

数字逻辑设计

高翠芸

School of Computer Science

gaocuiyun@hit.edu.cn

Unit 7 组合逻辑元件

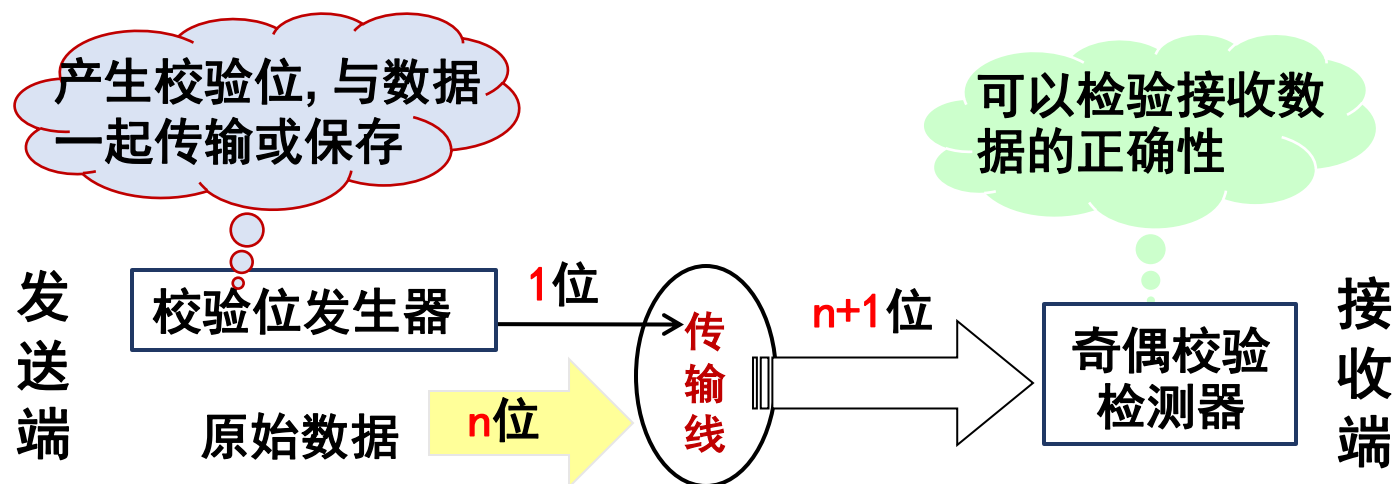
- 多路复用器(multiplexers)
- 三态器件(Three-state Buffer)
- 译码器(Decoders)
- 编码器(Encoders)
- 奇偶校验器
- 比较器
- 只读存储器(ROM)
- 利用MSI设计组合逻辑电路

奇偶校验器

- 用来检查数据传输和存取过程中是否产生错误的组合逻辑电路。
(就是检测数据中包含“1”的个数是奇数还是偶数)

- 广泛用于计算机的内存储器以及磁盘等外部设备中

{ 奇偶校验**发生器**: 可产生奇偶校验位, 与数据一起传输或保存
奇偶校验**检测器**: 可以检验所接受数据的正确性



被校验的原始数据和1位校验位组成 $n+1$ 位校验码。



校验码: $n+1$ 位

偶校验位逻辑值的表达式:

$$P_E = A_3 \oplus A_2 \oplus A_1 \oplus A_0$$

奇校验位逻辑值的表达式:

$$P_O = A_3 \oplus A_2 \oplus A_1 \oplus A_0$$

偶校验位逻辑值电路是在奇校验位逻辑值电路输出端加非门实现

4位二进制数校验码真值表

| $A_3A_2A_1A_0$ | P_E | P_O |
|----------------|-------|-------|
| 0 0 0 0 | 0 | 1 |
| 0 0 0 1 | 1 | 0 |
| 0 0 1 0 | 1 | 0 |
| 0 0 1 1 | 0 | 1 |
| 0 1 0 0 | 1 | 0 |
| 0 1 0 1 | 0 | 1 |
| 0 1 1 0 | 0 | 1 |
| 0 1 1 1 | 1 | 0 |
| 1 0 0 0 | 1 | 0 |
| 1 0 0 1 | 0 | 1 |
| 1 0 1 0 | 0 | 1 |
| 1 0 1 1 | 1 | 0 |
| 1 1 0 0 | 0 | 1 |
| 1 1 0 1 | 1 | 0 |
| 1 1 1 0 | 1 | 0 |
| 1 1 1 1 | 0 | 1 |

奇偶校验器一般由异或门构成

异或门真值表

| A | B | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

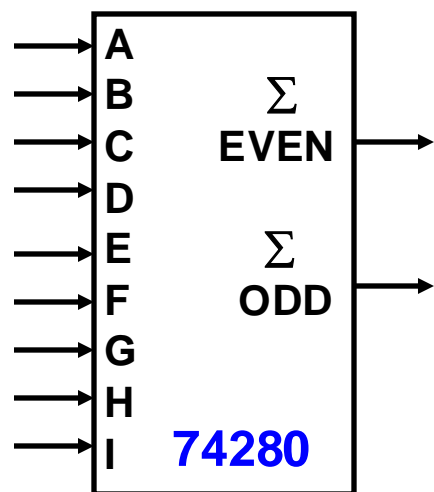
异或门特性

- 两个输入中有奇数个“1”，输出为1；有偶数个“1”，输出为0。
- 扩展：n个1位二进制数中有奇数个“1”，输出为1；有偶数个“1”，输出为0。

奇偶校验器

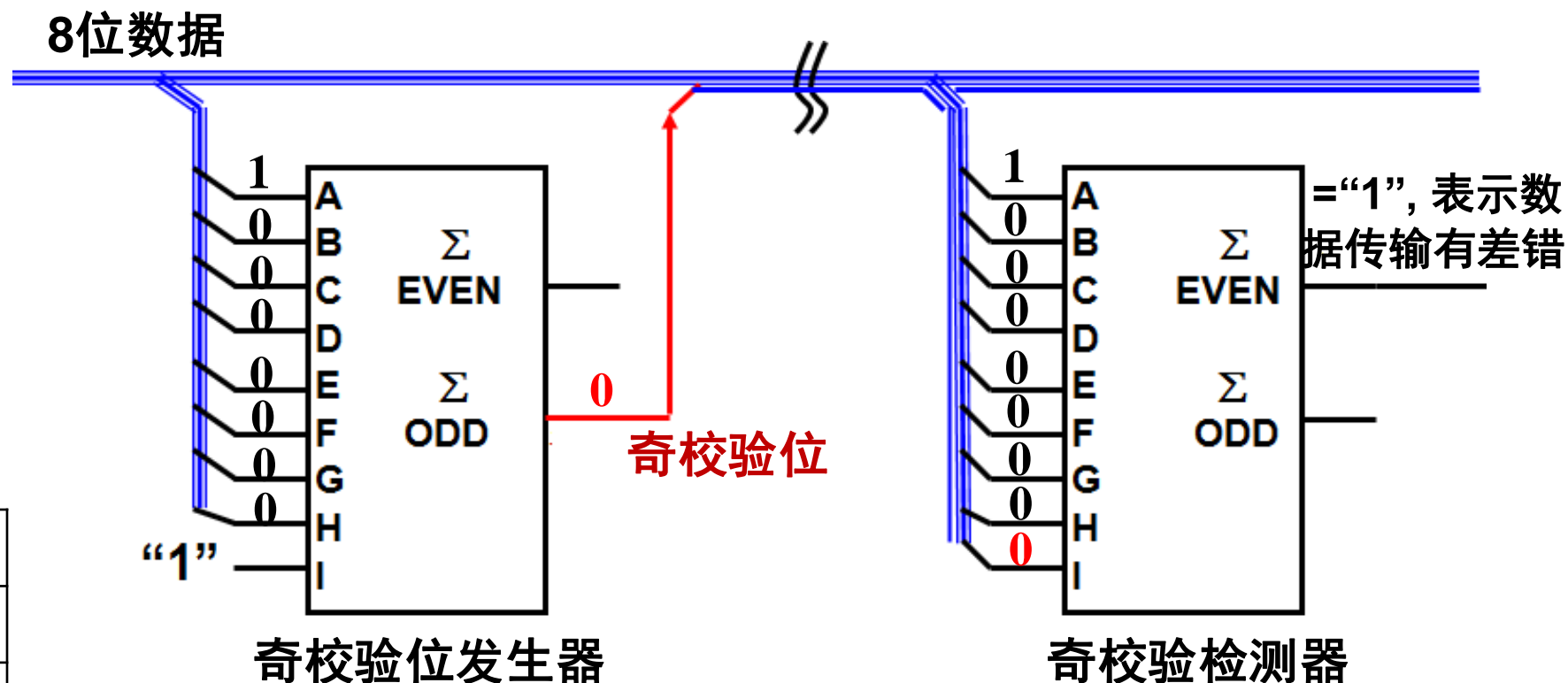
奇偶校验器/产生器：
74xx180、74xx280

例) 用9位奇偶校验器74LS280设计一个8位二
进制码的奇校验位发生器和检测器。



74XX280功能表

| A~I | EVEN | ODD |
|--------|------|-----|
| 偶数个“1” | 1 | 0 |
| 奇数个“1” | 0 | 1 |



奇偶校验器

奇偶校验实际应用意义

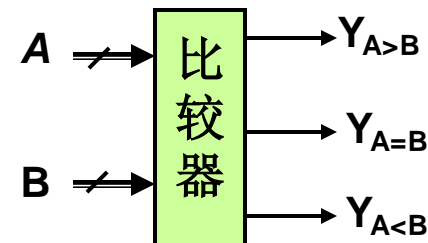
- ① 能够检测传送出错，但不能确定错误位置，不能纠错；
- ② 数据在存储或传送过程中，发生一位错误的可能性占96%以上；
- ③ 电路简单，容易实现，且有实际应用意义。

Unit 7 组合逻辑元件

- 多路复用器(multiplexers)
- 三态器件(Three-state Buffer)
- 译码器(Decoders)
- 编码器(Encoders)
- 奇偶校验器
- 比较器
- 只读存储器(ROM)
- 利用MSI设计组合逻辑电路

数值比较器

- 计算机中对数据的基本处理方法
 - 加、减、乘、除
 - 比较运算
- 数值比较器：一种关系运算电路
 - 能对2个 n 位二进制数 A 和 B 进行比较的多输入、多输出的组合逻辑电路
 - 比较结果： $Y_{A>B}$ 、 $Y_{A<B}$ 、 $Y_{A=B}$



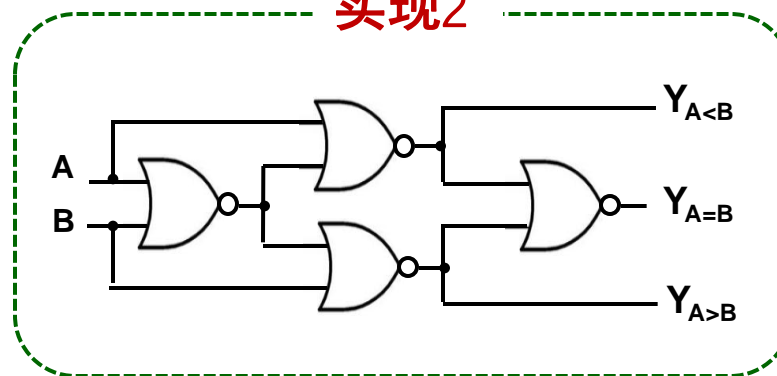
一位数值比较器

真值表

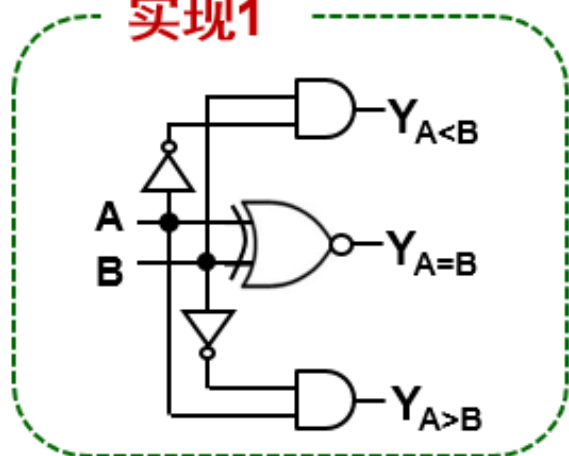
| A | B | $Y_{A=B}$ | $Y_{A>B}$ | $Y_{A<B}$ |
|---|---|-----------|-----------|-----------|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |

$$\begin{cases} Y_{A=B} = A \odot B \\ Y_{A>B} = A\bar{B} \\ Y_{A<B} = \bar{A}B \end{cases}$$

实现2



实现1



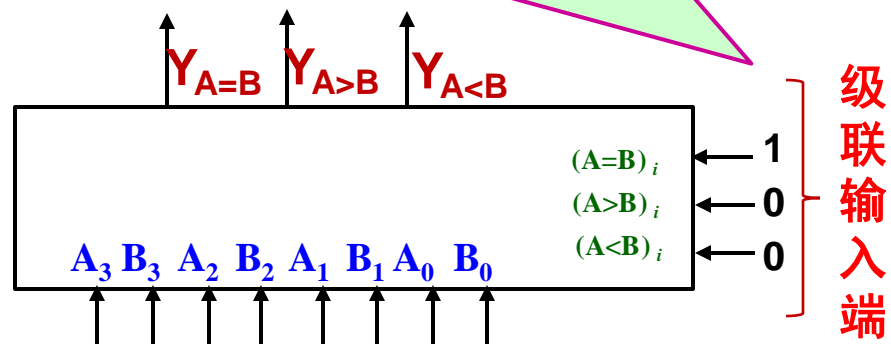
$$\begin{cases} Y_{A=B} = \bar{A}\bar{B} + AB = (A + \bar{B})(\bar{A} + B) = \overline{(A + \bar{A} + \bar{B})(B + \bar{A} + B)} \\ \quad = \overline{(A + \bar{A} + \bar{B}) + (B + \bar{A} + B)} \\ Y_{A>B} = A\bar{B} = \overline{\overline{A(\bar{A} + \bar{B})}} = \overline{A + (\bar{A} + \bar{B})} \\ Y_{A<B} = \bar{A}B = \overline{\overline{B(\bar{A} + \bar{B})}} = \overline{B + (\bar{A} + \bar{B})} \end{cases}$$

多位数值比较器

- 自高而低逐位比较，只有在高位相等时，才需要比较低位。

接低位芯片的比较结果，用于芯片扩展。

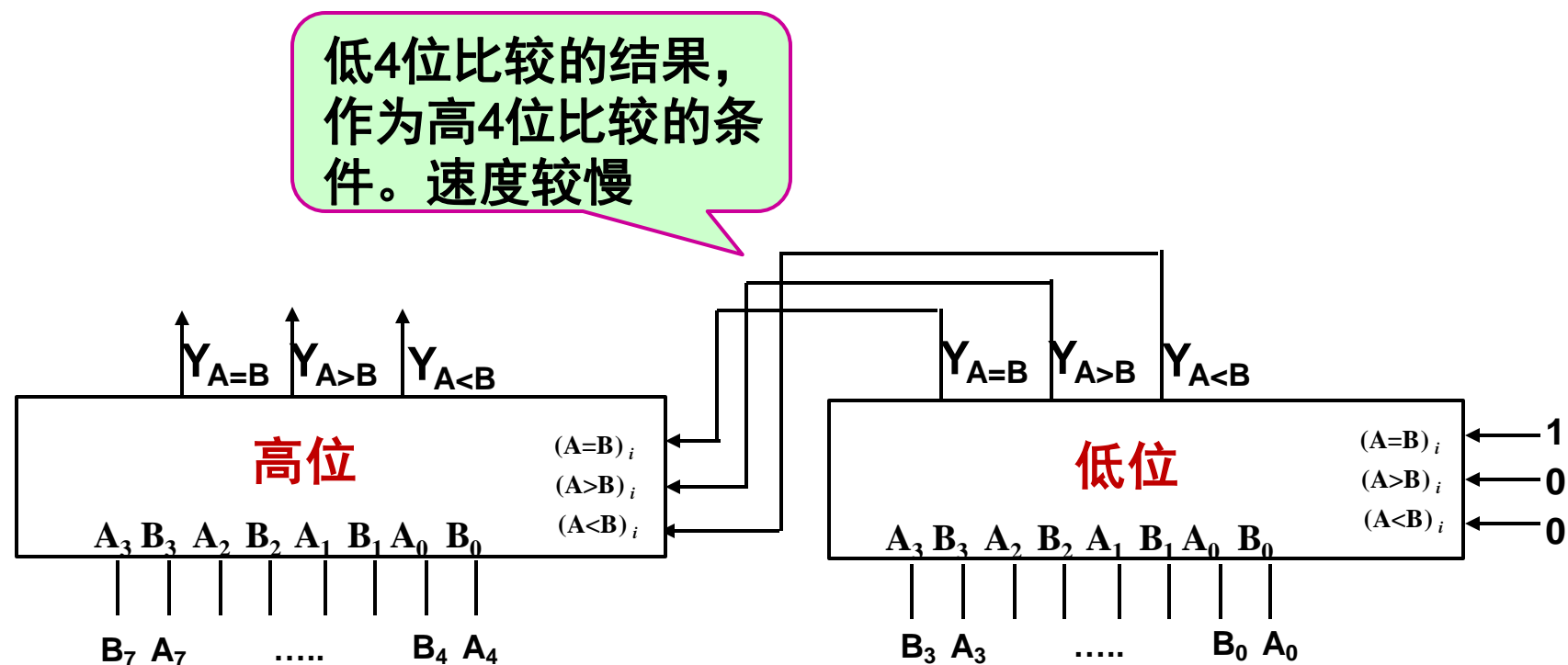
比较2个4位二进制数的大小时，3个输入端 $(A=B)_i$ 、 $(A>B)_i$ 、 $(A<B)_i$ 应接100，当 $A_3A_2A_1A_0 = B_3B_2B_1B_0$ 比较器的输出 $Y_{A=B}Y_{A>B}Y_{A<B} = 100$



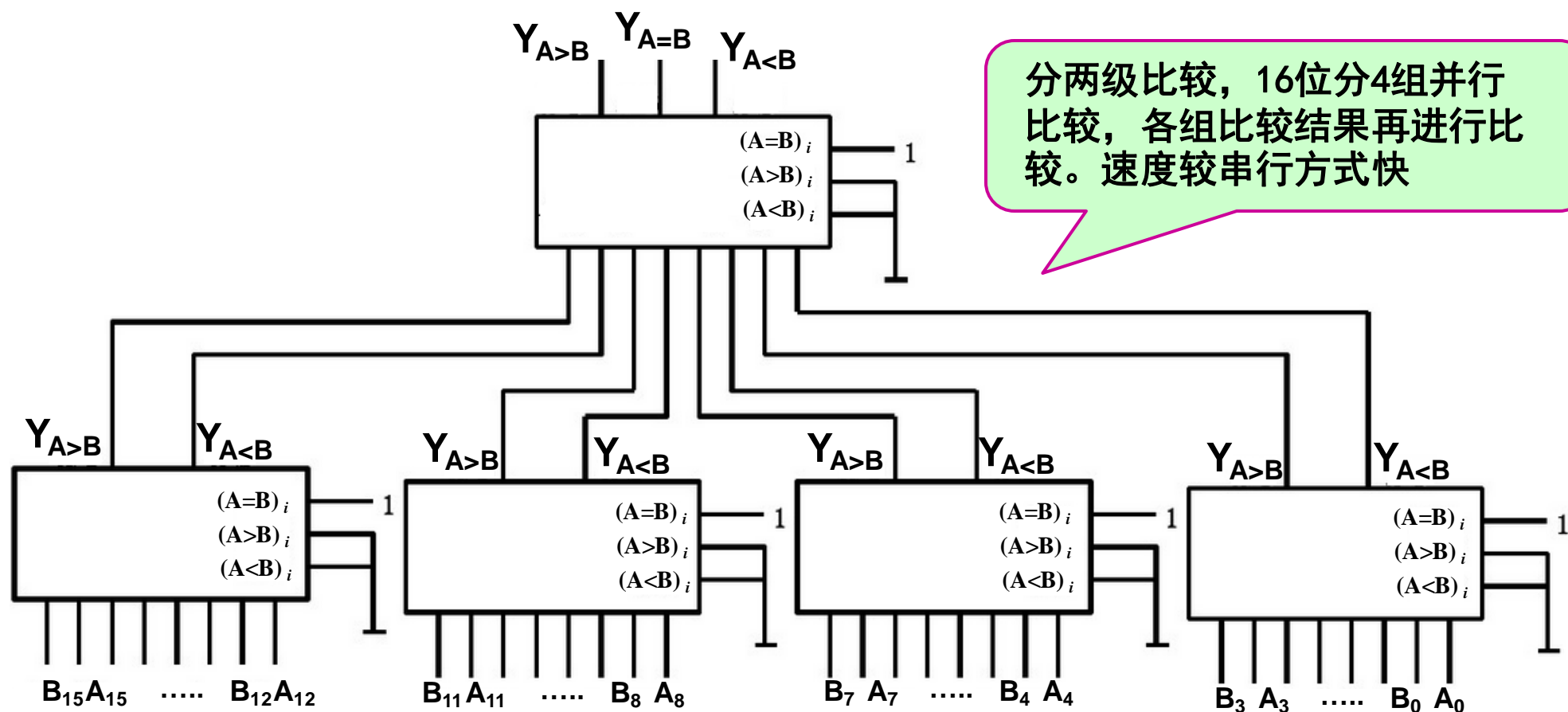
当 $A_3A_2A_1A_0 = B_3B_2B_1B_0$ ，比较器的输出复现3个输入端 $(A=B)_i$ 、 $(A>B)_i$ 、 $(A<B)_i$ 的状态。

| 比较输入 | | | | 级联输入 | | | 输出 | | |
|-------------|-------------|-------------|-------------|-----------|-----------|-----------|-----------|-----------|-----------|
| A_3 B_3 | A_2 B_2 | A_1 B_1 | B_0 A_0 | $(A>B)_i$ | $(A<B)_i$ | $(A=B)_i$ | $Y_{A>B}$ | $Y_{A<B}$ | $Y_{A=B}$ |
| $A_3 > B_3$ | X | X | X | X | X | X | 1 | 0 | 0 |
| $A_3 < B_3$ | X | X | X | X | X | X | 0 | 1 | 0 |
| $A_3 = B_3$ | $A_2 > B_2$ | X | X | X | X | X | 1 | 0 | 0 |
| $A_3 = B_3$ | $A_2 < B_2$ | X | X | X | X | X | 0 | 1 | 0 |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 > B_1$ | X | X | X | X | 1 | 0 | 0 |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 < B_1$ | X | X | X | X | 0 | 1 | 0 |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 > B_0$ | X | X | X | 1 | 0 | 0 |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 < B_0$ | X | X | X | 0 | 1 | 0 |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | 1 | 0 | 0 | 1 | 0 | 0 |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | 0 | 1 | 0 | 0 | 1 | 0 |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | 0 | 0 | 1 | 0 | 0 | 1 |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | 0 | 1 | 1 | 0 | 1 | 1 |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | 1 | 0 | 1 | 1 | 0 | 1 |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | 1 | 1 | 0 | 1 | 1 | 0 |
| $A_3 = B_3$ | $A_2 = B_2$ | $A_1 = B_1$ | $A_0 = B_0$ | 1 | 1 | 1 | 1 | 1 | 1 |

数值比较器的级联——①串行方式



数值比较器的级联——②并行方式



Unit 7 组合逻辑元件

- 多路复用器(multiplexers)
- 三态器件(Three-state Buffer)
- 译码器(Decoders)
- 编码器(Encoders)
- 奇偶校验器
- 比较器
- 利用MSI设计组合逻辑电路
- 只读存储器(ROM)

Unit 7 组合逻辑元件

- 多路复用器(multiplexers)
- 三态器件(Three-state Buffer)
- 译码器(Decoders)
- 编码器(Encoders)
- 奇偶校验器
- 比较器
- 利用MSI设计组合逻辑电路
- 只读存储器(ROM)

Design with MSI blocks

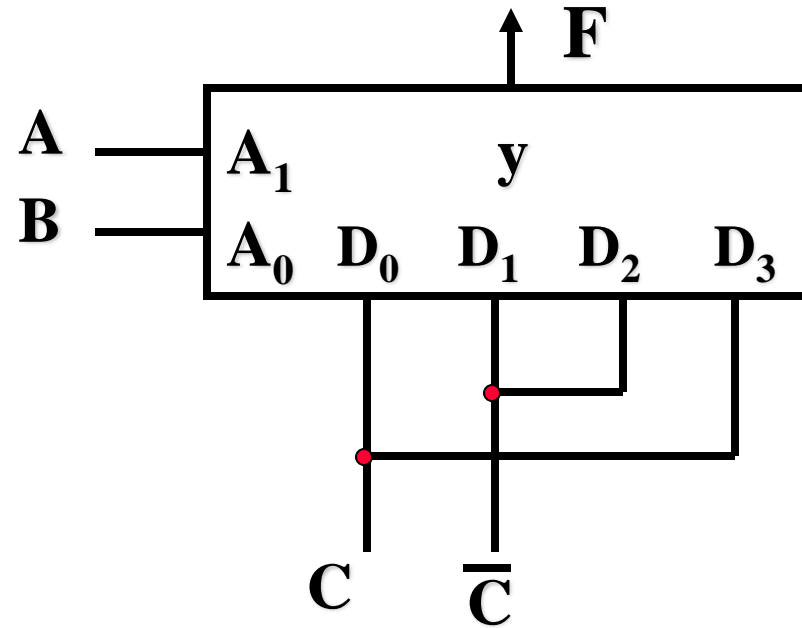
- ① 了解各类典型集成电路芯片的功能、外特性；
- ② 学会查阅器件资料；
- ③ 能灵活运用，完成最佳设计。

MSI blocks {

- Multiplexers
- Decoders

1.用数据选择器来实现组合逻辑电路

Write the expression of F

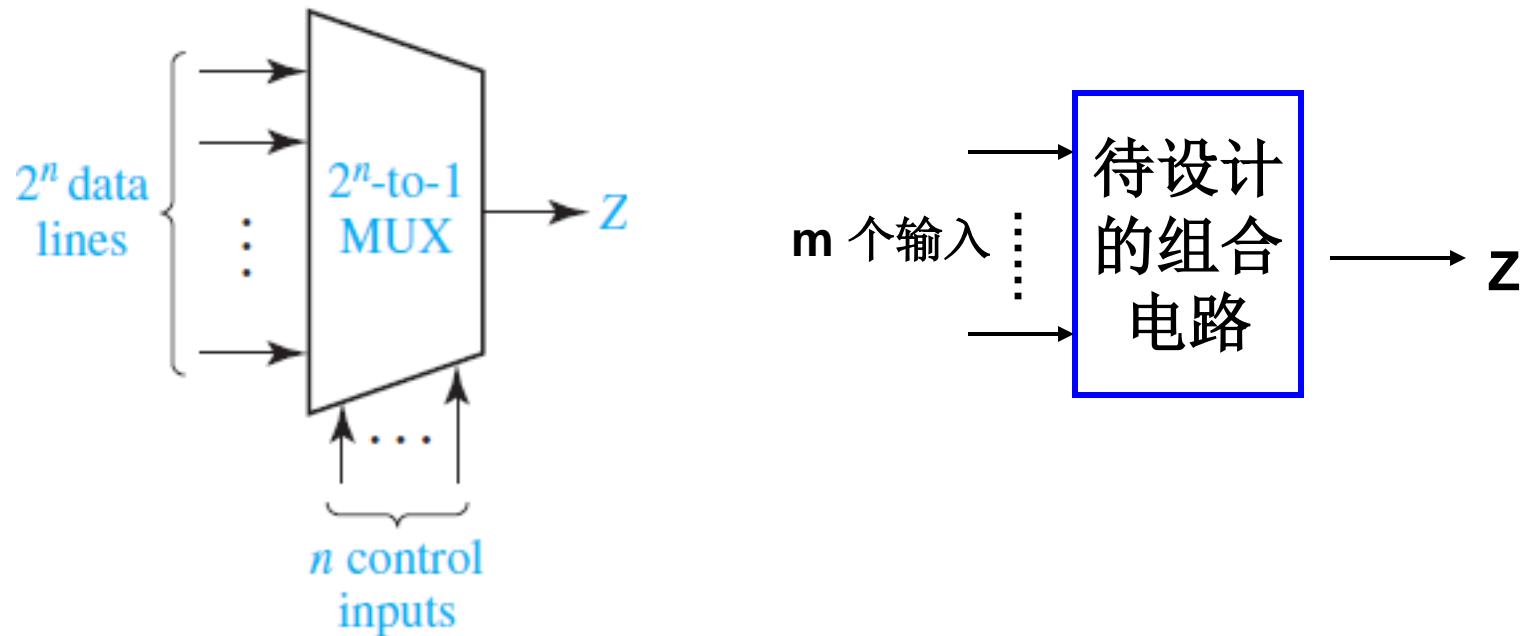


$$F = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

1.用数据选择器来实现组合逻辑电路

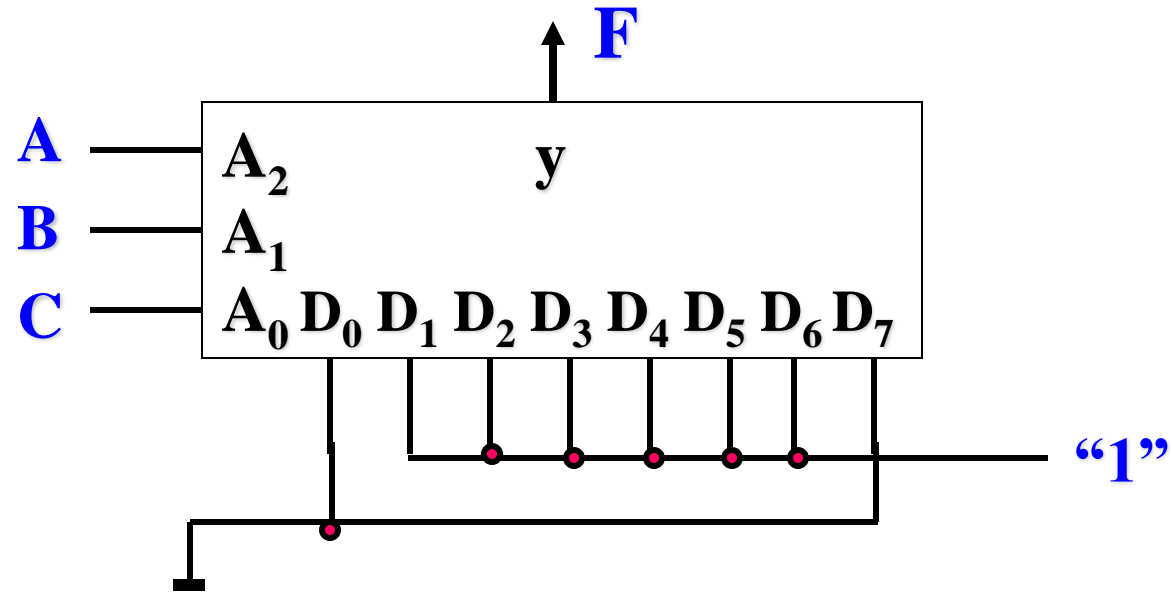
(1) $m = n$

数据选择器的控制端个数
组合电路的输入变量个数



Use 8 to 1 MUX realize $F = A\bar{B} + \bar{A}C + B\bar{C}$

| $A_2A_1A_0$ | y |
|-------------|-------|
| 0 0 0 | D_0 |
| 0 0 1 | D_1 |
| 0 1 0 | D_2 |
| 0 1 1 | D_3 |
| 1 0 0 | D_4 |
| 1 0 1 | D_5 |
| 1 1 0 | D_6 |
| 1 1 1 | D_7 |



| A | BC | | | |
|---|----|----|----|----|
| | 00 | 01 | 11 | 10 |
| 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |

K.Map of F

| A_2 | A_1A_0 | | | |
|-------|----------|-------|-------|-------|
| | 00 | 01 | 11 | 10 |
| 0 | D_0 | D_1 | D_3 | D_2 |
| 1 | D_4 | D_5 | D_7 | D_6 |

K.Map of MUX



1.用数据选择器来实现组合逻辑电路

(2) $m = n+1$

└───┐
└───┘ 数据选择器的控制端个数
└───┘ 组合电路的输入变量个数

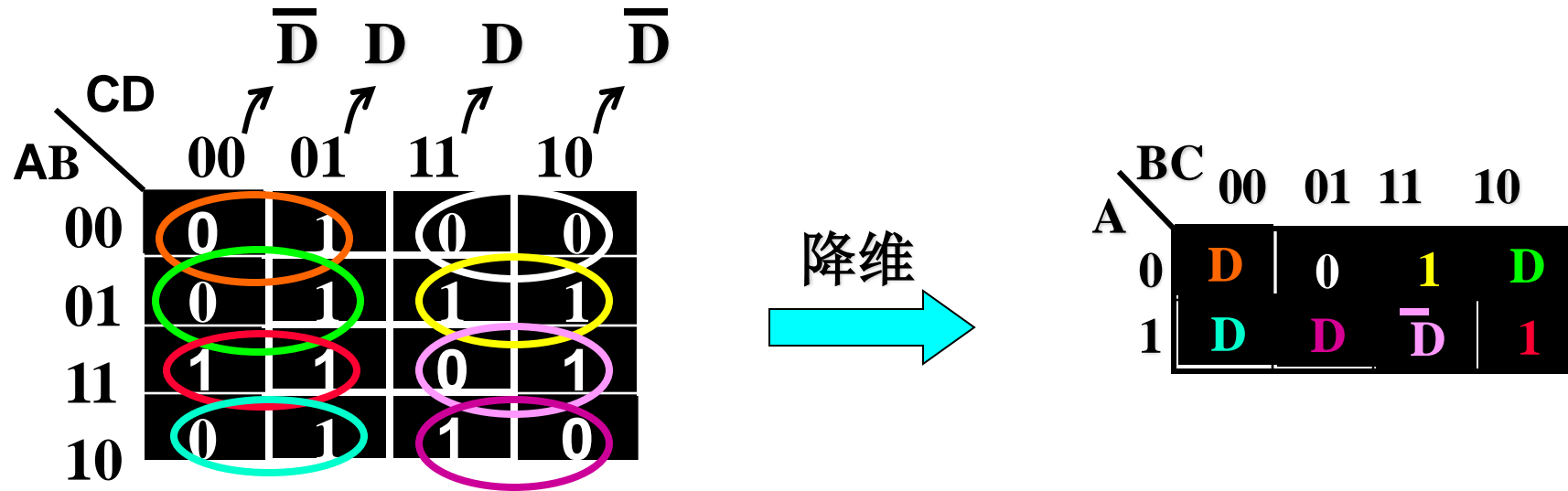
Method: 降维

利用 8 to 1 MUX 设计组合逻辑：

$$F(A,B,C,D)=\sum m(1,5,6,7,9,11,12,13,14)$$

1.用数据选择器来实现组合逻辑电路

$$F(A,B,C,D)=\sum m(1,5,6,7,9,11,12,13,14)$$



$$f(x_1x_2\dots x_i\dots x_n)$$

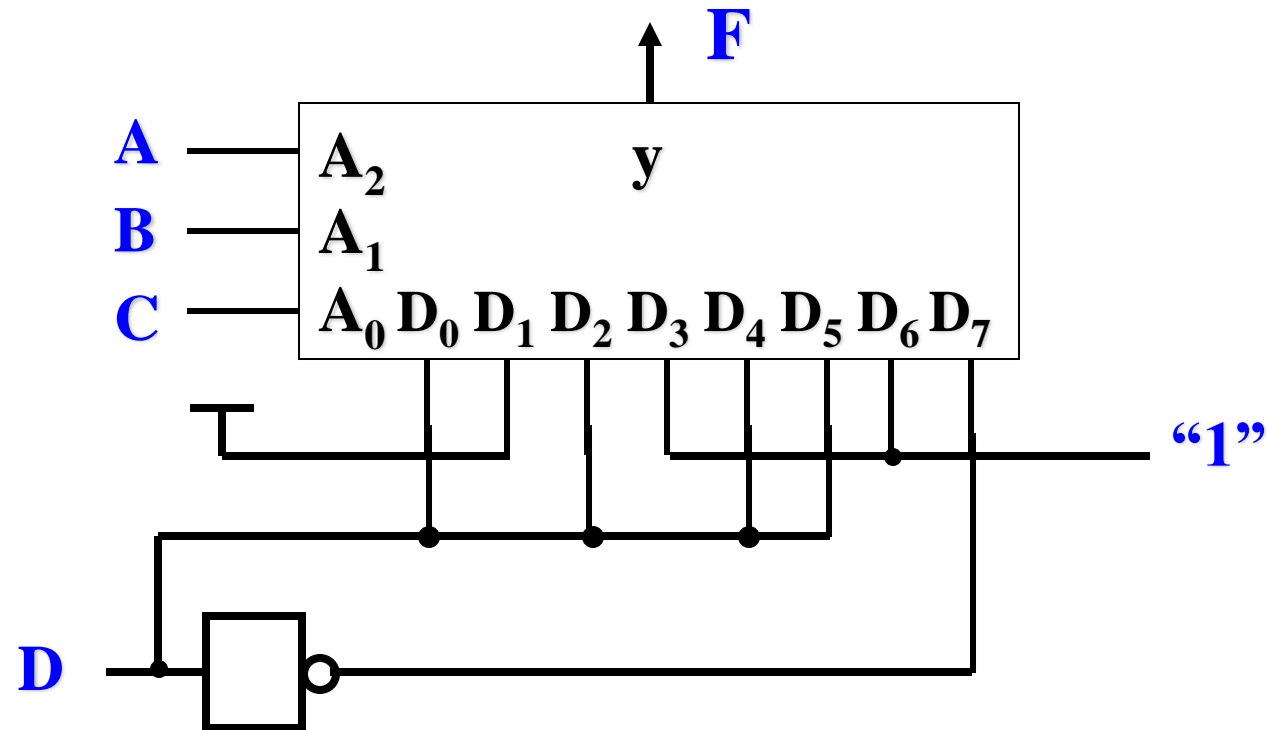
$$= x_i \cdot f(x_1x_2\dots \mathbf{1}\dots x_n) + \overline{x_i} \cdot f(x_1x_2\dots \mathbf{0}\dots x_n)$$

| A_2 | $A_1 A_0$ | | 00 | 01 | 11 | 10 |
|-------|-----------|--|-------|-------|-------|-------|
| | | | D_0 | D_1 | D_3 | D_2 |
| 0 | | | D_0 | D_1 | D_3 | D_2 |
| 1 | | | D_4 | D_5 | D_7 | D_6 |

K.Map of MUX

| A | BC | | | |
|-----|------|-----|-----------|-----|
| | 00 | 01 | 11 | 10 |
| 0 | D | 0 | 1 | D |
| 1 | D | D | \bar{D} | 1 |

K.Map of F



1.用数据选择器来实现组合逻辑电路

(3) $m = n+2$

└───┐
└───┘ 数据选择器的控制端个数
└───┘ 组合电路的输入变量个数

Method: 降维

Use 4-to-1 MUX realize :

$$F(A,B,C,D)=\sum m(0,1,5,6,7,9,10,14,15)$$

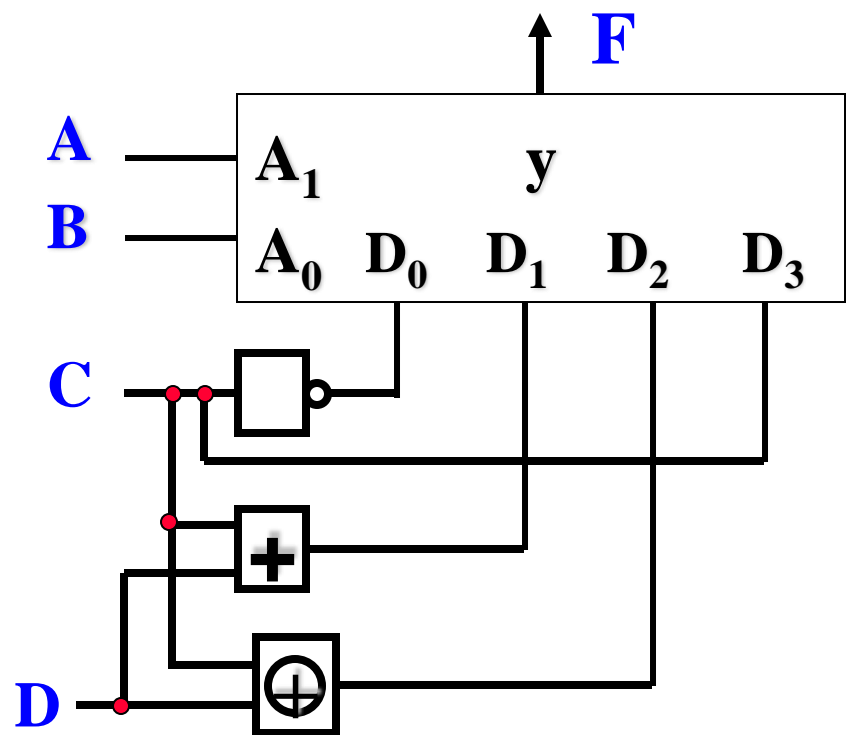
| | | \overline{D} | D | D | \overline{D} |
|----|----|----------------|------------|------------|----------------|
| | | \nearrow | \nearrow | \nearrow | \nearrow |
| CD | | 00 | 01 | 11 | 10 |
| AB | 00 | 1 | 1 | 0 | 0 |
| | 01 | 0 | 1 | 1 | 1 |
| | 11 | 0 | 0 | 1 | 1 |
| | 10 | 0 | 1 | 0 | 1 |

降维

| | | BC | | | |
|---|---|----|----------------|----|----|
| | | 00 | 01 | 11 | 10 |
| A | 0 | 1 | 0 | 1 | D |
| | 1 | D | \overline{D} | 1 | 0 |

降维

| | | B | |
|---|---|----------------|-------|
| | | 0 | 1 |
| A | 0 | \overline{C} | $C+D$ |
| | 1 | $C \oplus D$ | C |

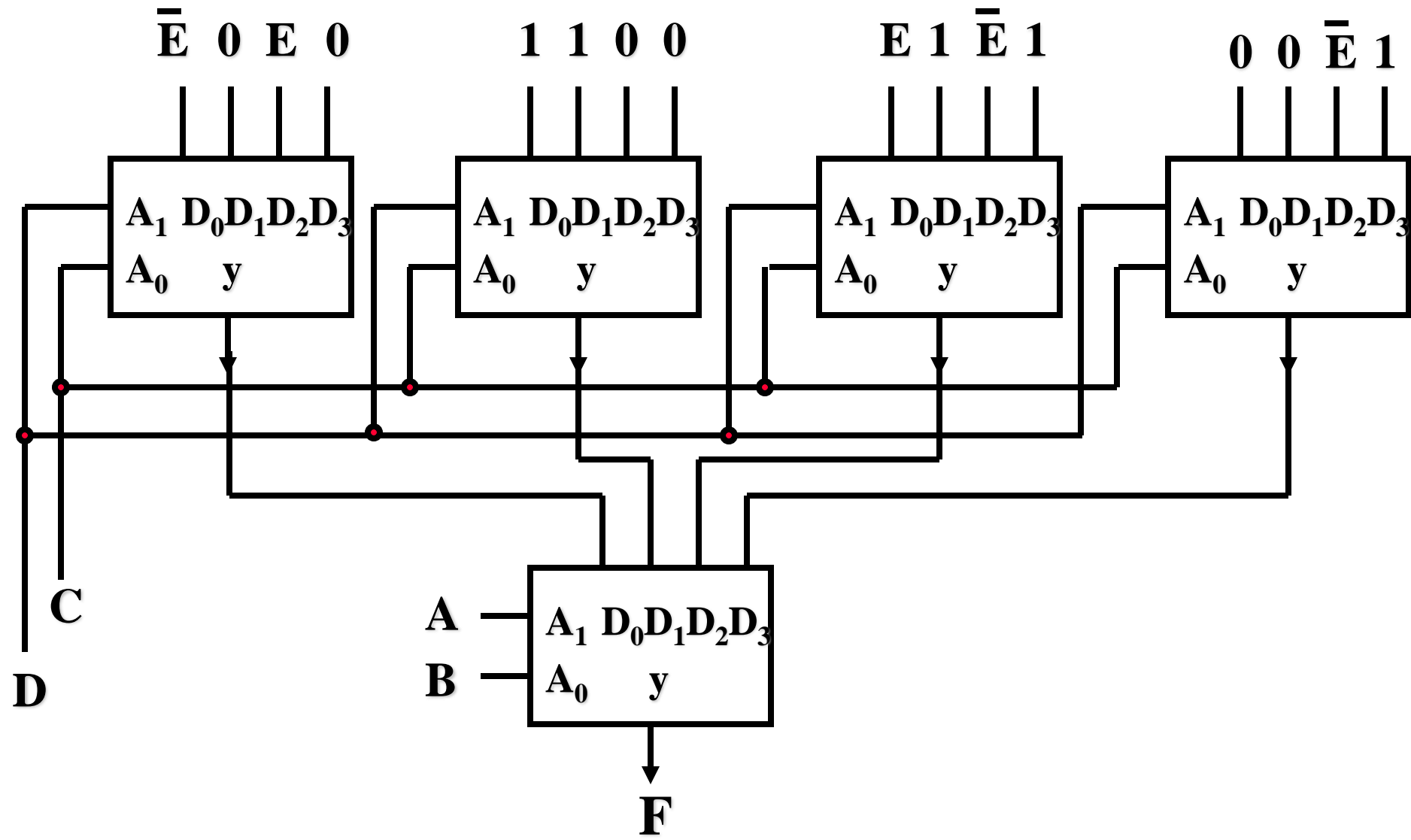


利用4-to-1 MUX 设计组合逻辑

$$F(A,B,C,D,E) = \sum m(0,5,8,9,10,11,17,18,19,20,22,23,28,30,31)$$

| A B | C D | E | F |
|-----|-----|---|-----------|
| 00 | 0 0 | | \bar{E} |
| | 0 1 | | |
| | 1 0 | | |
| | 1 1 | | |
| 01 | 0 0 | | 0 |
| | 0 1 | | |
| | 1 0 | | |
| | 1 1 | | |

| A B | C D | E | F |
|-----|-----|---|---|
| 10 | 0 0 | | E |
| | 0 1 | | |
| | 1 0 | | |
| | 1 1 | | |
| 11 | 0 0 | | 1 |
| | 0 1 | | |
| | 1 0 | | |
| | 1 1 | | |



2.用数据选择器来实现组合逻辑电路

Use 4-to-1 MUX realize

$$F(A,B,C,D)=\sum m(1,2,4,9, 10,11,12,14,15)$$

Method: 降维

从函数的多个输入变量中选出2个作为MUX的选择控制变量。原则上讲，这种选择是任意的，但选择合适时可使设计简化。

① Choose A and B

① Choose A and B

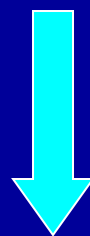
| CD | | \bar{D} | D | D | \bar{D} |
|----|----|-----------|----|----|-----------|
| | | 00 | 01 | 11 | 10 |
| AB | 00 | 0 | 1 | 0 | 1 |
| | 01 | 1 | 0 | 0 | 0 |
| | 11 | 1 | 0 | 1 | 1 |
| | 10 | 0 | 1 | 1 | 1 |

降维

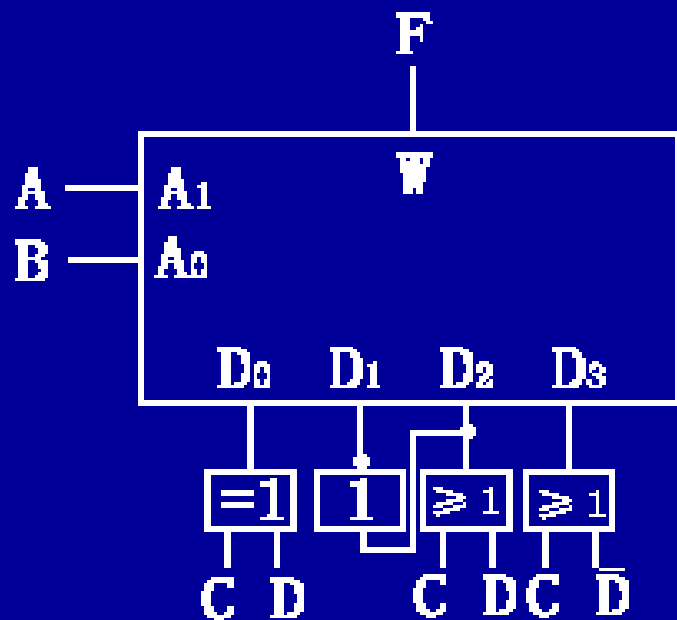


| BC | | 00 | 01 | 11 | 10 |
|----|---|----|-----------|----|-----------|
| | | 0 | 1 | 1 | 0 |
| A | 0 | D | \bar{D} | 0 | \bar{D} |
| | 1 | D | 1 | 1 | \bar{D} |

降维



| B | | 0 | 1 |
|---|---|--------------|------------------|
| | | 0 | 1 |
| A | 0 | $C \oplus D$ | $\bar{C}\bar{D}$ |
| | 1 | $C + D$ | $C + \bar{D}$ |

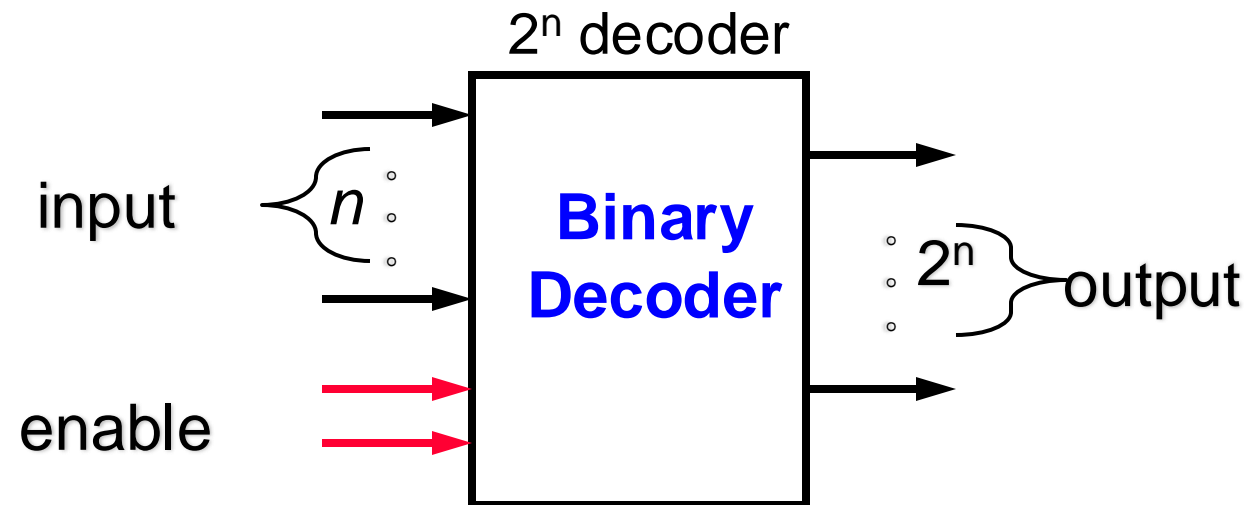


(b)

② Choose B and C

2.用译码器来实现组合逻辑电路

MSI blocks {
▪ Multiplexers
▪ Decoders



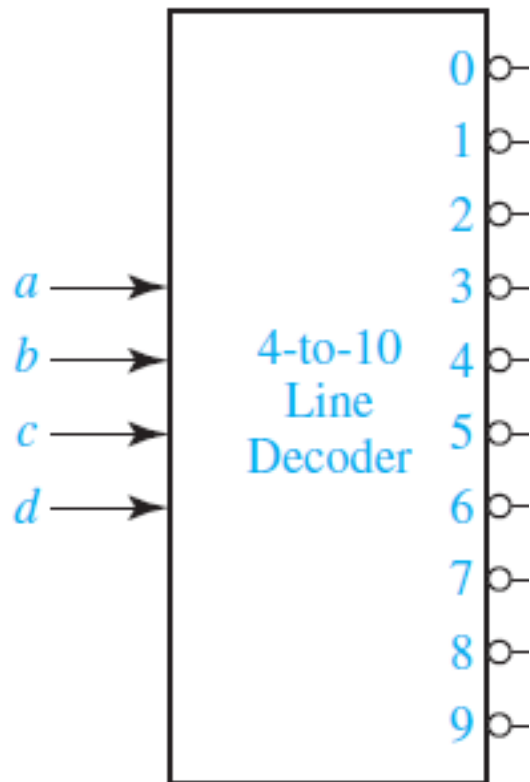
$$y_i = m_i, \quad i = 0 \text{ to } 2^n - 1 \quad (\text{noninverted outputs})$$

$$y_i = m_i' = M_i, \quad i = 0 \text{ to } 2^n - 1 \quad (\text{inverted outputs})$$

2.用译码器来实现组合逻辑电路

$$f_1(a, b, c, d) = m_1 + m_2 + m_4$$

$$f_2(a, b, c, d) = m_4 + m_7 + m_9$$



利用 74LS138 设计 1-bit FA

| a_i | b_i | C_{i-1} | S_i | C_i |
|-------|-------|-----------|-------|-------|
| 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 |

74138功能表

| 使能端 | | | 输入 | | | 译码输出 | | | | | | | |
|-------|----------|----------|----|---|---|-------|-------|-------|-------|-------|-------|-------|-------|
| G_1 | G_{2A} | G_{2B} | C | B | A | Y_0 | Y_1 | Y_2 | Y_3 | Y_4 | Y_5 | Y_6 | Y_7 |
| 0 | X | X | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X | 1 | X | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X | X | 1 | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

$$y_i = \overline{m}_i$$

$$S_i = \sum (1,2,4,7) = \overline{\overline{m}_1} \overline{\overline{m}_2} \overline{\overline{m}_4} \overline{\overline{m}_7}$$

$$c_{i-1} = \sum (3,5,6,7) = \overline{\overline{m}_3} \overline{\overline{m}_5} \overline{\overline{m}_6} \overline{\overline{m}_7}$$

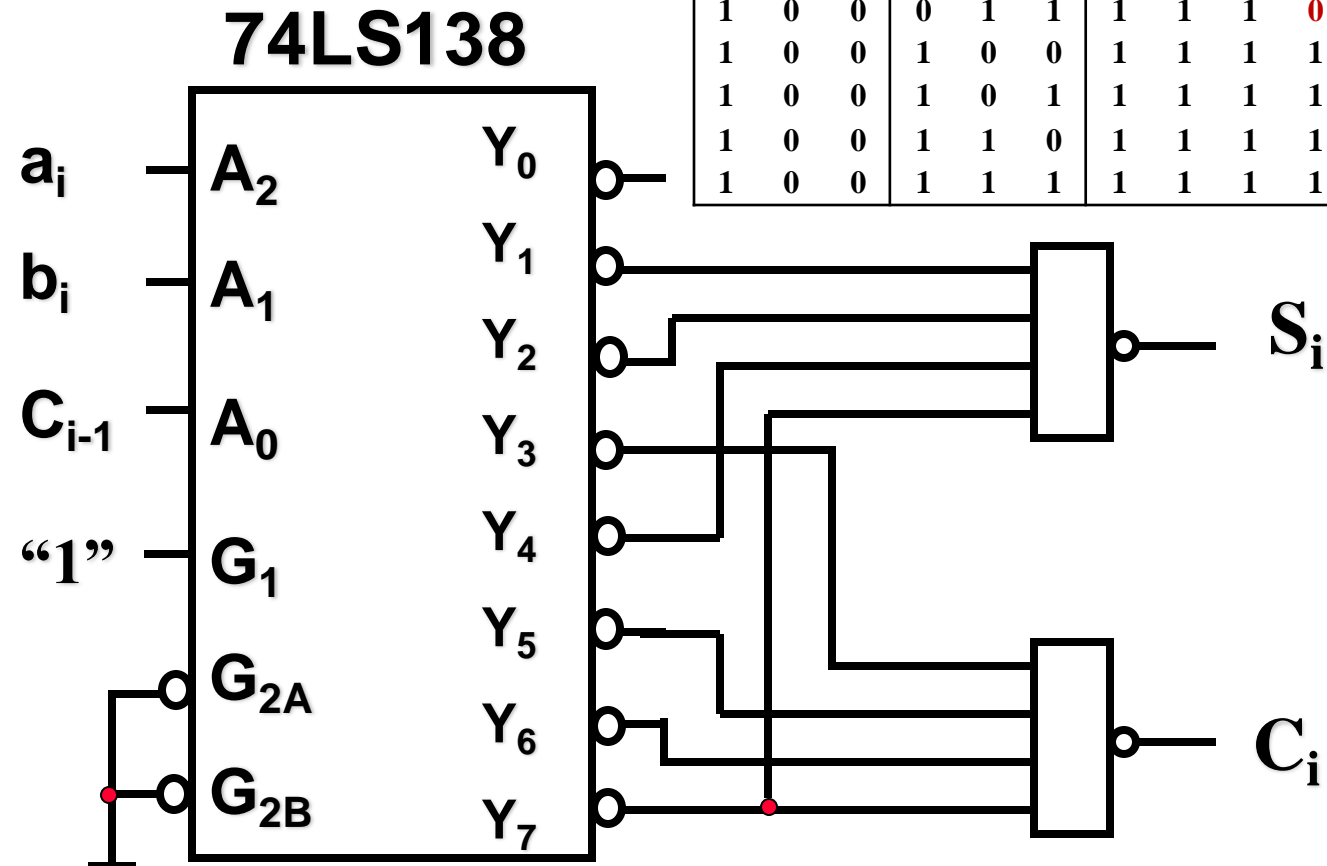
74138功能表

$$y_i = \overline{m}_i$$

$$S_i = \sum (1,2,4,7) = \overline{m}_1 \overline{m}_2 \overline{m}_4 \overline{m}_7$$

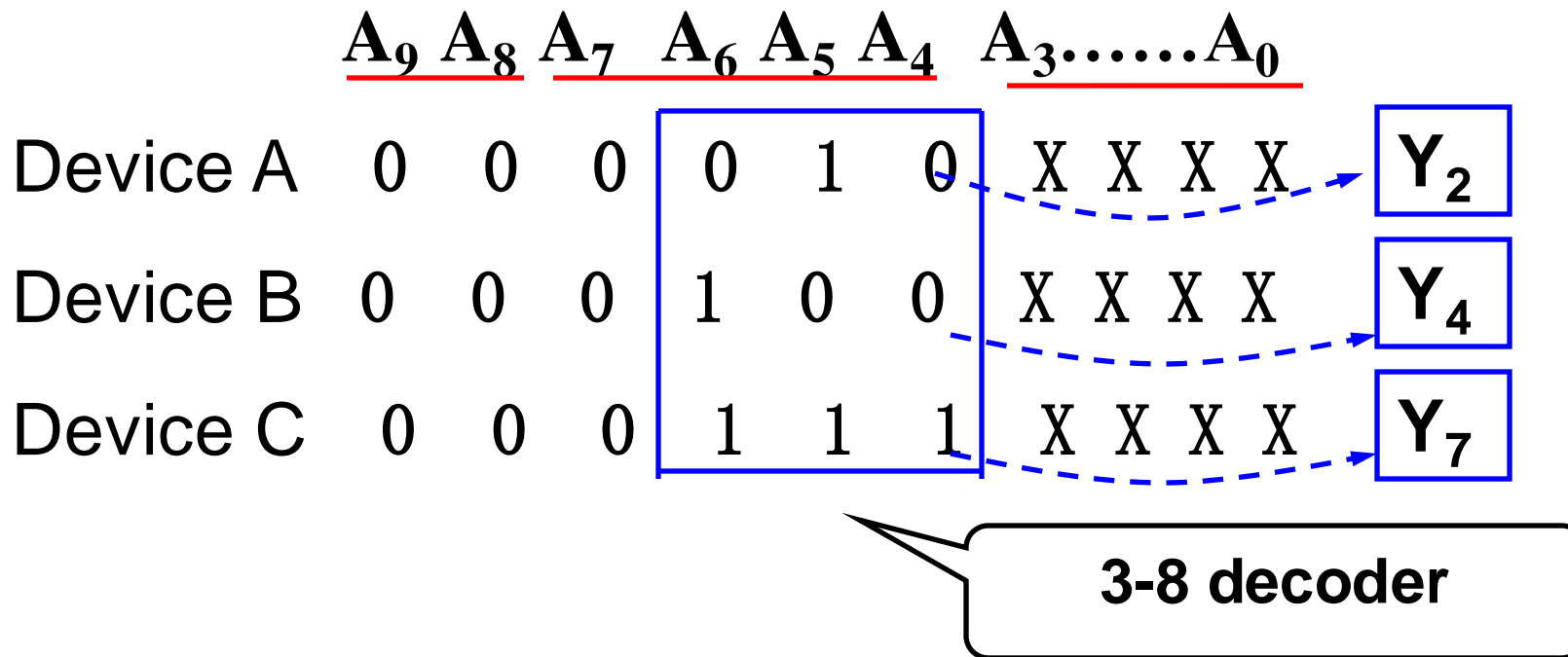
$$c_{i-1} = \sum (3,5,6,7) = \overline{m}_3 \overline{m}_5 \overline{m}_6 \overline{m}_7$$

| 使能端 | | | 输入 | | | 译码输出 | | | | | | | |
|-------|----------|----------|----|---|---|-------|-------|-------|-------|-------|-------|-------|-------|
| G_1 | G_{2A} | G_{2B} | C | B | A | Y_0 | Y_1 | Y_2 | Y_3 | Y_4 | Y_5 | Y_6 | Y_7 |
| 0 | X | X | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X | 1 | X | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X | X | 1 | X | X | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |



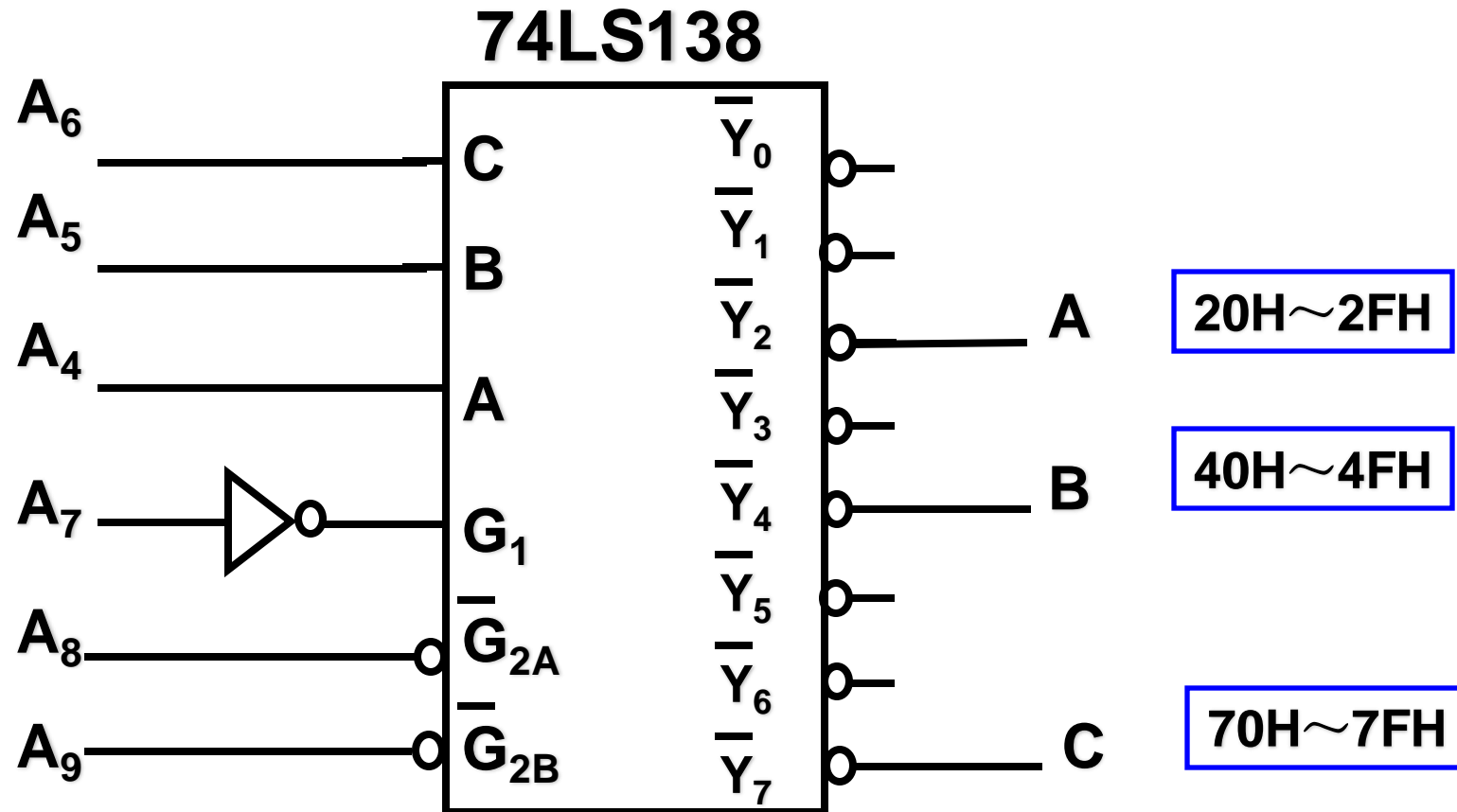
2.用译码器来实现组合逻辑电路

设计一个地址译码器，利用地址线 $A_9 A_8 \dots A_0$ 选择外设 A, B, C。三个外设的地址分别是 20H~2FH, 40H~4FH, 70H~7FH。



2.用译码器来实现组合逻辑电路

Circuit

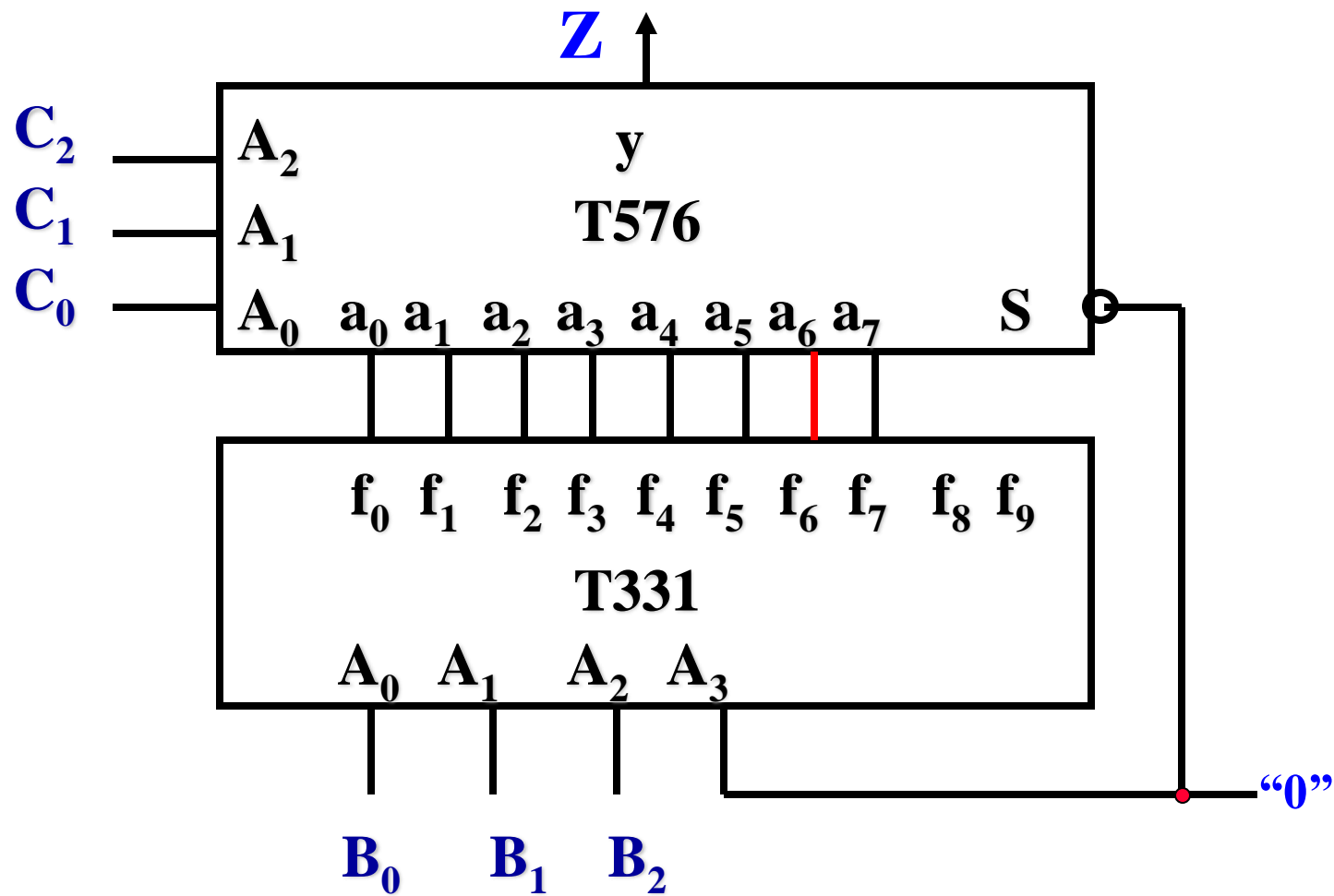


例：利用 **8-to -1 MUX**以及 **4-10**线译码器设计一个能实现**2组3位**数码等值比较的电路。

| $A_3 A_2 A_1 A_0$ | $f_0 f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 f_9$ |
|-------------------|---|
| 0 0 0 0 | 0 1 1 1 1 1 1 1 1 1 |
| 0 0 0 1 | 1 0 1 1 1 1 1 1 1 1 |
| 0 0 1 0 | 1 1 0 1 1 1 1 1 1 1 |
| 0 0 1 1 | 1 1 1 0 1 1 1 1 1 1 |
| 0 1 0 0 | 1 1 1 1 0 1 1 1 1 1 |
| 0 1 0 1 | 1 1 1 1 1 0 1 1 1 1 |
| 0 1 1 0 | 1 1 1 1 1 1 0 1 1 1 |
| 0 1 1 1 | 1 1 1 1 1 1 1 0 1 1 |
| 1 0 0 0 | 1 1 1 1 1 1 1 1 0 1 |
| 1 0 0 1 | 1 1 1 1 1 1 1 1 1 0 |

8-to -1 MUX

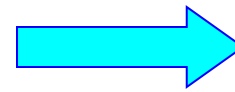
| S | $A_2 A_1 A_0$ | y |
|---|---------------|-------|
| 1 | × × × | 0 |
| 0 | 0 0 0 | a_0 |
| 0 | 0 0 1 | a_1 |
| 0 | 0 1 0 | a_2 |
| 0 | 0 1 1 | a_3 |
| 0 | 1 0 0 | a_4 |
| 0 | 1 0 1 | a_5 |
| 0 | 1 1 0 | a_6 |
| 0 | 1 1 1 | a_7 |



if: $B_2B_1B_0 = 110$, then $f_6 = a_6 = 0$

if: $C_2C_1C_0 = 110$, then $y = a_6 = 0$

if: $C_2C_1C_0 = 111$, then $y = a_7 = 1$



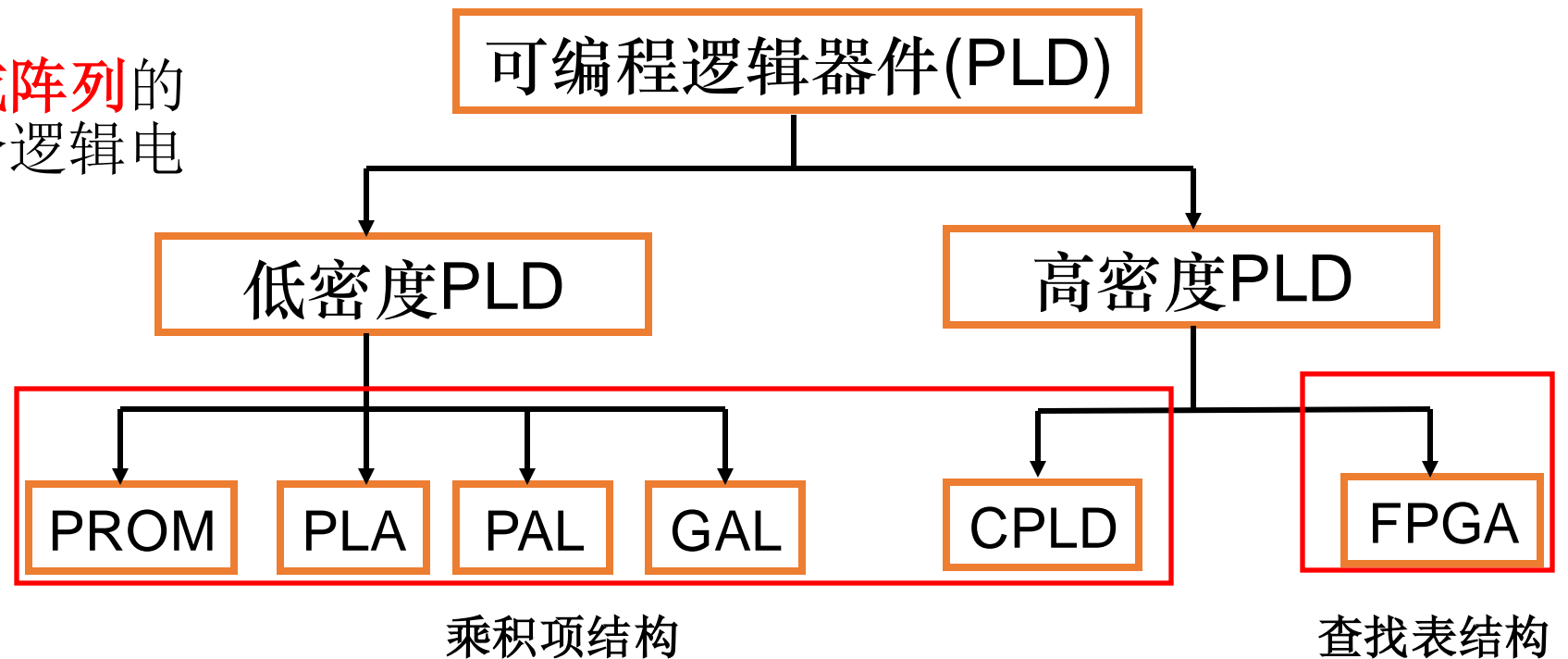
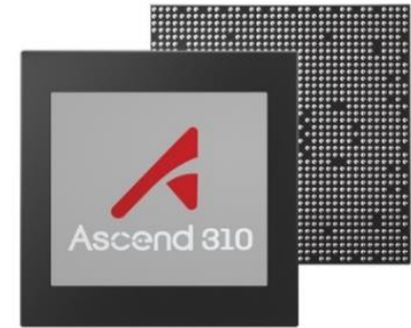
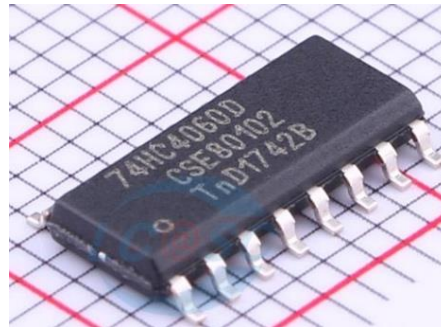
If $B_2B_1B_0 = C_2C_1C_0$,
Then $Z=0$

Unit 7 组合逻辑元件

- 多路复用器(multiplexers)
- 三态器件(Three-state Buffer)
- 译码器(Decoders)
- 编码器(Encoders)
- 奇偶校验器
- 比较器
- 利用MSI设计组合逻辑电路
- 只读存储器(ROM)

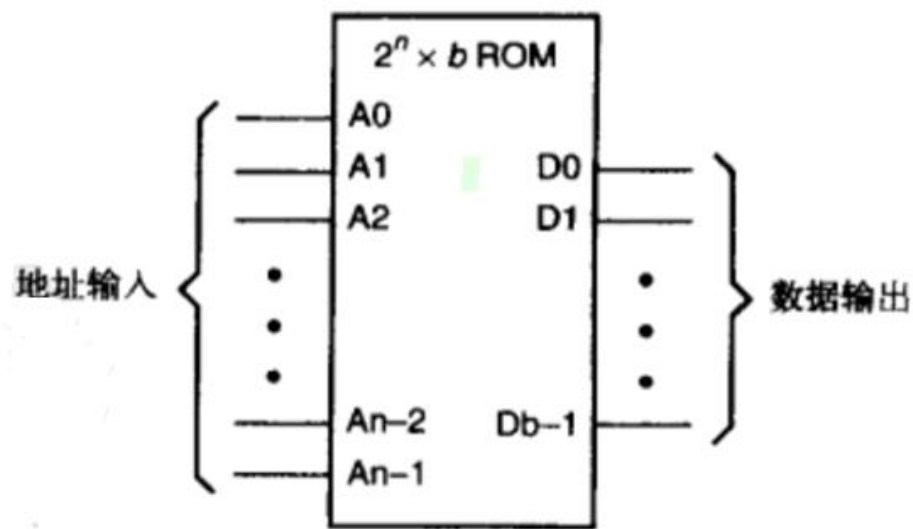
可编程逻辑器件

- 固定逻辑器件
- 任何组合逻辑函数均可以化为“**与或**”表达式，用“与门-或门”二级电路实现。
- 所以可以采用**与或阵列**的结构实现任何组合逻辑电路。

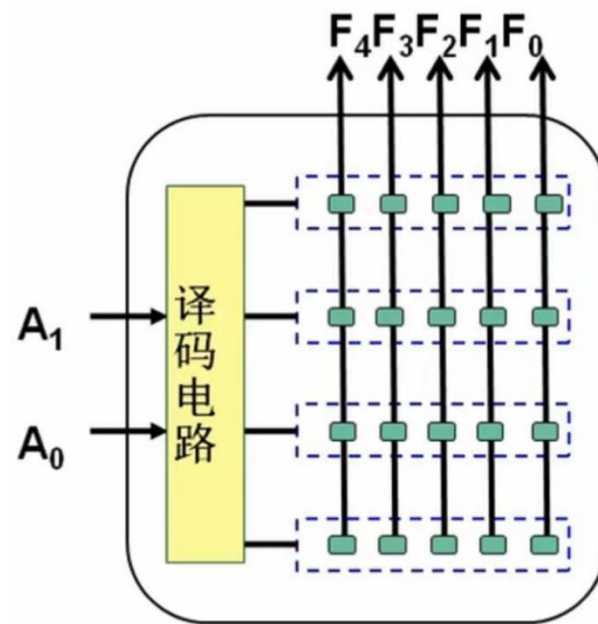


ROM (Read-Only Memory)

- **半导体存储器**：是一种具有 n 个输入 b 个输出的组合逻辑电路，能够存储大量二值数字信息，以**类似矩阵**的形式存储，每次取其中的一行。
- 输入被称为**地址输入** (address input)，通常命名为 A_0, A_1, \dots, A_{n-1} 。
- 输出被称为**数据输出** (data output)，通常命名为 D_0, D_1, \dots, D_{b-1} 。



一个 $2^n \times b$ ROM的基本结构



ROM和真值表

- ROM “存储” 了一个n输入、b输出的组合逻辑功能的**真值表**。
- 一个3输入、4输出的组合功能的真值表，可以被存储在一个 $2^3 * 4$ ($8 * 4$) 的只读存储器中。忽略延迟，ROM的数据输出总是等于真值表中由**地址输入所选择的那行输出位**。

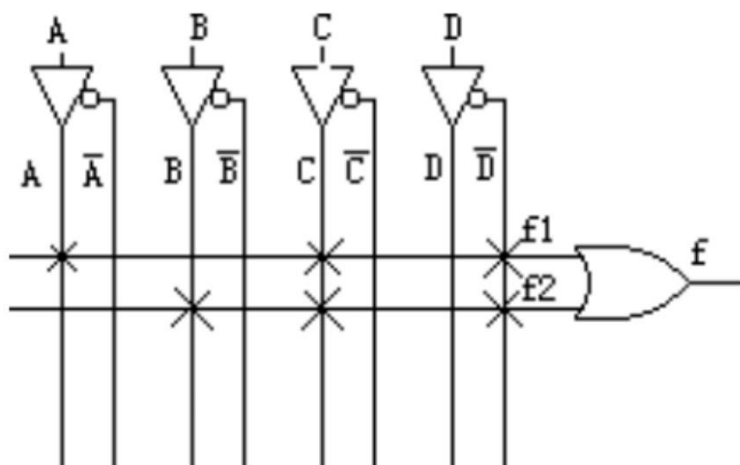
每一组**地址输入**
对应ROM的一个
存储单元的地址

| 输入 | | | 输出 | | | |
|----|----|----|----|----|----|----|
| A2 | A1 | A0 | D3 | D2 | D1 | D0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

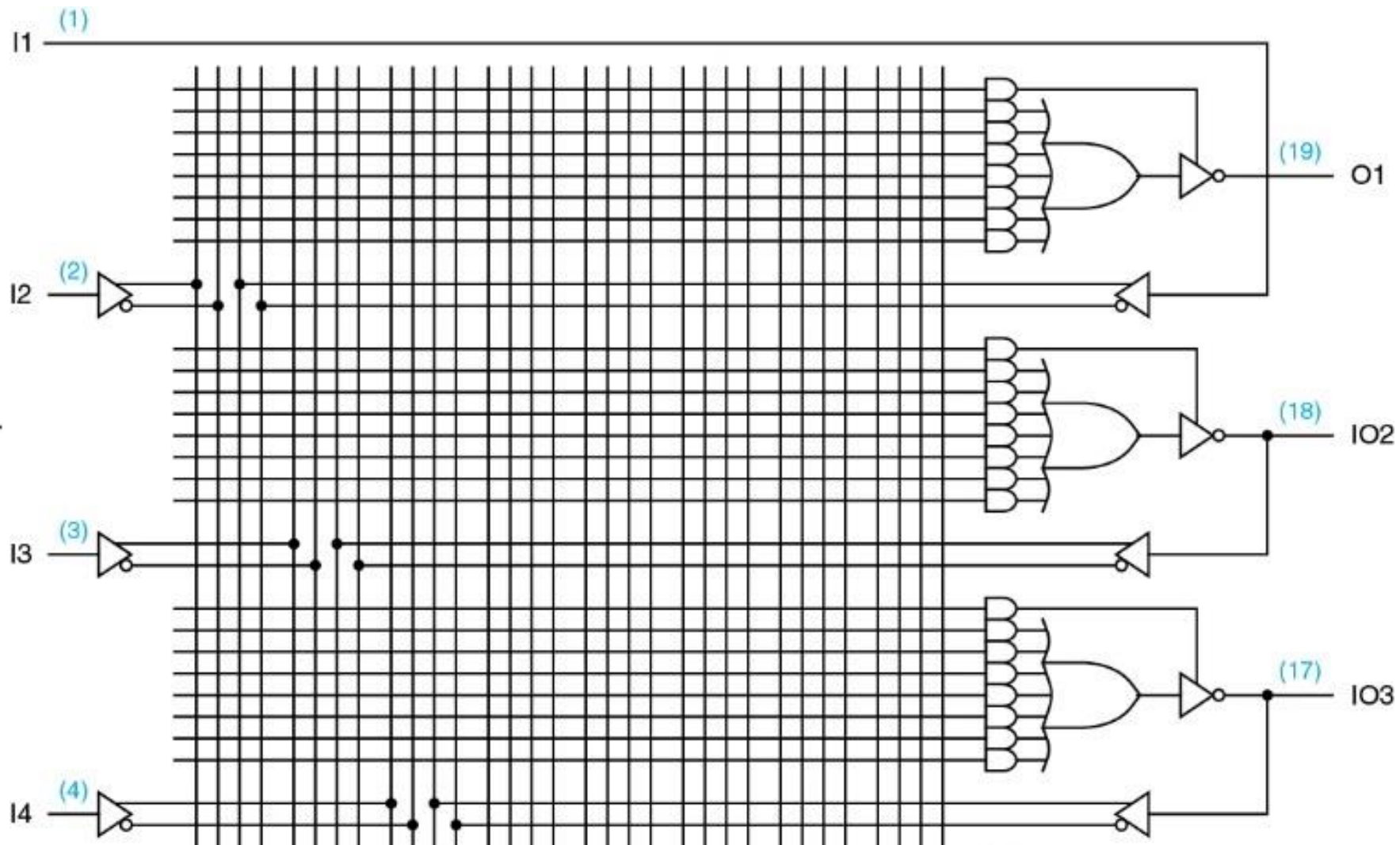
每一组输出对应
ROM的一个存储单
元中的**存放内容**。

一个3输入4输出组合逻辑函数的真值表（具有输出极性控制的2-4译码器）

乘积项结构

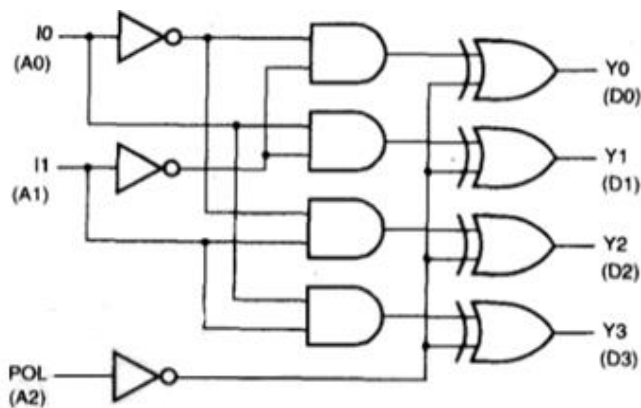


所表示的逻辑
函数是？

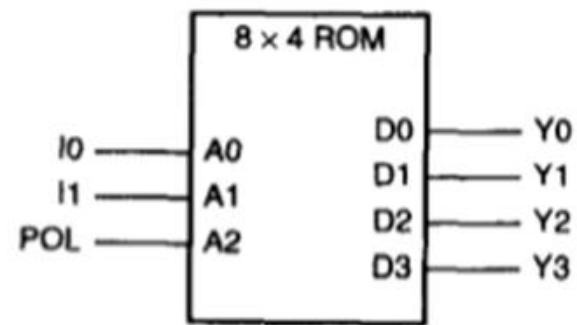


用ROM实现组合逻辑函数

- 两种不同的方式来构建译码器：
 - 使用分立的门
 - 用包含真值表的 8×4 ROM
- 使用ROM的物理实现并不是唯一的。



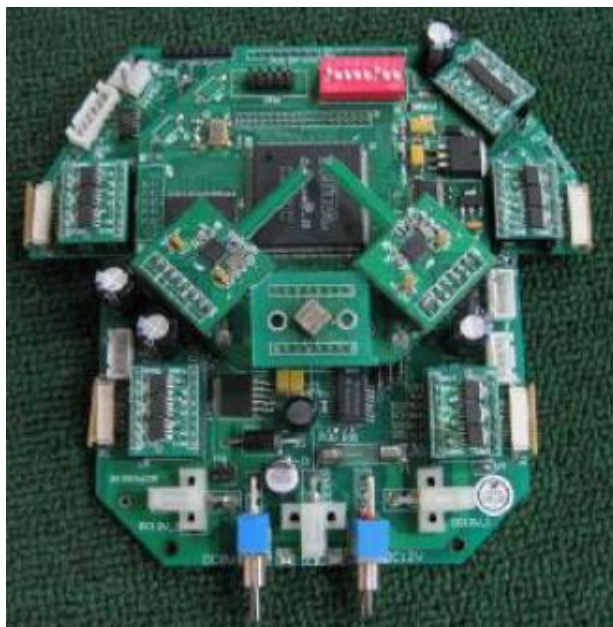
具有输出极性控制的2-4译码器



用存储真值表的 8×4 ROM构建2-4译码器

FPGA (Field Programmable Gate Array)

- 即现场可编程门阵列
- FPGA能完成任何数字逻辑功能，上至高性能计算，下至简单的74系列电路，也常用于ASIC流片前的原型验证。



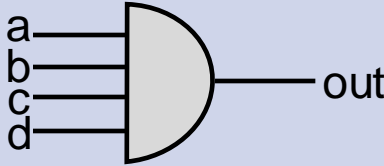
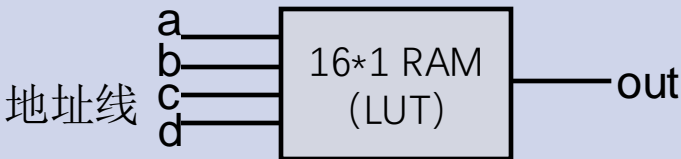
嵌入式硬件算法加速



协处理器

FPGA中的查找表（LUT）

- 例：使用LUT实现一个4与门电路逻辑功能

| 实际逻辑电路 | | LUT的实现方式 | |
|--|------|---|-----------|
|  | |  | |
| a、b、c、d | 逻辑输出 | 地址 | RAM中存储的内容 |
| 0000 | 0 | 0000 | 0 |
| 0001 | 0 | 0001 | 0 |
| | 0 | | 0 |
| 1111 | 1 | 1111 | 1 |

LUT本质就是RAM，主流的FPGA是5输入或6输入LUT

A,B,C,D由FPGA芯片的管脚输入后进入可编程连线，然后作为地址线连到LUT，LUT中已经事先写入了所有可能的逻辑结果，通过地址查找到相应的数据然后输出，这样组合逻辑就实现了。

FPGA内部结构

- 内部资源分类:
- 逻辑资源: **CLB**、块存储 (block ram)、DSP等;
- 连接资源: 可编程互联线 (PI)、输入输出块 (IOB) 等;
- 其他资源: 全局时钟网络、时钟管理模块等
- 高端FPGA还会集成ARM核、PCIE核等。
- 资源分布采用ASMBL架构, 相同资源以列方式排布。

