EDA技术

第10章 Verilog状态机设计技术



《EDA技术实用教程——Verilog HDL版》（第七版）配套课件）

EDA Lab

什么是状态机

广义而论，只要是涉及触发器的电路，无论电路大小，都能归结为状态机。状态机的设计对系统的高速性能、高可靠性、稳定性都具有决定性作用。

(1)状态机是用来描述系统逻辑功能的一种方式;

(2)定义了系统所处的不同状态;

(3)描述了系统状态转化的条件及转换过程;

(4)刻画了系统状态和输入输出之间的关系.

EDA Lab

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。

状态机的工作方式是根据控制信号按照预先设定的状态进行顺序运行的。

状态机是纯硬件数字系统中的顺序控制模型（相对于基于软件工作的CPU），因此状态机在其运行方式上类似于控制灵活和方便的CPU，是高速高效过程控制的首选。

EDA Lab

10.1 Verilog状态机的一般形式

10.1.1 状态机的特点与优势

（2）容易利用现成的EDA工具进行优化设计。

由于状态机构建简单，设计方案相对固定，特别是可以作一些独具特色的规范固定的表述，使得这一切为HDL综合器尽可能自动地发挥其强大的优化功能提供了便利条件。

而且，性能良好的综合器都具备许多可控或自动的优化状态机的功能，如编码方式选择、安全状态机生成等。

EDA Lab

10.1 Verilog状态机的一般形式

10.1.1 状态机的特点与优势

（3）性能稳定。

状态机容易构成性能良好的同步时序逻辑模块，这对于对付大规模逻辑电路设计中令人深感棘手的竞争冒险现象无疑是一个上佳的选择。

EDA Lab

10.1 Verilog状态机的一般形式

10.1.1 状态机的特点与优势

（4）高速性能。

状态机状态变换周期只有一个时钟周期。而且在每一状态中，状态机可以并行同步完成许多运算和控制操作。所以状态机能提供更高的工作速度。

EDA Lab

10.1 Verilog状态机的一般形式

10.1.1 状态机的特点与优势

（5）高可靠性能。

首先，状态机是由纯硬件电路构成，它的运行不依赖软件指令的逐条执行；

其次，状态机的设计中能使用各种完整的容错技术；

再次，当状态机进入非法状态并从中跳出，所耗的时间十分短暂，通常只有2~3个时钟周期，数十纳秒，尚不足以对系统的运行构成损害。

EDA Lab

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

状态机的不同形式：

- 从状态机的信号输出方式上分，有**Mealy**型和**Moore**型两种状态机。
- 从状态机的描述结构上分，有单过程状态机和多过程状态机。
- 从状态表达方式上分，有符号化状态机和确定状态编码的状态机。
- 从状态机编码方式上分，有顺序编码、一位热码编码或其他编码方式状态机。

EDA Lab

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

同步状态机的一般结构通常包含

- 说明部分
- 主控时序过程
- 主控组合过程
- 辅助过程等

EDA Lab

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：包含状态转换变量的定义和所有可能状态的说明，必要时还需确定每一状态的编码形式

Verilog HDL:

```
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：负责状态机运转和在时钟驱动下负责状态转换的过程。

状态机随外部时钟信号以同步时序方式工作，当时钟发生有效跳变时，状态机的状态才改变。

EDA Lab

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

3. 主控组合过程：也称为状态译码过程，其任务是根据外部输入的控制信号和当前状态的状态值，确定下一个状态的取向，以及确定输出控制信号。

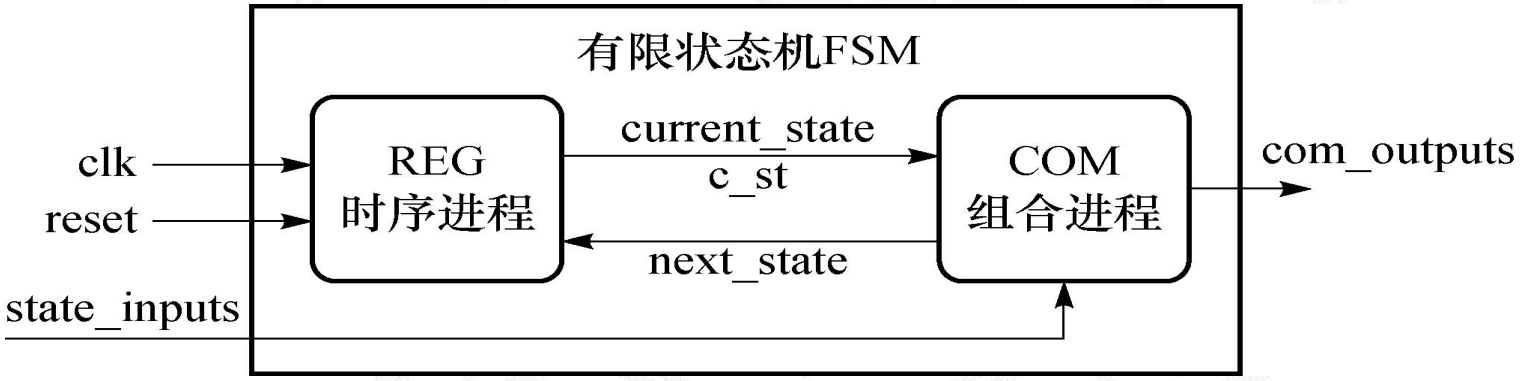


图10-1 状态机一般结构示图

4. 辅助过程：配合状态机工作的组合过程或时序过程，如配合状态机工作的其他时序过程，或为了稳定输出设置的数据锁存等。

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

状态机举例：

两个主控过程构成

组合过程执行条件分支语句

【例10-1】

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

EDA Lab

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

状态机举例：

两个主控过程构成

组合过程执行条件分支语句

```
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
```

EDA Lab

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

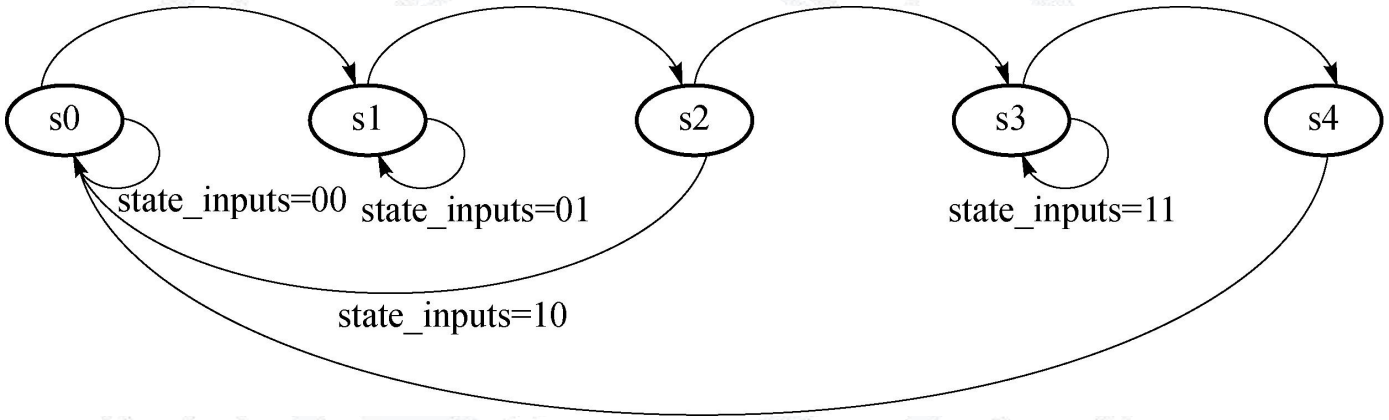


图10-2 例10-1状态机的状态转换图

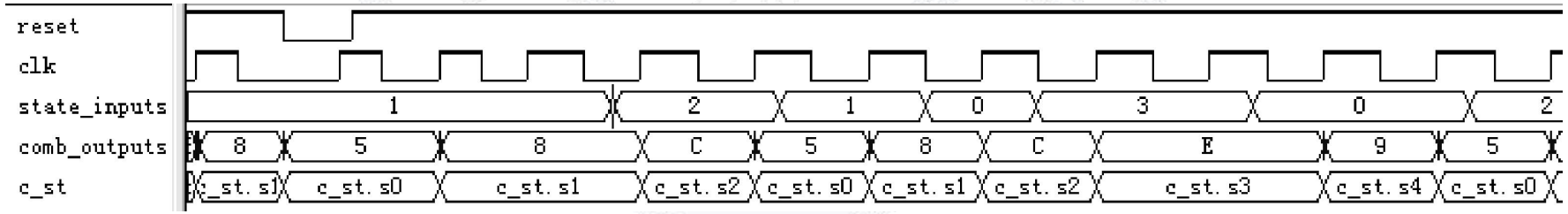


图10-3 例10-1状态机的工作时序图

10.1 Verilog状态机的一般形式



10.1.3 初始控制与表述

关于Verilog状态机的相关设置控制应注意以下几点：

（1）打开“状态机萃取”开关

如果确定自己描述的是状态机，并且希望综合器按状态机方式编译和优化，不要忘了打开综合器的“状态机萃取”开关。该选择一旦打开，综合器就会努力辨别输入的设计是否属于状态机，并对其施加状态机优化方案和各种约束。

EDA Lab

10.1 Verilog状态机的一般形式



10.1.3 初始控制与表述

（2）关于参数定义表述

在状态机设计中，用parameter进行参数定义虽然十分必要（综合器萃取状态机的主要依据），但一旦打开“状态机萃取”开关，其定义的形式可以十分随意。

如例10-1中的定义可以表述为“parameter s0=0, s1=1”等，或“parameter s0=4’b1001, s1=4’0011”等，因为最后状态机被综合的结果未必会按照此表述方式来构建。

EDA Lab

10.1 Verilog状态机的一般形式



10.1.3 初始控制与表述

（3）状态变量定义表述

正确定义状态变量的表述也是综合器萃取状态机的主要依据。

例10-1的状态定义语句是“reg [2:0] c_st, next_state;”，其中定义了现态和次态变量c_st和next_state，它们分别为寄存器型变量。

EDA Lab

状态机分类

从信号的输出方式上分，有Moore型和Mealy型两类状态机。从输出时序看，前者属于同步输出状态机，后者属于异步输出状态机（注意工作时序方式都属于同步时序）。

(1) Moore 状态机

如果时序逻辑的输出仅仅取决于当前状态，而与当前时刻的输入无关，称它为Moore状态机。这类状态机在输入发生变化时还必须等待时钟的到来，时钟使状态发生变化时才导致输出的变化。

(2) Mealy 状态机

如果时序逻辑的输出不但取决于当前状态还取决于输入，称它为Mealy状态机。它的输出是在输入变化后立即发生的，不依赖于时钟同步。

EDA Lab

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

（例：用状态机设计一个A/D采样控制器，控制ADC0809芯片工作）

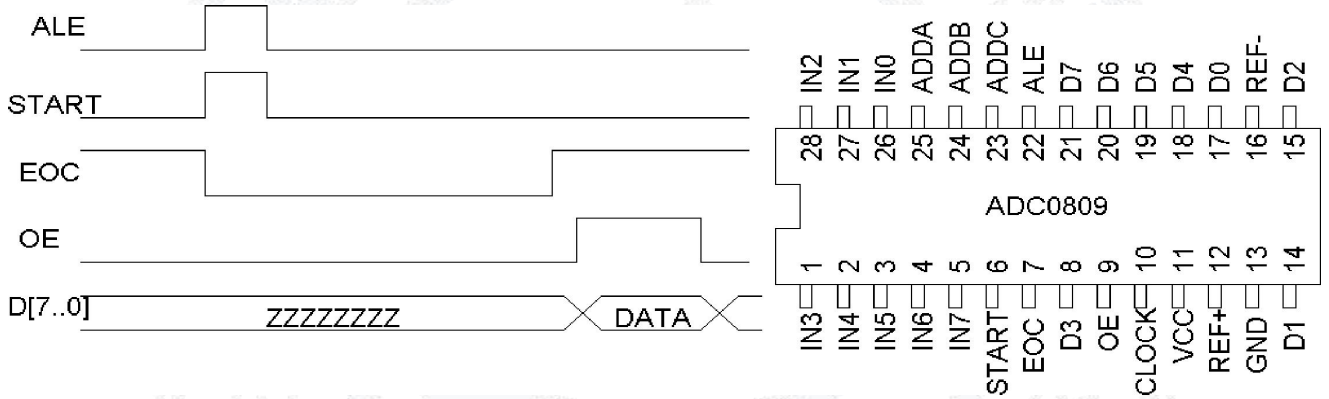


图10-4 ADC0809工作时序和芯片引脚图

（1）START为转换启动控制信号，高有效；ALE为模拟信号输入选通端口地址锁存信号，上升沿有效。一旦START有效后，状态信号EOC变为低电平，进入转换状态（~100us）。转换结束后，EOC变为高电平，控制器可以据此了解转换情况。

（2）外部控制可以使OE由低电平变为高电平（输出有效），此时，0809的输出数据总线D[7..0]从原来的高阻态变为输出数据有效。

（3）对0809输出数据进行锁存。

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

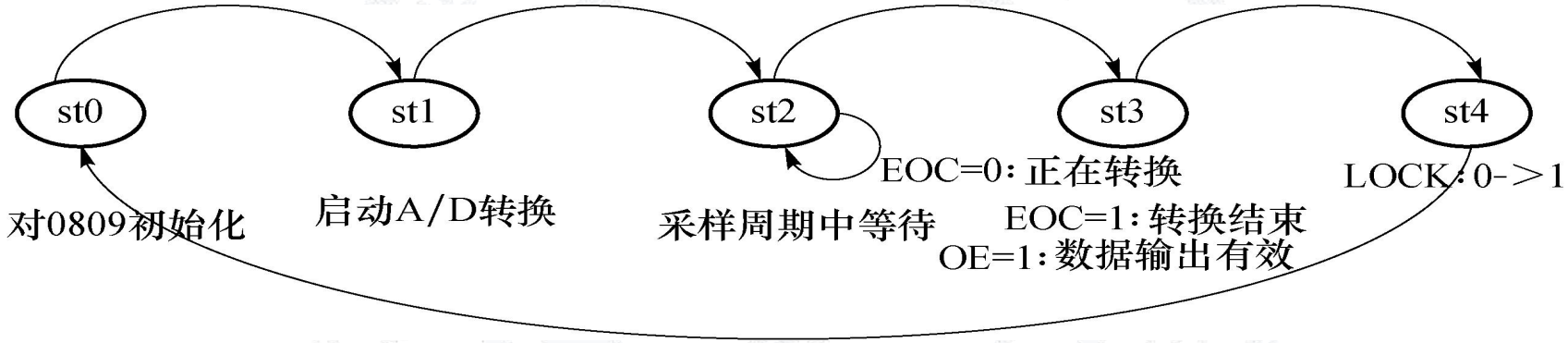


图10-5　控制ADC0809采样状态图

在st2状态需对0809工作状态信号EOC进行监测，如为低电平，表示转换未结束，停留在st2状态等待，直到变成高电平才说明转换结束，在下一时钟脉冲到来时转向状态st3。在st4，由状态机向锁存器发出锁存信号（LOCK上升沿），将0809输出的数据进行锁存。

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

【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;                   //采样数据输出
  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
```

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

```
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
                 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
```

EDA Lab

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

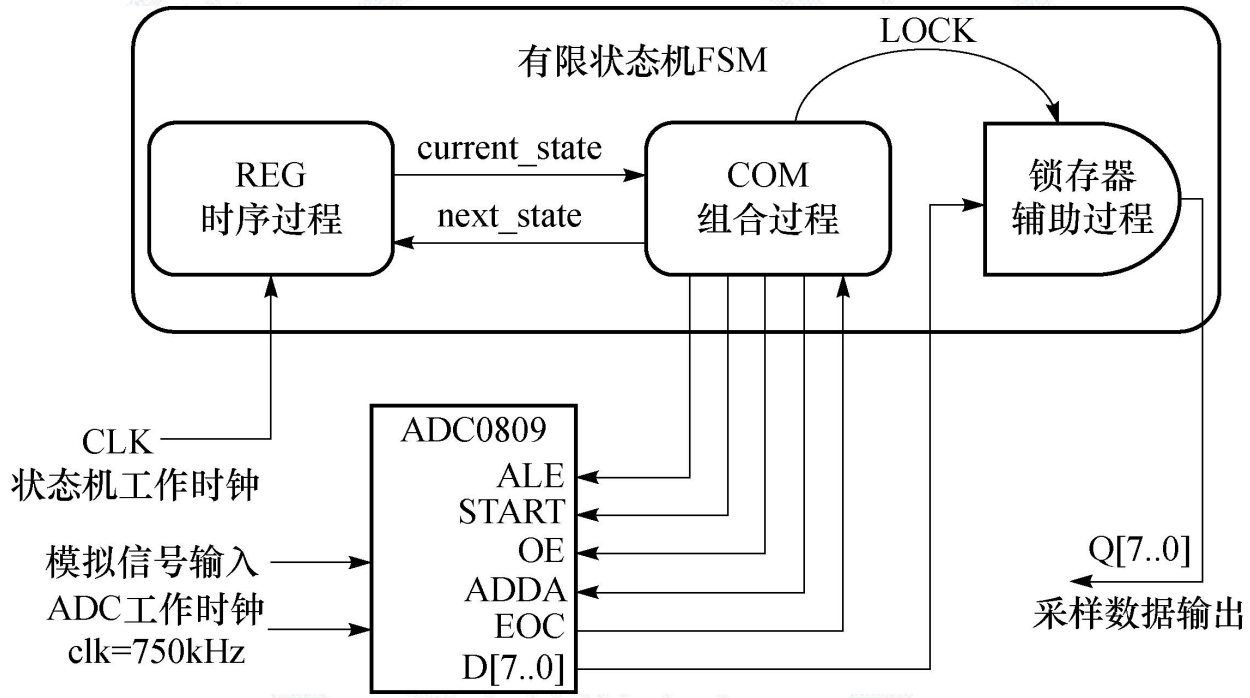


图10-6　采样状态机结构框图

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

- REG时序过程，在CLK驱动下不断将next_state的状态元素赋给现态信号cs
- COM组合过程有两个主要功能：
  - 状态译码器功能，即根据从现态cs信号中获得的状态变量，以及来自0809的状态线信号EOC，决定下一状态的转移方向。
  - 采样控制功能，即根据cs中的状态变量确定对0809的控制信号ALE、START、OE等输出相应的控制信号。当采样结束后还要通过LOCK向锁存器过程LATCH发出锁存信号，以便将输出数据锁存起来。

EDA Lab

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

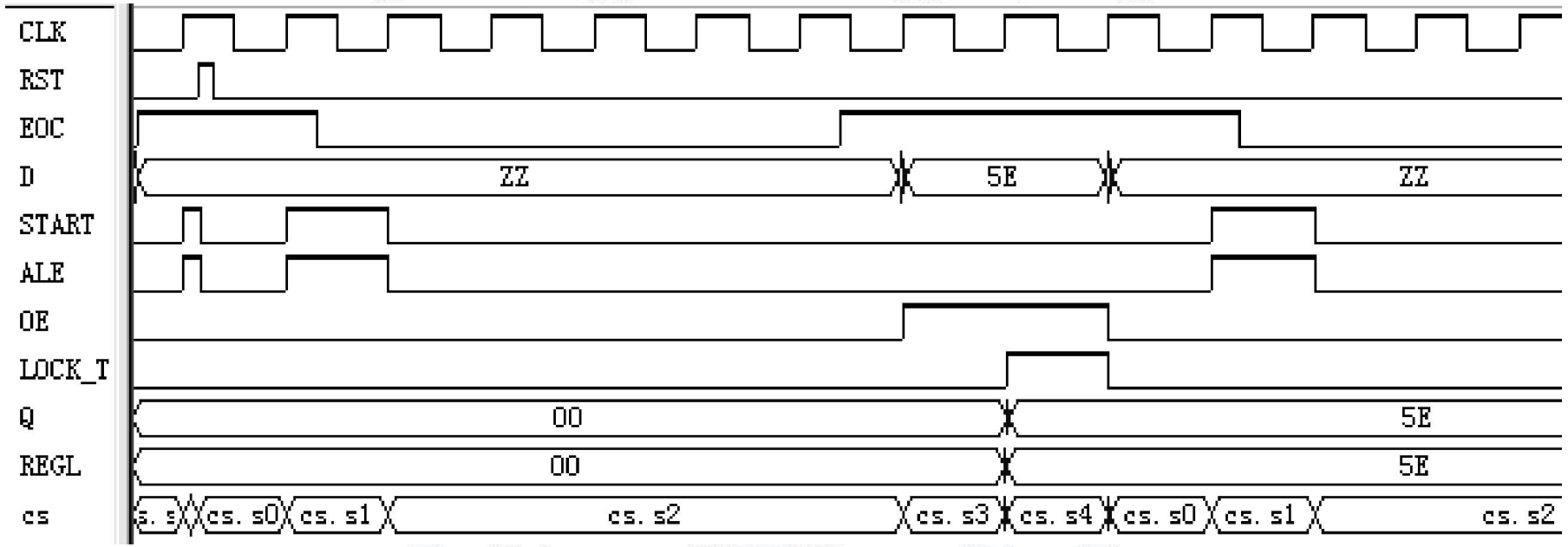


图10-7 ADC0809采样状态机工作时序图

LOCK_T是由内部LOCK信号引出的测试信号 `(* synthesis, probe_port, keep *)`

10.2 Moore型状态机及其设计

10.2.1 多过程结构状态机

【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
```

例10-2中的主控组合过程可以分成两个组合过程：一个负责状态译码和状态转换，另一个负责对外控制信号输出。

EDA Lab

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

序列检测器可用于检测一组或多组由二进制码组成的脉冲序列信号，当序列检测器连续收到一组串行二进制码后，如果这组码与检测器中预先设置的码相同，则输出1，否则输出0。

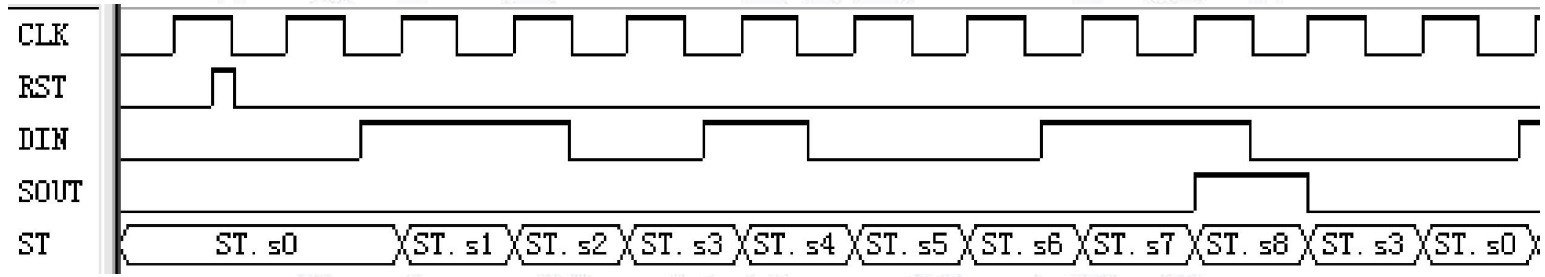


图10-8　例10-4的序列检测器时序仿真波形

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

检测“11010011”

【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<=s2;
            s3 :  if (DIN==1'b1)  NST<=s4; else NST<=s0;
            s4 :  if (DIN==1'b0)  NST<=s5; else NST<=s2;
            s5 :  if (DIN==1'b0)  NST<=s6; else NST<=s1;
            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<=s2;
           default :  NST<=s0;
       endcase          end
     assign  SOUT=(ST==s8);
endmodule
```

10.3 Mealy型状态机设计



相比Moore型，Mealy型状态机的输出变化要领先一个周期，即一旦输入信号或状态信号发生变化，输出信号即可发生变化。

两者设计基本相同，不同之处在于Mealy机的组合过程结构中的输出信号时当前状态和当前输入的函数。

EDA Lab

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
```

将状态机的主控时序电路和主控状态译码电路同时用一个过程来表达。

PST现态转换

过程COM则负责根据状态和输入信号给出不同的对外的控制信号输出。

EDA Lab

10.3 Mealy型状态机设计

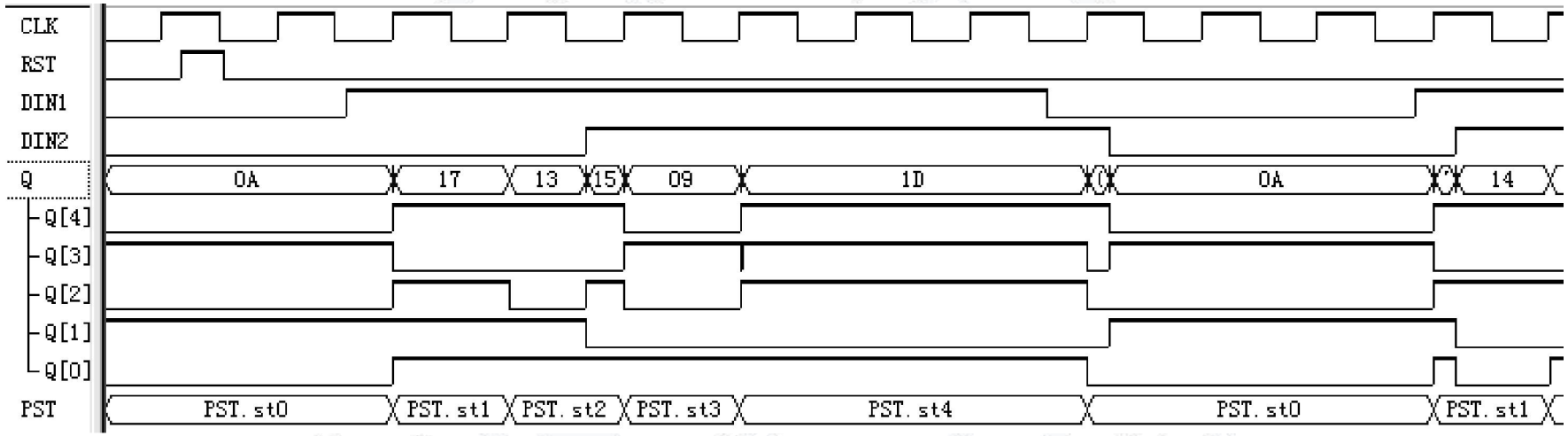


图10-9　例10-5的双过程Mealy机仿真波形

一旦输入信号或状态发生变化，输出信号即可发生变化。

10.3 Mealy型状态机设计



可以将例10-5的两过程Mealy状态机改为单过程结构的Mealy状态机

【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
```

EDA Lab

10.3 Mealy型状态机设计

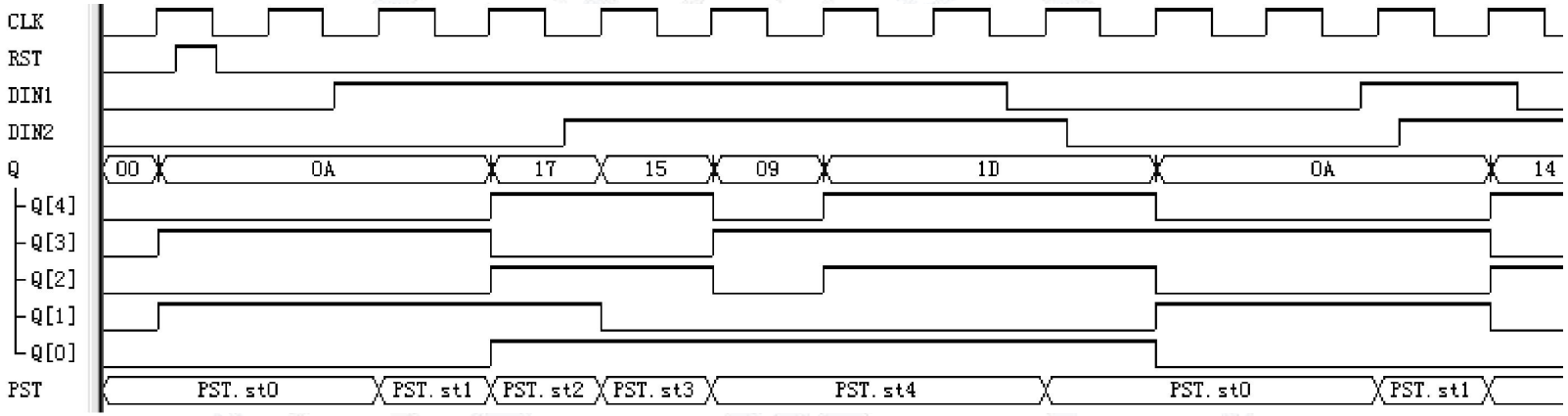


图10-10　例10-6的单过程Mealy机仿真波形

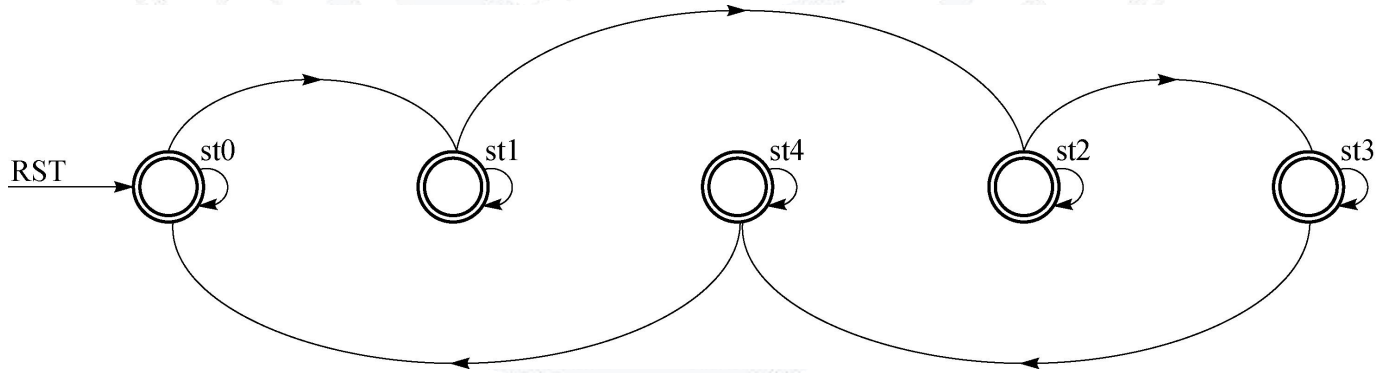


图10-11　例10-5和例10-6的状态图

10.3 Mealy型状态机设计



与例10-5不同，例10-6的每一待输出的数据必须等到时钟边沿到后才能输出，而在时钟边沿未到时，如果数据输出控制信号DIN2发生改变，则必定影响时钟后的输出数据。

事实上，只要将例10-5下面的COM过程的always引导语句改成以下形式，例10-5就和例10-6的功能全等了。

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

EDA Lab

10.3 Mealy型状态机设计

将描述序列检测器的例10-4的双过程结构Moore型机改成单过程的Mealy型机：

【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
```

10.3 Mealy型状态机设计

将描述序列检测器的例10-4的双过程结构Moore型机改成单过程的Mealy型机：

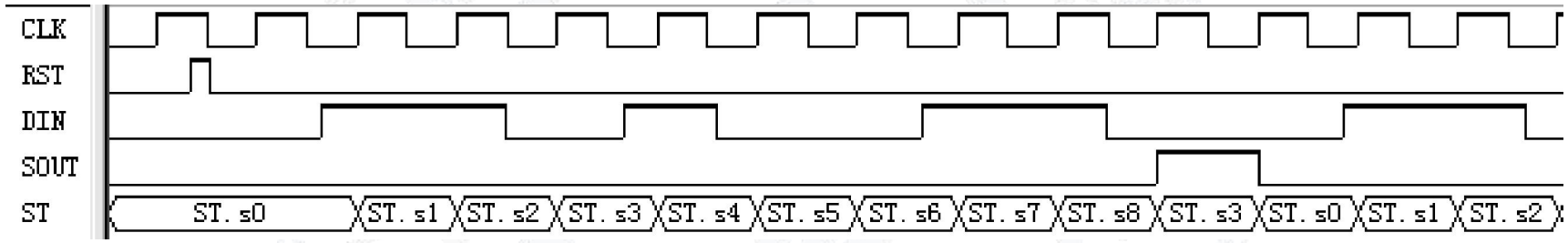


图10-12　例10-7的单过程Mealy机仿真波形

EDA Lab

10.4不同编码类型状态机

10.4.1 直接输出型编码

这类编码方式最典型的应用实例就是计数器。计数器本质上就是一个主控时序过程与一个主控组合过程合二为一的状态机，它的计数输出就是各状态的状态码。

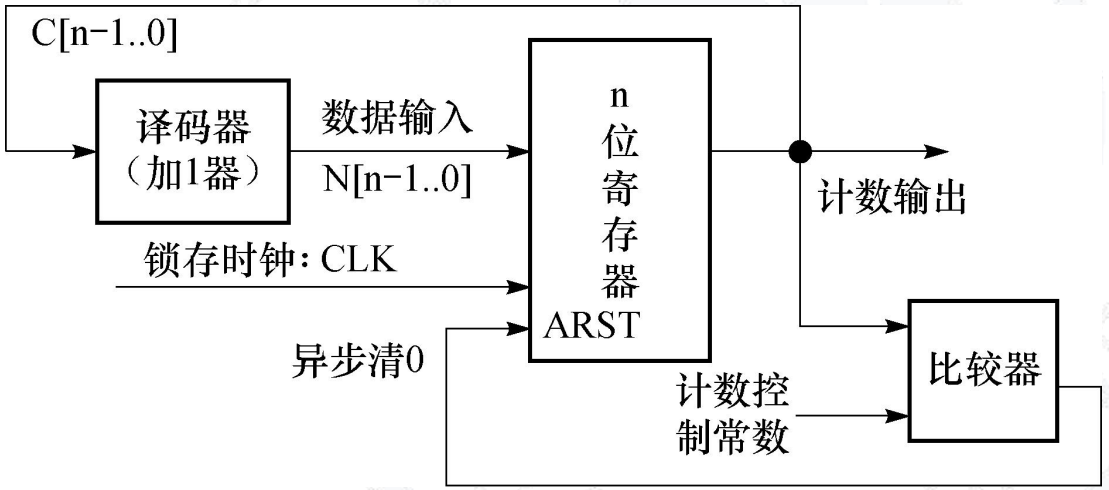


图10-13　加法计数器一般模型

此计数器的计数输出即为此状态机状态码输出。

将状态编码直接输出作为控制信号，即output=state,这种状态机称为状态码直接输出型状态机。

10.4不同编码类型状态机

10.4.1 直接输出型编码

对于状态码直接输出型状态机，要求对状态机各状态的编码作特殊的安排，以适应控制对象的要求。

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

从状态S0到S4各状态的编码可分别设为00000、11000、00001、00100、00110，每一位编码值都赋予了实际的控制功能。

10.4不同编码类型状态机

10.4.1 直接输出型编码

【例10-8】

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

10.4不同编码类型状态机

10.4.1 直接输出型编码

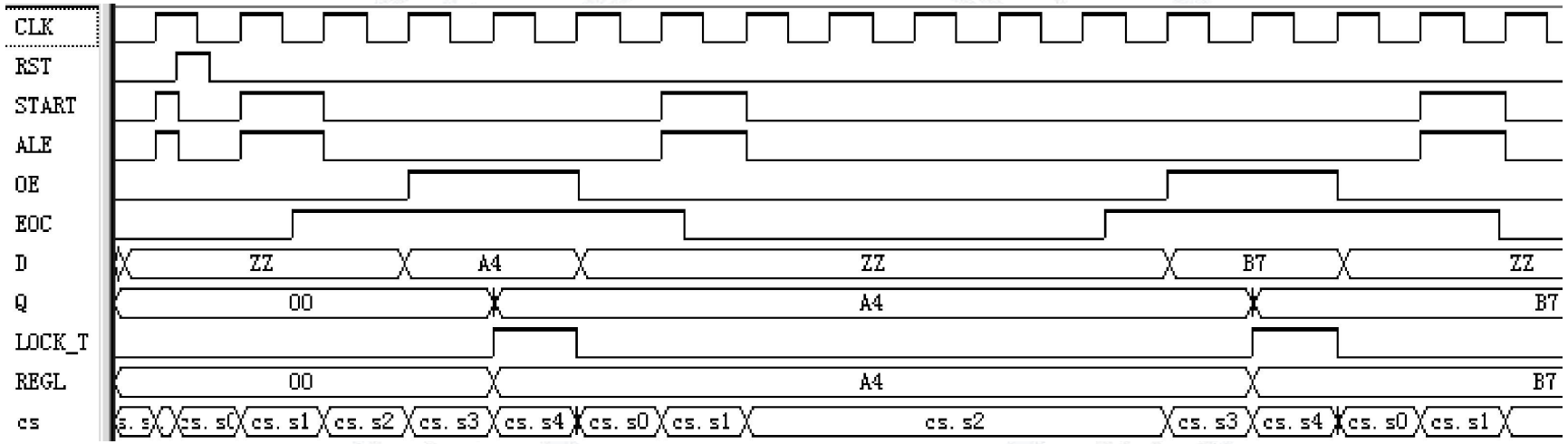


图10-14　例10-8状态机工作时序图

10.4不同编码类型状态机

10.4.2 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
  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
```

10.4不同编码类型状态机



10.4.2 用宏定义语句定义状态编码

- 用`define和parameter来定义状态元素的编码的区别是：前者定义可以针对整个设计全局，它定义的可以是全局符号常量，可以在各个不同的模块中通用，所以这时定义语句必须放在模块语句module外；
- 而后者定义在某个模块语句module中，只有局部特征（当然也可以将`define语句放在模块中）。

EDA Lab

10.4不同编码类型状态机

10.4.2 用宏定义语句定义状态编码

用宏定义的一个好处是可以在状态机的仿真波形中直接看到各状态的编码。

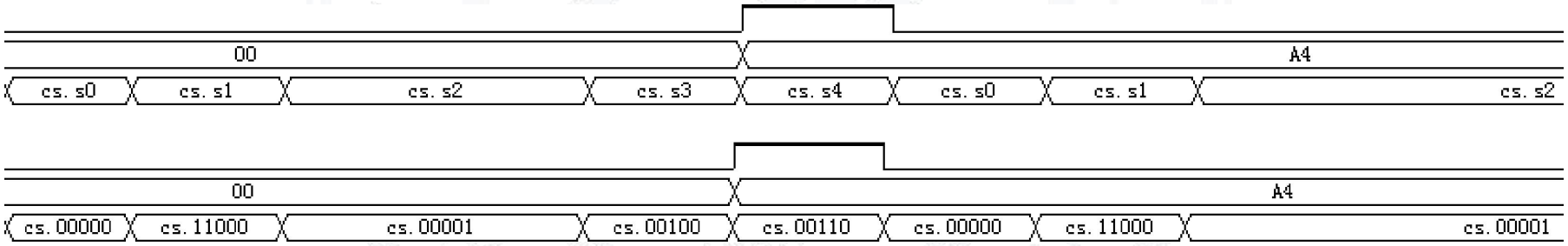


图10-15　不同状态元素定义状态机的仿真波形

10.4不同编码类型状态机

10.4.3 宏定义命令语句

宏定义语句`define属于编译指示语句，不参与综合；通过`define语句的定义，可以用简单的名称或称为宏名的标识符来替代一个复杂的名字、字符串、或表达式等，一般格式：

`define 宏名（标志符） 宏内容（字符串）

```
`define  s   A+B+C+D

assign   DOUT = A+B+C+D+E;

          `s
```

10.4不同编码类型状态机



10.4.3 宏定义命令语句

(1) `define定义可以针对整个设计全局，它定义的可以是全局符号常量，可以在各个不同的模块中通用。而parameter定义在某个模块语句module中，只有局部特征。

(2) 宏定义语句行末尾不加分号。

(3) 在程序中引用已定义的宏名时，必须在定义了宏名的标识符前面加上符号“`”，以示该标识符是一个宏定义的名字。

EDA Lab

10.4不同编码类型状态机

10.4.4 顺序编码

表10-2 编码方式

状态 (states)	顺序编码 (sequential-encoded)	一位热码编码 (one-hot-encoded)	约翰逊码编码 (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

顺序编码，最简单，其使用的触发器最少，剩余的非法状态也最少。但有时，相邻状态或不相邻状态转换时涉及多个触发器的同时状态转换，耗用更长的转换时间，而且容易出现毛刺。

EDA Lab

10.4不同编码类型状态机



10.4.5 一位热码编码

- 一位热码编码，就是用n个触发器来实现具有n个状态的状态机。状态机中的每一个状态都由其中一个触发器表示。当处于该状态时，对应的触发器为1，其余的触发器为0。
- 一位热码编码尽管用了较多的触发器，但其简单的编码方式大为简化了状态译码逻辑，提高状态转换速度，增强状态机的工作稳定性。

EDA Lab

10.4不同编码类型状态机



10.4.6 状态编码设置

1. 用户自定义方式

就是将需要的编码方式写在程序中，不需要EDA软件工具进行干预。

2. 用属性定义语句设置

直接在Verilog程序中使用属性定义语句指示编译器按照要求选择编码方式。

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

【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
```

EDA Lab

10.4不同编码类型状态机

10.4.6 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LC	触发器数REG
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

EDA Lab

10.5 异步有限状态机设计



10.5.1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
```

EDA Lab

10.5 异步有限状态机设计

10.5.1异步状态机设计示例1

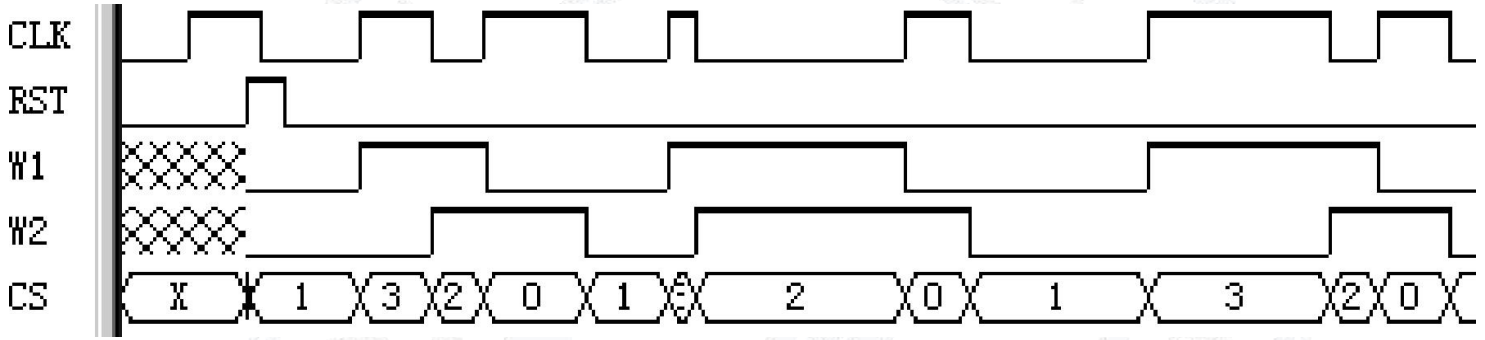


图10-16　例10-11的工作时序图

10.5 异步有限状态机设计

10.5.2异步状态机设计示例2

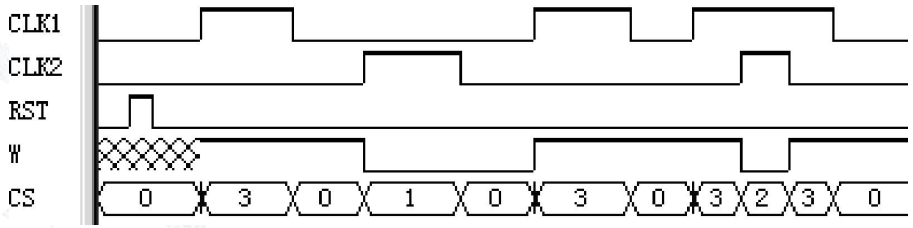


图10-17　例10-12的工作时序图

【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
```

10.5 异步有限状态机设计

10.5.2异步状态机设计示例2

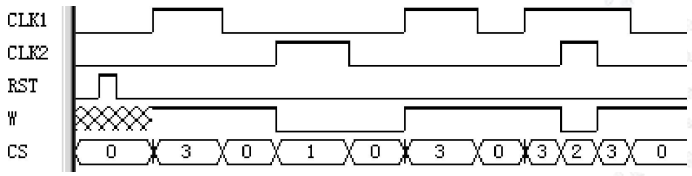


图10-17　例10-12的工作时序图

```
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
          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
```

10.5 异步有限状态机设计

10.5.3异步状态机设计示例3

【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
```

10.5 异步有限状态机设计

10.5.3异步状态机设计示例3

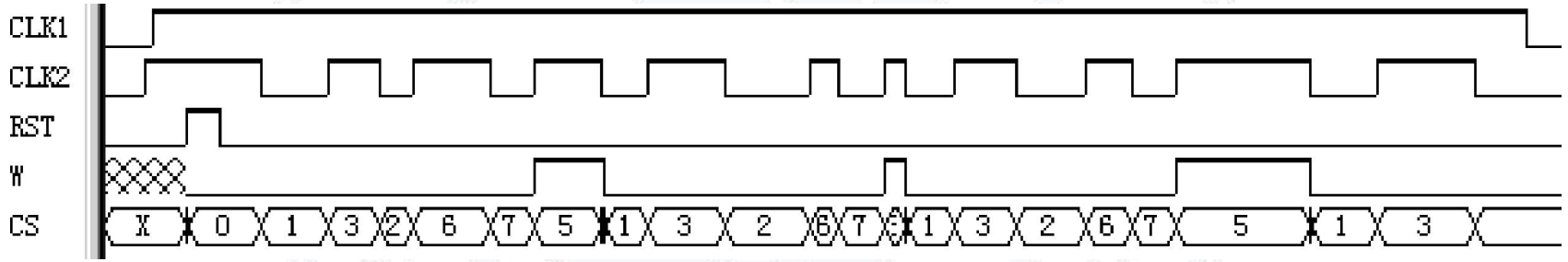


图10-18　例10-13的工作时序图

10.6 安全状态机设计

除了满足功能特性和速度，有限状态机的安全性和稳定性是重要的技术指标。

表10-4中无剩余状态。但在很多时候，总是不可避免地出现剩余状态（即未被定义的编码组合）。

以例10-1为例，该程序共定义了5个合法状态（有效状态），即s0,s1,s2,s3,s4。如果使用顺序编码方式指定状态，则最少需3个触发器，这样最多8种可能状态，编码方式如表10-4所示，最后3个状态s5,s6,s7都是非法状态。

因此，为了使状态机有可靠的工作性能，必须设法使系统在任何不利情况下都在落入这些非法状态后还能返回正常的状态转移路径中。

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

10.6 安全状态机设计



10.6.1 状态导引法

在状态元素定义中针对所有的状态，包括多余状态都作出定义，并在以后的语句中加以处理。即在语句中对每一个非法状态都作出明确的状态转换指示。

```
...
s5 :    next_state = s0;
s6 :    next_state = s0;
s7 :    next_state = s0;
default : begin  next_state=s0;
```

这种方法的优点是直观可靠，但缺点是可处理的非法状态少，如果非法状态太多，则耗用逻辑资源太大。所以只适合于顺序编码类状态机。

EDA Lab

10.6 安全状态机设计

10.6.2 状态编码监测法

对于一位热码，剩余状态较多，如果采用状态导引法较为复杂。

可以设计逻辑检测模块，加入对状态编码中1的个数是否大于1的监测判断逻辑。如将任一状态的编码相加，大于1，则必为非法状态，发出警告信号。

10.6 安全状态机设计



10.6.3 借助EDA工具自动生成安全状态机

需通过仿真，验证综合出的电路确实增加了安全措施；

另外，还可以利用属性语句，如（*syn_encoding="safe, one-hot"*）。

EDA Lab

10.7 硬件数字技术排除毛刺

在数字系统设计中，无论是进入还是输出FPGA的信号，还是数字系统内部，总是难免会有一些不希望的毛刺或随机干扰信号，它们的排除或避免常常成为数字系统设计工程师必须面对的棘手问题。

10.7 硬件数字技术排除毛刺

10.7.1 延时方式去毛刺

使用触发器或寄存器对信号进行适当延时，或延时采样，使处理过的信号在输出后能避开毛刺。

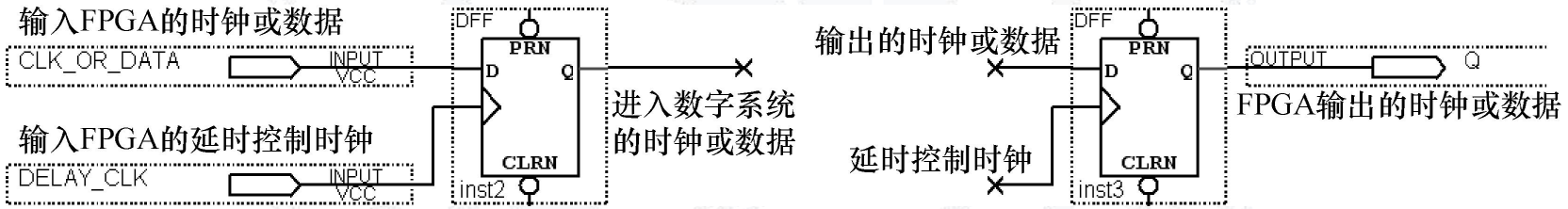


图10-19　单触发器输入(左图)和输出(右图)延时电路

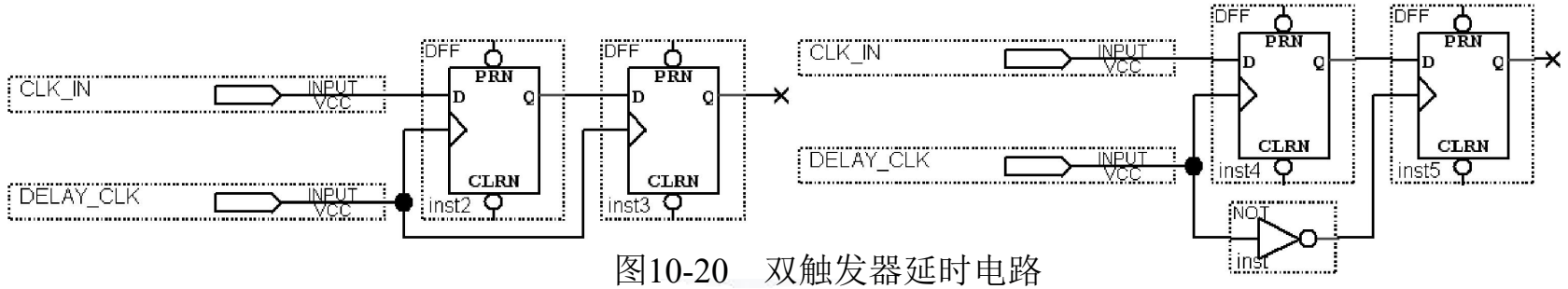


图10-20　双触发器延时电路

10.7 硬件数字技术排除毛刺

10.7.1 延时方式去毛刺

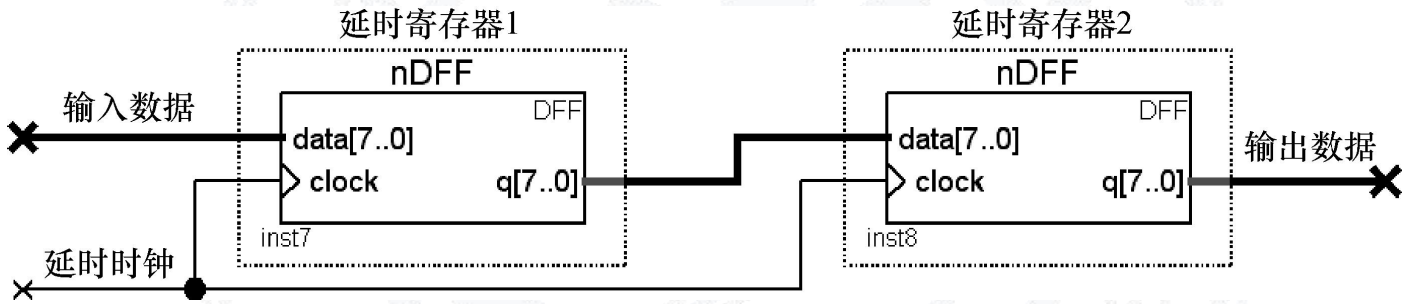


图10-21　双寄存器数据延时电路

该延时电路用意与前同，但主要针对总线数据通道的延时。

10.7 硬件数字技术排除毛刺

10.7.2 逻辑方式去毛刺

以上介绍的延时电路主要针对数据或时钟单边沿的毛刺，且延时时钟信号的周期应该与主通道的输入数据时序宽度或时钟的周期有较好的配合。对于双边延都有毛刺或抖动的情况效果不佳。可采用以下消抖电路的方法（功能上是一个信号滤波器）。

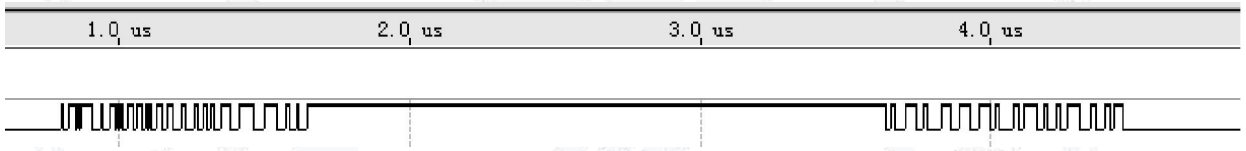


图10-22　信号上升与下降沿都含随机干扰抖动信号

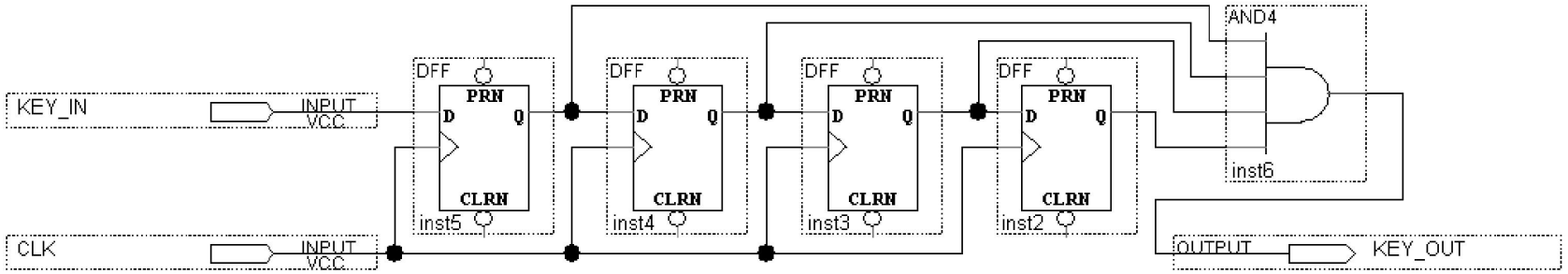


图10-23　消抖动电路

10.7 硬件数字技术排除毛刺

10.7.2 逻辑方式去毛刺

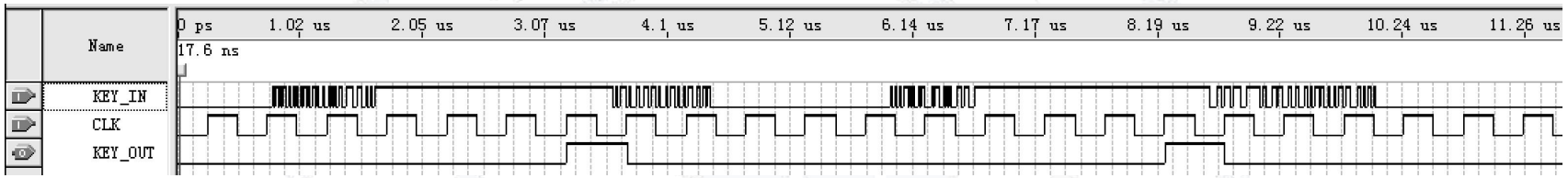


图10-24　消抖动电路仿真波形

（1）输出的脉宽变小了，它只等于CLK的一个周期宽度，且输出信号比原来的信号有一个明显的延迟。

（2）CLK的频率要足够高，否则所有信号都无法通过此电路。

（3）CLK的频率也不能太高，即其周期不能太小于干扰信号的脉宽。

（4）如果增加D触发器的数量，可提高滤波性能。

10.7 硬件数字技术排除毛刺

10.7.3 定时方式去毛刺

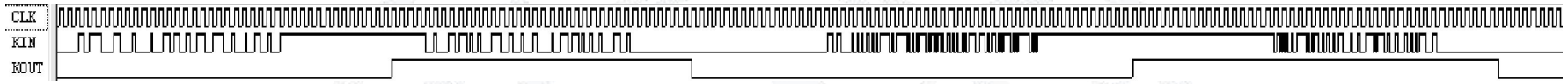


图10-25　例10-14消抖动电路仿真波形

【例10-14】

```
module ERZP  (CLK, KIN,KOUT);
   input   CLK, KIN;                          //工作时钟和输入信号
   output  KOUT;      reg KOUT;
   reg[3:0] KH,KL;                            //定义对高电平和低电平脉宽计数之寄存器
   always @(posedge CLK) if(!KIN) KL<=KL+1;//对键输入的低电平脉宽计数
          else  KL<=4'b0000;                  //若出现高电平，则计数器清0
   always @(posedge CLK) if(KIN) KH<=KH+1;//同时对键输入的高电平脉宽计数
           else KH<=4'b0000;                  //若出现高电平，则计数器清0
   always @(posedge CLK)  begin
          if(KH>4'b1100) KOUT<=1'B1;   //对高电平脉宽计数一旦大于12，则输出1
    else if(KL>4'b0111) KOUT<=1'B0;  end //对低电平脉宽计数若大于7，则输出0
endmodule
```

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
- 10-3 数据采集模块设计
- 10-4 五功能智能逻辑笔设计
- 10-5 通用异步收发器UART设计
- 10-6 硬件消抖动电路设计

EDA Lab

10-2 ADC采样控制电路设计

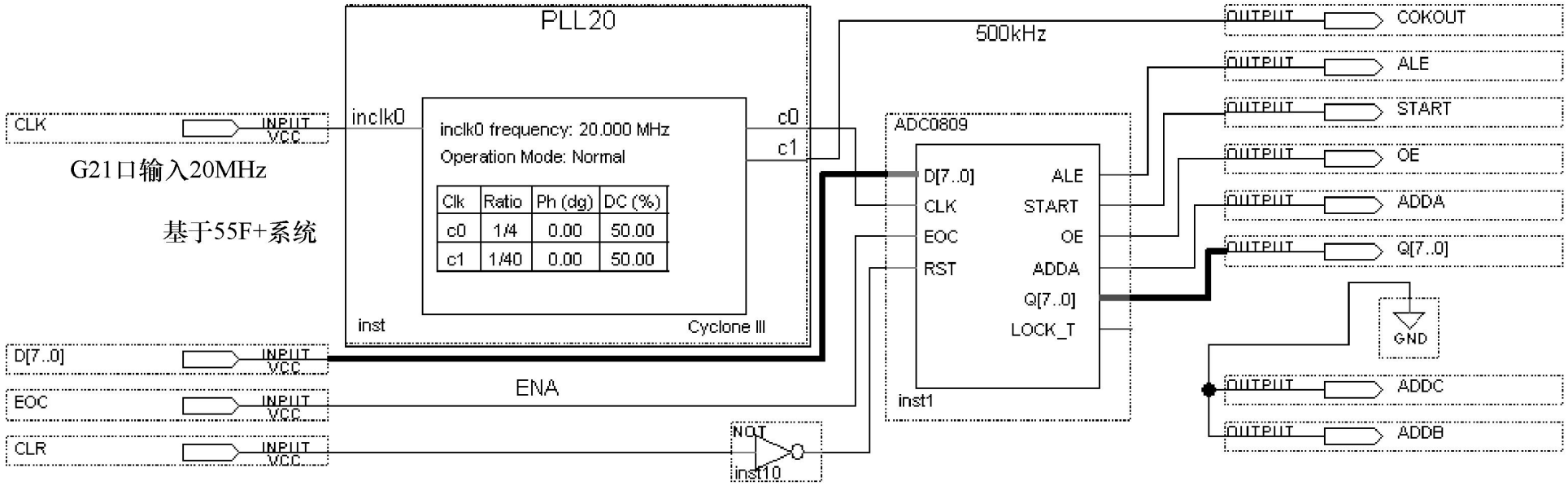


图10-26 ADC0809采样控制实验电路

10-3 数据采集模块设计

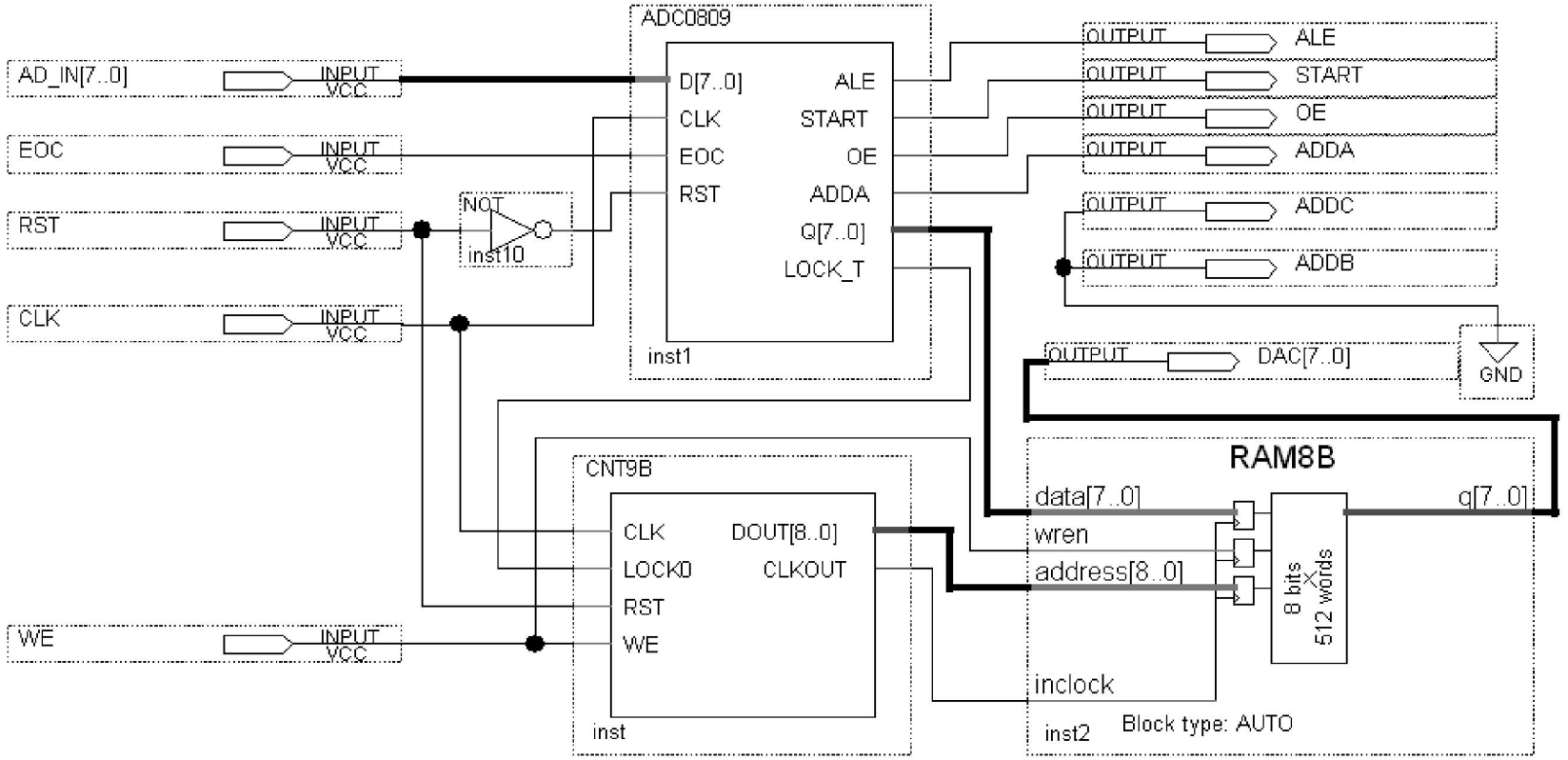


图10-27 ADC0809采样电路及简易存储示波器控制系统

10-3 数据采集模块设计

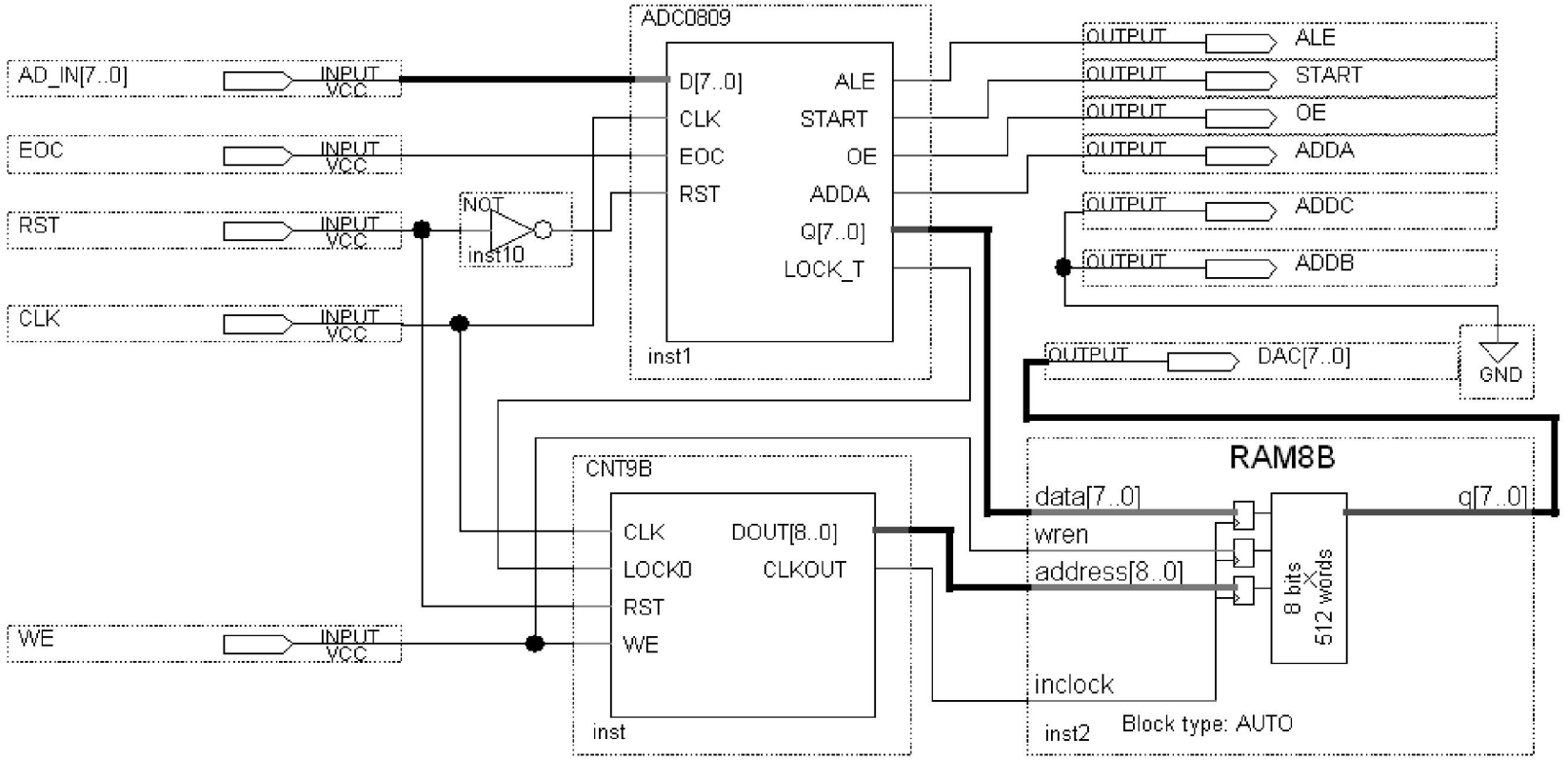


图10-27 ADC0809采样电路及简易存储示波器控制系统

10-3 数据采集模块设计

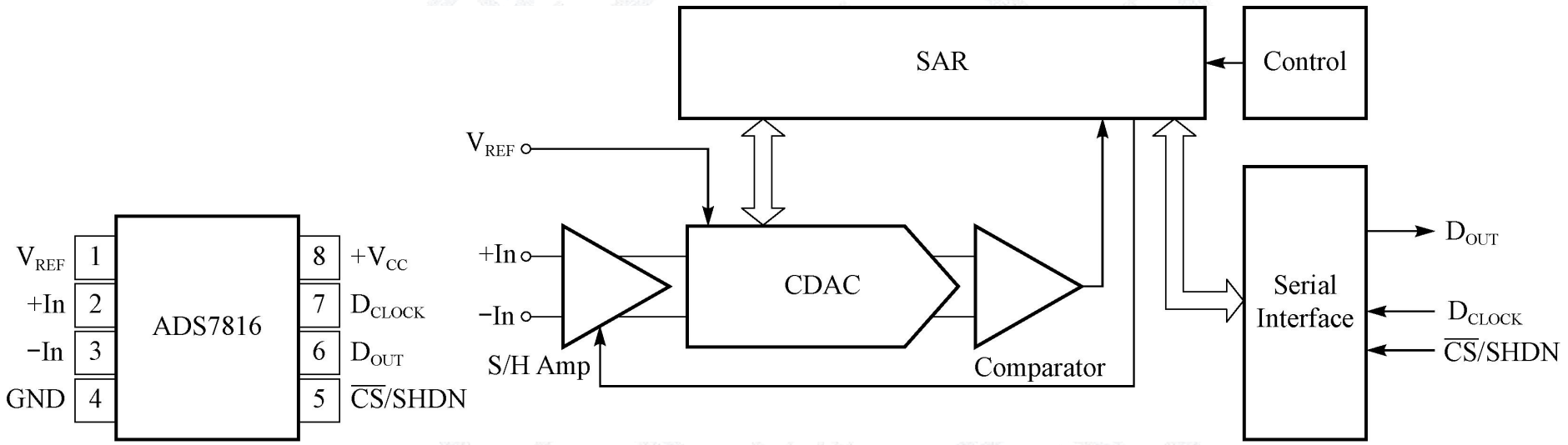


图10-28　ADS7816的引脚和结构

10-3 数据采集模块设计

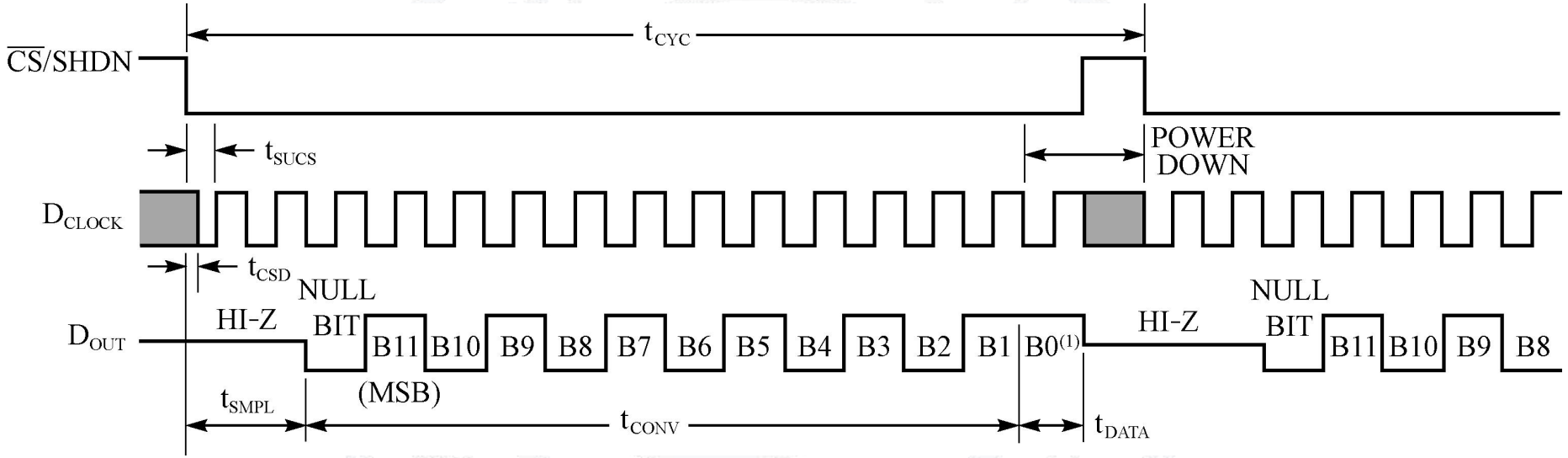


图10-29 ADS7816的工作时序图

10-4 五功能智能逻辑笔设计

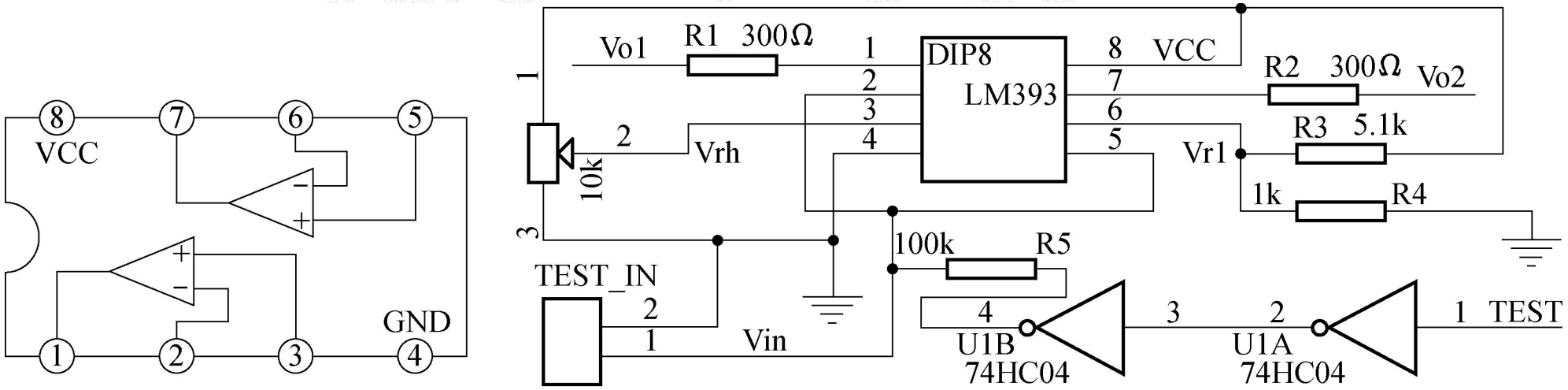


图10-30 五功能智能逻辑笔电平信号采样电路，左图是LM393引脚图



【例10-15】五功能智能逻辑笔参考程序

```
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: if(VO==2'b01)  NST<=s5; else NST<=s7;
      s5: if(VO==2'b01)  NST<=s6; else NST<=s7;
      s6: if(VO==2'b01)  begin LED<=4'b0001; NST<=s0; end  else NST<=s7;
      s7: if(VO==2'b10)  NST<=s8; else NST<=s10;
      s8: if(VO==2'b10)  NST<=s9; else NST<=s10;
      s9: if(VO==2'b10)  begin LED<=4'b0010; NST<=s0; end  else NST<=s10;
     s10: if(VO==2'b11)  NST<=s11; else NST<=s13;
     s11: if(VO==2'b11)  NST<=s12; else NST<=s13;
     s12: if(VO==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
```

10-5 通用异步收发器UART设计

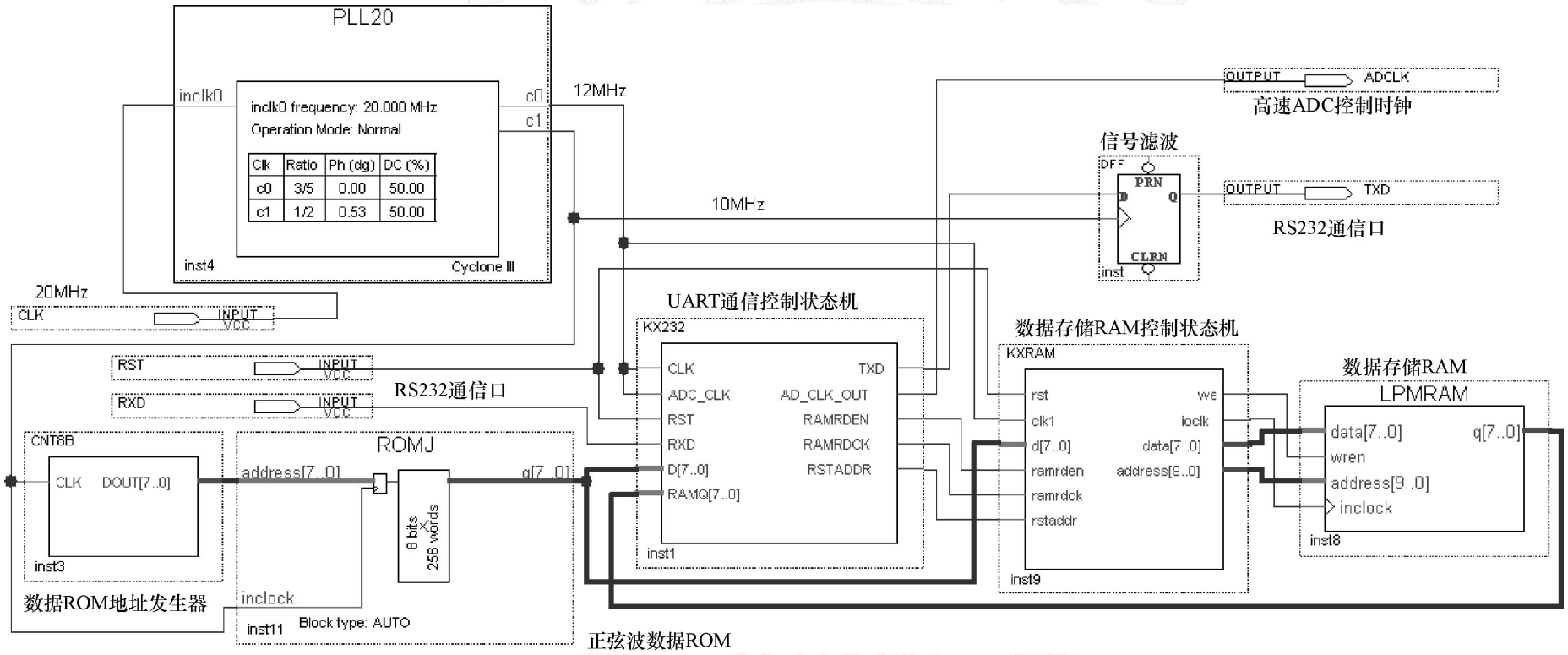


图10-31　UART通信设计顶层电路

10-5 通用异步收发器UART设计

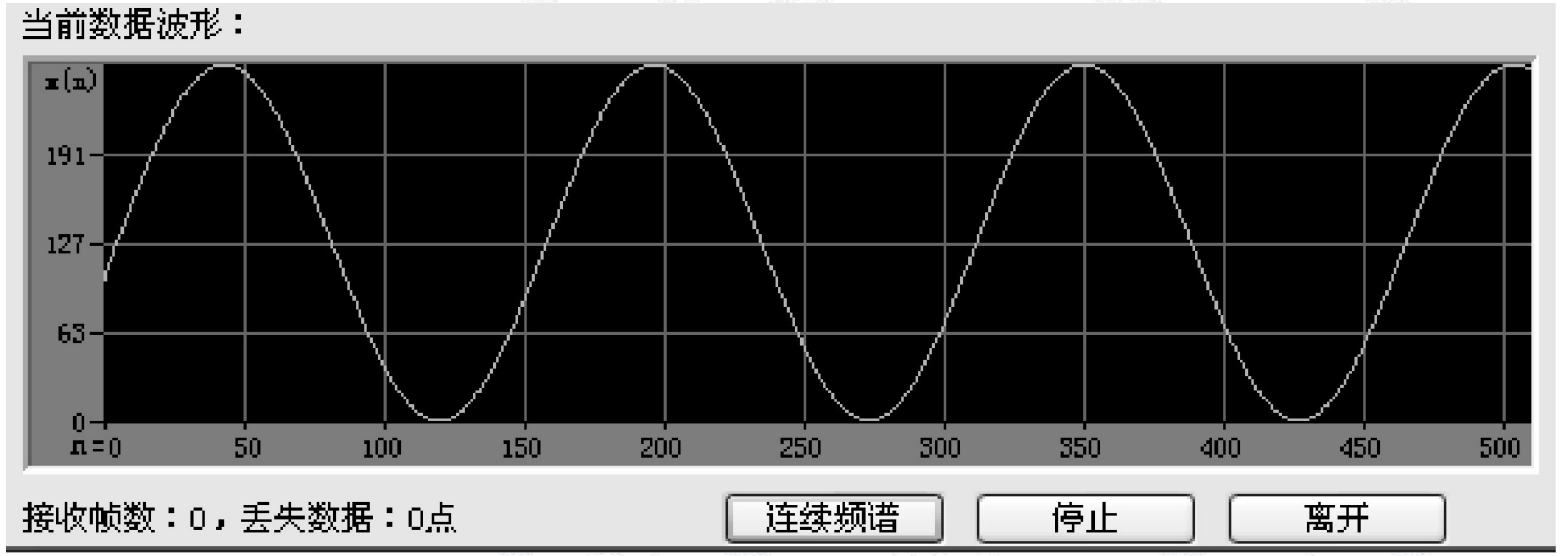


图10-32　通过RS232通信来自FPGA的正弦波信号