21世纪
信息与通信技术教程

数字电路与FPGA

刘常澍　赵雅兴　编著

人民邮电出版社
POSTS & TELECOM PRESS

21世纪信息与通信技术教程

数字电路与FPGA

刘常澍　赵雅兴　编著

人民邮电出版社

图书在版编目（CIP）数据

数字电路与FPGA/刘常澍，赵雅兴编著. —北京：人民邮电出版社，2004.8
21世纪信息与通信技术教程
ISBN 7-115-12522-8

Ⅰ. 数… Ⅱ. ①刘… ②赵… Ⅲ. ①数字电路—教材②可编程序逻辑器件—教材 Ⅳ. ①TN79 ②TP332.1

中国版本图书馆CIP数据核字（2004）第081279号

内 容 提 要

本书依据高等教育本科数字电路教学改革的思想编写，全书共8章，内容包括：数字逻辑基础、门电路、组合逻辑电路、触发器与波形变换、产生电路、时序逻辑电路、程序逻辑电路、CPLD与FPGA、硬件描述语言（VHDL）。本书内容在编写上注意深入浅出，注重实践，并且各章附有大量的例题、思考题和习题，供读者学习参考和练习。

本书可作为高等院校电子信息工程、通信工程、自动化、电子科学与技术等专业技术基础课教材，也可供相关科技人员参考。

21世纪信息与通信技术教程

数字电路与FPGA

◆ 编　　著　刘常澍　赵雅兴
责任编辑　王晓明

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
读者热线　010-67129258
北京汉魂图文设计有限公司制作
北京密云春雷印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：17.75
字数：424千字　　2004年8月第1版
印数：1-3 500册　　2004年8月北京第1次印刷

ISBN 7-115-12522-8/TN・2323

定价：29.00元

前　言

自从20世纪80年代以来，微电子技术和计算机技术得到了飞速发展，在电子电路设计领域，大规模、超大规模的可编程逻辑器件（CPLD和FPGA）及其软件开发工具相继推出，从而引起了数字电路设计的巨大变革。考虑到传统数字电路设计模式中，利用卡诺图变换布尔方程的设计方法和相应的中小规模集成电路堆砌技术逐步退出历史舞台的客观现实，在高等院校的电子信息工程、通信工程、自动化以及电子科学与技术等专业的本科生教学中，数字电路课程的教学应当进行改革。为此，本书对原有的传统教材内容进行了删繁就简，本着能完成CPLD与FPGA片上电路与系统的设计原则，并保持数字电路内容的完整性和理论的系统性，精练阐述数字电路的基本知识。本书所涉及的内容包括数制、码制、门电路、组合逻辑电路、触发器与波形变换、产生电路、时序逻辑电路的必要知识，并在此基础上从应用的角度阐述CPLD与FPGA的基本概念、基本知识以及软件开发工具的基本使用方法，较为详细地说明硬件描述语言 VHDL 的语法规则。为了便于理解和入门，本书还给出了大量的VHDL设计举例，特别是用VHDL对时序逻辑电路采用状态机的描述方法进行设计，大大简化了数字电路传统的设计方法。

本书由赵雅兴教授和刘常澍教授通力合作完成，由于时间较短和水平所限，书中出现的不妥之处，欢迎读者批评指正。

作　者

2004年7月于天津大学

目　录

第1章　数字逻辑基础

21世纪是信息化时代，数字化是进入信息化时代的必要条件。数字化即将信息用数字0、1编码来表述，对于任意的信息传输、处理、存储均用0、1码的形式进行。

1和0两个数码可用电路中的两种对立状态表示，如电压的高低、电流的大小，脉冲的有无等。基于这种原理的电路即是数字电路，它通过对电路中状态的变换，来对其代表的信息进行运算、处理、控制、变换等操作。这种电路表现出许多优点：

1．电路仅稳定工作在不连续的、特征差别大的两个对立状态下，因此电路的可靠性和稳定性非常高；

2．从理论上讲，信号在处理的过程中不会产生失真。

3．信息的传输、运算、处理和保存变得更为方便。

4．对电路的实时控制准确有效。

5．与计算机及外围电路兼容，便于利用计算机进行运算和控制处理。

6．数字电路可以很方便地级联和扩展，中、大规模数字集成电路的生产和应用都呈现出广阔的空间。

正是基于上述原因，数字电路在近年来得到飞速的发展，使得电子设备的可靠性和准确性得到充分实现。

数字电路的应用越来越多，已被广泛应用于工业、交通、军事、广播电视等几乎所有领域，近些年消费电子的发展正方兴未艾。

1.1　数字信号及数字电路

1.1.1　模拟量与数字量

自然界中存在的物理量千变万化，但就其变化规律而言，可以分成模拟量和数字量两大类。模拟量是在时间上和数值上连续变化的物理量，如温度、海拔高度和气压等等。数字量是时间上和数值上都是不连续的、或者说是离散的物理量，例如对生产线上产品的计件，人口统计等。

模拟量的数字化是对模拟量分离取值的过程。比如对于气温的记录统计，每间隔一定时间记录一次，只按整度数记录，最小的表示单位是“度”，而实际气温变化是连续的。所以，记录气温的过程实际是对模拟量数字化的结果。

数字量有一个最小数量单位，每次数值变化的增量或减量都是该最小数量单位的整数倍，而小于这个最小数量单位的数值没有任何意义的。比如人口统计中的最小数量单位是一人。

1.1.2 数字信号和数字电路

表示数字量的信号叫做数字信号，以数字信号方式工作的电子电路被称为数字电路。一位（1 比特，1bit）数字信号只有 0 和 1 两个数码，用电路状态表示这两个数码非常方便，如电压的高低、晶体管的饱和与截止、开关的通断等等。由于数字电路中的状态只对应 0 和 1 两个数字，因此在数字运算时采用二进制数制，与人们习惯的十进制有所不同。相对模拟电路而言，数字电路具有误差小、抗干扰性强、精度高、容易保存等优点。

1.2 二 进 制 数

1.2.1 二进制数表示法

人们在计数的过程中，采取一定的计数规则，即是数制。例如，中国人的祖先最早采用的现在称为科学计数法的数制是十进制，它有 0、1、…、9 十个数字，计数的规则是“逢十进一”，即在计数过程中，一旦计数满十，就向高位进一。可以用十的幂之和表达式来表示一个 n 位整数、m 位小数的十进制数：

$$(D)_{10}=k_{n-1}\times10^{n-1}+k_{n-2}\times10^{n-2}+\cdots+k_1\times10^1+k_0\times10^0+k_{-1}\times10^{-1}+k_{-2}\times10^{-2}+\cdots+k_{-m}\times10^{-m}$$

$$=\sum_{i=-m}^{n-1}k_i\times10^i \tag{1-1}$$

式中 n、m、k、i 都是自然数，k_i 为第 i 位的系数，10 为基数，10^i 为第 i 位的权。不同数值的数，主要体现出权的大小，如十位、百位、千位等等；其次是系数不同。任意一个十进制数可按位展开，即把每一位的位权值与各自的系数相乘，然后对所有项求和。如：

$$(7654.328)_{10}=7\times10^3+6\times10^2+5\times10^1+4\times10^0+3\times10^{-1}+2\times10^{-2}+8\times10^{-3}$$

除了十进制外，人们还用其他进制，如十二进制、六十进制等。按照上述方法，可以写出任意进制数的表达式：

$$(D)_R=k_{n-1}\times R^{n-1}+k_{n-2}\times N^{n-2}+\cdots+k_1\times R^1+k_0\times R^0+k_{-1}\times R^{-1}+k_{-2}\times R^{-2}+\cdots+k_{-m}\times R^{-m}$$

$$=\sum_{i=-m}^{n-1}k_i\times R^i \tag{1-2}$$

上式可表示为一个 R 进制的数，其系数为 0、1、…、$R-1$。

若 $R=2$，则是二进制数，即采用“基数是 2，逢二进一，系数只有 0 和 1 两个数字”的计数方法，称为自然二进制数，可以用下式表示：

$$(D)_2=\sum_{i=-m}^{n-1}k_i\times2^i \tag{1-3}$$

之所以特殊提出二进制数，是因为在数字电路（计算机由数字电路组成）中，采用二进

制数容易实现对数字信号（数值、代码）的处理、运算、传输、存储。但二进制数通常位数很多，且与人们习惯的计数方法不同，因而有时需要进行二进制数与其它进制数相互转换，以适应不同的应用。

1.2.2　二进制数和十进制数的互相转换

1．二—十进制数转换

把一个二进制数转换成十进制数的过程很简单，只需将二进制数按位（权）展开相加即可。

例 1-1　将二进制数（1101001.011）$_2$转换成十进制数。

解：（1101001.011）$_2$=

$$1\times2^6+1\times2^5+0\times2^4+1\times2^3+0\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}+1\times2^{-3}=(105.375)_{10}$$

2．十—二进制数转换

十进制数转换到二进制数的过程须分两步走，对整数部分和小数部分分别进行转换，然后再相加得到结果。

整数部分的转换过程如下：

一个十进制的整数$(D)_{10}$总可以用二进制数展开，即：

$$(D)_{10}=k_n2^n+k_{n-1}2^{n-1}+\cdots+k_12^1+k_02^0$$

其二进制数写为$(k_nk_{n-1}\cdots k_0)_2$。若把$(D)_{10}$除以 2，则得到的商为：

$$k_n2^{n-1}+k_{n-1}2^{n-2}+\cdots+k_1$$

而余数即 k_0；然后，再把商数除以 2，则所得余数即 k_1；…。将每次得到的商反复除以 2，取它们的余数 $k_0\cdots k_n$，就可求出二进制数的每一个数位了。所以，十进制数（整数部分）转换为二进制数的过程是采用逐次除以基数 2、再取余数的方法。

例 1-2　将十进制数（57）$_{10}$转换成二进制数。

解：按上述步骤进行连除运算如图 1-1，得：

$$(57)_{10}=(111001)_2$$

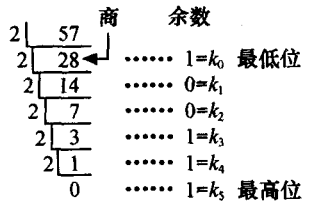

图 1-1 十—二进制数整数的转换

小数部分的转换与整数部分的转换不同，是通过基数乘法实现的。

若$(D)_{10}$是一个十进制数的小数，对应的二进制小数为$(0.k_{-1}k_{-2}\cdots k_{-m})_2$，应有：

$$(D)_{10}=k_{-1}2^{-1}+k_{-2}2^{-2}+\cdots+k_{-m}2^{-m}$$

如果将两边同乘以 2 得到：

$$2(D)_{10}=k_{-1}+(k_{-2}2^{-1}+k_{-3}2^{-2}+\cdots+k_{-m}2^{-m+1})$$

说明将小数$(D)_{10}$乘以 2 后，等式右边所得乘积的整数部分即 k_{-1}，其余为小数部分。

取出 k_{-1}，将小数部分再乘以 2 又可以得到：

$$k_{-2}+(k_{-3}2^{-1}+\cdots+k_{-m}2^{-m+1})$$

同样取出 k_{-2}，将小数部分再乘以 2，…当乘积为 0 后，就依次得到 $k_{-1}k_{-2}\cdots k_{-m}$。所以，十进制数（小数部分）转换为二进制数的过程是采用逐次乘以基数 2、再取整数的方法。

例 1-3　将十进制小数（0.8125）$_{10}$转换成二进制小数。

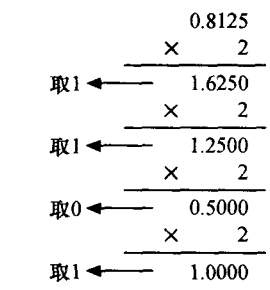

图 1-2 十一二进制数小数的转换

解：按上述步骤进行连乘运算，如图 1-2 所示，得：

$$(0.8125)_{10}=(0.1101)_2$$

例 1-3 所转换的数在进行 4 次乘以 2 后，小数部分已经是 0 了，则转换结束。

然而有的小数在多次乘以 2 后，小数部分总是不能为 0，使得二进制数的位数很多，有的形成了循环，在这种情况下，计算达到一定的精度，即二进制数的小数达到一定的位数，就不再继续乘以 2，舍去后面的尾数。

例 1-4 将（0.4）$_{10}$转换成二进制小数。

解：按上述步骤进行连乘运算，如图 1-3 所示。

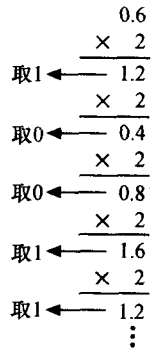

图 1-3 小数转换出现循环

连续 3 次乘以 2 后再乘以 2，小数部分出现了开始的数值，可以推断，再继续进行则进入了小数的循环，因而得：

$(0.6)_{10}=(0.100110011001\cdots)_2$

根据精度取小数点后所需位数即可。

1.2.3 八进制数、十六进制数

用二进制数表示一个数字位数会很长，书写和记忆都很不方便，因而可以把二进制数化成八进制或十六进制数。

八进制数有 0、1、…、7 八个数字组成，根据数制规律可知，一位八进制数可表示成三位二进制数，因而转换时只需将二进制数的每三位对应转换成一位八进制数。当然转换带小数的数时，整数与小数应分别进行转换。即以小数点为基准，向左、向右每三位为一组（首尾不足三位的补足三位），每组对应转换成一位八进制数。

例 1-5 将（10110.01011）$_2$转换成八进制数。

解：如图 1-4 所示，以小数点为界，向左每三位分为一组，若最左一组不足三位，则用 0 补足三位，写出对应的八进制数的整数部分；从小数点向右每三位分为一组，若最右一组不足三位，则用 0 补足三位写出对应的八进制数的小数部分。得到：

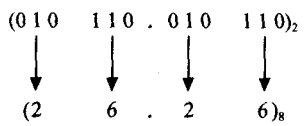

图 1-4 二一八进制数转换

$$(10110.01011)_2=(26.26)_8$$

十六进制数由 0、1、…、9 和 A、B、C、D、E、F 共十六个数字组成，其中 A、…、F 分别等值于十进制数中的 10、…、15 等数字。四位二进制数可转换为一位十六进制数，转换方法可参照八进制数的转换方法进行，但须将二进制数的每四位对应转换成一位十六进制数。

例 1-6 将（1011010.01111）$_2$转换成十六进制数。

解：如图 1-5 所示，以小数点为界，向左每四位分为一组，若最左一组不足四位，则用 0 补足四位，写出对应的八进制数的整数部分；从小数点向右每四位分为一组，若最右一组不足四位，则用 0 补足四位写出对应的十六进制数的小数部分。得到：

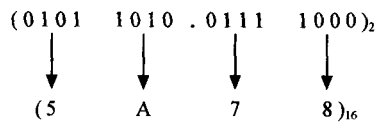

图 1-5 二一十六进制数的转换

$$(1011010.01111)=(5A.78)_{16}$$

欲将八进制数和十六进制数转换成十进制数，只需仿照

二进制数转换的方法，按位权展开求和即可。

例 1-7　将$(7B.E4)_{16}$转换成十进制数。

解：$(7B.E4)_{16}=7\times16^{1}+11\times16^{0}+14\times16^{-1}+4\times16^{-2}=(123.880625)_{10}$

十进制数直接转换成八进制和十六进制数比较繁琐和不习惯，有些计算容易出错。一般是先化成二进制数，然后再变成八进制和十六进制数。

例 1-8　将$(43.3125)_{10}$转换成八进制和十六进制数。

解：$(57.8125)_{10}=(111001.1101)_{2}=(71.64)_{8}=(39.D)_{16}$

1.3　码制与编码

数字系统和计算机可以存储和处理各种各样的信息，而实际上机器只能识别二进制数，因而要用二进制数来表示不同的信息。用数来表示信息的方法很多，建立这种对应关系的过程就叫做编码，编码所依据的编码规则就称为码制。所以，数码不仅可以表示数值的大小，而且还能表示不同的事物。表示事物的这些数码只是事物的代号，而没有数值的含意。例如，不同列车的行车编号，只是区别不同的行车路线，并不表示数值大小。代替不同事物的数码就叫代码。当然，数值也可以用数码来表示，但可以有不同的计数规则，甚至同一个数因为依据的码制不同可以有不同的代码。

1.3.1　原码、反码和补码

书写一个带符号的数时，可以在数的前面加上一个符号，如+3、–0.5 等等。而计算机当中的正负号是用数码来表示的。通常带符号数的最高位为符号位：　0 表示为符号为正；　1 则表示符号为负。符号位后面的代码表示的是数值。在数字电路中，带符号数通常有原码、反码和补码三种表示方法。

1．原码

用原码表示带符号数时，只需将符号位用 0 或 1 表示即可，后面的数码不变。

例 1-9　用原码表示两个带符号的数 $N_1=+1100101$ 和 $N_2=-1100101$。

解：　$N_1=(01100101)_{原}$　　$N_2=(11100101)_{原}$

原码表示方法简单，但在做运算时比较麻烦（尤其是减法运算），运算过程加长，造成电路成本增加和运算速度降低。为此，采用补码表示方法来加以改进。

2．反码

用反码表示一个数时，正数的表示方法与原码表示方法相同；如果是负数，最高位仍为符号位，记为 1，其余各位把原数值按位取反即可。

例 1-10　用反码表示两个带符号的数 $N_1=+1100101$ 和 $N_2=-1100101$。

解：　$N_1=(01100101)_{反}$　　$N_2=(10011010)_{反}$

3．补码

在补码表示中，正数的表示方法也与原码表示方法相同；如果是负数，符号位仍为 1，其余各位不用数值本身，而取对 2^N 的补数。由于是二进制数，求负数的补码可先将其变成反

码，然后在最低位加 1 即可。

例 1-11 用反码表示两个带符号的数 N_1= + 1100101 和 N_2= −1100101。

解： N_1=(01100101)$_{补}$

N_2 =(10011010)$_{反}$=(10011011)$_{补}$

带符号三位二进制数的三种表示方法如表 1-1 所示。

表 1-1　　原码、反码和补码的对应关系

十进制	二 进 制			十进制	二 进 制		
	原码	反码	补码		原码	反码	补码
−8	−	−	1000	−0	1000	1111	−
−7	1111	1000	1001	0	0000	0000	0000
−6	1110	1001	1010	1	0001	0001	0001
−5	1101	1010	1011	2	0010	0010	0010
−4	1100	1011	1100	3	0011	0011	0011
−3	1011	1100	1101	4	0100	0100	0100
−2	1010	1101	1110	5	0101	0101	0101
−1	1001	1110	1111	6	0110	0110	0110
				7	0111	0111	0111

注：原码、反码处的空格为 5 位码；补码处的空格为无编码

原码表示方法简单，符合人们通常的记数习惯，但 0 的表示不是唯一的。补码的转换虽然稍微麻烦一些，但是 0 的表示方法是唯一的，而且补码减法运算可以用加法来实现。

4．小数的表示与字长

不论原码、反码或补码，都是按进位关系和整数连续排列起来，但是当两个数进行运算时，必须整数和小数字长分别相等，即两个数的整数位数应相同，小数的位数也应相同。如果两个数的位数不同或在运算时产生的进位超出了原来的位数（称为溢出），就应增加位数少的数的字长（扩展位数）。

扩展位数应整数、小数分别扩展，扩展的方法如下。

（1）正数：无论带符号和不带符号的原码、反码和补码表示的正整数，一律在高位填 0 补足所少的位数；正小数，一律在低位填 0 补足所少的位数；

（2）原码带符号的负数：整数在符号位后的高位填 0 补足所少的位数，小数在低位填 0 补足所少的位数；

（3）反码带符号的负数：整数在符号位后的高位填 1 补足所少的位数，小数在低位填 1 补足所少的位数；

（4）补码带符号的负数：整数在符号位后的高位填 1 补足所少的位数，小数在低位填 0 补足所少的位数；

例如，7.5 和 4.6875 的正、负数各种表示由原来的 4 位字长扩展为 8 位字长：

正数：7.5—0111.1000，4.6875—0100.1011 扩展后为

7.5—00000111.10000000，4.6875—00000100.10110000

负数原码：−7.5—1111.1000，−4.6875—1100.1011 扩展后为

−7.5—10000111.10000000，−4.6875—10000100.10110000

负数反码：−7.5—1000.0111，−4.6875—1011.0100 扩展后为

−7.5—11111000.01111111，−4.6875—11111011.01001111

负数补码：−7.5—1001.1000，−4.6875—1011.0101 扩展后为

−7.5—11111000.10000000，−4.6875—11111100.01010000

例 1-12　有两个数 N_1 =+1011000 和 N_2=+0100110，计算 N_1−N_2 的值。

解：　N_1−N_2

$=N_1+(-N_2)$

$=N_{1补}+(-N_2)_{补}$

=01011000+11011010

用竖式表示

$$\begin{array}{r} 01011000 \\ +\ 11011010 \\ \hline \boxed{1}00110010 \end{array}$$

舍去符号位前的溢出 1，则 N_1−N_2=+011010

例 1-13　有两个数：N_1 = +0100110 和 N_2= +1010010，计算 N_1−N_2 的值。

解：　N_1−N_2

$=N_1+(-N_2)$

$=N_{1补}+(-N_2)_{补}$

=00100110+10101110

用竖式表示

$$\begin{array}{r} 00100110 \\ +\ 10101110 \\ \hline 11001100 \end{array}$$

得到的结果符号位为 1，则 N_1−N_2= −1001100

1.3.2　二—十进制（BCD）码

人们通常熟悉的是十进制数，而数字电路中采用的是二进制的计数方法，为了用二进制数表示十进制数，则采用 BCD（Binary Coded Decimal）码的表示方法。一位十进制数最少要四位二进制数来表示，即可以从 0000、…、1111 中任取十个数来表示 0、1、…、9。通常的一种编码方法是取前十个数，称为 8421BCD 码，其对应关系见表 1-2。它是一种含权码，即将二进制数按位相加就可得到相应的十进制数。这种码表示意义明显，转换也很方便，因而应用最多。除了 8421 码外，还有几种 BCD 码，分别介绍一下。

含权码还有 2421 码、5211 码等，但与 8421 码不同，它们的编码方式不是唯一的。如 2421 码中的 7 既可以表示成 1101，也可以表示成 0111。5211 码中 7 可以表示成 1100 或 0111。

余 3 码是一种无权码，将 8421 码表示的数码加 3 即得到相应的余 3 码。用余 3 码作加法时，若两个十进制数之和为 10 应产生进位，而两个对应余 3 码之和恰好等于二进制数的 16，

于是向高位进位而不再进行修正。另外,0 和 9、1 和 8、2 和 7、3 和 6、4 和 5 互为反码（2421 码也有相似的特性），这对于求取对 10 的补码运算非常方便。

表 1-2 列出了几种常用的 BCD 代码。

1.3.3 格雷(Gray)码

格雷码是一种无权码，也称为循环码。其特点是：每两个相邻代码中的数码仅有一位不同，其余各位均相同。而且首尾（0 和 15）两个代码也仅有一位不同，构成“循环”。显然，在数码变化时采用格雷码可大大减少错码的可能性。表 1-3 列出了 4 位格雷码与十进制数的对应关系。

表 1-2　　几种常用的 BCD 代码

十进制数	8421	2421	5211	余 3 码
0	0000	0000	0000	0011
1	0001	0001	0001	0100
2	0010	0010	0100	0101
3	0011	0011	0101	0110
4	0100	0100	0111	0111
5	0101	1011	1000	1000
6	0110	1100	1001	1001
7	0111	1101	1110	1010
8	1000	1110	1101	1011
9	1001	1111	1111	1100

表 1-3　　格雷码数码表

十进制数	格雷码	十进制数	格雷码
0	0000	8	1100
1	0001	9	1101
2	0011	10	1111
3	0010	11	1110
4	0110	12	1010
5	0111	13	1011
6	0101	14	1001
7	0100	15	1000

1.4 逻辑代数基本知识

逻辑代数最初是由英国数学家布尔（G. Boole）首先提出来的，被称为布尔代数。后来香农（Shannon）将布尔代数用到开关矩阵电路中，因而又称为开关代数。现在逻辑代数被广泛用于数字逻辑电路和计算机电路的分析和设计中，成为数字逻辑电路的理论基础。逻辑代数的变量称为逻辑变量。逻辑变量与一般代数变量不同，逻辑变量的取值只有 0 和 1，就是说逻辑电路中只有两种逻辑状态。这里的 1 和 0 可以由数字系统中的电平的高低、开关的断通和信号的有无来表示。因而，它们已没有数量大小的概念，只表示两种不同的逻辑状态。

1.4.1 基本逻辑运算

逻辑运算中，最基本的运算是与、或、非三种。

1．与逻辑

图 1-6 中给出的灯控制电路可以看出：只有当两个开关同时闭合时，灯才会亮，否则，灯就不亮。从而可以得出这样的因果关系：只有当决定事物某一结果的全部条件同时具备时，这个结果才会发生，这种因果关系叫做逻辑与，或者叫逻辑乘。这种逻辑关系表达式为：