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

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、现态、旧状态）
  - 电路中有存储元件，用于存储逻辑信号的值，表示电路过去时刻的行为（当前状态、现态、旧状态）
  - 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

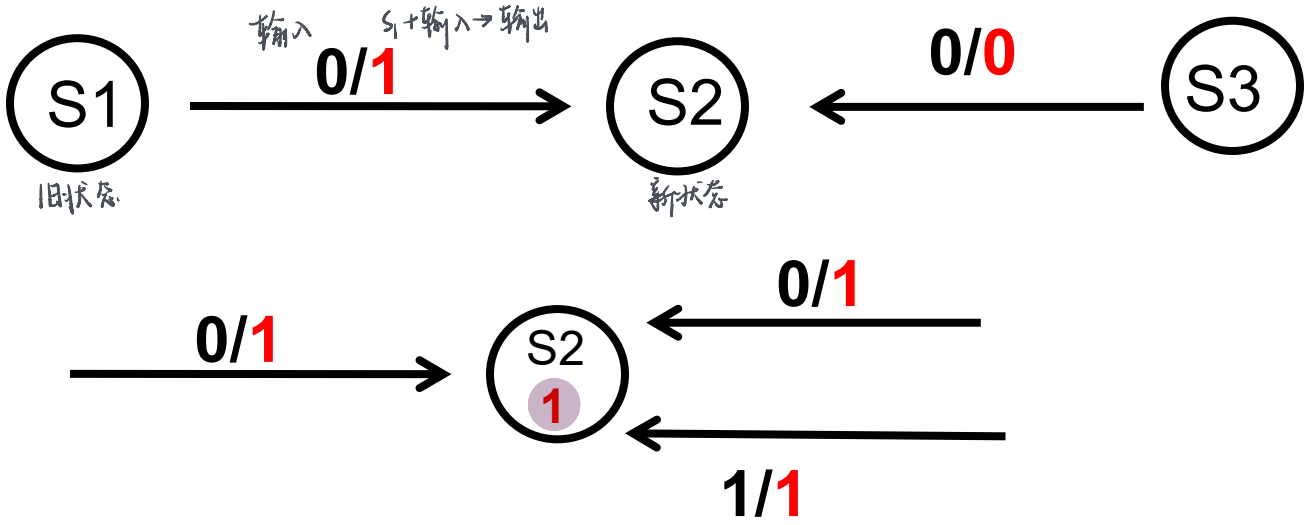
如,电视机的音量控制

通过音量按钮“+”和“-”输入当前信号，得到的下一次音量（下一个状态）由原音量（当前状态）和当前所按按钮决定

如何有效反映下一状态和当前状态及当前输入之间的转换关系呢?

1.1 有限状态机

- 有限状态机（Finite State Machine，FSM）是一种刻画状态以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（若输出只与当前状态有关，则可以把输出标注在状态圆圈中）



1.1 有限状态机—状态图

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

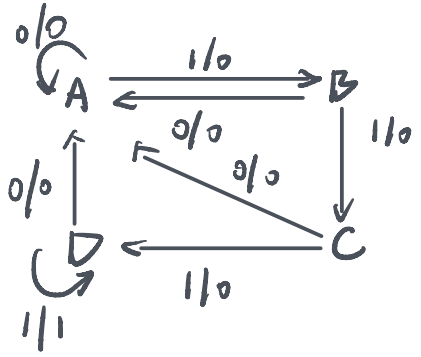
B-连续1个“1”：若输入1，则转C

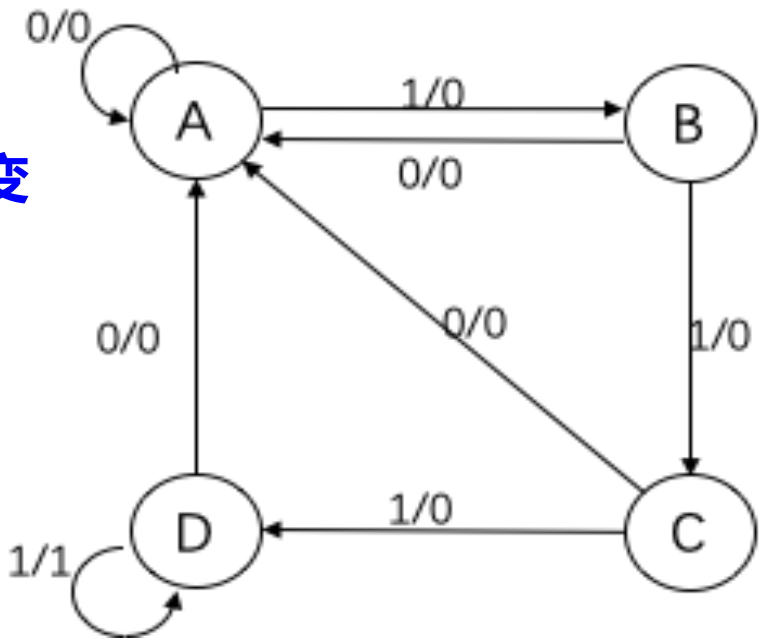
C-连续2个“1”：若输入1，则转D

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





如何用数字逻辑实现有限状态机呢?

1.1 用数字逻辑实现有限状态机

- 用数字逻辑实现一个有限状态机，需要完成的主要工作：
  - 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。
  - 设计一种状态记忆电路（存储元件）。使用双稳态器件记忆状态，如SR锁存器、D触发器、JK触发器等；
  - 设计状态记忆电路的激励函数（根据当前输入把旧状态改为新状态的函数）和输出函数（根据旧状态和当前输入来改变电路输出结果的函数），并完成定时分析。

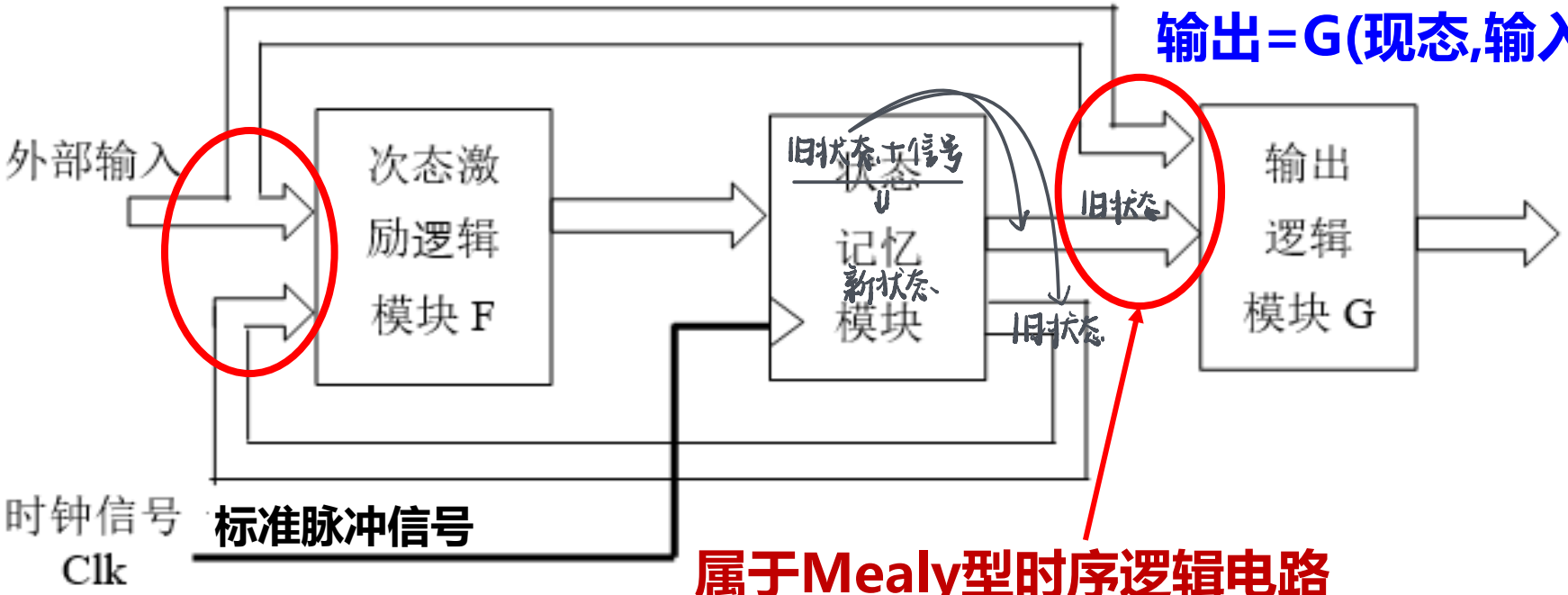
能完成上述工作（实现有限状态机）的数字逻辑电路一定是一个时序逻辑电路！

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（存储元件）
  - **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

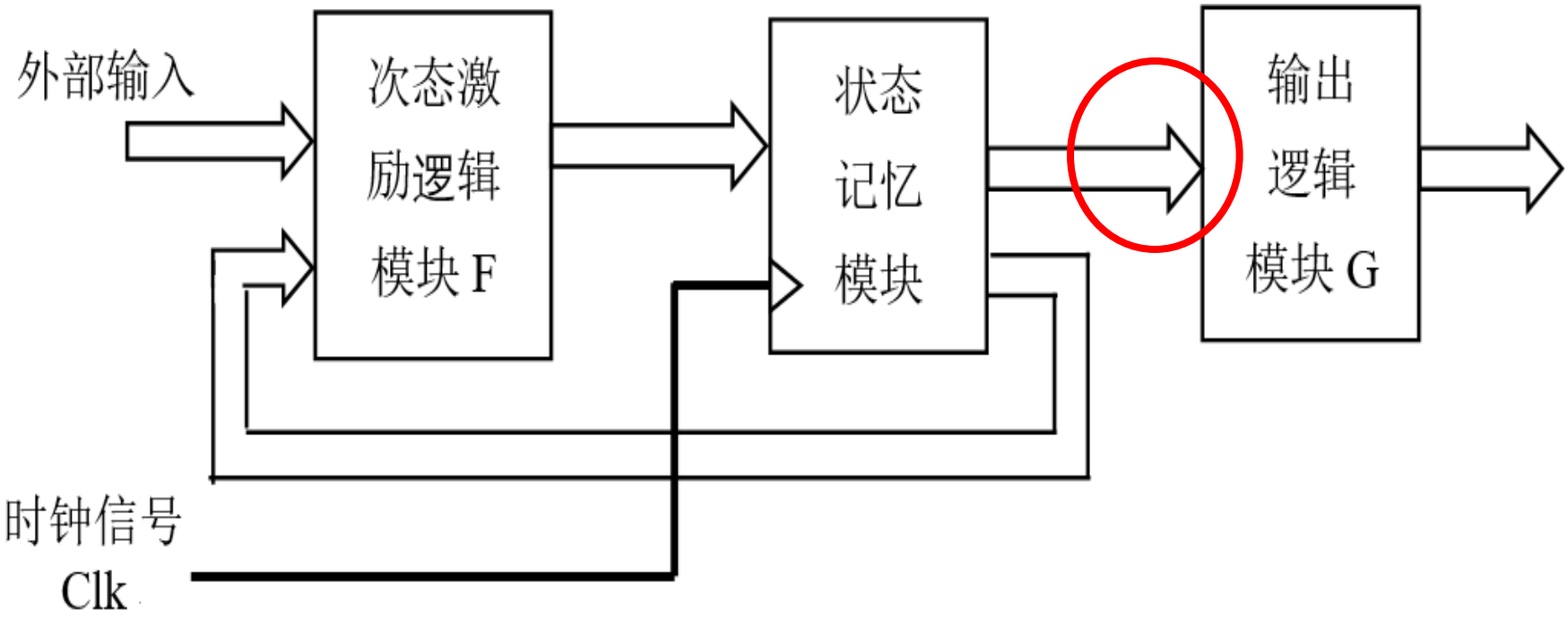
    Moore型：输出仅依赖于当前状态，和当前输入信号无关



1.2 时序逻辑电路基本结构

- **Moore型：输出信号仅依赖于当前状态。**

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 e.g. CPU
  - 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
- **Clk：固定周期的标准脉冲信号**

补：时钟脉冲

- **按一定电压幅度、按固定时间间隔、连续发出的脉冲信号。**

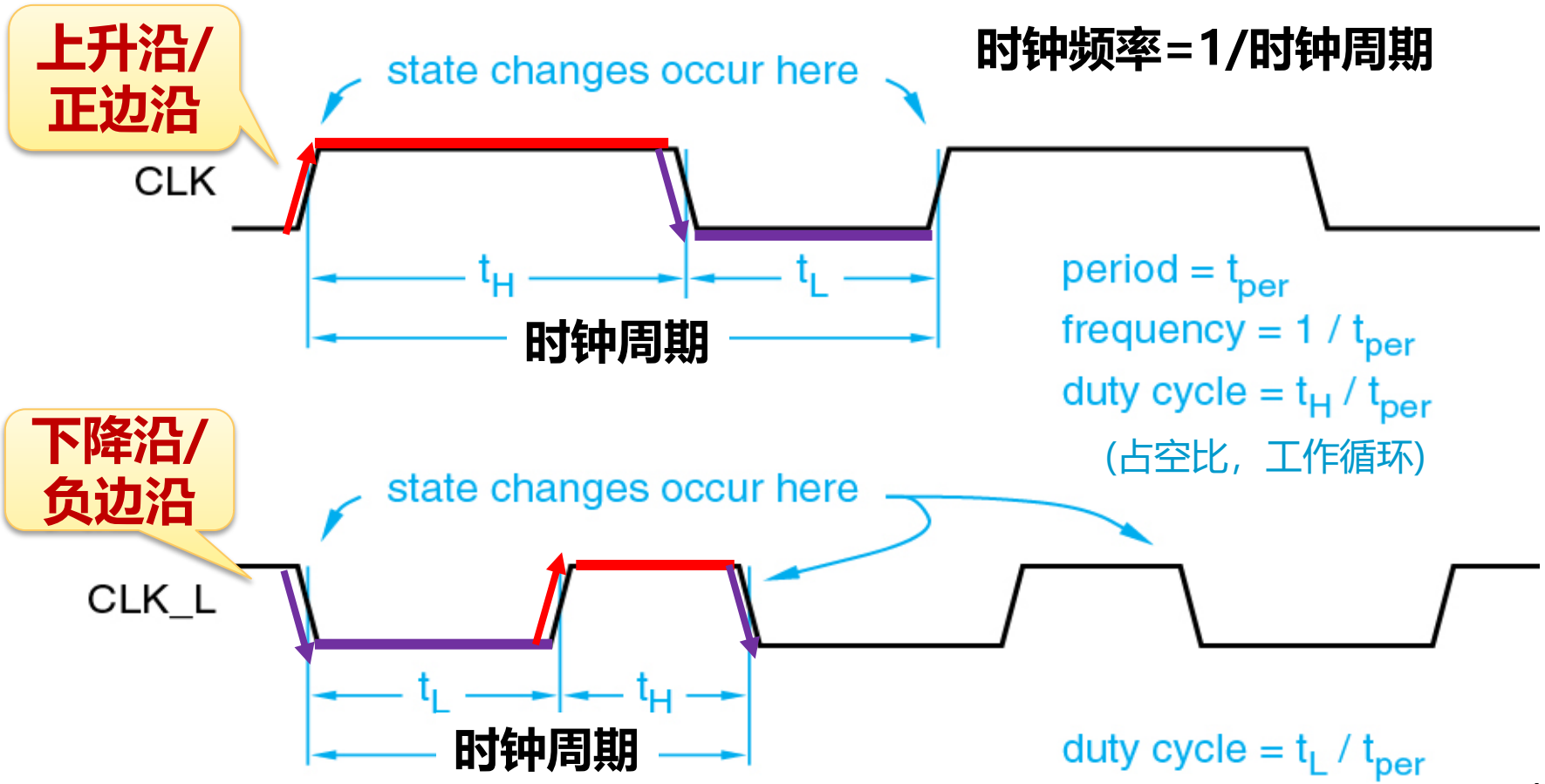
(*) - 时钟都是通过振荡器产生的。振荡器有很多种，根据不同的时钟需求会使用不同的振荡器。

(*) - 原始时钟信号一般会通过晶体振荡器产生。根据晶体特性存在一个谐振频率，而且品质因子（目标频率能量占总能量的比值）非常高。从而能够产生一个噪声非常小、震荡频率非常精确的时钟信号。

- **时钟脉冲之间的时间间隔称为时钟周期。单位是秒。**
- **通常将1秒内所产生的脉冲个数称为时钟频率。单位是Hz**
- **计算机中的系统时钟就是一个典型的、频率精确和稳定的脉冲信号发生器。**

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
- 边沿触发方式分为上升沿触发和下降沿触发两种类型

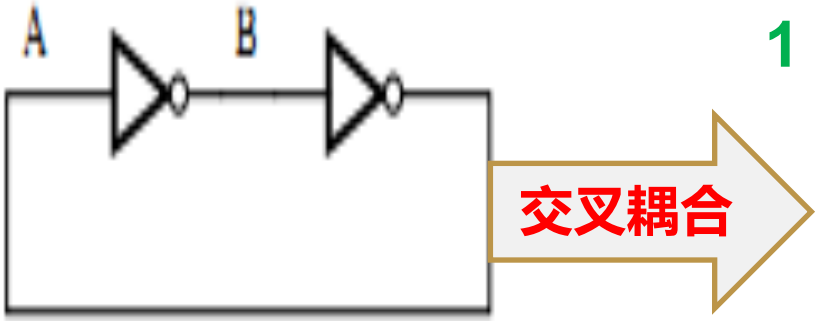


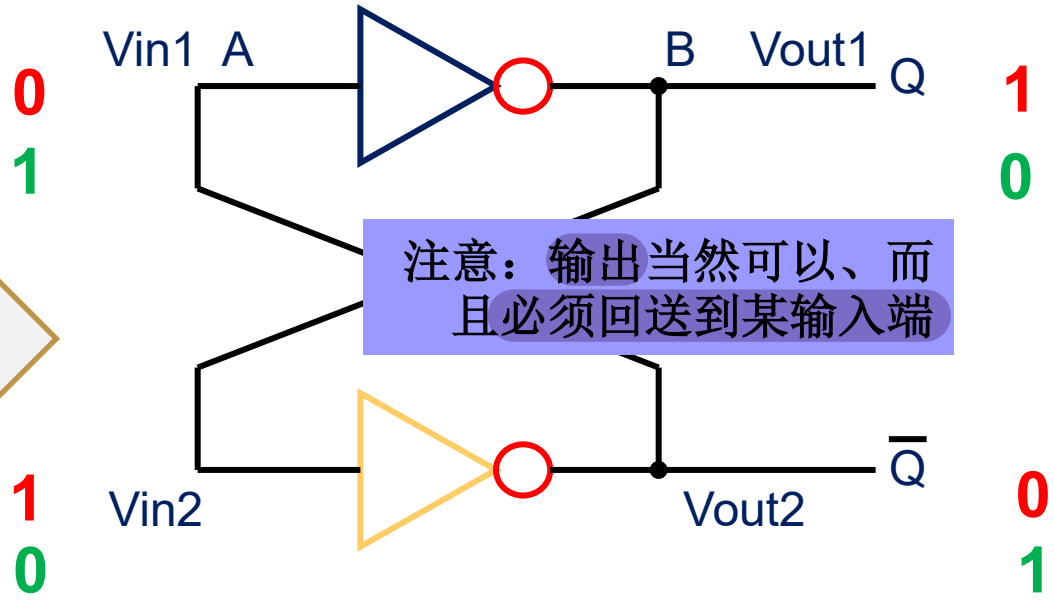
第二讲 锁存器和触发器

- 双稳态元件
- SR锁存器
- D锁存器
- D触发器
- T触发器

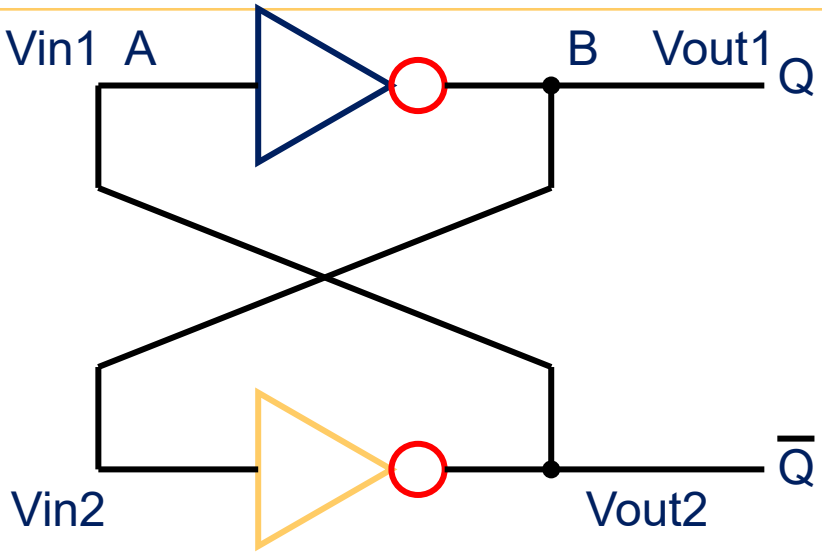
2.1 双稳态元件

- 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，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





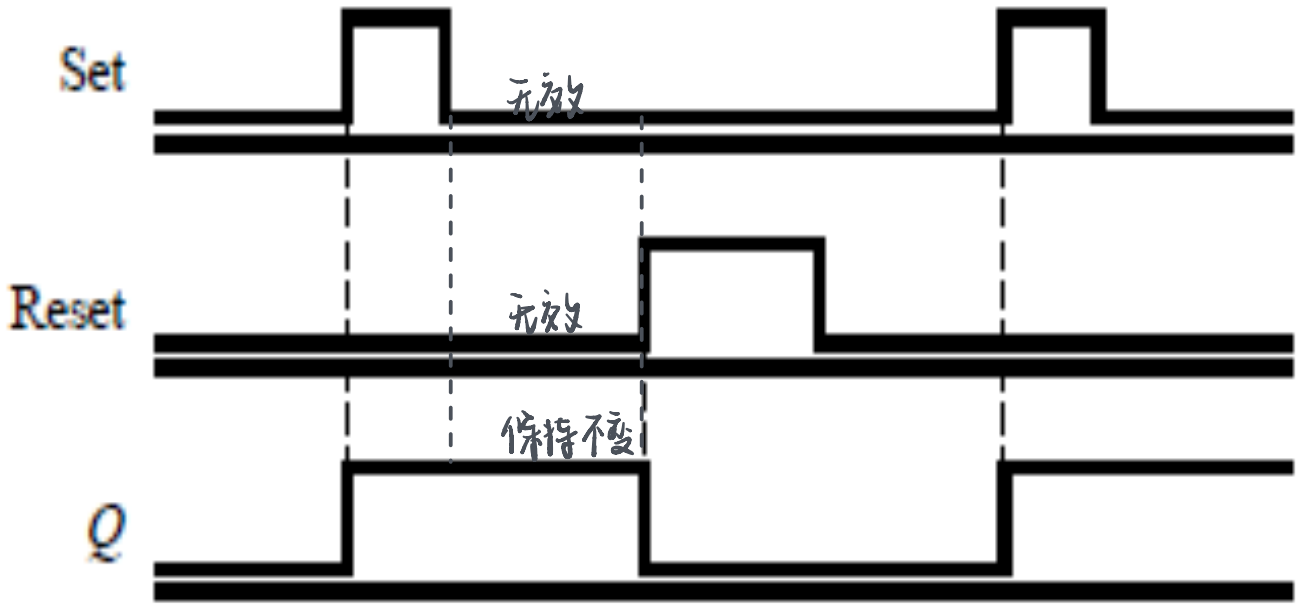
2.1 双稳态元件



- 用两个反相器串联构建的双稳态元件无法改变电路状态
- 用1个或多个输入信号能驱动双稳态元件进入稳定状态，这些输入信号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
- 根据触发方式的不同，基于双稳态元件的构建思路，可以实现两种类型的存储元件：
  - 锁存器(latch) 、触发器(flip-flop)

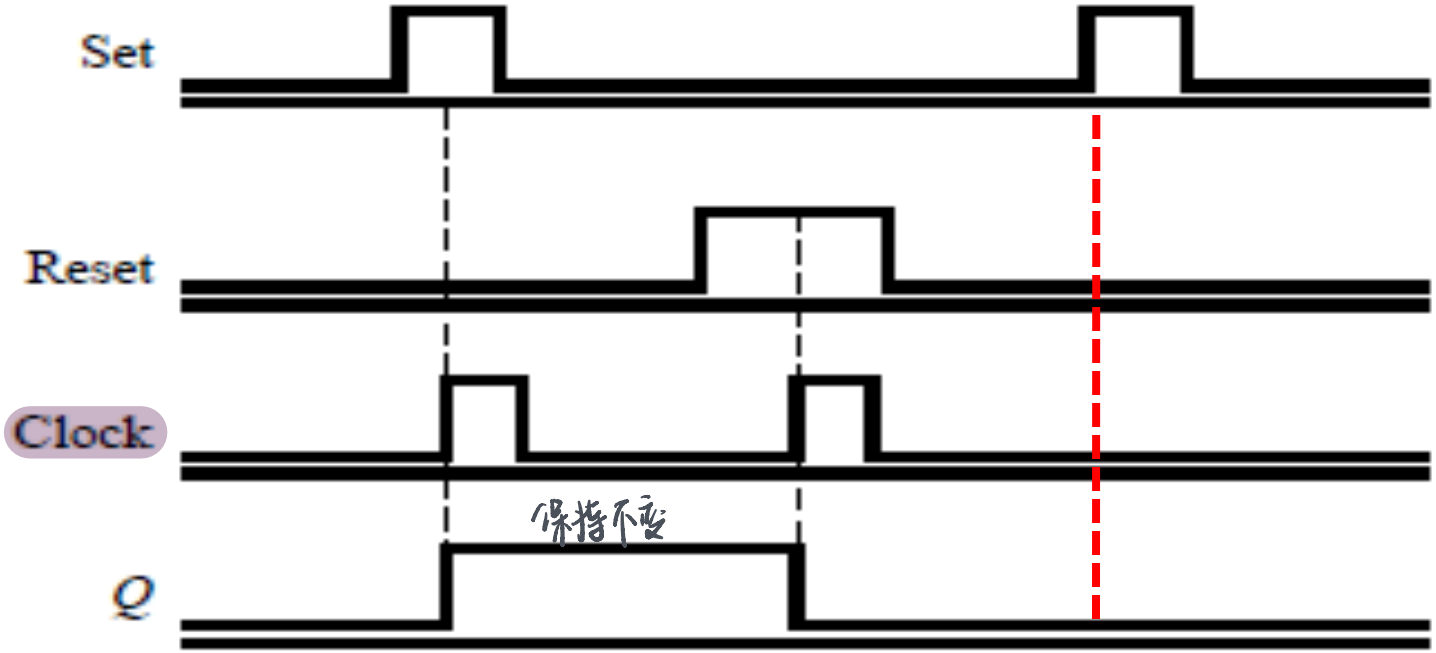
2.1 双稳态元件-锁存器

- 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

- 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

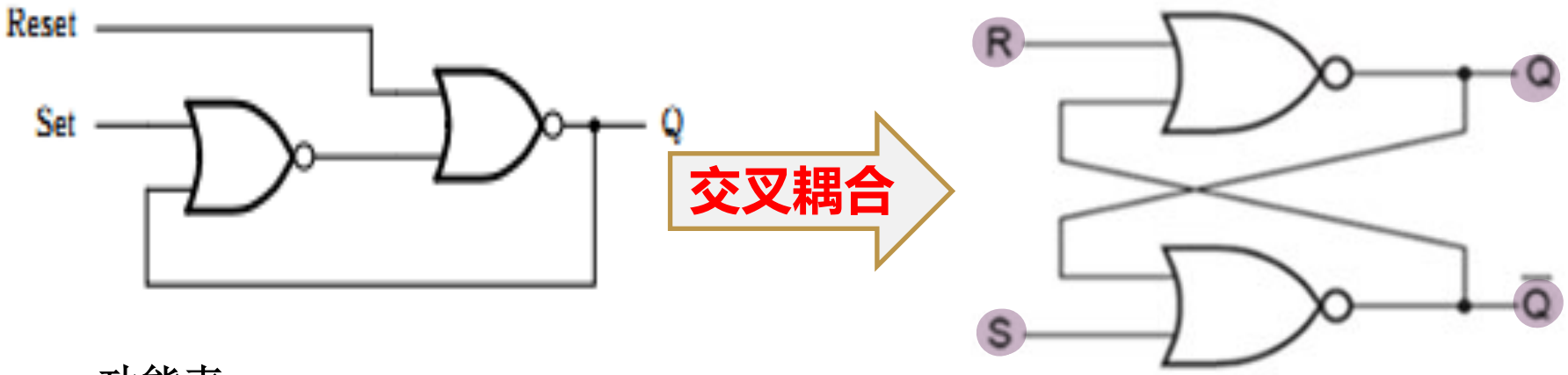


Set信号有效，但必须等到时钟边沿到来，Q才变化

Set信号有效，但没有时钟边沿到来，Q不变

2.2 SR锁存器

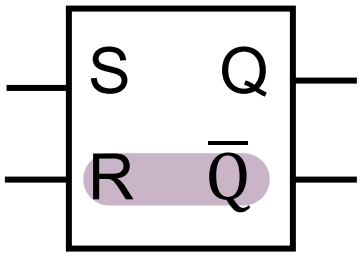
- SR锁存器：使用一对交叉耦合的或非门构成双稳态电路，也称为置位-重置（复位）锁存器。S是置位输入端，R是重置输入端



功能表

S	R	Q	$\overline{Q}$
0	0	状态不变	
0	1	0	1
1	0	1	0
1	1	禁	止

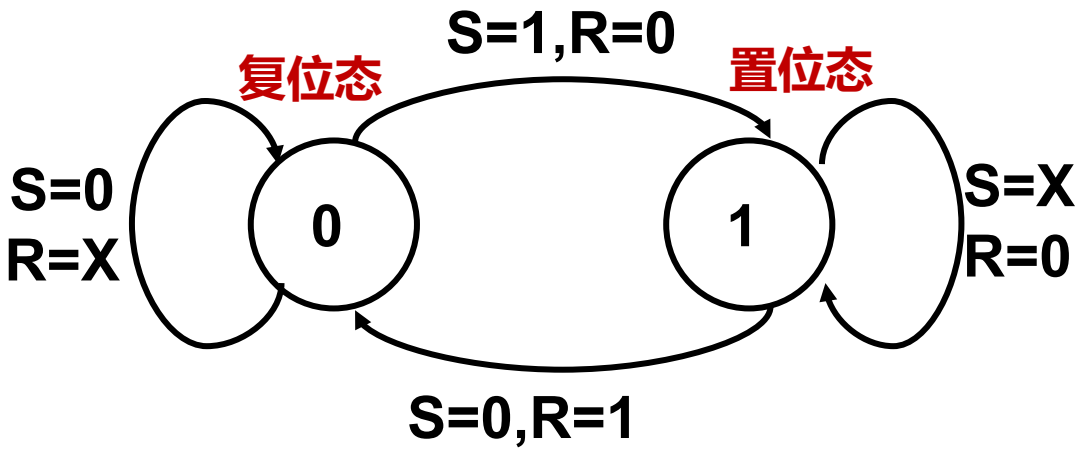
逻辑符号

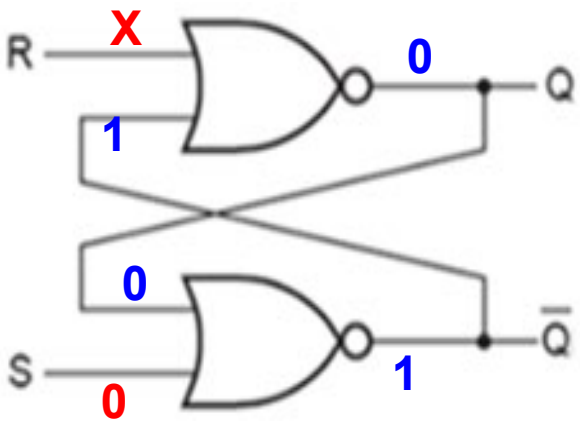


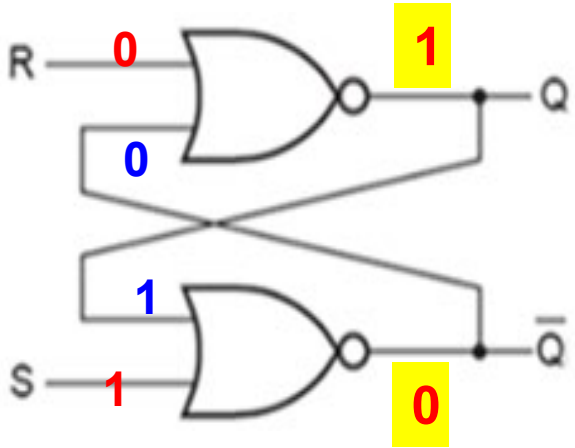
R=S=1时，Q、$\overline{Q}$状态不相反，无效

2.2 SR锁存器

- 状态图







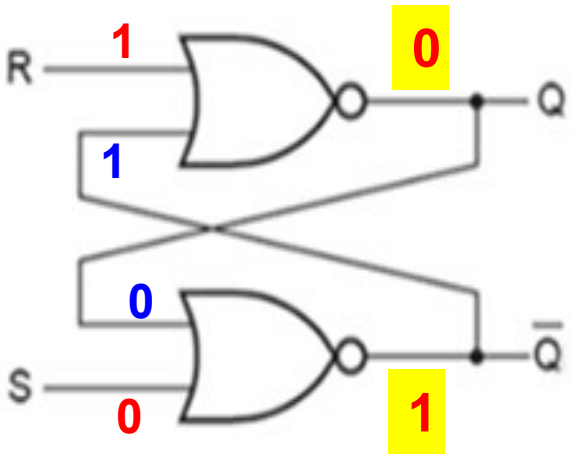
2.2 SR锁存器

- 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
    - （右表中没有“输出”）只有状态转换

状态表

现态 Q	输入RS 00	01	10	11
0	0	1	0	0* (不允许)
1	1	1	0	0*

- 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、次态方程之间可相互转换！

类似真值表

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

- **次态（特征）方程**

次态方程

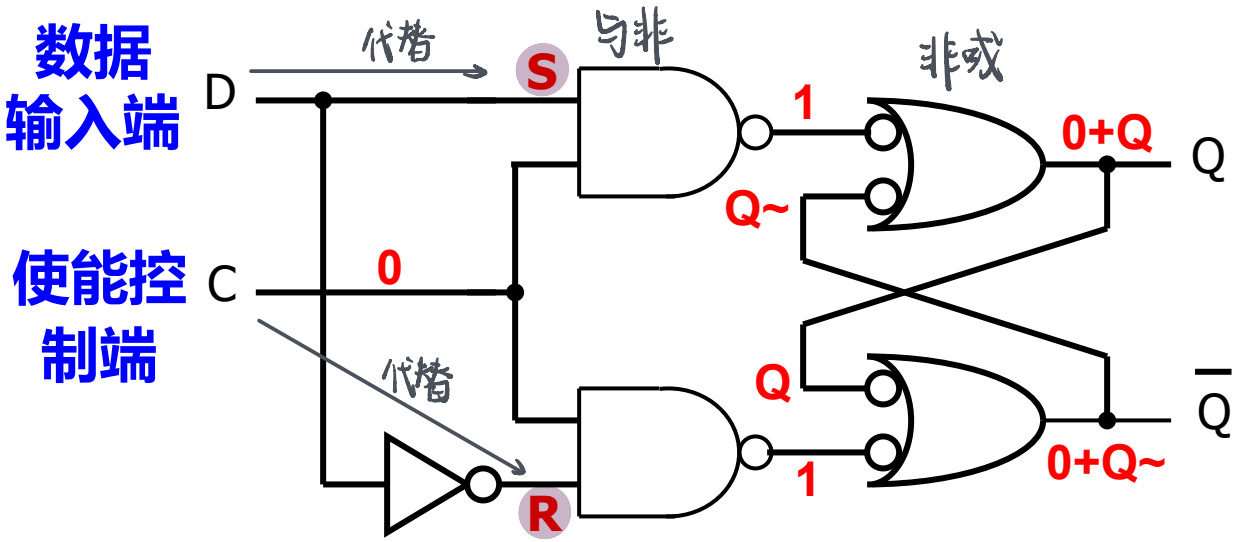
$$\begin{cases} Q^* = S + \overline{R}\cdot Q \\ S\cdot R \neq 1 \quad \text{约束条件} \end{cases}$$

卡诺图化简？ 无效

功能：SR锁存器只有状态，没有输出，常用来设置标志位

2.3 D锁存器

- 如何利用锁存器来存储信息位?



C=0时, 电路状态（Q和Q~）保持不变

只要是0就不变

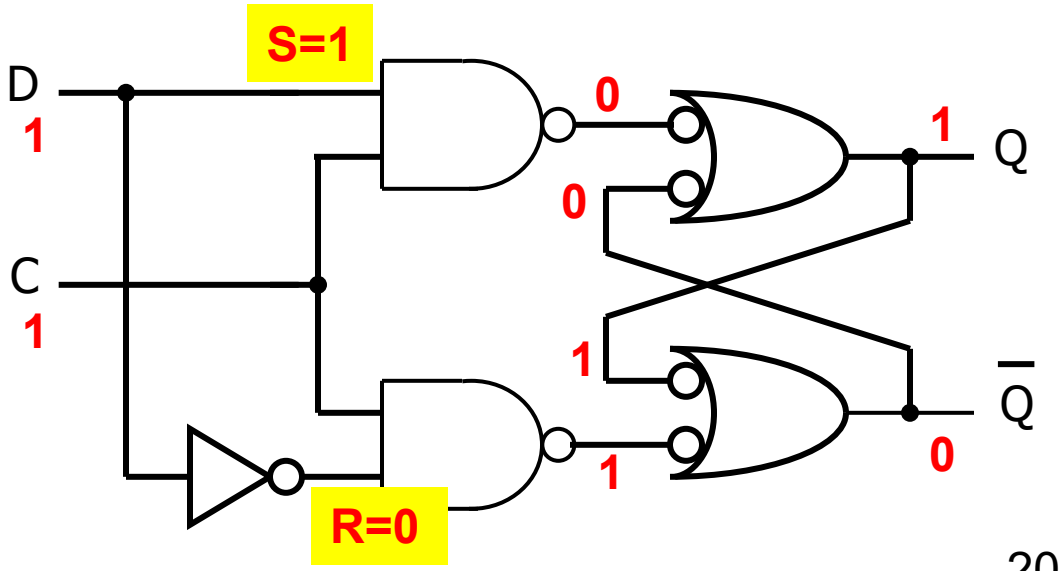
C=1时,

D = 1 时，Q = 1

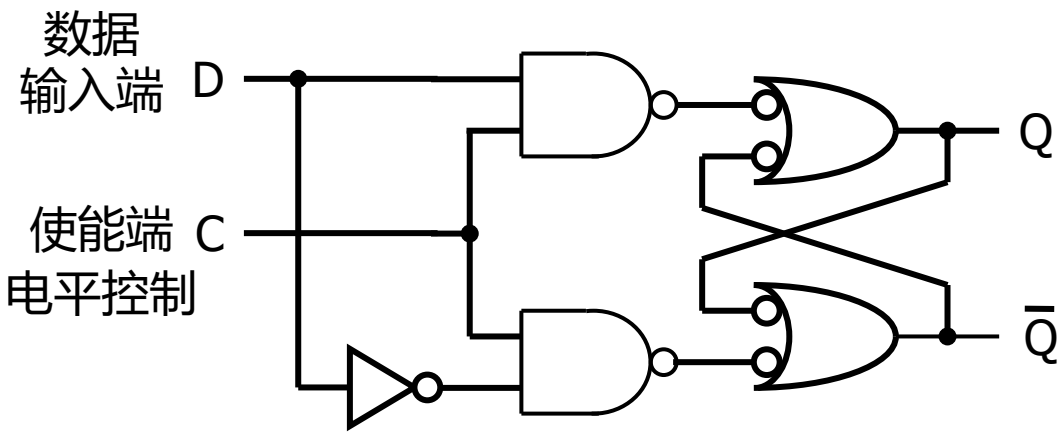
D = 0 时，Q = 0

也就是：Q = D

电路状态随输入而改变



2.3 D锁存器



D锁存器功能表

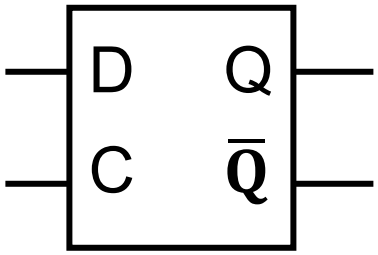
C	D	Q	$\overline{Q}$
0	X	保	持
1	0	0	1
1	1	1	0

C=0时，电路状态保持不变

C=1时，将数据端D锁存

（输入D让电路状态改变了）

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

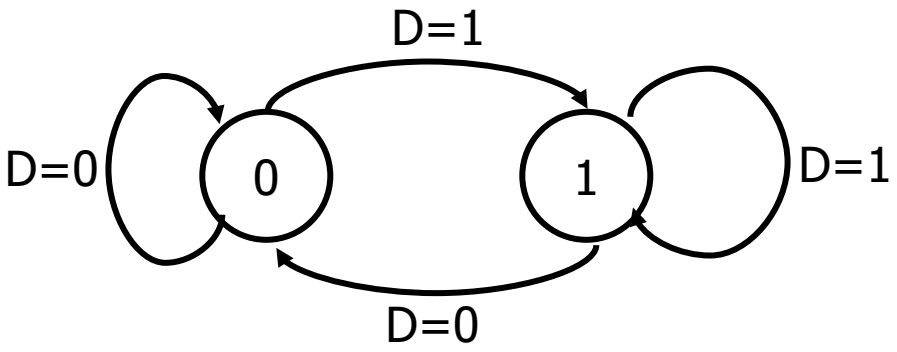
- D锁存器状态转移表、状态图和次态（特征）方程

状态转移表

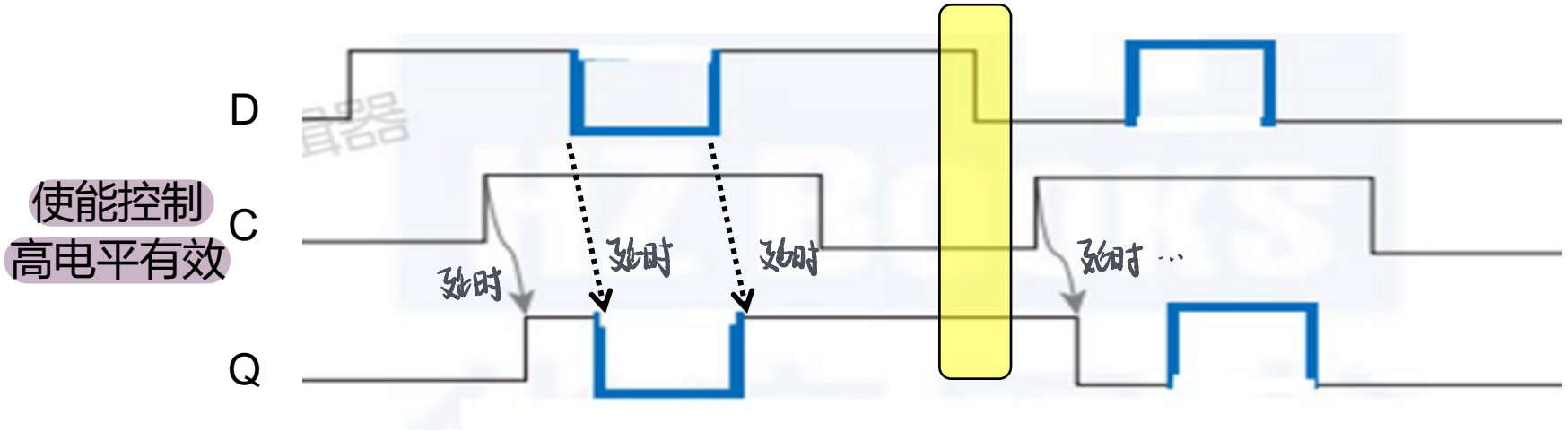
D	Q*
0	0
1	1

特征方程：Q* = D（C=1）

状态图

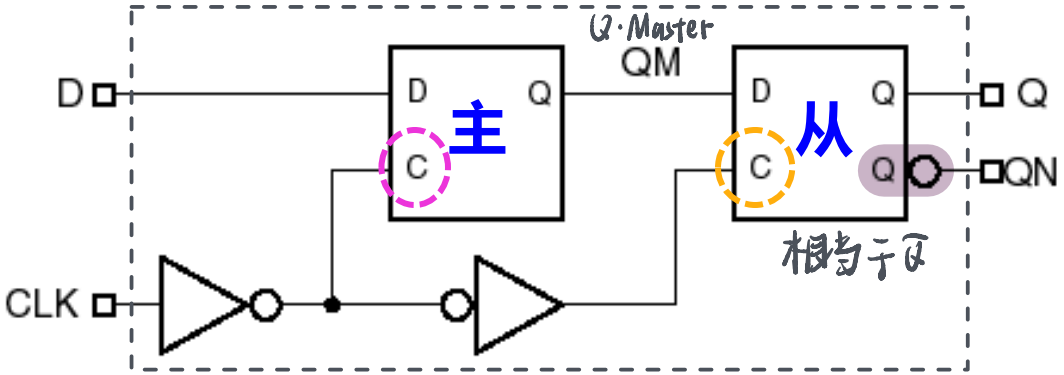


- D锁存器的时序图



2.4 D触发器——开始有时钟信号了！

- 由一对主、从D锁存器构成



CLK	主锁存器	从锁存器
L	写入 QM变为D	不变 Q=last
(上升沿结束) 上升沿	锁存 QM=D	写入 Q变为QM
H	不变 QM=D	不变 Q=QM

(保持，一直工作)

- 从锁存器在时钟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

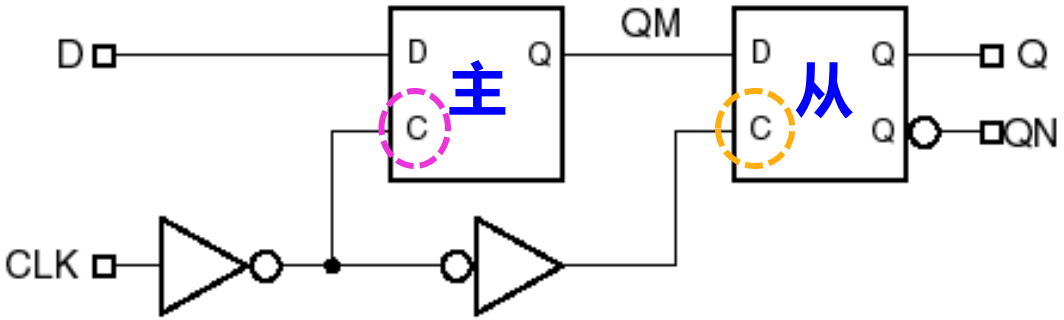
从锁存器完成写入

此时主锁存器是在写入的（C为高电平），但不影响最后输出的Q

此时从锁存器是在持续写入的，但一定是在之前上升沿到来之后就完成了，所以也就相当于是保持不变了

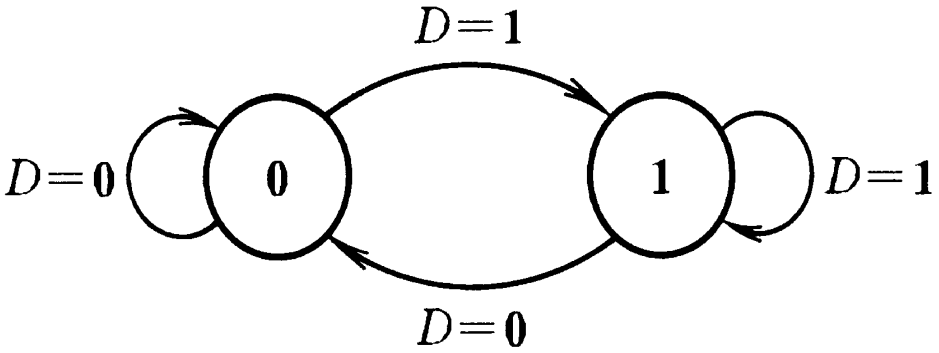
2.4 D触发器

- 由一对主、从D锁存器构成

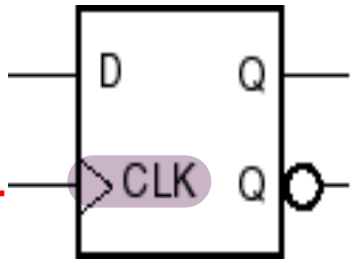


D	CLK	Q	QN
0	↑	0	1
1	↑	1	0
x	0	last Q	last QN
x	1	last Q	last QN

- 状态图



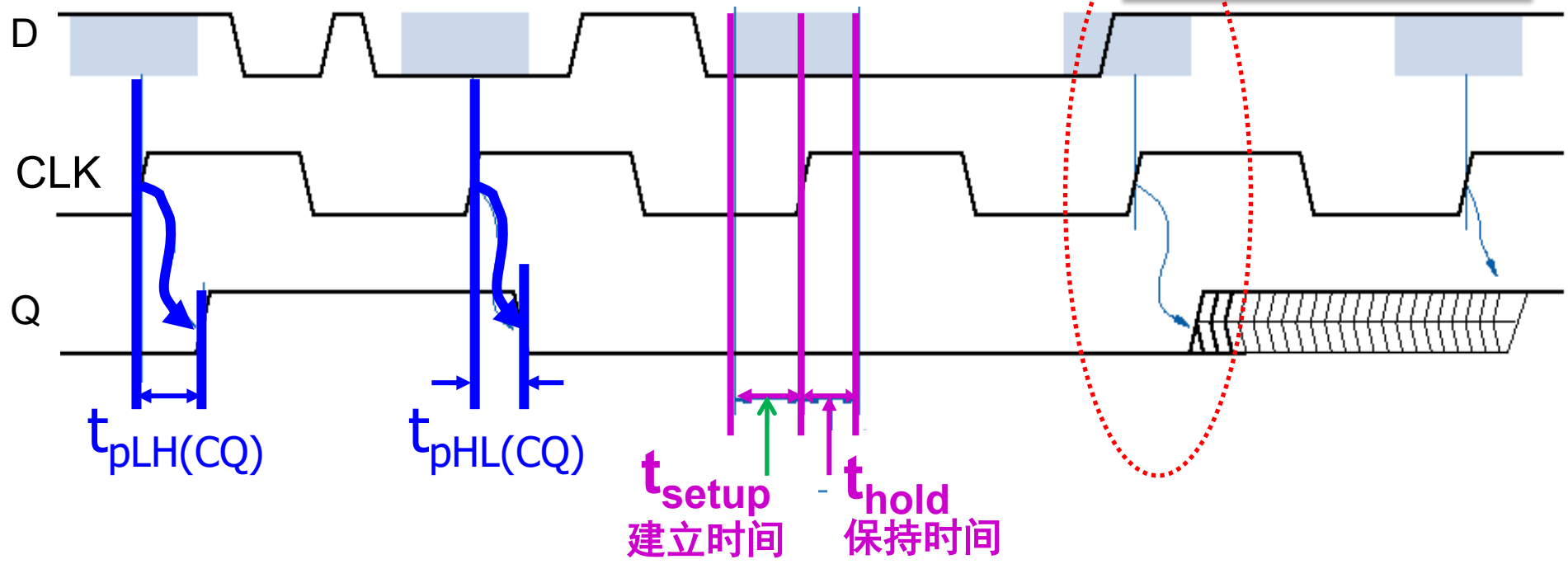
D触发器符号



- D触发器次态（特征）方程: $Q^* = D$

D锁存器还要强调C=1
这里不用强调时钟信号了

2.4 D触发器



- 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

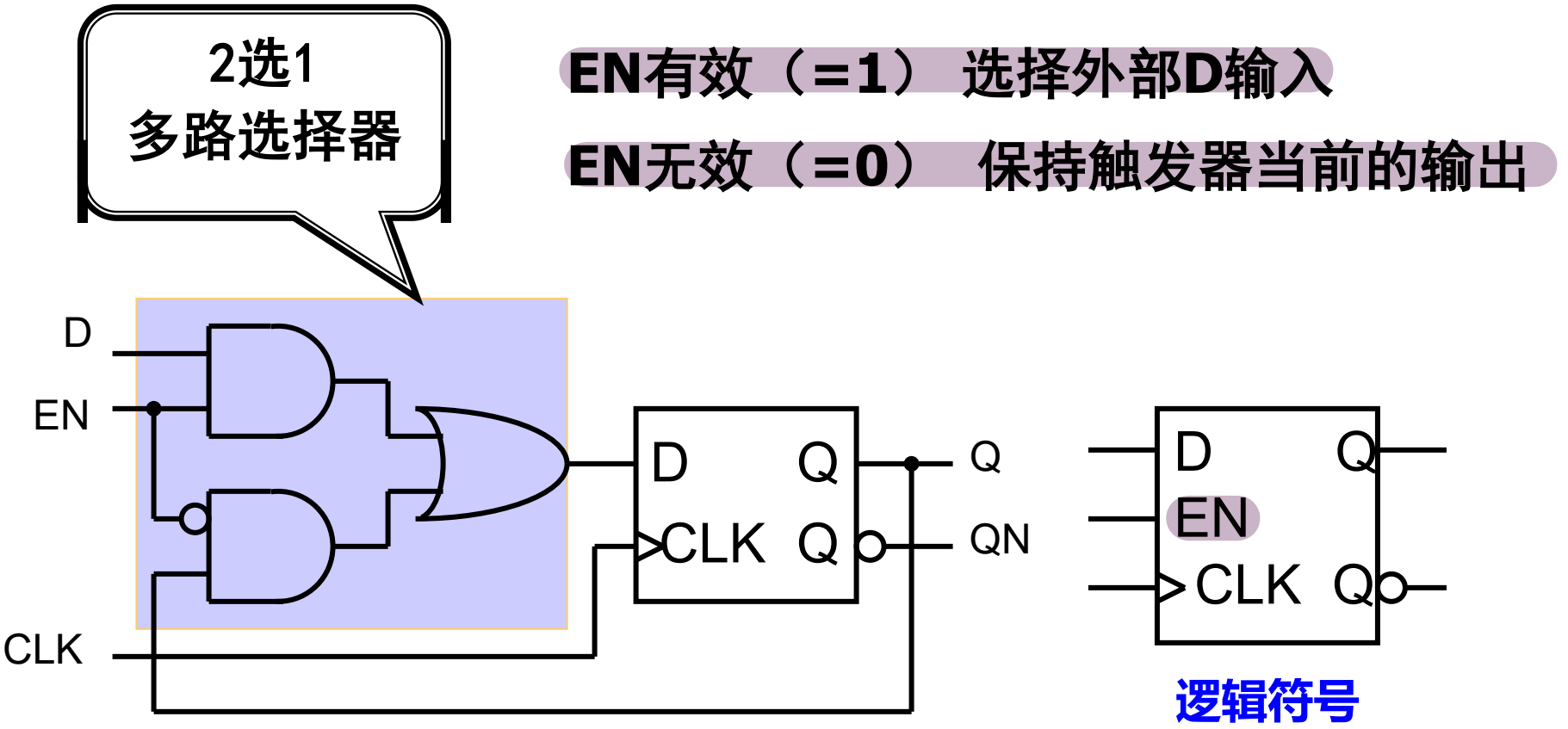
提示: 此时主锁存器是在写入的！要等它写完

- 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

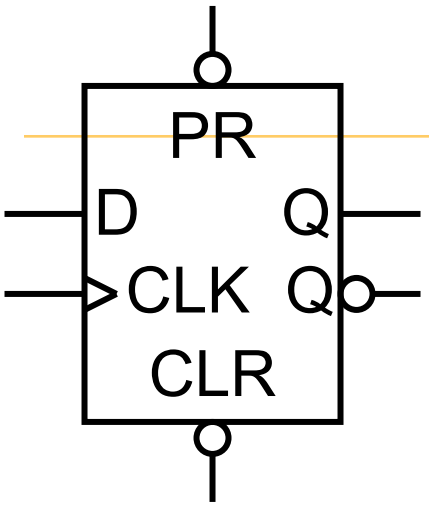
提示：要等从锁存器完成写入！避免主锁存器在上升沿过程中写入新的D值。

(*) 2.4 带使能端的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。

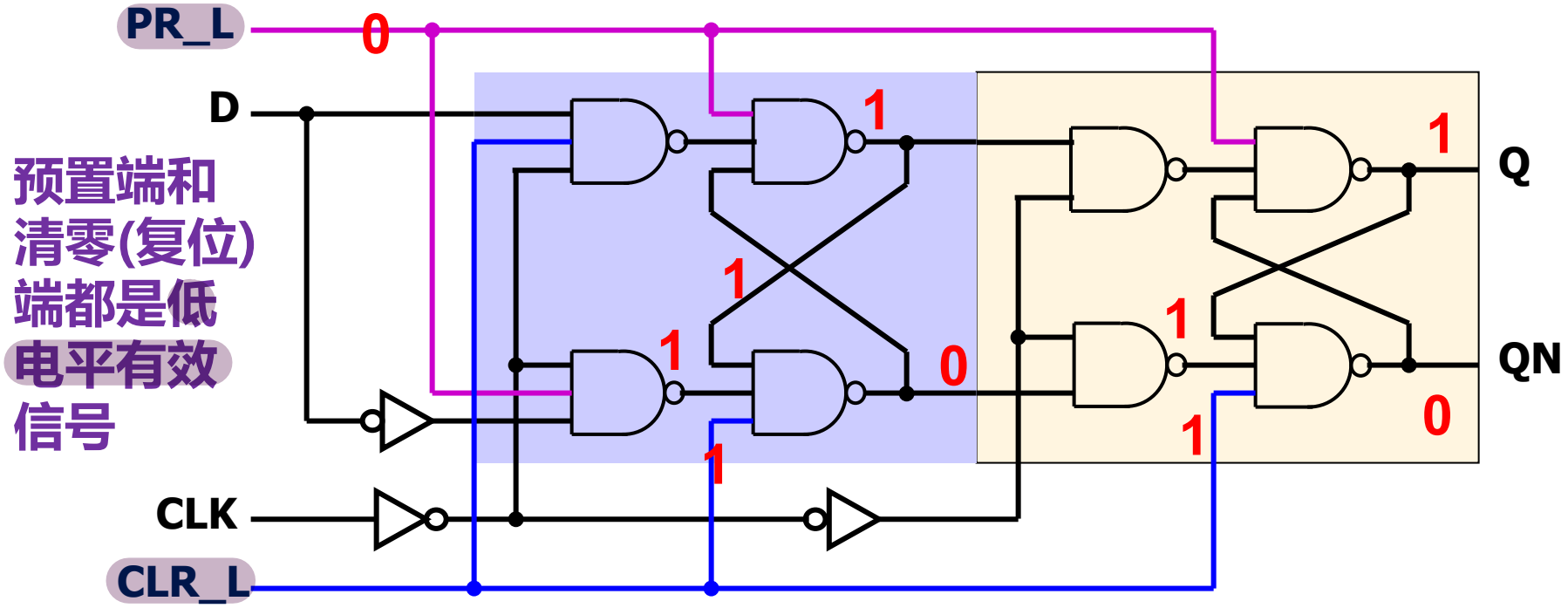


(*) 2.4具有预置和清零（复位）端的D触发器



- 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



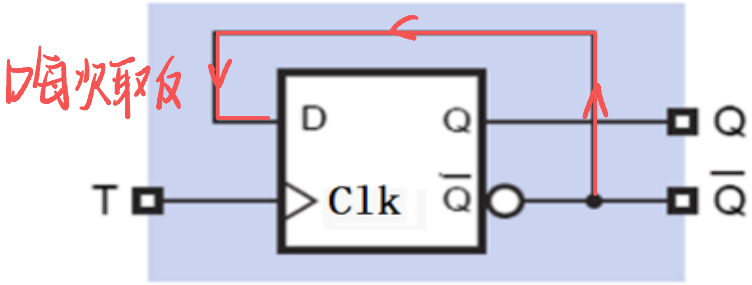
预置端和清零(复位)端都是低电平有效信号

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

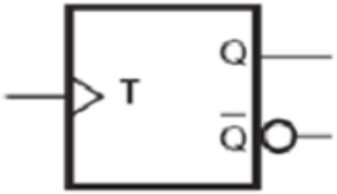
2.5 T触发器

- **T触发器：在每个时钟脉冲T的触发边沿都会改变状态**

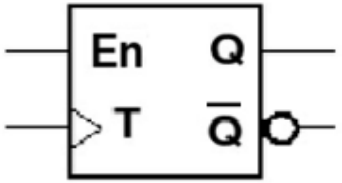
  基于D触发器实现；可用于实现计数器、分频器等功能



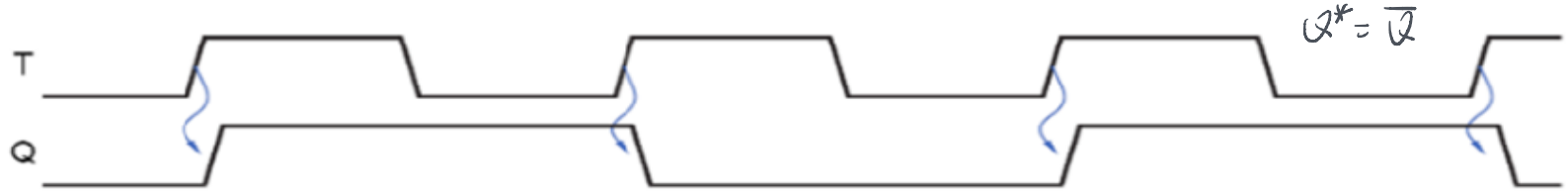
a) T触发器原理图



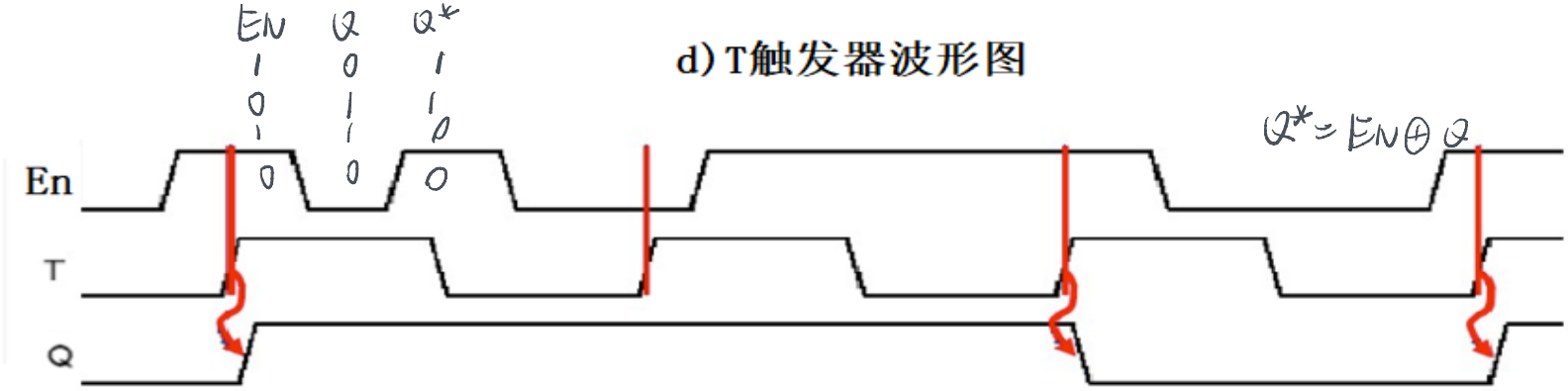
b) T触发器电路符号



C) 带使能端T触发器电路符号



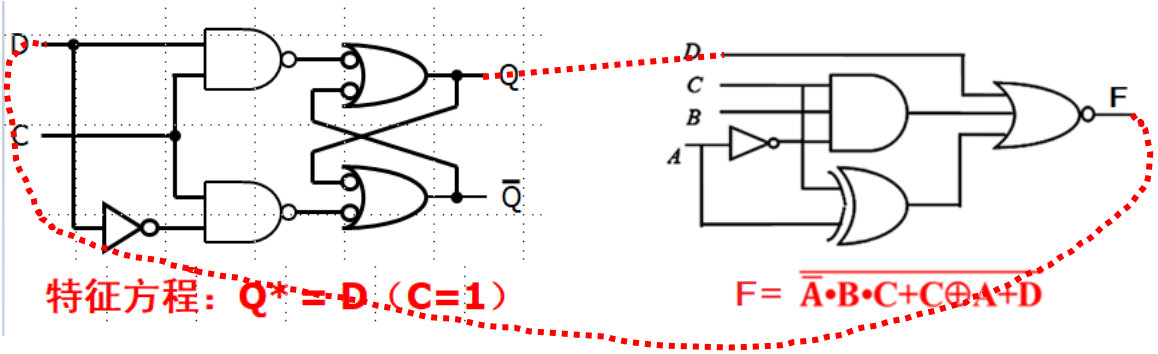
d) T触发器波形图



e) 带使能端T触发器波形图

补充&预告

- 最简单的时序逻辑电路：存储元件（锁存器，触发器）
  - 由很简单的少量逻辑门构成，交叉耦合，实现双稳态（虽然下次状态和当前状态不直接相关，但仍然算是一种特殊的时序逻辑电路）
  - 加电后，电路一直工作，在激励信号符合要求（可变可不变）的情况下，输出一直不变（所以Q的值一直稳定输出，此即为【存储】）
  - 电路中每条线路的数据会在什么时候稳定，此即为【时序分析】



  - 组合逻辑电路中，输入不变，输出才不变，按表达式计算，非存储

- 典型的时序逻辑电路（例如寄存器）： 0或若干逻辑门+若干存储元件
- 复杂的时序逻辑电路（例如CPU）： 复杂的组合逻辑+很多存储元件

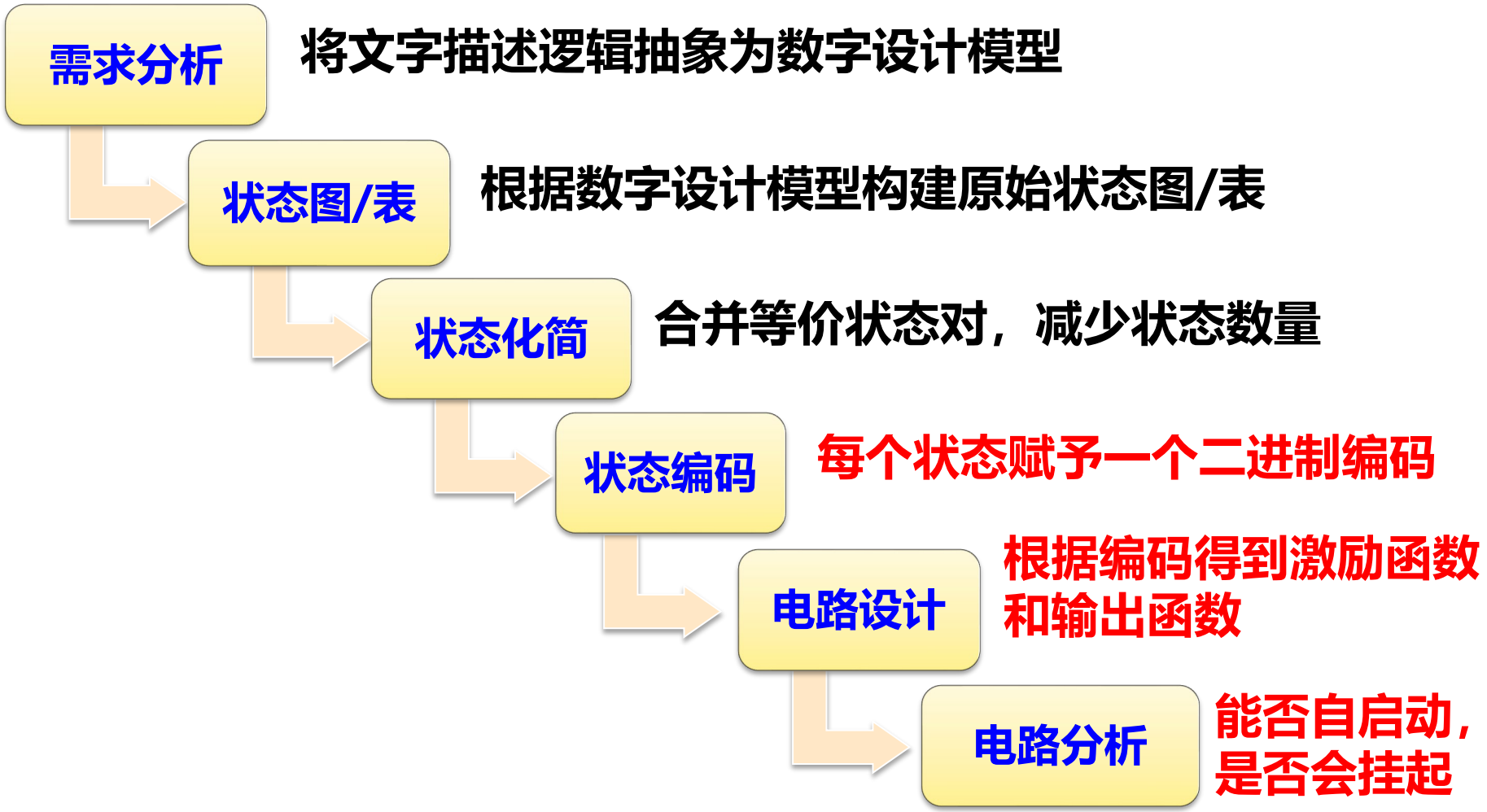
简单小结

- SR锁存器
  - ✓ 置位端(S)/复位端(R)；用于设置标志信息
- D锁存器
  - ✓ 控制端C有效时，锁存数据D
- D触发器
  - ✓ 时钟触发边沿开始后，经过Clk-Q时间，Q变成D；输入端D在时钟触发边沿到来前，须稳定Setup时间；之后须继续保持hold时间
  - ✓ 可带EN控制端、置位/清零控制端
- T触发器
  - ✓ 由D触发器构成，T连接Clk，D连接$\overline{Q}$，每个时钟发生状态变化

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

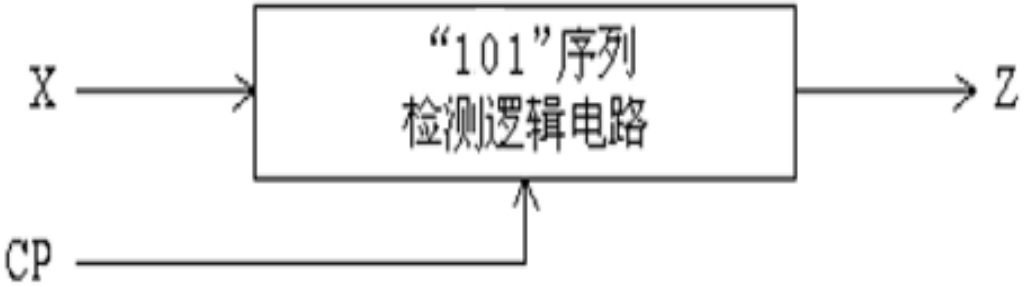
3.1 同步时序逻辑设计步骤



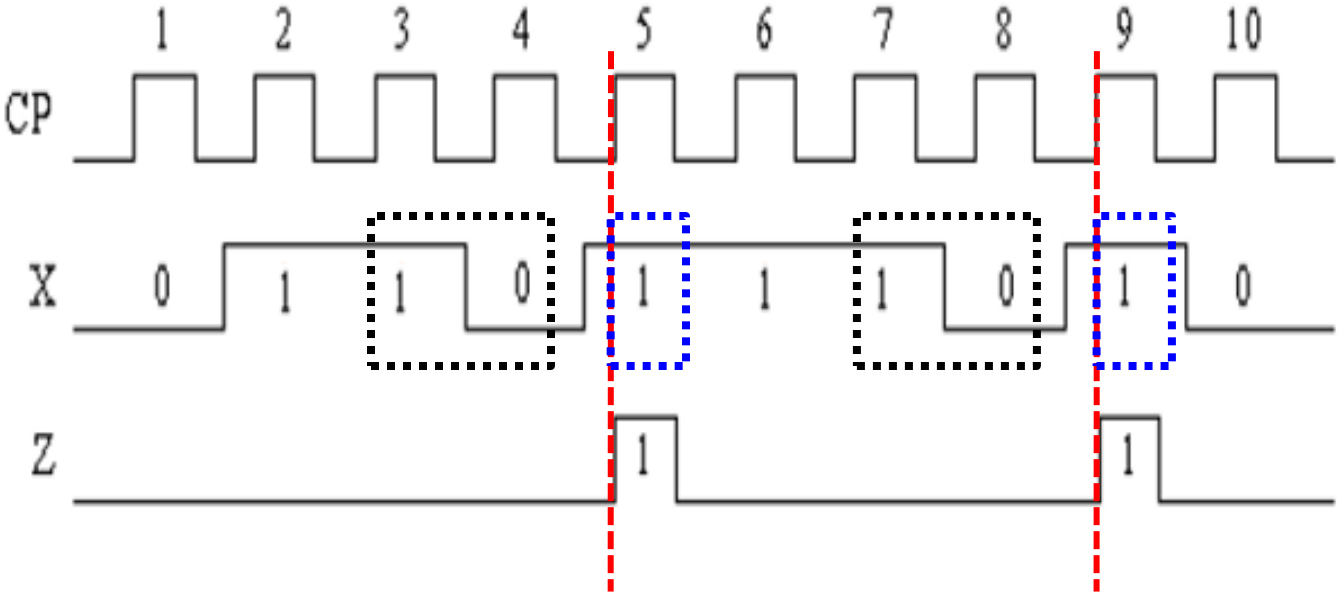
3.2 需求分析

例：检测一连串0/1输入序列中是否出现“101”

1位输入端X；
1位输出端Z。



CP脉冲到来时，根据当前X的值，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：分析系统内部的状态转换关系

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；

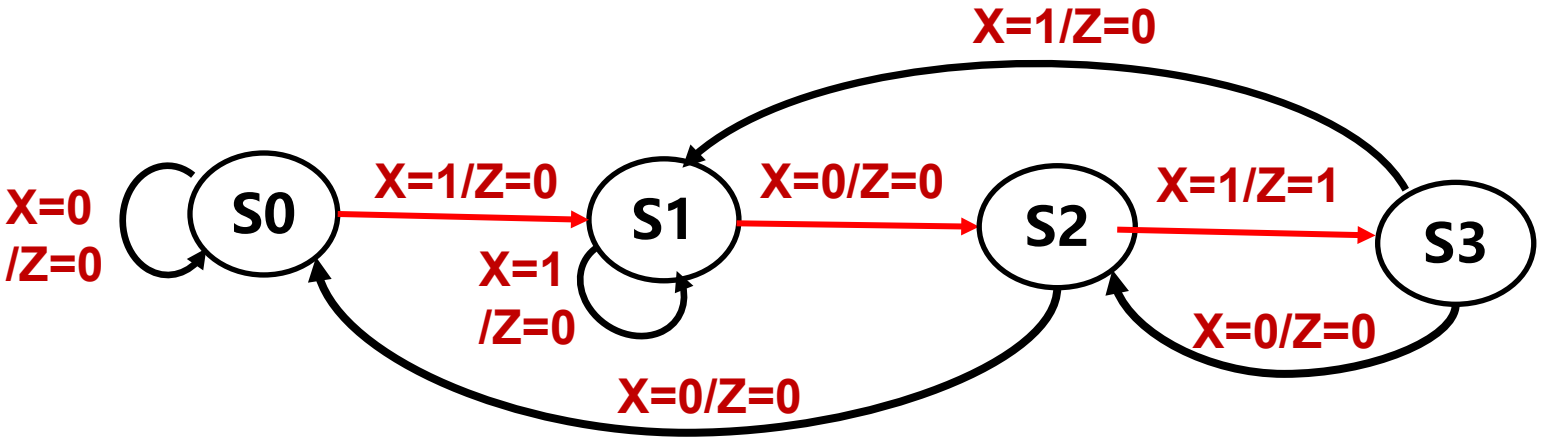
IV. 重复第II、III两步，直到不产生新状态为止。

S0：初始状态，等待接收输入

S1：接收到“101”序列中第一个1

S2：接收到该序列中的10

S3：接收到一个“101”序列



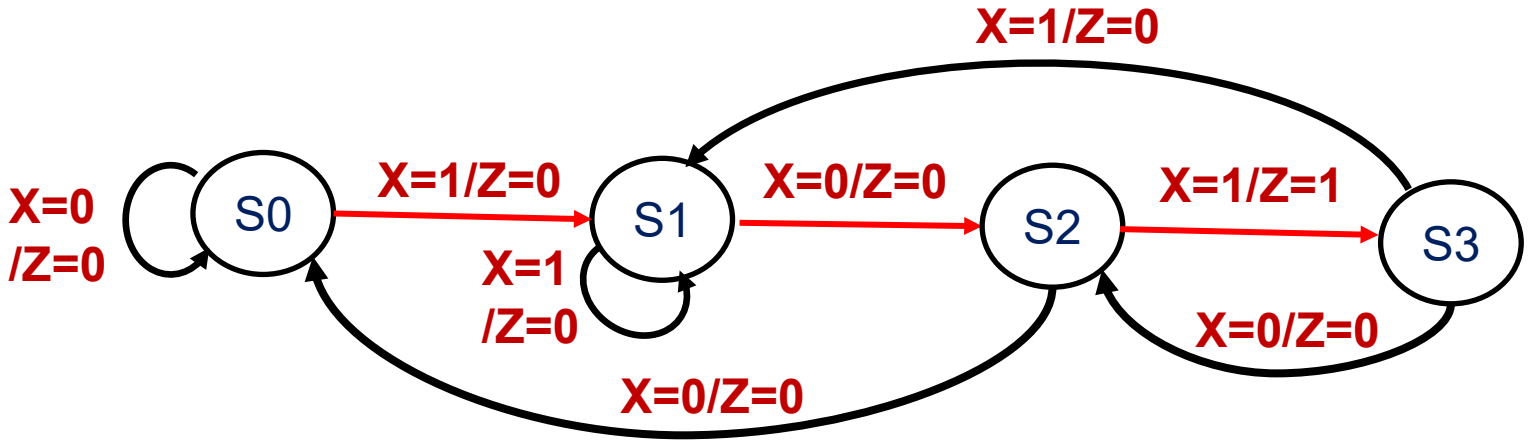
3.2 状态图/状态表设计

S0：初始状态，等待接收输入

S1：接收到“101”序列中第一个1

S2：接收到该序列中的10

S3：接收到一个“101”序列



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

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

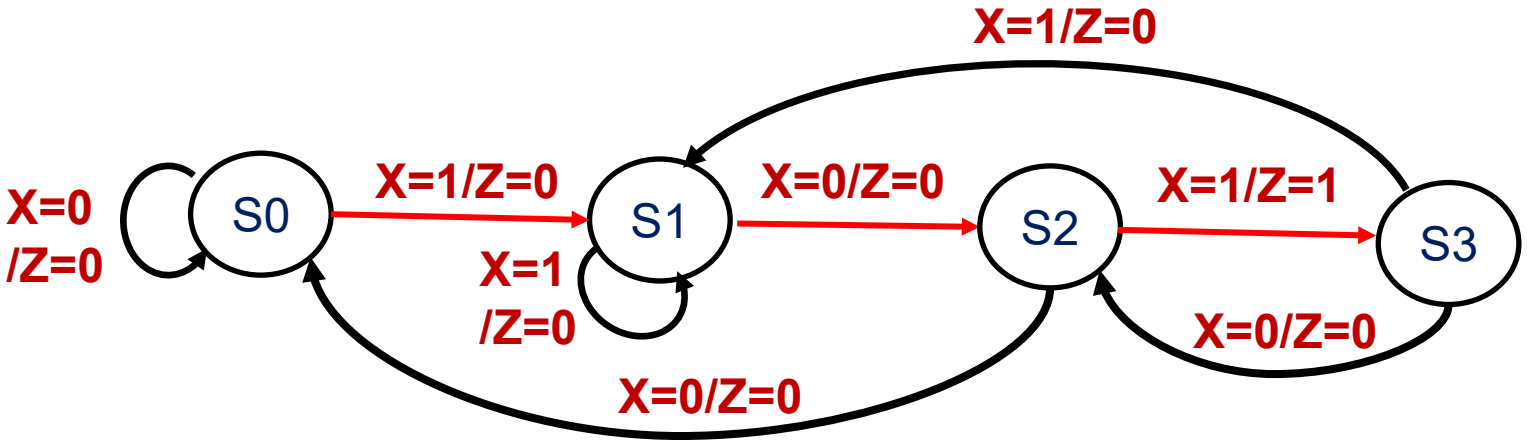
3.2 状态图/状态表设计

- 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，如本例中，转移条件分别是X=0和X=1，互斥。

  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）。如X=0和X=1，0+1=1。

- 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$\overline{X}$分别表示输入X=1和X=0。



3.3a 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

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

- 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。

3.3b 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
- 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较优编码方案：

  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，则编码方案为：S0:00，S1:01，S2:11

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

3.4 电路设计（次态函数，输出函数）

- 在选定的状态编码方案基础上进行电路设计

若编码方案S0:00，S1:01，S2:11

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

生成状态转移表（下表是简化的）

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

- 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：

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

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

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

$$Z = Y1 \cdot Y0 \cdot X$$

3.4 电路设计（函数化简）

Y1Y0	Y1*Y0*/Z	
	X=0	X=1
00	00/0	01/0
01	11/0	01/0
11	00/0	01/1
10	00/0	01/1

采用代数法化简

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

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

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

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

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

将无关项编码Y1Y0=10引入化简，则

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

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

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

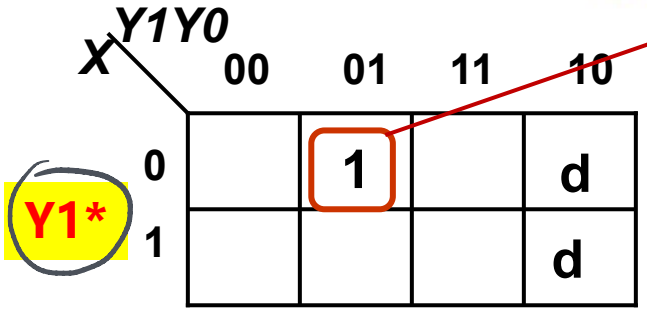
3.4 电路设计（函数化简）

利用卡诺图化简

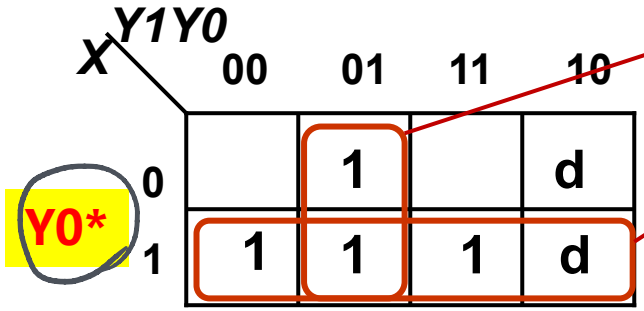
状态转移表

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

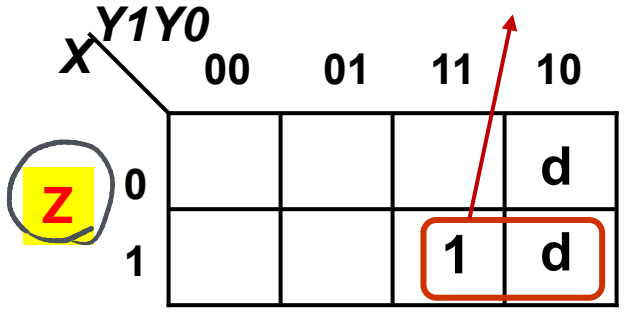
$Y1^* = \overline{Y1} \cdot Y0 \cdot \overline{X}$



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



$Z = Y1 \cdot X$



3.4 电路设计（确定元件，画电路图）

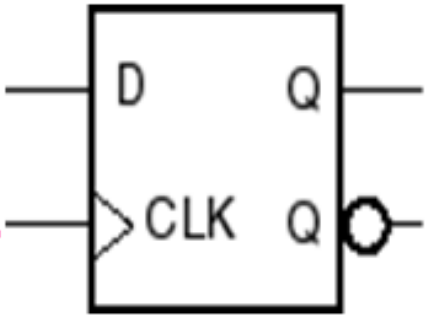
- 根据次态函数和选择的状态记忆单元（触发器），推导出激励函数
  - 假设采用D触发器，其特征方程Q* = D，则：

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

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

- 输出函数为：$Z=Y1\cdot X$

要基于Q和激励函数，计算出新状态，传给D



Q一直稳定输出，代表的是当前状态

要基于Q和输入X，用输出函数计算出输出Z

需要几个D触发器呢?

3.4 电路设计（确定元件，画电路图）

- 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：

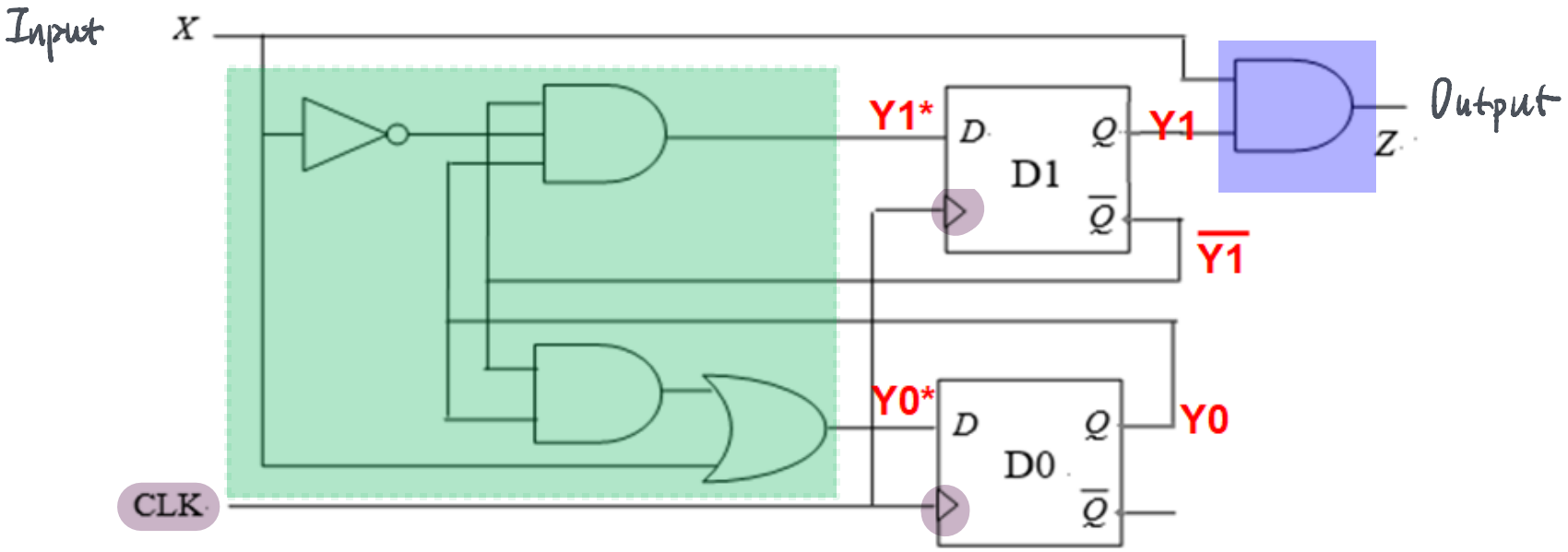
$D1=Y1^*=\overline{Y1}\cdot Y0\cdot\overline{X}$

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

- 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

(1) 触发器存储着旧状态Y0，Y1
(2) 结合旧状态和新输入X，激励模块和输出模块计算出新状态和输出结果
(3) 新状态存入触发器，回到 (1)

clock → Q



3.4 电路设计后的分析（未用状态分析）

- 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；若不能或有错误输出，则需调整电路设计

分析要点：使用未用状态化简后，能否“自启动”而不会发生“挂起”

如果初始是未用状态，经过几个时钟能够进入工作状态，但进入的状态是错的（比如才输入一个1，就进入了连续输入两个1才应该进入的状态），那么就会导致后续输出错误。一般也会归入不能自启动的情况。

3.4 电路分析（未用状态分析）

Y1Y0	Y1*Y0*/Z X=0	X=1
00	00/0	01/0
01	11/0	01/0
11	00/0	01/1
10	00/0	01/1

◆未用状态分析举例

- 对于前述例子，利用未用状态10作为无关项化简后，得到：

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

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

$Z=Y1\cdot X$

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

最终得到的函数

- 当处于未用状态Y1Y0=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　错误输出

- 分析是否具有“自启动”能力

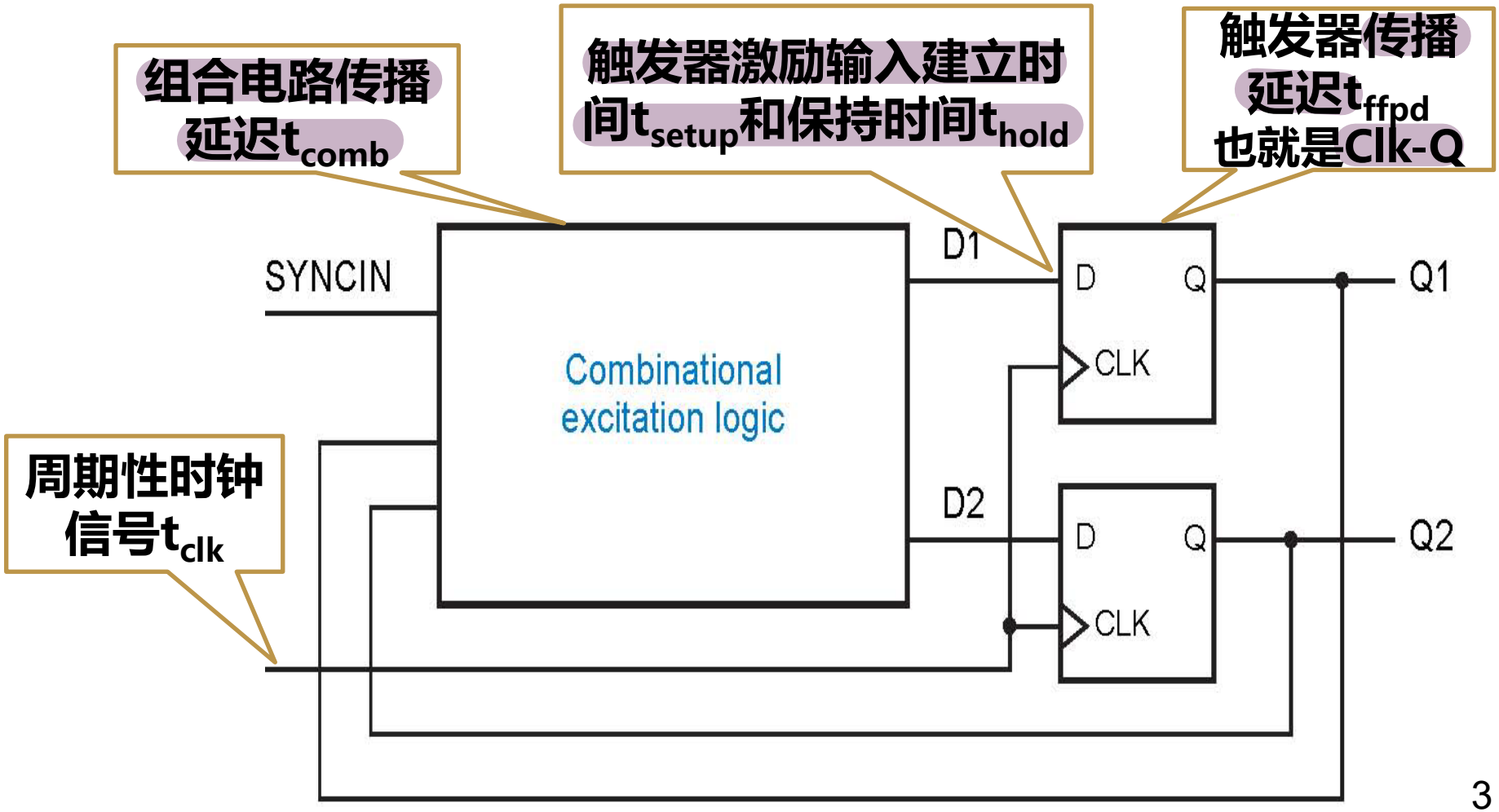
经过1个时钟周期就能进入正常工作状态，但是，当输入X=1时，输出Z=1，是错误输出，需调整输出模块的设计

- 重新设计逻辑电路中的输出模块（调整为化简前的表达式）

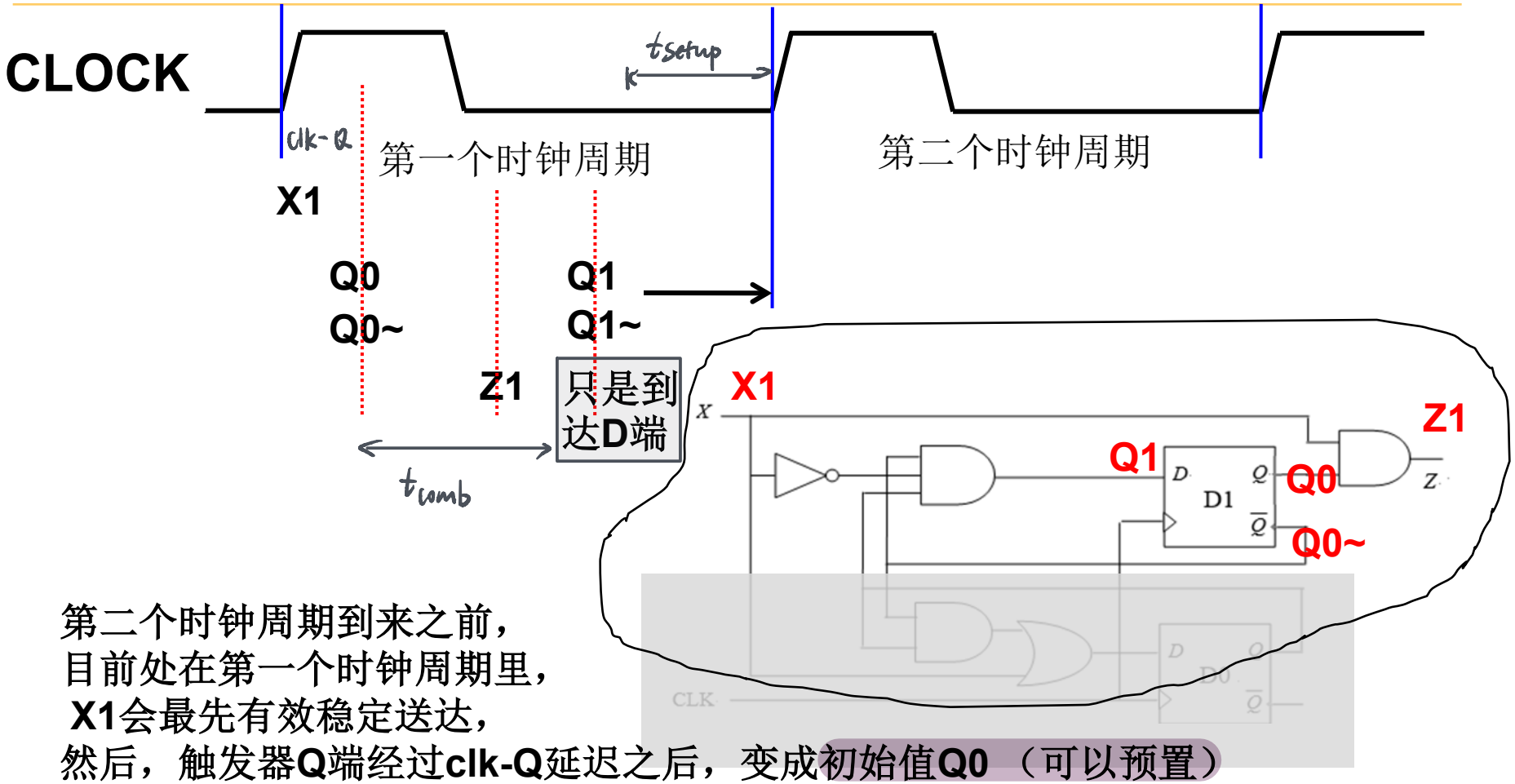
Z=Y1·Y0·X 在未用状态10时，若输入X=1时，则输出Z=0。此时，不会发生误输出

3.4 电路分析（定时分析）

- 时序逻辑电路定时分析
  - 电路的工作频率与组合逻辑电路传输延迟、触发器建立和保持时间、触发器传输延迟等密切相关。

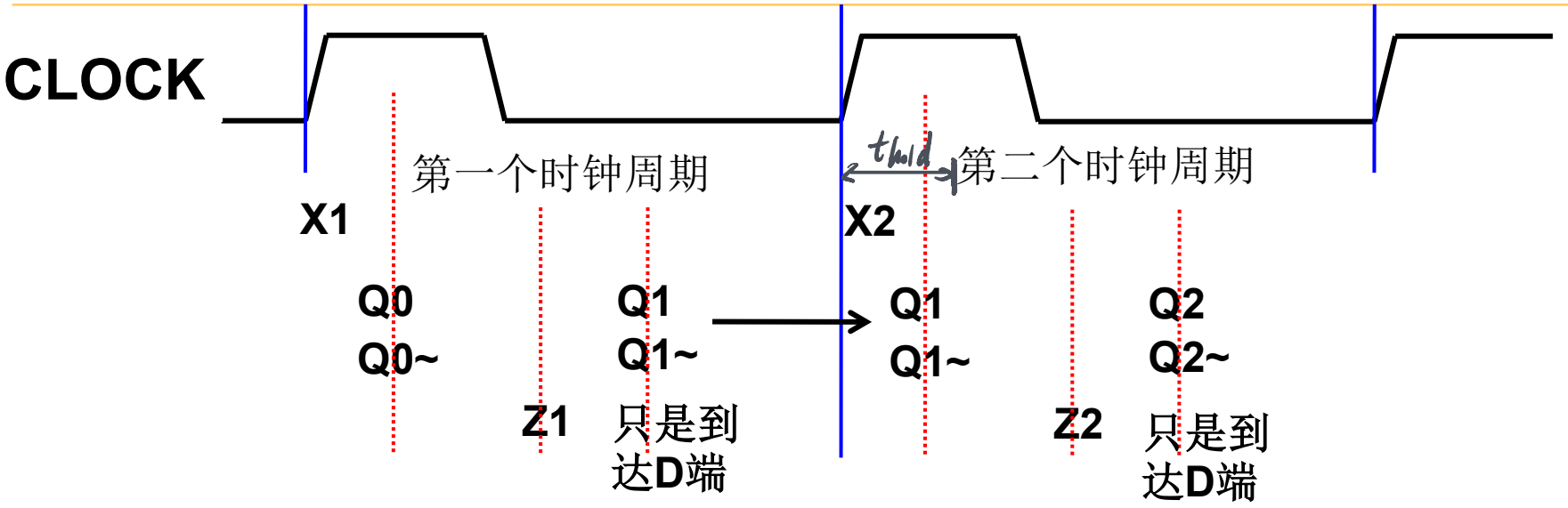


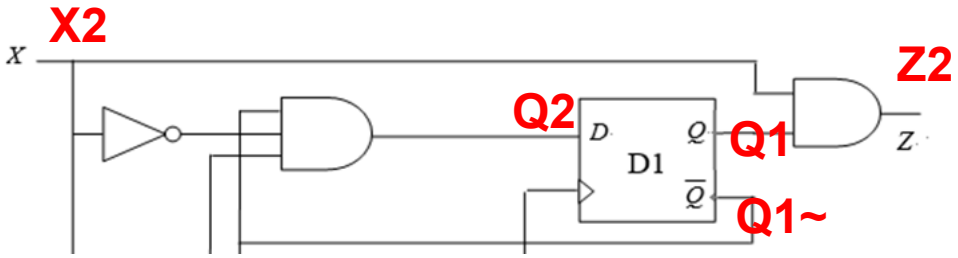
3.4 电路分析（定时分析）



第二个时钟周期到来之前，
目前处在第一个时钟周期里，
X1会最先有效稳定送达，
然后，触发器Q端经过clk-Q延迟之后，变成初始值Q0（可以预置）
——过了次态运算逻辑延迟之后，基于Q0和X1，算出了Q1，并到达触发器D端
（第二个时钟周期到来之前，这个D端必须保持稳定超过setup时间）
——过了输出运算延迟之后，基于Q0和X1完成了Z1的计算，且该结果不影响其它过程，所以这个延迟只要小于时钟周期就行了，与次态计算无关

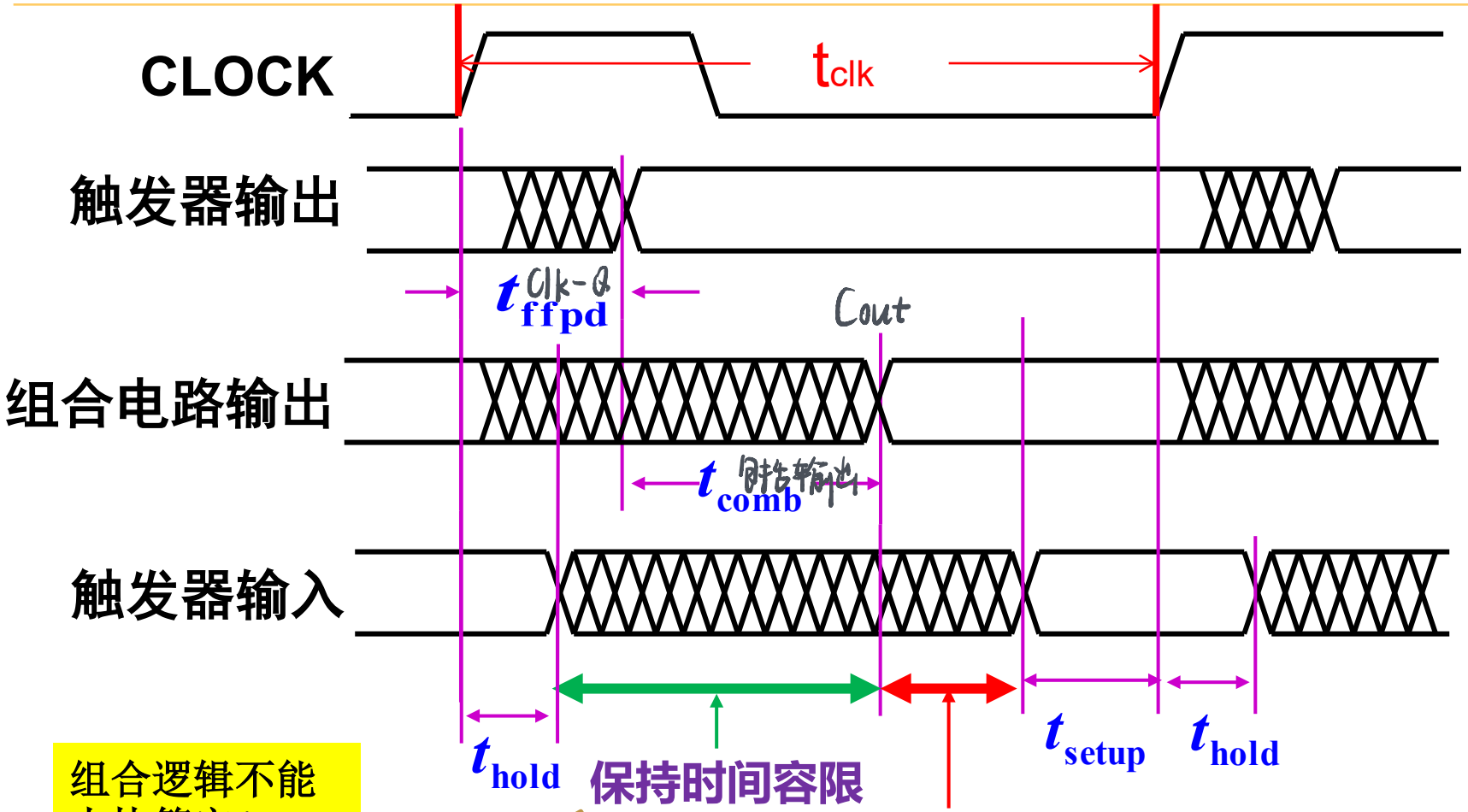
3.4 电路分析（定时分析）





第二个时钟周期到来之后，
X2会最先有效稳定送达，
算出的Q1保持不变，稳定在触发器的D端（hold时间）
然后，触发器Q端经过clk-Q延迟之后，变成Q1
——过了次态运算逻辑延迟之后，基于Q1和X2，算出了Q2，并到达触发器D端
（第3个时钟周期到来之前，这个D端必须保持稳定超过setup时间）
——过了输出运算延迟之后，基于Q1和X2完成了Z2的计算

3.4 电路分析（定时分析）



组合逻辑不能太快算完！

$$t_{ffpd(min)}+t_{comb(min)}-t_{hold}$$

不能来不及建立！

$$t_{clk}-t_{ffpd(max)}-t_{comb(max)}-t_{setup}$$

时间容限指为保证电路正常工作某信号定时所允许的时间范围，都大于0

3.4 电路分析（定时分析）

- 建立时间容限=$t_{clk}-t_{ffpd(max)}-t_{comb(max)}-t_{setup}$，$>0$
- 保持时间容限=$t_{ffpd(min)}+t_{comb(min)}-t_{hold}$，$>0$

因此，得到**时序约束关系**：

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

时钟周期不能小于这个值，
但也不需要大过很多

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

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

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_{hold}内能保持稳定不变。这就要求次态信号不能反馈太快。

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

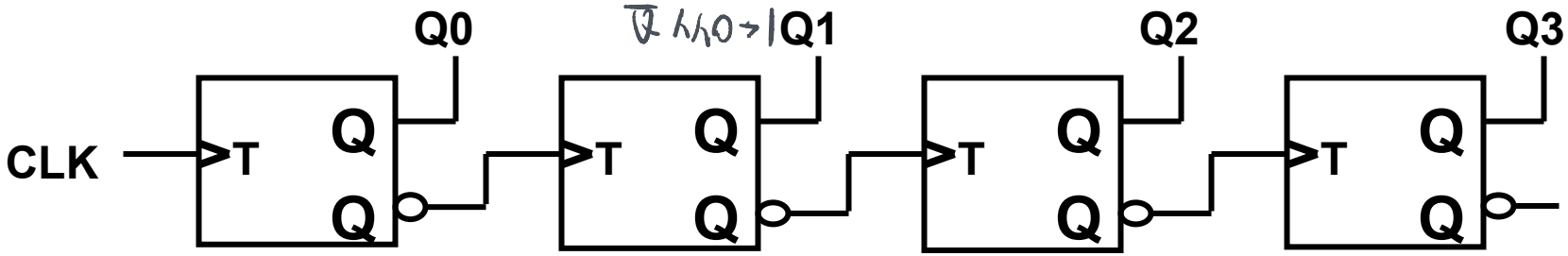
4.1 计数器

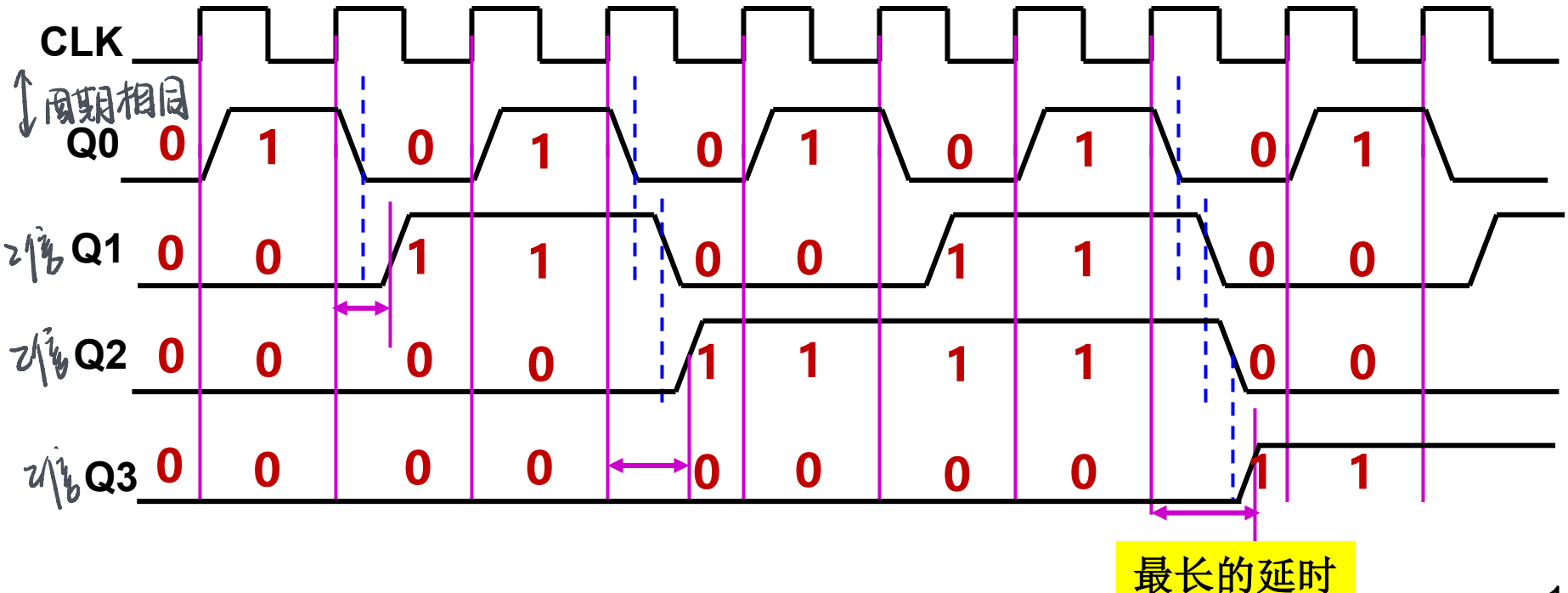
- 计数器是一种对外部信号进行总数统计的时序逻辑元件
- 一般从0开始计数，在达到最大计数值时输出一次计数完成信号，并重新开始计数
- 最大计数值为计数器的模
- 计数器的分类
  - 按时钟定时的使用方式：同步、异步
  - 按计数方式：加法、减法、可逆
  - 按编码方式：二进制、十进制BCD码、循环码
  - 按进位方式：行波（串行）进位、并行进位

4.1 计数器——异步行波加法计数器

- 用T 触发器实现（上升沿触发），激励输入串行传递，每个时钟周期传递一次。

Q_{i+1}总是在Q_i由1变0时开始改变





4.1 计数器——异步行波加法计数器

Q0每个时钟转1次，Q1每2个转1次，
Q2每4个转1次，Q3每8个转1次

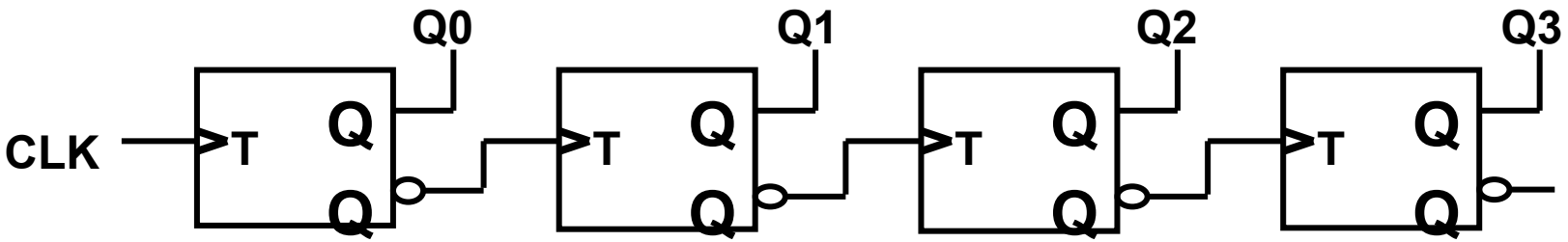
Q_i 每 2^i 个时钟转一次

0~15

- $Q_3Q_2Q_1Q_0$转换过程为（注意：最高位为Q3）

  0000→0001→0010→0011→0100→

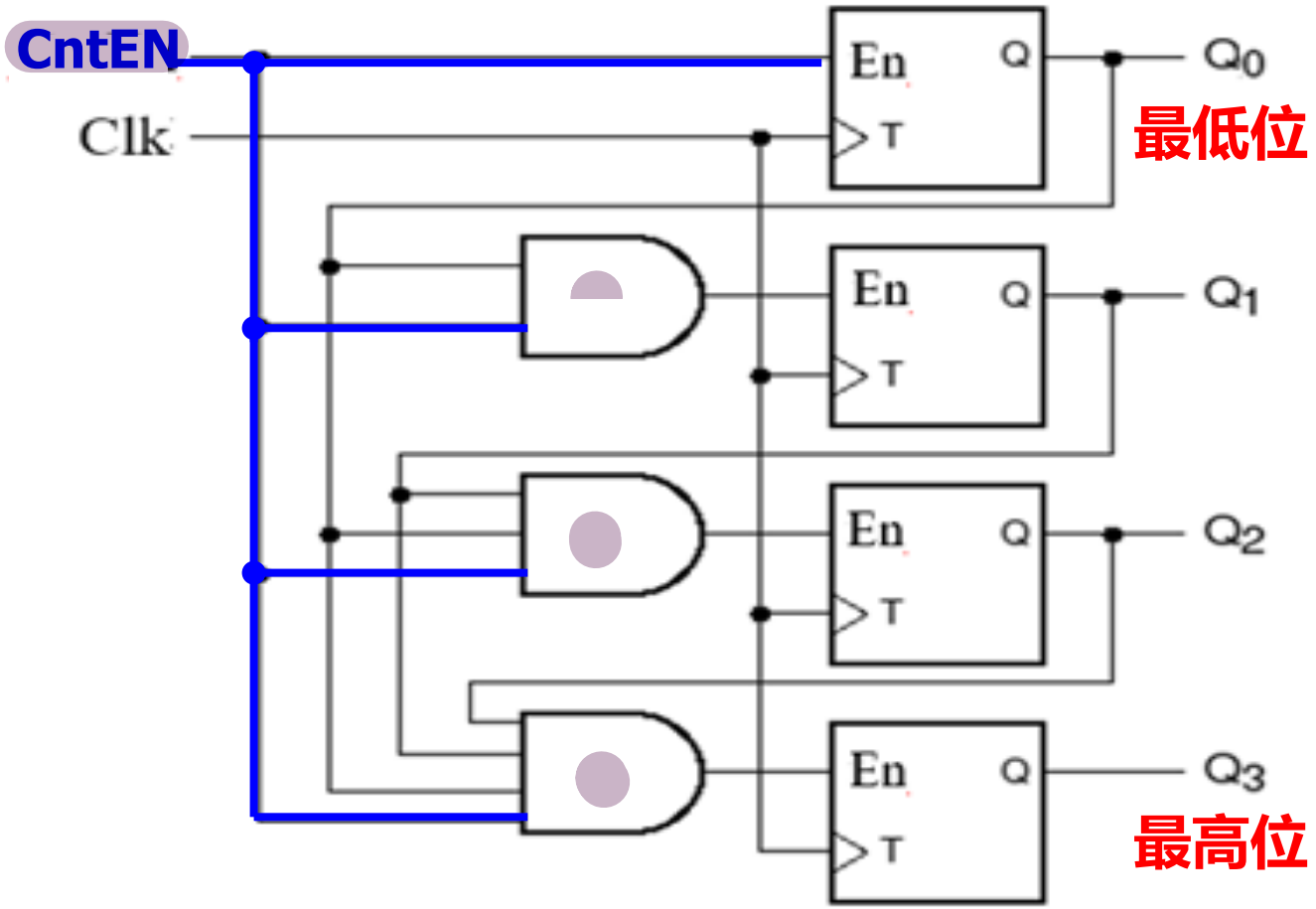
  0101→0110→ … … →1111



当编码为1111时，下个时钟到达后，经过最长的延时（4个T触发器的锁存延迟），又回到编码0000

4.1 计数器——同步并行加法计数器

- 所有触发器共用同一个时钟信号，
- 在时钟信号边沿到达后，所有触发器的输出同时发生变化。
- 带使能端EN的T触发器，上升沿触发



低位为1才会启动高位

每个Qi一定会变化吗？

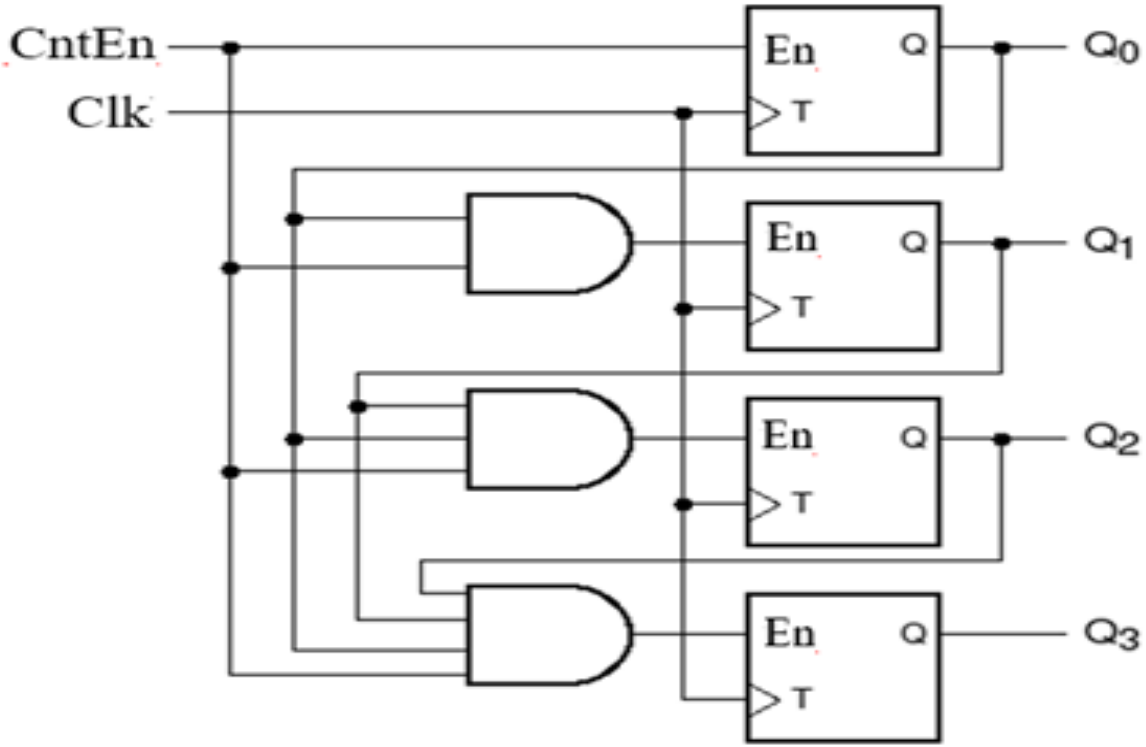
——不一定。也不应该每次都变化：用EN信号（低位Q和CntEN相与的结果）来控制

4.1 计数器——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→……→1101→1110→1111

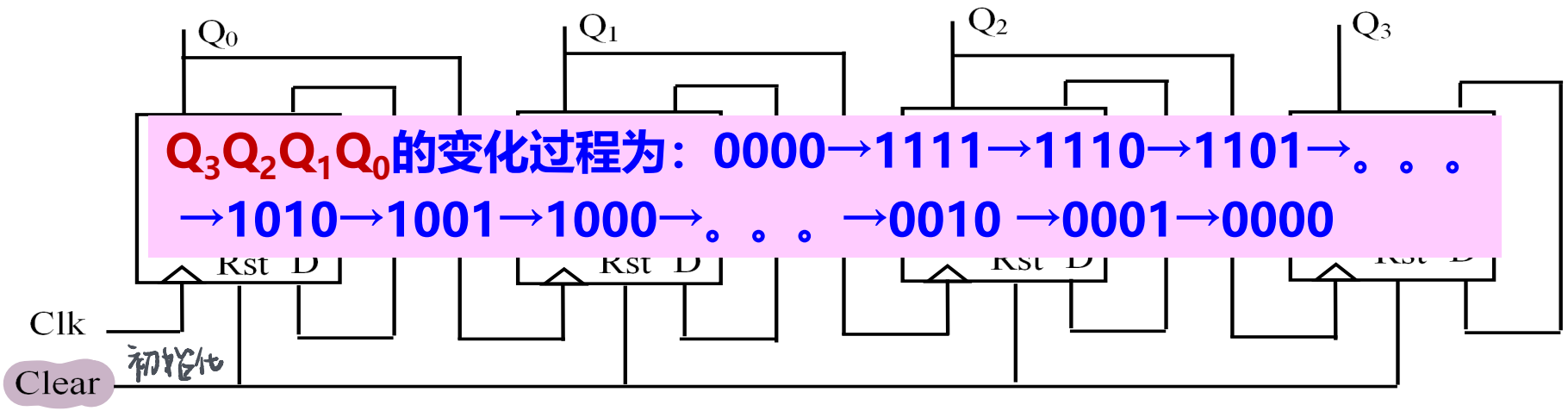
CntEN有效时，每个时钟Q_0都会发生状态改变；

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。



当编码为1111时，只要经过一个与门+1个T触发器的锁存延时，就可回到编码0000，比行波（串行）加法计数器快

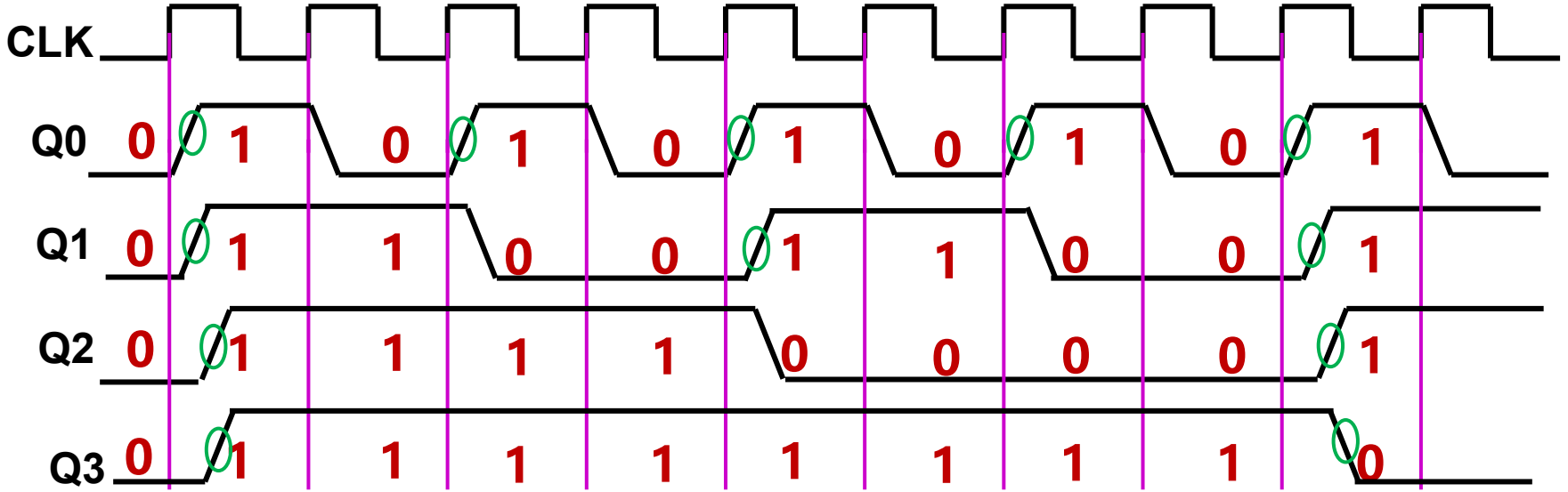
4.1 计数器 ◆二进制异步行波减法计数器



$Q_3Q_2Q_1Q_0$的变化过程为：0000→1111→1110→1101→。。。
→1010→1001→1000→。。。→0010 →0001→0000

Q_{i+1}总是在Q_i由0变1时 (上升沿) 发生改变

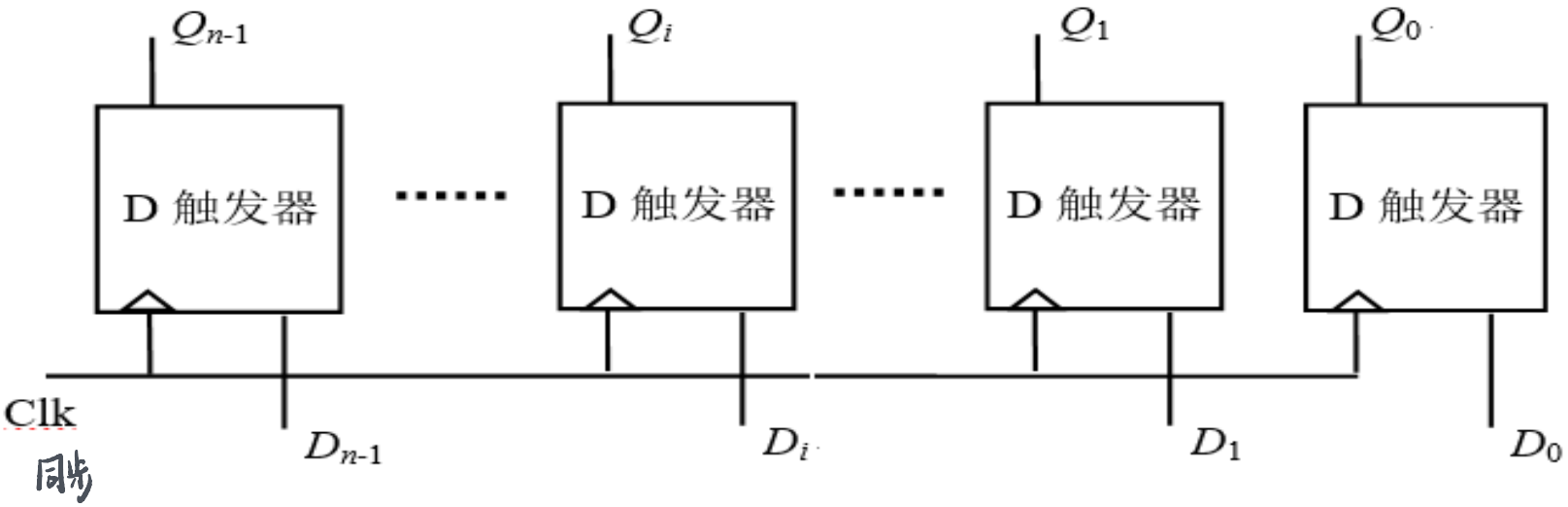
Clear清0，初态为0000

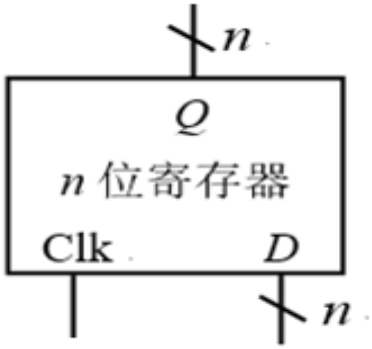


Q0每个时钟转1次，Q1每2个转1次，Q2每4个转1次，Q3每8个转1次

4.2 寄存器

- 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





每个时钟到来，存放内容都会改变吗？

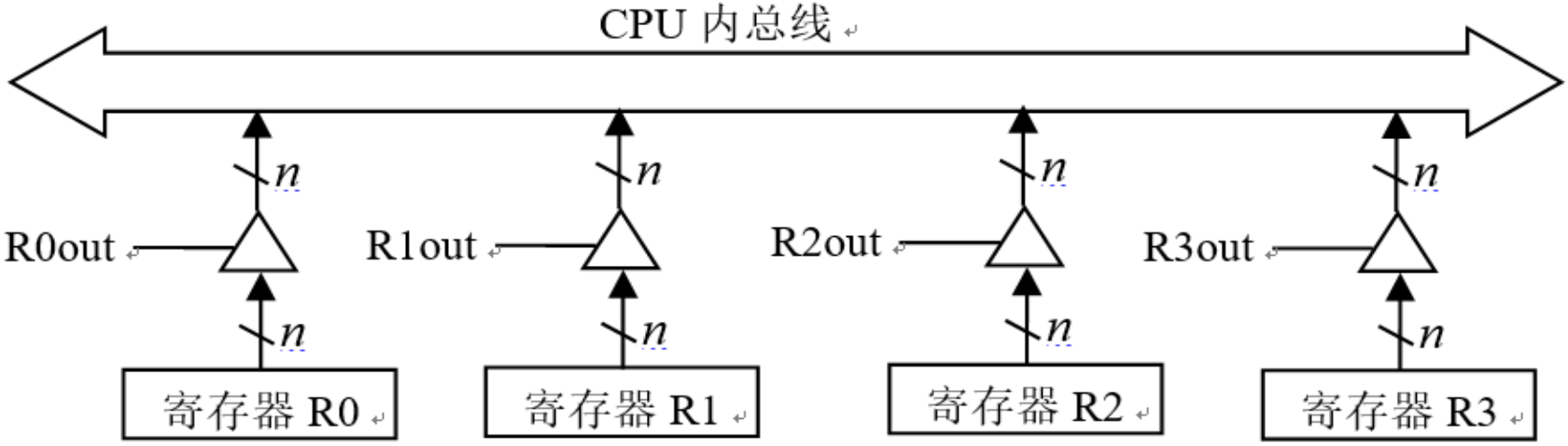
——不一定。

——可以加使能端，或者让D输入不变

4.2 寄存器

- **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



寄存器是一种时序逻辑电路，但只包含存储电路

在没有新的CLK脉冲来之前，寄存器一定能保存原有内容不变 存储

附加一些控制电路就能够实现：置零、保持（ CLK信号到达时触发器不随D端的输入信号而改变状态，保持原来的状态不变）等功能。

接收数据时所有各位信息都是同时输入的，而且触发器中的数据是并行地出现在输出端的。

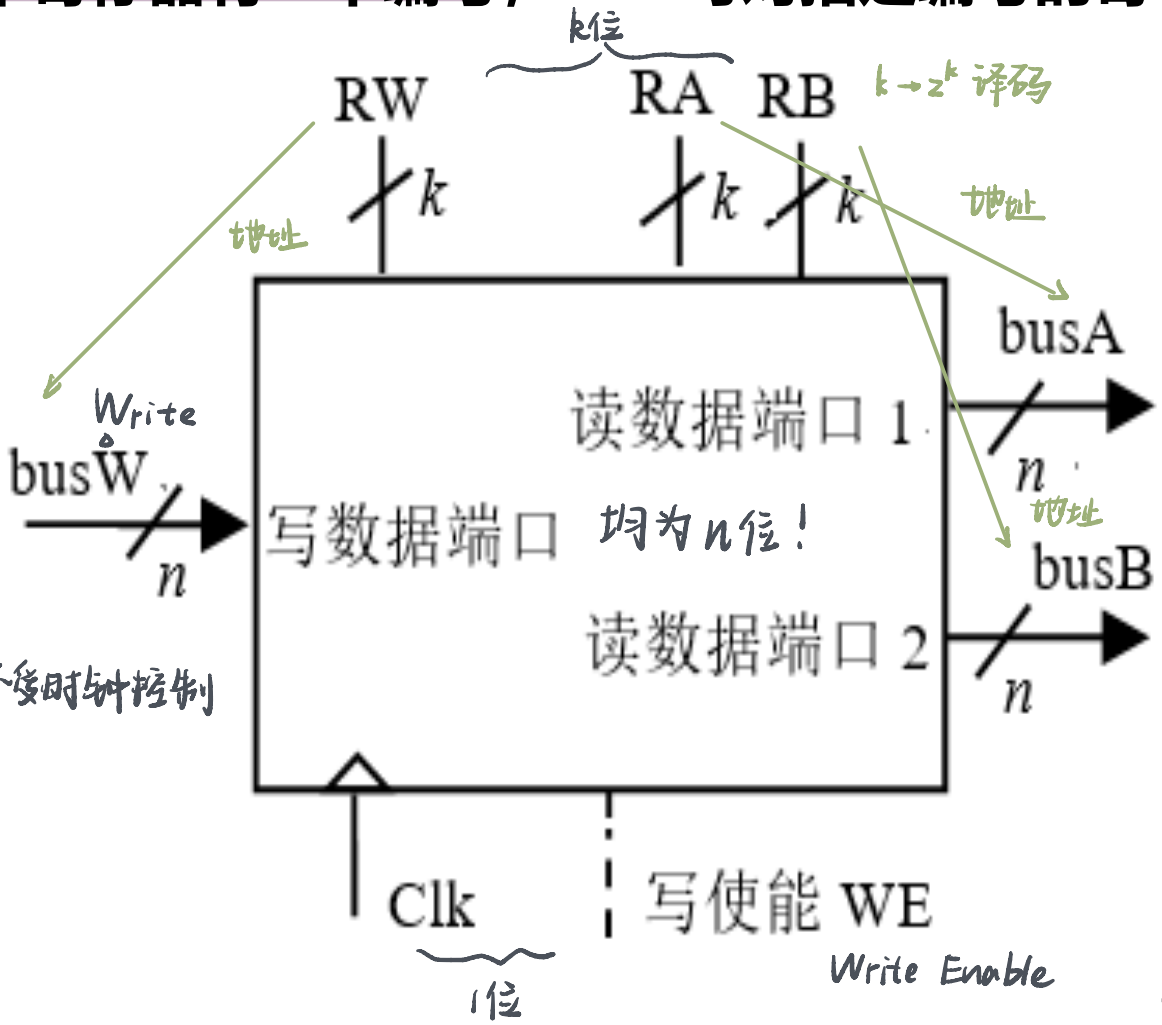
4.2 寄存器堆

- 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

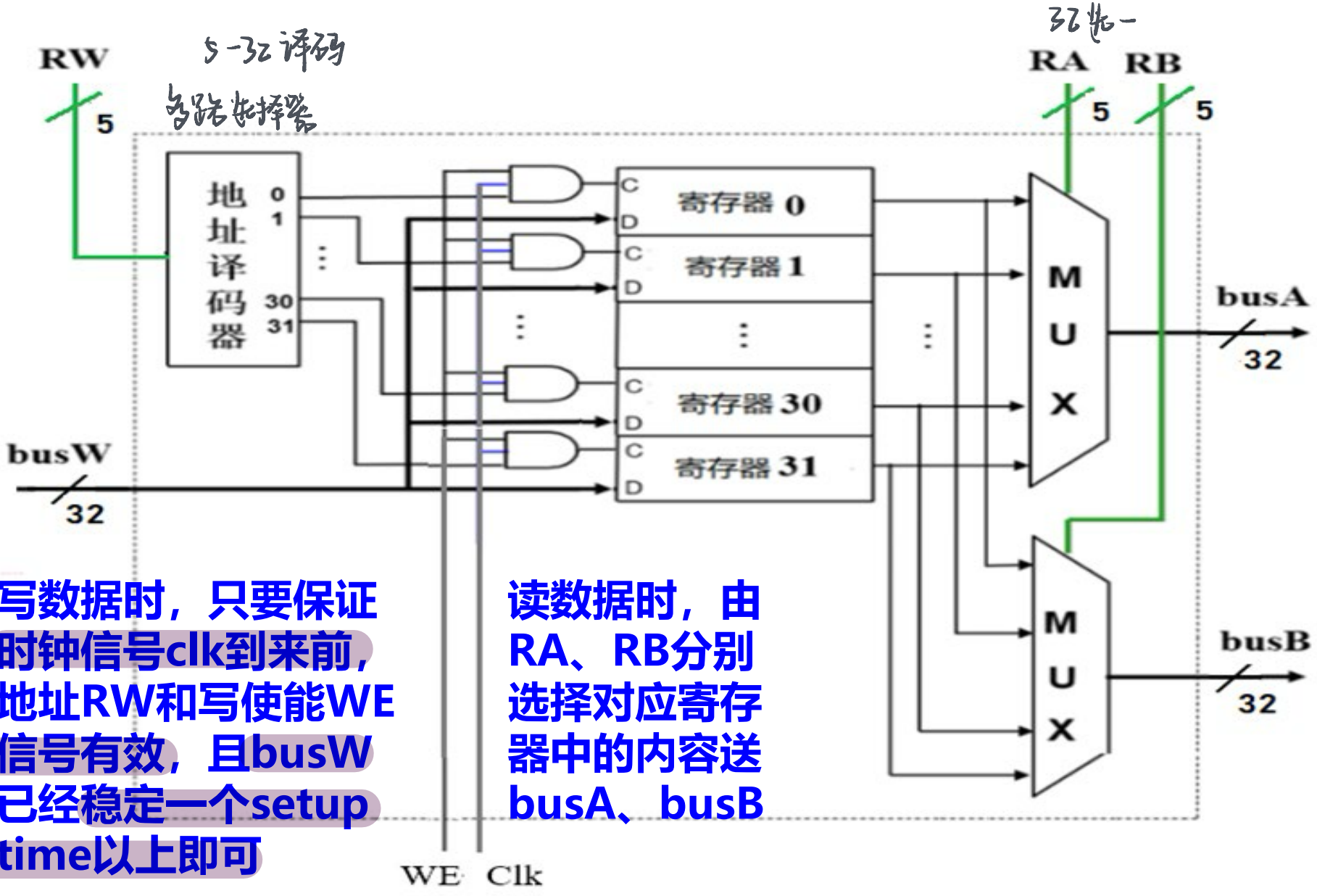
寄存器堆中共有2^k个寄存器，每个寄存器位数为n，RA和RB分别是读口1和读口2的寄存器编号，RW是写口的寄存器编号

读操作属于组合逻辑操作；不受时钟控制

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 寄存器堆内部结构



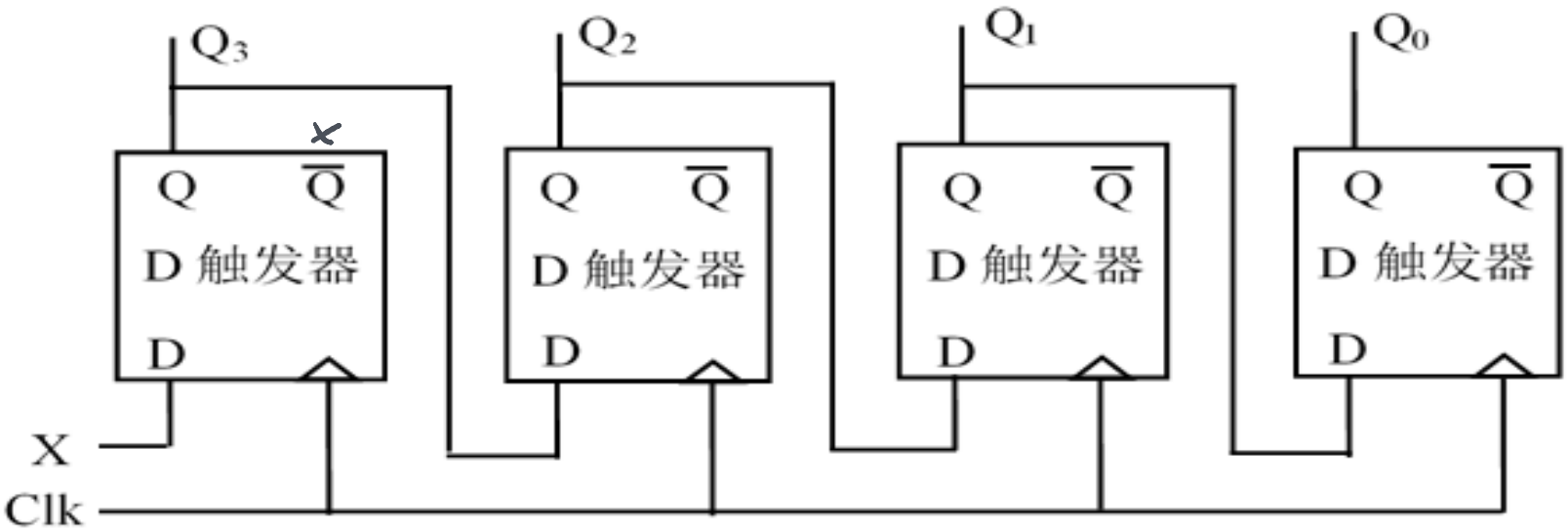
写数据时，只要保证时钟信号clk到来前，地址RW和写使能WE信号有效，且busW已经稳定一个setup time以上即可

读数据时，由RA、RB分别选择对应寄存器中的内容送busA、busB

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

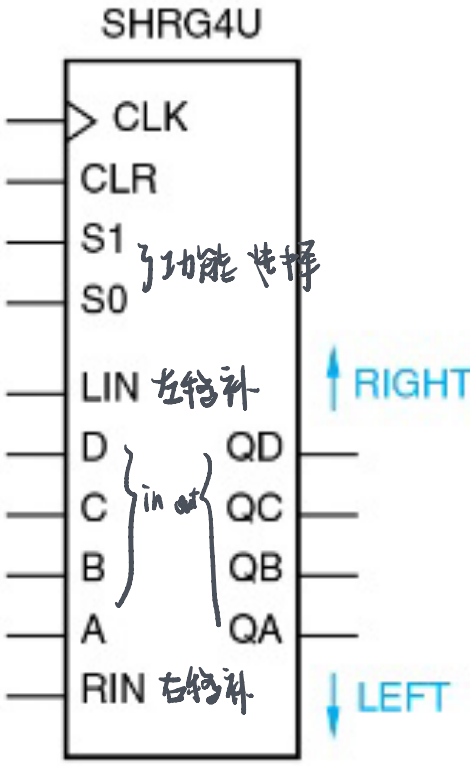


假设初始状态编码$Q_3Q_2Q_1Q_0$=0000，X输入为序列10011011

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
  - 用CLR，S1，S0这三个信号的排列组合来表示不同功能



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

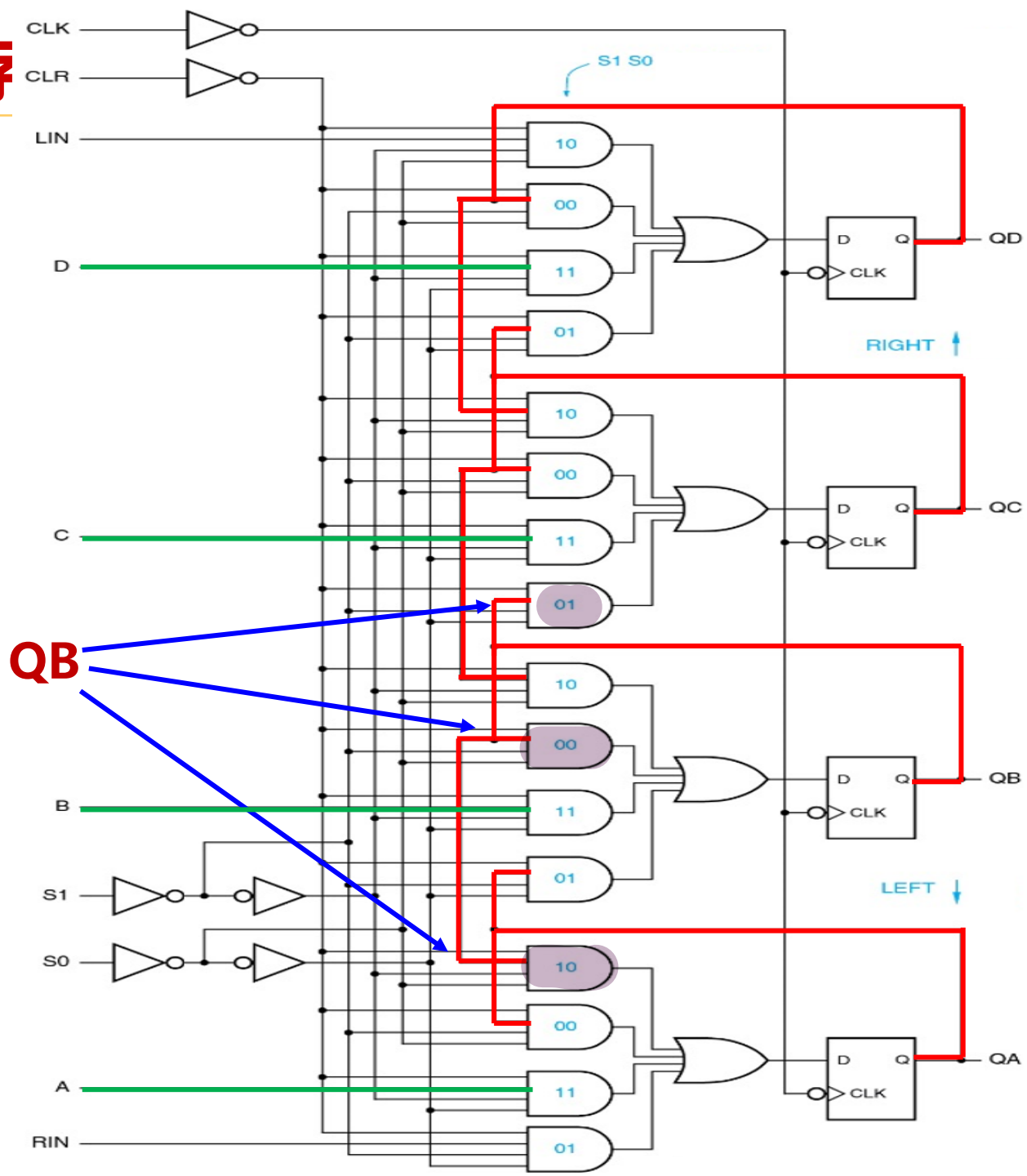
- 4位通用移位寄存器结构图

S1S0=00：保持

S1S0=01：上(右)移

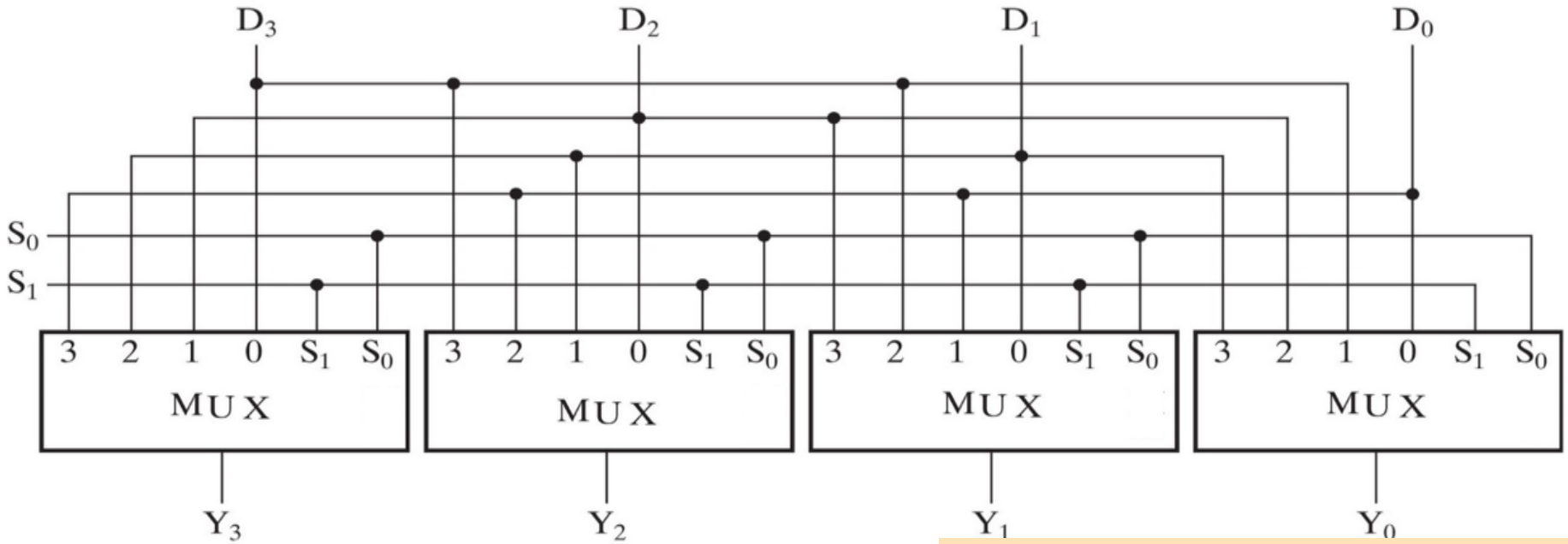
S1S0=10：下(左)移

S1S0=11：加载



4.3 桶形移位器（无寄存功能）

- 一次移动多位。组合逻辑电路，采用大量多路选择器实现



非循环移位：
逻辑右（左）移：高（低）位补0
算术右（左）移：高（低）位补符（0）

只有移位功能

没有寄存功能

Function Table for 4-

Select		Output				
S_1	S_0	Y_3	Y_2	Y_1	Y_0	Operation
0	0	D_3	D_2	D_1	D_0	No rotation
0	1	D_2	D_1	D_0	D_3	Rotate one position
1	0	D_1	D_0	D_3	D_2	Rotate two positions
1	1	D_0	D_3	D_2	D_1	Rotate three positions

第4章总结

- 时序逻辑电路不仅依赖当前输入，还依赖电路当前的状态
- 可用时序逻辑电路实现有限状态机。有Mealy型和Mooer型两类
- 可用状态图或状态表描述有限状态机，圈表示状态，有向边表示输入/输出
- 锁存器(电平触发): SR锁存器(设置标志)、D锁存器(锁存数据D)
- 触发器(时钟信号clk边沿触发)：D触发器(寄存器)、T触发器(计数或分频)
- 时序电路设计：功能分析-状态图-状态化简和编码-逻辑表达式-画图-评价
- 电路分析：未用状态分析(挂起/无法自启动)
  定时分析(clk-Q时间、时钟周期、setup时间、hold时间)
- 典型组合逻辑部件：计数器、寄存器/通用寄存器组、移位寄存器
- 计数器：同步/异步、加1/减1、行波（串行）进位/并行进位
- 寄存器：由n个D触发器构成，同时由时钟信号clk定时
- 通用寄存器组：两个读口(组合逻辑)；一个写口(时序逻辑，clk和写使能)
- 移位寄存器（时序逻辑）：每次固定左移或右移1位（或几位）
- 桶型移位器（组合逻辑）：移位位数可变，用大量多路选择器实现

作业：习题4、5、6、9、11、12。提交截止日期：10月19日24:00