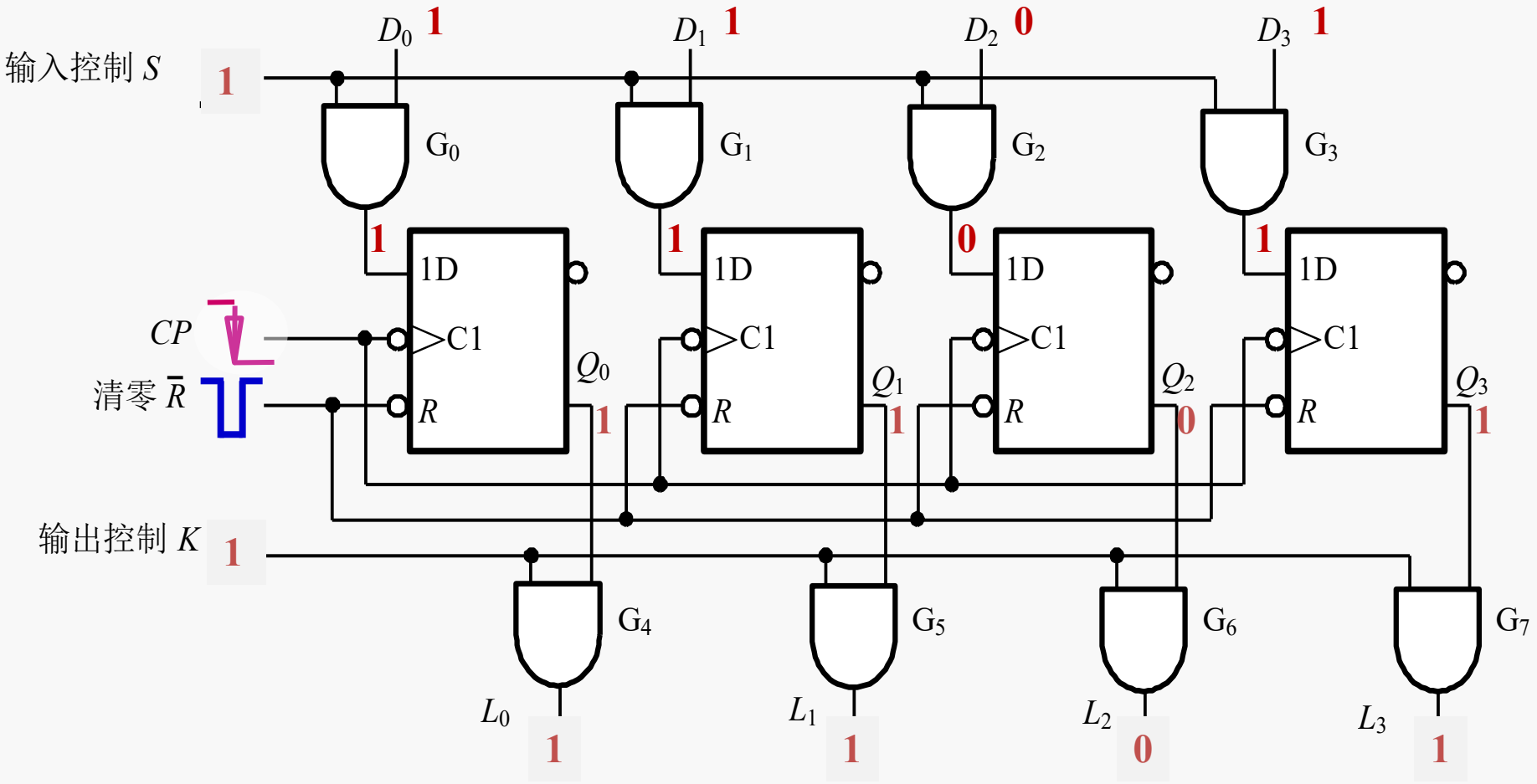
# 6 时序逻辑电路

# 6.4 寄存器

**寄存器：**数字系统中用来存储**少量**代码或数据的逻辑部件，主要由若干触发器组成。

一个触发器能存储1位二值代码，存储$N$位数码的寄存器则应由$N$个触发器组成。

## 4位寄存器




## 脉冲边沿敏感的寄存器

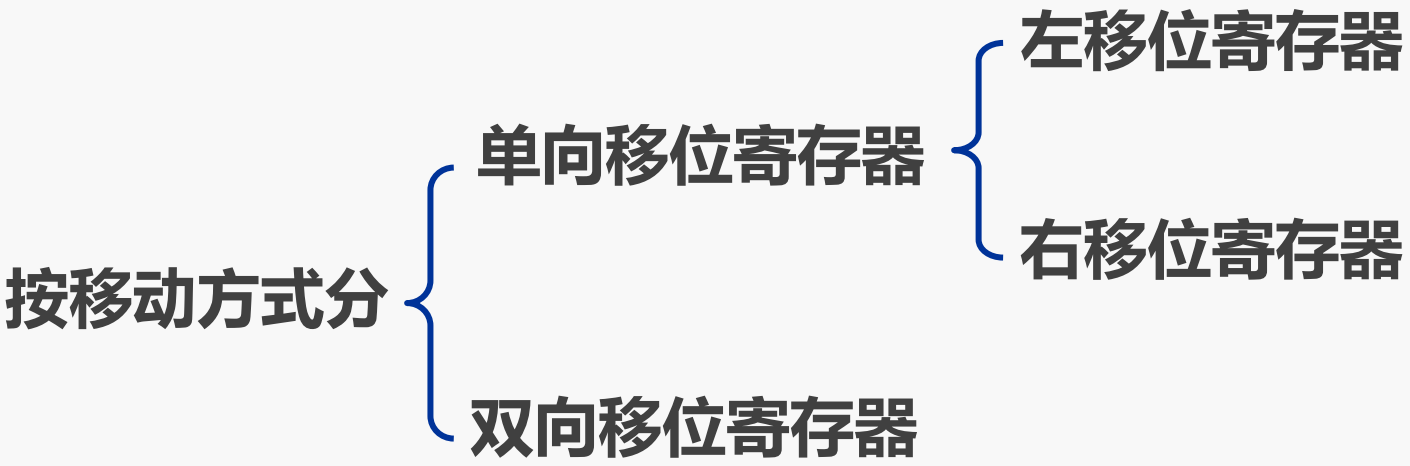
## 8位CMOS寄存器74LV374

| 工作模式 | 输入 | | | 内部触发器 $Q_N^{n+1}$ | 输出 |
|---|---|---|---|---|---|
| | $\overline{OE}$ | $CP$ | $D_N$ | | $Q_0$~$Q_7$ |
| 存入和读出数据 | L | ↑ | L | L | 对应内部触发器的状态 |
| | L | ↑ | H | H | |
| 存入数据，禁止输出 | H | ↑ | L | L | 高阻 |
| | H | ↑ | H | H | 高阻 |

# 6.4 移位寄存器

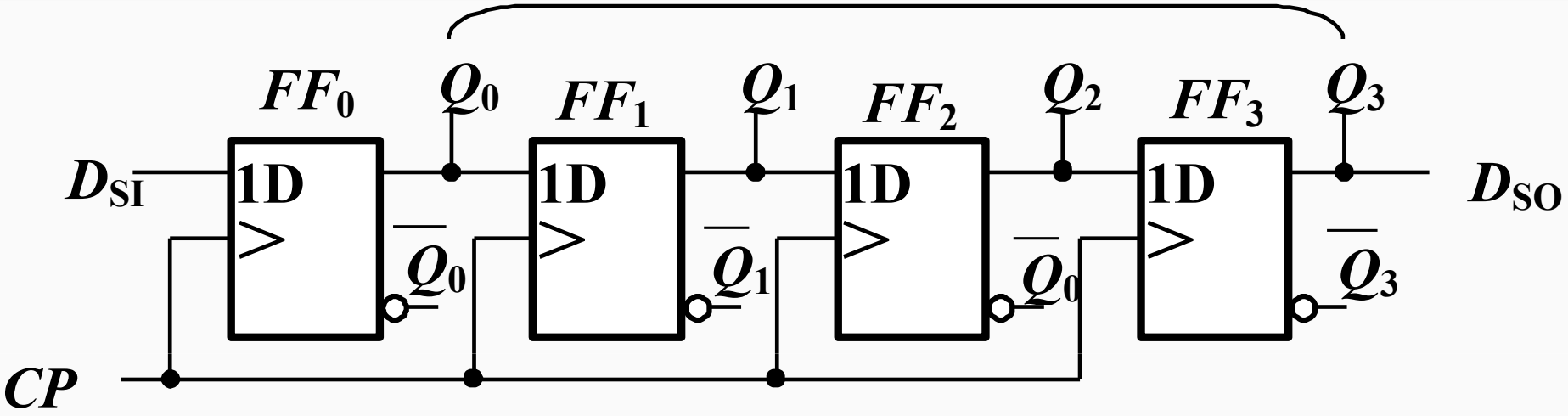
**移位寄存器：**既能寄存数码，又能在时钟脉冲的作用下使数码**向高位或向低位**移动的逻辑功能部件。

移位寄存器的分类



# 6.4 移位寄存器

## 4位单向右移移位寄存器




**激励方程：**

$D_0=D_{SI}$　　$D_1=Q_0^n$　　$D_2=Q^n_1$　　$D_3=Q^n_2$

**状态方程：**

$Q_0^{n+1}=D_{SI}$　　$Q_1^{n+1}=D_1=Q_0^n$

$Q_2^{n+1}=D_2=Q^n_1$　　$Q_3^{n+1}=D_3=Q_2^n$

**用此方法分析移位寄存器有何问题？**

# 6.4 移位寄存器

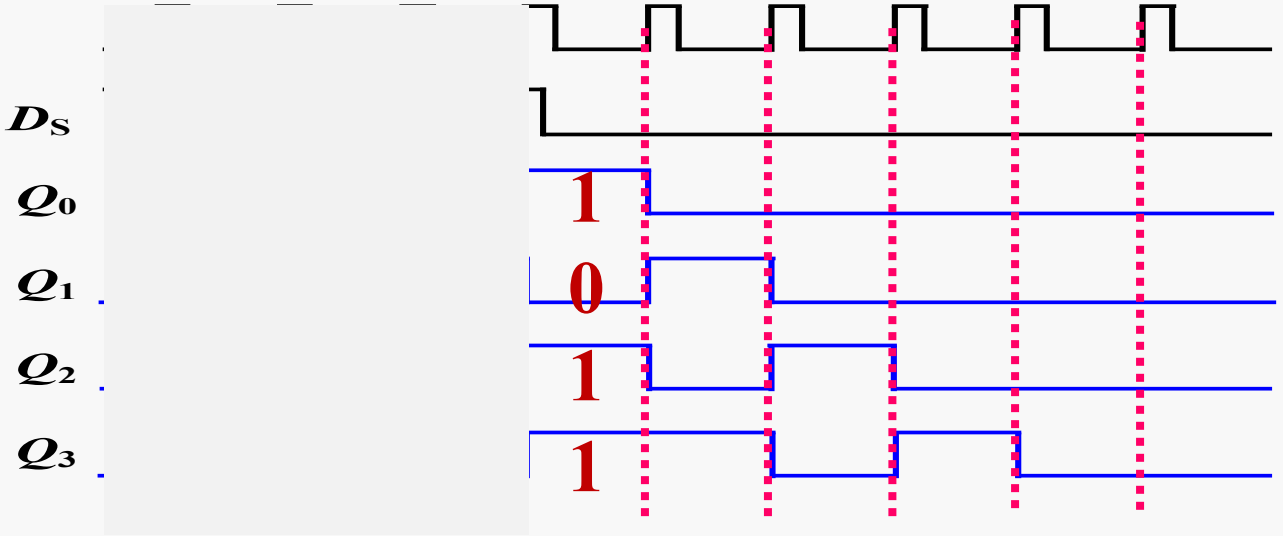
| 输入 | 初态 | 次态 | 输入 | 初态 | 次态 |
|---|---|---|---|---|---|
| 0 | 0000 | 0000 | 1 | 0000 | 0001 |
| 0 | 0001 | 0010 | 1 | 0001 | 0011 |
| 0 | 0010 | 0100 | 1 | 0010 | 0101 |
| 0 | 0011 | 0110 | 1 | 0011 | 0111 |
| 0 | 0100 | 1000 | 1 | 0100 | 1001 |
| 0 | 0101 | 1010 | 1 | 0101 | 1011 |
| 0 | 0110 | 1100 | 1 | 0110 | 1101 |
| 0 | 0111 | 1110 | 1 | 0111 | 1111 |
| 0 | 1000 | 0000 | 1 | 1000 | 0001 |
| 0 | 1001 | 0010 | 1 | 1001 | 0011 |
| 0 | 1010 | 0100 | 1 | 1010 | 0101 |
| 0 | 1011 | 0110 | 1 | 1011 | 0111 |
| 0 | 1100 | 1000 | 1 | 1100 | 1001 |
| 0 | 1101 | 1010 | 1 | 1101 | 1011 |
| 0 | 1110 | 1100 | 1 | 1110 | 1101 |
| 0 | 1111 | 1110 | 1 | 1111 | 1111 |

# 6.4 移位寄存器

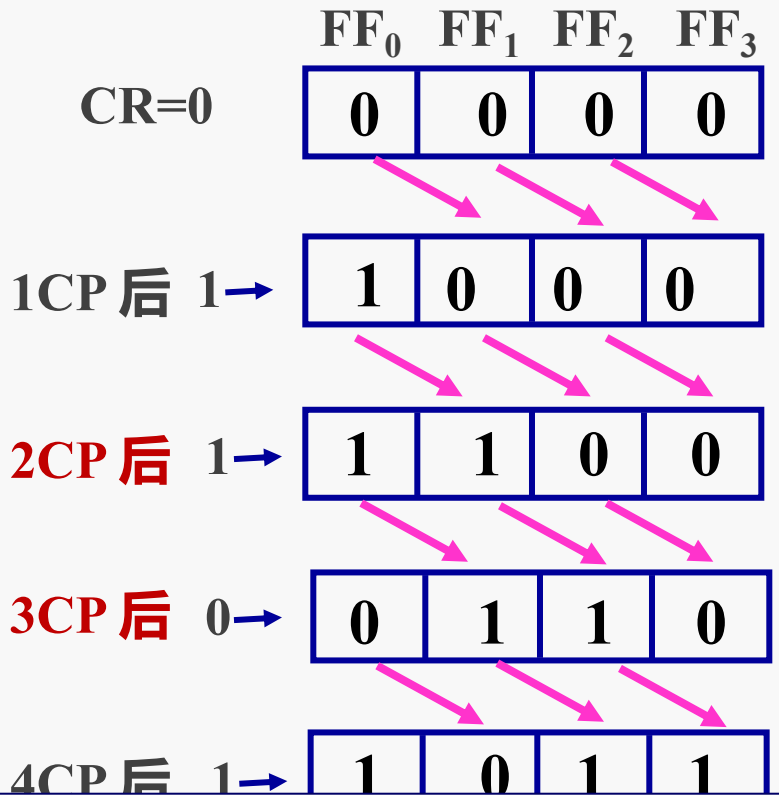
经过4个CP脉冲作用后，从$D_S$端串行输入的数码就可以从$Q_0\ Q_1\ Q_2\ Q_3$并行输出。 **串入→并出**

经过3个CP脉冲作用后，从$D_I$端串行输入的数码就可以从$D_O$端串行输出。 **并入→串出**

每个触发器的输出波形相同，只是依次延时了1个时钟周期。 **信号延时**



# 6.4 移位寄存器




$D_S=1101$

$Q_0^{n+1}=D_I$ $Q_1^{n+1}=Q_0^n$

$Q_2^{n+1}=Q^n_1$ $Q_3^{n+1}=Q^n_2$

移位寄存器右移，除2

移位寄存器左移，乘2

# 6.4 移位寄存器

移位寄存器的应用：

1、串并转换

2、并串转换

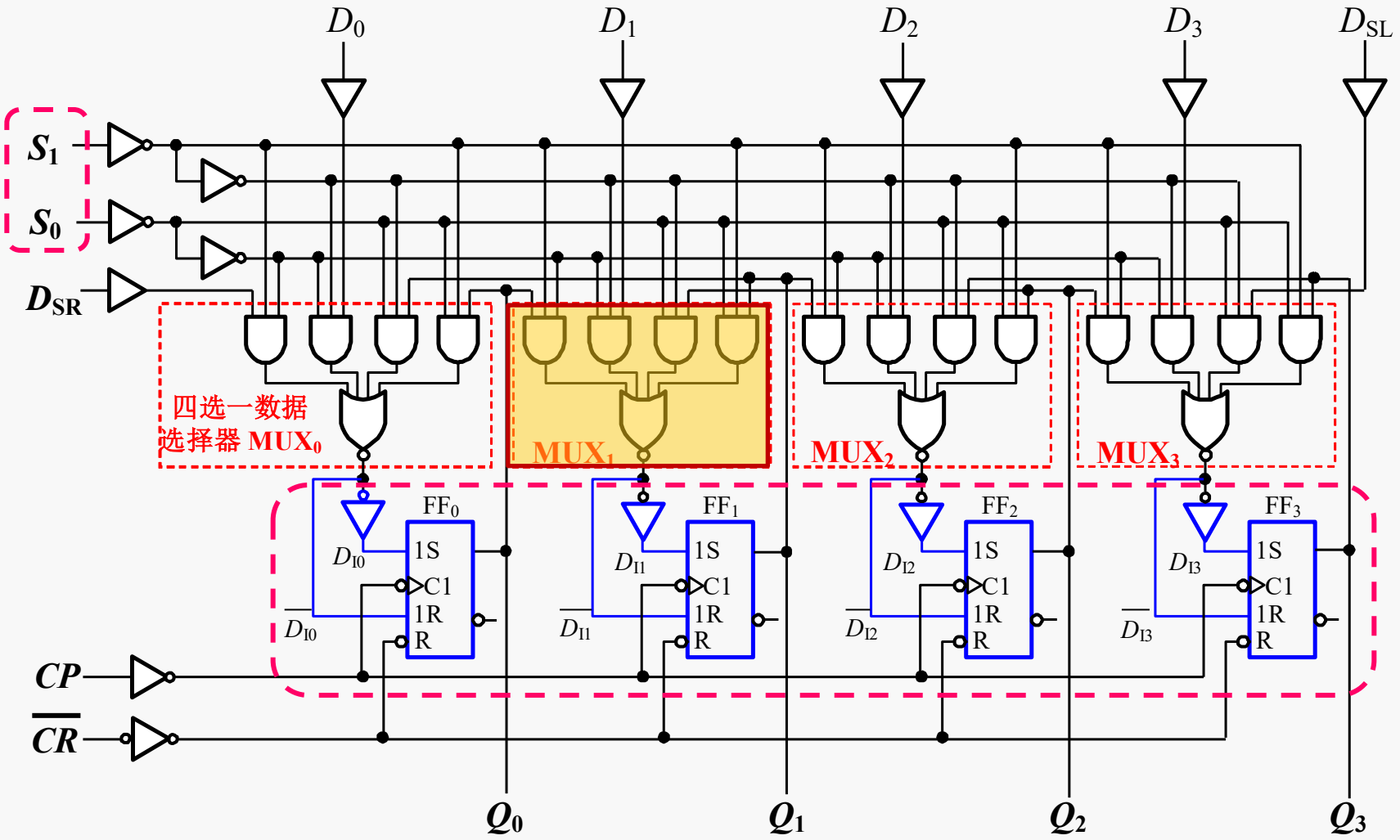
3、信号延时

4、乘/除2^n

5、与其他电路配合完成复杂逻辑功能（110检测）

# 6.4 移位寄存器

## CMOS 4位双向移位寄存器74HCT194



# 6.4 移位寄存器

## 74HCT194功能表

| 输入 | | | | | | | | | | 输出 | | | | 功能 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 清零 | 控制信号 | | 串行输入 | | 时钟 | 并行输入 | | | | | | | | |
| $\overline{CR}$ | $S_1$ | $S_0$ | 右移 $D_{SR}$ | 左移 $D_{SL}$ | $CP$ | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $Q_0^{n+1}$ | $Q_1^{n+1}$ | $Q_2^{n+1}$ | $Q_3^{n+1}$ | |
| L | × | × | × | × | × | × | × | × | × | L | L | L | L | 1 |
| H | L | L | × | × | × | × | × | × | × | $Q_0^n$ | $Q_1^n$ | $Q_2^n$ | $Q_3^n$ | 2 |
| H | L | H | L | × | ↑ | × | × | × | × | L | $Q_0^n$ | $Q_1^n$ | $Q_2^n$ | 3 |
| H | L | H | H | × | ↑ | × | × | × | × | H | $Q_0^n$ | $Q_1^n$ | $Q_2^n$ | 4 |
| H | H | L | × | L | ↑ | × | × | × | × | $Q_1^n$ | $Q_2^n$ | $Q_3^n$ | L | 5 |
| H | H | L | × | H | ↑ | × | × | × | × | $Q_1^n$ | $Q_2^n$ | $Q_3^n$ | H | 6 |
| H | H | H | × | × | ↑ | $D_{I0}$* | $D_{I1}$* | $D_{I2}$* | $D_{I3}$* | $D_{I0}$ | $D_{I1}$ | $D_{I2}$ | $D_{I3}$ | 7 |

异步清零　　低位向高位移动　　同步置数

保持不变　　高位向低位移动

# 6.4 移位寄存器的例题

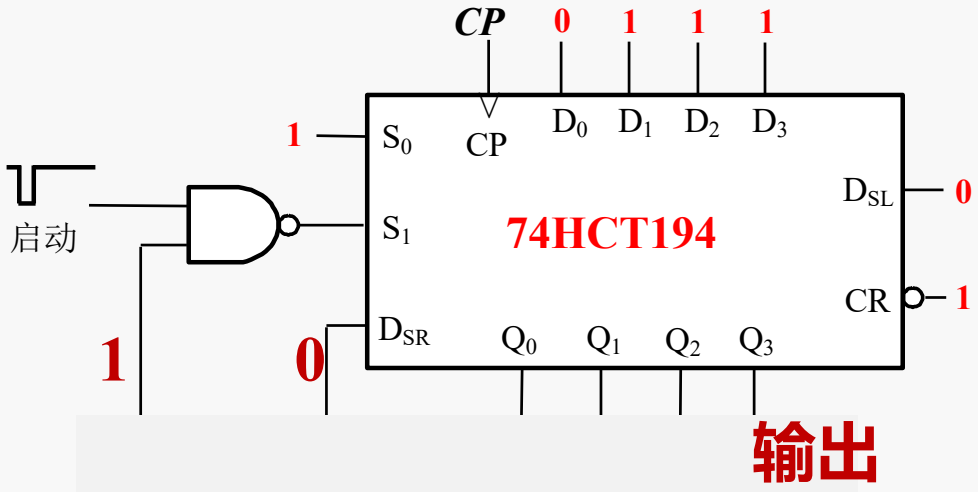
例 时序脉冲产生器。电路如图所示。画出 $Q_0 \sim Q_3$波形，分析逻辑功能。

解：（1）启动信号为0：$S1$=1，$S0$=1，**同步置数**，$Q_0$~$Q_3$=0111

（2）启动信号为1后：$S1$=0，$S0$=1，**向右移位**：

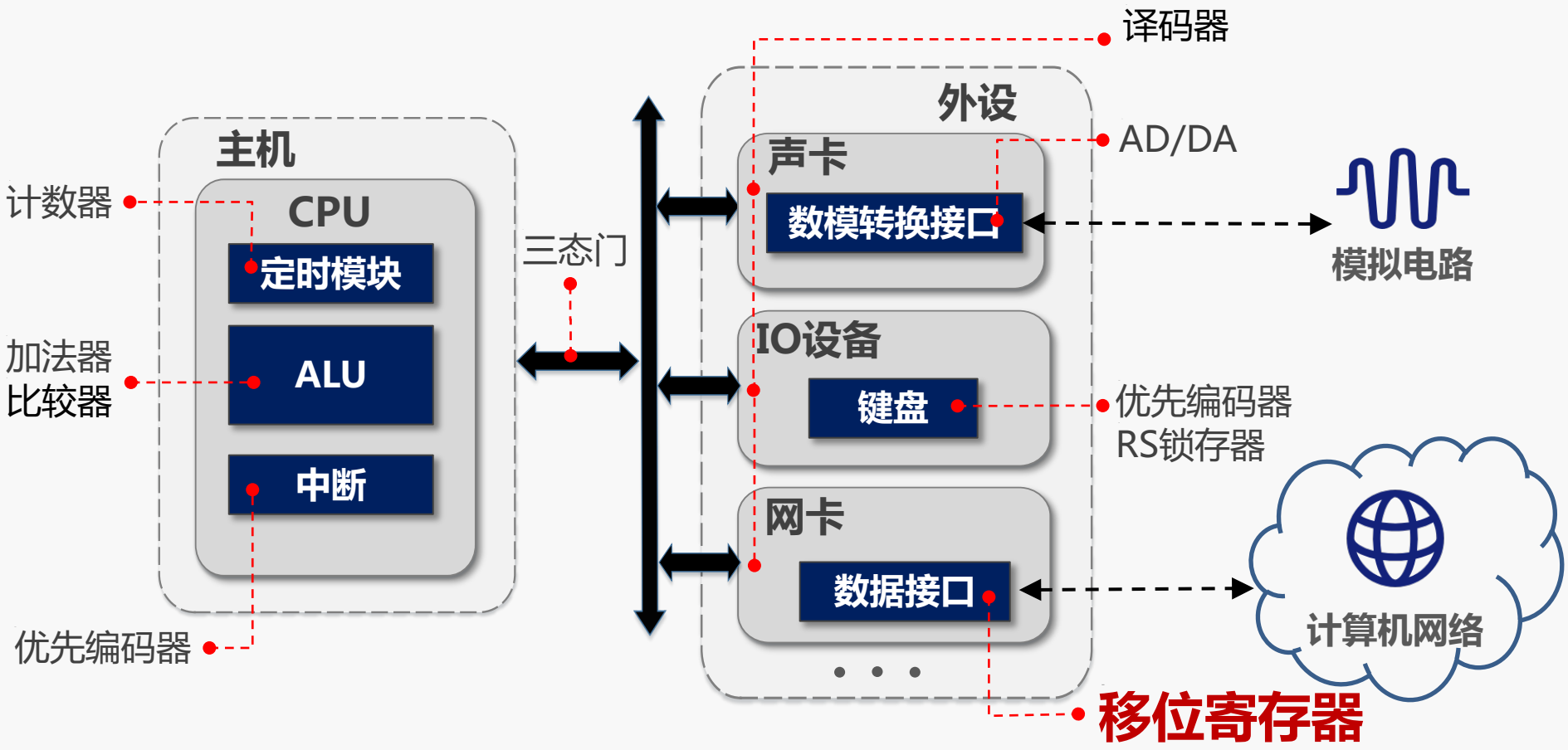
$$DSR \rightarrow Q_0 \rightarrow Q_1 \rightarrow Q_2 \rightarrow Q_3 \text{（ DSR ）}$$

（3）因为$Q0$~$Q3$中有一个必定为0，则$S1$=0，$S0$=1，74HCT194将始终工作在右移（低位向高位移动）循环移位的状态。




**输入波形该如何配合，来实现并/串转换呢？**

# 6.4 移位寄存器的应用


译码器
外设
主机
CPU
计数器
定时模块
加法器
比较器
ALU
中断
优先编码器
三态门
声卡
AD/DA
数模转换接口
模拟电路
IO设备
键盘
优先编码器
RS锁存器
网卡
数据接口
计算机网络
移位寄存器

# 6.4 移位寄存器的Verilog描述

**阻塞赋值“=”**：一个always块中的多条赋值语句的执行有先后顺序，前面的赋值语句执行完毕后，后面的赋值语句才可以执行，即**串行执行**。适用于**组合逻辑电路**的描述。

**非阻塞赋值“<=”**：一个always块中，所有的赋值语句同时计算并完成赋值操作，即**并行执行**。适用于**时序逻辑电路**的描述。

**串行执行**：只是为了**便于理解而做的语法设计**，并没有串行执行的实际电路。

**并行执行**：module中的多个always之间，多个assing赋值语句之间，always和assign之间。

**组合逻辑应使用阻塞式赋值**

```
always @ (a or b or c or d)
    begin
        tmp1 <= a & b;
        tmp2 <= c | d;
        y <= tmp1 | tmp2;
    end
```

**(1) 初始： a=1, b=0, c=0, d=0, tmp1=0, tmp2=0, y=0**

**(2) b=1： tmp1=1, tmp2=0, y=0**

**(3) b=0： tmp1=0, tmp2=0, y=1**

**组合逻辑应使用阻塞式赋值**

```
always @ (a or b or c or d or tmp1 or tmp2)
always@(*)
    begin
        tmp1 <= a & b;
        tmp2 <= c | d;
        y <= tmp1 | tmp2;
    end
```

**(1) 初始： a=1, b=0, c=0, d=0, tmp1=0, tmp2=0, y=0**

**(2) b=1： tmp1=1, tmp2=0, y=0**

**(3) tmp1=1： tmp1=1, tmp2=0, y=1**

**(4) b=0： tmp1=0, tmp2=0, y=1**

**(5) tmp1=0： tmp1=0, tmp2=0, y=0**

**组合逻辑应使用阻塞式赋值**

```
always @ (a or b or c or d)
    begin
        tmp1 = a & b;
        tmp2 = c | d;
        y = tmp1 | tmp2;
    end
```

**(1) 初始： a=1, b=0, c=0, d=0, tmp1=0, tmp2=0, y=0**

**(2) b=1： tmp1=1, tmp2=0, y=1**

**(3) b=0： tmp1=0, tmp2=0, y=0**

# 6.4 移位寄存器的Verilog描述

**组合逻辑与时序逻辑混用时，应使用非阻塞式赋值**

```
always @ (posedge clk or negedge rst)
    begin
        if (!rst) q<=0;
        else q<=a & b;
    end
```

```
always @ (a or b)
    y=a &b;
always @ (posedge clk or negedge rst)
    begin
        if (!rst) q<=0;
        else q<=y;
    end
```

# 6.4 移位寄存器的Verilog描述

```
always @ (posedge clk)    begin
        q1=d;
        q2=q1;
        q3=q2;
        q4=q3;
end
```

```
always @ (posedge clk)    begin
       q4=q3;
       q3=q2;
        q2=q1;
        q1=d;
   end
```

# 6.4 移位寄存器的Verilog描述

```
always @ (posedge clk)    begin
    q4<=q3;
    q3<=q2;
    q2<=q1;
    q1<=d;
    end
```

```
 always @ (posedge clk)
     begin
     q2<=q1;
     q4<=q3;
     q1<=d;
     q3<=q2;
    end
```

# 6.4 移位寄存器的Verilog描述

```
reg[3:0] q;
always @ (posedge clk)
   begin
   q[3:1]<=q[2:0];
   q[0]<=d;
   end
```

```
reg[3:0] q;
always @ (posedge clk)
   q[3:0]<={q[2:0],d};
```