7. 半导体存储器

7.1 存储器与寄存器

7.2 存储器的分类

7.3 随机存储器

7.4 只读存储器

一、半导体存储器与寄存器

存储器是数字系统用来存储信息的存储部件，它和**寄存器**的区别主要在于：

① 存储时间

- **寄存器**：短时间存放，犹如车站的小件寄存处
- **存储器**：长时间存储，犹如工厂的仓库

② 工作速度

- **寄存器**：速度较快
- **存储器**：速度较慢

③ 容量与价格

- 寄存器：集成度低，容量小，价格高
- 存储器：集成度高，容量大，价格低

④ 应用

- 寄存器：常用来临时存放少量数据，如CPU中的寄存器用来存储*操作数*和*运算中间结果*。
- 存储器：常用来存储*程序*、*数据*和*数表*。

二、存储器的分类

根据信息存取方式的不同，可分为：

- 半导体存储器
  - 只读存储器 ROM　Read – Only Memory
    - 固定ROM（不可编程）
    - PROM（可一次性编程）
    - EPROM (可紫外线擦除, 可多次编程)
    - E^2PROM（可电擦除, 可多次编程）
    - Flash Memory（可电擦除, 可多次编程）
  - 顺序存取存储器 SAM　Sequential Access Memory
    - FIFO（First In First Out 先入先出）
    - FILO（First In Last Out先入后出）
  - 随机存取存储器 RAM　Random – Access Memory
    - SRAM（Static 静态的）
    - DRAM（Dynamic 动态的）

1、随机存储器 RAM（Random Access Memory）

(1) 特点

A、随机读写　B、断电后信息丢失

(2) 用途

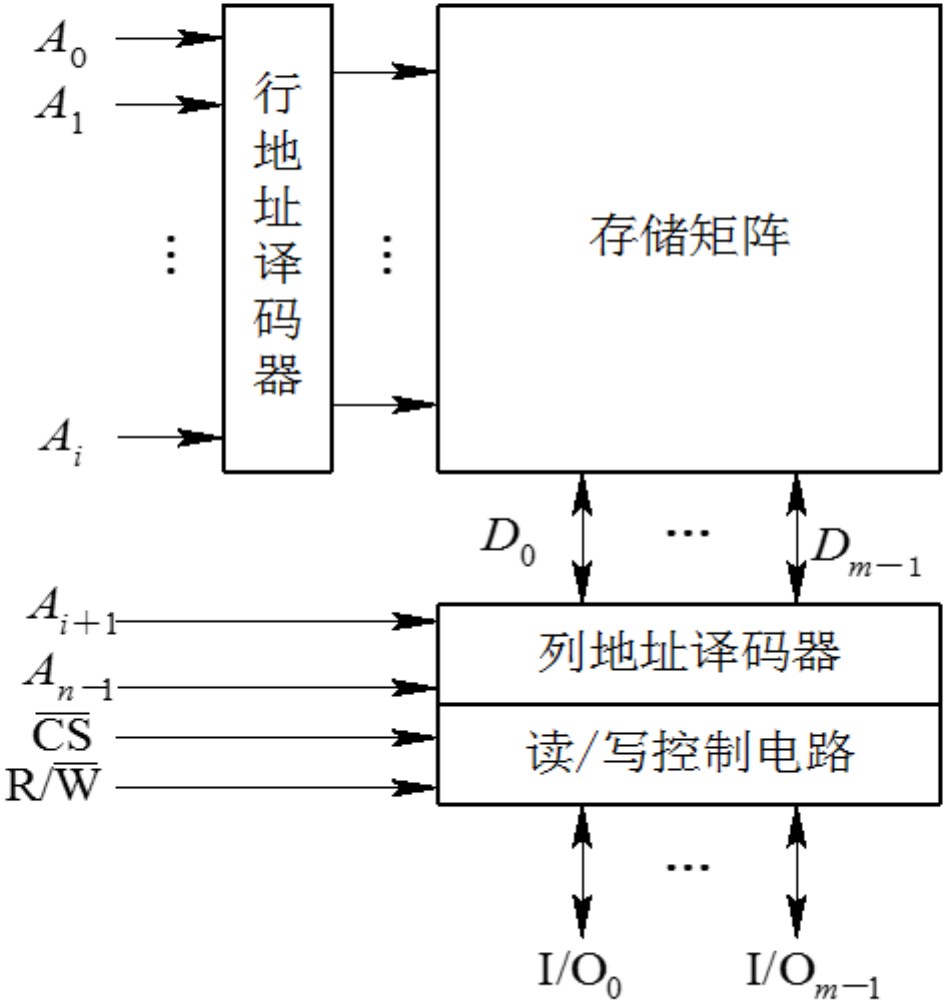
用于需要经常随机修改存储单元内容的场合，用作*数据存储器*。

(3) 分类

- **DRAM**：以MOS管栅、源极间的寄生电容存储信息，需要刷新。
- **SRAM**：以双稳态触发器存储信息，集成度低，价格高，不需要刷新。

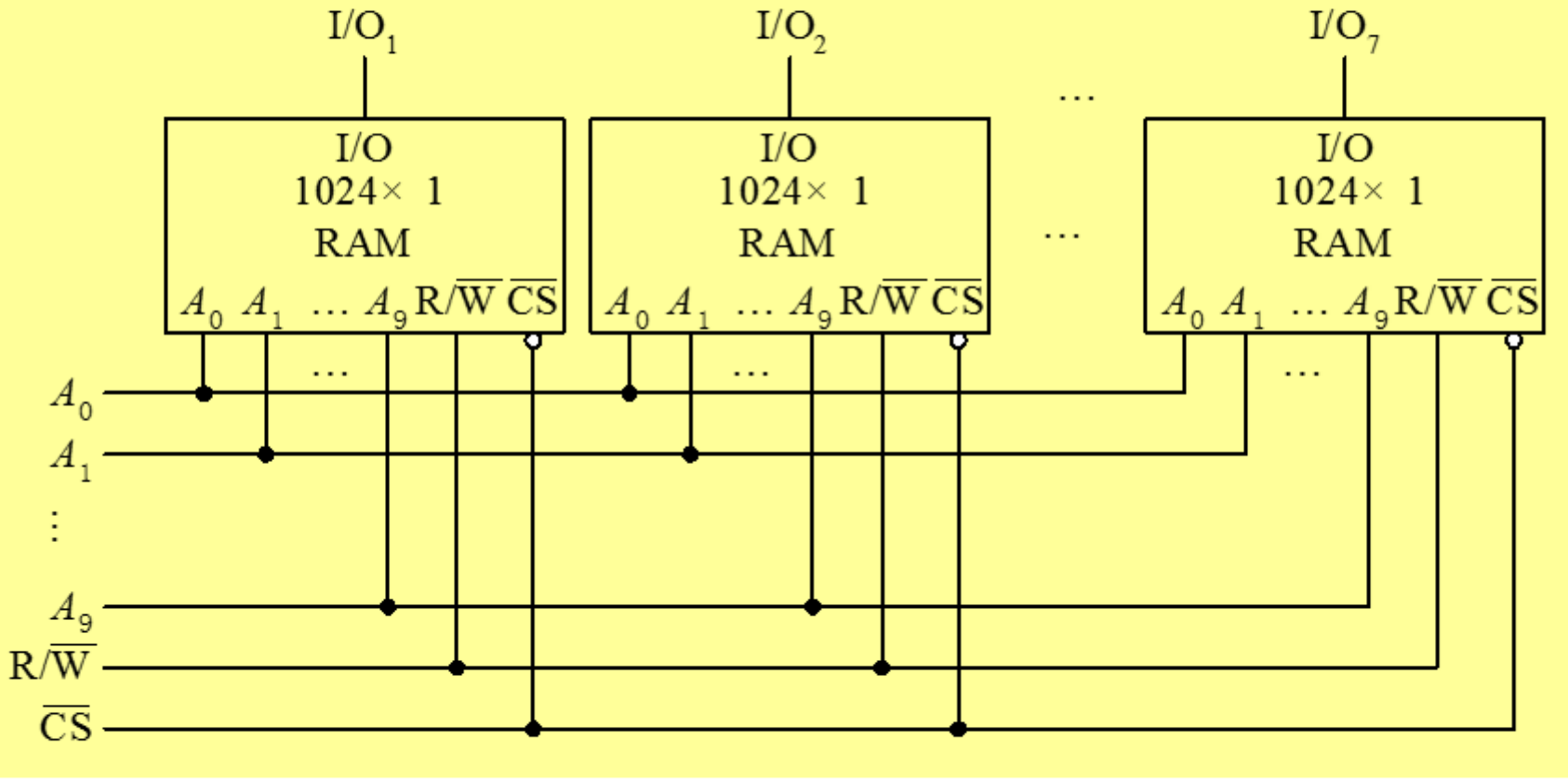
2. 随机存取存储器（RAM）

随机存储器SRAM的基本结构

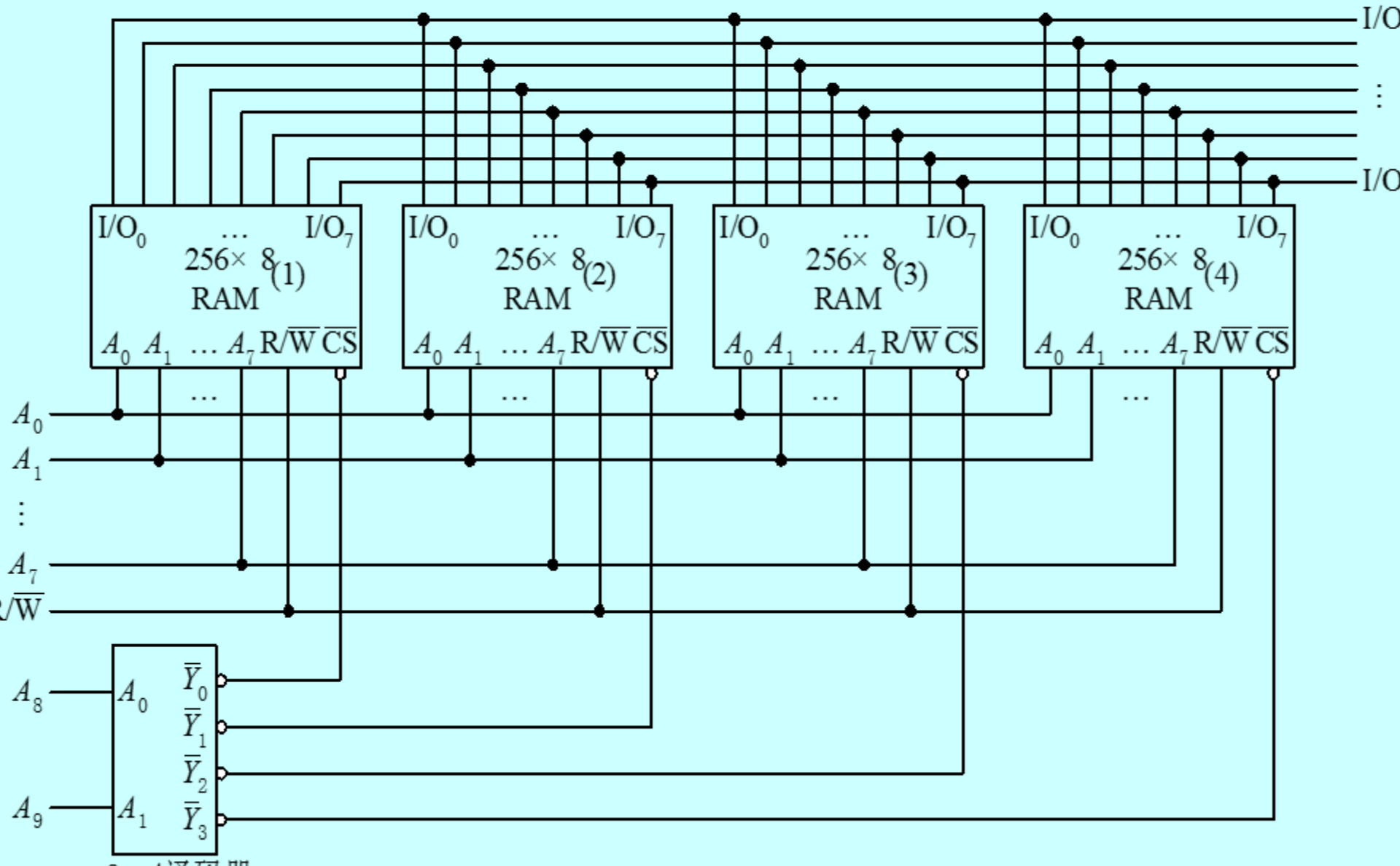


3. 存储器容量的扩展

(1) 位(数据位)的扩展



(2) 字(地址数)的扩展


I/O
I/O
I/O0 … I/O7
256× 8
RAM
(1)
(2)
(3)
(4)
A0 A1 … A7 R/W CS
A0
A1
A7
R/W
A8
A9
Y0
Y1
Y2
Y3

2、顺序存取存储器 SAM （Sequential Access Memory）

(1) 特点

A、顺序读写　B、断电后信息丢失

(2) 用途

用于需要顺序读写存储单元内容的场合，如在CPU中用作*堆栈*，以保护程序断点和寄存器内容。

3、只读存储器 ROM （Read Only Memory）

（1） 特点

A、正常工作时，只能读出，不能写入；

B、断电后信息不丢失

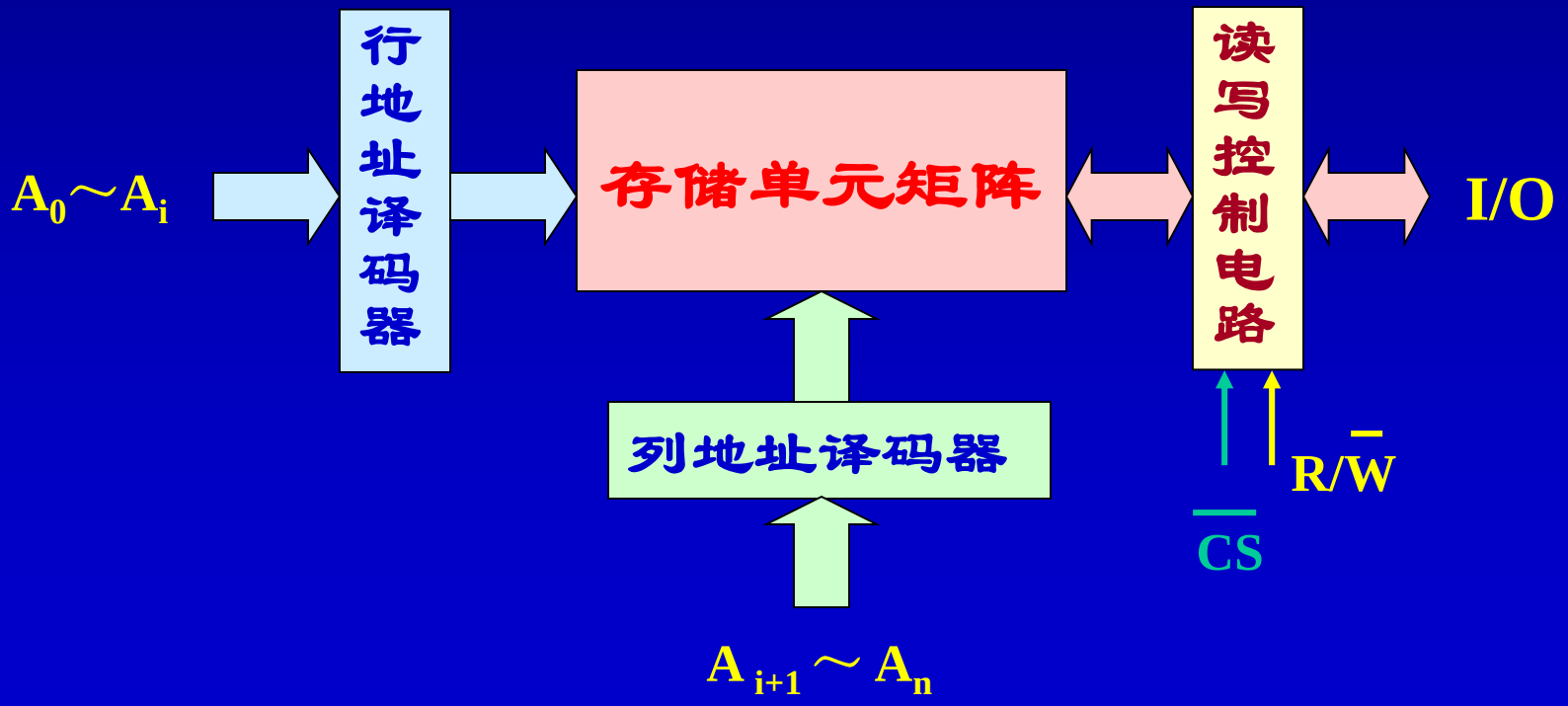
（2） 用途

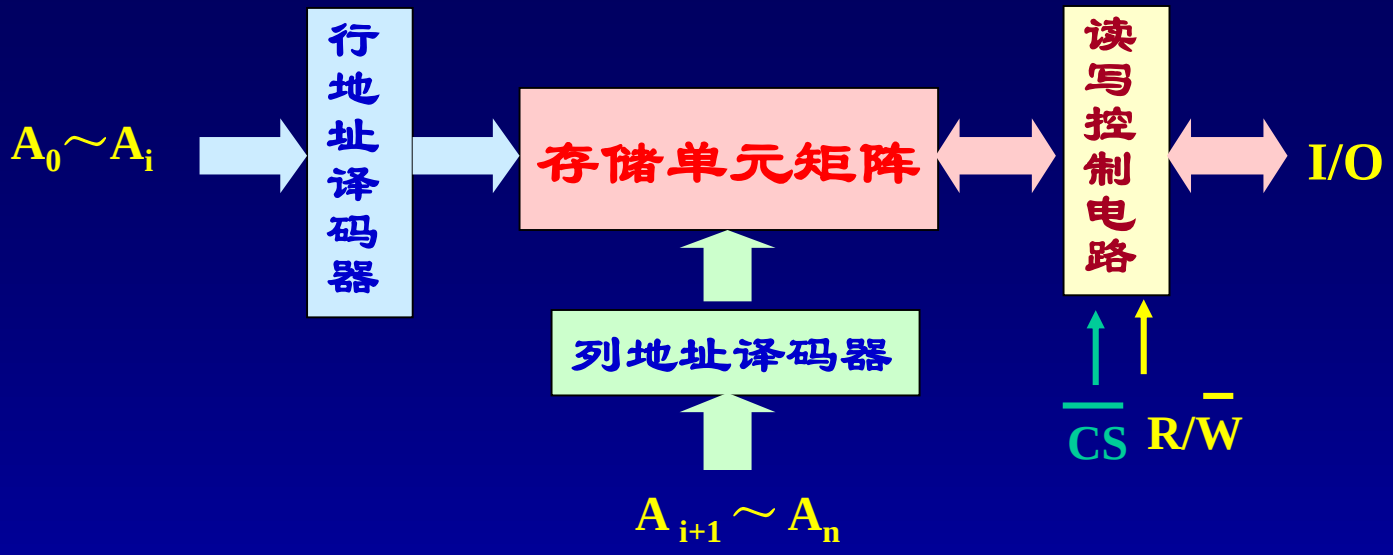
用于工作时不需要修改存储内容、断电后信息不能丢失的场合，如在计算机中用作*程序存储器*和*常数表存储器*。

三、静态随机存储器

SRAM 电路通常由 存储单元矩阵 、 地址译码器 、 读写控制电路 三部分构成 。

结构框图

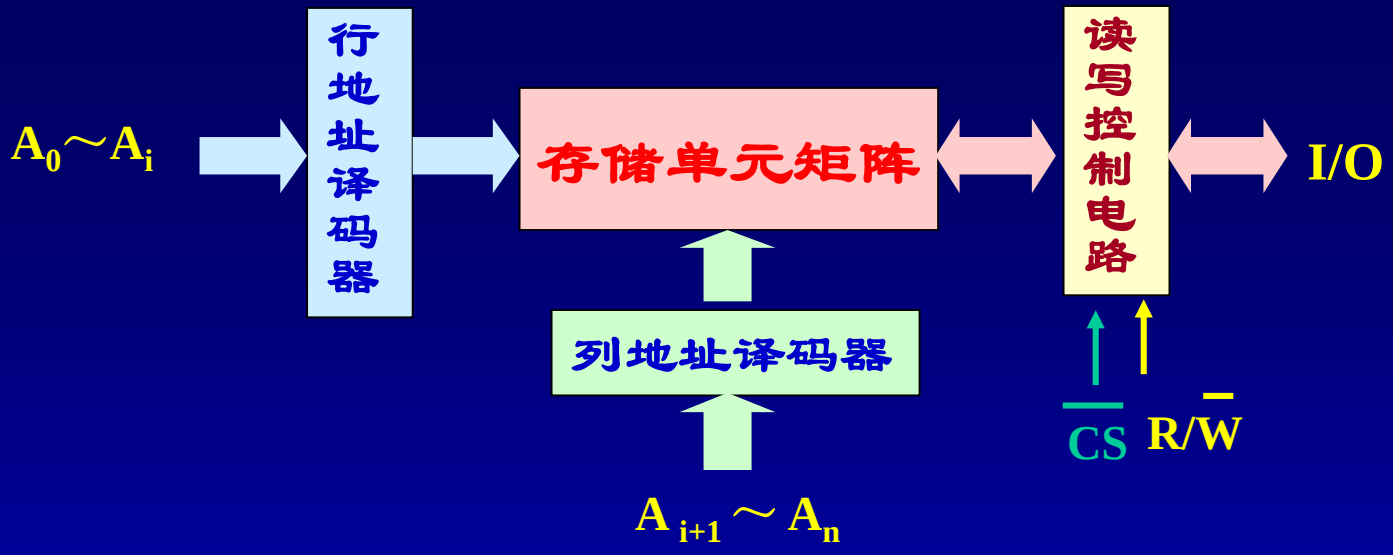




A、存储单元矩阵

是RAM的核心，每个存储单元中存放着由若干位二进制数构成的“字”，“字”的二进制位数称为“字长”。

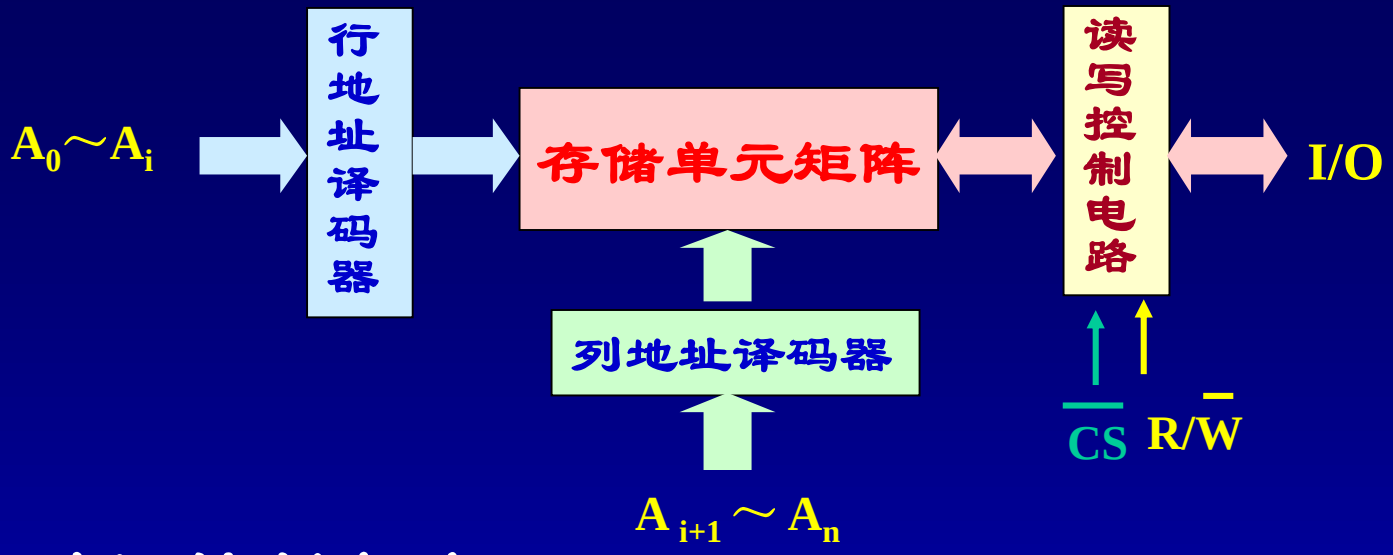
输入不同的地址码，可以选中不同的存储单元，并对该单元进行读写操作。



B、地址译码器

存储器容量很大，地址线位数较多，直接对地址译码，地址译码器非常庞大。将地址码分为行地址和列地址两部分，用行地址译码器、列地址译码器分别译码（二维译码）。

只有同时被行地址译码器和列地址译码器选中的存储单元，才能进行读写操作。



C、读写控制电路

① 读写控制

R/$\overline{W}$ {
- 1，表示从选中存储单元读取信息
- 0，表示向选中存储单元写入信息

② 片选控制

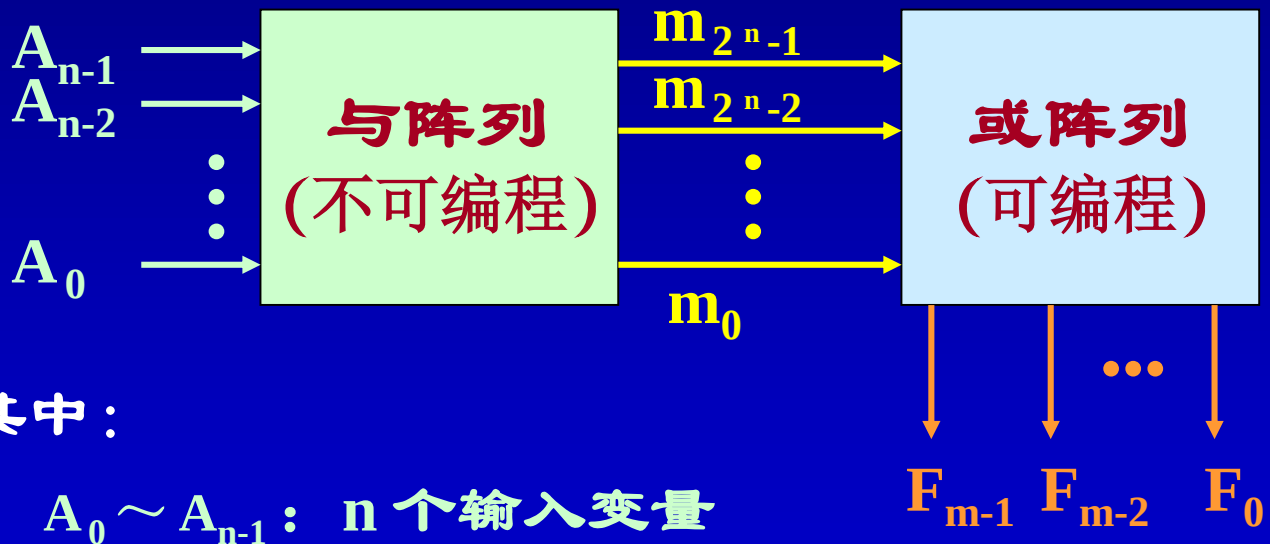
$\overline{CS}$ {
- 0，芯片被选中，可对选中单元进行读写操作
- 1，芯片未被选中，数据线处于高阻状态

四、只读存储器 ROM

1、结构

ROM 的主体是一个不可编程的与阵列和一个可编程的或阵列

① 与－或阵列结构



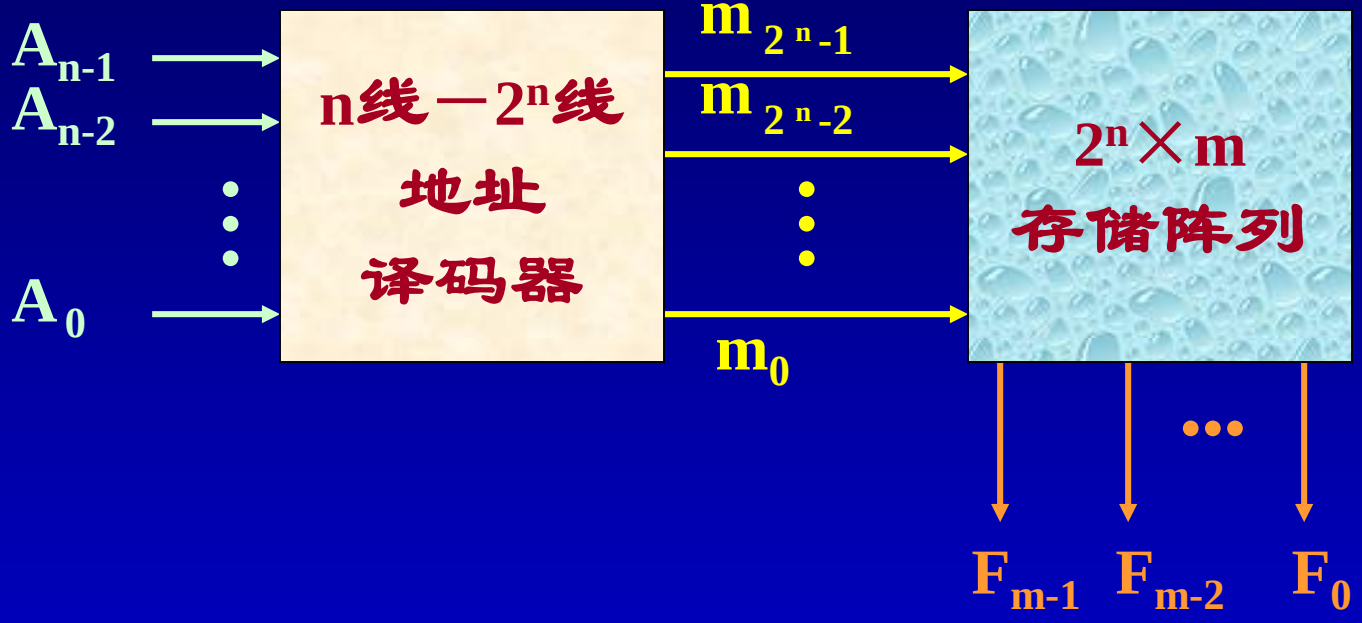
其中：

$A_0 \sim A_{n-1}$：n 个输入变量

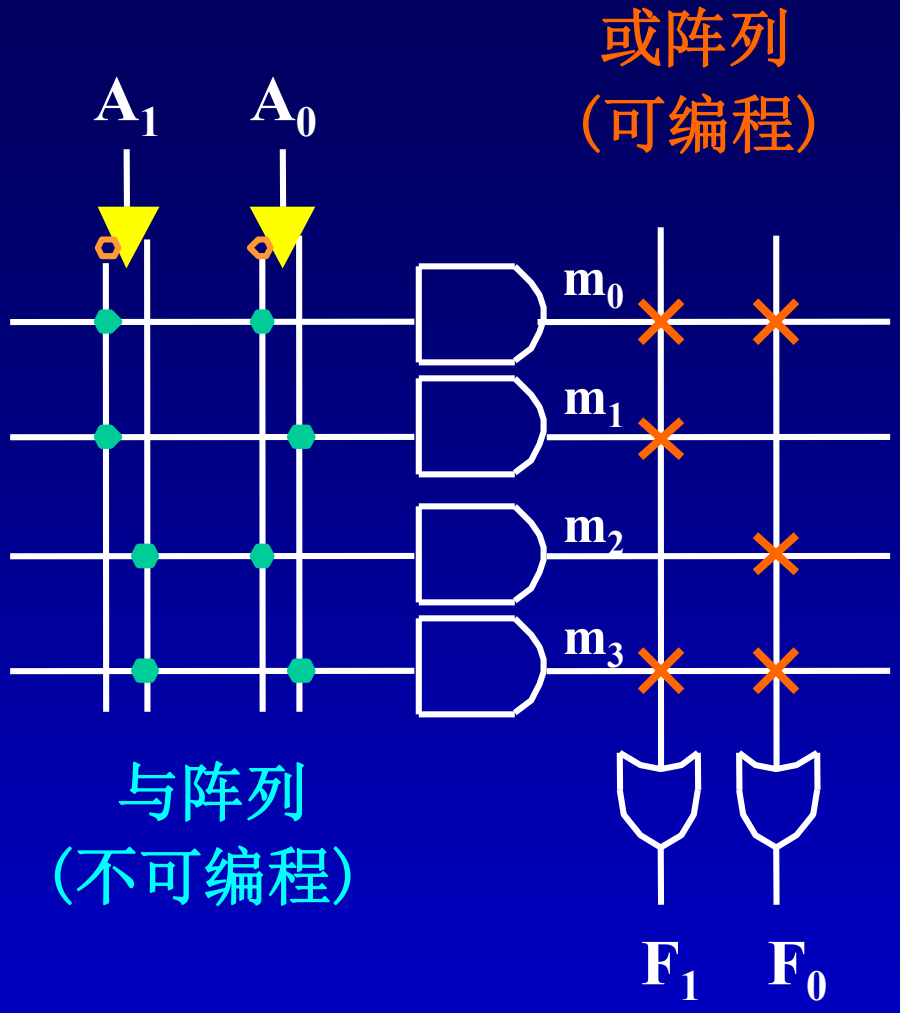
$m_0 \sim m_{2^n-1}$：2^n 个最小项

$F_0 \sim F_{m-1}$：m 个输出函数

② 存储器结构

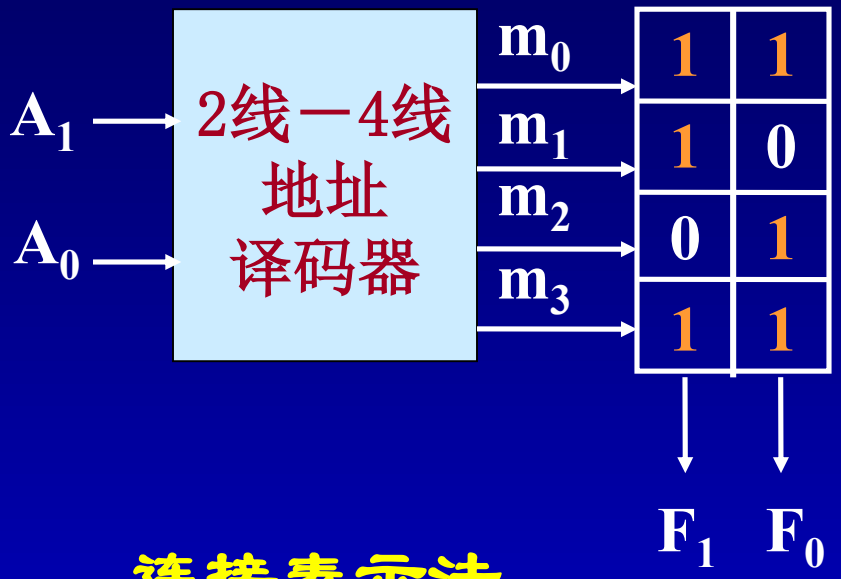


由于任何逻辑函数都可以写成最小项之和的形式，因此，只要合理地对或阵列进行编程，ROM的这种结构可以实现任意m个n变量逻辑函数。



$F_1 = \Sigma m\ (0, 1, 3)$

$F_0 = \Sigma m\ (0, 2, 3)$



连接表示法

固定连接　　编程连接　　不连接

2、ROM 的分类

（1）按制造工艺分类

- 双极型
- MOS型：功耗低、集成度高，大容量ROM都采用MOS工艺制造。

（2）按编程工艺和擦除方法分类

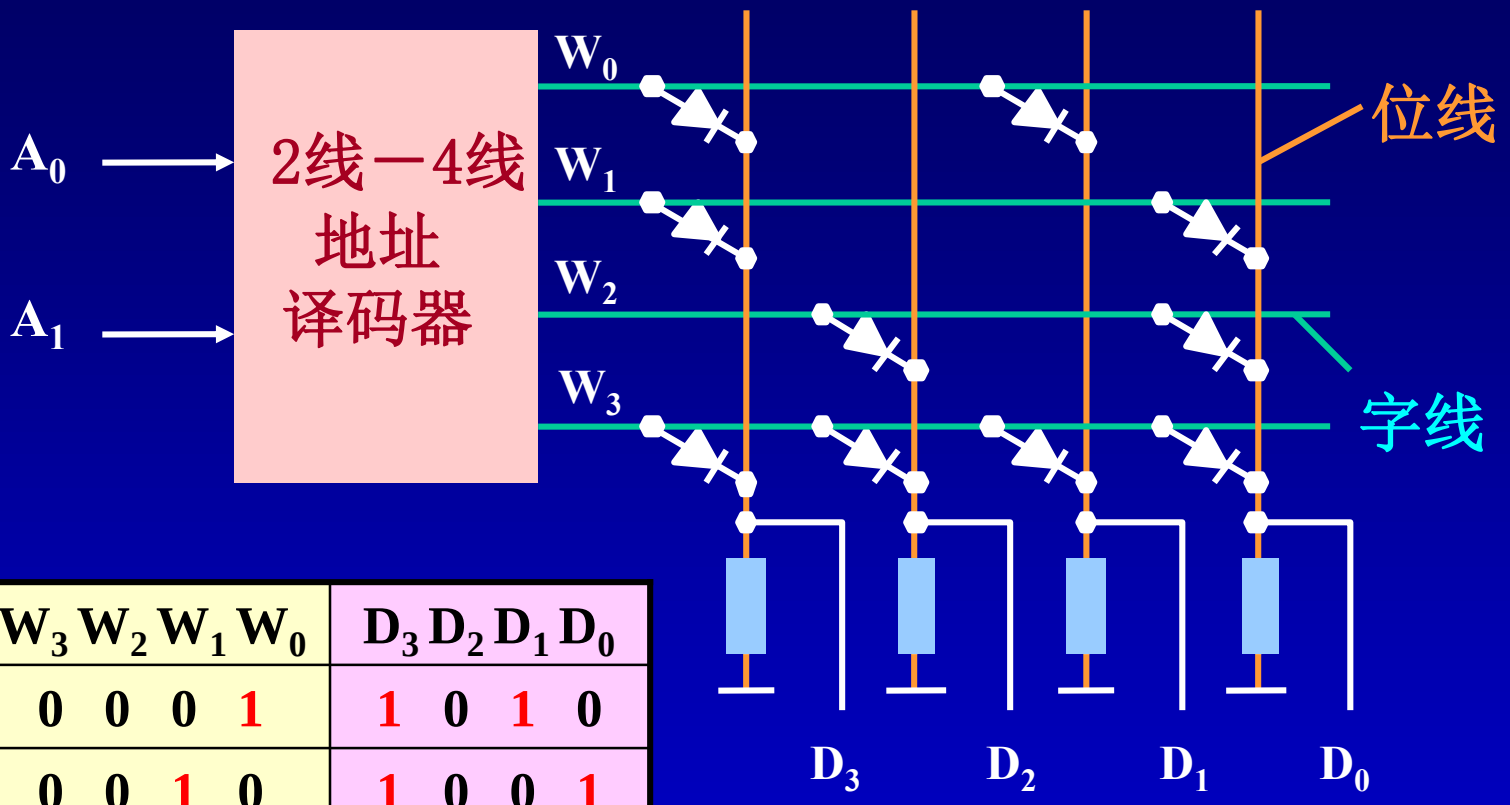
A、固定ROM（掩模ROM）

在生产的最后一道工序，将用户要求的数据“写入”存储器 —— 掩模。

特点：出厂后不能被修改，不可编程

应用：量大、成熟产品

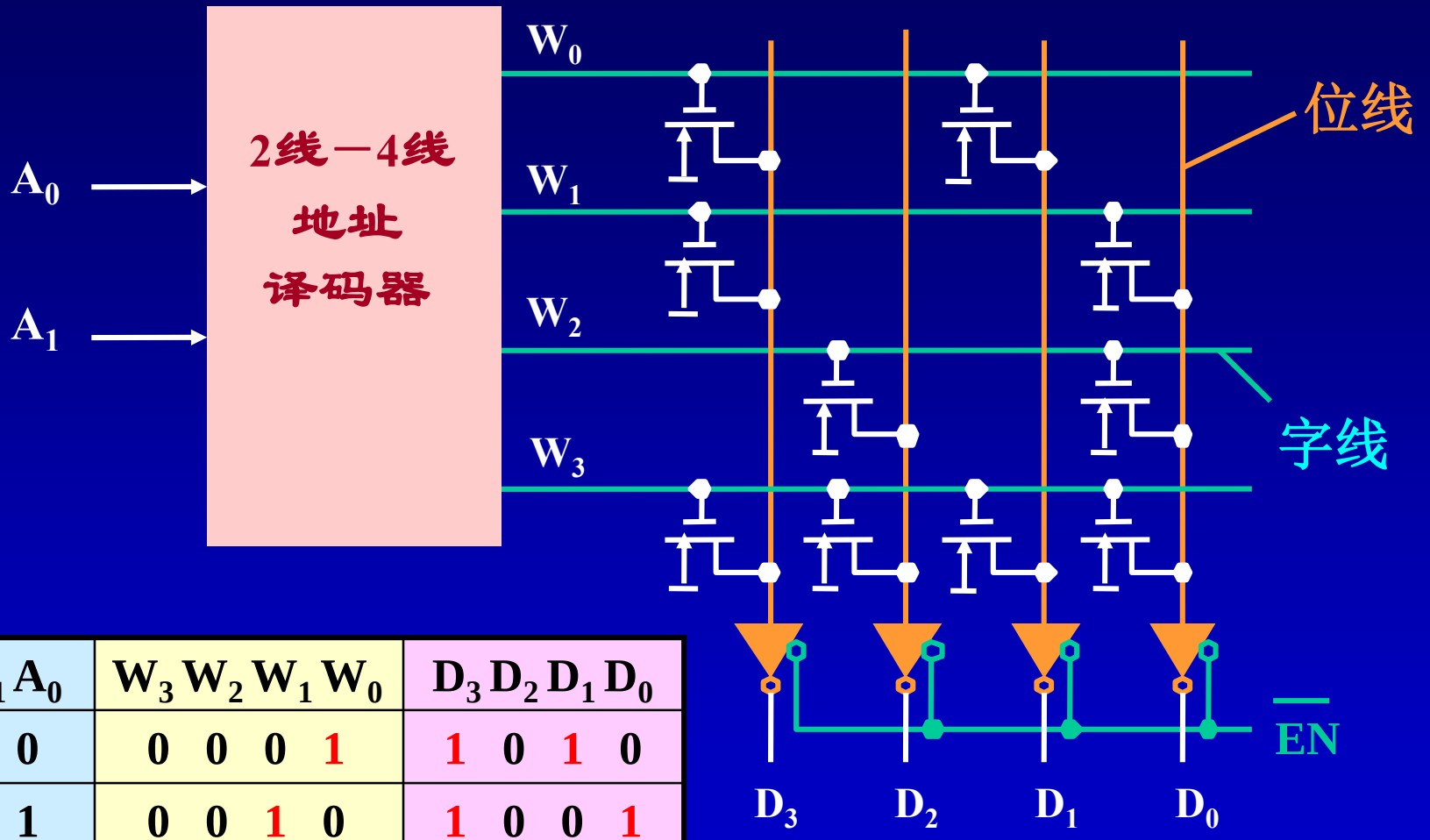
4×4 位 二极管 ROM



A_1A_0	$W_3W_2W_1W_0$	$D_3D_2D_1D_0$
0 0	0 0 0 1	1 0 1 0
0 1	0 0 1 0	1 0 0 1
1 0	0 1 0 0	0 1 0 1
1 1	1 0 0 0	1 1 1 1

字线与位线交叉点上：
接有二极管表示该位存“1”，
无二极管表示存“0”。

4×4 位 MOS管 ROM



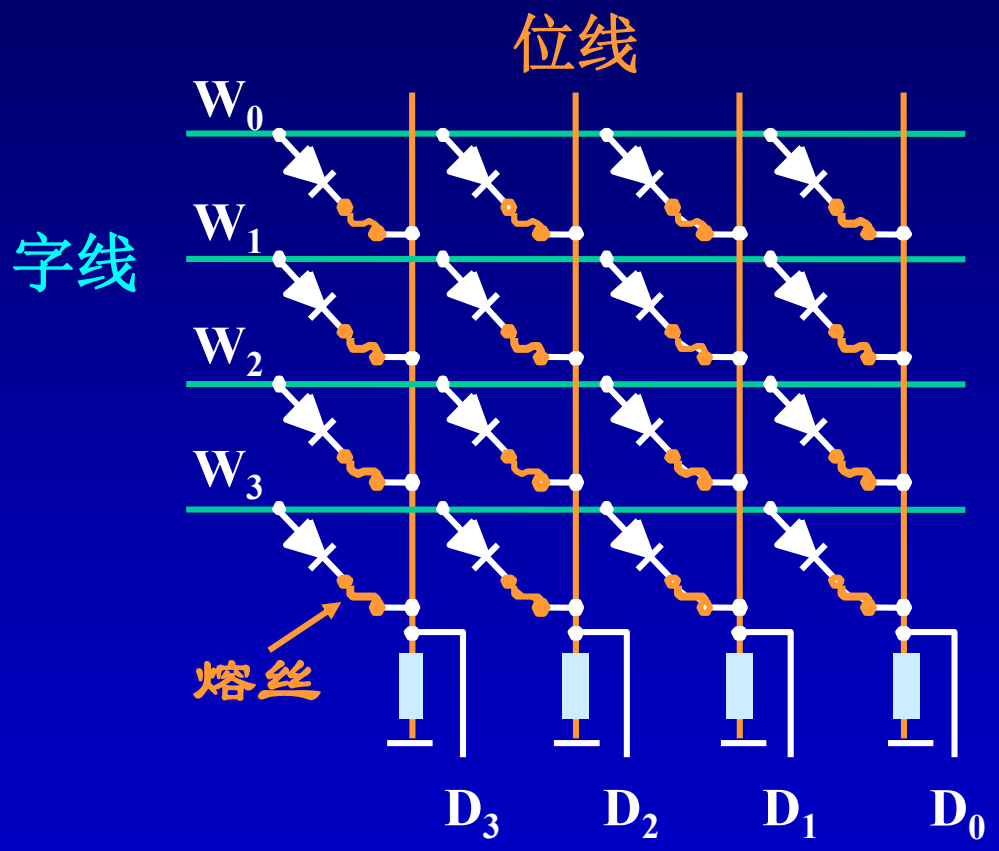
A_1 A_0	W_3 W_2 W_1 W_0	D_3 D_2 D_1 D_0
0 0	0 0 0 1	1 0 1 0
0 1	0 0 1 0	1 0 0 1
1 0	0 1 0 0	0 1 0 1
1 1	1 0 0 0	1 1 1 1

B、可编程 ROM（PROM）

出厂前，在PROM存储阵列的所有字线与位线的交叉点上都制作了存储元件（二极管、三极管或MOS管），并使存储元件通过熔丝与位线相连。

熔丝：很细的低熔点合金丝，是最早的编程元件

4×4 位 二极管 PROM

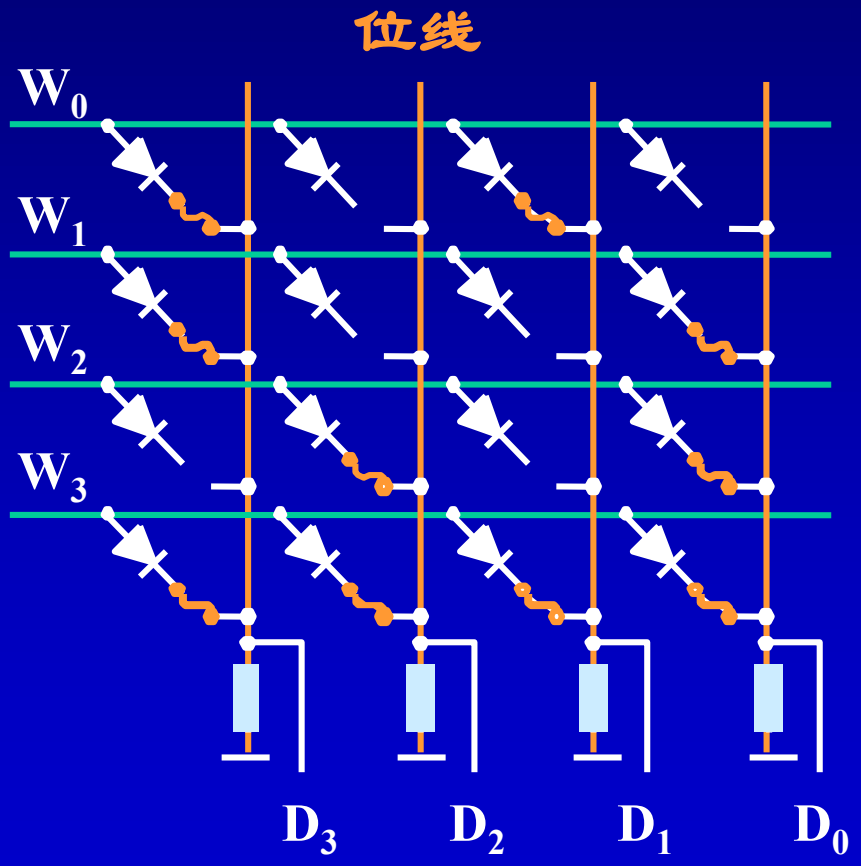


- PROM 出厂时，所有熔丝连通，相当于存储“1”

编程后的 PROM存储阵列

一次性编程

用编程器给需要写入“0”的存储单元通电流，将对应熔丝熔断。



A_1 A_0	W_3 W_2 W_1 W_0	D_3 D_2 D_1 D_0
0 0	0 0 0 1	1 0 1 0
0 1	0 0 1 0	1 0 0 1
1 0	0 1 0 0	0 1 0 1
1 1	1 0 0 0	1 1 1 1

C、可擦除可编程 ROM（EPROM）

在结构上与 PROM类似，只是采用了不同的存储元件和编程工艺。

EPROM
- UVEPROM（紫外线擦除）：整片擦除
- E^2PROM（电擦除）：擦除、写入同时进行
- Flash Memory（闪存）：快速擦除、写入

3、ROM 在组合逻辑设计中的应用

例1、试用ROM产生如下的一组多输出逻辑函数

$$\begin{cases} Y_1 = \bar{A}BC + \bar{A}\bar{B}C \\ Y_2 = A\bar{B}C\bar{D} + BC\bar{D} + \bar{A}BCD \\ Y_3 = ABC\bar{D} + \bar{A}B\bar{C}\bar{D} \\ Y_4 = \bar{A}\bar{B}C\bar{D} + ABCD \end{cases}$$

解：

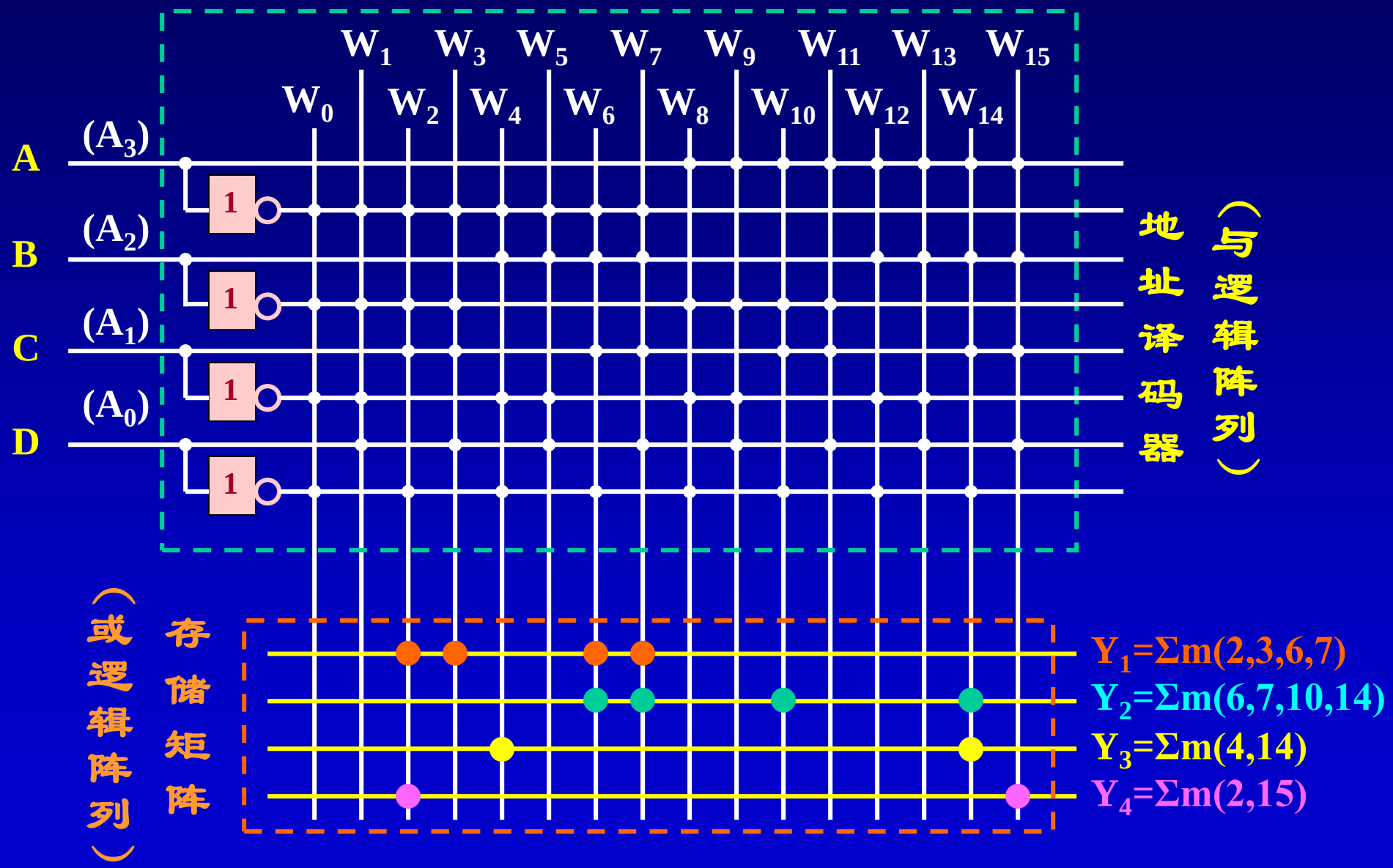
化为最小项之和的形式

$$\begin{cases} Y_1 = \bar{A}BCD + \bar{A}BC\bar{D} + \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D} = \Sigma m(2,3,6,7) \\ Y_2 = A\bar{B}C\bar{D} + \bar{A}BC\bar{D} + ABC\bar{D} + \bar{A}BCD = \Sigma m(6,7,10,14) \\ Y_3 = ABC\bar{D} + \bar{A}B\bar{C}\bar{D} = \Sigma m(4,14) \\ Y_4 = \bar{A}\bar{B}C\bar{D} + ABCD = \Sigma m(2,15) \end{cases}$$

$Y_1 = \Sigma m(2,3,6,7)$　　$Y_3 = \Sigma m(4,14)$

$Y_2 = \Sigma m(6,7,10,14)$　　$Y_4 = \Sigma m(2,15)$

（P381 图7.5.2）



例2、试用PROM构造两位二进制数的乘法器

解：

分析：

输入：两个 2位 二进制数

被乘数 $(A_1A_0)_2$　　乘数 $(B_1B_0)_2$

输出：4位 二进制数 $(D_3D_2D_1D_0)_2$

则有：$(A_1A_0)_2 \times (B_1B_0)_2 = (D_3D_2D_1D_0)_2$

乘法器真值表

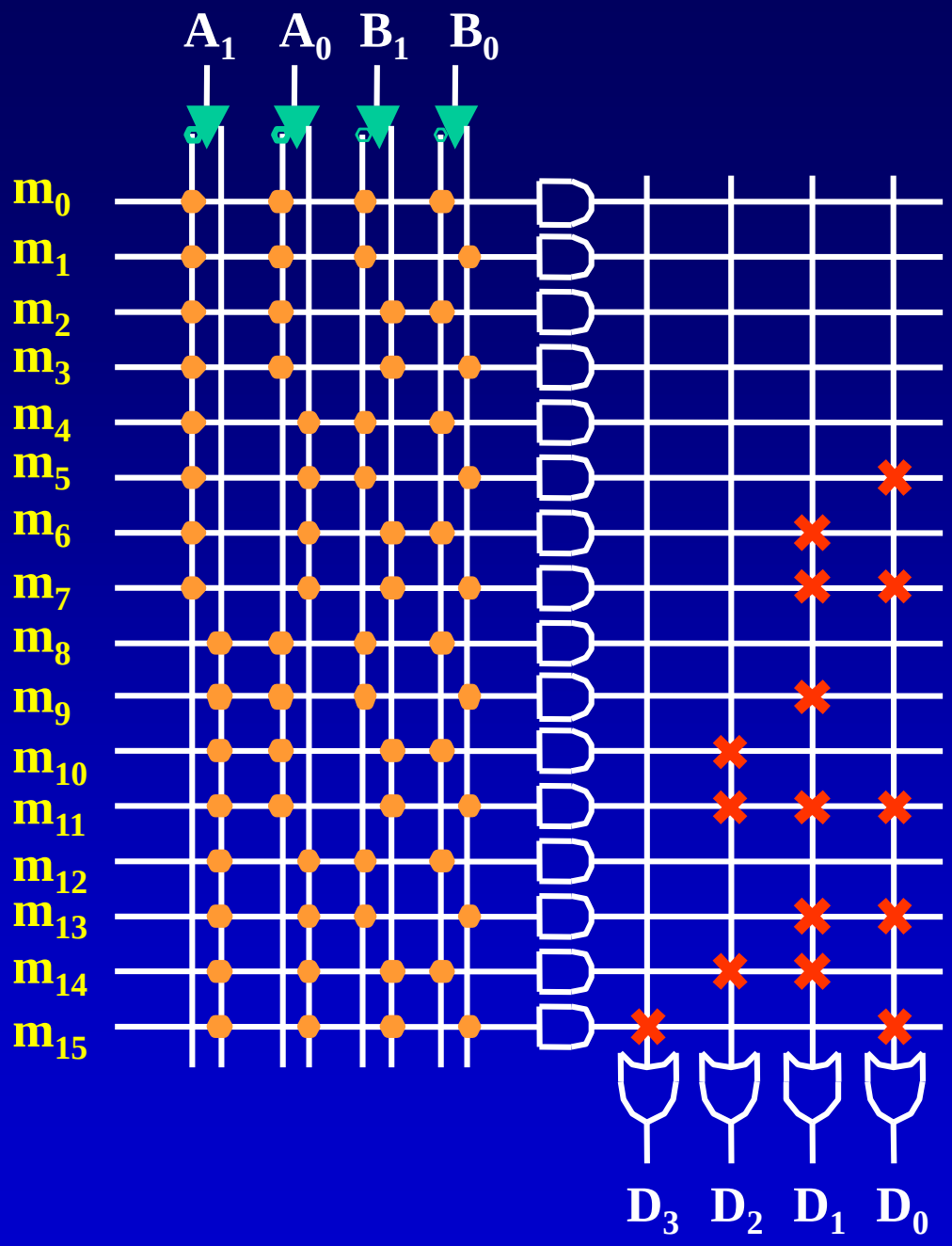
	A_1A_0 B_1B_0	$D_3D_2D_1D_0$
0×0	0 0 0 0	0 0 0 0
0×1	0 0 0 1	
0×2	0 0 1 0	
0×3	0 0 1 1	
1×0	0 1 0 0	0 0 0 0
1×1	0 1 0 1	0 0 0 1
1×2	0 1 1 0	0 0 1 0
1×3	0 1 1 1	0 0 1 1

	A_1A_0 B_1B_0	$D_3D_2D_1D_0$
2×0	1 0 0 0	0 0 0 0
2×1	1 0 0 1	0 0 1 0
2×2	1 0 1 0	0 1 0 0
2×3	1 0 1 1	0 1 1 0
3×0	1 1 0 0	0 0 0 0
3×1	1 1 0 1	0 0 1 1
3×2	1 1 1 0	0 1 1 0
3×3	1 1 1 1	1 0 0 1

由真值表，可得

$$\begin{cases} D_3 = \Sigma\, m\,(15) \\ D_2 = \Sigma\, m\,(10, 11, 14) \\ D_1 = \Sigma\, m\,(6, 7, 9, 11, 13, 14) \\ D_0 = \Sigma\, m\,(5, 7, 13, 15) \end{cases}$$

只要将 $A_1 A_0 B_1 B_0$ 按顺序作为PROM的地址，把它们的乘积存放在相应的存储单元，即可实现两个2位二进制数的乘法。



$D_3 = \Sigma\, m\,(15)$

$D_2 = \Sigma\, m\,(10, 11, 14)$

$D_1 = \Sigma\, m\,(6, 7, 9, 11, 13, 14)$

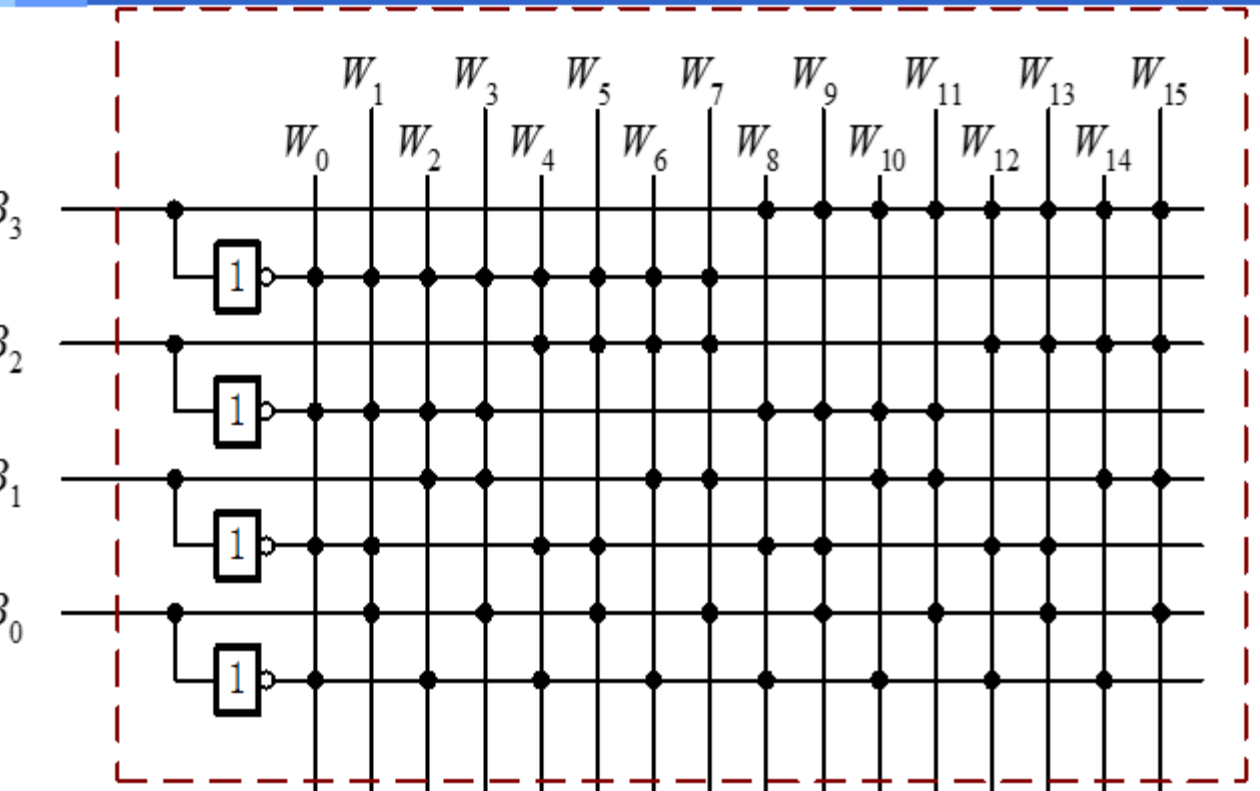
$D_0 = \Sigma\, m\,(5, 7, 13, 15)$

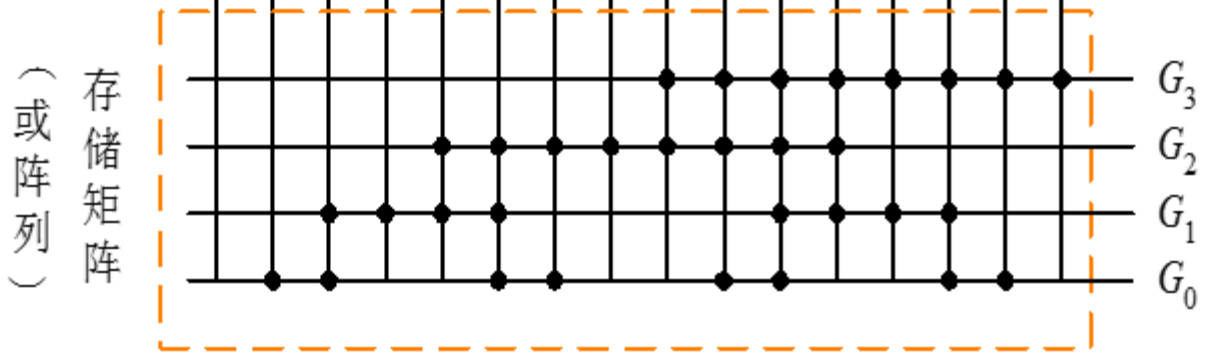
例2、用ROM实现二进制码转换为格雷码。

字	二进制码				格雷码			
	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
W_0	0	0	0	0	0	0	0	0
W_1	0	0	0	1	0	0	0	1
W_2	0	0	1	0	0	0	1	1
W_3	0	0	1	1	0	0	1	0
W_4	0	1	0	0	0	1	1	0
W_5	0	1	0	1	0	1	1	1
W_6	0	1	1	0	0	1	0	1
W_7	0	1	1	1	0	1	0	0
W_8	1	0	0	0	1	1	0	0
W_9	1	0	0	1	1	1	0	1
W_{10}	1	0	1	0	1	1	1	1
W_{11}	1	0	1	1	1	1	1	0
W_{12}	1	1	0	0	1	0	1	0
W_{13}	1	1	0	1	1	0	1	1
W_{14}	1	1	1	0	1	0	0	1
W_{15}	1	1	1	1	1	0	0	0

$G3=\sum m$（8, 9, 10, 11, 12, 13, 14, 15）
$G2=\sum m$（4, 5, 6, 7, 8, 9, 10, 11）
$G1=\sum m$（2, 3, 4, 5,10, 11, 12, 13）
$G0=\sum m$（1, 2, 5, 6, 9, 10, 13, 14）

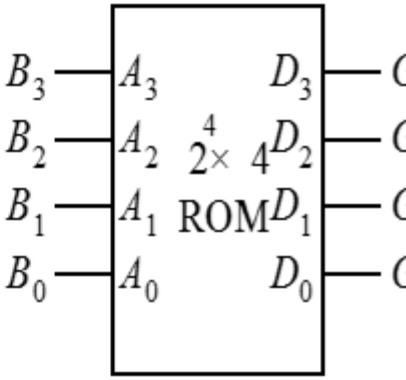
二进制码转换为格雷码的阵列图





(a)

G3=∑m（8, 9, 10, 11, 12, 13, 14, 15）
G2=∑m（4, 5, 6, 7, 8, 9, 10, 11）
G1=∑m（2, 3, 4, 5,10, 11, 12, 13）
G0=∑m（1, 2, 5, 6, 9, 10, 13, 14）



(b)

逻辑符号图