

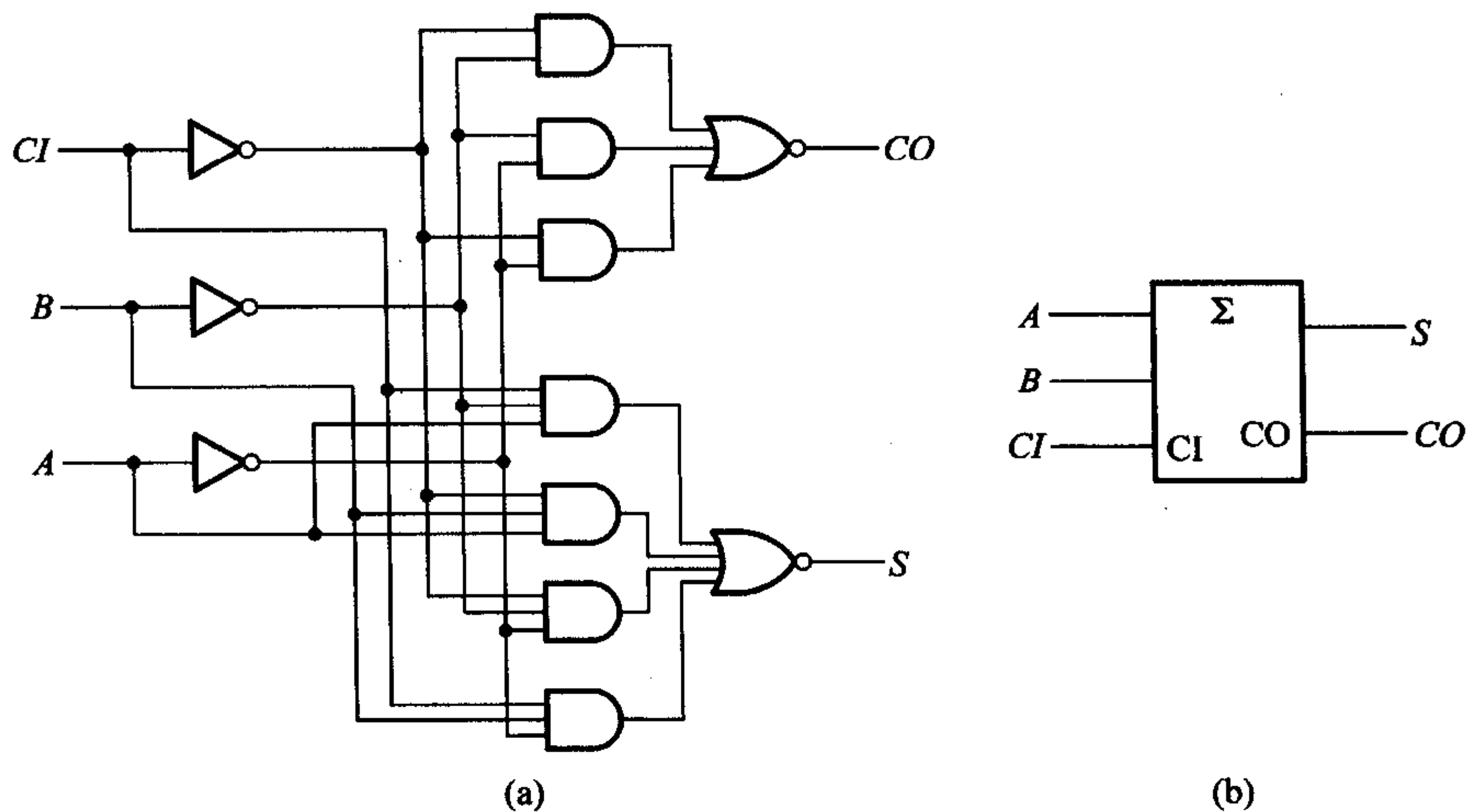


图 4.3.27 双全加器 74LS183

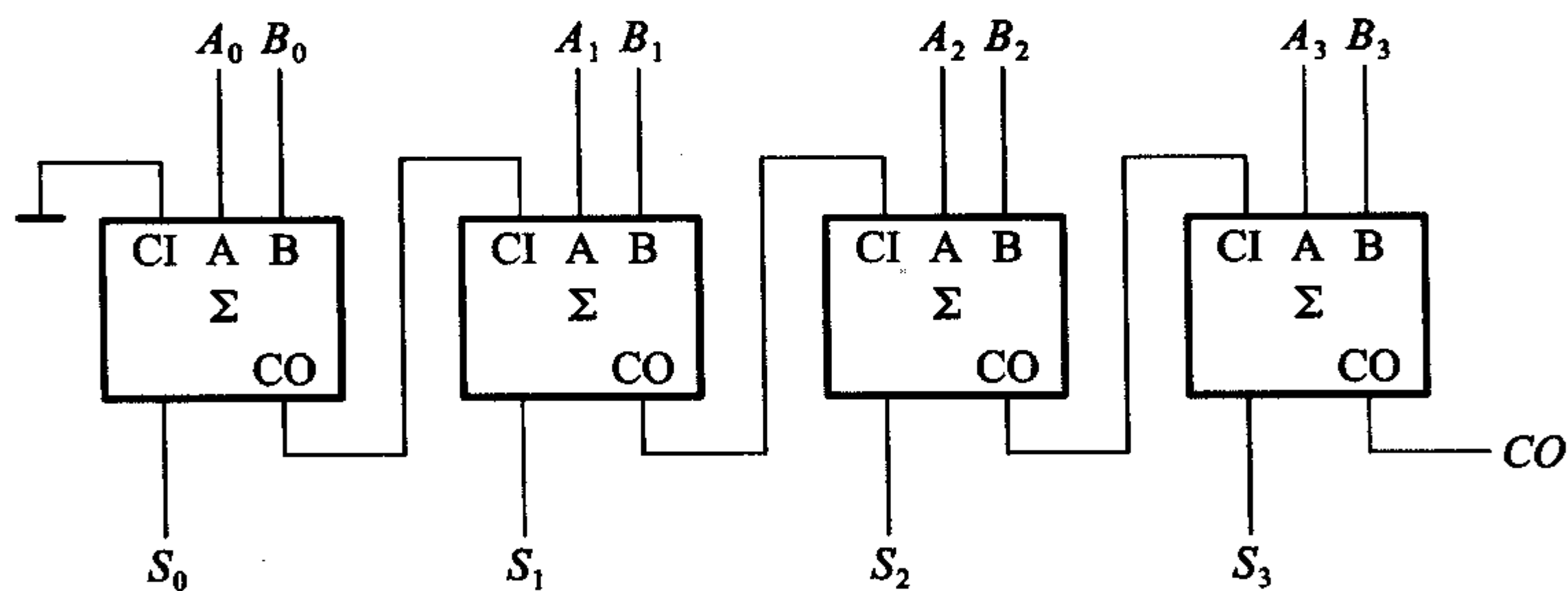
(a) $\frac{1}{2}$ 逻辑图 (b) 图形符号

图 4.3.28 4 位串行进位加法器

所需要的时间)才能得到稳定可靠的运算结果。但考虑到串行进位加法器的电路结构比较简单,因而在对运算速度要求不高的设备中,这种加法器仍不失为一种可取的电路。

2. 超前进位加法器

为了提高运算速度,必须设法减小由于进位信号逐级传递所耗费的时间。那么高位的进位输入信号能否在相加运算开始时就知道呢?

我们知道,加到第 i 位的进位输入信号是这两个加数第 i 位以下各位状态的函数,所以第 i 位的进位输入信号 $(CI)_i$ 一定能由 $A_{i-1}A_{i-2}\cdots A_0$ 和 $B_{i-1}B_{i-2}\cdots B_0$ 唯一地确定。根据这个原理,就可以通过逻辑电路事先得出每一位全加器的进

位输入信号,而无需再从最低位开始向高位逐位传递进位信号了,这就有效地提高了运算速度。采用这种结构形式的加法器称为超前进位(Carry Look-ahead)加法器,也称为快速进位(Fast Carry)加法器。

下面具体分析一下这些超前进位信号的产生原理。从表 4.3.9 所示的全加器的真值表中可以看到,在两种情况下会有进位输出信号产生。第一种情况是 $AB=1$,这时 $(CO)=1$ 。第二种情况是 $A+B=1$ 且 $(CI)=1$,也产生 $(CO)=1$ 的信号,这时可以把来自低位的进位输入信号 (CI) 直接传送到进位输出端 (CO) 。事实上在 $AB=1$ 时同样也可以将 CI 直接传送到输出端。于是两个多位数中第 i 位相加产生的进位输出 $(CO)_i$ 可表示为

$$(CO)_i = A_i B_i + (A_i + B_i)(CI)_i \quad (4.3.26)$$

若将 $A_i B_i$ 定义为进位生成函数 G_i ,同时将 $(A_i + B_i)$ 定义为进位传送函数 P_i ,则式(4.3.26)可改写为

$$(CO)_i = G_i + P_i(CI)_i \quad (4.3.27)$$

将上式展开后得到

$$\begin{aligned} (CO)_i &= G_i + P_i(CI)_i \\ &= G_i + P_i[G_{i-1} + P_{i-1}(CI)_{i-1}] \\ &= G_i + P_i G_{i-1} + P_i P_{i-1}[G_{i-2} + P_{i-2}(CI)_{i-2}] \\ &\vdots \\ &= G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + \cdots + P_i P_{i-1} \cdots P_1 G_0 \\ &\quad + P_i P_{i-1} \cdots P_0 (CI)_0 \end{aligned} \quad (4.3.28)$$

从全加器的真值表(表 4.3.9)写出第 i 位和 S_i 的逻辑式得到

$$S_i = A_i B_i' (CI)_i' + A_i' B_i (CI)_i' + A_i B_i' (CI)_i + A_i' B_i (CI)_i \quad (4.3.29)$$

有时也将上式变换为异或函数

$$\begin{aligned} S_i &= (A_i B_i' + A_i' B_i)(CI)_i' + (A_i B_i + A_i' B_i')(CI)_i \\ &= (A_i \oplus B_i)(CI)_i' + (A_i \oplus B_i)'(CI)_i \\ &= A_i \oplus B_i \oplus (CI)_i \end{aligned} \quad (4.3.30)$$

根据式(4.3.28)和式(4.3.30)构成的 4 位超前进位加法器 74LS283 如图 4.3.29 所示。现以第 1 位($i=1$)为例,分析一下它的逻辑功能。门 G_{22} 的输出 X_1 、门 G_{23} 的输出 Y_1 及和 S_1 分别为

$$\begin{aligned} X_1 &= (A_1 B_1)'(A_1 + B_1) = A_1 \oplus B_1 \\ Y_1 &= ((A_0 + B_0)' + (CI)_0'(A_0 B_0)')' = A_0 B_0 + (A_0 + B_0)(CI)_0 \\ &= G_0 + P_0(CI)_0 = (CO)_0 = (CI)_1 \\ S_1 &= X_1 \oplus Y_1 = A_1 \oplus B_1 \oplus (CI)_1 \end{aligned}$$

可见, $(CO)_0$ 和 S_1 的结果与式(4.3.28)和式(4.3.30)完全相符。

从图 4.3.29 上还可以看出,从两个加数送到输入端到完成加法运算只需三

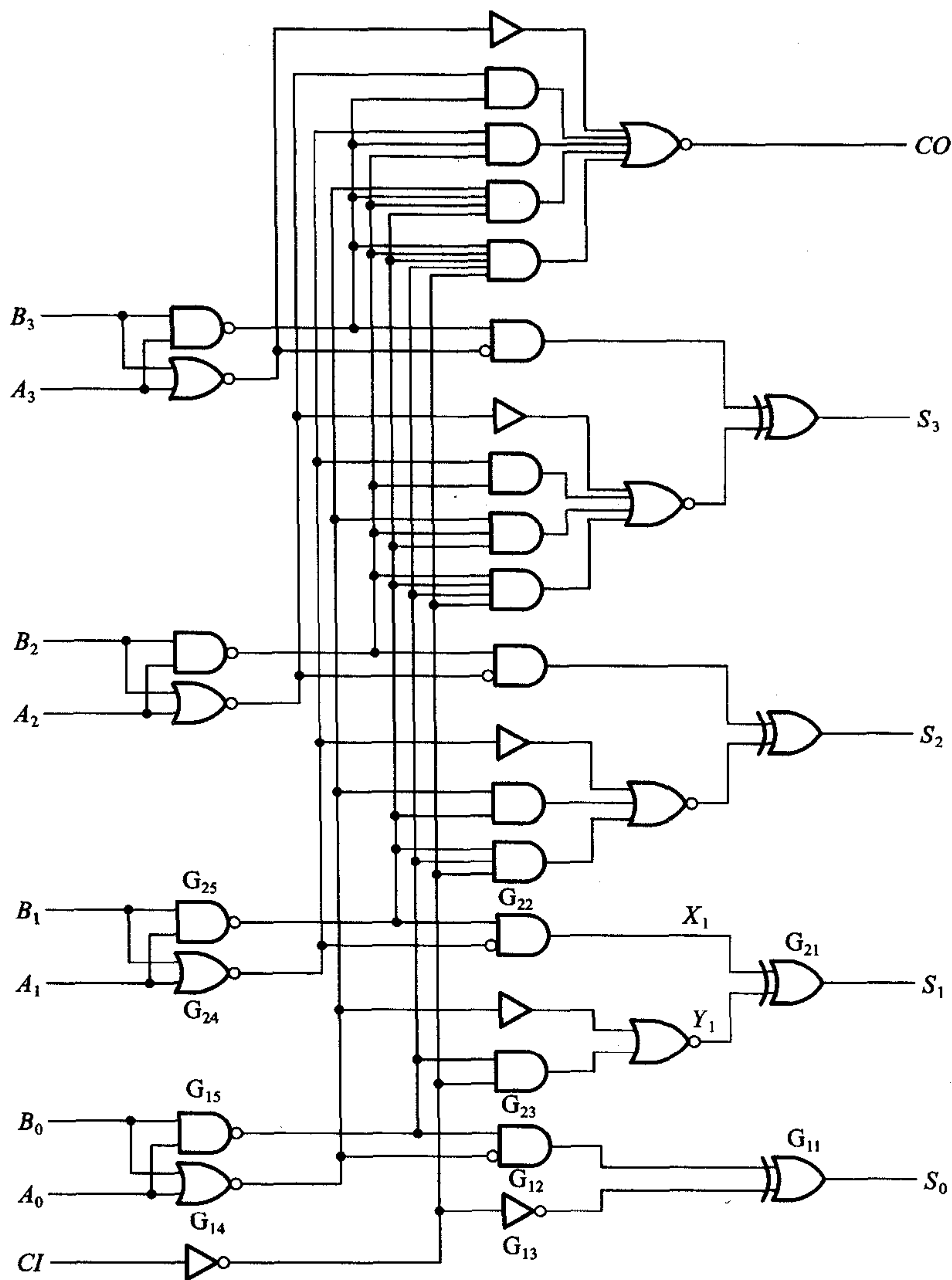


图 4.3.29 4 位超前进位加法器 74LS283

级门电路的传输延迟时间,而获得进位输出信号仅需一级反相器和一级与或非门的传输延迟时间。然而必须指出,运算时间得以缩短是用增加电路复杂程度的代价换取的。当加法器的位数增加时,电路的复杂程度也随之急剧上升。

三、用加法器设计组合逻辑电路

如果要产生的逻辑函数能化成输入变量与输入变量或者输入变量与常量在数值上相加的形式,这时用加法器来设计这个组合逻辑电路往往会非常简单。

【例 4.3.7】 设计一个代码转换电路,将十进制代码的 8421 码转换为余 3 码。

解: 以 8421 码为输入、余 3 码为输出,即可列出代码转换电路的逻辑真值表,如表 4.3.10 所示。

仔细观察一下表 4.3.10 不难发现, $Y_3Y_2Y_1Y_0$ 和 $DCBA$ 所代表的二进制数始终相差 0011,即十进制数的 3。故可得

$$Y_3Y_2Y_1Y_0 = DCBA + 0011 \quad (4.3.31)$$

其实这也正是余 3 代码的特征。根据式 (4.3.31),用一片 4 位加法器 74LS283 便可接成要求的代码转换电路,如图 4.3.30 所示。

表 4.3.10 例 4.3.7 的逻辑真值表

输 入				输 出			
D	C	B	A	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

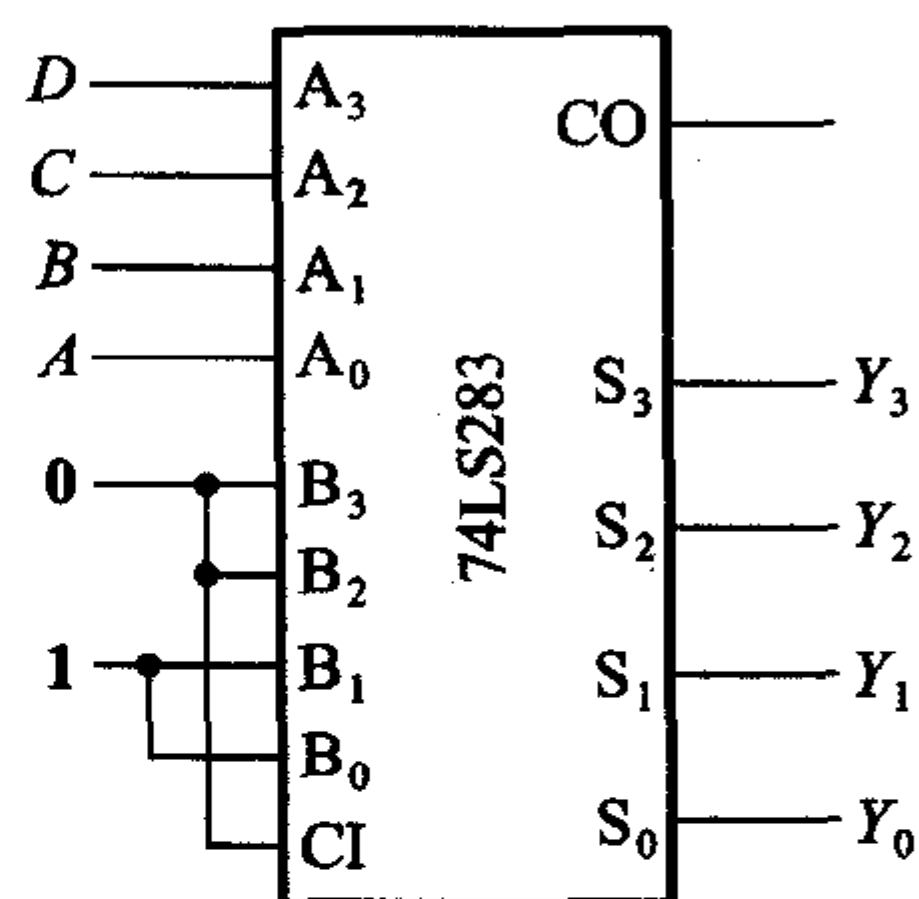


图 4.3.30 例 4.3.7 的代码转换电路

复习思考题

R4.3.6 串行进位加法器和超前进位加法器有何区别? 它们各有何优缺点?

4.3.5 数值比较器

在一些数字系统(例如数字计算机)当中经常要求比较两个数值的大小。为完成这一功能所设计的各种逻辑电路统称为数值比较器。

一、1 位数值比较器

首先讨论两个 1 位二进制数 A 和 B 相比较的情况。这时有三种可能:

(1) $A > B$ (即 $A = 1, B = 0$), 则 $AB' = 1$, 故可以用 AB' 作为 $A > B$ 的输出信号 $Y_{(A>B)}$ 。

(2) $A < B$ (即 $A = 0, B = 1$), 则 $A'B = 1$, 故可以用 $A'B$ 作为 $A < B$ 的输出信号 $Y_{(A < B)}$ 。

(3) $A = B$, 则 $A \odot B = 1$, 故可以用 $A \odot B$ 作为 $A = B$ 的输出信号 $Y_{(A = B)}$ 。

图 4.3.31 给出的是一种实用的 1 位数值比较器电路。

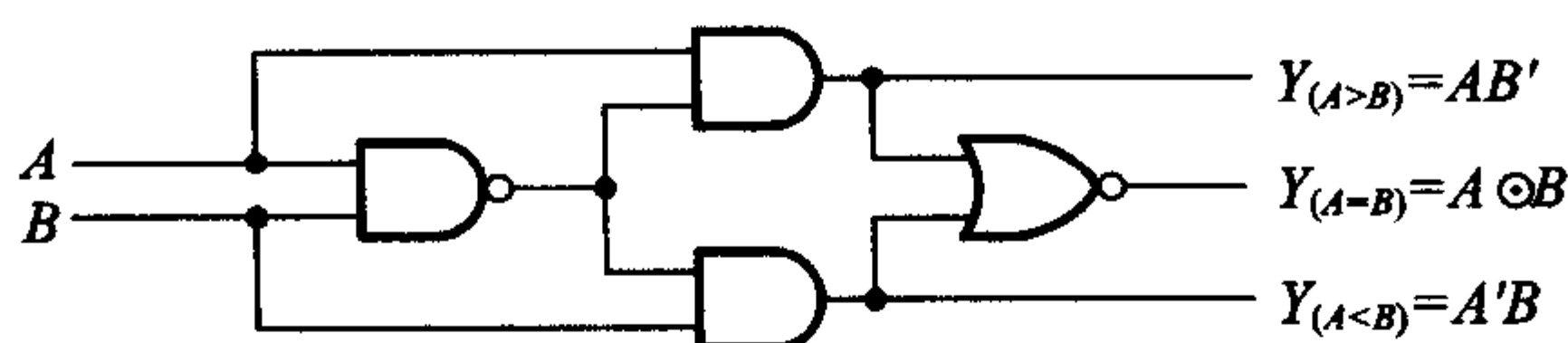


图 4.3.31 1 位数值比较器

二、多位数值比较器

在比较两个多位数的大小时, 必须自高而低地逐位比较, 而且只有在高位相等时, 才需要比较低位。

例如, A, B 是两个 4 位二进制数 $A_3A_2A_1A_0$ 和 $B_3B_2B_1B_0$, 进行比较时应首先比较 A_3 和 B_3 。如果 $A_3 > B_3$, 那么不管其他几位数码各为何值, 肯定是 $A > B$ 。反之, 若 $A_3 < B_3$, 则不管其他几位数码为何值, 肯定是 $A < B$ 。如果 $A_3 = B_3$, 这就必须通过比较下一位 A_2 和 B_2 来判断 A 和 B 的大小了。依此类推, 定能比出结果。

如果 A, B 是两个多位数的高 4 位数, 那么, 当 A, B 相等时, 就需要以低位的比较结果来决定两个数的大小了。根据上述原理, 我们就得到了表示 $A > B, A < B$ 和 $A = B$ 的逻辑函数式为

$$Y_{(A > B)} = A_3B'_3 + (A_3 \odot B_3)A_2B'_2 + (A_3 \odot B_3)(A_2 \odot B_2)A_1B'_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)A_0B'_0 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)I_{(A > B)} \quad (4.3.32)$$

$$Y_{(A < B)} = A'_3B_3 + (A_3 \odot B_3)A'_2B_2 + (A_3 \odot B_3)(A_2 \odot B_2)A'_1B_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)A'_0B_0 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)I_{(A < B)} \quad (4.3.33)$$

$$Y_{(A = B)} = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)I_{(A = B)} \quad (4.3.34)$$

$I_{(A > B)}, I_{(A < B)}$ 和 $I_{(A = B)}$ 是来自低位的比较结果。相比较的两数都只有 4 位, 没有来自低位的比较结果时, 应令 $I_{(A > B)} = I_{(A < B)} = 0, Y_{(A = B)} = 1$ 。由于 A 和 B 比较的结果只有 $A > B, A < B$ 和 $A = B$ 三种可能, 所以“不是 $A > B$ 或者 $A = B$, 就是 $A < B$ ”、“不是 $A < B$ 或者 $A = B$, 就是 $A > B$ ”, 因此又得到如下关系式

$$Y_{(A > B)} = (Y_{(A < B)} + Y_{(A = B)})' \quad (4.3.35)$$

$$Y_{(A < B)} = (Y_{(A > B)} + Y_{(A = B)})' \quad (4.3.36)$$

图 4.3.32 是 4 位数值比较器 74LS85 的逻辑图。这个电路就是按照式

(4.3.32) ~ (4.3.36) 接成的。利用 $I_{(A>B)}$ 、 $I_{(A<B)}$ 和 $I_{(A=B)}$ 这三个输入端, 可以将两片以上的 74LS85 组合成位数更多的数值比较器电路。

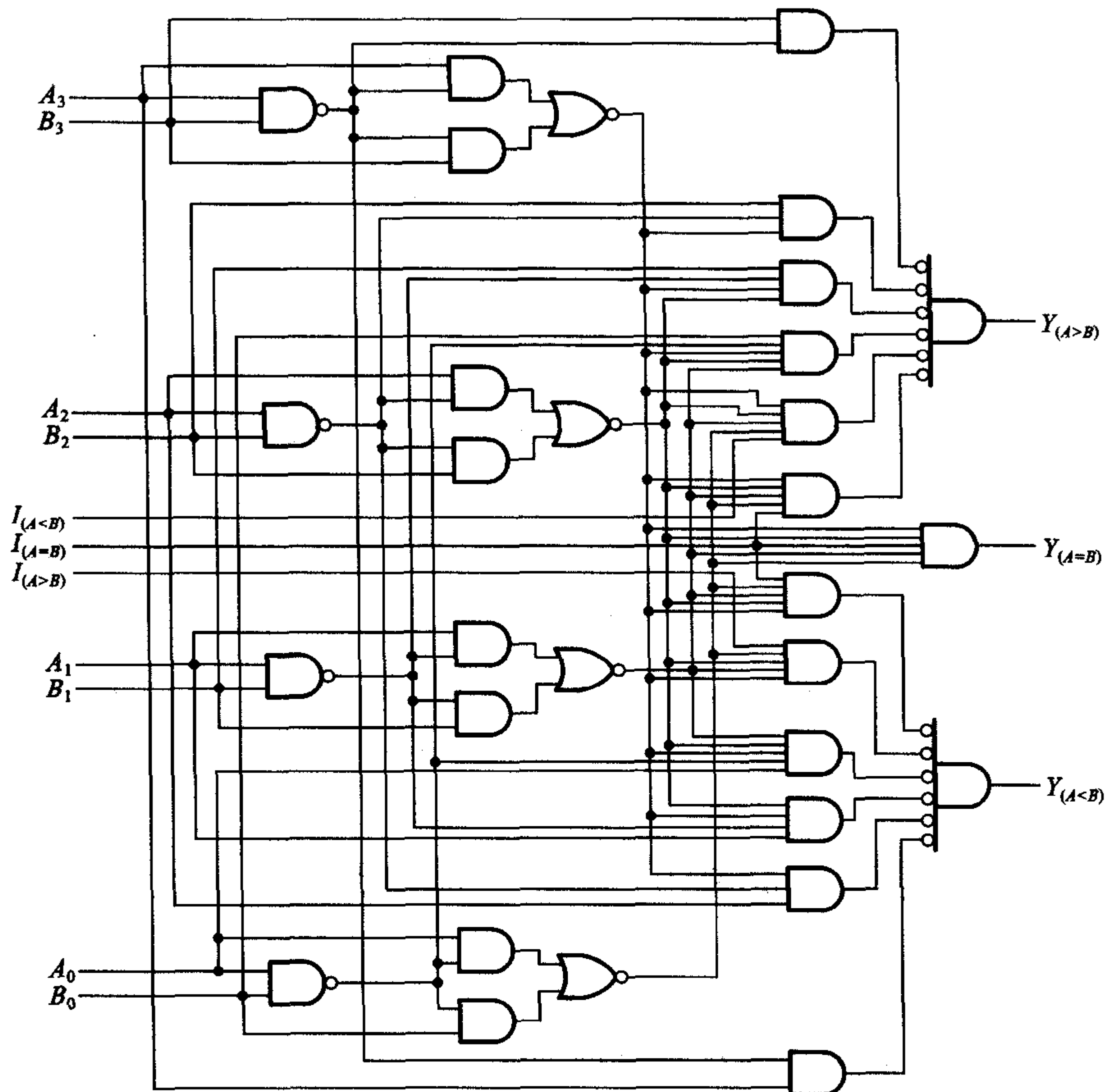


图 4.3.32 4 位数值比较器 74LS85

【例 4.3.8】 试用两片 74LS85 组成一个 8 位数值比较器。

解： 根据多位数比较的规则, 在高位相等时取决于低位的比较结果。因此只要将两个数的高 4 位 $C_7C_6C_5C_4$ 和 $D_7D_6D_5D_4$ 接到第(2)片 74LS85 上, 而将低 4 位 $C_3C_2C_1C_0$ 和 $D_3D_2D_1D_0$ 接到第(1)片 74LS85 上, 同时把第(1)片的 $Y_{(A>B)}$ 、 $Y_{(A<B)}$ 和 $Y_{(A=B)}$ 接到第(2)片 $I_{(A>B)}$ 、 $I_{(A<B)}$ 和 $I_{(A=B)}$ 就行了。

因为第(1)片 74LS85 没有来自低位的比较信号输入, 所以将它的 $I_{(A>B)}$ 和 $I_{(A<B)}$ 端接 0, 同时将它的 $I_{(A=B)}$ 端接 1。这样就得到了图 4.3.33 所示的 8 位数值比较电路。

目前生产的数值比较器产品中, 也有采用其他电路结构形式的。因为电路

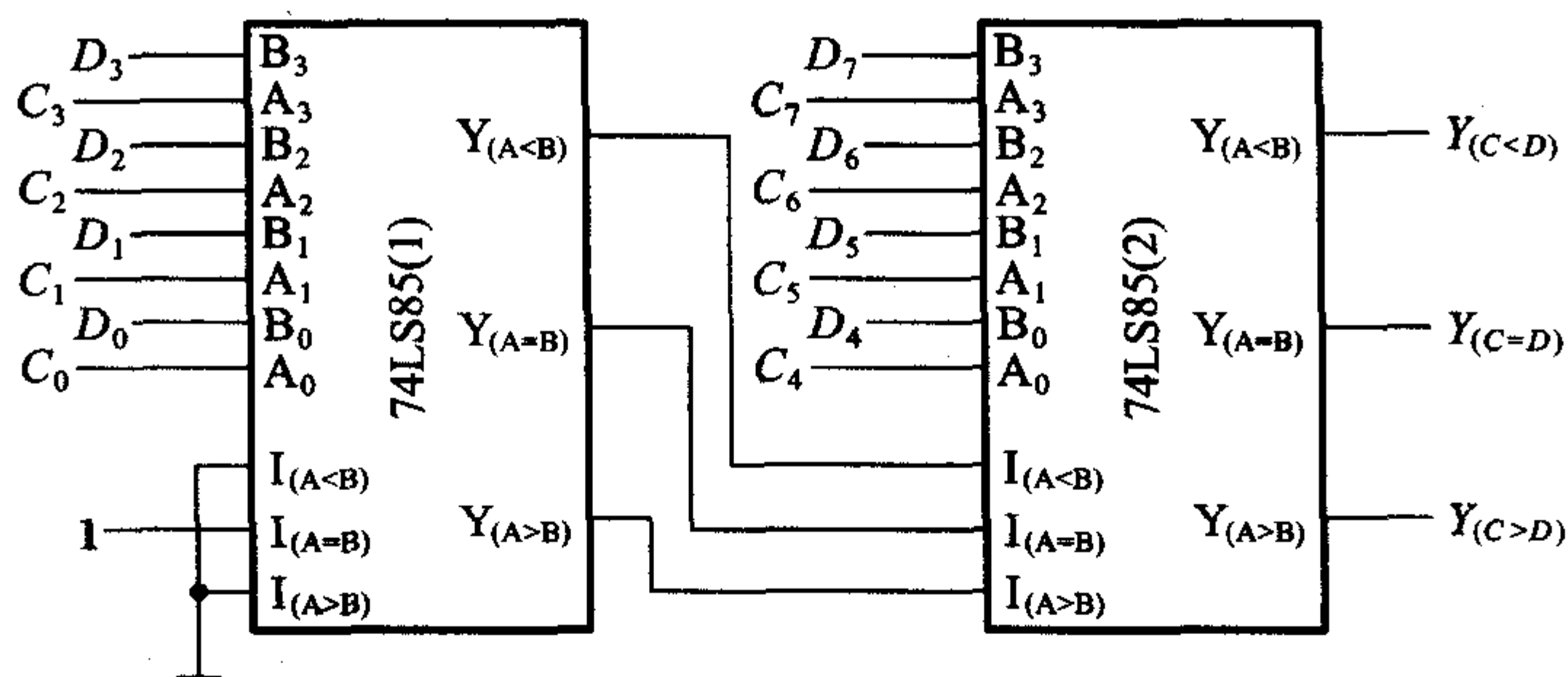


图 4.3.33 将两片 74LS85 接成 8 位数值比较器

结构不同,扩展输入端的用法也不完全一样,使用时应注意加以区别。

复习思考题

R4.3.7 如果用 4 位数值比较器比较两个 3 位的二进制数,可以有多少种接法?

4.4 组合逻辑电路中的竞争 - 冒险现象

4.4.1 竞争 - 冒险现象及其成因

在前面的章节里我们系统地讲述了组合逻辑电路的分析方法和设计方法。这些分析和设计都是在输入、输出处于稳定的逻辑电平下进行的。为了保证系统工作的可靠性,有必要再观察一下当输入信号逻辑电平发生变化的瞬间电路的工作情况。

首先让我们看两个最简单的例子。在图 4.4.1(a)所示的与门电路中,稳态下无论 $A=1$ 、 $B=0$ 还是 $A=0$ 、 $B=1$,输出皆为 $Y=0$ 。但是在输入信号 A 从 1 跳变为 0 时,如果 B 从 0 跳变为 1,而且 B 首先上升到 $V_{IL(max)}$ 以上,这样在极短的时间 Δt 内将出现 A 、 B 同时高于 $V_{IL(max)}$ 的状态,于是便在门电路的输出端产生了极窄的 $Y=1$ 的尖峰脉冲,或称为电压毛刺,如图中所示(在画波形时考虑了门电路的传输延迟时间)。显然,这个尖峰脉冲不符合门电路稳态下的逻辑功能,因而它是系统内部的一种噪声。

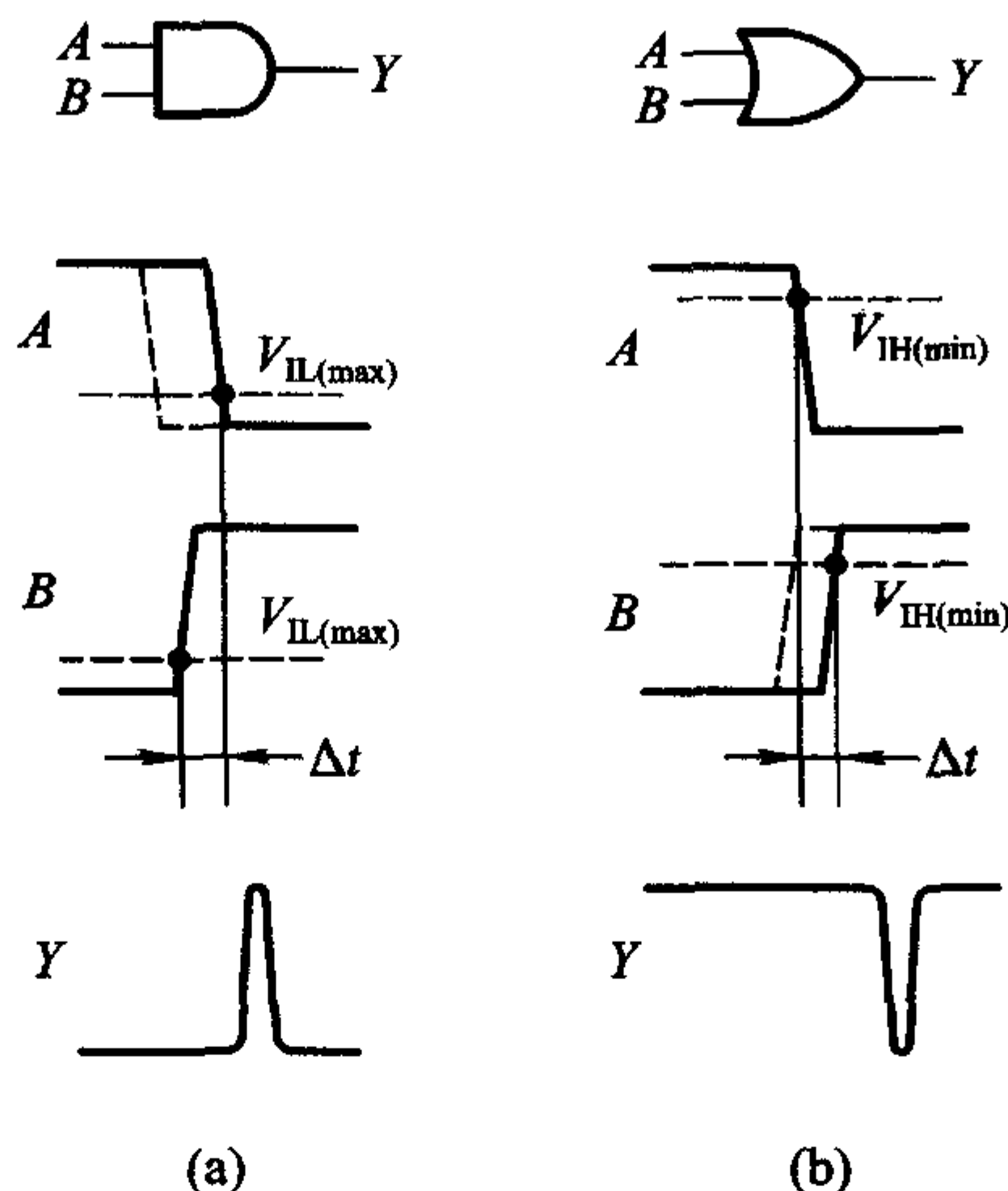


图 4.4.1 由于竞争而产生的尖峰脉冲

同样,在图 4.4.1(b)所示的或门电路中,稳态下无论 $A=0$ 、 $B=1$ 还是 $A=1$ 、 $B=0$,输出都应该是 $Y=1$ 。但如果 A 从 1 变成 0 的时刻和 B 从 0 变成 1 的时刻略有差异,而且在 A 下降到 $V_{IH(min)}$ 时 B 尚未上升到 $V_{IH(min)}$,则在暂短的 Δt 时间内将出现 A 、 B 同时低于 $V_{IH(min)}$ 的状态,使输出端产生极窄的 $Y=0$ 的尖峰脉冲。这个尖峰脉冲同样也是违背稳态下逻辑关系的噪声。

我们将门电路两个输入信号同时向相反的逻辑电平跳变(一个从 1 变为 0,另一个从 0 变为 1)的现象称为竞争。

应当指出,有竞争现象时不一定都会产生尖峰脉冲。例如,在图 4.4.1(a)所示的与门电路中,如果在 B 上升到 $V_{IL(max)}$ 之前 A 已经降到了 $V_{IL(max)}$ 以下(如图中虚线所示),这时输出端不会产生尖峰脉冲。同理,在图 4.4.1(b)所示的或门电路中,若 A 下降到 $V_{IH(min)}$ 以前 B 已经上升到 $V_{IH(min)}$ 以上(如图中虚线所示),输出端也不会有尖峰脉冲产生。

如果图 4.4.1 所示的与门和或门是复杂数字系统中的两个门电路,而且 A 、 B 又是经过不同的传输途径到达的,那么在设计时往往难于准确知道 A 、 B 到达次序的先后,以及它们在上升时间和下降时间上的细微差异。因此,我们只能说只要存在竞争现象,输出就有可能出现违背稳态下逻辑关系的尖峰脉冲。

由于竞争而在电路输出端可能产生尖峰脉冲的现象就称为竞争-冒险。

图 4.4.2 是一个 2 线-4 线译码器的电路和它的电压波形图。由图上可以看到,在 A 、 B 的稳定状态下输出 Y_0 和 Y_3 都应为 0 状态。然而由于门 G_4 和 G_5 的传输延迟时间不同,在 AB 从 10 跳变为 01 的过程中, Y_0 端有尖峰脉冲产生。此外,由于 A 、 B 在变化过程中到达 $V_{IL(max)}$ 的时刻不同, Y_3 端也有尖峰脉冲出现。

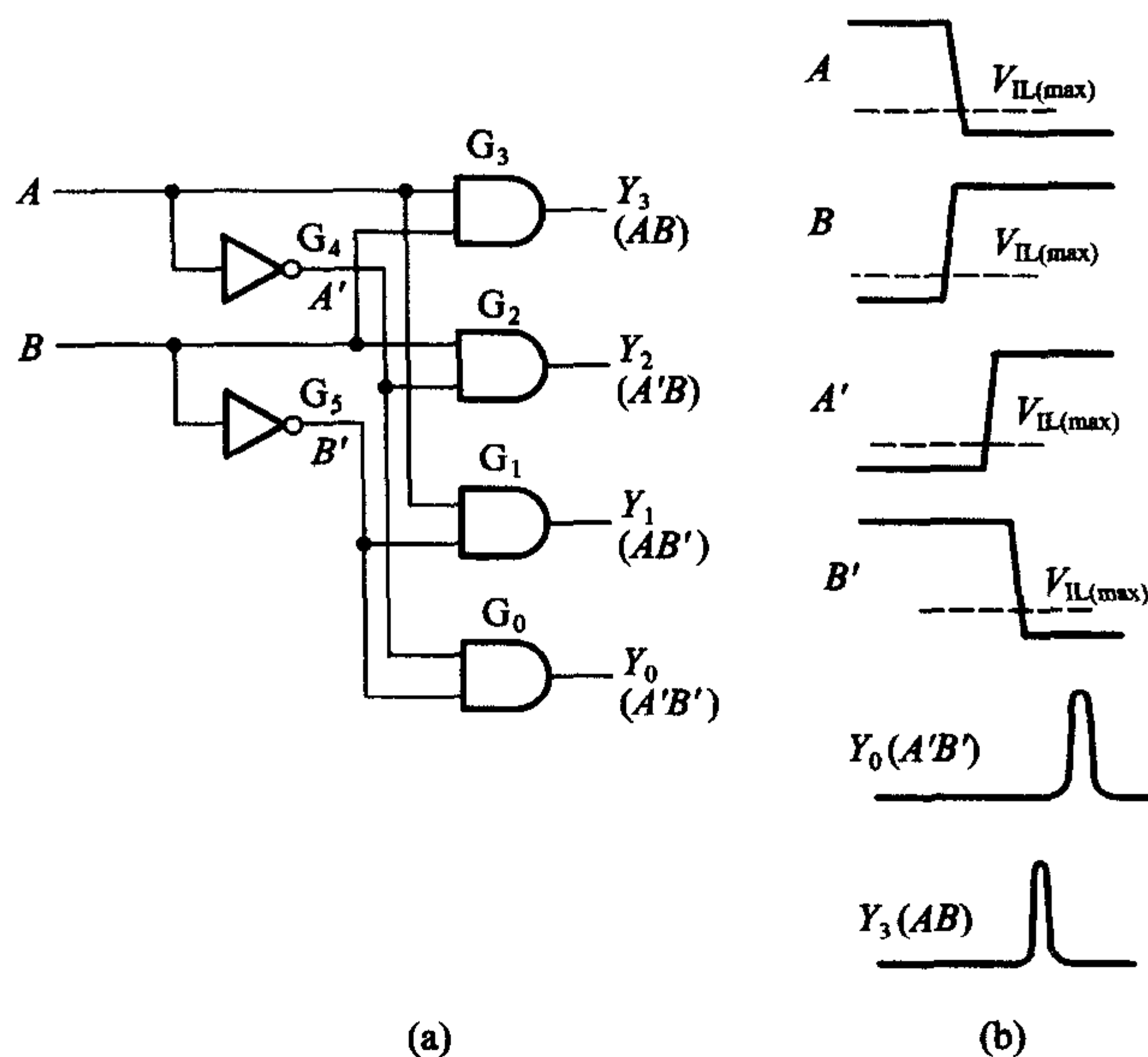


图 4.4.2 2 线 - 4 线译码器中的竞争 - 冒险现象

(a) 电路图 (b) 电压波形图

倘若译码器的负载是一个对尖峰脉冲敏感的电路(例如下一章将要讲到的触发器),那么这种尖峰脉冲将可能使负载电路发生误动作。对此应在设计时采取措施加以避免。

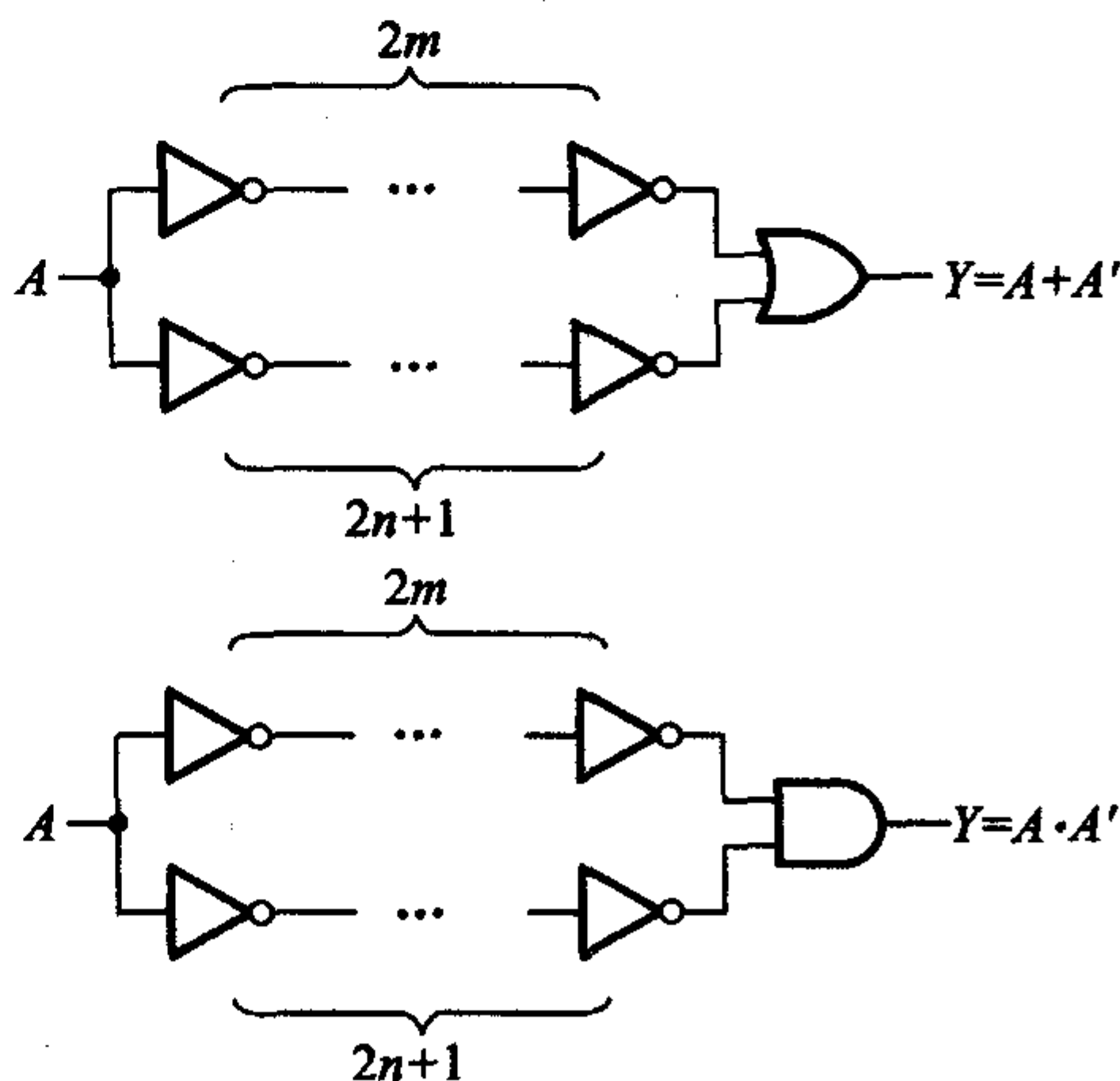
* 4.4.2 检查竞争 - 冒险现象的方法

在输入变量每次只有一个改变状态的简单情况下,可以通过逻辑函数式判断组合逻辑电路中是否有竞争 - 冒险现象存在。

如果输出端门电路的两个输入信号 A 和 A' 是输入变量 A 经过两个不同的传输途径而来的(如图 4.4.3 所示),那么当输入变量 A 的状态发生突变时输出端便有可能产生尖峰脉冲。因此,只要输出端的逻辑函数在一定条件下能简化成

$$Y = A + A' \quad \text{或} \quad Y = A \cdot A'$$

则可判定存在竞争 - 冒险现象。

图 4.4.3 同一输入变量经不同途径到达输出端的情况(m, n 均为正整数)

如果图 4.4.3 所示电路的输出端是或非门、与非门,同样也存在竞争-冒险现象。这时的输出应能写成 $Y = (A + A')'$ 或者 $Y = (A \cdot A)'$ 的形式。

【例 4.4.1】 试判断图 4.4.4 中的两个电路中是否存在竞争-冒险现象。已知任何瞬间输入变量只可能有一个改变状态。

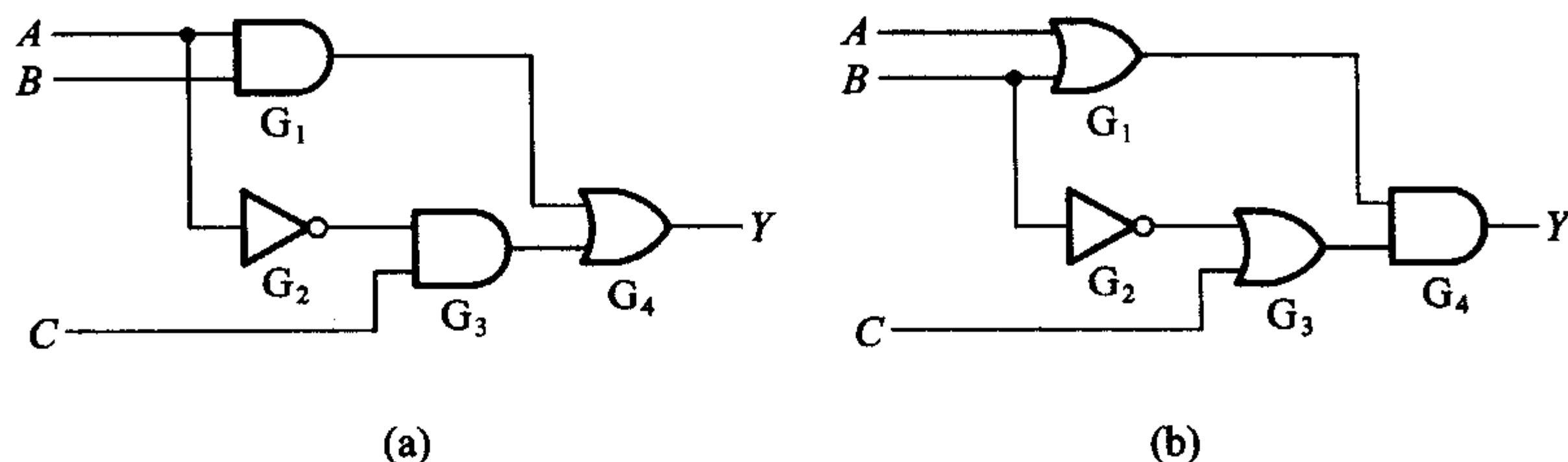


图 4.4.4 例 4.4.1 的电路

解: 图 4.4.4(a) 电路输出的逻辑函数式可写为

$$Y = AB + A'B$$

当 $B = C = 1$ 时, 上式将成为

$$Y = A + A'$$

故图 4.4.4(a) 电路中存在竞争-冒险现象。

图 4.4.4(b) 电路的输出为

$$Y = (A + B) \cdot (B' + C)$$

在 $A = C = 0$ 的条件下, 上式简化为

$$Y = B \cdot B'$$

故图 4.4.4(b) 电路中也存在竞争-冒险现象。

这种方法虽然简单, 但局限性太大, 因为多数情况下输入变量都有两个以上同时改变状态的可能性。如果输入变量的数目又很多, 就更难于从逻辑函数式上简单地找出所有产生竞争-冒险现象的情况了。

将计算机辅助分析的手段用于分析数字电路以后, 为我们从原理上检查复杂数字电路的竞争-冒险现象提供了有效的手段。通过在计算机上运行数字电路的模拟程序, 能够迅速查出电路是否会存在竞争-冒险现象。目前已有这类成熟的程序可供选用。

另一种方法是用实验来检查电路的输出端是否有因为竞争-冒险现象而产生的尖峰脉冲。这时加到输入端的信号波形应该包含输入变量的所有可能发生的状态变化。