

# 第十章 半导体存储器及可编程逻辑器件

## 10.1 半导体存储器概述

## 10.2 随机存储器

## 10.3 只读存储器

## 10.4 可编程逻辑器件

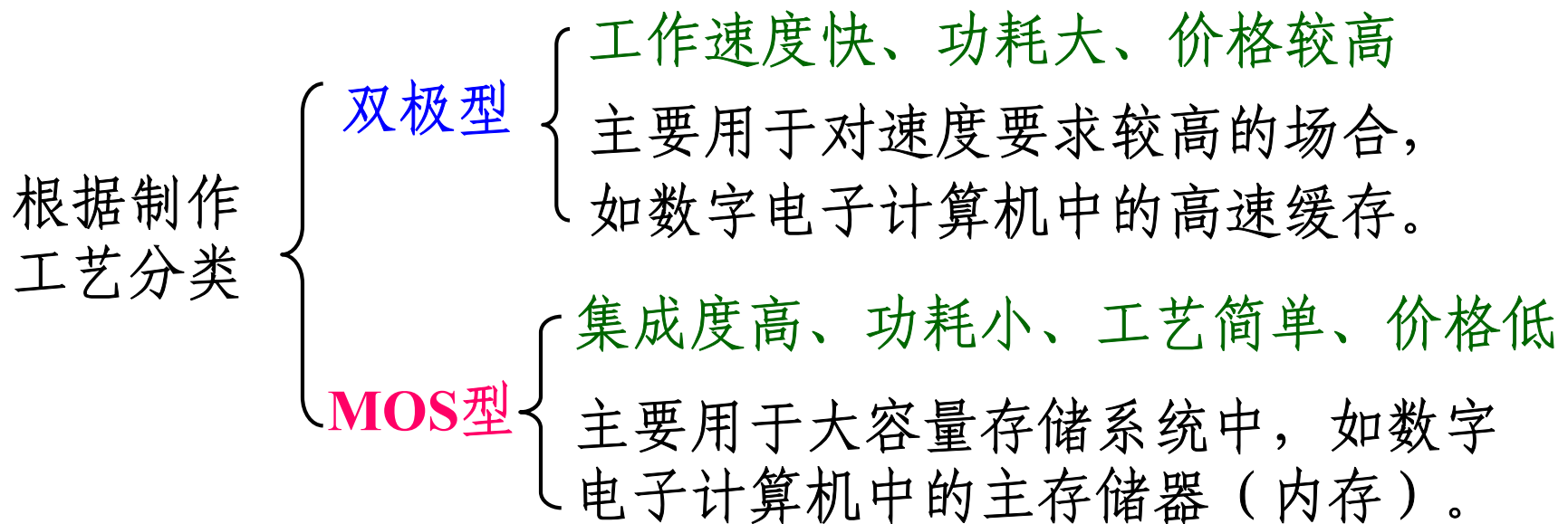
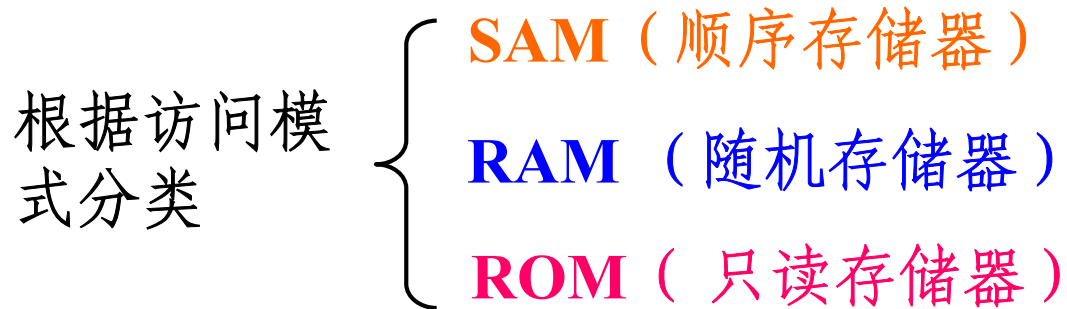
# 第十章 半导体存储器及可编程逻辑器件

## § 10.1 半导体存储器简介

半导体存储器是存储大量信息的器件，由许多存储单元组成。

每个存储单元都有唯一的地址代码，能存储一位（或一组）二进制信息。

## 10.1.1 存储器分类



## 10.1.2 存储器的技术指标

存储周期

存储器的性能基本上取决于从存储器读出信息和把信息写入存储器的速率。

把连续两次读（写）操作间隔的最短时间称为存取周期。存取周期越短越好。

存储容量

(1) 存储单元:

存储1位信息的单元

(2) 二进制数据单位

位

字节

存储字

最小的二进制单元是位。

8位二进制称作一个字节。

byte (字节) 缩写 B, 1 Byte = 8 bits

一个完整的信息单元叫做存储字。

一组存储单元, 表示某种类型的数据或信息

字长: 8位/字, 16 位/字, 32 位/字

$$2^{10} = 1024 = 1k; \quad 2^{20} = 1M; \quad 2^{30} = 1G.$$

(3) 容量 (存储单元总数)

字数  $\times$  位数

容量  $2k \times 16$

$\left\{ \begin{array}{l} 2k \text{ 字}(2^{11} \text{个字}) \\ 16 \text{位}(2 \text{ 字节}) \end{array} \right.$

字长16位

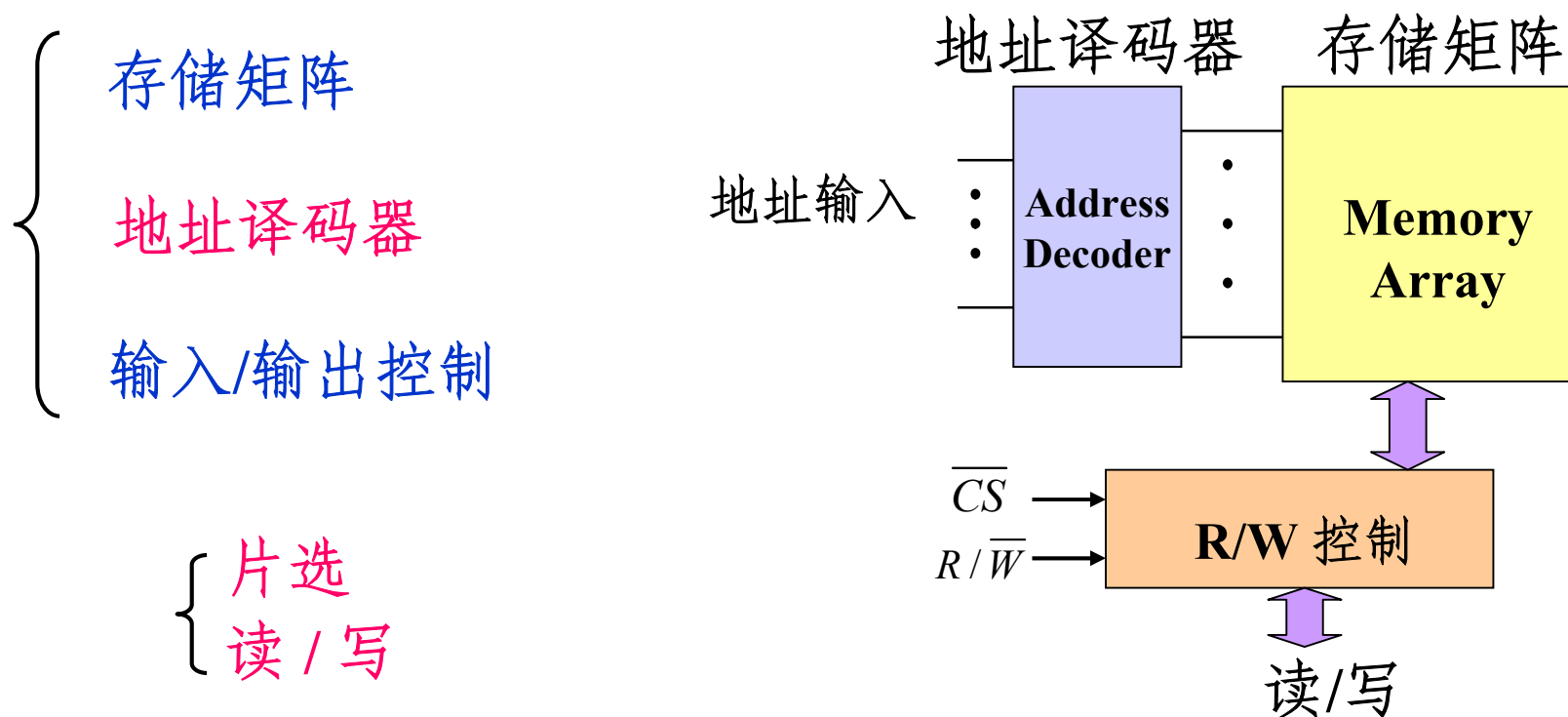
常用多少字节表示容量    32k 单元    或    4k 字节

## § 10.2 RAM 随机存储器

优点：所有随机存储器都有读和写的能力。

缺点：断电将丢失全部数据

### 10.2.1 随机存储器结构 (基本结构)



## 1. 存储矩阵

随机存储器主要部分

1024 存储字分配在  
一个  $32 \times 32$  的矩阵

编号:

32行编号为

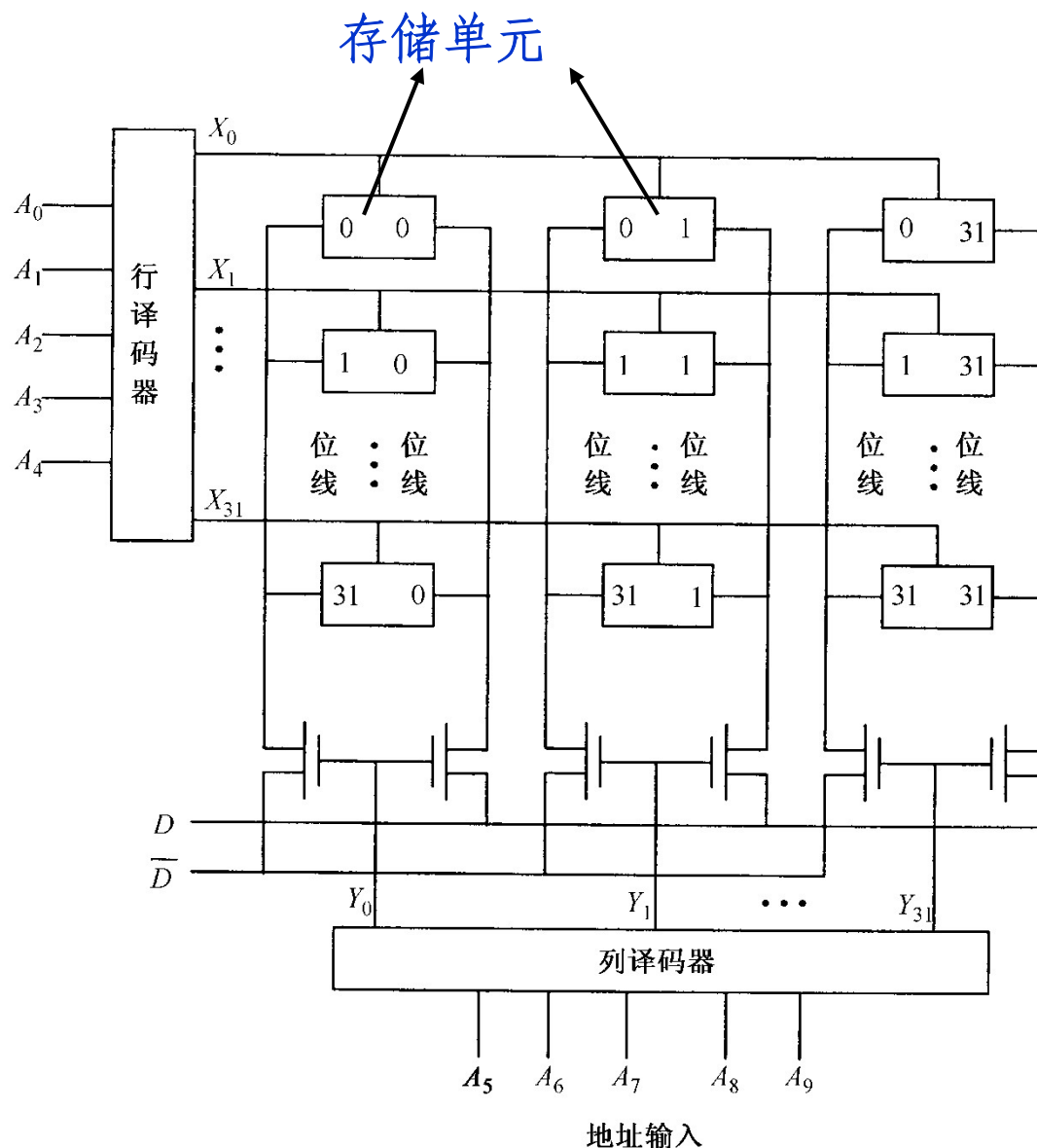
$X_0, X_1, \dots, X_{31}$

32列编号为

$Y_0, Y_1, \dots, Y_{31}$

每个存储单元都有固定的  
编号 ( $X_i$  行、 $Y_j$  列)

即地址

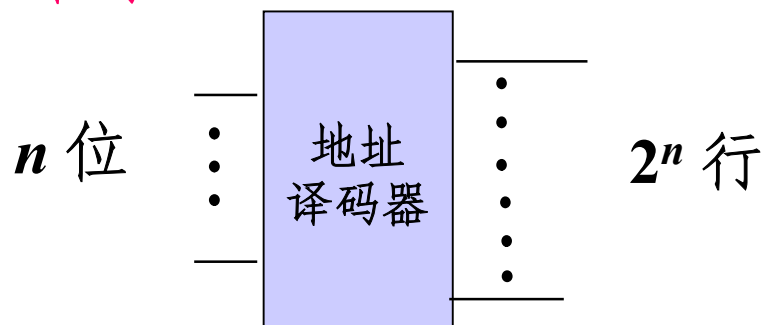


1k × 1位 存储矩阵和地址译码器

## 2. 地址译码器

{ 单译码  
双译码

单译码

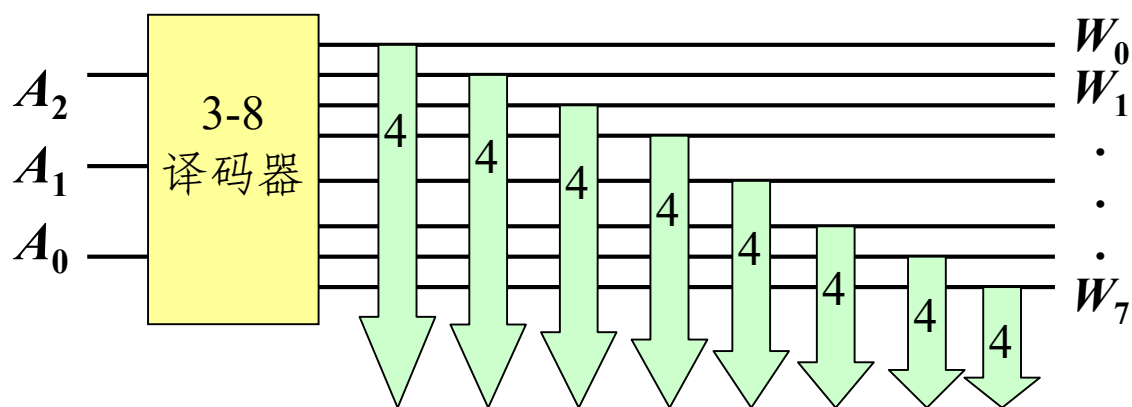


双译码

将地址所对应的二进制数译成：行选信号和列选信号，从而选中该存储单元。

如：

$2^3 \times 4$  RAM





1024 存储字:  $2^{10}$

存储字线(地址线): 10

地址译码器双译码结构:

行地址译码器:

5-32 译码器

输入:  $A_0, A_1, \dots, A_4$

输出:  $X_0, X_1, \dots, X_{31}$

列地址译码器:

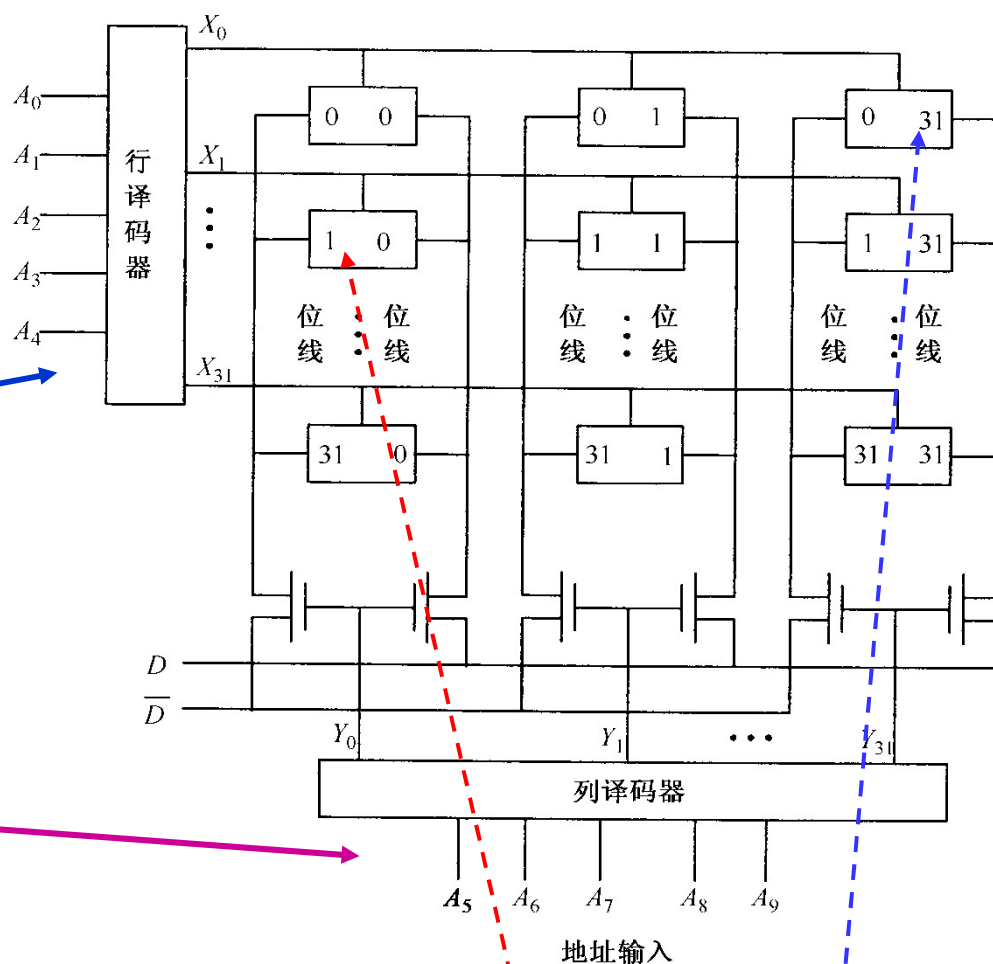
5-32 译码器

输入:  $A_5, A_6, \dots, A_9$

输出:  $Y_0, Y_1, \dots, Y_{31}$

地址:  $A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$   
 $= 00000\ 00001$

地址:  $A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$   
 $= 11111\ 00000$

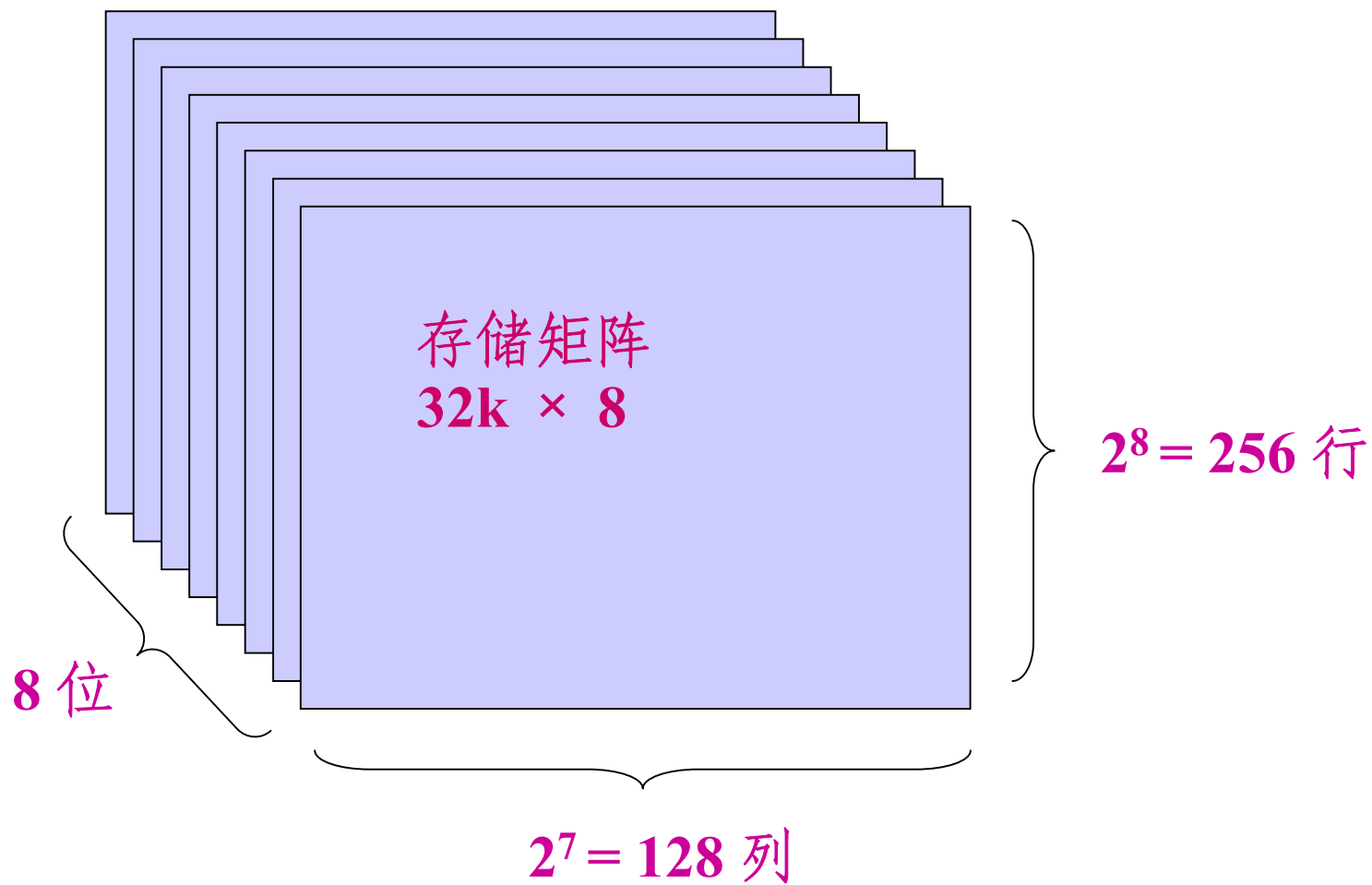


行线  $X_1 = 1$ 、  
列线  $Y_0 = 1$

行线  $X_0 = 1$ 、  
列线  $Y_{31} = 1$

立体图

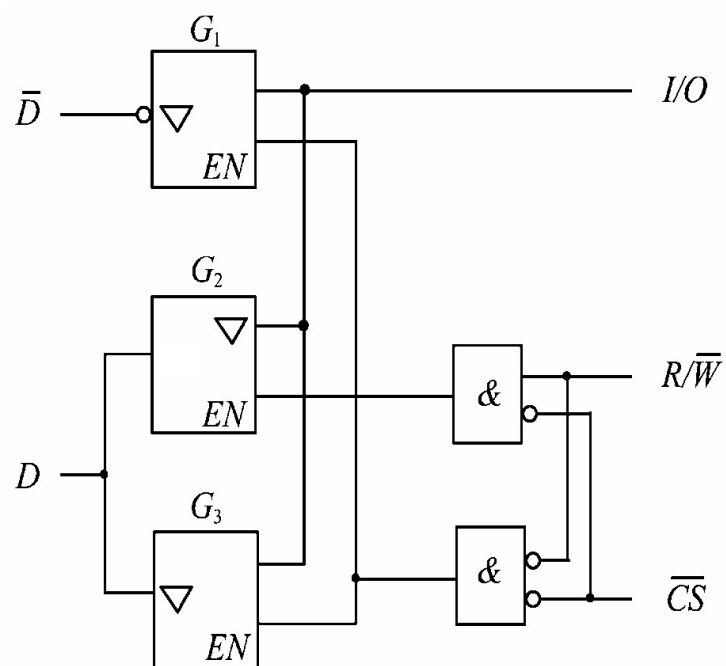
$$32k \times 8 = 2^{15} \times 8 = 2^8 \times 2^7 \times 8$$



### 3. RAM 输入/输出控制电路

读/写信号  $R/\overline{W}$   $\begin{cases} R/\overline{W}=1 & \text{读} \\ R/\overline{W}=0 & \text{写} \end{cases}$

片选信号  $\overline{CS}$  低电平有效



1位数据输入/输出控制电路

输入/输出端**数据线(I/O)**  
的条数 = 寄存器位数

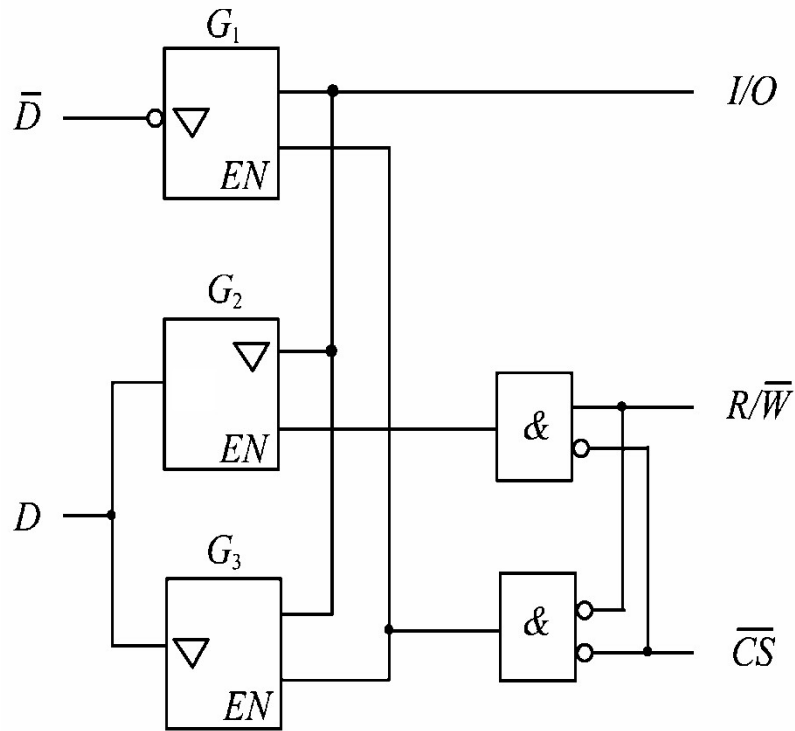
**存储容量  $2k \times 16$  RAM**

**数据线 16 条**

$D$  和  $\overline{D}$  分别与存储矩阵  
的两条位线相连。

$G_1$ 、 $G_2$ 、 $G_3$  为三态门

## 1位数据输入/输出控制电路



当  $\overline{CS}=1$  时, **RAM** 芯片呈高阻态, 不进行任何操作;

当  $\overline{CS}=0$  时, 芯片被选通

若  $R/\overline{W}=1$ , 则  $G_2$  导通,  
 $G_1$  和  $G_3$  呈高阻态截止;

被选中的字的存储单元  
与数据输入/输出端 ***I/O*** 相通,  
进行读操作;

若  $R/\overline{W}=0$ , 则  $G_1$  和  $G_3$  导通,  $G_2$  呈高阻态截止;

此时 ***I/O*** 段数据以互补形式 (***D, D-bar***) 出现在内部数据上, 并被存入所选中的字的存储单元中, 执行写操作。

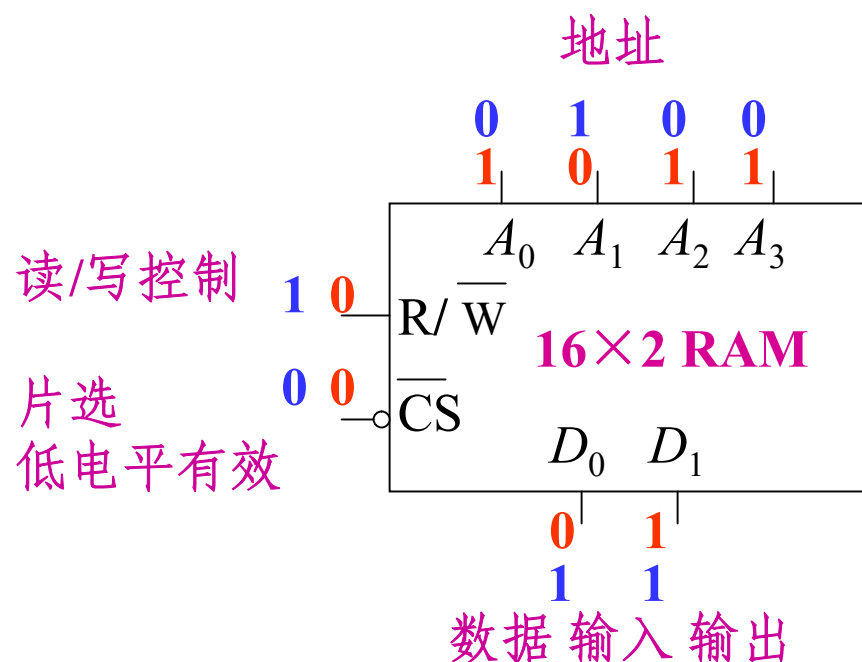
## 10.2.2 随机存储器（芯片简介）

### 例 1. $16 \times 2$ 随机存储器

容量:  $16 \times 2$  单元

存储字:  $16 = 2^4$       地址线: 4 ( $A_3A_2A_1A_0$ )

字长: 2 位      数据线: 2 ( $D_1D_0$ )



$A_3, D_1$  高位

练习

## 例 2. RAM2114

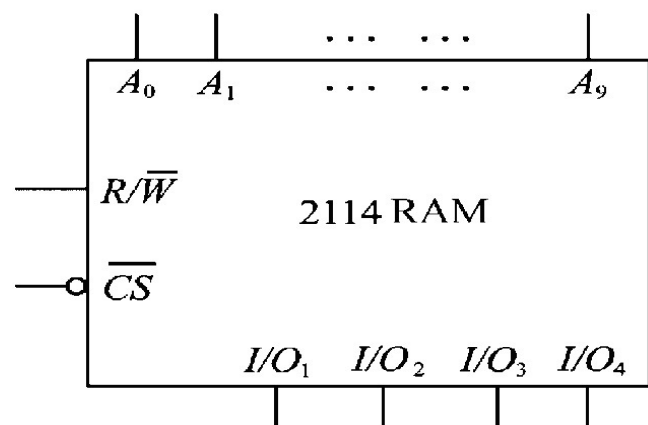
容量:

1024 字  $\times$  4 位

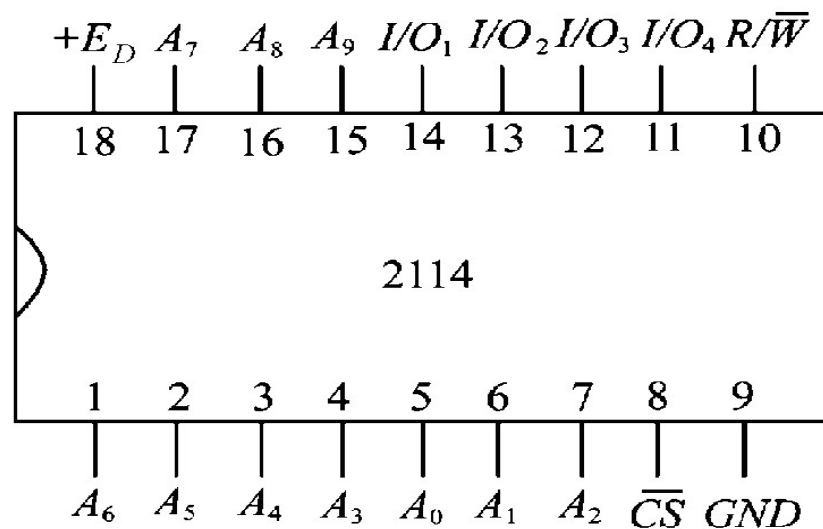
地址线: 10,  $A_0 \sim A_9$ ,

数据线 (输入/输出): 4,  $I/O_1 \sim I/O_4$

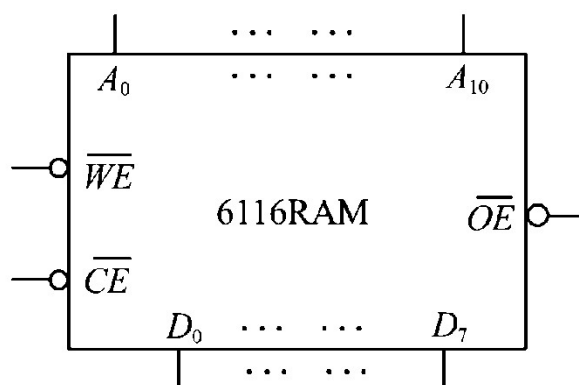
符号



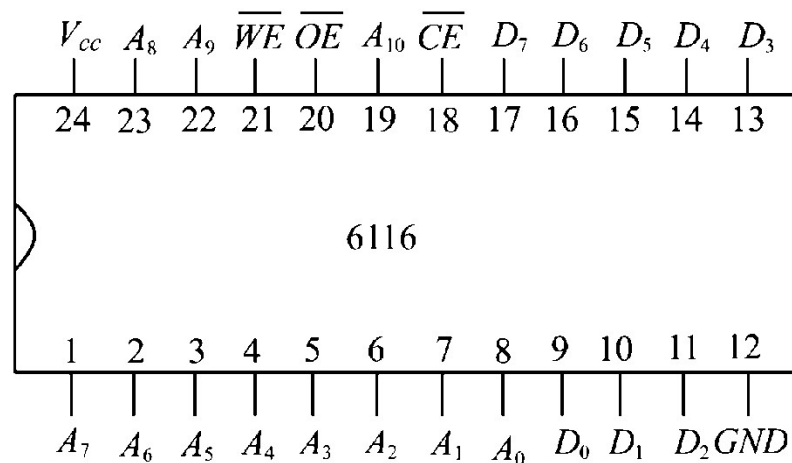
管脚图



## 例 3. RAM 6116



(a) 符号图



(b) 管脚图

字：地址线：11

$2^{11}$  字(2k)

位：8 位

容量： $2^{11} \times 8 = 2k \times 8 = 16k$  个单元 = 2k 字节

$\overline{CE}$ : 片选, 低电平有效;

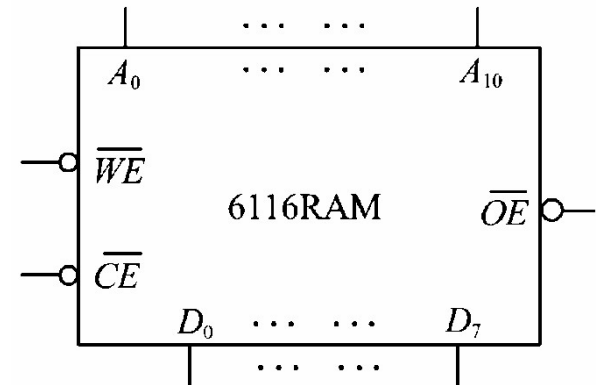
$\overline{OE}$ : 使能端, 低电平有效;

$\overline{WE}$ : 读/写控制

## 6116 RAM

### 3 种操作方式

{ 写入方式  
读出方式  
低功耗维持方式



写状态:  $\overline{CE} = 0, \overline{WE} = 0, \overline{OE} = 1;$

读状态:  $\overline{CE} = 0, \overline{WE} = 1, \overline{OE} = 0;$

低功耗维持:  $\overline{CE} = 1$

此时器件电流仅  $20 \mu A$  左右, 系统断电时可以用电池保持 RAM 内容。



## 10.2.4 RAM 扩展

容量不够大  
需要级联

{

字长扩展(位扩展)

字地址扩展 (字扩展)

### 1. 位扩展

方法：相同**RAM**并行连接

共用：

{

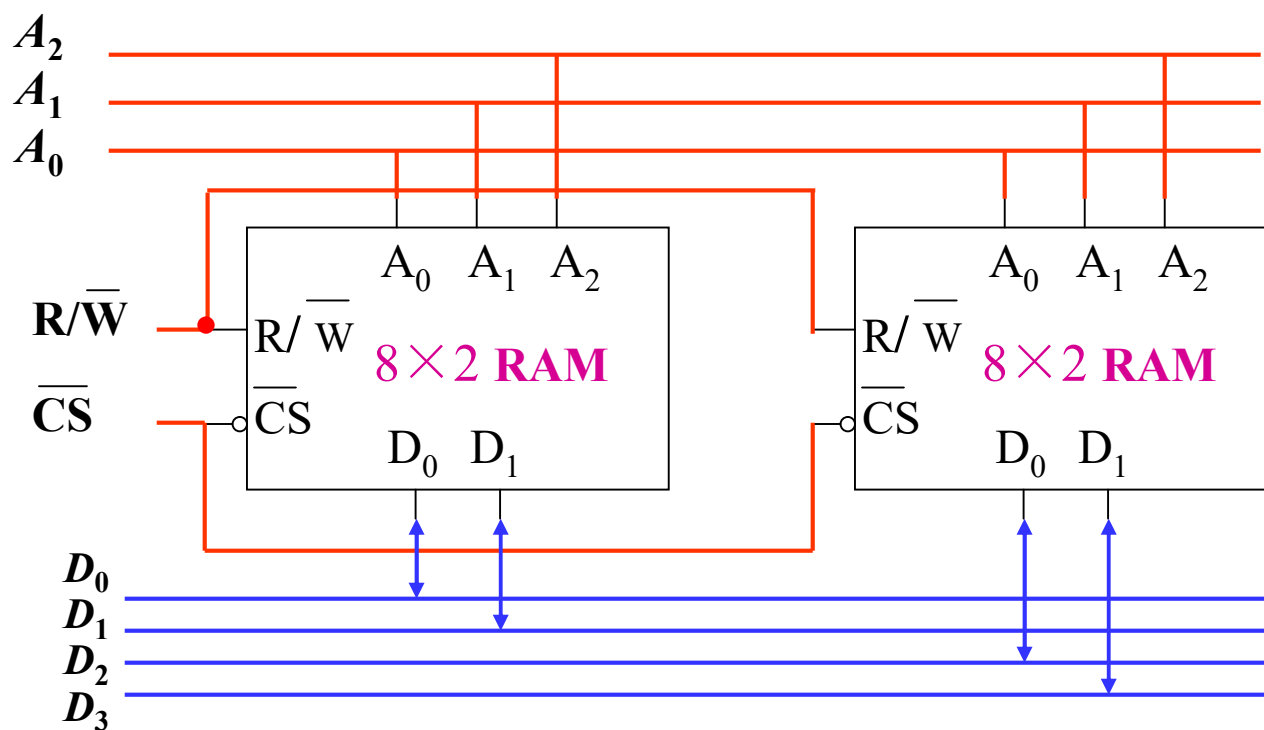
地址线

$R/\overline{W}$

$\overline{CS}$

例：将  $8 \times 2$  RAM 扩展为  $8 \times 4$  RAM

$$\frac{8 \times 4}{8 \times 2} = 2 \quad (8 \times 2 \text{ RAM})$$



存储器同时工作，地址范围不改变，字长扩展


2 位  $\rightarrow$  4 位

## 2. 字扩展

增加地址线

使用  $\overline{CS}$  扩展字

共用：

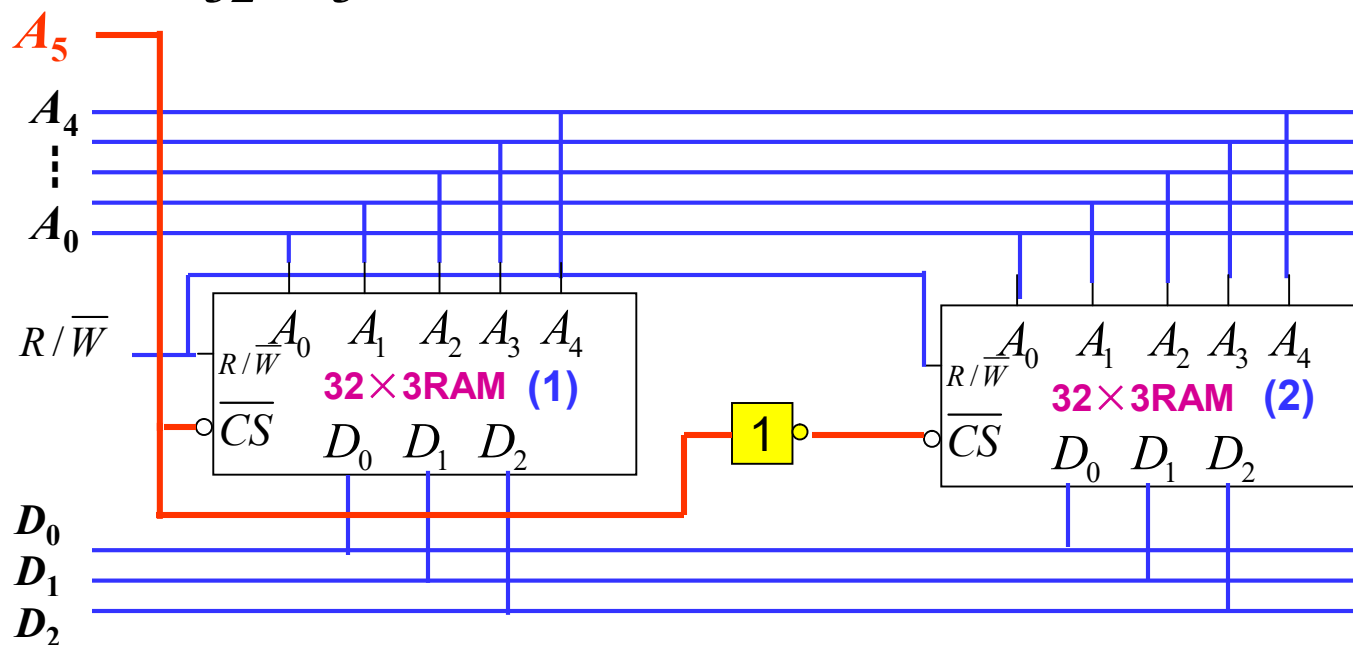


- 原始地址线
- $R / \overline{W}$
- 数据线

# 例 1. 将 $32 \times 3$ RAM 扩展为 $64 \times 3$ RAM

$$\frac{64 \times 3}{32 \times 3} = 2 \text{ (} 32 \times 3 \text{ RAM)}$$

地址线 5  $\rightarrow$  6



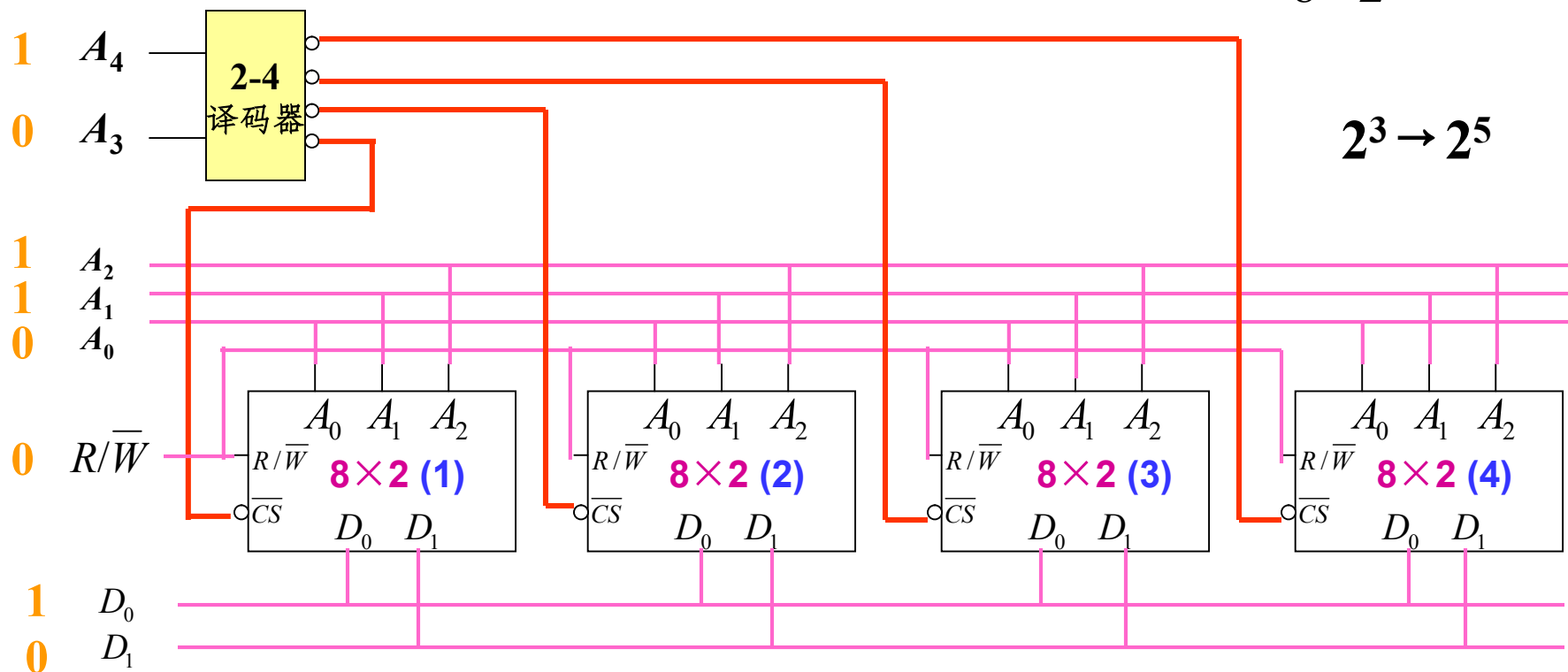
$A_5 = 0$  (1) 运转, (2) 停止; 地址 000000-011111

$A_5 = 1$  (1) 停止, (2) 运转; 地址 100000-111111

分时工作

## 例 2. 将 $8 \times 2$ RAM 扩展为 $32 \times 2$ RAM

$$\frac{32 \times 2}{8 \times 2} = 4$$



$$2^3 \rightarrow 2^5$$

地址范围:

$$A_4 A_3 A_2 A_1 A_0$$

(1) **00000-00111**

(2) **01000-01111**

(3) **10000-10111**

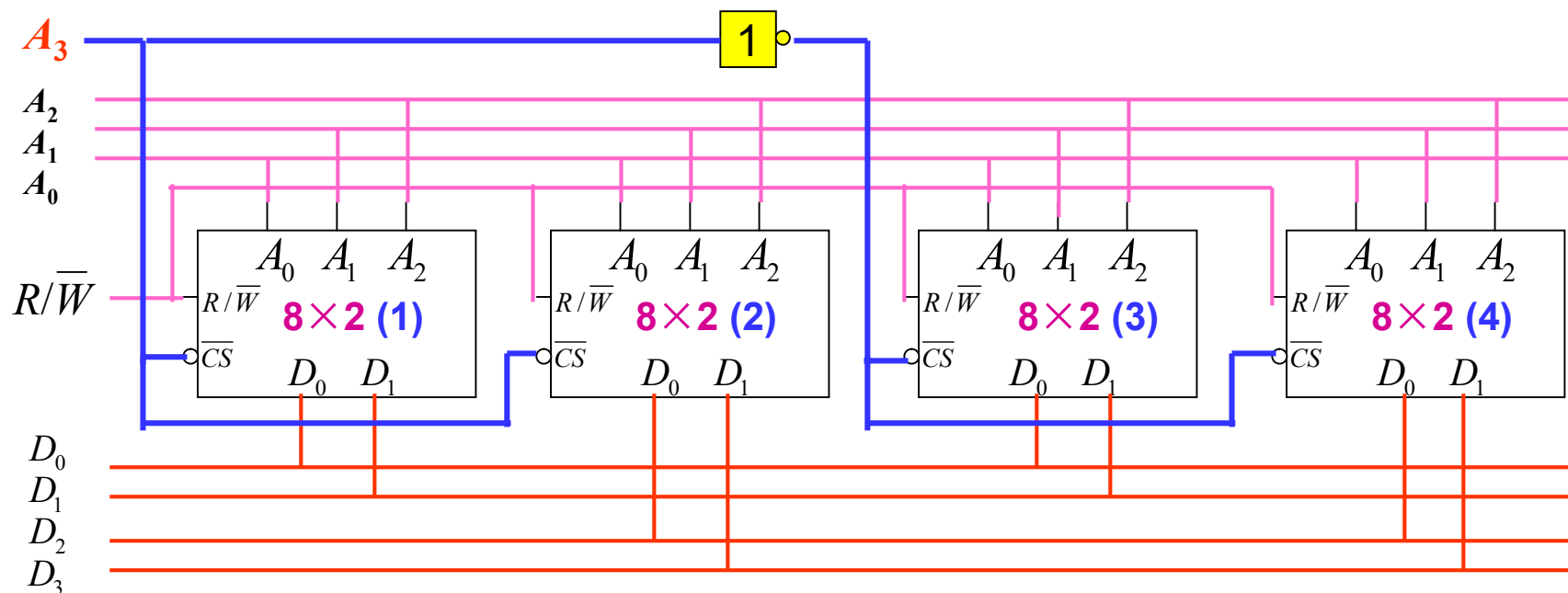
(4) **11000-11111**

练习

### 3. 字长和地址扩展

例 1. 将  $8 \times 2$  RAM 扩展为  $16 \times 4$  RAM.  $\frac{16 \times 4}{8 \times 2} = 4(8 \times 2 \text{ RAM})$

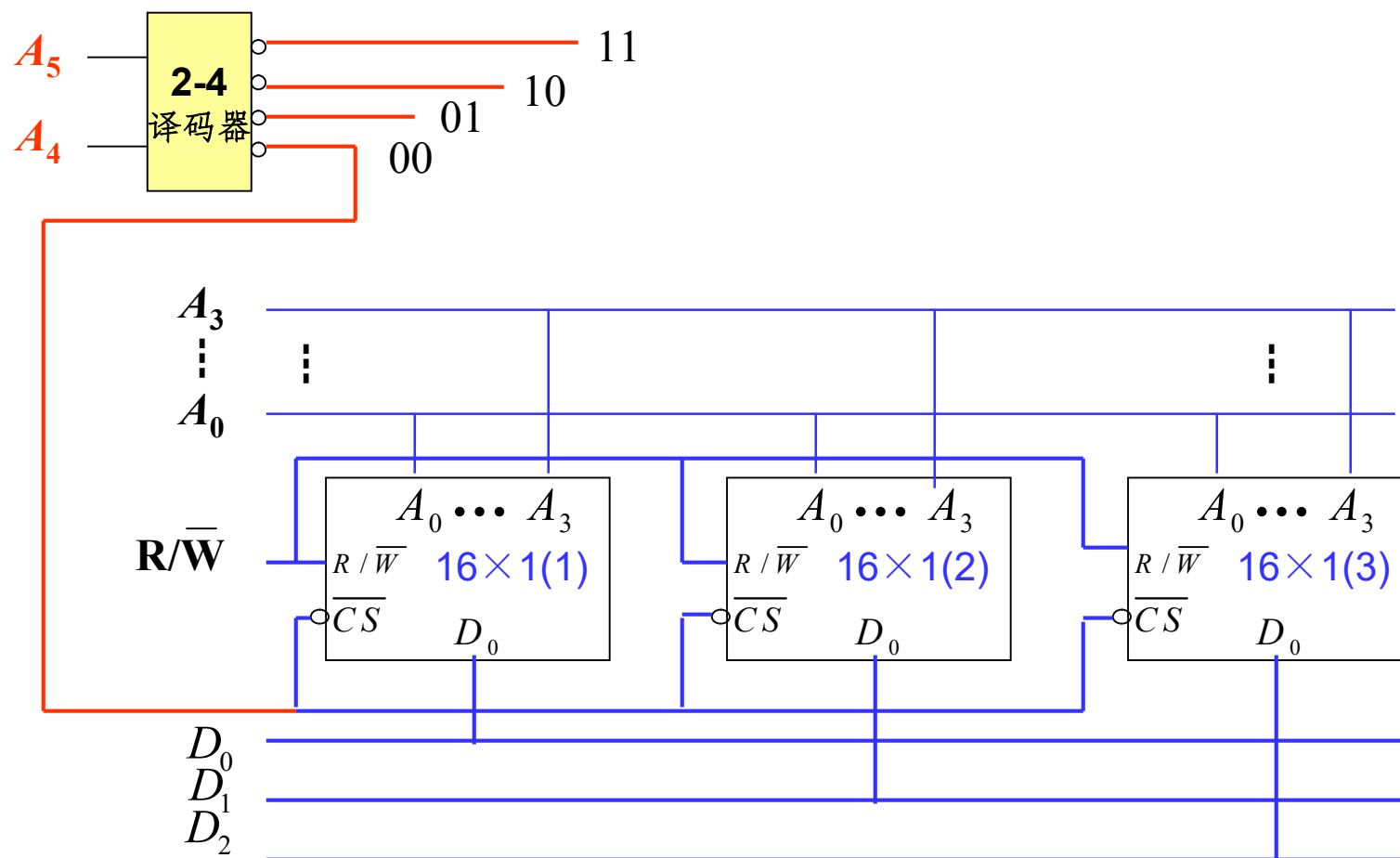
最好是先扩位，后扩字



地址范围  $\left\{ \begin{array}{l} (1)(2): 0000 - 0111 \\ (3)(4): 1000 - 1111 \end{array} \right.$

例 2. 将  $16 \times 1$  RAM 扩展为  $64 \times 3$  RAM。

$$\frac{64 \times 3}{16 \times 1} = 12 \quad (16 \times 1 \text{ RAM})$$



# 十六进制表示法

十六进制表示法有16个符号:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

基数是16

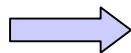
存储地址:

每4位二进制码表示一位十六进制码 后缀 H

例子: 0000 1011 1111  $\longrightarrow$  0BFH

地址: 0000 0000 ~ 0011 1111  $\longrightarrow$  00H - 3FH

地址: B00H ~ BFFH



$A_{11}A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0$

1 0 1 1 0 0 0 0 0 0 0 0

1 0 1 1 1 1 1 1 1 1 1 1



## § 10.3 ROM (只读存储器)

ROM一般由专用装置写入数据，数据一旦写入便不能随意改写，断电后，数据也不会丢失。

**ROM：组合逻辑电路**

地址 → 变量	}	真值表
输出数据 → 函数		

## 10.3.1 ROM 的分类

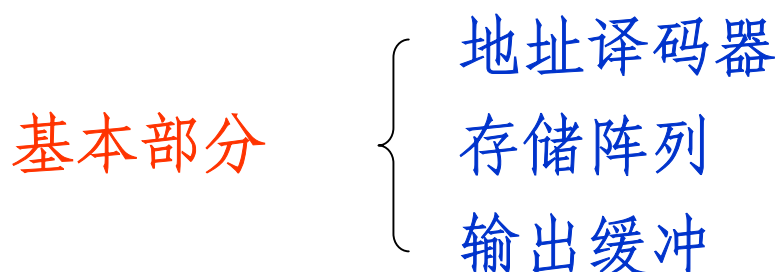
**ROM** { 固定ROM (Mask ROM)  
可编程ROM (Programmable ROM)

**PROM** { **PROM** (一次可编程ROM)  
**EPROM** (Erasable PROM 可擦除可编程)  
**EEPROM** (Electrically Erasable PROM  
电可擦除可编程)  
**Flash Memory** (快闪存储器)  
类似于E<sup>2</sup>ROM, 容量大

## 10.3.2 ROM结构

### 1. 固定ROM (掩膜 ROM)

在制造过程中数据被永久保存在存储器中。

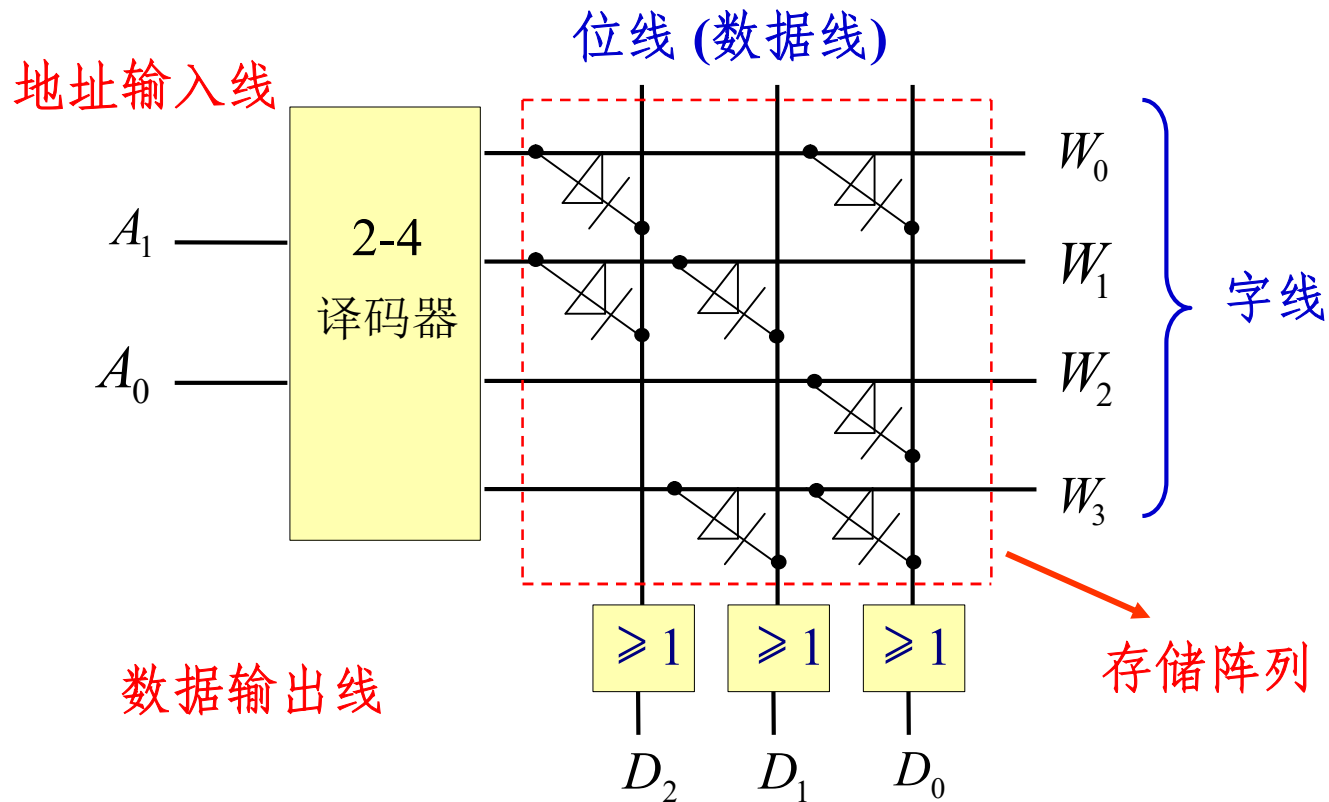


ROM的结构与RAM相似

地址译码器：与门阵列

$n$  - 位地址码输入

$2^n$  输出字线



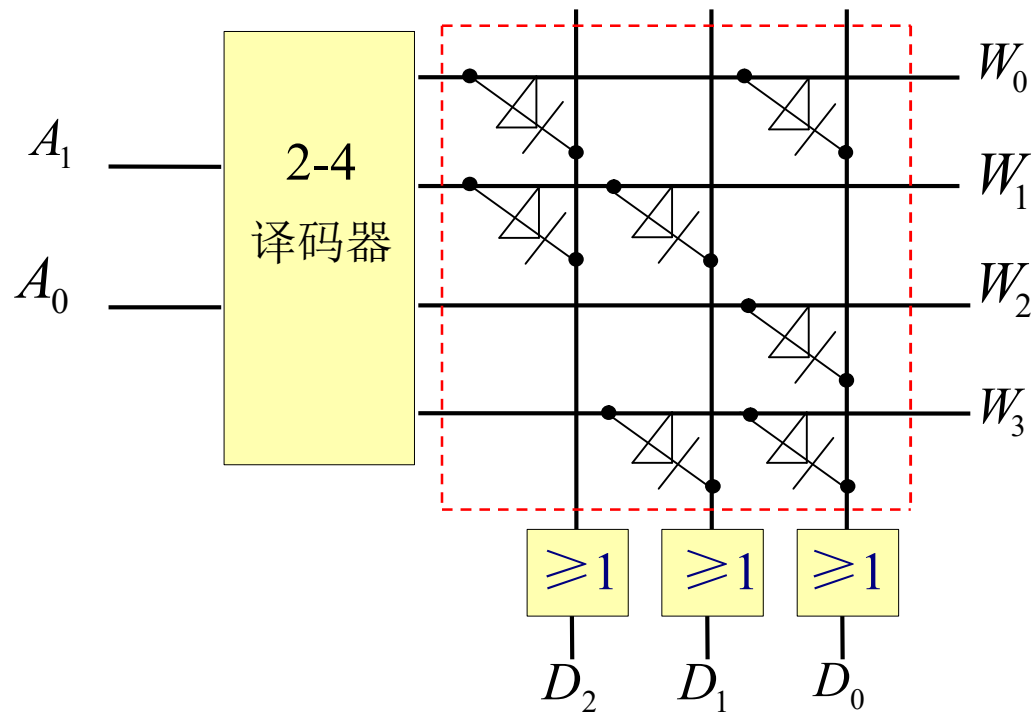
**地址译码器:** 把  $n$  条地址线译成  $2^n$  条字线  $W_0 \sim W_3$

**译码器** 高电平有效 一组地址只能使一条字线高电平，使二极管导通。

**存储阵列** 二极管，晶体管，MOS管

连接字线和位线

**输出缓冲** 或门, 3 位 ( $D_2 D_1 D_0$ )



当  $A_1A_0=00$ ,  $W_0=1$ , 二极管导通,  
输出  $D_2D_1D_0=101$

按上图

ROM 实现:

$$D_2 = \overline{A_1}\overline{A_0} + \overline{A_1}A_0 = \overline{A_1}$$

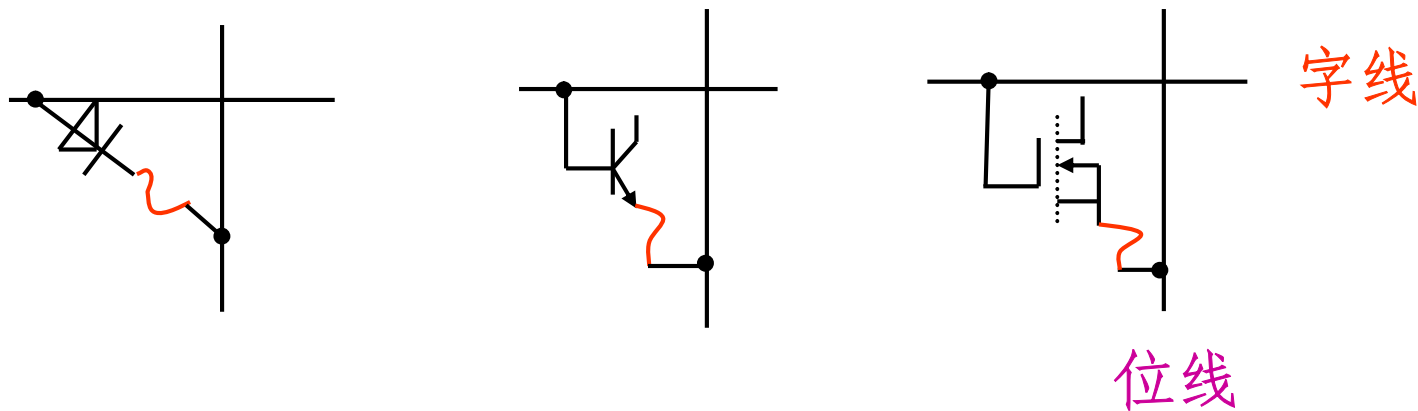
$$D_1 = \overline{A_1}A_0 + A_1A_0 = A_0$$

$$D_0 = \overline{A_1}\overline{A_0} + A_1\overline{A_0} + A_1A_0 = \overline{A_0} + A_1$$

ROM实现的是标准与或式

## 2. PROM (一次可编程ROM):

每个存储单元 { 二极管或电晶体或MOS管  
熔丝



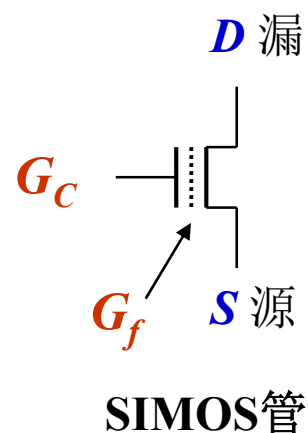
连接1，熔断0：出厂时全连(1)，用户根据需要将需要存入0的单元上的熔丝熔断

**PROM**一旦进行了编程，就不能再修改了。

### 3. EPROM

EPROM的存储单元多采用叠栅注入MOS管 (SIMOS: Stacked-gate Injection MOS)

叠层栅MOS管是一个N沟道增强型MOS管



两个栅极 { 控制栅  $G_C$ : 控制读出和写入  
浮置栅  $G_f$ : 长期保存注入电荷 (信息)

浮置栅上注入了电荷的SIMOS管相对于写入了1，未注入电荷相当于存入了0。

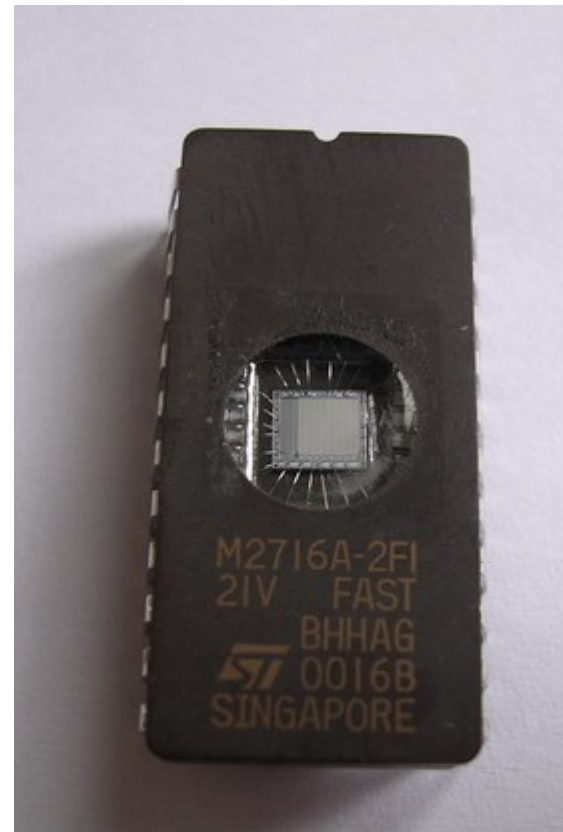
在浮置栅上注入电荷及使电荷消失的方法包括：

UVEPROM (通常只写EPROM)

( Ultra – Violet EPROM 紫外线擦除PROM)

EEPROM (Electrically EPROM 电擦除PROM,  $E^2$  ROM)

## 紫外线擦除器和EPROM 2716芯片





### 10.3.3 只读存储器应用

存储器(**ROM**主要功能), 也可以实现标准与或式逻辑功能。

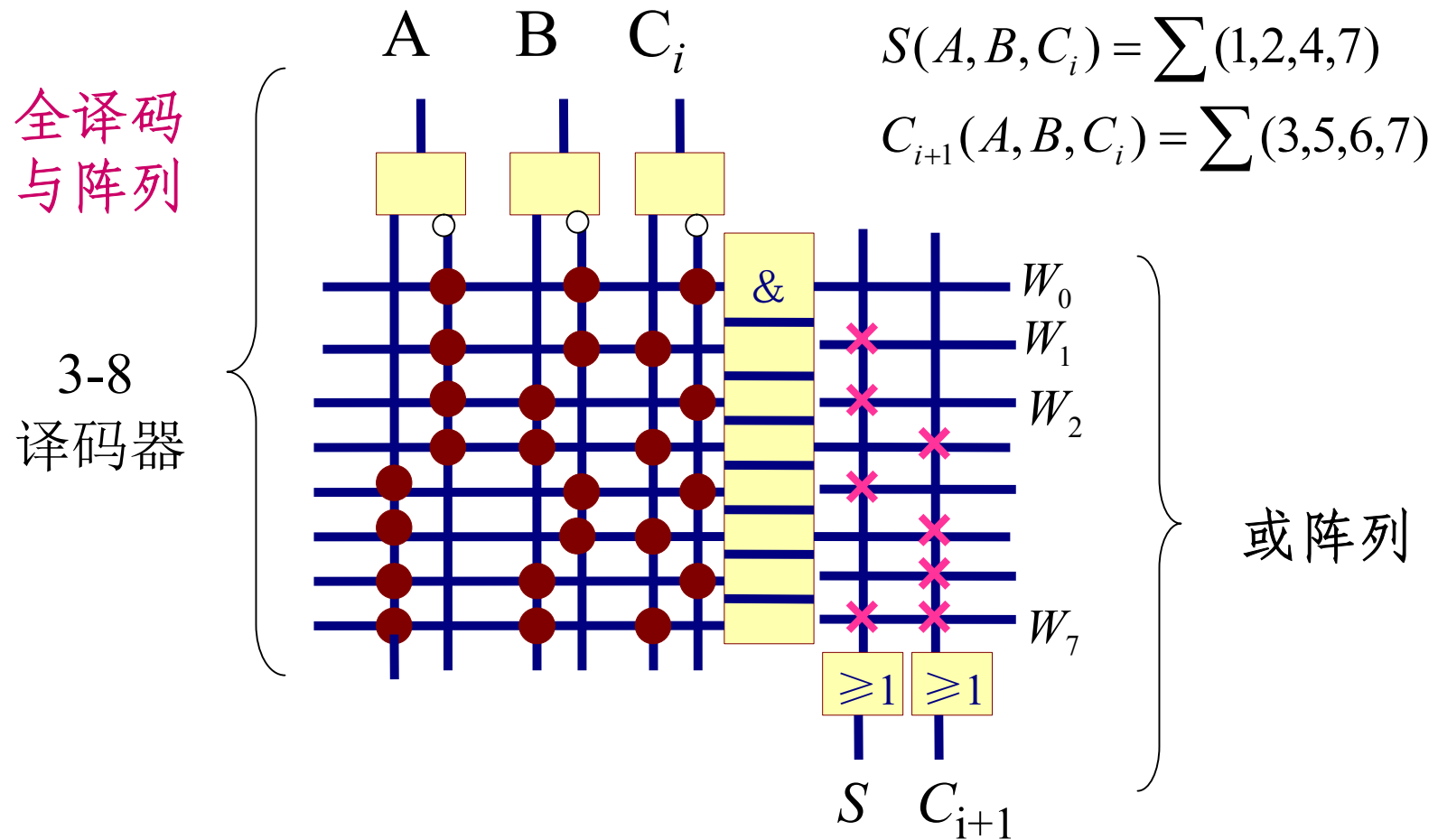
#### 1. 只读存储器实现组合逻辑功能函数

例 : 利用**ROM**实现一个全加器

解: 全加器真值表

**ROM** 与译码器相同,  
只能实现与或标准式

A	B	$C_i$	S	$C_{i+1}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



计算机编程，自动生成熔丝图，写入电路。

## 2. 用于函数运算表电路

例：用**ROM**构成能实现函数  $y = x^2$  的运算表电路， $x$  的取值范围为**0~15**的正整数。

解：（1）分析

变量 $x$ ：**0~15** 正整数，

→ **4 位二进制数**  $x = X_3X_2X_1X_0$

函数  $y = x^2$ ，

最大值 $y$ ： $15^2 = 225$ ，

→ **8 位二进制数**  $y = Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$

(2) 真值表

$X_3$	$X_2$	$X_1$	$X_0$	$Y_7$	$Y_6$	$Y_5$	$Y_4$	$Y_3$	$Y_2$	$Y_1$	$Y_0$	十进制数
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	0	0	1	0	0	4
0	0	1	1	0	0	0	0	1	0	0	1	9
0	1	0	0	0	0	0	1	0	0	0	0	16
0	1	0	1	0	0	0	1	1	0	0	1	25
0	1	1	0	0	0	1	0	0	1	0	0	36
0	1	1	1	0	0	1	1	0	0	0	1	49
1	0	0	0	0	1	0	0	0	0	0	0	64
1	0	0	1	0	1	0	1	0	0	0	1	81
1	0	1	0	0	1	1	0	0	1	0	0	100
1	0	1	1	0	1	1	1	1	0	0	1	121
1	1	0	0	1	0	0	1	0	0	0	0	144
1	1	0	1	1	0	1	0	1	0	0	1	169
1	1	1	0	1	1	0	0	0	1	0	0	196
1	1	1	1	1	1	1	0	0	0	0	1	225

### (3) 写标准与或表达式

$$Y_7 = m_{12} + m_{13} + m_{14} + m_{15}$$

$$Y_6 = m_8 + m_9 + m_{10} + m_{11} + m_{14} + m_{15}$$

$$Y_5 = m_6 + m_7 + m_{10} + m_{11} + m_{13} + m_{15}$$

$$Y_4 = m_4 + m_5 + m_7 + m_9 + m_{11} + m_{12}$$

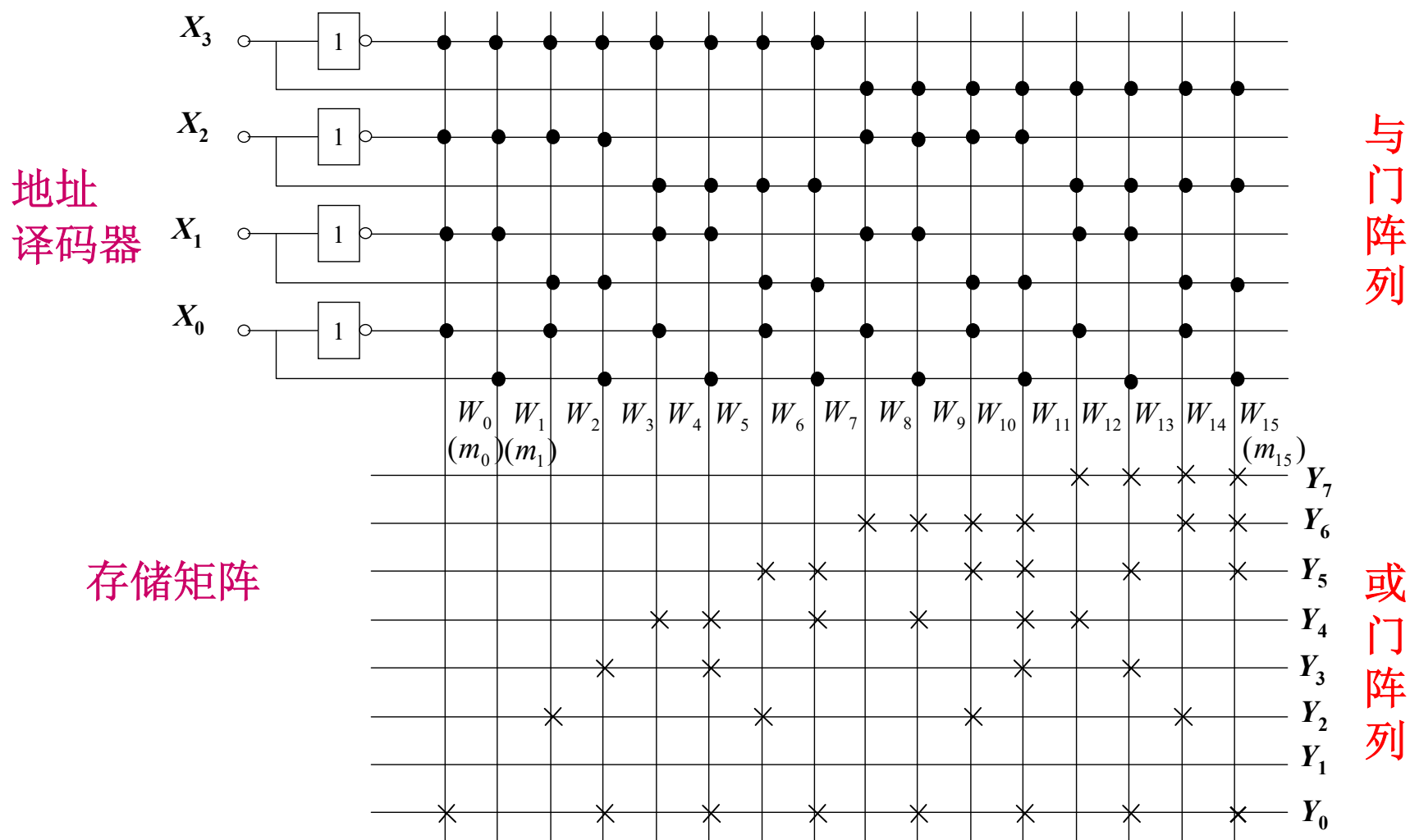
$$Y_3 = m_3 + m_5 + m_{11} + m_{13}$$

$$Y_2 = m_2 + m_6 + m_{10} + m_{14}$$

$$Y_1 = 0$$

$$Y_0 = m_1 + m_3 + m_5 + m_7 + m_9 + m_{11} + m_{13} + m_{15}$$

# (4) 画ROM存储矩阵节点连接图



## § 10.4 可编程逻辑器件

### 10.4.1 可编程逻辑器件简介

可编程逻辑器件，简称**PLD**

现场可编程逻辑阵列 **FPLA**

可编程阵列逻辑 **PAL**

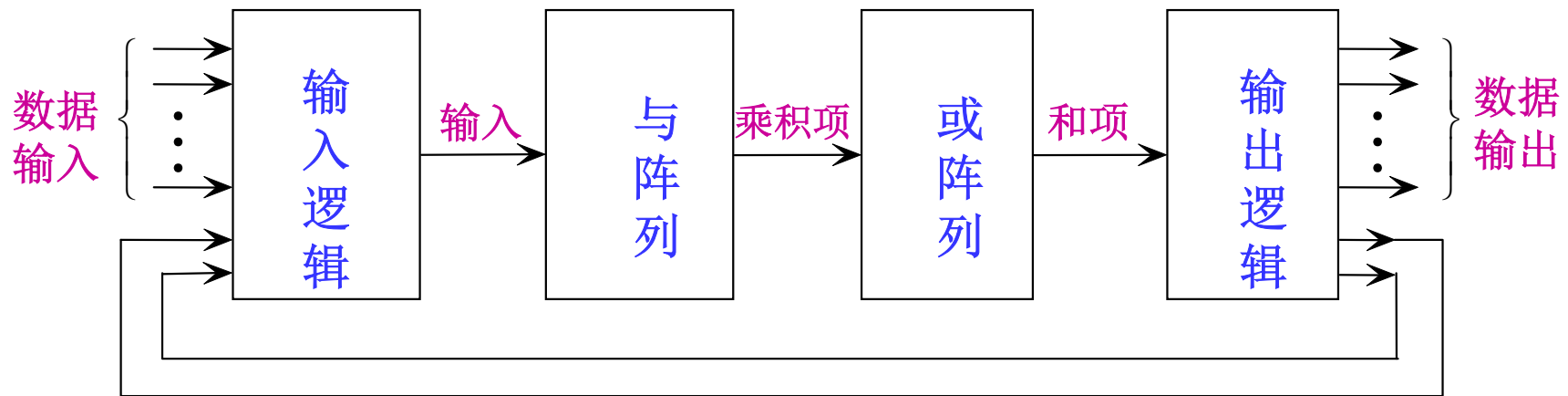
通用阵列逻辑 **GAL**

复杂可编程逻辑器件 **CPLD**

现场可编程门阵列 **FPGA**

## 10.4.2 可编程逻辑器件的基本结构和电路表示方法

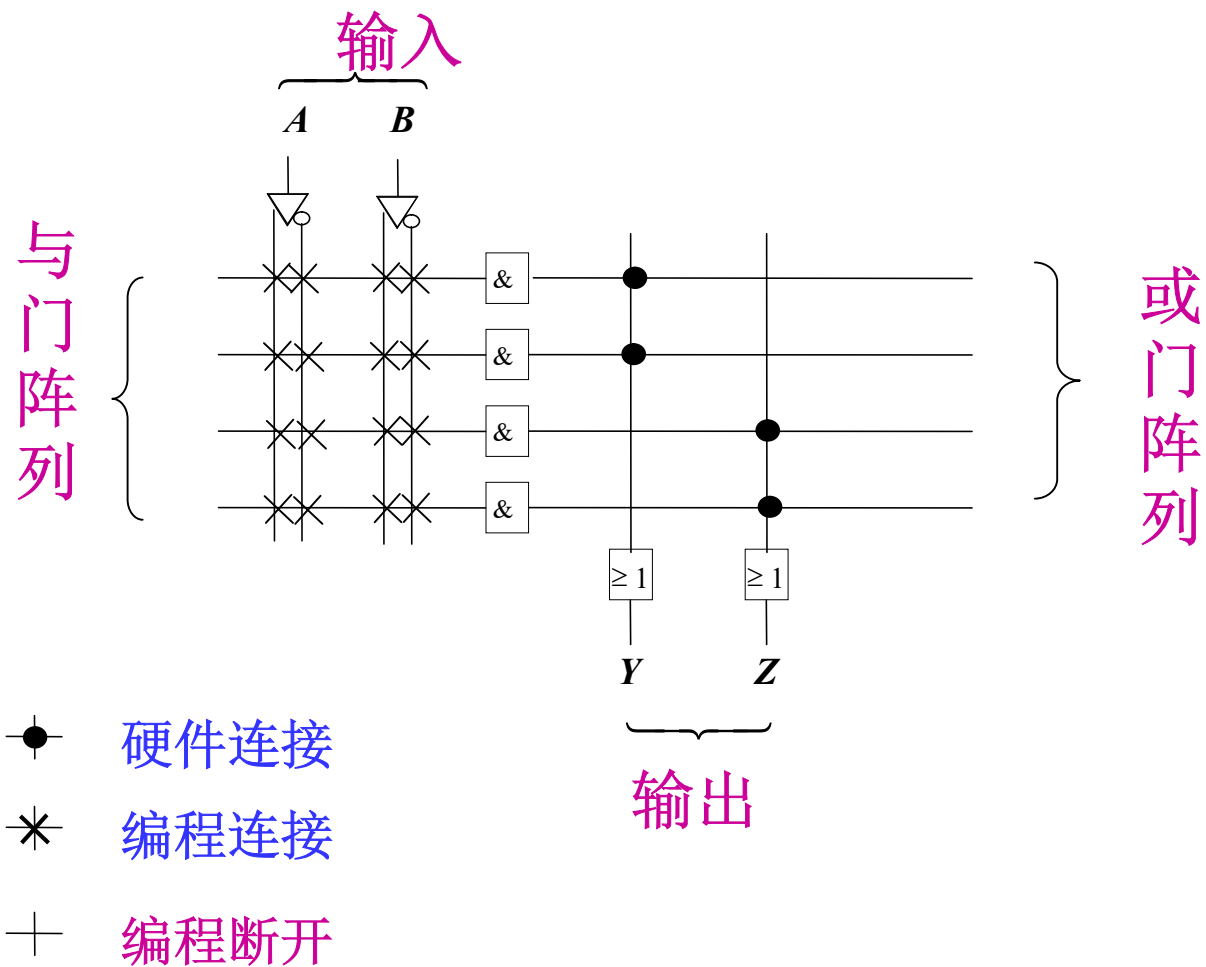
### 1. 可编程逻辑器件的基础结构



**PLD** { 与阵列可编程  
或阵列可编程  
与、或阵列都可以编程



## 2. 可编程逻辑器件的电路描述



## 10.4.3 可编程阵列逻辑 ( PAL )

### 1. PAL电路的基本结构

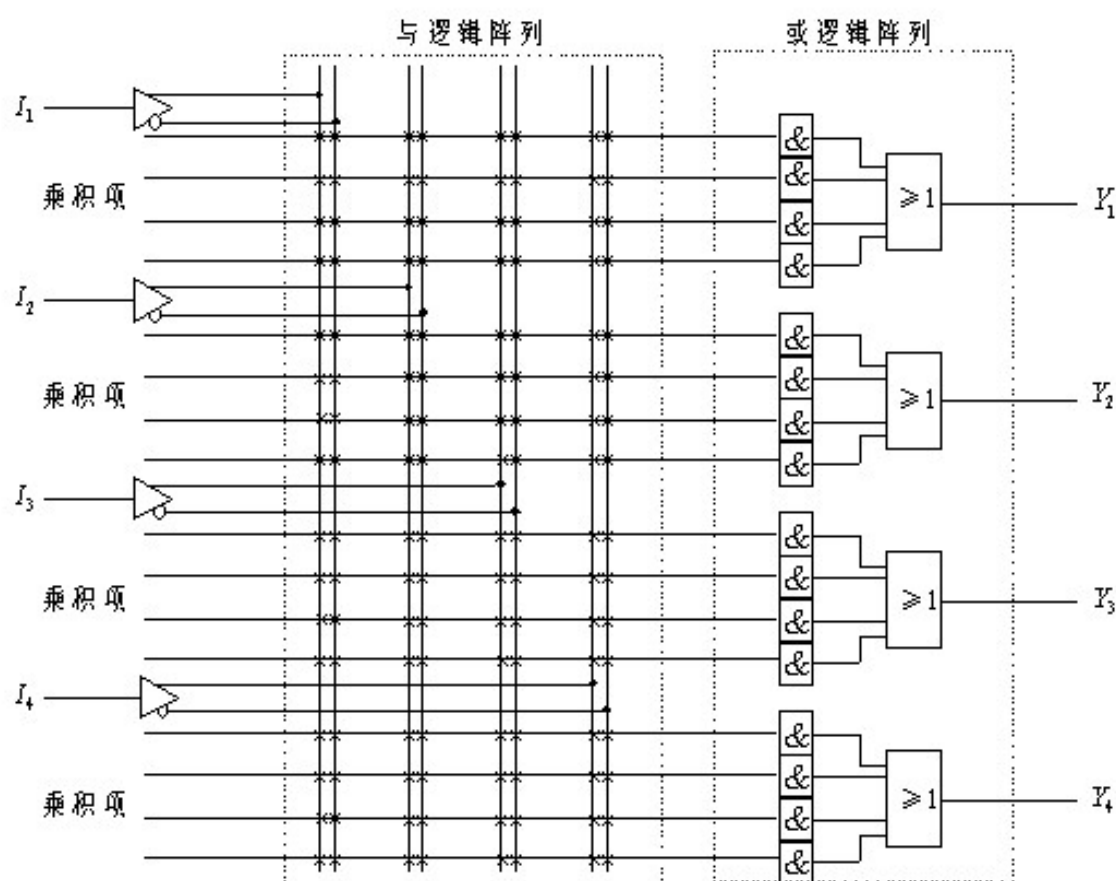
双极型工艺制  
作，熔丝编程

组成：

可编程的与逻辑阵列

固定的或逻辑阵列

输出电路



## 2. PAL 器件的应用

用PAL实现逻辑函数  $F_1 = A \oplus B \oplus C$  及其反函数  $F_2$ 。

用“×”表示编程连接，画出简化内部结构。

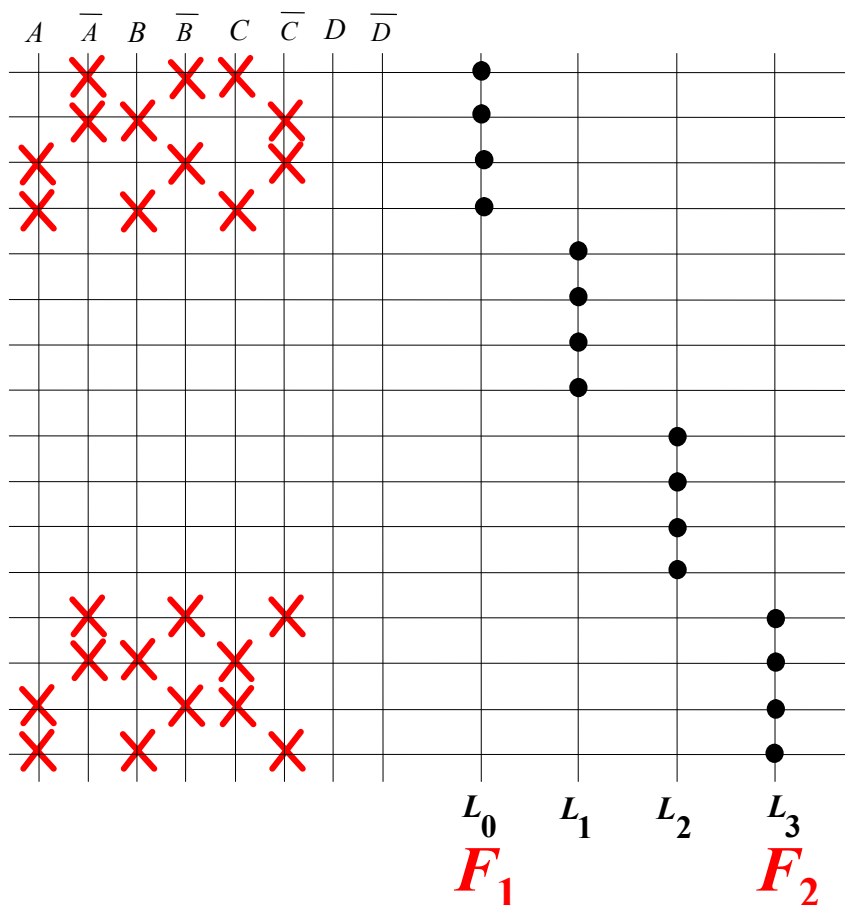
解：

PAL是一种与阵列可编程的PLD器件，按逻辑函数与或表达式中乘积项可确定与阵列的编程连接。实际应用中，用编程程序，输入函数表达式，编程程序会自动对PAL进行连接。

$$F_1(A, B, C) = \sum m(1, 2, 4, 7)$$

$$F_2(A, B, C) = \sum m(0, 3, 5, 6)$$

设 $L_0$ 输出 $F_1$ ， $L_3$ 输出 $F_2$ ，编程后内部结构如右图。



## 10.4.4 GAL (Generic Array Logic) 通用阵列逻辑电路

1985年由LATTICE公司推出，采用电可擦除的CMOS (E<sup>2</sup>CMOS) 制作；输出设置了可编程的输出逻辑宏单元OLMC (Output Logic Macro Cell)，通过编程可以将OLMC设置成不同的工作状态。

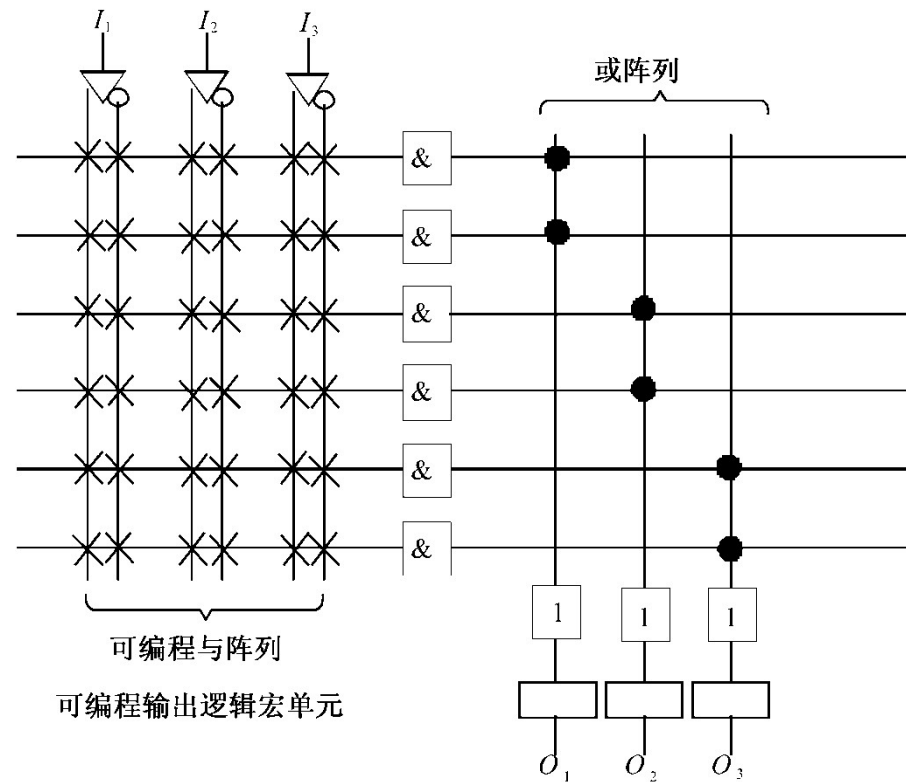
### 1. GAL基本结构

组成

可编程与阵列

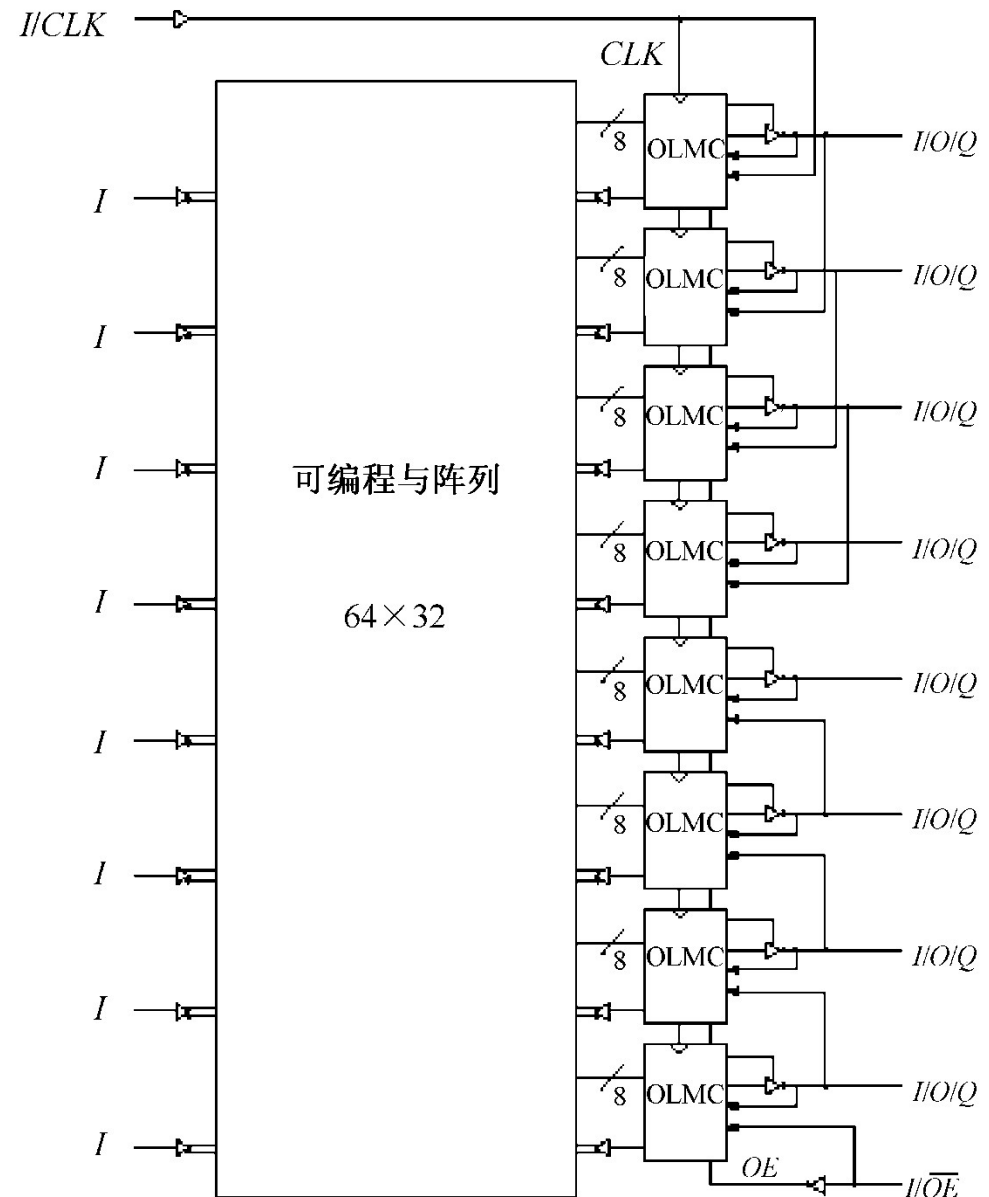
不可编程或阵列

可编程输出逻辑  
宏单元OLMC



## 2. GAL16V8的电路构成

可编程与阵列 ( $64 \times 32$ )、  
输入缓冲器、  
输出三态缓冲器、  
输出反馈/输入缓冲器、  
输出逻辑宏单元  
输出使能缓冲器 (OE)



## 小 结

- 随机存取存储**RAM**
  - 地址译码器、**RAM**扩展
- 只读存储器**ROM**
  - 分类、基本原理及实现逻辑函数
- 可编程逻辑器件

课后作业:

**10. 4**

**10. 5**

**10. 6**

**10. 8**

**10. 18**