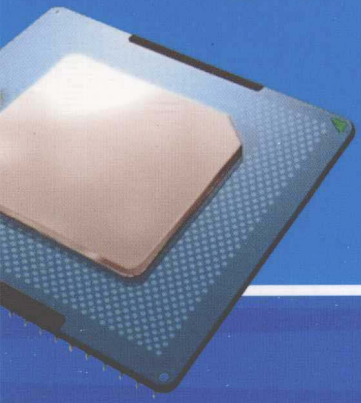
数字电路及数字系统设计

DIGITAL CIRCUIT & DIGITAL SYSTEM DESIGN



蒋万君／编

西南交通大学出版社
Http://press.swjtu.edu.cn

数字电路及数字系统设计

蒋万君　编

西南交通大学出版社
·成　都·

内 容 简 介

本书从应用角度出发，在保持数字电路教材经典内容的基础上，增加了可编程逻辑器件的数字系统设计。主要内容包括逻辑代数基础、半导体集成门电路、组合逻辑电路、时序逻辑电路、脉冲波形的产生与整形、D/A 与 A/D 转换、半导体存储器与可编程逻辑器件、硬件描述语言 VHDL、可编程逻辑器件的数字系统设计以及 GW48 型 EDA 实验开发系统的使用等。

本书以独特的视角，选取大量实例展开理论分析并做了详细解答，以区别传统的讲述方法，特别是在各章、节中插入的“说明”具有概括和提升作用。本书讲述精炼实用、深入浅出，插图规范美观，章节布局合理、系统完善，可作为计算机类、电子类等专业的本科教材。

图书在版编目（CIP）数据

数字电路及数字系统设计 /蒋万君编. —成都：西南交通大学出版社，2010.8
ISBN 978-7-5643-0790-5

Ⅰ.①数… Ⅱ.①蒋… Ⅲ.①数字电路－电路设计②数字系统－系统设计 Ⅳ.①TN790.2②TP271

中国版本图书馆 CIP 数据核字（2010）第 154637 号

数字电路及数字系统设计

责任编辑 李芳芳
特邀编辑 胡芬蓉
封面设计 墨创文化
西南交通大学出版社出版发行
成都二环路北一段 111 号 邮政编码：610031 发行部电话：028-87600564
http://press.swjtu.edu.cn
四川经纬印务有限公司印刷
*
成品尺寸：185 mm×260 mm 印张：13
字数：325 千字
2010 年 8 月第 1 版 2010 年 8 月第 1 次印刷
ISBN 978-7-5643-0790-5
定价：26.00 元

前　言

现代数字信息化时代，数字电子线路系统（简称数字系统）被广泛应用于计算机、电子与通信、工业控制与检测、家用电器等领域。数字电路成为高等院校计算机、电子工程、通信工程和自动控制等专业的重要基础课程。近年来，随着微电子技术的迅速发展，超大规模的 CPLD/FPGA（Complex Programmable Logic Device/Field Programmable Gate Array）集成电路芯片的推出，借助于通用的计算机软件平台，使数字系统的研发方便快捷，成本低，效率高，而且自主知识产权能得到可靠的保障。因此，将微电子技术与计算机技术结合而形成的 EDA（Electronic Design Automation）技术已成为 IC 工程师的必备技能。

将数字系统设计的新技术与传统的数字电路整合为一门课程，体现了基础与提高的紧密结合，符合当今教学改革的发展趋势。如何将数字系统设计的基本思想和方法有机地融入传统的数字电路课程之中而又避免篇幅冗长，是作者长期教学中的大胆探索。本书力求回避“以给定电路进行分析”的模式来介绍常用的逻辑电路，而是采用“提出问题、逻辑抽象、推导演绎”的方法展开讲述，即在不缺失传统的数字电路课程基本原理的基础上，突出数字系统的设计方法讲授。本书的另一特点是在一些章、节中恰当地插入“说明”，对讲述的内容具有概括、拓展和提升作用，有利于读者开阔视野。此外，本书还选取了大量实例展开理论分析并做了详细解答，以区别传统的讲述方法。

数字电路与数字系统设计课程的实践性环节非常重要，除理论教学外还应安排实验。实验可分为验证性实验和设计性实验。验证性实验比较简单，一般是对常见的组合逻辑电路和时序逻辑电路的功能进行检验。建议实验教学的重点放在设计性实验上，参见本书的附录部分。课程的理论教学约 60 学时，实验教学约 20 学时。章节中加“*”的为选讲内容。

本书汇集了作者从事数字电路课程十多年的教学经验和已公开发表的教研成果，在重庆三峡学院第二批自编教材立项的支助下出版发行。承蒙重庆三峡学院物理与电子工程学院的蒋行达老师对书稿进行审阅，特此致谢。

由于编者水平有限，书中难免存在疏漏之处，恳切读者批评指正。

作　者

2010 年 7 月于重庆万州

目　录

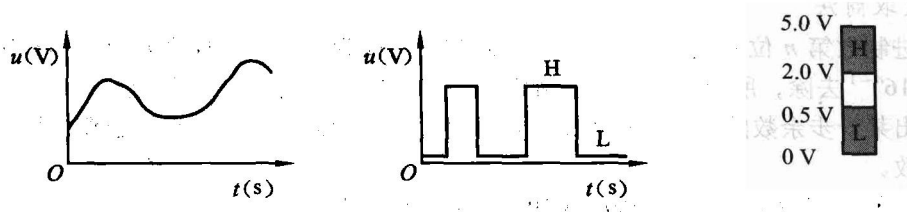
第 1 章　逻辑代数基础

逻辑代数亦称布尔代数，由英国数学家 George Boole 于 1849 年创立。布尔代数是建立在二元（抽象为 1 和 0）逻辑基础上具有与、或、非三种基本运算的逻辑代数体系。这种代数不仅广泛用于集合论、概率论和数理统计等领域，而且还是数字电路分析与设计中最重要的数学工具。

1.1　概　述

1.1.1　模拟信号与数字信号

现代电子技术主要涉及两种电信号：一种叫做模拟信号（Analog Signal），如图 1.1.1（a）所示，电压随时间连续变化；另一种叫做数字信号（Digital Signal），如图 1.1.1（b）所示，电压在时间上的变化是不连续的，电压要么处于高电平状态（H），要么处于低电平状态（L），除此之外，不会处于其他状态。所谓的状态并不是某一个固定电压值，而是一个允许的取值区间，如图 1.1.1（c）所示。



（a）模拟信号　　（b）数字信号　　（c）状态取值区间

图 1.1.1　模拟信号与数字信号

本书的研究对象是数字电子信号，传送这种信号的电子线路叫做数字逻辑电路，简称数字电路。数字信号有正负逻辑之分，若将高电平状态定义为 1，低电平状态定义为 0，这样的数字信号称为正逻辑信号；反之，则称为负逻辑信号。值得注意的是，这里的 1 和 0 是两种状态的抽象表示，没有大小之分。绝大多数数字电路采用正逻辑信号，以后不特别声明，数字信号指的都是正逻辑信号。

说明：电子计算机内部存储、传输和处理的信号就是数字信号。例如，某计算机的数据总线（Data BUS）由 32 根单线并列组成，每根单线的电平状态 1 和 0 定义为二进制数的 1 和 0，且各单线的权重依次为 2^0，2^1，…，2^{31}，那么该数据总线能传输 32 位的二进制数。

1.1.2 进制转换与十进制数的编码

二进制及其运算是德国著名数学家莱布尼兹最早提出的，1679 年莱布尼兹发表了论文《二进制算术》。因为二进制数易于电路实现且运算规则简单，所以二进制成为现代数字电子系统的基础。

1. 十进制与二进制的互换

1）二进制数的定义及一些特殊的数

（1）二进制数的定义：$(N)_2=k_n2^n+k_{n-1}2^{n-1}+\cdots+k_12^1+k_02^0+k_{-1}2^{-1}+\cdots+k_{-m}2^{-m}$，$k_i$为基数“2”的第 i 次幂系数，取值 0 或 1。

（2）二进制数 2^0，2^1，…，2^{10}，2^{11}，2^{12}，2^{13}对应的十进制数分别为 1，2，…，1 024，2 048，4 096，8 192。

（3）$2^n=(100\cdots0)_2$，其中 1 后面有 n 个 0。

（4）$2^n-1=(11\cdots1)_2$，其中有 n 个 1。

（5）$2^{-n}=(0.00\cdots01)_2$，其中小数点之后有 $n-1$ 个 0。

（6）$1-2^{-n}=(0.11\cdots1)_2$，其中小数点之后有 n 个 1。

2）2 的整幂加减拼凑法

对于接近 2^n的十进制数化为二进制数，采用 2 的整幂加减拼凑法进行口算简明快捷，后面介绍的“除权取商法”和“减权取 1 法”也可结合该方法使用。例如，将十进制数 135 视为 $128(2^7)$加 7，则其结果为 1 后面有 7 个 0 的二进制数再加上$(111)_2$，所以 $135=(10000111)_2$。将十进制数 2 034 视为 $2\,047(2^{11}-1)$减 13，则其结果为有 11 个 1 的二进制数再减去$(1101)_2$，所以 $2\,034=(11111110010)_2$。

3）除权取商法

用十六进制数第 n 位的权重 16^n去除十进制数，其商为十六进制数第 n 位上的数字；将其余数再用 16^{n-1}去除，所得商为十六进制数第 $n-1$ 位上的数字……重复这样的运算步骤，直到容易看出某一步余数的二进制数为止。最后将每一次的商和最后一步的余数按权重拼成一个二进制数。

【例 1.1.1】 将十进制数 87，969，3 393 分别化为二进制数。

解 （1）因为 87 除以 16 商 5 余 7，所以 $87=(101\ 0111)_2$。括号中插入一个空以方便读者理解。

（2）$969\div16^2=3\cdots\cdots201\ \rightarrow\ (11)_2\cdots\cdots201$，前者为商，后者为余数，以下同。

$201\div16=12\cdots\cdots9\ \rightarrow\ (1100)_2\cdots\cdots(1001)_2$

所以 $969=(11\ 1100\ 1001)_2$。

（3）$3\,393\div16^2=13\cdots\cdots65\ \rightarrow\ (1101)_2\cdots\cdots(1000001)_2$，余数 65 的二进制数用口算得到。

所以 $3393=(1101\ 01000001)_2$，注意二进制数$(1000001)_2$前必须添一个 0，使其达到 8 位，因为前段 4 位数字（1101）的权重为 16^2（即 2^8）。

4）减权取 1 法

对于较大的十进制数化为二进制数可采用“减权取 1 法”。该方法是：用十进制数减去

小于该数的最大的 2^i，将其差再减去小于此差数的最大的 2^j……重复这样的运算步骤，直到容易看出某一步差数的二进制数为止。最后将 2^i，2^j，…以及最后这一步差数的二进制数按权重拼成一个二进制数。

【例 1.1.2】 将十进制数 2 169，10 508 化为二进制数。

解

$$\begin{array}{r} 2\ 169 \\ -\ 2\ 048\ (2^{11}) \\ \hline 121\ (1111001)_2 \end{array}$$ 这一步口算

故 $2\ 169=(100001\underline{1111001})_2$

$$\begin{array}{r} 10\ 508 \\ -\ 8\ 192\ (2^{13}) \\ \hline 2\ 316 \\ -\ 2\ 048\ (2^{11}) \\ \hline 268\ (100001100)_2 \end{array}$$ 这一步口算

故 $10\ 508=(10100\underline{100001100})_2$

5）二进制数化为十进制数

从二进制整数的最低位起，将二进制数视为十六进制数，即每 4 位分为一段，然后再按十六进制数的权重展开求和。

【例 1.1.3】 将二进制数$(110001.10111)_2$ 化为十进制数。

解 $(110001.10111)_2=(110\ 0011\ 0111)_2/2^5=(6\times16^2+3\times16+7)/32=1\ 591/32=49.718\ 75$

2. 二进制的数学意义

将抽象的二进制数应用于具体的实例之中，是对二进制的数学意义最生动的诠释。

1）两个古典数学问题

（1）相传古代印度国王舍汗要褒奖国际象棋发明者达依尔，问他需要什么。达依尔回答说："国王只要在国际象棋棋盘（8×8 格）的第一格上放 1 粒麦子，第二格上放 2 粒麦子，第三格上放 4 粒麦子，第四格上放 8 粒麦子，按此规律一直放满棋盘的最后一格，我心足矣。"

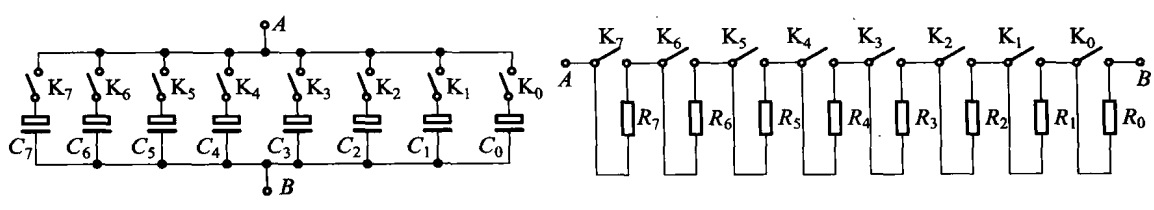
根据二进制数的定义，将棋盘上的麦子数用二进制数表示应为 64 个 1，那么麦子总数就是 $2^{64}-1$（$>10^{16}$）。这是一个惊人的天文数字，看来达依尔是在戏弄国王。

（2）"一尺之棰，日取其半，万世不竭。"

根据二进制数的定义，将前 n 日所得用二进制数表示之，该数应为小数点之后有 n 个 1，其总和为 $1-2^{-n}$。所以 $2^{-1}+2^{-2}+\cdots+2^{-n}=(0.11\cdots1)_2<1$ 总是成立的。

2）二进制与含权开关量

图 1.1.2（a）所示是由 8 个开关和 8 个电容器组成的含权电容电路，图 1.1.2（b）所示是由 8 个开关和 8 个电阻组成的含权电阻电路。其中电容器的取值为 $C_i=2^i\times1\ \mu F(i=0,1,\cdots,7)$，电阻的取值为 $R_j=2^j\times10\ \Omega$（$j=0$，1，…，7）。这是一种二进制含权电容或电阻电路，



（a）含权电容电路　　（b）含权电阻电路

图 1.1.2　含权电容、电阻电路

在图 1.1.2（a）中，用“1”表示开关 K_i 处于 ON 状态，用“0”表示开关 K_i 处于 OFF 状态；在图 1.1.2（b）中，用“1”表示开关 K_j 处于 OFF 状态，用“0”表示开关 K_j 处于 ON 状态。于是 8 个开关的每种组合状态对应于一个 8 位二进制数，所以电路取值为 0～255 之间的任一整数。

【例 1.1.4】 将电容 C_{AB} 的值设置为 168 μF，将电阻 R_{AB} 的值设置为 2 500 Ω。

解 （1）因为 $168=(10101000)_2$，所以在图 1.1.2（a）所示的电路中，将开关 K_7，K_5，K_3 闭合，其余 5 个开关断开。

（2）因为 $250=(11111010)_2$，所以在图 1.1.2（b）所示的电路中，将开关 K_2，K_0 闭合，其余 6 个开关断开。

1.1.3 十进制数的编码

十进制数的每一位由 0～9 十个数字组成，在数字逻辑系统中，1 位十进制数要用 4 位二进制数表示，即用二进制数对十进制数进行编码，这样的代码简称 BCD（Binary Coded Decimal）码。表 1.1.1 是几种常用的 BCD 码，其中 8421BCD 码是最常用的一种十进制编码。

4 位二进制数共有 2^4 个代码，除了对十进制数的编码外还有 6 个伪码，不同的 BCD 码伴有不同的伪码。在计算机中十进制数的运算是以 BCD 码进行的，一旦结果产生伪码就要对其进行修正处理。例如，当用 8421BCD 码表示的两个十进制数进行相加，若某位的和出现伪码（即大于 9）或者该位向高位产生了进位，则该位的和还要加 6 进行修正；当用 8421BCD 码表示的两个十进制数进行相减，若某位向高位产生了借位，则该位的差还要减 6 进行修正。

表 1.1.1 常用 BCD 码

十进制数	8421 码	2421 码	余 3 码
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100
编码规则	权 8421	权 2421	8421 码 $+(11)_2$

1.2 逻辑代数中的基本运算及基本公式

1.2.1 逻辑函数

图 1.2.1 所示为某数字逻辑电路，输入信号 A_1，A_2，…，A_n 叫做逻辑变量，输出信号 Y 叫做逻辑函数，它们的取值为逻辑值 1 或 0。显然输出信号的变化是受输入信号的影响，或者说输出信号是关于输入信号的函数，即存在逻辑函数式

$$Y=f(A_1, A_2, \cdots, A_n) \qquad (1.2.1)$$

逻辑函数的另一种表示方法是逻辑真值表，真值表的一行称为一个状态行，该行的内容是若干个变量的一组逻辑值和由此决定的函数值，n 个变量的逻辑真值表共有 2^n 个状态行。逻辑函数式和逻辑真值表可以相互转换。

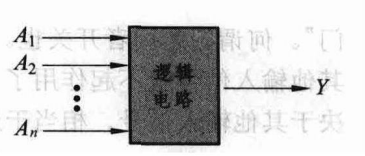
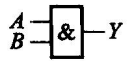
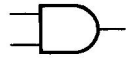
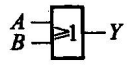
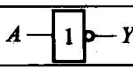
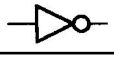
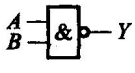
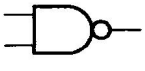
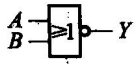
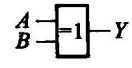
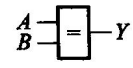
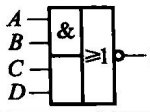
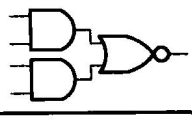


图 1.2.1 逻辑函数与变量的关系

1.2.2 基本运算及基本公式

在逻辑代数中，与、或、非是三种最基本的逻辑运算，其他逻辑运算是这三种基本运算的复合。表 1.2.1 是逻辑代数中的常见运算，其中列出了每种运算所对应的电路符号。表 1.2.2 是逻辑代数的基本公式，表 1.2.3 列出了逻辑代数的常用公式。

表 1.2.1 常见逻辑运算

逻辑运算	国际逻辑图符	国外流行图符	逻辑函数	逻辑真值表		
				A	B	Y
与			$Y=A\cdot B$	0 0 1 1	0 1 0 1	0 0 0 1
或			$Y=A+B$	0 0 1 1	0 1 0 1	0 1 1 1
非			$Y=\overline{A}$	0 1		1 0
与非			$Y=\overline{A\cdot B}$	0 0 1 1	0 1 0 1	1 1 1 0
或非			$Y=\overline{A+B}$	0 0 1 1	0 1 0 1	1 0 0 0
异或			$Y=A\oplus B$ $=\overline{A}B+A\overline{B}$	0 0 1 1	0 1 0 1	0 1 1 0
同或			$Y=A\odot B$ $=\overline{A}\overline{B}+AB$	0 0 1 1	0 1 0 1	1 0 0 1
与或非			$Y=\overline{AB+CD}$	略		

1. 与运算

与运算又称逻辑乘。由表 1.2.2 的第 1、2 行左边可知“信号 0 封锁与门，信号 1 开放与

门”。何谓门？门者开关也。这句话的意思是：一旦信号 0 打入与门，该与门的输出即为 0，其他输入信号就不起作用了，相当于这些信号被阻止了；信号 1 打入与门，该与门的输出取决于其他输入信号，相当于这些输入信号顺利地通过了与门。

2. 或运算

或运算又称逻辑加。由表 1.2.2 的第 1、2 行右边可知“信号 1 封锁或门，信号 0 开放或门”。

3. 非运算

非运算是求逻辑变量的相反状态，常常也称为逻辑取反。

4. 异或运算

异或运算又称模 2 加（减），两个变量取异或，可以视为这两个变量的值进行二进制加（减），只是进（借）位自动丢失；同或运算是当两个变量的值相同时结果为 1，当两个变量的值相异时结果为 0。同或与异或为互补运算，即同或取非为异或，异或取非为同或。

5. 反　演

表 1.2.2 中第 8 行是著名的德·摩根（De Mogen)律，亦称反演律。用一句口诀概括，就是“头上切一刀，展开运算变个号”。

表 1.2.2　逻辑代数基本公式

行号	基本公式（左、右成对偶关系）	
0		$\overline{1}=0;\ \overline{0}=1$
1	$0\cdot A=0$	$1+A=1$
2	$1\cdot A=A$	$0+A=A$
3	$A\cdot A=A$	$A+A=A$
4	$A\cdot\overline{A}=0$	$A+\overline{A}=1$
5	$A\cdot B=B\cdot A$	$A+B=B+A$
6	$A(BC)=(AB)C$	$A+(B+C)=(A+B)+C$
7	$A(B+C)=AB+AC$	$A+(BC)=(A+B)(A+C)$
8	$\overline{A\cdot B}=\overline{A}+\overline{B}$	$\overline{A+B}=\overline{A}\cdot\overline{B}$
9	$\overline{\overline{A}}=A$	

表 1.2.3　逻辑代数常用公式

行号	常用公式
1	$A+AB=A$
2	$A(A+B)=A$
3	$A+\overline{A}B=A+B$
4	$A(\overline{A}+B)=AB$
5	$AB+A\overline{B}=A$
6	$(A+B)(A+\overline{B})=A$
7	$AB+\overline{A}C+BC=AB+\overline{A}C$
8	$(A+B)(\overline{A}+C)(B+C)$ $=(A+B)(\overline{A}+C)$
奇数行与偶数行成对偶关系	

1.2.3　常用公式及定理

1. 常用公式

表 1.2.3 的第 7 行说明：若某原变量乘以一个因子，加上其反变量乘以另一个因子，则这两个因子的乘积是多余项。

2. 常用定理

（1）代入定理：逻辑等式两边所出现的同一变量代之以另一函数式，则逻辑等式仍成立。

（2）香农（Shannon）定理：

$$\overline{f(A_1, A_2 \cdots A_n, 0, 1, +, \cdot)} = f(\overline{A_1}, \overline{A_2}, \cdots, \overline{A_n}, 1, 0, \cdot, +) \tag{1.2.2}$$

该等式的意义是任何反函数（原函数取非）都可以通过对原函数中的所有变量取非，并将其中的 0 换为 1，1 换为 0，“ · ”换为“ + ”；“ + ”换为“ · ”而得到。其实香农定理就是德·摩根律的推广。

（3）对偶定理：若两逻辑表达式相等，则它们的对偶式也相等。

所谓对偶式是这样定义的：对于任何一个逻辑表达式 Y，若将其中的 0 换为 1，1 换为 0，“ · ”换为“ + ”，“ + ”换为“ · ”而得到一个新的逻辑表达式 Y'，Y'与 Y 互为对偶式。例如，表 1.2.3 中的第 1 与第 2 行、第 3 与第 4 行、第 5 与第 6 行、第 7 与第 8 行成对偶关系。利用对偶定理，我们只要证明了某等式成立，则该等式的对偶式自然也成立。

【例 1.2.1】 证明表 1.2.3 中第 7 行的公式。

证明：

$$\begin{aligned} AB + \overline{A}C + BC &= AB + \overline{A}C + ABC + \overline{A}BC \\ &= AB(1 + C) + \overline{A}C(1 + B) \\ &= AB + \overline{A}C \end{aligned}$$

因等式成立，根据对偶定理可知第 8 行的公式也成立。

1.2.4 由逻辑真值表写逻辑函数式

由逻辑真值表写原（反）函数表达式的方法：选取函数值为 1（0）的状态行相或，每行的状态决定变量组成一个与项，状态行中取值为 1 的记为原变量，取值为 0 的记为反变量。

【例 1.2.2】 两个二进制数 $A = A_n A_{n-1} \cdots A_i \cdots A_0$，$B = B_n B_{n-1} \cdots B_i \cdots B_0$ 相加，其中第 i 位的运算是 A_i 加 B_i 再加来自低位的进位 C_{i-1}，产生本位的和 S_i 以及向高位的进位 C_i，能实现这种运算的电路叫做一位全加器，试推导其逻辑函数式并画出电路图。

解 （1）根据二进制加法运算法则得逻辑真值表，如表 1.2.4 所示。

（2）由逻辑真值表得逻辑函数式并化简：

$$\begin{aligned} S_i &= \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} + A_iB_iC_{i-1} \\ &= \overline{A}_i(\overline{B}_iC_{i-1} + B_i\overline{C}_{i-1}) + A_i(\overline{B}_i\overline{C}_{i-1} + B_iC_{i-1}) \\ &= \overline{A}_i(B_i \oplus C_{i-1}) + A_i\overline{(B_i \oplus C_{i-1})} \\ &= A_i \oplus B_i \oplus C_{i-1} \end{aligned} \tag{1.2.3}$$

$$\begin{aligned} C_i &= \overline{A}_iB_iC_{i-1} + A_i\overline{B}_iC_{i-1} + A_iB_i\overline{C}_{i-1} + A_iB_iC_{i-1} \\ &= (\overline{A}_iB_i + A_i\overline{B}_i)C_{i-1} + A_iB_i(\overline{C}_{i-1} + C_{i-1}) \\ &= (A_i \oplus B_i)C_{i-1} + A_iB_i \\ &= \overline{\overline{(A_i \oplus B_i)C_{i-1} + A_iB_i}} \end{aligned} \tag{1.2.4}$$

（3）根据式（1.2.3）和式（1.2.4）作电路图，如图 1.2.2 所示。

说明：如何确定逻辑变量之间是“与”关系还是“或”关系呢？真值表中的一个状态行是某时刻若干个变量的取值，同时性决定“与”关系；真值表中不同的状态行是不同时刻若干个变量的取值，先后性决定“或”关系。

表 1.2.4　一位全加器真值表

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

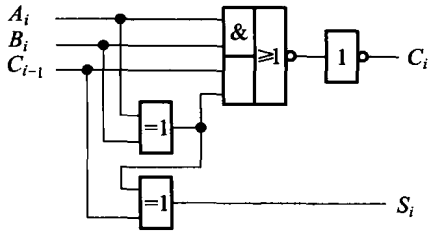


图 1.2.2　一位全加器电路

1.3　逻辑函数的标准形式及化简

1.3.1　逻辑函数的两种标准形式

1. 逻辑最小项 m_i 与标准式 1

在 n 变量逻辑函数中，若 m_i 为包含 n 个因子的乘积项，而且这 n 个变量以原变量或反变量的形式在 m_i 中必须出现一次，则该乘积项称为最小项 m_i。

（1）最小项 m_i 脚标的确定：将 n 个变量按某种顺序排定，原变量对应 1，反变量对应 0，所得二进制数为脚标。例如，对于四变量 $ABCD$ 的乘积，$m_9=A\bar{B}\bar{C}D$，$m_{13}=AB\bar{C}D$。

（2）最小项 m_i 的性质：

① 在输入变量的任何取值下必有一个最小项，而且仅有一个最小项的值为 1；

② 全体最小项之和为 1；

③ 任意两个最小项的乘积为 0。

（3）逻辑函数的标准式 1：可以将任何一个逻辑函数化为最小项之和的形式，即

$$Y=\sum m_i \tag{1.3.1}$$

例如，对于式（1.2.3），$S_i=m_1+m_2+m_4+m_7=\sum m_i\ (i=1,\ 2,\ 4,\ 7)$；对于式（1.2.4），$C_i=m_3+m_5+m_6+m_7=\sum m_i\ (i=3,\ 5,\ 6,\ 7)$。

2. 逻辑最大项 M_j 与标准式 2

在 n 变量逻辑函数中，若 M_j 为 n 个变量之和，而且这 n 个变量以原变量或反变量的形式在 M_j 中必须出现一次，则该和项称为最大项 M_j。

（1）最大项 M_j 脚标的确定：将 n 个变量按某种顺序排定，原变量对应 0，反变量对应 1，所得二进制数为脚标。例如，对于四变量 $ABCD$ 相或，$M_9=\bar{A}+B+C+\bar{D}$，$M_{13}=\bar{A}+\bar{B}+C+\bar{D}$。

（2）最大项 M_j 的性质：

① 在输入变量的任何取值下必有一个最大项，而且仅有一个最大项的值为 0；

② 全体最大项之积为 0；

③ 任意两个最大项之和为 1。

（3）逻辑函数的标准式 2：可以将任何一个逻辑函数化为最大项之积的形式，即

$$Y=\prod M_j \tag{1.3.2}$$

3. 逻辑函数标准式的互换

$$\boxed{\overline{Y}=\sum m_k} \xleftrightarrow{k\neq i} \boxed{Y=\sum m_i} \xleftrightarrow{i\neq j} \boxed{Y=\prod M_j}$$

【例 1.3.1】 将式（1.2.3）和式（1.2.4）化为标准式 1 和标准式 2。

解 标准式 1 $S_i=m_1+m_2+m_4+m_7=\sum m_i\ (i=1,\ 2,\ 4,\ 7)$

$$C_i=m_3+m_5+m_6+m_7=\sum m_i\ (i=3,\ 5,\ 6,\ 7)$$

标准式 2 $S_i=M_0\cdot M_3\cdot M_5\cdot M_6=\prod M_j\ (j=0,\ 3,\ 5,\ 6)$

$$C_i=M_0\cdot M_1\cdot M_2\cdot M_4=\prod M_j\ (j=0,\ 1,\ 2,\ 4)$$

标准式 1 比标准式 2 用途更广泛，特别是可编程逻辑器件几乎都是采用与或阵列（即标准式 1 的结构）布局，而标准式 2 是一种或与结构的表达式。

1.3.2 逻辑函数的卡诺图表示

卡诺图是美国工程师 Karnaugh 于 20 世纪 50 年代提出的。利用它可以表示任何逻辑函数，并且可以方便地将逻辑函数化为最简与或式。n 个变量的卡诺图由 2^n 个方格组成，每个方格对应一个最小项。图 1.3.1（a）、（b）、（c）所示分别为三变量、四变量和五变量的卡诺图。二变量的逻辑函数化简很简单，不必使用卡诺图。六变量卡诺图可以参考五变量卡诺图画出，在此省略。超过六个变量的逻辑函数化简不适宜直接使用卡诺图，可以将该逻辑函数分解为多个变量数较少的逻辑函数后，再使用卡诺图化简。

三变量卡诺图的纵坐标（为方便起见，称为纵坐标）取值依次为 0 和 1，横坐标取值依次为 00，01，11，10；四变量卡诺图的坐标值可以根据三变量卡诺图推得；五变量卡诺图可以看作两个四变量卡诺图对称排列，左边四变量卡诺图的横坐标依次为 00，01，11，10，右边四变量卡诺图的横坐标依次为 10，11，01，00。然后在左边四变量卡诺图各坐标值的前面添 0，在右边四变量卡诺图各坐标值的前面添 1。这样就不必死记硬背五变量卡诺图及六变量卡诺图的坐标值了。

A \ BC	00	01	11	10
0			1	
1		1	1	1

（a）三变量卡诺图

AB \ CD	00	01	11	10
00	1			1
01		1		
11				
10	1		1	1

（b）四变量卡诺图

AB \ CDE	000	001	011	010	110	111	101	100
00	1	1					1	1
01	1			1				
11		1	1			1	1	
10	1	1					1	1

（c）五变量卡诺图

图 1.3.1 三、四、五变量卡诺图

用卡诺图表示某逻辑函数的方法是：首先将逻辑函数化为标准式 1，然后将该式所包含的最小项以“1”的形式填入对应方格中，习惯上一个方格的纵坐标在前横坐标排后，构成最

小项的对应值。例如图 1.3.1（a）就是式（1.2.4）C_i的卡诺图，图 1.3.1（b）和图 1.3.1（c）表示的逻辑函数是

$$Y_b=\sum m_i\,(i=0,\ 2,\ 5,\ 8,\ 10,\ 11)$$

$$Y_c=\sum m_i\,(i=0,\ 1,\ 4,\ 5,\ 8,\ 10,\ 16,\ 17,\ 20,\ 21,\ 25,\ 27,\ 29,\ 31)$$

1.3.3 用常用公式化简逻辑函数

【例 1.3.2】 用公式将下列逻辑函数化简为与或式。

$$Y_1=A\bar{B}+\bar{A}B+B\bar{C}+\bar{B}C$$

$$Y_2=B\bar{C}+AB\bar{C}E+\bar{B}(\overline{\bar{A}\bar{D}+AD})+B(A\bar{D}+\bar{A}D)$$

$$Y_3=AC+\bar{B}C+B\bar{D}+C\bar{D}+A(B+\bar{C})+\bar{A}BC\bar{D}+A\bar{B}DE$$

解

$$\begin{aligned}Y_1&=A\bar{B}+\bar{A}B+B\bar{C}+\bar{B}C\\&=A\bar{B}+\bar{A}B(C+\bar{C})+B\bar{C}+(A+\bar{A})\bar{B}C\\&=A\bar{B}+\bar{A}BC+\bar{A}B\bar{C}+B\bar{C}+A\bar{B}C+\bar{A}\bar{B}C\\&=(A\bar{B}+A\bar{B}C)+(B\bar{C}+\bar{A}B\bar{C})+(\bar{A}BC+\bar{A}\bar{B}C)\\&=A\bar{B}+B\bar{C}+\bar{A}C\end{aligned}$$

$$\begin{aligned}Y_2&=B\bar{C}+AB\bar{C}E+\bar{B}(\overline{\bar{A}\bar{D}+AD})+B(A\bar{D}+\bar{A}D)\\&=B\bar{C}(1+AE)+\bar{B}(\overline{A\odot D})+B(A\oplus D)\\&=B\bar{C}+(\bar{B}+B)(A\oplus D)\\&=B\bar{C}+(A\oplus D)\\&=B\bar{C}+\bar{A}D+A\bar{D}\end{aligned}$$

$$\begin{aligned}Y_3&=AC+\bar{B}C+B\bar{D}+C\bar{D}+A(B+\bar{C})+\bar{A}BC\bar{D}+A\bar{B}DE\\&=AC+\bar{B}C+A\overline{\bar{B}C}+B\bar{D}+C\bar{D}(1+\bar{A}B)+A\bar{B}DE\\&=AC+A+A\bar{B}DE+\bar{B}C+B\bar{D}+C\bar{D}\\&=A(C+1+\bar{B}CD)+\bar{B}C+B\bar{D}+C\bar{D}\\&=A+\bar{B}C+B\bar{D}\end{aligned}$$

显然用公式化简逻辑函数需要一定的技巧，而且也不知道化简出来的结果是否为最简与或式。但是用卡诺图可以有规律地化简逻辑函数，并且得到的结果一定是最简与或式。

1.3.4 用卡诺图化简逻辑函数

1. 逻辑相邻

如果两个逻辑最小项中仅有一个因子不同，则称这两个最小项逻辑相邻。在卡诺图中，下列情况能直观地反映最小项的逻辑相邻：① 相邻两格；② 同一行的最左边与最右边两格；③ 同一列的最上边与最下边两格；④ 在五、六变量卡诺图中，轴对称位置上的两格，如图 1.3.1（c）中的 m_{27}和 m_{31}；⑤ 将五、六变量卡诺图视为由多个四变量卡诺图组成，若在其中的四变量卡诺图中逻辑相邻，那么在五、六变量的卡诺图中也逻辑相邻，如图 1.3.1（c）中

的 m_8 和 m_{10}。

2. 用卡诺图化简逻辑函数的方法

（1）将逻辑函数表示在卡诺图中。

（2）合并最小项：两个逻辑相邻项合并为一项并消去那个不同的因子；四个逻辑相邻项合并为一项并消去那两个不同的因子；八个逻辑相邻项合并为一项并消去那三个不同的因子……。

（3）合并最小项的原则：① 尽可能将矩形圈画大一些，每个圈中有 2^n 个“1”；② 每个圈中至少有一个未被圈过的“1”；③ 孤立项不能化简；④ 尽可能减少总圈数。

【例 1.3.3】 用卡诺图将下列逻辑函数化简为最简与或式。

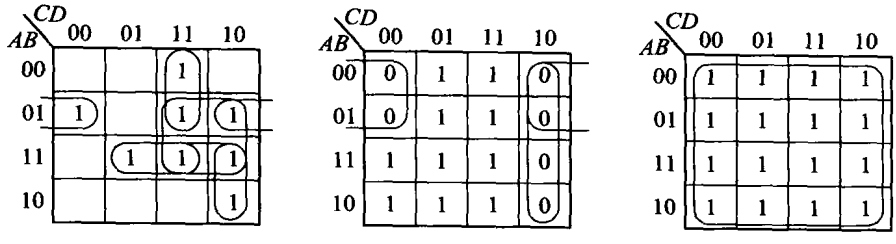
$$Y_1 = ABC + \bar{A}CD + \bar{A}B\bar{D} + ABD + A\bar{B}C\bar{D}$$

$$Y_2 = \overline{A\bar{C} + \bar{A}D + CD}$$

$$Y_3 = AC + BD + \bar{C} + \bar{A}\bar{B}C + \bar{A}BC$$

$$Y_4 = (\bar{A}\bar{B}C + \bar{A}B\bar{C} + AC) \oplus (A\bar{B}\bar{C}D + \bar{A}BC + CD)$$

说明： 如果逻辑函数本身是与或式，就不必将其化为最小项之和的形式，可以直接将逻辑函数写入卡诺图中。例如将 Y_1 中的 $\bar{A}CD$ 视为 0×11，将 ABC 视为 111×；将 Y_3 中的 AC 视为 1×1×，将 $\bar{C}$ 视为××0×。“×”表示取值为 1 和 0。在卡诺图中，含一个“×”的与项占 2 格，含两个“×”的与项占 4 格，含三个“×”的与项占 8 格……。合并最小项时，将上述方法反过来操作。例如，如图 1.3.2 所示，图 1.3.2（a）中右下角的圈中有 2 个“1”，表示为 1×10；图 1.3.2（b）中右边的圈中有 4 个“0”，表示为××10。



（a）Y_1 卡诺图　　（b）$\bar{Y}_2$ 卡诺图　　（c）Y_3 卡诺图

图 1.3.2　例 1.3.3 的 Y_1, $\bar{Y}_2$, Y_3 卡诺图

解　图 1.3.2（a）中含有 4 个“1”的那个圈是多余的，称为冗余项。它的每一个“1”都是其他圈已圈过的。

$$Y_1 \leftarrow 11\times1 + 1\times10 + 0\times11 + 01\times0 \qquad Y_1 = ABD + AC\bar{D} + \bar{A}CD + \bar{A}B\bar{D}$$

因为 $\bar{Y}_2 = A\bar{C} + A\bar{D} + CD$，作 $\bar{Y}_2$ 的卡诺图，如图 1.3.2（b）所示，用圈“0”法求 $\bar{Y}_2$ 的反函数：

$$Y_2 \leftarrow \times\times10 + 0\times\times0 \qquad Y_2 = C\bar{D} + \bar{A}\bar{D}$$

$$Y_3 \leftarrow \times\times\times\times \qquad Y_3 = 1$$

将 Y_4 视为两个函数取异或，作这两个函数的卡诺图，并将这两个卡诺图取异或得到 Y_4 的卡诺图，如图 1.3.3 所示。由图 1.3.3（c）可知