

第6章 可编程逻辑器件

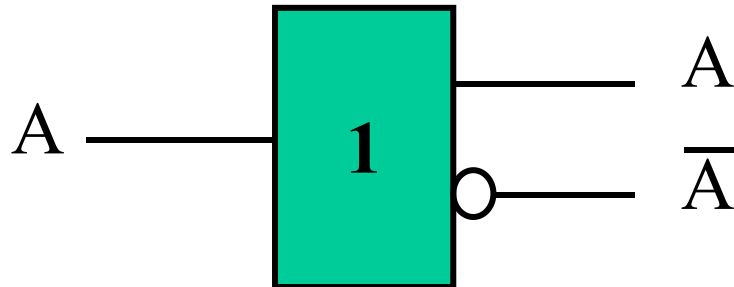
可编程逻辑器件：

PLD: Programmable Logic Device

6.1 PLD概述

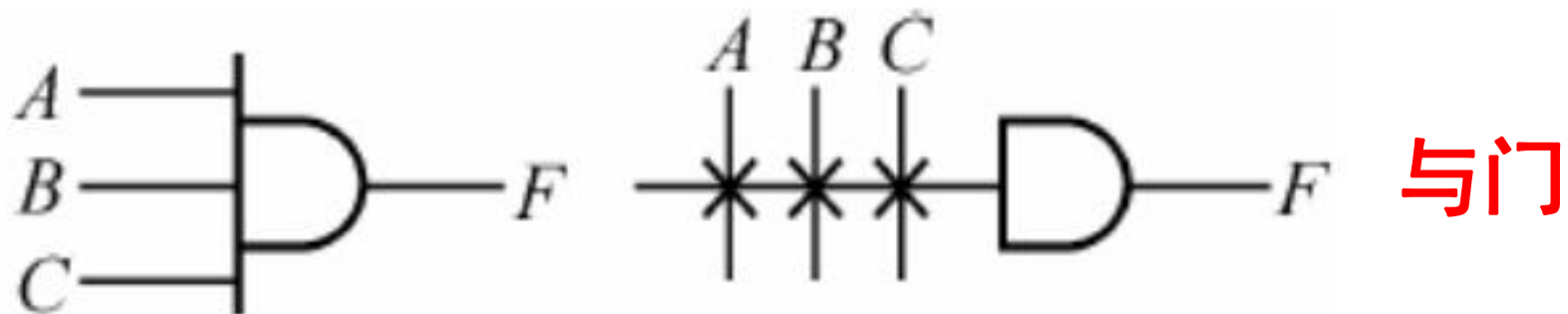
6.1.1 PLD的表示方法

1. 输入缓冲电路

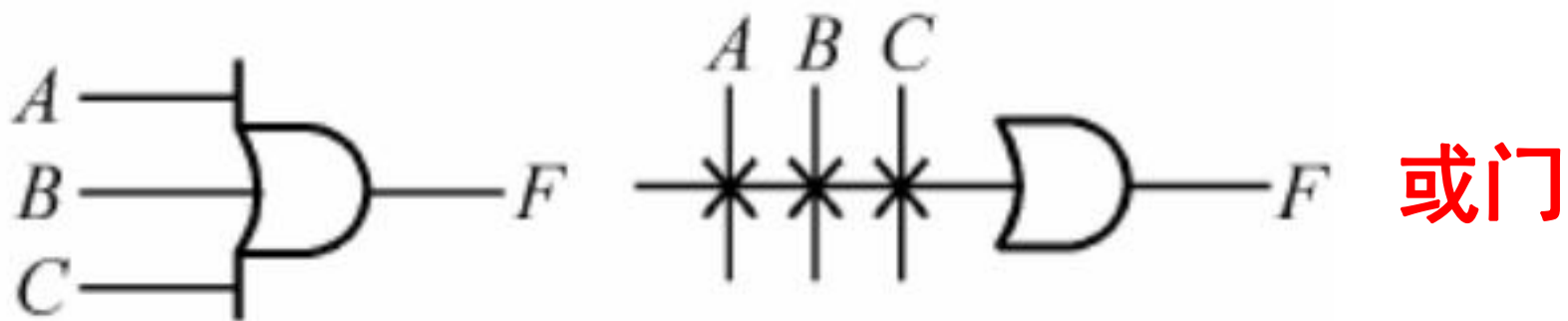


产生原变量和反变量两个互补的信号。

2. 门电路

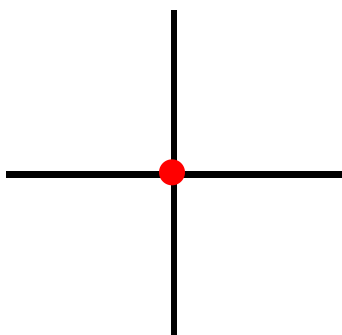


与门

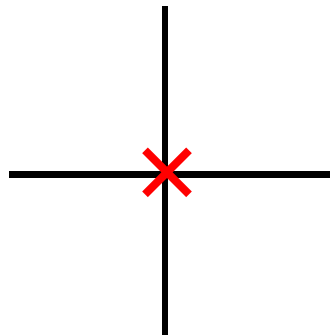


或门

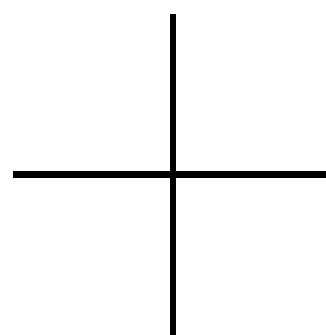
3. 导线连接



固定连接

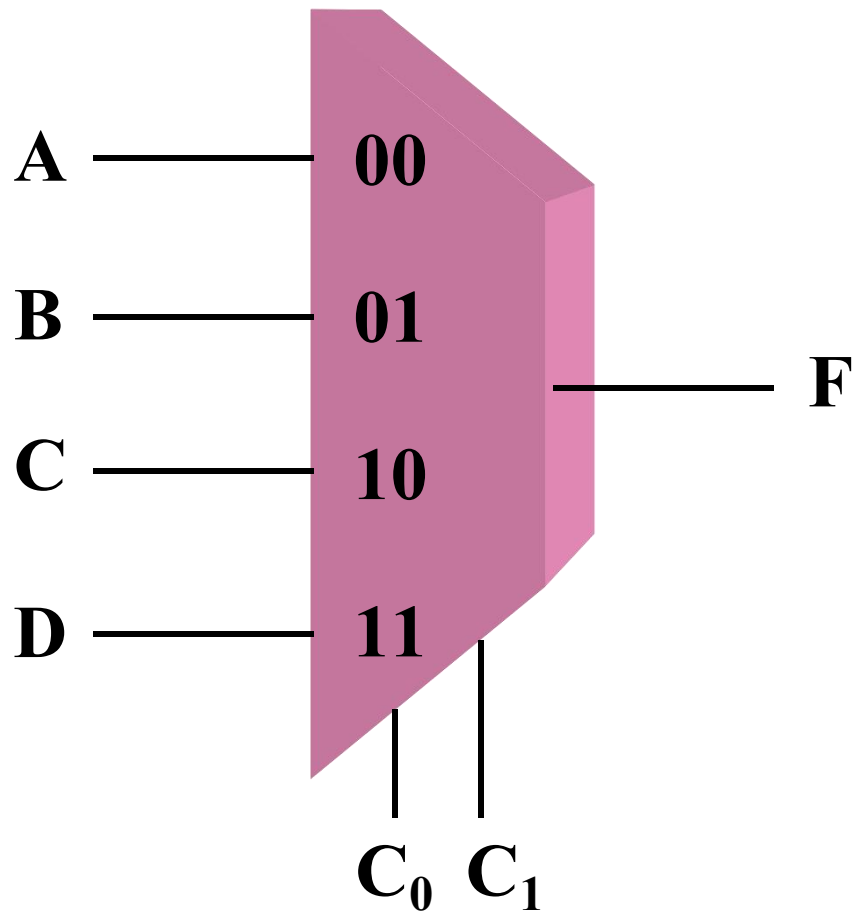


编程连接



断开

4、多路选择器

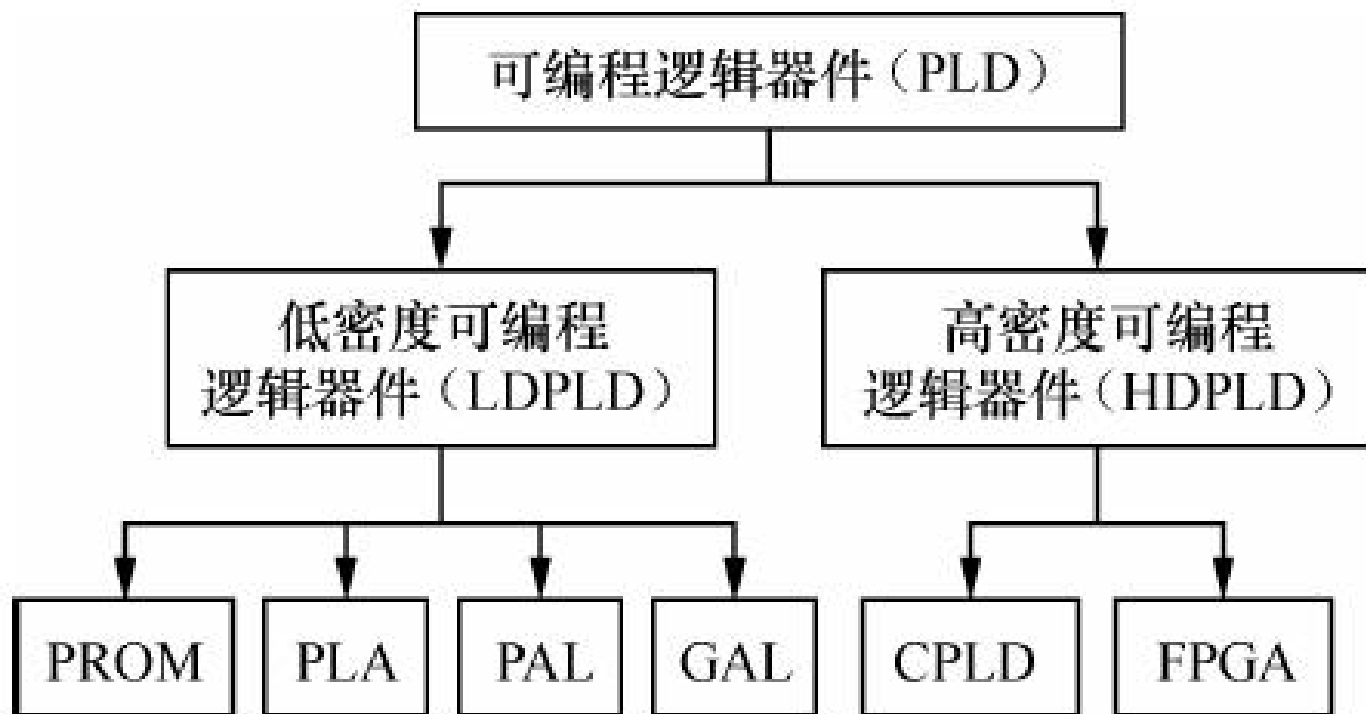


6.1.3 PLD的制造工艺

1. 基于掩膜技术的PLD
 2. 基于熔丝或反熔丝技术的PLD
 3. 紫外线可擦除的PLD
 4. 电可擦除的PLD
 5. 基于Flash技术的PLD
 6. 基于SRAM技术的PLD
- SRAM: 静态随机存取存储器

6.1.4 PLD的分类

1. 按集成度分类



(1) 低密度可编程逻辑器件 (LDPLD:Low-Density PLD)

① PROM (Programmable ROM)

20世纪70年代初。

② PLA(Programmable Logic Array)

20世纪70年代初。

③ PAL(Programmable Array Logic)

20世纪70年代末。

④ GAL(Generic Array Logic)

20世纪80年代初。

(2)高密度可编程逻辑器件(HDPLD:High-Density PLD)

① CPLD (Complex PLD)

复杂可编程逻辑器件。

20世纪 80年代中。

② FPGA(Field Programmable Gate Array)

现场可编程门阵列 。

20世纪 80年代中。

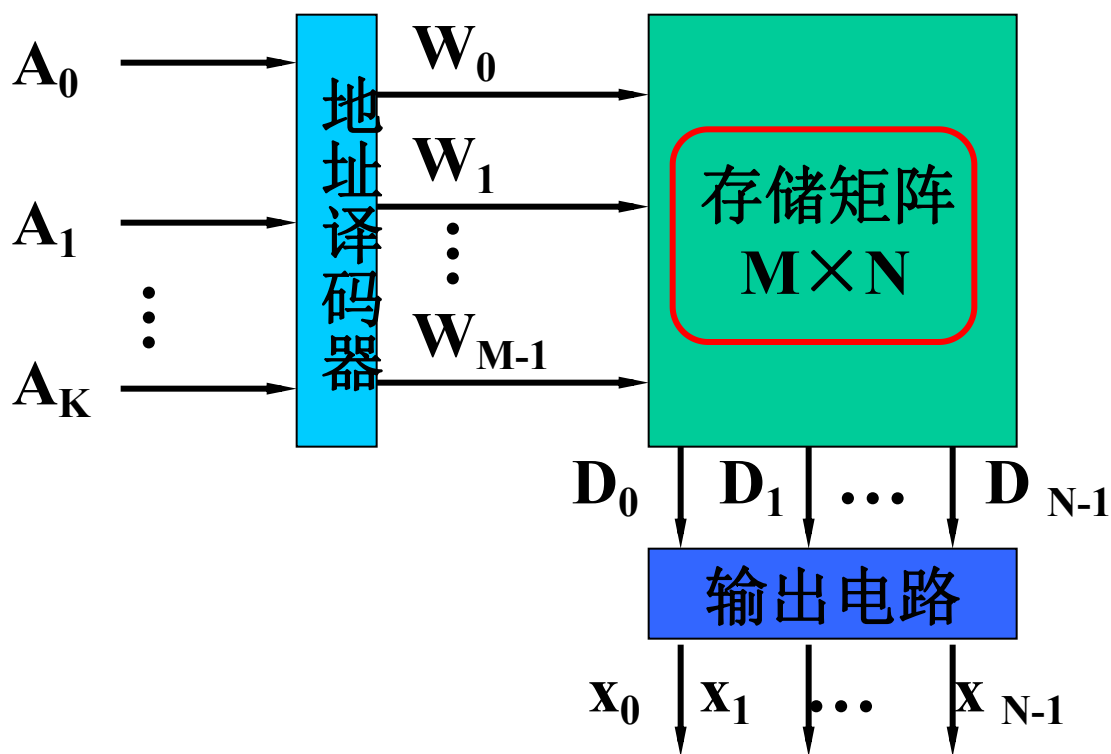
6.2 可编程只读存储器（PROM）

ROM: Read Only Memory

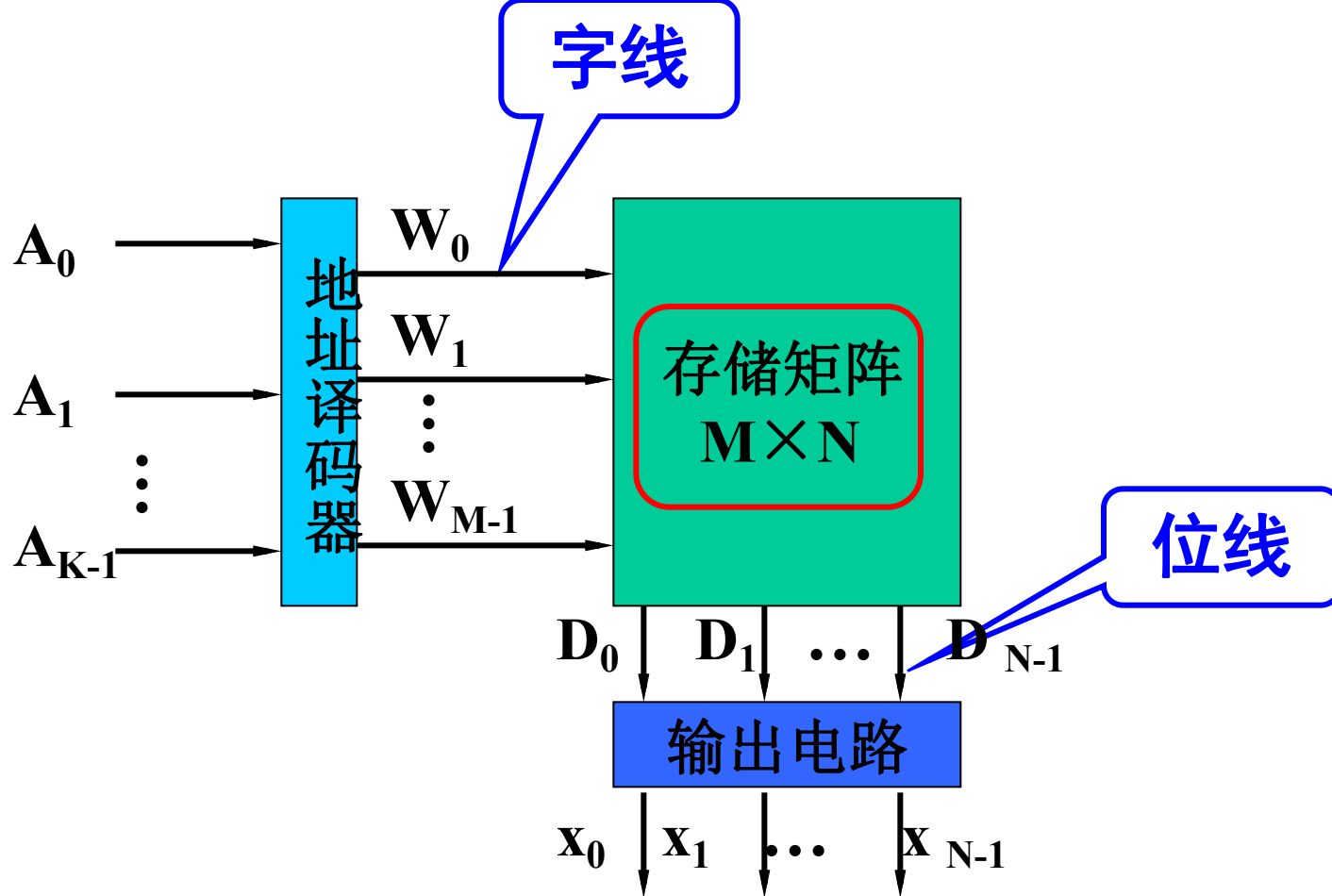
只能读出存储器中已有的信息；
切断电源后，信息也不会丢失。

6.2.1 PROM的结构和功能

ROM的结构框图：



ROM的结构由地址译码器、存储单元矩阵、输出电路组成。



存储器以**字**为单位存取，每字包含若干**位**。

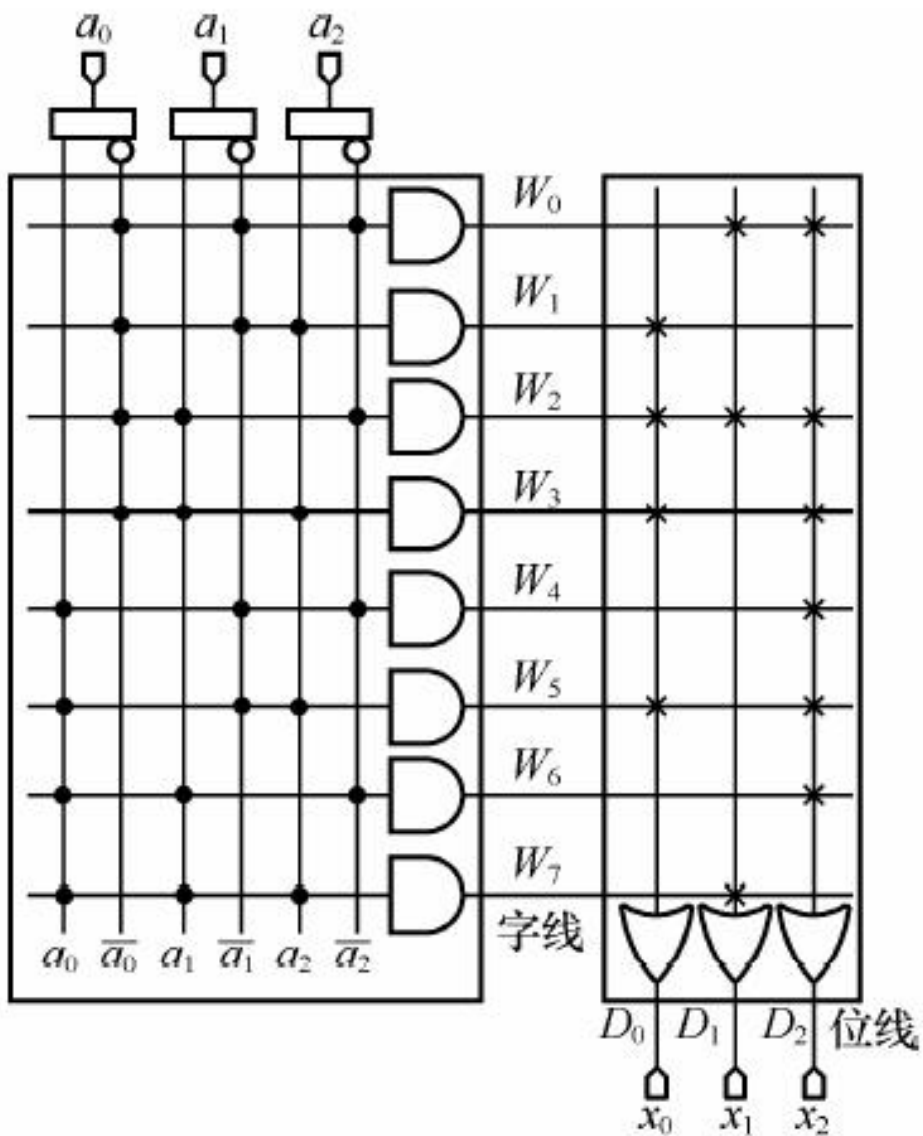
存储容量：写成“**字数**×**位数**”的形式。即：“ **$M \times N$** ”

每个字分配一个**地址**，因此内部有地址译码器。

补充：有容量为 256×4 , $64\text{K} \times 1$, $1\text{M} \times 8$, $128\text{K} \times 16$ 位的ROM，试分别回答：

- 1) 这些ROM有多少个基本存储单元？
- 2) 这些ROM每次访问几个基本存储单元？
- 3) 这些ROM有多少个地址线？

- (1) 分别有1024个, 1024×64 个, $1\text{M} \times 8$, $128\text{K} \times 16$ 个。
- (2) 分别为4个, 1个, 8个, 16个
- (3) 分别有8, 16, 20, 17条地址线



8×3位ROM的简化逻辑结构

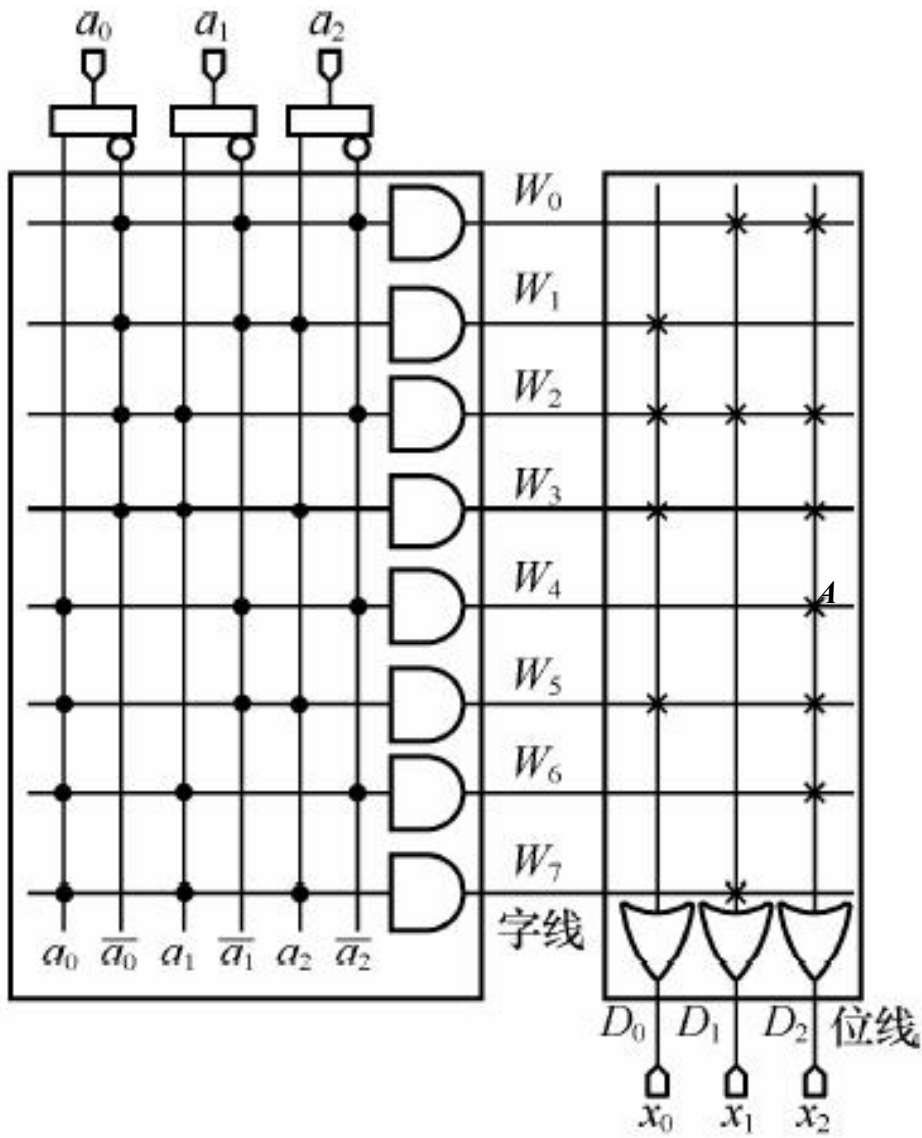
- 地址译码器的3位地址输入端 (a_0 、 a_1 、 a_2) 能指定8个不同的地址，地址译码器将这8个地址分别译成 $W_0 \sim W_7$ 8个高电平输出信号。
- 在读数据时，只要输入指定的地址码，则指定地址内各存储单元所存数据经驱动器输出在数据线上。存储矩阵中字线 W 和位线 D 的每个交叉点都是一个存储单元，若交叉点上接有二极管，则该交叉点的对应位存储“1”，否则存储“0”。

表 6.2.1

图 6.2.1 中 ROM 的地址输入与数据输出的对应关系

输入地址 ($a_0a_1a_2$)	000	001	010	011	100	101	110	111
选中字线	W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7
数据输出 ($x_0x_1x_2$)	011	100	111	101	001	101	001	010

- ROM属于组合逻辑电路，即给定一定输入（地址），存储器相应地给出一种输出（存储的字）。
- 译码器：将输入地址变量译成地址码，完成的是逻辑“与”功能。
- 存储矩阵和输出电路部分所实现的逻辑功能是逻辑“或”运算。



$$W_0 = \overline{A_0} \overline{A_1} \overline{A_2}$$

$$W_1 = \overline{A_0} \overline{A_1} A_2$$

$$W_2 = \overline{A_0} A_1 \overline{A_2}$$

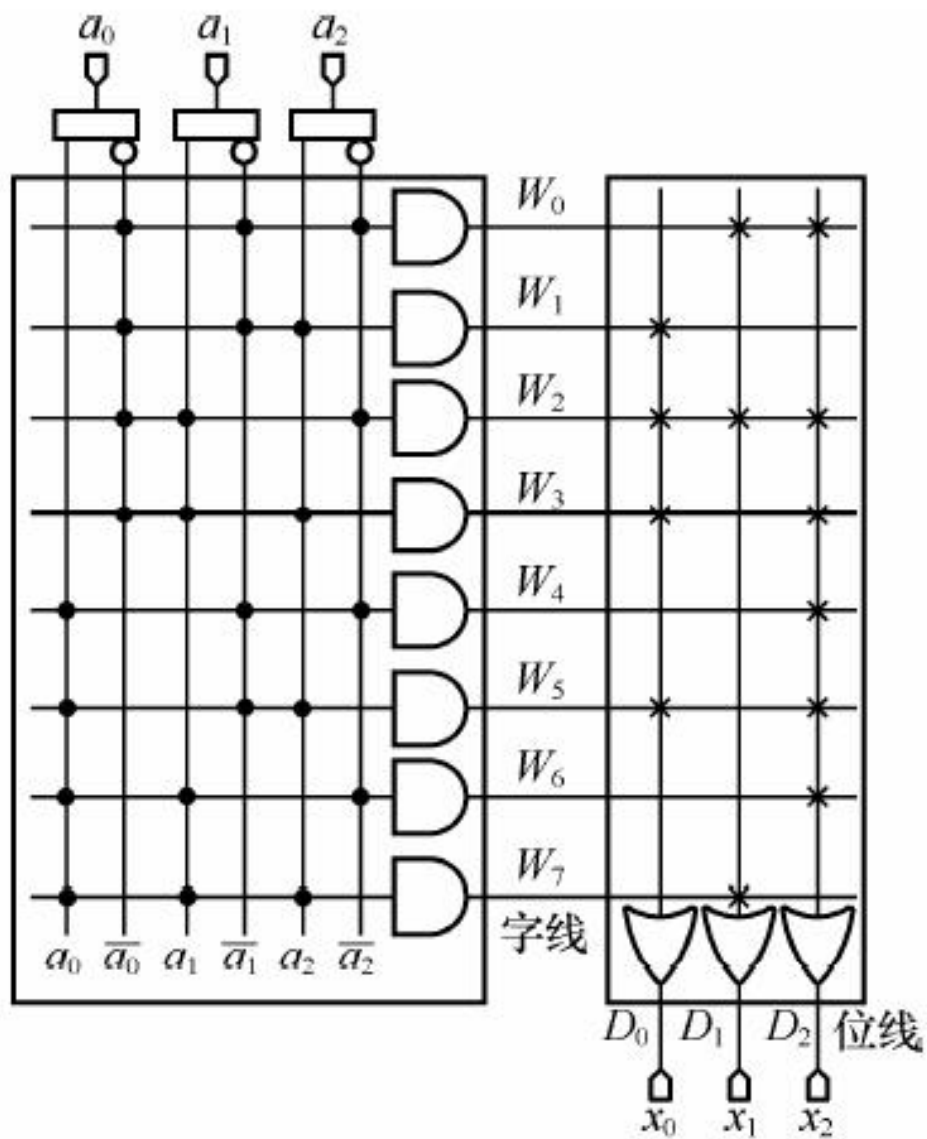
$$W_3 = \overline{A_0} A_1 A_2$$

$$W_4 = A_0 \overline{A_1} \overline{A_2}$$

$$W_5 = A_0 \overline{A_1} A_2$$

$$W_6 = A_0 A_1 \overline{A_2}$$

$$W_7 = A_0 A_1 A_2$$



$$D_0 = W_1 + W_2 + W_3 + W_5$$

$$D_1 = W_0 + W_2 + W_7$$

$$D_2 = W_0 + W_2 + W_3 + W_4 + W_5 + W_6$$

**PROM阵列中，与阵列固定，或阵列可变。
因此可根据需要实现任意组合逻辑电路。**

6.2.2 ROM的应用

1. 作为存储器
2. 实现组合逻辑函数
3. 存储容量的扩展

实现组合逻辑函数举例：

例1： 试用ROM实现如下组合逻辑函数。

$$F_1 = AB + \overline{AC}$$

$$F_2 = AB + \overline{BC}$$

解：

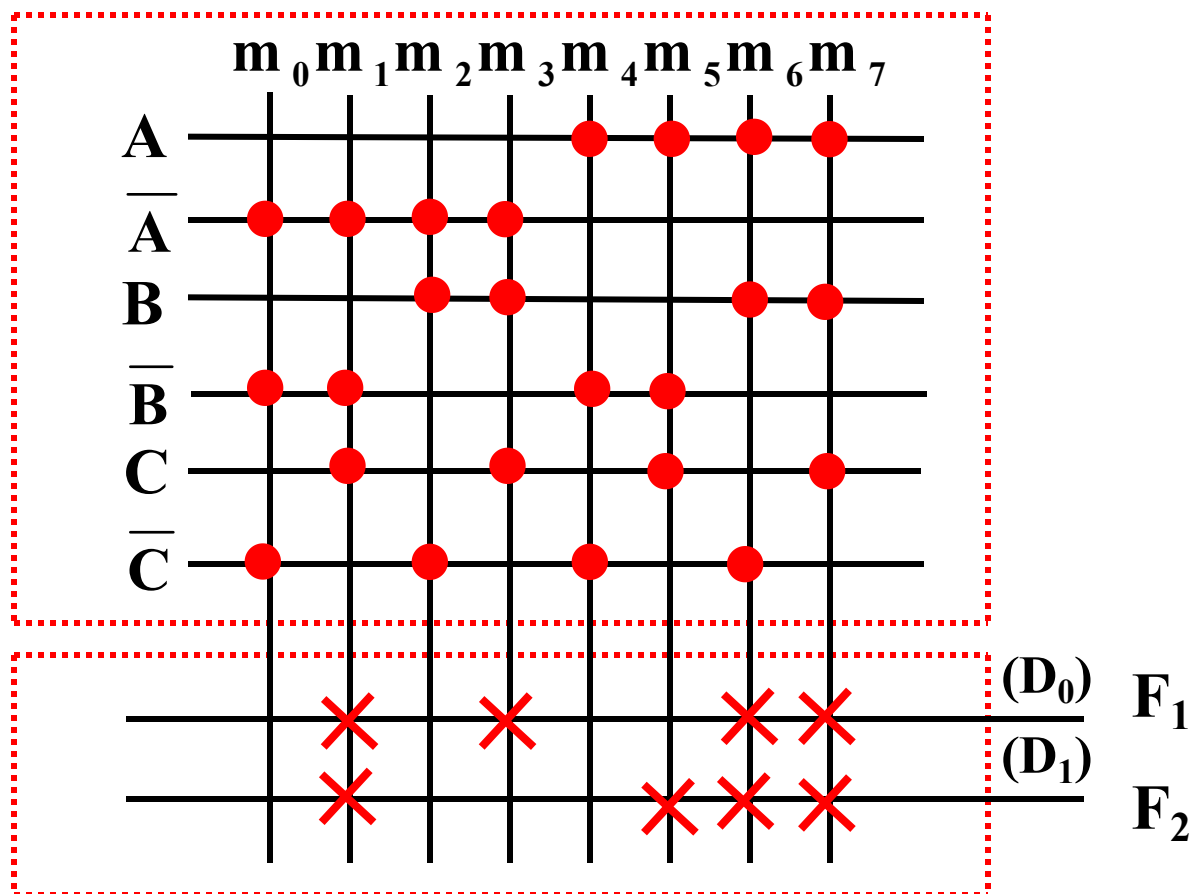
化成由最小项组成的标准“与-或”式,即

$$F_1 = ABC + AB\overline{C} + \overline{A}BC + \overline{A}B\overline{C} = m_7 + m_6 + m_1 + m_3$$

$$F_2 = ABC + AB\overline{C} + \overline{A}BC + \overline{A}B\overline{C} = m_7 + m_6 + m_5 + m_1$$

采用有3位地址码、2位数据输出的8字×2位ROM。

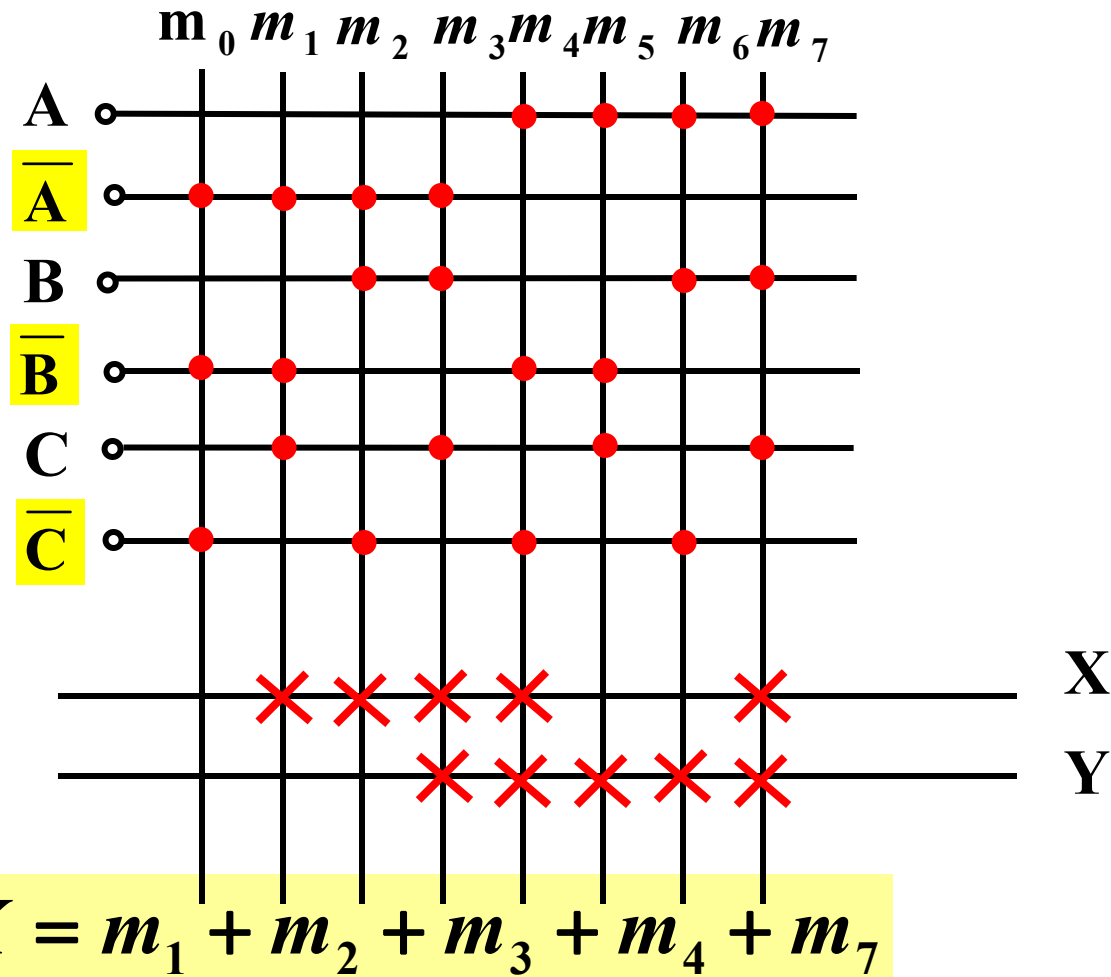
ROM 阵列如图所示：



$$F_1 = ABC + A\bar{B}\bar{C} + \bar{A}BC + \bar{A}\bar{B}C = m_7 + m_6 + m_1 + m_3$$

$$F_2 = ABC + A\bar{B}\bar{C} + \bar{A}BC + \bar{A}\bar{B}C = m_7 + m_6 + m_5 + m_1$$

例2：根据ROM存储矩阵连线图写出输出逻辑表达式，能化简则化简至最简与或式。



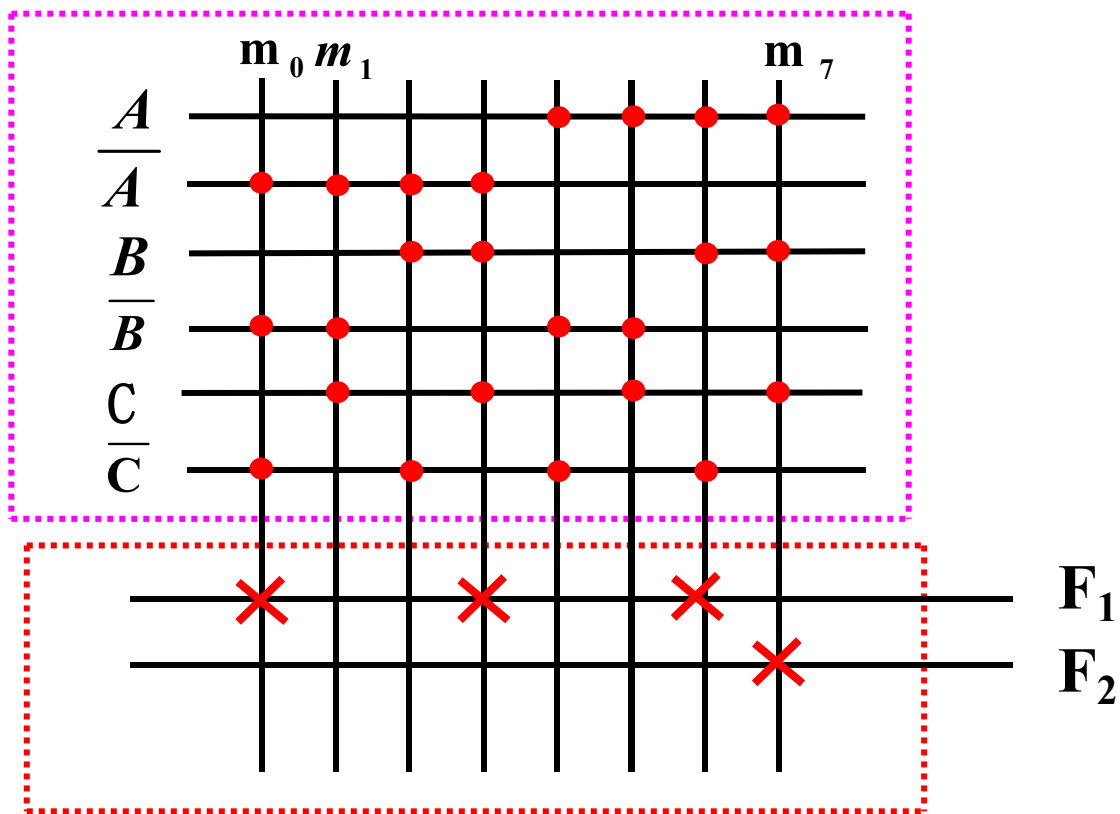
$$X = m_1 + m_2 + m_3 + m_4 + m_7$$

$$Y = m_3 + m_4 + m_5 + m_6 + m_7$$

卡诺图化简略。

例3: 用ROM设计一下电路：设ABC为三位二进制数，
 (1)是否能被3整除，若能被3整除，则输出 $F_1=1$ 。
 (2)是否大于6，若大于6，则输出 $F_2=1$ 。

A	B	C	F_1	F_2
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	0	1



随机存储器(RAM):

RAM: Random Access Memory

能在存储器中任意指定的存储单元随时写入或读出信息；断电时，原写入的信息会随时消失。

根据存储单元的工作原理，可分为：

- SRAM** (Static Random Access Memory)
- DRAM** (Dynamic Random Access Memory)

静态RAM(SRAM): P108

由存储矩阵、地址译码器和读写控制电路组成。

与ROM相比，多了读写控制电路。

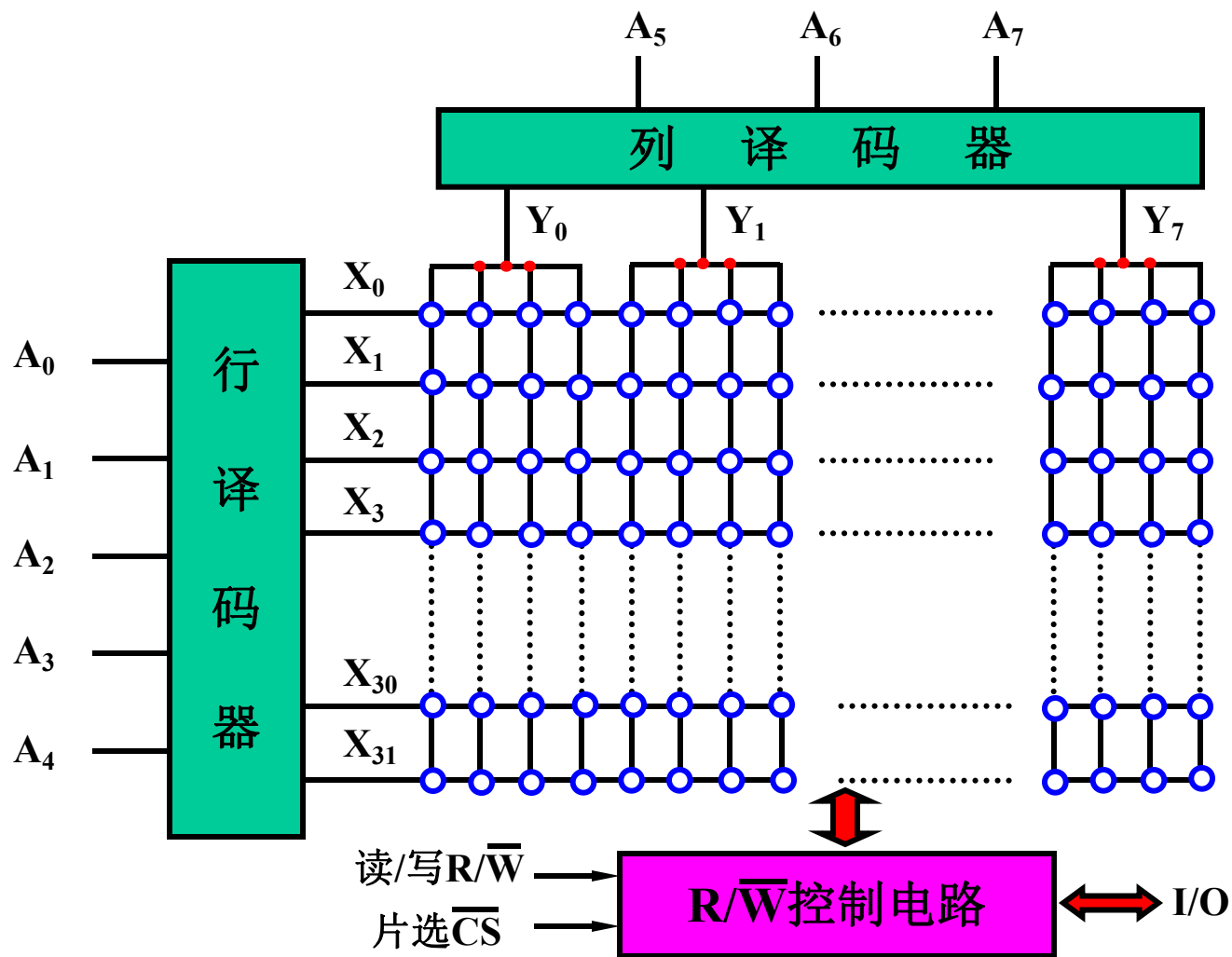
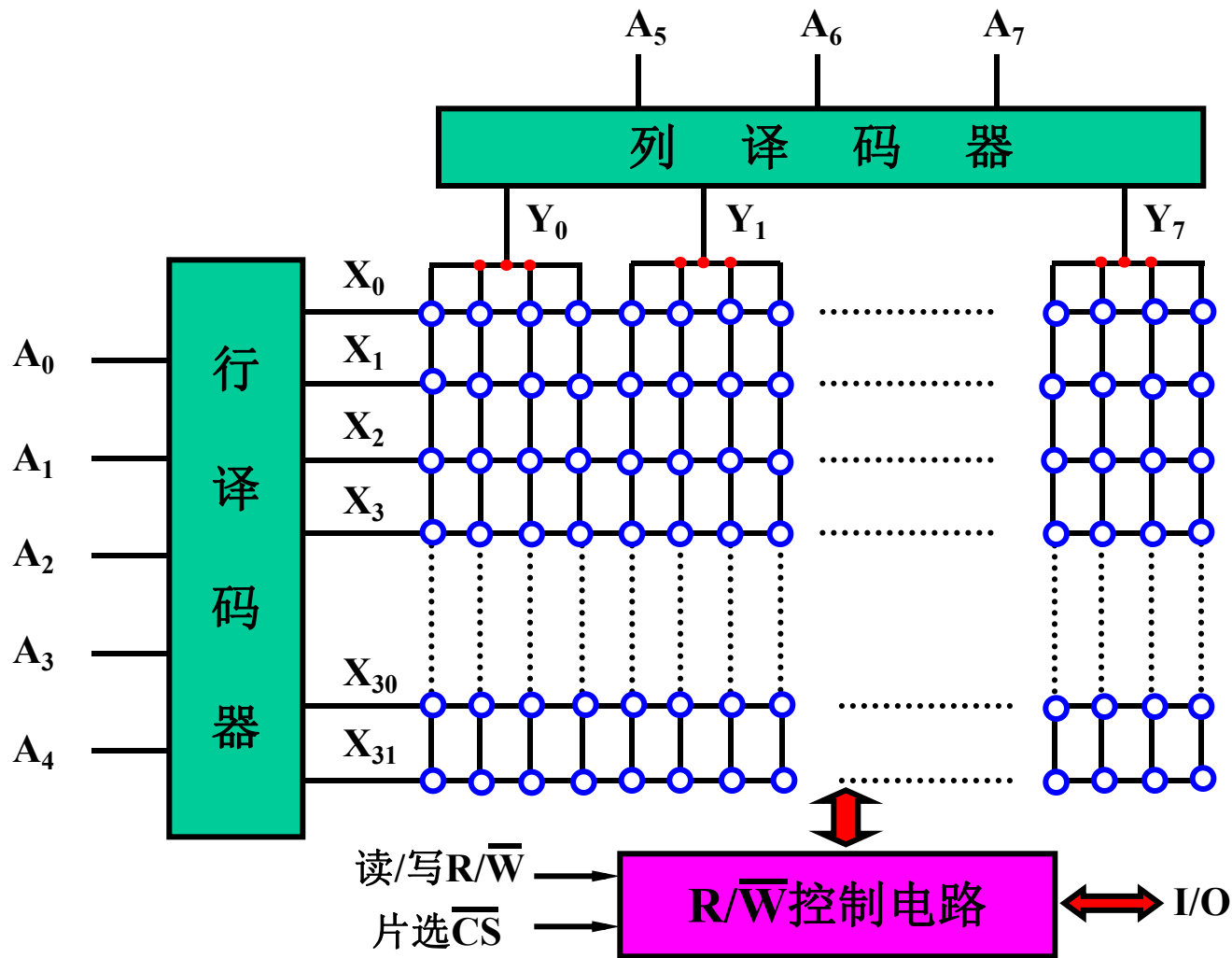
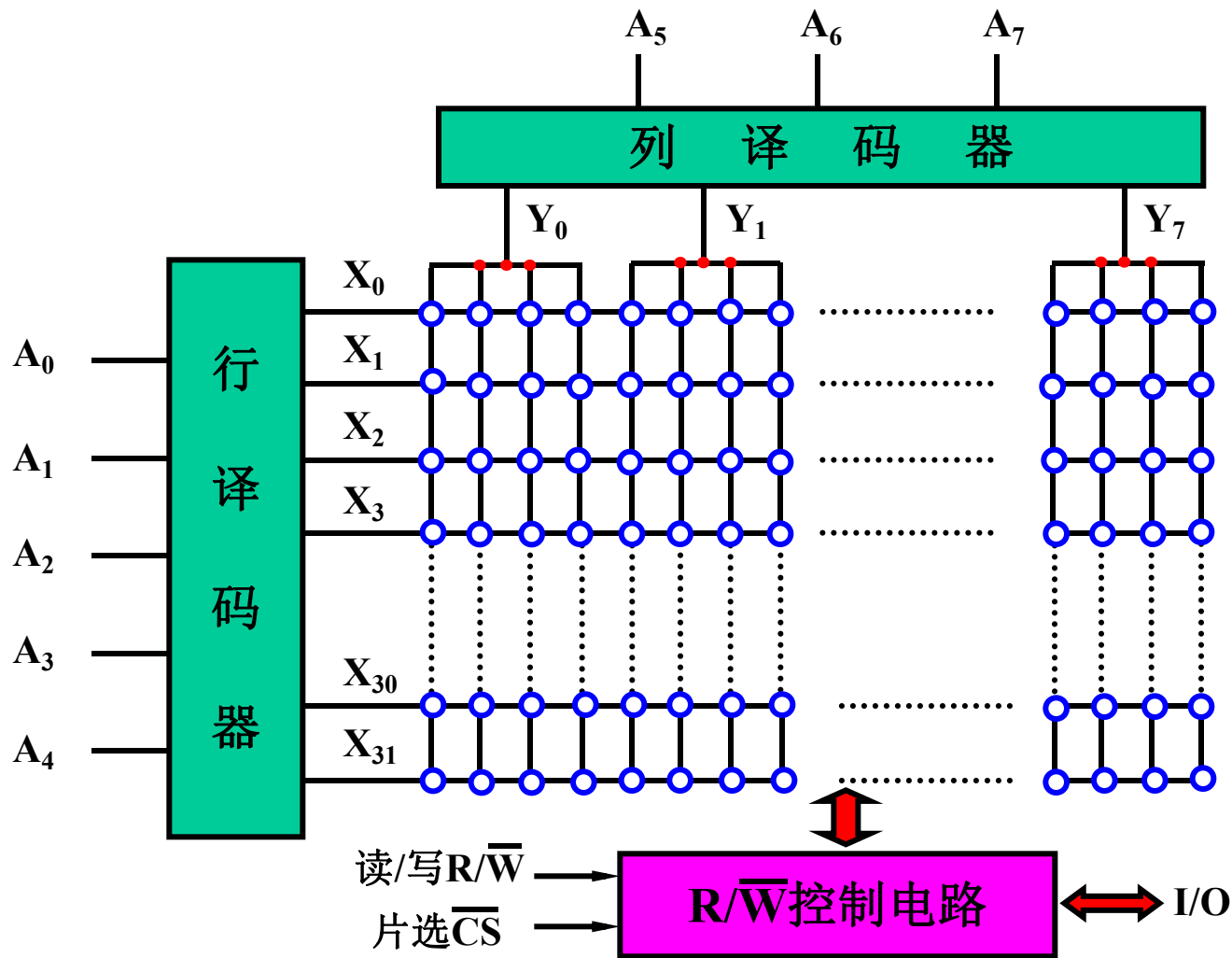


图 4.6.5 RAM结构示意图



• **存储矩阵：**

例：容量为256字 \times 4位的存储器，具有1024个基本存储单元。
 矩阵：32行 \times 32列。
 由于每个字占用4个存储单元，所以每4列为一个字列。



• 地址译码器：

读/写操作是以字节为单位进行的。

256个字节需要8位二进制地址 (A_7-A_0) 区分。

低5位作为行译码器输入；高3位作为列译码器输入。

- **RAM2114芯片:**

其存储容量为： $1024\text{字节} \times 4\text{位}$ （或 $1\text{K} \times 4$ ）。

存储器为 64×64 的矩阵，

思考其行地址线、列地址线分别为多少？

6 4

动态RAM (DRAM) :

计算机系统中最为常见的系统内存。

- **存储容量的扩展**

将多片芯片适当联接，以扩展存储容量。

- 1) **位扩展**

字节数不变，增加RAM的位数。

- 2) **字扩展**

每字节的位数不变，增加RAM的字节数。

1. 位扩展 $1024 \times 4 \Rightarrow 1024 \times 8$

需要片数 $N=2$ 。 $N = \frac{\text{目标存储器容量}}{\text{已有存储器容量}}$

所有输入信号都并联（地址信号、片选信号和读写信号）。
输出并列。

2. 字扩展 $1024 \times 4 \Rightarrow 4096 \times 4$

需要片数 $N=4$ 。 $N = \frac{\text{目标存储器容量}}{\text{已有存储器容量}}$

特点：必须使用译码器。

例：试用1K × 4位RAM扩展一个4K × 8位的存储器。

解：（1）确定芯片数：

$$N = \left(\frac{4K}{1K} \right) \times \left(\frac{8}{4} \right) = 8 \text{ (片)}$$

（2）确定地址线数D

$$2^D = 4096, D=12。$$

（3）用8片1K × 4位芯片构成，图略。

**概念：ROM和RAM的主要区别是什么？
它们各适用于那些场合？**

主要区别：

- 1) ROM工作时只能读出，不能写入，
但断电以后所存数据不会丢失；**
- 2) RAM工作时能进行读写操作，
但掉电以后数据丢失。**

应用：

**ROM适用于存放固定信息；
RAM适用于存放暂存信息。**

1. 只读存储器ROM的功能是(**A**)。

- A. 只能读出存储器的内容，且掉电后仍保持
- B. 只能将信息写入存储器中
- C. 可以随机读出或存入信息
- D. 只能读出存储器的内容，且掉电后信息全丢失

2. 将 $1K \times 4ROM$ 扩展为 $8K \times 8ROM$ 需用 $1K \times 4ROM$ (**C**)。

- A. 4片
- B. 8片
- C. 16片
- D. 32片

3. $16K \times 8RAM$ ，其地址线和数据线的数目分别为 (**D**)。

- A. 8条地址线，8条数据线
- B. 10条地址线，4条数据线
- C. 16条地址线，8条数据线
- D. 14条地址线，8条数据线

4. 已知某存储器芯片有地址线12条，数据线8条，则该存储器的存储容量是 **D** 位。

A. 1024×8 B. 4096×4 C. 2048×8 D. 4096×8

5. 将Intel 2114($1K \times 4$ 位)RAM扩展成为 $16K \times 8$ 位的存储器，需要Intel 2114芯片的片数以及需要增加的地址线条数分别为 **B**。

A. 16片和3条 B. 32片和4条
C. 64片和5条 D. 128片和6条

6. 信息可随时读出或写入，断电后信息立即全部消失的存储器是 **B**。

A、ROM B、RAM C、PROM

7. 只能读出、不能写入，但信息可永久保存的存储器是 A。

A、ROM B、RAM C、EPROM

8. 在下列电路中，不属于时序逻辑电路的器件是 D。

A. 计数器

B. 移位寄存器

C. 半导体随机存储器RAM

D. 半导体只读存储器ROM

作业1： 试用ROM阵列图实现下列一组多输出逻辑函数：

$$\begin{cases} F_1(A, B, C) = \overline{A}B + A\overline{B} + BC \\ F_2(A, B, C) = \sum m(3, 4, 5, 7) \\ F_3(A, B, C) = \overline{\overline{A}\overline{B}\overline{C}} + \overline{\overline{A}\overline{B}C} + \overline{\overline{A}B\overline{C}} + \overline{A\overline{B}\overline{C}} + \overline{ABC} \end{cases}$$

作业2： 用ROM设计两个一位二进制数A和B及进位输入CI的全加器，设本位和为S，进位输出为CO。

作业1： 试用ROM阵列图实现下列一组多输出逻辑函数：

$$\begin{cases} F_1(A, B, C) = \overline{A}B + A\overline{B} + BC \\ F_2(A, B, C) = \sum m(3,4,5,7) \\ F_3(A, B, C) = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC \end{cases}$$

解： 将以上三个逻辑函数化成由最小项组成的标准“与-或”式，即

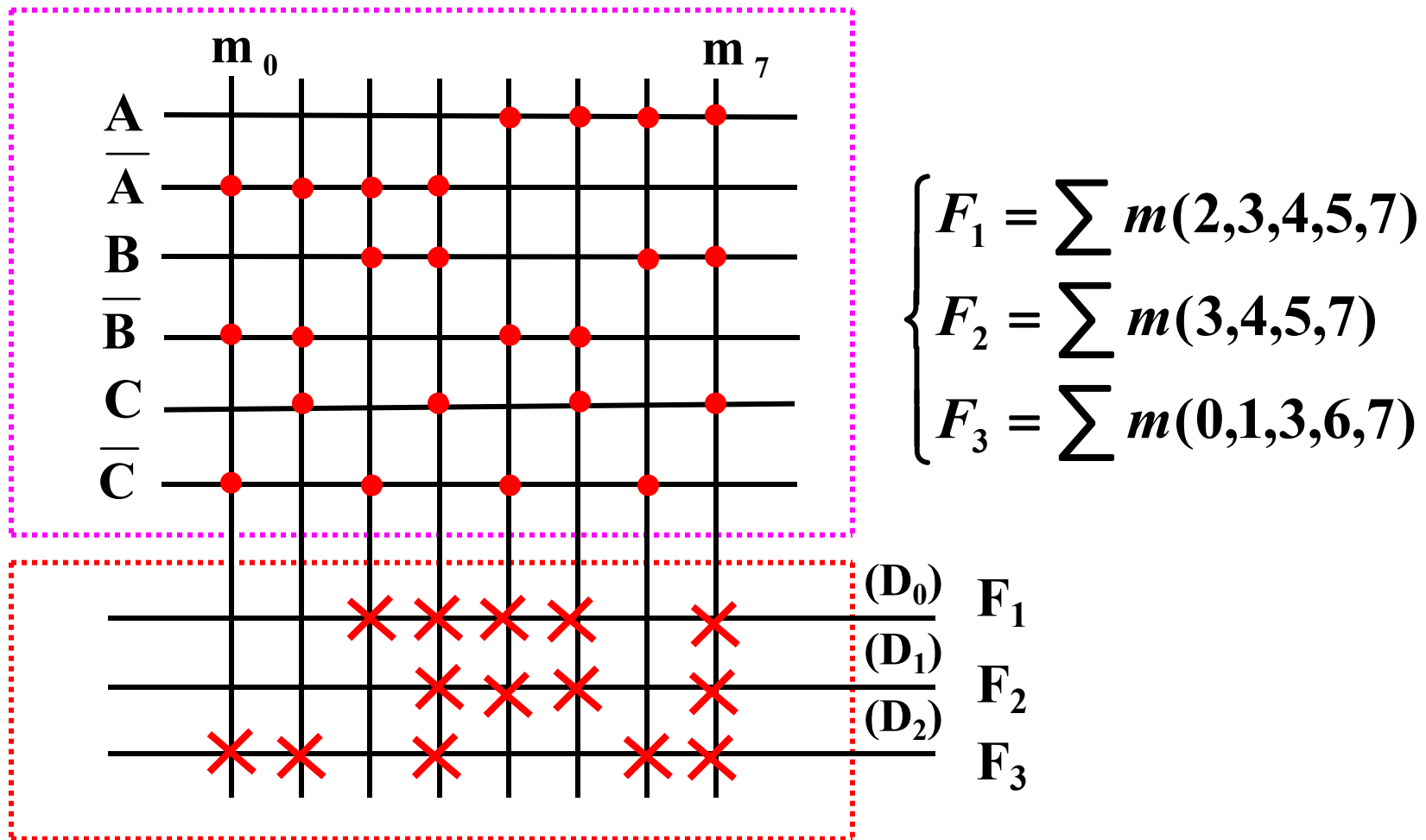
$$\begin{cases} F_1(A, B, C) = \sum m(2,3,4,5,7) \\ F_2(A, B, C) = \sum m(3,4,5,7) \\ F_3(A, B, C) = \sum m(0,1,3,6,7) \end{cases}$$

采用有**3位地址、3位数据输出的8字节×3位ROM**。

将A、B、C分别接至 $A_2A_1A_0$ 。

按逻辑函数要求存入相应数据。

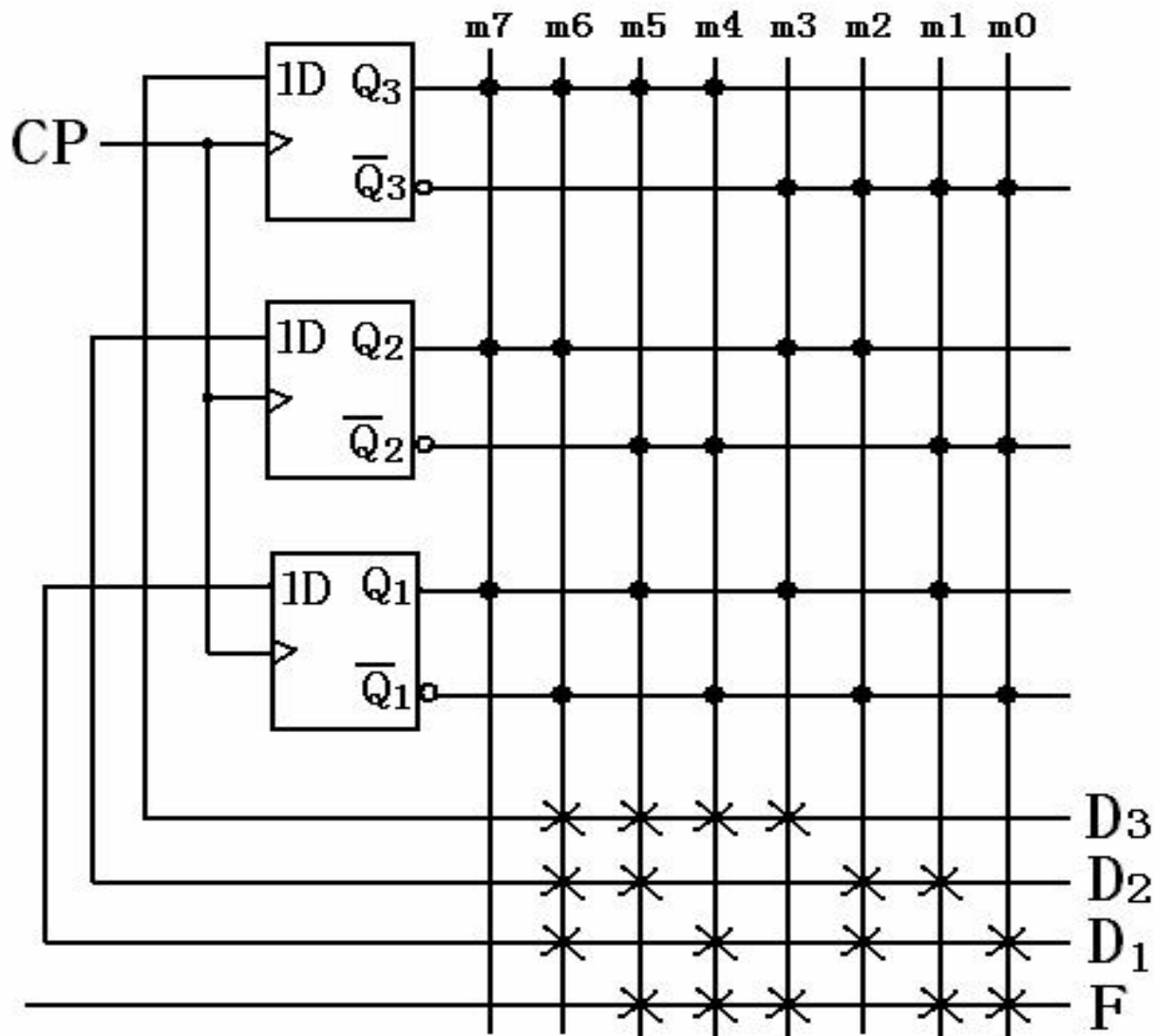
即可在数据输出端 D_0 、 D_1 、 D_2 得到 F_1 、 F_2 和 F_3 。



由PROM和DFF构成的电路如图所示，设 $Q_3Q_2Q_1$ 的初态为000。

1) 试填写 $Q_3Q_2Q_1$ 的状态转移表。

2) 试写出序列码F码型。




$$Q_3^{n+1} = D_3 = m_3 + m_4 + m_5 + m_6$$

$$Q_2^{n+1} = D_2 = m_1 + m_2 + m_5 + m_6$$

$$Q_1^{n+1} = D_1 = m_0 + m_2 + m_4 + m_6$$

$$F = m_0 + m_1 + m_3 + m_4 + m_5$$

Q_3	Q_2	Q_1	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



6.3 可编程逻辑阵列（PLA）和可编程阵列逻辑（PAL）

PLA是LDPLD中用户可配置性最好的器件，他的与门阵列和或门阵列都是可配置的。

一、PLA的结构与应用

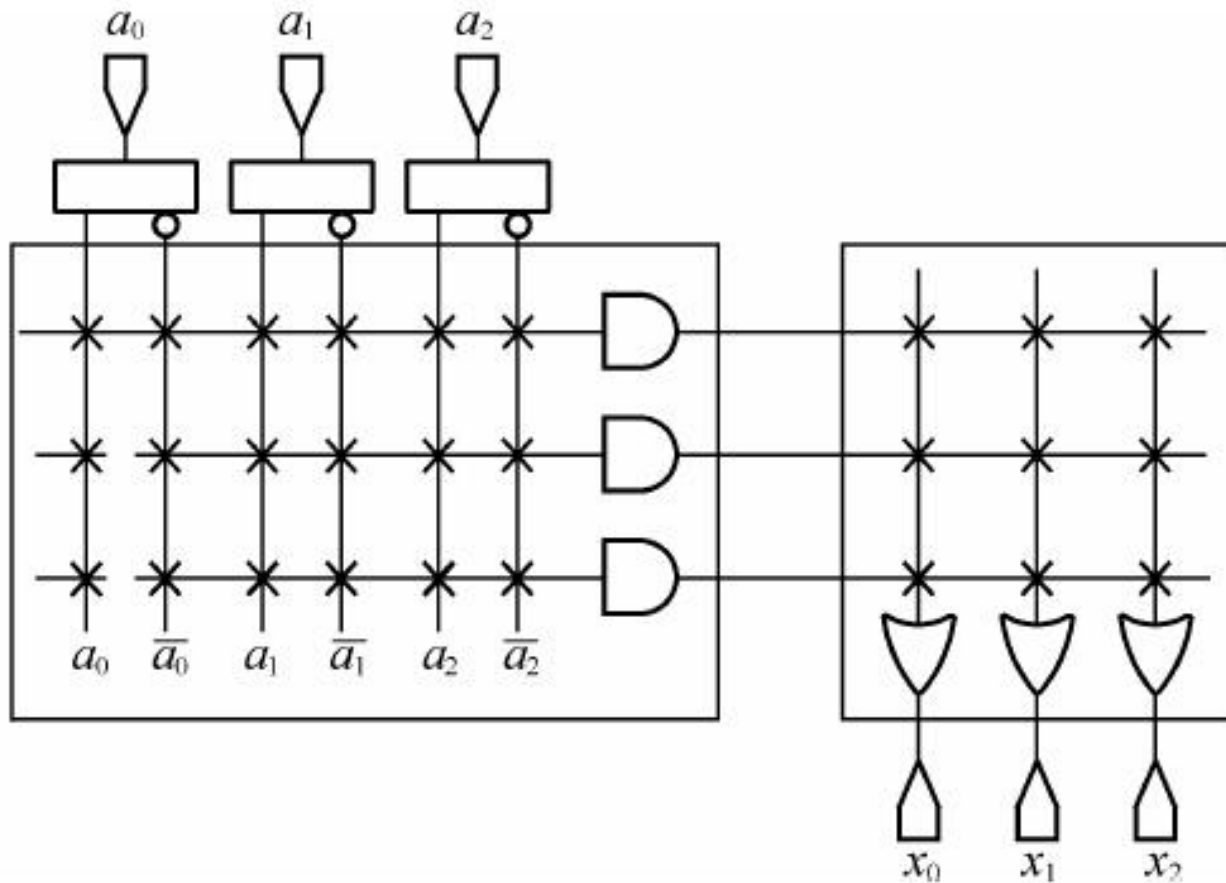


图6.3.1 一个未编程的3输入、3输出PLA的结构

PLA应用举例：

例 用PLA器件实现函数

$$F_1(A_2, A_1, A_0) = \sum m(3, 4, 6, 7),$$

$$F_2(A_2, A_1, A_0) = \sum m(0, 2, 3, 4, 7).$$

解：

F_1 、 F_2 的最简与或式：

$$F_1(A_2, A_1, A_0) = A_2 \bar{A}_0 + A_1 A_0$$

$$F_2(A_2, A_1, A_0) = A_1 A_0 + \bar{A}_2 A_1 + \bar{A}_1 \bar{A}_0$$

相应的实现电路如图所示。

$$F_1(A_2, A_1, A_0) = A_2 \bar{A}_0 + A_1 A_0$$

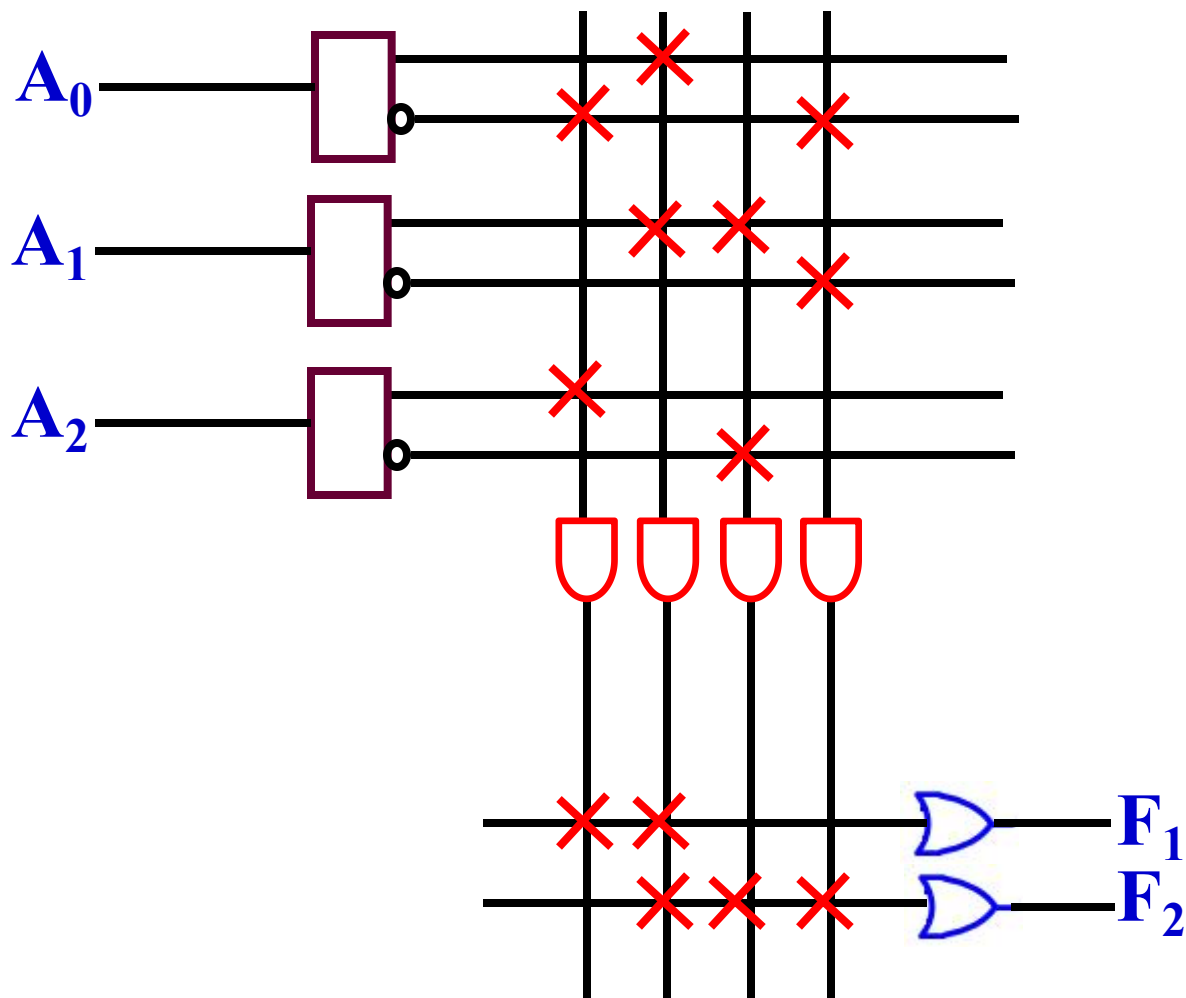
$$F_2(A_2, A_1, A_0) = A_1 A_0 + \bar{A}_2 A_1 + \bar{A}_1 \bar{A}_0$$

乘积项: $A_2 \bar{A}_0$

$A_1 A_0$

$\bar{A}_2 A_1$

$\bar{A}_1 \bar{A}_0$

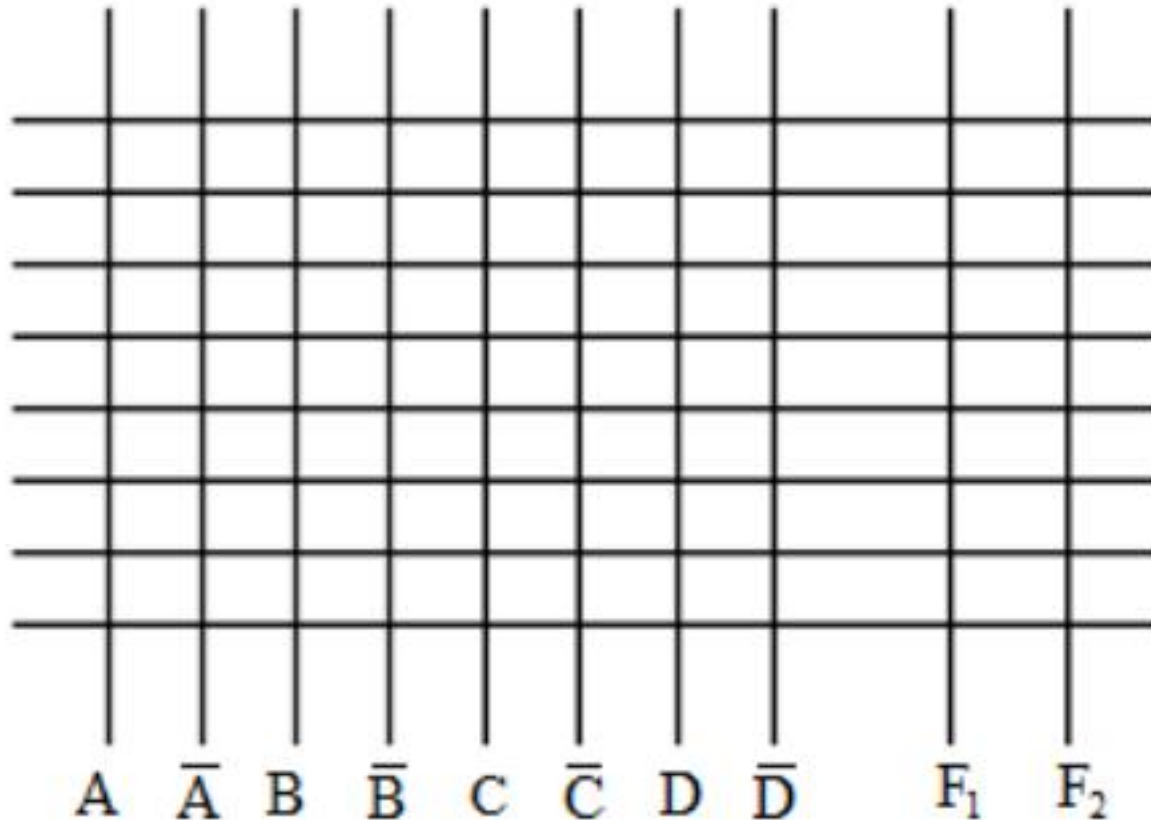


已知多输出组合电路的输出函数如下：

$$F_1(A,B,C,D)=\sum_m(0,1,4,7,9,10,13) + \sum_\phi(2,5,8,12,15),$$

$$F_2(A,B,C,D)=\sum_m(5,7,13,15)$$

试用PLA实现该电路，且要求电路最简，请写出化简过程并将图中PLA的阵列结构图画完整。



PROM与PLA实现组合逻辑函数的区别：

PROM的与阵列固定，或阵列可编程。

PLA的与阵列和或阵列均可以编程。

PROM实现函数，与阵输出要用最小项表达式。

PLA实现函数，与阵输出用最简与或式。

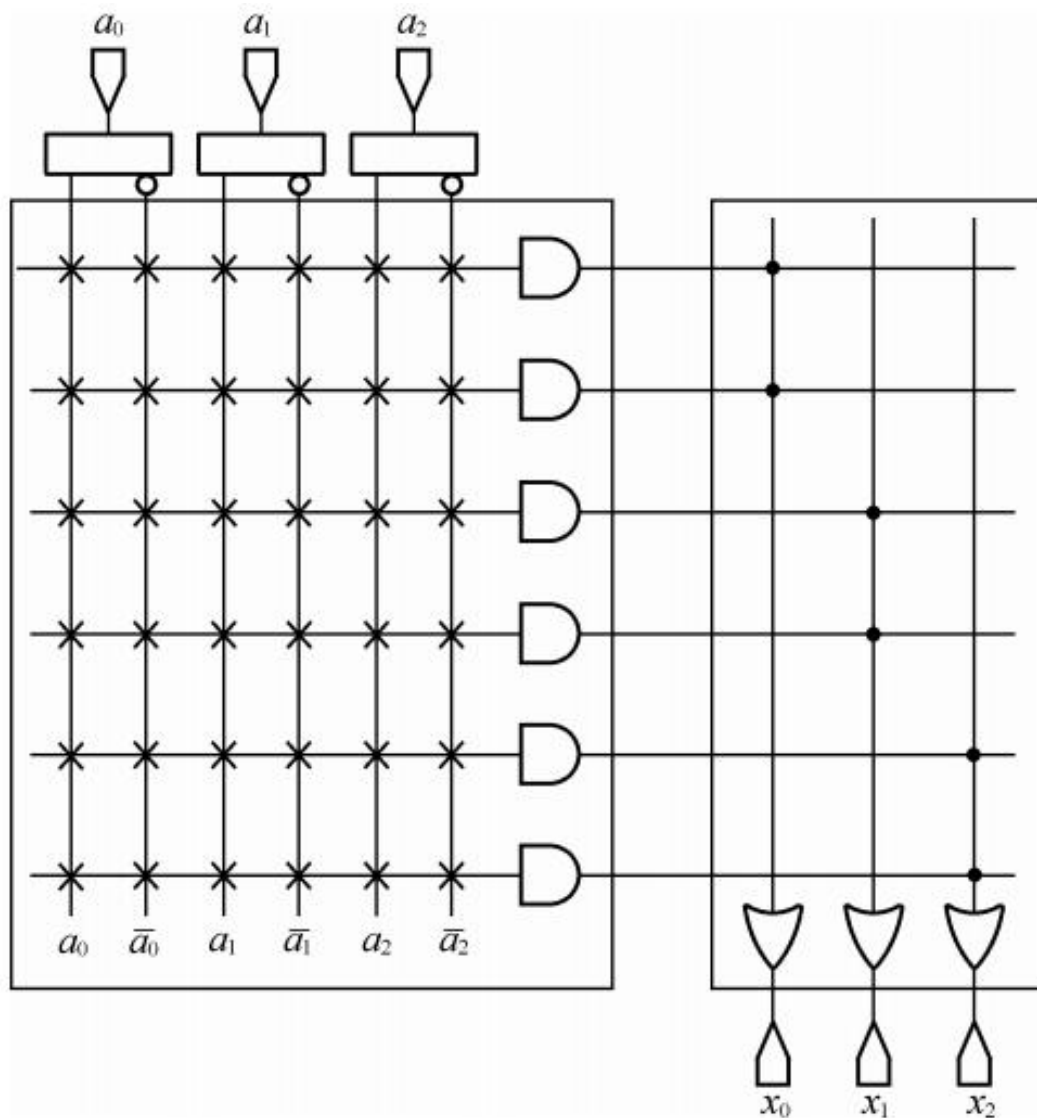
PROM点阵图中，与阵列打点，或阵列打×。

PLA点阵图中，与阵列或阵列均打×。

6.3.2 可编程阵列逻辑 (PAL)

可编程阵列逻辑 (PAL) 的诞生是为了解决PLA的速度问题。在概念上，PAL几乎与PROM相反，它的与门阵列是可编程的，而或门阵列是固定的。

一个简单的3输入、3输出，每个输出包含两个乘积项的PAL的结构如图6.3.3所示。



- 相比于PLA，PAL由于在结构上只有一个阵列可编程，因而工作速度要比PLA快很多。
- 由于PAL的或门阵列中每个或门所包含的乘积项的个数是固定的，所以由PAL实现的组合逻辑函数最多只能包含不超过预定个数的乘积项。
- 但随着使用的深入，工程师发现PAL除了存在只允许少量的乘积项相或这一问题，而且其衍生品种过多还带来了新的问题：原有设计若有功能上的更新换代，则必须更换不同类型的PAL芯片。
- 由于这些弱点的存在，不久，Lattice公司在PAL的基础上加以改进，发明了通用阵列逻辑（GAL），一举取代了PAL。

6.4 通用阵列逻辑 (GAL)

GAL英文全称: Generic Array Logic。

GAL的结构沿袭了PAL的设计（与门阵列可编程、或门阵列固定），但增加了输出可配置的功能。

1. GAL的总体结构

- 1) 输入端
- 2) 与阵列部分
- 3) 输出宏单元
- 4) 系统时钟
- 5) 输出三态控制端

- 该器件包含8个类PAL块，每个类PAL块的与门阵列中包含8个32输入与门，或门阵列包含在输出模块OLMC中，OLMC的输出具有反馈功能；同时增加了1个全局时钟输入缓冲器和1个选通信号输入缓冲器，以供实现较大的数字系统时使用。
- 在GAL16V8芯片中，有8个管脚（2~9脚）只能用作输入，另有8个管脚（12~19脚）既可配置为输入也可配置为输出，所以它最多可以有16个输入管脚，8个输出管脚，这也是芯片信号中16和8两个数字的含义。另外还有1个管脚专门负责时钟输入（1脚），1个管脚专门负责使能控制（11脚）。

组成:

1.输入端: 2~9脚共8个输入端, 每个输入端有一个缓冲器, 并由缓冲器引出两个互补的输出到与阵列;

2.与阵列部分: 它由8根输入及8根输出各引出两根互补的输出构成32列; 8×8个与门, 可实现64个乘积项;

3.输出宏单元: GAL16V8共有8个输出宏单元, 分别对应于12~19脚。每个宏单元的电路可以通过编程实现所有PAL输出结构实现的功能;

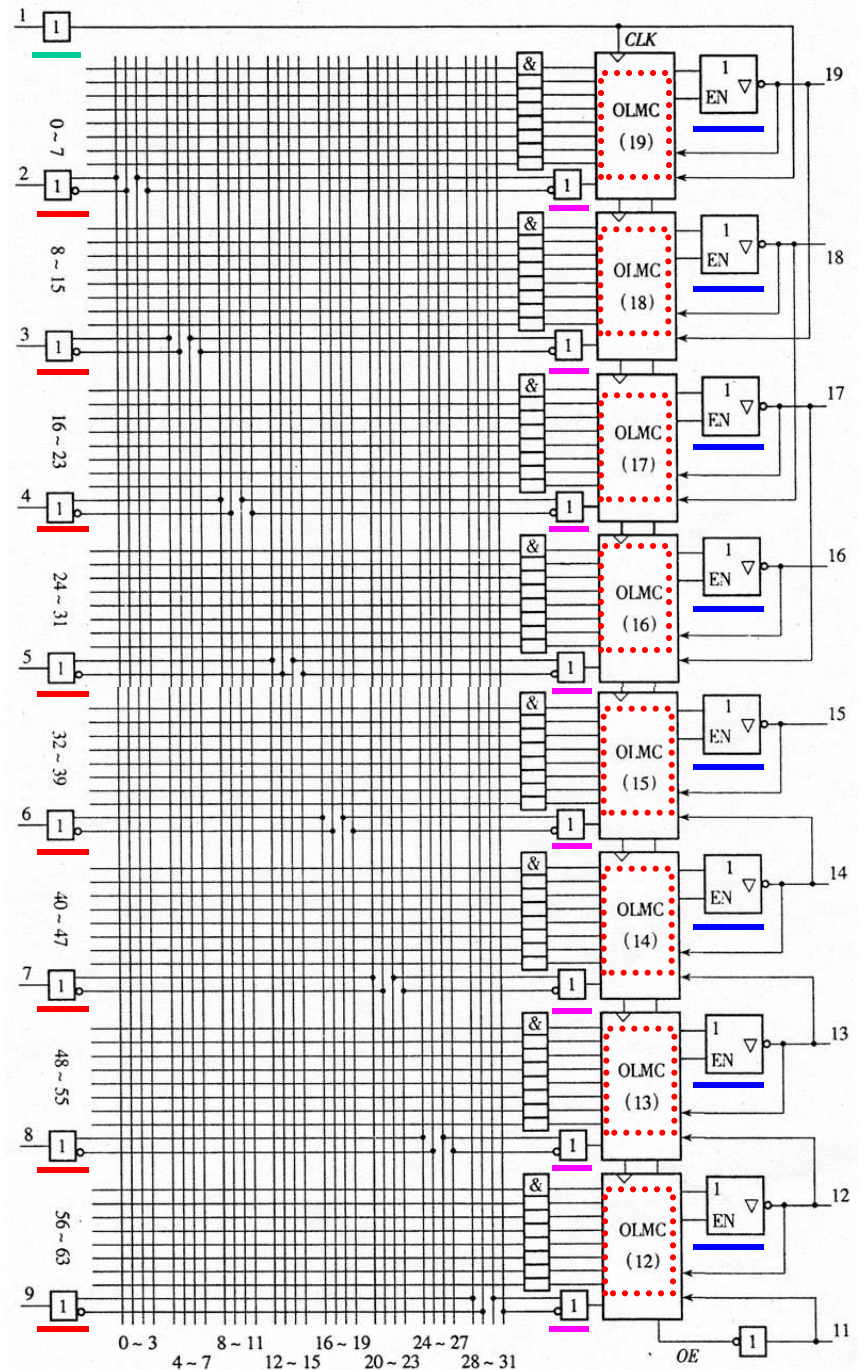


图 10.6.1 GAL16V8 逻辑图

组成：

4.系统时钟： GAL16V8的**1脚**为系统时钟输入端,与每个输出宏单元中D触发器时钟输入端相连,可见GAL器件**只能实现同步时序电路**,而无法实现异步的时序电路；

5.输出三态控制端： 11脚为器件的三态控制公共端。

共有20个引脚。使用时最多可以有16个引脚作为输入端,而输出端最多有8个,这是GAL16V8中的16和8的含义。

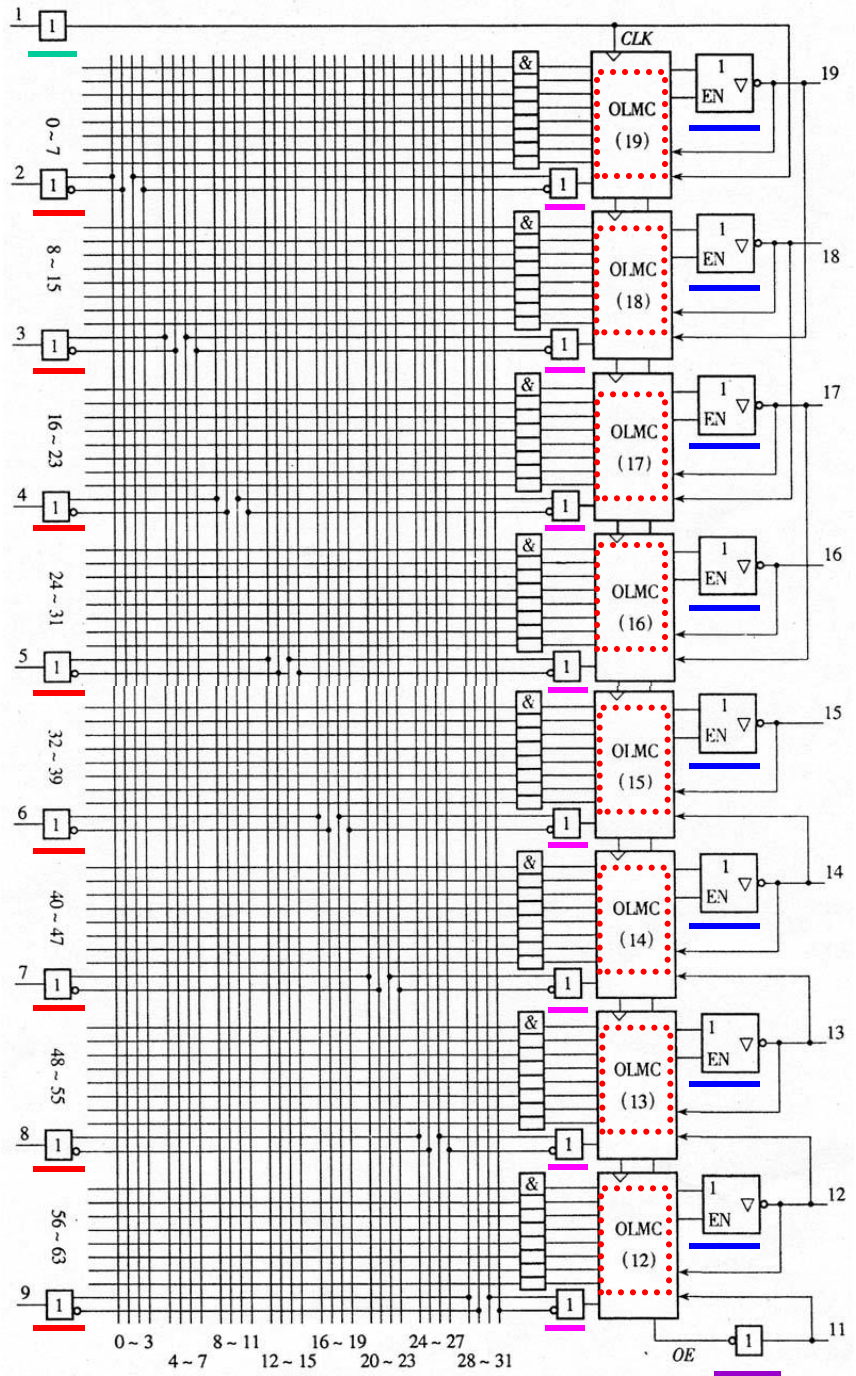


图 10.6.1 GAL16V8 逻辑图