数 字 逻 辑

丁 贤 庆

ahhfdxq@163.com

通知　实验安排

数字逻辑电路课程有16个学时的实验，初步安排：具体安排参见公共邮箱中的word文档。

本周开始进行实验环节，实验结束后16周周日（6月16号）晚23点前，各班学委要提交实验报告的电子版给我的邮箱ahhfdxq@163.com。不用收纸质报告了。

实验地点：综合实验楼306房间

第六章　作业布置

1、本周有实验。

2、下次交作业第11周。

3、本周作业：从第6章课后习题中选1题写到作业本上。

第 六 章

时序逻辑电路

6.6 简单的时序可编程逻辑器件(GAL)

6.6.1 GAL的结构

6.6.2 GAL的输出逻辑宏单元

6.6.3 GAL的控制字

1. 时序可编程逻辑器件的主要类型

1、GAL
2、CPLD
3、FPGA

（1）通用阵列逻辑（GAL）

在PLA和PAL基础上发展起来的增强型器件.电路设计者可根据需要编程，对宏单元的内部电路进行不同模式的组合，从而使输出功能具有一定的灵活性和通用性。

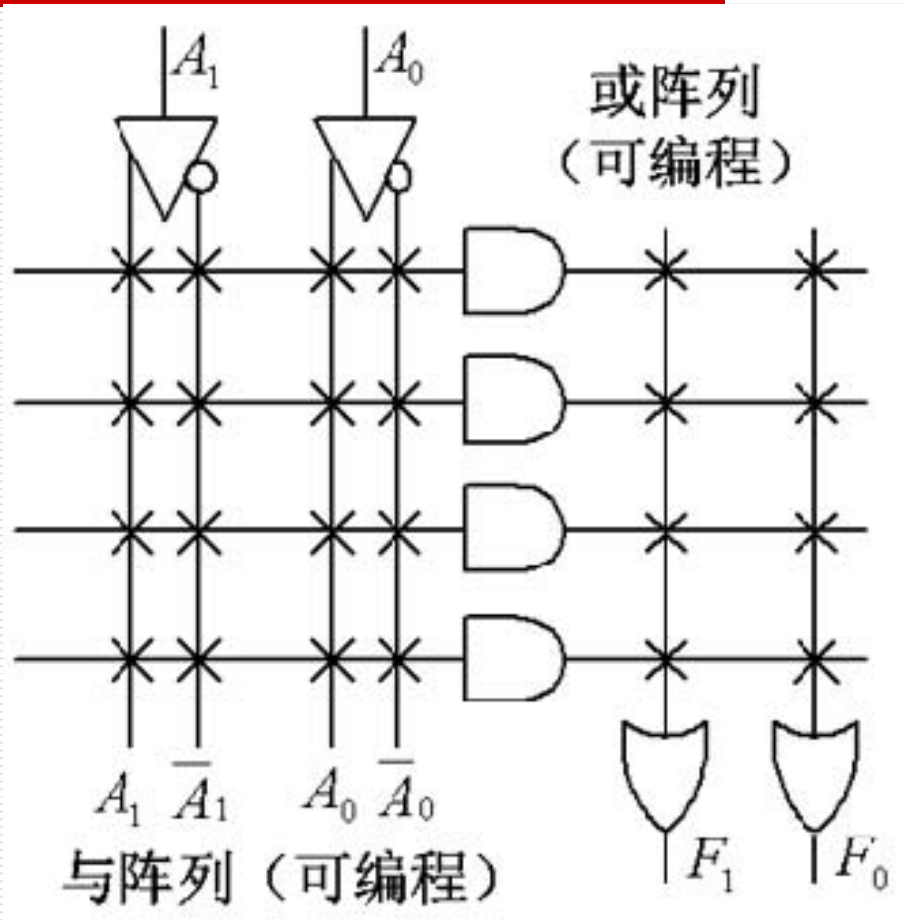
（2）复杂可编程逻辑器件（CPLD）

集成了多个逻辑单元块，每个逻辑块就相当于一个GAL器件。这些逻辑块可以通过共享可编程开关阵列组成的互连资源，实现它们之间的信息交换，也可以与周围的I/O模块相连，实现与芯片外部交换信息。

（3） 现场可编程门阵列（FPGA）

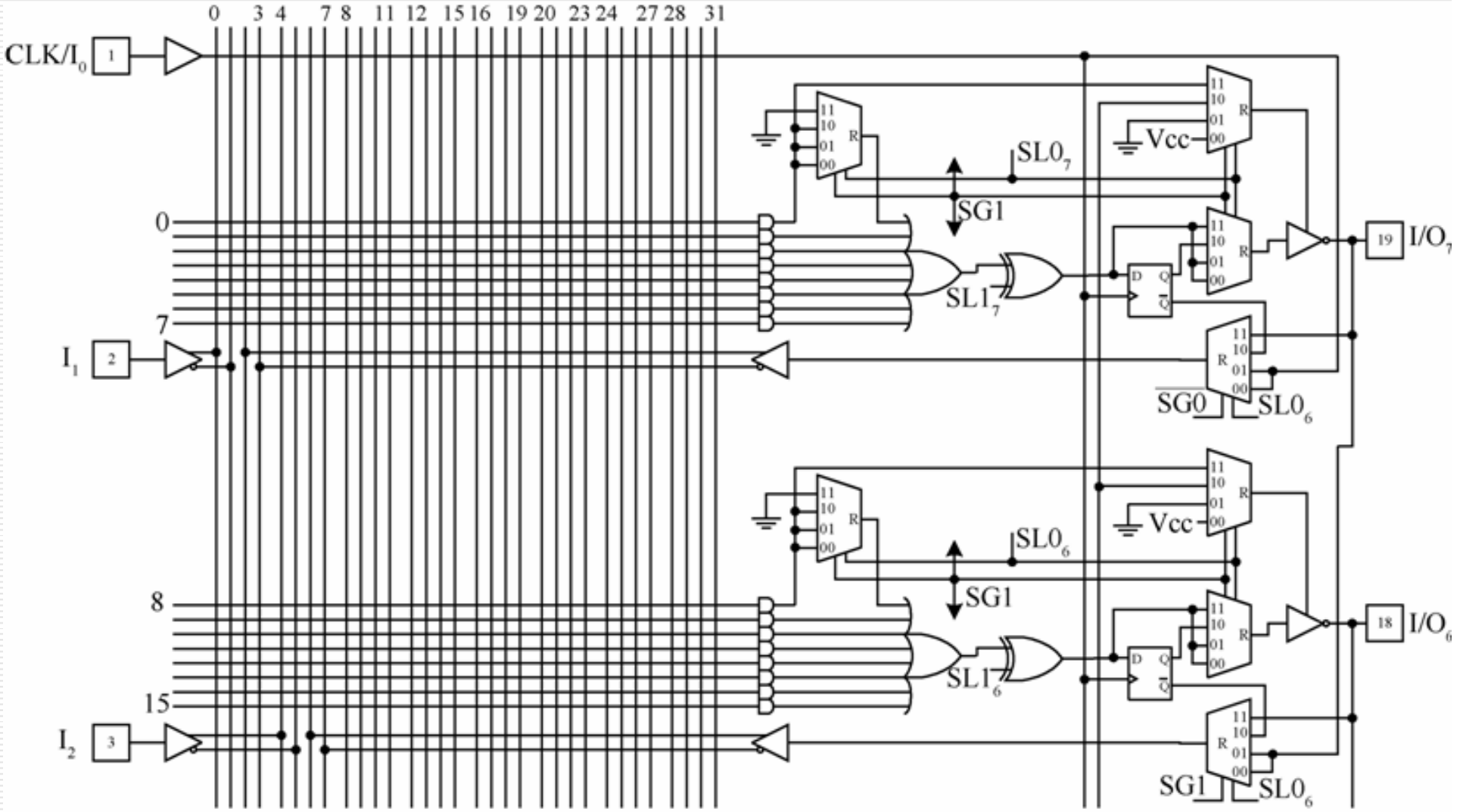
芯片内部主要由许多不同功能的可编程逻辑模块组成，靠纵横交错的分布式可编程互联线连接起来，可构成极其复杂的逻辑电路。它更适合于实现多级逻辑功能，并且具有更高的集成密度和应用灵活性在软件上，亦有相应的操作系统配套。这样，可使整个数字系统（包括软、硬件系统）都在单个芯片上运行，即所谓的SOC技术。

2. PAL的不足:



PLA逻辑阵列示意图

PAL16V8部分结构图



2. PAL的不足:

（1）由于采用的是双极型熔丝工艺，一旦编程后不能修改;

（2）输出结构类型太多，给设计和使用带来不便。

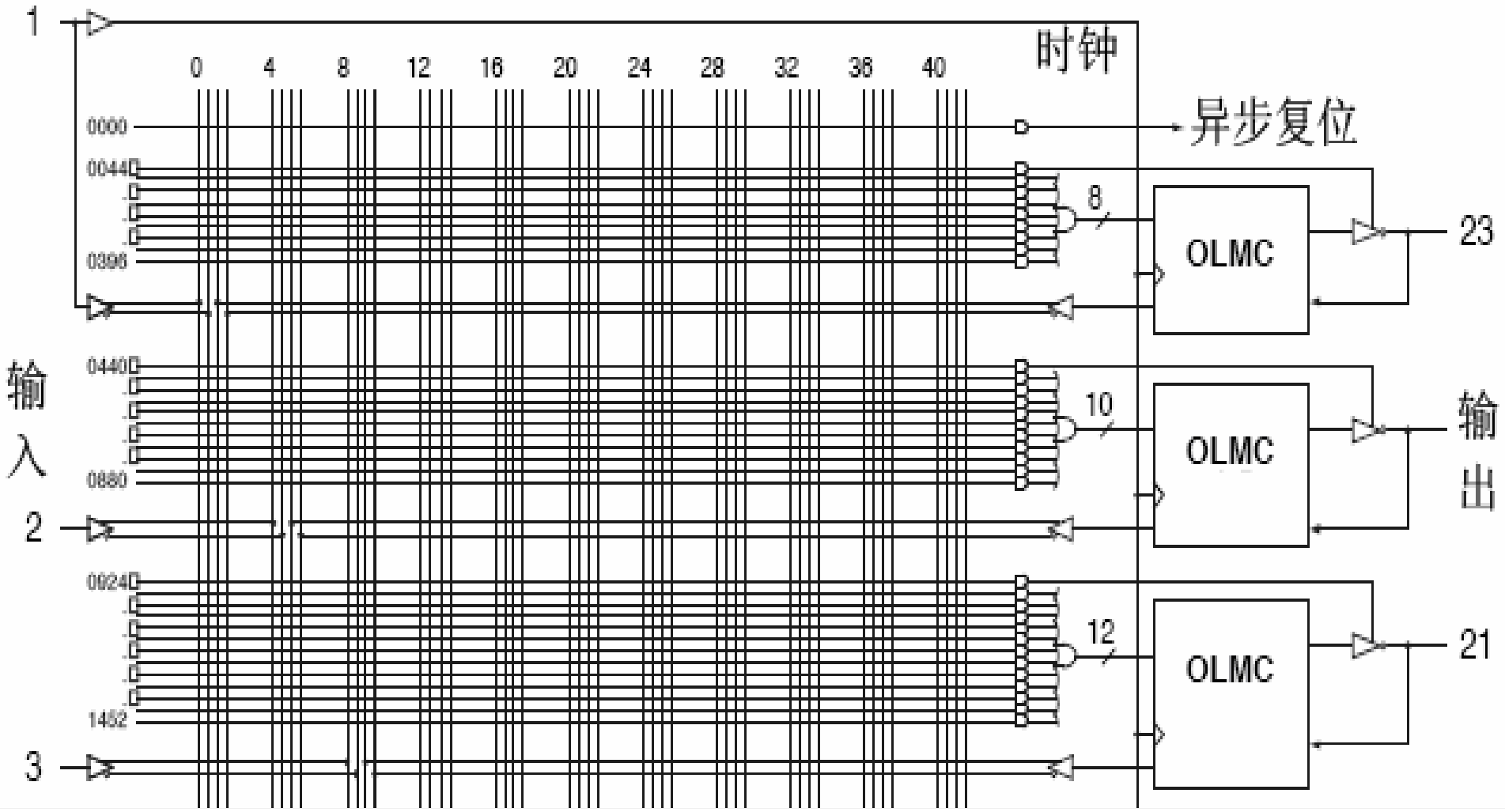
3. GAL的优点:

（1）采用电可擦除的E^2CMOS工艺可以多次编程;

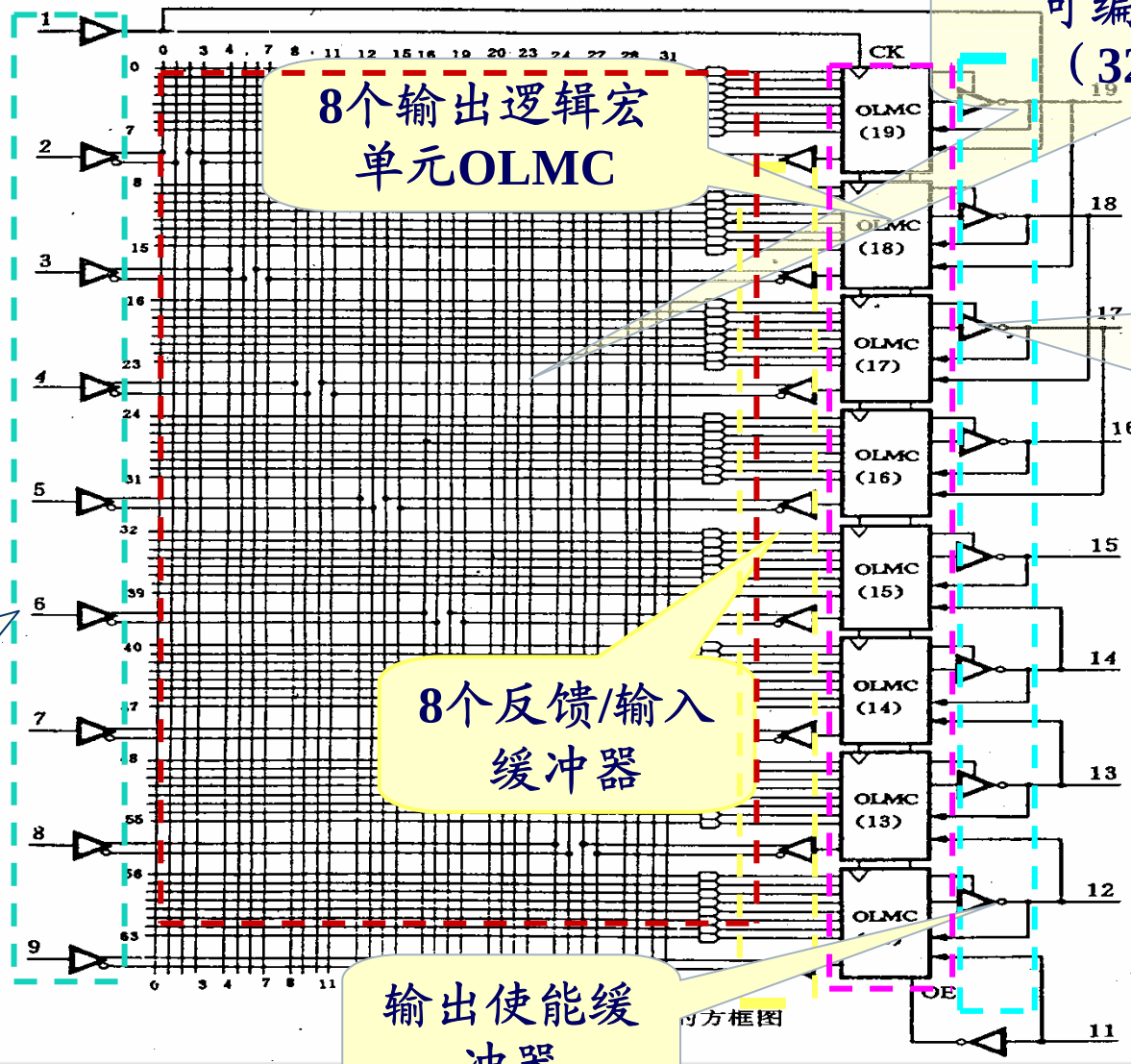
（2）输出端设置了可编程的输出逻辑宏单元（OLMC）通过编程可将OLMC设置成不同的工作状态，即一片GAL便可实现PAL 的5种输出工作模式。器件的通用性强;

（3）GAL工作速度快，功耗小

GAL22V10的结构（局部）

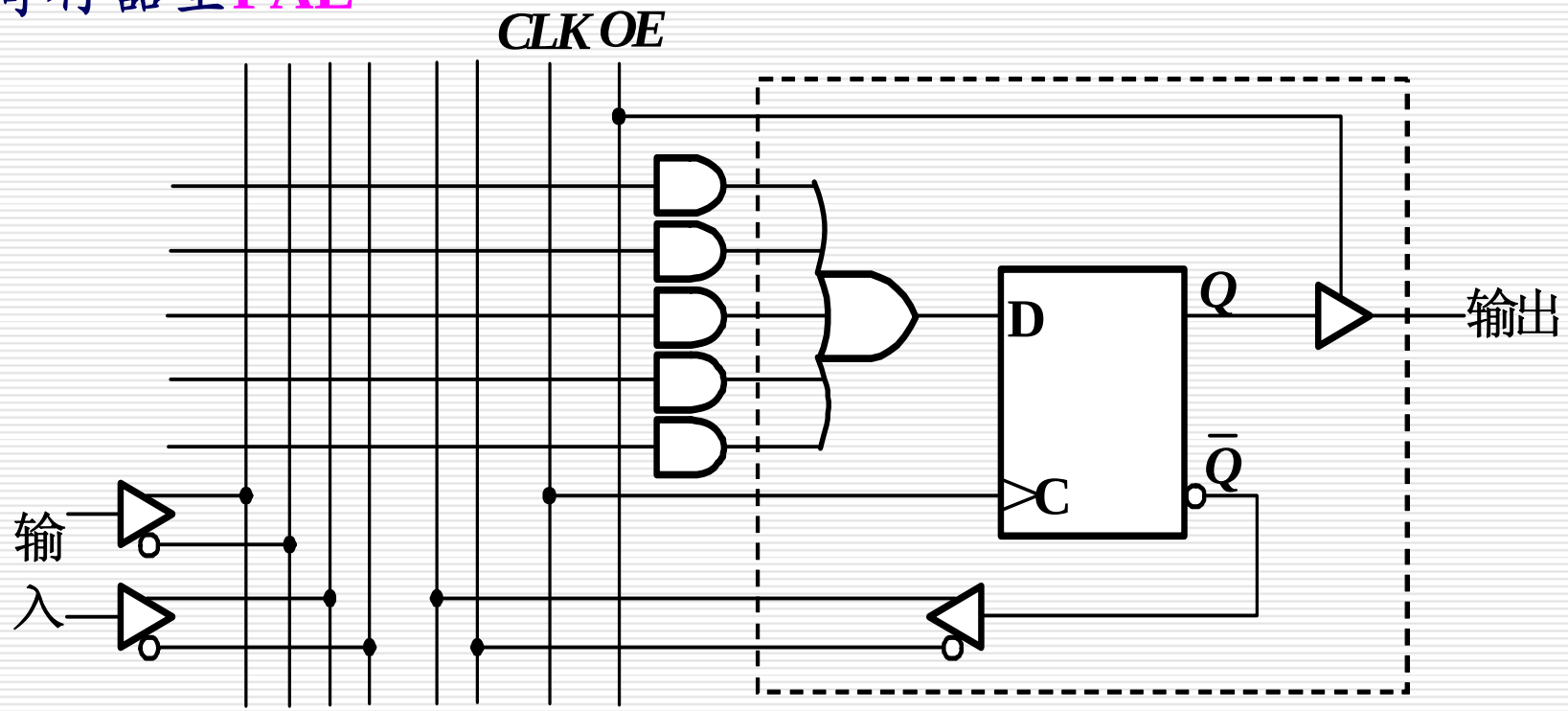


6.6.1 GAL的结构—GAL16V8的结构为例



6.6.2 GAL中的输出逻辑宏单元

1. 寄存器型PAL

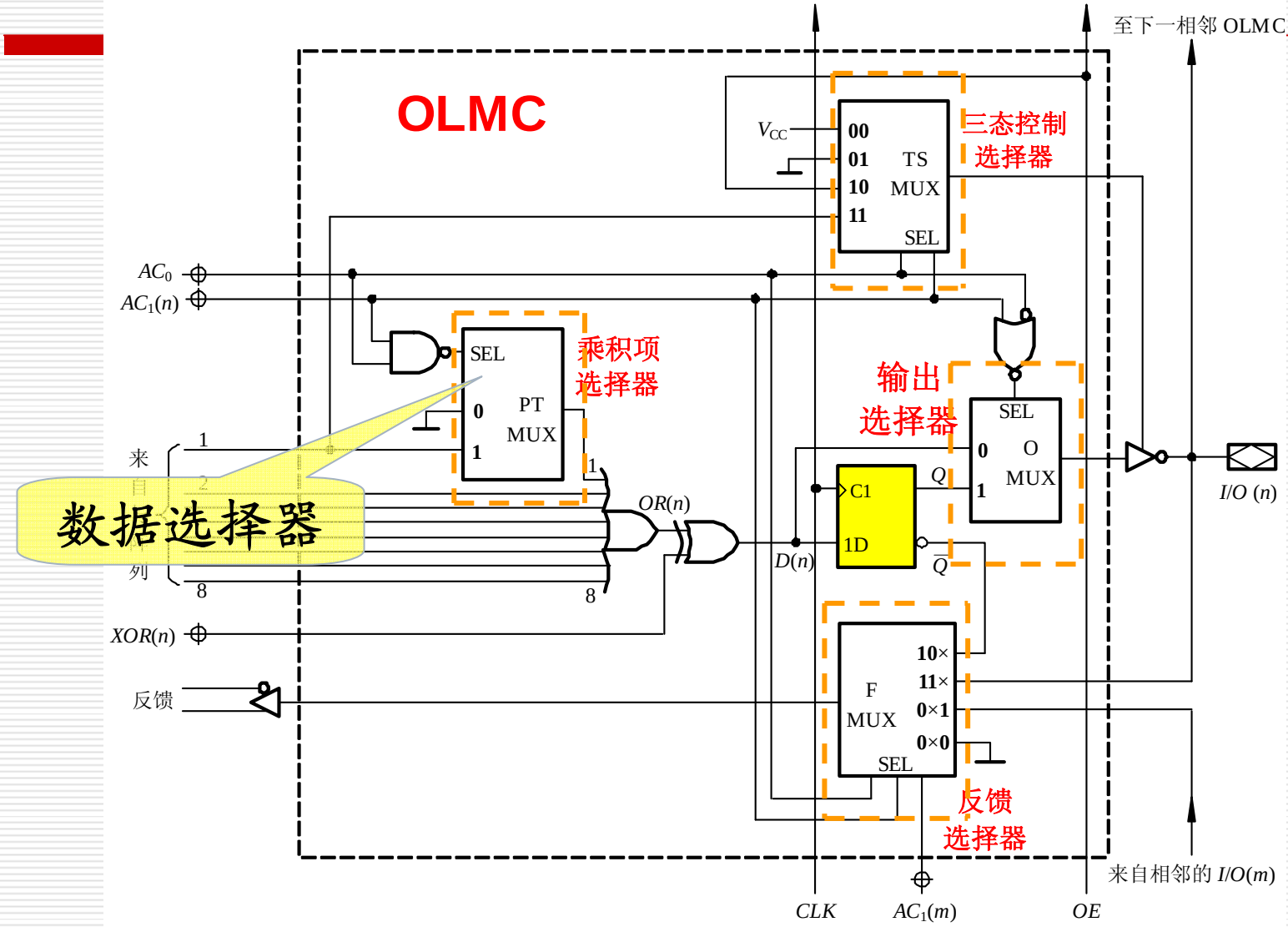


寄存器型PAL如图所示，在组合PLD基础上增加了D触发器，并反馈回到输入与阵列，满足时序电路设计要求。

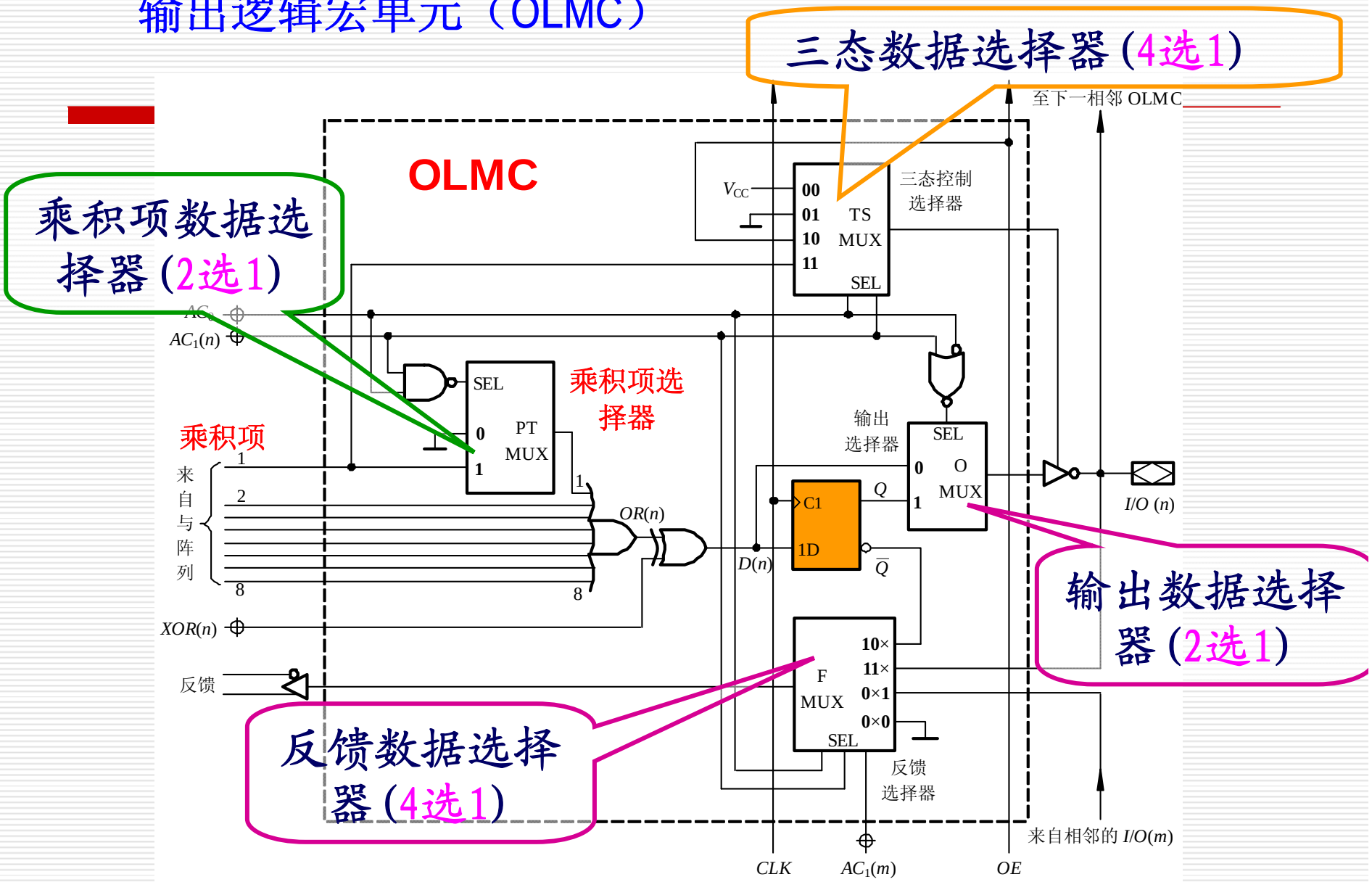
2. GAL中的输出宏单元

GAL的电路结构与PAL类似，由可编程的与阵列、固定的或阵列和输出电路组成，但GAL的输出端增设了可编程的的输出逻辑宏单元（OLMC）。通过编程可将OLMC设置为不同的工作状态，可实现PAL的所有输出结构，产生组合、时序逻辑电路输出。

输出逻辑宏单元（OLMC）

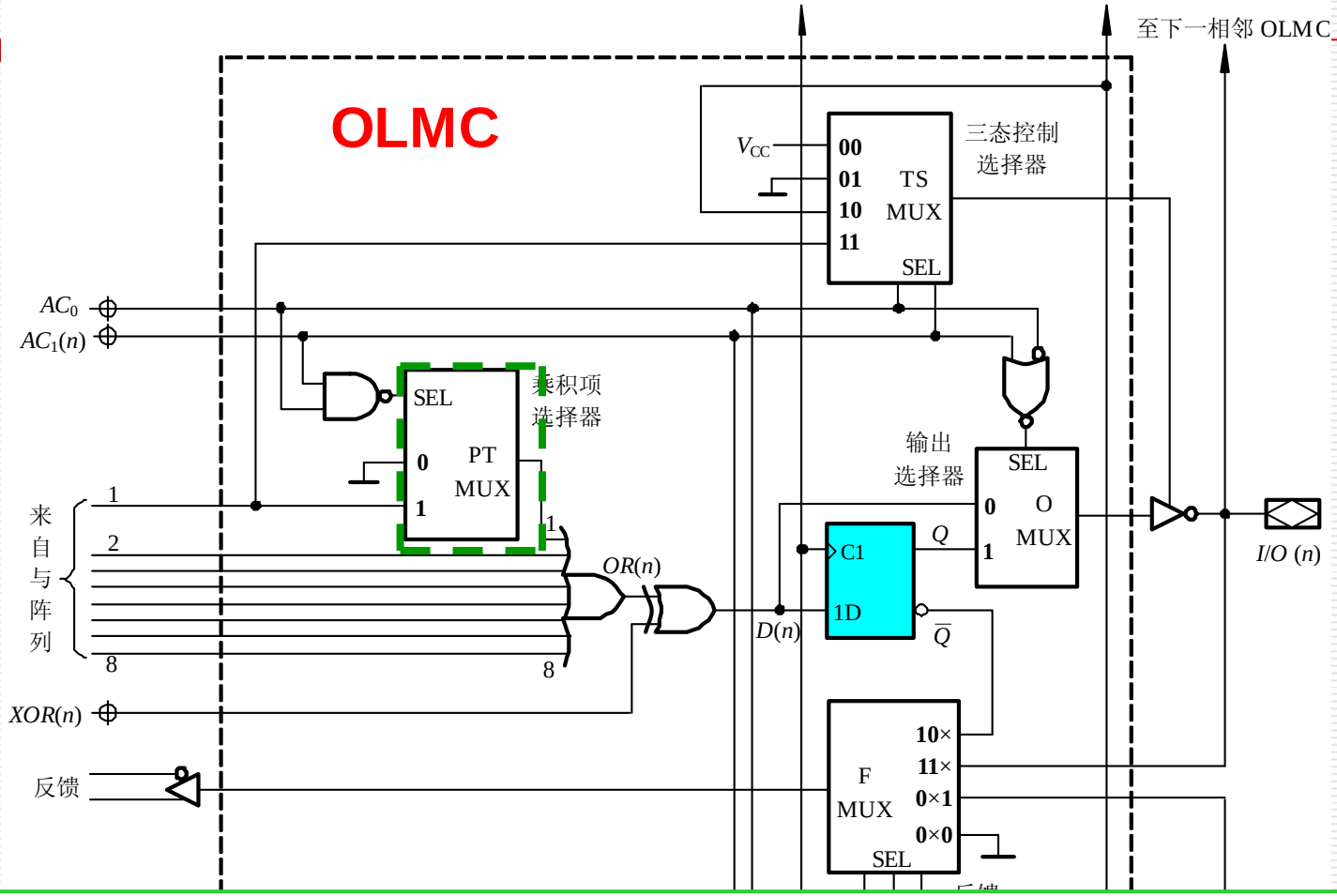


输出逻辑宏单元（OLMC）



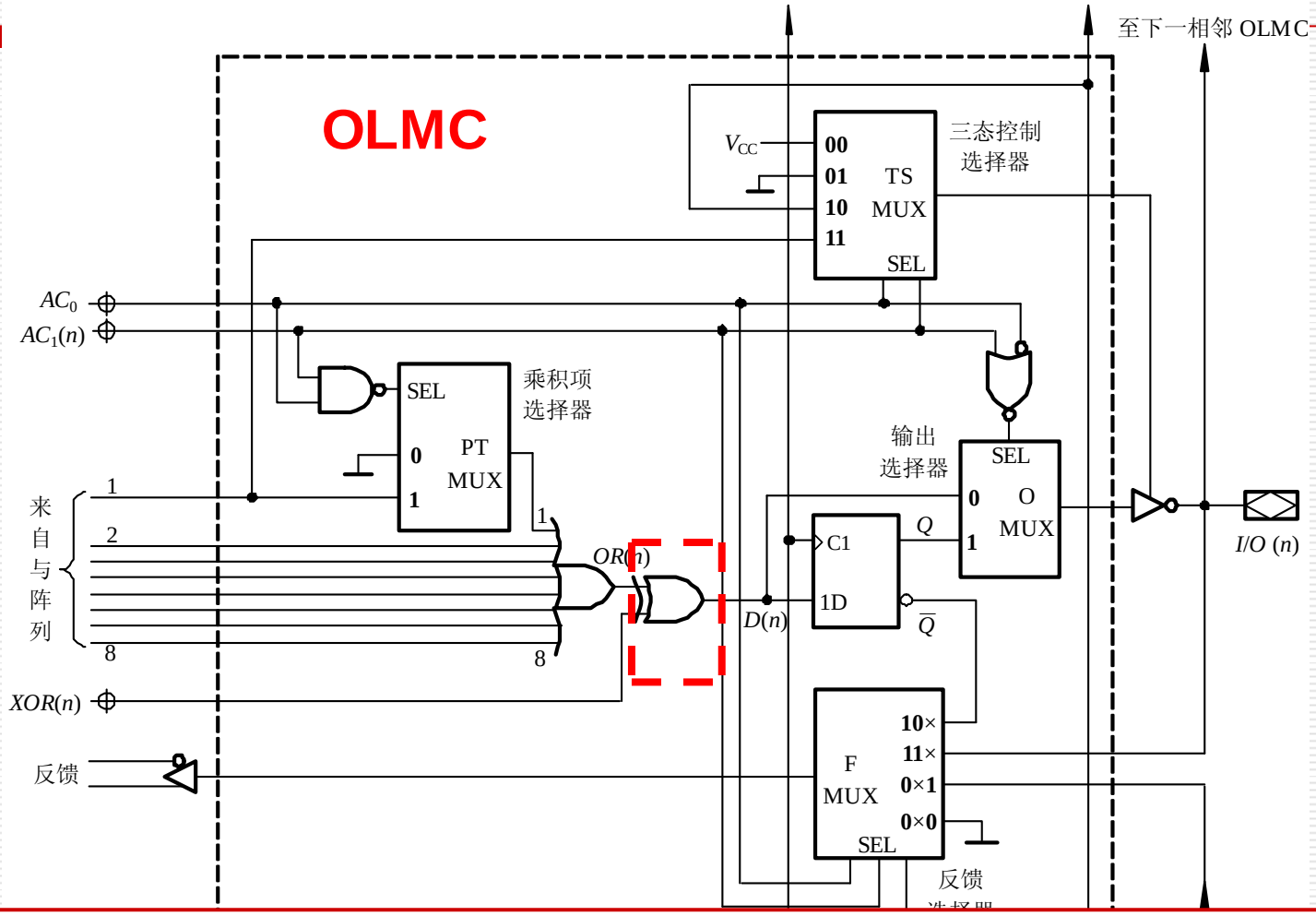
4个数据选择器：用不同的控制字实现不同的输出电路结构形式

(1) 输入电路—由乘积项数据选择器(2选1)PTMUX控制



乘积项数据选择器：根据AC0和AC1(n)决定与逻辑阵列的第一乘积项是否作为或门的一个输入端。只有在G_2的输出为1时，第一乘积项是或门的一个输入端。

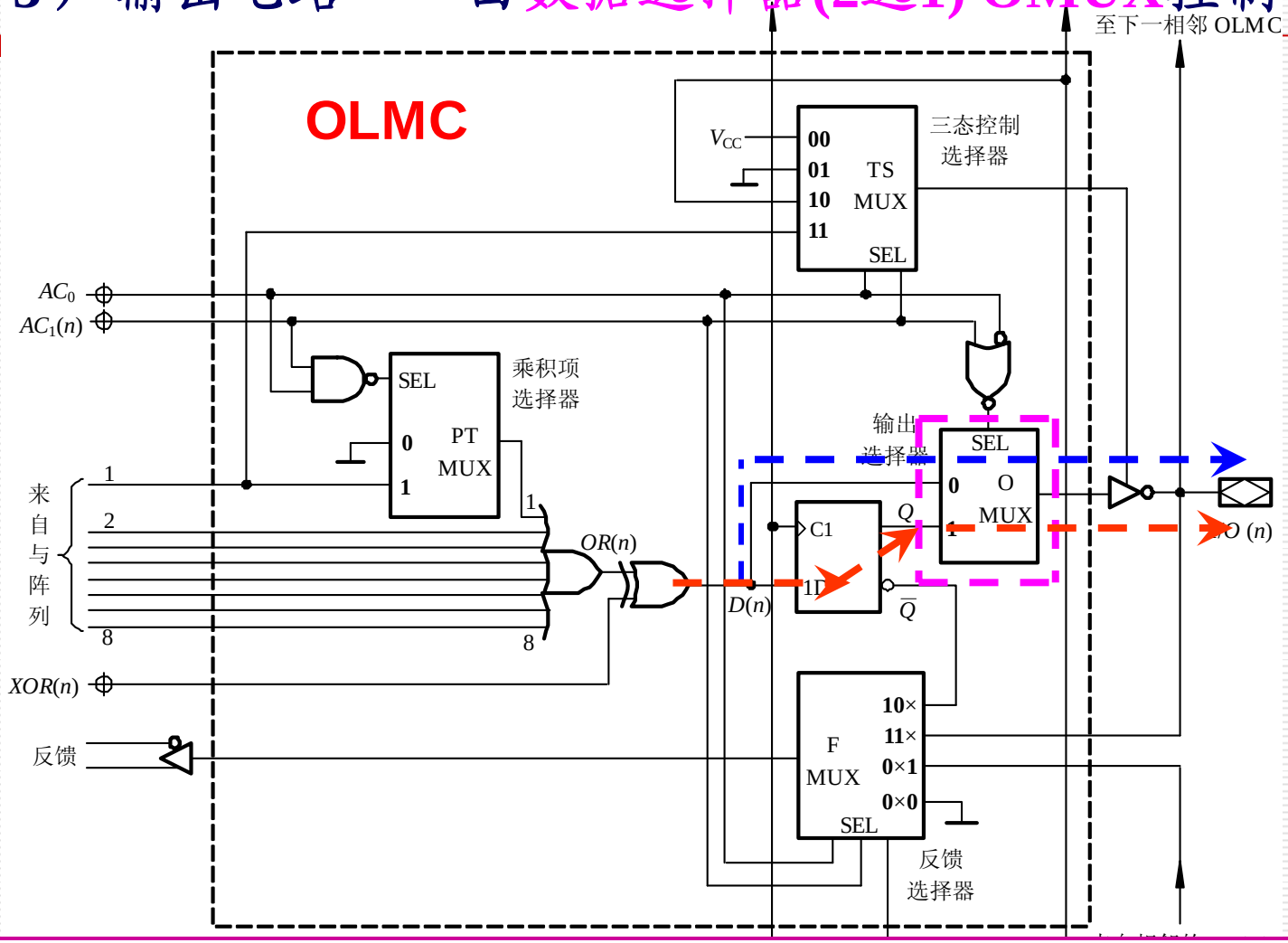
(2) 原变量/非变量输出电路—由异或门控制



异或门输出为或门输出OR(n) 与XOR(n)进行异或运算。

XOR(n)=0，则D(n)=OR(n)，若XOR(n)=1，则D(n)=$\overline{OR(n)}$ 。

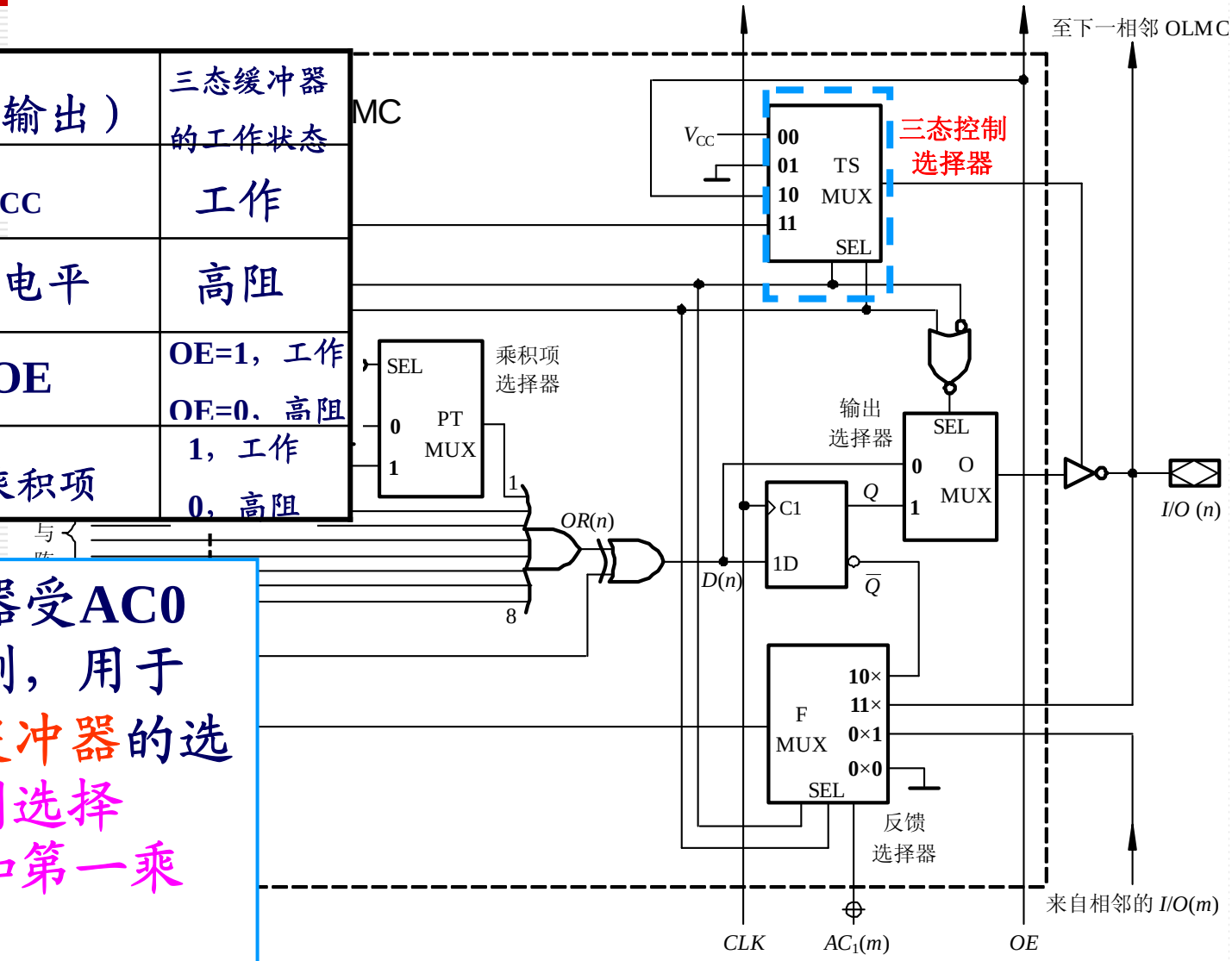
（3）输出电路——由数据选择器(2选1) OMUX控制



OMUX：根据AC0和AC1(n)决定OLMC是组合输出还是寄存器输出模式

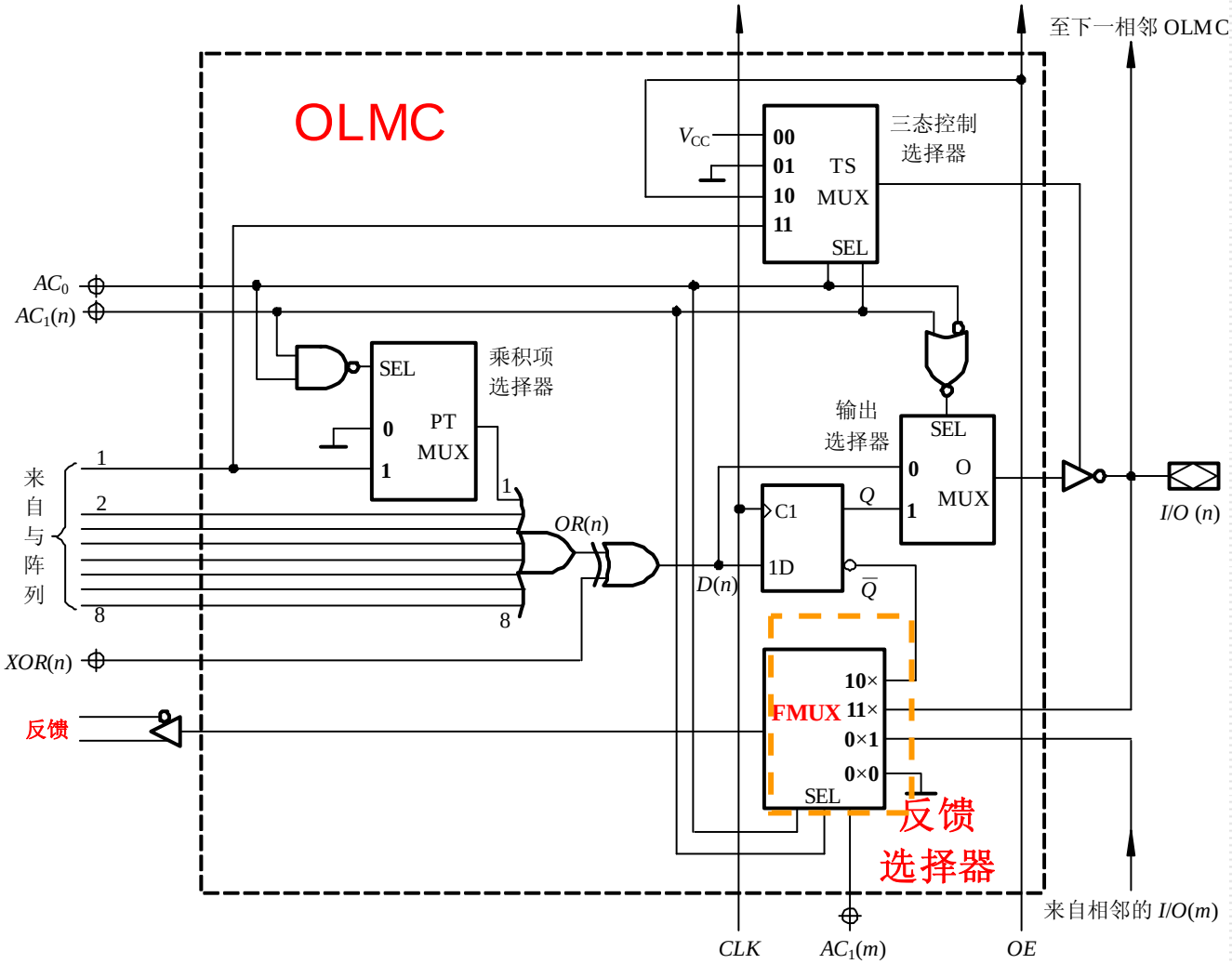
由三态数据选择器(4选1)控制输出选择器的选通端SEL

AC0 AC1(*n*)	TX（输出）	三态缓冲器的工作状态
0 0	V_{CC}	工作
0 1	地电平	高阻
1 0	OE	OE=1，工作 OE=0，高阻
1 1	第一乘积项	1，工作 0，高阻



三态数据选择器受AC0和AC1(n)的控制，用于选择输出三态缓冲器的选通信号。可分别选择V_{CC}、地、OE和第一乘积项。

反馈数据选择器(4选1)——FMUX



FMUX:

根据AC0和AC1(n)的不同编码，使反向传输的电信号也对应不同。

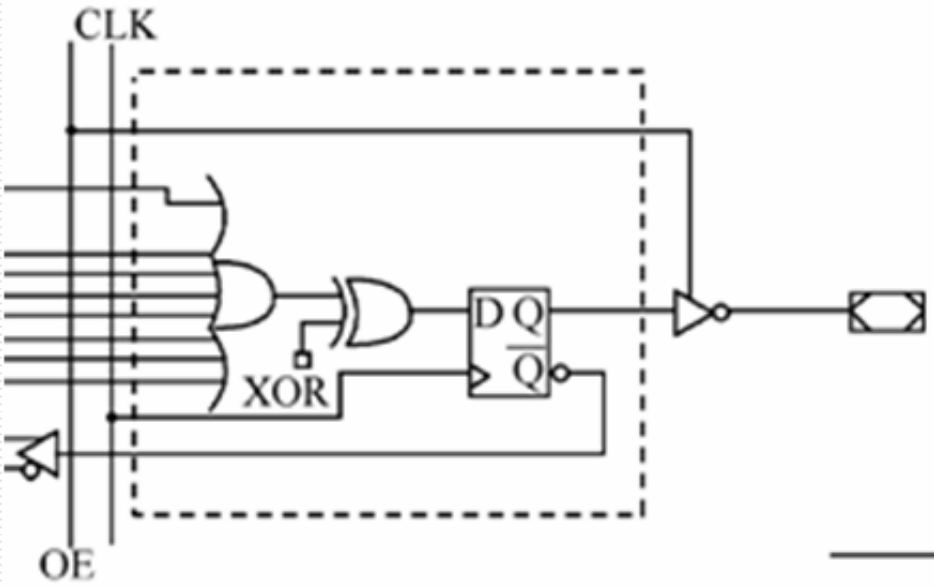
输出逻辑宏单元（OLMC）

（1）寄存器模式

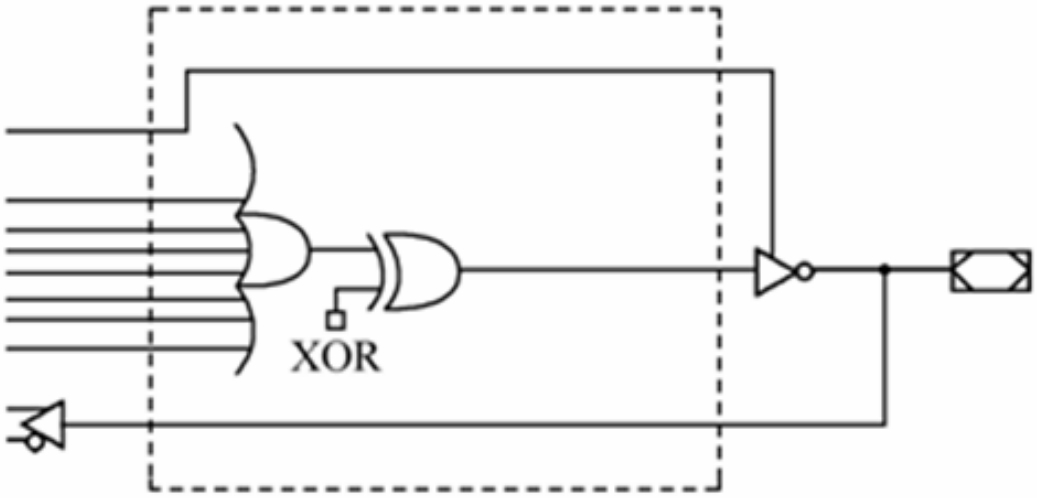
（2）复合模式

（3）简单模式

（1）寄存器模式

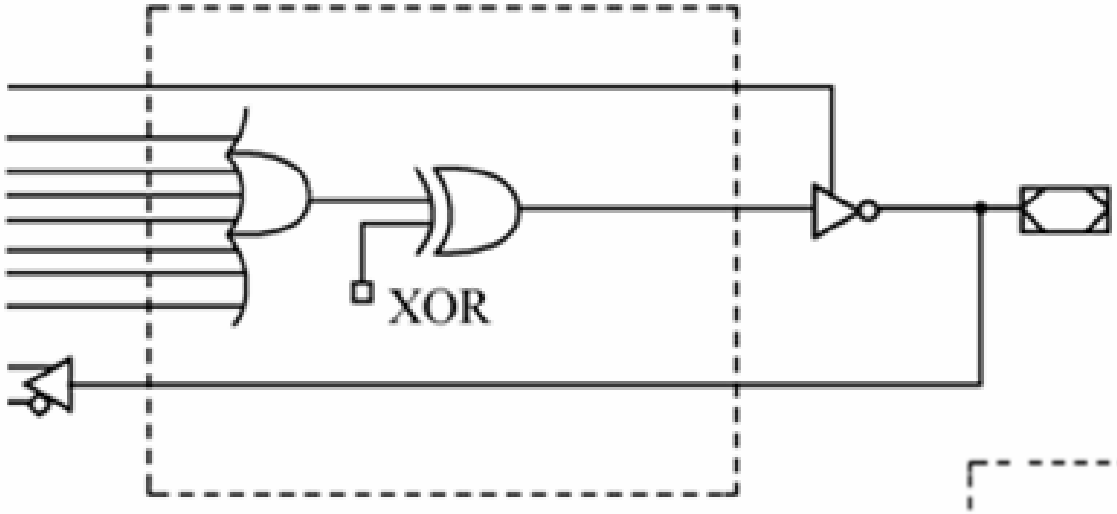


寄存器输出结构

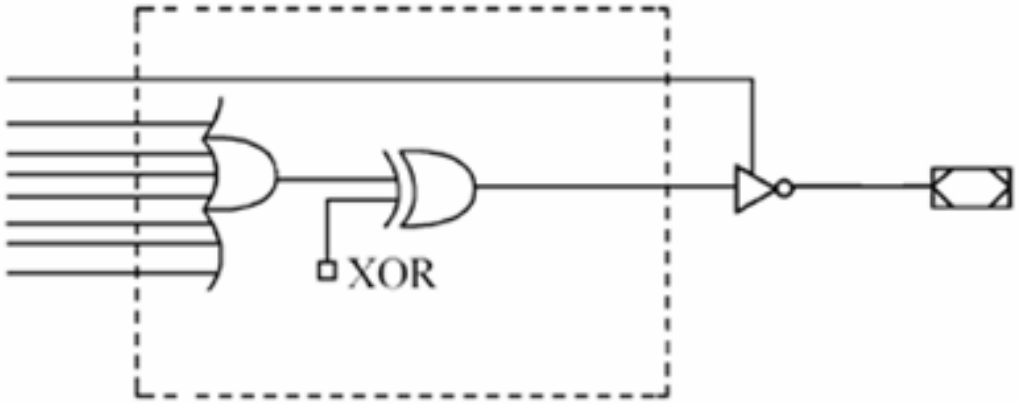


寄存器模式组合双向输出结构

（2）复合模式

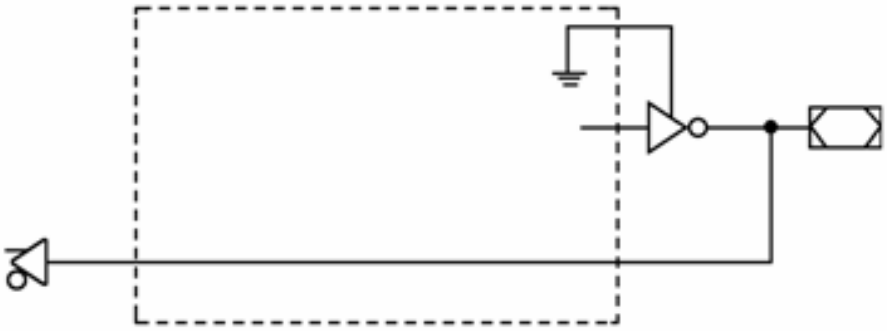


组合输出双向结构

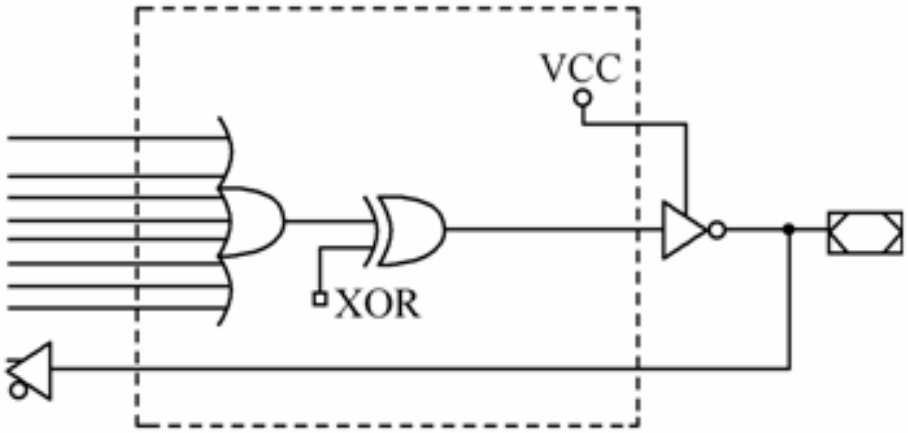


复合型组合输出结构

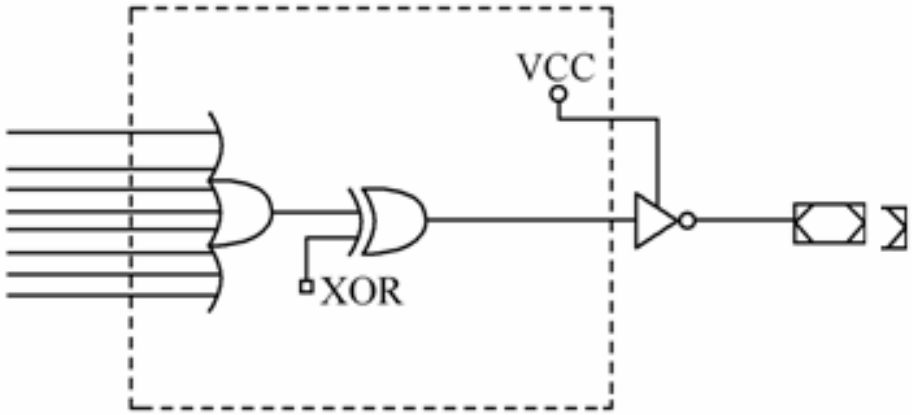
（3）简单模式



反馈输入结构



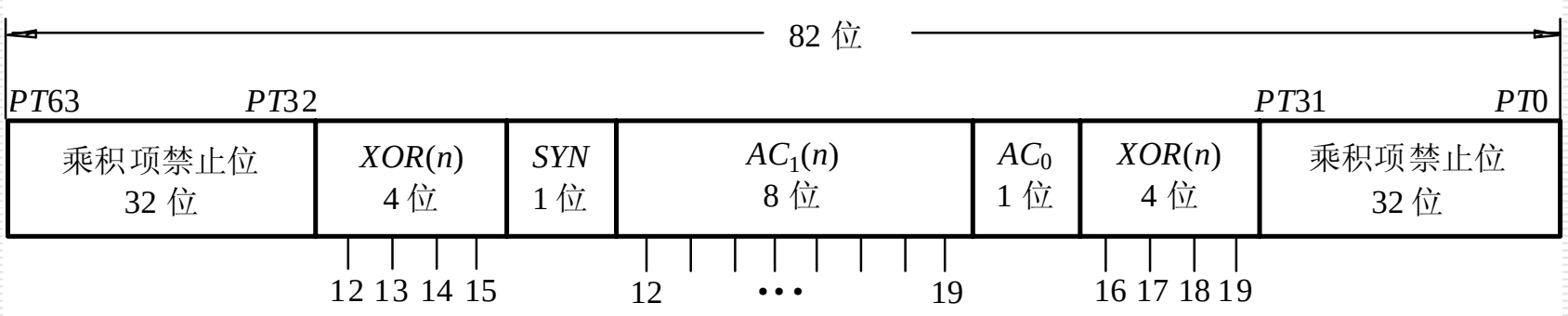
输出反馈结构



简单模式输出结构

6.6.3 GAL的结构控制字

GAL16V8的结构控制字共有82位，它们的定义如图。每个OLMC有2个编程单元AC1(n)和XOR(n)，一个全局编程单元AC0，同步控制单元SYN 。



6.7 用Verilog HDL描述时序逻辑电路

6.7.1 移位寄存器的Verilog建模

用行为级描述always描述一个 4 位双向移位寄存器，有异步清零、同步置数、左移、右移和保持。功能同74xx194。

```
module shift74x194 (S1, S0, D, Dsl, Dsr, Q, CP, CR);
    input S1, S0;                    //控制输入
    input Dsl, Dsr;                  //串行输入
    input CP, CR;                    //时钟及清零
    input [3:0] D;                   //并行输入
    output [3:0] Q;                  //寄存器输出
    reg [3:0] Q;
```

6.7.1 移位寄存器的Verilog建模

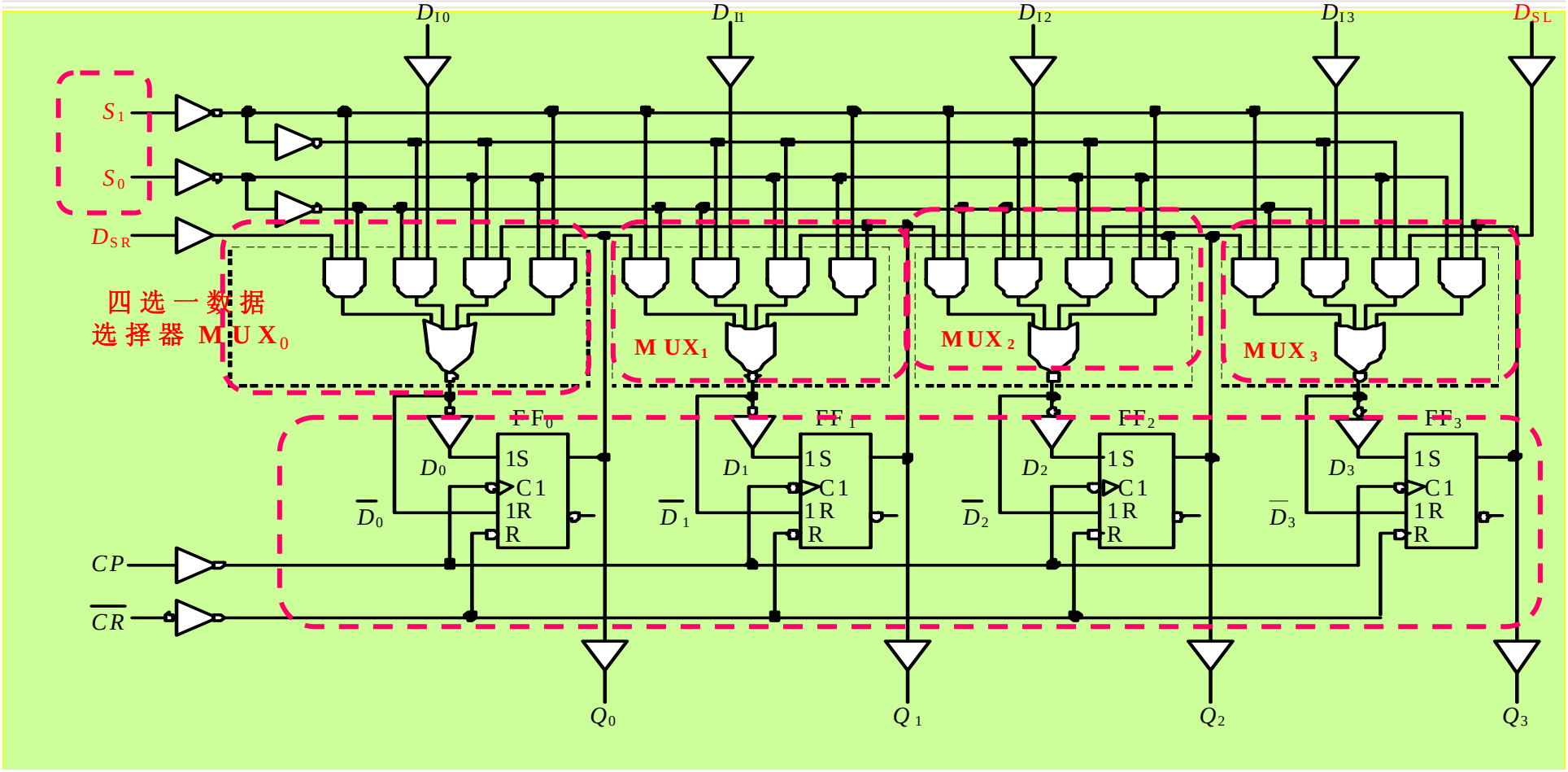
```
always @ (posedge CP or negedge CR)
    if (~CR) Q <= 4'b0000;
    else
      case ({S1,S0})
          2‘b00: Q <= Q;                  //保持
          2‘b01: Q <= {Q[2:0],Dsr};       //右移
          2‘b10: Q <= {Dsl,Q[3:1]};       //左移
          2‘b11: Q <= D;                  //并行输入
      endcase
endmodule
```

（b）典型集成电路

CMOS 4位双向移位寄存器74HC/HCT194

74HCT194 的功能表

S1	S0	功能
0	0	保持
0	1	低位往高位移动
1	0	高位往低位移动
1	1	并行置入



6.7.2 计数器的Verilog建模实例

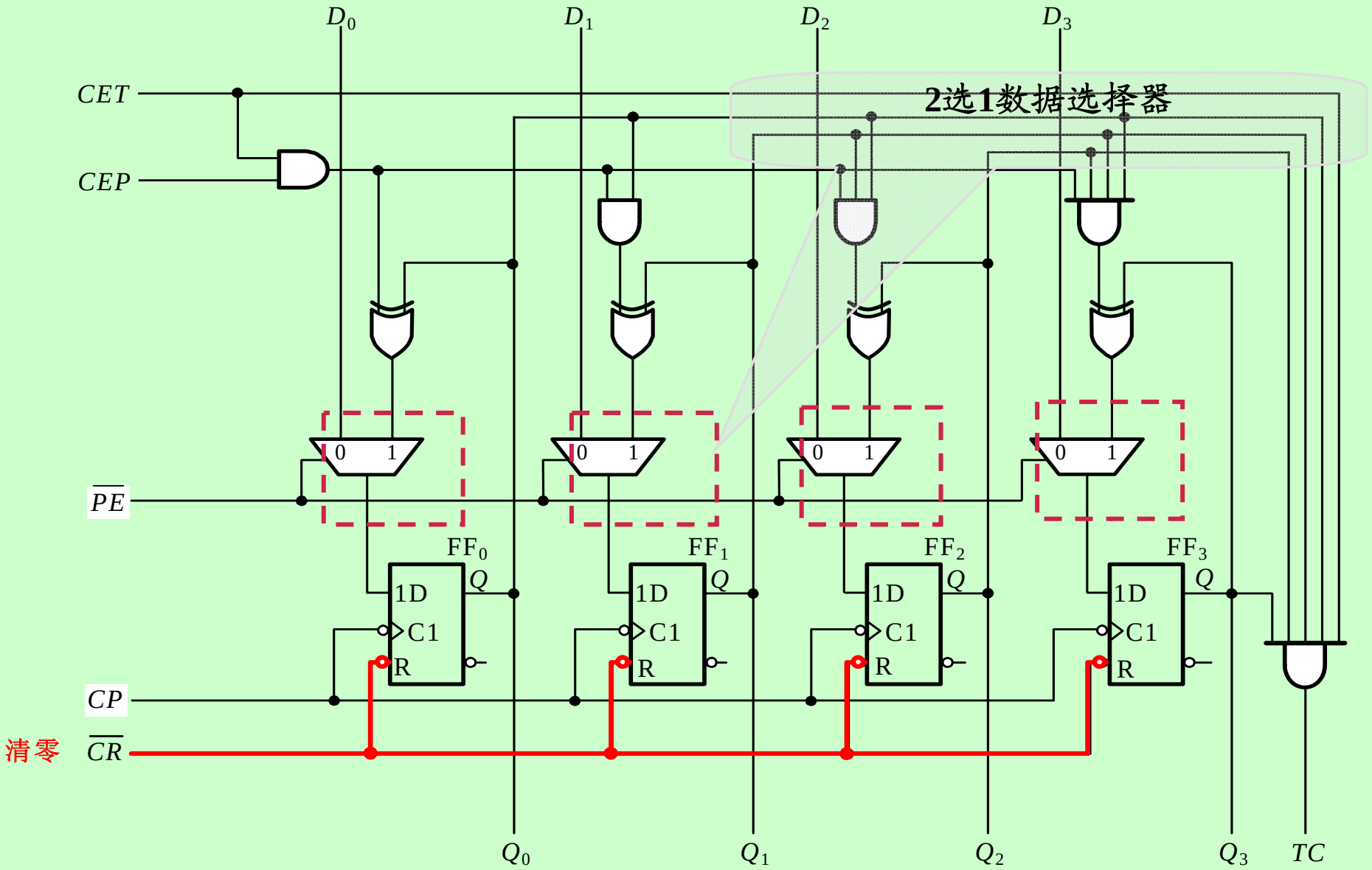
用Verilog描述具有使能端、异步置零、同步置数、计数、保持的16进制计数器

```
module counter74x161_beh (    //Verilog 2001, 2005 syntax
  input CEP, CET, PE, CP, CR,  //输入端口声明
  input [3:0] D,                  //并行数据输入
  output TC,                  //进位输出
  output reg [3:0] Q     //数据输出端口及变量的数据类型声明
  );
  wire CE;                  //中间变量声明
```

6.7.2 计数器的Verilog建模实例

```
assign CE=CEP&CET;   //CE=1时，计数器计数
assign TC=CET&PE&(Q == 4'b1111);  //产生进位输出信号
always @(posedge CP, negedge CR)  //Verilog 2001, 2005 syntax
    if (~CR) Q<=4'b0000;          //实现异步清零功能
    else if (~PE) Q<=D;             //PE=0, 同步装入输入数据
    else if (CE) Q<=Q+1'b1;       //加1计数
     else Q<=Q;                   //输出保持不变
endmodule
```

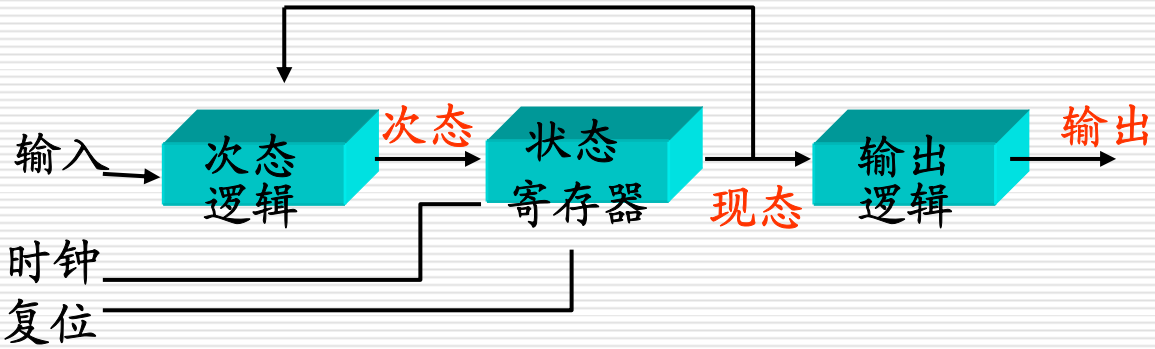
芯片74LVC161


D_0
D_1
D_2
D_3
CET
CEP
2选1数据选择器
0
1
$\overline{PE}$
FF_0
FF_1
FF_2
FF_3
1D
C1
R
Q
CP
清零
$\overline{CR}$
Q_0
Q_1
Q_2
Q_3
TC

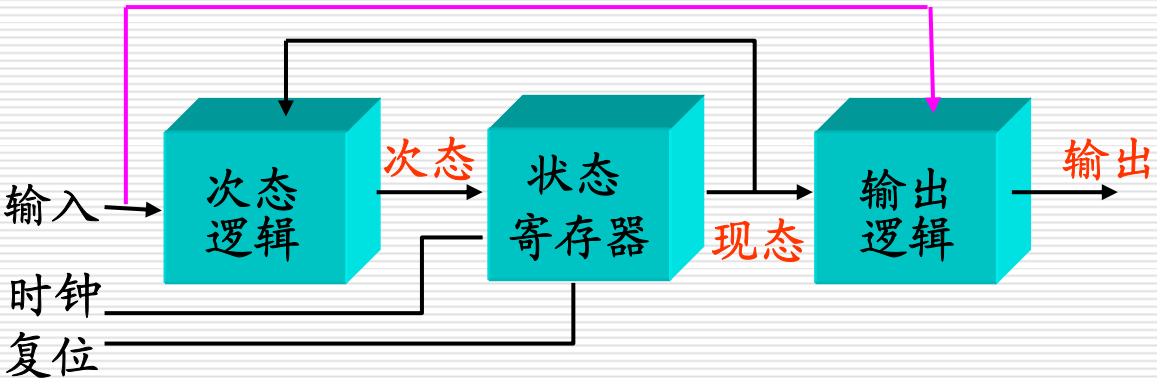
有限状态机

有限状态机类型

- 摩尔型（Moore）
- 米里型（Mealy）



摩尔型有限状态机的结构图



米里型有限状态机的结构图

描述有限状态机中：

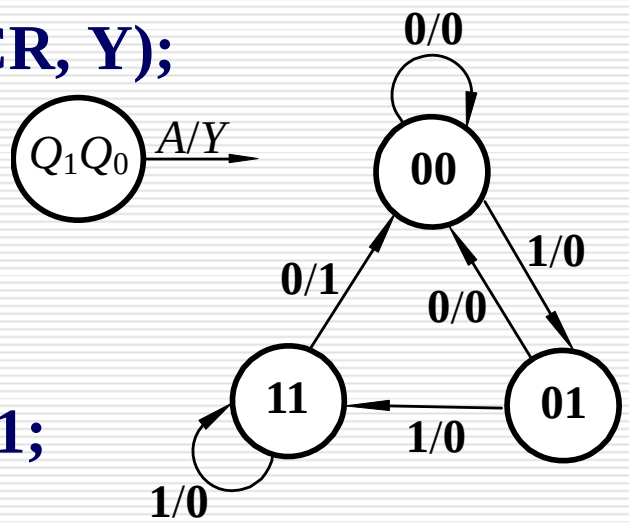
* 至少包含一个状态信号，它们用来指定有限状态机的状态。
* 包含状态转移指定和输出指定，它们对应于控制单元中与每个控制步骤有关的转移条件。
* 包含时钟信号，它是用来同步的。
* 包含同步或异步复位信号。

描述方式		进程描述功能	所用进程数
三进程描述方式		进程1：描述次态逻辑	3
		进程2：描述状态寄存器	
		进程3：描述输出逻辑	
双进程描述方式	形式1	进程1：描述次态逻辑、状态寄存器	2
		进程2：描述输出逻辑	
	形式2	进程1：描述次态逻辑	2
		进程2：描述状态寄存器、输出逻辑	
	形式3	进程1：描述次态逻辑、输出逻辑	2
		进程2：描述状态寄存器	
单进程描述方式		进程1：描述次态逻辑、状态寄存器和输出逻辑	1

6.7.3 状态图的Verilog建模实例

用Verilog描述状态图非常方便，常用always或case语句

```
module  Mealy_sequence_detector (A, CP, CR, Y);
    input A, CP, CR;
    output Y;
    reg Y;
    reg [1:0] current_state, next_state;
    parameter  S0=2'b00, S1=2'b01, S2=2'b11;
    always @( negedge CP or negedge CR)
    begin
       if (~CR)  current_state <=S0;    //在CR下降沿设s0为初态
       else        current_state <= next_state;
    end
```



6.7.3 状态图的Verilog建模实例

第二个always是将current_state和输入A作为敏感变量

```
always @(current_state or A)
 begin
  case(current_state)
       S0: begin Y<=0; next_state=(A==1)? S1: S0; end
       S1: begin Y<=0; next_state=(A==1)? S2: S0; end
       S2: if (A==1)
              begin Y<=0; next_state<=S2; end
           else
              begin Y=1; next_state<=S0; end
       default: begin Y<=0; next_state<=S0; end
  endcase
 end
endmodule
```