

数字逻辑设计

Digital Logic Design

高翠芸

School of Computer Science

gaocuiyun@hit.edu.cn

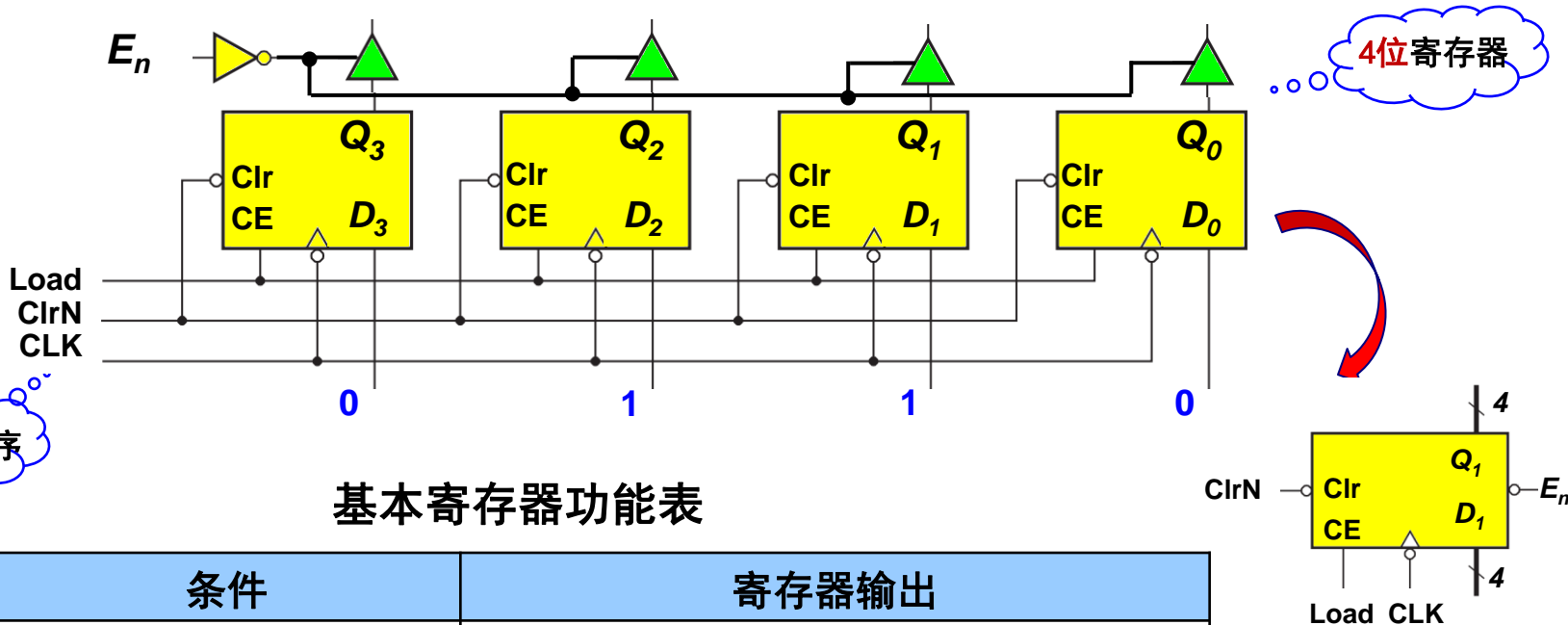
Unit 9 Registers and Counters



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

基本寄存器

- 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。

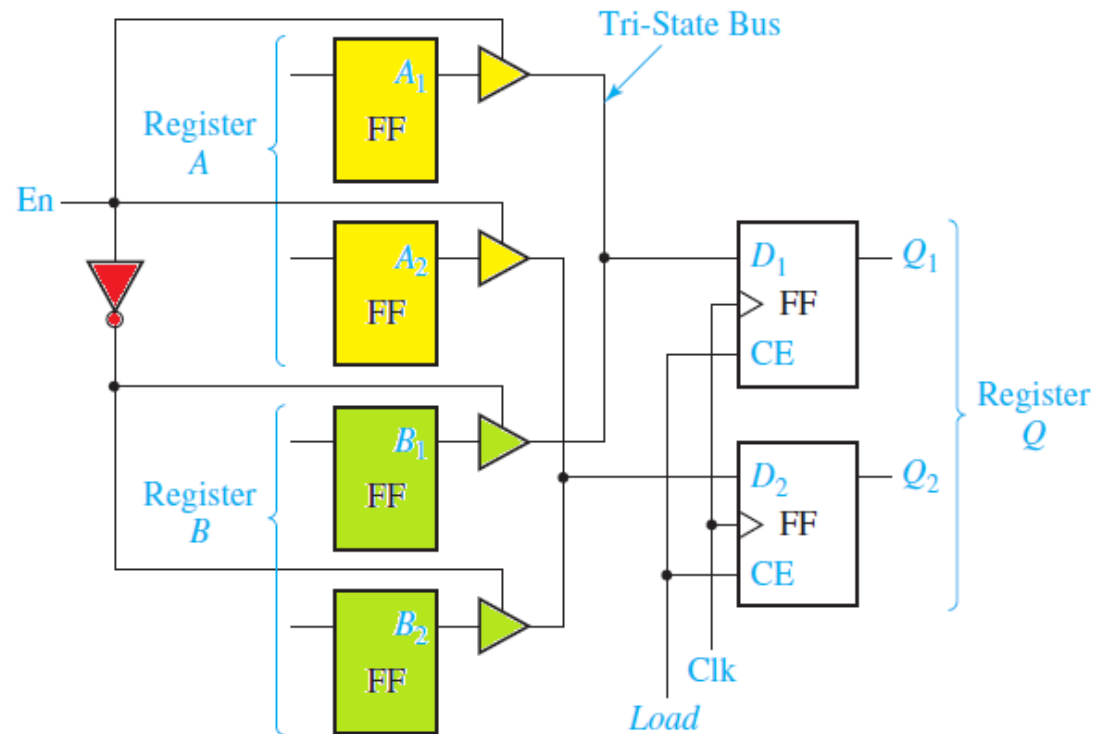


基本寄存器功能表

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	$ClrN=1$, $Load=1$, $clk \downarrow$	$Q_3Q_2Q_1Q_0=D_3D_2D_1D_0$
读出	$En=0$	$Q_3Q_2Q_1Q_0=D_3D_2D_1D_0$

基本寄存器

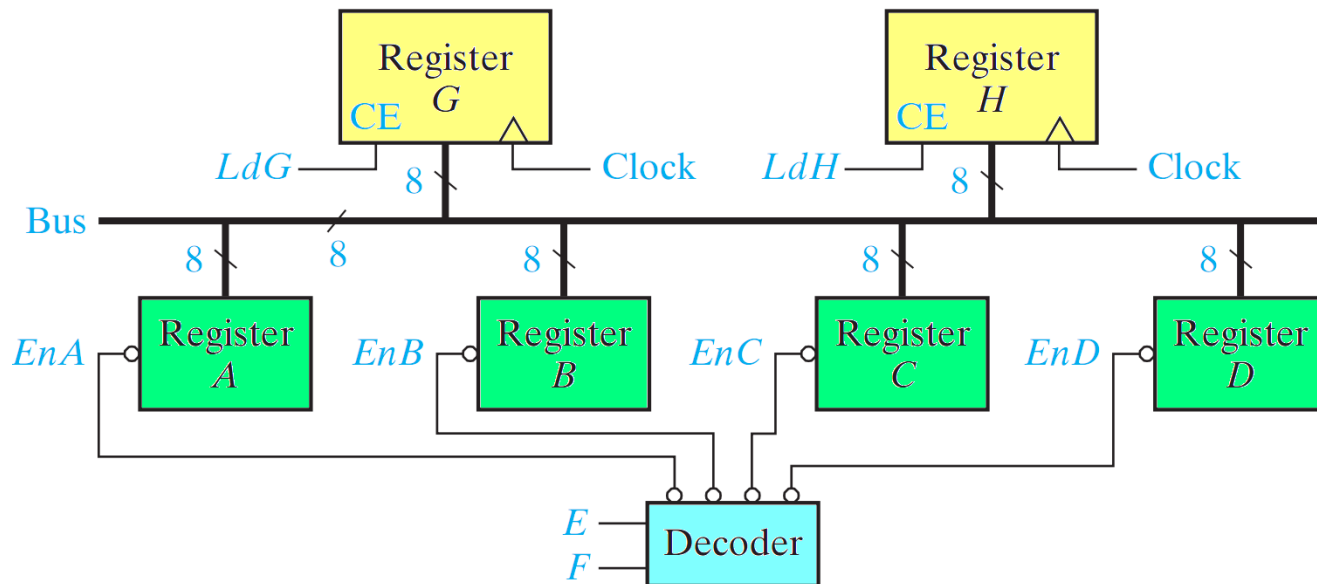
Transferring data between registers



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

基本寄存器

- 利用三态总线进行数据传送

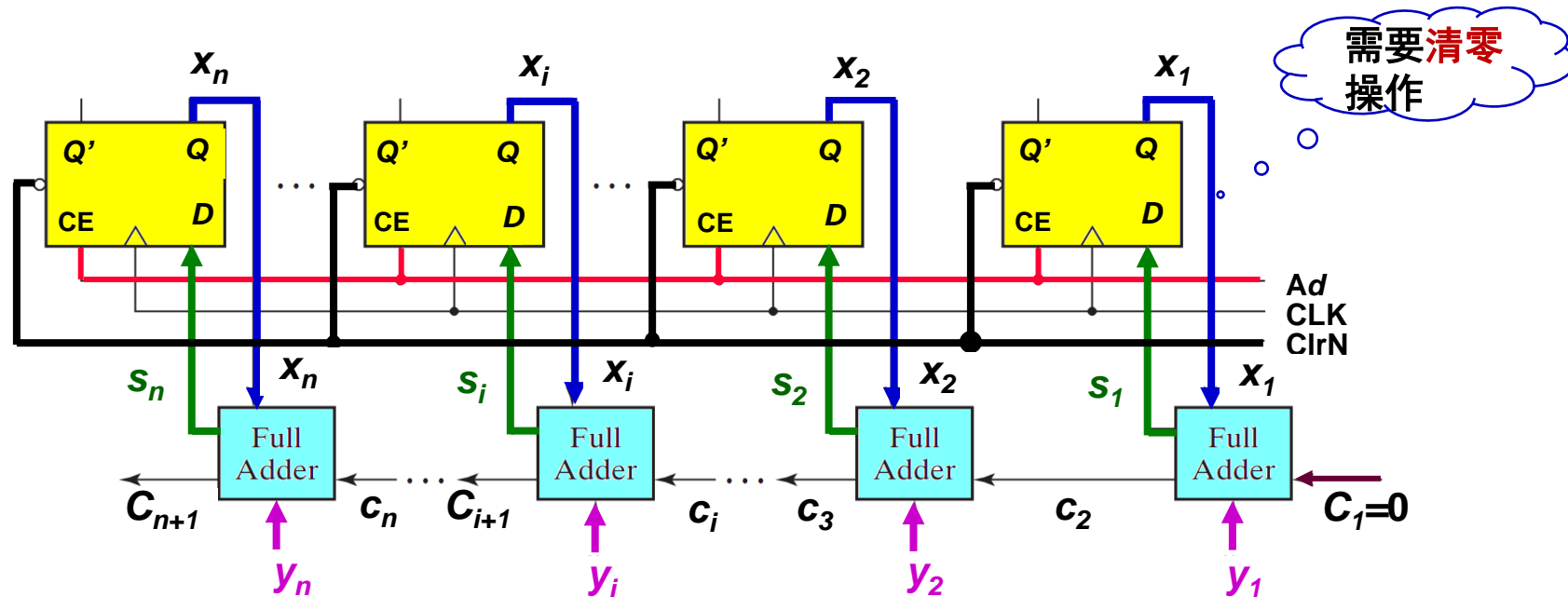


- Register A to G, H: $EF=00$, 且 $LdG=1, LdH=1, clk \uparrow$
- Register B to G, H: $EF=01$, 且 $LdG=1, LdH=1, clk \uparrow$

基本寄存器

■ 具有累加功能的并行加法器1

$$X = X + Y$$

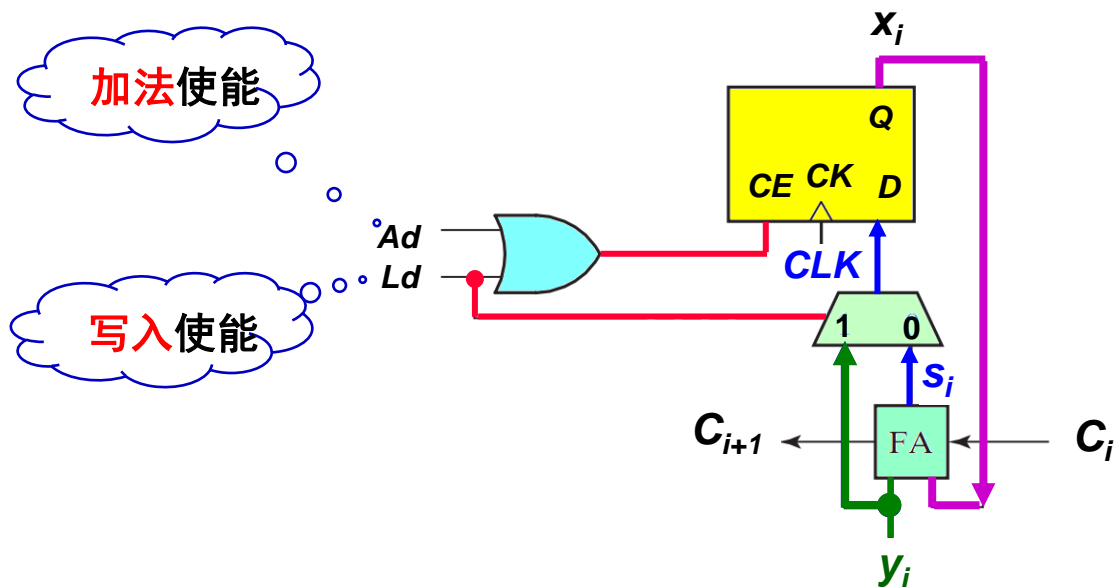


1. 初始化清零: $ClrN=0$, 则 $Q_n \dots Q_0=0$, 即 $X_n \dots X_0=0$
2. $ClrN=1$, 将 y_i 送到全加器输入端
3. 执行 $S_i = y_i + x_i$
4. 存储累加和: $ClrN=1, Ad=1, CLK \uparrow$ 到来时, 寄存器 $Q_i = S_i$

基本寄存器

■ 具有累加功能的并行加法器2

$$X = X + Y$$



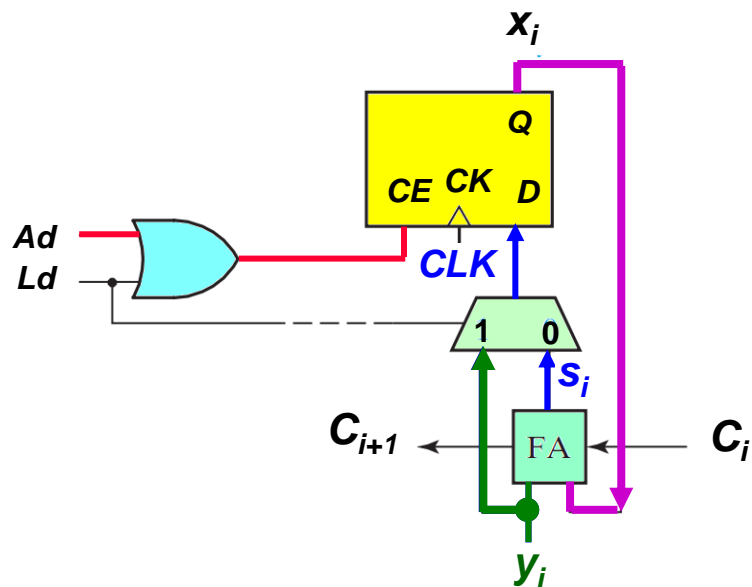
■ 初始化:

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 $S_i = y_i + x_i$

基本寄存器

■ 具有累加功能的并行加法器2



■ 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 $S_i=y_i+x_i$

■ $Ld=0, Ad=1, ck \uparrow$: Then $x_i=s_i$

■ 保持: $Ld=0, Ad=0$

Unit 9 Registers and Counters

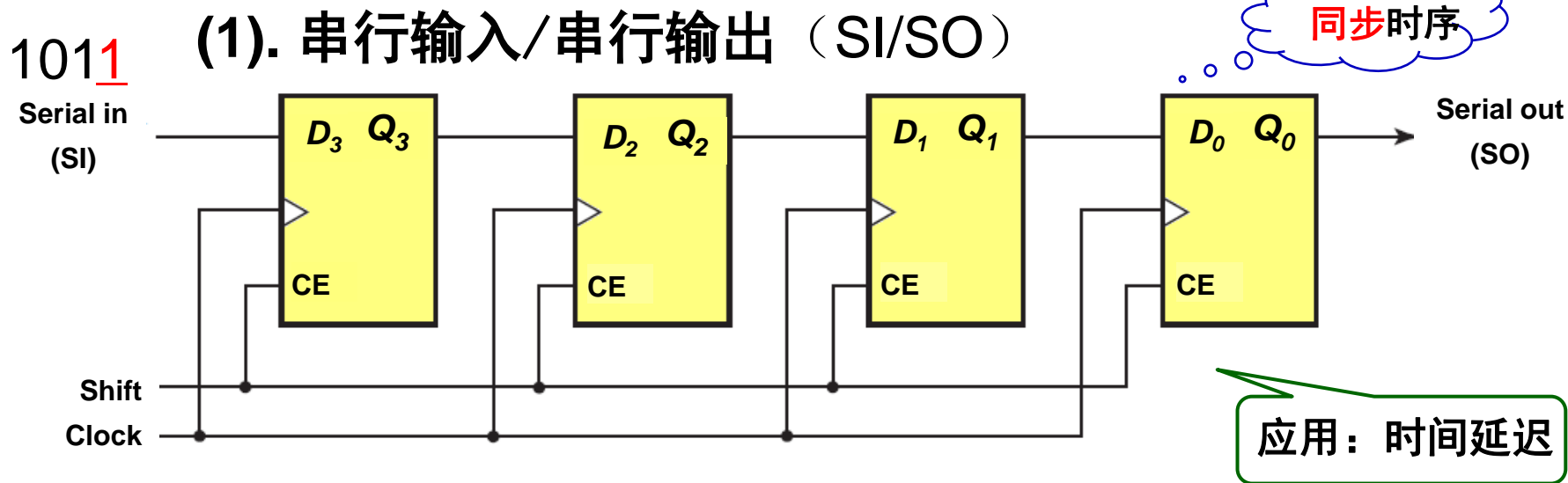
- 基本寄存器 (Registers)
- 👉 ■ 移位寄存器 (Shift Registers)
- 计数器 (Counters)
- 节拍发生器 (Beat Generator)

移位寄存器 (Shift Registers)

■ 单向移位寄存器——

- 寄存器里存储的数据在**移位脉冲**的作用下依次**左移**或**右移**。
- 可以实现代码的**串行→并行转换**、数值运算和数据处理等。

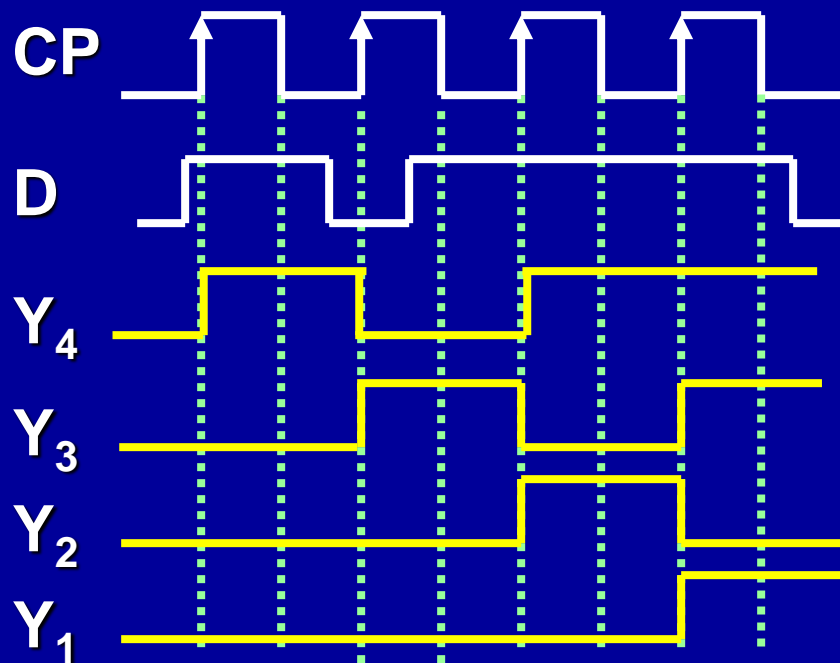
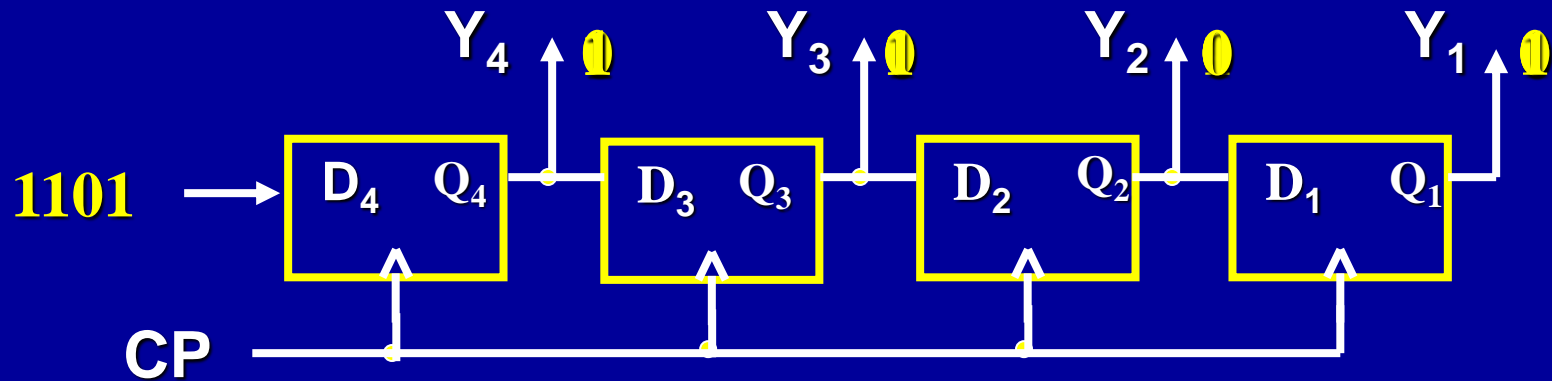
1. 右移寄存器 (Right-Shift Register)



串行输出：最后一个触发器的输出作为整个电路的输出。

右移：数据从串行输入端送入，应该先送**最低位**。

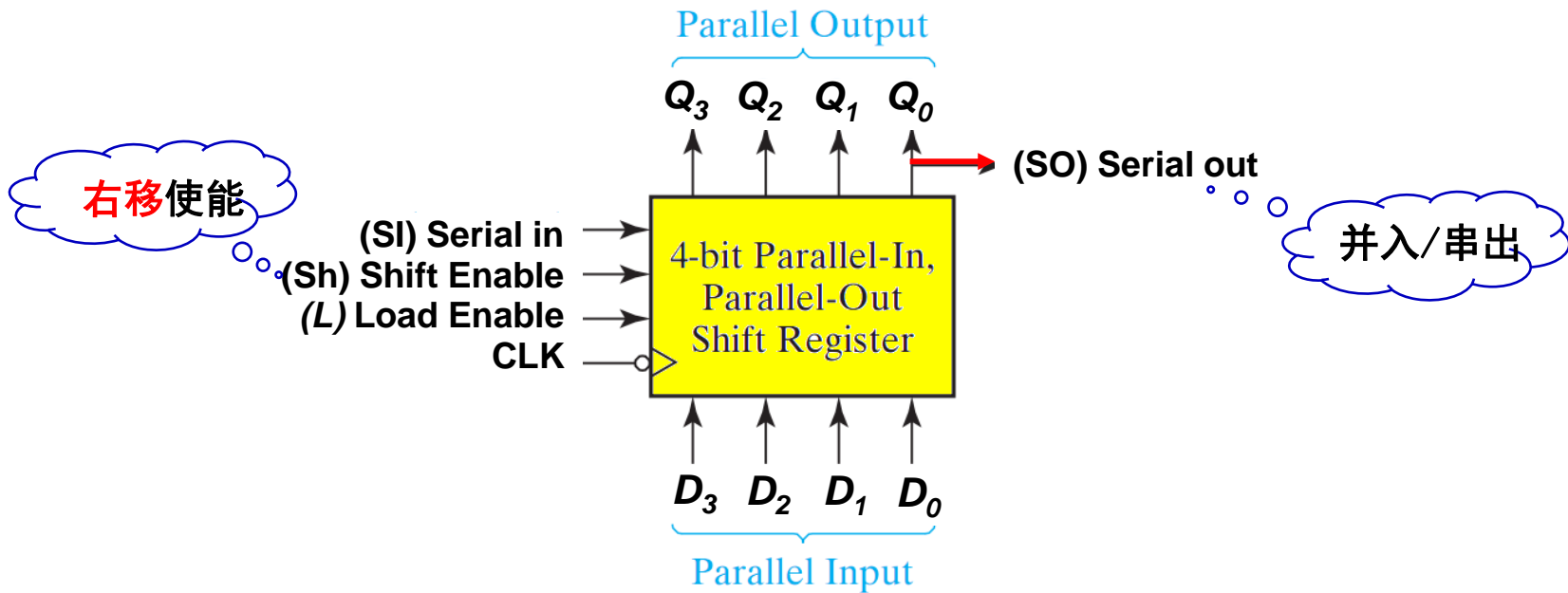
(2). 串入/并出 (Serial in / Parallel out) S/P signal convertor



CP	D_1	Y_4	Y_3	Y_2	Y_1
		0	0	0	0
	1	1	0	0	0
	0	0	1	0	0
	1	1	0	1	0
	1	1	1	0	1

移位寄存器 (Shift Registers)

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



Inputs		Next State				Action
Sh (Shift)	L (Load)	Q_3^+	Q_2^+	Q_1^+	Q_0^+	
0	0	Q_3	Q_2	Q_1	Q_0	No change
0	1	D_3	D_2	D_1	D_0	Load
1	X	SI	Q_3	Q_2	Q_1	Right shift

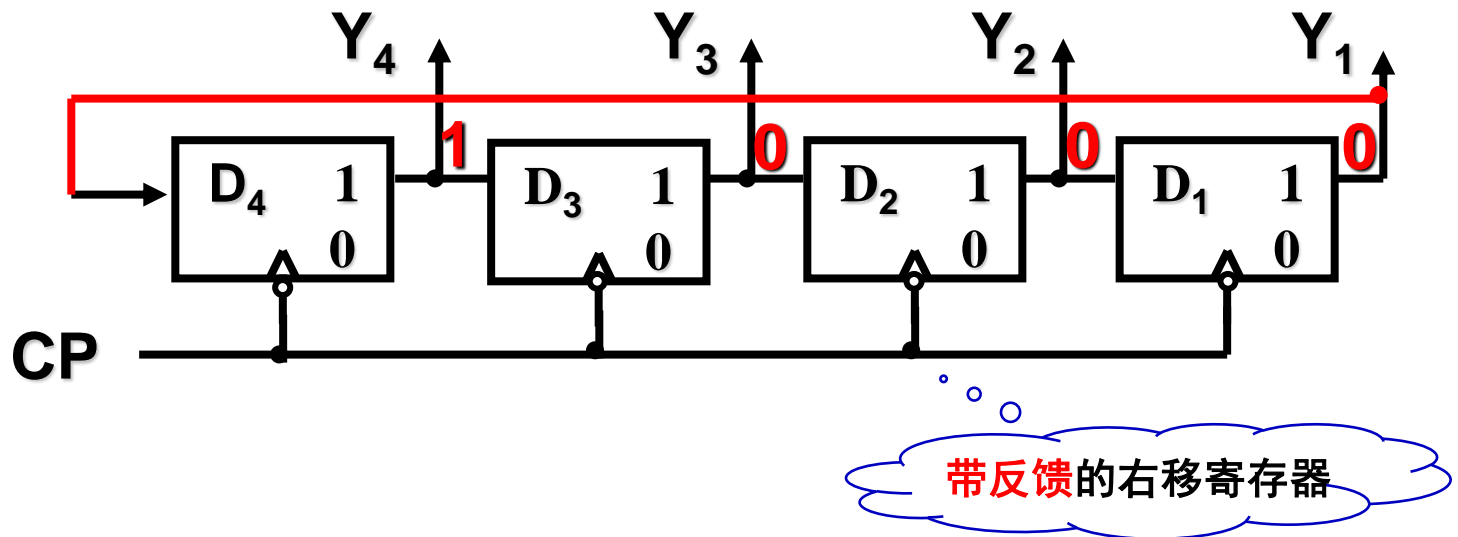
移位寄存器 (Shift Registers)

2. Applications——

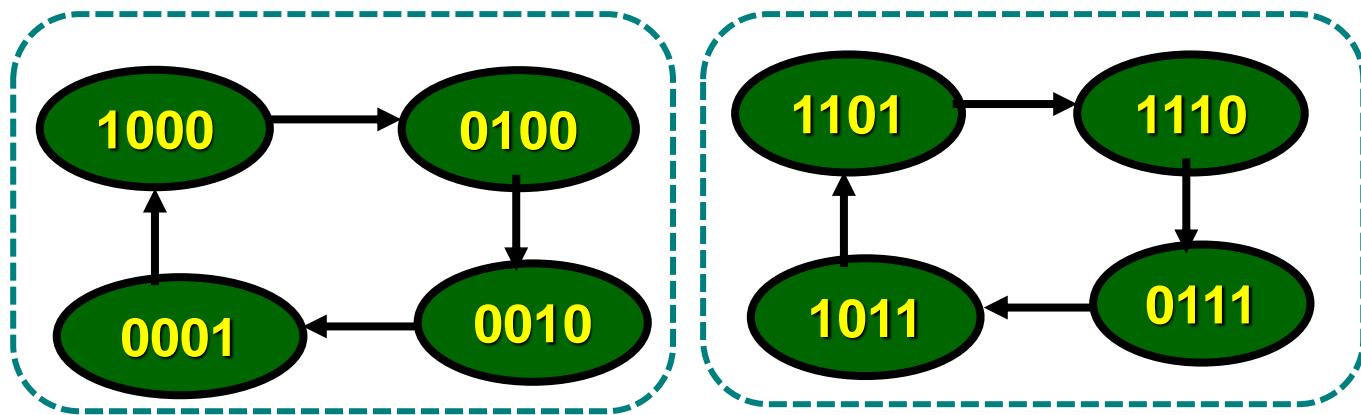
(1). 环形计数器 (Ring Counter)

- 计数器:

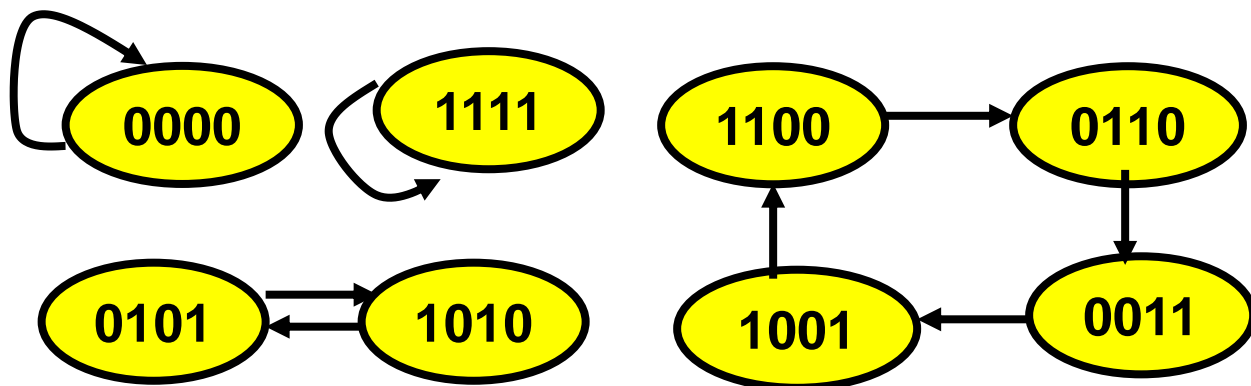
一种能在输入信号的作用下依次循环通过预定状态的时序逻辑电路。



常用状态图



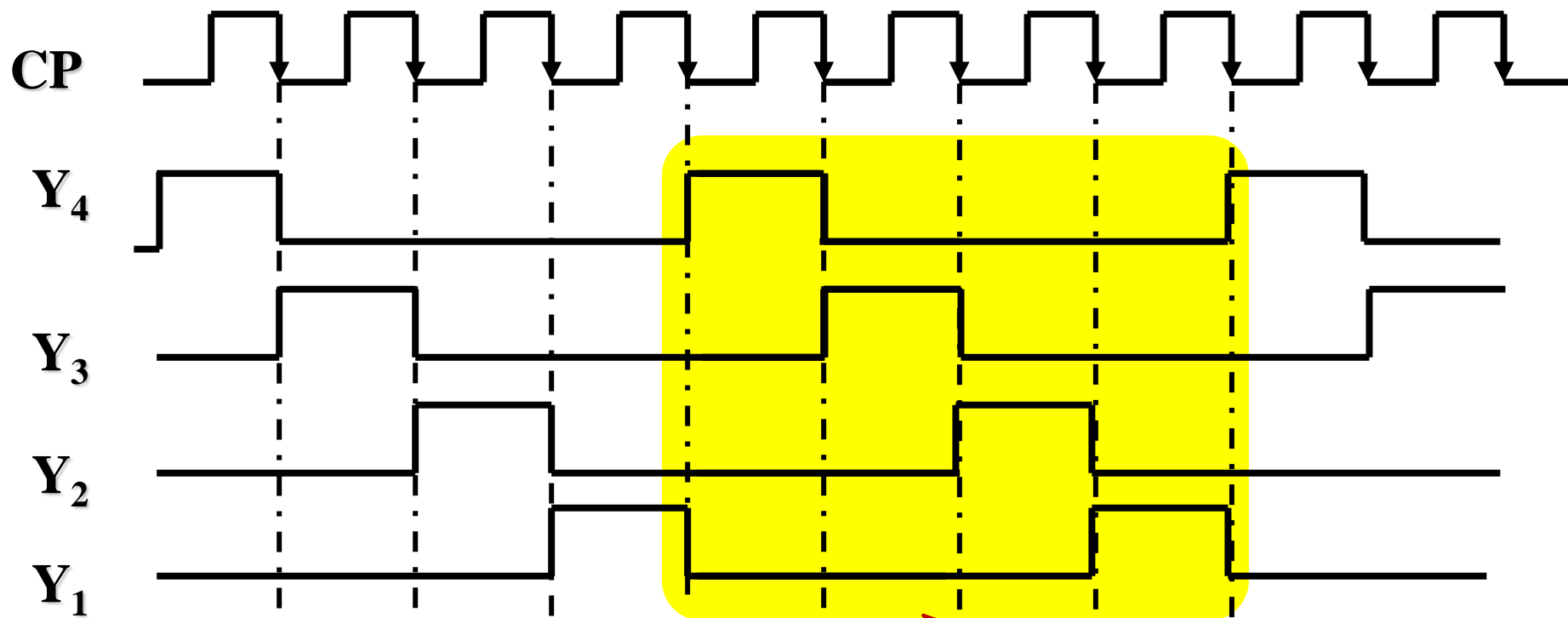
不常用状态图



- **优点:** 电路简单, 输出具有**二进制译码器**的特点
- **缺点:** 只使用了 **n** 个状态 (total states: 2^n)

不能**自启动**, 需要**预置**

移位寄存器 (Shift Registers)

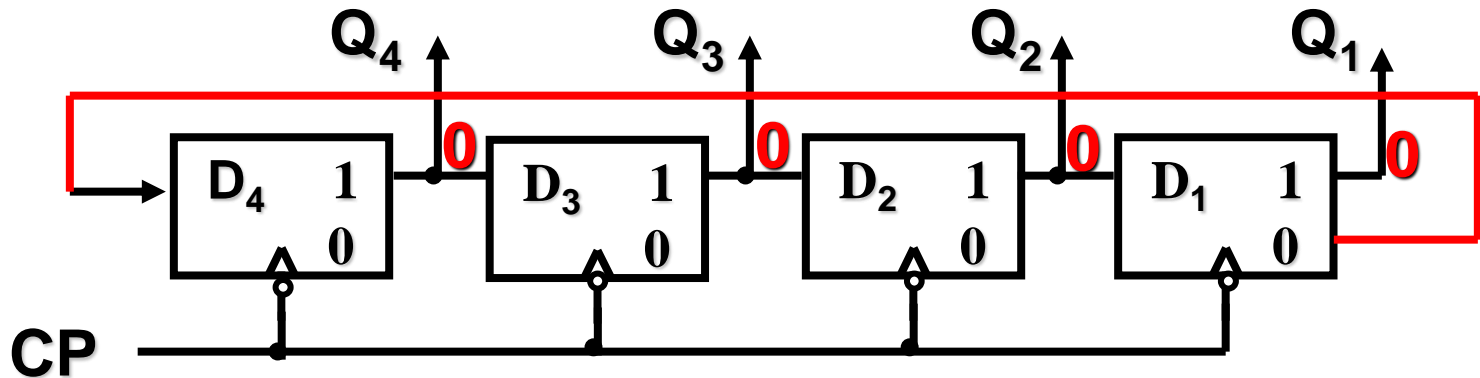


特点：一个周期
之内的波形与译
码器输出相同

移位寄存器 (Shift Registers)

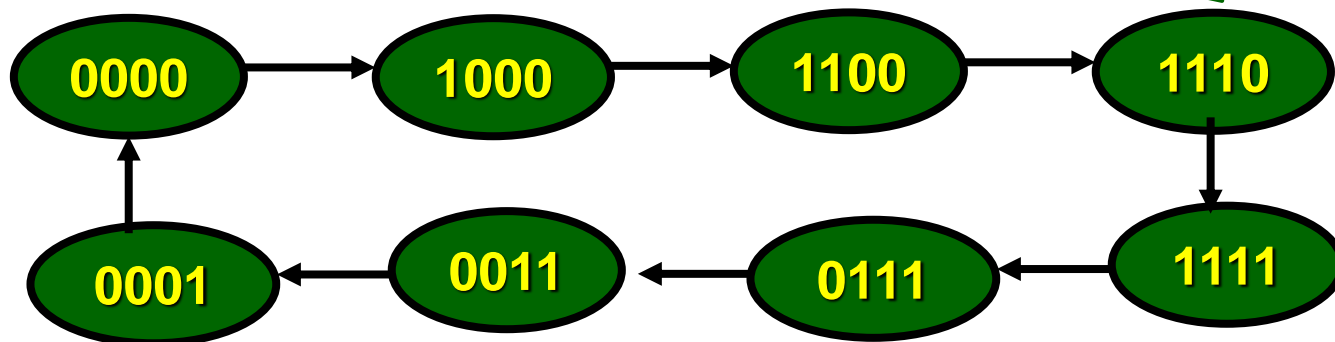
2. Applications

(2). **扭**环形计数器 (Johnson Counter / Twisted Ring Counter)



带反馈的右移寄存器

常用状态图

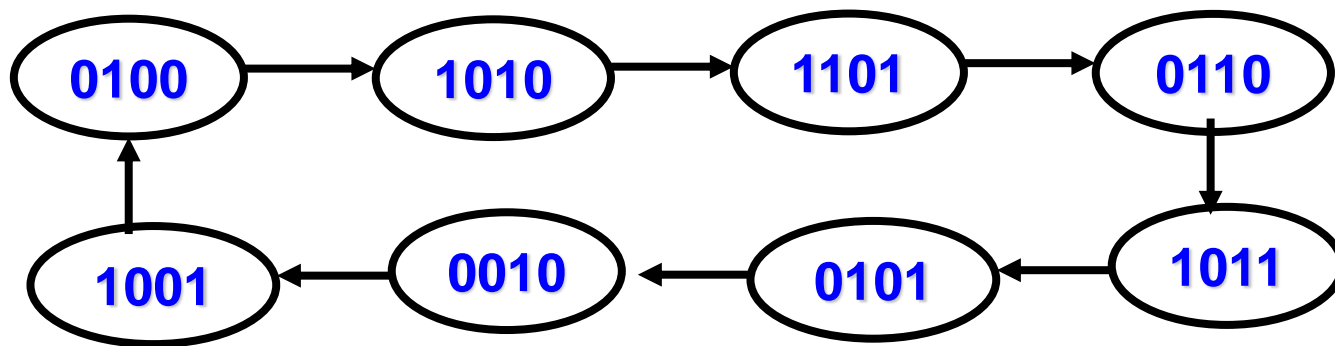


只使用了 $2n$ 个状态；
不能自启动，需要预置

模8计数器

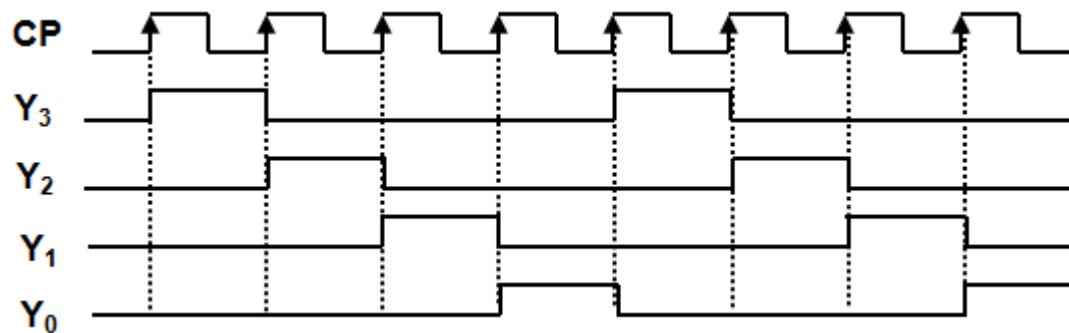
电路具有格雷码的特点

不常用状态图

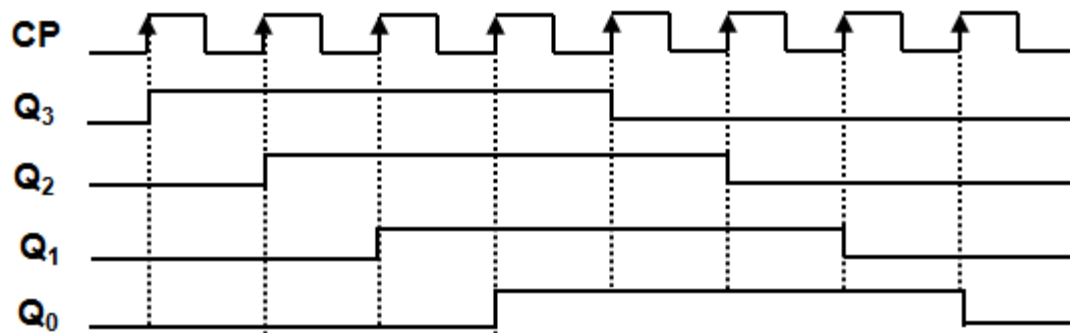


移位寄存器 (Shift Registers)

环形计数器波形图



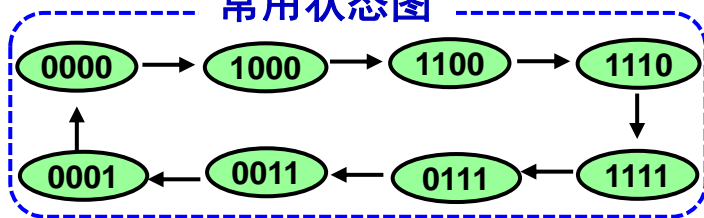
扭环形计数器波形图



扭环形计数器构成节拍发生器

4位扭环形计数器可构成**8节拍发生器** $T_0 \sim T_7$ ，但需加译码电路。

常用状态图



$Q_3 \backslash Q_2$	00	01	11	10
00	1	0	0	X
01	X	X	0	X
11	0	X	0	0
10	0	X	X	X

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

$Q_3 \backslash Q_2$	00	01	11	10
00	0	0	0	X
01	X	X	0	X
11	0	X	0	0
10	1	X	X	X

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

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

优点:

- **无险象:** 由于电路在每次状态转换时，只有一位触发器改变状态，电路译码时不会产生**竞争冒险**现象
- 后级每个译码门只需要**2**个输入端

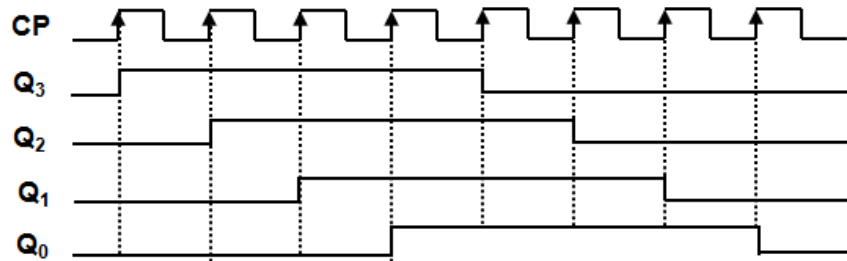
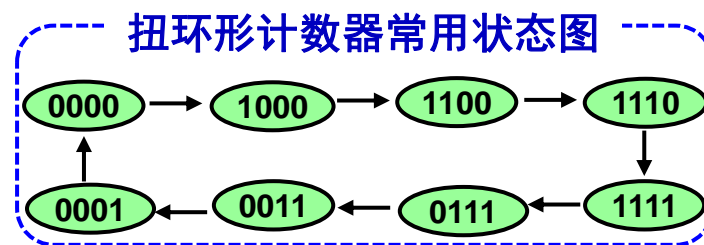
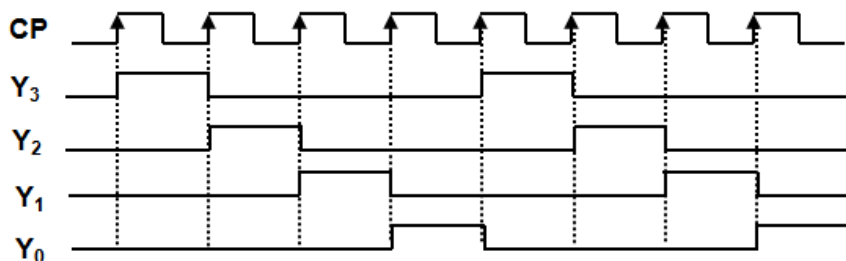
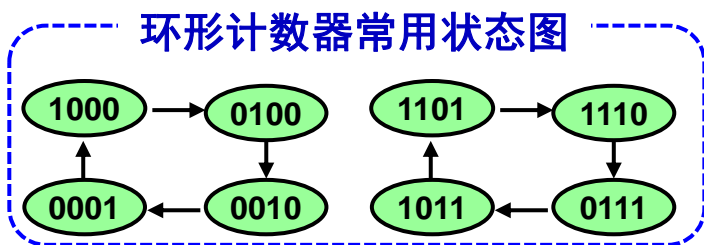
移位寄存器 (Shift Registers)

□ 环形、扭环形计数器总结——

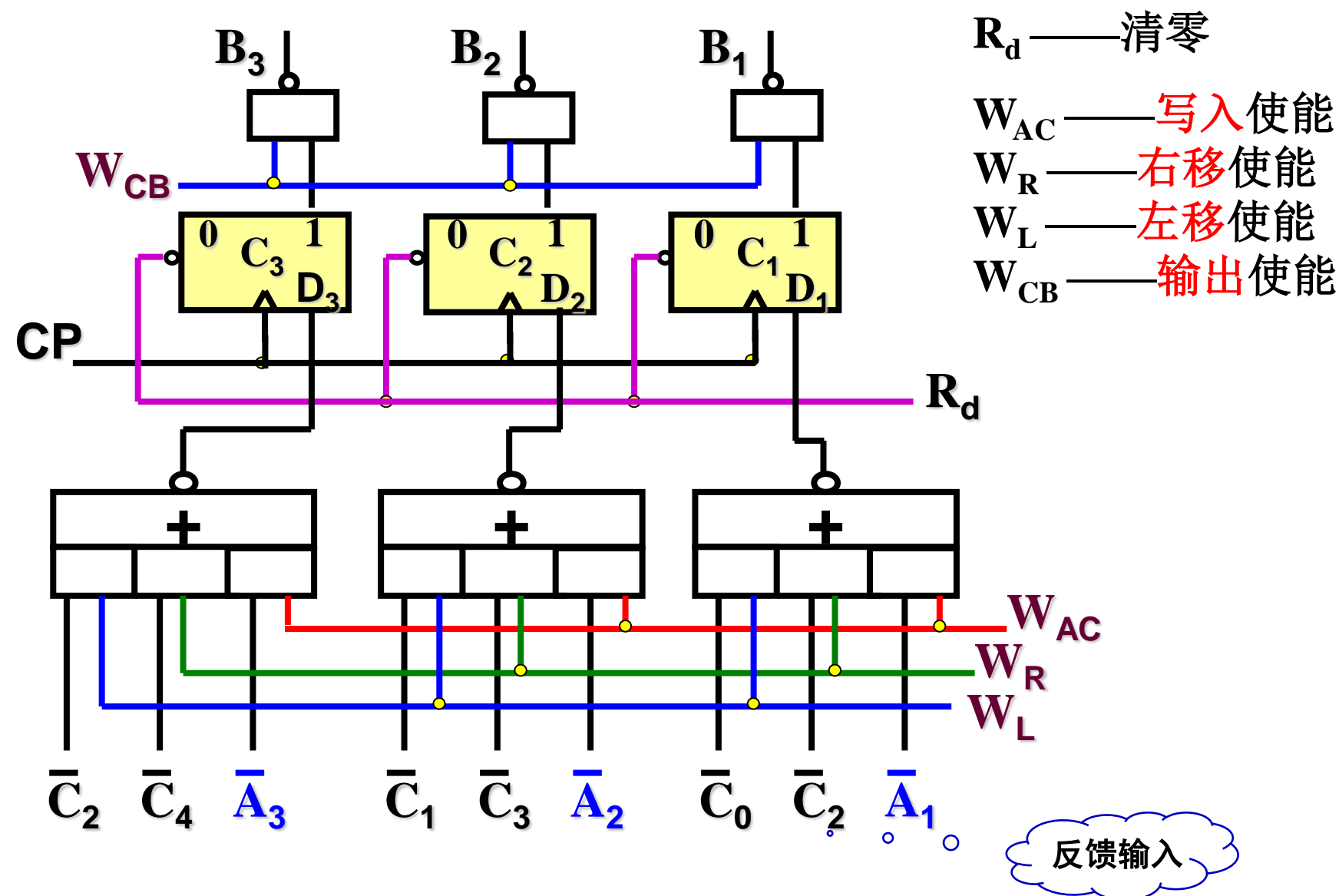
特点：在移位寄存器的基础上，增加**反馈**逻辑电路组成。

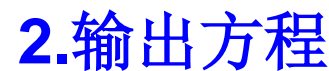
用途：

- 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**）



双向移位寄存器





$$\mathbf{B}_3 = \overline{\mathbf{C}_3} \mathbf{W}_{CB}$$

$$\mathbf{B}_2 = \overline{\mathbf{C}_2} \mathbf{W}_{CB}$$

$$\mathbf{B}_1 = \overline{\mathbf{C}_1 \mathbf{W}_{CB}}$$

3.次态方程

$$\mathbf{C}_3^{n+1} = \mathbf{D}_3$$

$$\mathbf{C}_2^{n+1} = \mathbf{D}_2$$

$$\mathbf{C}_1^{n+1} = \mathbf{D}_1$$

$$\mathbf{D}_3 = \overline{\mathbf{A}}_3 \mathbf{W}_{AC} + \overline{\mathbf{C}}_4 \mathbf{W}_R + \overline{\mathbf{C}}_2 \mathbf{W}_L$$

$$\mathbf{D}_2 = \overline{\mathbf{A}}_2 \mathbf{W}_{AC} + \overline{\mathbf{C}}_3 \mathbf{W}_R + \overline{\mathbf{C}}_1 \mathbf{W}_L$$

$$\mathbf{D}_1 = \overline{\mathbf{A}}_1 \mathbf{W}_{AC} + \overline{\mathbf{C}}_2 \mathbf{W}_R + \overline{\mathbf{C}}_0 \mathbf{W}_L$$

1.输入方程

双向移位寄存器

(1) 写入：将 $A_1 \sim A_3$ 存放在寄存器中

Let: $W_{AC} = 1$, $W_R = W_L = 0$

输入方程

$$D_3 = \overline{A_3} \circ 1 + \overline{C_4} \circ 0 + \overline{C_2} \circ 0 = A_3$$

$$D_2 = \overline{A_2} \circ 1 + \overline{C_3} \circ 0 + \overline{C_1} \circ 0 = A_2$$

$$D_1 = \overline{A_1} \circ 1 + \overline{C_2} \circ 0 + \overline{C_0} \circ 0 = A_1$$

When $cp \uparrow$

次态方程

$$C_3^{n+1} = D_3 = A_3$$

$$C_2^{n+1} = D_2 = A_2$$

$$C_1^{n+1} = D_1 = A_1$$

双向移位寄存器

(2) 右移

Let: $W_R = 1$, $W_L = W_{AC} = 0$

When $cp \uparrow$

输入方程

$$\begin{aligned} C_3^{n+1} = D_3 &= \overline{A_3} \circ 0 + \overline{C_4} \circ 1 + \overline{C_2} \circ 0 = C_4 \\ C_2^{n+1} = D_2 &= \overline{A_2} \circ 0 + \overline{C_3} \circ 1 + \overline{C_1} \circ 0 = C_3 \\ C_1^{n+1} = D_1 &= \overline{A_1} \circ 0 + \overline{C_2} \circ 1 + \overline{C_0} \circ 0 = C_2 \end{aligned}$$

次态方程

双向移位寄存器

(3) 左移

Let: $W_L=1$, $W_R = W_{AC}=0$

When $cp \uparrow$

输入方程

$$\begin{aligned} C_3^{n+1} = D_3 &= \overline{A_3} \cdot 0 + \overline{C_4} \cdot 0 + \overline{C_2} \cdot 1 = C_2 \\ C_2^{n+1} = D_2 &= \overline{A_2} \cdot 0 + \overline{C_3} \cdot 0 + \overline{C_1} \cdot 1 = C_1 \\ C_1^{n+1} = D_1 &= \overline{A_1} \cdot 0 + \overline{C_2} \cdot 0 + \overline{C_0} \cdot 1 = C_0 \end{aligned}$$

次态方程

双向移位寄存器

(4) 读出

Let: $W_{CB} = 1$

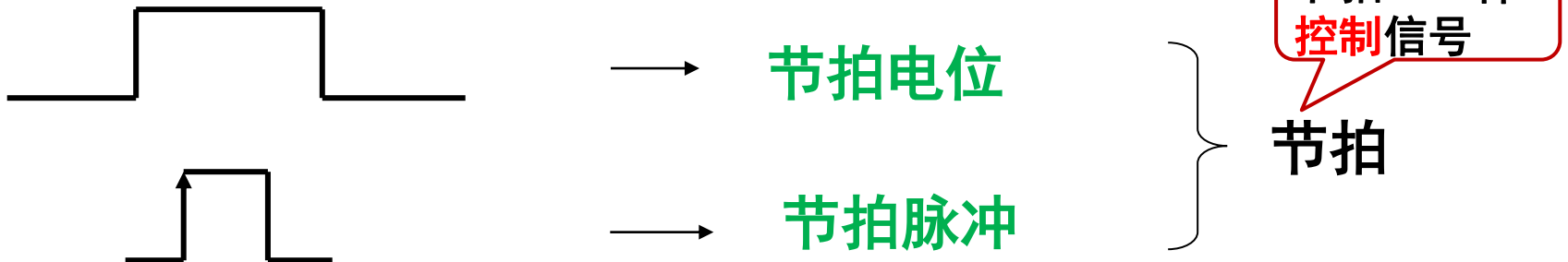
输出方程

$$\left\{ \begin{array}{l} B_3 = \overline{C_3} W_{CB} = \overline{C_3} \\ B_2 = \overline{C_2} W_{CB} = \overline{C_2} \\ B_1 = \overline{C_1} W_{CB} = \overline{C_1} \end{array} \right. \longrightarrow \begin{array}{l} B_3 = C_3 \\ B_2 = C_2 \\ B_1 = C_1 \end{array}$$

双向移位寄存器



- 寄存器每一个操作（写入、左移、右移）都是在**节拍的控制**（节拍脉冲CP和节拍电平Wac、Wr、Wl）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**

- 写入操作，需要 $W_{AC} = 1$, $cp \uparrow$
- 左移操作，需要 $W_L = 1$, $cp \uparrow$
- 读出操作，需要 $En = 1$

用Verilog实现移位寄存器

程序1：串入并出8位移位寄存器模块

```
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;          //Synchronous clear
        else Q <= {Q[WID-2:0], SERIN}; // Shift Left
endmodule
```

用Verilog实现移位寄存器

程序2：通用4位移位寄存器模块

```
module Vrshrg4u ( CLK, CLR, RIN, LIN, S0, S1, A, B, C, D, QA, QB, QC, QD
);
  input CLK, CLR, S0, S1, RIN, LIN, A, B, C, D;
  output reg QA, QB, QC, QD;

  always @ (posedge CLK)
    if (CLR == 1'b1) {QA,QB,QC,QD} <= 4'b0;
    else case ({S1,S0})
      2'b00: ; // Hold
      2'b01: {QA,QB,QC,QD} <= {RIN,QA,QB,QC}; // Shift right
      2'b10: {QA,QB,QC,QD} <= {QB,QC,QD,LIN}; // Shift left
      2'b11: {QA,QB,QC,QD} <= {A,B,C,D}; // Load
      default: {QA,QB,QC,QD} <= 4'bx; // should not occur
    endcase
endmodule
```

Unit 9 Registers and Counters

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



计数器

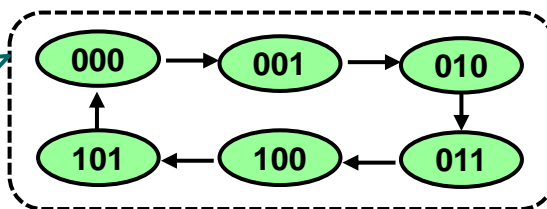
分类方式		种类	特点	电路框图示例
时序逻辑电路	按照 时钟 信号的连接方式	同步时序—	<ul style="list-style-type: none"> ■ 特点：所有的时钟端连接在一起，状态的改变同时发生（在数字系统中用到的最多） 	
		异步时序—	<ul style="list-style-type: none"> ■ 没有统一的时钟脉冲同步，状态的改变有先有后，不同时发生 ■ 容易产生毛刺 	
	按照 电路输出与输入及电路状态 的关系	摩尔型电路 (<i>Moore</i>)	<ul style="list-style-type: none"> ■ 电路的输出仅与现态有关，与电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

计数器

计数器？

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**等。

模**6**加法计
数器



特点：只要连续提供**时钟脉冲**，不需要人为干预，周而复始地工作

- 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

□ 计数器的种类

按触发方式分

- 同步计数器
- 异步计数器

按功能分

- 加法计数器
- 减法计数器
- 可逆计数器

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）

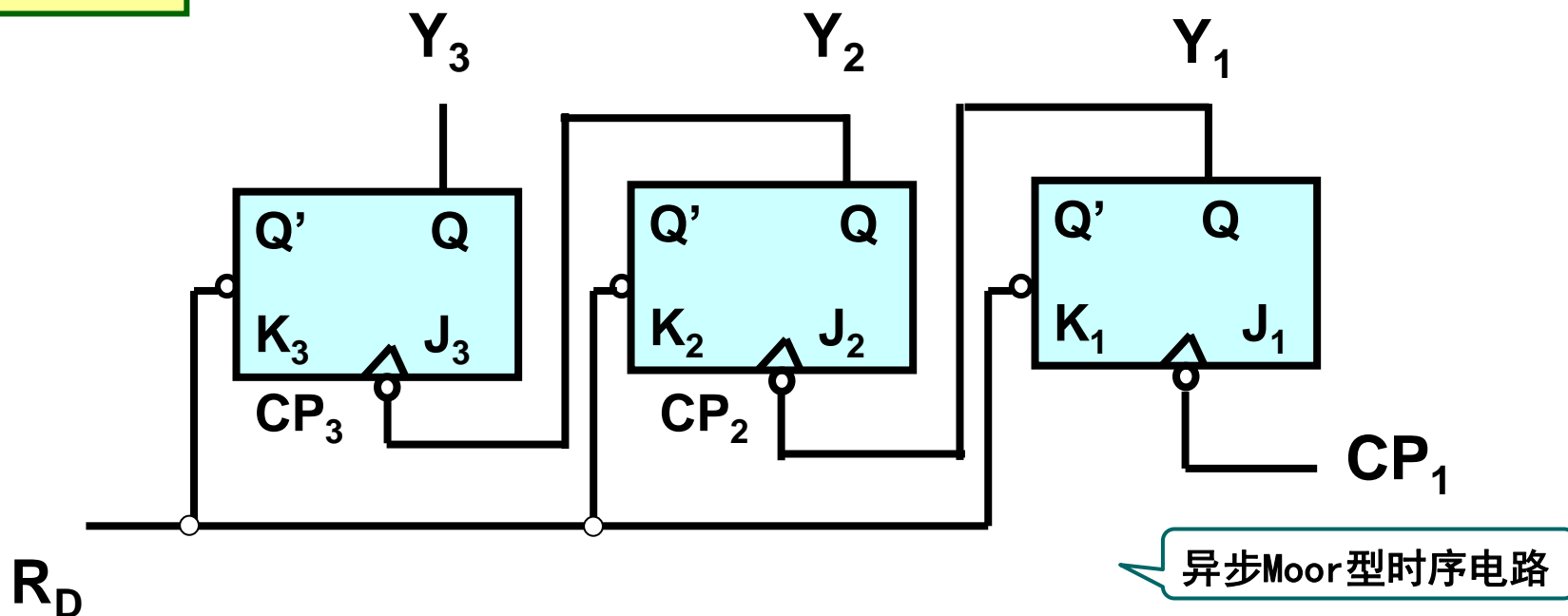
④ 列写状态转换表

⑤ 画出状态图

⑥ 画出波形图（如必要）

异步计数器

Example



① 输入方程

$$J_1 = K_1 = 1$$

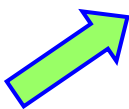
$$CP_1 \downarrow$$

$$J_2 = K_2 = 1$$

$$CP_2 = Y_1 \downarrow$$

$$J_3 = K_3 = 1$$

$$CP_3 = Y_2 \downarrow$$



② 次态方程

$$Y_1^{n+1} = J_1 \bar{Q}_1 + \bar{K}_1 Q_1 = \bar{Y}_1$$

$$Y_2^{n+1} = J_2 \bar{Q}_2 + \bar{K}_2 Q_2 = \bar{Y}_2$$

$$Y_3^{n+1} = J_3 \bar{Q}_3 + \bar{K}_3 Q_3 = \bar{Y}_3$$

③状态转换表

	Y_3	Y_2	Y_1	Y_3^{n+1}	Y_2^{n+1}	Y_1^{n+1}	CP_3	CP_2	CP_1
1	0	0	0	0	0	1			↓
2	0	0	1	0	1	0		↓	↓
3	0	1	0	0	1	1			↓
4	0	1	1	1	0	0	↓	↓	↓
5	1	0	0	1	0	1			↓
6	1	0	1	1	1	0		↓	↓
7	1	1	0	1	1	1			↓
8	1	1	1	0	0	0	↓	↓	↓

$$J_1 = K_1 = 1$$

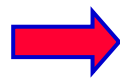
$$CP_1 \downarrow$$

$$J_2 = K_2 = 1$$

$$CP_2 = Y_1 \downarrow$$

$$J_3 = K_3 = 1$$

$$CP_3 = Y_2 \downarrow$$



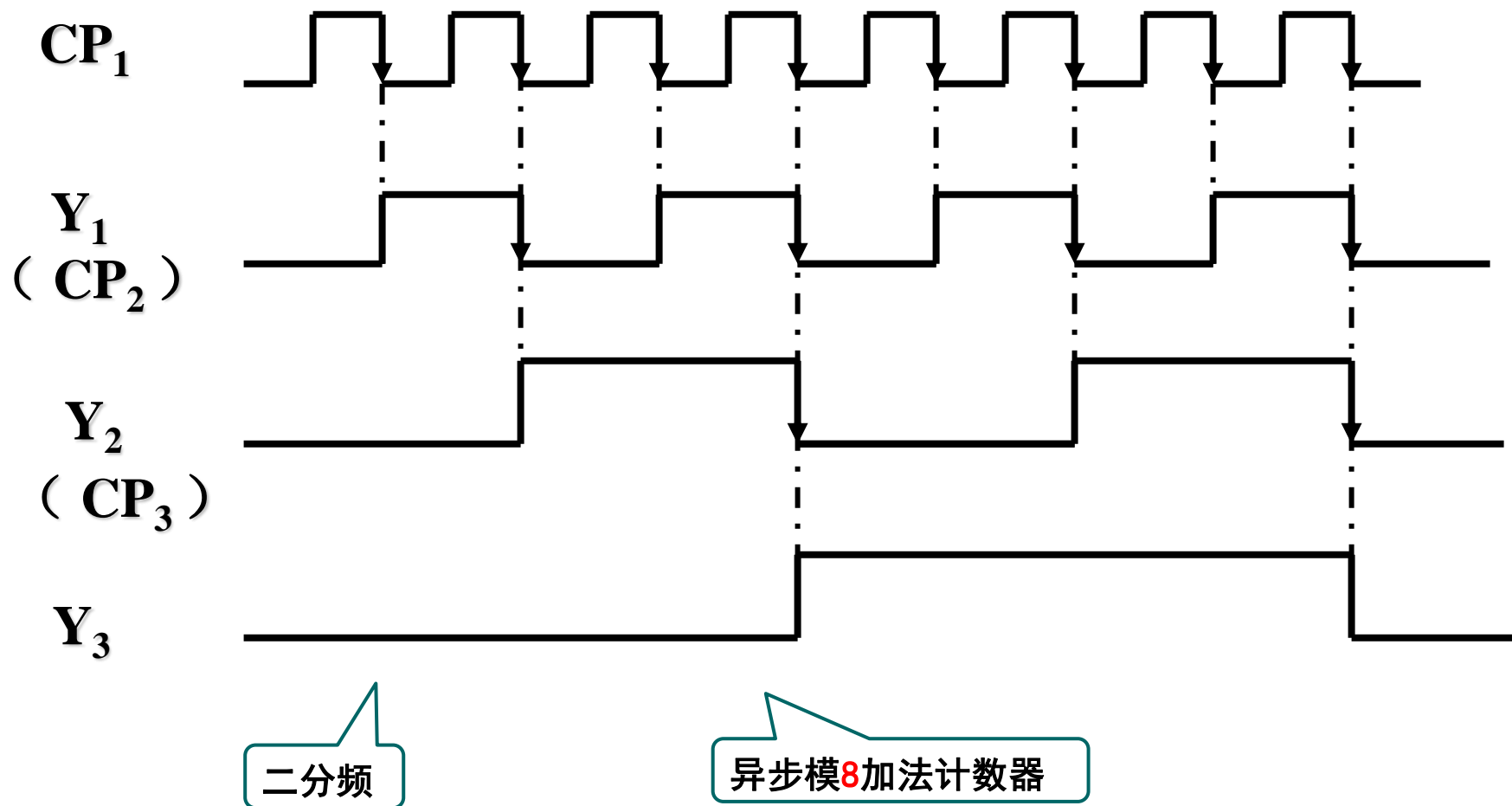
$$Y_1^{n+1} = J_1 \bar{Q}_1 + \bar{K}_1 Q_1 = \bar{Y}_1$$

$$Y_2^{n+1} = J_2 \bar{Q}_2 + \bar{K}_2 Q_2 = \bar{Y}_2$$

$$Y_3^{n+1} = J_3 \bar{Q}_3 + \bar{K}_3 Q_3 = \bar{Y}_3$$

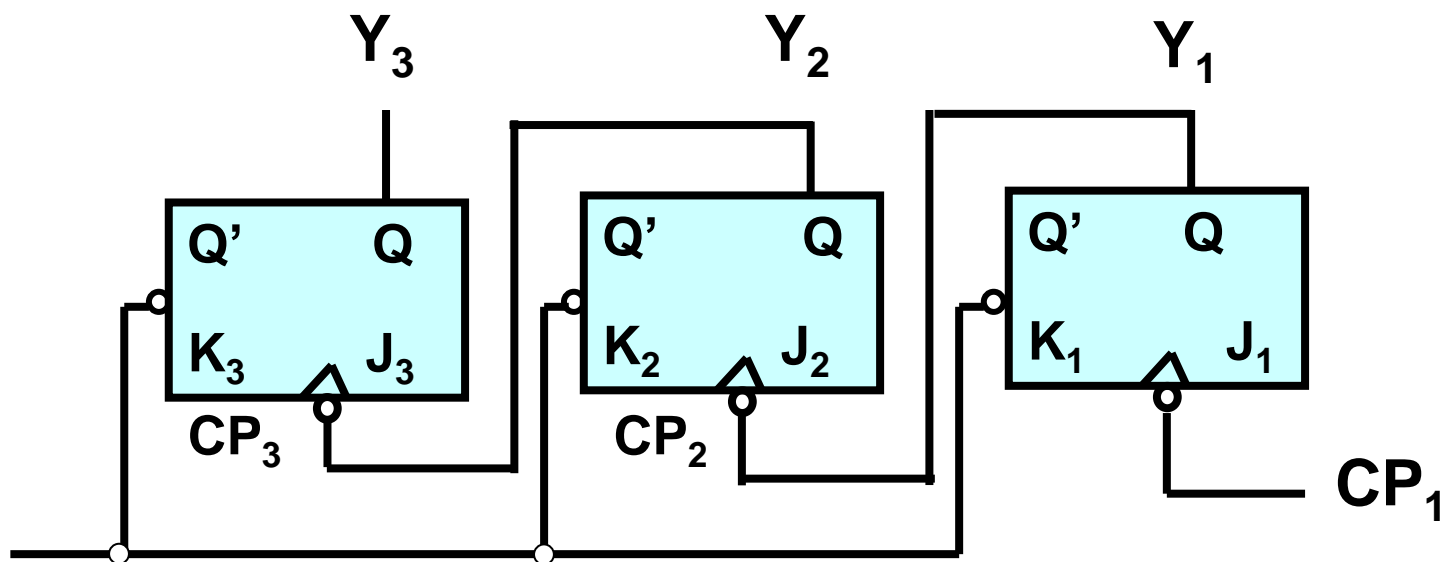
异步计数器

④波形图



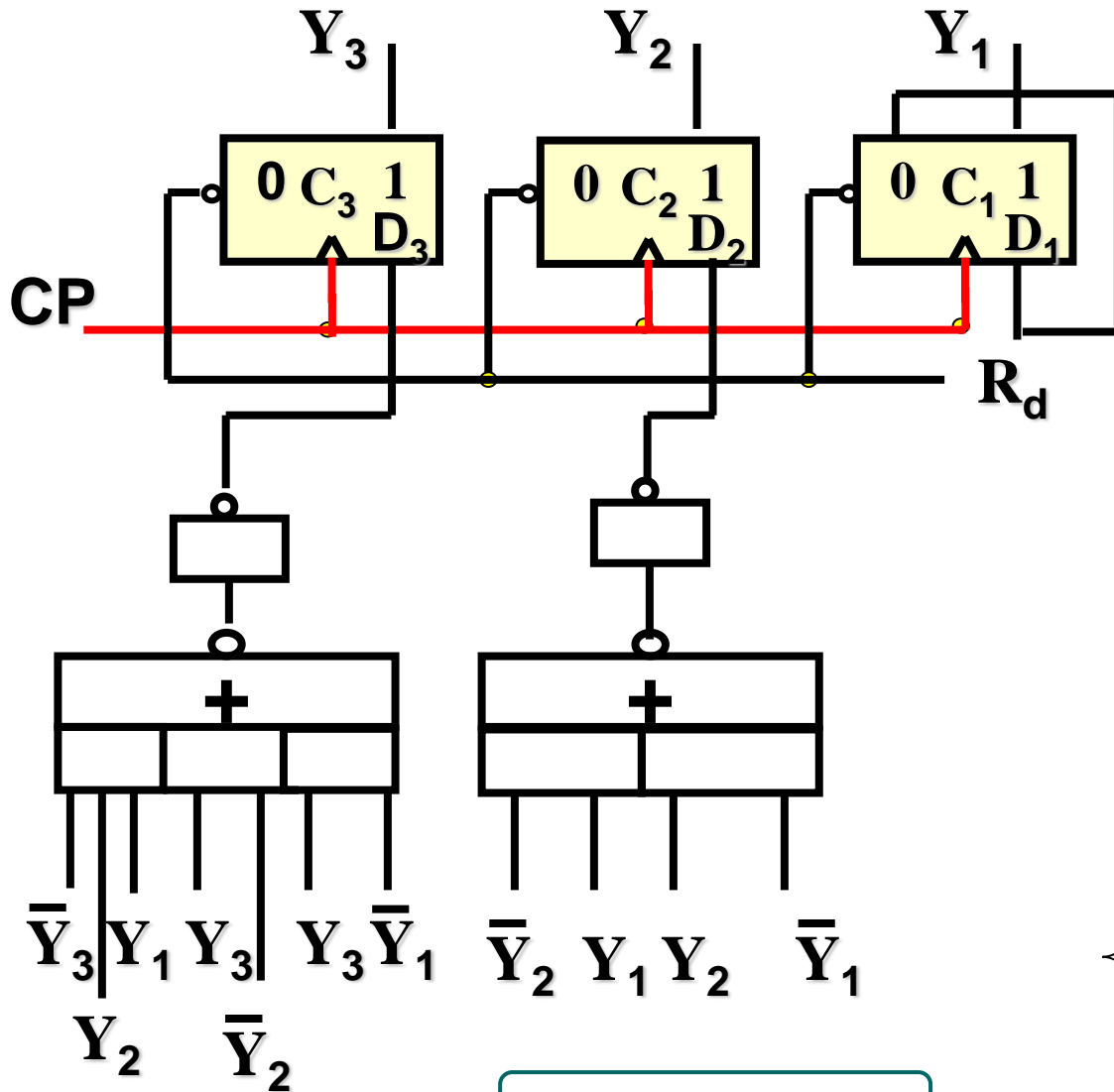
异步计数器

- 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，在大型的数字设备中较少采用。



同步计数器

Example



1. 输入方程

$$D_3 = \bar{Y}_3 Y_2 Y_1 + Y_3 \bar{Y}_2 + Y_3 \bar{Y}_1$$

$$= \bar{Y}_3 Y_2 Y_1 + Y_3 \bar{Y}_2 \bar{Y}_1$$

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

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









2. 次态方程

$$Y_1^{n+1} = D_1$$

$$Y_2^{n+1} = D_2$$

$$Y_3^{n+1} = D_3$$

同步Moore型时序电路

	Y_3	Y_2	Y_1	Y_3^{n+1}	Y_2^{n+1}	Y_1^{n+1}	CP
1	0	0	0	0	0	1	
2	0	0	1	0	1	0	
3	0	1	0	0	1	1	
4	0	1	1	1	0	0	
5	1	0	0	1	0	1	
6	1	0	1	1	1	0	
7	1	1	0	1	1	1	
8	1	1	1	0	0	0	

次态方程

$$Y_1^{n+1} = D_1$$

$$Y_2^{n+1} = D_2$$

$$Y_3^{n+1} = D_3$$

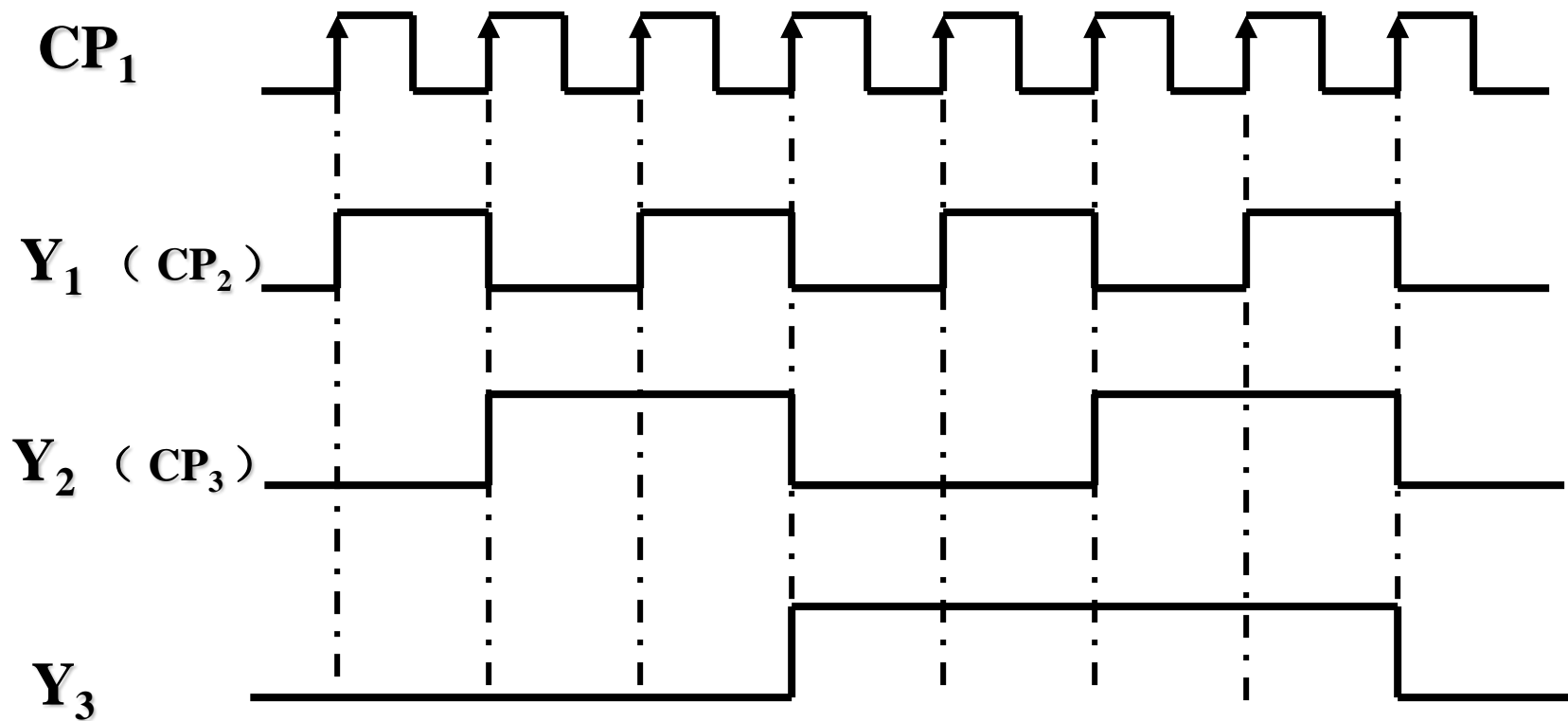
输入方程

$$\begin{aligned}
 D_3 &= \bar{Y}_3 Y_2 Y_1 + Y_3 \bar{Y}_2 + Y_3 \bar{Y}_1 \\
 &= \bar{Y}_3 Y_2 Y_1 + Y_3 \overline{Y_2 Y_1}
 \end{aligned}$$

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

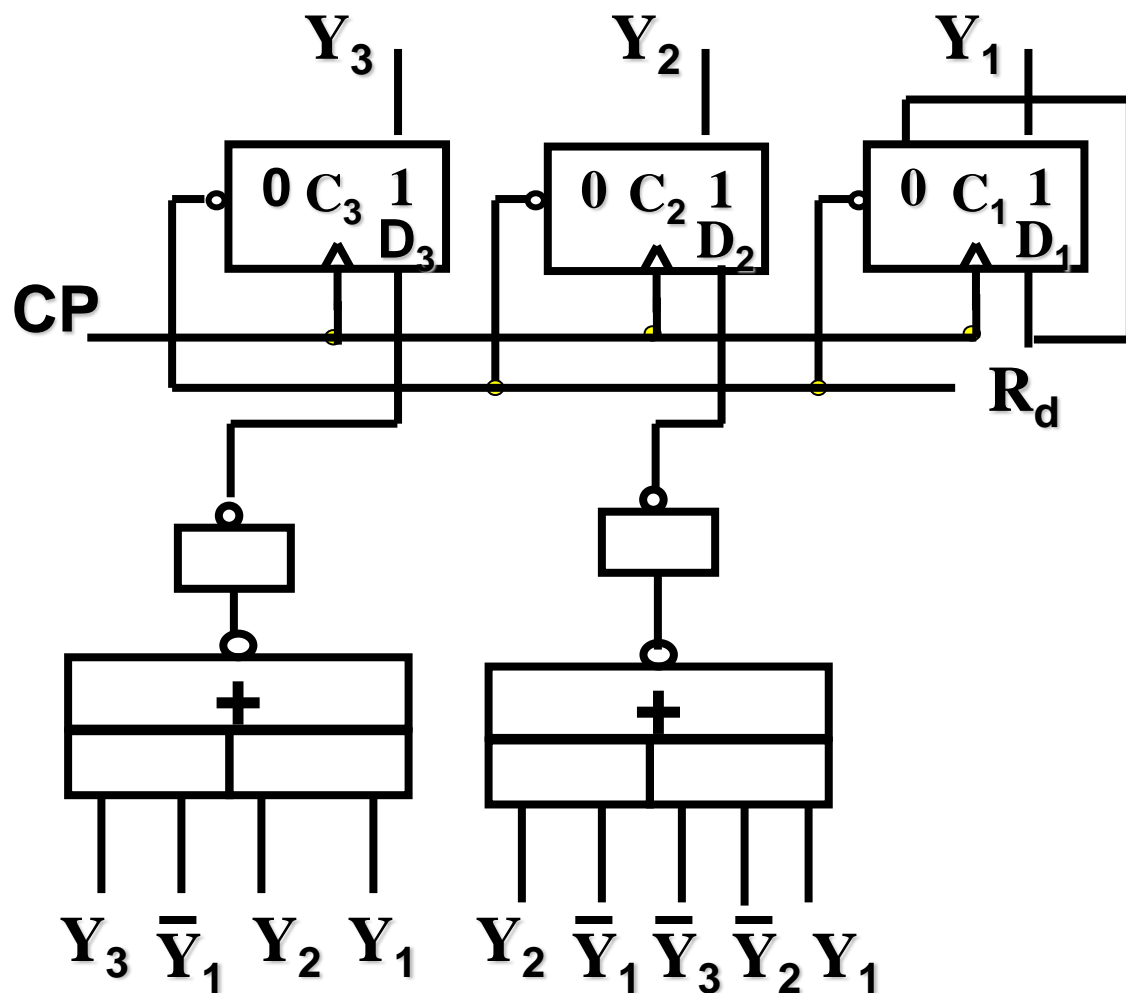
$$D_1 = \bar{Y}_1$$

同步计数器



同步计数器

Example



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

$$Y_1^{n+1} = D_1$$

$$Y_2^{n+1} = D_2$$

$$Y_3^{n+1} = D_3$$

4.状态转换表

	Y_3	Y_2	Y_1	Y_3^{n+1}	Y_2^{n+1}	Y_1^{n+1}	CP
1	0	0	0	0	0	1	
2	0	0	1	0	1	0	
3	0	1	0	0	1	1	
4	0	1	1	1	0	0	
5	1	0	0	1	0	1	
6	1	0	1	0	0	0	
7	1	1	0	1	1	1	
8	1	1	1	1	0	0	

输入方程

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

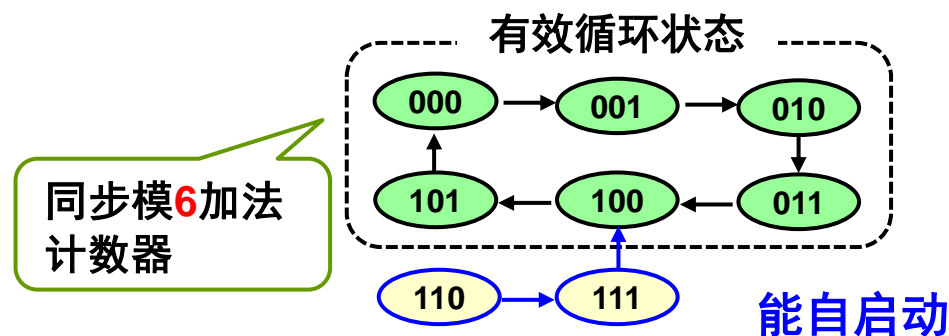
$$D_2 = Y_2 \bar{Y}_1 + \bar{Y}_3 \bar{Y}_2 Y_1$$

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

3.次态方程

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

5.状态图



同步计数器

- 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（各个触发器的输入由多个Q输出的组合逻辑得到），所以用到**元件较多**。

用Verilog实现计数器

程序3：4位通用二进制计数器模块

```
module Vrcntr4u ( CLK, CLR, LD, ENP, ENT, D, Q, RCO );
    input CLK, CLR, LD, ENP, ENT;
    input [3:0] D;
    output reg [3:0] Q;
    output reg RCO;

    always @ (posedge CLK) // Create the counter f-f behavior
        if (CLR)           Q <= 4'd0;
        else if (LD)        Q <= D;
        else if (ENT && ENP) Q <= Q + 1;
        else                Q <= Q;

    always @ (Q or ENT)    // Create RCO combinational output
        if (ENT && (Q == 4'd15)) RCO = 1;
        else                RCO = 0;
endmodule
```

用Verilog实现计数器

程序4：4位十进制计数器模块

```
module Vrcntr4udec ( CLK, CLR, LD, ENP, ENT, D, Q, RCO );
    input CLK, CLR, LD, ENP, ENT;
    input [3:0] D;
    output reg [3:0] Q;
    output reg RCO;

    always @ (posedge CLK) // Create the counter f-f behavior
        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 @ (Q or ENT)    // Create RCO combinational output
        if (ENT && (Q == 4'd9)) RCO = 1;
        else                RCO = 0;
endmodule
```

用Verilog实现计数器

程序5：余3十进制计数器模块

```
module Vrexcess3 ( CLK, CLR, LD, ENP, ENT, D, Q, RCO );
    input CLK, CLR, LD, ENP, ENT;
    input [3:0] D;
    output reg [3:0] Q;
    output reg RCO;

    always @ (posedge CLK) // Create the counter f-f behavior
        if (CLR)            Q <= 4'd3;
        else if (LD)         Q <= D;
        else if (ENT && ENP && (Q == 4'd12)) Q <= 4'd3;
        else if (ENT && ENP)  Q <= Q + 1;
        else                 Q <= Q;

    always @ (Q or ENT) // Create RCO combinational output
        if (ENT && (Q == 4'd12)) RCO = 1;
        else                     RCO = 0;
endmodule
```

用Verilog实现计数器

程序6：4位递增/递减计数器模块

```
module Vrupdn4 ( CLK, CLR, LD, ENP, ENT, UPDN, D, Q, RCO );
    input CLK, CLR, LD, ENP, ENT, UPDN;
    input [3:0] D;
    output reg [3:0] Q;
    output reg RCO;

    always @ (posedge CLK)    // Create the counter f-f behavior
        if (CLR )            Q <= 4'b0;
        else if (LD)          Q <= D;
        else if (ENT && ENP && UPDN)    Q <= Q + 1;
        else if (ENT && ENP && !UPDN)   Q <= Q - 1;
        else                  Q <= Q;

    always @ (Q or ENT or UPDN) // Create RCO combinational output
        if (ENT && UPDN && (Q == 4'd15)) RCO = 1;
        else if (ENT && !UPDN && (Q == 4'd0 )) RCO = 1;
        else RCO = 0;
endmodule
```

Unit 9 Registers and Counters

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



节拍发生器

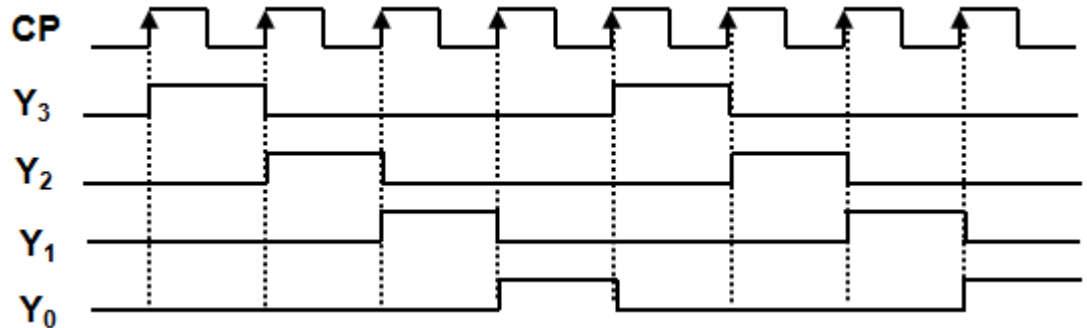
节拍发生器（顺序脉冲发生器）——

定义

在每个**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 |

节拍发生器

Example 1

3.次态方程

$$Y_1^{n+1} = J_1 \bar{Q}_1 + \bar{K}_1 Q_1 = \bar{Y}_1$$

$$Y_2^{n+1} = J_2 \bar{Q}_2 + \bar{K}_2 Q_2 = \bar{Y}_2$$

2.输出方程

$$W_0 = \bar{Y}_2 \bar{Y}_1$$

$$W_1 = \bar{Y}_2 Y_1$$

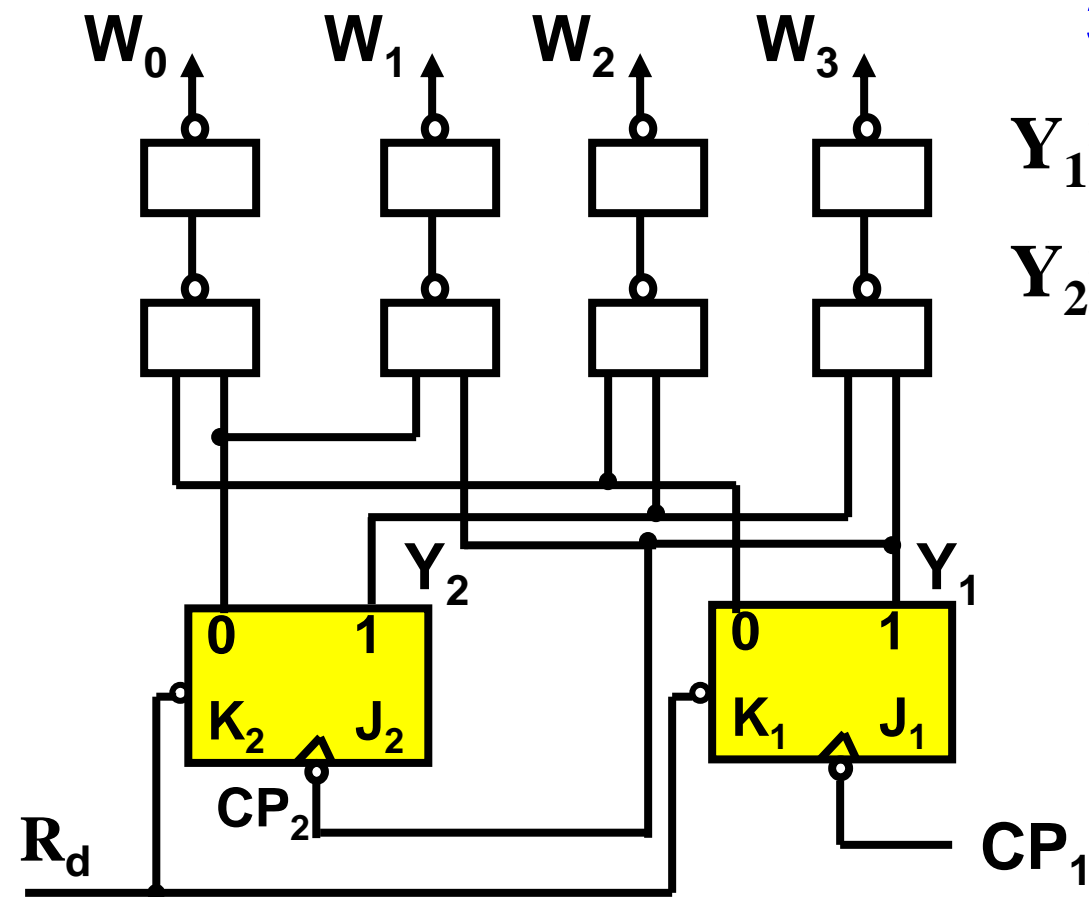
$$W_2 = Y_2 \bar{Y}_1$$

$$W_3 = Y_2 Y_1$$







1.输入方程

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

$$J_2 = K_2 = 1, \quad CP_2 = Y_1 \downarrow$$



4.状态转换表

	Y_2	Y_1	Y_2^{n+1}	Y_1^{n+1}	CP_2	CP_1
1	0	0	0	1		
2	0	1	1	0		
3	1	0	1	1		
4	1	1	0	0		

输入方程

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

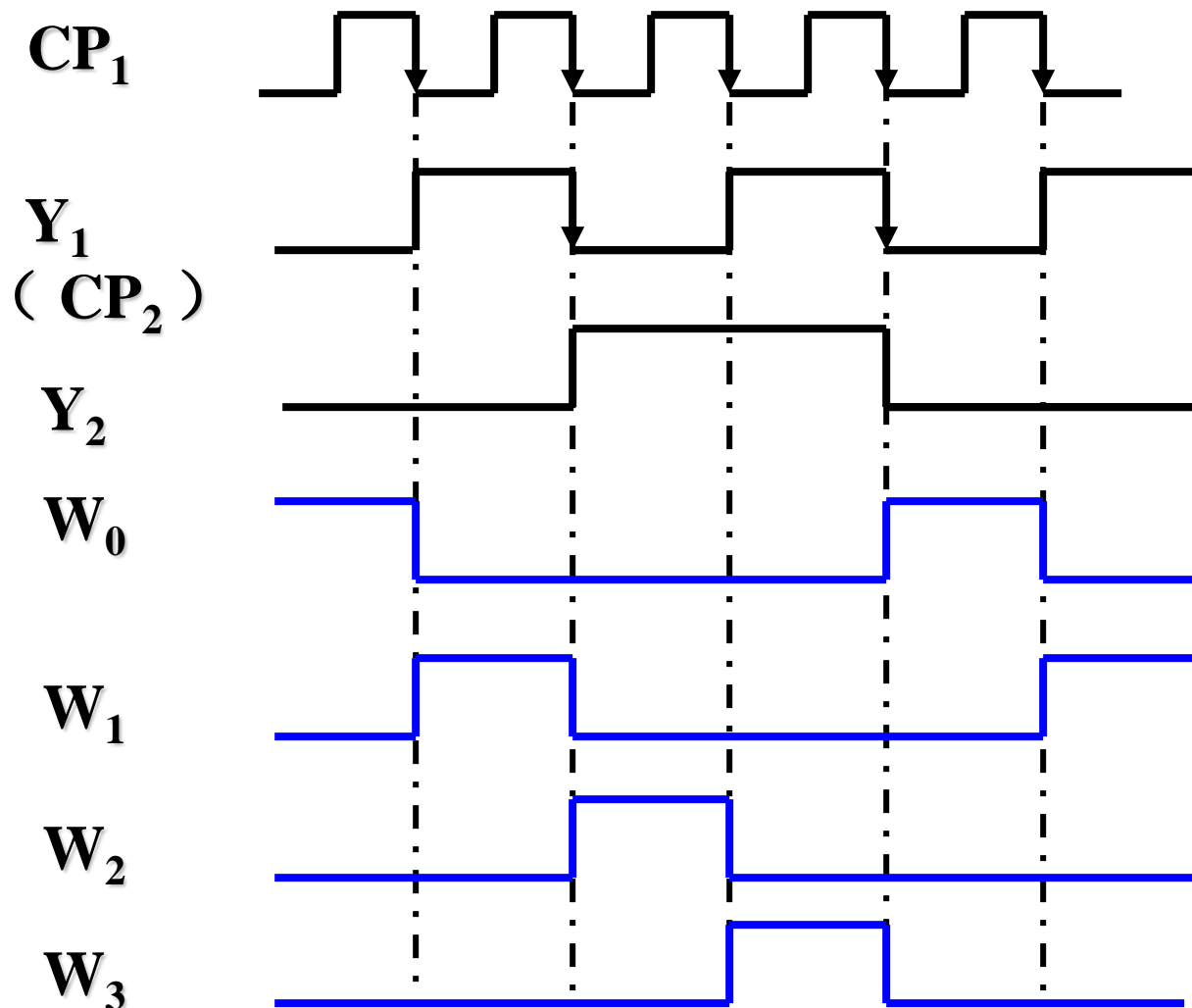
$$J_2 = K_2 = 1, \quad CP_2 = Y_1 \downarrow$$

次态方程

$$Y_1^{n+1} = J_1 \bar{Q}_1 + \bar{K}_1 Q_1 = \bar{Y}_1$$

$$Y_2^{n+1} = J_2 \bar{Q}_2 + \bar{K}_2 Q_2 = \bar{Y}_2$$

5. 波形图



$$W_0 = \overline{Y_2} \overline{Y_1}$$

$$W_1 = \overline{Y_2} Y_1$$

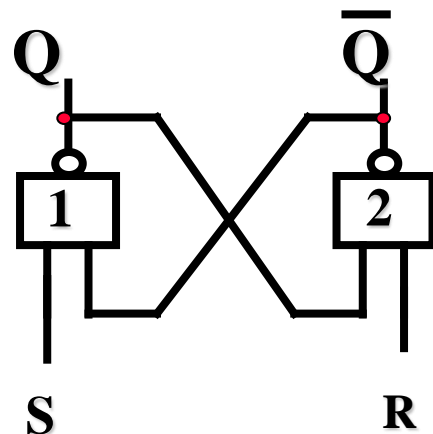
$$W_2 = Y_2 \overline{Y_1}$$

$$W_3 = Y_2 Y_1$$



4-节拍发生器 ($W_0 \sim W_3$)

Example 2



1.输入方程

$$\mathbf{D}_1 = \mathbf{Y}_2$$

$$\mathbf{D}_2 = \mathbf{Y}_3$$

$$\mathbf{D}_3 = \mathbf{Y}_4$$

$$\mathbf{D}_4 = \overline{\mathbf{Y}_4 + \mathbf{Y}_3 + \mathbf{Y}_2}$$

2.次态方程

$$\mathbf{Y}_1^{n+1} = \mathbf{Y}_2, \mathbf{Y}_2^{n+1} = \mathbf{Y}_3, \mathbf{Y}_3^{n+1} = \mathbf{Y}_4, \mathbf{Y}_4^{n+1} = \overline{\mathbf{Y}_4 + \mathbf{Y}_3 + \mathbf{Y}_2}$$

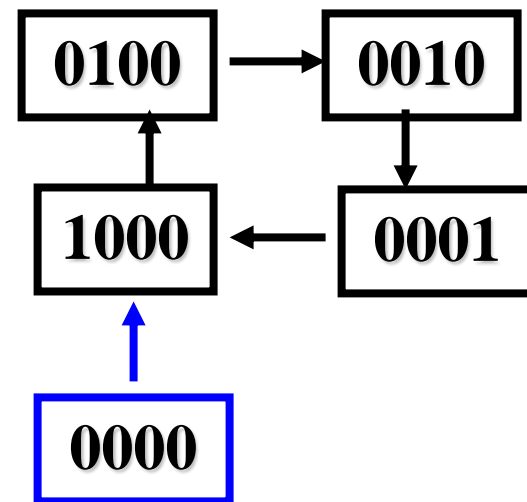
Example 2

节拍发生器

3. 状态转换表

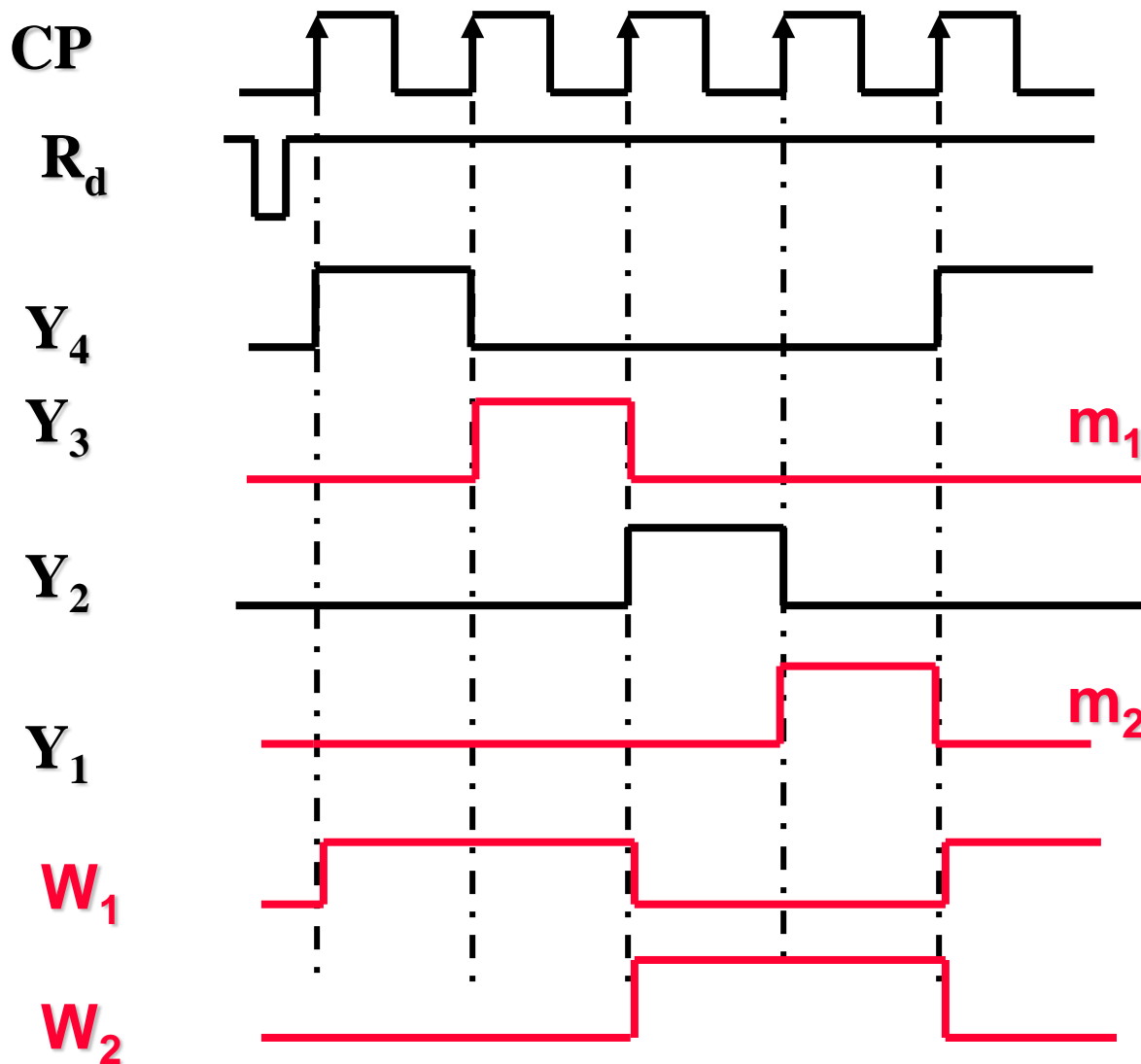
	$Y_4 Y_3 Y_2 Y_1$	Y_4^{n+1}	Y_3^{n+1}	Y_2^{n+1}	Y_1^{n+1}	CP
1	0 0 0 0	1	0	0	0	↑
2	1 0 0 0	0	1	0	0	↑
3	0 1 0 0	0	0	1	0	↑
4	0 0 1 0	0	0	0	1	↑
5	0 0 0 1	1	0	0	0	↑

4. 状态图



$$Y_1^{n+1} = Y_2, Y_2^{n+1} = Y_3, Y_3^{n+1} = Y_4, Y_4^{n+1} = \overline{Y_4} + Y_3 + Y_2$$

5.波形图



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

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

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

用Verilog实现节拍发生器

程序7：6相时序发生器模块

```
module Vrtimegen6 ( CLK, RESET, RUN, RESTART, P_L );
    input CLK, RESET, RUN, RESTART;
    output [1:6] P_L;
    reg [1:6] IP; // internal active-high phase signals
    reg T1;      // first tick within phase

    always @ (posedge CLK)
        if (RESET == 1) begin T1 <= 1; IP <= 6'b0; end
        else if ( (IP == 6'b0) || (RESTART == 1) )
            begin T1 <= 1; IP <= 6'b100000; end
        else if (RUN == 1)
            begin T1 <= ~T1; if (T1==0) IP <= {(IP[1:5]==0),IP[1:5]}; end

    assign P_L = ~IP; // active-low phase outputs
endmodule
```

Unit 9 Registers and Counters

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