

# 第8章

## VHDL状态机设计与应用

# 8.1 VHDL状态机的一般形式

## 8.1.1 状态机的特点与优势

- (1) 高效的过程控制模型。
- (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) 高速性能。
- (5) 高可靠性能。

# 8.1 VHDL状态机的一般形式

## 8.1.2 VHDL状态机的一般结构

### 1. 说明部分

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

### 2. 主控时序进程

# 8.1 VHDL状态机的一般形式

## 8.1.2 VHDL状态机的一般结构

### 3. 主控组合进程

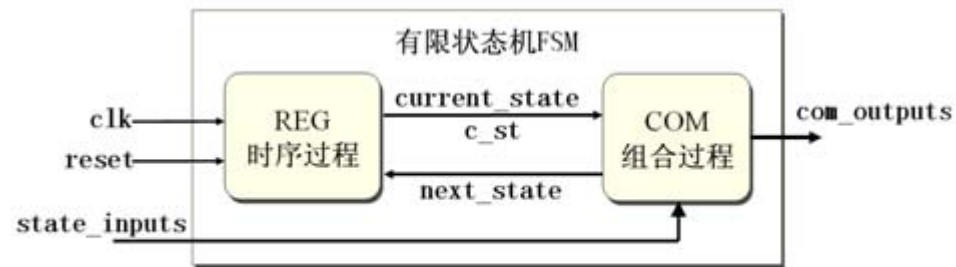


图 8-1 状态机一般结构示意图

### 4. 辅助进程

### 【例 8-1】

```
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;
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;
```

# 8.1 VHDL状态机的一般形式

## 8.1.2 VHDL状态机的一般结构

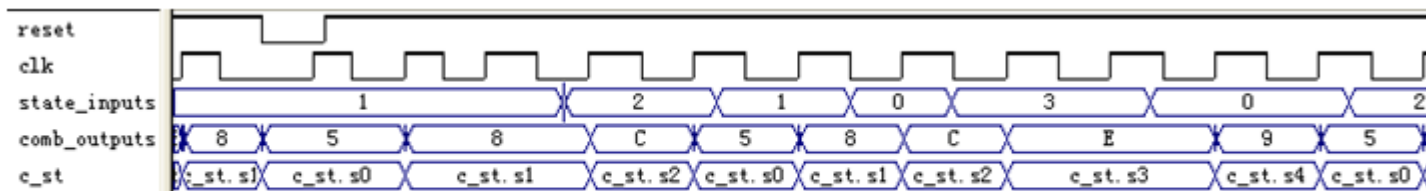


图 8-2 例 8-1 状态机的工作时序

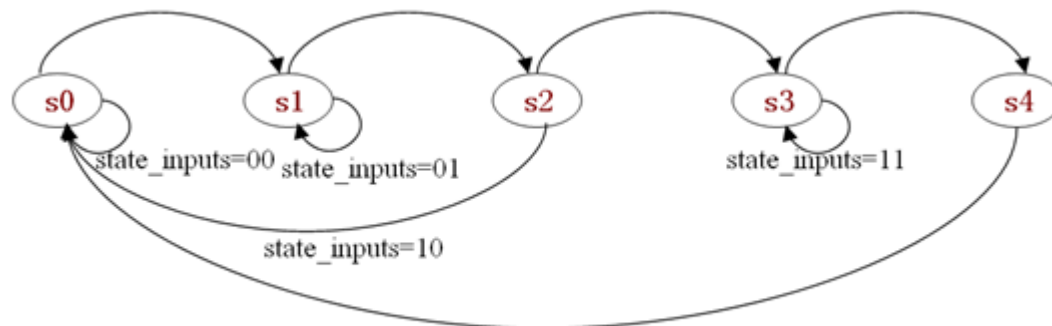


图 8-3 例 8-1 状态机的状态转换图

# 8.1 VHDL状态机的一般形式

## 8.1.3 状态机设计初始约束与表述

(1) 打开“状态机萃取”开关。

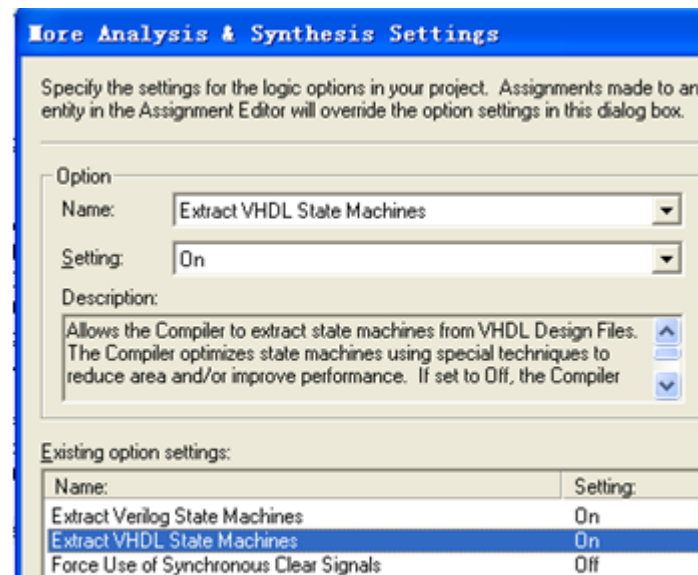


图 8-4 打开 VHDL 状态机萃取开关

(2) 状态图观察。

## 8.2 Moore型状态机的设计

### 8.2.1 多进程状态机

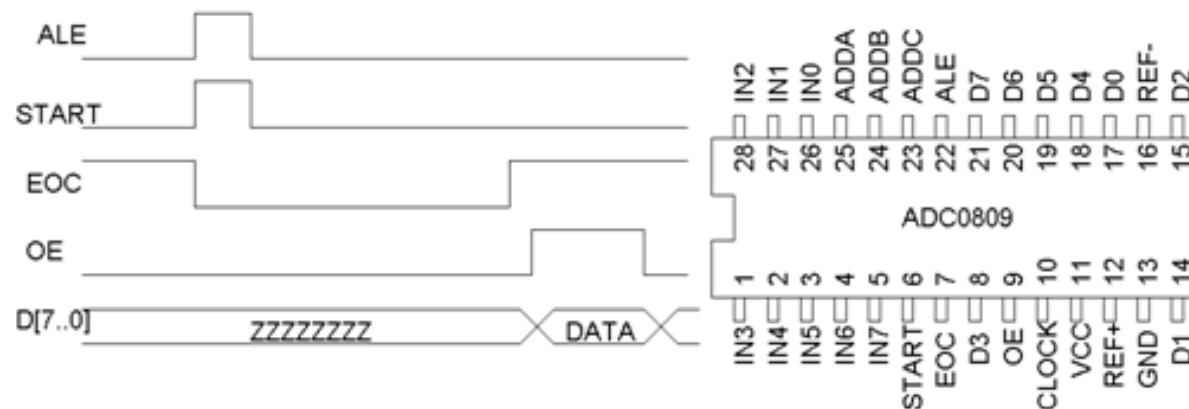


图 8-5 ADC0809 工作时序和芯片引脚图



## 8.2 Moore型状态机的设计

### 8.2.1 多进程状态机

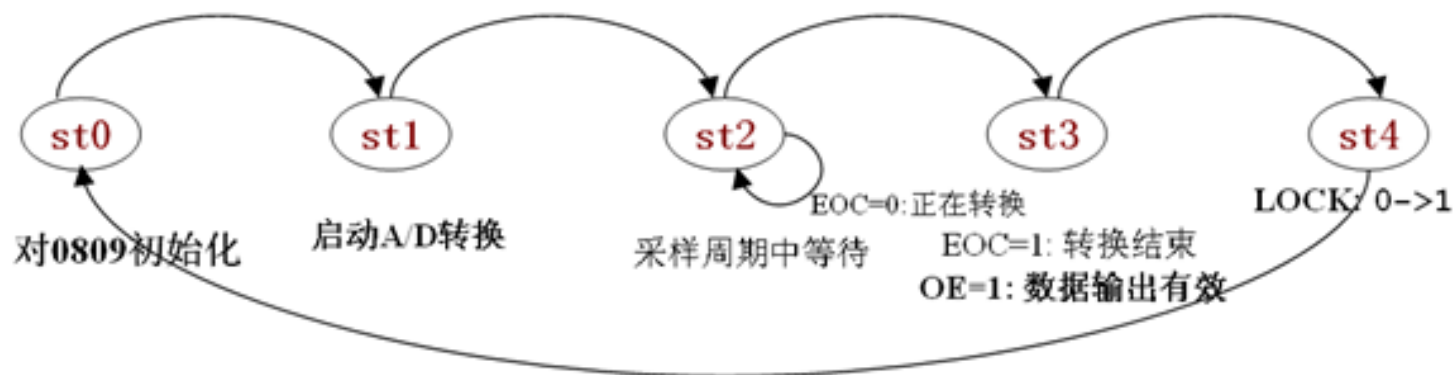


图 8-6 控制 ADC0809 采样状态图

## 8.2 Moore型状态机的设计

### 8.2.1 多进程状态机

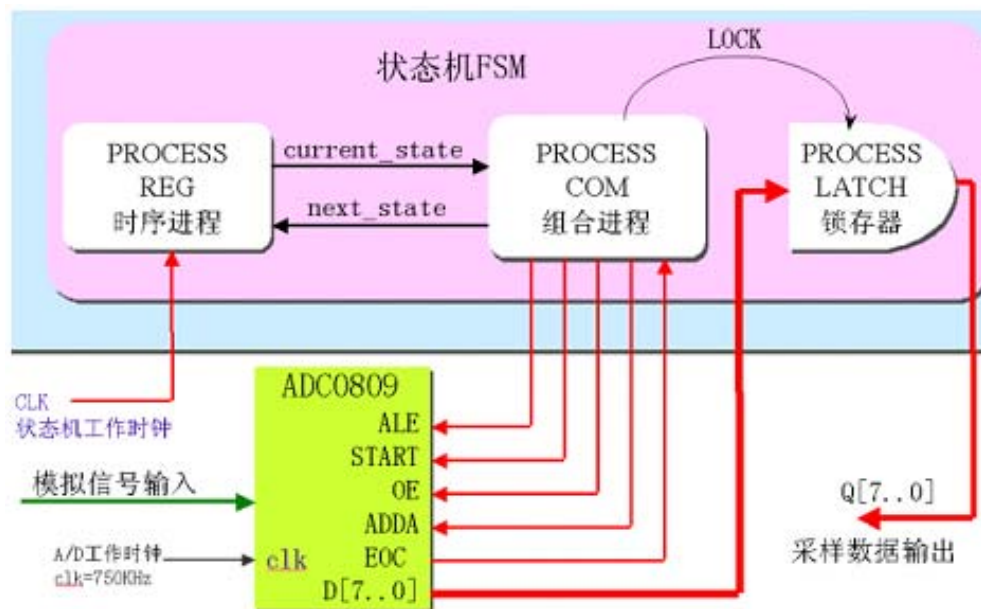


图 8-7 采样状态机结构框图

### 【例 8-2】

```
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
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;
```

## 8.2 Moore型状态机的设计

### 8.2.1 多进程状态机

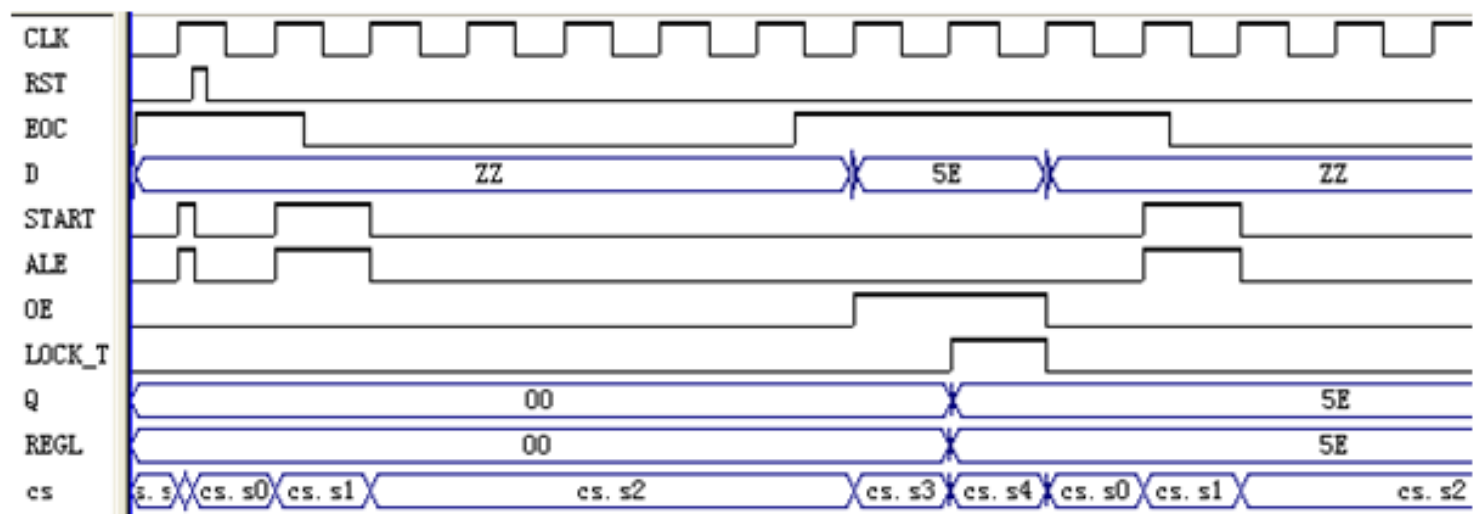


图 8-8 ADC0809 采样状态机工作时序

### 【例 8-3】

```
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 ;
```

## 8.2 Moore型状态机的设计

### 8.2.2 序列检测器之状态机设计

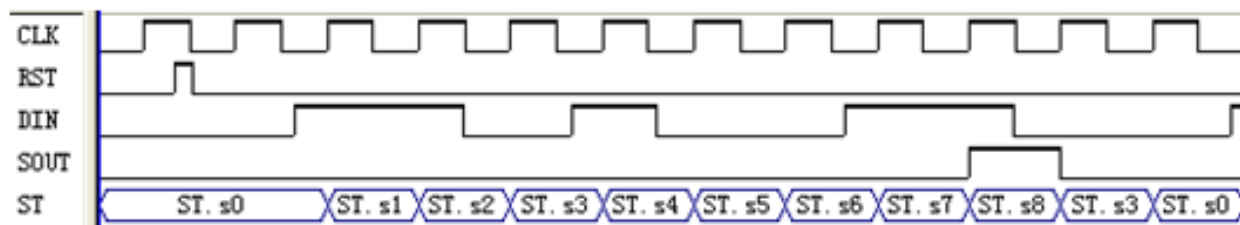


图 8-9 例 8.4 之序列检测器时序仿真波形

【例 8-4】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
    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 ;
```

## 8.3 Mealy型状态机的设计

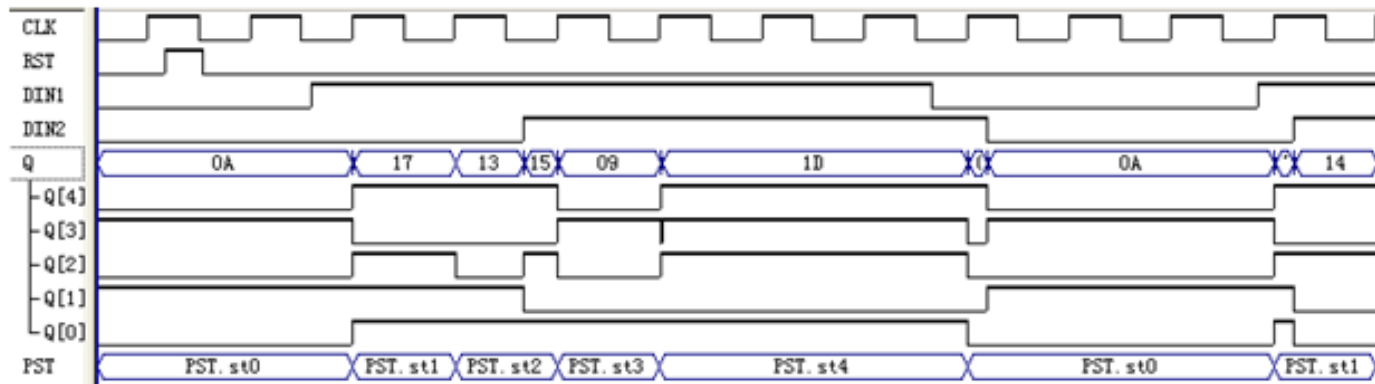


图 8-10 例 8-5 之双进程 Mealy 机仿真波形



### 【例 8-5】

```
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
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;
```

## 8.3 Mealy型状态机的设计

### 【例 8-6】

... --以上部分与例 8-5 相同

```
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 ;
    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;
```

## 8.3 Mealy型状态机的设计

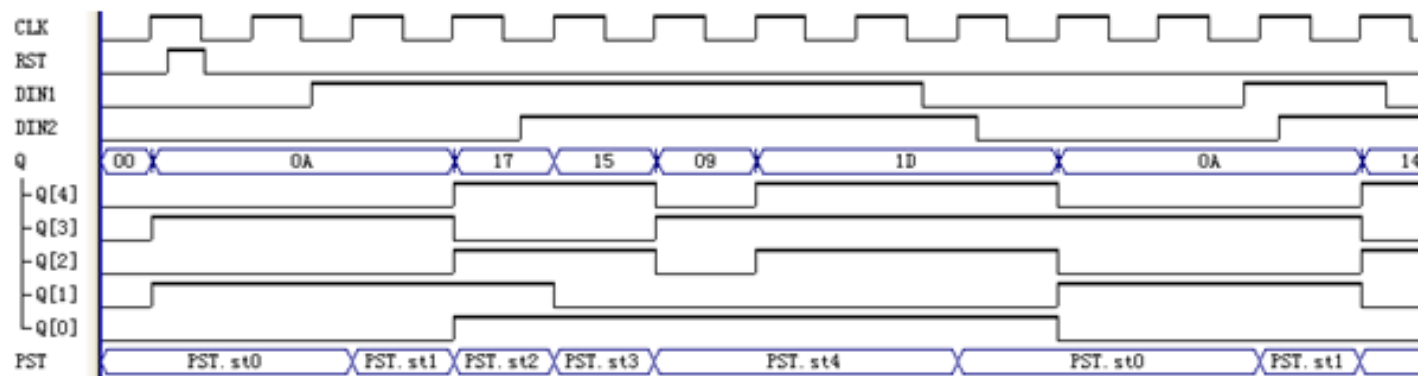


图 8-11 例 8-6 之单进程 Mealy 机仿真波形

## 8.3 Mealy型状态机的设计

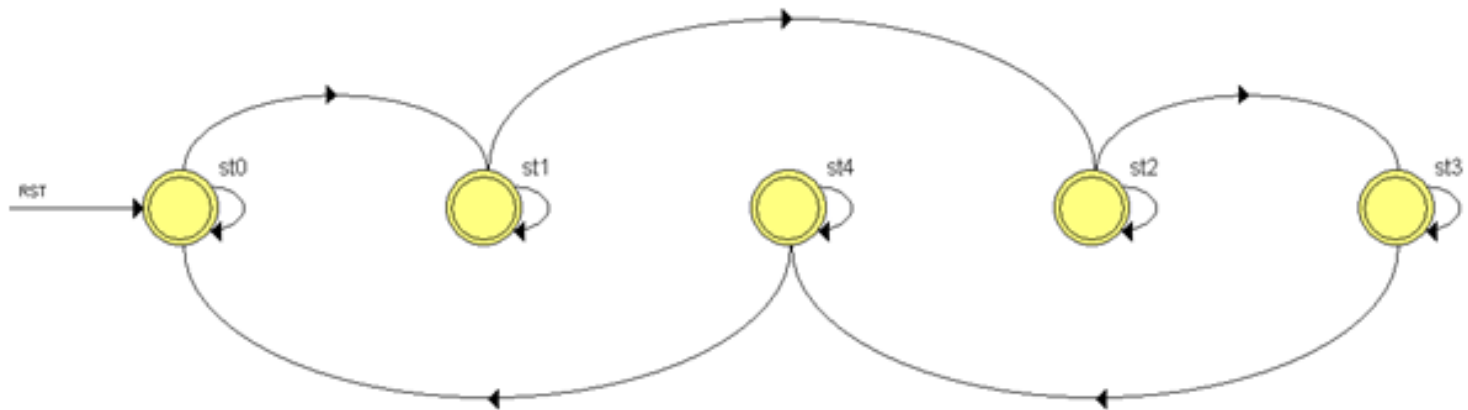


图 8-12 例 8-6, 8-5 的状态图

## 8.3 Mealy型状态机的设计

### 【例 8-7】

```
LIBRARY IEEE ;
LIBRARY IEEE ; --11010011
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY SCHK IS
    PORT(DIN,CLK, RST : IN STD_LOGIC;  SOUT : OUT STD_LOGIC);
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 ;
    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 ;
```

## 8.3 Mealy型状态机的设计

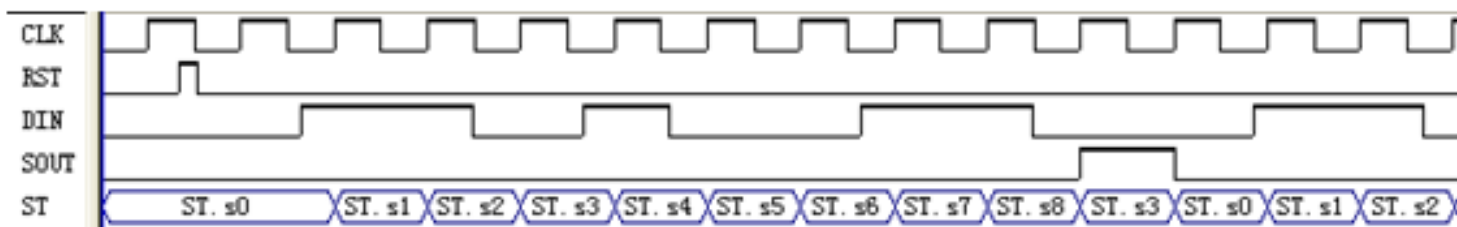


图 8-13 例 8-7 之单进程 Mealy 机仿真波形

# 8.4 状态编码

## 8.4.1 直接输出型编码

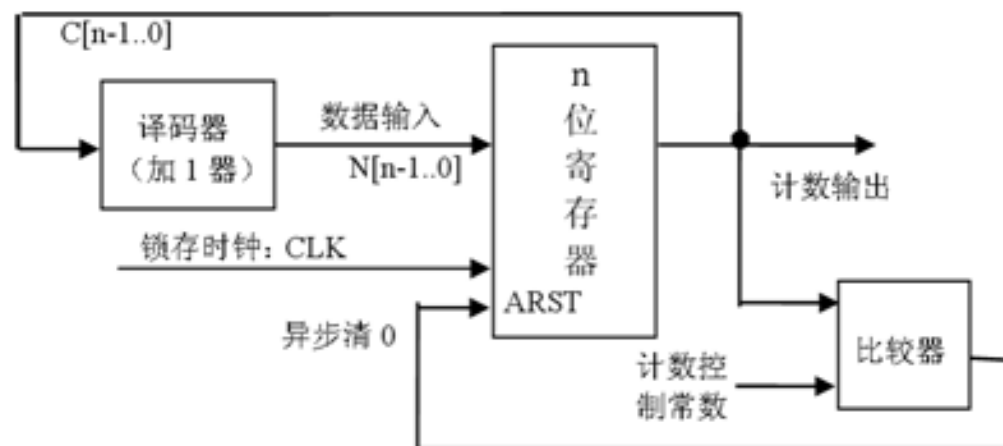


图 8-14 加法计数器一般模型

# 8.4 状态编码

## 8.4.1 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					功能说明
	START	ALE	OE	LOCK	B	
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 8-8】

... -- 以上部分与例 8-2 相同

```
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';
    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;
```

# 8.4 状态编码

## 8.4.1 直接输出型编码

### 【例 8-9】

```
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 ;
```

# 8.4 状态编码

## 8.4.2 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	10000	0000
State1	001	01000	1000
State2	010	00100	1100
State3	011	00010	1110
State4	100	00001	1111
State5	101	00000	0111

## 8.4.3 一位热码状态编码

# 8.4 状态编码

## 8.4.4 状态编码设置

### 1. 用户自定义方式

### 2. 直接设置方法

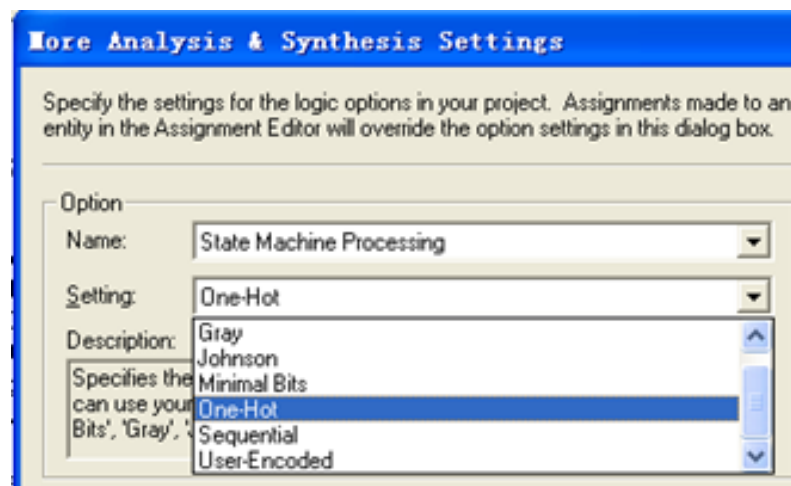


图 8-15 选择了一位热码编码形式

# 8.4 状态编码

## 8.4.4 状态编码设置

### 3. 用属性定义语句设置

#### 【例 8-10】

```
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
```

## 8.4 状态编码

### 8.4.4 状态编码设置

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	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

## 8.5 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

# 8.5 安全状态机设计

## 8.5.1 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;
```



# 8.5 安全状态机设计

## 8.5.2 状态编码监测法

$$\text{alarm} \leftarrow (\text{st0 AND } (\text{st1 OR st2 OR st3 OR st4 OR st5})) \text{ OR } (\text{st1 AND } (\text{st0 OR st2 OR ...}))$$

## 8.5.3 借助EDA优化控制工具生成安全状态机



图 8-16 选择安全状态机设计

# 实验与设计

## 8-1 序列检测器设计

## 8-2 并行ADC采样控制电路实现与硬件验证

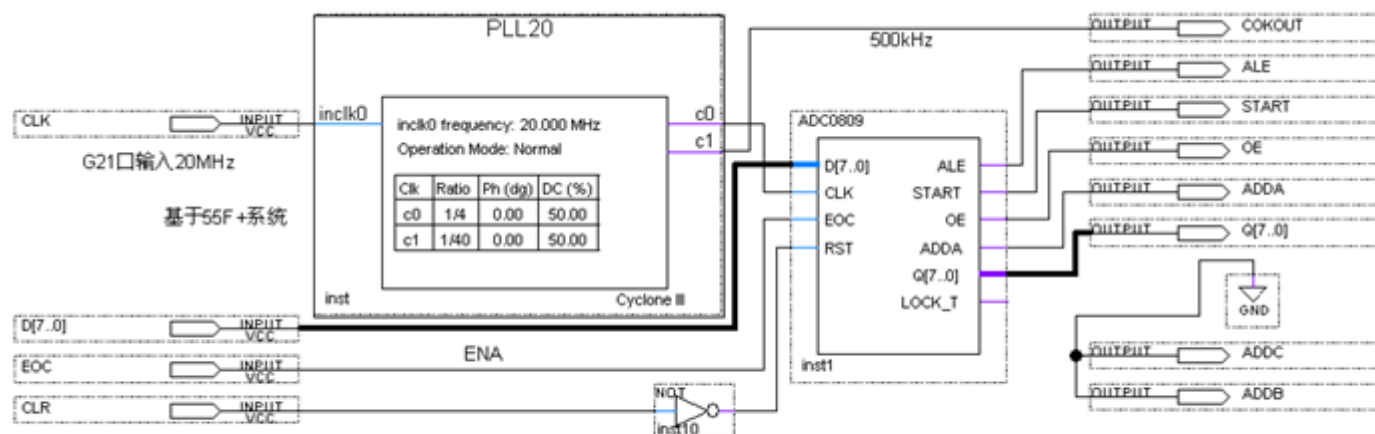


图 8-17 ADC0809 采样控制实验电路

# 实验与设计

## 8-3 数据采集模块设计

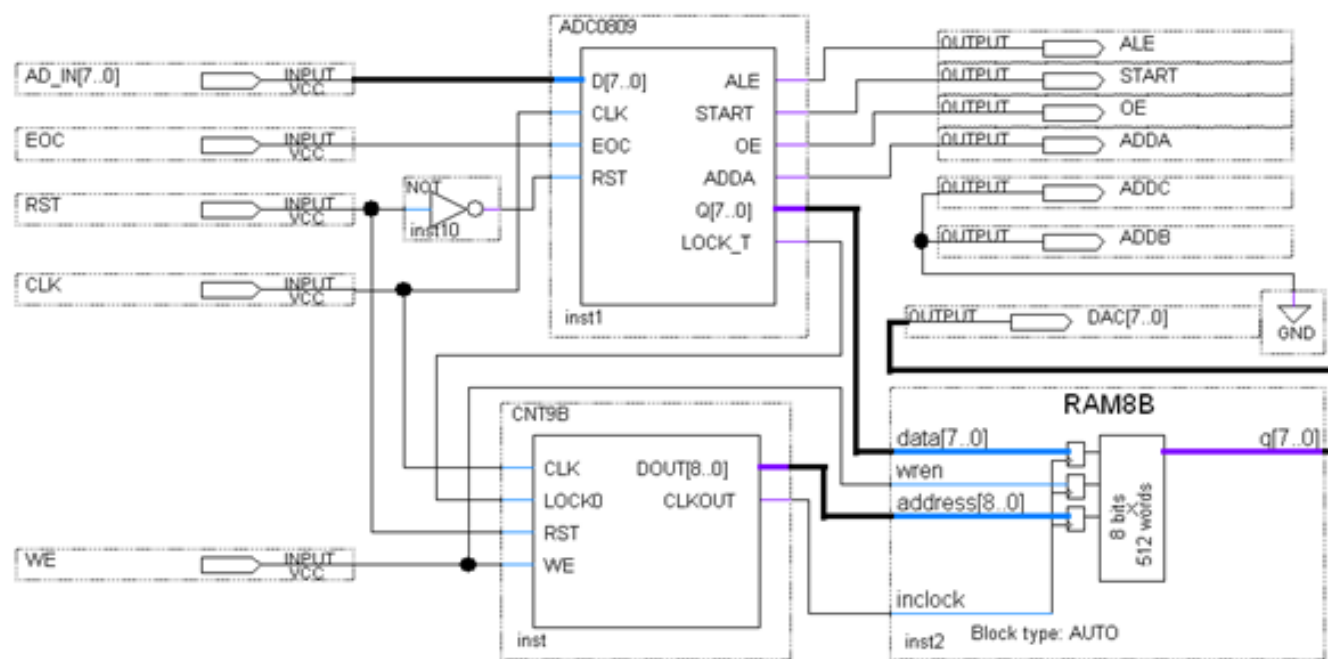


图 8-18 ADC0809 采样电路及简易存储示波器控制系统

# 实验与设计

## 8-4 五功能智能逻辑笔设计

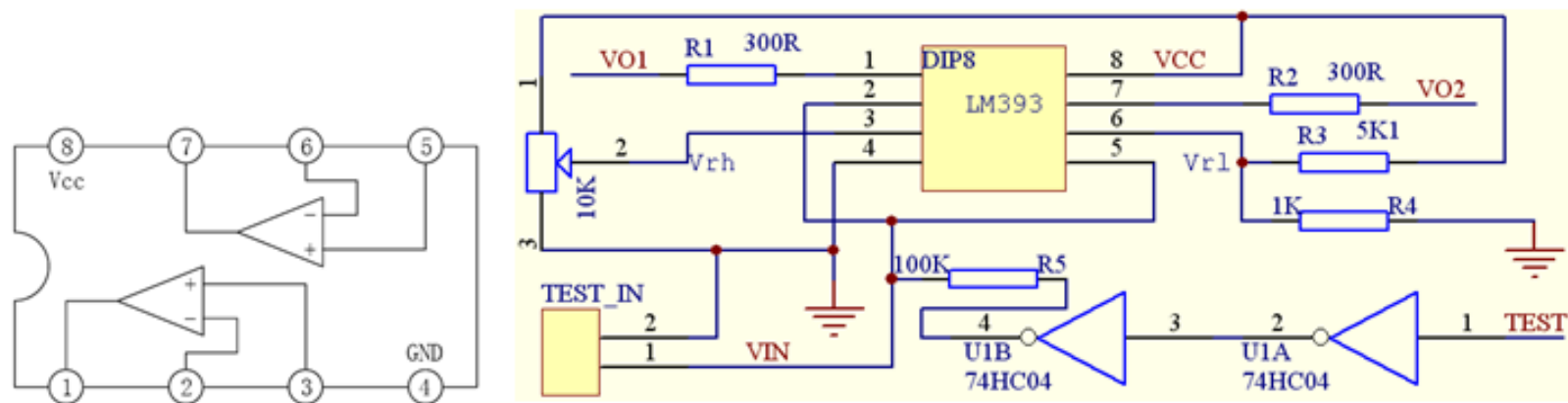


图 8-19 五功能智能逻辑笔电平信号采样电路，左图是 LM393 引脚图

## 8-5 串行ADC/DAC采样或信号输出控制电路设计