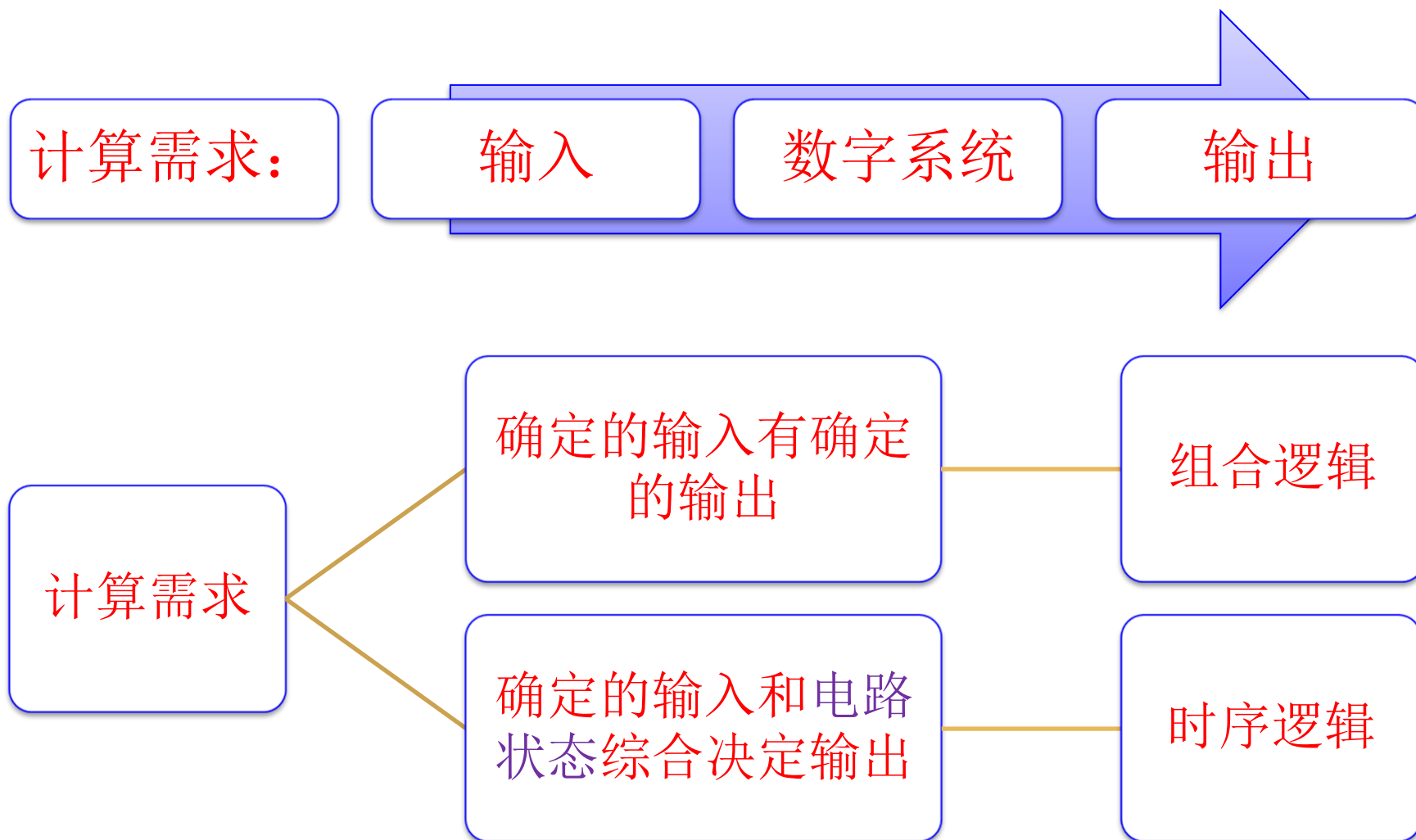
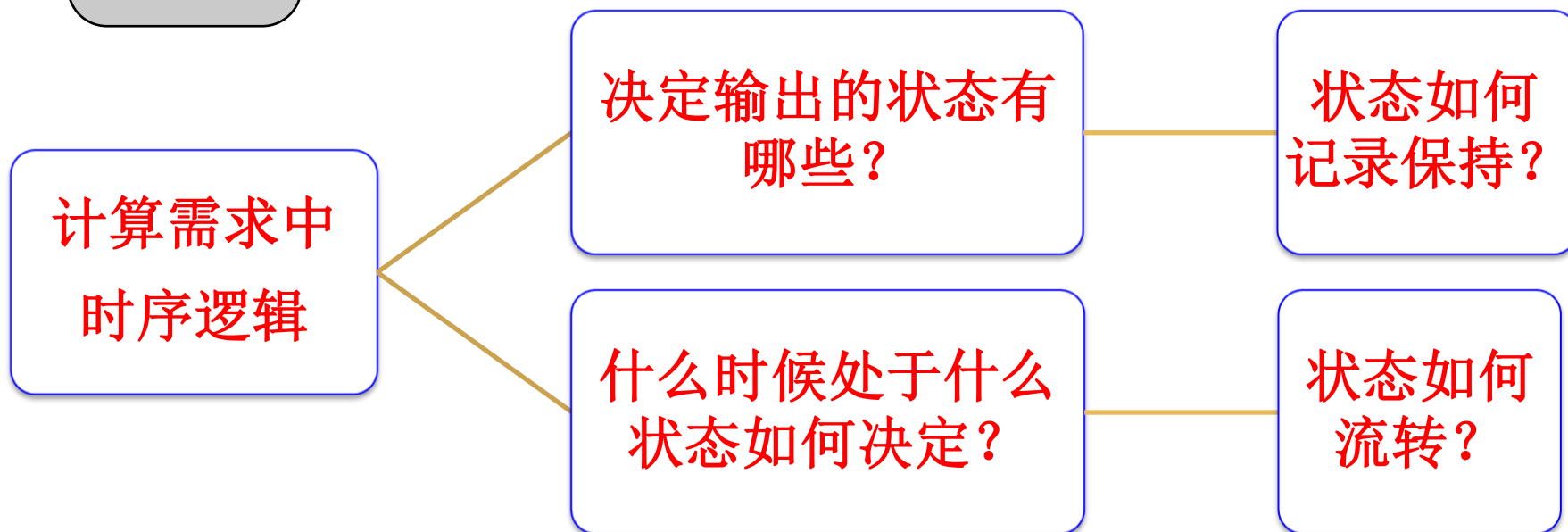
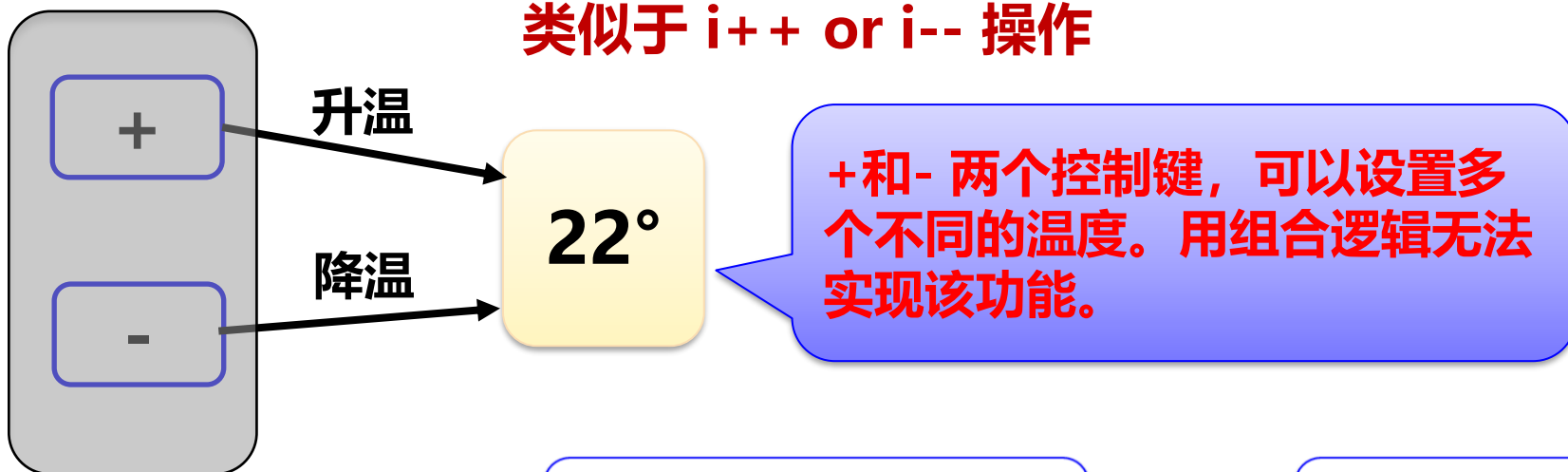


# 第4章 时序逻辑电路

- 第一讲 时序逻辑电路概述
- 第二讲 锁存器和触发器
- 第三讲 同步时序电路设计
- 第四讲 典型时序逻辑部件设计



类似于  $i++$  or  $i--$  操作



# 第一讲 时序逻辑电路概述

- ◆ 时序逻辑与有限状态机
- ◆ 时序逻辑电路的基本结构
- ◆ 时序逻辑电路的定时

# 1 时序逻辑电路概述

- ◆ 组合逻辑：输出结果仅取决于当前的输入信号
- ◆ 时序逻辑：输出结果不仅取决于**当前时刻的输入值**，而且取决于电路**过去时刻的行为（当前状态）**
  - 电路中有**存储元件**，用于记录电路**过去时刻的行为（当前状态）**
  - 当电路输入值发生变化时，新的输入值可能使得电路保持当前状态，也可能使得电路状态发生改变，进入新的状态
- ◆ 时序逻辑：
  - 表征工具
  - 状态记忆
  - 状态流转

# 1.1 时序逻辑与有限状态机

## ◆ 用有限状态机刻画时序逻辑。

- 有限状态机(Finite State Machine, FSM)是一种刻画状态及状态转换的理论工具。

## ◆ 通常用状态图描述有限状态机。

- 状态：用包含状态符号的圆圈表示
- 状态转换方向：用有向边表示，并在边上标注引起状态变换的输入信号值和相应输出响应（若输出响应只与当前状态有关，则把输出响应标注在状态圆圈中）

例：检测输入序列是否为连续4个“1”

A-初始态：若输入1，则转B

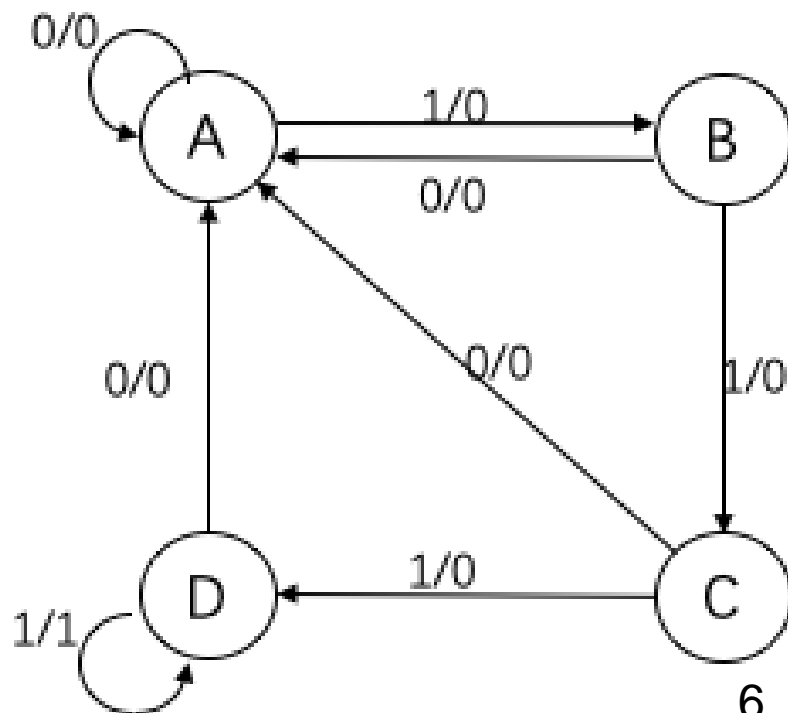
B-连续1个“1”：若输入1，则转C

C-连续2个“1”：若输入1，则转D

D-连续3个“1”：若输入1，则状态不变

并输出为1（表示检测到连续4个1）

任何状态下，输入0都会转到初态A



# 1.1 时序逻辑与有限状态机

- ◆ 用数字逻辑实现一个有限状态机（时序逻辑），需要完成工作：
  - 设计实现一种电路能进行**状态的记忆**。
    - 可使用双稳态器件来记忆状态，如**SR锁存器**、**D触发器**、**JK触发器**等；
    - 也可使用电路的稳态来实现，如反馈时序逻辑电路等。
  - 把状态机的输入、输出以及内部状态都转换成二进制表示。
    - 这里的关键是**状态的二进制编码**。
  - 设计出符合状态转换逻辑要求的状态记忆器件的**激励函数和输出函数**，并完成**定时分析**。

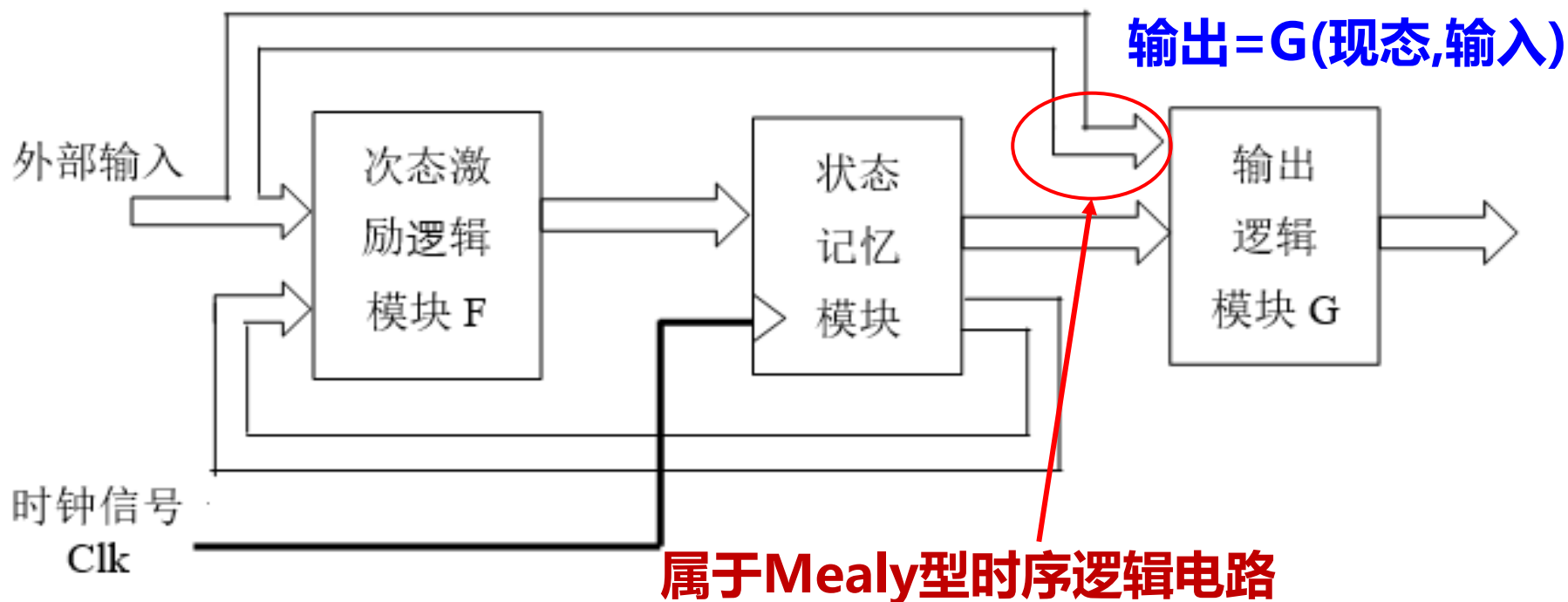
能完成上述工作（实现有限状态机）的数字逻辑电路是**时序逻辑电路**！

# 1.2 时序逻辑电路基本结构

## ◆ 时序逻辑电路的一般结构

- 状态记忆模块：由多个状态记忆单元构成
- 次态激励逻辑模块F：激励函数（现态和外部输入的逻辑函数）
- 输出逻辑模块G：输出函数（现态和外部输入的逻辑函数）

**Mealy型：**输出依赖于当前状态和当前输入信号

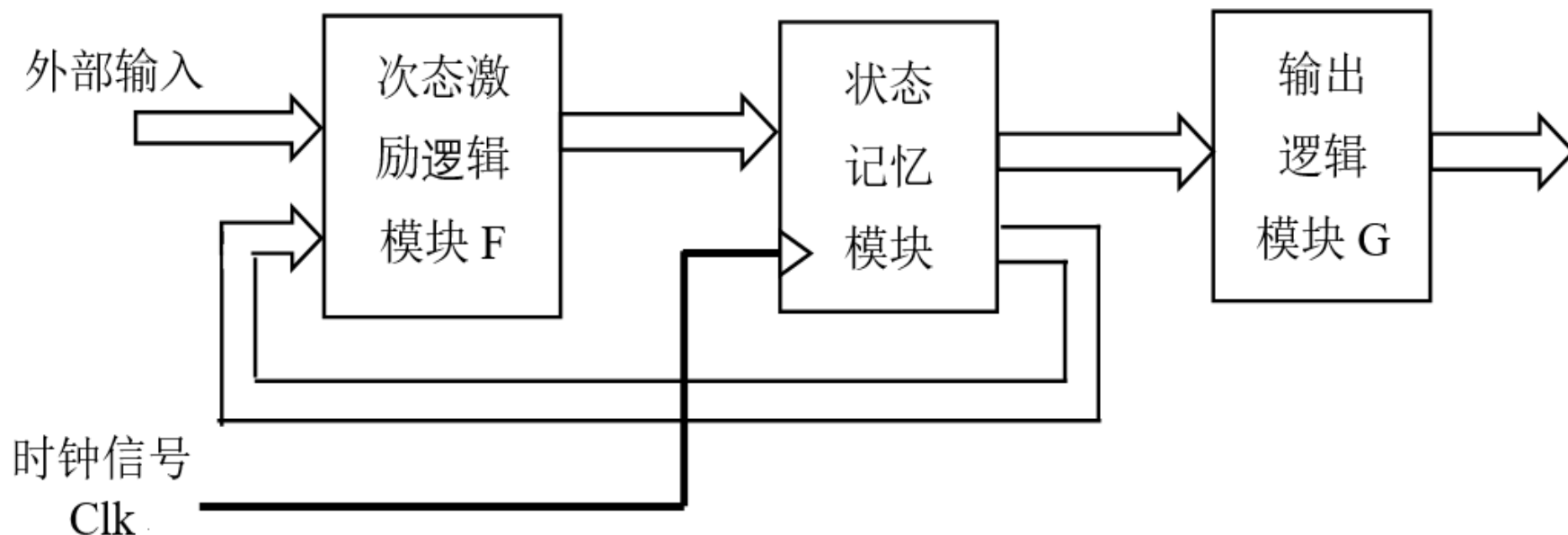




# 1.2 时序逻辑电路基本结构

**Moore型：**输出仅依赖于当前状态，和当前输入信号无关

输出 = G(现态)

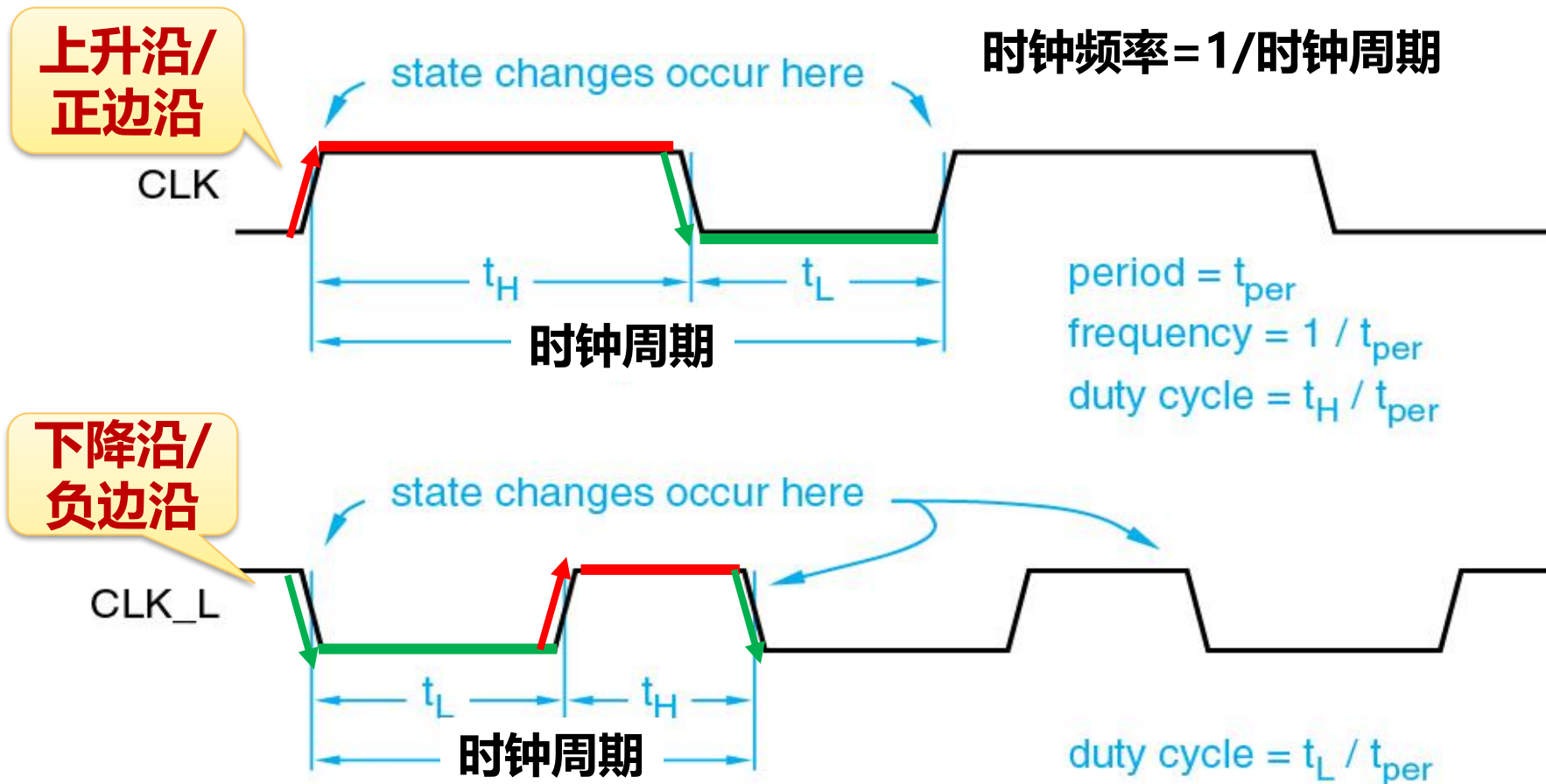


- ◆ 根据状态转换方式的不同有**同步时序**逻辑电路和**异步时序**逻辑电路
  - **同步时序逻辑电路：**在统一的时钟信号控制下进行状态转换
  - **异步时序逻辑电路：**没有统一的时钟信号来控制状态的改变

# 1.3 时序逻辑电路的定时

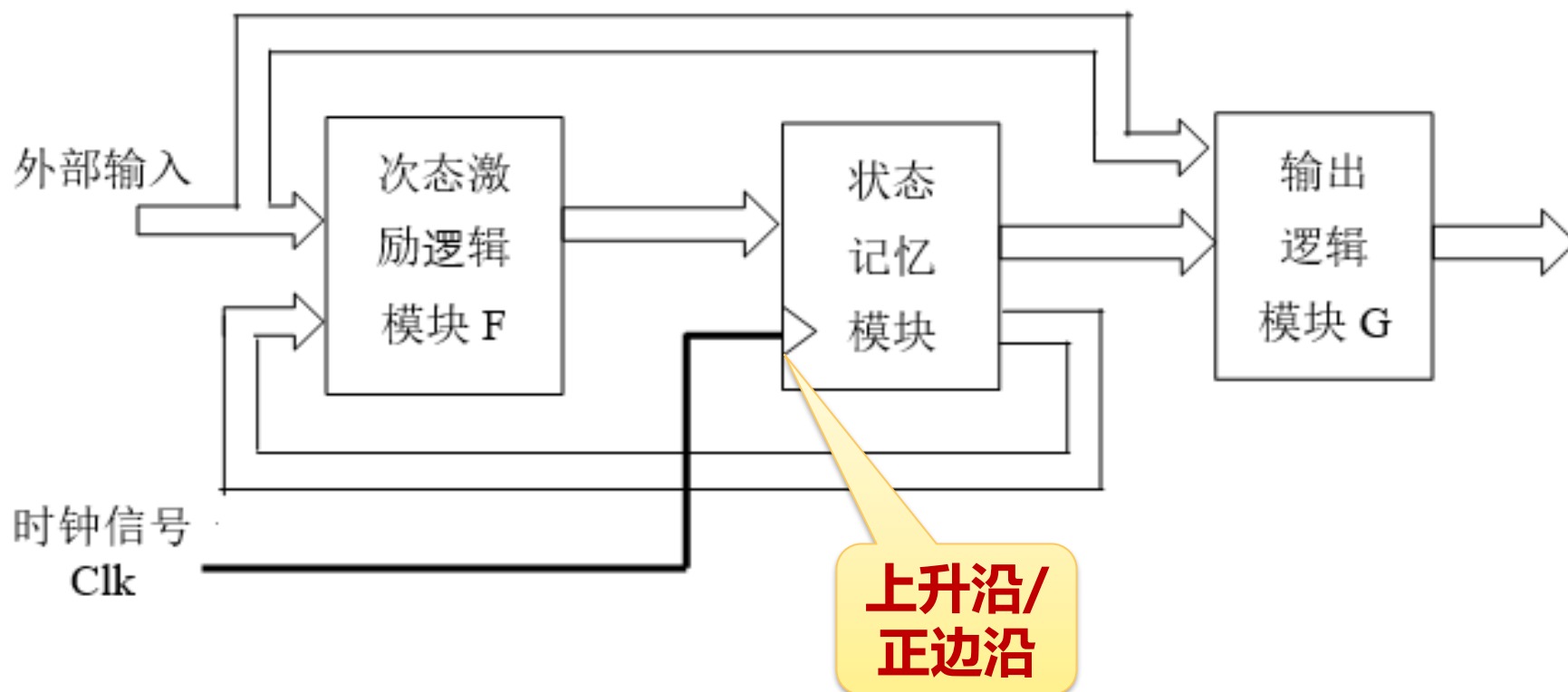
◆ 什么时候状态会发生变换？ **电平触发或边沿触发**

- 时序逻辑电路的状态多采用**边沿触发方式**
- 边沿触发方式分为**上升沿触发**和**下降沿触发**两种类型



# 1.3 时序逻辑电路的定时

- ◆ 什么时候状态会发生变换？ **电平触发或边沿触发**
  - 时序逻辑电路的状态多采用**边沿触发方式**
  - 边沿触发方式分为**上升沿触发**和**下降沿触发**两种类型



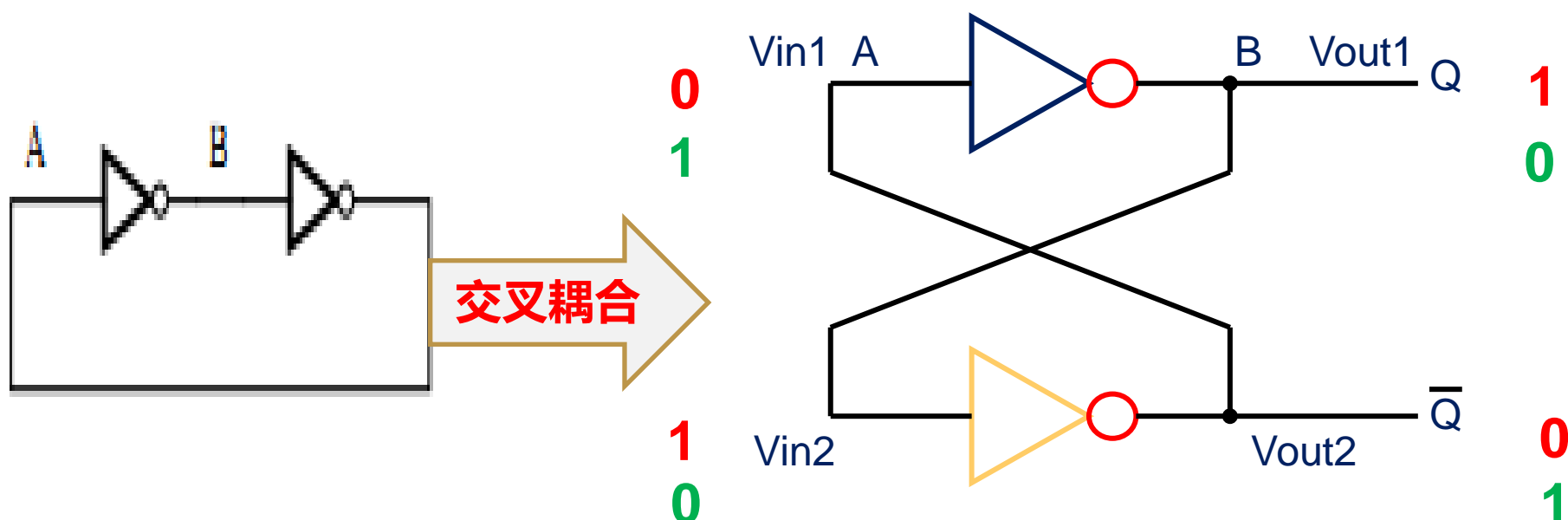
## 第二讲 锁存器和触发器

状态记忆  
器件设计

- ◆ 双稳态元件
- ◆ SR锁存器
- ◆ D锁存器
- ◆ D触发器
- ◆ T触发器

## 2.1 双稳态元件

- ◆ 用于存储1位二进制数据，有两个**互补**的输出状态
  - 状态 1：置位(Set)状态，表示存储逻辑“1”
  - 状态 0：复位(Reset)状态，Q为低电平，表示存储逻辑“0”
- ◆ 双稳态元件的简单实现
  - 串联两个反相器，则反相器的输出状态不同，且保持稳定
  - Q为高电平时，为置位状态；Q为低电平时，为复位状态



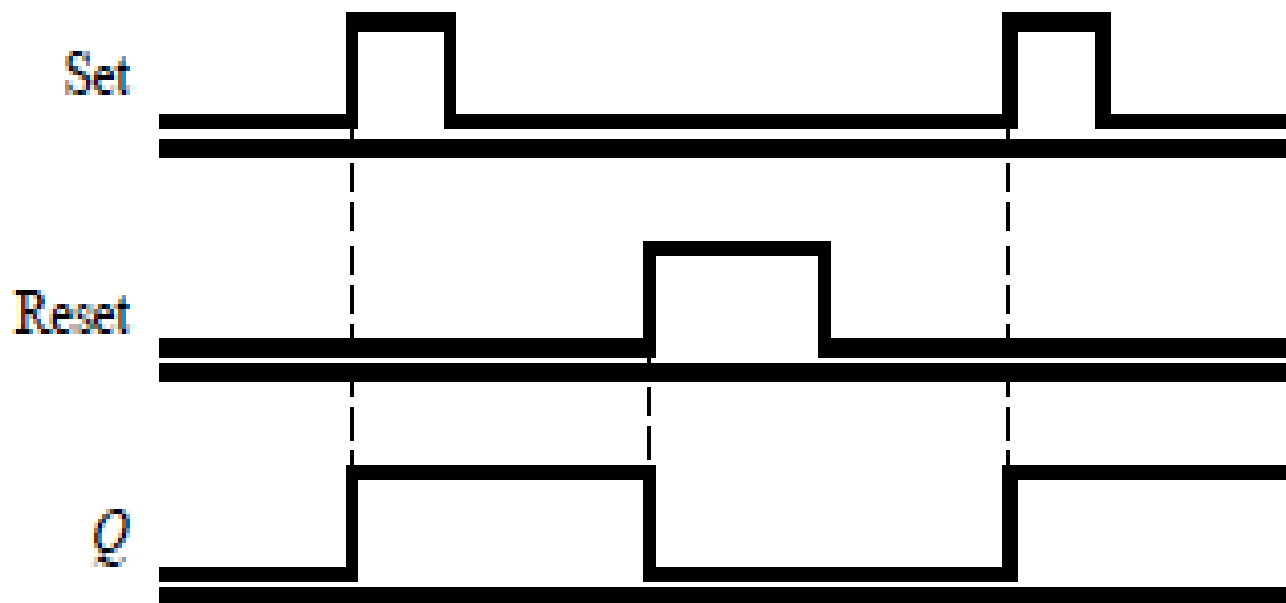
基于简单双稳态元件构建思路，可实现锁存器和触发器

## 2.1 双稳态元件

- ◆ 用两个反相器串联构建的双稳态元件，由于无法改变电路的状态，因此功能受到限制。
- ◆ 用1个或多个输入信号能**驱动**双稳态元件进入特定的**稳定状态**，这样的双稳态元件才能成为实用的存储元件，这些输入信号被称为**激励信号或激励输入**。
  - 通常根据不同的激励输入信号来命名存储元件，如SR、JK、D、T等不同的激励输入信号。
- ◆ 根据存储元件触发方式的不同，分成两种类型：
  - **电平触发：锁存器(latch)**
  - **边沿触发：触发器(flip-flop)**

## 2.1 双稳态元件

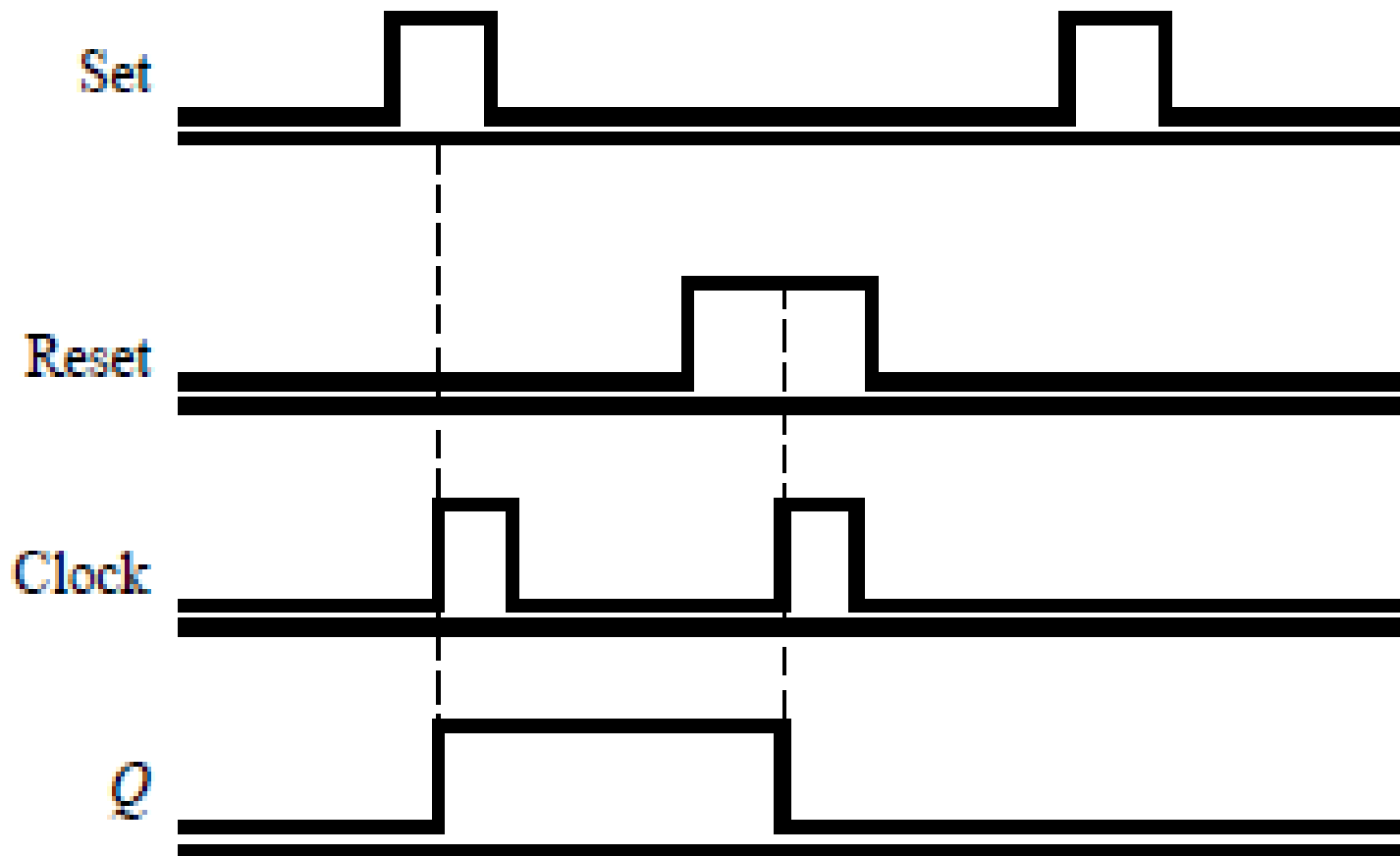
- ◆ 锁存器 (latch) : 通过激励输入的**电平信号**来控制存储元件的状态
- ◆ **置位复位**锁存器(Set-Reset latch): 具有置位和复位激励信号
  - 置位激励信号Set**有效时**, 强制存储元件的输出Q为**1**
  - 复位激励信号Reset**有效时**, 强制存储元件的输出Q为**0**



## 2.1 双稳态元件

### ◆ 触发器 flip-flop

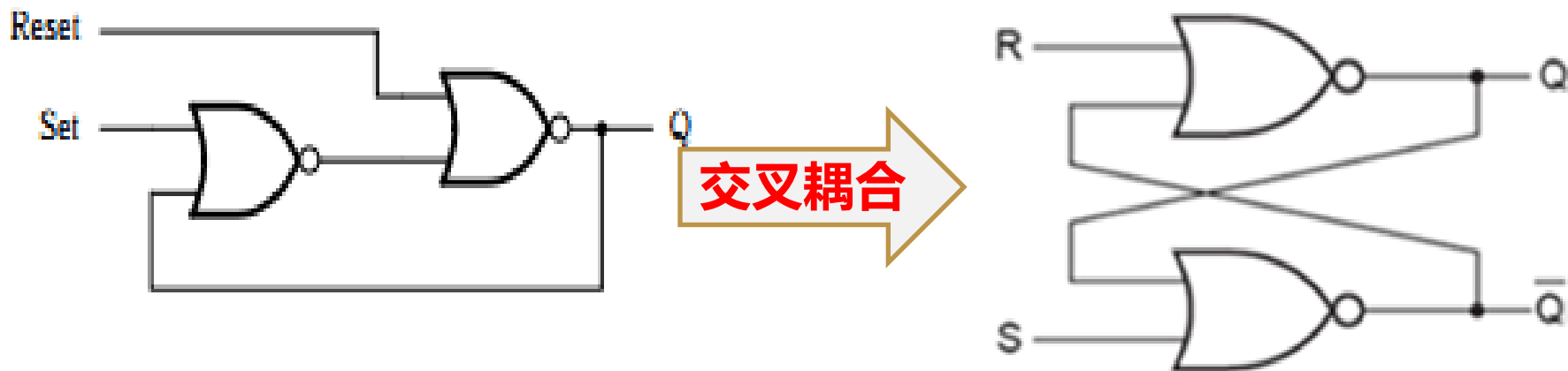
- 具有时钟控制信号(clock)
- 通过时钟信号的**边沿**来触发存储元件改变状态



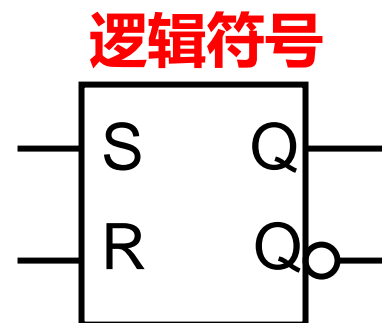
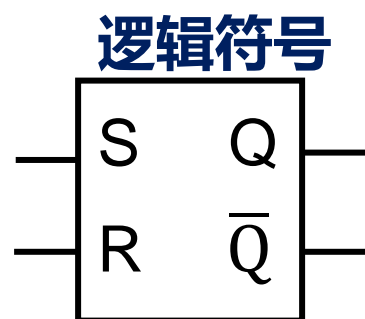


## 2.2 SR锁存器

- ◆ SR锁存器：使用一对交叉耦合的或非门构成双稳态电路，也称为置位-重置（复位）锁存器。S是置位输入端，R是重置输入端



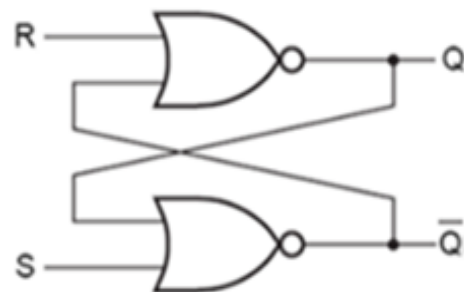
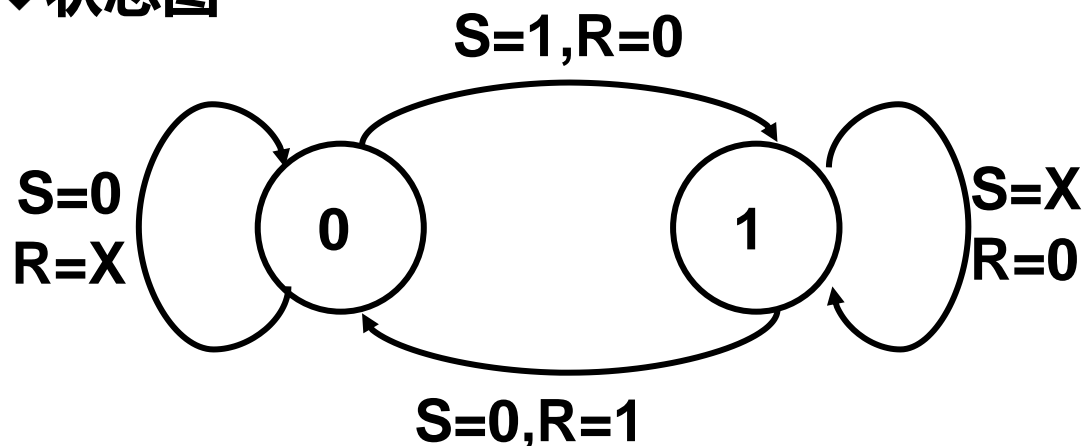
S	R	Q	$\bar{Q}$
0	0	状态不变	
0	1	0	1
1	0	1	0
1	1	禁止	



R=S=1时，Q、 $\bar{Q}$ 状态不相反，无效

## 2.2 SR锁存器

### ◆ 状态图



状态表

现态 Q	输入RS			
	00	01	10	11
0	0	1	0	0*
1	1	1	0	0*

### ◆ 通过状态变化来描述时序电路

### ◆ 构建状态表

- 顶部为输入信号，左侧为现态Q
- 右侧填入次态Q\*和输出信号

### ◆ 在置位态下，若R输入变为高电平，则经过两级门延迟变为复位态

### ◆ 从输入驱动信号有效开始，到输出达到稳定为止有一定的延迟，这个延迟称为触发延迟或锁存延迟。

## 2.2 SR锁存器

### ◆ 状态表转换成状态转移表

状态表	
现态 Q	次态Q*
	输入RS
	00 01 10 11
0	0 1 0 0
1	1 1 0 0



状态图、状态表、  
特征方程之间可  
相互转换！

状态转移表

S	R	现态Q	次态Q*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0* d
1	1	1	0* d

### ■ 特征方程

$$\text{次态方程} \begin{cases} Q^* = S + \bar{R} \cdot Q \\ S \cdot R \neq 1 \quad \text{约束条件} \end{cases}$$

**功能：SR锁存器常用来设置标志位**

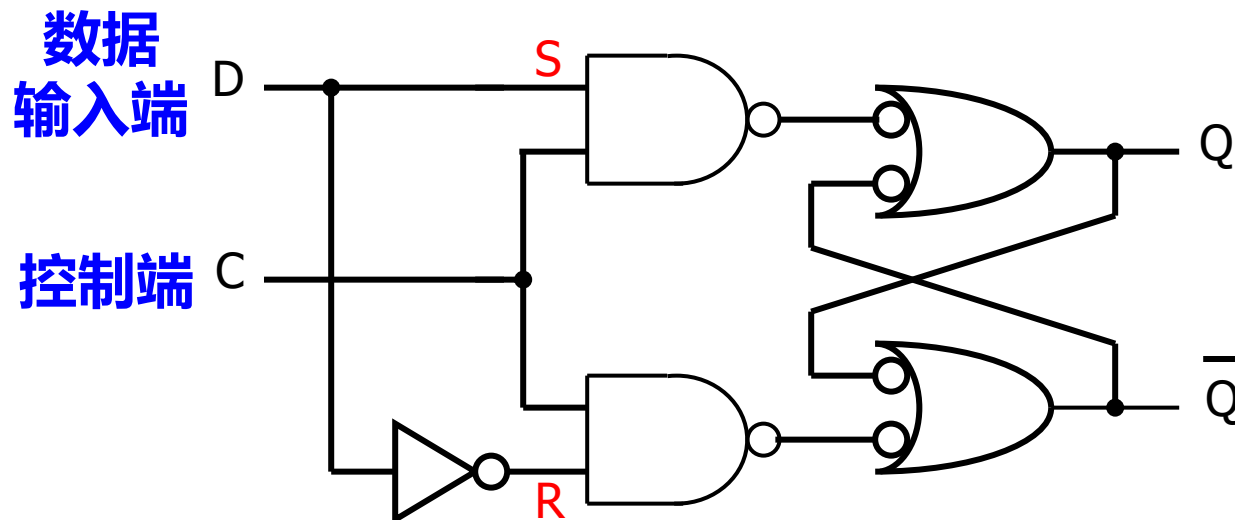
## 2.3 D锁存器

$S=1, R=0$ 时,  $Q=1$

$S=0, R=1$ 时,  $Q=0$

$S$ 、 $R$ 输入互补

◆ 如何利用锁存器来设置、储存信息位？



D锁存器功能表

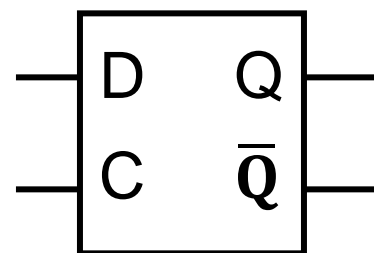
C	D	Q	$\bar{Q}$
0	X	保持	
1	0	0	1
1	1	1	0

$C=0$ 时, 输出状态保持不变

$C=1$ 时,  $\left. \begin{array}{l} D=1 \text{ 时, } Q=1 \\ D=0 \text{ 时, } Q=0 \end{array} \right\} Q=D$

输出随输入状态而改变

逻辑符号



只有一个数据输入端D, 称为D锁存器, 也称为透明锁存器

## 2.3 D锁存器

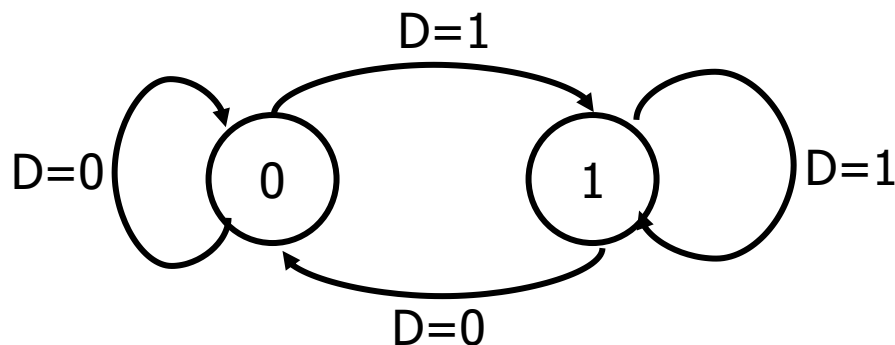
### ◆ D锁存器状态表、状态图和特征方程

状态转移表

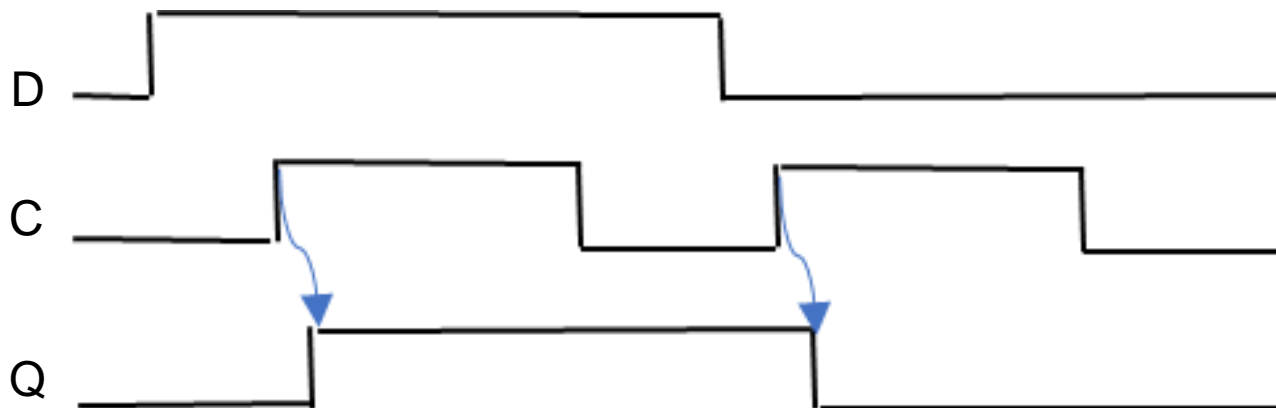
D	$Q^*$
0	0
1	1

特征方程:  $Q^* = D$  ( $C=1$ )

状态图

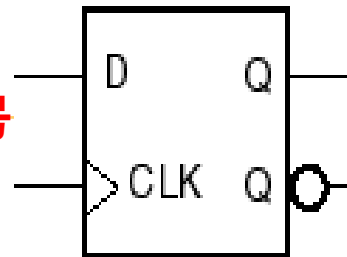


### ◆ D锁存器的时序图

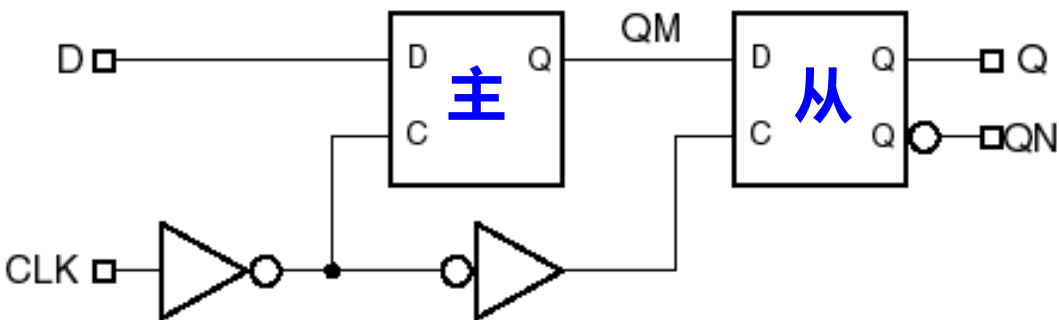


# 2.4 D触发器

D触发器符号



◆由一对主、从D锁存器可构成一个D触发器

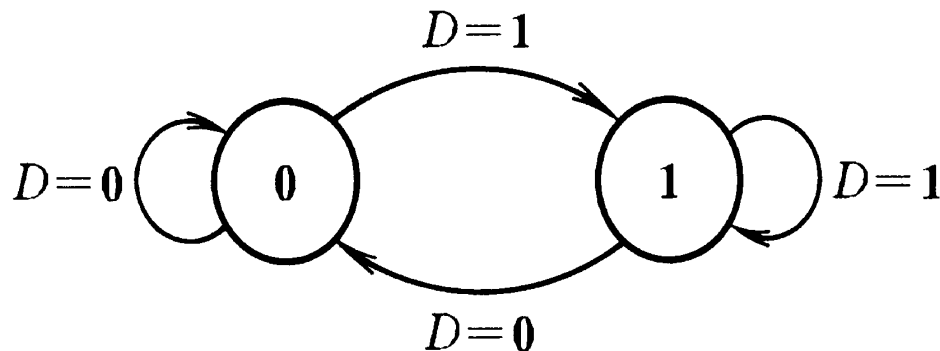


CLK	主锁存器	从锁存器
L	写入	不变
上升沿	锁存	开始写入
H	不变	写入

- 从锁存器只在时钟CLK的上升沿到来时采样主锁存器的输出QM的值，并确定Q和QN的输出

D	CLK	Q	QN
0		0	1
1		1	0
x	0	last Q	last QN
x	1	last Q	last QN

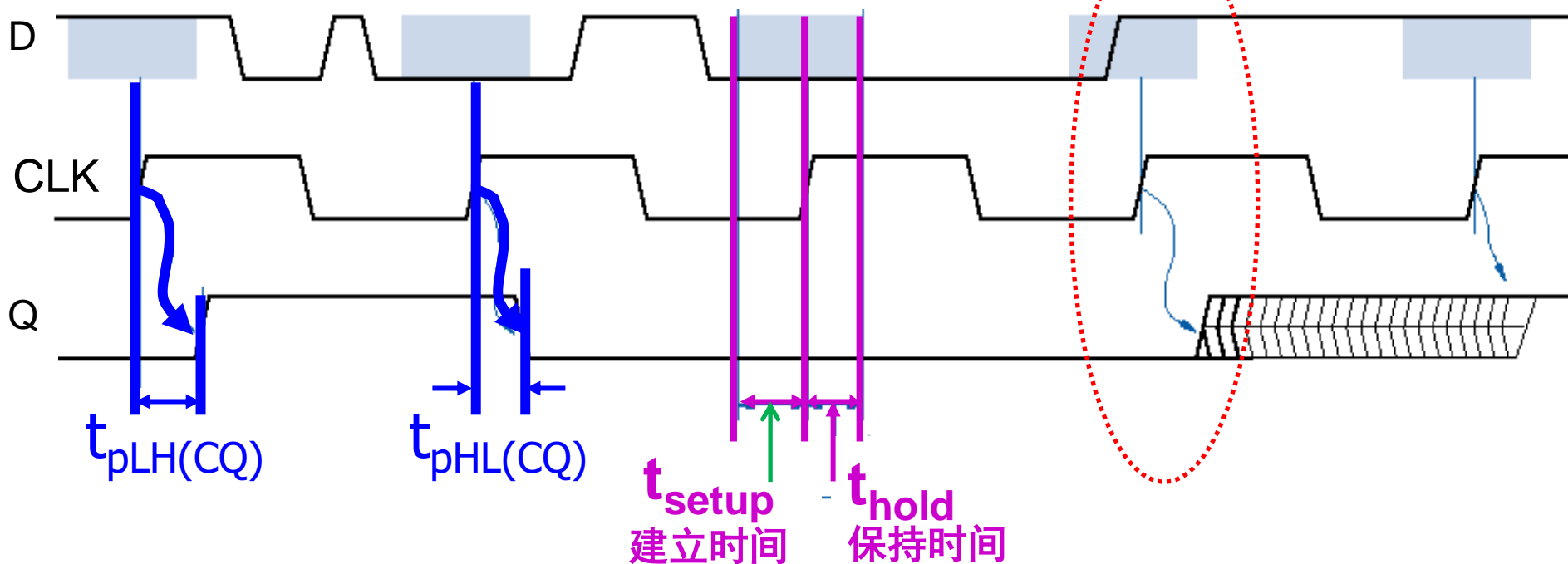
◆状态转移图



◆D触发器特征方程:  $Q^* = D$

## 2.4 D触发器

窗口期内D输入的改变导致输出不可预测



- ◆ 从时钟触发边沿到来,到输出端Q改变为D值的时间称为**锁存延迟** $t_{CQ}$  (latch prop), 即**CLK**→**Q**时间, 分 $t_{pLH}(CQ)$ 、 $t_{pHL}(CQ)$ 两种时间
- 建立时间 $t_{setup}$ : 输入信号D在时钟边沿到达前需稳定的时间
- 保持时间 $t_{hold}$ : 输入信号D在时钟边沿到达后需继续稳定的时间

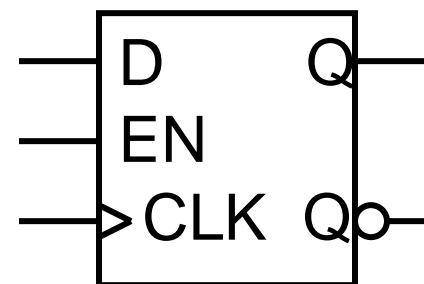
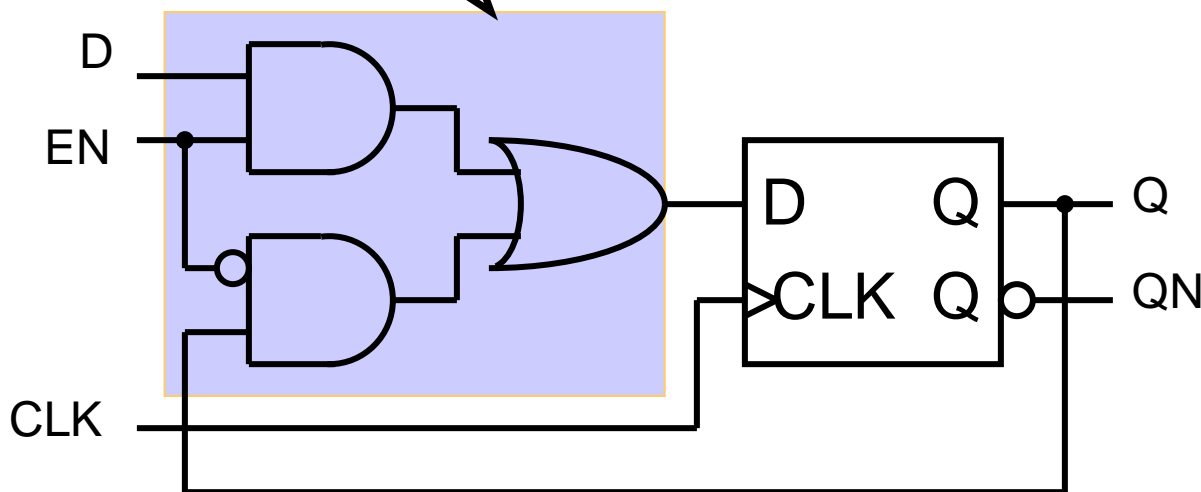
## 2.4 D触发器

- ◆ 帶使能端的D触发器：通过使能端EN信号来控制是否在时钟信号的触发边沿进行数据的存储。

2选1  
多路复用器

EN有效 (=1) 选择外部D输入

EN无效 (=0) 保持触发器当前的输出



逻辑符号

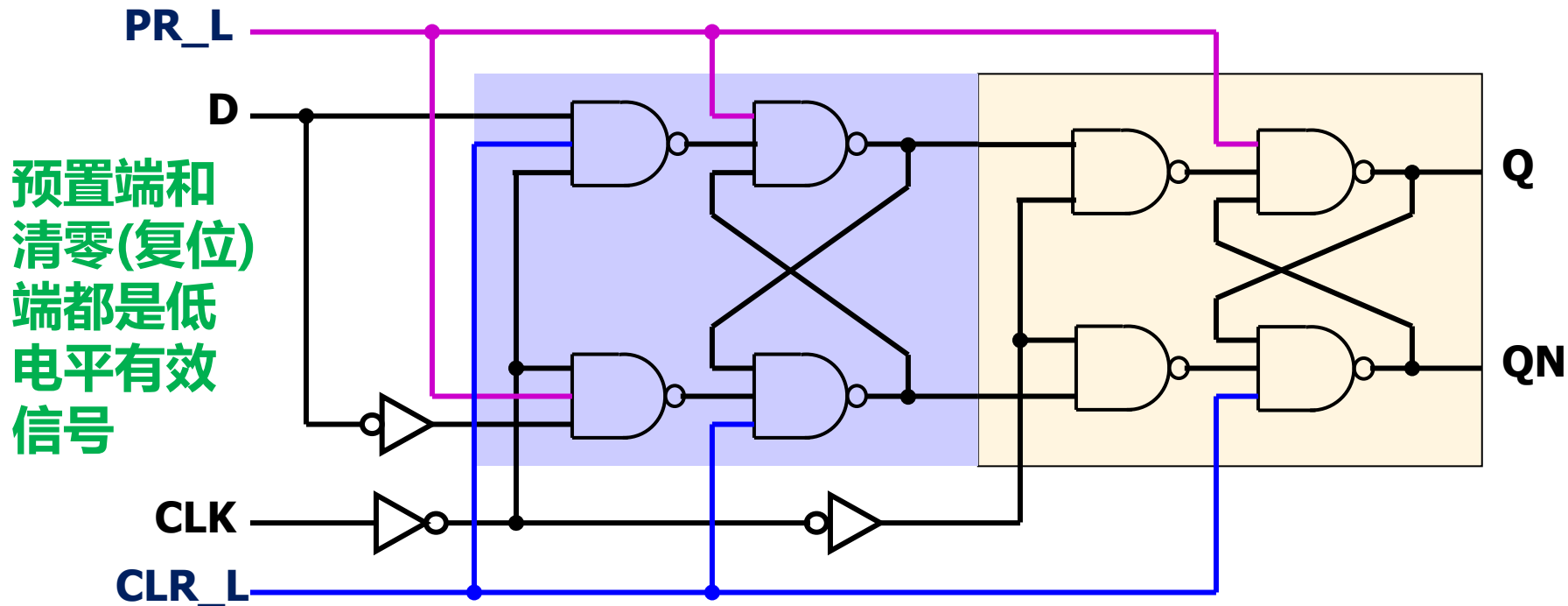
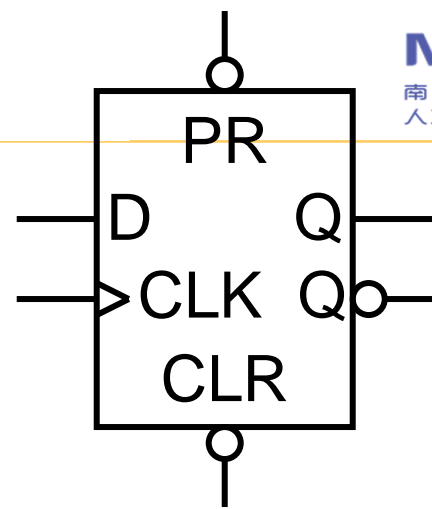


## 2.4 D触发器

### ◆ 具有预置和清零（复位）端的D触发器

- 预置端PR (preset) : 将Q置1
- 清零端CLR (clear) : 将Q清0

在电路工作的最开始进行置位或清0

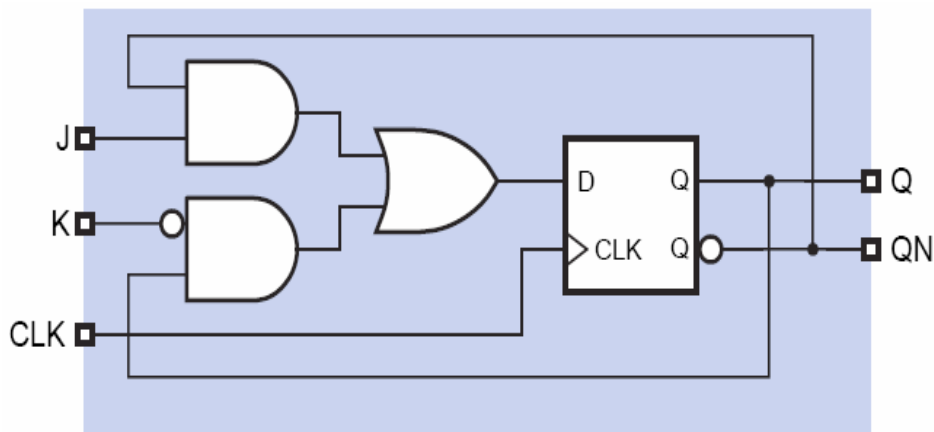


预置端和清零端有同步、异步之分。同步方式下只能在CLK的触发边沿进行预置和清零，异步方式下与时钟信号无关

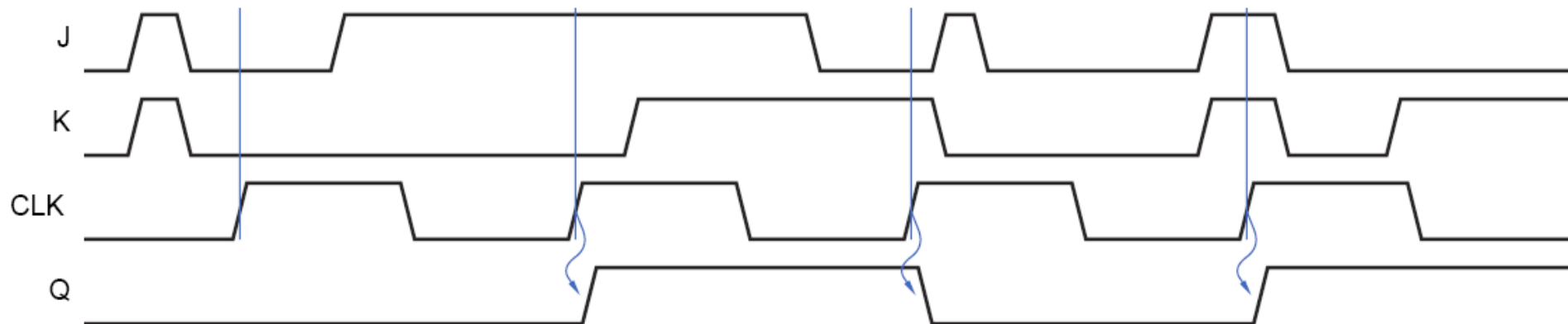
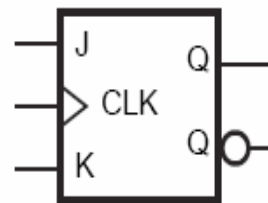
# 补：边沿触发式J-K触发器

◆在上升沿时采样输入信号。

特征方程：  $Q^* = JQ' + K'Q$



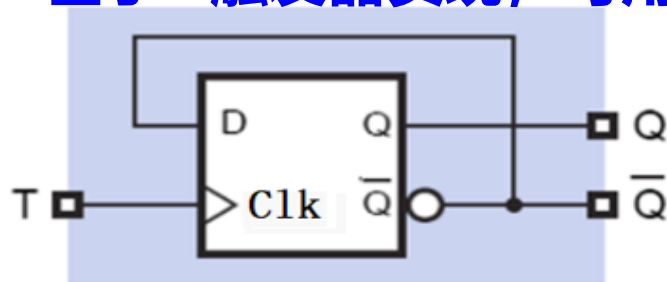
J	K	CLK	Q	QN
x	x	0	last Q	last QN
x	x	1	last Q	last QN
0	0		last Q	last QN
0	1		0	1
1	0		1	0
1	1		last QN	last Q



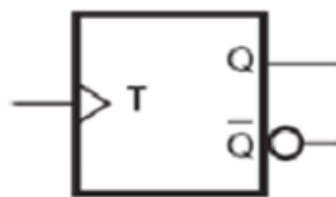
## 2.5 T触发器

◆ T触发器：在每个时钟脉冲的触发边沿都会改变状态

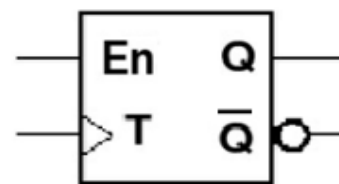
基于D触发器实现；可用于实现计数器、分频器等功能



a) T触发器原理图



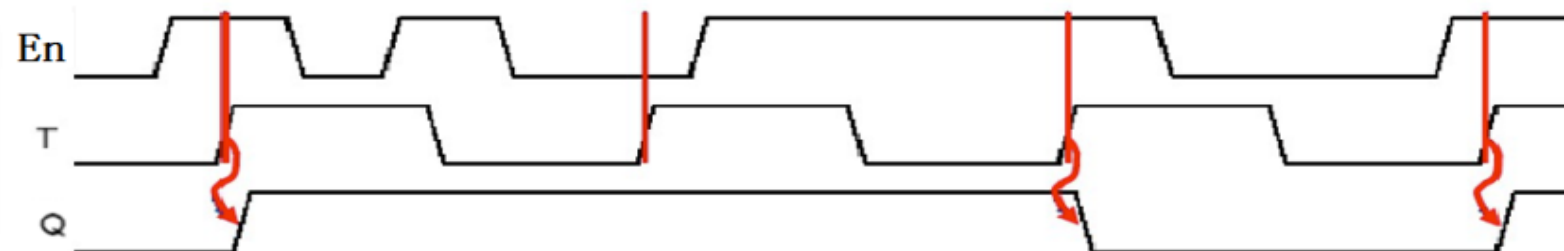
b) T触发器  
电路符号



c) 带使能端T触  
发器电路符号



d) T触发器波形图



e) 带使能端T触发器波形图

# 第三讲 同步时序逻辑设计

- ◆同步时序逻辑设计步骤
- ◆状态图/状态表设计
- ◆状态化简和状态编码
- ◆电路设计和分析

# 3.1 同步时序逻辑设计步骤

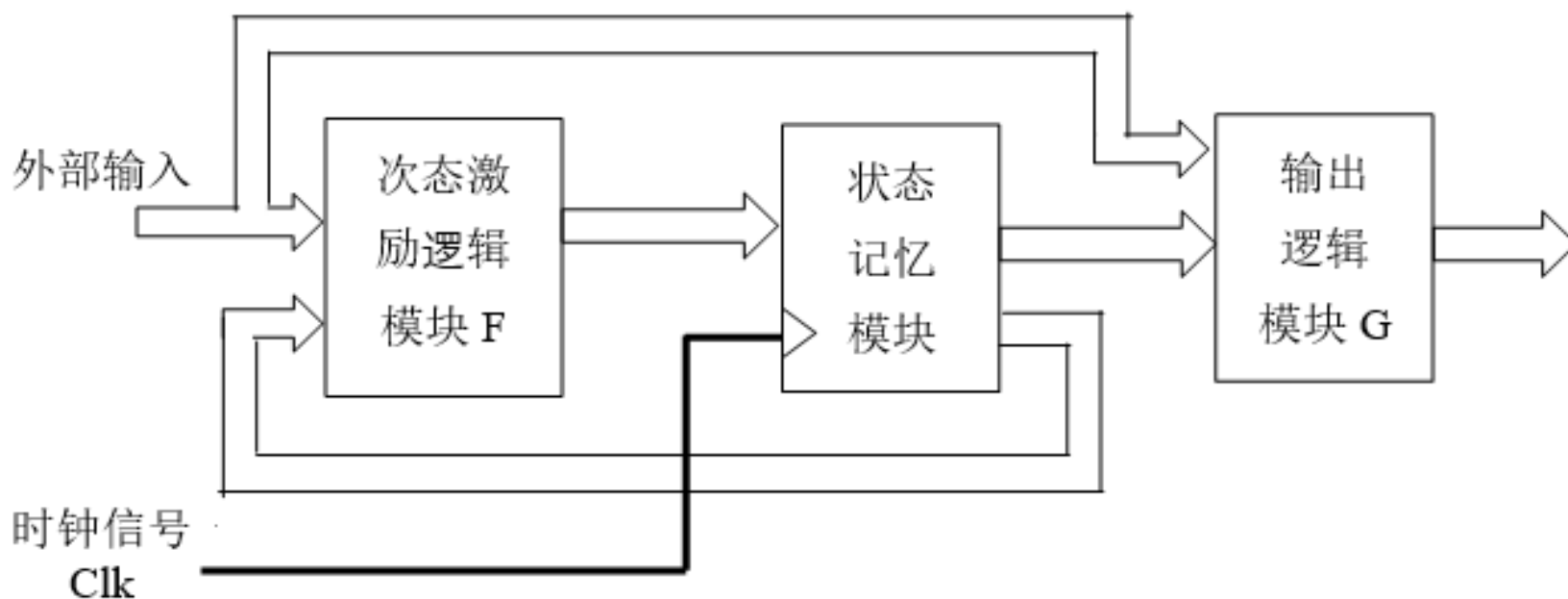
◆同步时序逻辑的设计，就是要把F和G函数设计出来。

- F函数：外部输入+内部状态 → 次态的激励信号

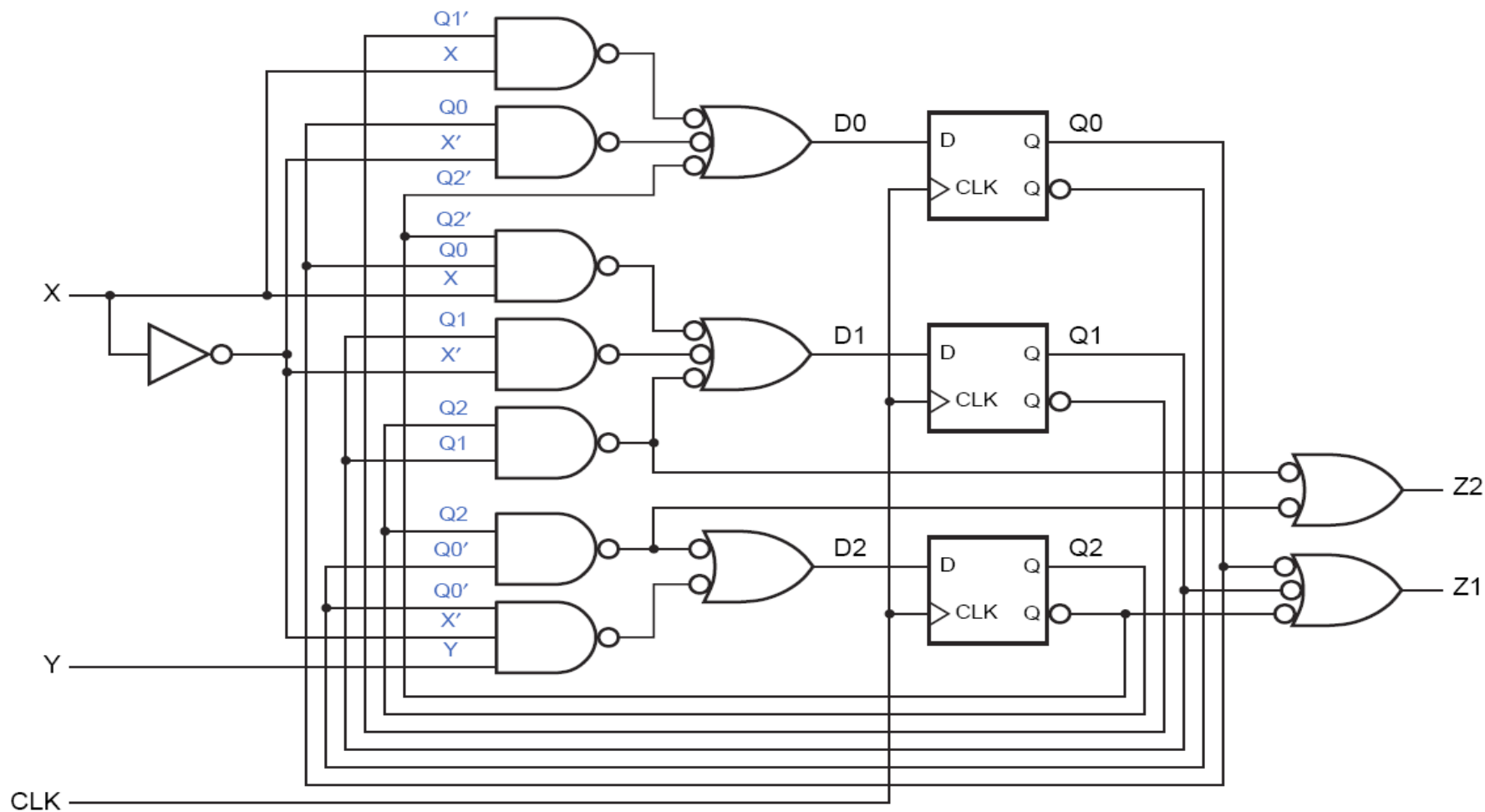
- 外部输入+内部状态 → 次态

- 次态 → 次态的激励信号（记忆器件的次态方程）

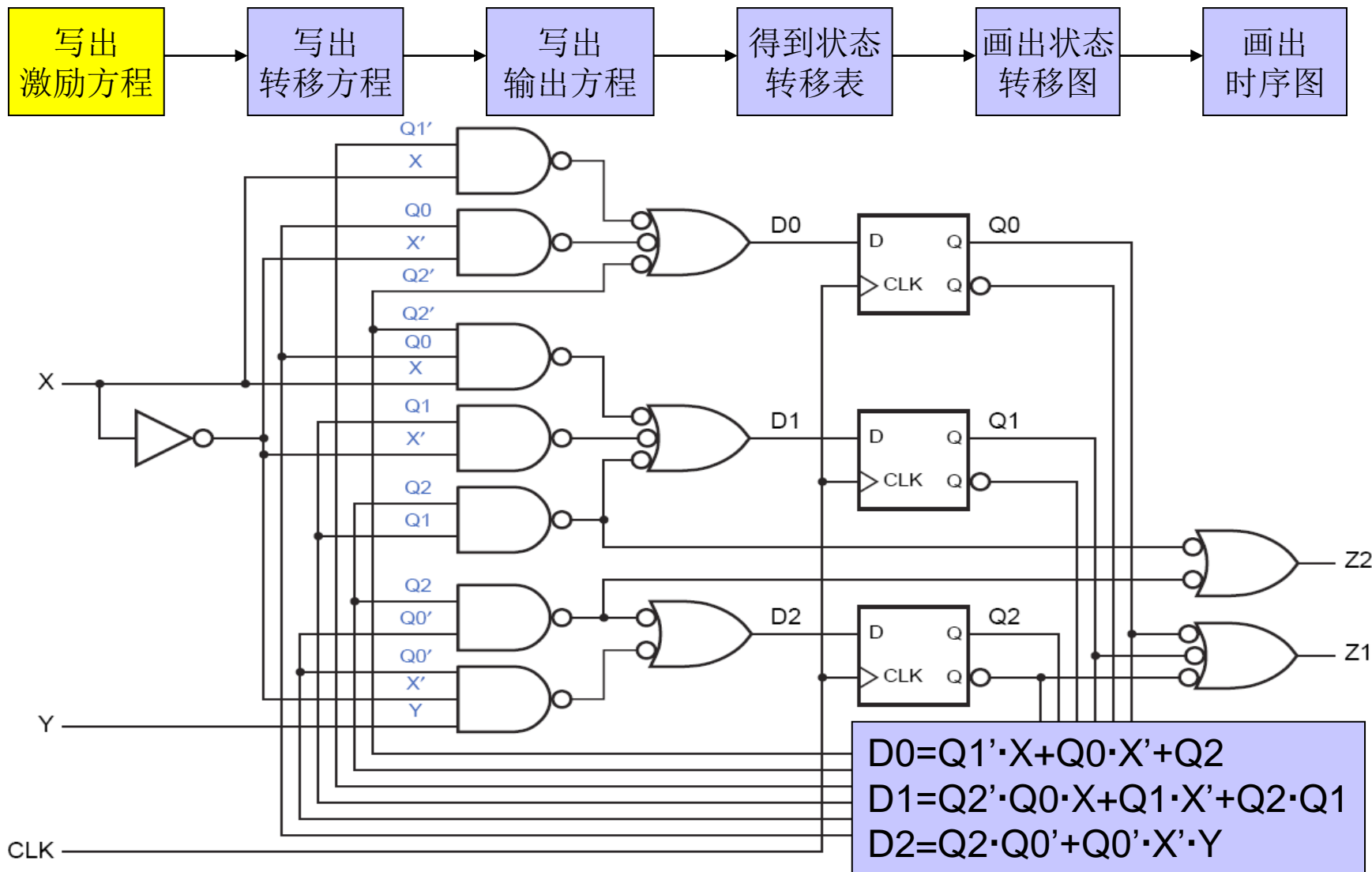
- G函数：外部输入+内部状态 → 输出



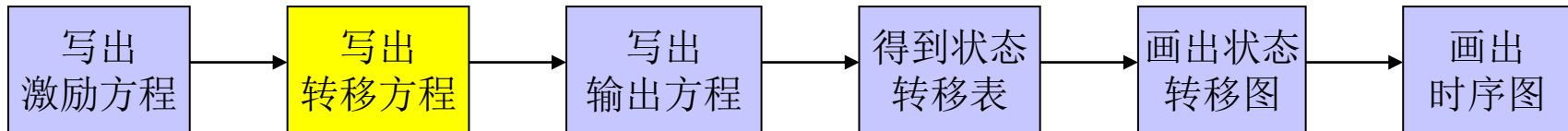
# 补：同步时序电路分析例1



# 第一步：写出激励方程



## 第二步：得到转移方程



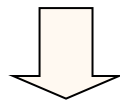
激励方程

$$D0 = Q1' \cdot X + Q0 \cdot X' + Q2$$

$$D1 = Q2' \cdot Q0 \cdot X + Q1 \cdot X' + Q2 \cdot Q1$$

$$D2 = Q2 \cdot Q0' + Q0' \cdot X' \cdot Y$$

D触发器的特征方程  $Q^{n+1} = D$



转移方程

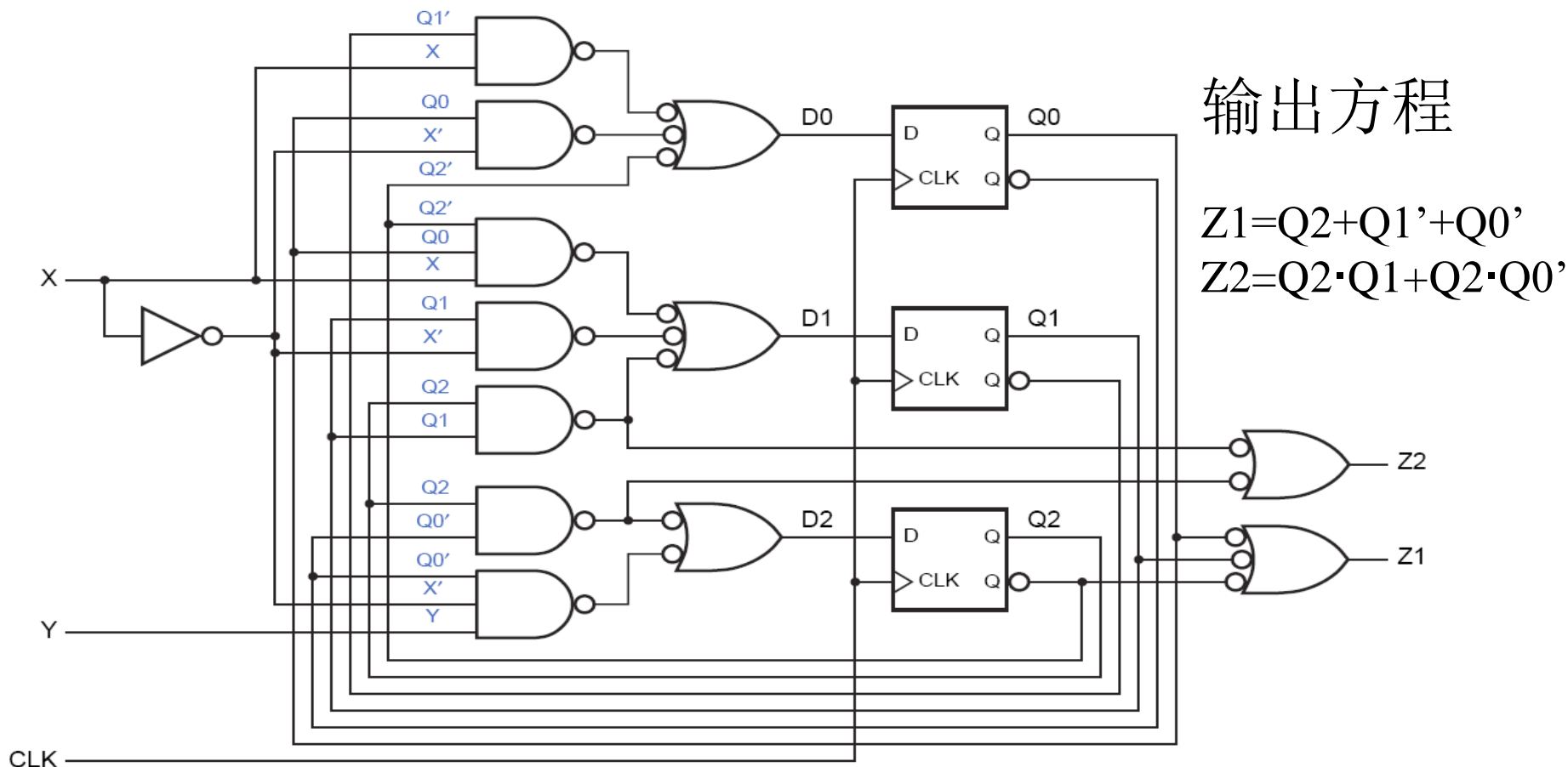
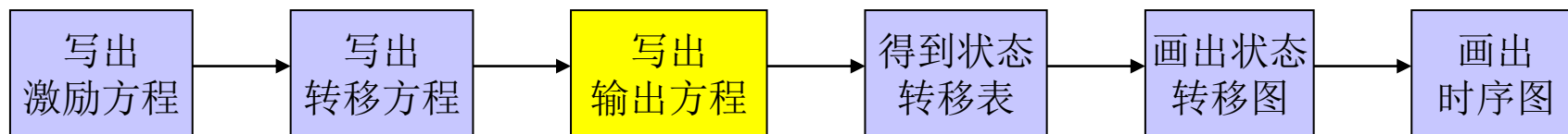
$$Q0^{n+1} = Q1' \cdot X + Q0 \cdot X' + Q2$$

$$Q1^{n+1} = Q2' \cdot Q0 \cdot X + Q1 \cdot X' + Q2 \cdot Q1$$

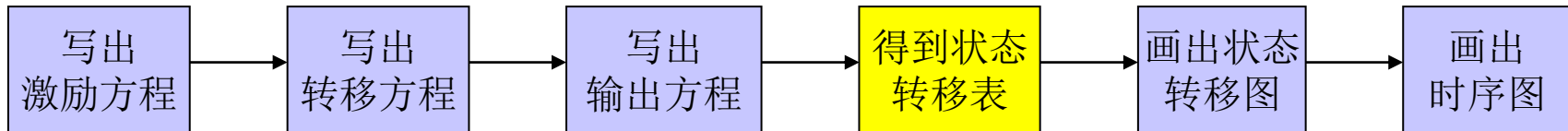
$$Q2^{n+1} = Q2 \cdot Q0' + Q0' \cdot X' \cdot Y$$



# 第三步：得到输出方程



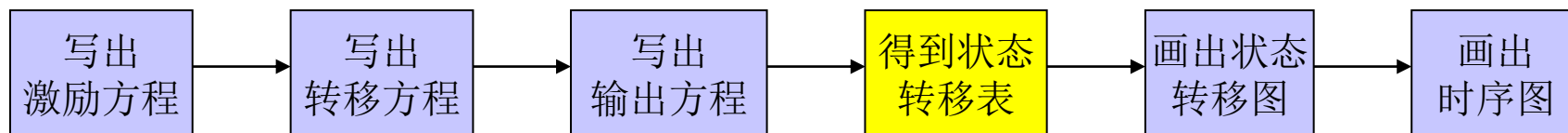
## 第四步：构建转移表



$$\begin{aligned}
 Q0^{n+1} &= Q1' \cdot X + Q0 \cdot X' + Q2 \\
 Q1^{n+1} &= Q2' \cdot Q0 \cdot X + Q1 \cdot X' + Q2 \cdot Q1 \\
 Q2^{n+1} &= Q2 \cdot Q0' + Q0' \cdot X' \cdot Y \\
 Z1 &= Q2 + Q1' + Q0' \\
 Z2 &= Q2 \cdot Q1 + Q2 \cdot Q0'
 \end{aligned}$$

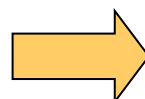
			XY					
			00	01	10	11	Z1	Z2
Q2	Q1	Q0						
0	0	0						
0	0	1						
0	1	0						
0	1	1						
1	0	0						
1	0	1						
1	1	0						
1	1	1						
			$Q2^{n+1} Q1^{n+1} Q0^{n+1}$					

## 第四步：构建状态表



XY						
Q2 Q1 Q0	00	01	10	11	Z1 Z2	
000	000	100	001	001	10	
001	001	001	011	011	10	
010	010	110	000	000	10	
011	011	011	010	010	00	
100	101	101	101	101	11	
101	001	001	001	001	10	
110	111	111	111	111	11	
111	011	011	011	011	11	
$Q2^{n+1} Q1^{n+1} Q0^{n+1}$						

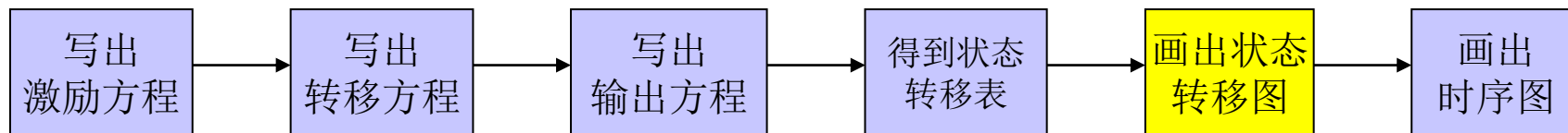
转移表



XY					
S	00	01	10	11	Z1 Z2
A	A	E	B	B	10
B	B	B	D	D	10
C	C	G	A	A	10
D	D	D	C	C	00
E	F	F	F	F	11
F	B	B	B	B	10
G	H	H	H	H	11
H	D	D	D	D	11
$S^{n+1}$					

状态表

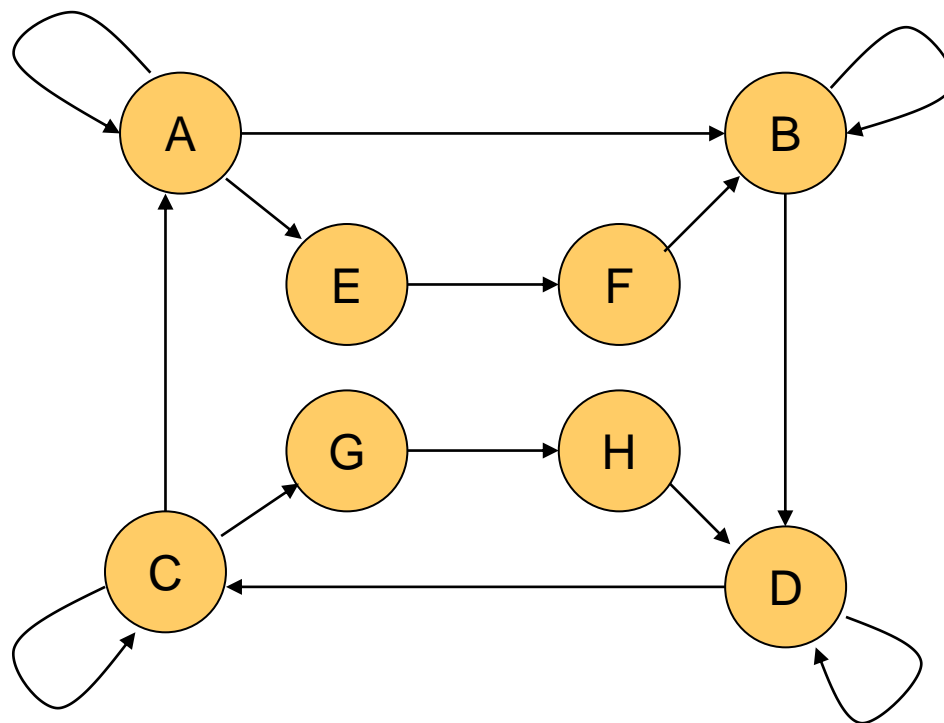
# 第五步：画出状态转移图



S	XY				Z1 Z2
	00	01	10	11	
A	A	E	B	B	10
B	B	B	D	D	10
C	C	G	A	A	10
D	D	D	C	C	00
E	F	F	F	F	11
F	B	B	B	B	10
G	H	H	H	H	11
H	D	D	D	D	11

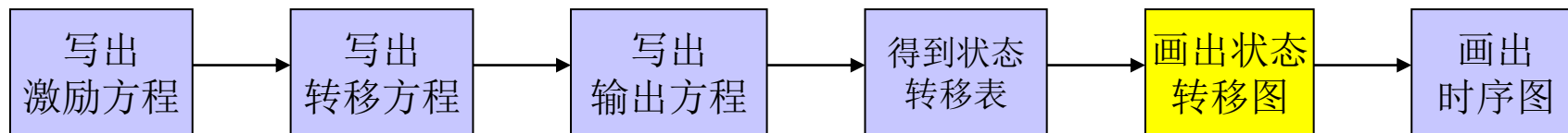
$S^{n+1}$

状态表



转移图

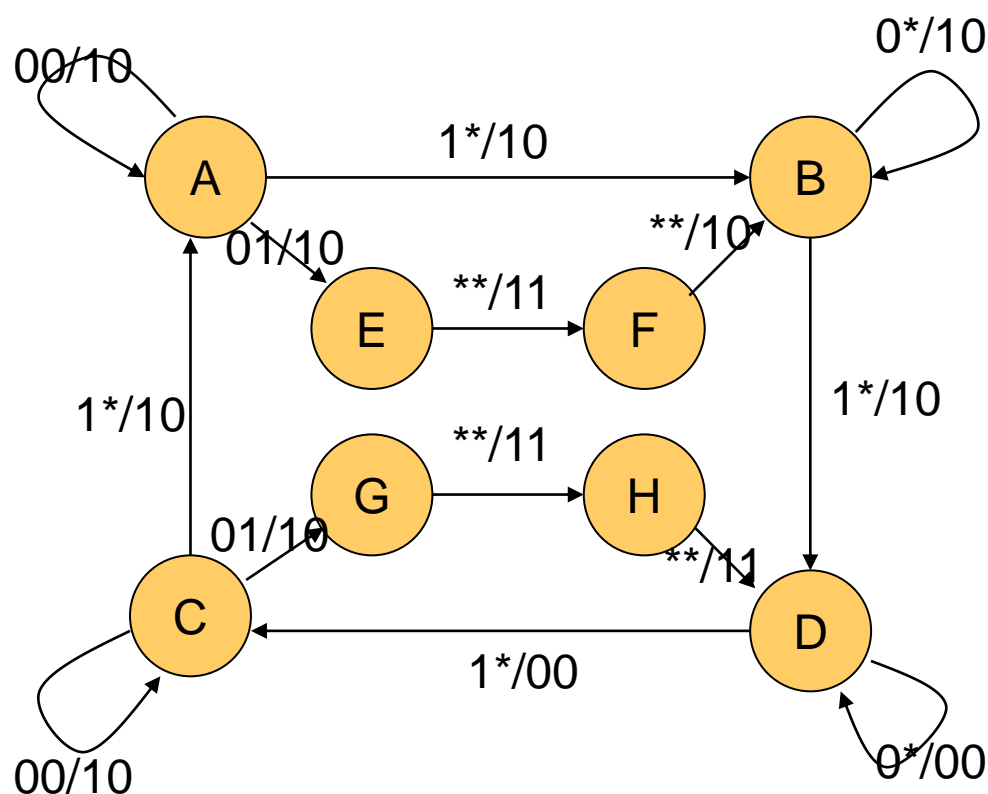
# 第五步：画出状态转移图



S	XY				Z1 Z2
	00	01	10	11	
A	A	E	B	B	10
B	B	B	D	D	10
C	C	G	A	A	10
D	D	D	C	C	00
E	F	F	F	F	11
F	B	B	B	B	10
G	H	H	H	H	11
H	D	D	D	D	11

$S_{n+1}$

状态表

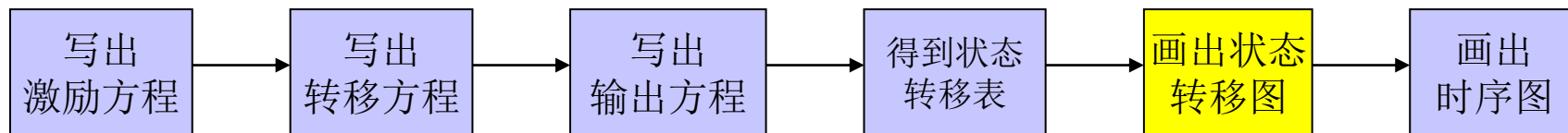


转移图

\*表示与该输入无关

转移表达式必须是互斥的，并且是完备的。

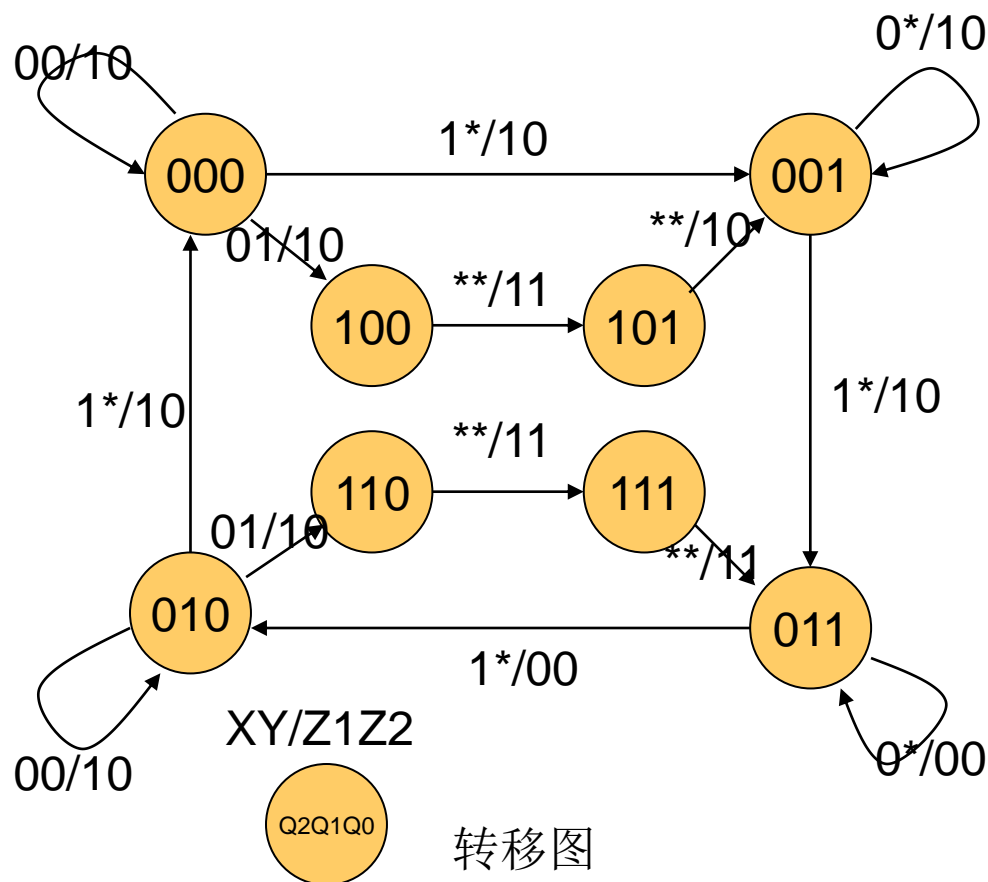
# 第五步：画出状态转移图



Q2 Q1 Q0	XY				Z1 Z2
	00	01	10	11	
000	000	100	001	001	10
001	001	001	011	011	10
010	010	110	000	000	10
011	011	011	010	010	00
100	101	101	101	101	11
101	001	001	001	001	10
110	111	111	111	111	11
111	011	011	011	011	11

$Q2^{n+1} Q1^{n+1} Q0^{n+1}$

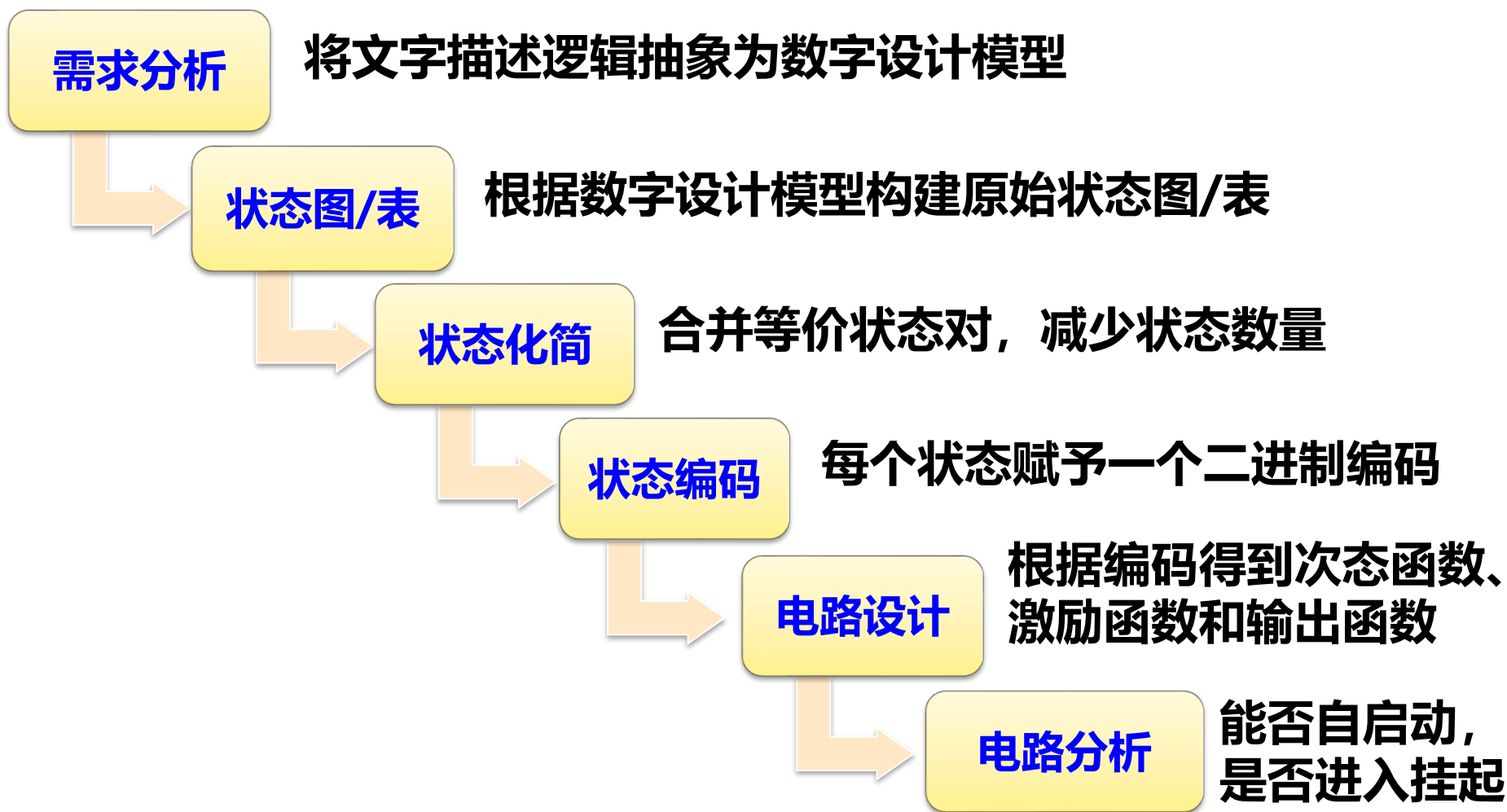
状态表



转移图

\*表示与该输入无关

# 3.1 同步时序逻辑设计步骤



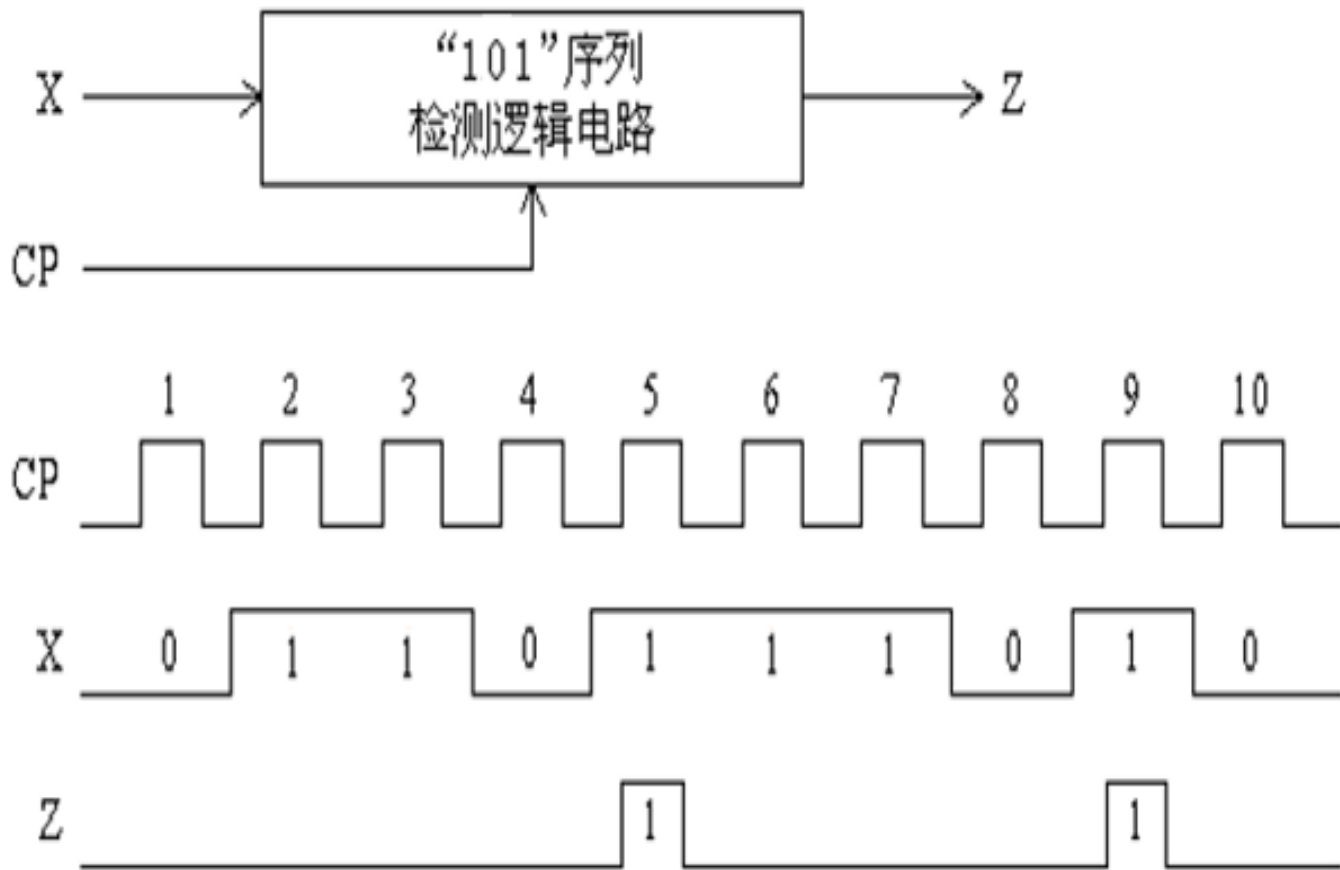
## 3.2 状态图/状态表设计

◆ 状态图/状态表设计：分析系统内部的状态转换关系

例：设计一个能检测出一连串外部输入中是否出现了0/1序列“101”的状态机。

### 1. 需求分析

1位输入端X；  
1位输出端Z。



CP脉冲到来时，  
根据输入端X的  
当前输入值，  
确定输入序列  
中是否出现  
“101”。若  
是，则输出Z为  
1；否则Z为0。



## 3.2 状态图/状态表设计

### 2. 构建状态图/表

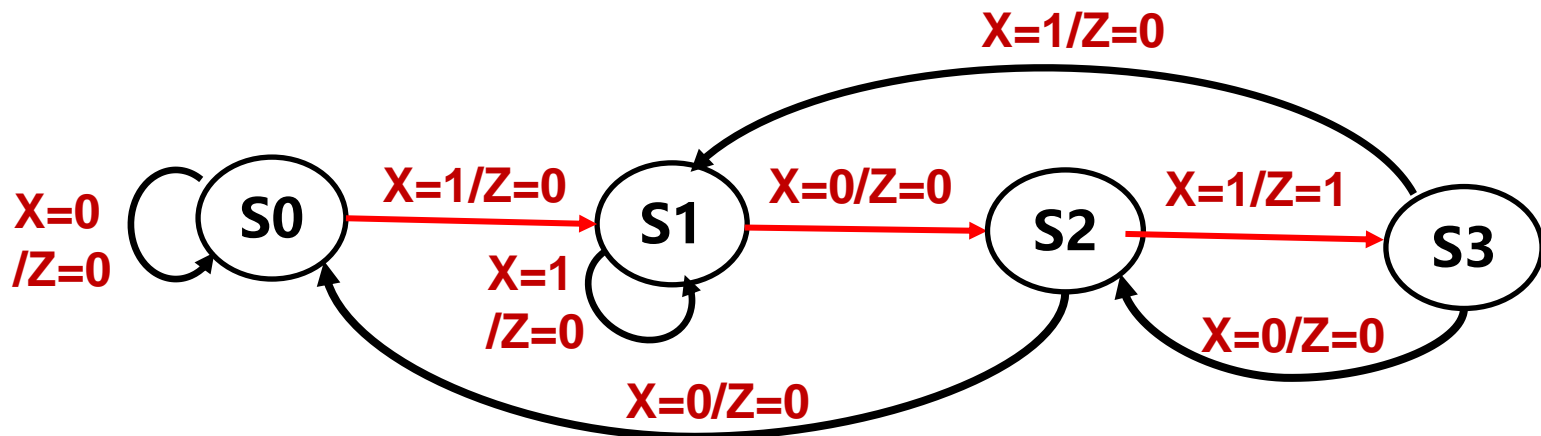
- I. 设定电路初始状态;
- II. 从初始状态开始, 分析每一个状态在不同输入作用下的状态转移情况和输出取值;
- III. 如果某状态下出现的输出响应 (次态、输出) 不能用已有状态表示, 则产生新的状态;
- IV. 重复第II、III两步, 直到不产生新状态为止。

S0: 初始状态, 等待接收输入

S2: 接收到该序列中的10

S1: 接收到“101”序列中第一个1

S3: 接收到一个“101”序列



## 3.2 状态图/状态表设计

状态表

现态S	S*/Z	
	X=0	X=1
S0	S0/0	S1/0
S1	S2/0	S1/0
S2	S0/0	S3/1
S3	S2/0	S1/0

### 2. 构建状态图/表

- 根据状态图构建状态表

X: 输入数据; Z: 检测结果

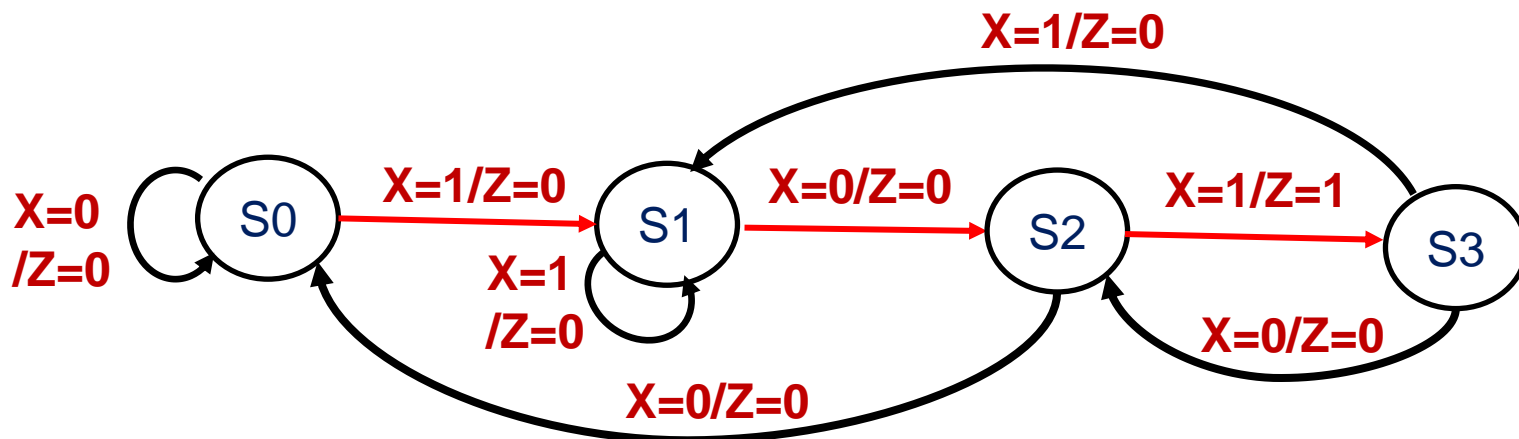
S: 当前状态; S\*: 次态

S0: 初始状态, 等待接收输入

S2: 接收到该序列中的10

S1: 接收到“101”序列中第一个1

S3: 接收到一个“101”序列



### 2. 构建状态图/表

- 构建状态图/表时，状态转移需满足下列两个条件：

**互斥性**：从每个状态出发的所有状态转换路径上的转换条件都是互斥的，也即任意**两个**转移表达式的**逻辑与等于0**。

**完备性**：从每个状态出发的**所有**状态转换路径上的转移表达式的**逻辑或等于1**（逻辑真）。

本例中，转移条件分别是 $X=0$ 和 $X=1$ ，满足互斥性和完备性。

- 在状态图中，也可以使用逻辑表达式来表示转移条件。本例中，可以使用 $X$ 和 $\bar{X}$ 分别表示输入 $X=1$ 和 $X=0$ 。

## 3.3 状态化简和状态编码

### ◆ 状态化简

- 合并等价状态，以得到更加精简的状态表
- 两个状态等价指在所有输入组合下，它们的输出相同且次态相同或次态等价
- 等价关系具有传递性
  - 例如，若状态A和B等价，同时B和C等价，则A和C也等价。状态A、B和C属于一个等价类，可以合并为一个状态。

状态表

现态S	S*/Z	
	X=0	X=1
S0	S0/0	S1/0
S1	S2/0	S1/0
S2	S0/0	S3/1
S3	S2/0	S1/0

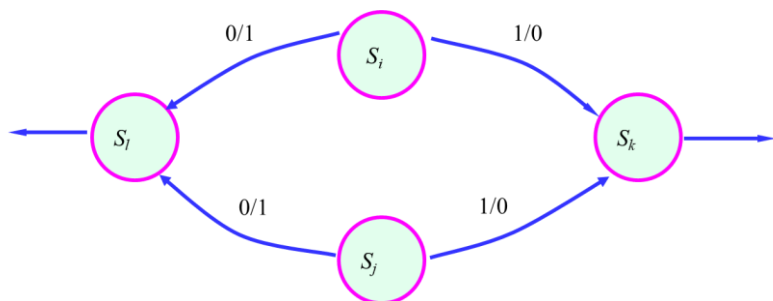
S1和S3构成等价类，可合并化简后，有3个状态

现态S	S*/Z	
	X=0	X=1
S0	S0/0	S1/0
S1	S2/0	S1/0
S2	S0/0	S1/1

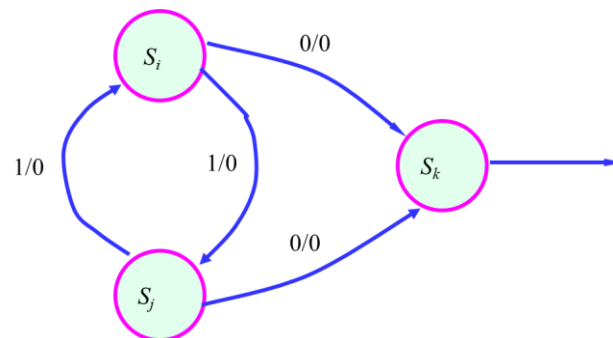
# 3.3 状态化简和状态编码

## ◆ 状态化简

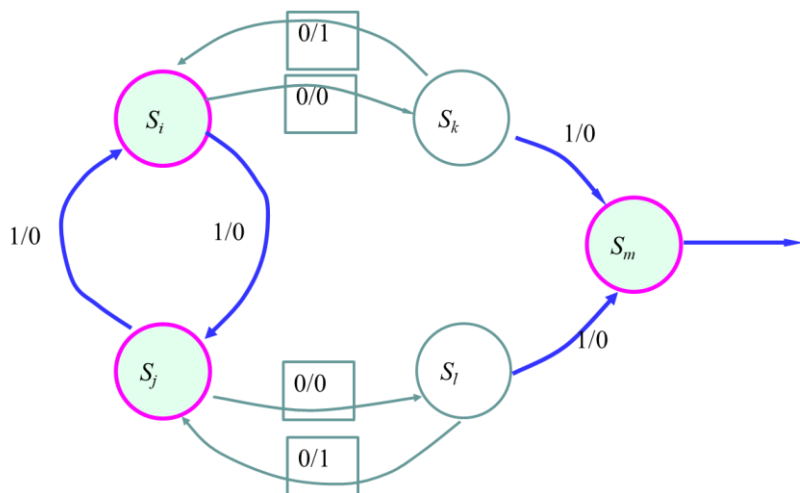
### • 等价关系判定\*



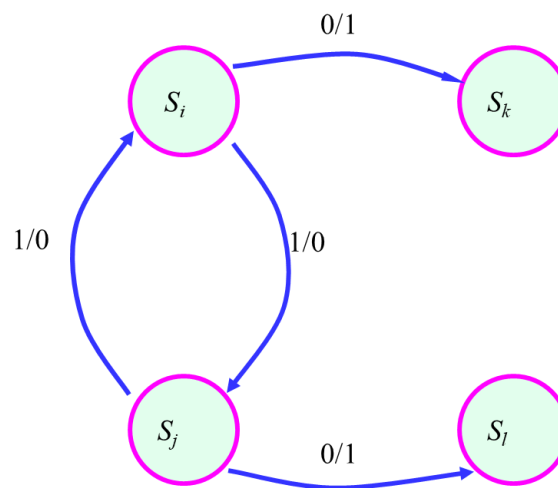
(1) 次态相同;



(2) 次态交错;



(3) 次态循环;



(4) 次态对等价。

## 3.3 状态化简和状态编码

### ◆ 状态编码

- 对状态表中每个状态赋予**唯一**的二进制编码，也称状态赋值
  - 不同状态编码方案，电路设计不一样，存在优劣之分。
- 寻找最优编码方案是一个非常复杂的问题
- 通常在具体设计时采用**相邻法**寻求**次优编码方案**：
  - 准则1：若两个状态的次态相同，则其对应编码应尽量**相邻**
  - 准则2：同一个现态的各个次态其编码应尽量相邻
  - 准则3：若两个现态的输出相同，则它们的编码应尽量相邻

现态S	S*/Z	
	X=0	X=1
S0	S0/0	S1/0
S1	S2/0	S1/0
S2	S0/0	S1/1

根据准则1：S0和S2可相邻

根据准则2：S0和S1、S1和S2可相邻

根据准则3：S0和S1可相邻

根据不同的取舍可得到不同编码方案

若S0和S1、S1和S2相邻，则编码方案如下：  
S0:00, S1:01, S2:11

若S0和S2、S0和S1相邻，则编码方案如下：  
S0:00, S1:01, S2:10

# 3.4 电路设计与分析

## ◆ 在选定的状态编码方案基础上进行电路设计

### • 生成状态转移表

对于前面的例子，若编码方案为  
S0:00, S1:01, S2:11，则得到  
右边的状态转移表

Y1Y0	Y1*Y0*/Z	
	X=0	X=1
00	00/0	01/0
01	11/0	01/0
11	00/0	01/1

						Q*=D		Q*=JQ'+K'Q			
X	Y1	Y0	Y1*	Y0*	Z	D1	D0	J1	K1	J0	K0
0	0	0	0	0	0	0	0	0	d	0	d
0	0	1	1	1	0	1	1	1	d	d	0
0	1	0	d	d	d	d	d	d	d	d	d
0	1	1	0	0	0	0	0	d	1	d	1
1	0	0	0	1	0	0	1	0	d	1	d
1	0	1	0	1	0	0	1	0	d	d	0
1	1	0	d	d	d	d	d	d	d	d	d
1	1	1	0	1	1	0	1	d	1	d	0

## 3.4 电路设计与分析

### ◆ 在选定的状态编码方案基础上进行电路设计

#### • 生成状态转移表

对于前面的例子，若编码方案为  
S0:00, S1:01, S2:11，则得到  
右边的状态转移表

X	Y1	Y0	Y1*	Y0*	Z
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	d	d	d
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	d	d	d
1	1	1	0	1	1

○ 根据状态转移表，推导次态逻辑函数和输出逻辑函数

○ 次态函数/次态方程为：

$$Y1^* = \overline{Y1} \cdot Y0 \cdot \overline{X}$$

$$\begin{aligned} Y0^* &= \overline{Y1} \cdot Y0 \cdot \overline{X} + X \cdot (\overline{Y1} \cdot \overline{Y0} + \overline{Y1} \cdot Y0 + Y1 \cdot \overline{Y0}) \\ &= \overline{Y1} \cdot Y0 + X \cdot \overline{Y1} + X \cdot Y0 \end{aligned}$$

将无关项编码Y1Y0=10引入化简，则 $Y0^* = \overline{Y1} \cdot Y0 + X$

○ 输出函数/输出方程为：

$$Z = Y1 \cdot Y0 \cdot X + Y1 \cdot \overline{Y0} \cdot X = Y1 \cdot X$$



## 3.4 电路设计与分析

◆ 根据次态函数和选择的状态记忆单元（触发器），推导出激励函数

• 假设采用D触发器，其特征方程 $Q^* = D$ ，则：

$$D1 = Y1^* = \overline{Y1} \cdot Y0 \cdot \overline{X}$$

$$D0 = Y0^* = \overline{Y1} \cdot Y0 + X$$

◆ 输出函数为： $Z = Y1 \cdot X$

X	Y1	Y0	Y1*	Y0*	Z	Q*=D		Q*=JQ'+K'Q			
						D1	D0	J1	K1	J0	K0
0	0	0	0	0	0	0	0	0	d	0	d
0	0	1	1	1	0	1	1	1	d	d	0
0	1	0	d	d	d	d	d	d	d	d	d
0	1	1	0	0	0	0	0	d	1	d	1
1	0	0	0	1	0	0	1	0	d	1	d
1	0	1	0	1	0	0	1	0	d	d	0
1	1	0	d	d	d	d	d	d	d	d	d
1	1	1	0	1	1	0	1	d	1	d	0

## 3.4 电路设计与分析

◆ 根据次态函数和选择的状态记忆单元（触发器），推导出激励函数

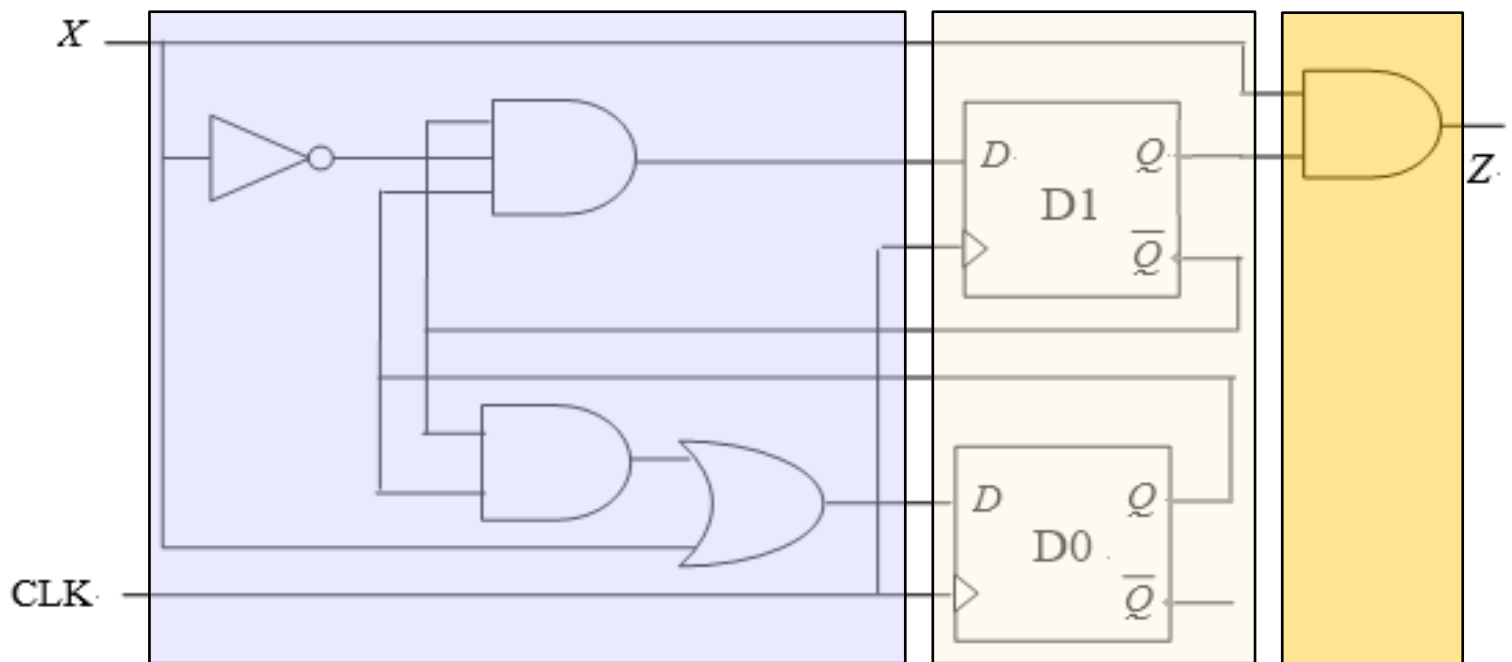
• 假设采用D触发器，其特征方程 $Q^* = D$ ，则：

$$D1 = Y1^* = \overline{Y1} \cdot Y0 \cdot \overline{X}$$

$$D0 = Y0^* = \overline{Y1} \cdot Y0 + X$$

◆ 输出函数为： $Z = Y1 \cdot X$

◆ 根据激励函数和输出函数，画出逻辑电路图



## 3.4 电路设计与分析

### ◆ 电路分析：包括未用状态分析和电路定时分析等

- 通常编码空间比状态机的状态集合大，因而存在未用状态

如前述例子中，编码(2位)空间为4，而实际状态数为3

- 若电路加电后进入未用状态，且在未用状态之间形成循环转换而无法进入工作状态，则称其为“挂起”现象
- 若时序逻辑电路中的触发器具有预置功能，则可以通过预置处理，使电路进入正常的初始工作状态，从而避免“挂起”
- 可利用未用状态的无关项进行化简。但需对未用状态进行分析，以判定电路进入未用状态时能否在有限个时钟周期后进入到工作状态。若能，且没有错误输出，则称电路为具有“自启动”能力；若不能，则需调整电路设计

## 3.4 电路设计与分析

### ◆ 未用状态分析举例

- 对于前述的例子，利用未用状态10作为无关项化简后，得到：

$$Y1^* = \overline{Y1} \cdot Y0 \cdot \overline{X}$$

$$Y0^* = \overline{Y1} \cdot Y0 + X$$

$$Z = Y1 \cdot X$$

- 当处于未用状态10时，根据上述逻辑表达式，可知：

若输入 $X=0$ ，则次态=00，输出 $Z=0$

若输入 $X=1$ ，则次态=01，输出 $Z=1$

- 分析是否具有“自启动”能力

经过1个时钟周期就能进入正常工作状态，但是，当输入 $X=1$ 时，输出 $Z=1$ ，是错误输出，需要调整输出模块的设计

- 重新设计逻辑电路中的输出模块

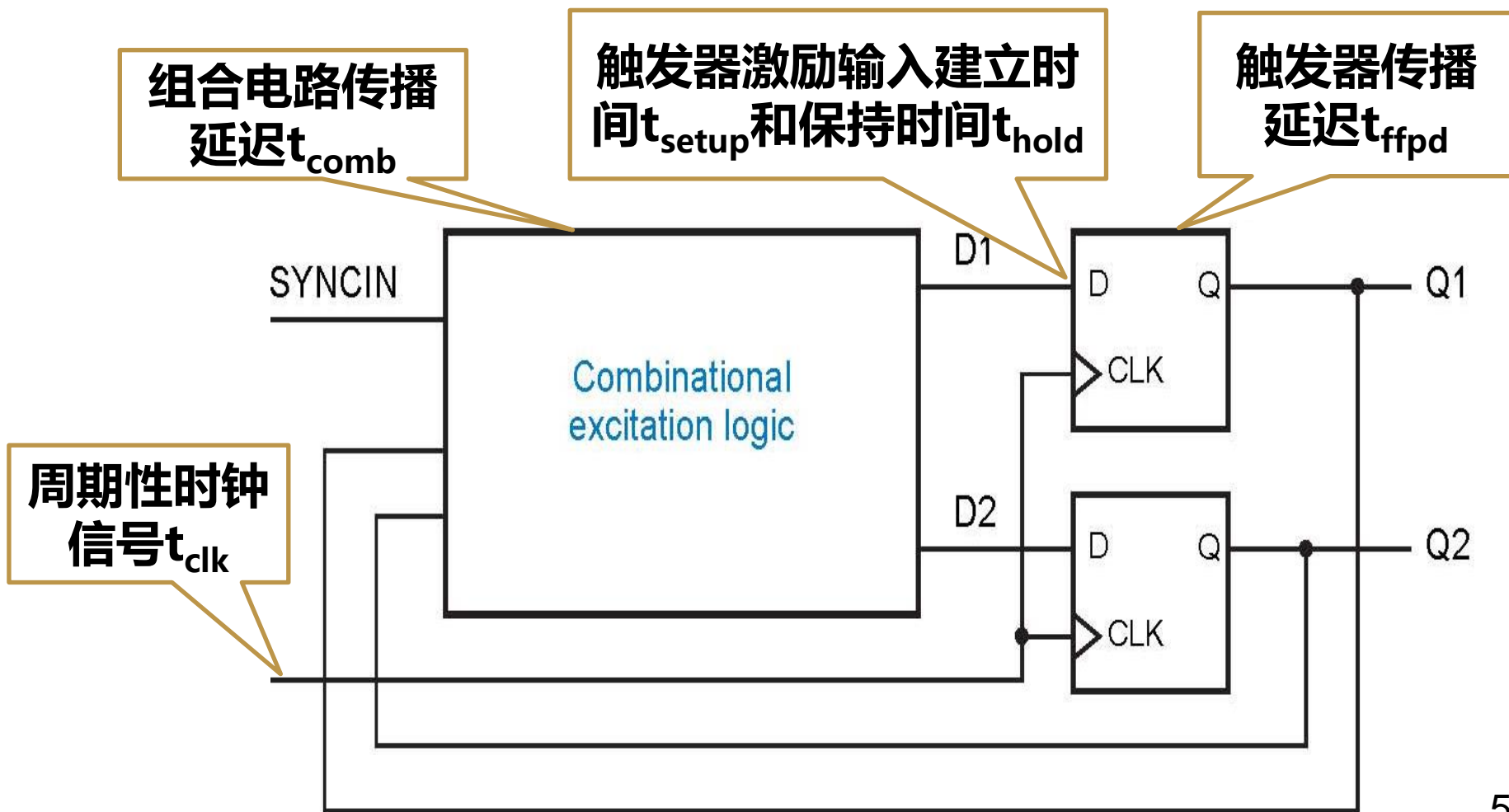
$Z = Y1 \cdot Y0 \cdot X$  在未用状态10时，若输入 $X=1$ 时，则输出 $Z=0$ 。

此时，不会发生误输出

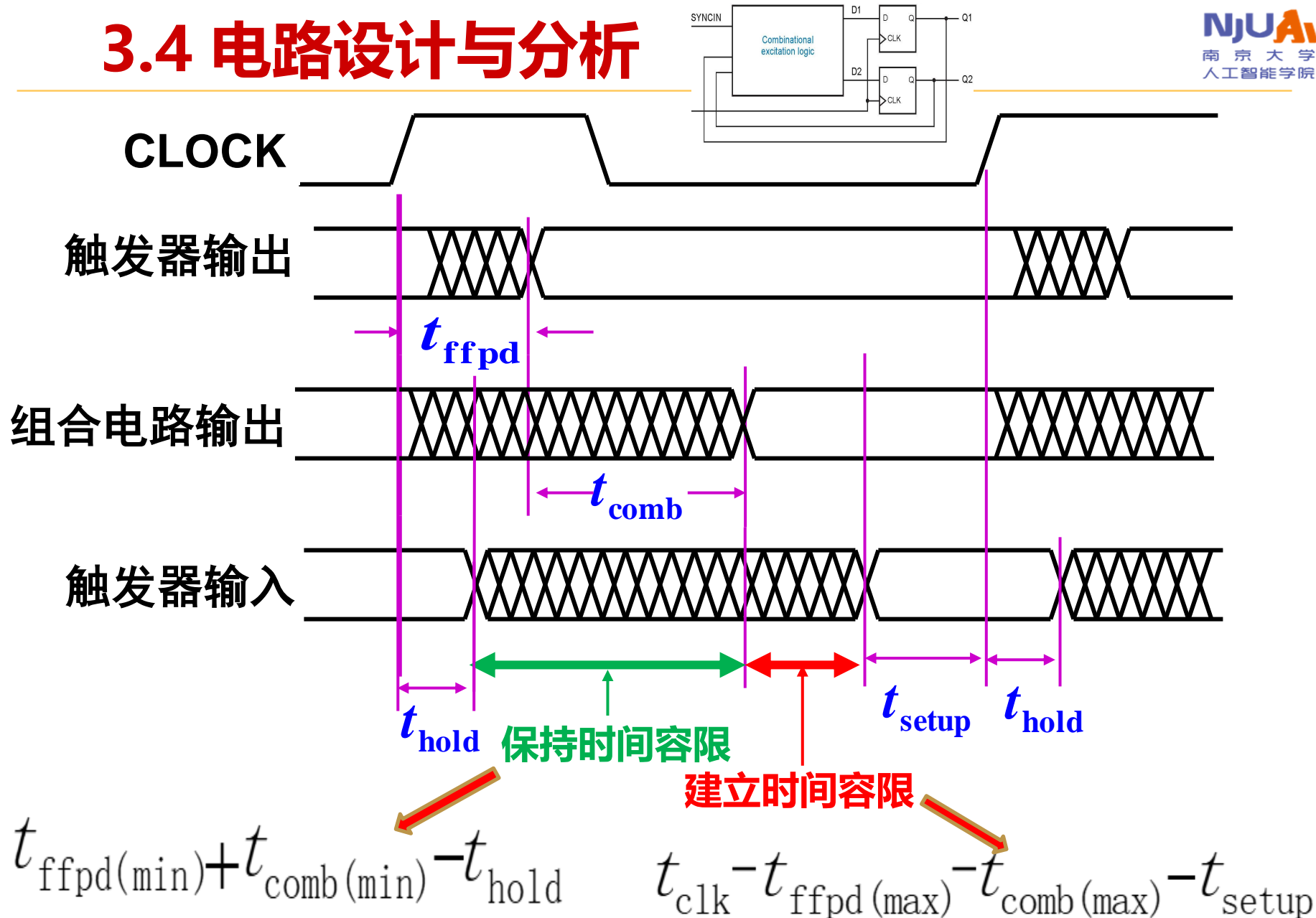
## 3.4 电路设计与分析

### ◆ 电路定时分析

- 电路的工作频率和组合逻辑电路传输延迟、触发器建立和保持时间、触发器传输延迟等时间密切相关。



# 3.4 电路设计与分析



**时间容限**指为保证电路正常工作某信号定时所允许的最大时间范围

## 3.4 电路设计与分析

◆ 建立时间容限 =  $t_{\text{clk}} - t_{\text{ffpd}(\text{max})} - t_{\text{comb}(\text{max})} - t_{\text{setup}}$ ,  $> 0$

◆ 保持时间容限 =  $t_{\text{ffpd}(\text{min})} + t_{\text{comb}(\text{min})} - t_{\text{hold}}$ ,  $> 0$

因此, 得到**时序约束关系**:

$$(1) t_{\text{clk}} > t_{\text{ffpd}(\text{max})} + t_{\text{comb}(\text{max})} + t_{\text{setup}}$$

$$(2) t_{\text{hold}} < t_{\text{ffpd}(\text{min})} + t_{\text{comb}(\text{min})}$$

(1) 为使触发器正常工作, 必须保证时钟周期 $t_{\text{clk}}$ 不能小于触发器锁存延迟 $t_{\text{ffpd}}$  + 次态信号经过激励逻辑延迟 $t_{\text{comb}}$  + 触发器的建立时间 $t_{\text{setup}}$ 。

(2) 为使触发器正常工作, 必须保证外部激励信号在时钟有效边沿到来后的保持时间 $t_{\text{hold}}$ 内能保持稳定不变。这就要求次态信号不能反馈太快, 即触发器锁存延迟 $t_{\text{ffpd}}$  + 次态信号经过激励逻辑延迟 $t_{\text{comb}}$ 不能小于触发器的保持时间 $t_{\text{hold}}$ 。

# 第四讲 典型时序逻辑部件设计

- ◆计数器
- ◆寄存器和寄存器堆
- ◆移位寄存器



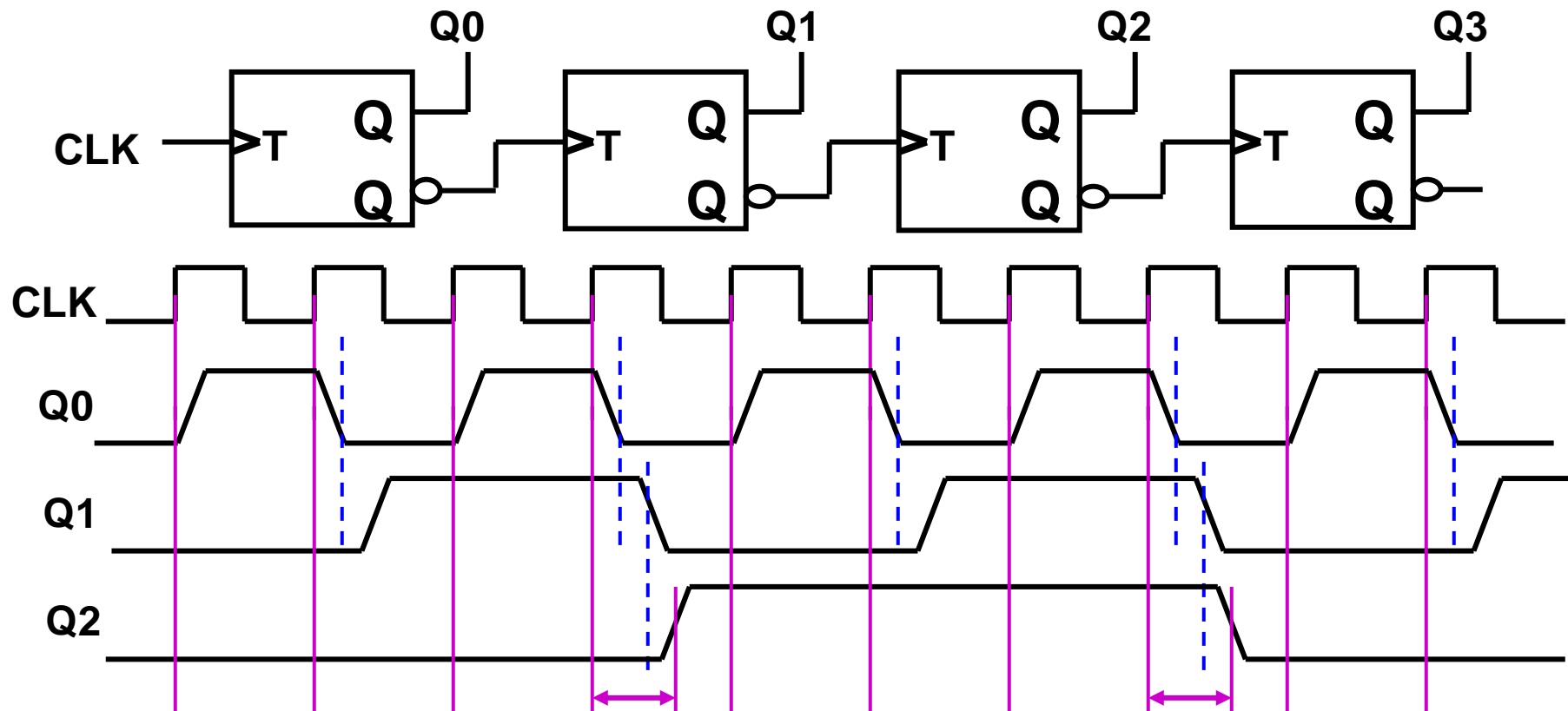
# 4.1 计数器

- ◆ 计数器是一种对外部激励信号进行总数统计的时序逻辑元件
- ◆ 一般从0开始计数，在达到最大计数值时输出一次计数完成信号，并重新开始计数
- ◆ 最大计数值为计数器的模
- ◆ 计数器的分类
  - 按时钟：同步、异步
  - 按计数方式：加法、减法、可逆
  - 按编码方式：二进制、十进制BCD码、循环码
  - 按进位方式：行波（串行）进位、并行进位

# 4.1 計數器

## ◆ 異步行波加法計數器

- 利用 T 觸發器實現，激勵輸入像波浪一樣由低位向高位傳遞，每個時鐘周期傳送一次。



$Q_{i+1}$  总是在  $Q_i$  由 1 变为 0 时开始改变状态  
第  $n$  位状态变换最长要经过  $n \times t_{TQ}$  的延迟时间

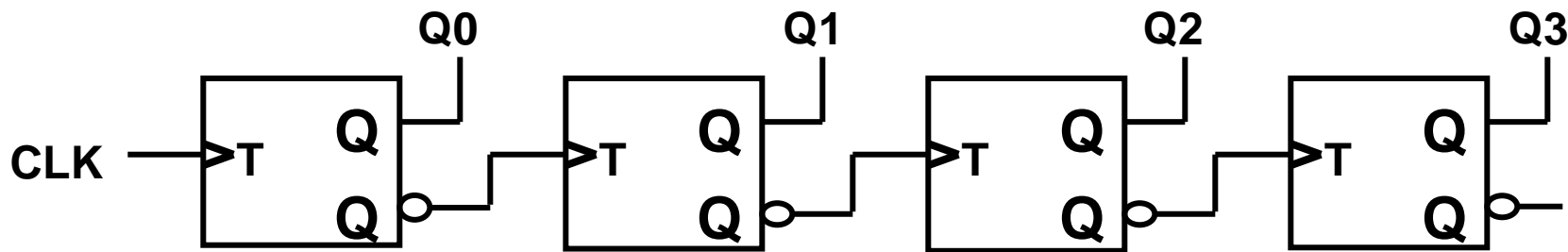
# 4.1 计数器

当编码为1111时，下个时钟到达后，经过 $n \times t_{TQ}$ 延时，又回到编码0000

## ◆ 异步行波加法计数器

- 计数器的状态编码 $Q_3Q_2Q_1Q_0$ 从0000开始，转换过程为  
0000→0001→0010→0011→0100→0101→0110→0111→  
1000→...→1111

$Q_{i+1}$ 总是在 $Q_i$ 由1变0时开始改变状态



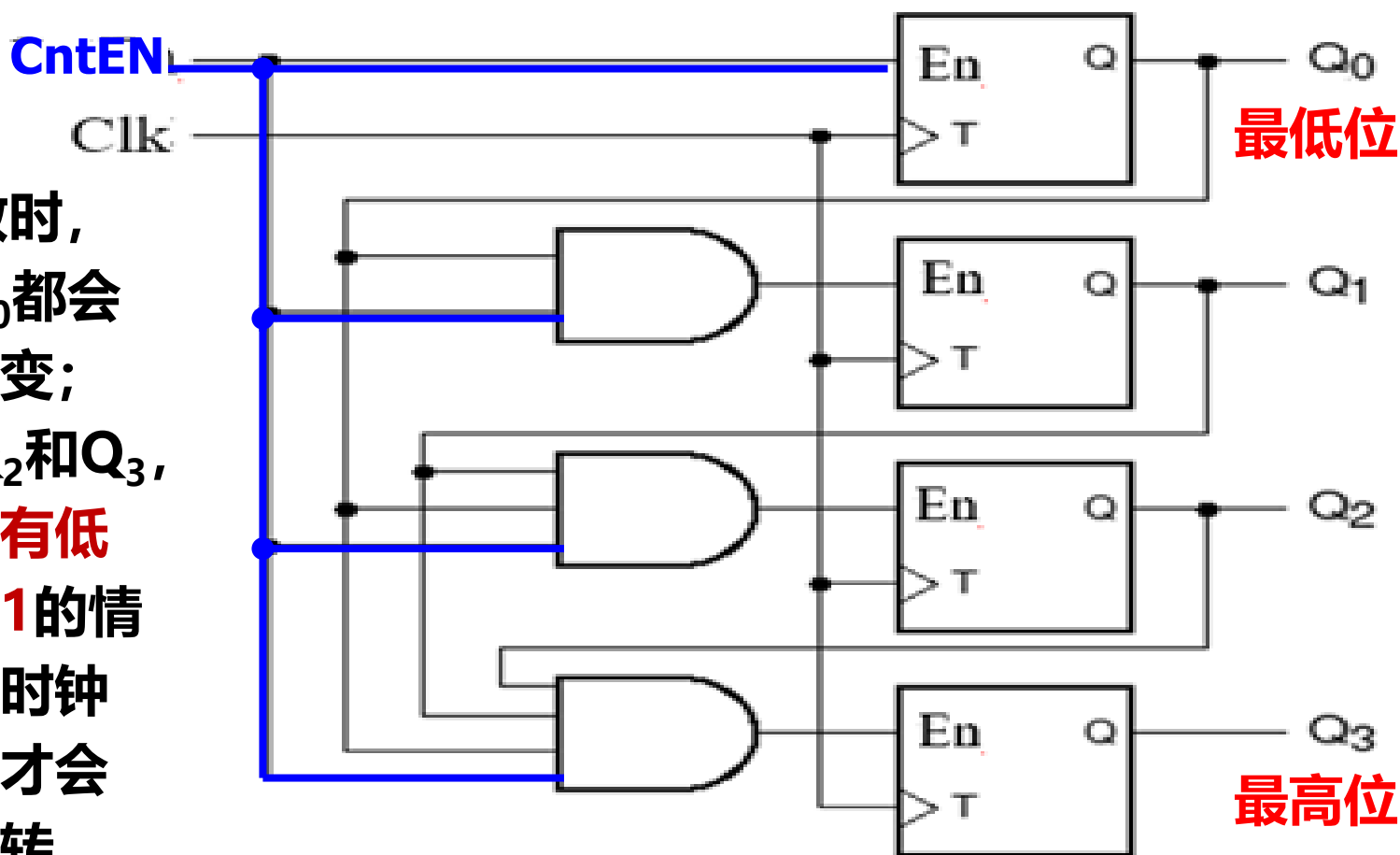
第1个Clk上升沿到来后，最低位状态 $Q_0$ 从0变成1，此时其他三个状态位不变，因而得到状态编码0001；  
第2个Clk到来后， $Q_0$ 从1变成0，此时 $Q_1$ 从0变成1，而其他两个状态位不变，因而得到状态编码0010；  
第3个Clk有效信号到来后， $Q_0$ 从0变成1，此时其他三个状态位不变，因而得到状态编码0011；  
第4个Clk有效信号到来后， $Q_0$ 从1变成0，此时 $Q_1$ 从1变成0， $Q_2$ 从0变成1， $Q_3$ 状态位不变，因而得到状态编码0100；……。

# 4.1 計數器

## ◆ 同步并行加法計數器

- 同步計數器中所有觸發器共用同一個時鐘信號，在時鐘信號邊沿到達后，所有觸發器的輸出同時發生變化。

CntEN有效時，  
每個時鐘 $Q_0$ 都會  
發生狀態改變；  
對於 $Q_1$ 、 $Q_2$ 和 $Q_3$ ，  
只有在其**所有低  
位狀態都是1**的情  
況下，下個時鐘  
邊沿到來后才會  
發生狀態反轉。



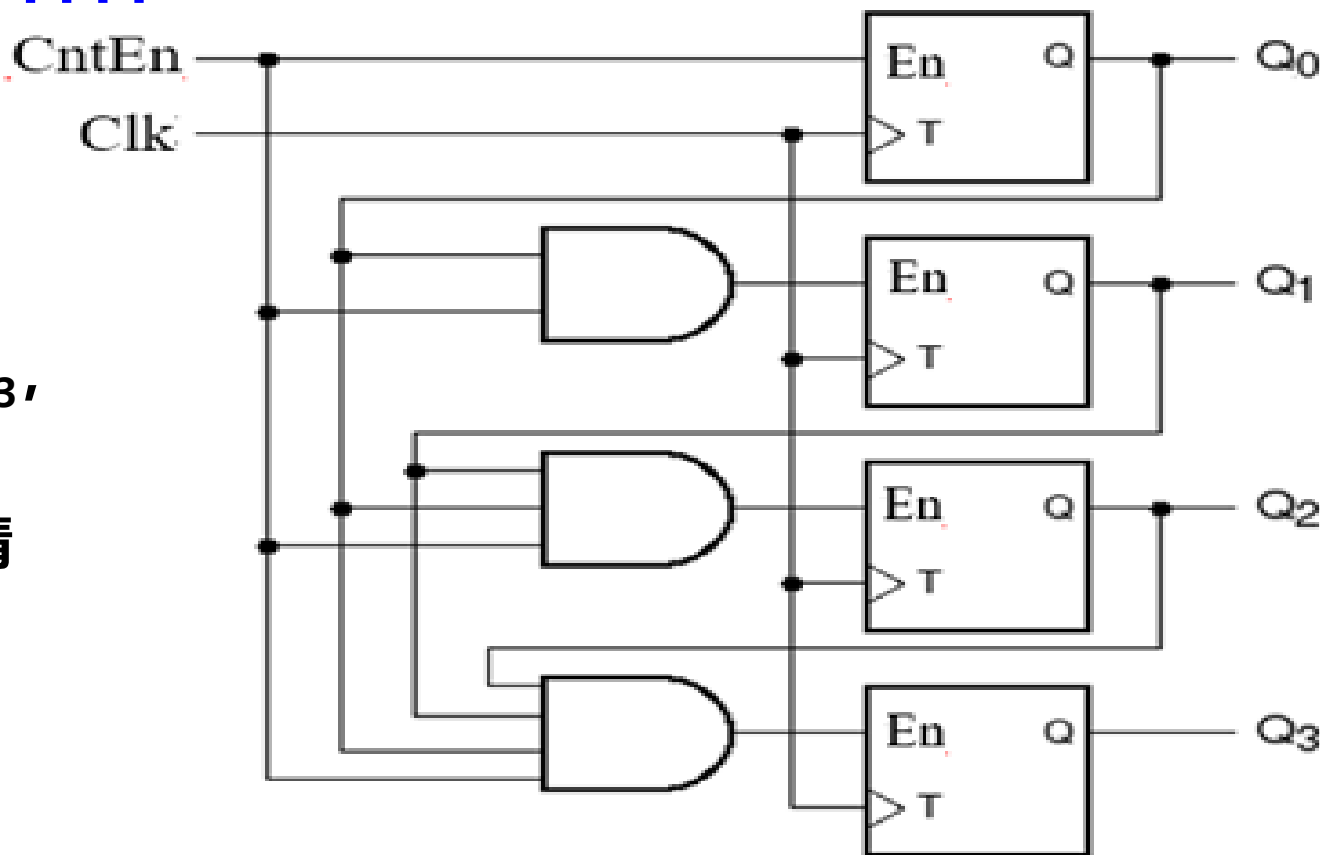
# 4.1 计数器

当编码为1111时，只要经过一个与门  
+  $t_{TQ}$  延时，就可回到编码0000，比行  
波（串行）加法计数器快得多！

## ◆ 同步并行加法计数器

- 计数器的状态编码  $Q_3Q_2Q_1Q_0$  从0000开始，转换过程为  
0000 → 0001 → 0010 → 0011 → 0100 → 0101 → 0110 → 0111 →  
1000 → ... → 1111

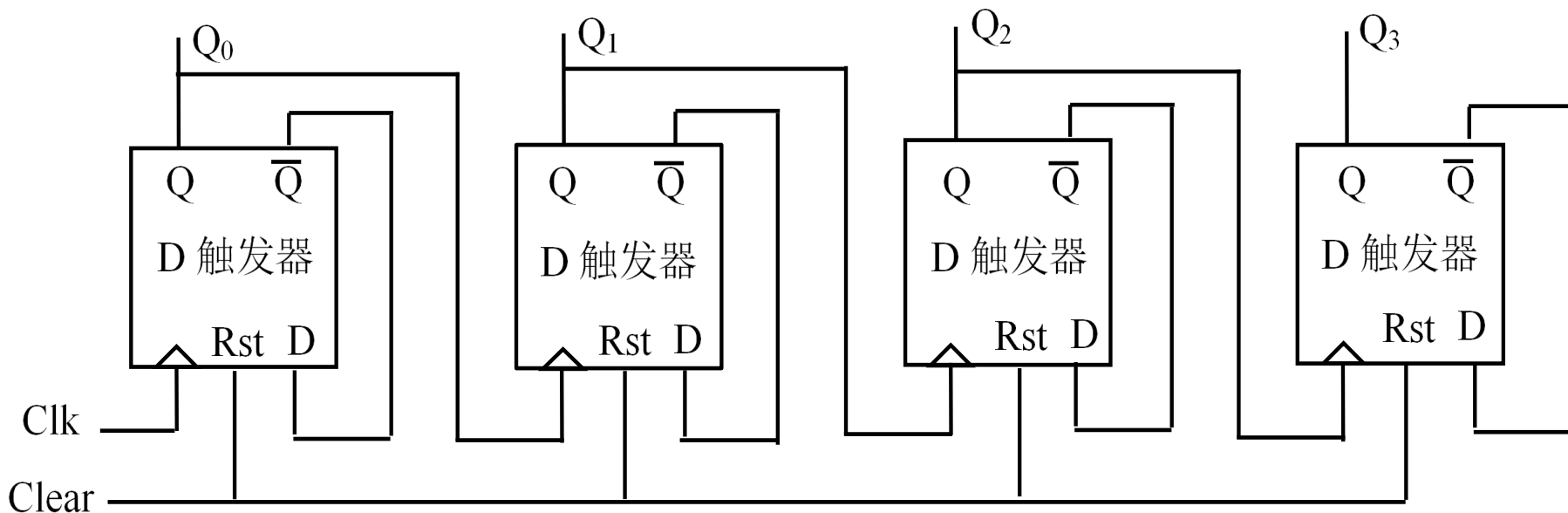
CntEN有效时，  
每个时钟 $Q_0$ 都会  
发生状态改变；  
对于 $Q_1$ 、 $Q_2$ 和 $Q_3$ ，  
只有在其所有低  
位状态都是1的情  
况下，下个时钟  
边沿到来后才会  
发生状态反转。



# 4.1 計數器

## ◆ 二進制異步行波減法計數器

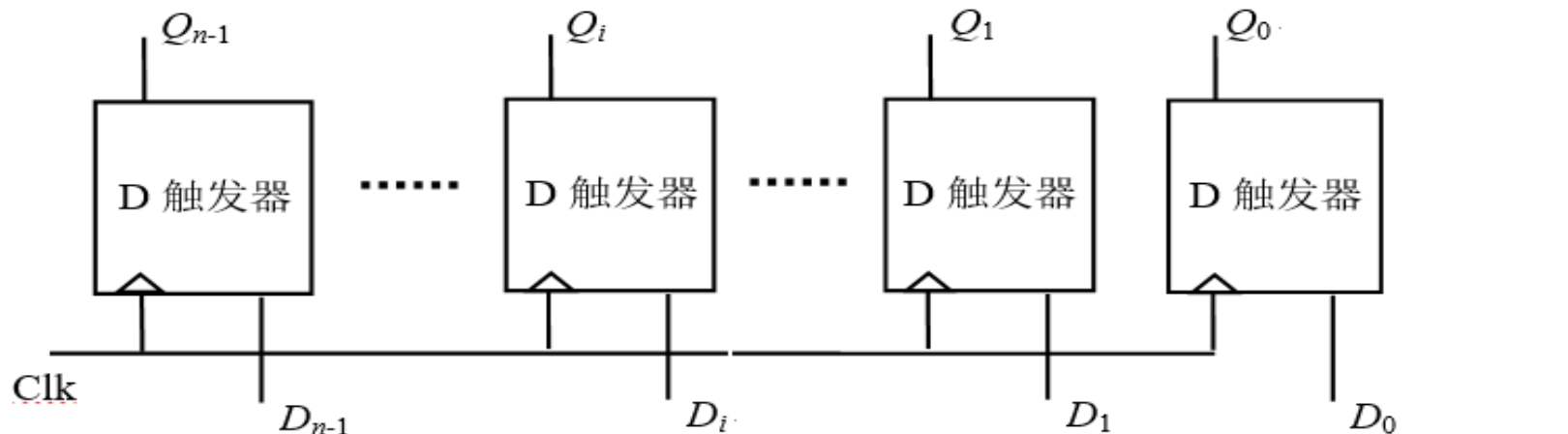
- 由4個上升沿觸發的D觸發器組成，Clear為復位（清0）信號



- 通過Clear信號使所有D觸發器清0，得到初始狀態編碼0000
- 以後每來一個時鐘，發生一次狀態轉換，其過程為  
0000→1111→1110→1101→1100→1011→1010→1001→  
1000→...→0001→0000。

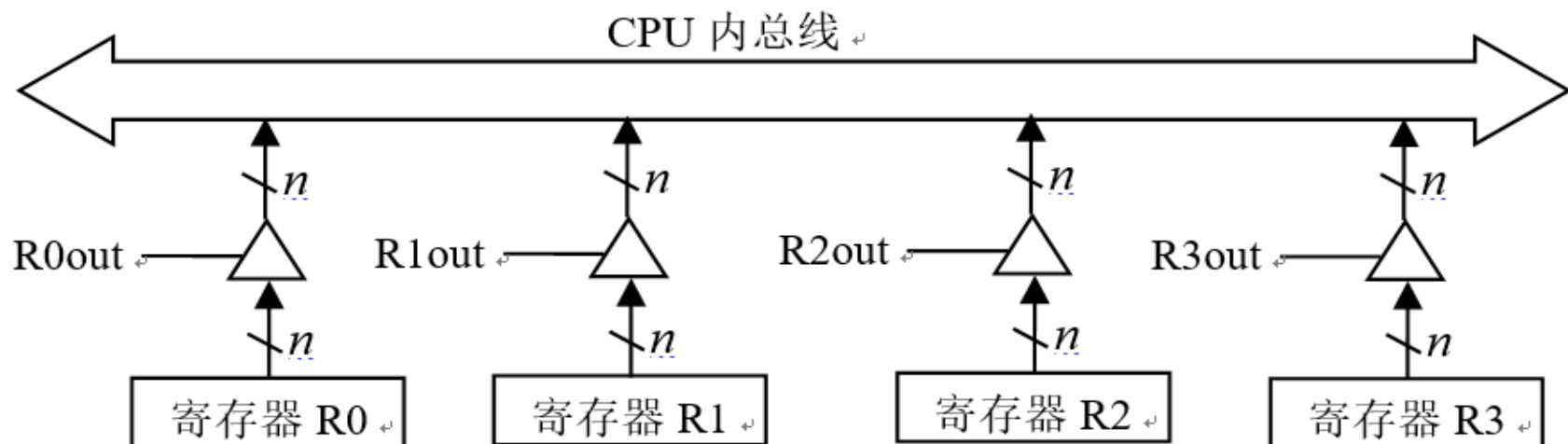
## 4.2 寄存器和寄存器堆

- ◆ 寄存器是用来暂存信息的逻辑部件
- ◆ 寄存器可直接由若干个触发器组成



- ◆ 寄存器通过三态门和总线互连

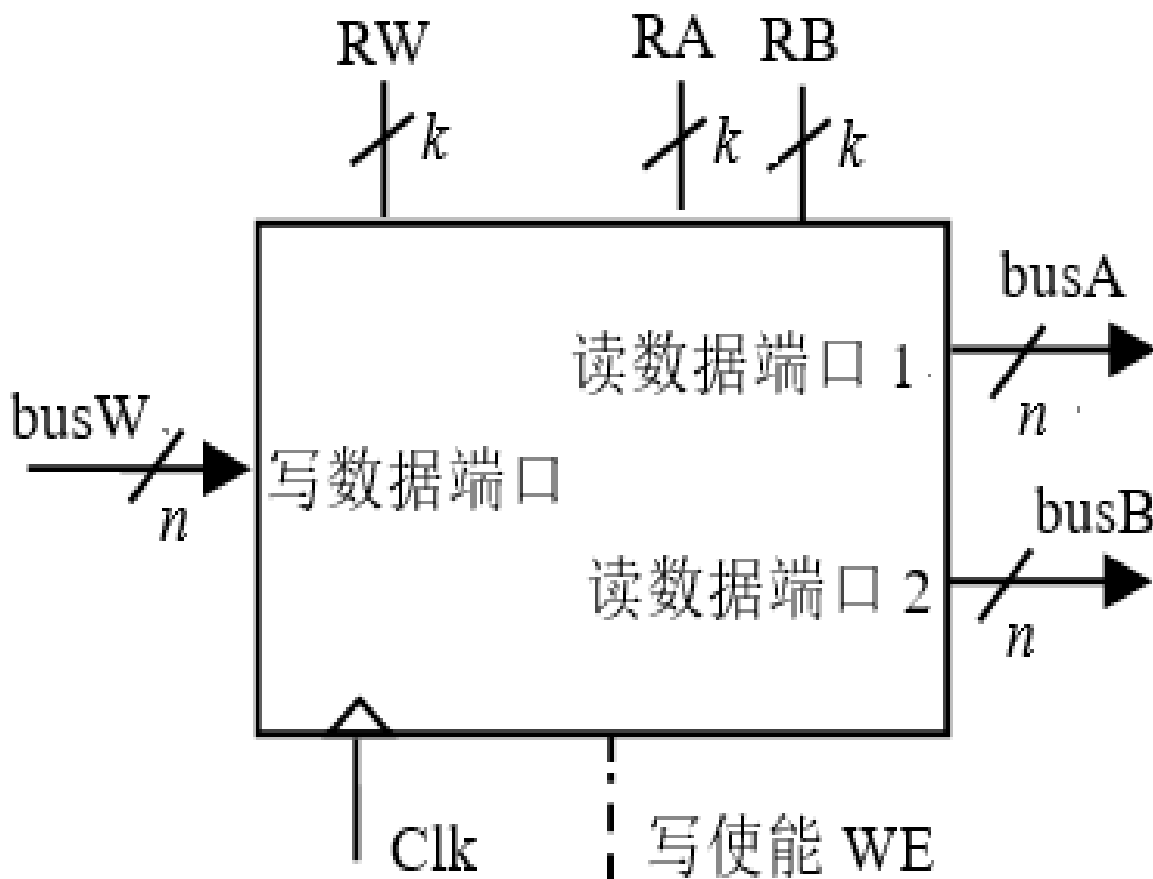
任何时刻至多只能一个Rout有效



## 4.2 寄存器和寄存器堆

- ◆ 寄存器堆(Register File): CPU内部用于暂存指令执行过程中的中间数据, 也称**通用寄存器组**(General Purpose Register set, GPRs)
- ◆ 由许多寄存器组成, 每个寄存器有一个编号, CPU可对指定编号的寄存器进行读写

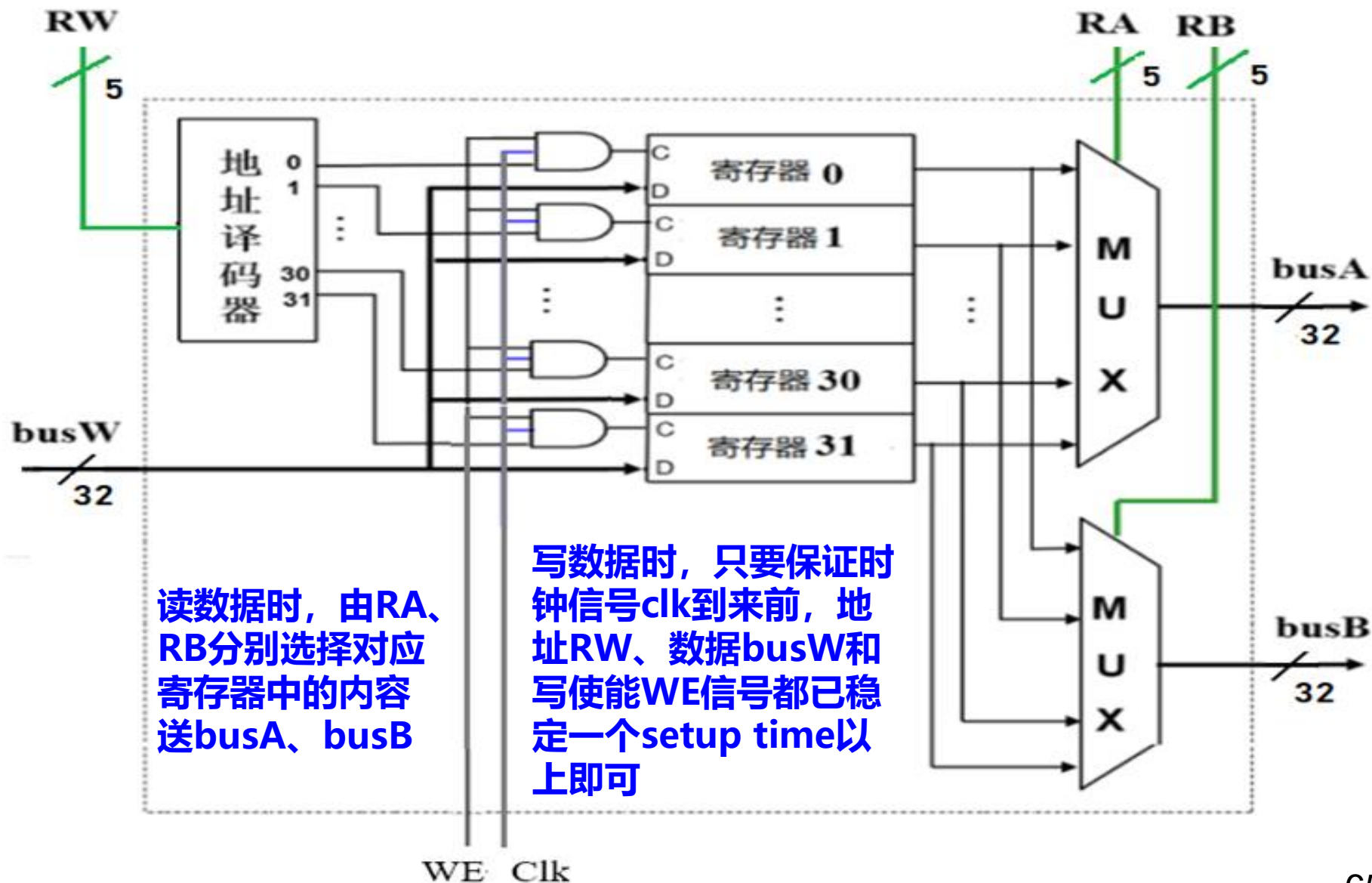
寄存器堆中共有 $2^k$ 个寄存器, 每个寄存器位数为 $n$ ,  $RA$ 和 $RB$ 分别是读口1和读口2的寄存器编号,  $RW$ 是写口的寄存器编号  
读操作属于组合逻辑操作; 写操作属于时序逻辑操作, 需要时钟信号 $Clk$ 和写使能信号 $WE$ 的控制





## 4.2 寄存器和寄存器堆

### ◆ 寄存器堆内部结构

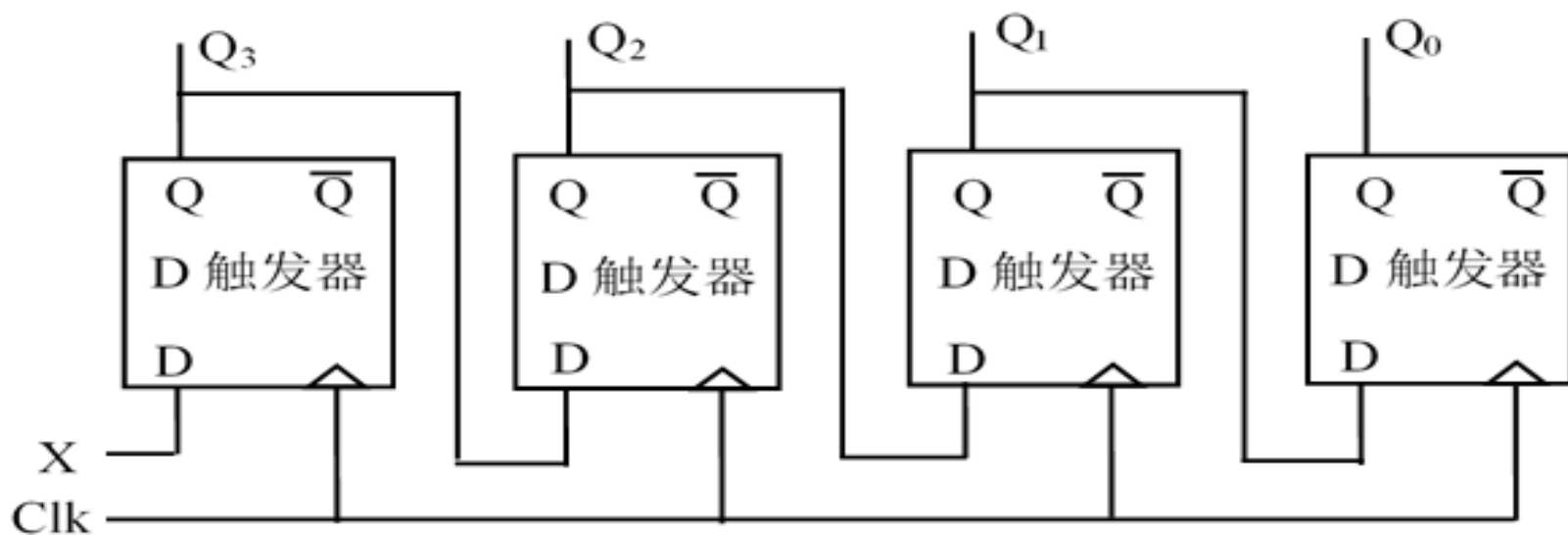


## 4.3 移位寄存器

### ◆ 移位寄存器

- 能够实现暂存信息的左移或右移功能，通常由时钟信号控制

例如：4个D触发器可构成一个右移寄存器



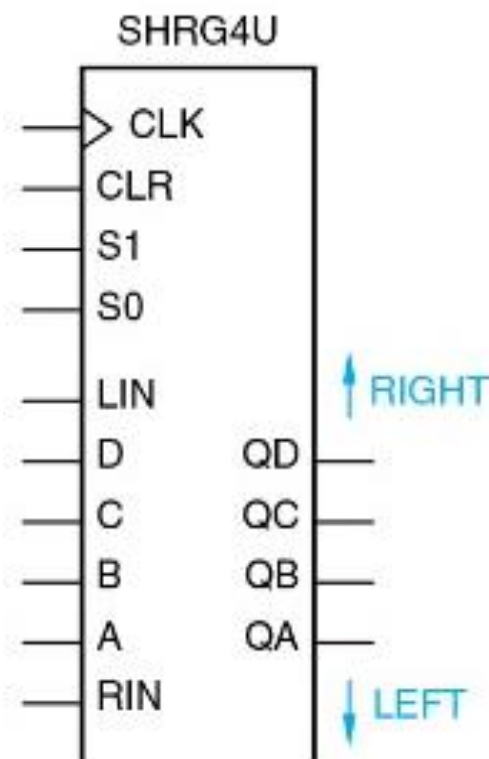
假设初始状态编码 $Q_3Q_2Q_1Q_0=0000$ ，X输入为序列10011011

则 $Q_3Q_2Q_1Q_0$ 的输出编码依次为：0000、1000、0100、0010、  
1001、1100、0110、1011、1101

## 4.3 移位寄存器

### ◆ 4位通用移位寄存器，如74X194

- 具有数据左移、数据右移、数据保持和数据载入功能



Function	Inputs			Next state			
	CLR	S1	S0	QA*	QB*	QC*	QD*
Clear	1	x	x	0	0	0	0
Hold	0	0	0	QA	QB	QC	QD
Shift right	0	0	1	RIN	QA	QB	QC
Shift left	0	1	0	QB	QC	QD	LIN
Load	0	1	1	A	B	C	D

左移从QD移到QA向下移 ↓ 右移从QA移到QD向上移 ↑

## 4.3 移位寄存

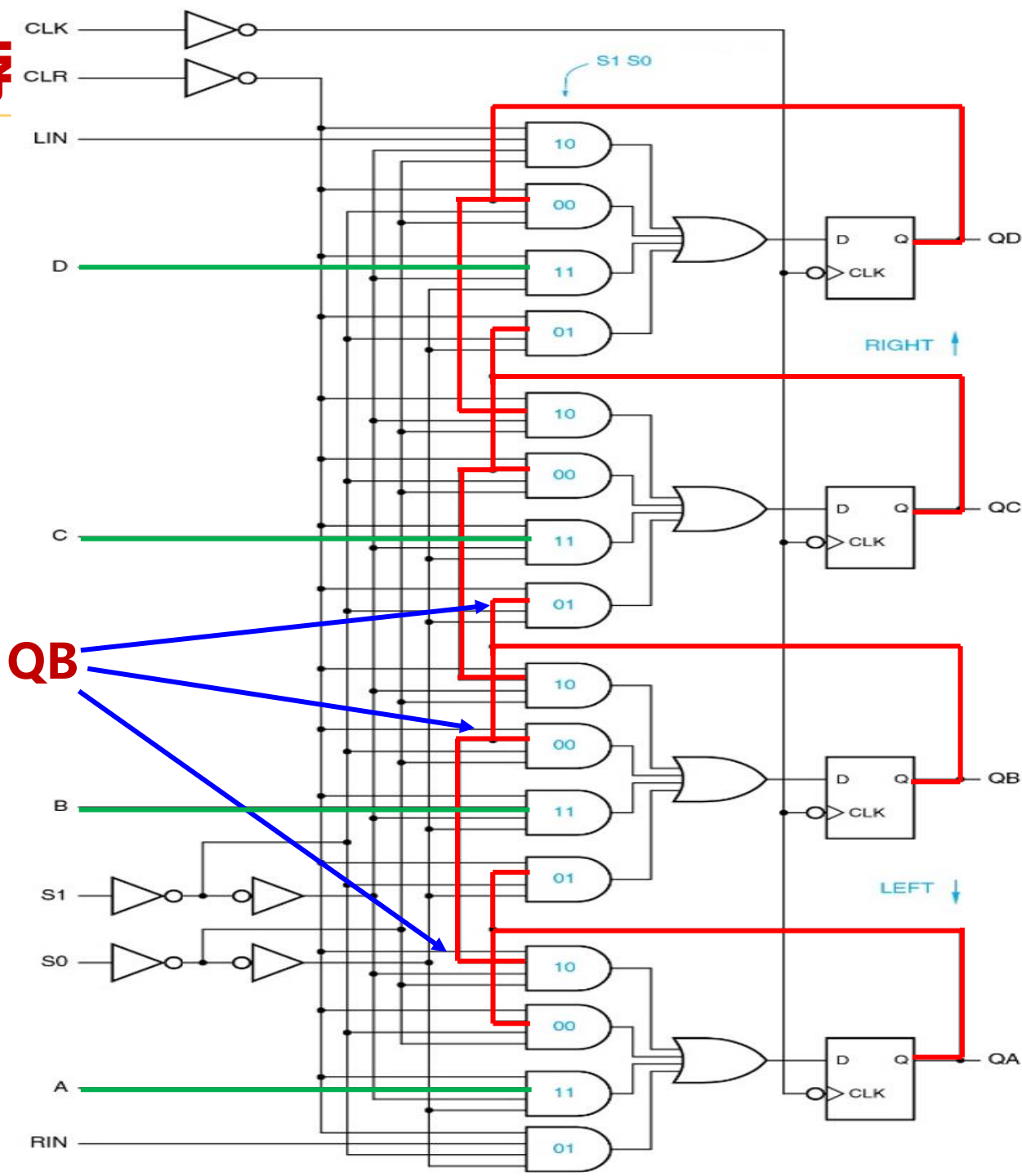
### ◆ 4位通用移位寄存器结构图

**S1S0=00:** 保持

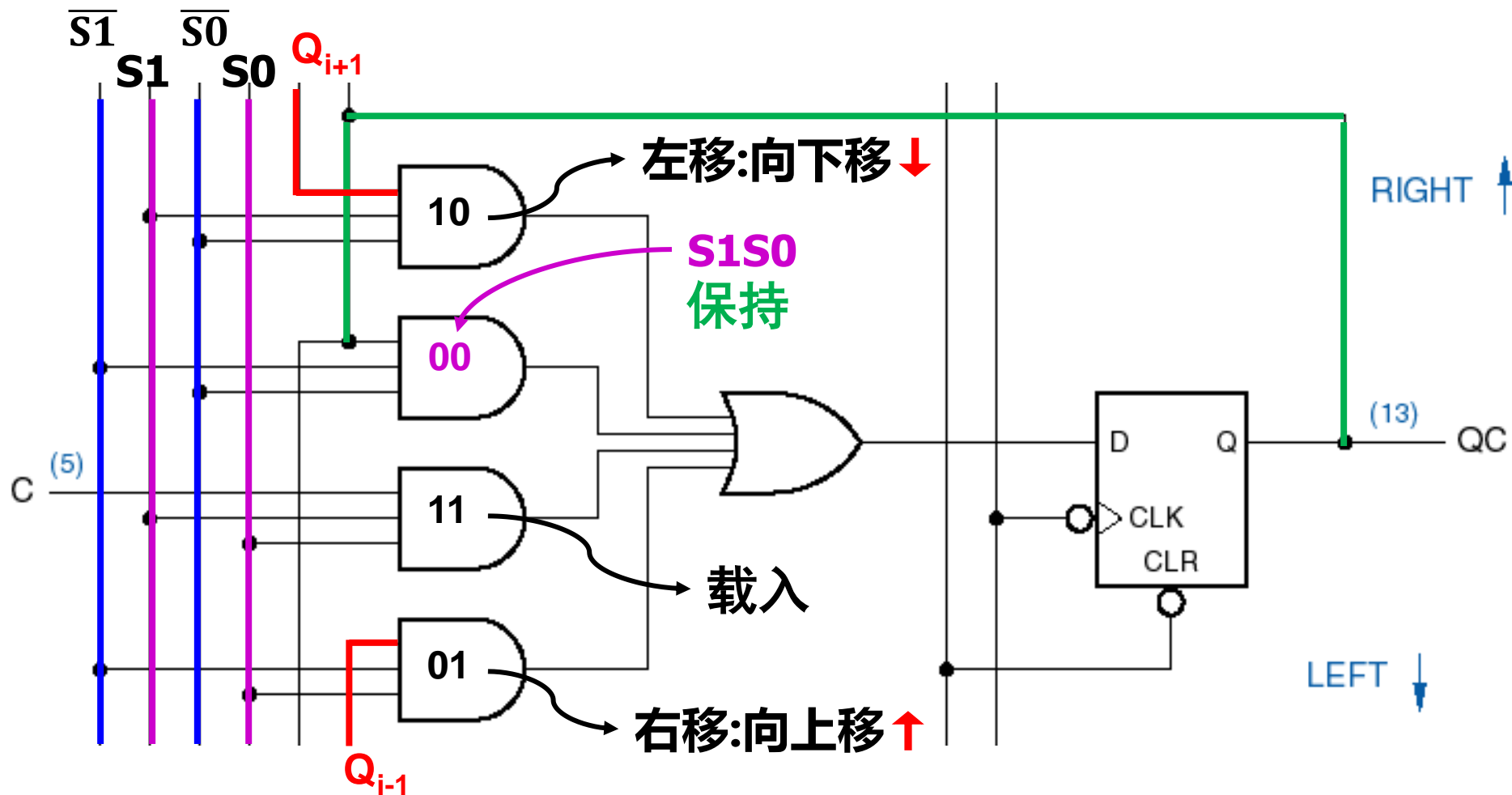
**S1S0=01:** 上移

**S1S0=10:** 下移

**S1S0=11:** 加载



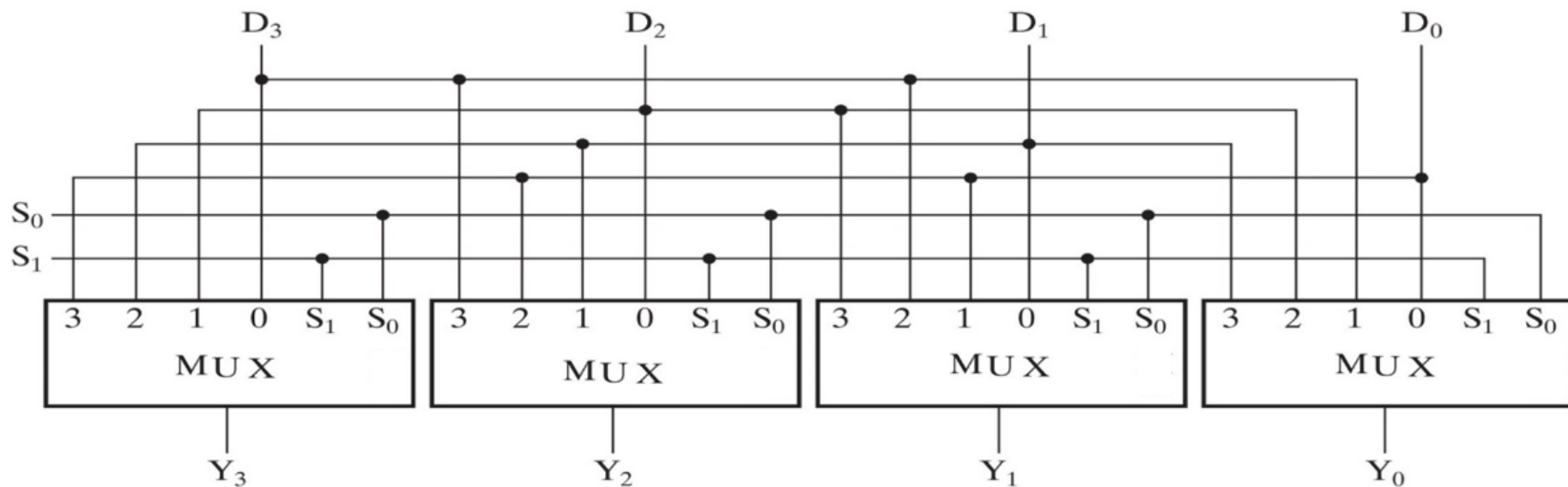
## 4.3 移位寄存器



$$D_i = \overline{S1} \cdot \overline{S0} \cdot Q_i + \overline{S1} \cdot S0 \cdot Q_{i-1} + S1 \cdot \overline{S0} \cdot Q_{i+1} + S1 \cdot S0 \cdot IN_i$$

## 4.3 移位寄存器

◆ 桶形移位寄存器：組合邏輯電路，採用大量多路選擇器實現



Function Table for 4-Bit Barrel Shifter

Select		Output				Operation
S <sub>1</sub>	S <sub>0</sub>	Y <sub>3</sub>	Y <sub>2</sub>	Y <sub>1</sub>	Y <sub>0</sub>	
0	0	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	No rotation
0	1	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	D <sub>3</sub>	Rotate one position
1	0	D <sub>1</sub>	D <sub>0</sub>	D <sub>3</sub>	D <sub>2</sub>	Rotate two positions
1	1	D <sub>0</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	Rotate three positions

4位桶形移  
位寄存器的  
實現和功能