第4章　常用组合逻辑功能器件

本章将介绍几种常用的中规模集成电路(MSI),这些中规模集成电路分别具有特定的逻辑功能,称为功能模块,用功能模块设计组合逻辑电路,具有许多优点。

4.1 自顶向下的模块化设计方法

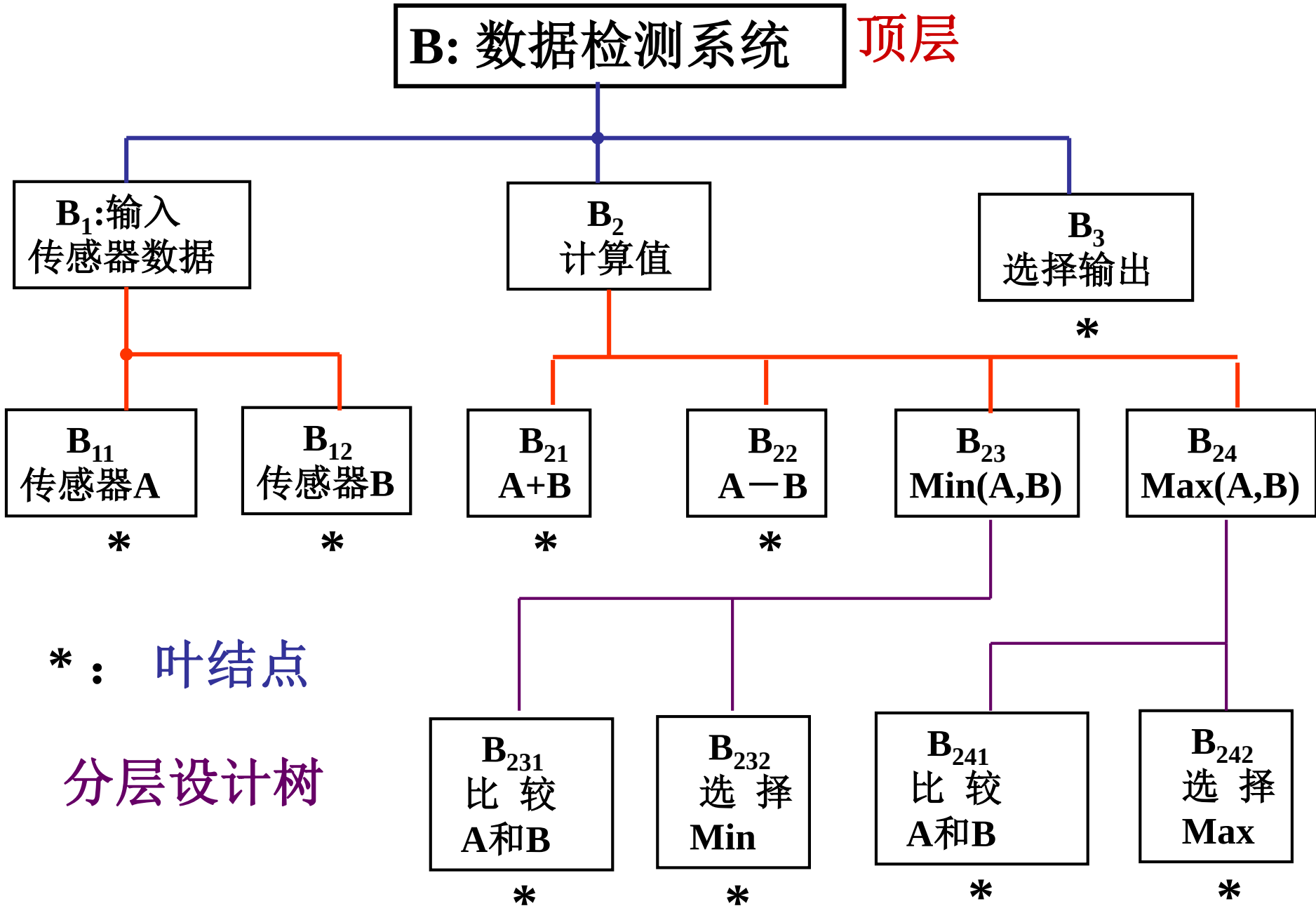
顶: 指系统功能,即系统总要求,较抽象.

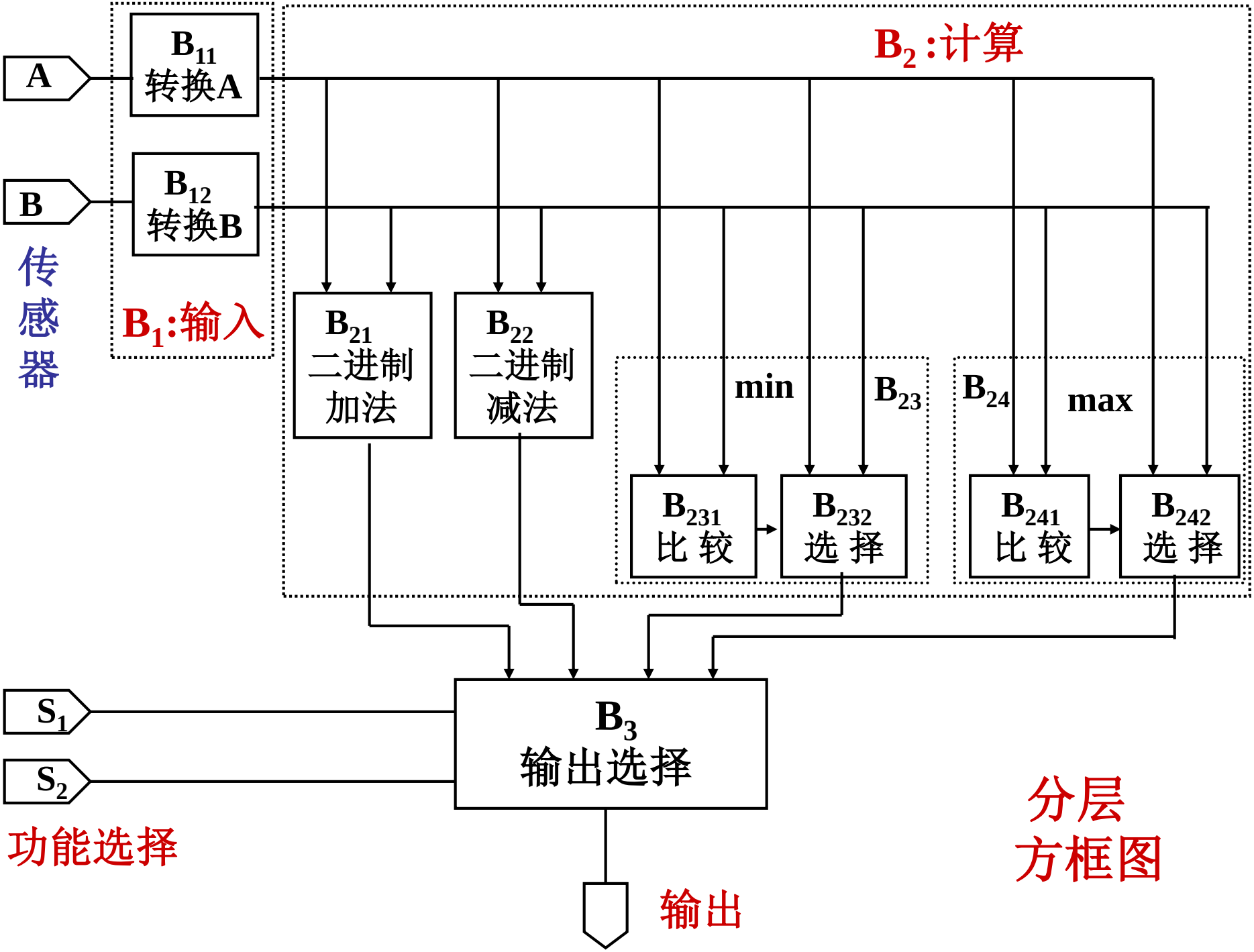
向下:指根据系统总要求,将系统分解为若干个子系统,再将每个子系统分解为若干个功能模块… …,直至分成许多各具特定功能的基本模块为止.

例: 设计一个数据检测系统,功能表如下:

数据A、B分别来自两个传感器.

S_1	S_2	输出功能
0	0	A + B
0	1	A－B
1	0	Min(A,B)
1	1	Max(A,B)


B: 数据检测系统
顶层
B1:输入
传感器数据
B2
计算值
B3
选择输出
*
B11
传感器A
*
B12
传感器B
*
B21
A+B
*
B22
A－B
*
B23
Min(A,B)
B24
Max(A,B)
B231
比 较
A和B
*
B232
选 择
Min
*
B241
比 较
A和B
*
B242
选 择
Max
*
*： 叶结点
分层设计树


A
B
传感器
B11
转换A
B12
转换B
B1:输入
B2 :计算
B21
二进制
加法
B22
二进制
减法
min
B23
B24
max
B231
比 较
B232
选 择
B241
比 较
B242
选 择
S1
S2
功能选择
B3
输出选择
输出
分层
方框图

4.2 编码器

将信息(如数和字符等)转换成符合一定规则的二进制代码.

4.2.1 二进制编码器

用n位二进制代码对$N=2^n$个特定信息进行编码的逻辑电路.

设计方法: 以例说明

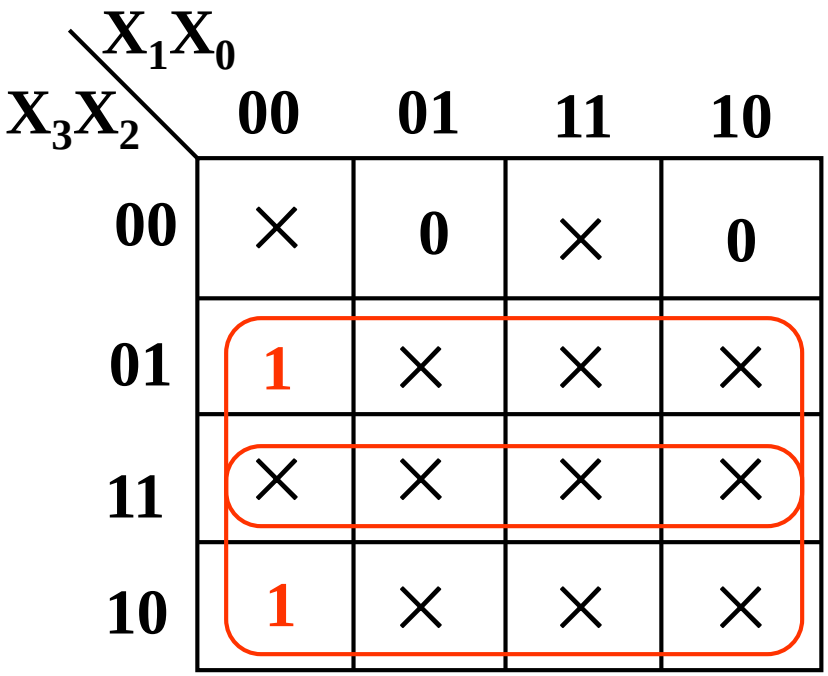
设计一个具有互相排斥输入条件的编码器.

输入：X_0、X_1、X_2、X_3　　输出：A_1、A_0

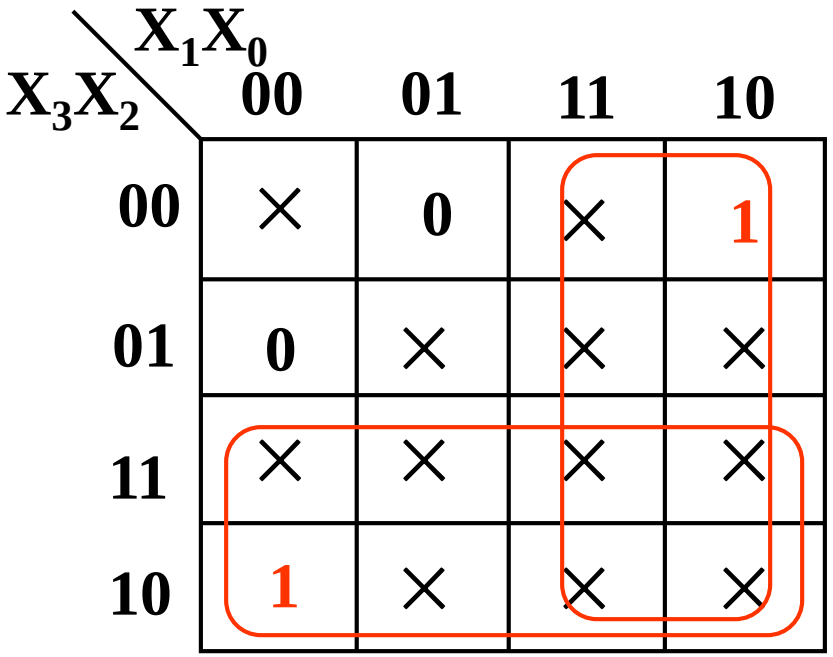
对应关系：

输入	A_1	A_0
X_0	0	0
X_1	0	1
X_2	1	0
X_3	1	1

X_3	X_2	X_1	X_0	A_1	A_0
0	0	0	0	×	×
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	×	×
0	1	0	0	1	0
0	1	0	1	×	×
0	1	1	0	×	×
0	1	1	1	×	×
1	0	0	0	1	1
1	0	0	1	×	×
1	0	1	0	×	×
1	0	1	1	×	×
1	1	0	0	×	×
1	1	0	1	×	×
1	1	1	0	×	×
1	1	1	1	×	×



$A_1=X_2+X_3$



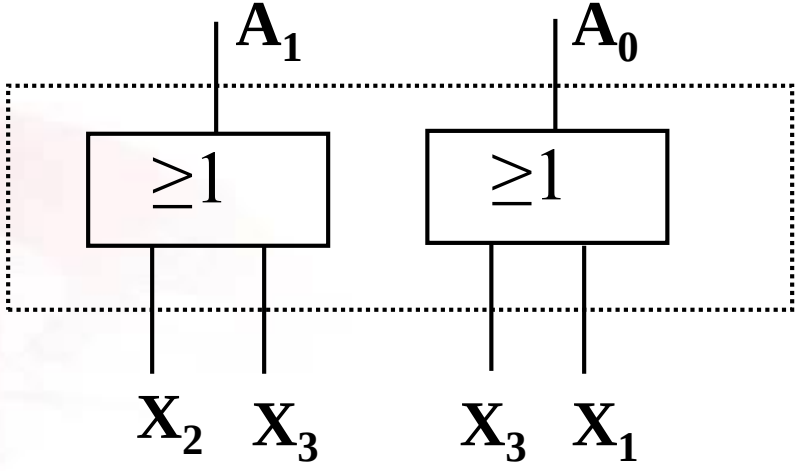
$A_0=X_1+X_3$

4线—2线编码器电路图：

(1) 编码器在任何时候只允许有一个输入信号有效;

(2) 电路无X_0输入端;

(3) 电路无输入时,编码器的输出与X_0编码等效.



带输出使能(Enable)端的优先编码器:

输出使能端：用于判别电路是否有信号输入.

优先：对输入信号按轻重缓急排序,当有多个信号同时输入时,只对优先权高的一个信号进行编码.

下面把上例4线—2线编码器改成带输出使能(Enable)端的优先编码器,假设输入信号优先级的次序为:X_3,X_2,X_1,X_0.

X_3	X_2	X_1	X_0	A_1	A_0	E0
0	0	0	0	0	0	1
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	1	1	0
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	1	1	0
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	0

X_3X_2 \ X_1X_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

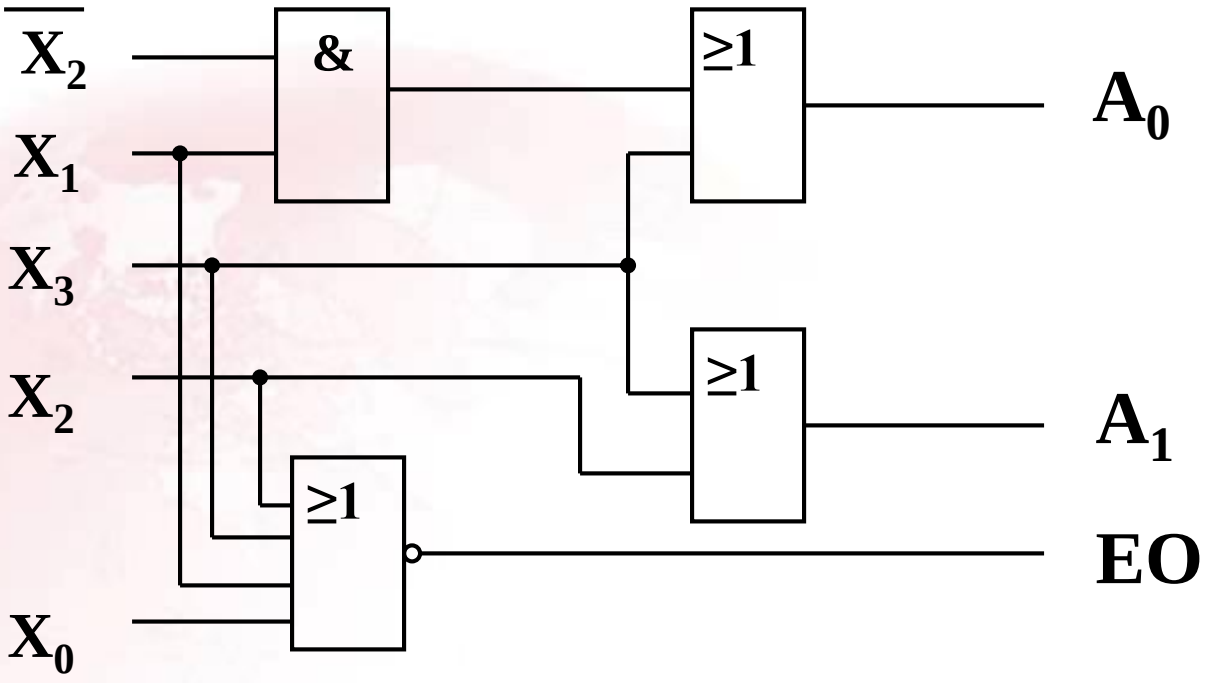
$A_1=X_2+X_3$

X_3X_2 \ X_1X_0	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$A_0=X_3+\overline{X}_2X_1$

$$EO=\overline{X_3}\,\overline{X_2}\,\overline{X_1}\,\overline{X_0}=\overline{X_3+X_2+X_1+X_0}$$

编码器
电路图



4.2.2 二—十进制编码器

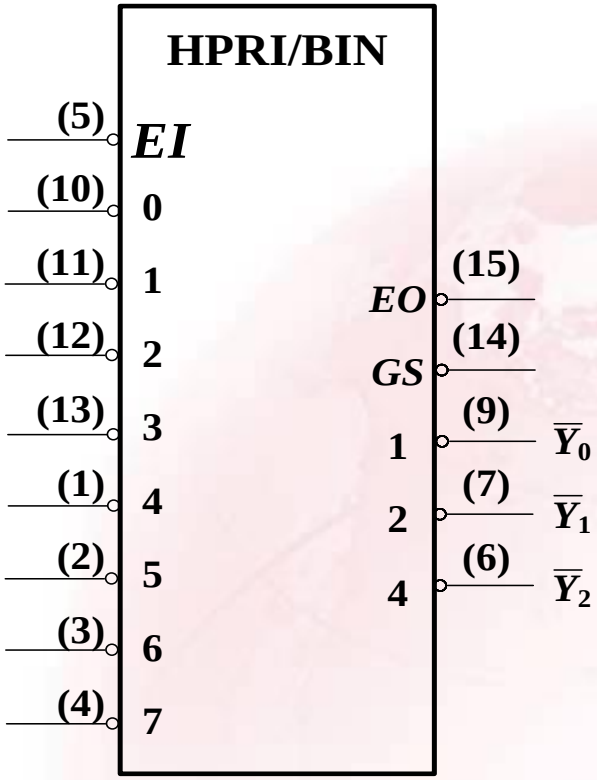
输入: I_0 ,I_1 ,I_2 … …I_9,表示十个要求编码的信号.

输出: BCD码.

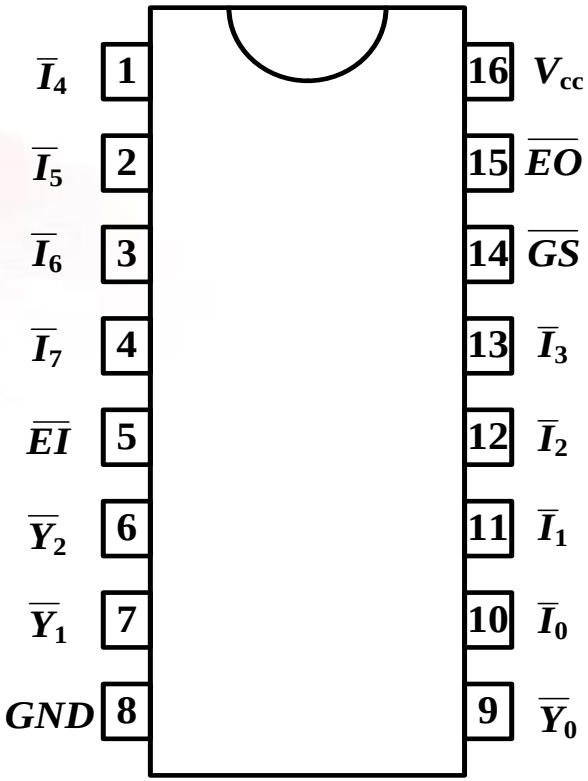
电路有十根输入线,四根输出线,常称为10线—4线编码器

4.2.3 通用编码器集成电路

1. 8线—3线优先编码器74148



逻辑图



引脚图

74148功能说明：

1）74148为8线—3线优先编码器，HPRI是最高位优先编码器的说明。

2）编码器输入为低电平有效，输出为3位二进制反码。

3）$\overline{EI}$端为输入使能端,当$\overline{EI}$ =0时,电路处于正常工作状态; 当$\overline{EI}$ =1时,电路禁止工作, $\overline{Y}_2\overline{Y}_1\overline{Y}_0$=111 。

4）$\overline{EO}$为选通输出端.

$$\overline{EO}=\overline{EI\ \overline{I}_0\overline{I}_1\overline{I}_2\overline{I}_3\overline{I}_4\overline{I}_5\overline{I}_6\overline{I}_7}$$

当$\overline{EI}$=0(即正常工作时),若编码输入信号I_i均为1(即无编码信号输入),则$\overline{EO}$=0。说明当$\overline{EO}$=0时,电路在工作状态，但无编码信号输入. 这时$\overline{Y}_2\overline{Y}_1\overline{Y}_0$=111.

5) $\overline{GS}$ 为扩展输出端：

$$\overline{GS}=\overline{EI(I_0+I_1+I_2+I_3+I_4+I_5+I_6+I_7)}$$

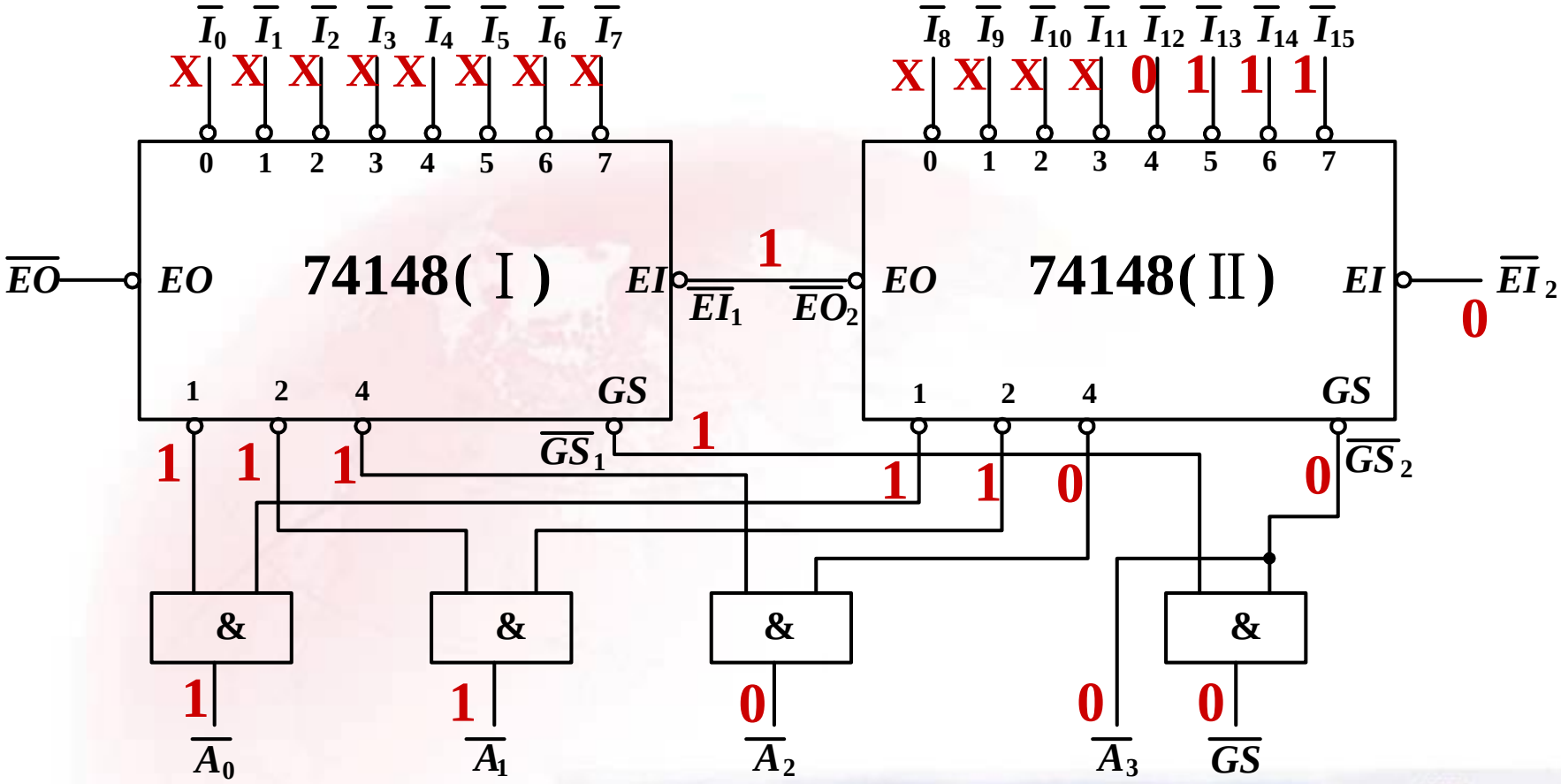
当$\overline{EI}$=0(即正常工作时),若有编码信号输入(即至少有一个$\overline{I}_i$为0),则$\overline{GS}$=0。说明当$\overline{GS}$=0时,电路在工作状态, 而且有编码信号输入。

74148功能表

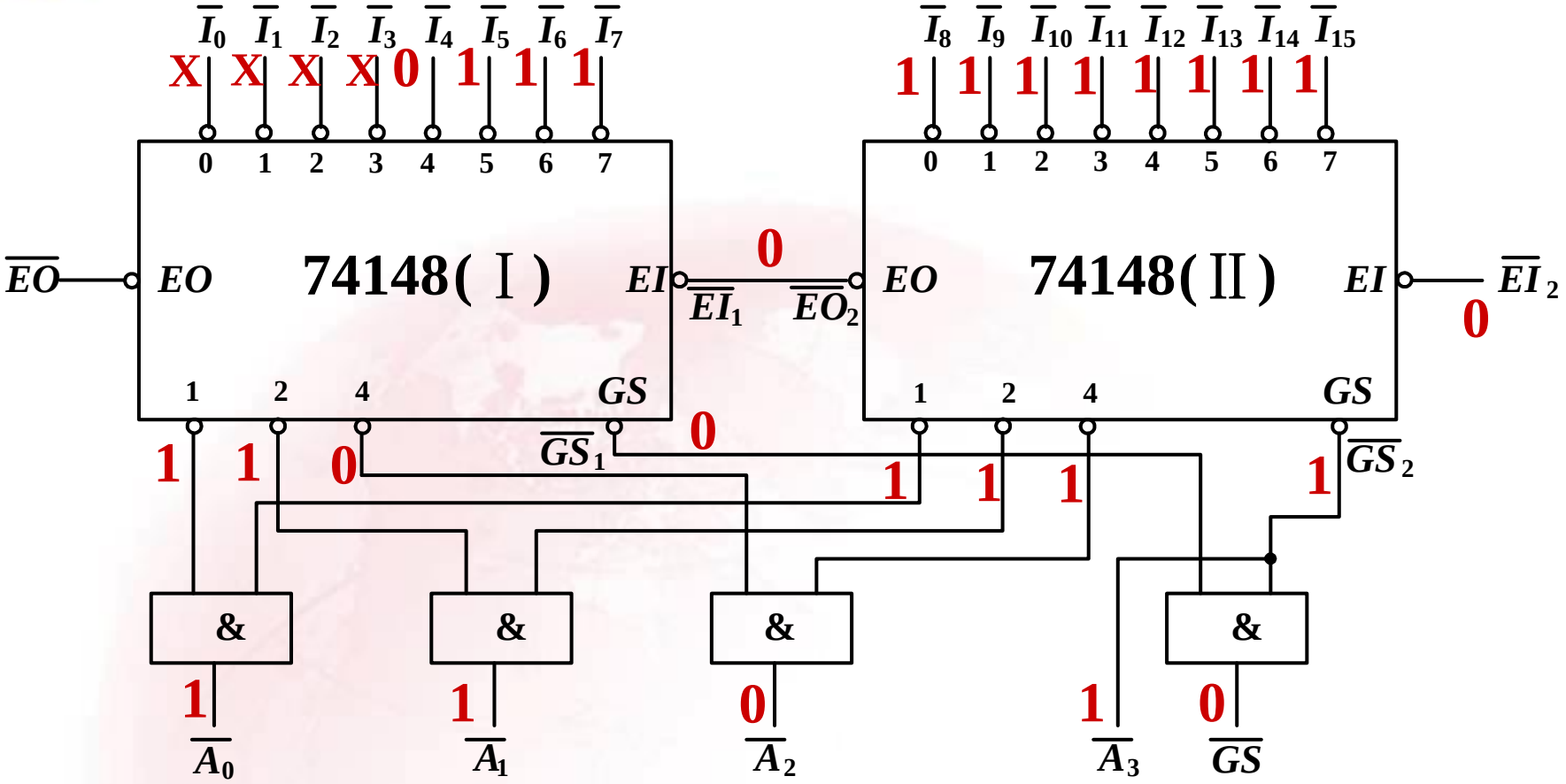
输入									输出				
$\overline{EI}$	$\overline{I_0}$	$\overline{I_1}$	$\overline{I_2}$	$\overline{I_3}$	$\overline{I_4}$	$\overline{I_5}$	$\overline{I_6}$	$\overline{I_7}$	$\overline{Y_2}$	$\overline{Y_1}$	$\overline{Y_0}$	$\overline{GS}$	$\overline{EO}$
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	×	×	×	×	×	×	×	0	0	0	0	0	1
0	×	×	×	×	×	×	0	1	0	0	1	0	1
0	×	×	×	×	×	0	1	1	0	1	0	0	1
0	×	×	×	×	0	1	1	1	0	1	1	0	1
0	×	×	×	0	1	1	1	1	1	0	0	0	1
0	×	×	0	1	1	1	1	1	1	0	1	0	1
0	×	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

例：用两片74148构成16线—4线优先编码器。

高位芯片工作情况：



低位芯片工作情况：


$\overline{I_0}$ $\overline{I_1}$ $\overline{I_2}$ $\overline{I_3}$ $\overline{I_4}$ $\overline{I_5}$ $\overline{I_6}$ $\overline{I_7}$
X X X X 0 1 1 1
0 1 2 3 4 5 6 7
$\overline{EO}$
EO
74148(Ⅰ)
EI
1 2 4 GS
1 1 0
$\overline{GS_1}$
0
$\overline{EI_1}$
$\overline{EO_2}$
$\overline{I_8}$ $\overline{I_9}$ $\overline{I_{10}}$ $\overline{I_{11}}$ $\overline{I_{12}}$ $\overline{I_{13}}$ $\overline{I_{14}}$ $\overline{I_{15}}$
1 1 1 1 1 1 1 1
0 1 2 3 4 5 6 7
EO
74148(Ⅱ)
EI
$\overline{EI_2}$
0
1 2 4 GS
1 1 1
$\overline{GS_2}$
1
&
&
&
&
1
$\overline{A_0}$
1
$\overline{A_1}$
0
$\overline{A_2}$
1
$\overline{A_3}$
0
$\overline{GS}$

问题思考：若用四片74148构成一个32线—5线编码器，电路如何设计？

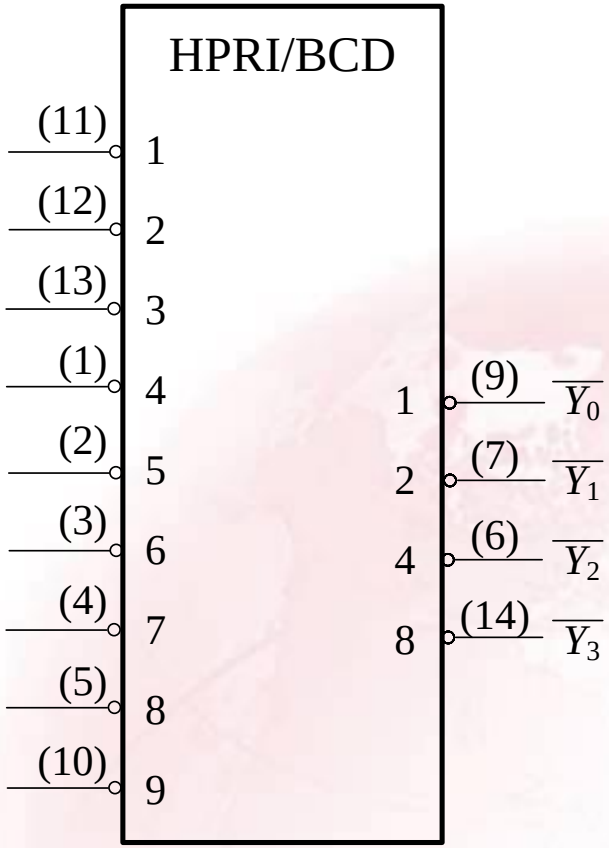
若用八片74148构成一个64线—6线编码器，电路又如何设计？

扩展电路设计提示：

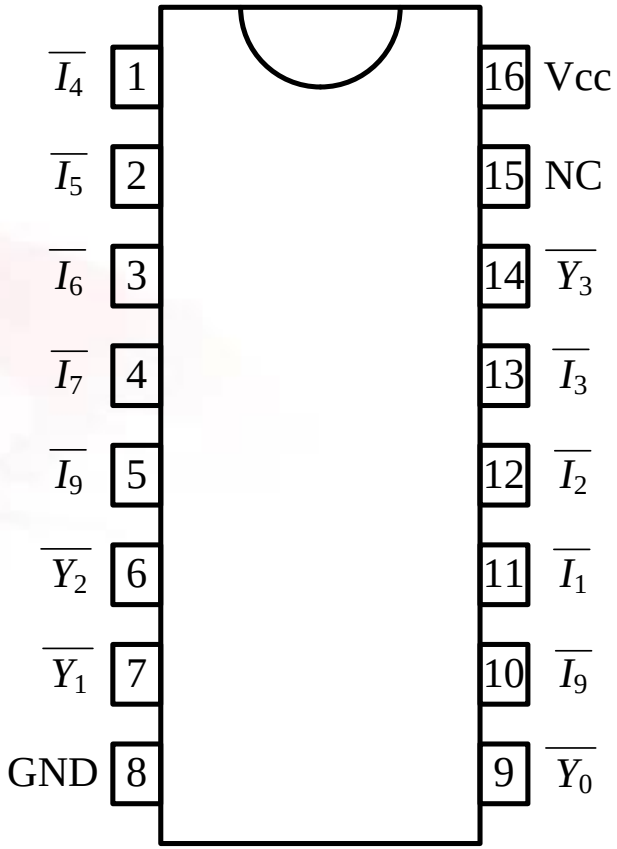
1）观察上例编码器低三位输出电路结构，并找出规律；

2）分析高位输出和各$\overline{GS}$之间的关系，将$\overline{GS}$作为输入，高位信号作为输出，设计一输出电路。

2. 10线－4线优先编码器74147



逻辑图

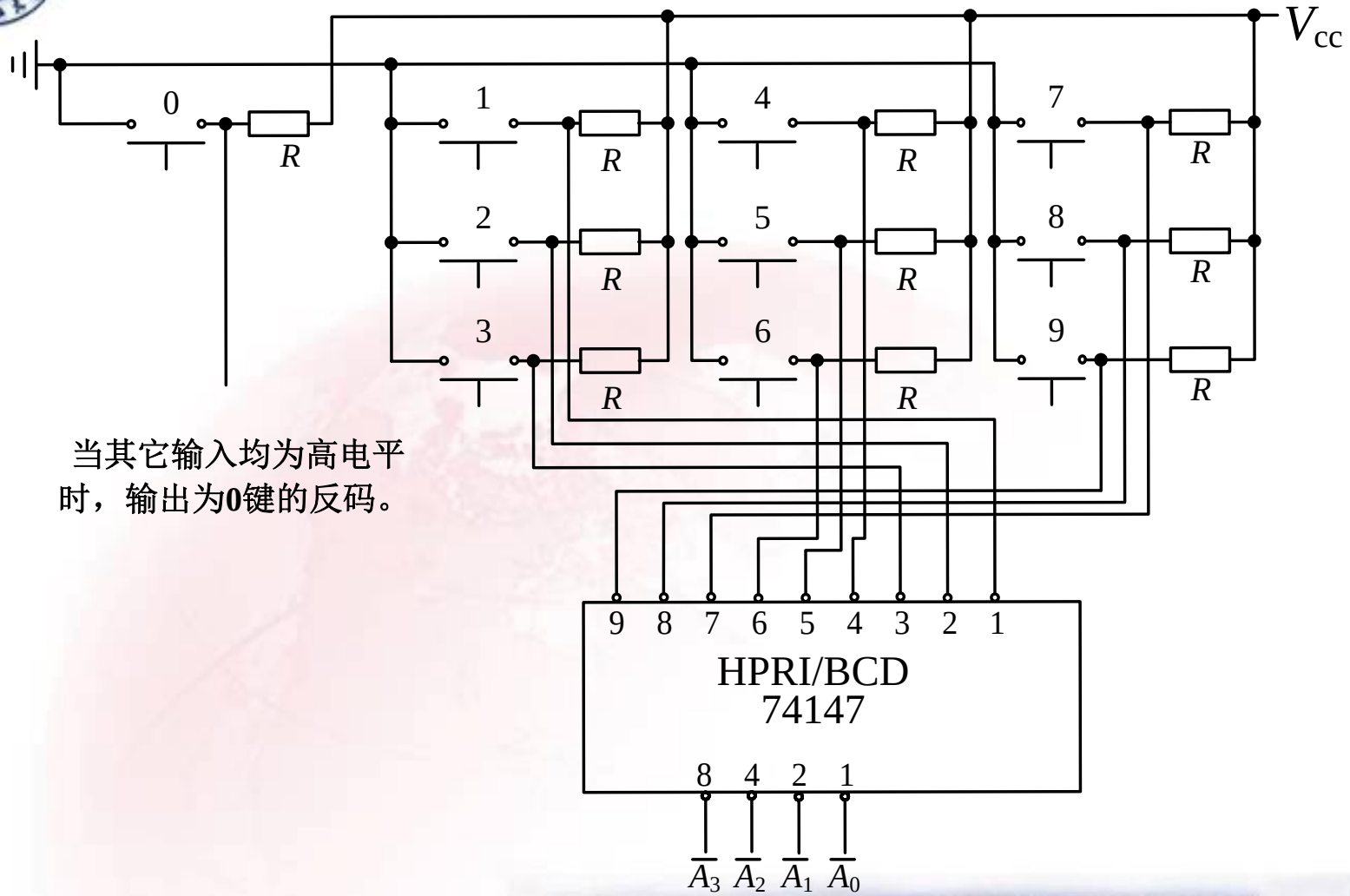


引脚图

74147功能表

十进制数	输入									输出			
	$\overline{I_1}$	$\overline{I_2}$	$\overline{I_3}$	$\overline{I_4}$	$\overline{I_5}$	$\overline{I_6}$	$\overline{I_7}$	$\overline{I_8}$	$\overline{I_9}$	$\overline{Y_3}$	$\overline{Y_2}$	$\overline{Y_1}$	$\overline{Y_0}$
0	1	1	1	1	1	1	1	1	1	1	1	1	1
9	×	×	×	×	×	×	×	×	0	0	1	1	0
8	×	×	×	×	×	×	×	0	1	0	1	1	1
7	×	×	×	×	×	×	0	1	1	1	0	0	0
6	×	×	×	×	×	0	1	1	1	1	0	0	1
5	×	×	×	×	0	1	1	1	1	1	0	1	0
4	×	×	×	0	1	1	1	1	1	1	0	1	1
3	×	×	0	1	1	1	1	1	1	1	1	0	0
2	×	0	1	1	1	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	1	0

4.2.4 编码器应用举例



当其它输入均为高电平时，输出为**0**键的反码。

4.2.5 编码器的VHDL描述

一个普通编码器的例子：

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY encoder IS
PORT(input  : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
           y  : OUT STD_LOGIC_VECTOR(2 DOWNTO 0));
END encoder;
ARCHITECTURE rtl OF encoder IS
BEGIN
PROCESS(input)
  BEGIN
```

```
CASE input IS
        WHEN “01111111”=>y<=“111”;
        WHEN “10111111”=>y<=“110”;
        WHEN “11011111”=>y<=“101”;
        WHEN “11101111”=>y<=“100”;
        WHEN “11110111”=>y<=“011”;
        WHEN “11111011”=>y<=“010”;
        WHEN “11111101”=>y<=“001”;
        WHEN “11111110”=>y<=“000”;
        WHEN OTHERS=>y<=“XXX”;
    END CASE;
 END PROCESS;
END rtl;
```

一个优先编码器的例子：

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY priorityencoder IS
PORT( input:  IN STD_LOGIC_VECTOR(7 DOWNTO 0);
            y:  OUT STD_LOGIC_VECTOR(2 DOWNTO 0));
END priorityencoder;
ARCHITECTURE rtl OF priorityencoder IS
BEGIN
PROCESS(input)
BEGIN
   IF (input(7)='0') THEN
      y<= “111”;
```

```
    ELSIF (input(6)='0') THEN y<= “110”;
    ELSIF (input(5)='0') THEN y<= “101”;
    ELSIF (input(4)='0') THEN y<= “100”;
    ELSIF (input(3)='0') THEN y<= “011”;
    ELSIF (input(2)='0') THEN y<= “010”;
    ELSIF (input(1)='0') THEN y<= “001”;
    ELSE  y<= “000”;
    END IF;
  END PROCESS;
END rtl;
```

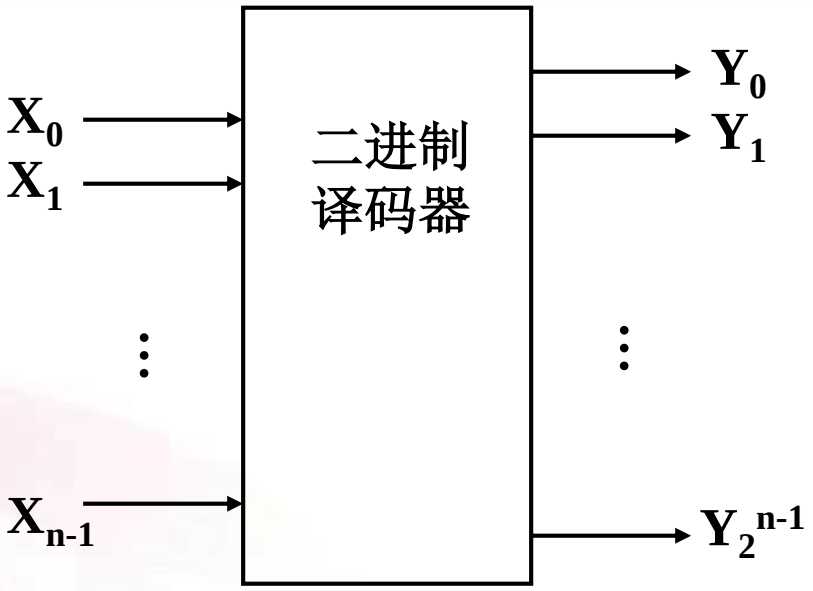
4.3 译码器/数据分配器

译码是编码的逆过程，作用是将一组码转换为确定信息。

4.3.1 二进制译码器

输入：二进制代码，有n个；

输出：2^n 个特定信息。



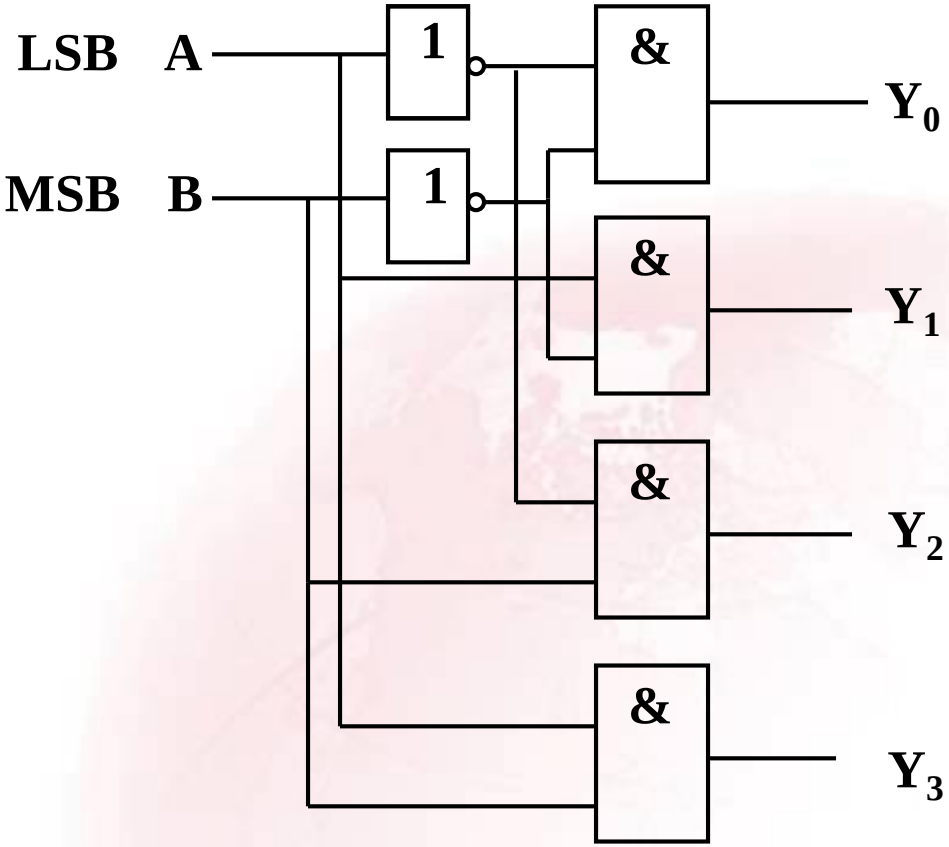
1. 译码器电路结构

以2线— 4线译码器为例说明

2线— 4线译码器的真值表为:

B	A	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

下图为高电平输出有效的2线– 4线译码器电路图，



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

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

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

$Y_3=BA=m_3$

由真值表容易得出：

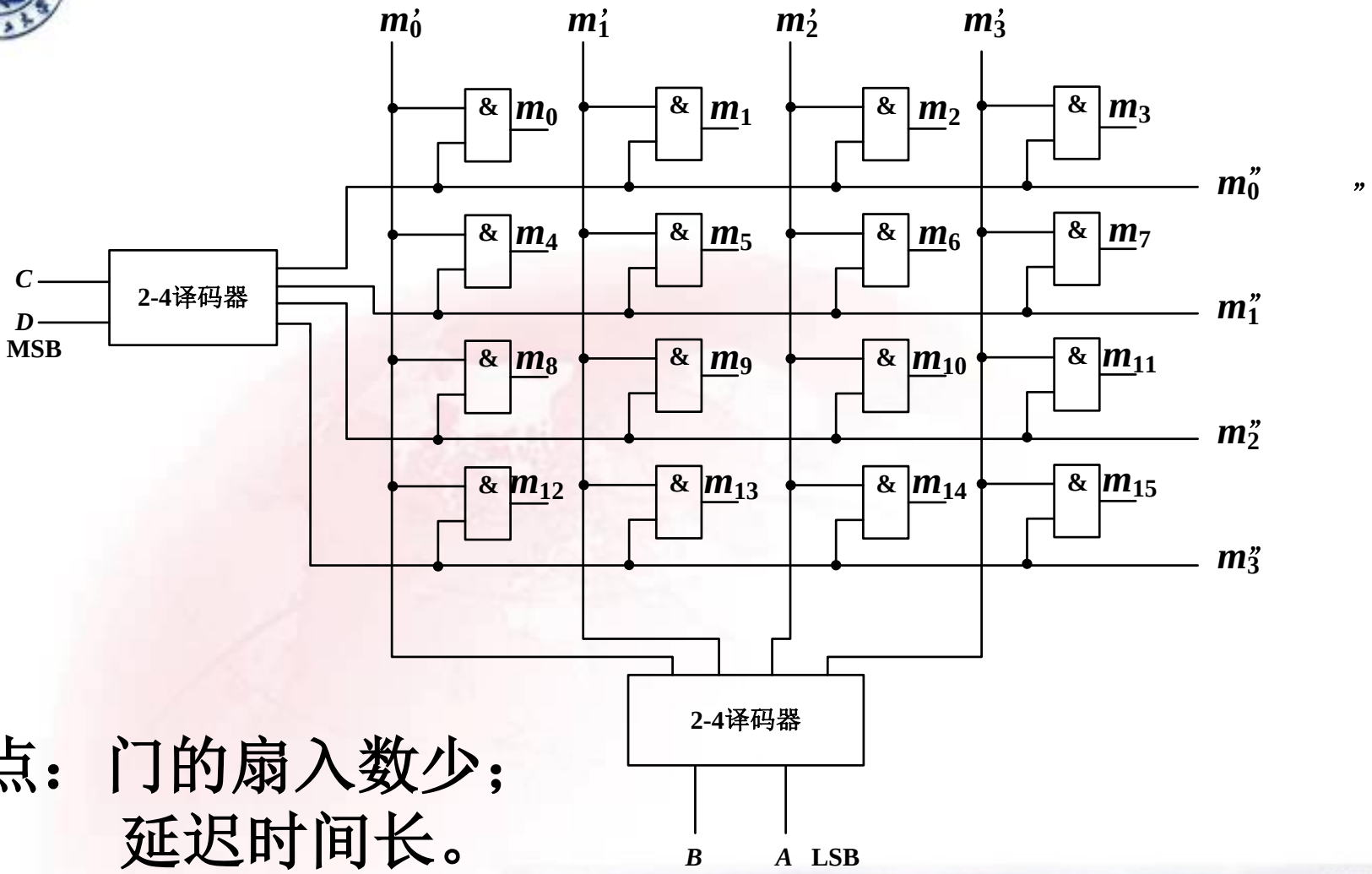
① 高电平输出有效二进制译码器,其输出逻辑表达式为：

$Y_i=m_i$ (m_i为输入变量所对应的最小项)

② 低电平输出有效二进制译码器,其输出逻辑表达式为：

$\overline{Y}_i=\overline{m}_i$ (m_i为输入变量所对应的最小项)

译码器的另一种结构：矩阵式结构



特点：门的扇入数少；
延迟时间长。

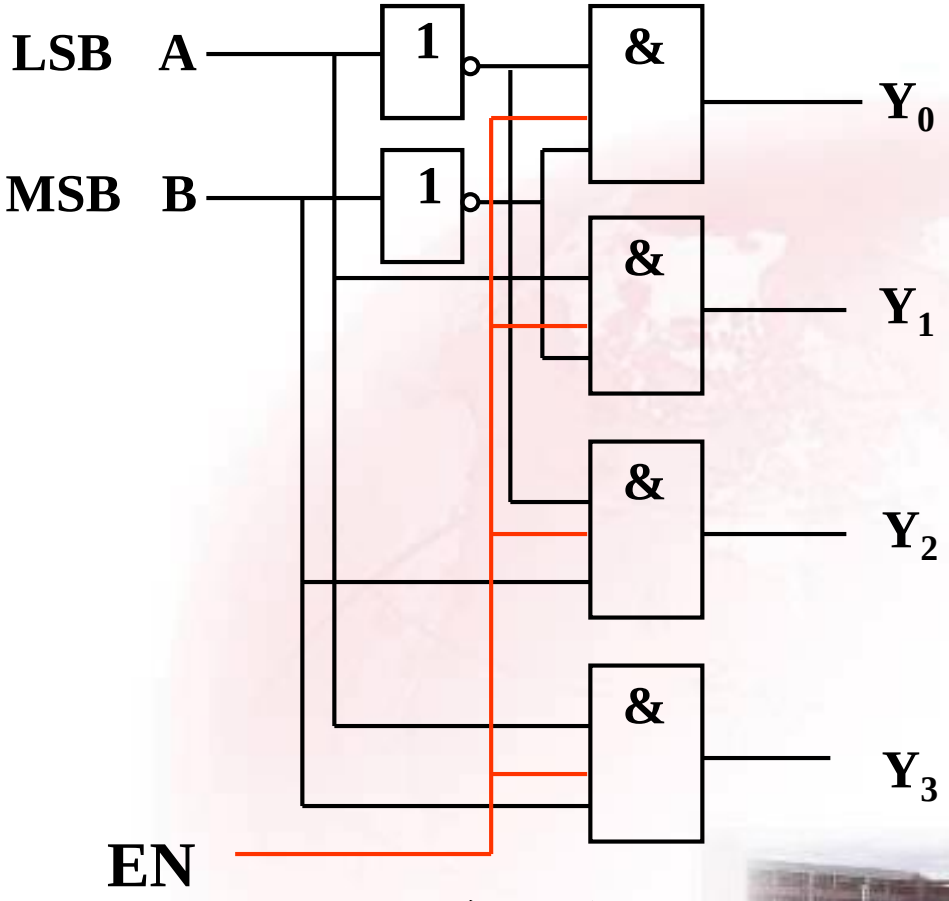
2. 译码器的使能控制输入端

1）利用使能输入控制端，既能使电路正常工作，也能使电路处于禁止工作状态；

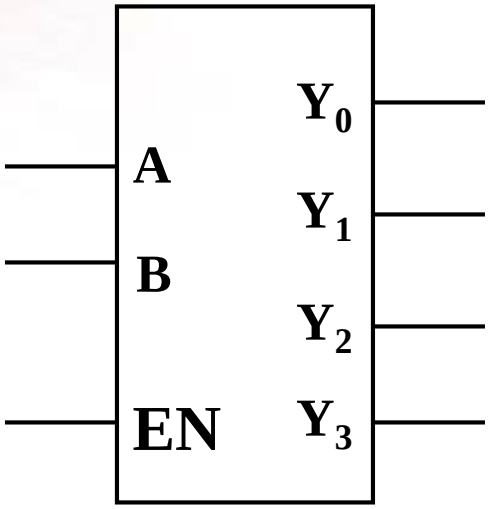
2）利用使能输入控制端，能实现译码器容量扩展。

EN为使能控制输入端，
EN=0，输出均为0；
EN=1，输出译码信号。

电路满足：$Y_i=m_i EN$

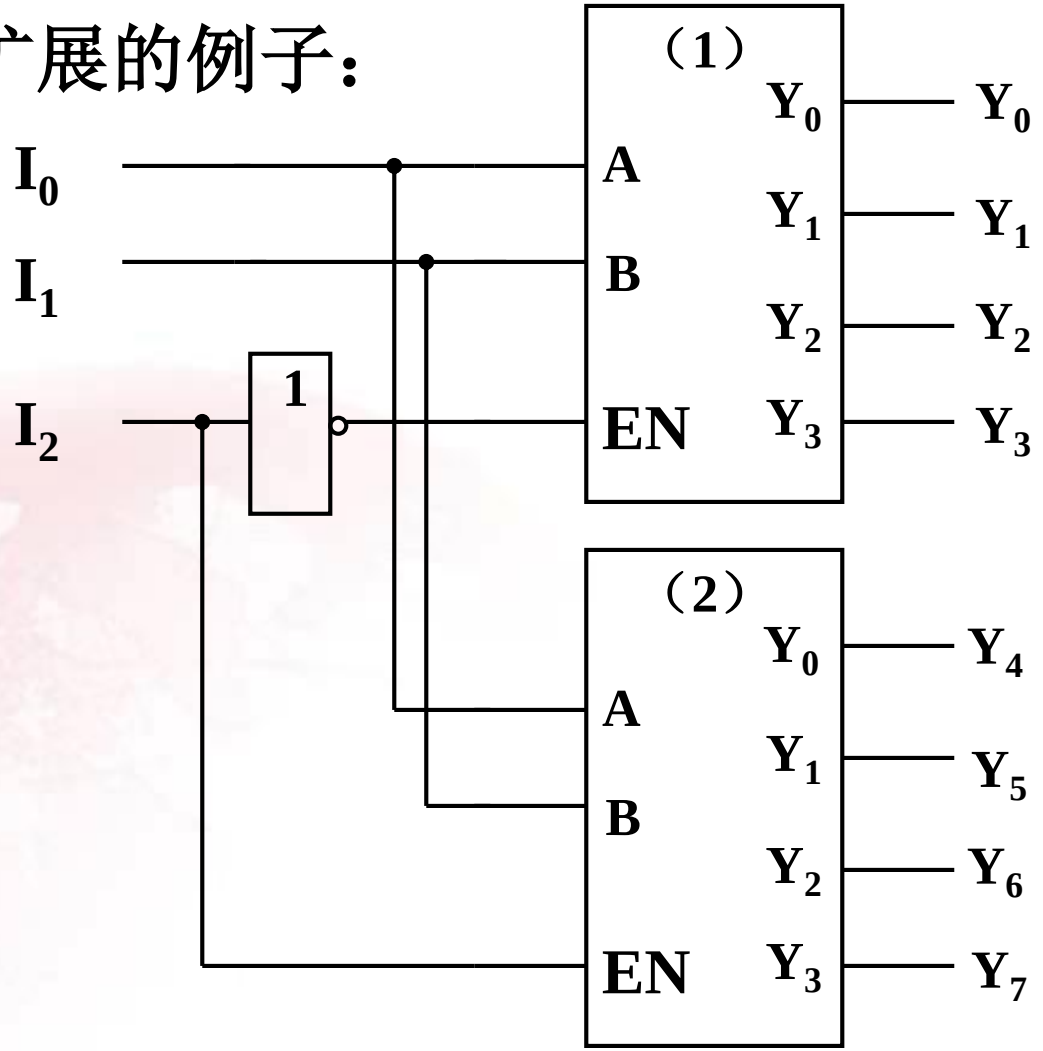


逻辑图



逻辑符号

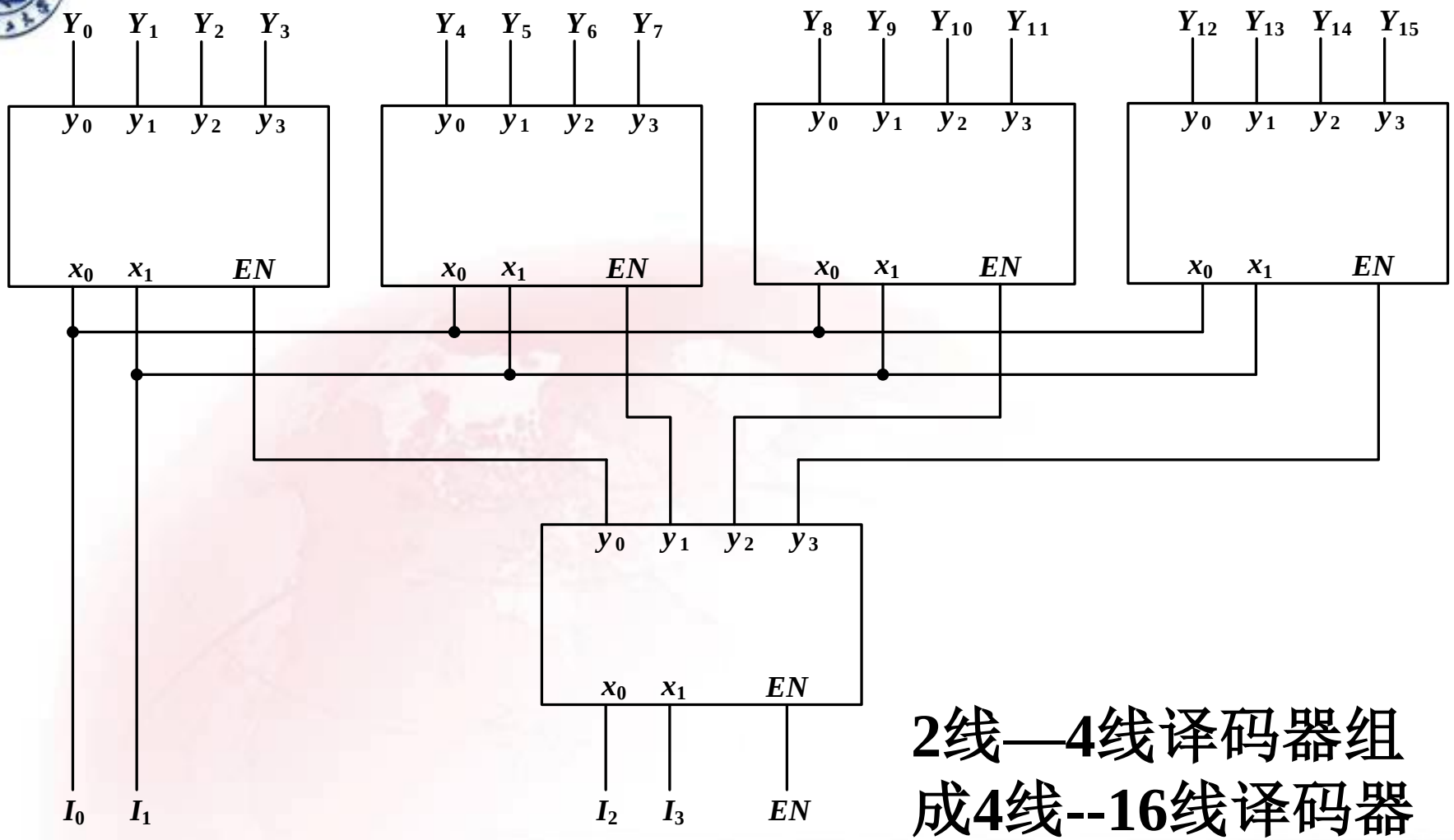
利用使能端实现扩展的例子：



当I_2=0时，(1)片工作，(2)片禁止.

当I_2=1时，(1)片禁止，(2)片工作.

由两片2线—4线译码器组成3线—8线译码器



2线—4线译码器组成4线--16线译码器

4.3.2 二—十进制译码器 (常称4线—10线译码器)

输入：BCD码. 输出：十个高、低电平.

真值表

A_3	A_2	A_1	A_0	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

输出低电平有效

伪码

4线—10线译码器逻辑表达式：

$$\overline{Y}_0=\overline{\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0}$$

$$\overline{Y}_1=\overline{\overline{A}_3\overline{A}_2\overline{A}_1A_0}$$

$$\overline{Y}_2=\overline{\overline{A}_3\overline{A}_2A_1\overline{A}_0}$$

$$\overline{Y}_3=\overline{\overline{A}_3\overline{A}_2A_1A_0}$$

$$\overline{Y}_4=\overline{\overline{A}_3A_2\overline{A}_1\overline{A}_0}$$

$$\overline{Y}_5=\overline{\overline{A}_3A_2\overline{A}_1A_0}$$

$$\overline{Y}_6=\overline{\overline{A}_3A_2A_1\overline{A}_0}$$

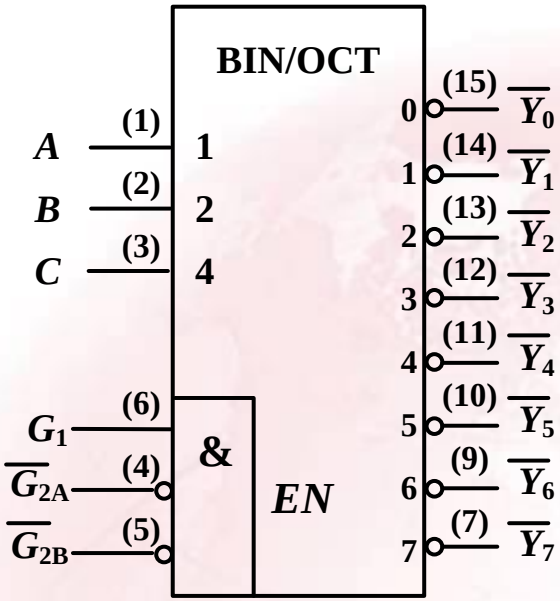
$$\overline{Y}_7=\overline{\overline{A}_3A_2A_1A_0}$$

$$\overline{Y}_8=\overline{A_3\overline{A}_2\overline{A}_1\overline{A}_0}$$

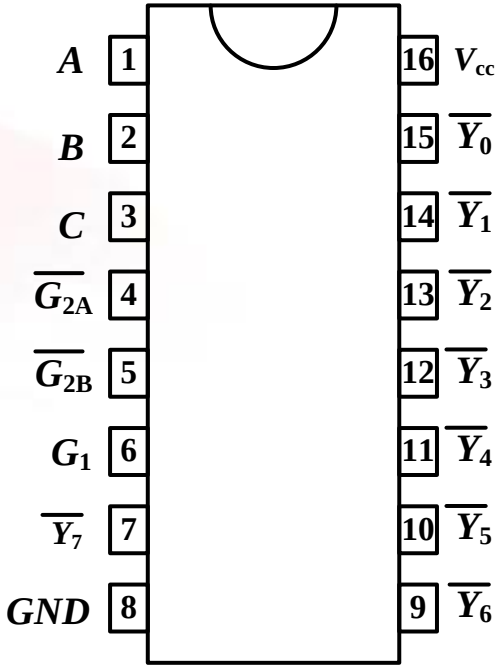
$$\overline{Y}_9=\overline{A_3\overline{A}_2\overline{A}_1A_0}$$

4.3.3 通用译码器集成电路

(1) 74138　带使能端3线—8线译码器



逻辑图



引脚图

74138功能表

G_1	$\overline{G_2}^*$	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	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
×	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	1	1	1	1	1	1	1	1

$\overline{G_2}^* = \overline{G}_{2A} + \overline{G}_{2B}$

74138特性：

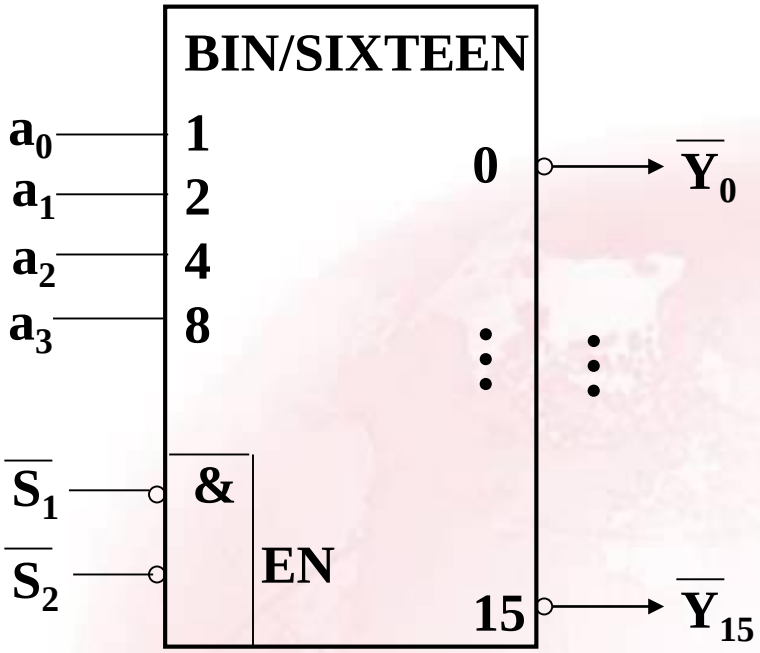
①74138的逻辑表达式为：

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

② 电路输出低电平有效;

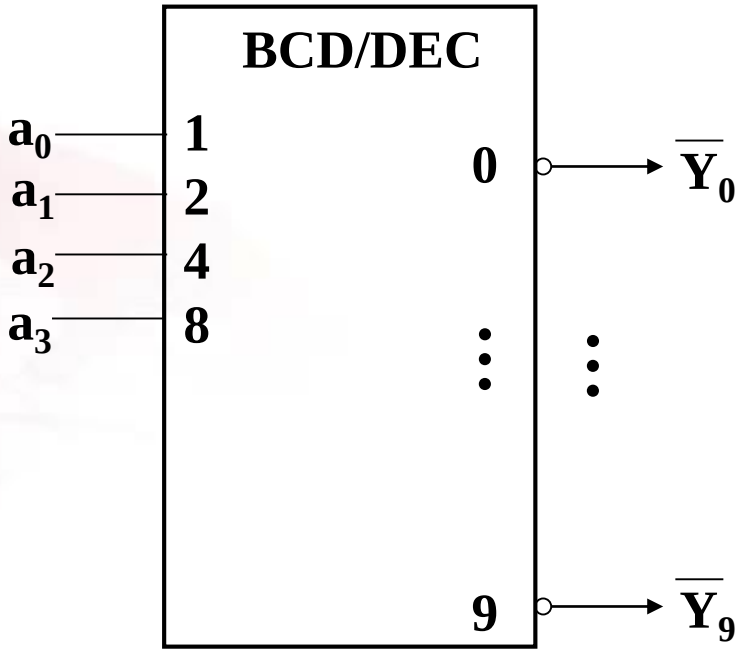
③ $G_1\overline{G_{2A}}\,\overline{G_{2B}}$ =100,电路工作;否则,电路禁止工作,电路输出均为1。

(2) 74154



4线—16线译码器

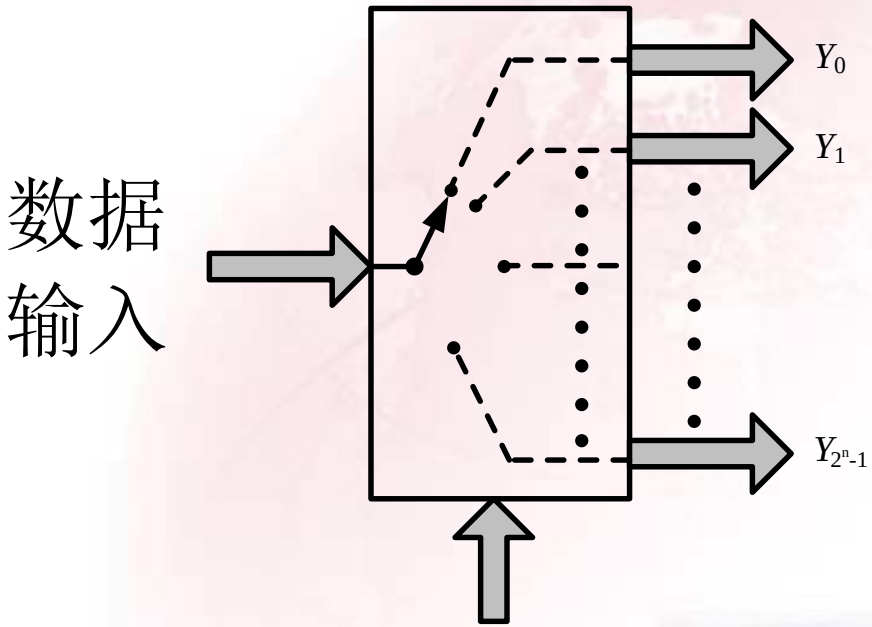
(3) 7442



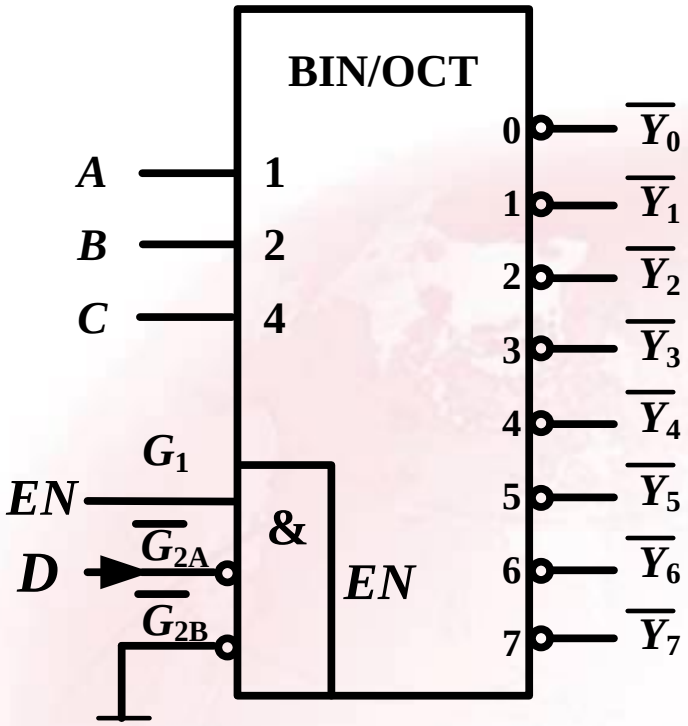
4线—10线译码器

4.3.4 数据分配器

数据分配是将一个数据源输入的数据根据需要送到不同的输出端上去，实现数据分配功能的逻辑电路称为数据分配器。分配器又叫多路复用器。



数据分配器一般用带使能控制端的二进制译码器实现。



74138输出表达式：

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

分配器输出表达式：

$$\overline{Y_i} = \overline{m_i EN\,\overline{D}}$$

用**74138**作为数据分配器

74138译码器作为数据分配器时的功能表

G_1	$\overline{G}_{2A}$	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	D	0	0	0	D	1	1	1	1	1	1	1
1	D	0	0	1	1	D	1	1	1	1	1	1
1	D	0	1	0	1	1	D	1	1	1	1	1
1	D	0	1	1	1	1	1	D	1	1	1	1
1	D	1	0	0	1	1	1	1	D	1	1	1
1	D	1	0	1	1	1	1	1	1	D	1	1
1	D	1	1	0	1	1	1	1	1	1	D	1
1	D	1	1	1	1	1	1	1	1	1	1	D
0	×	×	×	×	1	1	1	1	1	1	1	1

$\overline{G}_{2B}=0$

4.3.5显示译码器

显示器分类：

（1）半导体显示器，也称发光二极管显示器；

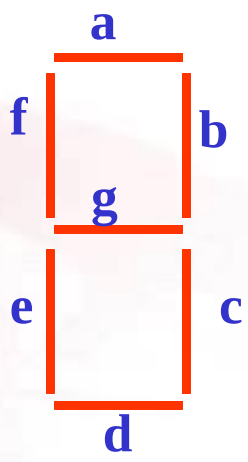
（2）荧光数字显示器，如荧光数码管、场致发光数字板等；

（3）液体数字显示器，如液晶显示器、电泳显示器等；

（4）气体放电显示器，如辉光数码管、等离子体显示板等。

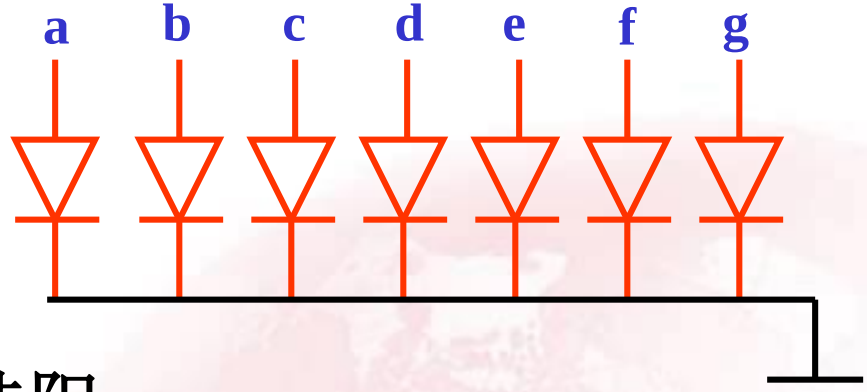
1. 半导体数码管（Light Emitting Diode简称LED）

七段数码管
显示器

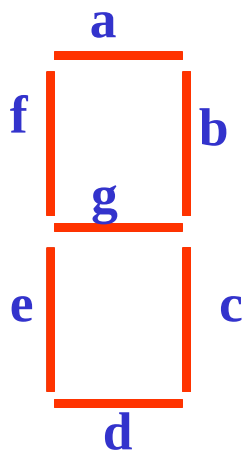


七段数码管的两种连接方法：

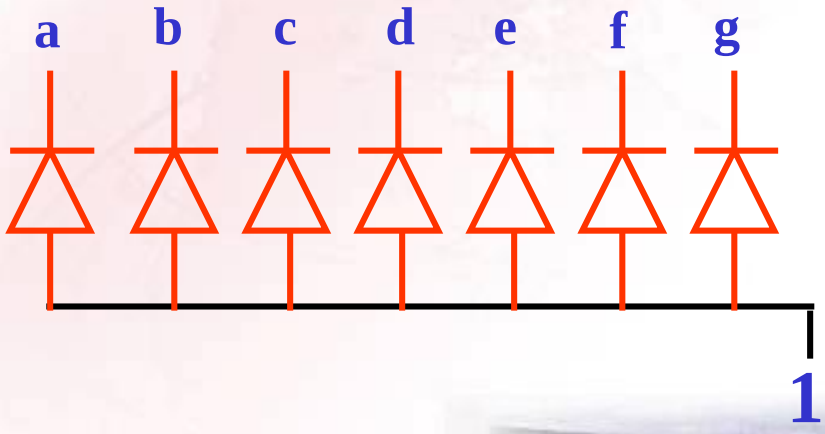
① 共阴



阳极加高电平字段亮。



② 共阳



阴极加低电平字段亮。

半导体数码管的工作电压比较低（1.5～ 3V），能直接用TTL或CMOS集成电路驱动。除电压比较低外，半导体数码管还具有体积小、寿命长、可靠性高等优点，而且响应时间短（一般不超过0.1μs）,亮度也比较高。LED显示器的缺点是工作电流大，每一段的工作电流在10mA左右。

2. 液晶显示器（Liquid Crystal Display,简称LCD）

液晶是一种既具有液体的流动性又具有光学特性的有机化学物。

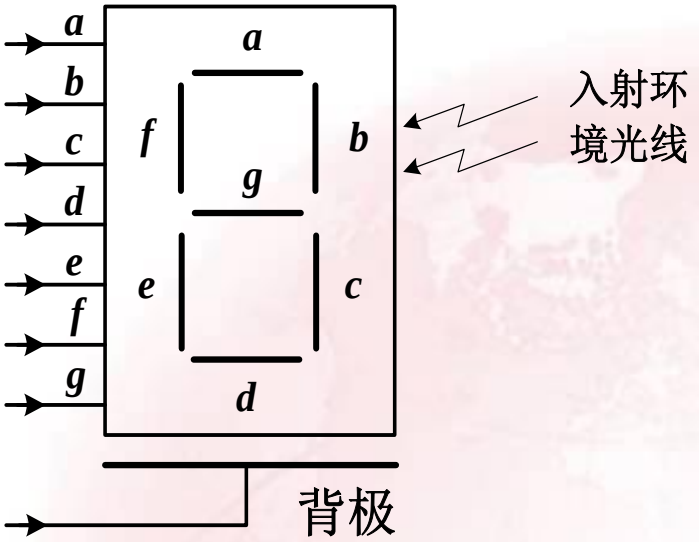
液晶显示器通过控制可见光的反射来达到显示目的。

液晶显示器分两类：反射式和背光式。

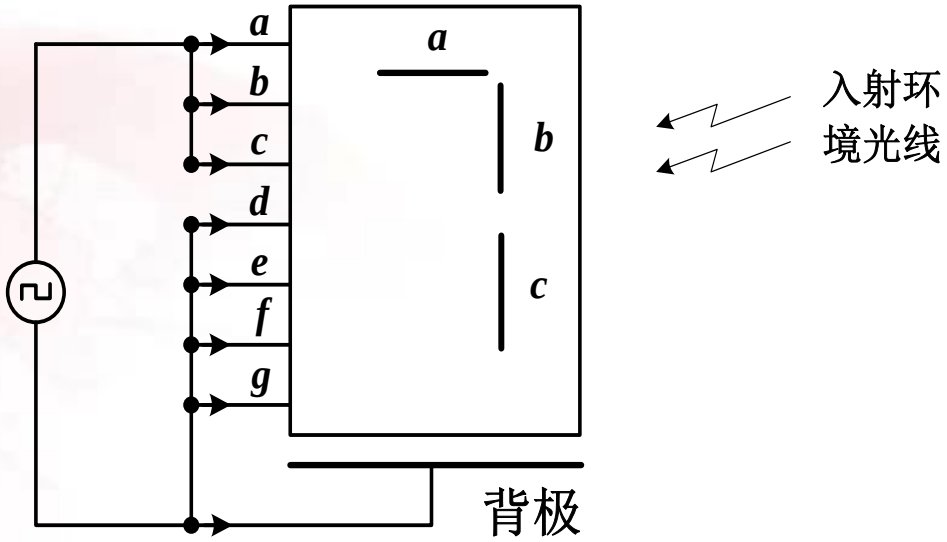
反射式液晶显示器使用的可见光是环境光线。

而背光式液晶显示器的可见光则由在显示器内特制的小光源提供。

LCD须用低频交流信号驱动，一般使用方波信号，工作频率约为25～60Hz，信号幅值可以很低，在1V以下仍能工作 。



七段液晶显示器示意图



加驱动电压，显示‘7’字形

液晶显示器的最大优点是功耗极低，每平方厘米的功耗的1μW以下。

液晶显示器工作电压低，功耗小的特点，使其在各种小型、便携式仪器、仪表中得到了广泛的应用。

当前，在电视机、计算机等设备中使用液晶显示器已越来越普及，并成为一种发展趋势。

3. 显示译码器设计

功能：将表示数字的BCD码转换成七段显示码。

输入：BCD码

输出：七段显示码

显示译码器设计步骤：

（以输入8421BCD码、输出驱动共阳显示器为例）

① 列真值表；

② 化简、写最简函数表达式；

③ 画电路图。

真　值　表

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

化简后表达式:

$a=A\overline{B}\overline{C}\overline{D}+\overline{A}\overline{B}C$

$b=A\overline{B}C+\overline{A}BC$

$c=\overline{A}B\overline{C}$

$d=\overline{A}\overline{B}C+ABC+A\overline{B}\overline{C}\overline{D}$

$e=A+\overline{A}\overline{B}C$

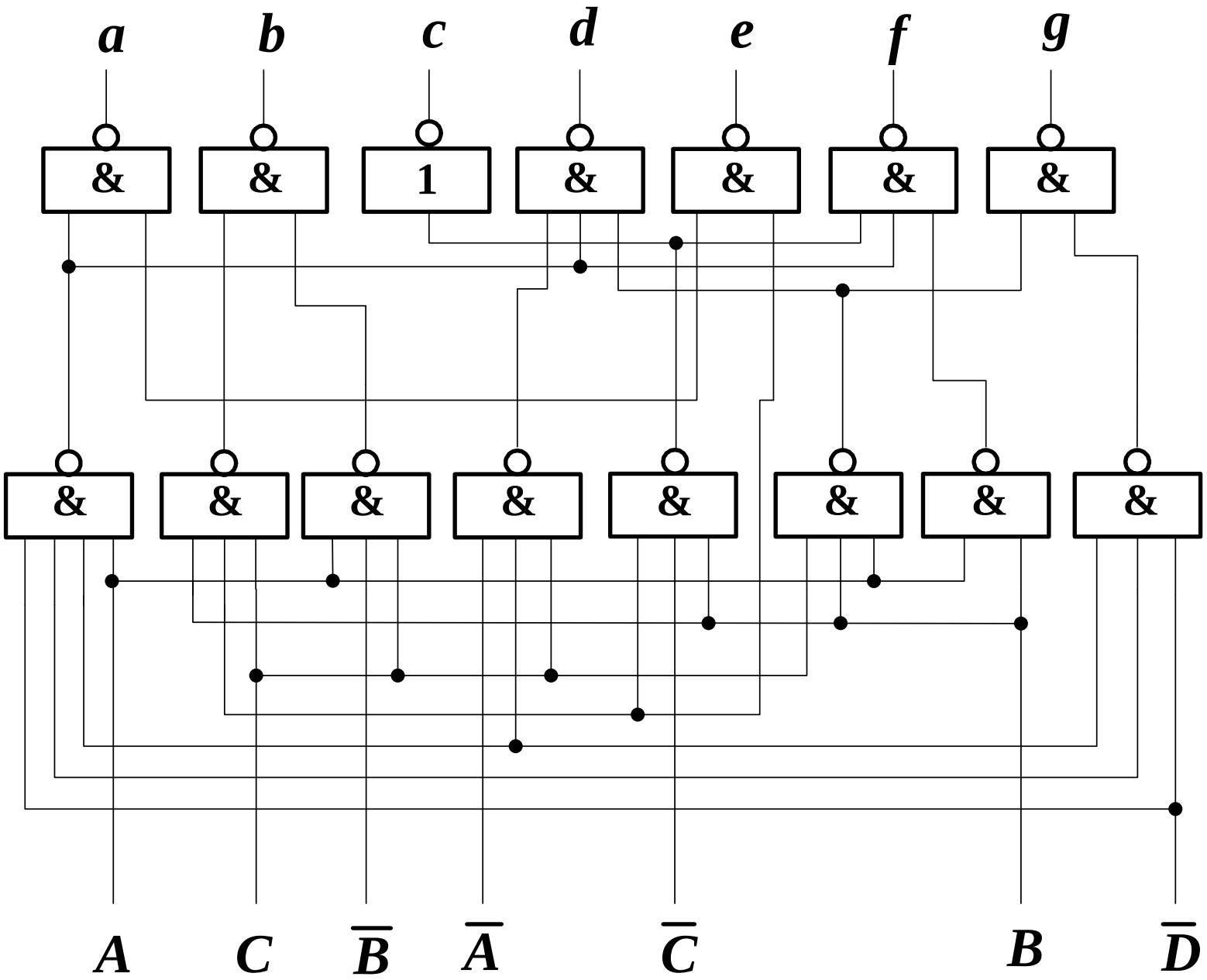
$f=AB+A\overline{B}C\overline{D}+\overline{A}B\overline{C}$

$g=ABC+\overline{B}\overline{C}\overline{D}$

化简说明:

① 利用了无关项;

② 考虑了多输出逻辑函数化简中的公共项.



七段显示译码器逻辑图

思考题：

根据上面设计，判断当输入DCBA为1010时，LED显示什么字形？

4. 通用七段显示译码器集成电路

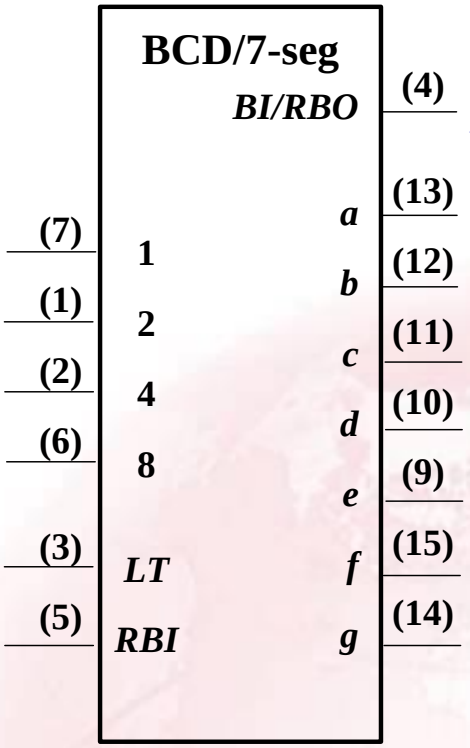
常用的七段显示译码器集成电路有7446、7447、7448、7449和4511等。下面重点介绍七段显示译码器7448。

七段显示译码器7448输出高电平有效，用以驱动共阴极显示器。

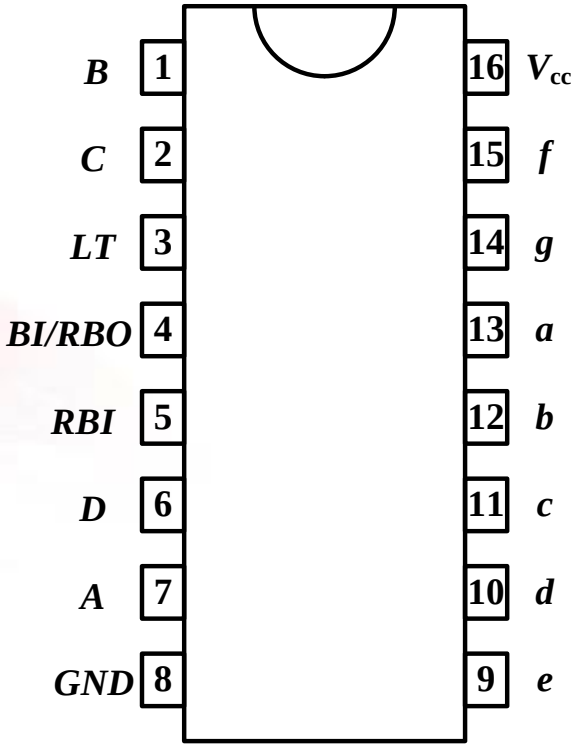
BCD
码输入

试灯输入

动态灭零输入



逻辑图

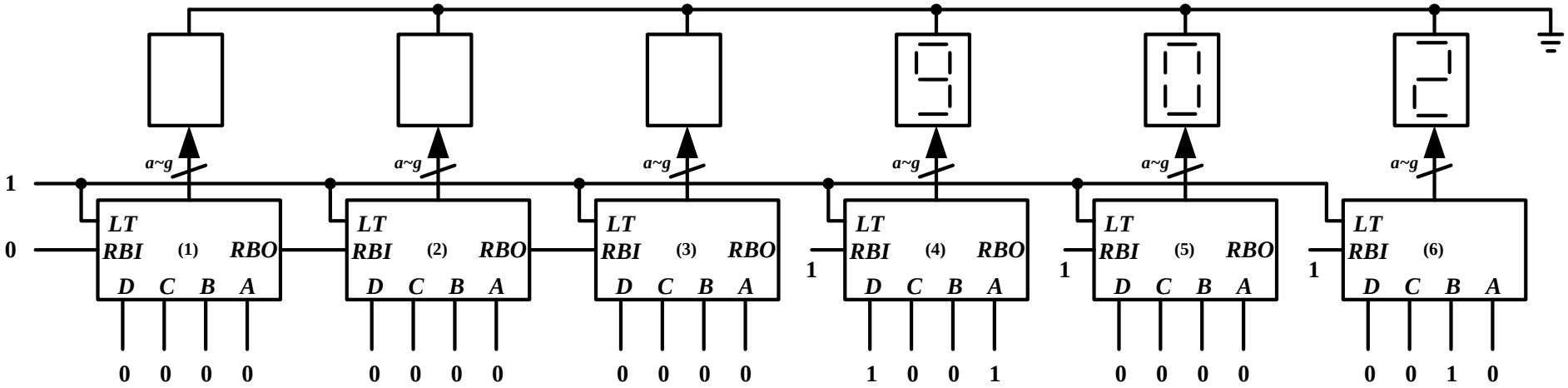


引脚图

七段显示译码器7448功能表

十进制或功能	输入						BI/RBO	输出							字形
	LT	RBI	D	C	B	A		a	b	c	d	e	f	g	
0	1	1	0	0	0	0	1	1	1	1	1	1	1	0	0
1	1	×	0	0	0	1	1	0	1	1	0	0	0	0	1
2	1	×	0	0	1	0	1	1	1	0	1	1	0	1	2
3	1	×	0	0	1	1	1	1	1	1	1	0	0	1	3
4	1	×	0	1	0	0	1	0	1	1	0	0	1	1	4
5	1	×	0	1	0	1	1	1	0	1	1	0	1	1	5
6	1	×	0	1	1	0	1	0	0	1	1	1	1	1	6
7	1	×	0	1	1	1	1	1	1	1	0	0	0	0	7
8	1	×	1	0	0	0	1	1	1	1	1	1	1	1	8
9	1	×	1	0	0	1	1	1	1	1	1	0	1	1	9
10	1	×	1	0	1	0	1	0	0	0	1	1	0	1	c
11	1	×	1	0	1	1	1	0	0	1	1	0	0	1	ɔ
12	1	×	1	1	0	0	1	0	1	0	0	0	1	1	∪
13	1	×	1	1	0	1	1	1	0	0	1	0	1	1	⊂
14	1	×	1	1	1	0	1	0	0	0	1	1	1	1	⊢
15	1	×	1	1	1	1	1	0	0	0	0	0	0	0	
消隐	×	×	×	×	×	×	0	0	0	0	0	0	0	0	
脉冲消隐	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
灯测试	0	×	×	×	×	×	1	1	1	1	1	1	1	1	8

7448实现多位显示

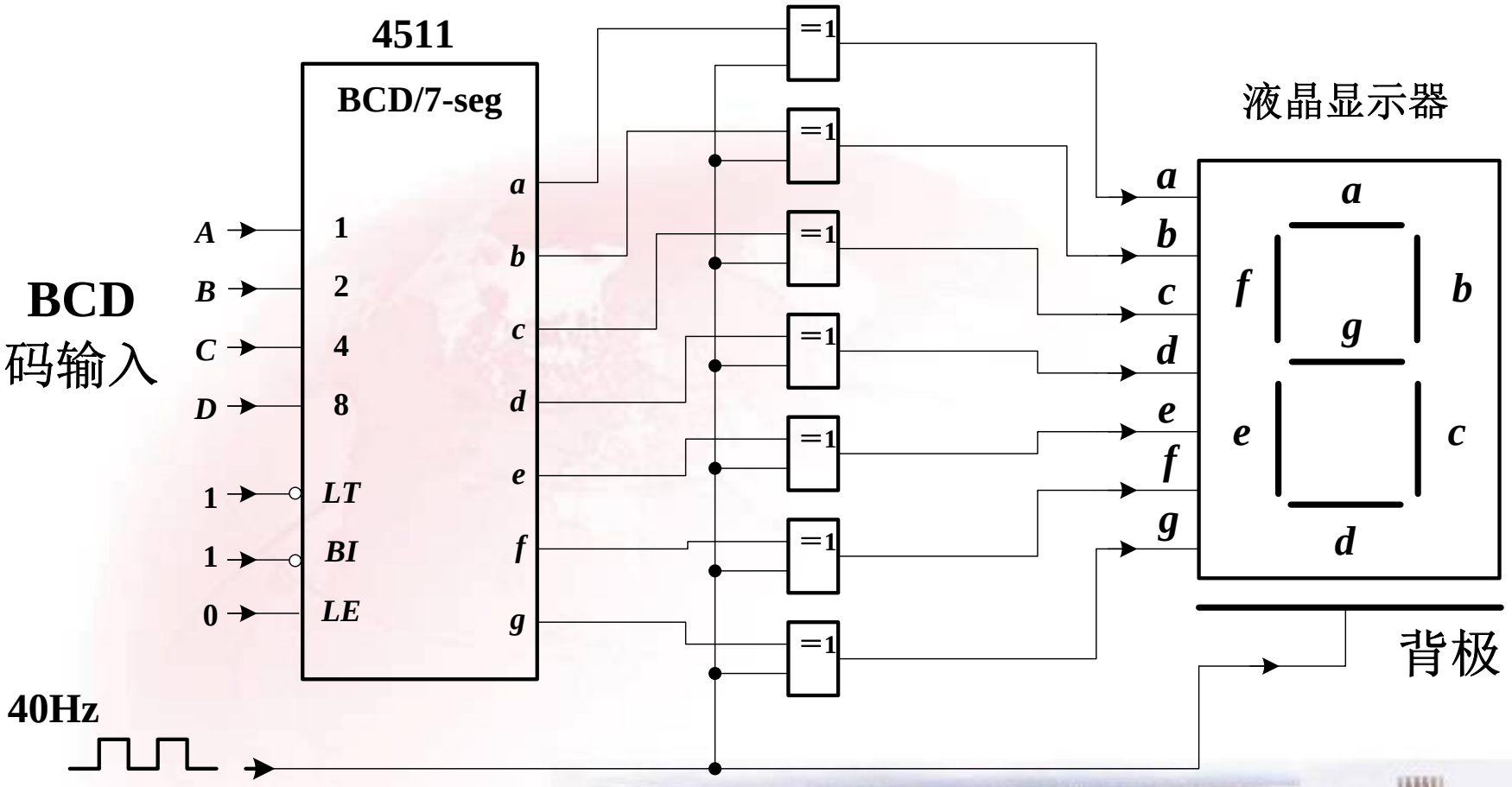


由于第1片的*RBI*为0，而*DCBA*=0000, 所以满足灭零条件，*RBO*=0。第2、3片也满足灭零条件。

第4、5、6片驱动正常显示。

思考题：如第1片输入*DCBA*不等于0000，2、3两片灭零条件吗？

74HC4511显示译码器驱动液晶数码管的一个例子



4.3.6 译码器应用举例

1. 译码器实现组合逻辑函数

原理：二进制译码器能产生输入信号的全部最小项,而所有组合逻辑函数均可写成最小项之和的形式.

例　试用3线– 8线译码器和逻辑门实现下列函数

$$F(Q,X,P)=\Sigma m\ (0，1，4，6，7）$$
$$=\Pi M（2，3，5）$$

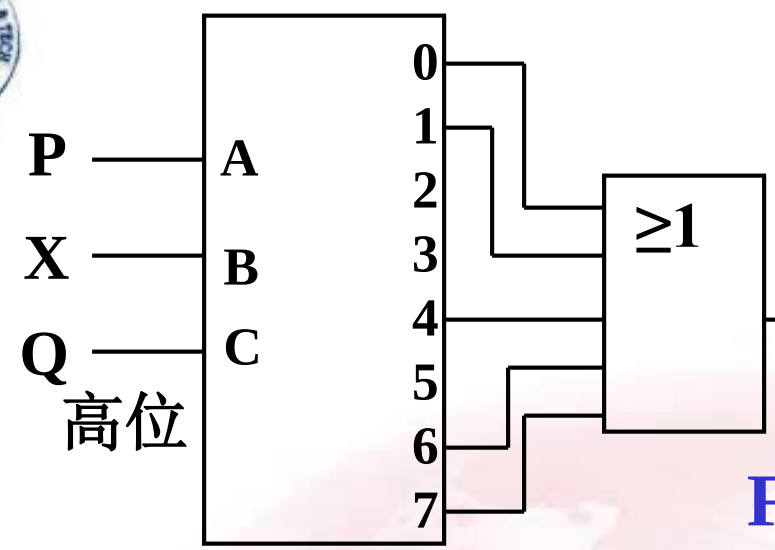
解题的几种方法：

① 利用高电平输出有效的译码器和或门。

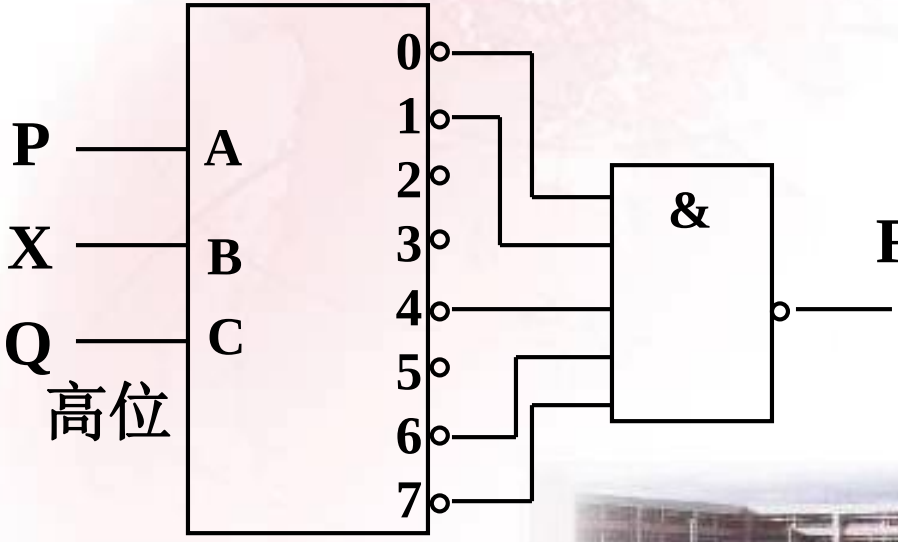
$$F(Q,X,P)=m_0+m_1+m_4+m_6+m_7$$

② 利用低电平输出有效的译码器和与非门。

$$F(Q,X,P)=\overline{\overline{m_0}\,\overline{m_1}\,\overline{m_4}\,\overline{m_6}\,\overline{m_7}}$$



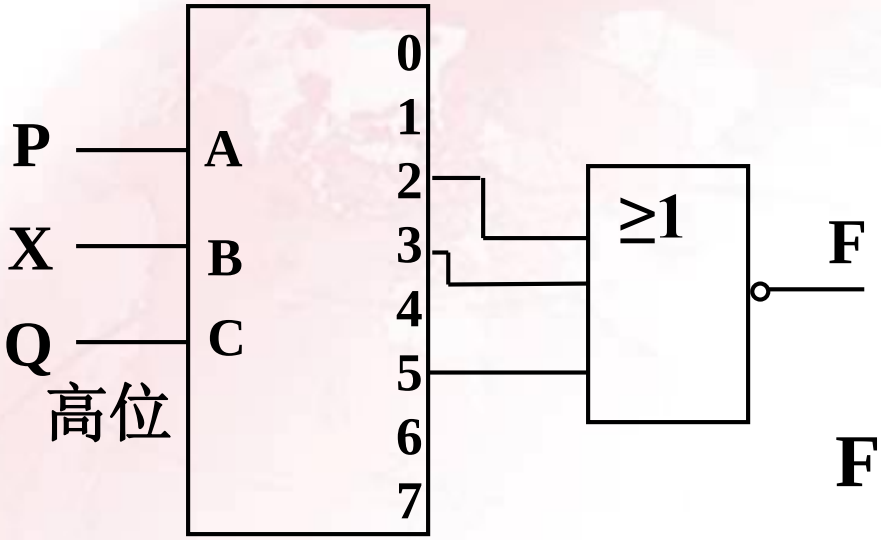
$$F(Q,X,P)=m_0+m_1+m_4+m_6+m_7$$



$$F(Q,X,P)=\overline{\overline{m_0}\,\overline{m_1}\,\overline{m_4}\,\overline{m_6}\,\overline{m_7}}$$

③ 利用高电平输出有效的译码器和或非门。

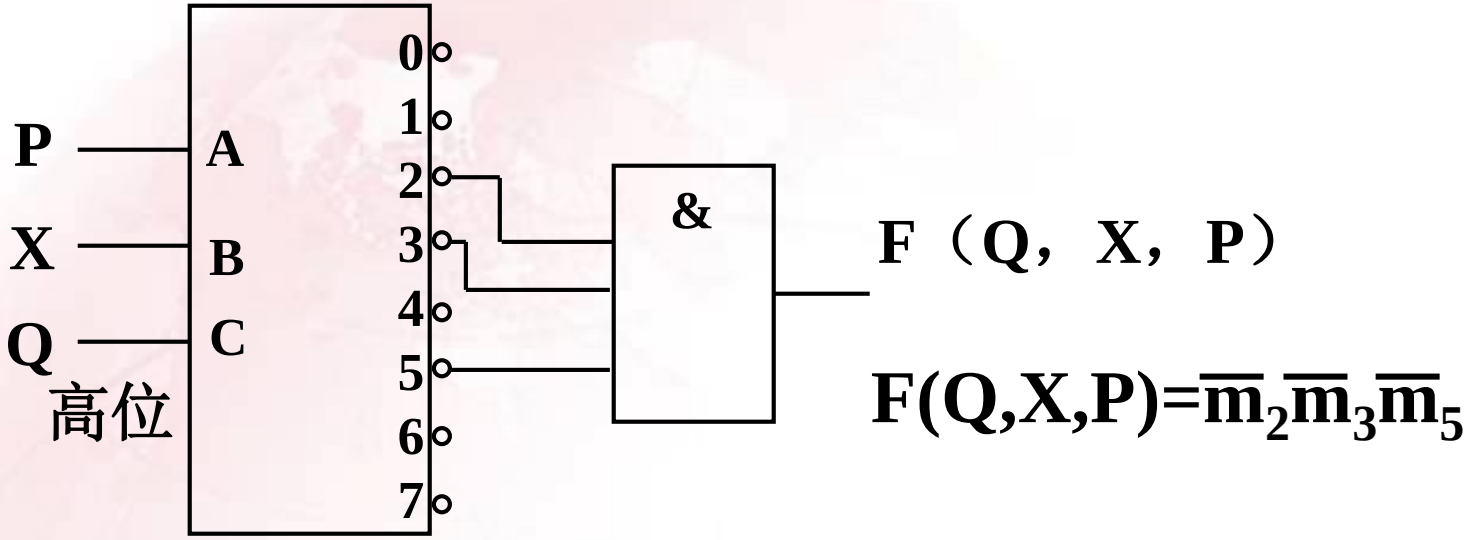
$$F(Q,X,P)=\overline{m_2+m_3+m_5}$$



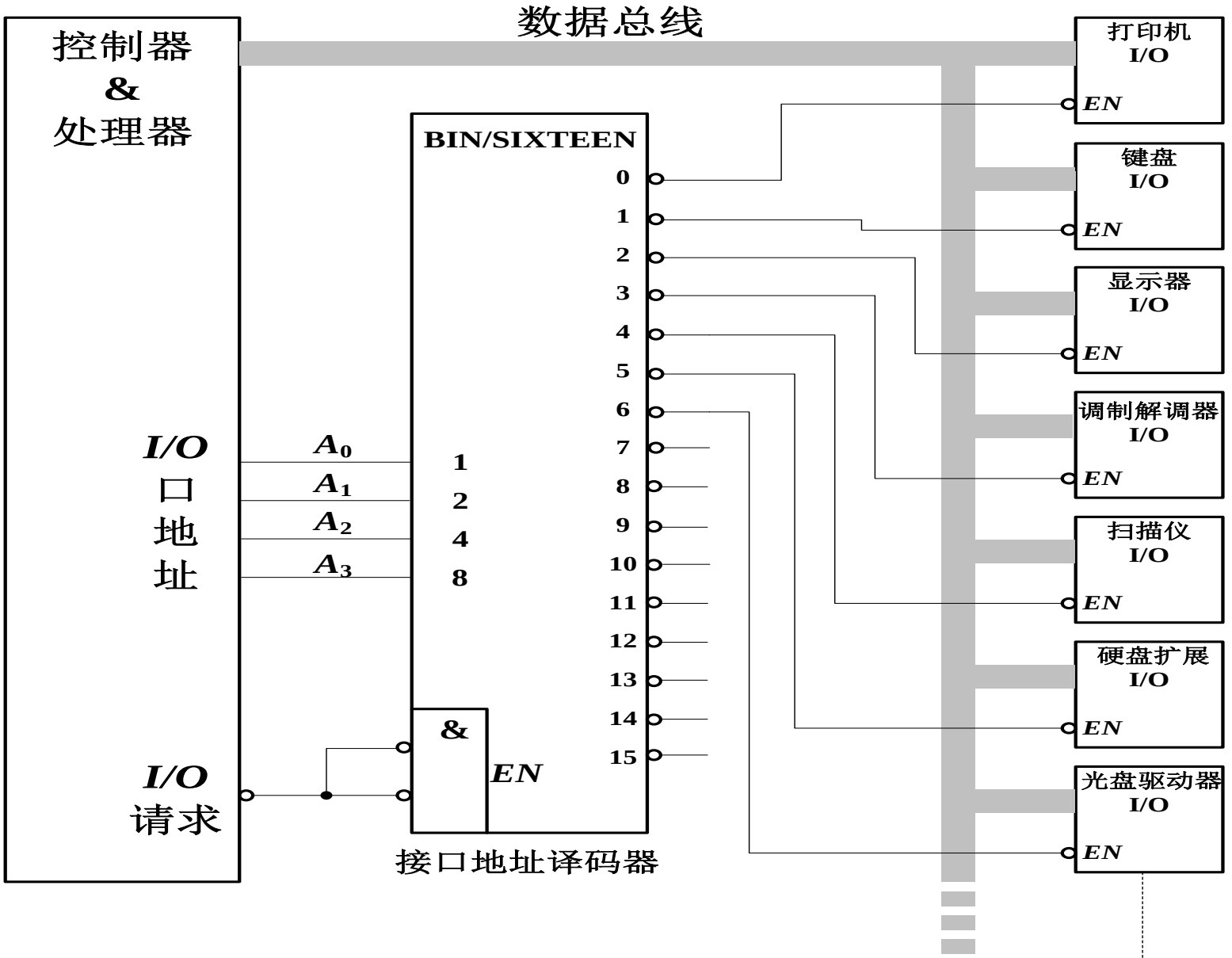
$$F(Q,X,P)=\overline{m_2+m_3+m_5}$$

③ 利用低电平输出有效的译码器和与门。

$F(Q,X,P)=\overline{m}_2\overline{m}_3\overline{m}_5$



2. 计算机输入/输出接口地址译码电路



4.3.7 译码器的VHDL描述

3线－8线译码器74138的VHDL描述。

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY decode_3to8 IS
PORT(a,b,c,G1,G2A,G2B:  IN STD_LOGIC;
            y:  OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END decode_3to8;
ARCHITECTURE rtl OF decode_3to8 IS
SIGNAL indata: STD_LOGIC_VECTOR(2 DOWNTO 0);
BEGIN
```

```
 indata<=c&b&a;
PROCESS(indata,G1,G2A,G2B)
  BEGIN
    IF(G1='1' AND G2A='0' AND G2B='0') THEN
      CASE indata IS
        WHEN "000"=>y<="11111110";
        WHEN "001"=>y<="11111101";
        WHEN "010"=>y<="11111011";
        WHEN "011"=>y<="11110111";
        WHEN "100"=>y<="11101111";
        WHEN "101"=>y<="11011111";
        WHEN "110"=>y<="10111111";
        WHEN "111"=>y<="01111111";
        WHEN OTHERS=>y<="XXXXXXXX";
      END CASE;
```

```
  ELSE
      y<="11111111";
  END IF;
 END PROCESS;
END rtl;
```

七段显示译码器的VHDL描述

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY  bcd_7seg  IS
PORT ( bcd_led :  IN STD_LOGIC_VECTOR(3 DOWNTO 0);
              ledseg :  OUT STD_LOGIC_VECTOR(6 DOWNTO 0));
END bcd_7seg;


ARCHITECTURE behavior OF bcd_7seg IS
BEGIN
  WITH bcd_led SELECT
```

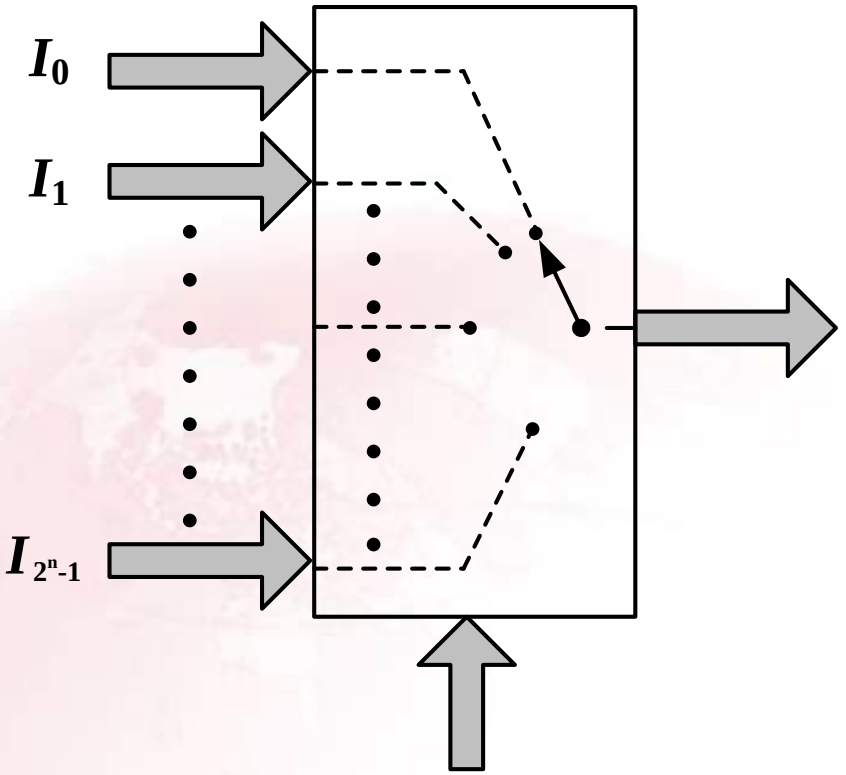
```
ledseg<= “0111111” WHEN “0000”,--0
         “0000110” WHEN “0001”,--1
         “1011011” WHEN “0010”,--2
         “1001111” WHEN “0011”,--3
         “1100110” WHEN “0100”,--4
         “1101101” WHEN “0101”,--5
         “1111101” WHEN “0110”,--6
         “0100111” WHEN “0111”,--7
         “1111111” WHEN “1000”,--8
         “1101111” WHEN “1001”,--9
         “1000000” WHEN “1110”,--minus
         “0000000” WHEN OTHERS;
END behavior;
```

4.4 数据选择器

功能: 从多路输入数据中选择其中的一路送至输出端.

数据选择器简称MUX,数据选择器的数据输入端数称为通道数.

数据选择器功能示意图:



数据输入

数据输出

选择信号
（地址码）输入

4.4.1 数据选择器的电路结构

以四选一数据选择器为例讨论

功能表

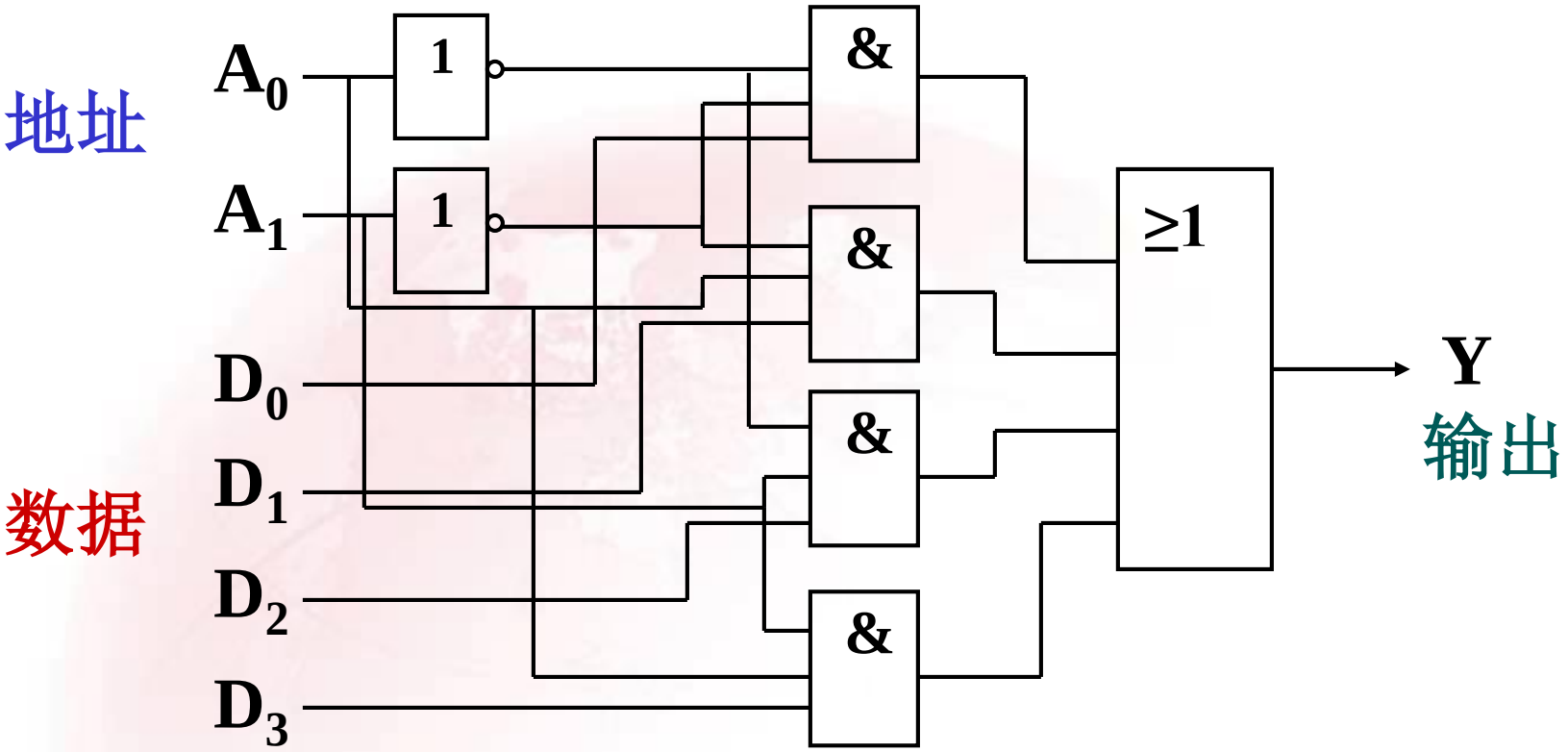
A_1	A_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

输出函数表达式:

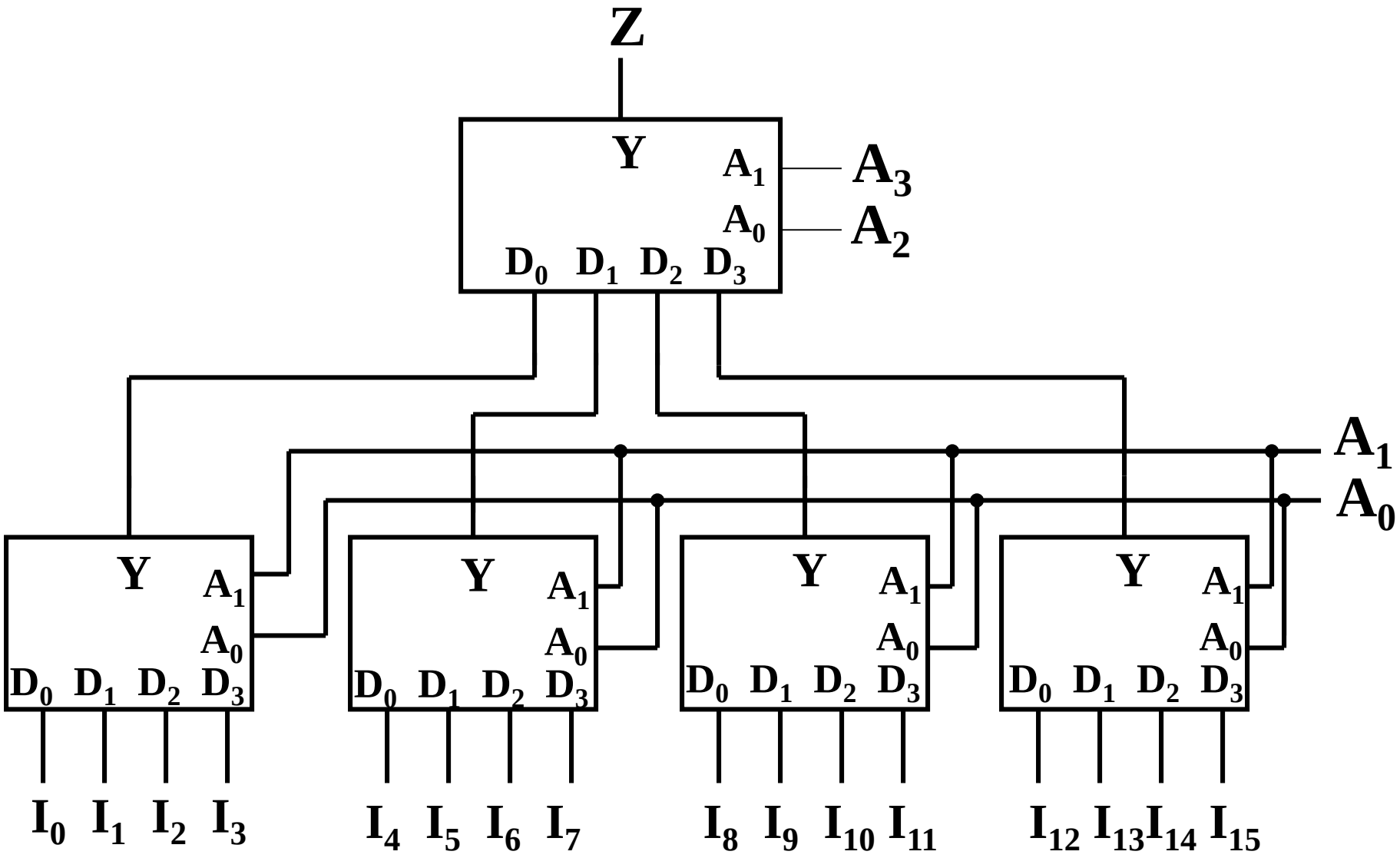
$$Y= (\overline{A}_1\overline{A}_0)D_0+ (\overline{A}_1A_0)D_1 + (A_1\overline{A}_0)D_2+ (A_1A_0)D_3$$

$$Y=\sum_{i=0}^{3} m_iD_i$$

电路图：



数据选择器通道扩展：由四选一数据选择器组成十六选一数据选择器的例子



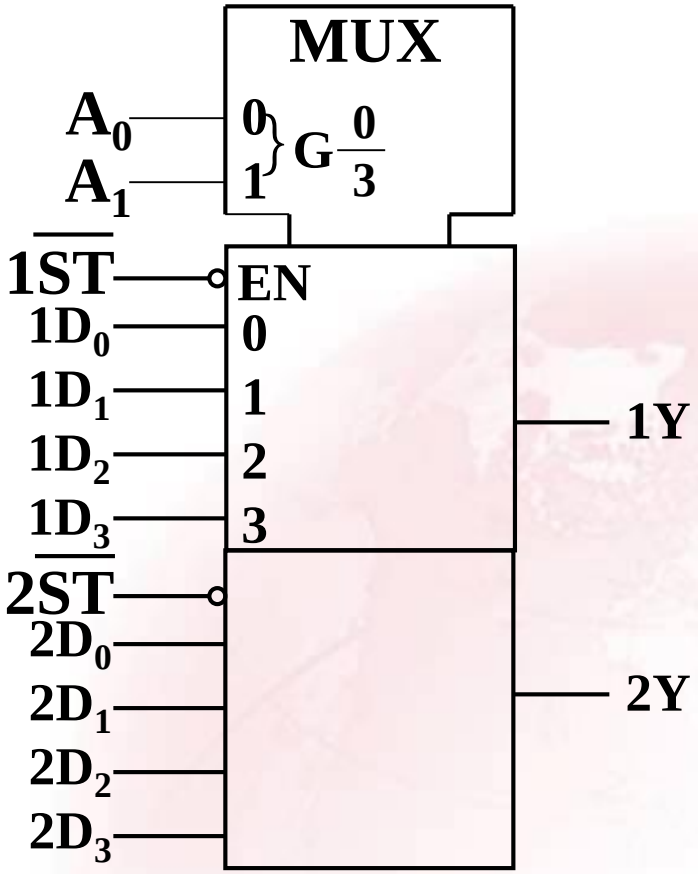
4.4.2 通用数据选择器集成电路

常用MUX集成电路

输入数	TTL	CMOS(数字)	CMOS(模拟)	ECL
16	74150	4515	4067	
2×8	74451		4096	
8	74151	4512	4051	10164
4×4	74453			
2×4	74153	4539	4052	10174
8×2	74604			
4×2	74157	4519		10159

数据选择器的逻辑符号及输入选通端：

以双四选一MUX74153和MUX74HC4539说明之。

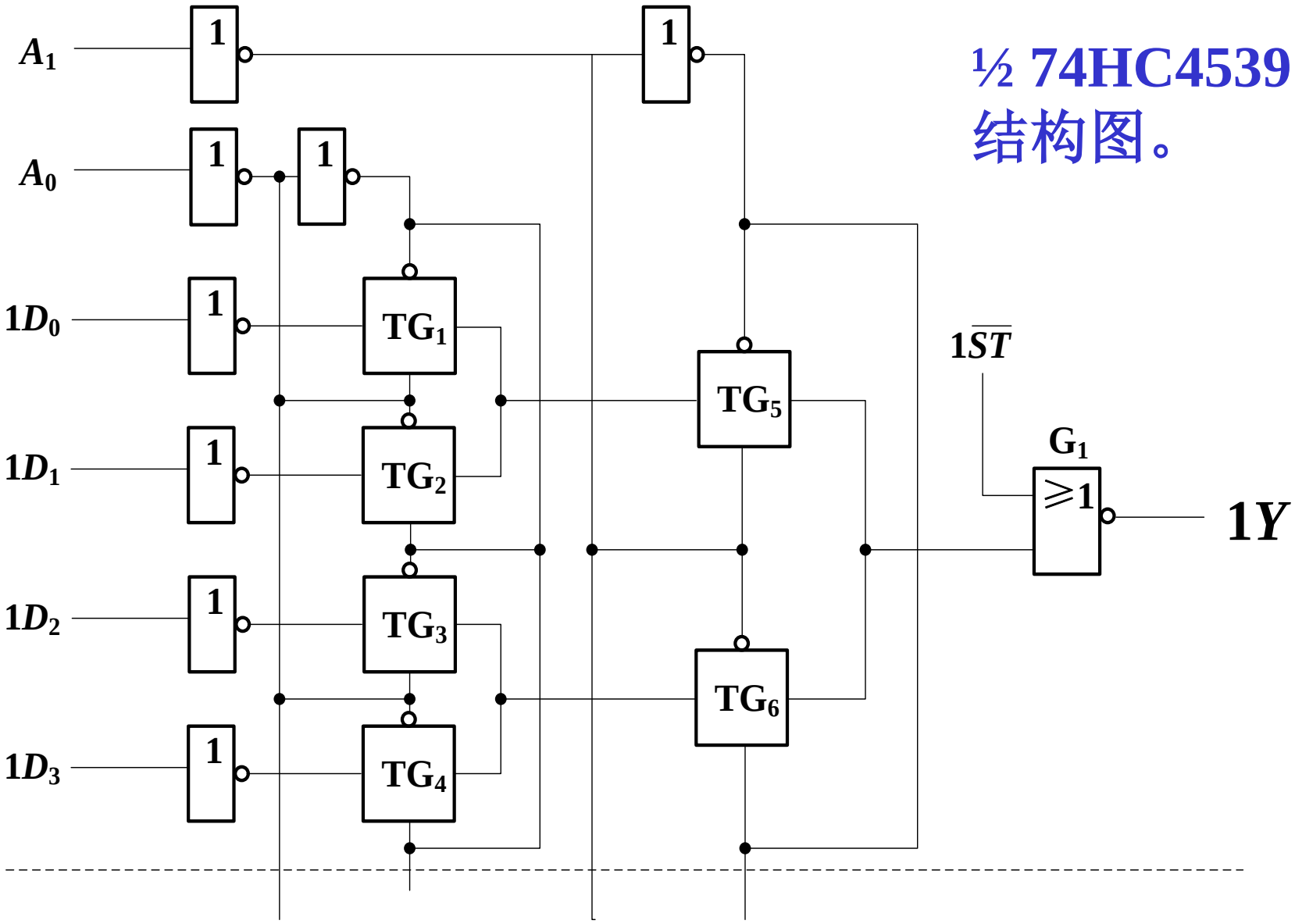


74153

$$Y=(\overline{A}_1\overline{A}_0)D_0+(\overline{A}_1A_0)D_1+(A_1\overline{A}_0)D_2+(A_1A_0)D_3)ST$$

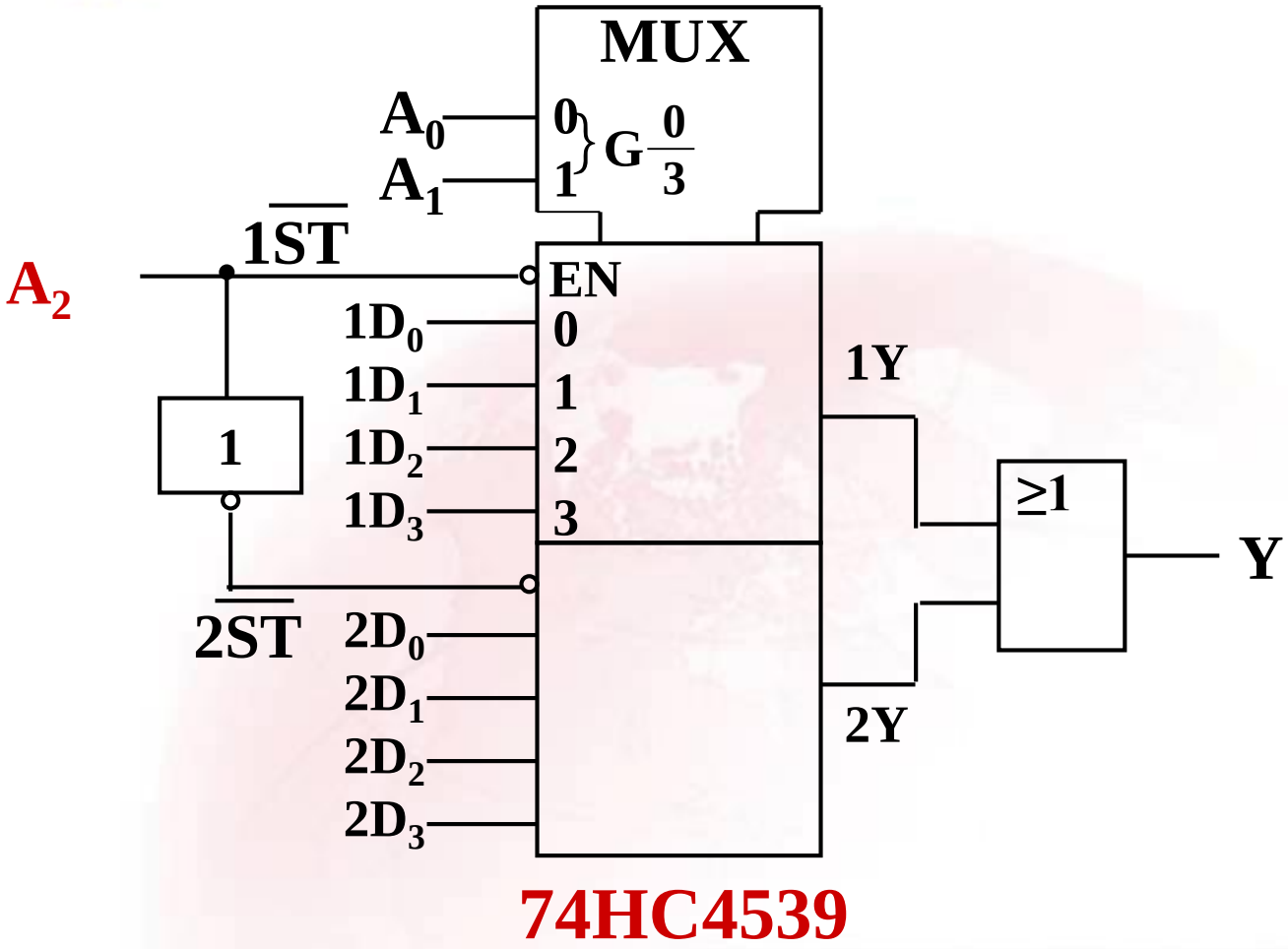
内部结构由与、或、非等门组成。

74HC4539的功能和逻辑符号和74153相同，但芯片内部由CMOS传输门组成。



½ 74HC4539结构图。

利用选通控制端实现通道扩展的例子：



74HC4539

A_2=0 时,由
A_1A_0选择1D_i

A_2=1时,由
A_1A_0选择2D_i

4.4.3 数据选择器应用举例

1. 用数据选择器实现组合逻辑函数

基本思想:

由数据选择器的一般表达式 $Y=\Sigma m_i D_i$

可知,利用地址变量产生所有最小项,通过数据输入信号D_i的不同取值,来选取组成逻辑函数的所需最小项.

例 试用八选一数据选择器74151实现逻辑函数

$F(A, B, C)=\Sigma m(0, 2, 3, 5)$

解：　待实现的函数为：$F(A, B, C)=\Sigma m(0, 2, 3, 5)$

$=\overline{A}\overline{B}\overline{C}+\overline{A}B\overline{C}+\overline{A}BC+A\overline{B}C$

74151的输出表达式为：

$$Y=(\overline{A}_2\overline{A}_1\overline{A}_0D_0+\overline{A}_2\overline{A}_1A_0D_1+\overline{A}_2A_1\overline{A}_0D_2+\overline{A}_2A_1A_0D_3 + A_2\overline{A}_1\overline{A}_0D_4+A_2\overline{A}_1A_0D_5+A_2A_1\overline{A}_0D_6+A_2A_1A_0D_7)ST$$

比较两式：

令：　$\overline{ST}=0$　　$A_2=A$; $A_1=B$; $A_0=C$

$D_0=D_2=D_3=D_5=1$　　$D_1=D_4=D_6=D_7=0$

$Y=F$

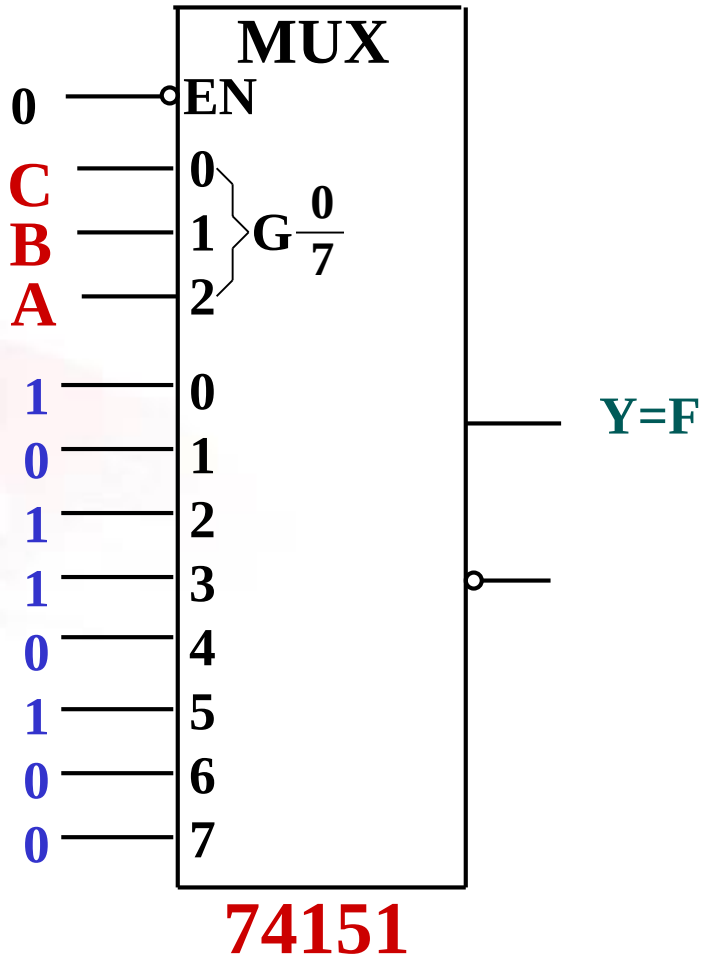
$\overline{ST}=0$

$A_2=A$; $A_1=B$; $A_0=C$

$D_0=D_2=D_3=D_5=1$

$D_1=D_4=D_6=D_7=0$

$Y=F$



注意：①用MUX实现逻辑函数时，MUX必须被选通，即$\overline{ST}=0$

②变量和地址端之间的连接必须正确。

例：试用四选一MUX实现逻辑函数

$F=\overline{A}\,\overline{B}\,\overline{C}+\overline{A}\,\overline{B}C+A\overline{B}\,\overline{C}+ABC$

解：当MUX被选通时，其输出逻辑表达式为：

$Y=(\overline{A}_1\overline{A}_0)D_0+(\overline{A}_1A_0)D_1+(A_1\overline{A}_0)D_2+(A_1A_0)D_3$

将函数F写成：$F=\overline{A}\,\overline{B}\cdot 1+\overline{A}B\cdot 0+A\overline{B}\cdot\overline{C}+AB\cdot C$

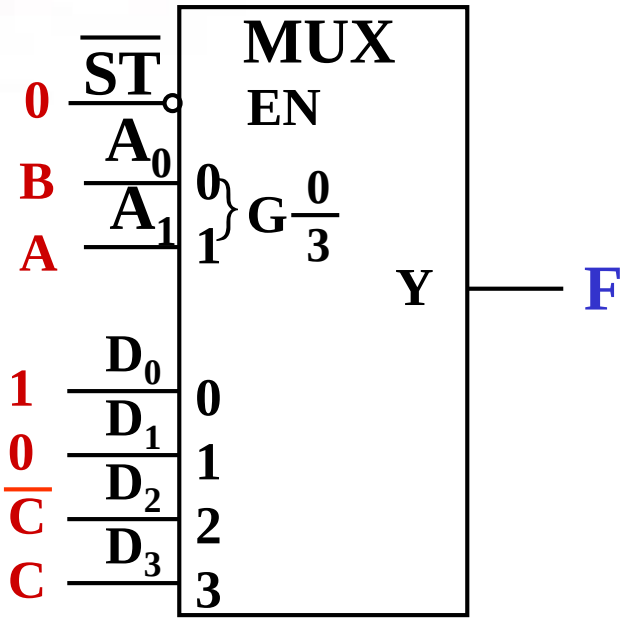
比较两式，令

$A_1=A$；$A_0=B$；

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

则 $Y=F$

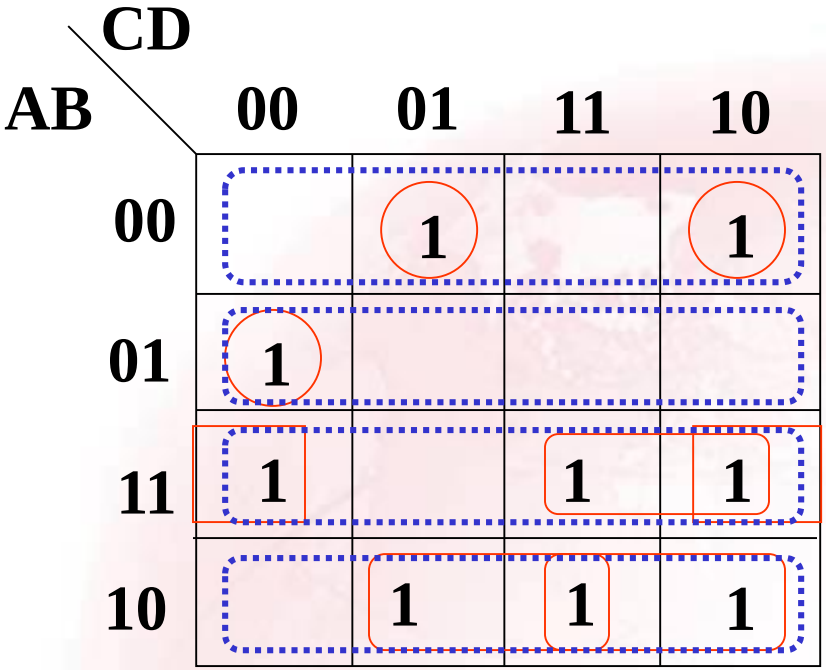
注：该题的解法不唯一。



例：用四选一数据选择器实现逻辑函数：

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

解：　令数据选择器的地址$A_1A_0=AB$



$\overline{A}\,\overline{B}(\overline{C}D+C\overline{D})=\overline{A}_1\overline{A}_0D_0$

$\overline{A}B(\overline{C}\,\overline{D})=\overline{A}_1A_0D_1$

$AB(C+\overline{D})=A_1A_0D_3$

$A\overline{B}(C+D)=A_1\overline{A}_0D_2$

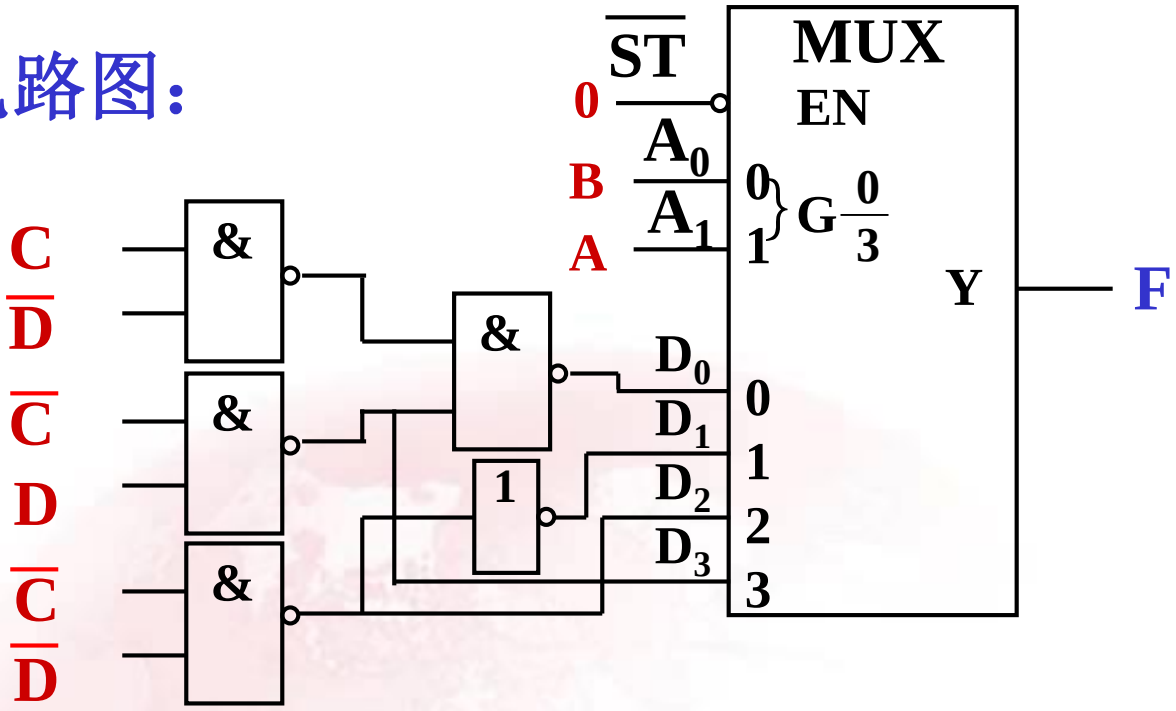
$D_0=\overline{C}D+C\overline{D}=\overline{\overline{C}D}\cdot\overline{C\overline{D}}$

$D_1=\overline{C}\,\overline{D}=\overline{\overline{C}\,\overline{D}}$

$D_2=C+D=\overline{\overline{C}\,\overline{D}}$

$D_3=C+\overline{D}=\overline{\overline{C}D}$

电路图：



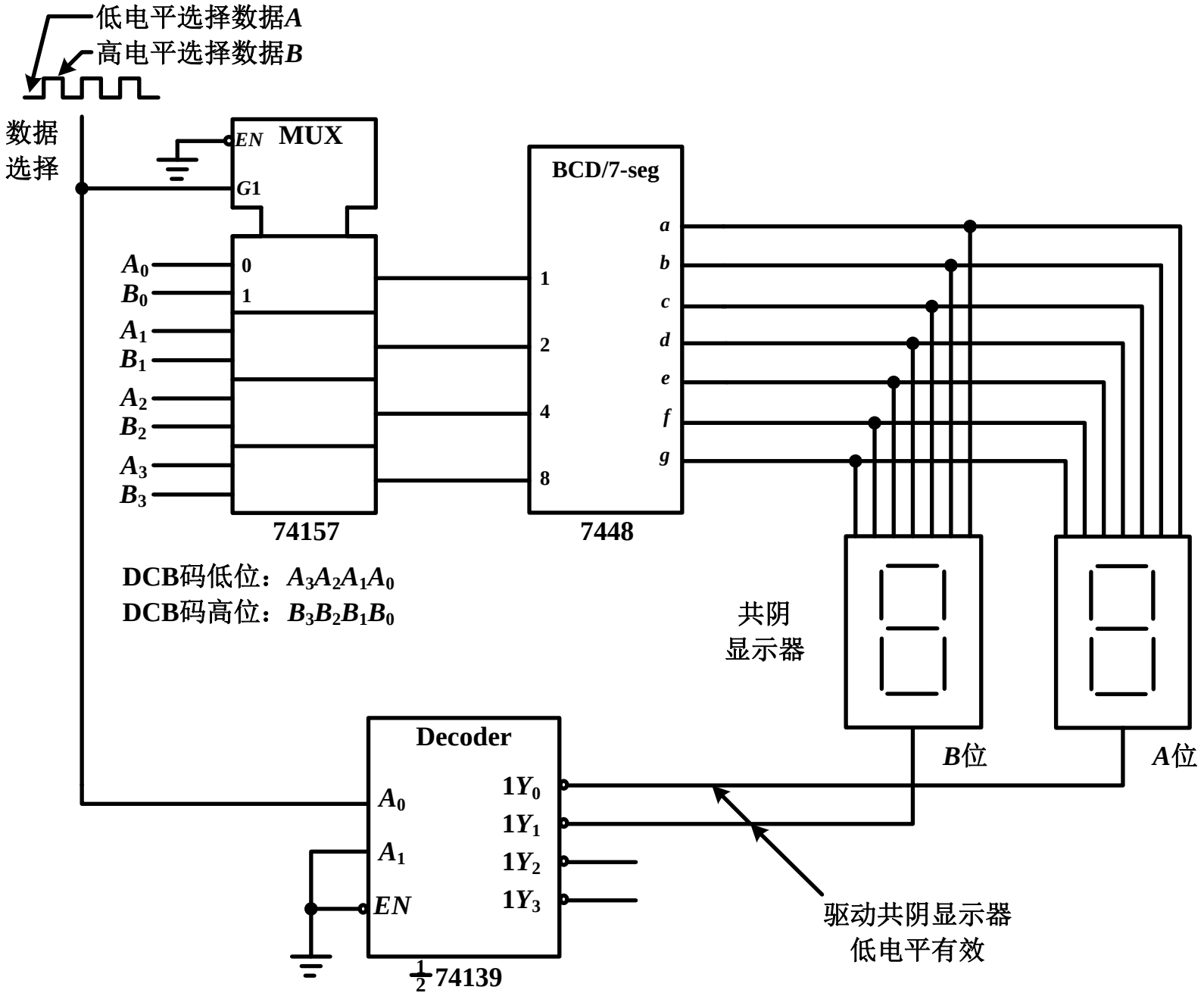
注：上面采用A、B作为地址变量。实际上，地址变量的选取是任意的，选不同的变量为地址变量时，数据输入端的信号也要随之变化。

2. 动态显示电路

七段数码管驱动电路可分为两种，一种称为静态显示，另一种称为动态显示。

静态显示：每一个数码管由单独的七段显示译码器驱动。

动态显示：使用数据选择器的分时复用功能，将任意多个数码管的显示驱动，由一个七段显示译码器来完成。



利用数据选择器实现数码管动态显示

4.4.4数据选择器的VHDL描述

4选1数据选择器的VHDL描述

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY mux4 IS
PORT( i0,i1,i2,i3,a,b:  IN STD_LOGIC;
                    q :  OUT STD_LOGIC);
END mux4;
```

```
ARCHITECTURE behavior OF  mux4 IS
SIGNAL  sel: STD_LOGIC_VECTOR(1 DOWNTO 0) ;
BEGIN
  sel<=b&a;
  q<=i0 WHEN sel="00" ELSE
        i1 WHEN sel="01" ELSE
        i2 WHEN sel="10" ELSE
        i3 WHEN sel="11" ELSE
      'X';
END behavior;
```

总线数据选择器的VHDL描述

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY bus_mux4 IS
PORT( i0,i1,i2,i3:  IN STD_LOGIC_VECTOR(3 DOWNTO 0);
                a,b:  IN STD_LOGIC;
                  q :  OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END bus_mux4;
```

```
ARCHITECTURE behavior OF  bus_mux4 IS
SIGNAL  sel: STD_LOGIC_VECTOR(1 DOWNTO 0) ;
BEGIN
  sel<=b&a;
  q<=i0 WHEN sel="00" ELSE
        i1 WHEN sel="01" ELSE
        i2 WHEN sel="10" ELSE
        i3 WHEN sel="11" ELSE
        "XXXX";
END behavior;
```

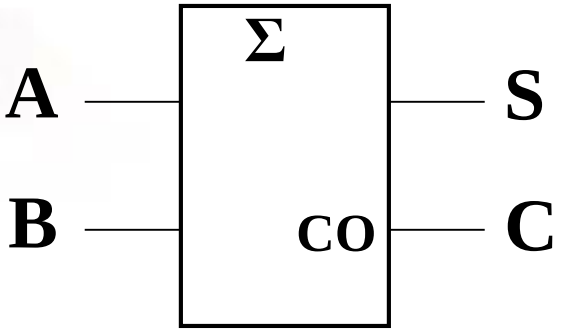
4.5 算术运算电路

算术运算电路的核心为加法器.

4.5.1 基本加法器

1. 半加器(HA)

仅考虑两个一位二进制数相加，而不考虑低位的进位,称为半加。



半加器逻辑符号

设: A、B为两个加数，S 为本位的和，C 为本位向高位的进位。则半加器的真值表、方程式、逻辑图如下所示

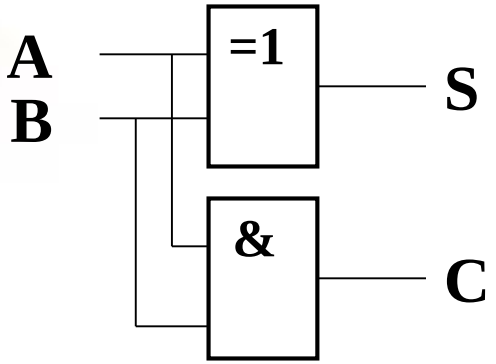
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

真值表

$S=A\oplus B$

$C=AB$

逻辑方程



逻辑图

2. 全加器

在多位数相加时,除考虑本位的两个加数外,还须考虑低位向本位的进位.

例:

	1	1	0	1	加数
	1	1	1	1	加数
+)	1	1	1	1	0	低位向高位的进位
	1	1	1	0	0	和

实际参加一位数相加,必须有三个量,它们是:

本位加数 A_i、B_i;　　低位向本位的进位 C_{i-1}

一位全加器的输出结果为:

本位和 S_i;　　本位向高位的进位 C_i

全加器电路设计：

A_i	B_i	C_{i-1}	C_i	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

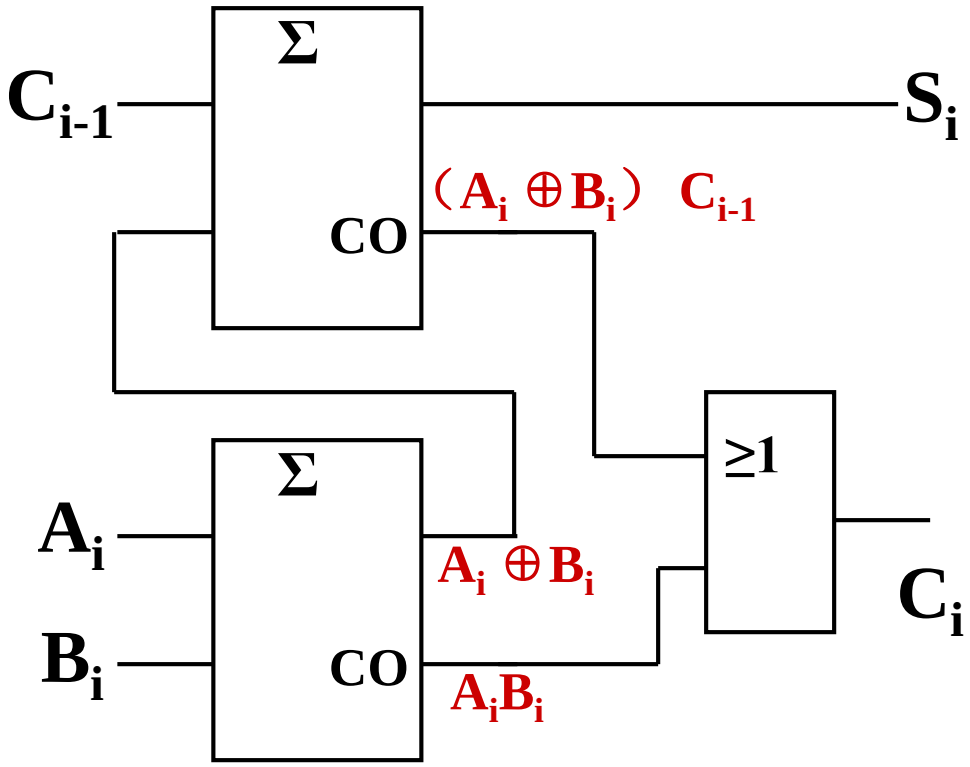
$$S_i=(\overline{A}_i\overline{B}_i+A_iB_i)C_{i-1}+(\overline{A}_iB_i+A_i\overline{B}_i)\overline{C}_{i-1}$$

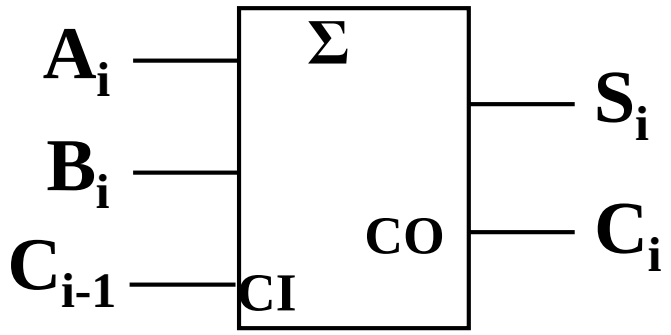
$$=A_i\oplus B_i\oplus C_{i-1}$$

$$C_i=(\overline{A}_iB_i+A_i\overline{B}_i)C_{i-1}+A_iB_i$$

$$=(A_i\oplus B_i)C_{i-1}+A_iB_i$$

由两个半加器实现一个全加器

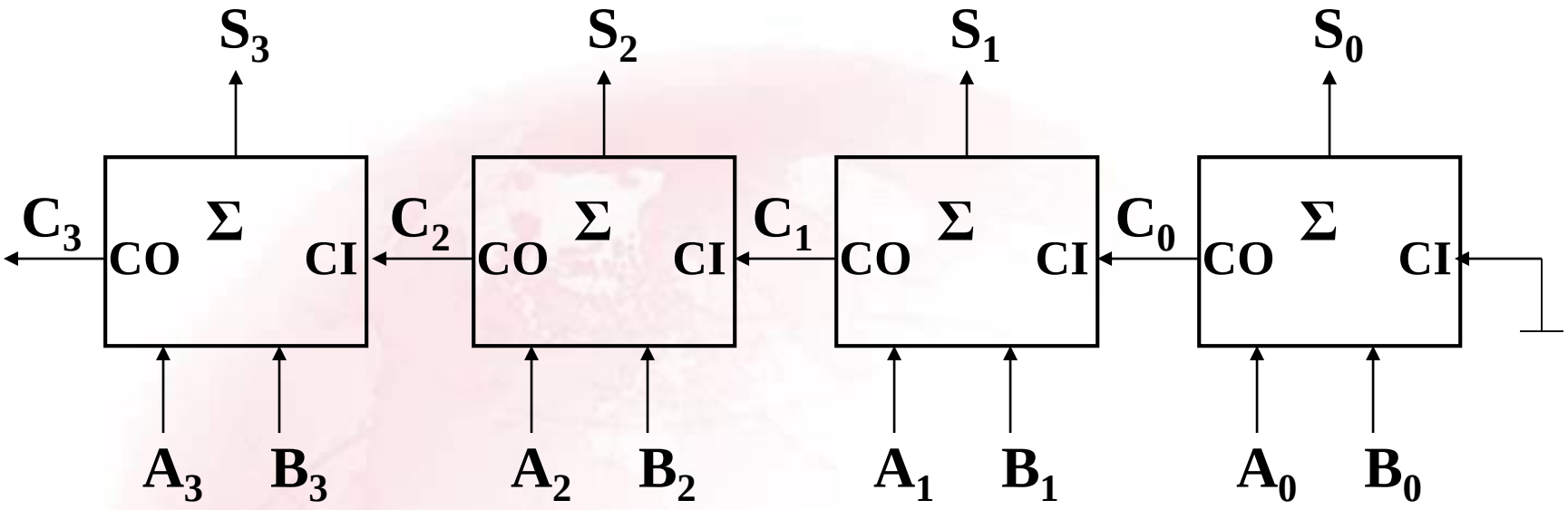




全加器逻辑符号

3. 串行进位加法器

当有多位数相加时,可模仿笔算,用全加器构成串行进位加法器.

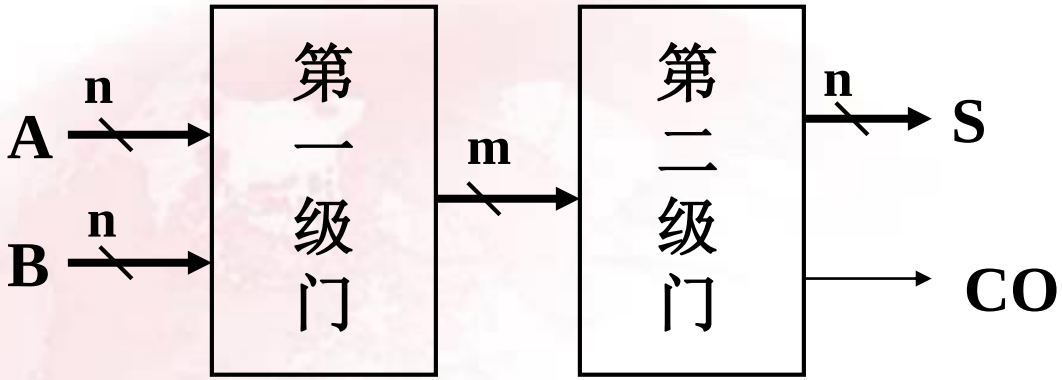


四位串行进位加法器

串行进位加法器特点: 结构简单; 运算速度慢.

4.5.2 高速加法器

(1) 全并行加法器



特点：速度最快，电路复杂。

(2) 超前进位加法器

设计思想：由两个加数,首先求得各位的进位,然后再经全加器算出结果.

全加器的进位表达式:

$$C_i = (\overline{A}_i B_i + A_i \overline{B}_i) C_{i-1} + A_i B_i$$

$$= A_i B_i + (A_i \oplus B_i) C_{i-1} = A_i B_i + (A_i + B_i) C_{i-1}$$

令: $G_i = A_i B_i$---进位产生项

$P_i = (A_i \oplus B_i) = (A_i + B_i)$---进位传送项

则: $C_i = G_i + P_i C_{i-1}$

若两个三位二进制数相加

$A=A_2A_1A_0$　　$B=B_2B_1B_0$

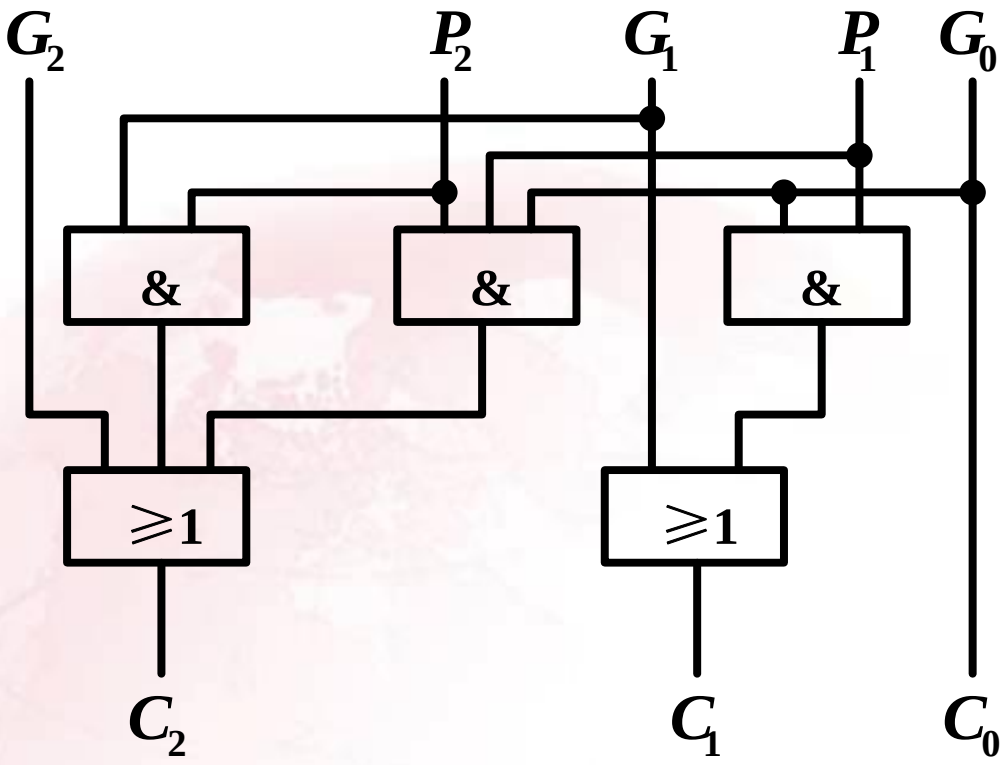
则：$C_0=G_0$；　　$C_1=G_1+P_1C_0=G_1+P_1G_0$；

$C_2=G_2+P_2C_1=G_2+P_2G_1+P_2P_1G_0$

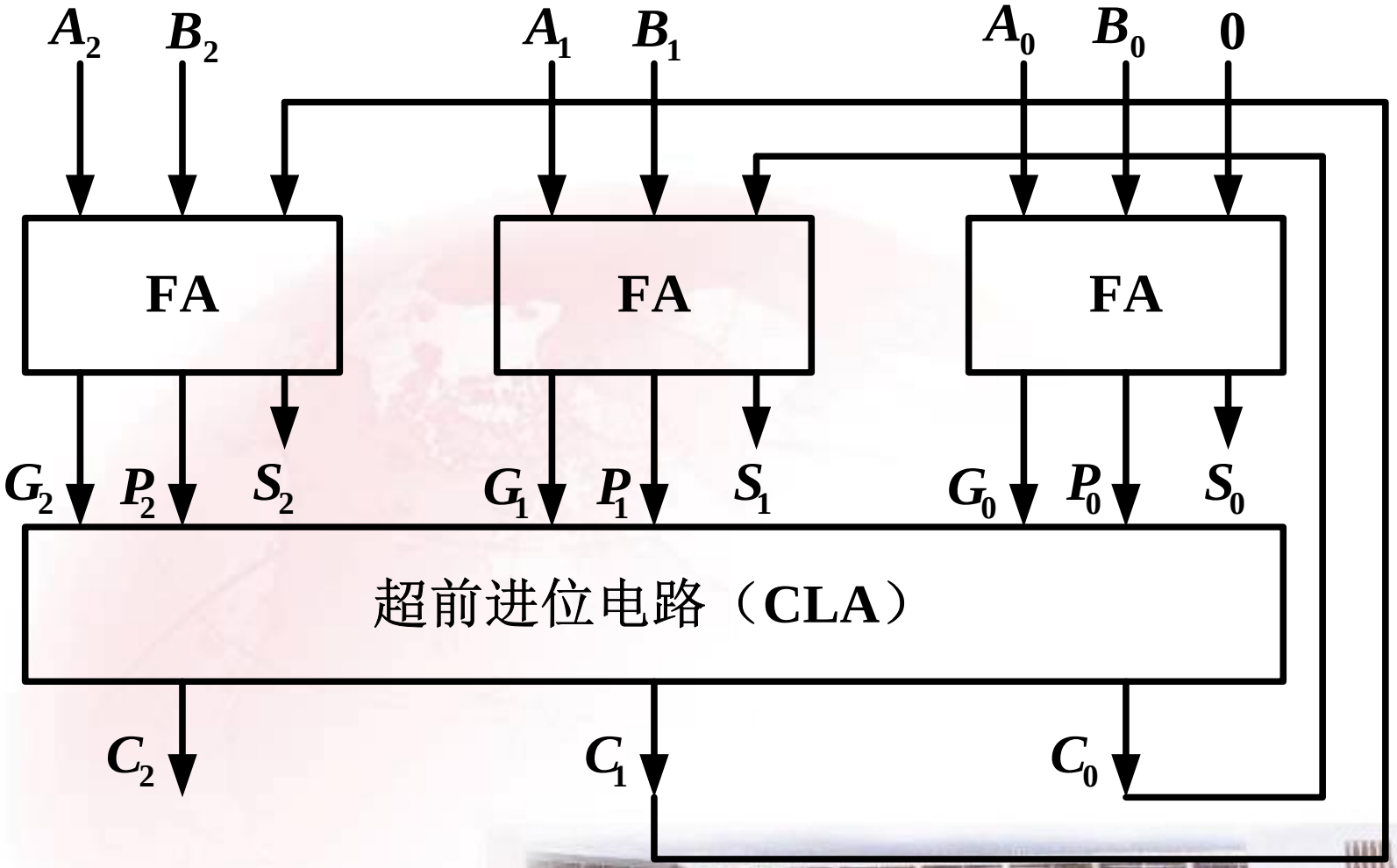
由P_i、G_i并经过两级门电路就可求得进位信号C.实际实现中,是将求G_i和P_i的电路放进全加器中,而将全加器中求进位信号的电路去除.

根据G_i、P_i来求进位信号C 的电路称为超前进位电路（CLA）

CLA逻辑图：



3位超前进位加法器


A_2 B_2
A_1 B_1
A_0 B_0 0
FA
FA
FA
G_2 P_2 S_2
G_1 P_1 S_1
G_0 P_0 S_0
超前进位电路（CLA）
C_2
C_1
C_0

超前进位加法器的缺点在于超前进位逻辑的产生，随着位数的增加，C_i会变得很复杂。

超前进位加法器的实现通常以4位为基本模块，以分层结构实现位数为4 的倍数的加法器。由C_3完整表达式：

$$C_3=G_3+P_3C_2=G_3+P_3G_2+P_3P_2G_1+P_3P_2P_1G_0+P_3P_2P_1P_0C_{-1}$$

可写为：

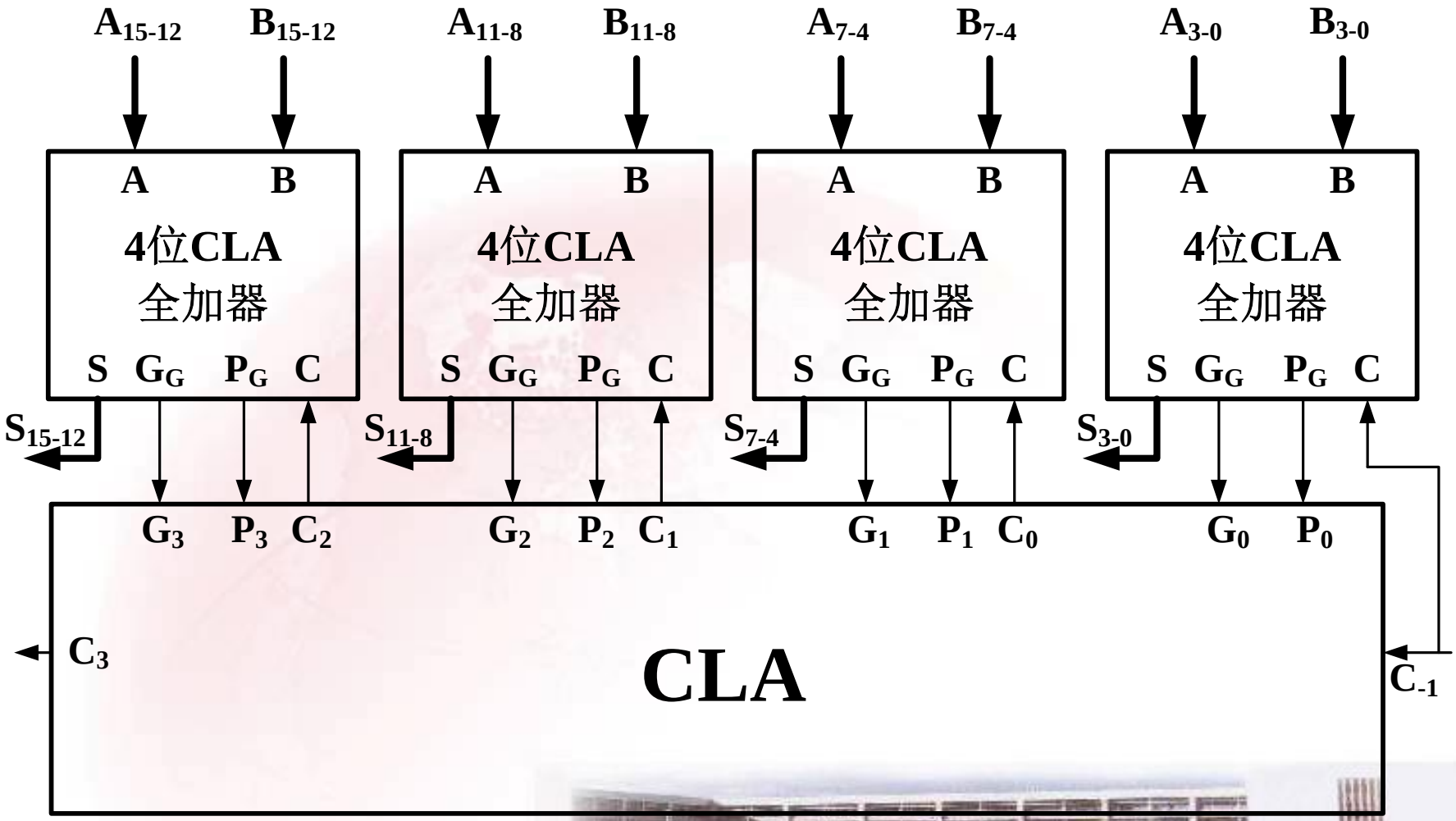
$$C_3=G_G+P_GC_{-1}=G_3+P_3G_2+P_3P_2G_1+P_3P_2P_1G_0+P_3P_2P_1P_0C_{-1}$$

其中 $G_G=G_3+P_3G_2+P_3P_2G1+P_3P_2P_1G_0$

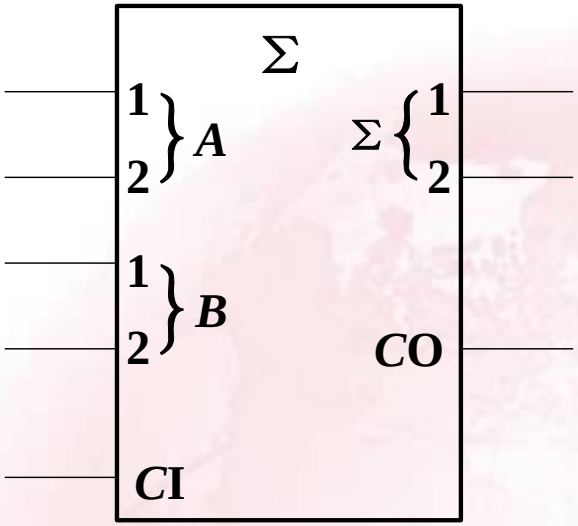
$P_G=P_3P_2P_1P_0$

表达式中的C_3作为本模块向高位模块的进位，而C_{-1}为低位模块给本模块的进位。

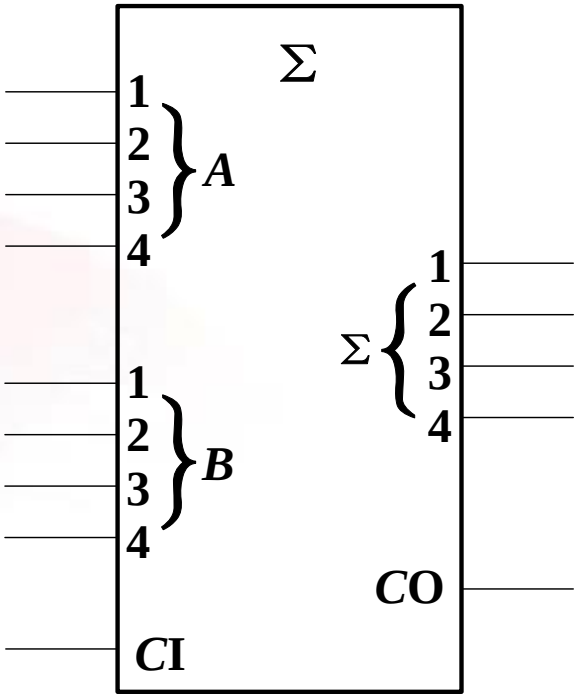
16位超前进位加法器：


A15-12
B15-12
A11-8
B11-8
A7-4
B7-4
A3-0
B3-0
A
B
4位CLA
全加器
S
GG
PG
C
S15-12
S11-8
S7-4
S3-0
G3
P3
C2
G2
P2
C1
G1
P1
C0
G0
P0
C3
C-1
CLA

4.5.3 通用加法器集成电路



2位并行加法器7482



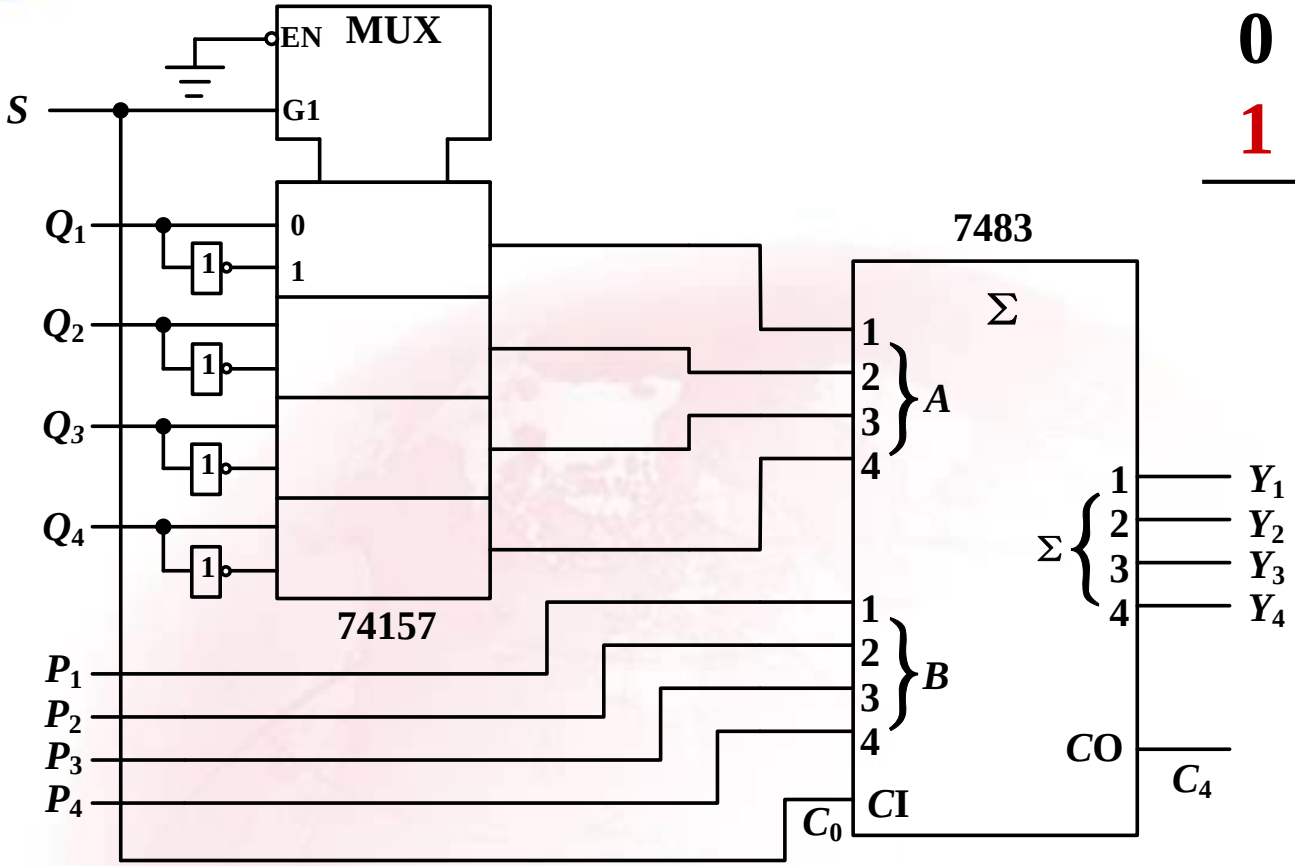
4位并行加法器7483

4.5.4 加法器应用举例

1. 用4×2选1数据选择器74157和4位全加器7483，构成4位二进制加/减器。

在二进制补码系统中，减法功能由加“减数”的补码来实现。

S	功能
0	$(P)_2+(Q)_2$
1	$(P)_2-(Q)_2$



二进制加/减器

- 关于减法电路探讨

1. 二进制减法运算

$N_{补}=2^n - N_{原}$ ($N_{原}$为n位)

$N_{原}= 2^n - N_{补}$

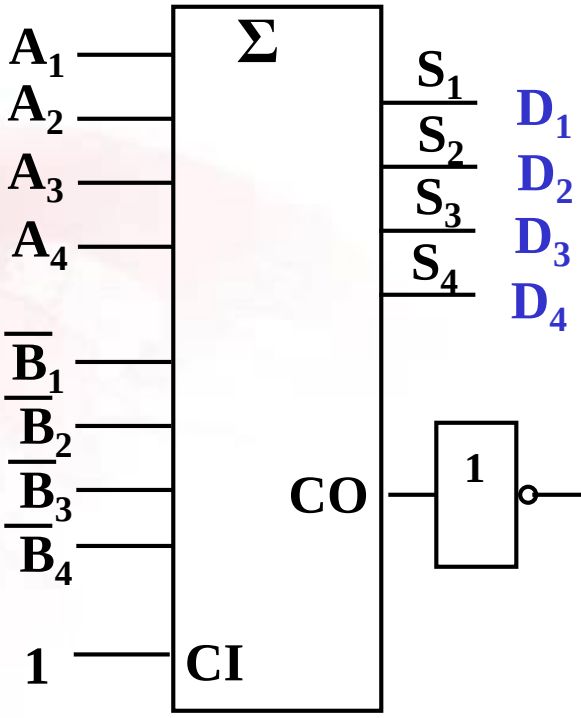
$N_{补}=N_{反} + 1$

$A-B = A-B_{原}$

$= A-(2^n - B_{补})$

$= A+B_{反}+1 - 2^n$

(1)式的实现方法: (以4位数相减为例)



7483

(1)

借位信号实现减2^n 的功能：当$A+B_{反}+1$ 的高位有进位时，该进位信号和2^n 相减使最高位为0，反之为1。

2. 分两种情况讨论：

(1)A－B≥0

设　A=0101，B=0001

求补码相加演算过程如下：

```
    0101  (A)
+   1110  (B反)
          1  (加1)
-----------
  1 0100
    ↓
  0 0100  (进位反相)
借位↑
```

运算结果为4和实际相同。

(2) A－B <0

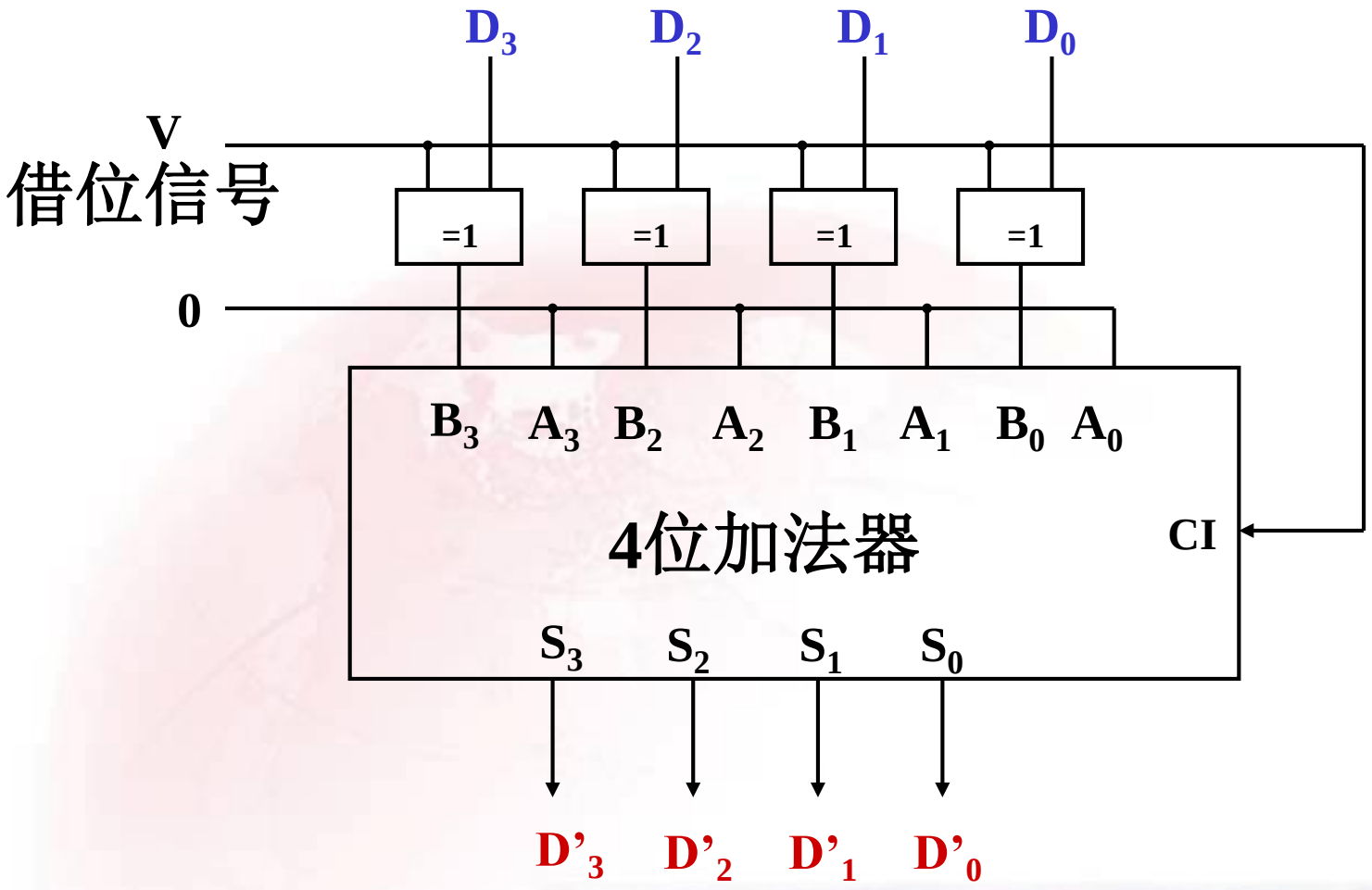
设　A= 0001，B= 0101

求补码相加演算过程如下：

```
    0001  (A)
+   1010  (B反)
          1  (加1)
-----------
  0 1100
    ↓
  1 1100  (进位反相)
借位↑
```

运算结果为－4的补码，最高位的1为符号位。

3. 由符号决定求补的逻辑图



2. 利用7483(四位二进制加法器)构成8421BCD码加法器.

二进制数和8421BCD码对照表

十进制数	二进制数(和)					8421BCD码(和)				
	C_4	S_4	S_3	S_2	S_1	K_4	B_8	B_4	B_2	B_1
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1

$S=S_4S_3S_2S_1$

$B=B_8B_4B_2B_1$

$K_4=C_4=0$

$B=S$

$K_4=\overline{C}_4=1$

$B=S+0110$ 有溢出

十进制数	二进制数(和)					8421BCD码(和)				
	C_4	S_4	S_3	S_2	S_1	K_4	B_8	B_4	B_2	B_1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

$K_4=C_4=1$

$B=S+0110$ 无溢出

总结上表,可得:

① $K_4=1$ 时,需进行加6 (0110) 校正;

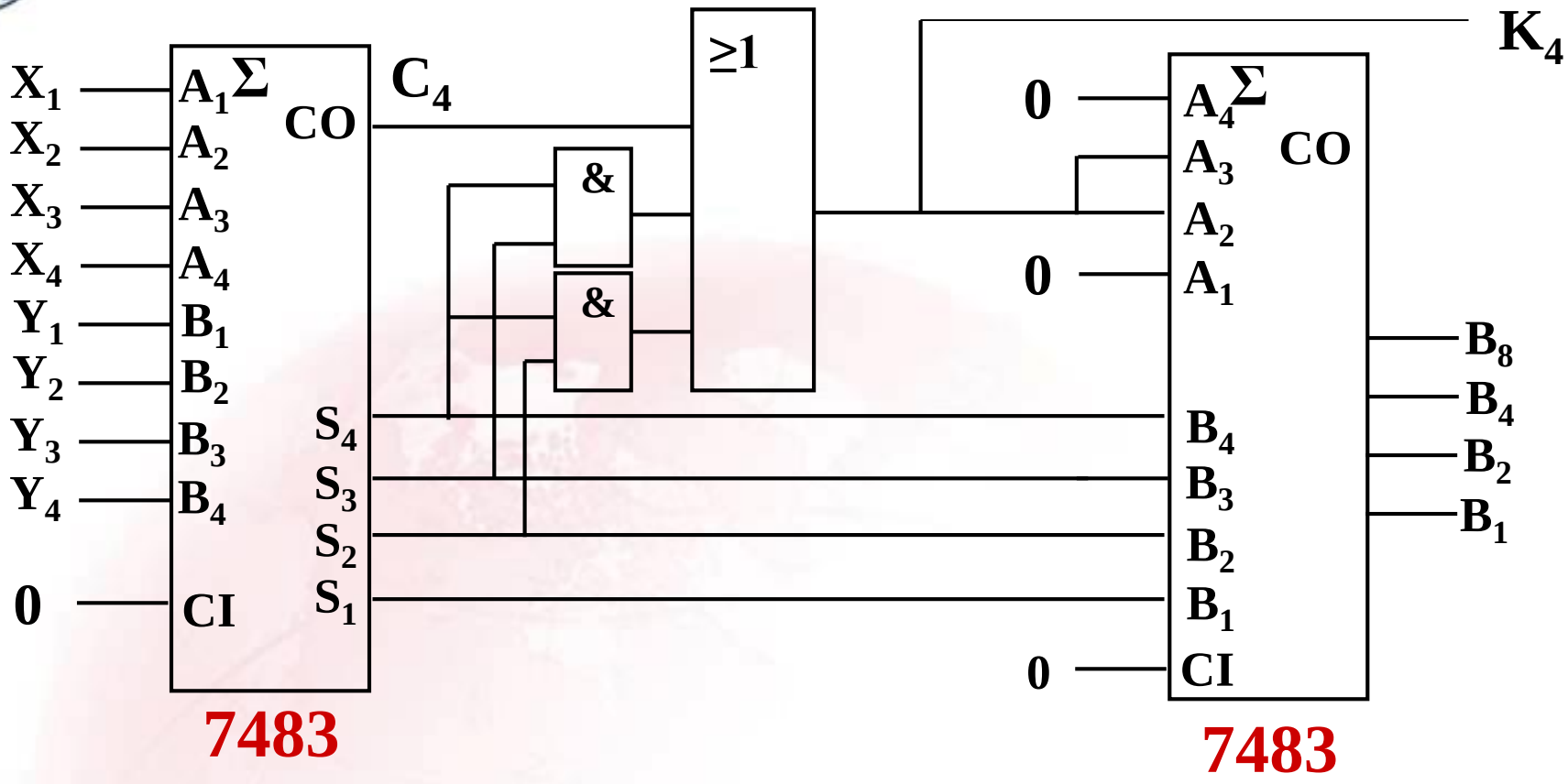
② $K_4=1$ 有三种情况:

a. $C_4=1$ (对应十进制数16,17,18,19) ;

b. $S_4=S_3=1$ (对应十进制数12,13,14,15) ;

c. $S_4=S_2=1$(对应十进制数10,11,14,15) .

所以: $K_4=C_4+S_4S_3+S_4S_2$


X1
X2
X3
X4
Y1
Y2
Y3
Y4
0
A1
A2
A3
A4
B1
B2
B3
B4
CI
Σ
CO
C4
S4
S3
S2
S1
&
&
≥1
7483
K4
0
0
0
A4
A3
A2
A1
B4
B3
B2
B1
CI
Σ
CO
B8
B4
B2
B1
7483

8421码加法器

4.5.5 加法器电路的VHDL描述

1. 用VHDL语言描述半加器

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY half_adder IS
PORT( a,b:   IN STD_LOGIC;
          s,co:  OUT STD_LOGIC);
END half_adder;
ARCHITECTURE  rtl OF  half_adder IS
BEGIN
      s<= a XOR b;
     co<= a AND b;
END rtl;
```

2. 全加器的VHDL描述

下面全加器的设计思路为：利用已有的半加器相连接，构成全加器。（另要使用一次或运算）

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY full_adder IS
PORT( a,b,cin:  IN STD_LOGIC;
              s,co:  OUT STD_LOGIC);
END full_adder;
ARCHITECTURE  str OF  full_adder IS
  COMPONENT half_adder
```

```
  PORT ( a,b:  IN STD_LOGIC;
              s,co:  OUT STD_LOGIC);
END COMPONENT;
SIGNAL u0_co,u0_s,u1_co: STD_LOGIC;
  BEGIN
   u0:half_adder PORT MAP(a,b,u0_s,u0_co);
   u1:half_adder PORT MAP(u0_s ,cin,s,u1_co);
co <= u0_co OR u1_co;
END str;
```

元件描述语句：

```
COMPONENT 元件名
  PORT   说明；   --端口说明
END COMPONENT;
```

元件例化语句：

标号名：元件名 **PORT MAP** (信号，…)

位置映射：

u0:half_adder PORT MAP(a,b,u0_s,u0_co)

名称映射：

u0:half_adder PORT MAP(a＝>a,s=>u0_s, co=>u0_co,b=b)

3. 一位8421BCD码加法器的VHDL描述

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_ARITH.ALL;
ENTITY bcdadder IS
    PORT (  cin      : IN    STD_LOGIC ;
            op1,op2  : IN    UNSIGNED(3 DOWNTO 0);
            sum      : OUT   UNSIGNED(3 DOWNTO 0);
             co      : OUT   STD_LOGIC);
END bcdadder;
```

```
ARCHITECTURE  behavior OF  bcdadder IS
   CONSTANT  adjustnum : UNSIGNED:=“110”;
    --定义一常量:整数型,值为6
   SIGNAL  binadd,result     : UNSIGNED(4 DOWNTO 0);
   --定义一个信号binadd,以保存两数二进制相加的和;
   --result用于暂存和值的BCD码。
BEGIN
   binadd<=('0'&op1)+op2+cin;
   --信号赋值,获得两数的二进制和。
PROCESS(binadd)
     VARIABLE   tmp : UNSIGNED(2 DOWNTO 0):=“000”;
      -- 定义一个变量，初值为0。
```

```
BEGIN
   IF  binadd>9 THEN  --判断结果是否大于9，
                      --如大于9，需要加6调整。
      tmp:=adjustnum;
   ELSE
      tmp:="000";
   END  IF;
   result<=binadd+tmp; --信号赋值,获得两数和的BCD码。
  END PROCESS;
 sum<=result(3 DOWNTO 0); --得到BCD码的低4位
 co<= result(4); --得到进位信号
END  behavior;
```

4.6 数值比较器

数值比较器用来判断两个二进制数的大小或相等.

4.6.1 一位数值比较器

真 值 表

A	B	$Y_{(A>B)}$	$Y_{(A<B)}$	$Y_{(A=B)}$
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

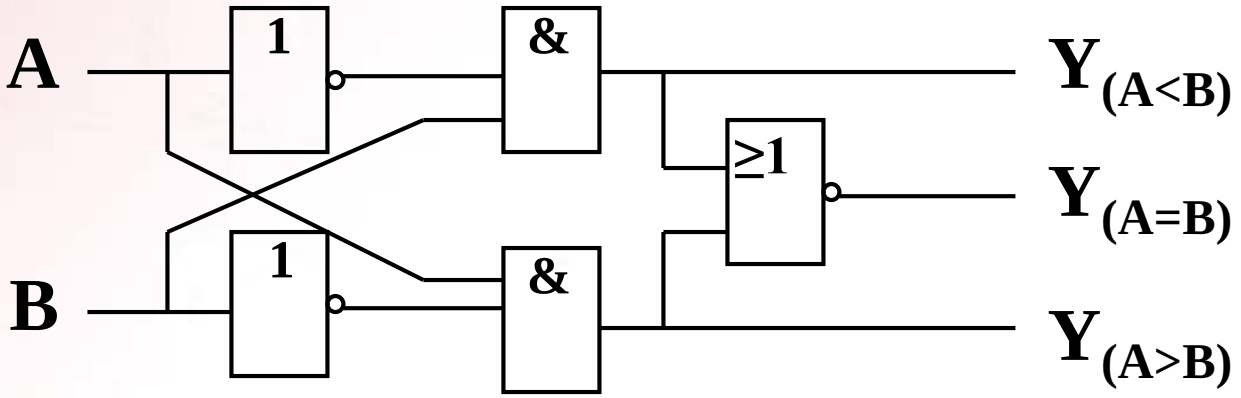
表达式:

$$Y_{(A>B)}=A\overline{B}$$

$$Y_{(A<B)}=\overline{A}B$$

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

逻辑图



4.6.2 多位数值比较器

比较两个多位数,应首先从高位开始,逐位比较。

例如: $A=A_3A_2A_1A_0$ $B=B_3B_2B_1B_0$

比较方法为:

① 首先比较A_3和B_3，如$A_3B_3=10$，则$A>B$,如$A_3B_3=01$，则$A<B$; 如$A_3B_3=00$或11(相等), 则比较A_2和B_2;

② 比较A_2和B_2，如$A_2B_2=10$，则$A>B$,如 $A_2B_2=01$，则$A<B$;如$A_2B_2=00$或11 (相等), 则比较A_1和B_1;

③ 比较A_1和B_1，如$A_1B_1=10$，则$A>B$,如 $A_1B_1=01$，则$A<B$;如$A_1B_1=00$或11 (相等), 则比较A_0和B_0;

④ 比较A_0和B_0，如$A_0B_0=10$，则$A>B$,如 $A_0B_0=01$，则$A<B$;如$A_0B_0=00$或11 (相等), 则比较$A=B$.

四位数值比较器逻辑表达式:

$$Y_{(A>B)}=A_3\overline{B}_3+(A_3\odot B_3)\,A_2\overline{B}_2+(A_3\odot B_3)\,(A_2\odot B_2)\,A_1\overline{B}_1 + (A_3\odot B_3)\,(A_2\odot B_2)\,(A_1\odot B_1)\,A_0\overline{B}_0$$

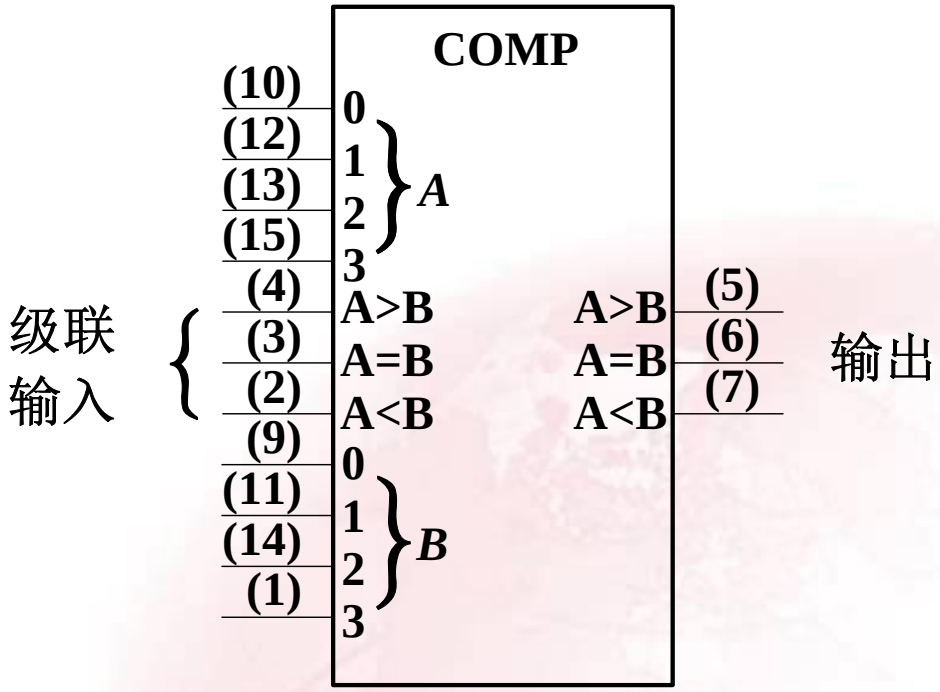
$$Y_{(A<B)}=\overline{A}_3B_3+(A_3\odot B_3)\,\overline{A}_2B_2+(A_3\odot B_3)\,(A_2\odot B_2)\,\overline{A}_1B_1 + (A_3\odot B_3)\,(A_2\odot B_2)\,(A_1\odot B_1)\,\overline{A}_0B_0$$

$$Y_{(A=B)}=(A_3\odot B_3)\,(A_2\odot B_2)\,(A_1\odot B_1)(A_0\odot B_0)$$

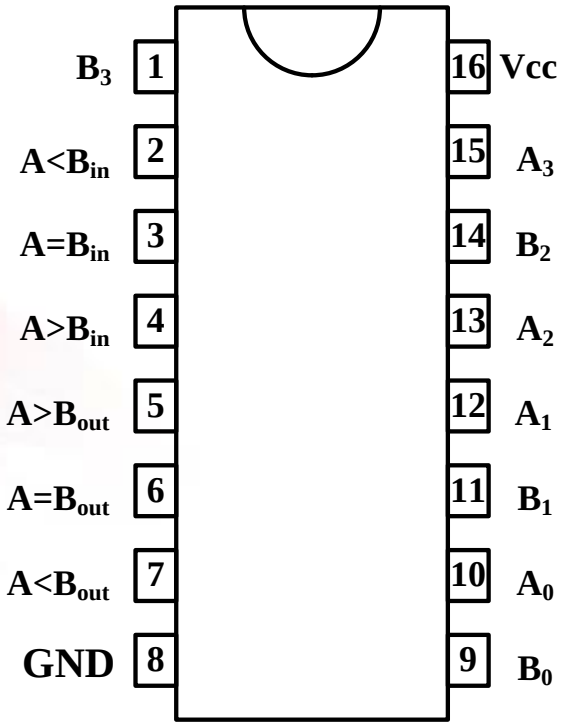
4.6.3 通用数值比较器集成电路

通用数值比较器集成电路有多个品种，属CMOS电路的4位数值比较器的有74HC85(对应的TTL电路为74LS85)、CC14585等。

74HC85为带级联输入的4位数值比较器。


COMP
(10)
(12)
(13)
(15)
0
1
2
3
A
(4)
(3)
(2)
A>B
A=B
A<B
(9)
(11)
(14)
(1)
0
1
2
3
B
A>B
A=B
A<B
(5)
(6)
(7)
级联输入
输出

逻辑符号

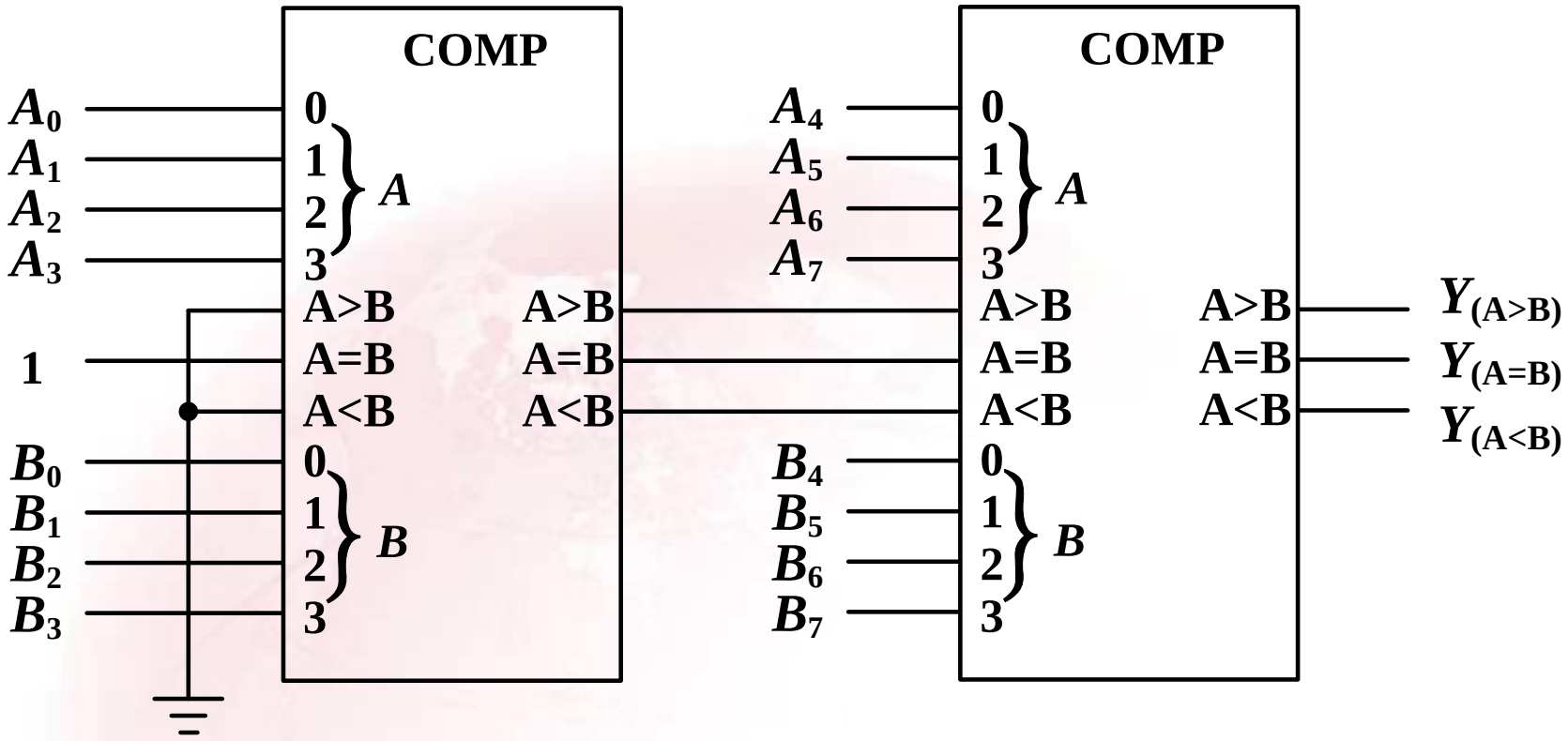

B3 1
A<Bin 2
A=Bin 3
A>Bin 4
A>Bout 5
A=Bout 6
A<Bout 7
GND 8
16 Vcc
15 A3
14 B2
13 A2
12 A1
11 B1
10 A0
9 B0

引脚图

74HC85功能表

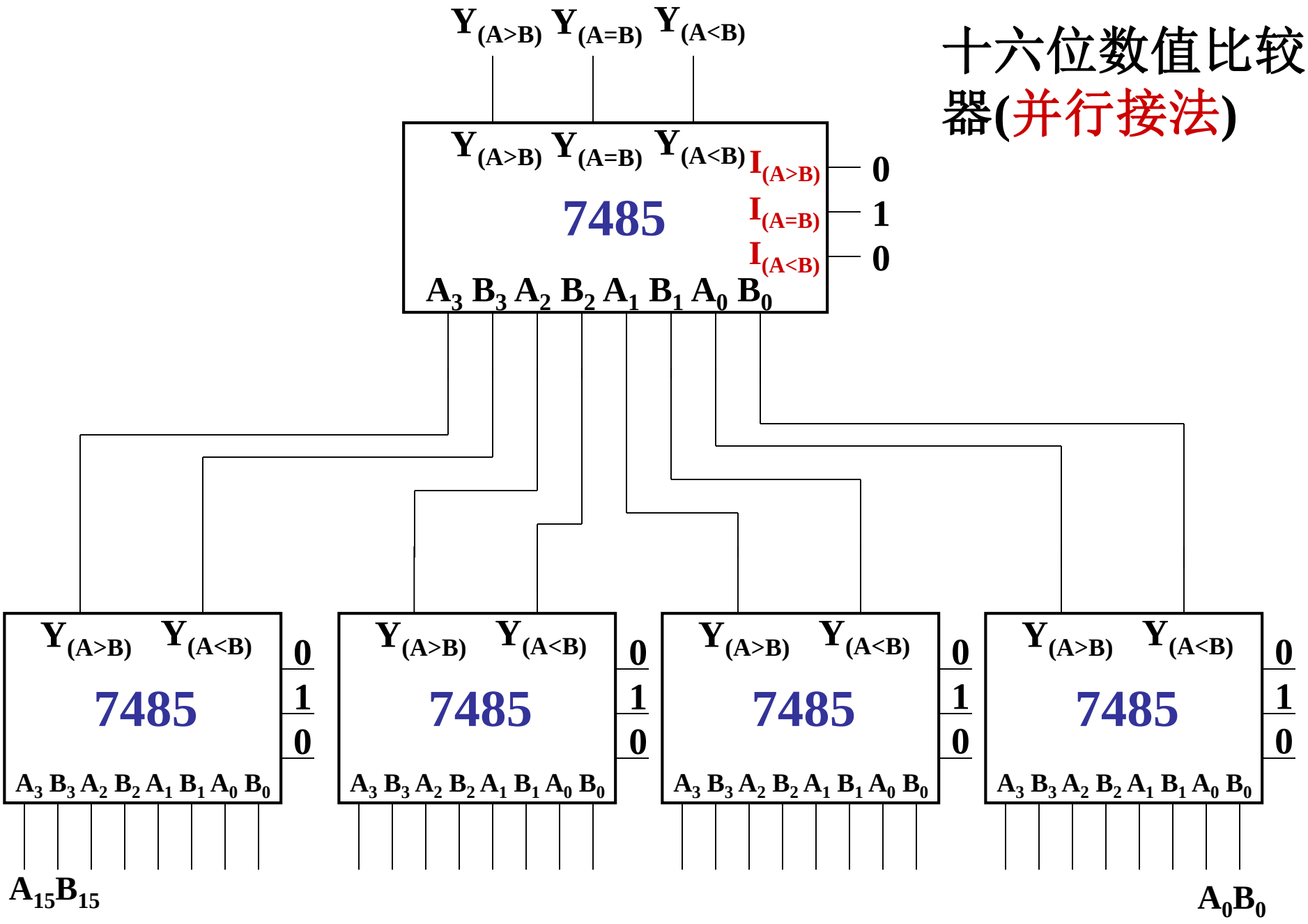
比较输入				级联输入			输出		
A_3 B_3	A_2 B_2	A_1 B_1	A_0 B_0	$I_{(A>B)}$	$I_{(A<B)}$	$I_{(A=B)}$	$Y_{(A>B)}$	$Y_{(A<B)}$	$Y_{(A=B)}$
$A_3>B_3$	×	×	×	×	×	×	1	0	0
$A_3<B_3$	×	×	×	×	×	×	0	1	0
$A_3=B_3$	$A_2>B_2$	×	×	×	×	×	1	0	0
$A_3=B_3$	$A_2<B_2$	×	×	×	×	×	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	×	×	×	×	1	0	0
$A_3=B_3$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

比较器的扩展：



8位位数值比较器（串行接法）

十六位数值比较器(并行接法)



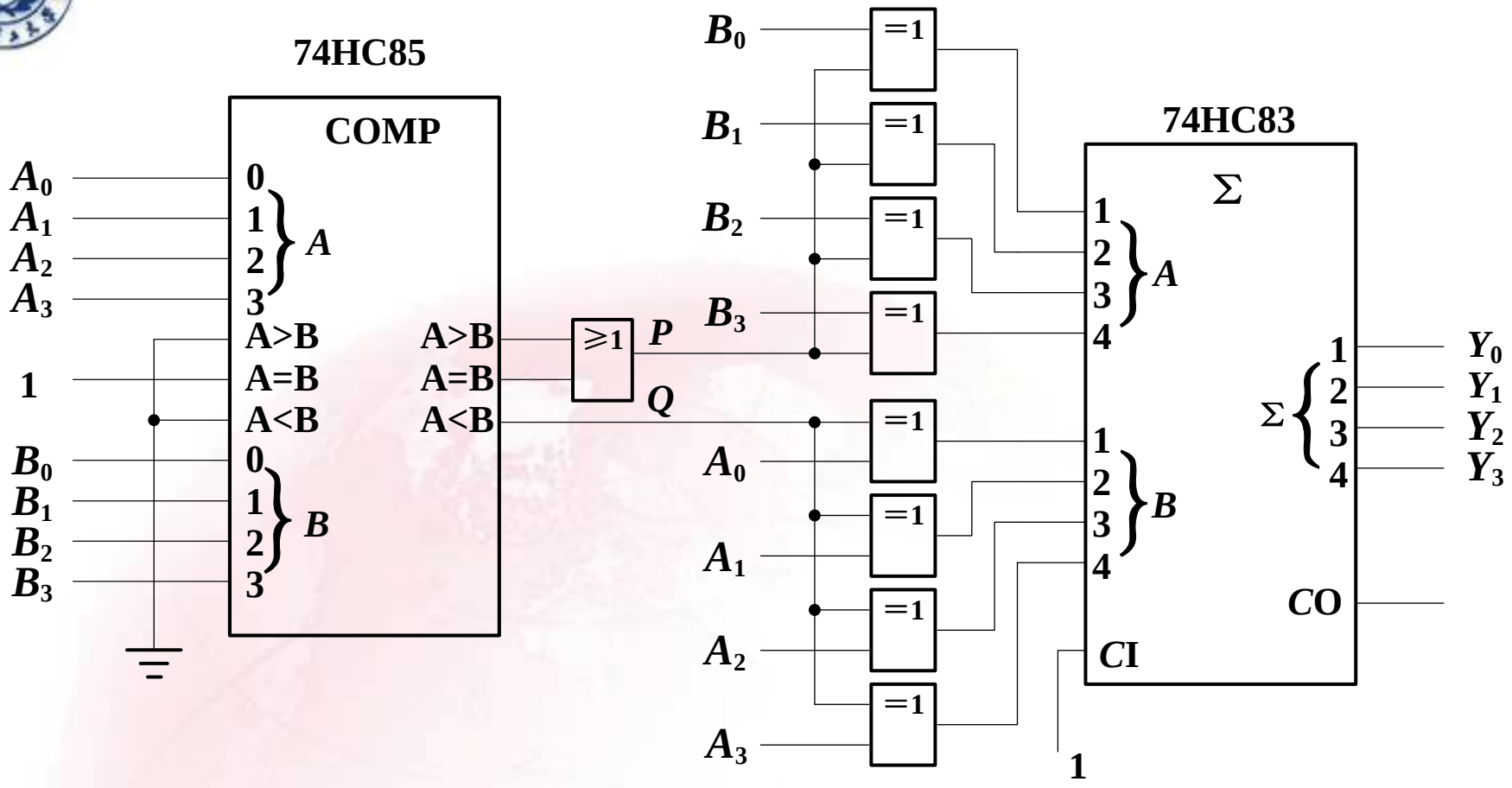
串行接法和并行接法性能比较:

串行接法电路简单,但速度慢;并行接法电路复杂,速度快.

4.6.4 数值比较器应用举例

例：设计一个求两数之差绝对值电路。

设计思路：先将两数比较，对小的数求补，将得到的补码与另一数相加，得到结果。


74HC85
COMP
A_0
A_1
A_2
A_3
1
B_0
B_1
B_2
B_3
0
1
2
3
A
A>B
A=B
A<B
0
1
2
3
B
A>B
A=B
A<B
≥1
P
Q
B_0
B_1
B_2
B_3
=1
=1
=1
=1
A_0
A_1
A_2
A_3
=1
=1
=1
=1
74HC83
Σ
1
2
3
4
A
1
2
3
4
B
CI
1
Σ
1
2
3
4
CO
Y_0
Y_1
Y_2
Y_3

求两数之差绝对值电路

4.6.5 数值比较器的VHDL描述

带级联输入的4位数值比较器VHDL描述：

```
ENTITY comparator IS
  PORT (  a,b          : IN    INTEGER RANGE 0 TO 15 ;
          gtin,ltin,eqin   : IN    BIT;  --级联输入
          agtb,altb,aeqb  : OUT  BIT);
  END comparator;
ARCHITECTURE  behavior OF  comparator IS
BEGIN
PROCESS(a,b,gtin,ltin,eqin)
```

```
BEGIN
   IF     a<b   THEN  agtb<=’0’; altb<=’1’; aeqb<=’0’;
   ELSIF  a>b    THEN  agtb<=’1’; altb<=’0’; aeqb<=’0’;
   ELSE                agtb<= gtin; altb<= ltin; aeqb<= eqin;
   END  IF;
  END PROCESS;
END  behavior;
```

4.7 代码转换器

重点介绍能实现BCD码和自然二进制码之间转换的代码转换器的设计方法，并介绍通用代码转换器集成电路的使用方法。

4.7.1 BCD－二进制码转换器

转换过程：

（1） 将BCD码中的每一位的权值用二进制数表示；

（2） 将所给BCD码中‘1’所代表的二进制数相加；

（3） 相加的结果即为所给BCD码的等效二进制数。

如两位十进制数的8421BCD码为：

$B_3B_2B_1B_0$　$A_3A_2A_1A_0$

（十位）　（个位）

BCD位权与二进制数对照表

BCD位	BCD权值	二进制数表示 64	32	16	8	4	2	1
A_0	1	0	0	0	0	0	0	1
A_1	2	0	0	0	0	0	1	0
A_2	4	0	0	0	0	1	0	0
A_3	8	0	0	0	1	0	0	0
B_0	10	0	0	0	1	0	1	0
B_1	20	0	0	1	0	1	0	0
B_2	40	0	1	0	1	0	0	0
B_3	80	1	0	1	0	0	0	0

例如，要将BCD码1000 0111（十进制数87）转换为二进制，其算式如下：

80 40 20 10 8 4 2 1

1 0 0 0 0 1 1 1

0 0 0 0 0 0 0 1

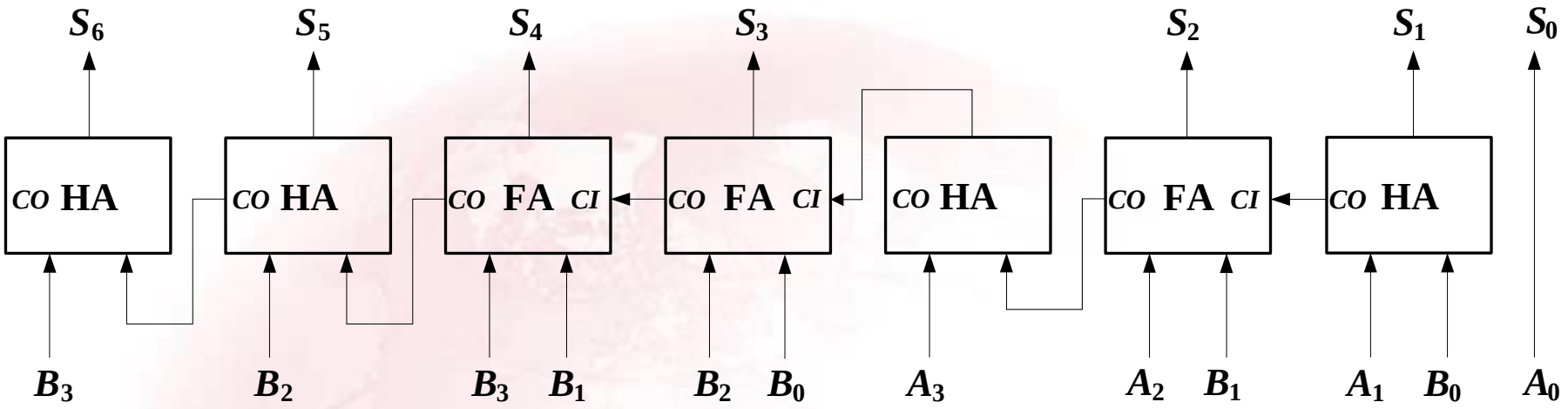
0 0 0 0 0 1 0

0 0 0 0 1 0 0

+1 0 1 0 0 0 0

1 0 1 0 1 1 1

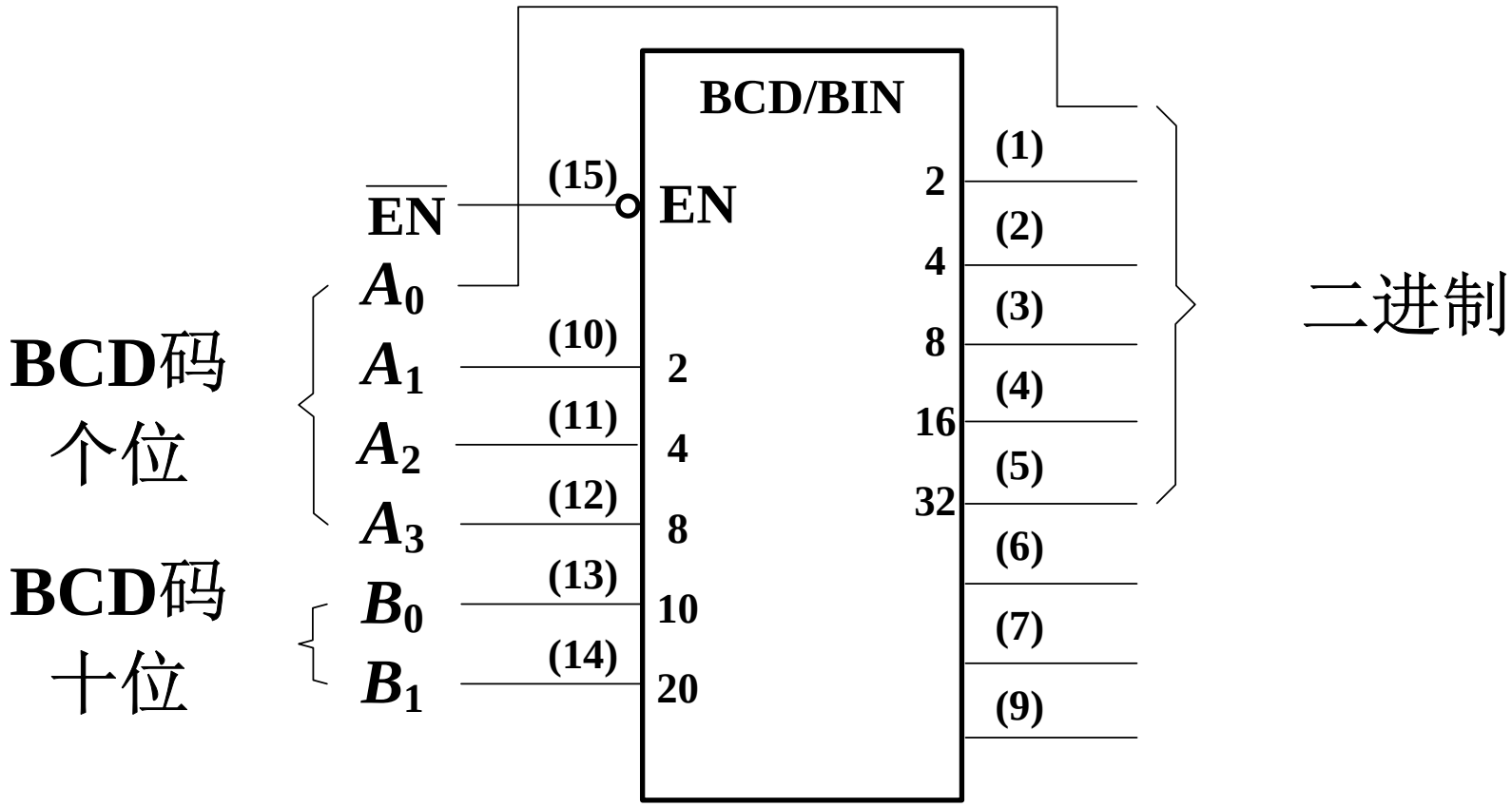
根据对照表，借助半加器和全加器，可设计出转换电路。



由半加器和全加器组成的两位BCD－二进制代码转换器

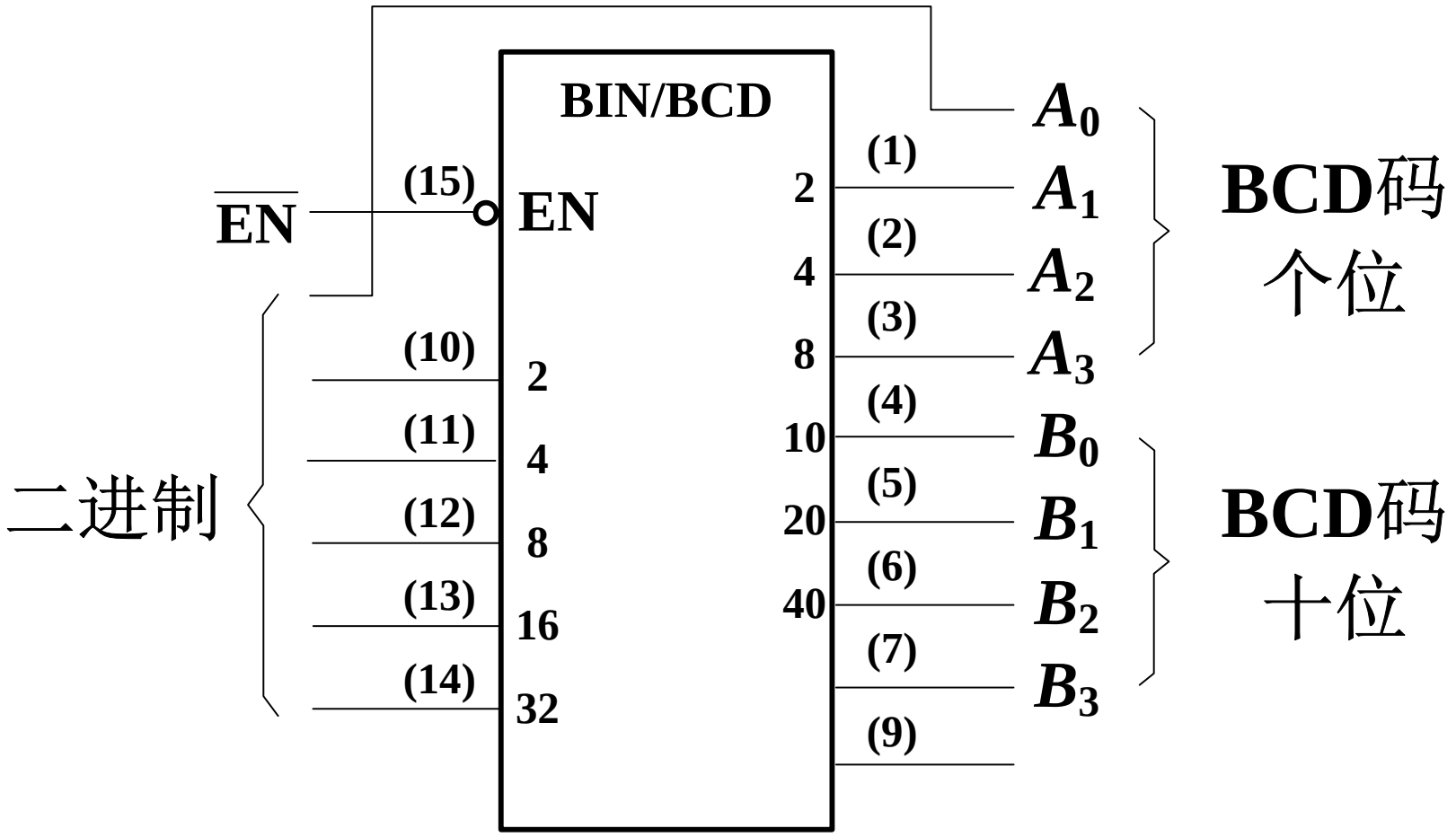
4.7.2 通用BCD－二进制和二进制－BCD码转换器集成电路

1. BCD－二进制代码转换器74184



74184 6位BCD－BIN代码转换器

2. 二进制－BCD代码转换器74185



74185 6位BIN－BCD代码转换器

4.7.3 代码转换电路的VHDL描述

1.两位BCD码（个位和十位）转换为二进制数的代码转换器的VHDL描述

```
ENTITY bcd_to_bin IS
  PORT ( ones, tens     : IN    INTEGER RANGE 0 TO 9 ;
                binary      : OUT  INTEGER RANGE 0 TO 99);
END bcd_to_bin;
ARCHITECTURE  rtl OF  bcd_to_bin IS
SIGNAL times10   : INTEGER RANGE 0 TO 90;
```

```
BEGIN
   times10<=tens*10;
   binary<=times10+ones;
END rtl;
```

2. n位二进制码转换为格雷码的码转换电路

假定n位二进制码为 $a_n \ldots a_2 a_1$

n位格雷码为 $g_n \cdots g_2 g_1$

转换表达式为：$g_n = a_n$

$$g_i = a_{i+1} \oplus a_i \qquad i \neq n$$

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY bin_to_gray IS
GENERIC(n: INTEGER:=8)
  PORT (a:  IN STD_LOGIC_VECTOR(n DOWNTO 1);
        g:  OUT  STD_LOGIC_VECTOR(n DOWNTO 1));
END bin_to_gray;
```

```
ARCHITECTURE rtl OF bin_to_gray IS
BEGIN
 PROCESS (a)
   VARIABLE tmp:STD_LOGIC_VECTOR(n DOWNTO 1);
   BEGIN
    tmp(n):=a(n);
     FOR i IN n-1 DOWNTO 1 LOOP
       tmp(i):=a(i) XOR a(i+1);
     END LOOP;
     g<=tmp;
END PROCESS;
END rtl;
```

两个语法现象：

1. 语句GENERIC(n: INTEGER:=8)定义了一个类属参数n，取值为8

2. 循环语句的书写格式为：

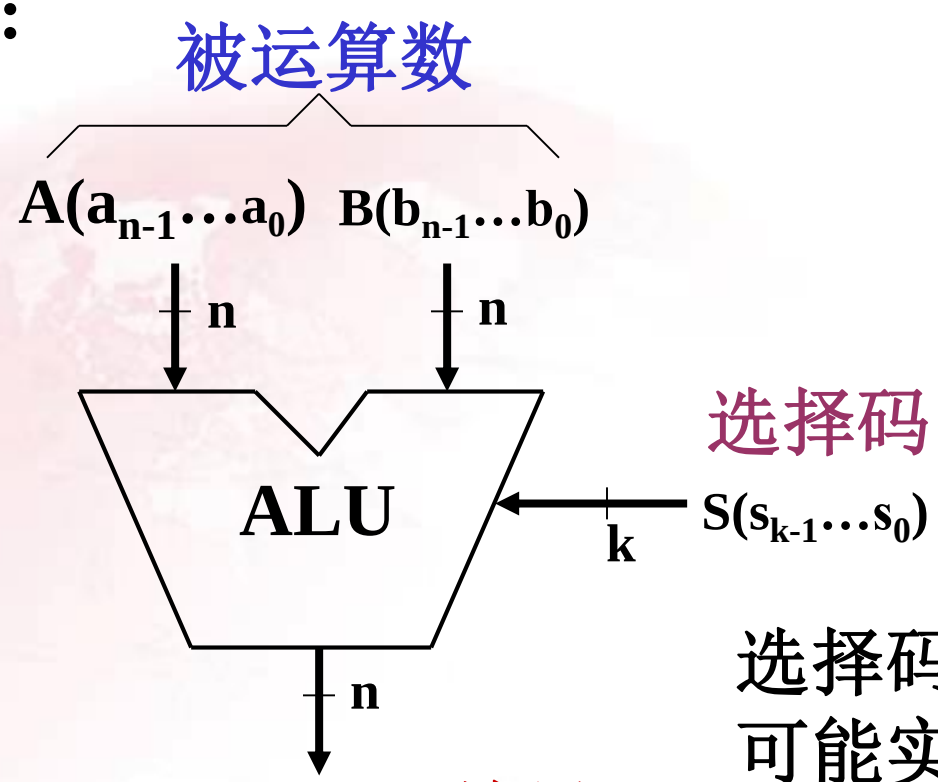
[标号]：FOR 循环变量 IN 离散变量 LOOP
顺序处理语句；
END LOOP [标号]；

4.8 数字系统设计举例: 算术逻辑单元(ALU)

算术逻辑单元(ALU)是计算机等数字系统的主要运算部件.

ALU的逻辑符号:

运算数A、B均为n 位



结果F为n 位

选择码S为k 位, 可能实现的运算为2^k 种.

设计具有八种功能的ALU. S有3 位,假设功能表如下

选择码 S_2 S_1 S_0	ALU 功能	说明
0 0 0	F=A + B	加
0 0 1	F=A－B	减
0 1 0	F=A + 1	加 1
0 1 1	F=A－1	减 1
1 0 0	F=A · B	与
1 0 1	F=A + B	或
1 1 0	F=$\bar{A}$	非
1 1 1	F=A⊕B	异或

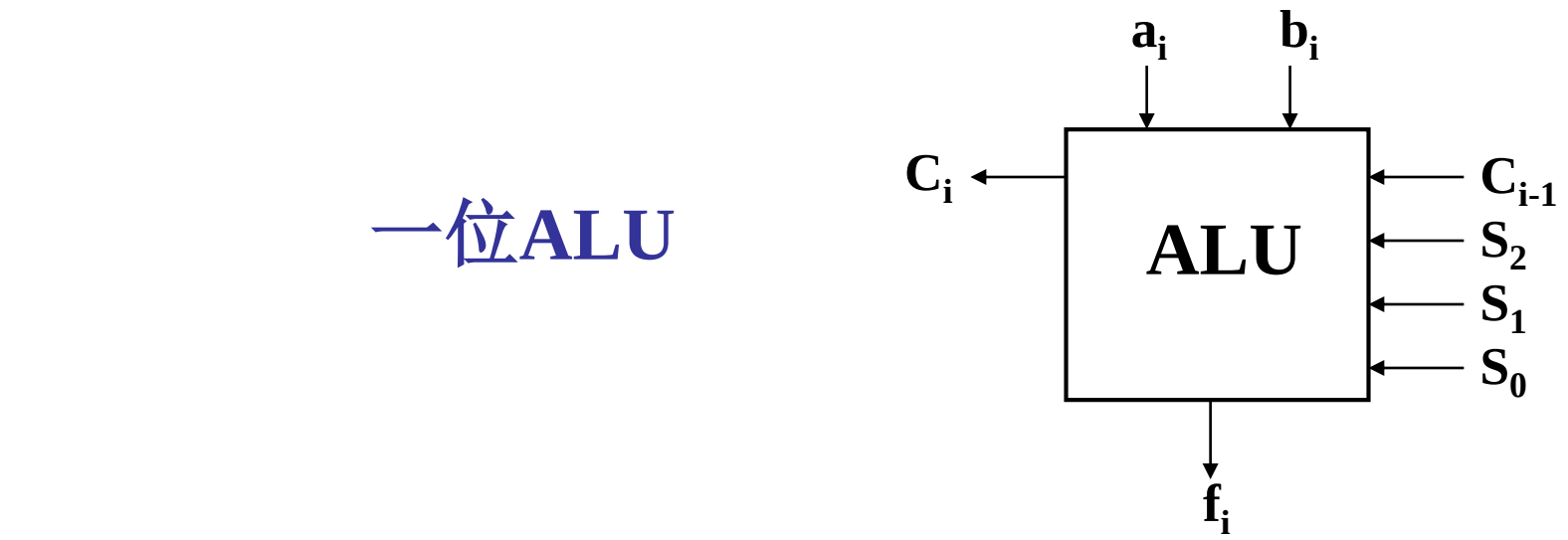
S_2=0:算术运算

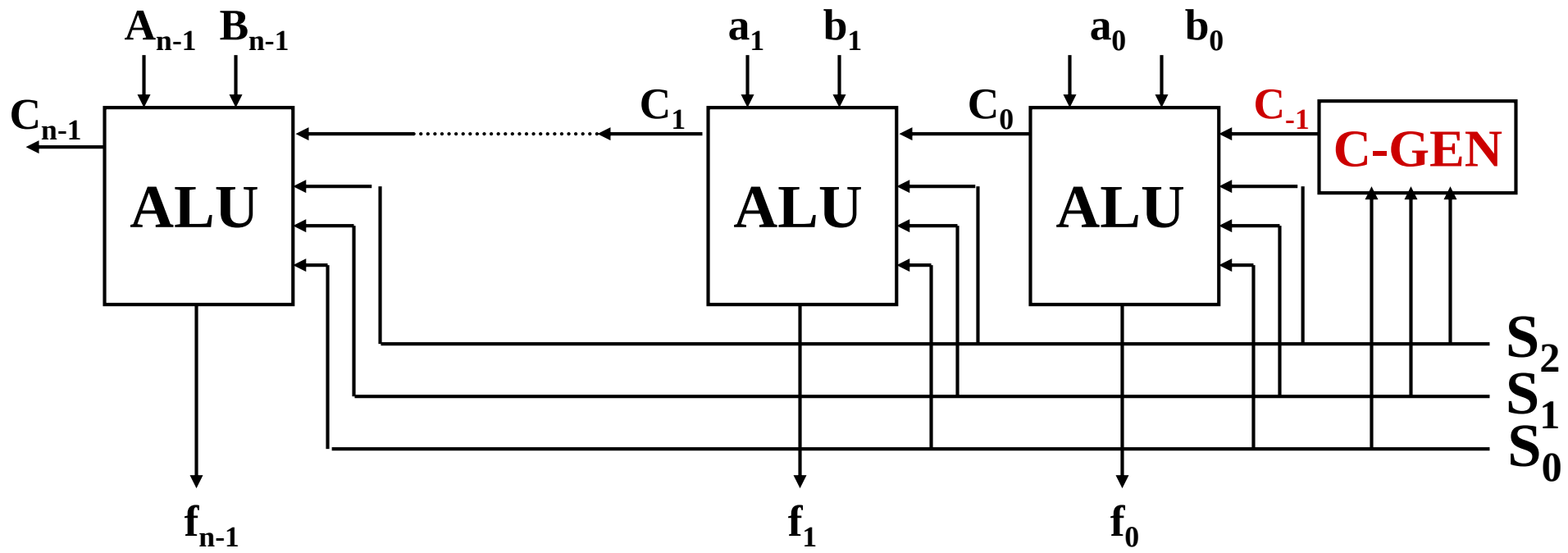
S_2=1:逻辑运算

设计思想:

设计采用自顶向下的方法,将能进行n 位运算的ALU分解为n 个能进行一位运算的ALU,最后将n 个一位ALU连接成n 位ALU.

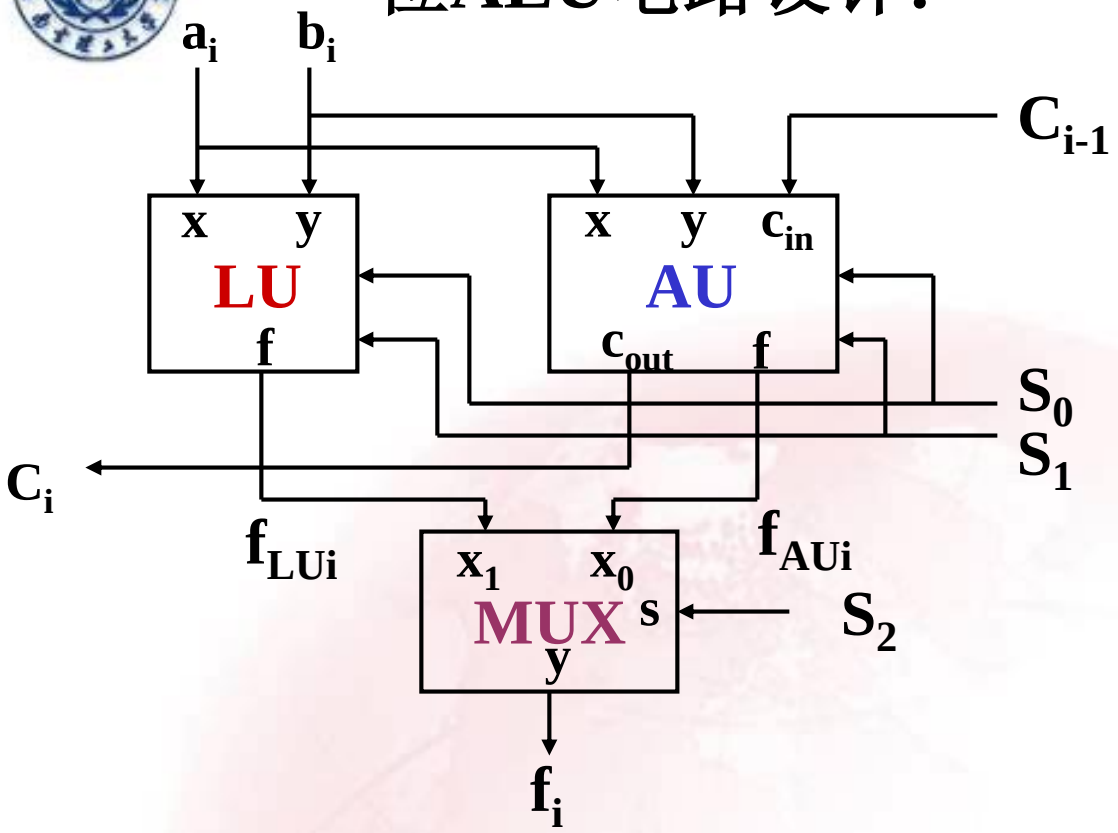
一位ALU


a_i
b_i
C_i
ALU
C_{i-1}
S_2
S_1
S_0
f_i


A_{n-1}
B_{n-1}
a_1
b_1
a_0
b_0
C_{n-1}
C_1
C_0
C_{-1}
C-GEN
ALU
ALU
ALU
S_2
S_1
S_0
f_{n-1}
f_1
f_0

n位ALU

一位ALU电路设计:



一位ALU分解图:

AU: 算术单元;

LU:逻辑单元;

MUX:数据选择器,根据S_2的值,对AU和LU的运算结果进行选择.

(1) MUX电路设计: 这里的MUX为二选一数据选择器, 设计方法前面已介绍;

(2) LU电路设计:

计算机中的逻辑运算是位操作(即对应位之间进行运算).

LU单元功能表

S_1	S_0	功 能 (f_{LUi})
0	0	a_ib_i
0	1	a_i+b_i
1	0	$\overline{a}_i$
1	1	$a_i \oplus b_i$

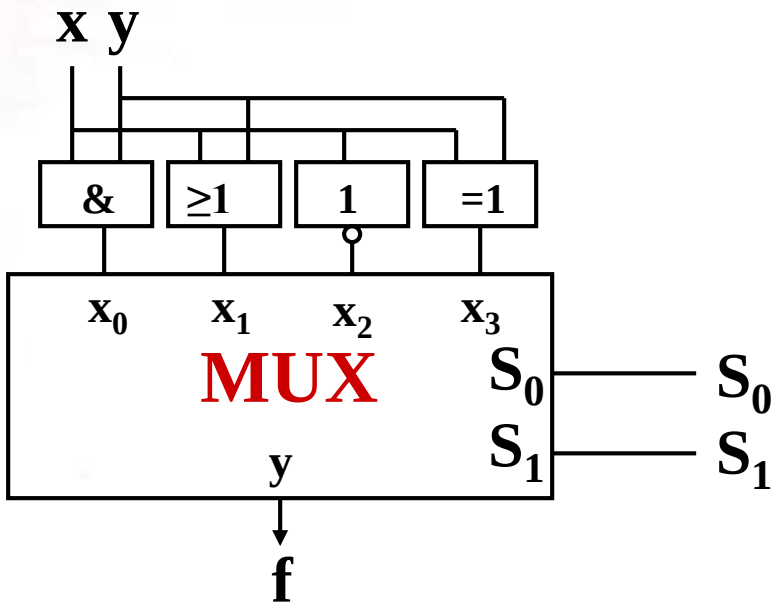
逻辑方程:

$$f=\overline{S}_1\overline{S}_0(xy)+\overline{S}_1S_0(x+y)+S_1\overline{S}_0(\overline{x})+S_1S_0(x \oplus y)$$

逻辑方程化简为:

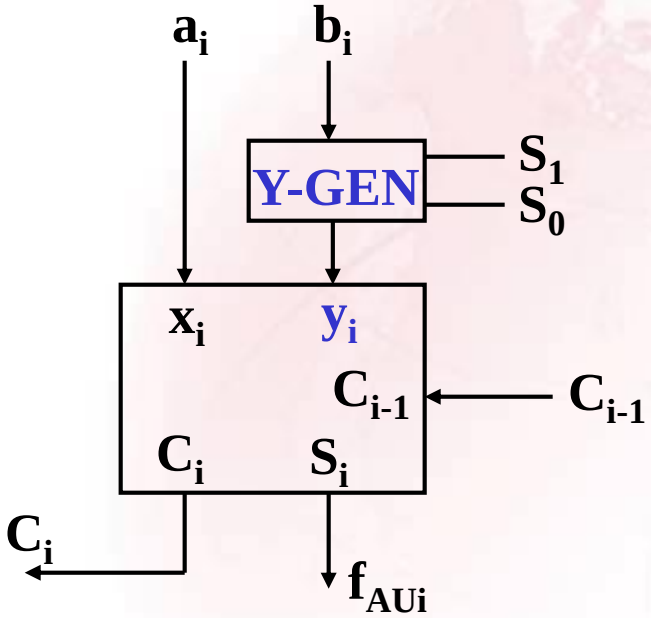
$$f=\overline{S}_1xy+S_0x\overline{y}+S_0\overline{x}y+S_1\overline{S}_0\overline{x}$$

根据化简的逻辑方程,可用逻辑门实现LU功能. 另外,也可直接根据功能表,用数据选择器实现。



(3) 算术单元

算术单元要进行加、减、加1、减1等四种运算，当采用补码运算时，可利用全加器实现。



根据ALU功能表，可利用下表求得y_i和C_{-1}的表达式。

y_i和C_{-1}取值表

功能	S_1	S_0	y_i	C_{-1}
加	0	0	b_i	0
减	0	1	$\overline{b}_i$	1
加1	1	0	0	1
减1	1	1	1	0

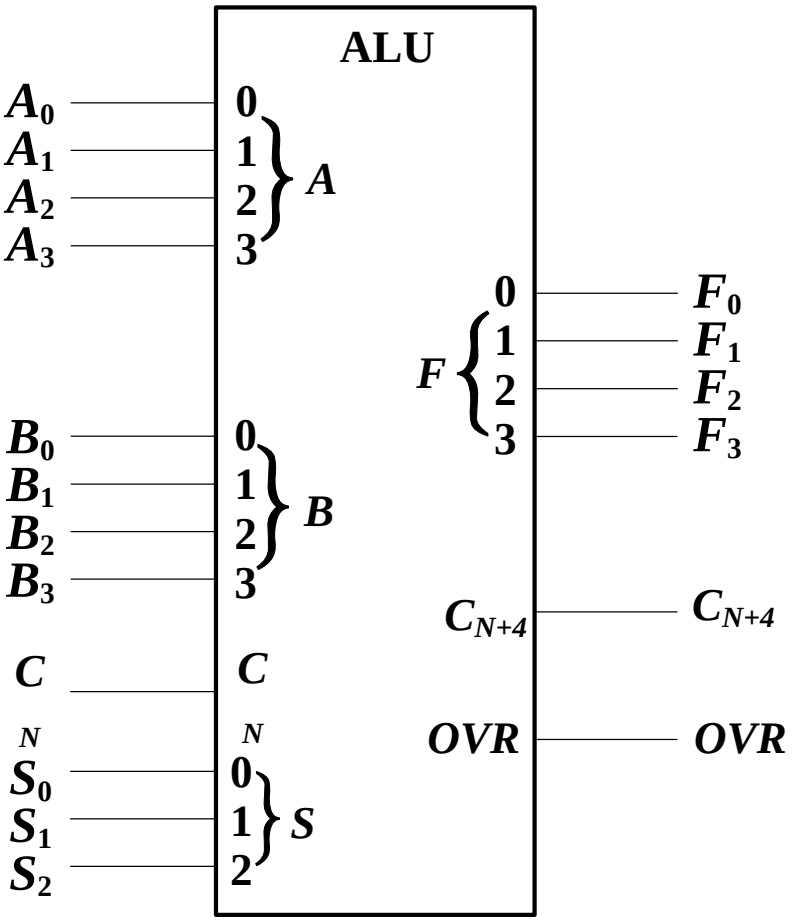
对Y-GEN,可写出y_i的表达式:

$$y_i=S_0 \oplus (\overline{S}_1 b_i)$$

对C-GEN可写出C_{-1}的表达式 ：$C_{-1}=S_1 \oplus S_0$

将上述运算求得的MUX、LU、AU电路连接，可得到一位ALU; 将多位ALU和C－GEN连接，可完成多位ALU电路设计。

ALU集成电路74LS/HC382



逻辑符号

S_2	S_1	S_0	操 作	说 明
0	0	0	清零	$F_3F_2F_1F_0=0000$
0	0	1	$F=B-A$	} 要求C_N=1
0	1	0	$F=A-B$	
0	1	1	$F=A+B$	要求C_N=0
1	0	0	$F=A\oplus B$	异或
1	0	1	$F=A+B$	或
1	1	0	$F=A\cdot B$	与
1	1	1	预置	$F_3F_2F_1F_0=1111$

功能表