数字系统综合设计实践

第 10 章 Verilog状态机设计

主讲教师： 邱德慧

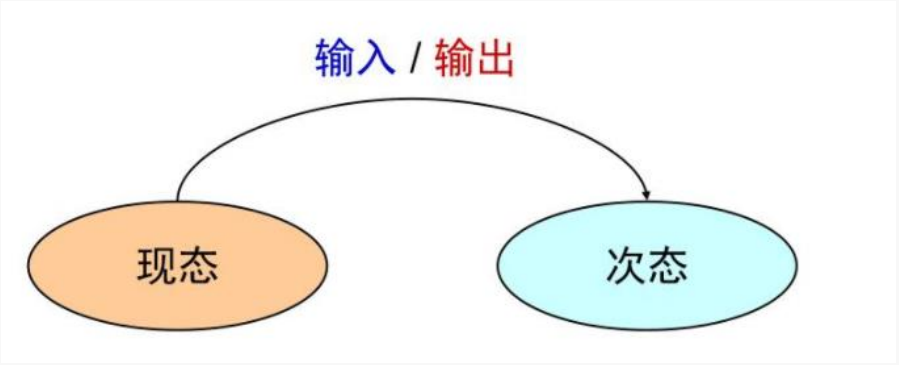
10.1 Verilog状态机的一般形式

为什么要使用状态机

- **高效的顺序控制模型：**克服了纯硬件数字系统顺序方式控制不灵活的缺点
- **性能稳定：**易构成性能良好的同步时序逻辑模块
- **设计实现效率高：**结构模式简单、层次分明、易读易懂、易排错
- **高速性能：**在高速运算和控制方面，有其巨大的优势
- **高可靠性：**非法状态易控制

什么是有限状态机(Finite State Machine)

- 有限状态机是由寄存器组和组合逻辑构成的硬件时序电路；
- 状态（即由寄存器组的1和0的组合状态所构成的有限个状态）只能在同一时钟跳变沿的情况下才能从一个状态转向另一个状态；
- 究竟转向哪一状态不但取决于各个输入值还取决于当前状态；
- 状态机可用于产生在时钟跳变沿时刻开关的复杂的控制逻辑，是数字逻辑的控制核心。

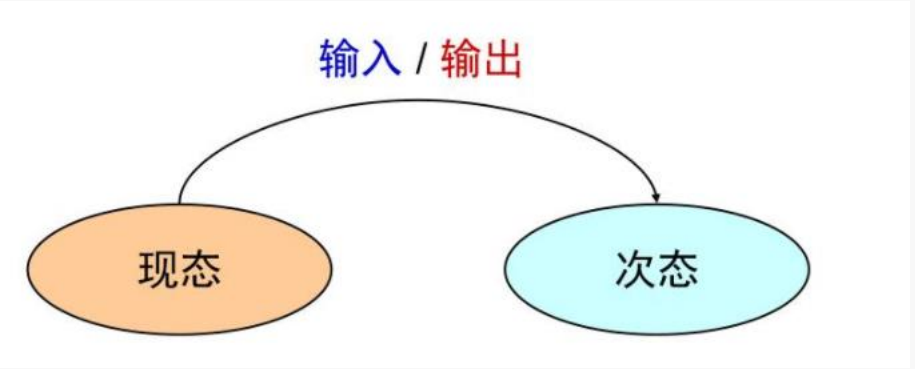


10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

有限状态机三要素

- 状态（当前状态，下一个状态）
- 输入信号（事件）
- 输出控制信号（相应操作）



10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

1. 说明部分

各元素状态用参数关键词parameter来定义

```
parameter[2:0] s0=0,s1=1,s2=2,s3=3,s4=4;
reg[2:0] current_state, next_state;
```

Parameter旁的位宽可写可不写

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

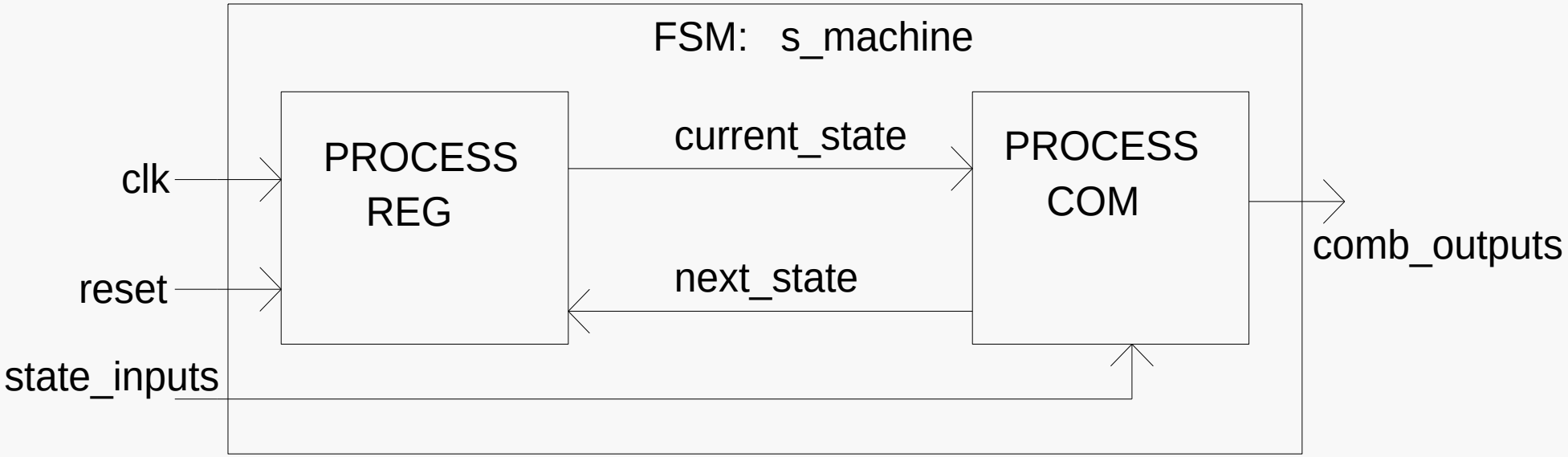


图10-1 一般状态机结构框图

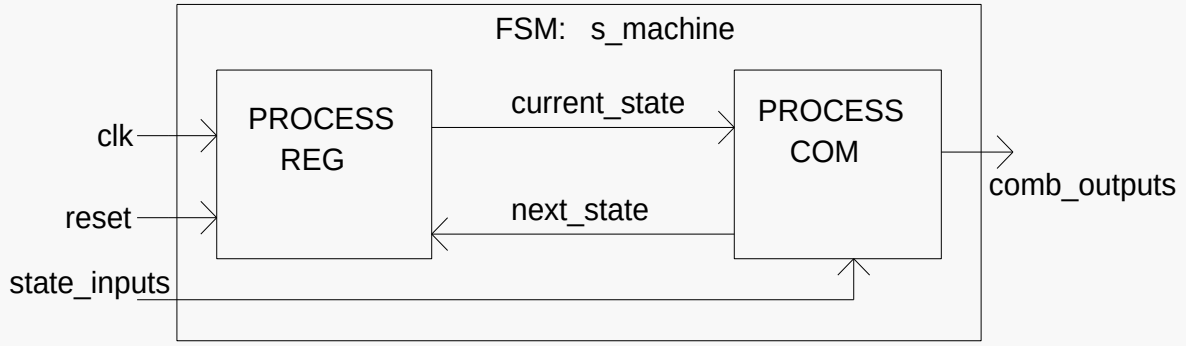
2. 主控时序过程:在时钟驱动下负责状态转换

3. 主控组合过程:给出状态机的输出和确定次态

4. 辅助进程:配合 例如： 数据锁存器

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构




```
1  module FSM_EXP (clk, reset, state_inputs, comb_outputs)
2  input clk;
3  input reset;
4  input[1:0] state_inputs;
5  output[3:0] comb_outputs
6  reg[3:0] comb_outputs;
7  parameter s0=0,s1=1,s2=2,s3=3,s4=4,s5=5;
8  reg[4:0] c_st, next_state;
```

```
9   always @(posedge clk or negedge reset) begin
10    if (!reset) c_st<=s0;
11        else    c_st<=next_state;  end
```

10.1 Verilog状态机的一般形式

```
12  always @(c_st or state_inputs) begin
13    case (c_st)
14      s0: begin comb_outputs<=5;
15          if (state_inputs==2'b00) next_state<=s0;
16              else next_state<=s1;  end
17      s1: begin comb_outputs<=8;
18          if (state_inputs==2'b01) next_state<=s1;
19              else next_state<=s2;  end
20      s2: begin comb_outputs<=12;
21          if (state_inputs==2'b10) next_state<=s0;
22              else next_state<=s3;  end
23      s3: begin comb_outputs<=14;
24          if (state_inputs==2'b11) next_state<=s3;
25              else next_state<=s4;  end
26      s4: begin comb_outputs<=9; next_state<=s0; end
27      default: next_state<=s0;
28    endcase
29    end
30  endmodule
```

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

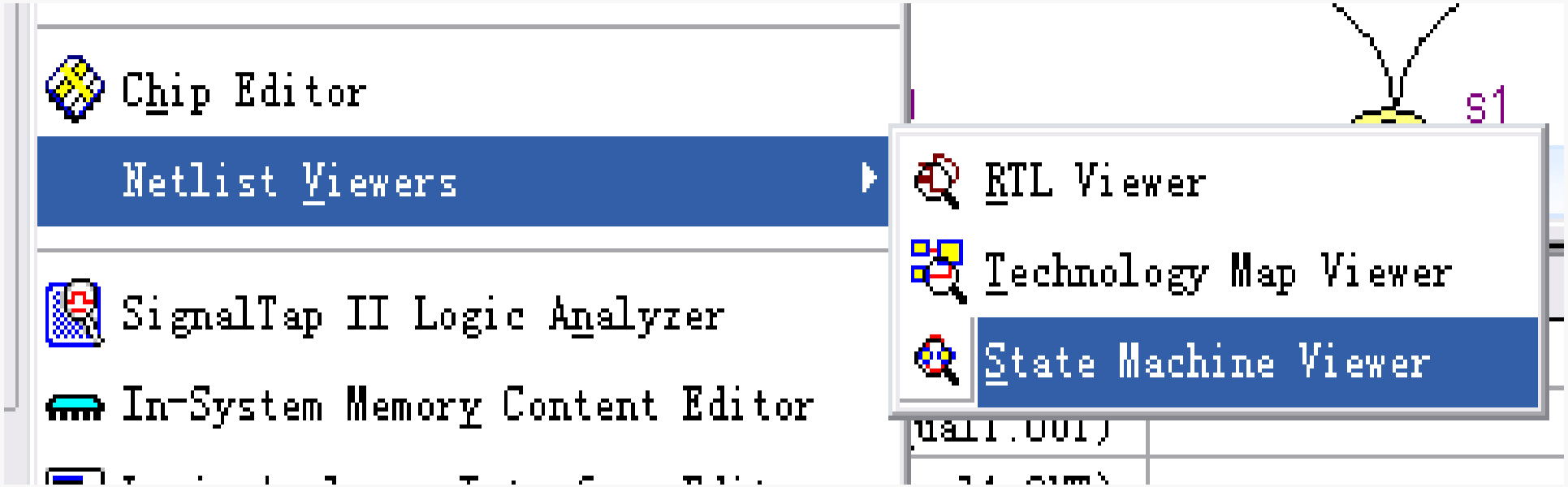


图10-2a 打开QuartusII状态图观察器

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

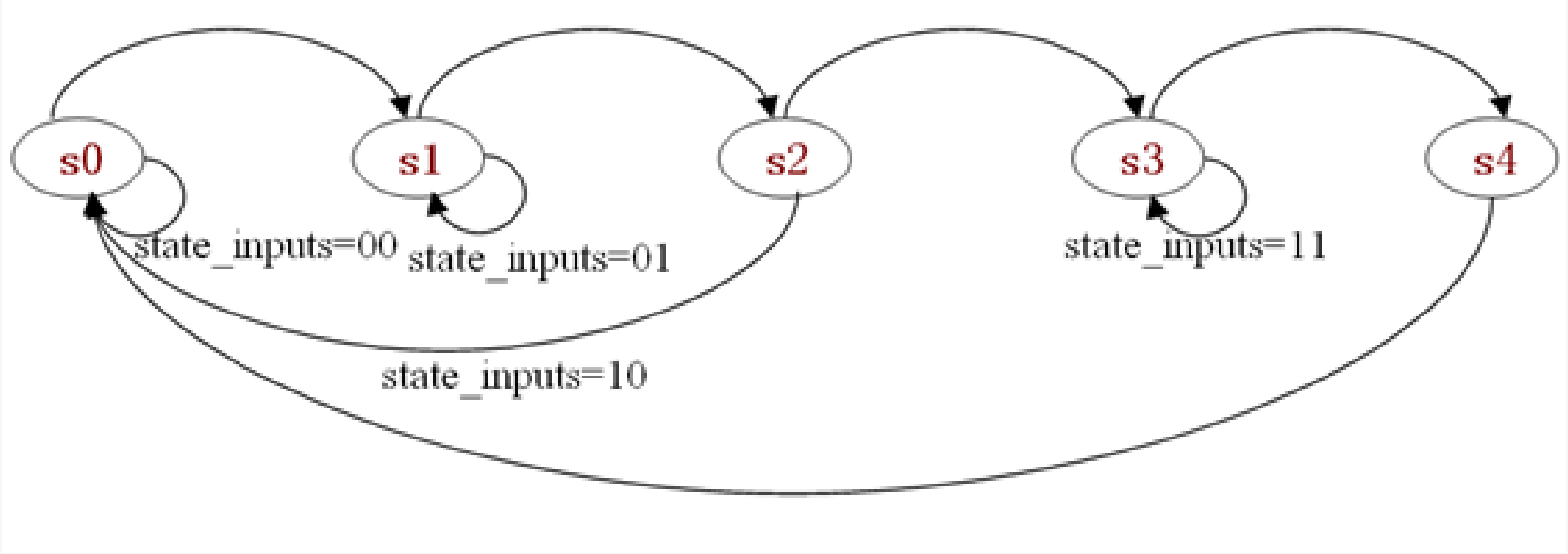


图10-2b 例10-1的状态转换图

10.1 Verilog状态机的一般形式

10.1.2 状态机的一般结构

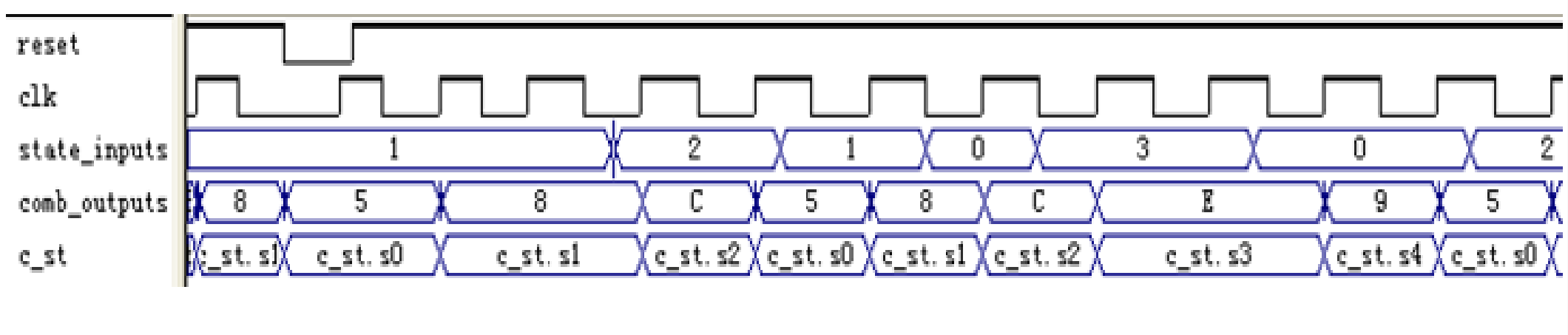


图10-3 例10-1状态机的工作时序

10.1 Verilog状态机的一般形式

10.1.3 有限状态机的分类

按照输出逻辑可以分为:

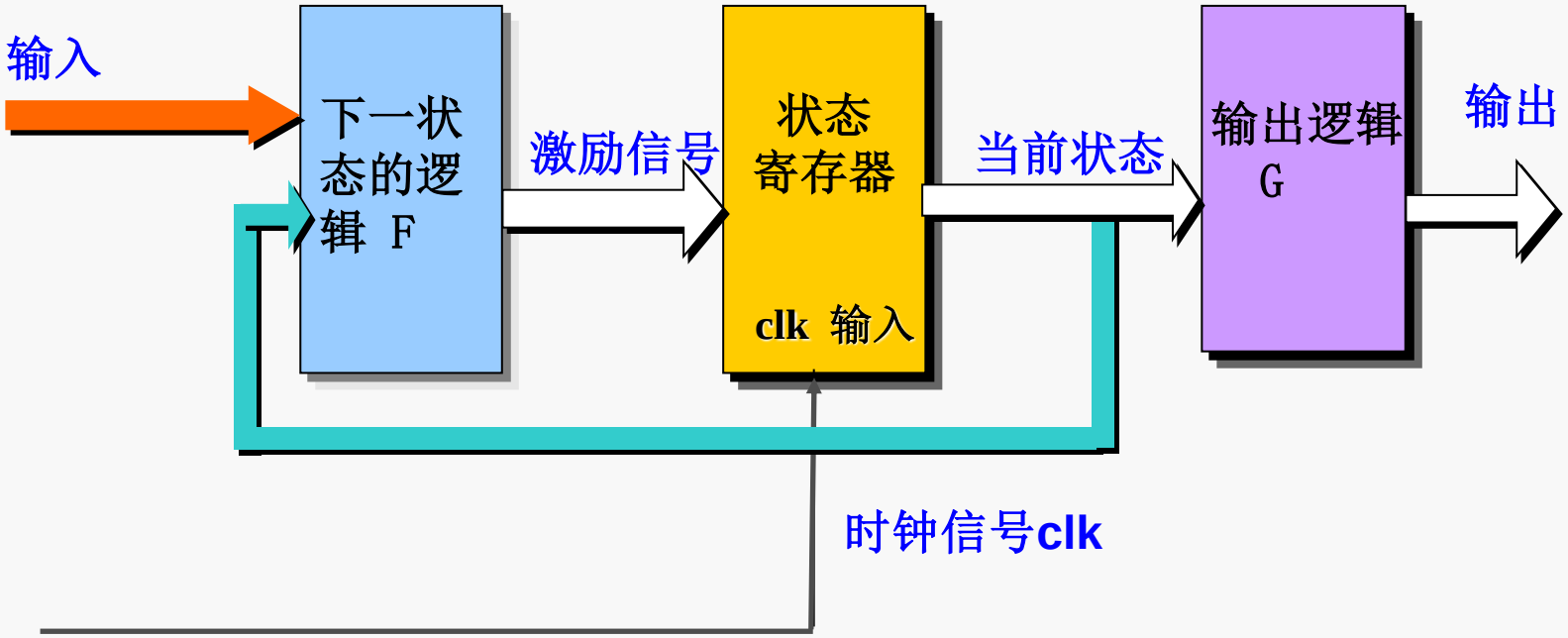
- Moore状态机：时序逻辑的输出只取决于当前状态。
- Mealy状态机：时序逻辑的输出不仅取决于当前状态，还取决于输入。

10.1 Verilog状态机的一般形式

- **Moore** 状态机

  下一个状态 = (当前状态，输入信号)

  输出信号 = (当前状态)



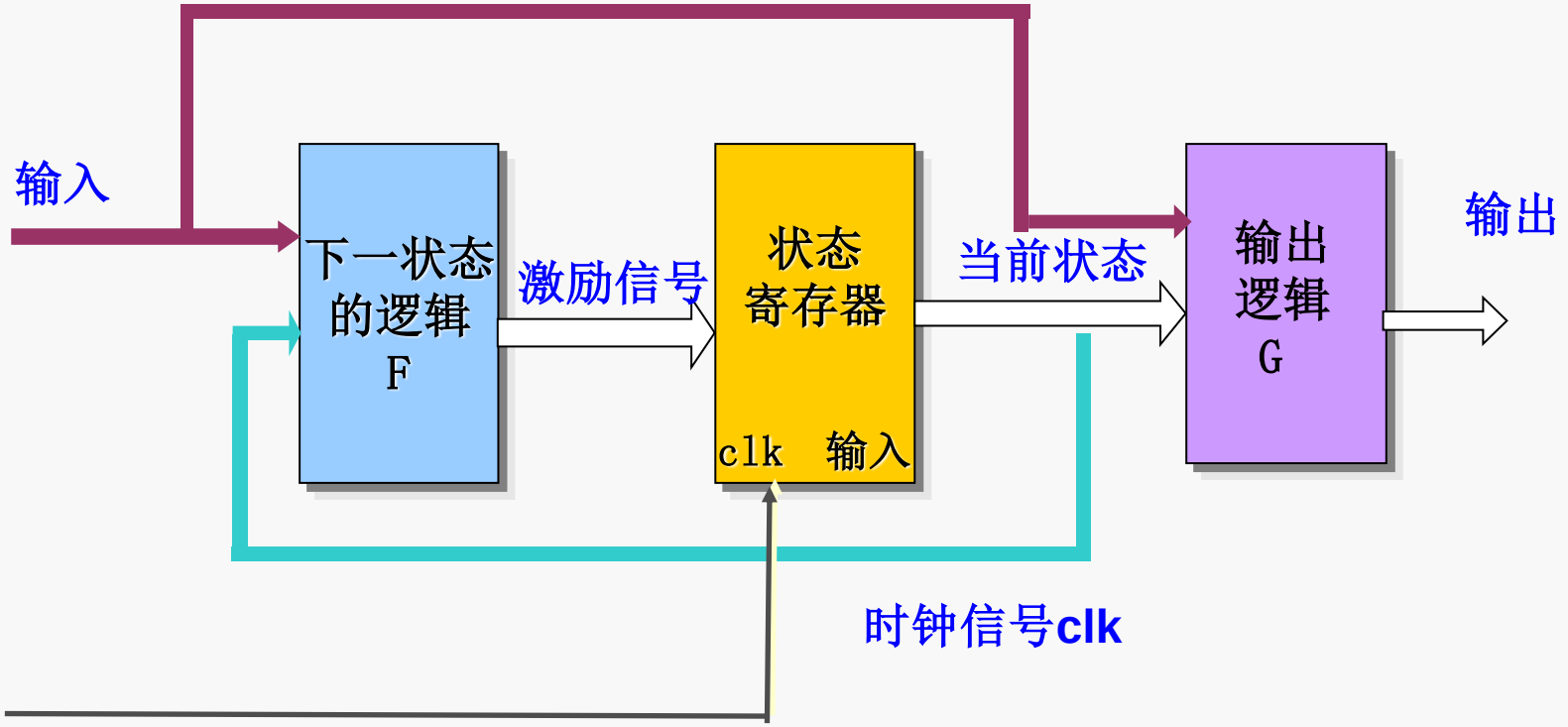
Moore有限状态机的结构框图

10.1 Verilog状态机的一般形式

- **Mealy** 状态机

  下一个状态 = (当前状态，输入信号)

  输出信号 = (当前状态，输入信号)



Mealy有限状态机的结构框图

10.2 Moore型有限状态机设计

10.2.1 多进程有限状态机

例：用状态机设计**AD0809**采样控制器

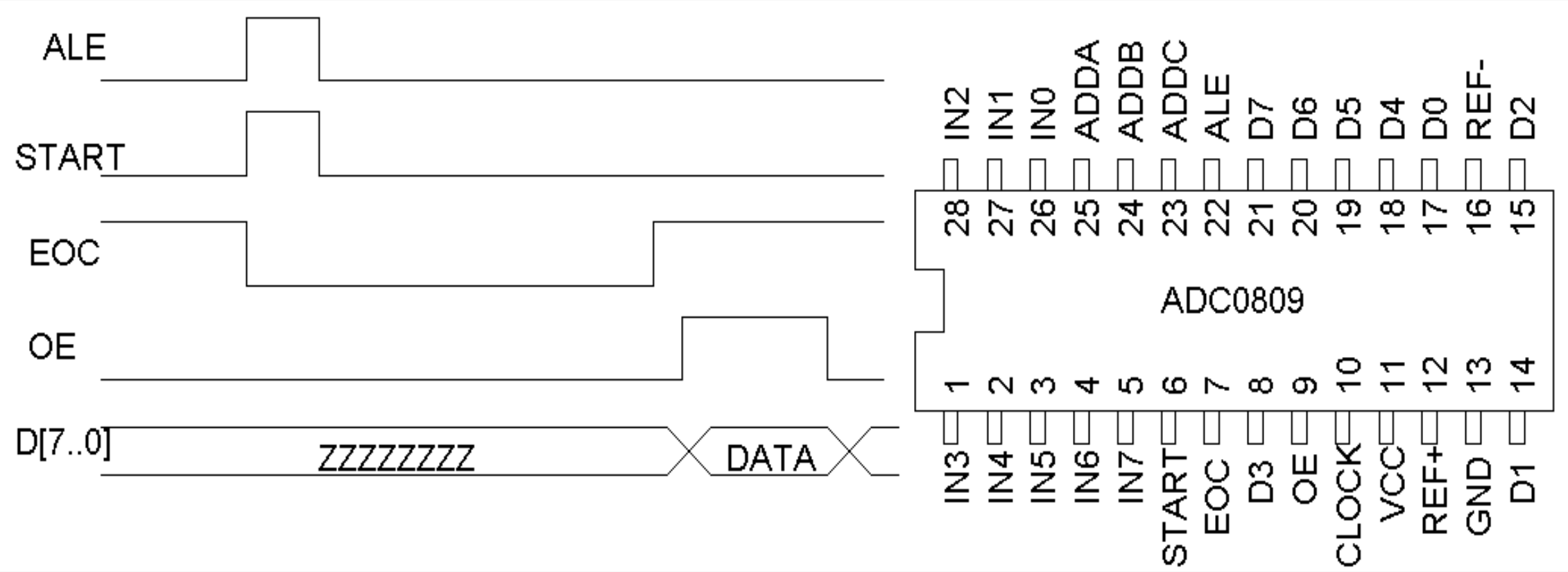
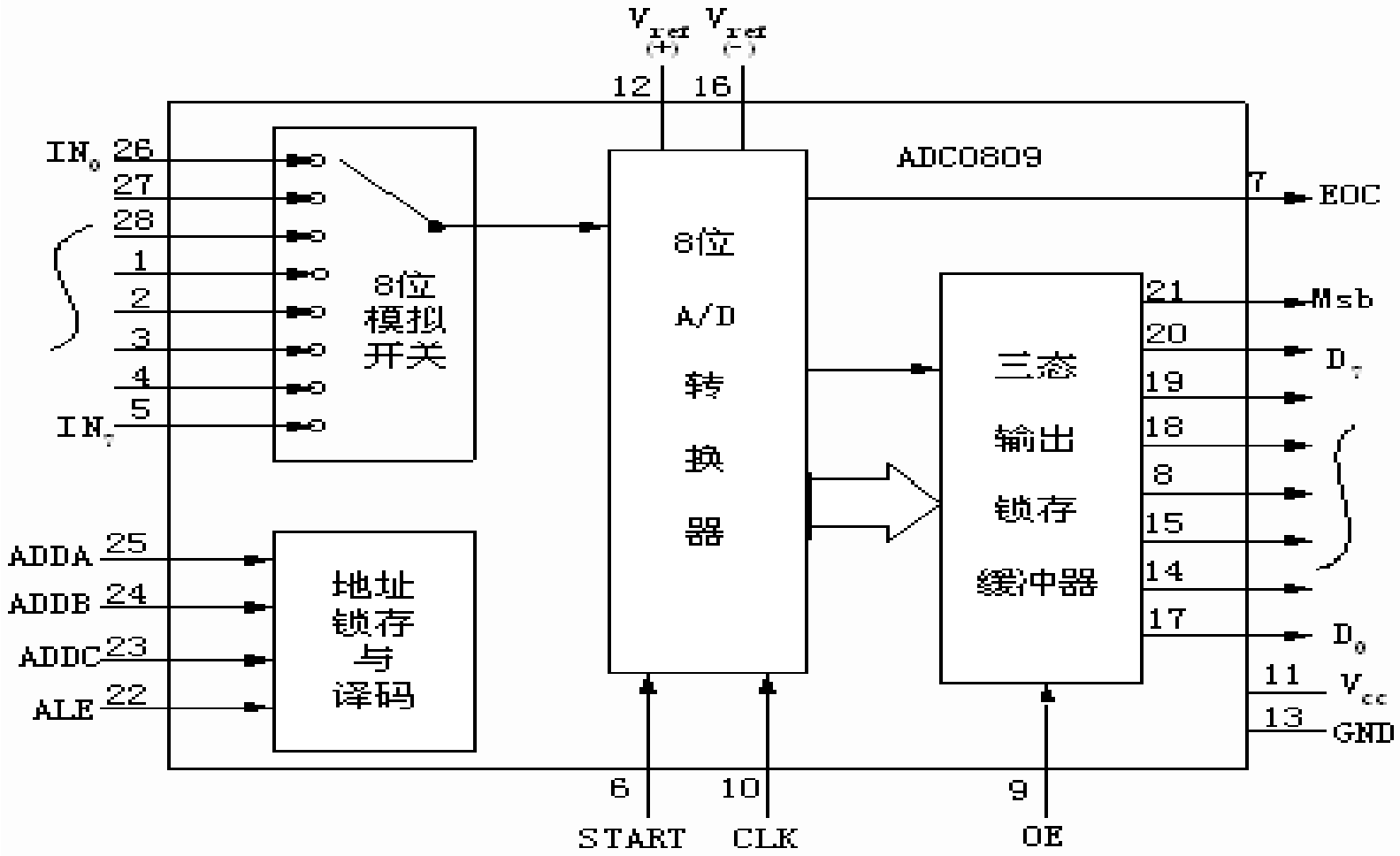


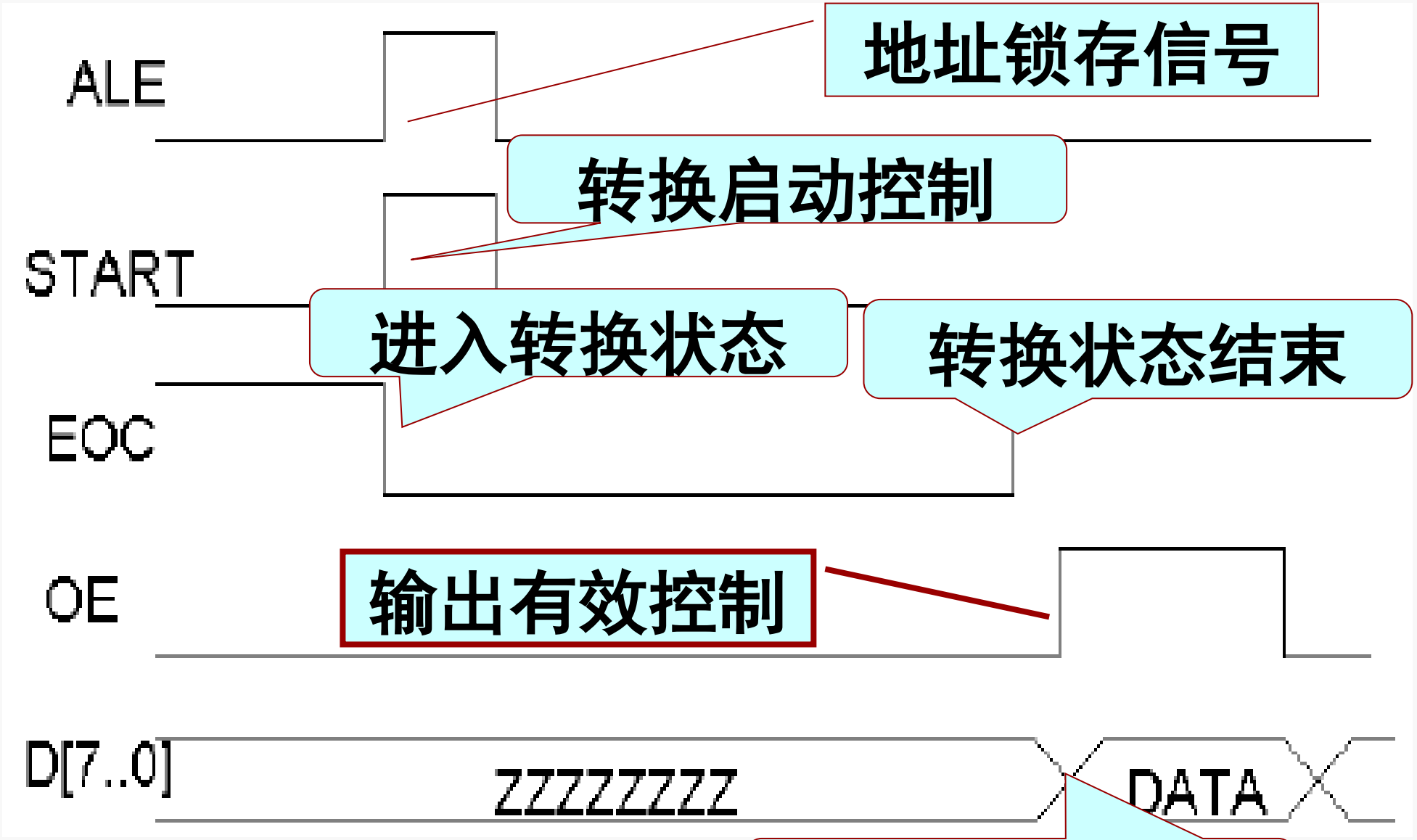
图10-5 ADC0809工作时序

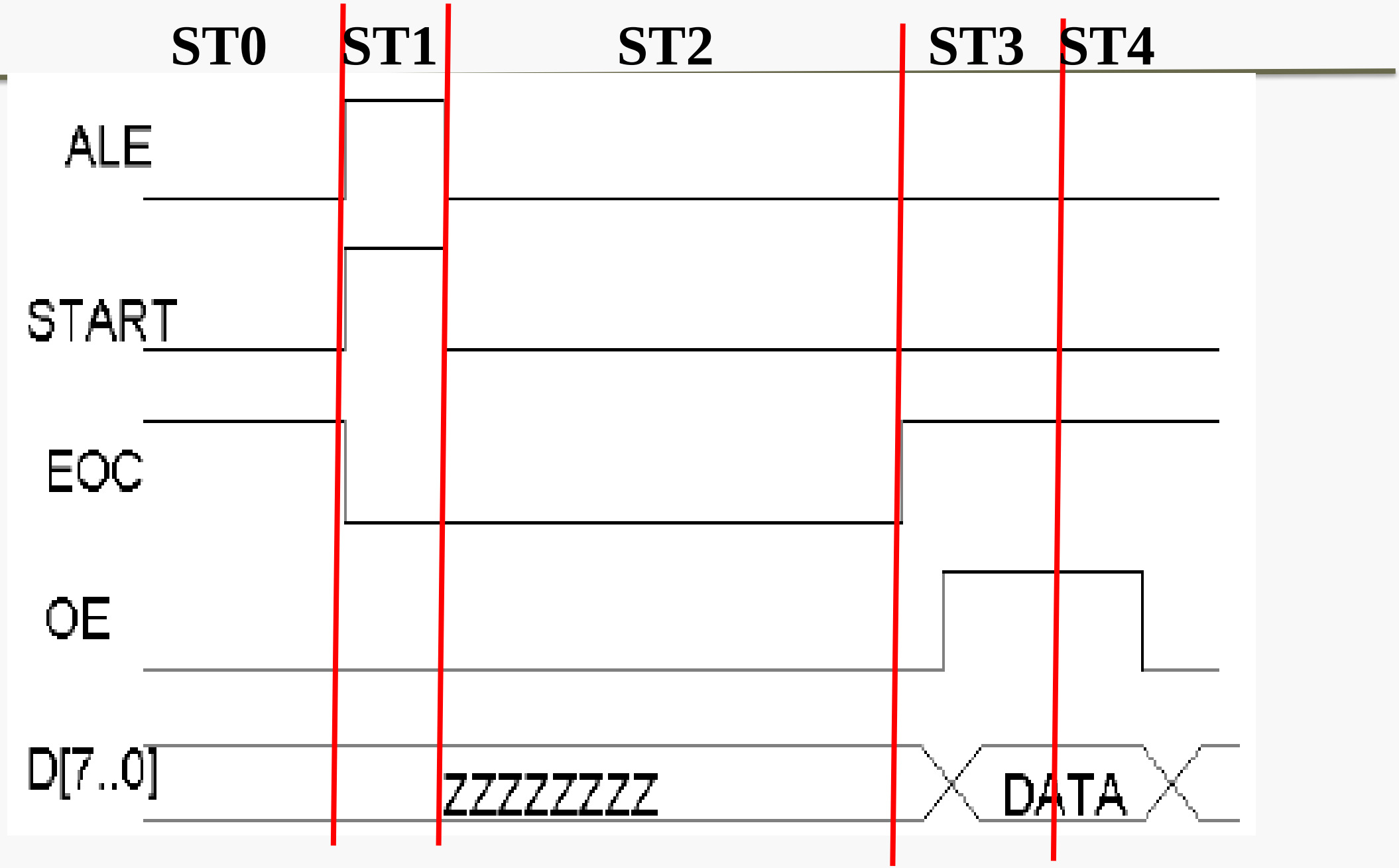
10.2 Moore型有限状态机设计



ADC0809引脚图与接口电路

图 ADC0809工作时序




ST0
ST1
ST2
ST3
ST4
ALE
START
EOC
OE
D[7..0]
ZZZZZZZZ
DATA

10.2 Moore型有限状态机设计

10.2.1 多进程有限状态机

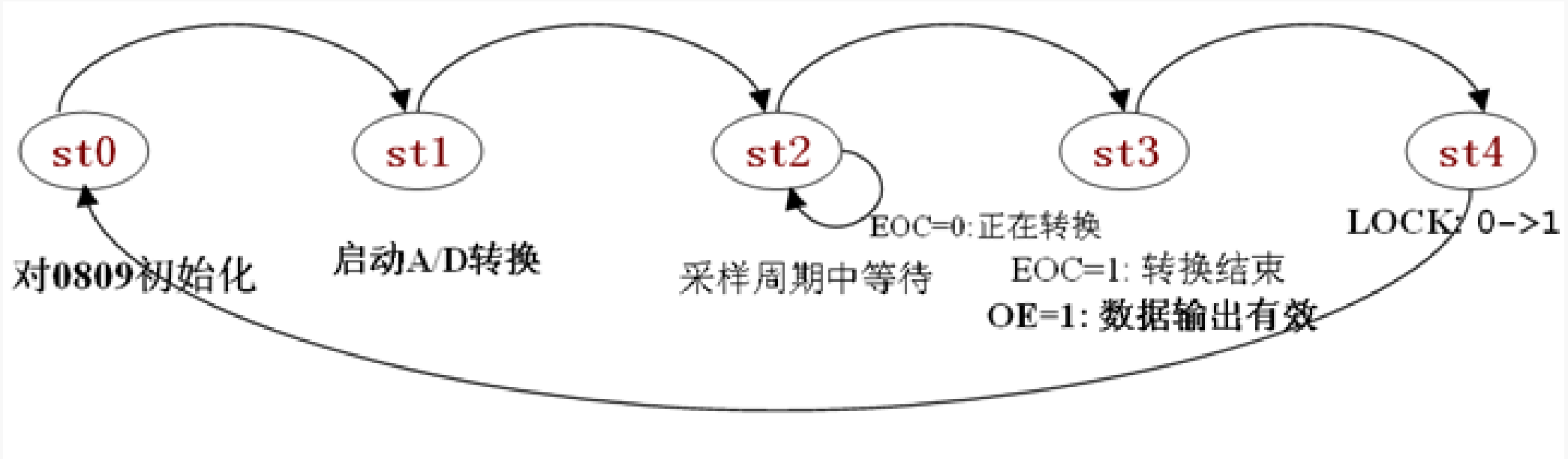


图10-6 控制ADC0809采样状态图

10.2.1 多进程有限状态机

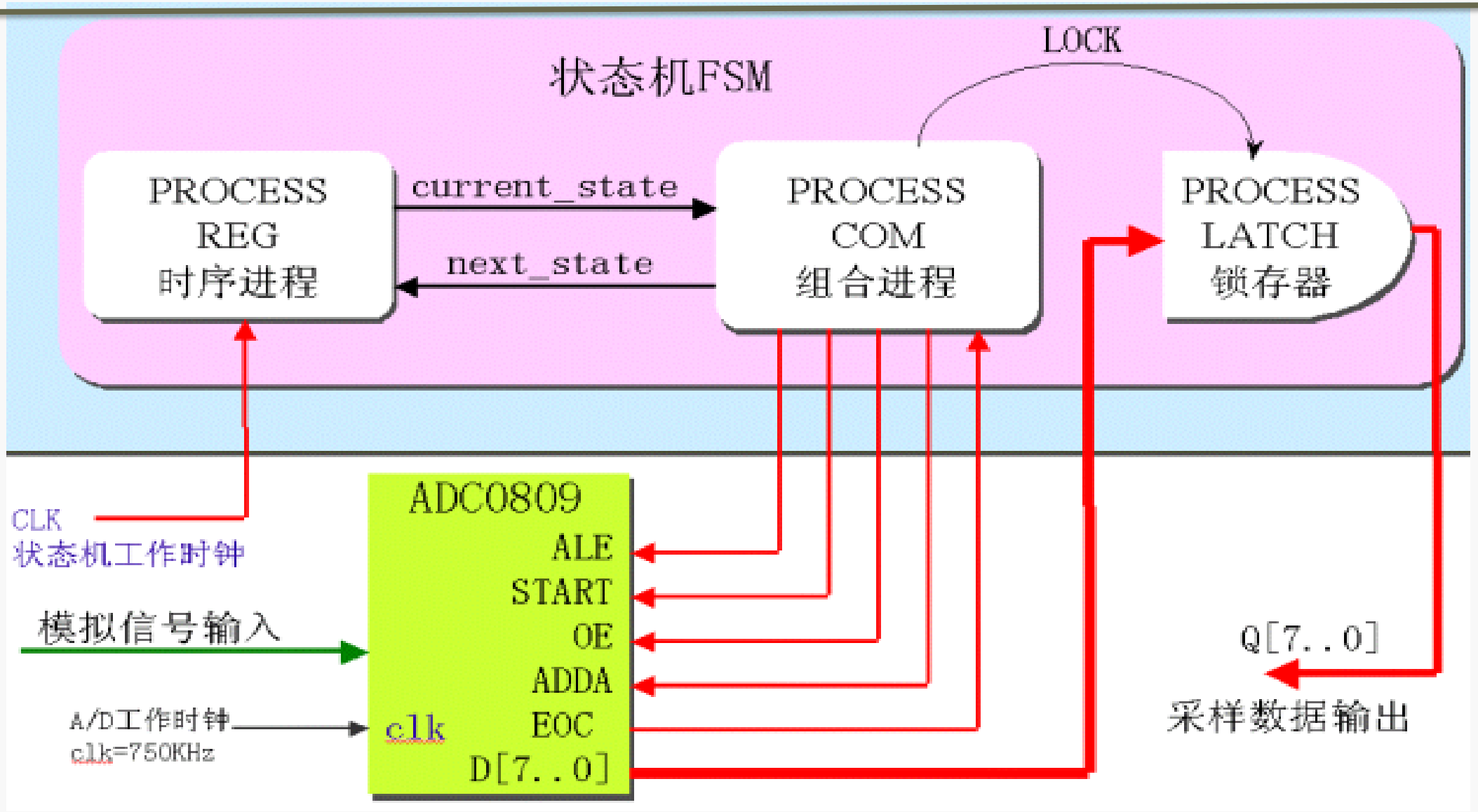


图10-7 采样状态机结构框图

10.2.1 多进程有限状态机

```
1  module adc0809 (D,CLK,EOC,RST,ALE,START,OE,ADDA,Q,LOCK_T);
2    input[7:0] D;                  //来自0809转换好的8位数据
3    input CLK,RST;                 //状态机工作时钟，和系统复位控制
4    input EOC;                     //转换状态指示，低电平表示正在转换
5    output ALE;                    //8个模拟信号通道地址锁存信号
6    output START,OE;               //转换启动信号，和数据输出三态控制信号
7    output ADDA,LOCK_T;            //信号通道控制信号和锁存测试信号
8    output[7:0]Q;  reg ALE,START,OE;
9    parameter s0=0,s1=1,s2=2,s3=3,s4=4; //定义各状态子类型
10   reg[4:0] cs, next_state;             //为了便于仿真显示，现态名简为cs
11   reg[7:0] REGL; reg LOCK;             //转换后数据输出锁存时钟信号
```

```
parameter s0=0,s1=1,s2=2,s3=3,s4=4;
reg[4:0] cs,next_state;
```

10.2.1 多进程有限状态机

```
13  always@(cs or EOC ) begin//组合过程，规定各状态转换方式
14    case (cs)
15      s0 : begin  ALE=0; START=0; OE=0; LOCK=0;
16                  next_state <= s1; end              //0809初始化
17      s1 : begin  ALE=1; START=1; OE=0; LOCK=0;
18                  next_state <= s2; end              //启动采样信号START
19      s2 : begin  ALE=0; START=0; OE=0; LOCK=0;
20            if(EOC==1'b1) next_state <= s3;         //EOC=0表明转换结束
21            else next_state <= s2; end              //转换未结束，继续等待
22      s3 : begin  ALE=0; START=0; OE=1; LOCK=0;     //开启OE，打开AD数据口
23                  next_state <= s4; end             //下一状态无条件转向s4
24      s4 : begin  ALE=0; START=0; OE=1; LOCK=1;     //开启数据锁存信号
25                  next_state <= s0; end
26      default:    begin ALE=0; START=0; OE=0; LOCK=0;
27                  next_state <= s0; end
28    endcase
29  end
```

10.2.1 多进程有限状态机

```
30  always @(posedge CLK or posedge RST) begin  //时序过程
31    if(RST) cs <= s0;
32       else cs <= next_state; end
```

```
33  always @(posedge LOCK)
34    if(LOCK) REGL <= D;              //在LOCK上升沿将转换好的数据锁入
35  assign ADDA= 0; assign Q= REGL;  //选择模拟信号进入通道IN0
36  assign LOCK_T= LOCK;             //将测试信号输出
37  endmodule
```

10.2 Moore型有限状态机设计

10.2.1 多进程有限状态机

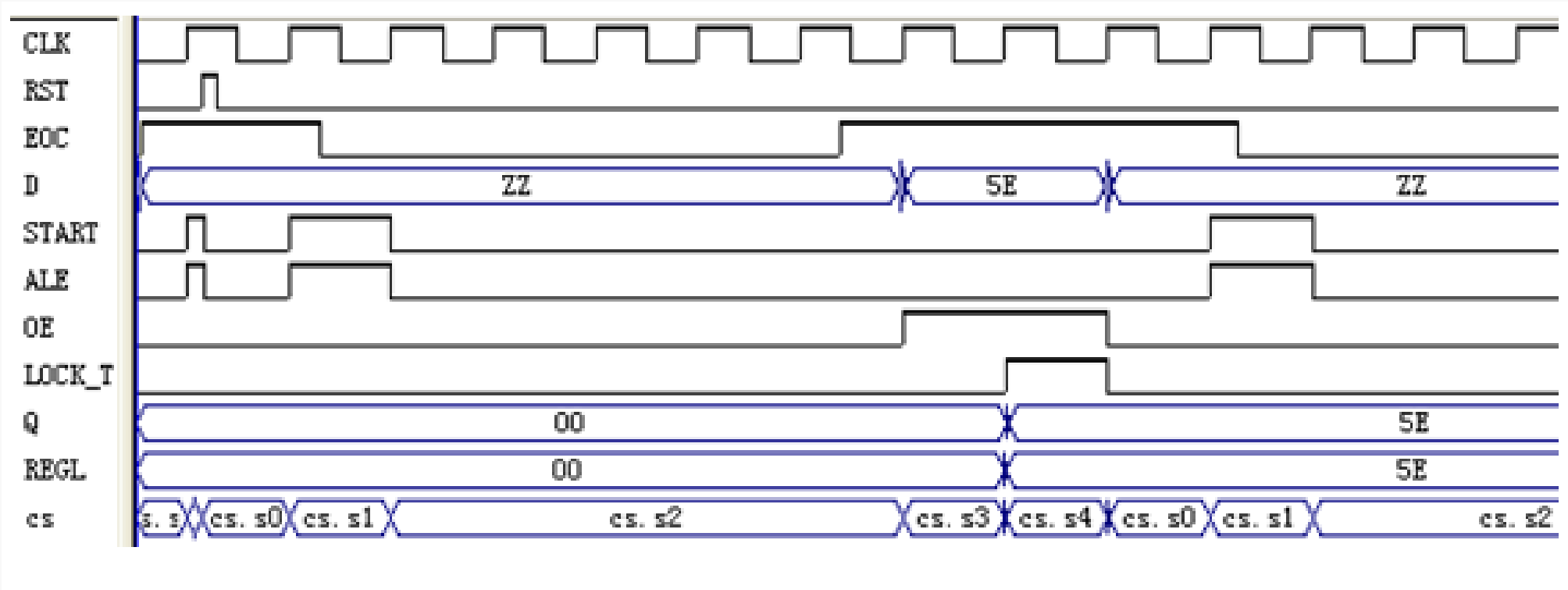


图10-9 ADC0809采样状态机工作时序

10.2 Moore型有限状态机设计

例10-3

```
1  always @(cs or EOC) begin
2    case (cs)
3      s0: next_state <= s1;
4      s1: next_state <= s2;
5      s2: if (EOC==1'b1) next_state <= s3; else next_state <= s2;
6      s3: next_state <= s4;
7      s4: next_state <= s0;
8      default: next_state <= s0;
9    endcase
10   end
```

```
12 always @(cs) begin
13   case(cs)
14       s0: begin ALE=0; START=0; OE=0; LOCK=0; end
15       s1: begin ALE=1; START=1; OE=0; LOCK=0; end
16       s2: begin ALE=0; START=0; OE=0; LOCK=0; end
17       s3: begin ALE=0; START=0; OE=1; LOCK=0; end
18       s4: begin ALE=0; START=0; OE=1; LOCK=1; end
19  default: begin ALE=0; START=0; OE=0; LOCK=0; end
20    endcase
21    end
```

10.2 Moore型有限状态机设计

10.2.2 序列检测器及其状态机设计

例10-4

```
1  module SCHK(input CLK,DIN,RST, output OUT);
2    parameter s0=40, s1=41, s2=42, s3=43, 84=44,
3              s5=45, s6=46, s7=47, s8=48; //设定9个状态参数
4    reg[8:0] ST,NST;              //代设定现态变量和次态变量
```

```
6     always @(posedge CLK or posedge RST)
7       if (RST) sT<=s0 ; else ST<=NST;
```

```
10      case(ST)
11         s0: if(DIN==1'b1) NST<=s1; else NST<=s0;
12         s1: if(DIN==1'b1) NST<=s2; else NST<=s0;
13         s2: if(DIN==1'b0) NST<=s3; else NST<=s0;
14         s3: if(DIN==1'b1) NST<=s4; else NST<=s0;
15         s4: if(DIN==1'b0) NST<=s5; else NST<=s0;
16         s5: if(DIN==1'b0) NST<=s6; else NST<=s0;
17         s6: if(DIN==1'b1) NST<=s7; else NST<=s0;
18         s7: if(DIN==1'b1) NST<=s8; else NST<=s0;
19         s8: if(DIN==1'b0) NST<=s3; else NST<=s0;
20         default: NST<=s0;
21      endcase
22    end
23    assign SOUT=(ST==S8);
24  endmodule
```

10.2 Moore型有限状态机设计

10.2.2 序列检测器及其状态机设计

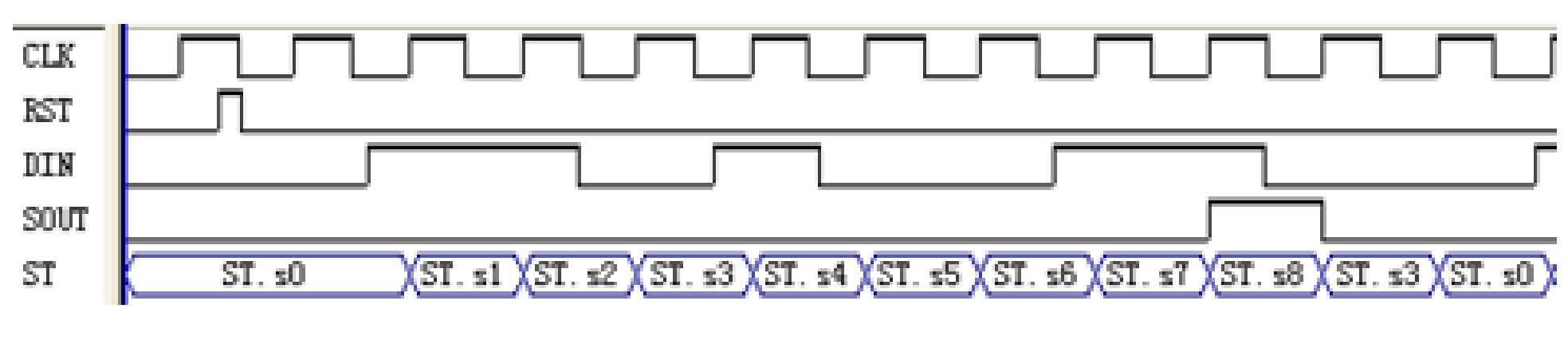


图10-9 例10-4序列检测器时序仿真波形

10.3 Mealy型状态机设计

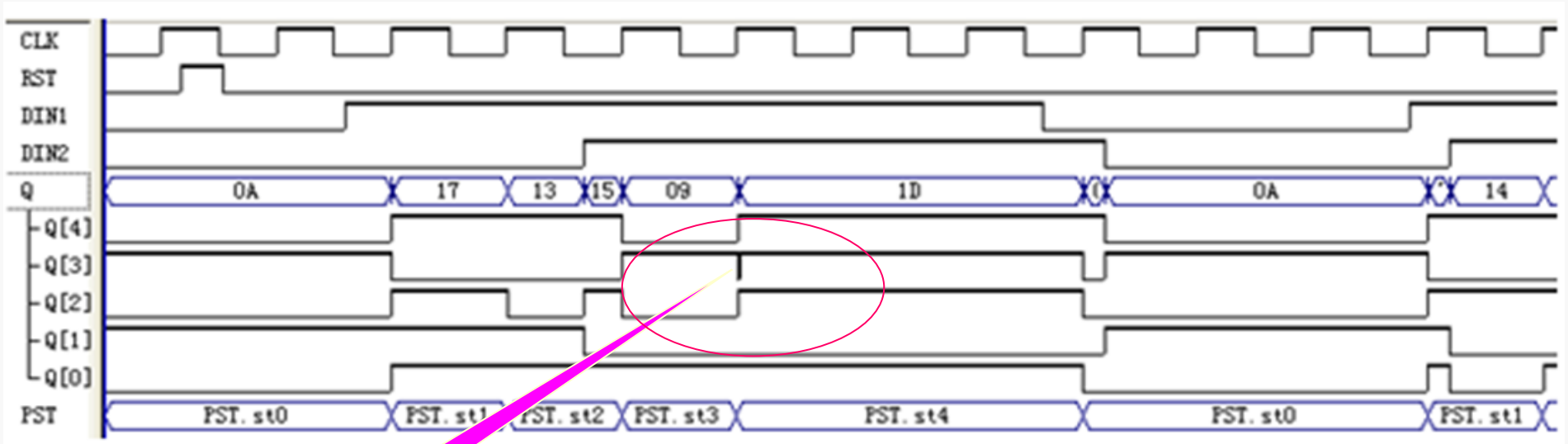
例10-5

```
1  module MEALY1(input CLK,DIN1,DIN2,RST, output reg [4:0] Q);
2    reg[4:0] PST;
3    parameter st0=0, st1=1, st2=2, st3=3, st4=4;
```

```
5  always @(posedge CLK or posedge RST) begin: REG
6  if (RST) PST<=st0; else begin
7     case(PST)
8        st0: if (DIN1==1'b1) PST<=st1; else PST<=st0;
9        st1: if (DIN1==1'b1) PST<=st2; else PST<=st1;
10       st2: if (DIN1==1'b1) PST<=st3; else PST<=st2;
11       st3: if (DIN1==1'b1) PST<=st4; else PST<=st3;
12       st4: if (DIN1==1'b0) PST<=st0; else PST<=st4;
13       default:             PST<=st0;
14    endcase end end
```

```
16 always @(PST or DIN2) begin: COM  //输出控制信号的过程
17    case (PST)
18       st0: if (DIN2==1'b1) Q<=5'H10; else Q<=5'H0A;
19       st1: if (DIN2==1'b0) Q<=5'H17; else Q<=5'H14;
20       st2: if (DIN2==1'b1) Q<=5'H15; else Q<=5'H13;
21       st3: if (DIN2==1'b0) Q<=5'H1B; else Q<=5'H09;
22       st4: if (DIN2==1'b1) Q<=5'H1D; else Q<=5'H0D;
23       default:             Q<=5'b0000;
24    endcase
25    end
26 endmodule
```

10.3 Mealy型状态机设计



0-10 例 10-5 之双过程 Mealy 机仿真波形

10.3 Mealy型有限状态机设计

- 单进程有限状态机 (解决毛刺)

例10-6

```
1  module MEALY1(input CLK,DIN1,DIN2,RST, output reg [4:0] Q);
2    reg[4:0] PST;
3    parameter st0=0, st1=1, st2=2, st3=3, st4=4;
```

```
5  always @(posedge CLK or posedge RST) begin: REG
6  if (RST) PST<=st0; else begin
7    case(PST)
8      st0: begin if (DIN2==1'b1) Q<=5'H10; else Q<=5'H0A;
9           if (DIN1==1'b1) PST<=st1; else PST<=st0; end
10     st1: begin if (DIN2==1'b0) Q<=5'H17; else Q<=5'H14;
11          if (DIN1==1'b1) PST<=st2; else PST<=st1; end
12     st2: begin if (DIN2==1'b1) Q<=5'H15; else Q<=5'H13;
13          if (DIN1==1'b1) PST<=st3; else PST<=st2; end
14     st3: begin if (DIN2==1'b0) Q<=5'H1B; else Q<=5'H09;
15          if (DIN1==1'b1) PST<=st4; else PST<=st3; end
16     st4: begin if (DIN2==1'b1) Q<=5'H1D; else Q<=5'H0D;
17          if (DIN1==1'b0) PST<=st0; else PST<=st4; end
18     default: begin       PST<=st0; Q<=5'b0000; end
19   endcase end
20 endmodule
```

10.3 Mealy型有限状态机设计

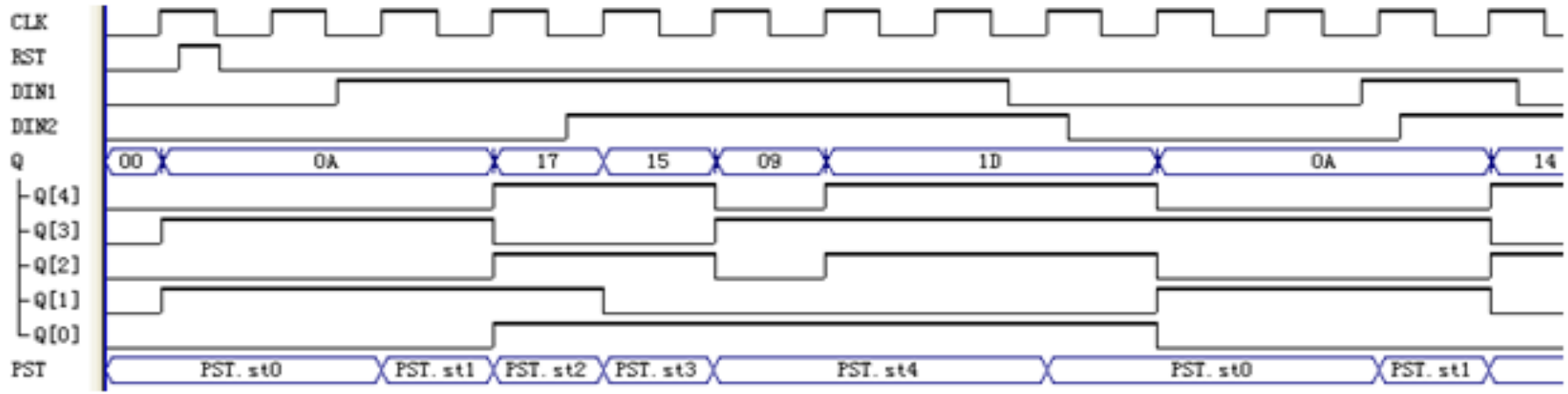


图 10-11　例 10-6 之单过程 Mealy 机仿真波形

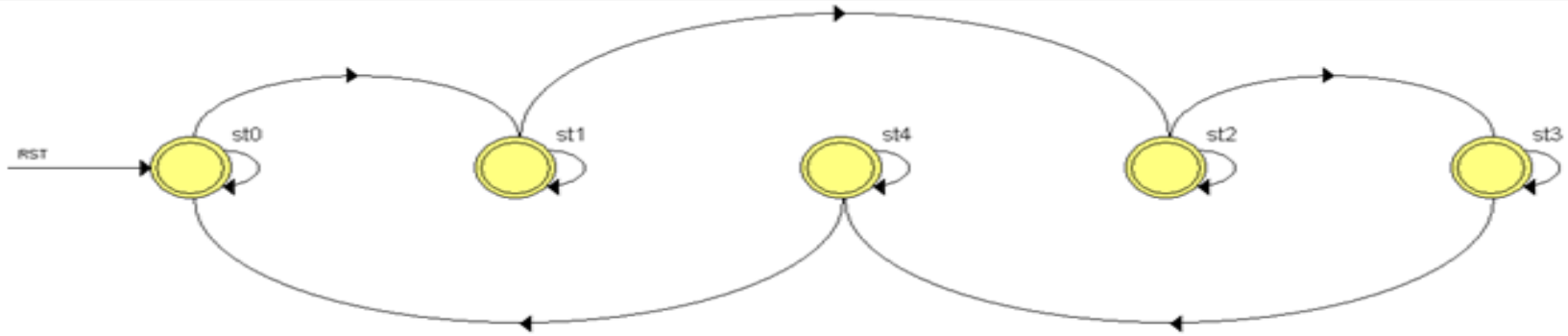


图 10-12　例 10-6，10-5 的状态图

10.3 Mealy型有限状态机设计

例10-7

```
1  module SCHK(input CLK,DIN,RST, output OUT);
2    parameter s0=0, s1=1, s2=2, s3=3, 84=4,
3              s5=5, s6=6, s7=7, s8=8; //设定9个状态参数
4    reg[8:0] ST;                    //代设定现态变量和次态变量
```

```
6    always @(posedge CLK )
7      SOUT=0;
8      if (RST) sT<=s0 ; begin  //11010011串行输入，高位在前
9      case(ST)
10        s0: if(DIN==1'b1) ST<=s1; else ST<=s0;
11        s1: if(DIN==1'b1) ST<=s2; else ST<=s0;
12        s2: if(DIN==1'b0) ST<=s3; else ST<=s0;
13        s3: if(DIN==1'b1) ST<=s4; else ST<=s0;
14        s4: if(DIN==1'b0) ST<=s5; else ST<=s0;
15        s5: if(DIN==1'b0) ST<=s6; else ST<=s0;
16        s6: if(DIN==1'b1) ST<=s7; else ST<=s0;
17        s7: if(DIN==1'b1) ST<=s8; else ST<=s0;
18        s8: SOUT=1;
19            if(DIN==1'b0) ST<=s3; else ST<=s0; end
20        default: NST<=s0;
21     endcase end end
22  endmodule
```

10.3 Mealy型有限状态机设计

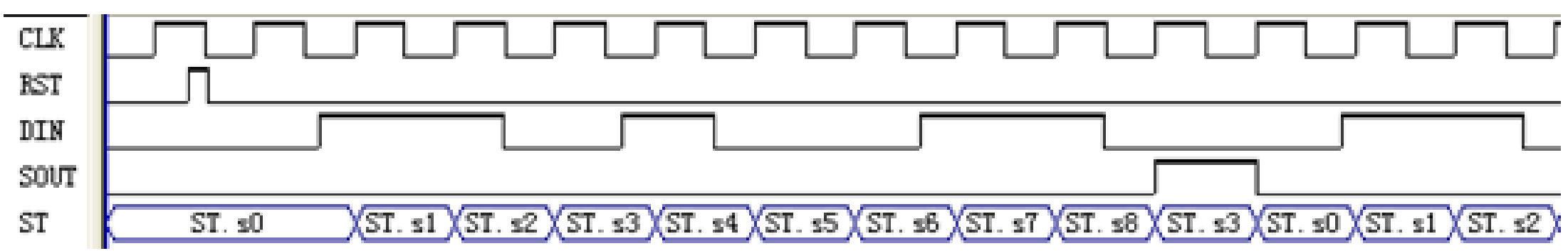


图 10-13　例 10-7 之单过程 Mealy 机仿真波形

有限状态机的描述列表

描述方式		进程描述功能
三过程描述方式		过程1：描述状态转换； 过程2：描述次态逻辑； 过程3：描述输出逻辑。
双过程描述方式	形式1	过程1：描述次态逻辑、输出逻辑； 过程2：描述状态转换。
	形式2	过程1：描述次态逻辑、状态转换； 过程2：描述输出逻辑。
	形式3	过程1：描述状态转换、输出逻辑； 过程2：描述次态逻辑。
单过程描述方式		过程1：描述次态逻辑、状态转换和输出逻辑。

10.4 状态机图形编辑设计

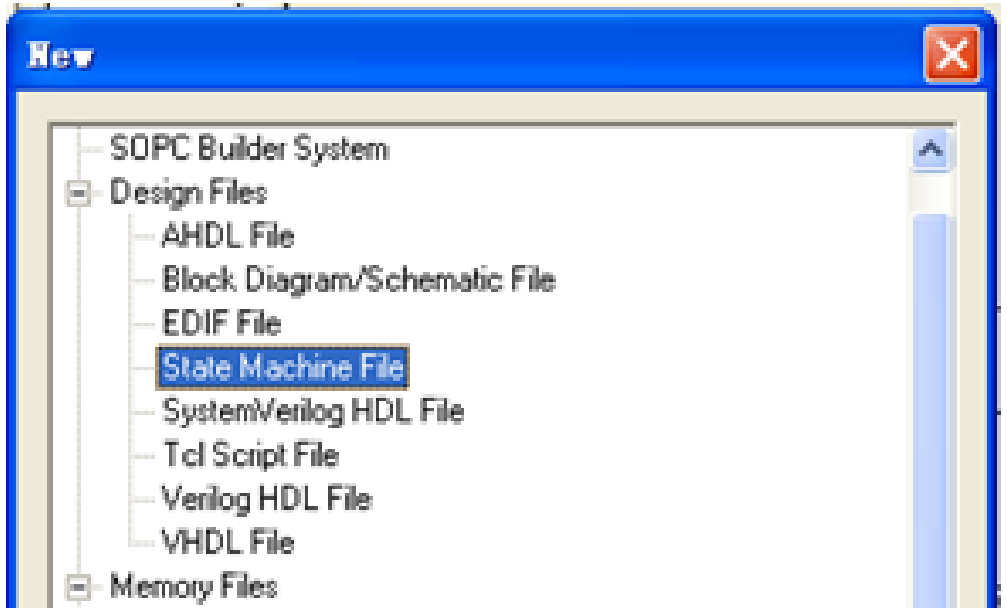


图 10-14 打开状态机图形编辑窗

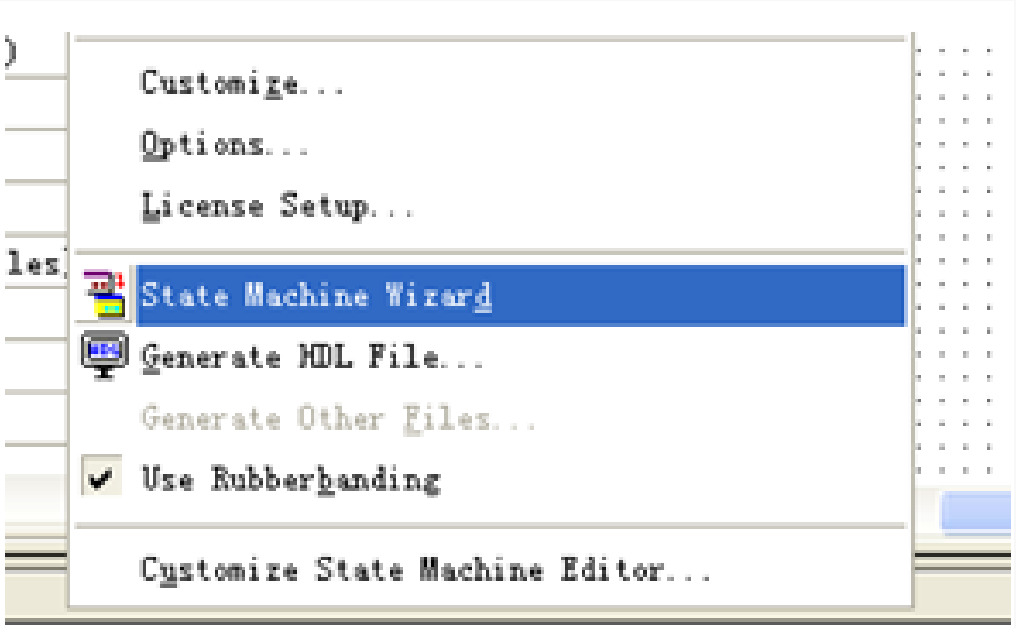


图 10-15 打开状态机编辑器

10.4 状态机图形编辑设计

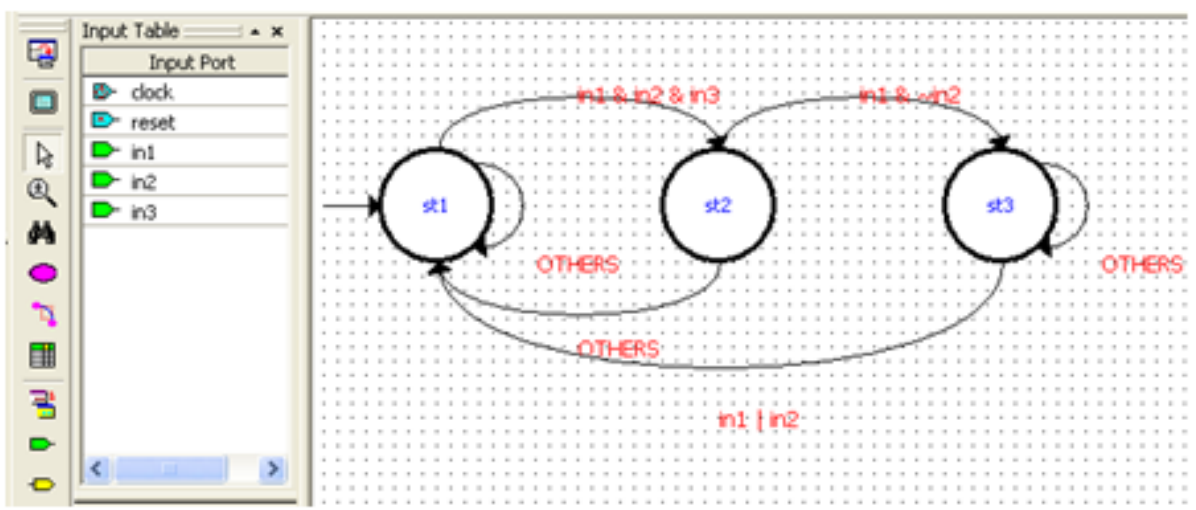


图 10-18 状态机图形编辑器上的状态图

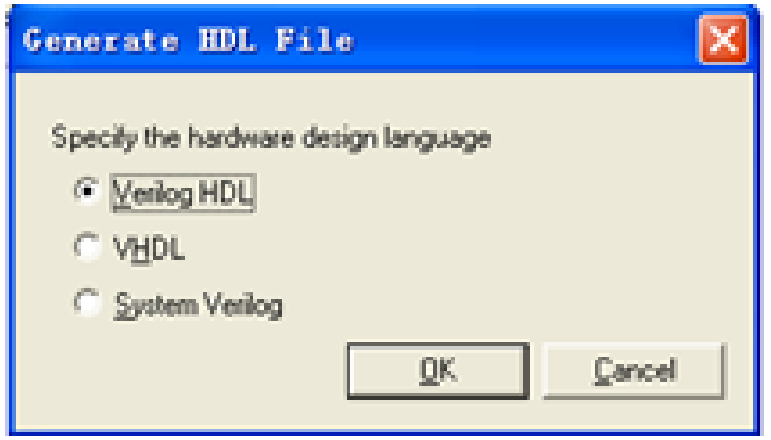


图 10-19 状态机转变成语言

10.5 不同编码类型状态机

10.5.1 状态位直接输出型编码

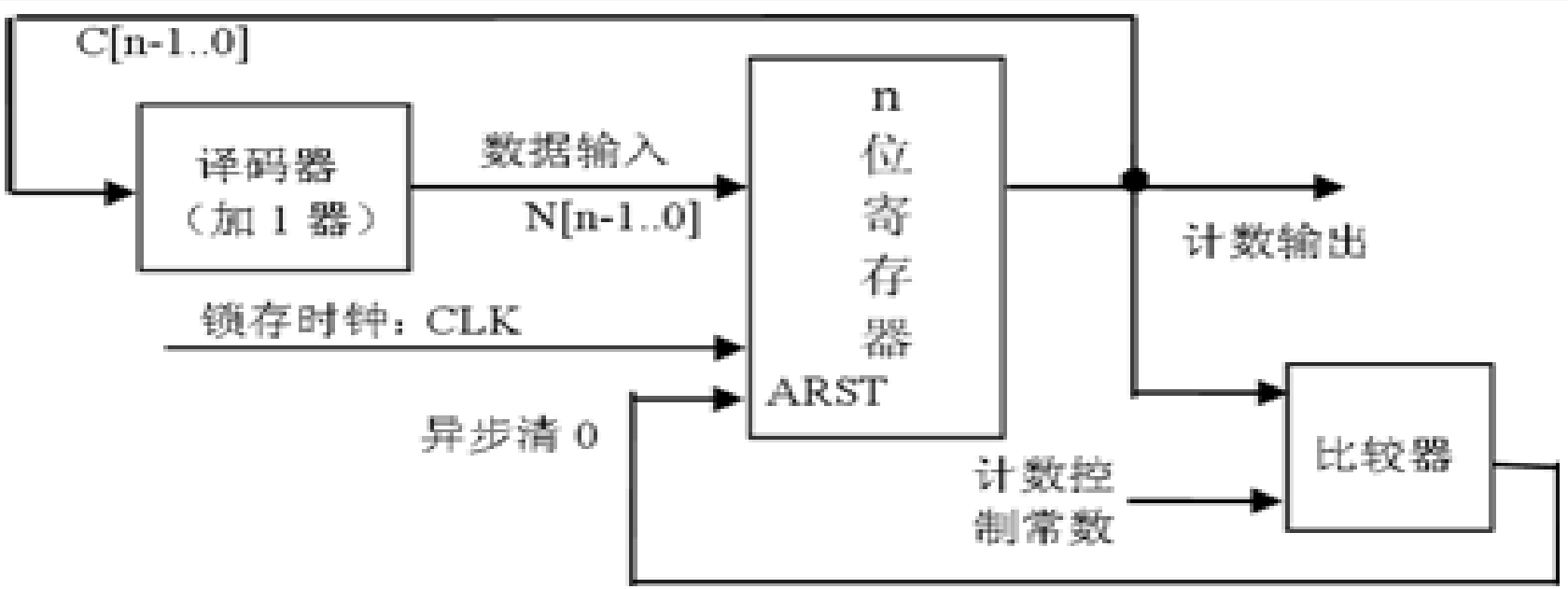
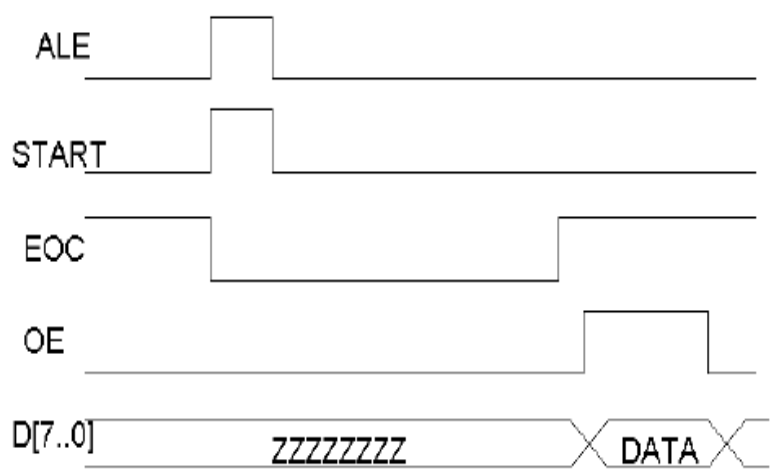


图 10-20 加法计数器一般模型

10.5 不同编码类型状态机



10.5.1 状态位直接输出型编码

表7-1 设计控制0809采样的状态机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

START=current_state(4); ALE= current_state(3);
OE=current_state(2); LOCK=current_state(1);

10.5 不同编码类型状态机

例10-8

```
1  module adc0809 (D,CLK,EOC,RST,ALE,START,OE,ADDA,Q,LOCK_T);
2    input[7:0] D;              //来自0809转换好的8位数据
3    input CLK,RST;             //状态机工作时钟，和系统复位控制
4    input EOC;                 //转换状态指示，低电平表示正在转换
5    output ALE;                //8个模拟信号通道地址锁存信号
6    output START,OE;           //转换启动信号，和数据输出三态控制信号
7    output ADDA,LOCK_T;        //信号通道控制信号和锁存测试信号
8    output[7:0]Q;  reg ALE,START,OE;
9    parameter s0=5'B00000,s1=5'B11000,s2=5'B00001,s3=5'B00100,s4=5'B00110; //定义各状态子类型
10   reg[4:0] cs, SOUT, next_state;          //为了便于仿真显示，现态名简为cs
11   reg[7:0] REGL; reg LOCK;              //转换后数据输出锁存时钟信号
```

```
Parameter s0=5'B00000,s1=5'B11000,s2=5'B00001,s3=5'B00100,s4=5'B00110;
```

10.5 不同编码类型状态机

```
13  always@(cs or EOC ) begin//组合过程，规定各状态转换方式
14    case (cs)
15      s0 : begin  next_state <= s1; SOUT<=s0; end          //0809初始化
16      s1 : begin  next_state <= s2; SOUT<=s1; end          //启动采样信号START
17      s2 : begin  SOUT<=s2;
18           if(EOC==1'b1) next_state <= s3;          //EOC=0表明转换结束
19           else next_state <= s2; end               //转换未结束，继续等待
20      s3 : begin  next_state <= s4; SOUT<=s3; end      //开启OE，打开AD数据
21      s4 : begin  next_state <= s0; SOUT<=s4; end      //开启数据锁存信号
22      default:begin next_state <= s0; SOUT<=s0; end
23    endcase
24  end
```

```
25  always @(posedge CLK or posedge RST) begin  //时序过程
26    if(RST) cs <= s0;
27       else cs <= next_state; end
```

```
28  always @(posedge SOUT[1])
29    if(SOUT[1]) REGL <= D;              //在LOCK上升沿将转换好的数据锁入
30  assign ADDA= 0; assign Q= REGL; assign LOCK_T=SOUT[1];  //选择模拟信号进入通道IN0
31  assign OE=SOUT[2];  assign ALE=SOUT[3]; assign START=SOUT[4];
32  endmodule
```

```
parameter s0=5'B00000,s1=5'B11000,s2=5'B00001,s3=5'B00100,s4=5'B00110;
```

10.5 不同编码类型状态机

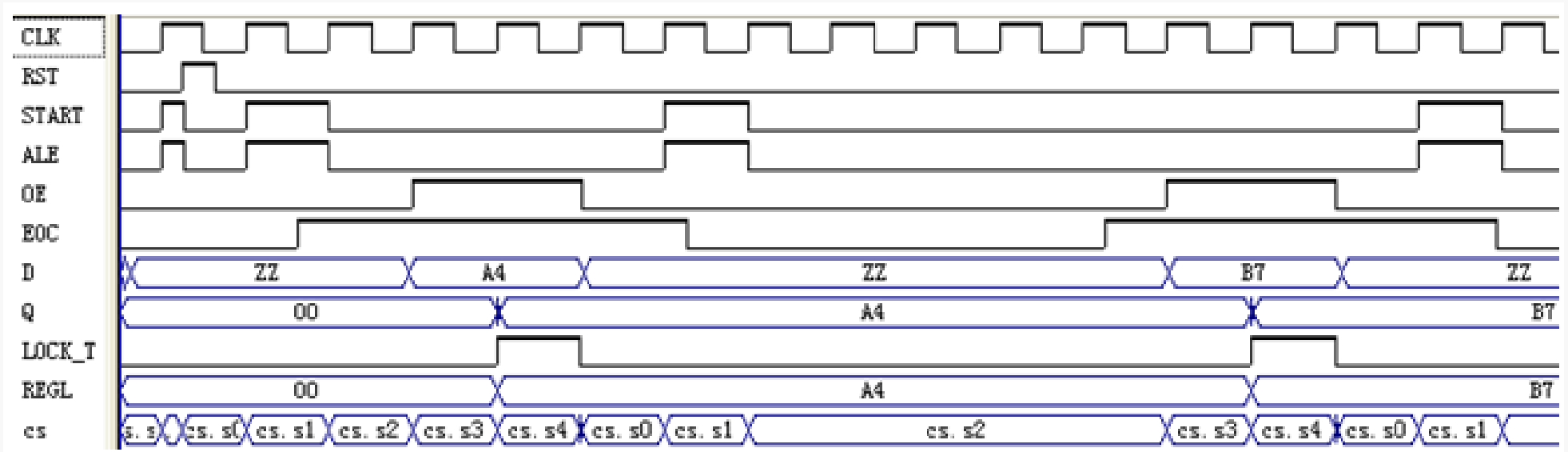


图 10-21 例 10-10 状态机工作时序图

优点：输出速度快，没有毛刺现象

缺点：程序可读性差，用于状态译码的组合逻辑资源多

10.5 不同编码类型状态机

10.5.2 用宏定义语句定义状态编码

```
1   `define s0 5'B00000
2   `define s1 5'B11000
3   `define s2 5'B00001
4   `define s3 5'B00100
5   `define s4 5'B00110
```

```
7   module adc0809 (D,CLK,EOC,RST,ALE,START,OE,ADDA,Q,LOCK_T);
8   input[7:0] D; input CLK,RST, EOC;
9   output START, OE, ALE, ADDA, LOCK_T;output[7:0]Q;
10  reg[4:0] cs, SOUT, next_state;
11  reg[7:0] REGL; reg LOCK;
```

```
13  always@(cs or EOC ) begin//组合过程，规定各状态转换方式
14    case (cs)
15      `s0 : begin  next_state <=`s1; SOUT<=`s0; end          //0809初始化
16      `s1 : begin  next_state <=`s2; SOUT<=`s1; end          //启动采样信号START
17      `s2 : begin      SOUT<=`s2;
18           if(EOC==1'b1) next_state <=`s3;        //EOC=0表明转换结束
19           else next_state <=`s2; end             //转换未结束，继续等待
20      `s3 : begin  next_state <=`s4; SOUT<=`s3; end       //开启OE，打开AD数据
21      `s4 : begin  next_state <=`s0; SOUT<=`s4; end       //开启数据锁存信号
22     default:begin next_state <=`s0; SOUT<=`s0; end
23    endcase
24  end
```

10.5 不同编码类型状态机

10.5.2 用宏定义语句定义状态编码

```
25  always @(posedge CLK or posedge RST) begin  //时序过程
26    if(RST) cs <=`s0;
27       else cs <= next_state; end
28  always @(posedge SOUT[1])
29    if(SOUT[1]) REGL <= D;              //在LOCK上升沿将转换好的数据锁入
30  assign ADDA= 0; assign Q= REGL; assign LOCK_T=SOUT[1];  //选择模拟信号进入通道IN0
31  assign OE=SOUT[2];  assign ALE=SOUT[3]; assign START=SOUT[4];
32  endmodule
```

10.5 不同编码类型状态机

10.5.3 宏定义命令语句

- `define 属于编译指示指令，不参与综合
- 在综合前做一些数据控制操作，类似于汇编的伪指令
- `define 语句格式：

  `define 宏名（标志符） 宏内容（字符串）

  `define s A+B+C+D

10.5 不同编码类型状态机

10.5.4 顺序编码

表10-2 编码方式

状 态	顺序编码	一位热码编码
STATE0	**000**	**100000**
STATE1	**001**	**010000**
STATE2	**010**	**001000**
STATE3	**011**	**000100**
STATE4	**100**	**000010**
STATE5	**101**	**000001**

顺序编码方式最简单，且使用的触发器数量最少，剩余的非法状态最少。

10.5 不同编码类型状态机

10.5.5 一位热码编码 （One-hot Encoding）

- 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的状态表示

状 态	顺序编码	一位热码编码
STATE0	**000**	**100000**
STATE1	**001**	**010000**
STATE2	**010**	**001000**
STATE3	**011**	**000100**
STATE4	**100**	**000010**
STATE5	**101**	**000001**

10.5 不同编码类型状态机

10.5.5 一位热码编码

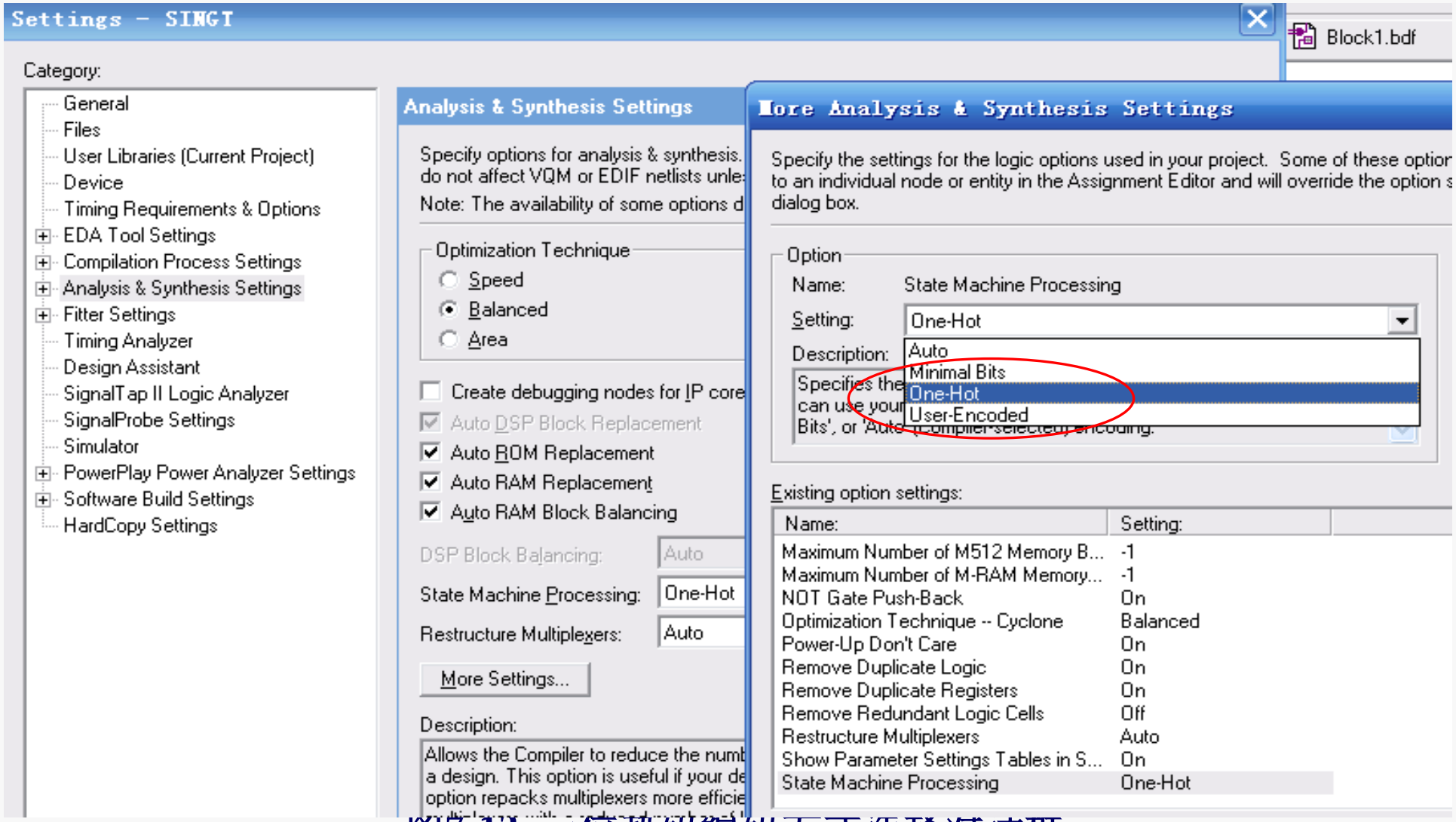


图7-13 一位热码编码方式选择对话框

10.7 安全状态机设计

10.7.1 状态导引法

表10-4 剩余状态

状 态	st0	St1	St2	St3	St4	st_ilg1	st_ilg2	st_ilg3
顺序编码	000	001	010	011	100	101	110	111

在语句中对每一个非法状态都作出明确的状态转换指示，如在原来的CASE语句中增加诸如以下语句：

```
st_ilg1:  next_state = st0;
st_ilg2:  next_state = st0;
   ...
default:  next_state=0;
```

10.7 安全状态机设计

1位热码编码方式剩余状态的处理：

【例7-10】

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

本章小结

- **Moore和Mealy有限状态机定义**
- 有限状态机的编程方法(1,2,3个过程）
- 其他状态机形式
  - 状态位直接输出编码
  - 顺序编码
  - 一位热码
- 消除毛刺的方法