数字逻辑

课程要求

平时成绩组成：
课后作业15 + 实验10 +考勤5
小测验20分

第一章 概念介绍

1.1 数字和模拟量

连续(Continuous)和离散(Discrete)信号

模拟(Analog)和数字(Digital)信号：

- 从模拟信号到数字信号的转换一定有误差
- **采样频率**指从连续的模拟信号中搜集离散信号的频率，每个采样点存储的bits数决定了存储信息的多少
- 数字信号转模拟信号**DAC**，模拟信号转数字信号**ADC**

> 理论上，一个模拟信号的周期通常采样两次，则保留信息频率为采样频率的$\frac{1}{2}$

1.2 二进制数字、逻辑电平和数字波形

表示数字0和1的电压是**逻辑电压(logic levels)**，高电压和低电压之间有**容限。**低电压可降低功耗，高电压则能增大容错率

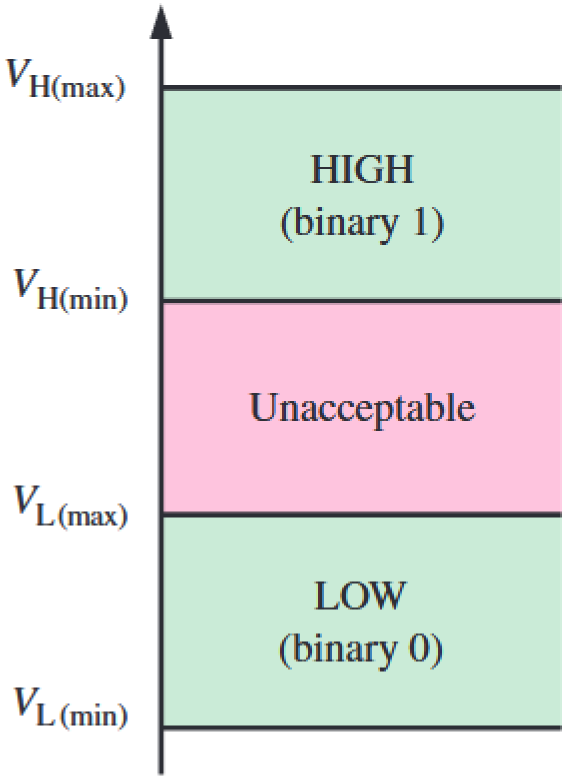
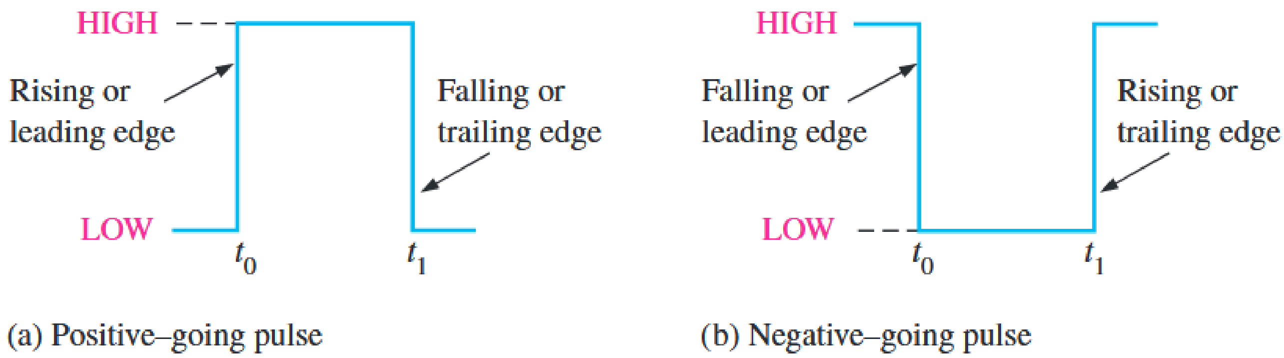


FIGURE 1–6 Logic level ranges of voltage for a digital circuit.

数字波形(digital waveforms) 具有理论上和实际上的两种形状



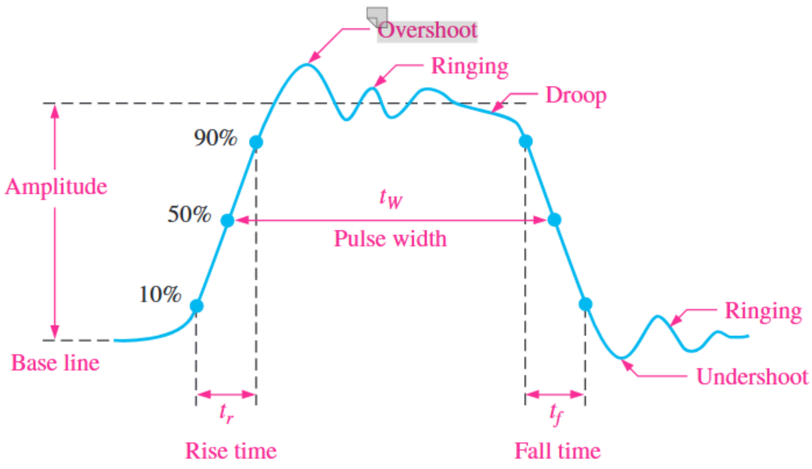


FIGURE 1–8 Nonideal pulse characteristics.

> 单位大小：$k=10^3$，$M=10^6$，$G=10^9$(在频率和周期中，不同于内存单位)

占空比(duty cycle)：

$$\text{Duty cycle} = \frac{t_w}{T}100\%$$

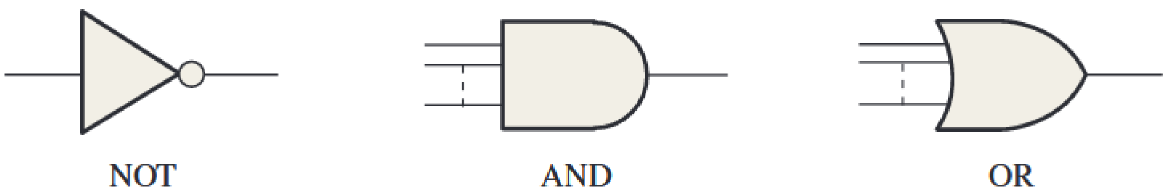
其中t_w表示高电平时间，T表示周期

数字系统中的所有波形都与**时钟(clock)** 同步，时钟的每个周期表示*传输1bit*的时间。显示多个波形实际变化情况的图是**时序图(timing diagrams)**

1.3 基本逻辑功能

逻辑门(logit gate) 拥有一或两个输入和一个输出，基础的门分为：

- **非门(NOT)**：输出与输入相反
- **与门(AND)**：只要有0则输出0，否则输出1
- **或门(OR)**：只要有1就输出1，否则输出0



1.4 组合和顺序逻辑功能

逻辑门的组合可以实现：

1. **比较器(comparator)**，比较数字大小
2. **加法器(adder)**，实现加法；加法实现后同理实现其他四则运算
3. **编码器(code conversion function)** 和 **解码器(encoding function)**，实现信号和二进制数的转换
4. **计数器(counting function)**，对信号计数

5. **多路复用器(multiplexing)** 和 **解多路复用器(demultiplexing)**，将多路信号转换到一路信号和一路信号转换到多路信号
6. **存储器(Storage function)**，存储数据
7. **进程控制系统(process control system)**，对各种进程进行控制

第二章 数字系统，操作符和编码

2.1-2.3 十进制与二进制

重要概念：

- 十进制数和二进制数的**基数(base)** 分别是10和2
- 最右边和最左边的位分别叫**最低有效位(least significant bit, LSB)** 和 **最高有效位(most significant bit, MSB)**
- **八进制(Octal)**，**十六进制(Hexadecimal)** 和 **十进制(decimal)**

> 十进制转八进制、十六进制前再转换为二进制，二进制再转其他，按位数转。
> 补位时，要从远离小数点的地方补齐，不能改变原来的大小

二进制转十进制

按照基数+指数的方法直接转换

$$\begin{aligned}110_{(2)} &= 1*2^2+1*2^1+0*2^0\\ &= 4+2\\ &= 6_{(10)}\end{aligned}$$

十进制转二进制

整数部分
减法：

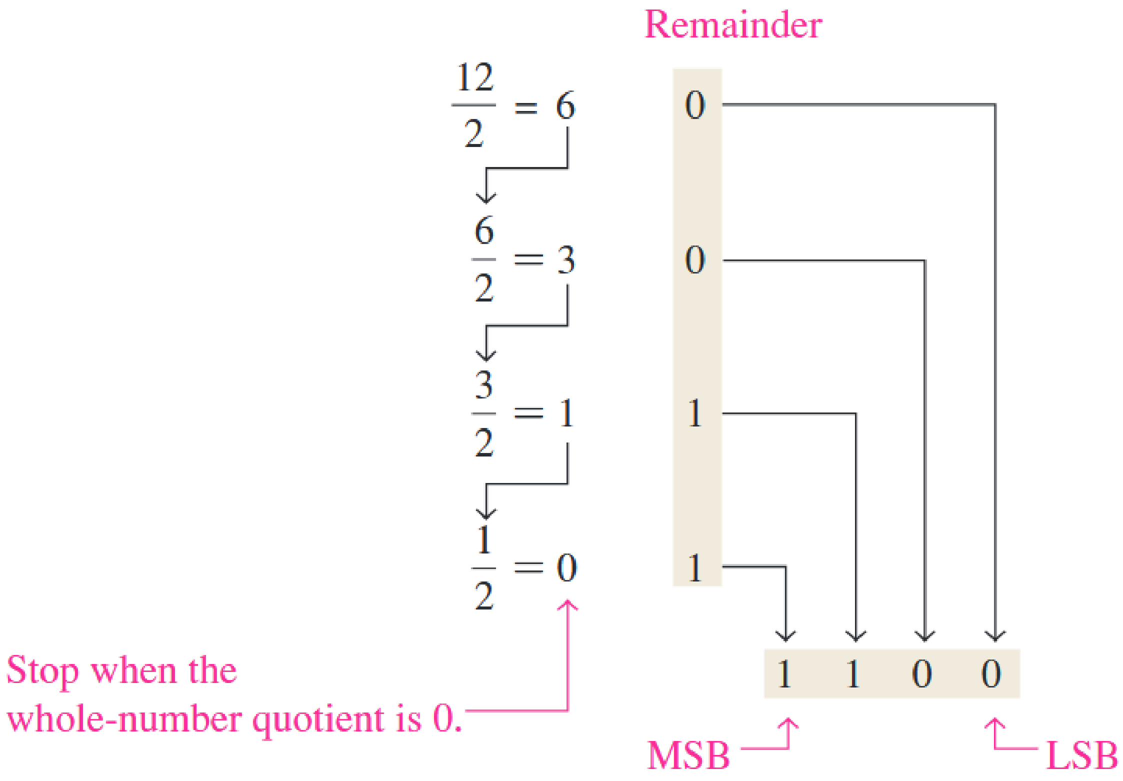
1. 找到距离原十进制数最近的2^x数，减去
2. 对于剩余的被减数，重复此操作
3. 将得到的2^x数的指数位置摘出为1，其他为0

$$\begin{aligned}49_{(10)} &= 32+16+1\\ &= 2^5+2^4+2^0\\ &= 110001_{(2)}\end{aligned}$$

除二取余法：

4. 原数除以2取出余数放边上
5. 连续进行此操作，直到原数变为0

6. 将余数倒置

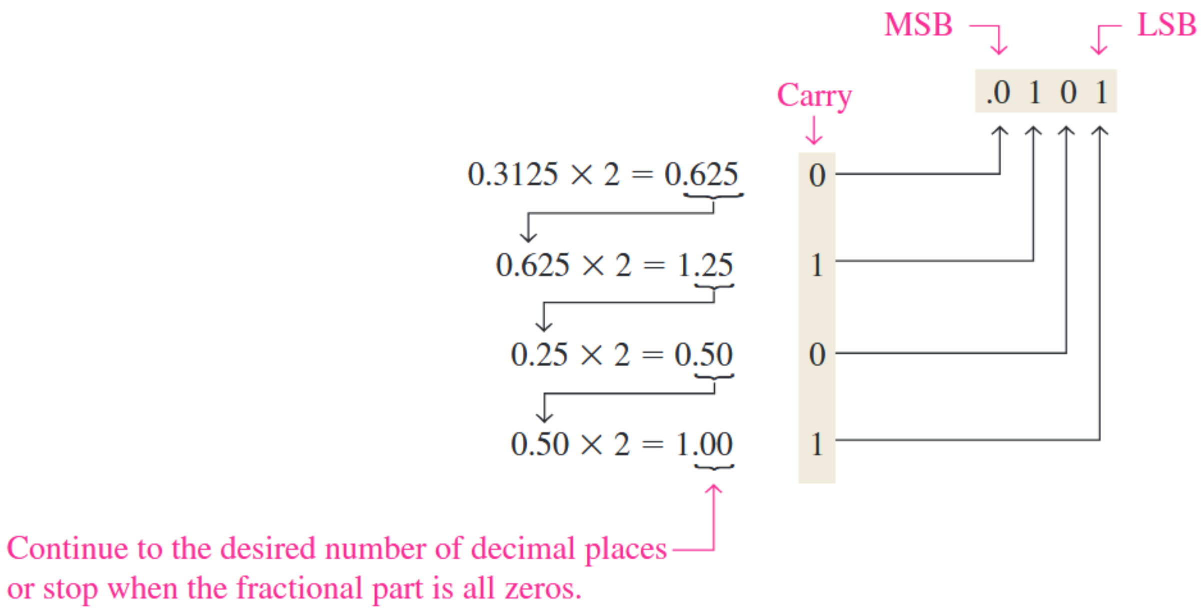


小数部分

乘二取整法：

7. 小数部分每次乘以2，留下整数
8. 重复此操作，直到小数点后面为0，或者*达到需求精度*

> 有可能无限循环，因为十进制转二进制留下误差



2.4-2.6 二进制运算，补码和符号数

重要概念

- 原码 符号位+二进制位
- 补数 设一个r进制的数N_r，则它们补数$[N_r] = r^n - N_r$，其中n表示位数

设一个无符号数N，则它的：

1. 有符号正数$+N$，即在原来的数前加一个0，反码与之相同
2. 有符号负数$-N$，即在$+N$取补码(各位取反再加一)

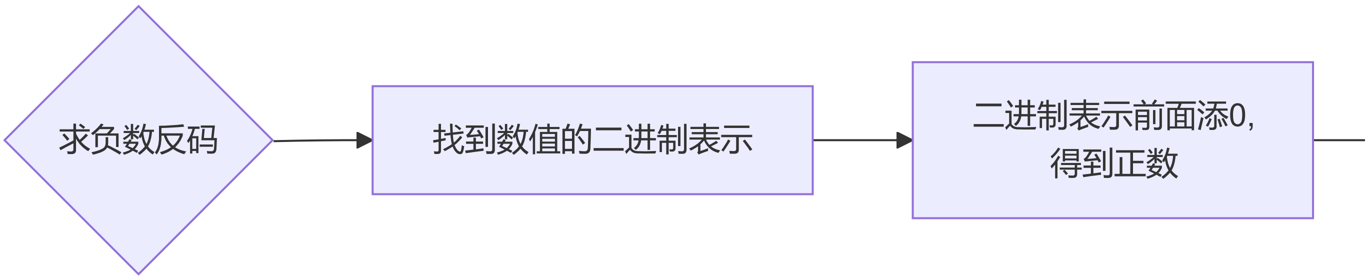
> 我们用的有符号数为一个符号位加上数值，因此求反码必须转换为有符号数再做

注意：

1. “补码等于反码加一”仅针对负数
2. 负数的反码是正符号数的各位取反(包括符号位也取反，符号位本身是0)
3. 补码与原码相同针对二进制正数，十进制正数补码和原码不同
4. 求补数和求补码不同。二进制正数的**补码(complement code)** 和**补数(complement)** 不同，前者同原码，后者各位取反加一

求反码的算法

负数的反码等于正数各位取反(正数是符号位为0+数值)



求补码的算法

定义法：

$$[N_r] = r^n - N_r$$

LSB-MSB算法：

1. 从LSB往MSB，若遇到0，则复制，直到遇到非0
2. 对第一个非0数，用r减去它，r表示进制
3. 后续的数，用$r-1$减去
   减法：
4. 对于每一位的数，用$r-1$减去之
5. 对结果，加一

> 正数的补码和原码同样

特别地，求二进制数的补码：

1. 找到对应有符号正数
2. 包括符号位的所有位取反
3. 加一

2.7-2.9 有符号数的计算和其他进制

有符号数的加法，本质上是补码的加法和符号位的**溢出 (overflow)**。符号位的溢出可能出错，即超过有符号数位数的表示范围
二进制数的减法为加法与负数

> 计算时，先把位次对齐 (补零) 后再计算；其中正数直接补零，负数找补码时，先找对应正数补零，再求补码

二进制数的除法可用乘法优化：

- 对 2 的整数幂，用移位的方法
- 对非 2 的整数幂，改为乘以小数
- 对非常量 (如 `X / Y`)，对 `Y` 构造乘法表。需要一个较小范围的 `Y`

> 探测两个正数相加或两个负数相加的溢出，可检测符号位是否改变

2.10 二进制编码

二进制表示的十进制码 (binary coded decimal, BCD)，即对每一位的数字都用二进制表示，例如：

$$11_{(10)} = 1011_{(2)} = 0001,0001_{(BCD)}$$

这体现了二进制数和二进制编码的差异，常用于时钟显示等

> BCD 码分为**8421 BCD**和**84-2-1 BCD**等，表示 4 位二进制的表示的权重，即各位和

Gray码(Gray code) 的特点：

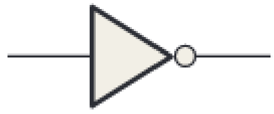
1. 相邻位差异最小：Gray码中任意两个连续的数值仅有一位二进制位不同。例如，十进制的 `1` 和 `2` 在Gray码中分别表示为 `0001` 和 `0011`，只有第二位不同
2. 循环性：Gray码是一个循环码，即最后一个码字与第一个码字之间也只有一位不同。例如，对于3位Gray码，`000` 和 `100` 之间只有一位不同
3. 非权码：Gray码是一种无权码，每一位的值没有固定的权重，不像普通二进制码那样可以直接转换为十进制数

奇偶校验 (parity method) 用于信号传输过程中的检误。奇校验即保证传输数据的'1'的个数为奇数个，偶校验类似

> 奇偶校验只能检查一位错误；CRC 校验可以检测多位的错误

第三章门电路

3.1 非门

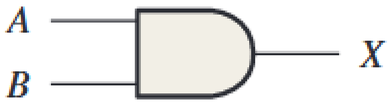


非门(the Inverter/ NOT gate) 将输入电平转换为相反的电平

下面是非门的**真值表 (truth table)**

输入 (A)	输出 (NOT A)
0	1
1	0

3.2 与门



与门 (AND gate) 在输入中有 0 时输出 0，否则输出 1；把某些位设置为 0，只需要与一个这些对应位为 0，其余为 1 的数：

$$1001\&1100 = 1000$$

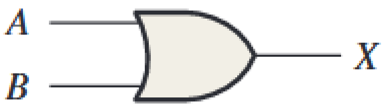
> 在 ASCII 码中，大写字母和小写字母仅有一位区别，可用与运算转换

输入 A	输入 B	输出 (A AND B)
0	0	0
0	1	0
1	0	0
1	1	1

在求波形图逻辑运算时，只要找到对应高电平留下，其余设置为低电平即可

逻辑与和二进制乘法按位运算的结果相同，记为 $X = AB$

3.3 或门



或门 (OR gate) 在输入中有 1 时输出 1，否则输出 0，记为 $X = A + B$

输入 A	输入 B	输出 (A OR B)
0	0	0
0	1	1
1	0	1
1	1	1

3.4-3.5 与非门和或非门

与或非门称为**基本门 (basic gate)**，用它们构成的与非和或非门是**通用门 (universal gate)**

与门后接非门为**与非门(NAND)**，表现为同高才低，否则为高。逻辑运算为 $X = \overline{AB}$，真值表：

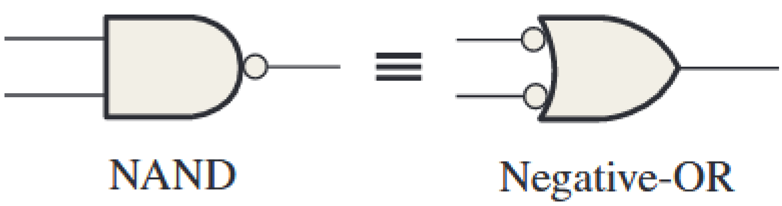
输入A	输入B	输出（A NAND B）
0	0	1
0	1	1
1	0	1
1	1	0

问题： 用 NAND 构建 NOT
解决： 将 NAND 的一端接入高电平，则另一端输入时，最终输出相反

问题： 用 NAND 构建 OR
解决： 将两个输入端取反即得

实际上，NAND 和输入取非再取或的门 (Negative-OR) 等价，即摩根定律：

$$\overline{AB} = \overline{A} + \overline{B}$$



或门后接非门为**或非门 (NOR)**，表现为有高则低，否则为高，逻辑运算 $X=\overline{A+B}$，真值表：

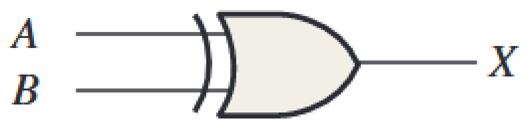
输入A	输入B	输出（A NOR B）
0	0	1
0	1	0
1	0	0
1	1	0

问题： 用 NOR 构建 NOT
解决： 将 NOR 的一端接入低电平，则另一端输入时，输出为反

问题： 用 NOR 构建 AND
解决：输入端取反，原理即摩根定律

3.6 异或门和同或门

异或门 (exclusive-OR gate, XOR) 在两个输入不同时输出为 1，相同时输出为 0



异或的真值表为：

输入A	输入B	输出（A XOR B）
0	0	0
0	1	1
1	0	1
1	1	0

表达式为：

$$X=A\oplus B=\overline{A}B+A\overline{B}$$

异或门的特性：

- 只有一个输入反向，输出反向
- 两个输入反向，输出不变

在异或门后加上非门得到**同或门(XNOR)**，真值表为：

输入A	输入B	输出（A XNOR B）
0	0	1
0	1	0
1	0	0
1	1	1

表达式为：

$$X = A \odot B = \overline{A}\overline{B} + AB$$

第四章 布尔运算与逻辑简化

4.1-4.2 布尔运算的算符和表达式和规则

在布尔运算中，'+'表示**或(boolean addtion)** 运算，'*'表示**与(boolean multiplication)** 运算

> 和式为 0，表示参数全为 0；乘式为 1，表示参数全为 1

布尔运算满足**交换律**：

$$\begin{aligned} &A + B = B + A \\ &AB = BA \end{aligned}$$

布尔运算满足**结合律**：

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

布尔运算满足**分配率**：

$$AB + AC = A(B + C)$$

一些指的注意的运算规律：

- **补码律** $A + \overline{A} = 1, A\overline{A} = 0$
- **幂等律** $A + A = A$，$AA = A$
- **补码律** $A + \overline{A} = 1$, $A\overline{A} = 0$
- **吸收律** $A + AX = A$，$A + \overline{A}X = A + X$
- 吸收律推广 $(A + B)(A + C) = A + BC, A(A + X) = A$
  证明方法可以使用韦恩图或者真值表

4.3 摩根定律

摩根定律 (DeMorgan's Theorems) 就是脱帽法，有以下两条：

$$
\overline{XY} = \overline{X} + \overline{Y}
$$
$$
\overline{X+Y} = \overline{X} \cdot \overline{Y}
$$

摩根定律可以扩展到多个相乘或相加

常用摩根定律化简表达式，例如：

$$
\overline{(A+B+C)D} = \overline{A+B+C} + \overline{D} = \overline{A} \cdot \overline{B} \cdot \overline{C} + \overline{D}
$$

此外，可用摩根定律化简式子后求一个电路的真值表

特别地，双重否定律和摩根定律不能混淆：

- 摩根定律

$$
\overline{\overline{A}B} = A + \overline{B}
$$

- 双重否定律

$$
\overline{\overline{AB}} = AB
$$

4.4-4.5 用布尔表达式分析逻辑电路和表达式化简

用布尔表达式分析逻辑电路，可以方便的画出电路的真值表，步骤：

1. 找到电路的布尔表达式
2. 化简表达式，使用运算规律或者摩根定律
3. 在表达式中，找到输出值为 1 的情况
4. 其余情况输出为 0
   需要注意的是，可用幂等性添项来消去其他项，即 $A = A + A = AA$

注意 $A\overline{B} + \overline{A}B$ 就是异或的表达式，$AB + \overline{A}\overline{B}$ 就是同或的表达式，二者互相取反可以得到

4.6 标准布尔表达式

标准表达式有两种形式**和的乘积 (product-of-sums, POS)** 或者 **乘积的和 (sum-of-products, SOP)**，并且要求所有的字母都要出现在标准表达式中

将一个普通表达式转换为标准表达式，可以通过添加项的方法实现，例如：

$$
A\overline{B}C = A\overline{B}C(D + \overline{D}) = A\overline{B}CD + A\overline{B}C\overline{D}
$$

其中 $(X + \overline{X})$ 是 1 因子，可添加到乘积项中

> SOP 和标准 SOP (standstard SOP) 不是一回事！后者需要包含所有的字母

4.7 布尔表达式和真值表

对于任意真值表，都可写出其 POS 和 SOP 表达式，且该两种表达式等价

问题：已知真值表，如何获得真值表的标准布尔表示?
方法：

1. 求 SOP 表达式，则先找输出等于 0 的组合。要使输出为 0，且表示为乘积的和，则各项乘积都为 0
2. 求 POS 表达式，则先找输出等于 1 的组合。要使输出为 1，且表示为和的乘积，则各项乘积都为 1

问题：已知标准布尔表达式，如何获得真值表?
方法：

1. 已知 SOP，则对和式的每一个因子，若该因子为 1，找到对应的字母数值，则真值表输出为 1；其余组合必为 0
2. 已知 POS，则对积式的每一个因子，若该因子为 0，找到对应的字母数值，则真值表输出为 0,；其余组合必为 1

在 SOP 中，每项称为**最小项(minterm)**，当最小项值为 1 时，SOP 整体输出为 1。用二进制表示最小项实现，则该二进制对应的十进制数 i 作为下标，即 m_i 表示最小项，该 SOP 可用最小项的和改写，例如：

$$A\overline{B}C+\overline{A}BC \to 101+011=m_5+m_3$$

在 POS 中，每项称为**最大项 (maxterm)**，当最大项为 0 时，POS 整体输出为 0,。用二进制表示最大项实现， 则该二进制对应的十进制数 i 为下标，即 M_i 表示最大项，该 POS 可用最大项的积改写，例如：

$$(\overline{A}+B+C)(\overline{A}+\overline{B}+C) \to 100 \cdot 110 = M_4 \cdot M_6$$

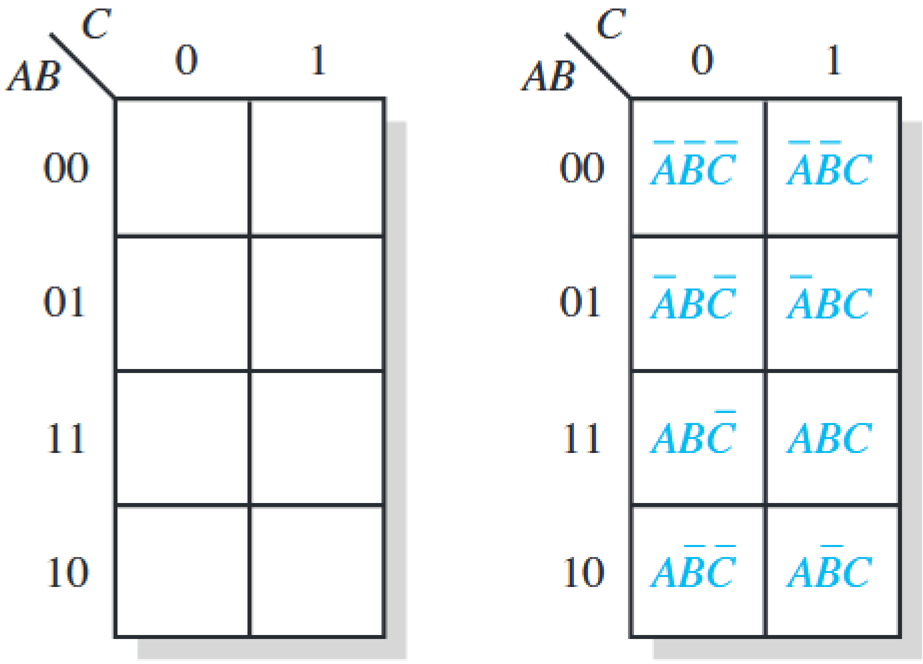
> 最小项要等于 1，最大项要等于 0

作一个标准布尔表达式的真值表，则真值表的每一行都能找对应的最小项和最大项，它们之间的关系是互补，即 $m_i=\overline{M}_i$

4.8 卡洛图

卡洛图 (the Karnaugh Map) 是布尔表达式的图形表示，看起来是一个二维的图像，每个方格满足空间上和逻辑上相邻

三个变量的卡洛图：

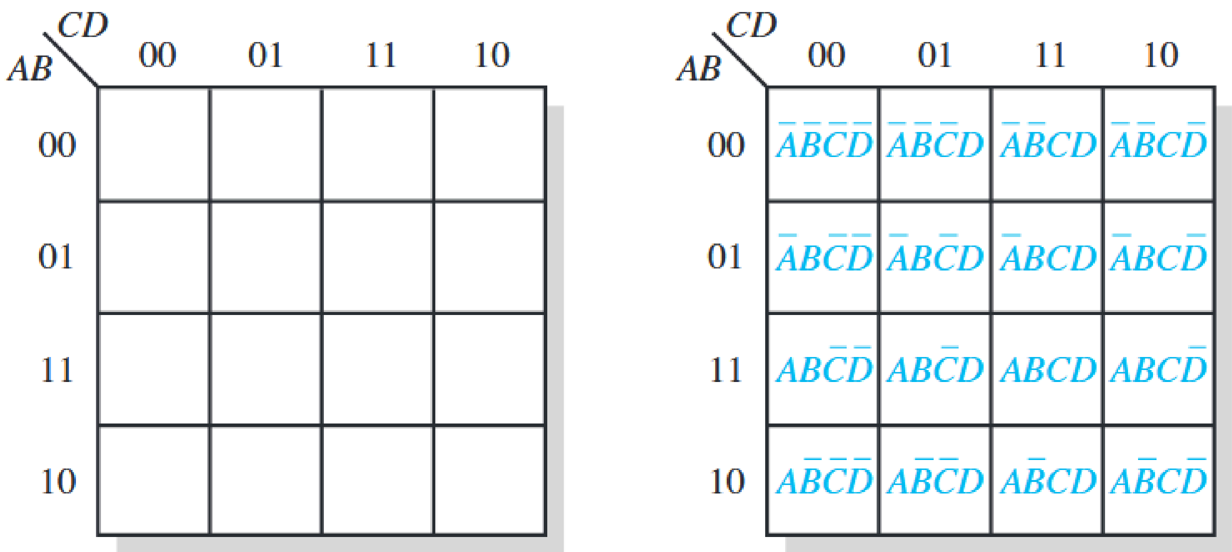


(a) (b)

读数时，按照 A,B,C 的字母顺序。注意 AB 的第三个格子是 11，这是为了满足两种相邻

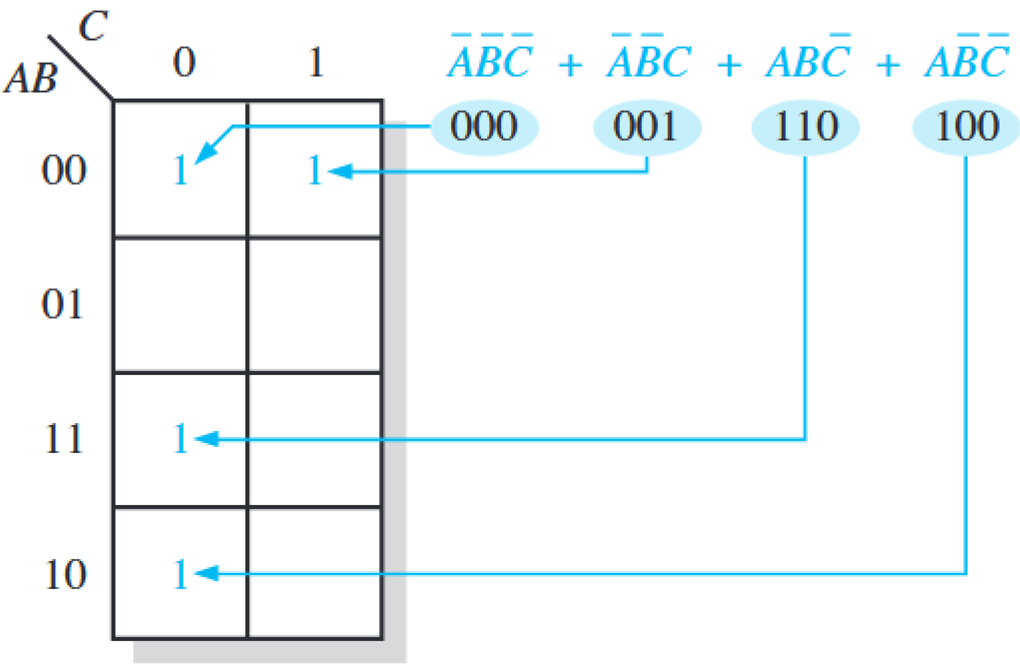
> 卡洛图的每一行末尾与下一行的行头满足两种相邻

四个变量卡洛图：



(a) (b)

将 SOP 映射到卡洛图，只需要将每一项为 1 的填入对应格子：



非标准形式首先需要转换为标准形式

使用卡洛图，可以化简布尔表达式。对于一个布尔表达式，我们首先填充它的卡洛图，然后根据 2^n 个数相邻的 1，进行画圈，接着在被圈的范围内，对照行列中被修改的变量并消去，留下剩下的变量 (注意若该变量是 0 要取反)，最后相加

> 本节建议参考 CSDN

部分概念：

- **蕴含项(implicant)**，即卡诺图中的 1
- **质蕴含项 (prime implicant, PI)**，即卡诺图中可以圈出来的 1 的组合
- **必要质蕴涵项 (essential prime implicant, EPI)**，即卡诺图中可以圈出来 1 的组合，且该组合中至少有一个项不会被其他的 PI 圈到
  化简表达式，本质上是找 EPI 的和

含"Don't Care"项的卡诺图化简，即将不可能输入或不关心输出的项写作'X'，'X'可以随意指定值，故指定为 1 与已经存在的正常 1 进行化简

> 'Don't care'项是人为规定的；任意指定的 1 必须要至少带上一个正常 1 才能化简