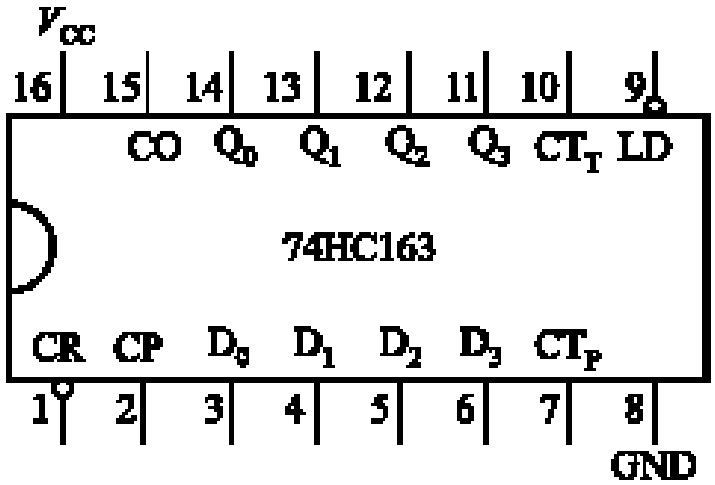
4.2.4 中规模集成计数器

中规模集成计数器不但具有计数功能，还有预置数、保持等功能，同时能方便地实现多片连接和功能的扩展，用处十分广泛。

对中规模集成计数器，主要要求能读懂真值表（功能表）、引脚排列，就可以使用它。

一、74HC163　4位二进制加法计数器

- 引脚排列



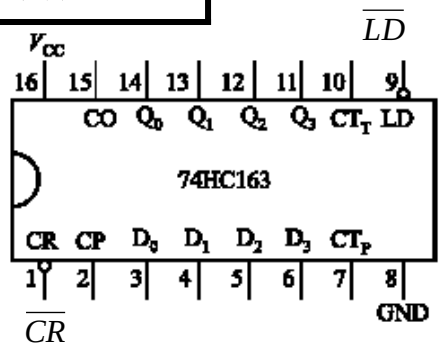
功能说明　74HC163计数器功能表

输入									触发器状态			
CP	$\overline{CR}$	$\overline{LD}$	CT_P	CT_T	D_3	D_2	D_1	D_0	Q_3	Q_2	Q_1	Q_0
↑	0	×	×	×	×	×	×	×	0	0	0	0
↑	1	0	×	×	A_3	A_2	A_1	A_0	A_3	A_2	A_1	A_0
↑	1	1	1	1	×	×	×	×	4位二进制加计数			
↑	1	1	0	×	×	×	×	×	保持功能			
×	1	1	×	0	×	×	×	×	保持功能			

同步清零: ①“清零”控制端首先有效，即$\overline{CR}=0$；
②时钟上升沿触发“清零”。

同步置数: ①“清零”控制端无效、“置数”控制端有效$\overline{CR}=1$　$\overline{LD}=0$　；

②时钟脉冲上升沿触发“置数”



同步清零、同步置数在“清零”端、“置数”端加上规定的逻辑电平外，还需要CP脉冲；而异步清零、异步置数只要在“清零”端、“置数”端加上规定的逻辑电平即可，不需要加时钟脉冲CP 。

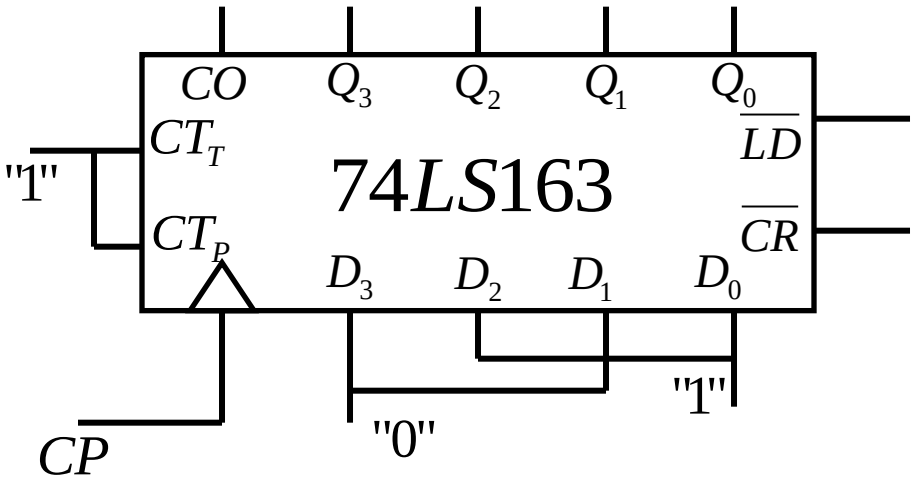
74HC161引脚排列与163完全相同，功能上74HC161是异步清零，同步置数,使用时应十分注意。

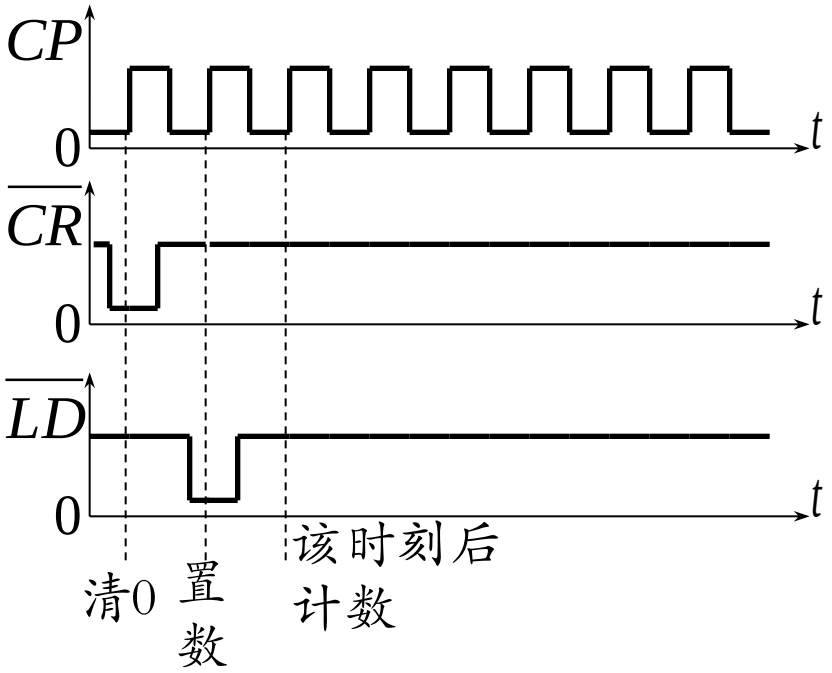
74HC161功能表

输入									触发器状态				
CP	$\overline{CR}$	$\overline{LD}$	CT_P	CT_T	D_3	D_2	D_1	D_0	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}	
X	0	x	x	x	x	x	x	x	0	0	0	0	← 异步清零
↑	1	0	x	x	A_3	A_2	A_1	A_0	A_3	A_2	A_1	A_0	← 同步置数
↑	1	1	1	1	x	x	x	x	4位二进制加计数				← 加法计数
↑	1	1	0	x	x	x	x	x	保持				
x	1	1	x	0	x	x	x	x	不变				

➢ 使用说明

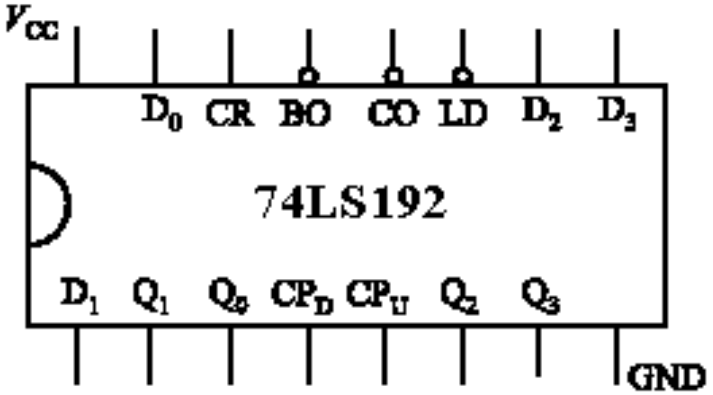
【例】根据功能表，画出将74LS163连接成从清“0”开始，然后置入0101数据后开始计数的各端波形安排和连接图。



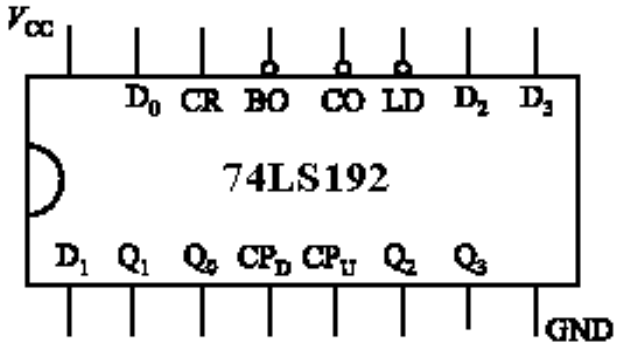


2. 8421BCD码十进制可逆计数器

74LS192



输入								触发器状态			
CR	$\overline{LD}$	CP_U	CP_D	D_3	D_2	D_1	D_0	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}
1	x	x	x	x	x	x	x	0	0	0	0
0	0	x	x	A_3	A_2	A_1	A_0	A_3	A_2	A_1	A_0
0	1	↑	1	x	x	x	x	8421十进制加法计数			
0	1	1	↑	x	x	x	x	8421十进制减法计数			
0	1	1	1	x	x	x	x	保　　持			



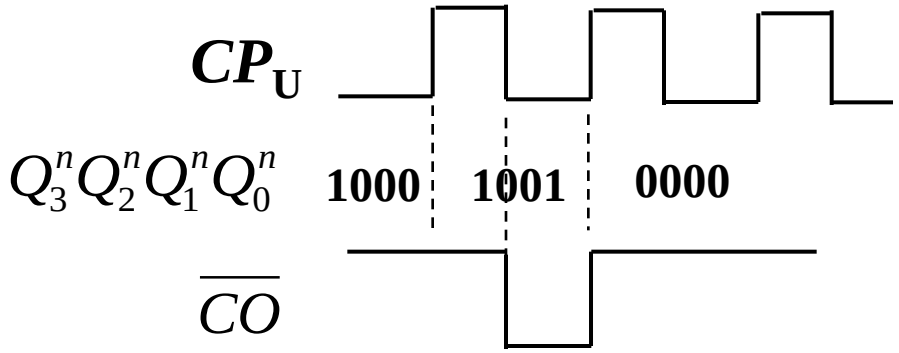
$$\overline{CO} = \overline{Q_3^n Q_0^n CP_U}$$ 进位输出

$$\overline{BO} = \overline{Q_3^n Q_2^n Q_1^n Q_0^n CP_D}$$ 借位输出

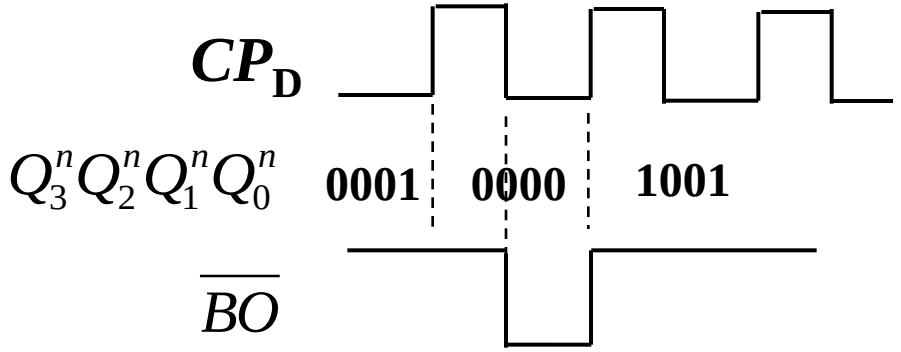
输入								触发器状态			
CR	$\overline{LD}$	CP_U	CP_D	D_3	D_2	D_1	D_0	Q_3^{n+1}	Q_2^{n+1}	Q_1^{n+1}	Q_0^{n+1}
1	x	x	x	x	x	x	x	0	0	0	0
0	0	x	x	A_3	A_2	A_1	A_0	A_3	A_2	A_1	A_0
0	1	↑	1	x	x	x	x	8421十进制加法计数			
0	1	1	↑	x	x	x	x	8421十进制减法计数			
0	1	1	1	x	x	x	x	保　　持			

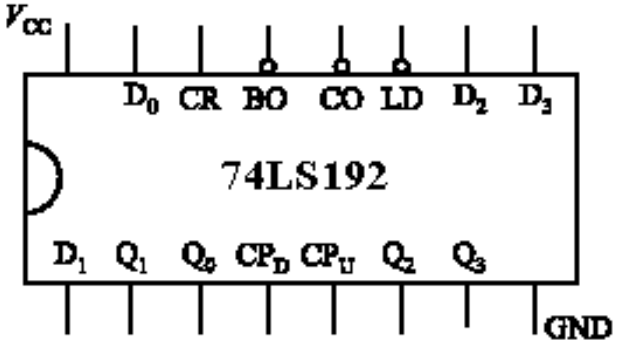
特点：双时钟触发，异步高电平清零，异步低电平置数

$$\overline{CO} = \overline{Q_3^n Q_0^n \overline{CP_U}}$$



$$\overline{BO} = \overline{\overline{Q_3^n}\,\overline{Q_2^n}\,\overline{Q_1^n}\,\overline{Q_0^n}\,\overline{CP_D}}$$





二、利用中规模集成计数器实现N进制计数

中规模集成计数器有清零端、置数端、数据输入端、进位借位输出端、扩展控制端等，利用这些端可以把中规模集成计数器连接成各种进制的计数器。

1. 反馈清零法

在正常计数时，清零端 $\overline{CR}$ 或 CR 应在高电平（或低电平），当计到某个数时，清零端变为低电平，然后又回到高电平，计数器重新开始计数。

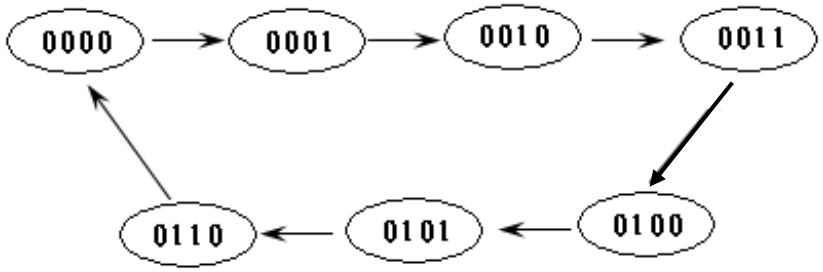
具体步骤：① 确定N进制计数器的S_N代码；

② 求出 $\overline{CR}$（或 CR）的控制逻辑关系；

③ 画出逻辑电路图。

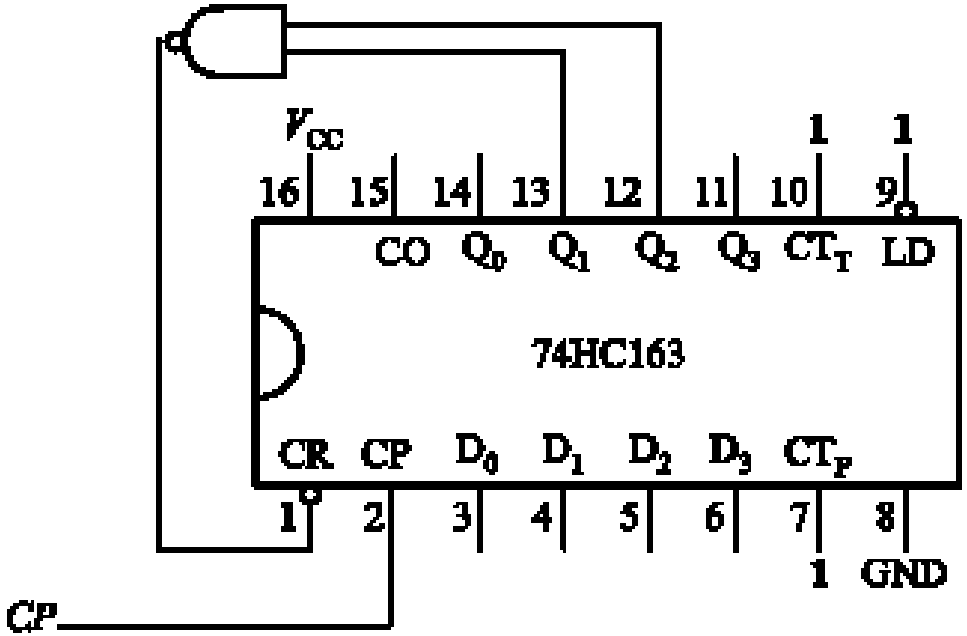
必须注意：同步清零与异步清零的区别。

【例】试分别用74HC161和74HC163型中规模集成计数器连接成8421BCD编码的七进制加法计数器。

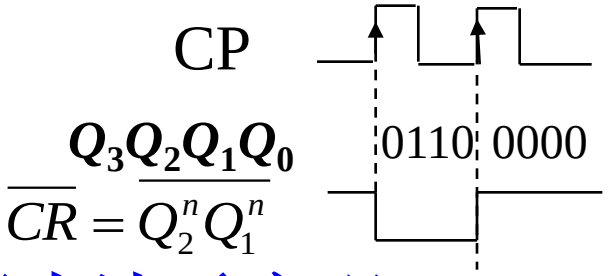


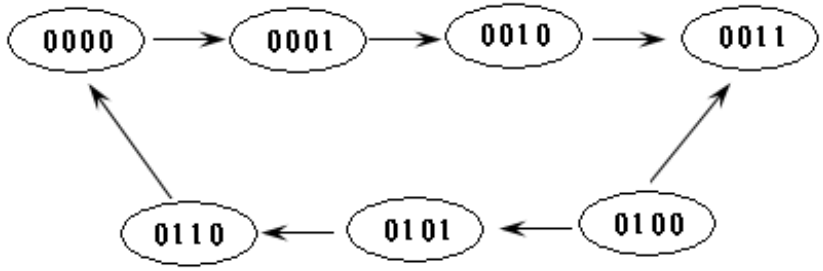
解：74LS163是一个4位二进制加法计数器，模为16，是同步低电平清零。

用$S_{N-1}=Q_3Q_2Q_1Q_0$=0110代码



同步清零时：当计数状态未到0110前，清零端为高电平，一当状态到达0110时，清零端为“0”，等待下一个CP到达后清零，然后清零端又恢复高电平，并重新开始下一轮计数。



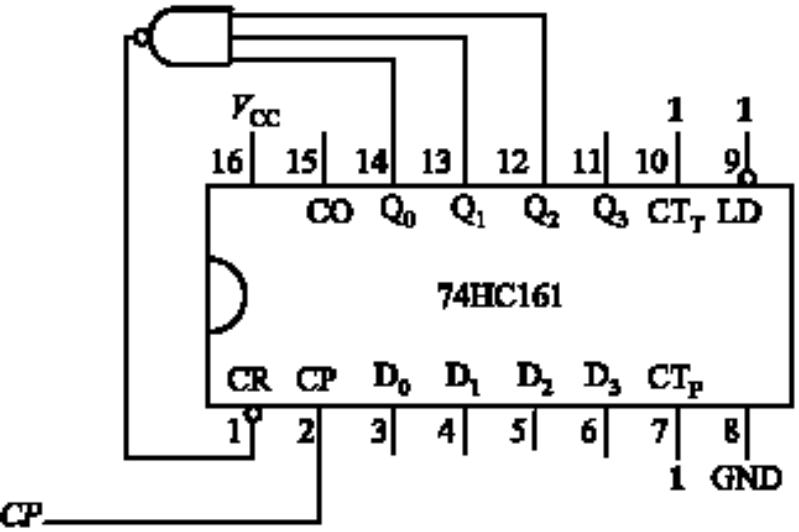


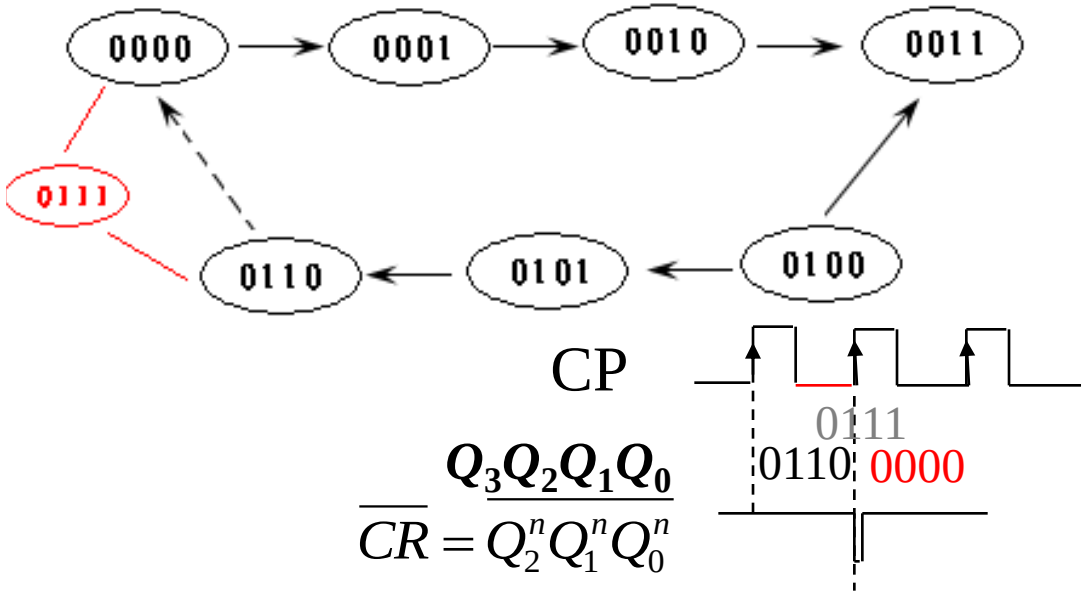
74HC163同步清零实现

异步清零时：当计数状态达到0110时，清零端仍为高电平，只有当状态进入0111时，清零端为“0”，立刻将计数器状态清为0000，然后清零端又恢复高电平，并重新开始下一轮计数。

$S_7=Q_3Q_2Q_1Q_0=0111$

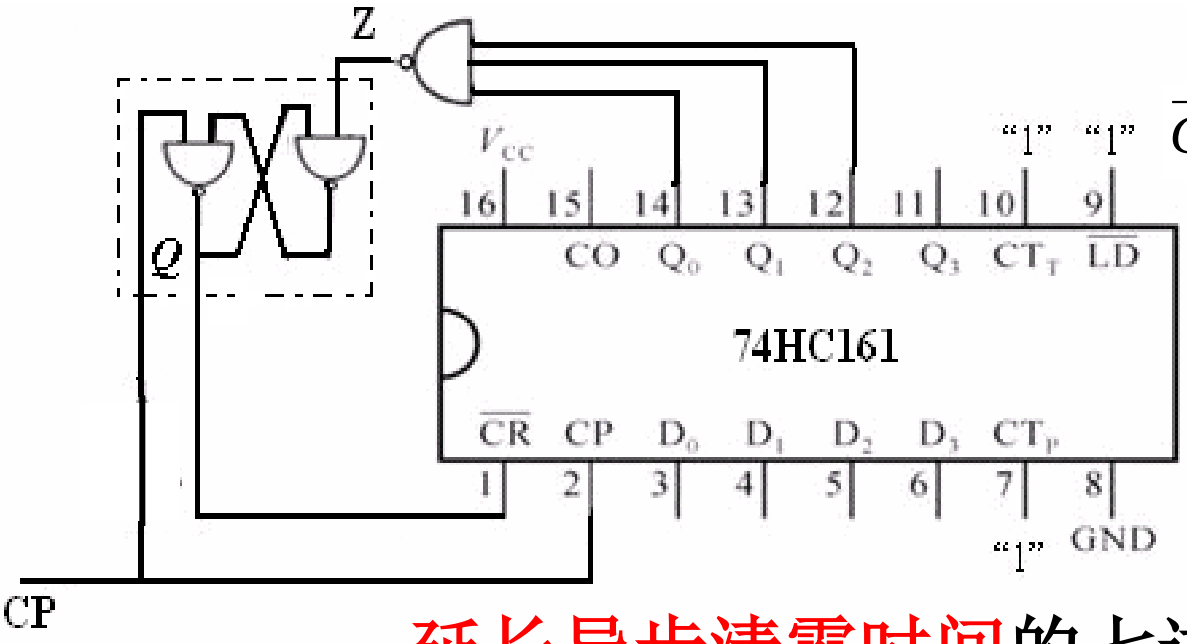
$$\overline{CR}=\overline{Q_2^nQ_1^nQ_0^n}$$





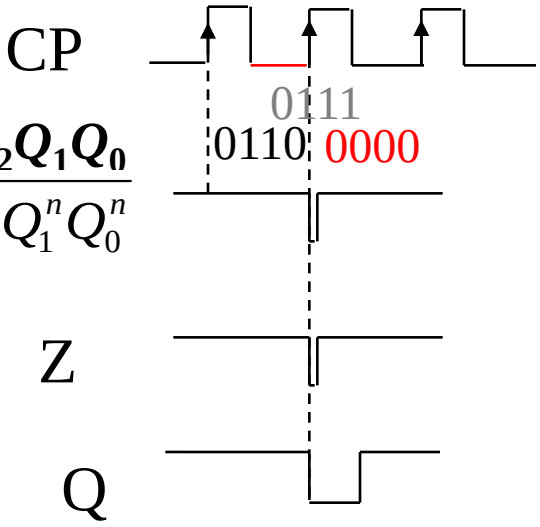
74HC161异步清零实现

由于异步清零时间非常短暂，而且出现了一个非有效计数循环中的状态0111（即过渡状态），当各触发器的翻转时间不一致，例如，若Q3、Q2先被清零，则 $\overline{CR}=\overline{Q_2^nQ_1^nQ_0^n}$ 将无效，从而会导致Q1、Q0无法清零的不可靠现象。

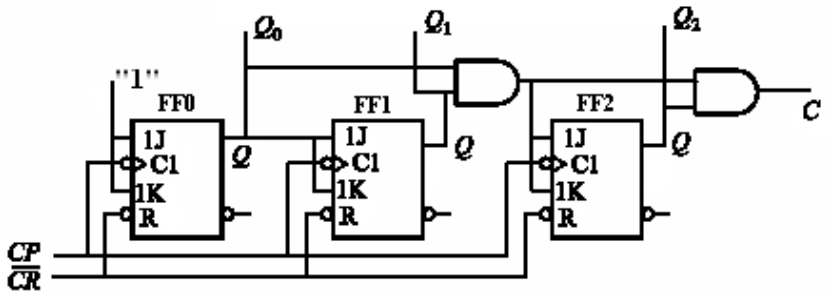


$Q_3Q_2Q_1Q_0$

$\overline{CR}=\overline{Q_2^nQ_1^nQ_0^n}$



延长异步清零时间的七进制加法计数器



2. 反馈置数法

基本思路：计数器可以从0…0开始计数，也可从某一个数字开始计数，而0…0或某个数字可以从数据输入端预置入计数器，然后计数。

具体步骤：

①画出计数器的状态转换图；

②将状态图中的最小数从预置数输入端输入，最大数状态作置数控制，求出置数控制端 $\overline{LD}$ 或 LD 的逻辑函数（指加法计数）；方法同清零法。

③画出逻辑电路图。

必须十分注意：异步置数时，置数控制函数式应取计数循环中最大数加1。

【例】

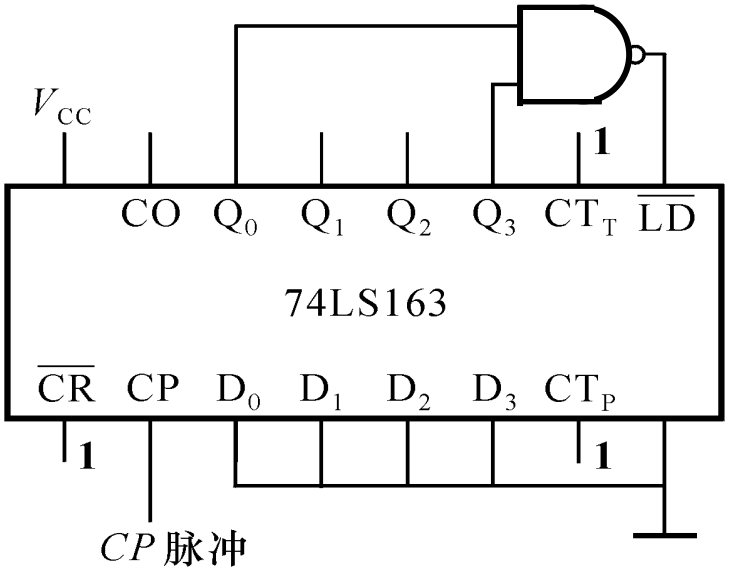
试用置数法将74LS163中规模集成四位二进制加法计数器连接成8421编码的十进制加法计数。

解：74LS163是同步低电平置数，而8421计数的状态转换图$Q_3Q_2Q_1Q_0$最小数为0000，最大数是1001。所以，数据端的数据应为：$D_3D_2D_1D_0$=0000，

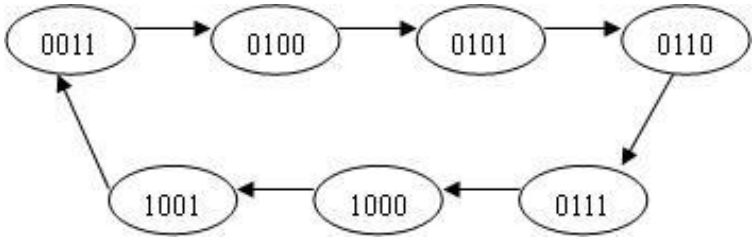
置数控制逻辑为：

$$\overline{LD}=\overline{Q_3Q_0}$$

连接成的电路图为：

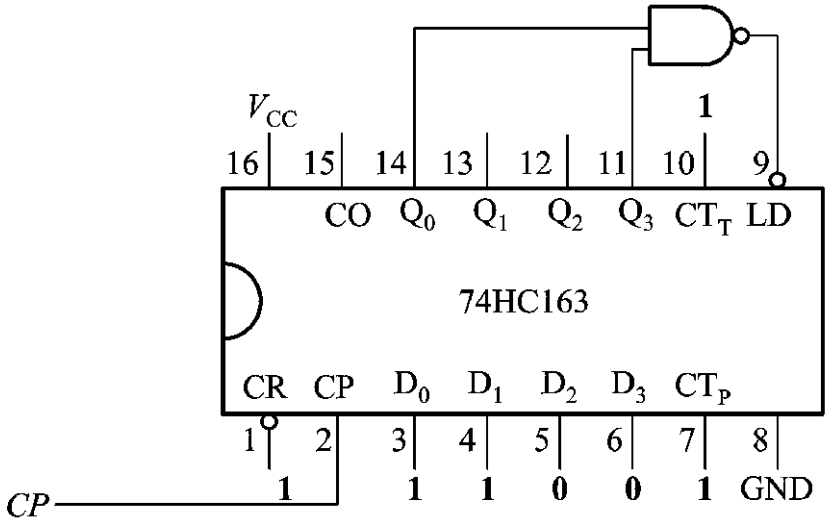


例： 试用74HC163型中规模集成计数器连接成一个余3BCD编码的七进制加法计数器。



解： 74HC163是4位二进制加法计数器，同步清零和同步置数，而且低电平置数。余3BCD码计数时，应将初态0011从数据端输入，而最后状态1001作为置数控制状态，所以：

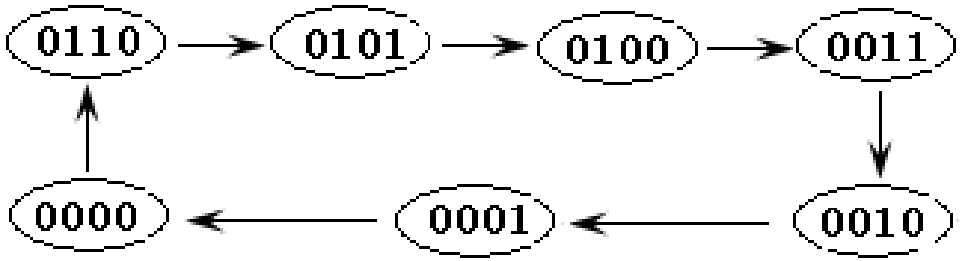
$$\overline{LD} = \overline{Q_3^n Q_0^n}$$

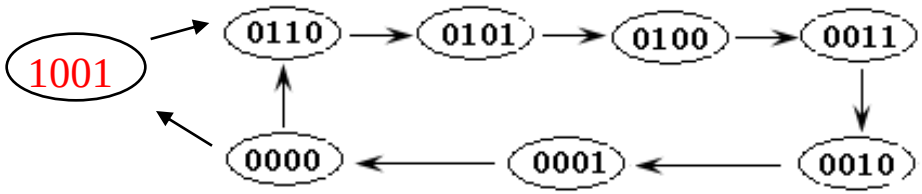


例： 试用74LS192型中规模集成计数器连接成一个8421BCD编码的七进制减法计数器。

解： 74LS192原是一个双时钟8421BCD码的十进制可逆计数器，异步高电平清零和异步低电平置数。因此，要实现减法时，必须设置在减法模式下。

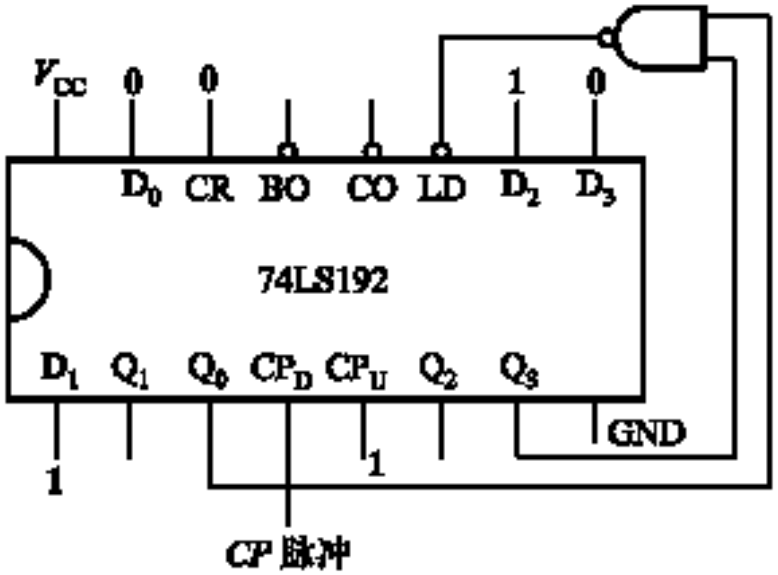
8421BCD码七进制减法状态转换图为：





初始状态$Q_3Q_2Q_1Q_0$=0110从数据端加入，置数控制用最后状态的后一个状态，即0000减1后出现1001，用$Q_3Q_2Q_1Q_0$=1001作置数控制。

$$\overline{LD}=\overline{Q_3^nQ_0^n}$$



3. 多次置数法实现*N* 进制

清零法和置数法都是依据原计数器的状态转换规律是连续的自然二进制。因此我们可以取其中的某一区段，实现*N*进制的目的。

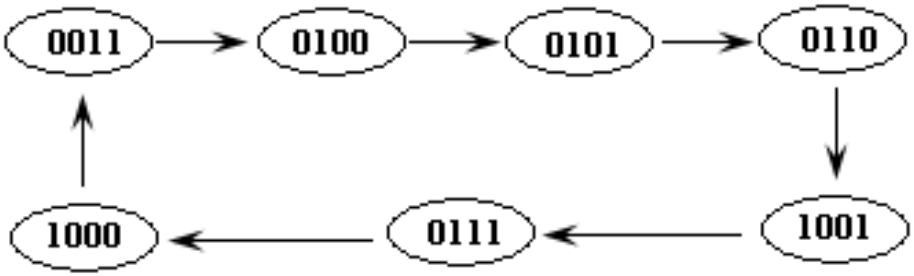
如果原状态转换不是自然二进制，则清零和置数法都无法实现*N*进的目的。

解决的方法是采用计数和置数交替进行，实现*N*进制。

具体方法：

如果在计数的状态转换中初态到次态是以二进制数的自然规律递增或递减时，按原计数规律计数，如状态转换中的初次态不连续时，则次态用置数法实现，最终完成整个计数循环。

例5:试用74HC161中规模集成四位二进制加法计数器连接成按图示状态进行转换的七进制计数器。

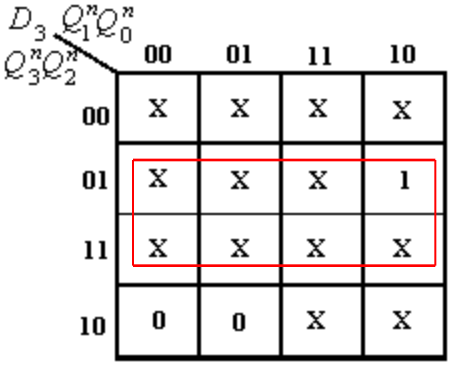


解:分析状态转换要求得知，状态3到状态6，及状态7到状态8按原计数规律计数。

而状态6到状态9，状态9到状态7，状态8到状态3这三种转换必须用置数法才能实现。

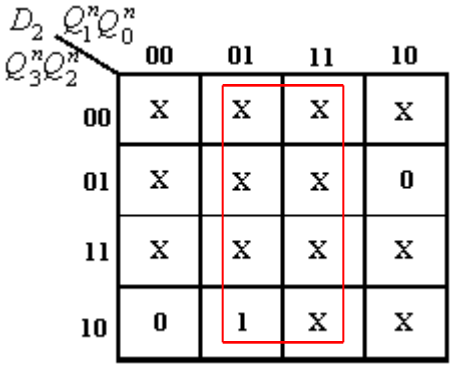
状态转换真值表

时钟	初态				次态				置控	数据输入			
CP	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}	$\overline{LD}$	D_3	D_2	D_1	D_0
1	0	0	1	1	0	1	0	0	1	×	×	×	×
2	0	1	0	0	0	1	0	1	1	×	×	×	×
3	0	1	0	1	0	1	1	0	1	×	×	×	×
4	0	1	1	0	1	0	0	1	0	1	0	0	1
5	1	0	0	1	0	1	1	1	0	0	1	1	1
6	0	1	1	1	1	0	0	0	1	×	×	×	×
7	1	0	0	0	0	0	1	1	0	0	0	1	1

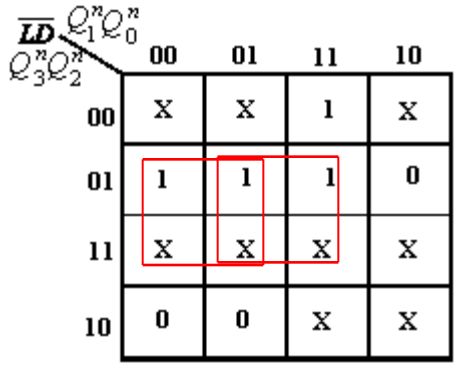


$D_3 = Q_2^n$

用卡诺图求置数控制端和$D_3D_2D_1D_0$逻辑函数。注意，计数循环以外的9个状态当约束项处理。

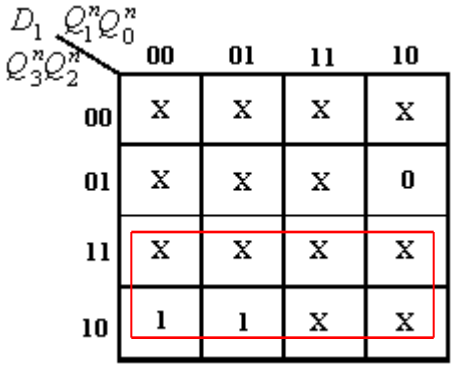


$$D_2 = Q_0^n$$

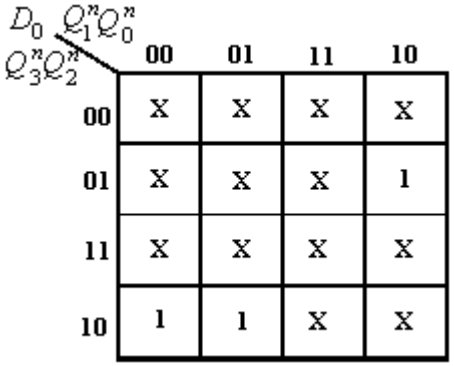


$$\overline{LD} = Q_2^n\overline{Q_1^n} + Q_2^nQ_0^n$$

$$= Q_2^n(\overline{Q_1^n} + Q_0^n)$$



$$D_1 = Q_3^n$$



$$D_0 = \text{"1"}$$

3. 大容量计数器的实现

大容量计数器可由小容量计数器级联而成，

$M=M_1 \cdot M_2 \cdot M_3 \cdots$

如一个60进制计数器可用一个6进制和一个10进制计数器串联构成，即60=6×10。其中6进制和10进制计数器可选用清零法和置数法中的任一种实现。

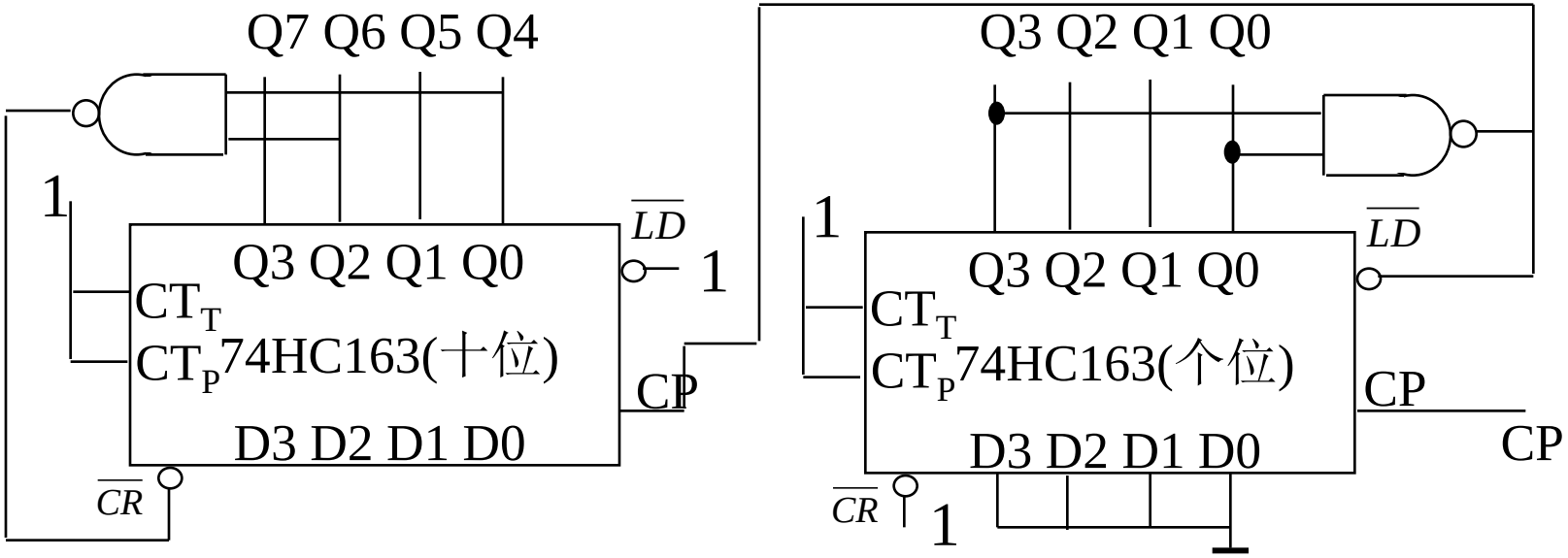
同理，100进制可用两个10 进制计数器构成，电路可采用同步或异步连接方式。

【例】 试用中规模集成计数器74HC163设计一个8421BCD编码的60进制计数器。

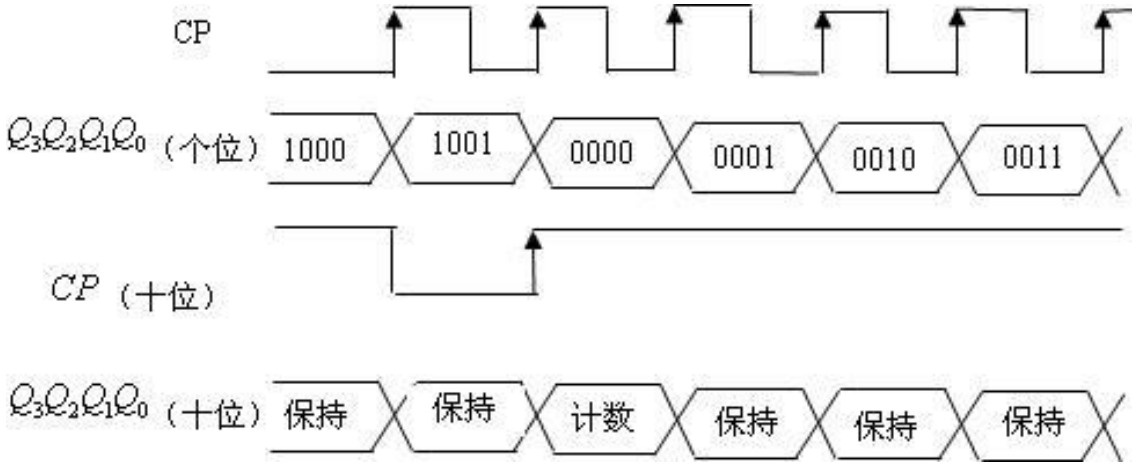
60进制计数器必须分解成二级实现，拾位和个位各用一片集成计数器实现

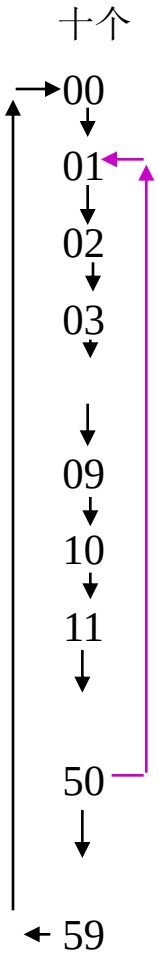
1. 异步级联法

- 74LS163是同步清零、置数，上升沿触发。

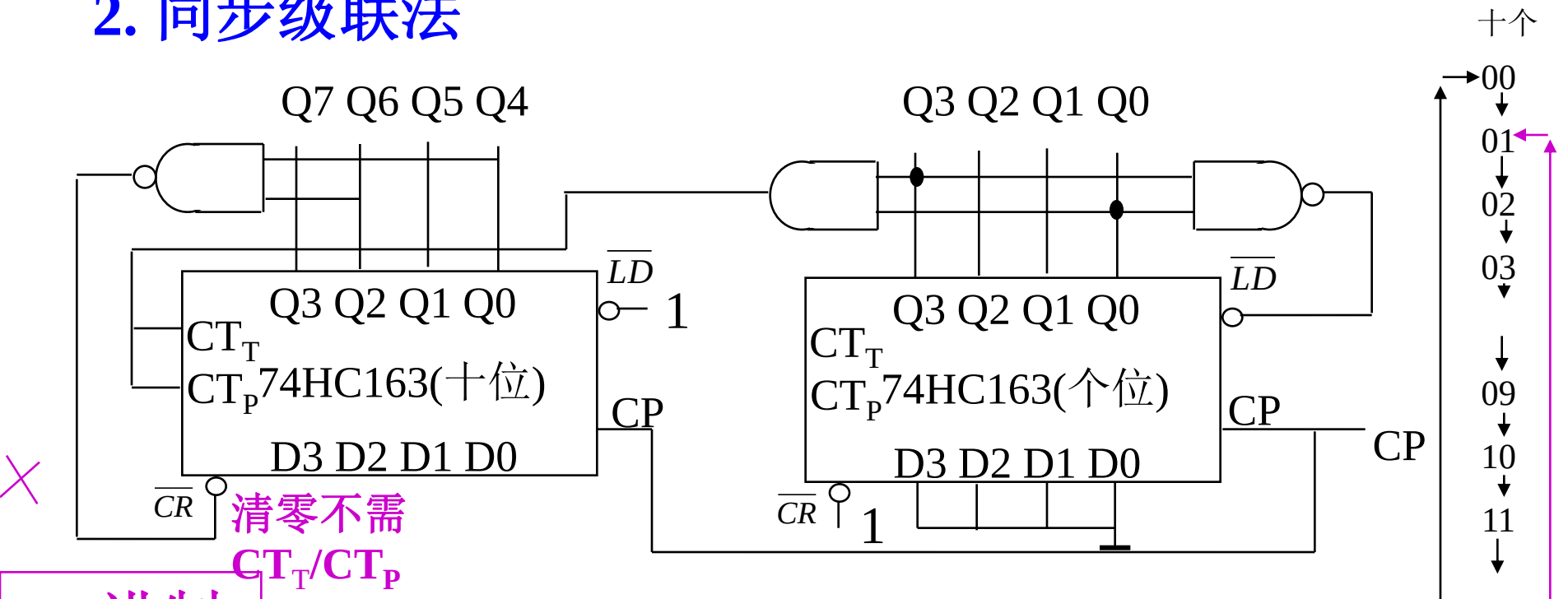


异步方式实现8421BCD编码的60进制计数器



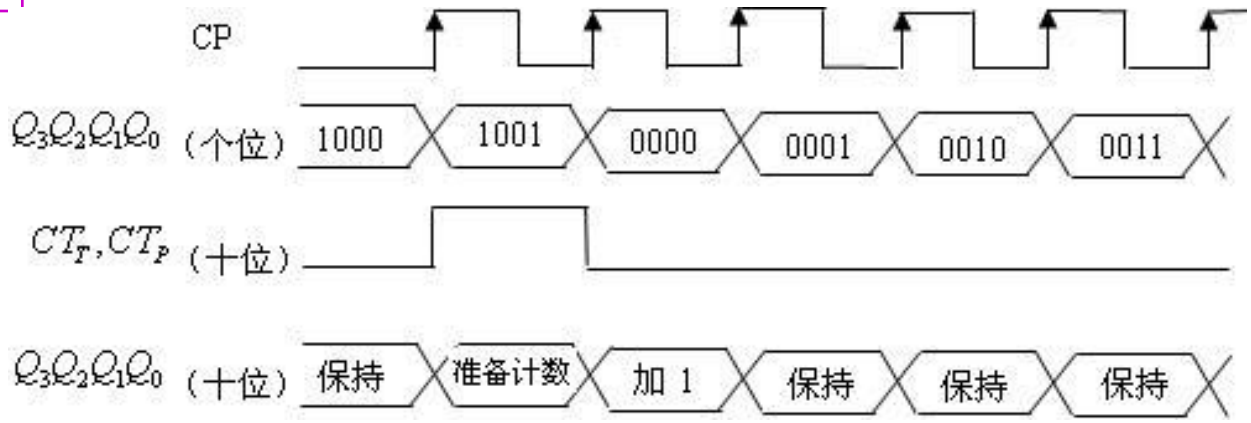


2. 同步级联法

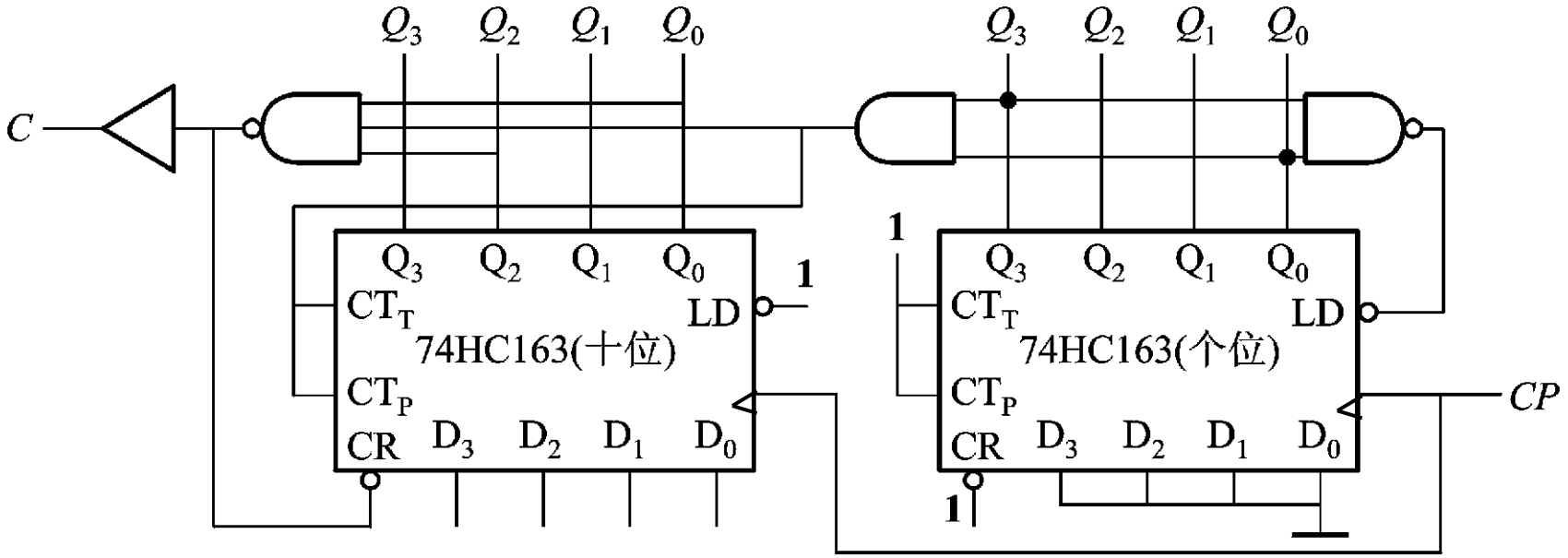


50进制

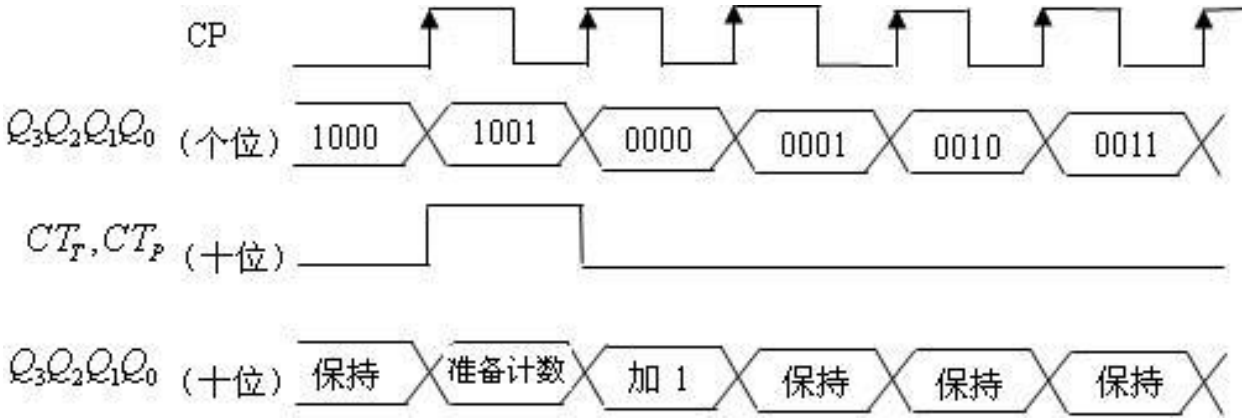
8421BCD编码的60进制同步计数器

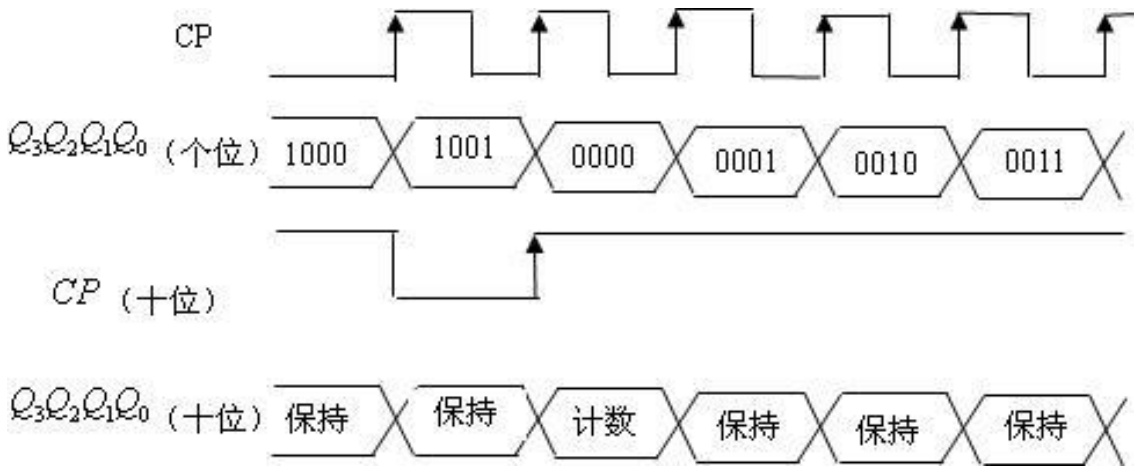


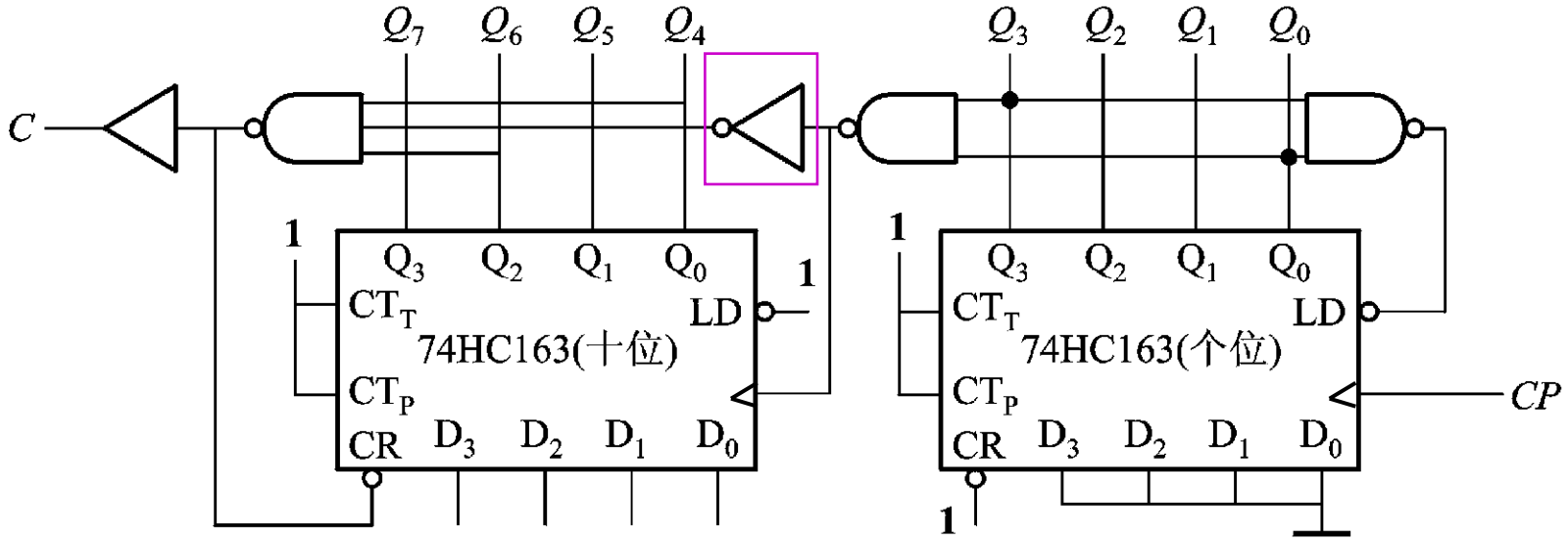
- 74LS163是同步清零、置数，上升沿触发。



8421BCD编码的60进制同步计数器

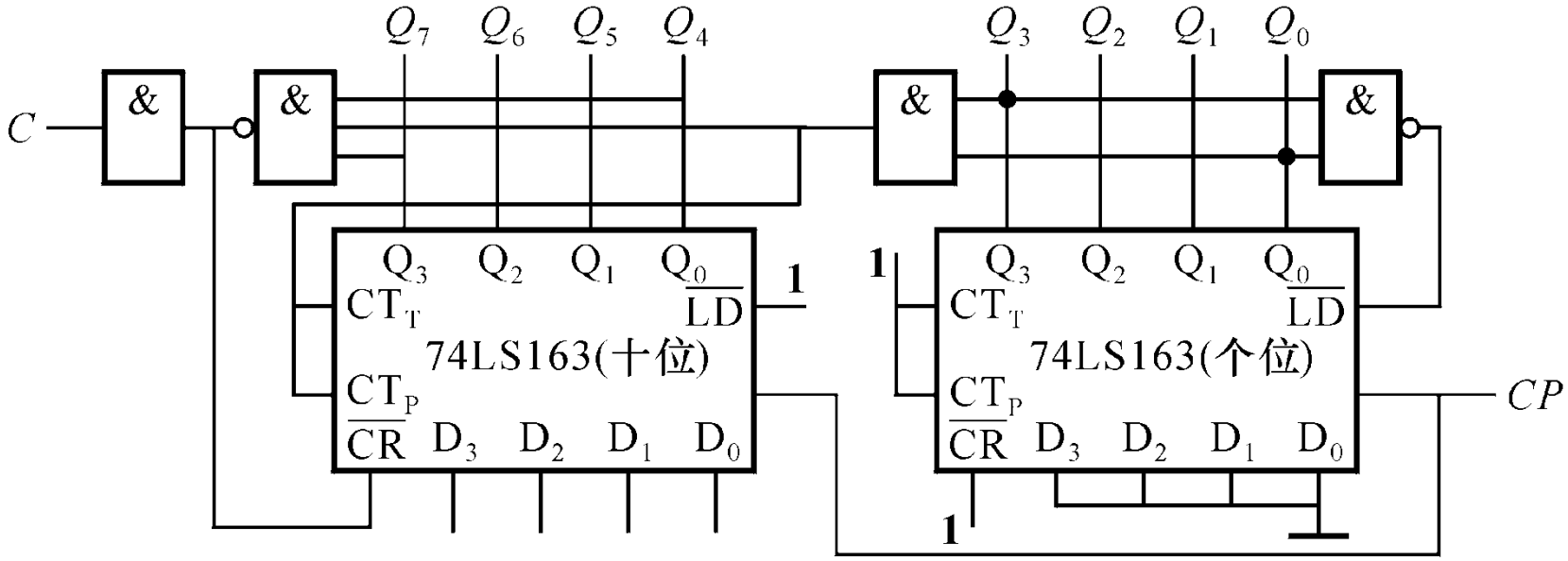






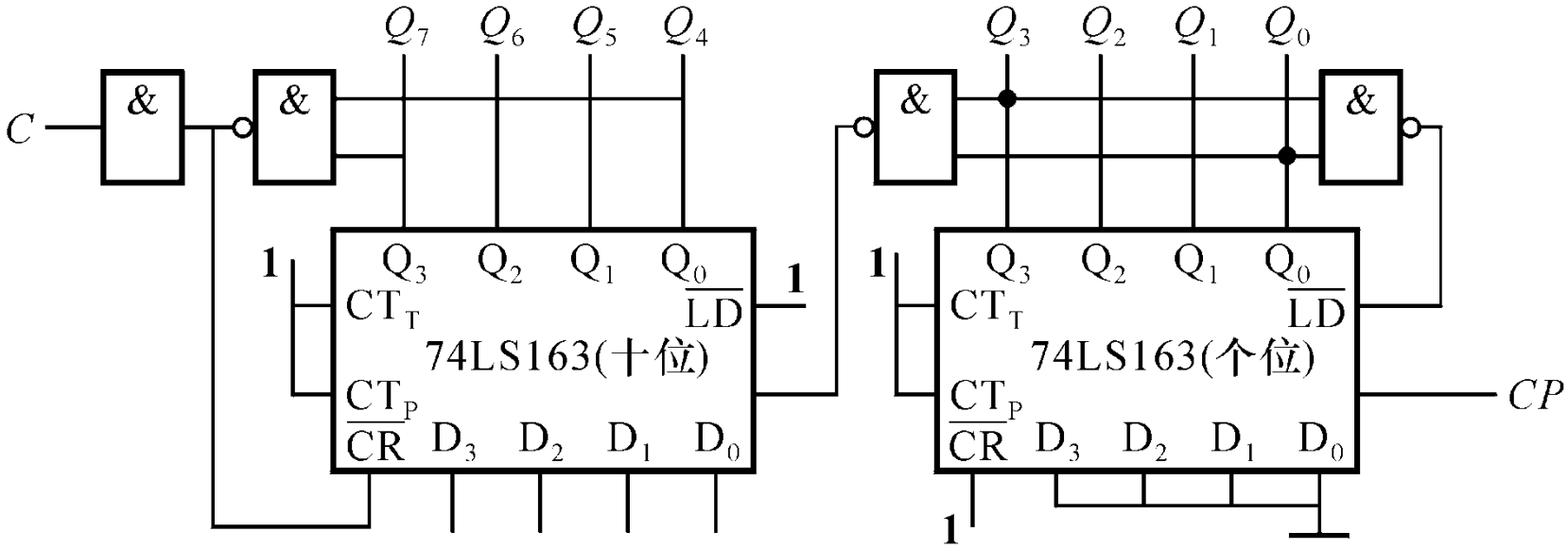
异步方式实现8421BCD编码的60进制计数器

同步式的100进制BCD码计数器

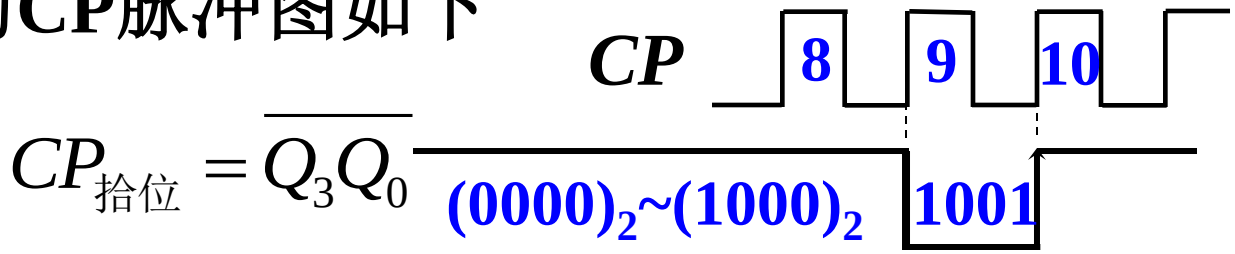


- 拾位10进制用清零法实现，个位10进用置数法实现。
- 当个位尚未计到1001前，拾位计数器的CT_P、CT_T为低电平，拾位计数器不计数。
- 当个位计到9时，拾位的CT_P、CT_T为1，而下一个计数脉冲CP来到后，拾位计一个1，个位计数器回到0，然后又封锁拾位计数器，只有个位计数。

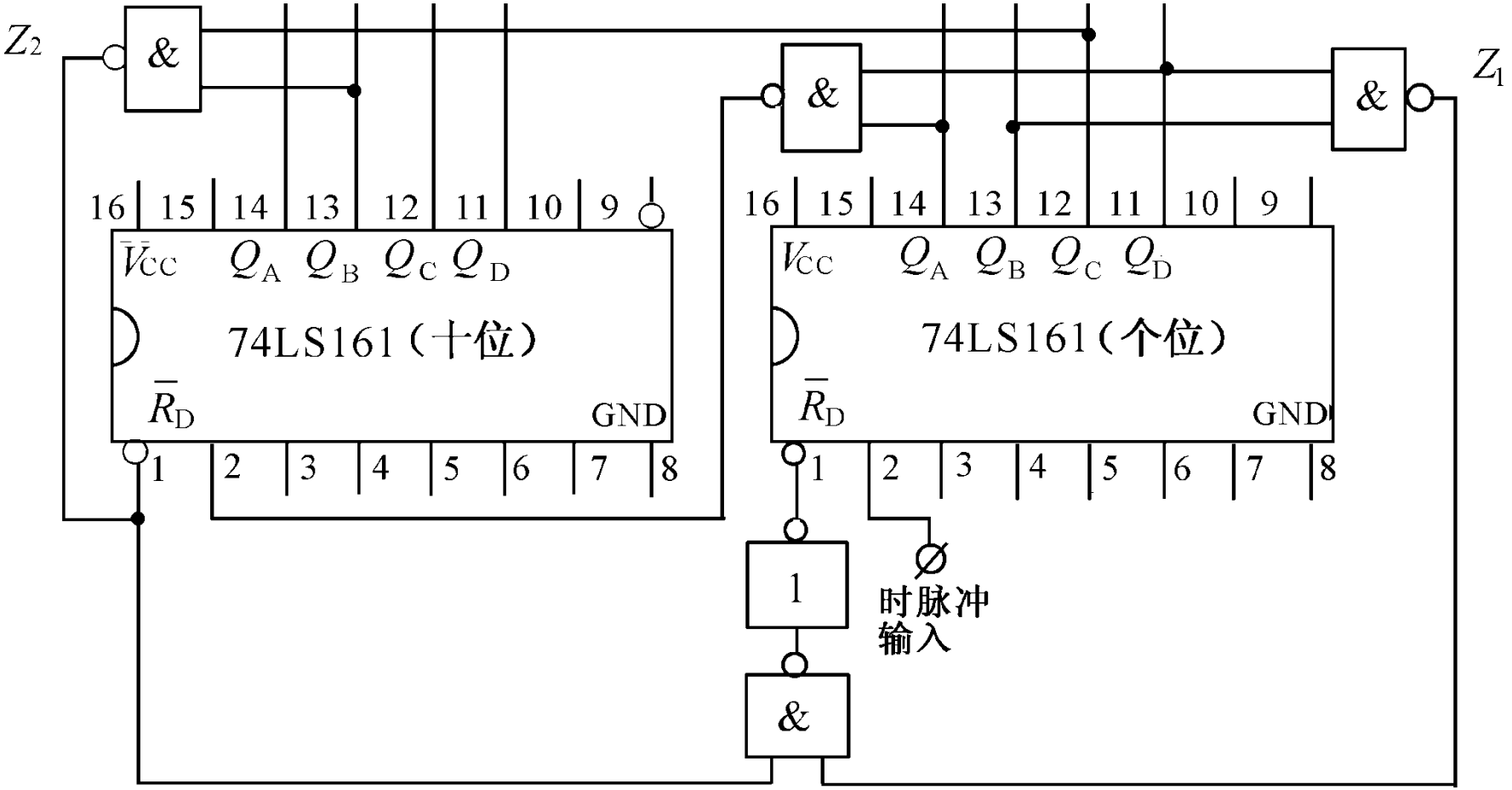
异步式的100进制计数器



- **74LS163是同步清零、置数，上升沿触发。**
- **拾位的CP脉冲图如下**



24进制计数器（显示小时）

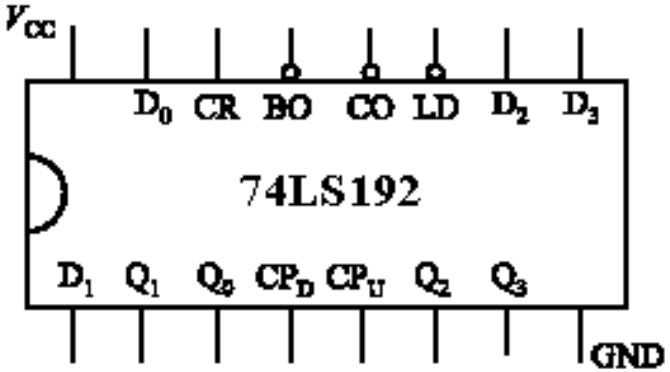


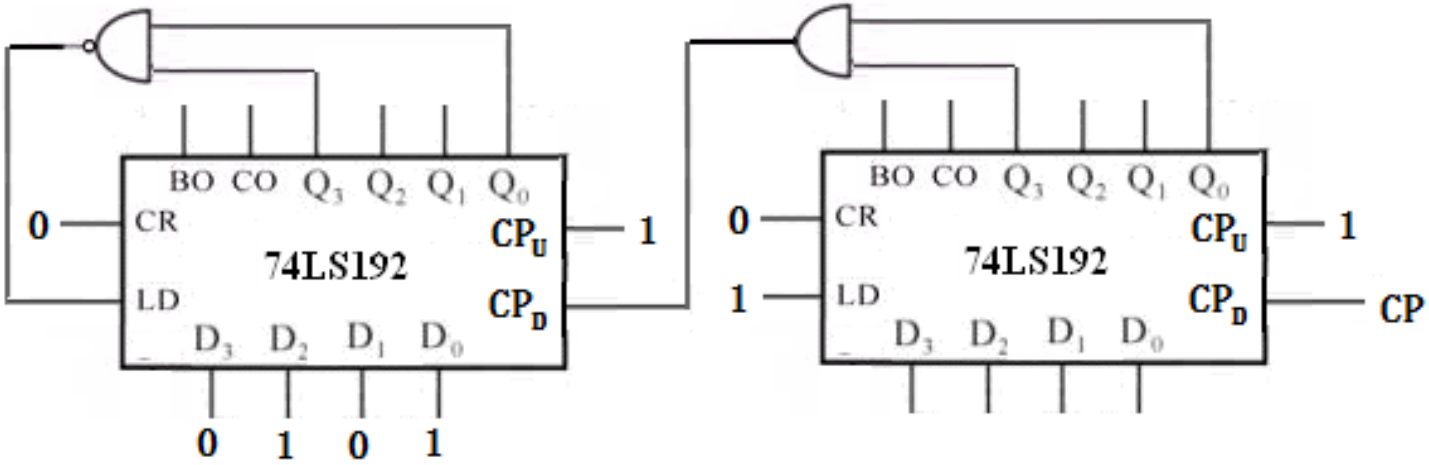
Q_D为高位，Q_A为低位

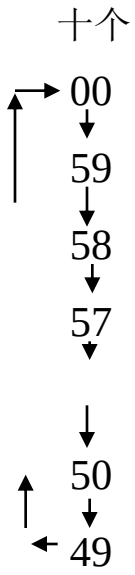
【例】用中规模集成计数器74LS192设计8421BCD编码的60进制减法计数器。$\overline{CO}=\overline{Q_3^nQ_0^n\overline{CP_U}}$ $\overline{BO}=\overline{\overline{Q_3^n}\overline{Q_2^n}\overline{Q_1^n}\overline{Q_0^n}\overline{CP_D}}$

8421BCD码十进制可逆计数器

异步清零，异步置数

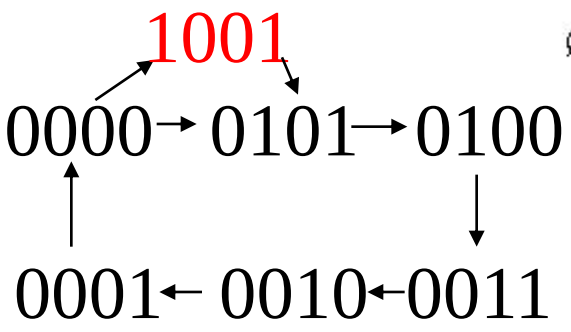


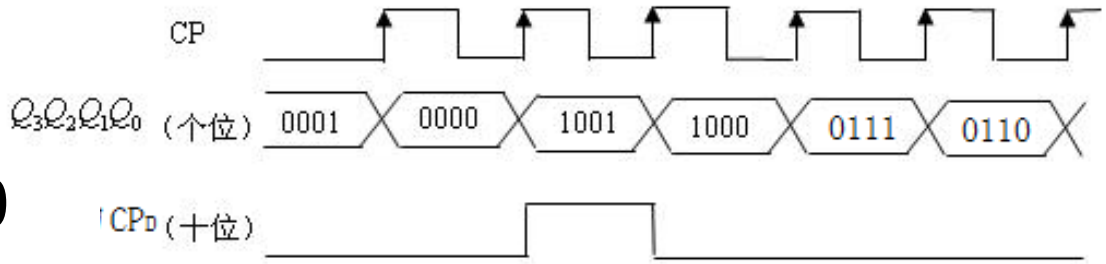


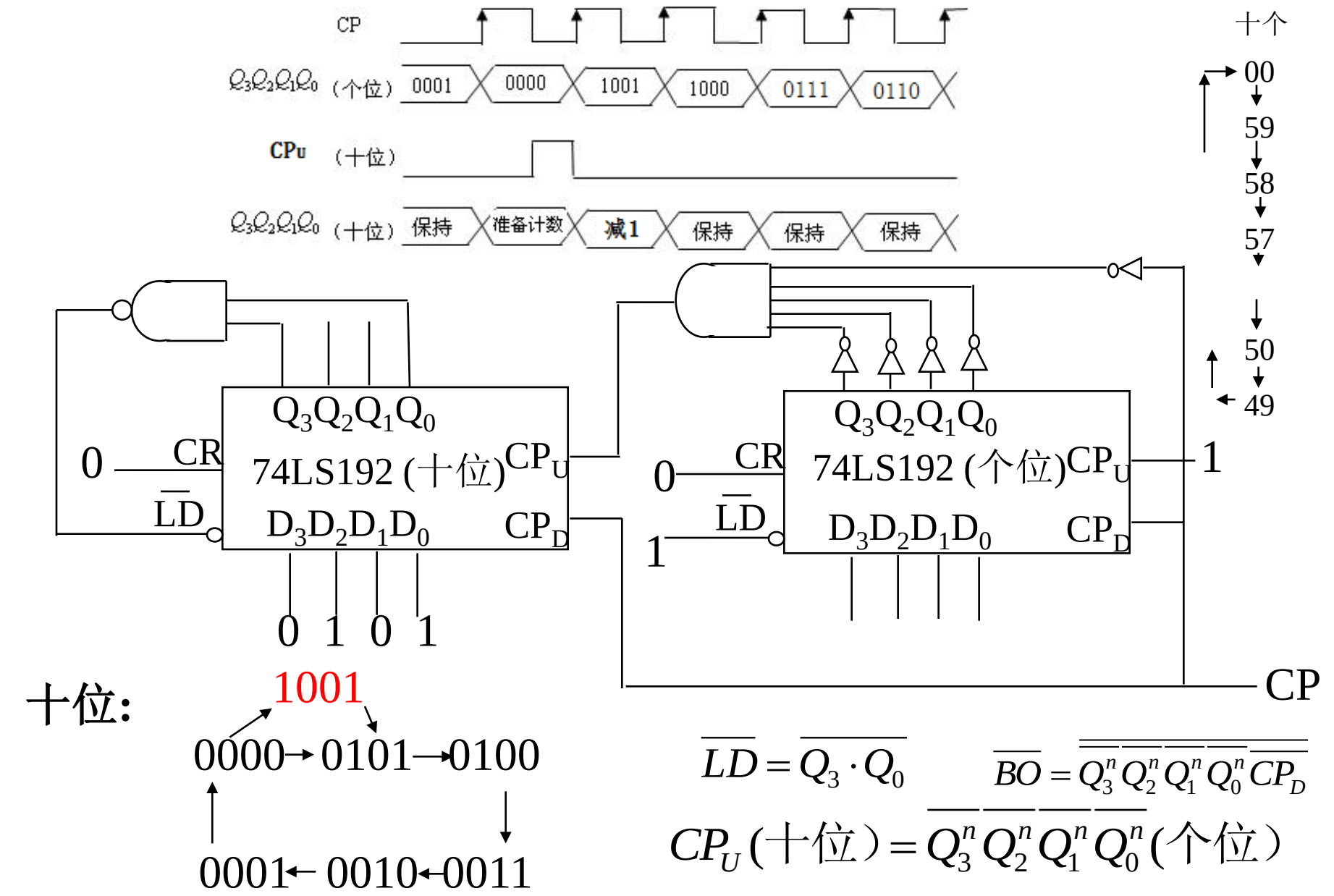


异步级联8421BCD编码的60进制减法计数器

十位：







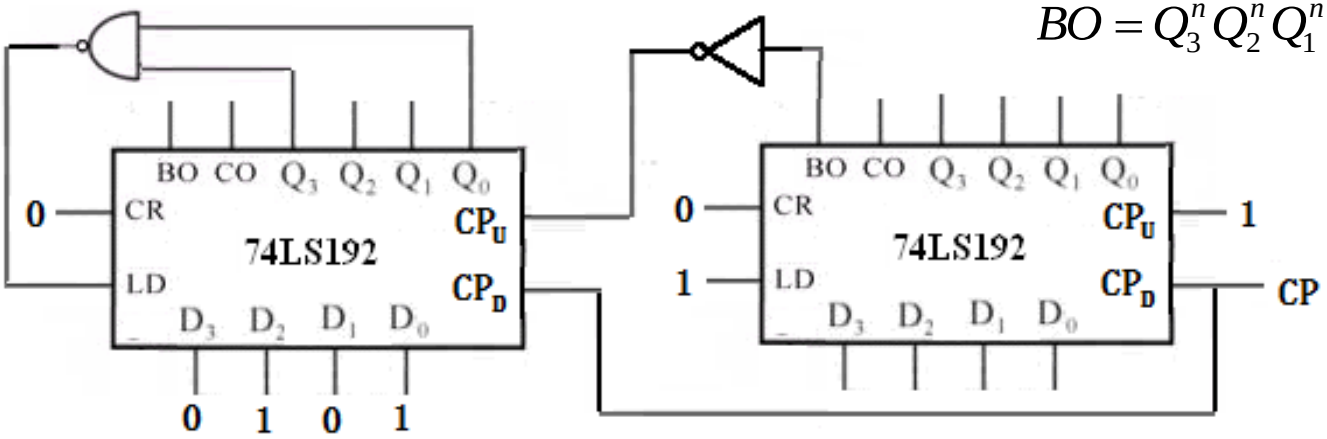
十位：

$$\overline{LD} = \overline{Q_3 \cdot Q_0} \qquad \overline{BO} = \overline{Q_3^n Q_2^n Q_1^n Q_0^n \overline{CP_D}}$$

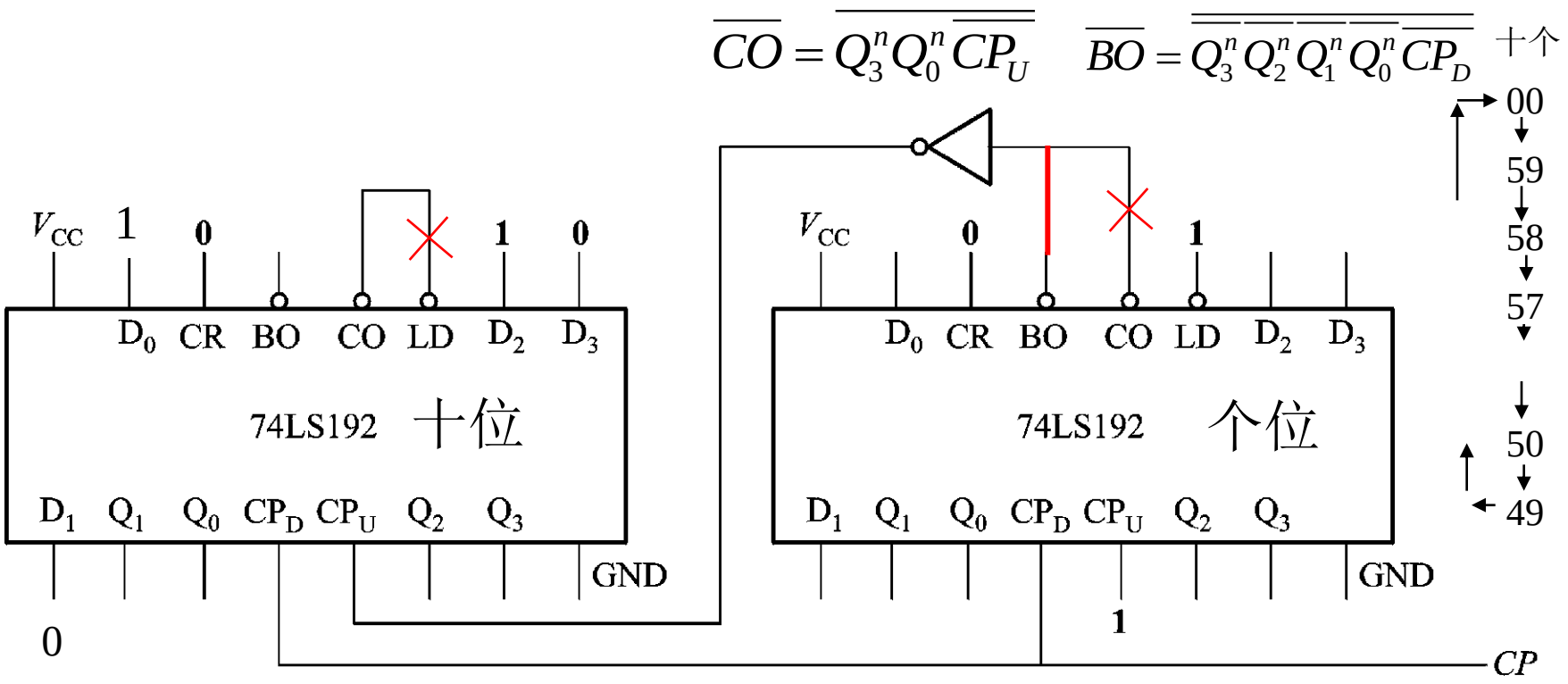
$$CP_U(\text{十位}) = \overline{Q_3^n}\,\overline{Q_2^n}\,\overline{Q_1^n}\,\overline{Q_0^n}(\text{个位})$$

同步级联8421BCD编码的60进制减法计数器

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

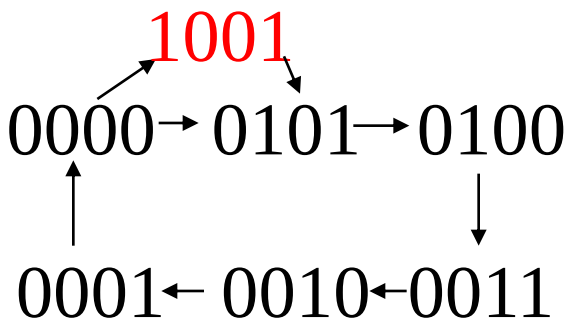


同步级联8421BCD编码的60进制减法计数器

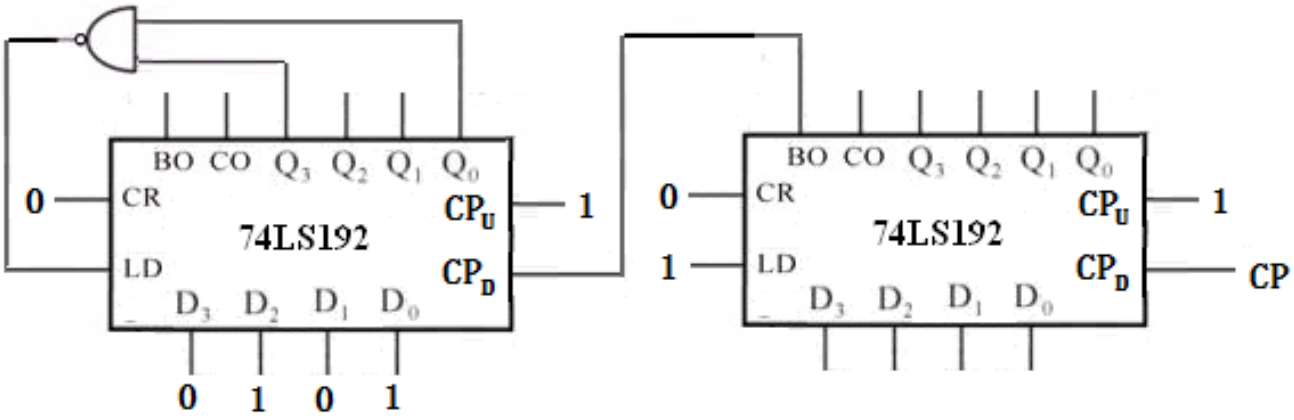


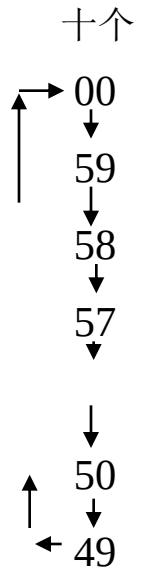
同步方式实现的8421BCD编码60进制减法计数器

十位：

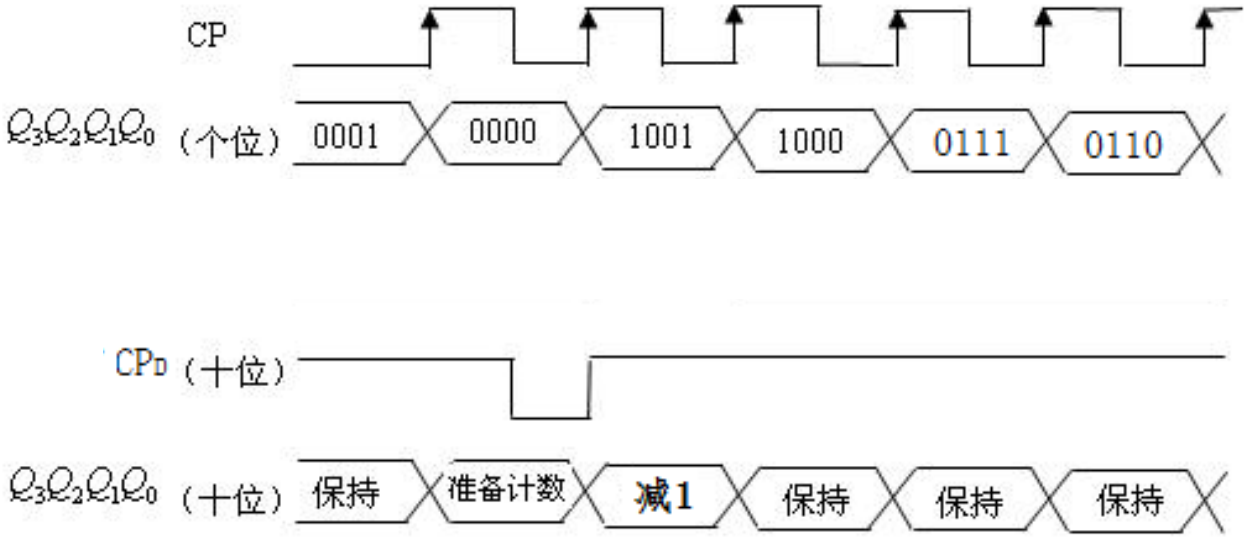


$$\overline{LD} = \overline{Q_3 \cdot Q_0}$$





利用$\overline{BO}$实现异步级联的60进制减法计数器



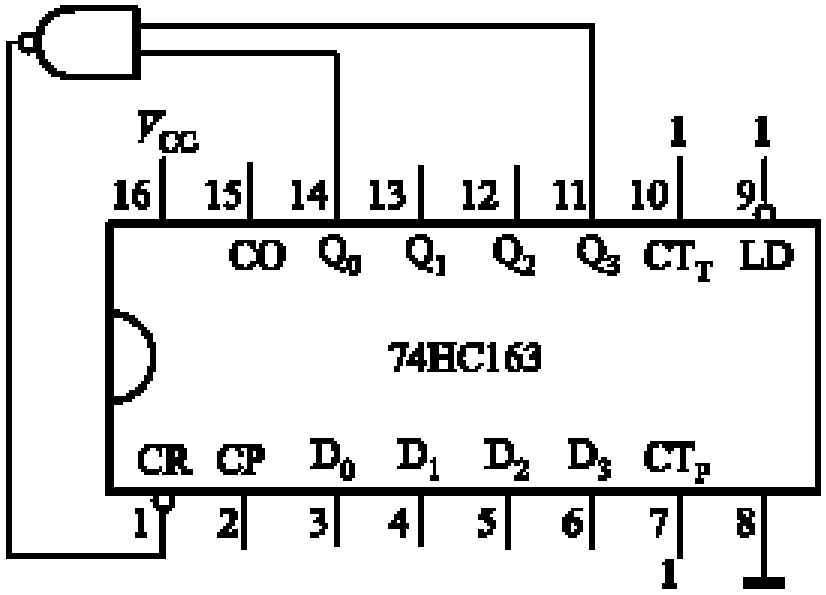
四、组建其它时序电路

用中规模集成计数器还可以组建其它时序逻辑电路，如序列脉冲发生等。

例：试用74HC163连接出一个能产生1100110001序列的脉冲发生器电路。

解：要求产生的序列需要10个状态，因此，首先把74HC163连接出一个10进制计数器，然后再通过译码器产生所要求的序列。

因74HC163是同步清零、同步置数的4位二进制加法计数器，选取前10个状态进行计数后的电路如图。

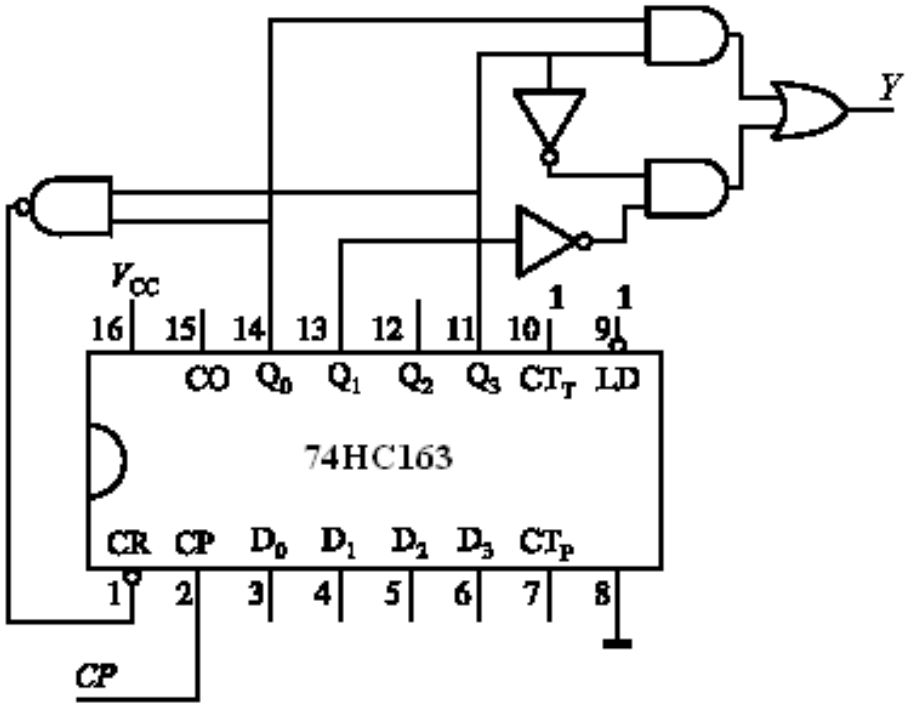


然后将10个状态按要求的输出序列1100110001译码，得到序列输出逻辑函数Y。画出电路图。

$Q_3^nQ_2^n$ \ $Q_1^nQ_0^n$	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	×	×	×	×
10	0	1	×	×

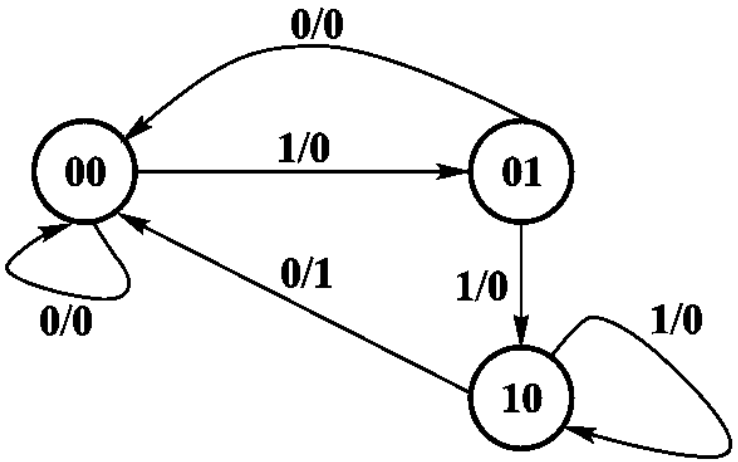
产生1100110001序列

$$Y = f(Q_3^nQ_2^nQ_1^nQ_0^n) = \overline{Q_3^n}\,\overline{Q_1^n} + Q_3^nQ_0^n$$



例：试用74HC161设计一个将输入随机序列X，检测出该输入序列中110序列的时序逻辑电路。

解：前面第5章用触发器设计过该序列脉冲检测器，其状态转换图也已清楚。因此，设计任务只要将74HC161的状态按该状态规律转换即可。



同样，若状态以自然二进制数转换时，自然计数完成，不是时，次态用置数法完成。

输入X	现态 Q^n		次态 Q^{n+1}		置数和数据端状态			输出Y
	Q_1	Q_0	Q_1	Q_0	$\overline{LD}$	D_1	D_0	
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1
1	0	0	0	1	1	×	×	0
1	0	1	1	0	1	×	×	0
1	1	0	1	0	0	1	0	0

求置数控制、数据输入、时序输出的逻辑函数。

输入 X	现态 Q^n		次态 Q^{n+1}		置数和数据端状态			输出 Y
	Q_1	Q_0	Q_1	Q_0	$\overline{LD}$	D_1	D_0	
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1
1	0	0	0	1	1	×	×	0
1	0	1	1	0	1	×	×	0
1	1	0	1	0	0	1	0	0

$\overline{LD}$ $Q_1^nQ_0^n$ / X	00	01	11	10
0	0	0	×	0
1	1	1	×	0

$$\overline{LD} = X\overline{Q_1^n}$$

D_1 $Q_1^nQ_0^n$ / X	00	01	11	10
0	0	0	×	0
1	×	×	×	1

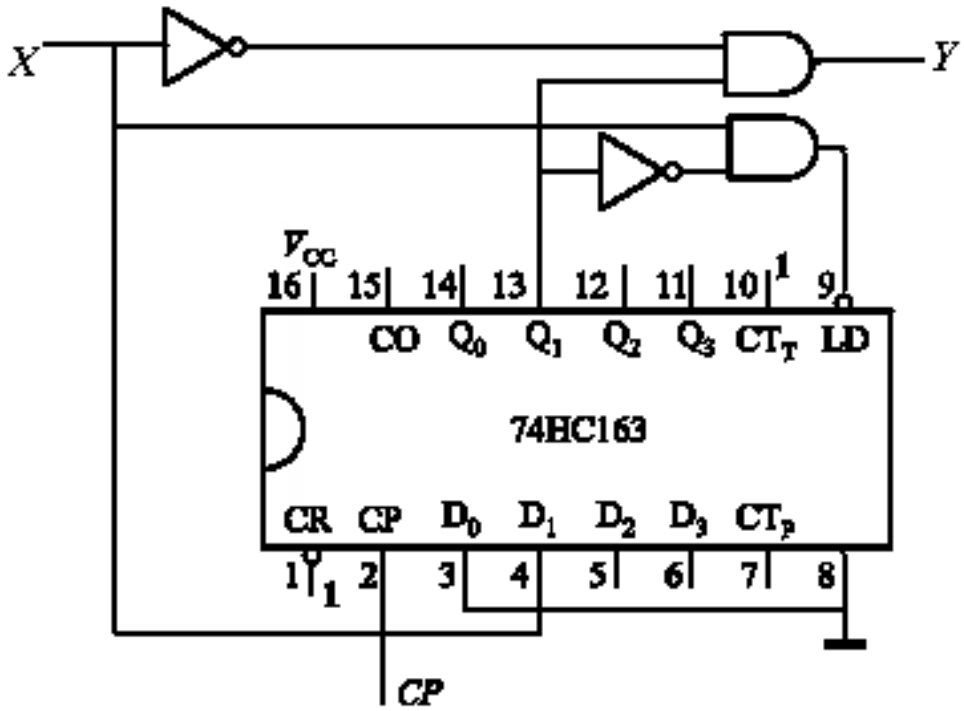
$$D_1 = X$$

D_0 $Q_1^nQ_0^n$ / X	00	01	11	10
0	0	0	×	0
1	0	0	×	0

$$D_0 = 0$$

Y $Q_1^nQ_0^n$ / X	00	01	11	10
0	0	0	×	1
1	0	0	×	0

$$Y = \overline{X}Q_1^n$$


X
Y
V_{CC}
16
15
14
13
12
11
10
1
9
CO
Q_0
Q_1
Q_2
Q_3
CT_T
LD
74HC163
CR
CP
D_0
D_1
D_2
D_3
CT_P
1
1
2
3
4
5
6
7
8
CP

用VHDL语言描述时序电路逻辑功能

用VHDL语言描述逻辑功能的方法有两种：行为描述法和结构描述法。

行为描述法，只管逻辑功能能否实现，其它由EDA自动完成，描述方法简单。

结构描述法注重实现逻辑功能的结构，该语言的设计结果与硬件电路有明确的对应关系，能体现设计者的设计思想。

例1： 试用VHDL语言设计一个具有异步清零8421BCD编码的十进制加法计数器。

解： 采用VHDL语言的行为描述法：

```
library IEEE;
                              --库说明
use IEEE.STD_LOGIC_1164.ALL;
                    --引用库中包
use IEEE.STD_LOGIC_ARITH.ALL; use
IEEE.STD_LOGIC_UNSIGNED.ALL;
entity CNTbev is--实体说明
    Port ( CRn : in  STD_LOGIC;                --端口说明
           CP : in  STD_LOGIC;
           Carry : out  STD_LOGIC;
           QQ : out  STD_LOGIC_VECTOR (3 downto 0));
```

```
end CNTbev;
architecture Behavioral of CNTbev is        --结构体
signal tmp: STD_LOGIC_VECTOR (3 downto 0);
begin
process (CP,CRn)                            --过程函数
begin
   if CRn='0' then                           --异步清0
      tmp <= (others => '0');
   elsif CP='1' and CP'event then            --上升沿触发
      if tmp ="1001" then                    --同步计数
         tmp<="0000";
      elsif tmp<=tmp+1;
  end if;
   end if;
 QQ<=tmp;
 Carry<=tmp(3) and tmp(0);

 end process;
 end Behavioral;                            --结束行为描述
```

十进制计数器的VHDL语言结构描述法：

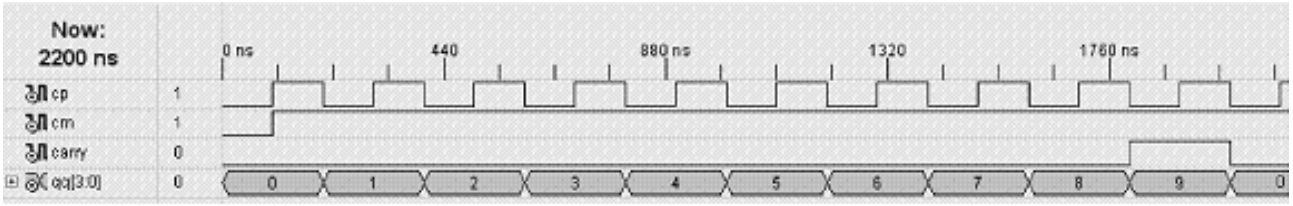
```
library IEEE;                                    --库说明
use IEEE.STD_LOGIC_1164.ALL;                     --引用库中包
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity CNT10 is                                  --实体说明
   Port ( CP : in  STD_LOGIC;                    --端口说明
          CRn : in  STD_LOGIC;
          Carry : out  STD_LOGIC;
          QQ : inout  STD_LOGIC_VECTOR (3 downto 0));
```

```
end CNT10;
architecture rtl of CNT10 is                    --结构体
component myJKFF     --JK FF元件说明
        port(SETn,CLRn,J,K,CLK: in      STD_LOGIC;
                                Q,Qn: out       STD_LOGIC);
end component;
signal Qb:STD_LOGIC_VECTOR(3 downto 0):="1111";  --中间信号说明
signal j:STD_LOGIC_VECTOR(3 downto 0):="0000";
signal k:STD_LOGIC_VECTOR(3 downto 0):="0000";
signal logic_one :STD_LOGIC;
begin
        logic_one<='1';
        j(0)<='1';                              --各触发器驱动方程
```

```
        k(0)<='1';
        j(1)<=QQ(0) and Qb(3);
        k(1)<=QQ(0) ;
        j(2)<=QQ(0) and QQ(1);
        k(2)<=QQ(0) and QQ(1);
        j(3)<=QQ(0) and QQ(1) and QQ(2);
        k(3)<=QQ(0) ;
        Carry<=QQ(0) and QQ(3);
  jkff0:myJKFF                    --FF0触发器
  port
map(SETn=>logic_one,CLRn=>CRn,J=>j(0),K=>k(0),CLK=>C
P,Q=>QQ(0),Qn=>Qb(0));
        jkff1:myJKFF              --FF1触发器
  port
map(SETn=>logic_one,CLRn=>CRn,J=>j(1),K=>k(1),CLK=>C
P,Q=>QQ(1),Qn=>Qb(1));
        jkff2:myJKFF              --FF2触发器
```

```
port
map(SETn=>logic_one,CLRn=>CRn,J=>j(2),K=>k(2),CLK=
>CP,Q=>QQ(2),Qn=>Qb(2));
        jkff3:myJKFF                --FF3触发器
port
map(SETn=>logic_one,CLRn=>CRn,J=>j(3),K=>k(3),CLK=
>CP,Q=>QQ(3),Qn=>Qb(3));
end rtl;
                                    --结束结构描述
```

两种仿真结果图：



例2：试用VHDL语言描述110序列脉冲检测器逻辑功能。

解：设110检测器的四个状态分别为s0,s1,s2,s3，则该序列脉冲发生器的一种VHDL语言描述为：

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity SEQ is    Port ( X : in  STD_LOGIC;
           CLK : in  STD_LOGIC;
           Z : out  STD_LOGIC);
end SEQ;
```

```
architecture Behavioral of SEQ is
   type STATE_TYPE IS (s0, s1,s2,s3);
   signal  state      : STATE_TYPE := s0;
begin
process (X,CLK)
begin
   if CLK='1' and CLK'event then
            case state is
                     when s0=>
                              if (X='1') then state <= s1;
                              else state <= s0;
                              end if;
                     when s1=>
                              if (X='1') then state <= s2;
                              else state <= s0;
```

```
                    end if;
            when s2=>
                    if (X='0') then state <= s3;
                    else state <= s2;
                    end if;
            when s3=>
                    if (X='1') then state <= s1;
                    else state <= s0;
                    end if;
        end case;
    end if;
  end process;
  Z <= '1' when  state = s3 else '0';
end Behavioral;
```

仿真结果图：