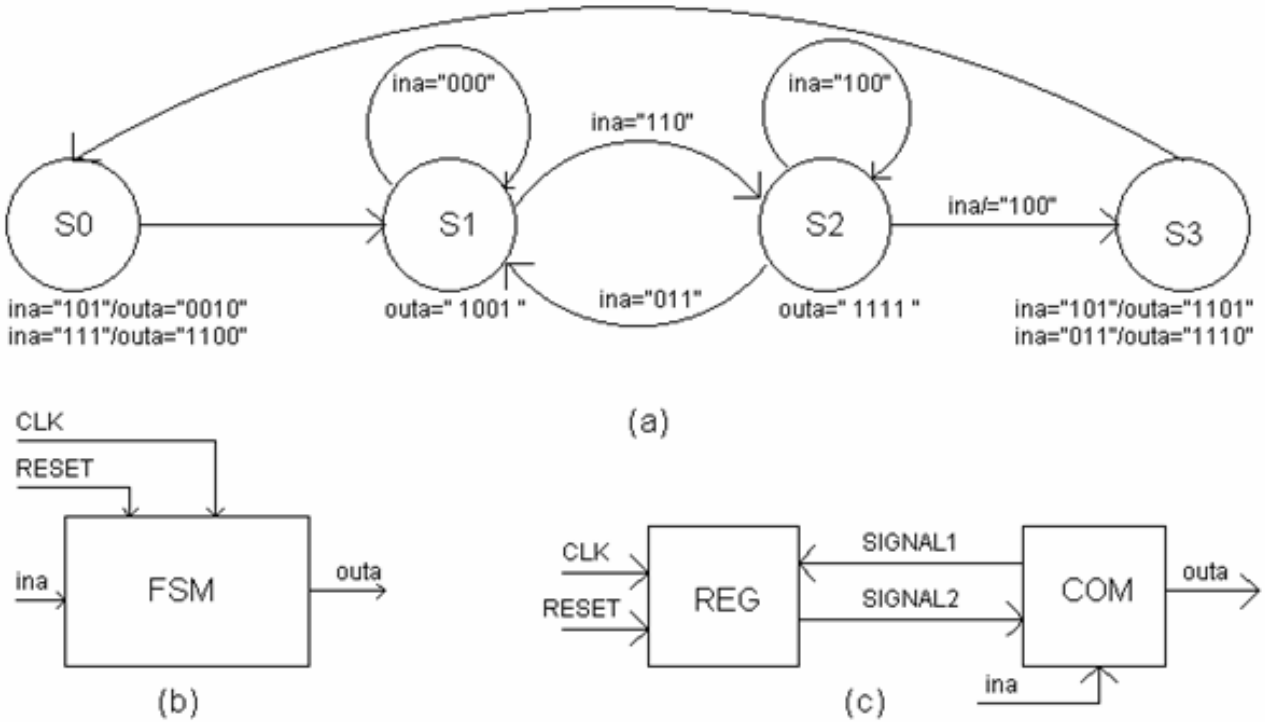


图 7-30 习题 7-1 状态图

# 习　题

**7-2** 举二例说明，有那些常用时序电路是状态机比较典型特殊形式，并说明它们属于什么类型的状态机（编码类型，时序类型和结构类型）。

**7-3** 用**mealy**机类型，写出控制**ADC0809**采样的状态机。

**7-4** 请设计一种信号去抖动的电路模型，仿真后，讨论其优缺点和使用范围。

**7-5** 根据**7.5**节，用表格法和绘图法设计状态机，实现例**7-2**的功能，用时序仿真波形图验证之。最后将其转变成**VHDL**程序，将此程序与例**7-2**相比，讨论他们的表述风格。

# 实验与设计

## 7-1 序列检测器设计

（1）实验目的：

（2）实验任务：

（3）实验思考题：。

（4）实验报告：

基于5E+系统的演示示例：/KX_7C5EE+/EXPERIMENTs/EXP34_SCHK/

# 实验与设计

## 7-2 并行ADC采样控制电路实现与硬件验证

（1）实验目的：

（2）实验原理：

（3）实验任务1：

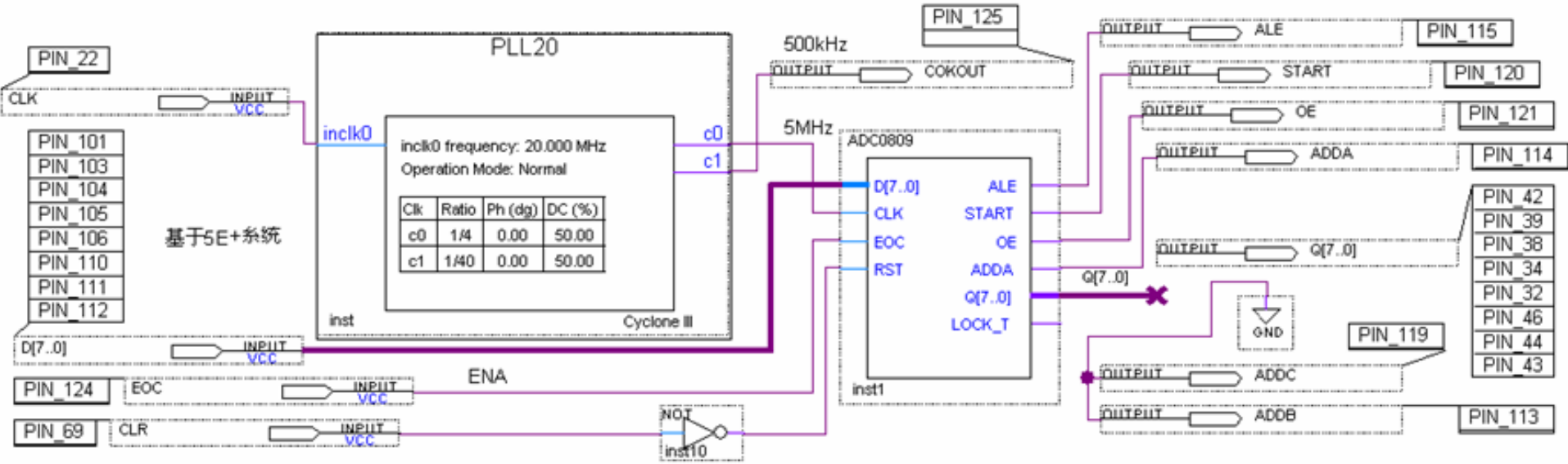


图 7-31 ADC0809 采样控制实验电路与引脚锁定指示

# 实验与设计

## 7-2 并行ADC采样控制电路实现与硬件验证

（4）实验任务2：

（5）实验任务3：

（6）实验任务4：

（7）实验报告：

# 实验与设计

## 7-3 数据采集模块和简易存储示波器设计

（1）实验目的：

（2）实验原理：

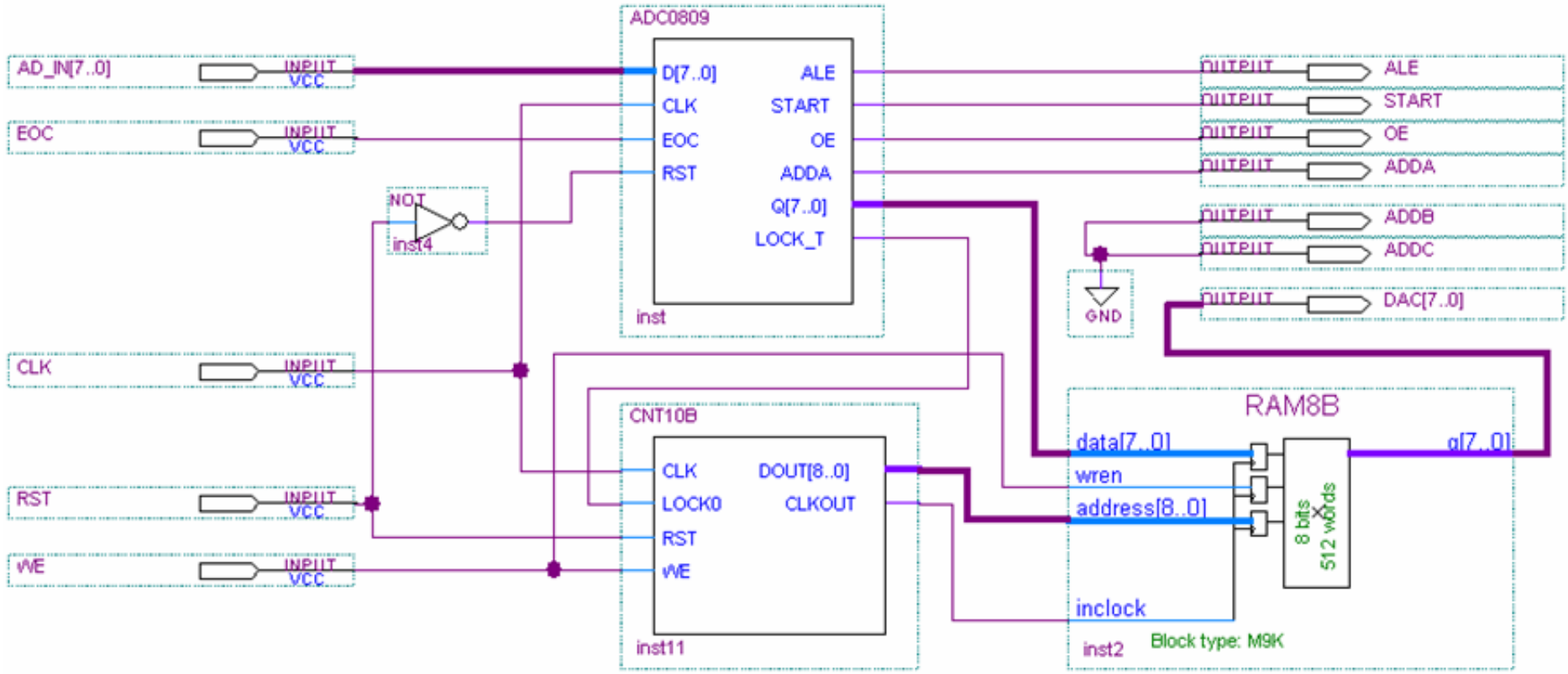


图 7-32　ADC0809 采样电路及简易存储示波器控制系统

# 实验与设计

**7-3** 数据采集模块和简易存储示波器设计

（3）实验内容1：

（4）实验内容2：

（5）实验内容3：

（6）实验内容4：

向另一5E+系统下载：

/KX_7C5EE/DEMOs/EXP10_DDS_Core_DAC0832/MCU8951。

（7）实验内容5：

（8）实验内容6：

（9）实验内容7：

# 实验与设计

## 7-4 五功能智能逻辑笔设计

（1）实验目的：

（2）实验原理：

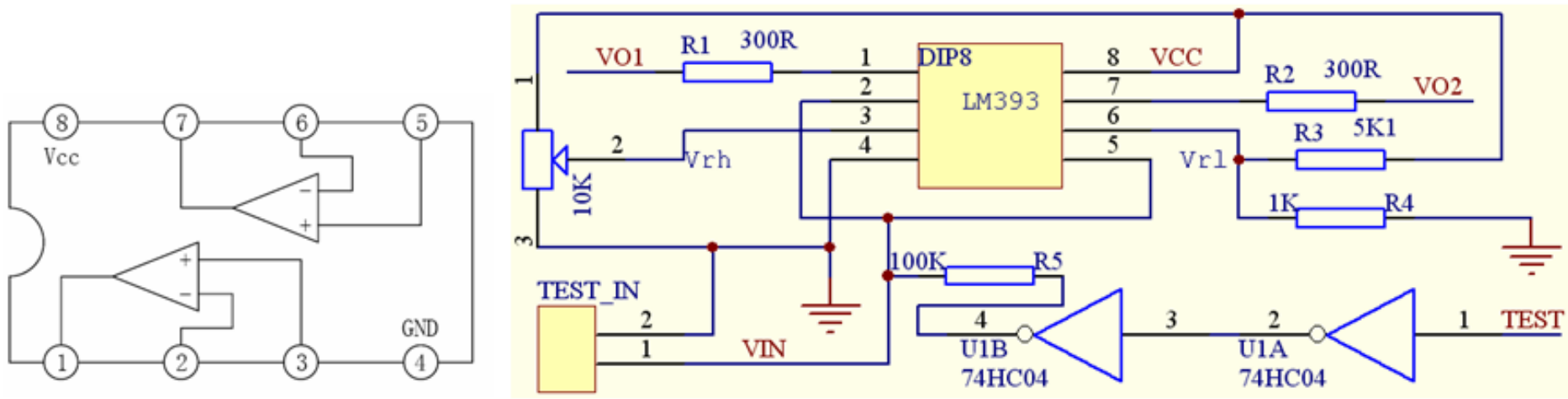


图 7-33 五功能智能逻辑笔电平信号采样电路，左图是 LM393 引脚图

# 实验与设计

## 7-4 五功能智能逻辑笔设计

（1）实验目的：

（2）实验原理：

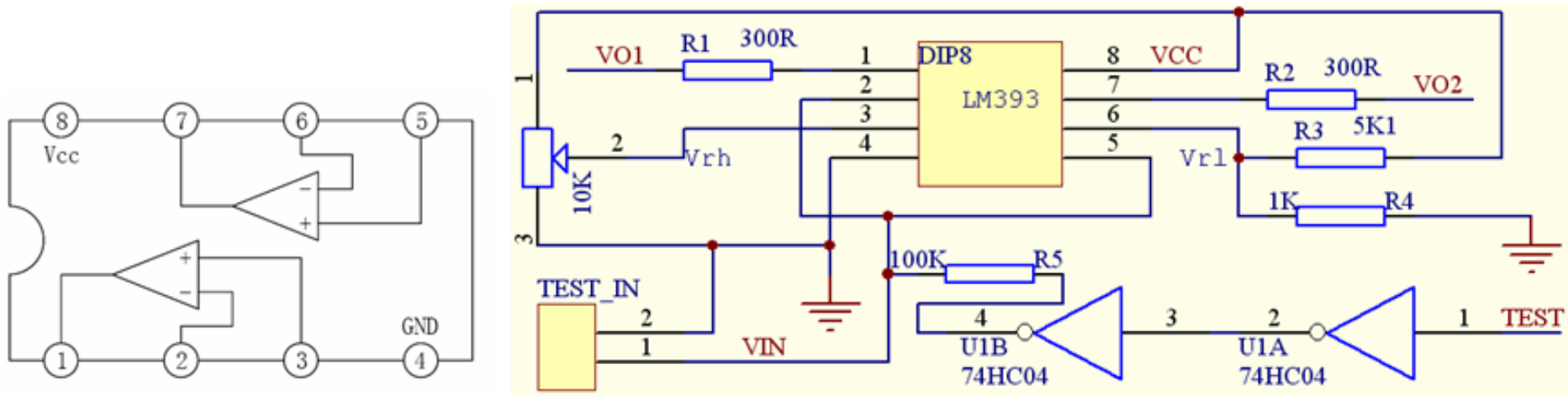


图 7-33 五功能智能逻辑笔电平信号采样电路，左图是 LM393 引脚图

示例文件：**/KX_7C5EE+/EXPERIMENTs/EXP14_LOGIC_PEN/**

（3）实验内容：

# 实验与设计

## 7-5 比较器加DAC器件实现ADC转换功能电路设计

（1）实验原理：

（2）实验内容1：

（3）实验内容2：

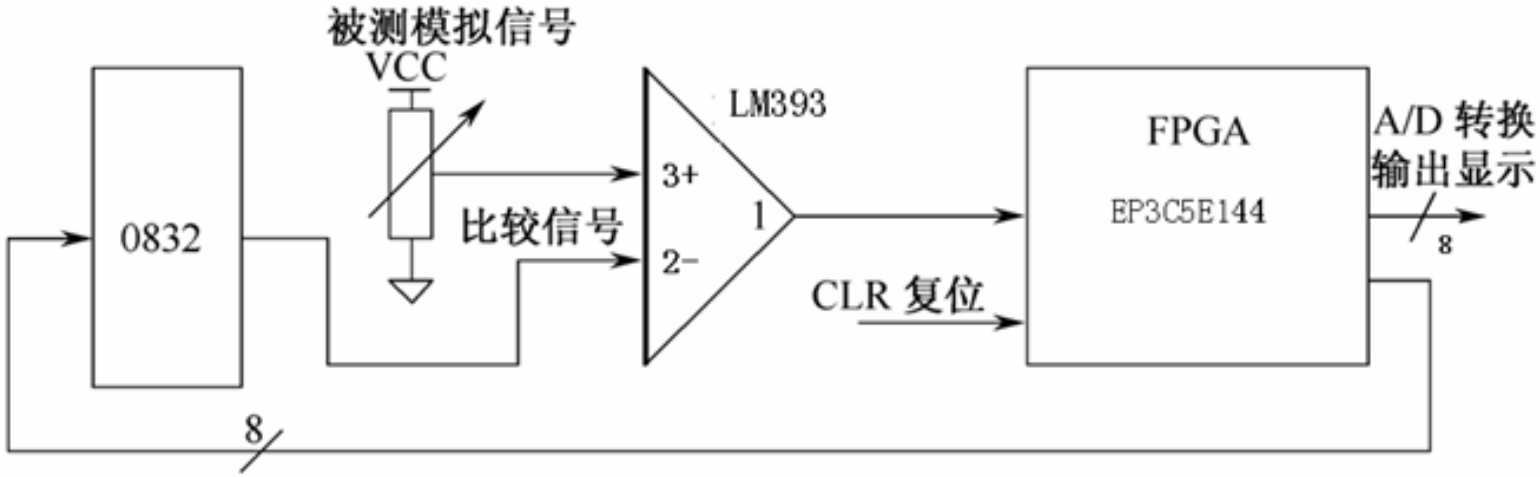


图 7-34 比较器和 D/A 构成 A/D 电路框图

# 实验与设计

## 7-6 通用异步收发器UART设计

**(1)** 实验目的：

**(2)** 实验内容1：

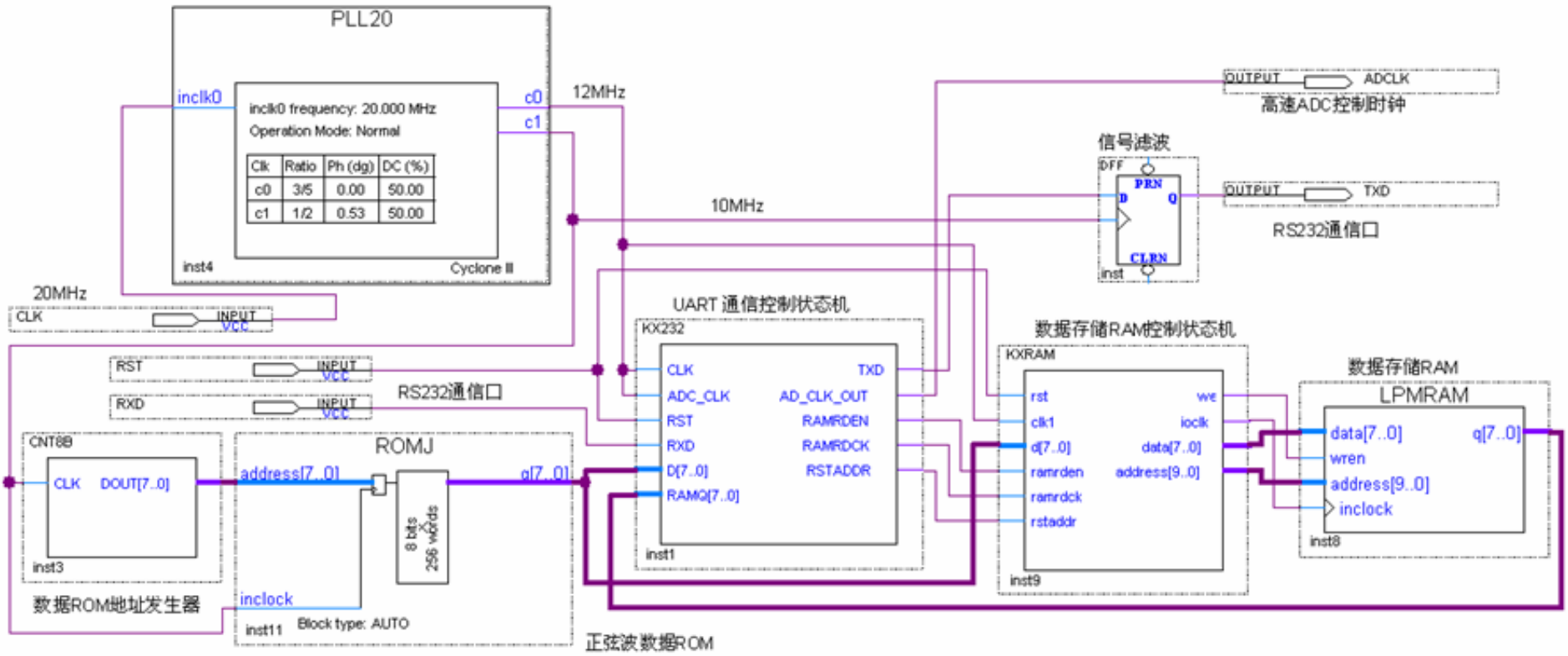


图 7-35 UART 通信设计顶层电路

# 实验与设计

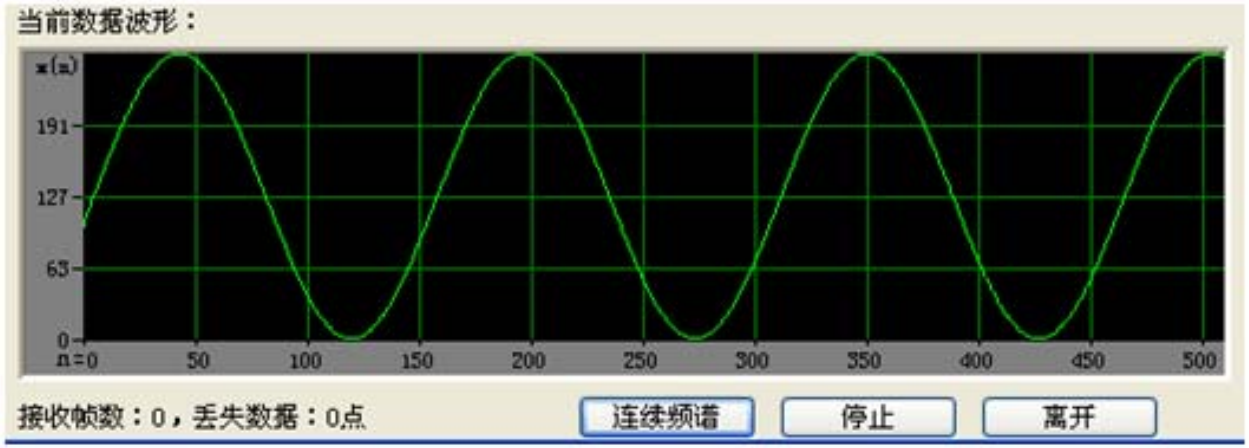


图 7-36 通过 RS232 通信来自 5E+系统的正弦波信号

**(3)** 实验内容**2**：

**(4)** 实验内容**3**：

另一**UART**演示示例：**/KX_7C5EE+/DEMOs/EXPL14_RS232_PIANO**。

# 实验与设计

## 7-7 点阵型与字符型液晶显示器驱动控制电路设计

（1）实验目的：

（2）实验原理：

（3）实验任务1：

（4）实验任务2：

以上2类基于5E+系统的LCD的演示示例是：

/KX_7C5EE/DEMOs//EXPL18_PS2_64X128LCD/；或

/EXPERIMENTs/EXP20_8051_LCD128X64/；

或/EXPERIMENTs/EXP17_KX8051_GPS_FTEST/。

# 实验与设计

## 7-8 串行ADC/DAC采样或信号输出控制电路设计

通过网络查阅一些常用串行**ADC**器件，包括它们的工作性能、使用方法、时序特点。设计出对应的电路，然后用状态机对其控制，最后比较用状态机和**CPU**的优缺点。

串行**ADC/DAC**资料查阅文件夹：**“PDF**实验设计文件”。

基于**5E+**的示例：**/KX_7C5EE+/EXPERIMENTs/EXP40_SADC_SDAC/**。

**TLV5637**和**TLV5618A**都是双通道**DAC**，比较适合于实现之前的提出的几个需要双通道**DAC**输出的实验，如基于**DDS**的移相信号发生器，里萨如图信号发生器，存储示波器等。

# 实验与设计

## 7-9 硬件消抖动电路设计

设计任务：**FPGA**中的去抖动电路十分常用，在以后的实验中会多次用到。

基于**5E+**系统的演示示例：**/KX_7C5EE+/EXPERIMENTs/EXP41_Di_Tremble/**

# 实验与设计

**7-10 状态机控制串/并转换8数码静态显示**

（1）实验原理：

（2）实验任务1：

（3）实验任务2：

（4）实验任务3：

演示示例：**/KX_7C5EE+/EXPERIMENTs/EXP43_74HC164_8LED/**

# 实验与设计

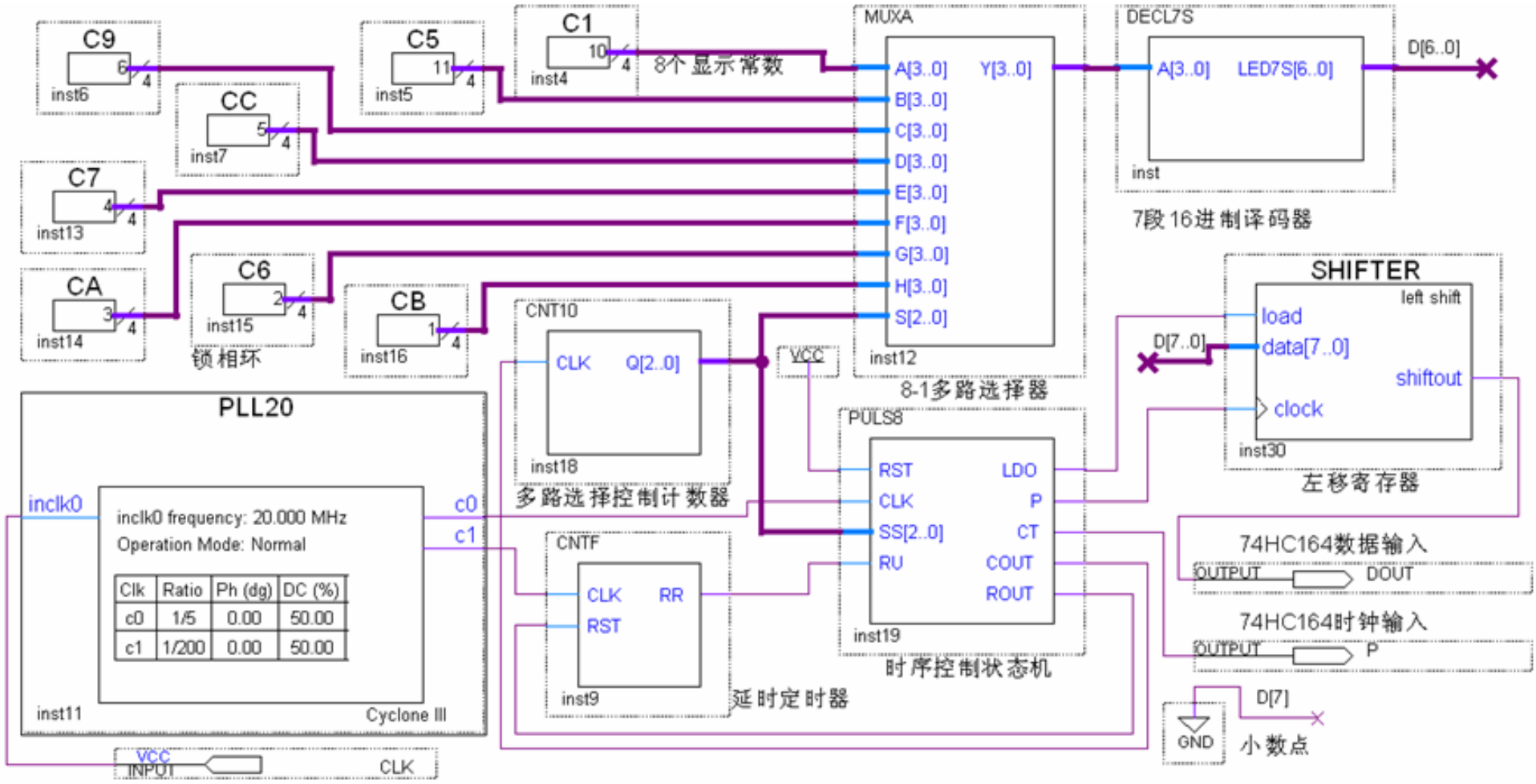


图 7-37　8 位串行静态显示状态机控制电路

# 实验与设计

## 7-11 数字温度器件DS18B20测控电路设计

设计任务：查阅**DALLAS**公司数字温度传感器**DS18B20**的使用方法和工作时序，设计一个状态机，控制并处理来自**DS18B20**的数据，将测得的温度显示在数码管上。

演示示例文件：

**/KX_7C5EE+/EXPERIMENTs/EXP19_8051_DS18B20/MCU8951**。按复位键**K1**即可看到液晶的温度显示，可用键设置温控上下限，超出温度限，蜂鸣器报警。