

数字逻辑设计

王鸿鹏

计算机科学与技术学院

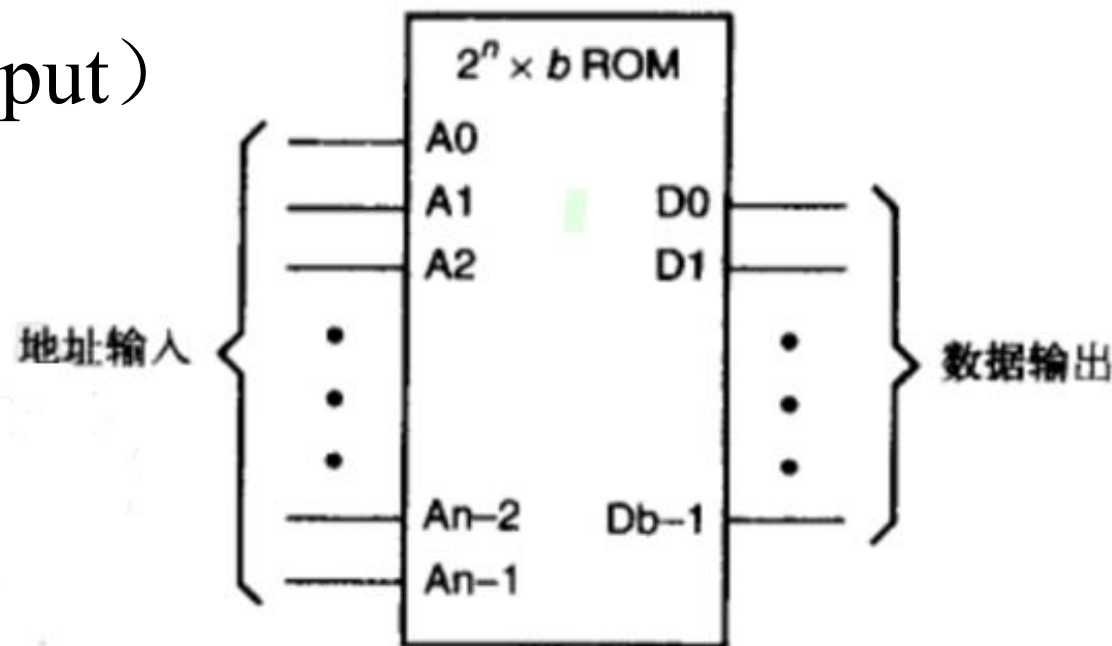
wanghp@hit.edu.cn

组合逻辑元件

- 只读存储器(ROM)
- 译码器 (Decoders)
- 多路复用器(multiplexers)
- 三态器件(Three-state Buffer)
- 编码器 (Encoders)
- 异或门和奇偶校验功能
- 比较器

ROM (Read-Only Memory)

- ROM是一种具有 n 个输入 b 个输出的组合逻辑电路。
- 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} 。



$2^n * b$ 位ROM

ROM和真值表

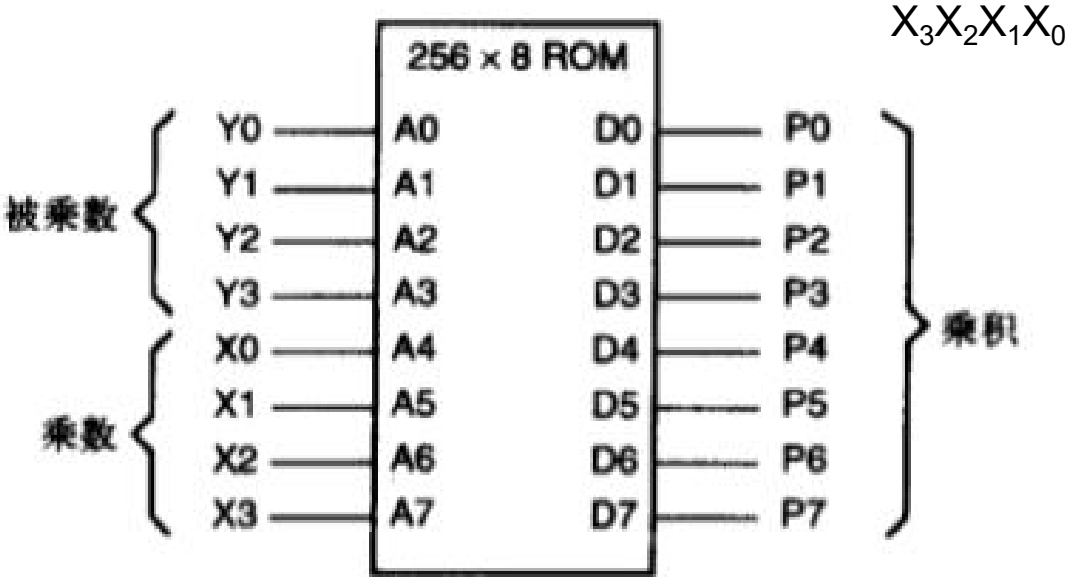
- ROM “存储” 了一个n输入、b输出的组合逻辑功能的真值表。
- 3输入、4输出的组合功能的真值表，可以被存储在一个 $2^3 * 4$ ($8 * 4$) 的只读存储器中。
- 忽略延迟，ROM的数据输出等于真值表中由地址输入所选择的那行输出。

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

具有输出极性控制的2-4译码器

用ROM实现4位*4位无符号二进制数乘法

- 多少种组合？
- 乘积最多为几位？



地址	$Y_3Y_2Y_1Y_0$															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
20	00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
30	00	03	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
40	00	04	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
50	00	05	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
60	00	06	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
70	00	07	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
80	00	08	10	18	20	28	30	38	40	48	50	58	60	68	70	78
90	00	09	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A0	00	0A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B0	00	0B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C0	00	0C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D0	00	0D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E0	00	0E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F0	00	0F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

基于ROM的设计方法的优点

- 通常可以用高级程序语言来计算存储在ROM中的内容。

```
#include <stdio.h>

/*Procedure to print d as a hex digit. */
void PrintHexDigit(int d)
{
    if (d<10) printf("%c", '0'+d);
    else printf("%c", 'A'+d-10);
}

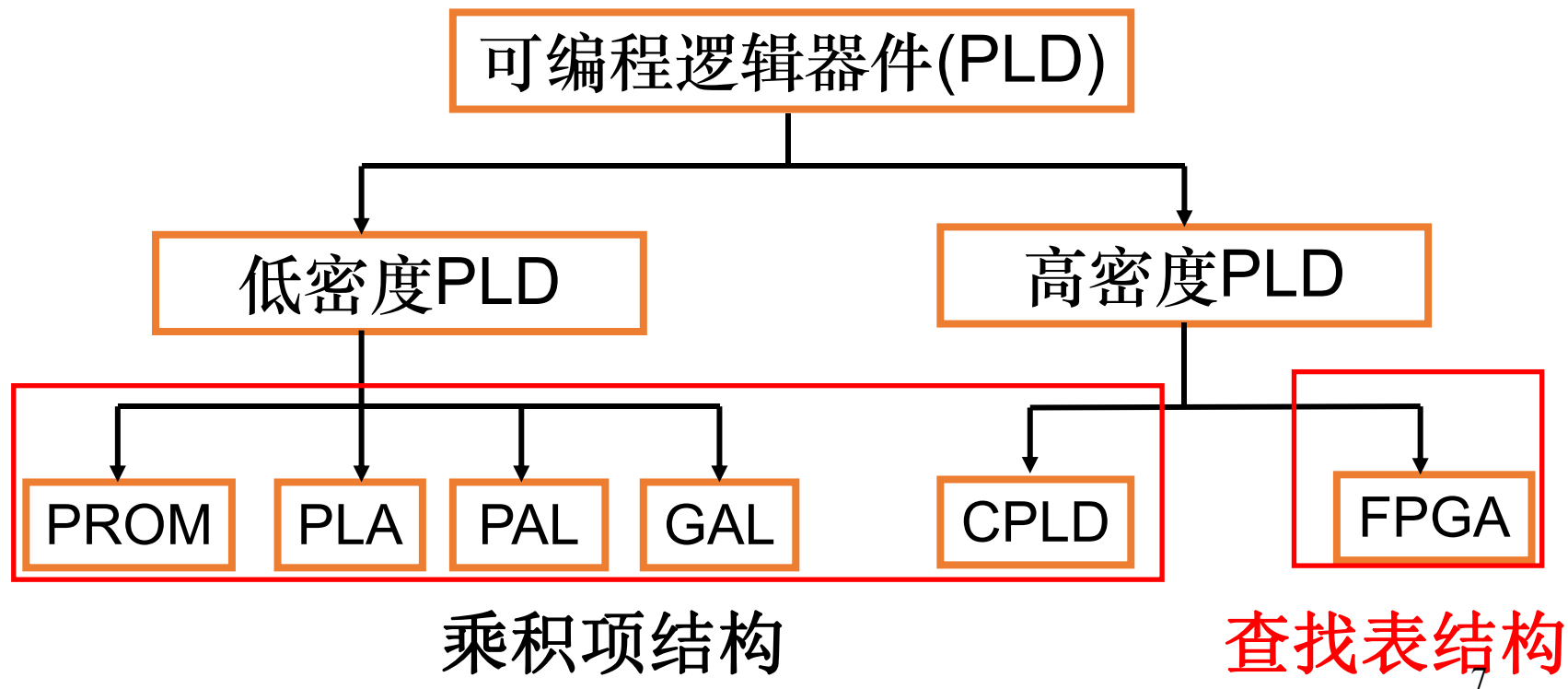
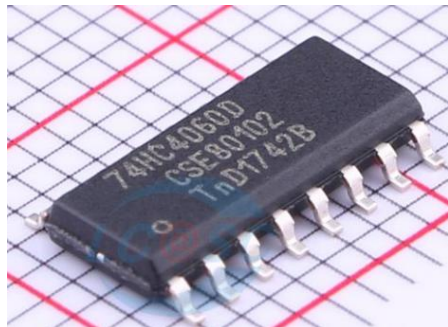
/*Procedure to print i as two hex digits. */
void PrintHex2(int i)
{
    PrintHexDigit((i / 16) % 16);
    PrintHexDigit(i % 16);
}

void main()
{
    int x, y;

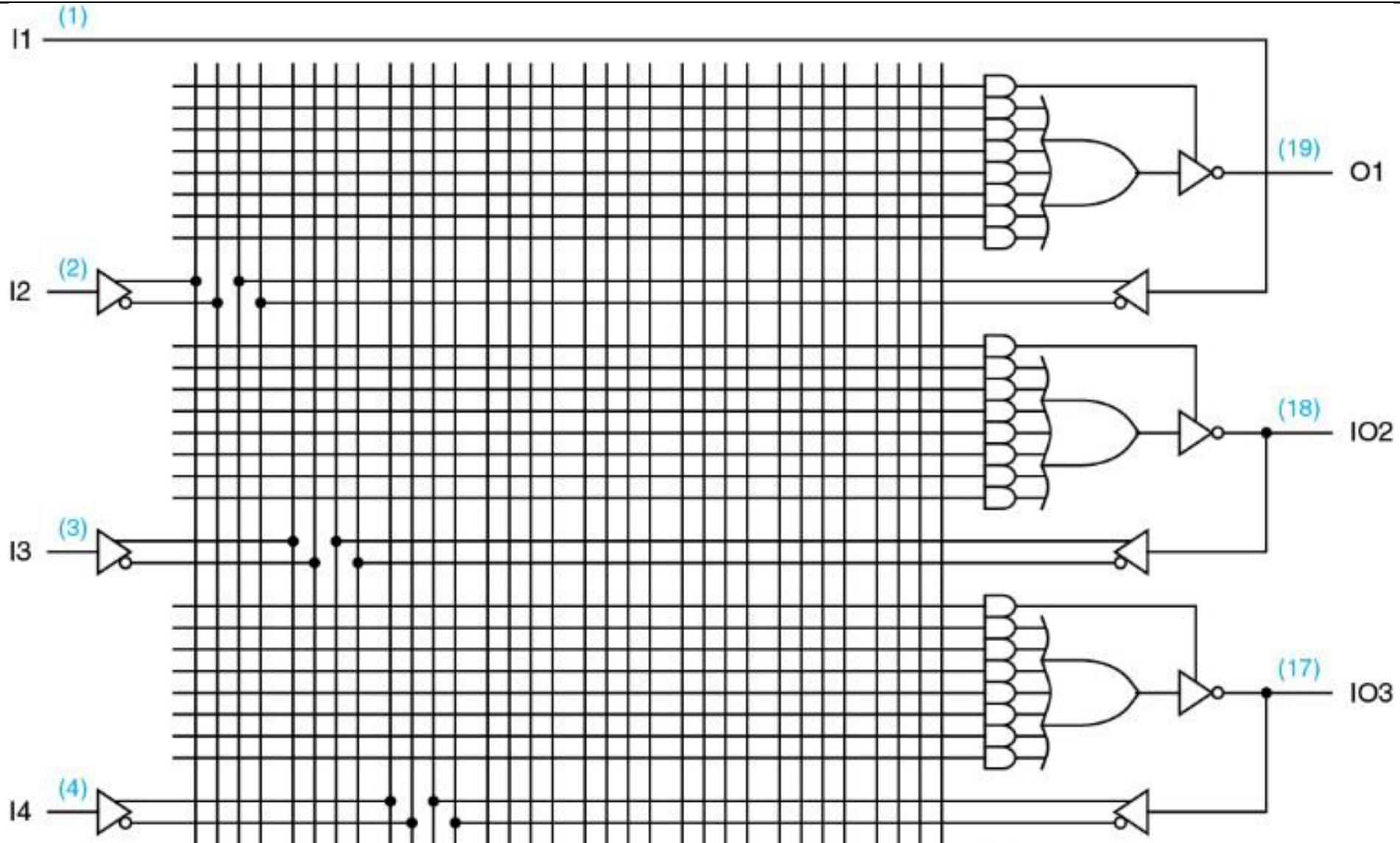
    for (x=0; x<=15; x++) {
        PrintHex2(x*16); printf(":");
        for (y=0; y<=15; y++) {
            printf(" ");
            PrintHex2(x*y);
        }
        printf("\n");
    }
}
```

可编程逻辑器件 (Programmable Logic Device)

- 固定逻辑器件



乘积项结构



FPGA (Field Programmable Gate Array)

- FPGA: 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

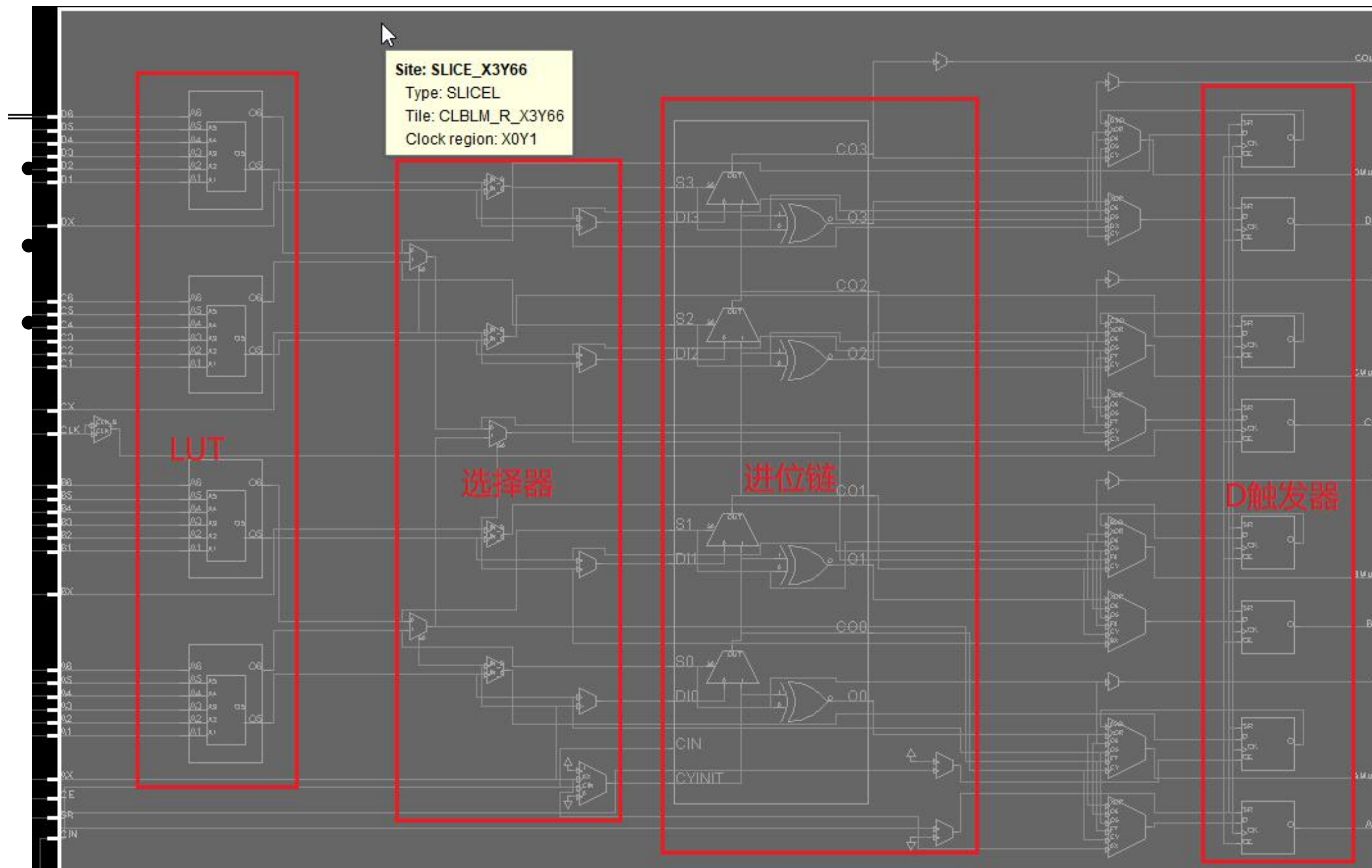
FPGA中的查找表（LookUp Table）

- 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

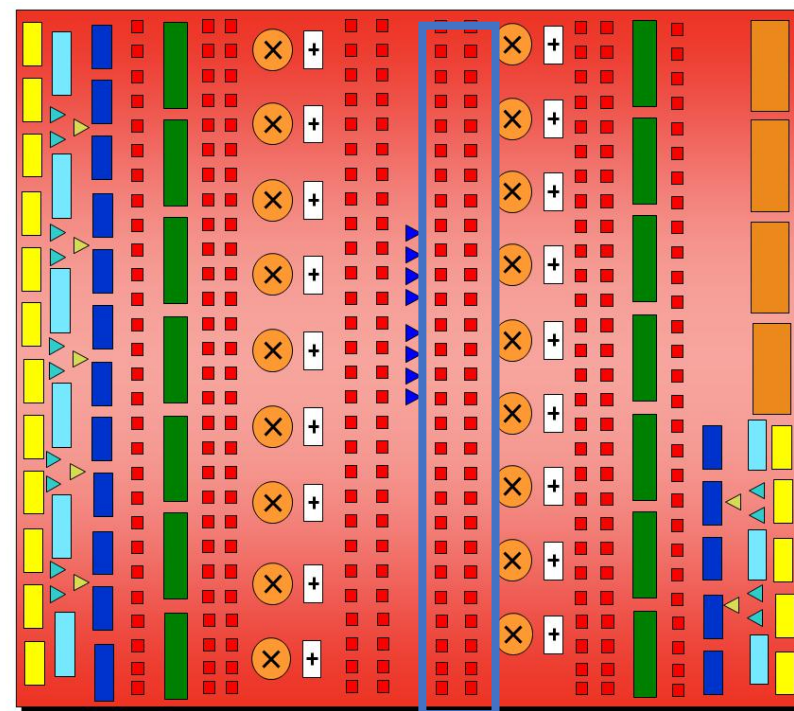
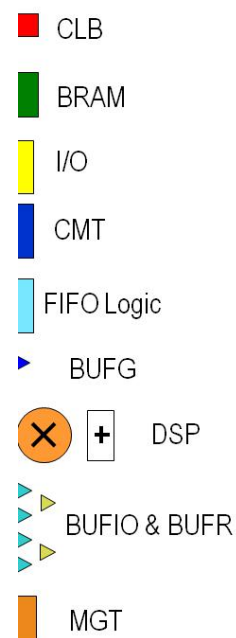
LUT本质就是RAM，主流的FPGA是5输入或6输入LUT

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。



哪部分有疑问?

- ☐ A ROM实现组合逻辑函数
- ☐ B 可编程逻辑器件
- ☐ C FPGA查找表
- ☐ D 无

提交

组合逻辑元件

- 只读存储器(ROM)
- 译码器(Decoders)
- 多路复用器(multiplexers)
- 三态器件(Three-state Buffer)
- 编码器(Encoders)
- 异或门和奇偶校验功能
- 比较器

译码器及分类

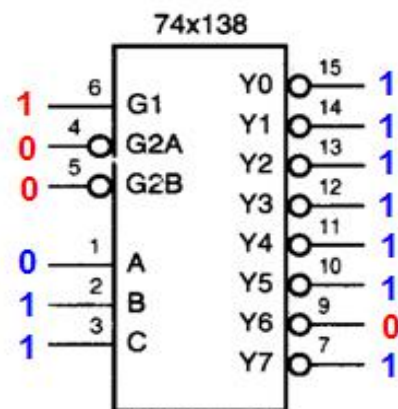
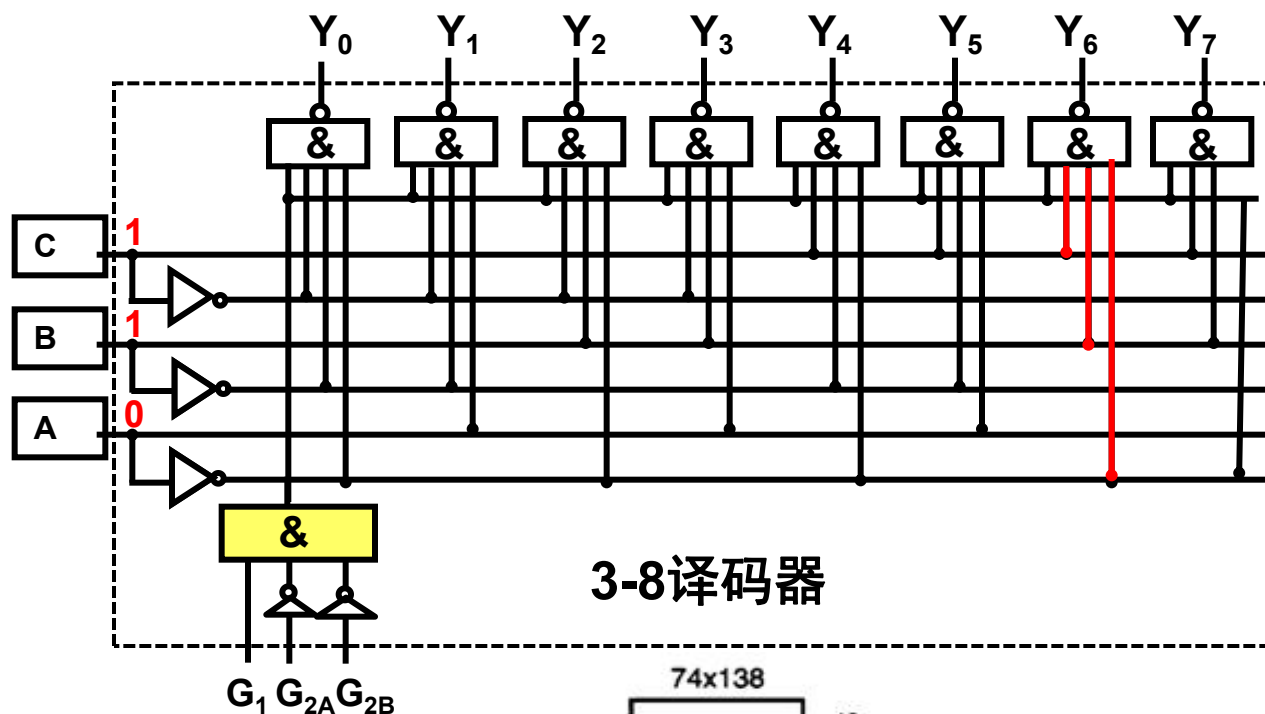
- 特点：多输入、多输出的组合逻辑电路
- 功能：将一种编码转换为另一种编码

分类	特点	译码演示
二进制译码器	<ul style="list-style-type: none">• 输入：n 位二进制码• 输出：N 位 ($N=2^n$)，每根输出线都与一个输入最小项唯一对应 (输出线编号值=最小项编号值)• 每个最小项输入，只能使 N 根输出线中的一个输出有效• N ($N=2^n$) 中取一译码器，也称最小项译码器。	<p>(3 线-8 线译码器)</p>
代码转换译码器	从一种编码转换为另一种编码 (例如：8421BCD码→余3码)	
显示译码器	将输入的编码信号转换为十进制码或其它特定编码， 用来驱动显示器件显示相应的文字符号。	

二进制译码器举例——3线-8线译码器

输入			译码输出							
C	B	A	Y ₀	Y ₁	Y ₂	Y ₃	Y ₄	Y ₅	Y ₆	Y ₇
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0

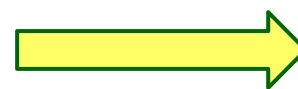
二进制译码器举例——3线-8线译码器



典型芯片: 74LS138

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

译码器输出: 低电平有效



$$y_i = \overline{m_i} = M_i$$

用ROM实现2-4译码器——1

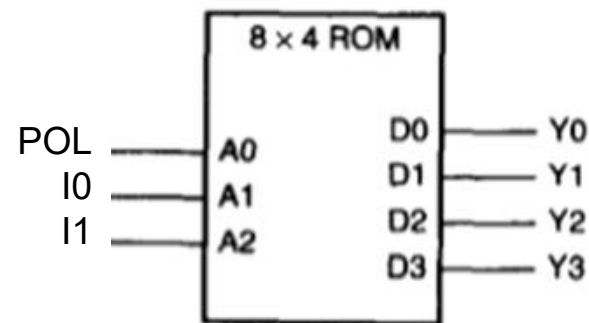
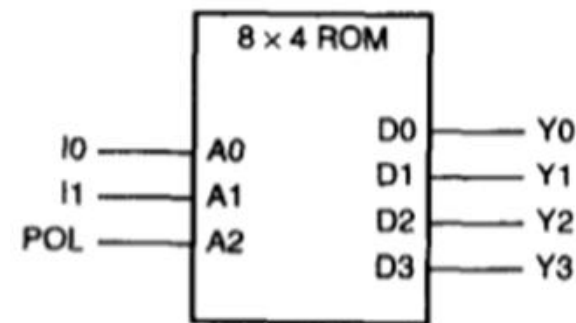
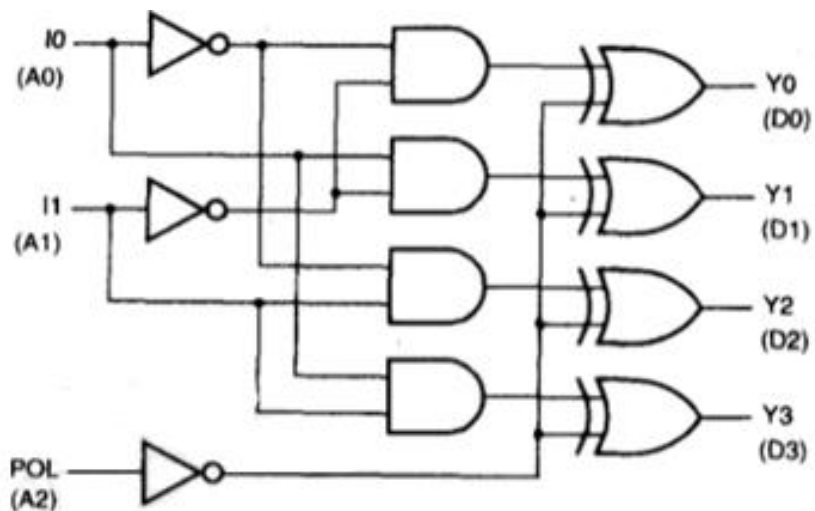
- ROM “存储” 了一个n输入、b输出的组合逻辑功能的真值表。
- 3输入、4输出的组合功能的真值表，可以被存储在一个 $2^3 * 4$ ($8 * 4$) 的只读存储器中。
- 忽略延迟，ROM的数据输出等于真值表中由地址输入所选择的那行输出。

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

具有输出极性控制的2-4译码器

用ROM实现2-4译码器——2

- 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。



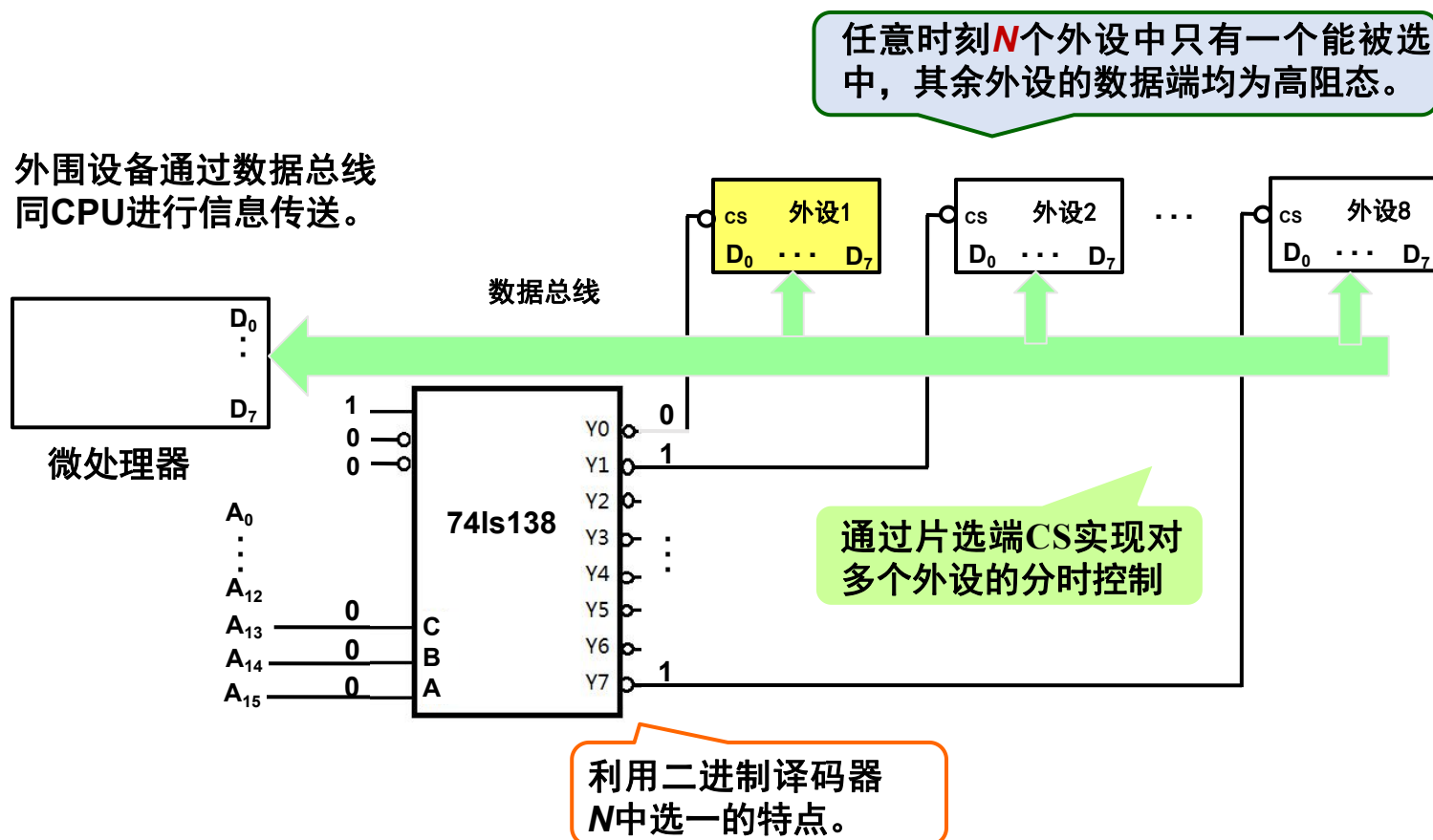
用8 * 4 ROM构建2-4译码器

具有输出极性控制的2-4译码器

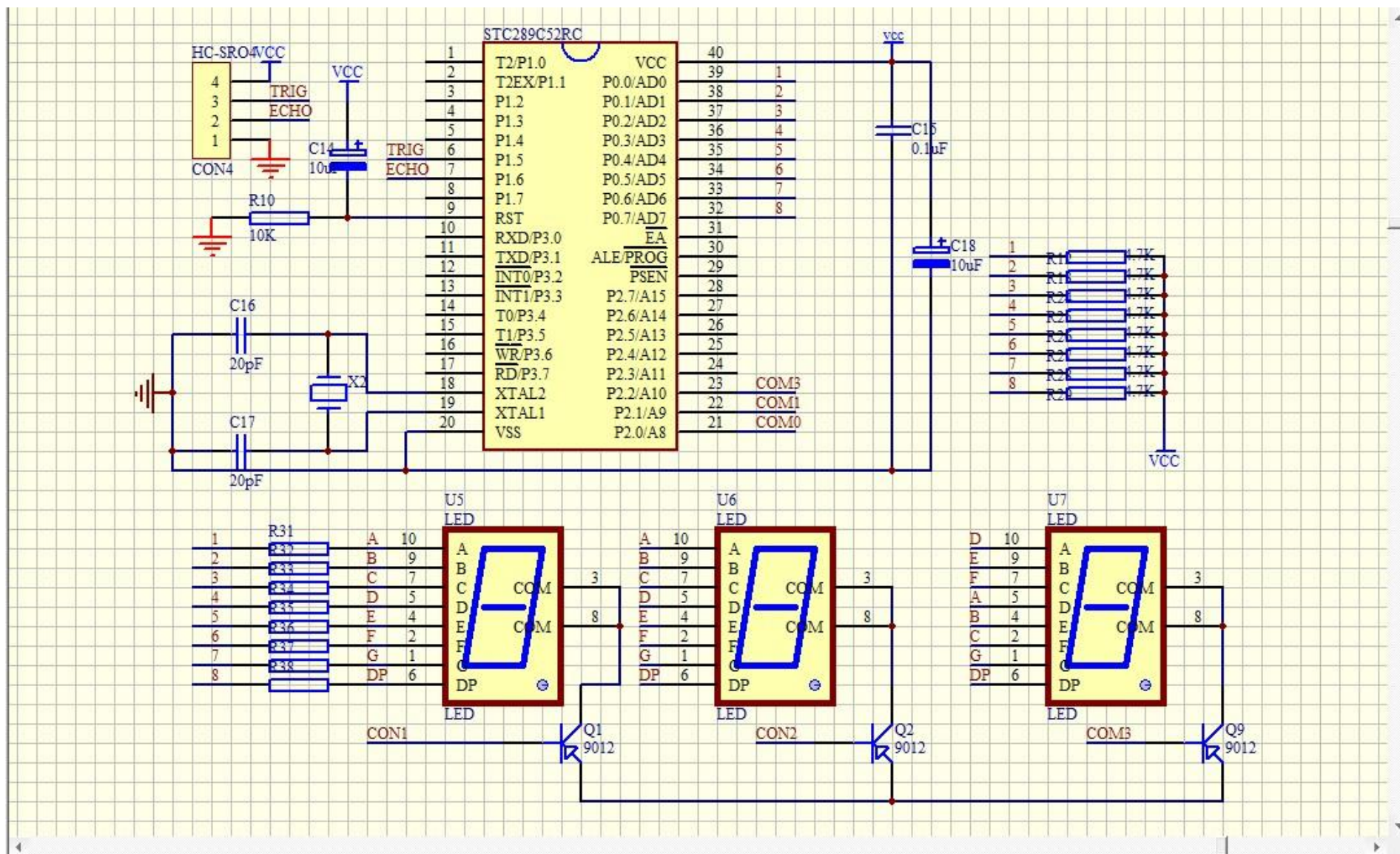
二进制译码器的典型应用——地址译码

■ 微处理器的地址译码

*假设A0—A7连接到各个外设的低8位地址线。

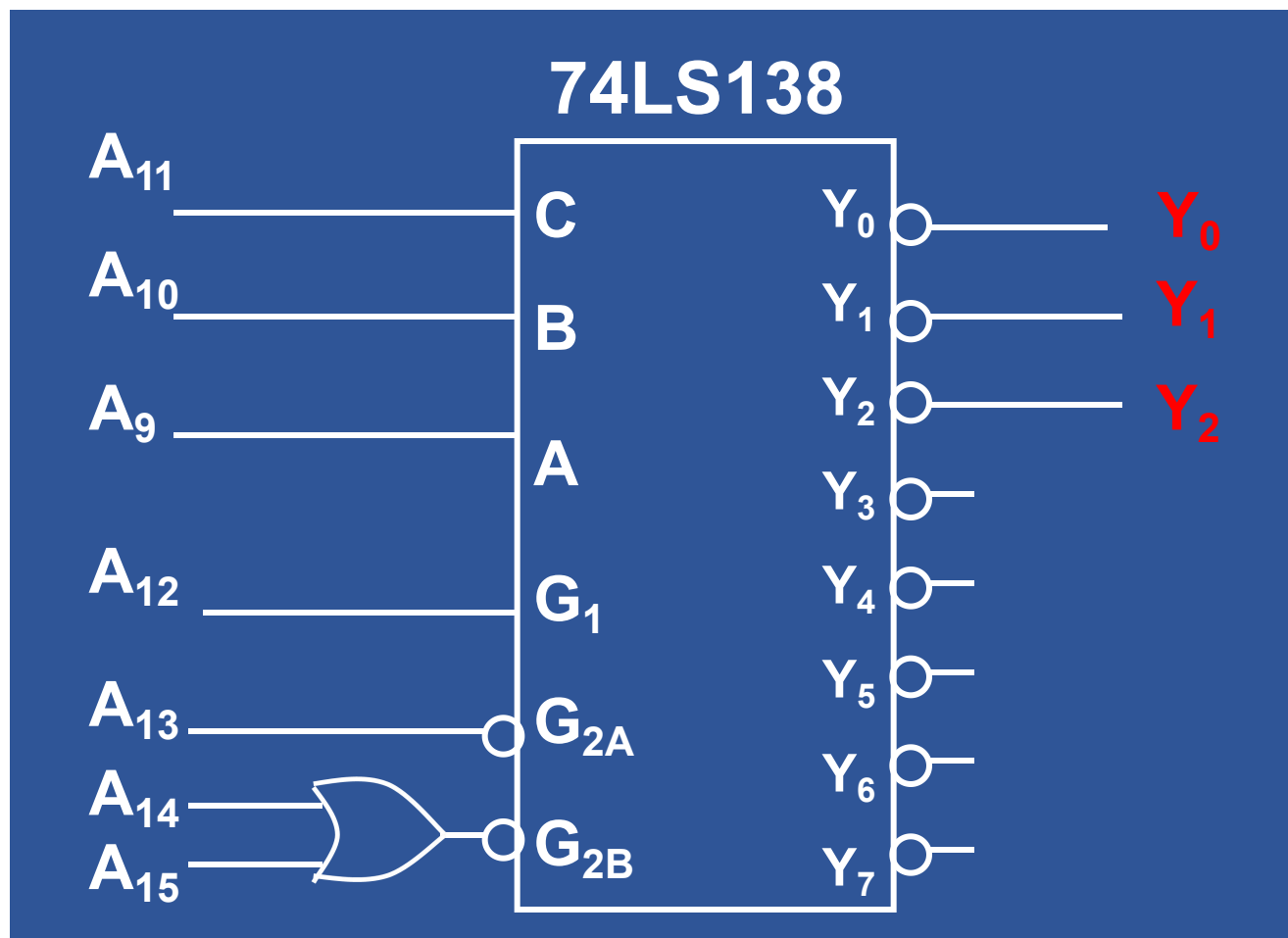


原理图示例



地址译码

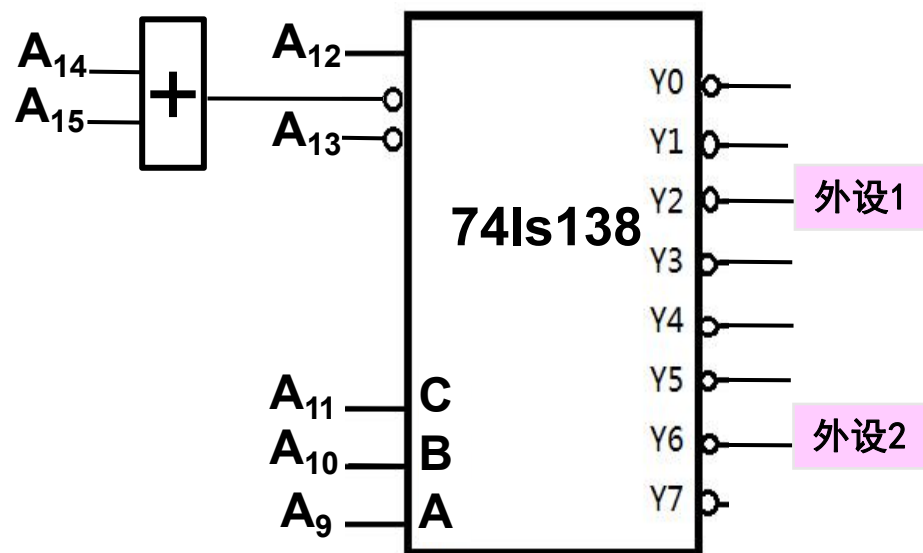
- 图示电路整个地址译码范围？ 各个外设的地址译码范围？



1 整个地址译码范围为： [填空1]H— [填空2]H

2 外设1的地址译码范围为： [填空3] H— [填空4] H

3 外设2的地址译码范围为： [填空5] H— [填空6] H



作答

地址译码例题

■ 地址译码

图示电路的整个地址译码范围？

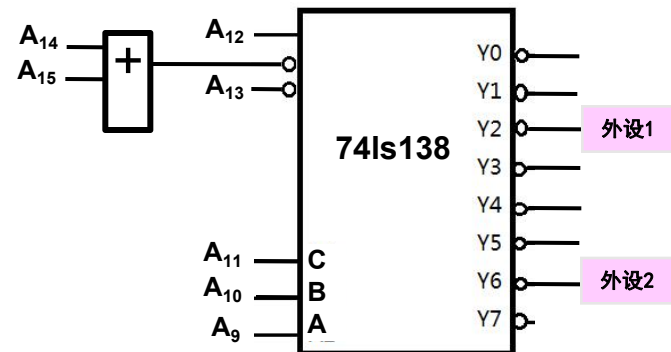
各个外设的地址译码范围？

整个译码器的地址译码范围：

A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	A ₁₀	A ₉	A ₈	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	
0	0	0	1	0	0	0	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	最小取值 1000H
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	最大取值 1FFFH

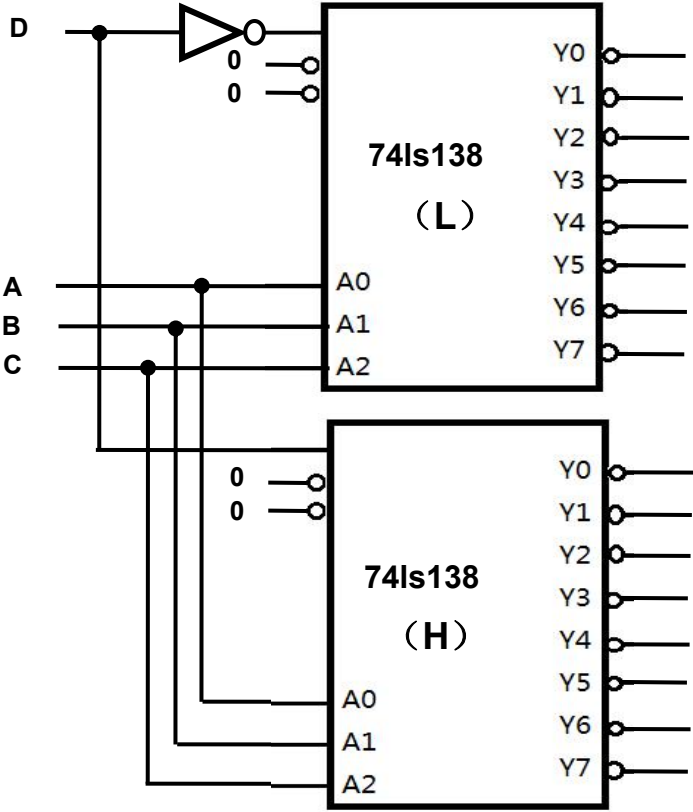
外设1的地址译码范围：

A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	A ₁₀	A ₉	A ₈	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	
0	0	0	1	0	1	0	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	⋯	最小取值 1400H
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	最大取值 15FFH



二进制译码器的典型应用——译码器级联

- 3线-8线译码器扩展为4线-16线译码



输入				译码输出L								译码输出H							
D	C	B	A	Y ₀	Y ₁	Y ₂	Y ₃	Y ₄	Y ₅	Y ₆	Y ₇	Y _{0H}	Y _{1H}	Y _{2H}	Y _{3H}	Y _{4H}	Y _{5H}	Y _{6H}	Y _{7H}
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

利用74LS138设计1位二进制全加器

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

$$S_i = \sum (1,2,4,7) = \overline{m_1 m_2 m_4 m_7}$$

$$c_{i-1} = \sum (3,5,6,7) = \overline{m_3 m_5 m_6 m_7}$$

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	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

$$y_i = \overline{m_i}$$

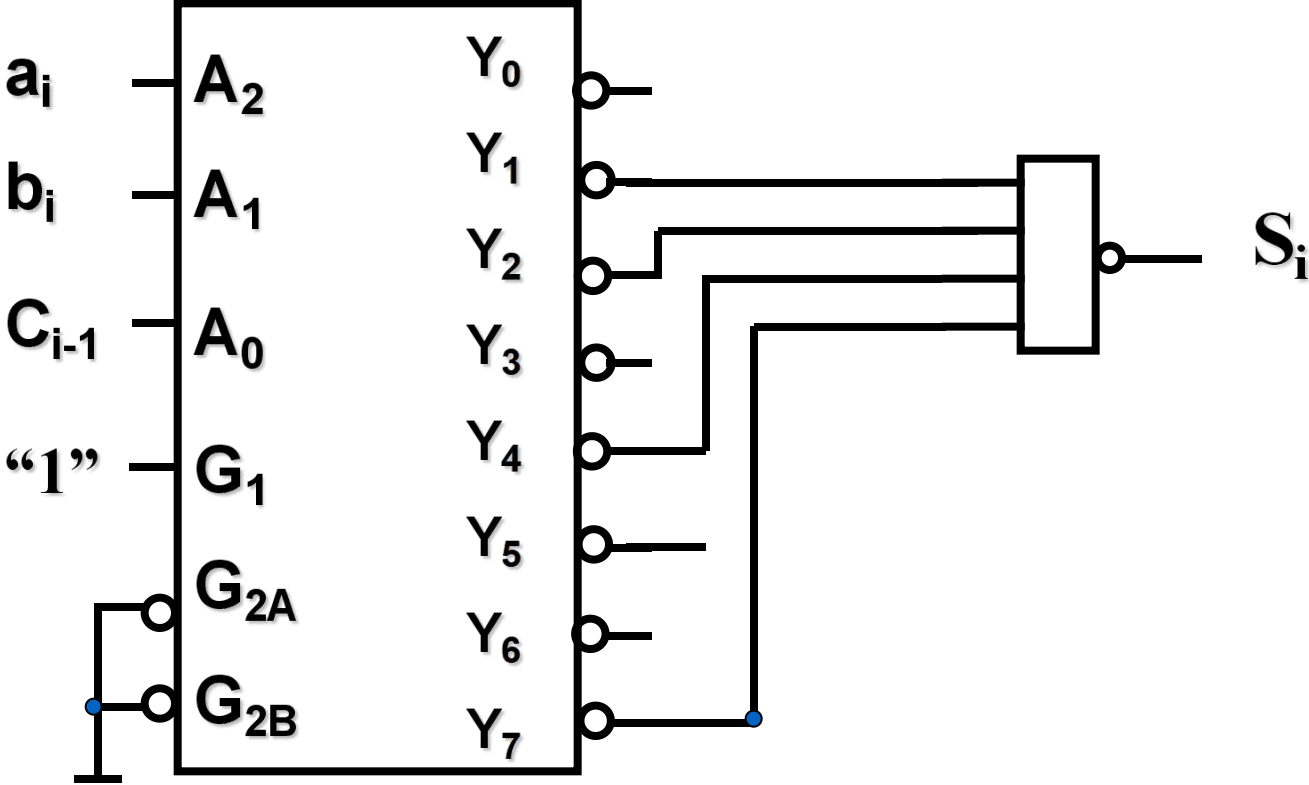
利用74LS138设计1位二进制全加器——续

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

$$S_i = \sum (1,2,4,7) = \overline{m_1} m_2 m_4 m_7$$

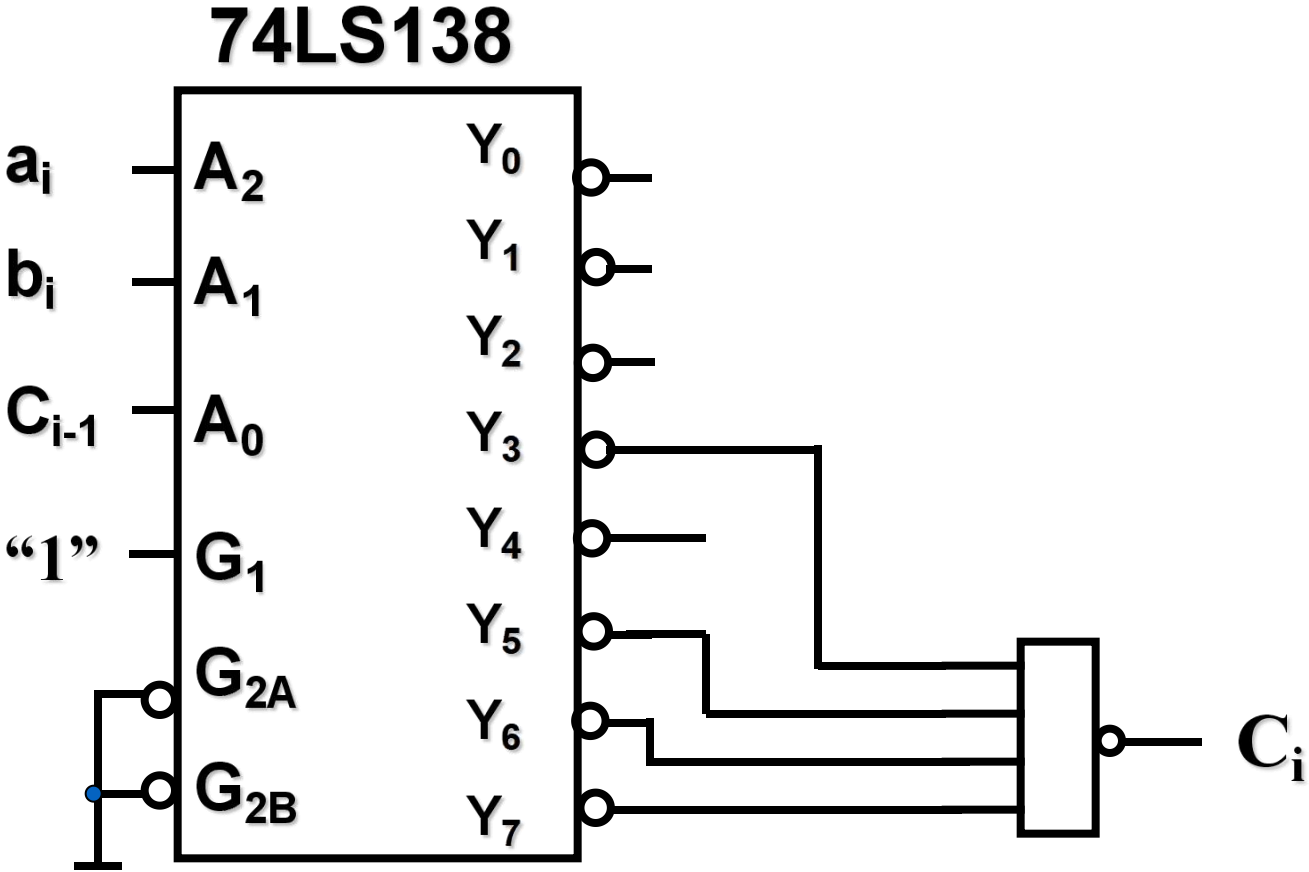
74LS138



利用74LS138设计1位二进制全加器——续

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



$$C_{i-1} = \sum (3, 5, 6, 7) = \overline{m_3} \overline{m_5} \overline{m_6} \overline{m_7}$$

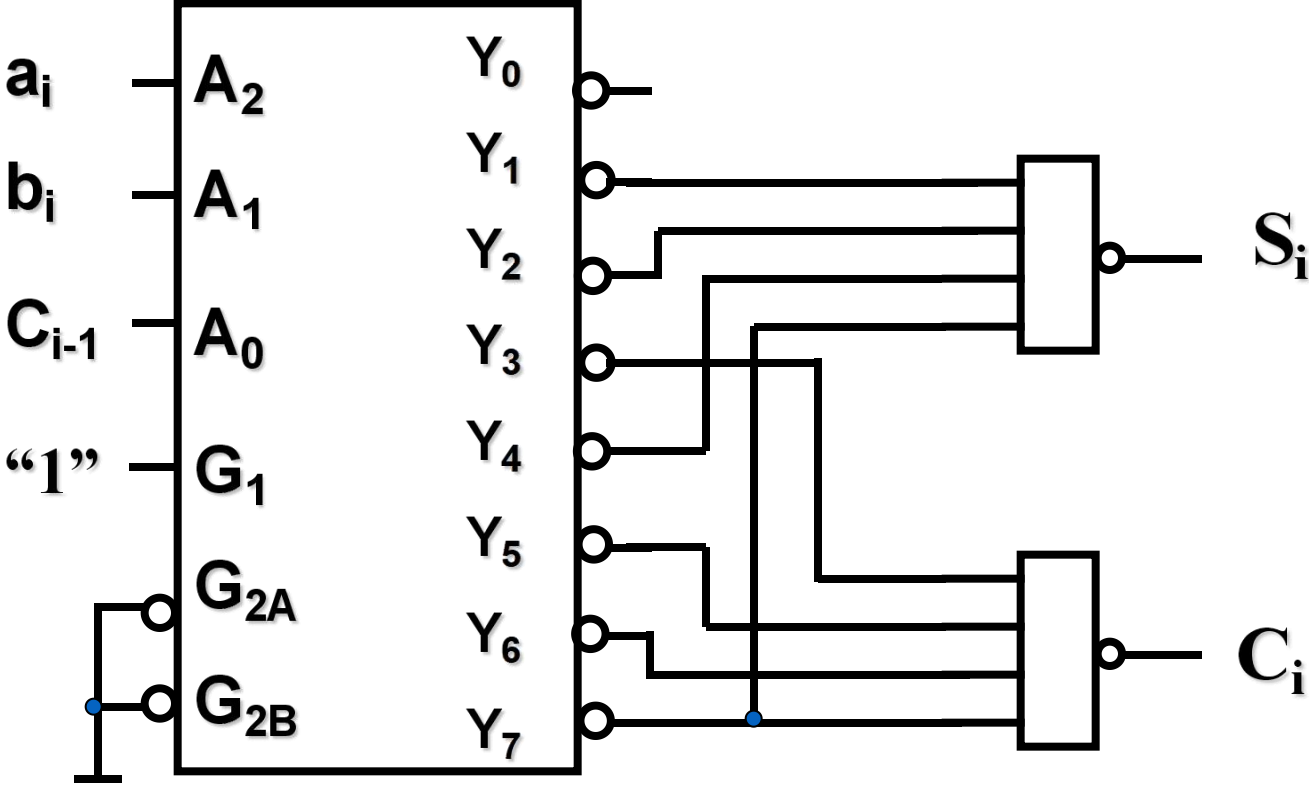
利用74LS138设计1位二进制全加器——续

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

$$S_i = \sum (1, 2, 4, 7) = \overline{\overline{m_1 m_2 m_4 m_7}}$$

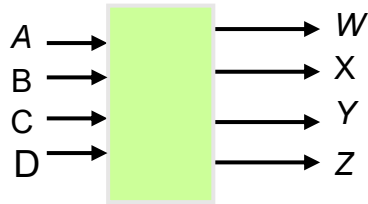
74LS138



$$C_{i-1} = \sum (3, 5, 6, 7) = \overline{\overline{m_3 m_5 m_6 m_7}}$$

编码转换译码器

- 例：设计一个译码器，
将输入的4位二进制数转换为典型格雷码。



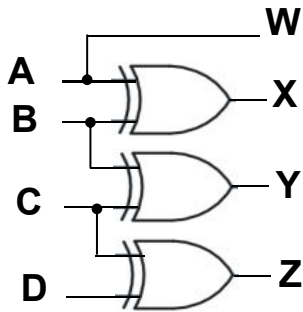
CD	00	01	11	10
AB				
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

CD	00	01	11	10
AB				
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$\begin{aligned} W &= A \\ X &= A \oplus B \\ Y &= B \oplus C \\ Z &= C \oplus D \end{aligned}$$

CD	00	01	11	10
AB				
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

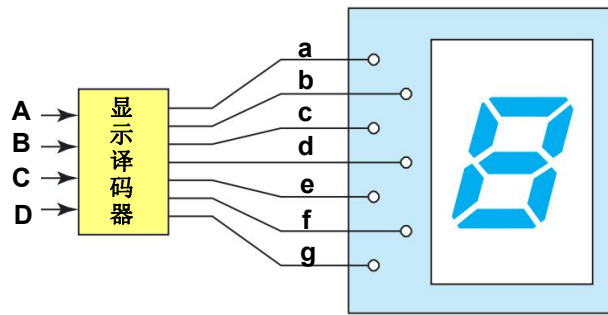
CD	00	01	11	10
AB				
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1



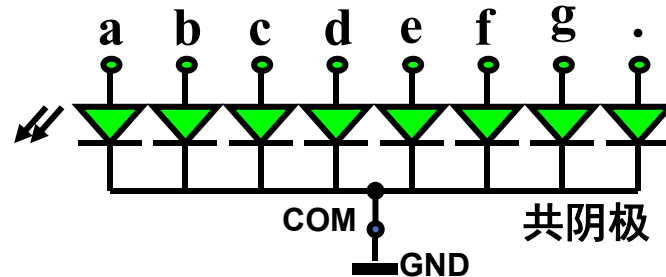
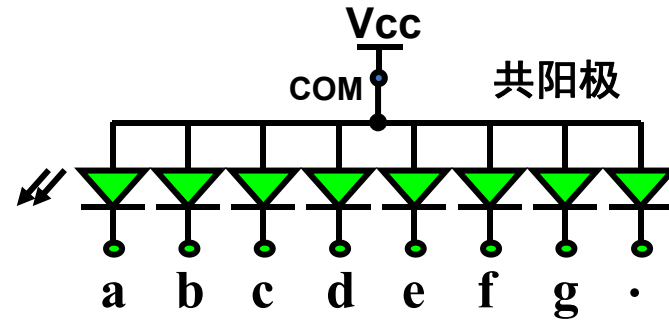
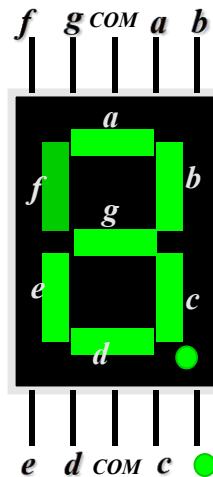
ABCD	WXYZ	ABCD	WXYZ
0000	0000	1000	1100
0001	0001	1001	1101
0010	0011	1010	1111
0011	0010	1011	1110
0100	0110	1100	1010
0101	0111	1101	1011
0110	0101	1110	1001
0111	0100	1111	1000

显示译码器

- 与显示器件（如数码管）配合，将输入代码转换为十进制码或特定编码，并在显示器件上显示相应的字形



七段数码管



8421BCD码驱动的**共阴极**七段
数码管显示译码器功能表

输入				译码输出						
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

共阴极七段数码管显示译码器逻辑表达式

a

CD \ AB	00	01	11	10
00	0	1	0	0
01	1	0	0	0
11	×	×	×	×
10	0	0	×	×

$$\bar{a} = A + C + BD + \bar{B}\bar{D}$$

b

CD \ AB	00	01	11	10
00	0	0	0	0
01	0	1	0	1
11	×	×	×	×
10	0	0	×	×

$$\bar{b} = \bar{B} + \bar{C}\bar{D} + CD$$

c

CD \ AB	00	01	11	10
00	0	0	0	1
01	0	0	0	0
11	×	×	×	×
10	0	0	×	×

$$\bar{c} = B + \bar{C} + D$$

组合逻辑元件

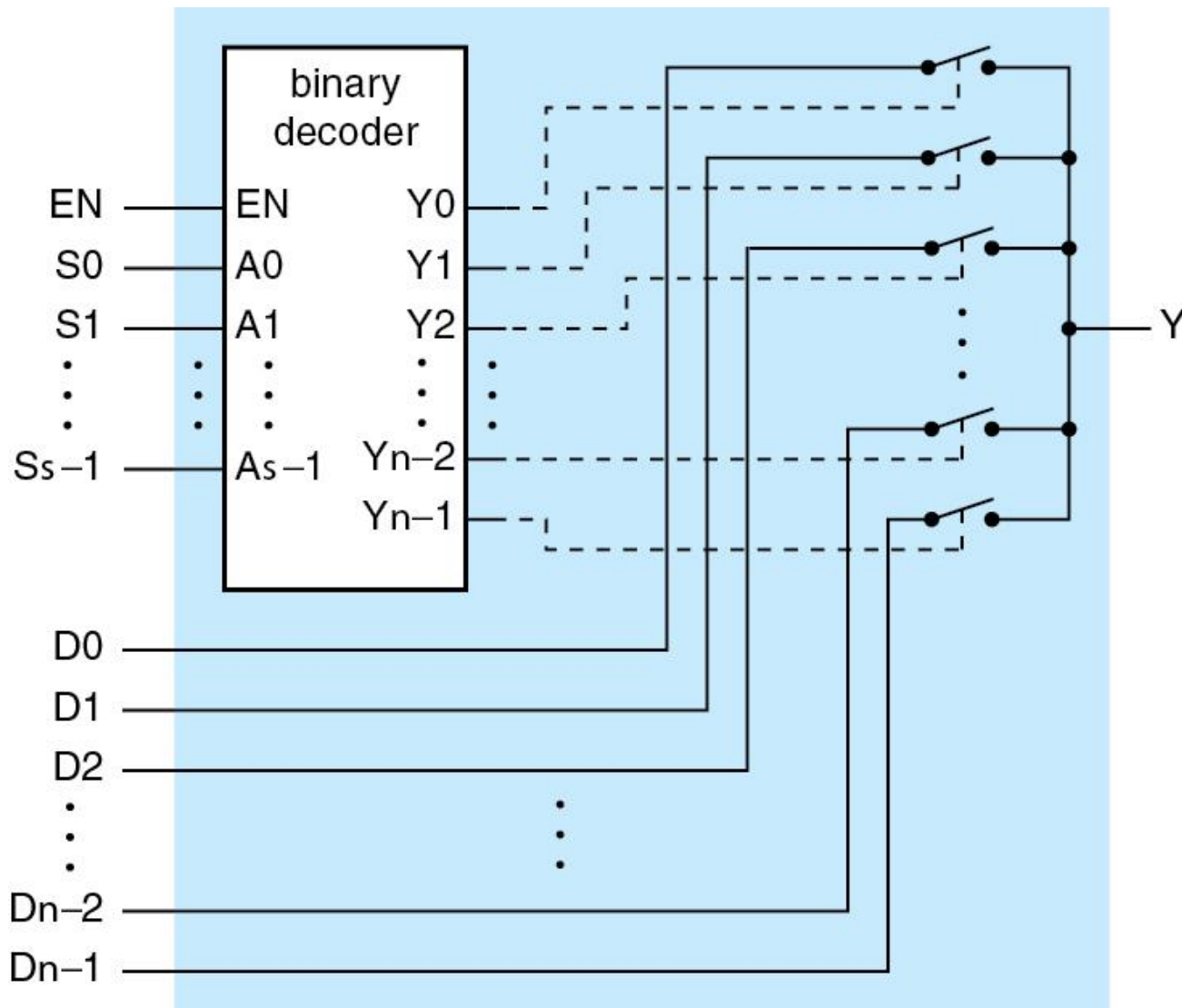
- 只读存储器(ROM)
- 译码器(Decoders)
- 多路复用器(multiplexers)
- 三态器件(Three-state Buffer)
- 编码器(Encoders)
- 异或门和奇偶校验功能
- 比较器

多路复用器/数据选择器/多路选择器/多路开关

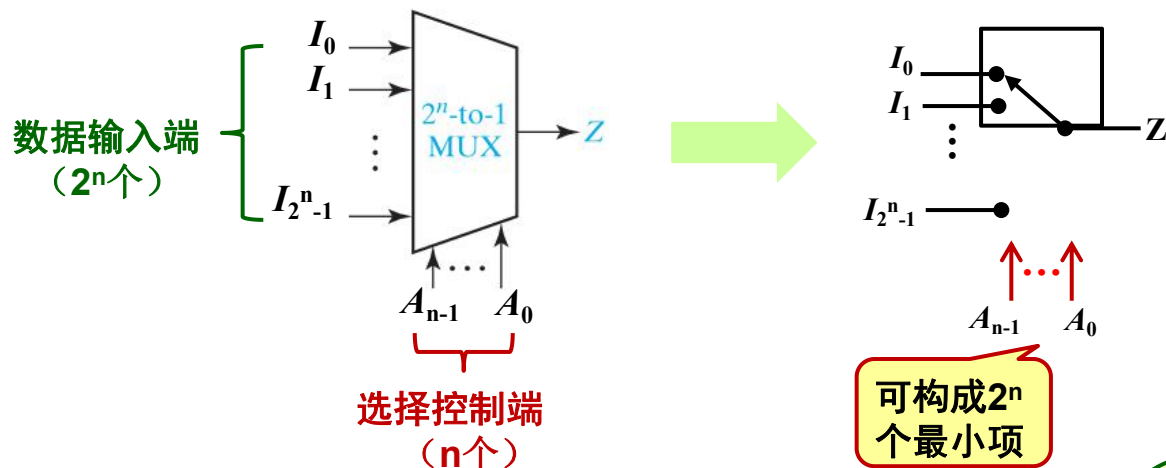
- 多路选择器是一种数据开关
- 它从 n 个数据源里选择一个数据，连到输出端
- 多路选择器与二进制译码器相似，因为它们都实现了选择功能，而且都会基于选择实现数据传送。
- 多路选择器可以被当做是一个由二进制译码器控制的单个开关的集合。

用一个译码器和开关实现的多路选择器

- 多路选择器可以被当做是一个由二进制译码器控制的单个开关的集合。



多路选择器



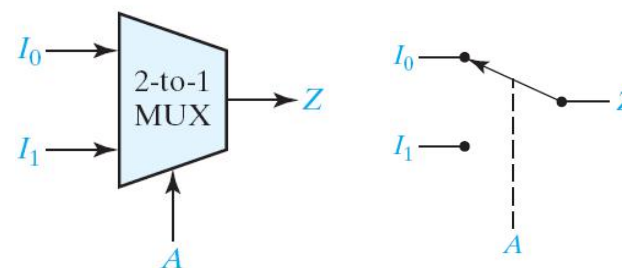
$$Z = \sum_{k=0}^{2^n-1} m_k I_k$$

控制端最小项 m_k 的序号 k , 指向了第 k 路数据输入端 I_k 。

m_k —— n 个控制变量的最小项

I_k —— 第 k 路数据输入

2选1多路选择器



$$Z = A' I_0 + A I_1$$

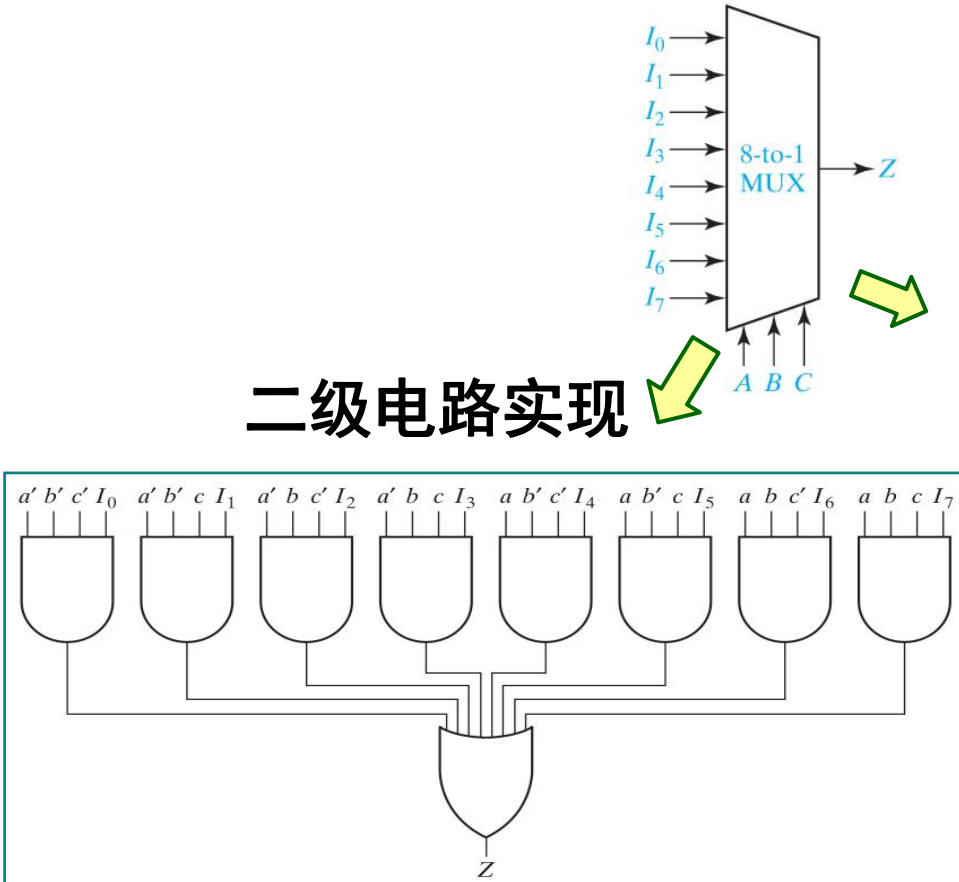
多路选择器的功能:

- ① 从多路输入中选择一个送往输出端 (2^n 选1);
- ② 选择哪一路输入送到输出端由控制信号决定;

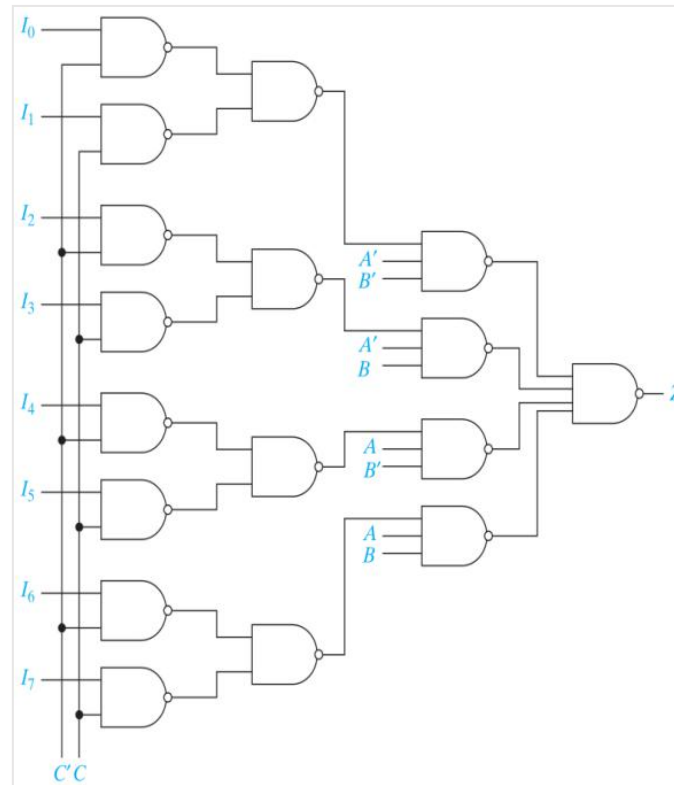
用途: 实现多通道的数据传送;

8选1多路选择器

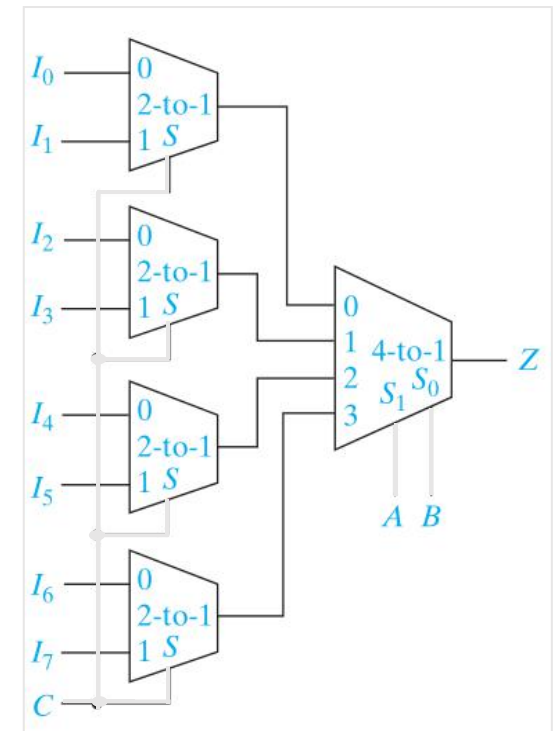
二级电路实现



单一逻辑门实现



多路选择器级联实现



$$Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$$

多路选择器的典型应用——实现逻辑函数

- 多路选择器的表达式

$$Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$$

$$Z = \sum_{k=0}^{2^n-1} m_k I_k$$

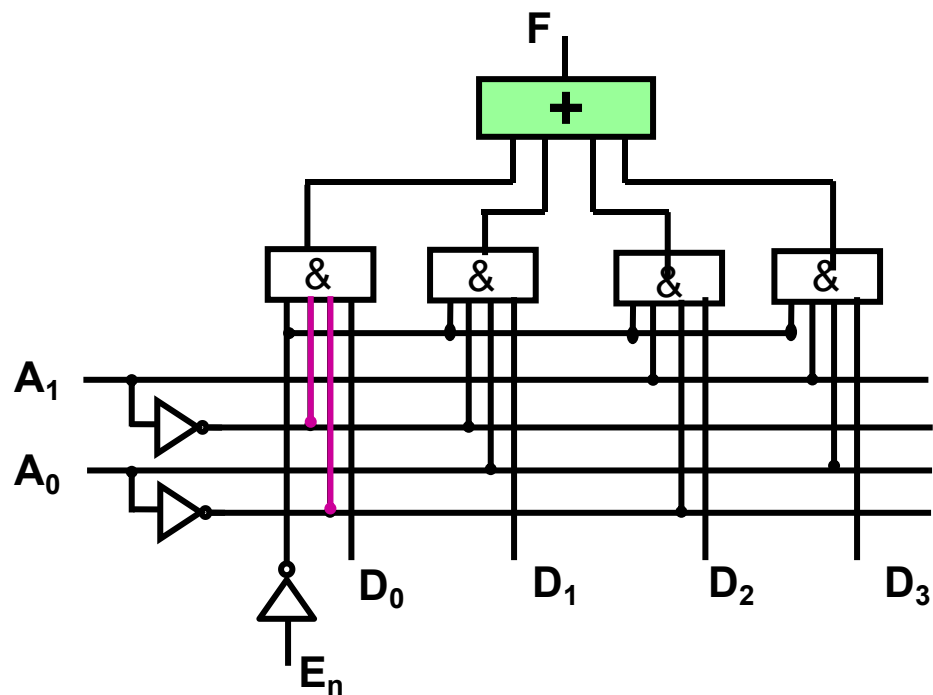
- 逻辑函数的标准与或式

$$F = \sum_{k=0}^{2^n-1} m_k (\text{对应的第}k\text{行输出为}1) = \sum_{k=0}^{2^n-1} m_k F_k$$

多路选择器的典型应用

功能表

E_n	A_1	A_0	F
1	x	x	0
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3



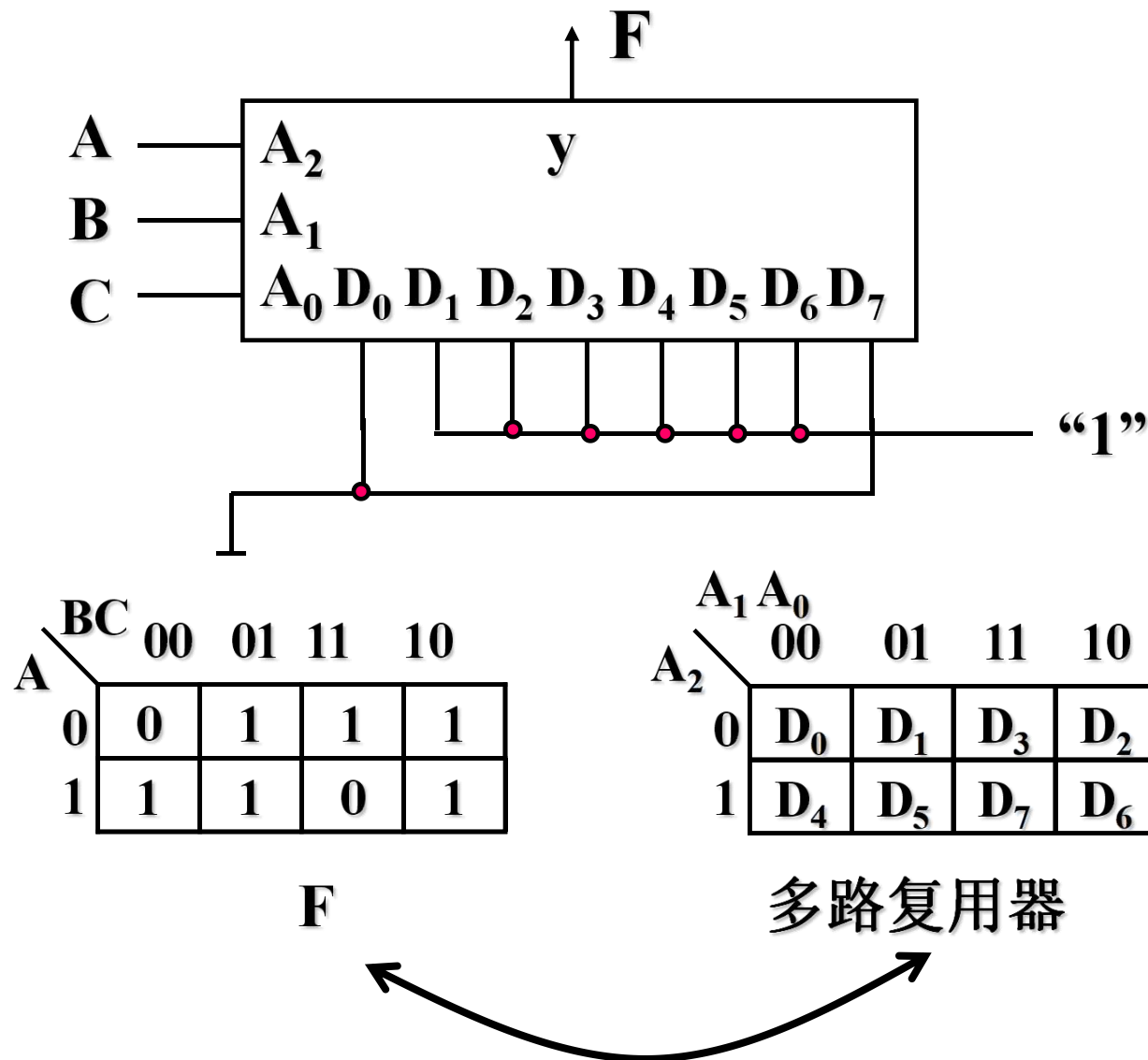
$$F = \overline{E_n} (D_0 \overline{A_1} \overline{A_0} + D_1 \overline{A_1} A_0 + D_2 A_1 \overline{A_0} + D_3 A_1 A_0)$$

- 例如：用四选一多路选择器实现 $F = A \text{ XOR } B$

使用八选一多路复用器实现逻辑函数

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

$A_2 A_1 A_0$	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



四变量卡诺图变换

CD \ AB	00	01	11	10
00	1	0	1	0
01	1	1	0	1
11	0	0	1	0
10	0	1	0	1

		<div>D'</div>			
		0		1	
C \ AB					
00	$D' * 1$	$D * 0$	$D * 1$	$D' * 0$	
01	$D' * 1$	$D * 1$	$D * 0$	$D' * 1$	
11	$D' * 0$	$D * 0$	$D * 1$	$D' * 0$	
10	$D' * 0$	$D * 1$	$D * 0$	$D' * 1$	

四变量卡诺图降维

C AB	0		1	
	00	01	11	10
00	$D' * 1$	$D * 0$	$D * 1$	$D' * 0$
01	$D' * 1$	$D * 1$	$D * 0$	$D' * 1$
11	$D' * 0$	$D * 0$	$D * 1$	$D' * 0$
10	$D' * 0$	$D * 1$	$D * 0$	$D' * 1$

C AB	0		1	
	00	01	11	10
00	$D' * 1 + D * 0$	$D * 1 + D' * 0$		
01	$D' * 1 + D * 1$	$D * 0 + D' * 1$		
11	$D' * 0 + D * 0$	$D * 1 + D' * 0$		
10	$D' * 0 + D * 1$	$D * 0 + D' * 1$		

五变量卡诺图降维

$$F=f(A,B,C,D,E)$$

		CDE							
		000	001	011	010	110	111	101	100
AB	00	0	1	3	2	6	7	5	4
	01	8	9	11	10	14	15	13	12
	11	24	25	27	26	30	31	29	28
	10	16	17	19	18	22	23	21	20

		CD			
		00	01	11	10
AB	00	$f_0(E,E')$	$f_1(E,E')$	f_3	f_2
	01	f_4	f_5	f_7	f_6
	11	f_{12}	f_{13}	f_{15}	f_{14}
	10	f_8	f_9	f_{11}	f_{10}

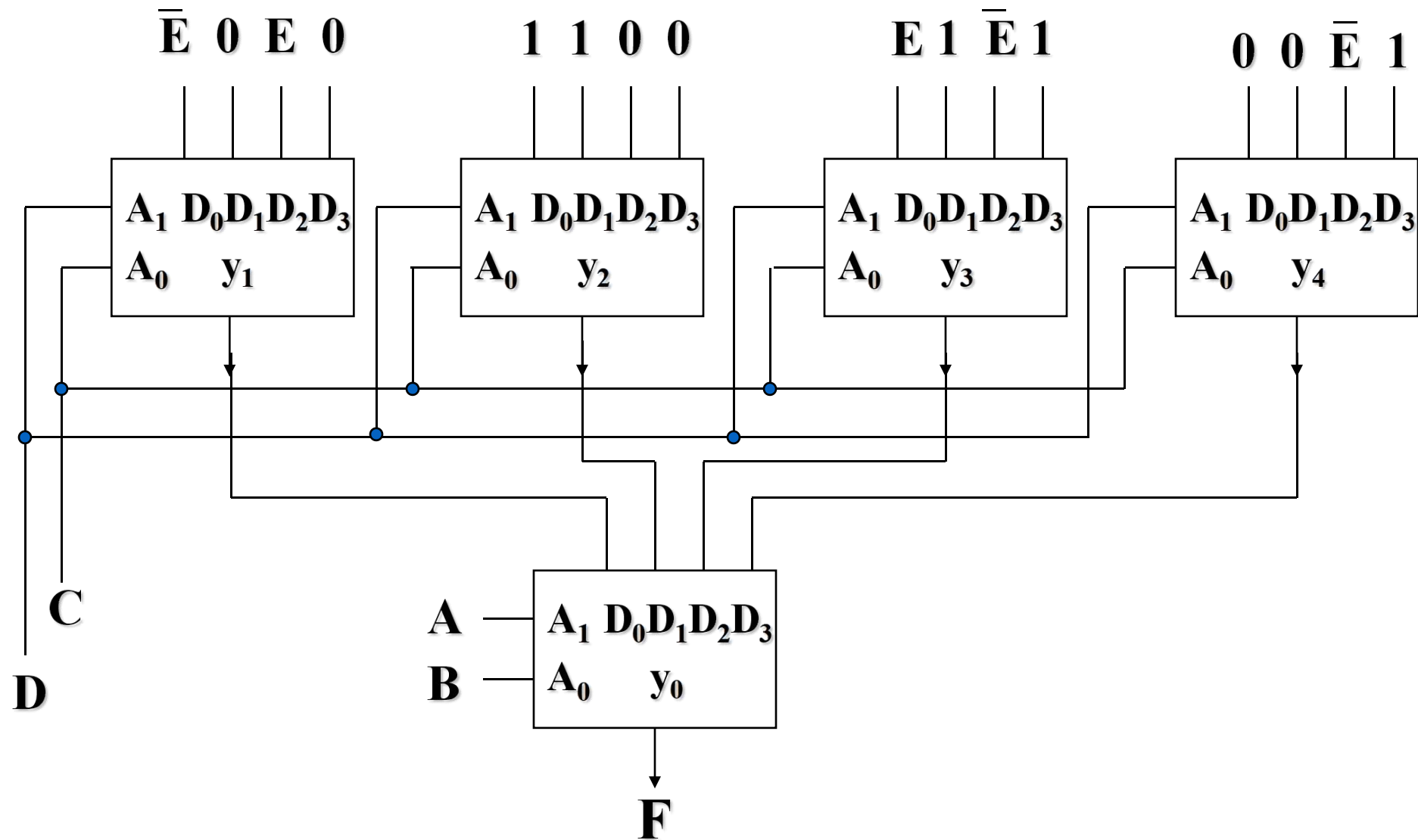
利用四选—多路选择器设计组合逻辑

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

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

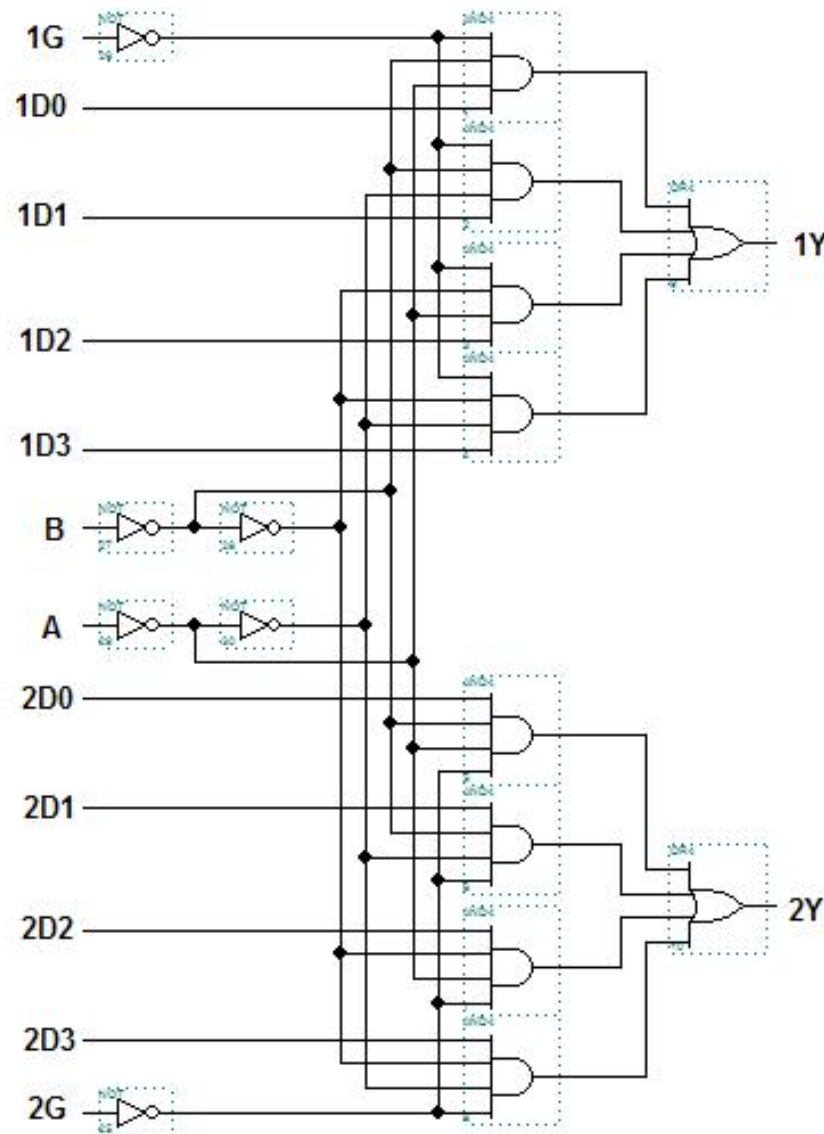
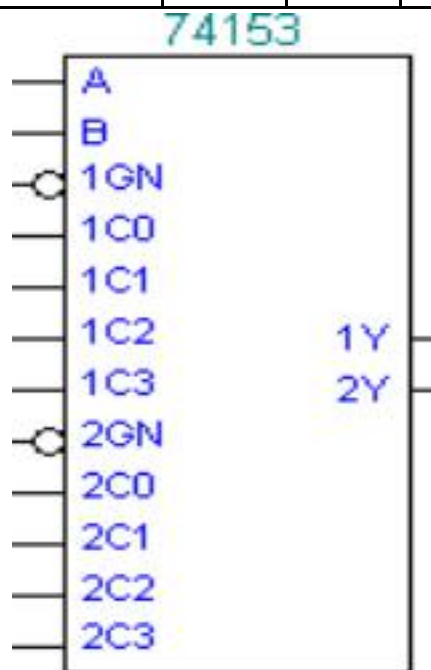
A B	C D	E	F	
10	0 0	0	0	E
		1	1	
	0 1	0	1	1
		1	1	
	1 0	0	1	\bar{E}
		1	0	
	1 1	0	1	1
		1	1	
11	0 0	0	0	0
		1	0	
	0 1	0	0	0
		1	0	
	1 0	0	1	\bar{E}
		1	0	
	1 1	0	1	1
		1	1	

利用四选—多路选择器设计组合逻辑



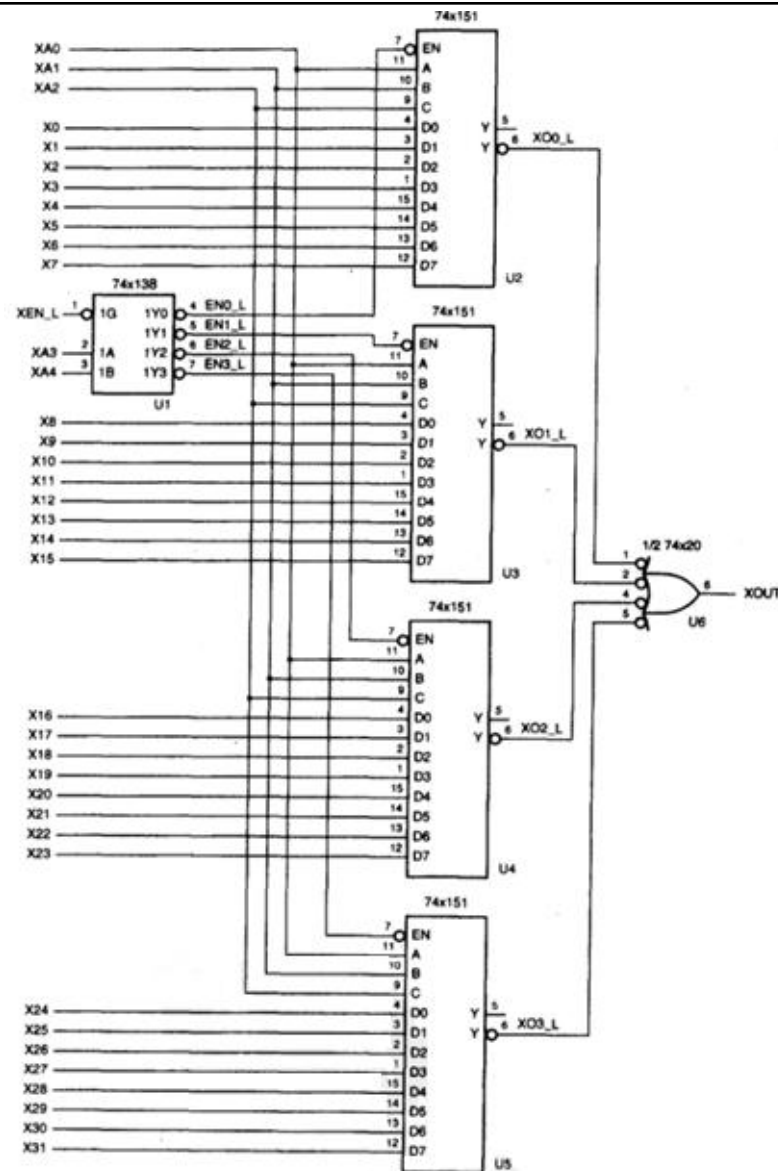
双4选1多路选择器典型器件74LS153

1Gn	2Gn	A	B	1Y	2Y
1	1	x	x	0	0
0	0	0	0	1C ₀	2C ₀
0	0	0	1	1C ₁	2C ₁
0	0	1	0	1C ₂	2C ₂
0	0	1	1	1C ₃	2C ₃



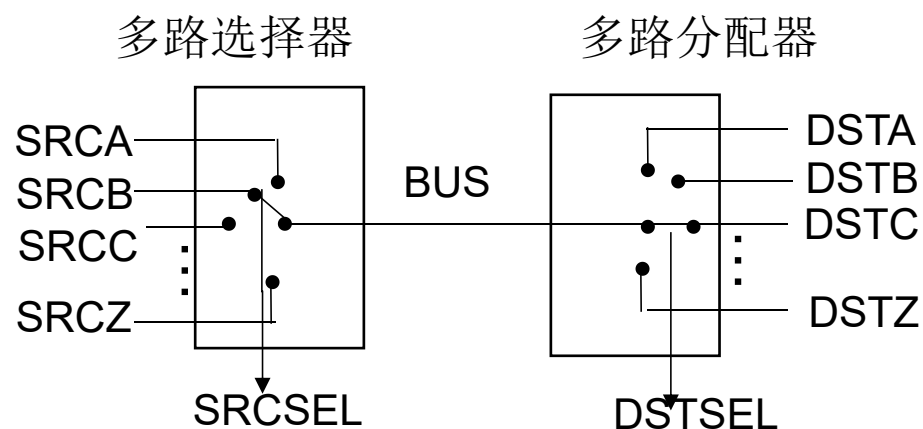
扩展多路选择器

- 给定8选1多路选择器74LS151
- 实现32选1多路选择器
- 用一个2-4译码器对2个最高选择位进行译码，以从4个74*151多路选择器中选择一个

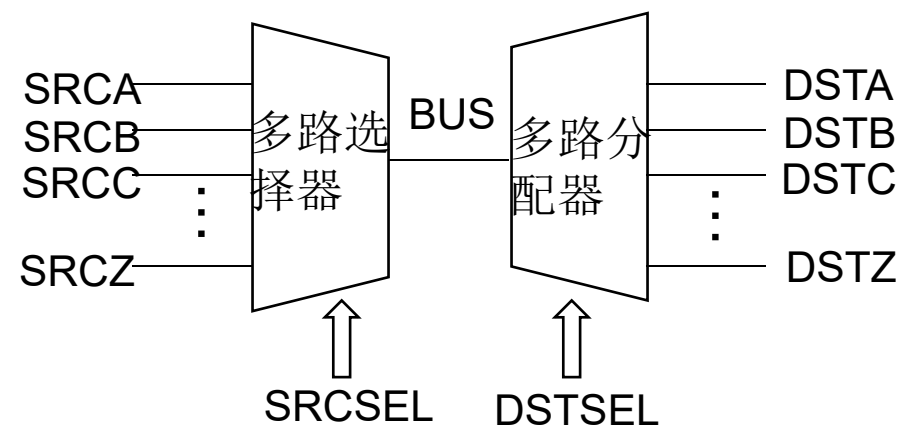


多路选择器、多路分配器和总线

- 多路选择器可以用于选择发往总线的n个数据源之一，即从多路信号中选择一路信号输出。
- 在总线的远端，多路分配器可以用于把总线数据送到m个目的地之一，即将总线数据传送到所选择的输出端口。



开关等效



符号框图

哪部分有疑问?

- ☐ A 译码器
- ☐ B 多路复用器
- ☐ C 利用译码器实现组合逻辑电路
- ☐ D 利用多路复用器实现组合逻辑电路
- ☐ E 无

提交

组合逻辑元件

- 只读存储器(ROM)
- 译码器(Decoders)
- 多路复用器(multiplexers)
- 三态器件(Three-state Buffer)
- 编码器(Encoders)
- 异或门和奇偶校验功能
- 比较器

三态缓冲器 three-state buffer

■ 又称三态驱动器

➤ 4种物理上不同的三态缓冲器的逻辑符号

三态缓冲器还可用来
增强输出驱动能力。

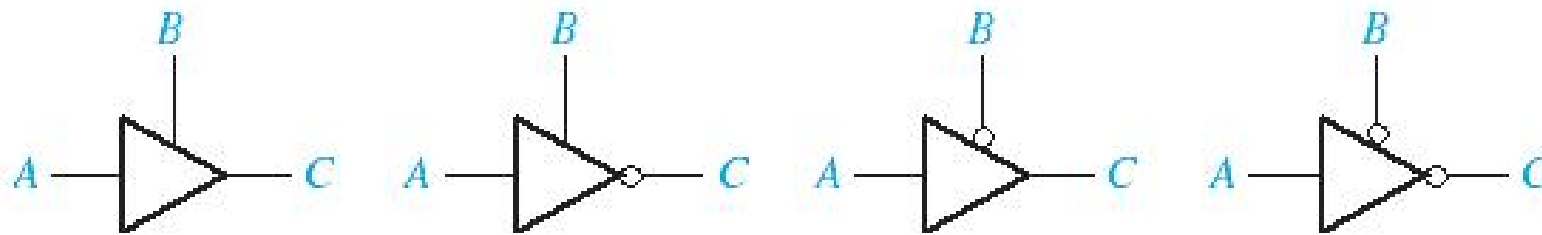
三态——

■ 0

■ 1

■ Z: 高阻态

电阻很大,
相当于开路



B	A	C
0	0	Z
0	1	Z
1	0	0
1	1	1

(a)

B	A	C
0	0	Z
0	1	Z
1	0	1
1	1	0

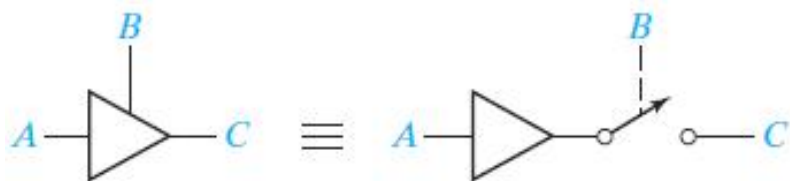
(b)

B	A	C
0	0	0
0	1	1
1	0	Z
1	1	Z

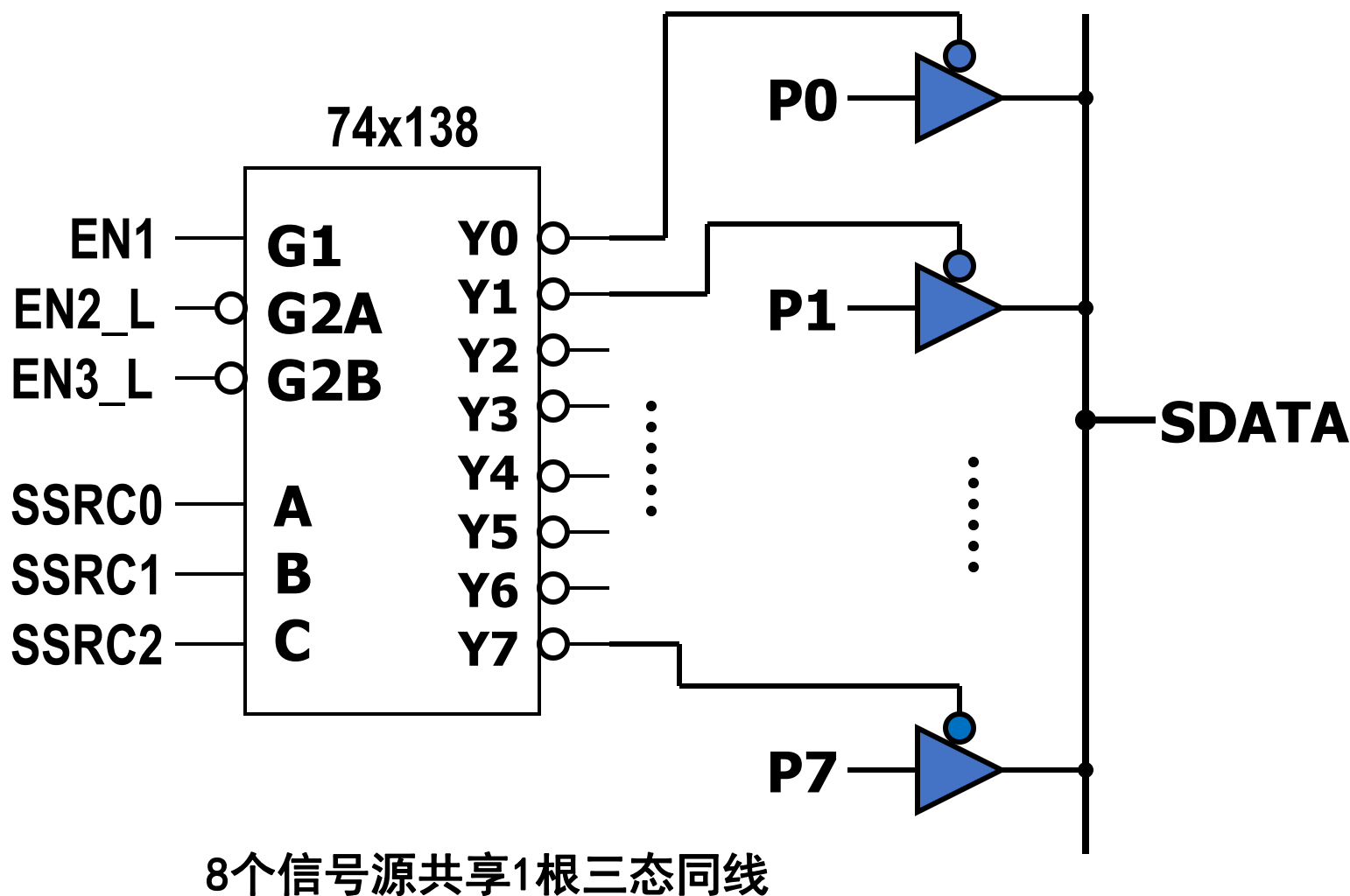
(c)

B	A	C
0	0	1
0	1	0
1	0	Z
1	1	Z

(d)



三态缓冲器 Three-state buffer

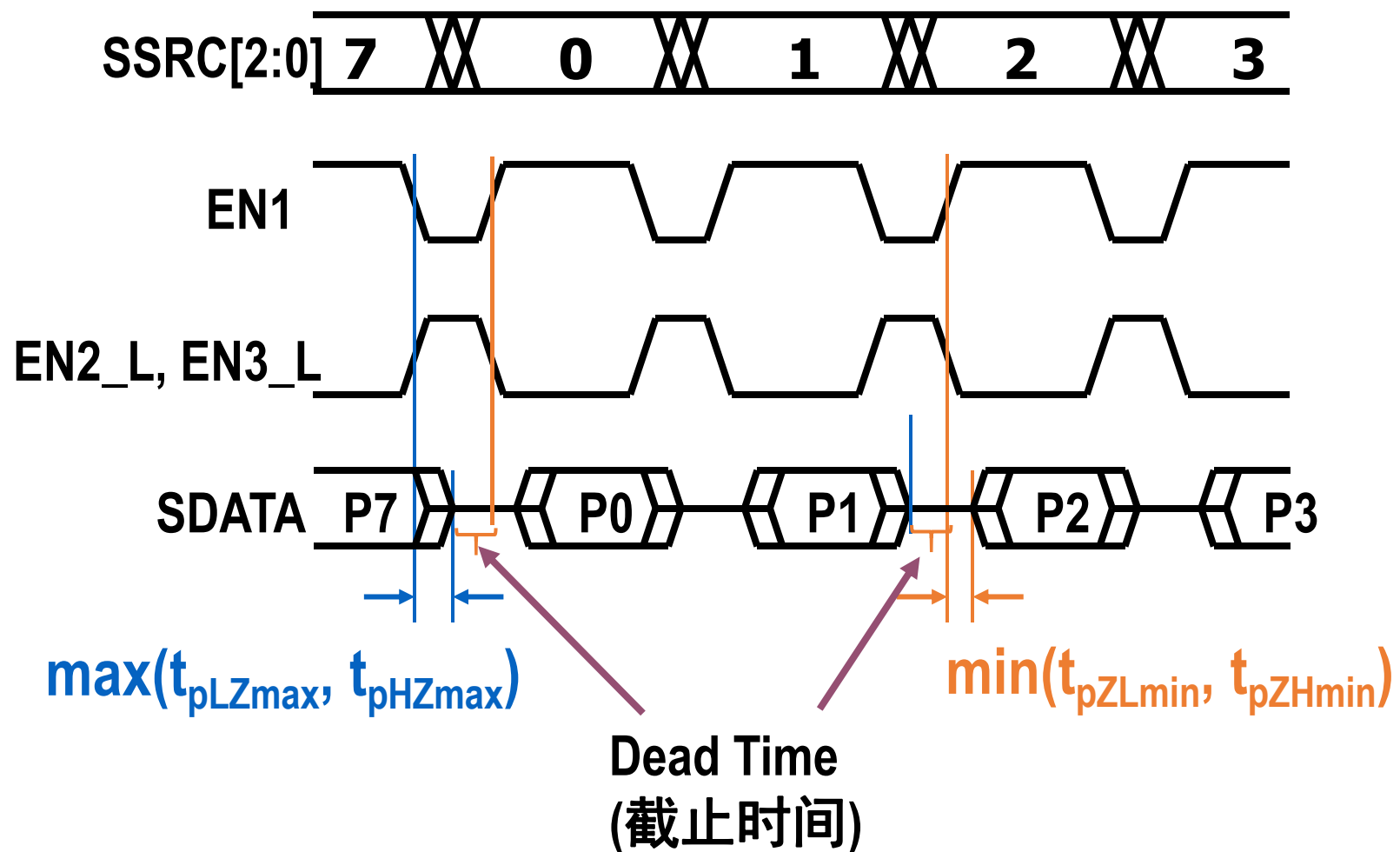


三态器件允许多个信号源共享单个“同线”，但线上每次仅一个器件“谈话”

假如不是全部EN线有效，则没有一个三态缓冲器能被使能，此时SDATA上的逻辑值是“未定义”，悬空信号的实际电压值依赖于电路细节。

一般进入高阻态比离开快可以避免冲突（两个三态器件同时驱动同一根线）

三态缓冲器——截止时间

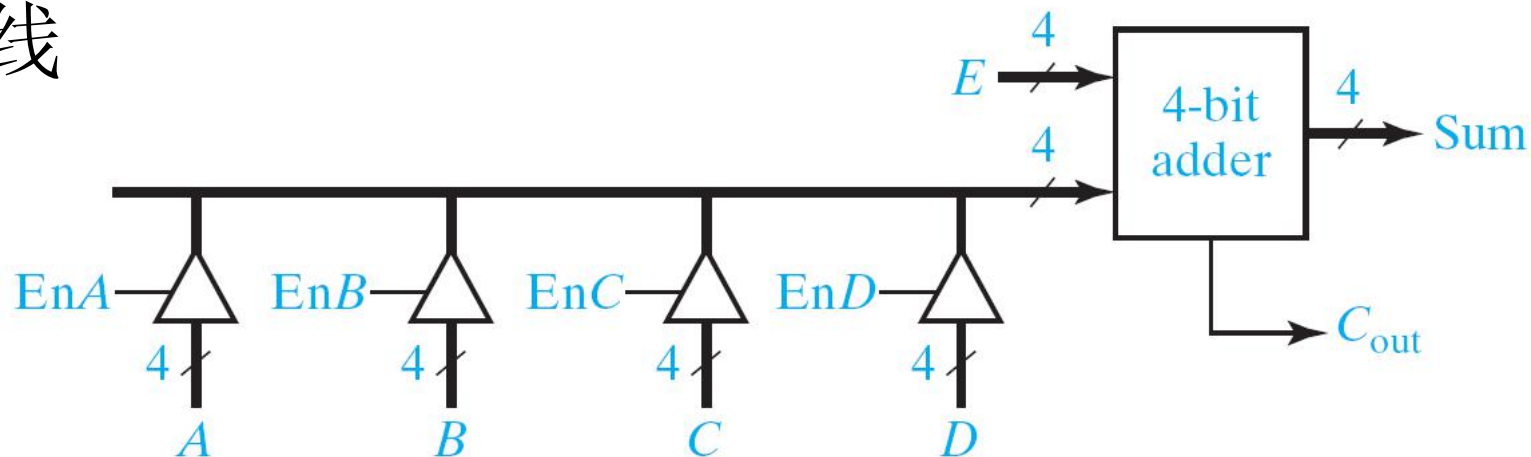


使用三态器件唯一真正安全的方法是：设计逻辑控制，以保证同线上有一段截止时间（dead time），在此期间不应有任何器件驱动同线。

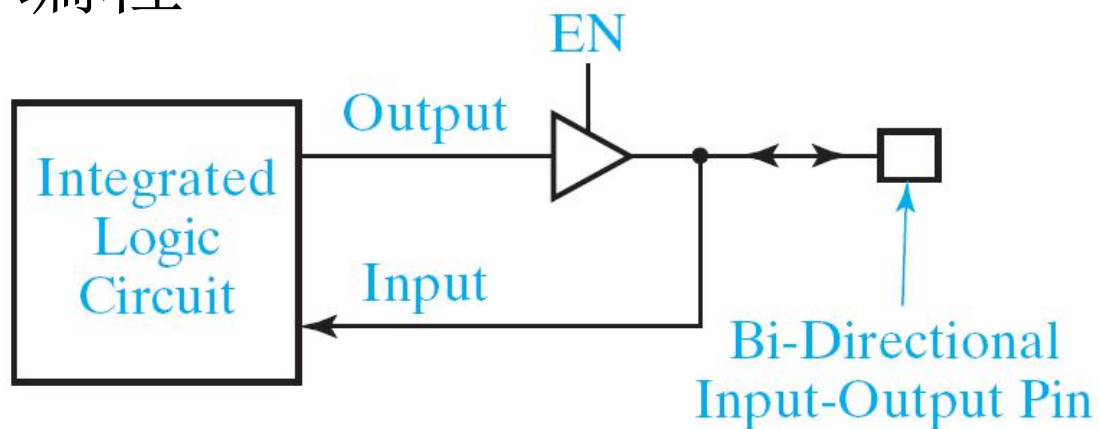
三态同线时序图

三态缓冲器的应用

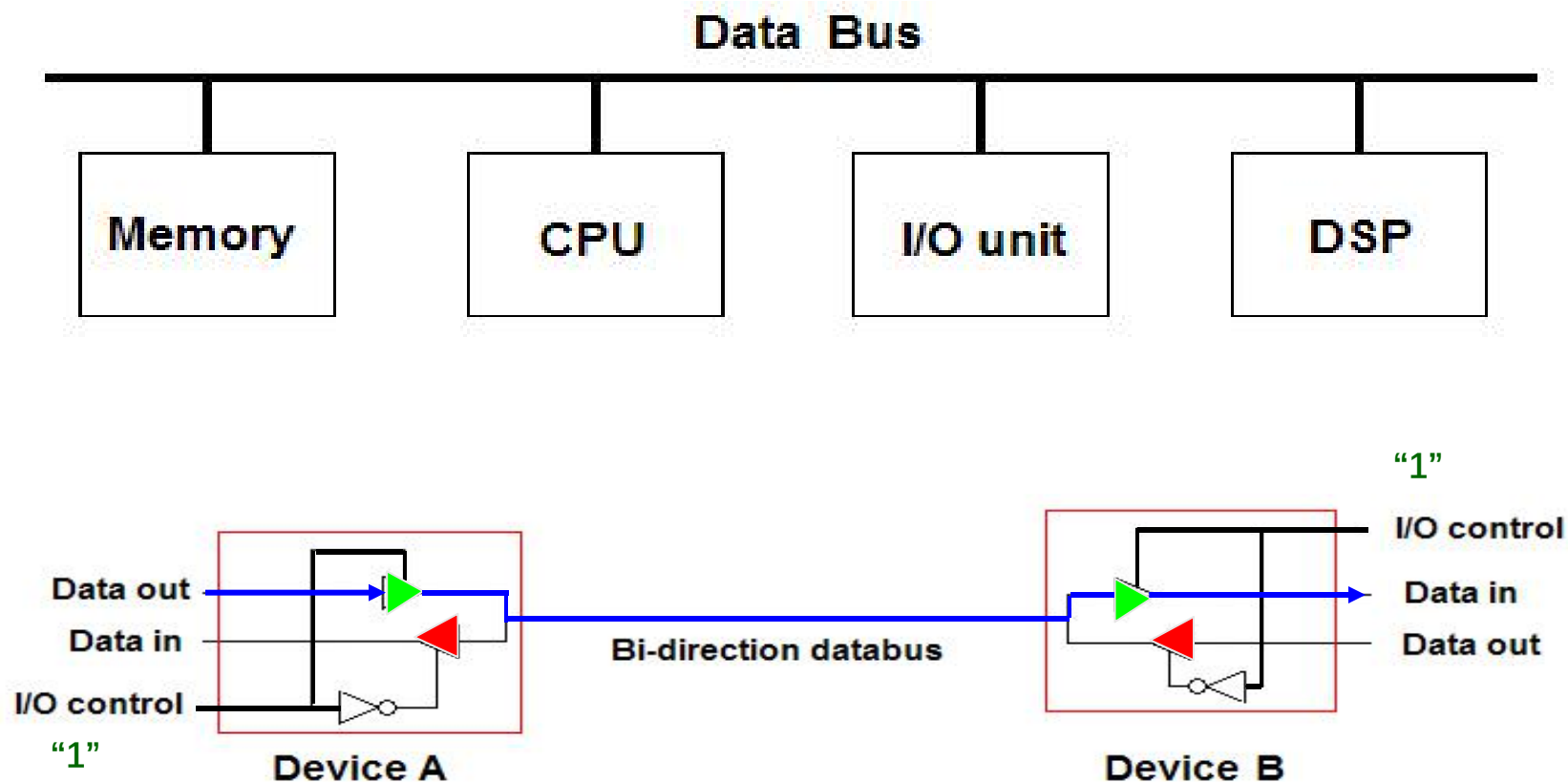
- 三态总线



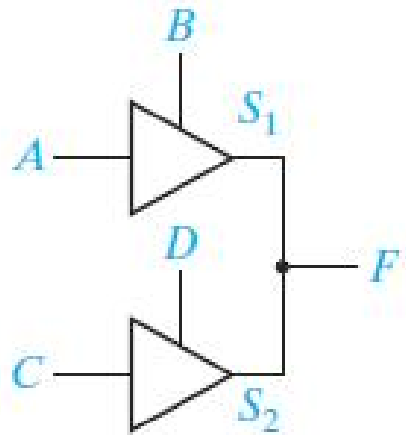
- 管脚输入输出可编程



三态缓冲器的应用——双向数据总线

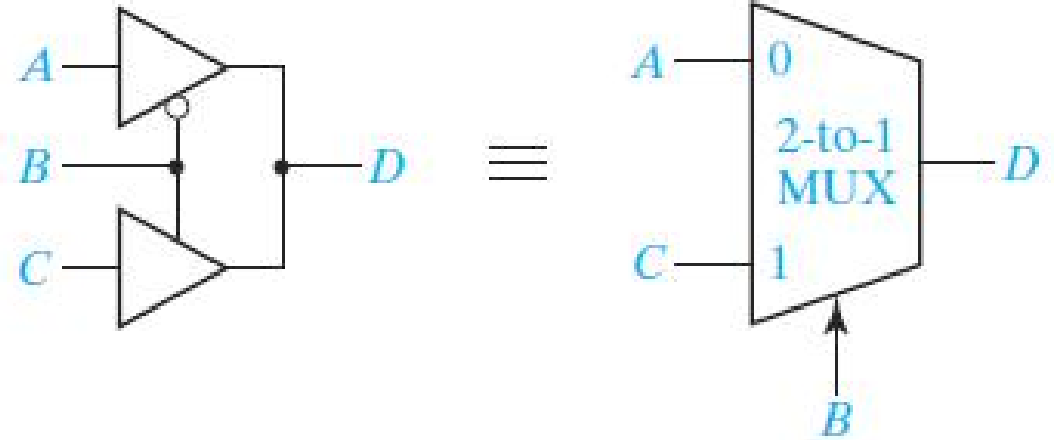


三态缓冲器的应用——续



S_1	X	S_2 0	1	Z
X	X	X	X	X
0	X	0	X	0
1	X	X	1	1
Z	X	0	1	Z

具有两个三态缓冲器的电路

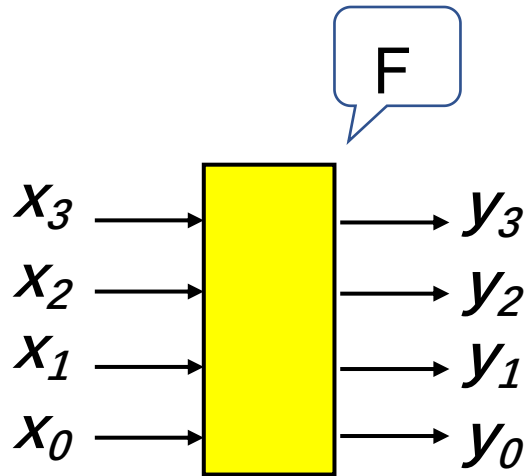


用三态缓冲器实现数据选择

三态缓冲器的应用——MOD 5选择电路

- $X=X_3X_2X_1X_0$ 为8421BCD码，设计一个MOD 5选择电路，要求选择那些能被5整除的数输出。

①真值表(F为控制信号)



$X_3 X_2 X_1 X_0$	F	$X_3 X_2 X_1 X_0$	F
0 0 0 0	1	1 0 0 0	0
0 0 0 1	0	1 0 0 1	0
0 0 1 0	0	1 0 1 0	×
0 0 1 1	0	1 0 1 1	×
0 1 0 0	0	1 1 0 0	×
0 1 0 1	1	1 1 0 1	×
0 1 1 0	0	1 1 1 0	×
0 1 1 1	0	1 1 1 1	×

MOD 5选择电路——续

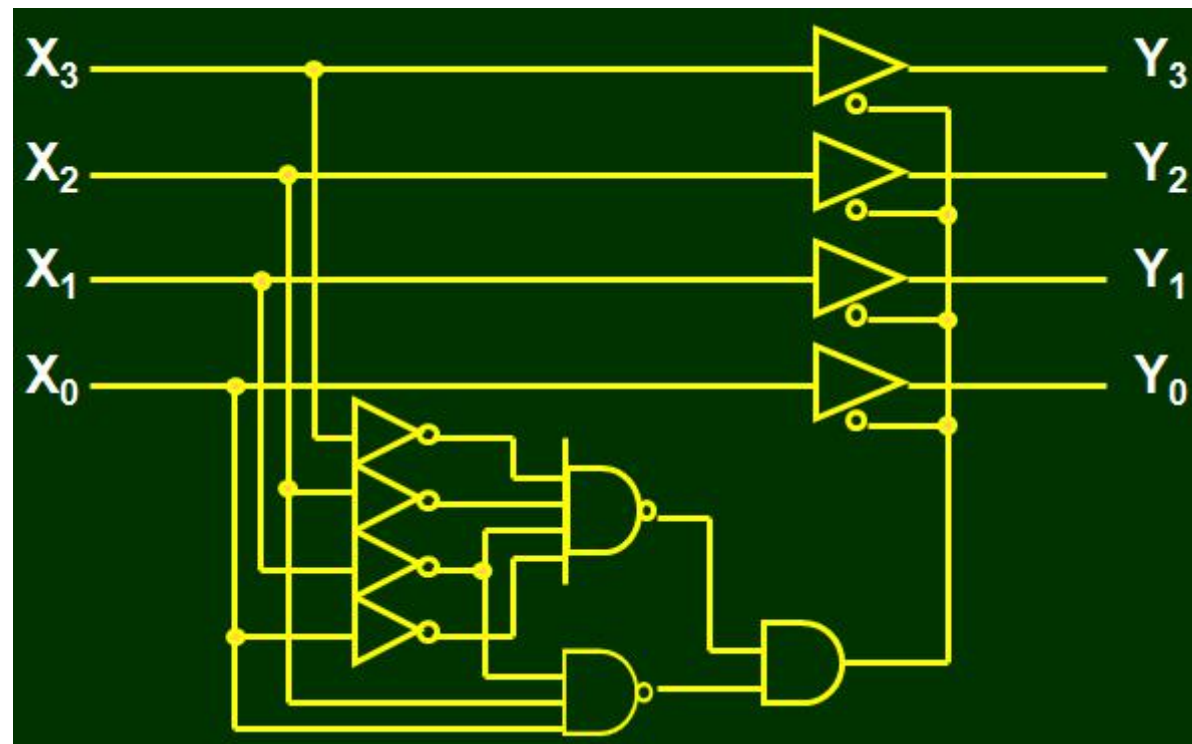
② 化简

x_1x_0					
x_3x_2	00	01	11	10	
00	1	0	0	0	
01	0	1	0	0	
11	x	x	x	x	
10	0	0	x	x	

$$\begin{aligned} F &= \overline{\overline{X_2 \bar{X}_1 X_0} + \bar{X}_3 \bar{X}_2 \bar{X}_1 \bar{X}_0} \\ &= \overline{(X_2 \bar{X}_1 X_0) (\bar{X}_3 \bar{X}_2 \bar{X}_1 \bar{X}_0)} \end{aligned}$$

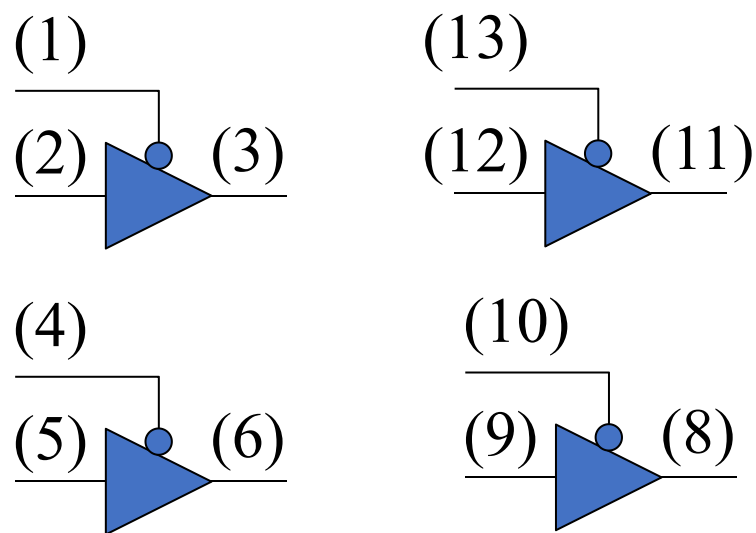
$$\bar{F} = \overline{(X_2 \bar{X}_1 X_0)} \overline{(\bar{X}_3 \bar{X}_2 \bar{X}_1 \bar{X}_0)}$$

③ 逻辑图



典型三态缓冲器——四线缓冲器74LS125

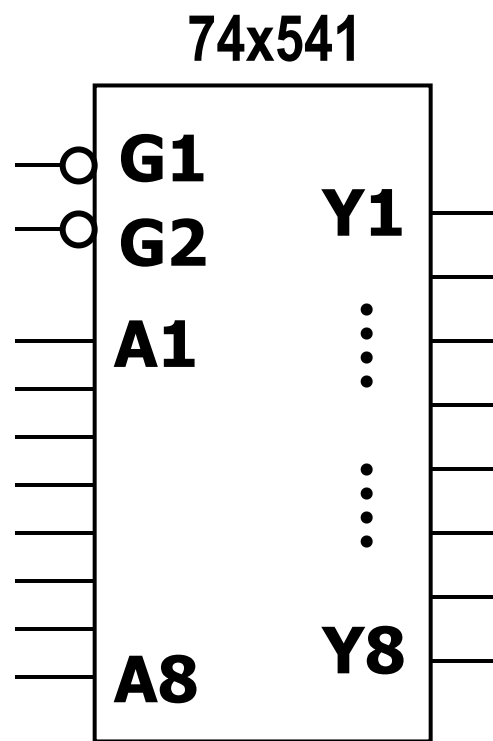
- 几个独立的三态缓冲器可以封装在单个IC中



74x125三态缓冲器的管脚引线

典型三态缓冲器——八缓冲器74x541

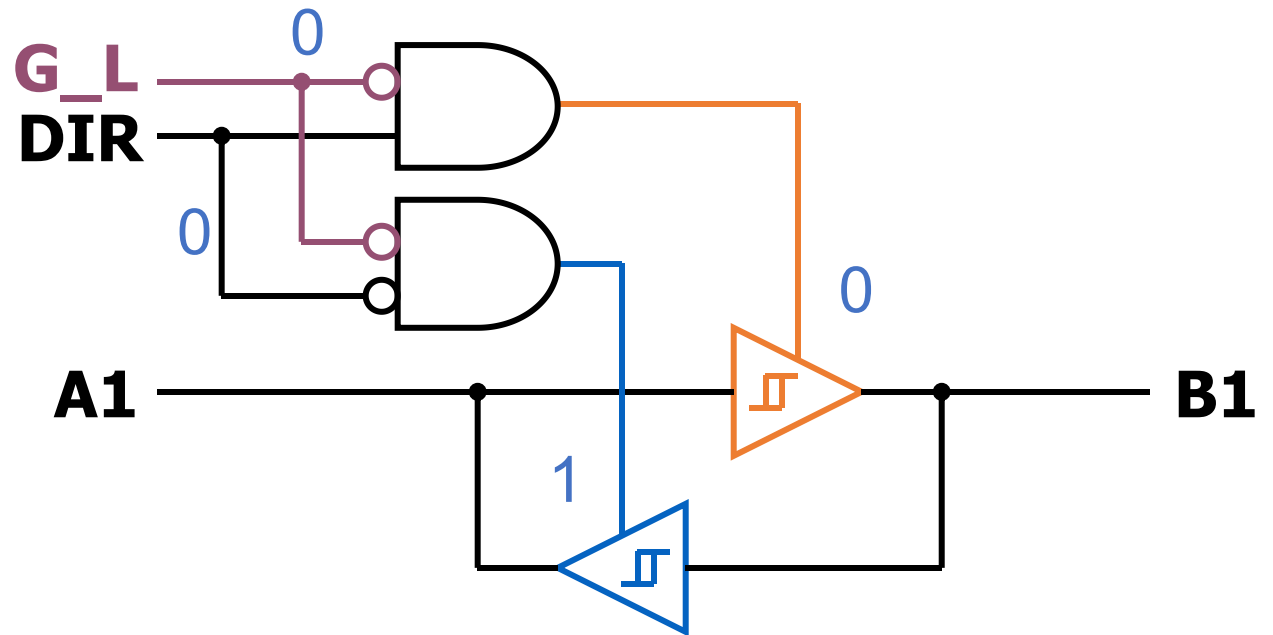
- 在宽总线应用中为了减少封装尺寸，多数常用的MSI部件包含带有公共使能输入的多个三态缓冲器。



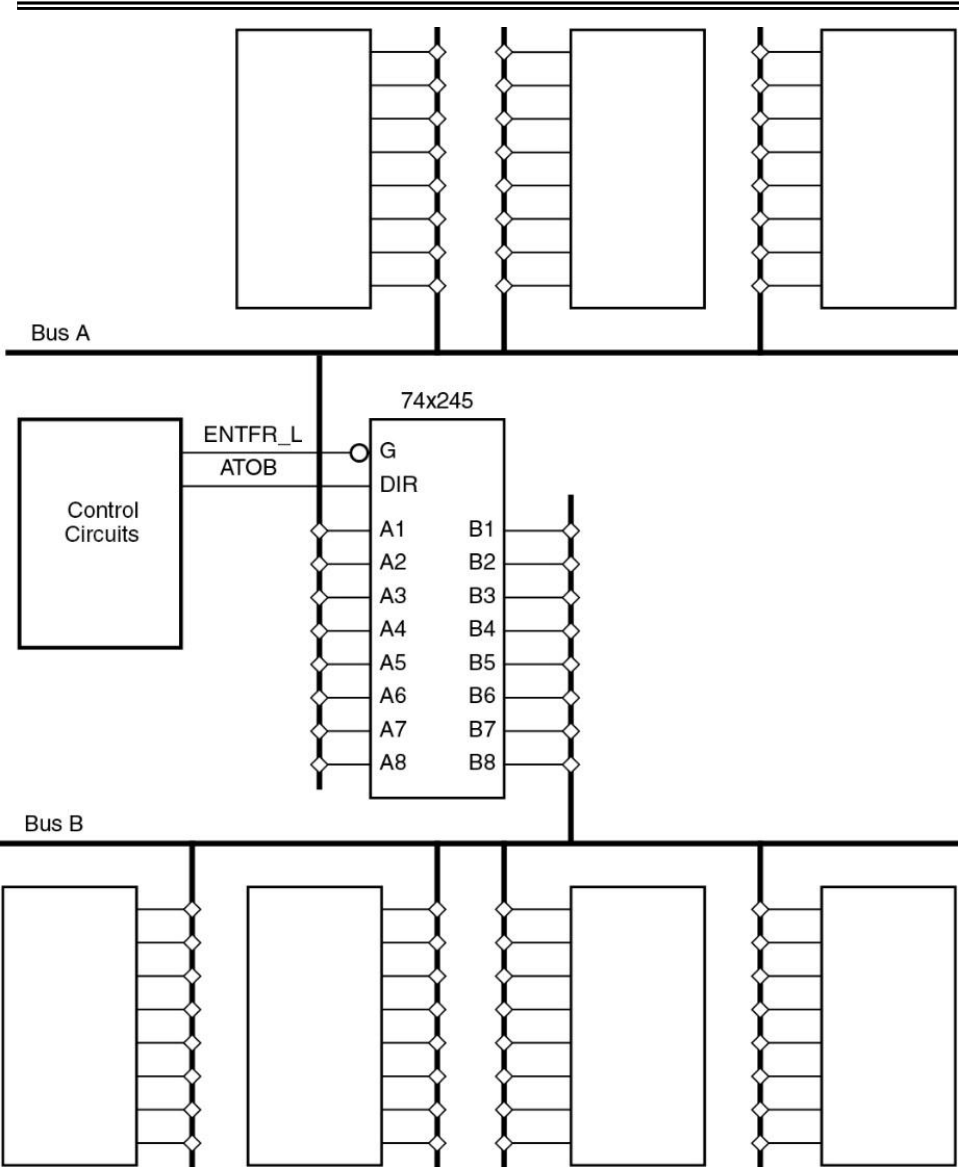
8三态缓冲器74x541

典型三态缓冲器应用——总线收发器

- 包含三态缓冲器对，每对引脚之间以相反方向连接，所以数据可以双向传输。
- 常用于两个双向总线之间，根据G_L和DIR的状态，控制操作模式。



双向总线示例



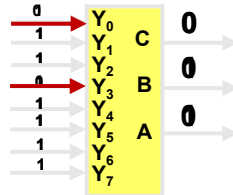
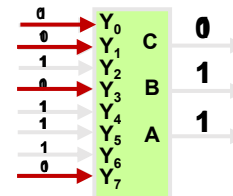
ENTRF_L	ATOB	操作
0	0	B——>A
0	1	A——>B
1	x	AB单独传输

组合逻辑元件

- 只读存储器 (ROM)
- 译码器 (Decoders)
- 多路复用器 (multiplexers)
- 三态器件 (Three-state Buffer)
- 编码器 (Encoders)
- 异或门和奇偶校验功能
- 比较器

编码器

- 特点：多输入、多输出的组合逻辑电路
- 功能：将二进制码按照一定规律编排，使其具有特定含义，与译码器互逆。

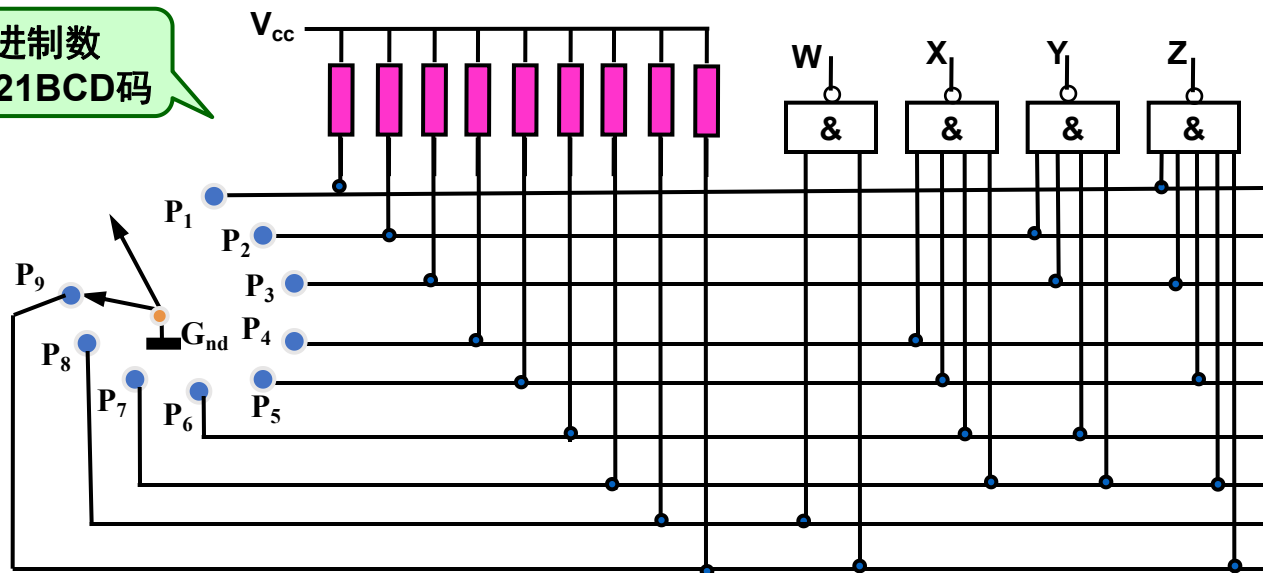
常用编码器	特点	编码演示
普通编码器 (二进制编码器)	N 位，任何时刻 N 根输入线中只能有一个输入有效， N ($N=2^n$) 中取一。 n 位二进制码	 <p>(8 线-3 线编码器)</p>
优先编码器	<ul style="list-style-type: none">• 允许同时输入两个以上有效输入信号• 能按照预先设定的优先级别，只对其优先级最高的输入进行编码。	 <p>(8 线-3 线优先编码器)</p>

键盘编码器

键盘编码器功能表

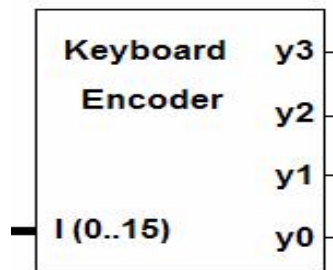
$P_9 \dots P_1$	按键	WXYZ
111111111	0	0000
111111110	1	0001
111111101	2	0010
111111011	3	0011
111110111	4	0100
111101111	5	0101
111011111	6	0110
110111111	7	0111
101111111	8	1000
011111111	9	1001

输入：十进制数
输出：8421BCD码



扩展

7	8	9	/
4	5	6	*
1	2	3	-
0	.	+	=



0	I0	0000	8	I8	1000
1	I1	0001	9	I9	1001
2	I2	0010	/	I10	1010
3	I3	0011	*	I11	1011
4	I4	0100	-	I12	1100
5	I5	0101	+	I13	1101
6	I6	0110	.	I14	1110
7	I7	0111	=	I15	1111

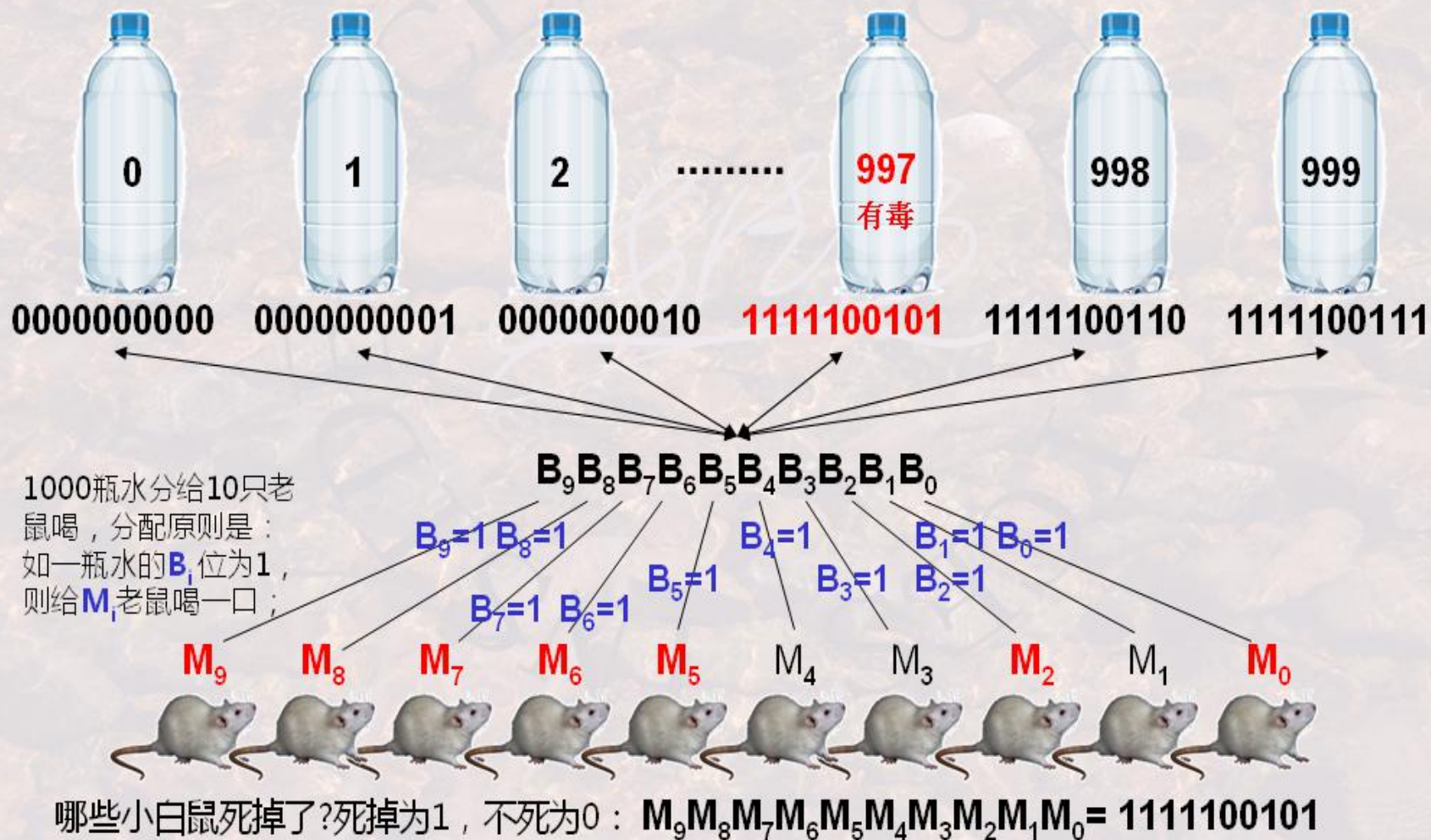
$$W=(P_8 \cdot P_9)'$$

$$Y=(P_2 \cdot P_3 \cdot P_6 \cdot P_7)'$$

$$X=(P_4 \cdot P_5 \cdot P_6 \cdot P_7)'$$

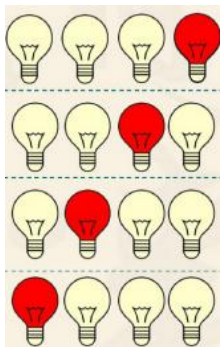
$$Z=(P_1 \cdot P_3 \cdot P_5 \cdot P_7 \cdot P_9)'$$

有1000瓶水，其中有一瓶有毒，小白鼠只要尝一点带毒的水24小时内就会死亡，至少要多少只小白鼠才能在24小时内鉴别出那瓶水有毒。



优先编码器

抢答器



4:2编码器

A_3	A_2	A_1	A_0	B_1	B_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

计算机配有四个外部设备：声卡(A0)，硬盘驱动器(A1)，鼠标(A2)，网卡(A3)， B_0 、 B_1 为编码输出。

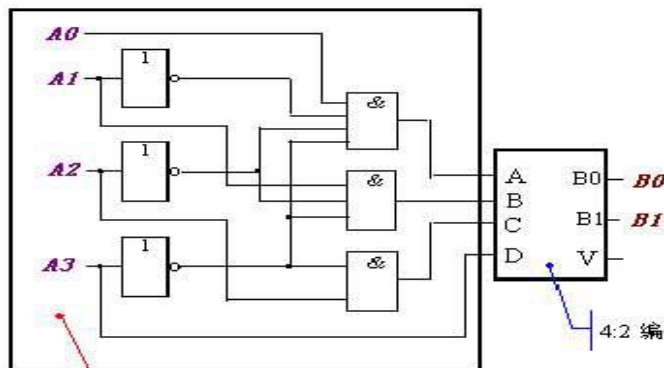


某一时刻只允许输入一个编码信号，如 A_1 ($A_1=1$) 向 CPU 请求传送数据，CPU 根据接收的编码 $B_1B_0=01$ ，启动硬盘驱动器，开始传送数据。

4:2优先编码器

A_3	A_2	A_1	A_0	B_1	B_0
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

普通编码器：无法避免错误输入（同时输入多路有效信号），容易造成混乱。



优先排队电路

优先编码器

$$A = A_0 \overline{A_1} \overline{A_2} \overline{A_3}$$

$$B = A_1 \overline{A_2} \overline{A_3}$$

$$C = A_2 \overline{A_3}$$

$$D = A_3$$

二进制编码器：

- 可以对 2^n 个输入对象编码
- 只需 n 个输出端（每个对象获得一个 n 位编码）
- 编码具有唯一性

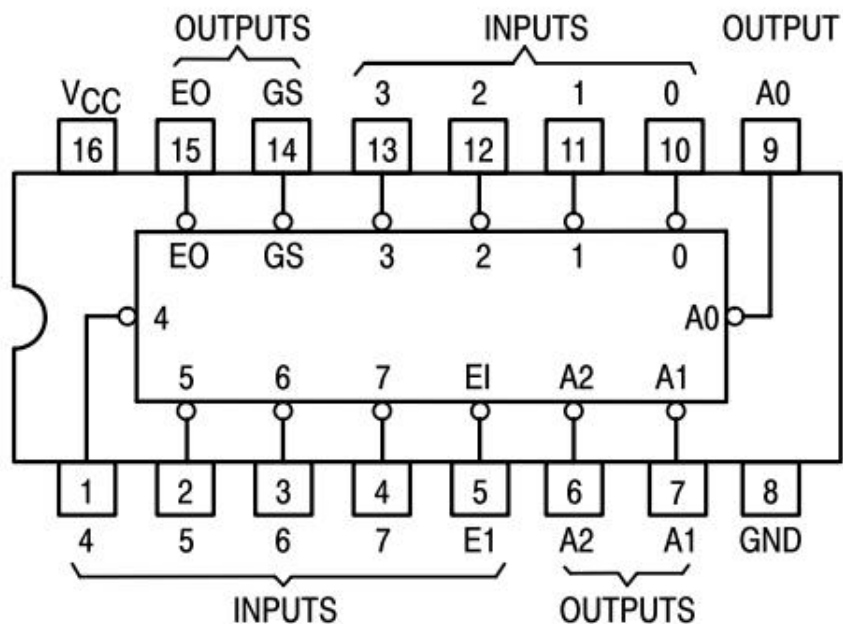
优先编码器：

- 允许同时输入多路有效信号
- 按照预先设定的优先级，只对其中优先级最高的输入进行编码。

编码器典型芯片74LS148

输入和输出均为低电平有效

SN54/74LS148
SN54/74LS748
(TOP VIEW)



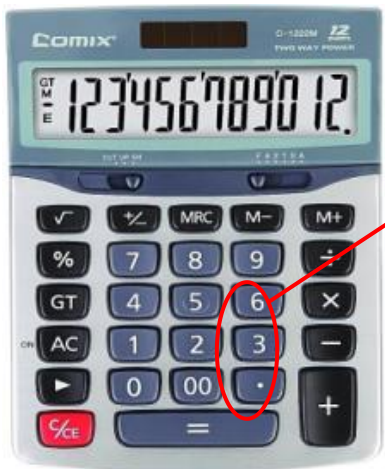
8线-3线优先编码器

输 入									输 出				
\overline{S}	\overline{I}_7	\overline{I}_6	\overline{I}_5	\overline{I}_4	\overline{I}_3	\overline{I}_2	\overline{I}_1	\overline{I}_0	\overline{Y}_2	\overline{Y}_1	\overline{Y}_0	\overline{Y}_{EX}	Y_S
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	×	×	×	×	×	×	×	0	0	0	0	1
0	1	0	×	×	×	×	×	×	0	0	1	0	1
0	1	1	0	×	×	×	×	×	0	1	0	0	1
0	1	1	1	0	×	×	×	×	0	1	1	0	1
0	1	1	1	1	0	×	×	×	1	0	0	0	1
0	1	1	1	1	1	0	×	×	1	0	1	0	1
0	1	1	1	1	1	1	0	×	1	1	0	0	1
0	1	1	1	1	1	1	1	0	1	1	1	0	1

标志位
0: 编码输出;
1: 非编码输出

输出使能

编码器与译码器的实际应用



键盘编码器

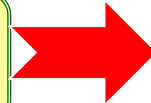


存储、运算等
处理....

计算结果
1001



显示译码器



显示器件



哪部分有疑问?

- A ROM实现组合逻辑函数
- B 译码器
- C 多路复用器/多路开关
- D 三态缓冲器
- E 编码器
- F 优先编码器
- G 无

提交

组合逻辑元件

- 只读存储器(**ROM**)
- 译码器(**Decoders**)
- 多路复用器(**multiplexers**)
- 三态器件(**Three-state Buffer**)
- 编码器(**Encoders**)
- 异或门和奇偶校验功能
- 比较器

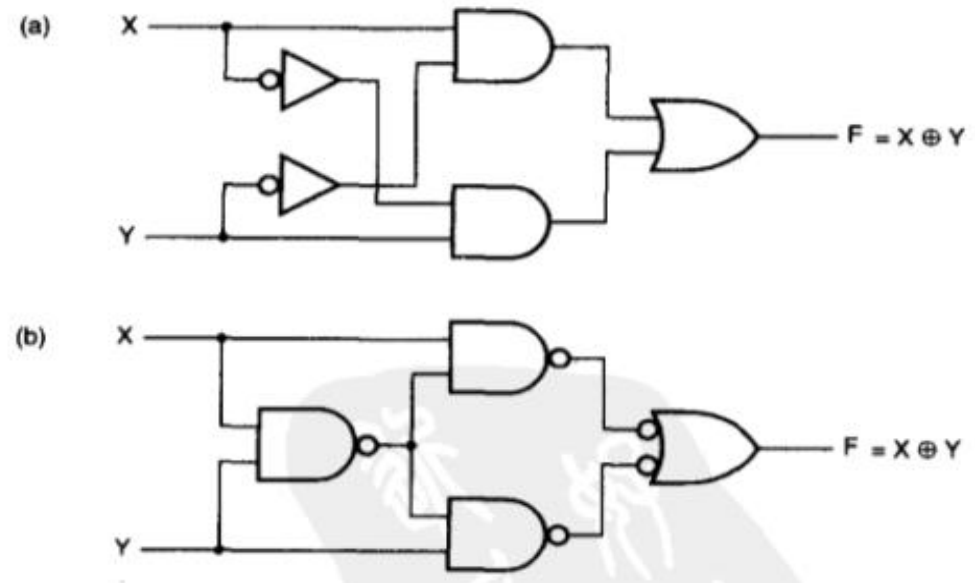
异或门和异或非门

异或门是二输入门，如果两个输入不同，则输出为1。

异或非门正好相反，如果两个输入相同，则输出为1。

X	Y	$X \oplus Y$ (XOR)	$(X \oplus Y)'$ (XNOR)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

异或和异或非功能的真值表

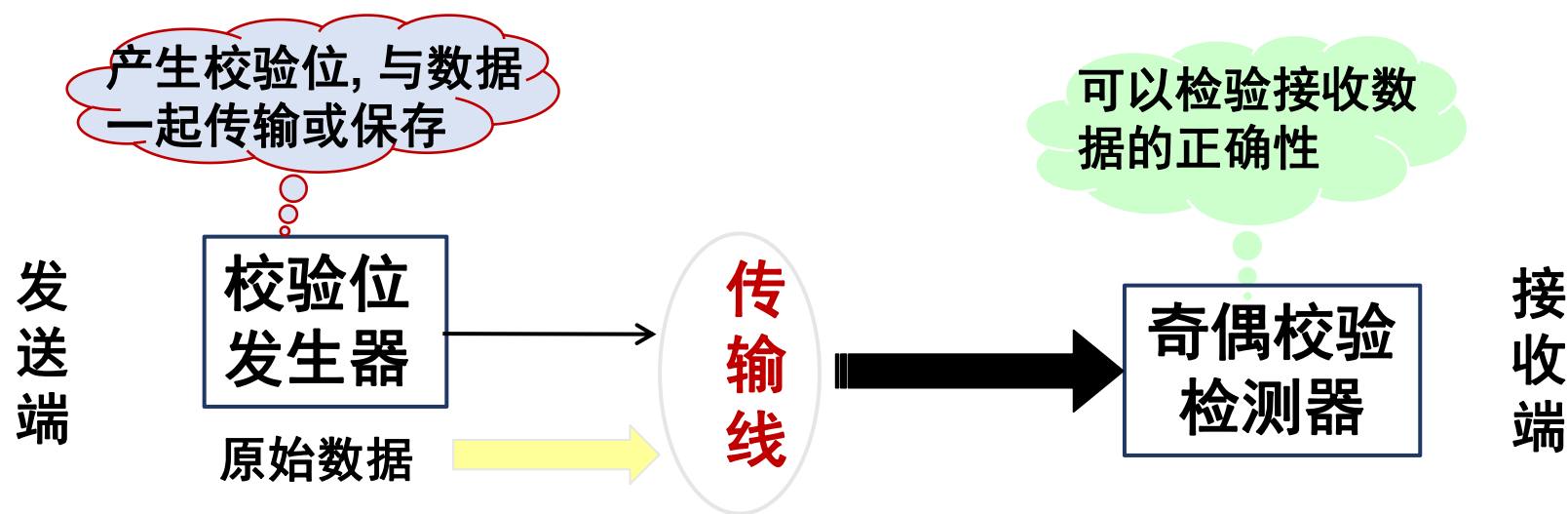


2输入异或函数的多门设计

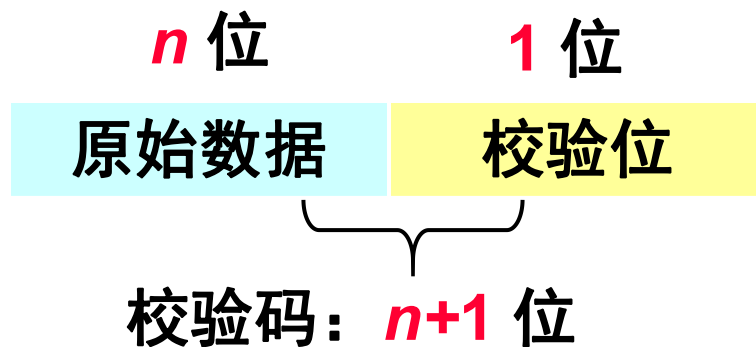
(a)与-或门 (b)三级与非门

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。
 - 发生一位错误的可能性一般占96%以上；
- 就是检测数据中包含“1”的个数是奇数还是偶数。
- 能够检测传送出错，但不能确定错误位置，不能纠错。
- 电路简单，容易实现。
- 广泛用于计算机的内存储器以及磁盘等外部设备中。



校验位计算方法



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:

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

异或门真值表

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

奇偶校验器一般
由异或门构成

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。

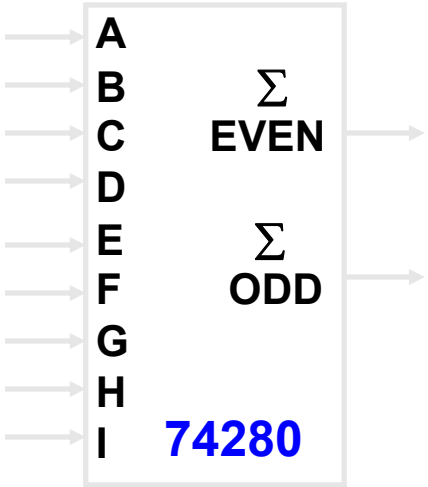
4位二进制数校验码真值表

$A_3 A_2 A_1 A_0$	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74LS280

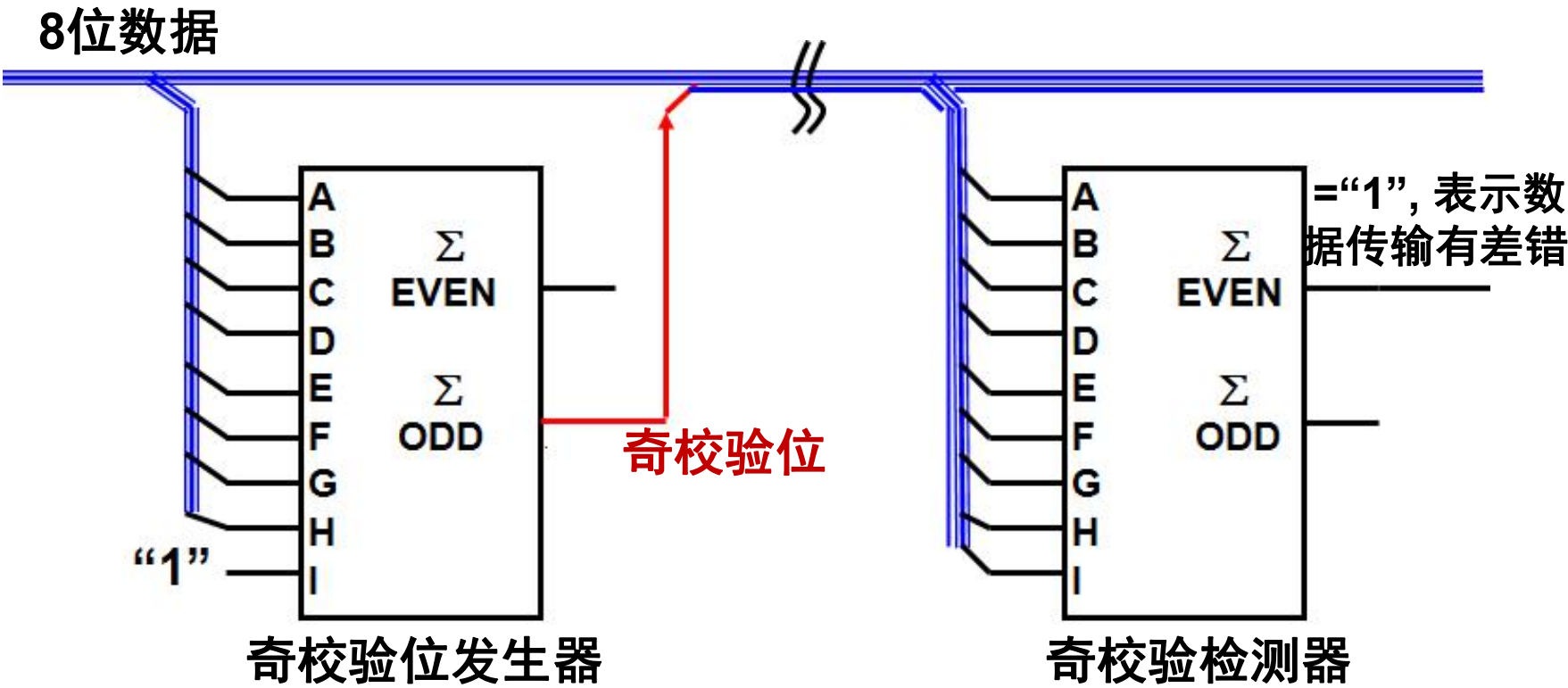
奇偶校验器/产生器：
74xx180/74xx280

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奇校验检测器。



74XX280功能表

A~I	EVEN	ODD
偶数个“1”	1	0
奇数个“1”	0	1



组合逻辑元件

- 只读存储器(ROM)
- 译码器(Decoders)
- 多路复用器(multiplexers)
- 三态器件(Three-state Buffer)
- 编码器(Encoders)
- 异或门和奇偶校验功能
- 比较器

数值比较器

- 计算机中对数据的基本处理方法

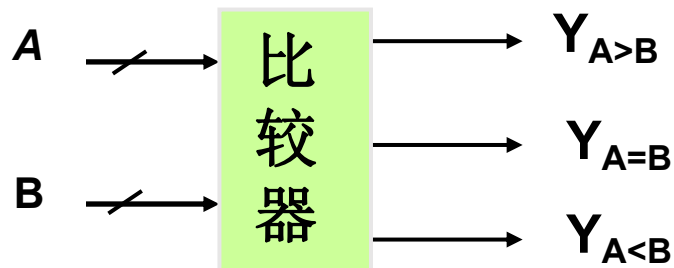
- 加、减、乘、除

- 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： $Y_{A>B}$ 、 $Y_{A<B}$ 、 $Y_{A=B}$

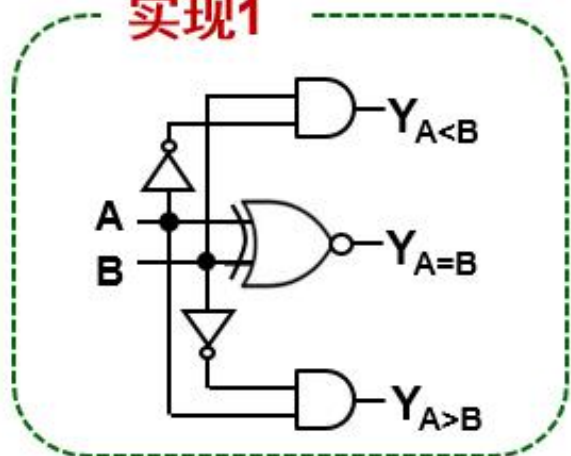


一位数值比较器

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

实现1

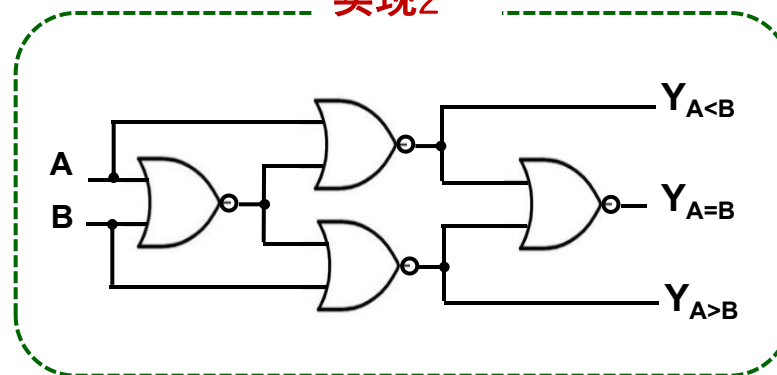


$$Y_{A=B} = A \odot B$$

$$Y_{A>B} = A\bar{B}$$

$$Y_{A<B} = \bar{A}B$$

实现2



$$Y_{A=B} = \bar{A}\bar{B} + AB = (A + \bar{B})(\bar{A} + B) = \overline{(A + \bar{A} + B)(B + \bar{A} + B)}$$

$$= \overline{(A + \bar{A} + B) + (B + \bar{A} + B)}$$

$$Y_{A>B} = A\bar{B} = \overline{\overline{A\bar{B}}} = \overline{\bar{A} + B}$$

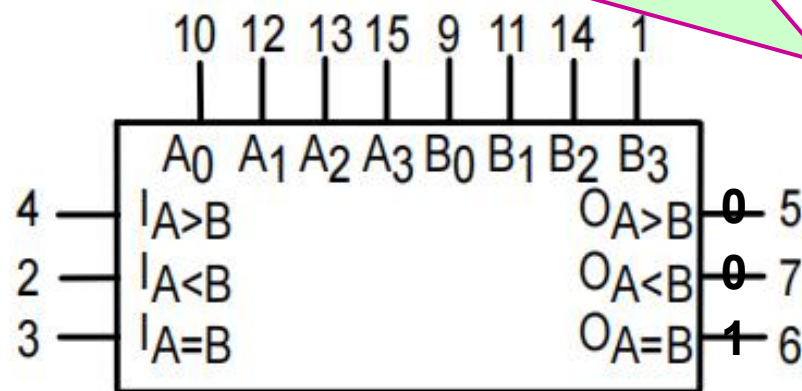
$$Y_{A<B} = \bar{A}B = \overline{\overline{\bar{A}B}} = \overline{A + \bar{B}}$$

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。

四位数值比较器74LS85

比较2个4位二进制数的大小时，3个输入端 $I_{A>B}$ 、 $I_{A<B}$ 、 $I_{A=B}$ 应接001；当 $A_3A_2A_1A_0=B_3B_2B_1B_0$ 时，比较器的输出 $Y_{A>B}Y_{A<B}Y_{A=B}=001$

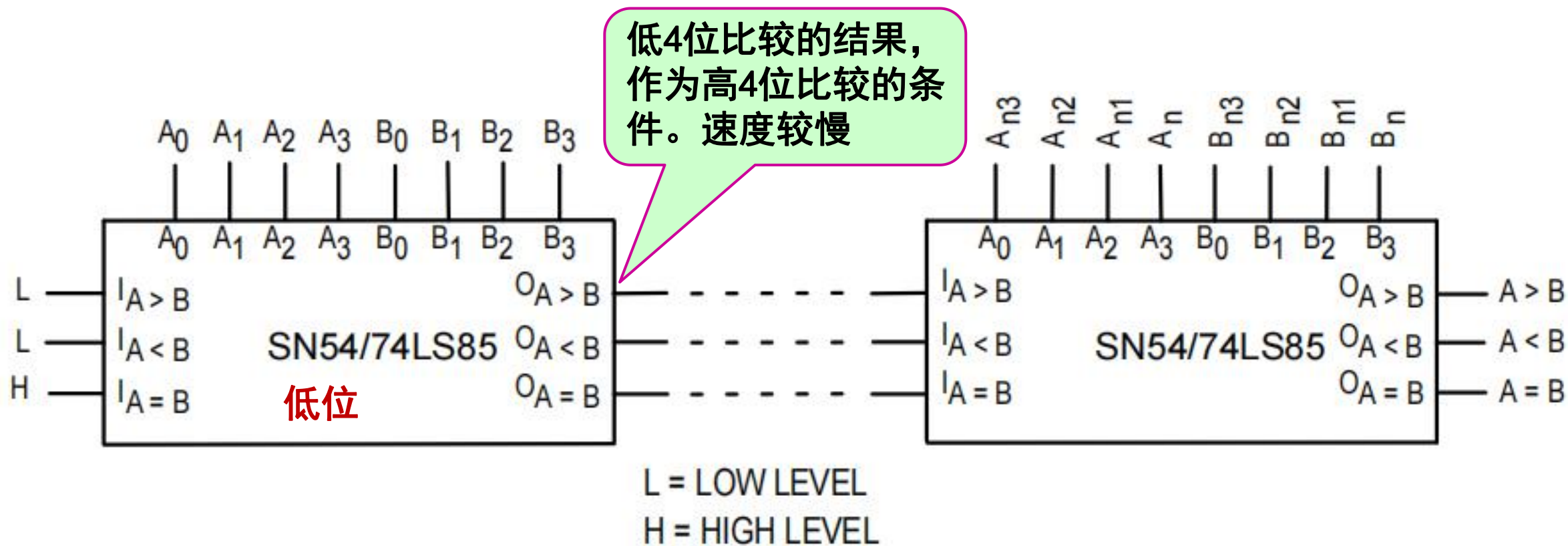


当 $A_3A_2A_1A_0=B_3B_2B_1B_0$ 时，比较器的输出复现3个输入端 $I_{A>B}$ 、 $I_{A<B}$ 、 $I_{A=B}$ 的状态。

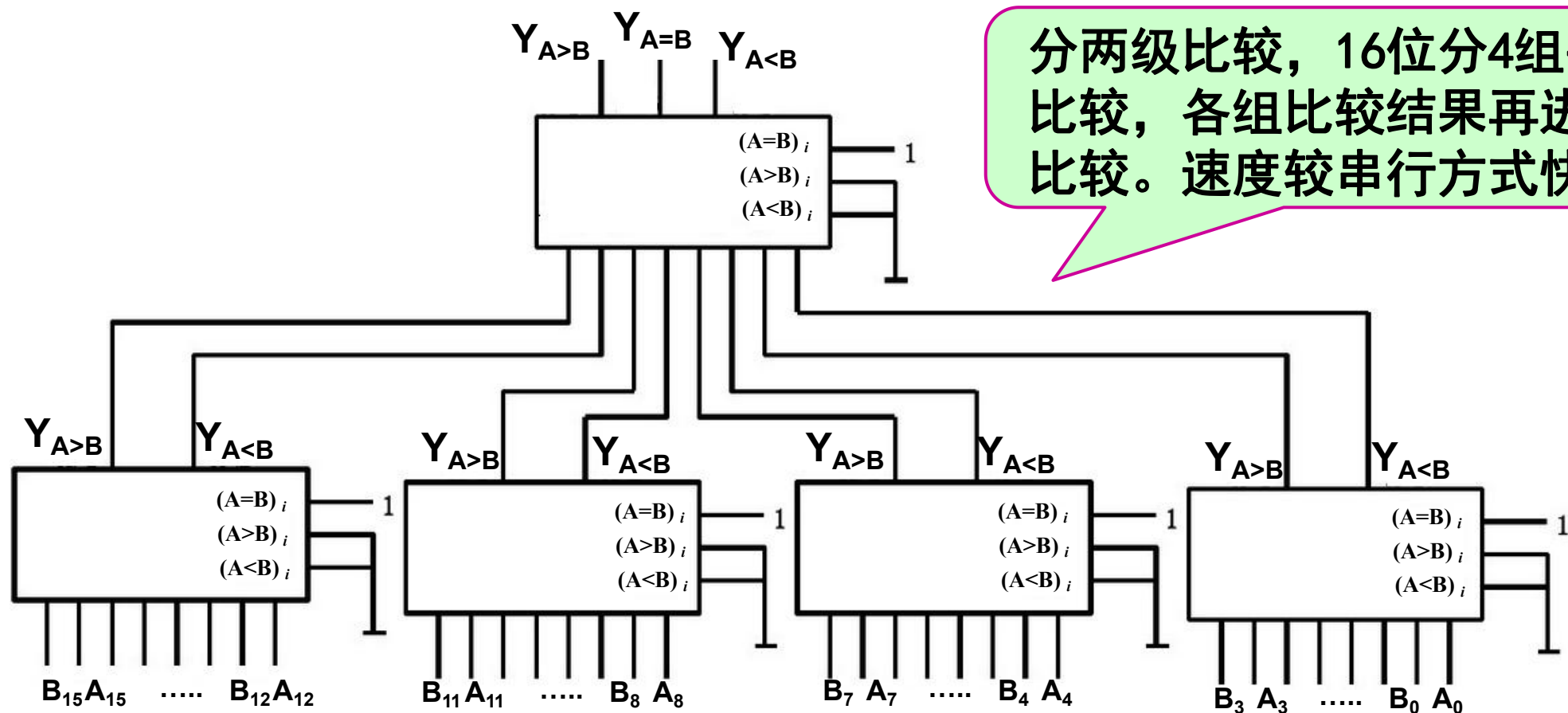
接低位芯片的比较结果，用于芯片扩展。

比较输入				级联输入I			输出O						
A ₃	B ₃	A ₂	B ₂	A ₁	B ₁	A ₀	B ₀	(A>B)	(A<B)	(A=B)	O _{A>B}	O _{A<B}	O _{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					X	X	1	0	0	1
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					1	0	0	1	0	0
A ₃ = B ₃	A ₂ = B ₂	A ₁ = B ₁	A ₀ = B ₀					1	1	0	0	0	0
A ₃ = B ₃	A ₂ = B ₂	A ₁ = B ₁	A ₀ = B ₀					0	0	0	1	1	0

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



哪部分有疑问?

- | | |
|---|-------|
| A | 异或门 |
| B | 奇偶校验器 |
| C | 一位比较器 |
| D | 多位比较器 |
| E | 无 |

提交

组合逻辑元件

- 只读存储器(ROM)
- 译码器(Decoders)
- 多路复用器(multiplexers)
- 三态器件(Three-state Buffer)
- 编码器(Encoders)
- 异或门和奇偶校验功能
- 比较器