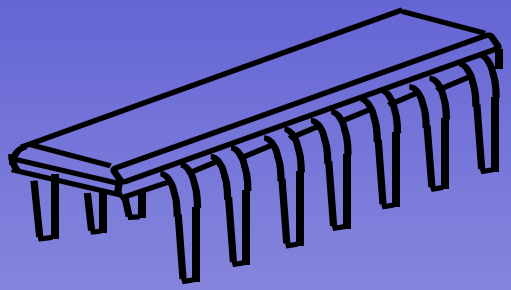
第4章 组合逻辑功能器件



4.1 数据选择器

4.2 译码器/数据分配器

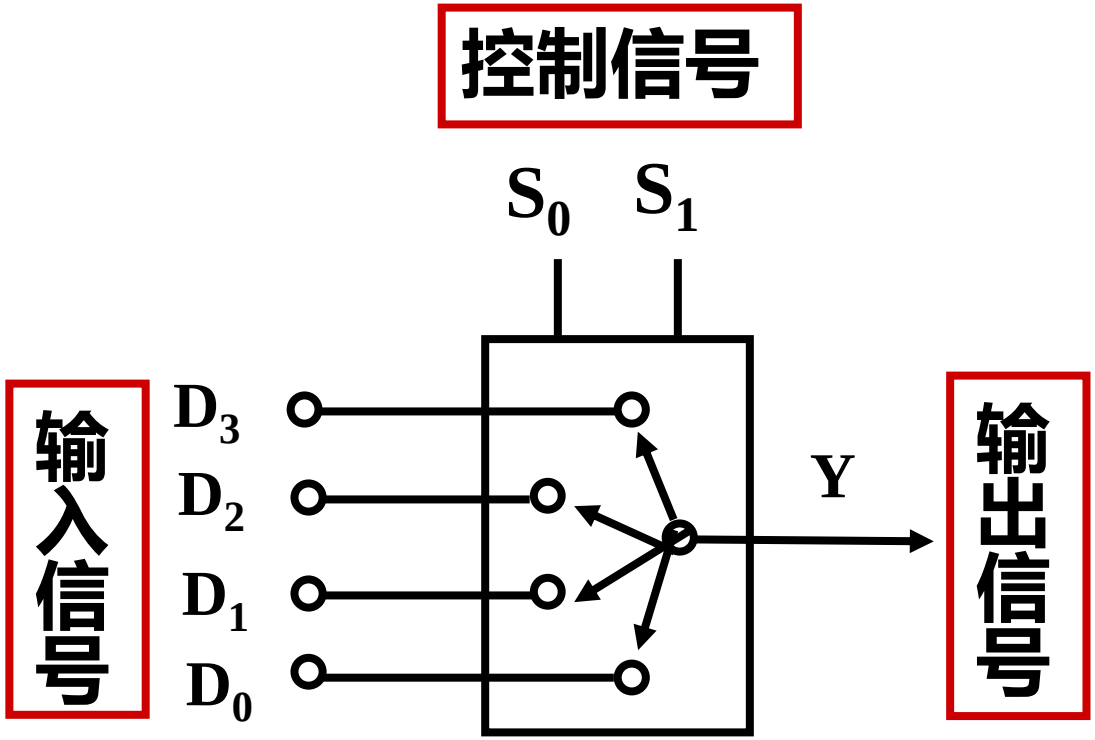
4.3 编码器

4.4 数值比较器

4.5 加法器

4.1 数据选择器（MUX）

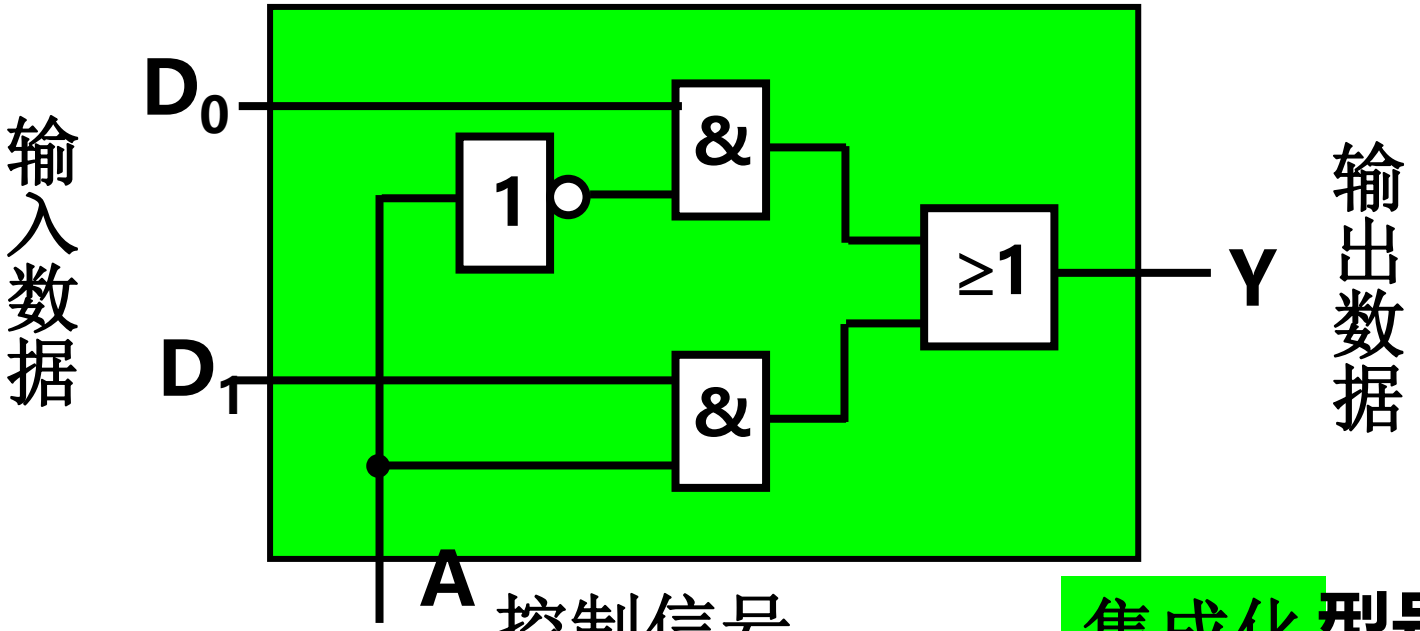
一、功能：从多路数据中选择一路信号进行传输的电路，称为数据选择器。



数据选择器类似一个多投开关。选择哪一路信号由相应的一组控制信号控制。

二、结构及符号：

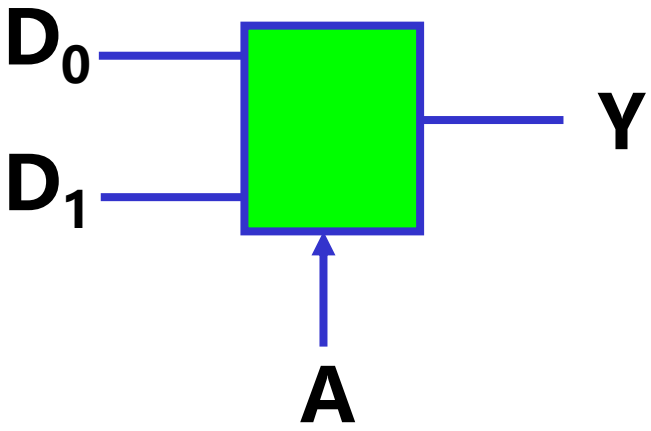
1、2选1数据选择器



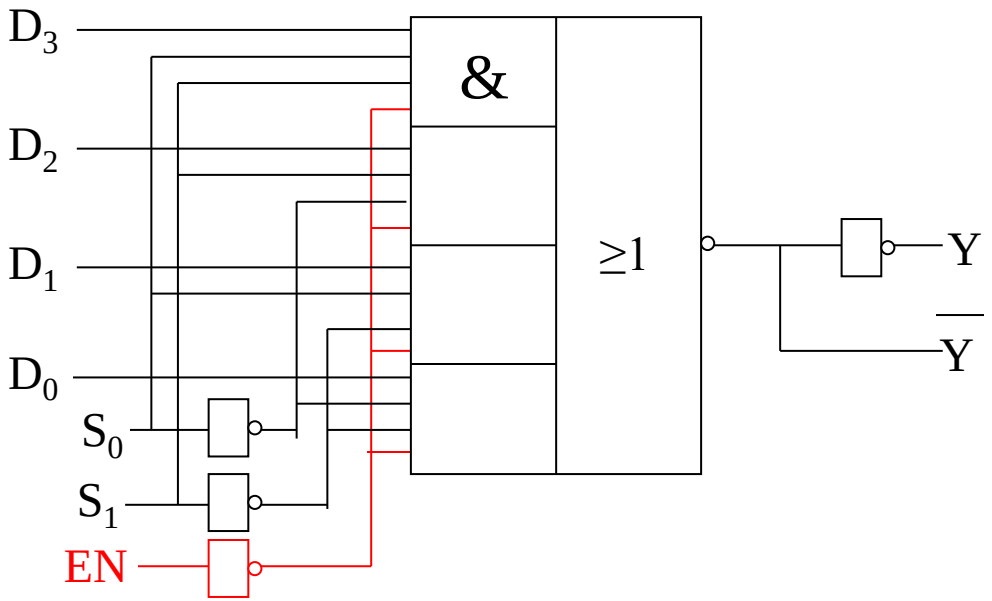
$$Y = \bar{A}D_0 + AD_1$$

A	Y
0	D_0
1	D_1

集成化型号:74LS157



2、4选1数据选择器(集成电路型号:74LS153)



D_0、D_1、D_2、D_3——数据输入端，数据选择器通常按数据输入端数命名，常用的有：四路选择器、八路选择器、十六路选择器。

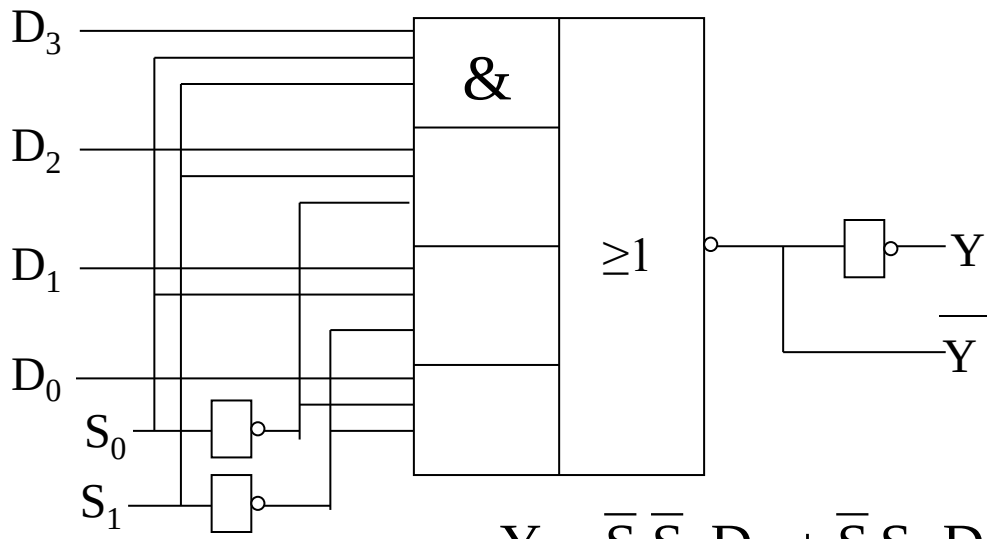
S_0、S_1——地址输入端。（选择控制端）

EN——使能端（控制端，允许端）；

EN=1时，禁止数据选通（不工作），

EN=0时，选择器工作。

2、4选1数据选择器



选择变量 S_1S_0	数据输入 D	输出 Y
0 0	D_0	D_0
0 1	D_1	D_1
1 0	D_2	D_2
1 1	D_3	D_3

$$Y = \overline{S}_1\overline{S}_O D_0 + \overline{S}_1 S_O D_1 + S_1\overline{S}_O D_2 + S_1 S_O D_3 = \sum_{i=0}^{3} m_i D_i$$

其中：m_i为选择变量S_1、S_0的四个最小项；

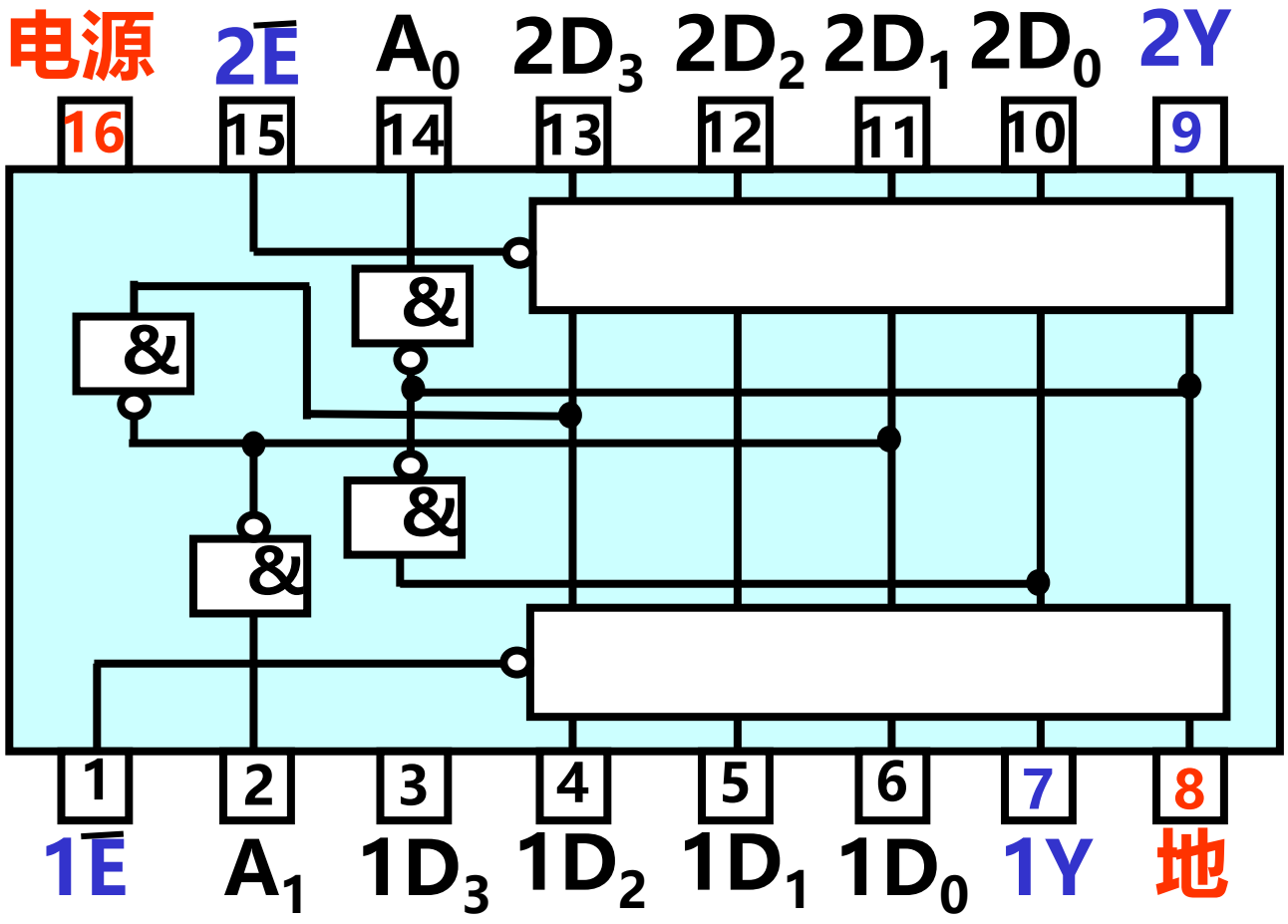
D_i为四路选择器的四个输入数据

对于2^n路选择器，它应有n个地址输入端（设为S_0，S_1，…，S_{n-1}），2^n个数据输入端（设为D_0，D_2，…，D_{2^n-1}），则其输出函数为：

$$Y = \sum_{i=0}^{2^n-1} m_i D_i$$

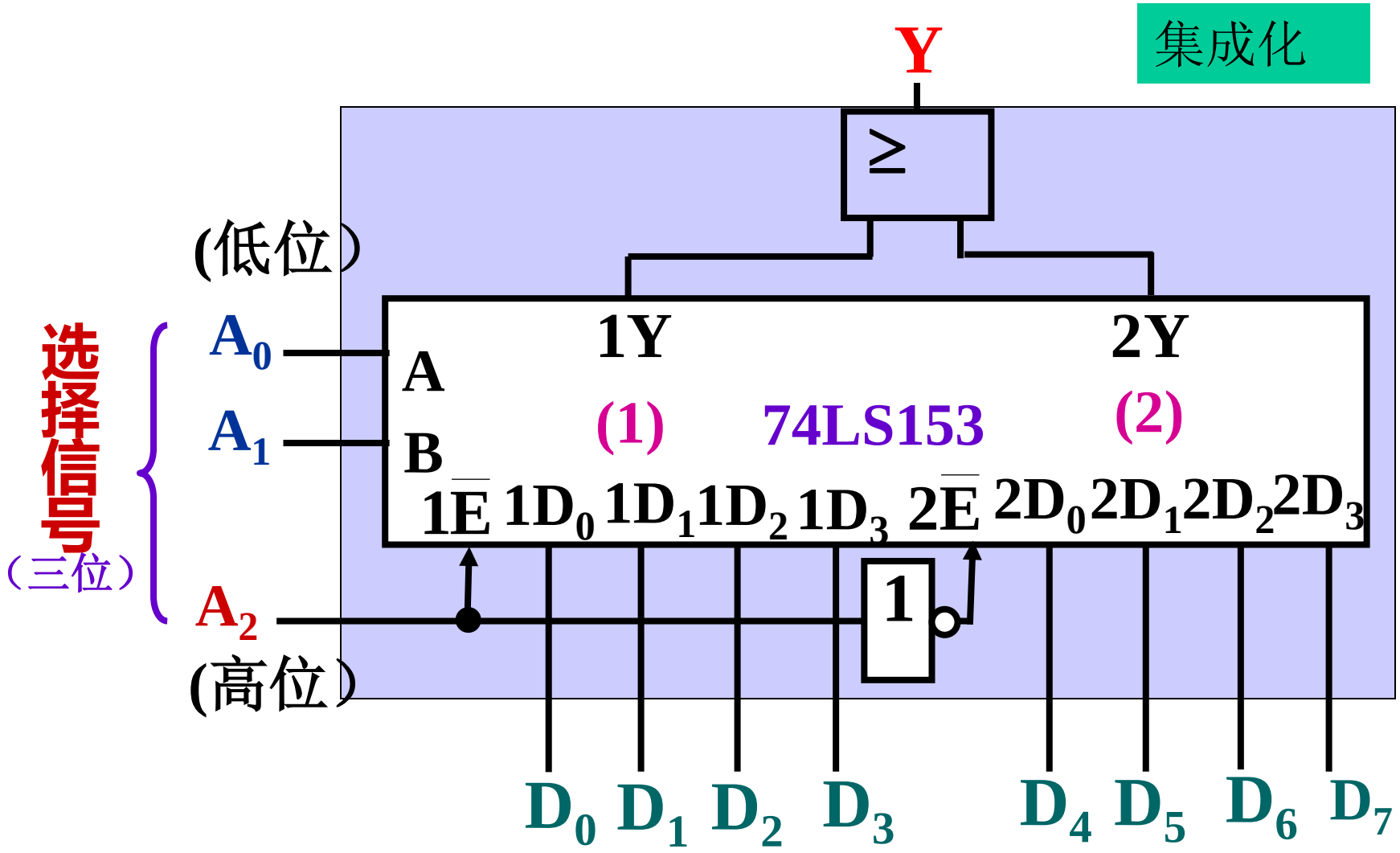
3、TTL集成电路：双4选1数据选择器

型号:74LS153（国产T1153--T4153)

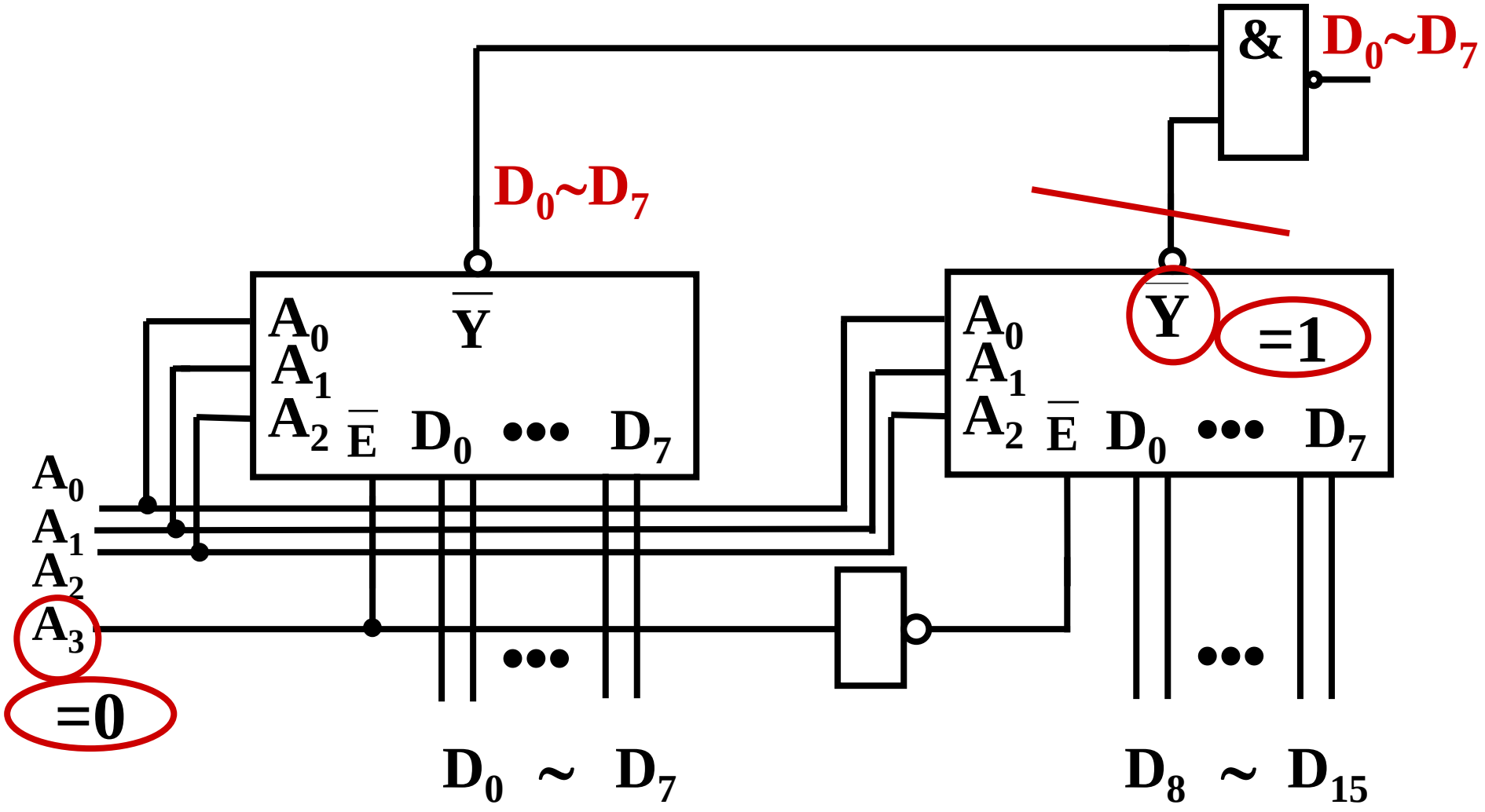


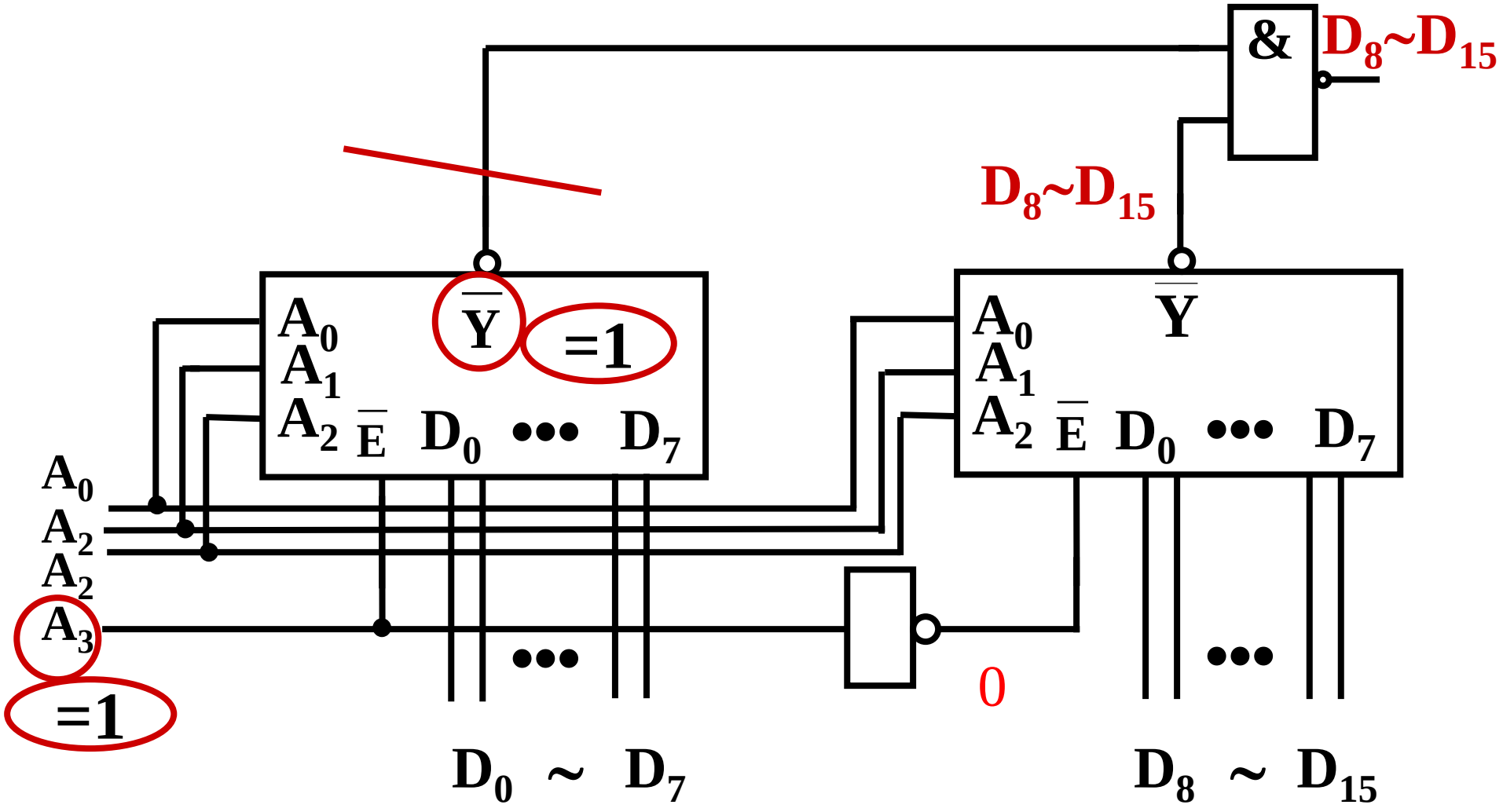
输入			输出
A_0	A_1	$\overline{E}$	Y
Φ	Φ	1	0
0	0	0	D_0
0	1	0	D_1
1	0	0	D_2
1	1	0	D_3

例：用一片74LS153组成8选1: A_2=0:(1)工作；
A_2=1:(2)工作。



例：用两片74LS151构成十六选一数据选择器




&
D8~D15
D8~D15
A0
A1
A2 E
D0 ••• D7
Y
=1
A0
A1
A2 E
D0 ••• D7
Y
A0
A2
A2
A3
=1
0
D0 ~ D7
D8 ~ D15

三、用数据选择器设计逻辑电路

例：用多路选择器实现逻辑函数：$F(A,B,C)=\sum m(2,3,5,6)$

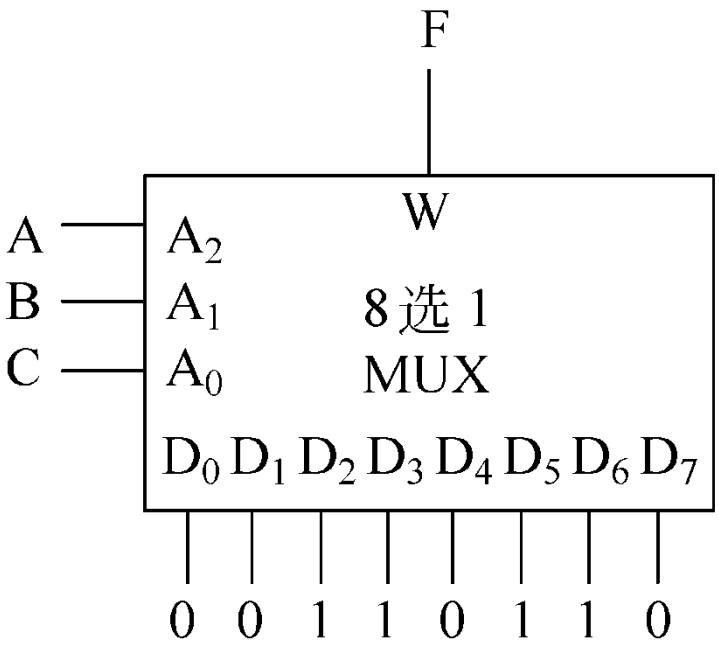
- 解：根据多路选择器输出表达式的特点，可采用两种不同规模的MUX实现给定函数的功能。
- 方案一：采用8路数据选择器实现。
- 对比8路选择器的输出表达式和函数W的表达式

$$W=\overline{A_2}\,\overline{A_1}\,\overline{A_0}\cdot D_0+\overline{A_2}\,\overline{A_1}A_0\cdot D_1+\overline{A_2}A_1\overline{A_0}\cdot D_2+\overline{A_2}A_1A_0\cdot D_3$$
$$+A_2\overline{A_1}\,\overline{A_0}\cdot D_4+A_2\overline{A_1}A_0\cdot D_5+A_2A_1\overline{A_0}\cdot D_6+A_2A_1A_0\cdot D_7$$

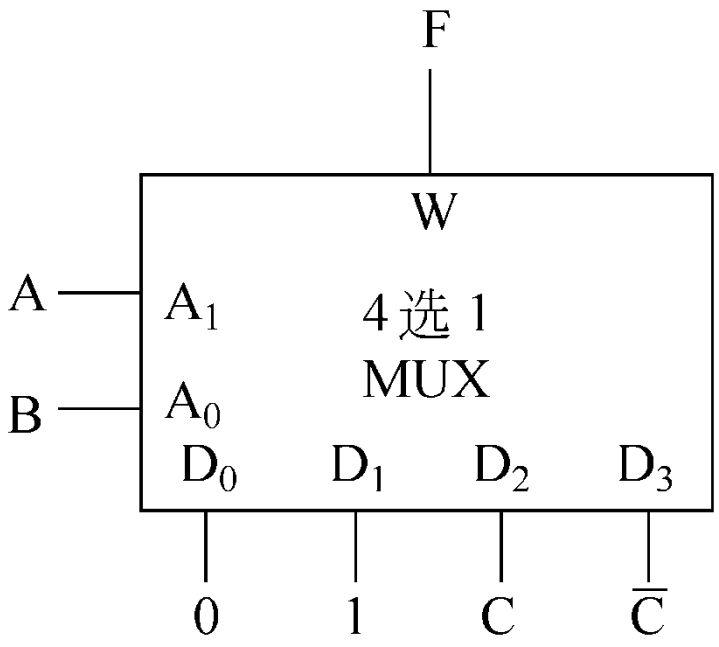
$$F(A,B,C)=\overline{A}B\overline{C}+\overline{A}BC+A\overline{B}C+AB\overline{C}$$

- 可知，要使W=F，只需令$A_2=A$，$A_1=B$，$A_0=C$且$D_0=D_1=D_4=D_7=0$，而$D_2=D_3=D_5=D_6=1$即可。如图（a）所示

$$F(A,B,C)=\overline{A}B\overline{C}+\overline{A}BC+A\overline{B}C+AB\overline{C}$$



(a)



(b)

- 方案二：采用4路数据选择器实现
- 首先从函数的3个输入变量中任意选择两个与4路MUX的两个地址选择端相连。假定选择A、B与地址端A_1、A_0相连，则可将函数F变换如下：

$$F(A,B,C)=\overline{A}B\overline{C}+\overline{A}BC+A\overline{B}C+AB\overline{C}$$
$$=\overline{A}\,\overline{B}\cdot 0+\overline{A}B(\overline{C}+C)+A\overline{B}\cdot C+AB\cdot\overline{C}$$
$$=\overline{A}\,\overline{B}\cdot 0+\overline{A}B\cdot 1+A\overline{B}\cdot C+AB\cdot\overline{C}$$

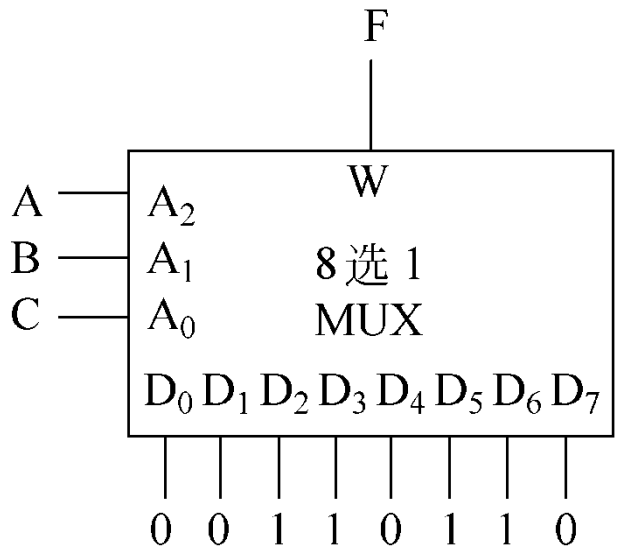
$$Y=\overline{S}_1\overline{S}_O D_0+\overline{S}_1 S_O D_1+S_1\overline{S}_O D_2+S_1 S_O D_3=\sum_{i=0}^{3} m_i D_i$$

- 将该式与4路MUX的输出W对比后可知，要使W与F相等，只需$D_0=0$，$D_1=1$，$D_2=C$，$D_3=\overline{C}$，如图（b）所示。

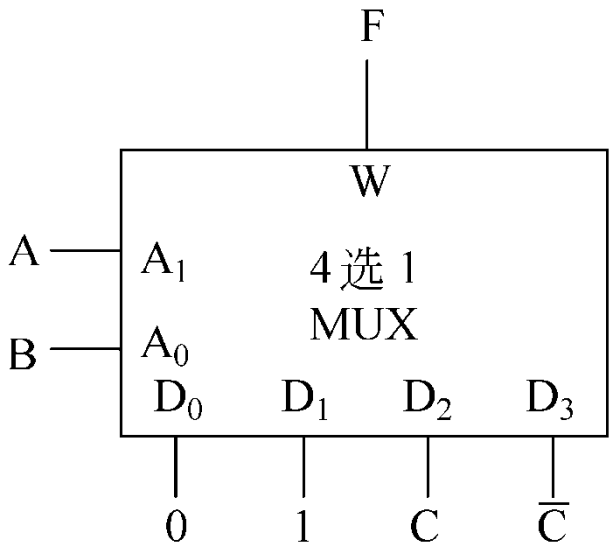
$$F(A,B,C) = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + AB\overline{C}$$

$$= \overline{A}\overline{B}\cdot 0 + \overline{A}B(\overline{C}+C) + A\overline{B}\cdot C + AB\cdot\overline{C}$$

$$= \overline{A}\overline{B}\cdot 0 + \overline{A}B\cdot 1 + A\overline{B}\cdot C + AB\cdot\overline{C}$$



(a)



(b)

- 方案一给出了用2^n路MUX实现n个变量函数的一般方法：将函数的n个变量依次连接到MUX的n个地址选择输入端，并将函数表达式表示成“最小项之和”的形式。若函数表达式中包含最小项m_i，则将MUX相应的D_i接1，否则D_i接0。该方法简单，但不经济，因为MUX的数据输入端未能得到充分利用。
- 方案二给出了用2^{n-1}路MUX实现含有n个变量的逻辑函数功能的一般方法：从函数的n个变量中任意选择n-1个作为地址选择输入端，并根据所选定的地址输入端将函数变换成$F=\sum_{i=0}^{2^{n-1}-1} m_i D_i$ 的形式，以确定各数据输入端D_i的取值。假定剩余变量为X，则D_i的取值只可能是0，1，X或 $\overline{X}$ 之一。

4.2 译码器

译码：将某二进制编码翻译成电路的某种状态

一、二进制译码器

二进制译码器的作用：将n个输入的不同组合译成2^n种电路状态。也叫n---2^n线译码器。例如：计算机中的地址译码电路

译码器的输入——一组二进制代码

译码器的输出——一组高低电平信号

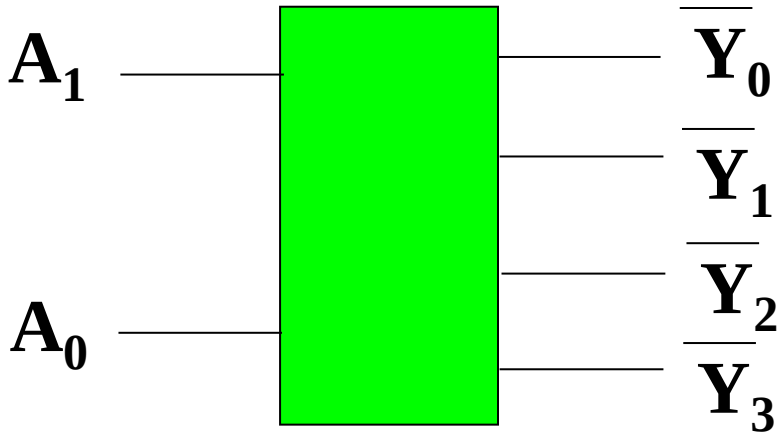
常用类型：

2线— 4线译码器　　型号：74LS139

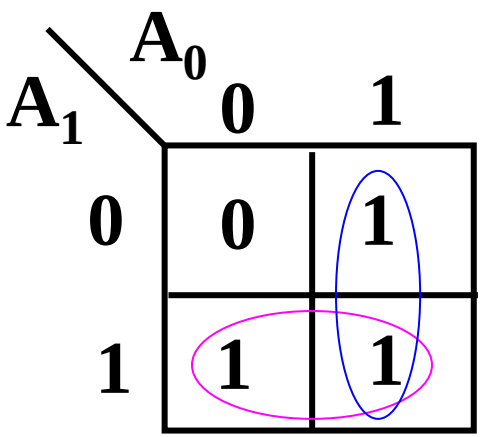
3 线— 8线译码器　　型号：74LS138

4 线— 16线译码器　　型号：74LS154

1、 2 线— 4线译码器



画关于 $\overline{Y}_0$ 的卡诺图



真值表

A_1	A_0	$\overline{Y}_3$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

$$\overline{Y}_0 = A_1 + A_0 = \overline{\overline{A}_1 \overline{A}_0}$$

$$\overline{Y}_1 = \overline{A}_1 + A_0 = \overline{\overline{A}_1 A_0}$$

$$\overline{Y}_2 = A_1 + \overline{A}_0 = \overline{A_1 \overline{A}_0}$$

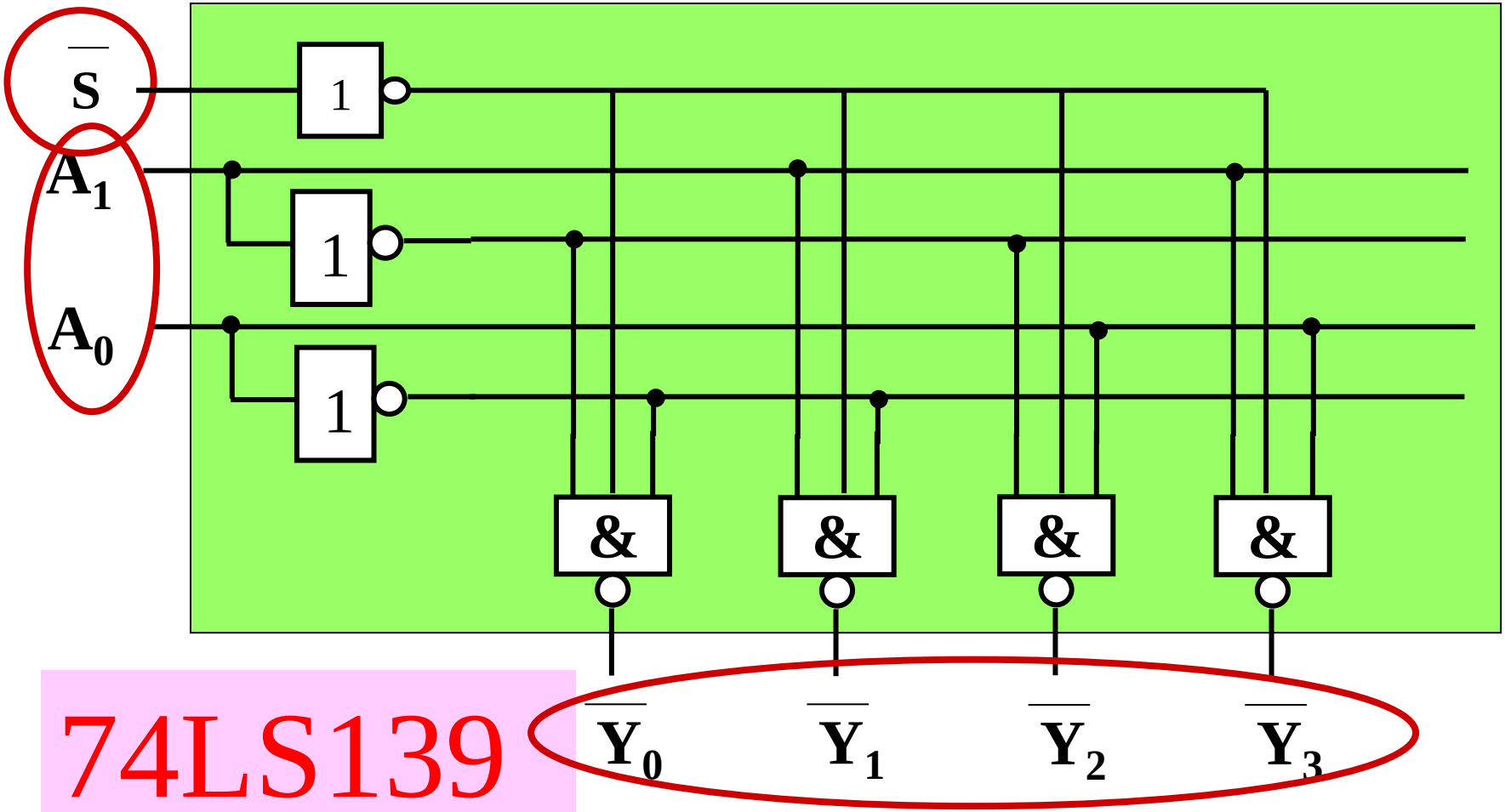
$$\overline{Y}_3 = \overline{A}_1 + \overline{A}_0 = \overline{A_1 A_0}$$

$$\overline{Y}_0 = A_1 + A_0 = \overline{\overline{A}_1 \overline{A}_0}$$

$$\overline{Y}_1 = \overline{A}_1 + A_0 = \overline{A_1 \overline{A}_0}$$

$$\overline{Y}_2 = A_1 + \overline{A}_0 = \overline{\overline{A}_1 A_0}$$

$$\overline{Y}_3 = \overline{A}_1 + \overline{A}_0 = \overline{A_1 A_0}$$

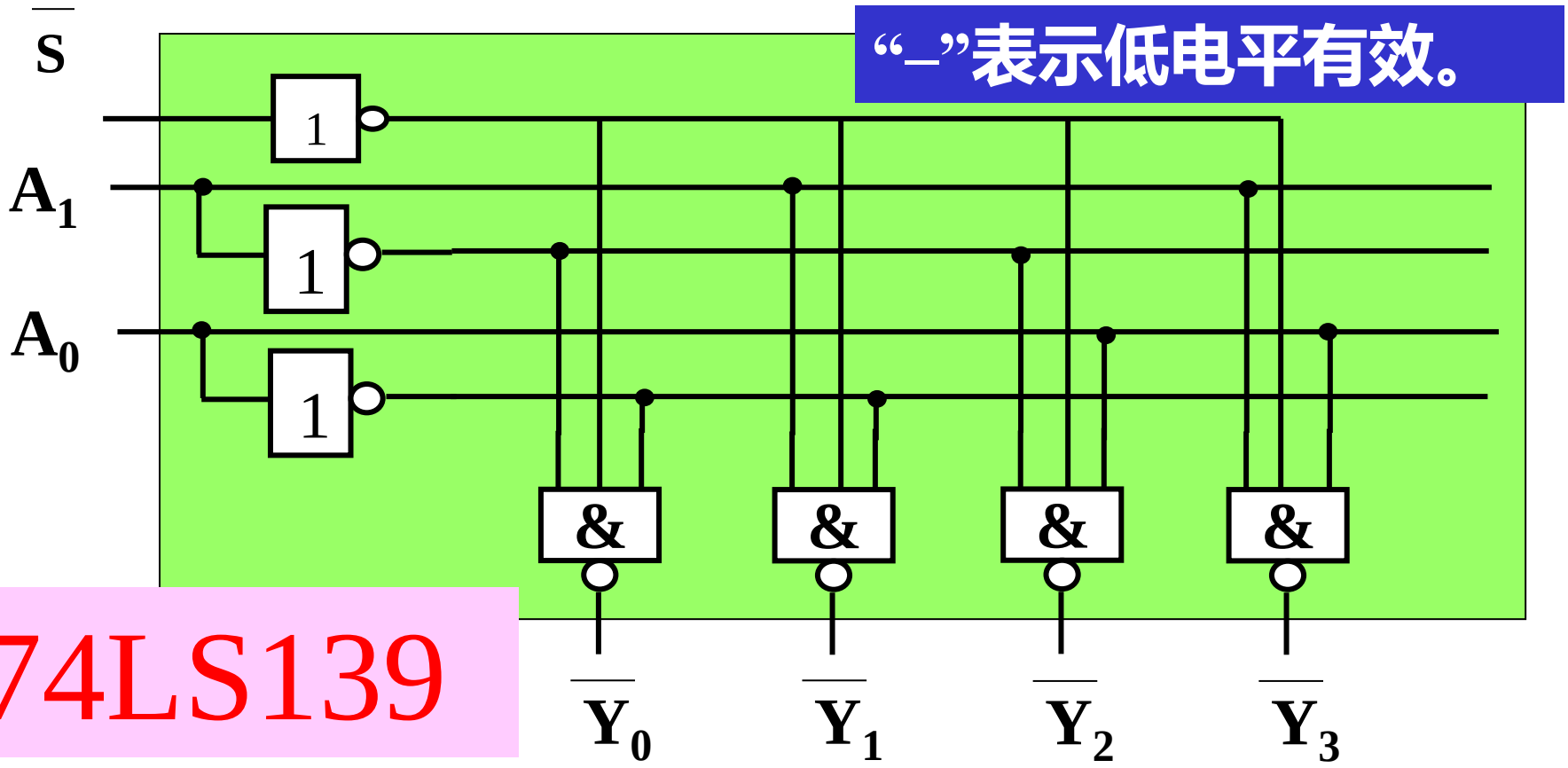


74LS139

74LS139的功能表

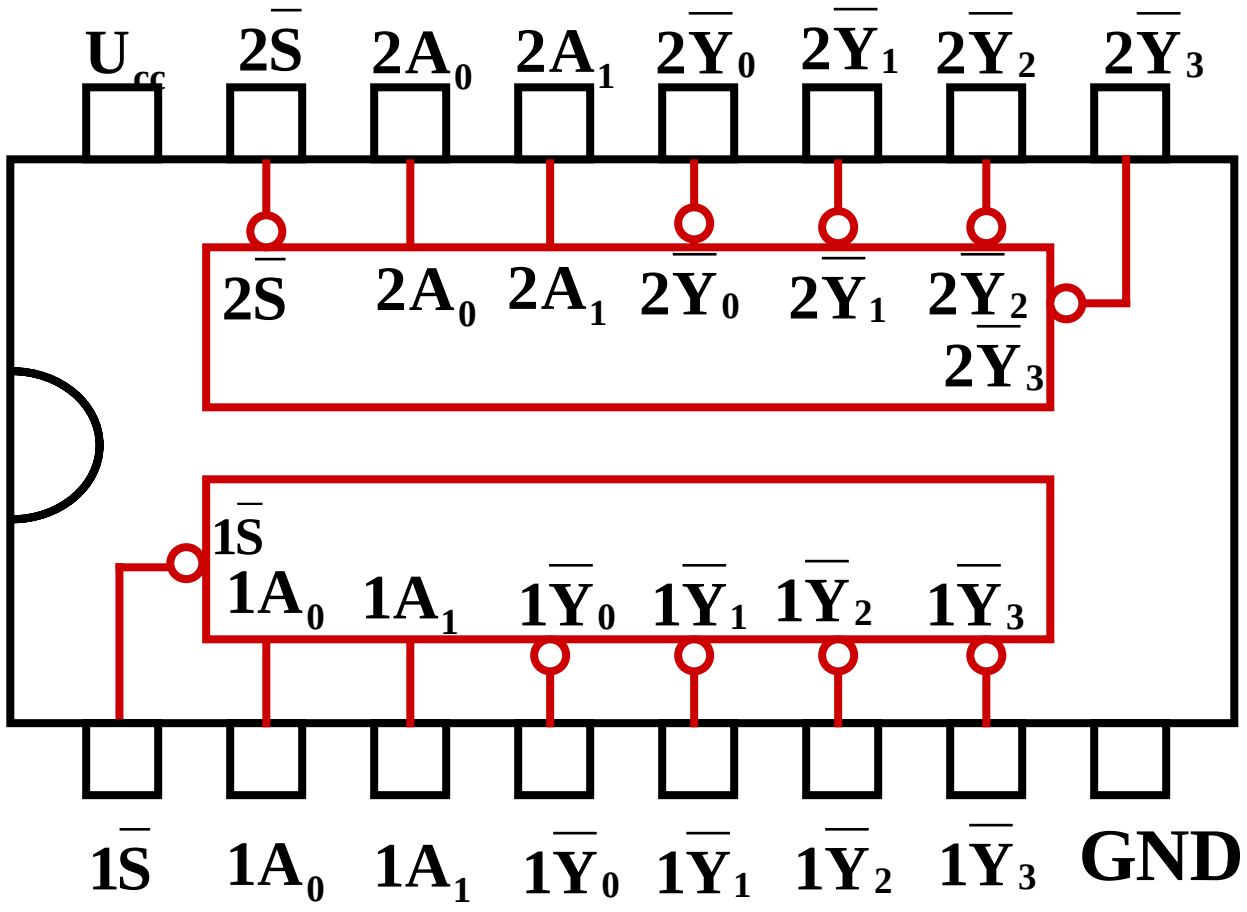
$\overline{S}$	A_1	A_0	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

“_”表示低电平有效。



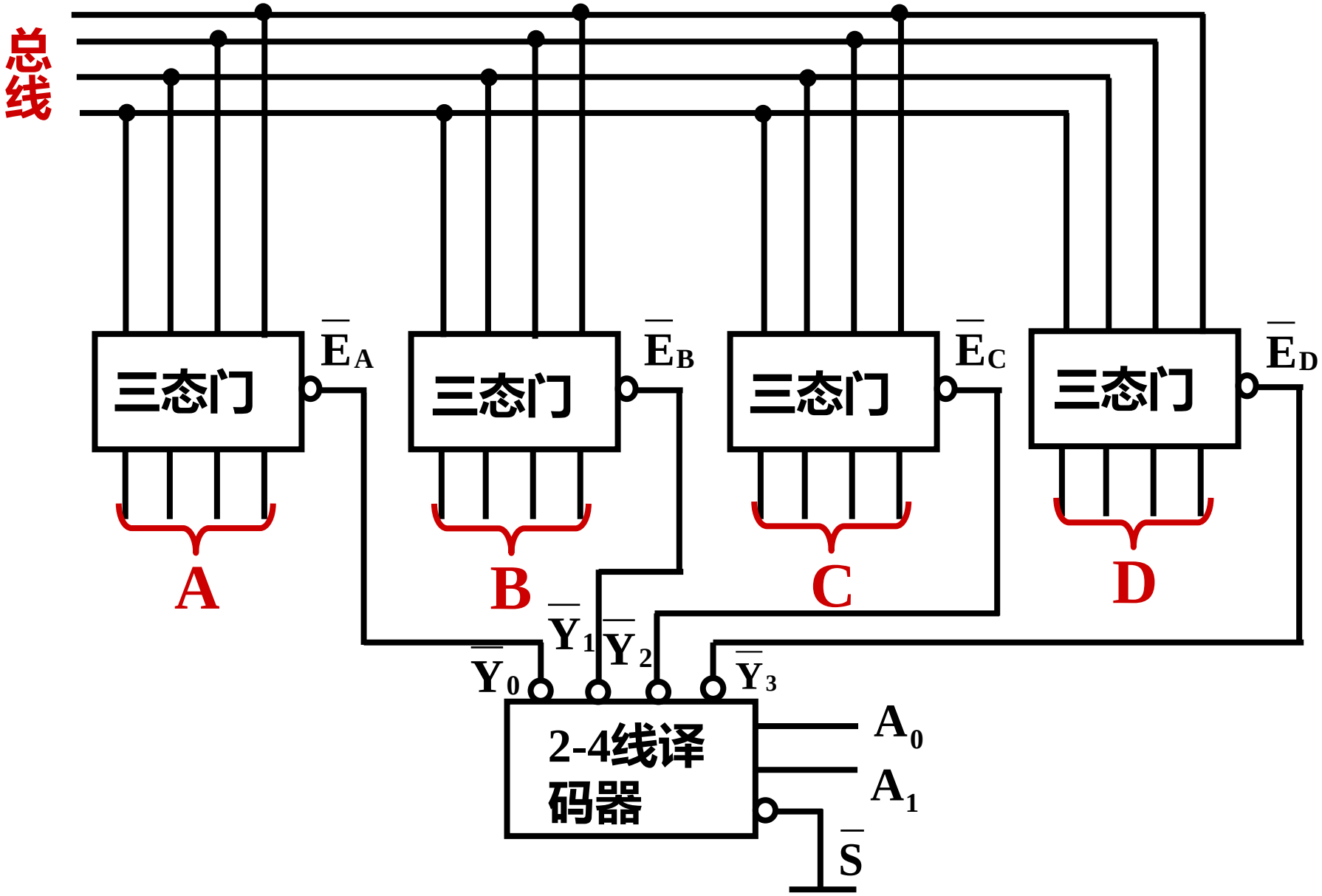
74LS139

74LS139管脚图

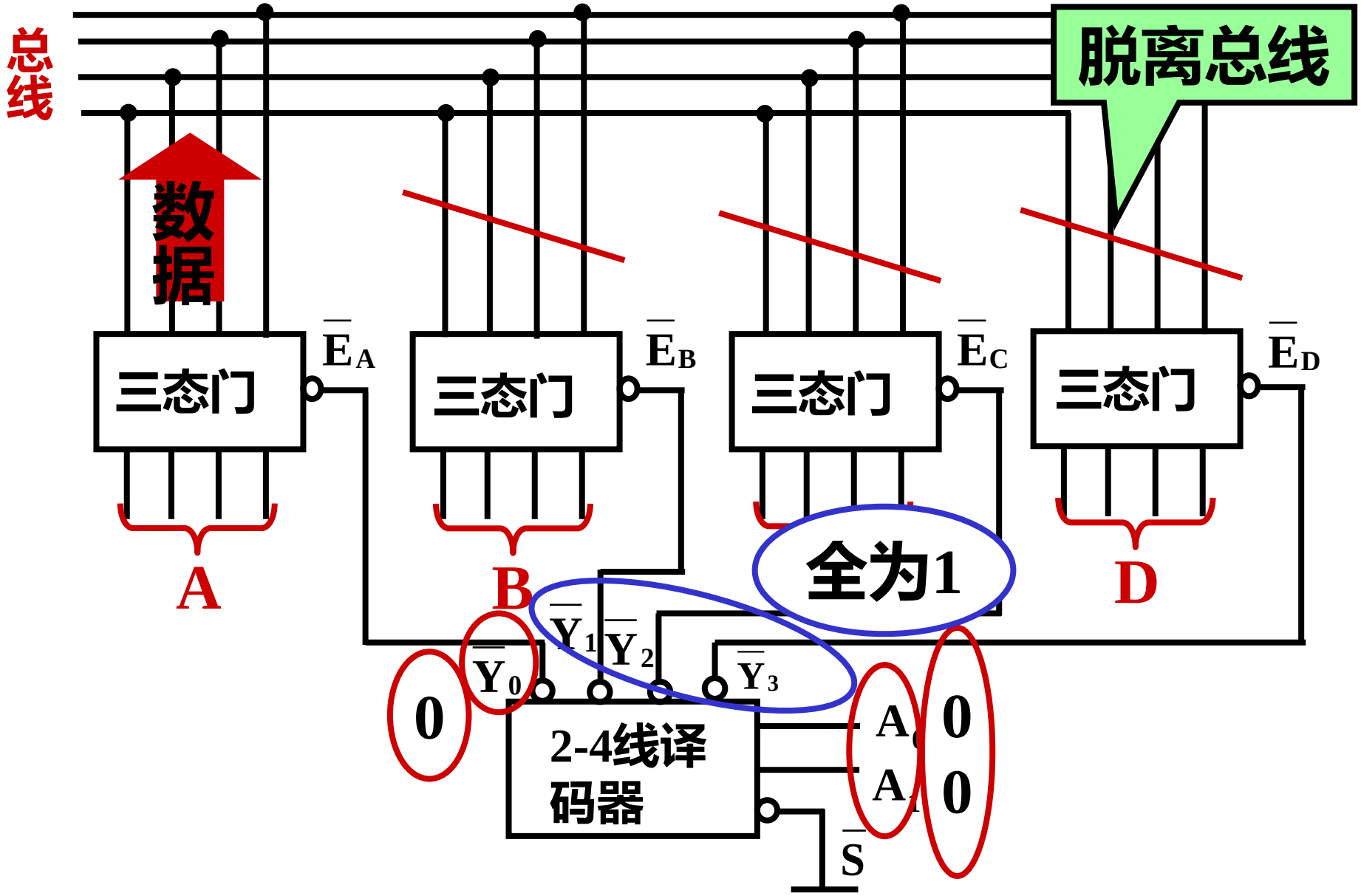


一片139中含两个2-4译码器

例：利用2-4线译码器分时将采样数据送入计算机。

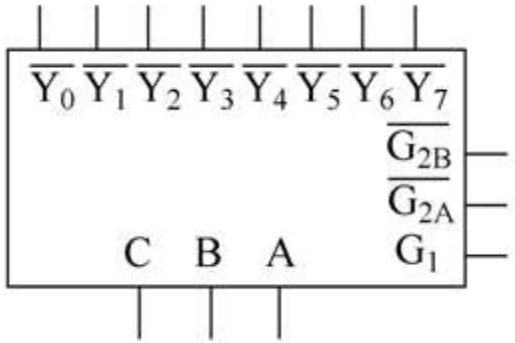


工作原理：（以A_0A_1=00为例）

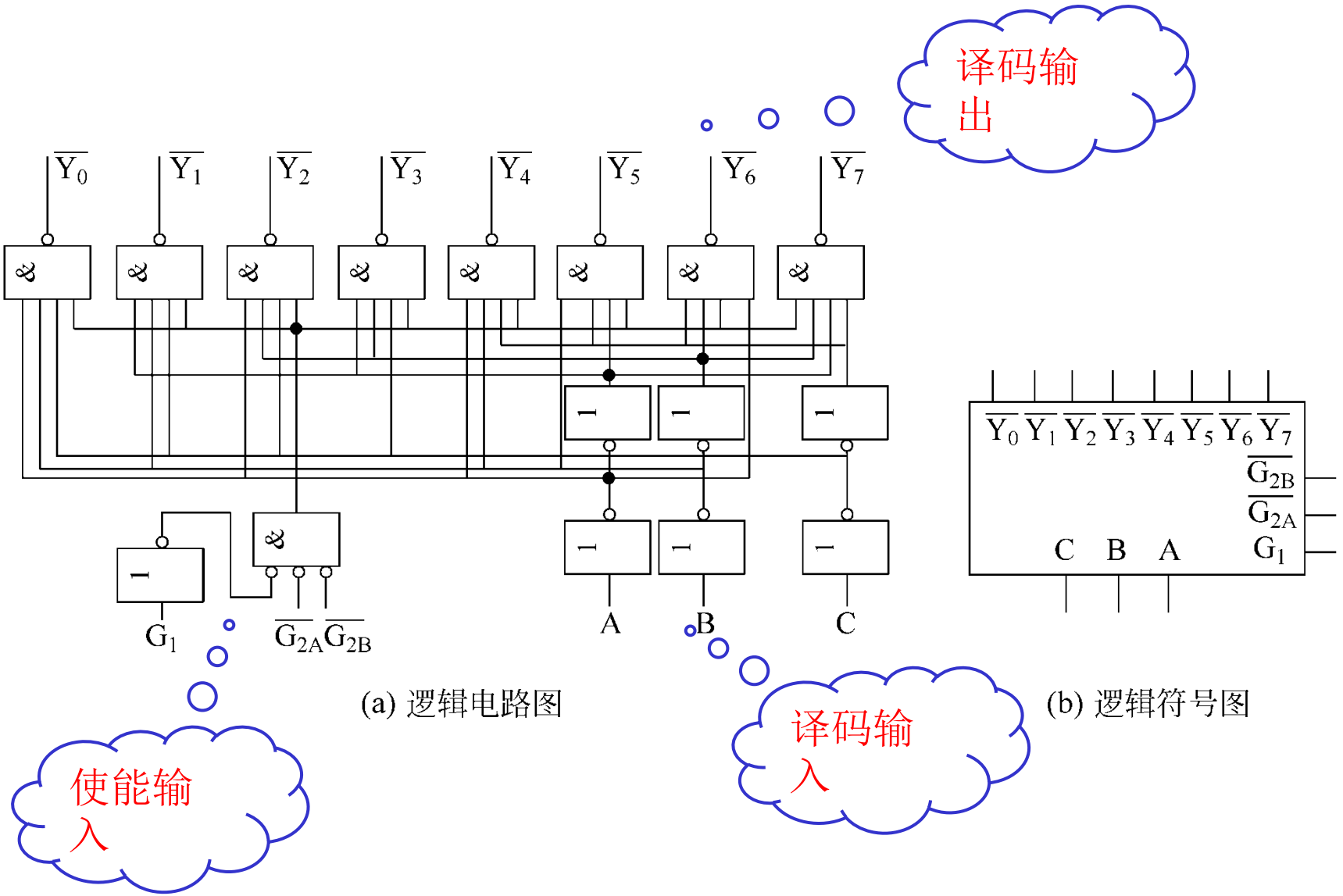


2、 3线—8线译码器 74LS138

- 引脚
  - 译码输入端：A、B、C
  - 译码输出端：$\overline{Y_0}$ 、 $\overline{Y_1}$ 、…、$\overline{Y_7}$
  - 使能输入端：G_1、$\overline{G_{2A}}$、 $\overline{G_{2B}}$
- 真值表
- 电路图
- 输出表达式



74LS138 — 电路图



(a) 逻辑电路图

(b) 逻辑符号图

74LS138 — 真值表

使能输入		译 码 输 入			输　　出							
G_1	G_2	C	B	A	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$	$\overline{Y_4}$	$\overline{Y_5}$	$\overline{Y_6}$	$\overline{Y_7}$
×	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0

注：$G_2 = \overline{G_{2A}} + \overline{G_{2B}}$ ，1为高电平，0为低电平，×为任意。

74LS138 —输出表达式

$$\overline{Y_0} = \overline{\overline{C}\,\overline{B}\,\overline{A}} = \overline{m_0}$$

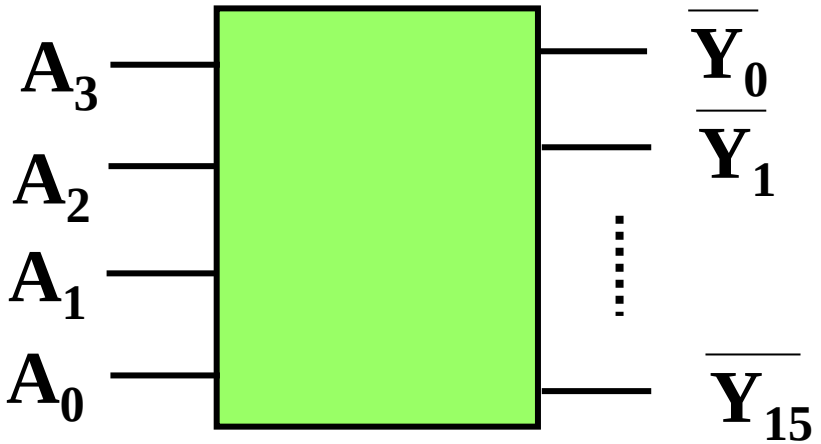
$$\overline{Y_1} = \overline{\overline{C}\,\overline{B}A} = \overline{m_1}$$

$$\overline{Y_2} = \overline{\overline{C}B\overline{A}} = \overline{m_2}$$

$$\overline{Y_7} = \overline{CBA} = \overline{m_7}$$

每个输出对应一个最小项的非

3、 4线—16线译码器（74LS154）



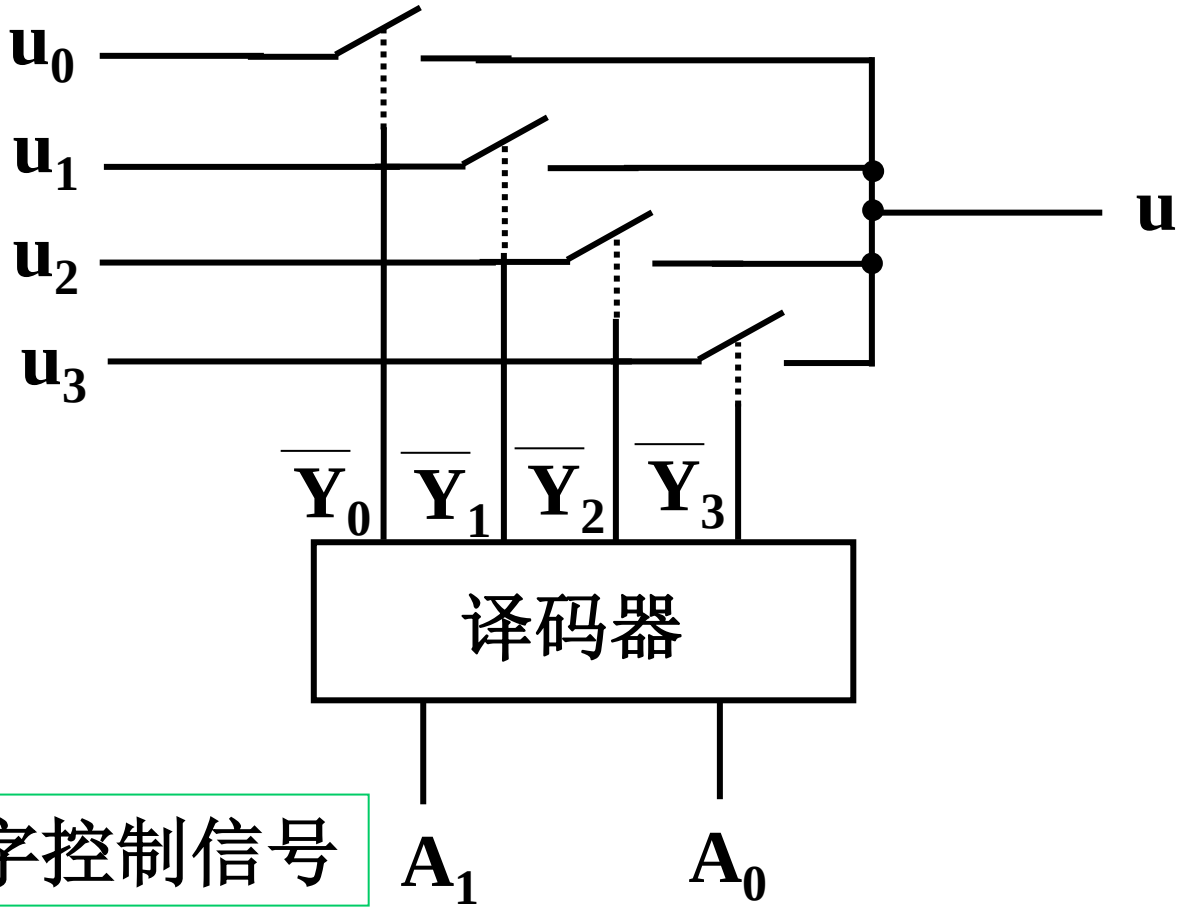
A_3	A_2	A_1	A_0	
0	0	0	0	只 $\overline{Y_0}=0$
0	0	0	1	只 $\overline{Y_1}=0$
		⋮		
1	1	1	1	只 $\overline{Y_{15}}=0$

(逻辑电路设计略,设计方法同2—4译码器)

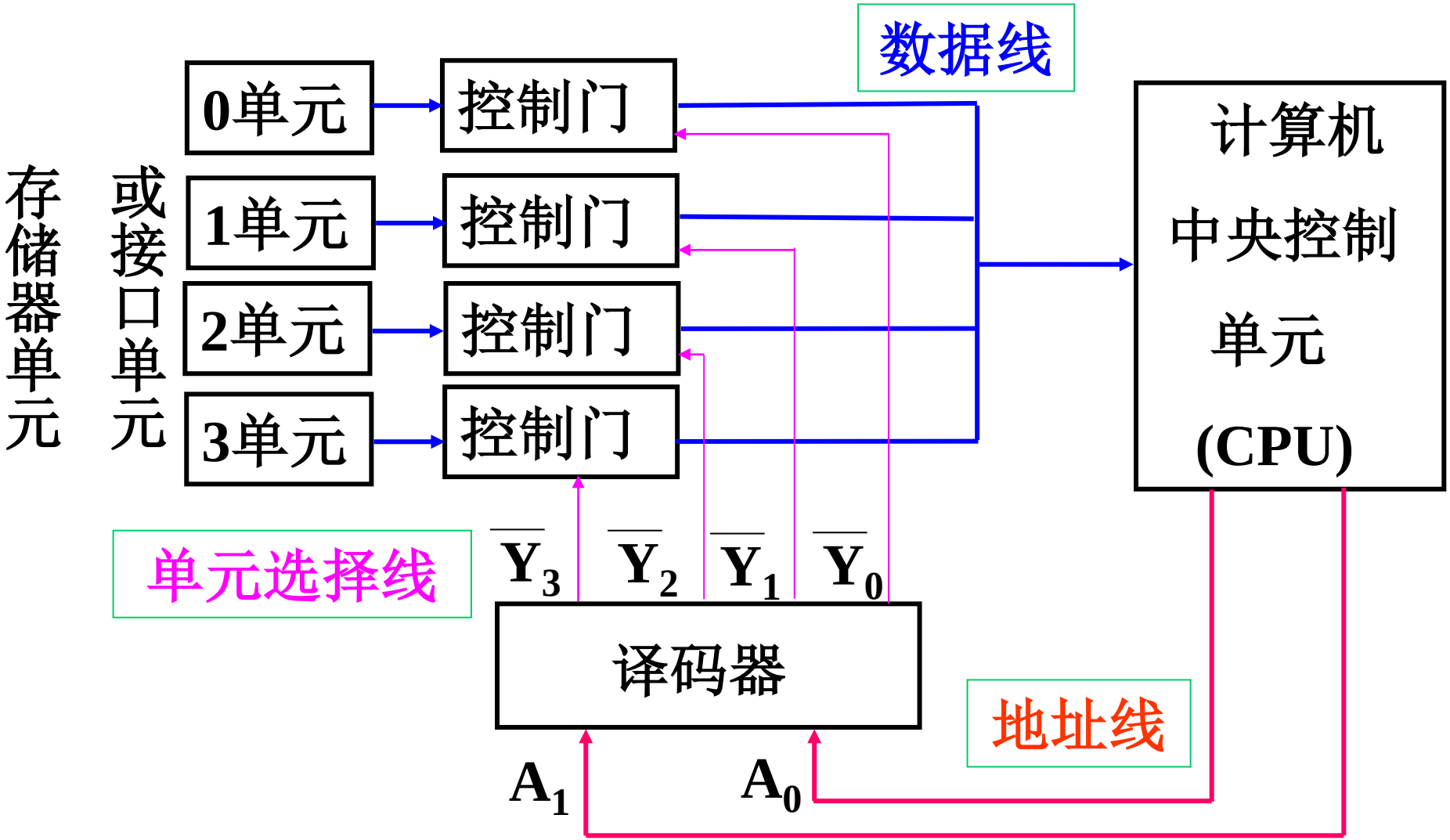
4、译码器的应用举例：

(1) 模拟信号多路转换的数字控制

输入模拟电压　模拟电子开关　输出模拟电压



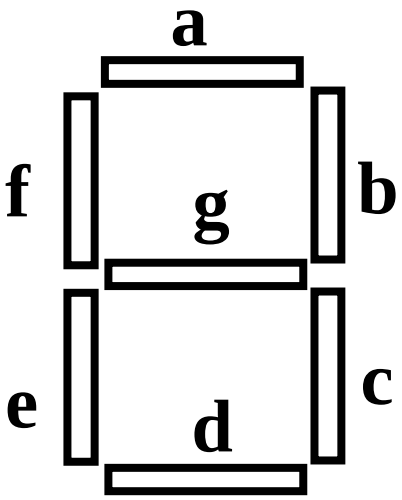
(2) 计算机中存储器单元及输入输出接口的寻址



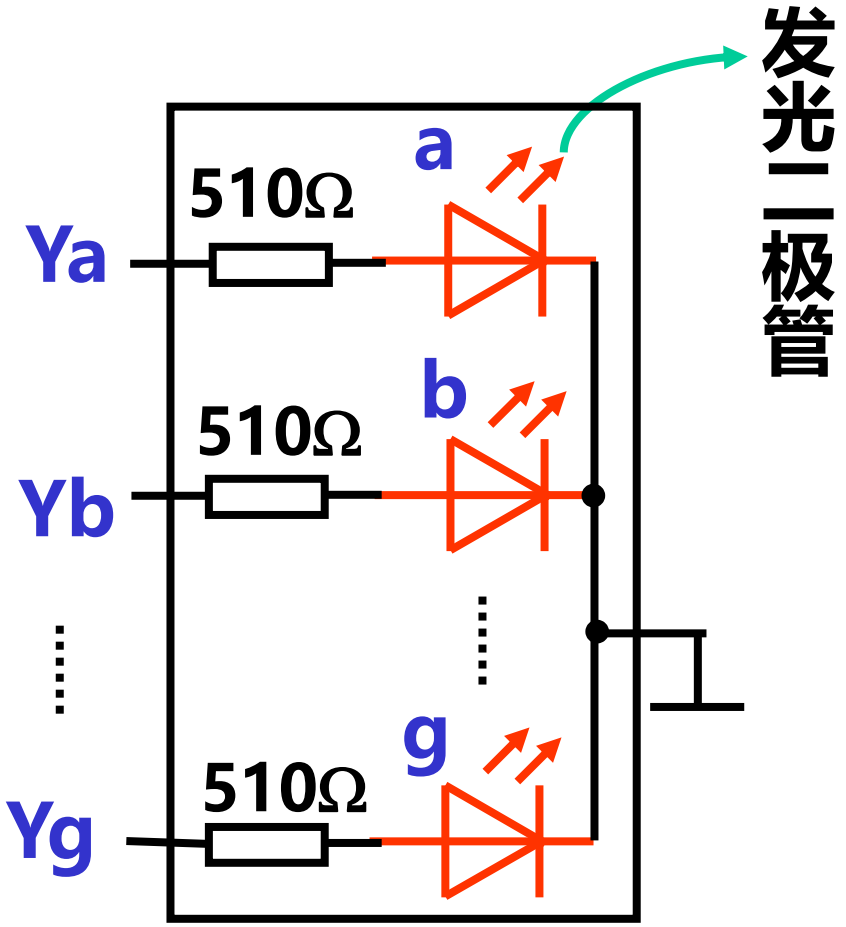
二、显示译码器

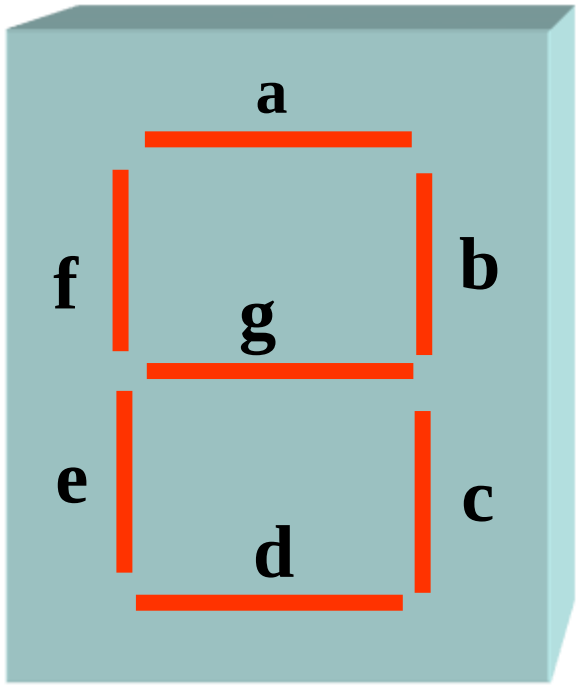
在数字系统中，常常需要将运算结果用人们习惯的十进制显示出来，这就要用到显示译码器。

显示器件：常用的是七段显示器件。



七段显示器件的工作原理：





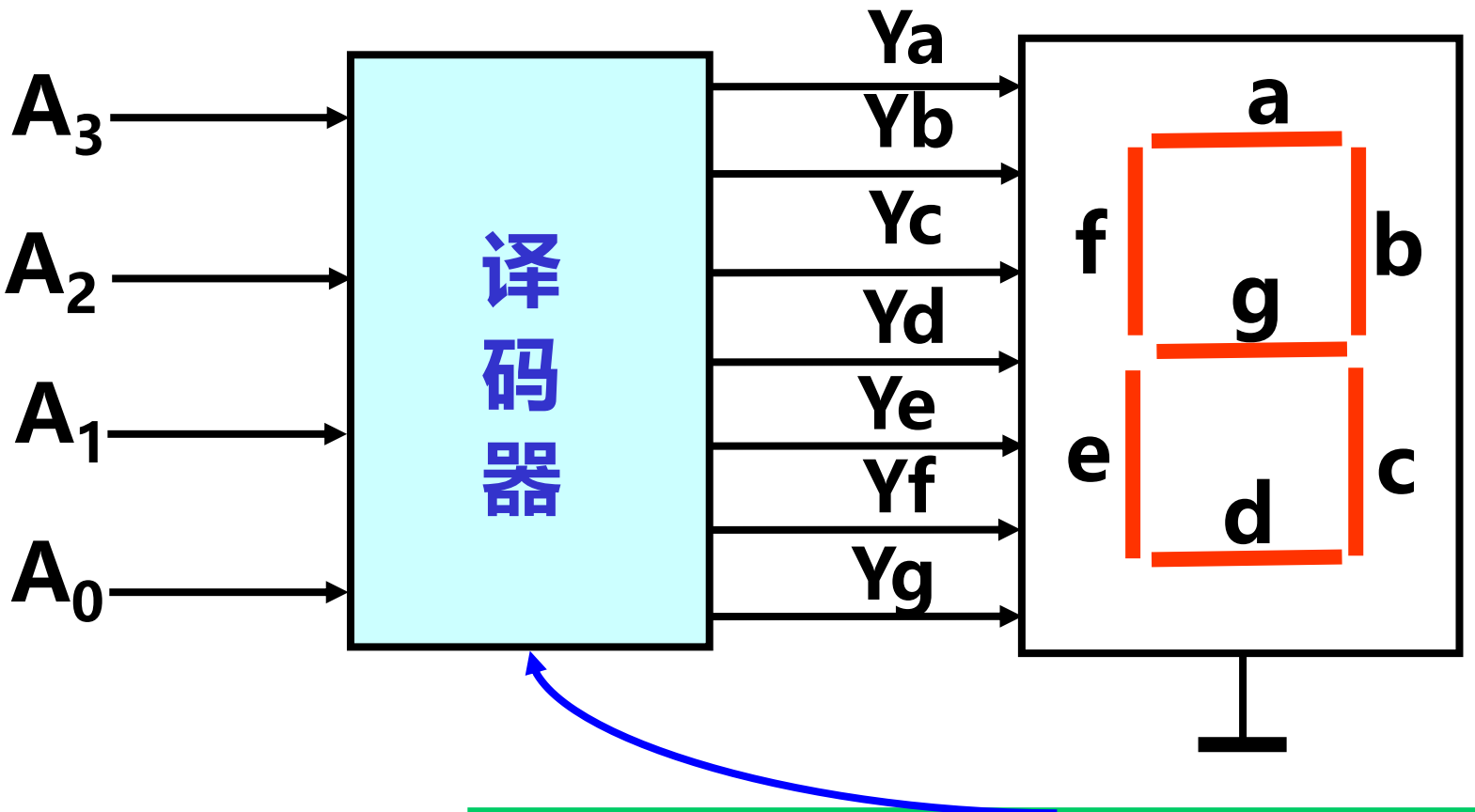
Ya-Yg: 控制信号

高电平时,对应的LED亮

低电平时,对应的LED灭

七段数码管显示译码器

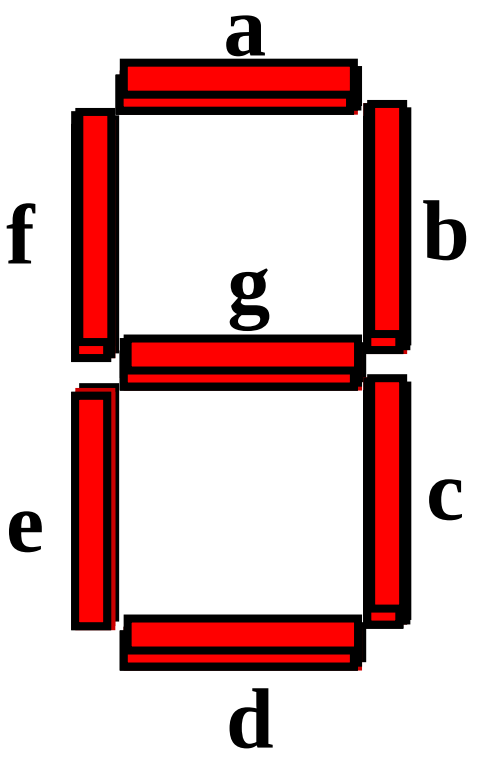
A_3-A_0: 输入数据



要设计的七段数码管显示译码器

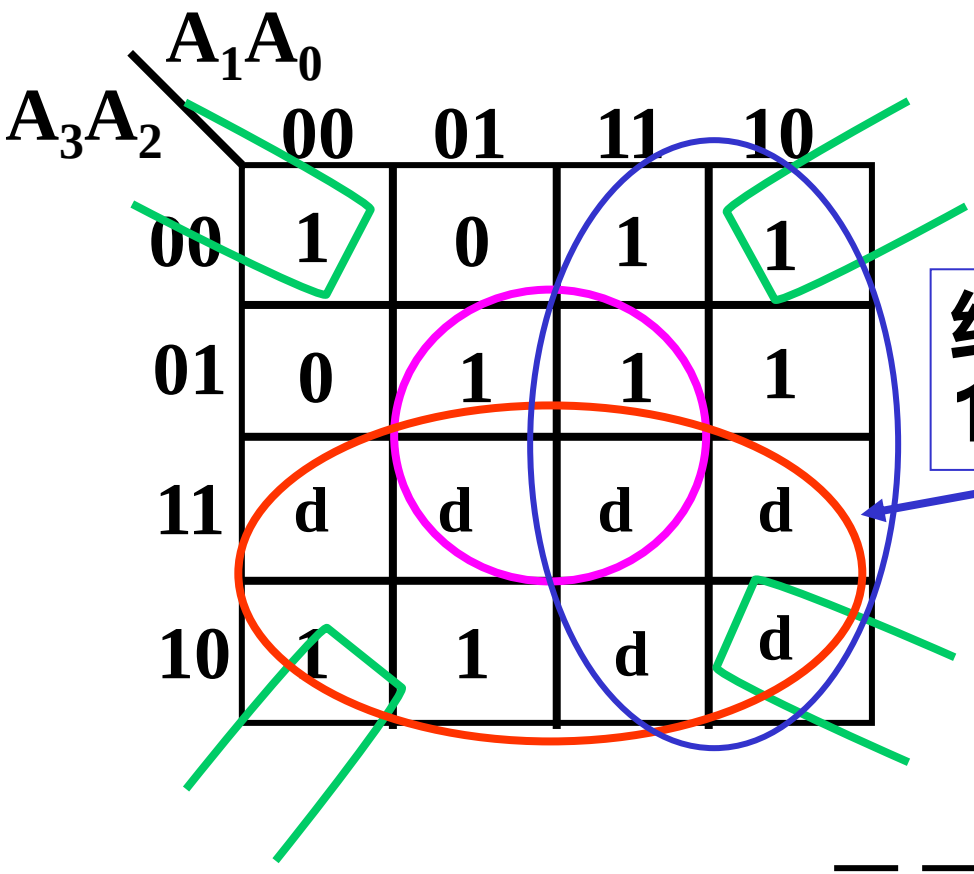
七段显示器件的工作原理：

字型	$A_3A_2A_1A_0$	a	b	c	d	e	f	g
0	0 0 0 0	1	1	1	1	1	1	0
1	0 0 0 1	0	1	1	0	0	0	0
2	0 0 1 0	1	1	0	1	1	0	1
3	0 0 1 1	1	1	1	1	0	0	1
4	0 1 0 0	0	1	1	0	0	1	1
5	0 1 0 1	1	0	1	1	0	1	1
6	0 1 1 0	1	0	1	1	1	1	1
7	0 1 1 1	1	1	1	0	0	0	0
8	1 0 0 0	1	1	1	1	1	1	1
9	1 0 0 1	1	1	1	1	0	1	1



	$A_3A_2A_1A_0$				a
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1

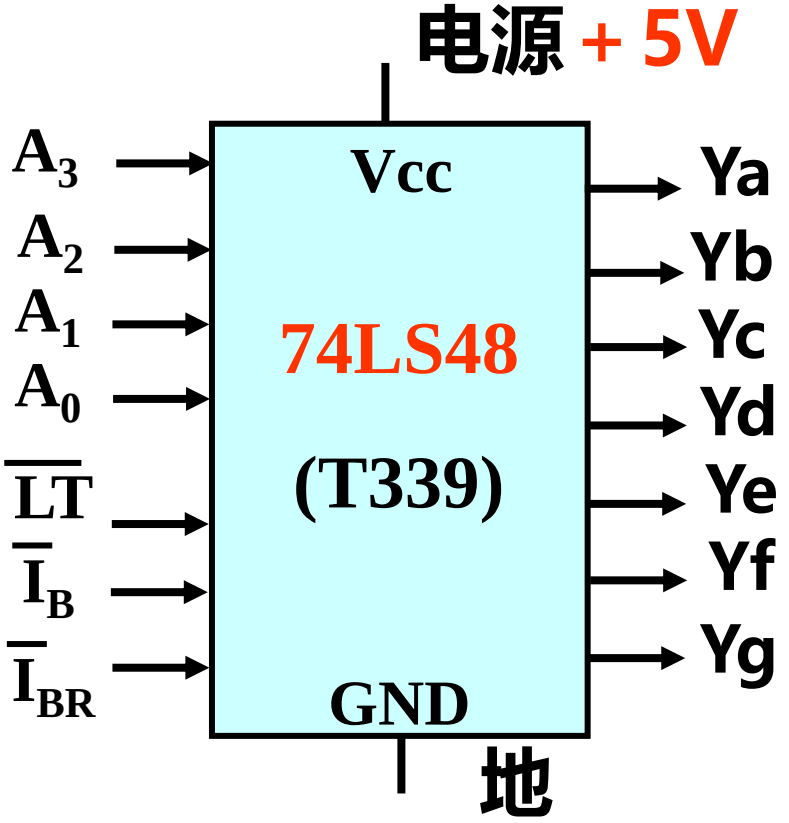
先设计输出a的逻辑表示式及电路图



$$a=A_3+A_2A_0+A_1+\overline{A}_2\overline{A}_0$$

以同样的方法可设计出b-g的逻辑表示式及其电路图；将所有电路图画在一起，就得到总电路图。

将此电路图集成化，得到七段显示译码器的集成电路74LS48（国产型号：T339）



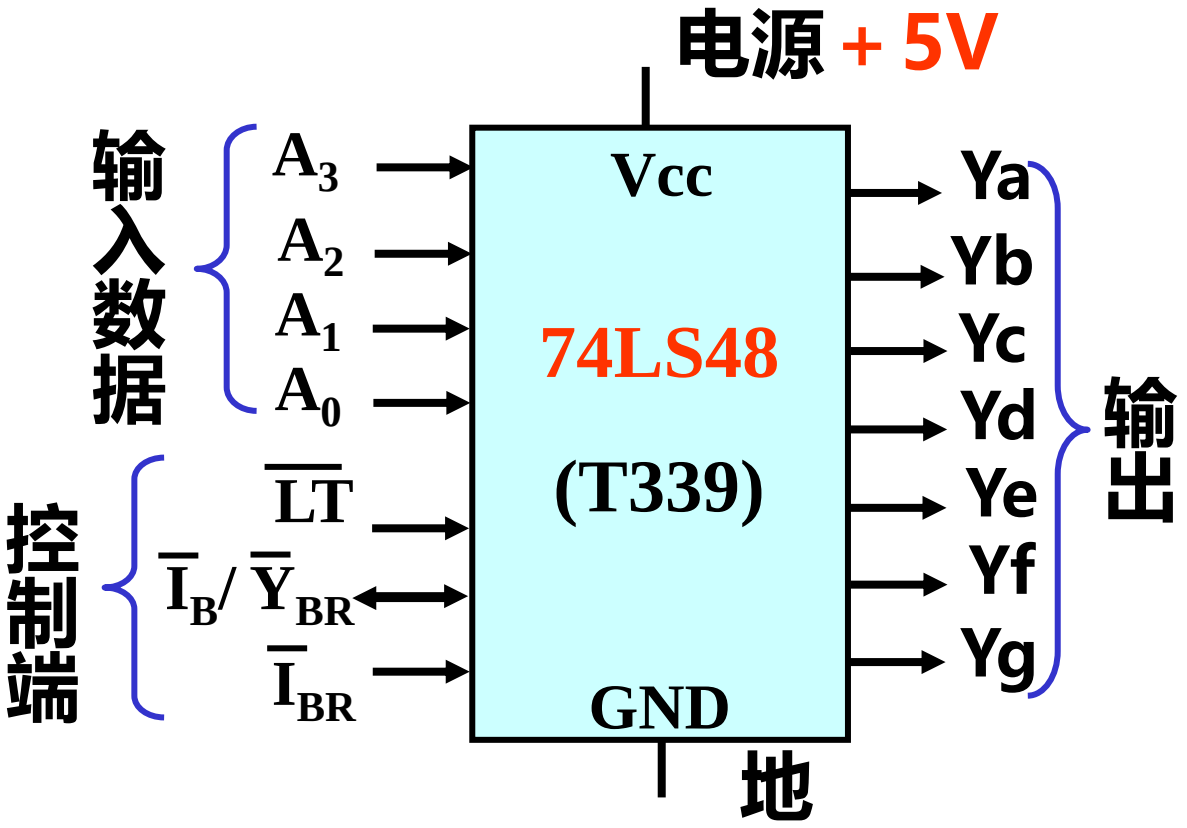
控制端

$\overline{LT}$ ： 测试端

$\overline{I}_B$ ： 灭灯端(输入)

$\overline{I}_{BR}$： 灭零输入端

$\overline{Y}_{BR}$ ： 灭零输出端



控制端功能

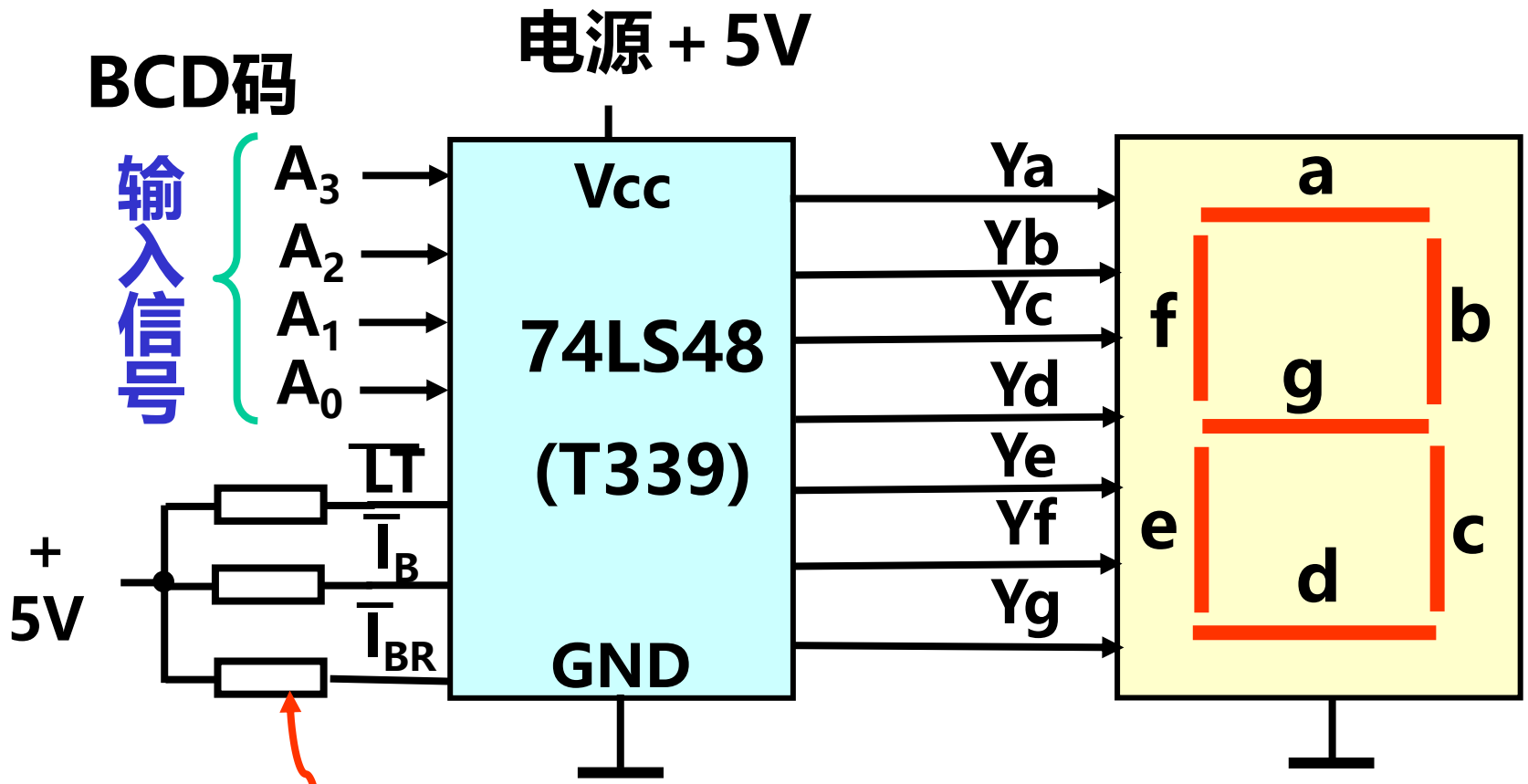
$\overline{LT}$ 为0时，使Ya--Yg=1,亮“8”，说明工作正常。

$\overline{I}_B$ 为0时，使Ya--Yg=0，全灭。

$\overline{I}_{BR}$ 为0且$A_3 \sim A_0 = 0$时，使Ya-Yg=0，全灭。

$\overline{Y}_{BR}$ ，当$\overline{I}_{BR} = 0$且$A_3 \sim A_0 = 0$时，$\overline{Y}_{BR} = 0$；否则$\overline{Y}_{BR} = 1$

七段显示译码器74LS48与数码管的连接

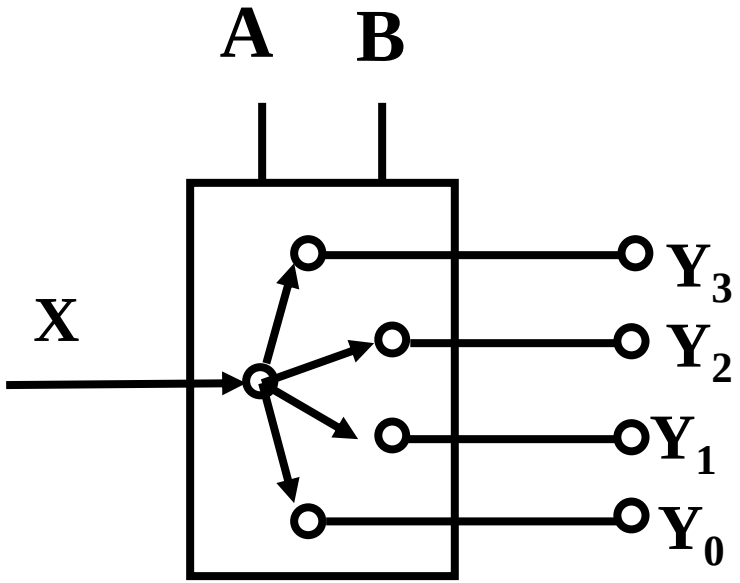


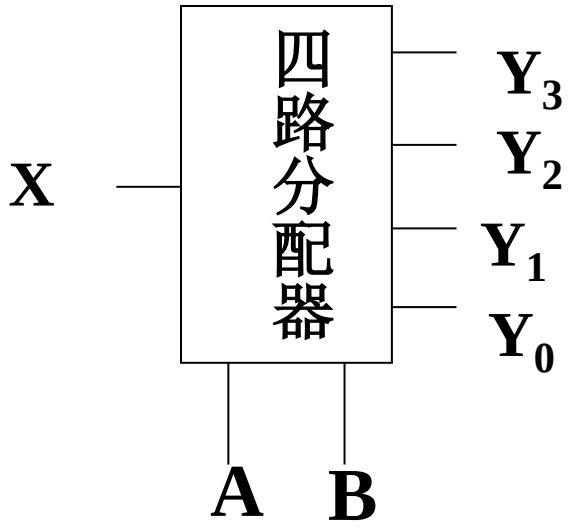
此三控制端不用时，
通过电阻接高电平。

三、数据分配器

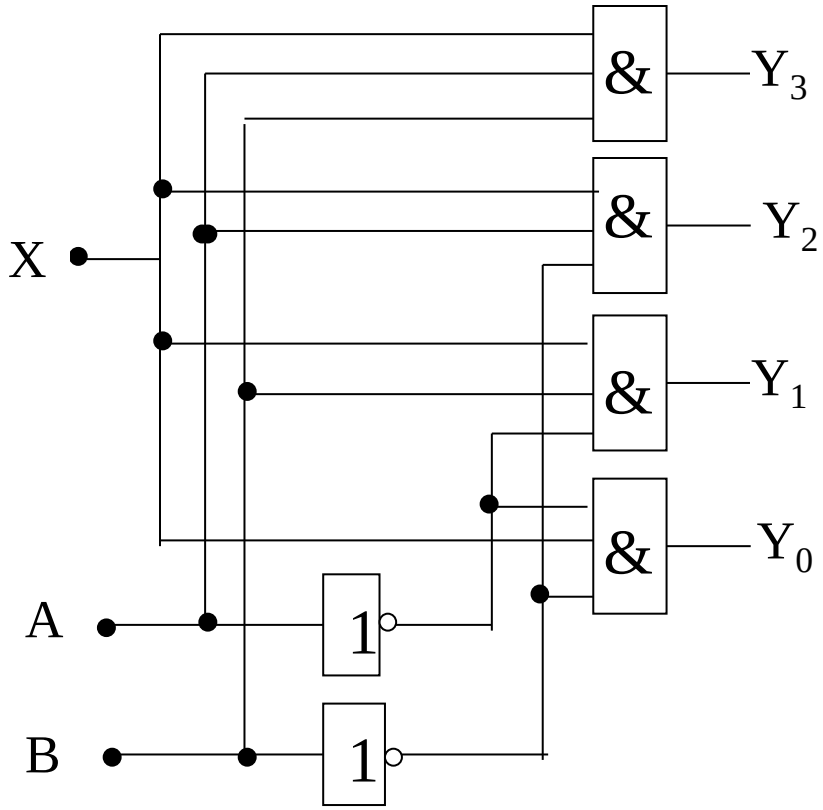
数据分配器是将一个数据源来的数据根据需要送到多个不同的通道上去的逻辑电路。

它将一个输入x分时地送到多路输出上去。具体选择哪一路输出由一组选择变量确定。它有一根输入线，n根选择线，2^n根输出线。





三、数据分配器



AB	$Y_0Y_1Y_2Y_3$
00	X 0 0 0
01	0 X 0 0
10	0 0 X 0
11	0 0 0 X

4.3 编码器

所谓编码就是赋予选定的一系列二进制代码以固定的含义。具有编码功能的逻辑电路——编码器

一、二进制编码器

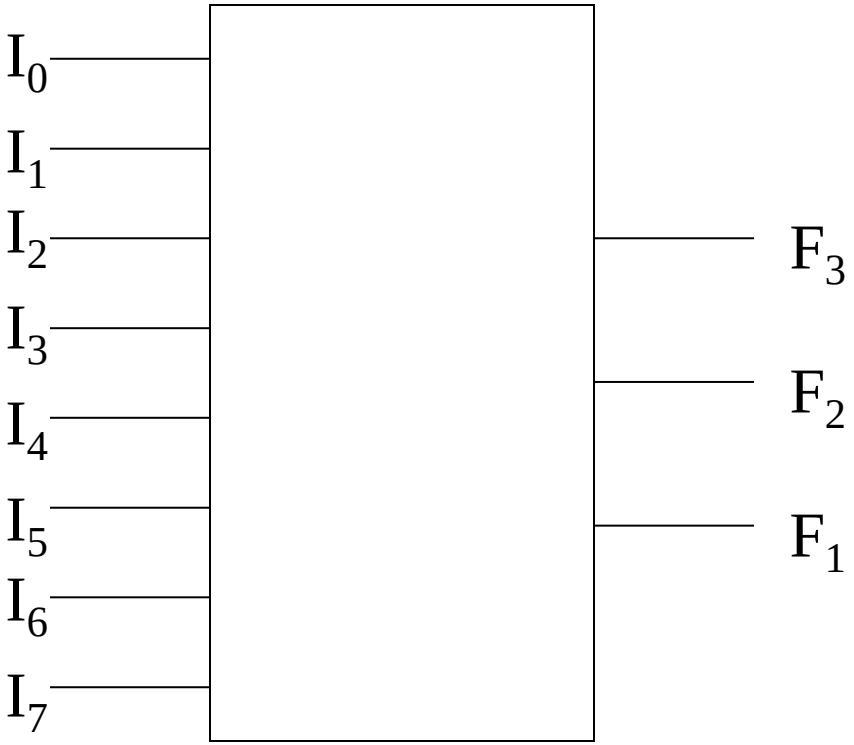
n个二进制代码（n位二进制数）有2^n种不同的组合，可以表示2^n个信号。

对N个信号进行编码时，可以用公式$2^n \geq N$来确定需要使用的二进制数的位数n

设计过程就是一般组合电路的设计过程。

例：将0、1、2、…、7这八个十进制数码编成二进制代码。

设八个输入端为I_0~I_7八种状态，与之对应的输出设为F_1、F_2、F_3，共三位二进制数。



八线-三线编码器

设计编码器的过程与设计一般的组合逻辑电路相同，首先要列出状态表（即真值表），然后写出逻辑表达式并进行化简，最后画出逻辑图。

真值表

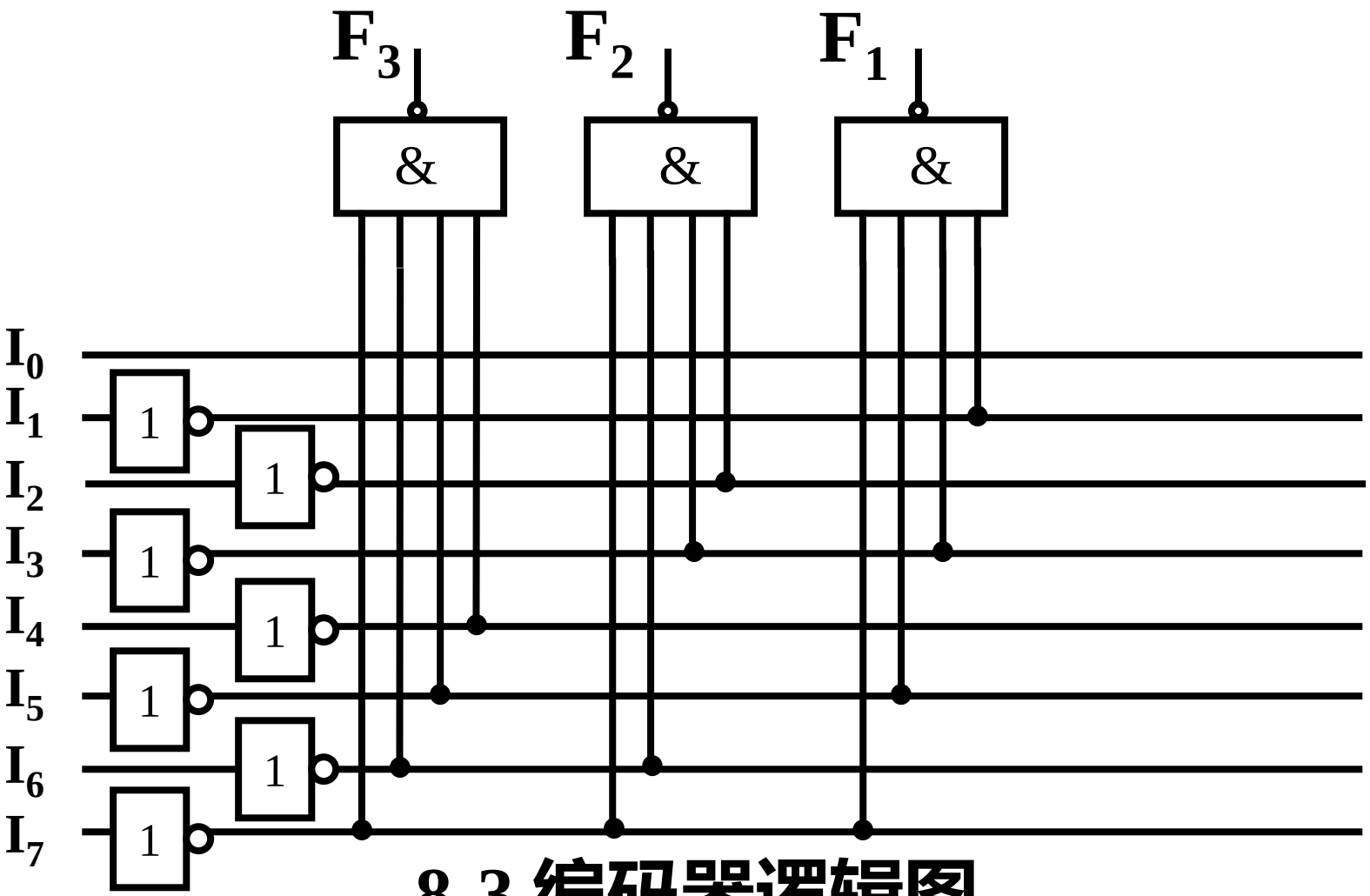
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	F_3	F_2	F_1
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$$F_1 = I_1 + I_3 + I_5 + I_7 = \overline{\bar{I}_1 \bar{I}_3 \bar{I}_5 \bar{I}_7}$$

$$F_2 = I_2 + I_3 + I_6 + I_7 = \overline{\bar{I}_2 \bar{I}_3 \bar{I}_6 \bar{I}_7}$$

$$F_3 = I_4 + I_5 + I_6 + I_7 = \overline{\bar{I}_4 \bar{I}_5 \bar{I}_6 \bar{I}_7}$$

$$F_3 = \overline{\bar{I}_4 \bar{I}_5 \bar{I}_6 \bar{I}_7} \qquad F_2 = \overline{\bar{I}_2 \bar{I}_3 \bar{I}_6 \bar{I}_7} \qquad F_1 = \overline{\bar{I}_1 \bar{I}_3 \bar{I}_5 \bar{I}_7}$$



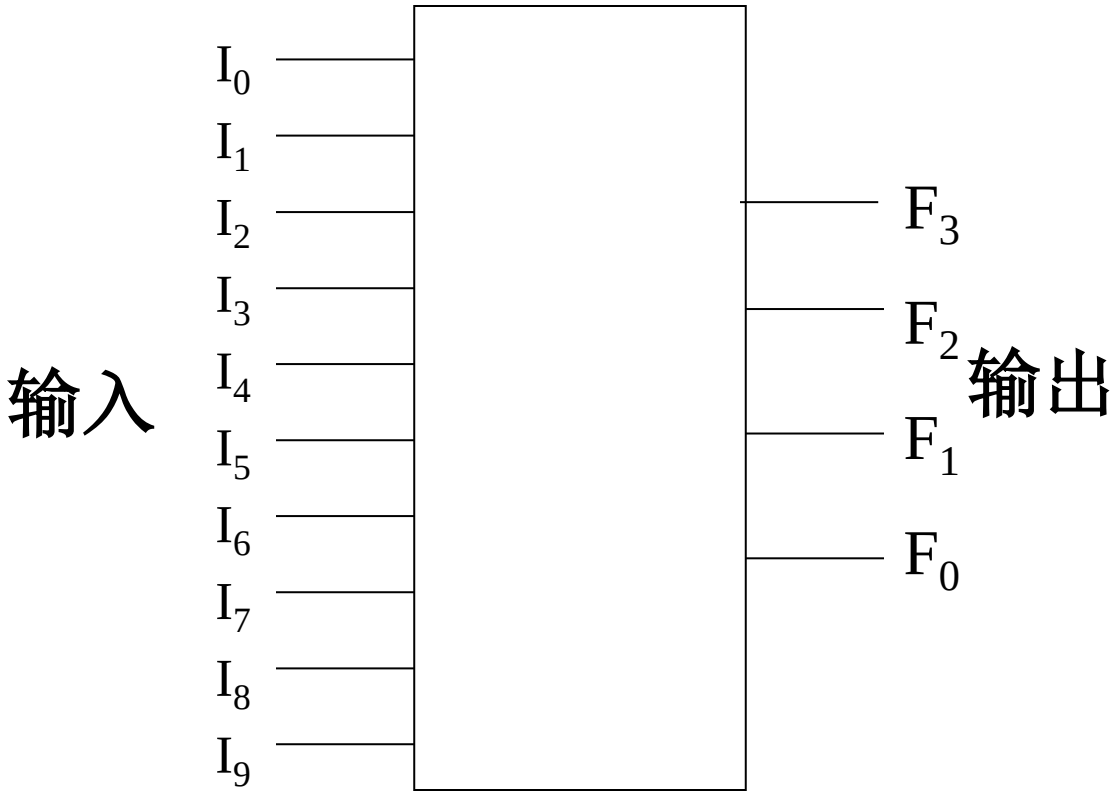
8-3 编码器逻辑图

二、二---十进制编码器

二---十进制编码器的作用：将十个状态（对应于十进制的十个代码）编制成BCD码。

十个输入 → 需要几位输出？ → 四位

$2^3 < 10 < 2^4$



状态表

输入	F_3	F_2	F_1	F_0
I_0	0	0	0	0
I_1	0	0	0	1
I_2	0	0	1	0
I_3	0	0	1	1
I_4	0	1	0	0
I_5	0	1	0	1
I_6	0	1	1	0
I_7	0	1	1	1
I_8	1	0	0	0
I_9	1	0	0	1

$I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8 I_9$	$F_3F_2F_1F_0$
0 1 1 1 1 1 1 1 1 1	0 0 0 0
1 0 1 1 1 1 1 1 1 1	0 0 0 1
1 1 0 1 1 1 1 1 1 1	0 0 1 0
1 1 1 0 1 1 1 1 1 1	0 0 1 1
1 1 1 1 0 1 1 1 1 1	0 1 0 0
1 1 1 1 1 0 1 1 1 1	0 1 0 1
1 1 1 1 1 1 0 1 1 1	0 1 1 0
1 1 1 1 1 1 1 0 1 1	0 1 1 1
1 1 1 1 1 1 1 1 0 1	1 0 0 0
1 1 1 1 1 1 1 1 1 0	1 0 0 1

$$F_3 = \bar{I}_8 + \bar{I}_9 = \overline{I_8 \cdot I_9}$$

$$F_2 = \overline{I_4 I_5 I_6 I_7}$$

$$F_1 = \overline{I_2 I_3 I_6 I_7}$$

$$F_0 = \overline{I_1 I_3 I_5 I_7 I_9}$$

$I_0\ I_1\ I_2\ I_3\ I_4\ I_5\ I_6\ I_7\ I_8\ I_9$	$F_3 F_2 F_1 F_0$
0 1 1 1 1 1 1 1 1 1	0 0 0 0
1 0 1 1 1 1 1 1 1 1	0 0 0 1
1 1 0 1 1 1 1 1 1 1	0 0 1 0
1 1 1 0 1 1 1 1 1 1	0 0 1 1
1 1 1 1 0 1 1 1 1 1	0 1 0 0
1 1 1 1 1 0 1 1 1 1	0 1 0 1
1 1 1 1 1 1 0 1 1 1	0 1 1 0
1 1 1 1 1 1 1 0 1 1	0 1 1 1
1 1 1 1 1 1 1 1 0 1	1 0 0 0
1 1 1 1 1 1 1 1 1 0	1 0 0 1

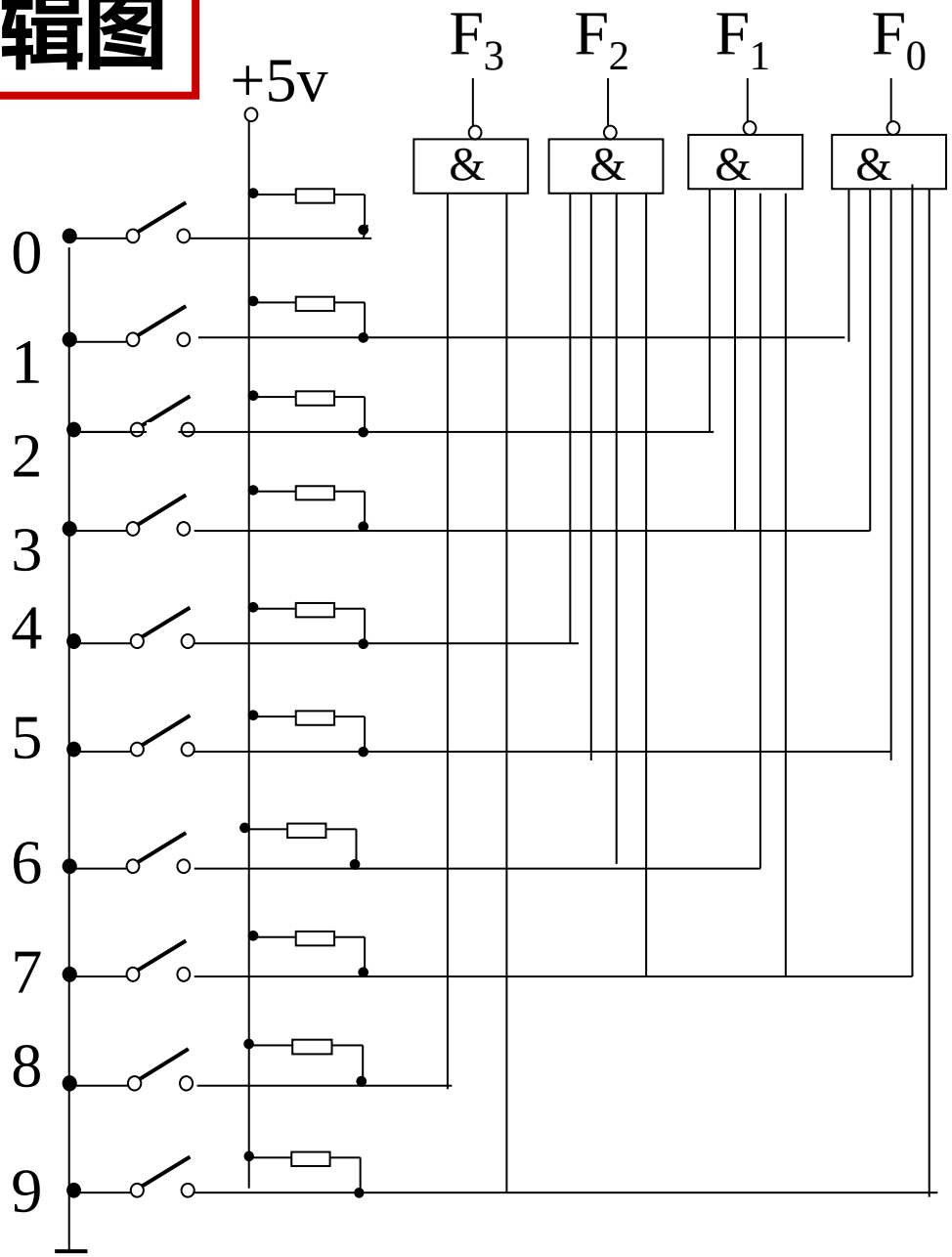
逻辑图

$$F_3 = \bar{I}_8 + \bar{I}_9 = \overline{I_8 \cdot I_9}$$

$$F_2 = \overline{I_4 I_5 I_6 I_7}$$

$$F_1 = \overline{I_2 I_3 I_6 I_7}$$

$$F_0 = \overline{I_1 I_3 I_5 I_7 I_9}$$



二进制优先编码器74LS148

表 2-8　74LS148 功能表

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

- $\overline{I_0}$～$\overline{I_7}$为8个输入端
- $\overline{Y_2}$，$\overline{Y_1}$，$\overline{Y_0}$ 为3位二进制代码输出端
- 输出编码为反码形式、输入下标大者优先级高
- $\overline{S}$ 为工作状态选择输入端(或称输入允许端)，当 $\overline{S}$=0时编码器正常工作，否则不进行编码工作
- $\overline{Y_S}$ 允许输出端，当允许编码(即 $\overline{S}$=0)而无有效信号输入时（即所有编码输入端都为1），$\overline{Y_S}$ 0
- $\overline{Y_{EX}}$ 为扩展输出端，当不允许编码时，或虽然允许编码但无信号输入时，$\overline{Y_{EX}}$ 为1。即只有当允许编码并且有有效信号输入时，$\overline{Y_{EX}}$ 才为0。

4.4 数字比较器

比较器的分类：

（1）仅比较两个数是否相等。

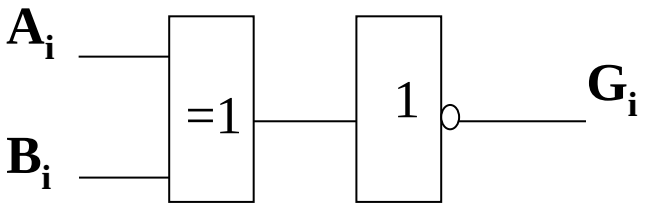
（2）除比较两个数是否相等外，还要比较两个数的大小。

一、同比较器

1、一位同比较器

A_iB_i	G_i
0 0	1相同
0 1	0相异
1 0	0相异
1 1	1相同

$$G_i = \overline{A}_i \overline{B}_i + A_i B_i$$



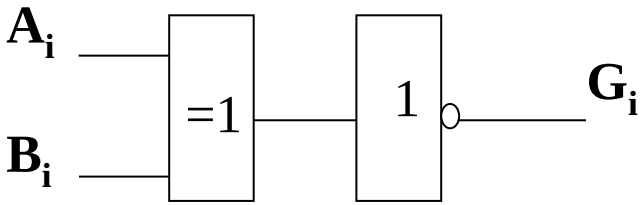
一、同比较器

$$G_i = \overline{A}_i \overline{B}_i + A_i B_i$$

2、四位同比较器

$A=A_3A_2A_1A_0$

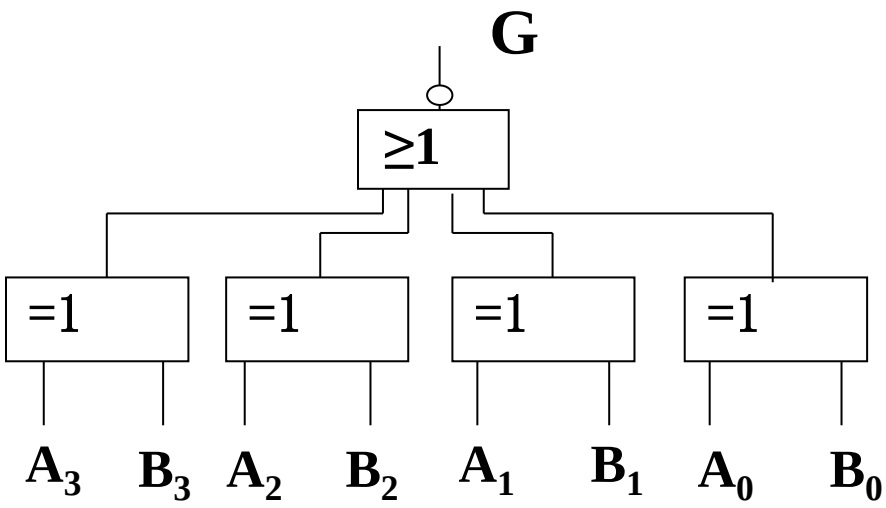
$B=B_3B_2B_1B_0$



只有对应的四位二进制数都相同，即$G_0=G_1=G_2=G_3=1$时，则两数相同，否则，两数不同。即$G=G_3G_2G_1G_0$

$$= \overline{A_3 \oplus B_3} \cdot \overline{A_2 \oplus B_2} \cdot \overline{A_1 \oplus B_1} \cdot \overline{A_0 \oplus B_0}$$

$$= \overline{(A_3 \oplus B_3) + (A_2 \oplus B_2) + (A_1 \oplus B_1) + (A_0 \oplus B_0)}$$



二、一位大小比较器

功能表

输入		输出		
A	B	A>B	A=B	A<B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$$“A>B”=A\overline{B}$$

$$“A=B”=\overline{A}\,\overline{B}+AB$$

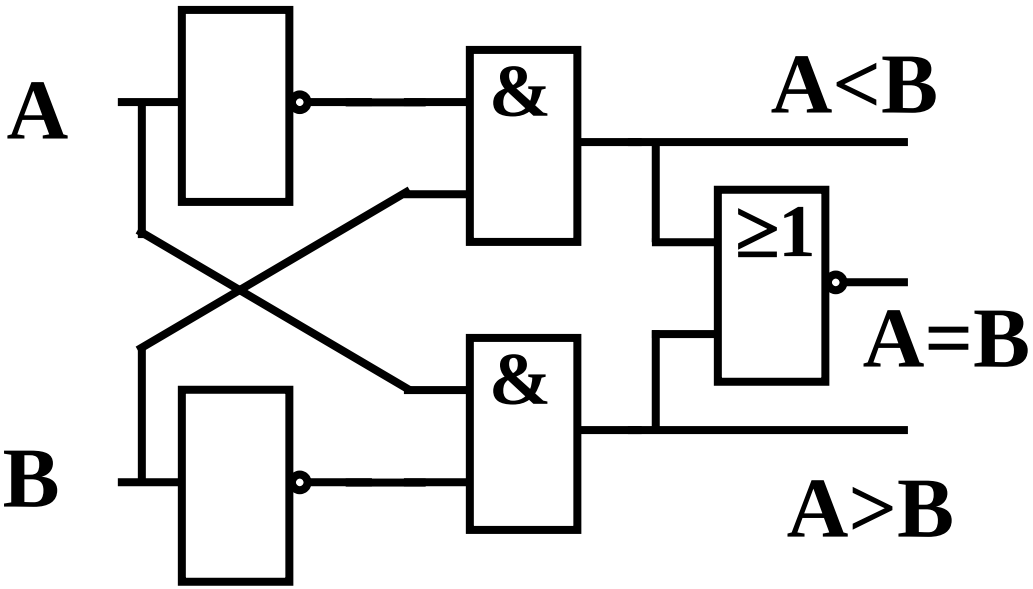
$$“A<B”=\overline{A}B$$

$$“A>B”=A\overline{B}$$

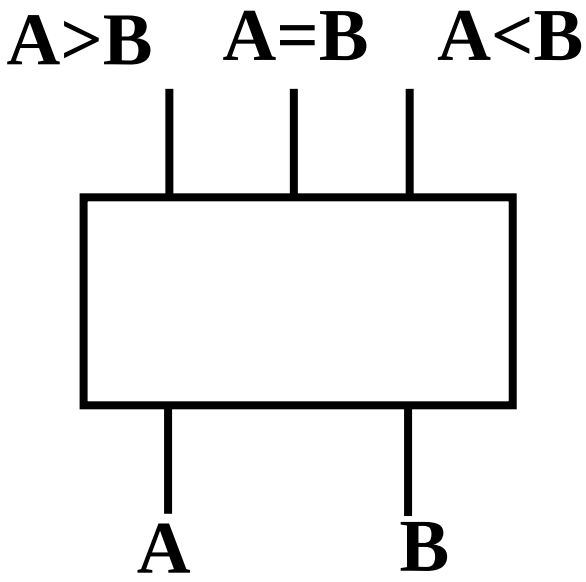
$$“A=B”=\overline{A}\,\overline{B}+AB$$

$$“A<B”=\overline{A}B$$

逻辑图



逻辑符号



三、多位数值比较器

比较原则：

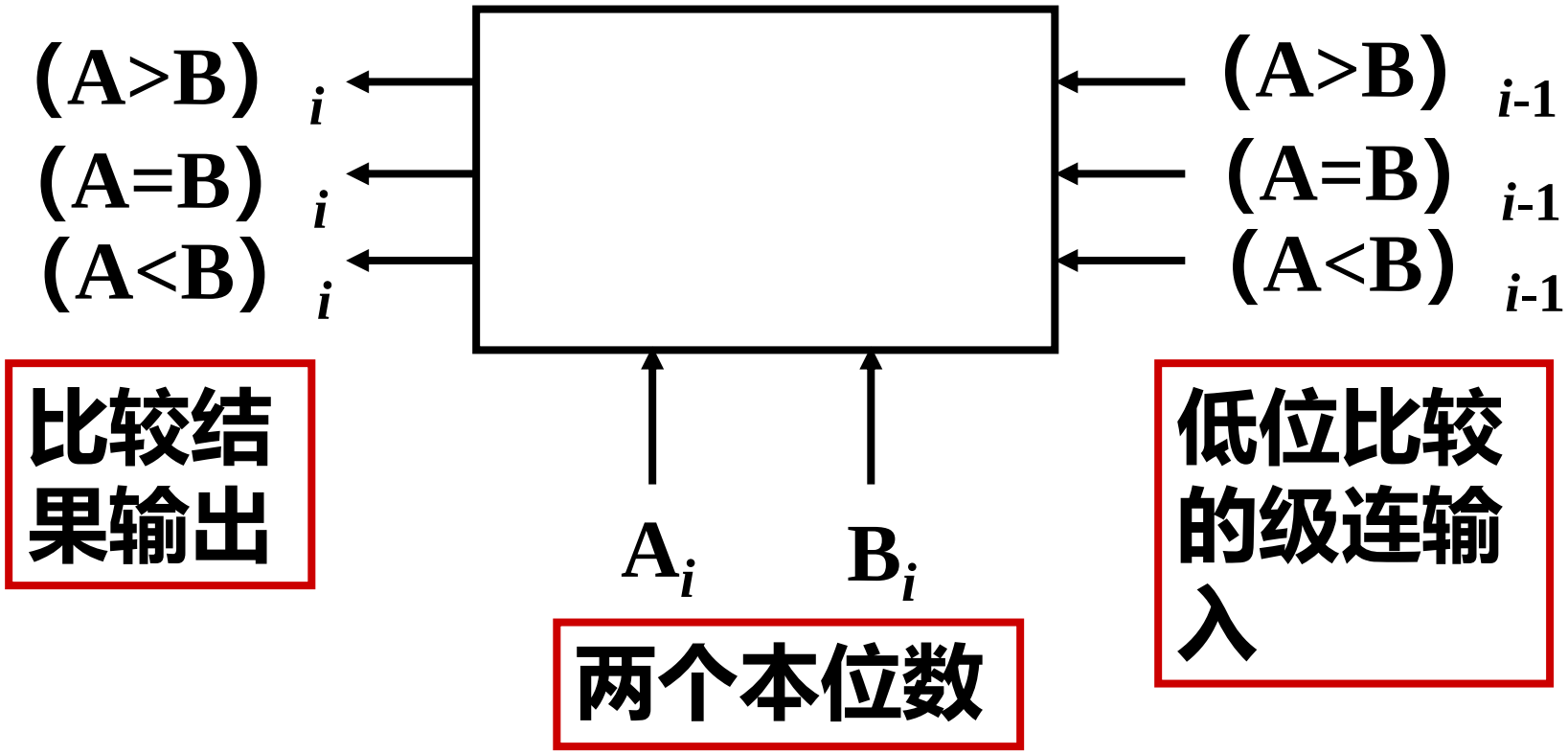
1. 先从高位比起，高位大的数值一定大。

例如：比较925和697

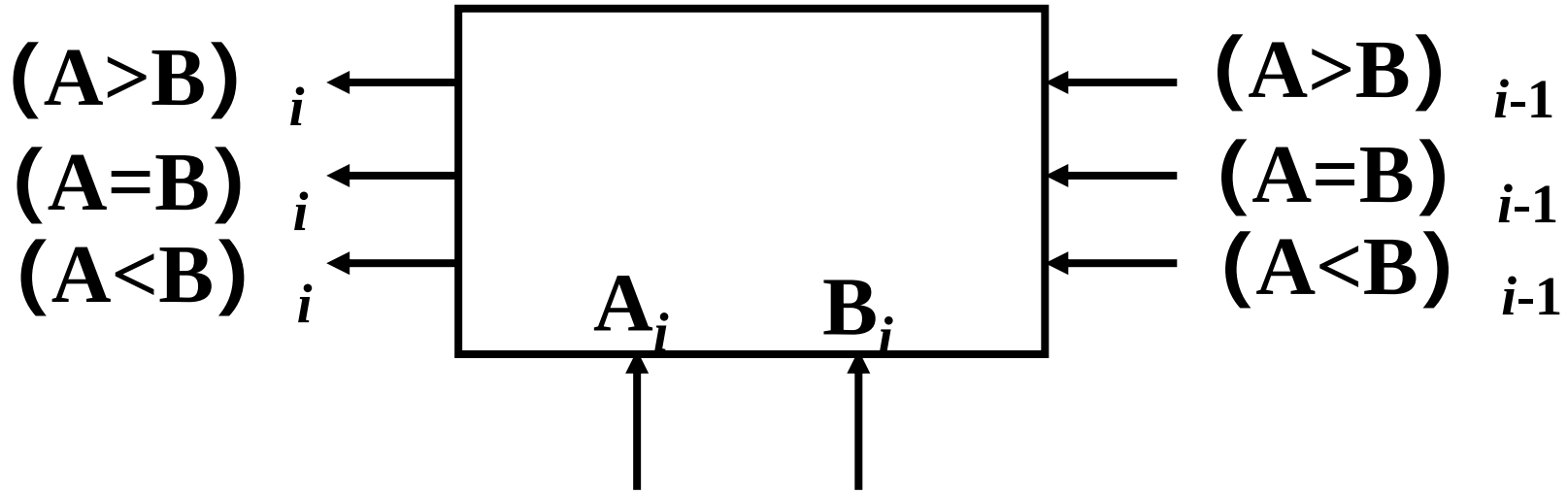
2. 若高位相等，则再比较低位数，最终结果由低位的比较结果决定。

例如：比较925和997

A、B两个多位数的比较：



当两个本位相等时，本位输出的结果由低位的比较结果决定。



输入					输出		
A_i	B_i	$(A>B)_{i-1}$	$(A=B)_{i-1}$	$(A<B)_{i-1}$	$(A>B)_i$	$(A=B)_i$	$(A<B)_i$
1	0	ϕ	ϕ	ϕ	1	0	0
0	1	ϕ	ϕ	ϕ	0	0	1
$A_i=B_i$		输出 $(A>B)_i$ 、$(A=B)_i$ 和$(A<B)_i$ 分别等于$(A>B)_{i-1}$、$(A=B)_{i-1}$ 和$(A<B)_{i-1}$					

四位数码比较器的真值表

比较输入				输出		
$a_3\ b_3$	$a_2\ b_2$	$a_1\ b_1$	$a_0\ b_0$	L (A>B)	E (A=B)	S (A<B)
$a_3>b_3$	×	×	×	1	0	0
$a_3<b_3$	×	×	×	0	0	1
$a_3=b_3$	$a_2>b_2$	×	×	1	0	0
$a_3=b_3$	$a_2<b_2$	×	×	0	0	1
$a_3=b_3$	$a_2=b_2$	$a_1>b_1$	×	1	0	0
$a_3=b_3$	$a_2=b_2$	$a_1<b_1$	×	0	0	1
$a_3=b_3$	$a_2=b_2$	$a_1=b_1$	$a_0>b_0$	1	0	0
$a_3=b_3$	$a_2=b_2$	$a_1=b_1$	$a_0<b_0$	0	0	1
$a_3=b_3$	$a_2=b_2$	$a_1=b_1$	$a_0=b_0$	0	1	0

根据比较规则，可得到四位数码比较器逻辑式：

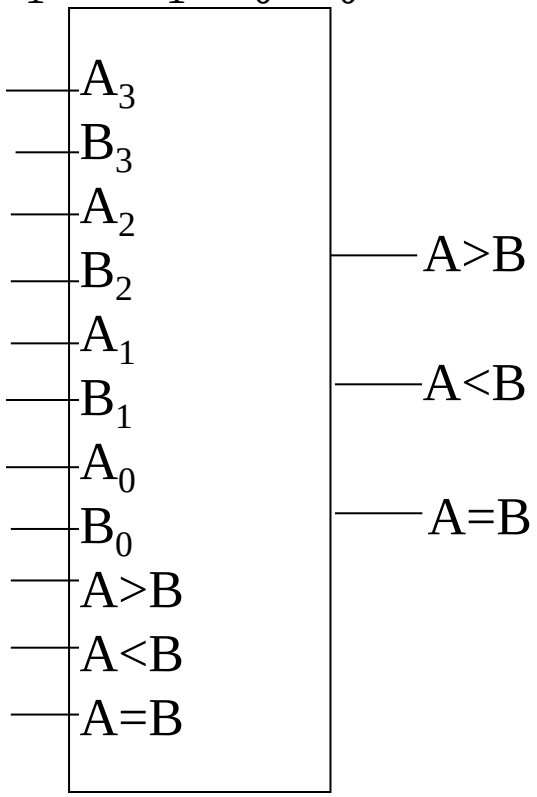
A=B: $E = \overline{A \oplus B} = \overline{(a_3 \oplus b_3)}\,\overline{(a_2 \oplus b_2)}\,\overline{(a_1 \oplus b_1)}\,\overline{(a_0 \oplus b_0)}$

A<B: $S = \overline{a_3}\,b_3 + \overline{(a_3 \oplus b_3)}\,\overline{a_2}\,b_2 + (a_3 \oplus b_3)(a_2 \oplus b_2)\overline{a_1}\,b_1 + (a_3 \oplus b_3)(a_2 \oplus b_2)(a_1 \oplus b_1)\overline{a_0}\,b_0$

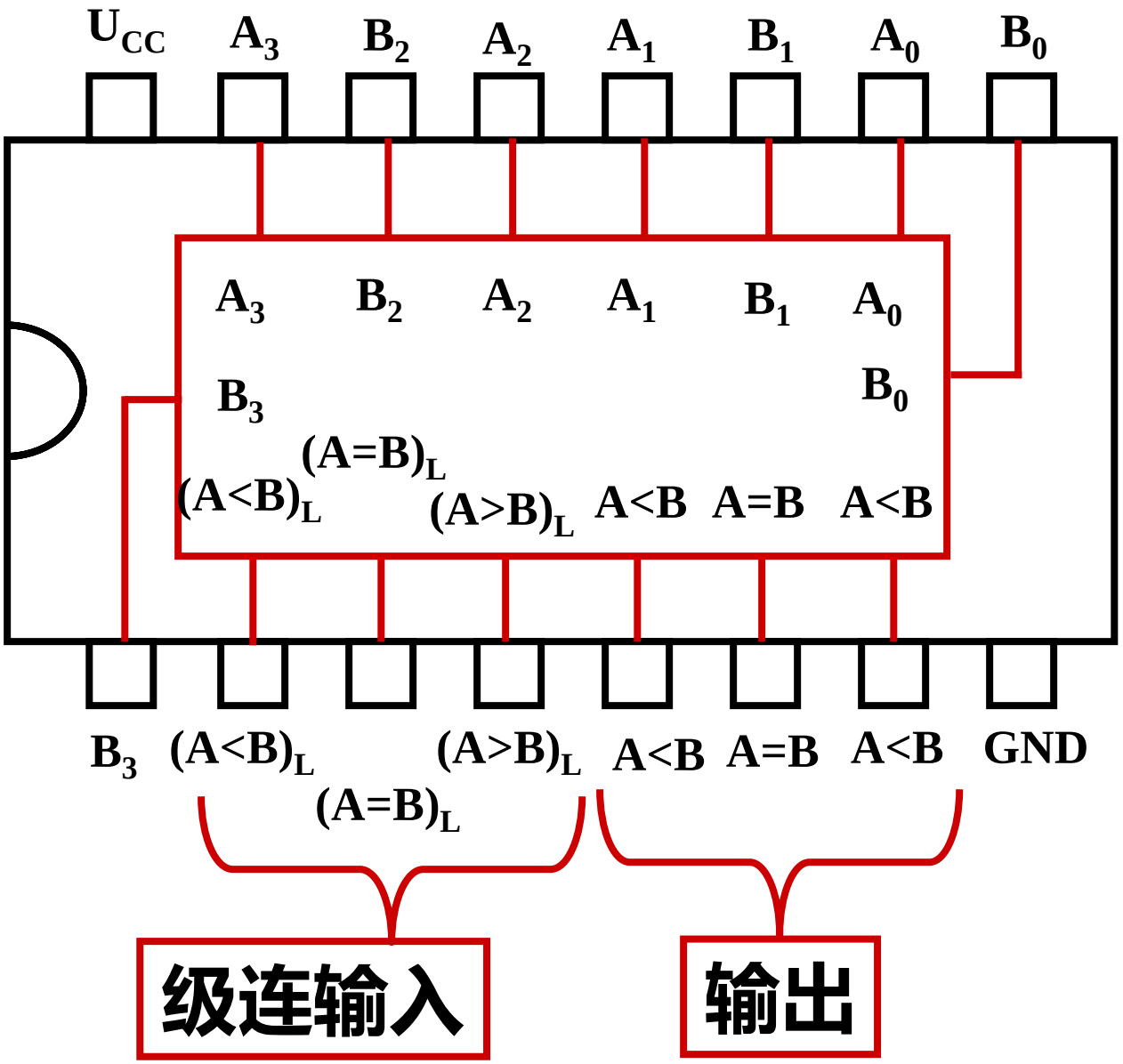
A>B: $L = \overline{E + S}$

共有11个输入端，3个输出端

当有一个输出为1时，另两个输出为0；



四位集成电路比较器74LS85



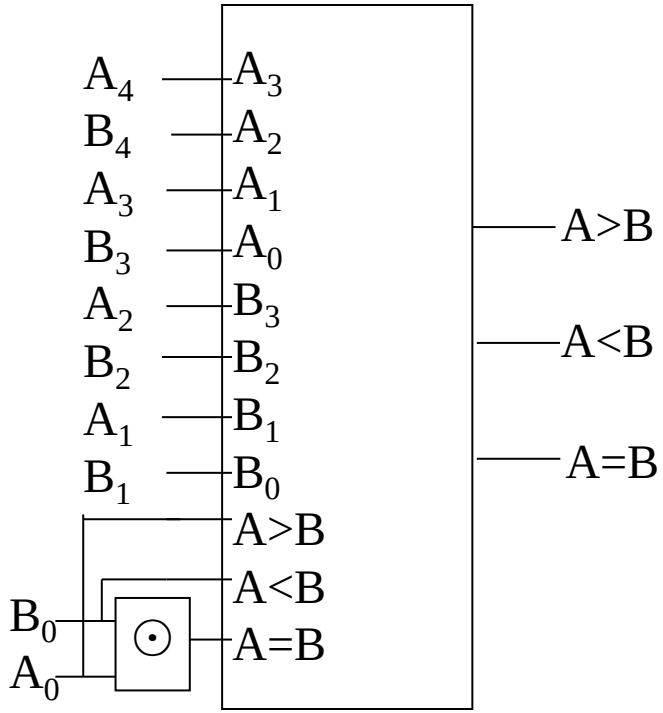
用一片四位数值比较器实现两个五位二进制数的比较。

设两个待比较的数是$A=A_4A_3A_2A_1A_0$、$B=B_4B_3B_2B_1B_0$，将级连输入端作为最低位的数据输入端。当$A_4A_3A_2A_1=B_4B_3B_2B_1$时，两数的比较结果取决于$A_0$、$B_0$，

若$A_0>B_0$，只能是$A_0=1$，$B_0=0$

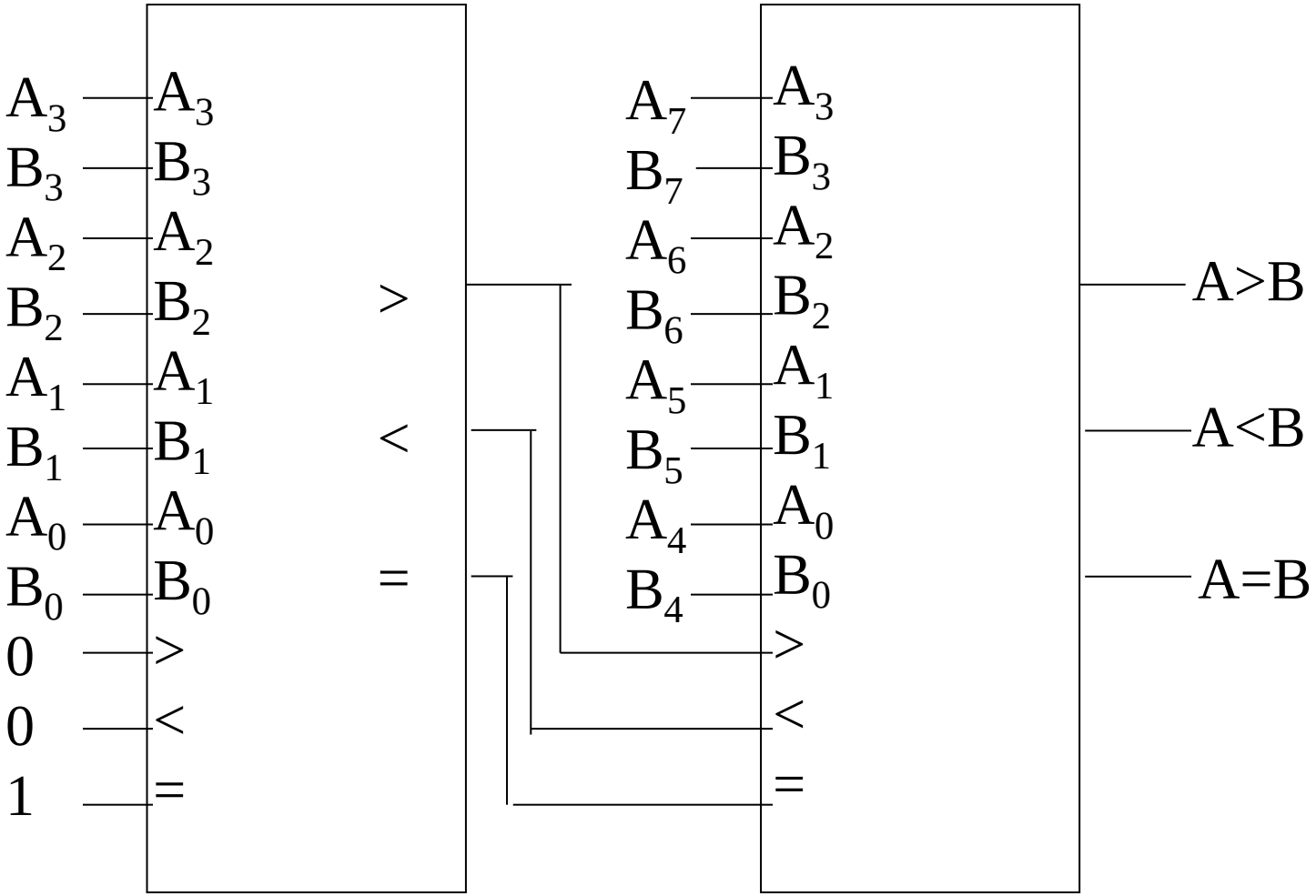
$A_0<B_0$，只能是$A_0=0$，$B_0=1$

$A_0=B_0$，有两种可能$A_0=B_0=1$，$A_0=B_0=0$

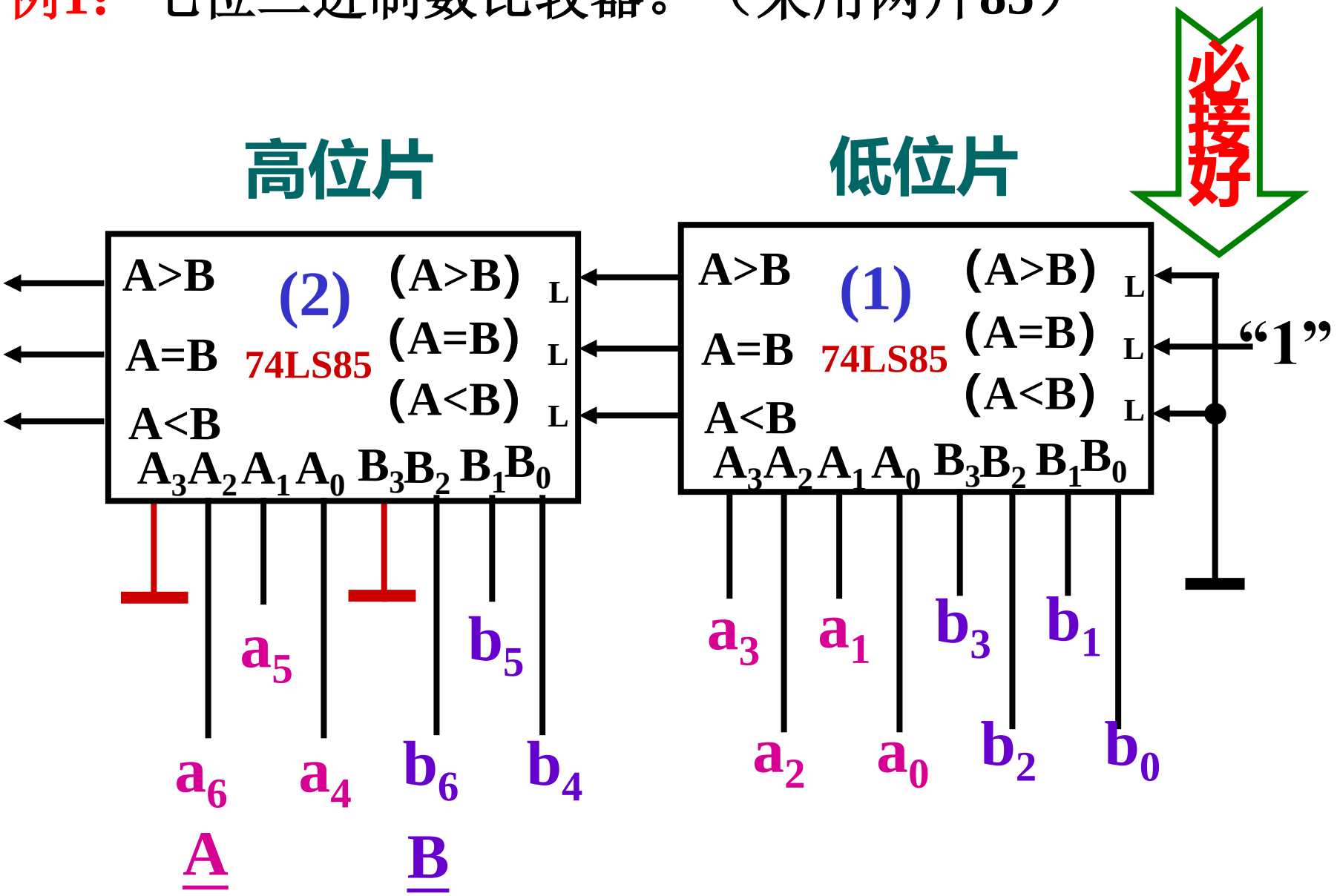


比较器的位数扩展

串联方式扩展比较器的位数



例1：七位二进制数比较器。（采用两片85）



例2： 设计三个四位数的比较器，可以对A、B、C进行比较，能判断：

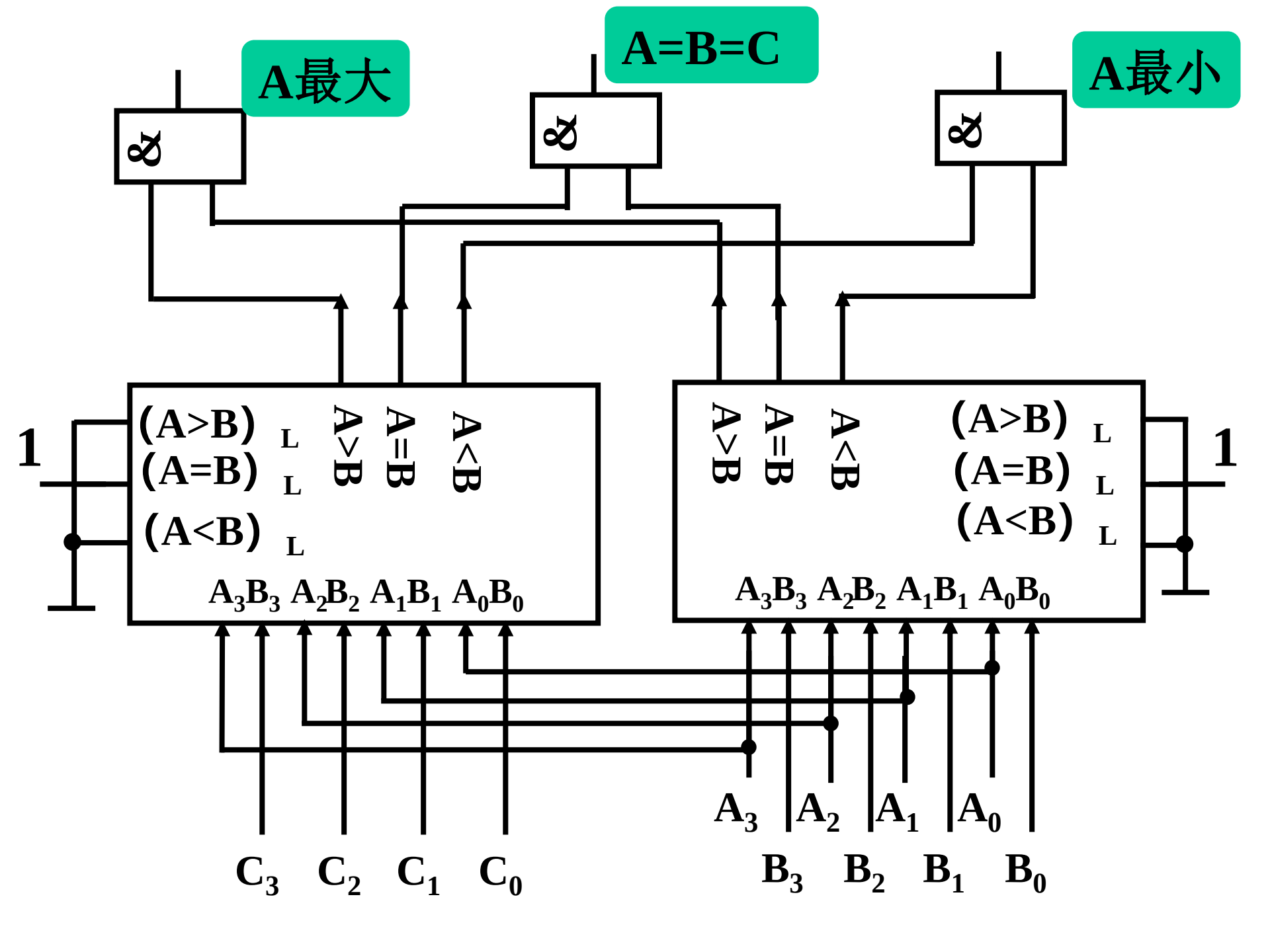
(1) 三个数是否相等。

(2) 若不相等，A数是最大还是最小。

比较原则：

先将A与B比较，然后A与C比较，若A=B A=C，则A=B=C；若A>B A>C，则A最大；若A<B A<C，则A最小。

可以用两片74LS85实现。


A最大
A=B=C
A最小
&
&
&
1
(A>B) L
(A=B) L
(A<B) L
A>B
A=B
A<B
A3B3 A2B2 A1B1 A0B0
A>B
A=B
A<B
(A>B) L
(A=B) L
(A<B) L
A3B3 A2B2 A1B1 A0B0
1
C3 C2 C1 C0
A3 A2 A1 A0
B3 B2 B1 B0

4.5 加法器

举例：A=1101, B=1001，
计算A+B。

```
   1 1 0 1
+  1 0 0 1
  1 0 0 1
-----------
 1 0 1 1 0
```

加法运算的基本规则：

(1) 逢二进一。

用半加器实现

(2) 最低位是两个数的最低位的叠加，没有低位进位。

(3) 其余各位都是三个数相加，包括被加数、加数和低位来的进位。

用全加器实现

(4) 任何位相加都可能产生两个结果：本位和、向高位的进位。

一、半加器

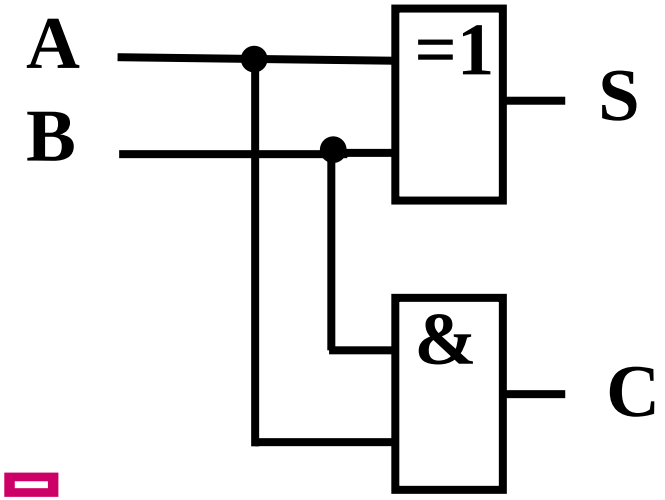
半加运算不考虑从低位来的进位。设：

A---被加数； B---加数； S---本位和； C---进位。

真值表

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

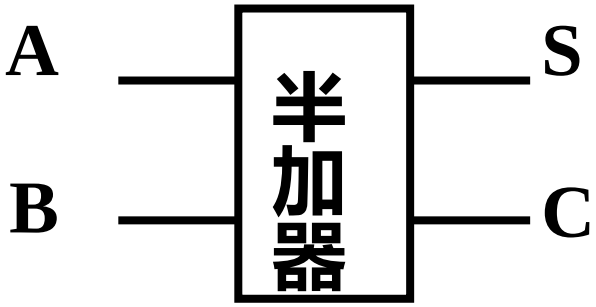
逻辑图



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

$$C = AB$$

逻辑符号



二、全加器：

a_n---被加数；b_n---加数；c_{n-1}---低位的进位；s_n---本位和；c_n---进位。

真值表

a_n	b_n	c_{n-1}	s_n	c_n
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

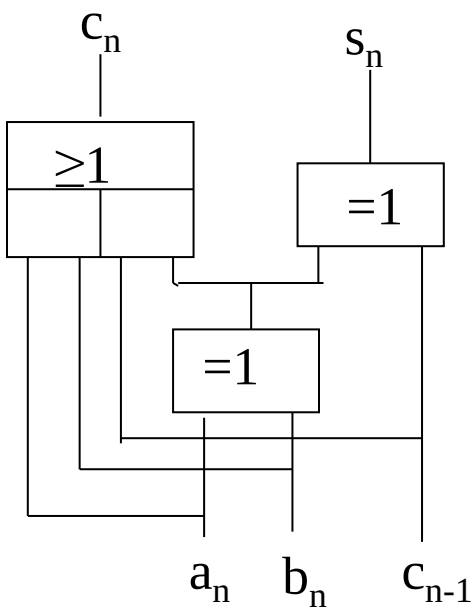
$$s_n = a_n \oplus b_n \oplus c_{n-1}$$

$$c_n = a_n b_n + (a_n \oplus b_n) c_{n-1}$$

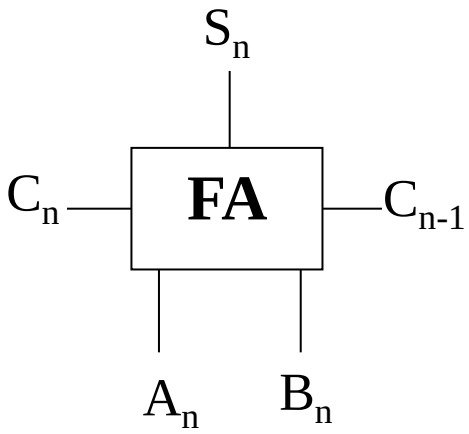
a_n \ b_n		1		1
	1		1	

C_{n-1}

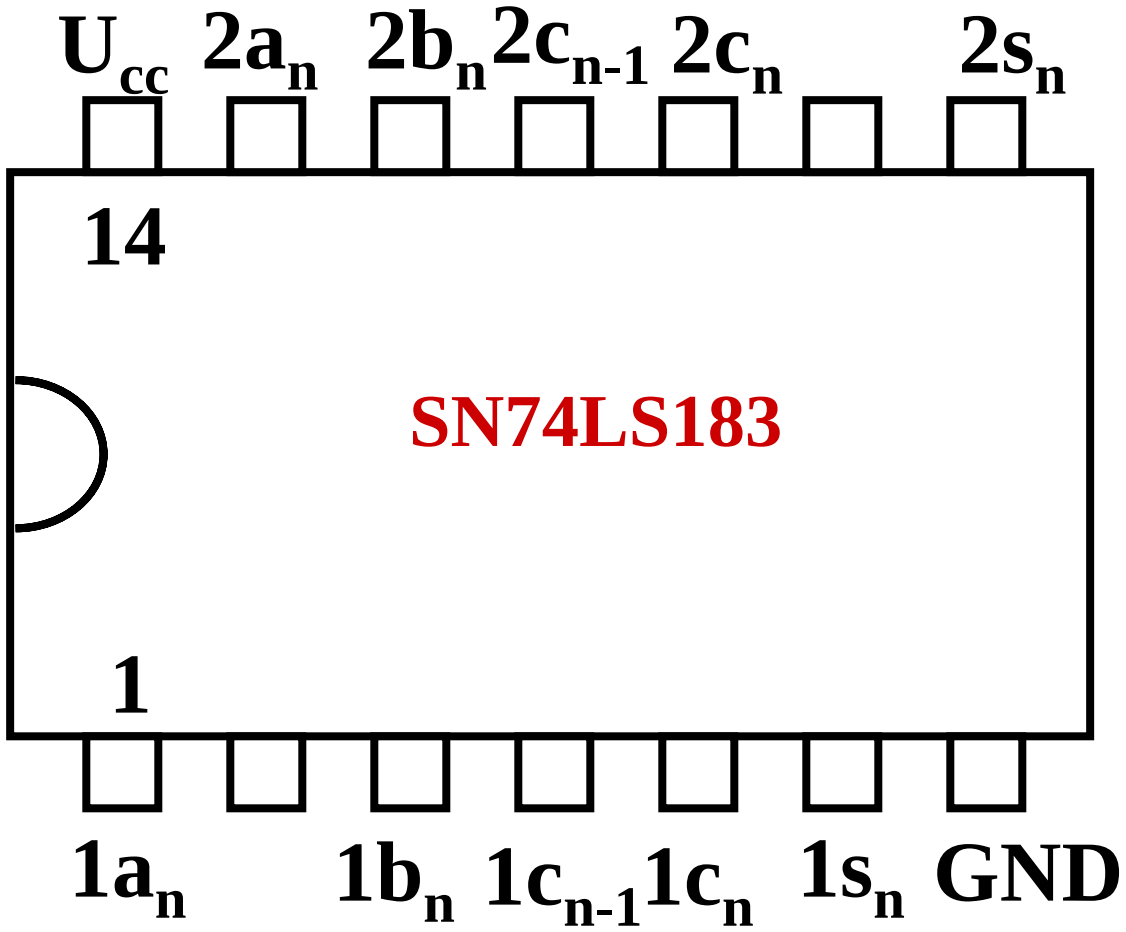
逻辑图



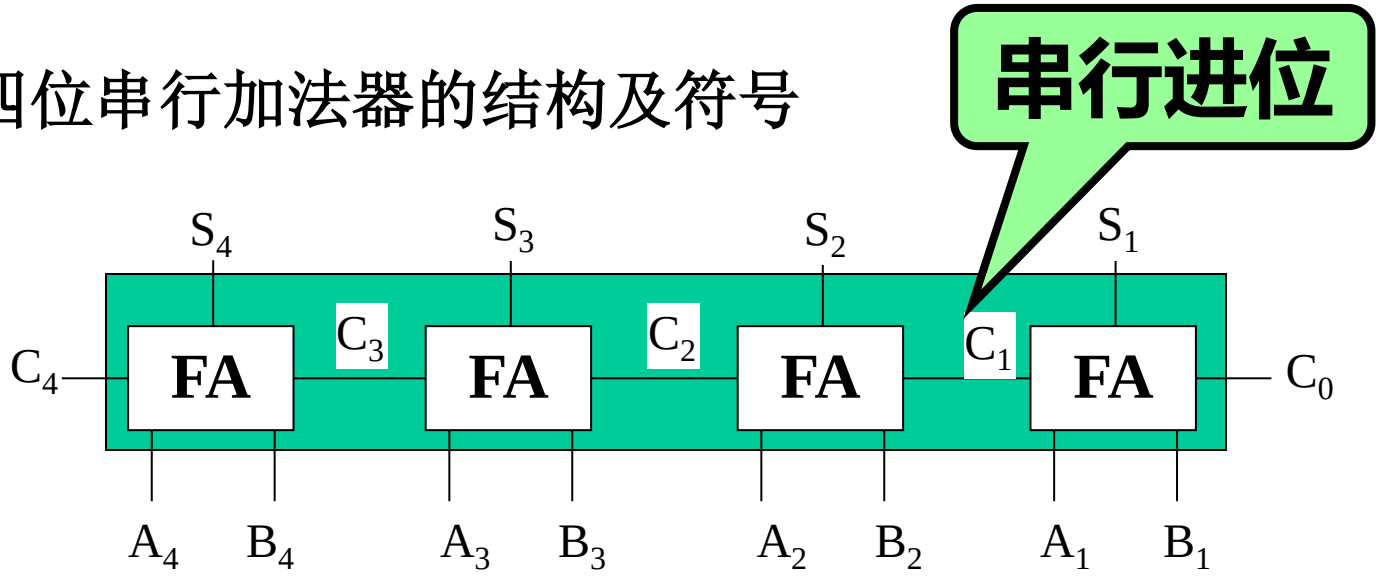
逻辑符号

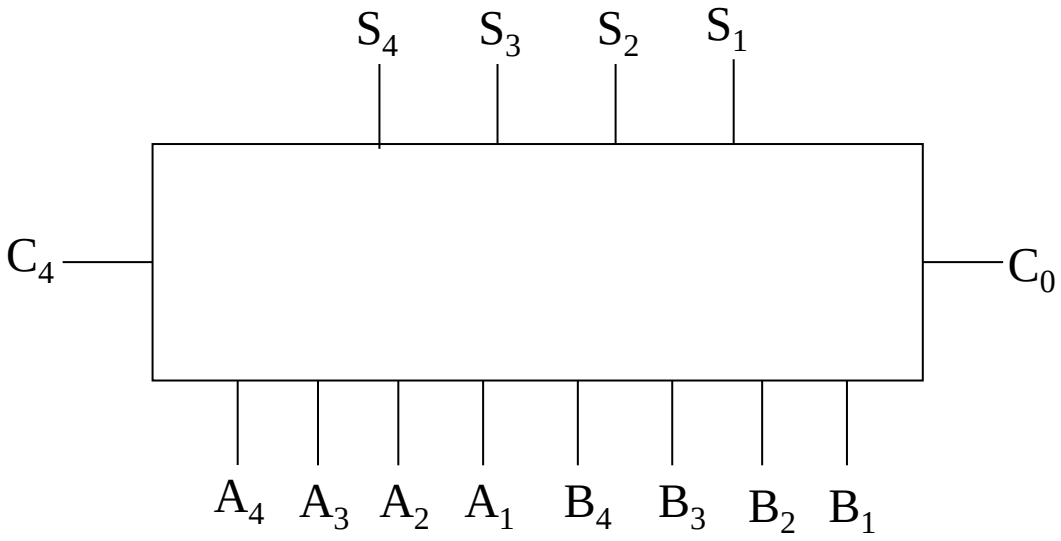


全加器SN74LS183的管脚图



四位串行加法器的结构及符号





进位按逐级串行传输方式进行，由于各个进位的产生依赖于低位的进位，因此，运算速度慢。

超前进位加法器：各位的进位直接由被加数和加数决定。

∵ $C_i=(A_i \oplus B_i)C_{i-1}+A_iB_i$　　令$P_i=A_i \oplus B_i$　$G_i=A_iB_i$

则$C_i=P_iC_{i-1}+G_i$　　当i=1，2，3，4时进位输出为

$C_1=P_1C_0+G_1$，$C_2=P_2C_1+G_2$，$C_3=P_3C_2+G_3$，$C_4=P_4C_3+G_4$，

上述表达式经代入整理后得：

$C_1=P_1C_0+G_1$

$C_2=P_2P_1C_0+P_2G_1+G_2$

$C_3=P_3P_2P_1C_0+P_3P_2G_1+P_3G_2+G_3$

$C_4=P_4P_3P_2P_1C_0+P_4P_3P_2G_1+P_4P_3G_2+P_4G_3+G_4$

可见，各位进位输出，都不依赖低位的进位，仅取决于A_i、B_i和C_0，一般情况下C_0在运算前已预置，使得各位的进位能同时产生，从而提高了运算速度。

超前进位加法器：各位的进位直接由被加数和加数决定。

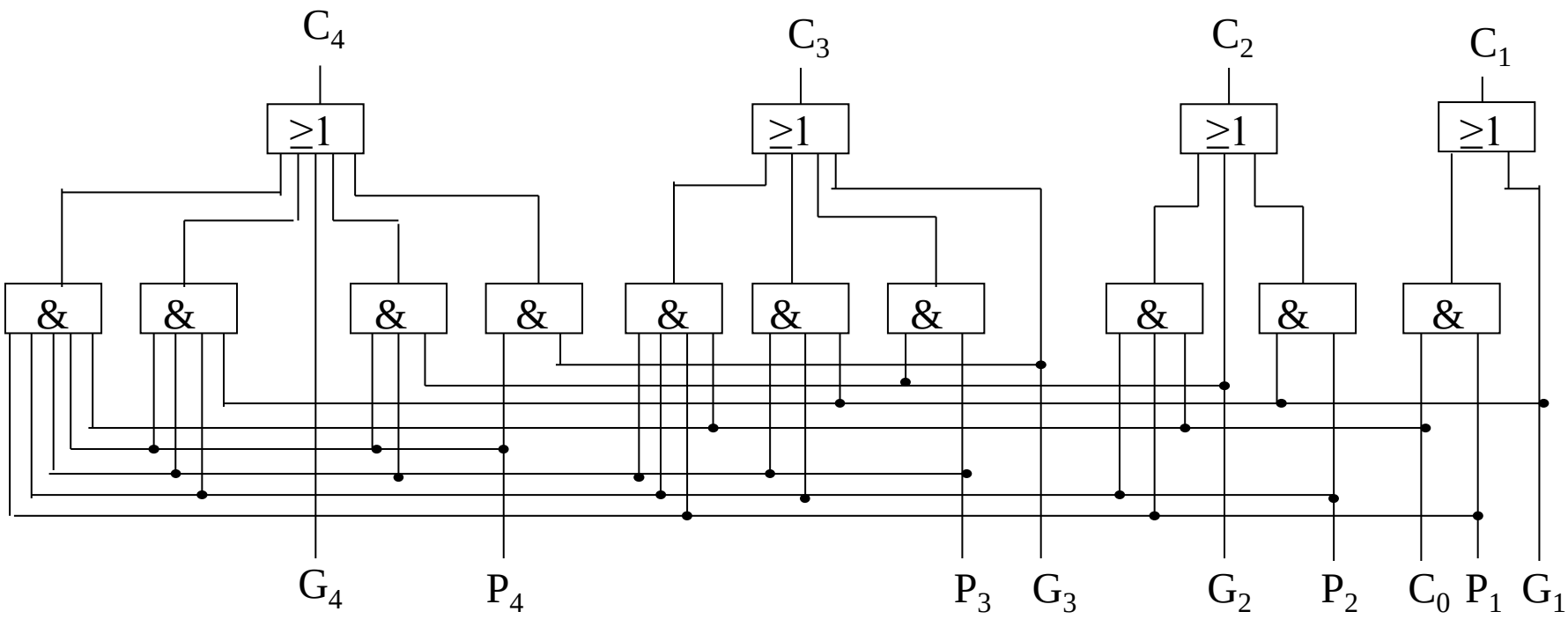
$C_1=P_1C_0+G_1$

$C_2=P_2P_1C_0+P_2G_1+G_2$

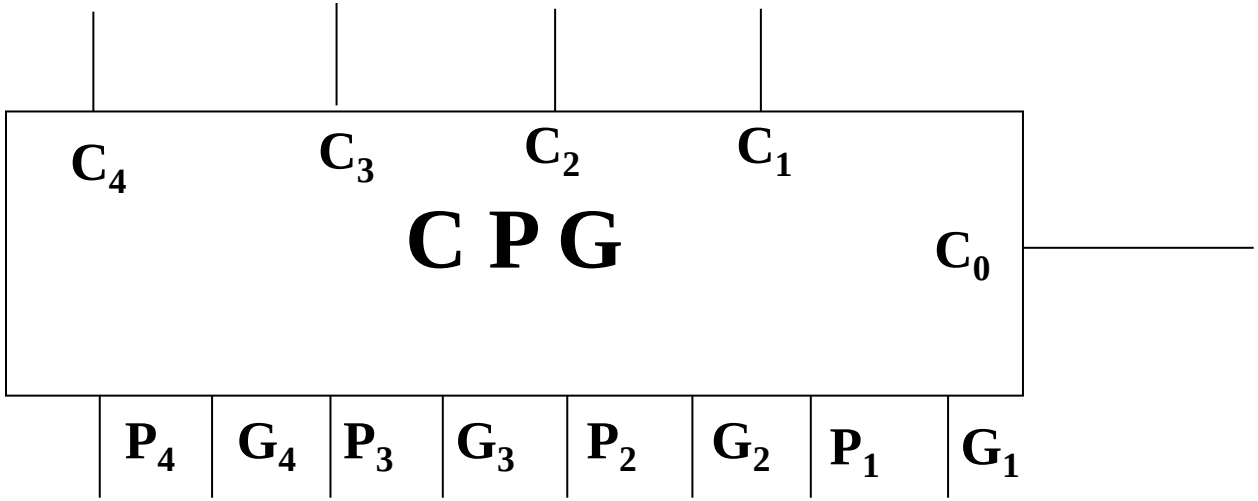
$C_3=P_3P_2P_1C_0+P_3P_2G_1+P_3G_2+G_3$

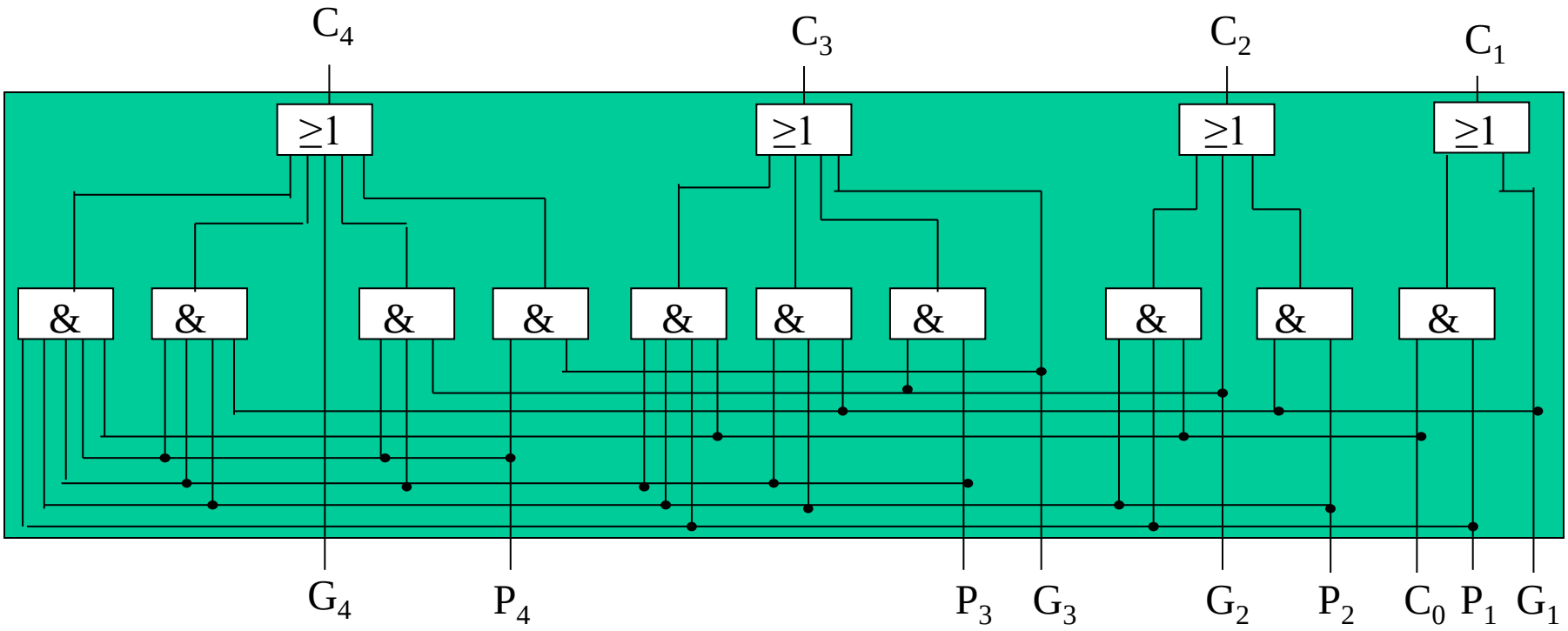
$C_4=P_4P_3P_2P_1C_0+P_4P_3P_2G_1+P_4P_3G_2+P_4G_3+G_4$

四位二进制串行加法器的先行进位逻辑图CPG

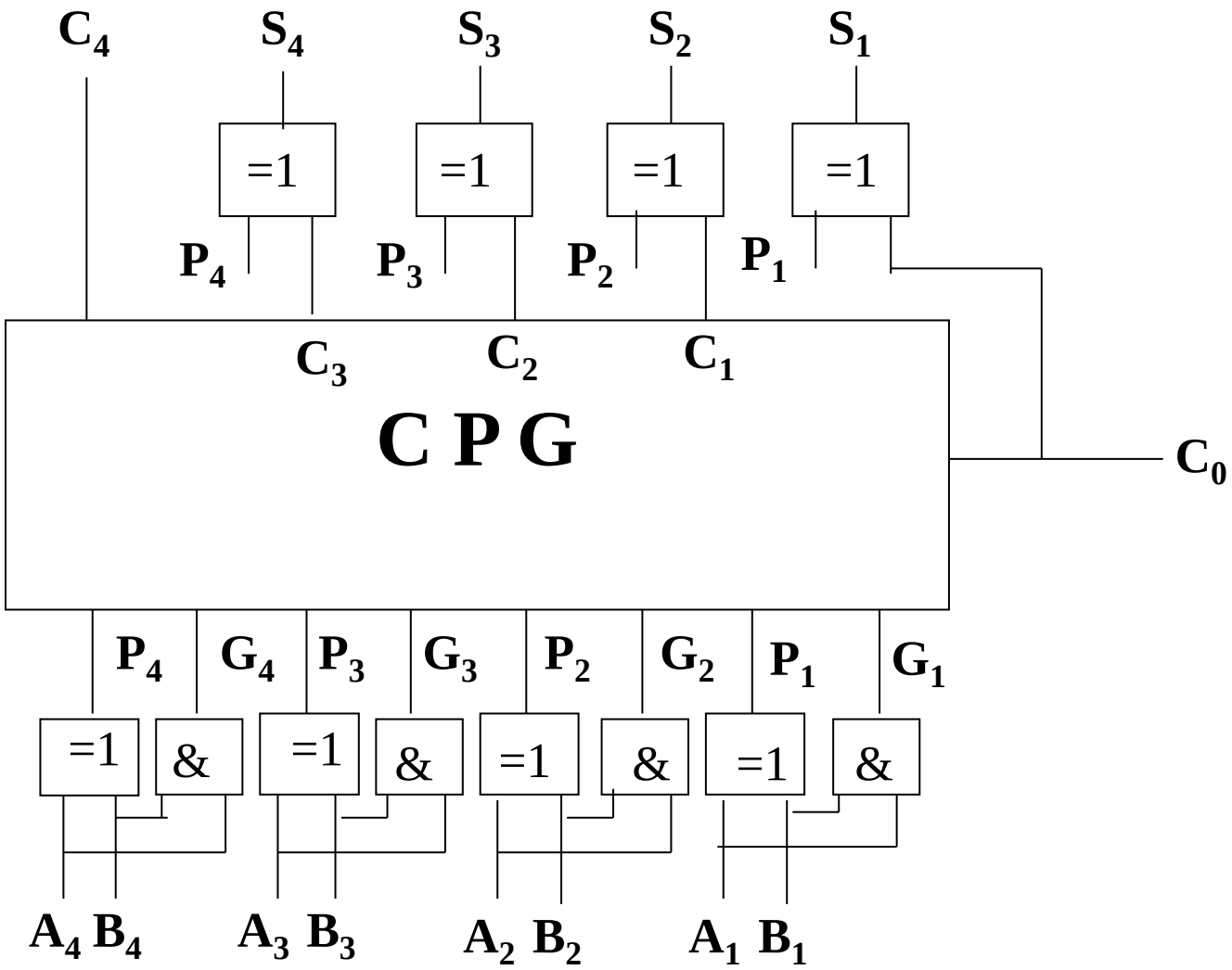


超前进位加法器：各位的进位直接由被加数和加数决定。





超前进位加法器：各位的进位直接由被加数和加数决定。



第四章结束

谢谢