


数字电路分析与设计

jichengdianzijishu@163.com 2018start

电子技术



- 模拟电子技术
- 数字电子技术

数字电子技术

- 1.分析方法：真值表、卡诺图、逻辑表达式、特征方程、波形图、状态转换图
- 2.器件：基本门电路、组合逻辑单元基本电路、时序逻辑单元基本电路（触发器）
- 3.基本数字部件电路：集成组合逻辑电路、集成时序逻辑电路、大规模数字集成电路
- 4.模-数（A/D）和数-模（D/A）转换
- 5.函数信号发生电路(555电路)

总成绩=平时50%（作业20%+实验
操作及报告30%）+期末试卷50%
（试卷中包含有实验测试题10—15分）

参考书：面向21世纪课程教材

1、 电子技术基础

数字部分（第五版）

华中科技大学康华光主编

2、 数字电子技术基础

第五版

清华大学阎石主编

数字电子电路

第1章 数字电路与系统基本概念

1.1 数字信号和数字电路

1.2 数字电路中的数制

1.3 数字电路中的代码

1.4 数字电路中的基本逻辑函数

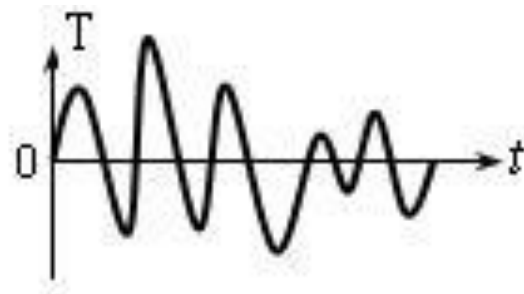
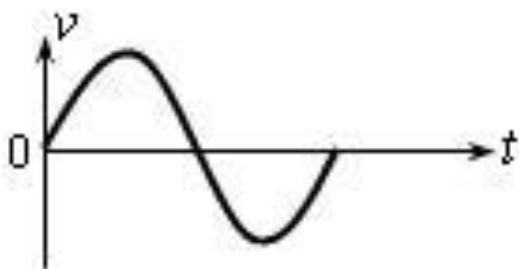
1.5 逻辑代数

1.1 数字信号和数字电路

模拟电子技术基础 和 数字电子技术基础

一、模拟信号和数字信号

模拟信号是时间上和数值上都连续的物理量



(a) 正弦电压波形

(b) 锯齿电压波形

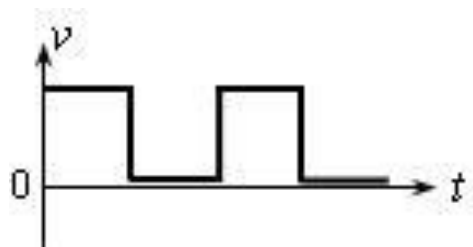
(c) 随时间温度变化波形

模拟信号其特征表现为随着时间的变化，幅度（大小）是连续变化的，没有突变或跳跃

数字信号特征是它的幅度（大小）随时间变化是不连续的，是断续的，时有时无的。

例如用一个电子电路记录从自动生产线上输出的零件数目，每送出一个零件就给电子电路一个信号，使之记1，没有零件送出时记0，零件数目这个信号的变化在时间上和数量上都不连续，是数字信号。

数字信号在时间上和数值上都是离散的



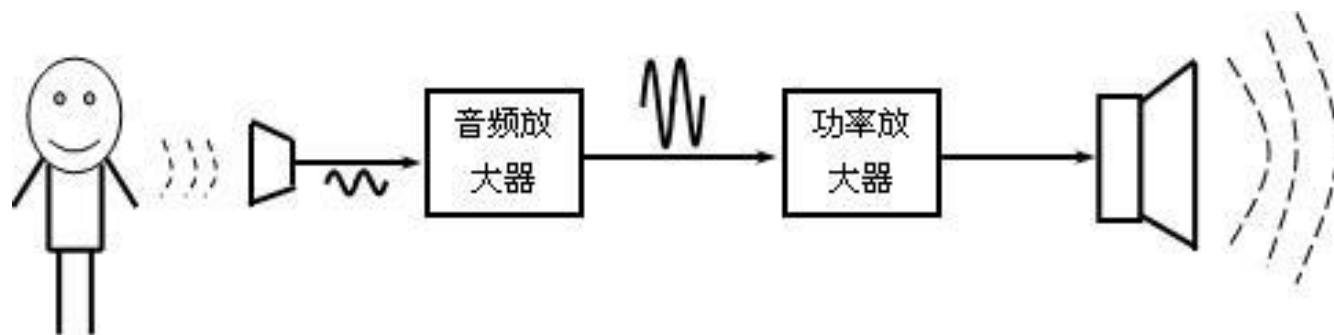
正逻辑体制：

规定高电平为逻辑1，低电平为逻辑0

典型的数字信号波形

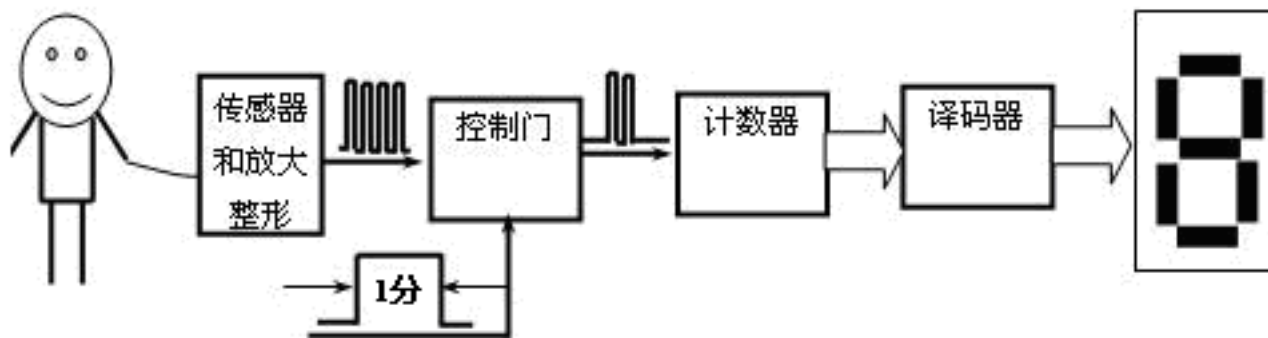
二、模拟电路和数字电路

模拟电路主要研究模拟信号的放大和处理；



音频功率放大电路框图

模拟电路中的器件主要工作在放大区，常用的分析方法是微变等效电路法。



检测人体心率的电路

数字电路中，着重研究输出信号与输入信号之间的逻辑关系，它常用能代表二种截然不同的状态或因果之间的关系来表示。如：来与去，有与无，高与低，开与关，亮与暗等等。所以，在电信号中，用电平的高低来区分。电路中的器件工作在饱和区和截止区。常用的分析方法是逻辑代数。

数字电路的特点：

- 1.抗干扰能力强
- 2.信号容易存储在电路中
- 3.便于计算机处理

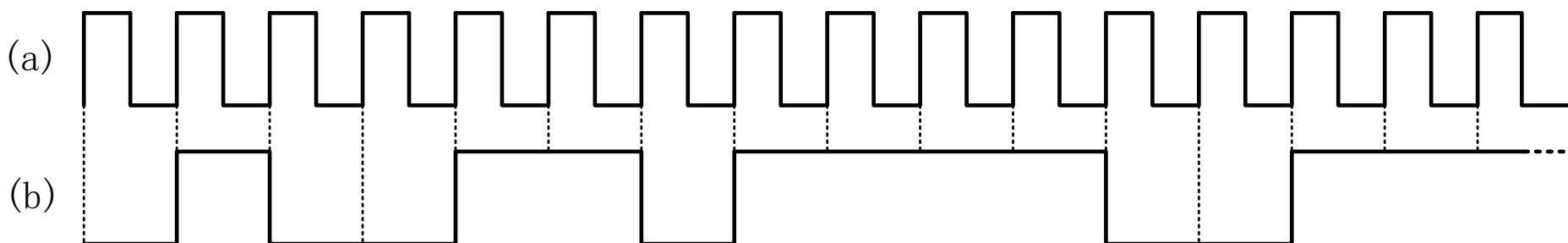
三、数字信号的描述方法

1、二值逻辑

数字信号只有两个离散值——高电平和低电平，是一种二值信号，常用数字0和数字1分别表示低电平和高电平。

2、数字波形

数字波形是逻辑电平对时间的图形表示。

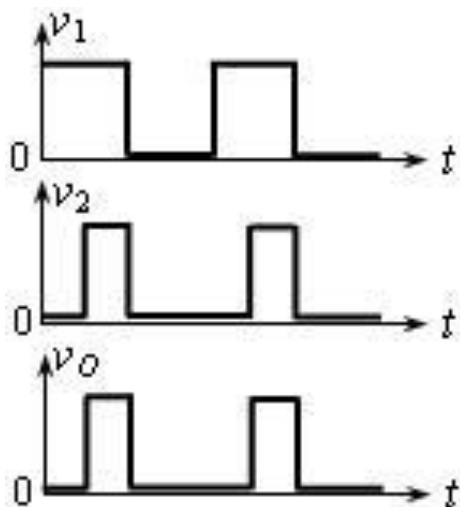


在(a)图所示的时钟脉冲控制下，(b)图所包含的信息是**0100110111100111**。

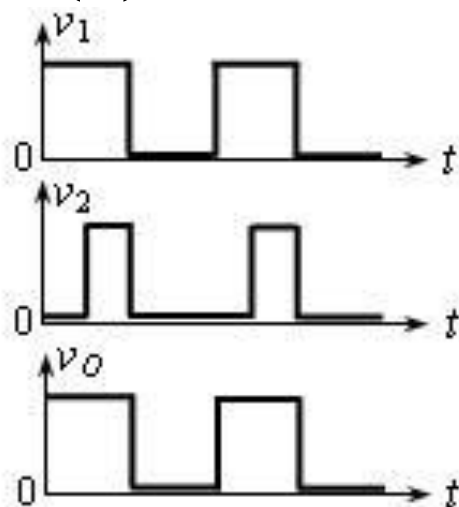
四、数字信号的处理和传输

简单的数字信号处理

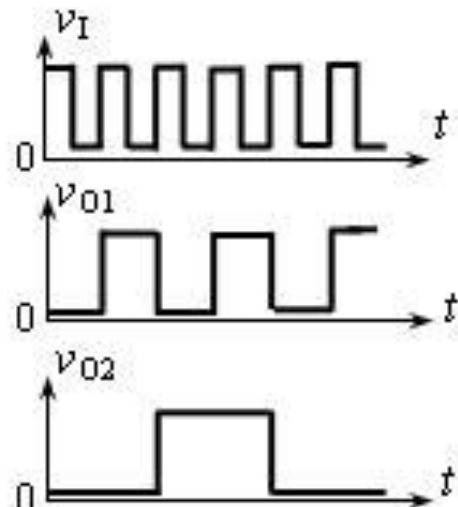
(a) “与” 逻辑处理



(b) “或” 逻辑处理



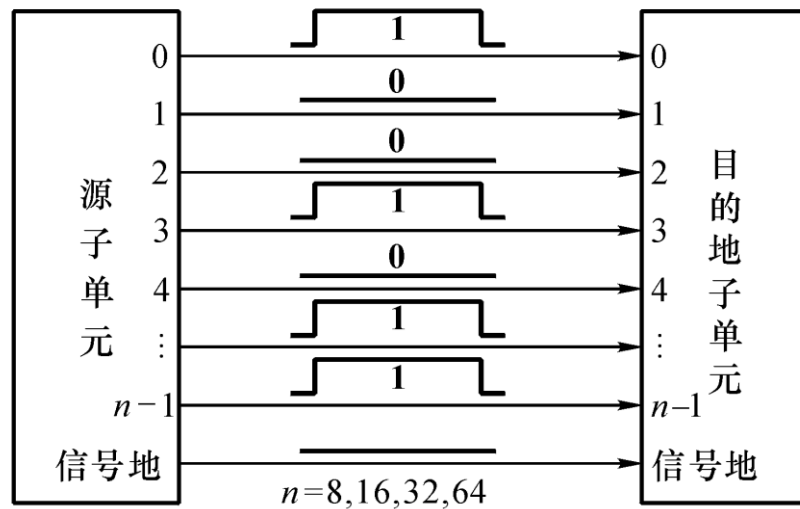
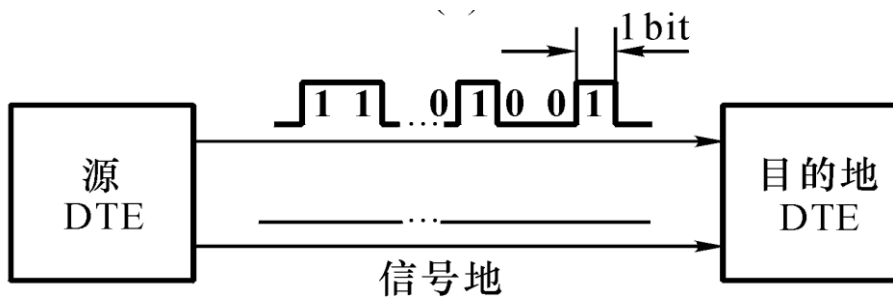
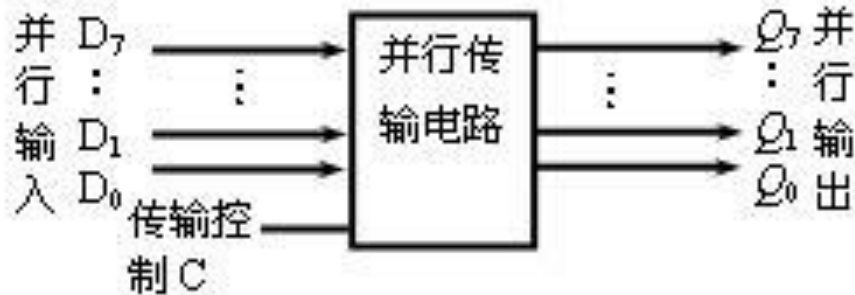
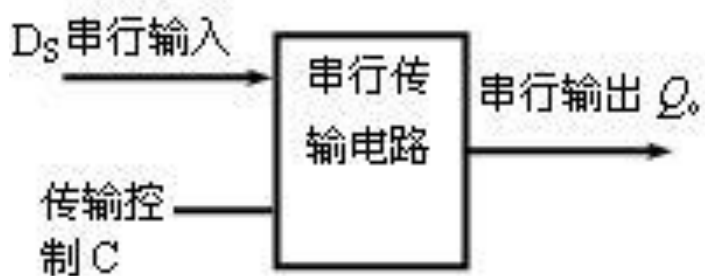
(c) 分频处理



数字信号的两种传输方式

(a) 串行传输

(b) 并行传输



串行传输方式

并行传输方式

1.2 数字电路中的数制

一、数制及相互间的转换

1. 计数体制

每一位的构成方法以及从低位到高位进位的规则

➤ 十进制数

$$475.6 = 4 \times 10^2 + 7 \times 10^1 + 5 \times 10^0 + 6 \times 10^{-1}$$

“数码”：0、1、...、9

“基数”：10，逢十进一

“权”表示价值 可表示成 $(475.6)_{10}$ 或 **475.6D**

10^2 —— 百位的“权”

10^1 —— 拾位的“权”

10^0 —— 个位的“权”

10^{-1} —— 拾分之一位的“权”

➤ **r进制数的通式:** $K_{n-1}K_{n-2}\dots K_2K_1K_0.K_{-1}K_{-2}\dots K_{-m}$

$$(N)_r = K_{n-1} \cdot r^{n-1} + K_{n-2} \cdot r^{n-2} + \dots + K_2 \cdot r^2 + K_1 \cdot r^1 + K_0 \cdot r^0 \\ + K_{-1} \cdot r^{-1} + K_{-2} \cdot r^{-2} + \dots + K_{-m} \cdot r^{-m}$$

r ——r进制数的基数，数码有**r**个

第**i**位的权: r^i

进位规则: 逢**r**进1

K_i ——某数中第**i**位的数码元素

n ——该数整数部分的位数

m ——小数部分的位数

$$(N)_r = \sum_{i=-m}^{n-1} K_i \cdot (r)^i$$

➤ 二进制数(binary Number)

基数 $r=2$ ，逢二进一

只有**0**和**1**二个数码元素 $K_{n-1}K_{n-2}\dots K_2K_1K_0.K_{-1}K_{-2}\dots K_{-m}$

$$(N)_r = K_{n-1} \cdot r^{n-1} + K_{n-2} \cdot r^{n-2} + \dots + K_2 \cdot r^2 + K_1 \cdot r^1 + K_0 \cdot r^0 \\ + K_{-1} \cdot r^{-1} + K_{-2} \cdot r^{-2} + \dots + K_{-m} \cdot r^{-m}$$

$$(1101.001)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ + 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$$

也可表示成**1101.001B**

二进制数从高位至低位的“**位权**”依次是：

2^{n-1} 、 2^{n-2} 、 \dots 、 2^0 、 2^{-1} 、 \dots 、 2^{-m} 。 ■

➤ 八进制数(octal Number)

基数 $r=8$ ，逢八进一

八个数码元素为0、1、...7

$(357.61)_8$ 或 357.61O

$$= 3 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 + 6 \times 8^{-1} + 1 \times 8^{-2}$$

从高位至低位的“位权”依次是：

8^{n-1} 、 8^{n-2} 、...、 8^0 、 8^{-1} 、...、 8^{-m} ■

➤十六进制数(hexadecimal Number)

十六个数码元素为 0、1、...9、
A、B、C、D、E、F

基数 $r=16$ ，逢十六进一

$(A8D.C6)_{16}$ 或A8D.C6H

$$=A \times 16^2 + 8 \times 16^1 + D \times 16^0 + C \times 16^{-1} + 6 \times 16^{-2}$$

从高位至低位的“位权”依次是：

16^{n-1} 、 16^{n-2} 、...、 16^0 、 16^{-1} 、...、 16^{-m}

➤ 几种常见数制间的关系

十进	二进	八进	十六进
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7

十进	二进	八进	十六进
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

2. 各种进制数间的相互转换

原因：数字电路运行在二值的二进制数字信号下，但为书写方便，常用八进和十六进制数表示，而日常又习惯于十进制数，所以要进行数制间的转换。

数制转换包括：

- r 进制数转换为十进制数
- 十进制数转换为 r 进制数
- 二、八、十六进制数之间的相互转换

➤ r 进制数转换成十进制数

按权展开再相加

$$\begin{aligned}(1101.001)_{\mathbf{2}} &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &\quad + 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ &= (13.125)_{\mathbf{10}}\end{aligned}$$

$$\begin{aligned}(357.6)_{\mathbf{8}} &= 3 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 + 6 \times 8^{-1} \\ &= (239.75)_{\mathbf{10}}\end{aligned}$$

$$\begin{aligned}(\text{A8D.C})_{\mathbf{16}} &= \text{A} \times 16^2 + 8 \times 16^1 + \text{D} \times 16^0 + \text{C} \times 16^{-1} \\ &= (2701.75)_{\mathbf{10}}\end{aligned}$$

➤ 十进制数转换为r 进制数

整数部分的转换采用**除r取余法**。将待转换的十进制数整数除以r，取余数，不断地进行，直至商为零。第一次的余数为r进制数的最低位(LSB)，最后的余数为转换后进制数的最高位(MSB)。

以十进制数转换成二进制数为例：

$$\begin{aligned}(N)_{10} &= (K_{n-1}K_{n-2}\dots K_2K_1K_0)_2 \\&= K_{n-1} \times 2^{n-1} + K_{n-2} \times 2^{n-2} + \dots + K_2 \times 2^2 + K_1 \times 2^1 + K_0 \times 2^0 \\&= 2\{ \underbrace{K_{n-1} \times 2^{n-2} + K_{n-2} \times 2^{n-3} + \dots + K_2 \times 2^1 + K_1 \times 2^0}_{\text{商}} \} + \overbrace{K_0}^{\text{余数}} \\&= 2\{ 2[\underbrace{K_{n-1} \times 2^{n-3} + \dots + K_2 \times 2^0}_{\text{商}}] + \overbrace{K_1}^{\text{余数}} \} + K_0 \\&\vdots \\&= 2\{ 2[\dots 2(\underbrace{0}_{\text{商}}) + \overbrace{K_{n-1}}^{\text{余数}}] + K_1 \} + K_0\end{aligned}$$

【例1.1】

将十进制数175转换成二进制，八进制和十六进制数。

解：

2	175...	$K_0=1$	(LSB)	8	175...7		16	175.....	$K_0=F$
2	87...	$K_1=1$		8	21...5		16	10.....	$K_1=A$
2	43...	$K_2=1$		8	2...2			0	
2	21...	$K_3=1$			0				
2	10...	$K_4=0$							
2	5...	$K_5=1$							
2	2...	$K_6=0$							
2	1...	$K_7=1$	(MSB)						
	0								

LSB—Least Significant Bit
MSB—Most Significant Bit

结果 $(175)_{10} = (10101111)_2 = (257)_8 = (AF)_{16}$ ■

• 小数部分的转换

一个十进制小数可用二进制数表示如下：

$$(N)_{10} = (0.K_{-1} K_{-2} \dots K_{-m+1} K_{-m})_2$$

$$= K_{-1} \times 2^{-1} + K_{-2} \times 2^{-2} + \dots + K_{-m+1} \times 2^{-m+1} + K_{-m} \times 2^{-m}$$

$$2 \times (N)_{10} = \underbrace{K_{-1}}_{\text{整数部分}} + \underbrace{K_{-2} \times 2^{-1} + \dots + K_{-m+1} \times 2^{-m+2} + K_{-m} \times 2^{-m+1}}_{\text{小数部分}}$$

$$2 \times (N)_{10} - K_{-1} = K_{-2} \times 2^{-1} + \dots + K_{-m+1} \times 2^{-m+2} + K_{-m} \times 2^{-m+1}$$

$$2[2 \times (N)_{10} - K_{-1}] = \underbrace{K_{-2}}_{\text{整数部分}} + \underbrace{\dots + K_{-m+1} \times 2^{-m+3} + K_{-m} \times 2^{-m+2}}_{\text{小数部分}}$$

依次类推，可得 K_{-1} 、 K_{-2} 、 \dots 、 K_{-m}

十进制数转换为 r 进制数时，**小数部分**的转换采用**乘 r 取整法**。将待转换的十进制小数乘以 r ，取整数，再将积的小数部分乘以 r ，不断地进行，直至积的小数部分为零。第一次的整数为转换后的最高位(MSB)，最后一次的整数为最低位(LSB)。

必须注意：

有时积的小数部分会达不到零，这时候，可按转换精度的要求来取位数。

【例1.2】

将十进制数小数 $(0.125)_{10}$ 转换成等值的二进制数、八进制数和十六进制数。

解：

	$\begin{array}{r} 0.125 \\ \times 2 \\ \hline \end{array}$		$\begin{array}{r} 0.125 \\ \times 8 \\ \hline \end{array}$	
(MSB)	$\boxed{0}.250 \quad K_0=0$		$1.0 \quad K_0=1$	
	$\begin{array}{r} \times 2 \\ \hline \end{array}$			
	$\boxed{0}.50 \quad K_1=0$		0.125	
	$\times 2$		$\times 16$	
(LSB)	$\boxed{1}.0 \quad K_2=1$		$2.0 \quad K_0=2$	

结果 $(0.125)_{10}=(0.001)_2=(0.1)_8=(0.2)_{16}$

➤ 2进制、8进制以及16进制数之间的转换

用二进制数作为桥梁

因为 $2^3=8$ ，所以一个八进制数码元素用一组三位二进制数表示。
因为 $2^4=16$ ，所以一个十六进制数码元素用一组四位二进制数表示。

$$(57)_8 = (\underline{101}\underline{111})_2$$

$$(3A.4)_{16} = (\underline{0011}\underline{1010}.\underline{0100})_2$$

$$(101111)_2 = (\underline{00}\underline{101111})_2 = (2F)_{16}$$

$$(110011.01)_2 = (\underline{1100}\underline{11}.\underline{01}\underline{0})_2 = (63.2)_8$$

$$(451)_8 = (\underline{000}\underline{100}\underline{101}\underline{001})_2 = (129)_{16}$$

4、数字电路中的正负数表示

数字电路只认识二进制数，所以正负数肯定也用二进制数表示。其方法是在一个数的最高位前设置一位**符号位**。

符号位为“**0**”时，表示该数为**正数**，符号位为“**1**”时为**负数**。

这种带符号位的数称为**机器数**，原正负数又称**真值**。

一个机器数的表示形式有三种：**原码**，**反码**和**补码**。

➤ 原码 (True Form) ■

由符号位加原数的数值部分，即

$$[X]_{\text{原}} = \text{符号位} + \text{原数值}$$

如 $x_1 = +1001010$ 则 $[x_1]_{\text{原}} = 01001010$ ■

$x_2 = -1001010$ 则 $[x_2]_{\text{原}} = 11001010$

特点：

原码表示简单，直观。适用于两数相乘，因为乘积的符号位只要将两乘数符号位异或即可。但减法运算的符号位较难求出。

➤ 反码 (One's Complement) ■

正数的反码为符号位加上原数值部分，负数的反码为符号位加上原数值的反码(原数值按位求反)。 ■

$$\begin{aligned} [x]_{\text{反}} &= \text{符号位} + \text{原数值} && (x \text{ 为 正数 时}) \blacksquare \\ &= \text{符号位} + \text{原数值按位求反} && (x \text{ 为 负数 时}) \end{aligned}$$

■

例 $x_1 = +1001010$ 则 $[x_1]_{\text{反}} = 01001010$ ■

$x_2 = -1001010$ 则 $[x_2]_{\text{反}} = 10110101$

➤ 补码 (Two's Complement)

补码（补数）可以从生活中来认识。如早晨7:00起床时，发现时钟停在10:00上。要校准到7点，有二种方法：

a. 顺拨时钟9个小时，相当于 $10+9=12+7$

b. 反拨时钟3个小时，相当于 $10-3=7$

对钟表走一圈为12的最大数而言，顺拨时的 $10+9$ 和反拨的 $10-3$ 是相等的。

数学上 $+9$ 和 -3 就称为最大数12的互为补数，或称 $+9$ 是 -3 对模12的补码。 ■

由上可见，通过补码，一个减法运算可以变换成加法运算。

一个n位的二进制数x（不包括符号位）的补码可用下式方法求取：

$$[x]_{\text{补}} = \begin{cases} [x] & (\text{当} x \text{为正数}) \\ 2^n - [x] & (\text{当} x \text{为负数}) \end{cases} \blacksquare$$

例如 $(-1010)_{\text{补}} = 2^4 - 1010 = 10000 - 1010 = 0110$

$$\begin{aligned} [x]_{\text{补}} &= \text{符号位} + \text{原数值} & (x \text{为} \text{正数}) \blacksquare \\ &= \text{符号位} + \text{原数值的补码} & (x \text{为} \text{负数}) \end{aligned}$$

x为正数时，补码和原码相同；

x为负数时，符号位仍为1，数值位按位取反，再在最低位加1。

例如 $x_1 = +1001010$ 的补码是 $[x_1]_{\text{补}} = 01001010 \blacksquare$

$x_2 = -1001010$ 的补码是 $[x_2]_{\text{补}} = 10110110 \blacksquare$

补码的运算规则

$$[x_1]_{\text{补}} + [x_2]_{\text{补}} = [x_1 + x_2]_{\text{补}}$$

补码再求补=原码

$$[x_1 - x_2]_{\text{补}} = [x_1]_{\text{补}} + [-x_2]_{\text{补}} \blacksquare$$

例如求12-9=?

$$[1100-1001]_{\text{补}} = [1100]_{\text{补}} + [-1001]_{\text{补}} = 01100 + 10111 = 100011$$

其中，最高位自然丢失(溢出)，次高位0为符号位，运算结果为+3。 ■

又如求9-12=?

$$[1001-1100]_{\text{补}} = [1001]_{\text{补}} + [-1100]_{\text{补}} = 01001 + 10100 = 11101$$

结果是负数，再求补后得10011，所以是-3。 ■

几个数的真值、原码、反码、补码

x	[x]_原	[x]_反	[x]_补	x	[x]_原	[x]_反	[x]_补
+1010	01010	01010	01010	-1001	11001	10110	10111
+0100	00100	00100	00100	-0011	10011	11100	11101

1.3 数字电路中的代码

生活中用一组十进制数来代表一个特定对象的情况是很多的。如电话号码、邮政编码等等。用一组十进制数代替一个特定对象的过程称为编码。

而在数字电路中，用一组二进制数来代替某一特定的对象，这组二进制数就是代表该对象的代码。代替的方法有非常多的种类。

➤ 二-十进制编码(BCD码)

十进制数的0~9十个数字分别用一个四位的二进制编码表示，称十进制数的二进制编码，简称BCD码(**Binary Coded Decimal**)。

四位二进制数有十六种不同组合，只要选出其中的十种分别代替0、1、...、9十个数码进行组合。

- **有权码**：8-4-2-1、5-4-2-1、...，分别表示这种代码方案中高位至低位的“权”，即每一位的1代表的十进制数值。
- **无权码**：某一位代码没有具体十进制数值的意义。

十进制数	有权码					无权码
	8421	5421	2421	2421*	5211	余三码
0	0000	0000	0000	0000	0000	0011
1	0001	0001	0001	0001	0001	0100
2	0010	0010	0010	0010	0100	0101
3	0011	0011	0011	0011	0101	0110
4	0100	0100	0100	0100	0111	0111
5	0101	1000	0101	1011	1000	1000
6	0110	1001	0110	1100	1001	1001
7	0111	1010	0111	1101	1100	1010
8	1000	1011	1110	1110	1101	1011
9	1001	1100	1111	1111	1111	1100

$$\begin{aligned}
\text{如 } (359)_{10} &= (0011 \ 0101 \ 1001)_{8421} \quad \square \\
&= (0011 \ 1000 \ 1100)_{5421} \quad \square \\
&= (0011 \ 0101 \ 1111)_{2421} \quad \square \\
&= (0101 \ 1000 \ 1111)_{5211} \quad \square \\
&= (0110 \ 1000 \ 1100)_{\text{余三码}} \quad \square \\
&= (101100111)_2
\end{aligned}$$

8421BCD码有时也简称为BCD码。

➤ 格雷码(Gray Code)循环码

是一种可靠性编码。因为这种代码中任何二组相邻代码之间只相差一位码不同，其它码相同的特性。

十进制数	4位循环码	十进制数	4位循环码
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

➤ 字符代码

■ **ISO编码**(International Standardization Organization)

国际标准组织制定的八位二进制代码，主要用于信息交换，它包括十进制数的10个数码，26个英文字母，以及+、-、×、÷、.....等20个符号，共56种特定对象。

■ **ASCII码**(American Standard Code for Information Interchange)

是美国国家信息交换标准代码的简称，也是八位二进制代码，其中一位作奇偶校验位。

1.4 数字电路中的基本逻辑函数

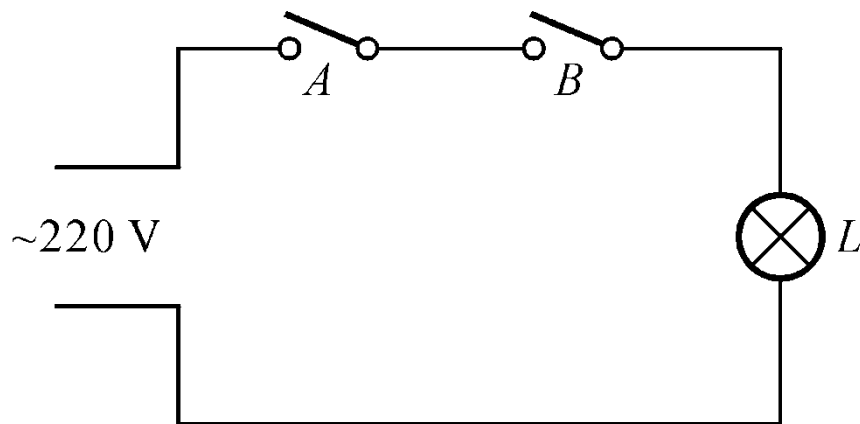
- 逻辑代数，又称**布尔代数**。由英国数学家乔治·布尔在1849提出。
- 它用来描述客观事物中的逻辑关系，约100年后才用在开关电路中。
- 用字母或符号表示变量，但是，该变量不代表具体数值大小，而只代表某种因果关系，或代表二种截然不同的状态，电平等。例如，开关的断开和闭合、晶体管的截止和饱和导电，灯的亮和暗，事件的是和非，真和假.....等

一、逻辑代数中的三种基本运算

1. 与逻辑关系

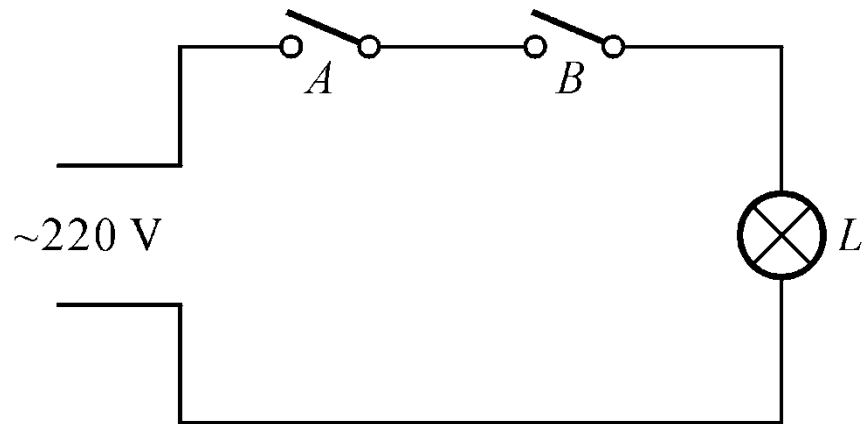
决定某一结果成立的各种条件都具备时，结果才成立。称为与逻辑。

如二只串联开关控制一只电灯，只有当二只开关都闭合时，电灯才亮。因此为与逻辑。



- 开关闭合:逻辑 “1”
断开:逻辑 “0”

- 灯亮暗结果
亮:逻辑 “1”
暗:逻辑 “0”



- 要使结果成立($L=“1”$),
二只串联的开关都必须
闭合($A=“1”$, $B=“1”$)。

$$L = f(A, B) = A \cdot B$$

条 件		结 果
A	B	L
0	0	0
0	1	0
1	0	0
1	1	1

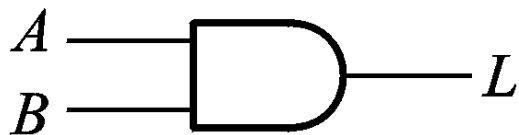
✧ 从逻辑运算上，与逻辑是**逻辑乘**关系：

$$0 \cdot 0 = 0, \quad 0 \cdot 1 = 0$$

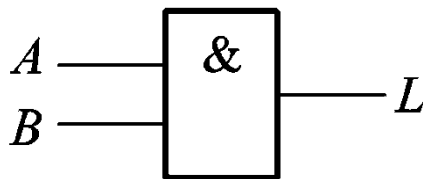
$$1 \cdot 0 = 0, \quad 1 \cdot 1 = 1$$

条 件		结 果
A	B	L
0	0	0
0	1	0
1	0	0
1	1	1

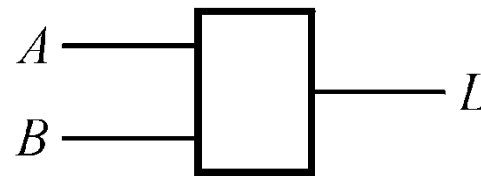
✧ 能完成“与”逻辑功能的电路称为与门。
逻辑符号为：



特定外型符号



国标符号

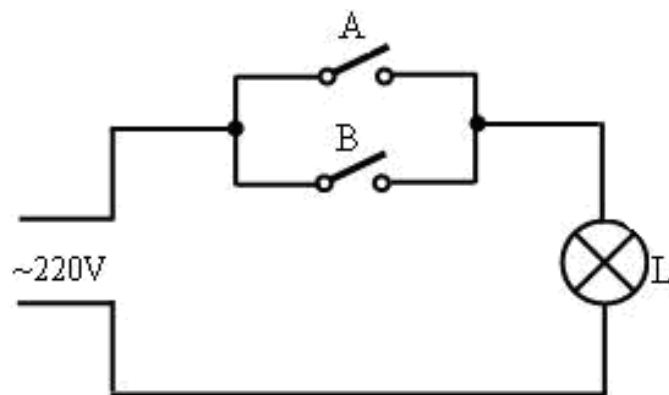


尚使用符号

2. 或逻辑关系

如果决定结果成立的条件中，只要有一个或一个以上的条件具备时，结果就能成立。称为或逻辑。

条 件		结 果
A	B	L
0	0	0
0	1	1
1	0	1
1	1	1



$$L = A + B$$

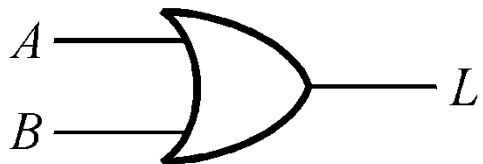
✧ 从逻辑运算上，或逻辑是逻辑加关系：

$$0+0=0, \quad 0+1=1$$

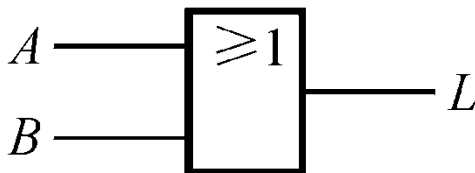
$$1+0=1, \quad 1+1=1$$

条 件		结 果
A	B	L
0	0	0
0	1	1
1	0	1
1	1	1

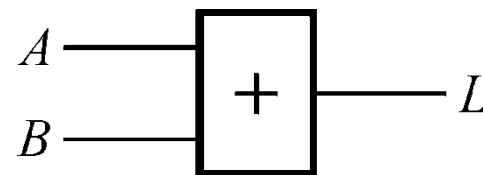
✧ 能完成“或”逻辑功能的电路称为或门。逻辑符号为：



特定外型符号



国标符号



尚使用符号

3. 非逻辑关系

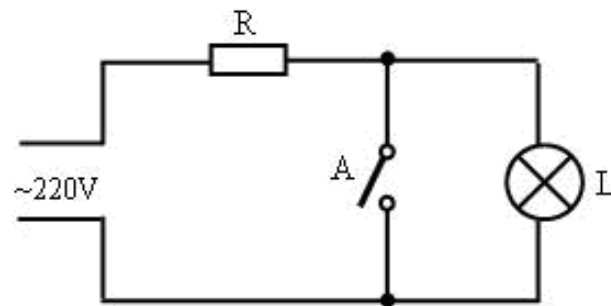
当条件具备时，结果不成立，反之，结果成立。

$$L = \overline{A}$$

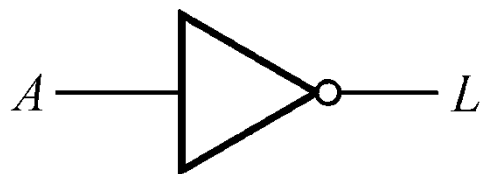
$$\overline{0} = 1$$

$$\overline{1} = 0$$

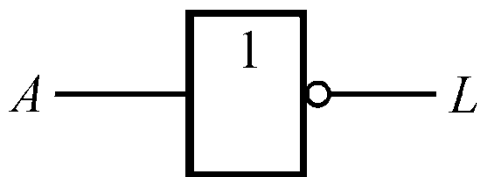
条 件	结 果
A	L
0	1
1	0



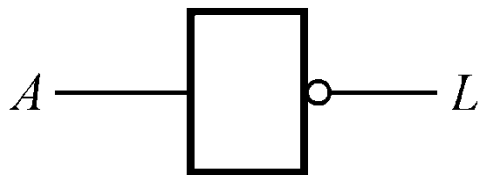
✧ 非门逻辑符号



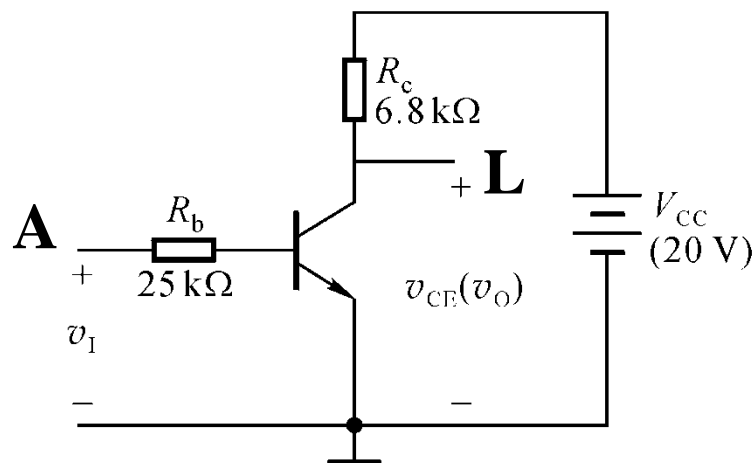
特定外型符号



国标符号



尚使用符号



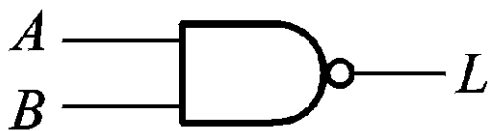
4. 复杂逻辑关系

➤ 与非

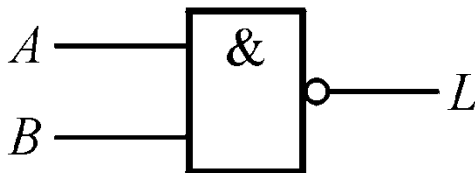
决定某一结果成立的各种条件都具备时，结果才不成立。

$$L = \overline{AB}$$

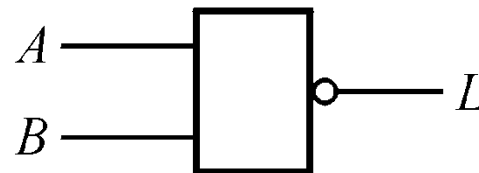
✧ 与非逻辑符号



特定外型符号



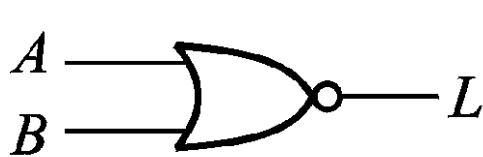
国标符号



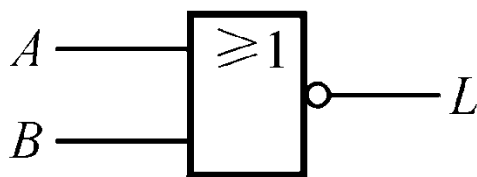
尚使用符号

条 件		结 果
A	B	L
0	0	1
0	1	1
1	0	1
1	1	0

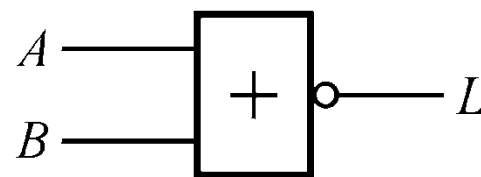
➤ 或非 $L = \overline{A+B}$



特定外型符号

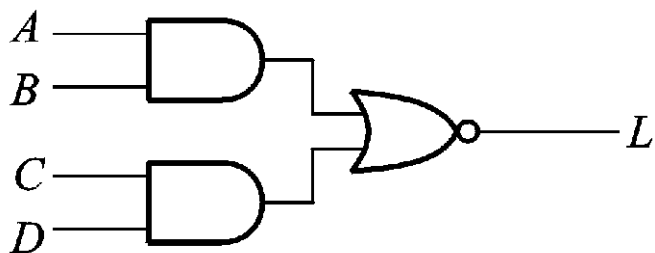


国标符号

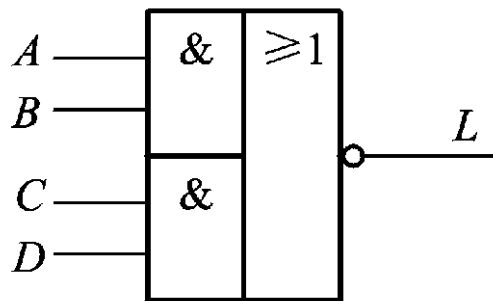


尚使用符号

➤ 与或非 $L = \overline{AB+CD}$



特定外型符号



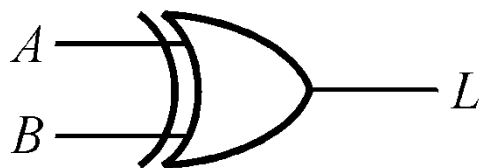
国标符号

➤ 异或逻辑关系

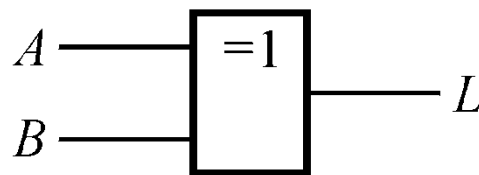
当决定结果的二个条件**相异**时，结果成立，二个条件相同时，结果不成立。

$$L = A \cdot \overline{B} + \overline{A} \cdot B = A \oplus B$$

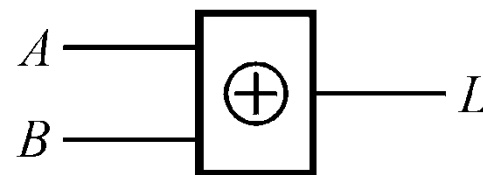
✧ 异或逻辑符号



国际流行符号



国标符号



尚使用符号

条 件		结 果
A	B	L
0	0	0
0	1	1
1	0	1
1	1	0

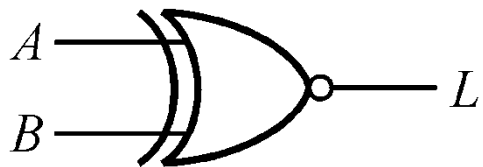
➤ 同或逻辑关系

当决定结果的二个条件**相同**时，结果成立，二个条件相异时，结果不成立。

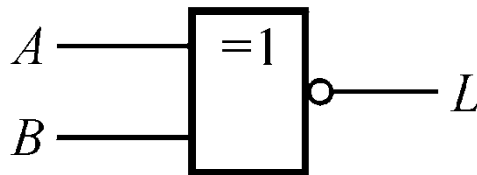
$$L = A \cdot B + \overline{A} \cdot \overline{B} = A \odot B$$

条 件		结 果
A	B	L
0	0	1
0	1	0
1	0	0
1	1	1

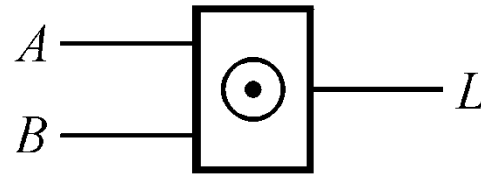
✧ 同或逻辑符号



国际流行符号



国标符号



尚使用符号