

# 《数字电路设计》复习笔记

由H DU-STE A\_Y Y整理于2020年1月5日

## 编码

### 十进制编码

#### BCD码

由于人们习惯使用十进制数，而电路单元最适宜于二进制操作，于是出现了一种用二进制码编写的十进制码，即二—十进制码，或称BCD码（Binary Coded Decimal）

十进制	8421码	2421码	5211码	4311码	84-2-1码	余3码
0	0000	0000	0000	0000	0000	0011
1	0001	0001	0001	0001	0111	0100
2	0010	0010	0011	0011	0110	0101
3	0011	0011	0101	0100	0101	0110
4	0100	0100	0111	1000	0100	0111
5	0101	1011	1000	0111	1011	1000
6	0110	1100	1010	1011	1010	1001
7	0111	1101	1100	1100	1001	1010
8	1000	1110	1110	1110	1000	1011
9	1001	1111	1111	1111	1111	1100

## 8421

8421 BCD码是最基本和最常用的BCD码，它和四位自然二进制码相似，各位的权值为8、4、2、1，故称为有权BCD码。和四位自然二进制码不同的是，它只选用了四位二进制码中前10组代码，即用0000~1001分别代表它所对应的十进制数，余下的六组代码不用。

## 5421和2421

5421 BCD码和2421 BCD码为有权BCD码，它们从高位到低位的权值分别为5、4、2、1和2、4、2、1。这两种有权BCD码中，有的十进制数码存在两种加权方法，例如，5421 BCD码中的数码5，既可以用1000表示，也可以用0101表示；2421 BCD码中的数码6，既可以用1100表示，也可以用0110表示。这说明5421 BCD码和2421 BCD码的编码方案都不是惟一的。

## 余3码

余3码是8421 BCD码的每个码组加3(0011)形成的。常用于BCD码的运算电路中。

## 余3循环码

余3循环码是无权码，即每个编码中的1和0没有确切的权值，整个编码直接代表一个数值。主要优点是相邻编码只有一位变化，避免了过渡码产生的“噪声”。

## 格雷码

任意相邻两个代码之间只有一位状态不同，这样在计数过程中就不会出现其它代码，译码时就不会产生抖动和毛刺

## 编码规则

二进制数:  $B_{n-1}B_{n-2}\cdots B_0$

格雷码:  $G_{n-1}G_{n-2}\cdots G_0$

$$G_i = B_{i+1} \oplus B_i \quad i = n-2 \cdots 0$$

保持最高位不变，其它位与前一位异或

## ASCII码

国际上常采用的有ASCII码(美国标准信息交换码)其用7位二进制数表示，可表示95个图形字符以及33个控制字符

数字0-9的高3位是011，低4位是0000-1001

大小写字母转换也很容易，只是a5位的不同(100->110 101->111)

# 逻辑代数基础

## 公式法化简

名称	公式1	公式2
0-1律	$A \cdot 1 = A$	$A + 0 = A$
	$A \cdot 0 = 0$	$A + 1 = 1$
互补律	$A\bar{A} = 0$	$A + \bar{A} = 1$
重叠率	$AA = A$	$A + A = A$
交换律	$AB = BA$	$A + B = B + A$
结合律	$A(BC) = (AB)C$	$A + (B + C) = (A + B) + C$
分配律	$(A + B)C = AC + BC$	$AB + C = (A + C)(B + C)$
反演律	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$
吸收律	$A(A + B) = A$	$A + AB = A$
	$A(\bar{A} + B) = AB$	$A + \bar{A}B = A + B$
	$(A + B)(\bar{A} + C)(B + C) = (A + B)(\bar{A} + C)$	$AB + \bar{A}C + BC = AB + \bar{A}C$
对合律	$\bar{\bar{A}} = A$	

## 并项化简法

利用公式  $A + \bar{A} = 1$ , 将两项合并成一项, 并消去一个变量。

例: 化简  $F = A\bar{B}\bar{C} + A\bar{B}C$

解:  $F = A\bar{B}\bar{C} + A\bar{B}C = A\bar{B}(C + \bar{C}) = A\bar{B}$

例: 化简  $F = ABC + \bar{A}BC + B\bar{C}$

解:  $F = ABC + \bar{A}BC + B\bar{C} = (A + \bar{A})BC + B\bar{C} = BC + B\bar{C} = B$

例: 化简  $F = A(BC + \bar{B}\bar{C}) + A(B\bar{C} + \bar{B}C)$

解:  $F = A(BC + \bar{B}\bar{C}) + A(B\bar{C} + \bar{B}C) = AB(C + \bar{C}) + A\bar{B}(C + \bar{C}) = AB + A\bar{B} = A$

## 吸收法

利用  $A + AB = A$  或  $A + \bar{A}B = A + B$  公式, 消去多余项

例:  $\bar{A}B + \bar{A}BCD(E + F) = \bar{A}B$

例: 化简  $F = A\bar{B} + C + \bar{A}\bar{C}D + B\bar{C}D$

解:  $F = A\bar{B} + C + \bar{A}\bar{C}D + B\bar{C}D = A\bar{B} + C + \bar{C}(\bar{A} + B)D$

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

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

## 消去法

利用  $A + \bar{A}C + BC = AB + \bar{A}C$  公式, 将冗余项  $BC$  消去, 且含冗余项  $BC$  的“与”项仍是冗余项

例: 化简  $F = AC + A\bar{B}CD + ABC + \bar{C}D + ABD$

解:

$$\begin{aligned} F &= AC + A\bar{B}CD + ABC + \bar{C}D + ABD \\ &= AC(1 + \bar{B}D + B) + \bar{C}D + ABD = AC + \bar{C}D + ABD = AC + \bar{C}D \end{aligned}$$

## 配项法

利用  $A = A(B + \bar{B})$  或  $A = A + AB$  配项, 消去多余项

例: 化简  $F = AB + \bar{A}\bar{C} + \bar{B}\bar{C}$

$$\begin{aligned} \text{解: } F &= AB + \bar{A}\bar{C} + \bar{B}\bar{C} = AB + \bar{A}\bar{C} + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} \\ &= AB + \bar{A}\bar{C} + A\bar{B}\bar{C} \\ &= AB + \bar{C}(\bar{A} + A\bar{B}) \\ &= AB + \bar{C}(\bar{A} + \bar{B}) \\ &= AB + \bar{C}\overline{AB} \\ &= AB + \bar{C} \end{aligned}$$

例: 化简  $F = ABC + AB\bar{C} + A\bar{B}C + \bar{A}BC$

解:

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

## 求公式表示函数的反函数

1. 与 $\rightarrow$ 或 $\rightarrow$ 与
2.  $0 \rightarrow 1$   $1 \rightarrow 0$
3. 原变量 $\rightarrow$ 反变量 反变量 $\rightarrow$ 原变量
4. 多个变量的公共非号保持不变
5. 原式运算的优先顺序保持不变

如 已知函数  $L = A + B\bar{C} + \overline{AC\bar{D}}$ , 求其反函数  $\bar{L}$

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

## 求公式表示函数的对偶函数

1.  $0 \rightarrow 1$   $1 \rightarrow 0$
  2. 与 $\rightarrow$ 或 $\rightarrow$ 与
  3. 原式运算的优先顺序保持不变
- 若一个定理是正确的, 则其对偶式也一定正确
  - 若两个逻辑式相等, 则它们的对偶式也相等
  - 对对偶式再求对偶就得原函数本身

例如:

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

$$F' = (\bar{A} + B)(\bar{B} + C)$$

## 逻辑函数的表示形式

$$F = \overline{A}B + AC + BC(\text{与或式})$$

$$= (A + B)(\overline{A} + C)(\text{或与式})$$

$$= \overline{\overline{A}B \cdot \overline{A}C \cdot \overline{B}C}(\text{与非式})$$

$$= \overline{A + B + \overline{A} + C}(\text{或非式})$$

## 与或

- 包含若干个与项
- 每个与项中各变量以原变量或反变量的形式出现
- 每个与项以或的形式相连

## 或与

- 包含若干个或项
- 每个或项中各变量以原变量或反变量的形式出现
- 每个或项以与的形式相连

## 标准式

- n个变量组成的函数式
- 每个变量在函数式的每一项中都必须以原变量或反变量的形式出现一次，且仅出现一次

## 标准与或表达式

- 若一个函数完全由最小项相或组成，则称其为标准与或表达式
- 最小项之和

## 标准或与表达式

- 若一个函数完全由最大项相与组成，则称其为标准或与表达式
- 最大项之积

## 用公式法将逻辑函数变成最小项的形式

1. 利用 $A(\overline{B} + B) = A$ ，令每一项都包含全部代号
2. 整理①的结果，令每一项都是  
A、B、C、D的顺序
3. 将每项的A、B、C、D变成1，  
将A、B、C、D变成0
4. 将③的结果由二进制数变成十进制数
5. 在m后加上④中的结果

## 将最小项的形式化成变量形式

1. 取出m后的数字
2. 将数字变成二进制数(3输入即变成3位，4输入即变成4位)
3. 每个二进制数的第一个数是0则变成 $\overline{A}$ ，是1则变成A  
第二个数是0则变成 $\overline{B}$ ，是1则变成B  
第三个数是0则变成 $\overline{C}$ ，是1则变成C  
第四个数是0则变成 $\overline{D}$ ，是1则变成D

## 组合逻辑电路

### 逻辑电路的分类

#### 组合逻辑电路

任意时刻的输出仅仅取决于该时刻的输入组合，而与输入信号作用前电路的原状态无关(与过去的输入无关)

结构无反馈、功能上无记忆

#### 时序逻辑电路

任意时刻的输出不仅仅与该时刻的输入有关，而且还与电路的原状态有关(与过去的输入有关)

结构有反馈、记忆功能

### 组合逻辑电路的分析

1. 由逻辑图写出函数表达式
2. 化简函数表达式
3. 列出所有情况
4. 分析出功能

如 由第一步得到函数表达式  $L = A\overline{A}BC + B\overline{A}BC + C\overline{A}BC$

化简得:  $L = \overline{A}B + \overline{B}C + C\overline{A}$

画出真值表:

A	B	C	L
0	0	0	0
0	0	1	1
0	1	0	1
1	0	0	1
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

(不一致电路)

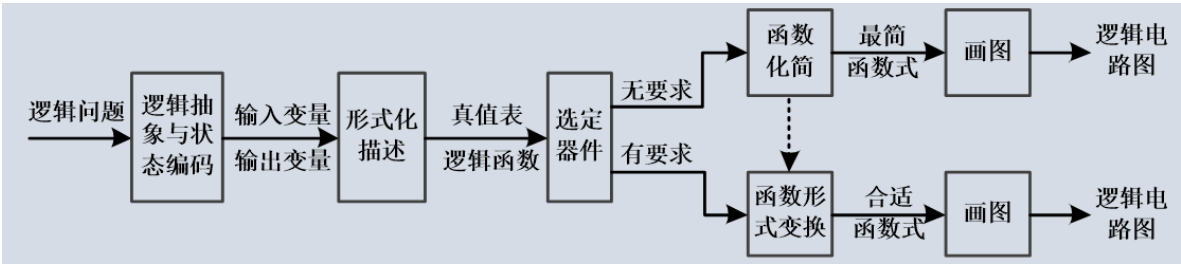
注意两种情况:

1. 得到最简逻辑表达式后，就能判断出电路的功能，不需要列出真值表；譬如  $F = \overline{A \oplus B}$ ，就是比较两个输入变量是否相同的电路（同或运算）。
2. 因为应用背景未知，可能无法用语言描述这个电路的功能，它只能通过逻辑函数来表示电路的逻辑功能，譬如  $F = \overline{A}B + BC$ 。

## 给出功能要求，设计电路

1. 分析命题中的事件因果关系，确定输入、输出变量。一般把引起事件的原因作为输入变量，把事件的结果作为输出变量。
2. 给变量赋值(0和1)
3. 列真值表
4. 确定使用的元件
5. 写出逻辑表达式
6. 化简
7. 画出电路图

列出真值表后，找到输出为1的情况，写出输入(输入之间取与)，之后将每种情况取或

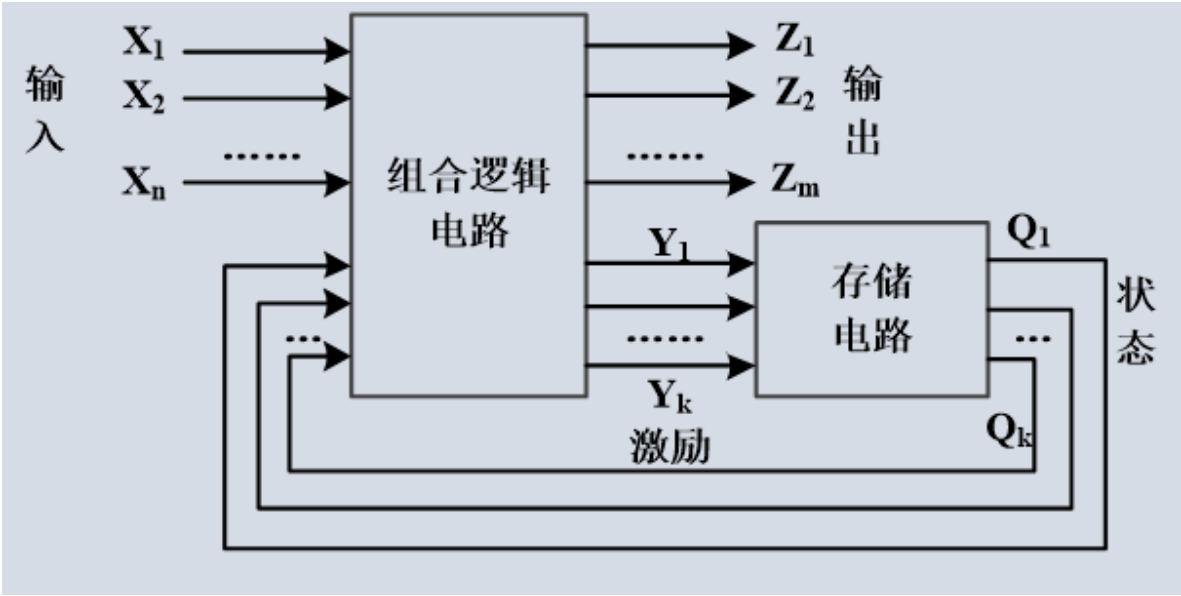


## 时序逻辑电路

### 时序逻辑电路的特点

- 由组合逻辑电路和存储元件构成
- 组合逻辑电路不是必要的，存储电路是必不可少的。
- 触发器是最常用的存储电路，具有记忆功能。

**时序逻辑电路的主要特点：**就一定含有存储元件，且电路上有反馈结构。



### 时序逻辑电路的描述方法

#### 方程组

描述输入  $X_1 \sim X_n$ ，输出  $Z_1 \sim Z_m$ ，激励输入  $Y_1 \sim Y_k$ ，状态变量  $Q_1 \sim Q_k$  之间的关系可用下面3个方程组来描述：

输出方程:

$$Z_i = f_i(\text{输入, 现态}) = f_i(X_1 \sim X_n, Q_1^n \sim Q_k^n), i = 1, \dots, m$$

激励方程:

$$Y_i = g_i(\text{输入, 现态}) = g_i(X_1 \sim X_n, Q_1^n \sim Q_k^n), i = 1, \dots, s$$

状态方程:

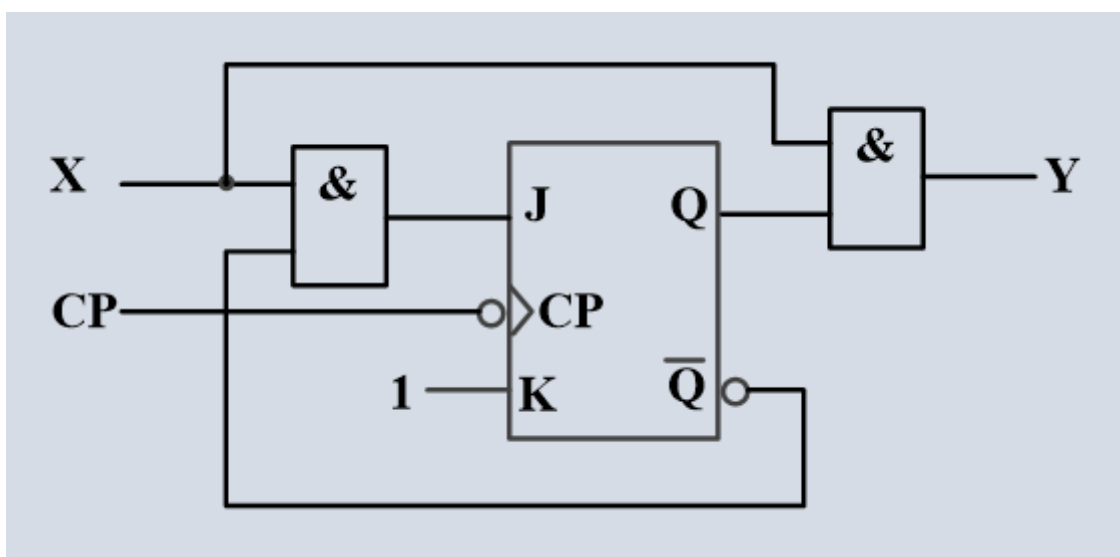
$$Q_i^{n+1} = h_i(\text{激励, 现态}) = h_i(Y_i, Q_i^n), i = 1, \dots, k$$

$Q^{n+1}$ : 表示电路中第*i*个触发器的下一个状态 (次态)

$Q^n$ : 表示电路中第*i*个触发器的现态 (或者原态)

**逻辑电路方程组的使用方法:**

例如下图:



1. 根据输入 $X$ 、现态 $Q^n$ 写出关于输出 $Z$ 的方程, 即输出方程

$$Y = X \cdot Q^n$$

2. 根据激励输入、现态 $Q^n$ 写出关于次态 $Q^{n+1}$ 的方程, 即激励方程

$$J = X \cdot \overline{Q^n} \quad K = 1$$

3. 根据存储元件( $JK$ 触发器)的特性方程和激励方程, 写出根据输入 $X$ 、现态 $Q^n$ 关于次态 $Q^{n+1}$ 的方程, 即状态方程

由于 $JK$ 触发器的特性方程是 $Q^{n+1} = J\overline{Q^n} + \overline{K}Q^n$ , 故得出:

$$Q^{n+1} = J\overline{Q^n} + \overline{K}Q^n = (X \cdot \overline{Q^n})\overline{Q^n} + \overline{1}Q^n = X \cdot \overline{Q^n}$$

## 状态转换真值表

### 时序逻辑电路的状态转换真值表

- 输入变量除了电路本身的输入变量外, 还包含电路的状态变量 (现态);
- 输出变量也包含两部分: 电路本身的输出以及电路的次态。
- 可以由问题本身的逻辑关系直接列出, 也可以由方程组计算得到。

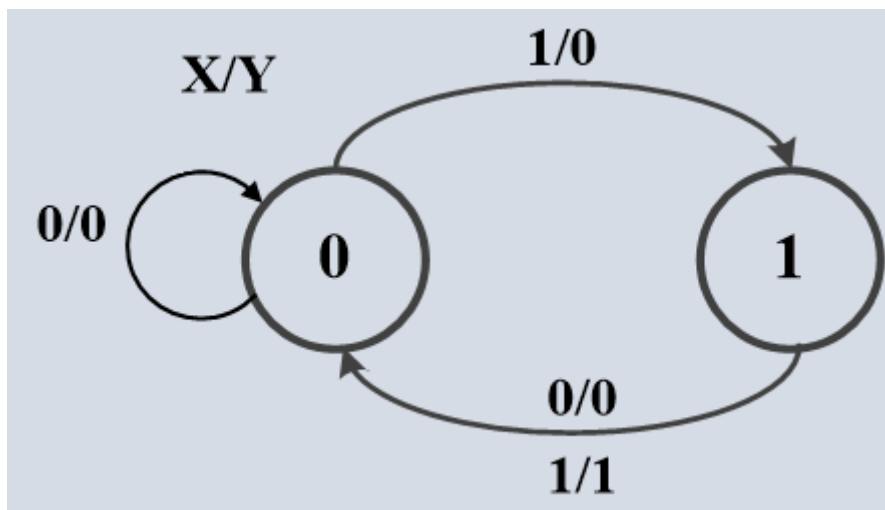


输入变量		输出变量	
电路输入	电路现态	电路次态	电路输出
$X$	$Q^n$	$Q^{n+1}$	$Y$
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

**注意：**由真值表画波形图时，不需要考虑次态 $Q^{n+1}$

## 状态转移图

最直观的描述方式，依据问题本身的逻辑关系或者由状态转换真值表画出



图中：

- 圆圈表示触发器的状态，即 $Q$ 的取值。电路中有几个触发器，就有 $2^k$ 个圆圈
- 带箭头的线条：连接各个圆圈，用来表示状态的转移关系；线条上是输入和输出信号的值，一般记为 $X/Z$ ，即表明在输入信号 $X$ 取什么值的状态下，从现态（原态）转到次态，并且输出信号 $Z$ 同时变化为新值。
- 状态转移图中，从每一个状态出发的线条数 $m$ 与输入信号的个数 $n$ 有关，在完整的状态转移图中， $m = 2n$

## 时序逻辑电路的分析

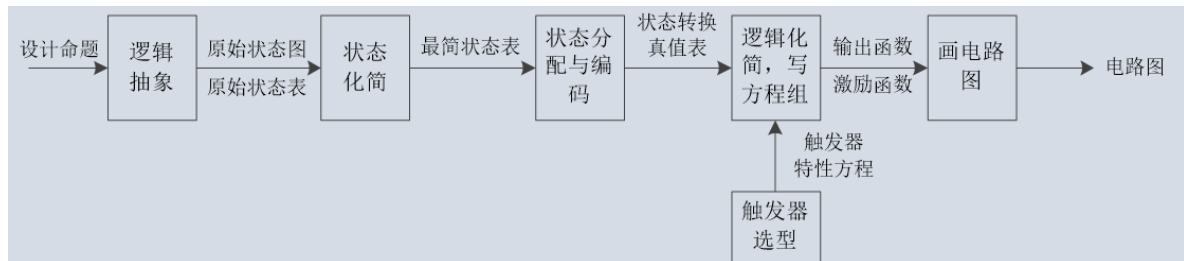
同步时序逻辑电路的分析步骤如下：

- 观察电路，写出方程组，即输出方程、激励方程和状态方程
- 列出状态转换真值表
- 根据状态转换真值表画出状态转移图
- 必要时，画出时序波形图（工作波形图）

## 时序逻辑电路的设计

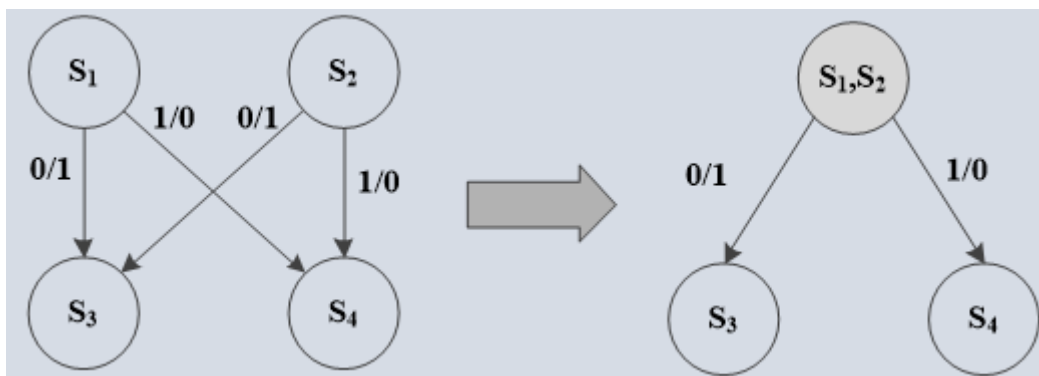
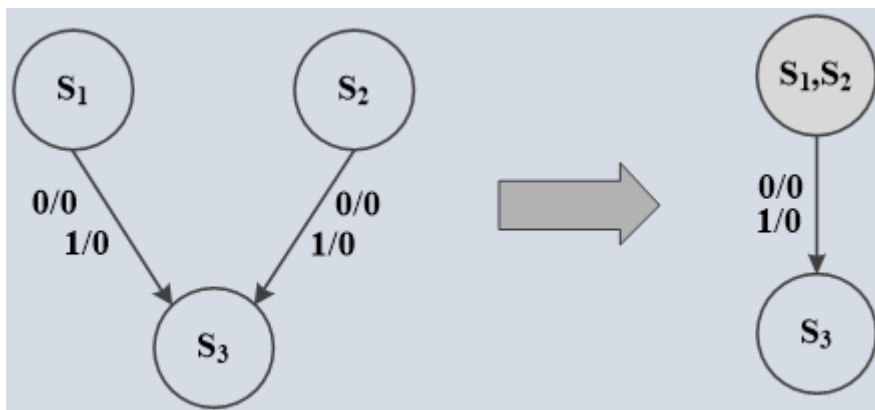
简单同步时序逻辑电路的设计通常经过以下几个步骤

- 根据题目要求，确定输入和输出变量以及电路的状态数(触发器数量)
- 建立原始状态转化图
- 列出原始状态表
- 找出原始状态图中的等价状态，合并他们，得到最小化状态表
- 状态分配/编码
- 列出状态转化真值表
- 选定触发器类型，写出方程式
- 检查电路
- 画出电路图

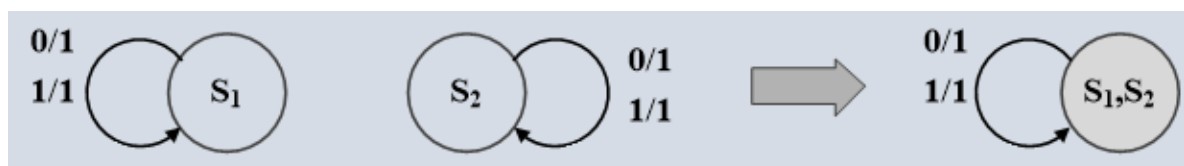


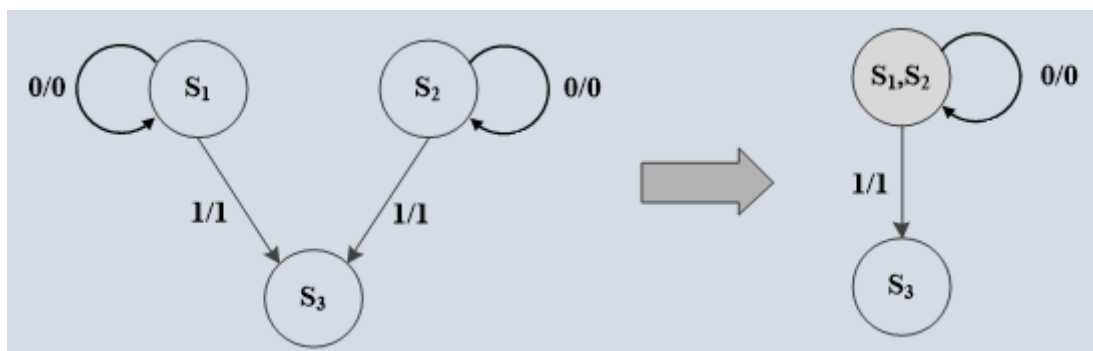
## 状态化简

### 次态相同

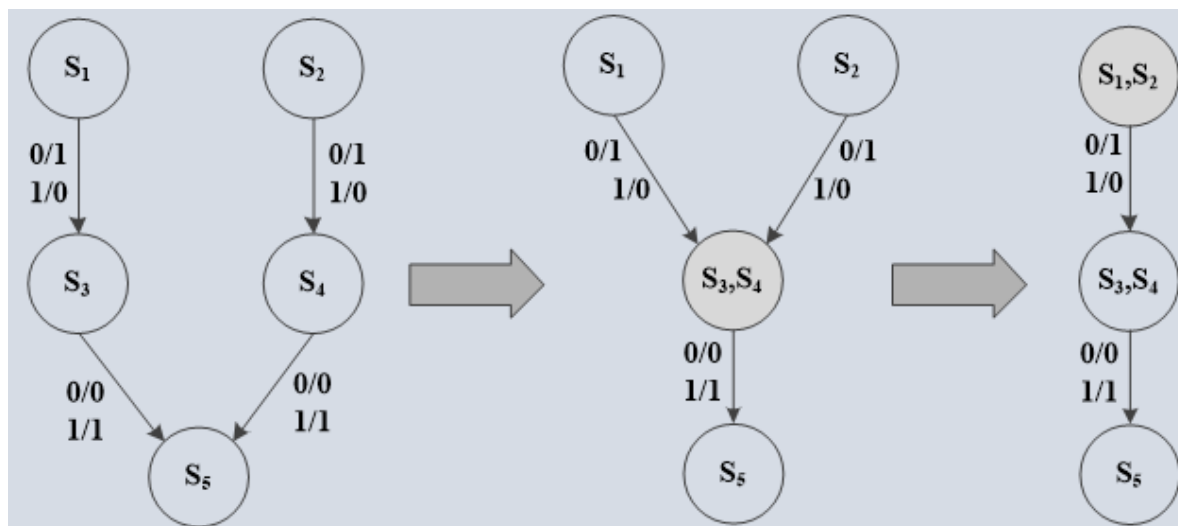


### 次态保持原态不变



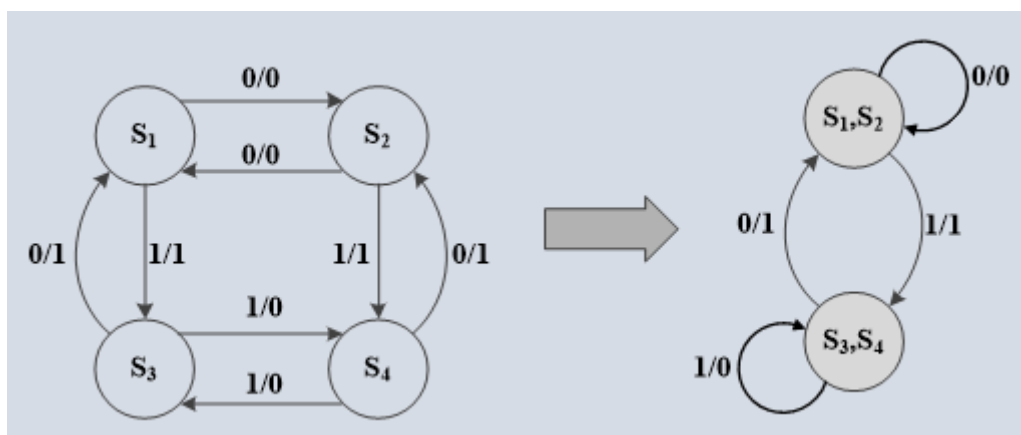
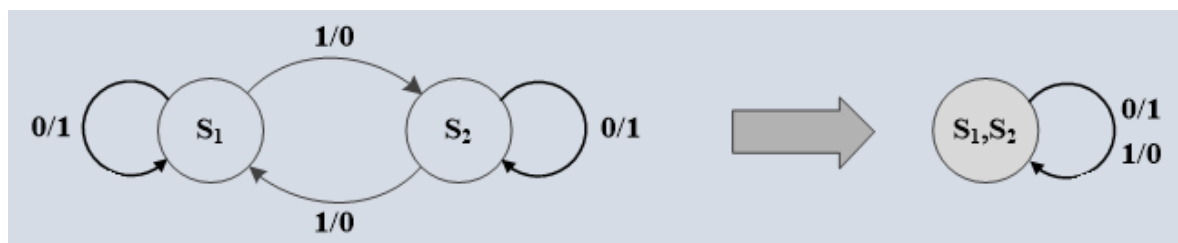


### 次态对等效



### 次态交错或者循环

- 次态交错：指在输入的某种组合条件下，两个状态互为对方的次态。
- 次态循环：指两个状态的次态虽然不相同，但这两个次态在某种输入组合下，经过一定的路径，又回到了原来的两个状态，则这两个状态等效。

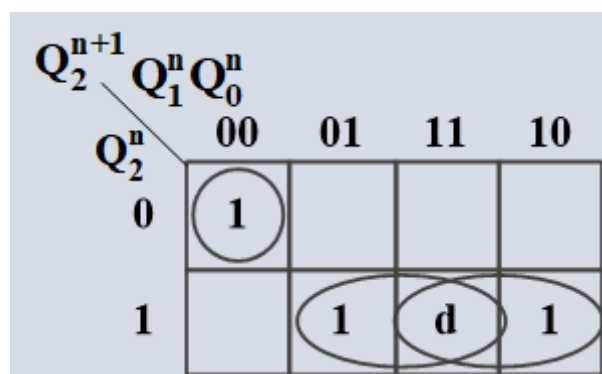
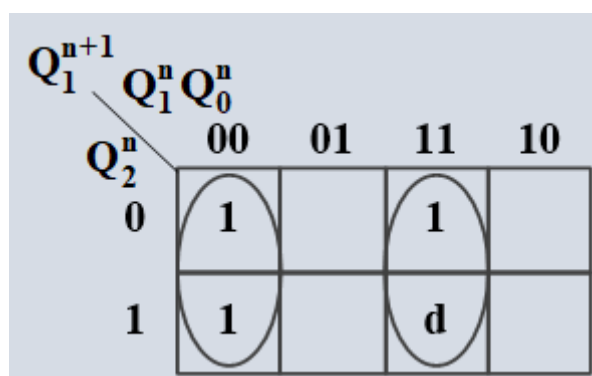
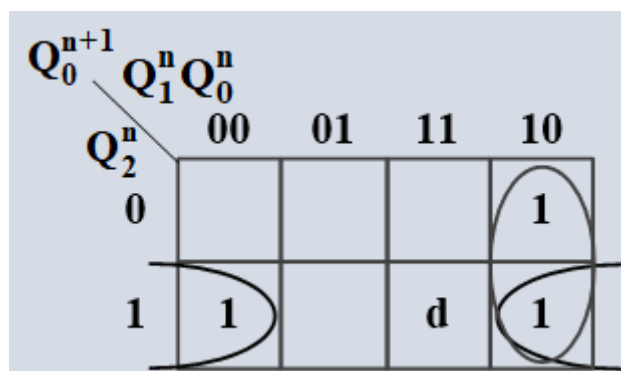


### 写出方程组

例如：对以下真值表：

$Q_2^n$	$Q_1^n$	$Q_0^n$	$Q_2^{n+1}$	$Q_1^{n+1}$	$Q_0^{n+1}$	Z
0	0	0	1	1	0	1
0	0	1	0	0	0	0
0	1	0	0	0	1	0
0	1	1	0	1	0	0
1	0	0	0	1	1	0
1	0	1	1	0	0	0
1	1	0	1	0	1	0
1	1	1	d	d	d	d

分别画出其三种次态和输出的卡诺图



		$G \quad Q_1^n Q_0^n$			
		00	01	11	10
$Q_2^n$	0	1			
	1			d	

得出状态方程：

$$Q_2^{n+1} = \overline{Q_1^n Q_0^n Q_2^n} + (Q_1^n + Q_0^n) Q_2^n$$

$$Q_1^{n+1} = \overline{Q_0^n Q_1^n} + Q_0^n Q_1^n$$

$$Q_0^{n+1} = (Q_2^n + Q_1^n) \overline{Q_0^n}$$

输出方程：

$$Z = \overline{Q_2^n Q_1^n Q_0^n}$$

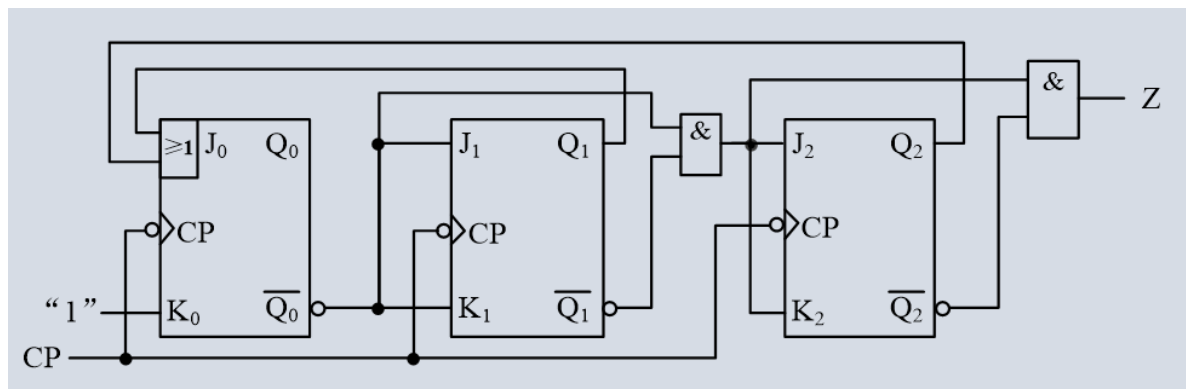
观察状态方程，可以看出选用3个JK触发器可以使激励方程最简。

$$J_2 = \overline{Q_1^n Q_0^n} \quad K_2 = \overline{Q_1^n + Q_0^n} = \overline{Q_1^n} \overline{Q_0^n}$$

$$J_1 = \overline{Q_0^n} \quad K_1 = \overline{Q_0^n}$$

$$J_0 = Q_2^n + Q_1^n \quad K_0 = 1$$

画出电路图



## 时序逻辑电路的分类

### 按工作方式分类

**同步：** 储存电路中所有触发器的时钟使用统一的clk，状态变化发生在同一时刻

**异步：** 没有统一的clk，触发器状态的变化有先有后

### 按电路输出对输入的依赖关系

**Mealy型：**  $Y = F(X, W)$  与  $X$ 、 $W$  有关

**Moore型：**  $Y = F(W)$  仅取决于电路状态

## 触发器的特性方程

### 基本 $R - S$ 触发器

$$\begin{cases} Q^{n+1} = S + \bar{R}Q^n \\ SR = 0 (\text{约束条件}) \end{cases}$$

## 基本 $\bar{R} - \bar{S}$ 触发器

$$\begin{cases} Q^{n+1} = S + \bar{R}Q^n \\ \bar{R} + \bar{S} = 1 (\text{约束条件}) \end{cases}$$

## $J - K$ 触发器

$$Q^{n+1} = J\bar{Q}^n + \bar{K}Q^n$$

## $D$ 触发器

$$Q^{n+1} = D \quad (CP = 1)$$

## $T$ 触发器

$$Q^{n+1} = T\bar{Q}^n + \bar{T}Q^n = T \oplus Q^n$$

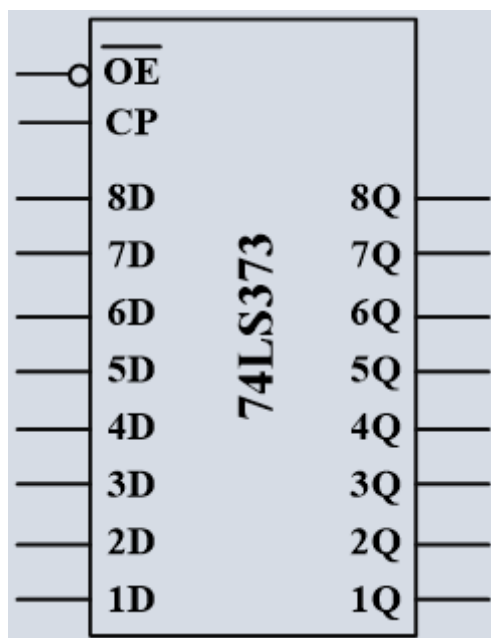
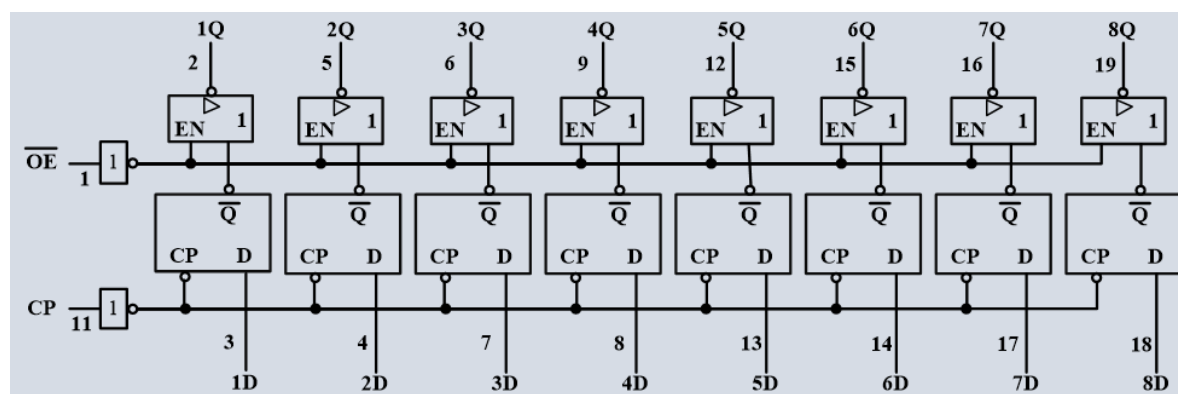
## 寄存器

看到这个概念 不要怕 寄存器(锁存器)很多时候指的就是边沿(电平) $D$ 触发器

将多个钟控锁存器用统一的时钟信号连接在一起，就可以统一控制数据的输入、储存和输出。

## 基本寄存器

74LS373：带输出使能端的8D锁存器



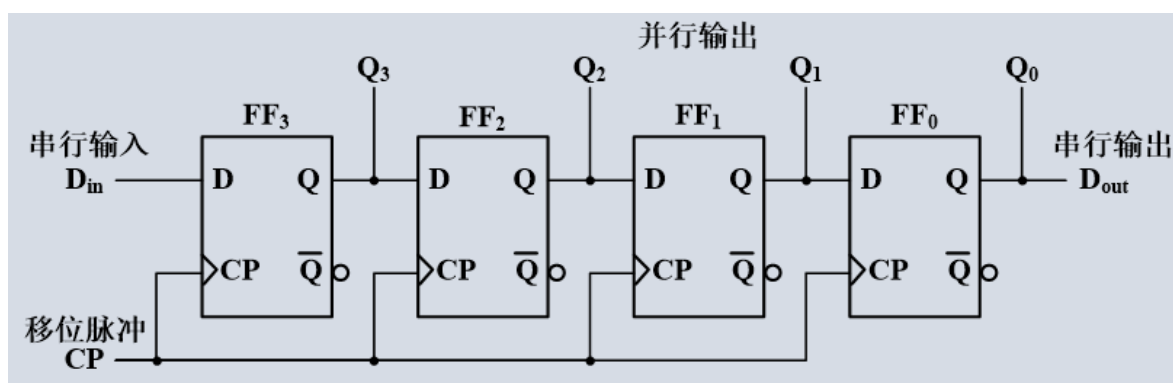
输入			输出 $Q^{n+1}$	功能
$\overline{OE}$	CP	D		
0	1	0	0	数据输入，并允许输出
0	1	1	1	
0	0	d	$Q^n$	数据保持，并允许输出
1	d	d	Z	禁止输出数据，高阻抗状态

**注意：** $\overline{OE}$ 是用来禁止输出数据的，并不会影响锁存器内部的操作。

由上面的例子可以看出，基本锁存器就是用一个统一的时钟信号将几个边沿D触发器连接起来，使得电路可以并行输入/储存/输出数据

## 移位寄存器

除了具有存储数据的功能外，还能将寄存器中的数据进行移位。所谓"移位"，就是指寄存器在一个时钟信号的驱动下，每个tick后将储存的数据向左/右移动一位，从而实现输入和输出的串行/并行转换。



CP序号	串行输入 $D_{in}$	$Q_3$	$Q_2$	$Q_1$	$Q_0$ /串行输出 $D_{out}$
0 (原始状态)	0	0	0	0	0
1	1	1	0	0	0
2	1	1	1	0	0
3	0	0	1	1	0
4	1	1	0	1	1

其结构就是将几个D触发器的输出Q和输入D首尾相接，从而实现数据的移位

举个不恰当的例子，就像流水灯电路。

学过数据结构的同学也可以用队列的概念来理解。

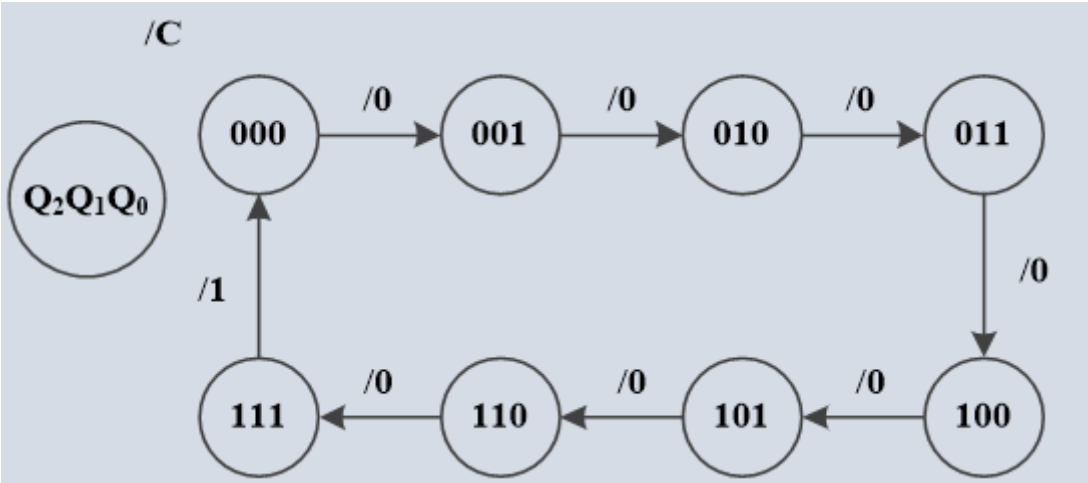
# 计数器

计数器的主要功能：实现对输入的某个信号进行计数

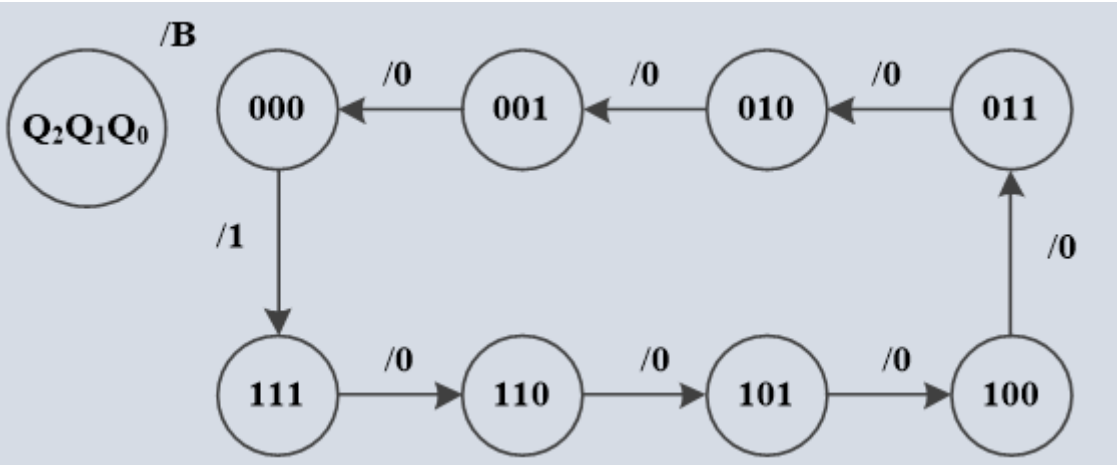
## 计数器的特点

计数器使用触发器来存储计数值，计数对象就是时钟脉冲CP（将要进行计数的输入信号），每来一个CP脉冲信号，计数器的计数值就发生相应的变化；其电路输出也通常是电路原态的函数，因此是一种Moore型时序电路。

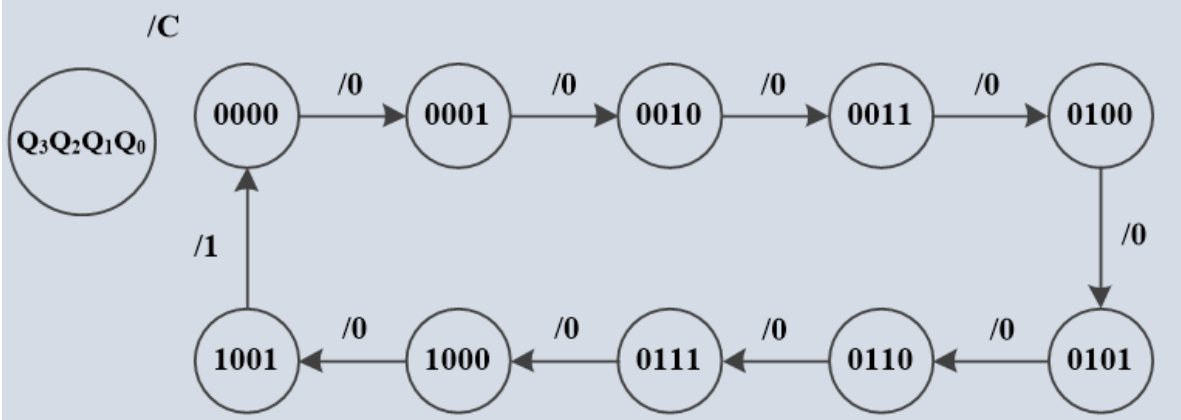
## 计数器的状态图



二进制同步加法计数器



二进制同步减法计数器



十进制同步加法计数器

可以看出，计数器的状态图一般都是环形结构

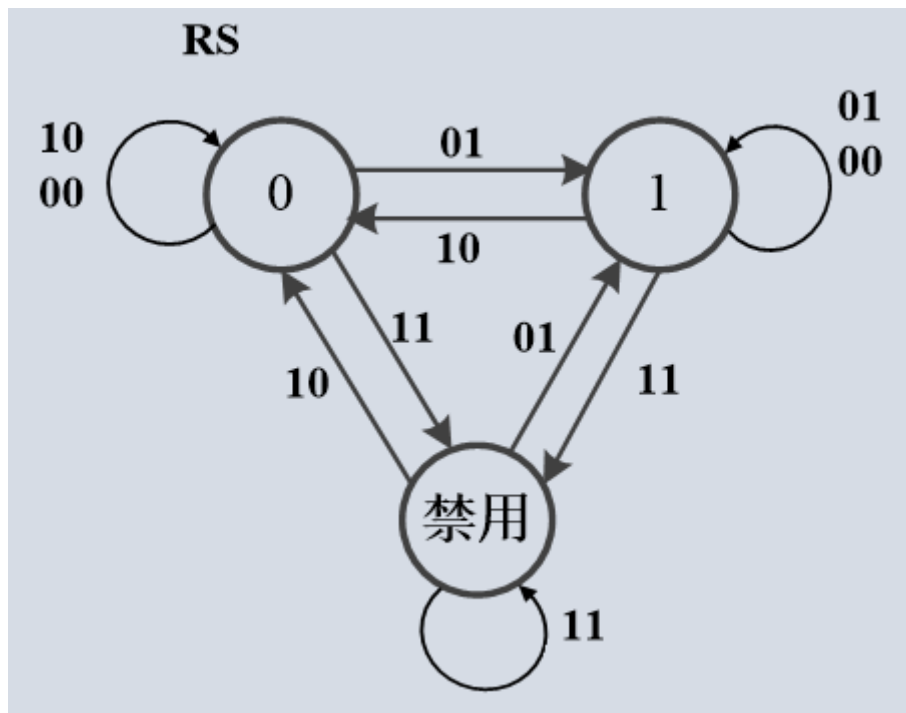
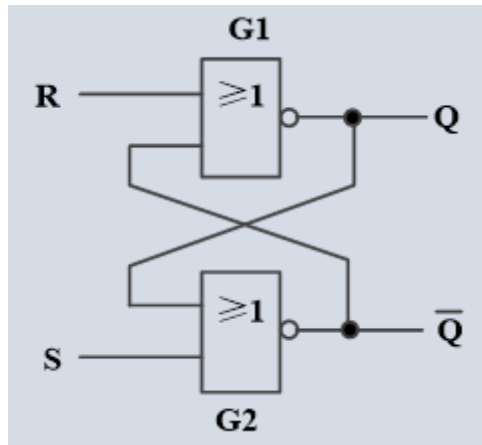


# 触发器

通常来说，触发器指的是用于储存数据的双稳态元件。一个触发器可以储存一位二进制数据。

- 根据功能特性或者激励方式（指由输入改变触发器状态的方式）的不同，可以分为 $R - S$ 触发器、 $JK$ 触发器、 $D$ 触发器、 $T$ 触发器和 $T'$ 触发器。
- 根据电路结构的不同，可以分为基本 $R - S$ 锁存器、钟控锁存器、主从式触发器和边沿触发器。
- 根据触发方式（指触发器状态变化时刻的控制方式）的不同，又可以分为电平触发、主从触发和边沿触发的触发器。

## 基本 $R - S$ 触发器



R	S	$Q^n$	$Q^{n+1}$	$\overline{Q^{n+1}}$	功能描述
0	0	0	0	1	保持
0	0	1	1	0	保持
0	1	0	1	0	置位
0	1	1	1	0	置位
1	0	0	0	1	复位
1	0	1	0	1	复位
1	1	0	d	d	禁用
1	1	1	d	d	禁用

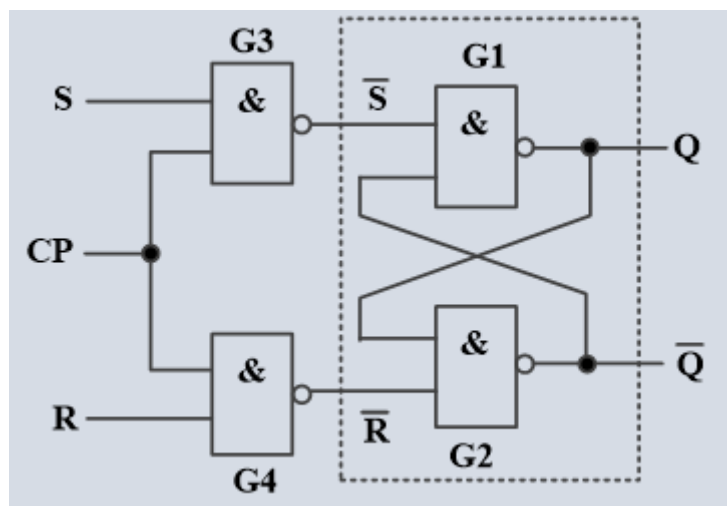
当基本  $R - S$  触发器的两个输入均为1时，触发器的状态无法判断。

将基本  $R - S$  锁存器中的或非门更改成与非门实现，就变成了基本  $\bar{R} - \bar{S}$  锁存器。

基本  $\bar{R} - \bar{S}$  锁存器是低电平触发的：

$\bar{R}$	$\bar{S}$	$Q^{n+1}$	功能描述
0	0	d	禁用
0	1	0	复位
1	0	1	置位
1	1	$Q^n$	保持

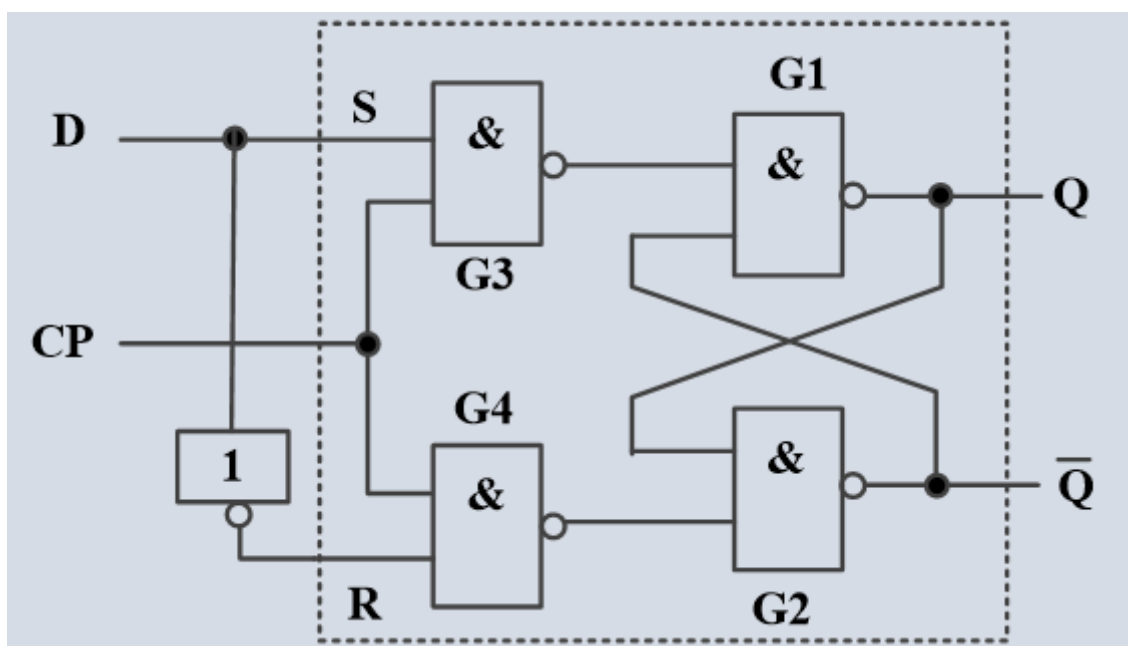
### 钟控 $R - S$ 触发器



将 $\bar{R} - \bar{S}$ 锁存器的两个输入端分别接上两个与非门，再接入一路时钟信号，就是钟控 $\bar{R} - \bar{S}$ 锁存器了。其特点是只有在 $CP$ 为1的情况下才可以触发。

CP	R	S	$Q^{n+1}$	功能描述
0	d	d	$Q^n$	保持
1	0	0	$Q^n$	保持
1	0	1	1	置位
1	1	0	0	复位
1	1	1	d	禁用

### 钟控D触发器



CP	D	$Q^{n+1}$	功能描述
0	d	$Q^n$	保持
1	0	0	复位
1	1	1	置位

功能非常简单，见上表

### 钟控触发器的空翻

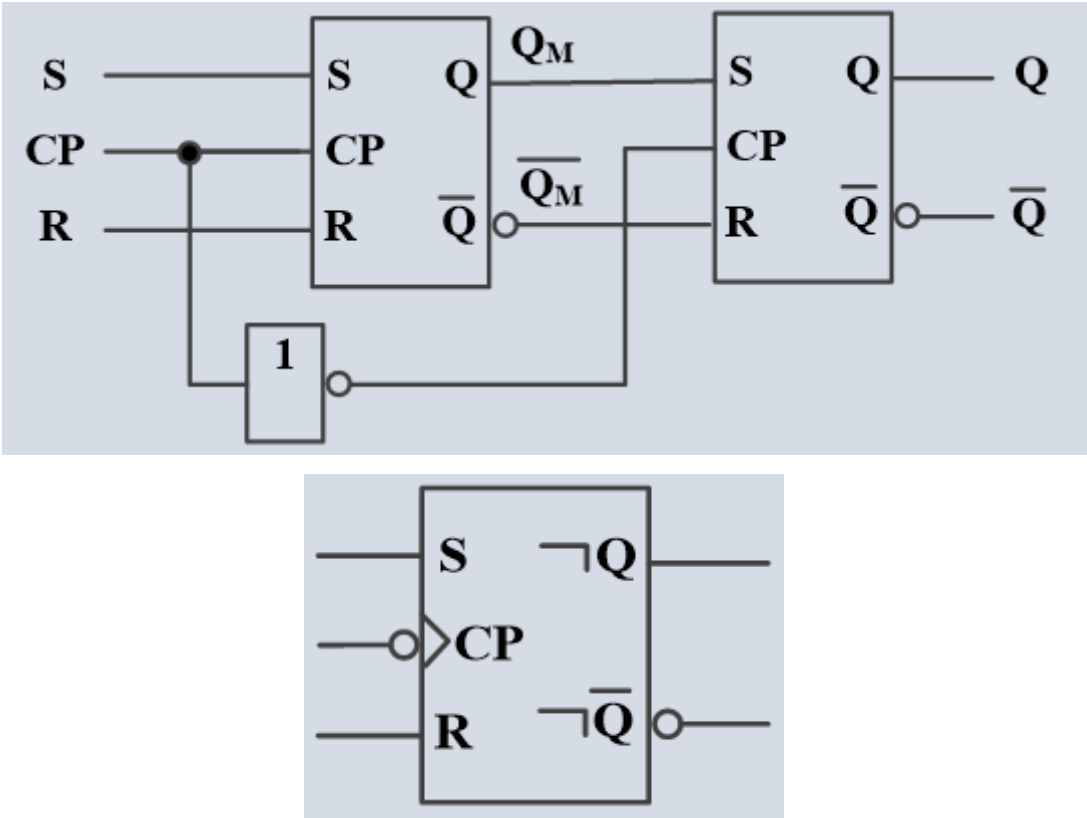
在钟控R-S锁存器的使用过程中，如果违反了 $RS=0$ 的约束条件，即 $R=S=1$ 时，则可能出现下列不正常的情况之一：





- 当 $CP = 1$ 时，如果 $RS = 11$ ，则将出现 $Q = \bar{Q} = 1$ 的非法状态；
- $CP = 1$ 时，如果 $RS = 11$ ，然后R和S同时从1变为0( $RS = 00$ )，则锁存器下一状态不可预测，结果不确定；
- 当 $RS = 11$ 时，如果 $CP$ 突然从1变为0，锁存器也会出现结果不确定的情况，下一状态不可预测，输出可能进入亚稳态。

因此，对于钟控R-S锁存器而言， $R = S = 1$ 既没有意义，同时也是禁用状态。

### 主从触发器

为了解决空翻现象，将一个触发器的输出接到另一个触发器的输入，同时接上时钟信号，就是主从触发器。

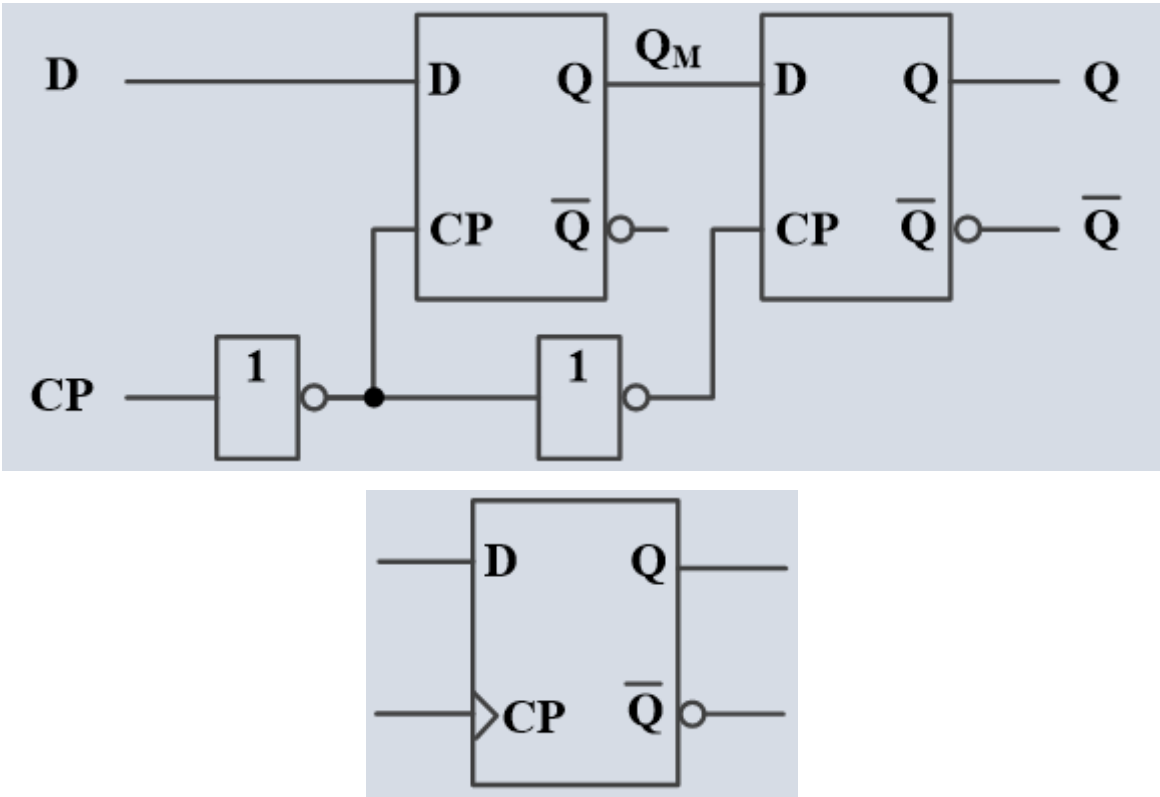




CP	R	S	$Q^{n+1}$	功能描述
0	d	d	$Q^n$	保持
	0	0	$Q^n$	保持
	0	1	1	置位
	1	0	0	复位
	1	1	d	禁用

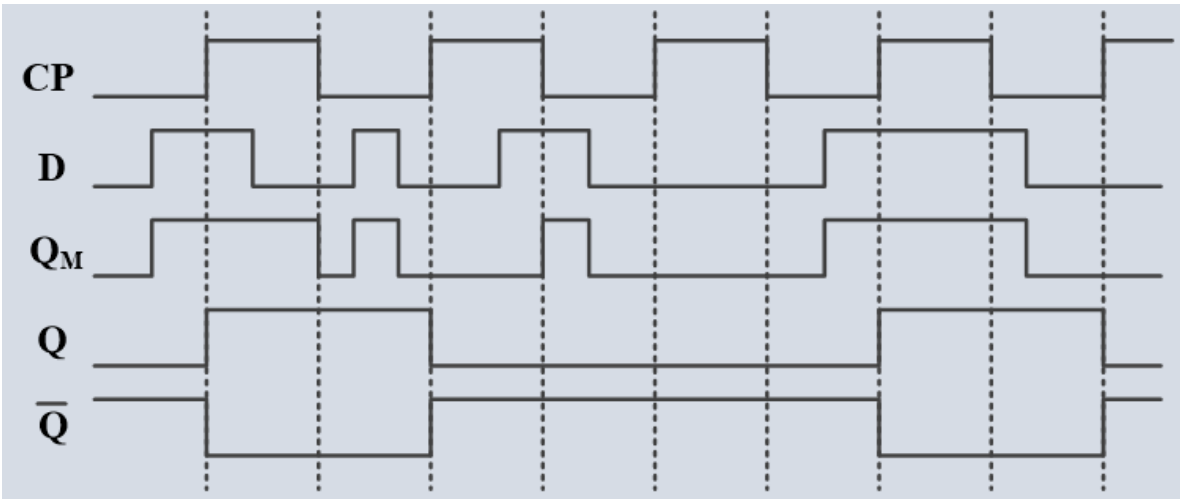
主从R－S触发器

边沿触发器

只在上升沿或下降沿触发的触发器



CP	D	$Q^{n+1}$	功能描述
0	×	$Q^n$	保持
1	×	$Q^n$	保持
	0	0	复位
	1	1	置位



边沿D触发器

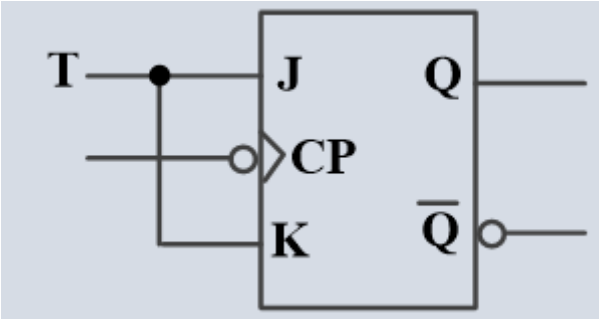
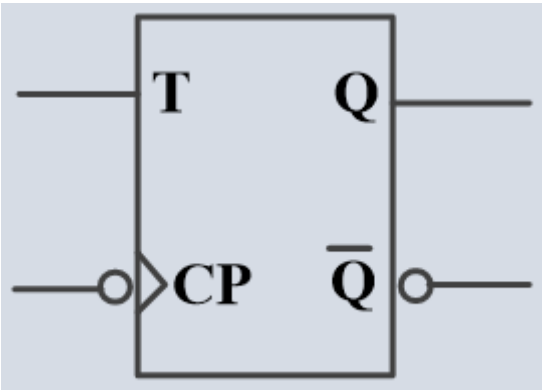
n边沿D触发器的主要特点是消除了约束条件，解决了空翻问题，在CP有效边沿的一瞬间发生状态变化，抗干扰能力强，功能简单；

n只有一个输入端，特性方程简单，在有些情况下，用D触发器设计的电路，可能会比较复杂。

### T、T'触发器

T触发器的激励输入信号为T，当 $T=0$ 时 $Q^{n+1} = Q^n$ ， $T=1$ 时 $Q^{n+1} = \overline{Q^n}$ ，因此T触发器具有保持和翻转功能。

一般没有专门的T触发器，它通常是将J-K触发器的J和K输入端并接在一起，作为T输入端，从而构成T触发器。

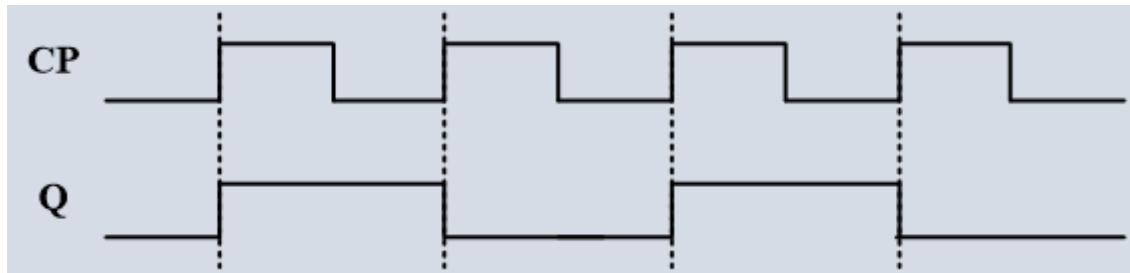
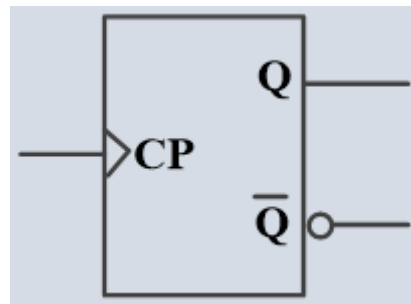


J-K触发器构成T触发器

CP	T	$Q^{n+1}$	功能描述
	0	$Q^n$	保持
	1	$\overline{Q^n}$	翻转

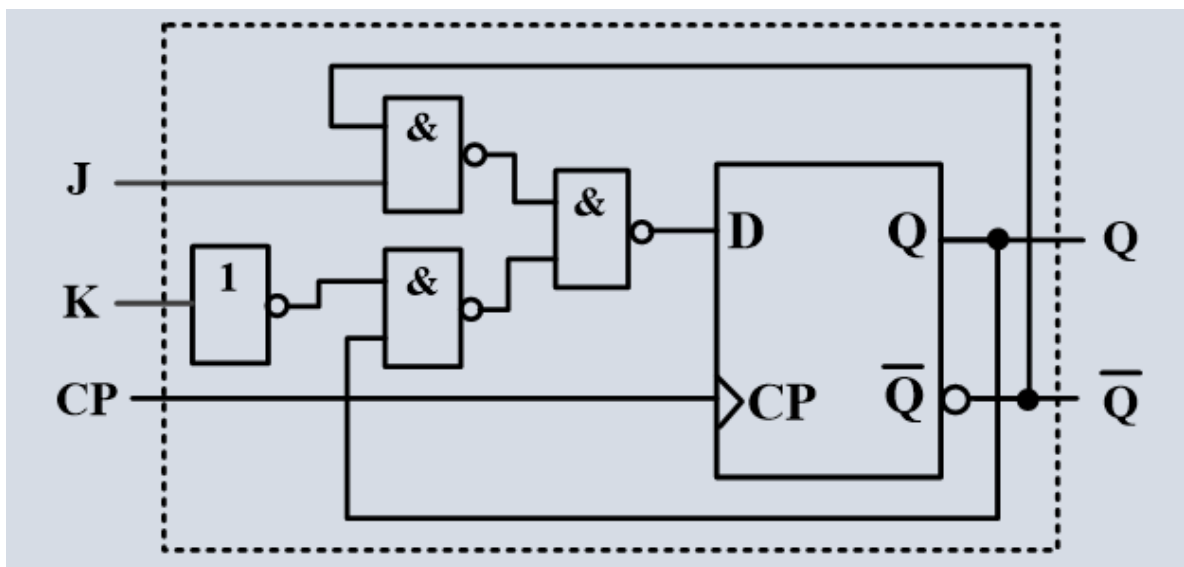
将T触发器的T输入端恒接逻辑“1”，则在每个时钟有效边沿，触发器的输出Q均进行翻转，这便是T'触发器（Toggle Flip-flop）

T'触发器没有激励输入端，其特性方程为： $Q^{n+1} = \overline{Q^n}$

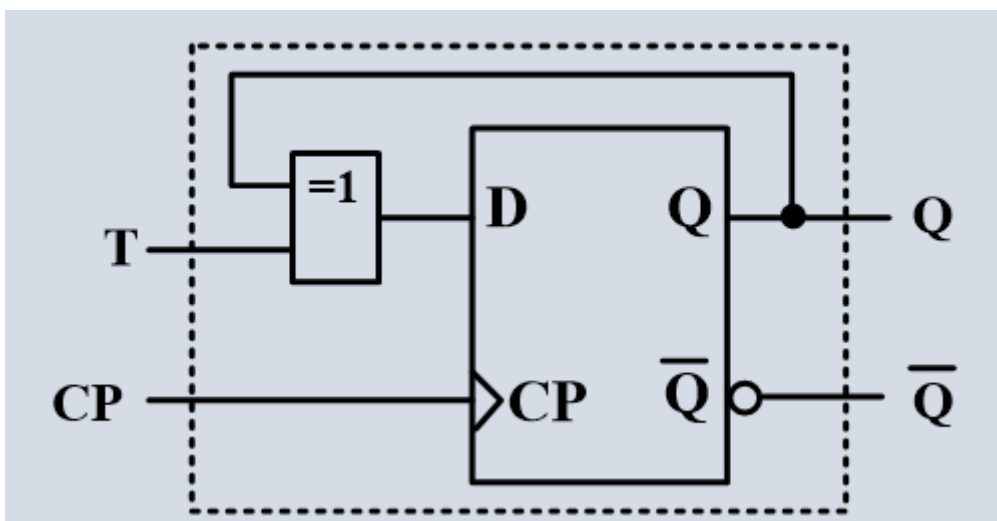


## 触发器之间的转换

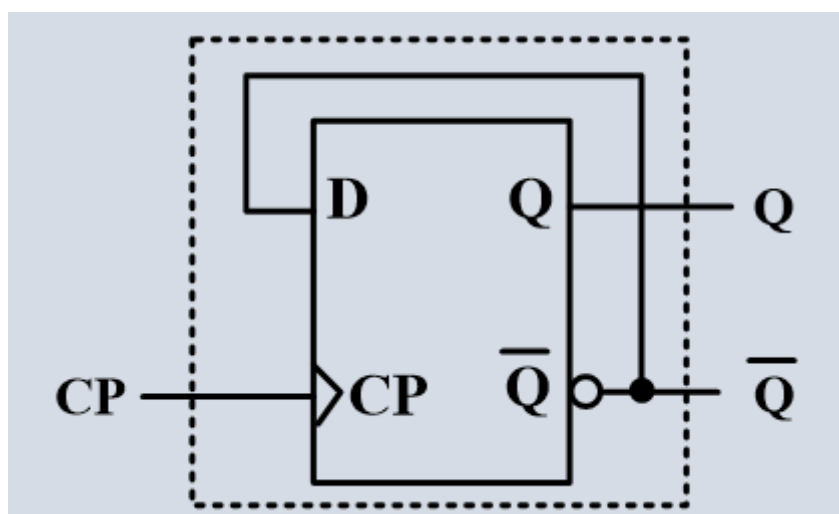
*D*触发器转换为*JK*触发器



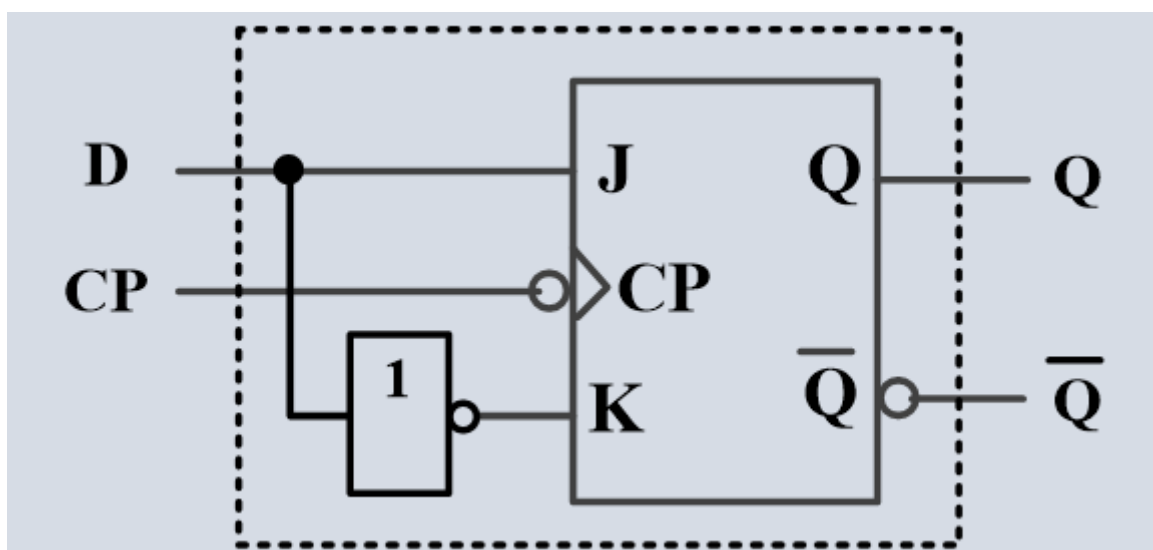
*D*触发器转换为*T*触发器



$D$ 触发器转换为 $T'$ 触发器

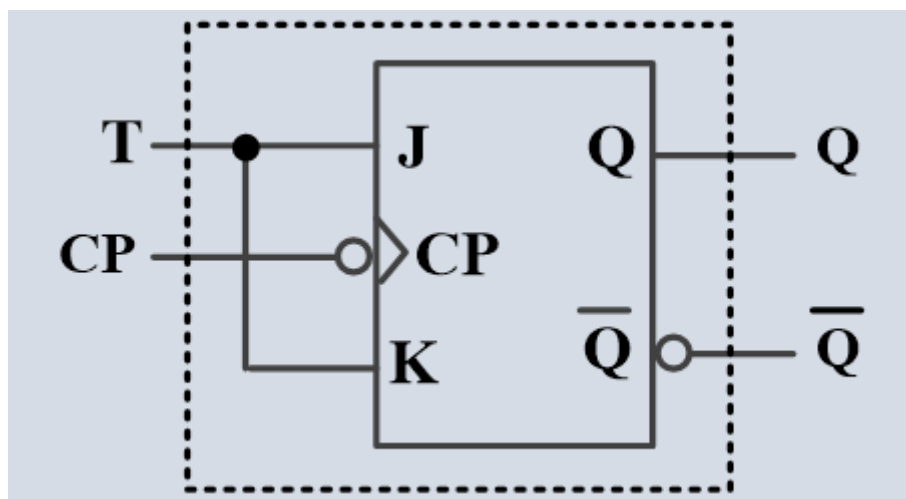


$JK$ 触发器转换为 $D$ 触发器

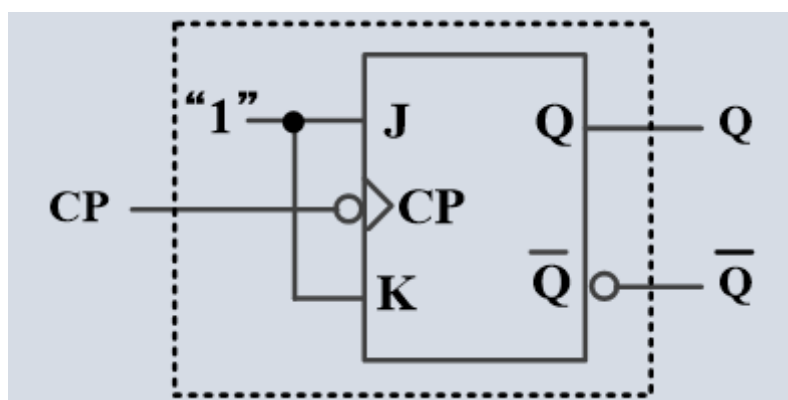


$JK$ 触发器转换为 $T$ 触发器





*JK*触发器转换为*T'*触发器



n只有一个输入端，特性方程简单，在有些情况下，用D触发器设计的电路，可能会比较复杂。