数字电路基础

一、逻辑代数定律和计算规则

定律/规则名称	表达式	解释
恒等律	$A + 0 = A$ $A \cdot 1 = A$	任何变量与0相加或与1相乘等于自身
零律	$A + 1 = 1$ $A \cdot 0 = 0$	任何变量与1相加或与0相乘等于1或0
幂等律	$A + A = A$ $A \cdot A = A$	任何变量与自身相加或相乘等于自身
互补律	$A + \overline{A} = 1$ $A \cdot \overline{A} = 0$	任何变量与其补码相加等于1，相乘等于0
交换律		
加法交换律	$A + B = B + A$	加法运算的交换律
乘法交换律	$A \cdot B = B \cdot A$	乘法运算的交换律
结合律		
加法结合律	$(A + B) + C = A + (B + C)$	加法运算的结合律
乘法结合律	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$	乘法运算的结合律
分配律		
乘法分配律	$A \cdot (B + C) = A \cdot B + A \cdot C$	乘法对加法的分配律
加法分配律	$A + (B \cdot C) = (A + B) \cdot (A + C)$	加法对乘法的分配律
吸收律		
吸收律1	$A + A \cdot B = A$	吸收律的第一种形式
吸收律2	$A \cdot (A + B) = A$	吸收律的第二种形式
德摩根定律		
德摩根定律1	$\overline{A + B} = \overline{A} \cdot \overline{B}$	逻辑加法的德摩根定律

定律/规则名称	表达式	解释
德摩根定律2	$\overline{A \cdot B} = \overline{A} + \overline{B}$	逻辑乘法的德摩根定律
简化定律		
简化定律1	$A + \overline{A} \cdot B = A + B$	简化逻辑表达式
简化定律2	$A \cdot (\overline{A} + B) = A \cdot B$	简化逻辑表达式
共识定律		
共识定律(积之和形式)	$AB + \overline{A}C + BC = AB + \overline{A}C$	较难，常用于逻辑化简。项 `BC` 是 `AB` 和 `A`C 的共识项，是冗余的。
共识定律(和之积形式)	$(A + B)(\overline{A} + C)(B + C) = (A + B)(\overline{A} + C)$	较难，常用于逻辑化简。项 `(B+C)` 是 `(A+B)` 和 `(A` +C) 的共识项，是冗余的。
反演定律		
反演定律	$A = \overline{\overline{A}}$	变量的双重否定等于自身

推导过程

1. **基本定律**
   - **恒等律**：$A + 0 = A$ 和 $A \cdot 1 = A$ 是逻辑代数的基本定义。
   - **零律**：$A + 1 = 1$ 和 $A \cdot 0 = 0$ 也是逻辑代数的基本定义。
   - **幂等律**：$A + A = A$ 和 $A \cdot A = A$ 是因为逻辑加法和乘法运算的特性。
   - **互补律**：$A + \overline{A} = 1$ 和 $A \cdot \overline{A} = 0$ 是逻辑变量和其补码的定义。
2. **交换律**
   - **加法交换律**：$A + B = B + A$ 是逻辑加法的交换特性。
   - **乘法交换律**：$A \cdot B = B \cdot A$ 是逻辑乘法的交换特性。
3. **结合律**
   - **加法结合律**：$(A + B) + C = A + (B + C)$ 是逻辑加法的结合特性。
   - **乘法结合律**：$(A \cdot B) \cdot C = A \cdot (B \cdot C)$ 是逻辑乘法的结合特性。
4. **分配律**
   - **乘法分配律**：$A \cdot (B + C) = A \cdot B + A \cdot C$ 是逻辑乘法对加法的分配特性。
   - **加法分配律**：$A + (B \cdot C) = (A + B) \cdot (A + C)$ 是逻辑加法对乘法的分配特性。
5. **吸收律**
   - **吸收律1**：$A + A \cdot B = A$ 可以从 $A + A \cdot B = A \cdot (1 + B) = A \cdot 1 = A$ 推导得出。
   - **吸收律2**：$A \cdot (A + B) = A$ 可以从 $A \cdot (A + B) = A \cdot A + A \cdot B = A + A \cdot B = A$ 推导得出。
6. **德摩根定律**
   - **德摩根定律1**：$\overline{A + B} = \overline{A} \cdot \overline{B}$ 是逻辑加法的德摩根定律。

- **德摩根定律2**：$\overline{A \cdot B} = \overline{A} + \overline{B}$ 是逻辑乘法的德摩根定律。

7. **简化定律**
   - **简化定律1**：$A + \overline{A} \cdot B = A + B$ 可以从 $A + \overline{A} \cdot B = (A + \overline{A}) \cdot (A + B) = 1 \cdot (A + B) = A + B$ 推导得出。
   - **简化定律2**：$A \cdot (\overline{A} + B) = A \cdot B$ 可以从 $A \cdot (\overline{A} + B) = A \cdot \overline{A} + A \cdot B = 0 + A \cdot B = A \cdot B$ 推导得出。
8. **共识定律**
   - **共识定律**：$(A + B) \cdot (\overline{A} + C) = (A + B) \cdot (\overline{A} + C) \cdot (B + C)$ 可以从 $(A + B) \cdot (\overline{A} + C) = (A + B) \cdot (\overline{A} + C) \cdot (B + C)$ 推导得出，因为 $(A + B) \cdot (\overline{A} + C) \leq (B + C)$。
9. **反演定律**
   - **反演定律**：$A = \overline{\overline{A}}$ 是逻辑变量的双重否定特性。

二、基本门电路

1. 非门

$$Y = \overline{A}$$

2. 与门

$$Y = A \cdot B$$

真值表:

输入 A	输入 B	输出 Y
0	0	0
0	1	0
1	0	0
1	1	1

3. 或门

$$Y = A + B$$

真值表:

输入 A	输入 B	输出 Y
0	0	0
0	1	1

输入 A	输入 B	输出 Y
1	0	1
1	1	1

4. 与非门

与非门是“与门”和“非门”的结合。

$$Y = \overline{A \cdot B}$$

真值表:

输入 A	输入 B	输出 Y
0	0	1
0	1	1
1	0	1
1	1	0

5. 或非门

或非门是“或门”和“非门”的结合。

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

真值表:

输入 A	输入 B	输出 Y
0	0	1
0	1	0
1	0	0
1	1	0

6. 异或门

当两个输入不相同时，输出为高电平（1）；当两个输入相同时，输出为低电平（0）。这也被称为“半加器”的求和逻辑。

逻辑表达式:

$$Y = A \oplus B$$

真值表:

输入 A	输入 B	输出 Y
0	0	0
0	1	1
1	0	1
1	1	0

三、编码

1. 原码、反码和补码

为了在二进制系统中表示正负数，我们通常会使用最高位作为**符号位**。

- 符号位为 **0** 代表**正数**。
- 符号位为 **1** 代表**负数**。

原码

- **规则**: 符号位 + 数值的绝对值的二进制表示。
- **正数**: 符号位为0，其余位表示数值。
  - 例如，$+12$ 的原码是 **00001100**。
- **负数**: 符号位为1，其余位表示数值。
  - 例如，-12 的原码是 **10001100**。
- **缺点**:
  1. 零的表示不唯一：$+0$ 是 **00000000**，-0 是 **10000000**。
  2. 进行加减法运算时，需要单独处理符号位，硬件实现复杂。

反码

反码的出现是为了简化减法运算。

- **规则**:
  - **正数**的反码与其原码**相同**。
  - **负数**的反码是在其**原码**的基础上，**符号位不变**，其余各位**按位取反**。
- **示例**:
  - $+12$ 的原码是 `00001100`，其反码也是 **00001100**。
  - -12 的原码是 `10001100`，其反码是 **11110011** (符号位1不变，后面7位 `0001100` 按位取反得到 `1110011`)。
- **缺点**:
  - 仍然存在“双零”问题：$+0$ 的反码是 **00000000**，-0 的反码是 **11111111**。

- 跨零运算会产生循环进位问题。

补码

补码是现代计算机系统中最常用的有符号数表示法，它解决了原码和反码的缺点。

- **规则**:
  - **正数**的补码与其原码**相同**。
  - **负数**的补码是其**反码加 1**。
- **求负数补码的方式**:
  - 从其原码的**最低位（最右边）**向左找，找到的**第一个 1** 保持不变，这个 1 **左边**的所有位（不含符号位）按位取反，符号位仍为1。
- **示例**:
  - $+12$ 的补码是 **00001100**。
  - -12 的补码求法：
    1. 原码: `10001100`
    2. 反码: `11110011`
    3. 加 1: `11110011 + 1` = **11110100**。
- **优点**:
  1. **零的表示唯一**: **00000000**。
  2. **简化运算**: 可以将减法运算转换为加法运算。例如，计算 $A - B$ 等同于计算 $A + (-B)$ 的补码。
  3. 对于一个 n 位的补码系统，其表示范围为 $[-2^{n-1}, 2^{n-1} - 1]$。例如，8位补码的范围是 $[-128, 127]$。

总结表格 (以 ±12 为例)

值	原码	反码	补码
+12	00001100	00001100	00001100
-12	10001100	11110011	11110100

2. BCD 码

BCD码是用**二进制**来表示**十进制**数的一种编码方式。它与直接将十进制数转换为二进制数不同。

- **规则**: 用 **4 位二进制数**来表示一位十进制数（0-9）。最常用的是 **8421 BCD 码**，其中各位的权值从高到低分别是 8、4、2、1。
- **特点**:
  - 它介于二进制和十进制之间，便于人机交互（如数码管显示、计算器）。
  - 运算比纯二进制复杂，但比直接处理十进制字符简单。

- 由于用4位二进制表示一位十进制数，所以 `1010` 到 `1111` 这 6 个码是无效或非法的。

BCD 码对照表

十进制	BCD 码
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

示例: 将十进制数 **129** 转换为 BCD 码。

1. 将每一位十进制数分开：`1`、`2`、`9`。
2. 将每一位分别转换为对应的4位BCD码：
   - $1 \rightarrow 0001$
   - $2 \rightarrow 0010$
   - $9 \rightarrow 1001$
3. 将它们组合起来：

$$(129)_{10} = (0001\ 0010\ 1001)_{\text{BCD}}$$

对比: 如果将 $(129)_{10}$ 直接转换为纯二进制，结果是 **10000001**。这与它的 BCD 码是完全不同的。

四、加法器、编码器、译码器、选择器、比较器

五、触发器

1. RS 触发器

最基本的触发器，但存在一个不确定状态，在实际应用中较少直接使用。

- **输入**: S (Set, 置位), R (Reset, 复位)
- **输出**: Q (状态输出), $\overline{Q}$ (反向输出)

功能表

这张表描述了在不同输入下，下一个状态 Q_{n+1} 是什么。

S	R	Q_{n+1}	**功能**
0	0	Q_n	保持
0	1	0	复位/置0
1	0	1	置位/置1
1	1	**?**	**禁止/不定**

特性方程

$$Q_{n+1} = S + \overline{R}Q_n \quad (\text{约束条件: } S \cdot R = 0)$$

激励表

这张表在电路设计时非常有用，它回答了“为了让状态从 Q_n 变为 Q_{n+1}，输入 S 和 R 应该是什么？”。（X表示Don't Care，即0或1均可）

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

2. JK 触发器

JK 触发器是 RS 触发器的改进版，它解决了 RS 触发器的“禁止”状态问题，是最通用的触发器。

- **输入**: J (功能类似 S), K (功能类似 R)
- **输出**: Q, $\overline{Q}$

功能表

J	K	Q_{n+1}	**功能**
0	0	Q_n	保持
0	1	0	复0
1	0	1	置1
1	1	$\overline{Q_n}$	**翻转**

*JK触发器将RS触发器的禁止状态（1,1输入）变成了一个非常有用的**翻转**功能。*

特性方程

$$Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n$$

激励表

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

3. D 触发器

D 触发器的功能非常直接：在时钟脉冲到来时，将输入 D 的值传递给输出 Q。它常被用作数据锁存器或移位寄存器的基本单元。

- **输入**: D (Data)
- **输出**: Q, $\overline{Q}$

功能表

D	Q_{n+1}	**功能**
0	0	置0
1	1	置1

无论当前状态 Q_n 是什么，下一个状态 Q_{n+1} 都等于时钟边沿到来时的 D 输入值。

特性方程

$$Q_{n+1} = D$$

激励表

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

4. T 触发器

T 触发器是一个翻转触发器。当输入 $T=1$ 时，状态翻转；当 $T=0$ 时，状态保持不变。它常用于构建计数器。

- **输入**: T
- **输出**: Q, $\overline{Q}$

功能表

T	Q_{n+1}	**功能**
0	Q_n	保持
1	$\overline{Q_n}$	翻转

特性方程

$$Q_{n+1}=T\oplus Q_n=T\overline{Q_n}+\overline{T}Q_n$$

激励表

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0