数字系统设计

华东理工大学电子与通信工程系
主讲:木昌洪
Email：changhongmu@ecust.edu.cn

第7章 时序逻辑电路

本章介绍时序逻辑电路的描述方法和分析方法，具体介绍FSM、寄存器、移位寄存器、计数器等常用时序逻辑电路的工作原理、逻辑功能及使用方法，时序逻辑电路和数字系统的HDL设计方法。

- 7.1 概述
- 7.2 有限状态机
- 7.3 数码寄存器和移位寄存器
- 7.4 计数器
- 7.5 基于Verilog HDL的时序逻辑电路设计

本 章 重 点

- 时序逻辑电路的描述方法和分析方法；
- 有限状态机FSM的HDL设计方法；
- 常用时序逻辑电路的工作原理、逻辑功能及使用方法；
- 基于Verilog HDL的时序逻辑电路设计方法。

内容概要

7.1.1 时序逻辑电路的描述方法

7.1.2 时序逻辑电路的分析方法

7.1.1 时序逻辑电路的描述方法

数字逻辑

- 组合逻辑——由门电路构成，没有存储电路和反馈电路
- 时序逻辑——由组合逻辑电路和存储电路构成
- 程序逻辑——由控制电路(硬件)和程序数据(软件)构成
- 可编程逻辑——由用户定制构成各种类型的电路

时序逻辑电路分类

- **按存储单元状态改变的特点分类**
  - **同步时序逻辑电路: 构成时序逻辑电路的各级触发器受一个系统时钟统一控制。**
  - **异步时序逻辑电路: 构成时序逻辑电路的各级触发器**可以有各自的时钟信号，**不受系统时钟统一控制。**
    - 脉冲异步电路：记忆元件是触发器，电路的输入是脉冲信号
    - 电位异步电路：记忆元件由带反馈的门电路组成，电路的输入是电平信号

- **按输出信号的特点分类**

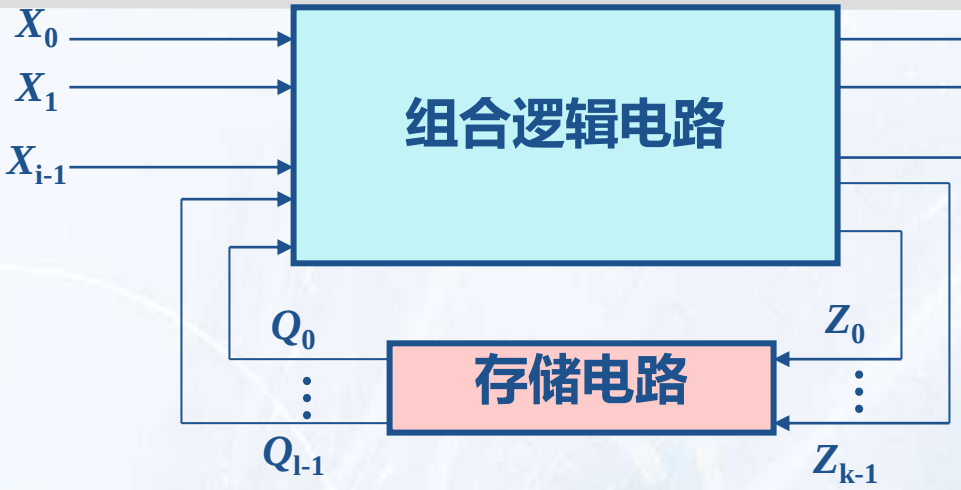
摩尔（Moore）型——时序电路的输出信号仅与电路当前状态有关

米里（Mealy）型——时序电路的输出信号与电路当前状态及输入信号有关

- **按时序电路的逻辑功能分类**

数码寄存器，移位寄存器，计数器

时序逻辑电路的描述方法



$$Y = f(X, Q) \qquad (7.1)$$

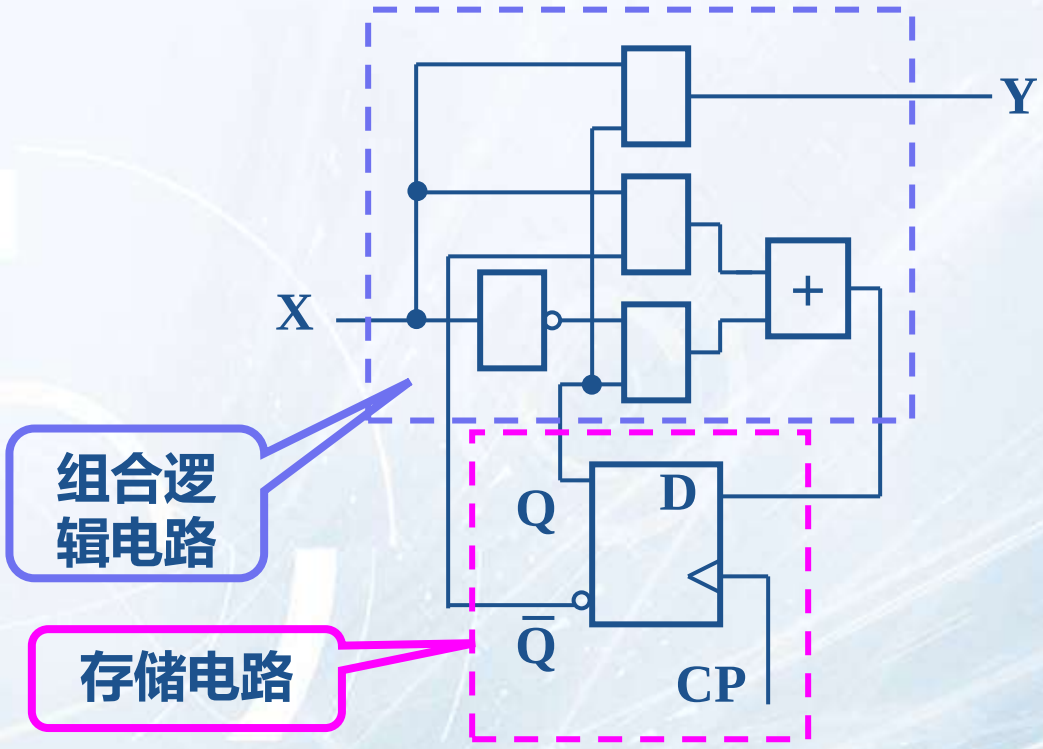
$$Z = g(X, Q) \qquad (7.2)$$

$$Q^{n+1} = h(Z, Q^n) \qquad (7.3)$$

- 时序逻辑电路可以用逻辑关系表达式（方程组）来描述。
  - X为电路的输入，Y为电路的输出， Q为存储电路的输出，式(7.1)称为输出方程：电路输出端的逻辑表达式；
  - Z为存储电路的输入，式(7.2)称为驱动方程（激励方程）：构成存储电路的触发器输入端的表达式；
  - 式(7.3)称为状态方程：表示触发器的状态变化特性，由驱动方程代入触发器的特性方程得到。

其他描述方法：状态转换表、状态转换图、时序图

同步时序逻辑电路举例1



输出方程： $Y=XQ^n$

驱动（激励）方程：

$D=X\overline{Q}^n+\overline{X}Q^n=X\oplus Q^n$

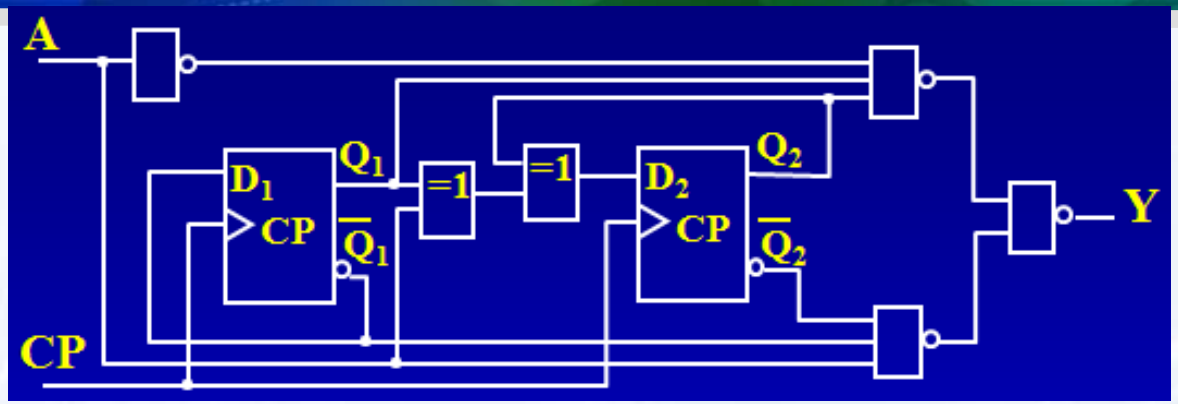
D FF特性方程：

$Q^{n+1}=D$

状态（特征）方程：

$Q^{n+1}=X\overline{Q}^n+\overline{X}Q^n=X\oplus Q^n$

同步时序逻辑电路举例2



输出方程： $Y=\overline{\overline{A}Q_1^nQ_2^n\,A\overline{Q}_1^n\overline{Q}_2^n}=\overline{A}Q_1^nQ_2^n+A\overline{Q}_1^n\overline{Q}_2^n$

驱动方程： $D_1=\overline{Q}_1^n$

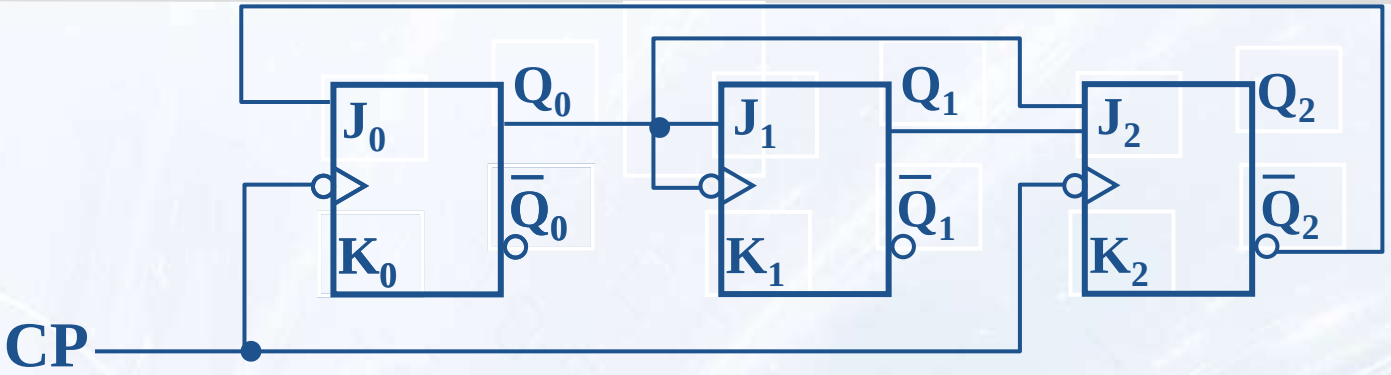
$D_2=A\oplus Q_1^n\oplus Q_2^n$

状态方程： $Q_1^{n+1}=\overline{Q}_1^n$

$Q_2^{n+1}=A\oplus Q_1^n\oplus Q_2^n$

- **同步**时序逻辑电路：有一个公共的时钟信号，电路中各记忆元件受它统一控制。 只有该时钟信号到来时，记忆元件的状态才能发生变化，从而使时序电路的输出发生变化。
- 每来一次时钟信号，记忆元件的状态和时序电路的输出才可能变化一次。

脉冲异步电路举例



时钟方程： $CP_0 = CP_2 = CP$　　$CP_1 = Q_0$

驱动方程： $J_0 = \overline{Q}_2^n$　　$J_1 = Q_0^n$　　$J_2 = Q_1^n Q_0^n$

$K_0 = 1$　　$K_1 = 1$　　$K_2 = 1$

状态方程：

$$\begin{cases} Q_0^{n+1} = \overline{Q}_2^n \overline{Q}_0^n & (CP\downarrow) \\ Q_1^{n+1} = \overline{Q}_1^n Q_0^n & (Q0\downarrow) \\ Q_2^{n+1} = \overline{Q}_2^n Q^n{}_1 Q_0^n & (CP\downarrow) \end{cases}$$

脉冲异步电路的分析过程

（1）状态方程：

$$Q_0^{n+1}=\overline{Q}_2^n\overline{Q}_0^n \quad （CP\downarrow）$$

$$Q_1^{n+1}=\overline{Q}_1^nQ_0^n \quad （Q0\downarrow）$$

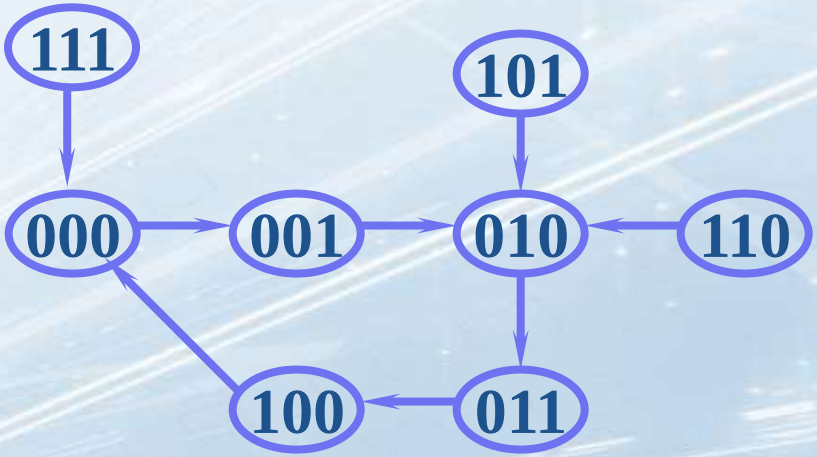
$$Q_2^{n+1}=\overline{Q}_2^nQ^n{}_1Q_0^n \quad （CP\downarrow）$$

（2）状态转换表

$Q_2^nQ_1^nQ_0^n$	$Q_2^{n+1}Q_1^{n+1}Q_0^{n+1}$
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	0 0 0
1 0 1	0 1 0
1 1 0	0 1 0
1 1 1	0 0 0

原态按照从小到大的顺序列出全部取值组合

（3）状态转换图



结论：电路为异步五进制加法计数器

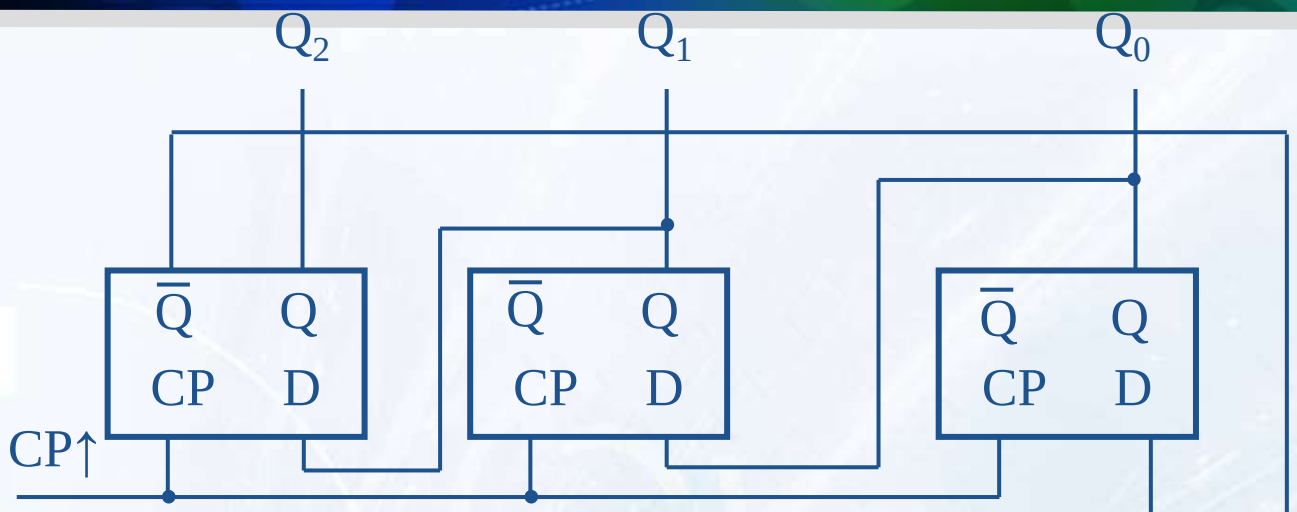
7.1.2 时序逻辑电路的分析方法

- **时序逻辑电路的分析**
  - **找出给定电路的逻辑功能，即找出在输入信号和时钟信号作用下的电路状态和输出信号的变化规律。**

时序逻辑电路的分析方法：

- 根据电路结构，写出方程式——时钟方程（异步时序电路）、输出方程、驱动方程和状态方程（将驱动方程代入触发器的特性方程得到）。
- 将输入变量和触发器初态的各种取值组合（按从小到大的顺序），代入状态方程和输出方程，计算出各级触发器的次态值和电路的输出值，得到状态转换表。
- 根据状态转换表，画状态转换图或时序图。
- 根据状态转换图和时序图，说明电路的逻辑功能。

【例7.1】分析下面电路的逻辑功能



驱动方程

$D_2 = Q_1^n$

$D_1 = Q_0^n$

$D_0 = \overline{Q}_2^n$

（1）写出驱动方程、状态方程和输出方程

- 根据电路结构，写出驱动方程；
- 将驱动方程代入触发器的特性方程得到状态方程；
- 由于该电路的输出就是各触发器的输出，所以输出方程同状态方程

D触发器的特性方程

$$Q^{n+1} = D$$

状态方程

$Q_2^{n+1} = D_2 = Q_1^n$

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

$Q_0^{n+1} = D_0 = \overline{Q}_2^n$

画出状态转换表

(2) 状态转换表

按从小到大的顺序写出触发器初态的各种取值组合

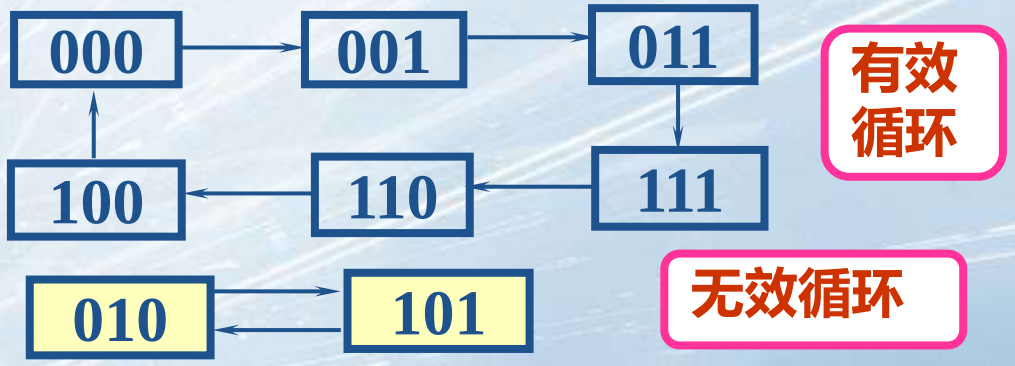
$Q_2^nQ_1^nQ_0^n$	$Q_2^{n+1}Q_1^{n+1}Q_0^{n+1}$
0 0 0	0 0 1
0 0 1	0 1 1
0 1 0	1 0 1
0 1 1	1 1 1
1 0 0	0 0 0
1 0 1	0 1 0
1 1 0	1 0 0
1 1 1	1 1 0

画出状态转换图

(3) 状态转换图

- 根据状态转换表，画出状态转换图
  - 从初始状态000画，在次态输出列中找到其对应下一状态的取值，画出；
  - 然后将其作为初态，在原态输出列中找到它，再在次态输出列中找到其对应下一状态的取值，画出；……
  - 确保每个状态在状态转换图中都应出现一次，且仅有一次。

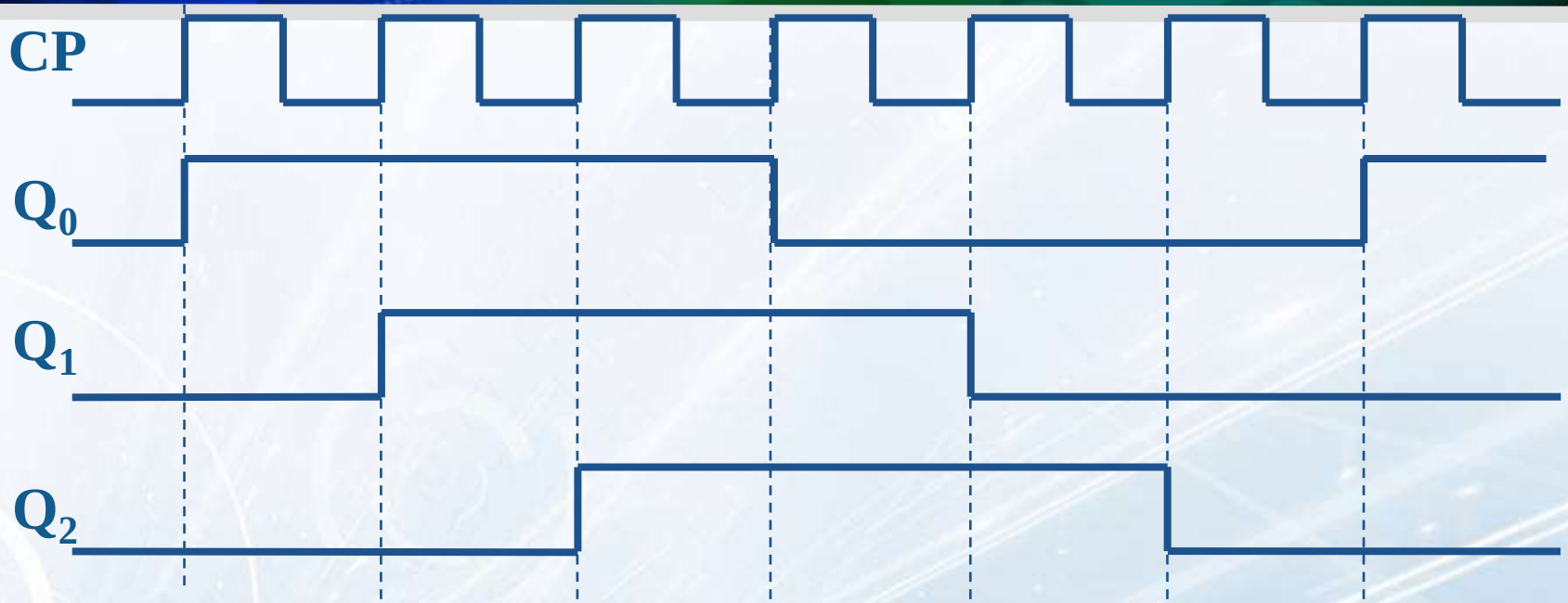
假设$Q_2Q_1Q_0$初始状态：0 0 0



计数器相关概念

- 几个概念
  - 有效状态（编码状态）——设计时选中进入计数循环的状态
  - 无效状态（非编码状态）——设计时未选中进入计数循环的状态（010、101）
  - 死循环（无效循环）——由无效状态构成的循环
  - 自启动——不存在死循环，计数循环以外的状态，都能回到计数循环中来

（4）时序图

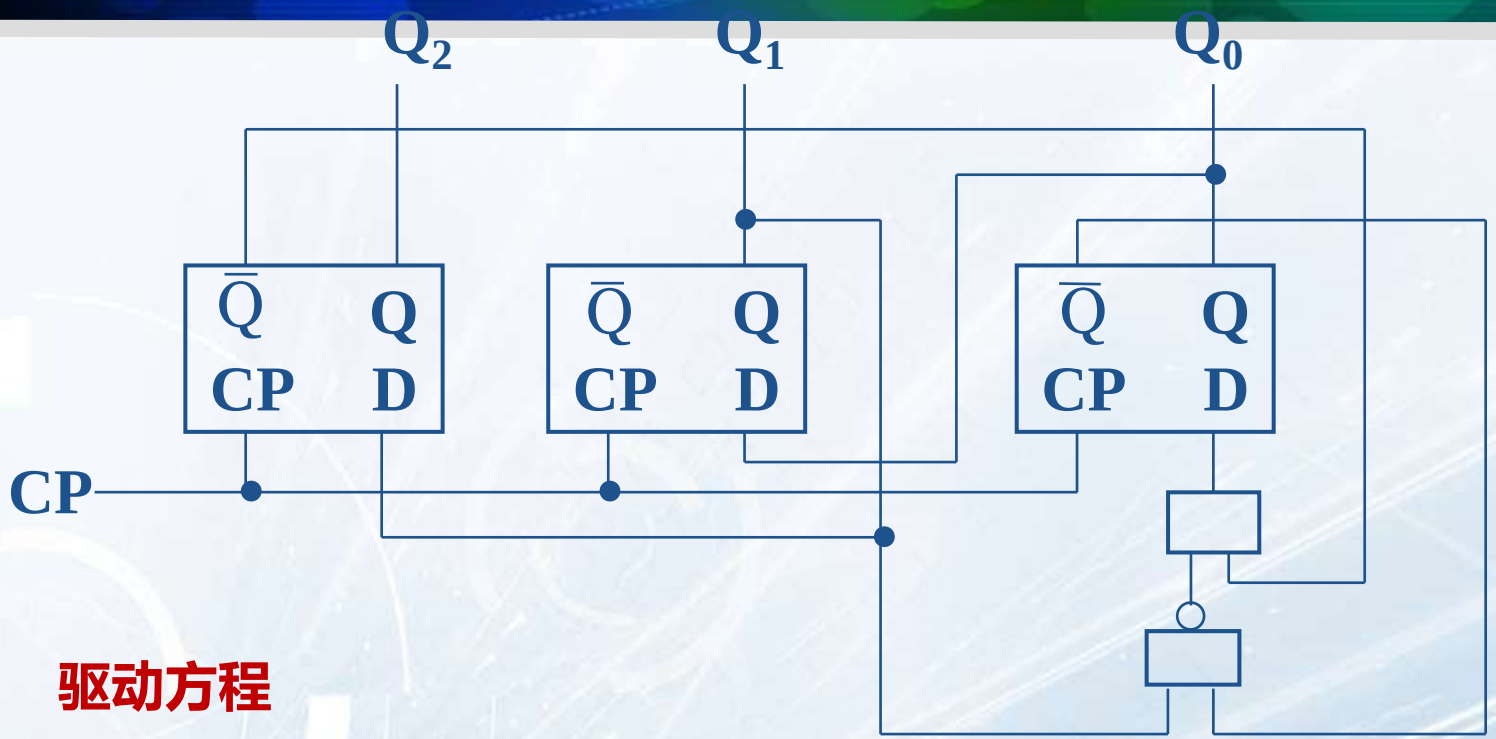


$Q_2Q_1Q_0$ 000 001 011 111 110 100 000

电路功能：同步3位格雷码计数器或同步六进制计数器

- 由于计数循环以外的状态不能回到计数循环里来，所以该计数器不能自启动

【例7.2】分析下面电路的逻辑功能



驱动方程

$D_2=Q_1^n$

$D_1=Q_0^n$

$D_0=\overline{Q}_0^n Q_1^n \overline{Q}_2^n=(Q_0^n+\overline{Q}_1^n)\overline{Q}_2^n$

D触发器的特性方程

$$Q^{n+1}=D$$

状态方程

$Q_2^{n+1}=Q_1^n$

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

$Q_0^{n+1}==\overline{Q}_0^n Q_1^n \overline{Q}_2^n=(Q_0^n+\overline{Q}_1^n)\overline{Q}_2^n$

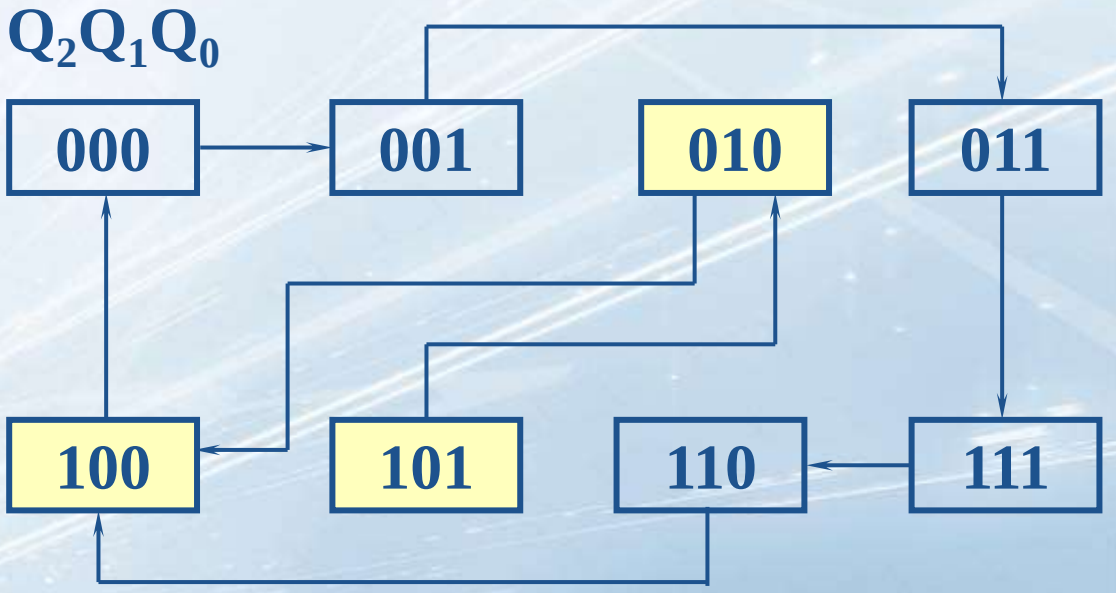
状态转换表和状态图

状态转换表

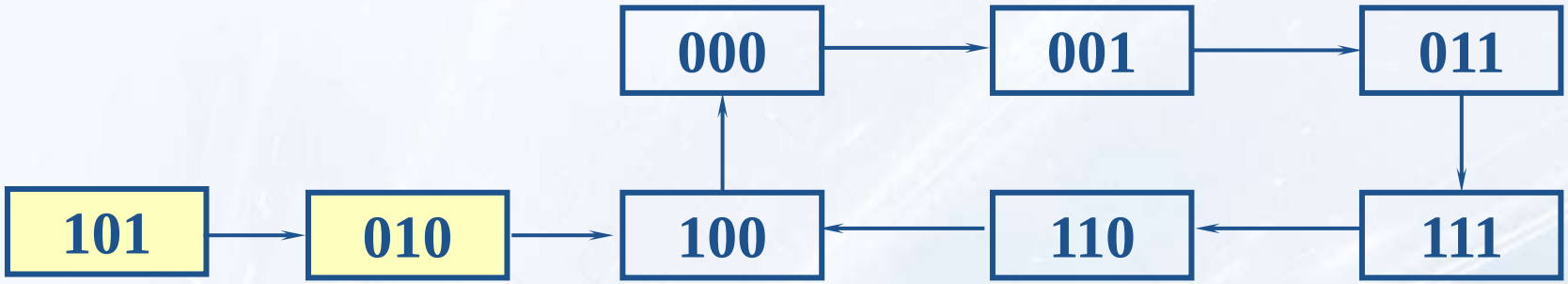
$Q_2^nQ_1^nQ_0^n$	$Q_2^{n+1}Q_1^{n+1}Q_0^{n+1}$
0 0 0	0 0 1
0 0 1	0 1 1
0 1 0	1 0 0
0 1 1	1 1 1
1 0 0	0 0 0
1 0 1	0 1 0
1 1 0	1 0 0
1 1 1	1 1 0

能够回到计数循环中来

状态转换图



最终状态图和电路功能



整理后的状态转换图

- 电路功能：能够自行进入工作循环的同步3位格雷码计数器（同步六进制计数器）
- 由于计数循环以外的状态都能回到计数循环里来，所以称该计数器为能自启动的同步六进制计数器

内容概要

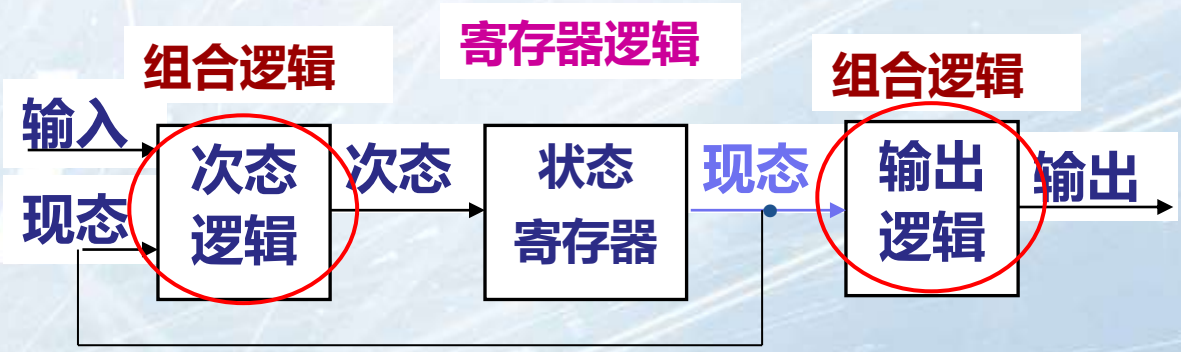
7.2.1 有限状态机概述

7.2.2 Moore型有限状态机

7.2.3 Mealy型有限状态机

7.2.1 有限状态机概述

- 有限状态机（Finite State Machine，FSM）是表示有限个状态以及这些状态之间的转移和动作等行为的离散数学模型。
- 有限状态机是组合逻辑和寄存器逻辑的特殊组合。组合逻辑部分包括次态逻辑（根据现态和输入产生次态）和输出逻辑，分别用于状态译码和产生输出信号；寄存器逻辑部分用于存储状态。

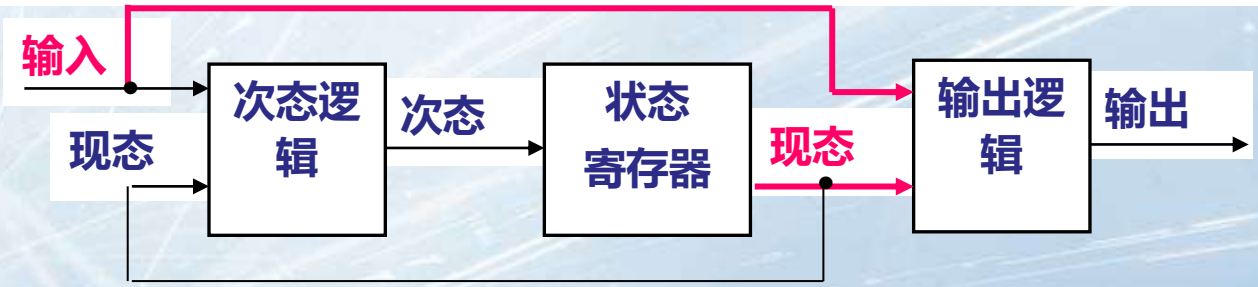


Moore型状态机典型结构

有限状态机的分类

- FSM常用于时序逻辑电路设计，尤其适于设计数字系统的控制模块。具有速度快、结构简单、可靠性高、逻辑清晰、复杂问题简单化的优点。

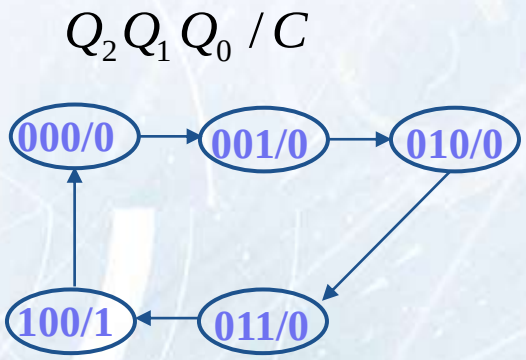
- 根据输出信号产生的机理不同，状态机可以分成两类：
  - **摩尔(Moore)型状态机**--输出信号仅与当前状态有关
  - **米里(Mealy)型状态机**--输出信号与当前状态及输入信号有关



Mealy型状态机的典型结构

有限状态机的表示方法

- 状态机有3种表示方法
  - 状态图（State Diagram）、状态表（State Table）和流程图
  - 三者等价，可以相互转换



状态（转换）图

状态（转换）表

$Q_2^nQ_1^nQ_0^n$	$Q_2^{n+1}Q_1^{n+1}Q_0^{n+1}$	C
0 0 0	0 0 1	0
0 0 1	0 1 0	0
0 1 0	0 1 1	0
0 1 1	1 0 0	0
1 0 0	0 0 0	1

有限状态机的设计方法

- 实用的状态机一般都设计为同步时序逻辑电路，它在同一个时钟信号的触发下，完成各状态之间的转移。
- 状态机设计步骤：
  1. 分析设计要求，列出全部可能状态；
  2. 画出状态转移图；
  3. 用Verilog HDL语言描述状态机，主要采用always块语句，完成3项任务：

（1）定义起始状态（敏感信号为时钟和复位信号）；

（2）用case或if-else语句描述出状态的转移（根据现态和输入产生次态）；

（3）用case或if-else语句描述状态机的输出信号（敏感信号为现态）。

状态机的设计要点

- 起始状态的选择

  起始状态指电路复位后所处的状态，选择一个合理的起始状态将使整个系统简捷高效。有限状态机必须有时钟信号和复位信号（建议采用异步复位）！

- 状态编码方式
  - 二进制编码（顺序编码）：采用顺序的二进制数编码每个状态，N个状态用$\log_2 N$个触发器来表示——节省逻辑资源，但可能产生毛刺
  - 格雷编码（Gray Code）： N个状态用$\log_2 N$个触发器来表示，在状态的顺序转换中，相邻状态每次只有一个比特位产生变化——节省逻辑资源；又避免产生毛刺
  - 一位热码编码（One-Hot Encoding）：采用N位二进制数（N个触发器）编码N个状态，每个状态下只有1位为“1”，其余位为“0”。相邻状态每次只有一个比特位产生变化——避免产生毛刺；但耗用逻辑资源多

对8个状态三种编码方式的对比

状态	二进制编码	格雷编码	一位热码编码
state0	000	000	00000001
state1	001	001	00000010
state2	010	011	00000100
state3	011	010	00001000
state4	100	110	00010000
state5	101	111	00100000
state6	110	101	01000000
state7	111	100	10000000

- 采用一位热码编码，虽然使用触发器较多，但可以有效节省和简化组合逻辑电路。
- FPGA有丰富的寄存器资源，门逻辑相对缺乏，采用一位热码编码可以有效提高电路的速度和可靠性，也有利于提高器件资源的利用率。

状态编码的HDL定义

- 状态编码的定义有两种方式：parameter和`define语句

【例】为state0, state1 ,state2 ,state3四个状态定义码字为：00，01，11，10

✓方式一：用parameter参数定义

用n个parameter常量表示n个状态

```
parameter   state0=2'b00 ,
            state1=2'b01,
            state2=2'b11,
            state3=2'b10;
            ……
case (state)
  state0:……;
  state1:……;
  ……
```

方式二：用`define语句定义

用n个宏名表示n个状态

```
`define  state0=2'b00 //不要加分号
`define  state1=2'b01
`define  state2=2'b11
`define  state3=2'b10
case (state)
`state0:……;
`state1:……;
    ……
```

状态转换的描述

- 一般用case、casez或casex语句，比用if-else语句更清晰明了！
- 实用的状态机都应设计为由唯一的时钟边沿触发的同步运行方式

- 多余状态（有效状态之外的状态）的处理
  - 在**case**语句中用**default**分支语句决定如果进入无效状态应采取的措施（如返回到起始状态）；
  - 编写必要的**Verilog**代码明确指定进入无效状态所采取的行为。

有限状态机的描述风格（1/3）

- 状态机设计中主要包含3个对象
  - 当前状态，现态（Current State）；
  - 下一个状态，次态（ Next State）；
  - 输出逻辑。

描述风格	功 能 划 分	所用进程数目
风格 A	1. 次态逻辑 2. 状态寄存器 3. 输出逻辑	3
风格 B	1. 次态逻辑、状态寄存器、输出逻辑	1
风格 C	1. 次态逻辑、状态寄存器（时序逻辑） 2. 输出逻辑（组合逻辑）	2
风格 D	1. 次态逻辑 2. 状态寄存器、输出逻辑	2
风格 E	1. 次态逻辑、输出逻辑 2. 状态寄存器	2

最常用

有限状态机的描述风格（2/3）

- 次态逻辑指实现状态的转换（根据现态和输入产生次态）；
- 状态寄存器指根据复位信号定义起始状态以及在时钟上升沿时将次态赋给现态（present<=next;）；
- 输出逻辑指Moore型状态机根据现态产生输出信号，或者Mealy型状态机根据现态和输入产生输出信号。

- 状态机的Verilog描述建议采用风格C（双过程）描述：一个过程描述现态和次态时序逻辑；另一个过程描述输出逻辑（组合逻辑）。这样写结构清晰；将时序逻辑和组合逻辑分开描述，便于修改。
- 描述包括2个always块：

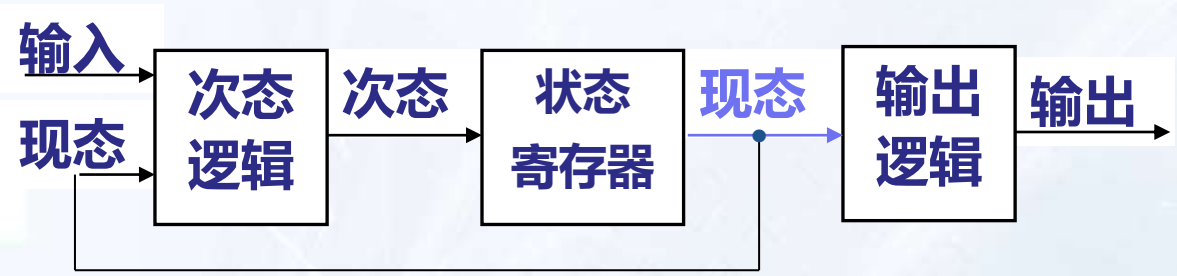
```
(1)  always@(posedge clk or posedge reset)
        if(reset) state=…;//复位时回到初始状态;
        else ……          //状态的转移;
(2)  always@(state)    //状态机的输出
```

有限状态机的描述风格（3/3）

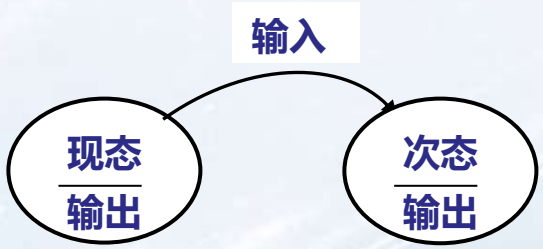
- 有时也采用风格B（单过程）描述方式，即将状态机的现态、次态和输出逻辑放在一个always块中进行描述。
- 其优点是采用时钟信号来同步输出信号，可以克服输出信号出现毛刺的问题，适于输出信号作为控制逻辑的场合使用，有效避免了因输出信号带有毛刺而产生错误的控制逻辑.
- 不足是输出信号会比双过程描述方式中的输出信号延迟一个时钟周期的时间。
- 描述只有1个always块：

```
always@(posedge clk or posedge reset)
  if(reset) state=…;//复位时回到初始状态;
  else                    //状态的转移和状态机的输出
    case(state)
      ……
```

7.2.2 Moore型有限状态机



Moore型状态机典型结构



Moore型状态图的表示

- Moore型状态机，其输出只为状态机当前状态的函数，而与外部输入无关。
- 外部输出是内部状态的函数。

Moore型有限状态机设计举例1

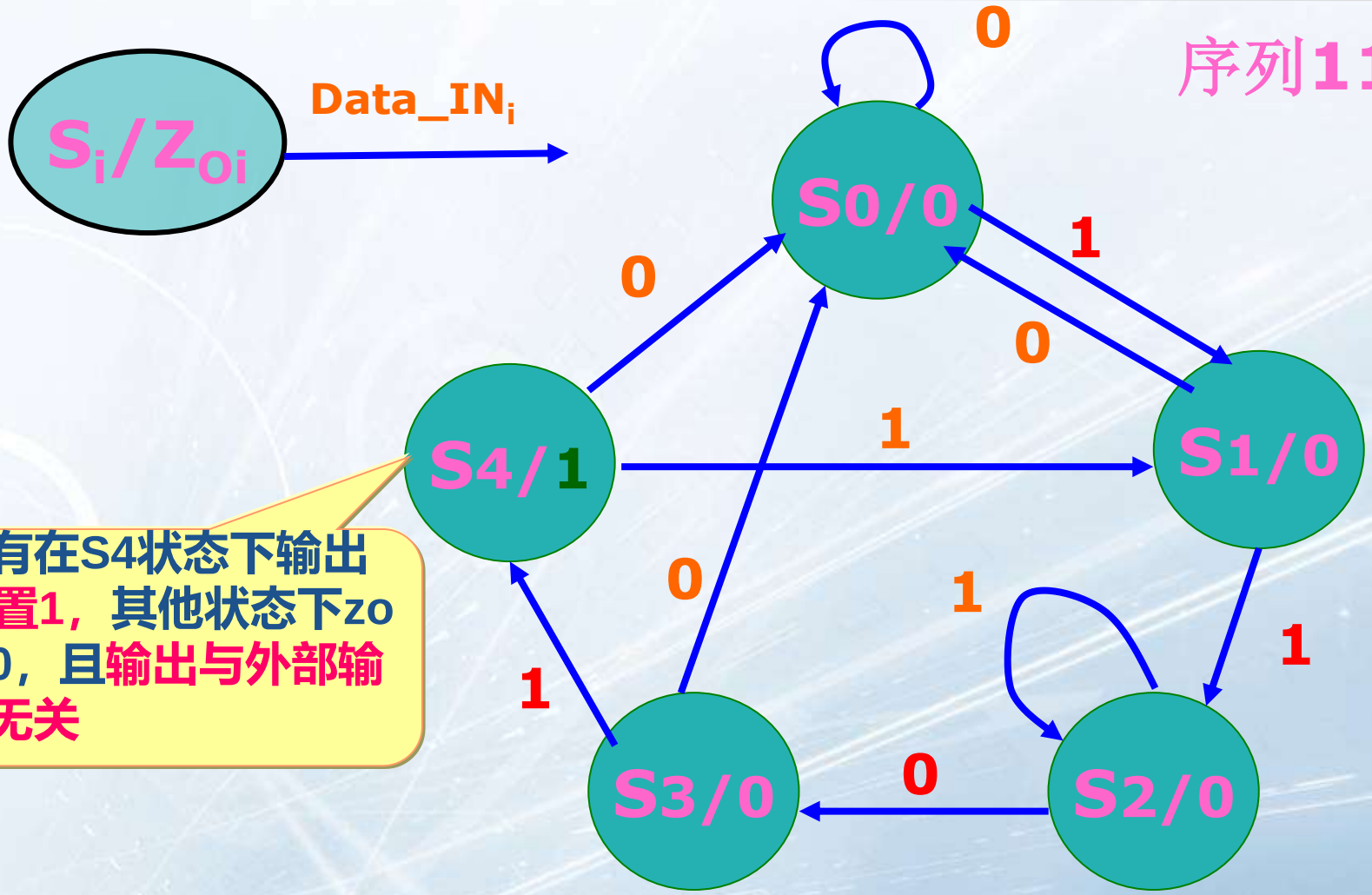
【例7.3】设计一个序列检测器。要求检测器连续收到串行码{1101}后，输出检测标志为1，否则输出检测标志为0。

第1步：分析设计要求，列出全部可能状态：

未收到一个有效位（0）	：	S0
收到一个有效位（1）	：	S1
连续收到两个有效位（11）	：	S2
连续收到三个有效位（110）	：	S3
连续收到四个有效位（1101）	：	S4

- **由于序列检测器的输出只为状态机当前状态的函数，而与外部输入无关，所以为Moore型状态机**

第2步：画出状态图


Si/Zoi
Data_INi
序列1101
S0/0
S1/0
S2/0
S3/0
S4/1
0
1
0
1
0
1
0
1
0
1
只有在S4状态下输出zo置1，其他状态下zo为0，且输出与外部输入无关

第3步：用Verilog语言描述状态机

- 在程序的开头定义状态机状态的编码形式
  - 用parameter或`define语句
- 复位时回到起始状态
  - 敏感信号为时钟和复位信号
- 状态转换描述
  - 用case或if-else语句描述出状态的转移（根据现态和输入产生次态，可与复位时回到起始状态的语句放在同一个always块中，即敏感信号为时钟和复位信号）
- 输出信号描述
  - 用case语句（Mealy型状态机还要用到if-else语句）描述状态机的输出信号（单独放在一个always块中，敏感信号为现态）

序列检测器源程序（Moore型状态机）

```
module monitor(clk,clr,data,zo,state);
   parameter S0=3'b000, S1=3'b001,
   S2=3‘b010,S3=3’b011,S4=3‘b100; //状态编码的定义
   input clk,clr,data;
   output zo;
   output[2:0] state;                          //状态机
   reg [2:0] state;
   reg zo;
   always @(posedge clk or posedge clr)
     begin
        if (clr) state=S0; //（1）复位时回到初始状态
        else
          begin
            case (state)//  （2）状态的转移
               S0: if (data==1’b1) state=S1;
                     else state=S0;
               S1: if (data==1’b1) state=S2;
                     else state=S0;
               S2: if (data==1’b0) state=S3;
                     else state=S2;
               S3: if (data==1’b1) state=S4;
                    else state=S0;
               S4: if (data==1’b1) state=S1;
                    else state=S0;
               default: state=S0;
            endcase
            zo=(state==S4)?1’b1:1’b0;  //（3）状态机的输出信号
          end
     end
endmodule
```

在S4时给zo置1——输出只为状态机当前状态的函数，而与外部输入无关

状态机的输出信号描述

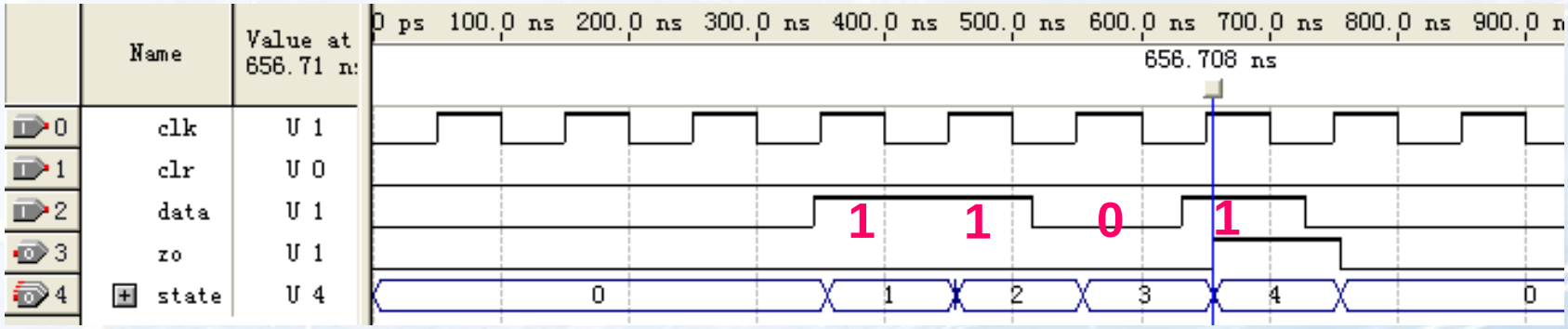
- 如果输出表达式很简单，可以单独用一条赋值语句写出
- 最好将状态机的输出语句单独写在另一个always块中（纯组合逻辑），这样逻辑清晰，不易出错：

语句“zo=(state==S4)?1’b1:1’b0;”去掉，换成如下always块（组合逻辑），monitor_good.v：

```
always @(state)      // (3) 状态机的输出信号
   begin
      case (state)
         S0:  zo=1'b0;
         S1:  zo=1'b0;
         S2:  zo=1'b0;
         S3:  zo=1'b0;
         S4:  zo=1'b1;
         default: zo=1'b0;
      endcase
   end
```

返回

序列检测器的仿真波形

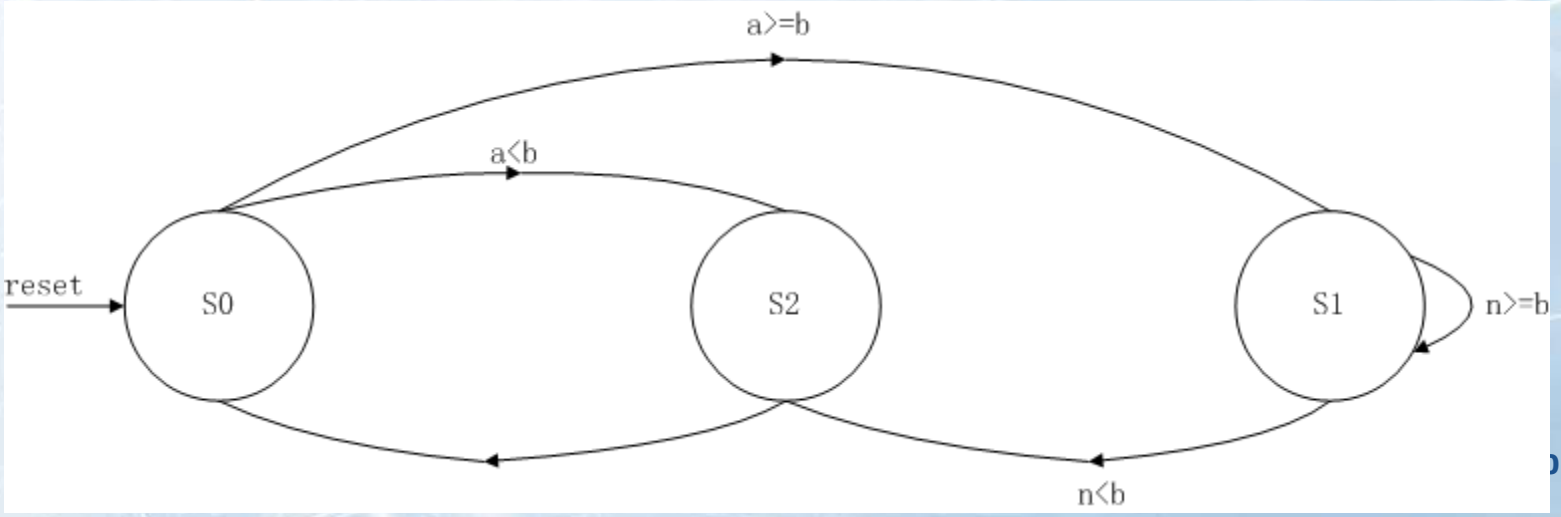


返回

Moore型有限状态机设计举例2

【例7.4】用有限状态机设计4位二进制数的除法电路（单过程）

- 设计思路：电路采用状态机来设计，一共包括3个状态
  - S0：判断被除数是否大于除数，若是，得到第1次的商和余数，跳转到S1；若不是，则商为0，余数等于被除数，跳转到S2
  - S1：进行除法运算，当余数小于被除数时，跳转到S2
  - S2：得到运算结果，然后返回S0



除法电路源程序

```
module division(a, b, clk, reset, result, yu, state);
    input[3:0] a, b;                    // 被除数和除数
    input clk, reset;
    output reg[3:0] result, yu;         // 最终结果：商和余数
    output reg[1:0] state;
    reg[3:0] m, n;                      // 每步除法运算的商和余数
    parameter S0=2'b00, S1=2'b01, S2=2'b10;//状态编码（顺序编码）
    always @(posedge clk or posedge reset)
      begin
        if(reset)
          begin
            m<=4'b0000; n<=4'b0000;
            result<=4'b0000; yu<=4'b0000;
            state<=S0;
          end
```

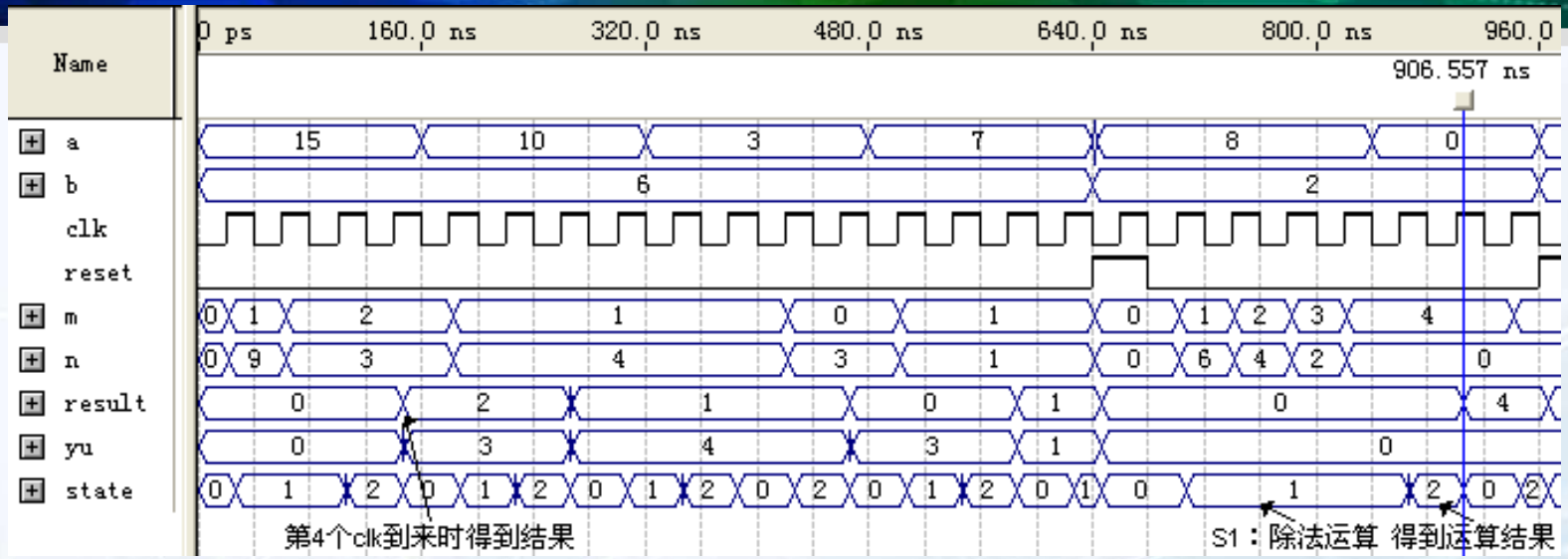
单过程

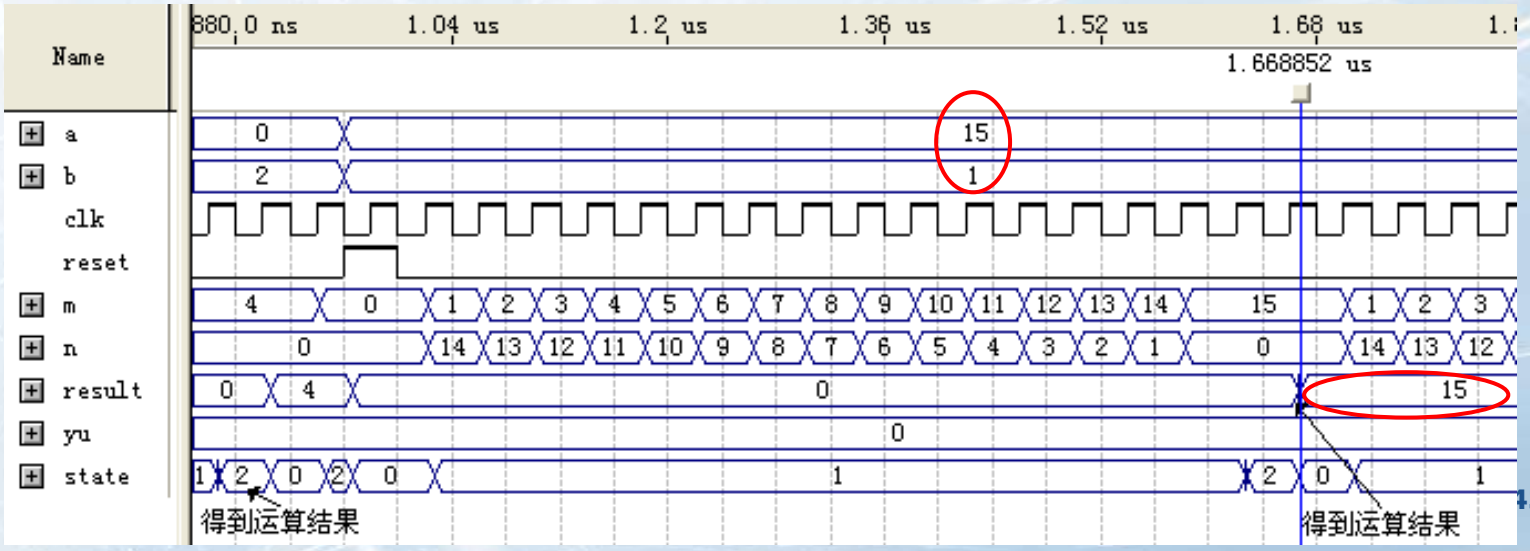
除法电路源程序（续）

```
    else
      case(state)     //次态逻辑和输出逻辑
          S0: begin //判断被除数是否大于除数
                  if (a>=b) begin m<=4'b0001; n<=a-b; state<=S1; end
                   else begin m<=4'b0000; n<=a; state<=S2; end
                end
          S1: begin //进行除法运算
                   if (n>=b) begin m<=m+1'b1; n<=n-b; state<=S1; end
                   else begin state<=S2; end
                end
          S2: begin //得到运算结果
                   result<=m; yu<=n; state<=S0;
                end
           default: state<=S0;
       endcase
     end
endmodule
```

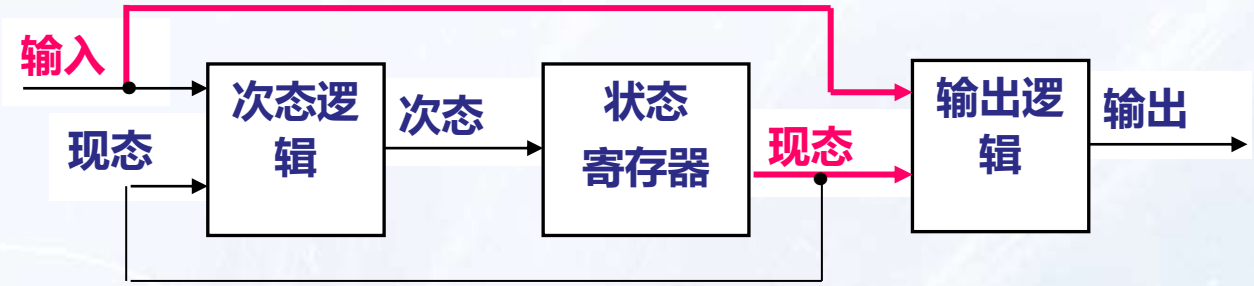
对多余状态的处理

除法电路仿真波形


Name
0 ps
160.0 ns
320.0 ns
480.0 ns
640.0 ns
800.0 ns
960.0
906.557 ns
a
b
clk
reset
m
n
result
yu
state
第4个clk到来时得到结果
S1：除法运算 得到运算结果


Name
880.0 ns
1.04 us
1.2 us
1.36 us
1.52 us
1.68 us
1.668852 us
a
b
clk
reset
m
n
result
yu
state
得到运算结果
得到运算结果

7.2.3 Mealy型有限状态机



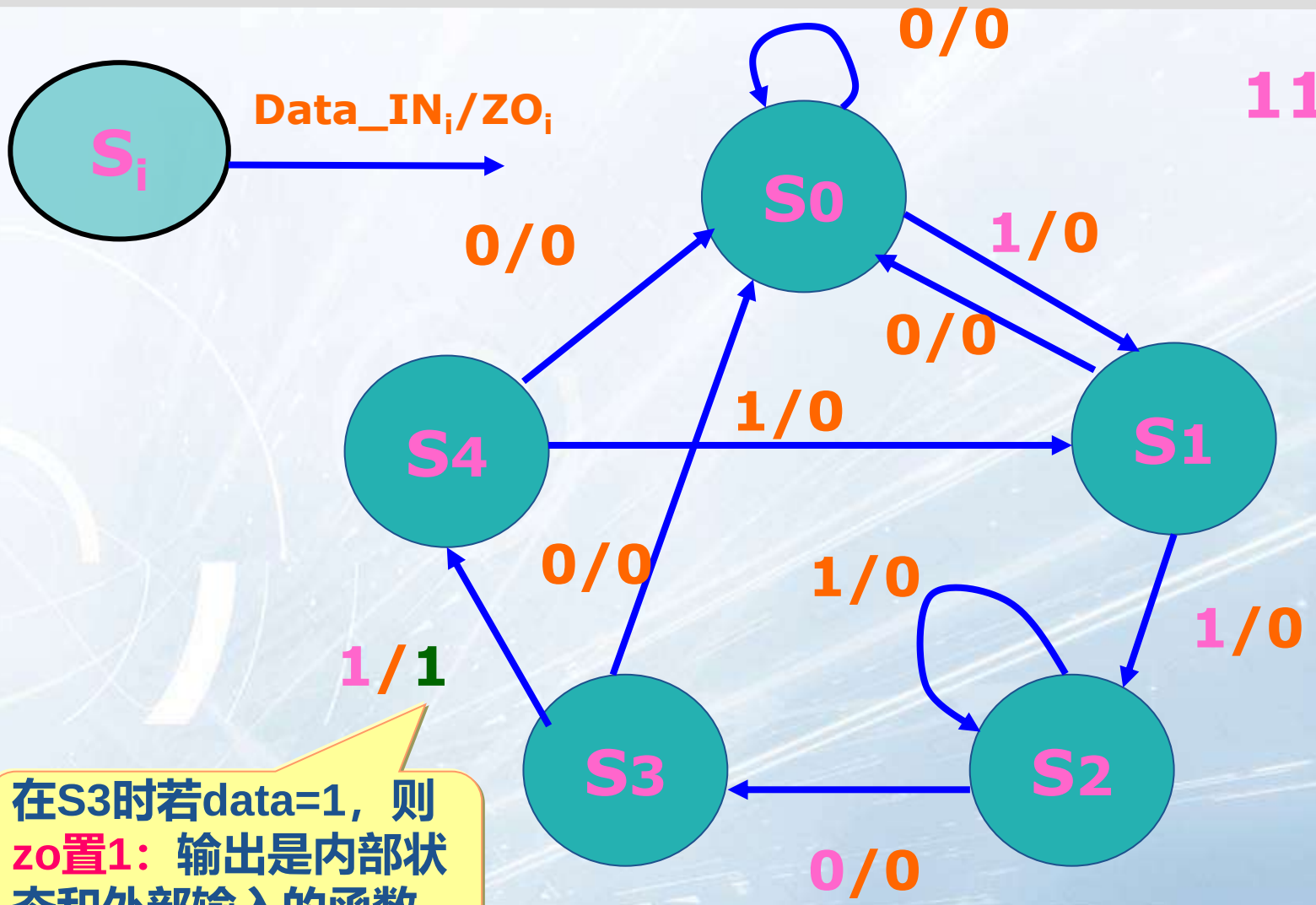
Mealy型状态机的典型结构

Mealy型状态图的表示

- Mealy型状态机，其输出不仅与状态机**当前状态**有关，而且与**输入**有关——外部输出是内部状态和外部输入的函数。
- 在“Mealy型状态图的表示”中，每个圆圈表示状态机的一个状态，每个箭头表示状态之间的一次转移；引起状态转换的输入信号及产生的输出信号标注在箭头上。

【例7.5】将【例7.3】采用Mealy型状态机实现。

序列检测器的Mealy型有限状态机状态图



在S3时若data=1，则zo置1：输出是内部状态和外部输入的函数

序列检测器源程序（Mealy型状态机）

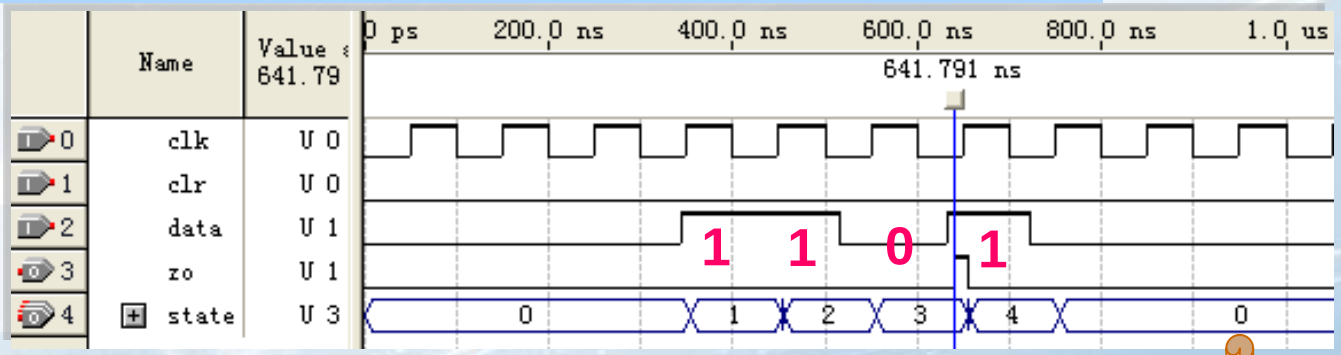
```
module monitor2_good(clk,clr,data,zo,state);
   parameter S0=3'b000, S1=3'b001,
   S2=3'b010,S3=3'b011,S4=3'b100;
   input clk,clr,data;
   output zo;
   output[2:0] state;
   reg [2:0] state;
   reg zo;
   always @(posedge clk or posedge clr)
     begin
       if (clr) state=S0; // （1）复位时回到初始状态
       else
         begin
           case (state) // （2）状态的转移
             S0:  if (data==1'b1) state=S1;
                  else state=S0;
             S1:  if (data==1'b1) state=S2;
                  else state=S0;
             S2:  if (data==1'b0) state=S3;
                  else state=S2;
             S3:  if (data==1'b1) state=S4;
                  else state=S0;
             S4:  if (data==1'b1) state=S1;
                  else state=S0;
             default: state=S0;
           endcase
         end
     end
endmodule
```

序列检测器源程序及仿真波形

```
always @(state)      //（3）状态机的输出信号
   begin
      case (state)
         S0:  zo=1'b0;
         S1:  zo=1'b0;
         S2:  zo=1'b0;
         S3:  if(data==1'b1) zo=1'b1;
              else zo=1'b0;
         S4:  zo=1'b0;
         default: zo=1'b0;
      endcase
   end
endmodule
```

与Moore型比较

在S3时若data=1，则zo置1：输出是内部状态和外部输入的函数



- 采用Mealy型状态机描述序列检测器与采用Moore型状态机的仿真波形稍有区别！

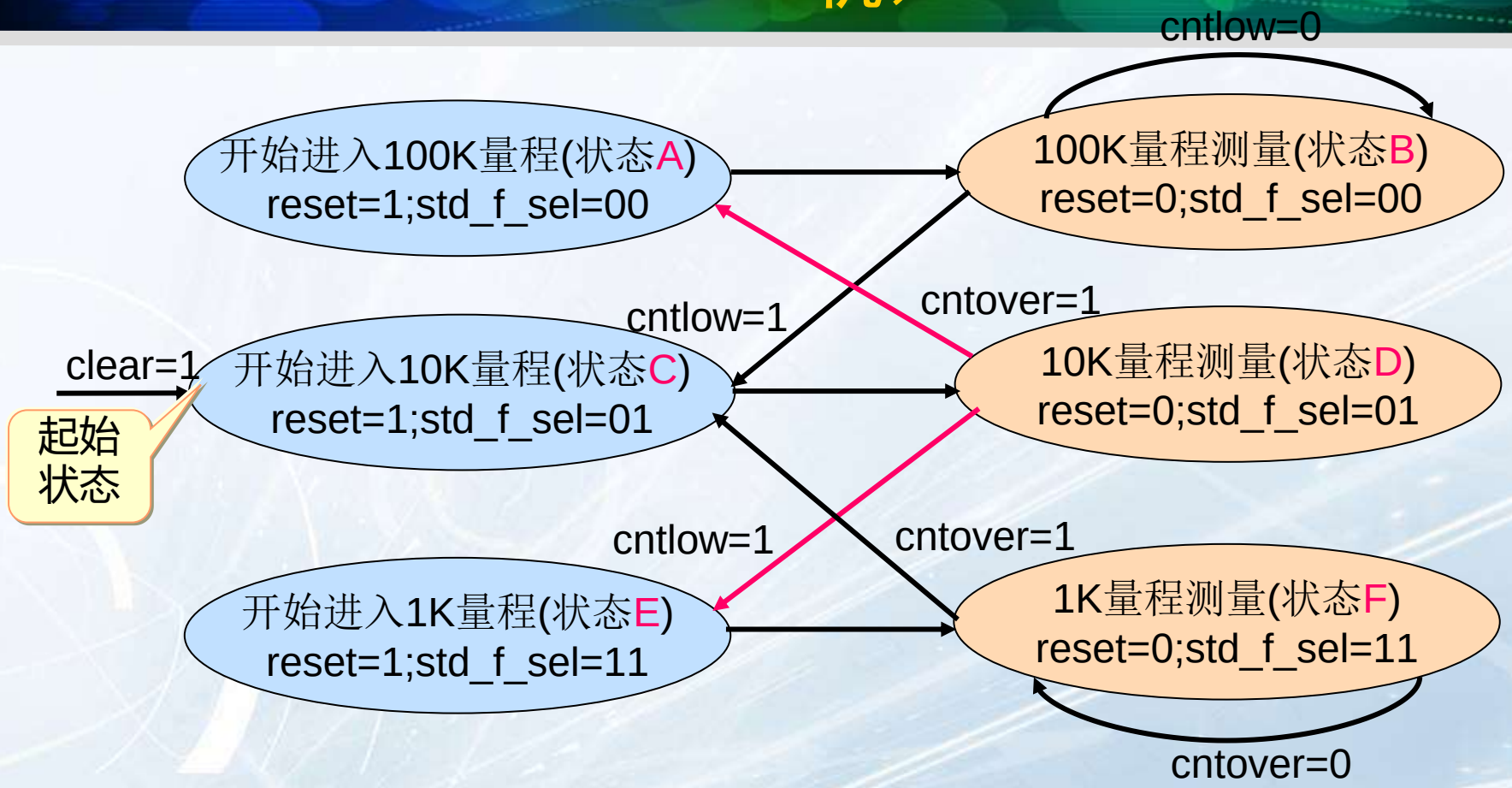
波形比较

【例7.6】自动转换量程频率计控制器

【例7.6】设计一个自动转换量程频率计控制器，要求根据超量程或欠量程输入信号，自动切换到合适的量程；并输出复位频率计计数器的信号，以及选择标准时基的信号

- **假定频率计有3个量程：1K、10K、100K**
- **控制器有6个工作状态：进入100K量程、 100K量程测量、进入10K量程、 10K量程测量、进入1K量程、 1K量程测量。**
- **输入信号**
  - **clk：系统时钟;**
  - **clear：系统复位信号;**
  - **cntover：超量程;**
  - **cntlow：欠量程。**
- **输出信号**
  - **reset：转换到某量程时复位频率计计数器信号;**
  - **std_f_sel：选择标准时基信号**

频率计控制器状态转移图（Moore型状态机）



- **起始状态选为一个中间量程，为C（进入10K量程）。**
- **从“进入某档量程”转移到“该档量程测量”为无条件转移；而当在测量时，根据超量程或欠量程，切换到适宜的量程。**

频率计控制器源程序（1/3）

```
module frequency_right(clk, clear, cntover, cntlow, reset, std_f_sel,state);
    input clk, clear, cntover, cntlow;
    output reset;                  //量程转换开始时复位频率计计数器
    output[1:0] std_f_sel;  //选择标准时基
    output[5:0] state;
    reg [5:0] state;
    reg reset;
    reg[1:0] std_f_sel;
    parameter start_f100k=6'b000001, f100k_cnt=6'b000010,
                  start_f10k=6'b000100, f10k_cnt=6'b001000,
                  start_f1k=6'b010000, f1k_cnt=6'b100000;
// ① 复位时回到起始状态
  always@(posedge clk or posedge clear)
    if(clear) state=start_f10k;//复位时“进入10K量程”为起始状态
    else
```

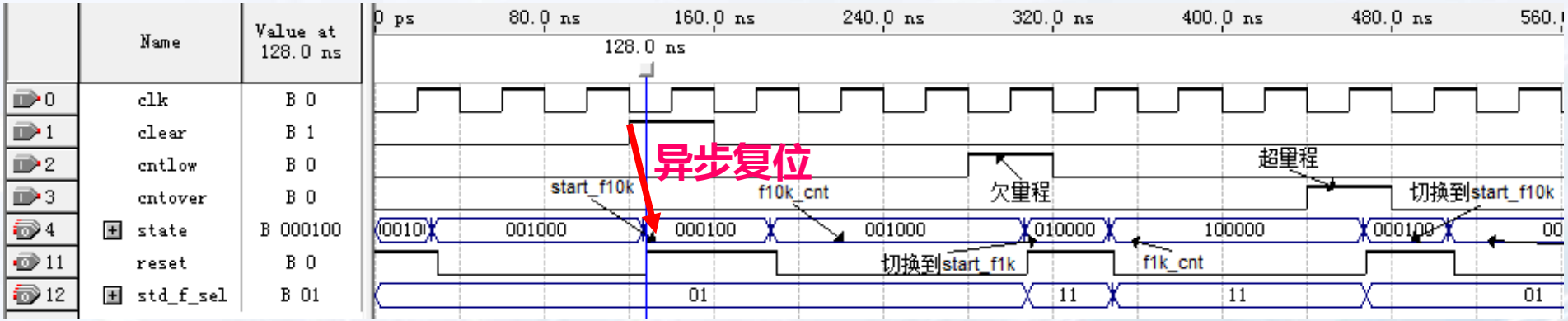
频率计控制器源程序（2/3）

```
// ② 状态的转换（根据现态和输入产生次态）
    begin
        case(state)
            start_f100k: state=f100k_cnt;
            f100k_cnt:    if (cntlow) state= start_f10k;     //欠量程
                          else state= f100k_cnt;
            start_f10k:   state=f10k_cnt;
            f10k_cnt:     if (cntover) state= start_f100k;   //超量程
                          else if (cntlow) state= start_f1k; //欠量程
            start_f1k:    state=f1k_cnt;
            f1k_cnt:      if (cntover) state= start_f10k;    //超量程
                          else state= f1k_cnt;
            default: state=start_f10k;   //默认状态为起始状态“进入10K量程”
        endcase
    end
```

频率计控制器源程序（3/3）

```
// ③ 状态机的输出信号
  always@(state)
      begin
          case(state)
              start_f100k:  begin reset=1; std_f_sel=2'b00; end
              f100k_cnt:    begin reset=0; std_f_sel=2'b00; end
              start_f10k:   begin reset=1; std_f_sel=2'b01; end
              f10k_cnt:     begin reset=0; std_f_sel=2'b01; end
              start_f1k:    begin reset=1; std_f_sel=2'b11; end
              f1k_cnt:      begin reset=0; std_f_sel=2'b11; end
              default:      begin reset=1; std_f_sel=2'b01; end
          endcase
    end
endmodule
```

频率计控制器时序仿真波形



内容概要

7.3.1 数码寄存器

7.3.2 移位寄存器

7.3.3 集成移位寄存器

54

寄存器的定义与分类

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。它被广泛用于各类数字系统和数字计算机中。
- 由有记忆功能的触发器组成；还有一些接收数据的控制门，以便在同一个接收命令作用下使各触发器同时接收数据。
- 一个触发器能储存1位二值代码，用N个触发器组成的寄存器能储存一组N位的二值代码
- 触发器的触发方式决定了寄存器的触发方式。数码寄存器常用的是上升沿触发的D型触发器（边沿触发）和电位触发器，较少采用主-从触发器。
- 寄存器的操作：读/写/复位（清零）

寄存器
- 数码寄存器
  - 数据寄存器（多位D型触发器组成）
  - 数据锁存器（多位电位触发器组成）
  - 寄存器阵列
- 移位寄存器
  - 单向移位寄存器
  - 双向移位寄存器

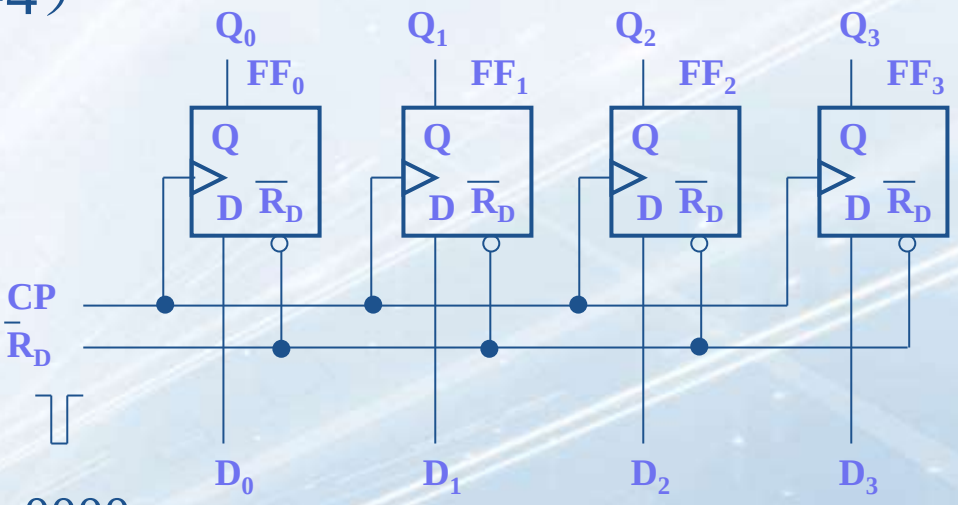
7.3.1 数码寄存器

- 数码寄存器具有接收、存放和传输数码的功能。
- 各种类型的触发器都具有置0、置1和保持（记忆）功能，它们都可以用来构成寄存器，而用D触发器和D锁存器构成寄存器最为方便。

- 4位D型寄存器电路结构（N=4）

$D_3D_2D_1D_0$: 并行数据输入

$Q_3Q_2Q_1Q_0$: 并行数据输出



- 工作原理

（1）清除（复位）

当 $\overline{R_D}=0$， $Q_0Q_1Q_2Q_3$=0000;

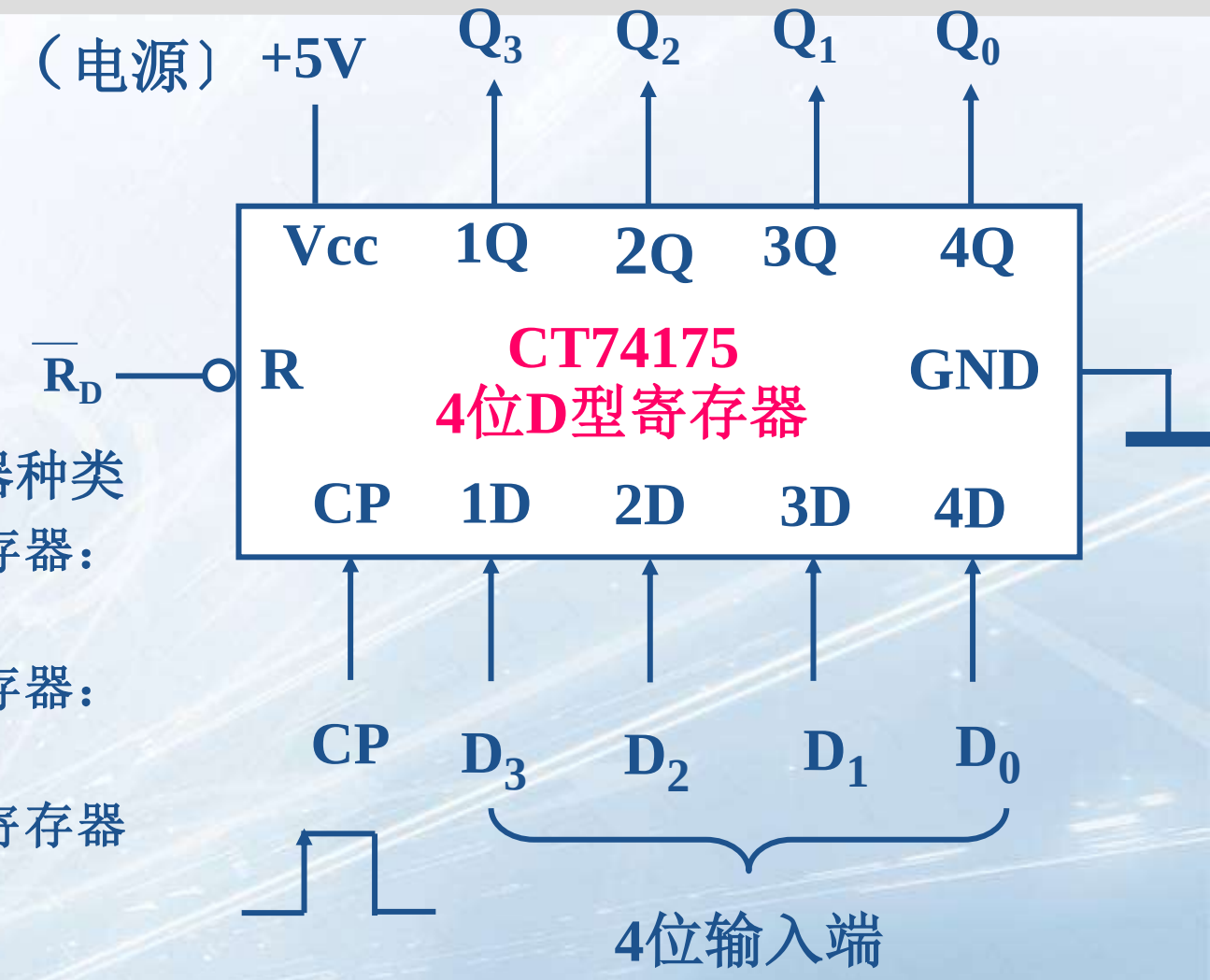
（2）置数（复位端无效时）

当 $CP=CP\uparrow$，$Q_0Q_1Q_2Q_3$= $D_0D_1D_2D_3$;

- 工作方式（数据输入输出方式）——只能并入并出

56

4位二进制数寄存器CT74175



- 常见集成数码寄存器种类

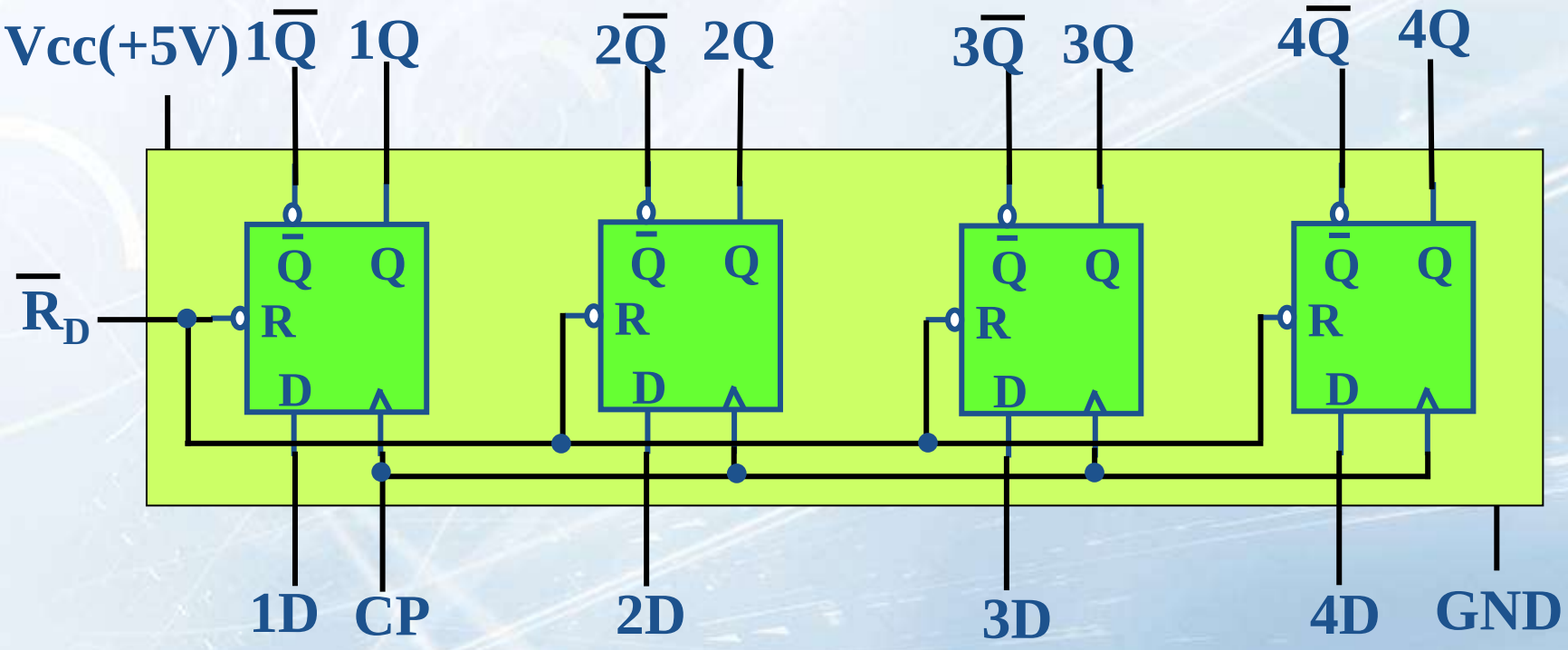
1、TTL型4位D型寄存器：CT74173、CT74175

2、TTL型8位D型寄存器：CT74273、CT74373

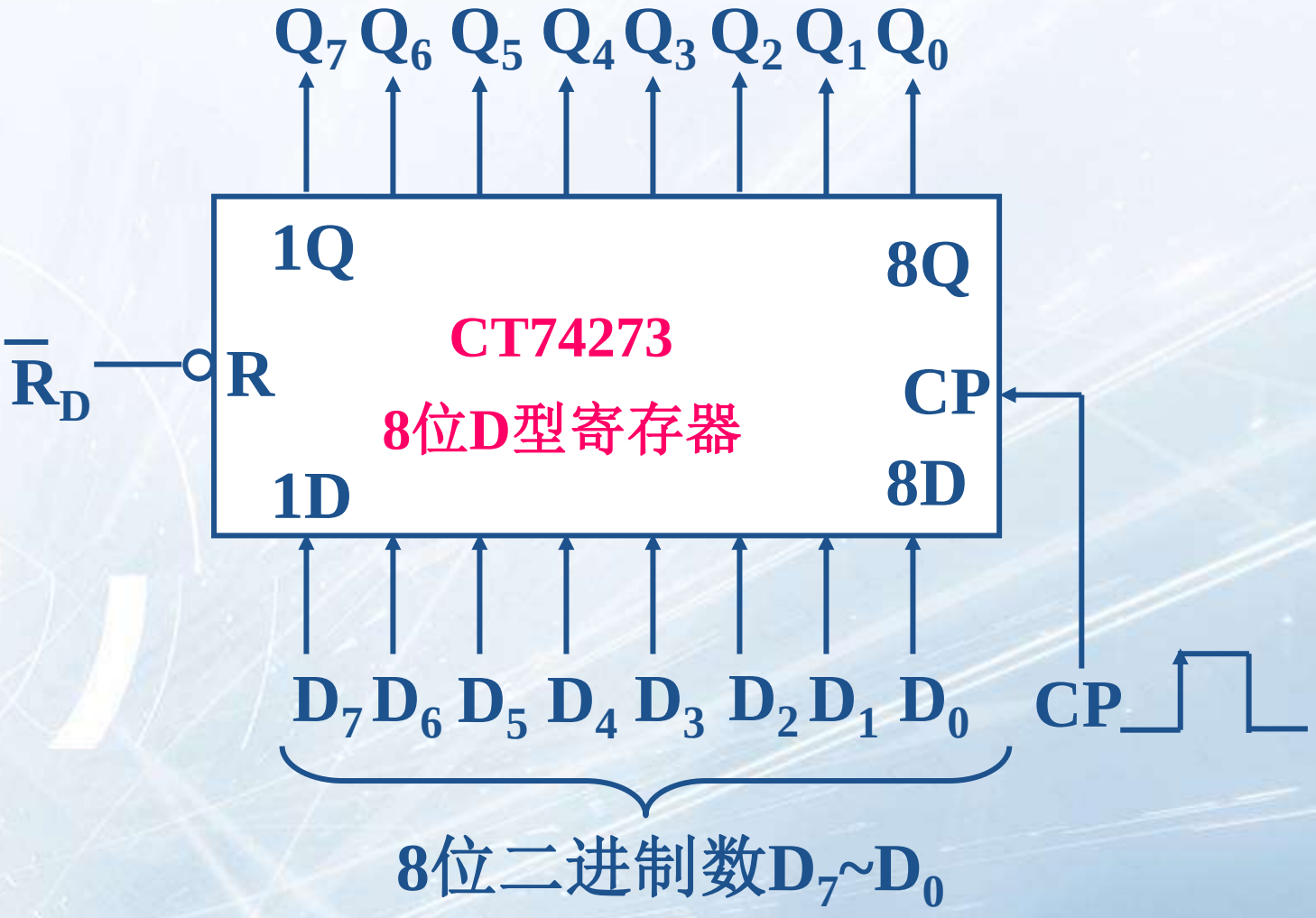
3、CMOS型4位D型寄存器：CC4076

CT74175电路结构

- 由边沿触发器组成
- 4级D FF的输入构成4位数码输入端1D、2D、3D、4D，输出构成4位数码输出端
- 各触发器的时钟连接在一起，作为整个寄存器的时钟端CP
- 异步置0端连在一起，作为整个寄存器的复位端

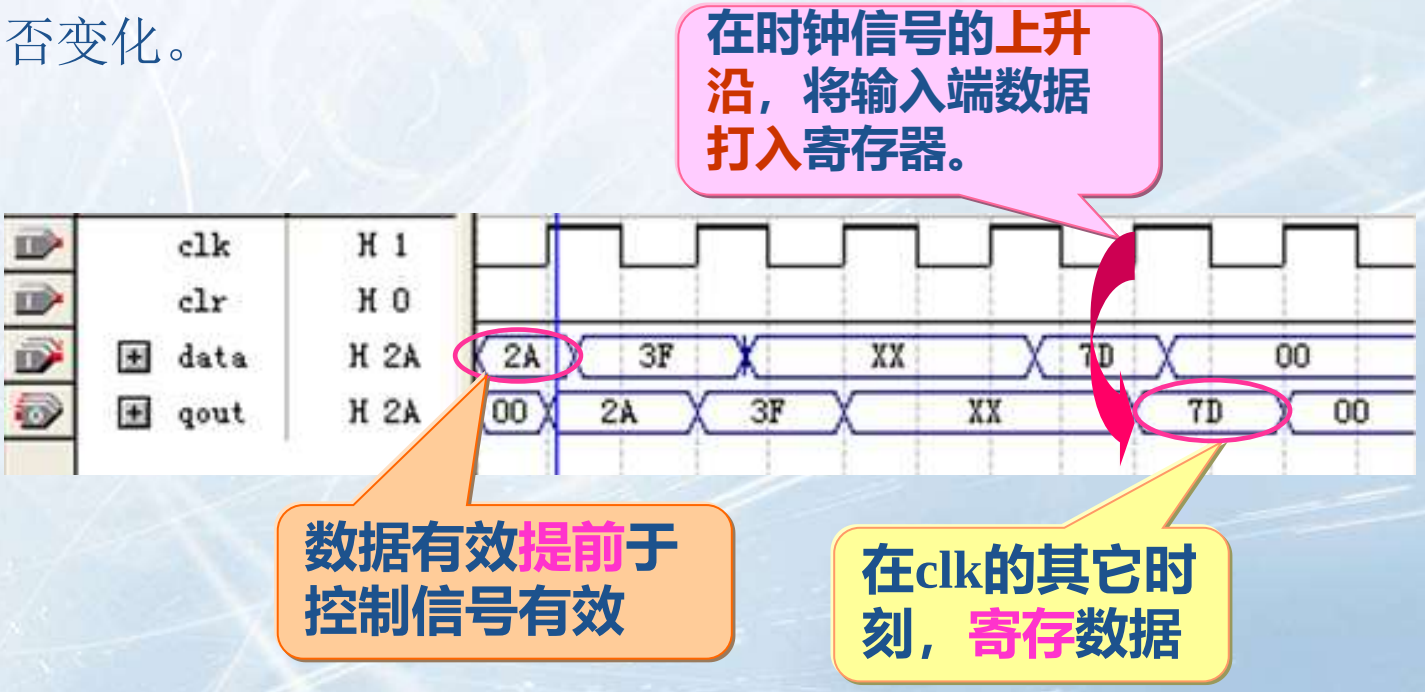


8位二进制数寄存器CT74273



数据寄存器

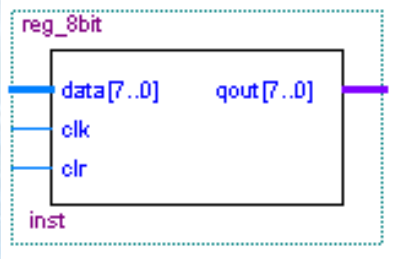
- 根据构成寄存器的记忆元件是D触发器（边沿触发）还是D锁存器（电位触发），将数码寄存器区分为数据寄存器和数据锁存器
- 数据寄存器——由多位边沿触发器组成的用于保存一组二进制代码的寄存单元。当时钟信号的上升沿或下降沿到来时，将输入端数据打入寄存器，即此时输出信号等于输入信号；在时钟信号的其它时刻，输出端保持刚才输入的数据，即为寄存状态，而不管此时输入信号是否变化。



8位数据寄存器的HDL设计

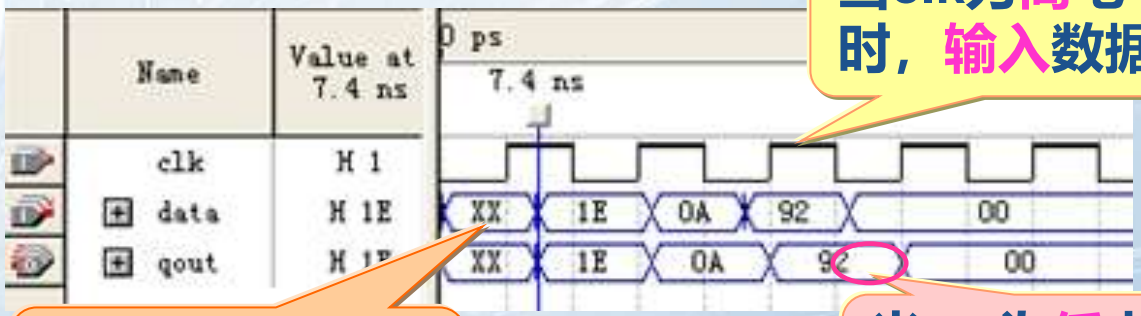
【例7.7】用always块语句描述的8位数据寄存器

```
module reg_8bit(qout,data,clk,clr);
      output[7:0] qout;
      input [7:0] data;
      input clk,clr;
      reg [7:0] qout;
      always @(posedge clk or posedge clr)    //沿触发
        begin
           if(clr) qout=0;                              //异步清零
           else     qout= data;
        end
endmodule
```



数据锁存器

- 数据锁存器：由多位电位触发器组成的用于保存一组二进制代码的寄存单元。
- 功能：当输入控制信号（如时钟）为高电平时，门是打开的，输出信号等于输入信号；当输入控制信号为低电平时，门是关闭的，输出端保持刚才输入的数据，即为锁存状态，而不管此时输入信号是否变化。
- 通常由电平信号来控制，属于电平敏感型，适于数据有效滞后于控制信号有效的场合



当clk为高电平时，输入数据

数据有效滞后于控制信号有效

当clk为低电平时，锁存数据

数据寄存器与数据锁存器的区别

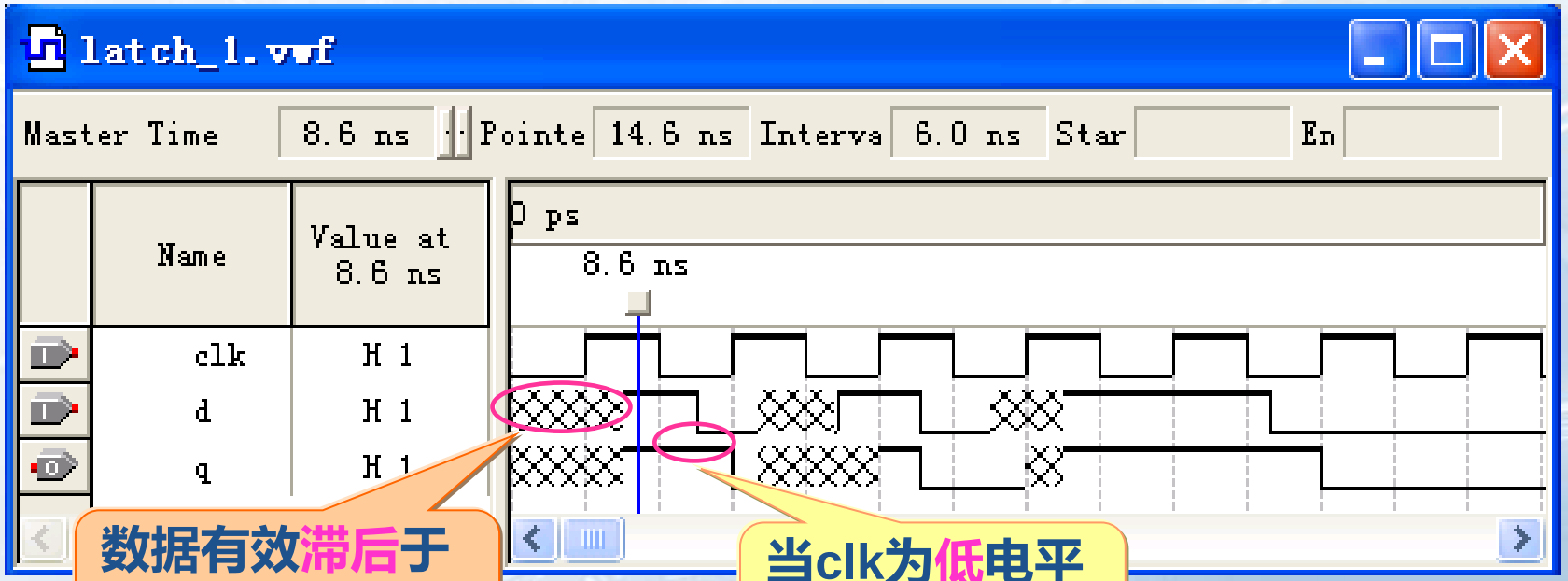
- 数据锁存器的实现方法：
  - 用**assign**语句——使用条件运算符，简便！
  - 用**always**块语句实现

- **数据寄存器和数据锁存器的区别：**
  - **数据寄存器由边沿触发的触发器组成。通常由同步时钟信号来控制，属于脉冲敏感型，适于数据有效提前于控制信号（一般为时钟信号）有效、并要求同步操作的场合。**
  - **数据锁存器由电位触发器（即D锁存器）组成。一般由电平信号来控制，属于电平敏感型，适于数据有效滞后于控制信号有效的场合。**

1位数据锁存器的HDL设计

【例7.8 】1位数据锁存器

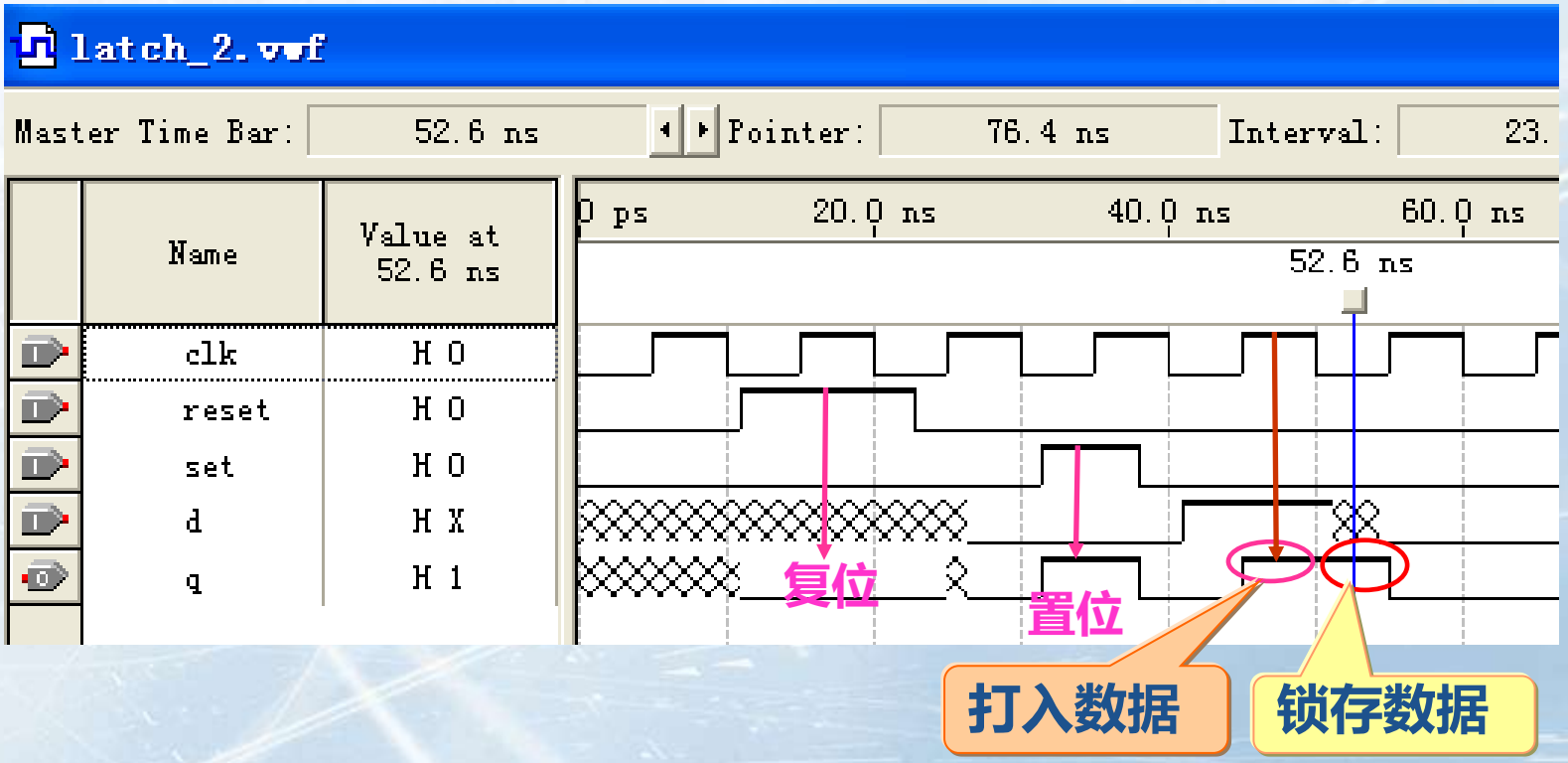
```
module latch_1(q,d,clk);
      output q;
      input d,clk ;
      assign q=clk?d:q;  /*当时钟信号为高电平时，将输入端信号打入锁存器；
                           当时钟信号为低电平时，锁存原来已打入的数据。*/
endmodule
```



带置位和复位端的1位数据锁存器的HDL设计

【例7.9 】带置位和复位端的1位数据锁存器

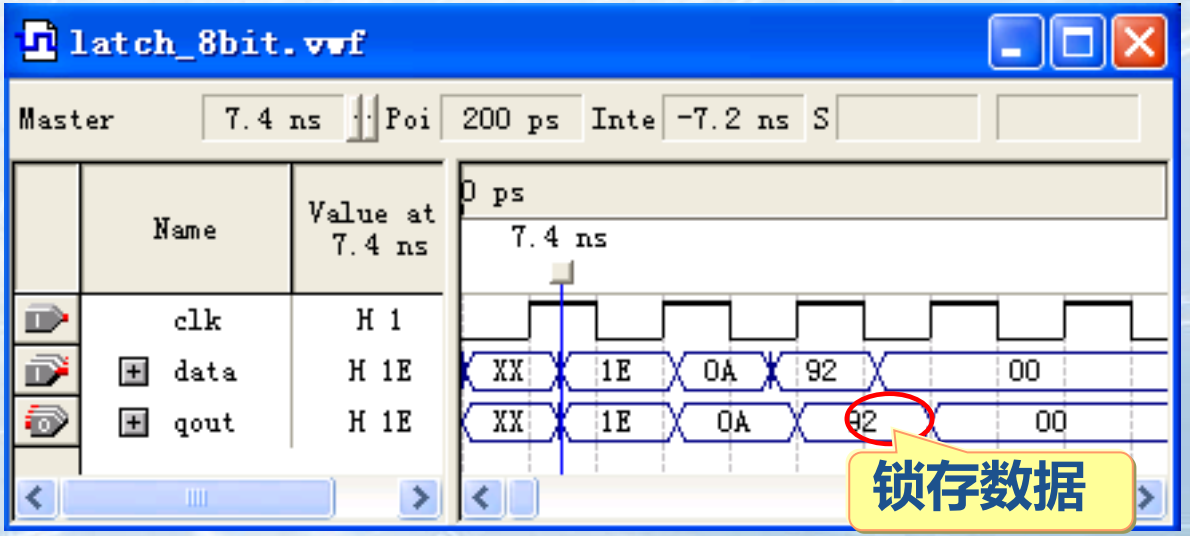
```
module latch_2(q,d,clk,set,reset);
    output q;
    input d,clk ,set,reset;
    assign q= reset ? 0: (set ? 1:(clk ? d:q));
endmodule
```



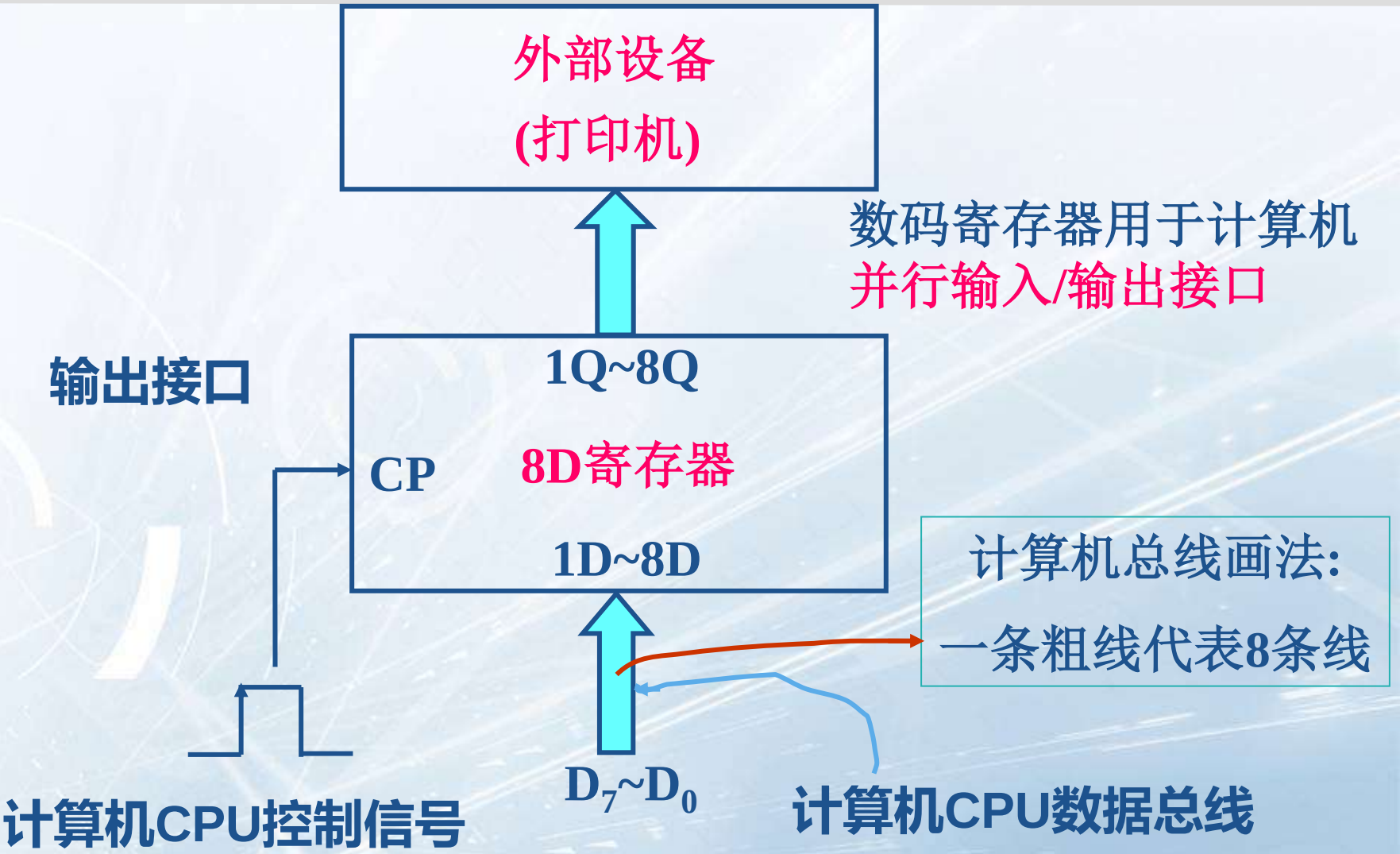
8位数据锁存器的HDL设计

【例7.10 】用always块语句描述的8位数据锁存器

```
module latch_8bit(qout,data,clk);
      output[7:0] qout;
      input [7:0] data;
      input clk;
      reg [7:0] qout;
      always @(clk or data)    //电平敏感
         if(clk) qout=data;
endmodule
```



数码寄存器的应用

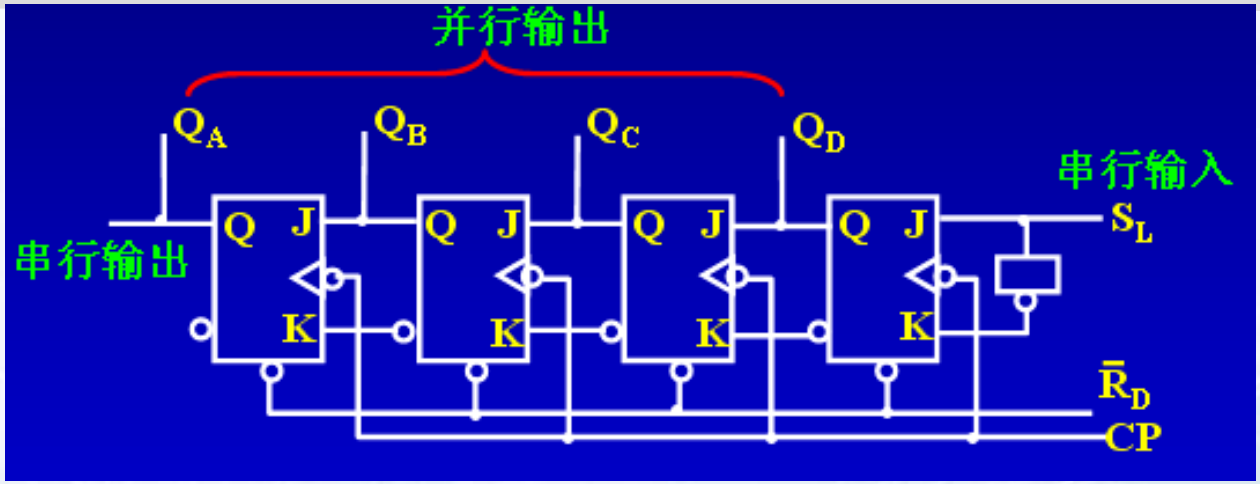

外部设备
(打印机)
数码寄存器用于计算机
并行输入/输出接口
输出接口
1Q~8Q
CP
8D寄存器
1D~8D
计算机总线画法:
一条粗线代表8条线
计算机CPU控制信号
D7~D0
计算机CPU数据总线

7.3.2 移位寄存器（移存器）

- 在计算机中，常要求寄存器有“移位”功能。例如，移位相加乘法器进行乘法运算时，要求将乘数右移，被乘数左移；除法运算时，要求将余数左移；将并行传递的数转换成串行数据以及将串行传递的数转换成并行数据的过程中，需要移位。
- 具有移位功能的寄存器称为移位寄存器，每来一个时钟脉冲，寄存器中数据就依次向左或向右移一位。
- 分类
  - 左移移位寄存器，右移移位寄存器，双向移位寄存器
- 数据输入方式
  - 串行输入，并行输入
- 数据输出方式
  - 串行输出：右移寄存器、左移寄存器
  - 并行输出：全部触发器的输出作为电路的输出
- 根据数据输入/输出方式，移位寄存器的工作方式有
  - 串入串出、串入并出、并入串出、并入并出

68

串入串出、串入并出移位寄存器



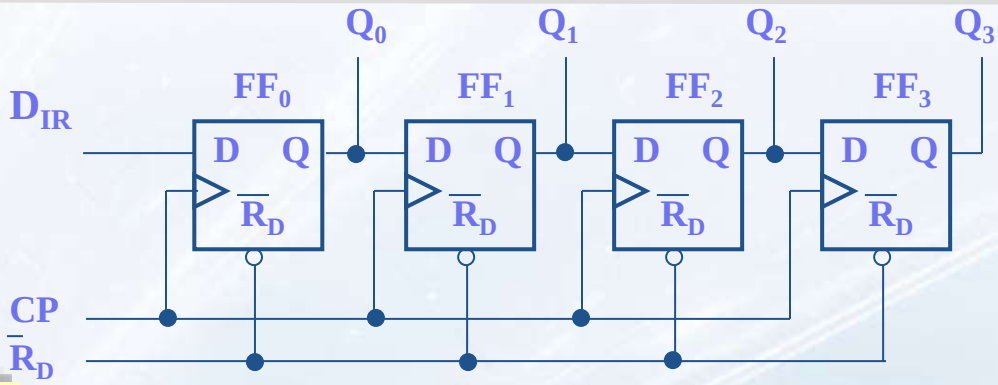
- 左移移位寄存器的串行输入方式：在同一个时钟的控制下，数据从寄存器的最右端串行输入，同时已存入的信息左移一位。

- **移位寄存器应采用边沿触发或主从触发方式的触发器，不能采用电位触发的触发器，以防止空翻。**

4位右移移位寄存器

- 电路结构（N=4右移）

  D_{IR}：右移串行数据输入1011

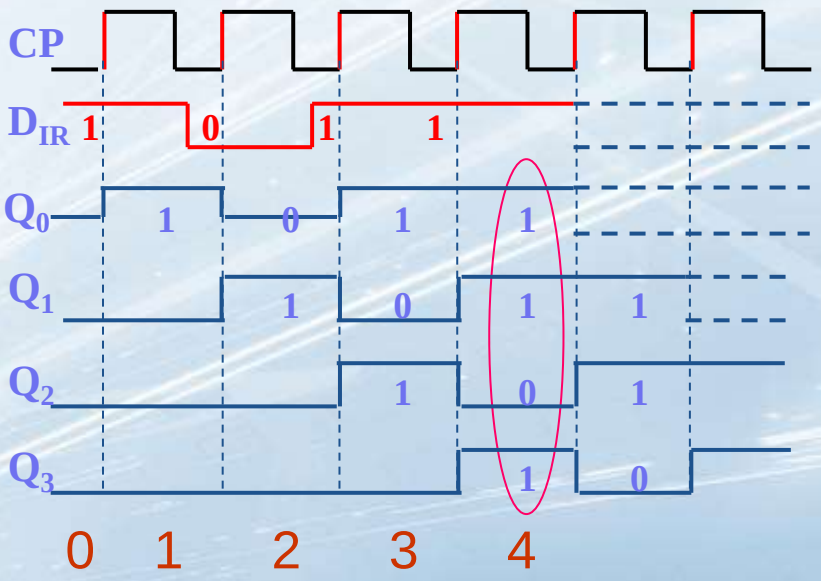


- 工作原理
  - 复位：

$$\overline{R_D}=0 \rightarrow Q_3Q_2Q_1Q_0=0000$$

  - 移位：

$$Q_0^{n+1}=D_0\mathrm{CP}\uparrow=D_{IR}\mathrm{CP}\uparrow$$
$$Q_1^{n+1}=D_1\mathrm{CP}\uparrow=Q_0^{n}\mathrm{CP}\uparrow$$
$$Q_2^{n+1}=D_2\mathrm{CP}\uparrow=Q_1^{n}\mathrm{CP}\uparrow$$
$$Q_3^{n+1}=D_3\mathrm{CP}\uparrow=Q_2^{n}\mathrm{CP}\uparrow$$



70

4位右移移位寄存器工作方式

- 工作方式
  - 串入并出

    串并转换（需要N个CP周期），经过4个CP，串行输入的4位数码全部移入移位寄存器中，并从$Q_3Q_2Q_1Q_0$并行输出1011
  - 串入串出

    把最右边的触发器的输出作为电路的输出。经过4个CP后， Q3 输出的是最先串行输入的数码。

    从每个触发器Q端输出的波形相同，但后级触发器Q端输出波形比前级触发器Q端输出波形滞后一个时钟周期。 把工作于串入串出方式的移位寄存器称为“延迟线”（第N级FF延迟N个CP周期）

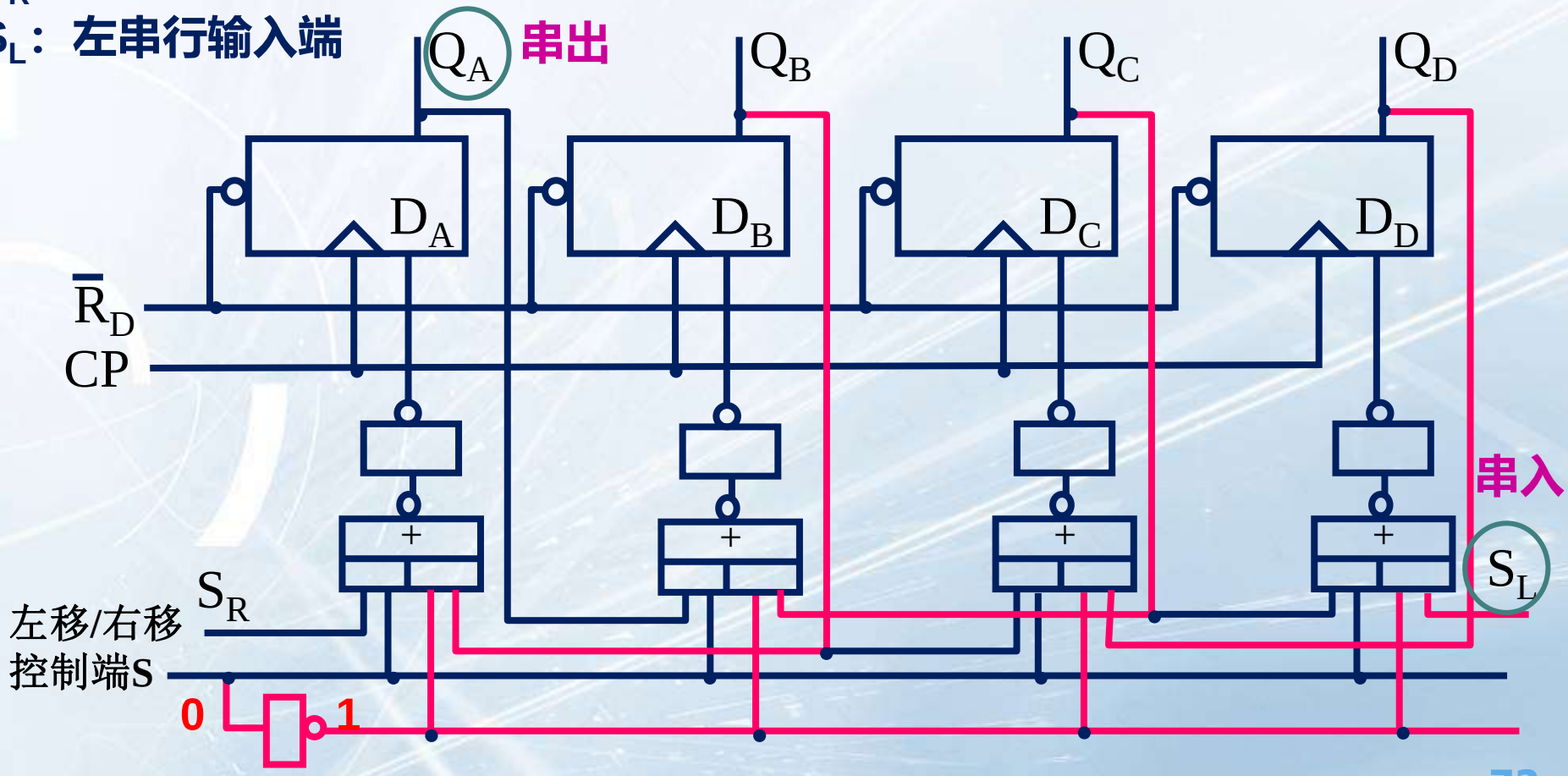
71

4位串行输入、串/并行输出双向移位寄存器

S=0，（往）左移
S=1，（往）右移
S_R：右串行输入端
S_L：左串行输入端

$D_A=S_R\,S+Q_B\,\bar{S}$　　$D_B=Q_A\,S+Q_C\,\bar{S}$

$D_C=Q_B\,S+Q_D\,\bar{S}$　　$D_D=Q_C\,S+S_L\,\bar{S}$



4位串行输入、串/并行输出双向移位寄存器工作原理

- 复位 $\overline{R_D}=0\rightarrow Q_AQ_BQ_CQ_D=0000$

- 移位

（1）当S=0 时：左移移位寄存器

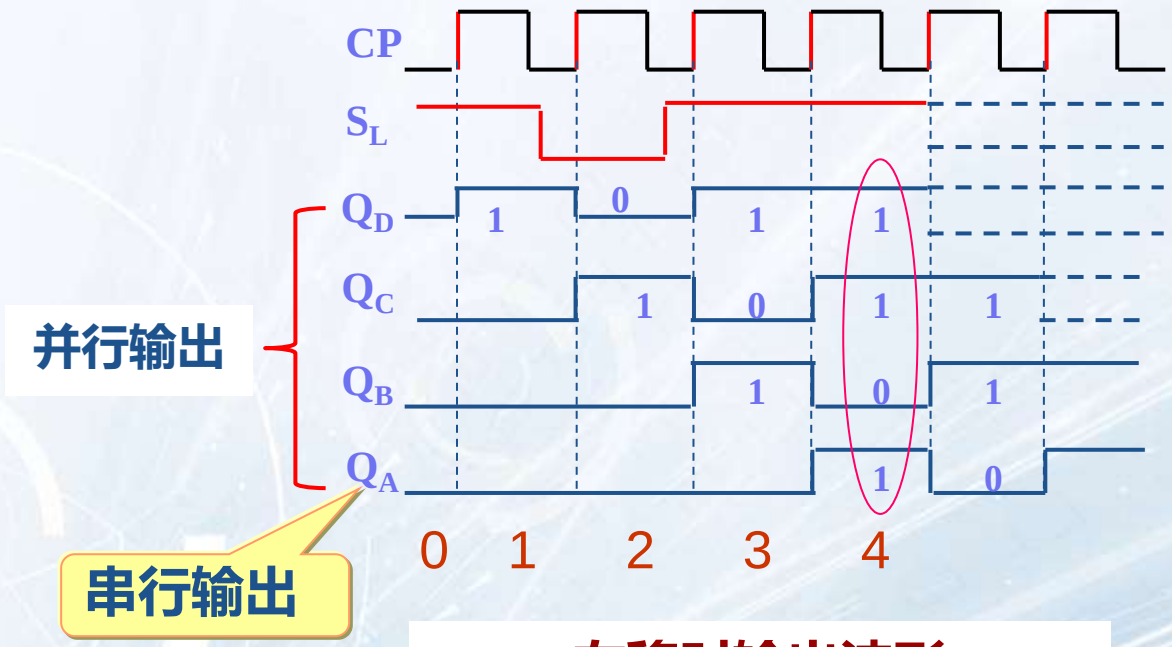
串入并出——数据从S_L端串行输入，顺序左移，$D_A=Q_B$， $D_B=Q_C$，$D_C=Q_D$， $D_D=S_L$。经过4个CP，串行输入的4位数码全部移入移位寄存器中，并从$Q_AQ_BQ_CQ_D$并行输出。

串入串出——左移（右移）移位寄存器把最左边（右边）的触发器的输出作为电路的输出。从每个触发器Q端输出的波形相同，但后级触发器Q端输出波形比前级触发器Q端输出波形滞后一个时钟周期。左移时，经过4个CP，最先串行输入的1位数码从Q_A输出，下一个CP上升沿到来时，Q_A输出串行移入的第2个数码……。

（2）S＝1时：右移移位寄存器

数据从S_R端串行输入，顺序右移，$D_A=S_R$， $D_B=Q_A$， $D_C=Q_B$， $D_D=Q_C$。

4位串行输入、串/并行输出双向移位寄存器输出波形

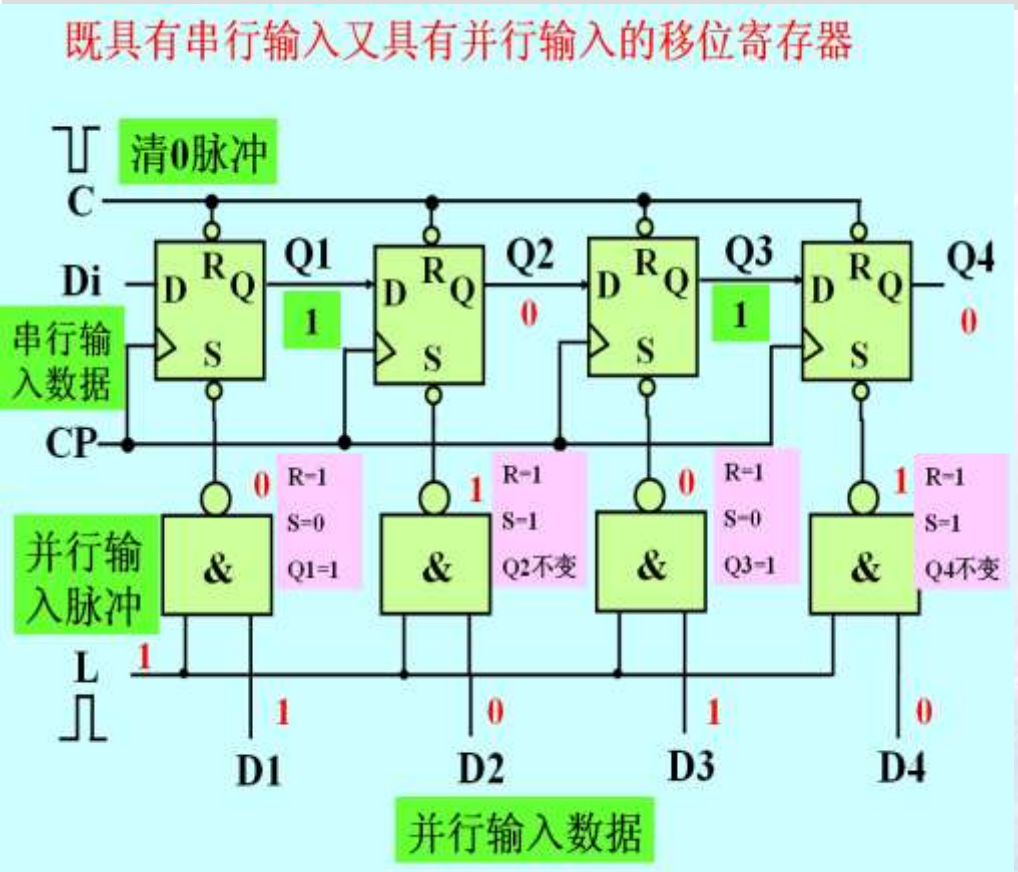


左移时输出波形

7.3.3 集成移位寄存器

- 按输入、输出方式，集成移位寄存器分为5类
  - 串入—并出单向移位寄存器
  - 串入—串出单向移位寄存器
  - 串入、并入—串出单向移位寄存器
  - 串入、并入—并出单向移位寄存器
  - 串入、并入—并出双向移位寄存器

串入、并入—串出单向移位寄存器



- 带异步置0和置1的D FF构成串入、并入—串出单向移位寄存器
- 并入—串出工作原理
  - 当C=0时DFF清0，$Q_1Q_2Q_3Q_4$=0000；
  - 当L为正脉冲、$D_1D_2D_3D_4$=1010时，并行输入数据， 4个与非门的输出从左至右依次为0 1 0 1； $Q_1Q_2Q_3Q_4$=1010；经过4个CP后，并行输入的4位数据从Q_4 串行移出（0101），最后输出的数据是D_1。

4位双向移位寄存器CT74194

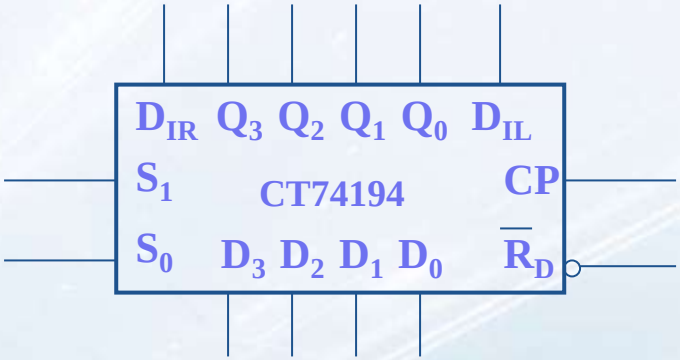
S_1S_0：功能控制输入端

$D_3D_2D_1D_0$：并行数据输入端

$Q_3Q_2Q_1Q_0$：数据输出

D_{IR}：右移串行输入（Q_3为串行输出端）

D_{IL}：左移串行输入（Q_0为串行输出端）



功能表

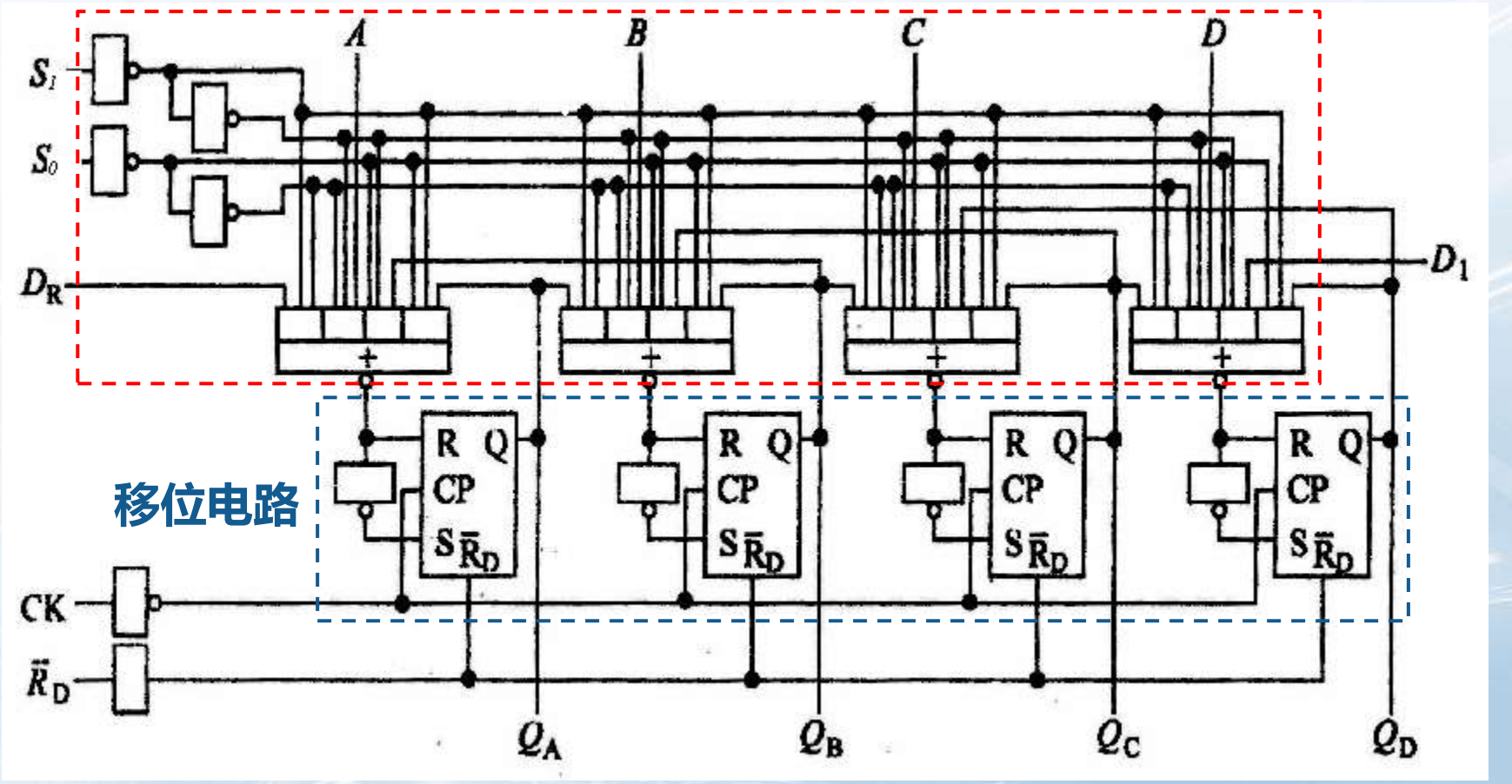
$\overline{R}_D$	CP	S_1	S_0	功能
0	X	X	X	置零
1	↑	0	0	保持
1	↑	0	1	右移
1	↑	1	0	左移
1	↑	1	1	并行输入

- 工作方式
  - 串入并出——串并转换
  - 串入串出——延迟线
  - 并入串出——并串转换
  - 并入并出——数据预置

77

CT74194电路结构图

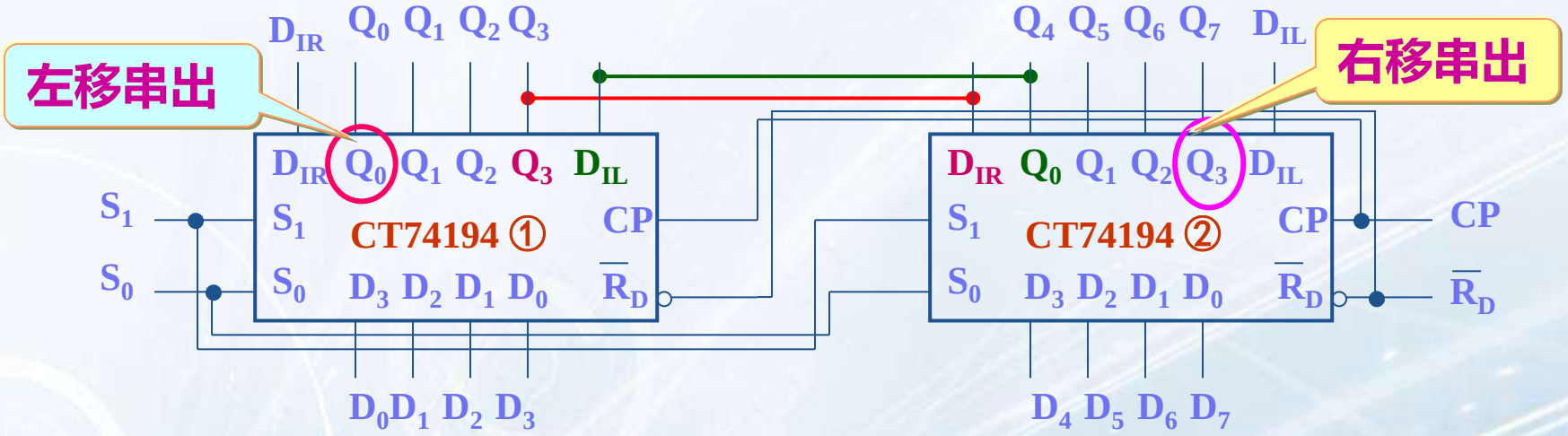
S_1S_0=00，保持；S_1S_0=01，右移；
S_1S_0=10，左移；S_1S_0=11，并入



78

集成移位寄存器扩展方法

❖ 用2片4位移位寄存器扩展为8位移位寄存器



- 将片①的Q_3接至片②的D_{IR}，当S1S0=01时，右移，片②的输出Q_3作为整个电路的右移串行输出端（Q_7）。
- 将片②的Q_0接至片① 的D_{IL}，当S1S0=10时，左移，片①的输出Q_0 作为整个电路的左移串行输出端Q_0。
- 同时把两片的S_1、S_0、CP和/R_D分别并联。

集成移位寄存器的用途

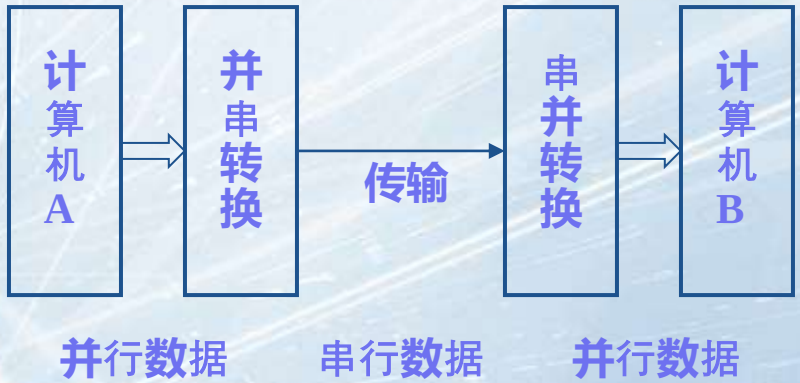
❖主要用途

- ◆ 数据保存与移位
- ◆ 计算机串行通信（使用串行接口RS232、RS485）中的并串转换及串并转换

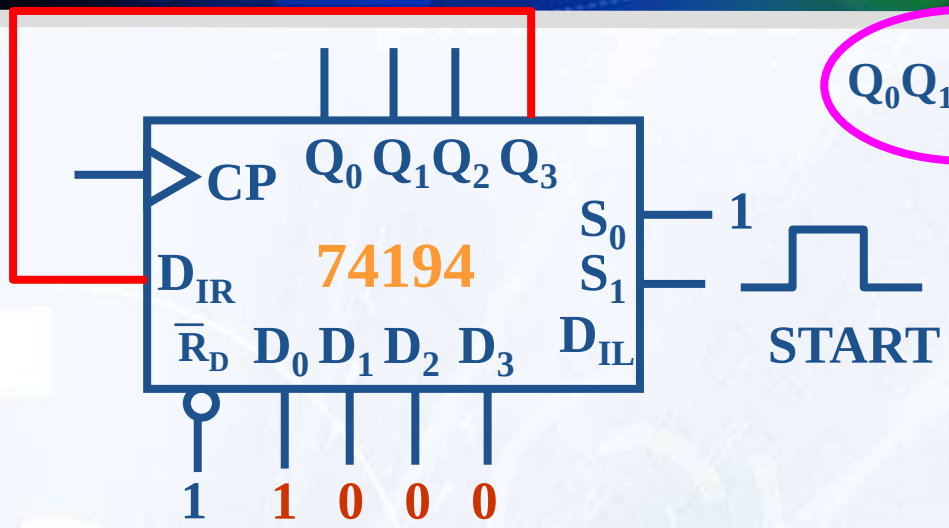
  串行通信——数据在一根传输线上一位一位地顺序传送。

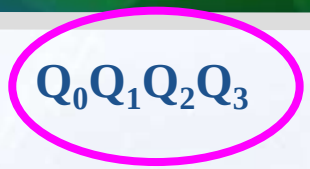
  并行通信——数据以字节（字）为单位在多根传输线上同时传送。
- ◆ 移存型计数器（环形计数器，扭环形计数器）

  利用移位寄存器组成的计数器叫做移存型计数器 。



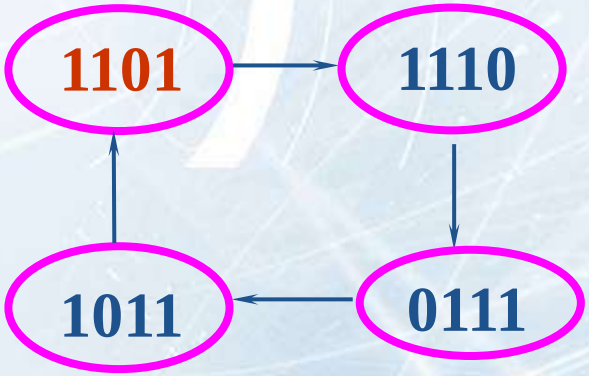
用移位寄存器实现环形计数器

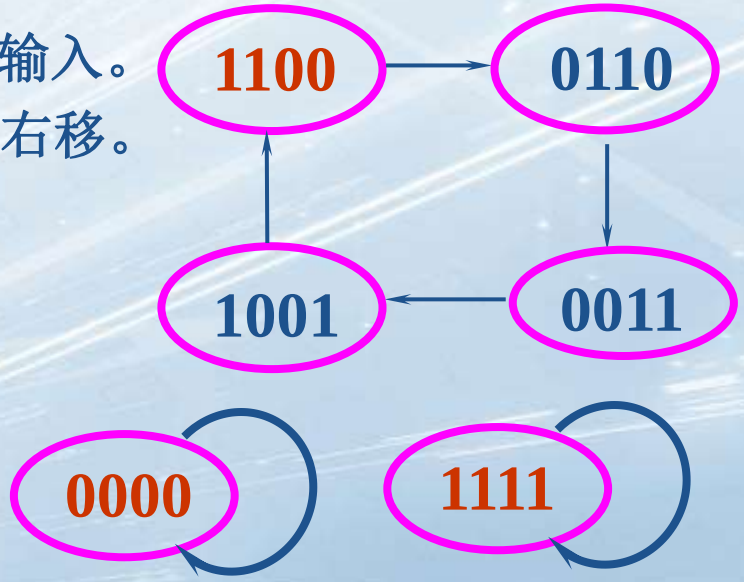




- 当$D_0D_1D_2D_3$为不同的初值，环形计数器的计数规律不同

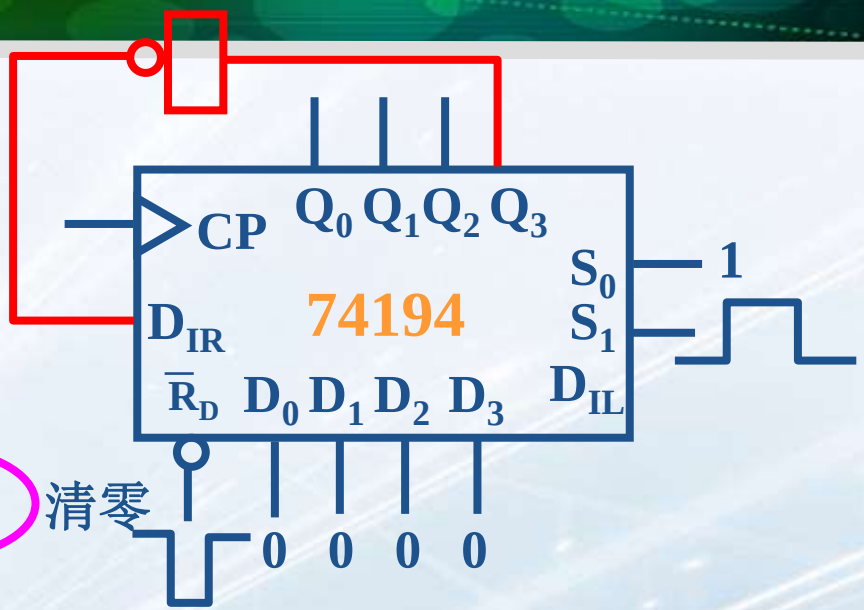
- 当START信号脉冲有效时， S_1S_0=11，并行输入。
- 当START信号脉冲无效时， S_1S_0=01，执行右移。

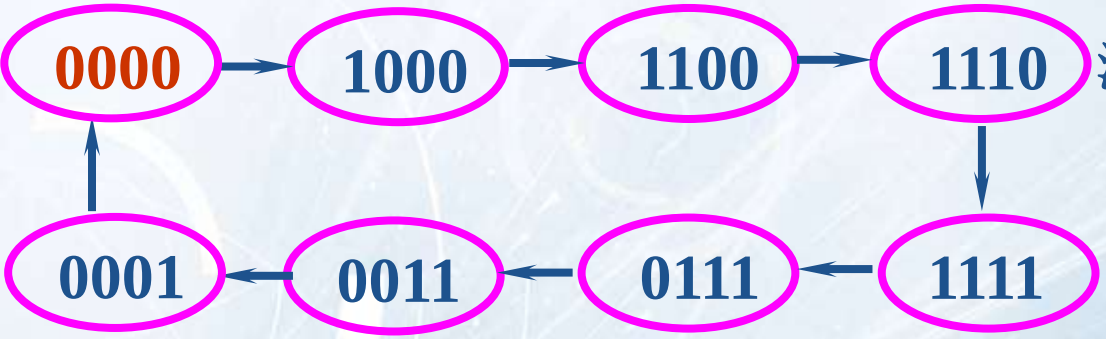




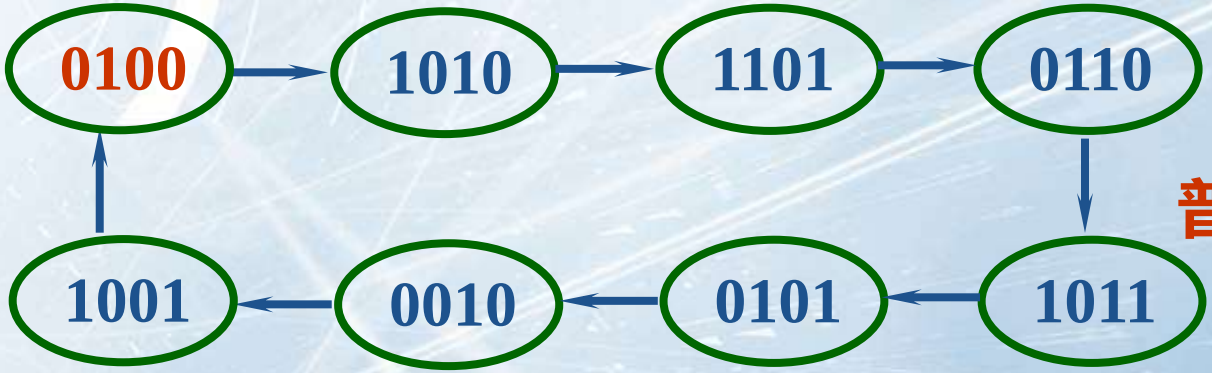
用移位寄存器实现扭环形计数器

- Q_3取反后接至右串行输入D_{IR}端。
- 当 START 信号脉冲有效时，S_1S_0=11，并行输入。
- 当 START 信号脉冲无效时，S_1S_0=01，执行右移。



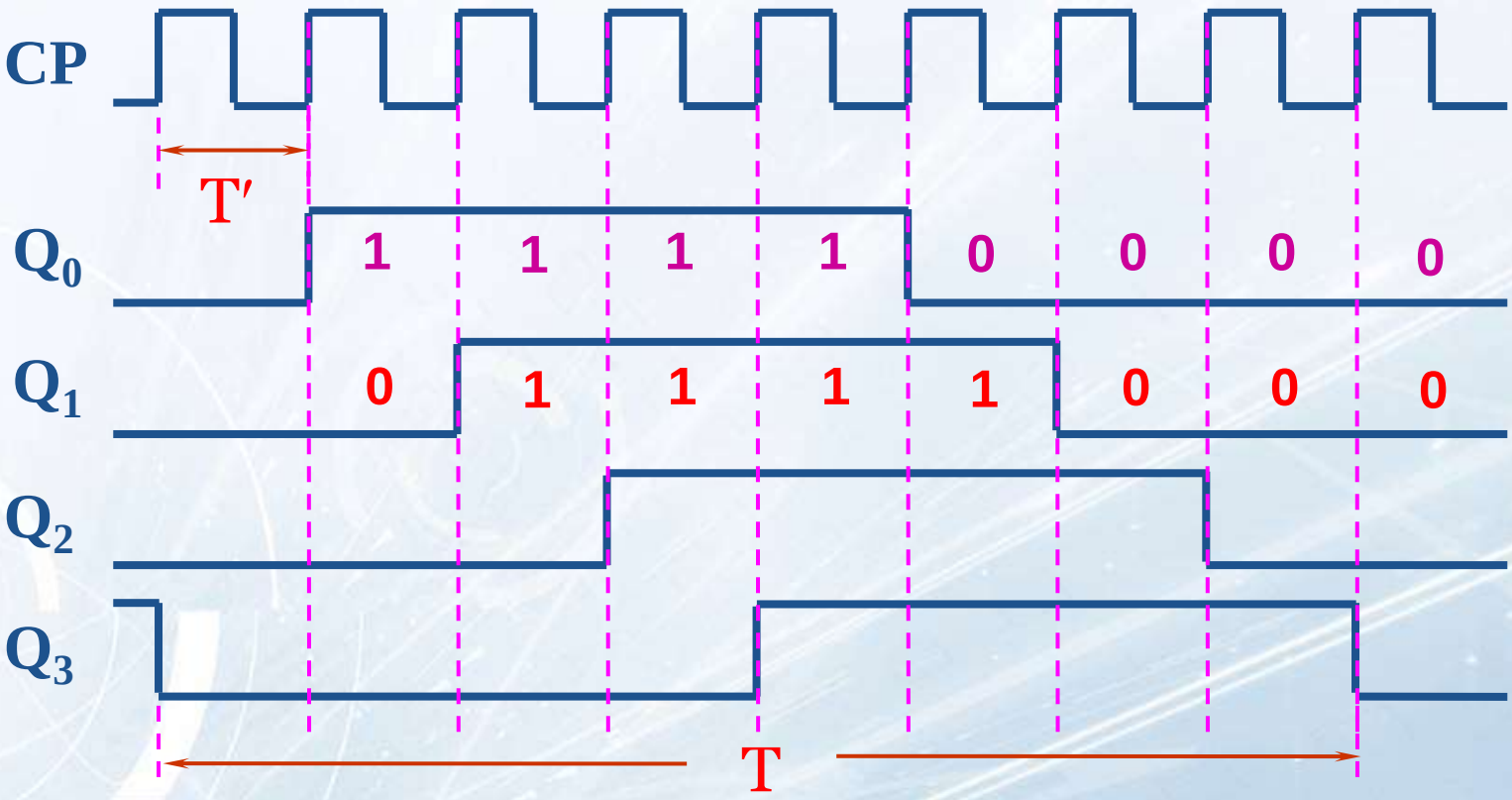


4位格雷码计数器（模8）



普通八进制计数器

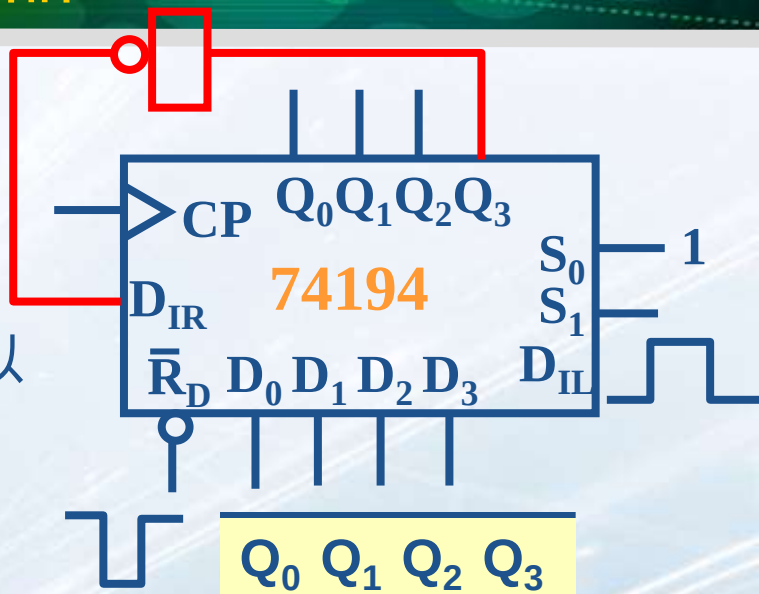
4位格雷码计数器输出波形



因为输出信号的频率为CP的1/8，所以4位格雷码计数器相当于÷8除法器

4位格雷码计数器也是序列信号发生器

- 电路工作先清零，$Q_0Q_1Q_2Q_3$=0000，使电路进入序列顺序中。
  - 在第1个CP作用下，D_{IR}=/Q_3=1进入Q_0，以后“1”不断移入Q_0；在第4个CP来到后，D_{IR}=/Q_3=0，以后“0”不断移入Q_0。
  - 得到Q_0序列为11110000、11110000、…。
  - Q_1为Q_0的右移1位关系，Q_1序列为01111000、01111000、…。
  - Q_2序列为00111100、00111100、…。
  - Q_3序列为00011110、00011110、…。



Q_0	Q_1	Q_2	Q_3
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1

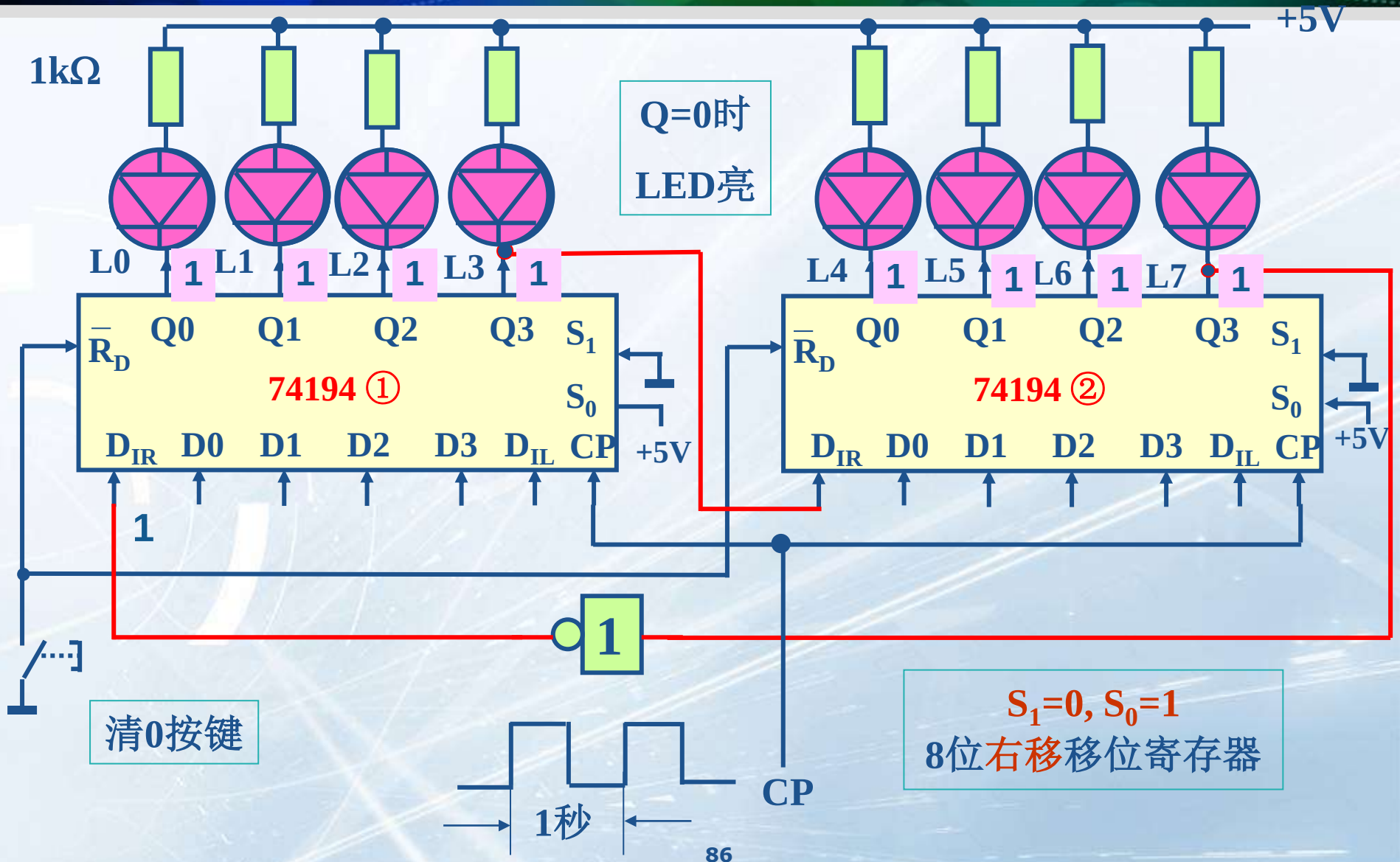
返回同步计数器

4位移位寄存器的应用——节日彩灯控制电路

- 【例7.11】利用4位双向移位寄存器CT74194设计一个节日彩灯控制电路，使其控制8个发光二极管，当按下清0键时，8个LED都亮；然后从第1个LED开始，每来一个CP，各LED依次熄灭；当第8个LED熄灭后，从第1个LED开始，每来一个CP， 各LED又依次点亮；重复此规律。

- 分析：因为要控制8个发光二极管，而CT74194为4位双向移位寄存器，所以需2片CT74194。
- 清0 信号（假定低有效）接CT74194的异步置零端/R_D，可使寄存器清零（Q0~Q7=0000 0000）；由于要求按下清0键时，8个LED都亮，则CT74194的的各输出端应接LED 的阴极。
- 利用CT74194的右移功能，Q3为串行输出端；将片①的Q_3接至片②的D_{IR}，构成8位右移移位寄存器；要使LED 熄灭，则串行输入数据应为“1”，故将片②的Q_3反相后接至片① 的D_{IR} 。

节日彩灯控制电路连接图



节日彩灯控制电路工作原理

- 把两片4位移位寄存器的CP和/R_D分别并联； S_1S_0=01，移位寄存器执行右移功能。
- 首先按清0键，移位寄存器清零，①和②的输出$Q_0Q_1Q_2Q_3$均为0，则8个LED都亮。
- 由于片①的Q_3接至片②的D_{IR}，则构成8位右移移位寄存器。
- 由于将片②的Q_3反相后接至片① 的D_{IR}，则在第1个CP作用下，片②的/Q_3=1进入片①的Q_0，使所驱动的L0熄灭，以后“1”不断移入Q_0，则其后的LED依次熄灭；
- 在第8个CP来到后，片①的D_{IR}=片②的/Q_3=0，以后“0”不断移入片①的Q_0。则从L0开始，每来一个CP， LED依次点亮。

① Q_0	Q_1	Q_2	Q_3	② Q_0	Q_1	Q_2	Q_3
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	0	1	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	1

7.4 计数器

内容概要

7.4.1 同步计数器

7.4.2 异步计数器

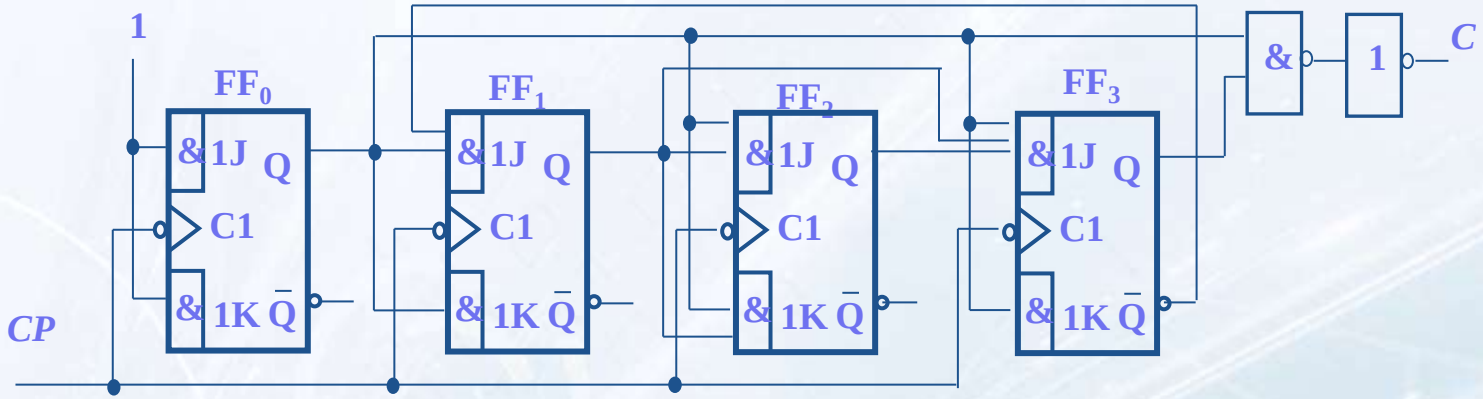
7.4.3 集成计数器

7.4.1 同步计数器

- 计数器是可以统计输入脉冲个数的器件
- 计数器的用途
  - （脉冲）计数
  - 计时——通过不同模值的计数器，计秒、分钟、小时等
  - 定时（定时器）
  - 分频——将高频时钟信号分频为较低频率的信号
  - 产生节拍脉冲（顺序脉冲）和序列脉冲　　序列信号发生器
- 计数器的分类
  - 时钟方式：根据计数器中触发器时钟端的连接方式，分为同步计数器，异步计数器
  - 计数方式：二进制计数器，十进制计数器，M进制计数器
  - 状态变化：根据计数器中的状态变化规律分为加法计数器，减法计数器，加/减法计数器

同步计数器的分析方法

【例7.12】分析下图电路，说明电路的特点。



解：(1)写方程式

$$J_0 = K_0 = 1;$$

$$J_1 = \overline{Q}_3^n Q_0^n, K_1 = Q_0^n;$$

$$J_2 = K_2 = Q_1^n Q_0^n;$$

$$J_3 = Q_2^n Q_1^n Q_0^n, K_3 = Q_0^n$$

$$C = \overline{\overline{Q_3^n Q_0^n}} = Q_3^n Q_0^n$$

$$Q_0^{n+1} = J_0 \overline{Q}_0^n + \overline{K}_0 Q_0^n = \overline{Q}_0^n$$

$$Q_1^{n+1} = J_1 \overline{Q}_1^n + \overline{K}_1 Q_1^n = \overline{Q}_3^n Q_0^n \overline{Q}_1^n + \overline{Q}_0^n Q_1^n$$

$$Q_2^{n+1} = J_2 \overline{Q}_2^n + \overline{K}_2 Q_2^n = Q_1^n Q_0^n \overline{Q}_2^n + \overline{Q_1^n Q_0^n} Q_2^n$$

$$Q_3^{n+1} = J_3 \overline{Q}_3^n + \overline{K}_3 Q_3^n = Q_2^n Q_1^n Q_0^n \overline{Q}_3^n + \overline{Q}_0^n Q_3^n$$

$CP_0 = CP_1 = CP_2 = CP_3 = CP \downarrow$ ——同步电路

90

状态转换表

（2）计算并列出状态转换表

$Q_3^nQ_2^nQ_1^nQ_0^n$	$Q_3^{n+1}Q_2^{n+1}Q_1^{n+1}Q_0^{n+1}$	C
0 0 0 0	0 0 0 1	0
0 0 0 1	0 0 1 0	0
0 0 1 0	0 0 1 1	0
0 0 1 1	0 1 0 0	0
0 1 0 0	0 1 0 1	0
0 1 0 1	0 1 1 0	0
0 1 1 0	0 1 1 1	0
0 1 1 1	1 0 0 0	0
1 0 0 0	1 0 0 1	0
1 0 0 1	0 0 0 0	1
1 0 1 0	1 0 1 1	0
1 0 1 1	0 1 0 0	1
1 1 0 0	1 1 0 1	0
1 1 0 1	0 1 0 0	1
1 1 1 0	1 1 1 1	0
1 1 1 1	0 0 0 0	1

$$Q_0^{n+1}=\overline{Q}_0^n$$

$$Q_1^{n+1}=\overline{Q}_3^nQ_0^n\overline{Q}_1^n+\overline{Q}_0^nQ_1^n$$

$$Q_2^{n+1}=Q_1^nQ_0^n\overline{Q}_2^n+\overline{Q_1^nQ_0^n}Q_2^n$$

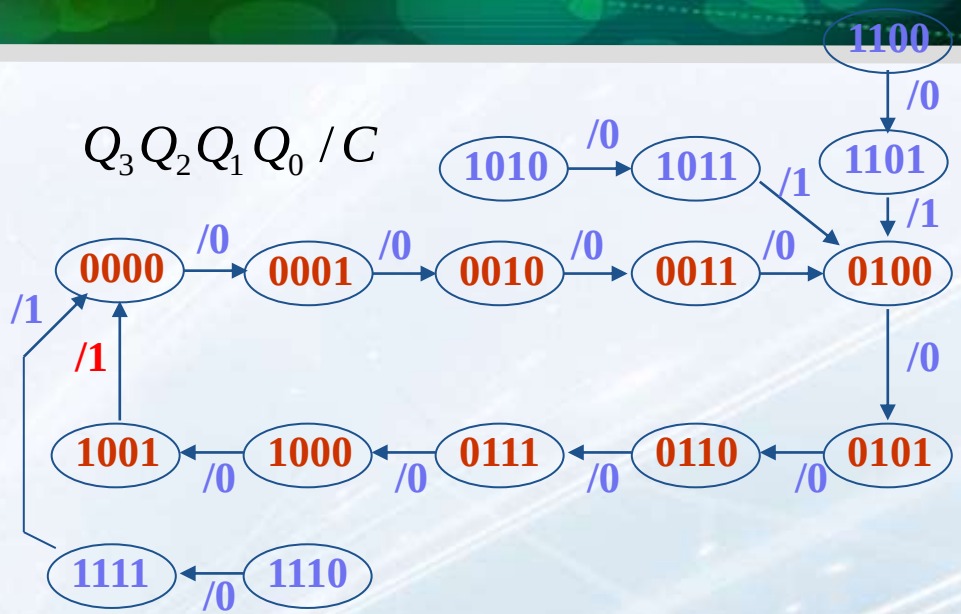
$$Q_3^{n+1}=Q_2^nQ_1^nQ_0^n\overline{Q}_3^n+\overline{Q}_0^nQ_3^n$$

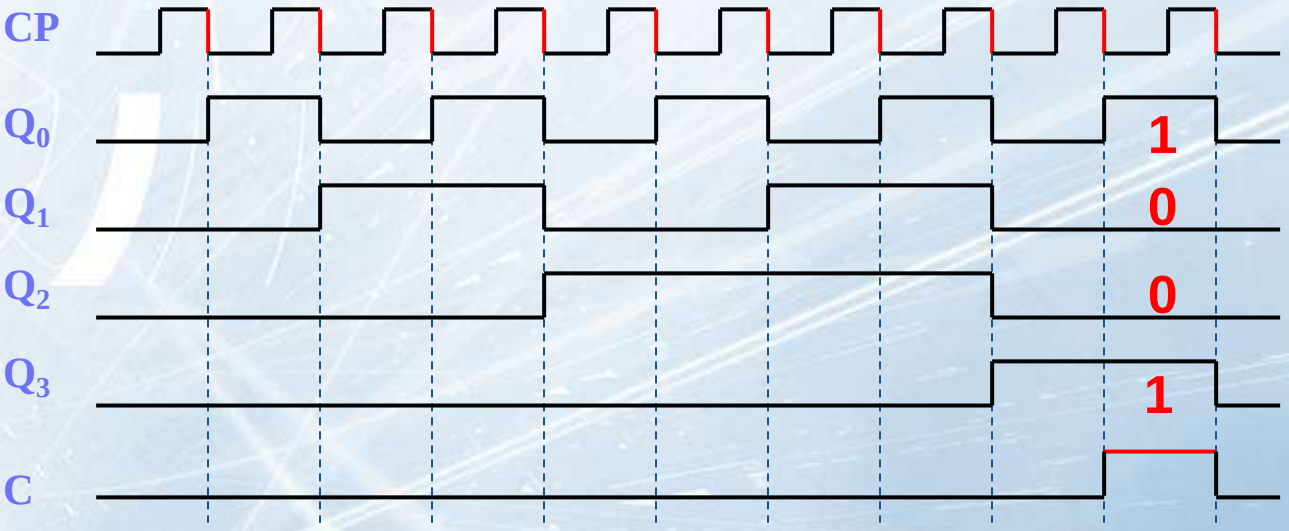
$$C=Q_3^nQ_0^n$$

状态转换图和时序图

（3）画状态转换图（或时序图）

- 画状态转换图时一定要画出全部状态的变化。
- 画时序图时只画出有效状态构成的计数循环的变化；注意触发器的时钟特性！





CP下降沿时触发器翻转！

电路功能

（4）说明电路功能

根据状态转换图，可知为同步十进制加法计数器，有自启动能力

- 计数器——由若干状态构成一个计数循环
- 同步——构成电路的全部FF的时钟端连接在一起
- 十进制——计数循环的状态个数为10（模10计数器）
- 加法——计数状态按递增方向变化
- 自启动——不存在死循环，计数循环以外的状态，都能回到计数循环中来

设计计数器时，不允许存在死循环！

- 死循环（无效循环）——由无效状态构成的循环

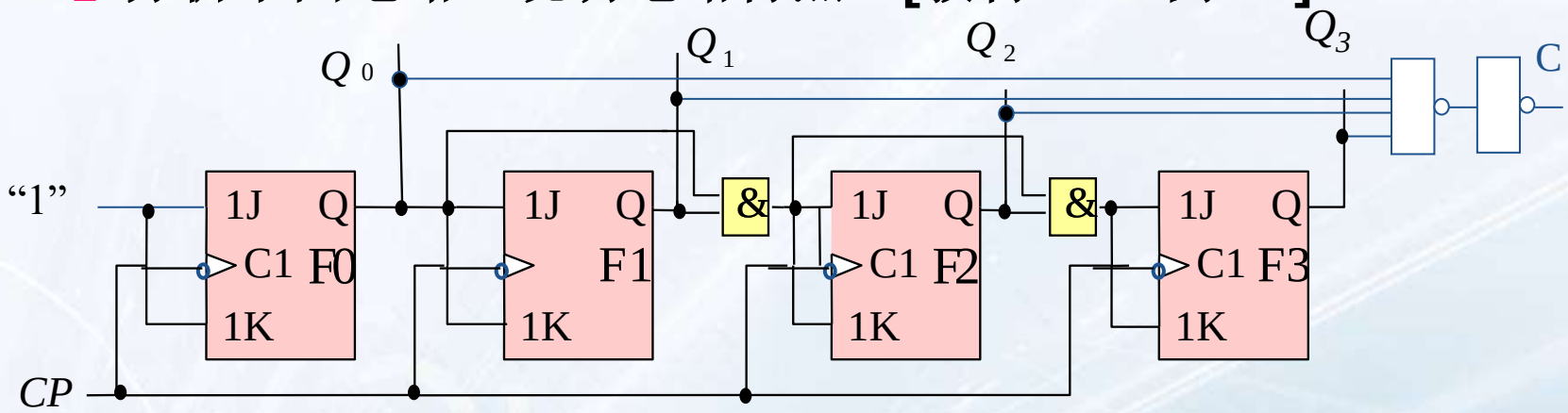
同步计数器的特点

- 所有触发器的时钟端并联在一起，作为计数器的时钟端
- 各触发器同时翻转，不存在时钟到各触发器输出的传输延迟的积累
- 由于其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。
- 由于计数器各触发器几乎是同时翻转的，因此，各触发器输出波形的偏移为各触发器时钟到输出的延迟之差，同步计数器输出经译码后所产生的尖峰信号宽度比较小。
- 缺点：结构比较复杂（各触发器的输入由多个Q输出相与得到），所用元件较多。

94

【例7.13】同步计数器的分析

【例7.13】分析下图电路，说明电路特点。[教材P155例6.3]



（1）写出驱动方程，状态方程和输出方程

$J_0 = K_0 = 1$

$J_1 = K_1 = Q_0^n$

$J_2 = K_2 = Q_1^n Q_0^n$

$J_3 = K_3 = Q_2^n Q_1^n Q_0^n$

$$Q_0^{n+1} = \overline{Q}_0^n$$

$$Q_1^{n+1} = Q_0^n \overline{Q}_1^n + \overline{Q}_0^n Q_1^n = Q_0^n \oplus Q_1^n$$

$$Q_2^{n+1} = Q_1^n Q_0^n \overline{Q}_2^n + \overline{Q_1^n Q_0^n} Q_2^n$$

$$Q_3^{n+1} = Q_2^n Q_1^n Q_0^n \overline{Q}_3^n + \overline{Q_2^n Q_1^n Q_0^n} Q_3^n$$

$$C = \overline{\overline{Q_3^n Q_2^n Q_1^n Q_0^n}} = Q_3^n Q_2^n Q_1^n Q_0^n$$

状态转换表

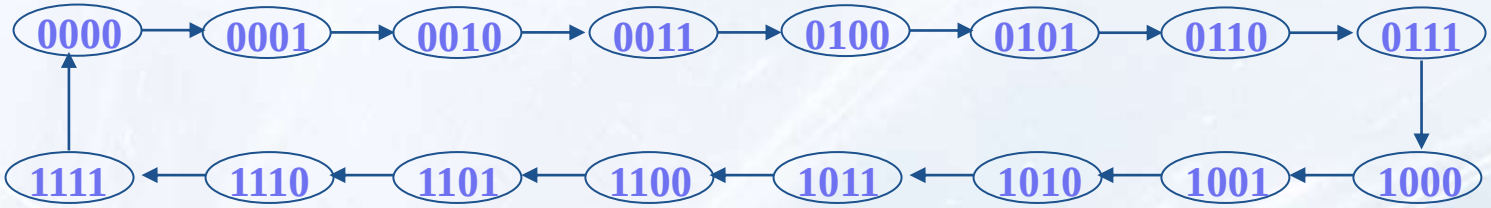
（2）列出状态转换表

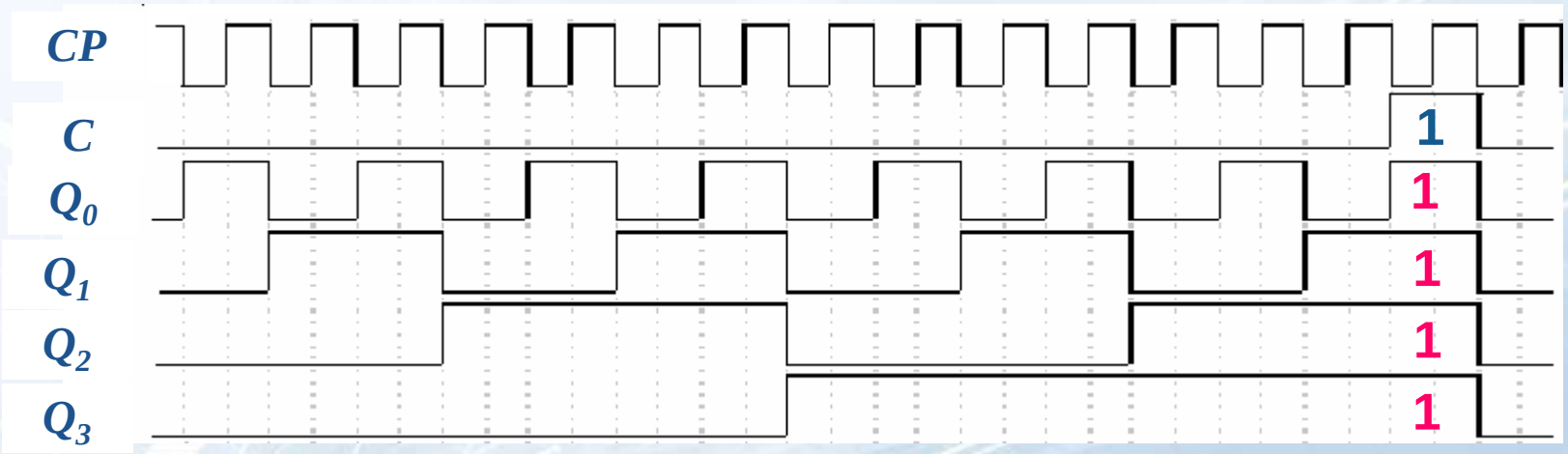
Q_3^n Q_2^n Q_1^n Q_0^n	Q_3^{n+1} Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}	C
0 0 0 0	0 0 0 1	0
0 0 0 1	0 0 1 0	0
0 0 1 0	0 0 1 1	0
0 0 1 1	0 1 0 0	0
0 1 0 0	0 1 0 1	0
0 1 0 1	0 1 1 0	0
0 1 1 0	0 1 1 1	0
0 1 1 1	1 0 0 0	0
1 0 0 0	1 0 0 1	0
1 0 0 1	1 0 1 0	0
1 0 1 0	1 0 1 1	0
1 0 1 1	1 1 0 0	0
1 1 0 0	1 1 0 1	0
1 1 0 1	1 1 1 0	0
1 1 1 0	1 1 1 1	0
1 1 1 1	0 0 0 0	1

对应原态

状态转换图和时序图

（3）根据状态转换表，画出状态转换图和时序图





（4）说明电路的功能

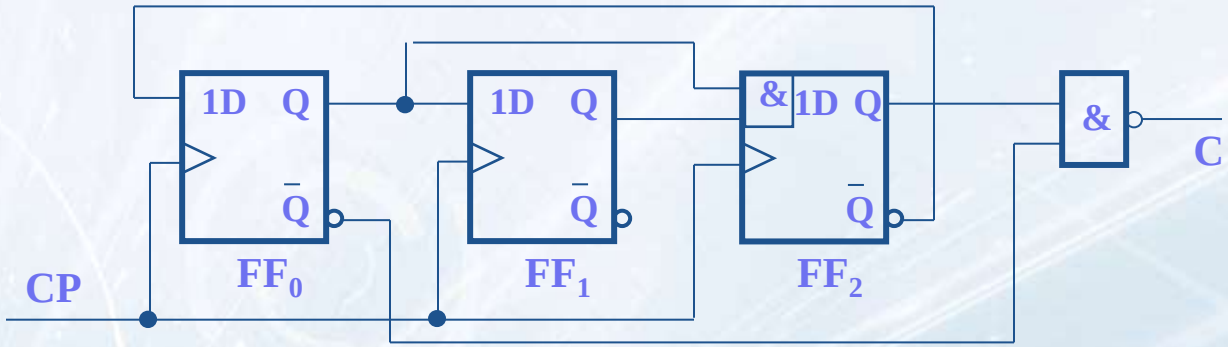
电路是一个同步二进制（模16）的加法计数器

二进制加法计数器的特点

- **二进制加法计数器的特点**
  - 计数器中触发器的状态按照二进制数的规律变化，构成计数循环的状态个数（**模值**）为2^n（n是触发器的级数）。
  - 二进制计数器没有非编码状态，所以不存在不能自启动的问题。
  - Q_0、Q_1、Q_2、Q_3的周期分别是计数脉冲（CP）周期的2倍、4倍、8倍、16倍，也就是说Q_0、Q_1、Q_2、Q_3分别对CP波形进行了2分频、4分频、8分频、16分频，因此二进制计数器也称为**分频器**。

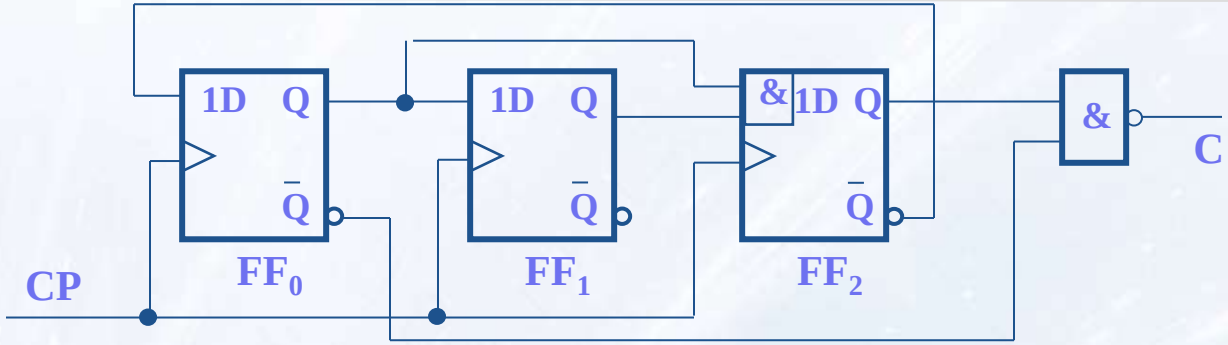
同步计数器练习题

练习题：分析下图电路，说明电路特点。



99

同步计数器练习详解（1/2）

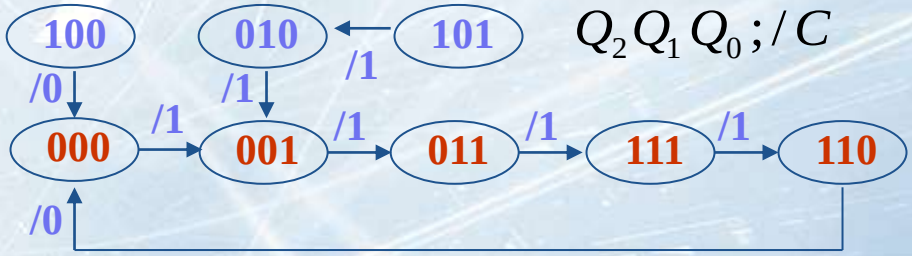


$$D_0 = \overline{Q_2^n};\ Q_0^{n+1} = D_0 = \overline{Q_2^n}$$

$$D_1 = Q_0^n;\ Q_1^{n+1} = D_1 = Q_0^n$$

$$D_2 = Q_1^n \cdot Q_0^n;\ Q_2^{n+1} = D_2 = Q_1^n \cdot Q_0^n$$

$$C = \overline{Q_2^n \cdot \overline{Q_0^n}}$$

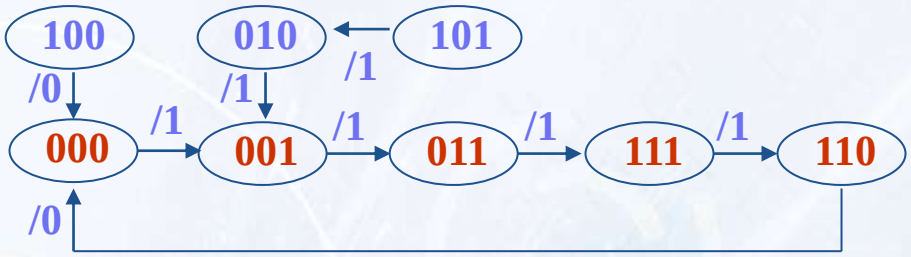


$Q_2^nQ_1^nQ_0^n$	$Q_2^{n+1}Q_1^{n+1}Q_0^{n+1}$	C	
0 0 0	0 0 1	1	1
0 0 1	0 1 1	1	2
0 1 0	0 0 1	1	
0 1 1	1 1 1	1	3
1 0 0	0 0 0	0	
1 0 1	0 1 0	1	
1 1 0	0 0 0	0	5
1 1 1	1 1 0	1	4

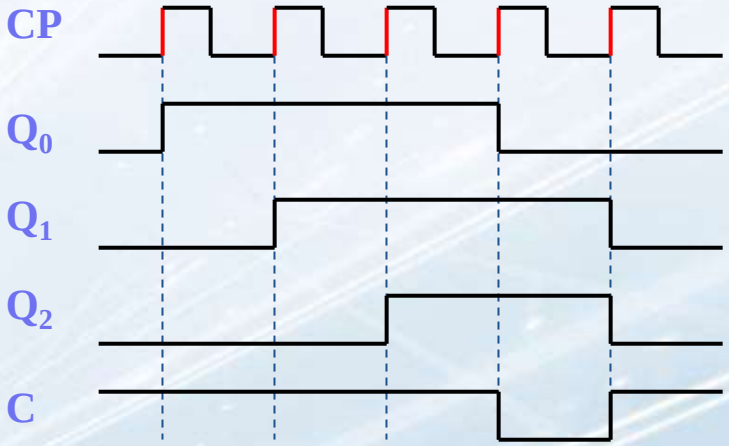
100

同步计数器练习详解（2/2）

$Q_2 Q_1 Q_0 ; / C$



根据状态转换图，画出时序图



D触发器在CP上升沿改变状态

电路特点：

同步五进制格雷码计数器，有自启动能力。

7.4.2 异步计数器

- 异步计数器也有二进制、十进制、任意进制等类型
- 异步计数器的特点
  - 输入系统时钟脉冲只作用于最低位触发器，高位触发器的时钟信号往往是由低一位触发器的输出提供的，高位触发器的翻转有待低一位触发器翻转后才能进行。
  - 由于每一级触发器都存在传输延迟，因此计数器工作速度慢，而且，位数越多计数越慢。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，尖峰信号也就越宽），使仪器设备产生误动作。
  - 优点：结构比较简单，所用元件较少。

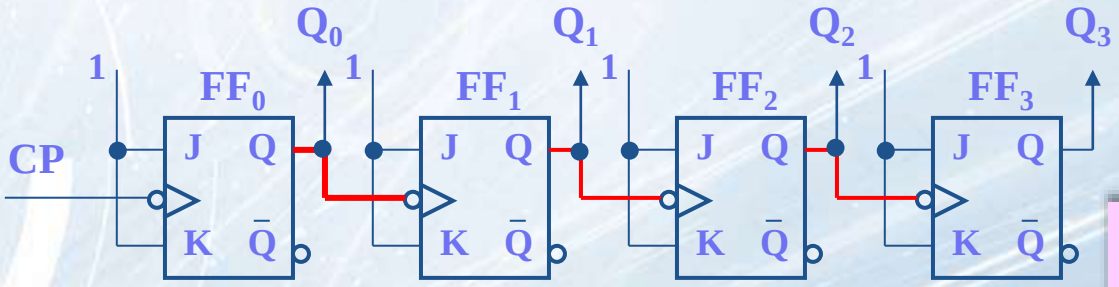
1、异步二进制计数器

电路结构特点：

（1）全部由T'触发器构成；

（2）第一级FF的CP由系统时钟控制，其余各级FF的CP端由前级FF的Q端或/Q端控制。

【例7.14】分析异步二进制（M=16）加法计数器电路（N=4）



FF_1、FF_2、FF_3的CP端分别接前级触发器的Q端

（1）状态方程

$$Q_0^{n+1}=\overline{Q_0^n}\cdot CP\downarrow;\ Q_1^{n+1}=\overline{Q_1^n}\cdot Q_0\downarrow;$$

$$Q_2^{n+1}=\overline{Q_2^n}\cdot Q_1\downarrow;\ Q_3^{n+1}=\overline{Q_3^n}\cdot Q_2\downarrow;$$

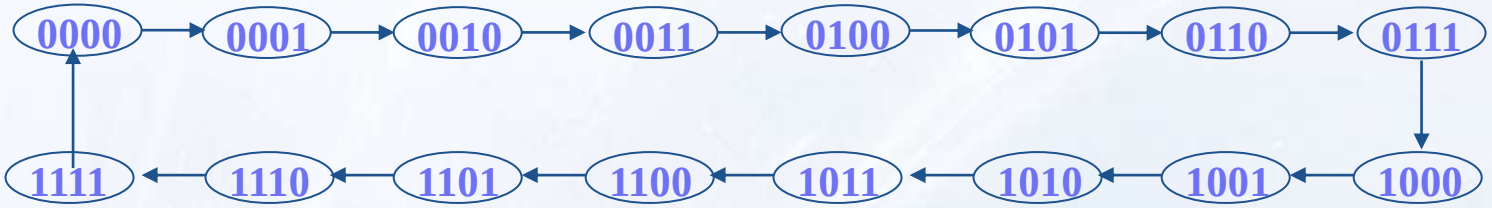
异步二进制计数器的状态转换表

（2）列出状态转换表

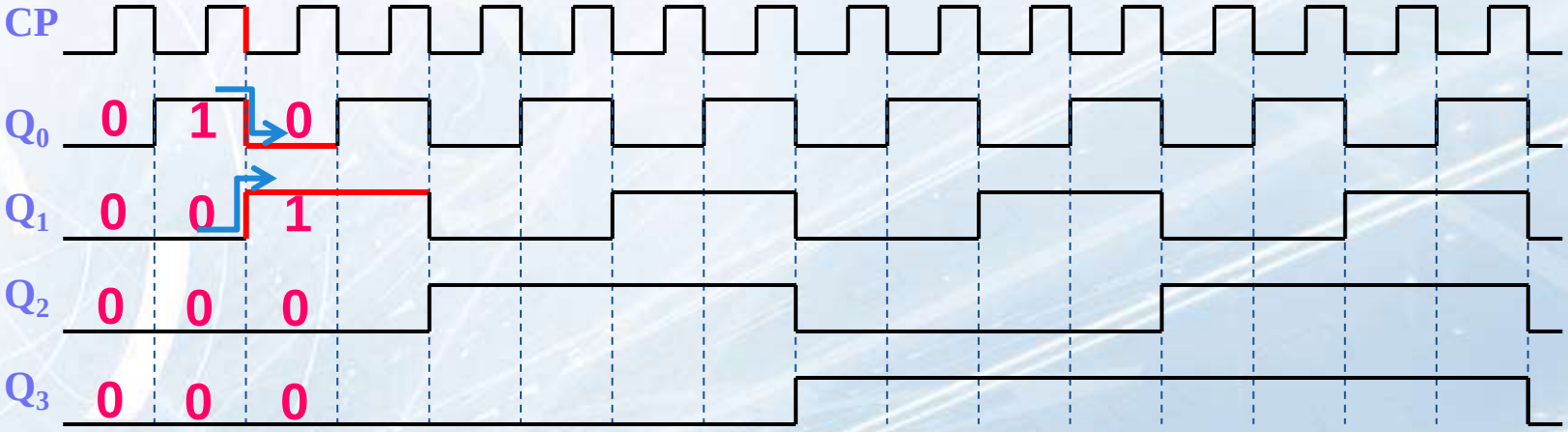
Q_3^n	Q_2^n	Q_1^n	Q_0^n	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

异步二进制计数器的状态转换图和时序图

（3）状态转换图 $Q_3Q_2Q_1Q_0$



时序图

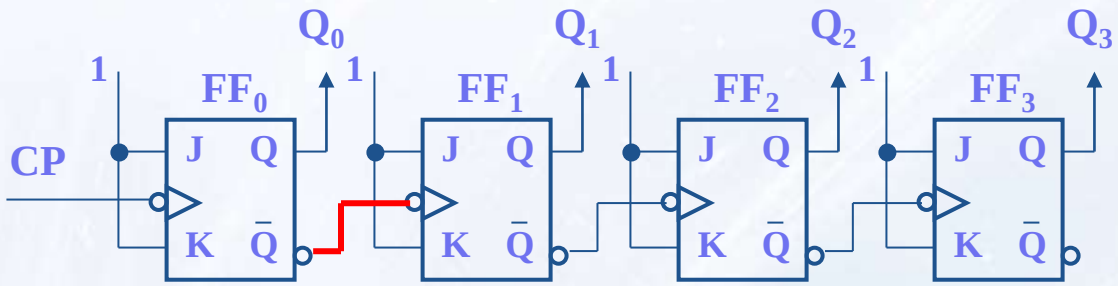


在第2个CP下降沿到来时，Q_0由1变为0， 则Q1由0变为1

（4）电路特点：异步二进制（M=16）加法计数器

【例7.15】异步二进制减法计数器

❖ 【例7.15】分析下图电路



FF_1、FF_2、FF_3的CP端分别接前级触发器的/Q端

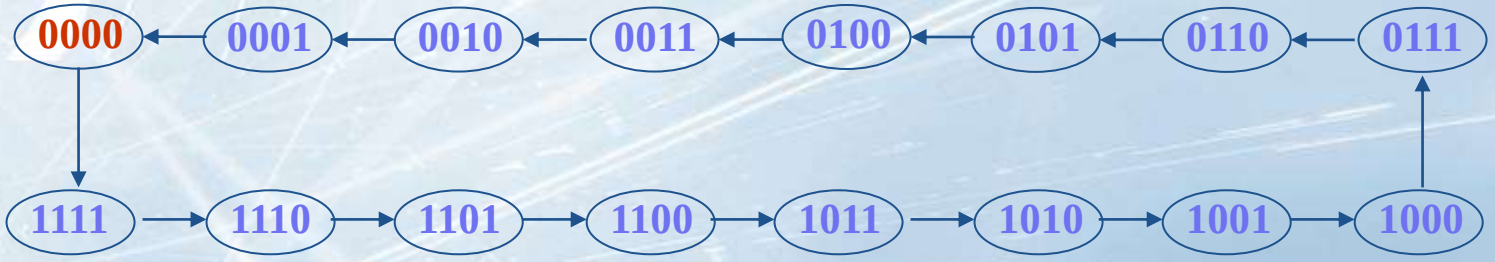
（1）状态方程

$$Q_0^{n+1}=\overline{Q_0^n}\cdot CP\downarrow;\ Q_1^{n+1}=\overline{Q_1^n}\cdot\overline{Q_0}\downarrow=\overline{Q_1^n}\cdot Q_0\uparrow;$$

$$Q_2^{n+1}=\overline{Q_2^n}\cdot\overline{Q_1}\downarrow=\overline{Q_2^n}\cdot Q_1\uparrow;\ Q_3^{n+1}=\overline{Q_3^n}\cdot\overline{Q_2}\downarrow=\overline{Q_3^n}\cdot Q_2\uparrow;$$

（3）状态转换图

$Q_3Q_2Q_1Q_0$



异步二进制减法计数器的状态转换表

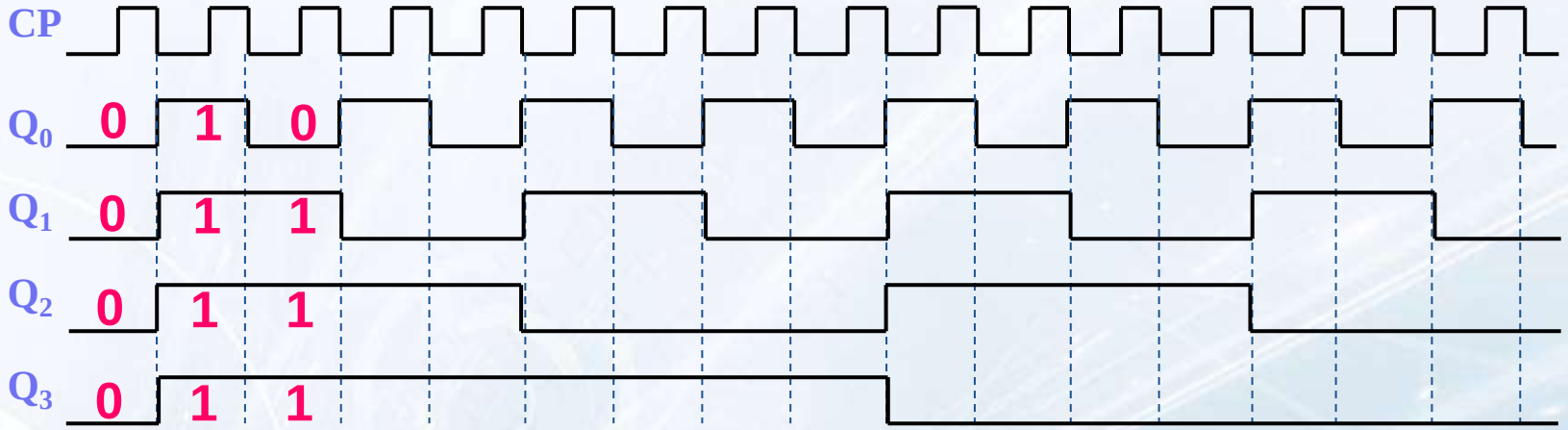
（2）列出状态转换表

不要省略!

Q_3^n	Q_2^n	Q_1^n	Q_0^n	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	0
0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0
1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	0

异步二进制减法计数器时序图和电路特点

（4）时序图



- 第1个CP下降沿到来时，Q_0由0变为1，Q_0的上升沿又使Q_1翻转，Q_1的上升沿又使Q_2翻转，Q_2的上升沿又使Q_3翻转，则Q_3 Q_2 Q_1 Q_0从0000变为1111；
- 第2个CP下降沿到来时，Q_0由1变为0，Q_0没有上升沿，则Q_1保持不变，同理，Q_2、Q_3保持不变，故Q_3 Q_2 Q_1 Q_0从1111变为1110——减计数。

（5）电路特点

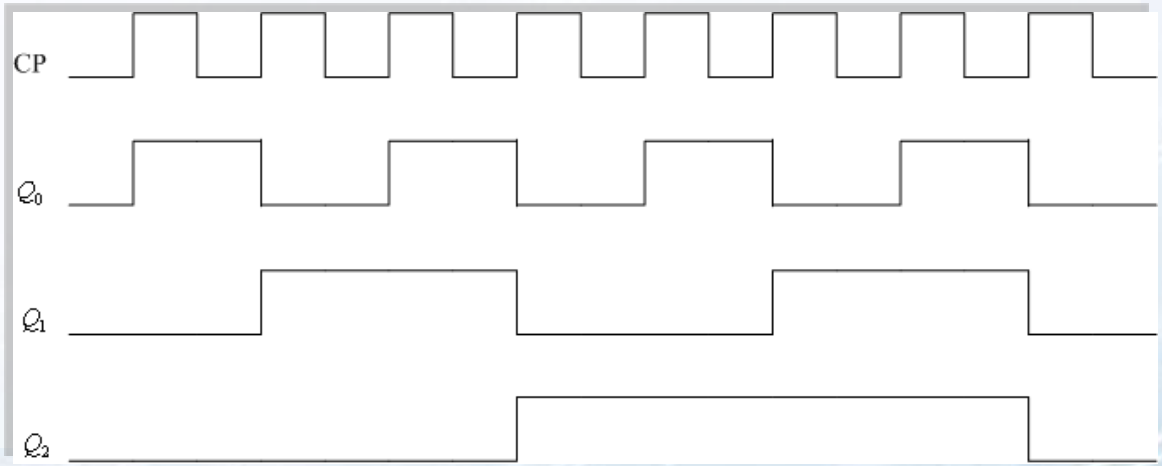
异步二进制（M=16）减法计数器

如何用D-FF构成异步二进制加法计数器（M=8）？

- 设计思路

假定D触发器CP上升沿触发

（1）根据题意画出加法计数器的时序图



（2）根据时序图写出状态方程

$$Q_0^{n+1} = \overline{Q_0}^{\,n} \cdot CP \uparrow$$

$$Q_1^{n+1} = \overline{Q_1}^{\,n} \cdot Q_0 \downarrow$$

$$Q_2^{n+1} = \overline{Q_2}^{\,n} \cdot Q_1 \downarrow$$

（3）将D-FF的特性方程代入状态方程，得到驱动方程

$$D_0 = \overline{Q_0}^{\,n} \cdot CP \uparrow$$

$$D_1 = \overline{Q_1}^{\,n} \cdot Q_0 \downarrow$$

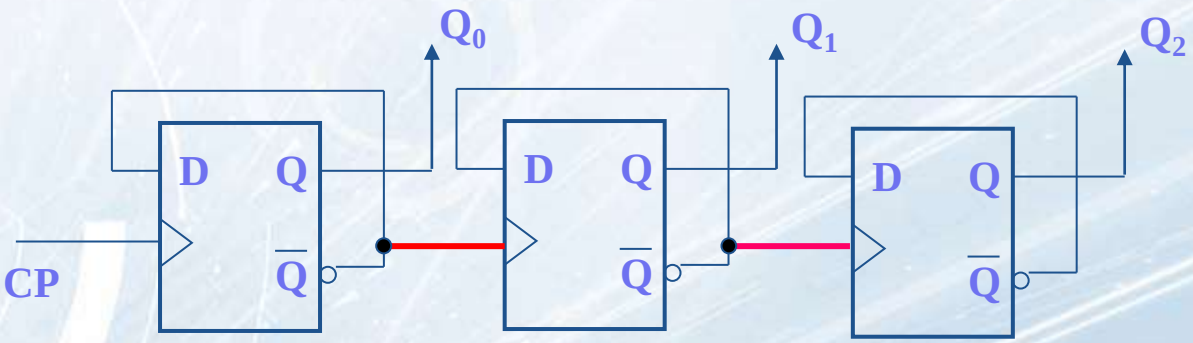
$$D_2 = \overline{Q_2}^{\,n} \cdot Q_1 \downarrow$$

109

画出电路图

（4）画出电路图

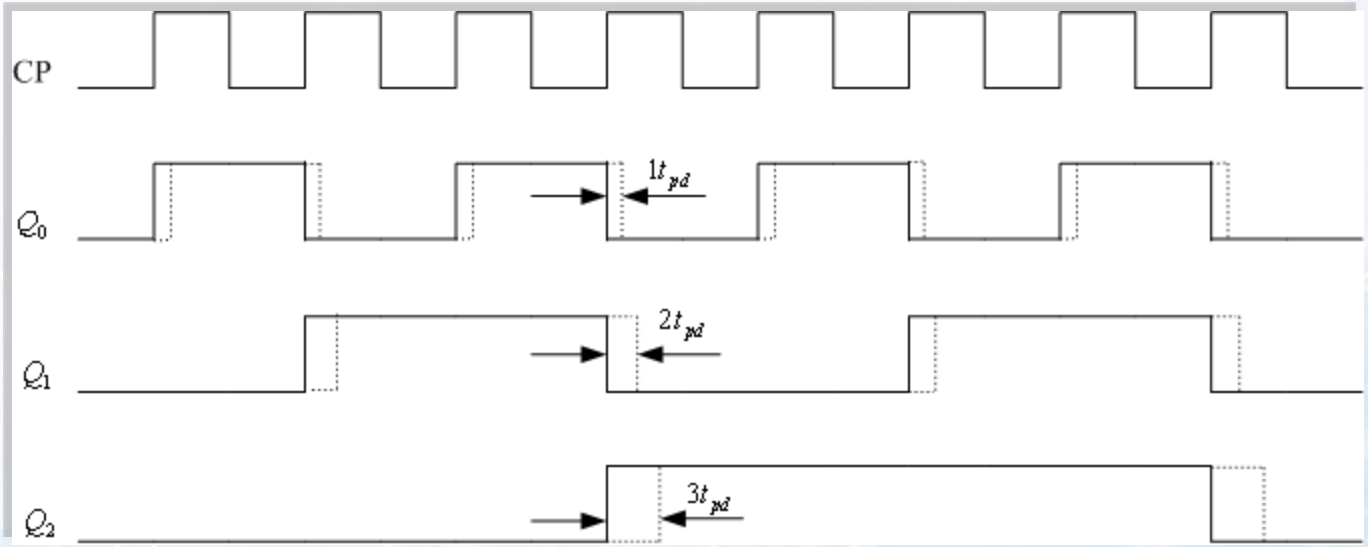
因为假定D触发器CP上升沿触发，而在驱动方程中FF_1、FF_2分别是在Q_0的下降沿和Q_1的下降沿翻转，也即在/Q_0的上升沿和/Q_1的上升沿翻转。故分别将/Q_0和/Q_1作为FF_1、FF_2的时钟信号，由此画出电路图。



3位二进制（M=8）异步加法计数器

110

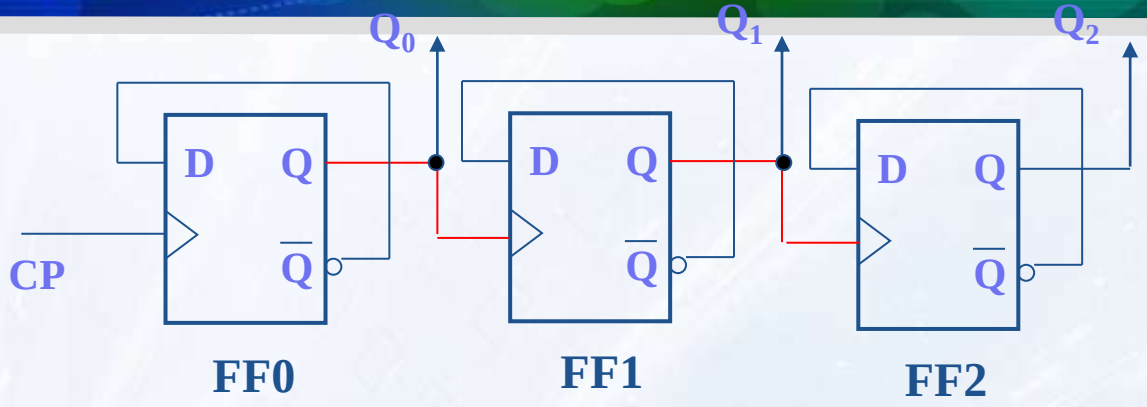
异步二进制加法计数器的实际时序图



- 假定各触发器的传输延迟时间为t_{pd}时，一个n 位二进制异步计数器，从一个计数脉冲（设上升沿触发）到来，到n 个触发器都翻转稳定，需要经历的最长时间是nt_{pd}。
- 为保证计数器的状态能正确反映计数脉冲的个数，下一个计数脉冲必须在nt_{pd} 后到来，因此计数脉冲的最小周期$T_{min}=nt_{pd}$。

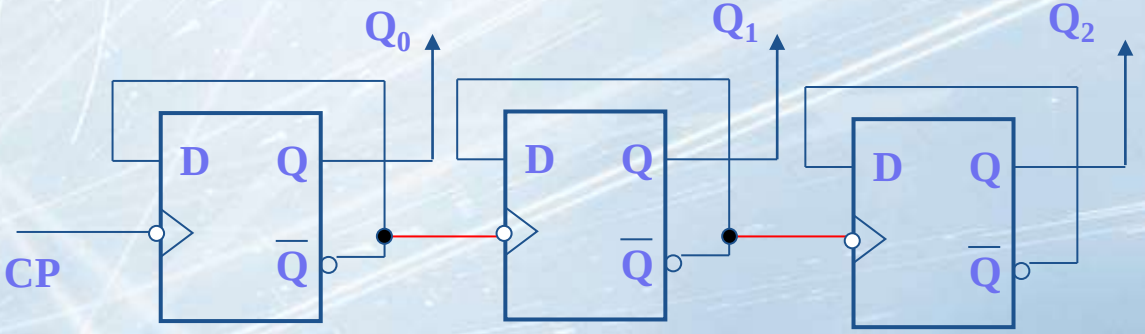
111

思考题



- 请同学们自行分析上图的工作原理，说明该电路构成了何种计数器，并画出时序图

- 问题：如果D触发器是下降沿触发，那么D触发器二进制加法/减法计数器如何构成？下图是何种计数器？



112

思考题的分析思路

FF_1、FF_2的CP端分别接前级触发器的Q端

- 假设初态$Q_2^nQ_1^nQ_0^n$为000，则当CP上升沿到来时，$Q_0^{n+1}=1$；由于Q_0由0变为1，所以Q_1翻转，则$Q_1^{n+1}=1$；由于Q_1由0变为1，所以Q_2翻转，则$Q_2^{n+1}=1$。
- 故对应原态000，次态为111。

（1）状态方程

$$Q_0^{n+1}=D_0=\overline{Q_0}^n\cdot CP\uparrow$$

$$Q_1^{n+1}=D_1=\overline{Q_1}^n\cdot Q_0\uparrow$$

$$Q_2^{n+1}=D_2=\overline{Q_2}^n\cdot Q_1\uparrow$$

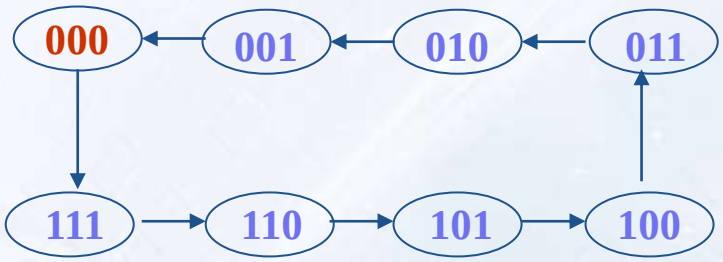
（2）列出状态转换表

$Q_2^nQ_1^nQ_0^n$	$Q_2^{n+1}Q_1^{n+1}Q_0^{n+1}$
0 0 0	1 1 1
0 0 1	0 0 0
0 1 0	0 0 1
0 1 1	0 1 0
1 0 0	0 1 1
1 0 1	1 0 0
1 1 0	1 0 1
1 1 1	1 1 0

思考题的状态转换图

（3）状态转换图

$Q_2 Q_1 Q_0$



（4）时序图

（略）

（5）电路特点

异步二进制（M=8）减法计数器

二进制计数器的设计方法

问题：如果D触发器是下降沿触发，那么D触发器二进制加法/减法计数器如何构成?

（1）根据题意画出计数器的时序图

假定各触发器的初始状态为0，根据计数器的计数规律，在每个CP的下降沿到来时，画出各触发器的输出电平。

（2）根据时序图写出状态方程

（3）将D-FF的特性方程代入状态方程，得到驱动方程

（4）根据触发器的触发方式，以及驱动方程中触发器是在Q的下降沿还是在上升沿翻转，确定是将前级触发器的Q端还是/Q端作为触发器的时钟信号，画出电路图

115

n位二进制异步计数器总结

- n位二进制异步计数器由n个处于计数工作状态（对于D 触发器，使$D_i=/Q_i$；对于JK 触发器，使$J_i=K_i=1$）的触发器组成。各触发器之间的连接方式由加、减计数方式及触发器的触发方式决定。
  - 对于加计数器，若用上升沿触发的触发器组成，则应将低位触发器的Q非端与相邻高一位触发器的时钟脉冲输入端相连（即进位信号应从触发器的Q非端引出）；若下降沿触发，则应将低位触发器的Q端与相邻高一位触发器的时钟脉冲输入端连接。
  - 对于减计数器，各触发器的连接方式则相反。若触发器上升沿触发，则应将低位触发器的Q端与相邻高一位触发器的时钟脉冲输入端相连；若下降沿触发，则应将低位触发器的Q非端与相邻高一位触发器的时钟脉冲输入端连接。
- 在二进制异步计数器中，高位触发器的状态翻转必须在低一位触发器产生进位信号（加计数）或借位信号（减计数）之后才能实现。故又称这种类型的计数器为串行计数器。也正因为如此，异步计数器的工作速度较低。

2、用反馈复位法实现异步M制计数器

- 异步二进制的计数器电路简单，它的模值是2^n，如果不能改变这个模值，则使用范围就要受到限制。
- 反馈复位法可以改变计数器的模值，得到任意模值的计数器
- 反馈复位法原理：当计数器计到规定的模值时，将计数器为“1”的输出送至反馈电路，产生置0信号/R_D使计数器复位，完成一次计数循环。

- 反馈复位法实现步骤：

（1）根据计数器模值，求反馈复位代码S_M，即计数器模值的二进制代码；

（2）求反馈复位逻辑：将输出为1的FF的Q端信号进行逻辑乘后取反，作为反馈复位信号；$\overline{R_D} = \overline{\Pi Q^1}$

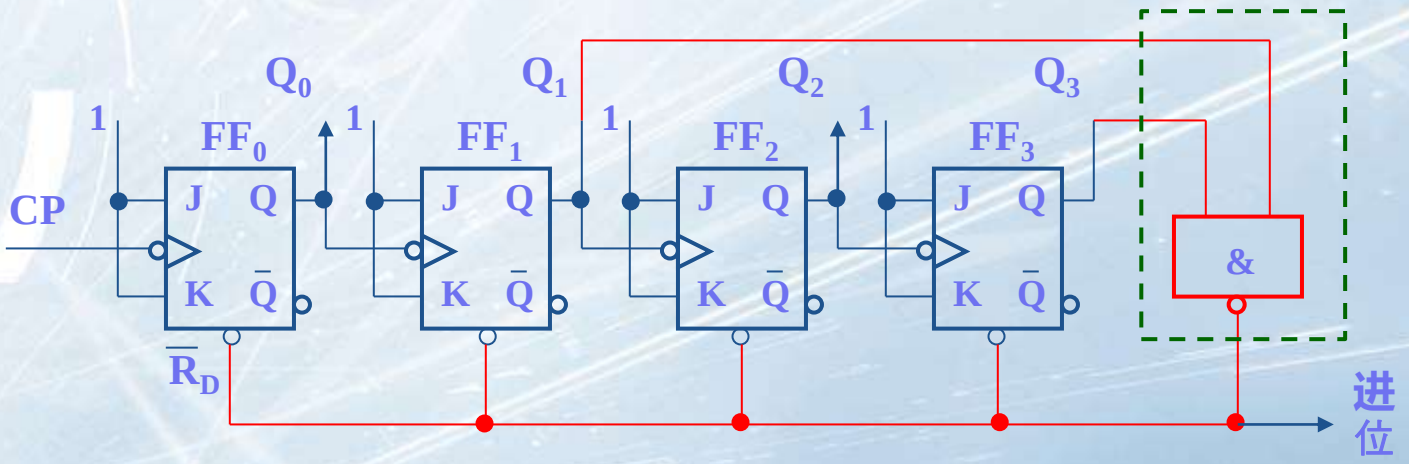
（3）画逻辑图（先画出由N级FF构成的异步二进制计数器，然后加入反馈复位逻辑)。

【例7.16】用反馈复位法设计异步十进制加法计数器。

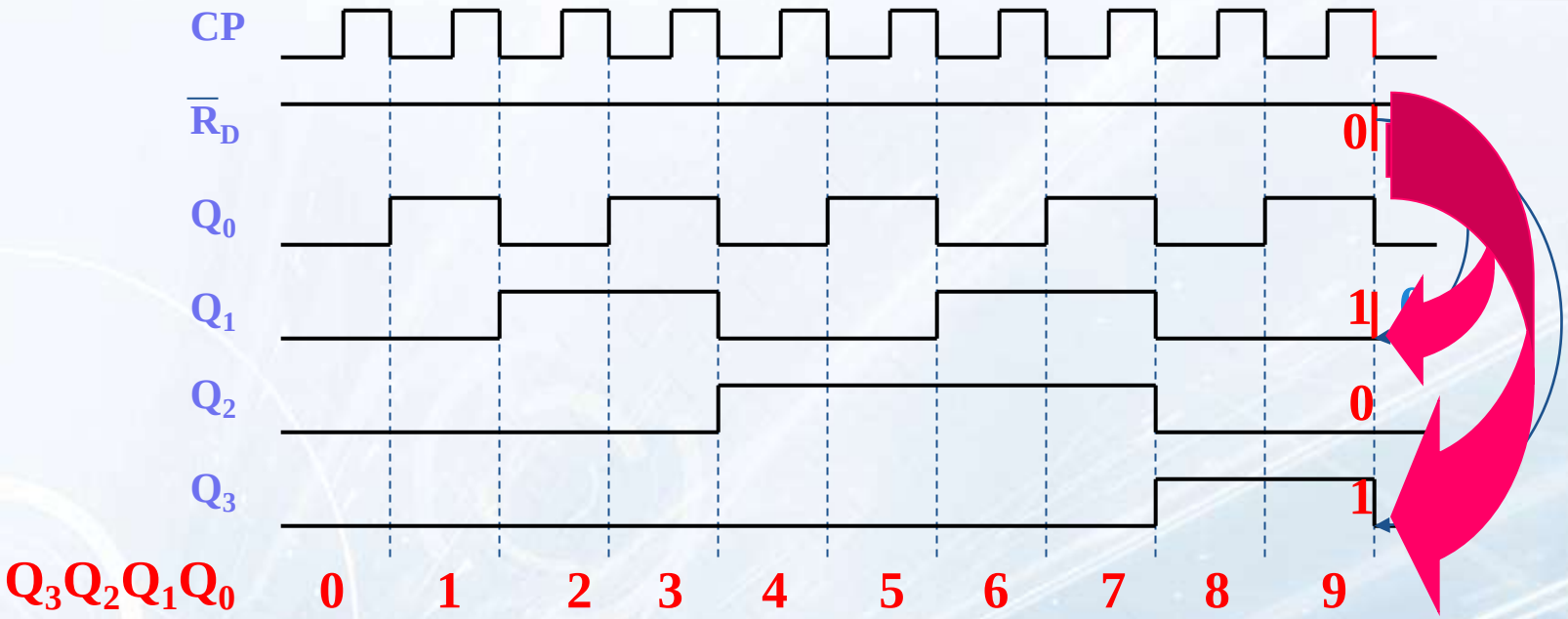
（1）求反馈复位代码 $S_M=(10)_{10}=(1010)_2(N=4即Q_3Q_2Q_1Q_0)$

（2）求反馈复位逻辑 $\overline{R_D}=\overline{\Pi Q^1}=\overline{Q_3Q_1}$

（3）画逻辑图



异步十进制加法计数器时序图

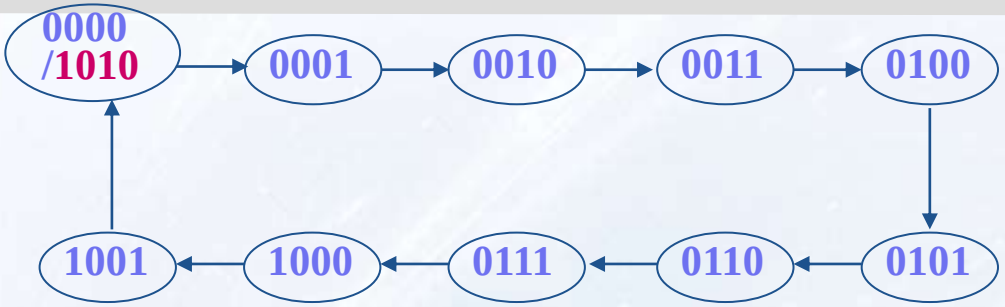


- 当第10个CP下降沿到来时，计数器进入1010状态，Q_3Q_1=11使/R_D=0，计数器复位，$Q_3Q_2Q_1Q_0$变为0000；Q_3Q_1=00又使/R_D变为1——1010状态和/R_D=0只出现了瞬间
- 1010状态称为过渡状态，它与0000状态占用一个时钟周期，所以把它们合并在一起。

异步十进制激发计数器状态转换图

$Q_3Q_2Q_1Q_0$

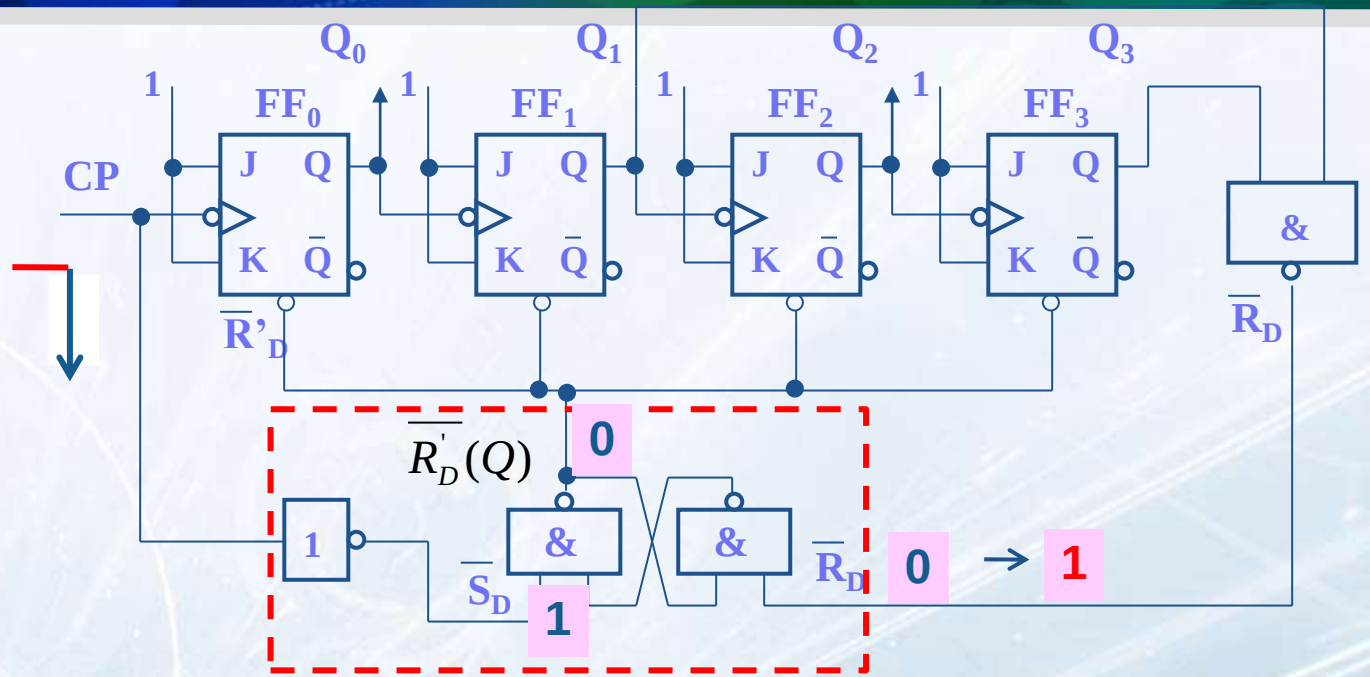
1010为过渡状态



状态转换图

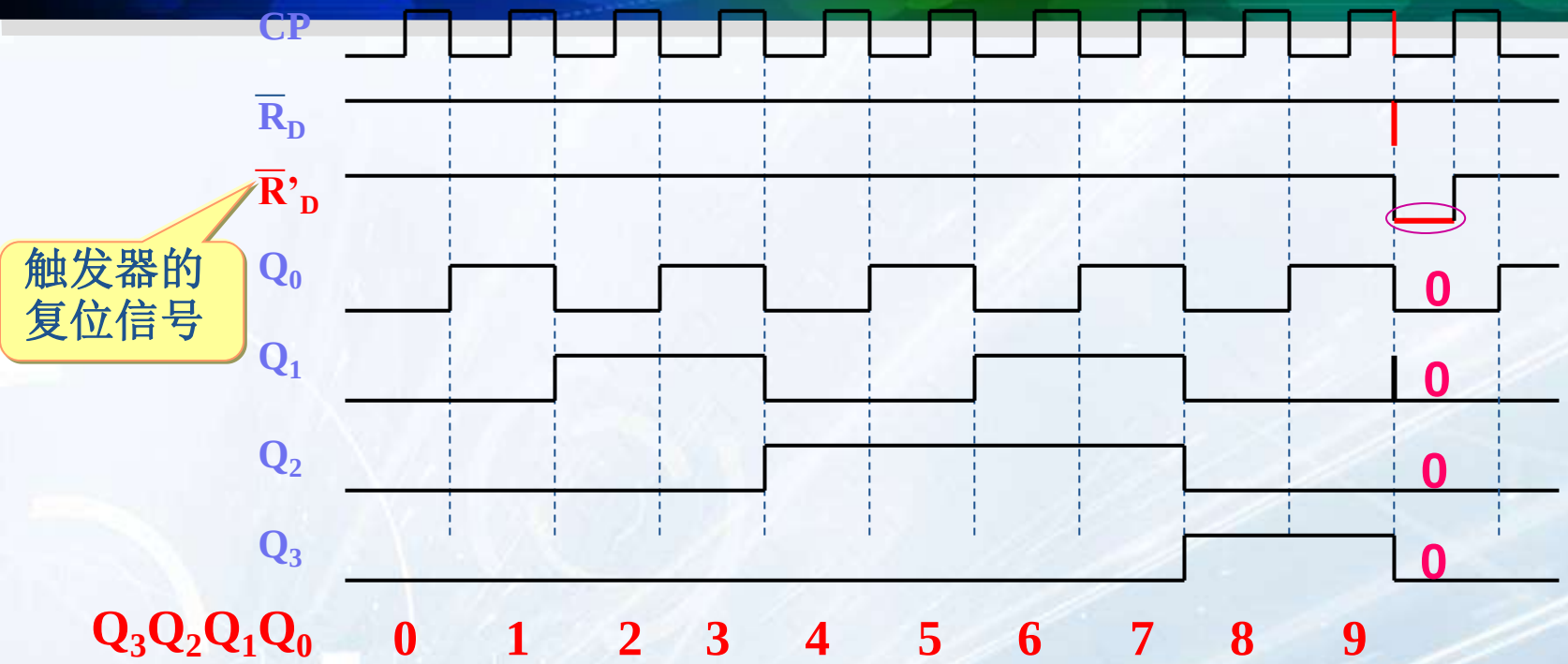
- 上述反馈复位电路存在问题： $/R_D$的作用时间非常短暂，可能不会使全部FF复位，达不到反馈复位的目的
- 改进：增加基本RS触发器，增加$/R_D$的作用时间

反馈复位改进电路



- 当CP为1时，/S_D变为0，基本RS触发器置1，计数器不复位。
- 当第10个CP下降沿到来时（则/S_D＝1），计数器进入1010状态，Q_3Q_1=11使/R_D=0，故基本RS触发器置0(/R_D'=0)，使得计数器复位。
- 当/R_D信号撤销后(即/R_D＝1），此时CP＝0，/S_D为1，则基本RS触发器保持不变(/R_D'仍=0)，从而保证计数器能够可靠复位。

反馈复位改进电路的时序图

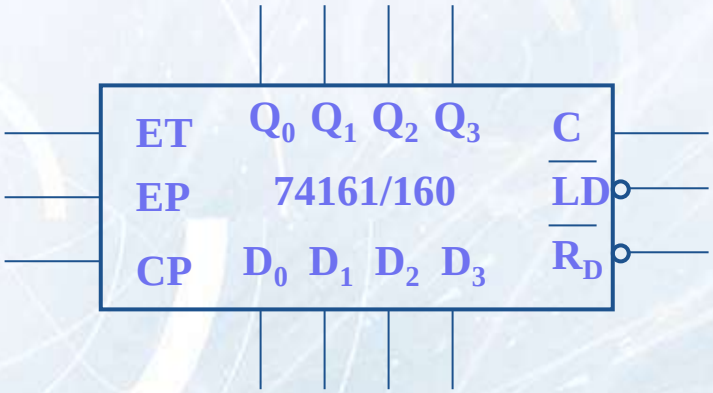


- 当CP为1时，/S_D变为0，基本RS触发器置1 (/R_D'=1)。
- 当第10个CP下降沿到来时（即/S_D = 1），计数器进入1010状态，Q_3Q_1=11使/R_D=0，则基本RS触发器置0 (/R_D'=0)，使得计数器复位。当/R_D信号撤销后，此时CP = 0，/S_D为1，基本RS触发器保持0不变(/R_D'仍=0)，计数器仍为0000。
- 只有当CP上升沿再次到来时，/R_D'才从0变为1。/R_D'的低电平时间等于时钟的低电平时间。

7.4.3 集成计数器

- 集成计数器包括异步和同步计数器。异步计数器有二进制、十进制及可变进制异步计数器。同步计数器包括加法、减法、 加/减（可逆）二进制、十进制同步计数器。
- 4位同步二进制加法计数器74161（异步清除）、74163（同步清除），同步十进制加法计数器74160 （异步清除） 、74162（同步清除）

逻辑符号（74161与74160相同）



进位输出C:

$$74161：\ C = ET \cdot Q_3^n \cdot Q_2^n \cdot Q_1^n \cdot Q_0^n$$

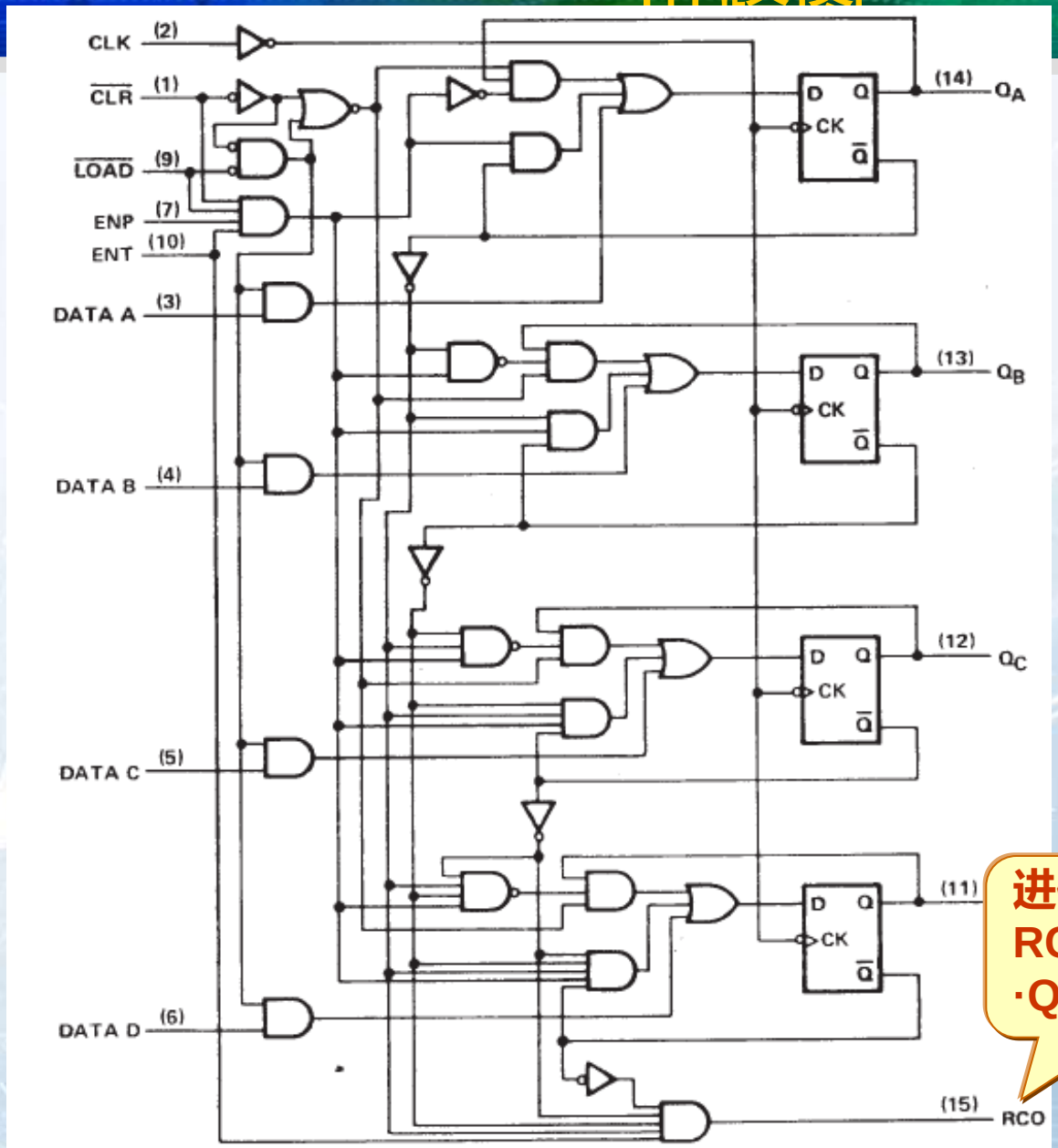
$$74160：\ C = ET \cdot Q_3^n \cdot Q_0^n$$

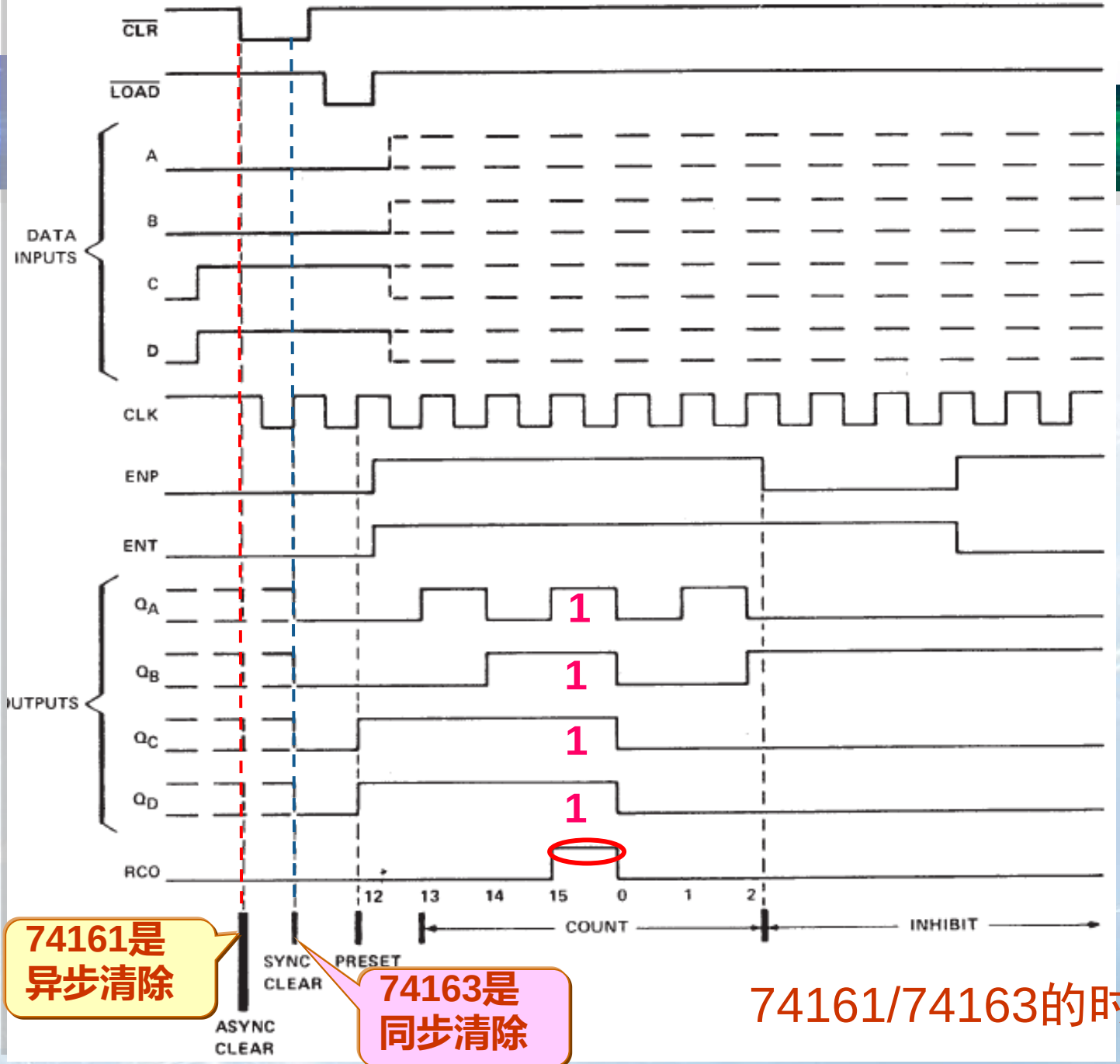
74161与74160的功能表

$\overline{R_D}$	$\overline{LD}$	EP	ET	CP	功能
0	×	×	×	×	异步复位
1	0	×	×	↑	同步预置
1	1	0	0	↑	保持
1	1	0	1	↑	保持
1	1	1	0	↑	保持
1	1	1	1	↑	计数

- EP、ET：功能控制端，为11时计数器计数，其他情况下计数器保持。用于多个计数器的级联

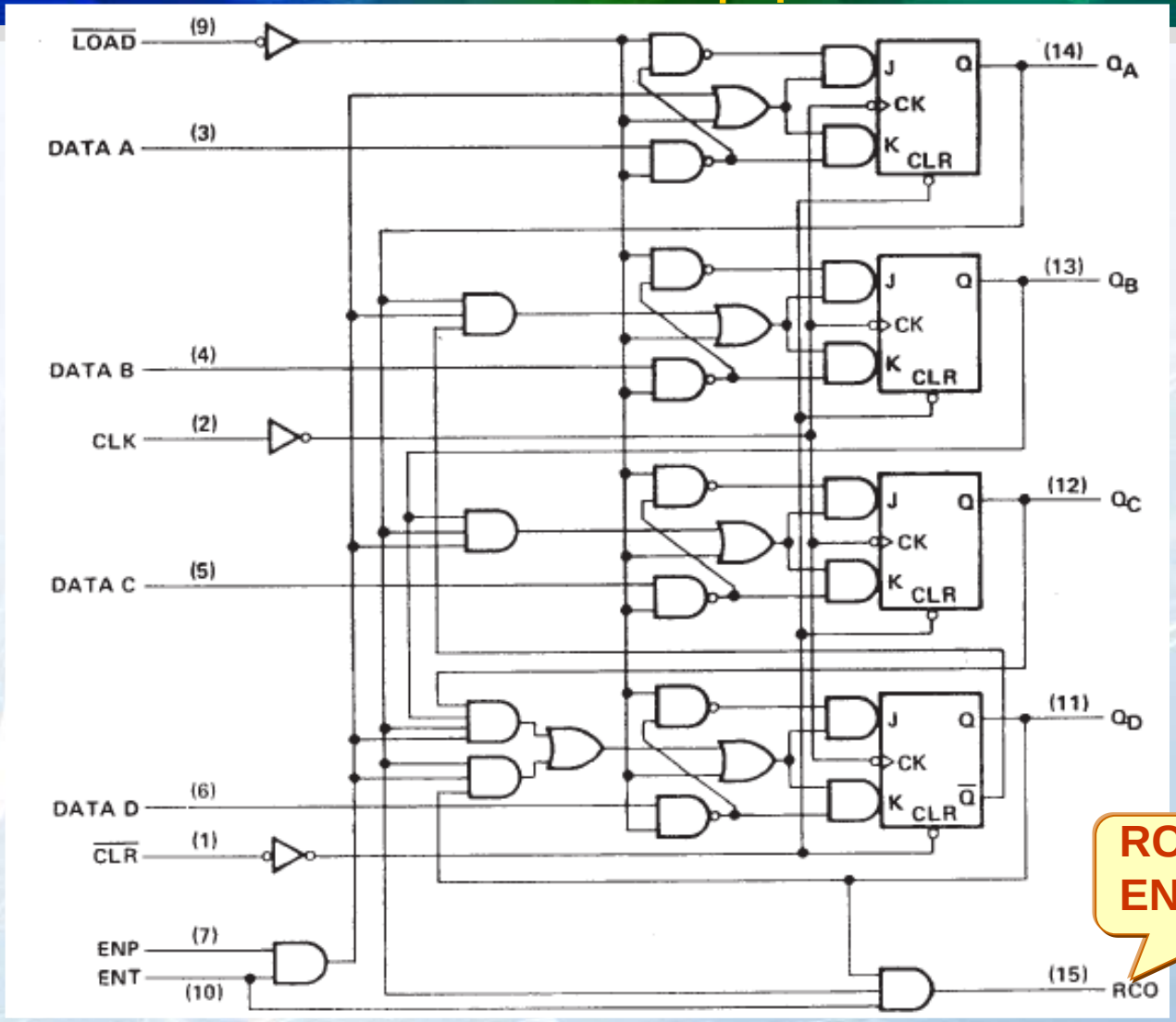
同步二进制加法计数器74161/74163的电路图

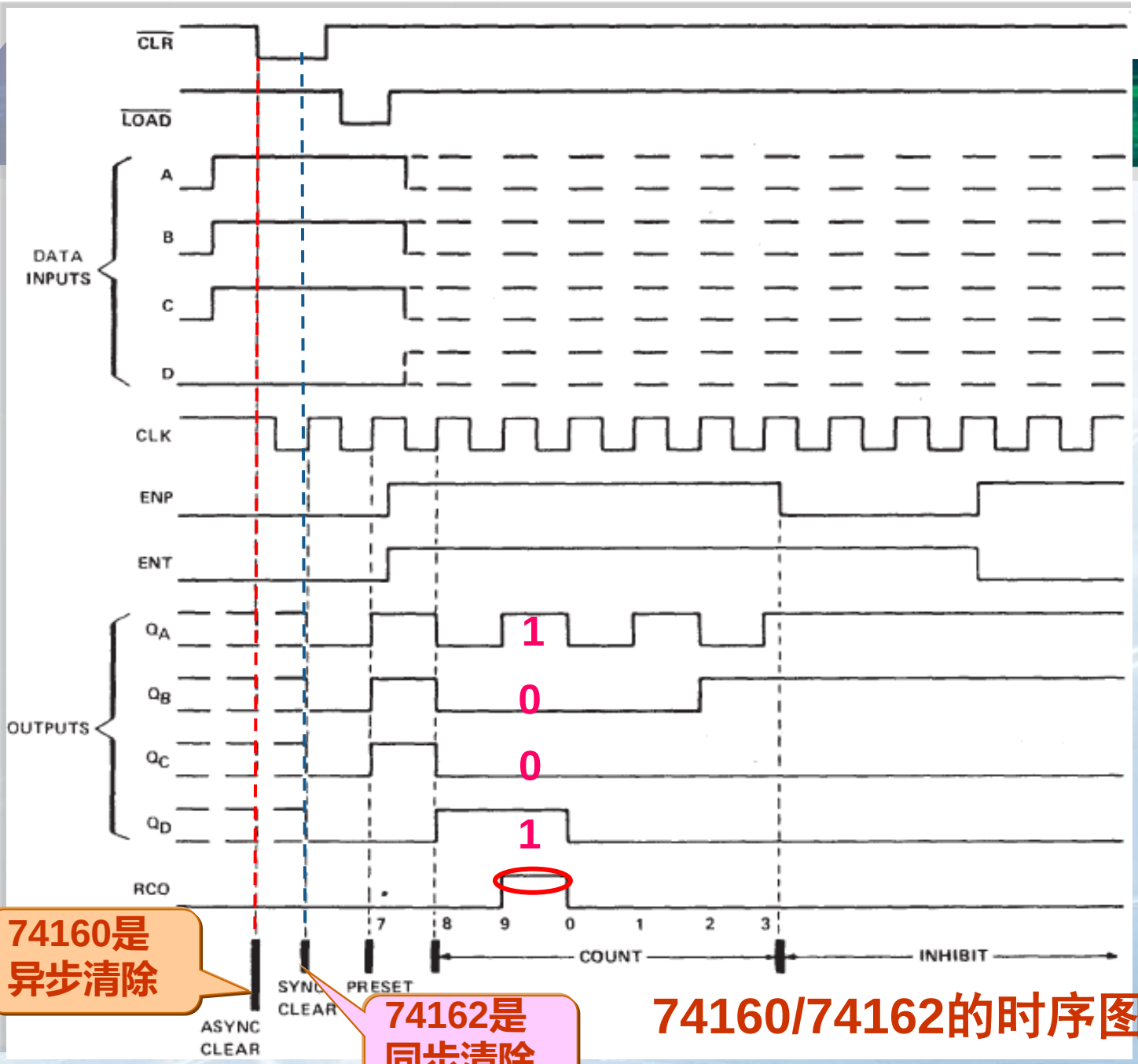




74161/74163的时序图

同步十进制加法计数器的电路图

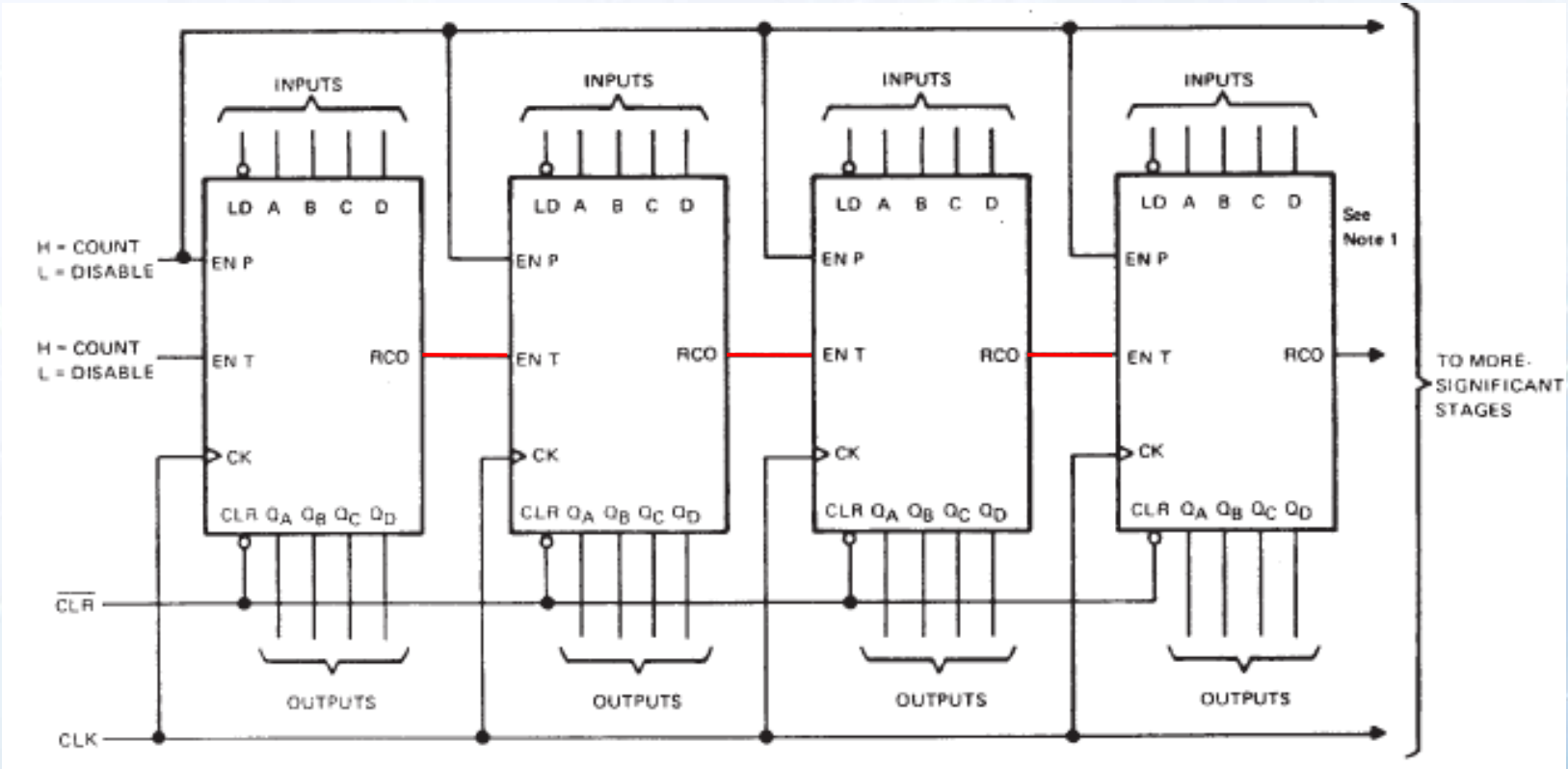




74160/74162的时序图

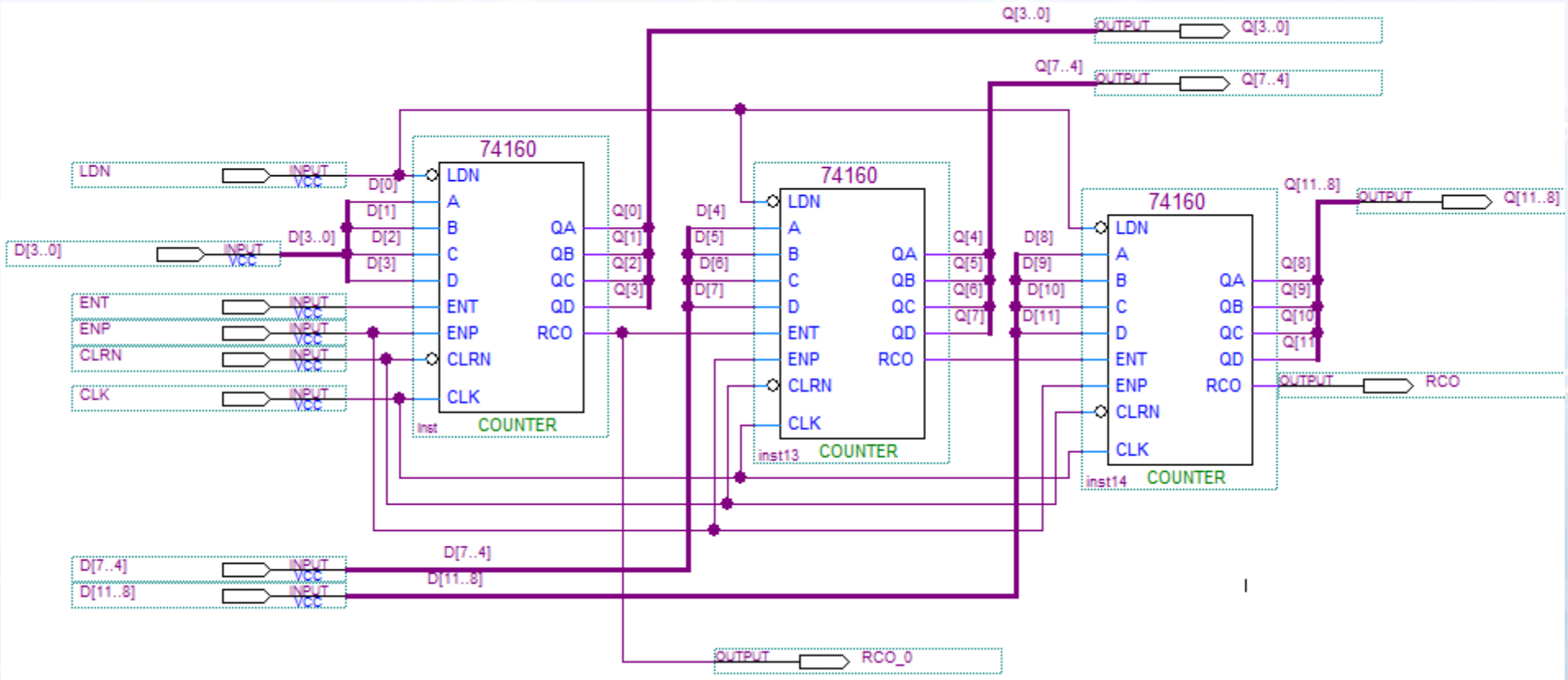
纹波模式进位电路构成N位同步计数器

- 将多个计数器级联，构成N位同步计数器。74160/162以BCD计数，74161/163以二进制计数。

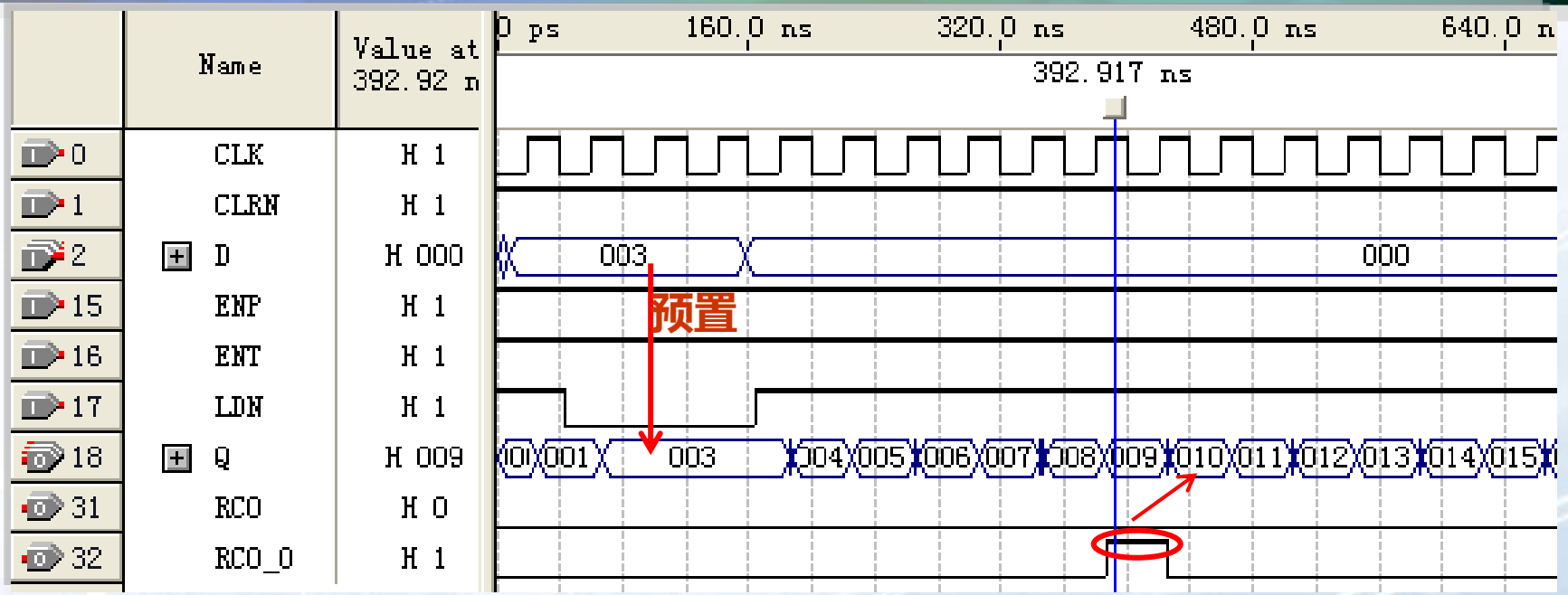


由于前级的RCO接后级的ENT，级数越多，系统工作频率越下降

由74160构成的3位BCD计数器



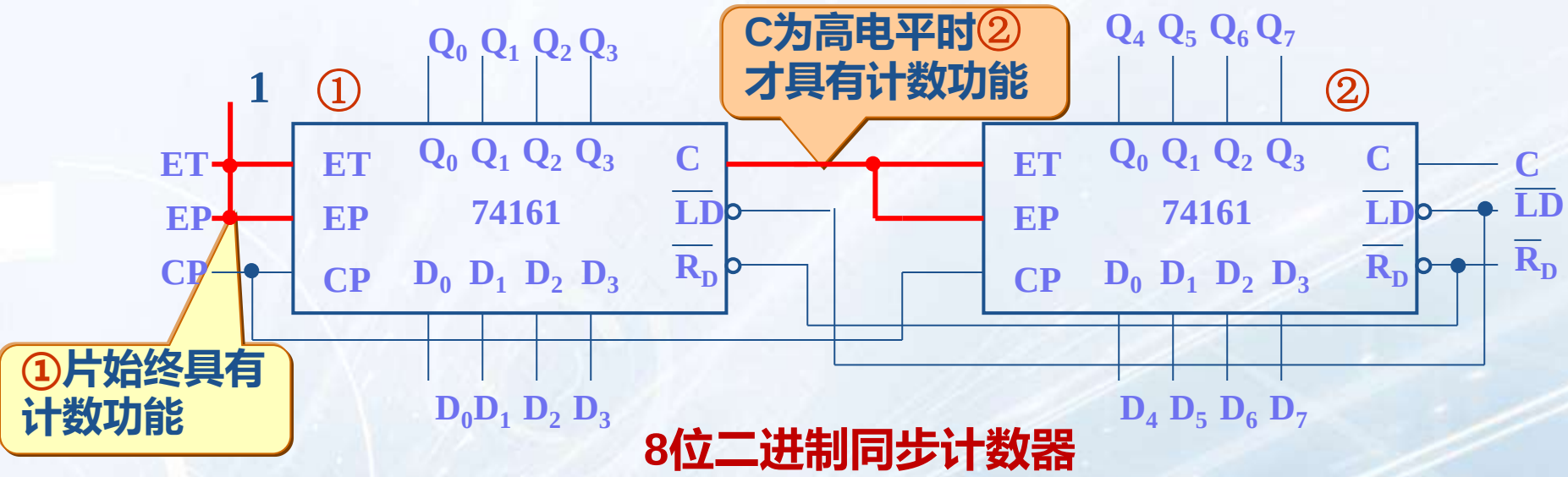
由74160构成的3位BCD计数器仿真波形



- 当第1级计数器计到1001（9_{10}）时，其RCO=1，使第2级计数器的ENT=1；在下一个CLK到来时，第2级计数器开始计数，即$Q_7Q_6Q_5Q_4$=0001（1_{10}），第1级计数器变为0000，则其RCO=0；在下一个CLK 到来时，第2级计数器保持刚才的状态0001，接着第1级计数器继续计数，故BCD计数器的十位保持为1，个位分别为1、2、3……，直到9。

集成计数器的同步扩展

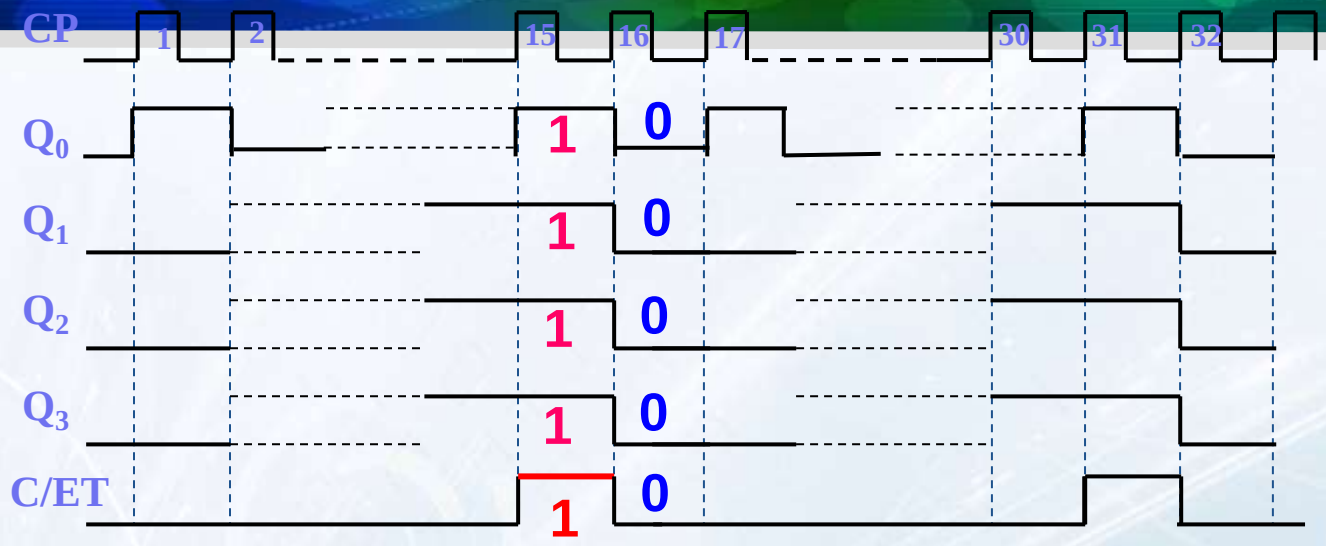
将若干片集成计数器级联起来，形成有较大模值的计数系统。



8位二进制同步计数器

- 同步扩展指所有计数器使用同一个时钟信号来同步，即把两片4位二进制计数器的CP并联后接时钟信号CP。
- 低位片的ET、EP接高电平，使低位片始终具有计数功能；高位片的ET、EP接低位片的进位输出端C，只有当C为高电平时，高位片才具有计数功能。

集成计数器的同步扩展——原理分析



①片的时序图

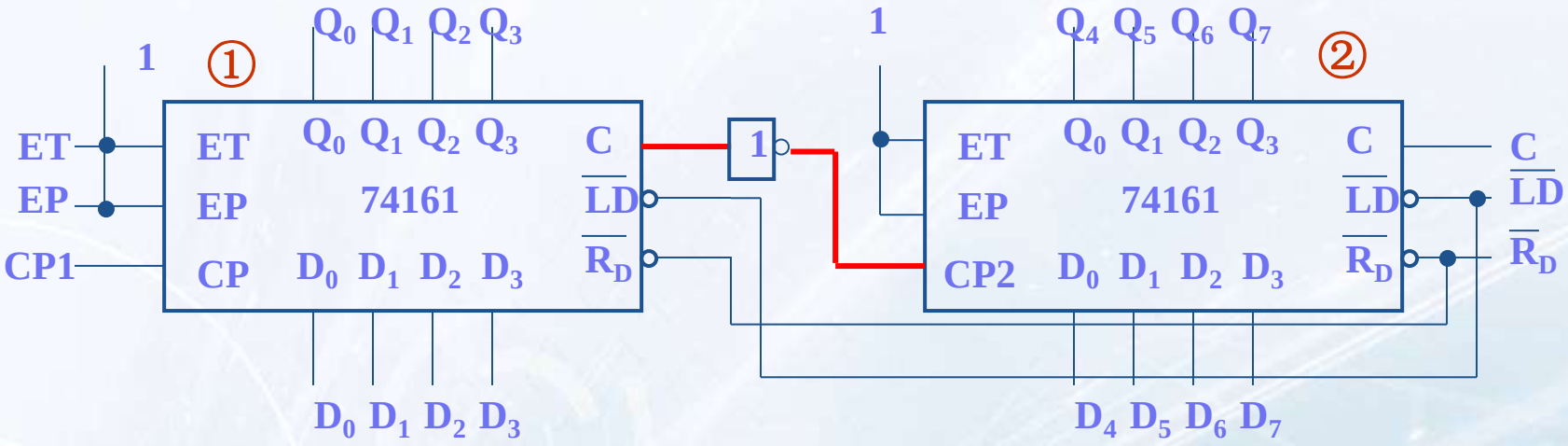
- 假定计数器从0000状态开始计数。在输入15个CP之前，低位片按时钟信号加1计数，其进位输出C都为0，则高位片的ET、EP=0，高位片不工作，保持0000不变。
- 输入15个CP后，低位片的状态变为1111，使其进位输出C=1；
- 当第16个CP到来后，低位片和高位片同时计数，低位片的状态由1111变为0000，其进位输出C从1变为0，高位片的状态由0000递增到0001。可见高位片是每隔16个CP，才能完成一次计数操作。当第16个CP到来后，低位片加1计数，而高位片保持状态0001不变（因为ET=EP=C=0）。

状态转换表

	Q_7^n	Q_6^n	Q_5^n	Q_4^n	Q_3^n	Q_2^n	Q_1^n	Q_0^n	Q_7^{n+1}	Q_6^{n+1}	Q_5^{n+1}	Q_4^{n+1}	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}	C
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0
3	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0
5	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0
6	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1	0
7	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0
				.....								.....					0
14	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1	0
15	0	0	0	0	1	1	1	1	0	0	0	**1**	0	0	0	0	0
16	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	**1**	0
17	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0
				.....								.....					0
	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0
255	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	**1**

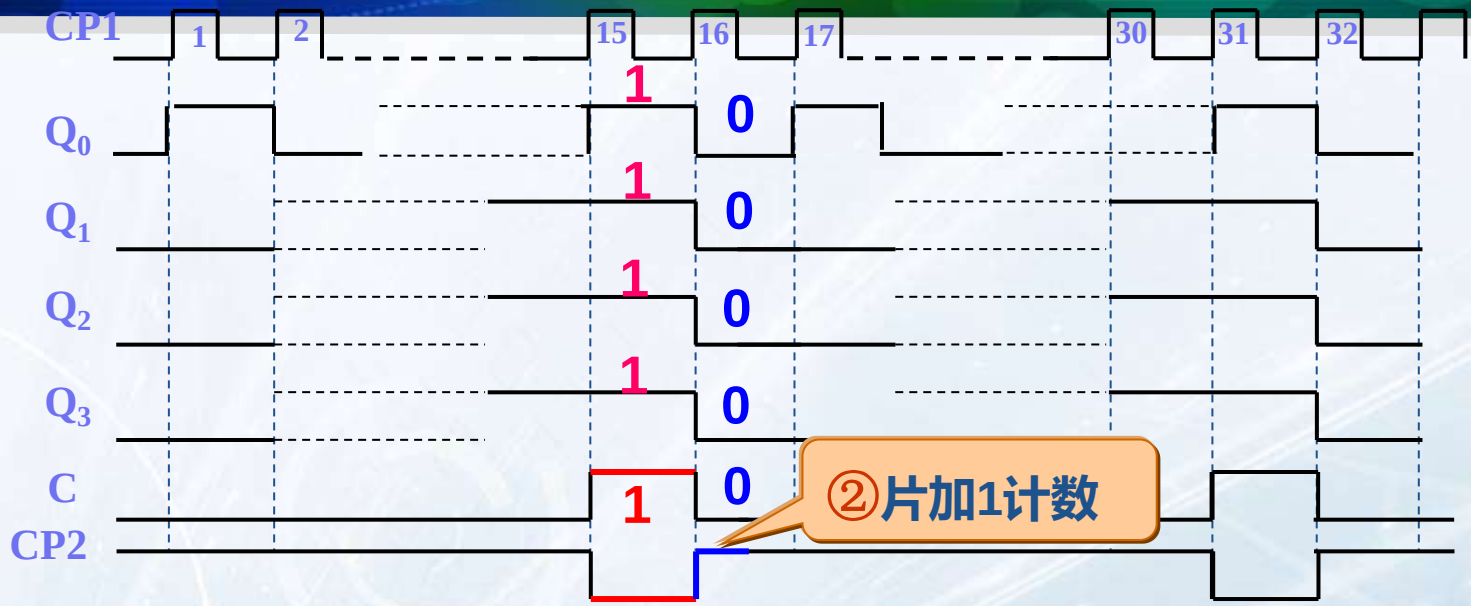
问题： 8位二进制同步计数器模值为多少?

集成计数器的异步扩展（异步法）



- 异步指多个计数器不是统一由一个时钟信号来同步的，各自有单独的时钟。
- 低位片4位二进制计数器的CP接系统时钟信号CP1，低位片的进位输出端C经反相后接高位片的CP端。

集成计数器的异步扩展——原理分析



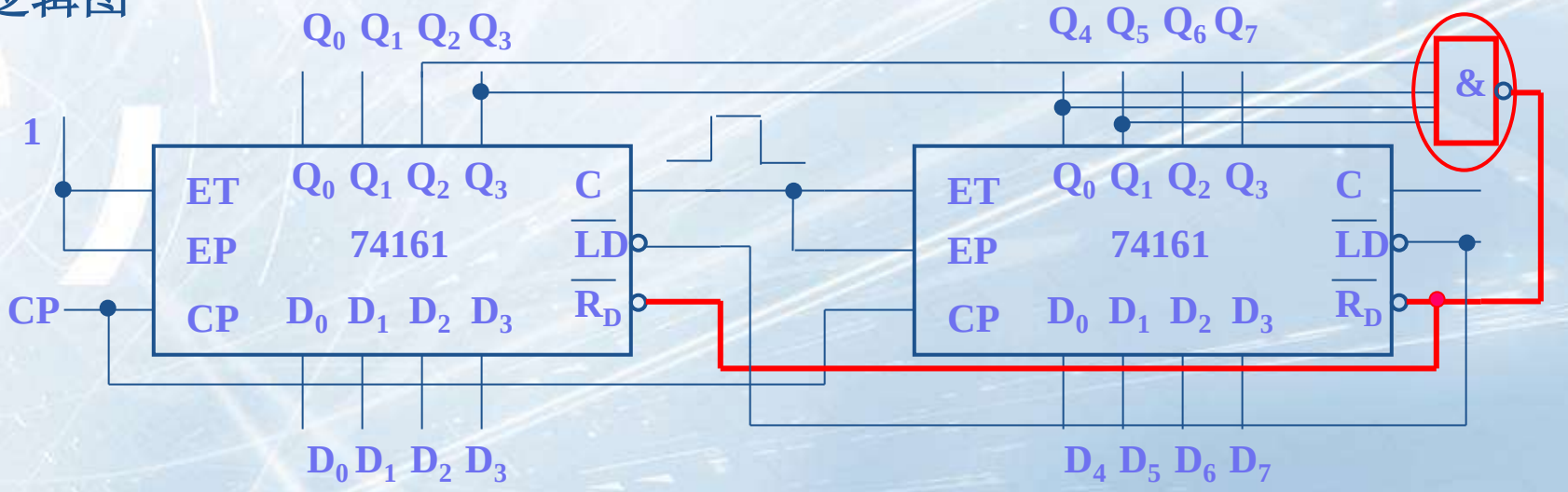
- 假定计数器从0000状态开始计数。在输入15个CP之前，低位片按时钟信号加1计数，其进位输出C都为0，则CP2=/C=1，高位片不工作，保持0000不变。
- 当输入15个CP后，低位片的状态变为1111，使其进位输出C从0变为1，则CP2=/C=0。
- 当第16个CP到来后，低位片的状态由1111变为0000，C从1变为0，则CP2从0变为1，使高位片加1计数，状态由0000变为0001。
- 当第17个CP到来后，低位片的状态由0000变为0001；由于低位片的C为0，则CP2=/C=1，高位片不工作，保持状态0001不变。

集成计数器实现M进制计数

- 假如一个计数器模值为M，则两片级联后，模值变为M^2。但有时需要的计数器模值不是M^2，例如60进制、24进制
- 反馈复位法或预置法，可以得到任意进制计数器

【例7.17】用74161（二进制）实现M=60的加法计数器

- 反馈复位代码 $S_M = (60)_{10} = (111100)_2 (N = 6 即 Q_5Q_4Q_3Q_2Q_1Q_0)$
- 反馈复位逻辑 $\overline{R_D} = \overline{\Pi Q^1} = \overline{Q_5Q_4Q_3Q_2}$
- 画逻辑图



（1）输出C预置法

- 预置法——利用计数器的预置功能，改变计数器的模值，得到任意进制计数器。
- 包括输出C预置法和输出Q预置法。

（1）输出C预置法

将输出C经反相后送预置端/LD；将计数器的预置数据输入端D3~D0接好计算得到的预置数据

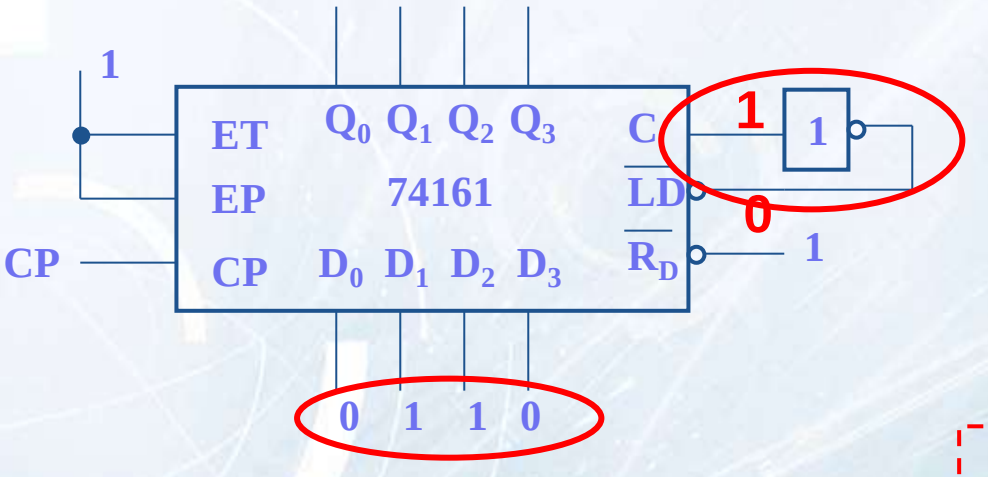
- $\overline{LD}=\overline{C}$ 即将进位输出C经反相后送 $\overline{LD}$ 端
- $(预置数据)_2$=(计数器的模值)-(改变后的模值M)

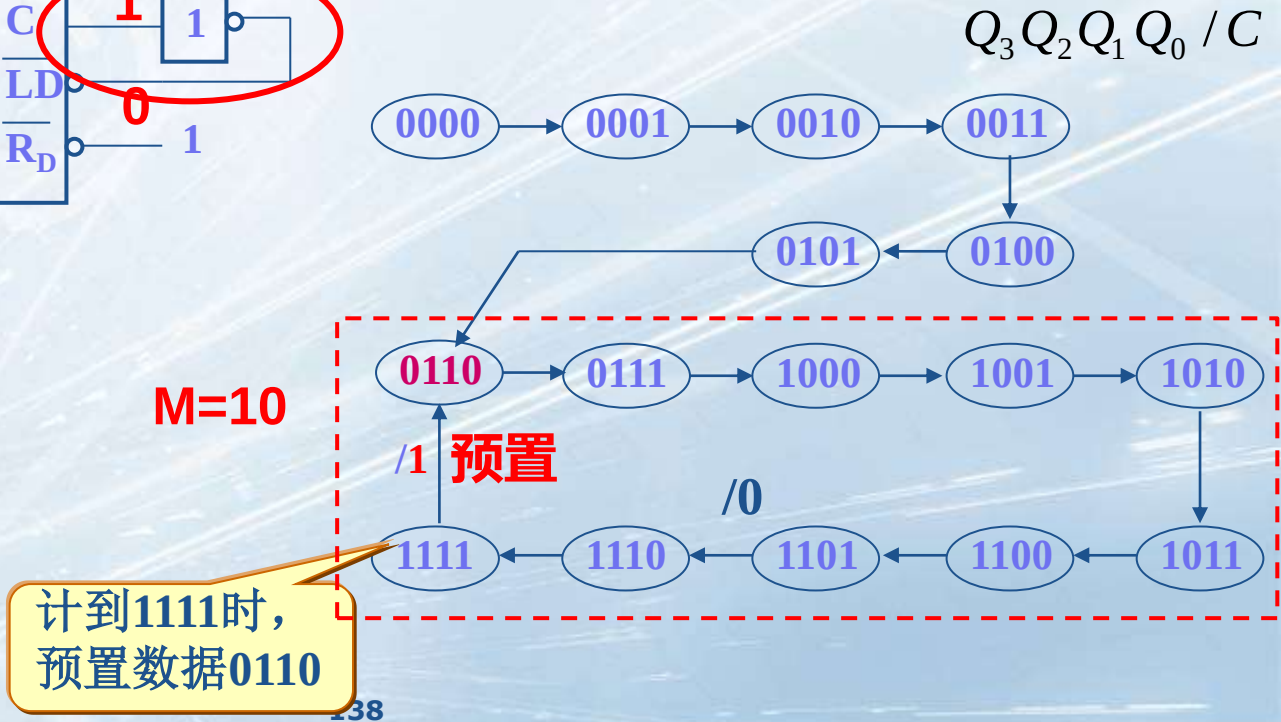
- 当计数器未计到最大值时，C=0，则/LD=1，计数器为计数方式；
- 当计数器计到最大值时，C=1，则/LD=0，计数器为预置方式，CP到来时，则计数器结束本次计数循环，打入预置数据，并开始下一轮循环。

输出C预置法举例

【例7.18】采用输出C预置法用74161实现M=10加法计数器

$(预置数据)_2=(16-10)_{10}=(6)_{10}=(0110)_2=D_3D_2D_1D_0$





（2）输出Q预置法

（2）输出Q预置法

利用计数器的Q输出接至预置端也可以改变计数器的模值

- **步骤：**

① 根据计数器的新模值，求预置代码S_{M-1}，即（计数器新模值-1）的二进制代码；

② 求预置逻辑：当计数器计到（新模值-1）时，将输出为1的FF的Q端信号进行逻辑乘后取反，作为预置信号；

$$\overline{LD}=\overline{\Pi(M-1)^1}$$

③ 画逻辑图（使并行数据输入即预置数据输入信号恒等于0。）

$(预置数据)_2=0000$

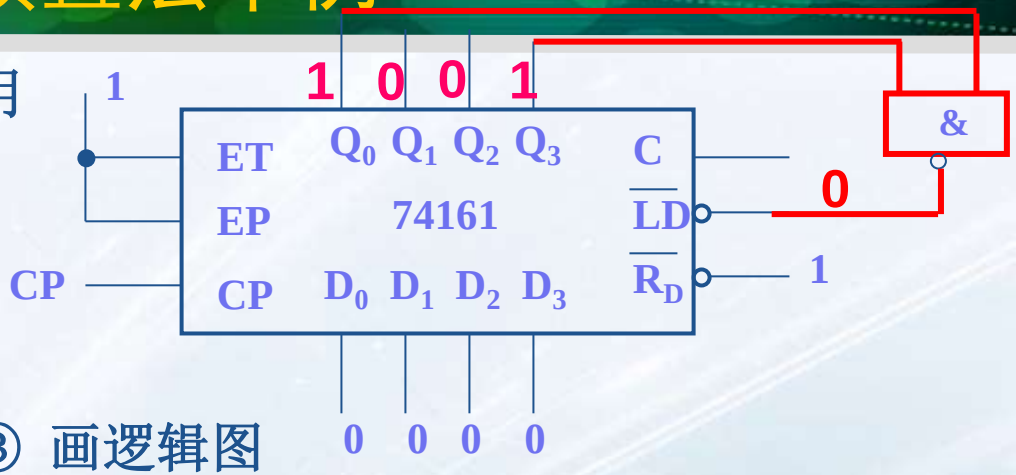
输出Q预置法举例

【例7.19】采用输出Q预置法用74161实现M=10加法计数器

① 求预置代码$S_{M-1} = S_{10-1}=(1001)_2$

② 求预置逻辑

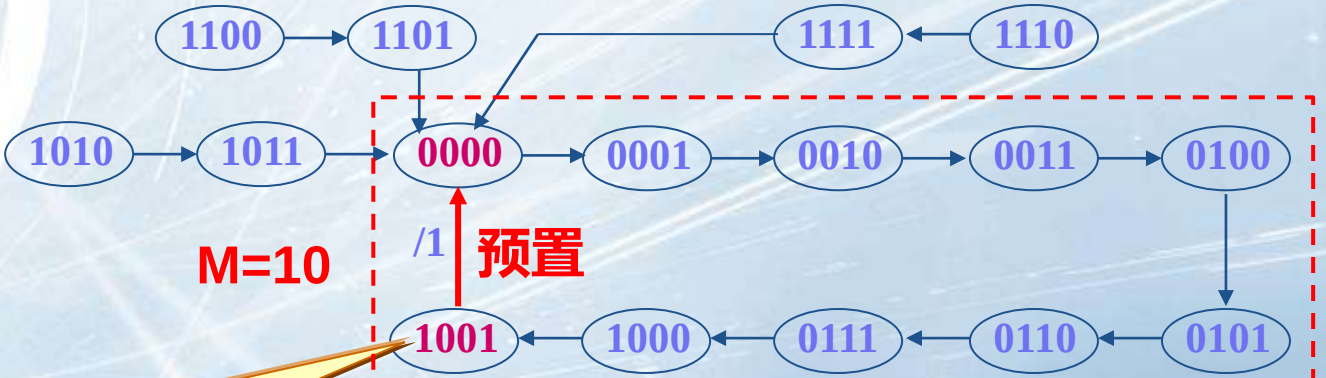
$$\overline{LD} = \overline{\prod Q^1} = \overline{Q_3 Q_0}$$



③ 画逻辑图

$(预置数据)_2=0000 = D_3D_2D_1D_0$

$Q_3Q_2Q_1Q_0 / C$



计到1001时，预置数据0000

3种改变模值方法比较

- 以将4位二进制计数器改变模值为10为例

（1）反馈复位法是使计数器在计到规定的模值（1010）时，产生一个复位信号，使计数器回到0000状态，完成一次计数循环。

（2）输出C预置法是当计数器在计到最大值时，C=1，产生一个预置信号（LD=/C），将计数器预置为一个计算好的预置数据=(计数器原模值)-(改变后的模值M)（0110）；当再来一个时钟脉冲，则计数器结束本次计数循环，并以预置数据作为起始值，开始下一轮计数循环。

（3）输出Q预置法是使计数器在计到（规定的模值-1）时，产生一个预置信号 $\overline{LD}=\overline{\prod Q^1}=\overline{Q_3Q_0}$ ，将计数器预置为0000；当再来一个时钟脉冲，则计数器结束本次计数循环，并以0000作为起始值，开始下一轮计数循环。

内容概要

7.5.1 数码寄存器的设计
7.5.2 移位寄存器的设计
7.5.3 计数器的设计
7.5.4 顺序脉冲发生器的设计
7.5.5 序列信号发生器的设计
7.5.6 序列信号检测器的设计

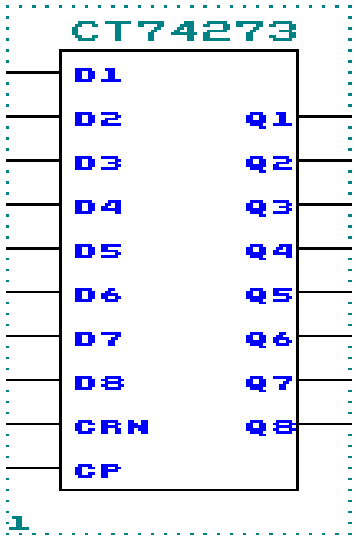
7.5.1 数码寄存器的设计

- 常用的数码寄存器有8D锁存器CT74273和CT74373
  - CT74273是普通的锁存器
  - CT74373是带三态输出的锁存器

1、CT74273的设计

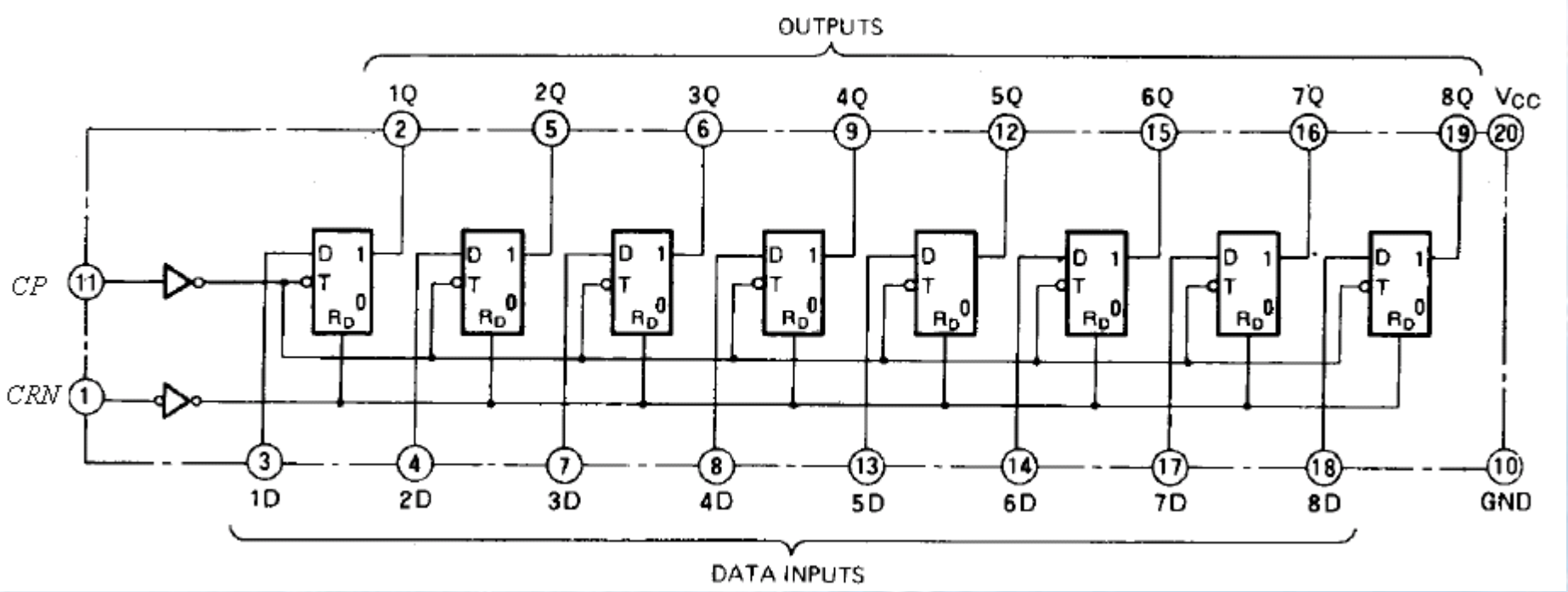
【例7.20】用always块语句设计8D锁存器CT74273

- 并行数据输入端：D8-D1，8位；
- 时钟输入端：CP，上升沿有效；
- 并行数据输出端： Q8-Q1， 8位
- 异步复位控制输入端：CRN，低电平有效，当CRN=0时，锁存器被复位（清零）。



CT74273的元件符号图

CT74273的电路结构图

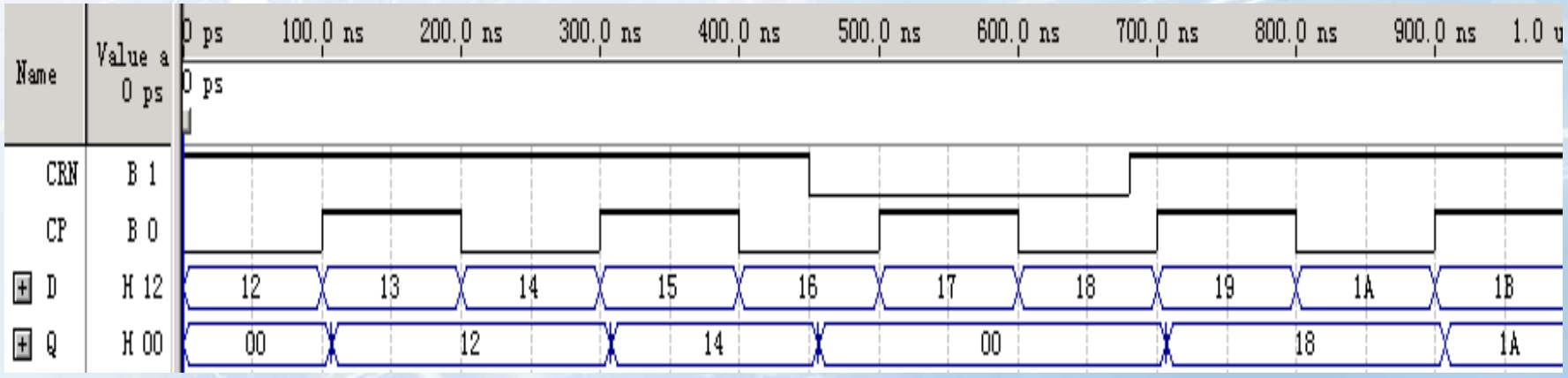


144

CT74273的HDL设计

```
module    CT74273(D1,D2,D3,D4,D5,D6,D7,
                  D8,CRN,CP,Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8);
    input          D1,D2,D3,D4,D5,D6,D7,D8,CRN,CP;
    output         Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8;
    reg            Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8;
    reg[1:8]       Q_TEMP;
    always @(posedge CP or negedge CRN)
       begin
          if (!CRN) Q_TEMP = 8'b00000000;
          else Q_TEMP = {D1,D2,D3,D4,D5,D6,D7,D8};
          {Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8} = Q_TEMP;
       end
endmodule
```

输入输出用总线表示更简洁!

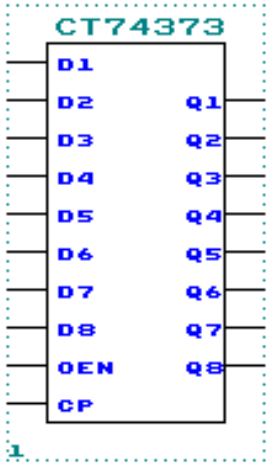


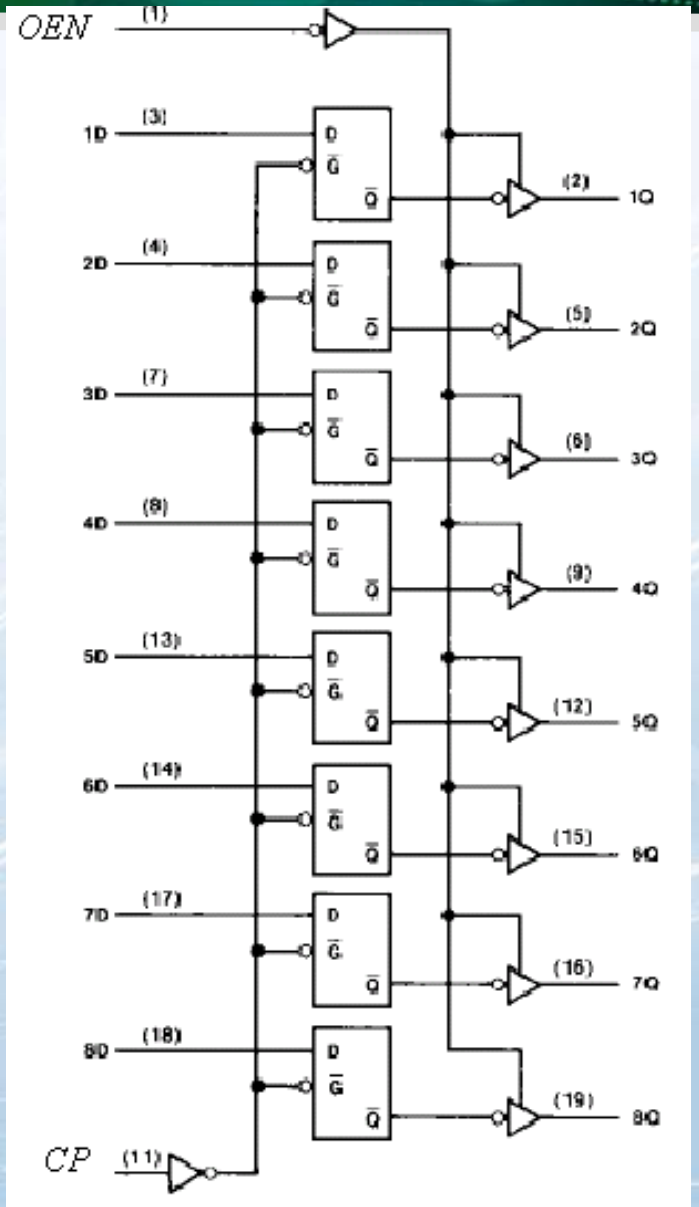
8D锁存器（三态输出）CT74373的设计

2、CT74373的设计

【例7.21】用always块语句设计8D锁存器CT74373

- 并行数据输入端：D8-D1，8位；
- 时钟输入端：CP，上升沿有效；
- 并行数据输出端： Q8-Q1， 8位
- 三态控制输入端：OEN，低电平有效，当OEN=0时，锁存器工作；当OEN=1时，锁存器被禁止，输出为高阻态。





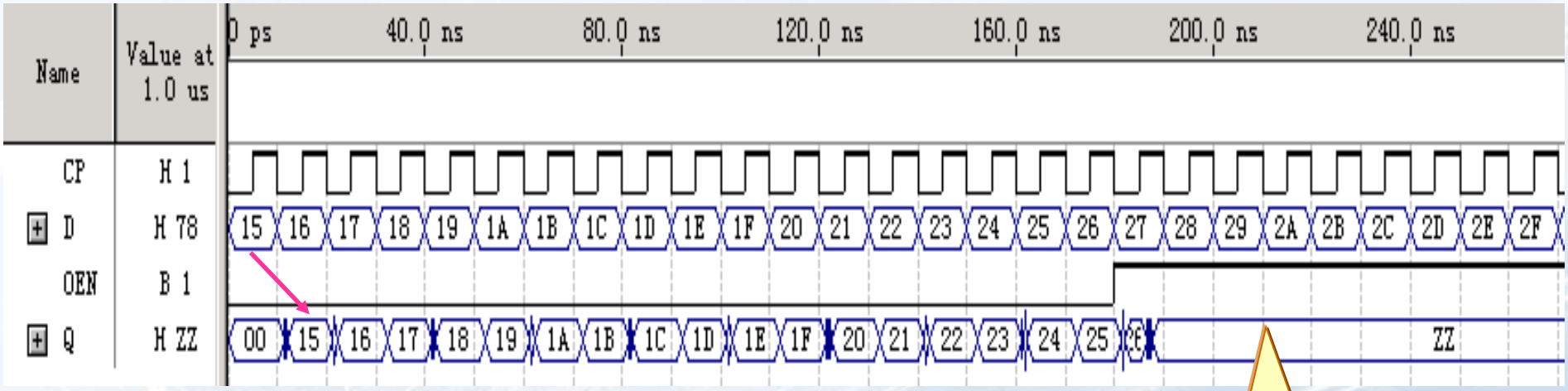
CT74373的HDL设计

```
module CT74373(D1,D2,D3,D4,D5,D6,D7,D8,OEN,CP,
                        Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8);
    input     D1,D2,D3,D4,D5,D6,D7,D8,OEN,CP;
    output    Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8;
    reg       Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8;
    reg[1:8]  Q_TEMP;
    always @(posedge CP)   //数据暂存到Q_TEMP中
        Q_TEMP = {D1,D2,D3,D4,D5,D6,D7,D8};
    always @(OEN)
        begin
          if (!OEN) {Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8} = Q_TEMP;//锁存器工作
          else {Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8} = 8'bzzzzzzzz; ;//锁存器为高阻态
        end
endmodule
```

思考：能否用一个always块表示?

147

CT74373的仿真波形图



OEN=0，
打入数据

OEN=1，
高阻态

148

7.5.2 移位寄存器的设计

【例7.22】用always块语句设计8位双向移位寄存器

功能：同步复位；同步预置；串入左移；串入右移

信号：d[7..0]是预置输入数据信号；clk是时钟信号，上升沿触发；

clr是复位控制输入端，低有效；lod是预置控制输入端，高有效；

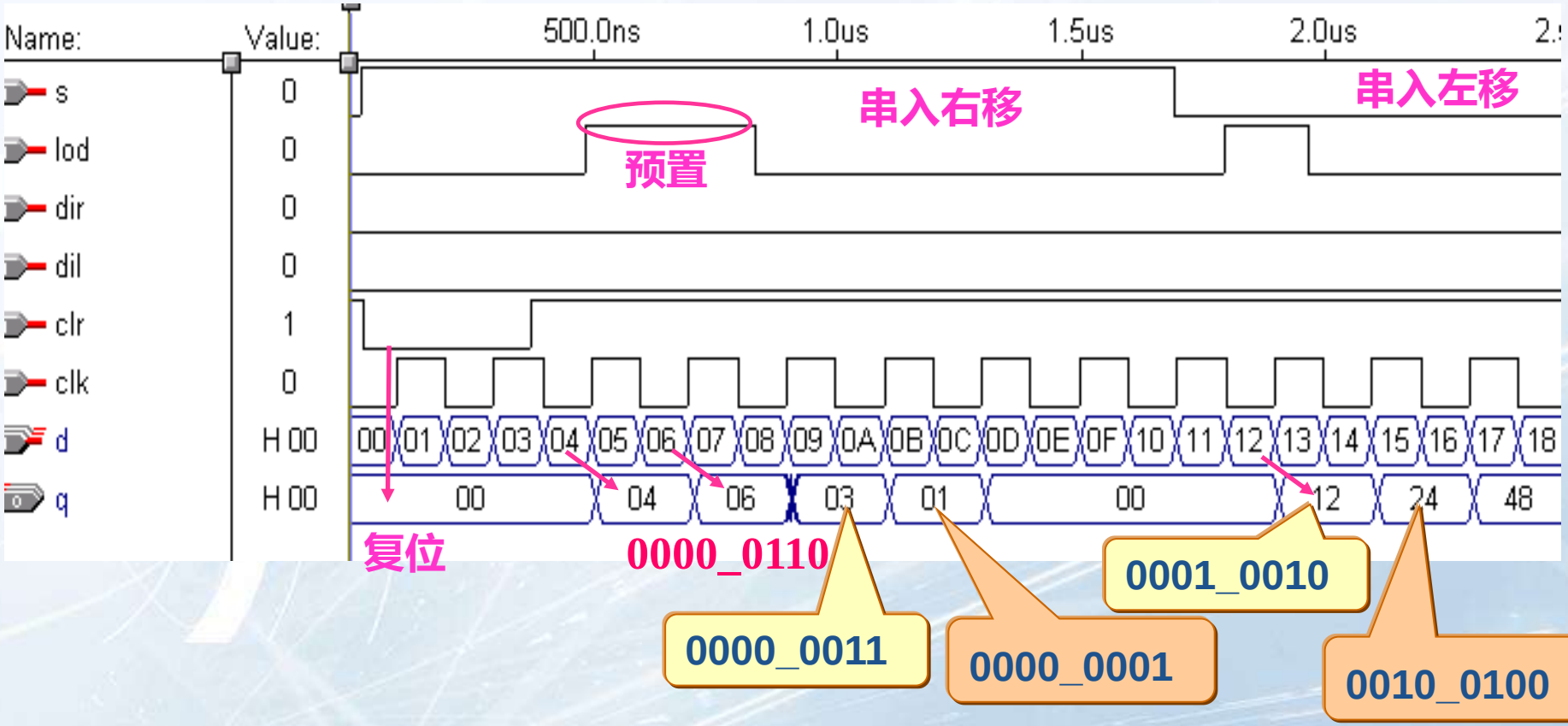
s是移位方向控制输入端，s=1时右移，s=0时左移；

dir是右移串入输入信号；dil是左移串入输入信号。

移位寄存器的HDL设计

```
module rlshift(q,d, clk,clr,lod, s,dir,dil);
    input [7:0]   d;
    input          clk, clr, lod, s, dir, dil;
    output [7:0] q;
    reg  [7:0]     q;
    always @(posedge clk) // 沿触发
      begin
        if (!clr)   q = 8'b00000000;//同步复位
        else if (lod)          q = d;    //同步预置
        else if (s) begin
          q = q >> 1;                    //实现右移操作
          q[7] = dir; end                //寄存器的最高位接收串行右移输入信号
        else  begin
          q = q << 1;                    //实现左移操作
          q[0] = dil; end               //寄存器的最低位接收串行左移输入信号
      end
endmodule
```

8位双向移位寄存器的仿真波形



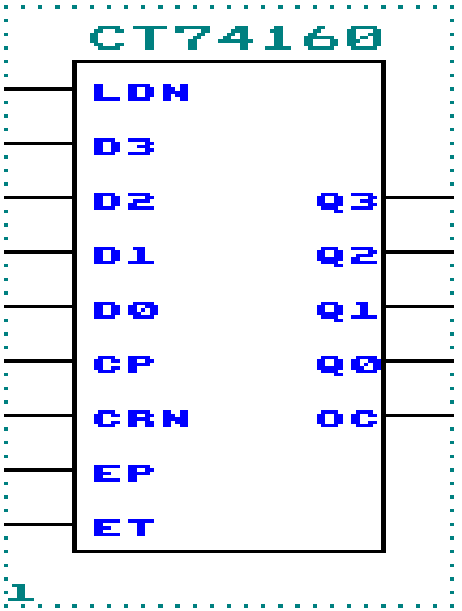
151

7.5.3 计数器的设计

1、十进制同步计数器（异步清除）CT74160

【例7.23】用always块语句设计十进制同步计数器CT74160

- 并行数据输入端：D3-D0
- 时钟输入端：CP，上升沿有效
- 状态输出端： Q3-Q0， Q3、Q2、Q1和Q0的权值依次为2^3、2^2、2^1和2^0
- 异步复位输入端：CRN，低电平有效
- 同步预置控制输入端：LDN，低电平有效
- 使能控制输入端：EP和ET，高电平有效，均为1时计数器工作，只要有1个信号为0，则计数器保持
- 进位输出端: OC, Q3Q2Q1Q0=1001且ET=1时，OC=1



功能：异步复位；同步预置；计数；保持

CT74160的HDL设计

```
module CT74160(LDN,D3,D2,D1,D0,CP,CRN,EP,ET,Q3,Q2,Q1,Q0,OC);
    input LDN,D3,D2,D1,D0,CP,CRN,EP,ET;
    output  Q3,Q2,Q1,Q0,OC;
    reg  Q3,Q2,Q1,Q0,OC;
    reg[3:0]  Q_TEMP;
  always@(posedge CP or negedge CRN)
      begin
          if (!CRN) Q_TEMP = 4'b0000;                    //异步复位
          else
            begin
              if (!LDN) Q_TEMP = {D3,D2,D1,D0};          //同步预置
              else if (EP && ET)                         //计数
                 if (Q_TEMP < 4'b1001) Q_TEMP = Q_TEMP + 1;
                 else  Q_TEMP = 4'b0000;
              else Q_TEMP = Q_TEMP; //保持
            end
       end
```

异步复位

最大只计到9

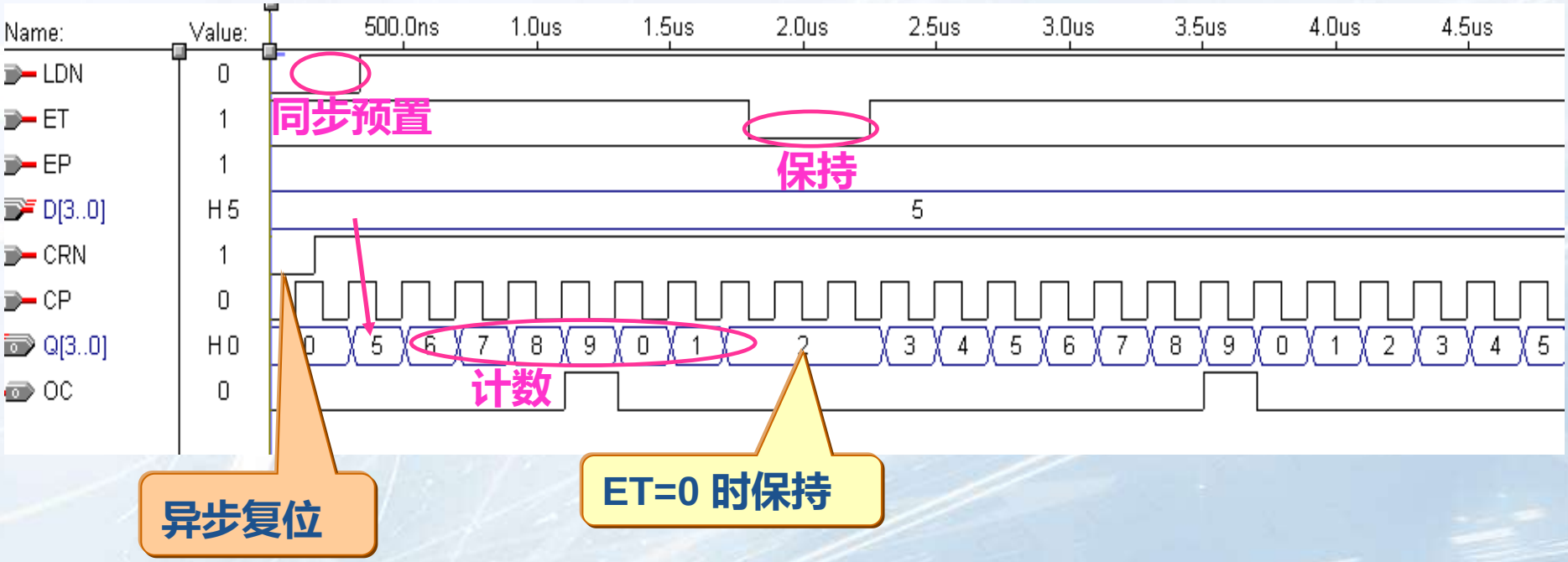
CT74160的HDL设计（续）

组合逻辑

```
always                                    //产生进位输出和对最终输出赋值
    begin
        if (Q_TEMP == 4'b1001 && ET == 1'b1) OC = 1'b1;
        else         OC = 1'b0;
        {Q3,Q2,Q1,Q0} = Q_TEMP;
    end
endmodule
```

返回74161的设计

CT74160的仿真波形图

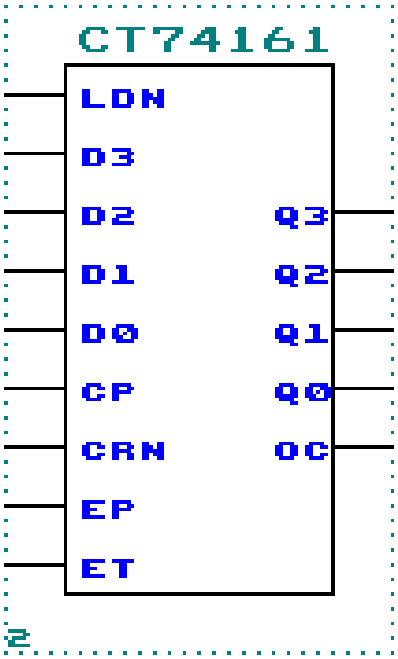


【例7.24】二进制同步计数器CT74161(异步清除)

2、4位二进制同步计数器（异步清除）CT74161

【例7.24】用always块语句设计二进制同步计数器CT74161

- 并行数据输入端：D3-D0
- 时钟输入端：CP，上升沿有效
- 状态输出端： Q3-Q0， Q3、Q2、Q1和Q0的权值依次为2^3、2^2、2^1和2^0
- 异步复位输入端：CRN，低电平有效
- 同步预置控制输入端：LDN ，低电平有效
- 使能控制输入端：EP和ET，高电平有效，均为1时计数器工作，只要有1个信号为0，则计数器保持
- 进位输出端：OC, Q3Q2Q1Q0=1111且ET=1时，OC=1



CT74161的HDL设计

```
module  CT74161(LDN,D3,D2,D1,D0,CP,CRN,EP,ET,Q3,Q2,Q1,Q0,OC);
    input       LDN,D3,D2,D1,D0,CP,CRN,EP,ET;
    output     Q3,Q2,Q1,Q0,OC;
    reg          Q3,Q2,Q1,Q0,OC;
    reg[3:0]   Q_TEMP;
    always@(posedge CP or negedge CRN )
      begin
        if (!CRN) Q_TEMP = 4'b0000;                    //异步复位
        else
          begin
            if (!LDN) Q_TEMP = {D3,D2,D1,D0}; //同步预置
            else if (EP && ET) Q_TEMP = Q_TEMP + 1; //计数
            else Q_TEMP = Q_TEMP; //保持
          end
      end
```

与CT74160的区别:不必判断计数最大值为多少，每来一个时钟脉冲，都加1计数

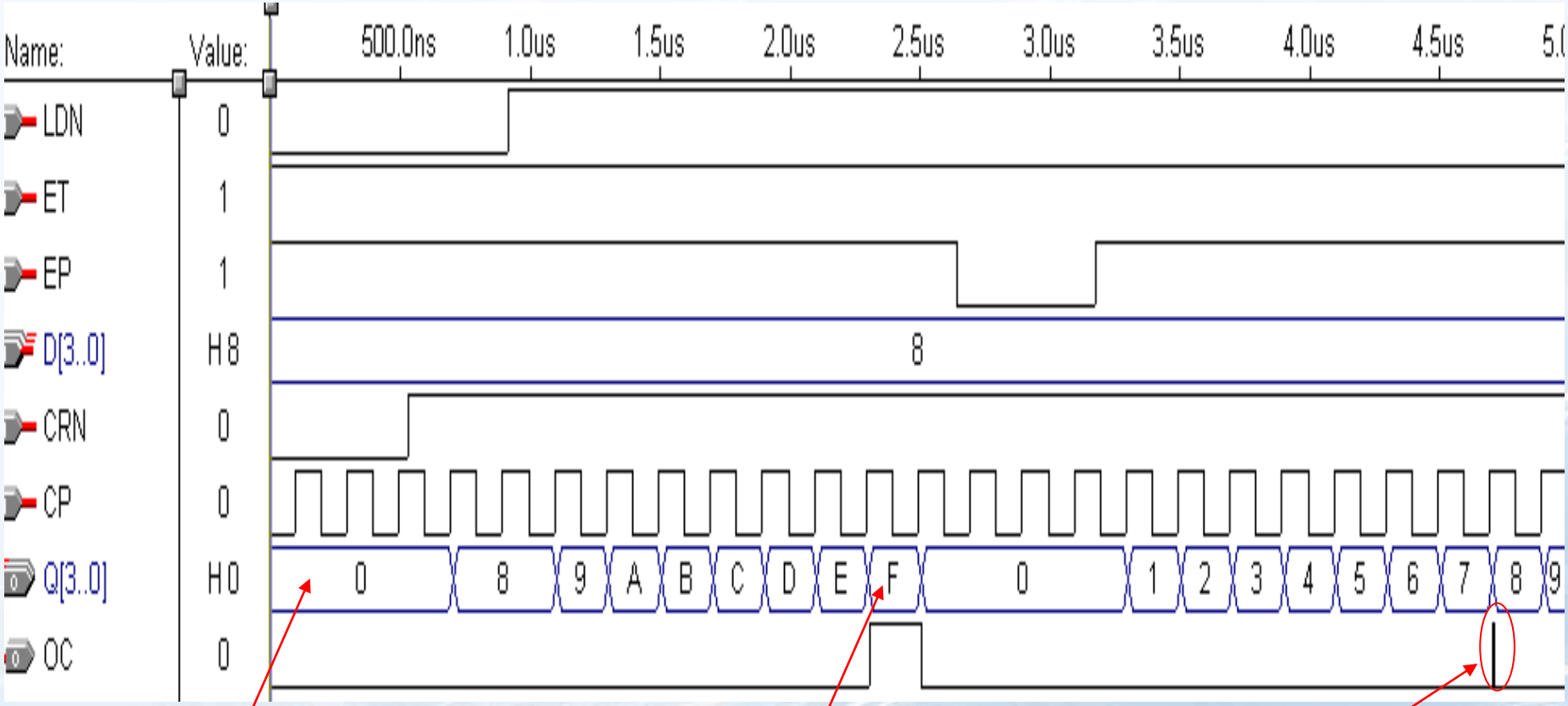
CT74161的HDL设计（续）

```
always//产生进位输出和对最终输出赋值
        begin
            if (Q_TEMP == 4'b1111 && ET == 1'b1) OC = 1'b1;
            else       OC = 1'b0;
           {Q3,Q2,Q1,Q0} = Q_TEMP;
        end
endmodule
```

与CT74160的区别

CT74160源程序

CT74161的仿真波形图



异步复位

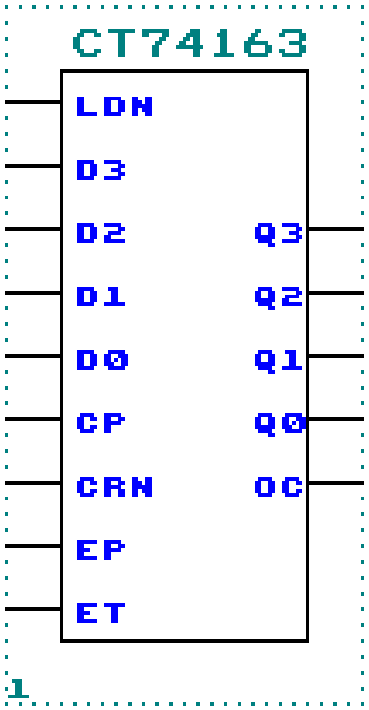
最大计到 $(1111)_2$

毛刺

3、4位二进制同步计数器CT74163

【例7.25】用always块语句设计二进制同步计数器CT74163 （同步清除）

- 并行数据输入端：D3-D0；
- 时钟输入端：CP，上升沿有效；
- 状态输出端： Q3-Q0， Q3、Q2、 Q1和Q0的权值依次为2^3、2^2、2^1和2^0
- 同步复位输入端：CRN，低电平有效
- 同步预置控制输入端：LDN ，低电平有效
- 使能控制输入端：EP和ET，高电平有效，均为1时计数器工作，只要有1个信号为0，则计数器保持
- 进位输出端：OC, Q3Q2Q1Q0=1111且ET=1时，OC=1

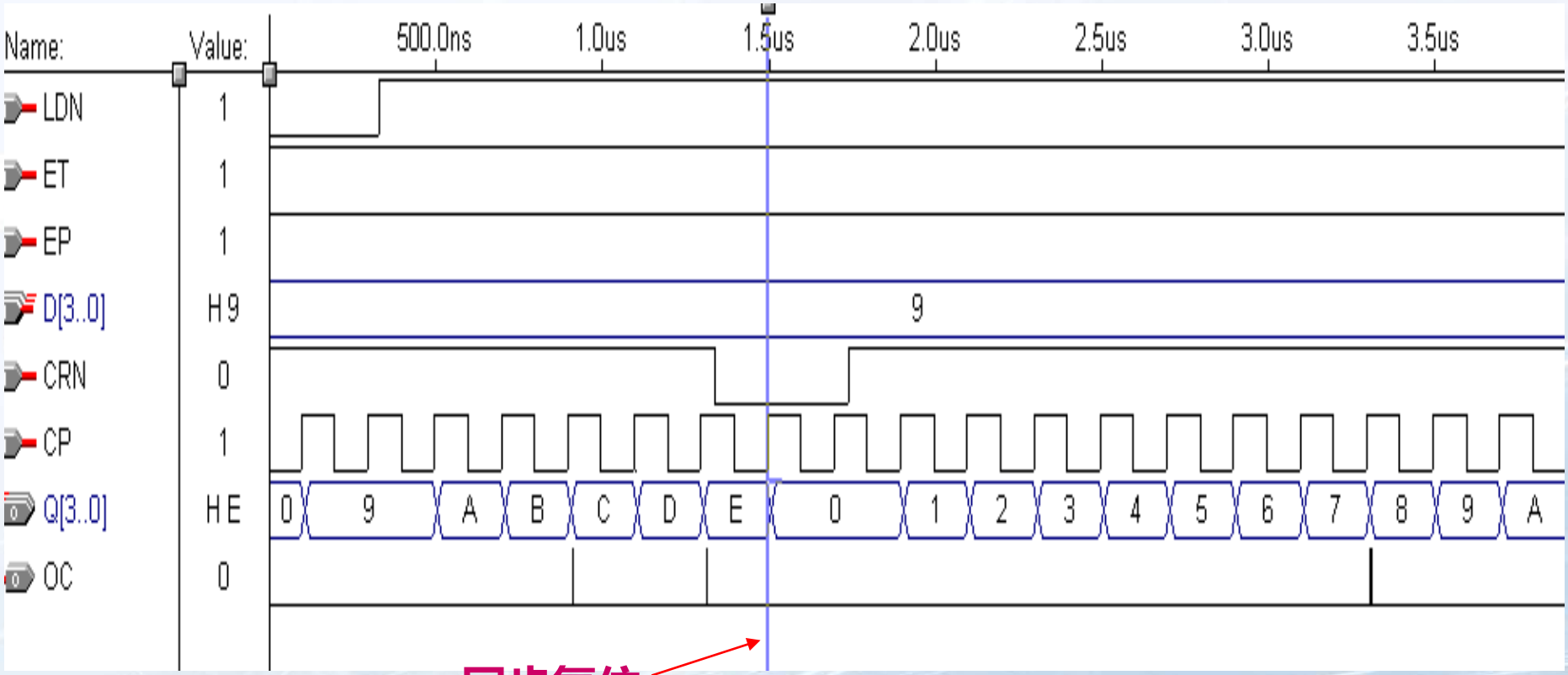


CT74163的HDL设计

```
module  CT74163(LDN,D3,D2,D1,D0,CP,CRN,EP,ET,Q3,Q2,Q1,Q0,OC);
    input          LDN,D3,D2,D1,D0,CP,CRN,EP,ET;
    output         Q3,Q2,Q1,Q0,OC;
    reg            Q3,Q2,Q1,Q0,OC;
    reg[3:0]       Q_TEMP;
    always         @(posedge CP )
       begin
          if (!CRN) Q_TEMP = 4‘b0000;                    //同步复位
          else if (!LDN) Q_TEMP = {D3,D2,D1,D0};         //同步预置
          else if (EP && ET) Q_TEMP = Q_TEMP + 1;        //计数
          else Q_TEMP = Q_TEMP;                          //保持
       end
    always
       begin
          if (Q_TEMP == 4'b1111 && ET == 1'b1) OC = 1'b1;
          else       OC = 1'b0;
          {Q3,Q2,Q1,Q0} = Q_TEMP;
       end
endmodule
```

？程序与CT74161有何区别？

CT74163的设计的仿真波形图

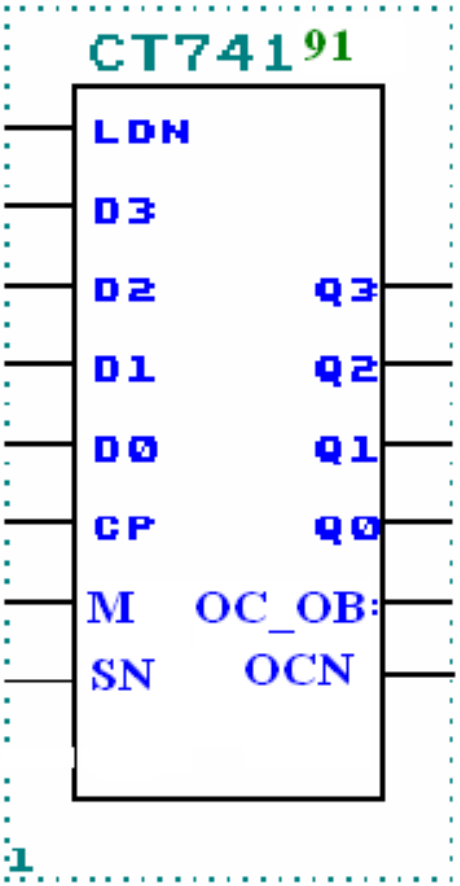


同步复位

4、同步加/减计数器CT74191

【例7.26】4位二进制同步加/减计数器CT74191

- 并行数据输入端：D3-D0
- 时钟输入端：CP，上升沿有效
- 状态输出端： Q3-Q0， Q3、Q2、 Q1和Q0的权值依次为2^3、2^2、2^1和2^0
- 加/减控制输入端：M，M=0时，计数器加计数；M=1时，计数器减计数
- 使能控制输入端：SN，低电平有效
- 同步预置控制输入端：LDN，低电平有效
- 进位/借位输出端：OC_OB，加法计数时，当Q3Q2Q1Q0=1111时，OC_OB=1；减法计数时，当Q3Q2Q1Q0=0000时，OC_OB=1；
- OC_OB的反相输出端: OCN，输出脉冲宽度为半个时钟周期（低电平段）



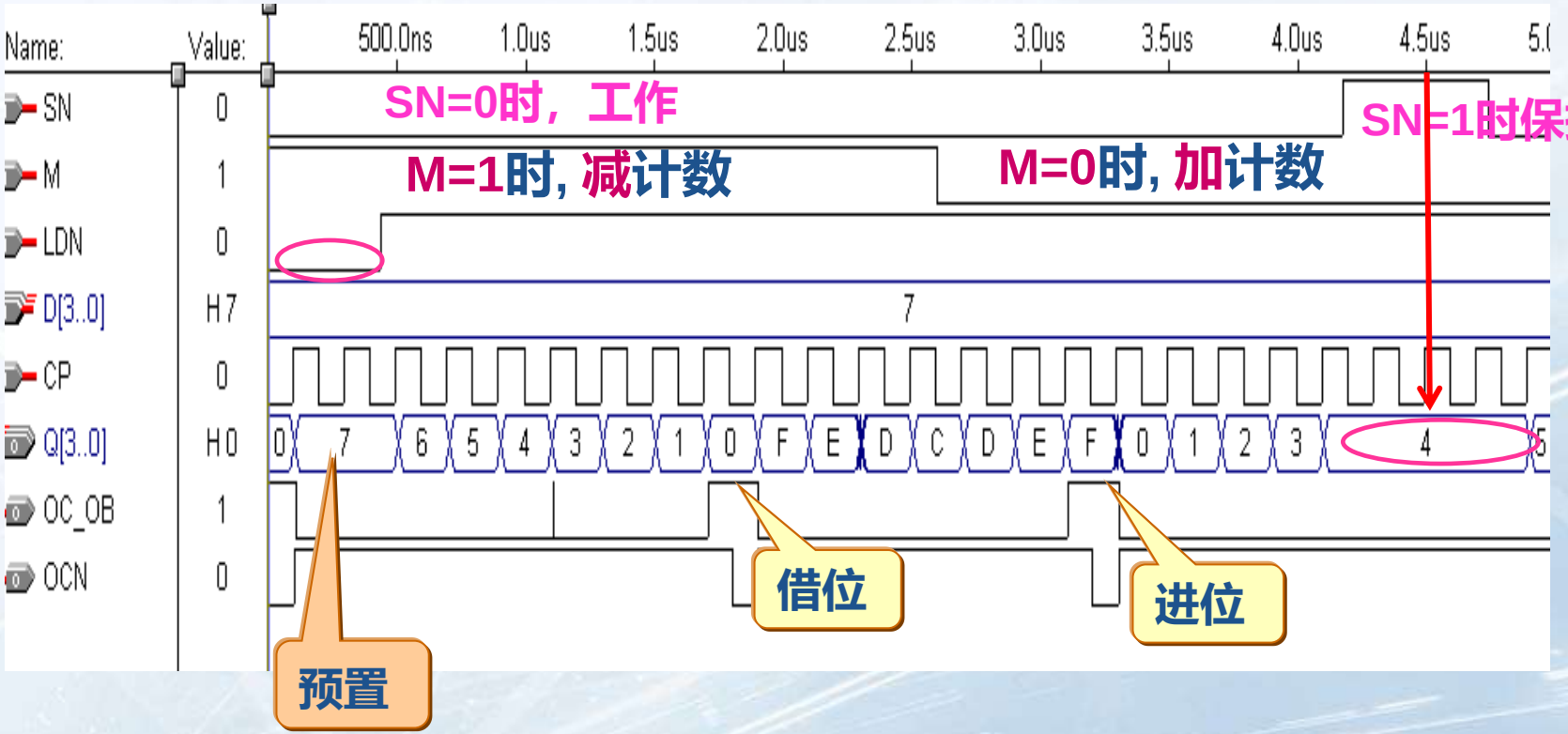
CT74191的HDL设计

```
module CT74191(LDN,D3,D2,D1,D0,CP,M,SN,
                    Q3,Q2,Q1,Q0,OC_OB,OCN);
    input      LDN,D3,D2,D1,D0,CP,M,SN;
    output     Q3,Q2,Q1,Q0,OC_OB,OCN;
    reg        Q3,Q2,Q1,Q0,OC_OB,OCN;
    reg[3:0]   Q_TEMP;
    always @(posedge CP ) // (1) 处理加/减操作
       begin
          if (!LDN) Q_TEMP = {D3,D2,D1,D0};    //同步预置
          else if (!SN)                         //计数器工作
             if (!M)  Q_TEMP = Q_TEMP + 1;  //若M为0，做加法
             else Q_TEMP = Q_TEMP - 1;      //若M为1，做减法
          else Q_TEMP = Q_TEMP;                 //计数器保持
       end
```

CT74191的HDL设计（续）

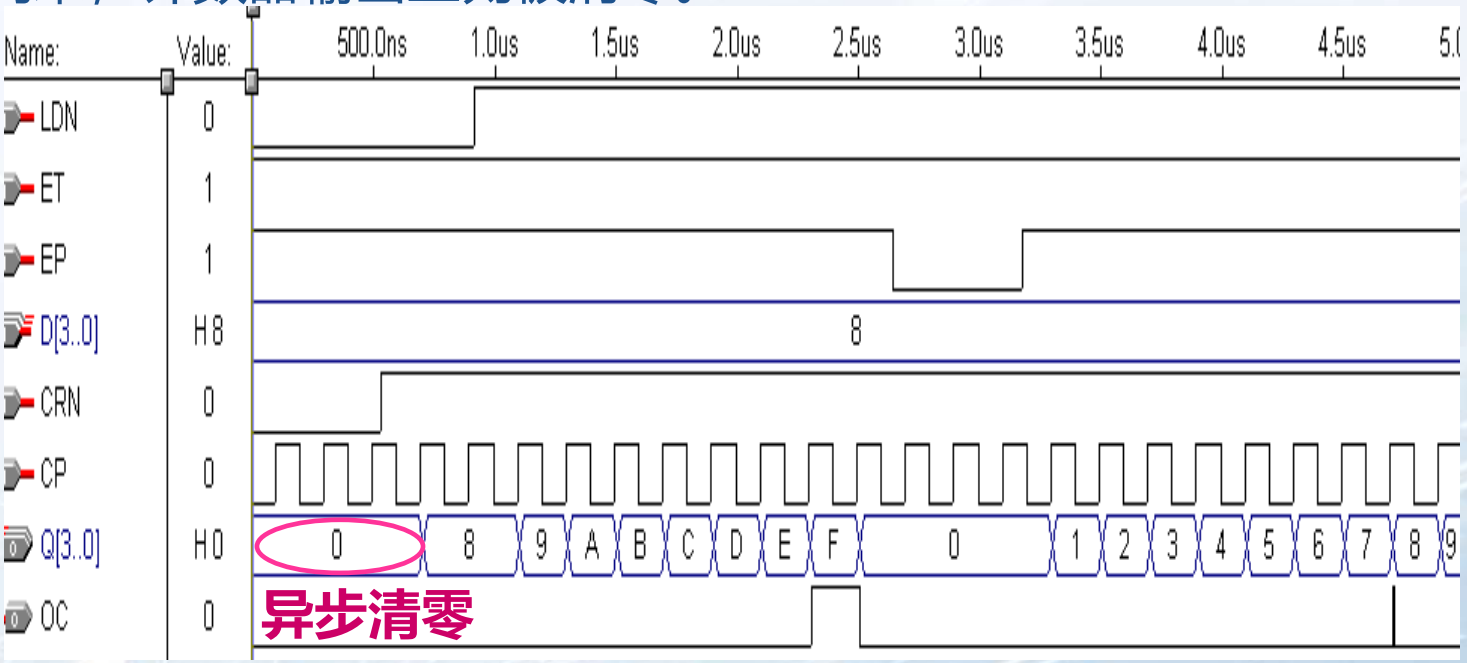
```
always //（2）处理进位/借位操作
    begin
        if (Q_TEMP == 4'b1111 && M == 1'b0)
          begin
            OC_OB = 1‘b1;                    //产生进位
            OCN = (OC_OB && CP);
          end
        else if (Q_TEMP == 4'b0000 && M == 1'b1)
          begin
            OC_OB = 1‘b1;                    //产生借位
            OCN = (OC_OB && CP);
          end
        else  OC_OB = 1'b0;
        {Q3,Q2,Q1,Q0} = Q_TEMP;
    end
endmodule
```

CT74191的仿真波形图



计数器同步清零与异步清零

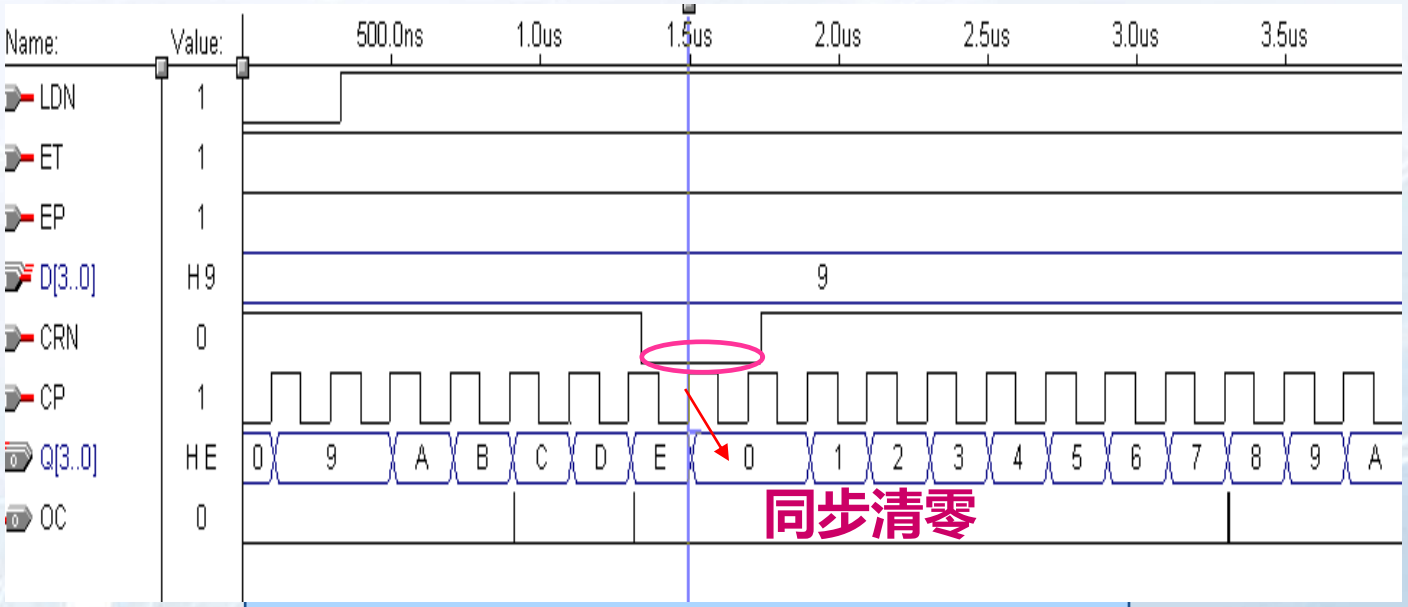
◆通常采用异步清零——只要清零信号有效，则无论有无时钟脉冲到来，计数器输出立刻被清零。




```
always @ (posedge CP or negedge CRN)
  begin
    if (!CRN)                  //异步清零
      Q<=4'b0000;
      ……
  end
```

计数器同步清零与异步清零（续）

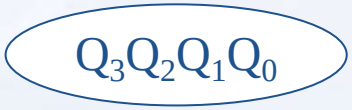
- 只有在时钟周期很小或清零信号为电平信号时（容易捕捉到清零信号）采用**同步**清零——当清零信号有效时，若来一个时钟脉冲，则计数器输出被清零。

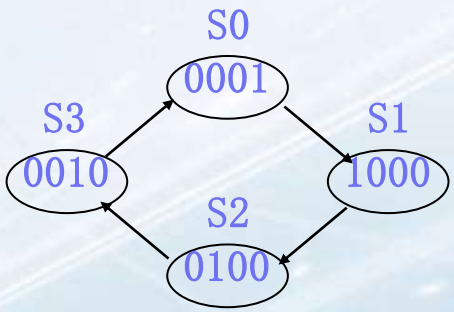



```
always @ (posedge CP)
  begin
    if (!CRN)                  //同步清零
      Q<=4'b0000;
      ……
  end
```

7.5.4 顺序脉冲发生器的设计

- 在时钟脉冲的控制下，各触发器输出端顺序产生脉冲信号的电路称为**顺序脉冲发生器**。
- 顺序脉冲发生器由多个触发器构成
- 4位顺序脉冲发生器，由4个状态构成，每个状态编码中“1”的个数都是1个，表示每个时钟周期内只有一个触发器的输出端为高电平（脉冲），而且是轮流出现，因而生成顺序脉冲信号。





4位顺序脉冲发生器的状态图

【例7.27】设计一个4位顺序脉冲发生器，使在每个时钟周期内只有一个触发器的输出端为高电平（脉冲），而且是轮流出现。

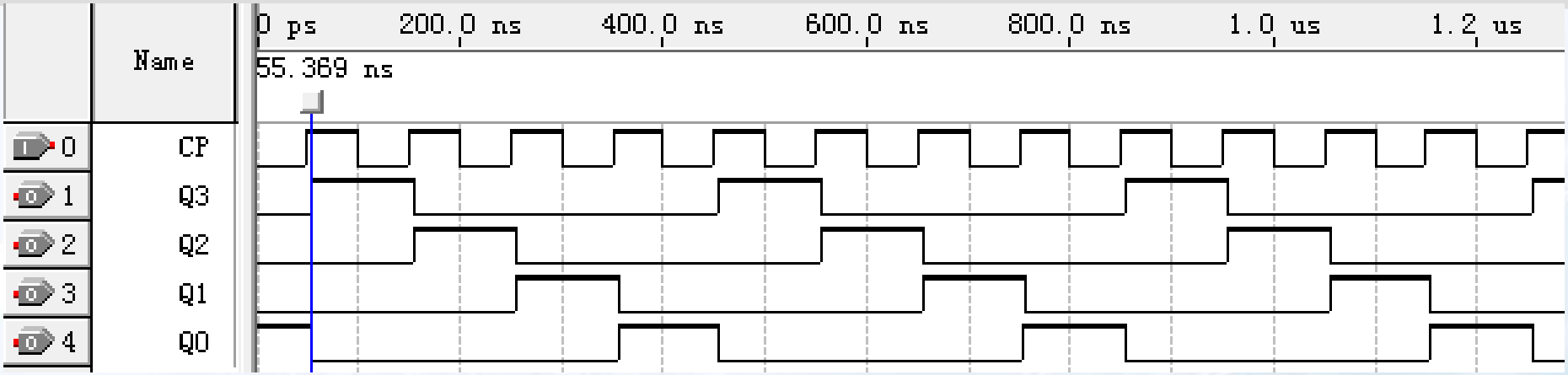
顺序脉冲发生器的HDL设计

```
module method4(CP,Q3,Q2,Q1,Q0);
    parameter   S0 = 'b0001,S1 = 'b1000,S2 = 'b0100,S3 = 'b0010;
    input       CP;
    output      Q3,Q2,Q1,Q0;
    reg         Q3,Q2,Q1,Q0;
    reg[3:0]    SS;              //状态机
    always @(posedge CP ) //状态的转移
       begin
          if (SS == S0) SS = S1;
          else if (SS == S1) SS = S2;
          else if (SS == S2) SS = S3;
          else if (SS == S3) SS = S0;
          else SS= S0;    //其他状态下返回初始状态
          {Q3,Q2,Q1,Q0} = SS;    //状态机的输出
       end
endmodule
```

状态编码定义

? 用case语句如何改写?

顺序脉冲发生器的仿真波形图



电路的Q3、Q2、Q1、Q0 输出端顺序产生脉冲信号，4个时钟周期后，又重复此规律

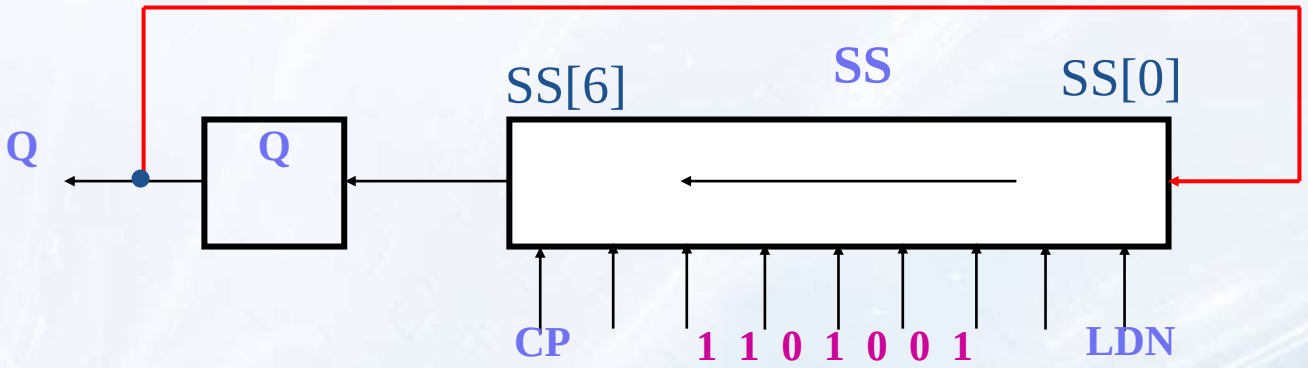
- 电话铃流控制是顺序脉冲发生器应用的一个实例
- 如果用4位顺序脉冲发生器的某一个输出作为铃流控制信号，且时钟CP周期为1秒，那么电话铃声就会以响1秒停3秒的节奏进行。

7.5.5 序列信号发生器的设计

- 序列信号是一系列重复产生的一组相同数码或符号的信号，如海难救助信号“SOS”。
- 在数字电路中，序列信号由一组二进制代码组成，代码的位数是序列信号的长度。例如，“1101001”就是一个7位的序列信号。

【例7.28】利用移位寄存器设计一个7位序列信号发生器，使在预置控制信号有效时，将序列信号（如1101001）打入寄存器中；在时钟脉冲到来后，依次输出序列信号。

序列信号发生器的电路结构图



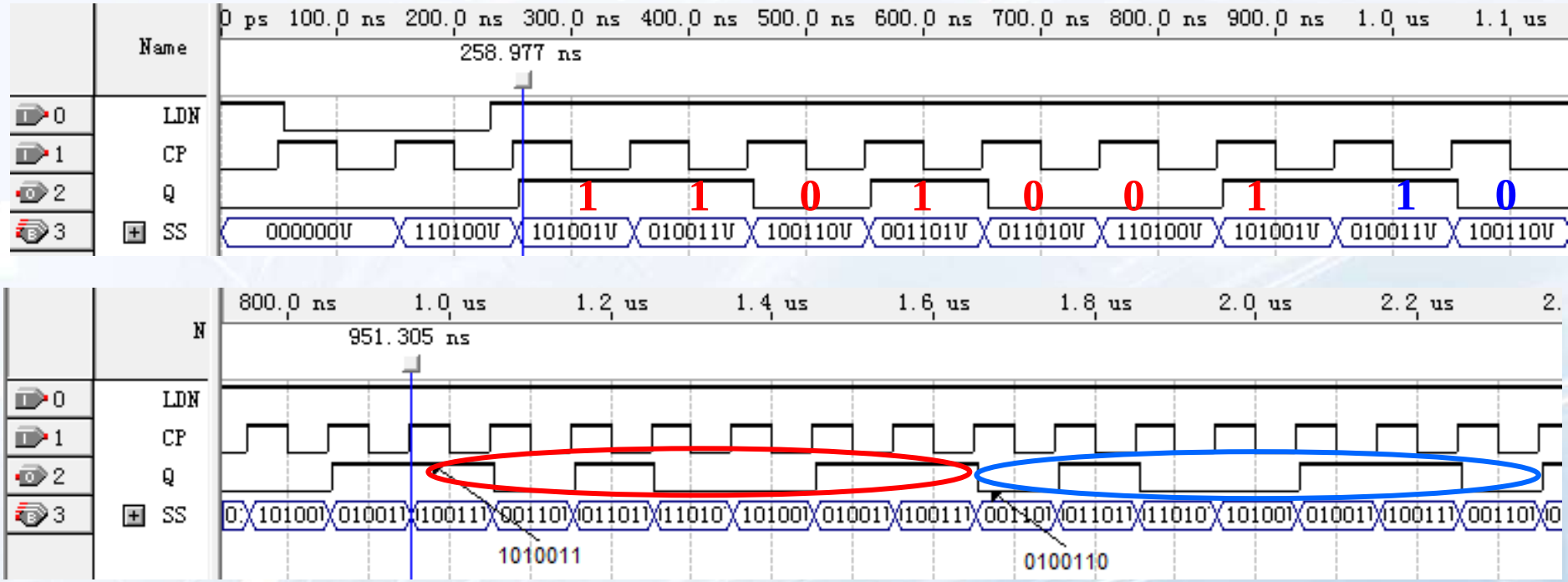
- CP：时钟输入端，上升沿有效
- LDN：预置控制输入端，低电平有效，LDN=0时，将序列信号（如1101001）锁存于SS中；
- SS：7位左移移位寄存器，SS[6]是最高位，SS[0]是最低位；当一个时钟脉冲到来后，把最高位SS[6] 送到输出Q，SS中的数据依次向左移1位，移位前的SS[6]送到最低位SS[0]
- Q：串行输出端，在CP的控制下，输出序列信号(SS[6]最先输出）

序列信号发生器的HDL设计（有错）

```
module signal7_error(CP,LDN,Q);
    parameter   sign =7'b1101001;//定义序列信号
    input        CP,LDN;
    output       Q;
    reg          Q;
    reg[6:0]     SS;
    always @ (posedge CP)
       begin
          if (!LDN) SS = sign;          //将序列信号打入移位寄存器中
          else
            begin
              Q = SS[6];                //从最高位输出序列信号
              SS[0] = SS[6];            //最高位同时送给最低位
              SS = SS << 1;             //然后数据左移
            end
       end
endmodule
```

SS[6] 会将SS[0]覆盖，原始序列被改变！

序列信号发生器的仿真波形图（不正确）



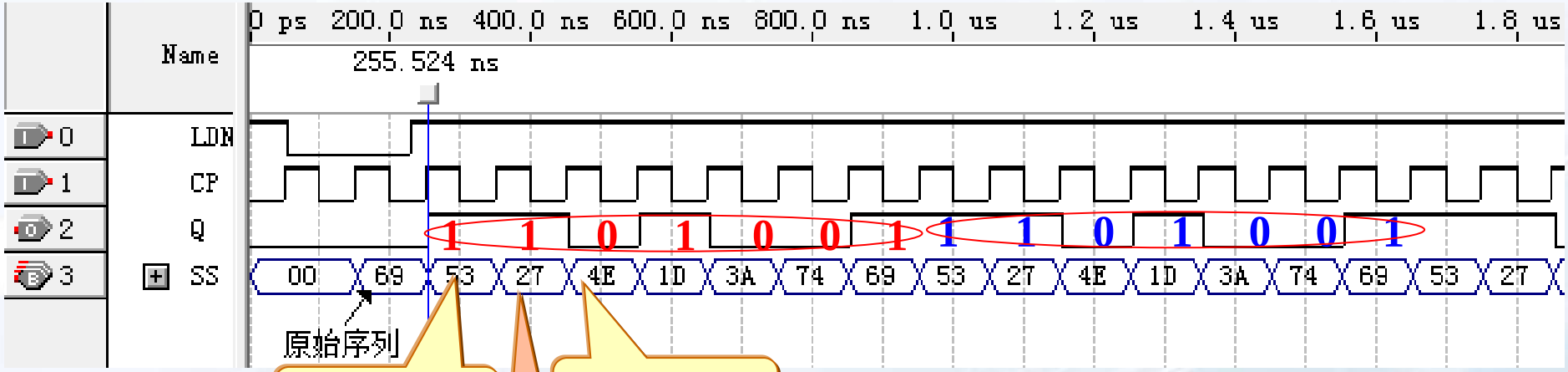
- 由于先将最高位送给最低位，再移位，则SS[6] 会将SS[0]覆盖，SS[0]为不定值，原始序列被改变！ 仿真不正确！
- 7个CP后，第1个输出序列似乎正确：1101011，但第2个输出序列变为：1010011；第3个输出序列变为：0100110……没有规律，不是重复序列！

序列信号发生器的改进设计

```
module signal7 (CP,LDN,Q);
    parameter   sign =7'b1101001;//定义序列信号
    input          CP,LDN;
    output         Q;
    reg            Q;
    reg[6:0]       SS;
    always @ (posedge CP)
       begin
          if (!LDN) SS = sign;          //将序列信号打入移位寄存器中
          else
             begin
                Q = SS[6];             //从最高位输出序列信号
                SS = SS << 1;          //数据左移
                SS[0] = Q;             //左移前的最高位同时送给最低位
             end
       end
endmodule
```

两条语句调换顺序，先左移，再用移位之前的SS[6] 填补SS[0]

改进的序列信号发生器的仿真波形图



101_0011

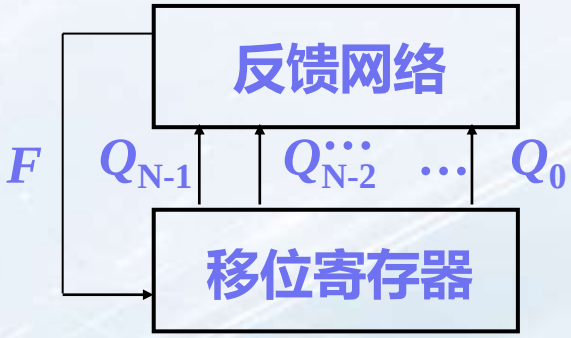
010_0110

010_0111

每当CP上升沿到来时，SS左移一位；同时将前一时刻SS的最高位补充到此时刻SS的最低位，从而保证重复产生序列信号，仿真正确！

【例7.29】伪随机码发生器的设计

- 随机码是一种没有规律（或叫做杂乱无章）的二进制代码。
- 如果序列信号发生器的序列长度超过2^{15}-1，其变化规律与随机码类似，则把它称为伪随机序列信号（伪随机码）。
- 伪随机序列信号可作为数字通信中的一个信号源，通过信道发送到接收机方，用于检测数字通信系统错码的概率，即误码率。



移存型计数器结构图

- 过去，伪随机序列信号发生器采用移存型计数器实现
- 移存型计数器包括一个 n位移位寄存器和一个反馈网络（输入：取自Q_{N-1}~ Q_0，输出：反馈至移位寄存器的串行输入端）
- 反馈网络不同，形成的移存型计数器则不同；若反馈网络由异或门构成，则形成最长线性序列移存型计数器。其反馈函数见教材P172表6.12

伪随机码发生器的设计思路

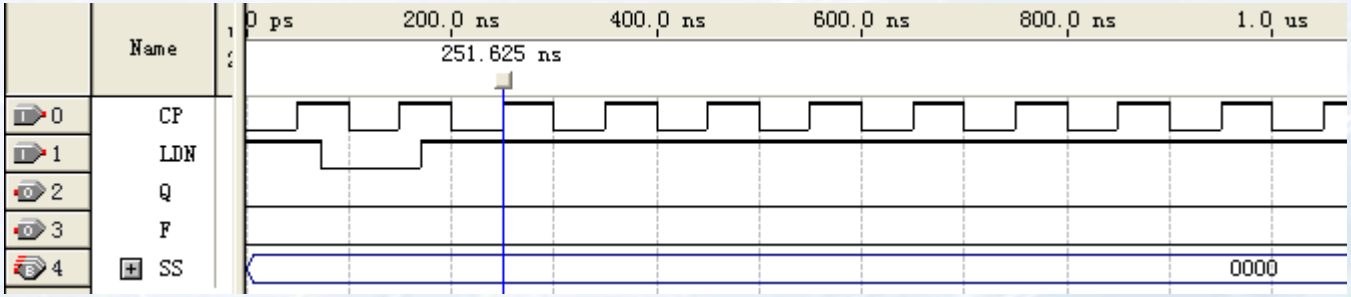
N=15时，反馈函数 $F = Q_1 \oplus Q_0$

- 时钟输入端：CP，上升沿有效
- SS：15位右移移位寄存器，SS[14]是最高位，SS[0]是最低位
- 中间变量F：反馈函数，为SS[1]和SS[0]的异或
- 中间变量Q：串行输出端，在CP的控制下，输出序列信号， Q =SS[14]
- 预置控制输入端：LDN，低电平有效，LDN=0时，将初始序列信号（如1000……000）打入SS中

- 当CP上升沿到来时，SS中的数据依次向右移1位，同时SS[14] 接收F的值，并从Q输出序列信号

修改反馈函数

- 但在此伪随机码发生器中，有一个由15个“0”构成的死循环
- 即当初始序列为000_0000_0000_0000时， SS状态始终全部为0，F、Q始终全部为0，不可能产生随机序列



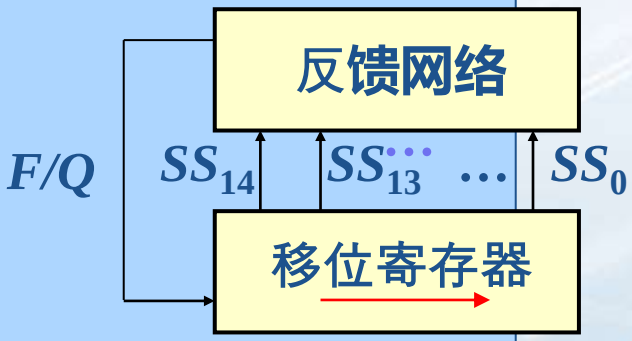
- 为了打破死循环，修改反馈函数为：

$$F = Q_1 \oplus Q_0 + \overline{Q}_{14}\overline{Q}_{13}\overline{Q}_{12}\overline{Q}_{11}\overline{Q}_{10}\overline{Q}_9\overline{Q}_8\overline{Q}_7\overline{Q}_6\overline{Q}_5\overline{Q}_4\overline{Q}_3\overline{Q}_2\overline{Q}_1\overline{Q}_0$$

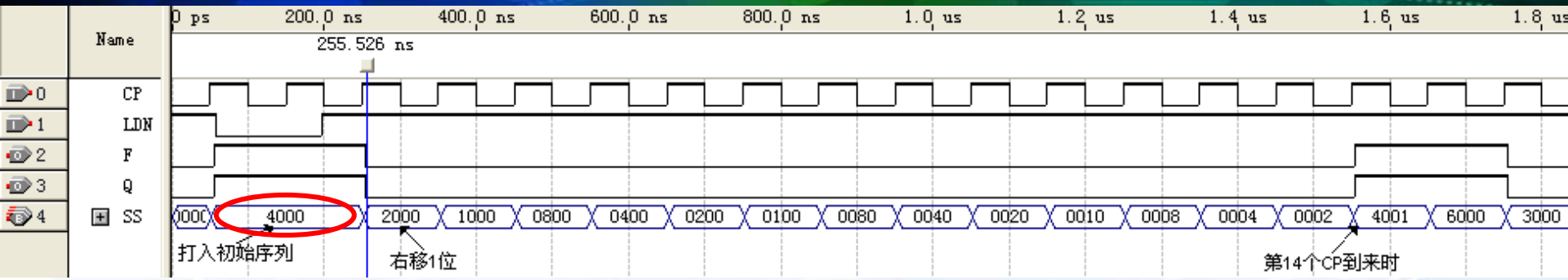
上式说明，当Q14～Q0=00000000000000时，反馈函数$F=1$，打破了原反馈函数$F=0$出现死循环的状态。

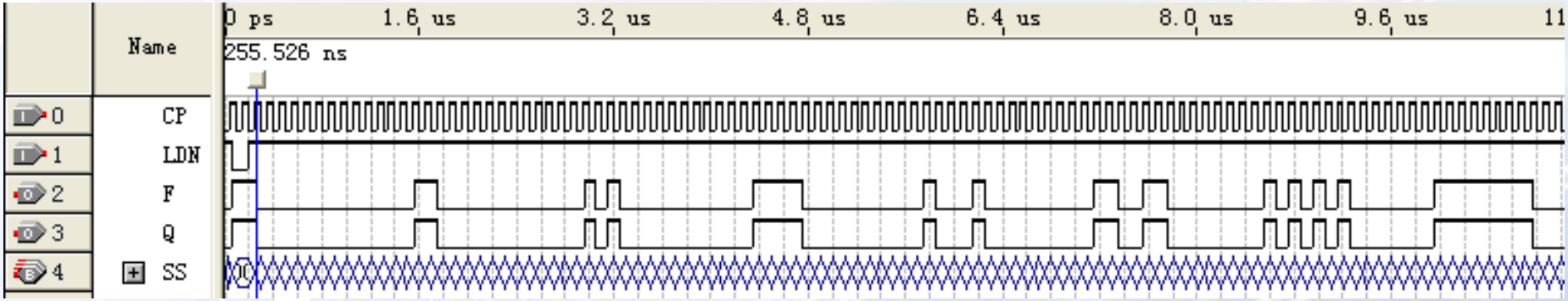
修改反馈函数后伪随机码发生器的HDL设计

```
module signal15(CP,LDN,Q);
    parameter    sign = 'b100000000000000;//初始序列信号
    input        CP,LDN;
    output       Q;
    reg          Q,F;
    reg[14:0]    SS;// 15位右移移位寄存器
    always @(posedge CP)
      begin
        if (!LDN) SS = sign;
        else
          begin
            F = (SS[1]^SS[0])||(~SS[14]&~SS[13]&~SS[12] &~SS[11]
             &~SS[10]&~SS[9]&~SS[8]&~SS[7]&~SS[6 ] &~SS[5]
             &~SS[4]&~SS[3]&~SS[2]&~SS[1]&~SS[0]);
             SS = SS >> 1; //在CP上升沿时，SS右移1位
             SS[14] = F;     //F反馈给SS的最高位
            Q = SS[14];     //Q是串行输出端，输出序列信号
          end
      end
endmodule
```

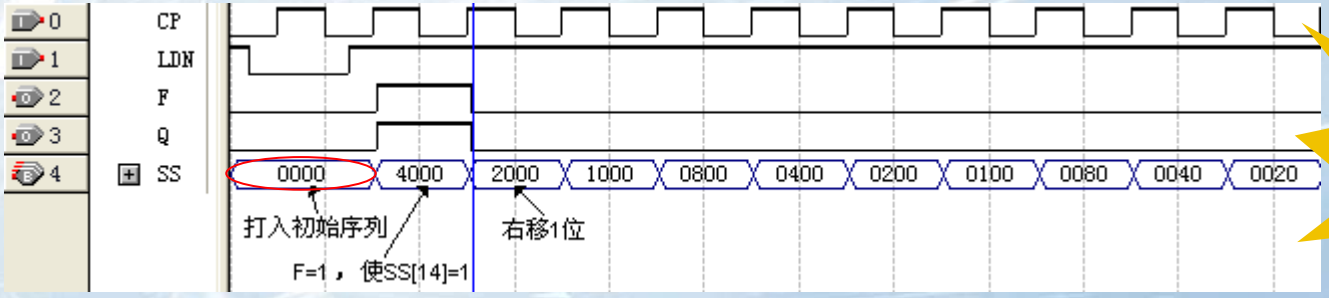


修改反馈函数后伪随机码发生器的仿真波形图





初始序列为15‘b100_0000_0000_0000时，Q输出随机序列信号



不存在死循环！

初始序列为15‘b000_0000_0000_0000，Q仍输出随机序列信号

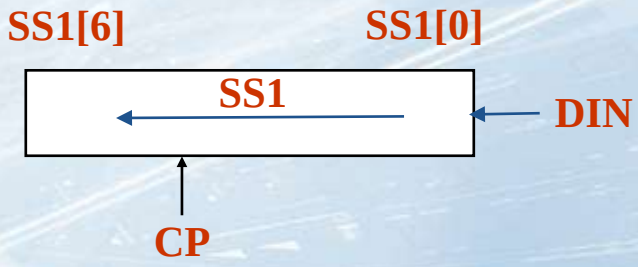
7.5.6 序列信号检测器的设计

- 序列信号检测器
  - 用于检测序列信号发生器送出的信号是否正确，如海难求助序列信号“SOS，SOS…”

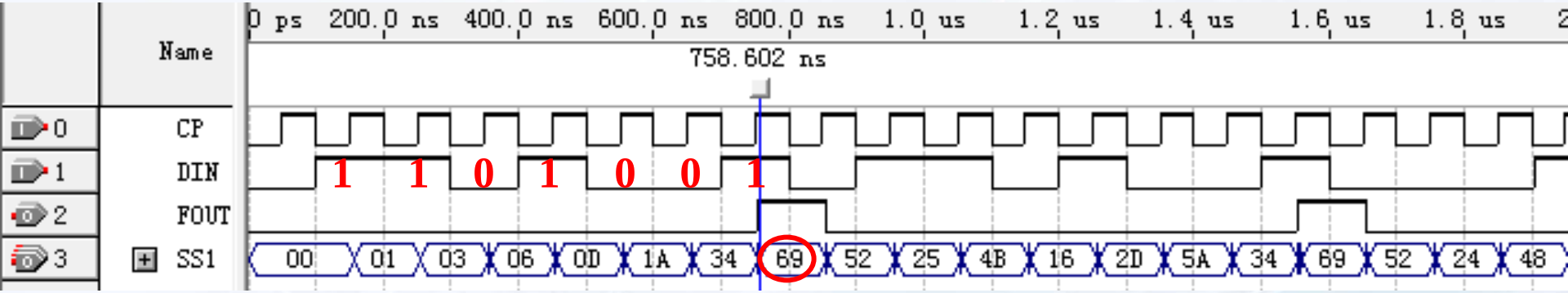
- 【例7.30】设计一个序列信号检测器
  - 检测 7位序列信号，当检测到从输入端DIN输入的序列信号为“1101001”（正确序列）时，输出FOUT=1
  - 否则，（未检测到正确序列或序列信号未检测结束）FOUT=0。

序列信号检测器的HDL设计

```
module monitor7_good(CP, DIN,FOUT);
    parameter                sign = 'b1101001;
    input                    CP,DIN;    //DIN为串行输入端
    output                   FOUT;
    reg                      FOUT;
    reg[6:0]                 SS0,SS1; //SS1为左移移位寄存器
    always @(posedge CP)
       begin
          SS1 = SS1 << 1;
          SS1[0] = DIN;      //从寄存器最低位串行输入序列信号
           if (SS1 == sign)   FOUT = 'b1;
          else        FOUT = 'b0;
       end
endmodule
```



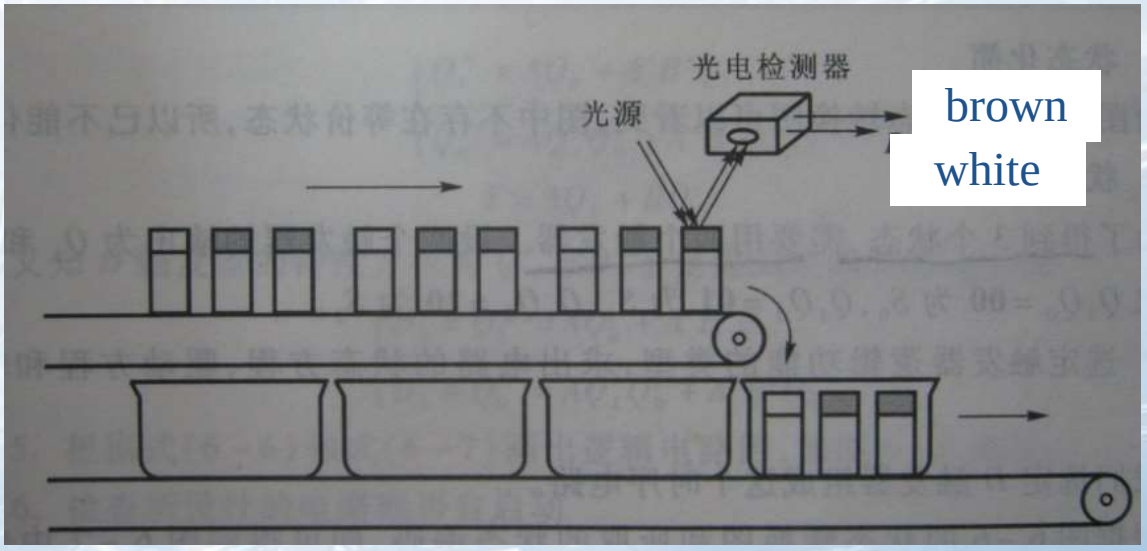
7位序列信号检测器的仿真波形图



- 编辑DIN：将序列信号发生器输出的序列信号“1101001”从最高位开始，依次输入DIN端
- 观察左移移位寄存器SS1的变化，7个CP后SS1 中寄存了7位序列信号，SS1[6]~SS1[0]为“110_1001”（69H），则输出FOUT=1
- 当DIN输入的序列不是1101001”时，可以看出FOUT=0。

【例7.29】状态机设计实例

❖【例7.31】设计一个咖啡产品包装线上用的检测逻辑电路。正常工作状态下，传送带顺序送出成品，每三瓶一组，装入一个纸箱中。每组含两瓶咖啡和一瓶咖啡伴侣，咖啡的顶盖为棕色，咖啡伴侣顶盖为白色。要求在传送带上的产品排列次序（咖啡，咖啡，咖啡伴侣）出现错误时检测逻辑电路能发出故障信号，同时自动返回初始状态。



解答分析

❖解： 首先需要得到区别两种瓶盖颜色的信号。

- 例如可以采用光电检测电路，利用棕、白两色瓶盖对入射光的反射率不同，在光电接收器的输出端得到两个不同的输出信号。
- 假定检测到棕色瓶盖时输出为brown（简写为B）=1，white（简写为W）=0。
- 检测到白色瓶盖时，输出为B=0，W=1。
- 没有检测到瓶盖时，光电接收器接收不到反射光，B=0，W=0。

1、进行逻辑抽象

- 输入变量：

  brown（B）=1表示棕色， white（W）=1表示白色。

  clk：时钟信号；clrn：复位信号，低有效，异步清零。
- 输出变量：用fail（F）表示故障，工作正常时fail=0， 有错误时fail=1。

定义逻辑状态的含义

- 确定电路的状态数

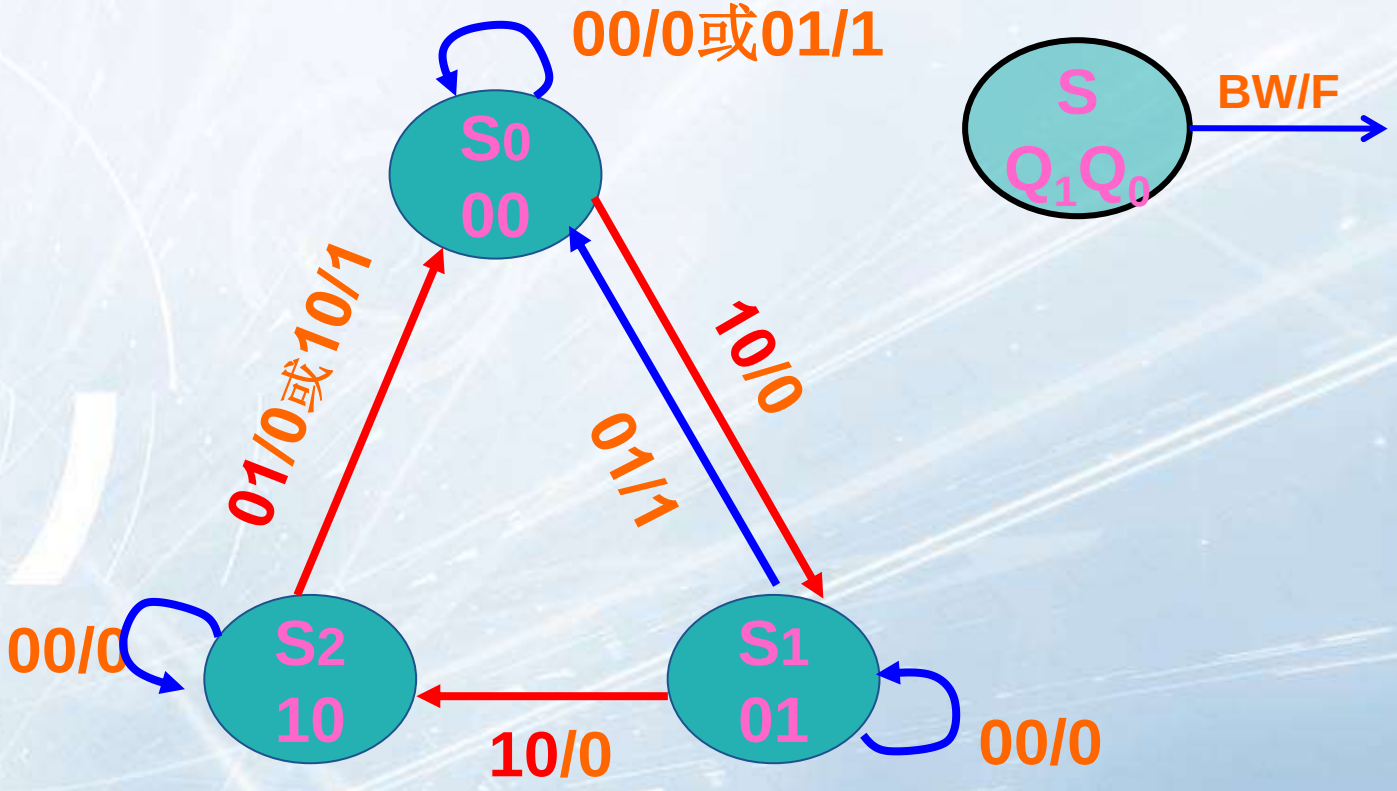
  设初始状态S_0，输入一个B=1后状态为S_1，再输入一个B=1后状态为S_2。状态S_2时，根据下一个输入信号就能决定输出信号，而且无论输出信号为0或1，电路都返回初始状态。故电路的状态数取3即可。

- 分析电路的状态转移情况
  - 状态S_0：（1）若BW=00，则F=0，时钟信号到达时保持S_0状态；（2）若BW=10，则F=0，时钟信号到达时转入次态S_1；（3）若BW=01，F=1，时钟信号到达时保持S_0不变。
  - 状态S_1：若BW=00，则F=0，时钟信号到达时保持S_1状态；若BW=10，则F=0，时钟信号到达时转入次态S_2；若BW=01，则F=1，时钟信号到达时返回S_0状态。
  - 状态S_2：若BW=00，则F=0，时钟信号到达时保持S_2状态；若BW=10，则F=1，时钟信号到达时电路返回S_0状态；若BW=01，则F=0，时钟信号到达时返回S_0状态。

状态转换图（Mealy型状态机）

2、画出状态转换图

- 由于状态机的输出不仅与状态机当前状态有关，而且与输入有关，所以本题属于Mealy型状态机。
- 每个状态下，输入BW有3种情况



咖啡产品包装线检测电路源程序（1/2）

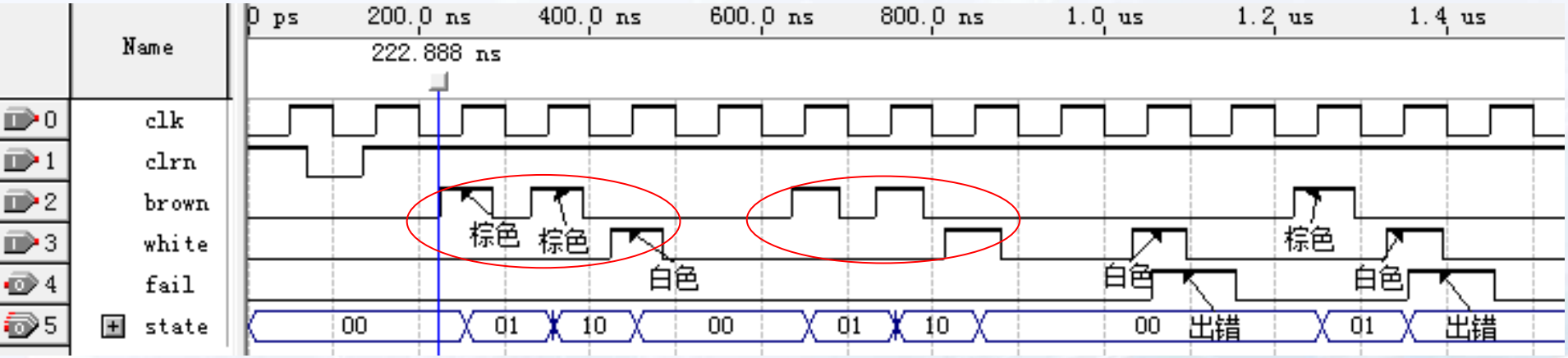
```
module coffee_line(brown,white,clk,clrn,fail,state);
    parameter S0=2'b00, S1=2'b01,S2=2'b10;
    input brown,white,clk,clrn;
    output fail;
    output[1:0] state;
    reg[1:0] state;
    reg fail;
    reg[1:0] in;
    always @(posedge clk or negedge clrn)
      begin
        in={brown,white};
        if (clrn==0) state=S0; //（1）复位时回到初始状态
        else
          begin
            case (state)      //（2）状态的转移及状态机输出
              S0: begin
                    case (in)
                      2'b00:begin state=S0;fail=1'b0;end
                      2'b01:begin state=S0;fail=1'b1;end//白色瓶盖
                      2'b10:begin state=S1;fail=1'b0;end
                    endcase
                  end
```

Mealy型状态机适宜采用单过程描述

咖啡产品包装线检测电路源程序（2/2）

```
                S1: begin
                        case (in)
                            2'b00:begin state=S1;fail=1'b0;end
                            2'b01:begin state=S0;fail=1'b1;end //白色瓶盖
                            2'b10:begin state=S2;fail=1'b0;end
                        endcase
                    end
                S2:begin
                        case (in)
                            2'b00:begin state=S2;fail=1'b0;end
                            2'b01:begin state=S0;fail=1'b0;end
                            2'b10:begin state=S0;fail=1'b1;end //棕色瓶盖
                         endcase
                    end
                default: state=S0;
            endcase
        end
    end
endmodule
```

咖啡产品包装线检测电路仿真波形



- 当检测到瓶盖为棕、棕、白时工作正常，fail=0；
- 若检测到瓶盖一开始就为白色，则说明排列顺序出错，fail=1
- 或者检测到瓶盖为棕、白，也说明排列顺序出错，fail=1

本章小结

1、时序逻辑电路概述

- 时序逻辑电路的描述方法
  - 方程组（输出方程、驱动方程、状态方程）
  - 状态转换表、状态转换图、时序图
- 时序逻辑电路分类
  - 同步时序逻辑电路
  - 异步时序逻辑电路
    - 脉冲异步电路
    - 电位异步电路

2、时序逻辑电路的分析方法

- 根据电路结构，写出方程式——时钟方程（异步时序逻辑电路要求）、输出方程、驱动方程和状态方程（将驱动方程代入触发器的特性方程得到）。
- 将输入变量和触发器初态的各种取值组合，代入状态方程和输出方程，计算出各级触发器的次态值和电路的输出值，计算得到状态转换表（真值表）。
- 根据状态转换表，画状态转换图或时序图。
- 根据状态转换图和时序图，说明电路的逻辑功能。

3、利用触发器构成计数器的设计方法

3、利用触发器构成计数器的设计方法

（1）确定采用何种触发器（JK、D触发器）、几个触发器，CP是上升沿触发还是下降沿触发

（2）根据题意画出计数器的时序图

（3）根据时序图写出状态方程

（4）将触发器的特性方程代入状态方程，得到驱动方程

（5）画出电路图

195

4、有限状态机

定义

有限状态机（Finite State Machine，FSM）是表示有限个状态以及这些状态之间的转移和动作等行为的离散数学模型。它由一组状态、一个初始状态（Reset电路）、输入和根据输入及现有状态转换为下一个状态的转换函数组成。

分类

- 摩尔（Moore）型状态机--输出信号仅与当前状态有关
- 米里（Mealy）型状态机--输出信号与当前状态及输入信号有关

有限状态机的设计

- 选择合理的起始状态
- 选择状态编码方式
  - 二进制编码、格雷编码、一位热码编码
- 状态编码的定义有两种方式：parameter和`define语句
- 用Verilog HDL语言描述状态机的方法（双过程）
  - 用1个always块描述（时序逻辑）

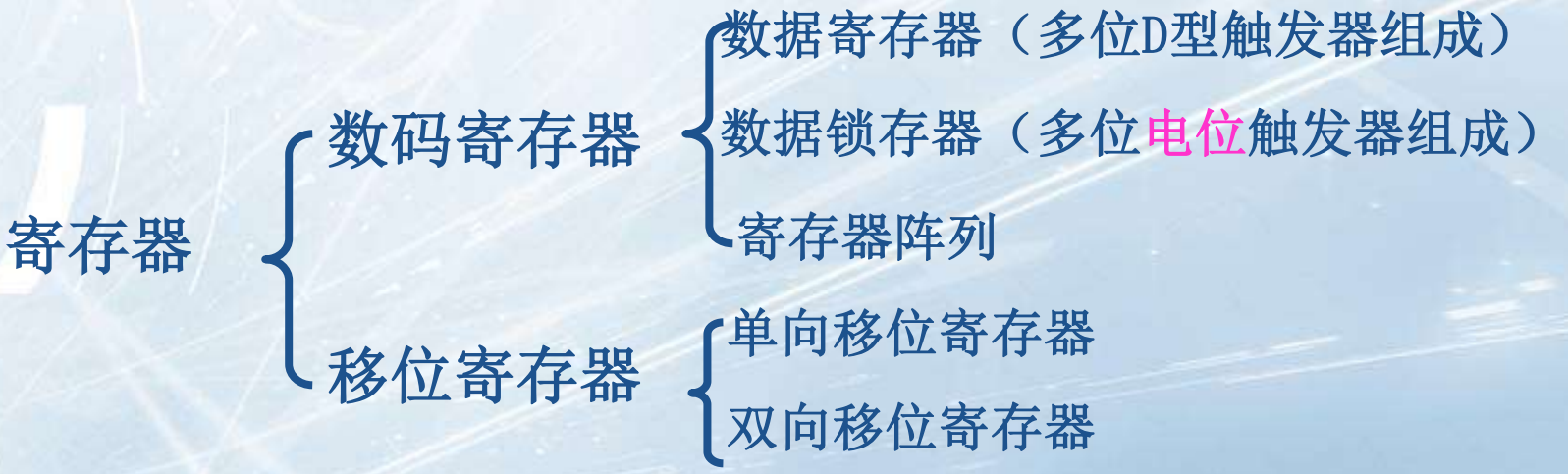
  （1）复位时回到初始状态

  （2）状态的转移（用case或if-else语句）
  - 用另1个always块描述

  （3）状态机的输出（组合逻辑）

5、寄存器

- 寄存器
  - 在计算设备中用于存储指令、数据、运算中间结果的存储单元
  - 寄存器由有记忆功能的触发器和一些接收数据的控制门组成
  - 触发器的触发方式决定了寄存器的触发方式



（1）数码寄存器

- 数码寄存器具有接收、存放和传输数码的功能。
- 各种类型的触发器都可以用来构成寄存器，用D触发器和D锁存器构成数码寄存器最为方便。

- 根据构成寄存器的记忆元件是D触发器还是D锁存器，数码寄存器区分为数据寄存器和数据锁存器
  - **数据寄存器和数据锁存器的区别：**
    - **数据寄存器由边沿触发器组成。同步时钟信号控制，脉冲敏感型，适于数据有效提前于控制信号有效、并要求同步操作的场合。**
    - **数据锁存器由电位触发器（即D锁存器）组成。电平信号控制，电平敏感型，适于数据有效滞后于控制信号有效的场合。**

199

（2）移位寄存器（移存器）

- 在时钟脉冲作用下，寄存器中数据依次向左移或向右移一位。
- 分类
  - 左移移位寄存器，右移移位寄存器，双向移位寄存器
- 数据输入方式
  - 串行输入，并行输入
- 数据输出方式
  - 串行输出：右移寄存器、左移寄存器
  - 并行输出：全部触发器的输出作为电路的输出
- 工作方式
  - 串入串出、串入并出、并入串出、并入并出

- **移位寄存器应采用边沿触发或主从触发方式的触发器，不能采用电位触发的触发器，以防止空翻。**

6、计数器

- 计数器的用途
  - （脉冲）计数，计时，定时（定时器），分频
  - 产生节拍脉冲（顺序脉冲）和序列脉冲
- 分类
  - 时钟方式：同步计数器，异步计数器
  - 计数方式：二进制计数器，十进制计数器，M进制计数器
  - 状态变化：加法计数器，减法计数器，加/减法计数器

（1）同步计数器

- 同步计数器的特点
  - 由JK触发器或D触发器构成
  - 所有触发器的时钟端并联在一起，作为时钟端
  - 在时钟脉冲的控制下，各触发器同时翻转
  - 工作速度比异步计数器快
  - 缺点：结构比较复杂，所用元件较多
- 同步计数器的分析
  - 根据驱动方程和触发器特性方程推出状态方程，列出状态转换表，画出状态转换图、时序图，说明电路特点
- 二进制加法计数器的特点
  - **模值为2^n（n是触发器的级数）**
  - **无非编码状态，能自启动**

（2）异步计数器

- 异步计数器特点
  - 全部由T’触发器构成（用D 触发器，使$D_i=/Q_i$；或用JK触发器，使$J_i=K_i=1$）
  - 第一级FF的CP由系统时钟控制，其余各级 FF的CP端由前级FF的Q端或/Q端控制
  - 工作速度比同步计数器慢
  - 各触发器之间的连接方式由加、减计数方式及触发器的触发方式决定。
- 异步计数器的分析
  - 与同步计数器的分析过程相同，但要格外注意触发器的触发方式
- 用反馈复位法实现异步M制计数器
  - 求反馈复位代码S_M，求反馈复位逻辑，画逻辑图

（3）集成计数器

- 集成计数器的同步扩展
  - 将若干片集成计数器级联起来，形成有较大模值的计数系统。
  - 假如一个计数器模值为M，则两片级联后，模值变为M^2。
  - 两片计数器的CP并联后接时钟信号CP——同步；低位片的ET、EP接高电平，使低位片始终具有计数功能；高位片的ET、EP接低位片的进位输出端C，只有当C为高电平时，高位片才具有计数功能。
- 集成计数器的异步扩展
  - 低位片的CP接系统时钟信号CP1，低位片的进位输出端C经反相后接高位片的CP端——异步。
- 集成计数器实现M进制计数
  - 反馈复位法：当计到规定的模值时，产生置0信号，使计数器复位
  - 输出C预置法：计到最大值时，输出C=1，反相后送 /LD端，预置数据
  - 输出Q预置法：求预置代码S_{M-1}，求预置逻辑，画逻辑图

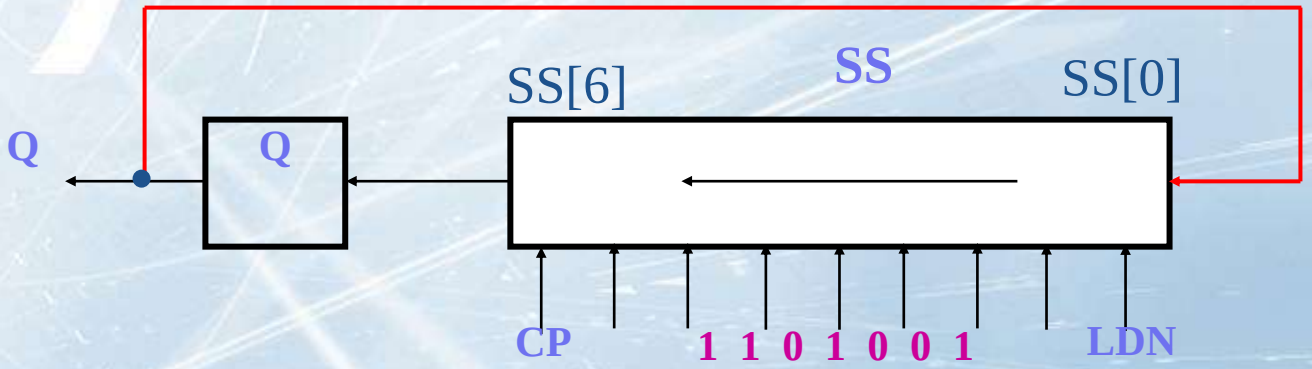
7、时序逻辑电路的HDL设计

- **一般采用行为描述方式：在always块中用if-else语句或case语句描述电路的逻辑功能**

- 数码寄存器的设计（CT74273、CT74373）
- 移位寄存器的设计
  - 使用移位运算符（<< 或 >>）
  - 双向移位寄存器
- 计数器的设计（同步计数器CT74160/161、191等的设计）
  - 注意同步复位与异步复位的区别
  - 要仿真所有的功能（复位、预置、计数、保持）
  - 一般分2个always块描述：计数操作（时序逻辑）、进位/借位操作（组合逻辑）

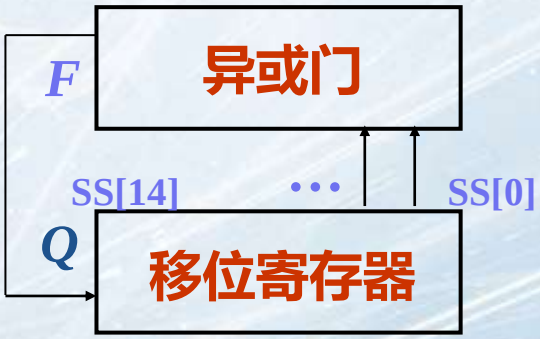
顺序脉冲发生器和序列信号发生器

- 顺序脉冲发生器的设计
  - 在时钟脉冲的控制下，各触发器输出端顺序产生脉冲信号
  - 实际为状态机的设计：状态编码定义、状态的转移（case语句）、状态机的输出
- 序列信号发生器的设计
  - 在时钟脉冲作用下，最高位（或最低位）触发器的输出端依次输出序列信号
  - 若利用左移移位寄存器，则串行输出端接移位寄存器的最高位；左移1位后，移位寄存器的前一时刻的最高位送入最低位



伪随机码发生器

- 伪随机码发生器的设计
  - 由一个移位寄存器和一个反馈网络组成（输入：取自Q_{N-1}~ Q_0，输出：反馈至移位寄存器的串行输入端）——当反馈网络为异或门，则构成最长线性序列移存型计数器
  - 为了打破死循环，要修改反馈函数，使Q全为0时F=1



序列信号检测器

- 序列信号检测器的设计
  - 检测序列信号发生器送出的信号是否正确
  - 利用移位寄存器
  - 将序列信号发生器输出的序列信号从最高位开始，从串行输入DIN端依次送入移位寄存器；n个CP后，序列信号全部移入寄存器中；然后与正确的序列信号比较，得出检测结果