

# 数字逻辑设计

高翠芸

School of Computer Science

gaocuiyun@hit.edu.cn

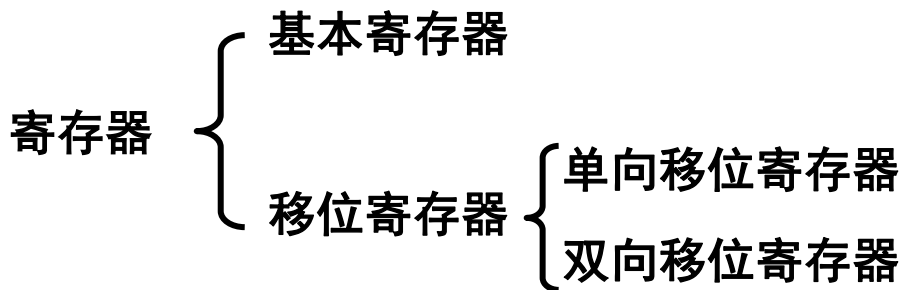
# 寄存器和计数器

---

- 寄存器 (Registers)
- 移位寄存器 (Shift Registers)
- 计数器 (Counters)
- 节拍发生器 (Beat Generator)

# 基本寄存器定义、分类等

- 寄存器是计算机的一个重要部件，用于暂时存放一组二值代码（如参加运算的数据、运算结果、指令等）。
- 由触发器及控制门组成



- 基本寄存器的操作：读出/写入/复位（清零）
- 移位寄存器的操作：读出/写入/复位（清零）/左移（右移）

# 基本寄存器

- 一个  $n$  位寄存器由  $n$  个触发器构成，能存放  $n$  位二进制数。
- 各种触发器均能构成寄存器，用 D 触发器最简单。

读出使能

En

4位寄存器

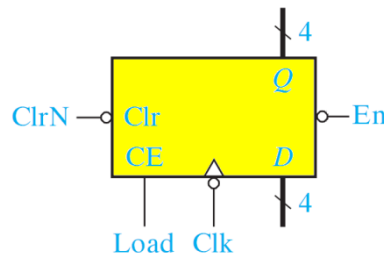
写入使能

Load  
ClrN  
Clk

同步时序

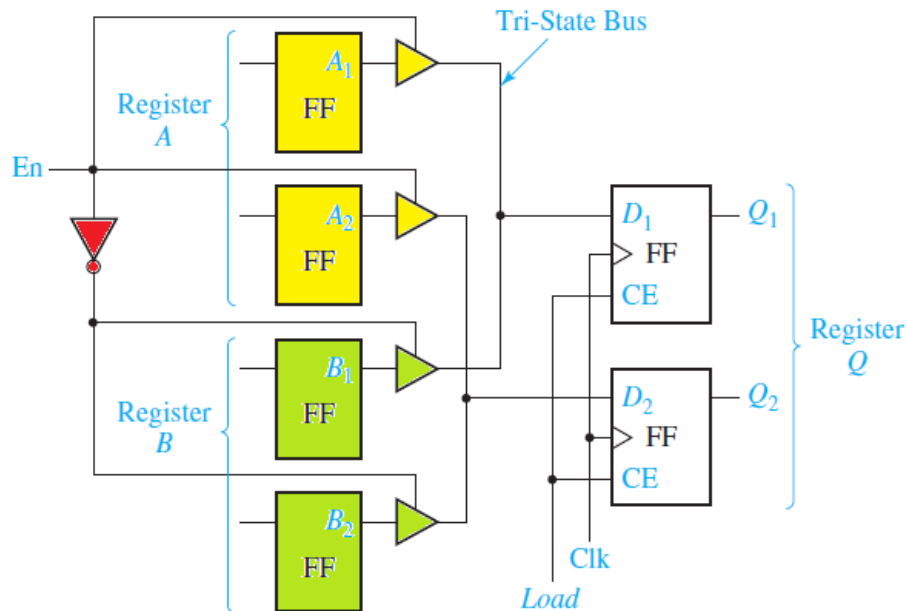
基本寄存器功能表

功能	条件	寄存器输出
异步清零	ClrN=0	$Q_3Q_2Q_1Q_0=0000$
保持	ClrN=1, 且 Load=0	$Q^{n+1}_3Q^{n+1}_2Q^{n+1}_1Q^{n+1}_0 = Q^n_3Q^n_2Q^n_1Q^n_0$
写入	ClrN=1, Load=1, clk ↓	$Q_3Q_2Q_1Q_0=D_3D_2D_1D_0$
读出	En=0	$Q_3Q_2Q_1Q_0=D_3D_2D_1D_0$



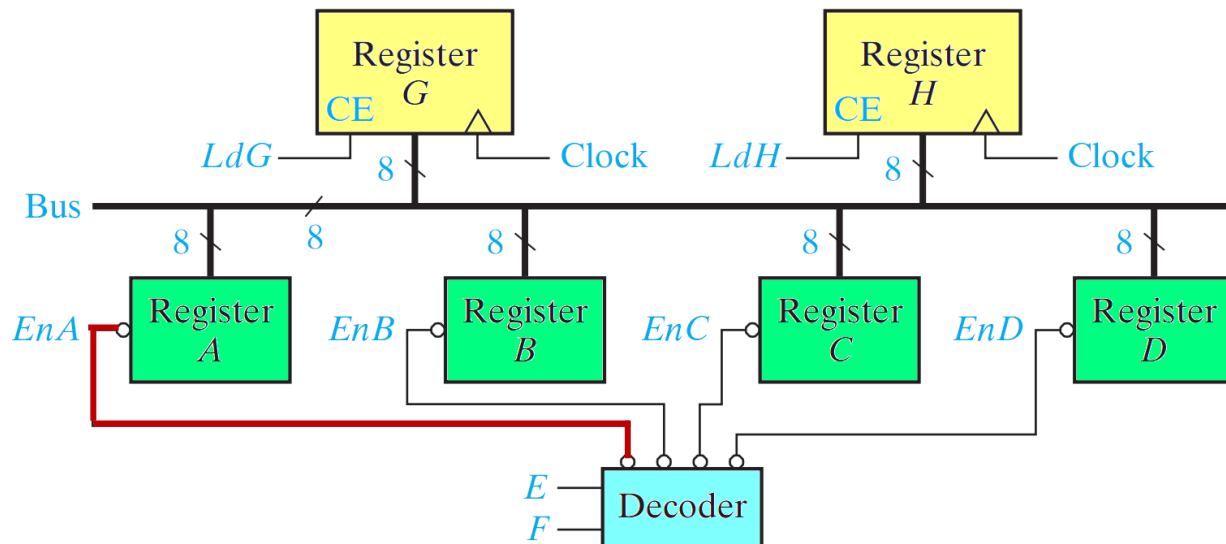
# 寄存器的应用——数据传送

## ■ 应用1——数据传送



- Register A to Q:  $en=1, load=1, clk \uparrow$
- Register B to Q:  $en=0, load=1, clk \uparrow$

# 寄存器的应用——利用三态总线进行数据传送

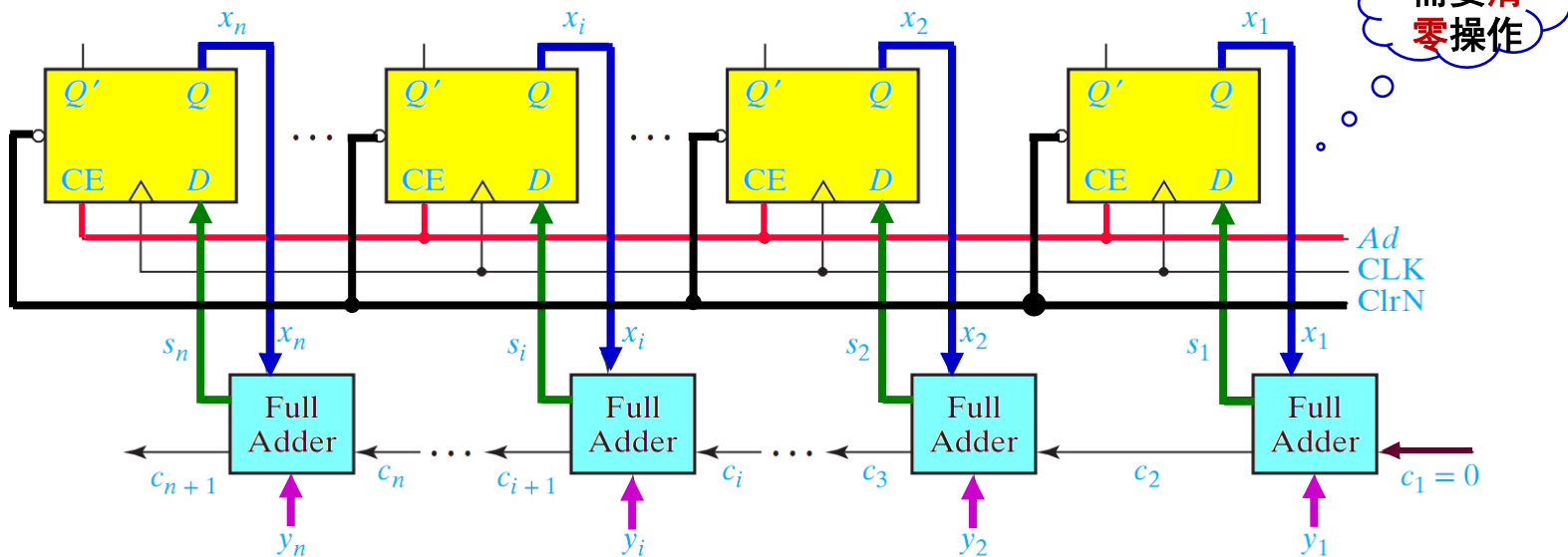


- Register A to G:  $EF=00$ , 且  $LdG=1, LdH=0$ ,  $clk \uparrow$
- Register B to H:  $EF=01$ , 且  $LdG=0, LdH=1$ ,  $clk \uparrow$

# 寄存器的应用

## ■ 应用2——具有累加功能的并行加法器1

$$X=X+Y$$

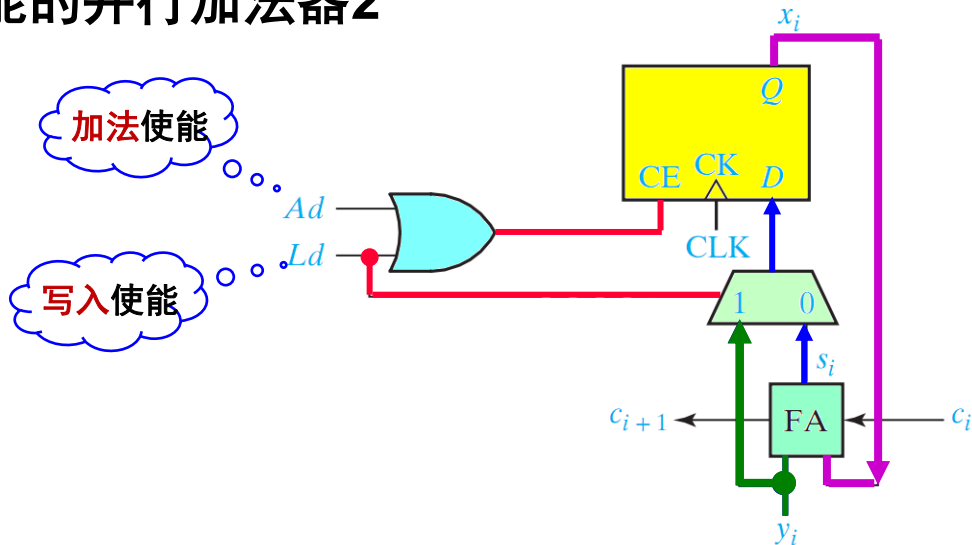


1. 初始化清零:  $ClrN=0$ , 则  $Q_n \dots Q_1=0$ , 即  $X_n \dots X_1=0$
2.  $ClrN=1$ , 将  $y_i$  送到全加器输入端
3. 执行  $S_i = y_i + x_i + C_i$
4. 存储累加和:  $ClrN=1, Ad=1, CLK \uparrow$  到来时, 寄存器  $Q_i=S_i$

# 寄存器的应用

## ■ 应用2——具有累加功能的并行加法器2

$$X = X + Y$$



### ■ 初始化:

$Ld=1$ , 则 $CE=1$ , 当 $ck \uparrow$ 到来时,  $Q_i = y_i$ 即 $y_i \rightarrow x_i$ , 将 $x_i$ 送到全加器的另一个输入端

### ■ 送入第二个操作数 $y_i$ , 执行 $S_i = y_i + x_i + C_i$



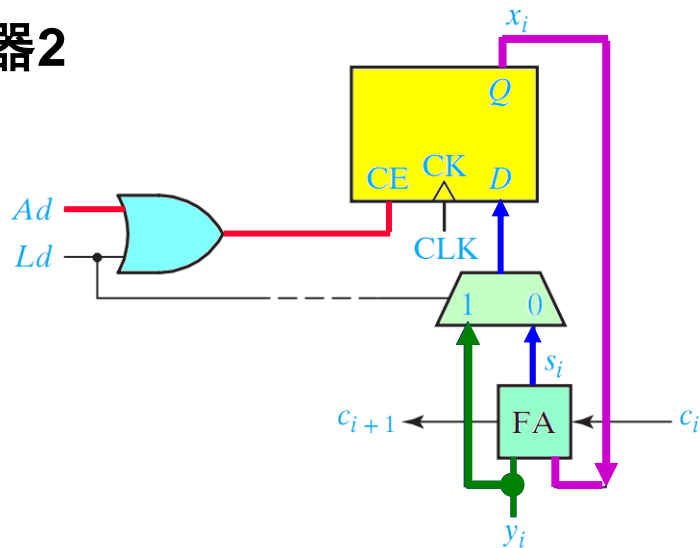
# 寄存器的应用

## ■应用2——具有累加功能的并行加法器2

$$X = X + Y$$

与方案1比较：

触发器不需要初始清零，通过一个二选一数据选择器，在第一个时钟沿送入一个操作数，之后在每个时钟沿送入累加和



### ■ 初始化：

$Ld=1$ ，则 $CE=1$ ，当 $ck \uparrow$ 到来时， $Q_i=y_i$ 即 $y_i \rightarrow x_i$ ，将 $x_i$ 送到全加器的另一个输入端

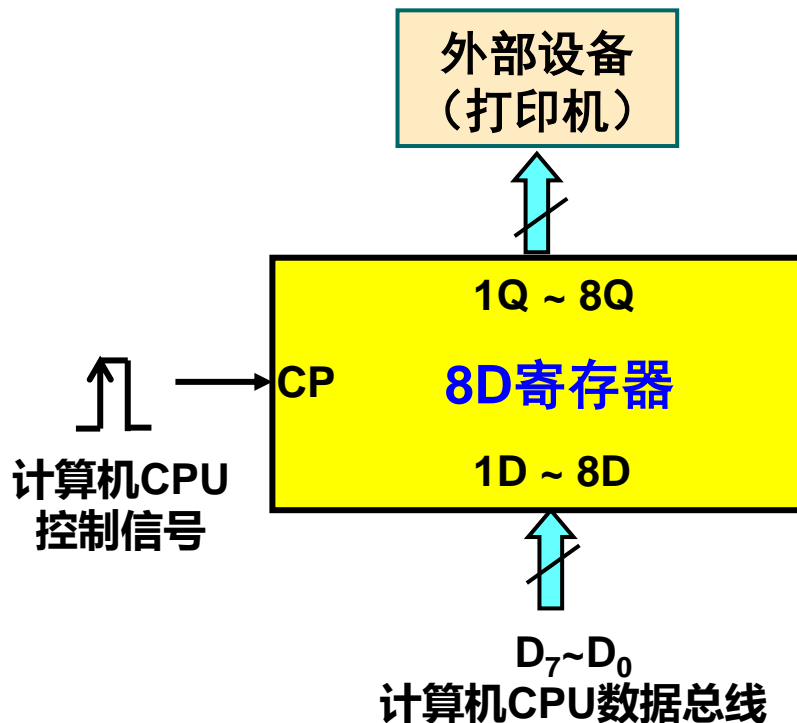
### ■ 送入第二个操作数 $y_i$ ，执行 $S_i=y_i+x_i$

### ■ $Ld=0$ ， $Ad=1$ ， $ck \uparrow$ 到来时： $x_i = s_i$

### ■ 保持： $Ld=0$ ， $Ad=0$

# 寄存器的应用

## ■ 应用3——计算机并行输入/输出接口



# 寄存器和计数器

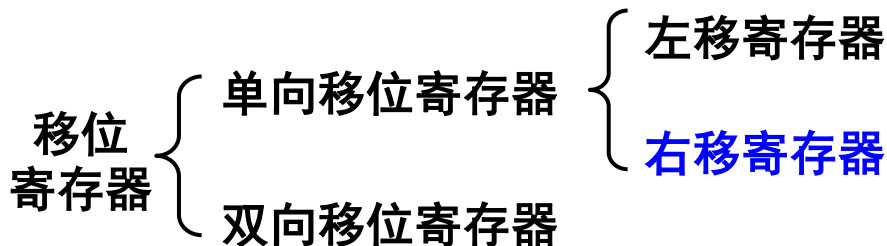
---

- 寄存器 (Registers)
- 移位寄存器 (Shift Registers)
- 计数器 (Counters)
- 节拍发生器 (Beat Generator)

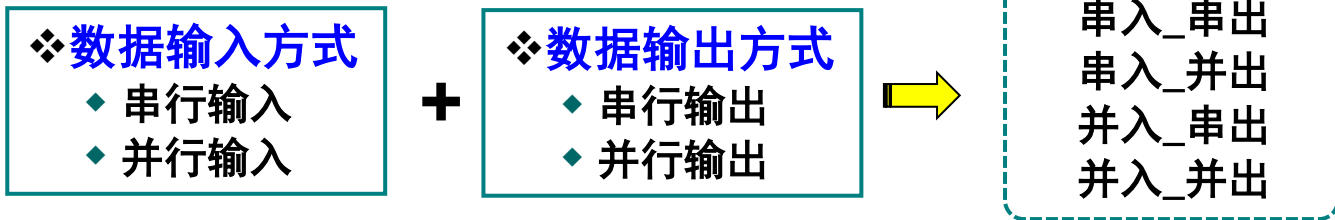
# 移位寄存器定义和分类

- 每来一个时钟脉冲，寄存器里存储的数据，能依次左移或右移1位。
- 可以实现代码的串、并行转换、数值运算和数据处理等。

## ➤ 分类



## ➤ 工作方式



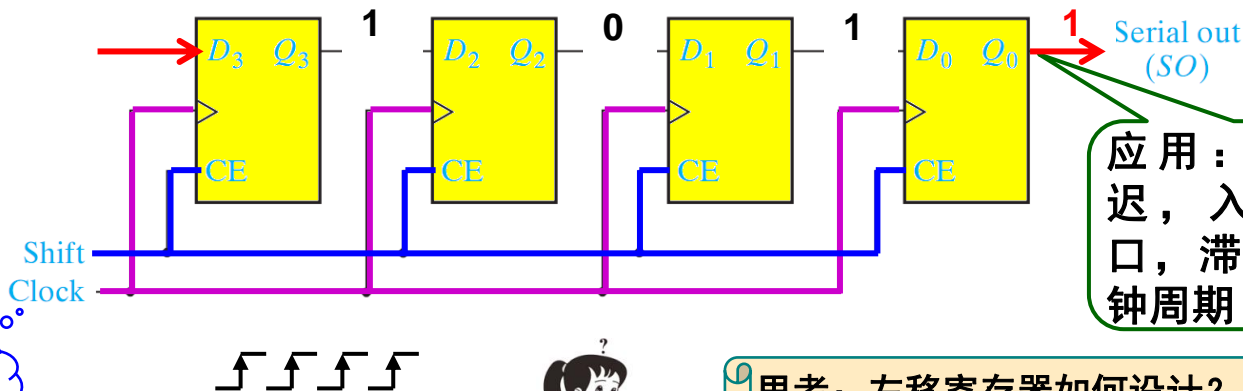
# 右移寄存器 (Right-Shift Register)

## (1). 串行输入/串行输出 ( Serial in / Serial out )

- 串行输出：移位路径上最后一个触发器的输出作为整个电路的输出

右移方式下：  
数据从串行  
输入端送入，  
应该**先送最  
低位**

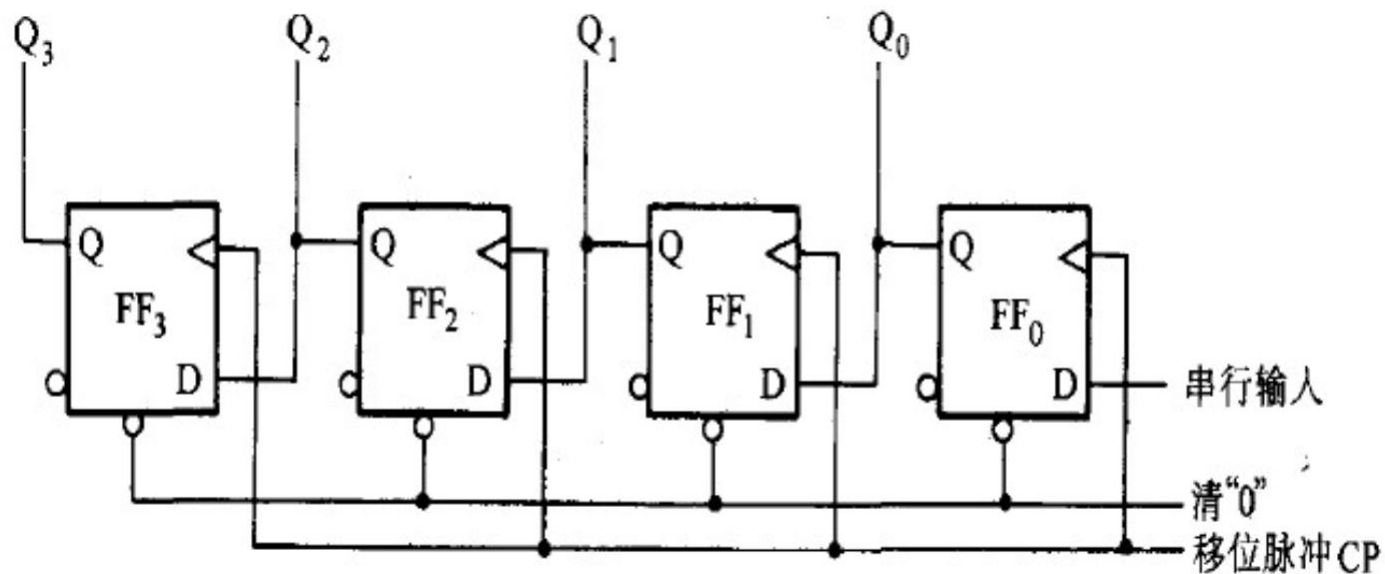
同步时序



应用：时间延迟，入口到出口，滞后4个时钟周期

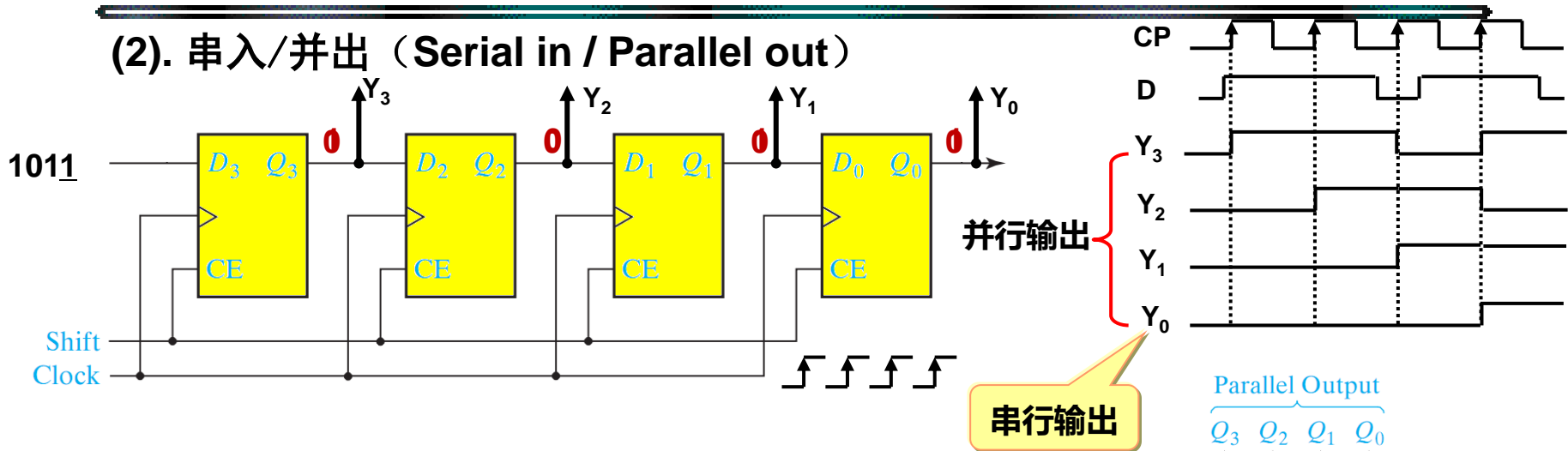
思考：左移寄存器如何设计？左移方式下从串行输入端送入数据，应该先送最低位还是最高位？

# 左移寄存器



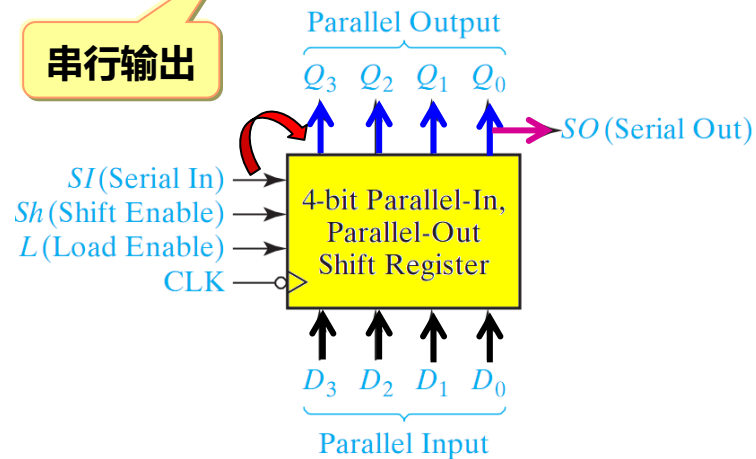
# 右移寄存器

## (2). 串入/并出 (Serial in / Parallel out)



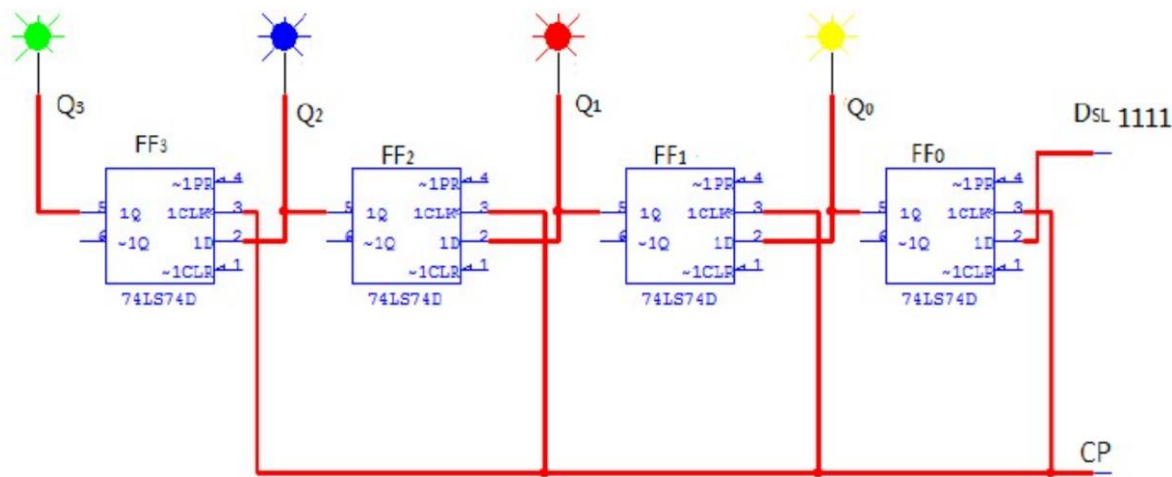
## (3). 并入/并出 (Parallel in / Parallel out)

## (4). 并入/串出 (Parallel in / Serial out)



Inputs		Next State				Action
Sh (Shift)	L (Load)	Q <sub>3</sub> <sup>+</sup>	Q <sub>2</sub> <sup>+</sup>	Q <sub>1</sub> <sup>+</sup>	Q <sub>0</sub> <sup>+</sup>	
0	0	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	No change
0	1	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	Load
1	X	SI	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Right shift

# 移位寄存器的应用举例

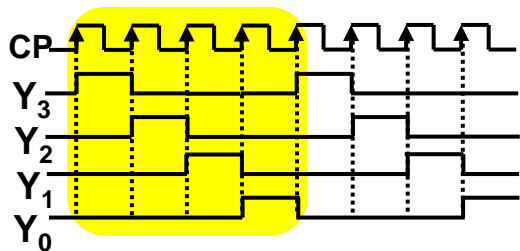
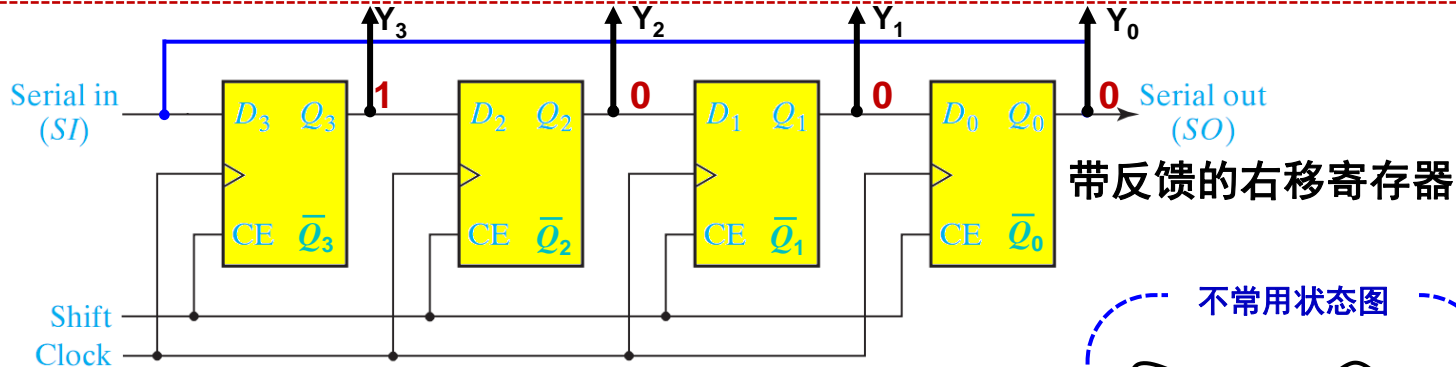


D触发器的输出接彩灯（LED）输出为1时LDE亮，输出为0时LDE灭，在输入端使 $D_{SL}=1$ ，在4个移位脉冲CP作用下，彩灯出现从右到左一个个点亮的效果。



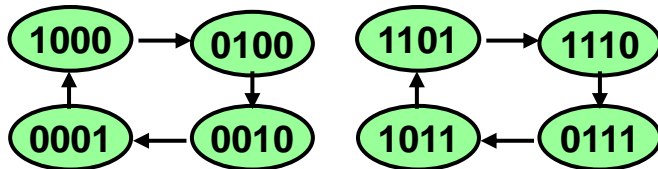
# 右移寄存器的应用——环形计数器

**计数器：**一种能在输入信号作用下依次循环通过预定状态的时序逻辑电路。



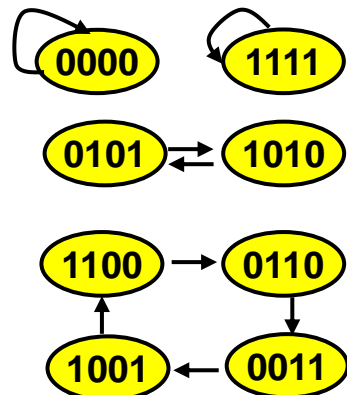
优点：电路简单  
输出具有**二进制译码器**的特点

常用状态图

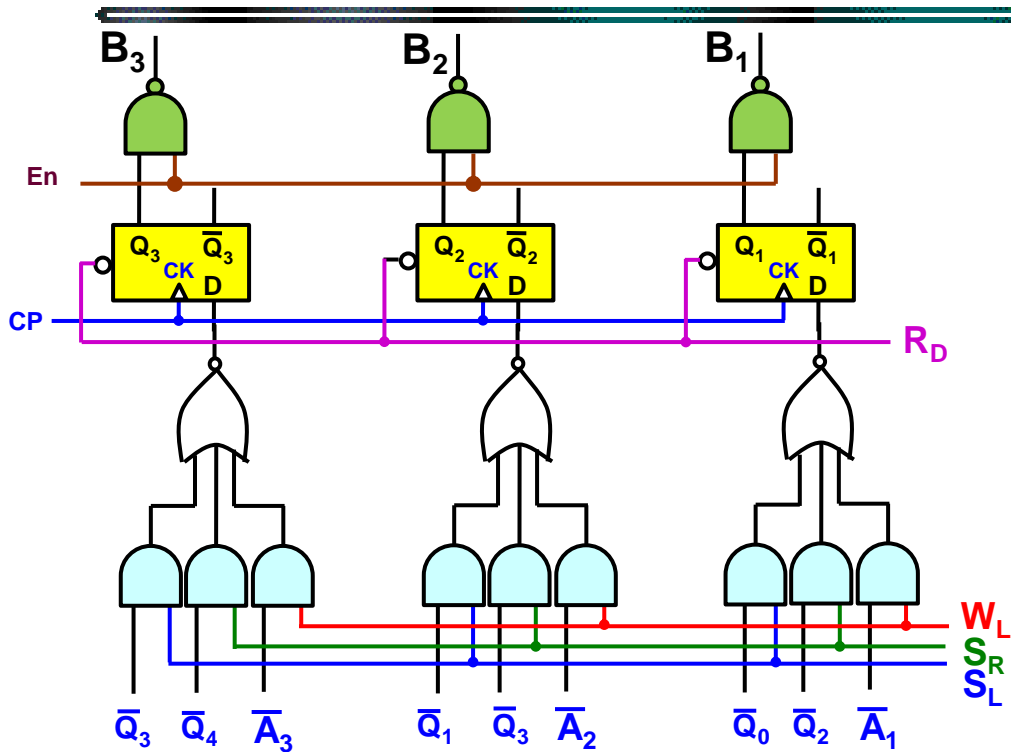


缺点： $2^n$ 个状态只用了  $n$  个；  
不能自启动，需要**预置**

不常用状态图



# 双向移位寄存器



**R<sub>d</sub>**——异步清零； **W<sub>L</sub>**——写入使能  
**S<sub>R</sub>**——右移使能； **S<sub>L</sub>**——左移使能  
**En**——输出使能

## 输入方程

$$\left\{ \begin{array}{l} \mathbf{D}_3 = \overline{\mathbf{A}_3} \mathbf{W}_L + \overline{\mathbf{Q}_4} \mathbf{S}_R + \overline{\mathbf{Q}_2} \mathbf{S}_L \\ \mathbf{D}_2 = \overline{\mathbf{A}_2} \mathbf{W}_L + \overline{\mathbf{Q}_3} \mathbf{S}_R + \overline{\mathbf{Q}_1} \mathbf{S}_L \\ \mathbf{D}_1 = \overline{\mathbf{A}_1} \mathbf{W}_L + \overline{\mathbf{Q}_2} \mathbf{S}_R + \overline{\mathbf{Q}_0} \mathbf{S}_L \end{array} \right.$$

## 输出方程

$$\left\{ \begin{array}{l} \mathbf{B}_3 = \overline{\mathbf{Q}_3 \mathbf{E}_n} \\ \mathbf{B}_2 = \overline{\mathbf{Q}_2 \mathbf{E}_n} \\ \mathbf{B}_1 = \overline{\mathbf{Q}_1 \mathbf{E}_n} \end{array} \right.$$

## 次态方程

$$\begin{cases} \mathbf{Q}_3^{n+1} = \mathbf{D}_3 \\ \mathbf{Q}_2^{n+1} = \mathbf{D}_2 \\ \mathbf{Q}_1^{n+1} = \mathbf{D}_1 \end{cases}$$



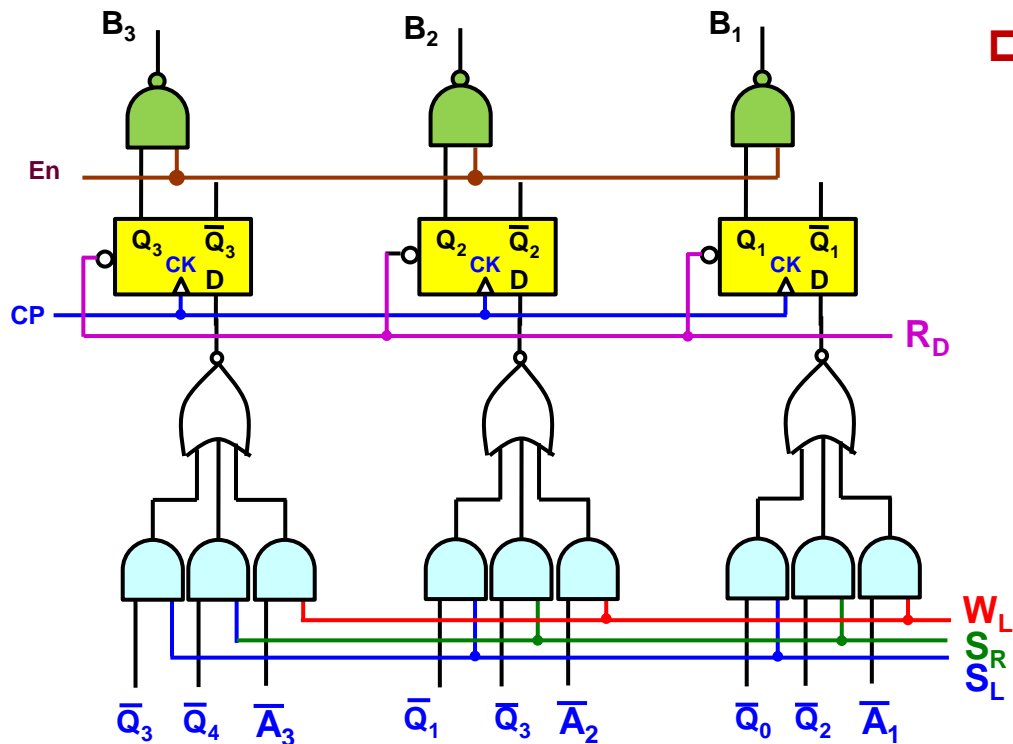
**Let:  $W_L = 1$ ,  $S_R = S_L = 0$**

**当 CP ↑ 上升沿到来时:**

次态方程  $\begin{cases} Q_3^{n+1} = D_3 = A_3 \\ Q_2^{n+1} = D_2 = A_2 \\ Q_1^{n+1} = D_1 = A_1 \end{cases}$

输入方程

$$\left\{ \begin{array}{l} D_3 = \overline{A_3} W_L + \overline{Q_4} S_R + \overline{Q_2} S_L = \overline{A_3} \cdot 1 + \overline{Q_4} \cdot 0 + \overline{Q_2} \cdot 0 = A_3 \\ D_2 = \overline{A_2} W_L + \overline{Q_3} S_R + \overline{Q_1} S_L = \overline{A_2} \cdot 1 + \overline{Q_3} \cdot 0 + \overline{Q_1} \cdot 0 = A_2 \\ D_1 = \overline{A_1} W_L + \overline{Q_2} S_R + \overline{Q_0} S_L = \overline{A_1} \cdot 1 + \overline{Q_2} \cdot 0 + \overline{Q_0} \cdot 0 = A_1 \end{array} \right.$$



## □ 功能——

### (2) 右移

Let:  $S_R = 1$ ,  $W_L = S_L = 0$

当  $CP \uparrow$  上升沿到来时:

次态方程

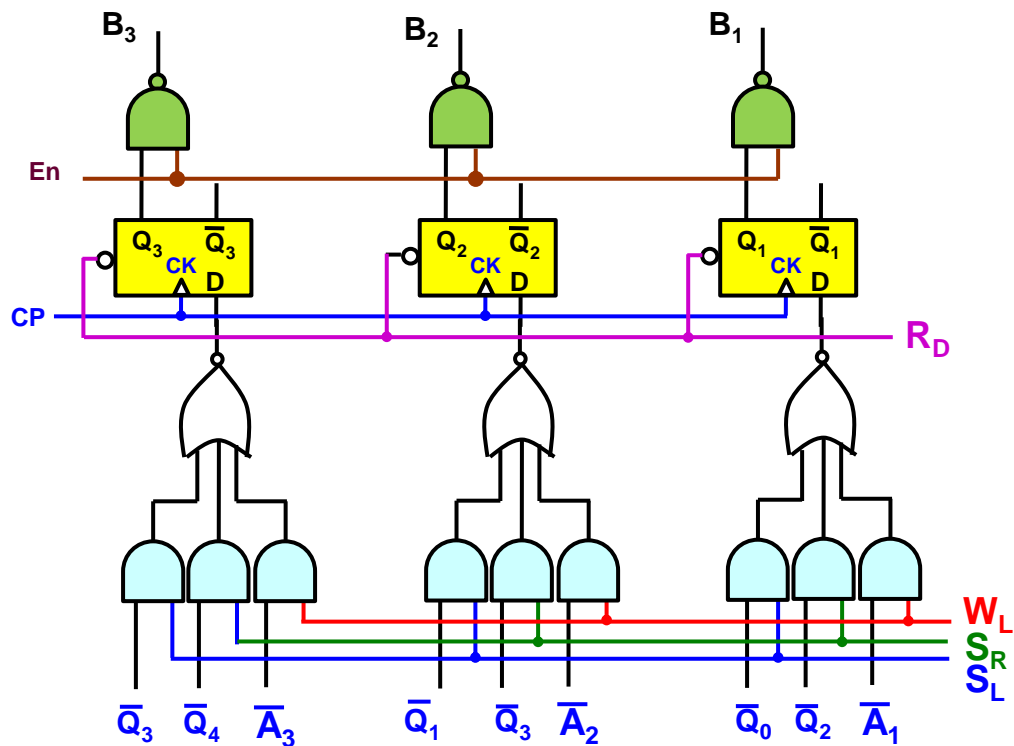
$$\begin{cases} Q_3^{n+1} = D_3 = Q_4 \\ Q_2^{n+1} = D_2 = Q_3 \\ Q_1^{n+1} = D_1 = Q_2 \end{cases}$$

输入方程

$$\begin{cases} D_3 = \overline{A_3} W_L + \overline{Q_4} S_R + \overline{Q_2} S_L = \overline{A_3} \cdot 0 + \overline{Q_4} \cdot 1 + \overline{Q_2} \cdot 0 = Q_4 \\ D_2 = \overline{A_2} W_L + \overline{Q_3} S_R + \overline{Q_1} S_L = \overline{A_2} \cdot 0 + \overline{Q_3} \cdot 1 + \overline{Q_1} \cdot 0 = Q_3 \\ D_1 = \overline{A_1} W_L + \overline{Q_2} S_R + \overline{Q_0} S_L = \overline{A_1} \cdot 0 + \overline{Q_2} \cdot 1 + \overline{Q_0} \cdot 0 = Q_2 \end{cases}$$



# 双向移位寄存器——读出



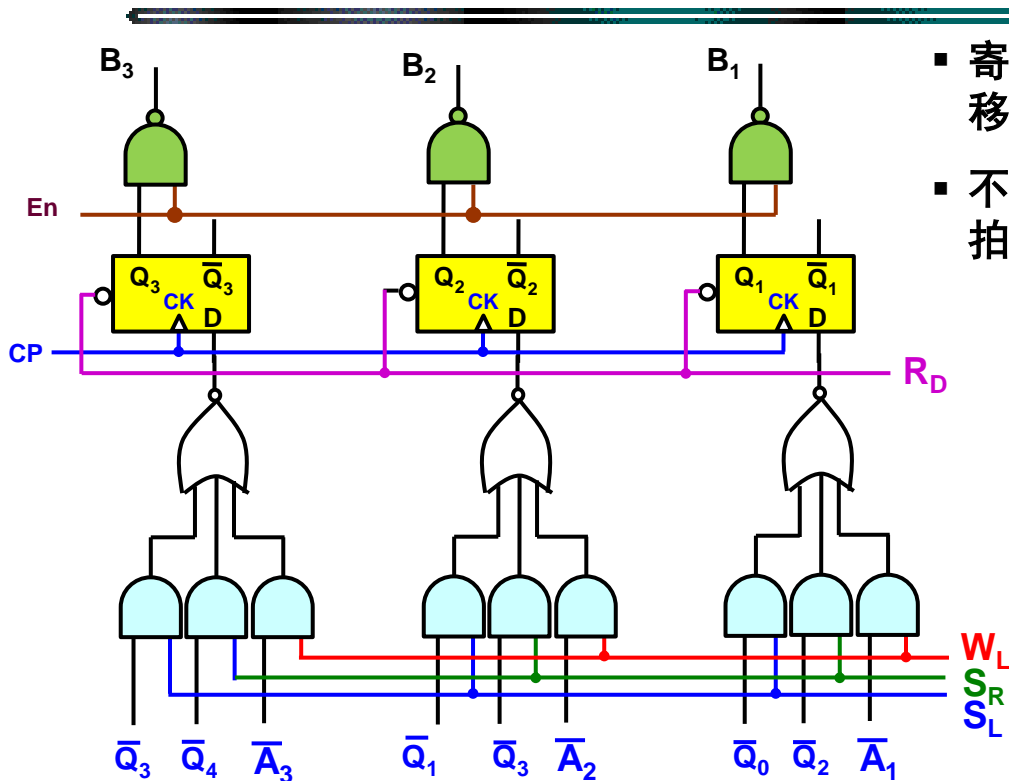
(4) 读出

Let:  $E_n = 1$

输出方程

$$\begin{cases} B_3 = \bar{Q}_3 E_n = \bar{Q}_3 \\ B_2 = \bar{Q}_2 E_n = \bar{Q}_2 \\ B_1 = \bar{Q}_1 E_n = \bar{Q}_1 \end{cases}$$

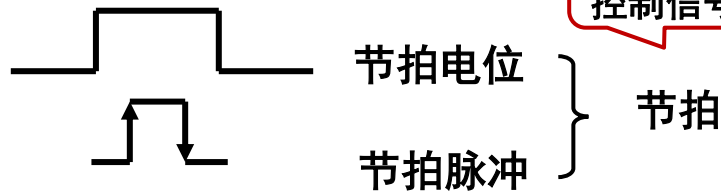
# 双向移位寄存器



- 寄存器的每一个操作（写入、读出、左移、右移）都是在**节拍**的控制下完成的。
- 不改变触发器状态的操作（读出），只需要节拍电位。

必须保证节拍脉冲的**边沿**被节拍电位的有效电平**完全覆盖**

节拍：一种控制信号



例如：

- 写入操作，需要  $W_L = 1$ ，同时  $CP \uparrow$
- 左移操作，需要  $S_L = 1$ ，同时  $CP \uparrow$
- 读出操作，只需要  $En=1$

# 寄存器总结

---

- ❑ 寄存器主要功能：存放二进制数据（存储位数由里面触发器的数量决定）
- ❑ 寄存器操作：写入、读出、保持、清零。
- ❑ 移位寄存器还可以：将数据依次左移或右移1位
- ❑ 特点：寄存器的每一个操作（写入、读出、左移、右移）都是在节拍的控制下完成的

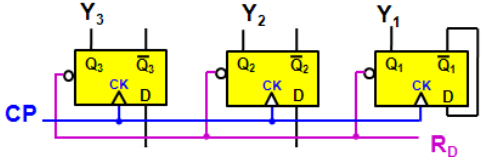
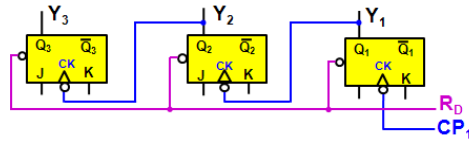
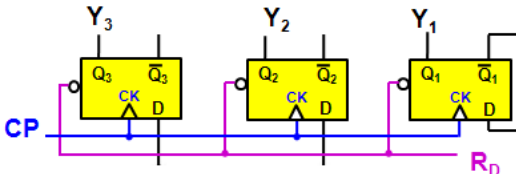
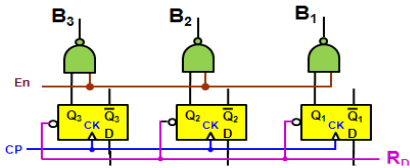


# 寄存器和计数器

---

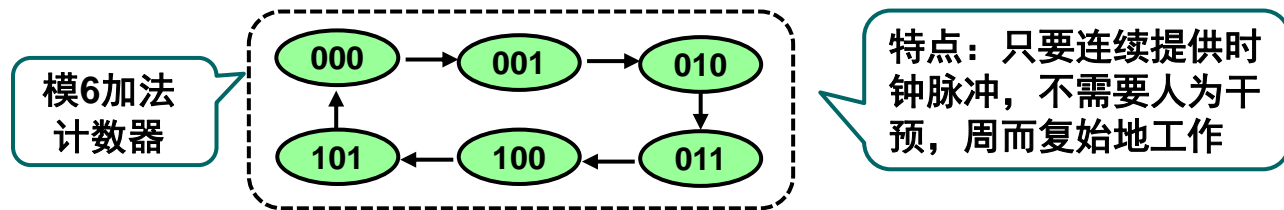
- 寄存器 (Registers)
- 移位寄存器 (Shift Registers)
- 计数器 (Counters)
- 节拍发生器 (Beat Generator)

# 时序逻辑电路的分类

分类方式		种类	特点	电路框图示例
时序逻辑电路	按照 <b>时钟</b> 信号的连接方式	同步时序——	<ul style="list-style-type: none"> <li>所有的<b>时钟端</b>连接在一起，状态的改变<b>同时</b>发生（数字系统中用到的最多）</li> </ul>	
		异步时序——	<ul style="list-style-type: none"> <li><b>没有</b>统一的时钟脉冲同步，状态的改变有先有后，<b>不同</b>时发生</li> <li>容易产生毛刺（有不利影响）</li> </ul>	
	按照电路 <b>输出</b> 与 <b>输入</b> 及电路 <b>状态</b> 的关系	摩尔型电路 (Moore)	<ul style="list-style-type: none"> <li>电路的<b>输出</b>仅与<b>现态</b>有关，与电路的<b>输入</b>无关；或者直接以电路状态作为输出。</li> </ul>	
		米里型电路 (Mealy)	<ul style="list-style-type: none"> <li>电路<b>输出</b>与电路的<b>现态</b>及电路的<b>输入</b>均有关；</li> </ul>	

# 典型时序逻辑部件——计数器

计数器是一种能在输入信号作用下依次通过预定状态的时序逻辑电路，是数字系统和计算机广泛使用的逻辑器件，可用于计数、分频、定时、控制、产生节拍脉冲（顺序脉冲）和序列脉冲等。



- 由一组触发器构成，计数器中的“数”是用触发器的状态组合来表示的。
- 计数器在运行时，所经历的状态是周期性的，总是在有限个状态中循环。
- 将一次循环所包含的**状态总数**称为计数器的“**模**”，记为N,包含n个触发器的最大模值  $N = 2^n$ 。
- 把作用于计数器的时钟脉冲称为计数脉冲，用 **CP** (或**CLK**)表示。

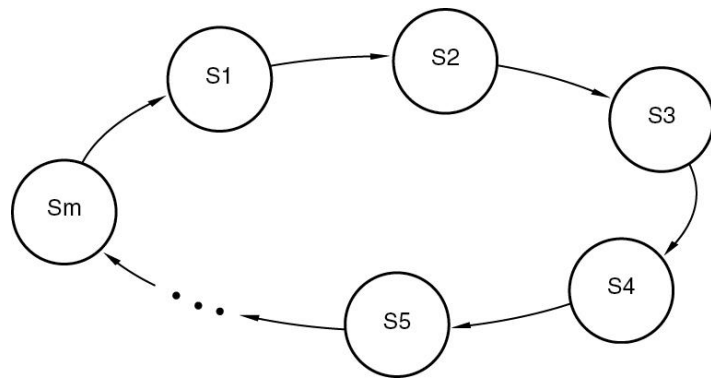
# 计数器

- 状态图中包含一个循环的任何时序电路都可称为计数器
- 计数器的模是循环中的状态个数

- 最常用的是n位二进制计数器

- 有n个触发器

- $2^n$ 种状态, 0, 1, 2, ...,  $2^n-1$ , 0, 1, ...



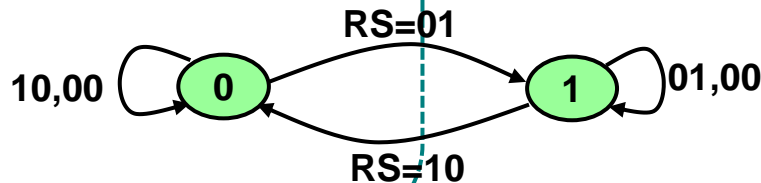
# 计数器的种类

- (1) 按时钟方式分为：同步计数器和异步计数器（行波计数器 ripple counter）；
- (2) 按功能分为：加法计数器、减法计数器和可逆计数器等。
- (3) 按计数方式分为：二进制计数器，十进制计数器，M进制计数器

## 时序逻辑电路的分析方法

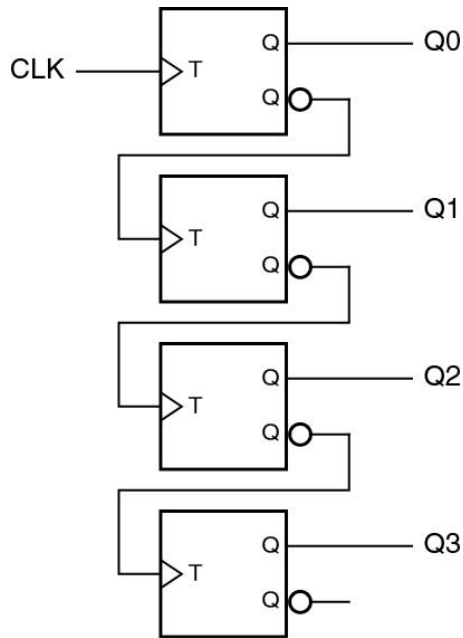
确定系统变量（输入变量、输出变量、状态变量）

- ① 列驱动方程（控制函数）
- ② 列输出方程（输出函数）
- ③ 列状态方程（次态方程）
- ④ 列写状态转换表
- ⑤ 画出状态图
- ⑥ 画出波形图（如必要）

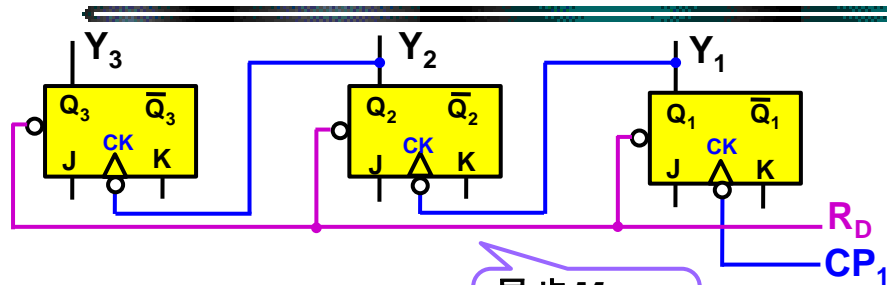


# 行波计数器(异步计数器)

- 只用 $n$ 个触发器即可实现 $n$ 位二进制计数器
- 各个触发器的时钟信号不同。
- T触发器每个时钟上升沿都会产生反转  
利用此特性，模拟二进制计数器的进位



# 异步计数器



## ① 输入方程

$$\begin{aligned} J_1 &= K_1 = 1 & CP_1 &\downarrow \\ J_2 &= K_2 = 1 & CP_2 &= Y_1 \downarrow \\ J_3 &= K_3 = 1 & CP_3 &= Y_2 \downarrow \end{aligned}$$

异步Moor  
型时序电路

异步模8加  
法计数器

## ② 次态方程

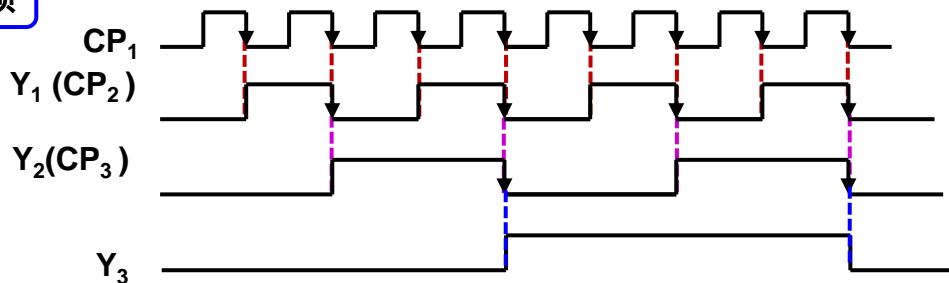
$$\begin{aligned} Y_1^{n+1} &= J_1 \bar{Q}_1 + \bar{K}_1 Q_1 = \bar{Y}_1 & CP_1 &\downarrow \\ Y_2^{n+1} &= J_2 \bar{Q}_2 + \bar{K}_2 Q_2 = \bar{Y}_2 & Y_1 &\downarrow \\ Y_3^{n+1} &= J_3 \bar{Q}_3 + \bar{K}_3 Q_3 = \bar{Y}_3 & Y_2 &\downarrow \end{aligned}$$

## ③ 状态转换表

现态			次态			时钟		
$Y_3^n$	$Y_2^n$	$Y_1^n$	$Y_3^{n+1}$	$Y_2^{n+1}$	$Y_1^{n+1}$	$CP_3$	$CP_2$	$CP_1$
0	0	0	0	0	1	无	无	↓
0	0	1	0	1	0	无	↓	↓
0	1	0	0	1	1	无	无	↓
0	1	1	1	0	0	↓	↓	↓
1	0	0	1	0	1	无	无	↓
1	0	1	1	1	0	无	↓	↓
1	1	0	1	1	1	无	无	↓
1	1	1	0	0	0	↓	↓	↓

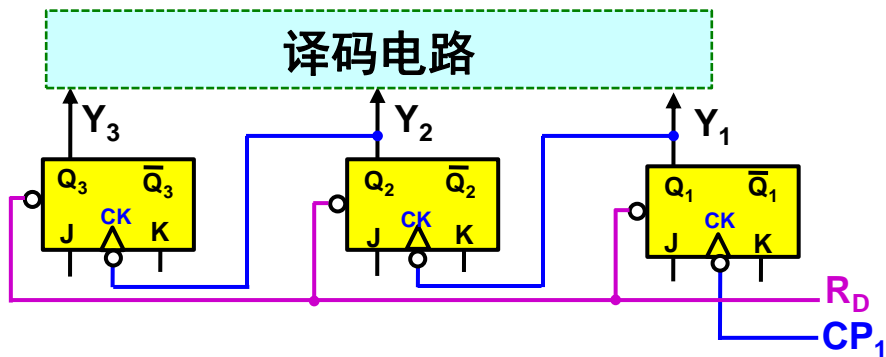
二分频

## ④ 波形图



# 异步计数器总结

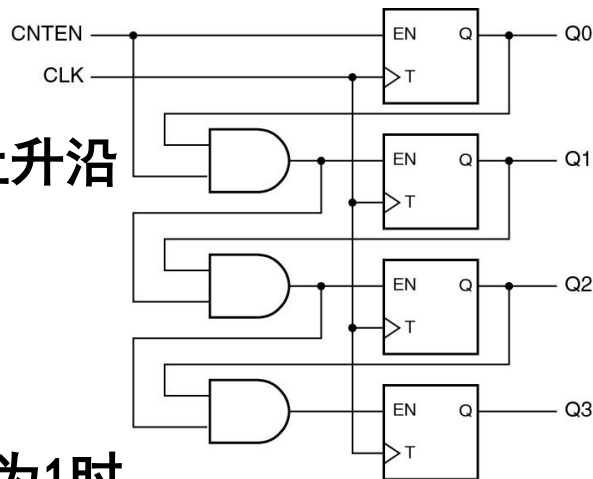
- 外接时钟源只作用于**最低位触发器**，高位触发器的时钟信号通常由低位触发器的输出提供，高位触发器的翻转**有待**低位触发器翻转后才能进行。
- 每一级触发器都存在**传输延迟**，位数越多计数器工作速度越慢，在大型数字设备中较少采用。
- 对计数器状态进行译码时，由于触发器不同步，译码器输出会出现**尖峰脉冲**（位数越多，尖峰信号越宽），使仪器设备产生误动作。
- **优点**：结构比较简单，所用元件较少。





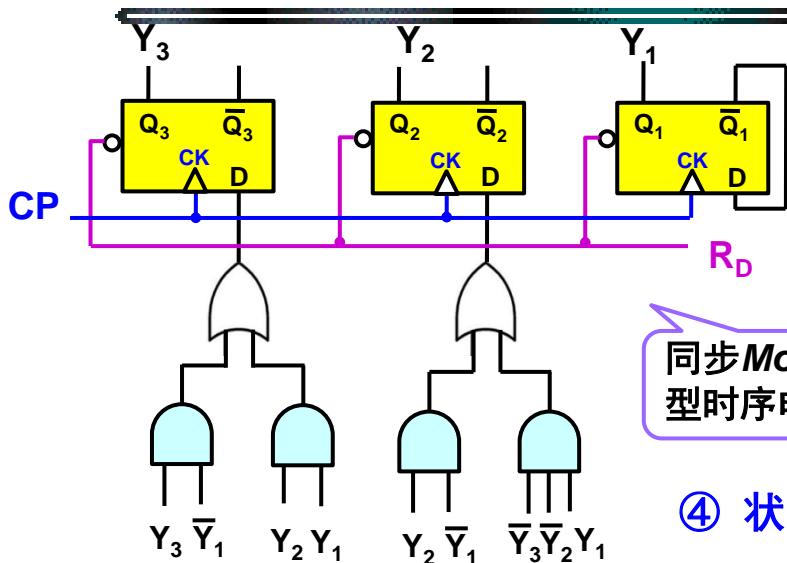
# 同步计数器

- 触发器共用时钟信号CLK
- 使能端EN有效时，触发器输出在T信号上升沿反转
- 主计数器使能信号CNTEN
  - CNTEN信号有效且所有低阶计数位为1时，每个计数器都翻转



串行4位二进制计数器

# 同步计数器



## ① 输入方程

$$D_3 = Y_3 \bar{Y}_1 + Y_2 Y_1$$

$$D_2 = Y_2 \bar{Y}_1 + \bar{Y}_3 \bar{Y}_2 Y_1$$

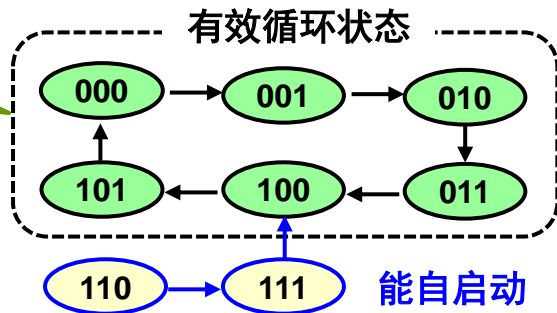
$$D_1 = \bar{Y}_1$$

## ② 次态方程

$$\begin{cases} Y_1^{n+1} = D_1 \\ Y_2^{n+1} = D_2 \\ Y_3^{n+1} = D_3 \end{cases}$$

## ④ 状态图

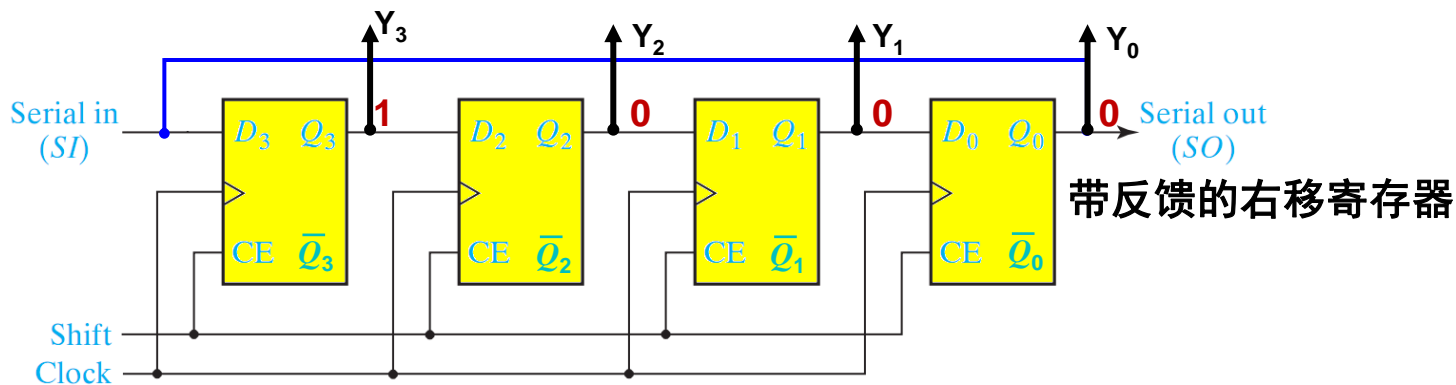
同步模6加法计数器



## ③ 状态转换表

现态			次态			时钟
$Y_3^n$	$Y_2^n$	$Y_1^n$	$Y_3^{n+1}$	$Y_2^{n+1}$	$Y_1^{n+1}$	CP
0	0	0	0	0	1	↑
0	0	1	0	1	0	↑
0	1	0	0	1	1	↑
0	1	1	1	0	0	↑
1	0	0	1	0	1	↑
1	0	1	0	0	0	↑
1	1	0	1	1	1	↑
1	1	1	1	0	0	↑

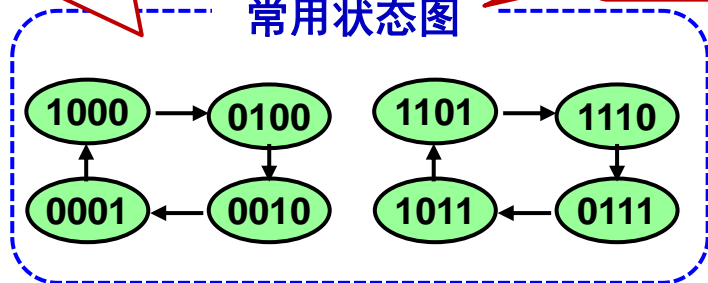
# 同步计数器——环形计数器



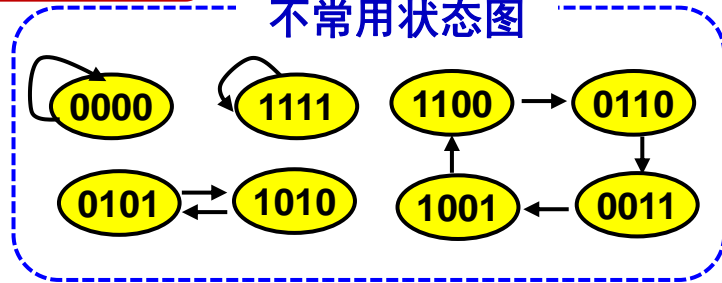
优点：电路简单，  
输出具有二进制  
译码器的特点

缺点： $2^n$ 个状态只使用了 $n$ 个；  
不能自启动，需要预置

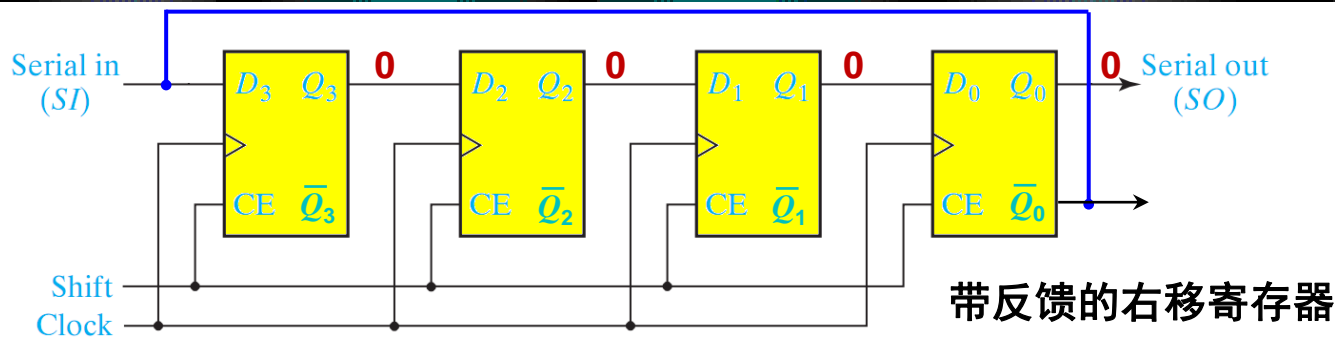
常用状态图



不常用状态图

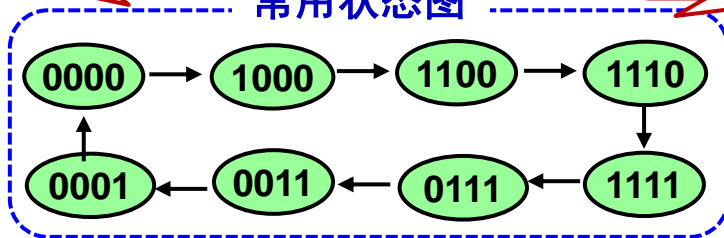


# 同步计数器——扭环形计数器Johnson Counter



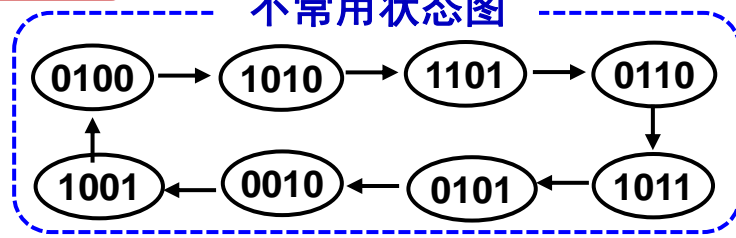
电路具有格雷码输出的特点

常用状态图



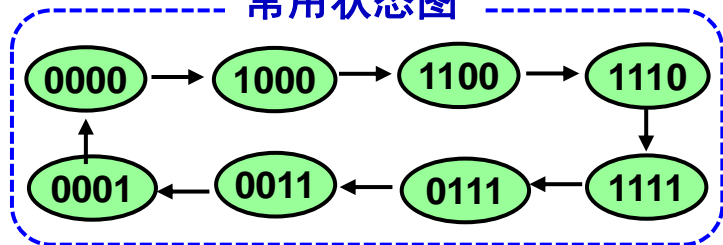
$2^n$ 个状态使用了 $2n$ 个;  
不能自启动, 需要预置

不常用状态图



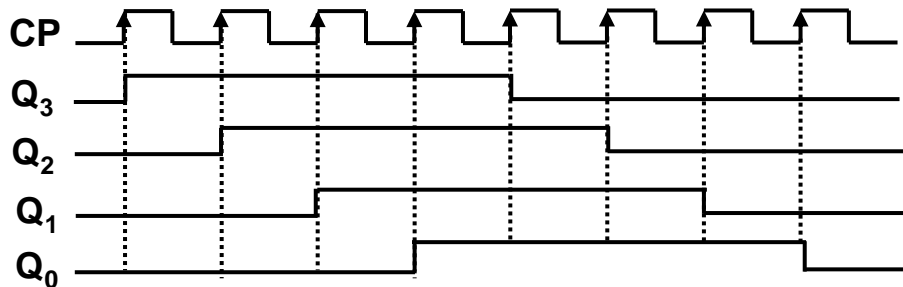
# 扭环形计数器

常用状态图



优点：①无险象  
②后级每个译码门只需要2个输入端。  
③模8计数器

输入				译码输出							
$Q_3$	$Q_2$	$Q_1$	$Q_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$	$Y_4$	$Y_5$	$Y_6$	$Y_7$
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0
1	1	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	1	0	0	0	0
1	1	1	1	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	1



$Q_3Q_2$ \ $Q_1Q_0$	00	01	11	10
00	1	0	0	X
01	X	X	0	X
11	0	X	0	0
10	0	X	X	X

$$Y_0 = \bar{Q}_3 \bar{Q}_0$$

$Q_3Q_2$ \ $Q_1Q_0$	00	01	11	10
00	0	0	0	X
01	X	X	0	X
11	0	X	0	0
10	1	X	X	X

$$Y_1 = Q_3 \bar{Q}_2$$

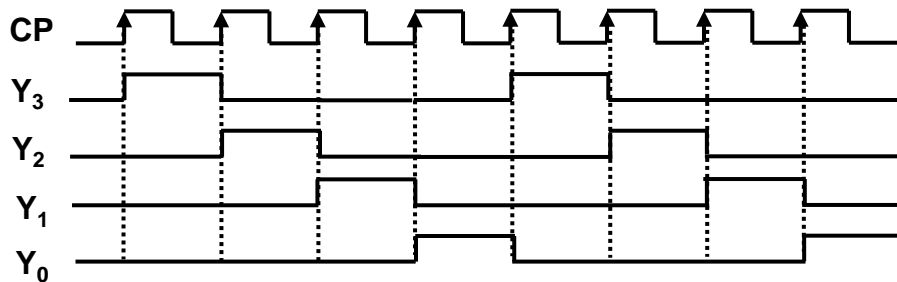
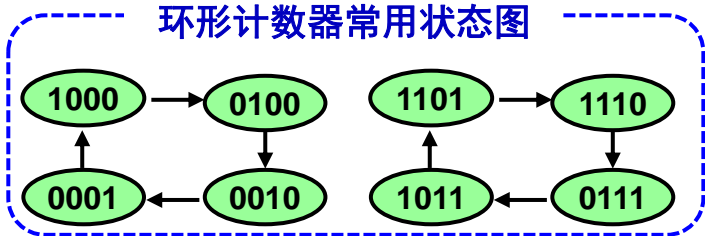
# 环形、扭环形计数器总结

特点：在移位寄存器的基础上，增加反馈逻辑电路组成。

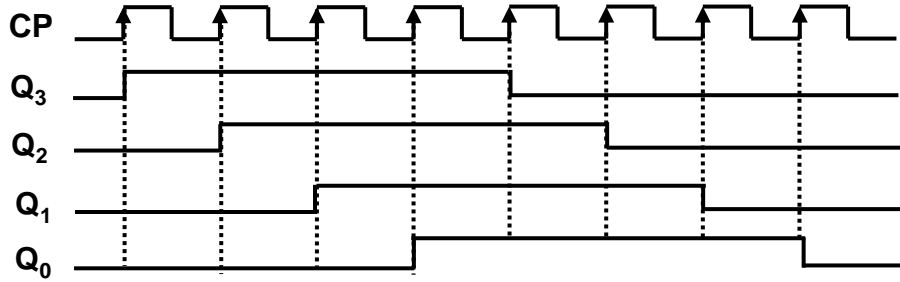
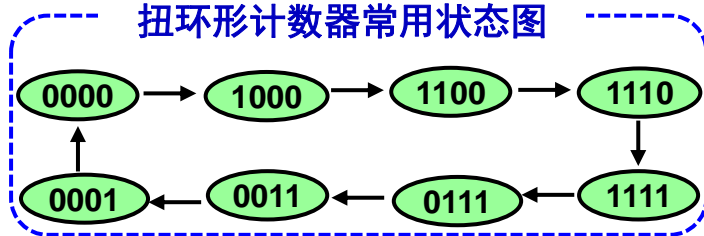
用途：

- 构成**特殊编码**的计数器（非二进制计数器）
- 环形计数器和扭环形计数器在计算机中可用于组成时序信号发生器（节拍发生器）

环形计数器常用状态图

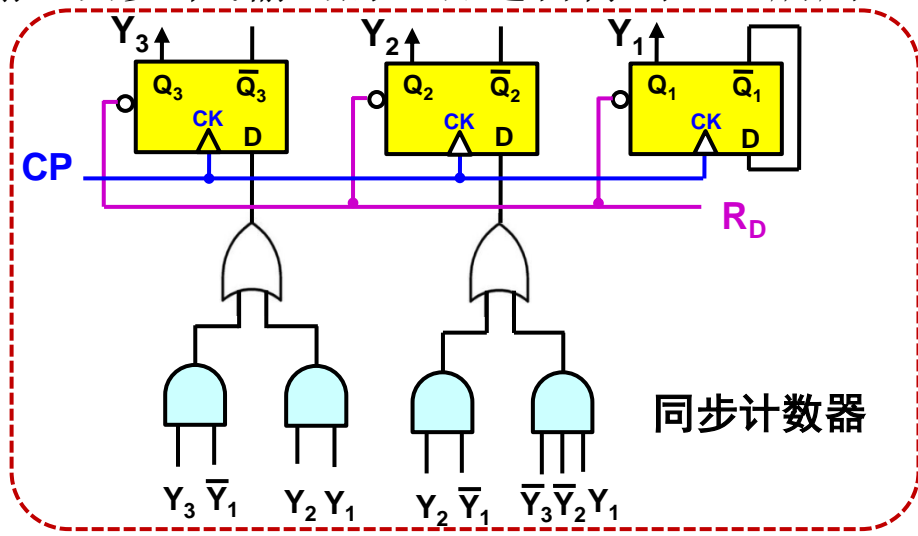
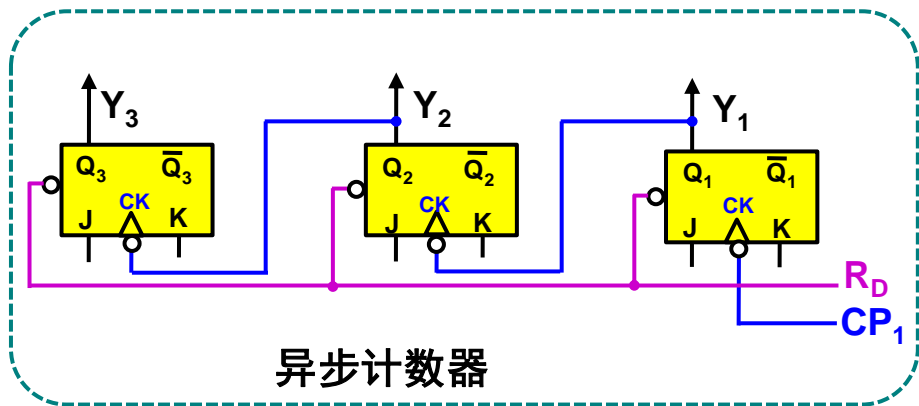


扭环形计数器常用状态图

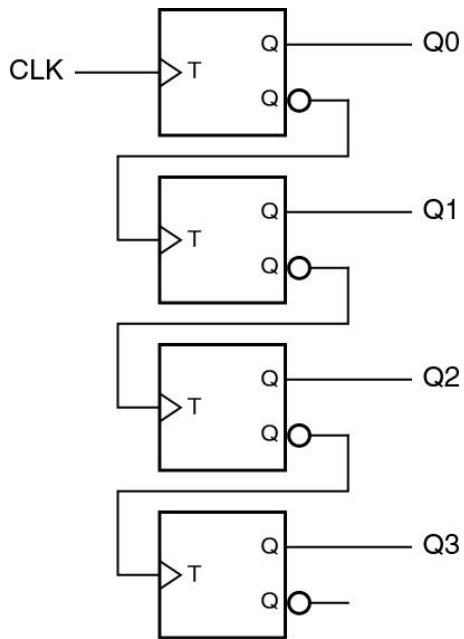


# 同步计数器总结

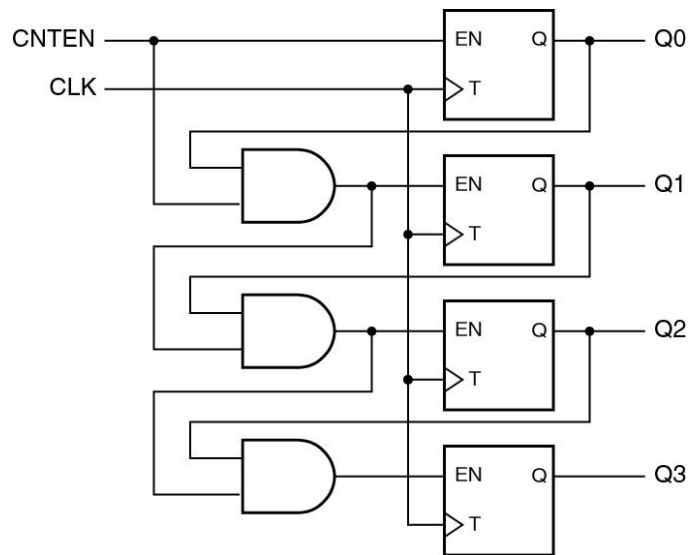
- ❑ 所有触发器的时钟端并联在一起，受控于同一个外接时钟源；
- ❑ 所有触发器**同时翻转**，不存在时钟到各触发器输出的传输延迟的积累；
- ❑ 同步计数器的工作频率只与一个触发器的时钟到输出的传输延迟有关，所以它的工作频率比异步计数器高；
- ❑ 由于各触发器同时翻转，因此，同步计数器的输出**不会产生毛刺**；
- ❑ 缺点：结构比较复杂（各触发器的输入由多个Q输出的组合逻辑得到），所用元件较多。



# 同步计数器总结



异步计数器



同步计数器



# 寄存器和计数器

---

- 寄存器 (Registers)
- 移位寄存器 (Shift Registers)
- 计数器 (Counters)
- 节拍发生器 (Beat Generator)

# 几个小问题

---

- 同步时序逻辑电路与异步时序逻辑电路的优缺点？
- 时序电路的分类？
- 什么是计数器？
- 环形计数器与扭环形计数器的特点和差异？
- 什么是节拍？什么是节拍发生器？
- 节拍发生器与计数器的关系？

# 典型时序逻辑部件——节拍发生器

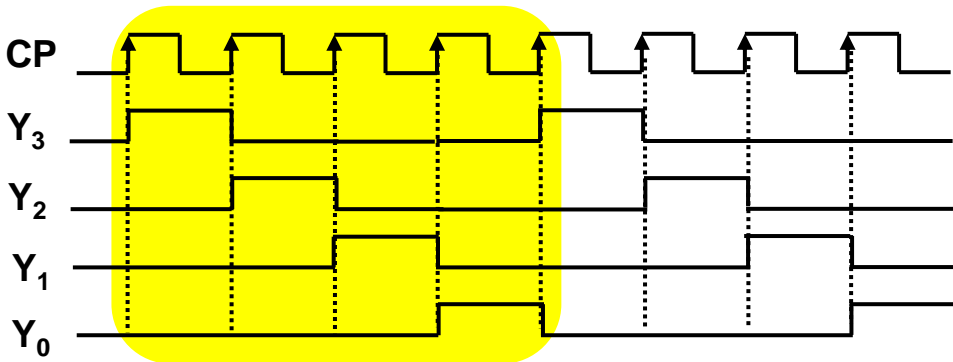
## □ 节拍发生器（时序发生器）——

### 定义

每个循环周期内, 在时钟脉冲的作用下, 产生一组在时间上有一定**先后顺序**的脉冲信号

### 作用

数字系统和计算机的控制部件利用顺序脉冲, 形成所需要的各种控制信号, 使某些设备按照事先规定的顺序进行运算或操作



例如:

执行  $\text{result} = A + 10;$



- |          |            |
|----------|------------|
| ①启动控制器工作 | ⑤取出操作数     |
| ②发送指令地址  | ⑥通知运算器计算   |
| ③取出指令    | ⑦发送保存结果的地址 |
| ④发送操作数地址 | ⑧保存结果      |

# 典型时序逻辑部件——节拍发生器

## □ 节拍发生器（时序发生器）——

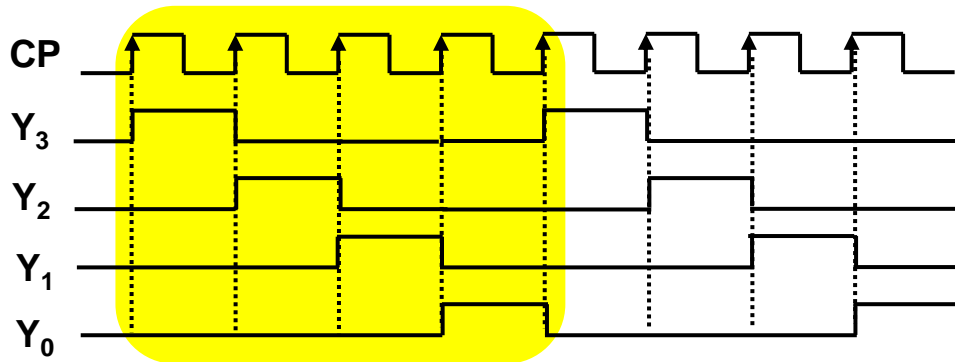
### 定义

每个循环周期内, 在时钟脉冲的作用下, 产生一组在时间上有一定**先后顺序**的脉冲信号

### 作用

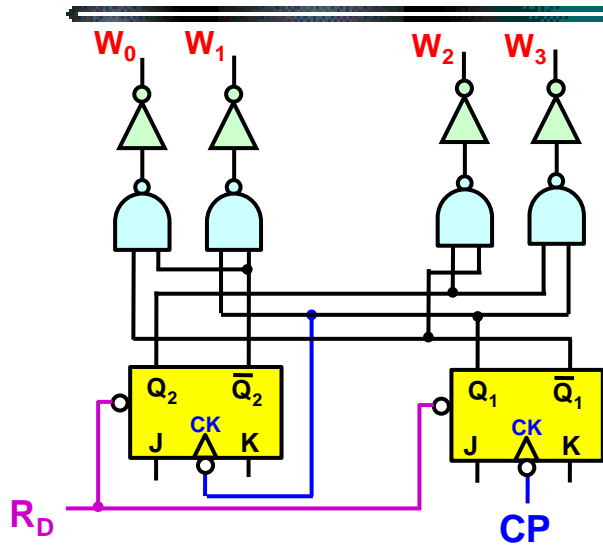
数字系统和计算机的控制部件利用顺序脉冲, 形成所需要的各种控制信号, 使某些设备按照事先规定的顺序进行运算或操作

例: 将4位二进制数 (如1000) 存入某寄存器, 然后将数据右移1位, 之后将数据读走, 再将右移后的数据左移1位。以上操作可以自动循环进行。



- ①执行写入操作: 写入使能有效 (存入1000)
- ②执行右移操作: 右移使能有效 (右移后0100)
- ③执行读出操作: 读出使能有效
- ④执行左移操作: 左移使能有效 (左移后1000)

# 节拍发生器1



## ① 输入方程

$$J_1 = K_1 = 1, CP_1 \downarrow$$

$$J_2 = K_2 = 1, CP_2 = Q_1 \downarrow$$

## ③ 输出方程

$$\begin{cases} W_0 = \bar{Q}_2 \bar{Q}_1 \\ W_1 = \bar{Q}_2 Q_1 \\ W_2 = Q_2 \bar{Q}_1 \\ W_3 = Q_2 Q_1 \end{cases}$$

结论：4-节拍发生器 ( $W_0 \sim W_3$ )

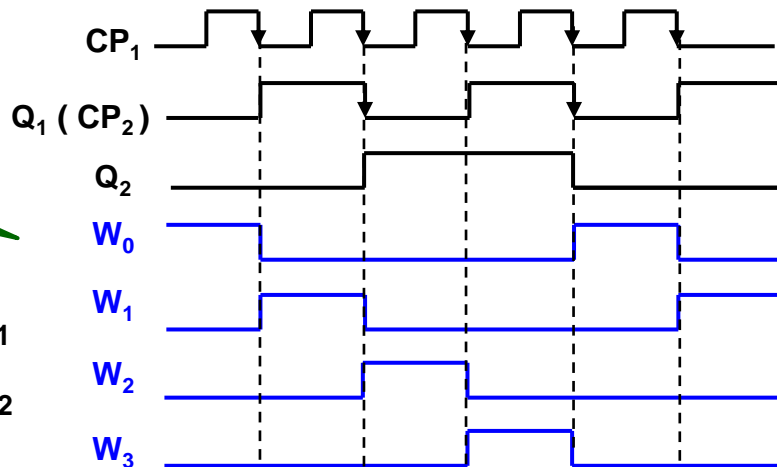
## ② 次态方程

$$Q_1^{n+1} = J_1 \bar{Q}_1 + K_1 \bar{Q}_1 = \bar{Q}_1$$

$$Q_2^{n+1} = J_2 \bar{Q}_2 + K_2 \bar{Q}_2 = \bar{Q}_2$$

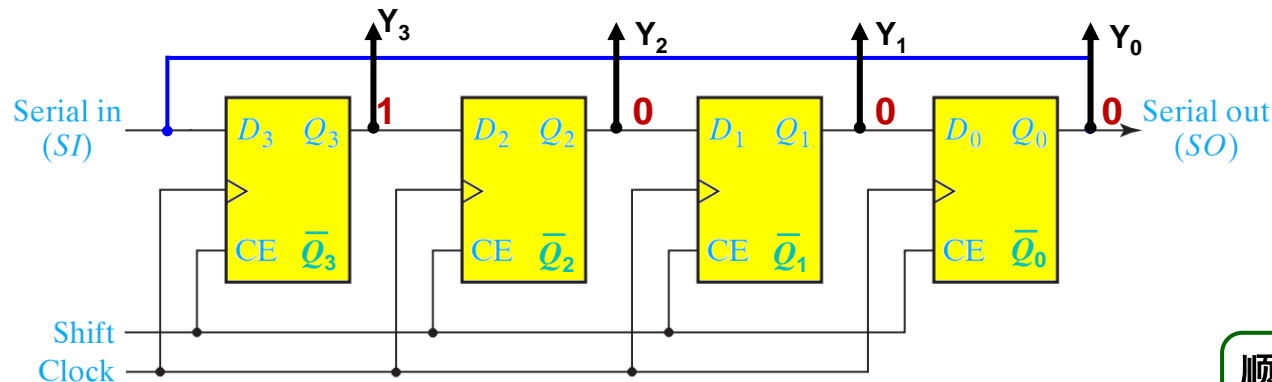
## ④ 状态转换表

现态		次态		时钟	
$Q_2^n$	$Q_1^n$	$Q_2^{n+1}$	$Q_1^{n+1}$	$CP_2$	$CP_1$
0	0	0	1	无	↓
0	1	1	0	↓	↓
1	0	1	1	无	↓
1	1	0	0	↓	↓

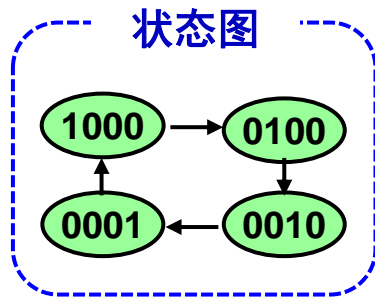


# 节拍发生器1

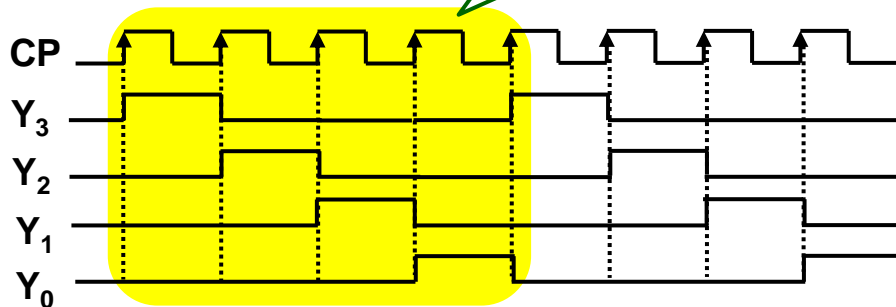
## 回顾：环形计数器



状态图

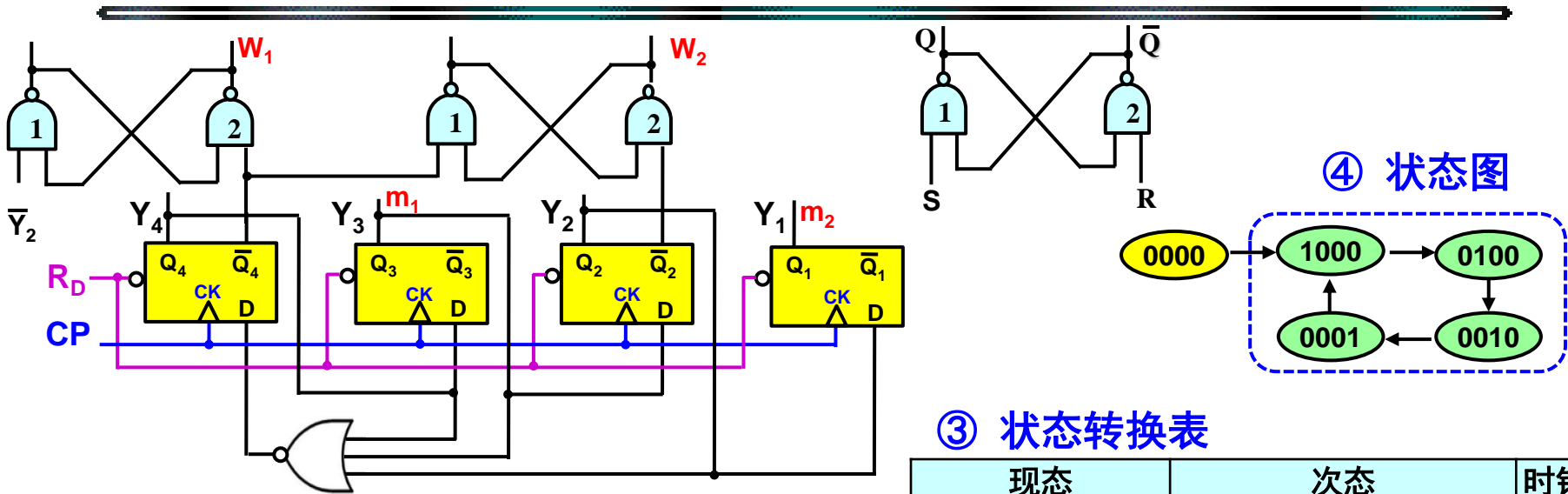


顺序脉冲发生器的波形



- 应用：电话响铃控制
- 用4位顺序脉冲发生器的某一个输出作为响铃控制信号，若时钟CP周期为1秒，电话铃声就是响1秒停3秒的节奏。

## 节拍发生器2



### ① 输入方程

$$\begin{cases} D_1 = Y_2 \\ D_2 = Y_3 \\ D_3 = Y_4 \\ D_4 = \overline{Y_4 + Y_3 + Y_2} \end{cases}$$

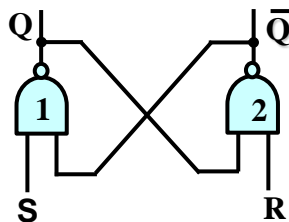
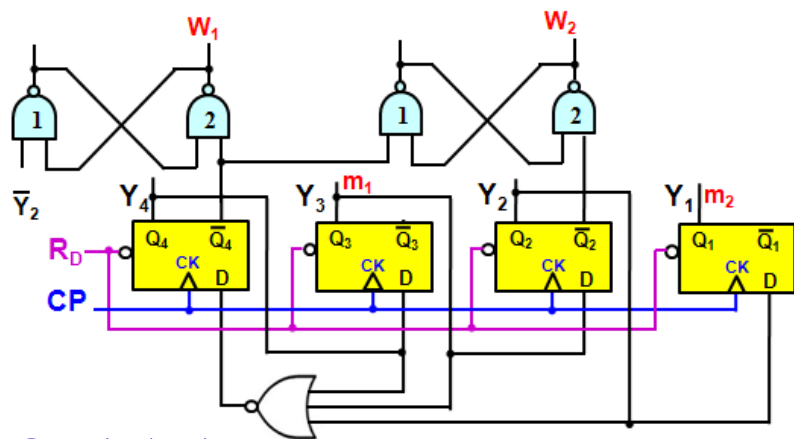
## ② 次态方程

$$\begin{aligned} Y_1^{n+1} &= Y_2 \\ Y_2^{n+1} &= Y_3 \\ Y_3^{n+1} &= Y_4 \\ Y_4^{n+1} &= \overline{Y_4 + Y_3 + Y_2} \end{aligned}$$

### ③ 状态转换表

现态				次态				时钟
$Y_4^n$	$Y_3^n$	$Y_2^n$	$Y_1^n$	$Y_4^{n+1}$	$Y_3^{n+1}$	$Y_2^{n+1}$	$Y_1^{n+1}$	CP
0	0	0	0	1	0	0	0	↑
1	0	0	0	0	1	0	0	↑
0	1	0	0	0	0	1	0	↑
0	0	1	0	0	0	0	1	↑
0	0	0	1	1	0	0	0	↑

# 节拍发生器2



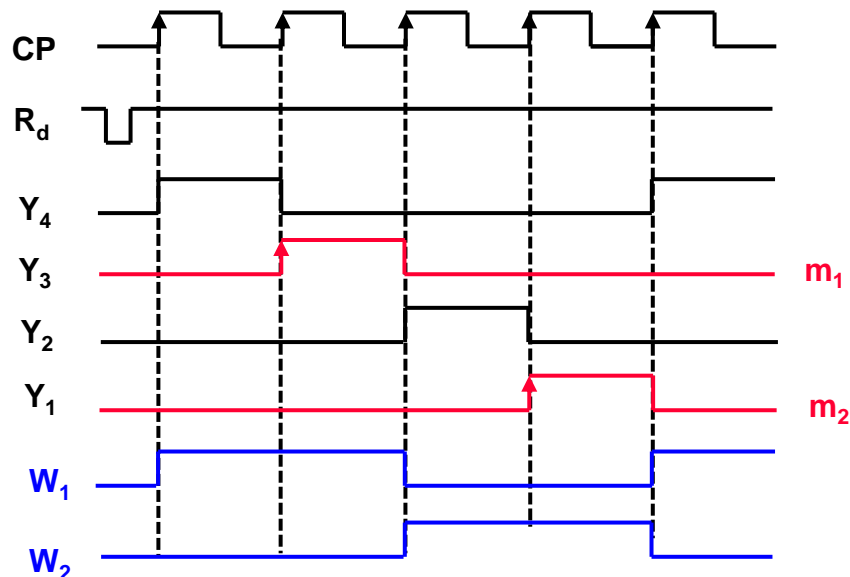
结论：2-节拍发生器

- $W_1\_m_1$ : 节拍电位\_节拍脉冲
- $W_2\_m_2$ : 节拍电位\_节拍脉冲

## ⑤ 确定输出

R	S	$Q_{n+1}$	$\bar{Q}_{n+1}$
$\bar{Y}_4$	$\bar{Y}_2$	$(W_1=\bar{Q})$	
1	1	$Q_n$	$\bar{Q}_n$
0	1	0	1
1	0	1	0
0	0	—	—

R	S	$Q_{n+1}$	$\bar{Q}_{n+1}$
$\bar{Y}_2$	$\bar{Y}_4$	$(W_2=\bar{Q})$	
1	1	$Q_n$	$\bar{Q}_n$
0	1	0	1
1	0	1	0
0	0	—	—





# 寄存器和计数器

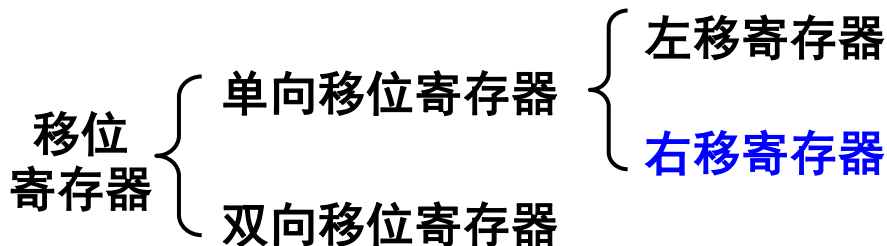
---

- 寄存器 (Registers)
- 移位寄存器 (Shift Registers)
- 计数器 (Counters)
- 节拍发生器 (Beat Generator)

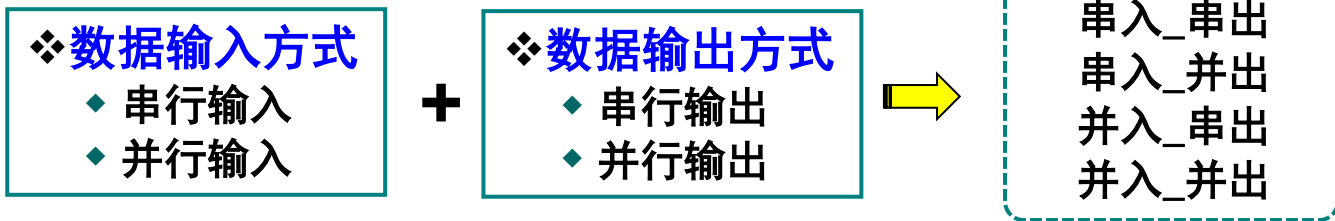
# 移位寄存器定义和分类

- 每来一个时钟脉冲，寄存器里存储的数据，能依次左移或右移1位。
- 可以实现代码的串、并行转换、数值运算和数据处理等。

## ➤ 分类



## ➤ 工作方式



# 用Verilog实现串入并出8位移位寄存器

---

```
module Vr8bitSRparout ( CLK, CLR, SERIN, Q
);
    input CLK, CLR, SERIN;
    output reg [WID-1:0] Q;
    parameter WID = 8;

    always @ (posedge CLK)
        if (CLR == 1) Q <= 0;      // 同步清零
        else Q <= {Q[WID-2:0], SERIN}; // 移位
endmodule
```

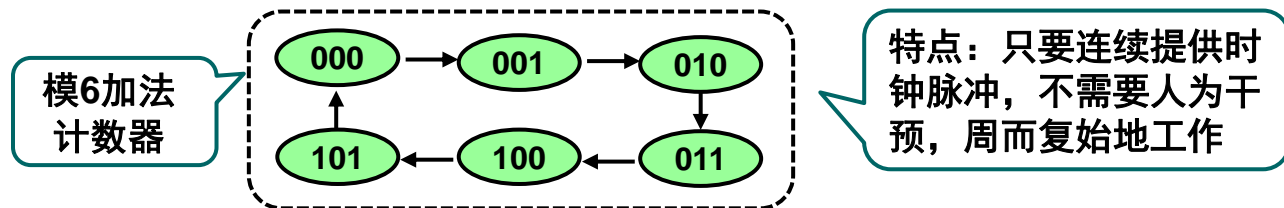
# 用Verilog实现通用4位移位寄存器

---

```
module Vrshrg4u (  
input CLK, CLR, RIN, LIN, S0, S1, A, B, C, D,  
output reg QA,QB,QC,QD);  
    always @ (posedge CLK)  
        if (CLR == 1'b1) {QA,QB,QC,QD} <= 4'd0;           //同步清零  
        else case ({S1,S0})  
            2'b00: {QA,QB,QC,QD} <= {QA,QB,QC,QD} ;           // 保持  
            2'b01: {QA,QB,QC,QD} <= {RIN,QA,QB,QC}; // 右移  
            2'b10: {QA,QB,QC,QD} <= {QB,QC,QD,LIN}; // 左移  
            2'b11: {QA,QB,QC,QD} <= {A,B,C,D};           // 载入  
            default: {QA,QB,QC,QD} <= {QA,QB,QC,QD} ;           // 不会发生  
        endcase  
endmodule
```

# 典型时序逻辑部件——计数器

计数器是一种能在输入信号作用下依次通过预定状态的时序逻辑电路，是数字系统和计算机广泛使用的逻辑器件，可用于计数、分频、定时、控制、产生节拍脉冲（顺序脉冲）和序列脉冲等。



- 由一组触发器构成，计数器中的“数”是用触发器的状态组合来表示的。
- 计数器在运行时，所经历的状态是周期性的，总是在有限个状态中循环。
- 将一次循环所包含的**状态总数**称为计数器的“**模**”，记为N,包含n个触发器的最大模值  $N = 2^n$ 。
- 把作用于计数器的时钟脉冲称为计数脉冲，用 **CP** (或**CLK**)表示。

# 用Verilog实现4位通用二进制计数器

---

```
module Vrcntr4u (  
    input CLK, CLR, LD, ENP, ENT,  
    input [3:0] D,  
    output reg [3:0] Q,  
    output reg RCO);  
    always @ (posedge CLK) // 创建f-f计数器的特性  
        if (CLR)                Q <= 4'd0;  
        else if (LD)             Q <= D;  
        else if (ENT && ENP)     Q <= Q + 1; //ENP,ENT有效且CLR, LD无效, 计数  
        else Q <= Q;  
    always @ (*) // 创建组合输出RCO, 便于级联  
        if (ENT && (Q == 4'd15)) RCO = 1;  
        else RCO = 0;  
endmodule
```

# 用Verilog实现十进制通用计数器

---

```
module Vrcntr4udec ( input CLK, CLR, LD, ENP, ENT,  
    input [3:0] D,  
    output reg [3:0] Q,  
    output reg RCO);  
    always @ (posedge CLK) // 创建f-f计数器的特性  
        if (CLR)          Q <= 4'd0;  
        else if (LD)       Q <= D;  
        else if (ENT && ENP && (Q == 4'd9)) Q <= 4'd0;  
        else if (ENT && ENP) Q <= Q + 1;  
        else Q <= Q;  
  
    always @ (*) // 创建组合输出RCO  
        if (ENT && (Q == 4'd9)) RCO = 1;  
        else RCO = 0;  
endmodule
```

# 用Verilog实现4位**递增/递减**的计数器

```
module Vrupdn4 (  
    input CLK, CLR, LD, ENP, ENT, UPDN,  
    input [3:0] D,  
    output reg [3:0] Q,  
    output reg RCO);  
  
    always @ (posedge CLK)  
        if (CLR)          Q <= 4'b0;  
        else if (LD)       Q <= D;  
        else if (ENT && ENP && UPDN)      Q <= Q + 1; //UPDN为1, 递增计数  
        else if (ENT && ENP && !UPDN)    Q <= Q - 1;  
        else Q <= Q;  
  
    always @ (*) // 创建组合输出RCO  
        if (ENT && UPDN && (Q == 4'd15)) RCO = 1; //递增计数到15进位  
        else if (ENT && !UPDN && (Q == 4'd0)) RCO = 1;  
        else RCO = 0;  
endmodule
```