数字逻辑电路

MSI组合逻辑器件及其应用

西安交通大学

电子物理与器件教育部重点实验室

等离子体与微波电子学研究所

张小宁

MSI组合逻辑器件及其应用

1. 译码器和编码器

- 译码器和编码器的一般结构、二进制译码器、MSI器件及其级联与应用
- 编码器、优先权编码器、MSI编码器及其级联和应用

2. 数据分配器和多路选择器

- 数据分配器原理和MSI数据多路选择器
- 多路选择器、MSI多路选择器及其扩展和应用

3. 三态门

- 三态门、多路收发器和双向收发器

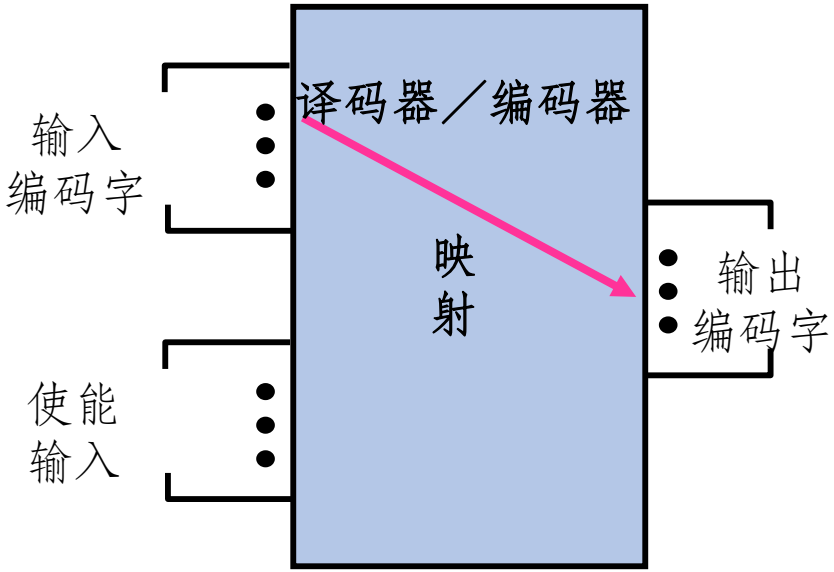
4. 加法器和比较器

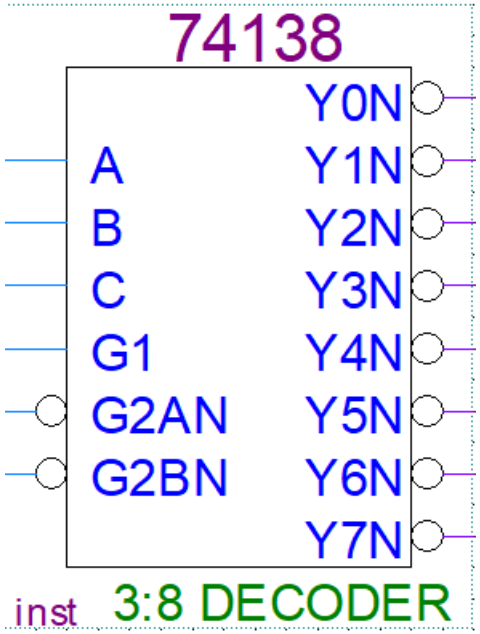
- 加法器
- 比较器

1 译码器和编码器

译码器与编码器的的一般结构：

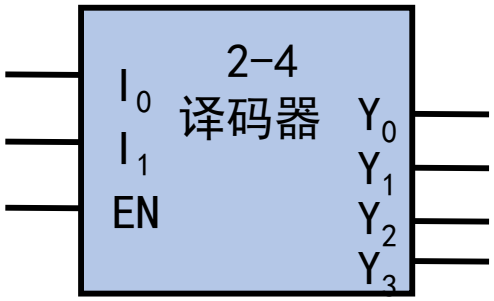
- 译码器输入端数n小于输出端数m；反之，为编码器。
- 输入编码为n位二进制编码；
- 一个n位字表示 2^n 个不同的编码值，通常为：0～(2^n-1)。
- 有时编码值可以少于 2^n 个。





1 译码器和编码器

（1）二进制译码器：最常用的译码器。又称为n - 2^n译码器。其中：输入编码为n位二进制数；输出编码为2^n取1码。换句话说，译码器输出为2^n个最小项，所以有时称最小项发生器。



逻辑框图

例 2-4 译码器

输入代码字：I_1、I_0 ；

输入使能：EN

输出代码字：Y_3 Y_2 Y_1 Y_0

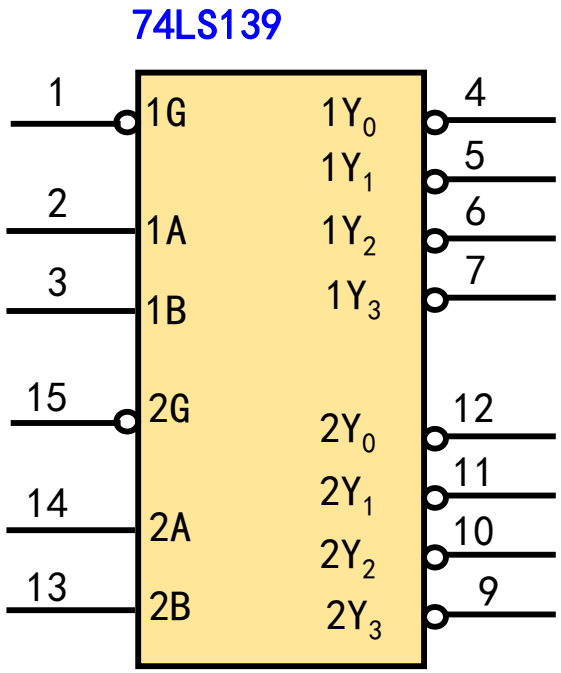
功能描述：当 EN = 1，输入代码字是i的二进制表示，则输出Y_i (i为十进制数)位为1，其他位均为0。

真值表

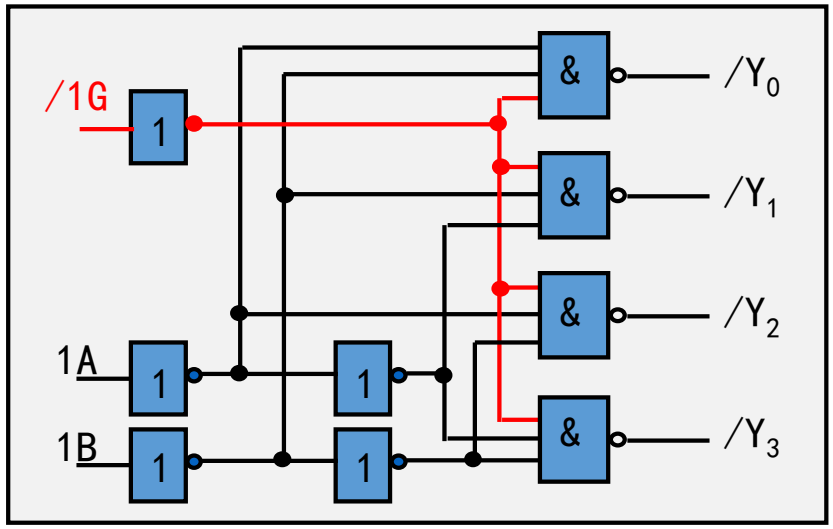
输 入			输 出			
EN	I_1	I_0	Y_3	Y_2	Y_1	Y_0
0	d	d	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

1 译码器和编码器

① 双2-4译码器74LS139 数据手册



输 入			输 出			
/G	B	A	$/Y_3$	$/Y_2$	$/Y_1$	$/Y_0$
1	d	d	1	1	1	1
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 译码器和编码器

② 3-8 译码器

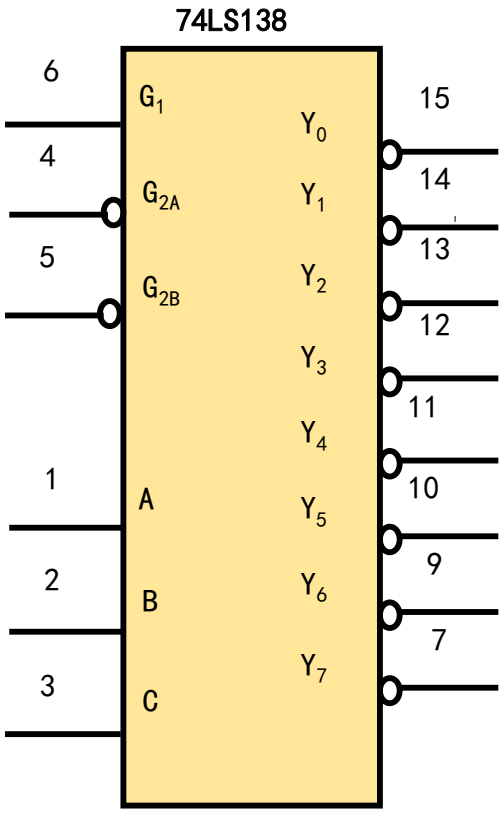
74LS138真值表

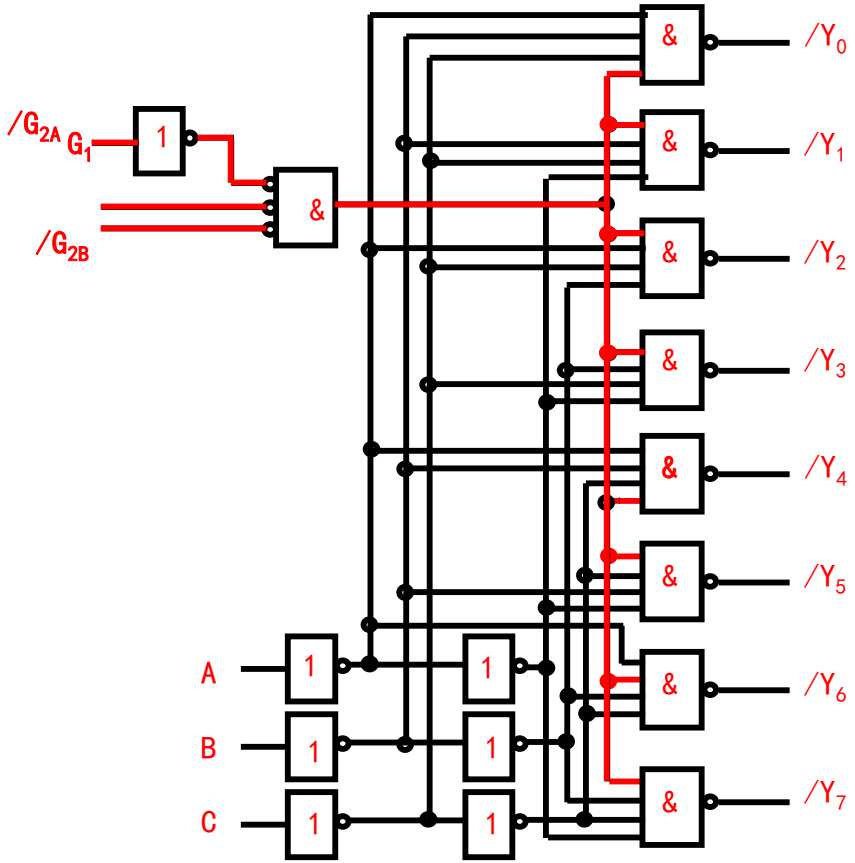
最小项表达式?

输入						输出							
G_1	$/G_{2A}$	$/G_{2B}$	C	B	A	$/Y_7$	$/Y_6$	$/Y_5$	$/Y_4$	$/Y_3$	$/Y_2$	$/Y_1$	$/Y_0$
0	d	d	d	d	d	1	1	1	1	1	1	1	1
d	1	d	d	d	d	1	1	1	1	1	1	1	1
d	d	1	d	d	d	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

1 译码器和编码器

74LS138数据手册





译码器74LS138的使用要点

- 74LS138的输出信号为低有效，它有三个使能输入端（G_1、$/G_{2A}$、$/G_{2B}$），只有在三个使能输入全部有效时，才能有正确的有效输出。

- 74LS138的内部功能可用逻辑表达式描述如下：

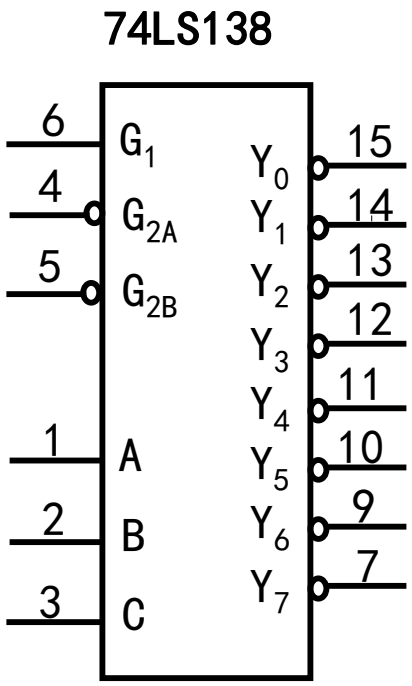
$$Y_i = G_1 \cdot \overline{G_{2A}} \cdot \overline{G_{2B}} \cdot m_i$$

其中，Y_i为内部输出编码字的第i位，

m_i为输入变量C、B、A的最小项。

- 74LS138 外部信号之间的关系为：

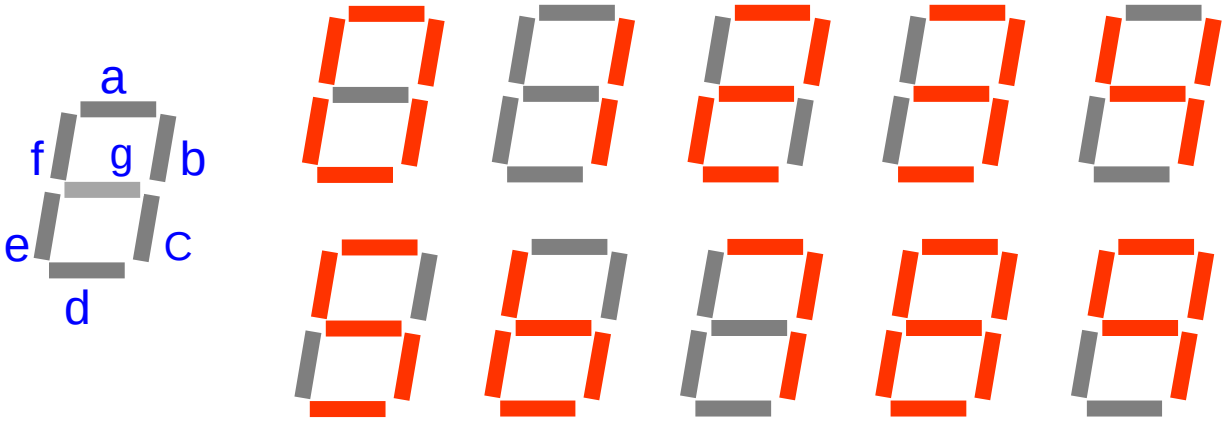
$$/Y_i = \overline{G_1 \cdot \overline{G_{2A}} \cdot \overline{G_{2B}} \cdot m_i}$$



如何构成4-16译码器?

（2）BCD译码器74LS49

74LS49是一种常用的BCD码MSI器件，它输入编码为4位的BCD码，输出为7位编码字。



七段显示器件结构

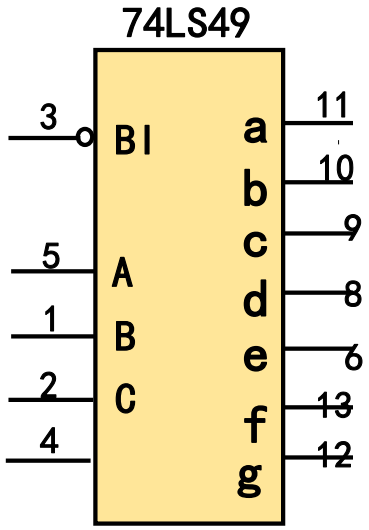
1 译码器和编码器

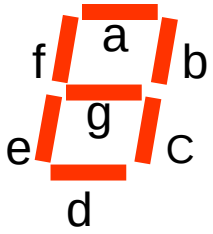
74LS49真值表

（BI：禁止显示控制端）

输入					输出						
/BI	D	C	B	A	a	b	c	d	e	f	g
0	d	d	d	d	0	0	0	0	0	0	0
0 1	0	0	0	0	1	1	1	1	1	1	0
1 1	0	0	0	1	0	1	1	0	0	0	0
2 1	0	0	1	0	1	1	0	1	1	0	1
3 1	0	0	1	1	1	1	1	1	0	0	1
4 1	0	1	0	0	0	1	1	0	0	1	1
5 1	0	1	0	1	1	0	1	1	0	1	1
6 1	0	1	1	0	0	0	1	1	1	1	1
7 1	0	1	1	1	1	1	1	0	0	0	0
8 1	1	0	0	0	1	1	1	1	1	1	1
9 1	1	0	0	1	1	1	1	0	0	1	1
1	1	0	1	0	0	0	0	1	1	0	1
1	1	0	1	1	0	0	1	1	0	0	1
1	1	1	0	0	0	1	0	0	0	1	1
1	1	1	0	1	1	0	0	1	0	1	1
1	1	1	1	0	0	0	0	1	1	1	1
1	1	1	1	1	0	0	0	0	0	0	0

逻辑符号：



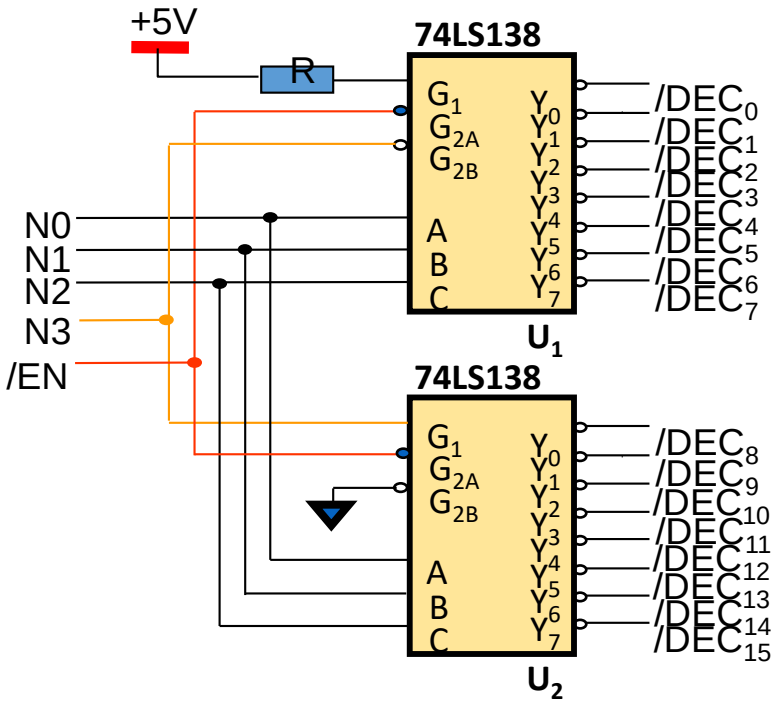


写出ABCD输入，A-F输出的译码器真值表，显示每个字符的逻辑表达式EDA工具输入仿真验证

（3）MSI译码器的级联

当输入变量数 n大于器件的输入变量数时，可以用多个二进制译码器的级联来实现。

例 用两个 3-8 译码器组成 4-16 译码器。



硬件连接：

（1）输入的最高位N_3分别接到U_1-/G_{2A}及U_2-G_1；

（2）使能输入/EN分别接到U_1-/ G_{2B}和U_2-/ G_{2A}。

工作模式（/EN = 0时）：

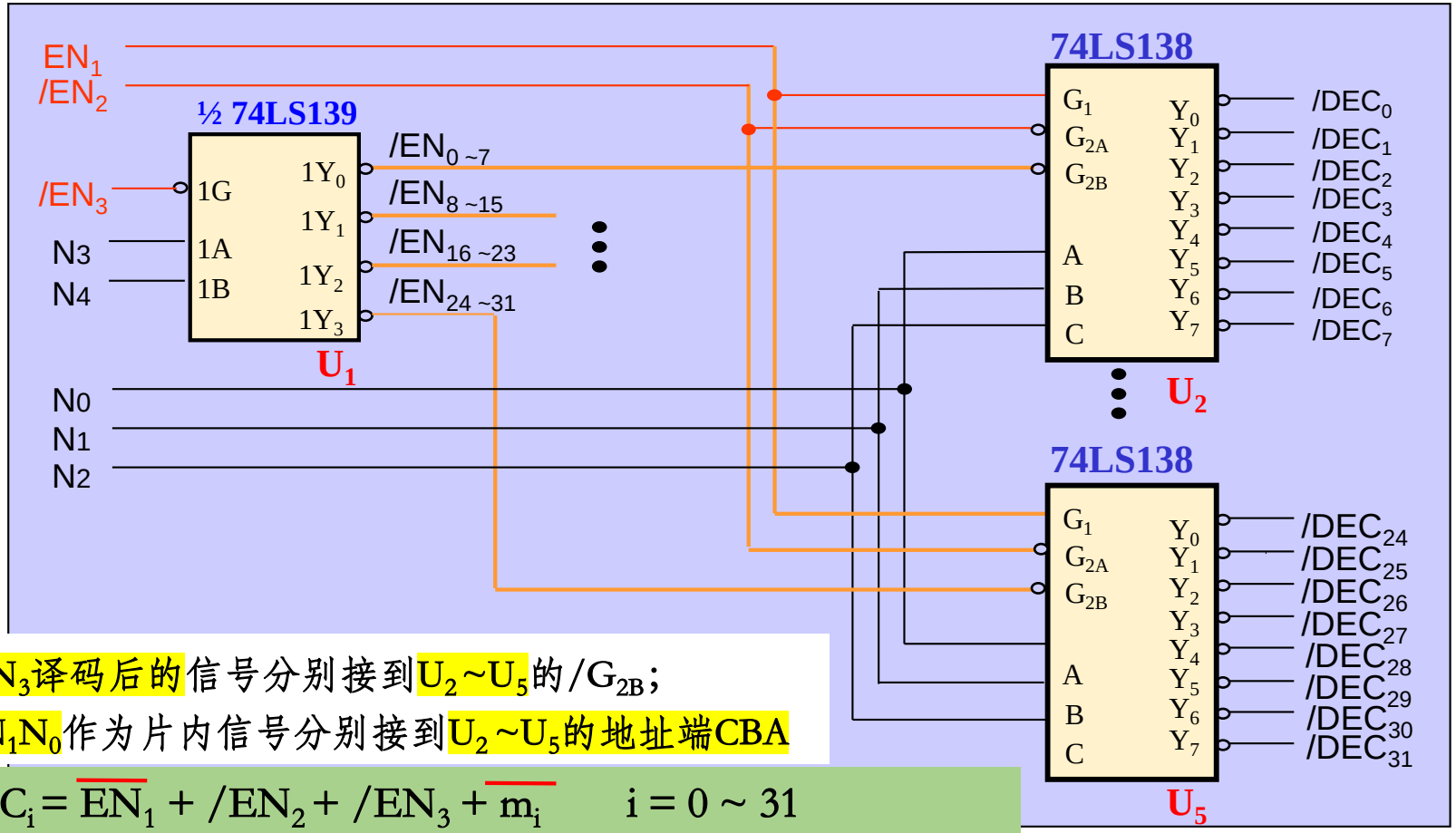
（1）若$N_3=0$，U_2的全部输出无效(输出1)，U_1的输出按$N_2N_1N_0$译码：$/DEC_i=m_i$ （i = 0 ~ 7）

（2）若$N_3=1$，U_1的全部输出无效(输出1)，U_2的输出为按$N_2N_1N_0$译码 $/DEC_i=m_i$ (i = 8 ~ 15)

1 译码器和编码器

例 设计一个 5-32 二进制译码器。

采用四片 74LS138 和一片74LS139组成一个树形结构的级联译码器。



① N_4N_3译码后的信号分别接到U_2~U_5的$/G_{2B}$；

②$N_2N_1N_0$作为片内信号分别接到U_2~U_5的地址端CBA

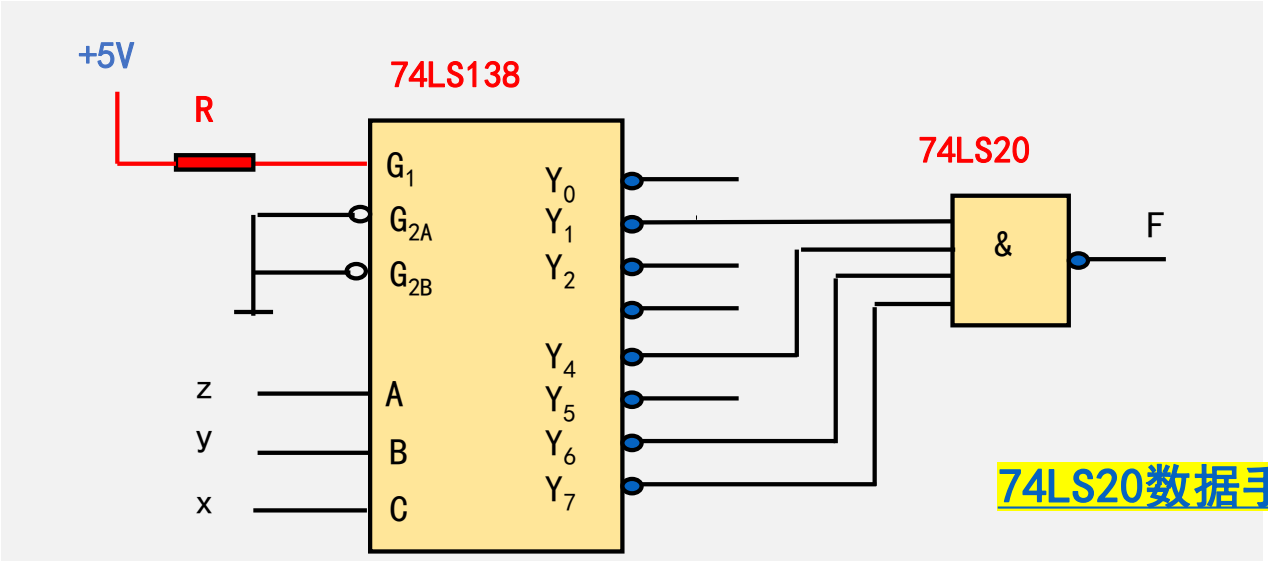
$$/DEC_i = \overline{EN_1} + /EN_2 + /EN_3 + \overline{m_i} \qquad i = 0 \sim 31$$

m_i 为 $N_4N_3N_2N_1N_0$ 的对应最小项。

（4）用MSI译码器实现组合逻辑函数

n - 2^n 二进制译码器的输出对应于 n 变量函数的2^n个最小项，所以可借用此器件来实现任何组合逻辑函数。

例1 用译码器74LS138实现 $F(x,y,z)=\sum m(1,4,6,7)$，逻辑图如下所示：



例2 设计一个一位全加器。

输入端分别为：被加数输入x_i、加数输入y_i、低位向本位的进位输入C_{i-1}

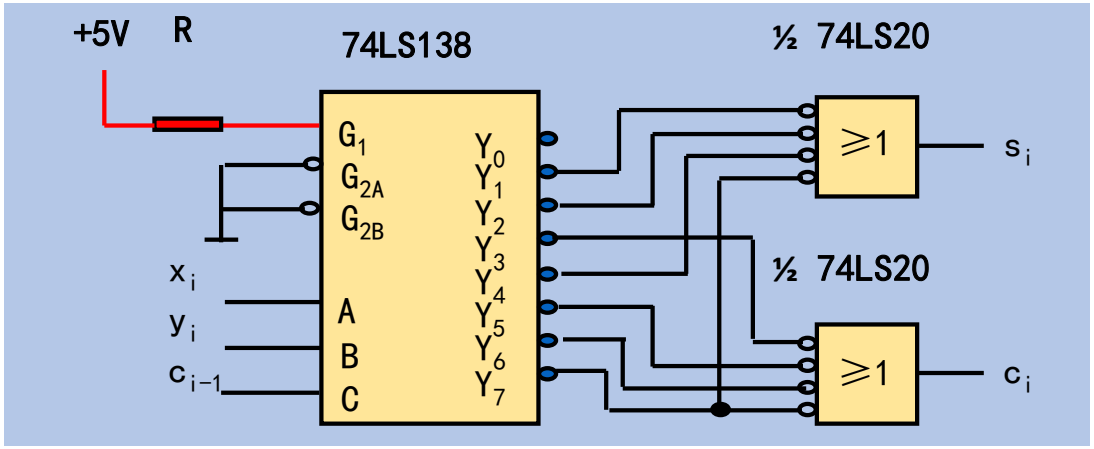
输出端分别为：本位的和输出S_i，本位向高位的进位输出C_i，

一位全加器的真值表如下：

C_{i-1} y_i x_i	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

由真值表得到：

$S_i = \sum m^3(1,2,4,7)$，$C_i = \sum m^3(3,5,6,7)$



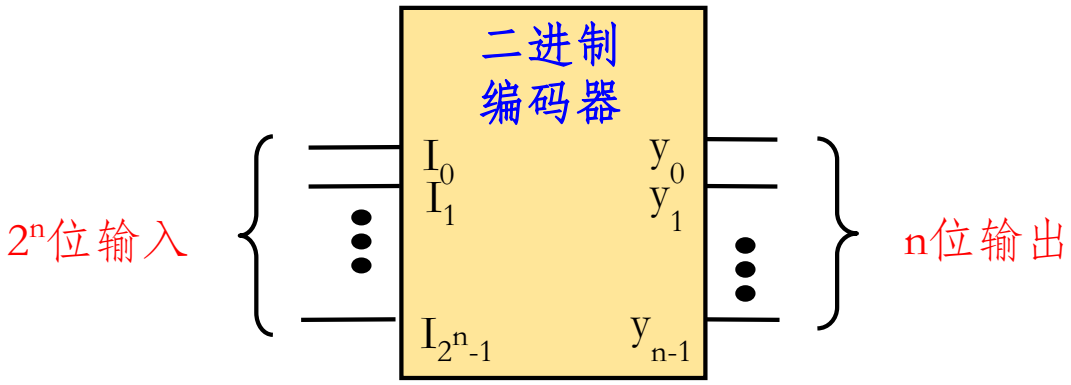
一位全加器逻辑图

2）二进制编码器

当译码器的输出编码位数少于输入编码位数时，这种器件称为编码器。

2^n-n二进制编码器通用结构如图所示，其中输入端为2^n个，输出为n位二进制数，因此它的输入输出关系正好与译码器的相反。

约束条件：同一时刻只能有一个输入端有效。



二进制编码器框图

1 译码器和编码器

例：一个8位输入、3位输出的编码器如图所示。

这是一个8-3二进制编码器。输入：$I_0 \sim I_7$，输出：$Y_0 \sim Y_2$

① 简化真值表(？)

I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0	Y_2 Y_1 Y_0
0 0 0 0 0 0 0 1	0 0 0
0 0 0 0 0 0 1 0	0 0 1
0 0 0 0 0 1 0 0	0 1 0
0 0 0 0 1 0 0 0	0 1 1
0 0 0 1 0 0 0 0	1 0 0
0 0 1 0 0 0 0 0	1 0 1
0 1 0 0 0 0 0 0	1 1 0
1 0 0 0 0 0 0 0	1 1 1

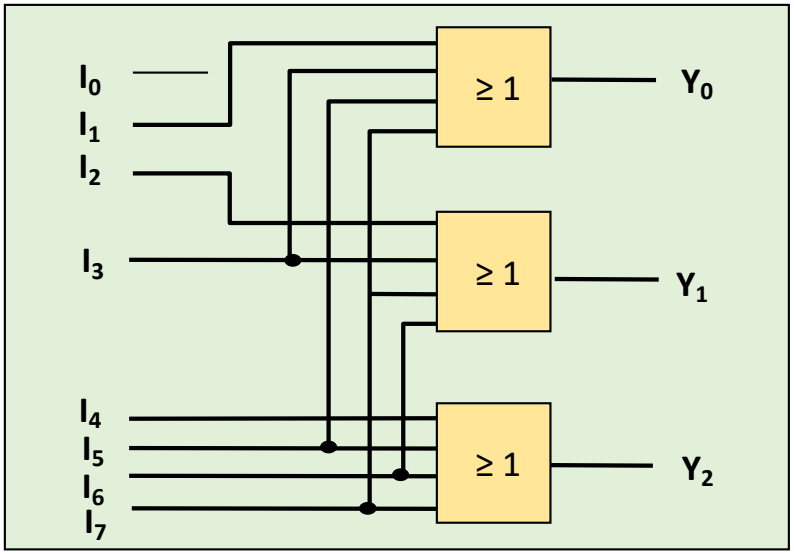
② 输出函数表达式

$$Y_0 = I_1 + I_3 + I_5 + I_7$$
$$Y_1 = I_2 + I_3 + I_6 + I_7$$
$$Y_2 = I_4 + I_5 + I_6 + I_7$$

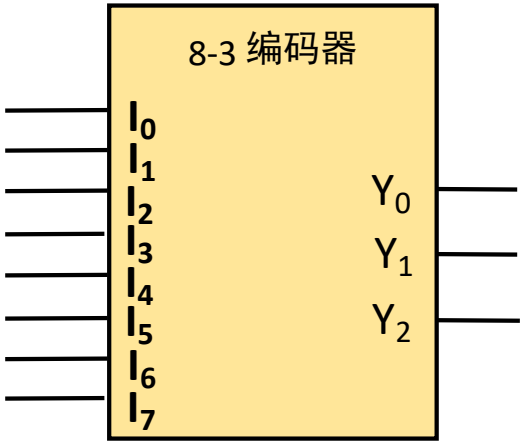
怎样保证？

当且仅当输入代码中的一位为1，输出编码不可能重复。如果：I_1=I_2=1？

③电路图



④ 逻辑符号



I_i与Y_j之间的关系：使Y_j为 1 的是那些I_i，其下标i的二进制数的第 j 位均为1。(From “0”)

例 $Y_1 = I_2+I_3+I_6+I_7$, 即 $Y_1 = I_{010}+I_{011}+I_{110}+I_{111}$ Y_0、Y_2？

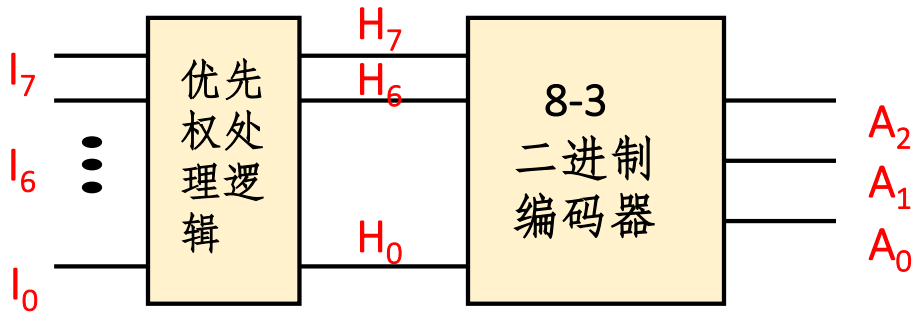
（1）优先权编码器 *Priority Encoders*

- 如果在任一时刻，允许2^n个部件中有多个器件同时提出请求，则2^n-n二进制编码器产生的n位编码必定有重复(?)，而不能与输入请求的条件一一对应了。

- 对输入端进行优先权分配，使编码器仅响应请求中优先权最高的有效输入端，并产生相应的输出编码，这种具有指定输入端优先权顺序的编码器，称为优先权编码器。

8-3 优先权编码器的结构框图如下所示：

设 优先权为：I_7(最高)→ I_6→ I_5 → I_4 → I_3 → I_2 → I_1 → I_0



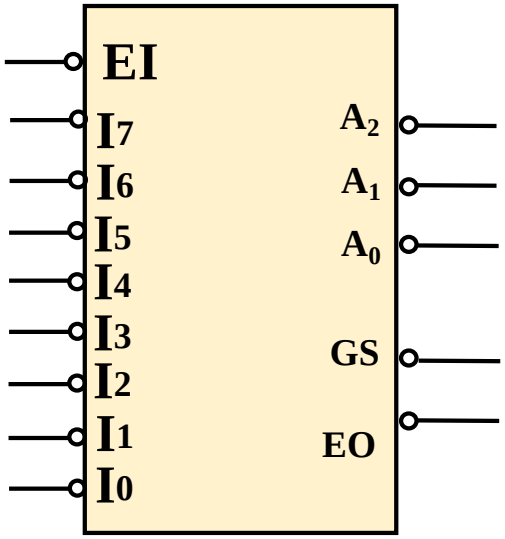
$$A_2 = H_4 + H_5 + H_6 + H_7$$
$$A_1 = H_2 + H_3 + H_6 + H_7$$
$$A_0 = H_1 + H_3 + H_5 + H_7$$

（2） MSI优先权编码器

74LS148

① 逻辑符号

74LS148



② 真值表

输 入									输 出				
/EI	$/I_0$	$/I_1$	$/I_2$	$/I_3$	$/I_4$	$/I_5$	$/I_6$	$/I_7$	$/A_2$	$/A_1$	$/A_0$	/GS	/EO
1	d	d	d	d	d	d	d	d	1	1	1	1	1
0	d	d	d	d	d	d	d	0	0	0	0	0	1
0	d	d	d	d	d	d	0	1	0	0	1	0	1
0	d	d	d	d	d	0	1	1	0	1	0	0	1
0	d	d	d	d	0	1	1	1	0	1	1	0	1
0	d	d	d	0	1	1	1	1	1	0	0	0	1
0	d	d	0	1	1	1	1	1	1	0	1	0	1
0	d	0	1	1	1	1	1	1	1	1	0	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

③ 优先权处理逻辑

设优先权为：I_7(最高) $\rightarrow I_6 \rightarrow I_5 \rightarrow I_4 \rightarrow I_3 \rightarrow I_2 \rightarrow I_1 \rightarrow I_0$

$$H_7 = \overline{I_7}$$
$$H_6 = \overline{I_7}\ I_6$$
$$H_5 = \overline{I_7}\ \overline{I_6}\ I_5$$
$$H_4 = \overline{I_7}\ \overline{I_6}\ \overline{I_5}\ I_4$$
$$H_3 = \overline{I_7}\ \overline{I_6}\ \overline{I_5}\ \overline{I_4}\ I_3$$
$$H_2 = \overline{I_7}\ \overline{I_6}\ \overline{I_5}\ \overline{I_4}\ \overline{I_3}\ I_2$$
$$H_1 = \overline{I_7}\ \overline{I_6}\ \overline{I_5}\ \overline{I_4}\ \overline{I_3}\ \overline{I_2}\ I_1$$
$$H_0 = \overline{I_7}\ \overline{I_6}\ \overline{I_5}\ \overline{I_4}\ \overline{I_3}\ \overline{I_2}\ \overline{I_1}\ I_0$$

④ 输出编码为：

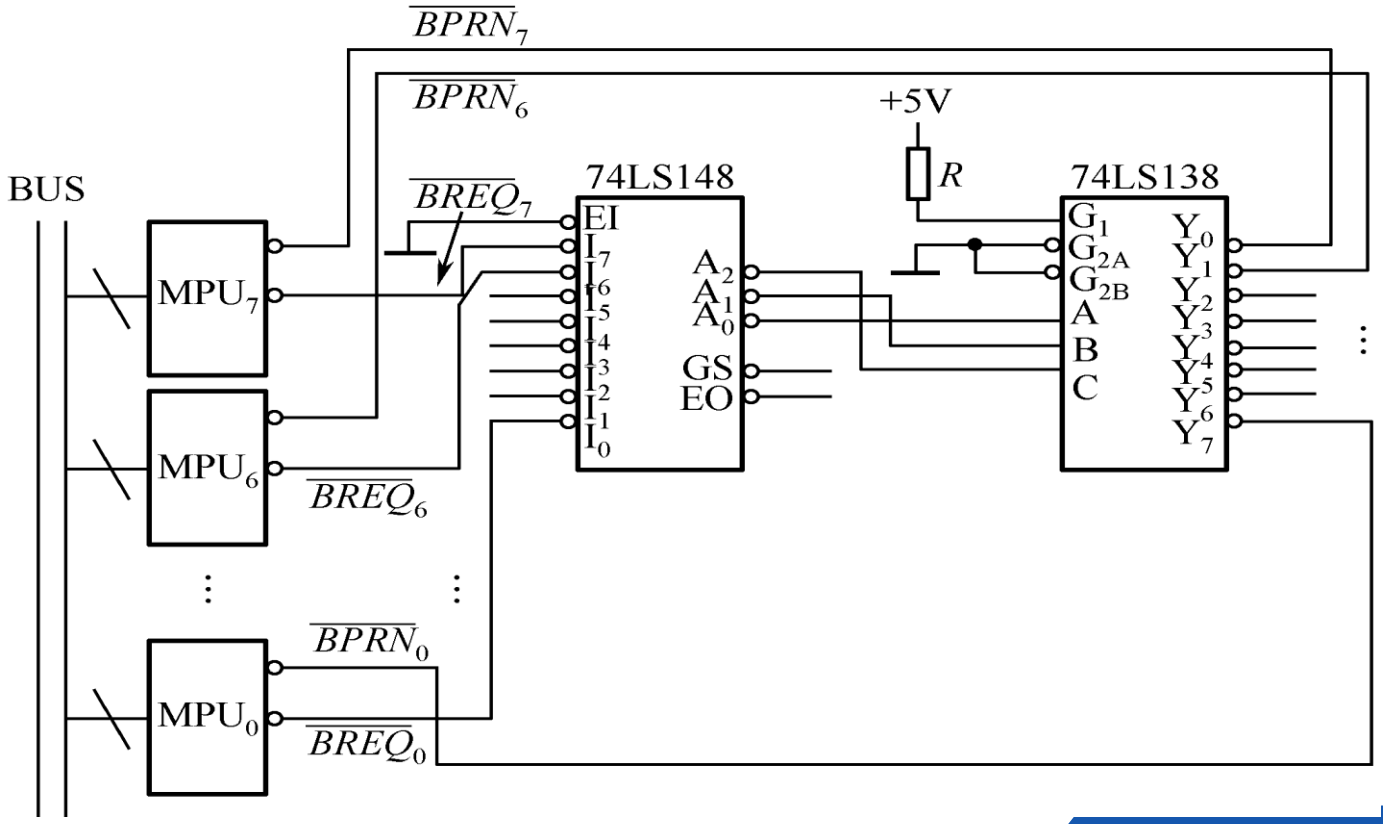
$$/A_2 = \overline{H_4 + H_5 + H_6 + H_7} = ?$$
$$/A_1 = \overline{H_2 + H_3 + H_6 + H_7} = ?$$
$$/A_0 = \overline{H_1 + H_3 + H_5 + H_7} = ?$$

输出使能为：/GS=?

$$/EO = \overline{/I_0\ /I_1 \ldots\ /I_7}$$

(3) 编码器应用举例

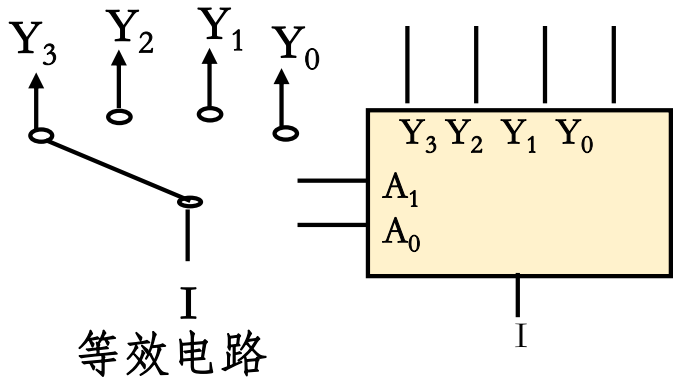
在多处理器系统中，需对各处理器争用总线作出仲裁。为提高仲裁速度，通常采用并行优先权仲裁方式。在争用总线的各处理器进行优先权分配后，通过优先权编码器和译码器进行裁决。逻辑电路图参见书P70图2.45。



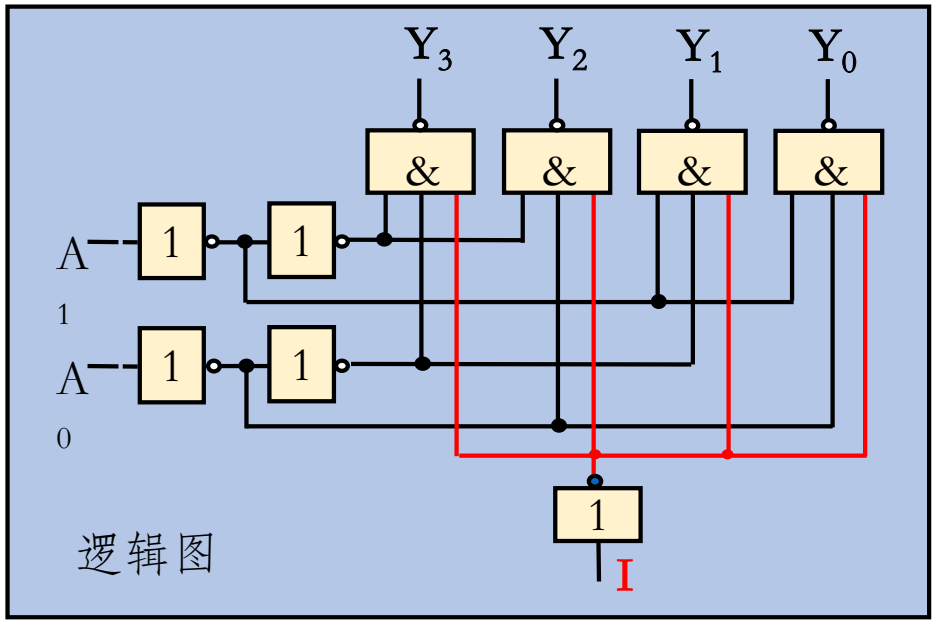
2 数据分配器和多路选择器

1）数据分配器

如图四路数据分配器的等效电路和框图。



单刀多掷开关



I：传送数据输入端；A_1,A_0:地址码输入端；y_3,y_2,y_1,y_0:输出的数据通道，这种分配器被称为“1~4多路分配器”。

一般表达式为：$y_i = I \cdot m_i$ 其中i为地址码$A_{n-1}\cdots\cdots A_0$的十进制值。

2 数据分配器和多路选择器

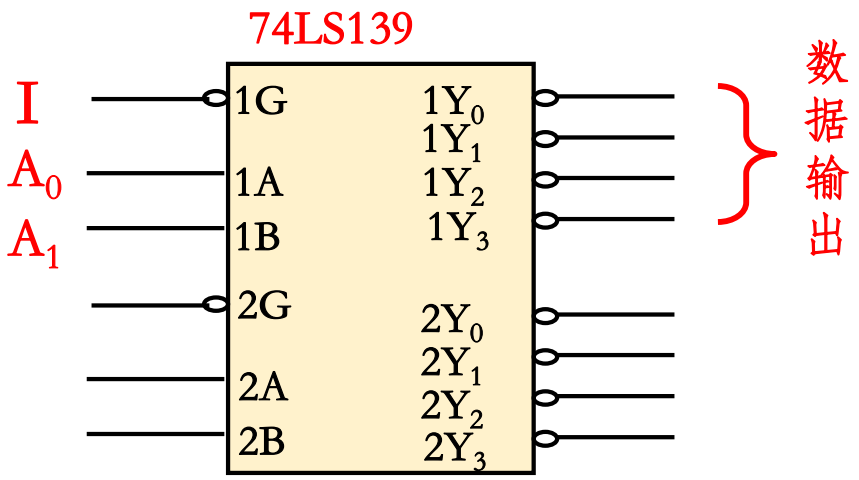
(1) 用二进制译码器作为数据分配器

例 用1/2 74LS139作为四输出数据分配器。

- 将使能端G作为数据输入端，即I接至G端；
- 将数据输入端作为地址输入端，即A端接地址A_0位；B端接地址A_1位。

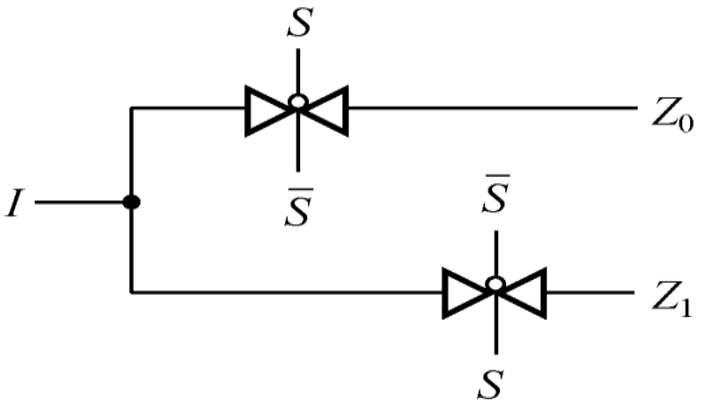
则： $/y_i = I$ i为地址码A_1A_0的十进制数

输 入			输 出			
/G	B	A	$/Y_3$	$/Y_2$	$/Y_1$	$/Y_0$
1	d	d	1	1	1	1
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



(2) 使用CMOS传输门实现的数据分配器的结构

用控制逻辑实现的数据分配器：如果S是有效的，则这一网络将把唯一的输入信号引导到Z_1端；如果S是失效的，则将被引导至Z_0端。这也被称为多路输出选择器，因为它刚好完成了与多路选择器相反的功能。

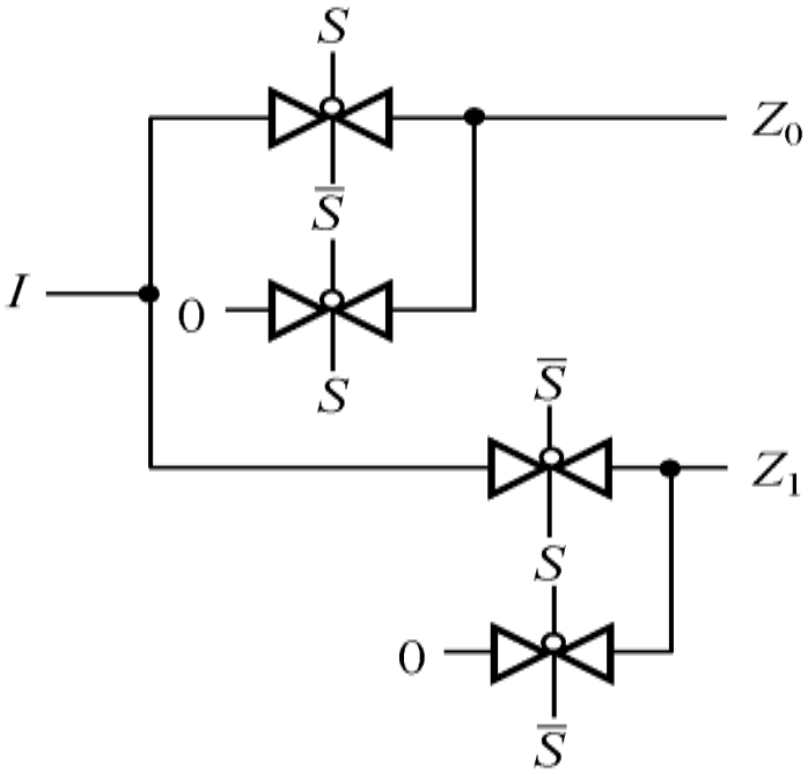


总有一个COMS传输门是高阻，但是Z0或Z1的状态是不确定的！

- 当S有效时，I被导向Z_1，此时输出端Z_0的值又是什么呢？输出端Z_0既不是“0”，也不是“1”，而是悬空的。
- 当S失效时，Z_1端也会有同样的问题,它违反了作为一个合适的功能网络所需的条件之一。

2 数据分配器和多路选择器

下图给出了一种有效的解决方法。当输入信号引导至Z_1时，一个额外的传输门将“0”引导至Z_0。当输入信号引导至Z_0时，对输出端Z_1也做同样的处理。



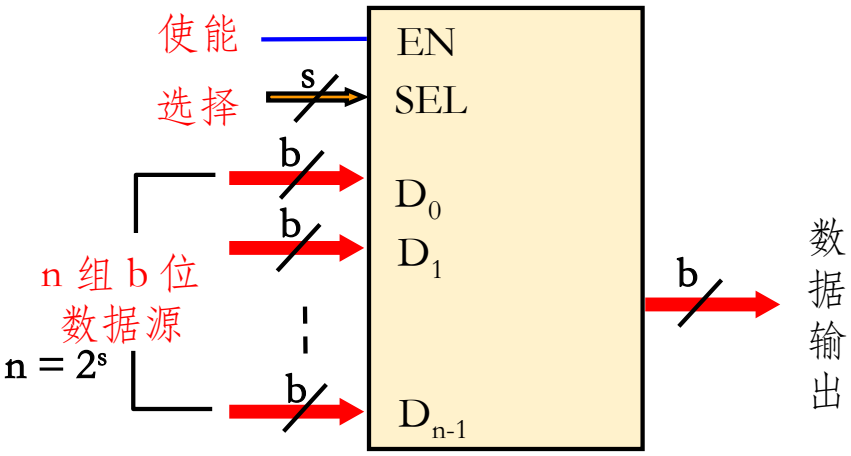
保证能Z0和Z1总有确定的额逻辑状态

2）多路选择器

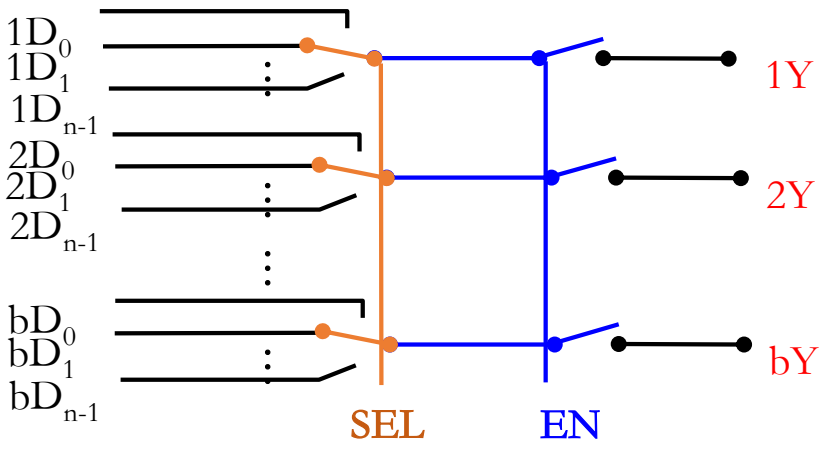
多路选择器又称数据选择器(MUX)，是一个数字开关，可以从 n 路源数据中选择一路送至输出端。

假设有 n 组输入数据源，每组数据源的宽度为b 位二进制数，则反映多路选择器输出关系的框图和等效的电路如下图所示，其中高有效使能端 EN 的功能为：当EN = 0 时，所有的输出为 0。

① 多路选择器的结构框图



② 多路选择器的等效功能



③ 多路选择器输出逻辑表达式

从n组数据源中选择哪一组源数据传送到输出端，由选择输入端的输入值 S 决定。S与 n 的关系为：

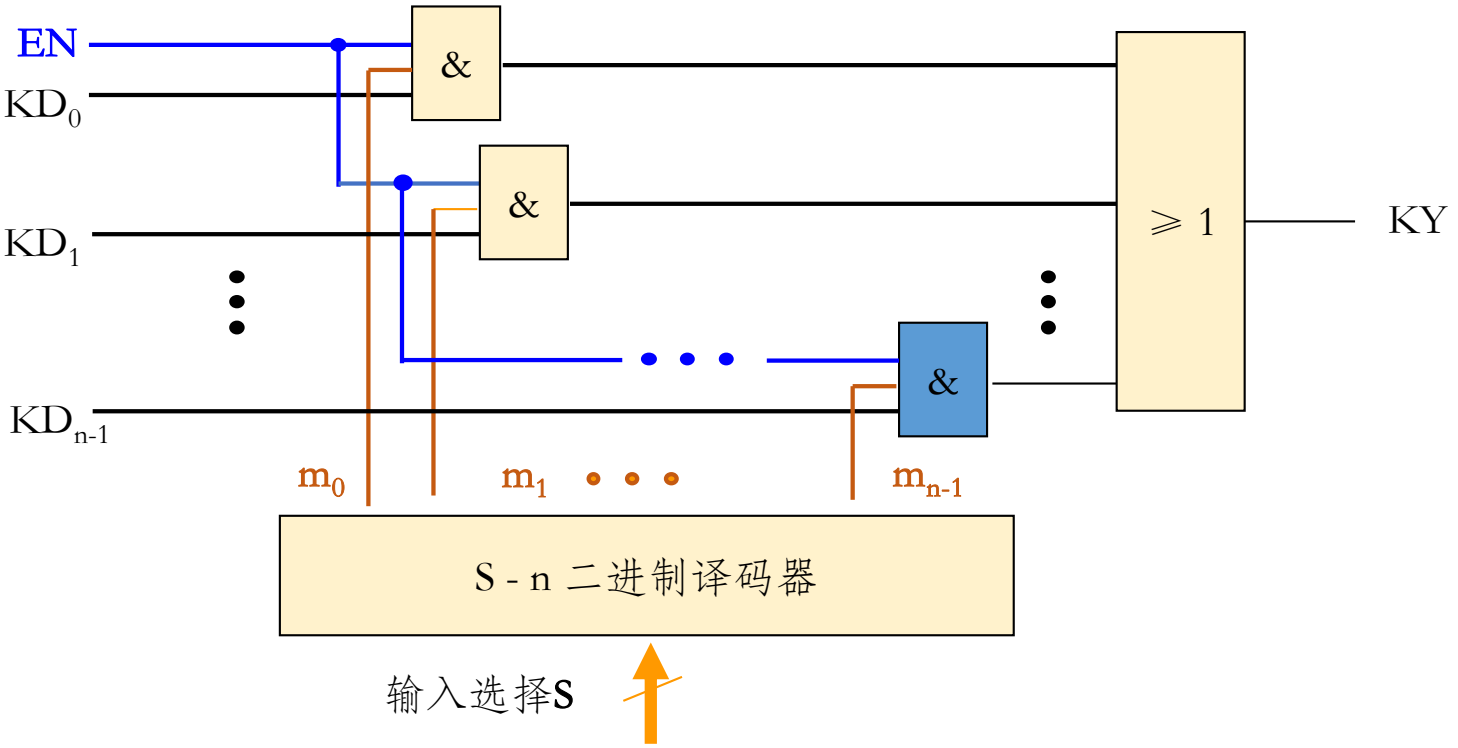
$n = 2^s$ (或 $S = \log_2 n$)

S 位选择信号有 2^s 种组合(即最小项)。每一种组合对应选择 n ($= 2^s$)组输入源数据中的一组。逻辑表达式为：

$$KY = \sum_{i=0}^{n-1} EN \cdot m_i \cdot KD_i \qquad K = 1, 2, \cdots, b$$

式中：KY为输出位，KD_i是第 i 组输入源数据的第K位，m_i是 S 位选择输入变量的最小项。

④ 多路选择器的原理图（某位输出多路选择器）



2 数据分配器和多路选择器

(1) MSI 多路选择器8输入 1 位输出

多路选择器74LS151

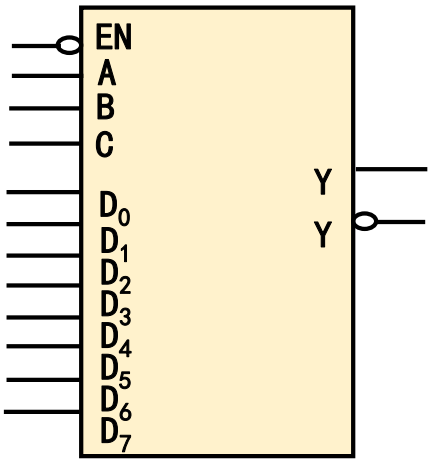
一个低有效使能输入端/EN

三个选择输入端C、B、A

8 个数据输入端 $D_7 \sim D_0$

2 个互反输出 Y、/Y

74LS151



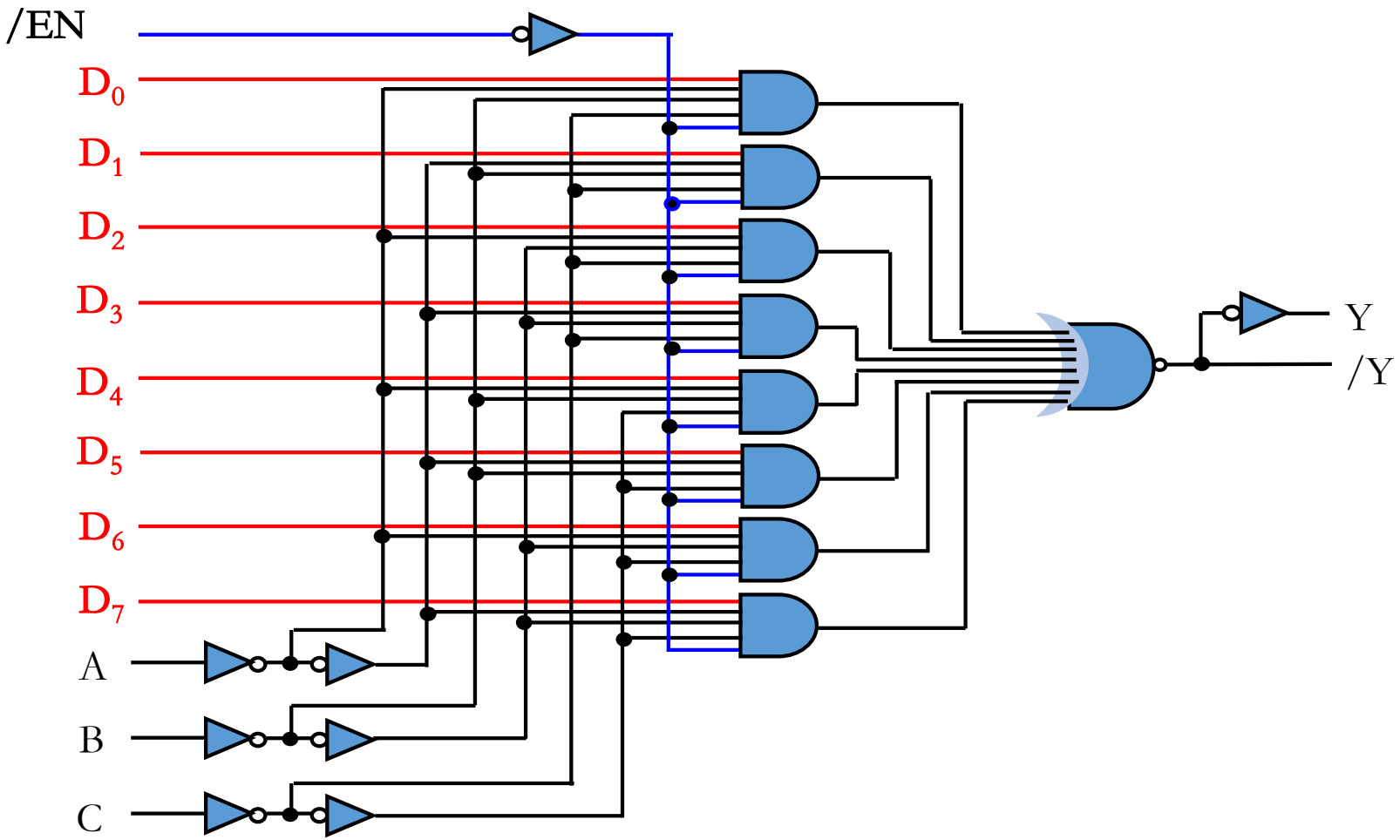
② 逻辑符号

① 简化真值表

输 入				输 出	
/EN	C	B	A	Y	/Y
1	d	d	d	0	1
0	0	0	0	D_0	$\overline{D_0}$
0	0	0	1	D_1	$\overline{D_1}$
0	0	1	0	D_2	$\overline{D_2}$
0	0	1	1	D_3	$\overline{D_3}$
0	1	0	0	D_4	$\overline{D_4}$
0	1	0	1	D_5	$\overline{D_5}$
0	1	1	0	D_6	$\overline{D_6}$
0	1	1	1	D_7	$\overline{D_7}$

2 数据分配器和多路选择器

③逻辑电路图



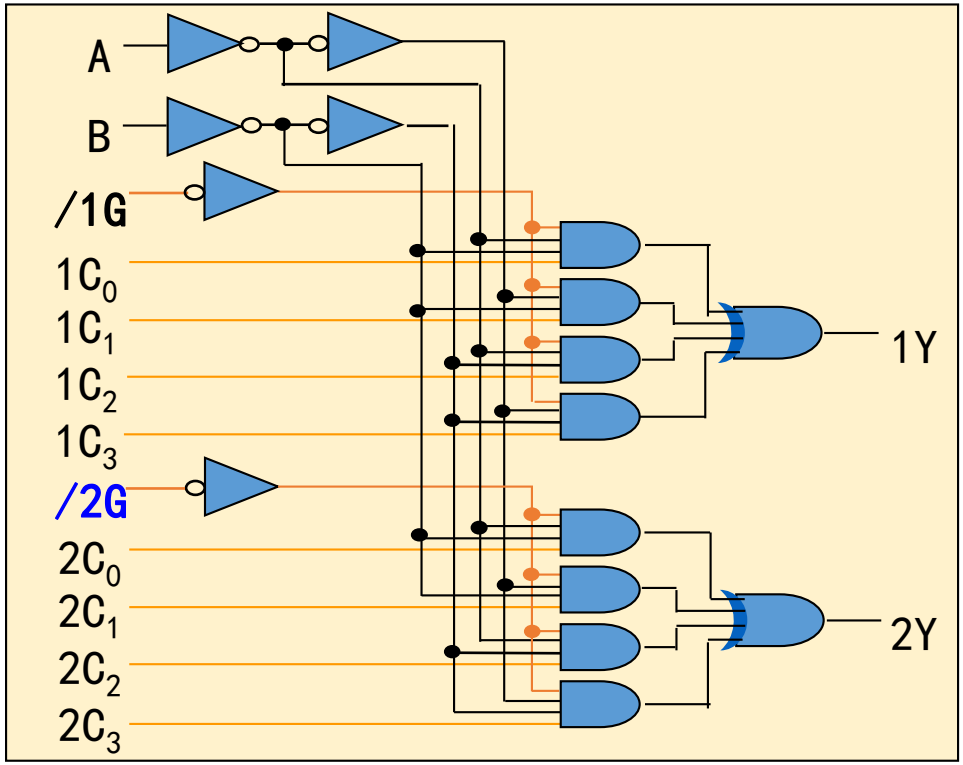
2 数据分配器和多路选择器

四输入2位多路选择器 74LS153

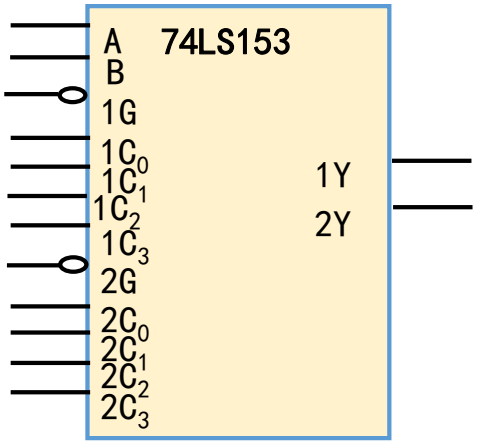
① 简化真值表

输 入			输 出	
/1G	/2G	B A	1Y	2Y
1	d	d	0	0
0	0	0	$1C_0$	$2C_0$
0	0	1	$1C_1$	$2C_1$
0	1	0	$1C_2$	$2C_2$
0	1	1	$1C_3$	$2C_3$

② 逻辑电路图



③ 逻辑符号

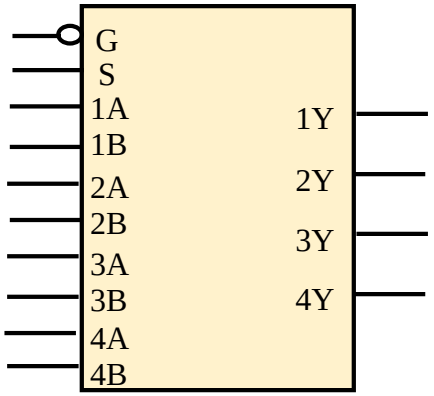


2 数据分配器和多路选择器

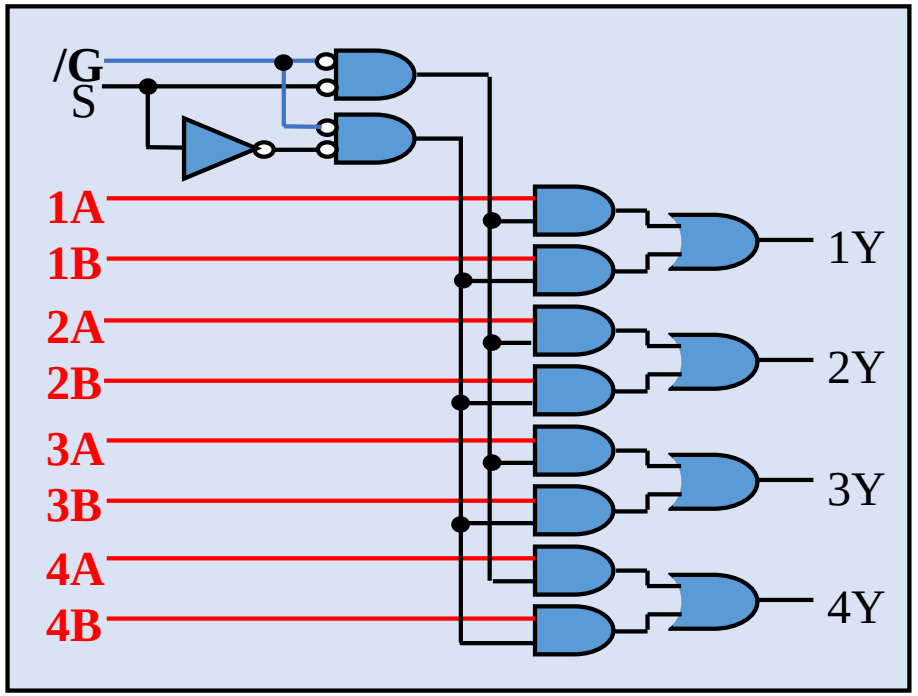
二输入4位多路选择器 74LS157

① 简化真值表

输 入		输 出			
/G	S	1Y	2Y	3Y	4Y
1	d	0	0	0	0
0	0	1A	2A	3A	4A
0	1	1B	2B	3B	4B



② 逻辑电路图



(2) 多路选择器的扩展

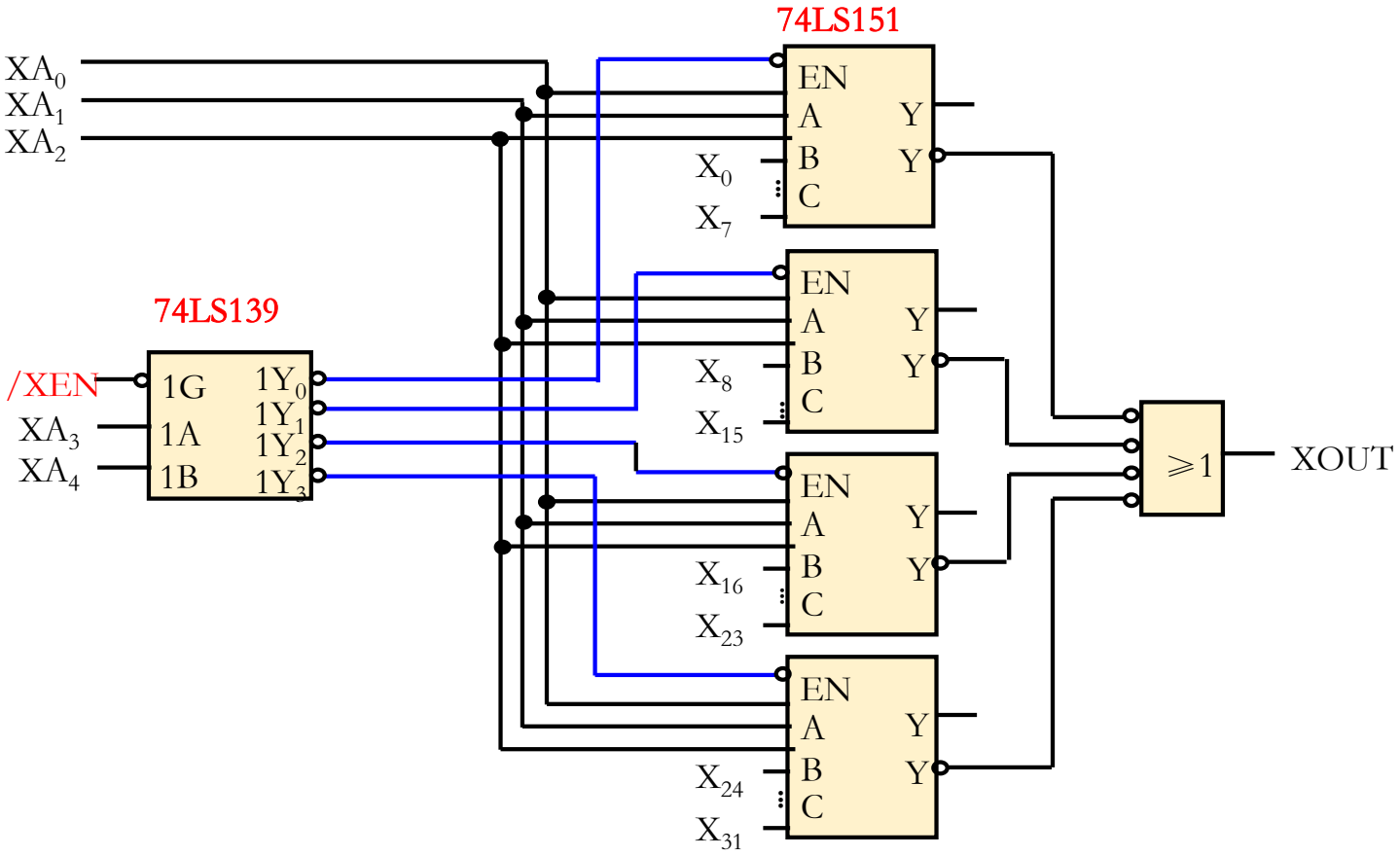
➢ 使用多路选择器及译码器

例：设计一个32输入1位多路选择器。

- 5个选择输入：$XA_4 \sim XA_0$　32路输入：$X_{31} \sim X_0$
- 采用 4 个74LS151，每个器件可处理8个输入，这样将输入分为4组，每组由一个74LS151处理
- 选择输入的低三位$XA_2 \sim XA_0$连接到 4 个74LS151的C、B、A端，决定组内选择
- 选择输入的高二位XA_4、XA_3通过一级2-4译码器1/2 74LS139产生 4 个输出，每个输出连接到一个74LS151的使能输入端

2 数据分配器和多路选择器

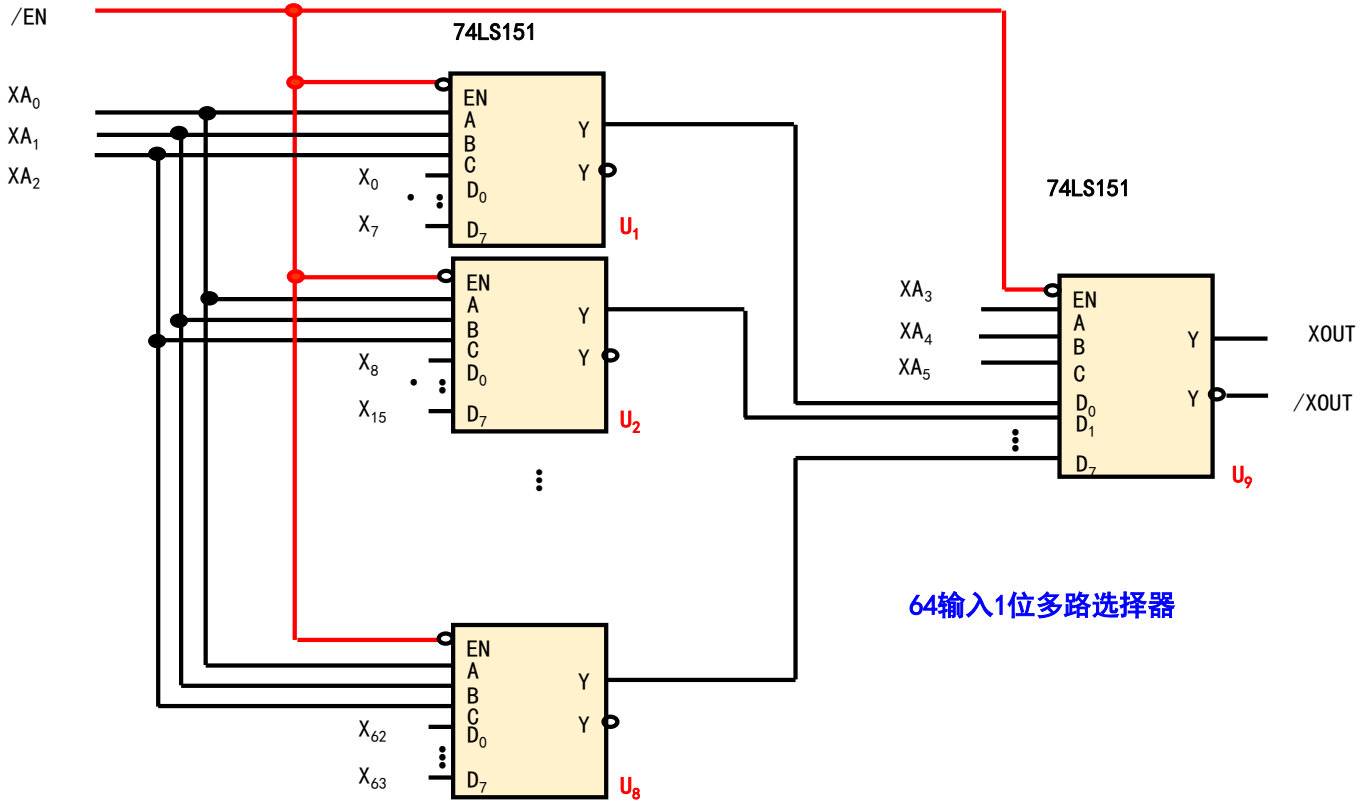
用74LS151组成的 32输入 1 位多路选择器



2 数据分配器和多路选择器

采用多级MUX的树形结构

将多路选择器MUX分级连接，低一级(前一级) MUX的输出作为其高一级(后一级) MUX的数据输入；用选择输入信号的低位控制低一级MUX，高位控制高一级MUX；各级的使能输入可以同一控制。例：采用多级树形结构组成64输入 1 位多路选择器。



(3) 用多路选择器实现任意组合逻辑函数

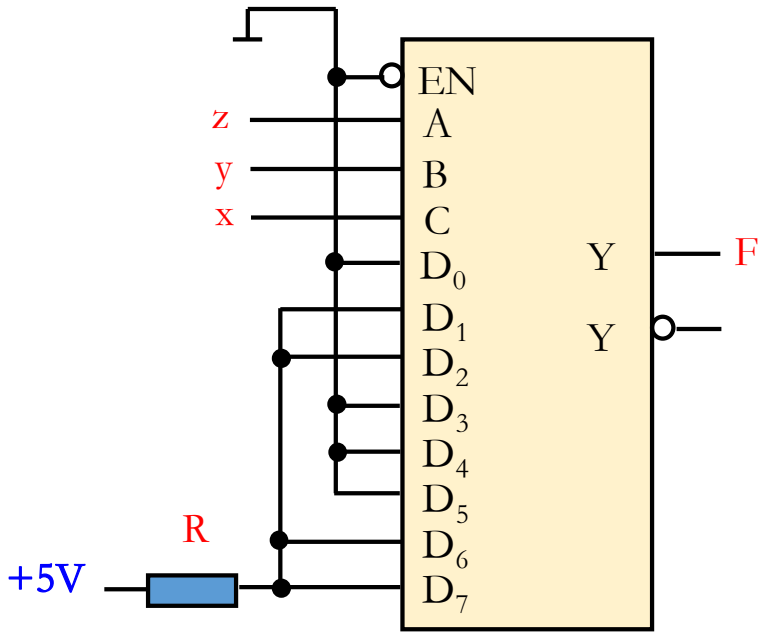
例1 $F(x,y,z) = \sum m^3(1,2,6,7)$

① 用“八选1”多路选择器实现该三变量逻辑函数，选择 S = 3 的MUX 74LS151，把 x、y、z 分别连到 74LS151 的 C、B、A 选择端，并使数据输入端为:

$D_0 = D_3 = D_4 = D_5 = 0$

$D_1 = D_2 = D_6 = D_7 = 1$

则输出端Y的输出即为F。



②用“四选1”多路选择器74LS153实现该三变量逻辑函数

将函数 F 改写成变量表达式： $F(x,y,z) = \sum m^3(1,2,6,7)$

$$= \overline{x}\,\overline{y}\,z + \overline{x}\,y\,\overline{z} + x\,y\,\overline{z} + x\,y\,z$$
$$= \overline{x}\,\overline{y}\,z + \overline{x}\,y\,\overline{z} + x\,y$$
$$= (\overline{x}\,\overline{y}) \cdot z + (\overline{x}\,y) \cdot \overline{z} + (x\,\overline{y}) \cdot 0 + (x\,y) \cdot 1$$

x、y作为地址选择变量，z、$\overline{z}$、0、1

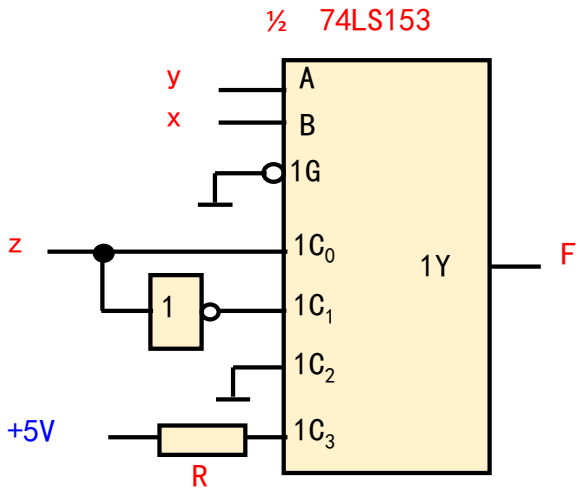
作为MUX的源数据输入D，则有：

$$F = \sum_{i=0}^{3} m_i D_i = m_0 D_0 + m_1 D_1 + m_2 D_2 + m_3 D_3$$

式中 m_i 为x、y的最小项， D_i为：

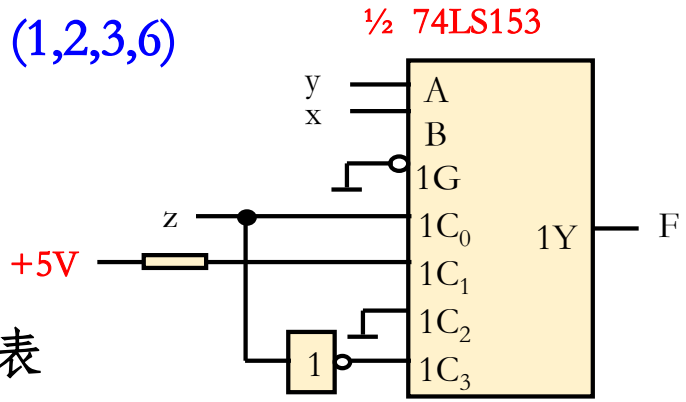
$D_0 = z$, $D_1 = \overline{z}$, $D_2 = 0$, $D_3 = 1$

电路逻辑图如图所示。



2 数据分配器和多路选择器

例2 $F(x,y,z) = \sum m^3(1,2,3,6)$

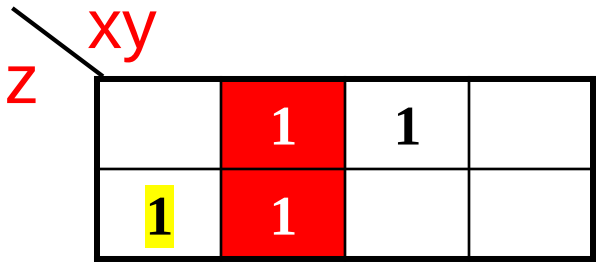


① 列出函数F的真值表

m_i	x y	z	D_i
m_1	0 0	1	$D_0 = z$
m_2	0 1	0	$D_1 = 1$
m_3	0 1	1	
m_6	1 1	0	$D_3 = \overline{z}$

y \ x	0	1
0	z	0
1	1	$\overline{z}$

② 列出函数F的卡诺图



$D_0 = z$

$D_1 = 1$

$D_2 = 0$

$D_3 = \overline{z}$

例3 $F(w,x,y,z)=\sum m^4(3,4,5,6,7,9,10,12,14,15)$

①选择有三个输入选择变量的 8 输入 1 位多路选择器74LS151。将w、x、y分别接入地址端，z 接入数据端。

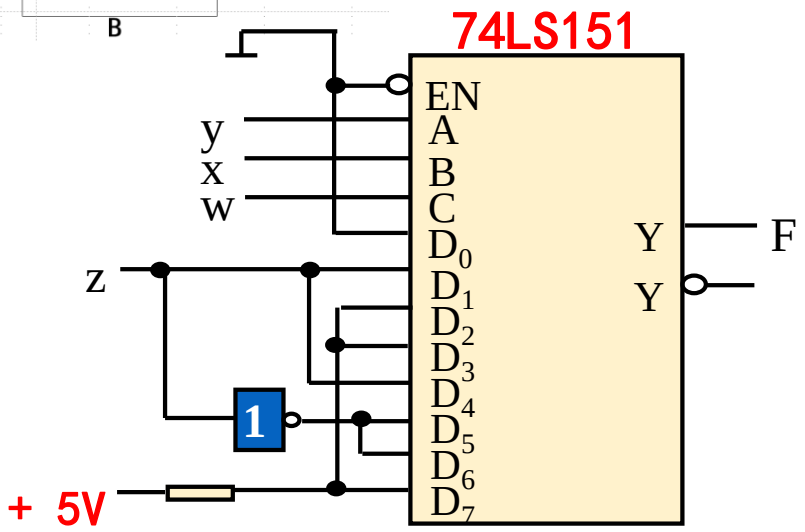
yz \ wx	00	01	11	10
00		1	1	
01		1		1
11	1	1	1	
10		1	1	1

AB \ CD	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

$D_0=0$ $D_1=z$ $D_2=1$

$D_3=1$ $D_4=z$ $D_5=\overline{z}$

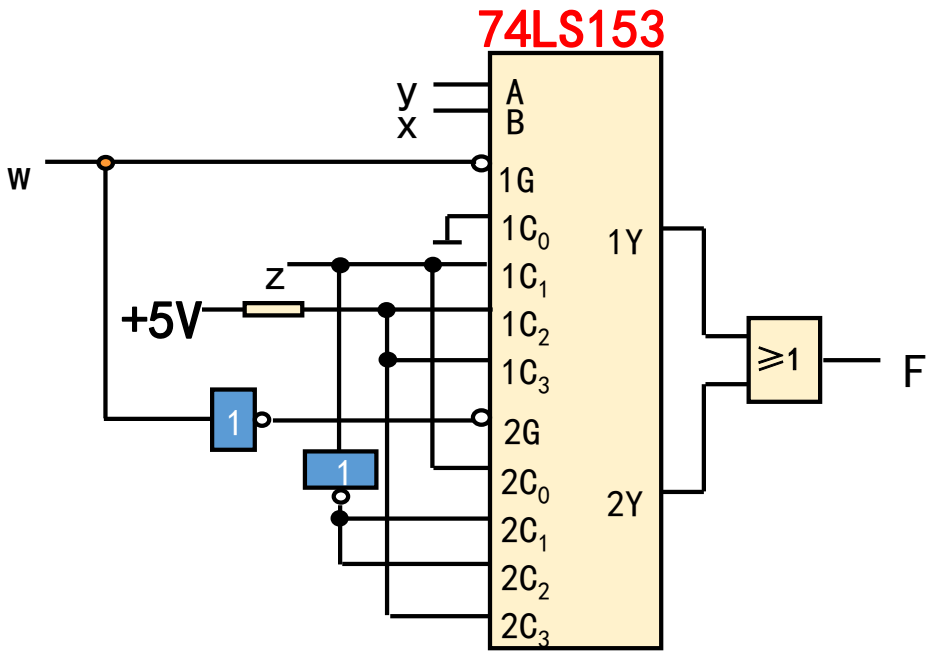
$D_6=\overline{z}$ $D_7=1$



例3 $F(w,x,y,z)=\sum m^4(3,4,5,6,7,9,10,12,14,15)$

②选择4输入2位多路选择器74LS153。将w、x、y作为地址端，z作为数据端。

yz \ wx			
	1	1	
	1		1
1	1	1	
	1	1	1



当w=0 时：$D_0=0$ $D_1=z$

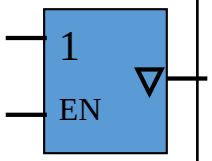
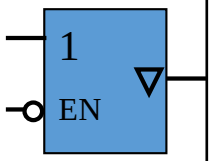
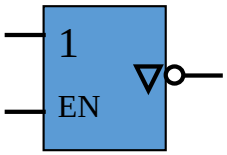
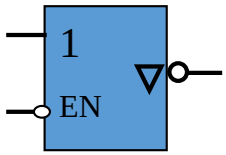
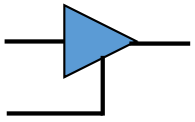
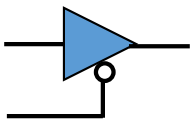
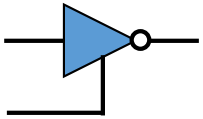
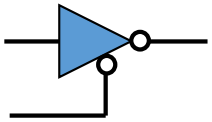
$D_2=1$ $D_3=1$

当w=1 时：$D_4=z$ $D_5=\overline{z}$

$D_6=\overline{z}$ $D_7=1$

3 三态门

三态是指器件的输出有三种状态：即逻辑0 (L电平)、逻辑1 (H电平)和高阻抗状态(或悬浮态)。 最基本的三态器件是三态缓冲器，又称为三态门或三态驱动器。

矩形符号				
变形符号				
	原码输出 高有效使能	原码输出 低有效使能	反码输出 高有效使能	反码输出 低有效使能

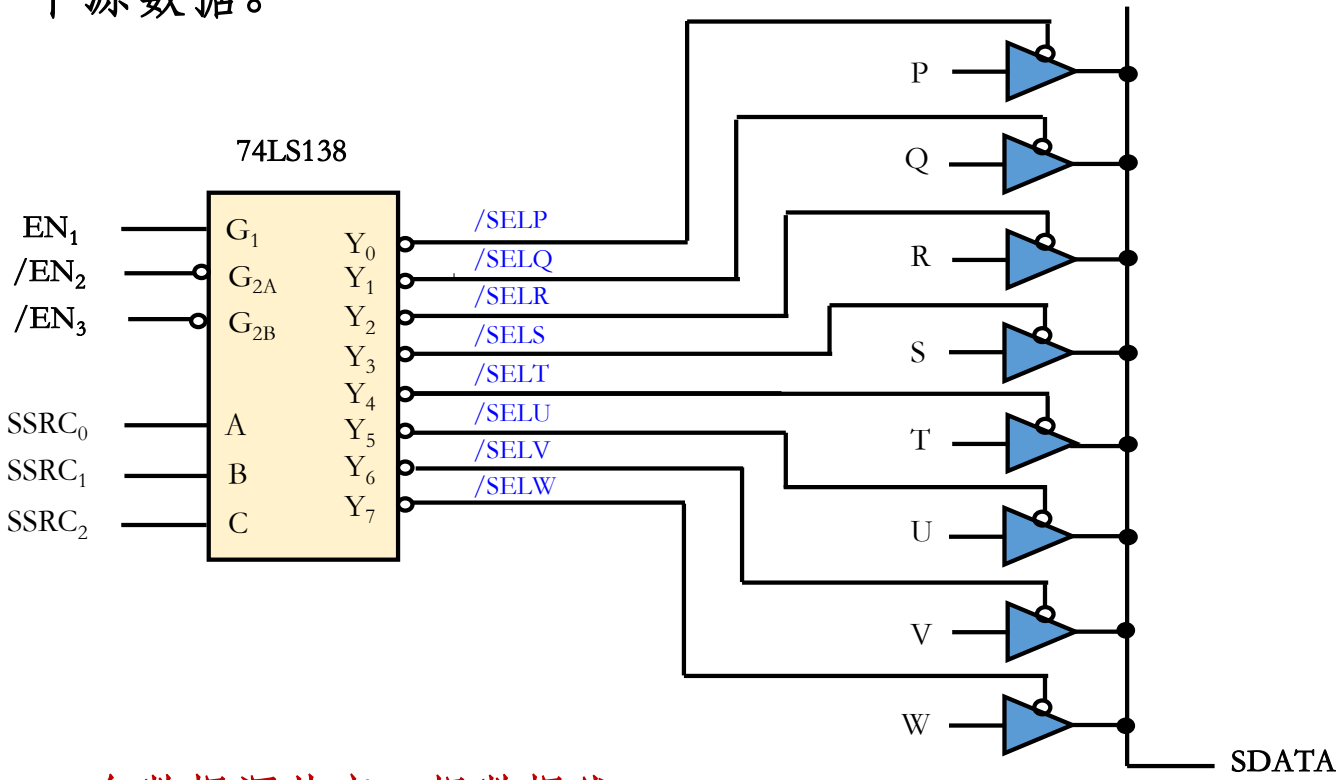
三态缓冲器逻辑符号

三角符号表示三态输出，它受使能输入（EN）端控制。

3 三态门

三态缓冲器可使多个源数据分时共享一根公用线：

为了避免多个源数据同时驱动共享线，则不能在使能一个源数据的同时使能另一个源数据。



8个数据源共享一根数据线

3 三态门

（1）SSI三态缓冲器

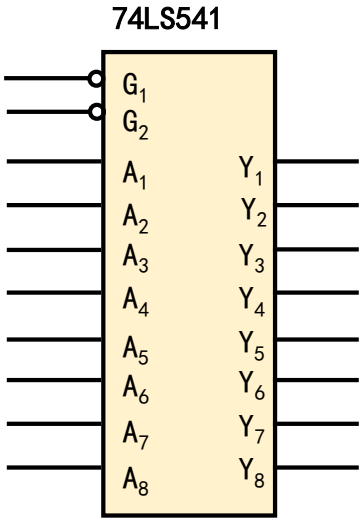
SSI：74LS125使能端低有效。

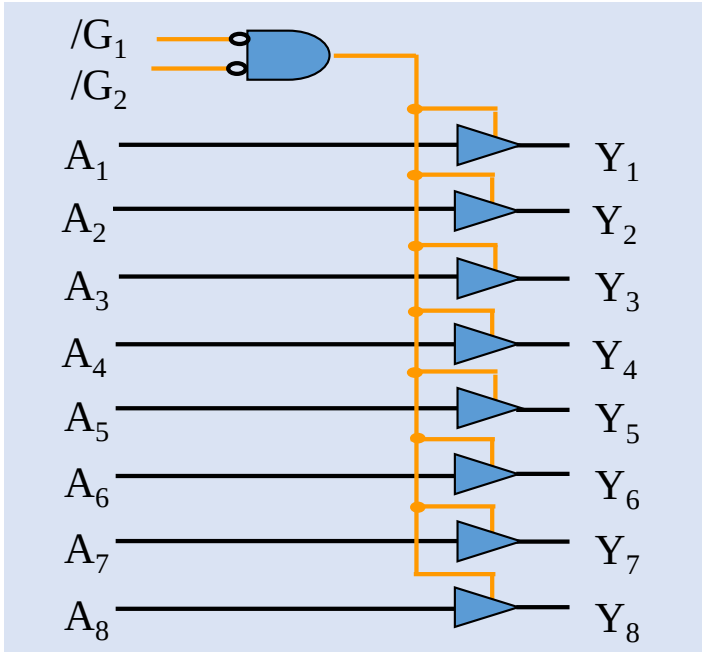
74LS126使能端高有效。四总线缓冲门。常使用共享线的场合是多位数据总线。

例如：在8位微处理机系统中，数据总线的宽度是8，并且外围器件通常一次置8位数据到总线上。这样外围器件都在同一时刻使能8个三态缓冲器，因此独立的使能输入端就多余了。

（2） MSI 三态缓冲器

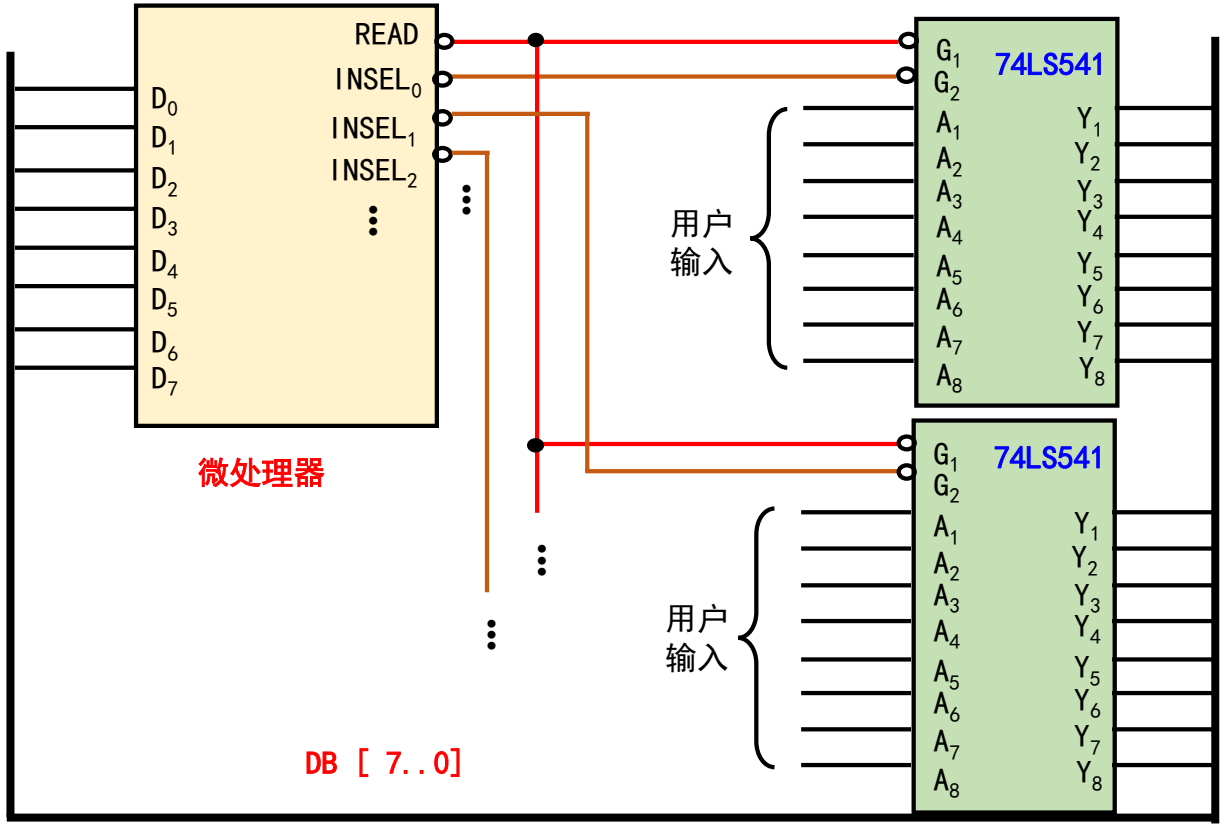
多端口输入，MSI 74LS541为八三态缓冲器



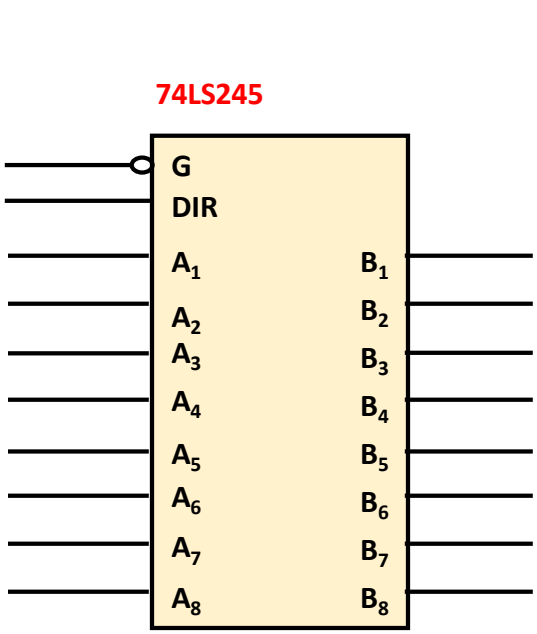


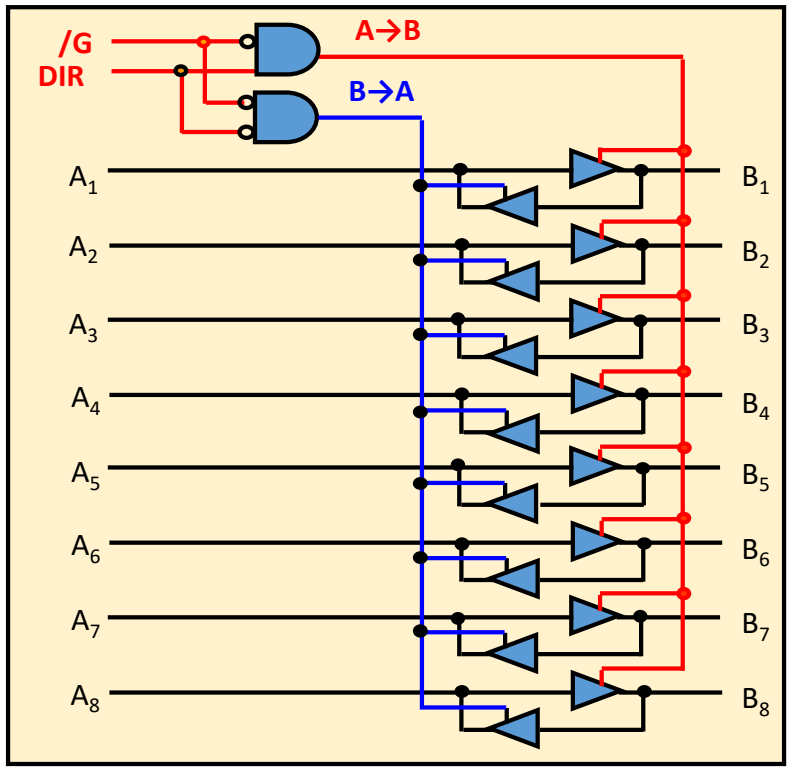
3 三态门

例如：MSI 74LS541（八 三态缓冲器）的应用



② 双向总线收发器74LS245八三态总线收发器





见书P81图2.65

3 三态门

三态输出多路选择器

具有三态输出的多路选择器，当其使能输入无效时，将强制输出端处于高阻抗。

有三态输出端的多路选择器的输出端可以直接连接在一起，使得用这种器件可以方便地组成更大的多路选择器MUX，常用的有74LS251，74LS253和74LS257等。

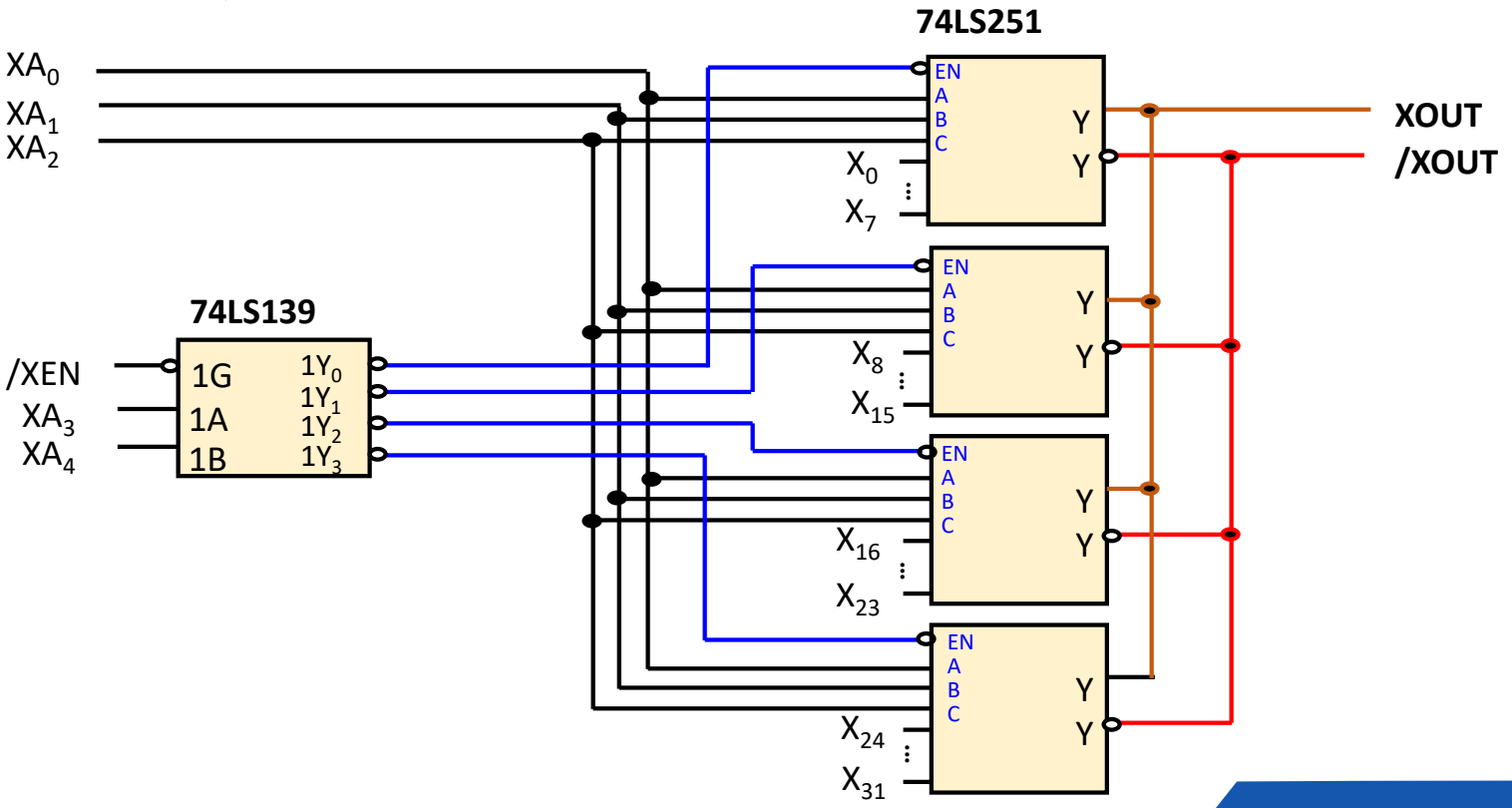
- 74LS151: INPUT: 8 , OUTPUT:1
- 74LS153: INPUT: 4 , OUTPUT:2
- 74LS157: INPUT: 2 , OUTPUT:4
- 74LS251,74LS253, 74LS257: 为三态输出多路选择器。

3 三态门

使用三态输出的多路选择器及译码器

例：用74LS251设计一个32输入1位输出的多路选择器。

（1）当输出处于高阻态时，该输出线可以与其他输出线直接连接在一起，并且不影响其他输出线的高、低电平。（2）在任意时刻只能有一个74LS251被74LS139使能，此时输出线XOUT和/XOUT上的逻辑值就是该被使能的74LS251的输出值。（3）当输入使能/XEN无效时，所有74LS251的输出为高阻态，输出线XOUT和/XOUT上的逻辑值不确定。



4 加法器和比较器

1）比较器

比较器是对两个位数相同的二进制整数进行数值比较，并判断大小关系的逻辑器件。

比较大小关系有三种：　　大于(>)、等于(=)、小于(<)

相等的比较

相等比较的过程总是从高位开始比较，只有当同位比较结果相等时，才进行低位比较。因此，两个一位数的比较是整个比较器操作的基础。

1 位数值比较器(设计)

输入：两个一位二进制数　A、B。

输出：　$F_{A>B}=1$，表示A大于B

$F_{A<B}=1$，表示A小于B

$F_{A=B}=1$，表示A等于B

1 位数值比较器

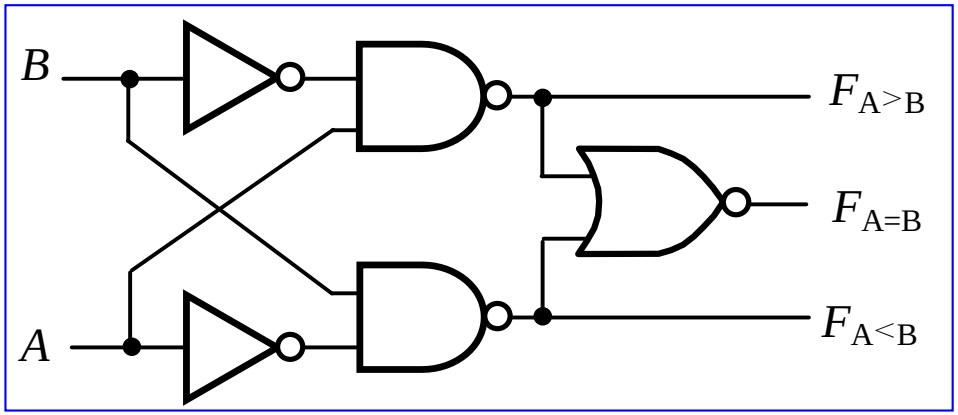
$$F_{A>B} = A\overline{B}$$

$$F_{A<B} = \overline{A}B$$

$$F_{A=B} = \overline{A}\,\overline{B} + AB$$

一位数值比较器真值表

输 入		输 出		
A	B	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1



2 位数值比较器：

比较两个2 位二进制数的大小的电路

输入：两个2位二进制数 $A=A_1A_0$、$B=B_1B_0$

能否用1位数值比较器设计两位数值比较器？Y

用一位数值比较器设计多位数值比较器的原则

当高位（A_1、B_1）不相等时，无需比较低位（A_0、B_0），高位比较的结果就是两个数的比较结果。

当高位相等时，两数的比较结果由低位比较的结果决定。

4 加法器和比较器

真值表

输　入				输　出		
A_1	B_1	A_0	B_0	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
$A_1 > B_1$		×		1	0	0
$A_1 < B_1$		×		0	1	0
$A_1 = B_1$		$A_0 > B_0$		1	0	0
$A_1 = B_1$		$A_0 < B_0$		0	1	0
$A_1 = B_1$		$A_0 = B_0$		0	0	1

$$F_{A>B} = (A_1>B_1) + (A_1=B_1)(A_0>B_0)$$

$$F_{A<B} = (A_1<B_1) + (A_1=B_1)(A_0<B_0)$$

$$F_{A=B}=(A_1=B_1)(A_0=B_0)$$

注意：上述不是真正的逻辑函数表达式，只示意逻辑关系。

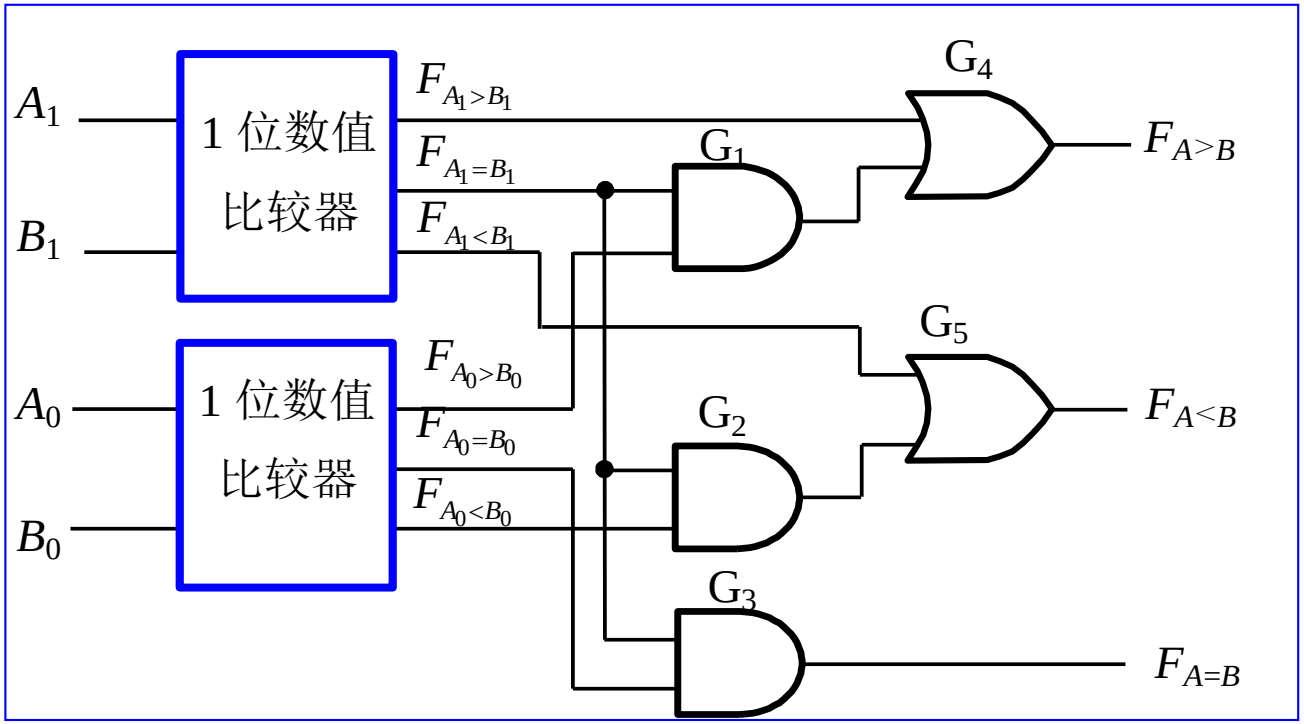
4 加法器和比较器

$$F_{A>B} = (A_1>B_1) + (A_1=B_1)(A_0>B_0)$$

$$F_{A=B}=(A_1=B_1)(A_0=B_0)$$

$$F_{A<B} = (A_1<B_1) + (A_1=B_1)(A_0<B_0)$$

两位数值比较器逻辑图

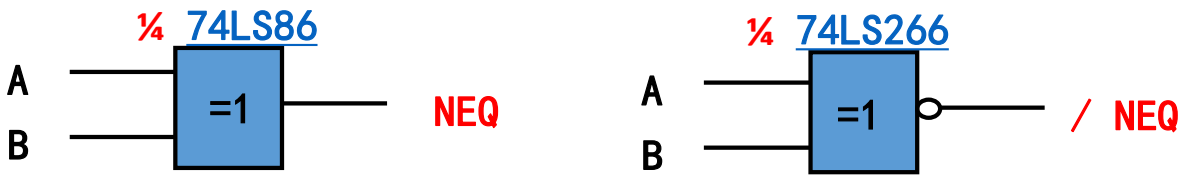


4 加法器和比较器

判断两个一位二进制数是否相等，可用异或门或异或非门实现：

$NEQ = A \oplus B$　　$/NEQ = \overline{A \oplus B}$

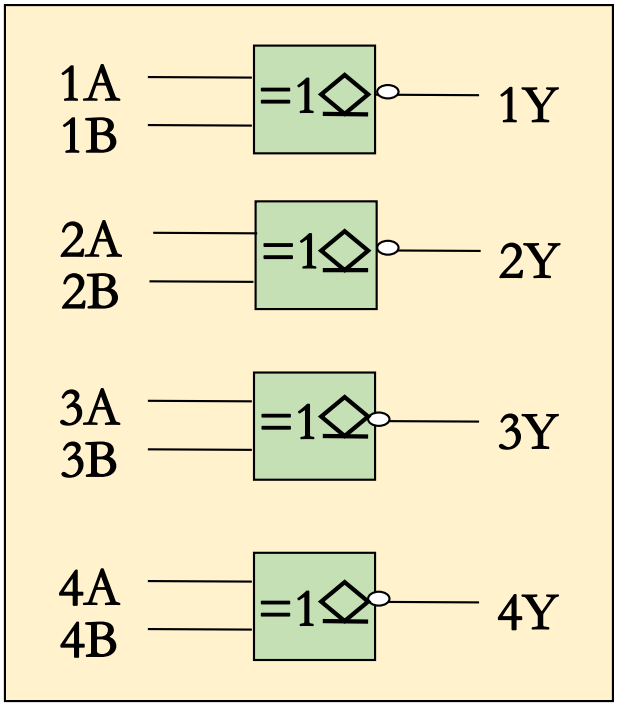
对应的逻辑电路：



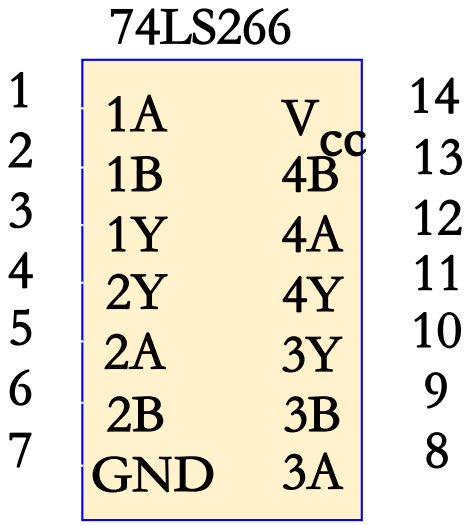
同理，异或门及异或非门也可以实现两个多位数的比较。

例：四位二进制数的相等比较器。$NEQ = (A0 \oplus B0) + (A1 \oplus B1) + (A2 \oplus B2) + (A3 \oplus B3)$

集电极开路输出2输入4异或非门74LS266

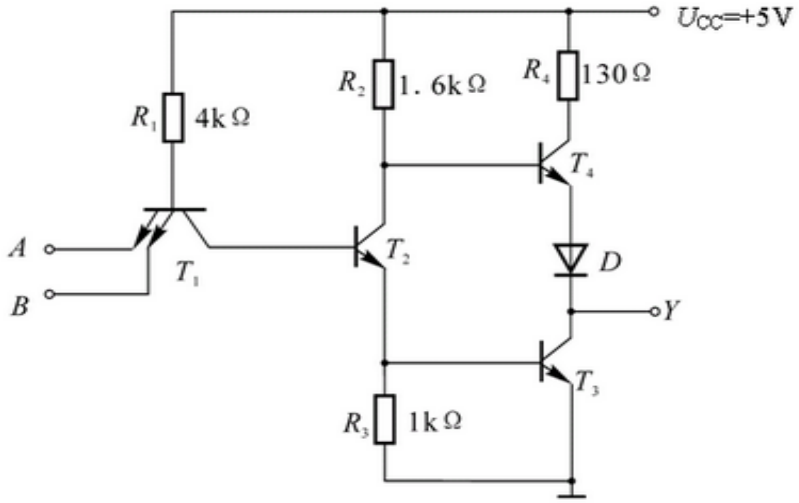


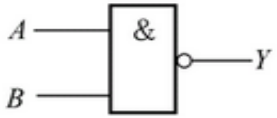
(a) 逻辑框图

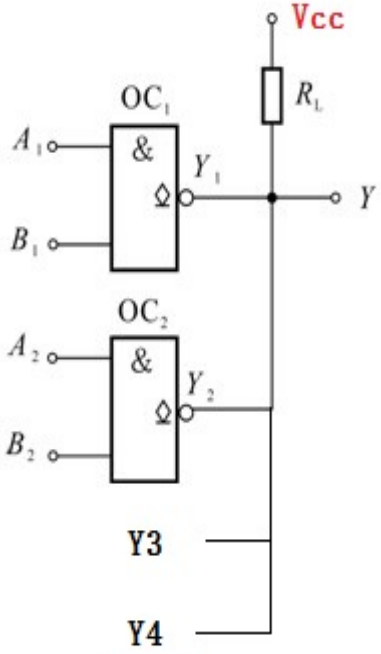


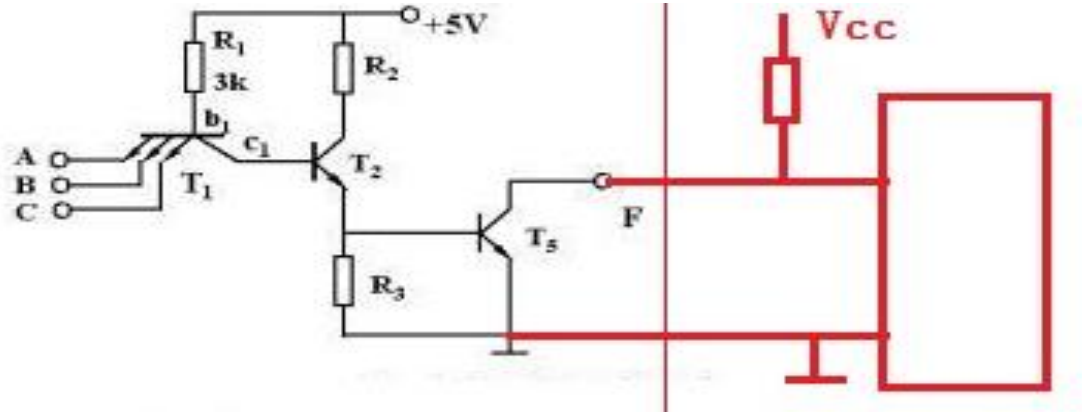
(b)引脚图

4 加法器和比较器


U_{CC}=+5V
R_1 4kΩ
R_2 1.6kΩ
R_4 130Ω
R_3 1kΩ
A
B
T_1
T_2
T_3
T_4
D
Y


A
B
&
Y

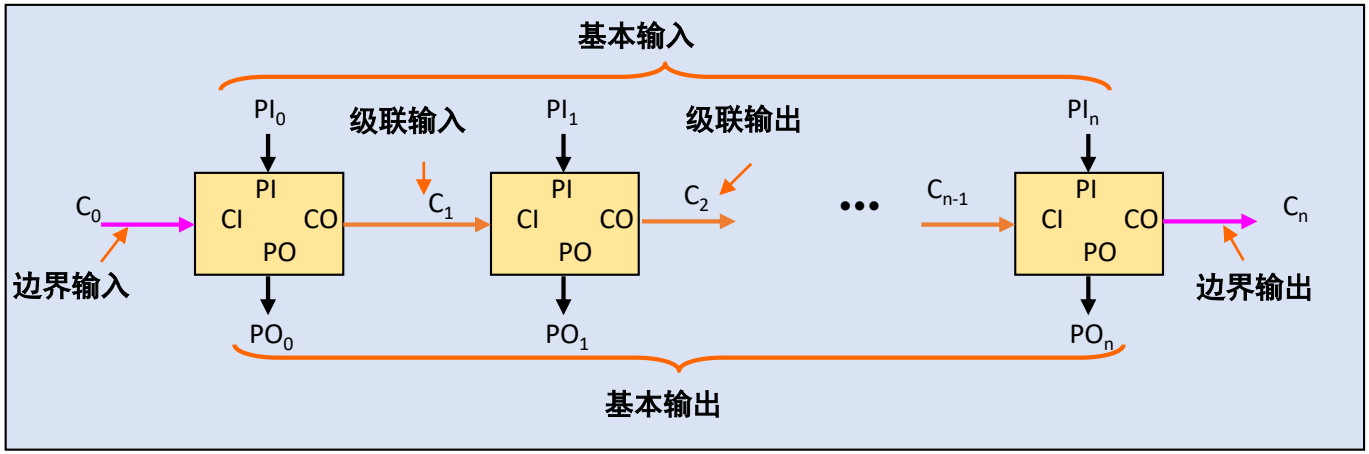

Vcc
R_L
OC_1
A_1
B_1
&
Y_1
Y
OC_2
A_2
B_2
&
Y_2
Y3
Y4


+5V
R_1
3k
R_2
b_1
c_1
A
B
C
T_1
T_2
T_5
R_3
F
Vcc

4 加法器和比较器

(1) 重复电路 (串行组合电路)

串行重复电路是一种组合逻辑电路，它包含几个同样的模块，每个模块有四种类型的输入和输出：基本输入、基本输出、级联输入、级联输出。其中：最左边的级联输入称为边界输入，最右边的级联输出称为边界输出。电路的一般结构如图所示：

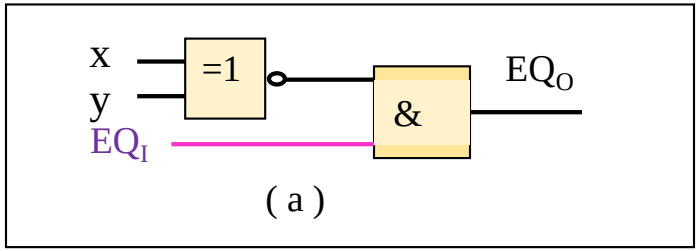


串行电路非常适用于简单、重复运算的逻辑问题，用串行重复电路可以组成串行比较器和串行加法器等。串行电路的重复操作步骤如下：

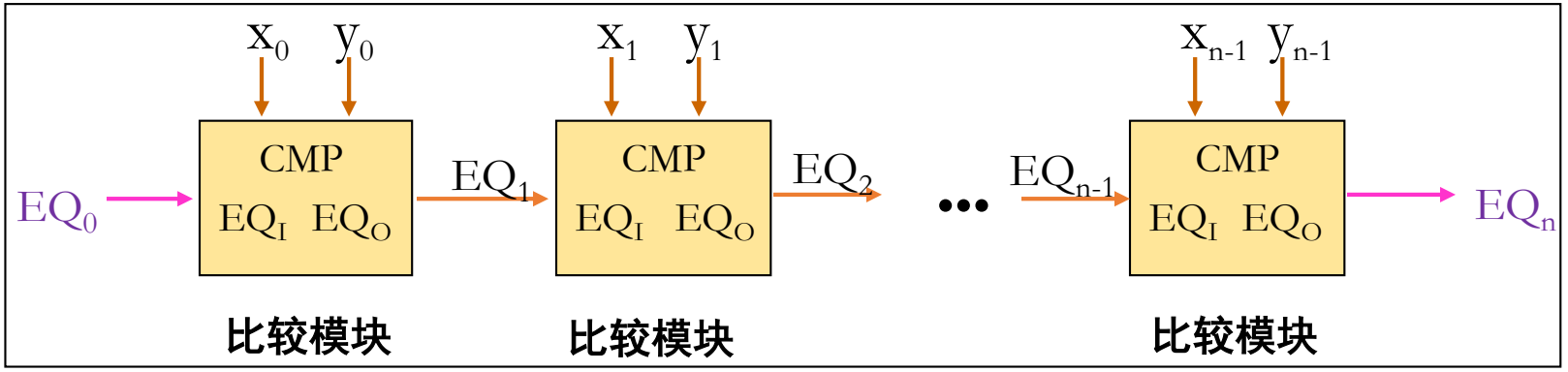
(1) C_0 置初值，并置 i 为0；(2) 用 C_i 和 PI_i 运算得到 PO_i 和 C_{i+1}；(3) $i+1 \rightarrow i$；(4) 如果 i 小于 n，返回步骤(2)。在串行重复电路中，通过提供给各个模块基本输入 $PI_0 \sim PI_n$，利用模块的串行级联完成步骤(2)～(4)的循环。

4 加法器和比较器

例　串行重复比较电路：判断二个 n 位数是否相等的串行比较电路，如图所示：一位串行比较模块参见图(a)



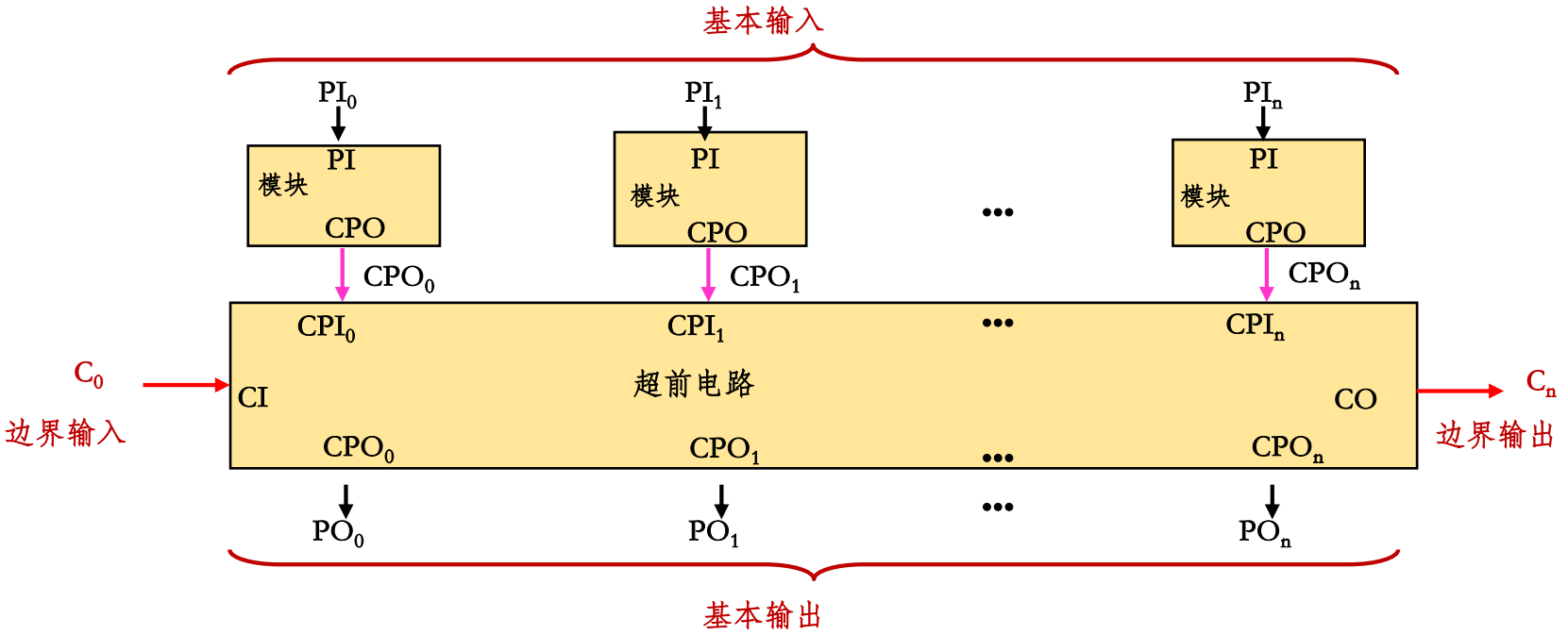
完整的 n 位串行比较电路参见图(b)



(b)

(2) 超前电路

在串行电路中，延迟时间随着位数 n 的增加而增大。为了提高速度，采用超前电路。即各个模块直接产生供超前电路进行运算的中间信号，由超前电路对这些信号同时进行处理，从而产生输出结果。超前电路框图如图所示。



例 超前相等比较器

基本输入：x_i 和 y_i

边界输入：EQ_0

边界输出：EQ_n

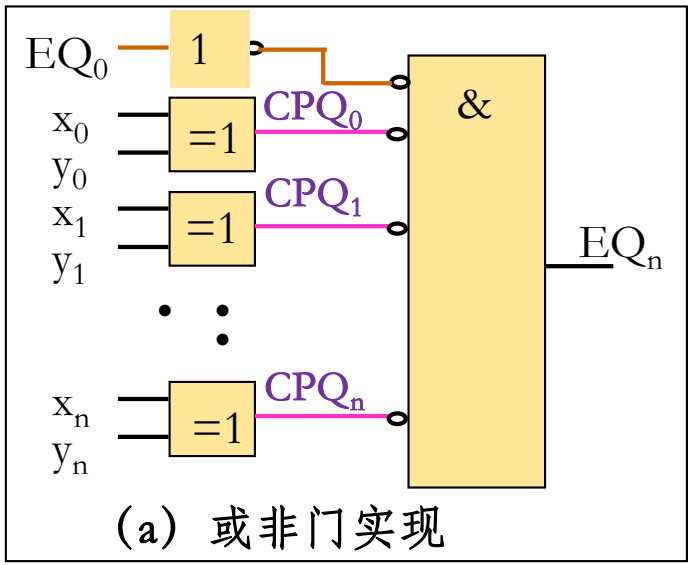
中间变量：CPO_i

$(i = 0,1,\cdots,n)$

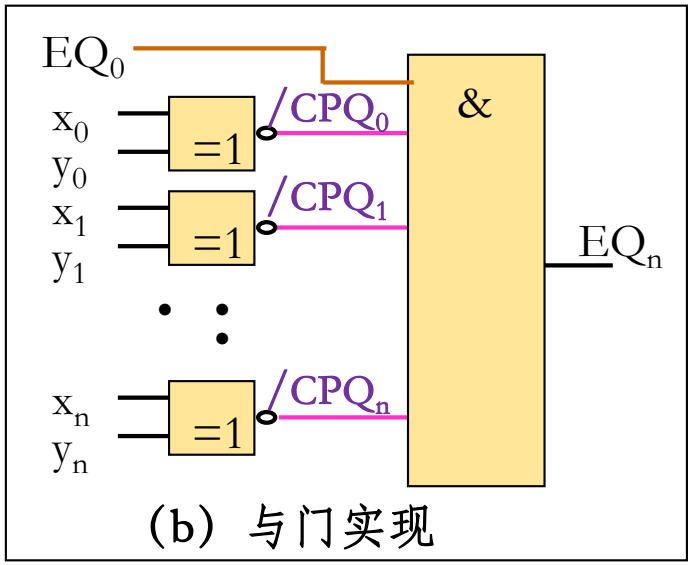
$$CPQ_i = x_i \oplus y_i \qquad i = 0,1,\ldots,n$$

$$EQ_n = EQ_0 \cdot \overline{CPQ_0} \cdot \overline{CPQ_1} \cdot \ldots \cdot \overline{CPQ_n} \quad (b)$$

$$= \overline{\overline{EQ_0} + CPQ_0 + CPQ_1 + \ldots + CPQ_n} \quad (a)$$



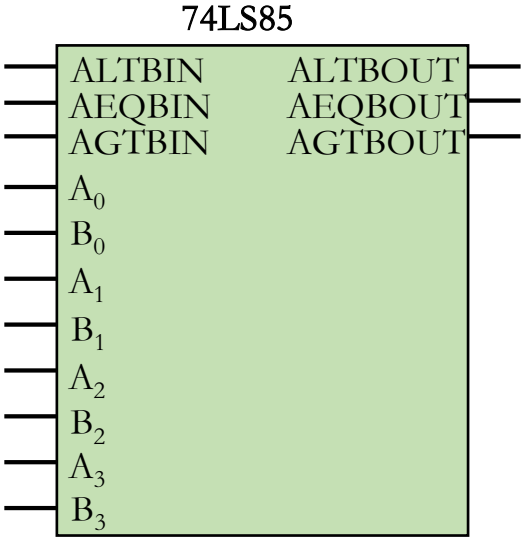
(a) 或非门实现



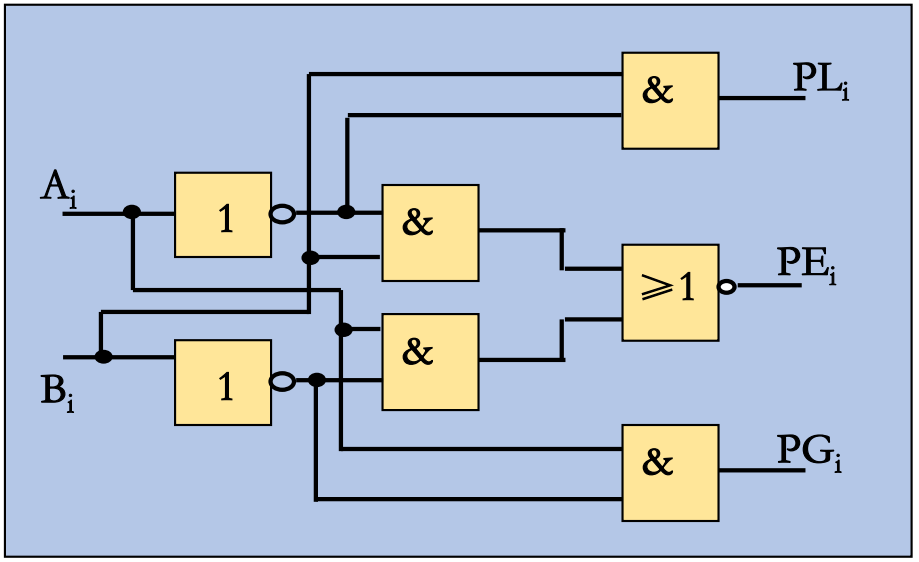
(b) 与门实现

(3) MSI比较器：四位比较器74LS85

① 电路的逻辑符号



② 一个 输入处理模块的逻辑框图



③ 逻辑表达式：四位比较器有四个输入处理模块，产生了12个中间变量：

$PG_0=A_0\overline{B_0}$ ；　$PE_0=\overline{A_0 \oplus B_0}$ ；　$PL_0=\overline{A_0}\,B_0$

$PG_1=A_1\overline{B_1}$ ；　$PE_1=\overline{A_1 \oplus B_1}$ ；　$PL_1=\overline{A_1}\,B_1$

$PG_2=A_2\overline{B_2}$ ；　$PE_2=\overline{A_2 \oplus B_2}$ ；　$PL_2=\overline{A_2}\,B_2$

$PG_3=A_3\overline{B_3}$ ；　$PE_3=\overline{A_3 \oplus B_3}$ ；　$PL_3=\overline{A_3}\,B_3$

4 加法器和比较器

④ 四位比较器的输出逻辑表达式

$$AGTBOUT = (A>B) + (A=B) \cdot AGTBIN$$
$$= PG_3 + PE_3 \cdot PG_2 + PE_3 \cdot PE_2 \cdot PG_1 + PE_3 \cdot PE_2 \cdot PE_1 \cdot PG_0 + PE_3 \cdot PE_2 \cdot PE_1 \cdot PE_0 \cdot AGTBIN$$

$$AEQBOUT = (A=B) \cdot AEQBIN$$
$$= PE_3 \cdot PE_2 \cdot PE_1 \cdot PE_0 \cdot AEQBIN$$

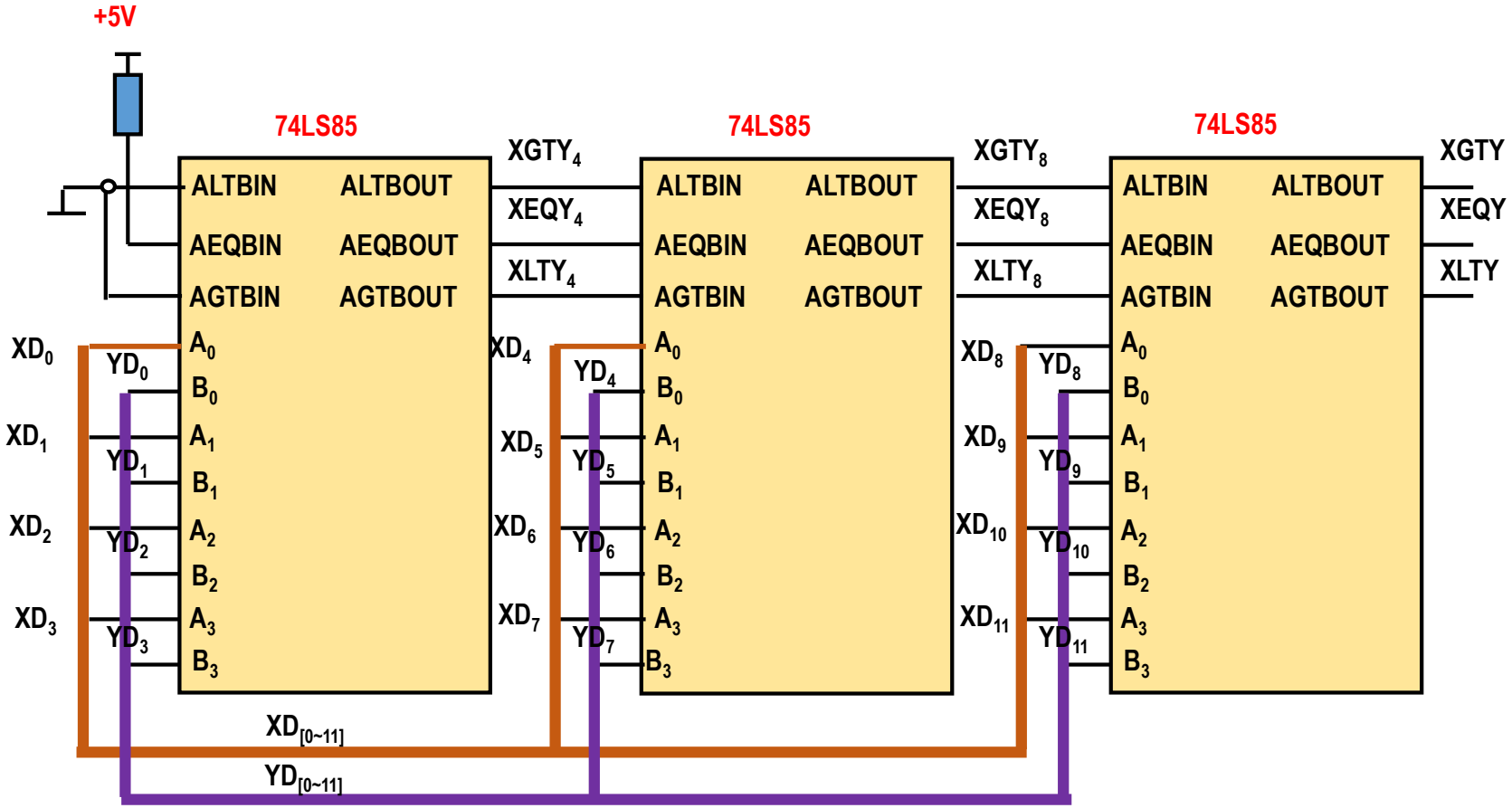
$$ALTBOUT = (A<B) + (A=B) \cdot ALTBIN$$
$$= PL_3 + PE_3 \cdot PL_2 + PE_3 \cdot PE_2 \cdot PL_1 + PE_3 \cdot PE_2 \cdot PE_1 \cdot PL_0 + PE_3 \cdot PE_2 \cdot PE_1 \cdot PE_0 \cdot ALTBIN$$

⑤ 逻辑电路图

74LS85的逻辑符号，参见书P84图2.71

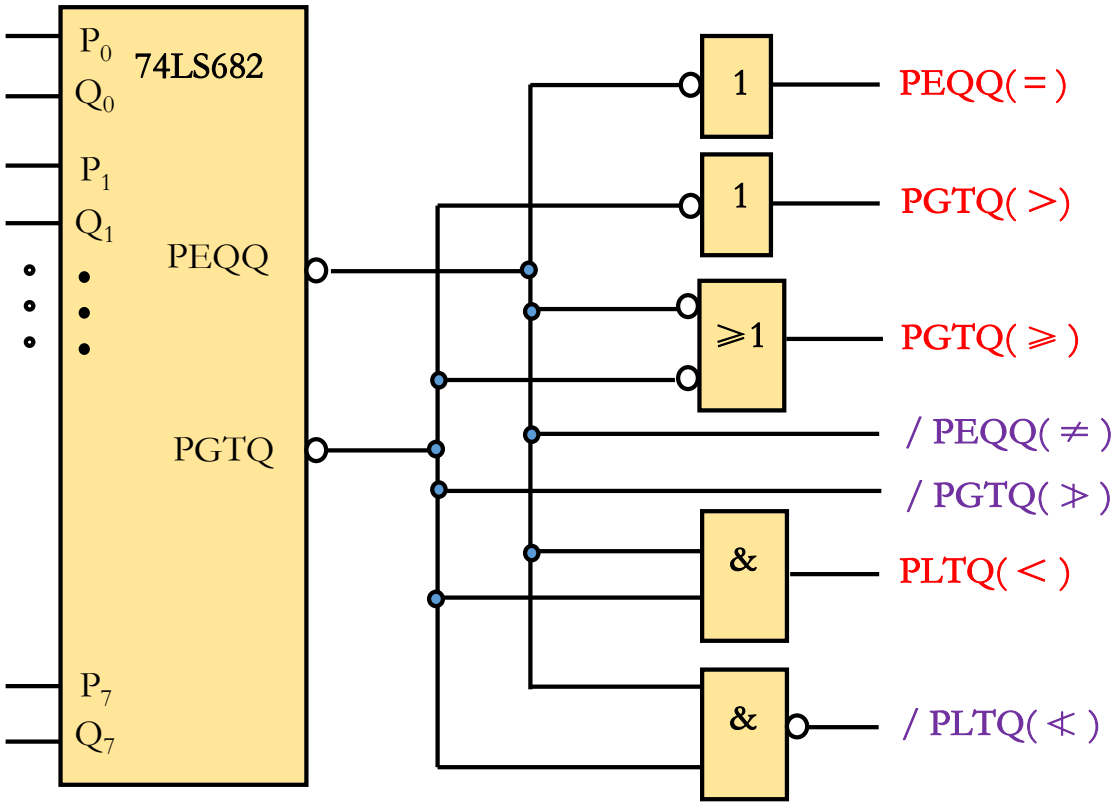
⑥ 74LS85比较器的级联

例 用三个74LS85级联构成12位比较器。



八位比较器74LS682

74LS682有两个低有效输出端：PEQQ(等于) 及PGTQ(大于)逻辑符号及各种条件输出如图所示。



2） 加法器

- 半加器的 HS 和 CO 的逻辑表达式为：

$$HS = x \oplus y$$

$$CO = x \cdot y$$

半加：两个1位二进制数相加时，不考虑低位来的进位的加法

- 全加器的 S 和 Cout 的逻辑表达式为：

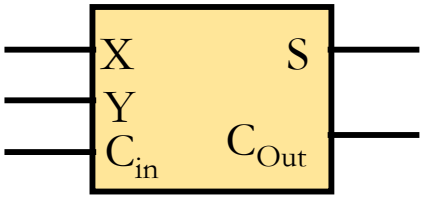
$$S = x \oplus y \oplus C_{in}$$

$$C_{out} = x \cdot y + x \cdot C_{in} + y \cdot C_{in}$$

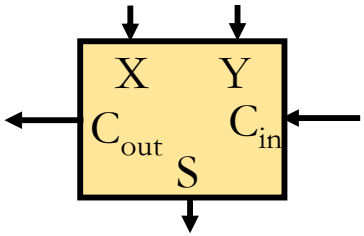
全加：在两个1位二进制数相加时，考虑低位进位的加法

半加器

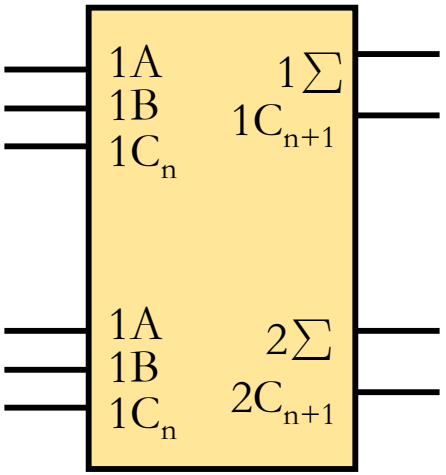
全加器



用于级联时的全加器符号



74LS183



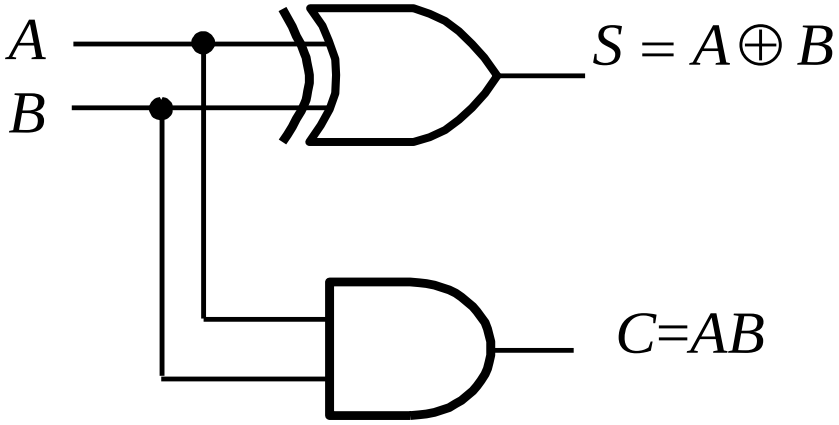
（1） 1位半加器（Half Adder）

不考虑低位进位，将两个1位二进制数A、B相加的器件。

- 半加器的真值表
- 逻辑表达式

$$S=\overline{A}B+A\overline{B}$$

$$C=AB$$



逻辑图

如用与非门实现最少要几个门?

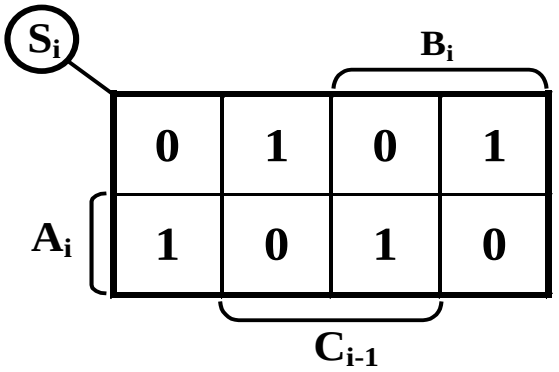
(2) 全加器（Full Adder）

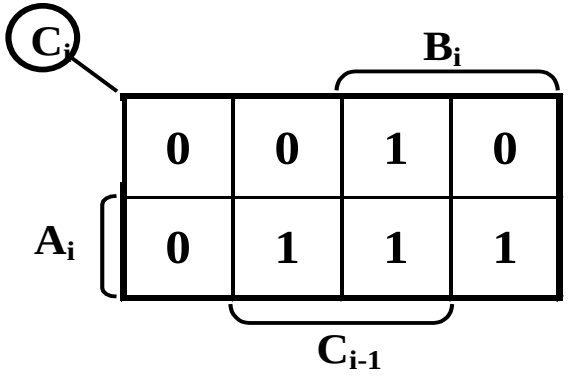
全加器能进行加数、被加数和低位来的进位信号相加，并根据求和结果给出该位的进位信号。

$$S = \overline{A}\overline{B}C_i + \overline{A}B\overline{C}_i + A\overline{B}\overline{C}_i + ABC_i$$
$$= A \oplus B \oplus C_i$$
$$C_o = AB + A\overline{B}C_i + \overline{A}BC_i$$
$$= AB + (A \oplus B)C_i$$

全加器真值表

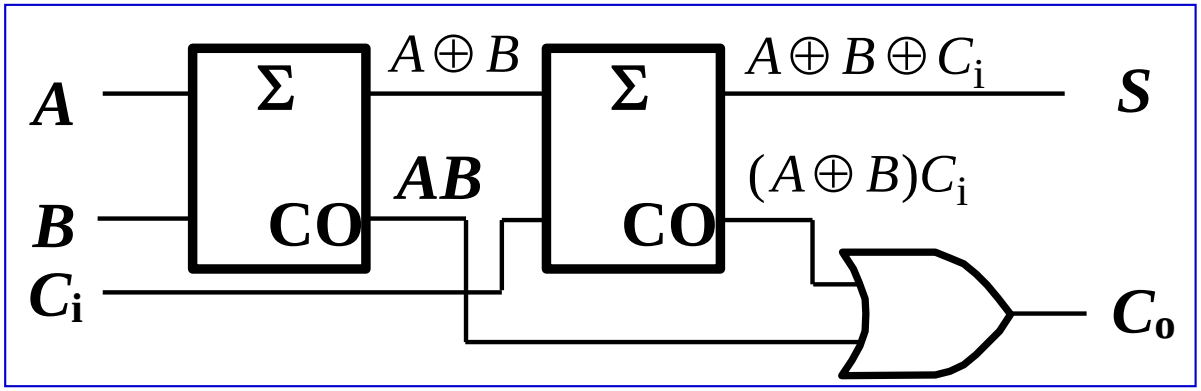
A	B	Ci	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1





4 加法器和比较器

$$S = \overline{A}\,\overline{B}C_{\mathrm{i}} + \overline{A}B\overline{C}_{\mathrm{i}} + A\overline{B}\,\overline{C}_{\mathrm{i}} + ABC_{\mathrm{i}}$$
$$= A \oplus B \oplus C_{\mathrm{i}}$$
$$C_{\mathrm{o}} = AB + A\overline{B}C_{\mathrm{i}} + \overline{A}BC_{\mathrm{i}}$$
$$= AB + (A \oplus B)C_{\mathrm{i}}$$



- 你能用74151\74138设计全加器吗?
- 用这两种器件组成逻辑函数产生电路,有什么不同?

加法器的应用

全加器真值表

A	*B*	*Ci*	*S*	*Co*
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

*ABC*有奇数个1时S为1；

*ABC*有偶数个1和全为0时S为0。

-----用全加器组成三位二进制代码奇偶校验器

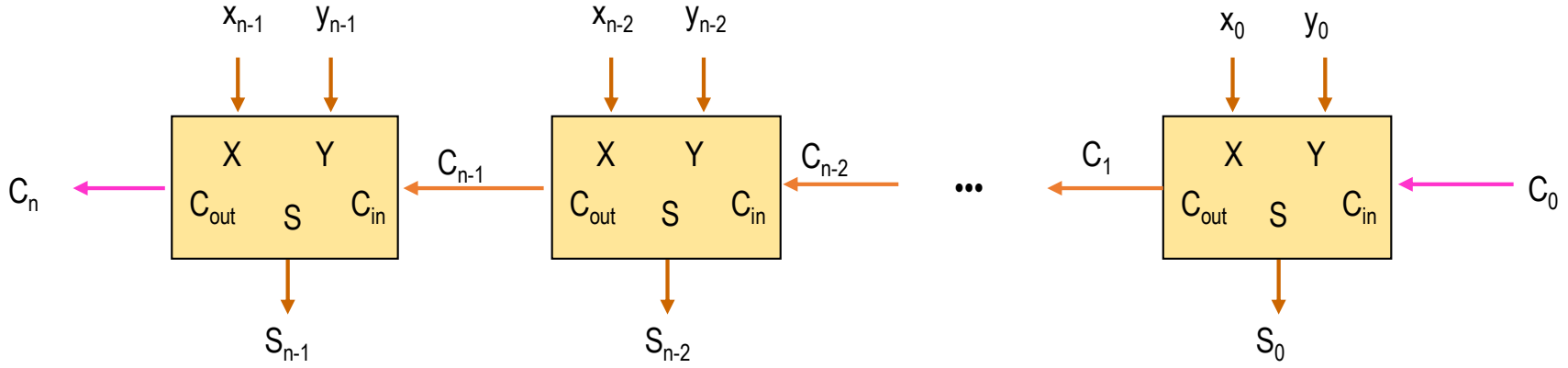
用全加器组成八位二进制代码奇偶校验器，电路应如何连接？

（1）并行加法器（行波加法器）

- n 个全加器级联，每个全加器处理两个一位二进制数，则可以构成两个 n 位二进制数相加的加法器。由于进位信号是一级一级地由低位向高位逐位产生，故又称为行波加法器。
- 由于进位信号逐位产生，这种加法器速度很低。最坏的情况是进位从最低位传送至最高位。行波加法器的最大运算时间为：

$$T_{ADD} = T_{XYCOUT} + (n-2) \cdot T_{CINCOUT} + T_{CINS}$$

其中：T_{XYCOUT} 是最低位全加器中由 x 和 y 产生进位 C_{out} 的延迟时间，$T_{CINCOUT}$ 是中间位全加器中由 C_{in} 产生 C_{out} 的延迟时间，T_{CINS} 是最高位全加器中由 C_{in} 产生 S 的延迟时间。

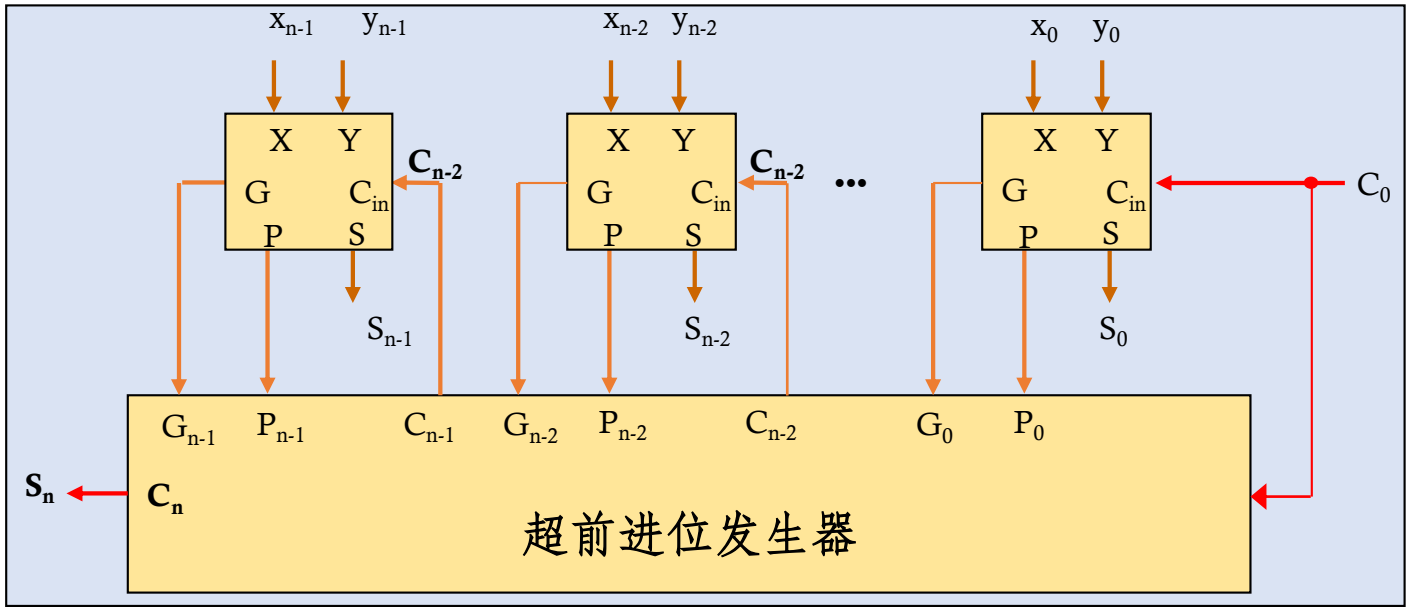


(2) 超前进位加法器

超前进位加法器输入处理模块的逻辑表达式为：

$S = x \oplus y \oplus C_{in} = P \oplus C_{in}$ ， $G = x \cdot y$ ， $P = x \oplus y$

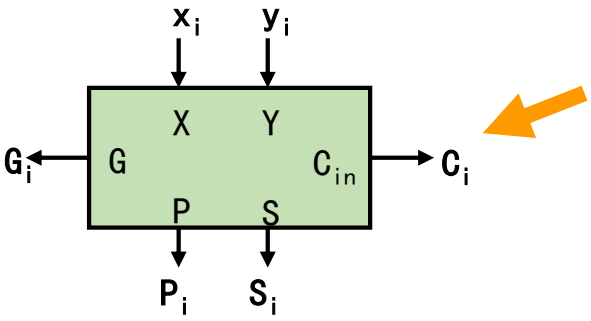
$C_{out} = x \cdot y + (x \oplus y) C_{in} = G + P \cdot C_{in}$

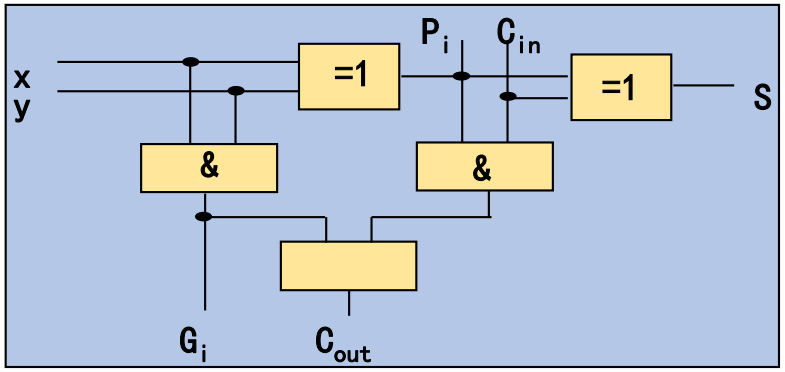


超前进位加法器的结构框图

4 加法器和比较器

可以由二个半加器组成的一个全加器构成输入处理模块，如下：





例 三位二进制加法的进位输出可写成：

$C_1 = G_0 + P_0 \cdot C_0$,

$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 \cdot (G_0 + P_0 \cdot C_0) = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0$

$C_3 = G_2 + P_2 \cdot C_2 = G_2 + P_2 \cdot (G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0)$

$= G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$

特点：（1）所有进位都是同时产生的，故电路延时时间与位数多少无关。

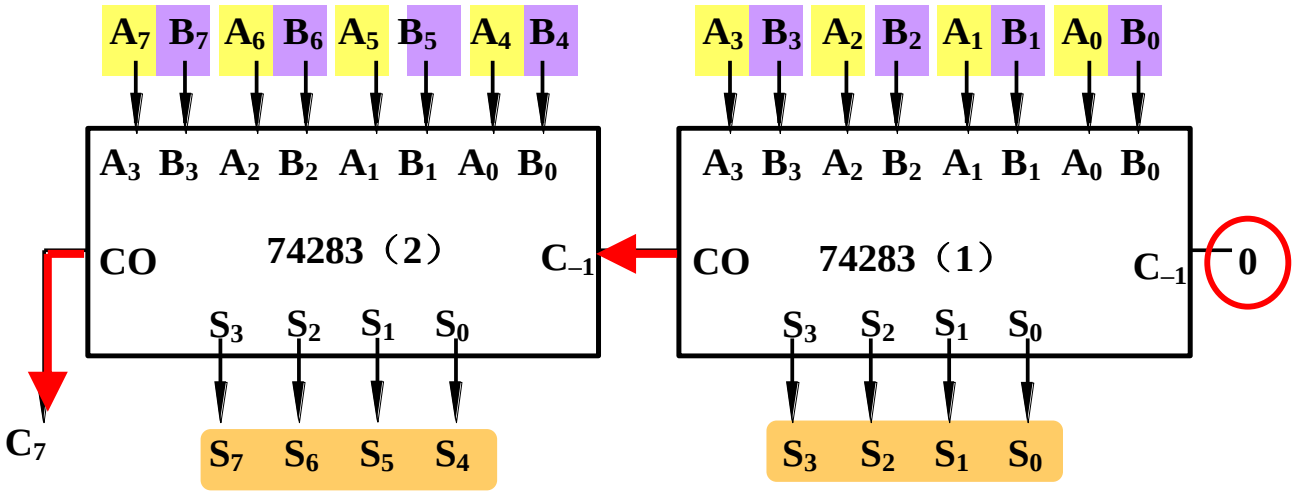
（2）在位数较多时其运算速度比行波加法器的要快得多。

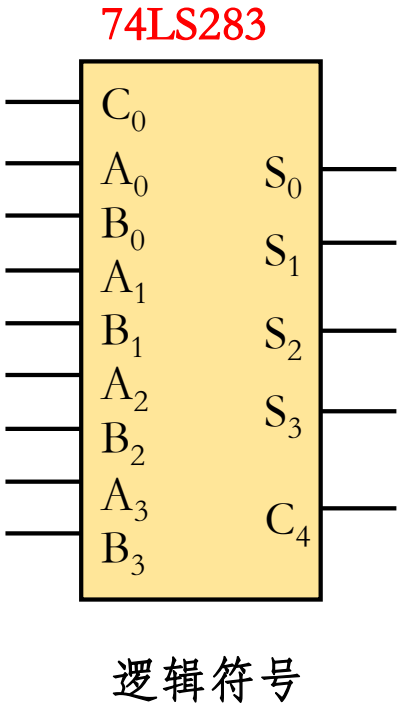
逻辑图参见书P87，图2.78。

（3） MSI加法器

74LS283是一个快速进位四位二进制加法器。用74LS283级联构成 n 位加法器，级联方式为：片内（4位）超前进位，片间为串行进位。

例1. 用两片74LS283构成一个8位二进制数加法器。

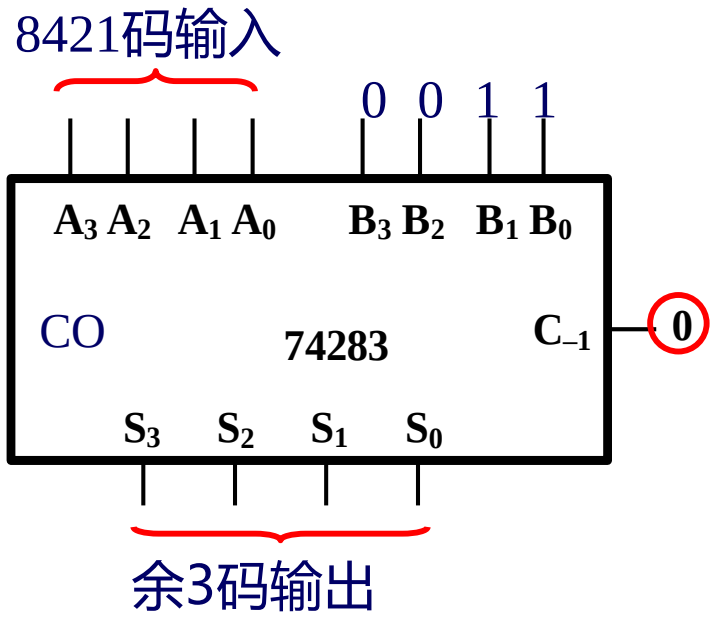




逻辑符号

例. 用74283构成将8421BCD码转换为余3码的码制转换电路 。

8421码		余3码
0000	+0011	0011
0001	+0011	0100
0010	+0011	0101
⋮		⋮



减法运算

若n位二进制的原码为$N_{原}$，则与它相对应的2 的补码为

$$N_{补}=2^N-N_{原}$$

补码与反码的关系式

$$N_{补}=N_{反}+1$$

设两个数A、B相减，利用以上两式，可得

$$A-B=A+B_{补}-2^n=A+B_{反}+1-2^n$$

在实际应用中，通常是将减法运算变为加法运算来处理，即采用加补码的方法完成减法运算。

4 加法器和比较器

1）$A-B \geq 0$的情况。

A=0101 ， B=0010

	0	**1**	**0**	**1**	***A***
	1	**1**	**0**	**1**	***B*** 反
+				**1**	
1	0	0	1	1	

舍弃

2）$A-B < 0$的情况。

A= 0010 ， B=0101

	0	**0**	**1**	**0**	***A***
	1	**0**	**1**	**0**	***B*** 反
+				**1**	
0	1	1	0	1	

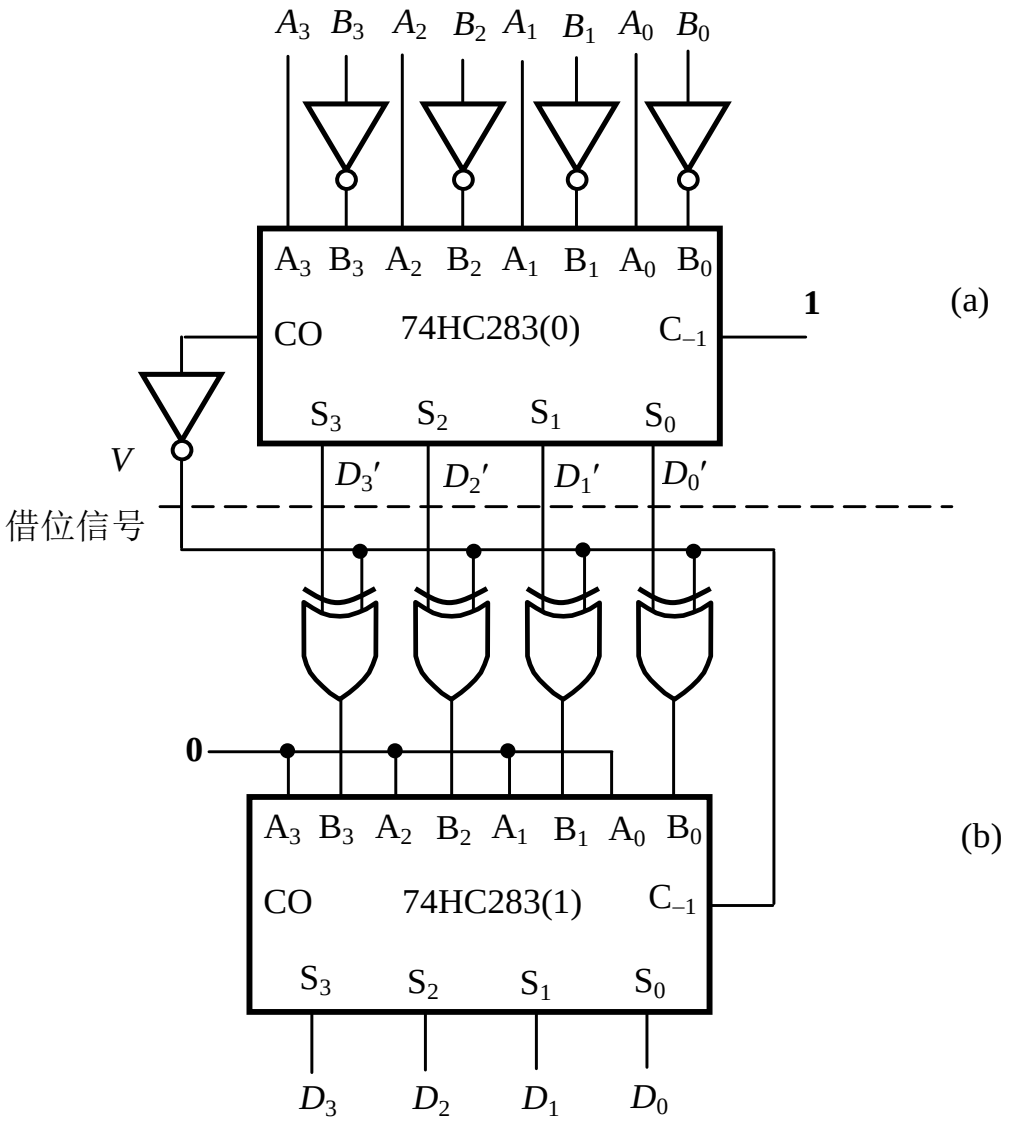
舍弃

当$A-B \geq 0$时，舍弃的进位为1，所得 结果就是差的原码，不需再求反补。

当$A-B < 0$时，舍弃的进位为0，所得结果是补码，要得到原码需再求补。

输出为原码的4位减法运算逻辑图

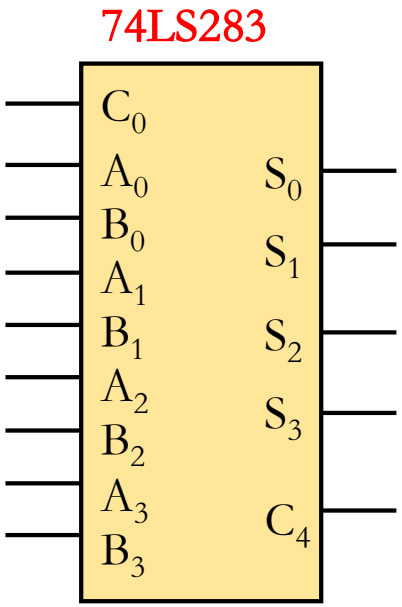
$$
\begin{array}{cccccc}
 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \boldsymbol{A} \\
 & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \boldsymbol{B}_{反} \\
+ & & & & \mathbf{1} & \\
\hline
1 & 0 & 0 & 1 & 1 &
\end{array}
$$



4 加法器和比较器

例1 用MSI四位二进制加法器实现两个一位十进制8421BCD码的加法器。

- 一位8421BCD码所能表示的值是0~9；
- 两个8421BCD码相加所得和的范围是0~18；
- 如果其和的范围在0~9之间，则不需要进行校正；
- 如果其和在10~18之间，则必须进行校正。参见下页真值表所示。



4 加法器和比较器

十进制数	未校正BCD码和 $C_4\ S_3\ S_2\ S_1\ S_0$	校正的BCD码和 $C_4'\ S_3'\ S_2'\ S_1'\ S_0'$	十进制数	未校正BCD码和 $C_4\ S_3\ S_2\ S_1\ S_0$	校正的BCD码和 $C_4'\ S_3'\ S_2'\ S_1'\ S_0'$
0	0 0 0 0	0 0 0 0	10	1 0 1 0	1 0 0 0 0
1	0 0 0 1	0 0 0 1	11	1 0 1 1	1 0 0 0 1
2	0 0 1 0	0 0 1 0	12	1 1 0 0	1 0 0 1 0
3	0 0 1 1	0 0 1 1	13	1 1 0 1	1 0 0 1 1
4	0 1 0 0	0 1 0 0	14	1 1 1 0	1 0 1 0 0
5	0 1 0 1	0 1 0 1	15	1 1 1 1	1 0 1 0 1
6	0 1 1 0	0 1 1 0	16	1 0 0 0 0	1 0 1 1 0
7	0 1 1 1	0 1 1 1	17	1 0 0 0 1	1 0 1 1 1
8	1 0 0 0	1 0 0 0	18	1 0 0 1 0	1 1 0 0 0
9	1 0 0 1	1 0 0 1			

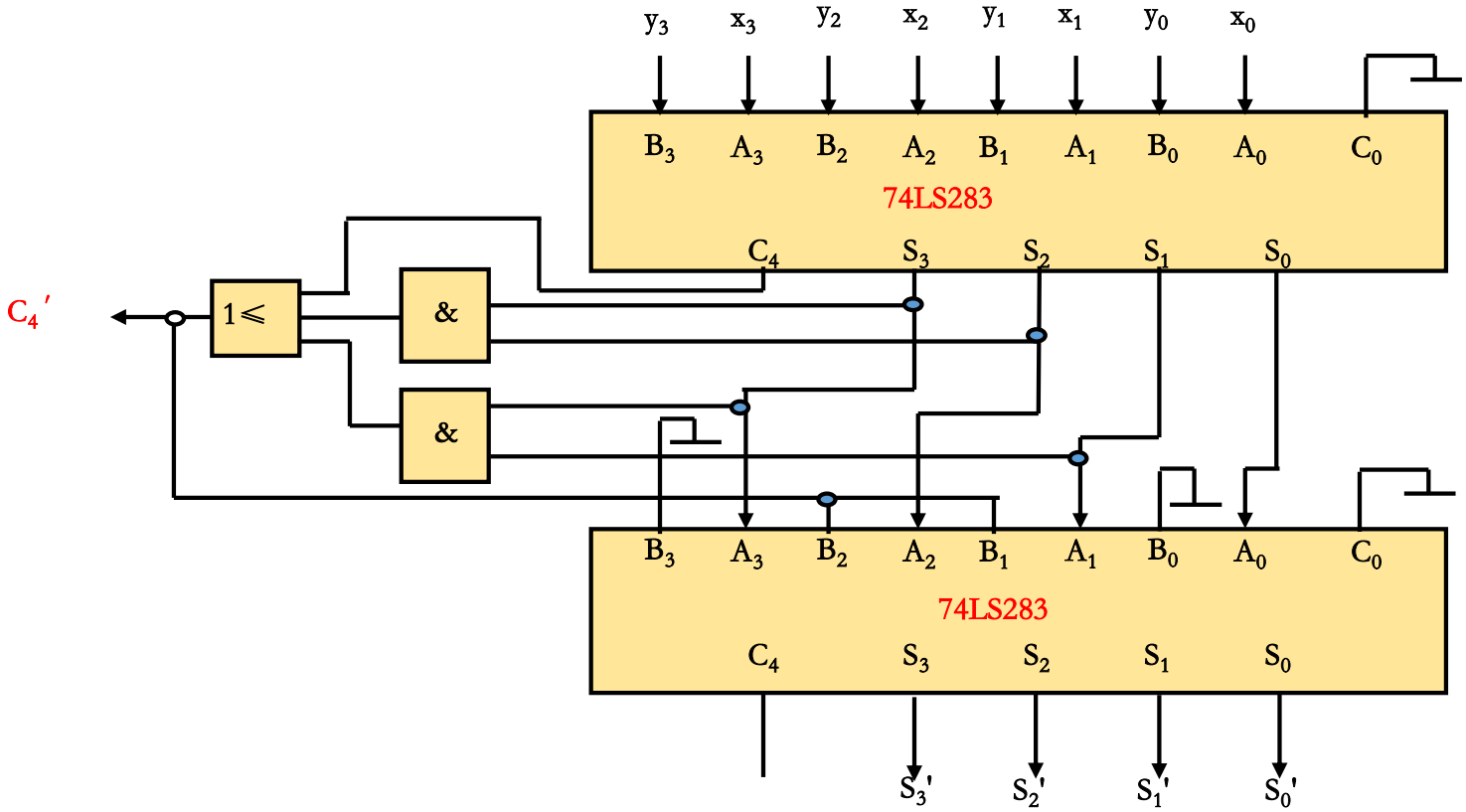
$$C_4' = S_3\overline{S_2}S_1\overline{S_0} + S_3\overline{S_2}S_1S_0 + S_3S_2\overline{S_1}\,\overline{S_0} + S_3S_2\overline{S_1}S_0 + S_3S_2S_1\overline{S_0} + S_3S_2S_1S_0 + C_4$$

$$= S_3S_1 + S_3S_2 + C_4$$

4 加法器和比较器

两个一位8421码加法器的逻辑图

当和需要校正(即$C_4'=1$)时，则需作+0110(+6)校正。



下一节内容：

时序电路的分析与设计