2020-2021学年秋季学期

第六部分-A
时序逻辑设计

授课团队：宋威
助　　教：薛子涵

- 数字电路
  - 组合逻辑电路

    Combinational Logic

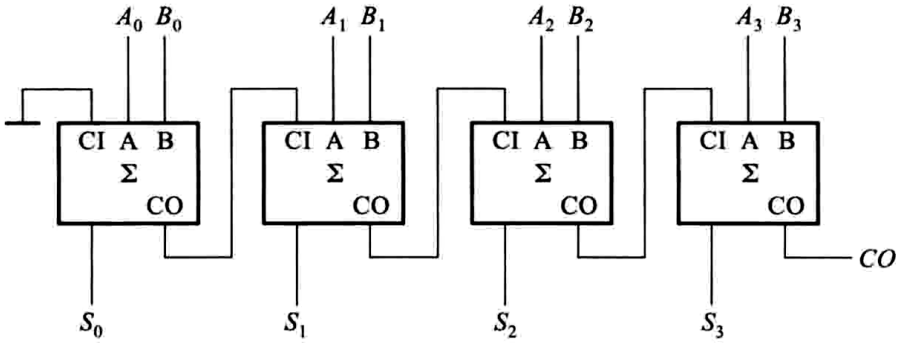
    在任意时刻，电路的输出仅决定于该时刻的输入，与电路原来的状态无关。
  - 时序逻辑电路

    Sequential Logic

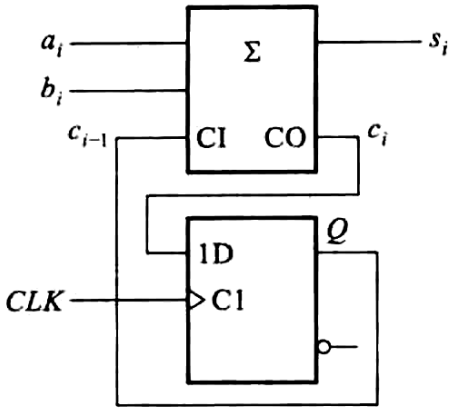
    电路的输出不仅受该时刻输入的影响，还受电路原来状态的影响（包含内部存储）。
- 时序逻辑电路
  - 通过存储状态可以用时间换空间
  - 状态存储扩大了电路的状态空间，可以用以描述更复杂的电路行为。

串行加法器

- N位串行加法器
  - 组合逻辑电路
    - N个全加器
    - 立即出结果

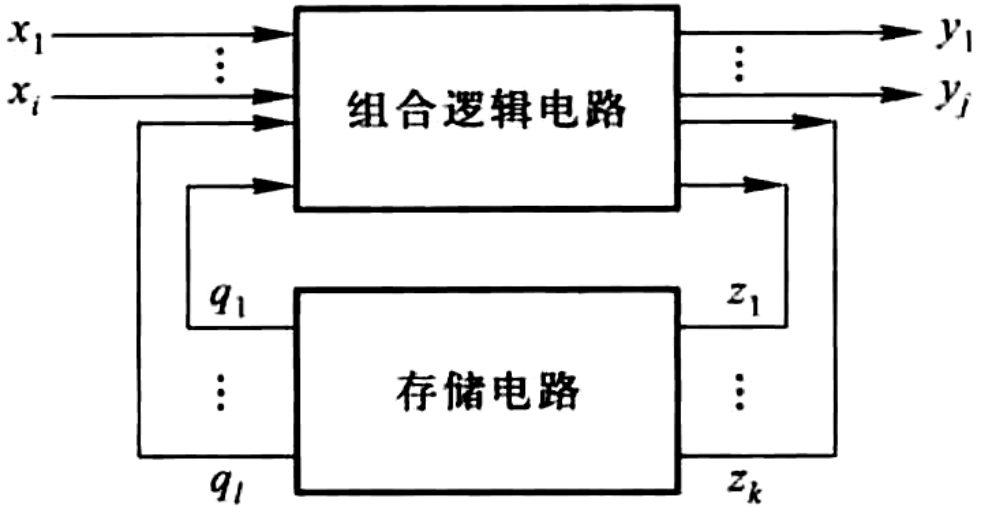


  - 时序逻辑电路
    - 1个全加器加上一个DFF
    - N个周期出结果



- 时序逻辑电路用时间换空间

时序逻辑电路的系统方程



$$Y = F(X, Q)$$
$$Z = G(X, Q)$$
$$Q^* = H(Z, Q)$$

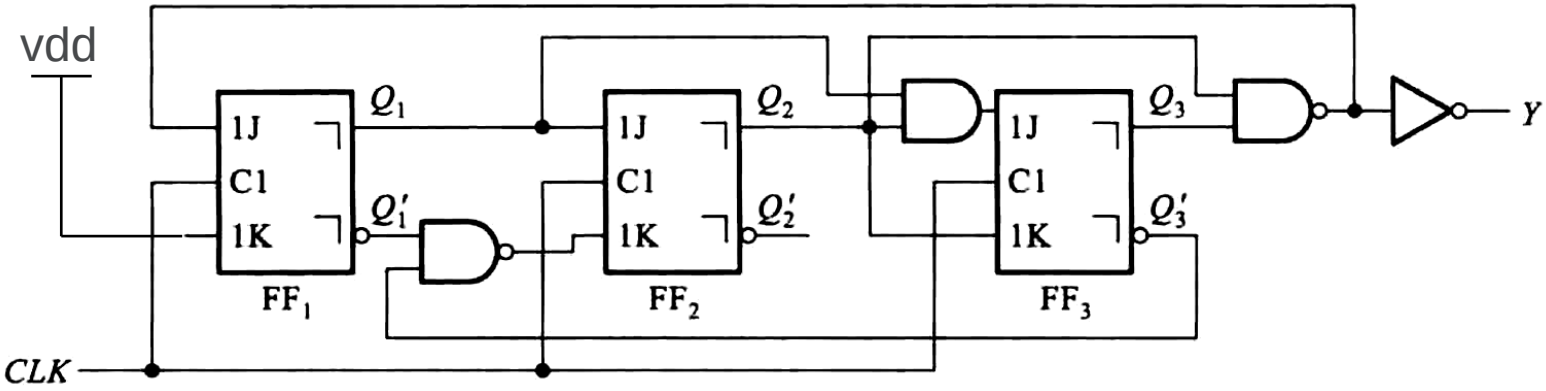
时序逻辑电路的分析方法

- 确定输出的逻辑方程：$Y = F(X, Q)$
- 分析每个触发器的驱动逻辑方程：$Z = G(X, Q)$
- 代入触发器的特性方程：$Q^* = H(Z, Q)$

当整个时序逻辑电路的系统方程被确定，电路的逻辑功能就被完全确定。

- 米利型(Mealy): $Y = F(X, Q)$
- 穆尔型(Moore): $Y = F(Q)$
  - 穆尔型时序电路为米利型时序电路的特殊形式

时序逻辑电路的功能分析示例 —触发器驱动方程



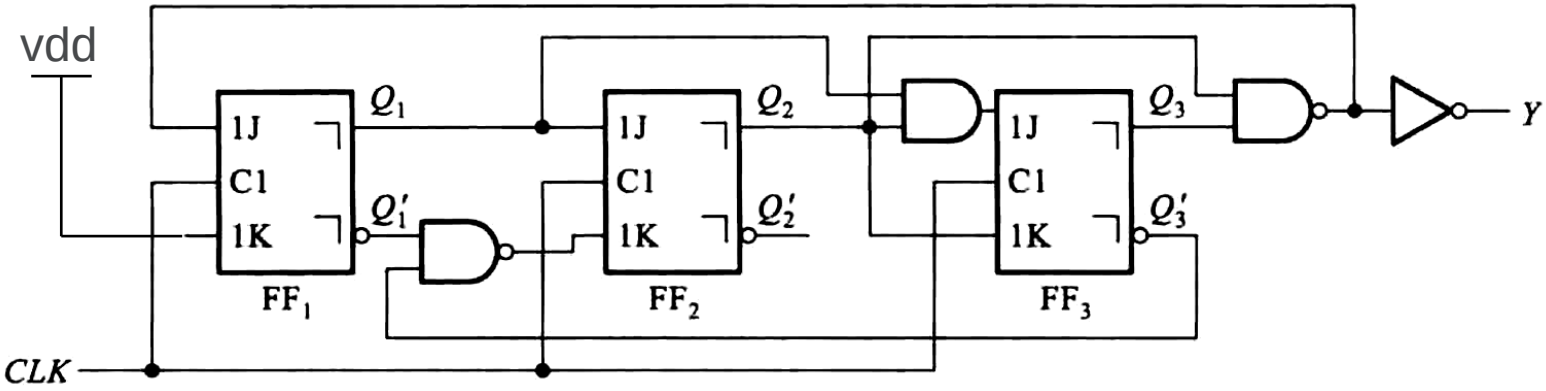
$$J_1 = \overline{Q_2 Q_3} \qquad K_1 = 1$$

$$J_2 = Q_1 \qquad K_2 = Q_1 + Q_3$$

$$J_3 = Q_1 Q_2 \qquad K_3 = Q_2$$

$$Z = G(X, Q)$$

时序逻辑电路的功能分析示例 —触发器特性方程



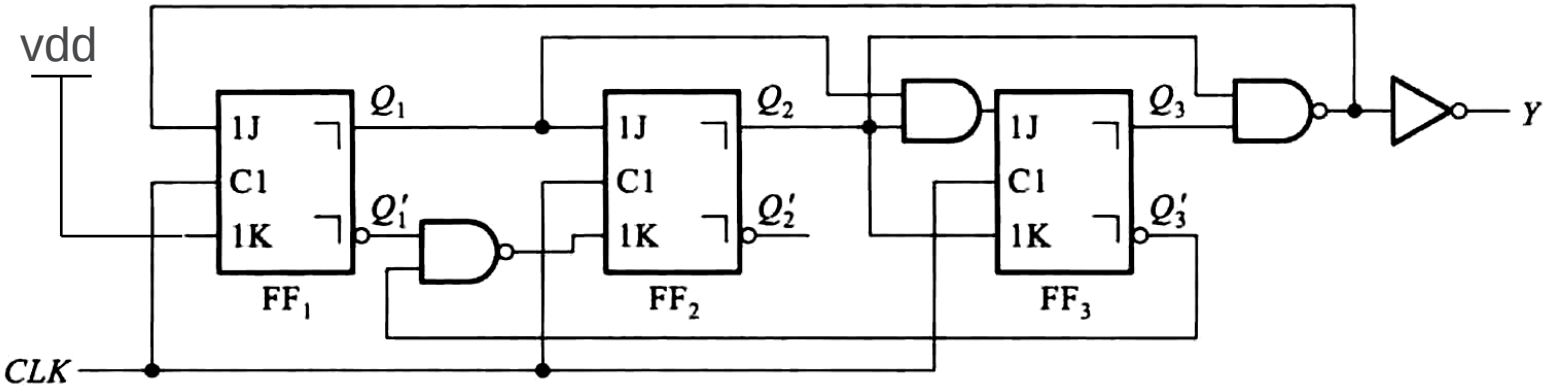
$$Q_1^* = \overline{Q_2 Q_3} \cdot \overline{Q_1}$$

$$Q_2^* = Q_1\overline{Q_2} + \overline{Q_1}Q_2\overline{Q_3}$$

$$Q_3^* = Q_1 Q_2 \overline{Q_3} + \overline{Q_2} Q_3$$

$$Q^* = H(Z, Q) = J\overline{Q} + \overline{K}Q$$

时序逻辑电路的功能分析示例 —输出方程

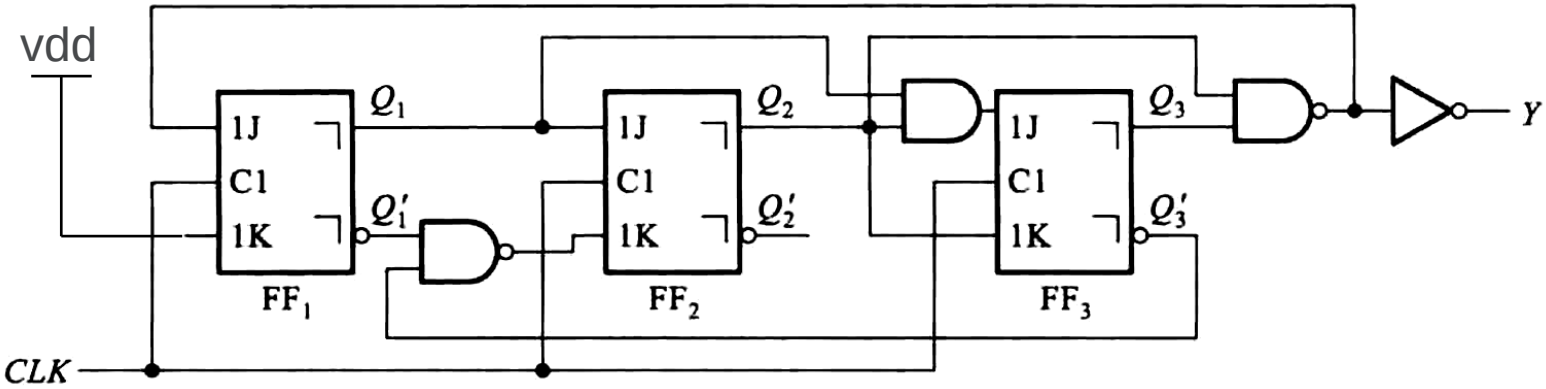


$$Y = Q_2 Q_3$$

穆尔型时序逻辑电路

$$Y = F(Q)$$

时序逻辑电路的功能分析示例 —系统方程

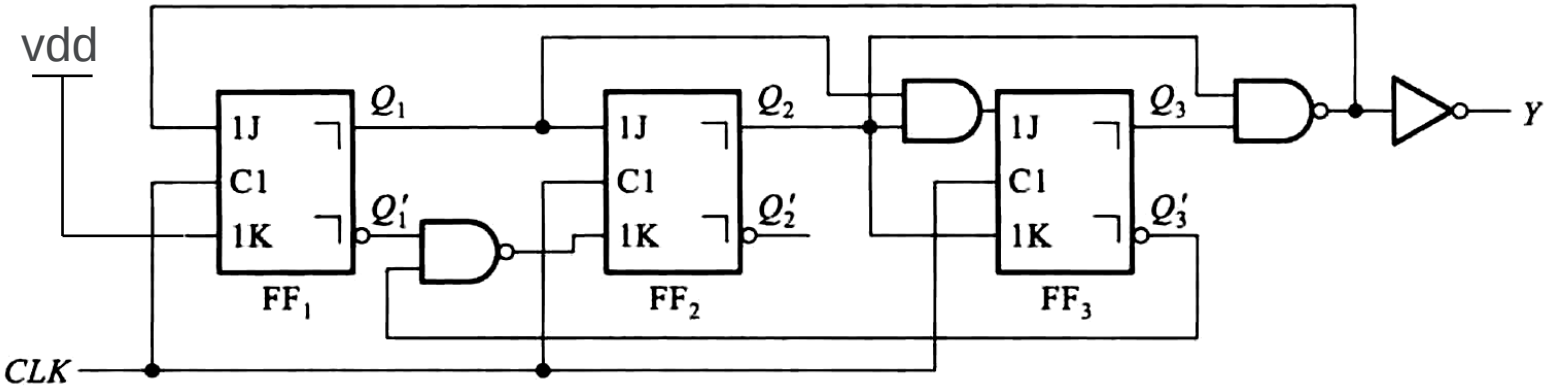


$$Q_1^* = \overline{Q_2 Q_3} \cdot \overline{Q_1}$$
$$Q_2^* = Q_1 \overline{Q_2} + \overline{Q_1} Q_2 \overline{Q_3}$$
$$Q_3^* = Q_1 Q_2 \overline{Q_3} + \overline{Q_2} Q_3$$

$$Y = Q_2 Q_3$$

我们仍然不太明白这个电路的功能。

时序逻辑电路的功能分析示例 —状态转换表



$Q_3Q_2Q_1$	$Q_3^*Q_2^*Q_1^*$	Y
000	001	0
001	010	0
010	011	0
011	100	0
100	101	0
101	110	0
110	000	1
111	000	0

7循环计数器

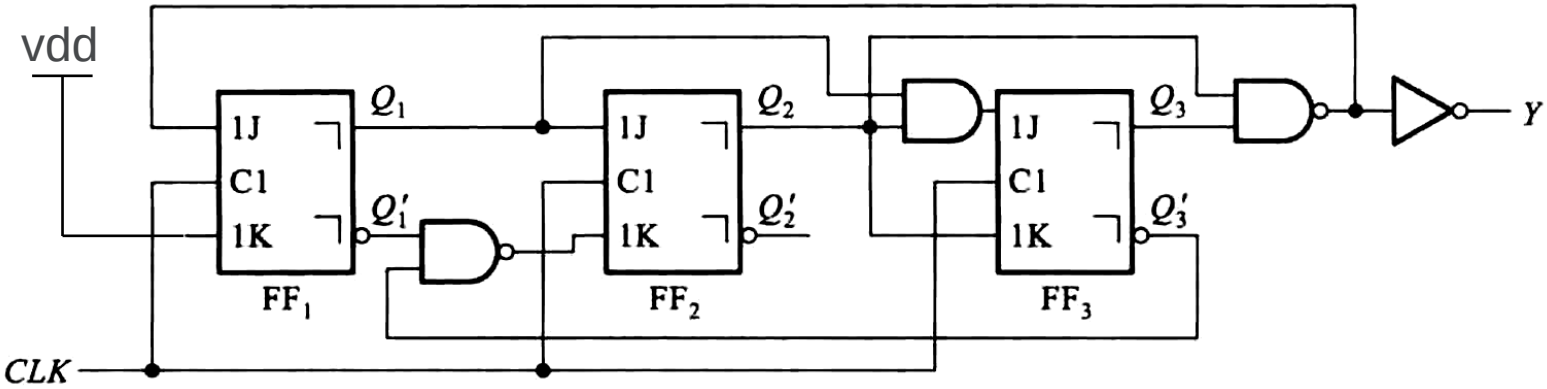
$$Y = Q_2Q_3$$

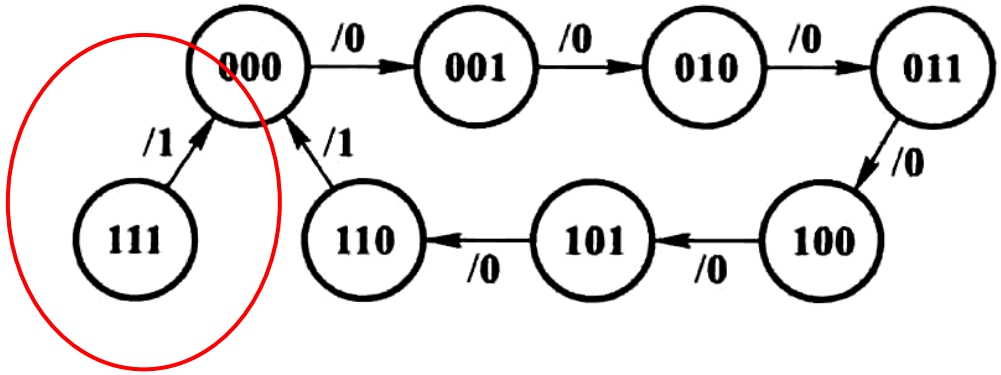
$$Q_1^* = \overline{Q_2Q_3} \cdot \overline{Q_1}$$

$$Q_2^* = Q_1\overline{Q_2} + \overline{Q_1}Q_2\overline{Q_3}$$

$$Q_3^* = Q_1Q_2\overline{Q_3} + \overline{Q_2}Q_3$$

时序逻辑电路的功能分析示例 —状态转换图





状态

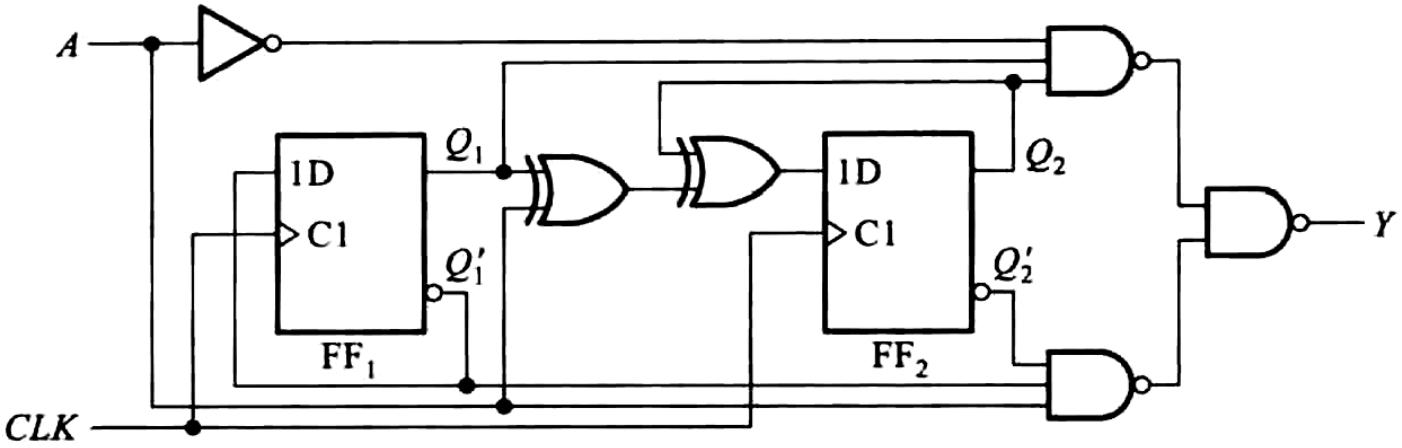
$Q_3Q_2Q_1$

/Y

输入 / 输出

无效状态向有效状态迁移
可自启的时序逻辑电路

D触发器构成的时序逻辑电路示例 —系统方程

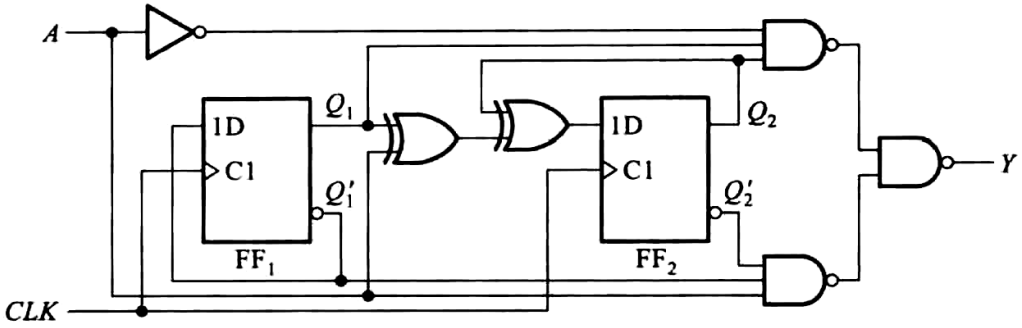


$$Q_1^* = \overline{Q_1}$$

$$Q_2^* = A \oplus Q_1 \oplus Q_2$$

$$Y = \overline{A} Q_1 Q_2 + A\overline{Q_1} \cdot \overline{Q_2}$$

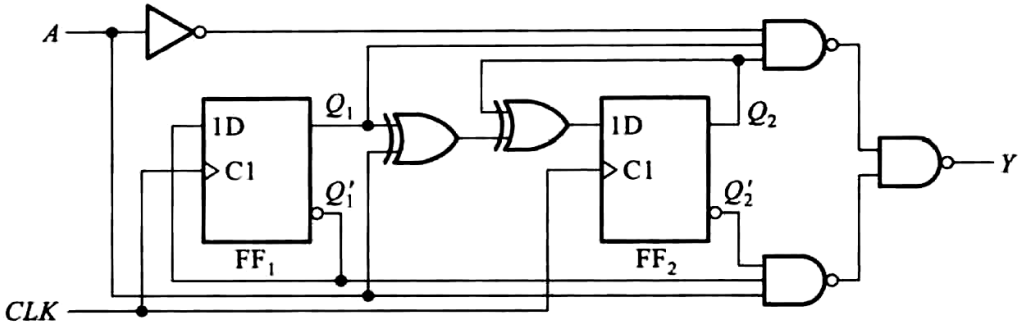
D触发器构成的时序逻辑电路示例 —状态转换表



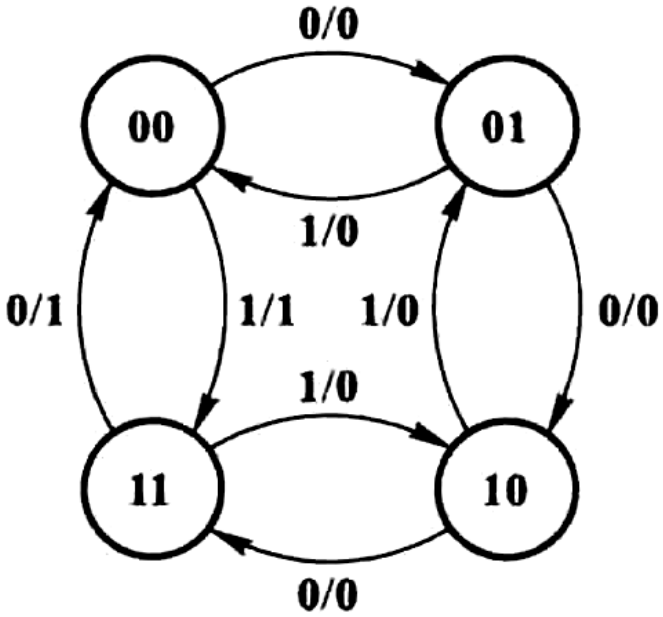
$$Q_1^* = \overline{Q_1}$$
$$Q_2^* = A \oplus Q_1 \oplus Q_2$$
$$Y = \overline{A}Q_1Q_2 + A\overline{Q_1} \cdot \overline{Q_2}$$

A	Q_2Q_1	$Q_2^*Q_1^*$	Y
0	00	01	0
0	01	10	0
0	10	11	0
0	11	00	1
1	00	11	1
1	01	00	0
1	10	01	0
1	11	10	0

D触发器构成的时序逻辑电路示例 —状态转换图



$$Q_1^* = \overline{Q_1}$$
$$Q_2^* = A \oplus Q_1 \oplus Q_2$$
$$Y = \overline{A}Q_1Q_2 + A\overline{Q_1} \cdot \overline{Q_2}$$



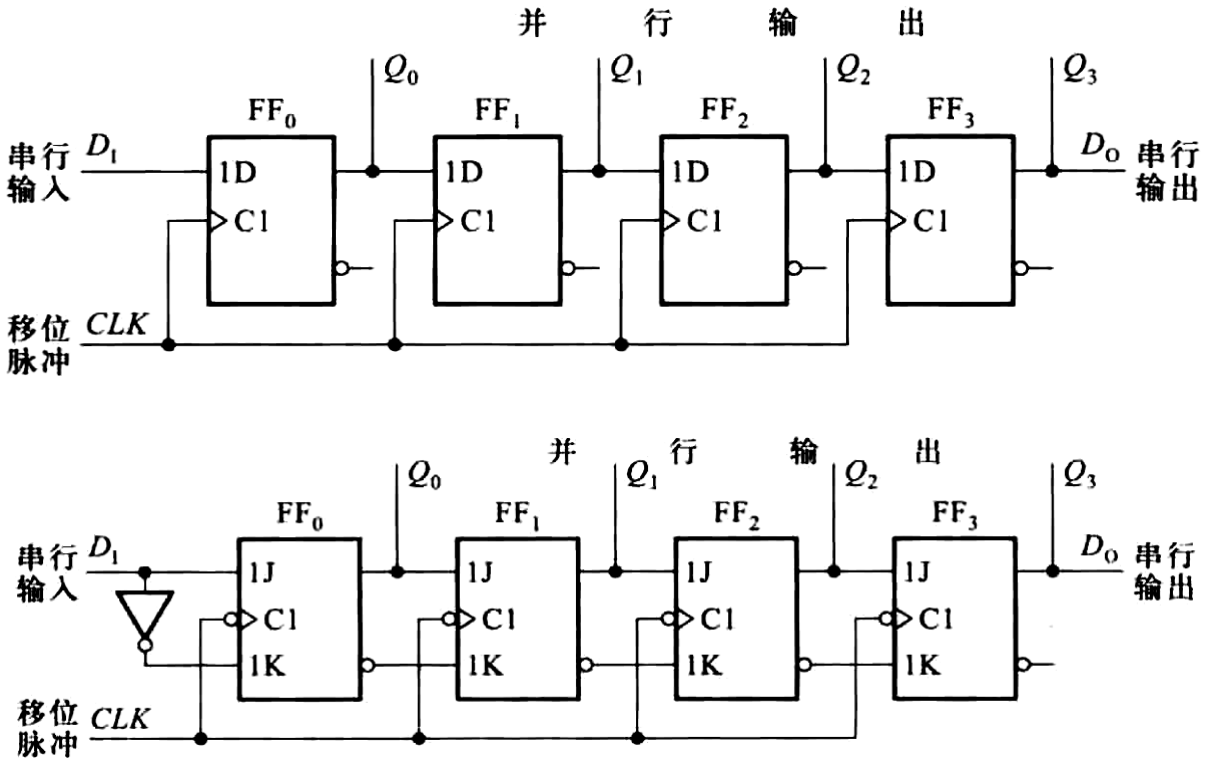
2位双向计数器

时序逻辑电路总结

- 时序逻辑电路
  - 电路包含内部状态存储
  - 输出和内部状态受系统上一时刻状态的影响
  - 状态空间复杂
  - 用时间换取电路空间
- 时序逻辑电路的系统方程
  - 时序电路的系统方程：
    - 触发器驱动方程
    - 触发器特性方程
    - 输出方程
  - 时序电路的状态
    - 状态转换表
    - 状态转换图
    - 一般来说，时序逻辑电路的状态是有意义的（人为设计）

关于流程图和时序图的部分，自行看书自学。

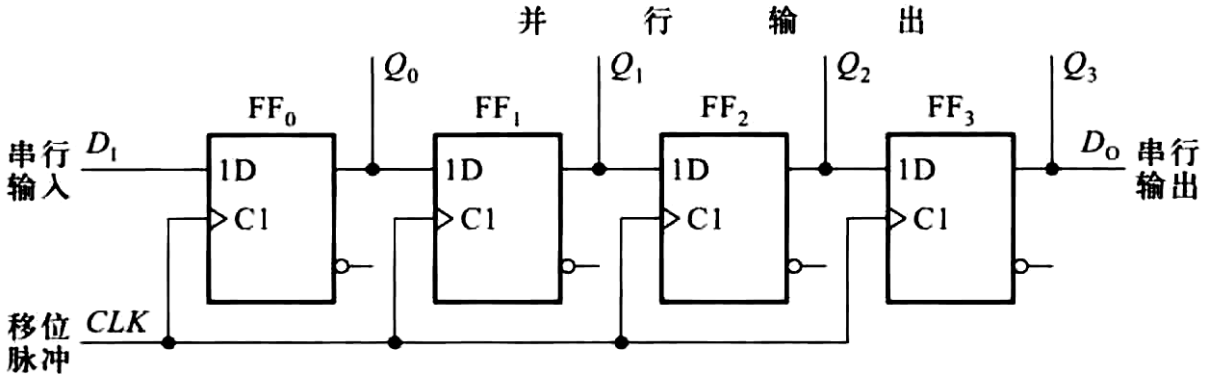
常用时序逻辑 —移位寄存器

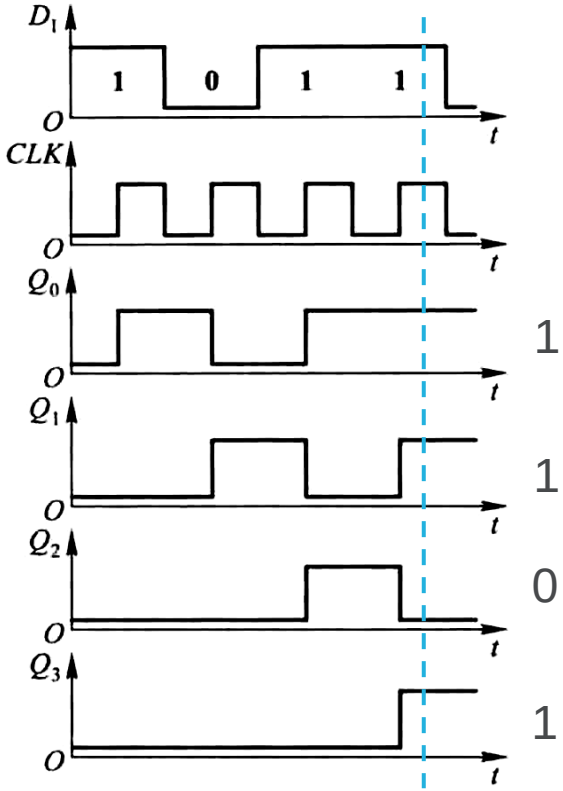


$$Q_i^* = Q_{i-1}$$
$$Y_{para} = Q$$
$$Y_{seri} = Q_{n-1}$$

不太适合用状态转换图描述，空间爆炸。
系统方程反而比较容易理解。

常用时序逻辑 —移位寄存器

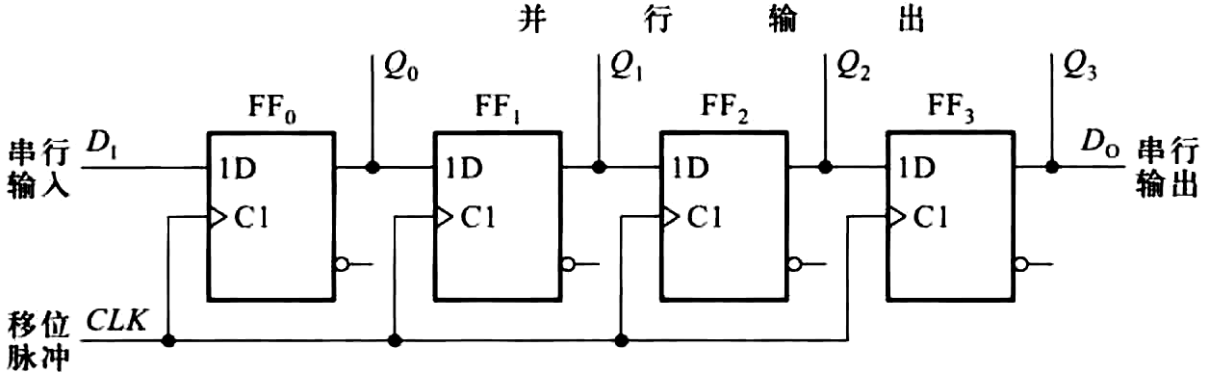




用途：

* 串并转换
* 延时输出

常用时序逻辑 —移位寄存器




```
module shifter(
  input clk, din,
  output [3:0] q,
  output dout);

  reg [3:0] q;
  always @(posedge clk)
    q <= {q[2:0],din}; // 移位操作

  assign dout = q[3];
endmodule
```

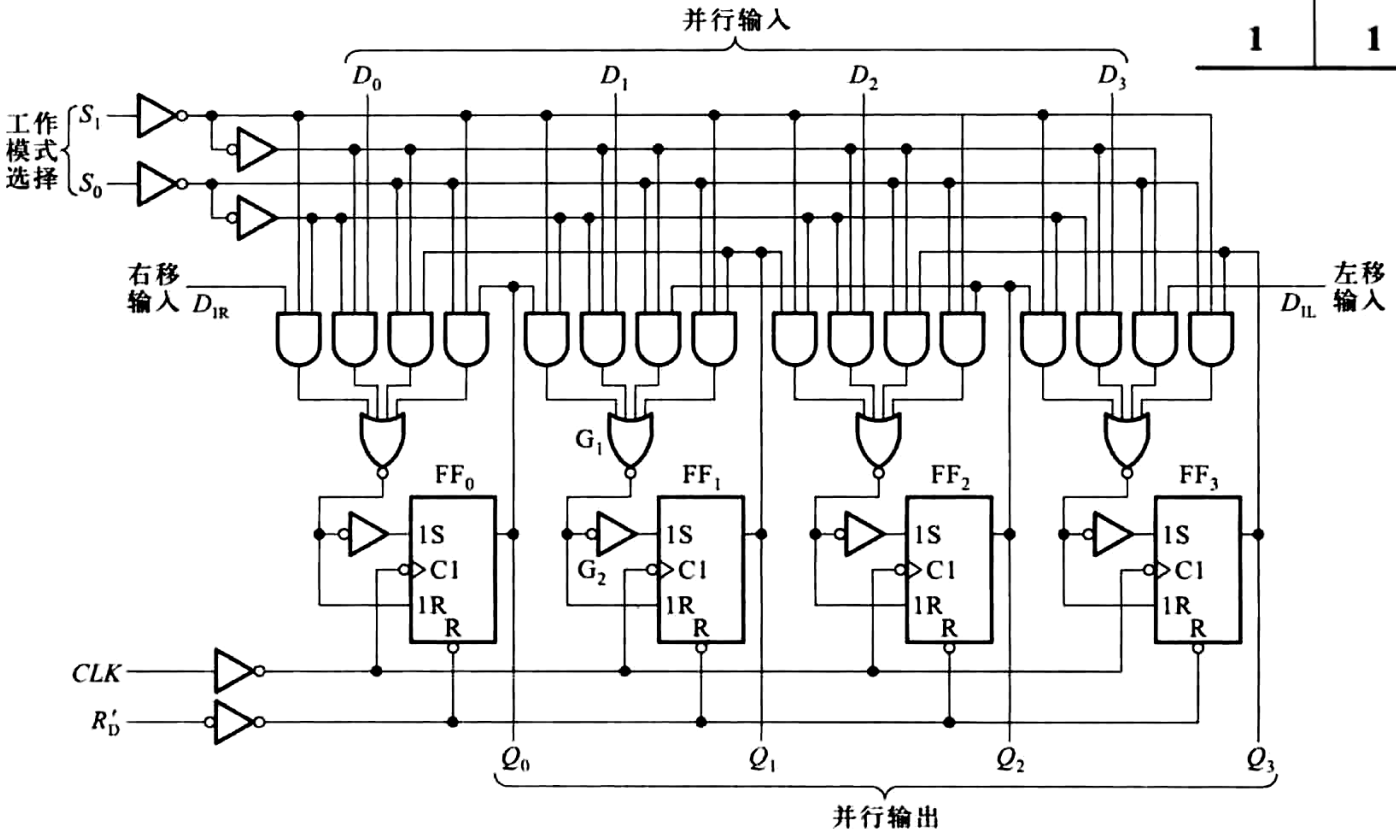
问题，系统刚刚上电的时候，输出q的值是多少？

常用时序逻辑 —移位寄存器

74HC194A，双向移位寄存器

74HC194A 的功能表

R'_D	S_1	S_0	工作状态
0	×	×	置零
1	0	0	保持
1	0	1	右移
1	1	0	左移
1	1	1	并行输入



常用时序逻辑 —计数器

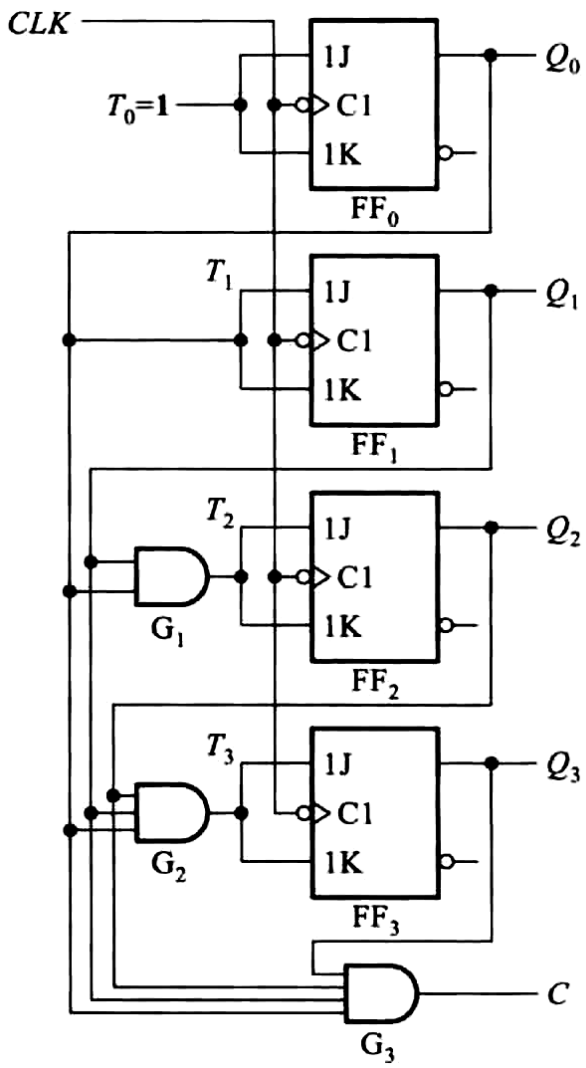
$$T_i = Q_{i-1} \cdot Q_{i-2} \cdot \cdots \cdot Q_1 \cdot Q_0$$

$$= \prod_{j=0}^{i-1} Q_j \qquad (i = 1, 2, \cdots, n-1)$$

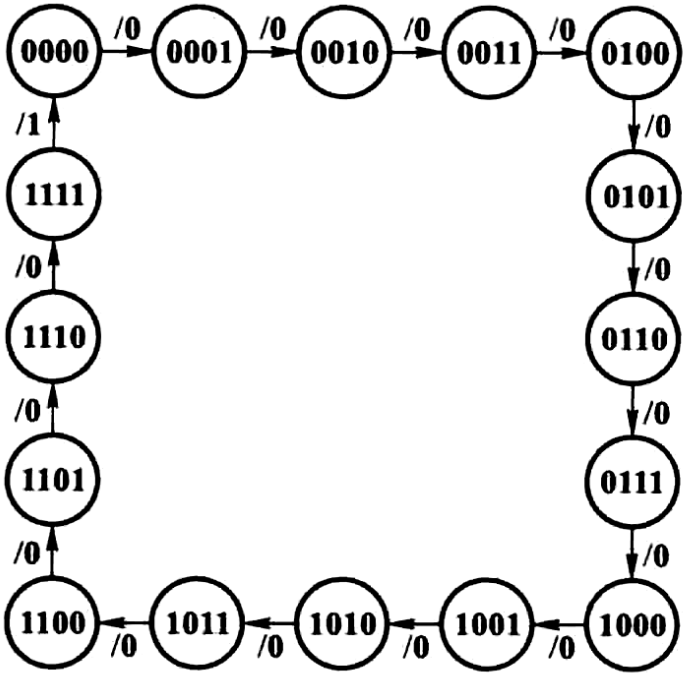
$$T_0 = 1$$
$$T_1 = Q_0$$
$$T_2 = Q_0 Q_1$$
$$T_3 = Q_0 Q_1 Q_2$$

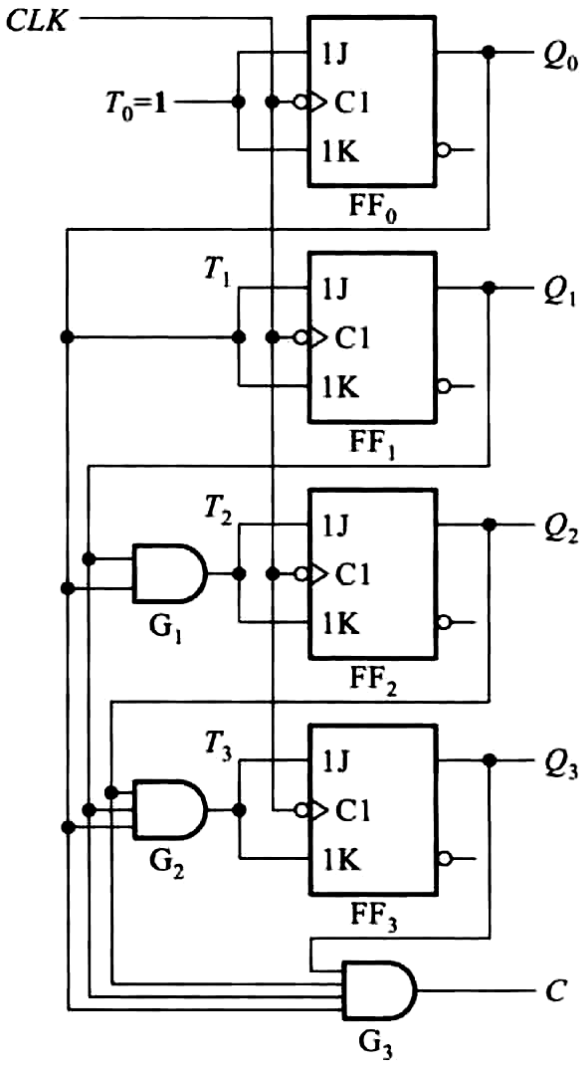
$$Q_0^* = \overline{Q_0}$$
$$Q_1^* = Q_0 \overline{Q_1} + \overline{Q_0} Q_1$$
$$Q_2^* = Q_0 Q_1 \overline{Q_2} + \overline{Q_0 Q_1} Q_2$$
$$Q_3^* = Q_0 Q_1 Q_2 \overline{Q_3} + \overline{Q_0 Q_1 Q_2} Q_3$$

$$C = Q_0 Q_1 Q_2 Q_3$$

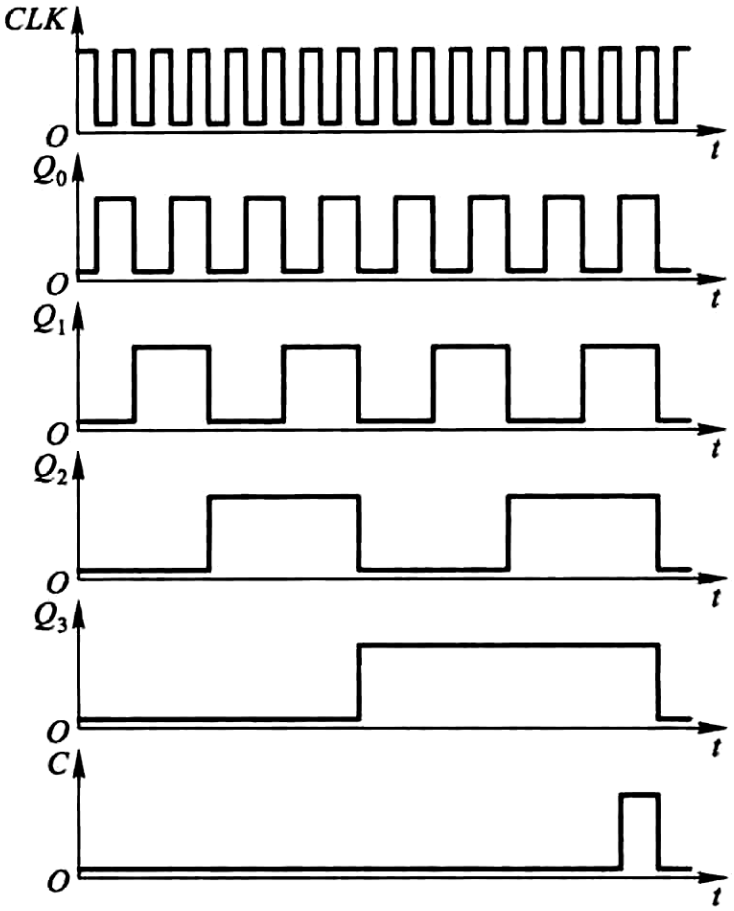


常用时序逻辑 —计数器

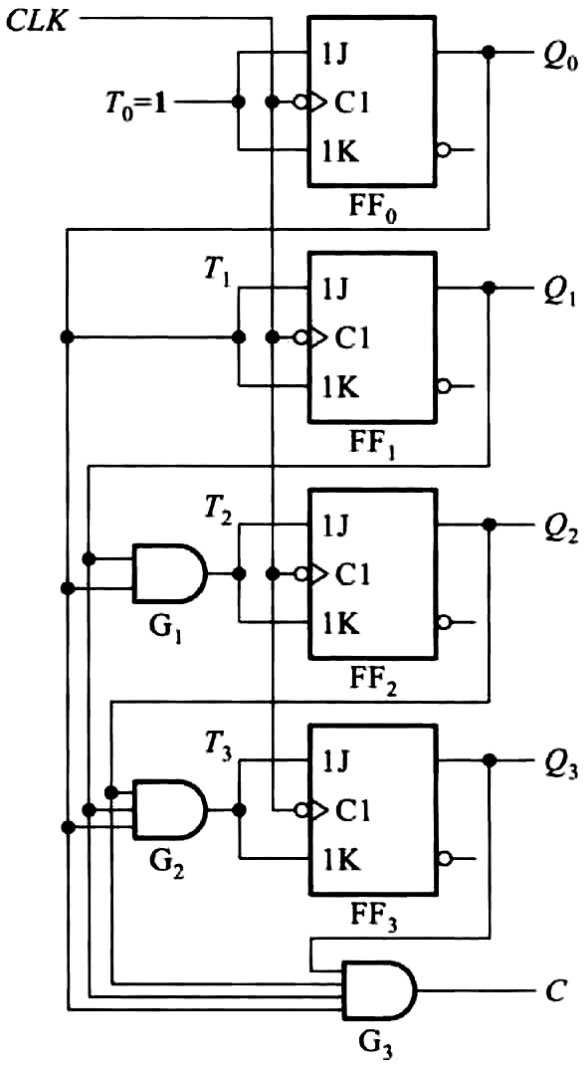

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
/0
/1


CLK
T_0=1
1J
C1
1K
FF_0
Q_0
T_1
FF_1
Q_1
T_2
G_1
FF_2
Q_2
T_3
G_2
FF_3
Q_3
G_3
C

常用时序逻辑 —计数器



计数器输出的分频效应。

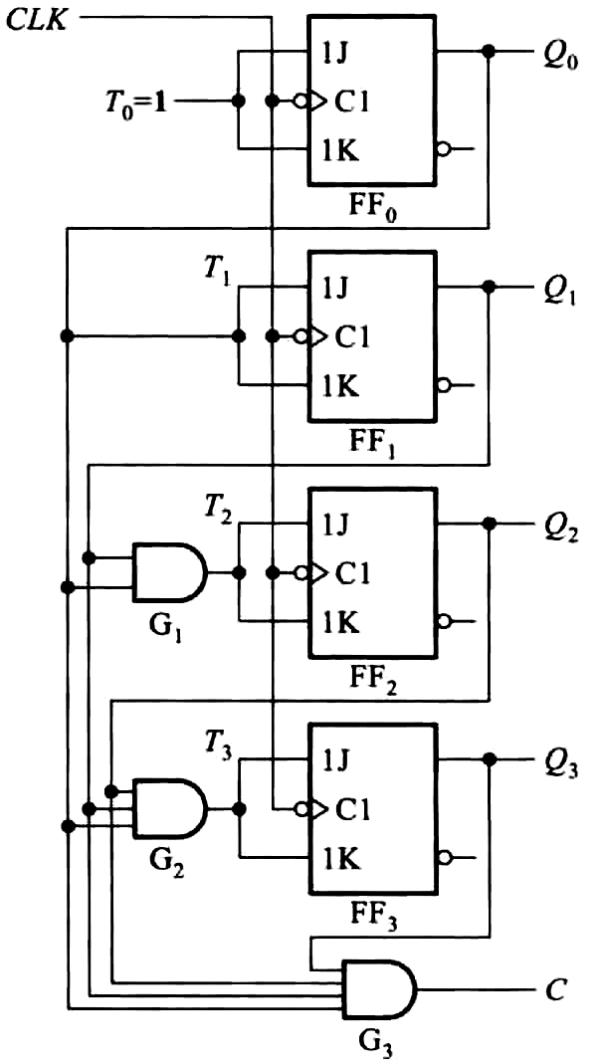


常用时序逻辑 —计数器

```
module counter(
  input clk,
  output [3:0] q,
  output c);

  reg [3:0] q;
  always @(negedge clk)
    q <= q + 1;

  assign c = &q;
endmodule
```

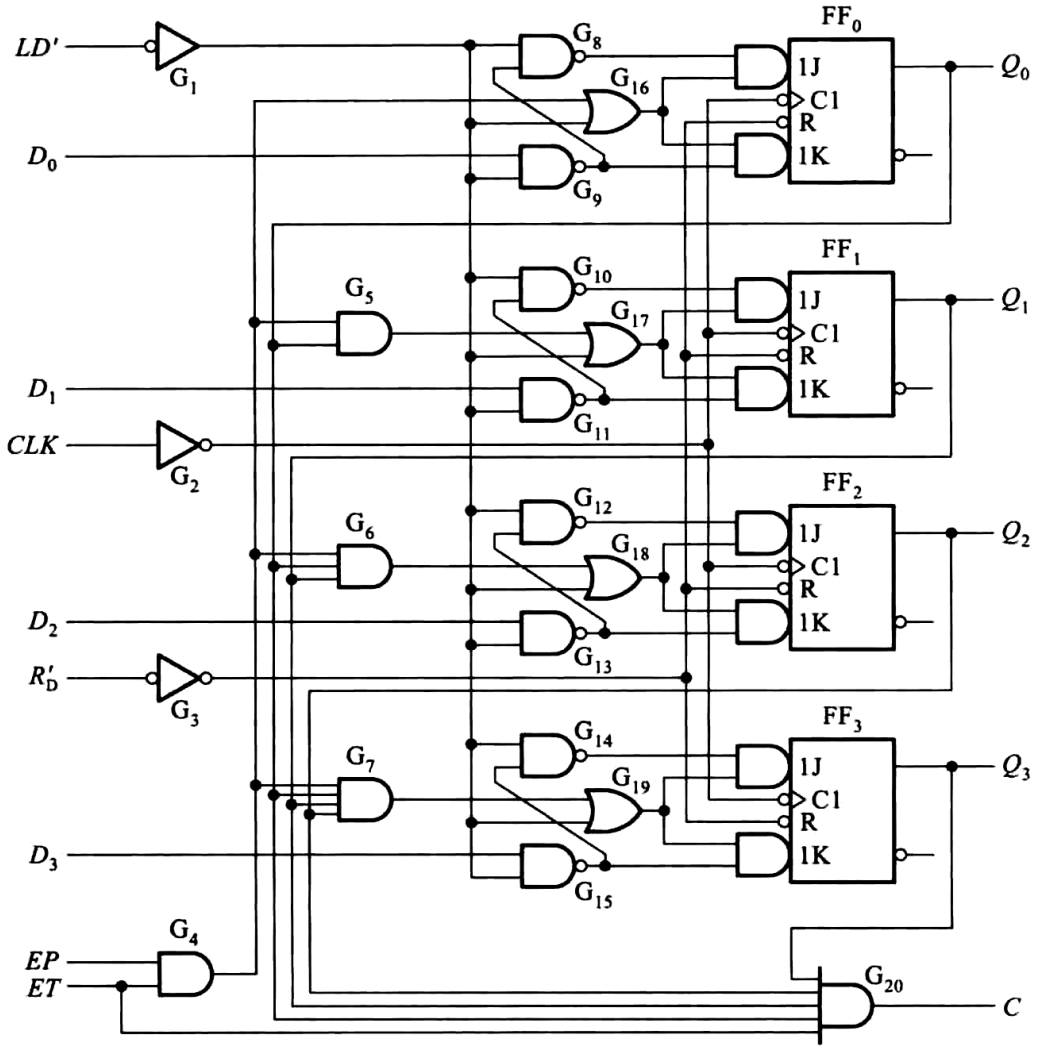
&q ：递归与，q的所有位与的结果
在这个例子中 &q == q[0]&q[1]&q[2]

同样Verilog HDL还支持:
|q ： 递归或
^q ： 递归异或



虽然Verilog的描述很简单，作为基础，我们必须能够手动推出电路的门级实现！

常用时序逻辑 —计数器



带同步置位和保持的二进制计数器74161

CLK	R'_D	LD'	EP ET	工作状态
×	0	×	× ×	置零
↑	1	0	× ×	预置数
×	1	1	0 1	保持
×	1	1	× 0	保持(但 C=0)
↑	1	1	1 1	计数

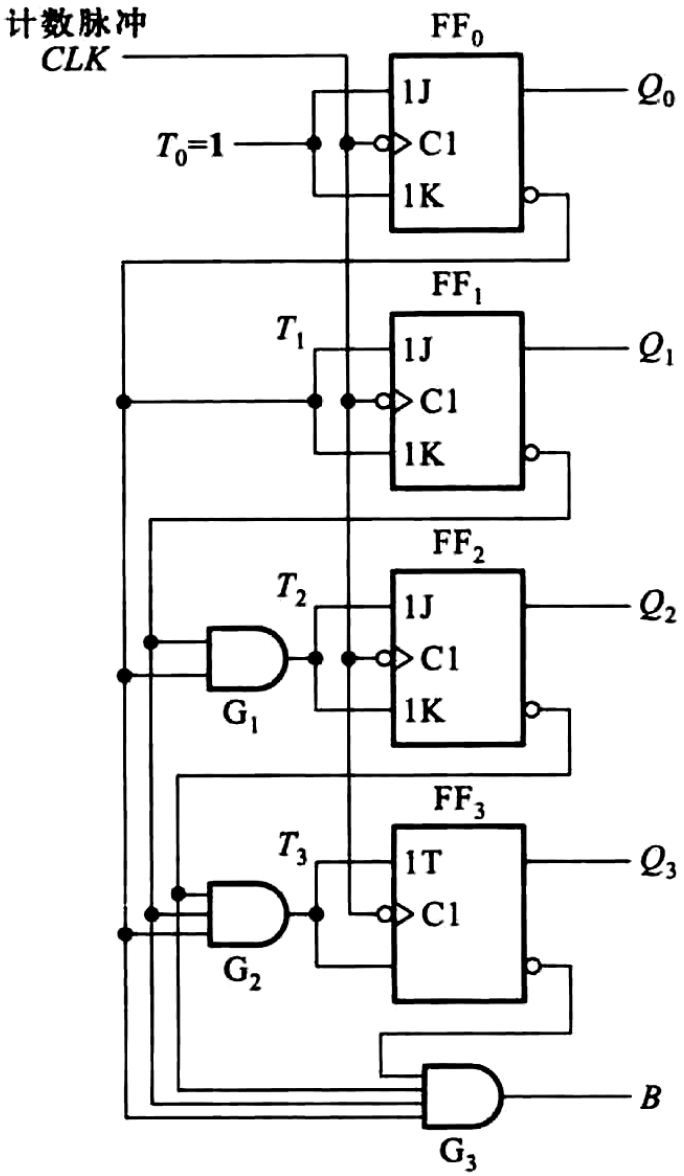
常用时序逻辑 —计数器（减法）

$$T_i = Q'_{i-1} \cdot Q'_{i-2} \cdot \cdots \cdot Q'_1 \cdot Q'_0 = \prod_{j=0}^{i-1} Q'_j$$

$$T_0 = 1$$
$$T_1 = \overline{Q_0}$$
$$T_2 = \overline{Q_0} \cdot \overline{Q_1}$$
$$T_3 = \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2}$$

$$Q_0^* = \overline{Q_0}$$
$$Q_1^* = \overline{Q_0} \cdot \overline{Q_1} + Q_0 Q_1$$
$$Q_2^* = \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2} + (Q_0 + Q_1) Q_2$$
$$Q_3^* = \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3} + (Q_0 + Q_1 + Q_2) Q_3$$

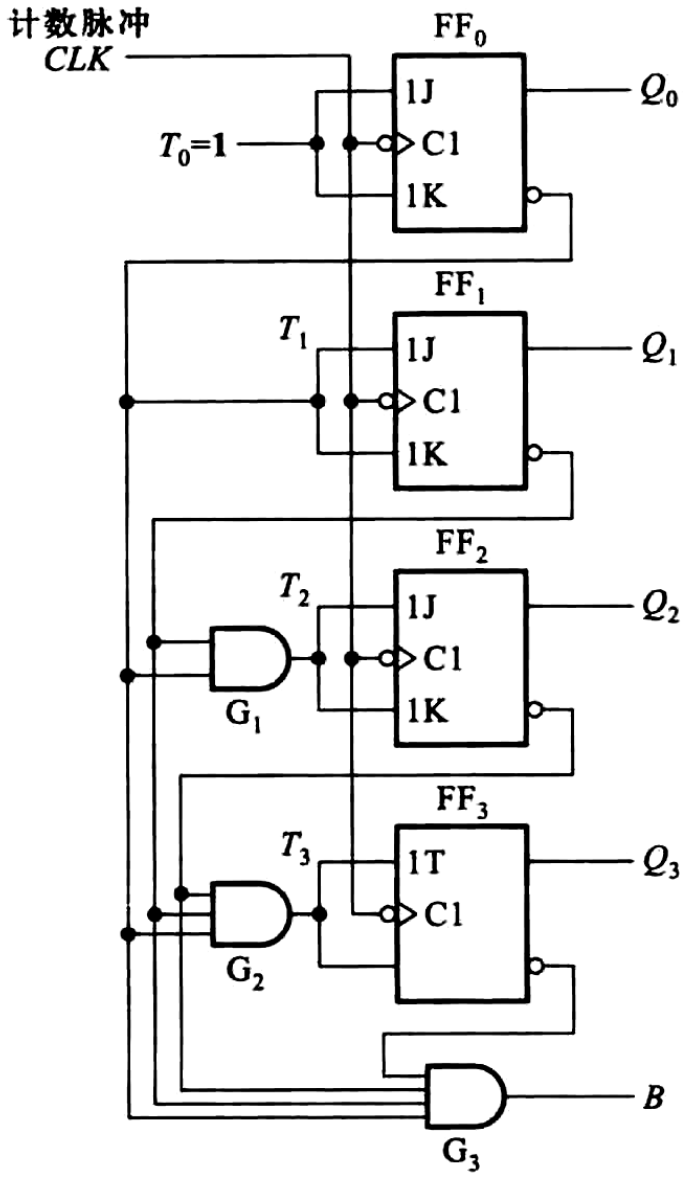
$$B = \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3}$$



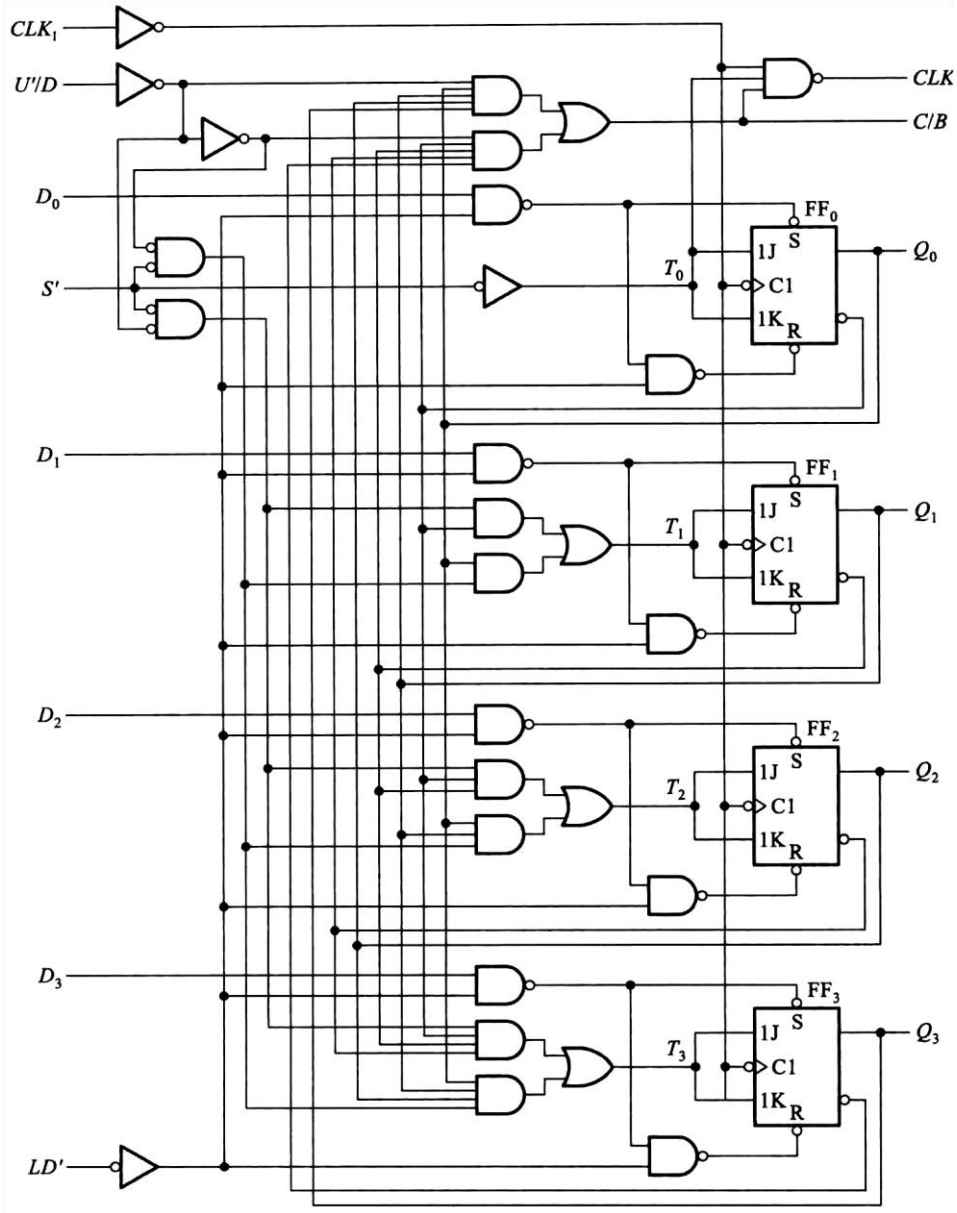
常用时序逻辑 —计数器（减法）

```
module counter(
  input clk,
  output [3:0] q,
  output b);

  reg [3:0] q;
  always @(negedge clk)
    q <= q - 1;

  assign b = &(~q);
endmodule
```

~q 是对q的按位取反操作。



常用时序逻辑—计数器（加减法）



同步十六进制加/减法计数器74LS191

CLK_1	S'	LD'	U'/D	工作状态
×	1	1	×	保持
×	×	0	×	预置数
↑	0	1	0	加法计数
↑	0	1	1	减法计数

常用时序逻辑—计数器（十进制）

$$T_0 = 1$$
$$T_1 = Q_0\overline{Q_3}$$
$$T_2 = Q_0Q_1$$
$$T_3 = Q_0Q_1Q_2 + Q_0Q_3$$

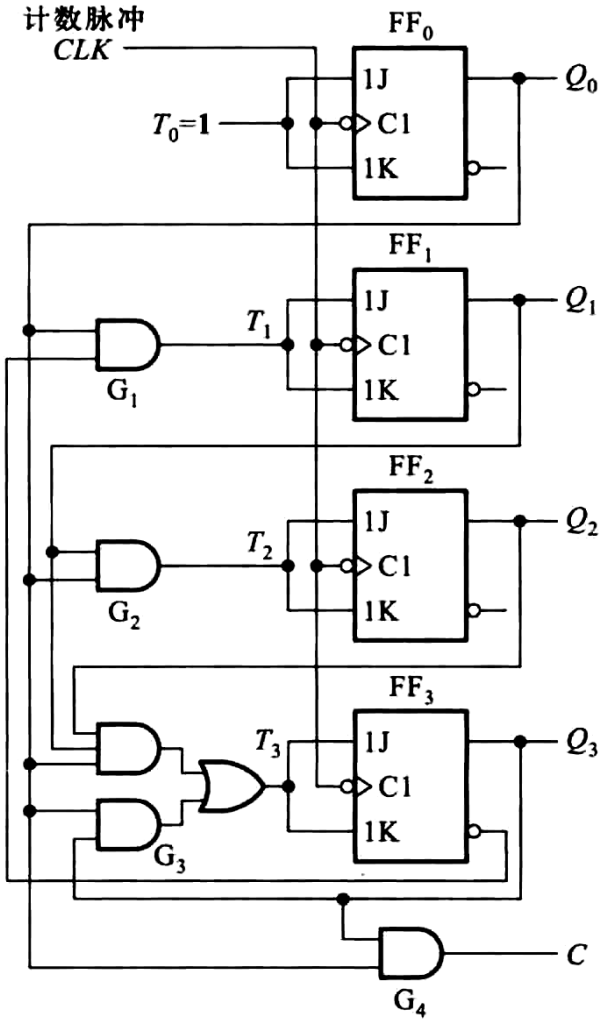
0xx1

1xx1

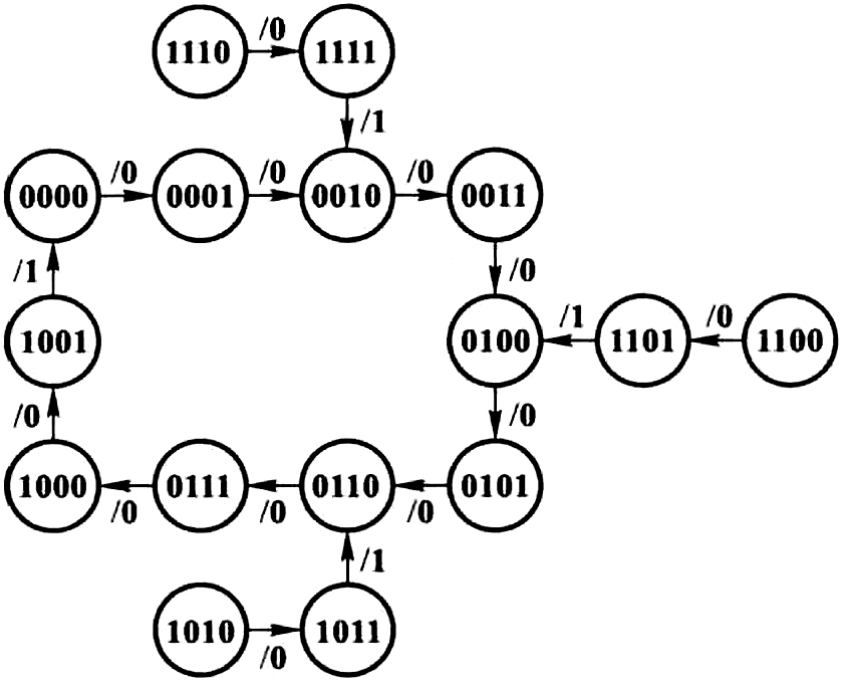
$$Q_0^* = \overline{Q_0}$$
$$Q_1^* = Q_0\overline{Q_3}\cdot\overline{Q_1} + \overline{Q_0\overline{Q_3}}Q_1$$
$$Q_2^* = Q_0Q_1\overline{Q_2} + \overline{Q_0Q_1}Q_2$$

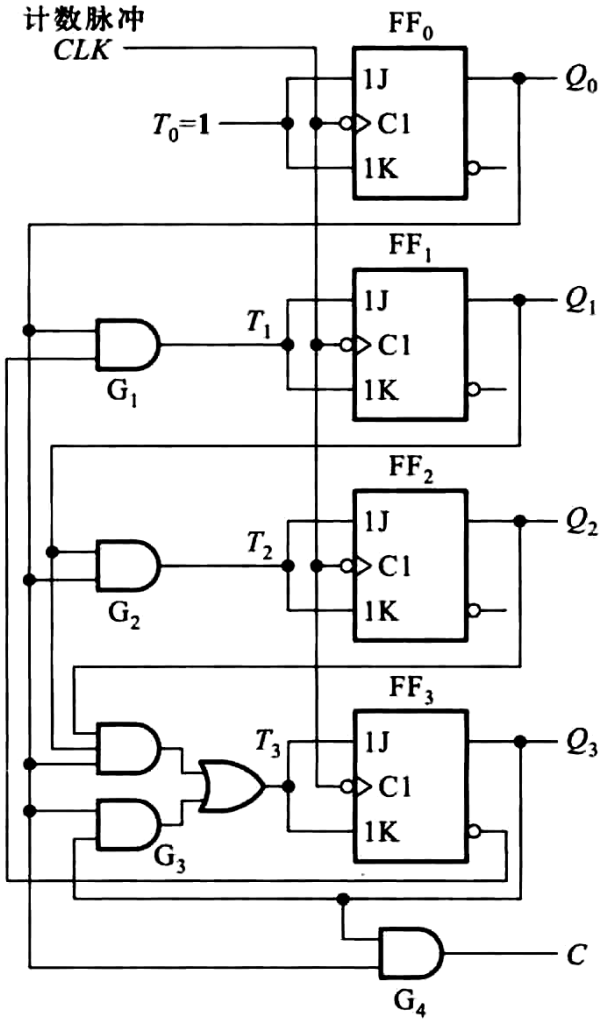
$$Q_3^* = (Q_0Q_1Q_2 + Q_0Q_3)\overline{Q_3} + \overline{(Q_0Q_1Q_2 + Q_0Q_3)}Q_3$$
$$= Q_0Q_1Q_2 + \overline{Q_0}Q_3$$

$$C = Q_0Q_3$$



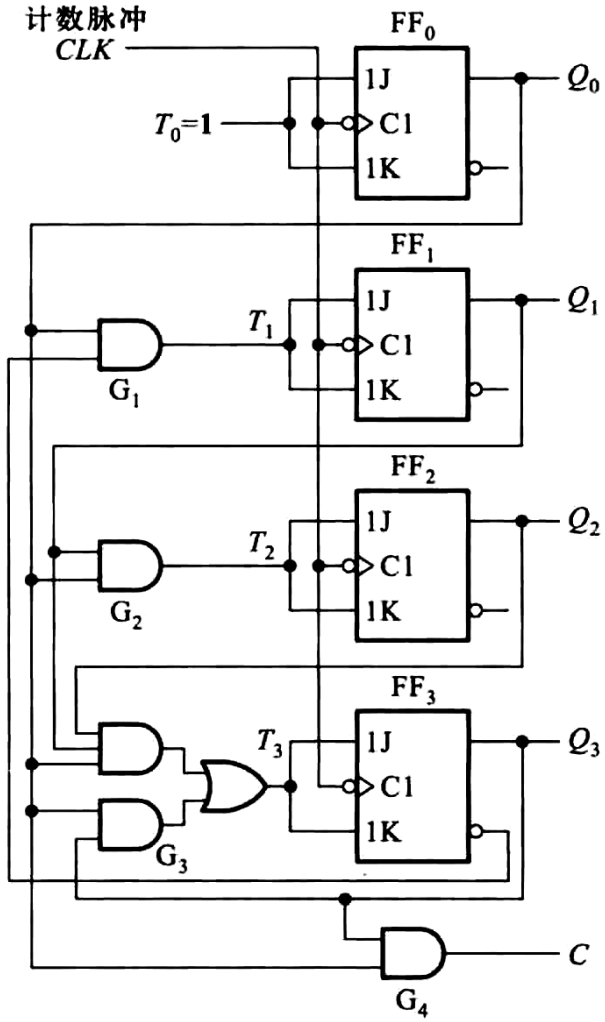
常用时序逻辑—计数器（十进制）



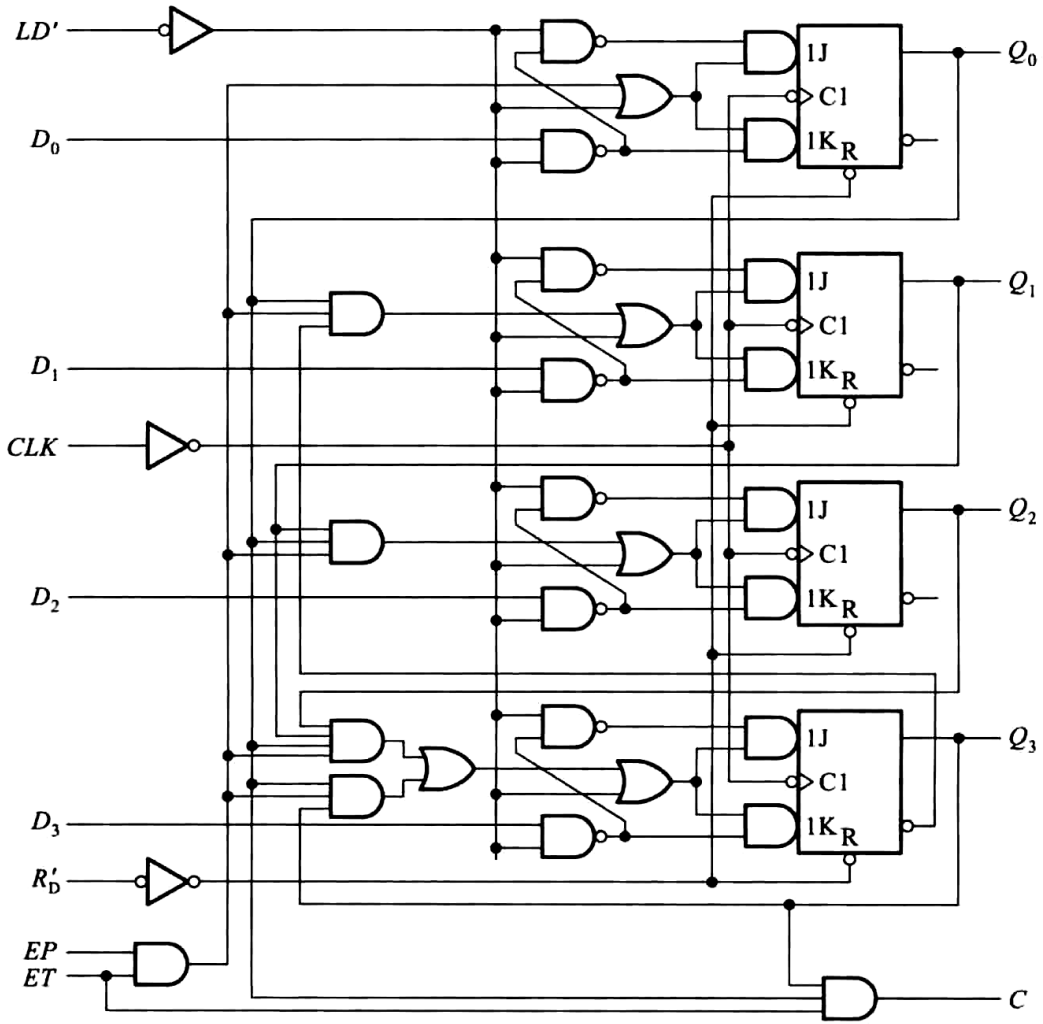


常用时序逻辑—计数器（十进制）

```
module counter(
  input clk,
  output [3:0] q,
  output c);

  reg [3:0] q;
  always @(negedge clk)
    if(q < 4'd9) q <= q + 1;
    else         q <= 0;

  assign c = q == 4'd9;
endmodule
```



常用时序逻辑—计数器（十进制）



带同步置位和保持的十进制计数器74160

CLK	R'_D	LD'	EP ET	工作状态
×	0	×	× ×	置零
↑	1	0	× ×	预置数
×	1	1	0 1	保持
×	1	1	× 0	保持(但 C=0)
↑	1	1	1 1	计数

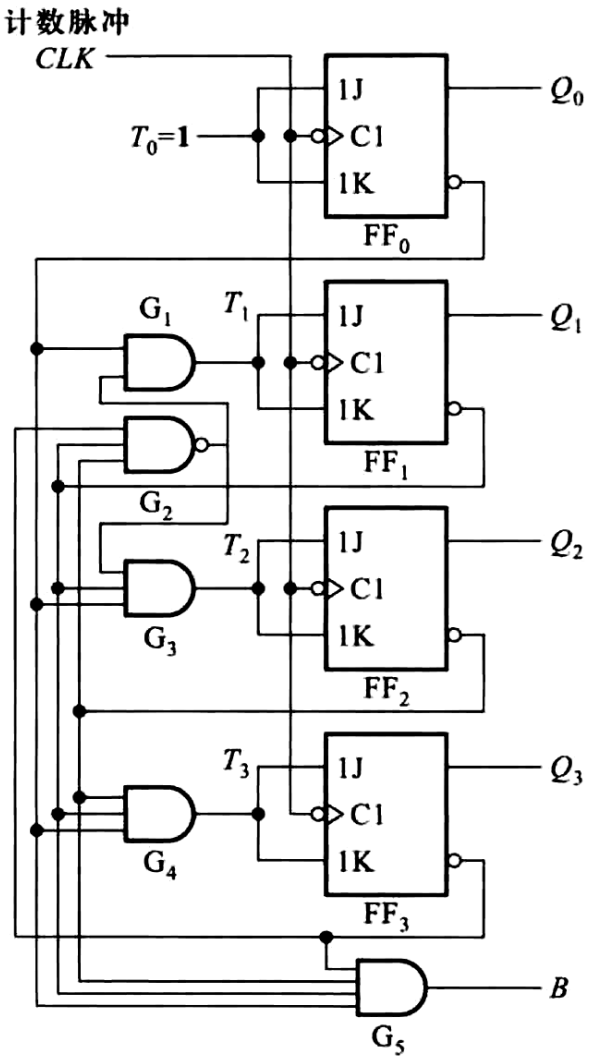
常用时序逻辑 —计数器（十进制减法）

$0 \rightarrow 9$

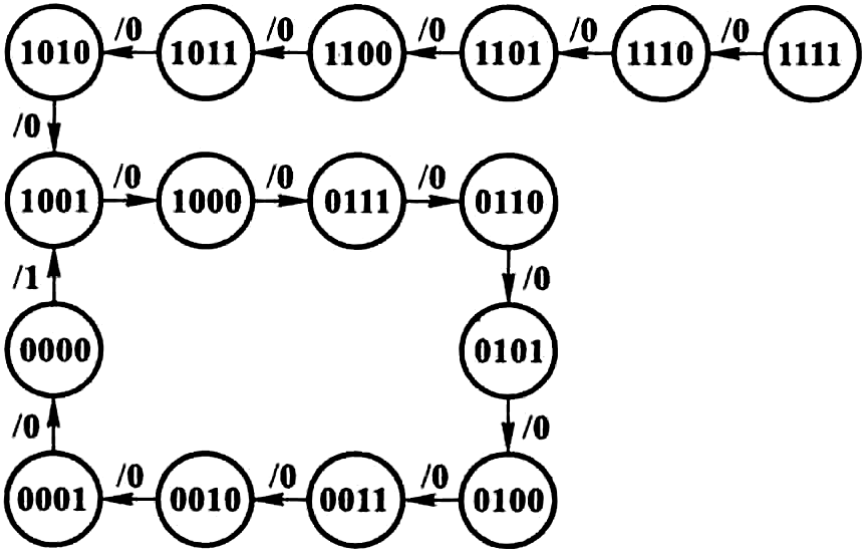
$$T_0 = 1$$
$$T_1 = \overline{Q_0} \cdot \overline{\overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3}}$$
$$T_2 = \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_1 \cdot Q_2 \cdot Q_3}$$
$$T_3 = \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2}$$

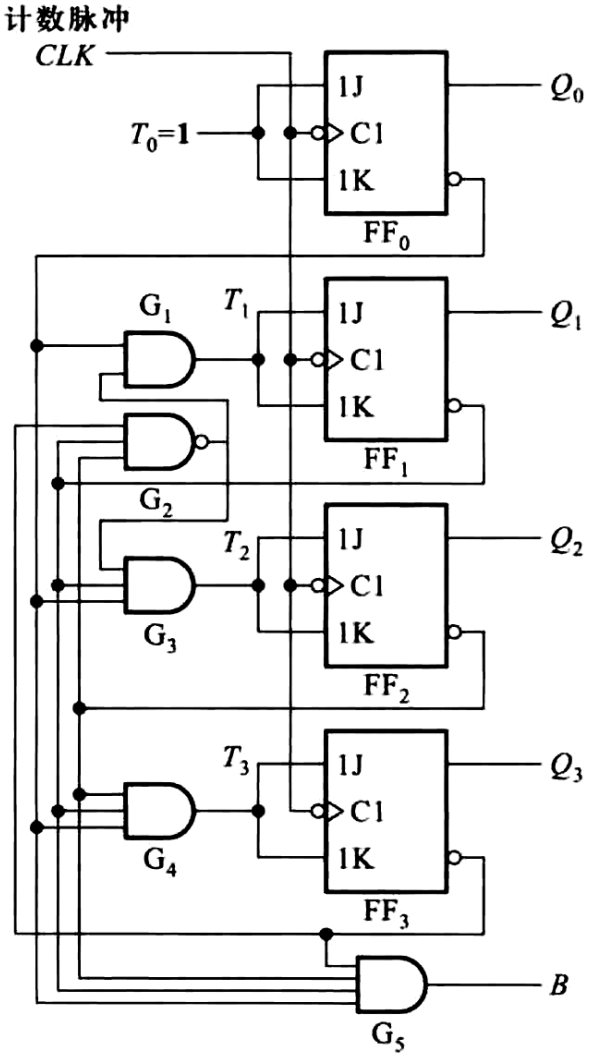
$$Q_0^* = \overline{Q_0}$$
$$Q_1^* = \overline{Q_0} \cdot \overline{\overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3}} \cdot \overline{Q_1} + (Q_0 + \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3}) Q_1$$
$$Q_2^* = \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{\overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3}} \cdot \overline{Q_2} + (Q_0 + Q_1 + \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3}) Q_2$$
$$Q_3^* = \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3} + (Q_0 + Q_1 + Q_2) Q_3$$

$$B = \overline{Q_0} \cdot \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3}$$



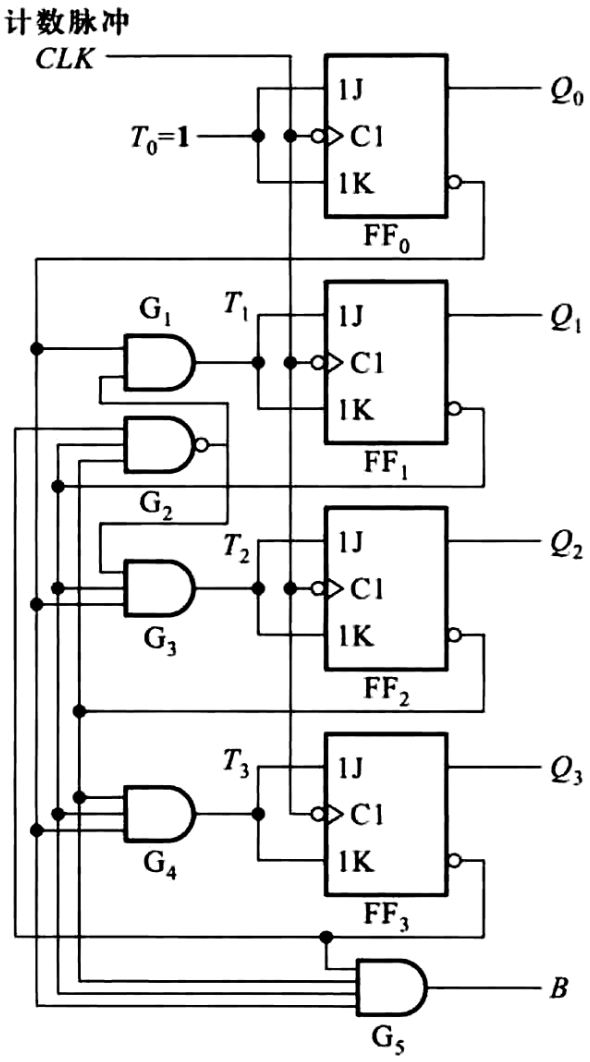
常用时序逻辑 —计数器（十进制减法）



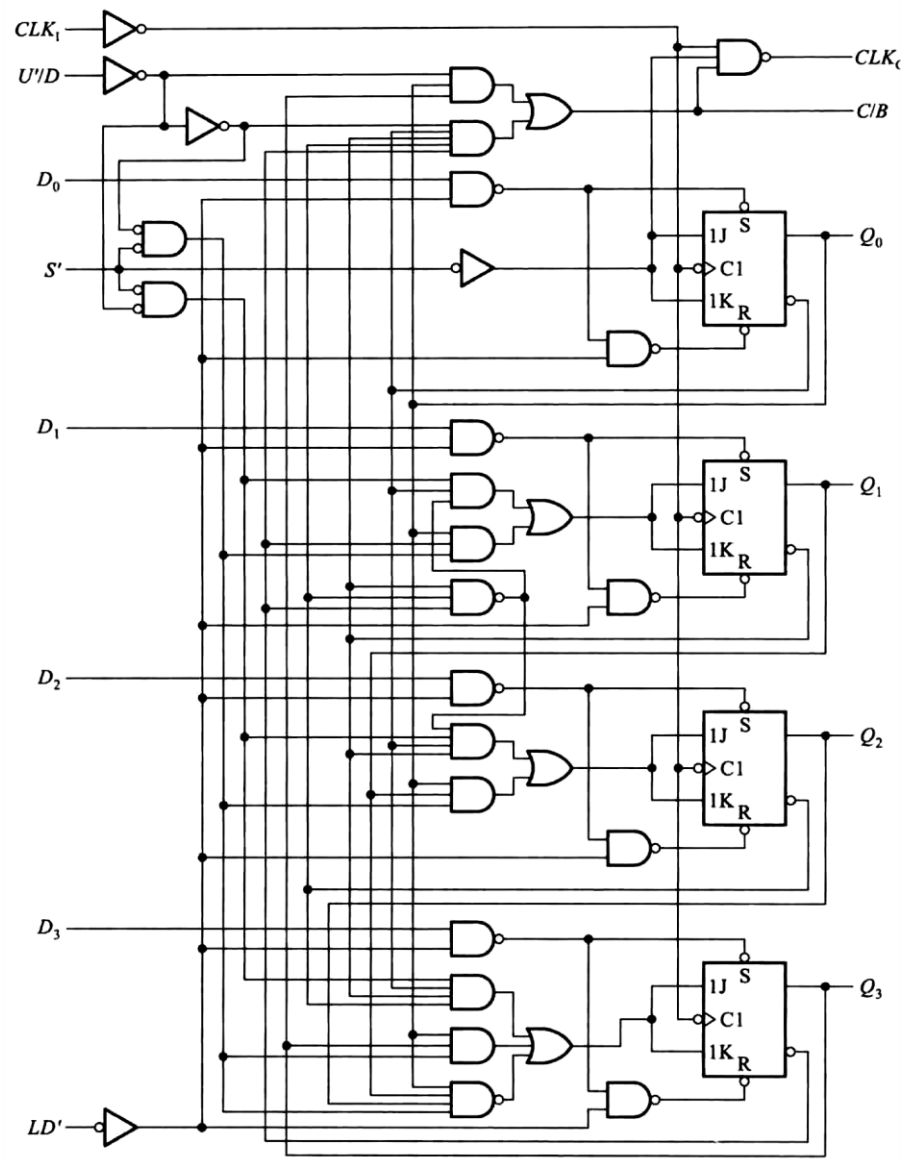


常用时序逻辑—计数器（十进制减法）

```
module counter(
  input clk,
  output [3:0] q,
  output b);

  reg [3:0] q;
  always @(negedge clk)
    if(q > 0) q <= q - 1;
    else      q <= 4'd9;

  assign b = q == 4'd0;
endmodule
```



常用时序逻辑—计数器（十进制加减法）

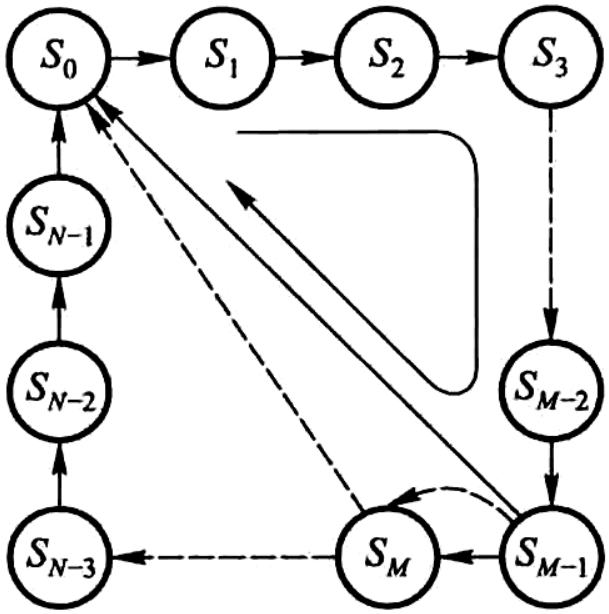


同步十进制加/减法计数器
74LS190

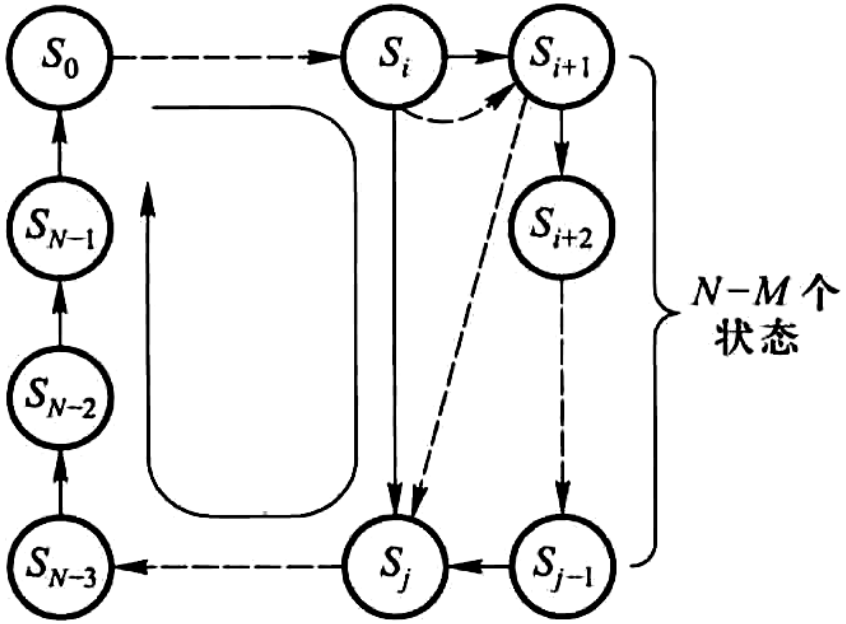
CLK_1	S'	LD'	U'/D	工作状态
×	1	1	×	保持
×	×	0	×	预置数
↑	0	1	0	加法计数
↑	0	1	1	减法计数

任意数制计数器（M<N）

- 利用现有集成电路模块构成M进制计数器，M小于电路模块的计数范围N（M<N）



置零法
当记到M-1时，返回到0

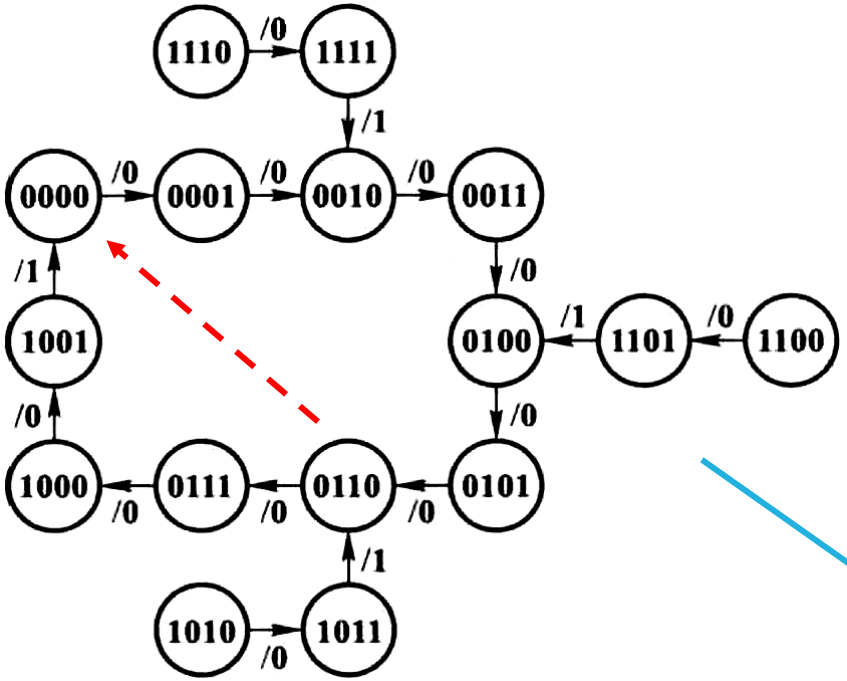


置数法
当记到i时，直接跳到j
j-i=N-M+1

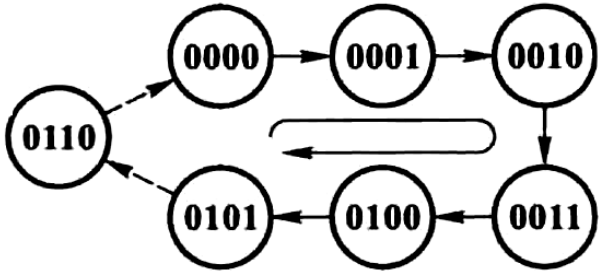
实线：同步置数；虚线：异步置数

十进制计数器实现计6计数器

- 使用74160，使用异步归零

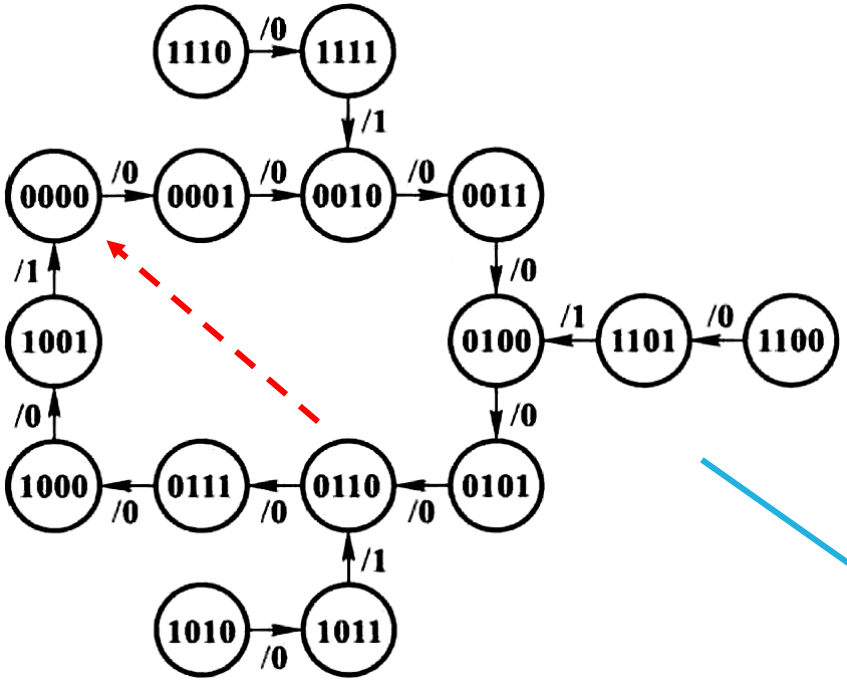


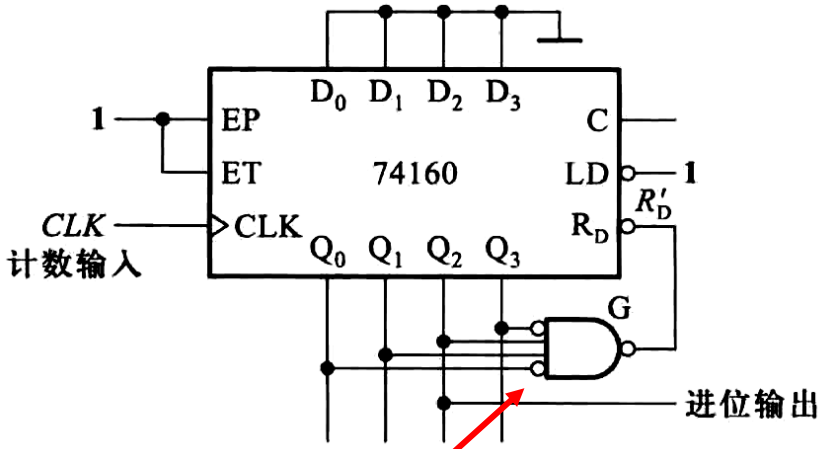
当计数器计到6时，立即将计数器归零（异步）。所以，6为瞬时状态。

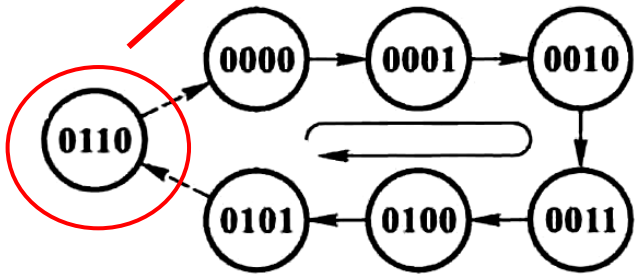


十进制计数器实现计6计数器

- 使用74160，使用异步归零

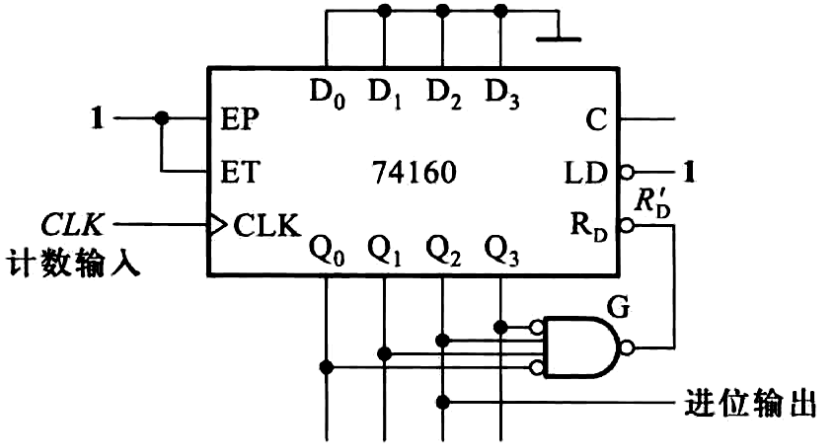


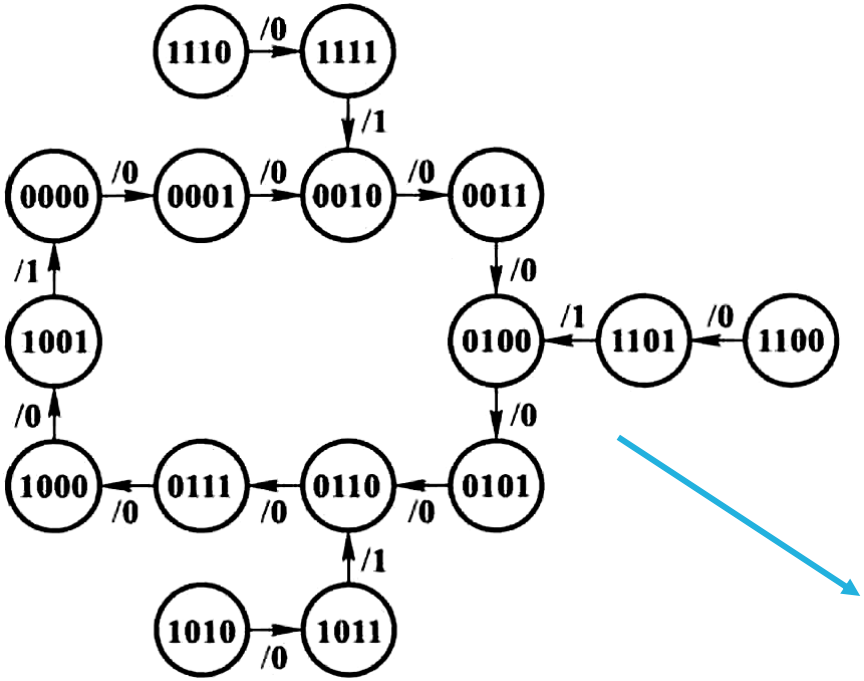


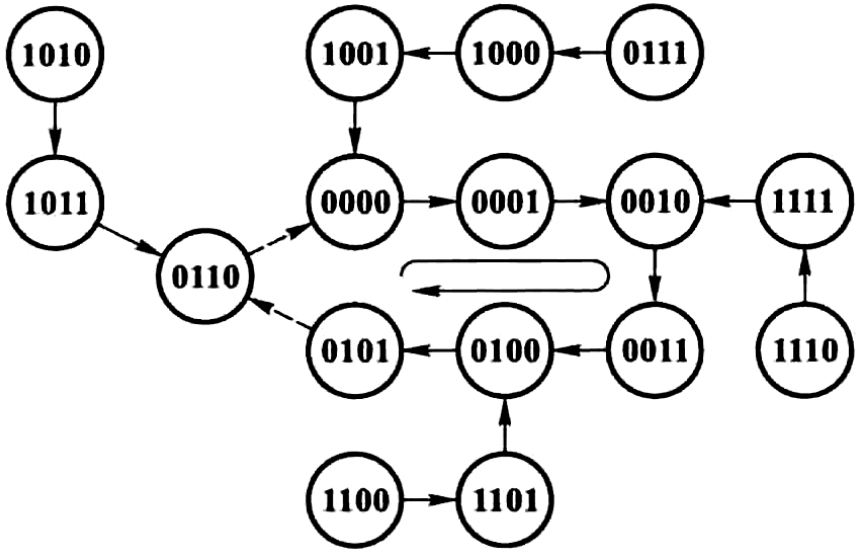


十进制计数器实现计6计数器

- 使用74160，使用异步归零



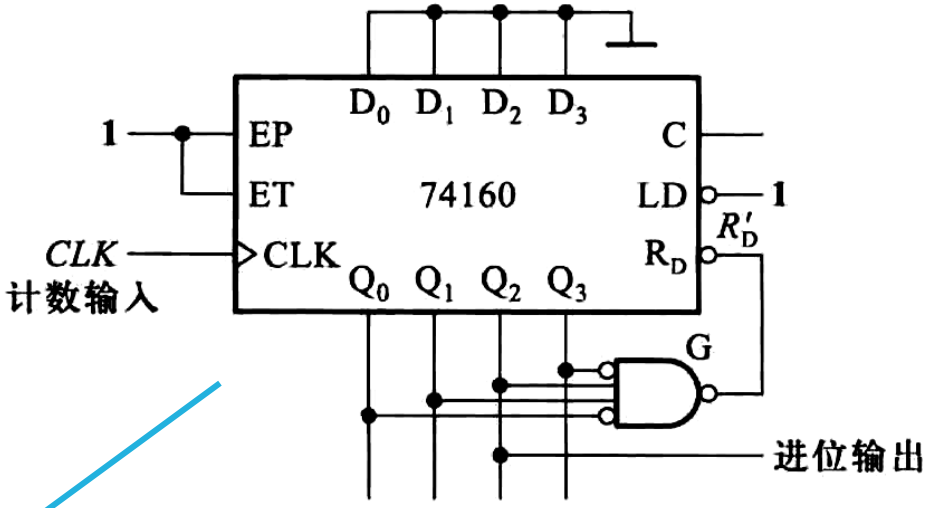


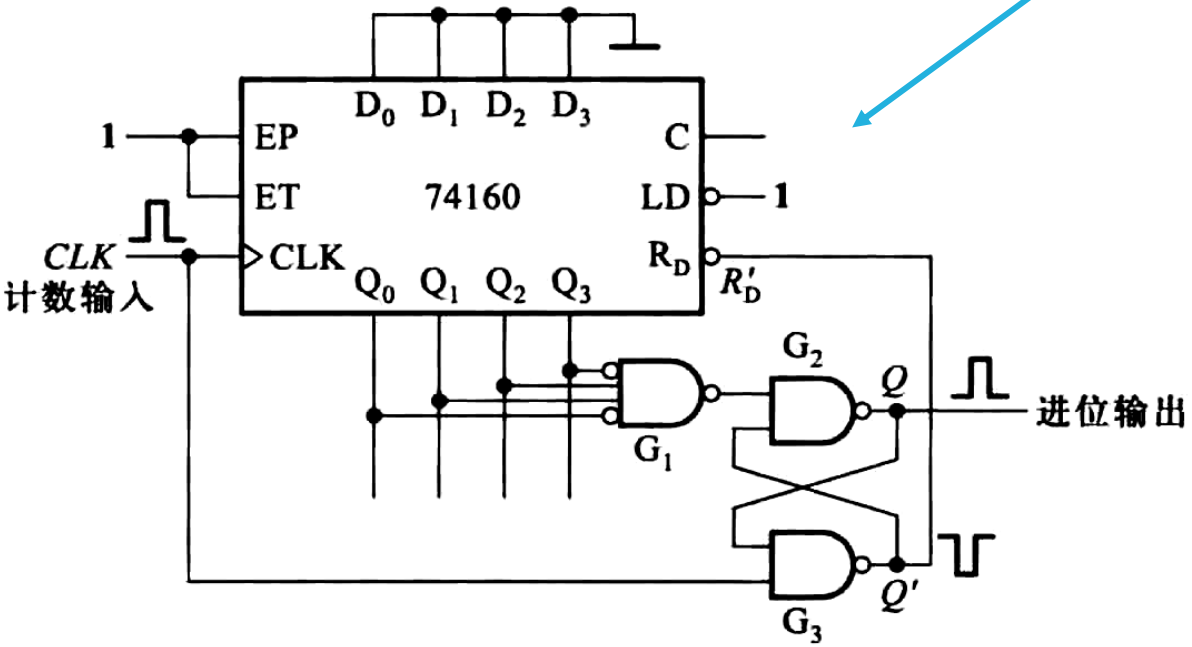


进位输出持续>2个时钟周期；
归零复位信号为一个短脉冲，不稳定。

十进制计数器实现计6计数器

- 使用74160，使用异步归零

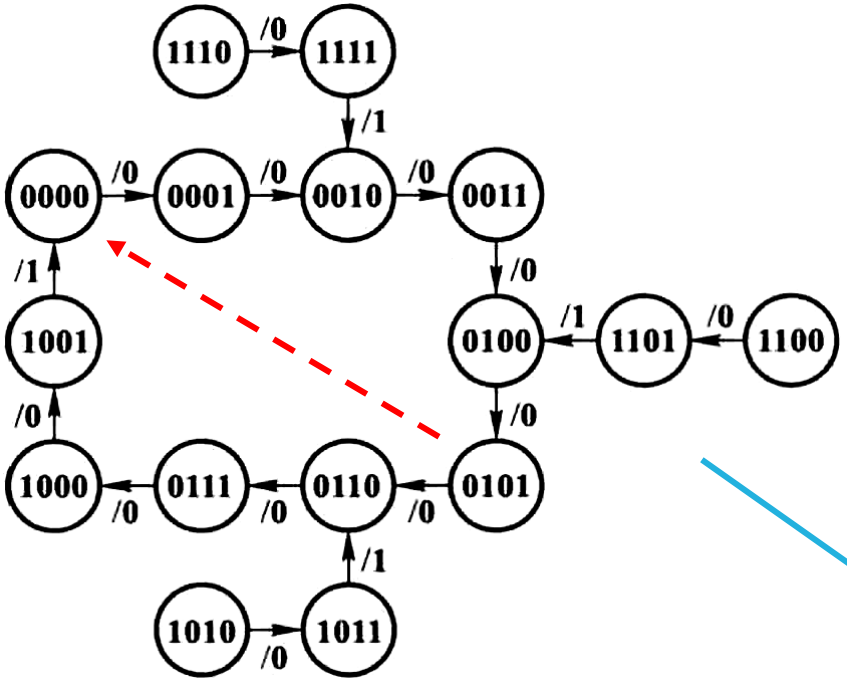




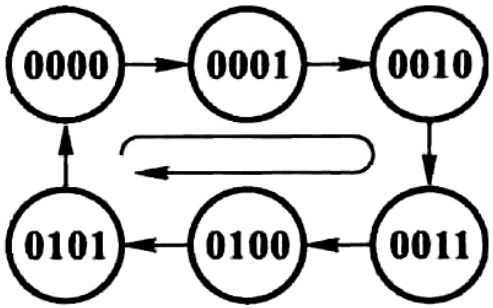
推导波形？

十进制计数器实现计6计数器

- 使用74160，使用同步置数（置0）

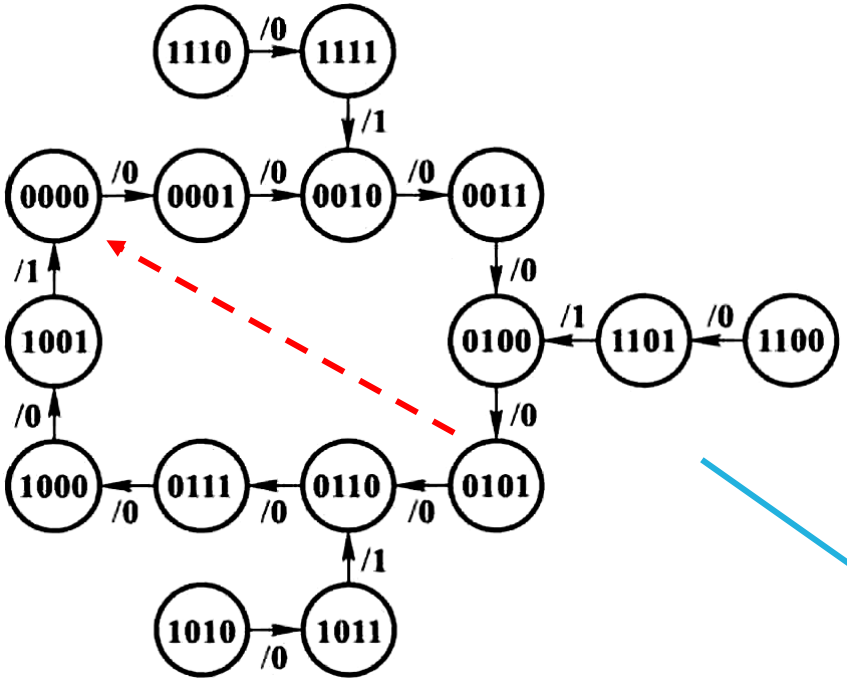


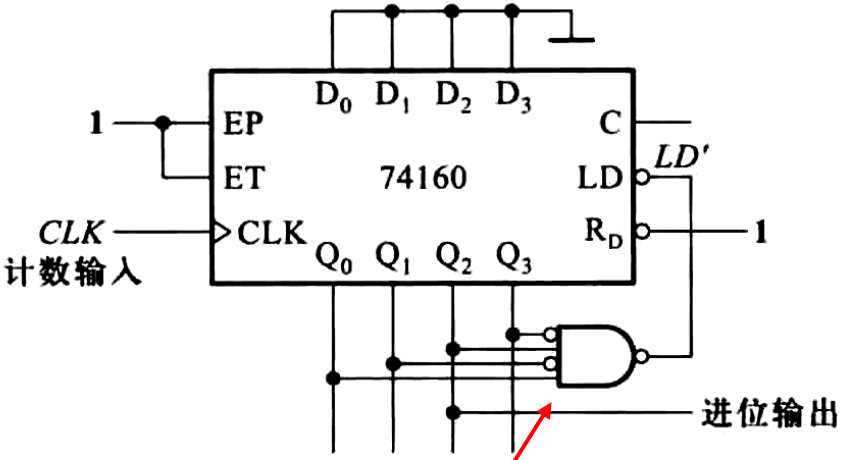
当计数器计到5时，使用同步置数端将下一个周期的状态置为0。

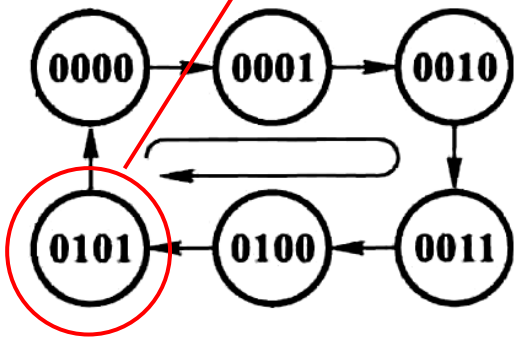


十进制计数器实现计6计数器

- 使用74160，使用同步置数（置0）

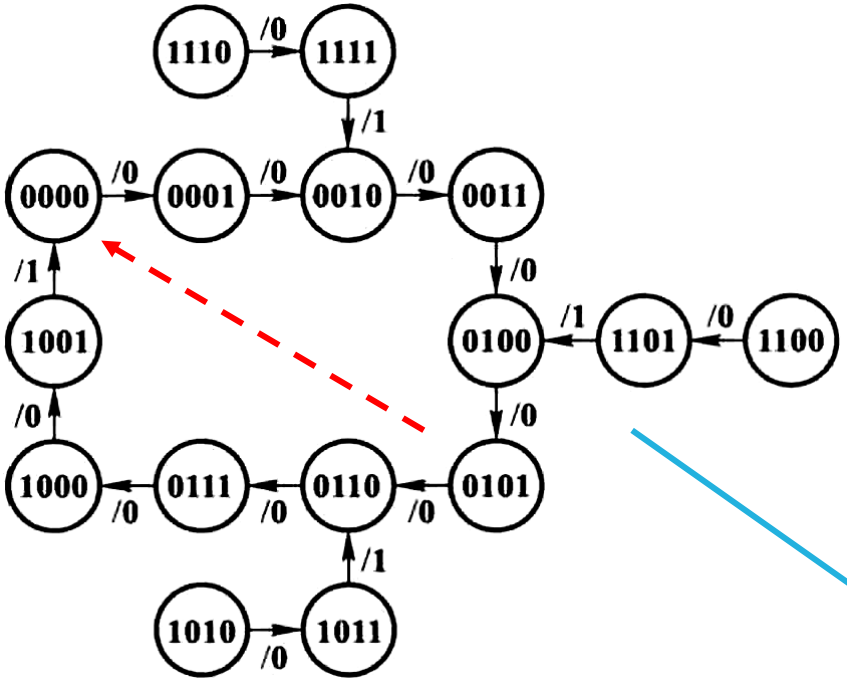


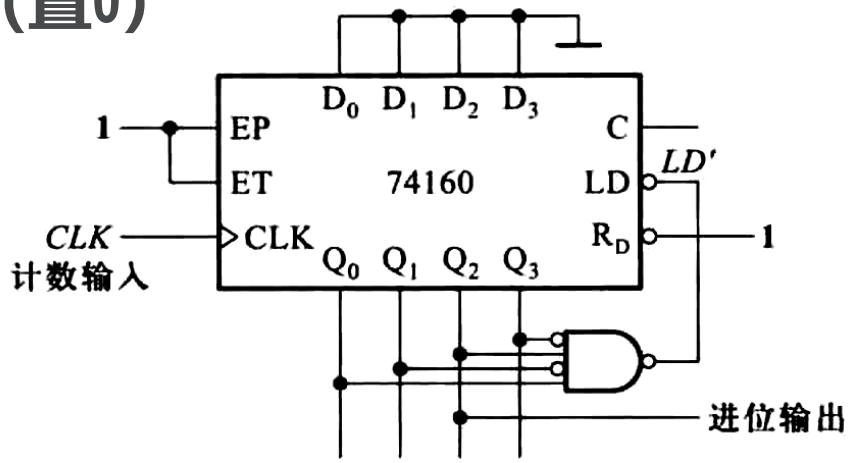


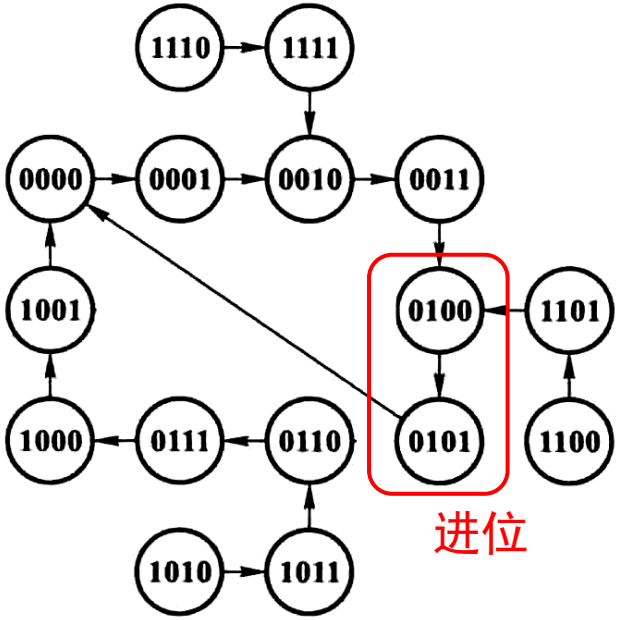


十进制计数器实现计6计数器

- 使用74160，使用同步置数（置0）

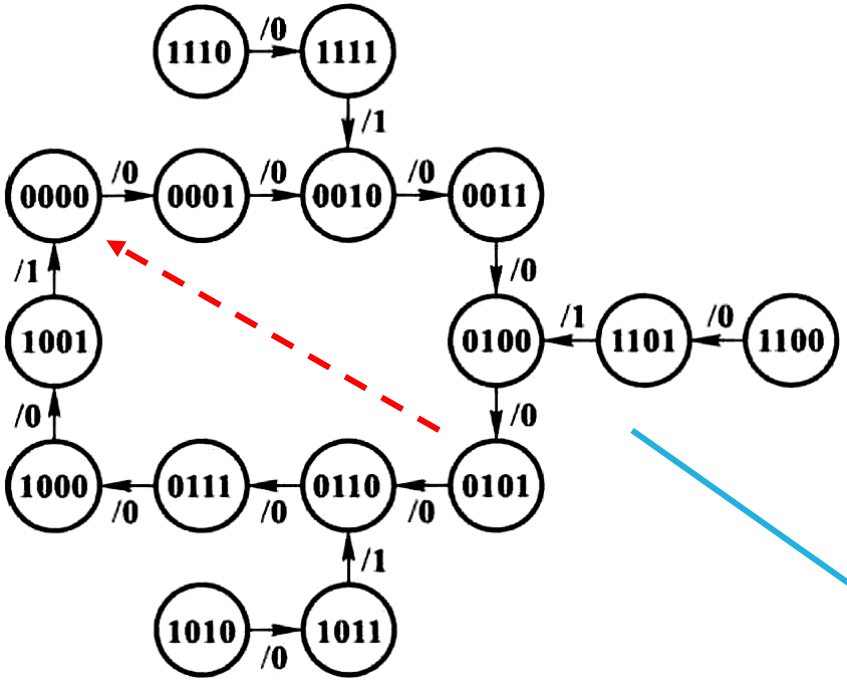


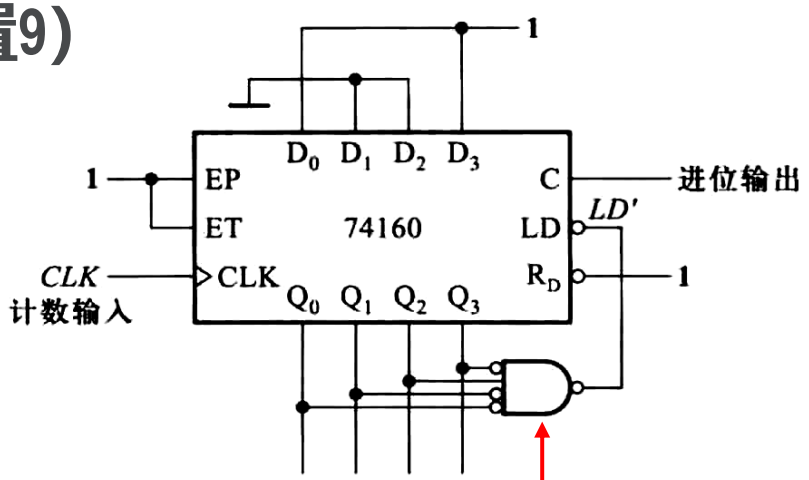




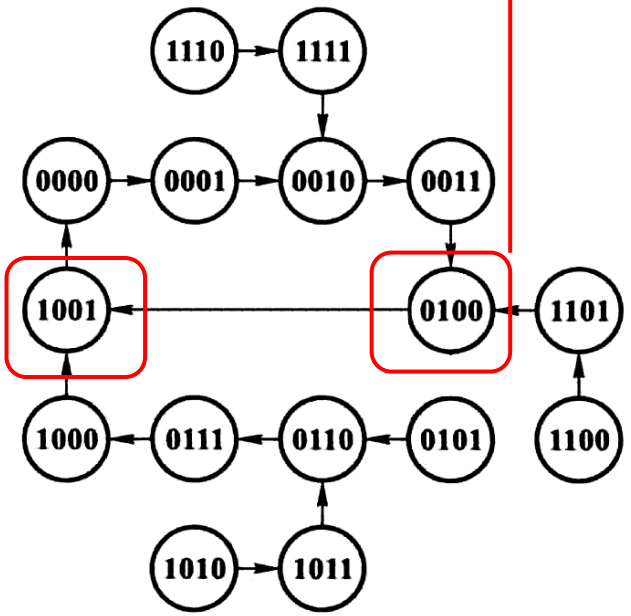
十进制计数器实现计6计数器

- 使用74160，使用同步置数（4置9）





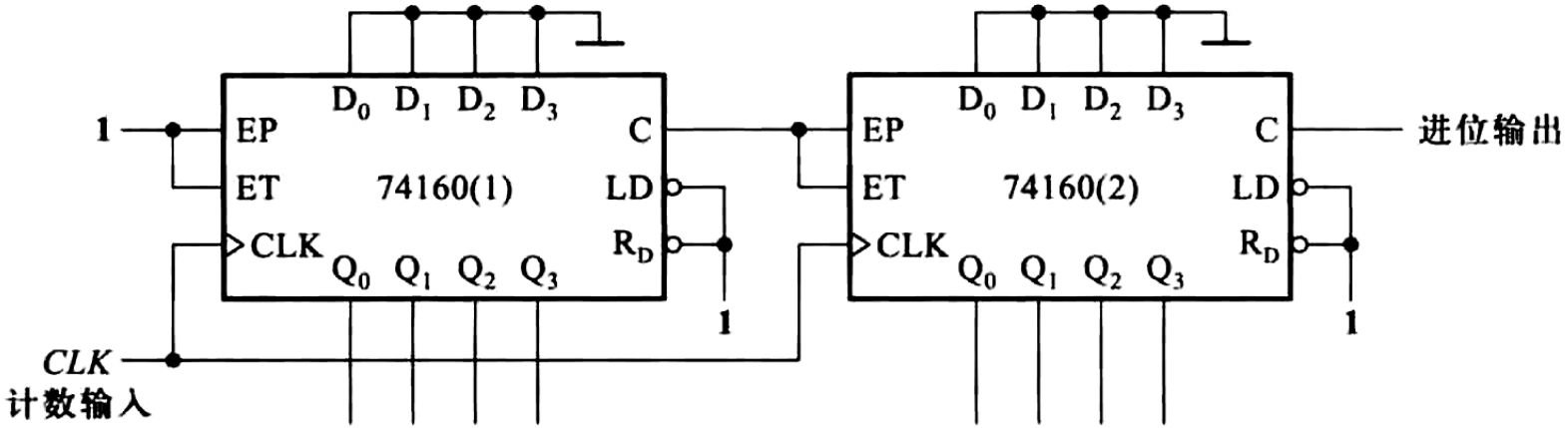
复用原有进位



任意数制计数器（M>N）

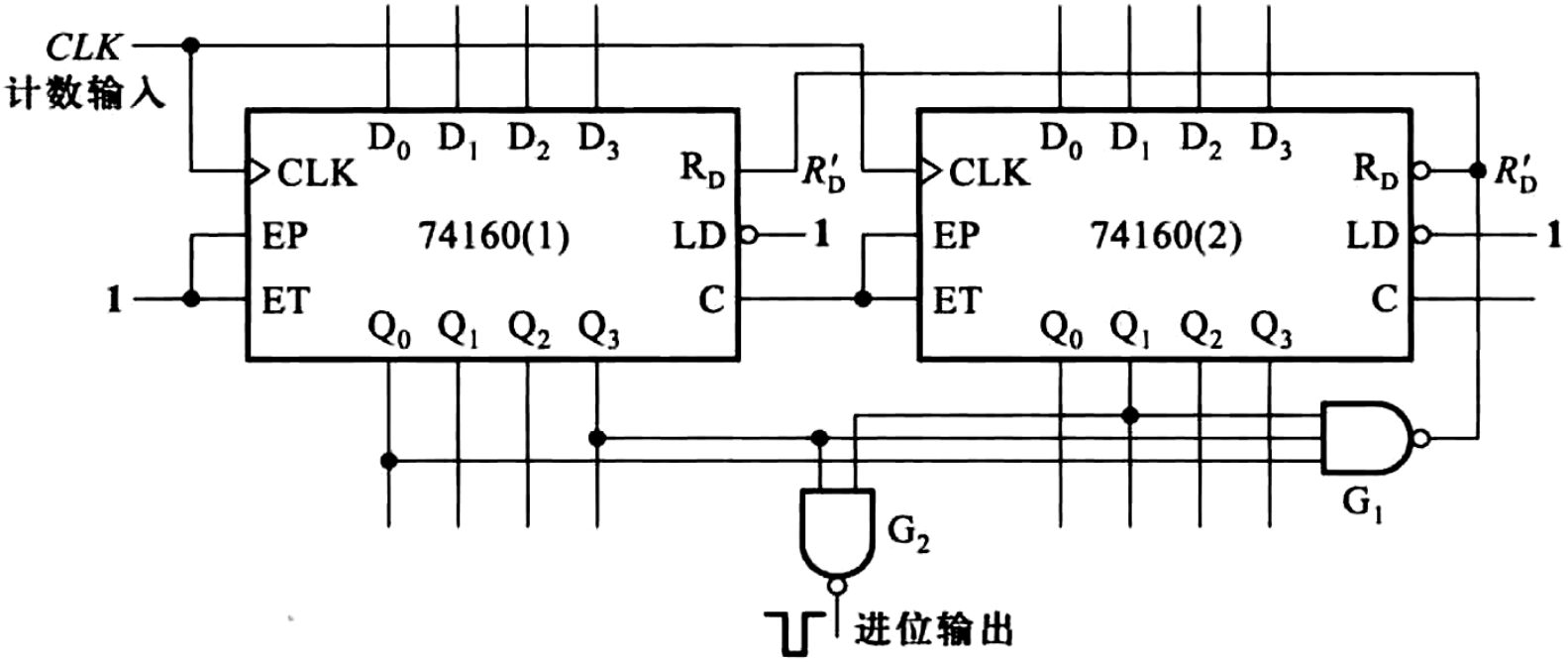
- 利用现有集成电路模块构成M进制计数器，M大于电路模块的计数范围N（M>N）
  - 多片级联

CLK	R'_D	LD'	EP ET	工作状态
×	0	×	× ×	置零
↑	1	0	× ×	预置数
×	1	1	0 1	保持
×	1	1	× 0	保持（但 C=0）
↑	1	1	1 1	计数



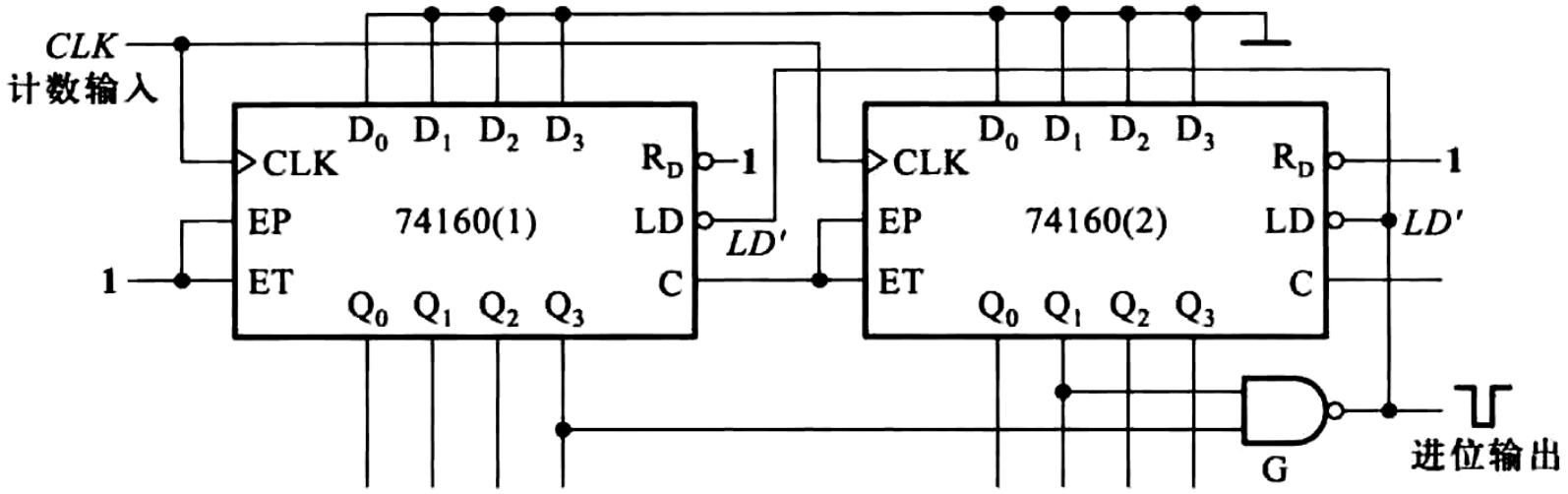
两片74160组成一个计100计数器

计29计数器（异步归零）



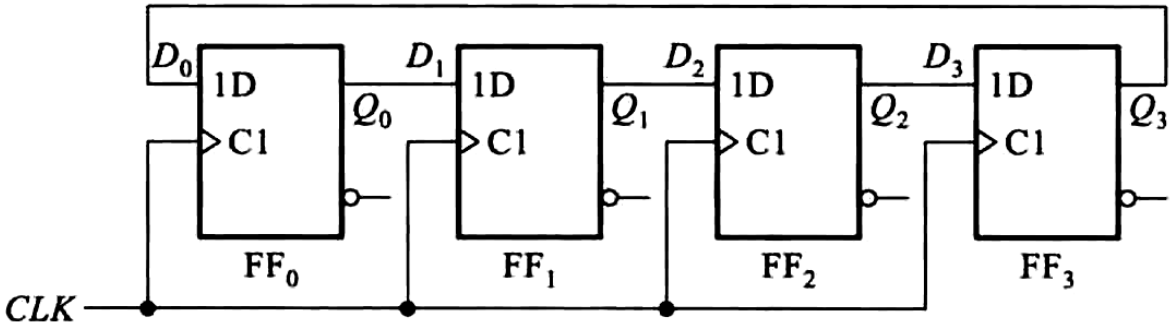
复位（归零）信号由短暂的29输出；
进位信号（低有效）由28给出。

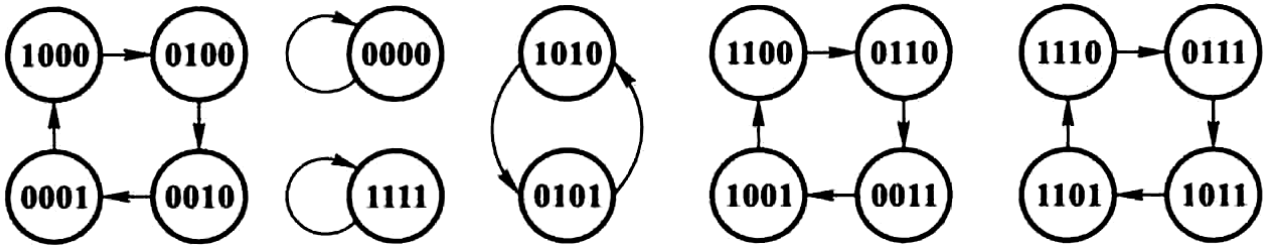
计29计数器（同步置零）



进位信号（低有效）和置零信号同时由28给出。

移位寄存器型计数器

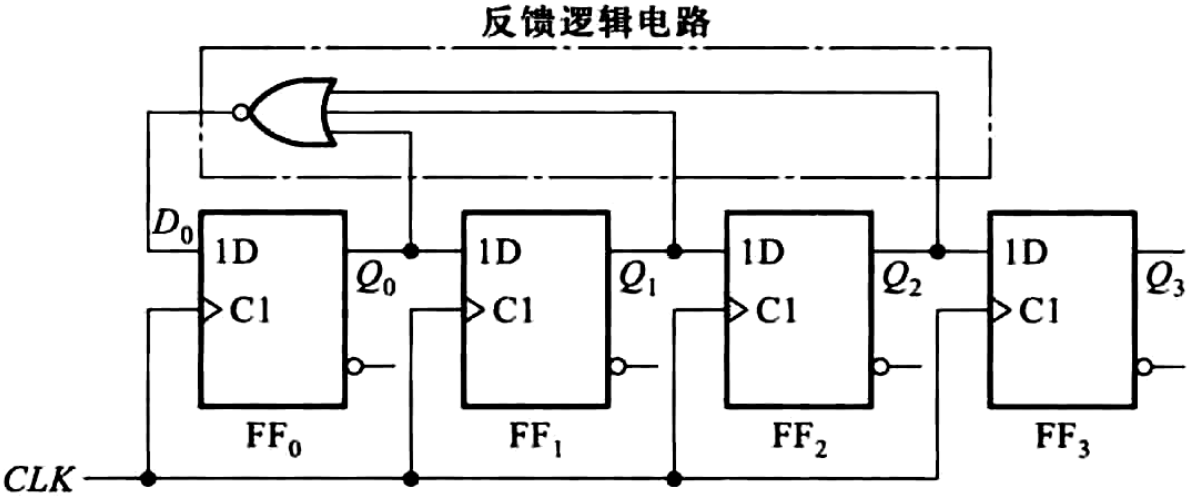


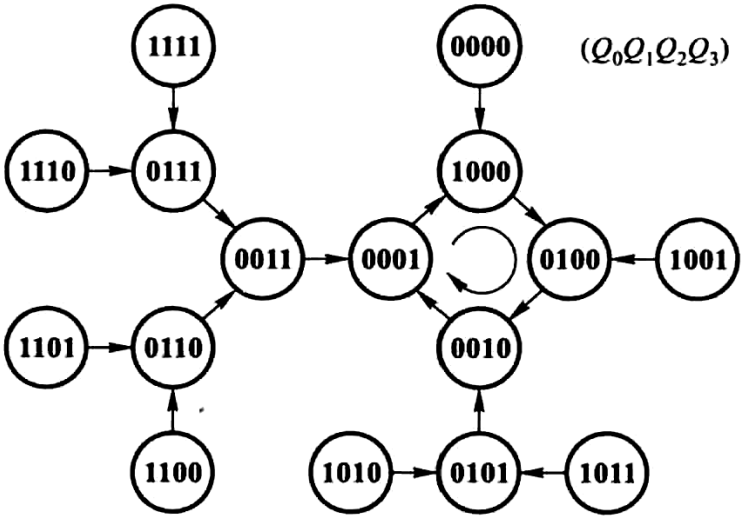


有效循环 无效循环

$(Q_0Q_1Q_2Q_3)$

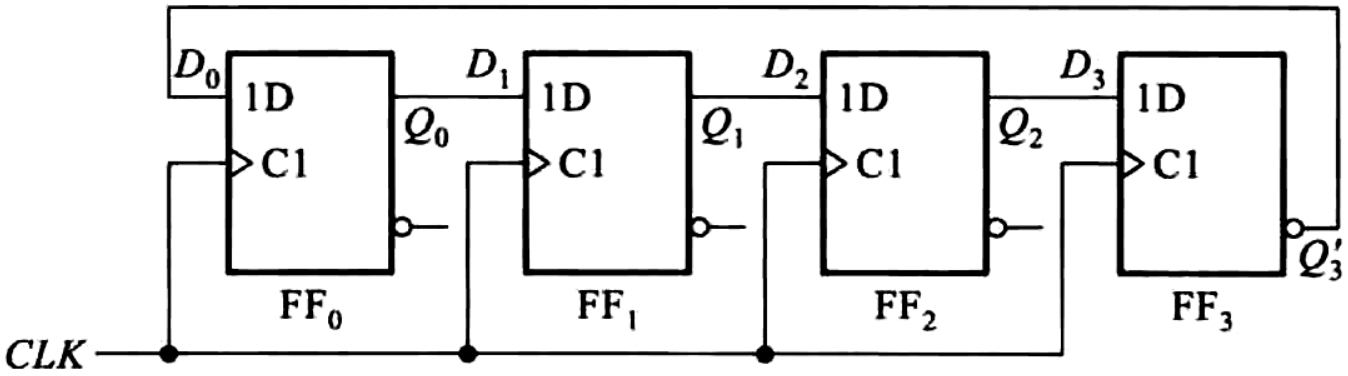
移位寄存器型计数器（自启）

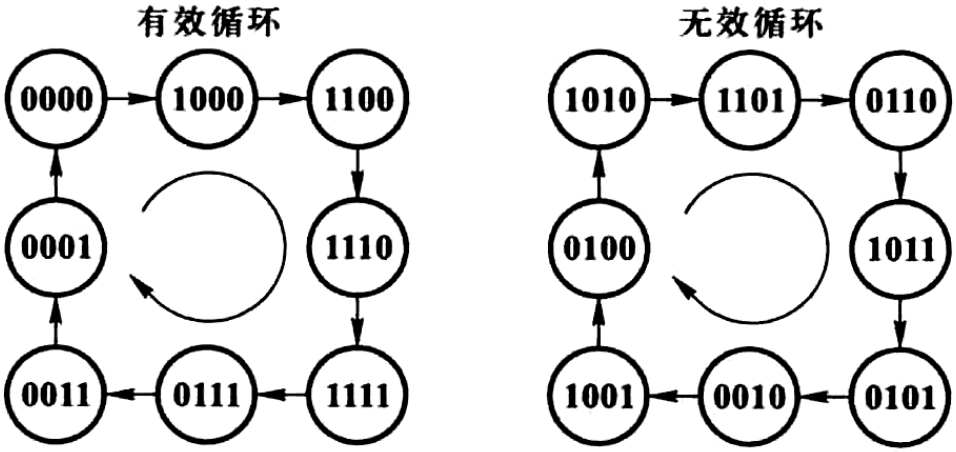




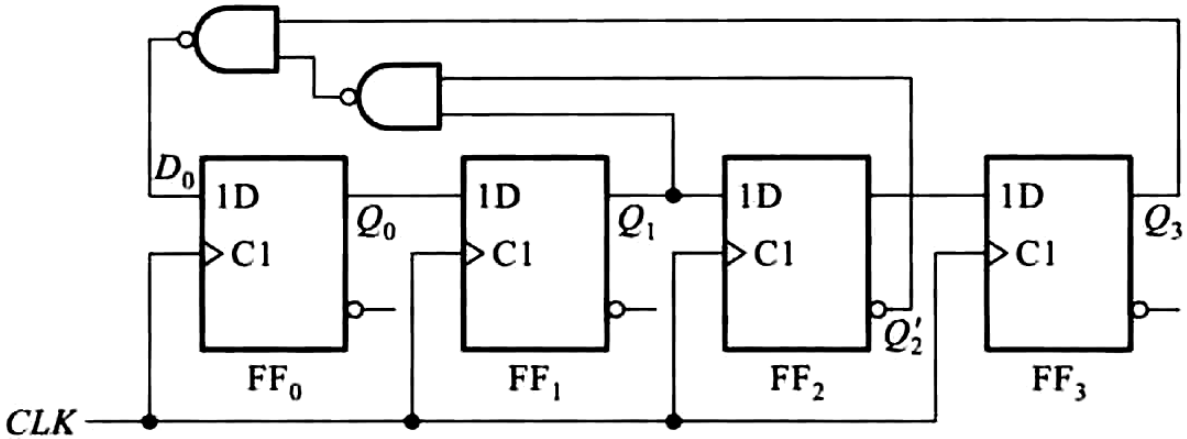
寄存器的使用效率较低

扭环型计数器

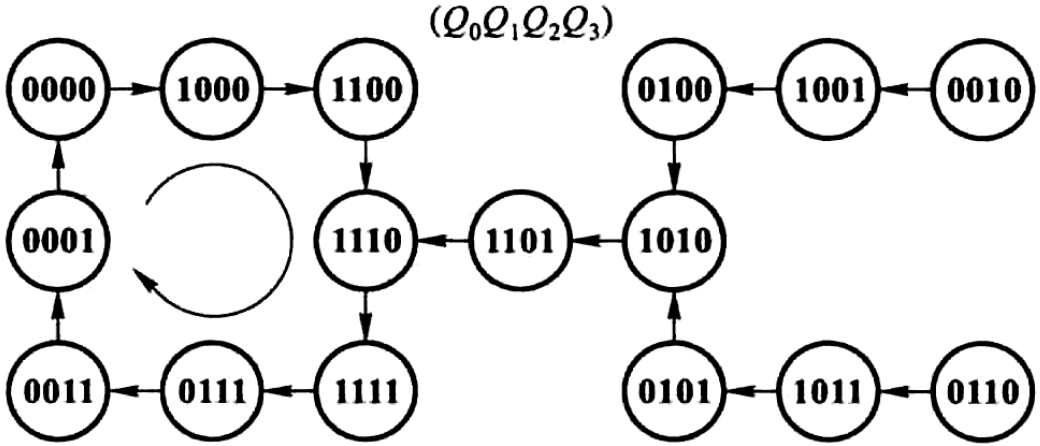

D_0
1D
C1
Q_0
FF_0
D_1
1D
C1
Q_1
FF_1
D_2
1D
C1
Q_2
FF_2
D_3
1D
C1
Q_3'
FF_3
CLK


有效循环
0000
1000
1100
0001
1110
0011
0111
1111
无效循环
1010
1101
0110
0100
1011
1001
0010
0101

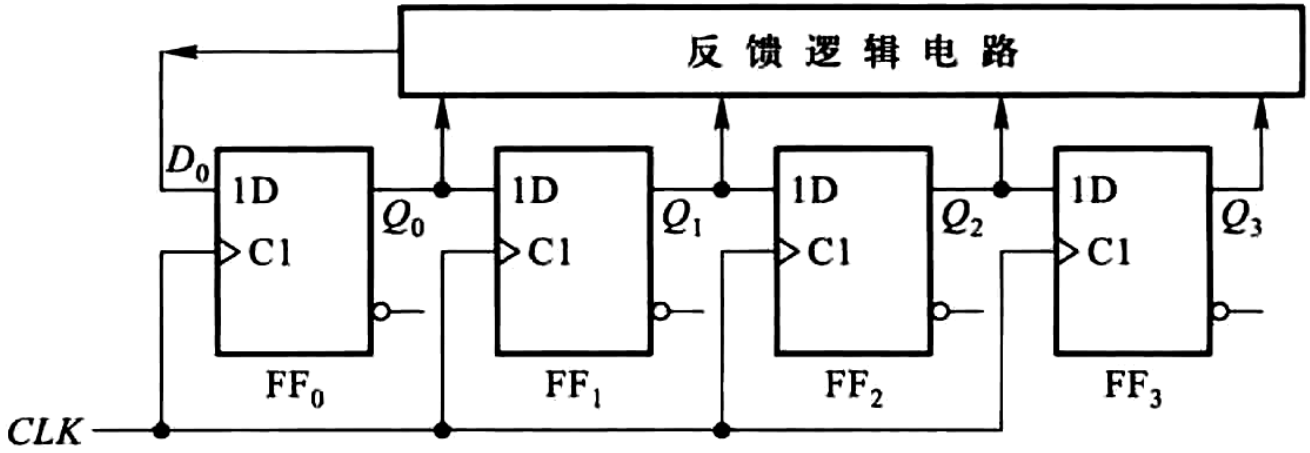
扭环型计数器（自启）



$$D_0 = Q_1\overline{Q_2} + \overline{Q_3}$$



移位寄存器型计数器的一般结构

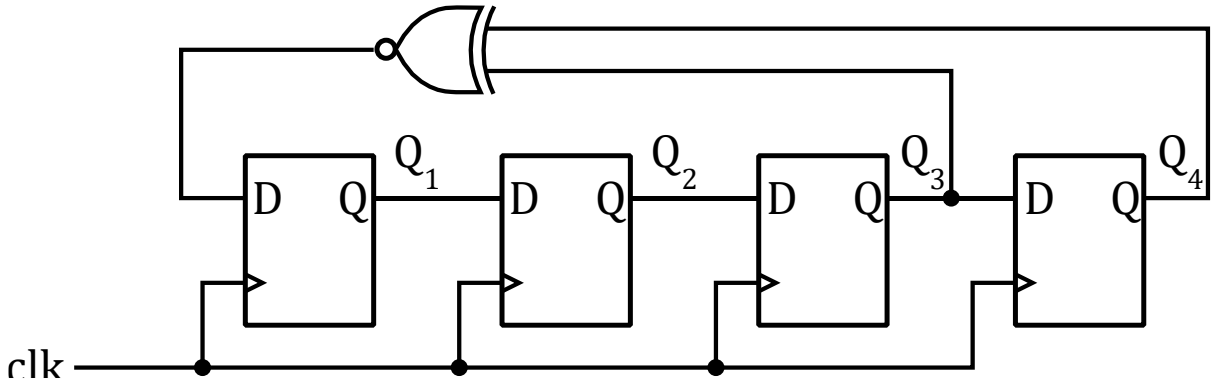


$$D_0 = F(Q_0, Q_1, \cdots, Q_{n-1})$$

如何能扩大移位寄存器的使用效率

理论上，N位寄存器可以在2^N个状态中循环（计数器）。

4位移位寄存器：伪随机数发生器（同或）

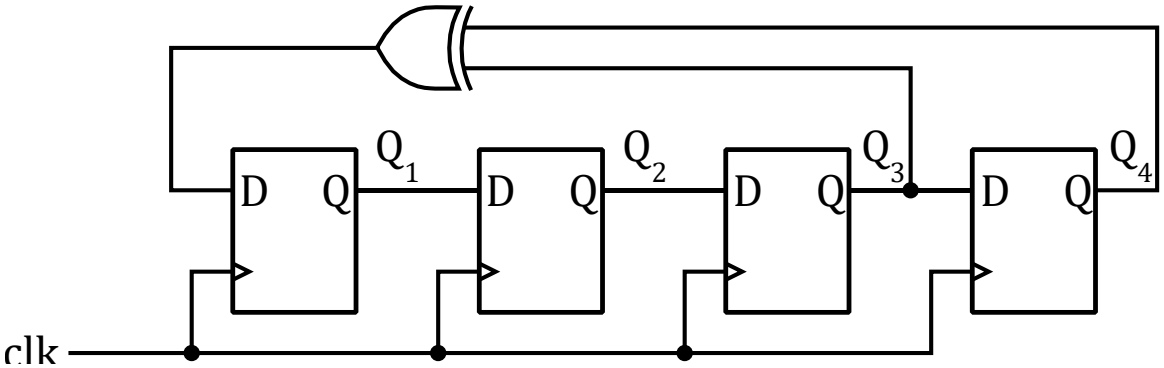


$$D_1 = \overline{Q_3 \oplus Q_4}$$

1	2	3	4	5	6	7	8
0000,	1000,	1100,	1110,	0111,	1011,	1101,	0110
0011,	1001,	0100,	1010,	0101,	0010,	0001,	0000
0	8	12	14	7	11	13	6
3	9	4	10	5	2	1	0

1111为非法状态

4位移位寄存器：伪随机数发生器（异或）



$$D_1 = Q_3 \oplus Q_4$$

1	2	3	4	5	6	7	8
0001,	1000,	0100,	0010,	1001,	1100,	0110,	1011
0101,	1010,	1101,	1110,	1111,	0111,	0011,	0001
1	8	4	2	9	12	6	11
5	10	13	14	15	7	3	1

0000为非法状态

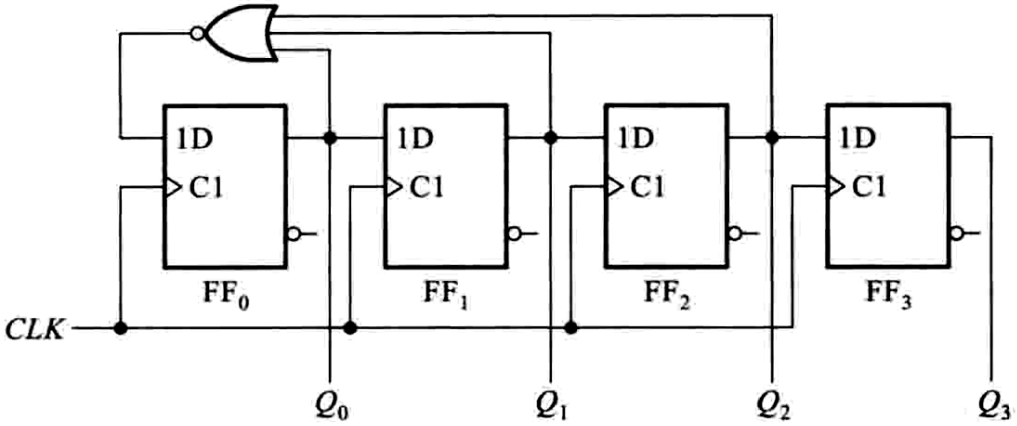
伪随机数发生器

- 使用特定的反馈函数，对于N位的移位寄存器，我们总是能生成长度为$2^N - 1$的伪随机数序列。

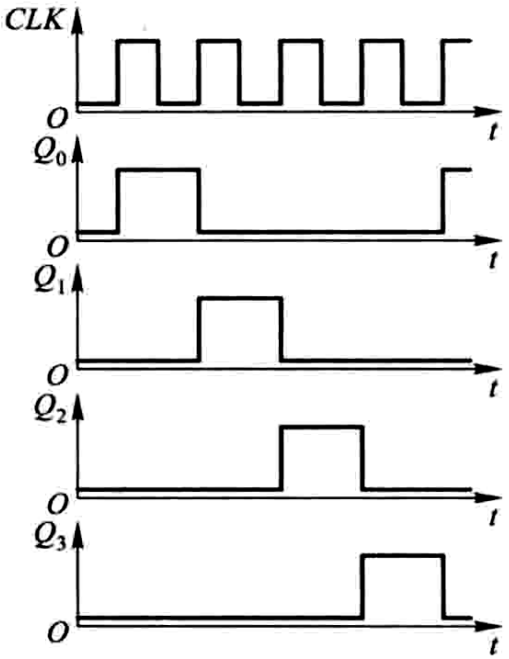
N	F()	N	F()
3	$Q_3 \oplus Q_2$	4	$Q_4 \oplus Q_3$
5	$Q_5 \oplus Q_3$	6	$Q_6 \oplus Q_5$
7	$Q_7 \oplus Q_6$	8	$Q_8 \oplus Q_6 \oplus Q_5 \oplus Q_4$
9	$Q_9 \oplus Q_5$	10	$Q_{10} \oplus Q_7$
11	$Q_{11} \oplus Q_9$	12	$Q_{12} \oplus Q_6 \oplus Q_4 \oplus Q_1$
13	$Q_{13} \oplus Q_4 \oplus Q_3 \oplus Q_1$	14	$Q_{14} \oplus Q_5 \oplus Q_3 \oplus Q_1$
15	$Q_{15} \oplus Q_{14}$	16	$Q_{16} \oplus Q_{15} \oplus Q_{13} \oplus Q_4$

https://www.xilinx.com/support/documentation/application_notes/xapp210.pdf

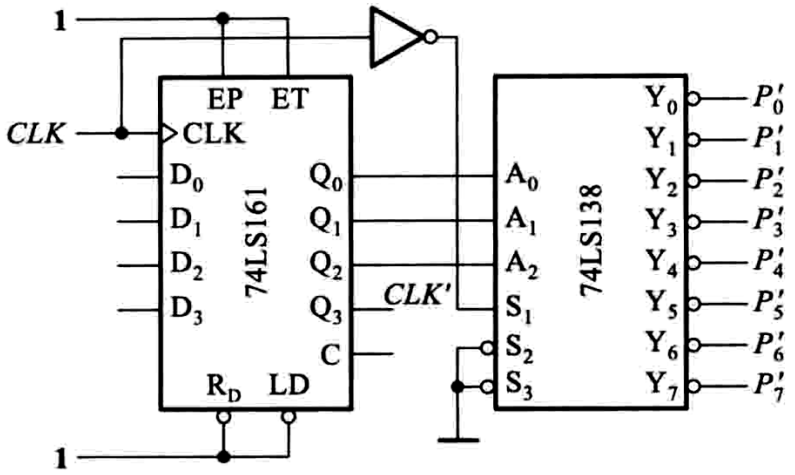
顺序脉冲发生器



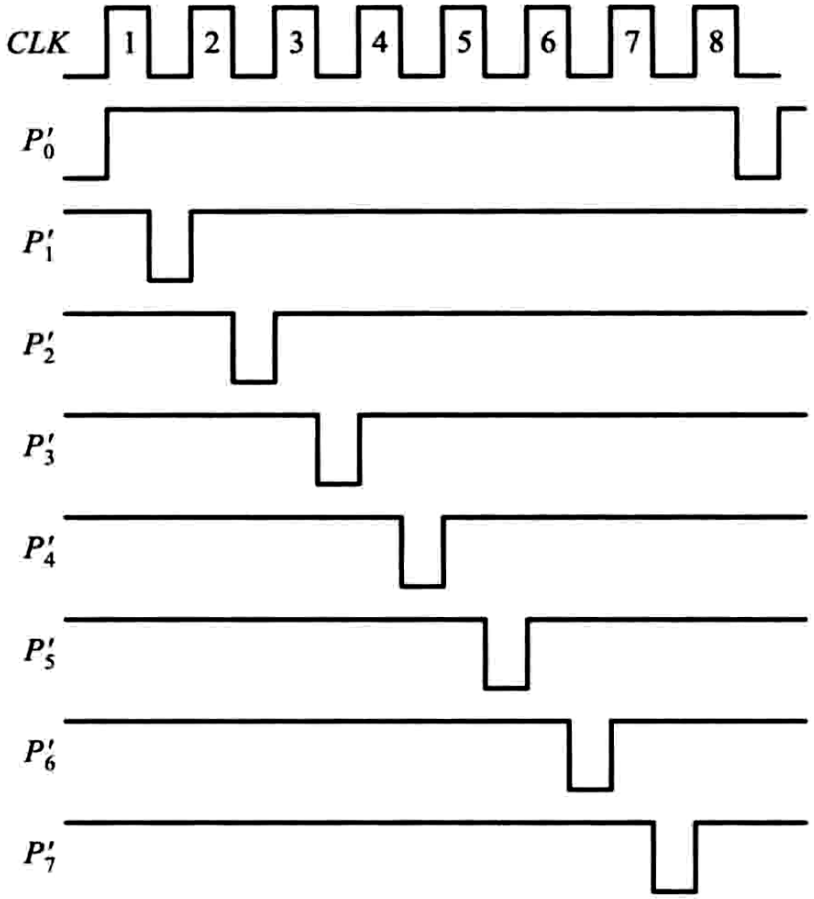
- 在N位信号上循环产生独热的脉冲信号



顺序脉冲发生器（使用集成电路模块）

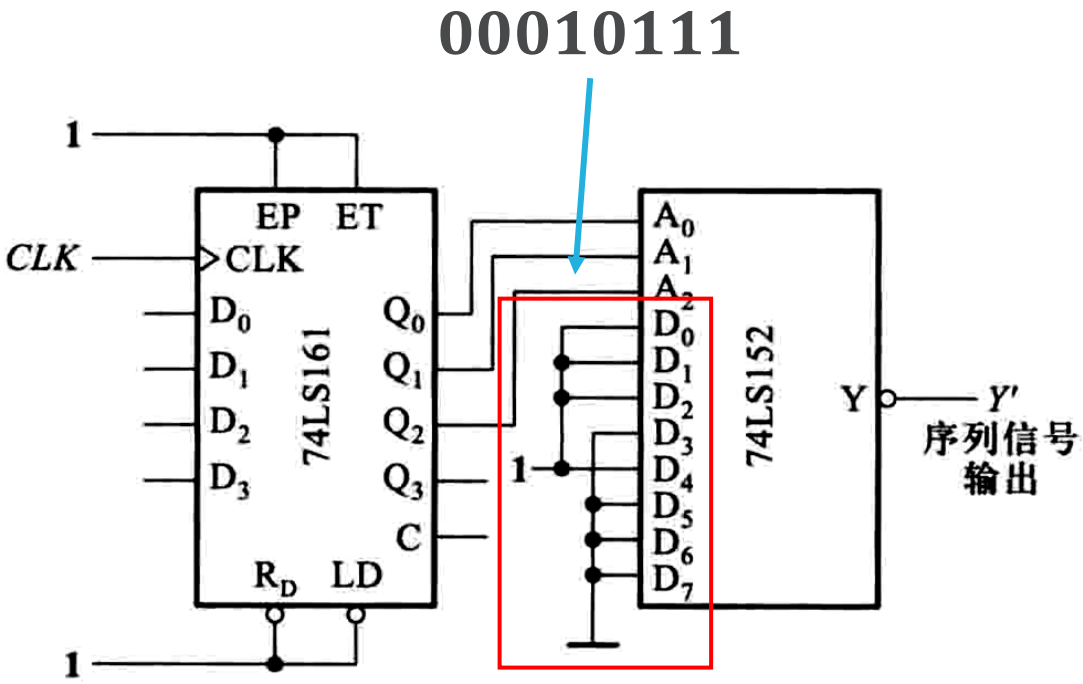


前端的74LS161被配置成自由计数器，计数的结果作为地址给74LS138译码器作为选择端选通一路输出。



序列信号发生器（使用集成电路模块）

- 周期性的输出一个预先设定的信号序列



74LS152：8路选择器。74LS161产生循环计数序列驱动选择器循环选通D_7至D_0。所以，将D_7至D_0连接成需要的信号序列。

- 周期性的输出一个预先设定的信号序列

00010111

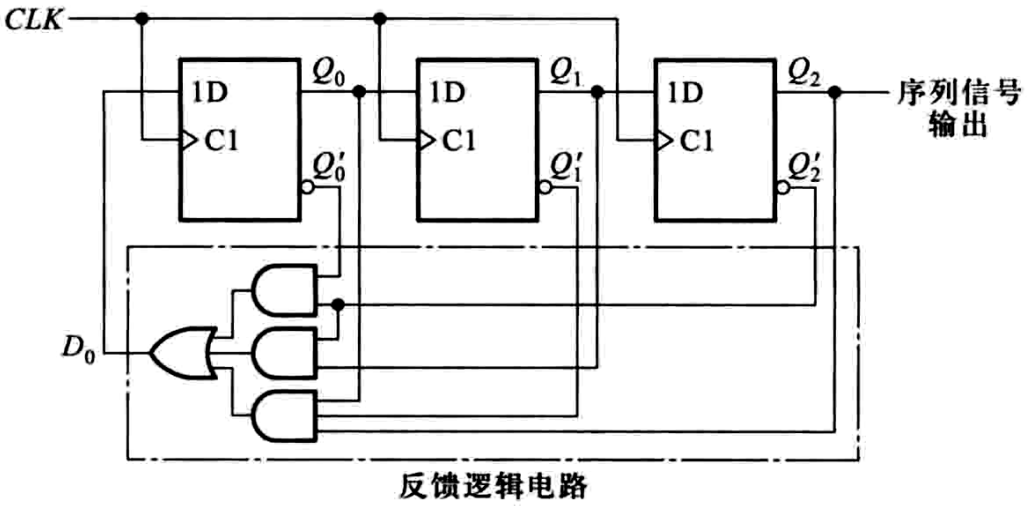
$D_0 = F(Q_2, Q_1, Q_0)$

$101 \rightarrow 011$

$D_0 = 1$

Q_2 \ Q_1Q_0	00	01	11	10
0	1	0	1	1
1	0	1	0	0

$$D_0 = \overline{Q_0} \cdot \overline{Q_2} + Q_1\overline{Q_2} + Q_0\overline{Q_1}Q_2$$



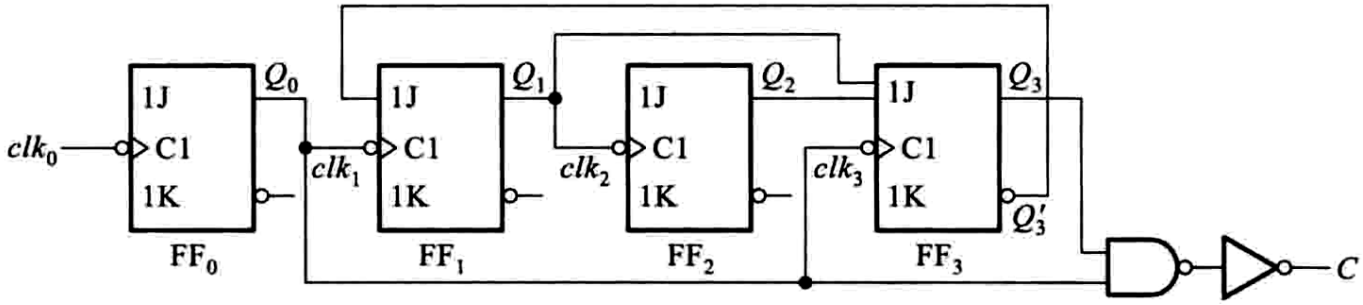
常见时序逻辑总结

- 移位寄存器
- 计数器
  - 二进制/16进制
  - 十进制
  - 任意进制 M < N
  - 任意进制 M > N
- 移位计数器
  - 伪随机数发生器
- 顺序脉冲发生器
- 序列信号发生器

这里讲的所有常见时序逻辑设计都是**同步逻辑**：所有触发器由同一个时钟信号驱动的电路。

- 异步时序逻辑电路
  - 异步时序逻辑电路的含义其实是有歧义的
  - 所有非同步逻辑电路都曾被称为异步电路
  - 教材中的异步逻辑电路：**一个电路中，触发器不使用同一个时钟。**
    - 非常狭义，仅仅是同步逻辑电路的一种违反常规的变种
  - 真实世界中可能被使用的异步电路
    - 完全不使用时钟
    - 使用基本的RS锁存器或者D-Latch为状态存储单元
    - 以事件为驱动的电路网络
    - 常见电路形式：
      delay-insensitive (DI), speed-independent (SI)
      bundled data, self-timed

异步时序逻辑电路分析示例



$$\begin{cases} J_0 = K_0 = 1 \\ J_1 = Q_3', \quad K_1 = 1 \\ J_2 = K_2 = 1 \\ J_3 = Q_1 Q_2, \quad K_3 = 1 \end{cases}$$

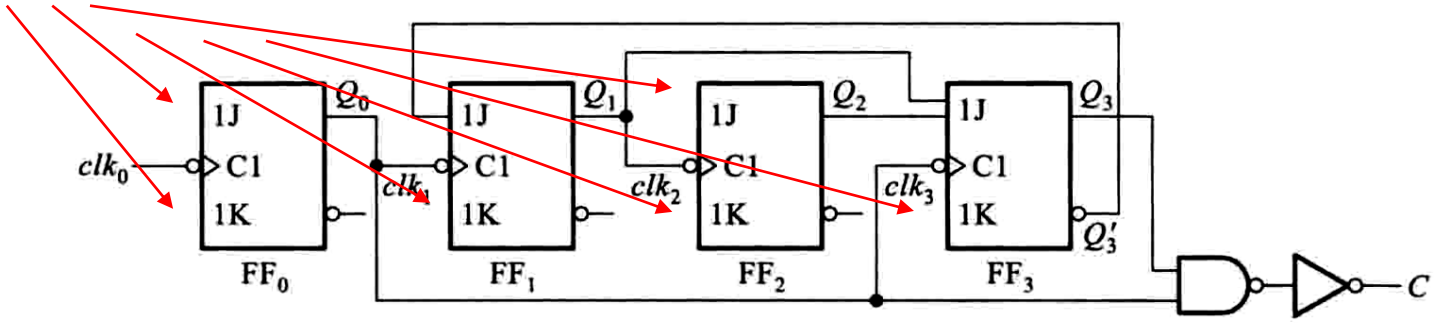
$$\begin{cases} Q_0^* = Q_0' \cdot clk_0 \\ Q_1^* = Q_3' Q_1' \cdot clk_1 \\ Q_2^* = Q_2' \cdot clk_2 \\ Q_3^* = Q_1 Q_2 Q_3' \cdot clk_3 \end{cases}$$

$$Q^* = JQ' + K'Q$$

$$C = Q_0 Q_3$$

异步时序逻辑电路分析示例

这里假设输入不接则为高电平是不对的！在电路设计上这个叫driverless input，综合器会报错，或者接任何逻辑，相当于无关项。



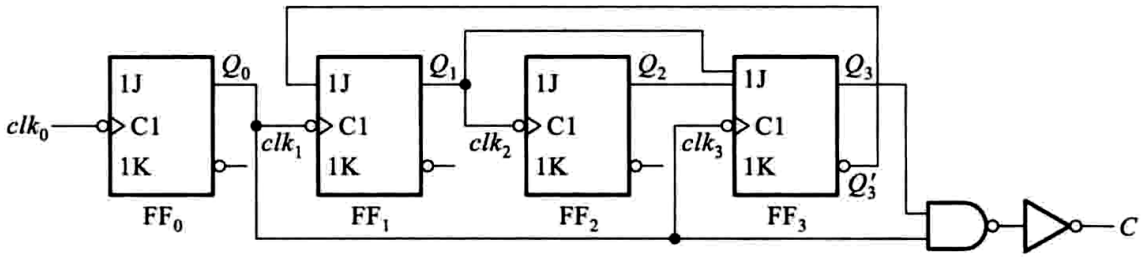
$$\begin{cases} J_0 = K_0 = 1 \\ J_1 = Q_3', \quad K_1 = 1 \\ J_2 = K_2 = 1 \\ J_3 = Q_1 Q_2, \quad K_3 = 1 \end{cases}$$

$$\begin{cases} Q_0^* = Q_0' \cdot clk_0 \\ Q_1^* = Q_3' Q_1' \cdot clk_1 \\ Q_2^* = Q_2' \cdot clk_2 \\ Q_3^* = Q_1 Q_2 Q_3' \cdot clk_3 \end{cases}$$

$$Q^* = JQ' + K'Q$$

$$C = Q_0 Q_3$$

异步时序逻辑电路分析示例

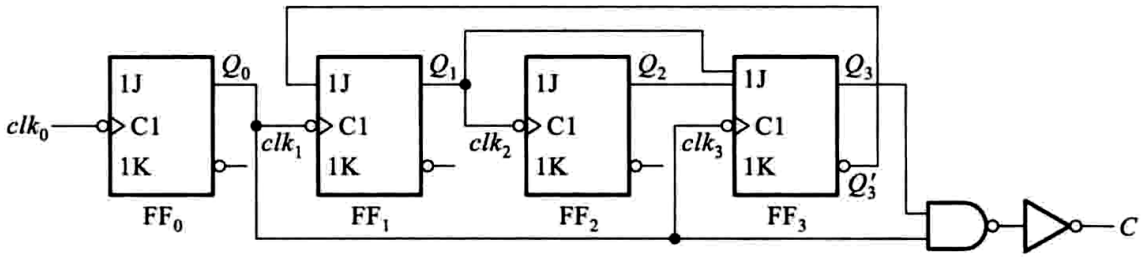


$$\begin{cases} Q_0^* = Q_0' \cdot clk_0 \\ Q_1^* = Q_3' Q_1' \cdot clk_1 \\ Q_2^* = Q_2' \cdot clk_2 \\ Q_3^* = Q_1 Q_2 Q_3' \cdot clk_3 \end{cases}$$

$$C = Q_0 Q_3$$

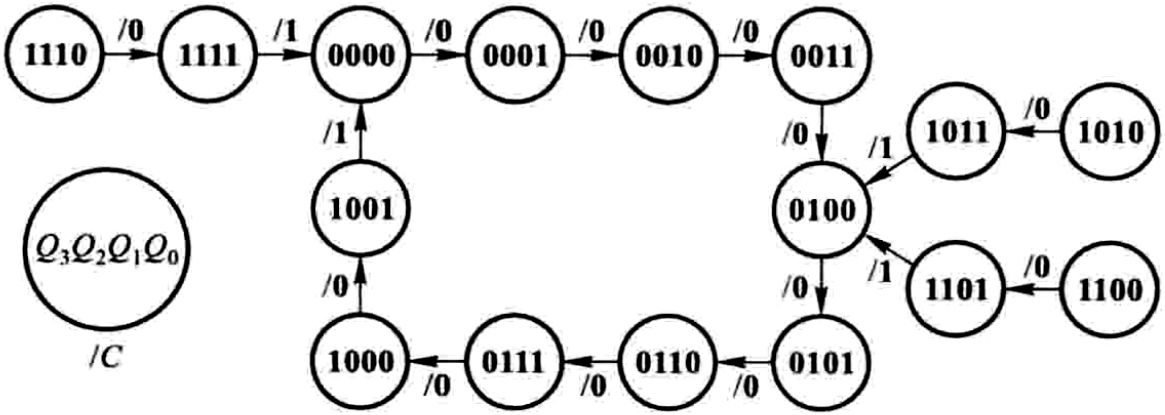
clk_0 的顺序	触发器状态				时钟信号				输出
	Q_3	Q_2	Q_1	Q_0	clk_3	clk_2	clk_1	clk_0	C
0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0
2	0	0	1	0	1	0	1	1	0
3	0	0	1	1	0	0	0	1	0
4	0	1	0	0	1	1	1	1	0
5	0	1	0	1	0	0	0	1	0
6	0	1	1	0	1	0	1	1	0
7	0	1	1	1	0	0	0	1	0
8	1	0	0	0	1	1	1	1	0
9	1	0	0	1	0	0	0	1	1
10	0	0	0	0	1	0	1	1	0

异步时序逻辑电路分析示例



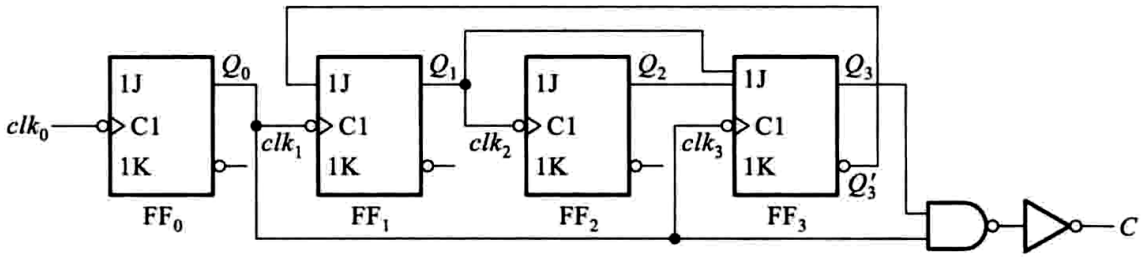
$$\begin{cases} Q_0^* = Q_0' \cdot clk_0 \\ Q_1^* = Q_3' Q_1' \cdot clk_1 \\ Q_2^* = Q_2' \cdot clk_2 \\ Q_3^* = Q_1 Q_2 Q_3' \cdot clk_3 \end{cases}$$

$$C = Q_0 Q_3$$



看起来用公式可以仔细并全面地分析该电路，但是，状态转换图/表其实只是对电路的一个仿真，本质和波形并无二样。之所以状态转移图能够完备地表达出所有地状态转换，那只是因为该电路的时钟逻辑并没有导致状态转换关系的爆炸。

异步时序逻辑电路分析示例



$$\begin{cases} Q_0^* = Q_0' \cdot clk_0 \\ Q_1^* = Q_3' Q_1' \cdot clk_1 \\ Q_2^* = Q_2' \cdot clk_2 \\ Q_3^* = Q_1 Q_2 Q_3' \cdot clk_3 \end{cases} \qquad C = Q_0 Q_3$$

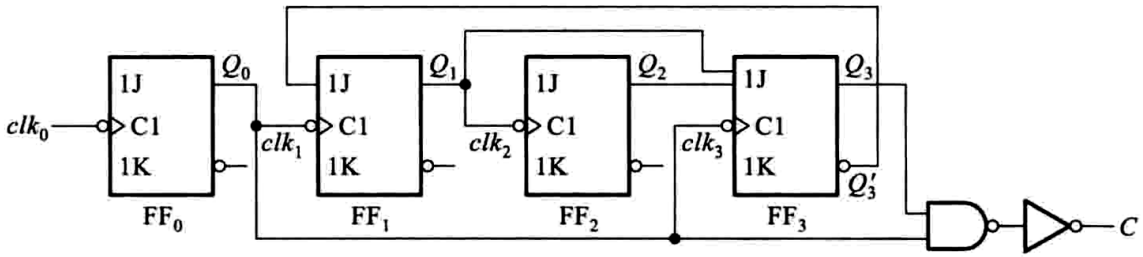
咱们来分析一下该电路的时钟逻辑（下降沿时钟事件）

$1 \rightarrow clk_0$
$Q_0 clk_0 \rightarrow clk_1, clk_3$
$Q_1 \overline{Q_1^*} clk_1 = Q_1 clk_1 \rightarrow clk_2$

我们用Q^t代表在$t(\mathrm{clk}_0)$时刻Q的值
$Q_2^t = \overline{Q_2^{t-1}} clk_2 = \overline{Q_2^{t-1}} Q_1^{t-2} clk_1 = \overline{Q_2^{t-1}} Q_1^{t-2} Q_0^{t-3} clk_0$

$xxx1 \rightarrow xx1x \rightarrow x0xx \rightarrow x1xx$

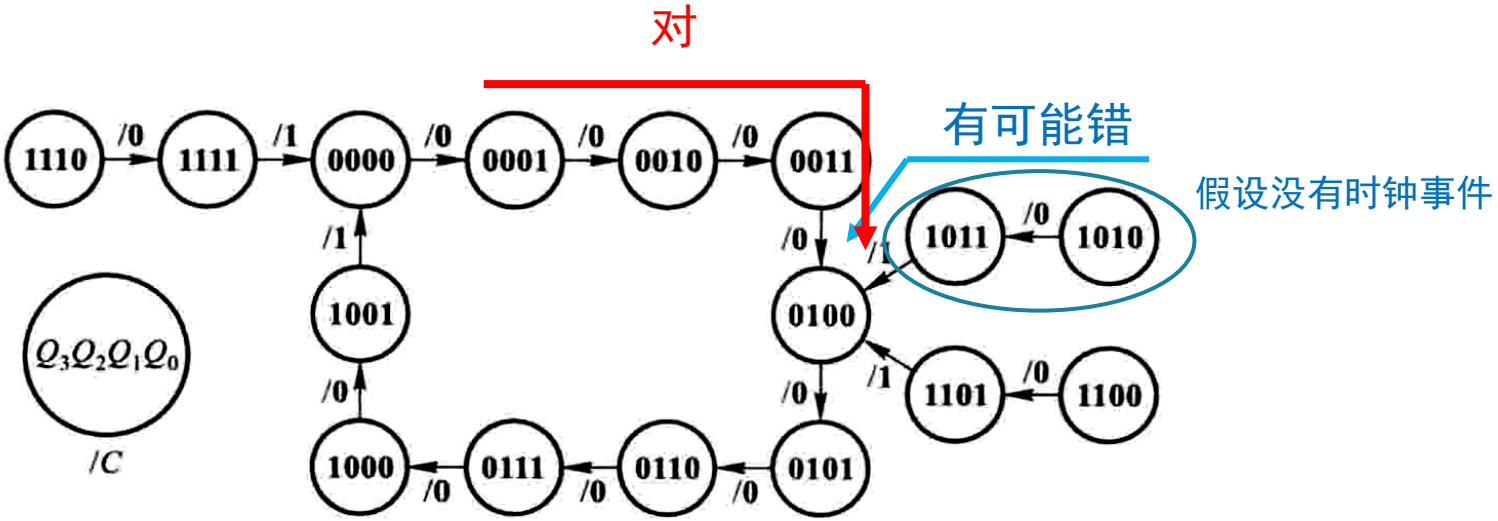
异步时序逻辑电路分析示例



$$\begin{cases} Q_0^* = Q_0' \cdot clk_0 \\ Q_1^* = Q_3' Q_1' \cdot clk_1 \\ Q_2^* = Q_2' \cdot clk_2 \\ Q_3^* = Q_1 Q_2 Q_3' \cdot clk_3 \end{cases}$$

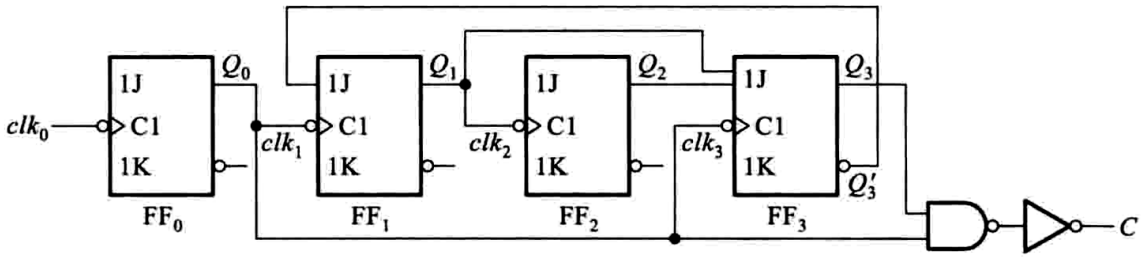
$$C = Q_0 Q_3$$

$xxx1 \rightarrow xx1x \rightarrow x0xx \rightarrow x1xx$



一个状态的到达，是由前面多个时刻的状态决定，而非同步时序逻辑中的上一时刻。

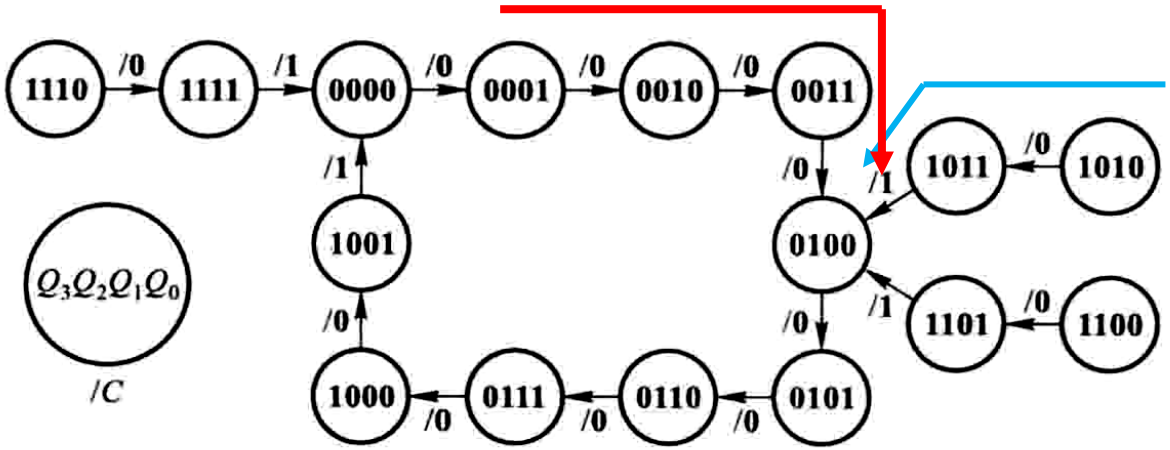
异步时序逻辑电路分析示例

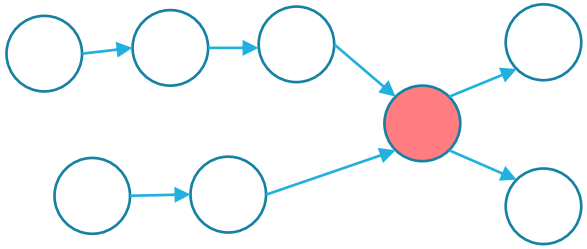


$$\begin{cases} Q_0^* = Q_0' \cdot clk_0 \\ Q_1^* = Q_3' Q_1' \cdot clk_1 \\ Q_2^* = Q_2' \cdot clk_2 \\ Q_3^* = Q_1 Q_2 Q_3' \cdot clk_3 \end{cases}$$

$$C = Q_0 Q_3$$

$xxx1 \rightarrow 0x1x \rightarrow x0xx \rightarrow x1xx$





更糟糕的问题，由于一个状态可能由多个路径到达，根据到达的路径不同，该状态的下一状态可能出现分叉。

异步时序逻辑电路分析总结(附加)

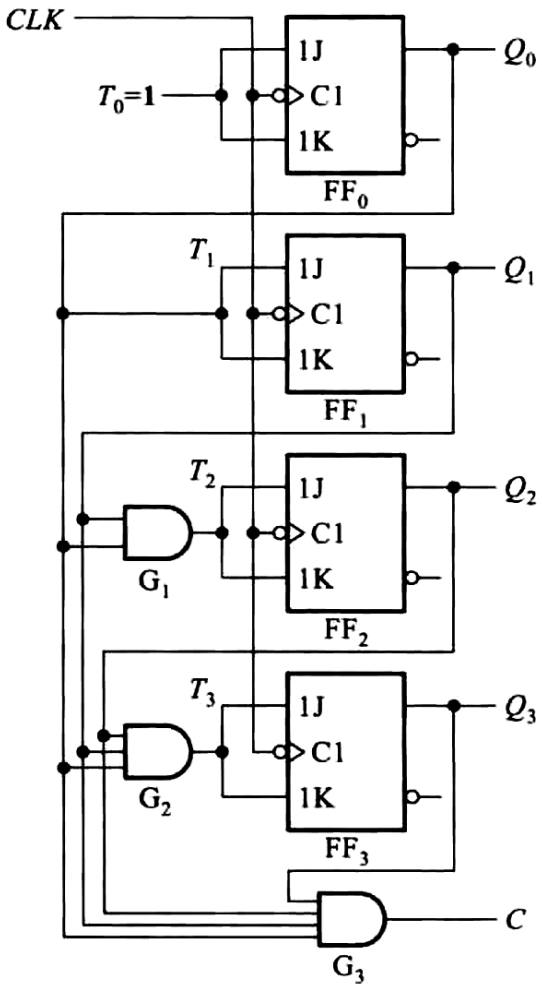
- 异步时序逻辑电路的分析
  - 驱动方程可以帮助分析状态的转换
  - 状态转换图/表只是在某一种初始条件下对系统的仿真
  - 状态很可能取决于当前时刻之前的历史路径而非上一时刻的状态
  - 状态转换图在穷举的情况下可能出现分叉，分叉的原因是到达同一状态的历史路径不同
  - 异步电路实际是事件驱动电路（event driven）

- 反观同步时序逻辑电路
  - 正是由于所有的触发器由同一个时钟驱动，所有的状态都由上一个时刻的状态完全确定，并不需要考虑历史路径

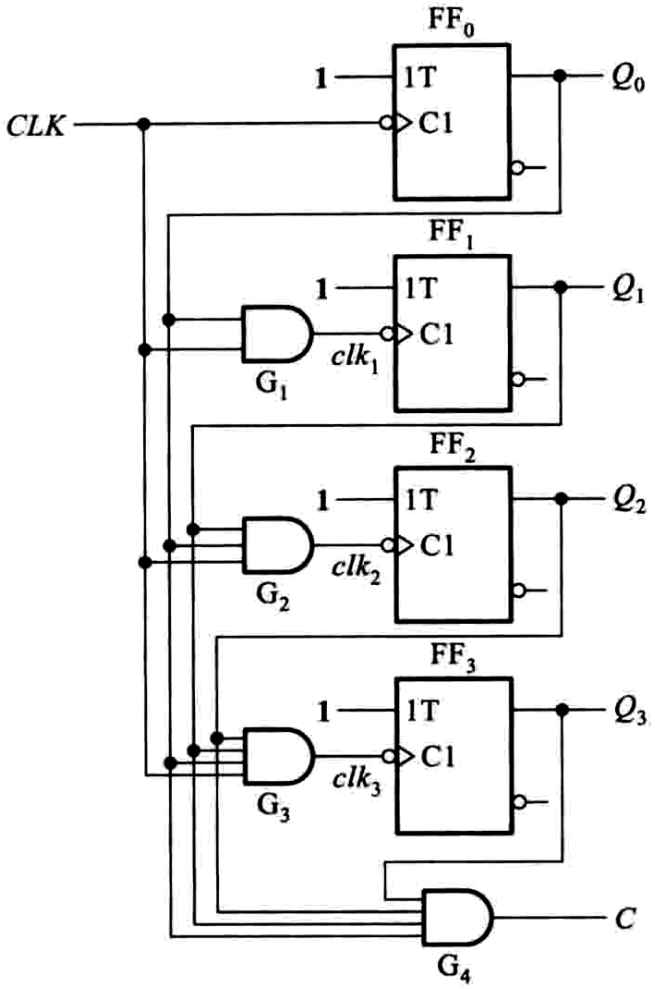
$$Q^* = f(Q)clk$$

（常用）异步时序逻辑电路 —计数器

同步

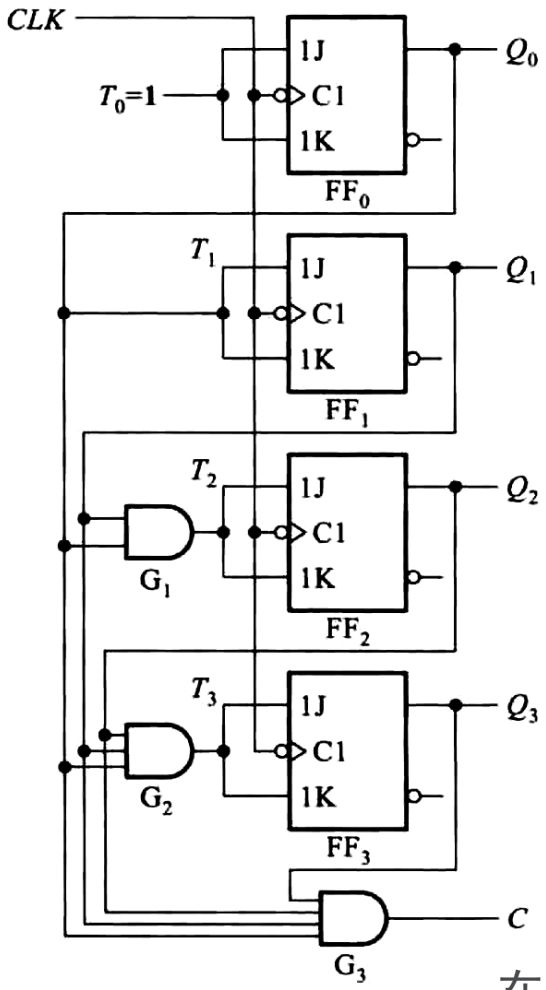


异步

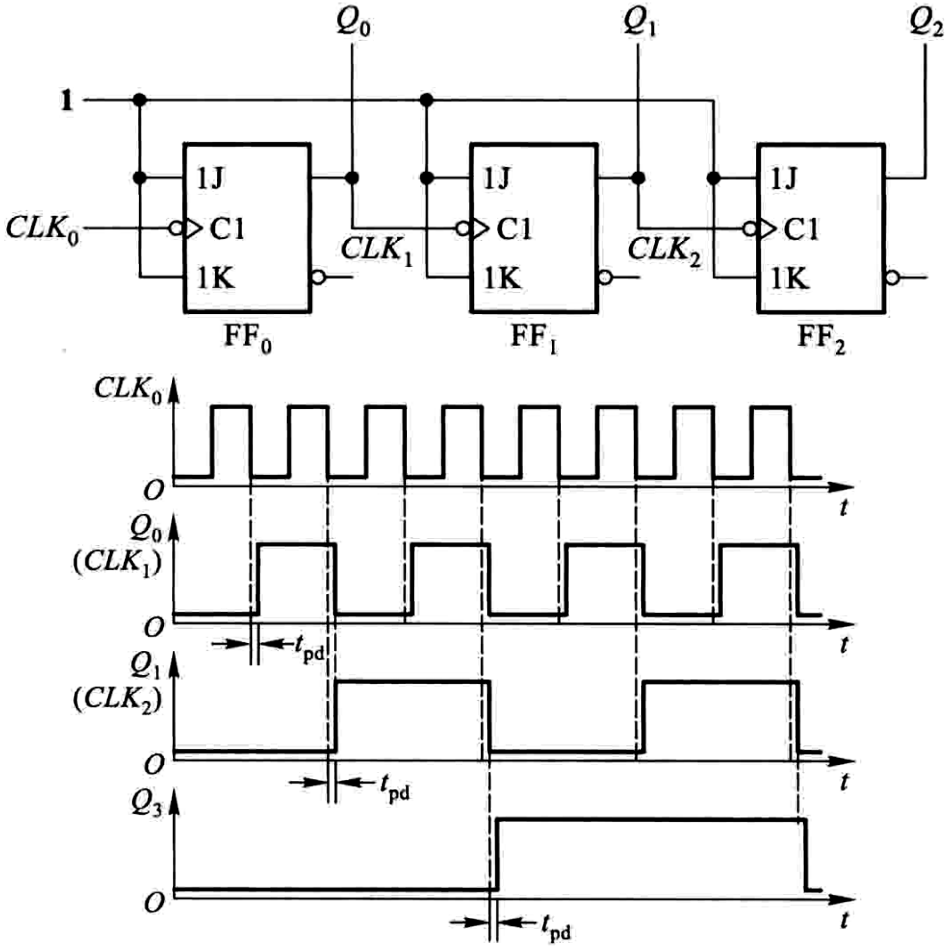


（常用）异步时序逻辑电路 —计数器（加法）

同步



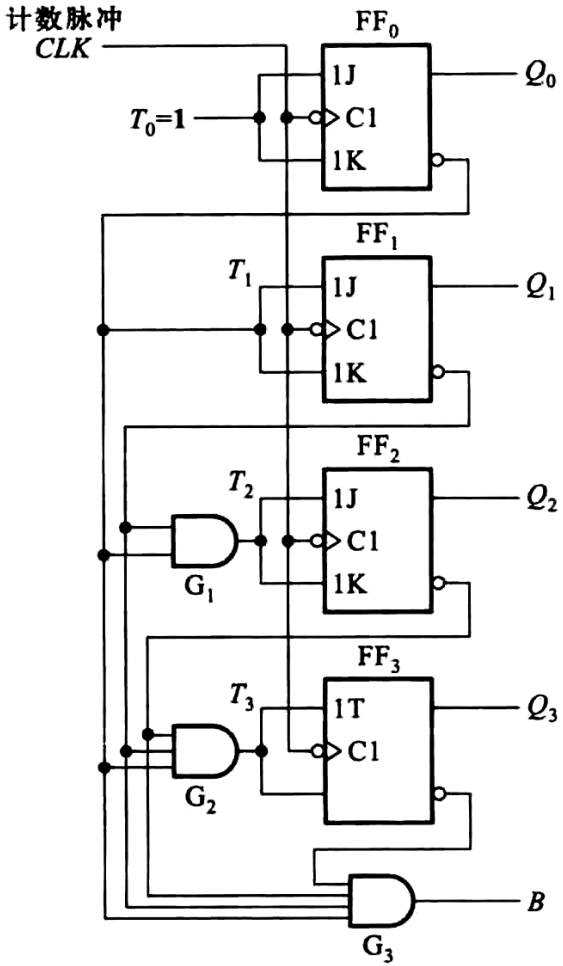
异步



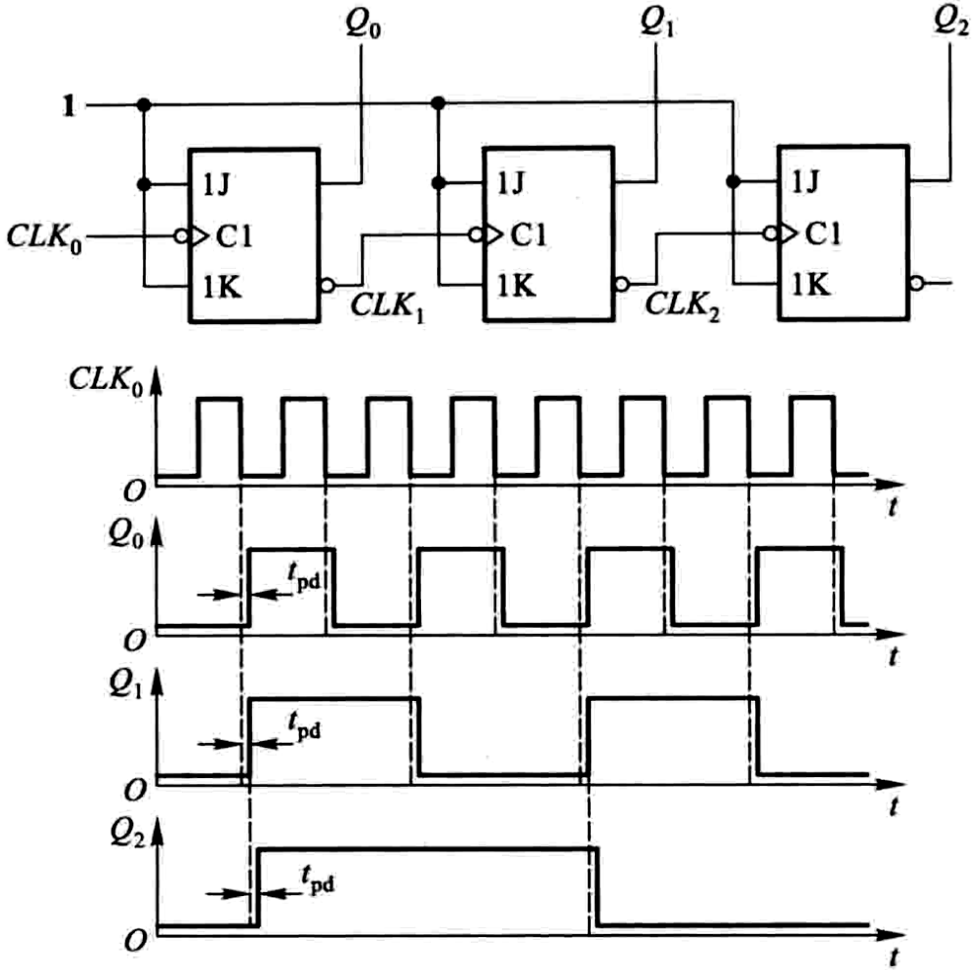
在时钟信号上加组合逻辑的弊病之一：**扩大时钟偏差**

（常用）异步时序逻辑电路 —计数器（减法）

同步



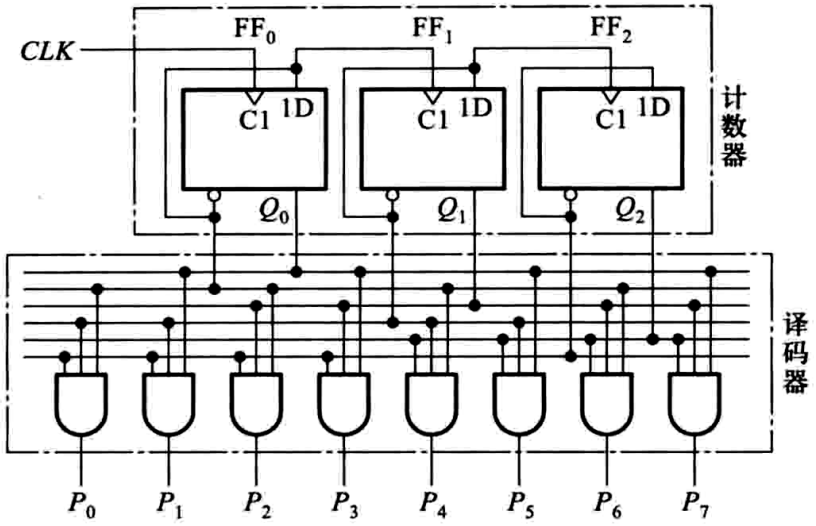
异步

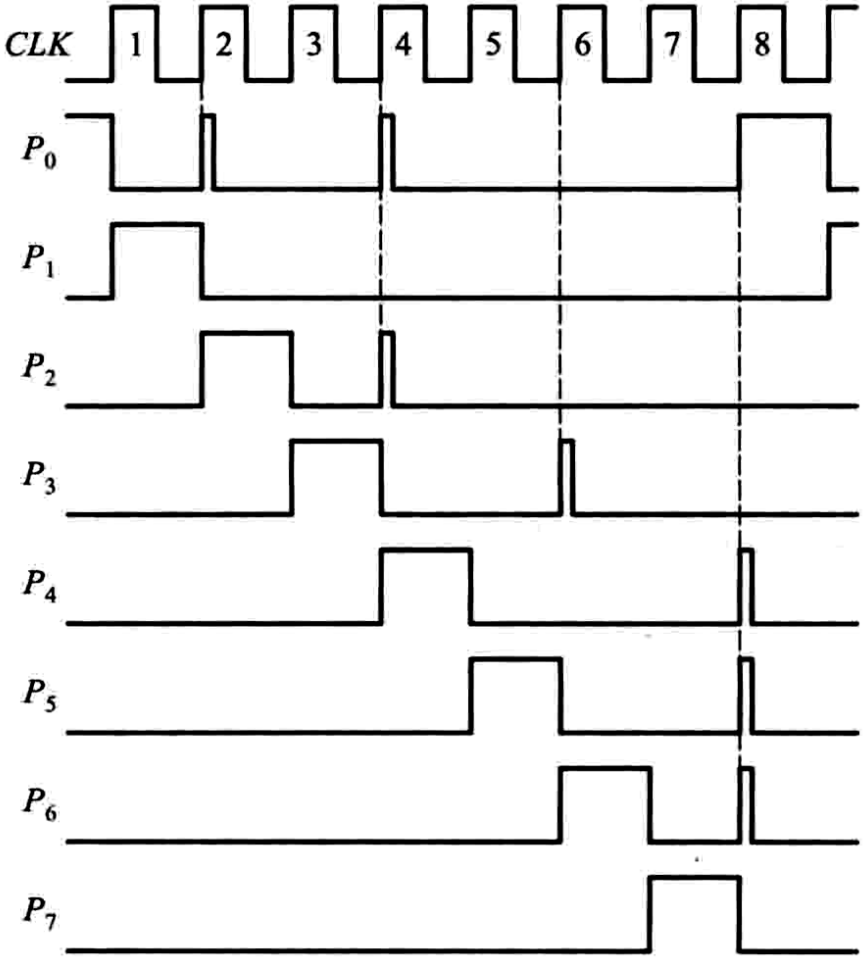


在时钟信号上加组合逻辑的弊病之一：**扩大时钟偏差**

（常用）异步时序逻辑电路—顺序脉冲发生

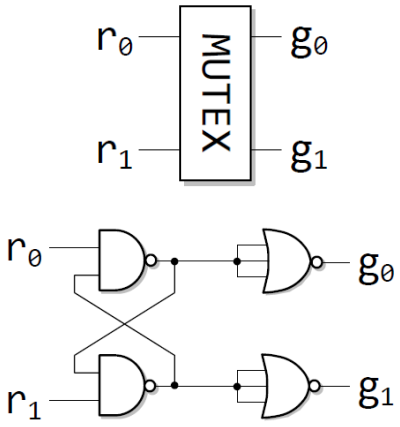
- 异步时序逻辑电路的时钟偏差导致了毛刺

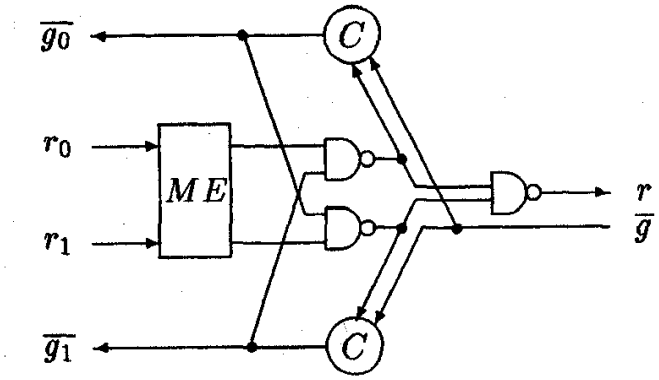




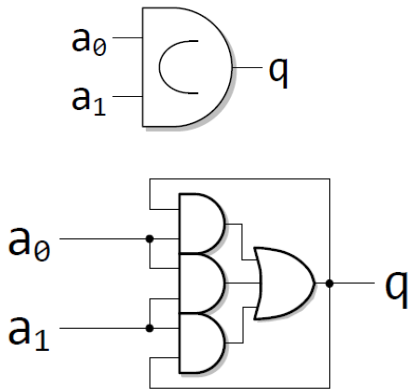
当m>n时，Q_m的变化滞后于Q_n。

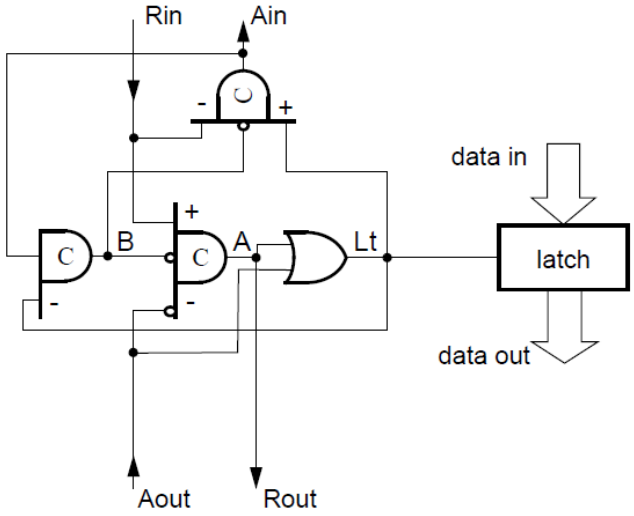
异步电路的例子(附加)



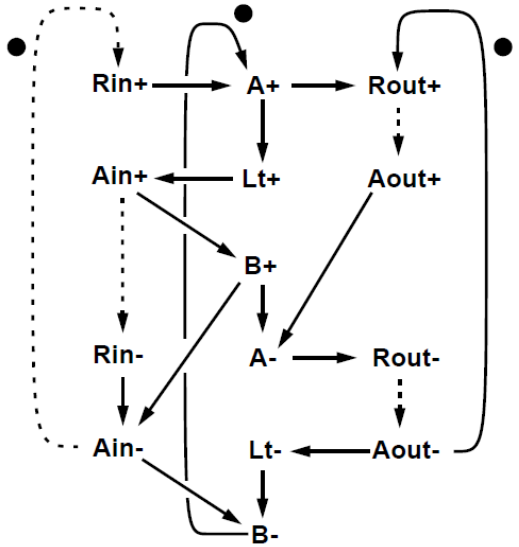


树型仲裁器单元





Latch控制单元



Latch控制单元的STG

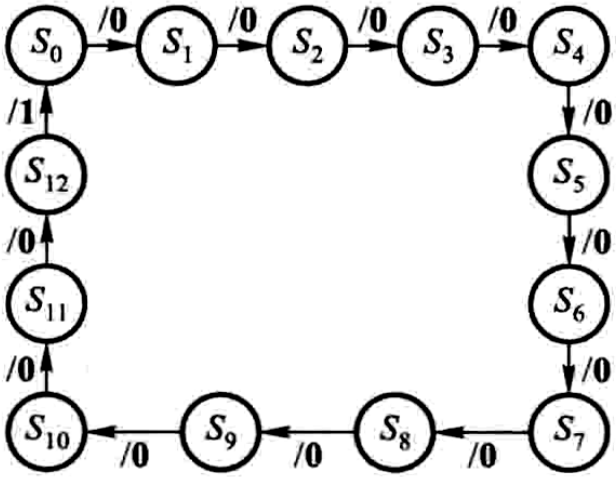
（常用）异步时序逻辑电路总结（附加）

- （常用）异步时序逻辑电路
  - 不需要严格逻辑推导，能够由现有同步逻辑电路直接推导出的异步逻辑电路形式
  - 仍然使用触发器，使用前一级触发器的输出驱动下一级触发器
  - 很难严格推导/遍历状态空间
  - 当考虑触发器的传输延时时，异步电路造成了时钟偏差
  - 时钟偏差可能导致毛刺
- 当前大规模集成电路的标准实现方法
  - 规避异步逻辑电路的使用
  - 大部分的功能单元为同步时序逻辑电路
  - 为了功耗原因可能使用多时钟
  - 不同时钟驱动的触发器之间的连接需要跨时钟域特殊处理
  - 特别的电路可能局部使用异步电路设计（**其设计方法和同步逻辑电路的设计方法区别巨大！**）

同步时序逻辑电路的设计方法

- 分析问题，得到问题的状态转换图/表
- 状态化简
  - 如果两个状态在相同的输入下有相同的输出，则可以被合并为同一个状态
- 状态分配
  - 对所有状态进行编码
  - 二进制编码
    - 使用n位编码编码m个不同状态：$2^{n-1} < M \leq 2^n$
  - 独热编码
    - 使用m位编码编码m个不同状态
- 选定触发器类型，确定系统方程
- 电路实现
- 检查电路是否可以自启

时序逻辑电路设计：13计数器



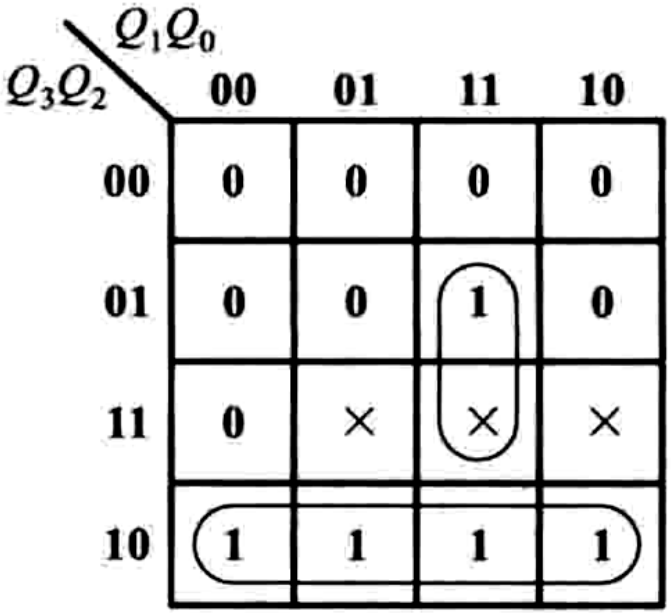
- 使用二进制编码，需要4个编码比特：$2^3 < 13 < 2^4$

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0001/0	0010/0	0100/0	0011/0
01	0101/0	0110/0	1000/0	0111/0
11	0000/1	××××/×	××××/×	××××/×
10	1001/0	1010/0	1100/0	1011/0

时序逻辑电路设计：13计数器

- 拆分总体卡诺图，得到每一个状态位的卡诺图，并化简卡诺图，得到每一个状态位的状态方程

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0001/0	0010/0	0100/0	0011/0
01	0101/0	0110/0	1000/0	0111/0
11	0000/1	××××/×	××××/×	××××/×
10	1001/0	1010/0	1100/0	1011/0

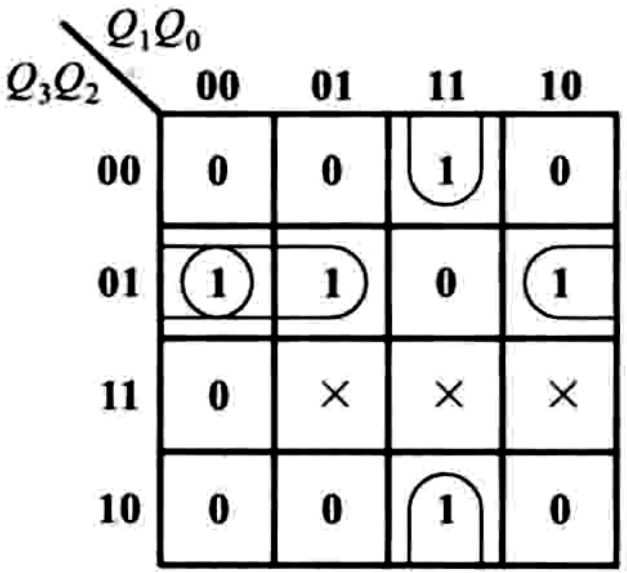


$$Q_3^* = Q_2Q_1Q_0 + Q_3\overline{Q_2}$$

时序逻辑电路设计：13计数器

- 拆分总体卡诺图，得到每一个状态位的卡诺图，并化简卡诺图，得到每一个状态位的状态方程

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0001/0	0010/0	0100/0	0011/0
01	0101/0	0110/0	1000/0	0111/0
11	0000/1	××××/×	××××/×	××××/×
10	1001/0	1010/0	1100/0	1011/0



$$Q_2^* = \overline{Q_2}Q_1Q_0 + \overline{Q_3}Q_2\overline{Q_0} + \overline{Q_3}Q_2\overline{Q_1}$$

Q_1，Q_0和Y自行推导。

时序逻辑电路设计：13计数器

- 从推导出的状态方程，根据选择的触发器(JK)，推导出触发器驱动方程

$$\begin{cases} Q_3^* = Q_3Q_2' + Q_2Q_1Q_0 \\ Q_2^* = Q_3'Q_2Q_1' + Q_3'Q_2Q_0' + Q_2'Q_1Q_0 \\ Q_1^* = Q_1'Q_0 + Q_1Q_0' \\ Q_0^* = Q_3'Q_0' + Q_2'Q_0' \end{cases}$$

$$C = Q_3Q_2$$

JK触发器的特性方程：

$$Q^* = J\bar{Q} + \bar{K}Q$$

关于Q的ite分解

$Q_0Q_1Q_2Q_3$无关项

$$\begin{cases} Q_3^* = Q_3Q_2' + Q_2Q_1Q_0(Q_3 + Q_3') = (Q_2Q_1Q_0)Q_3' + Q_2'Q_3 \\ Q_2^* = (Q_0Q_1)Q_2' + (Q_3'(Q_1Q_0)')Q_2 \\ Q_1^* = Q_0Q_1' + Q_0'Q_1 \\ Q_0^* = (Q_3' + Q_2')Q_0' + 1' \cdot Q_0 = (Q_3Q_2)'Q_0' + 1'Q_0 \end{cases}$$

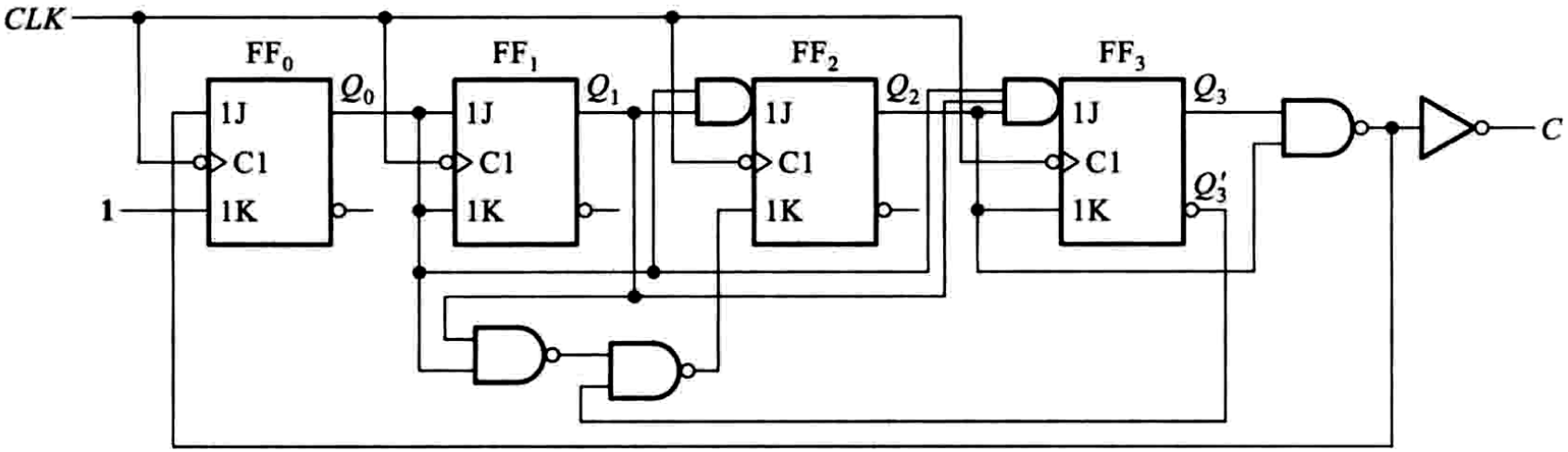
$$\begin{cases} J_3 = Q_2Q_1Q_0, & K_3 = Q_2 \\ J_2 = Q_1Q_0, & K_2 = (Q_3'(Q_1Q_0)')' \\ J_1 = Q_0, & K_1 = Q_0 \\ J_0 = (Q_3Q_2)', & K_0 = 1 \end{cases}$$

时序逻辑电路设计：13计数器

- 根据触发器驱动方程，设计电路

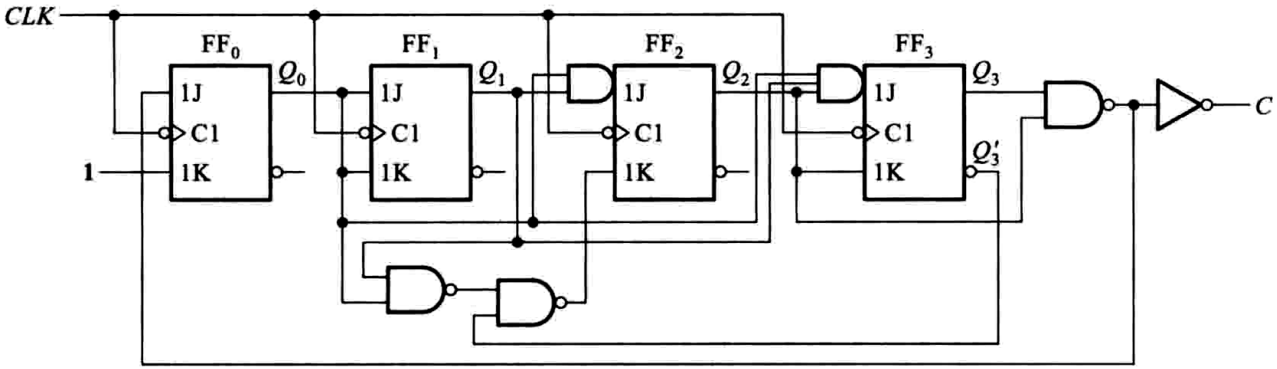
$$\begin{cases} J_3 = Q_2 Q_1 Q_0, & K_3 = Q_2 \\ J_2 = Q_1 Q_0, & K_2 = (Q_3'(Q_1 Q_0)')' \\ J_1 = Q_0, & K_1 = Q_0 \\ J_0 = (Q_3 Q_2)', & K_0 = 1 \end{cases}$$

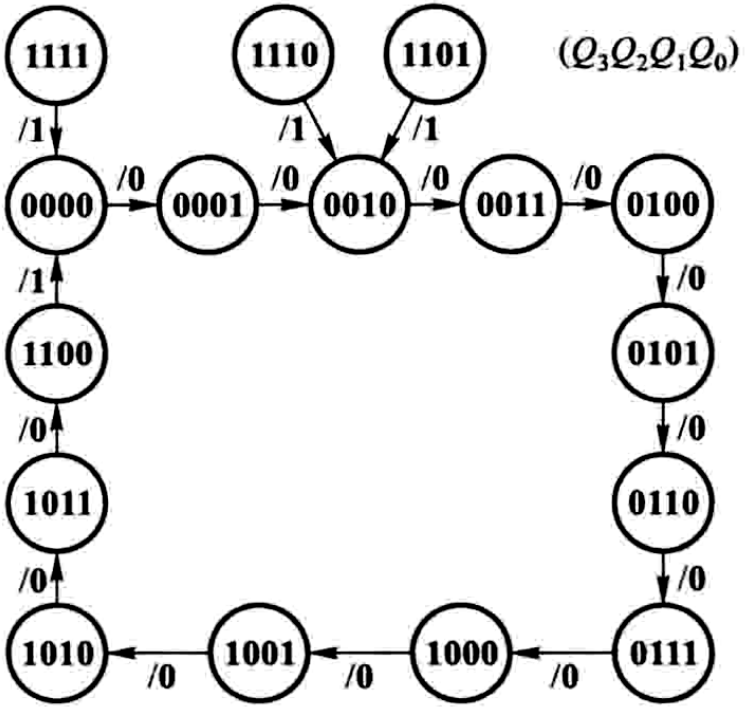
$$C = Q_3 Q_2$$



时序逻辑电路设计：13计数器

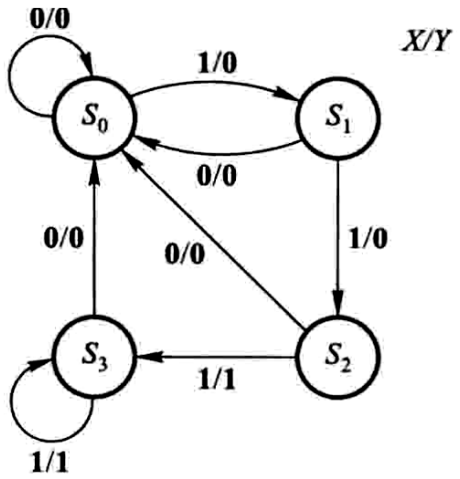
- 画出完整的状态转换图，检查是否能够自启



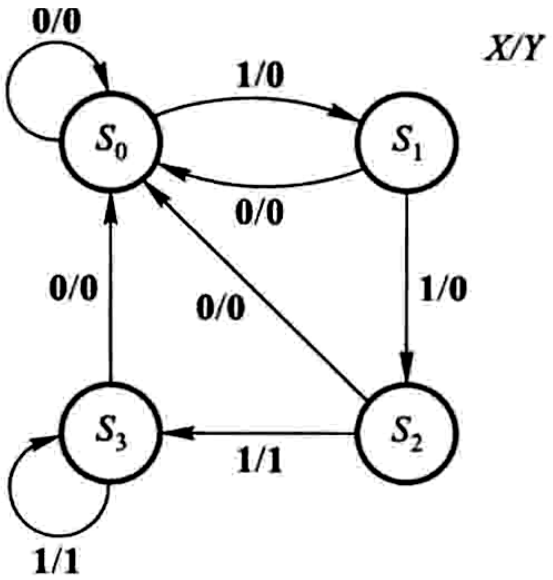


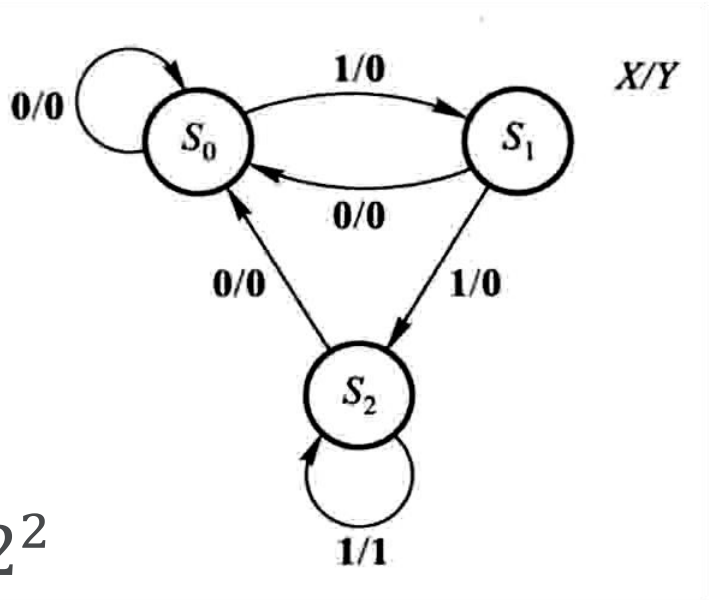
- 设计一个串行数据检测器，当连续输入3个或以上的1时输出1，其他为0。
  - 定义状态：S_n表示连续检测到n个输入1周期

S / S^*/Y / X	S_0	S_1	S_2	S_3
0	$S_0/0$	$S_0/0$	$S_0/0$	$S_0/0$
1	$S_1/0$	$S_2/0$	$S_3/1$	$S_3/1$



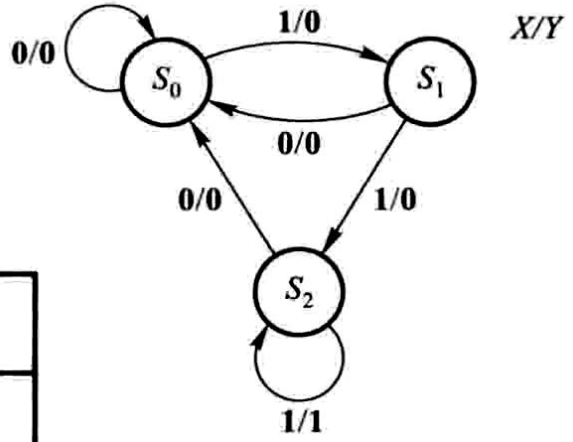
- 简化状态转换图
  - S_2和S_3的输入和输出一致，因而可以合并





$$2^1 < 3 < 2^2$$

- 拆分整体卡诺图



X \ Q_1Q_0	00	01	11	10
0	00/0	00/0	××/×	00/0
1	01/0	10/0	××/×	10/1

X \ Q_1Q_0	00	01	11	10
0	0	0	×	0
1	0	1	×	1

X \ Q_1Q_0	00	01	11	10
0	0	0	×	0
1	1	0	×	0

X \ Q_1Q_0	00	01	11	10
0	0	0	×	0
1	0	0	×	1

$$Q_1^* = X(Q_1 + Q_0) = XQ_0\overline{Q_1} + XQ_1$$

$$Q_0^* = X\overline{Q_1} \cdot \overline{Q_0} = X\overline{Q_1} \cdot \overline{Q_0} + 0 \cdot Q_0$$

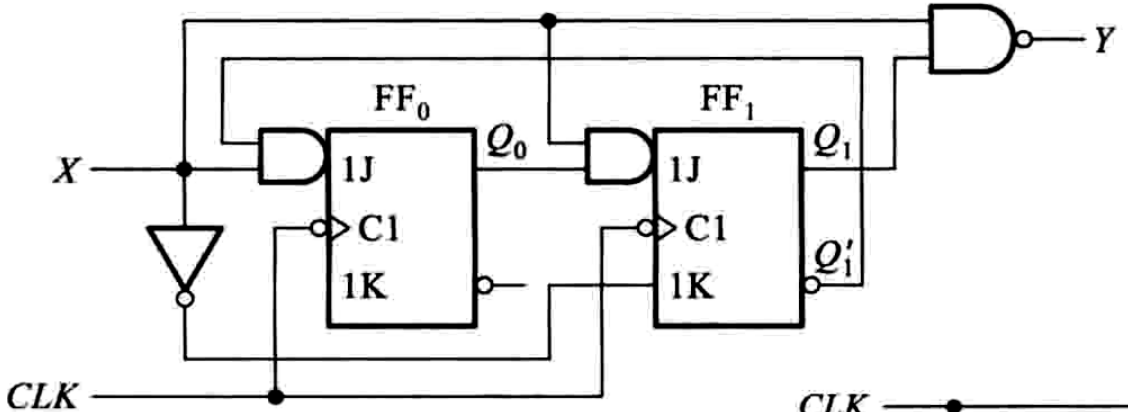
$$Y = XQ_1$$

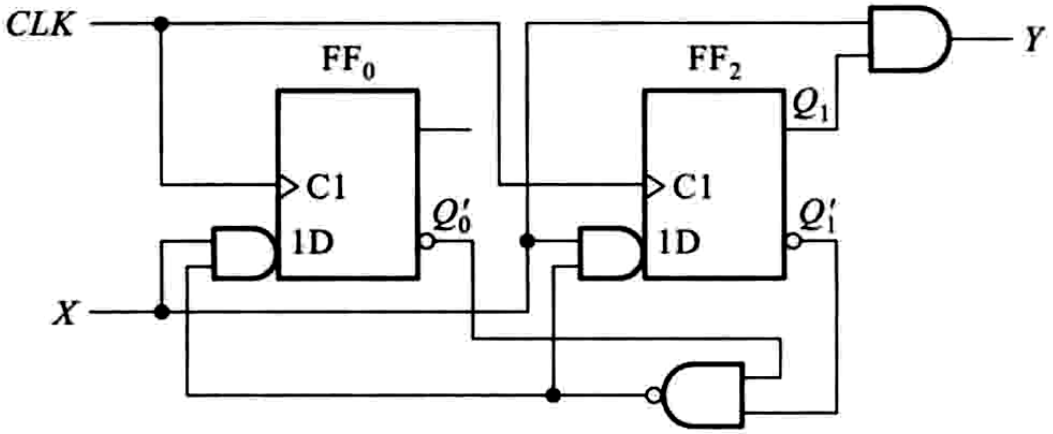
构建电路实现

$$Q_1^* = X(Q_1 + Q_0) = XQ_0\overline{Q_1} + XQ_1$$

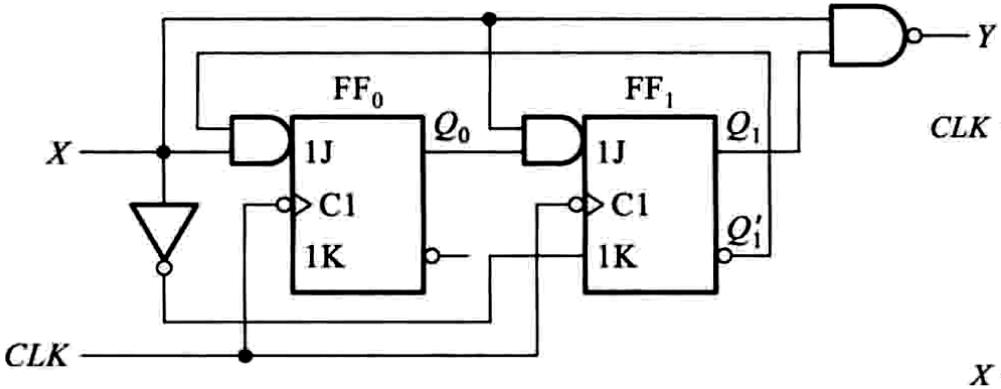
$$Q_0^* = X\overline{Q_1} \cdot \overline{Q_0} = X\overline{Q_1} \cdot \overline{Q_0} + 0 \cdot Q_0$$

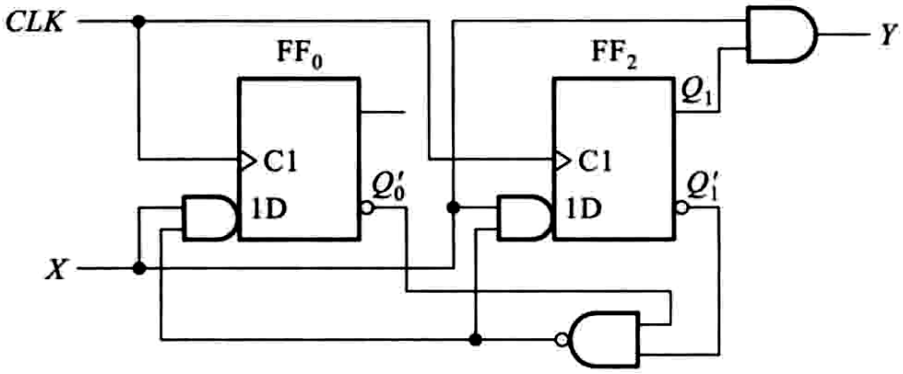
$$Y = XQ_1$$

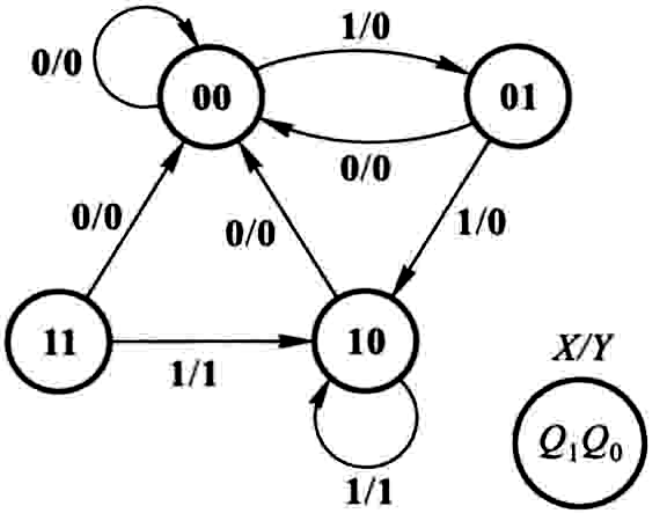




- 检查状态转换图是否自启

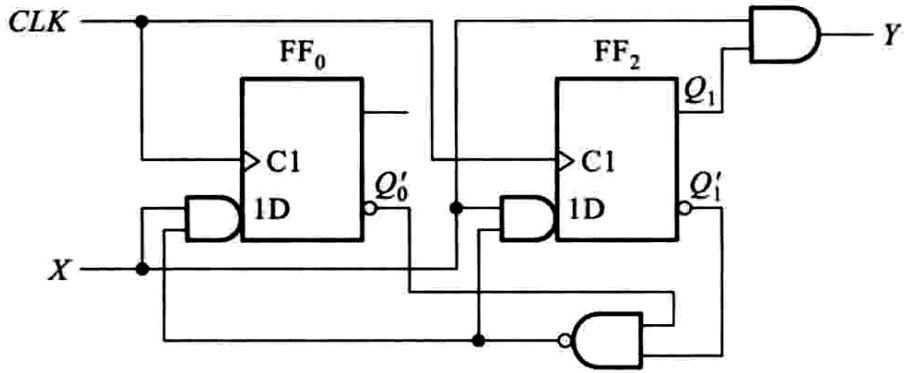


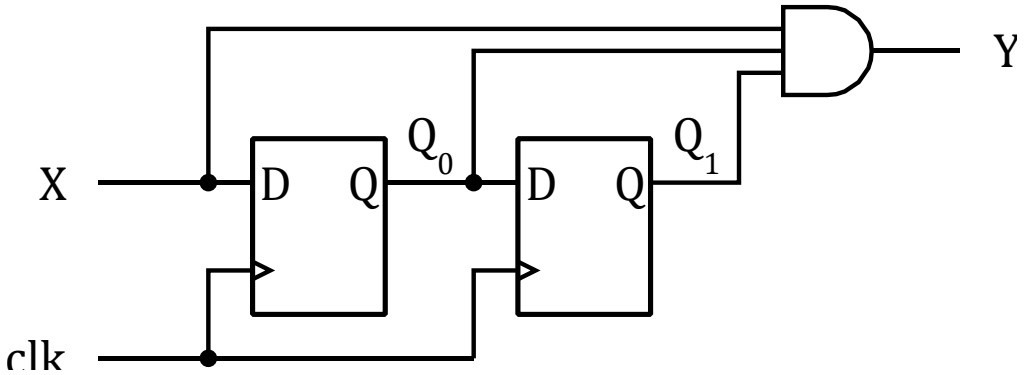




这是最优实现吗？

- 其实，二进制编码并非最优



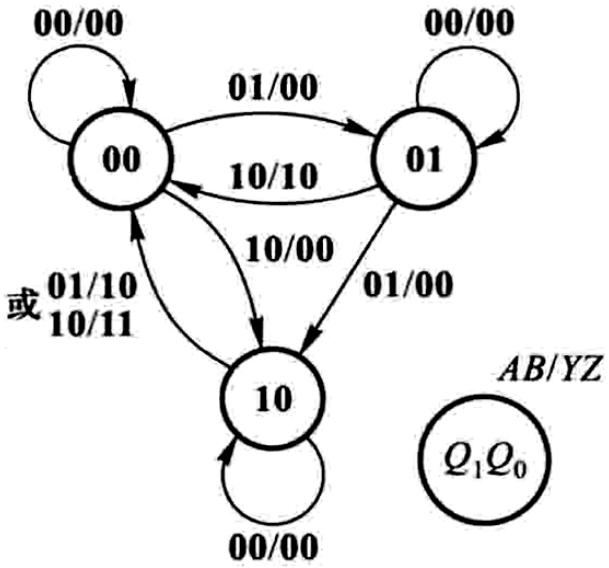


时序逻辑电路设计：贩卖机

- 设计一个饮料贩卖机，售价1.5元，每次接受一个硬币（1元/05毛）。如果投入1.5元，给一杯饮料。如果投入2元，给一杯饮料，吐出5毛。
  - A：投入1元硬币
  - B：投入5毛硬币
  - Y：给饮料
  - Z：退还5毛硬币

Q_1Q_0 \ AB	00	01	11	10
00	00/00	01/00	××/××	10/00
01	01/00	10/00	××/××	00/10
11	××/××	××/××	××/××	××/××
10	10/00	00/10	××/××	00/11

书错了



- 得出系统方程

Q_1Q_0 \ AB	00	01	11	10
00	00/00	01/00	××/××	10/00
01	01/00	10/00	××/××	00/10
11	××/××	××/××	××/××	××/××
10	10/00	00/10	××/××	00/11

$$\begin{cases} Q_1^* = Q_1A'B' + Q_1'Q_0'A + Q_0B \\ Q_0^* = Q_1'Q_0'B + Q_0A'B' \end{cases}$$

$$\begin{cases} D_1 = Q_1A'B' + Q_1'Q_0'A + Q_0B \\ D_0 = Q_1'Q_0'B + Q_0A'B' \end{cases}$$

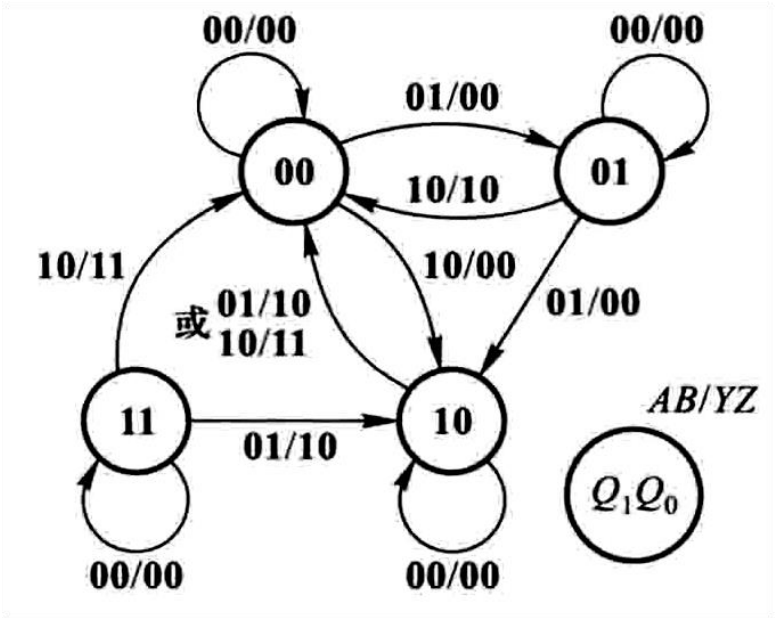
$$\begin{cases} Y = Q_1B + Q_1A + Q_0A \\ Z = Q_1A \end{cases}$$

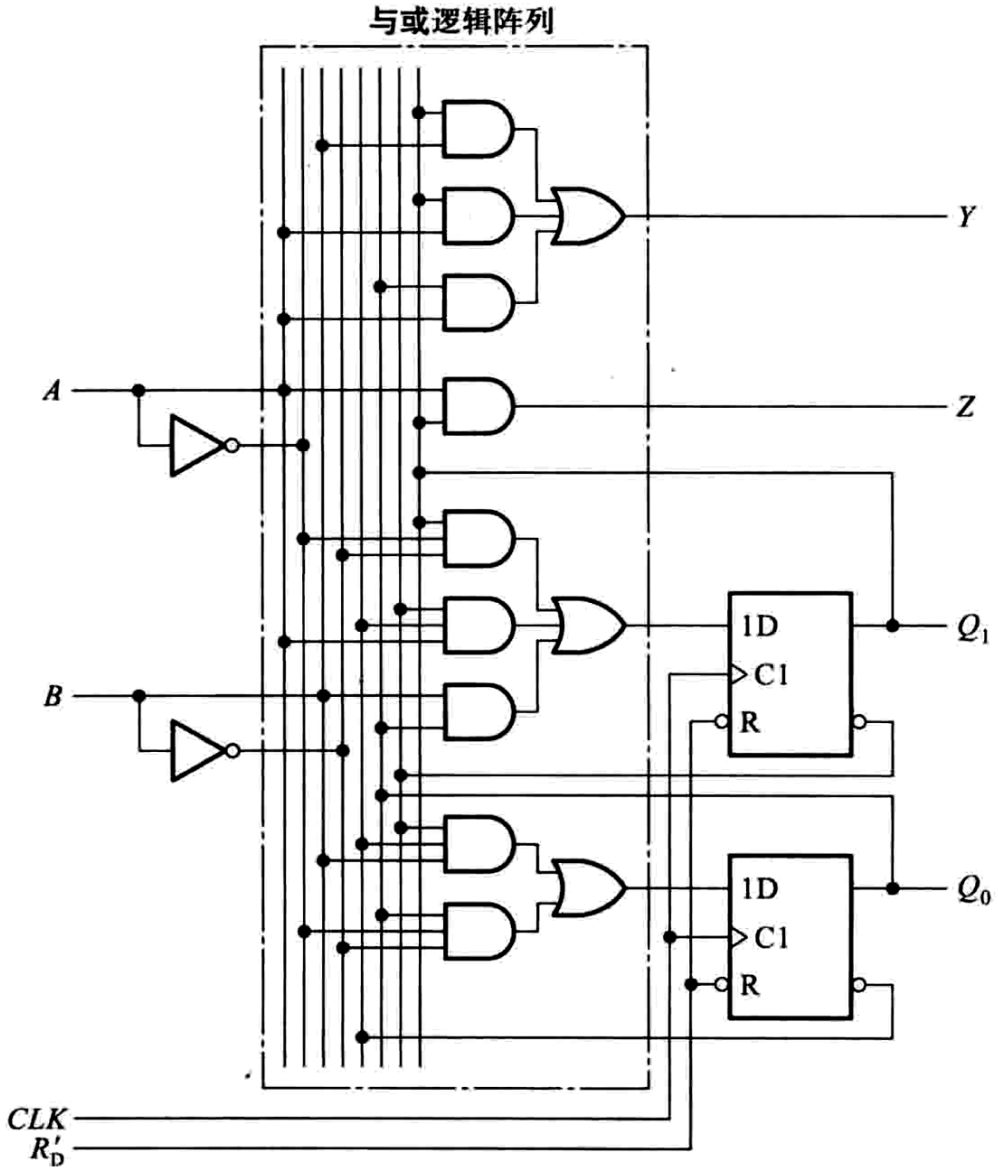
时序逻辑电路设计：贩卖机

$$\begin{cases} Q_1^* = Q_1A'B' + Q_1'Q_0'A + Q_0B \\ Q_0^* = Q_1'Q_0'B + Q_0A'B' \end{cases}$$

$$\begin{cases} D_1 = Q_1A'B' + Q_1'Q_0'A + Q_0B \\ D_0 = Q_1'Q_0'B + Q_0A'B' \end{cases}$$

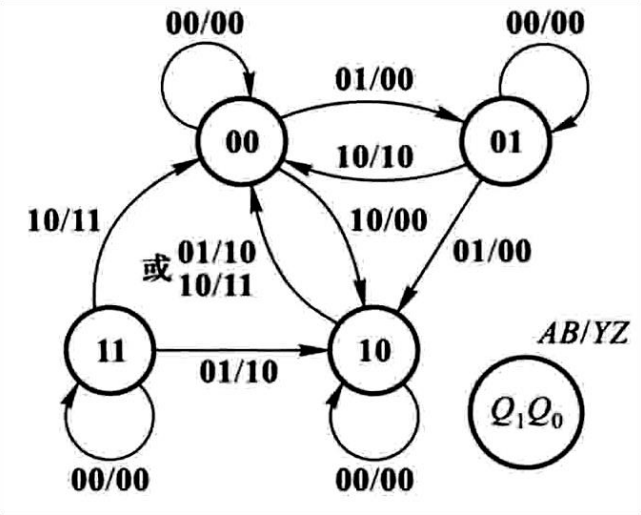
$$\begin{cases} Y = Q_1B + Q_1A + Q_0A \\ Z = Q_1A \end{cases}$$



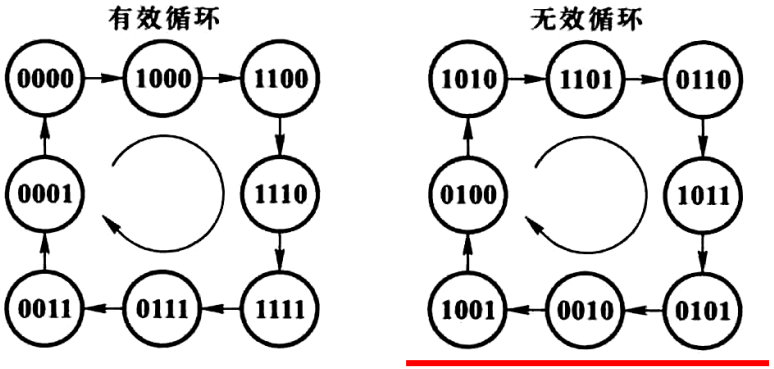


时序逻辑电路的自启设计

时序逻辑电路设计的最后一步是电路的自启检查，如果电路的状态转移图显示电路不可自启，则需要通过自启设计改进时序逻辑电路，使其可以自启。

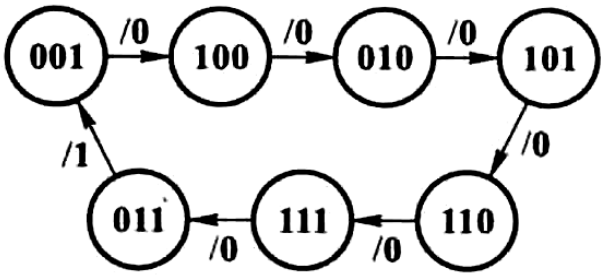


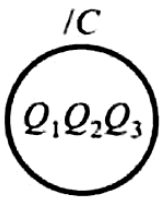
可自启



不可自启

自启设计示例：7计数器





Q_1 \ Q_2Q_3	00	01	11	10
0	×××	100	001	101
1	010	110	011	111

Q_1 \ Q_2Q_3	00	01	11	10
0	×	1	0	1
1	0	1	0	1

$$Q_1^* = Q_2 \oplus Q_3$$

Q_1 \ Q_2Q_3	00	01	11	10
0	×	0	0	0
1	1	1	1	1

$$Q_2^* = Q_1$$

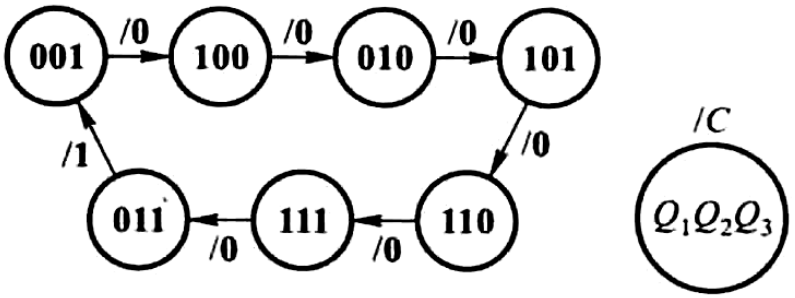
Q_1 \ Q_2Q_3	00	01	11	10
0	×	0	1	1
1	0	0	1	1

$$Q_3^* = Q_2$$

自启讨论卡诺图中的无关项。
如果无关项被覆盖，下一状态是1，如果没有被覆盖，则下一状态是0。

对于上面的卡诺图，状态000的下一状态仍然是000，不能转到有效状态。

自启设计示例：7计数器



Q_1 \ Q_2Q_3	00	01	11	10
0	×××	100	001	101
1	010	110	011	111

Q_1 \ Q_2Q_3	00	01	11	10
0	×	0	0	0
1	1	1	1	1

$$Q_2^* = Q_1$$

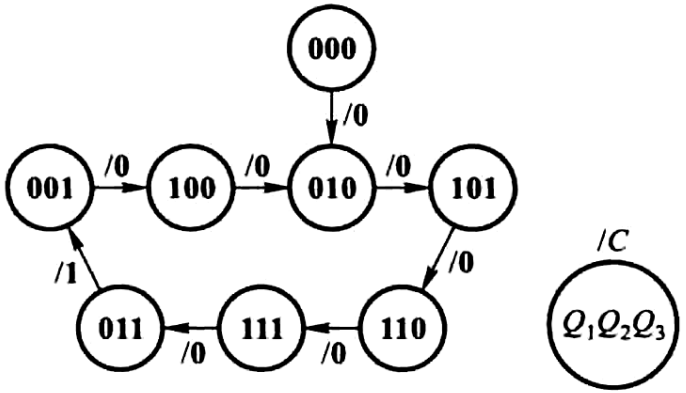
Q_1 \ Q_2Q_3	00	01	11	10
0	×	0	0	0
1	1	1	1	1

$$Q_2^* = Q_1 + \overline{Q_2} \cdot \overline{Q_3}$$

修改Q_2^*，覆盖一个无关项，迫使无关项000的下一状态为010。

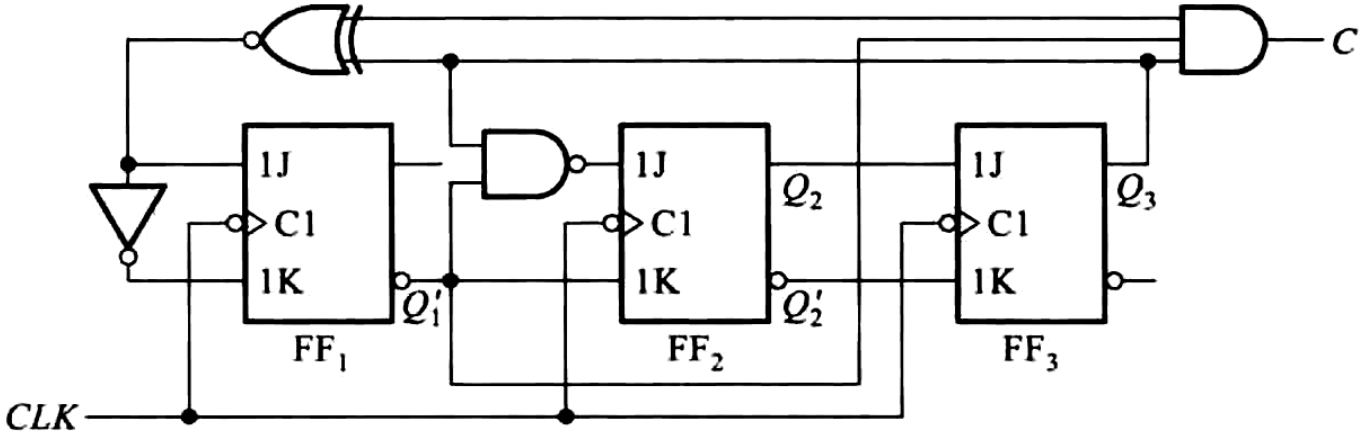
自启设计示例：7计数器

$$Q_1^* = Q_2 \oplus Q_3$$
$$Q_2^* = Q_1 + \overline{Q_2} \cdot \overline{Q_3}$$
$$Q_3^* = Q_2$$

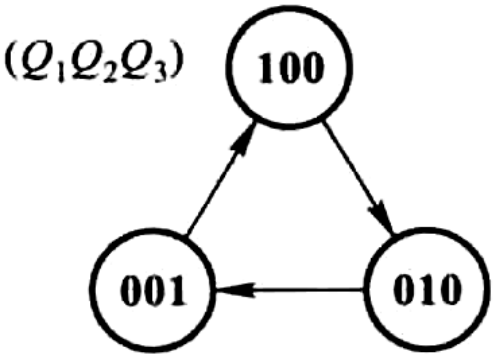


$$\begin{cases} J_1 = Q_2 \oplus Q_3, & K_1 = (Q_2 \oplus Q_3)' \\ J_2 = (Q_1' Q_3)', & K_2 = Q_1' \\ J_3 = Q_2, & K_3 = Q_2' \end{cases}$$

$$C = Q_1' Q_2 Q_3$$

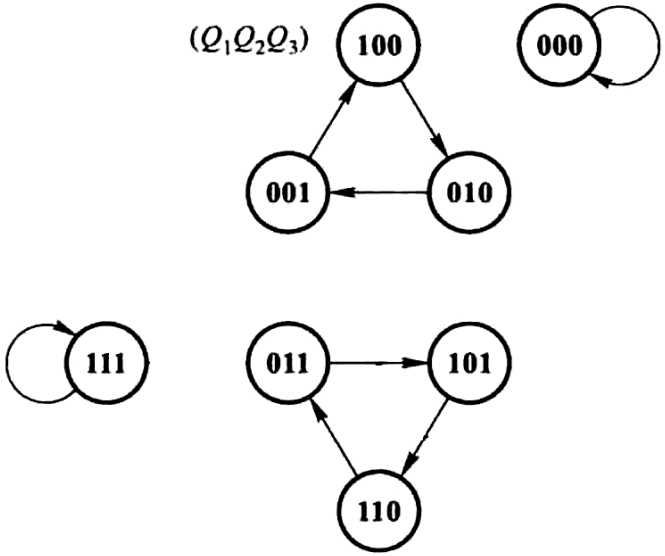


自启设计示例：3位循环计数器



Q_1 \ Q_2Q_3	00	01	11	10
0	×××	100	×××	001
1	010	×××	×××	×××

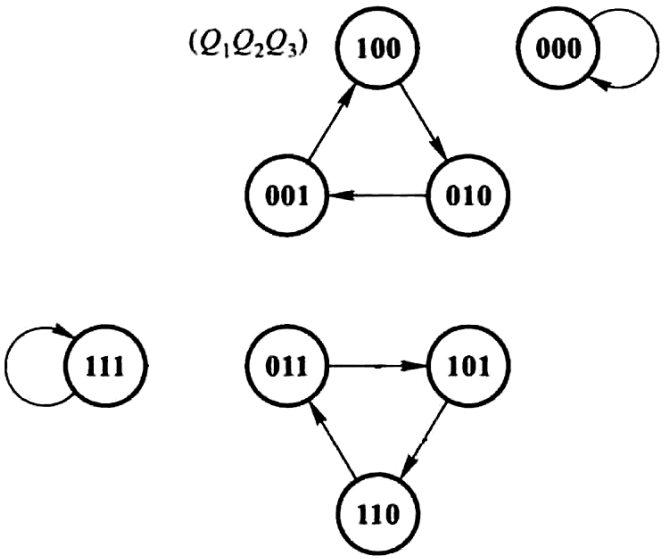
$$\begin{cases} Q_1^* = Q_3 \\ Q_2^* = Q_1 \\ Q_3^* = Q_2 \end{cases}$$



自启设计示例：3位循环计数器

Q_1 \ Q_2Q_3	00	01	11	10
0	×××	100	×××	001
1	010	×××	×××	×××

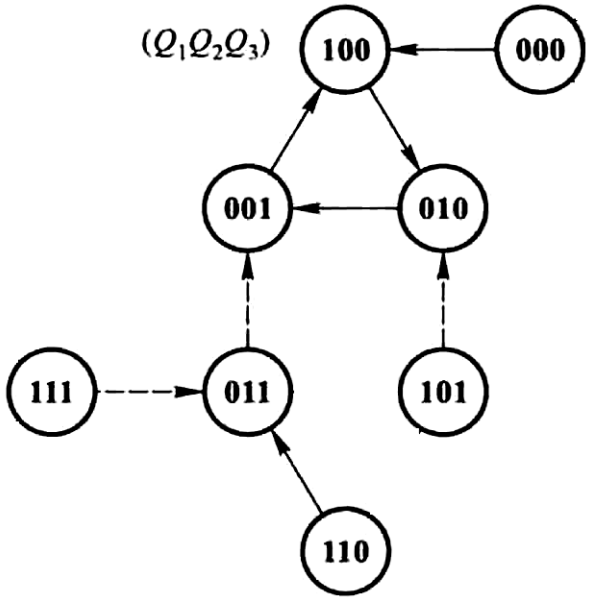
Q_1 \ Q_2Q_3	00	01	11	10
0	000	100	101	001
1	010	110	111	011



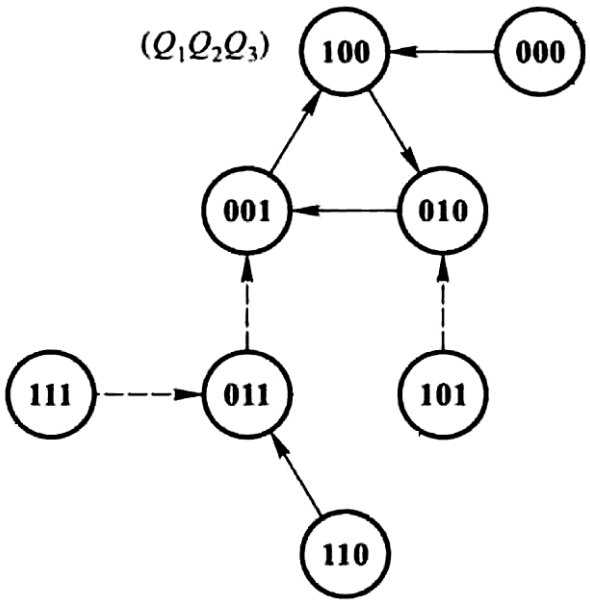
Q_1 \ Q_2Q_3	00	01	11	10
0	1 0 0	1 0 0	0 0 1	0 0 1
1	0 1 0	0 1 0	0 1 1	0 1 1

只变了Q_1的逻辑

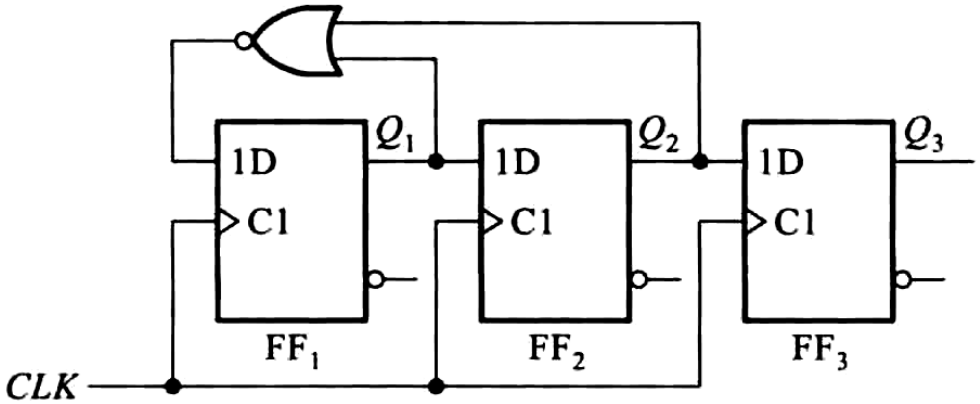
$$Q_1^* = \overline{Q_1} \cdot \overline{Q_2}$$



自启设计示例：3位循环计数器

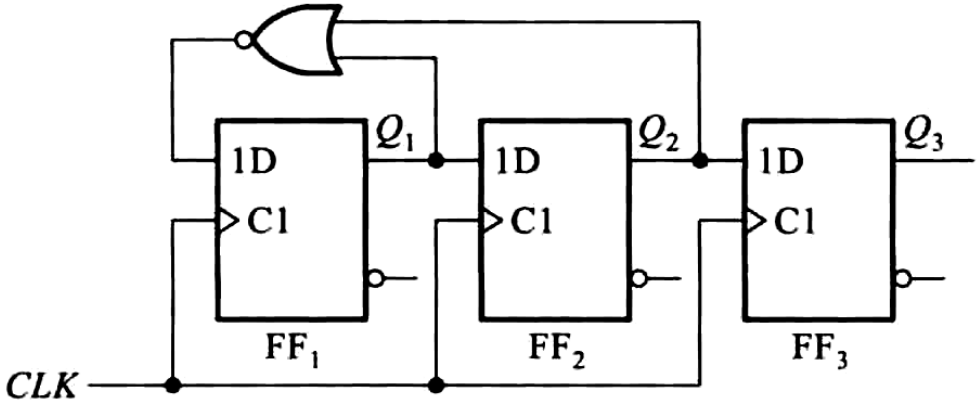


Q_1 \ Q_2Q_3	00	01	11	10
0	1 0 0	1 0 0	0 0 1	0 0 1
1	0 1 0	0 1 0	0 1 1	0 1 1



自启的实际意义（附加）

- 自启：当电路处于一个非法状态时，可以在有限时间内，返回到有效状态。
- 自我启动？实际电路是可以的，仿真当中却不一定。
  - 在仿真中，所有的触发器的默认初始值为X，未知状态
  - 下面的计数器在仿真中不会跳出默认初始状态“xxx”！



X状态的传递规则（附加）

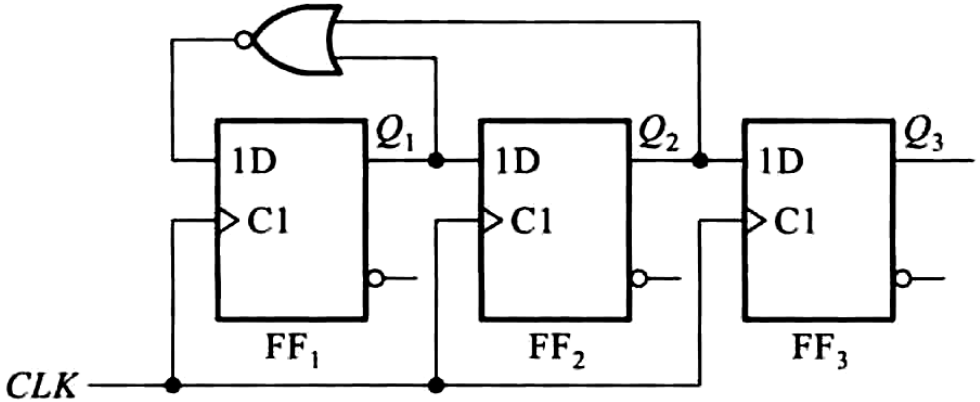
- X代表未知状态，在仿真中是信号的初始状态

$$\overline{x} = x$$

$$x \cdot 1 = x \qquad x \cdot 0 = 0 \qquad x \cdot x = x$$

$$x + 1 = 1 \qquad x + 0 = x \qquad x + x = x$$

$$x \oplus 1 = x \qquad x \oplus 0 = x \qquad x \oplus x = x$$



- 对于上面的电路，正确仿真：
  - 添加异步复位/置位
  - 在仿真开始时强制赋值（使用force语句）

X状态的传递规则（附加）

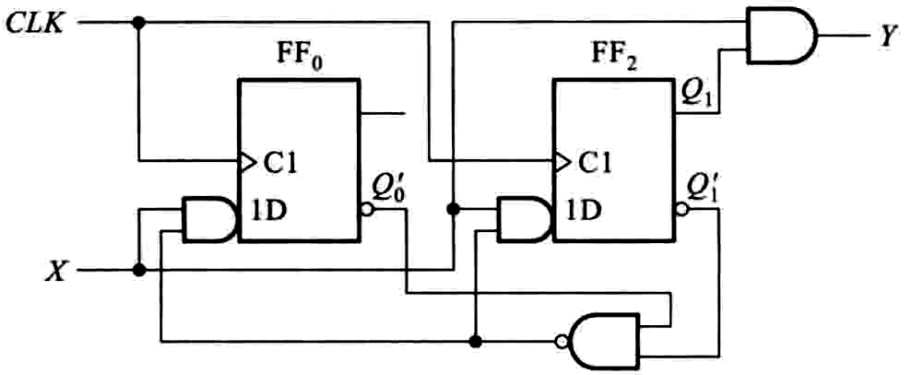
- X代表未知状态，在仿真中是信号的初始状态

$$\overline{x} = x$$

$$x \cdot 1 = x \qquad x \cdot 0 = 0 \qquad x \cdot x = x$$

$$x + 1 = 1 \qquad x + 0 = x \qquad x + x = x$$

$$x \oplus 1 = x \qquad x \oplus 0 = x \qquad x \oplus x = x$$



- 上面的电路，当x=0持续两个周期，可以脱离x状态

时序逻辑电路的设计方法总结

- 分析问题，得到问题的状态转换图/表
- 状态化简
  - 如果两个状态在相同的输入下有相同的输出，则可以被合并为同一个状态
- 状态分配
  - 对所有状态进行编码
  - 二进制编码
    - 使用n位编码编码m个不同状态：$2^{n-1} < M \leq 2^n$
  - 独热编码
    - 使用m位编码编码m个不同状态
- 选定触发器类型，确定系统方程
- 电路实现
- 检查电路是否可以自启以及自启的设计
  - 没有被覆盖的为0，覆盖了为1。
  - 通过修改无关项的取值，改变状态转换图。

总结：考试范围

- 时序逻辑电路的系统方程
  - 触发器驱动方程、触发器特性方程、输出方程
  - 状态转换表、状态转换图、自启
- 常用时序逻辑
  - 移位寄存器、计数器、移位计数器、伪随机数发生器、顺序脉冲发生器、序列信号发生器
  - 任意数制计数器
- 异步时序逻辑（附加）
- 同步时序逻辑的设计
  - 分析问题、状态设计、确定系统方程、设计电路
  - 自启设计、仿真自启（X状态传递）

任何问题？

作业

第六章习题：

7，8，15，16，17，22，26，29，32，33。

课堂习题

- 分析电路的状态转换图。检查电路是否自启，如果不能，如何修正?