数字逻辑电路

(2020级本科生课程)

清华大学计算机系

陶品

taopin@tsinghua.edu.cn

办公室：FIT 3－531 （13717813059）

3.3 常用的中规模组合逻辑电路

3.3.1 译码器

3.3.2 数据选择器

⇨ 3.3.3 编码器

3.3.4 数据比较器

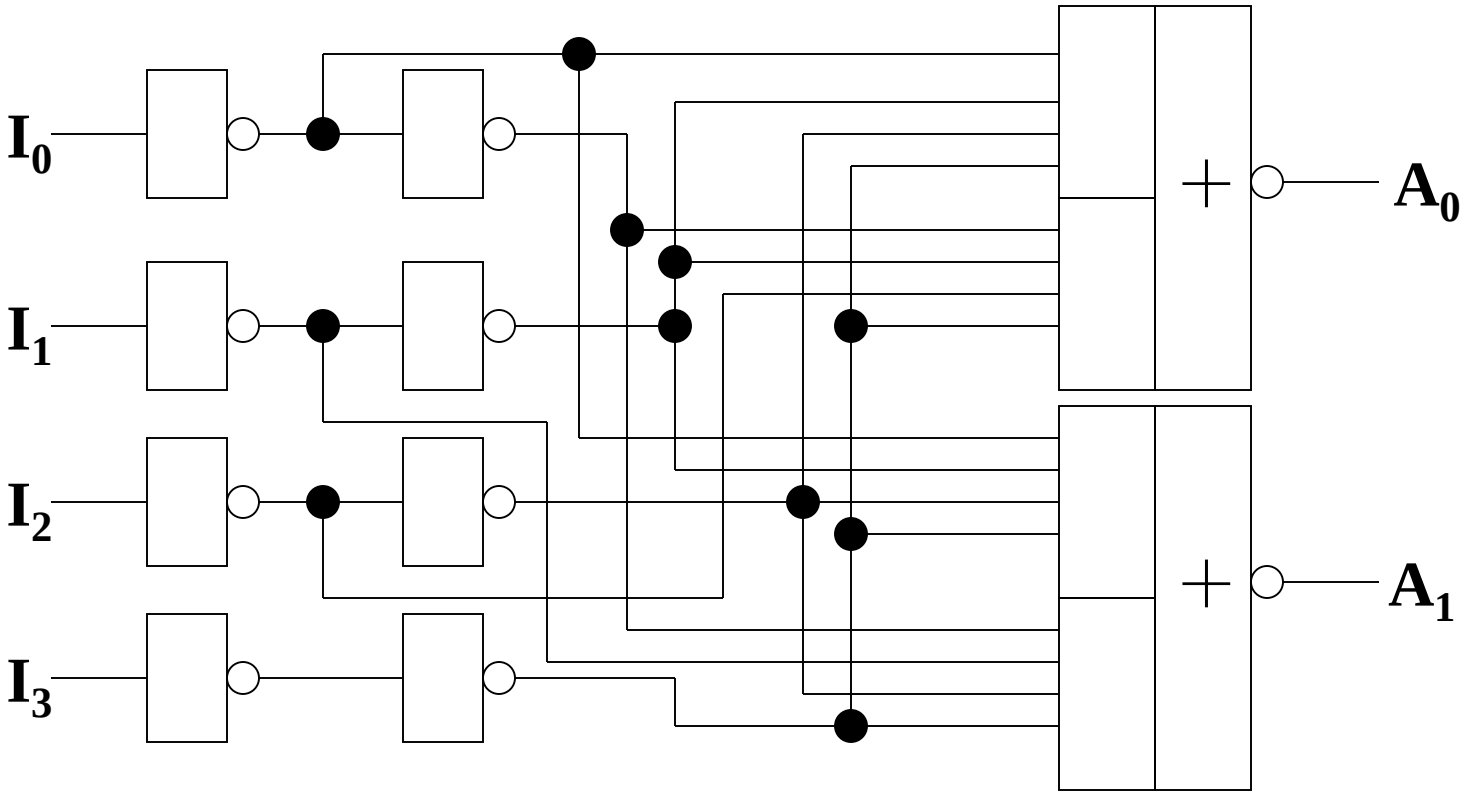
3.3.5 奇偶校验器

3.3.6 可编程逻辑器件

3.3.7 运算器（算数逻辑单元 ALU）

3.3.3 编码器（0）

- 如果给出一个这样的逻辑图，同学们能不能分析出逻辑功能？



3.3.3 编码器（1）

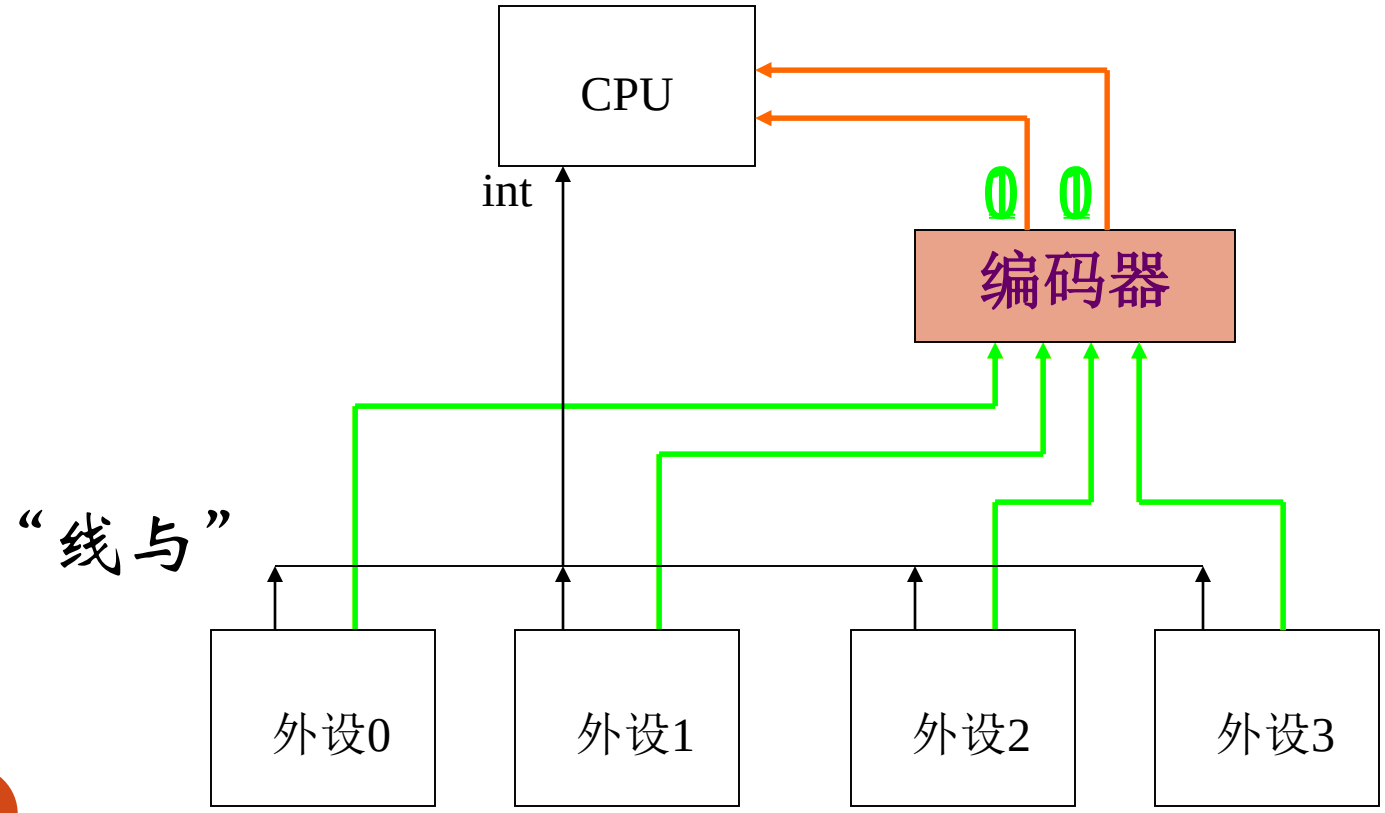
- 编码器(Encoder)
  - 编码器原理
  - 优先编码器(Priority Encoder)
  - 8-3优先编码器
  - 扩展应用：16-4 优先编码器

3.3.3 编码器（2）

- 编码器(Encoder)原理
  - 功能：将译码器反过来，对应输入的每一个状态，输出一个编码。
  - 常用编码器
    - 4-2编码，将输入的4个状态编成2位二进制数码；
    - 8-3编码，将输入的8个状态编成3位二进制数码；
    - BCD编码，将10个输入编成BCD码。

3.3.3 编码器（3）

- 为什么要用编码器？

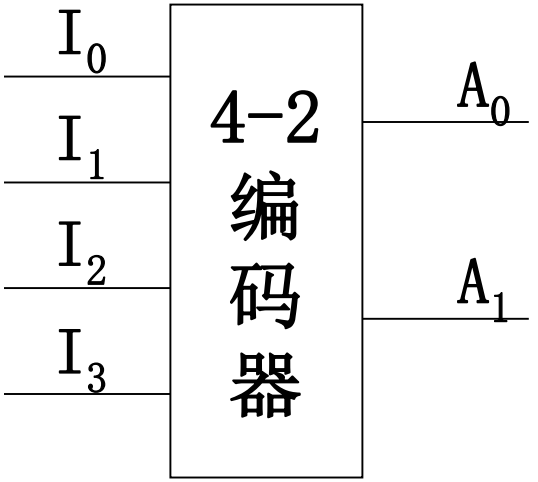


3.3.3 编码器（4）

- 4-2编码器：将输入的4个状态编成2位二进制数码

功能表

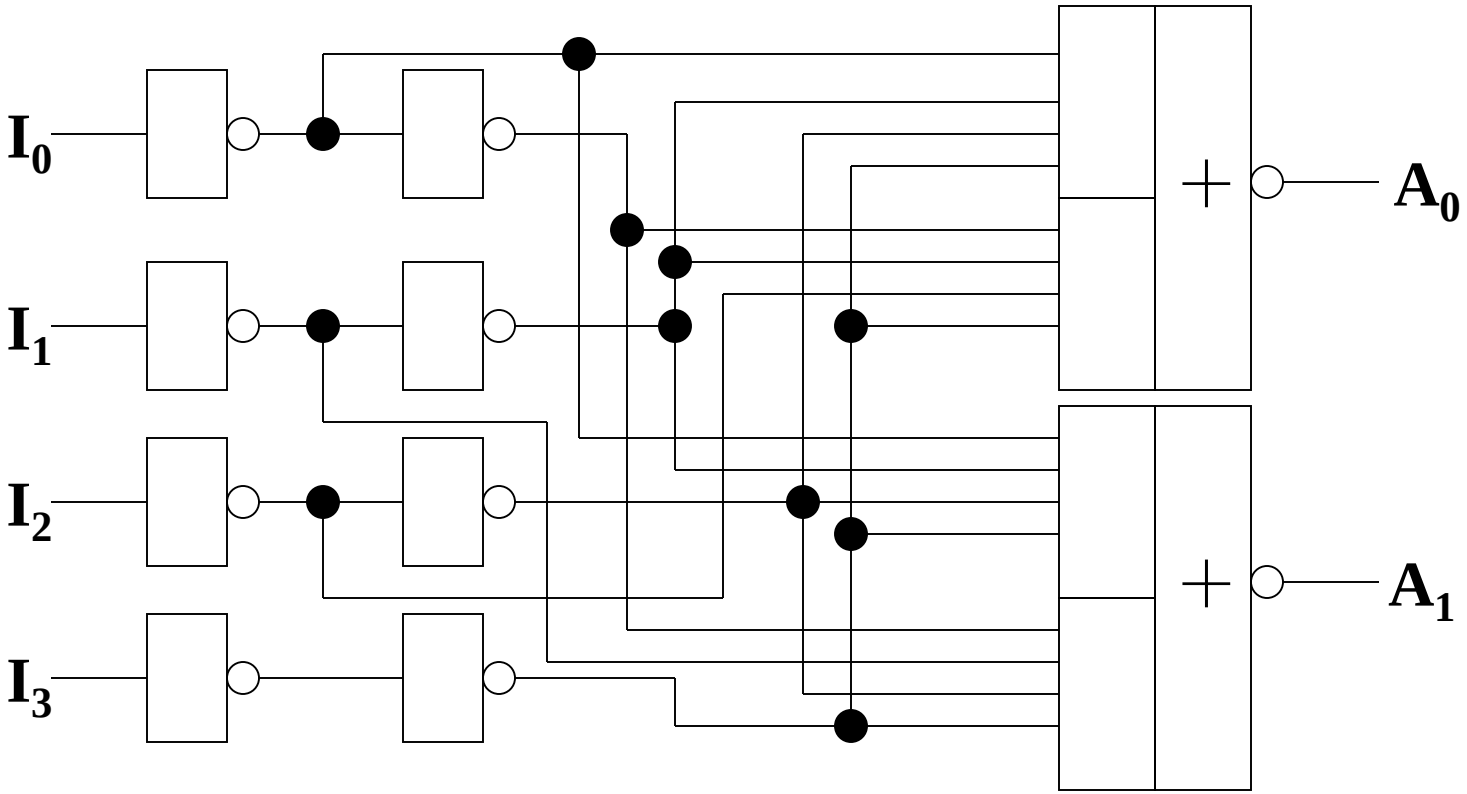
I_0	I_1	I_2	I_3	A_0	A_1
0	1	1	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	0	1	1



$$\begin{cases} A_0 = I_0\overline{I_1}I_2I_3 + I_0I_1I_2\overline{I_3} = \overline{\overline{I_0}I_1I_2I_3 + I_0I_1\overline{I_2}I_3} \\ A_1 = I_0I_1\overline{I_2}I_3 + I_0I_1I_2\overline{I_3} = \overline{\overline{I_0}I_1I_2I_3 + I_0\overline{I_1}I_2I_3} \end{cases}$$

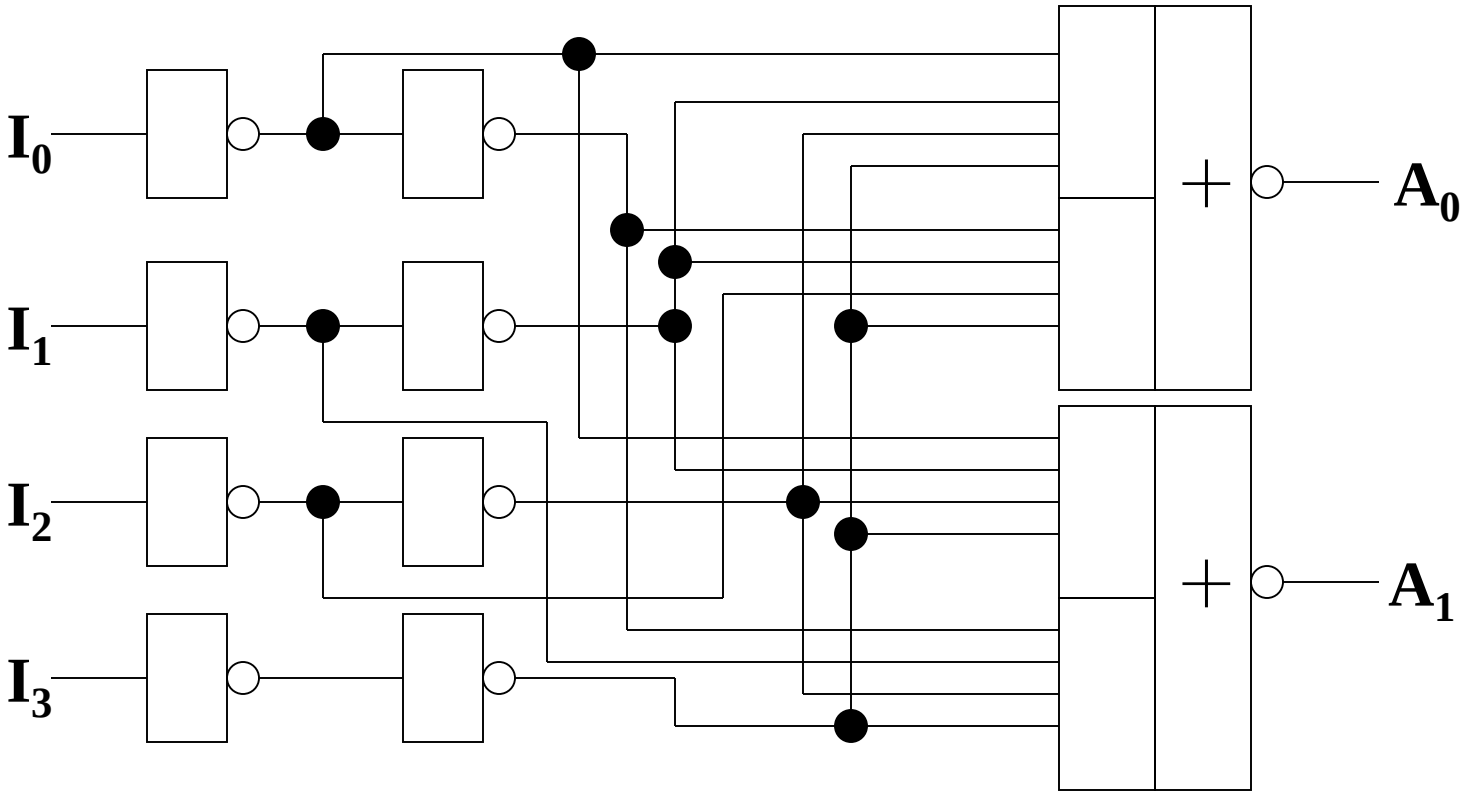
3.3.3 编码器（5）

$$\begin{cases} A_0 = I_0\overline{I_1}I_2I_3 + I_0I_1I_2\overline{I_3} = \overline{\overline{I_0}I_1I_2I_3 + I_0I_1\overline{I_2}I_3} \\ A_1 = I_0I_1\overline{I_2}I_3 + I_0I_1I_2\overline{I_3} = \overline{\overline{I_0}I_1I_2I_3 + I_0\overline{I_1}I_2I_3} \end{cases}$$



3.3.3 编码器（6）

- 如果给出一个这样的逻辑图，同学们能不能分析出逻辑功能？

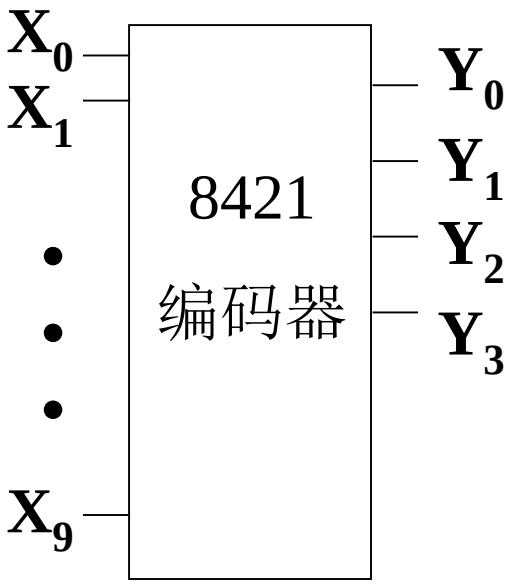


3.3.3 编码器（7）

- 8-3编码器的设计与4-2编码器类似，课堂上不在讲解，希望同学们自己课后练习。

3.3.3 编码器（8）

- 8421码编码器
  - 功能：将输入信号编码成8421码。
  - 输入：$X_0 \sim X_9$
  - 输出：$Y_0 \sim Y_3$



3.3.3 编码器（9）

- 8421码编码器功能表及输出表达式

X_9	X_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	1

$$Y_3 = X_8 + X_9$$

$$Y_2 = X_4 + X_5 + X_6 + X_7$$

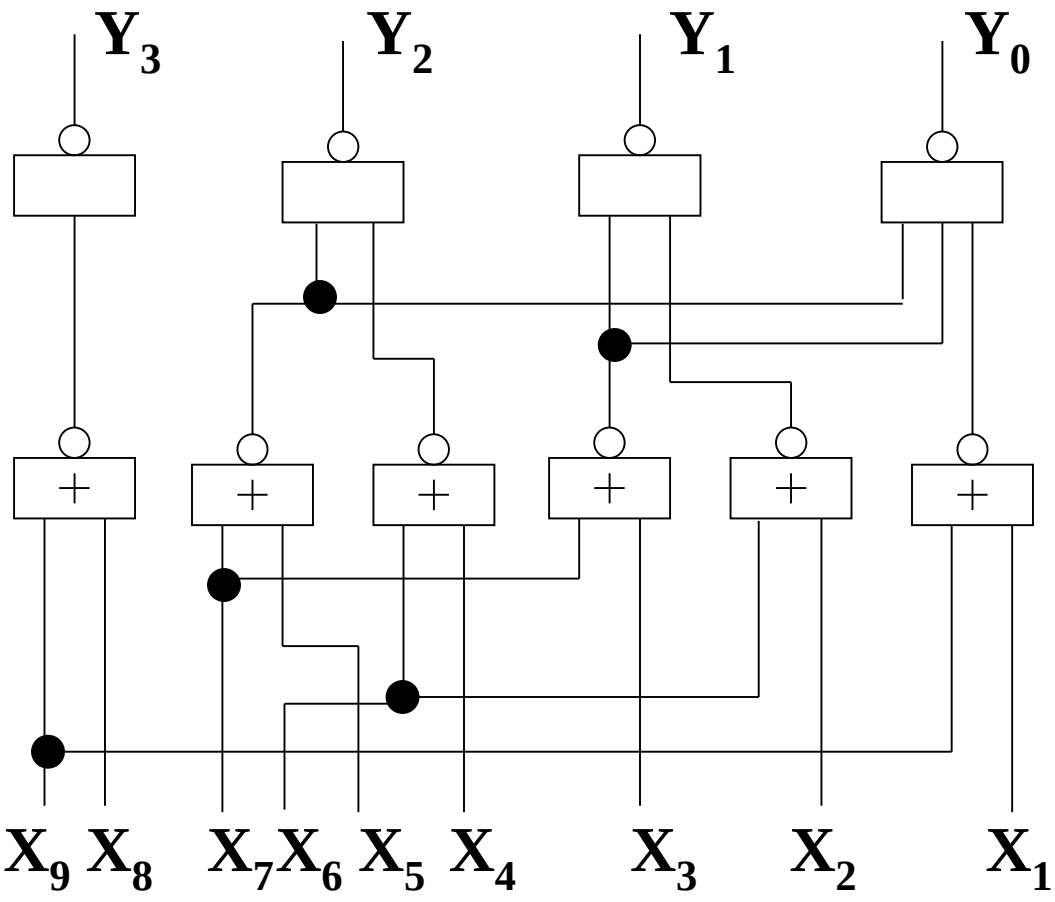
$$Y_1 = X_2 + X_3 + X_6 + X_7$$

$$Y_0 = X_1 + X_3 + X_5 + X_7 + X_9$$

3.3.3 编码器（10）

- 根据8421码编码器输出表达式画出逻辑图

$$Y_3 = X_8 + X_9$$
$$Y_2 = X_4 + X_5 + X_6 + X_7$$
$$Y_1 = X_2 + X_3 + X_6 + X_7$$
$$Y_0 = X_1 + X_3 + X_5 + X_7 + X_9$$



3.3.3 编码器（11）

- 前面设计的编码器存在的问题
  - 4－2编码器
  - 8421码编码器

3.3.3 编码器（12）

- 4－2编码器的问题

功能表

I_0	I_1	I_2	I_3	A_0	A_1
0	1	1	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	0	1	1

假设I_1＝”0”，I_2＝”0”，

从输出表达式中可以得到

A_0 ＝ “1” A_1 ＝ “1”

产生问题的原因是：2－4编码器的功能表中没有完全包含输入的全部逻辑组合。

$$\begin{cases} A_0 = \overline{\overline{I_0}I_1I_2I_3 + I_0I_1\overline{I_2}I_3} \\ A_1 = \overline{\overline{I_0}I_1I_2I_3 + I_0\overline{I_1}I_2I_3} \end{cases}$$

3.3.3 编码器（11）

- 前面设计的编码器存在的问题
  - 只有互斥输入时，才能用这种编码器。即在任一时刻所有输入线中最多只允许有一个为“0”（4－2编码器器）或“1”（8421码编码器），否则编码器会发生混乱。
- 解决办法：
  - 采用优先编码器。

3.3.3 编码器（13）

- 优先编码器
  - 当两条或两条以上线为“0”时，优先按输入编号大的编码，称优先编码器(Priority Encoder)。
  - 以8-3优先编码器为例

3.3.3 编码器（14）

8-3优先编码器

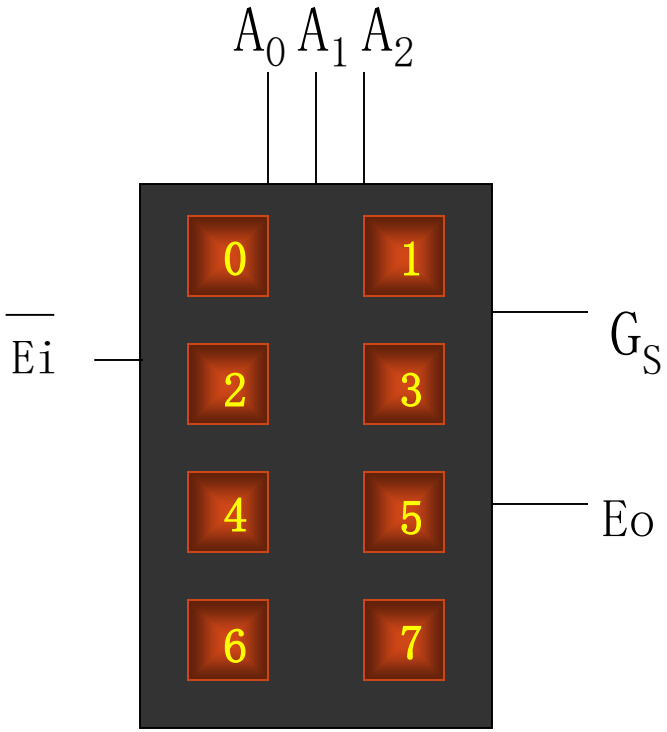
8键键盘优先编码器设计

1) 需要对各键进行编码 A_2,A_1,A_0

2) 需要设定输入使能 $\overline{Ei}$

3) 需要设定输出的编码是否有效指示Gs

4) 需要设定是否允许编码级联输出 E_0



8-3优先编码功能表

$\overline{Ei}$	0	1	2	3	4	5	6	7	A_0	A_1	A_2	G_S	Eo
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	1	0	0	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	1	1	0	0	1
0	X	X	X	0	1	1	1	1	0	0	1	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
1	X	X	X	X	X	X	X	X	1	1	1	1	1

(A_2,A_1,A_0用反码编码，Gs为编码输出有效，$\overline{Ei}$为使能输入，Eo为允许级联输出)

8-3优先编码功能表

$\overline{Ei}$	0	1	2	3	4	5	6	7	A_0	A_1	A_2	G_S	Eo
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	1	0	0	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	1	1	0	0	1
0	X	X	X	0	1	1	1	1	0	0	1	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
1	X	X	X	X	X	X	X	X	1	1	1	1	1

$$\overline{A_0} = \bar{7} + 76\bar{5} + 765\bar{4}3 + 7654321 = \bar{7} + 6\bar{5} + 64\bar{3} + 642\bar{1}$$

吸收率 $(A + \bar{A}B = A + B)$

8-3优先编码功能表

$\overline{Ei}$	0	1	2	3	4	5	6	7	A_0	A_1	A_2	G_S	Eo
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	1	0	0	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	1	1	0	0	1
0	X	X	X	0	1	1	1	1	0	0	1	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
1	X	X	X	X	X	X	X	X	1	1	1	1	1

$$\overline{A_1} = \overline{7} + \overline{76} + \overline{7654} + \overline{76543} = \overline{7} + \overline{6} + \overline{543} + \overline{542}$$

8-3优先编码功能表

$\overline{Ei}$	0	1	2	3	4	5	6	7	A_0	A_1	A_2	G_S	*Eo*
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	1	0	0	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	1	1	0	0	1
0	X	X	X	0	1	1	1	1	0	0	1	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
1	X	X	X	X	X	X	X	X	1	1	1	1	1

$$\overline{A_2} = \overline{7} + \overline{76} + \overline{765} + \overline{7654} = \overline{7} + \overline{6} + \overline{5} + \overline{4}$$

8-3优先编码功能表

$\overline{E_i}$	0	1	2	3	4	5	6	7	A_0	A_1	A_2	G_S	E_0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	1	0	0	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	1	1	0	0	1
0	X	X	X	0	1	1	1	1	0	0	1	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
1	X	X	X	X	X	X	X	X	1	1	1	1	1

$$E_0 = \overline{\overline{E_i} \bullet 76543210} \qquad G_s = \overline{E_0 \bullet \overline{E_i}}$$

优先编码功能表

$\overline{Ei}$	0	1	2	3	4	5	6	7	A_0	A_1	A_2	G_S	Eo
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	1	0	0	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	1	1	0	0	1
0	X	X	X	0	1	1	1	1	0	0	1	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
1	X	X	X	X	X	X	X	X	1	1	1	1	1

(A_2,A_1,A_0用反码编码，$\overline{E_i}$为使能输入，Gs为编码输出有效，Eo为允许级联输出)

$$\overline{A_0} = \overline{7} + 76\overline{5} + 765\overline{4} + 7654\overline{3} + 76543\overline{2} + 765432\overline{1} = \overline{7} + \overline{6}\overline{5} + \overline{6}\overline{4}\overline{3} + \overline{6}\overline{4}\overline{2}\overline{1}$$

$$\overline{A_1} = \overline{7} + 7\overline{6} + 765\overline{4} + 7654\overline{3} + 76543\overline{2} = \overline{7} + \overline{6} + 5\overline{4}\overline{3} + 5\overline{4}\overline{2}$$

$$\overline{A_2} = \overline{7} + 7\overline{6} + 76\overline{5} + 765\overline{4} = \overline{7} + \overline{6} + \overline{5} + \overline{4}$$

$$E_0 = \overline{\overline{Ei} \bullet 76543210} \qquad G_s = \overline{E_0 \bullet \overline{Ei}}$$

$\overline{Ei}$	0	1	2	3	4	5	6	7	A_0	A_1	A_2	G_S	Eo
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	1	0	0	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	1	1	0	0	1
0	X	X	X	0	1	1	1	1	0	0	1	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
1	X	X	X	X	X	X	X	X	1	1	1	1	1

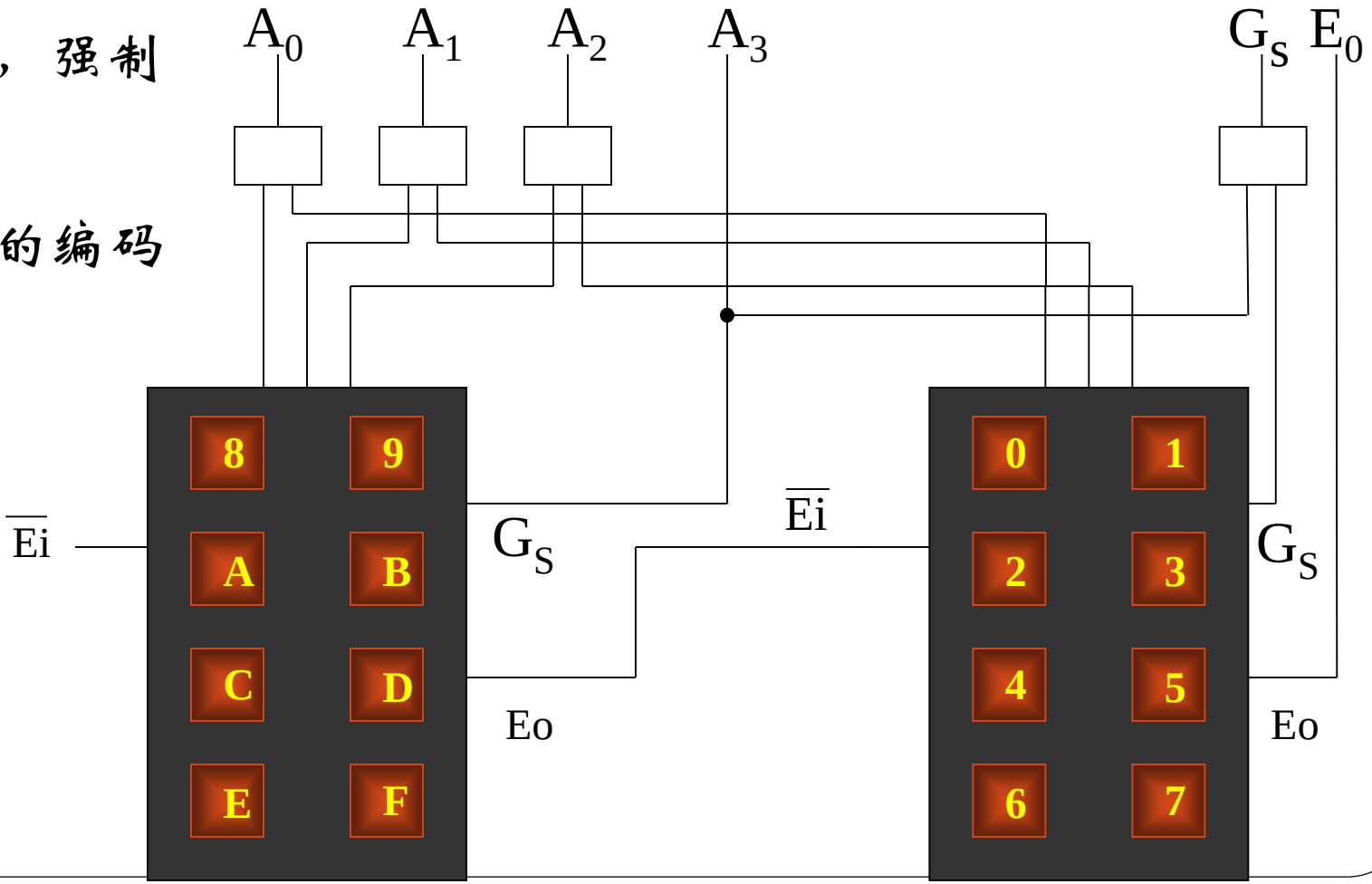
- 如果$\overline{Ei}$=0，输入有“0”，则Gs=“0”，Eo=“1”编码有效，级联禁止
- 如果$\overline{Ei}$=0，输入全为“1”，则Gs=“1”，Eo=“0”编码无效，级联有效
- 如果$\overline{Ei}$=1，无论输入为何值，Gs=“1”，Eo=“1”编码无效，级联禁止
- 输出编码为反码，即用“000”表示7，用“111”表示0

3.3.3 编码器（15）

- 将8-3优先编码器扩展为16-4优先编码器

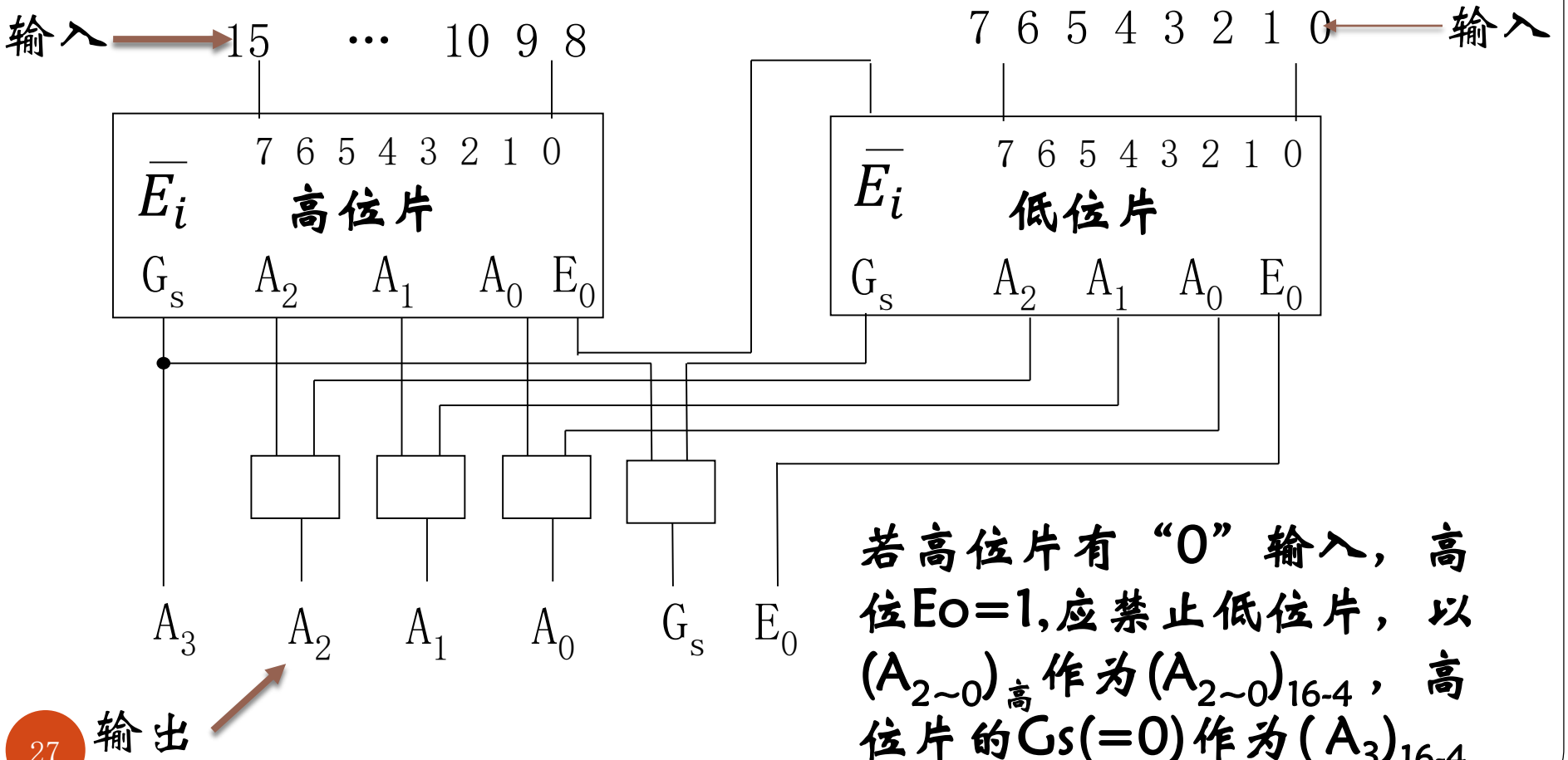
高位有效，强制低位无效

生成相应的编码输出



3.3.3 编码器（16）

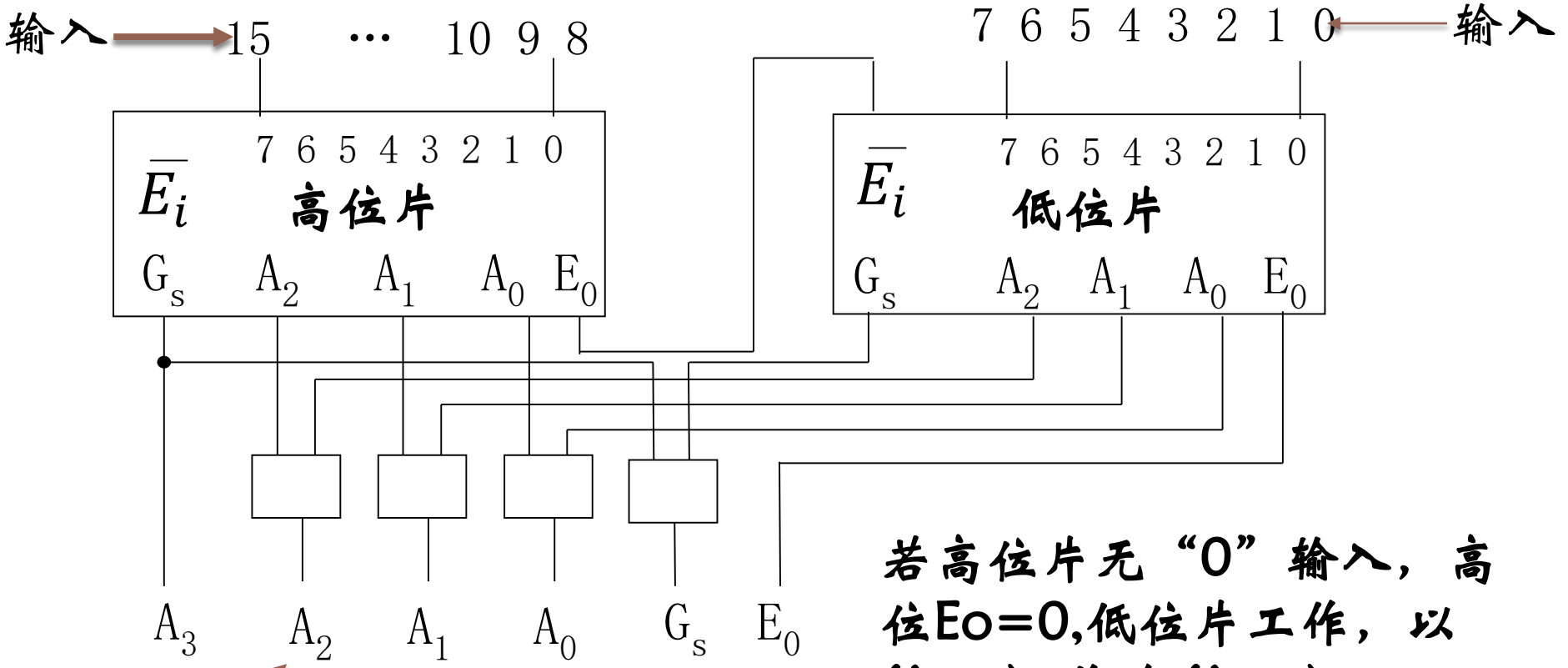
- 将8-3优先编码器扩展为16-4优先编码器



若高位片有“0”输入，高位Eo=1,应禁止低位片，以$(A_{2\sim0})_{高}$作为$(A_{2\sim0})_{16\text{-}4}$，高位片的Gs(=0)作为$(A_3)_{16\text{-}4}$

3.3.3 编码器（16）

- 将8-3优先编码器扩展为16-4优先编码器



若高位片无“0”输入，高位Eo=0,低位片工作，以$(A_{2\sim0})_{低}$作为$(A_{2\sim0})_{16\text{-}4}$，$(A_3)_{16\text{-}4}$为“1”

3.3.3 编码器（18）

- 优先编码器在计算机中的应用
  - 设备按照优先等级编码，用于中断响应
  - 键盘输入的读取
  - ……

3.3 常用的中规模组合逻辑电路

3.3.1 译码器

3.3.2 数据选择器

3.3.3 编码器

⇨ 3.3.4 数据比较器

3.3.5 奇偶校验器

3.3.X 可编程逻辑器件

3.3.6 运算器（算数逻辑单元 ALU）

3.3.4 数据比较器(1)

3.3.4 数据比较器

- 定义：数字系统中能够完成数据比较功能的部件
- 功能：比较A、B两数，判断A>B、A<B、A=B。
- 基本运算：以四位比较为例说明。

$A_3\ A_2\ A_1\ A_0$
$B_3\ B_2\ B_1\ B_0$

从高位开始比较，若$A_3>B_3$ 则A>B，
若$A_3<B_3$ 则A<B，
若$A_3=B_3$ 则再比较低位

3.3.4 数据比较器(2)

3.3.4 数据比较器

广义的比较条件

$A_i > B_i$的条件：$A_i=1$，$B_i=0$；即 $A_i \bullet \overline{B_i} = 1$或$Z = A_i \bullet \overline{A_i B_i} = 1$

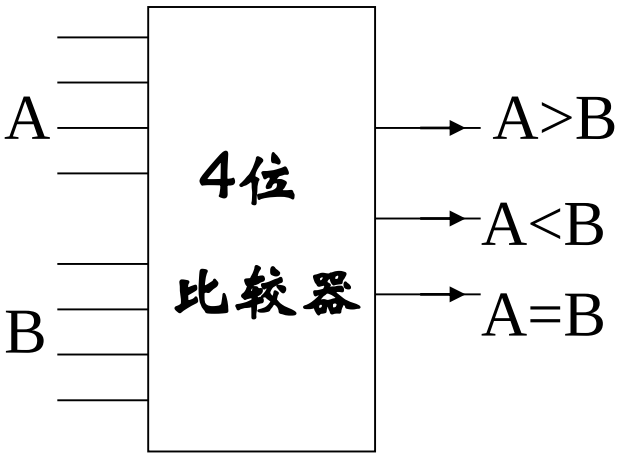
$A_i < B_i$的条件：$A_i=0$，$B_i=1$；即 $\overline{A_i} \bullet B_i = 1$或$W = B_i \bullet \overline{A_i B_i} = 1$

$A_i = B_i$的条件：$\overline{A_i \oplus B_i} = 1$ 或 $Y = \overline{(A_i \cdot \overline{A_i \cdot B_i})} + \overline{(\overline{A_i \cdot B_i} \cdot B_i)} = 1$

3.3.4 数据比较器(3)

数据比较器功能表

A_3 B_3	A_2 B_2	A_1 B_1	A_0 B_0	A>B	A<B	A=B
$A_3 > B_3$	X	X	X	1	0	0
$A_3 < B_3$	X	X	X	0	1	0
$A_3 = B_3$	$A_2 > B_2$	X	X	1	0	0
A3= B3	A2 <B2	X	X	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	0	0	1



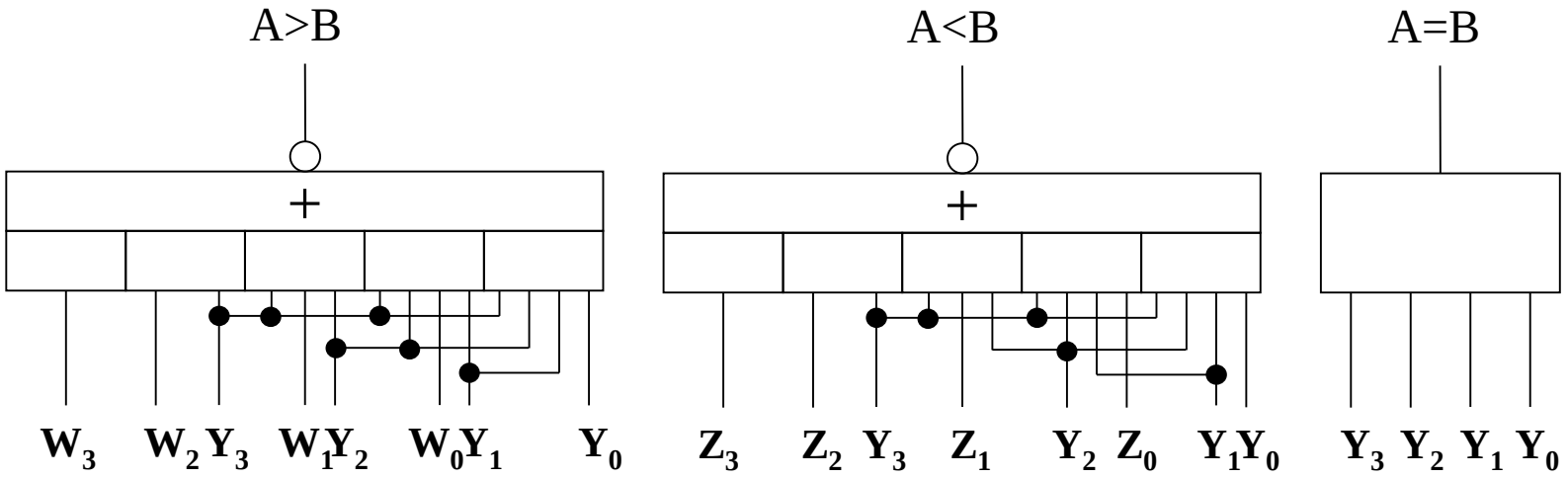
表达式：教材P128

3.3.4 数据比较器(3)

$$W_i = \text{“}A_i < B_i\text{”} = B_i \cdot \overline{A_i \cdot B_i}$$

$$Y_i = \text{“}A_i = B_i\text{”} = \overline{(A_i \cdot \overline{A_i \cdot B_i}) + (\overline{A_i \cdot B_i} \cdot B_i)}$$

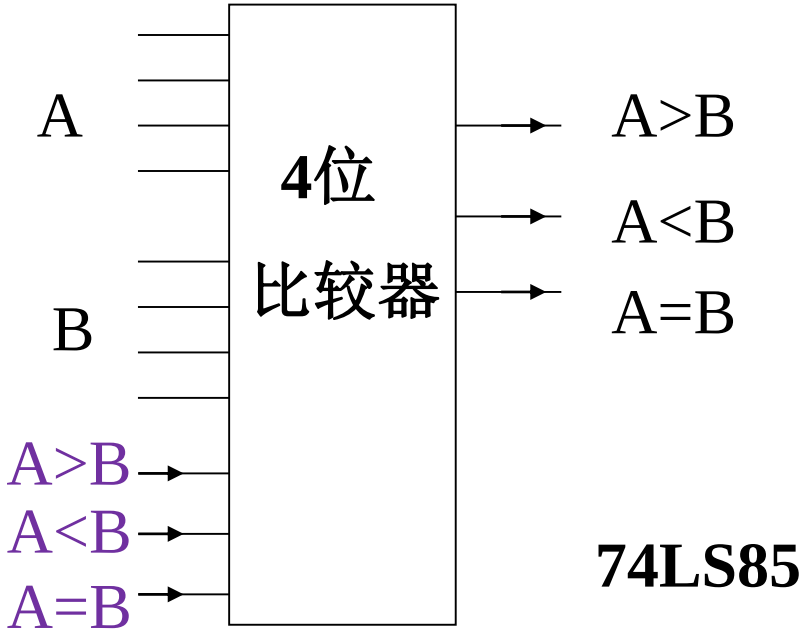
$$Z_i = \text{“}A_i > B_i\text{”} = A_i \cdot \overline{A_i \cdot B_i}$$



3.3.4 数据比较器(4)

3.3.4 数据比较器

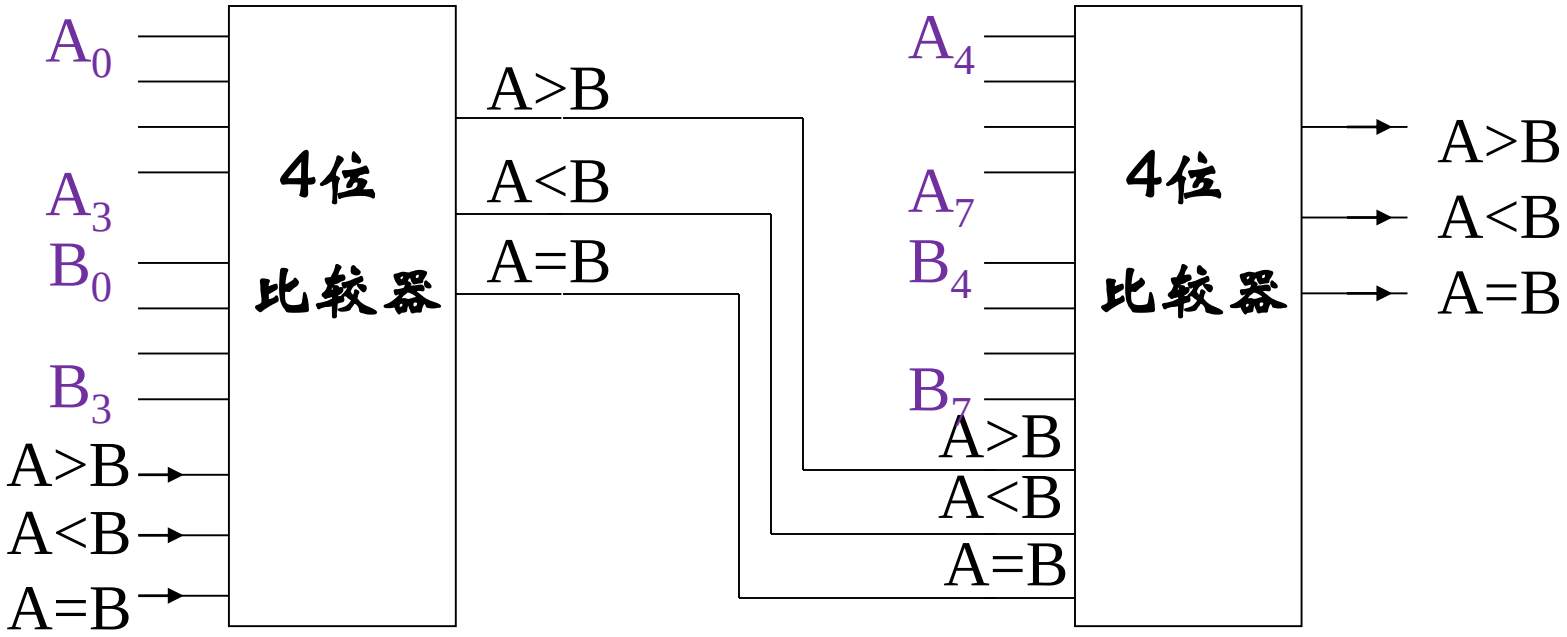
- 分段比较：多片比较器构成更长位数的方法
- 比较器不仅输出比较结果，还要能接受其它片输出的结果。



3.3.4 数据比较器(5)

3.3.4 数据比较器

- 应用举例：用两片4位数字比较器构成一个8位的数字比较器



3.3 常用的中规模组合逻辑电路

3.3.1 译码器

3.3.2 数据选择器

3.3.3 编码器

3.3.4 数据比较器

⇨ 3.3.5 奇偶校验器

3.3.X 可编程逻辑器件

3.3.6 运算器（算数逻辑单元 ALU）

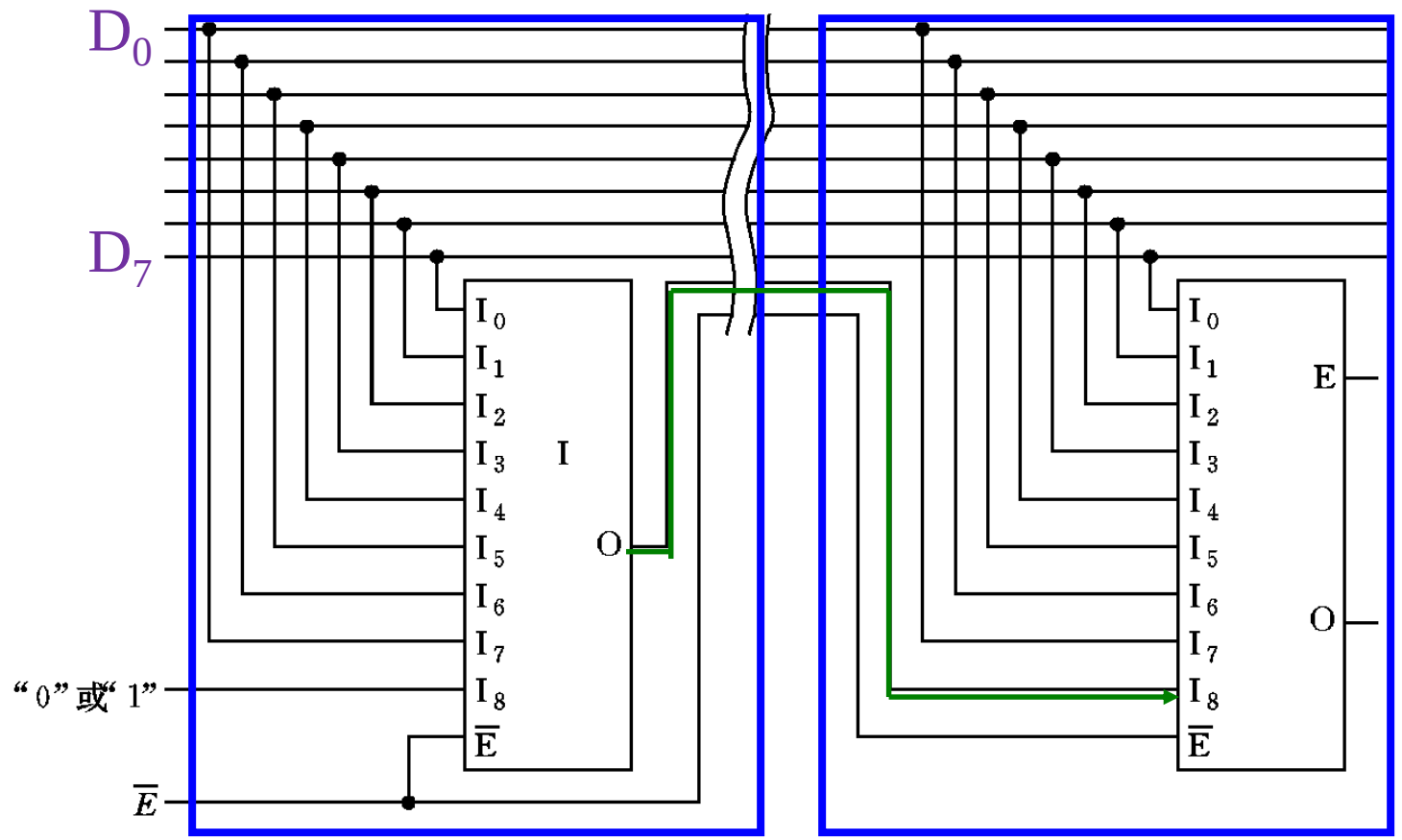
3.3.5 奇偶校验器

■ 3.3.5 奇偶校验器

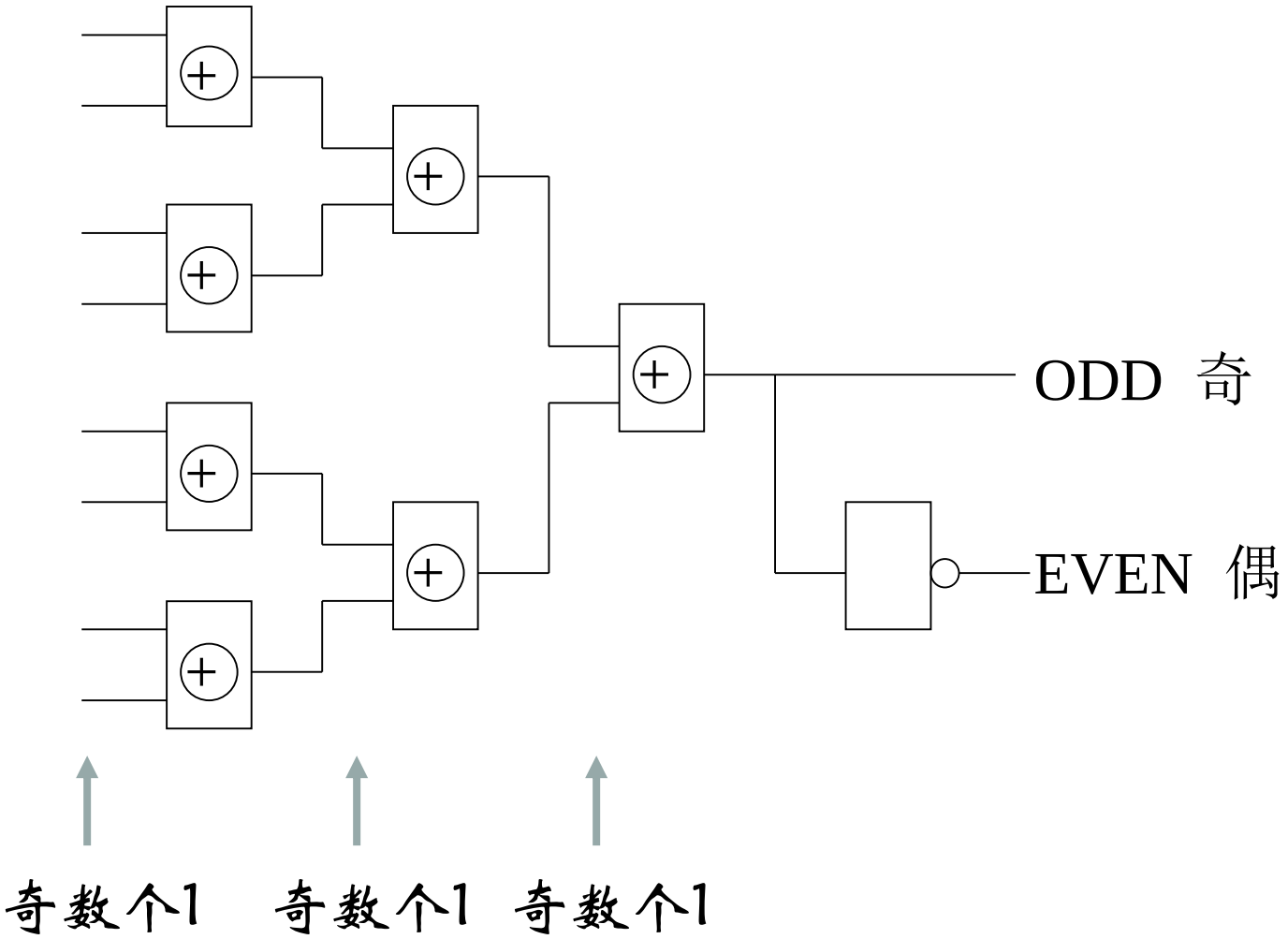
- “奇偶检测”是检测数据中包含奇数个“1”，还是偶数个“1”。
- 奇偶校验：发送或存储时产生奇偶校验码，在接收方或读出时进行奇偶校验，判断是否出错
- 采用奇偶校验方法，去检查数据传输和记录中是否产生了错误
- 但是只能发现“1位错”，而且不能纠错
- 要产生更强的检错和纠错能力，要增加校验位数

奇偶校验器逻辑结构图

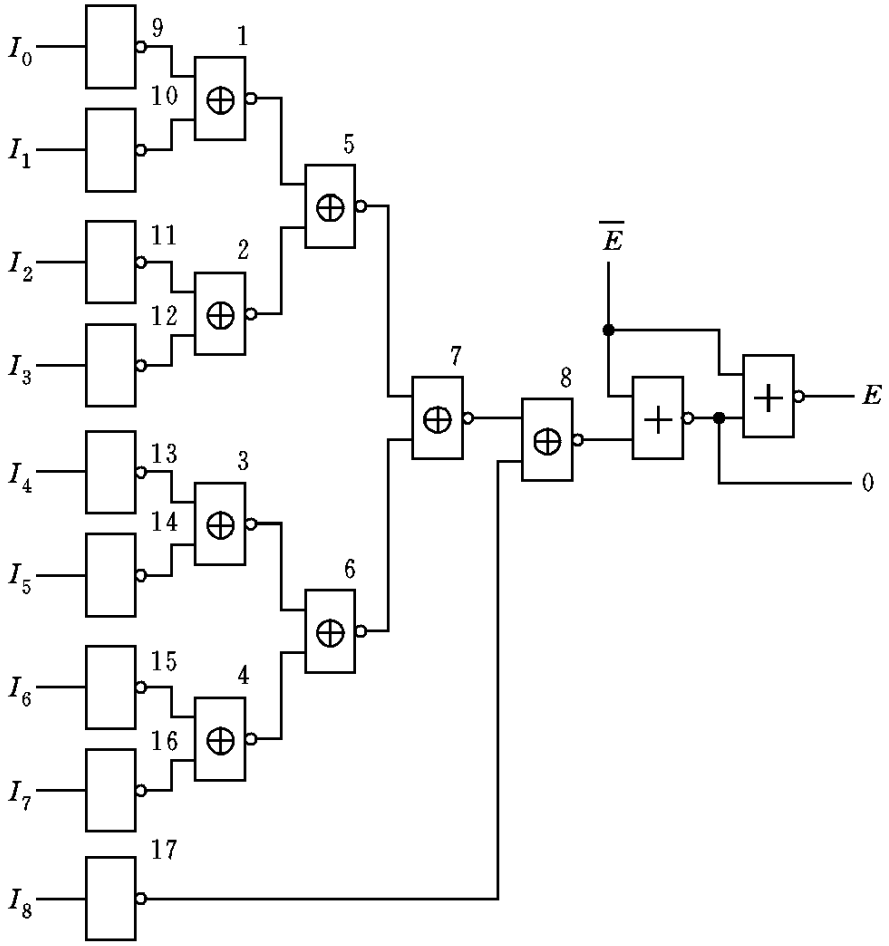
用于发送和接收的奇偶校验逻辑结构图



异或门构成八位奇偶校验位产生电路



九位奇偶检验电路


I_0
I_1
I_2
I_3
I_4
I_5
I_6
I_7
I_8
9
10
11
12
13
14
15
16
17
1
2
3
4
5
6
7
8
$\overline{E}$
E
0

3.3 常用的中规模组合逻辑电路小 结

3.3.1 译码器

3.3.2 数据选择器

3.3.3 编码器

3.3.4 数据比较器

3.3.5 奇偶校验器

3.3.X 可编程逻辑器件

3.3.6 运算器（算数逻辑单元 ALU）

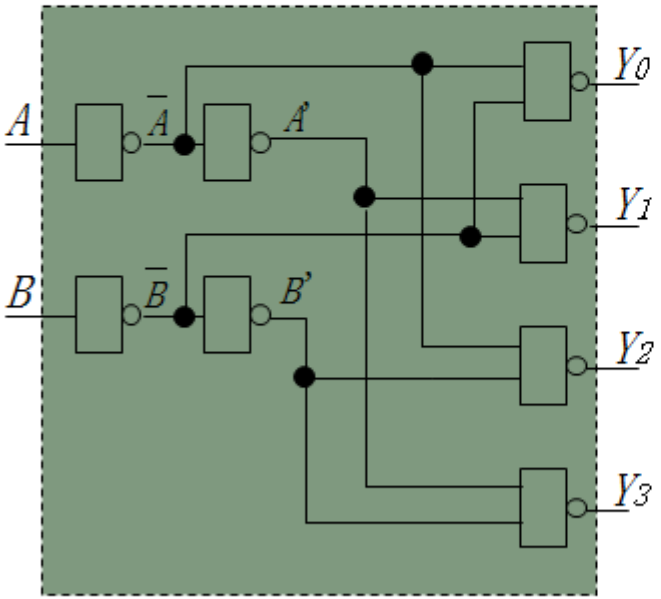
重提延时问题

延时导致了尖峰与零重叠，被称为组合逻辑电路的竞争与冒险。

3.3.1 译码器（25）

- 当AB从“11”变到“00”时，输出应从$Y_3=0$变成$Y_0=0$。

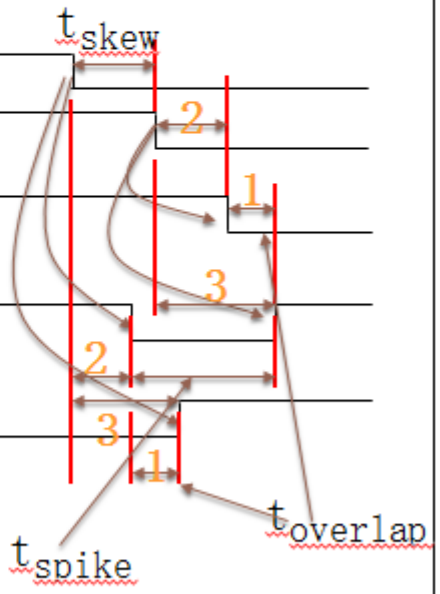
假设AB不能同时到来，存在偏移(Skew)，导致尖峰信号更宽。



$Y_0 = \overline{AB}$

$Y_2 = \overline{AB'}$

$Y_3 = \overline{A'B'}$



29

t_{spike}加宽、两处出现零重叠

$t_{overlap}$=1级延迟

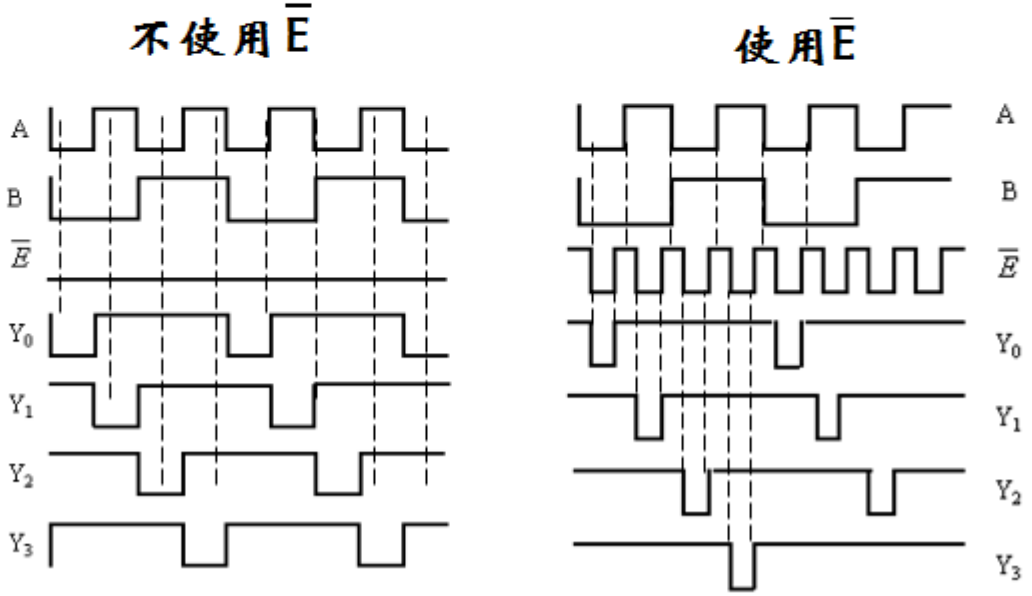
$t_{spike} = t_{skew}$+1级延迟

a）除了加使能端（$\overline{E}$），还有别的办法吗？

b）组合逻辑电路中的延时一定会导致错误吗？

3.3.1 译码器（27）

使用$\overline{E}$来抑制零重叠和尖峰，译码器的输出波形变窄了.



31

组合逻辑电路的竞争与冒险

竞争：由于延迟时间的影响，使得输入信号经过不同路径到达输出端的时间有先有后，这一现象称为竞争。

非临界竞争-------不产生错误输出的竞争称为非临界竞争。

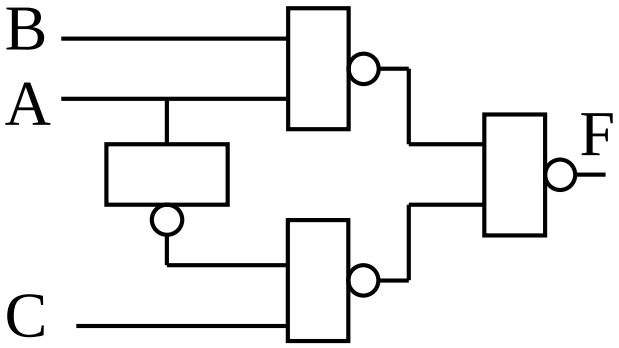
临界竞争-------导致错误输出的竞争称为临界竞争。

冒险：由竞争导致输出错误信号。

注意！组合电路中的险象是一种瞬态现象，它表现为在输出端产生不应有的尖脉冲，暂时地破坏正常逻辑关系。一旦瞬态过程结束即可恢复正常逻辑关系。

例如，如下图所示是由与非门构成的组合电路，该电路有3个输入变量，1个输出函数。

$$F = AB + \overline{A}C$$



根据逻辑电路图可写出输出函数表达式为

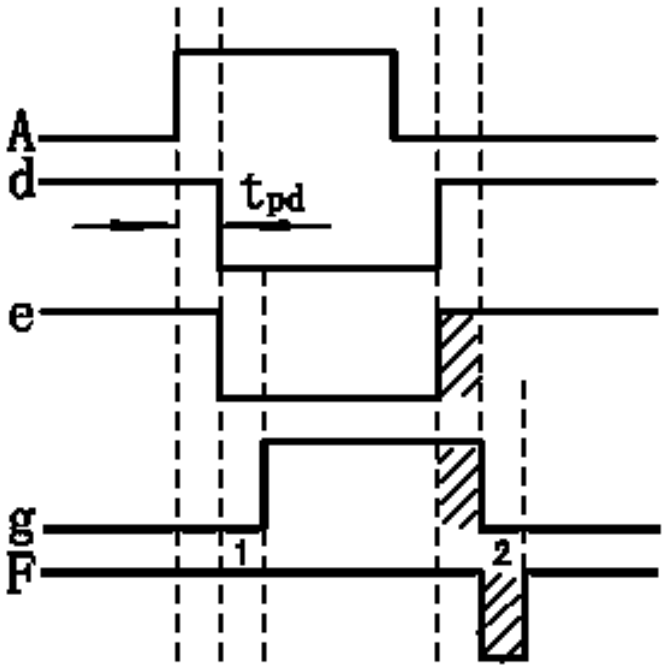
$$F = \overline{\overline{AB} \cdot \overline{\overline{A}C}} = AB + \overline{A}C$$

假设输入变量B=C=1，将B、C的值代入上述函数表达式，可得 $F = A + \overline{A}$

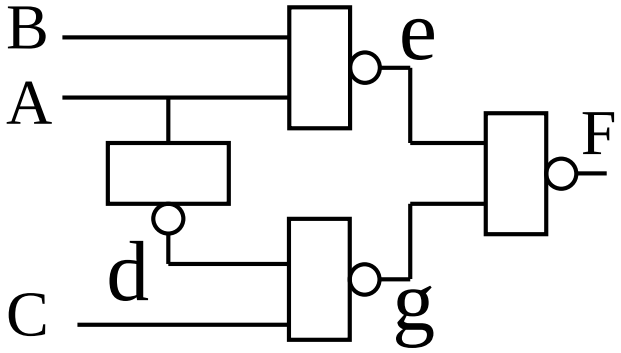
由互补律可知，函数 $F = A + \overline{A}$ 的值应恒为1，即B=C=1时，无论A怎样变化，输出F的值都应保持1不变。

当考虑电路中存在的时间延迟时，该电路的实际输入、输出关系又将怎样呢？

当B＝C ＝1时，假定每个门的延迟时间为t_{pd}，则实际输入、输出关系可用如下所示的时间图来说明。



$$F = AB + \overline{A}C$$



如果某种组合下函数同时存在A和$\overline{A}$，会出现冒险。

险象的判断

判断电路是否可能产生险象的方法有**代数法**和**卡诺图法**。

代数法：

- 检查函数表达式中是否存在具备竞争条件的变量，即是否有某个变量X同时以原变量和反变量的形式出现在函数表达式中。
- 若存在具备竞争条件的变量X，则尝试消去函数式中的其他变量，看函数表达式是否会变为 $X+\overline{X}$ 或者 $X\cdot\overline{X}$ 的形式。若会，则说明对应的逻辑电路可能产生险象。

例 已知描述某组合电路的逻辑函数表达式为 $F=\overline{A}\overline{C}+\overline{A}B+AC$，试判断该逻辑电路是否可能产生险象。

解 由表达式可知，变量A和C均具备竞争条件，所以，应对这两个变量分别进行分析。先考察变量A，为此将B和C的各种取值组合分别代入函数表达式中，可得到如下结果：

BC=00 $F=\overline{A}$

BC=01 $F=A$

BC=10 $F=\overline{A}$

BC=11 $F=A+\overline{A}$

可见，当B=C=1时，A的变化可能使电路产生险象。类似地，将A和B的各种取值组合分别代入函数表达式中，可由代入结果判断出变量C发生变化时不会产生险象。

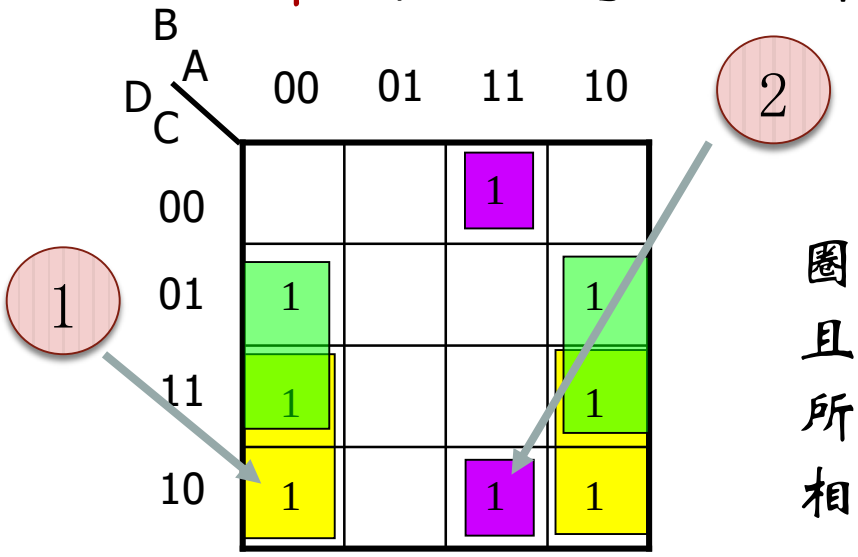
险象的判断

当描述电路的逻辑函数为“与-或”表达式时，采用卡诺图判断险象比代数法更为直观、方便。

卡诺图法：作出函数卡诺图，并画出和函数表达式中各“与”项对应的卡诺圈。若卡诺圈之间存在“相切”关系，即两卡诺圈之间存在不被同一卡诺圈包含的相邻最小项，则该电路可能产生险象。

例 已知某逻辑电路对应的函数表达式为 $F=\overline{A}D+\overline{A}C+AB\overline{C}$ 试判断该电路是否可能产生险象。

解 作出给定函数的卡诺图，如下图所示。



由卡诺图可知，卡诺圈 1 和卡诺圈 2 之间存在相邻最小项m_{10}和m_{11}，且m_{10}和m_{11}不被同一卡诺圈所包含，所以这两个卡诺圈“相切”。这说明相应电路可能产生险象。

所得结论可用代数法进行验证，假定 B=D=1，C=0，代入函数表达式F之后可得 $F=A+\overline{A}$，可见相应电路可能由于A的变化而产生险象。

险象的消除

如何消除或避开电路中可能出现的险象？

有如下几种常用的方法。

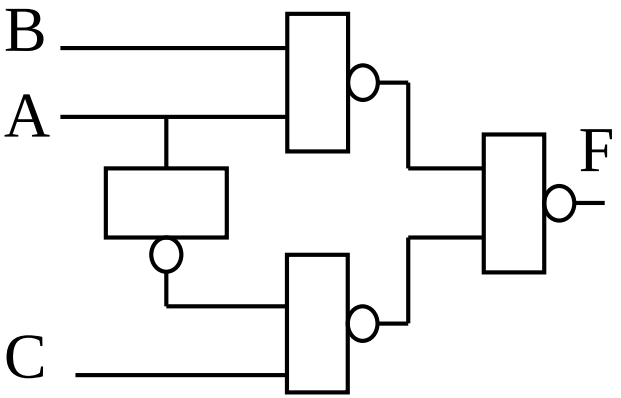
一、 用增加冗余项的方法消除险象

增加冗余项的方法是，通过在函数表达式中“或”上冗余的“与”项或者“与”上冗余的“或”项，消除可能产生的险象。

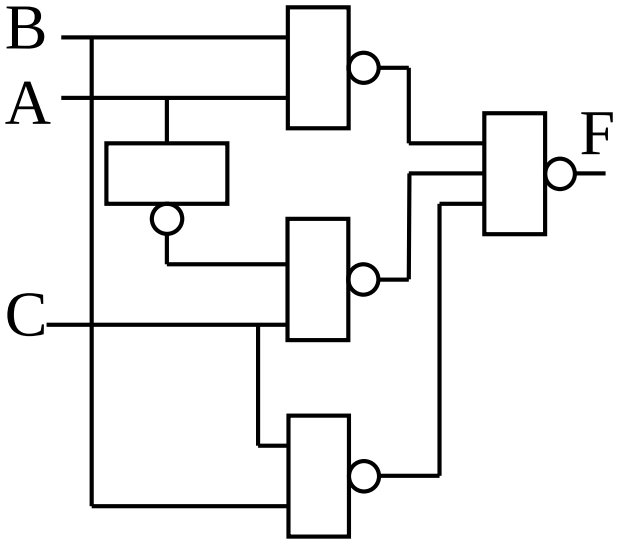
冗余项的选择可以采用代数法或者卡诺图法确定。

消除尖峰的方法：增加冗余项

$$F = AB + \bar{A}C$$



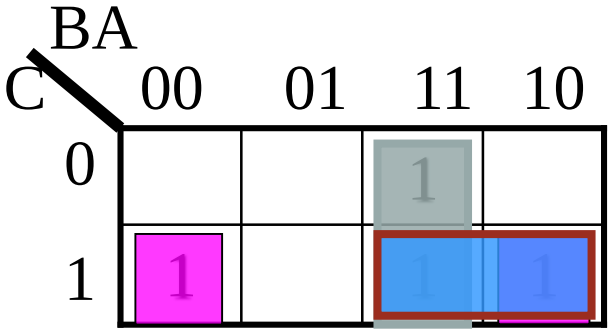
$$F = AB + \bar{A}C + BC$$



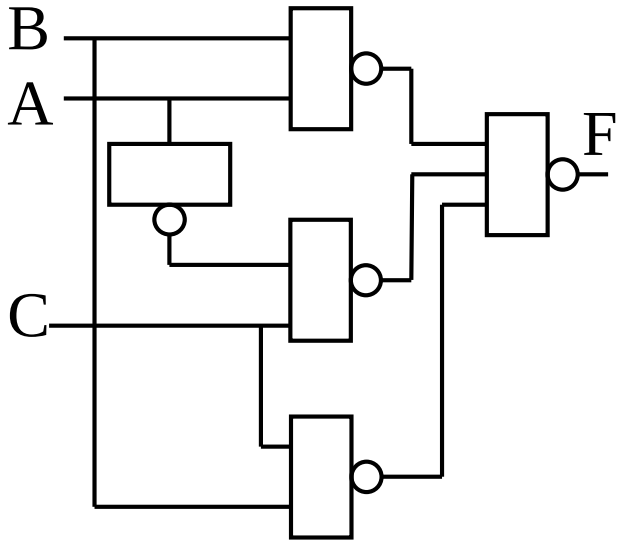
当B=C=1时 $F = A + \bar{A}$

卡诺图相切

在F中增加条件 BC



增加冗余项后的逻辑电路如下图所示。



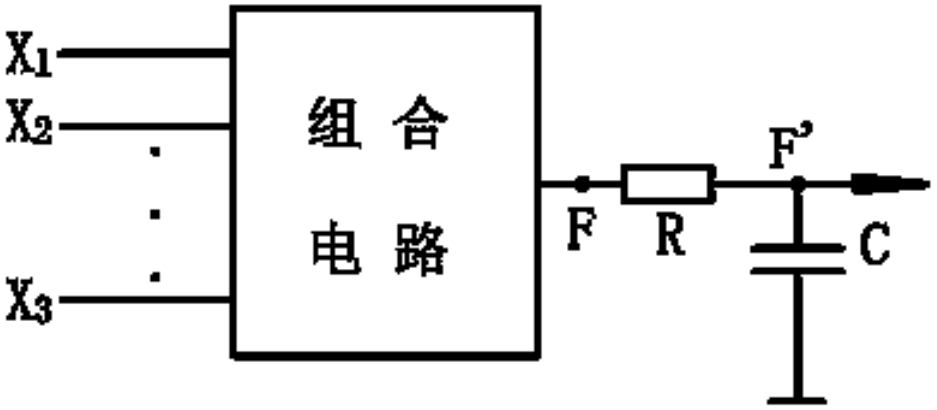
当只有一个输入变量发生变化时，该组合逻辑电路不再产生险象。

冗余项的选择也可以通过在函数卡诺图上增加多余的卡诺圈来实现。

具体方法：若卡诺图上某两个卡诺圈“相切”，则用一个多余的卡诺圈将它们之间的相邻最小项圈起来，与多余卡诺圈对应的“与”项即为要加入函数表达式中的冗余项。

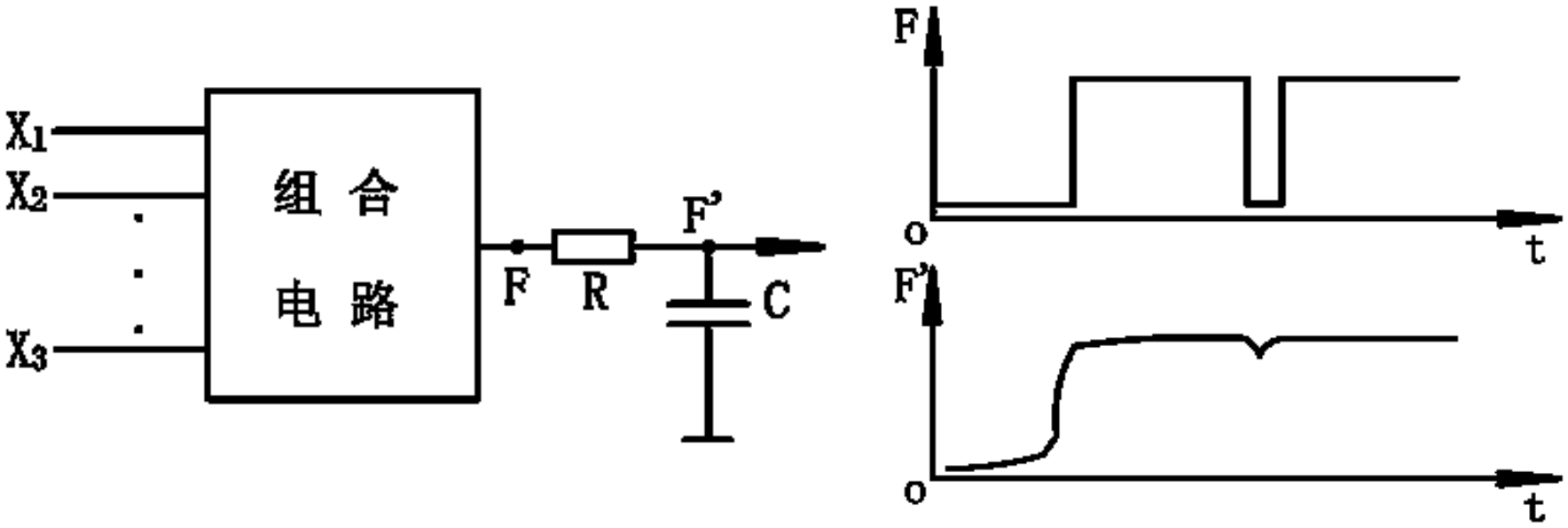
二、 电容滤波法

消除险象的另一种方法是在组合电路输出端连接一个惯性延时环节。通常采用RC电路作惯性延时环节，如图所示。



图中的RC电路实际上是一个低通滤波器。由于竞争引起的险象都是一些频率很高的尖脉冲信号，因此，险象在通过RC电路后能基本被滤掉，保留下来的仅仅是一些幅度极小的毛刺，它们不再对电路的可靠性产生影响

输出信号经滤波后的效果如下图所示。



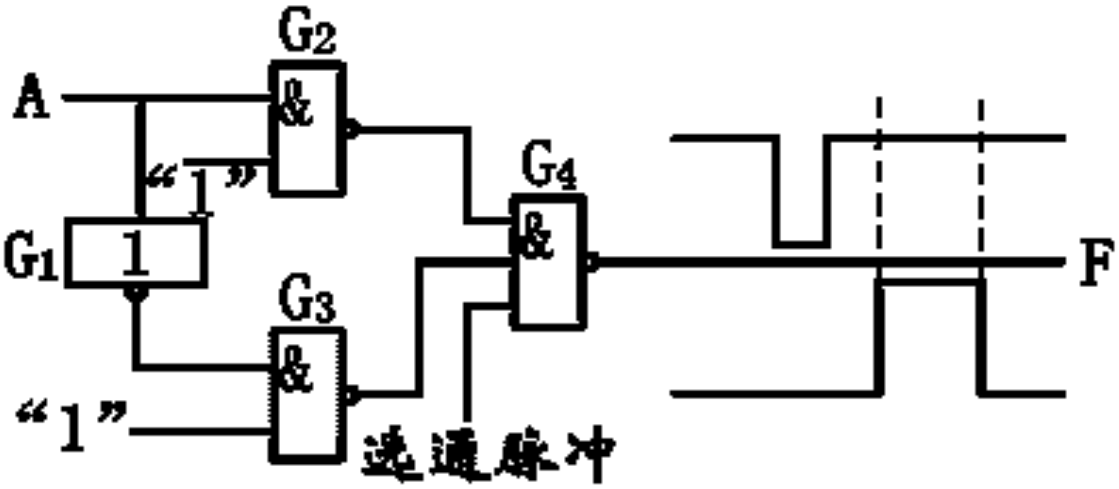
注意： 采用这种方法时，必须适当选择惯性环节的时间常数($\tau = RC$)，一般要求τ大于尖脉冲的宽度，以便能将尖脉冲“削平”；但也不能太大，否则将使正常输出信号产生不允许的畸变。

三、 选通法

选通法不必增加任何器件，仅仅是利用选通脉冲的作用，从时间上加以控制，使输出避开险象脉冲。

例如，下图所示与非门电路的输出函数表达式为

$$F = \overline{\overline{A \cdot 1} \cdot \overline{\overline{A} \cdot 1}} = A + \overline{A}$$



该电路当A发生变化时，可能产生“0”型险象。但通过选通脉冲对电路的输出门加以控制，令选通脉冲在电路稳定后出现，则可使输出避开险象脉冲，送出稳定输出信号。

组合逻辑电路的竞争与冒险

消除竞争与冒险的方法

1. 增加冗余项
   - ✓ 利用卡诺图相切
   - ✓ 该方法只适用于只有一个输入变量发生变化的情况
2. 电容滤波法
3. 选通法

3.3 常用的中规模组合逻辑电路

3.3.1 译码器

3.3.2 数据选择器

3.3.3 编码器

3.3.4 数据比较器

3.3.5 奇偶校验器

⇨ 3.3.6 可编程逻辑器件

3.3.7 运算器（算数逻辑单元 ALU）

第五章 可编程逻辑电路

- “软件固化”，“以存代算”思想的体现
- 用软件设计硬件：硬件描述语言(HDL)
- 硬件设计的进步：方便、灵活、可修改设计
  - 用户可编程
  - 设计方便
  - 易于实现

中小规模可编程逻辑器件

可编程逻辑器件PLD (Programmable Logic Device)是一大类器件的总称，包括：

- ROM (Read-Only Memory) 只读存储器
- PLA (Programmable Logic Array) 可编程逻辑阵列
- PAL (Programmable Array Logic) 可编程阵列逻辑
- GAL (General Array Logic) 通用阵列逻辑

第五章 可编程逻辑电路

- 中小规模可编程逻辑器件
  - 只读存储器（ROM）
  - 可编程逻辑阵列（PLA）
  - 可编程阵列逻辑（PAL）
  - 通用阵列逻辑（GAL）

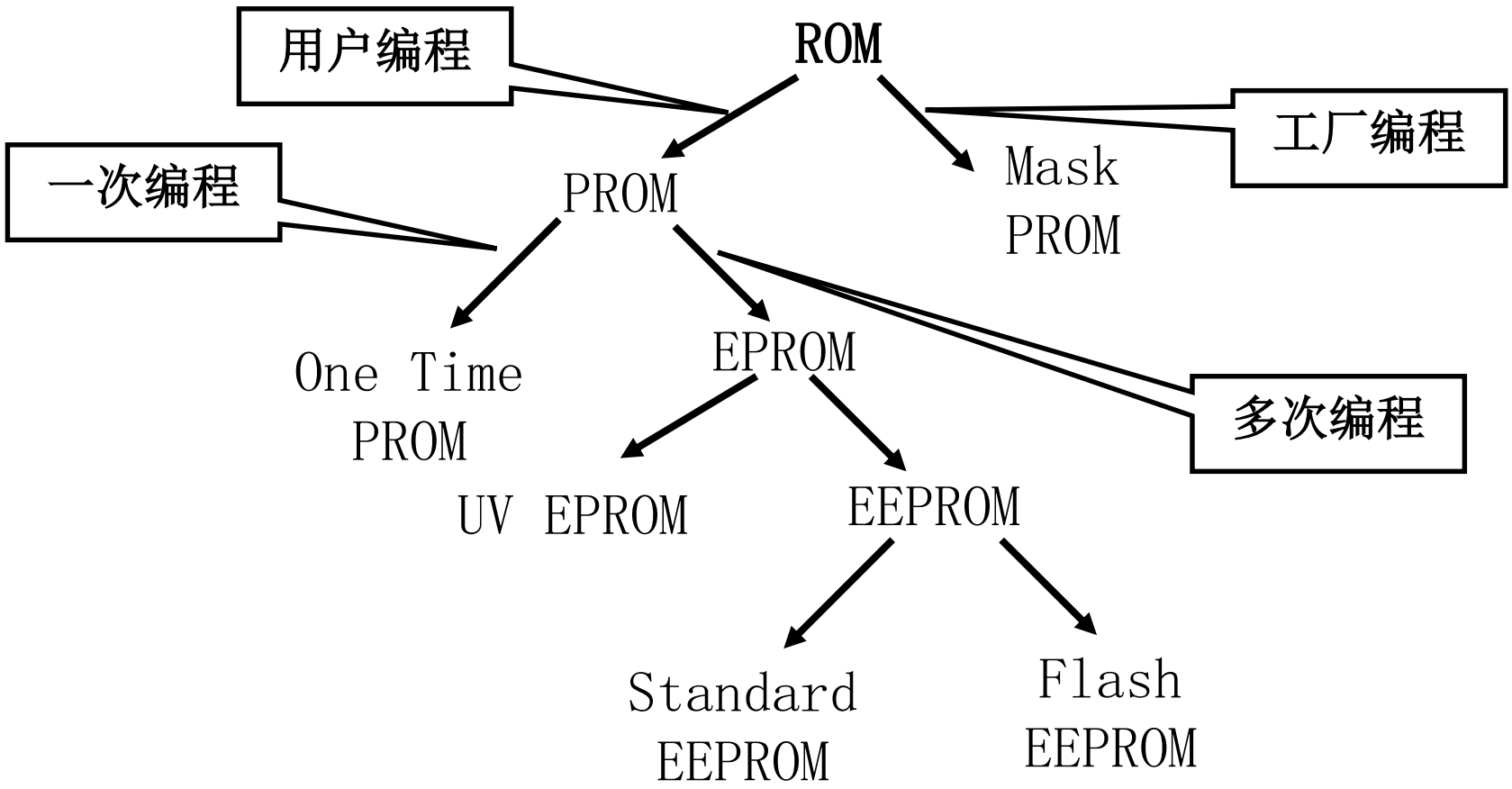
§1. ROM(只读存储器)

- 两大类存储器（Memory）
  - ROM (Read-Only Memory)
    - 一旦信息写入，在机器上只读
  - RAM (Random-Access Memory)
    - 随机存储器，在运行状态可读可写
- ROM功能
  - 存放固定信息
    - 程序，常数，指令，.....
- ROM的优点
  - 信息非“易失”(Nonvolatile)
  - 结构简单，规律性强，容量大

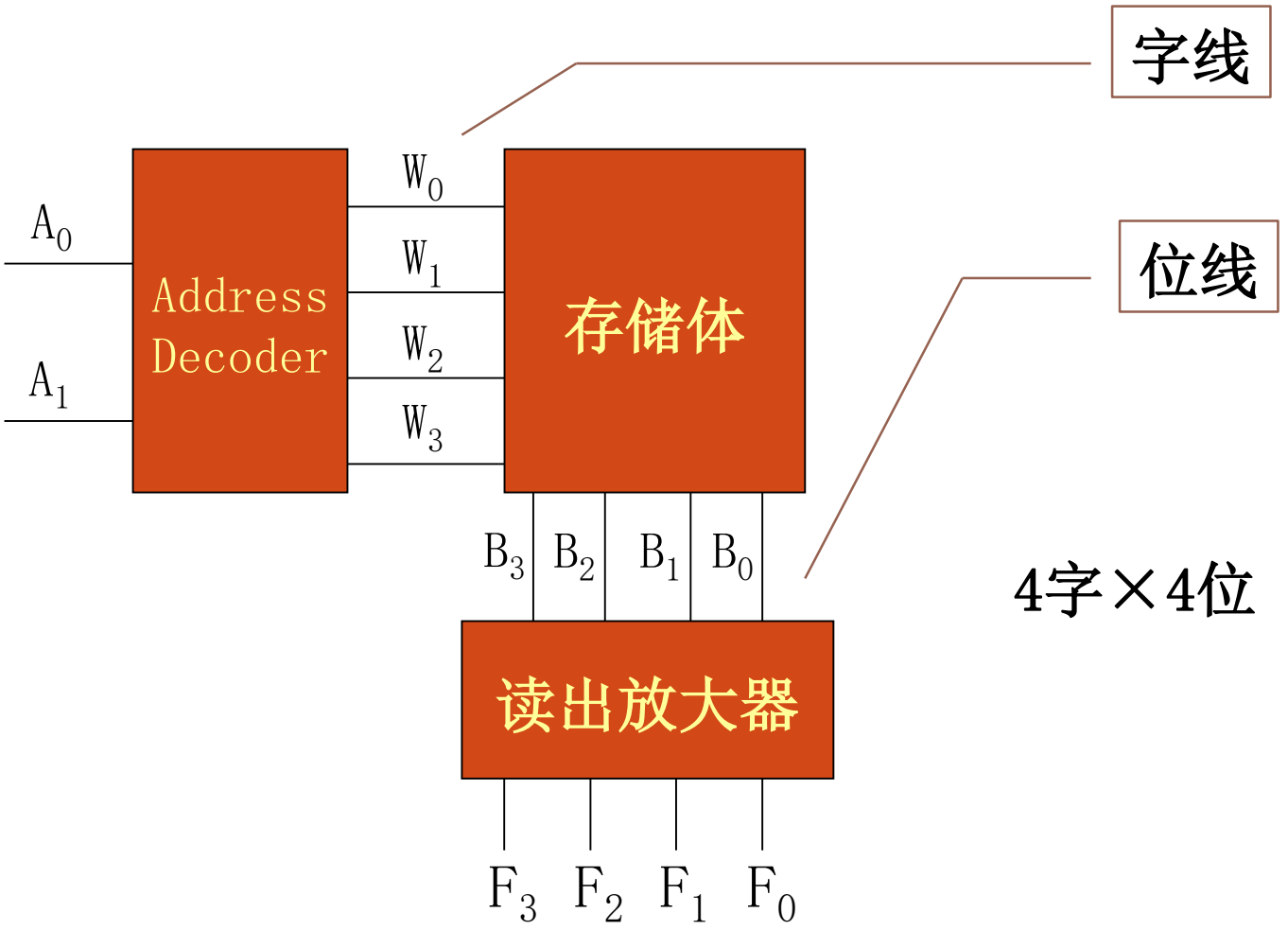
§1. ROM

- 三种ROM
  - 掩模型ROM（Mask ROM）（工厂编程）
    - 用户提交码点，在工厂编程
  - 可编程ROM（PROM）（用户一次编程）
    - 出厂保留全部熔丝，用户可编程但不可改写
  - 可改写ROM（EPROM）（用户多次编程）
    - 光可改写（UV EPROM）
    - 电可改写（E^2PROM）

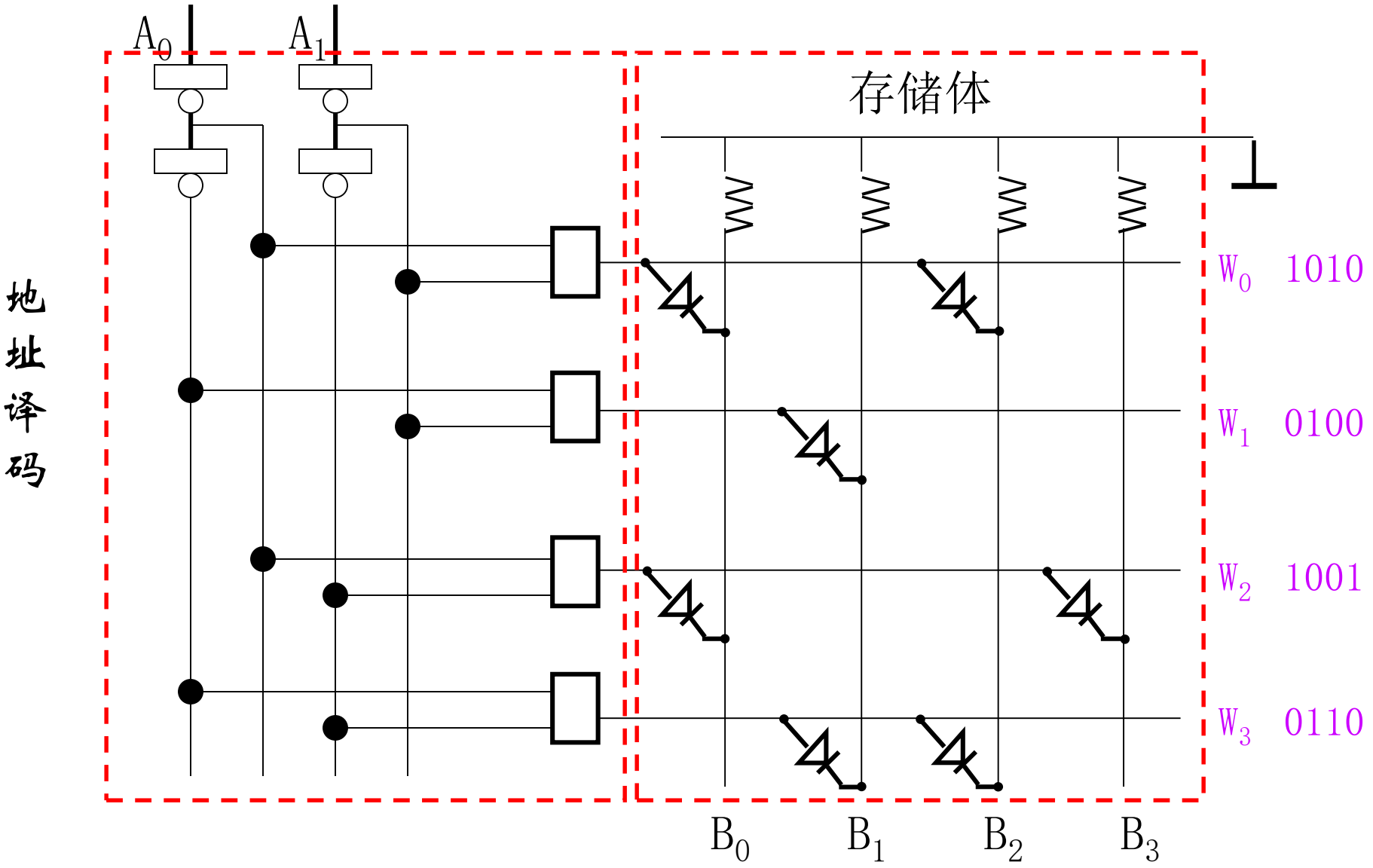
ROM分类

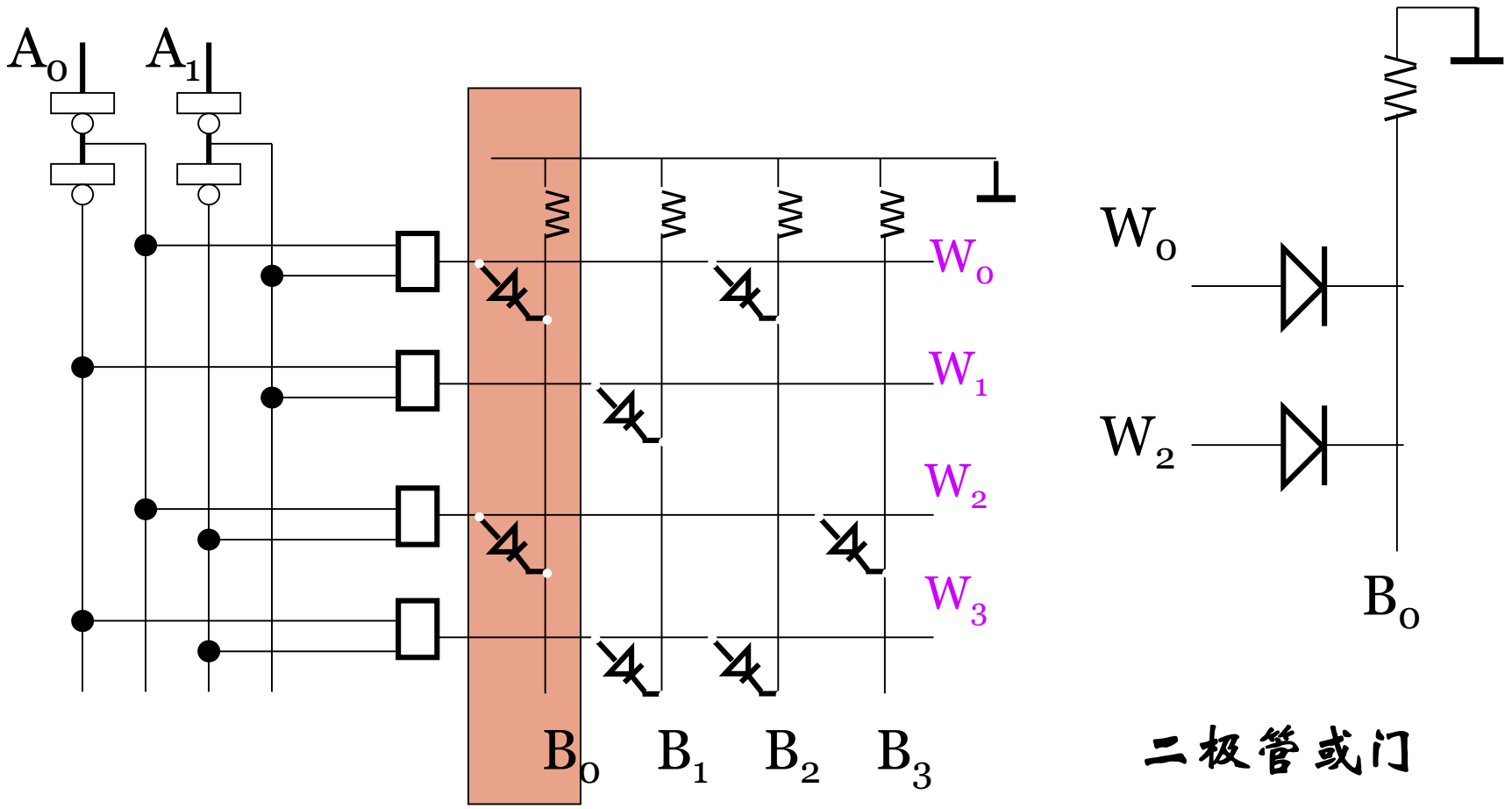

用户编程
ROM
工厂编程
一次编程
PROM
Mask
PROM
One Time
PROM
EPROM
多次编程
UV EPROM
EEPROM
Standard
EEPROM
Flash
EEPROM

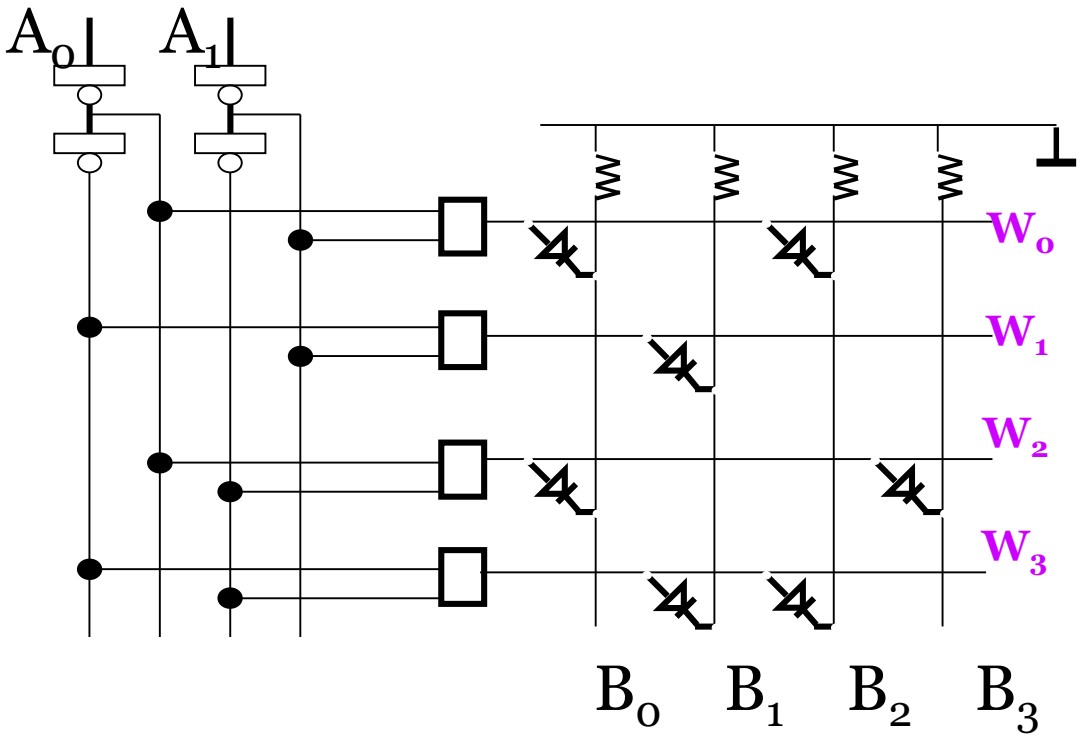
§1. ROM


字线
位线
A_0
A_1
Address Decoder
W_0
W_1
W_2
W_3
存储体
B_3 B_2 B_1 B_0
读出放大器
F_3 F_2 F_1 F_0
4字×4位

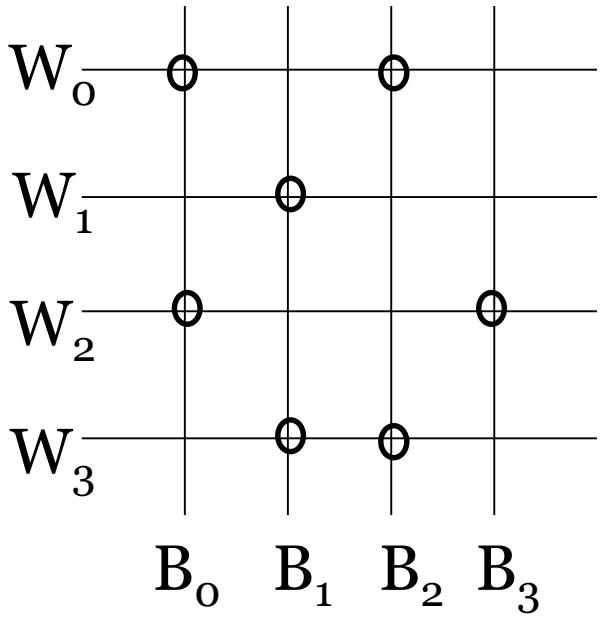
§1. ROM

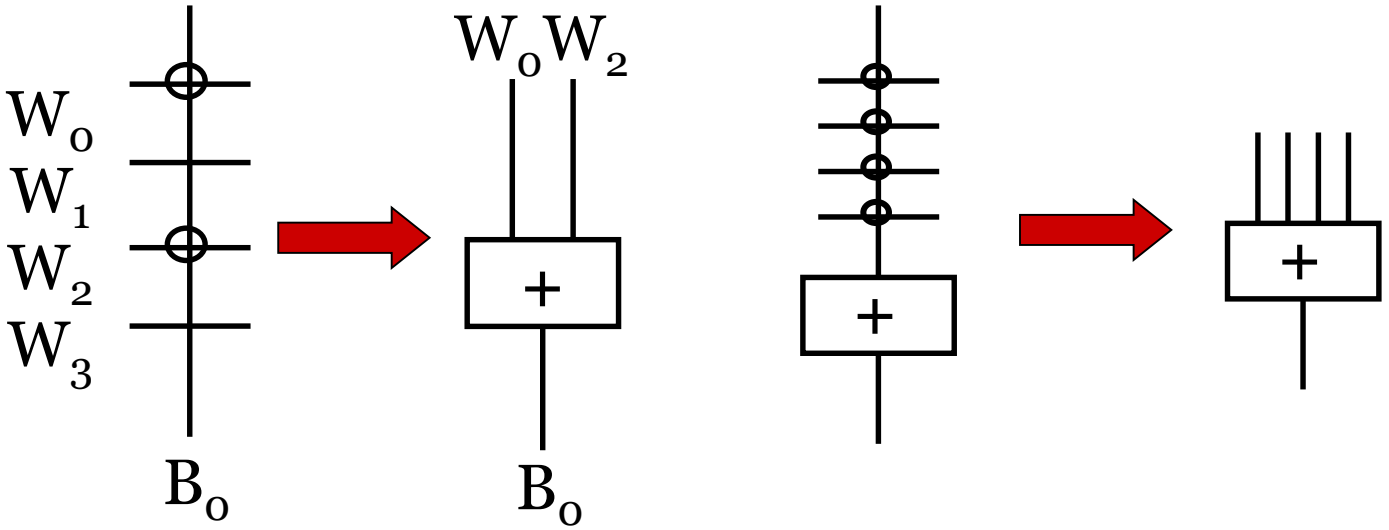



A0
A1
W0
W1
W2
W3
B0
B1
B2
B3
W0
W2
B0
二极管或门

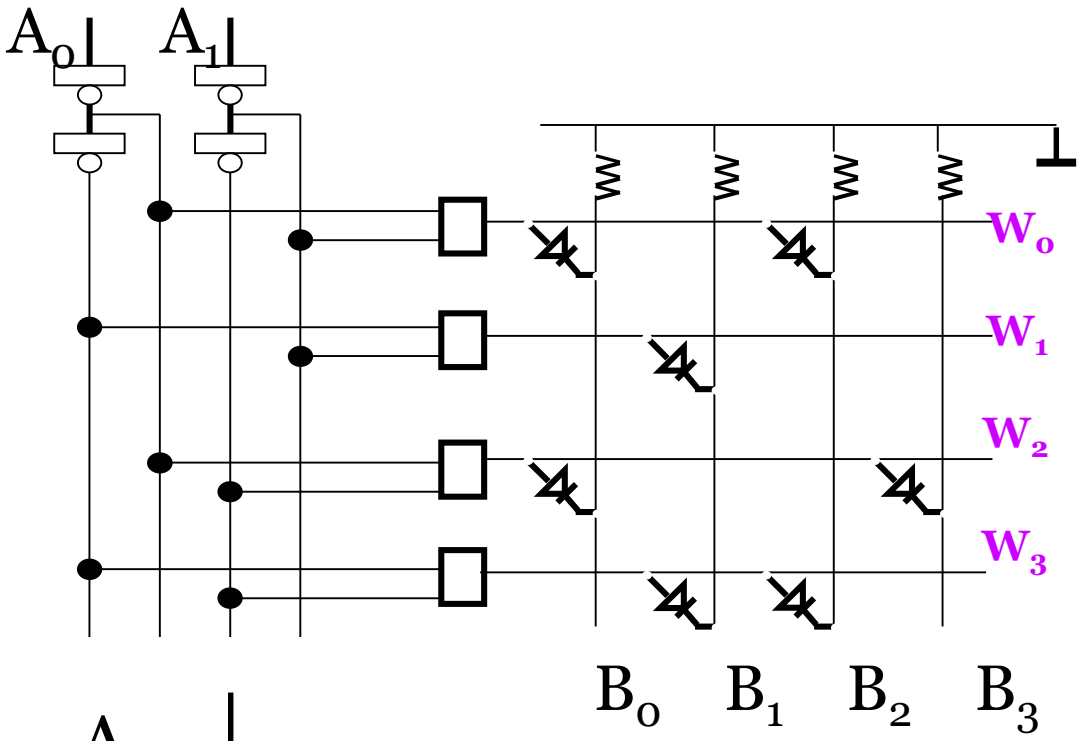

A0
A1
W0
W1
W2
W3
B0
B1
B2
B3

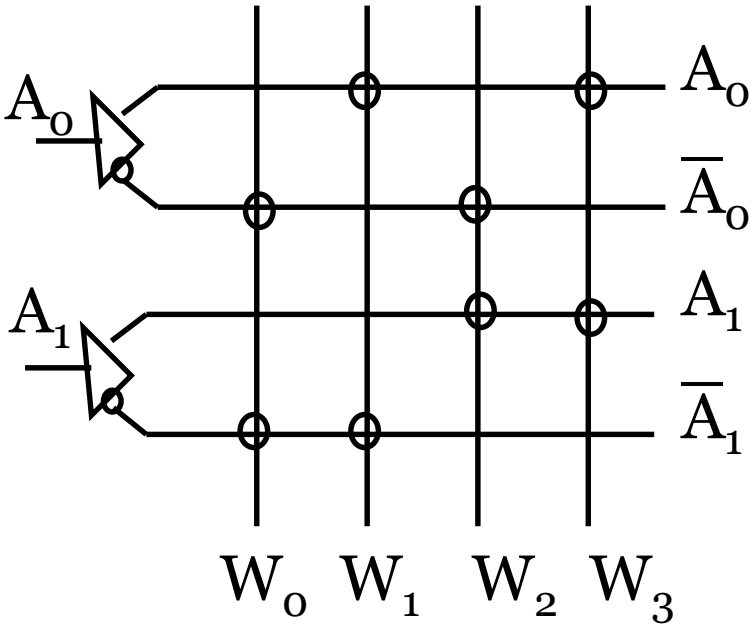
存储体可以画为：或阵列

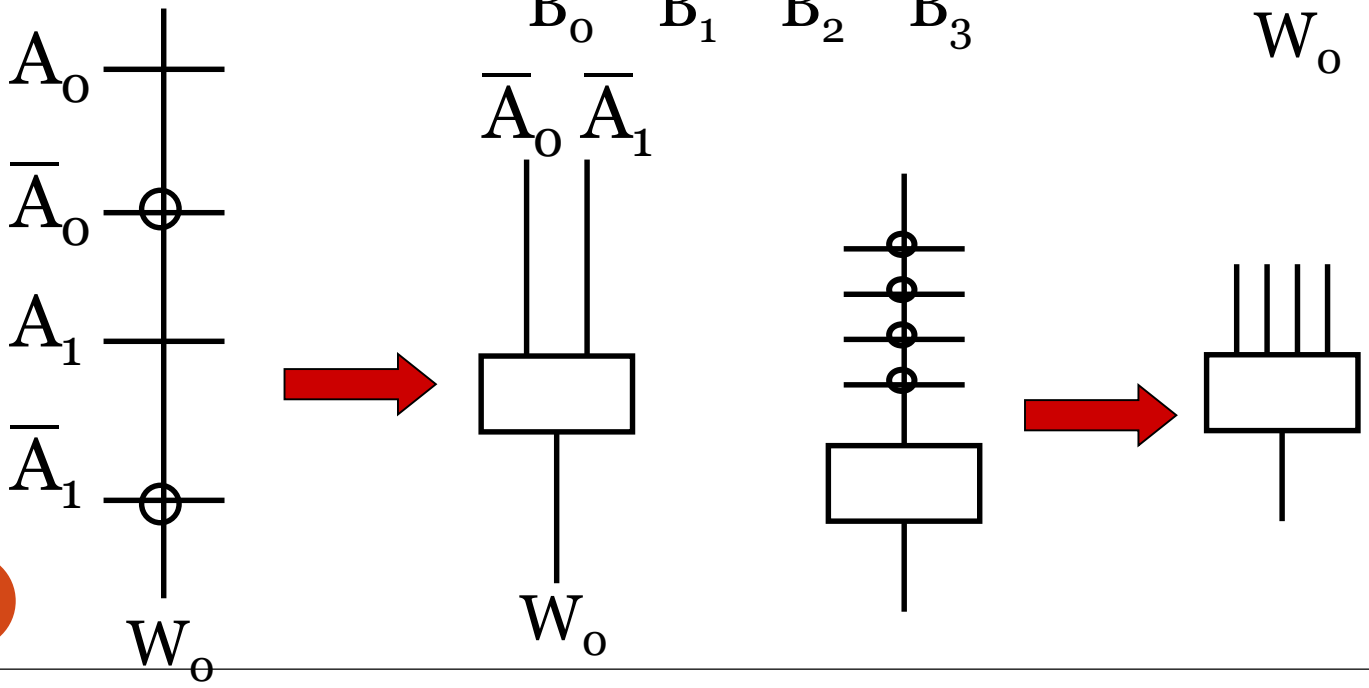

W0
W1
W2
W3
B0
B1
B2
B3


W0
W1
W2
W3
B0
W0W2
B0
+
+
+

地址译码：与阵列

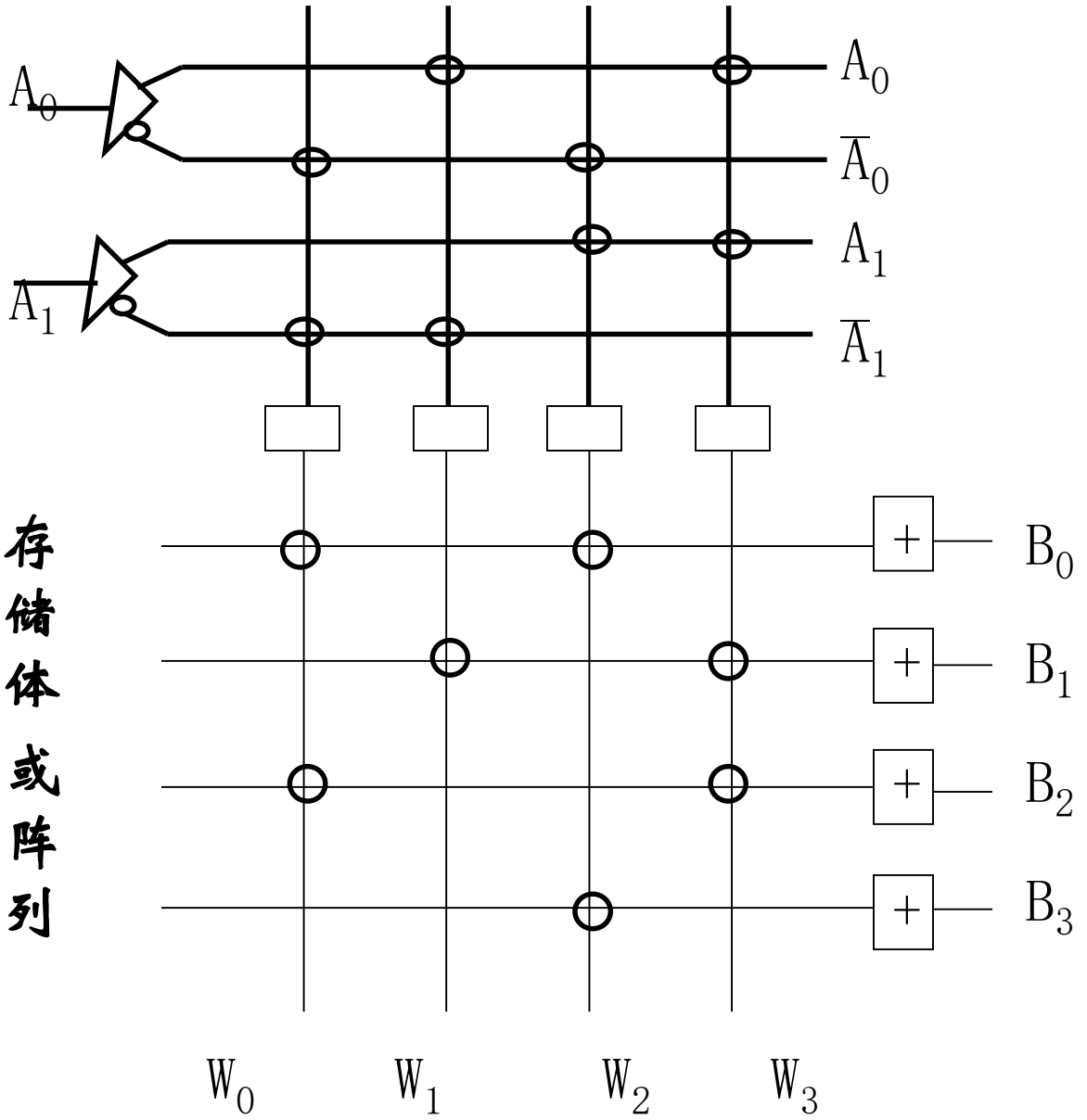

A0
A1
W0
W1
W2
W3
B0
B1
B2
B3


A0
A1
A0
A0
A1
A1
W0
W1
W2
W3


A0
A0
A1
A1
W0
A0 A1
W0

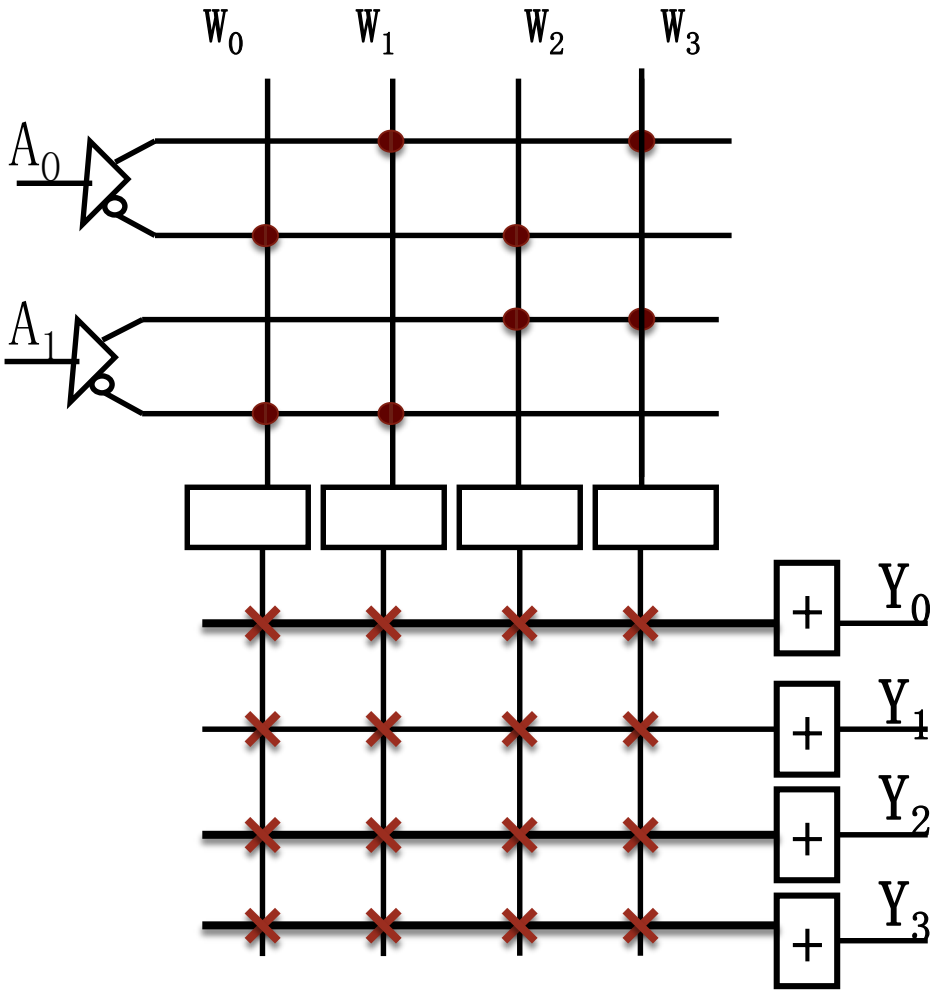
地址译码：与阵列

ROM存储器逻辑结构



PROM的结构

与阵列固定、或阵列可编程



固定连接

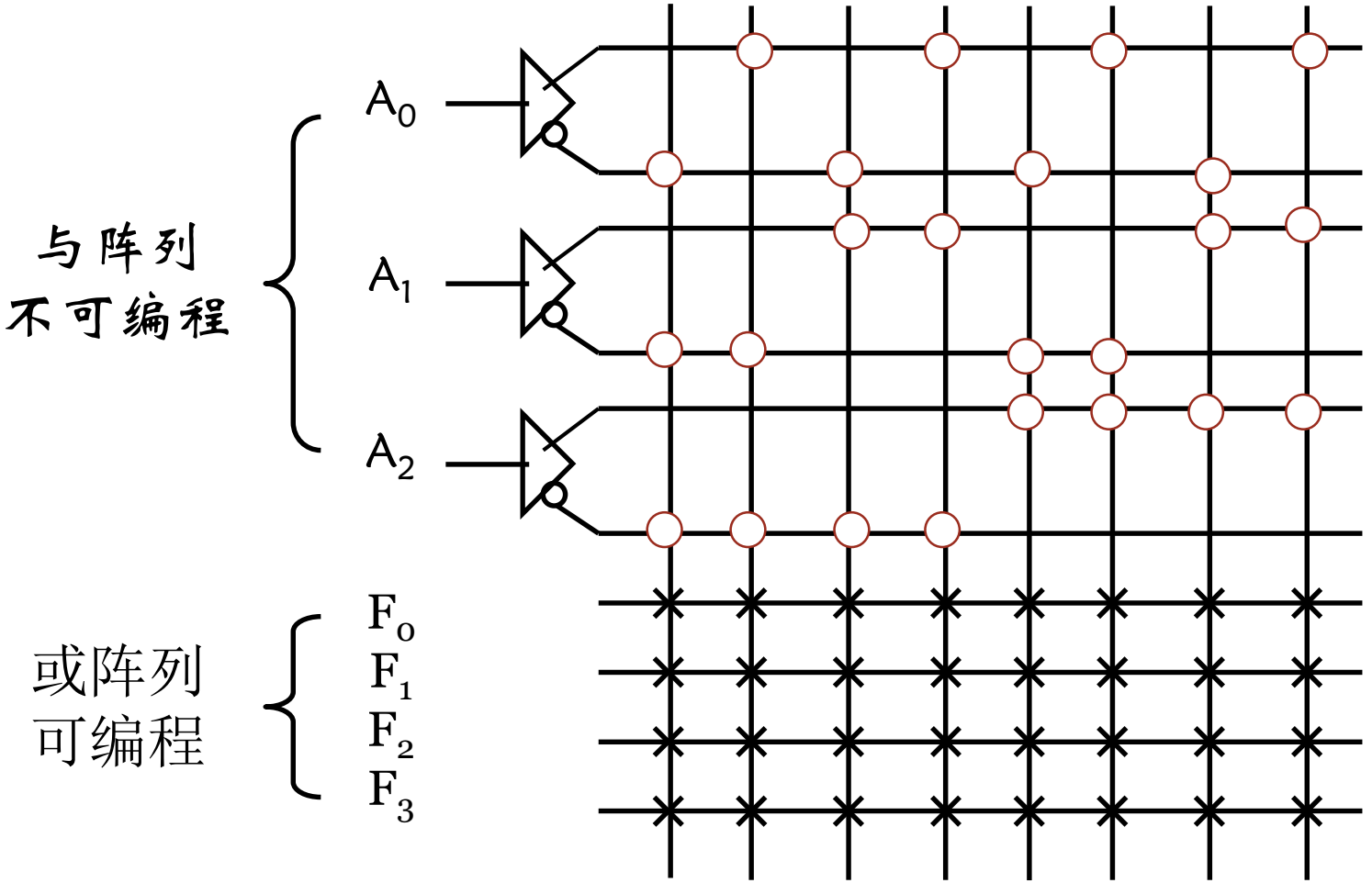
可编程连接

已编程连接

不连接

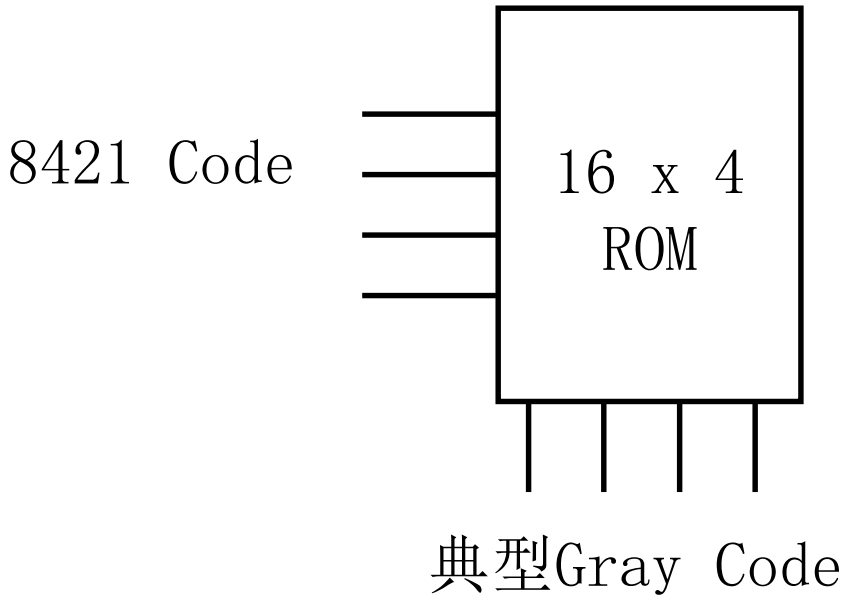
§1. ROM

例： 8x4 ROM

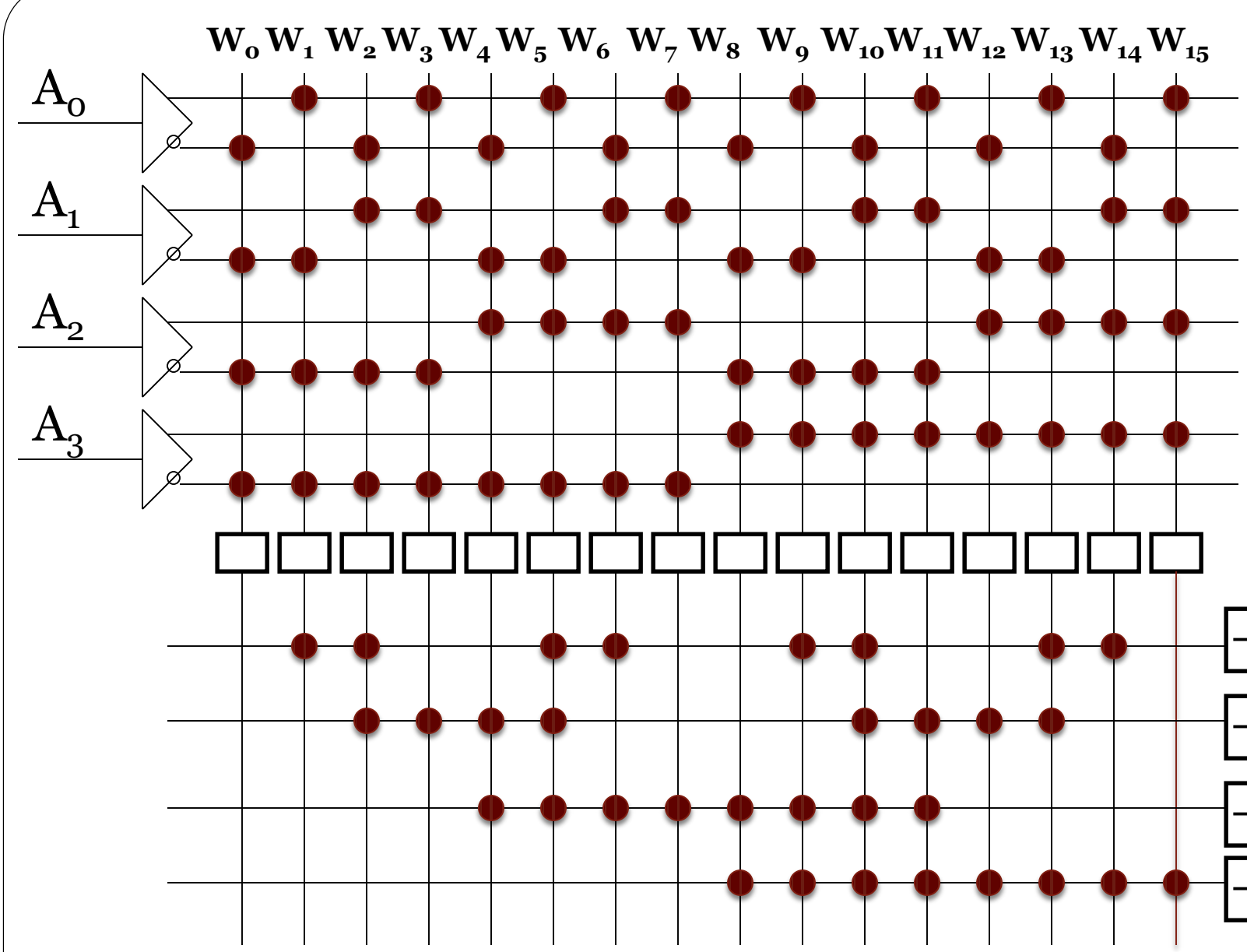


§1. ROM的应用

- 码制变换
  - 8421→典型的Gray Code



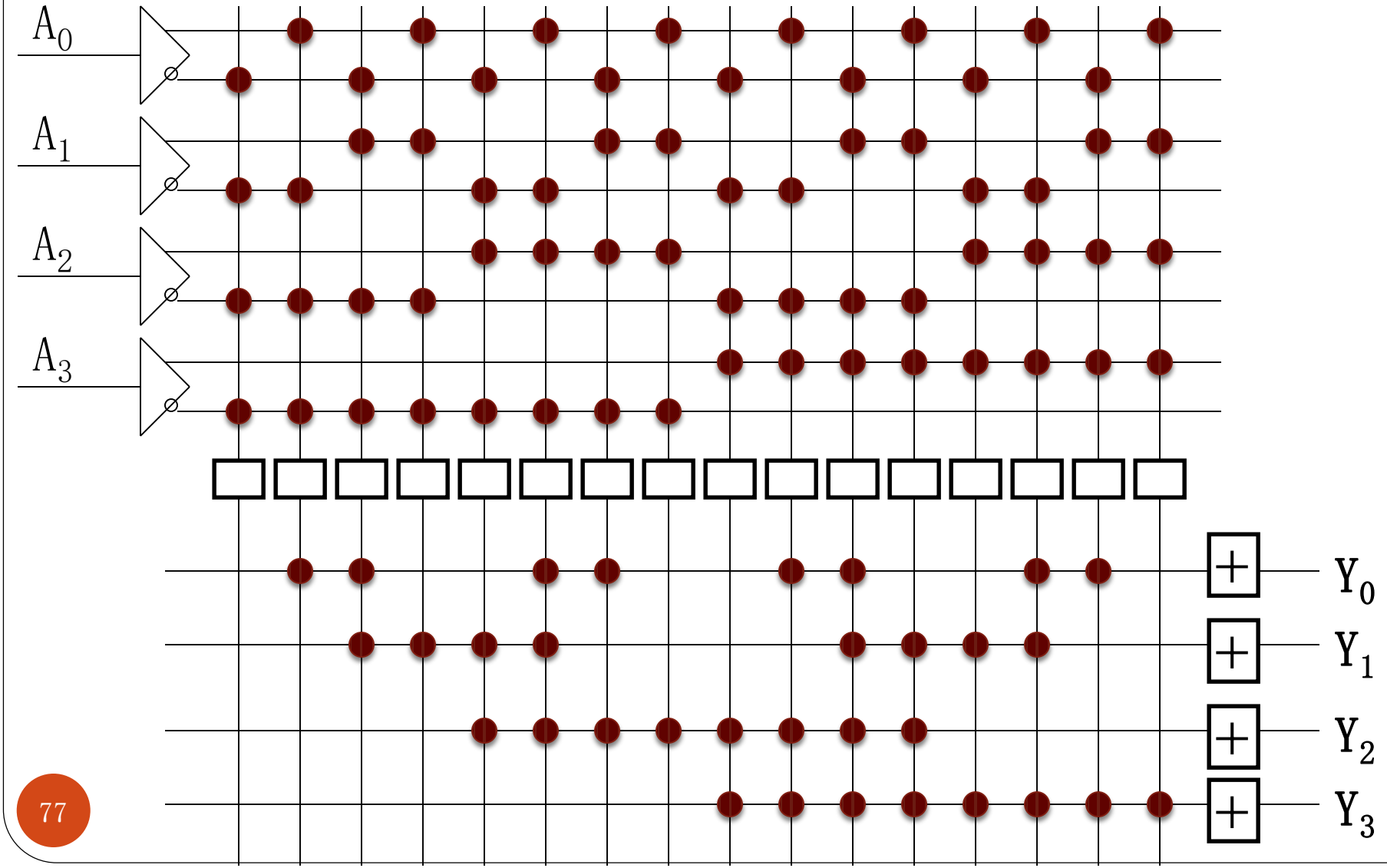
十进制数	8421码	典型格雷码
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8 ……	1000 ……	1100 ……
15	1111	1000



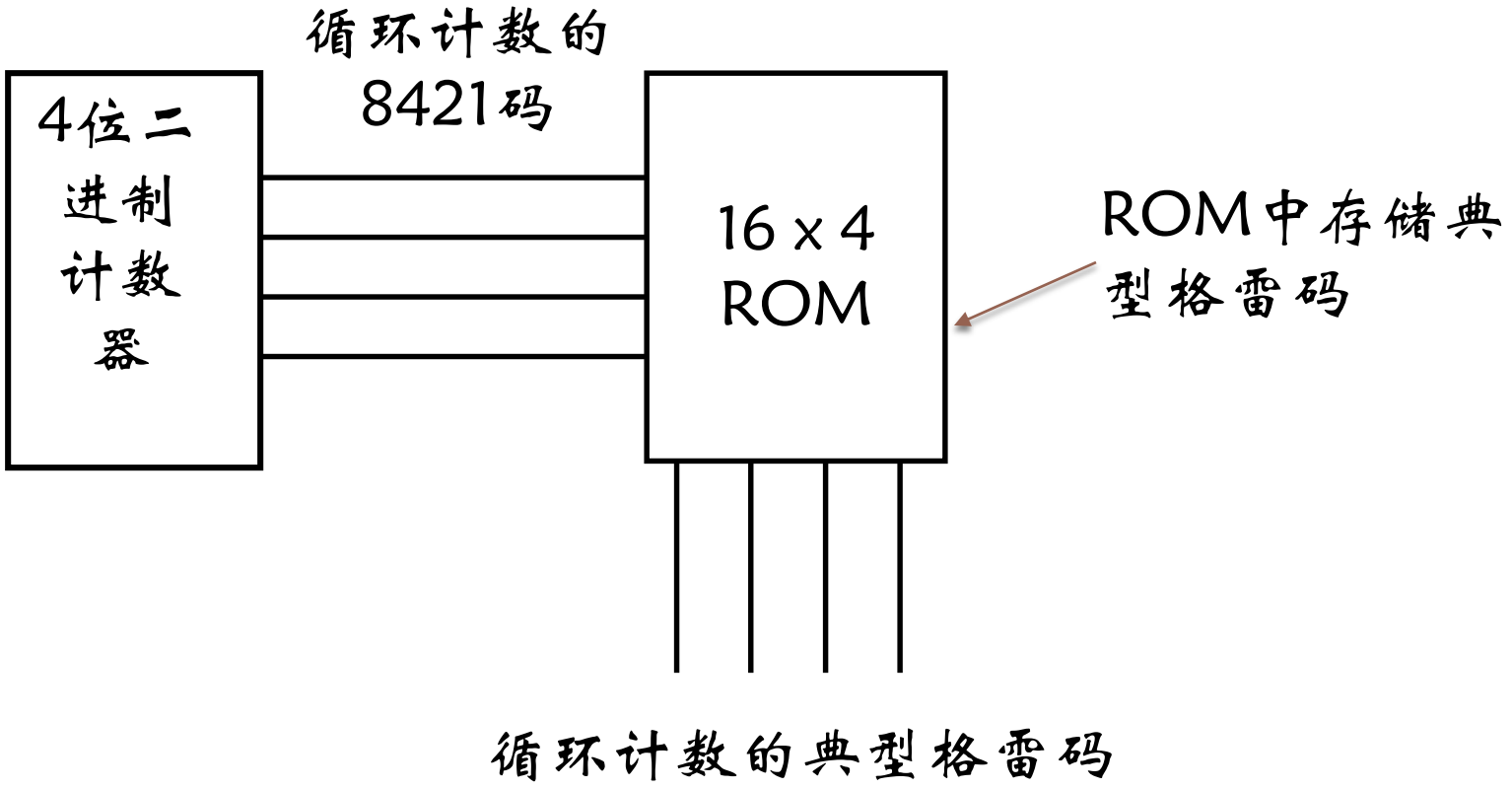
十进制数	典型格雷码
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
…	……
15	1000

与阵列8×16＝128，或阵列4×16＝64，总点数192

推荐ROM中与阵列的画法



用4位二进制计数器+ROM实现典型格雷码计数器



ROM实现--三位二进制数的平方

- ROM是用与门构成的变量译码器及或门组合而成
- 将真值表存储在ROM里实现组合函数的功能
- 可以用软件方式来建立布尔函数的信息

Inputs			Outputs						
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

B0与A0相同

B1为恒”0”

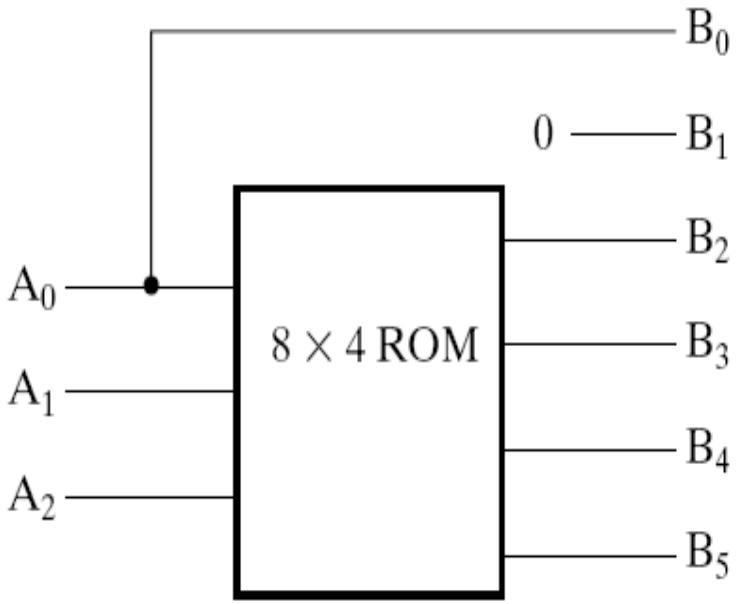
可选择8×4bit的ROM

ROM真值表---三位二进制平方

选择8×4bit的ROM

函数输入对应ROM的地址

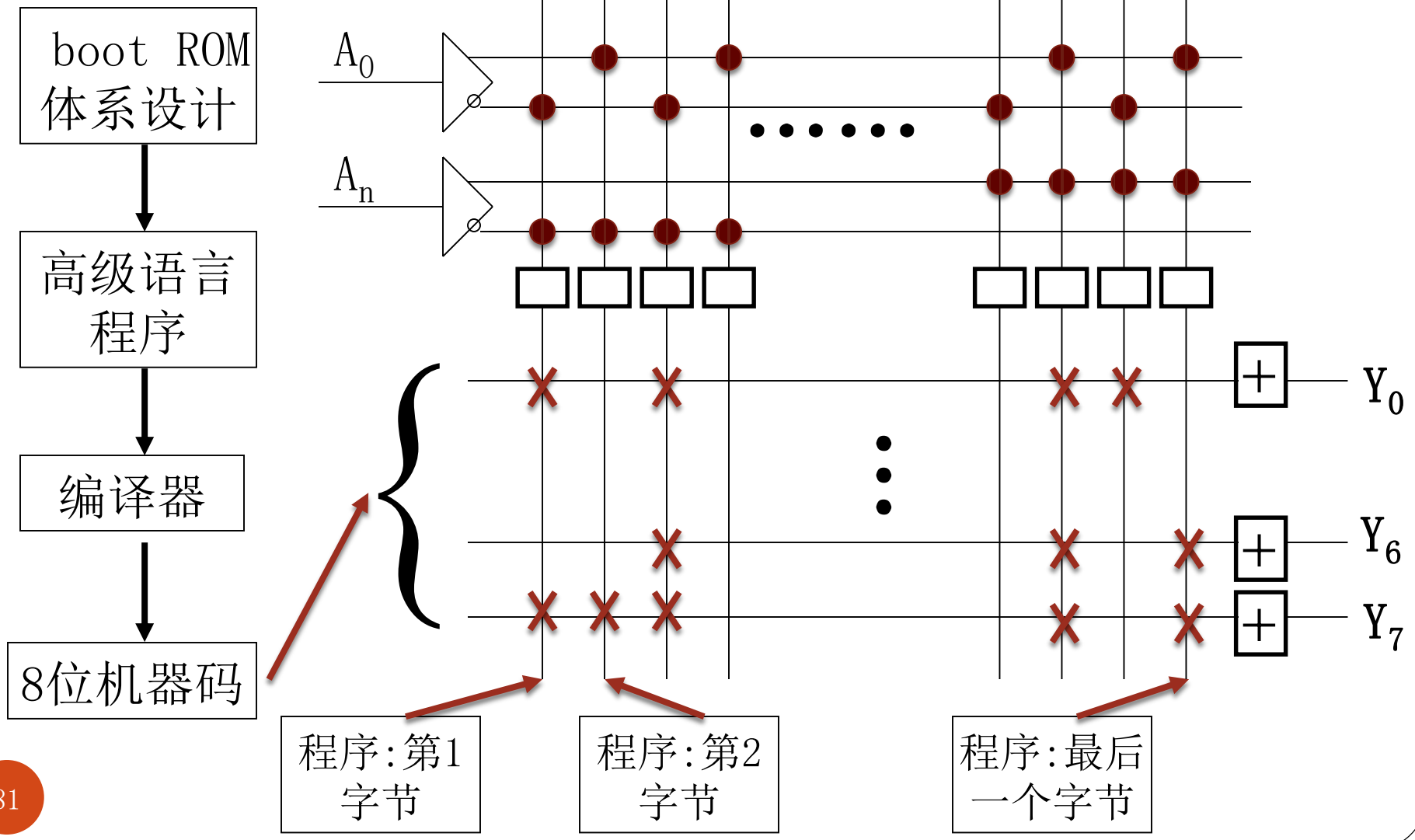
ROM输出对应平方的输出值



ROM真值表

A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

用ROM存储计算机程序



第五章 可编程逻辑电路

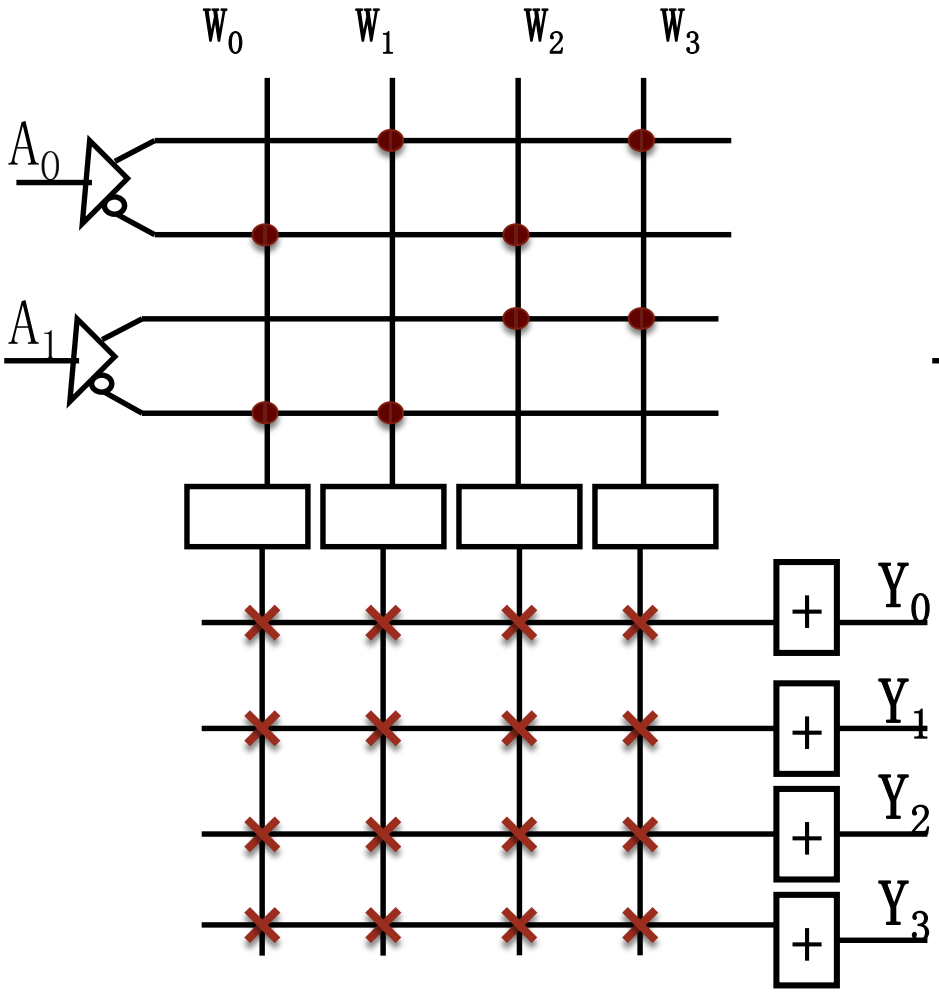
- 中小规模可编程逻辑器件
  - 只读存储器（ROM）
  - 可编程逻辑阵列（PLA）
  - 可编程阵列逻辑（PAL）
  - 通用整列逻辑（GAL）
- 大规模可编程逻辑电路

§2.可编程逻辑阵列(PLA)

- ROM的特点
  - 输入地址和存储信息一一对应。
  - 与阵列是译码器，包括全部最小项，信息表完全
- PLA针对ROM这一特点
  - 逻辑压缩

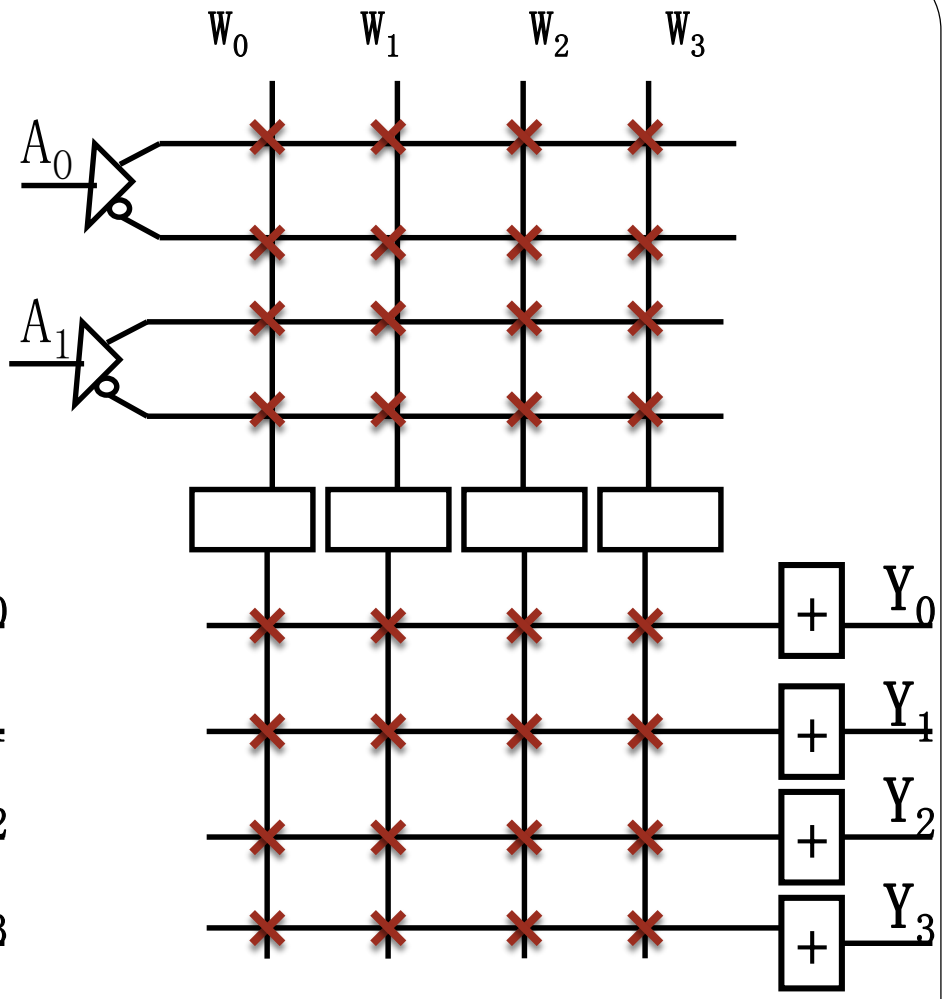
●从逻辑设计的角度

- ROM与阵列产生的全部最小项（2^n个）
- 不管实际的逻辑函数需要多少个最小项，这2^n个最小项都存在与ROM芯片中。
- 为了提高芯片的利用率，与阵列不一定产生所有的最小项，只需产生逻辑函数所需的乘积项即可。



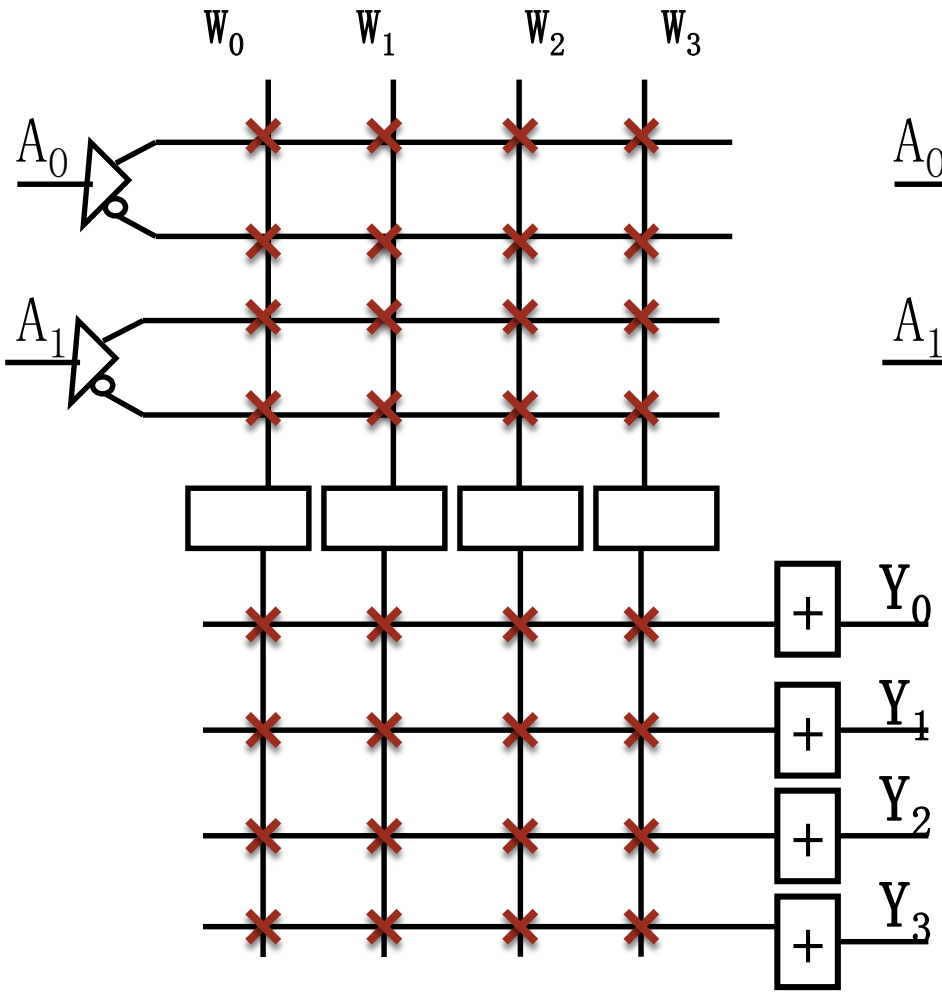
PROM

与阵列固定、或阵列可编程



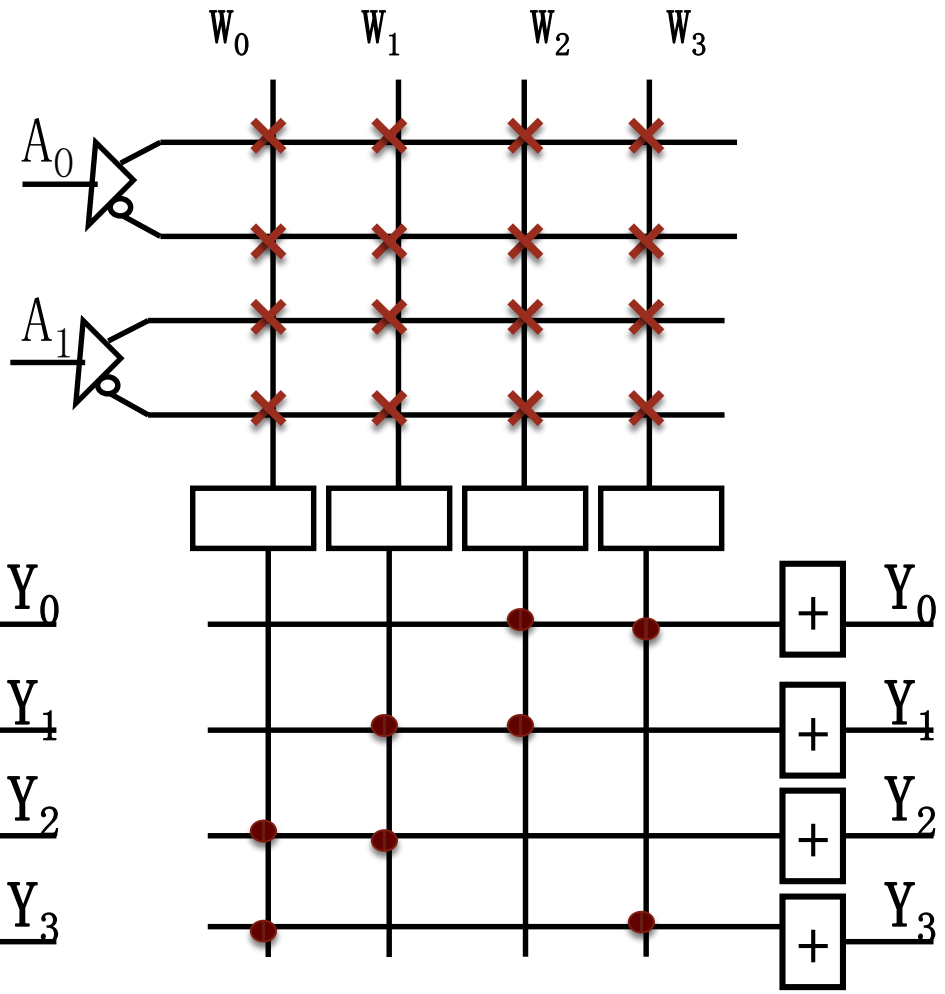
PLA

与、或阵列均可编程



PLA

与、或阵列均可编程



PAL

与阵列可编程、或阵列固定

PLA可编程逻辑阵列

- PLA实现组合逻辑
  - PLA实现组合逻辑时，将逻辑表达式写成最简与－或表达式
  - 再用与项(乘积项:product)的“或”操作构成或阵列

PLA可编程逻辑阵列

- 例：用PLA实现8421码到格雷码的转换
  - 8421码表示为$B_3B_2B_1B_0$，格雷码表示为$G_3G_2G_1G_0$。
  - 格雷码和8421码的转换关系如下：

十进制数	8421码	典型格雷码
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8 ……	1000 ……	1100 ……
15	1111	1000

PLA可编程逻辑阵列

- 例：用PLA实现8421码到格雷码的转换
  - 8421码表示为$B_3B_2B_1B_0$，格雷码表示为$G_3G_2G_1G_0$。
  - 格雷码和8421码的转换关系如下：

$$G_3 = B_3$$
$$G_2 = B_3 \oplus B_2 = \overline{B_3}B_2 + B_3\overline{B_2}$$
$$G_1 = B_2 \oplus B_1 = \overline{B_2}B_1 + B_2\overline{B_1}$$
$$G_0 = B_1 \oplus B_0 = \overline{B_1}B_0 + B_1\overline{B_0}$$

$$P_0 = B_3$$
$$P_1 = \overline{B_3}B_2$$
$$P_2 = B_3\overline{B_2}$$
$$P_3 = \overline{B_2}B_1$$
$$P_4 = B_2\overline{B_1}$$
$$P_5 = \overline{B_1}B_0$$
$$P_6 = B_1\overline{B_0}$$

将每个不同的与项用乘积项Pi表示

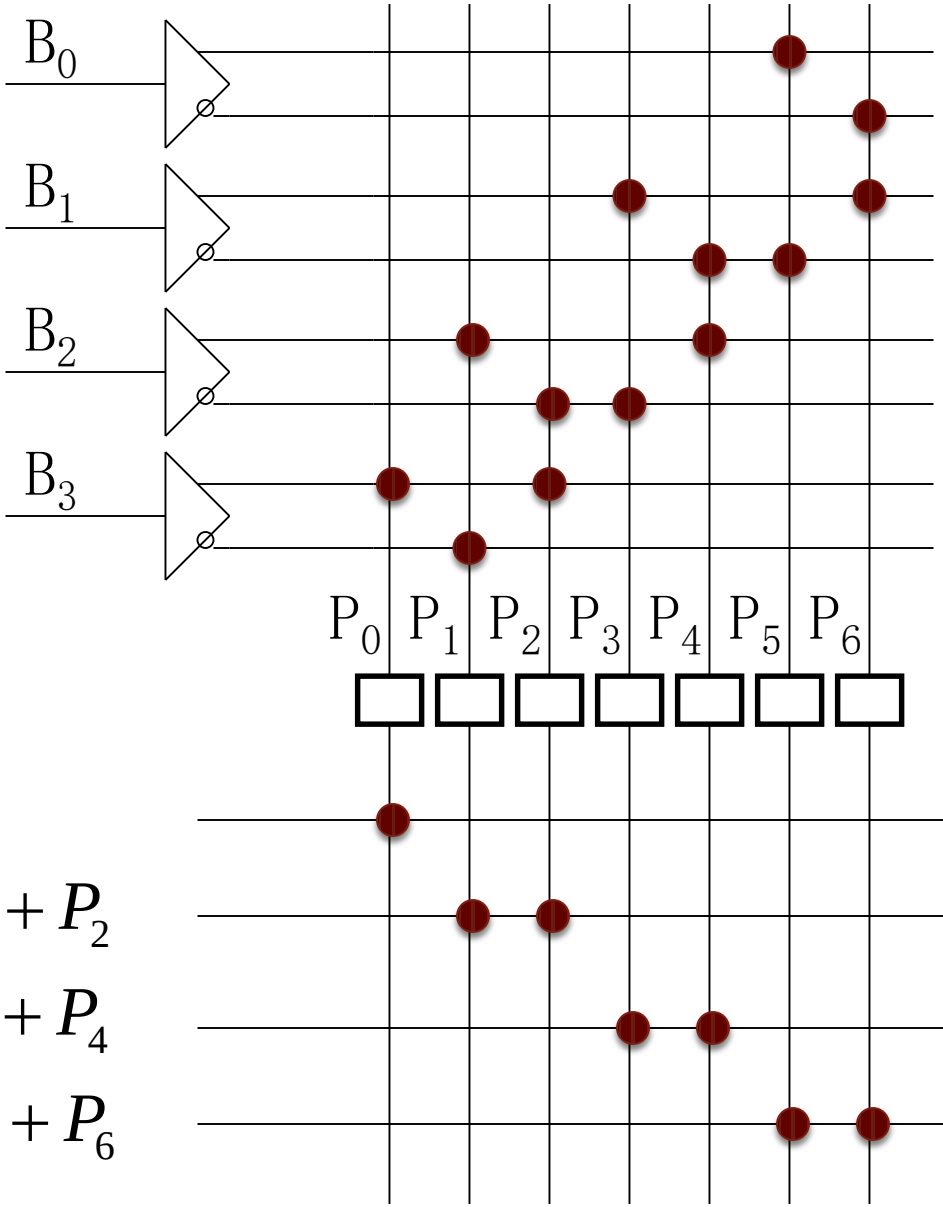
PLA可编程逻辑阵列

- 将G_i用P_i表示，可得

$$G_3 = P_0$$
$$G_2 = P_1 + P_2$$
$$G_1 = P_3 + P_4$$
$$G_0 = P_5 + P_6$$

$$P_0 = B_3$$
$$P_1 = \overline{B_3} B_2$$
$$P_2 = B_3 \overline{B_2}$$
$$P_3 = \overline{B_2} B_1$$
$$P_4 = B_2 \overline{B_1}$$
$$P_5 = \overline{B_1} B_0$$
$$P_6 = B_1 \overline{B_0}$$

P项用与阵列实现，
Gi用或阵列实现



$$P_0 = B_3$$
$$P_1 = \overline{B_3} B_2$$
$$P_2 = B_3 \overline{B_2}$$
$$P_3 = \overline{B_2} B_1$$
$$P_4 = B_2 \overline{B_1}$$
$$P_5 = \overline{B_1} B_0$$
$$P_6 = B_1 \overline{B_0}$$

$$G_3 = P_0$$
$$G_2 = P_1 + P_2$$
$$G_1 = P_3 + P_4$$
$$G_0 = P_5 + P_6$$

与阵列$8 \times 7 = 56$，或阵列$4 \times 7 = 28$，总点数84

§2 .PLA

例如：存储信息表

用16x8 ROM存储

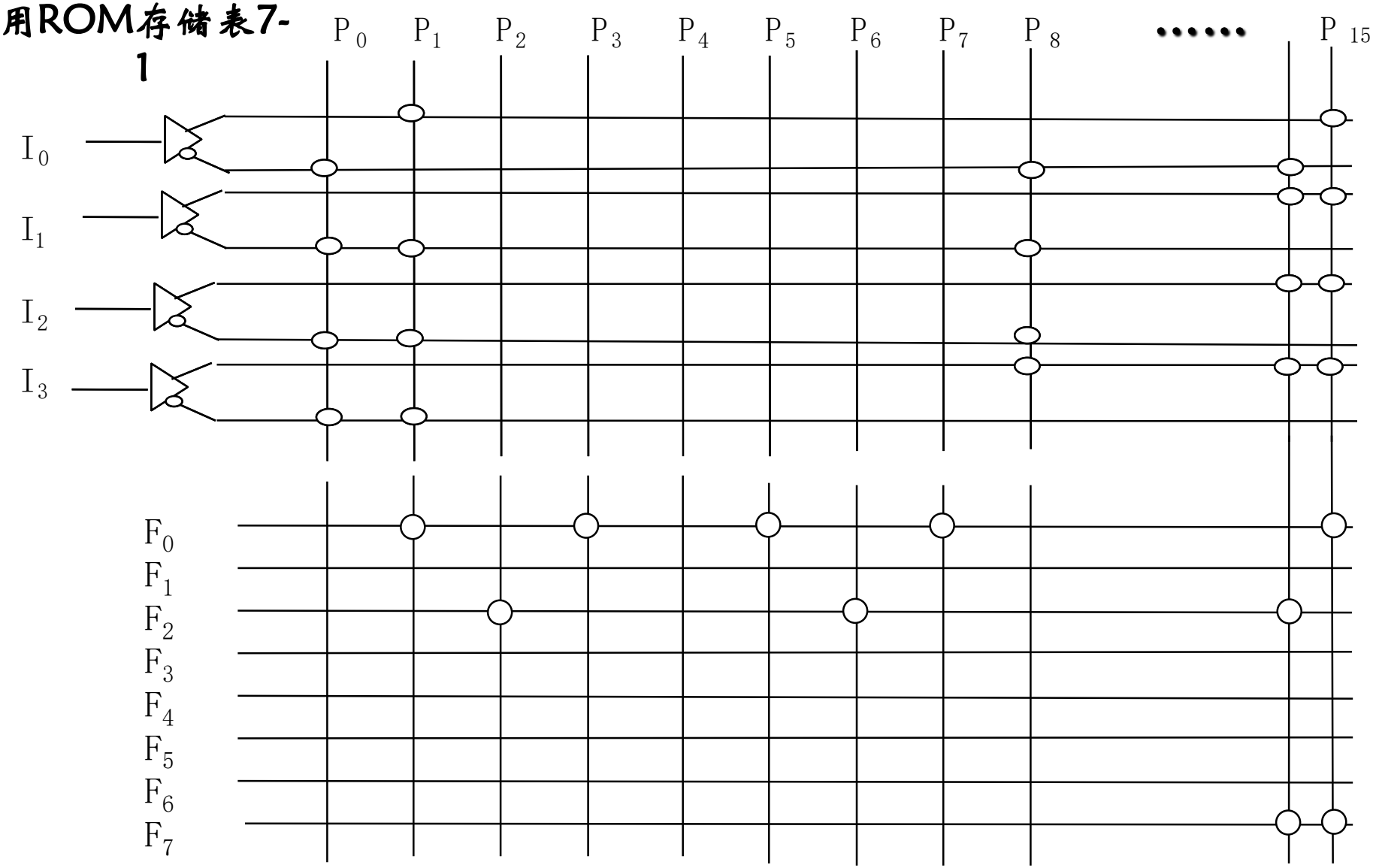
与阵列 ?x?

或阵列 ?x?

输入				输出							
I3	I2	I1	I0	F7	F6	F5	F4	F3	F2	F1	F0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	1	0	0	1
0	1	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	1	1	1	0	0	1
0	1	1	0	0	0	0	0	0	1	0	0
0	1	1	1	0	0	1	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	0	1	0	0	0	1
1	0	1	0	0	1	0	0	0	1	0	0
1	0	1	1	0	1	0	1	1	0	0	1
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	0	0	1	0	0
1	1	1	1	1	1	1	0	0	0	0	1

P267

用ROM存储表7-1



ROM容量：与阵列8x16，
或阵列8x16

（注:只画了F_0，F_2，F_7很多省略）

将表达式逻辑压缩

$$F_0 = I_0$$

$$F_1 = 0$$

$$F_2 = I_1\overline{I_0}$$

$$F_3 = I_2\overline{I_1}I_0 + \overline{I_2}I_1I_0$$

$$F_4 = I_2\overline{I_1}\,\overline{I_0} + \overline{I_3}I_2I_0 + I_3\overline{I_2}I_0$$

$$F_5 = \overline{I_3}I_2I_0 + I_3I_2I_1$$

$$F_6 = I_3\overline{I_2} + I_3I_2I_1$$

$$F_7 = I_3I_2I_1$$

输入				输出							
I3	I2	I1	I0	F7	F6	F5	F4	F3	F2	F1	F0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	1	0	0	1
0	1	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	1	1	1	0	0	1
0	1	1	0	0	0	0	0	0	1	0	0
0	1	1	1	0	0	1	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	0	1	0	0	0	1
1	0	1	0	0	1	0	0	0	1	0	0
1	0	1	1	0	1	0	1	1	0	0	1
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	0	0	1	0	0
1	1	1	1	1	1	1	0	0	0	0	1

§2 .PLA

- 表达式的或项转化为乘积项(product)

$$F_0 = I_0$$

$$F_1 = 0$$

$$F_2 = I_1\overline{I_0}$$

$$F_3 = I_2\overline{I_1}I_0 + \overline{I_2}I_1I_0$$

$$F_4 = I_2\overline{I_1}\,\overline{I_0} + \overline{I_3}I_2I_0 + I_3\overline{I_2}I_0$$

$$F_5 = \overline{I_3}I_2I_0 + I_3I_2I_1$$

$$F_6 = I_3\overline{I_2} + I_3I_2I_1$$

$$F_7 = I_3I_2I_1$$

$$P_0 = I_0$$

$$P_1 = I_1\overline{I_0}$$

$$P_2 = I_2\overline{I_1}I_0$$

$$P_3 = \overline{I_2}I_1I_0$$

$$P_4 = I_2\overline{I_1}\,\overline{I_0}$$

$$P_5 = \overline{I_3}I_2I_0$$

$$P_6 = I_3\overline{I_2}I_0$$

$$P_7 = I_3I_2I_1$$

$$P_8 = I_3\overline{I_2}$$

§2 .PLA

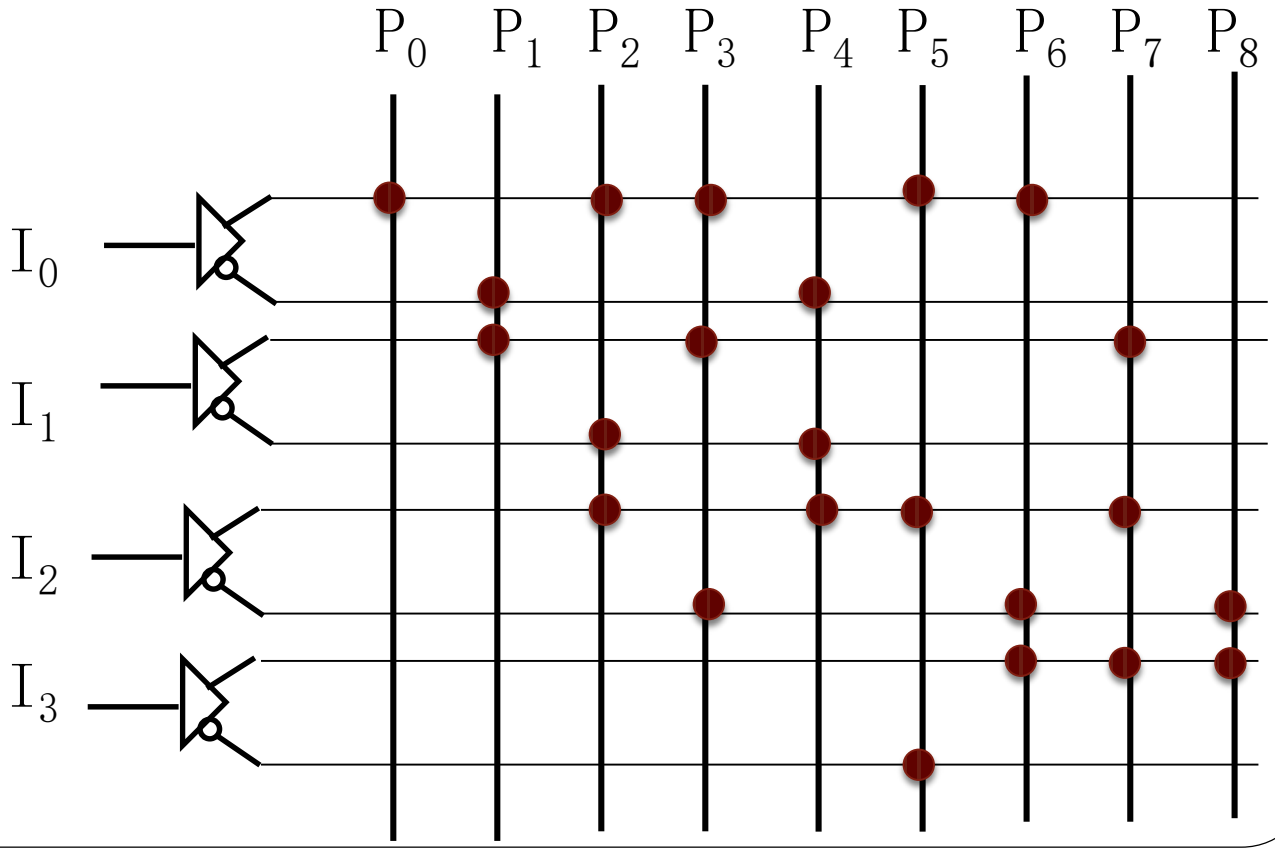
- 表达式的或项转化为乘积项(product)

$$F_0 = P_0$$
$$F_1 = 0$$
$$F_2 = P_1$$
$$F_3 = P_2 + P_3$$
$$F_4 = P_4 + P_5 + P_6$$
$$F_5 = P_5 + P_7$$
$$F_6 = P_7 + P_8$$
$$F_7 = P_7$$

§2 .PLA

- 根据含有乘积项的表达式作图

$$P_0 = I_0$$
$$P_1 = I_1\overline{I_0}$$
$$P_2 = I_2\overline{I_1}I_0$$
$$P_3 = \overline{I_2}I_1I_0$$
$$P_4 = I_2\overline{I_1}\,\overline{I_0}$$
$$P_5 = \overline{I_3}I_2I_0$$
$$P_6 = I_3\overline{I_2}I_0$$
$$P_7 = I_3I_2I_1$$
$$P_8 = I_3\overline{I_2}$$



§2 . PLA

$F_0 = P_0$

$F_1 = 0$

$F_2 = P_1$

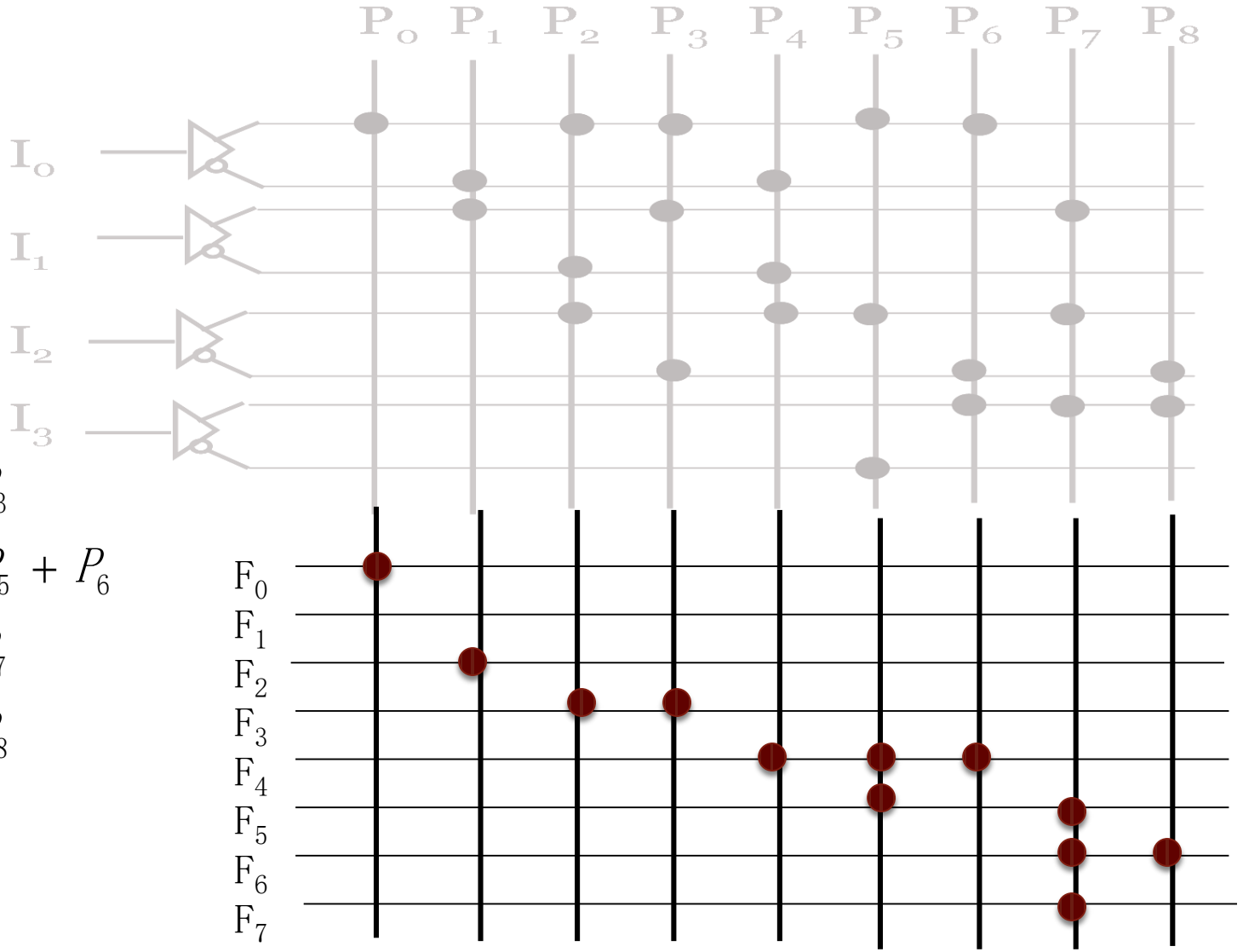
$F_3 = P_2 + P_3$

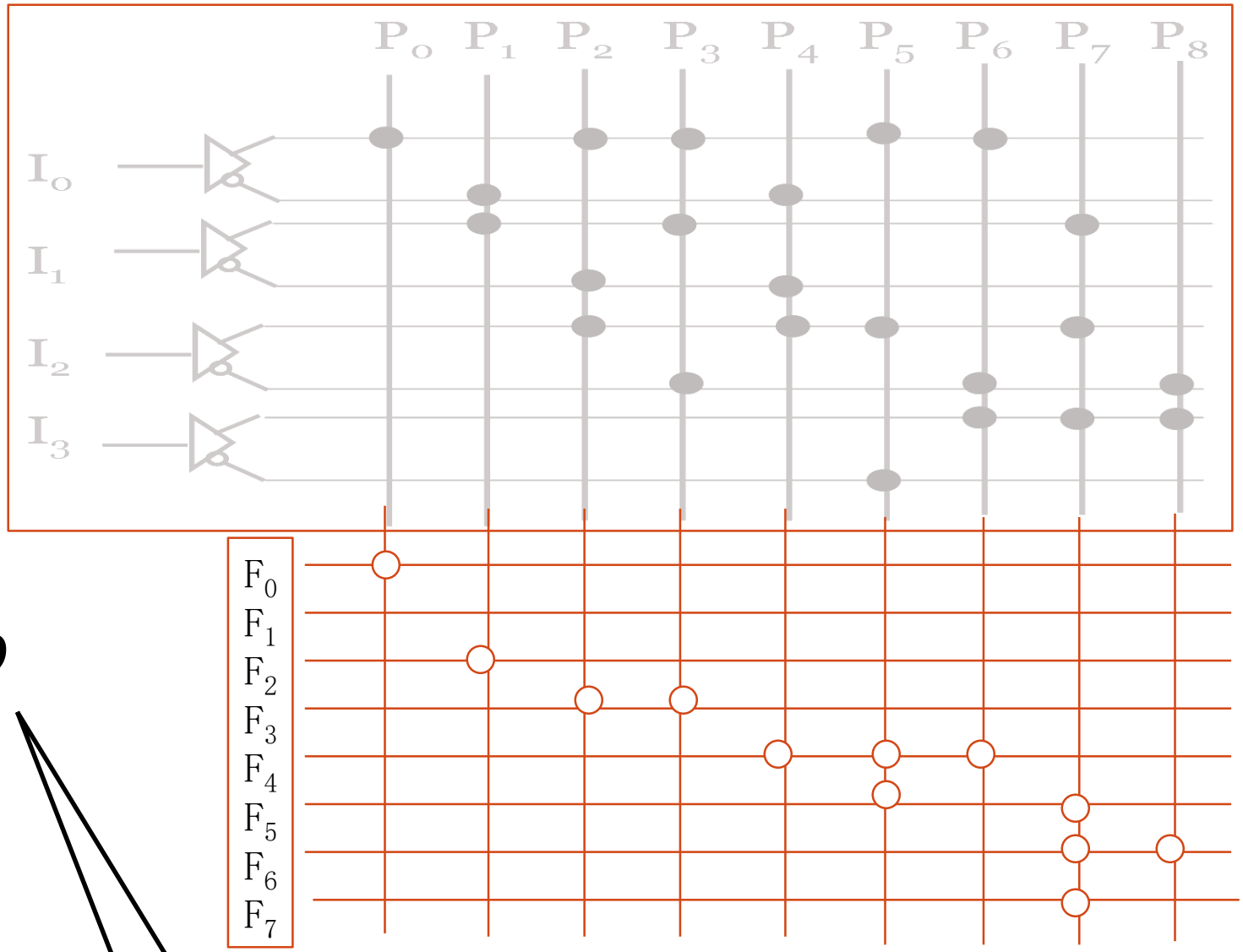
$F_4 = P_4 + P_5 + P_6$

$F_5 = P_5 + P_7$

$F_6 = P_7 + P_8$

$F_7 = P_7$





2N(输入数)

PLA容量

与阵列：8x9

或阵列:8x9

M(输出数)

P项数

总点数:PLA:144

ROM:16x8+8x16=256

§2 .PLA小结

- PLA的特点
  - 与阵列可编：形成P项（不是最小项）
  - 或阵列可编
- PLA的读出方式
  - 一组地址可选中多个P项
- PLA与ROM的区别
  - ROM的信息表原封不动，全译码
  - PLA作了逻辑压缩，信息表改动很大,但逻辑上等价
- PLA的容量表示：

  (输入数) X (P项数) X (输出数)

PROM,PLA,PAL对比

- PROM
  - 与阵列固定，或阵列可编程，实现组合逻辑
  - 实现时序逻辑电路需要外加触发器
- PLA
  - 与或阵列均可编程，实现组合逻辑
  - 实现时序逻辑电路需要外加触发器
- PAL
  - 与阵列可编程，或阵列固定
  - 不同的芯片可实现不同的逻辑，有些PAL只能实现组合逻辑电路，有些只能实现时序逻辑电路。
  - 一次性编程

PROM,PLA,PAL对比

- PROM、PLA、PAL共同存在的问题：
  - 不存在只用一种芯片，即可以实现组合逻辑电路，又可以实现时序逻辑电路

- GAL是为解决这一问题而产生的芯片
  - GAL:通用阵列逻辑（General Array Logic）

可编程器件工艺演化过程

- PROM → PLA
  - 或阵列可编程
  - 与、或阵列都可编程，灵活，节省码点
- PLA → PAL
  - 工艺：简化工艺,降低成本(熔丝工艺,一次编程)
  - 结构：输入/输出公用
  - PAL是专用词，MMI公司的产品
- PAL → GAL
  - 工艺:电可擦除,多次编程.(Lattice公司1985年专利)
  - 结构：输出宏单元，更通用，
  - 或阵列不可编程

作业：4.12, 4.19, 4.20, 4.21, 7.5

4.12 设计一种称之为 4 位桶状移位器的组合逻辑电路，它可将 4 位输入数据 $D_0D_1D_2D_3$ 直接从输出端 $Y_0Y_1Y_2Y_3$ 输出（即 $Y_0Y_1Y_2Y_3=D_0D_1D_2D_3$），也可将输入数据左移 1 位输出（即 $Y_0Y_1Y_2Y_3=D_1D_2D_3D_0$）、左移 2 位及左移 3 位输出，要求：

（1）列出功能表；

（2）写出输出逻辑表达式；

（3）画出逻辑图（用门电路实现）；

（4）为测试 S_1 至 Y_1 的传输延迟，应如何设置各输入端，并给出此时各输出端的状态或波形，画出此时输入输出时序图，给出传输延迟的定义表示。

4.19 用 BCD 译码器，8 选 1 数据选择器及“与非”门组成能控制的 3 位并行等值比较器，列其功能表(要求比较器禁止时输出为“1”)。用上述电路的开关参数，写出此比较器的开关参数的表示式。

4.20 用“与非”门及 8 选 1 数据选择器，4 选 1 数据选择器分别实现真值表所给出的函数，画出逻辑图。

表 4-9

A	B	C	F
0	0	0	0
1	0	0	0
0	1	0	1
1	1	0	1
0	0	1	0
1	0	1	1
0	1	1	1
1	1	1	0

4.21 用 8 选 1 数据选择器实现下列函数：

(1) $F=\overline{X\,\overline{YZ}+(XZ+\overline{XZ})W+\overline{X}\,\overline{YW}}$

(2) $F=\sum m^4(0,1,2,3,8,9,10,11)$

7.5 有一块 4×10×7 的 PLA 器件，由它来实现将 8421 码变换为 7 段显示译码的变换器。