第四章　可编程逻辑器件

学习要求：

1. 了解可编程逻辑PLD的发展、结构分类、编程工艺、表示方法、设计过程
2. 熟悉PROM、可编程逻辑阵列PLA、可编程阵列逻辑PAL和通用逻辑阵列GAL
3. 掌握现场可编程门阵列FPGA、在系统可编程ISP的结构与编程原理
4. 熟悉并掌握硬件描述语言VHDL

4.2简单可编程逻辑器件（SPLD）

1. PROM
2. PLA
3. PAL
4. GAL

4.2 简单可编程逻辑器件（SPLD）

简单可编程逻辑器件SPLD：“与-或”表达式

只读存储器 (ROM)： 与阵列固定或阵列可编程

可编程逻辑阵列 (PLA) ：与阵列可编程 或阵列可编程

可编程阵列逻辑 (PAL) ：与阵列可编程 或阵列固定

通用逻辑阵列(GAL)： 兼容PAL，增加可擦除、可重新编程及可组态结构等特点。使用最广泛的PLD产品之一。

目前流行的可编程逻辑器件有FPGA和ISP，是更高层次的CPLD，技术上更先进。

4.2.1 可编程只读存储器（PROM）

一个只能在制造（或编程）时方可将信息写入，而在应用时仅能读出的存储器。（不可再编程。）

特点：与阵列——固定　或阵列——可编程

只读存储器按内部结构可分为：

- 固定只读存储器　　ROM　　（掩模可编程）
- 可编程只读存储器　PROM　　（一次可编程）
- 可擦除可编程只读存储器　EPROM （可擦除可编程）
- 电可擦除可编程只读存储器 EEPROM
  （电可擦除可编程）

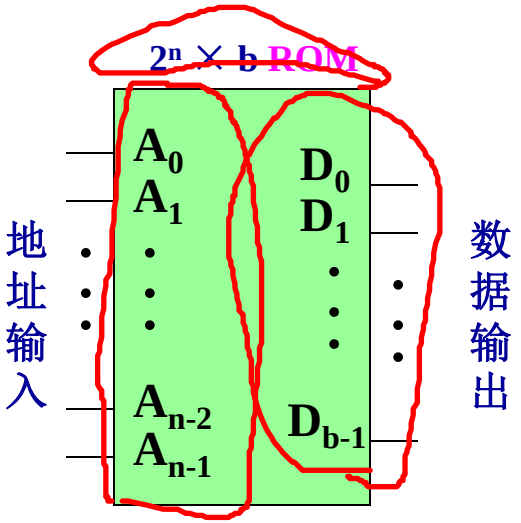
早期ROM： 在制造过程中厂商根据用户提供的存储内容设计光刻掩模板来存储信息，其存储内容在出厂后不能改变的固定芯片。即内容是在装入整机前事先写好的，工作过程中只能读出。

一次可编程ROM（PROM）： 内部存储位有二极管或晶体管连接，在交付前通常为相同位“1”，电击毁方式，编程点由地址线和数据线确定，高压脉冲由专门引脚输入。为一次性可编程。

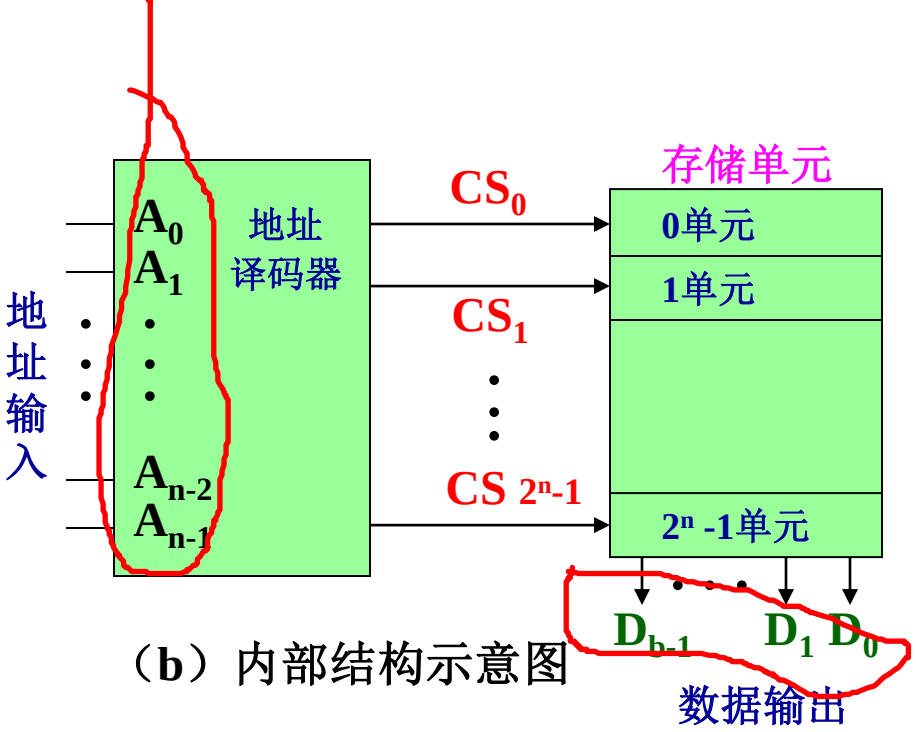
光可擦除可编程ROM（EPROM）： 浮栅MOS管连接，有两个门，被高绝缘材料包围；之前无电荷，MOS管截止（存1），编程时加高压，是电荷雪崩注入到浮栅，从而导通（存0），可以保存10年。紫外线照射擦除。

电可擦除可编程ROM（EEPROM）： 高电场完成擦除。

1 ROM存储器



（a）基本结构

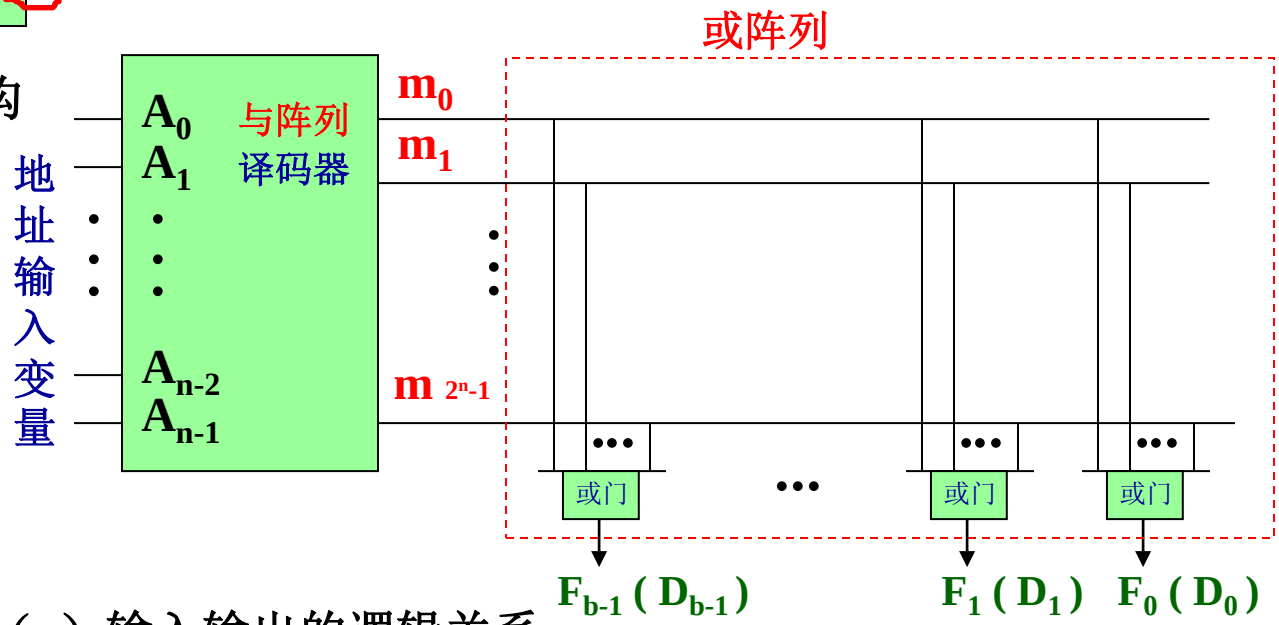


（b）内部结构示意图

若第j单元的第i位为jbi：

$$F_i = D_i = \sum_{j=0}^{2^n-1} jbi \cdot m_j$$

$$i = 0,1,2,\ldots,b-1$$

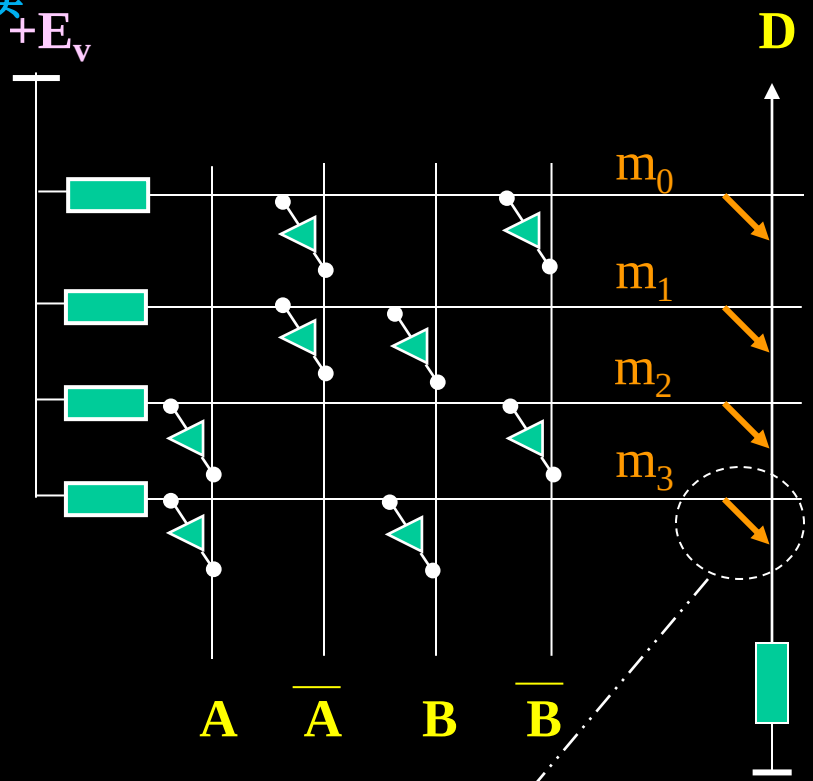


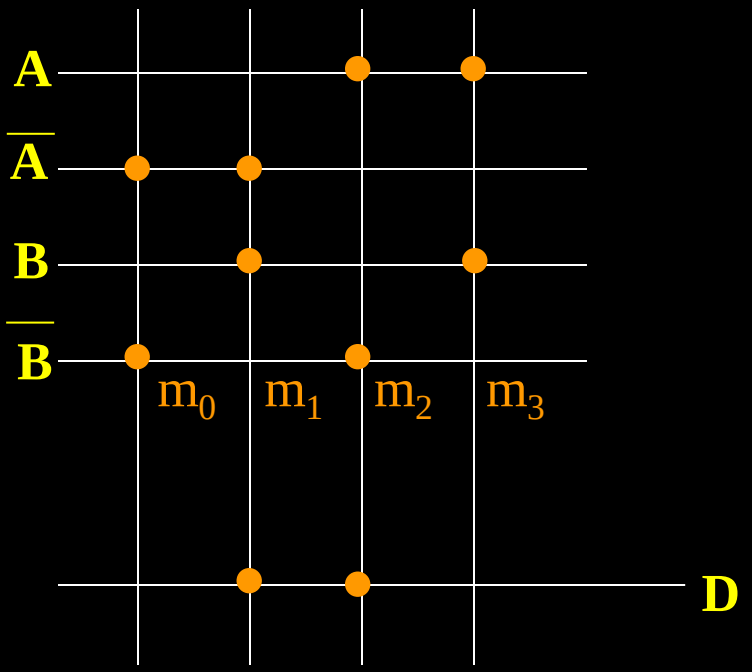
（c）输入输出的逻辑关系

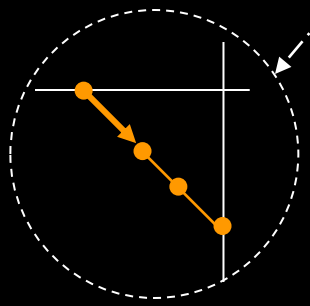
2 ROM 的内部结构及其点阵图：

1)二极管耦合的4×1ROM的电路图及阵列图

阅读





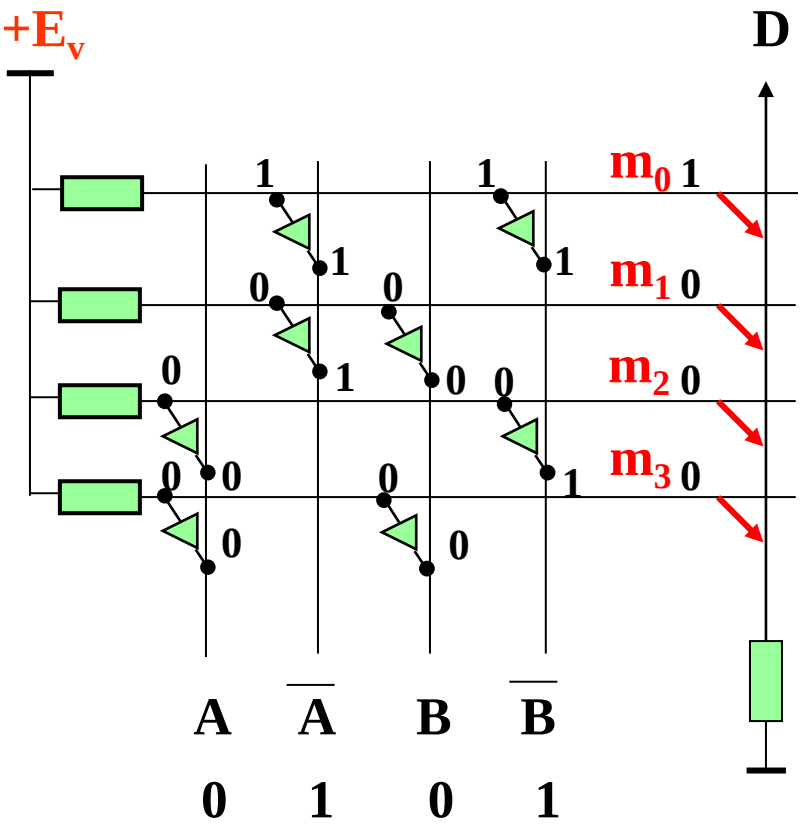


熔丝断开表示存“0”，

熔丝连接表示存“1”。

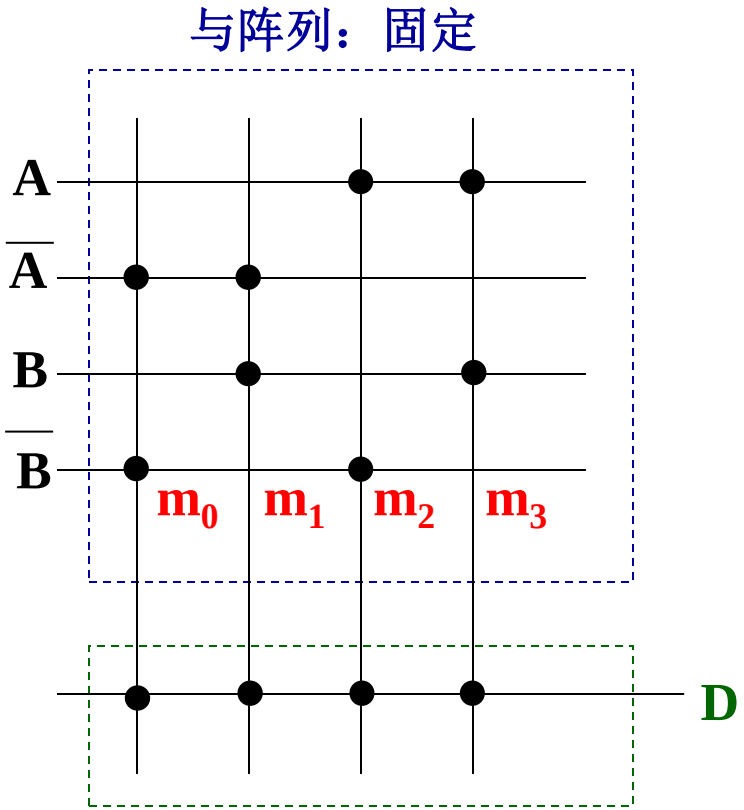
$$D = \overline{A}B + A\overline{B}$$

阅读



$D = 1\cdot m_0 + 1\cdot m_1 + 1\cdot m_2 + 1\cdot m_3$

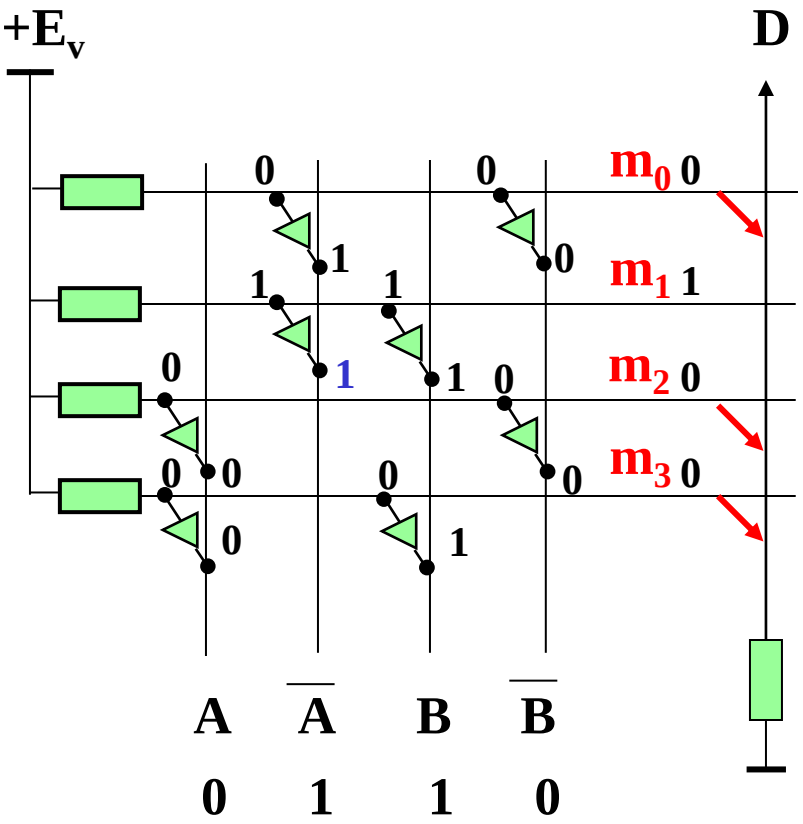
$= 1\cdot 1 + 1\cdot 0 + 1\cdot 0 + 1\cdot 0 = 1$



$D = m_0 + m_1 + m_2 + m_3$

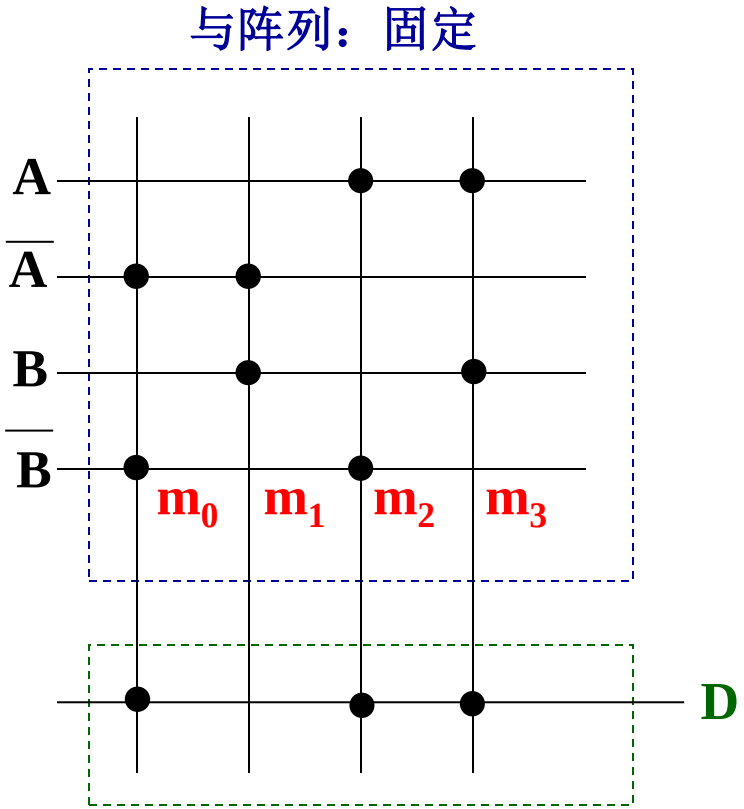
数字逻辑电路

阅读



$D = 1\cdot m_0 + 0\cdot m_1 + 1\cdot m_2 + 1\cdot m_3$

$= 1\cdot 0 + 0\cdot 1 + 1\cdot 0 + 1\cdot 0 = 0$



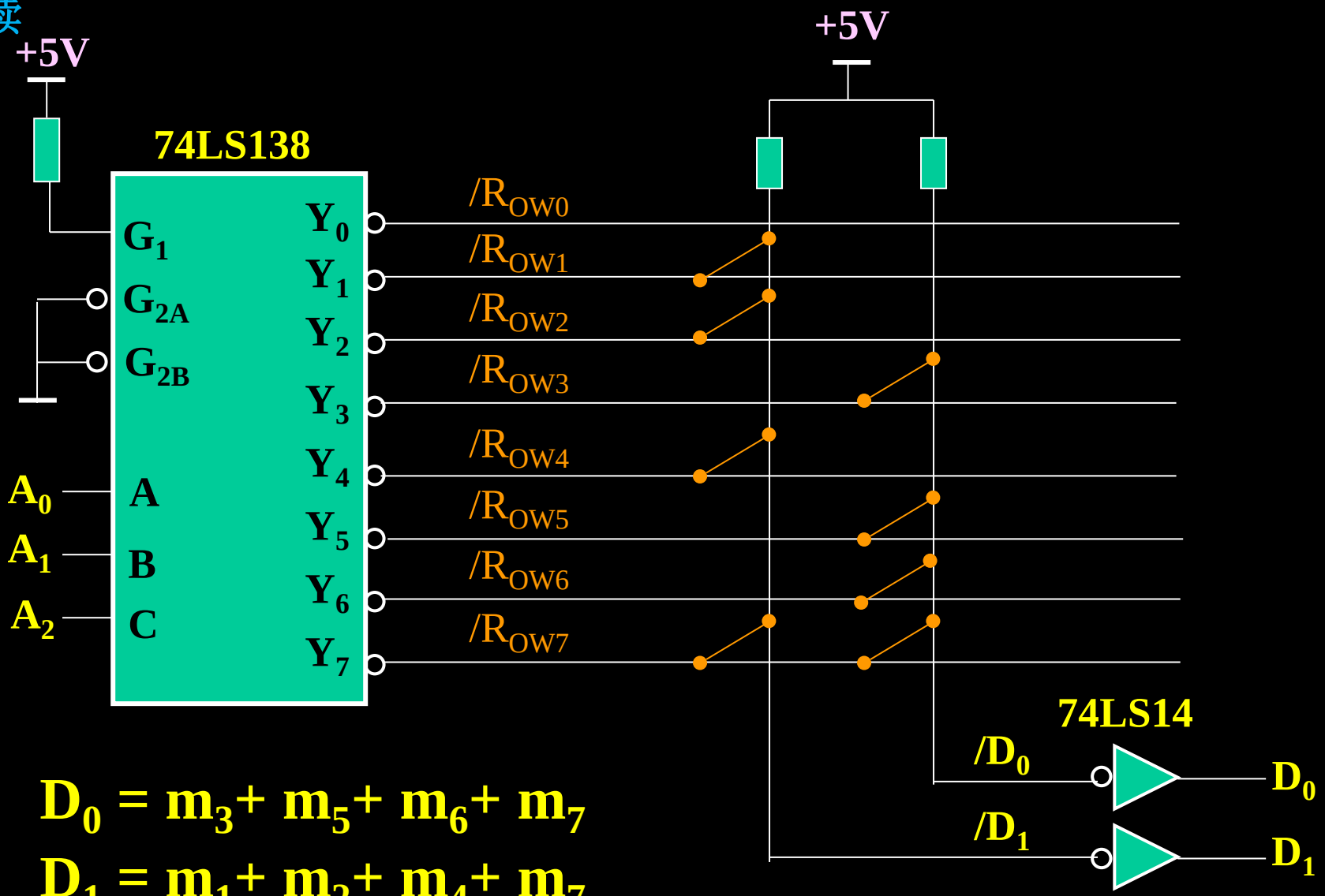
$D = m_0 + m_2 + m_3$

数字逻辑电路

ROM 的内部结构及其点阵图：

2) 用 TTL 电路构成的 8 ×2 ROM的逻辑图

阅读

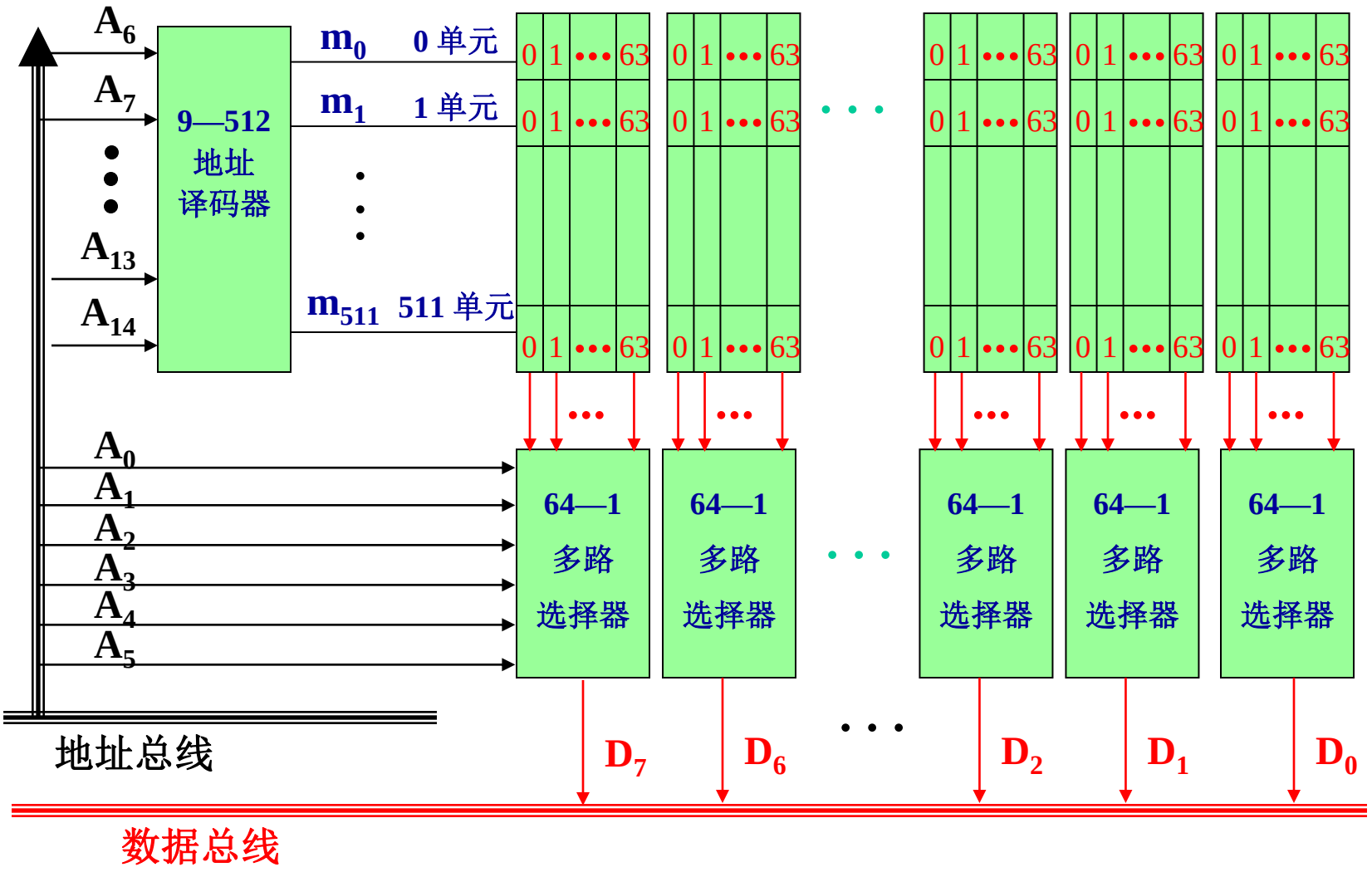


$D_0 = m_3 + m_5 + m_6 + m_7$

$D_1 = m_1 + m_2 + m_4 + m_7$

3）32K ×8 EPROM组成框图

（512 ×64 存储阵列）×8 位 = 512 ×512



1. 掩模ROM的阵列结构和存储元

大部分ROM芯片利用在行选线和列选线交叉点上的晶体管是导通或截止来表示存 0 、 1。

阅读

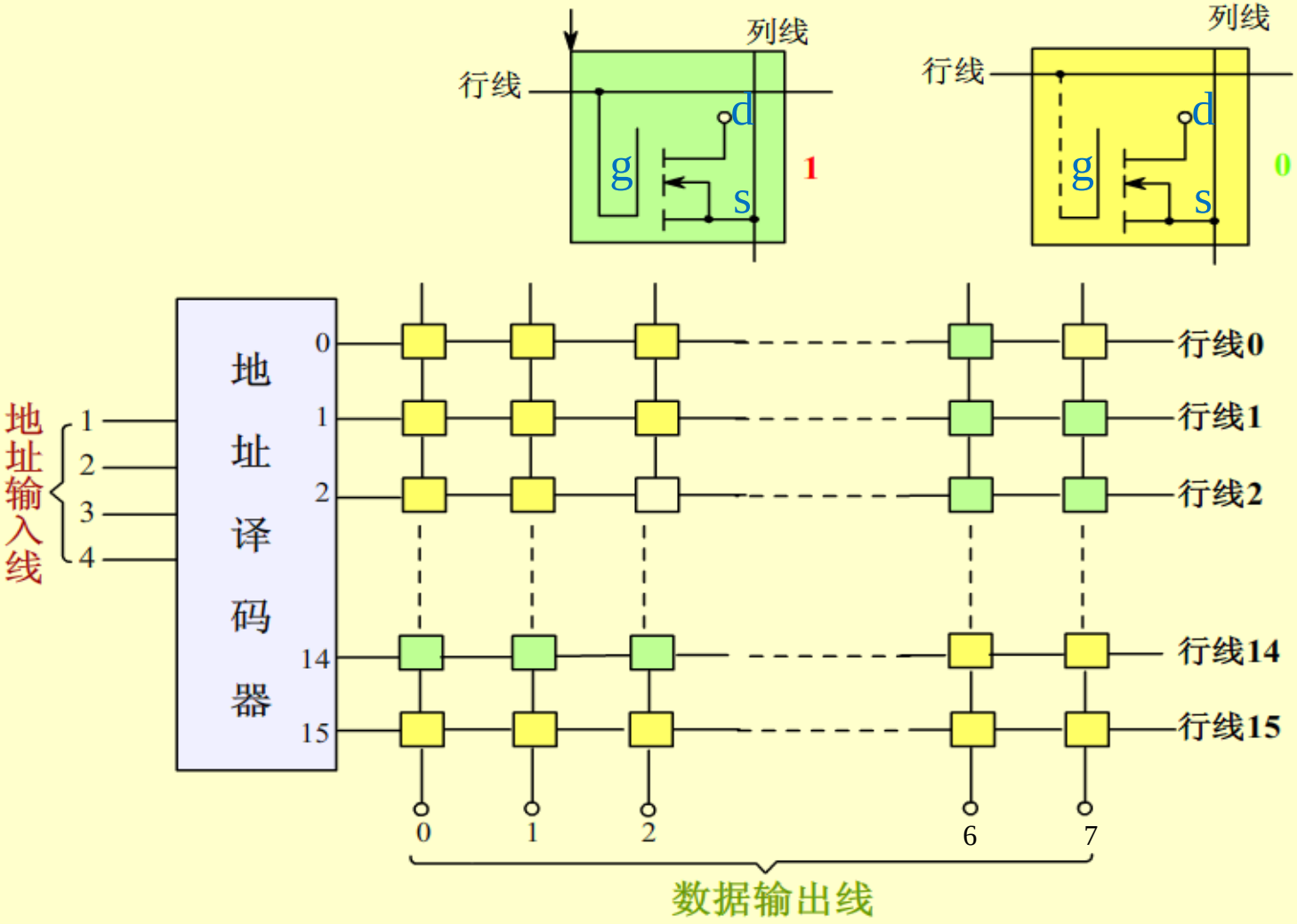


图4.7　16×8位ROM阵列结构示意图

1. EPROM存储元

EPROM： 光擦除可编程只读存储器

阅读

例： 浮栅雪崩注入型MOS管为存储元的EPROM

有两个栅极：G 1 （浮空栅）和G 2 （控制栅）

器件的上方有一个石英窗口，当用光子能量较高的紫外光照G 1 （浮空栅）时，存储器存“ 1 ”（出厂状态）。

允许多次重写。

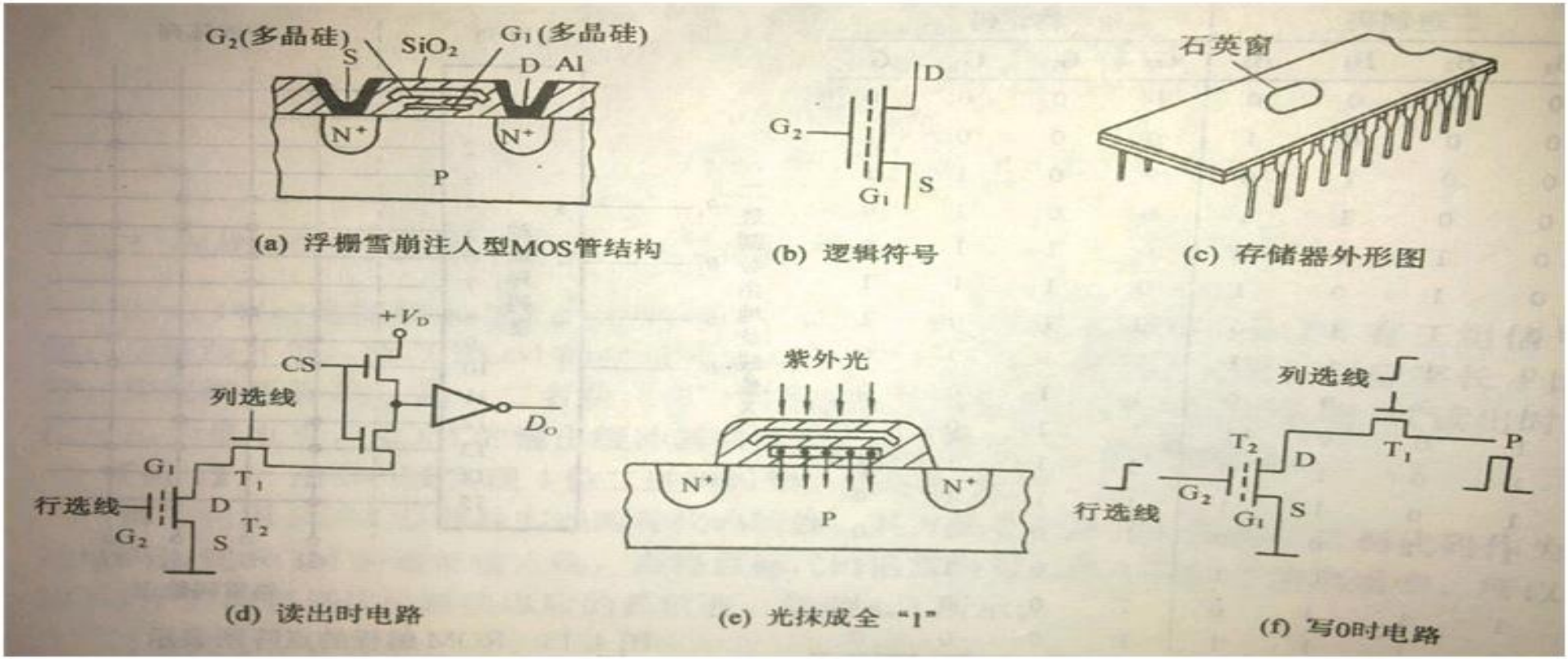


图4.16 EPROM存储元

2. E^2PROM存储元

E^2PROM叫做电擦除可编程只读存储器。

阅读

其存储元是一个具有两个栅极的NMOS管

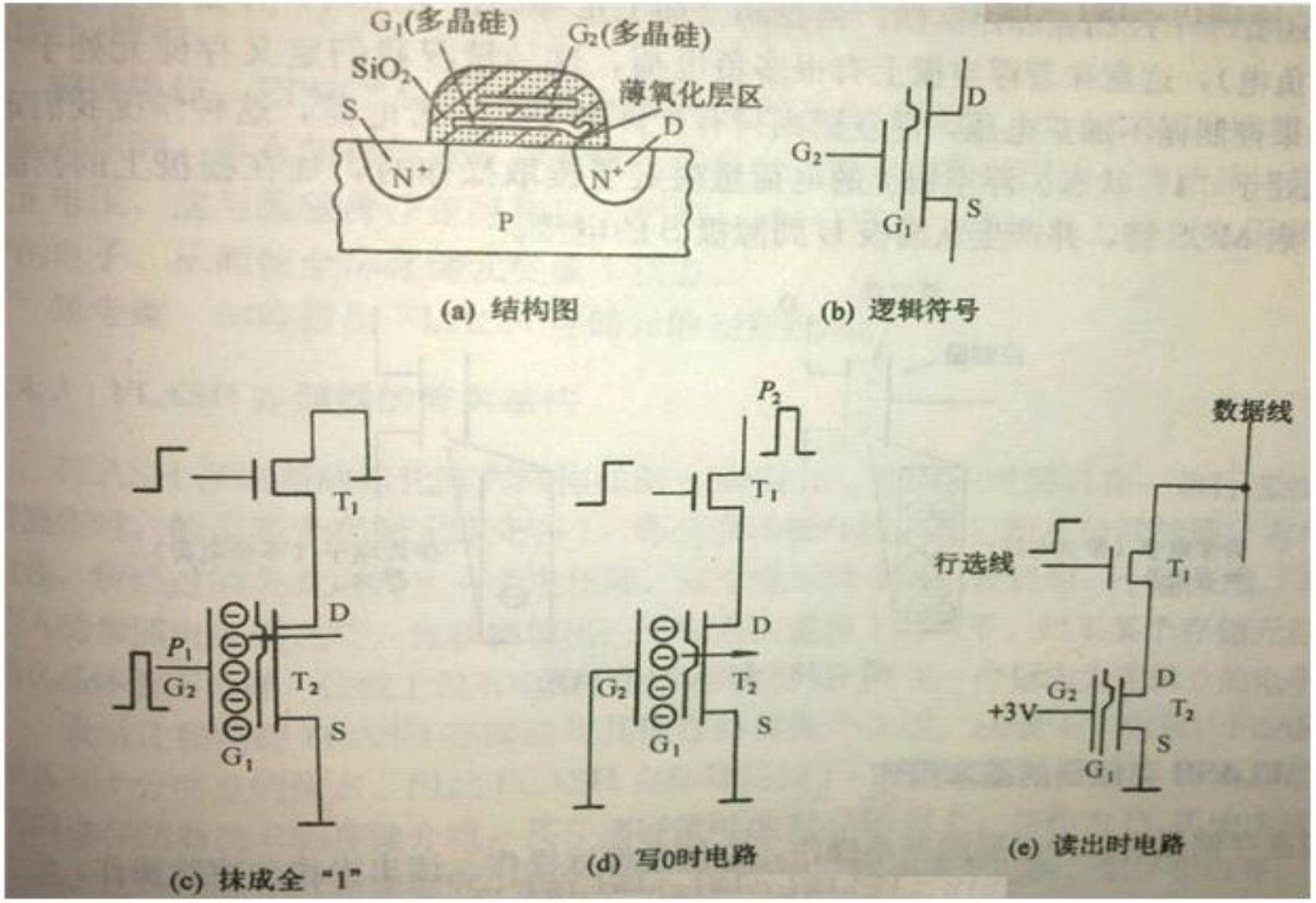
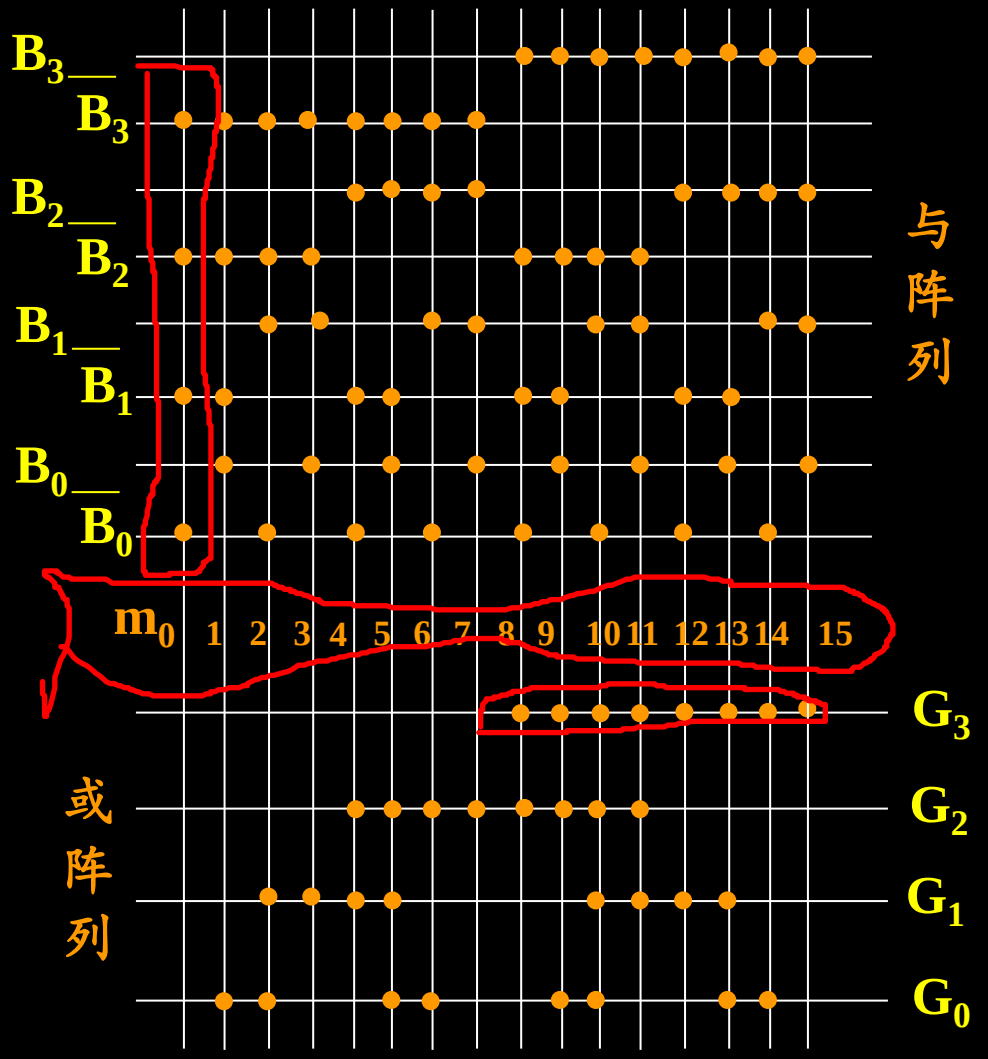


图4.17　E^2PROM存储元

3 ROM的应用—实现组合逻辑

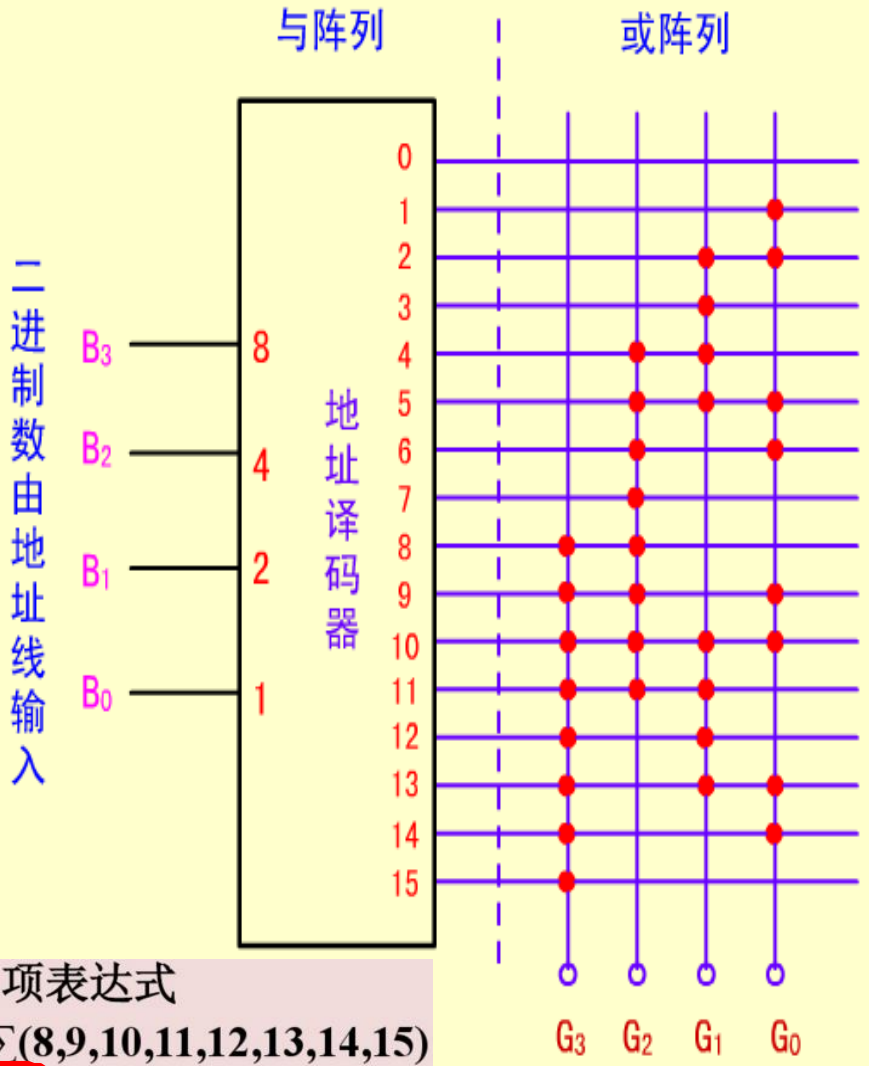
1）例1 将4位二进制数转换为Gray码。

B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0



阅读

二进制码				格雷码			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0



最小项表达式

$G_3=\sum(8,9,10,11,12,13,14,15)$

$G_2=\sum(4,5,6,7,8,9,10,11)$

$G_1=\sum(2,3,4,5,10,11,12,13)$

$G_0=\sum(1,2,5,6,9,10,13,14)$

ROM编程的点阵图表示

4.2.2 可编程逻辑阵列 (PLA)

特点：与、或阵列都可编程

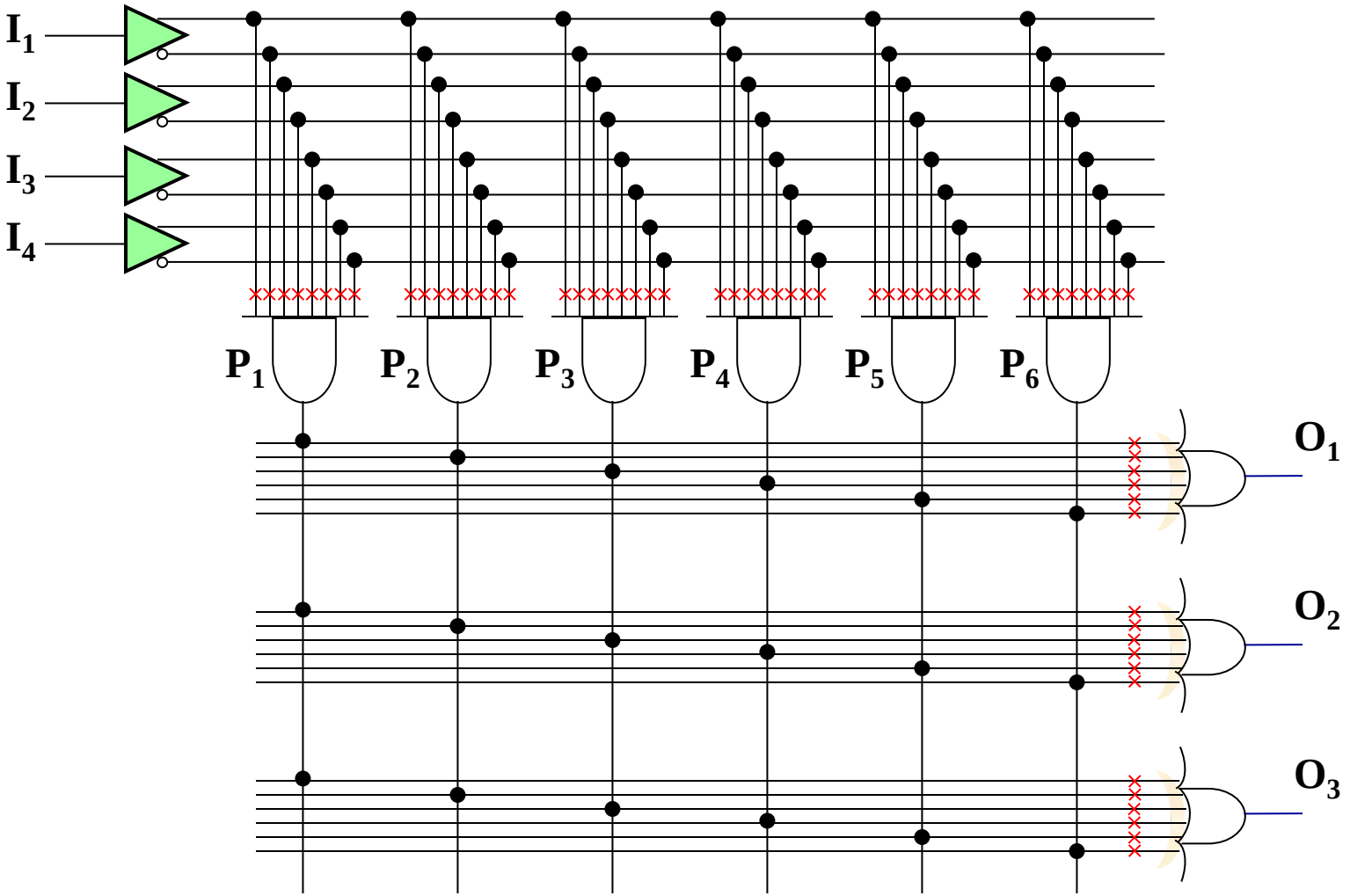
1) 针对逻辑函数的最简与或式——

- PLA中的与阵列被编程产生所需的全部与项
- PLA中的或阵列被编程完成相应与项间的或运算，并产生输出。逻辑功能越复杂，其优点越明显。

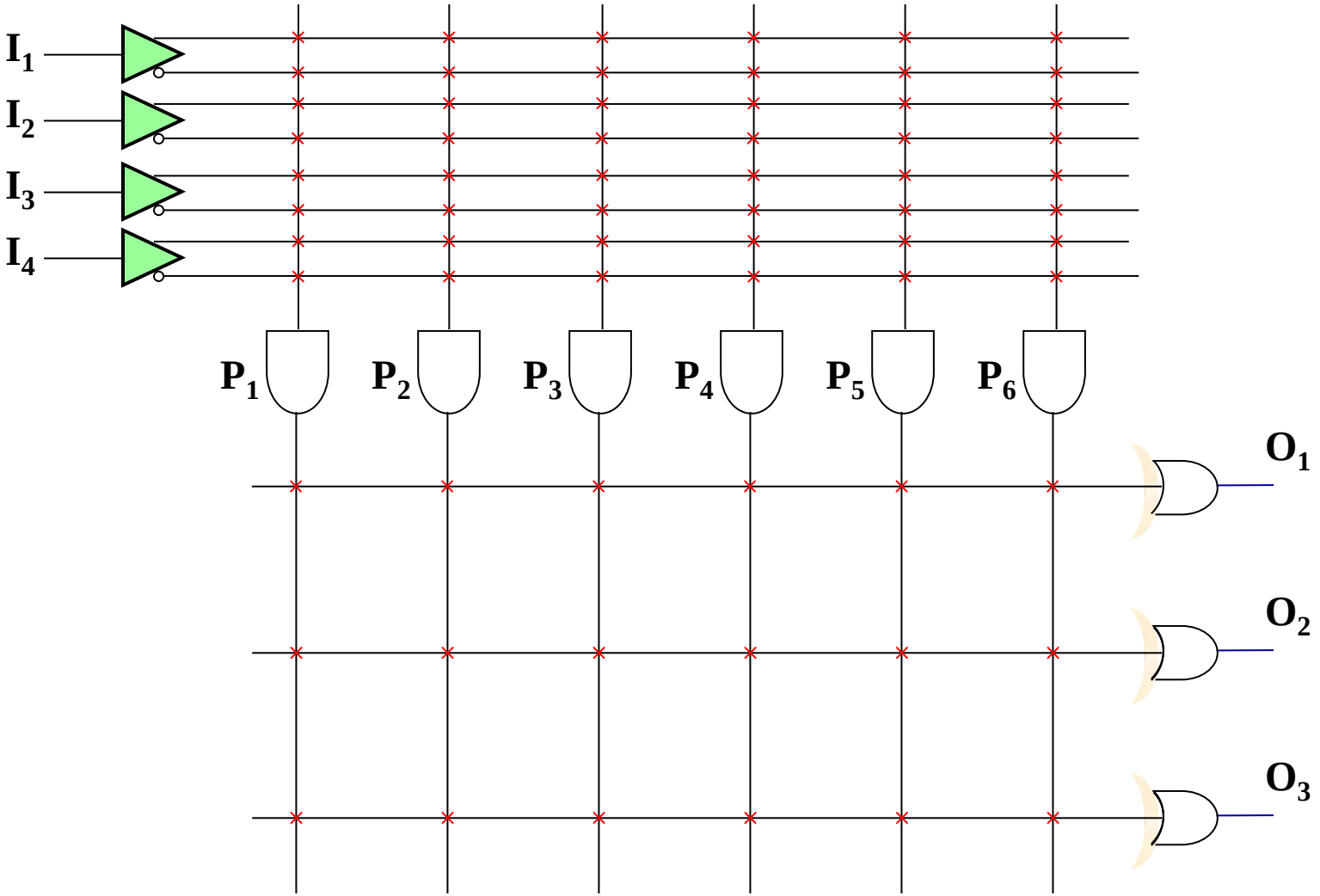
这样，就大大提高了芯片面积的有效利用率。

2) PLA分组合PLA和时序PLA(包含有触发器)。

例　具有6个与项的4×3PLA的电路。



例　具有6个与项的4×3PLA的电路。



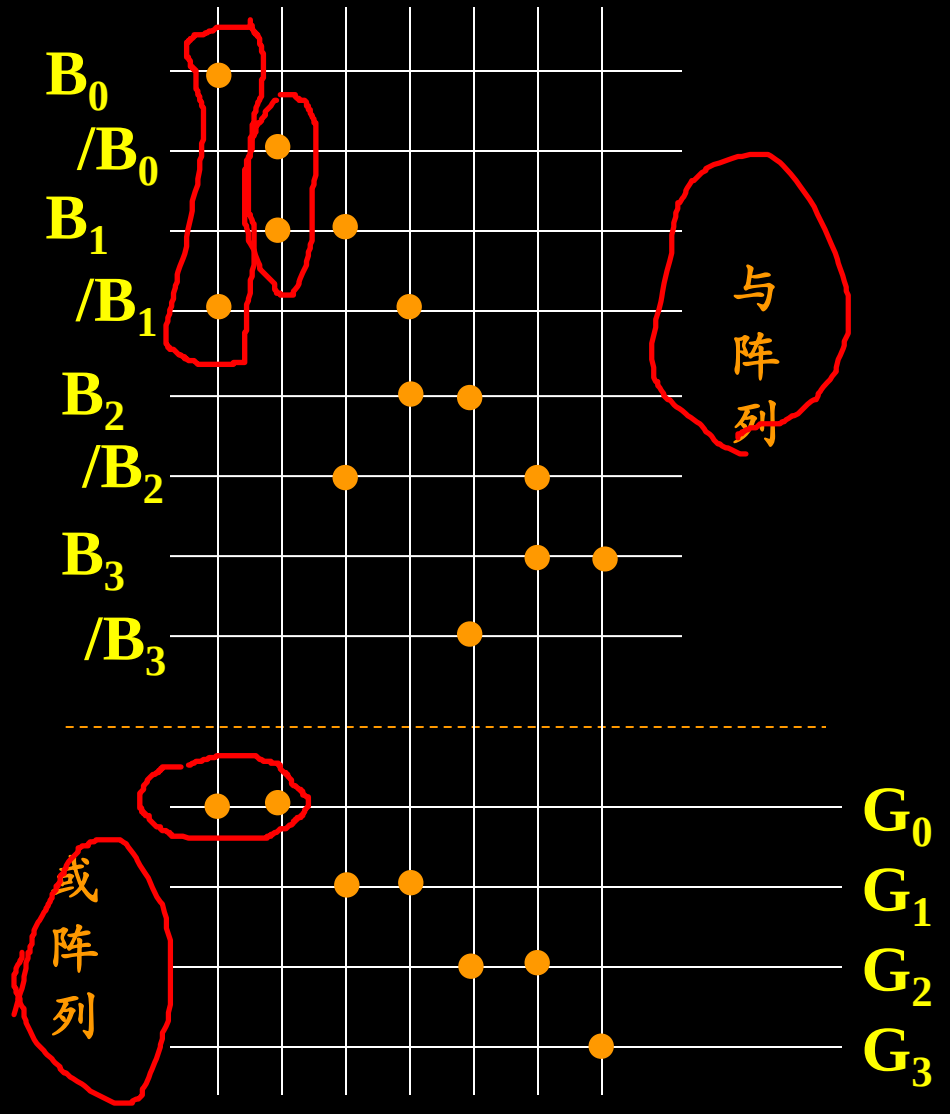
例　用PLA实现4位二进制数转换为Gray码的电路。

$$G_3 = B_3$$

$$G_2 = \overline{B_3}B_2 + B_3\overline{B_2}$$

$$G_1 = \overline{B_2}B_1 + B_2\overline{B_1}$$

$$G_0 = \overline{B_1}B_0 + B_1\overline{B_0}$$



4.2.3 可编程阵列逻辑 (PAL)

特点：

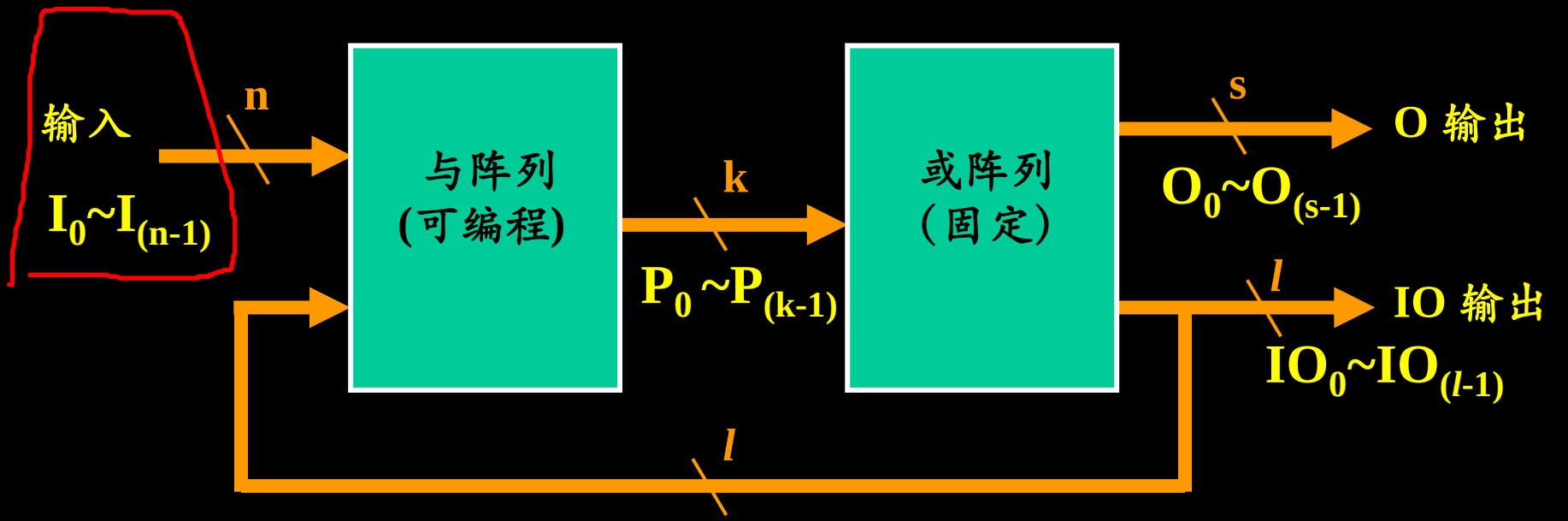
1) 可编程的与阵列和固定的或阵列

与同样位数的PLA相比，PAL减少了编程点数(或阵列固定)，简化了编程工作(仅对与阵列编程)。

- PAL器件内所提供的与项数目较少，通常为7～8个，但是典型的逻辑设计中一般只需要4～5个与项。
- 这种结构也提供了较高的性能和速度，更有利于辅助设计系统的开发，所以在较长时间里PAL成为了PLD发展史上的主流。

2) PAL分组合PAL和时序PAL（包含有触发器）。

1) 组合 PAL 器件



组合PAL 的基本结构框图

每组与阵列的输出被固定连接到一个或阵列的输入端，因此或阵列不能共享与门的输出。

1) 组合 PAL 器件

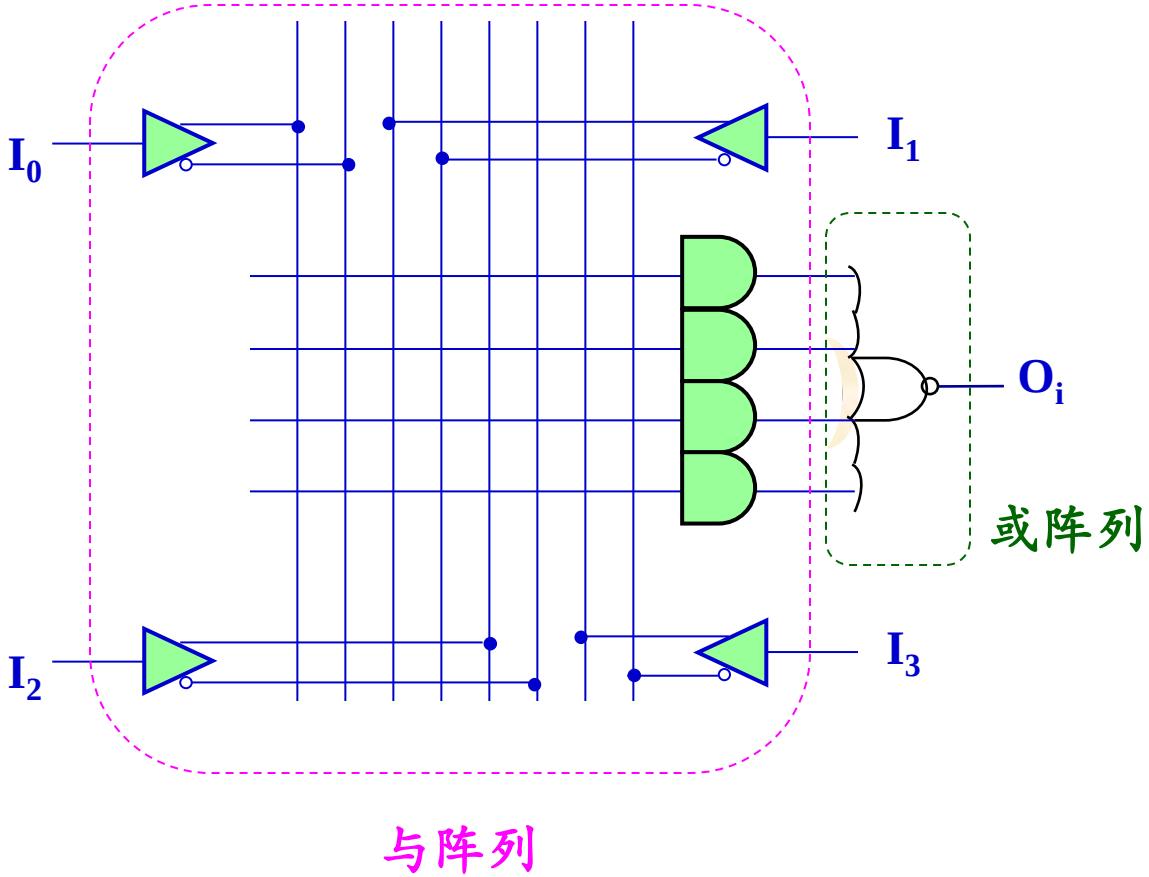
阅读

组合PAL器件的基本结构如上图所示，输出不带有寄存器，输出引脚分为两类：纯组合O输出和IO输出，每一个 O 输出或 IO 输出都对应一个独立的可编程与门阵列和固定的或门阵列。图中应有s+*l*个独立的与或阵列。

每个可编程与阵列可根据$I_{[0\sim(n-1)]}$和$IO_{[0\sim(l-1)]}$生成一组任意的输出$P_{[0\sim(k-1)]}$，与项通常为7～8个。每组与阵列的输出被固定连接到一个或阵列的输入端。因此,或阵列之间不能共享与门的输出。如果用PAL器件实现某一逻辑功能时，若有两个输出或门有相同的与项输入，则必须由相应的两个与阵列产生两个相同的与项。

1) 组合 PAL 器件

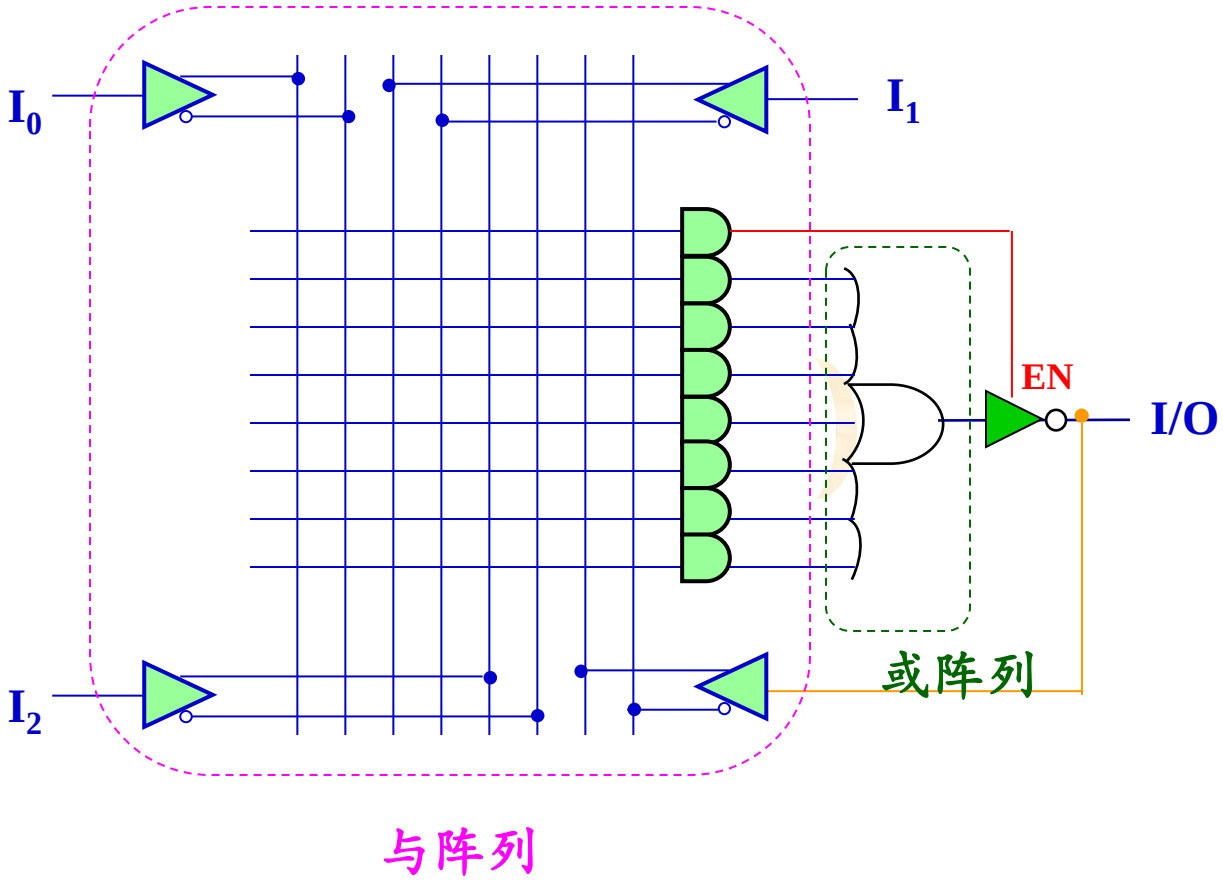
1. 基本与或阵列结构，如图所示：



$O_i = f(I_0, I_1, I_2, I_3)$

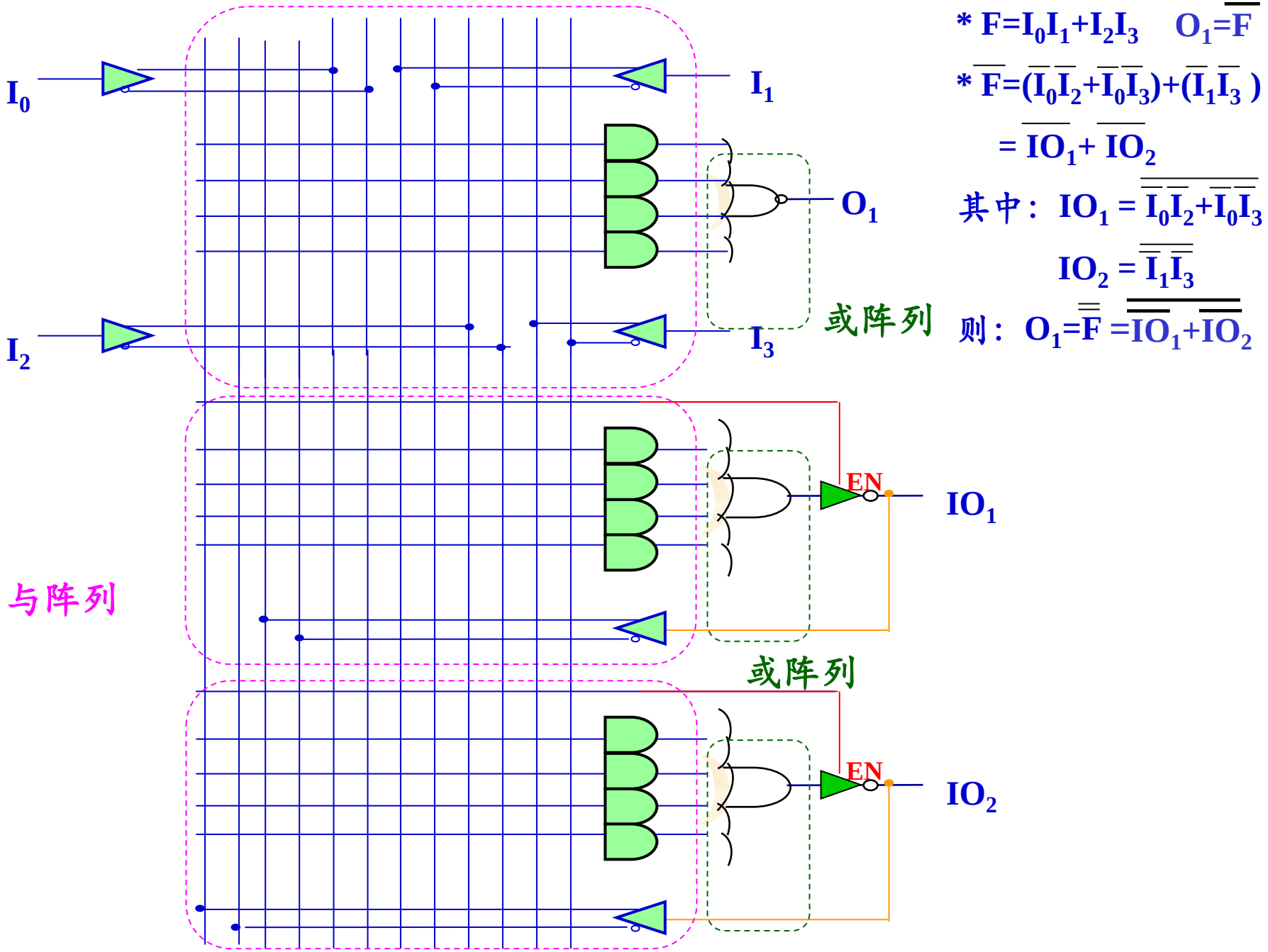
1) 组合 PAL 器件

2. 异步可编程I/O结构(三态输出)，如图所示：



三态门禁止，输出呈高阻态时，I/O引脚做输入用；

三态门被选通时，I/O引脚作输出，以及反馈输入用（中间变量）。



* $F=I_0I_1+I_2I_3$ $O_1=\overline{F}$

* $\overline{F}=(\overline{I}_0\overline{I}_2+\overline{I}_0\overline{I}_3)+(\overline{I}_1\overline{I}_3)$

$= \overline{IO_1}+\overline{IO_2}$

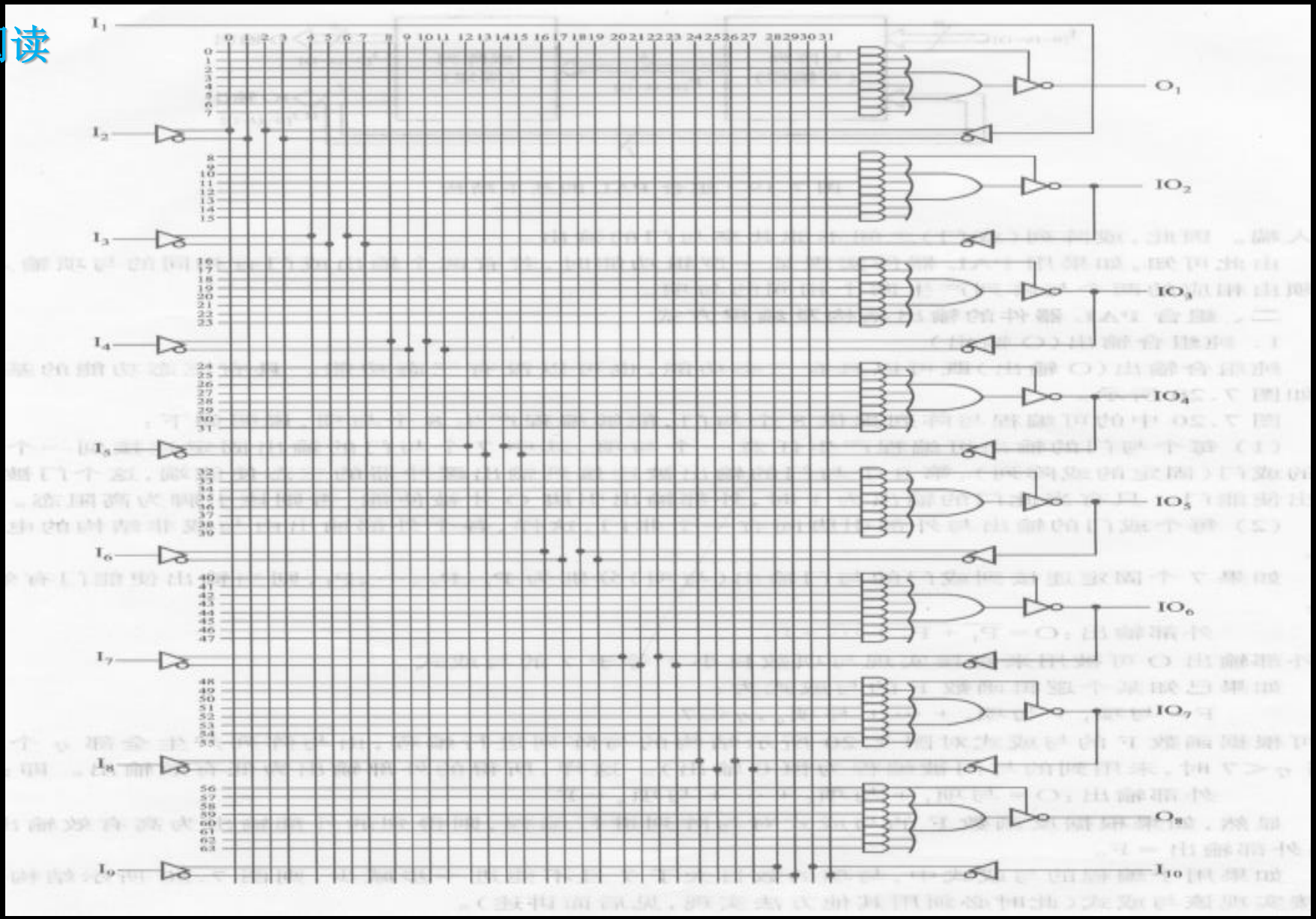
其中：$IO_1 = \overline{\overline{I}_0\overline{I}_2+\overline{I}_0\overline{I}_3}$

$IO_2 = \overline{\overline{I}_1\overline{I}_3}$

则：$O_1=\overline{\overline{F}} =\overline{\overline{IO_1}+\overline{IO_2}}$

1) 组合 PAL 器件具有三态输出功能的组合PAL逻辑结构：

阅读



1) 组合 PAL 器件(1)纯组合O输出

阅读

图中的可编程与阵列提供8个与门,能编程产生8个与项,说明如下:

①每个与门的输出可编程产生任意一个与项,其中7个与门的输出固定连接到一个7输入固定的或门阵列,第8个与门的输出被连接到输出缓冲器的三态使能端,这个门被称为输出使能门。

②每个或门的输出与外部引脚间有一个非门,这样各个外部输出由与或非结构的电路产生。如果7个固定连接到或门的与门输出分别为P_1、P_2、•••、P_7,则当输出使能门有效时,有:

外部输出: /O=P_1+P_2+…+P_7 ,此外部输出O可被用来直接实现与项数目小于等于7的与或式。这样所得的外部输出为低有效输出。即: /O=与项$_1$+与项$_2$+…+与项$_q$=F, q≤7。

如果根据反函数的与或式/F对与阵列进行编程,则得到的外部输出为高有效输出,即有: 外部输出O=F。

1) 组合 PAL 器件(1)纯组合O输出

阅读

图中的可编程与阵列提供8个与门,能编程产生8个与项,说明如下:

③对于PAL器件的应用来说,函数化简的首要目标是减少实现这一逻辑功能的成本,其关键是减少用于编程的与或式中与项的数目。但是,在化简减少与项的数目已不影响其实现的成本时,就没有必要再继续进行了。

例如,如果用于编程的与或式中,与项的数目小于或等于7,则用函数化简的方法进一步减少与项的数目或减少与项所包含的变量数,并不能减少用图所示结构实现该与或式的代价。

同理,某个逻辑函数的与或式有8个与项,如果用函数化简的方法减少一个与项,则可直接用图所示结构实现,此时就不必进一步化简了。

1) 组合 PAL 器件(1)纯组合O输出

阅读

从上述的分析中大家知道,用PAL器件实现某一逻辑函数时,如果需要高有效输出,则应根据反函数的与或式进行编程；如果需要低有效输出,则应根据原函数的与或式进行编程,且实现该函数的代价主要取决于用于编程的与或式中与项的数目。这样,在某些场合,为了减少实现某一逻辑函数的代价,需要有目的地选择输出的有效级。

例如,逻辑函数为： $F=I_1 \cdot I_2 \cdot I_3 \cdot I_4 \cdot I_5 \cdot I_6 + I_5 \cdot I_6 \cdot I_7 \cdot I_8 \cdot I_9$

1) 组合 PAL 器件(1)纯组合O输出

阅读

例如,逻辑函数为： $F=I_1\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot I_6+I_5\cdot I_6\cdot I_7\cdot I_8\cdot I_9$

（1）如果采用低有效输出,则函数F的与或式中仅有2个与项;

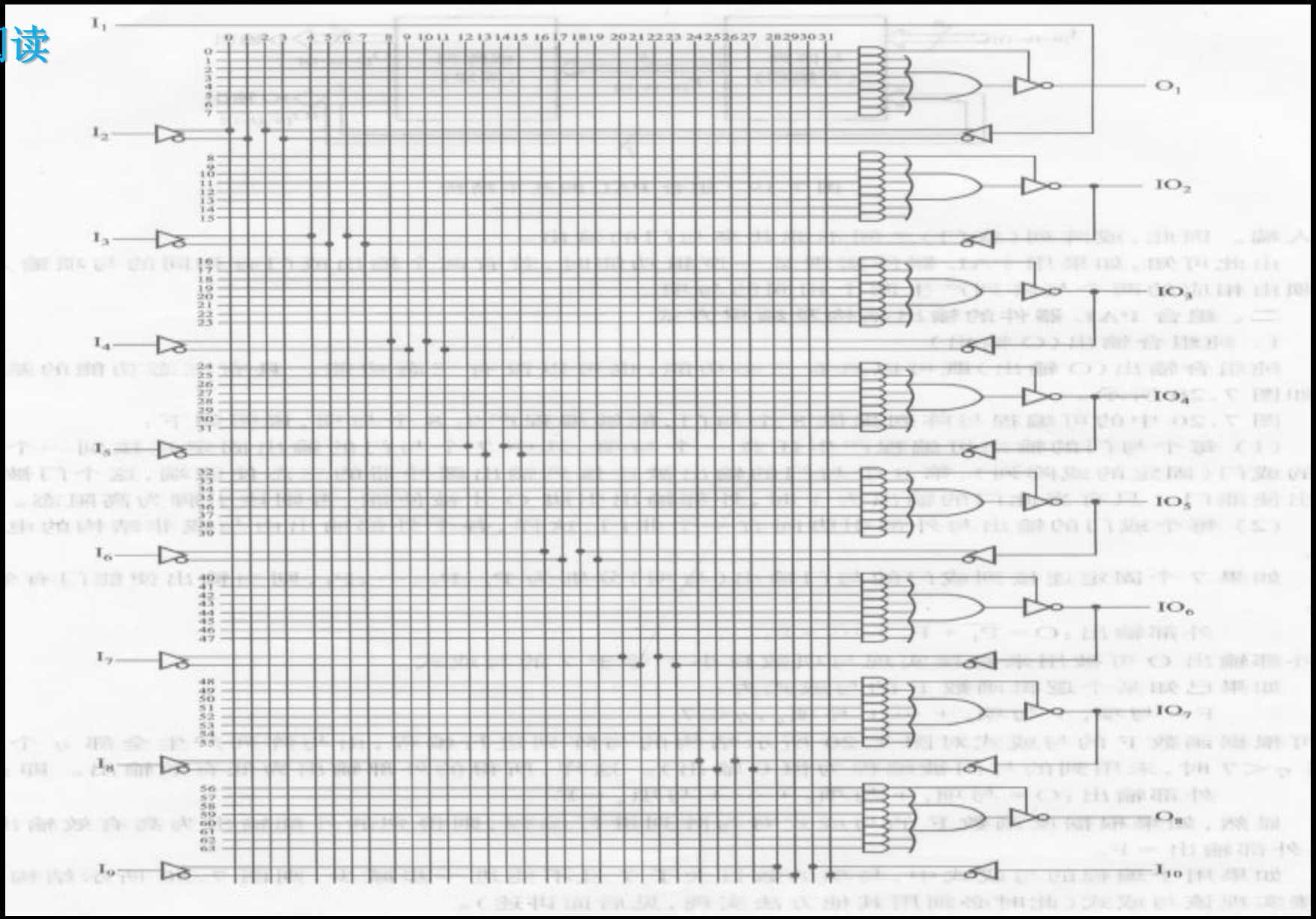
（2）但如果采用高有效输出,则反函数的与或式:

$$\overline{F}=\overline{I_1}\cdot\overline{I_7}+\overline{I_1}\cdot\overline{I_8}+\overline{I_1}\cdot\overline{I_9}+\overline{I_2}\cdot\overline{I_7}+\overline{I_2}\cdot\overline{I_8}+\overline{I_2}\cdot\overline{I_9}+\overline{I_3}\cdot\overline{I_7}+\overline{I_3}\cdot\overline{I_8}+\overline{I_3}\cdot\overline{I_9}$$
$$+\overline{I_4}\cdot\overline{I_7}+\overline{I_4}\cdot\overline{I_8}+\overline{I_4}\cdot\overline{I_9}+\overline{I_5}+\overline{I_6}$$

其中包含14个与项,因此实现此与或式必须占用多个独立的与或阵列,代价较大。显然,此时采用低有效输出较为合适。但是,如果在与或式中,与项的数目大于7且不能进一步减少,则图所示结构不能使用一个与或阵列直接实现该与或式,此时必须使用中间变量利用IO输出等其他方法实现。

1) 组合 PAL 器件(2) IO输出

阅读



1) 组合 PAL 器件(2) IO输出

阅读

从图4.13中可以看出,IO输出端具有三态输出功能,输出使能门的控制端可以为某个与项函数。图中与或阵列的结构与纯组合O输出相同,唯一的区别是IO输出端又通过反馈线及缓冲门连接到与阵列的输入。如果作为纯输出时,其反馈输入信号不能被与阵列中的任何门使用。这类输出端除能完成O输出的同等功能外,还具有如下功能:

①实现输入功能:当使能控制端函数值为0时,引脚为输入引脚的扩展,以增加输入变量数。此时,该输出三态门处于高阻抗;

②实现内部反馈输入功能:当使能控制端函数值为1时,用来产生中间变量,并利用扩展某个输出所包含的与项数目,即用4级电路“与-或-与-或”实现某一逻辑函数;

③用来实现电平异步时序电路的激励函数。此时,该引脚的输出使能控制端恒为l。

1) 组合 PAL 器件(2) IO输出

阅读

例如,逻辑函数为： $F=I_1\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot I_6+I_5\cdot I_6\cdot I_7\cdot I_8\cdot I_9$

第一种方法是应根据反函数的与或式进行编程。反函数的与或式中有14个与项。由于每一个与或阵列最多可直接实现具有7个与项的与或式,因此,至少需要两个中间变量。可分别令两个中间变量MF_1和MF_2为:

$$MF_1=\overline{I_1}\cdot\overline{I_7}+\overline{I_1}\cdot\overline{I_8}+\overline{I_1}\cdot\overline{I_9}+\overline{I_2}\cdot\overline{I_7}+\overline{I_2}\cdot\overline{I_8}+\overline{I_2}\cdot\overline{I_9}+\overline{I_3}\cdot\overline{I_7}$$

$$MF_2=\overline{I_3}\cdot\overline{I_8}+\overline{I_3}\cdot\overline{I_9}+\overline{I_4}\cdot\overline{I_7}+\overline{I_4}\cdot\overline{I_8}+\overline{I_4}\cdot\overline{I_9}+\overline{I_5}+\overline{I_6}$$

这样有： $\overline{F}=MF_1+MF_2$

根据MF_1和MF2的与或式分别对输出引脚IO_2和IO_3进行编程,得到的输出引脚功能为： $IO_2=MF_1$， $IO_3=MF_2$

1) 组合 PAL 器件(2) IO输出

阅读

例如,逻辑函数为：$F=I_1\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot I_6+I_5\cdot I_6\cdot I_7\cdot I_8\cdot I_9$

注意到反馈线通过输入缓冲器同时给与阵列提供原、反两种形式的变量,因此,实现中间变量功能的输出引脚IO_2和IO_3的输出有效级可任选。这样,根据式$/F=MF_1+MF_2$对输出引脚O_1进行编程,即得:

$$O_1=\overline{\overline{F}}=\overline{MF_1\cdot MF_2}=IO_2\cdot IO_3=F\text{（高有效）}$$

1) 组合 PAL 器件(2) IO输出

阅读

例如,逻辑函数为: $F=I_1\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot I_6+I_5\cdot I_6\cdot I_7\cdot I_8\cdot I_9$

第二种方法是根据原函数的与或式,令中间变量MF为:

$MF=I_1\cdot I_2\cdot I_3\cdot I_4\cdot I_5\cdot I_6+I_5\cdot I_6\cdot I_7\cdot I_8\cdot I_9$

则有:

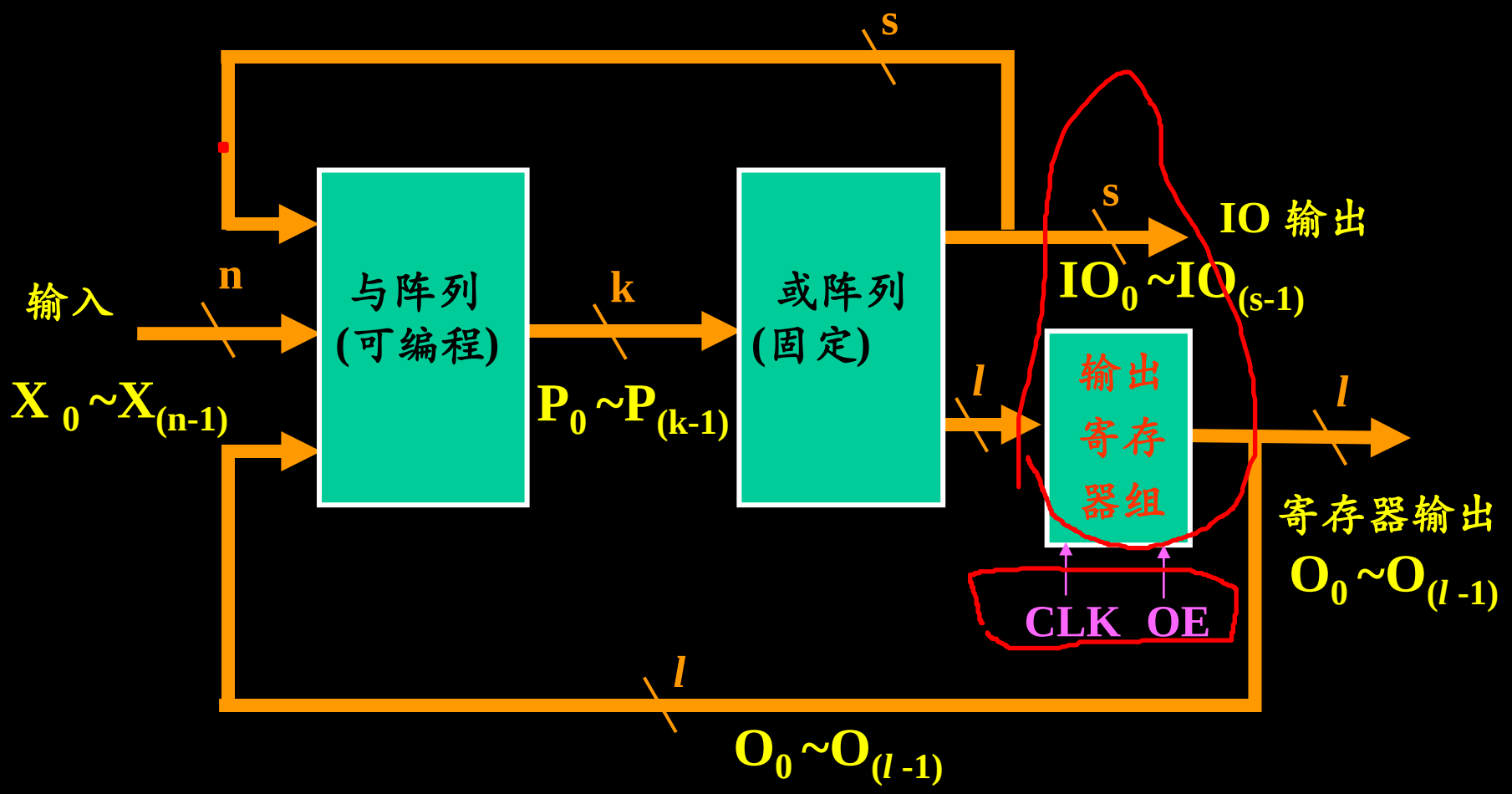
$F=MF$

其反函数为: $\overline{F}=\overline{MF}$

根据MF的表达式和上式的表达式分别对输出引脚O_1和IO_2进行编程,即可得$O_1=F$,仅占用两个输出引脚O_1和IO_2。

显然,第二种方法优于第一种方法。但如果以低有效输出的方式实现函数F的功能,仅需占用一个输出引脚,则这种方法更优。因此,在用PAL器件实现的数字系统的设计过程中,选择各个信号的有效级,可很好地减少系统实现的成本。

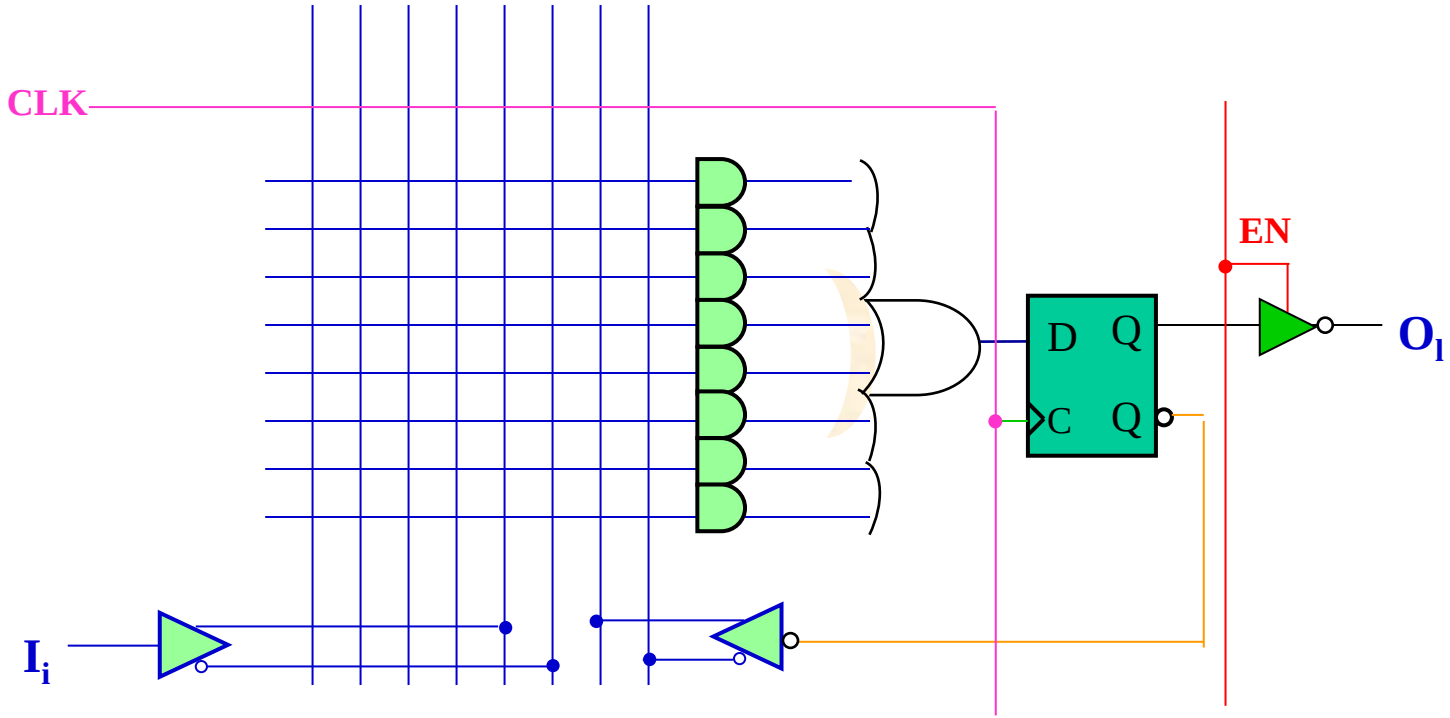
2) 时序 PAL 器件



时序PAL 的基本结构框图

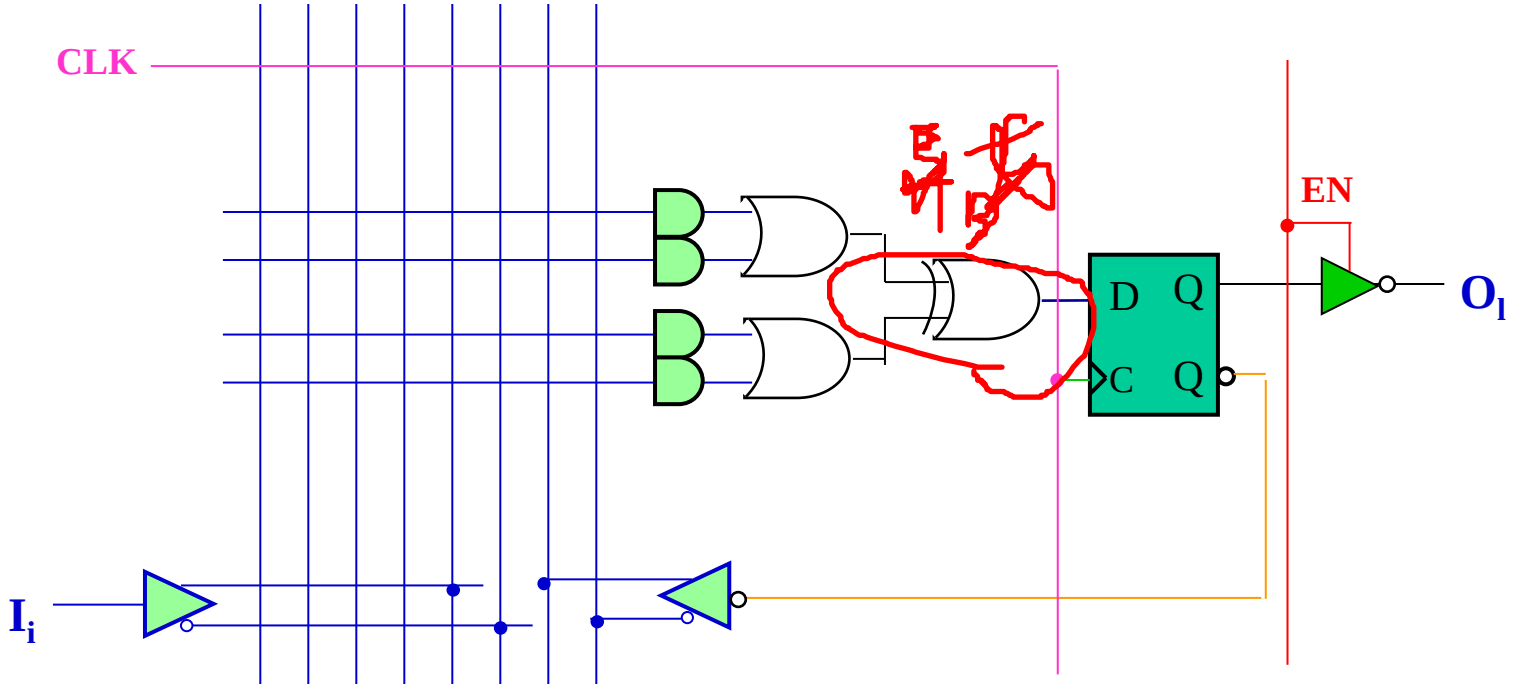
寄存器输出结构（R系列），如图所示：

时序 PAL 器件的部分输出连到D触发器的数据输入端D(输出寄存器)，寄存器受统一的时钟脉冲信号控制。



异或寄存器输出结构（X系列），如图所示：

时序 PAL 器件的部分输出连到D触发器的数据输入端D(输出寄存器)，寄存器受统一的时钟脉冲信号控制。



5. 通用阵列逻辑概述(GAL)*

1. GAL器件的主要特点

1)工艺上的改进

高速电可擦除CMOS(Electrically Erasable Comple-mentary Metal-Oxide Semiconductor (E^2CMOS)

特点:

⑴ 可测试性；（最大优势之一，直接测试各种特性，最适合于样机的研制）

⑵ 低功耗，使集成度更高；（CMOS）

⑶ 速度不低于其他TTL可编程器件

⑷ 可重复编程100次以上

2) 结构上的的改进　通用性→灵活性

(1) 每个输出端增加了一个逻辑输出宏单元(OLMC——Output Logic Macro Cell)：允许设计者以编程的方式确定每一个OLMC的组态和功能，设计者可以在每一个输出端上“随意”实现所要求的功能和结构。

(2) 加密：器件内增加了可被编程的保密位，以防对逻辑的复制。

制作者可以大量生产少数几个型号的GAL器件，从而进一步降低成本；也使得设计者简化了选择器件的过程，减少了数量与体积，降低了成本，提高了可靠性和稳定性。

2.GAL器件的基本结构

与阵列可编程，或阵列固定

- 基本结构

（1）有8个输入缓冲器（第2～9管脚）和8个反馈缓冲器，它们的输出作为与阵列的输入（与阵列的32条列线）。

（2）与阵列有64个乘积项输出，PT0～PT63（标有数字的行线），64行×32列＝2048个可编程单元构成与阵列。

（3）有8个输出逻辑宏单元（第12～19管脚）

（4）1个时钟输入端（第1脚）和1个三态使能输入端OE（第11脚），它们也可作为数据输入端。

（5）5 V电源端（第20脚）和接地端（第10脚），图中未画出

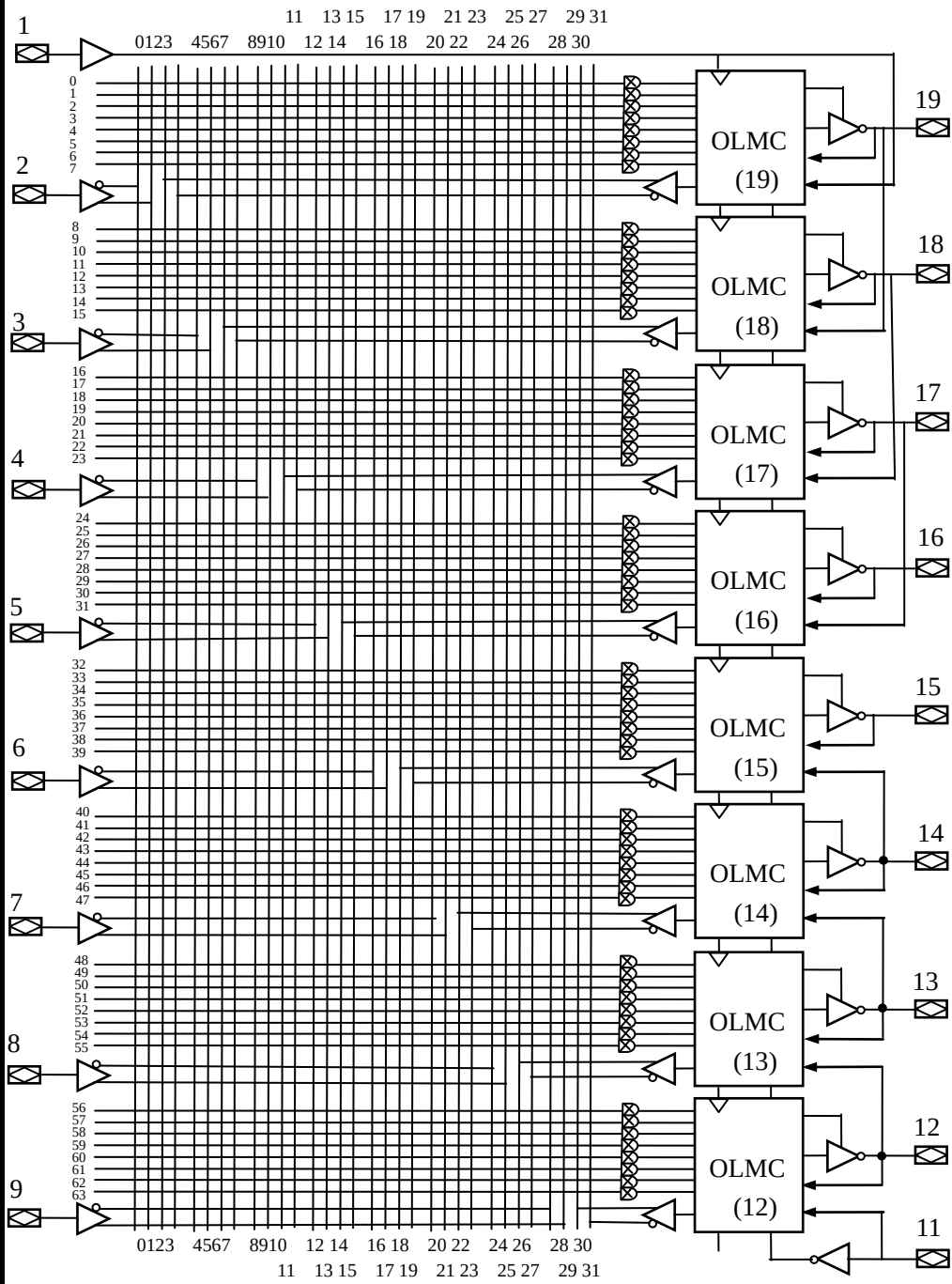


图4.17 GAL16V8

1）输出逻辑宏单元OLMC

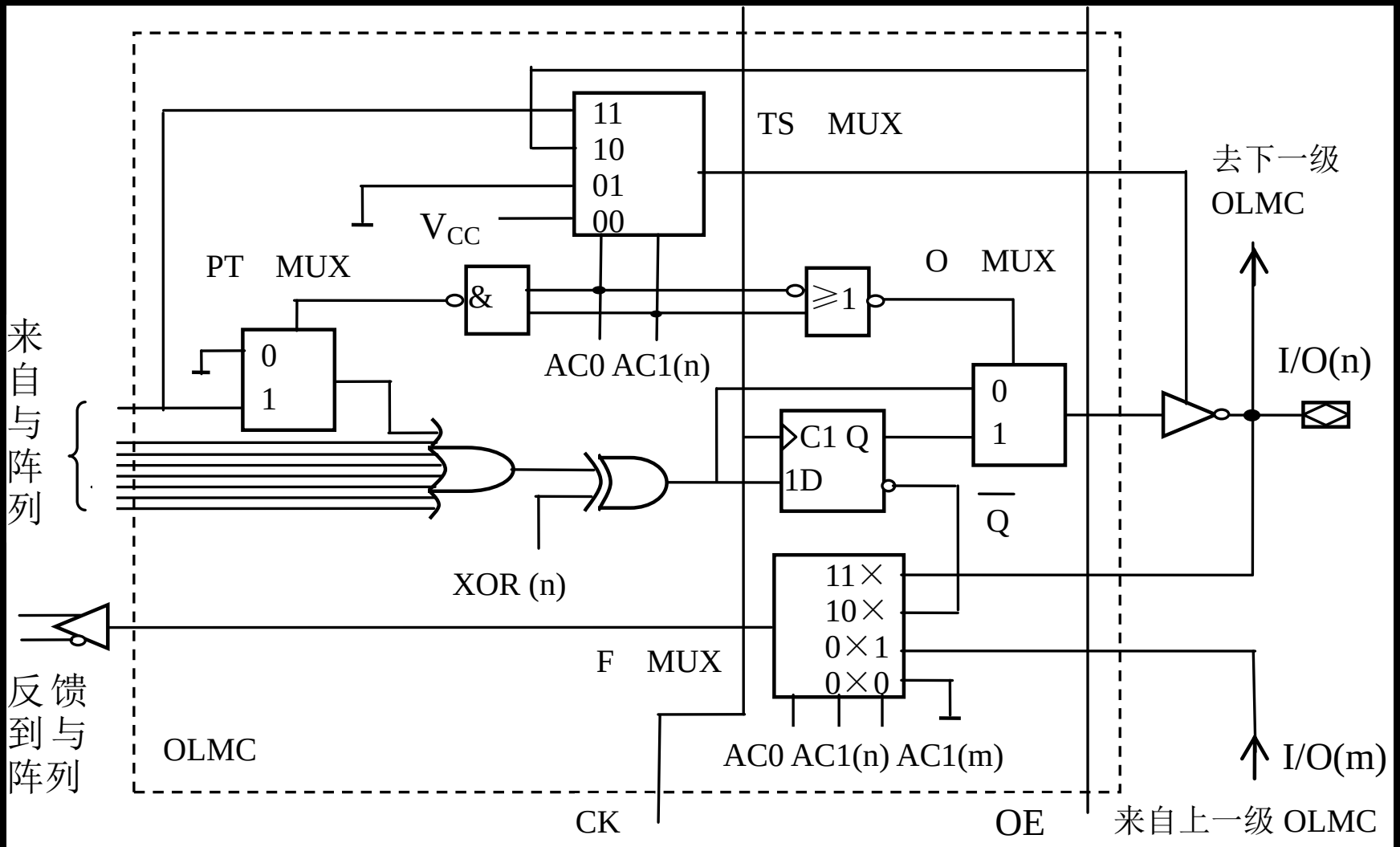


图4.19 GAL输出逻辑宏单元（n）

结构控制字

阅读　GAL16V8的结构控制字配置其片内资源。结构控制字如下图所示。8个OLMC有2个公共的结构控制单元AC0和SYN，每个OLMC还各有2个可编程的结构控制单元AC1(n)和XOR(n)[n=12～19]。PT0～PT63位分别控制与阵列的64个乘积项是否使用。

OLMC各种组态的实现是由开发软件和硬件完成的，对用户是透明的。开发软件将选择与配置控制字的所有位，并自动检查各引线的用法。

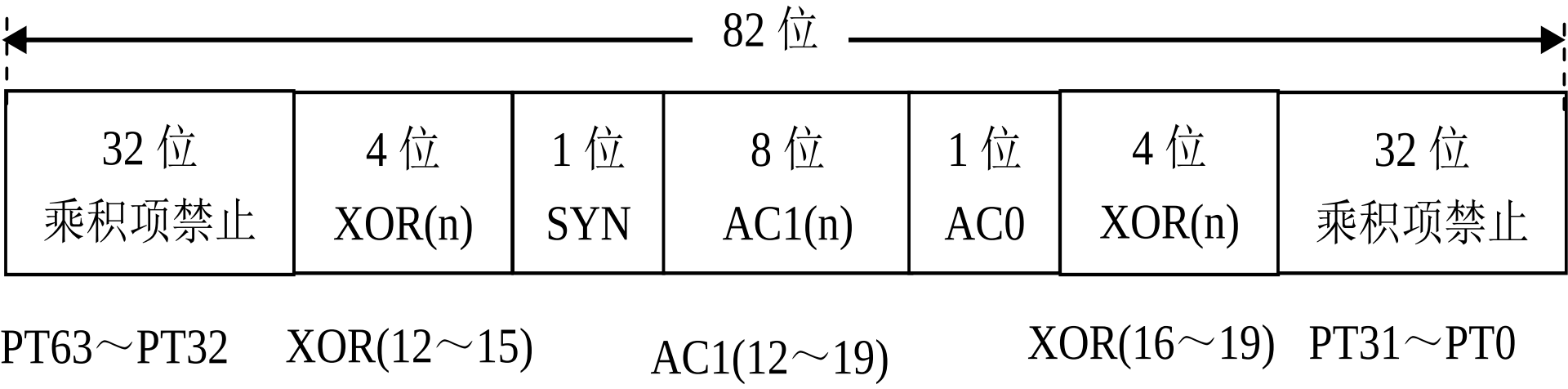
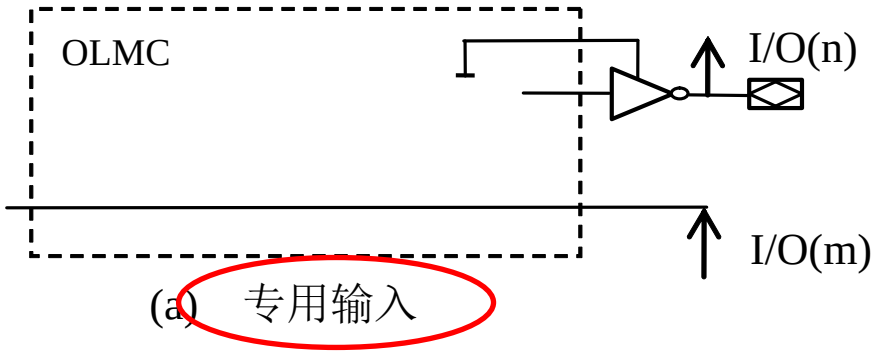


图4.20　GAL16V8结构控制字

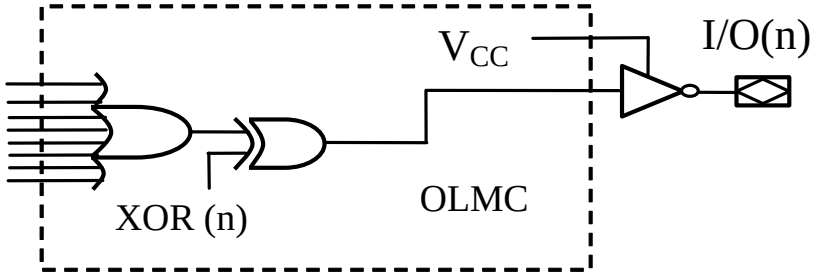
SYN=1，表示器件无寄存器输出；SYN=0，表示器件至少有一个寄存器输出。

OLMC的5种工作模式(1)：专用输入

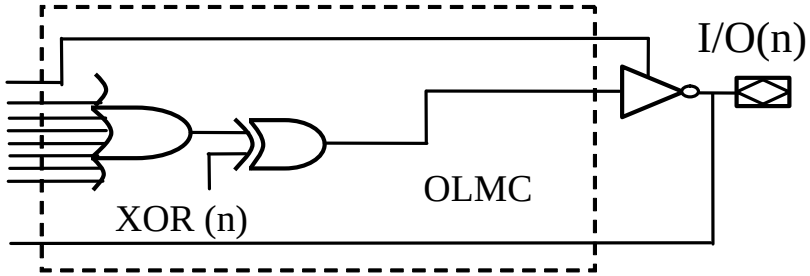
阅读



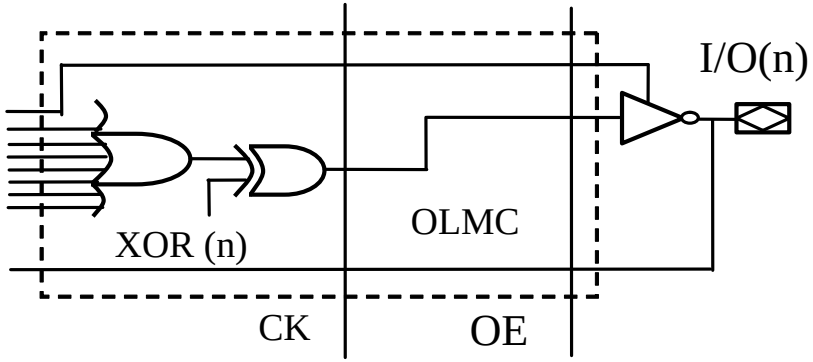
(a) 专用输入



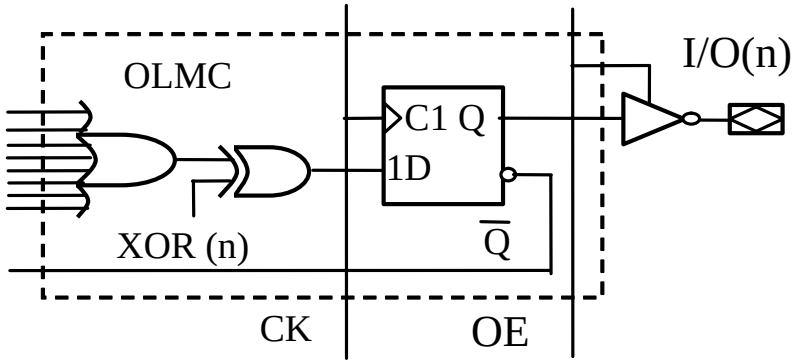
(b) 专用组合输出



(c) 反馈组合输出



(d) 时序组合输出



(e) 寄存器输出

图4.*.1 OLMC的5种工作模式

专用输入：

AC(0)	AC1(n)	AC1(m)	SYN	
0	1	1	1	(1脚CK和11脚OE为数据输入)

阅读

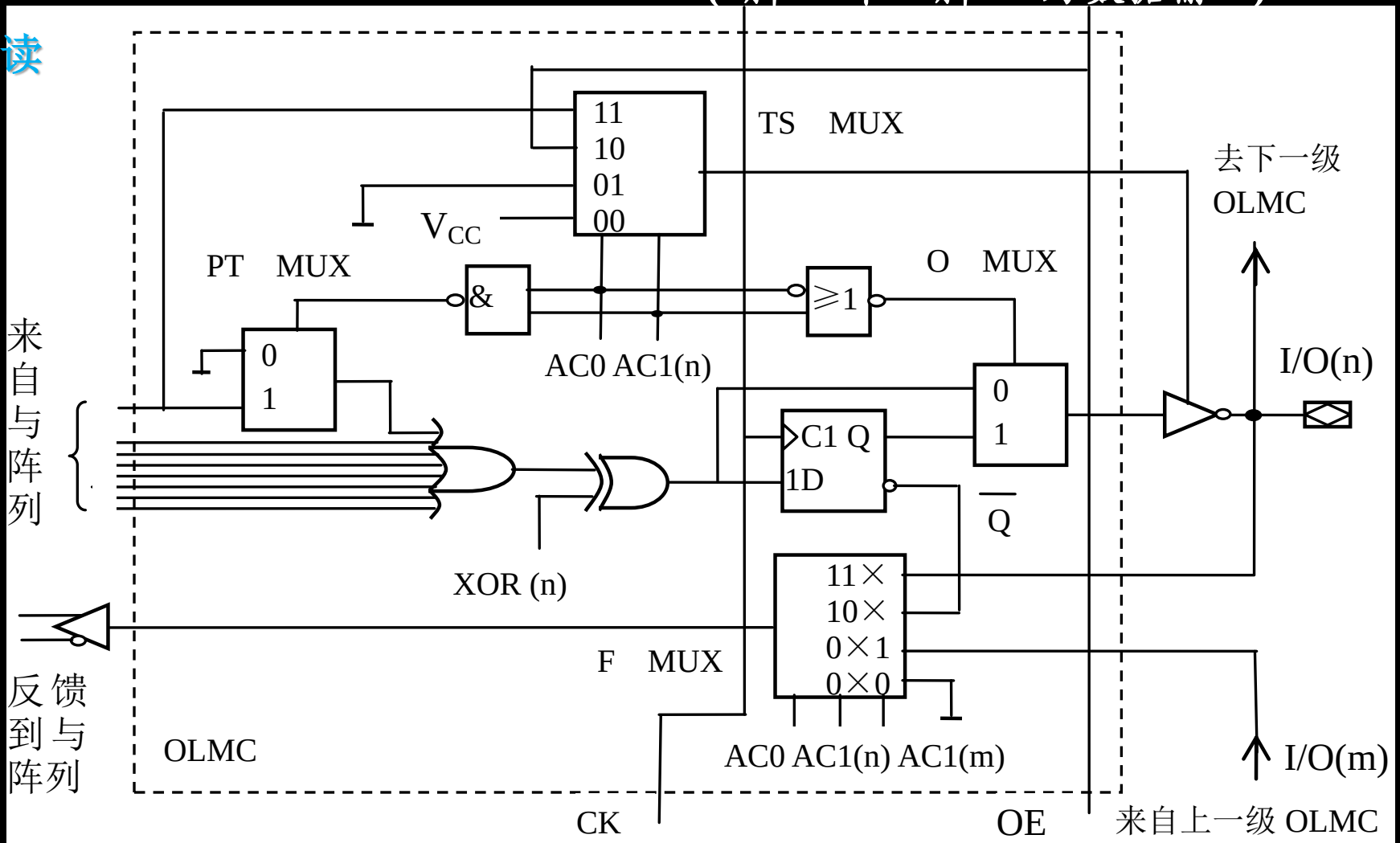
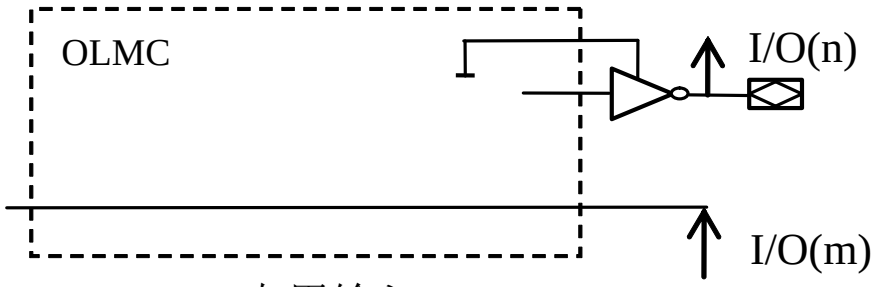


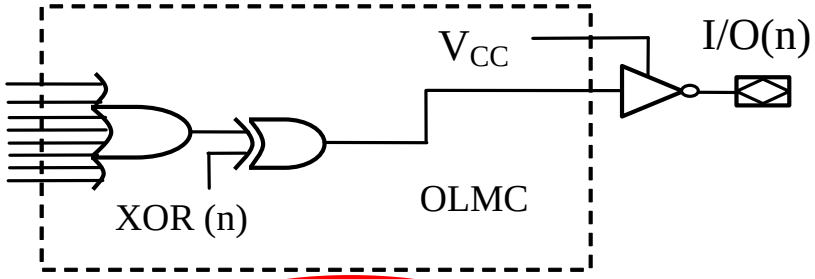
图4.19 GAL输出逻辑宏单元（n）

OLMC的5种工作模式(2)：专用组合输出

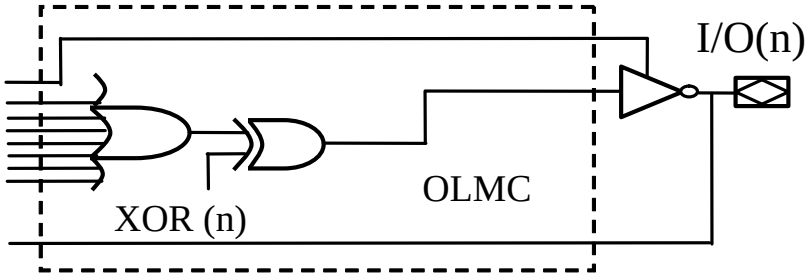
阅读



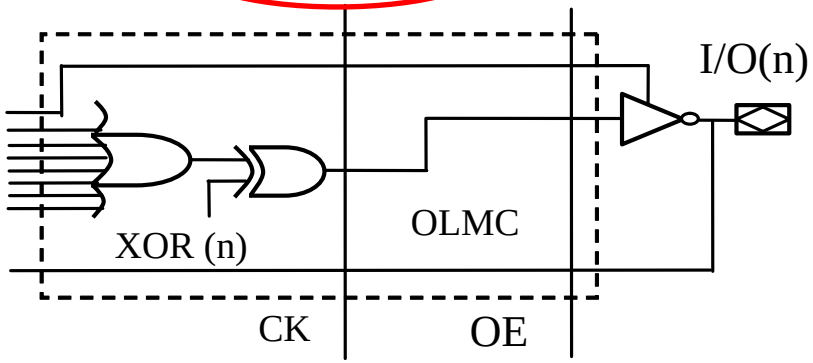
(a) 专用输入



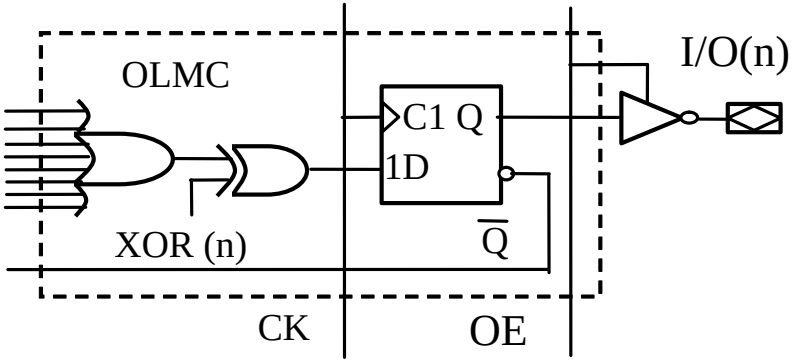
(b) 专用组合输出



(c) 反馈组合输出



(d) 时序组合输出



(e) 寄存器输出

图4.*.2 OLMC的5种工作模式

专用组合输出：

AC(0)	AC1(n)	AC1(m)	XOR(n)	SYN	
0	0	0	*	1	(1脚和11脚为数据输入)

阅读

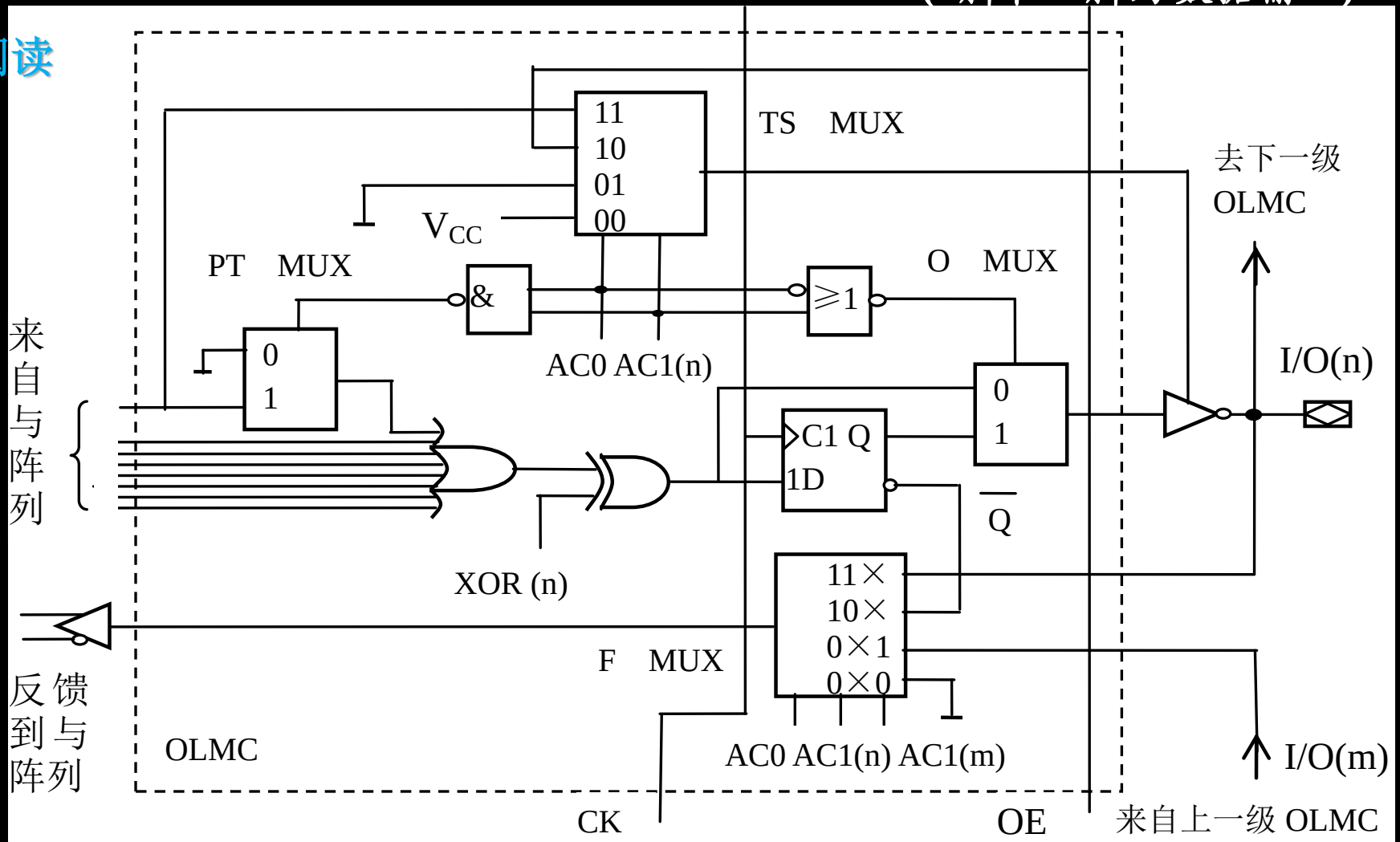
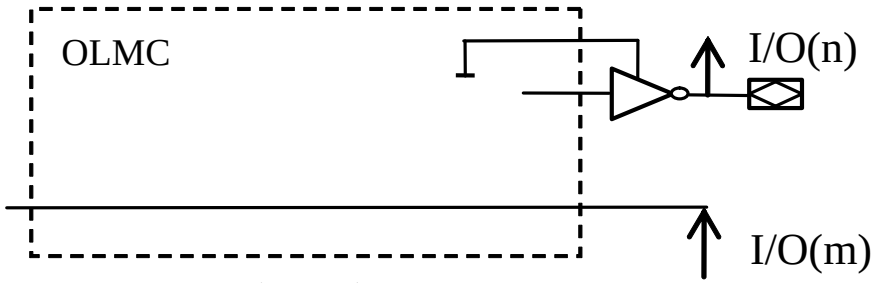


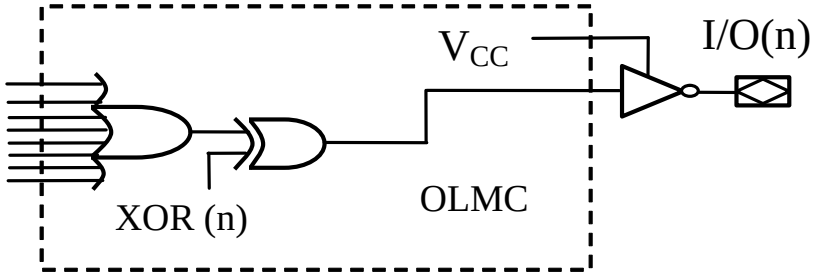
图4.19 GAL输出逻辑宏单元（n）

OLMC的5种工作模式(3)：反馈组合输出

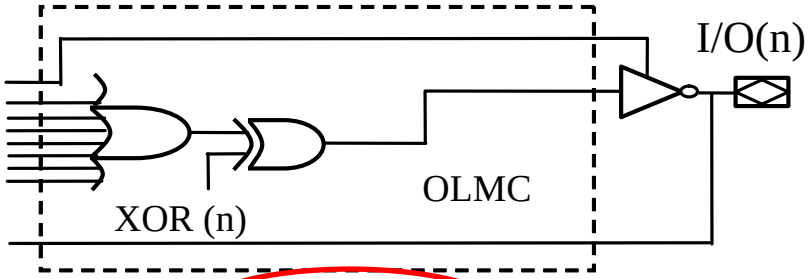
阅读



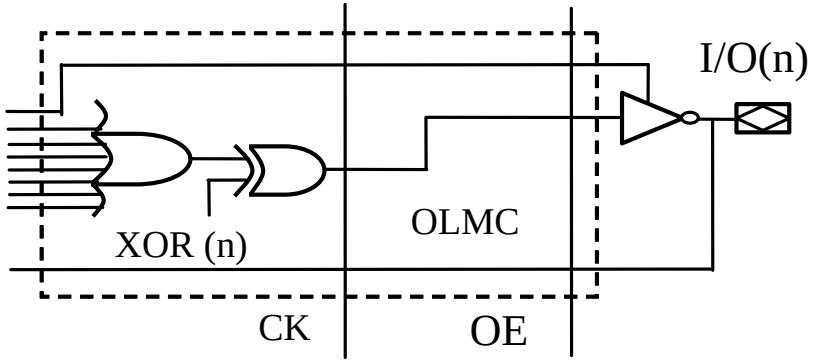
(a)　专用输入



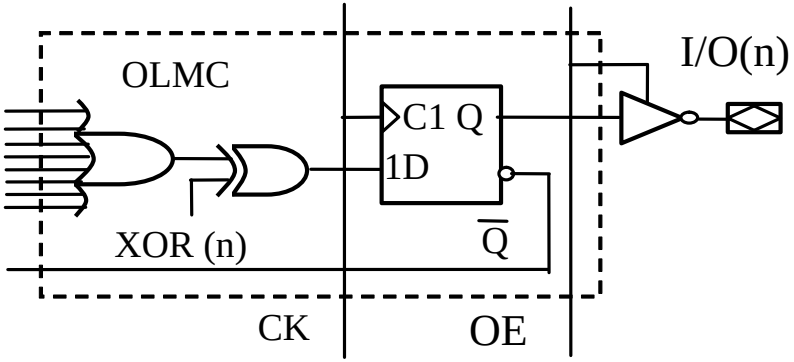
(b)　专用组合输出



(c)　反馈组合输出



(d)　时序组合输出



(e)　寄存器输出

图4.*.3　OLMC的5种工作模式

反馈组合输出：

AC(0)	AC1(n)	AC1(m)	XOR(n)	SYN	
1	1	*	*	1	(1脚和11脚为数据输入)

阅读

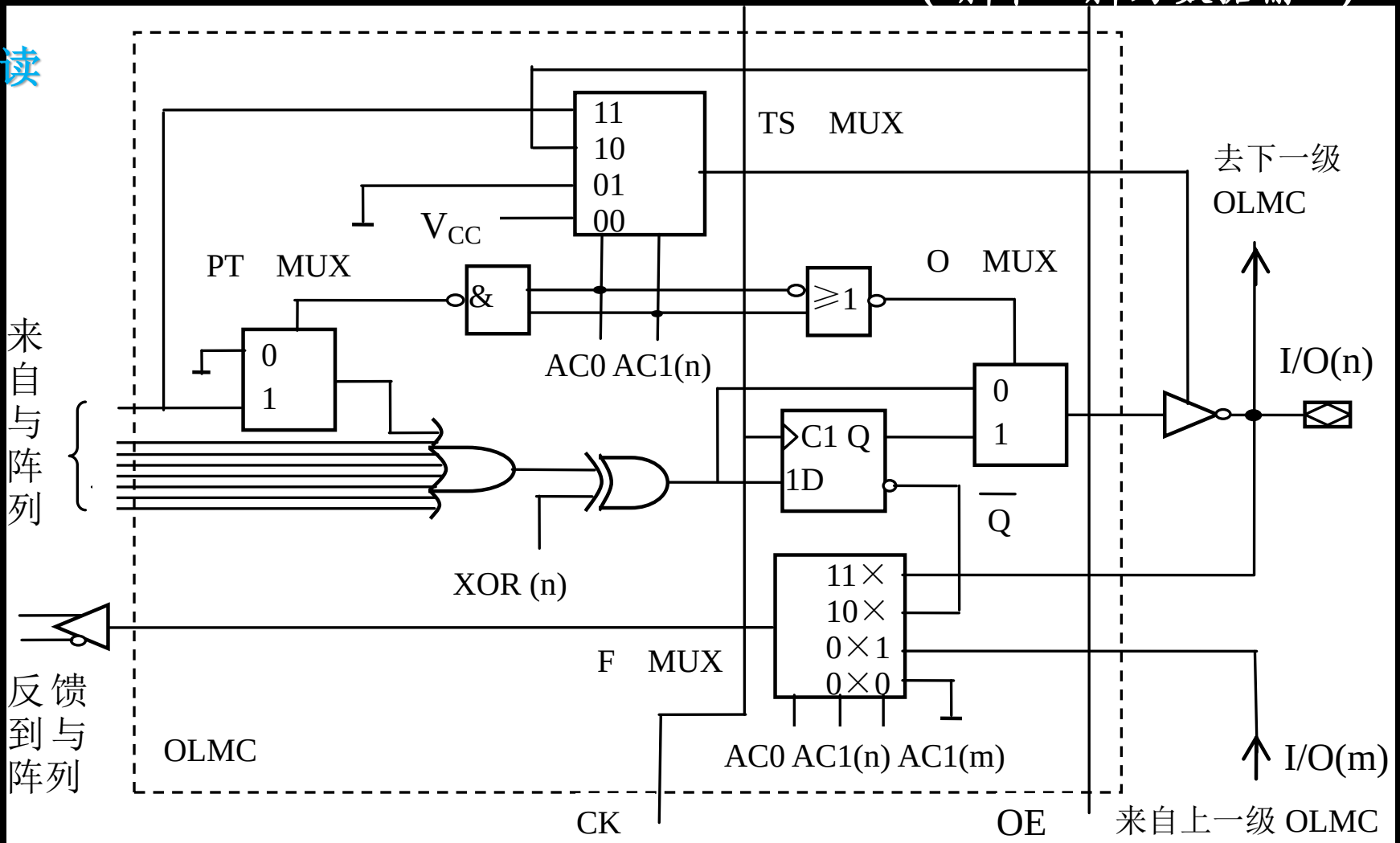
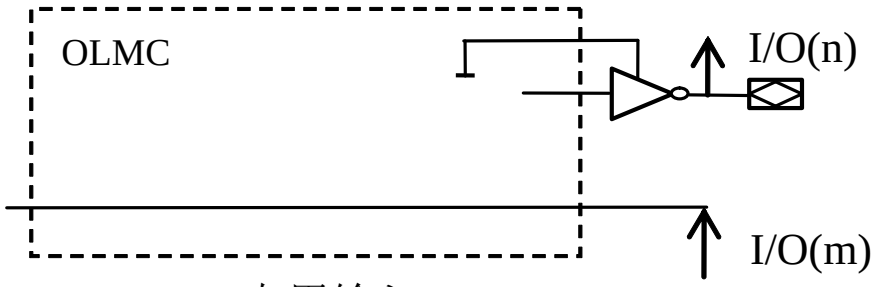


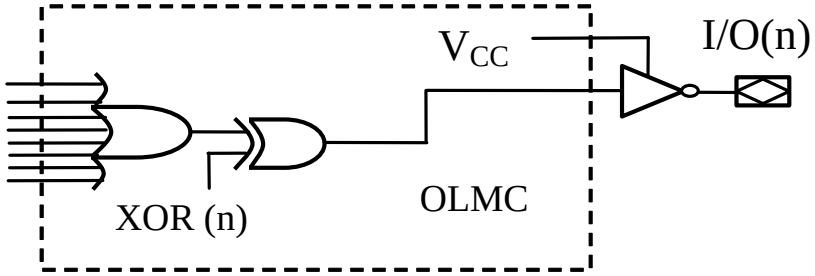
图4.19 GAL输出逻辑宏单元（n）

OLMC的5种工作模式(4)：时序组合输出

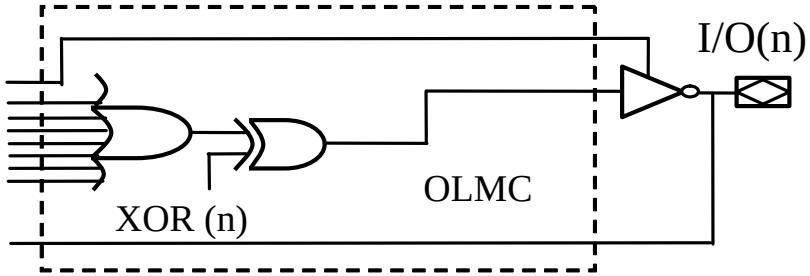
阅读



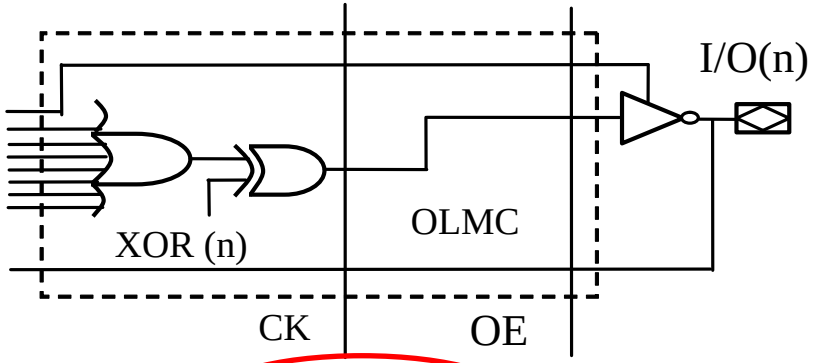
(a) 专用输入



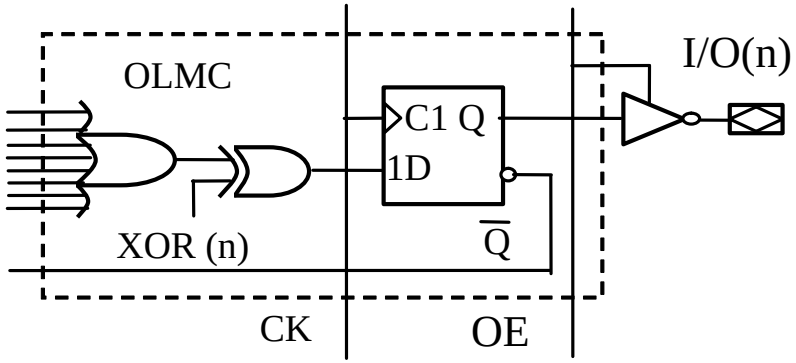
(b) 专用组合输出



(c) 反馈组合输出



(d) 时序组合输出



(e) 寄存器输出

图4.*.4 OLMC的5种工作模式

时序组合输出：

AC(0)	AC1(n)	AC1(m)	XOR(n)	SYN
1	1	*	*	0

(1脚接CLK，11脚接OE至少有一个OLMC为寄存器输出模式)

阅读

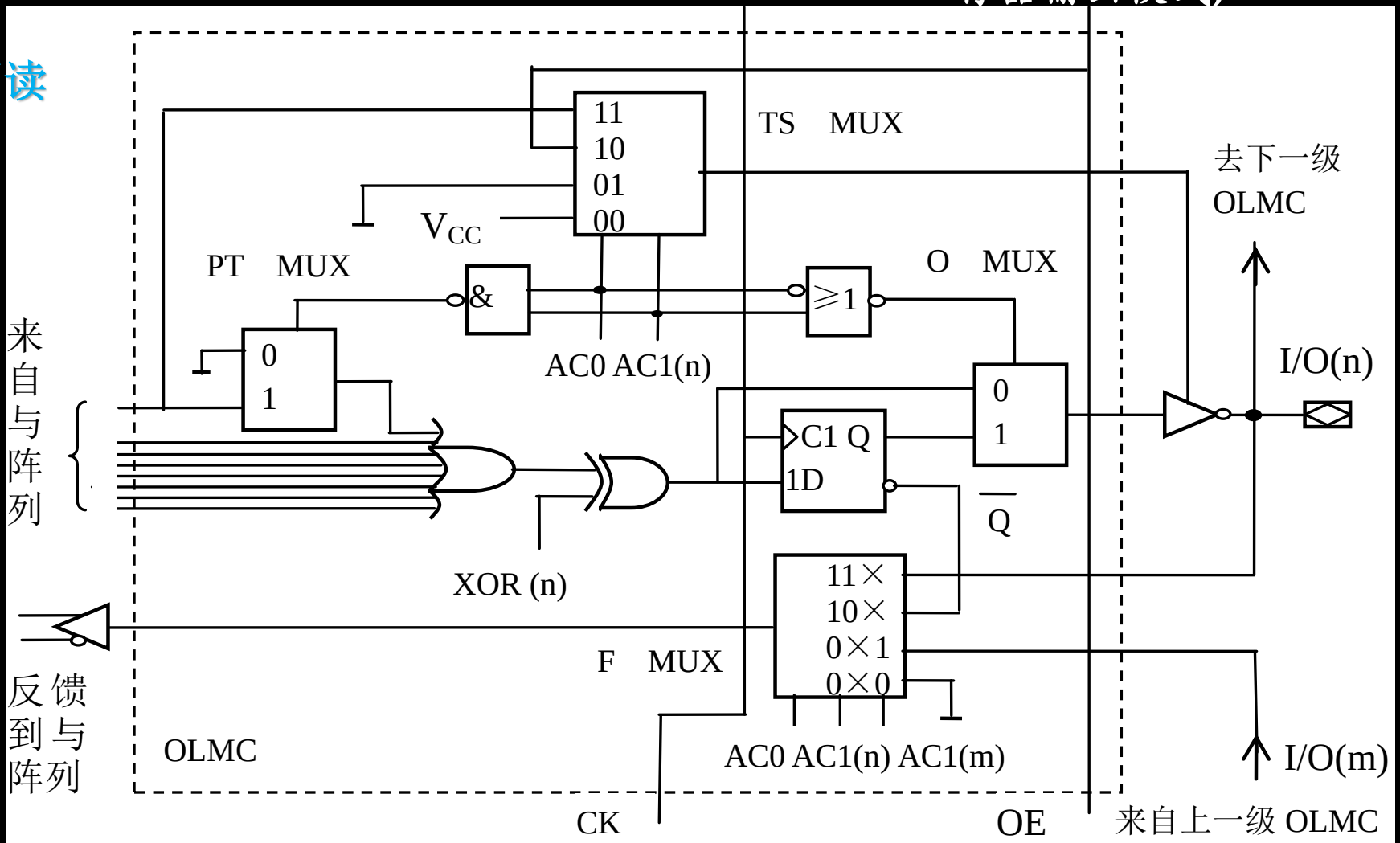
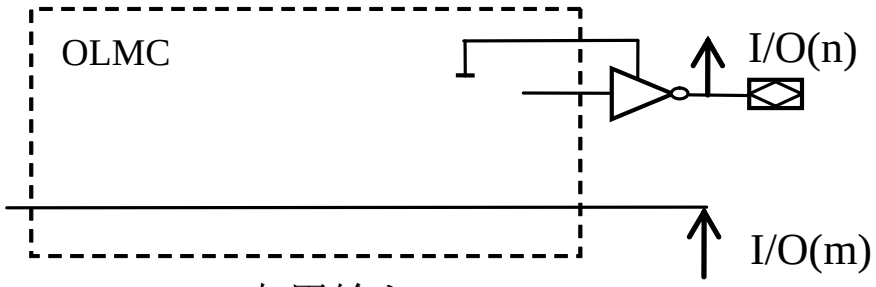


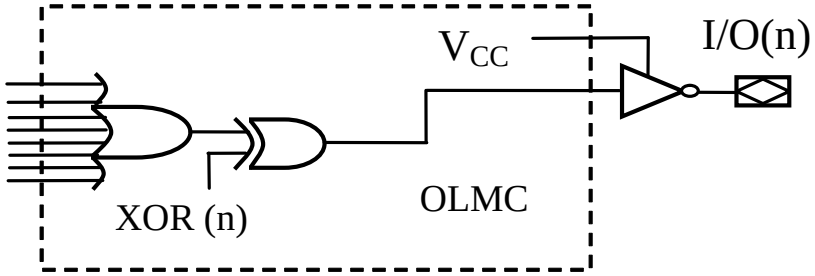
图4.19 GAL输出逻辑宏单元（n）

OLMC的5种工作模式(5)：寄存器输出

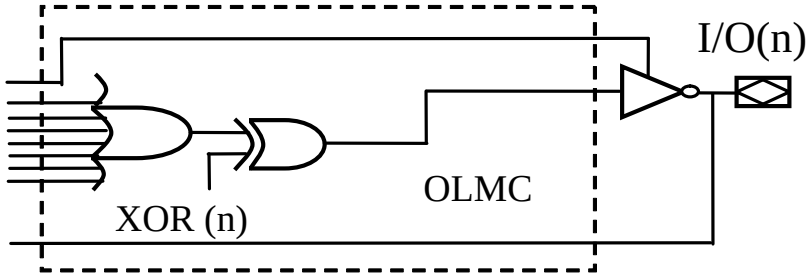
阅读



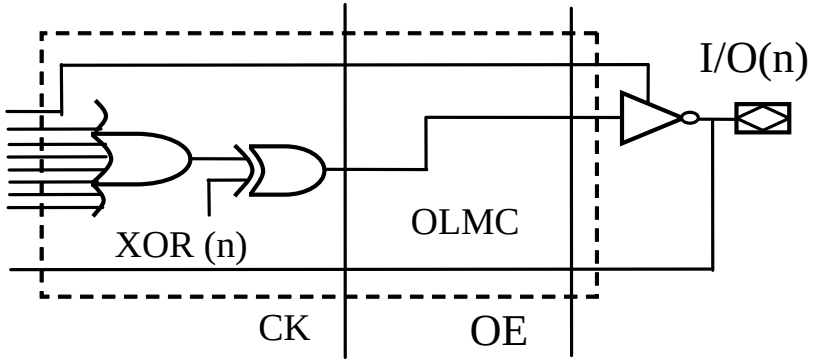
(a) 专用输入



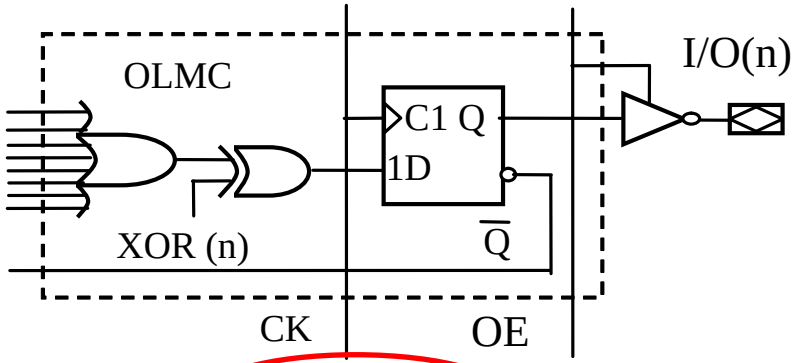
(b) 专用组合输出



(c) 反馈组合输出



(d) 时序组合输出



(e) 寄存器输出

图4.*.5 OLMC的5种工作模式

寄存器输出：

AC(0)	AC1(n)	AC1(m)	XOR(n)	SYN
1	0	*	*	0

(1脚接CLK，11脚接OE，OE控制输出三态缓冲器)

阅读

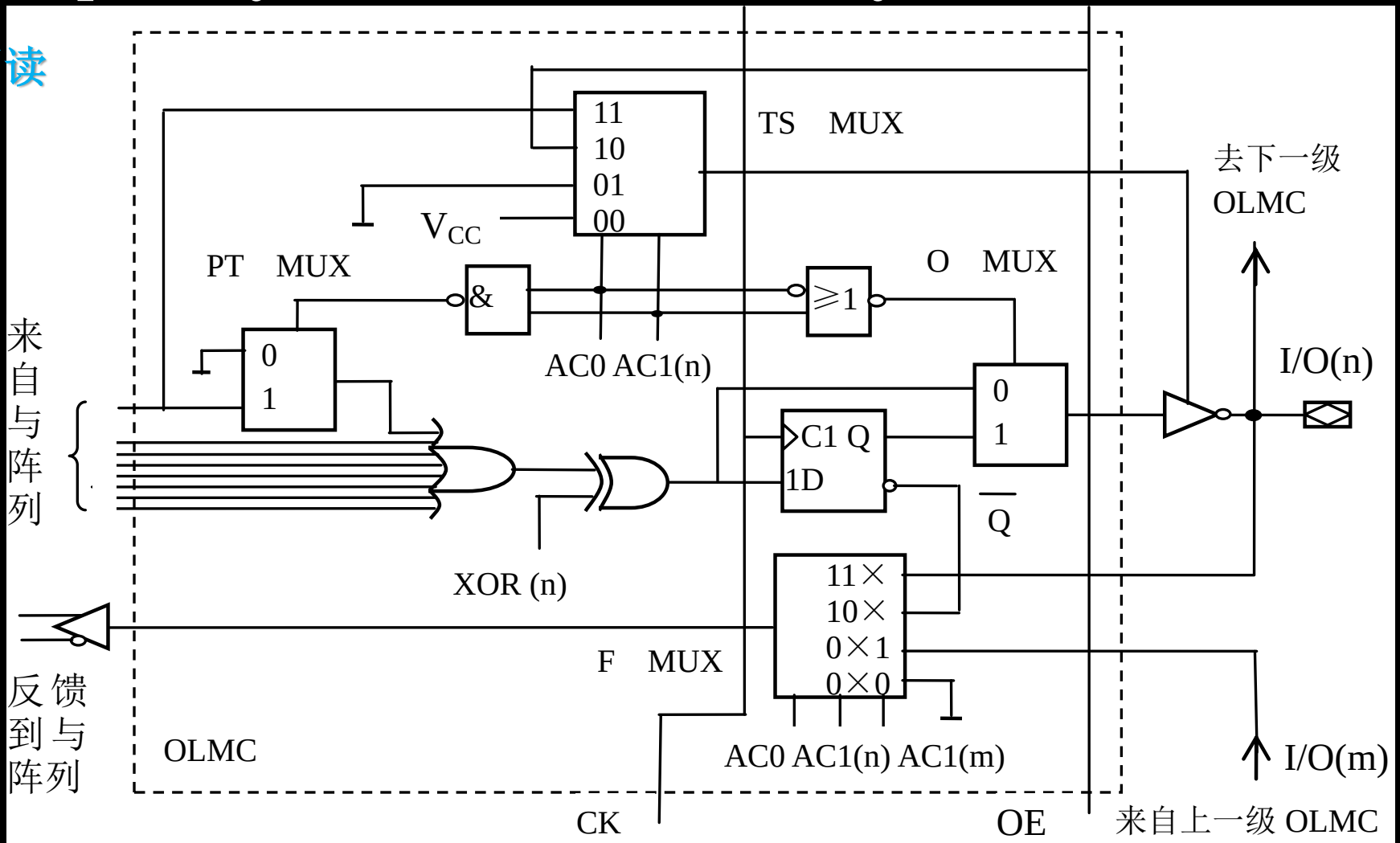


图4.19 GAL输出逻辑宏单元（n）