

第 3 章 组合逻辑电路

数字电路分成组合逻辑电路和时序逻辑电路两类。组合逻辑电路比时序逻辑电路简单得多。本章首先讨论组合逻辑电路的分析和设计方法,并给出用门电路、门阵列实现组合逻辑电路的实例;然后介绍常用的中规模组合逻辑电路功能模块的基本原理和逻辑器件,并对组合逻辑电路存在的竞争与险象进行简单分析;最后介绍组合逻辑电路的 VHDL 描述方法。

3.1 组合逻辑电路概述

组合逻辑电路不仅能够独立地实现各种复杂的逻辑功能,而且还是各种时序逻辑电路的重要组成部分,因此组合逻辑电路是各种逻辑电路的基本电路单元。

3.1.1 组合逻辑电路模型及特点

1. 组合逻辑电路模型

图 3-1 给出了组合逻辑电路的基本模型,由以下 3 部分组成。

- (1) 输入信号: $x_1, x_2, \dots, x_n$; n 个输入变量,可以有 2^n 种输入。
- (2) 输出信号: $Z_1, Z_2, \dots, Z_m$ 。
- (3) 输出函数: $Z_i = f_i(x_1, x_2, \dots, x_n), i = 1, \dots, m$ 。

从组合逻辑电路的模型可以看出,组合逻辑电路是布尔函数的直接电路实现。

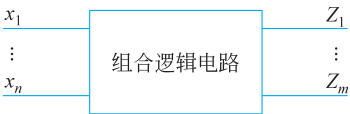


图 3-1 组合逻辑电路基本模型图

2. 组合逻辑电路的特点

组合逻辑电路的模型决定了组合逻辑电路的特点,电路在任意时刻的稳定输出,仅仅取决于当前的输入组合,而与过去的输入无关,即布尔函数的真值表直接决定了组合逻辑电路的逻辑关系。

3.1.2 组合逻辑电路的描述方法

由组合逻辑电路模型可知,组合逻辑电路完全可以由布尔函数表示,因此布尔函数的表示方法均可用于描述组合逻辑电路,如前面章节已介绍的真值表、布尔函数表达式、原理图、HDL 都可以用于描述组合逻辑电路。在此介绍另一种数字电路描述方法——时序图描述法。

时序图(timing diagram)是数字电路的输入信号和输出信号之间关系的图形表示方式。时序图一般采用电路实验或者逻辑仿真的方法得到。在一段时间里,将一组输入序列信号接入到电路的输入端,在示波器或者逻辑分析仪的屏幕上或者逻辑仿真程序的输出中,即可观察到与输入信号相对应的输出信号,输出信号能够完全反映电路的逻辑关系,时序图和真值表之间可以直接转换。

通过分析时序图,能够得到电路所实现的逻辑函数的真值表;反之,利用布尔函数的真值表也可以绘制电路工作的时序图。一个完整的组合逻辑电路时序图,应该完整描述输入信号在各种取值组合情况下对应的电路输出波形。

例如,函数 $F(A, B, C) = AB + BC$ 的真值表如表 3-1 所示。

表 3-1 $F=AB+BC$ 的真值表

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

描述其逻辑功能的典型时序图如图 3-2(a)所示。图 3-2 中给出了信号 A、B、C 的所有取值组合对应的输入的波形,然后给出了对应的电路输出 F 的波形。可以看到,当 $AB=11$ 、 $BC=11$ 时,函数值取值为 1;其他情况下,函数值取值为 0,与真值表描述完全一致。

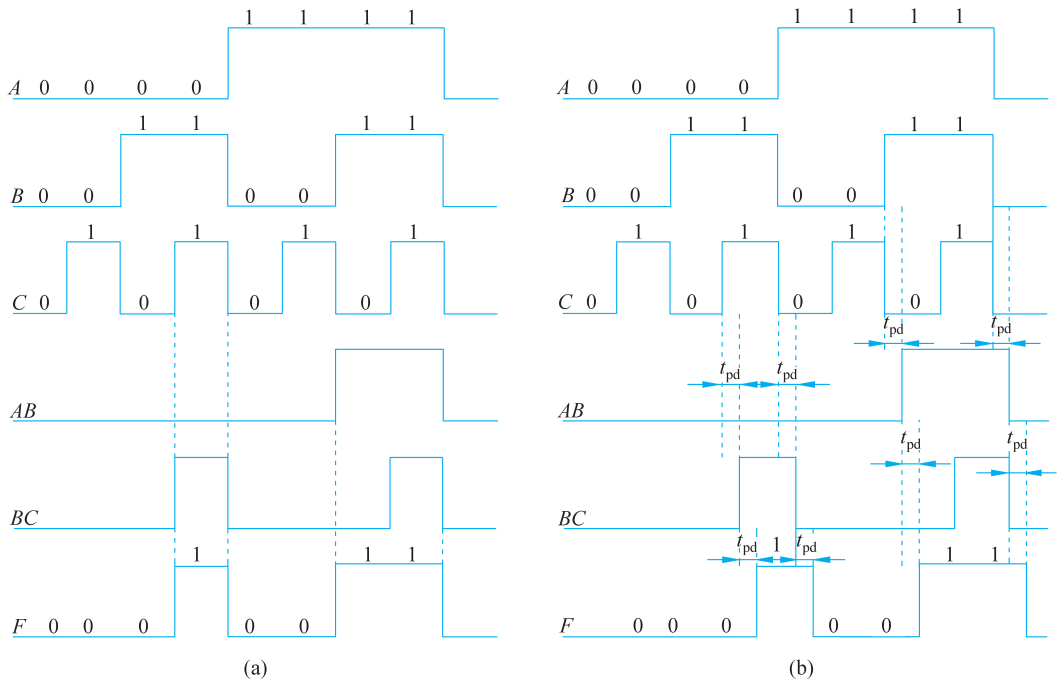


图 3-2 $F=AB+BC$ 的时序图

(a) 不考虑门延迟的时序图; (b) 考虑门延迟的时序图

如果在时序图中不仅考虑电路的逻辑功能,而且考虑各个门电路的延迟,这样的时序图更接近于实际电路的工作过程。例如布尔函数 $F(A,B,C)=AB+BC$,如果用与门和或门实现,并且假设与门和或门的门延迟均为 t_{pd} ,则考虑门延迟的时序图如图 3-2(b)所示。图中不仅考虑了输入和输出的逻辑关系,而且考虑了实际电路的门延迟。

3.2 组合逻辑电路设计

组合逻辑电路设计的基本任务是按照电路的功能描述,设计组合逻辑电路。组合逻辑电路的设计过程主要包括两个环节,首先根据电路的功能描述建立其逻辑函数模型,然后选择合适的逻辑器件实现组合逻辑电路。

3.2.1 组合逻辑电路设计过程

1. 建立组合逻辑电路模型

电路模型的建立是电路设计最关键的环节,电路模型建立好之后,后面的步骤才会有章可循。建立组合逻辑电路模型主要包括以下工作。

- (1) 分析功能描述,确定输入输出信号。
- (2) 建立真值表,写出布尔函数表达式。

在实际的电路设计中,功能描述是多种多样的,可以是直接的真值表、函数表达式,也可以是时序图、表格、文字等。无论什么样的描述方法,关键是要理解功能描述表达的意思,建立正确的电路模型,才能够设计出符合功能描述的电路。

2. 用集成逻辑器件实现组合逻辑电路

(1) 用门电路实现组合逻辑电路。用门电路实现组合逻辑电路,首先要把函数表达式化简为最简形式,然后把表达式变换为所选用门电路器件对应的表达式形式,最后用门电路画出电路原理图。

(2) 用基于标准积项的门阵列实现组合逻辑电路。用基于标准积项的门阵列实现组合逻辑电路,首先需要写出函数的标准积之和形式,然后再画出与门阵列固定、或门阵列可编程的门阵列电路原理图。

(3) 用基于最简积项的门阵列实现组合逻辑电路。用基于最简积项的门阵列实现组合逻辑电路,首先要把函数表达式化简为最简与或式,然后再画出与门阵列可编程、或门阵列也可编程的门阵列电路原理图。

(4) 用 HDL 描述组合逻辑电路。当使用 CPLD/FPGA 器件实现组合逻辑电路时,需要借助 HDL 编程,用 HDL 描述组合逻辑电路。

3.2.2 组合逻辑电路设计举例

1. 用门电路实现组合逻辑电路

例 3-1 设计能够实现如图 3-3 所示时序图的逻辑电路,其中波形 A、B、C 为输入信号,波形 F 为输出信号。要求用与非门、或非门设计此逻辑电路,并比较电路的复杂性。

解:

- (1) 由题意可知,电路的输入为 A、B、C,输出为 F。
- (2) 根据时序图的描述,可以建立电路真值表,如表 3-2 所示。

按照真值表,可以写出函数表达式 $F(A, B, C) = \sum m(5, 6, 7)$ 。

(3) 卡诺图化简布尔函数,如图 3-4 所示。由于题目要求用与非、或非门实现布尔函数,因此直接给出了原函数和反函数的卡诺图化简过程,以便于得到函数的最简与或式和最简或与式。

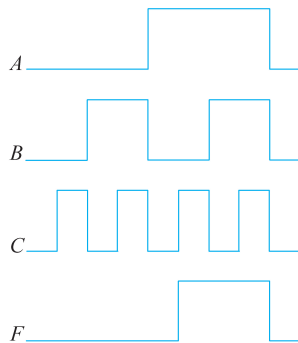


图 3-3 例 3-1 的时序图

表 3-2 例 3-1 真值表

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

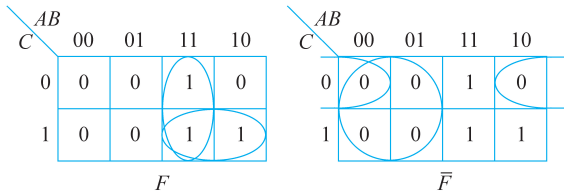


图 3-4 例 3-1 原函数与反函数的卡诺图

根据卡诺图,可以得到原函数和反函数的最简与或表达式。

$$F = AB + AC; \quad \bar{F} = \bar{A} + \bar{B}\bar{C}$$

对反函数取非,利用德·摩根定理容易得到函数的或与表达式: $F = A \cdot (B + C)$ 。

将与或表达式和或与表达式分别两次取非,再利用德·摩根定理,即可得到函数的与非-与非表达式和或非-或非表达式: $F = \overline{\overline{AB} \cdot \overline{AC}}, F = \overline{\overline{A} + \overline{(B + C)}}$ 。

图 3-5 为与非门及或非门实现的逻辑电路。

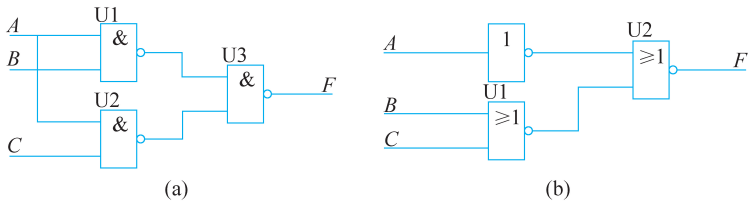


图 3-5 例 3-1 用与非门、或非门实现的原理图

(a) 与非门实现; (b) 或非门实现

比较发现,用与非门和或非门实现该电路需要的逻辑门数量和电路复杂度均相同。

例 3-2 为某水坝设计一个水位报警控制器,设水位高度用 4 位二进制数表示,当水位上升到 8m 时,白色指示灯开始亮;当水位上升到 10m 时,黄色指示灯开始亮;当水位上升到 12m 时,红色指示灯开始亮,其他颜色的灯灭。若水位不可能上升到 14m,要求用与非门、或非门设计此逻辑电路,并比较电路的复杂性。

解:

(1) 分析题意,首先建立电路的外部模型。

输入: 水位高度,0~13m(水位不可能上升到 14m),用 4 位二进制数表示,设为 $B_4B_3B_2B_1$ 。

输出：3 个指示灯，定义灯亮为 1，灯灭为 0，白色指示灯设为 F_w ，黄色指示灯设为 F_y ，红色指示灯设为 F_r 。

电路的外部模型如图 3-6 所示。

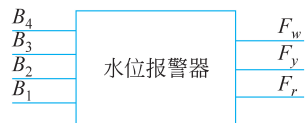


图 3-6 例 3-2 的电路模型

(2) 按照功能描述，建立电路真值表，如表 3-3 所示。由于水位不可能达到 14m，因此输入为 14m、15m 时，3 个指示灯输出为任意值(用“×”表示)。

表 3-3 例 3-2 的真值表

B_4	B_3	B_2	B_1	F_w	F_y	F_r
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	1	0
1	0	1	1	1	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	×	×	×
1	1	1	1	×	×	×

根据真值表，可写出如下函数表达式：

$$F_w = \sum m(8,9,10,11) + \sum d(14,15);$$

$$F_y = \sum m(10,11) + \sum d(14,15);$$

$$F_r = \sum m(12,13) + \sum d(14,15)$$

(3) 用与非门实现。将函数化简为最简与或表达式，原函数卡诺图如图 3-7 所示，由此可得 $F_w = B_4 \overline{B_3}$ ； $F_y = B_4 B_2$ ； $F_r = B_4 B_3$ 。

变换为与非-与非表达式为 $F_w = \overline{\overline{B_4} \overline{\overline{B_3}}}$ ； $F_y = \overline{\overline{B_4} \overline{\overline{B_2}}}$ ； $F_r = \overline{\overline{B_4} \overline{\overline{B_3}}}$ 。

与非门实现的电路如图 3-8 所示。

(4) 用或非门实现。将函数化简为最简或与式，首先要获得最简反函数，其卡诺图如图 3-9 所示。

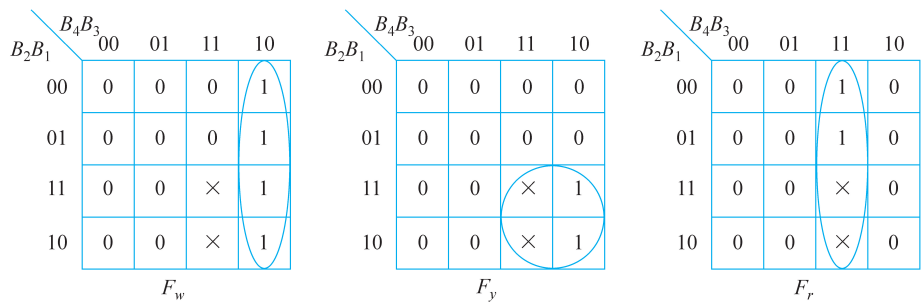


图 3-7 例 3-2 的卡诺图

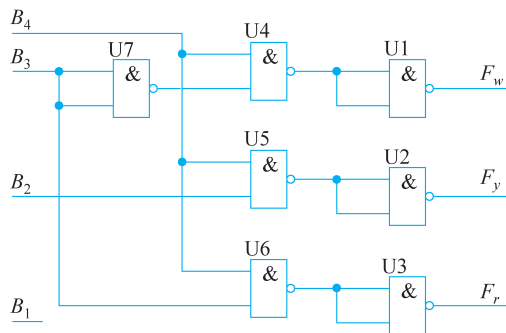


图 3-8 例 3-2 用与非门实现的原理图

由此可得 $\overline{F_w} = \overline{B_4} + B_3$; $\overline{F_y} = \overline{B_4} + \overline{B_2}$; $\overline{F_r} = \overline{B_4} + B_3$ 。

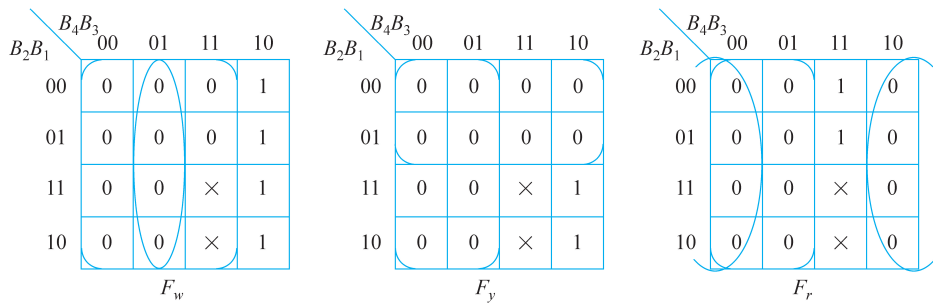


图 3-9 例 3-2 中原函数的卡诺图

将其变换为或非-或非表达式 $F_w = \overline{\overline{B_4} + B_3}$; $F_y = \overline{\overline{B_4} + \overline{B_2}}$; $F_r = \overline{\overline{B_4} + B_3}$ 。

或非门实现的电路如图 3-10 所示。

(5) 比较复杂性。比较两个电路可以看出,用或非门实现比用与非门实现减少了一个门,相对比较简单。

2. 用 SPLD 实现组合逻辑电路

例 3-3 设计 8421BCD 编码转换为七段数码管编码的码制转换电路,并用基于标准积项的门阵列、基于最简积项的门阵列实现该电路。

分段式半导体数码管是一种数码显示器件,是将多个发光二极管封装在一个器件内,组成特定形状。八段半导体数码管符号及内部原理结构如图 3-11 所示,其内部由 8 个发光二极管组成,其中 7 段

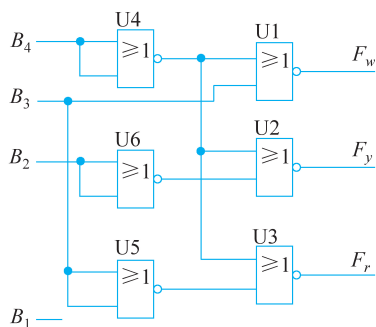


图 3-10 例 3-2 用或非门实现的原理图

为数码段 $a \sim g$, 1 段为小数点 DP, 其电平高、低的不同组合就构成了不同数码的显示。根据电路的结构, 可以分为共阳极和共阴极数码管。

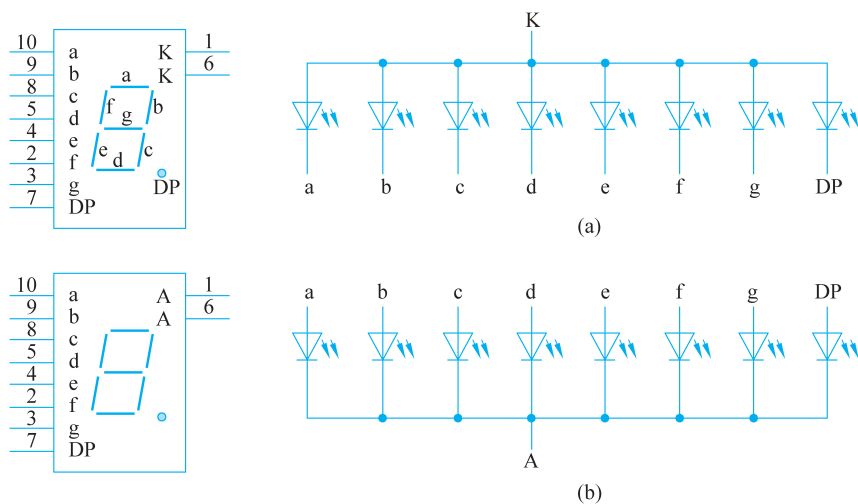


图 3-11 LED 数码管

(a) 共阳极; (b) 共阴极

半导体数码管目前主要用于电子计算器和数字仪表的显示部分中。七段数码管的 $a \sim g$ 分段及数字字符组成如图 3-12 所示。

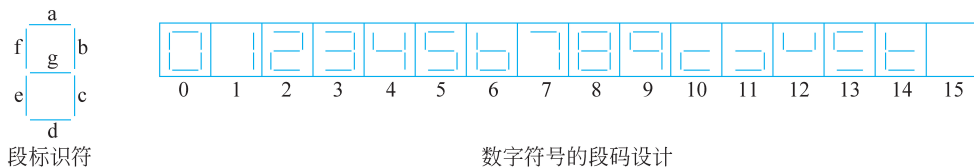


图 3-12 例 3-3 中数码管的七段数字字符

解: 建立电路的模型。图 3-13 所示为 BCD 转七段数码管编码模块的外部模型。

根据电路的功能描述可知。

输入: 4 位 8421BCD 编码, 设为 B_8 、 B_4 、 B_2 、 B_1 , 0~9 有意义, 大于 9 无意义。

输出: $a \sim g$ 七段数码管编码信号。

这是一个多输出布尔函数形成的电路, 真值表如表 3-4 所示。

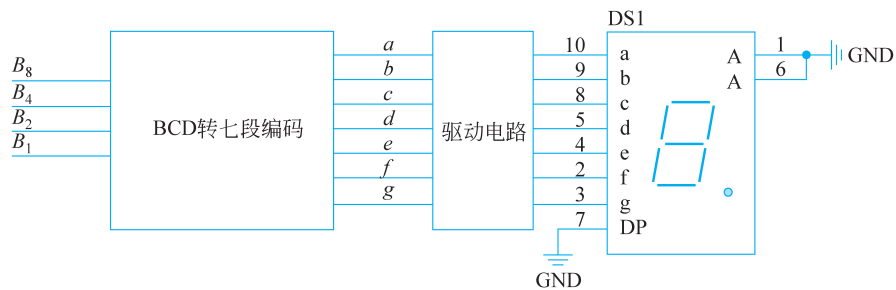


图 3-13 例 3-3 的电路模型

表 3-4 使用 8421BCD 码的七段数码管编码真值表

B_8	B_4	B_2	B_1	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	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	×	×	×	×	×	×	×

根据真值表可以写出布尔函数表达式：

$$\begin{aligned} a &= \sum m(0,2,3,5,6,7,8,9) + \sum d(10,11,12,13,14,15) \\ b &= \sum m(0,1,2,3,4,7,8,9) + \sum d(10,11,12,13,14,15) \\ c &= \sum m(0,1,3,4,5,7,8,9) + \sum d(10,11,12,13,14,15) \\ d &= \sum m(0,2,3,5,6,8) + \sum d(10,11,12,13,14,15) \\ e &= \sum m(0,2,6,8) + \sum d(10,11,12,13,14,15) \\ f &= \sum m(0,4,5,6,8,9) + \sum d(10,11,12,13,14,15) \\ g &= \sum m(2,3,4,5,6,8,9) + \sum d(10,11,12,13,14,15) \end{aligned}$$

(1) 用基于标准积项的门阵列实现该电路,电路图如图 3-14 所示。其中的与阵列是固定不变的，

用于产生函数的 16 个最小项 $m_0 \sim m_{15}$, 通过对或阵列编程即可实现上述函数表达式的电路。其中大于或等于 10 的最小项, 对应的函数值统一取值为 0。

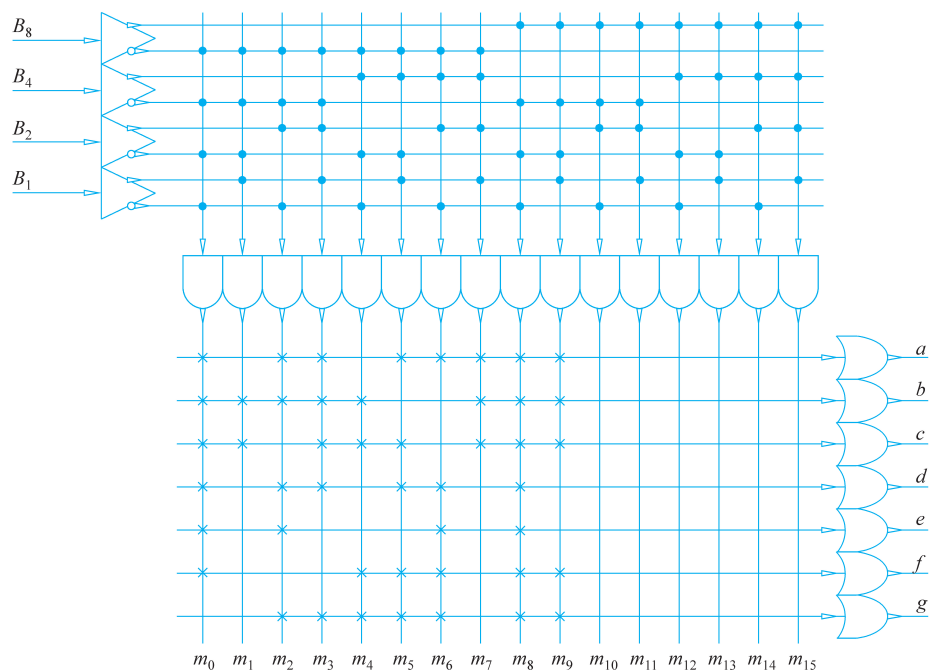


图 3-14 例 3-3 基于标准积项的门阵列电路图

(2) 用基于最简积项的门阵列实现该电路。需要将函数表达式变换为最简与或式, 才能使电路的规模最小。对上述函数表达式进行卡诺图化简, 如图 3-15 所示。

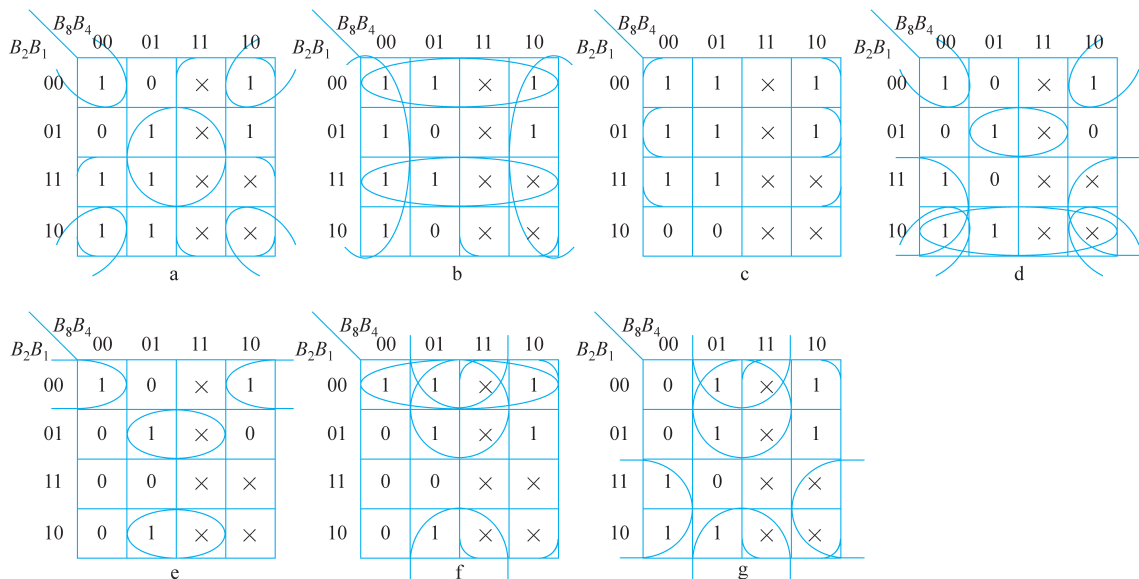


图 3-15 例 3-3 的卡诺图

七段编码的最简函数表达式为

$$a = B_8 + B_2 + B_4 B_1 + \overline{B_4} \overline{B_1}$$

$$b = \overline{B_4} + B_2 B_1 + \overline{B_2} \overline{B_1}$$

$$c = \overline{B_2} + B_1$$

$$d = \overline{B_4} \overline{B_1} + \overline{B_4} B_2 + B_2 \overline{B_1} + B_4 \overline{B_2} B_1$$

$$e = \overline{B_4} \overline{B_2} \overline{B_1} + B_4 \overline{B_2} B_1 + B_4 B_2 \overline{B_1}$$

$$f = B_8 + B_4 \overline{B_1} + B_4 \overline{B_2} + \overline{B_2} \overline{B_1}$$

$$g = B_8 + B_4 \overline{B_1} + B_4 \overline{B_2} + \overline{B_4} B_2$$

为了减少与门个数,达到整体最简,可以产生如下最简积项:

$$P_1 = \overline{B_4} \overline{B_2} \overline{B_1}; \quad P_2 = B_4 \overline{B_2} B_1; \quad P_3 = B_4 B_2 \overline{B_1}; \quad P_4 = \overline{B_4} B_2;$$

$$P_5 = B_4 \overline{B_2}; \quad P_6 = \overline{B_4} \overline{B_1}; \quad P_7 = B_4 \overline{B_1}; \quad P_8 = B_4 B_1;$$

$$P_9 = \overline{B_2} \overline{B_1}; \quad P_{10} = B_2 \overline{B_1}; \quad P_{11} = B_2 B_1; \quad P_{12} = B_8;$$

$$P_{13} = \overline{B_4}; \quad P_{14} = \overline{B_2}; \quad P_{15} = B_1$$

然后可以得到如下七段编码的或门表达式:

$$a = P_6 + P_8 + P_{10} + P_{11} + P_{12}$$

$$b = P_9 + P_{11} + P_{13}$$

$$c = P_{14} + P_{15}$$

$$d = P_2 + P_4 + P_6 + P_{10}$$

$$e = P_1 + P_2 + P_3$$

$$f = P_5 + P_7 + P_9 + P_{12}$$

$$g = P_4 + P_5 + P_7 + P_{12}$$

基于最简积项的门阵列实现的电路如图 3-16 所示。

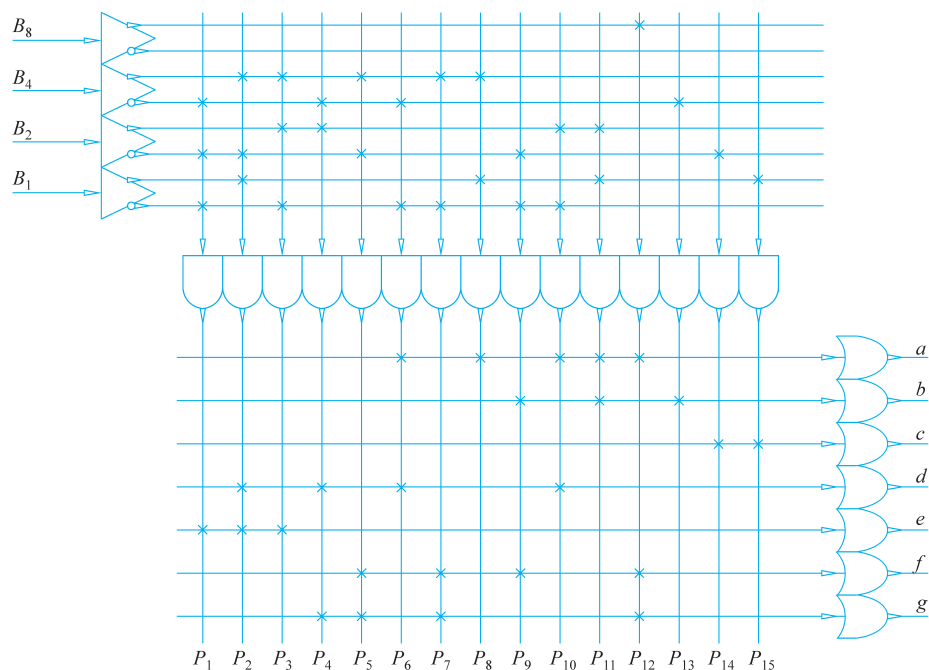


图 3-16 例 3-3 基于最简积项的门阵列电路图

3.3 组合逻辑电路分析

组合逻辑电路分析的基本任务是,根据组合逻辑电路的实现分析电路实现的功能。分析组合逻辑电路的功能,最直接的方法就是分析其真值表,因此无论使用什么样的电路实现都要首先设法得到其真值表。

实际的电路实现可以有电路原理图、VHDL 电路描述或两者混合的形式。对于原理图实现,可以根据原理图写出函数表达式,进而得到真值表;对于 VHDL 电路描述,也可以根据描述方式的不同获得电路的函数表达式,或者直接得到真值表。

门电路和门阵列器件实现的组合逻辑电路都采用了电路原理图的描述方法,所以它们都可以用原理图分析的方法进行电路分析。这里只给出门电路实现的组合逻辑电路分析方法,其他形式的电路也都是通过分析其真值表获得电路的功能,在此不再赘述。

原理图分析法的基本步骤如下。

- (1) 确定电路的外部模型,包括输入输出信号。
- (2) 写出每个门电路的函数表达式。
- (3) 用代入规则写出输出函数的表达式,并化简为最简与或表达式。
- (4) 列出函数真值表。
- (5) 描述电路功能。

例 3-4 分析图 3-17 所示逻辑电路的逻辑功能。

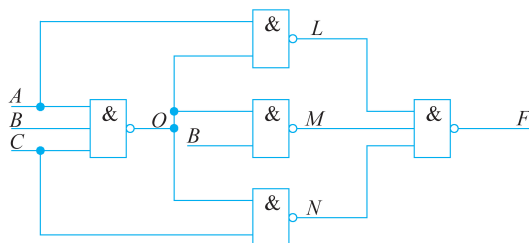


图 3-17 例 3-4 的电路图

解:

- (1) 确定电路的外部模型,电路输入为 A 、 B 、 C ,电路输出为 F 。
- (2) 写出每个门电路的函数表达式。

$$O = \overline{ABC}; \quad L = \overline{AO}; \quad M = \overline{BO}; \quad N = \overline{CO}; \quad F = \overline{LMN}$$

- (3) 用代入法写出输出函数的表达式,并化简为最简与或式。

$$\begin{aligned} F &= \overline{LMN} \\ &= \overline{L} + \overline{M} + \overline{N} \\ &= AO + BO + CO \\ &= A \overline{ABC} + B \overline{ABC} + C \overline{ABC} \\ &= A(\overline{A} + \overline{B} + \overline{C}) + B(\overline{A} + \overline{B} + \overline{C}) + C(\overline{A} + \overline{B} + \overline{C}) \\ &= A\overline{B} + A\overline{C} + \overline{A}B + B\overline{C} + \overline{A}C + \overline{B}C \end{aligned}$$

卡诺图如图 3-18 所示,出现了非必要质蕴含项,用最少的质蕴含项表示函数的最小覆盖,最简函数

表达式为 $F = A\overline{B} + \overline{A}C + B\overline{C}$ 。

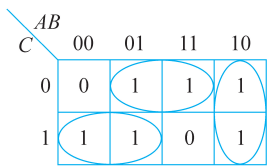


图 3-18 例 3-4 的卡诺图

(4) 列出函数真值表,如表 3-5 所示。

表 3-5 例 3-4 的真值表

A	B	C	F
0	0	0	0
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	0

(5) 描述电路功能。由真值表可知,电路的功能是,若输入 A 、 B 、 C 的值不一样,则输出 F 就为 1;否则, F 为 0。可进一步总结为,这是一个三变量非一致电路。

3.4 常用组合逻辑电路的分析与应用

常用组合逻辑电路是指在实际工程应用中,经常使用的组合逻辑电路功能单元电路。这些电路完全可以使用门电路自行设计实现,但由于在实际中频繁使用,器件生产厂家把这些功能单元电路制作成标准集成电路器件供电路设计人员选择使用。

常用组合逻辑电路主要包括二进制运算电路、编码器、译码器、数据选择器、数据分配器及数值比较器等,本节讨论这些功能单元电路的原理、器件及应用。

在讲解具体器件之前有一点需要说明,本书主要讲解这些常用器件的基本原理,以及通过查阅手册了解集成逻辑器件功能的一般方法,集成逻辑器件种类繁多不可能一一讲解,读者应该培养自行查阅器件手册学习新器件的能力。

3.4.1 二进制加法器

数字计算机能进行各种信息处理,最常用的是各种算术运算,二进制加法是最基本的算术运算。实现二进制加法运算的逻辑电路,通常称为加法器。下面围绕加法器的设计进行较为详细的讨论,主要包括半加器、全加器、串行进位的 4 位二进制并行加法器及先行进位的 4 位二进制并行加法器。

1. 半加器

对两个 1 位二进制数进行算术加法运算,求得和及进位的逻辑电路,称为半加器。

1) 半加器模型

按照组合逻辑电路设计方法,首先给出它的外部模型,如图 3-19 所示。

图 3-19 中,A、B 分别为两个 1 位二进制数输入; S_H 、 C_H 分别为加法运算产生的本位和及向高位的进位。根据半加器的功能要求,可列真值表如表 3-6 所示。



图 3-19 半加器模型

表 3-6 半加器的真值表

A	B	S_H	C_H
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

由真值表可写出函数表达式: $S_H = \overline{A}B + A\overline{B}$; $C_H = AB$ 。

2) 用集成逻辑门电路实现

(1) 直接用与、或、非门实现半加器。其实现方法比较简单,就是用门电路直接替换函数表达式中的逻辑关系。

(2) 直接用与非门实现半加器。需要将函数表达式直接变换为与非-与非形式,并用与非门实现。

$S_H = \overline{A}B + A\overline{B} = A \oplus B = \overline{\overline{\overline{A}B} \cdot \overline{A\overline{B}}}$; $C_H = \overline{\overline{AB}}$ 。读者可自行设计电路。

(3) 用异或门实现半加器。由于该函数中明显包含了异或关系,因此函数表达式可变换为如下比较简单的形式: $S_H = \overline{A}B + A\overline{B} = A \oplus B$; $C_H = AB$ 。电路实现如图 3-20 所示。

(4) 用无反变量输入的与非门实现半加器。观察发现, $C_H = \overline{\overline{AB}}$ 表达式中必须要用到与非项 $\overline{AB}$, 同时 $\overline{AB} = \overline{AB} \cdot B$ 、 $\overline{AB} = \overline{AB} \cdot A$, 因此,多输出布尔函数可以共用与非项 $\overline{AB}$, 函数表达式可以变换为

$$S_H = \overline{A}B + A\overline{B} = \overline{AB} \cdot B + \overline{AB} \cdot A = \overline{AB} \cdot B \cdot \overline{AB} \cdot A; \quad C_H = AB = \overline{\overline{AB}}$$

由函数表达式可以看出,该表达式不仅不用非门,而且与非门个数最少。实际应用中,这种表达式是用与非门实现无反变量输入的常用方式。电路实现如图 3-21 所示。

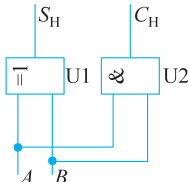


图 3-20 基于自选门电路实现的半加器

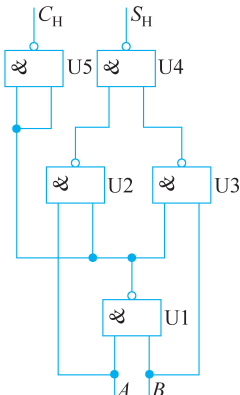


图 3-21 基于与非门实现的半加器

2. 全加器

对两个一位二进制数进行算术加法运算,并考虑低位进位,相当于 3 个一位二进制数相加,得到和及进位的逻辑电路,称为全加器。



图 3-22 全加器模型

1) 全加器模型

全加器外部模型如图 3-22 所示。

图 3-22 中, A_i 和 B_i 分别为两个一位二进制数的输入, C_{i-1} 为低位进位输入, S_i 和 C_i 分别为加法运算产生的本位和及向高位的进位。根据功能要求可列真值表如表 3-7 所示。

表 3-7 全加器的真值表

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

由真值表写出函数表达式: $S_i = \sum m(1,2,4,7); C_i = \sum m(3,5,6,7)$ 。

2) 用集成逻辑门电路实现全加器

画出 S_i 和 C_i 的卡诺图,如图 3-23 所示。

由卡诺图可得

$$S_i = \overline{A_i} \overline{B_i} C_{i-1} + \overline{A_i} B_i \overline{C_{i-1}} + A_i \overline{B_i} \overline{C_{i-1}} + A_i B_i C_{i-1}$$
$$C_i = A_i B_i + A_i C_{i-1} + B_i C_{i-1}$$

(1) 直接用与、或、非门电路实现全加器。从全加器的函数表达式中可以看到,选用与、或、非门直接实现全加器,需要用到 3 个非门、3 个两输入与门、4 个三输入与门、1 个三输入或门、1 个四输入或门,共需 12 个门电路,读者可以自行设计。

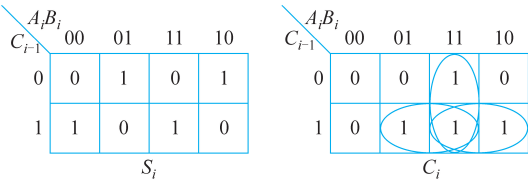


图 3-23 全加器的卡诺图

(2) 直接用与非门、或非门、与或非门实现全加器。由组合逻辑电路的门电路实现可以看出,用不同的逻辑器件实现布尔函数的关键是函数形式的变换。用与非门、或非门、与或非门实现全加器,首先要把布尔函数变换为与非-与非、或非-或非、与或非表达式,然后用不同的逻辑器件替换布尔函数中的逻辑关系,即可形成相应的逻辑电路。这些电路实现方法是组合逻辑电路实现必须要掌握的基本的方法,作为练习由读者自行设计这部分电路。

(3) 用半加器实现全加器。如果将全加器 $A_i + B_i + C_{i-1}$ 看成两个半加器,则其计算表达式可以表

示为半加器($A_i + B_i$),与半加器($A_i + B_i$)+ C_{i-1} 的加法运算。

第一个半加器($A_i + B_i$)表达式可以表示为 $S_{iH} = A_i \oplus B_i$; $C_{iH} = A_i B_i$ 。

第二个半加器实现($S_{iH} + C_{i-1}$),表达式为 $S_i = S_{iH} \oplus C_{i-1}$; $C_i = C_{iH} + S_{iH} C_{i-1}$ 。

电路实现如图 3-24 所示。图中用两个异或门实现的半加器实现了全加器电路。

如果将上面表达式中的异或关系都用与非门实现,则可得另一种半加器实现的全加器:

$$S_{iH} = \overline{\overline{A_i B_i} \cdot \overline{B_i} \cdot \overline{A_i B_i} \cdot A_i}; \quad C_{iH} = \overline{\overline{A_i B_i}};$$

$$S_i = \overline{\overline{S_{iH} C_{i-1}} \cdot \overline{C_{i-1}} \cdot \overline{S_{iH} C_{i-1}} \cdot S_{iH}}; \quad C_i = \overline{\overline{C_{iH}} \cdot \overline{S_{iH} C_{i-1}}}$$

其电路实现如图 3-25 所示。这实际上是一种用与非门实现的全加器方案。

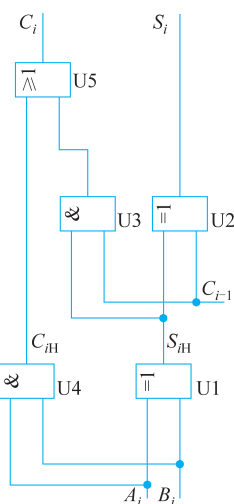


图 3-24 半加器实现的全加器(异或门)

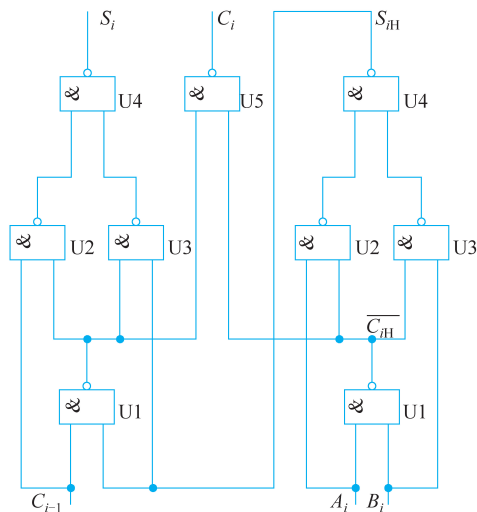


图 3-25 半加器实现的全加器(与非门)

3. 串行进位的 4 位二进制并行加法器

对两个 n 位二进制数进行算术加法运算,并考虑最低位进位,得到和及最高位进位的逻辑电路,称为二进制加法器。计算机的算术运算单元就是加法器。由于二进制加法器的两个 n 位二进制数和最低位进位同时输入,和及最高位进位也同时产生,因此也称为二进制并行加法器。

按照进位方式和电路结构的不同,二进制并行加法器可分为行波进位(或串行进位)的二进制并行加法器和先行进位(或并行进位)的二进制并行加法器两种。

1) 行波进位的二进制并行加法器

行波进位的二进制并行加法器的逻辑框图如图 3-26 所示。

将 n 位二进制数加法器看成由 n 个全加器的简单连接而成,只要把各位全加器串联起来,低位全加器的进位输出连到相邻的高位全加器的进位输入,即可构成行波进位的 n 位二进制并行加法器。其函数表达式为各个全加器的简单组合,输入为 $A_n \sim A_1$ 、 $B_n \sim B_1$ 、最低位进位 C_0 ,输出为 $S_n \sim S_1$ 、最高位进位 C_n 。各个全加器的函数表达式如下:

$$S_1 = A_1 \oplus B_1 \oplus C_0, \quad C_1 = A_1 B_1 + (A_1 \oplus B_1) C_0$$

$$\vdots$$

$$S_i = A_i \oplus B_i \oplus C_{i-1}, \quad C_i = A_i B_i + (A_i \oplus B_i) C_{i-1}$$

$$\vdots$$

$$S_n = A_n \oplus B_n \oplus C_{n-1}, \quad C_n = A_n B_n + (A_n \oplus B_n) C_{n-1}$$

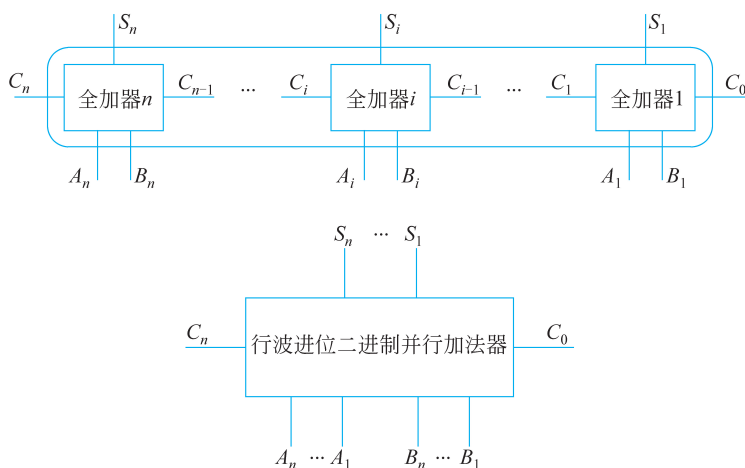


图 3-26 行波进位的二进制并行加法器

由函数表达式可见,这样构成的加法器,两个 n 位二进制数 A_n 到 A_1 、 B_n 到 B_1 和最低位进位 C_0 是同时输入的,加法运算结果 $S_n \sim S_1$ 、最高位进位 C_n 也是同时产生的,但其“正确”结果产生的时间并不一致,每一位全加器的结果依赖于其低位全加器进位信号的时间延迟。

行波进位的二进制并行加法器的进位链是由低位向高位逐级传递的,好像行波一样。正确的高位和及进位,需要等低位的进位形成后才能确定。如做手算加法那样,要从低位逐级求出进位,最后才能求得高位的和及进位。因此,这种加法器进位是串行的,形成进位的速度很慢,加法器的速度受进位传递时间的限制。

若每级全加器形成进位的延时为 $2t_{pd}$,则在最坏情况下,从最低位全加器的输入产生最高位进位 C_n 需要时间为 $2nt_{pd}$ 。当位数 n 增大时,完成一次加法所需时间也随之增加。在速度要求不高的场合,这是一种快速构成加法器的方法。在实际的计算机系统中,采用的是速度较快的先行进位二进制并行加法器。

2) 先行进位的二进制并行加法器

为了提高加法速度,在逻辑设计上采用先行进位的方法,即每位二进制加法的进位是同时产生的,不需要等到低位的进位送来后才形成。从全加器进位函数表达式 $C_i = A_i B_i + (A_i \oplus B_i) C_{i-1}$ 可以看出,每个高位进位输出都依赖于低位进位输入,要想使各个进位同时产生,必须消除这种依赖关系。

分析进位函数表达式 $C_i = A_i B_i + (A_i \oplus B_i) C_{i-1}$ 可以知道,产生进位有两种可能。

(1) 本位产生进位:当 $A_i B_i = 11$ 时,进位 $C_i = 1$,它只与本位的输入 A_i 、 B_i 有关,与低位来的进位 C_{i-1} 无关。

(2) 低位进位 C_{i-1} 向高位传递:当 $A_i \oplus B_i = 1$ 时,低位进位信号 $C_{i-1} = 1$ 就会向高位传递,即 $C_i = C_{i-1} = 1$ 。令

$$G_i = A_i B_i, \quad P_i = A_i \oplus B_i$$

其中, G_i 为第 i 位的进位生成项, P_i 为进位传递条件,则 $C_i = A_i B_i + (A_i \oplus B_i) C_{i-1} = G_i + P_i C_{i-1}$,进而可以把 S_i 写成 $S_i = P_i \oplus C_{i-1}$ 。

为便于讲解,这里只以 4 位全加器为例,由这两个公式写出各位全加器的表达式:

$$S_1 = P_1 \oplus C_0, \quad C_1 = G_1 + P_1 C_0$$

$$S_2 = P_2 \oplus C_1, \quad C_2 = G_2 + P_2 C_1$$

$$S_3 = P_3 \oplus C_2, \quad C_3 = G_3 + P_3 C_2$$

$$S_4 = P_4 \oplus C_3, \quad C_4 = G_4 + P_4 C_3$$

从上面表达式中可以看出,各个全加器之间仍然存在进位依赖关系,为了消除这种依赖关系,将上面表达式中的每一级 C_{i-1} 的表达式代入上一级 C_i 表达式中,可以得到

$$S_1 = P_1 \oplus C_0, \quad C_1 = G_1 + P_1 C_0$$

$$S_2 = P_2 \oplus C_1, \quad C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 C_0$$

$$S_3 = P_3 \oplus C_2, \quad C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0$$

$$S_4 = P_4 \oplus C_3, \quad C_4 = G_4 + P_4 C_3 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_0$$

在上面表达式中,各个全加器的进位 C_i 不再依赖于低位的进位,各级 C_i 变成了 G_i 到 G_1 、 P_i 到 P_1 、 C_0 的函数。其中, C_0 与 A_i 、 B_i 同时输入, G_i 到 G_1 、 P_i 到 P_1 是 A_i 与 B_i 的函数也同时产生,则理论上 C_i 到 C_1 会同时产生,进而 S_i 到 S_1 也会同时产生,如此即实现了先行进位二进制并行加法器。

图 3-27 给出了先行进位的二进制并行加法器的逻辑图。由图可见,利用先行进位方法,各位进位都只经过三级门延时(约 $4t_{pd}$, 假定异或门延时为 $2t_{pd}$),而形成各位和 S_i 需经四级门延时(约 $6t_{pd}$),即先形成进位,再形成加法的和。

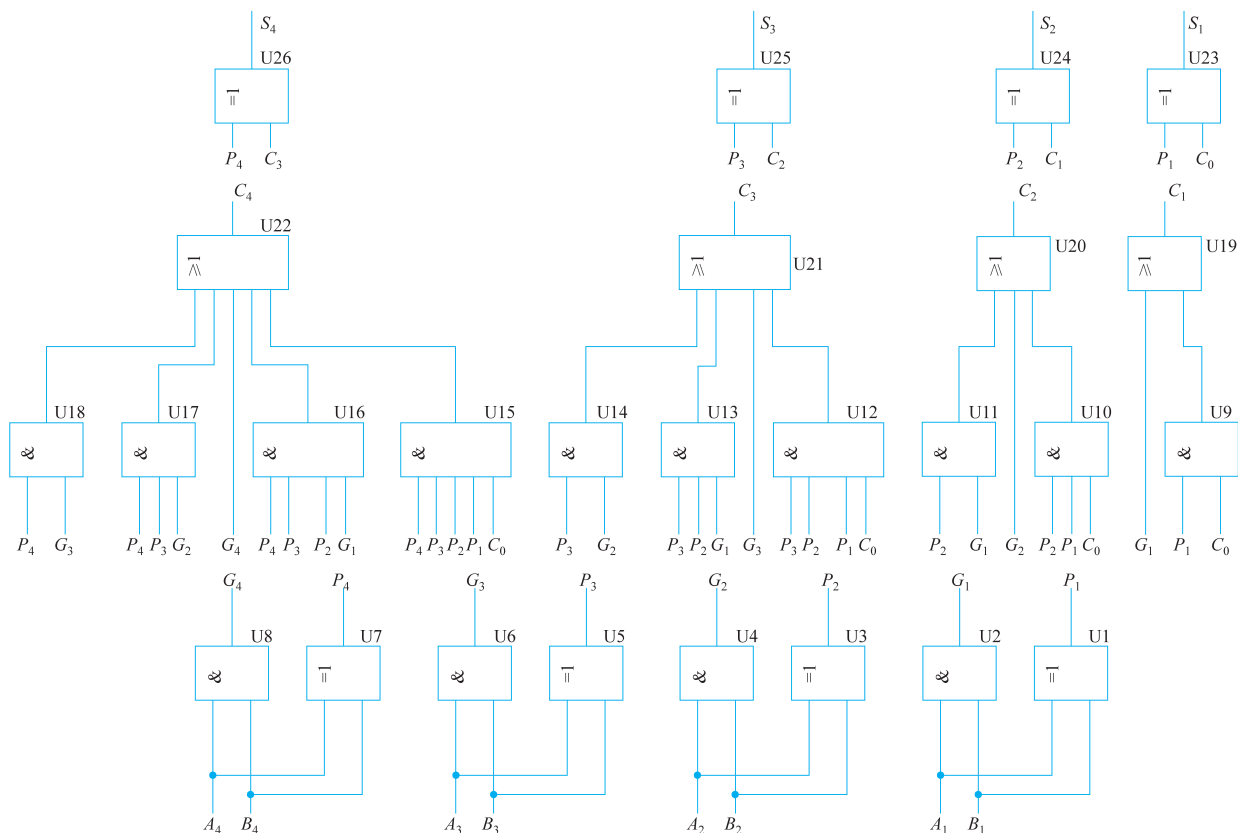


图 3-27 先行进位的二进制并行加法器

显然,进位传递时间的节省是以逻辑电路的复杂性为代价的。随着位数的增加,所需电路元件也迅速增加,而且门电路的输入端数目也会增大。因此,当位数较多时,通常采用折中的办法,即将字长 n 位分为若干组。例如,一个 16 位并行加法器,可分为 4 组,每组 4 位。每组内采用先行进位,组间采用串行进位,这样比较节省器件。

4. 先行进位的 4 位二进制并行加法器

4 位二进制并行加法器是一种典型的中规模集成电路(MSI)逻辑部件,在标准 TTL 系列中,4 位二进制并行加法器器件型号有 7483、74283、74181,其中 74283 器件引脚排列如图 3-28(a)所示,其原理图符号如图 3-28(b)所示。

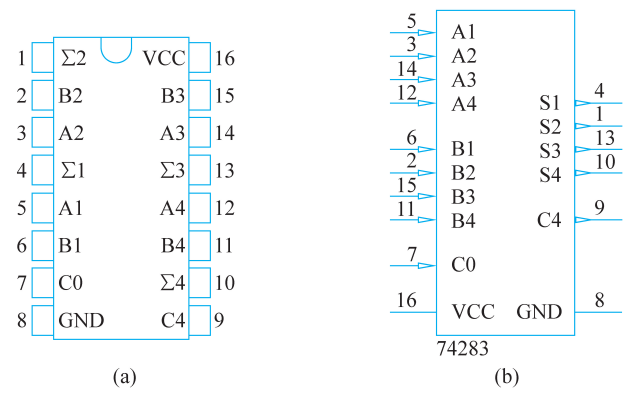


图 3-28 74283 的引脚排列和原理图符号
(a)引脚排列; (b)原理图符号

在 TTL 系列中还有专门的先行进位器件,如 7482、74182 等,利用这些器件,可以构成 16、64 位等 n 位并行进位加法器,在此不再赘述。

4 位二进制并行加法器不仅用于实现二进制算术加法运算,也可以用它们实现与加法相关的逻辑电路。下面举例说明 4 位二进制并行加法器的应用。

例 3-5 用 4 位二进制并行加法器设计一个 4 位二进制加法/减法器。

解: 加法器只能做加法运算,用加法器设计加法/减法器,可以使用补码运算规则 $[A \pm B]_{\text{补}} = [A]_{\text{补}} + [\pm B]_{\text{补}}$,将减法算术运算变换为加法运算,但需要计算负数的补码。

设 A 和 B 分别为 4 位二进制数,并令 $A = a_4 a_3 a_2 a_1, B = b_4 b_3 b_2 b_1$,其加减运算通过选择变量 M 加以控制,当 $M=0$ 时,执行 $A+B$ 运算;当 $M=1$ 时,执行 $[A]_{\text{补}} + [-B]_{\text{补}}$ 运算。

根据设计要求,加法运算时加 B 的原码,并使进位输入 $C_0=0$;减法运算时,取 B 的反码,并使 $C_0=1$,即加 B 的补码。上述逻辑功能可用图 3-29 表示。

由图 3-29 可以看出,4 位二进制数 A 直接加到全加器输入端,而 4 位二进制数 B 需求补后,同时也加到并行加法器的进位输入端。

求补运算利用了正数补码是自身、负数补码求反加 1 的运算规则。加法时 $M=0, B$ 不取反不加 1 ($C_0=M=0$);减法时 $M=1, B$ 与选择变量 M 进行异或运算实现取反功能,同时末尾加 1 ($C_0=M=1$)。

3.4.2 编码器

在数字系统中,常需将有特定意义的多个不同的信号,编码成相应的二进制编码,实现编码的电路称为编码器(encoder)。 n 位二进制编码器最多可以对 2^n 个输入信号进行编码。

图 3-30 为 3 位二进制编码器的外部模型。

8 个信号输入端: $I_0 \sim I_7$,用于输入不同物理含义的 8 个信号 $I_0 \sim I_7$ (物理含义可由用户定义)。

3 个编码输出端: $A_0 \sim A_2$,用于输出 8 个输入信号对应的二进制编码 $A_0 \sim A_2$ 。

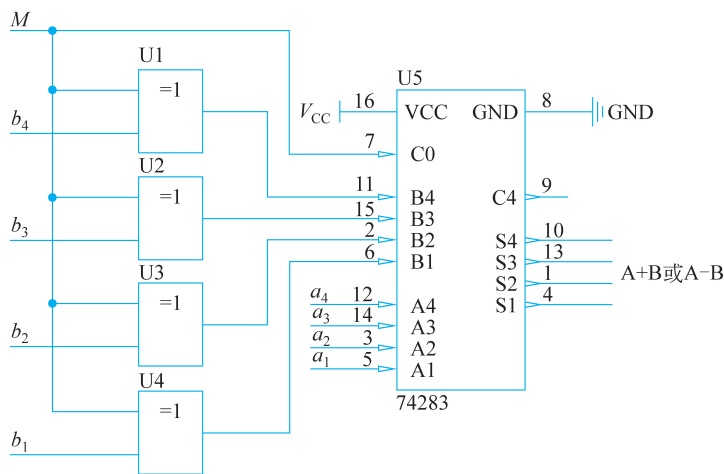


图 3-29 例 3-5 的原理图

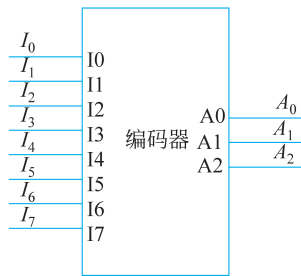


图 3-30 编码器的外部模型

1. 普通编码器的基本原理

普通编码器的基本功能是,在正常情况下,8 个输入信号中只有一个输入信号为 1,其余输入信号为 0,输出二进制编码对应于该信号的编号。其功能表如表 3-8 所示。

表 3-8 8-3 线编码器功能表

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	A_2	A_1	A_0
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

按照功能表,可以得到二进制输出编码 $A_0 \sim A_2$ 的布尔函数表达式:

$$A_2 = I_4 + I_5 + I_6 + I_7, \quad A_1 = I_2 + I_3 + I_6 + I_7, \quad A_0 = I_1 + I_3 + I_5 + I_7$$

根据编码器的函数表达式,读者很容易设计自己的编码器电路。

2. 优先编码器原理

实际电路中,不能保证8个输入信号中只有一个输入信号为1,当多个输入信号都为1时,将产生编码混乱,为了避免这种情况的发生,需要为输入信号设置优先级,这种编码器称为优先编码器(priority encoder)。对于优先编码器,假设优先级从 I_7 到 I_0 依次降低,则其功能表应如表 3-9 所示。

表 3-9 8-3 线优先编码器的功能表

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	A_2	A_1	A_0
1	0	0	0	0	0	0	0	0	0	0
×	1	0	0	0	0	0	0	0	0	1
×	×	1	0	0	0	0	0	0	1	0
×	×	×	1	0	0	0	0	0	1	1
×	×	×	×	1	0	0	0	1	0	0
×	×	×	×	×	1	0	0	1	0	1
×	×	×	×	×	×	1	0	1	1	0
×	×	×	×	×	×	×	1	1	1	1

功能表定义了对输入信号 I_i 进行编码输出的条件是 I_i 输入有效(为1),且比 I_i 优先级高的输入信号都无效(为0),比 I_i 优先级低的输入信号可任意。按照功能表,可以得到编码输出 $A_0 \sim A_2$ 的布尔函数表达式:

$$\begin{aligned} A_2 &= I_4 \overline{I_5} \overline{I_6} \overline{I_7} + I_5 \overline{I_6} \overline{I_7} + I_6 \overline{I_7} + I_7 \\ A_1 &= I_2 \overline{I_3} \overline{I_4} \overline{I_5} \overline{I_6} \overline{I_7} + I_3 \overline{I_4} \overline{I_5} \overline{I_6} \overline{I_7} + I_6 \overline{I_7} + I_7 \\ A_0 &= I_1 \overline{I_2} \overline{I_3} \overline{I_4} \overline{I_5} \overline{I_6} \overline{I_7} + I_3 \overline{I_4} \overline{I_5} \overline{I_6} \overline{I_7} + I_5 \overline{I_6} \overline{I_7} + I_7 \end{aligned}$$

化简后可得

$$\begin{aligned} A_2 &= I_4 + I_5 + I_6 + I_7 \\ A_1 &= I_2 \overline{I_4} \overline{I_5} + I_3 \overline{I_4} \overline{I_5} + I_6 + I_7 \\ A_0 &= I_1 \overline{I_2} \overline{I_4} \overline{I_6} + I_3 \overline{I_4} \overline{I_6} + I_5 \overline{I_6} + I_7 \end{aligned}$$

根据编码器的函数表达式,可以很容易地设计编码器电路。

为了方便应用,器件生产厂商提供了常用的编码器集成电路芯片,如 8-3 线优先编码器 74148 和二进制优先编码器 74147 等。

3. 8-3 线优先编码器 74148

74148 是一个 8-3 线优先编码器,其引脚排列和原理图符号如图 3-31(a)、图 3-31(b)所示。

74148 与优先编码器基本原理相同,不同之处如下。

- (1) 采用了反变量输入输出的逻辑规定,图中输入输出变量上的“ $\overline{}$ ”只是表示反变量有效,并不表示非运算,这种表示方法在其他集成电路符号中也非常常见。
 - (2) 增加了反变量有效的使能端。
 - (3) 增加了反变量有效的扩展输出,用于扩展多于 8 输入的信号编码。
- 74148 的输入输出端如下。