

《数字逻辑》 Digital Logic

时序逻辑电路

北京工业大学软件学院
王晓懿



§ 概述

§ 时序逻辑电路的分析方法

§ 若干常用的时序逻辑电路

§ 时序逻辑电路的设计方法

一. 重点掌握的内容:

- (1) 时序逻辑电路的概念及电路结构特点;
- (2) 同步时序电路的一般分析方法;
- (3) 同步计数器的一般分析方法;
- (4) 会用置零法和置数法构成任意进制计数器。

二. 一般掌握的内容:

- (1) 同步、异步的概念, 电路现态、次态、有效状态、无效状态、有效循环、无效循环、自启动的概念, 寄存的概念;
- (2) 同步时序逻辑电路设计方法。

概述

一、组合电路与时序电路的区别

1. 组合电路： 电路的输出只与电路的输入有关，
与电路的**前一时刻**的状态无关。

2. 时序电路：

电路在某一给定时刻的输出 { 取决于该时刻电路的输入
还取决于**前一时刻**电路的状态

由触发器保存

时序电路： 组合电路 + 触发器

电路的状态与**时间**顺序有关



时序电路在任何时刻的稳定输出，不仅与该时刻的输入信号有关，而且还与电路原来的状态有关。

构成时序逻辑电路的基本单元是触发器。

二、时序逻辑电路的分类：

按动作特点可分为

同步时序逻辑电路

所有触发器状态的变化都是在同一时钟信号操作下同时发生。

异步时序逻辑电路

触发器状态的变化不是同时发生。

三、时序逻辑电路的功能描述方法

逻辑方程组

状态表

卡诺图

状态图

时序图

逻辑图

1. 逻辑方程组

特性方程：描述触发器逻辑功能的逻辑表达式。

驱动方程：（激励方程）触发器输入信号的逻辑表达式。

时钟方程：控制时钟CLK的逻辑表达式。

状态方程：（次态方程）次态输出的逻辑表达式。

驱动方程代入特性方程得状态方程。

输出方程：输出变量的逻辑表达式。

2. 状态表

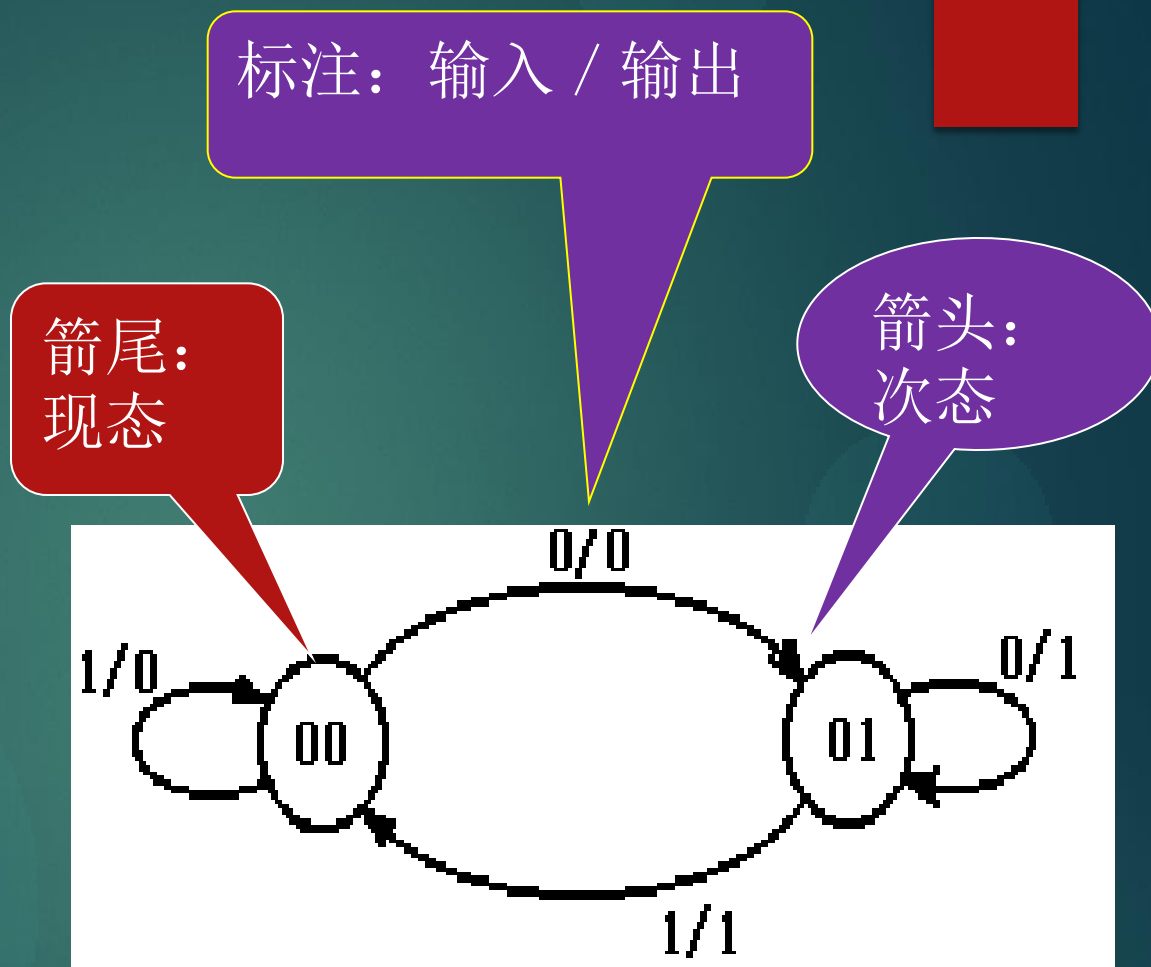
反映输出 Z 、次态 Q^* 与输入 X 、现态 Q 之间关系的表格。

次态/输出 现态		输入 X	
Q			Q^* / Z

X	现态	次态 Q^*	输出

3. 状态图

反映时序电路
状态转换规律，
及相应输入、
输出取值关系
的图形。



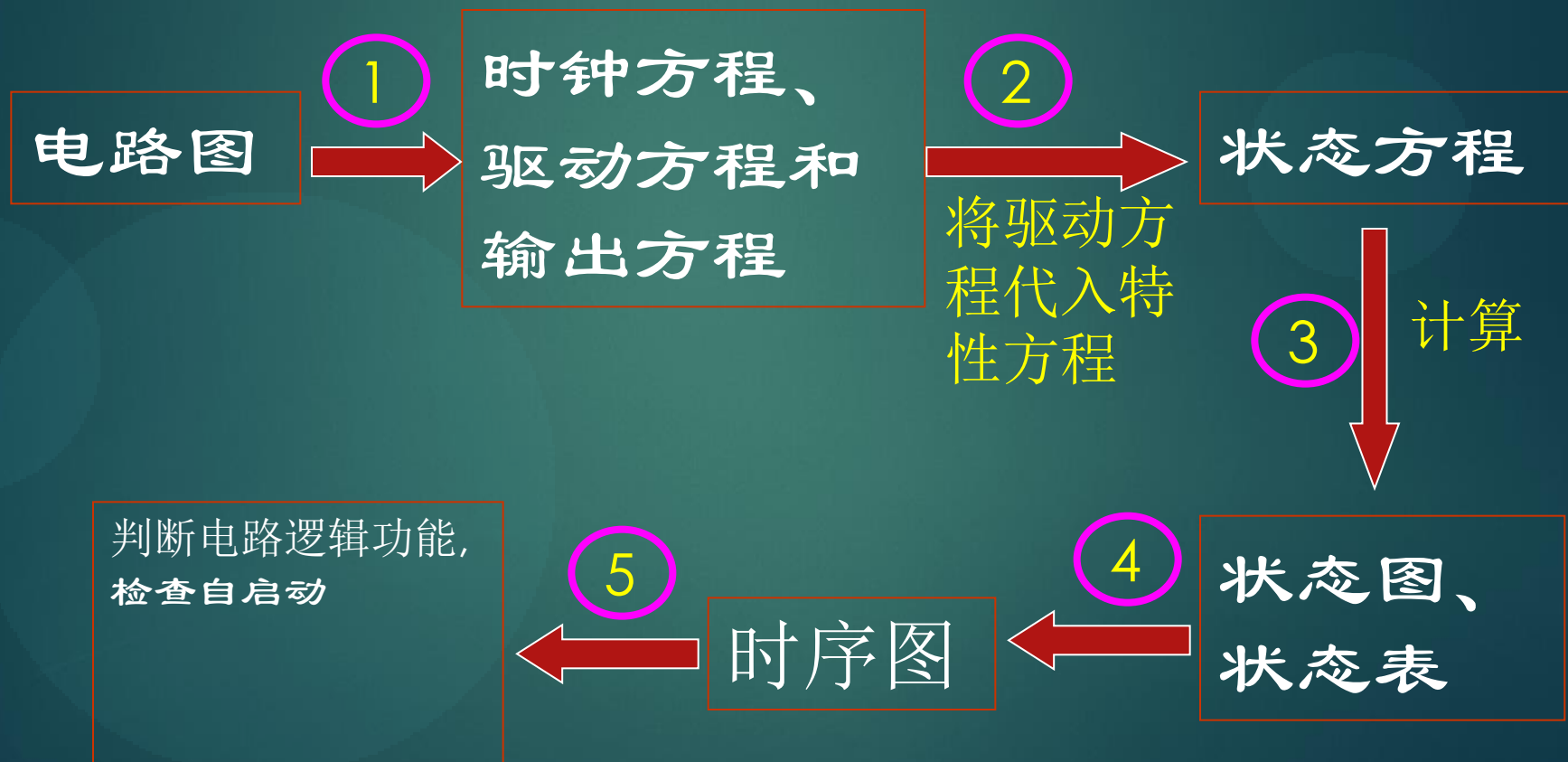
4. 时序图

时序图又叫**工作波形图**，它用波形的形式形象地表达了输入信号、输出信号、电路的状态等的取值在时间上的对应关系。

这四种方法从不同侧面突出了时序电路逻辑功能的特点，它们在本质上是相同的，可以互相转换。

时序逻辑电路的分析方法

时序电路的分析步骤：



几个概念

有效状态：在时序电路中，凡是被利用了的状态。

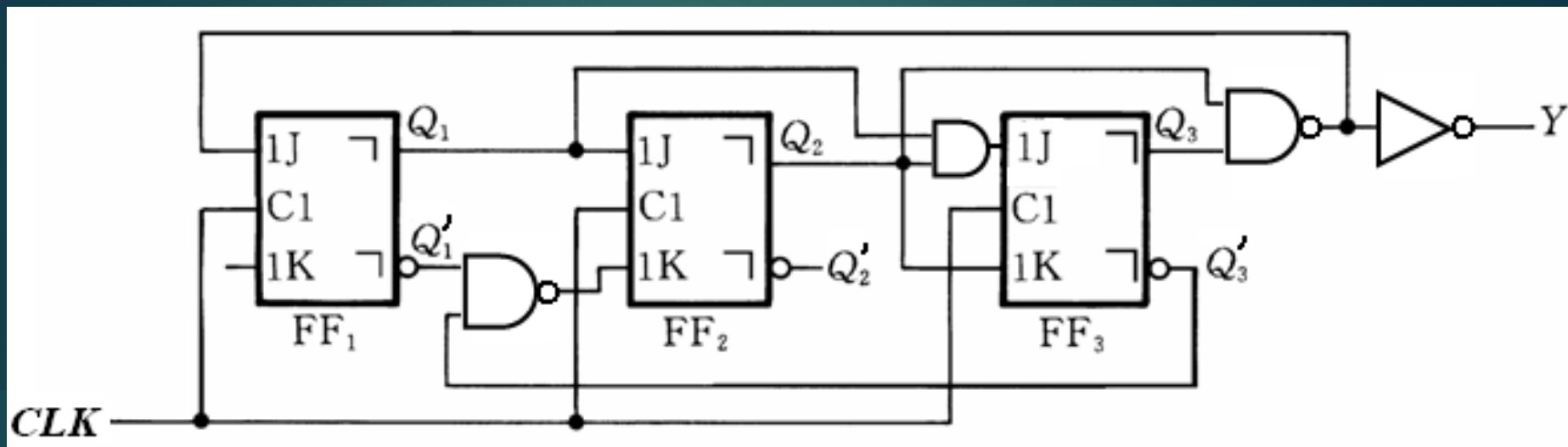
有效循环：有效状态构成的循环。

无效状态：在时序电路中，凡是没有被利用的状态。

无效循环：无效状态若形成循环，则称为无效循环。

自启动：在CLK作用下，无效状态能自动地进入到有效循环中，则称电路能自启动，否则称不能自启动。

例



解： ①写方程组

$$\begin{array}{l} \text{驱动方程} \end{array} \left\{ \begin{array}{l} J_1 = (Q_2 \cdot Q_3)' \\ J_2 = Q_1 \\ J_3 = Q_1 \cdot Q_2 \end{array} \right. \quad \begin{array}{l} K_1 = 1 \\ K_2 = (Q'_1 \cdot Q'_3)' \\ K_3 = Q_2 \end{array}$$

同步时序电路，时钟方程省去。

输出方程

$$Y = Q_2 \cdot Q_3$$

②求状态方程

将驱动方程代入JK触发器的特性方程
中得电路的状态方程：

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

$$\begin{cases} Q_1^* = J_1 Q_1' + K_1' Q_1 = (Q_2 \cdot Q_3)' \cdot Q_1' \\ Q_2^* = J_2 Q_2' + K_2' Q_2 = Q_1 \cdot Q_2' + Q_1' \cdot Q_3' \cdot Q_2 \\ Q_3^* = J_3 Q_3' + K_3' Q_3 = Q_1 \cdot Q_2 \cdot Q_3' + Q_2' \cdot Q_3 \end{cases}$$

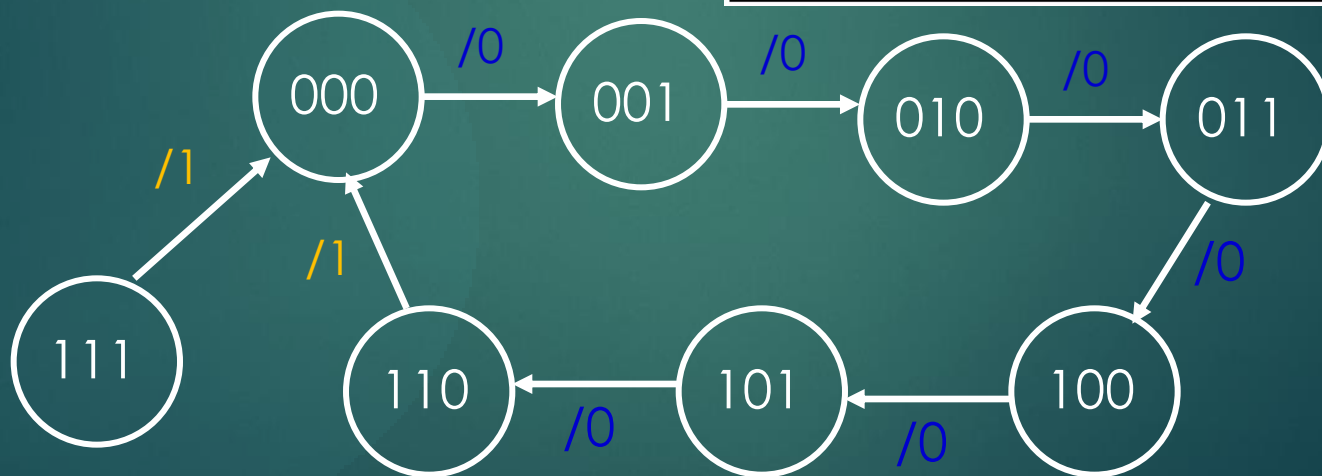
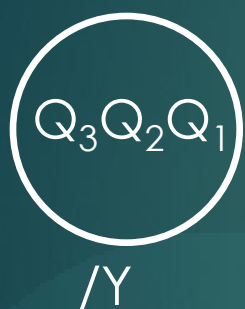
③计算、列状态转换表

$$\begin{cases} Q_1^* = (Q_2 \cdot Q_3)' \cdot Q_1' \\ Q_2^* = Q_1 \cdot Q_2' + Q_1' \cdot Q_3' \cdot Q_2 \\ Q_3^* = Q_1 \cdot Q_2 \cdot Q_3' + Q_2' \cdot Q_3 \end{cases}$$

现 态			次			
Q_3	Q_2	Q_1	Q_3^*	Q_2^*	Q_1^*	Y
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	1	0	1	0
					0	0
					0	1
					0	1

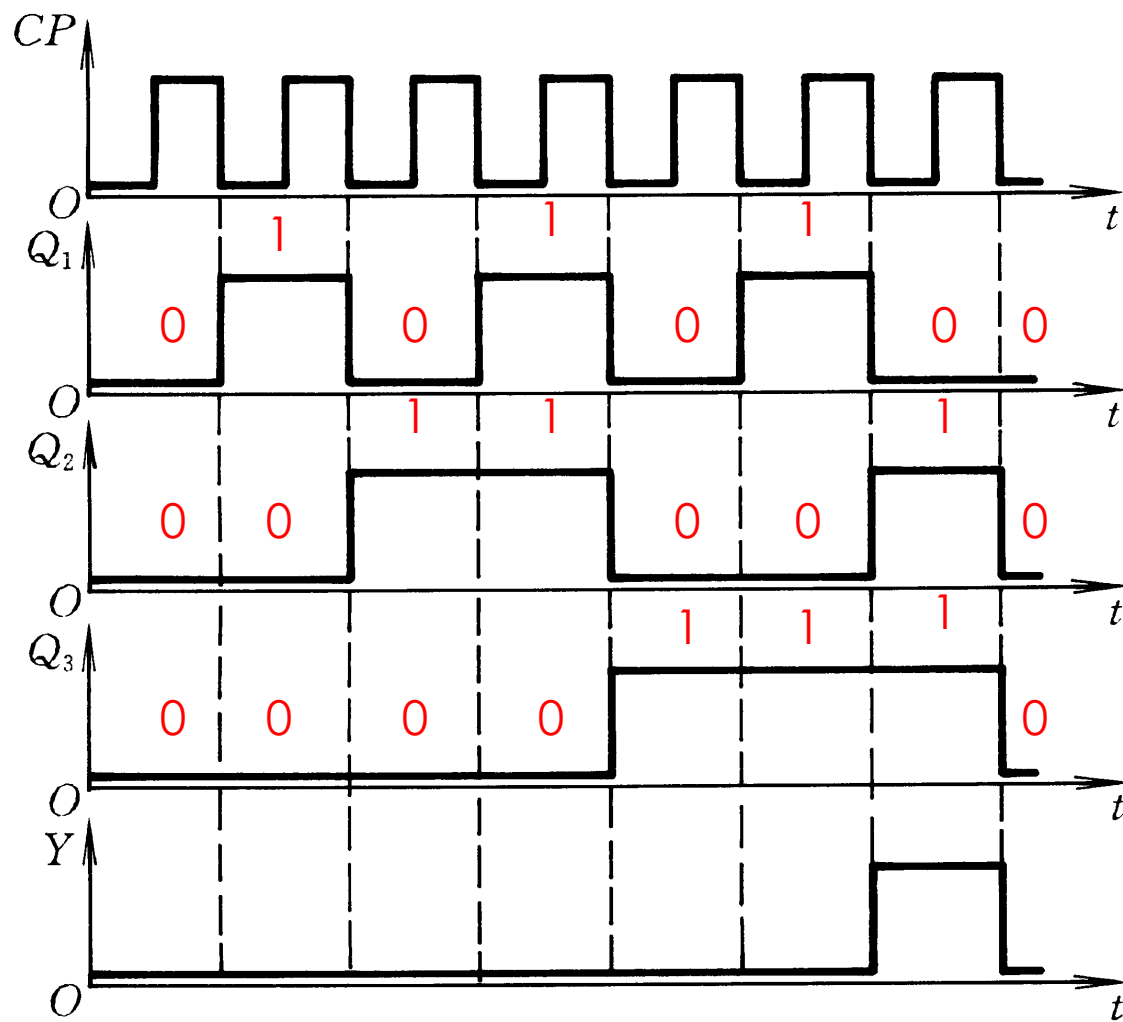
$$\begin{cases} Q_1^* = (Q_2 \cdot Q_3)' \cdot Q_1' \\ Q_2^* = Q_1 \cdot Q_2' + Q_1' \cdot Q_3' \cdot Q_2 \\ Q_3^* = Q_1 \cdot Q_2 \cdot Q_3' + Q_2' \cdot Q_3 \end{cases}$$

画状态转换图



现 态			次 态			输 出
Q_3	Q_2	Q_1	Q_3^*	Q_2^*	Q_1^*	Y
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	1	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	1	0	0	0	1

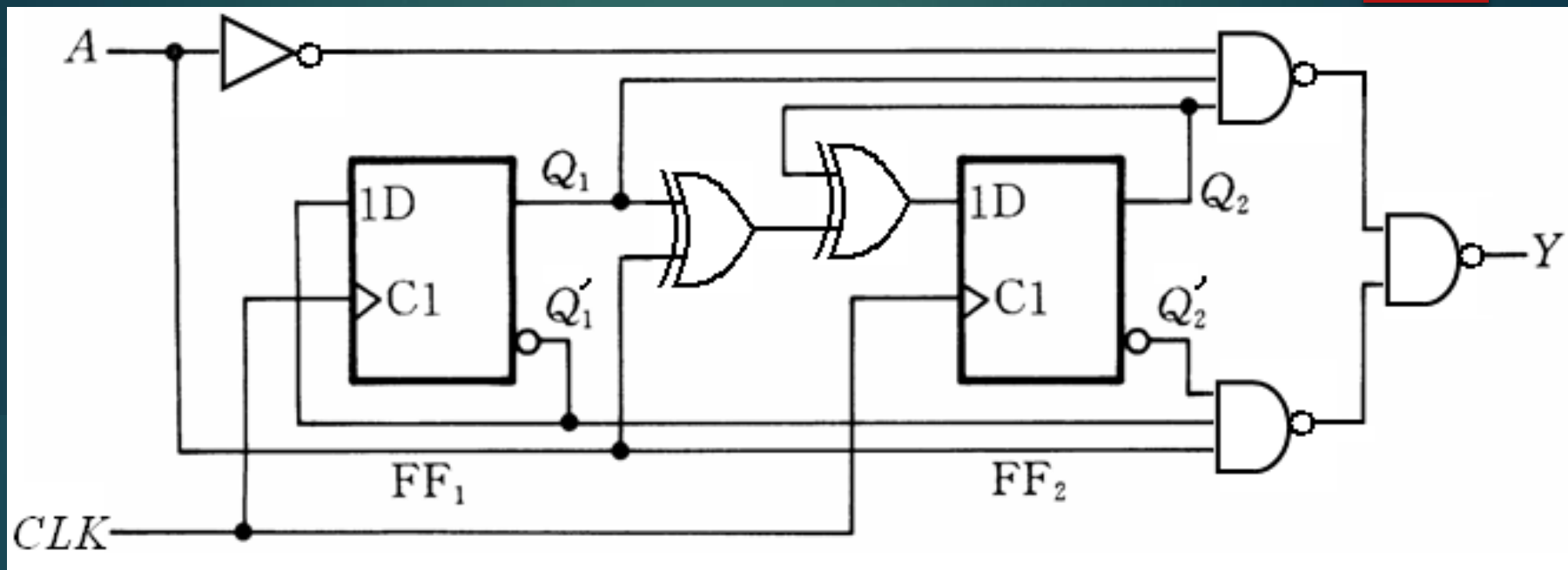
④作时序图



⑤说明电路功能

这是一个同步七进制加法计数器，能自启动。

例

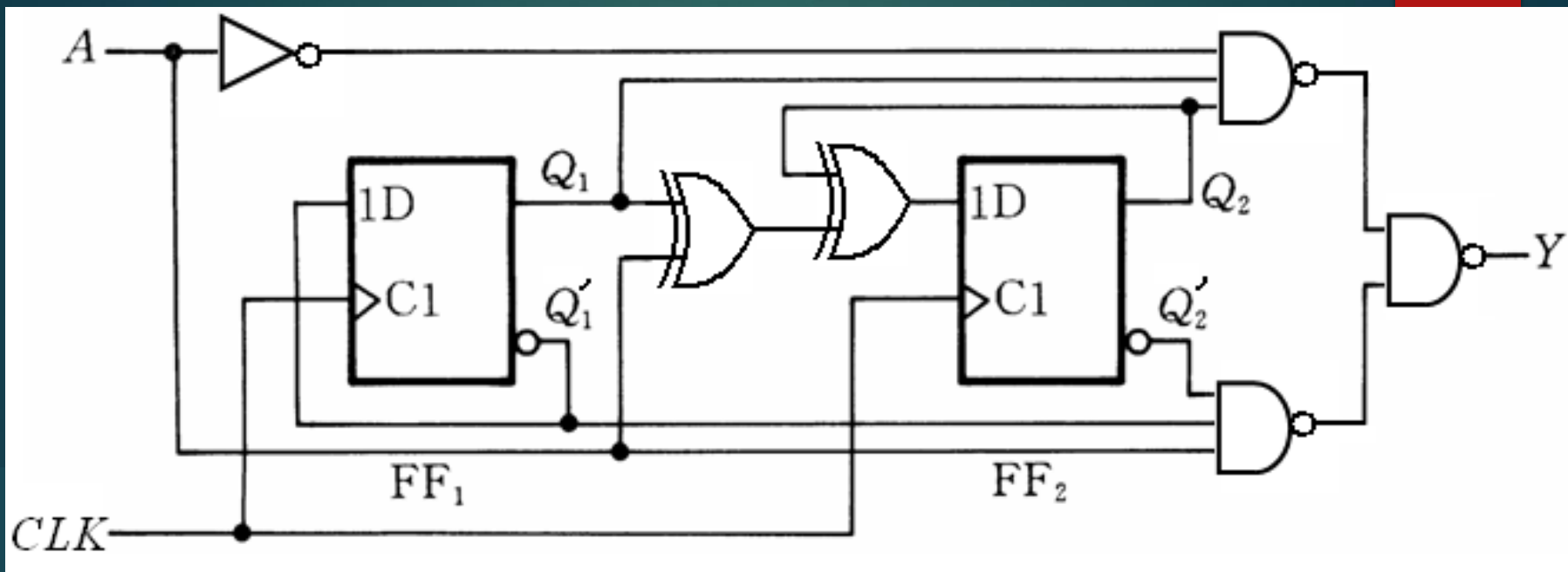


解: ①写方程式

驱动方程

$$\begin{cases} D_1 = Q'_1 \\ D_2 = A \oplus Q_1 \oplus Q_2 \end{cases}$$

②求状态方程



输出方程

$$Y = ((A'Q_1Q_2)' \cdot (AQ_1'Q_2'))' = A'Q_1Q_2 + AQ_1'Q_2'$$

③计算、
转换表

$$Y = A'Q_1Q_2 + AQ_1'Q_2'$$

现 态			次 态		输出
A	Q ₂	Q ₁	Q ₂ [*]	Q ₁ [*]	Y
0	0	0	0	1	0
0	0	1	1	0	0
			1	1	0
			0	0	1
			1	1	1
1	0	1	0	0	0
1	1	0	0	1	0
1	1	1	1	0	0

$$\begin{cases} Q_1^* = Q_1' \\ Q_2^* = A \oplus Q_1 \oplus Q_2 \end{cases}$$

$$\begin{cases} Q_1^* = D_1 = Q_1' \\ Q_2^* = D_2 = A \oplus Q_1 \oplus Q_2 \end{cases}$$

$$Y = A'Q_1Q_2 + AQ_1'Q_2'$$

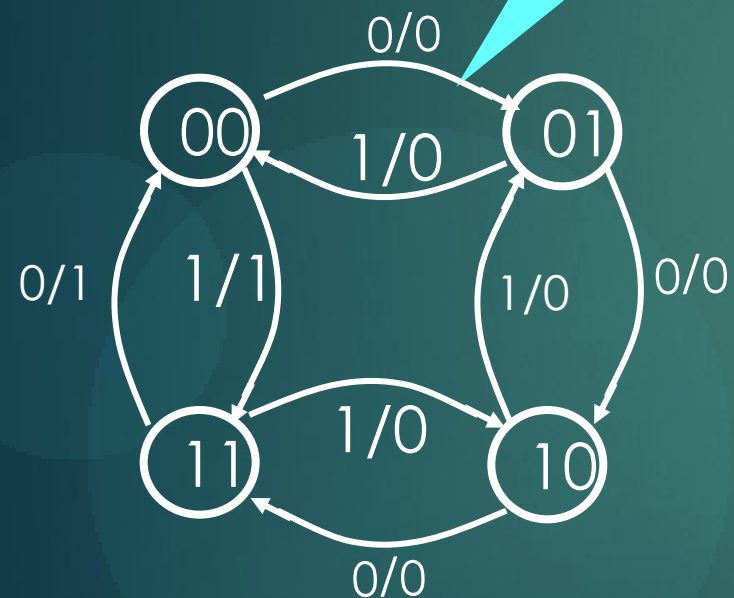
转换条件

A/Y

Q_2Q_1

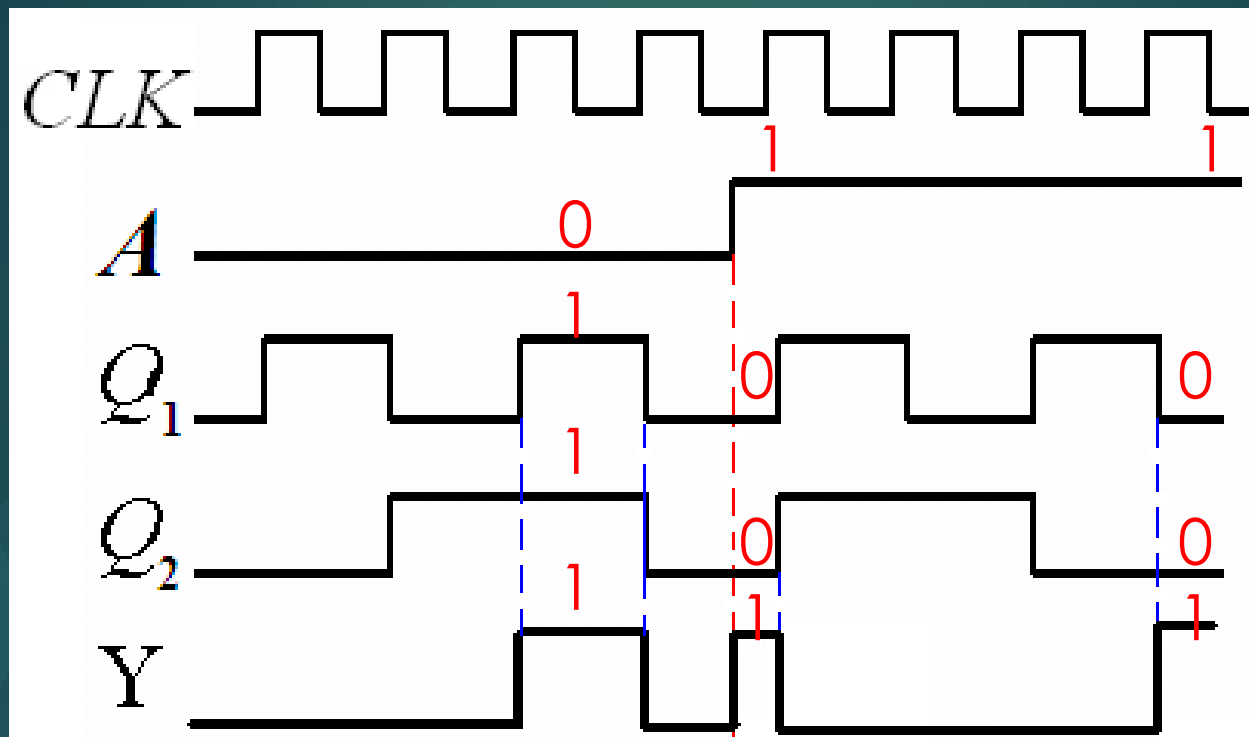
电路状态

转换方向



输入	现 态		次 态		输出
A	Q_2	Q_1	Q_2^*	Q_1^*	Y
0	0	0	0	1	0
0	0	1	1	0	0
0	1	0	1	1	0
0	1	1	0	0	1
1	0	0	1	1	1
1	1	1	1	0	0
1	1	0	0	1	0
1	0	1	0	0	0

④作时序图



⑤说明电路功能

$A=0$ 时是二位二进制加法计数器；
 $A=1$ 时是二位二进制减法计数器。

若干常用的时序逻辑电路

寄存器和移位寄存器

一、寄存器

在数字电路中，用来存放二进制数据或代码的电路称为寄存器。

寄存器是由具有存储功能的触发器组合起来构成的。一个触发器可以存储1位二进制代码，存放 n 位二进制代码的寄存器，需用 n 个触发器来构成。

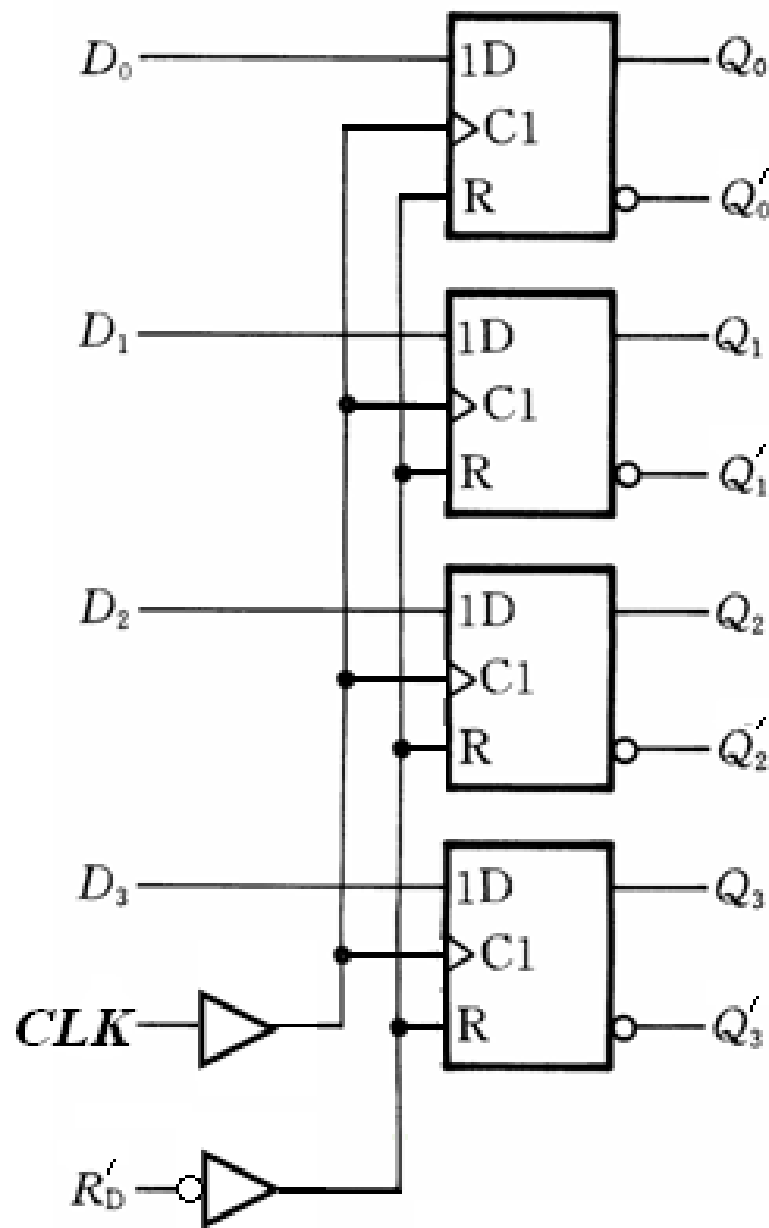
(1) 清零。 $R'_D = 0$ ，异步清零。即有：

$$Q_3 Q_2 Q_1 Q_0 = 0000$$

(2) 送数。 $R'_D = 1$ 时， CLK 上升沿送数。即有：

$$Q_3^* Q_2^* Q_1^* Q_0^* = D_3 D_2 D_1 D_0$$

(3) 保持。在 $R'_D = 1$ 、 CLK 上升沿以外时间，寄存器内容将保持不变。

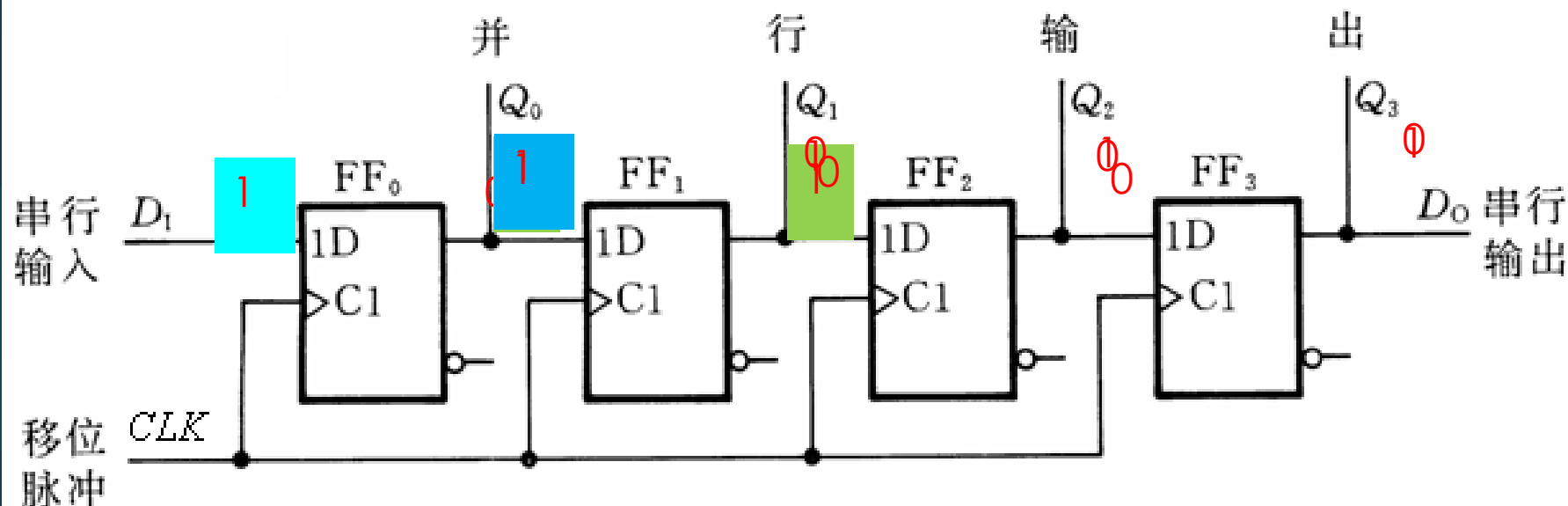


边沿触发器构成

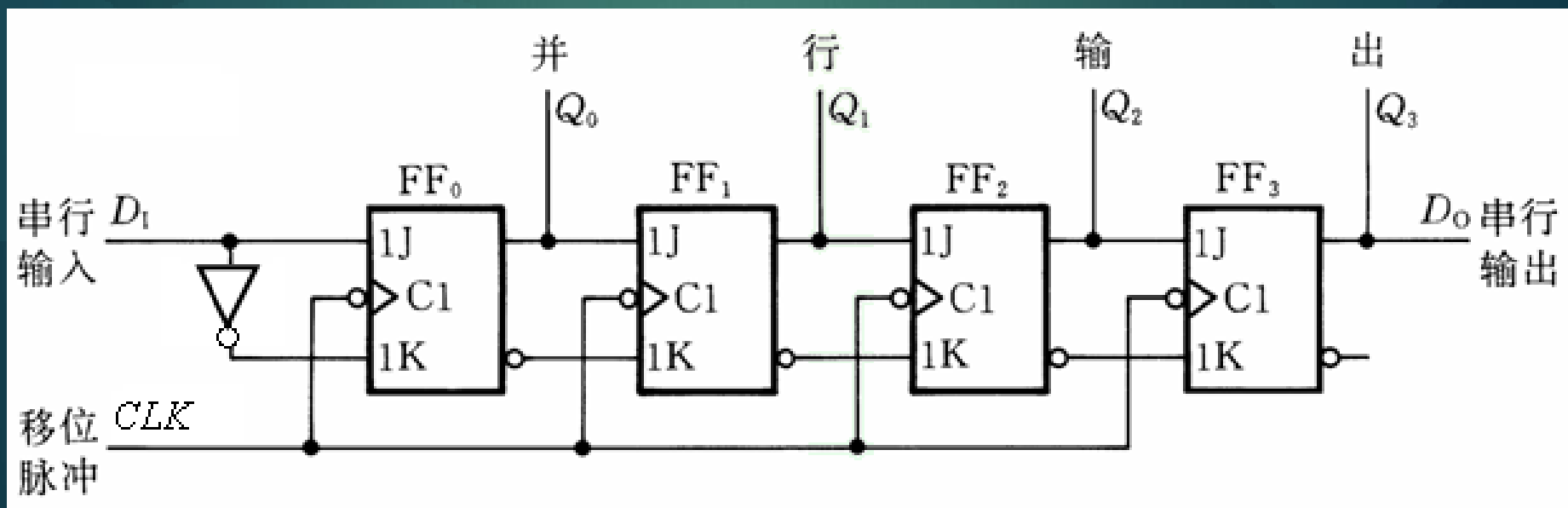
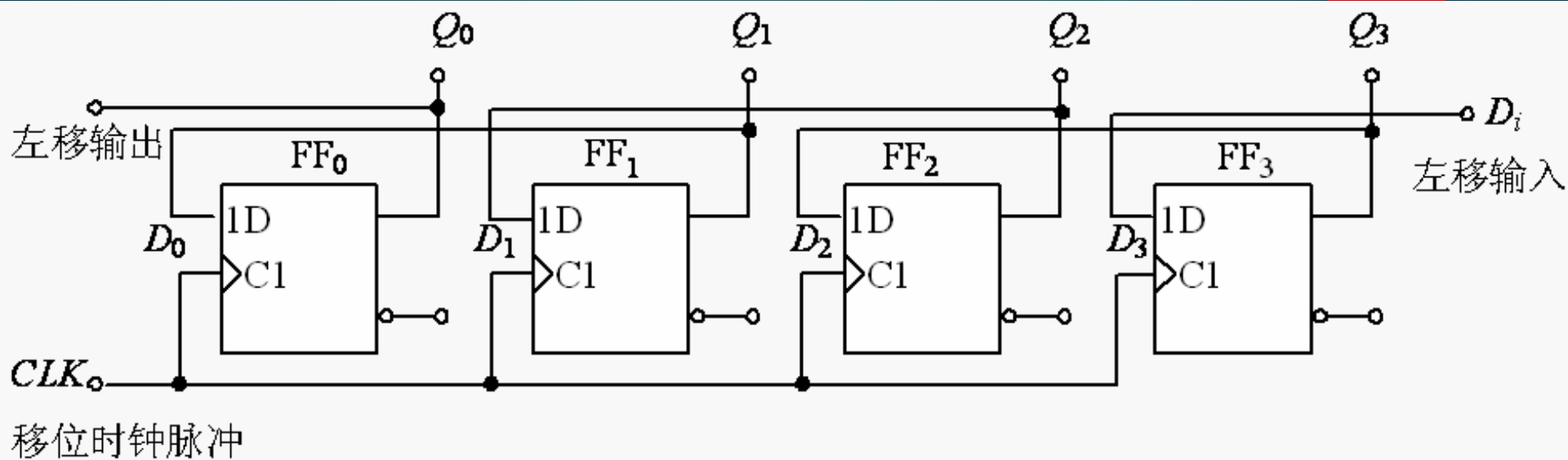
二、移位寄存器

单向移位寄存器

$$Q_0^* = D_i, \quad Q_1^* = Q_0, \quad Q_2^* = Q_1, \quad Q_3^* = Q_2$$



首先将4位数据并行置入移位寄存器的4个触发器中，经过4个CP, 4位代码将从串行输出端依次输出，实现数据的并行—串行转换。



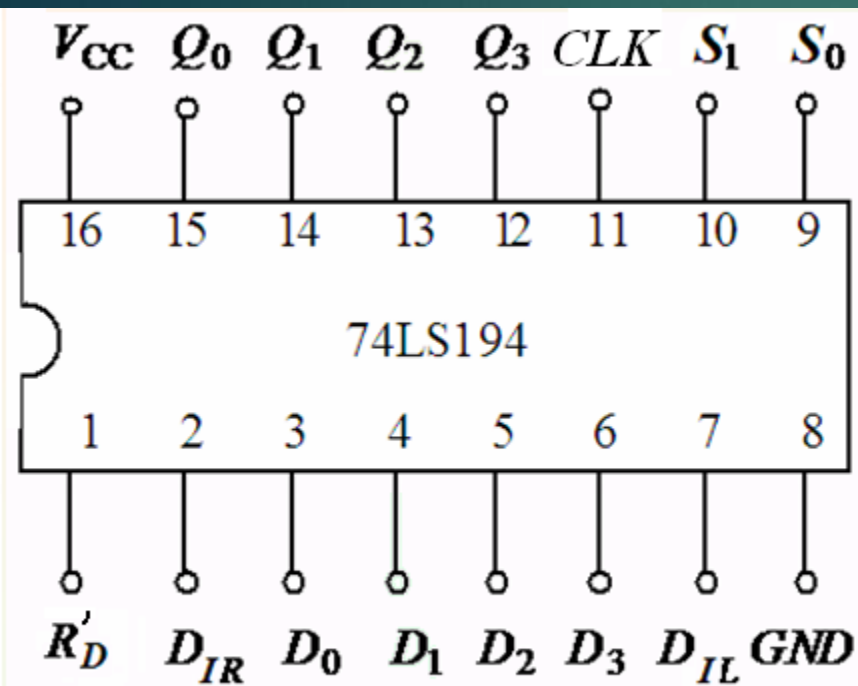
单向移位寄存器具有以下主要特点：

(1) 单向移位寄存器中的数码，在CLK脉冲操作下，可以依次右移或左移。

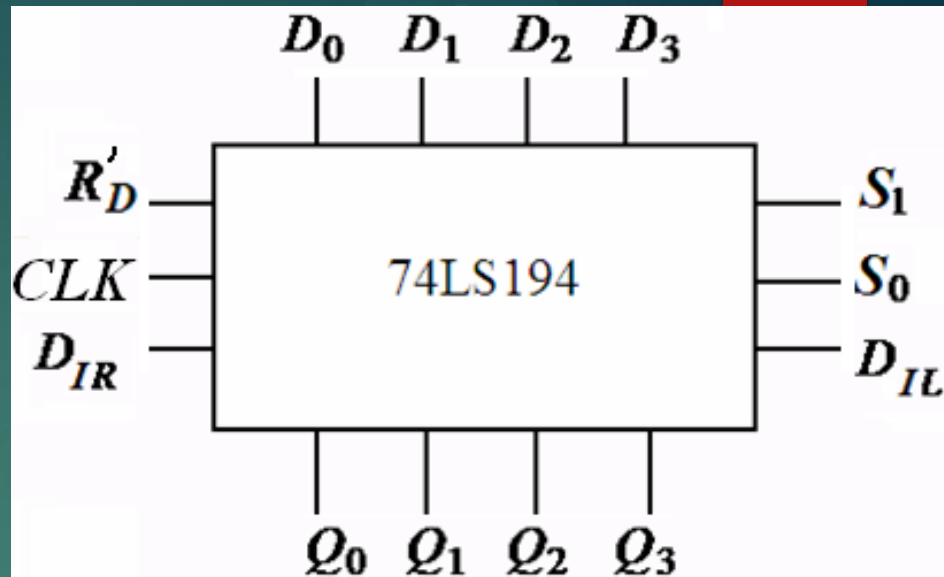
(2) n 位单向移位寄存器可以寄存 n 位二进制代码。 n 个CLK脉冲即可完成串行输入工作，此后可从 $Q_0 \sim Q_{n-1}$ 端获得并行的 n 位二进制数码，再用 n 个CLK脉冲又可实现串行输出操作。

(3) 若串行输入端状态为0，则 n 个CLK脉冲后，寄存器便被清零。

双向移位寄存器



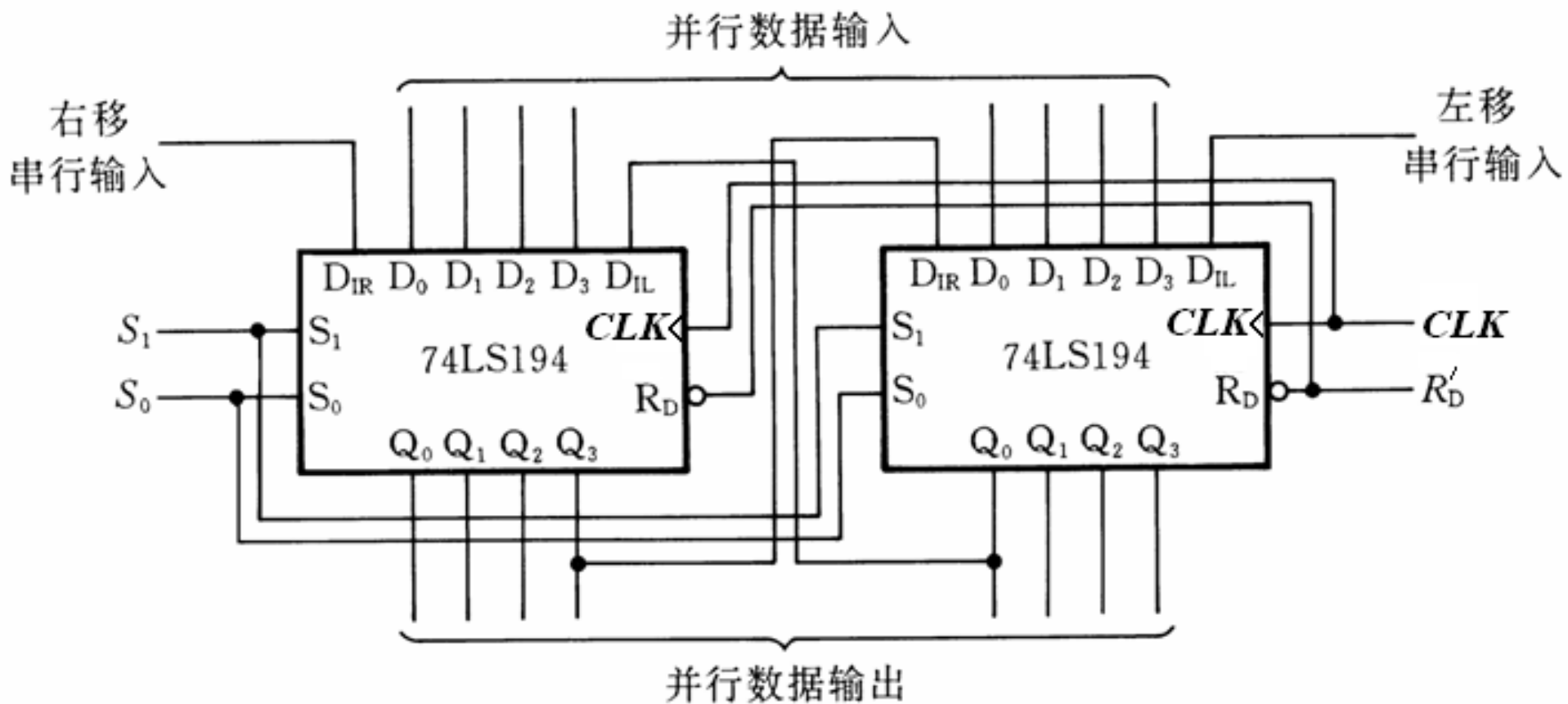
(a) 引脚排列图



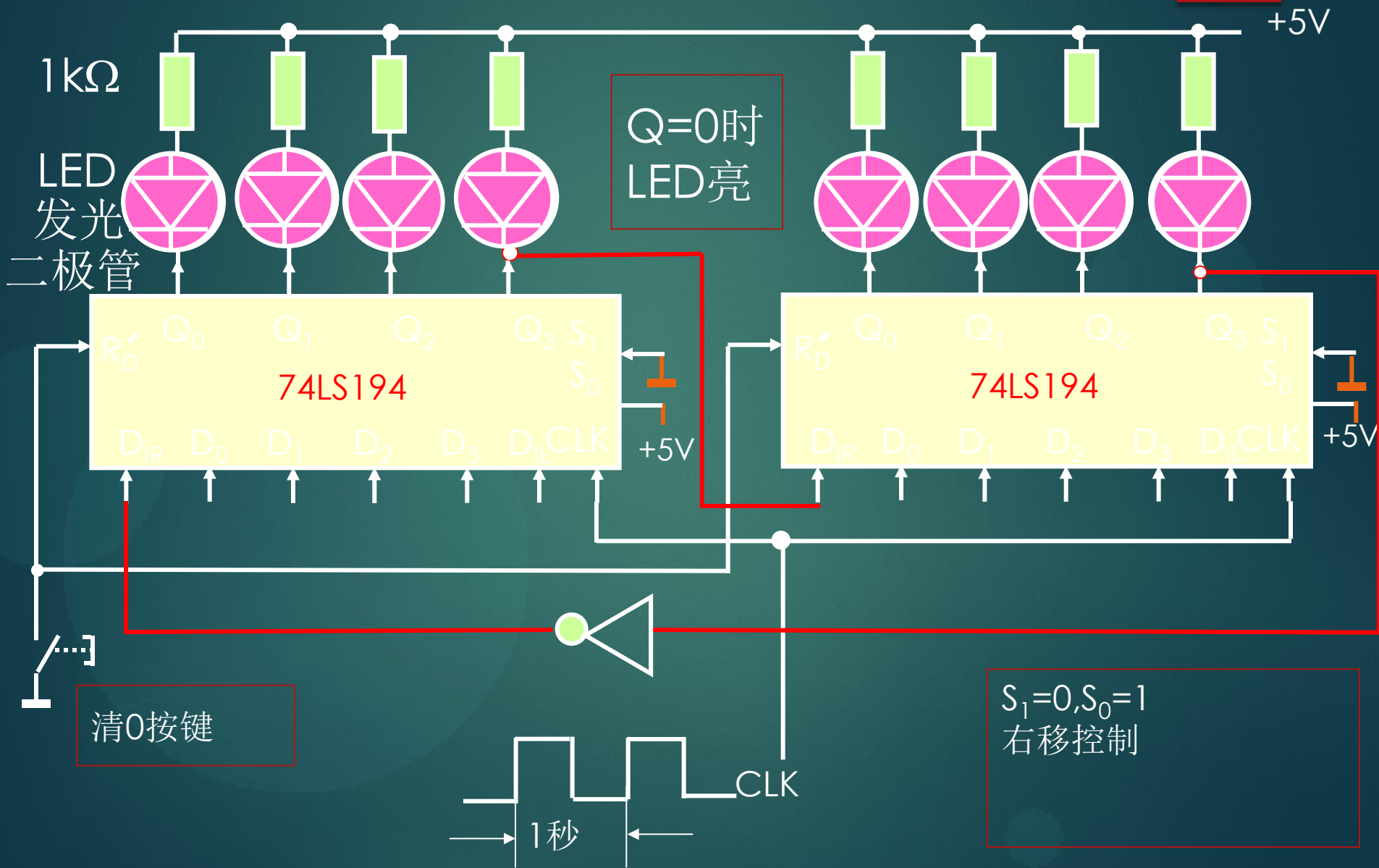
(b) 逻辑功能示意图

R'_D	S_1	S_0	CLK	工作状态
0	×	×	×	异步清零
1	0	0	×	保持
1	0	1	↑	右移
1	1	0	↑	左移
1	1	1	×	并行输入

2片74LS194A接成8位双向移位寄存器



用双向移位寄存器74LS194组成节日彩灯控制电路



寄存器小结：

寄存器是用来存放二进制数据或代码的电路，是一种基本时序电路。任何现代数字系统都必须把需要处理的数据和代码先寄存起来，以便随时取用。

寄存器小结：

寄存器分为基本寄存器和移位寄存器两大类。基本寄存器的数据只能并行输入、并行输出。移位寄存器中的数据可以在移位脉冲作用下依次逐位右移或左移，数据可以并行输入、并行输出，串行输入、串行输出，并行输入、串行输出，串行输入、并行输出。

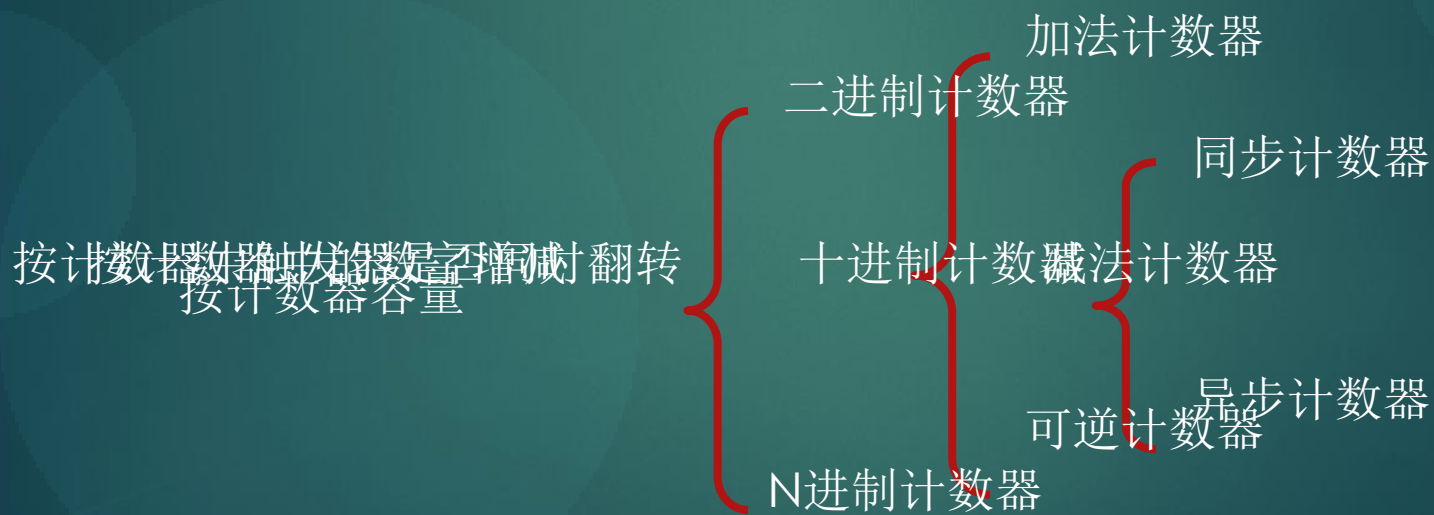
寄存器小结：

寄存器的应用很广，特别是移位寄存器，不仅可将串行数码转换成并行数码，或将并行数码转换成串行数码，还可以很方便地构成移位寄存器型计数器和顺序脉冲发生器等电路。

计数器

在数字电路中，能够记忆输入脉冲个数的电路称为计数器。

分类：



计数器

同步计数器

二进制计数器

加法计数器

减法计数器

可逆计数器

十进制计数器

加法计数器

减法计数器

可逆计数器

N进制计数器

.

.

.

.

.

.

异步计数器

二进制计数器

十进制计数器

N进制计数器

一、同步计数器

n 位二进制同步加法计数器的电路连接规律：

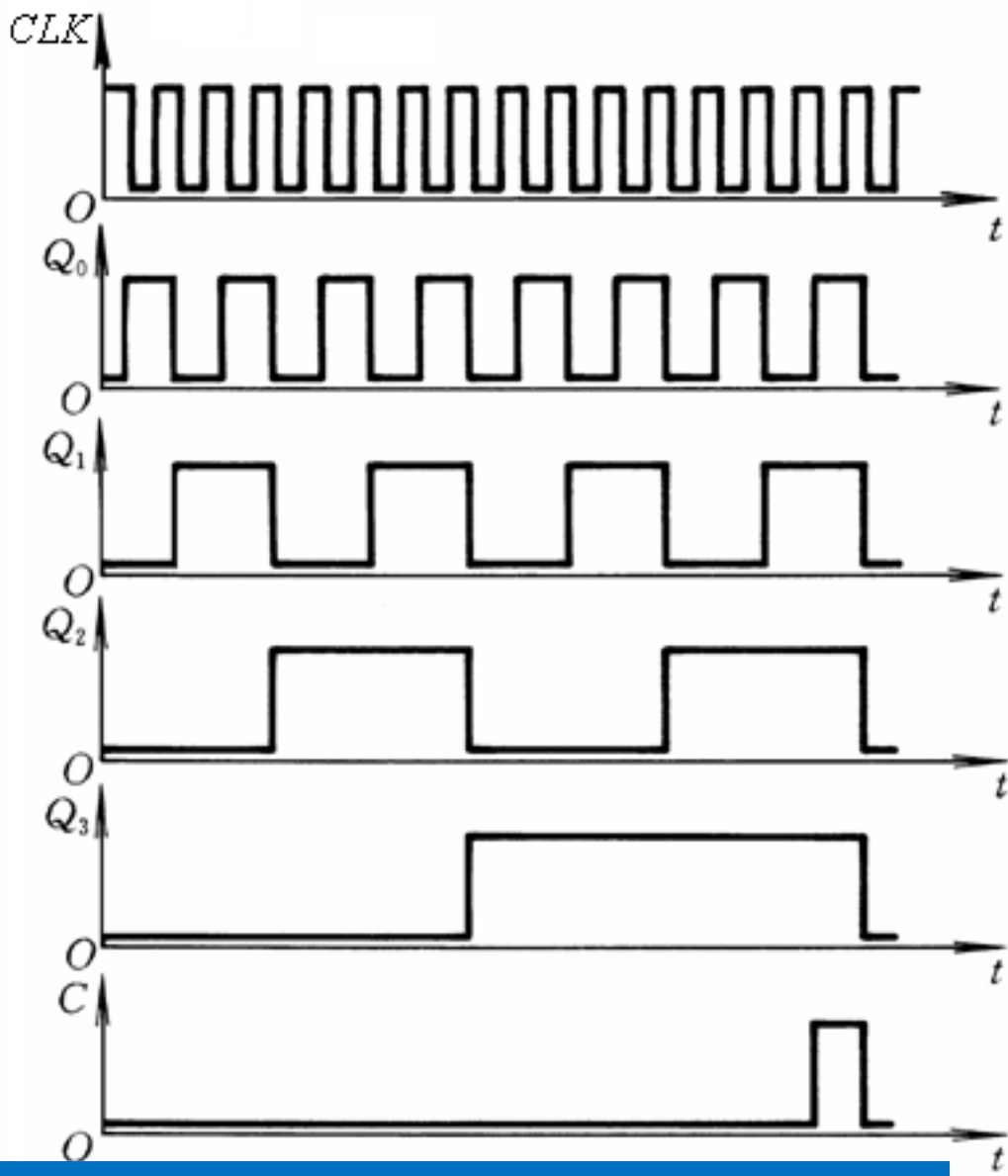
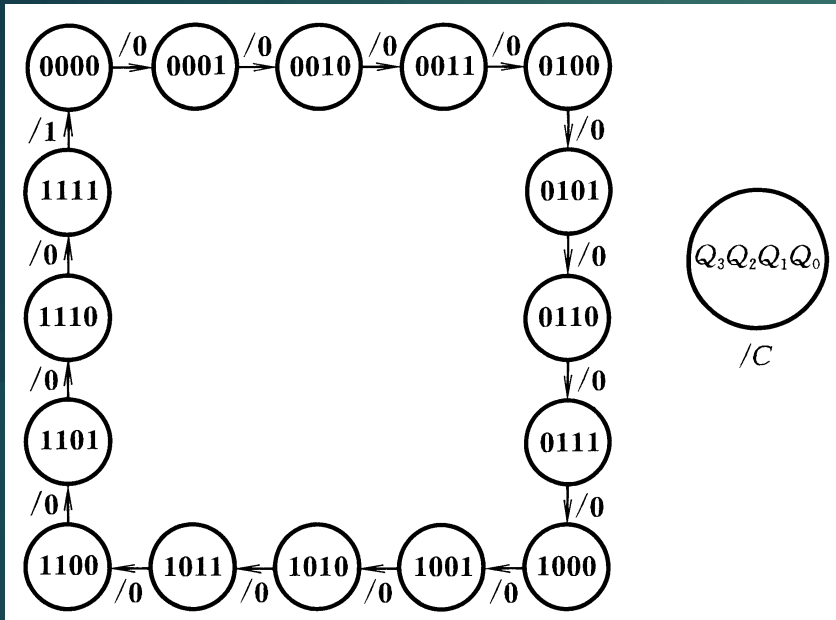
驱动方程

$$\left\{ \begin{array}{l} J_0 = K_0 = 1 \\ J_1 = K_1 = Q_0 \\ J_2 = K_2 = Q_1 Q_0 \\ \dots\dots\dots \\ J_{n-1} = K_{n-1} = Q_{n-2} Q_{n-3} \cdots Q_1 Q_0 \end{array} \right.$$

输出方程

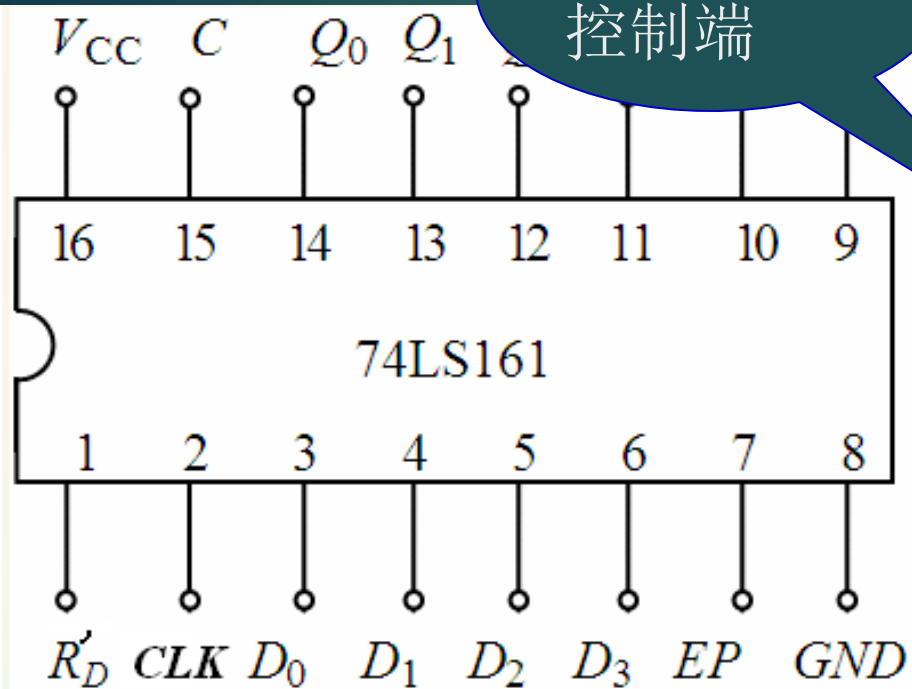
$$C = Q_{n-1} Q_{n-2} \cdots Q_1 Q_0$$

4位二进制同步加法计数器



若计数脉冲频率为 f_0 ，则 Q_0 、 Q_1 、 Q_2 、 Q_3 端输出脉冲的频率依次为 f_0 的 $1/2$ 、 $1/4$ 、 $1/8$ 、 $1/16$ 。因此又称为分频器。

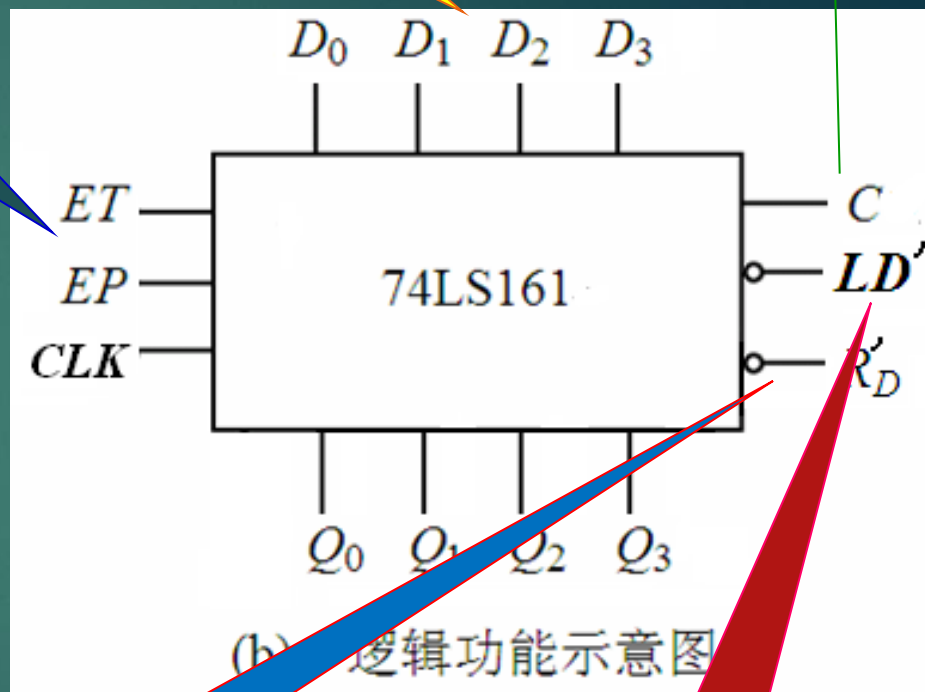
4位集成二进制同步加法计数器74LS161/163



工作状态
控制端

数据输入端

进位
输出

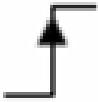
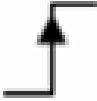


(b) 逻辑功能示意图

异步复位端



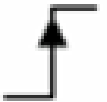
预置数控制
端

4位同步二进制计数器74161功能表

CLK	R_D'	LD'	EP	ET	工作状态
×	0	×	×	×	清零
	1	0	×	×	预置数
×	1	1	0	1	保持
×	1	1	×	0	保持 (C=0)
	1	1	1	1	计数

74161具有异步清零和同步置数功能.

4位同步二进制计数器74163功能表

CLK	R'_D	LD'	EP	ET	工作状态
	0	×	×	×	清零
	1	0	×	×	预置数
×	1	1	0	1	保持
×	1	1	×	0	保持 (C=0)
	1	1	1	1	计数

74163具有同步清零和同步置数功能。

74LS163的引脚排列和74LS161相同，不同之处是74LS163采用同步清零方式。

n 位二进制同步减法计数器的连接规律:

驱动方程

$$\begin{cases} J_0 = K_0 = 1 \\ J_1 = K_1 = Q'_0 \\ J_2 = K_2 = Q'_1 Q'_0 \\ \dots\dots\dots \\ J_{n-1} = K_{n-1} = Q'_{n-2} Q'_{n-3} \cdots Q'_1 Q'_0 \end{cases}$$

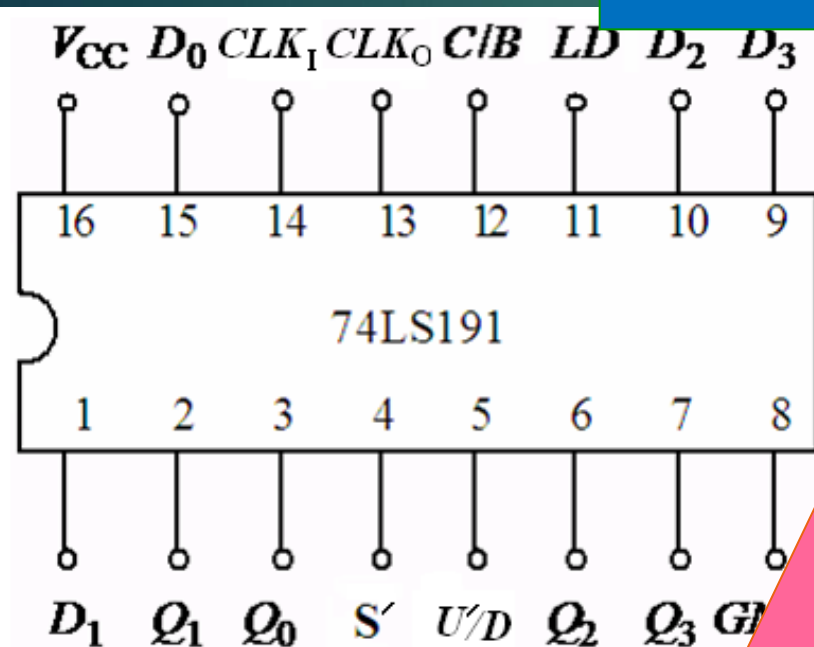
输出方程

$$B = Q'_{n-1} Q'_{n-2} \cdots Q'_1 Q'_0$$

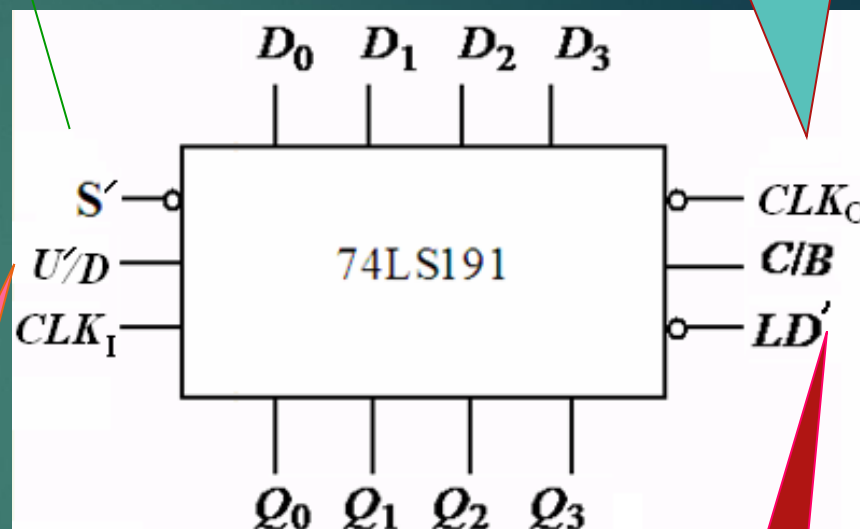
4位集成二进制同步可逆计数器74LS191

使能端

串行时钟输出



(a) 引脚排列图

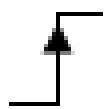
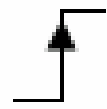


(b) 逻辑功能示意图

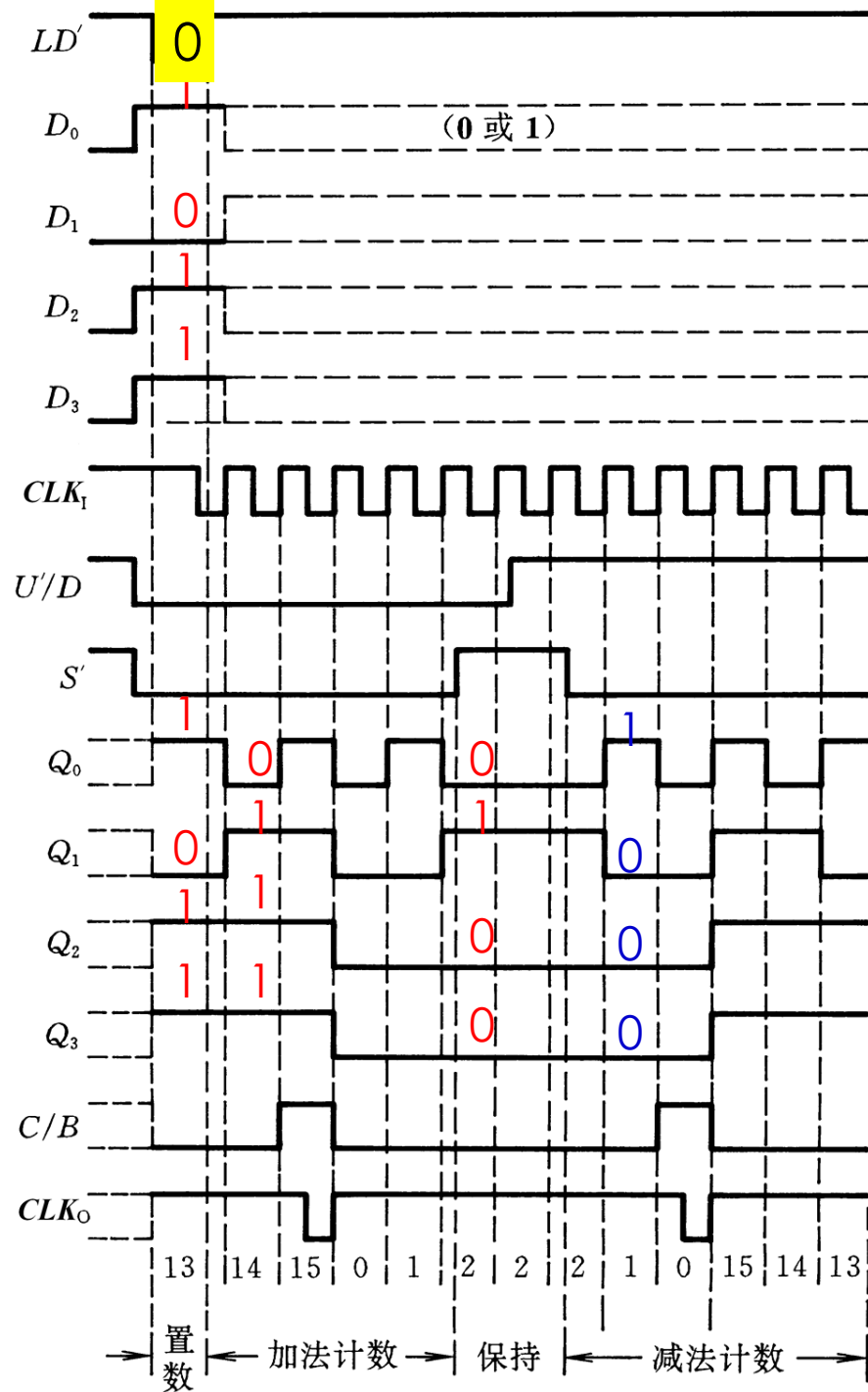
加 / 减控制端

预置数控制端

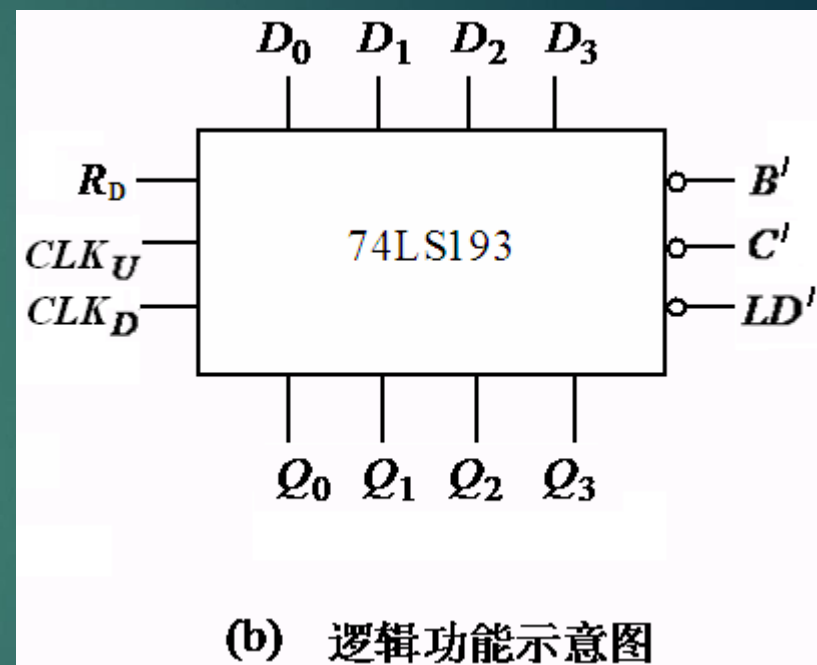
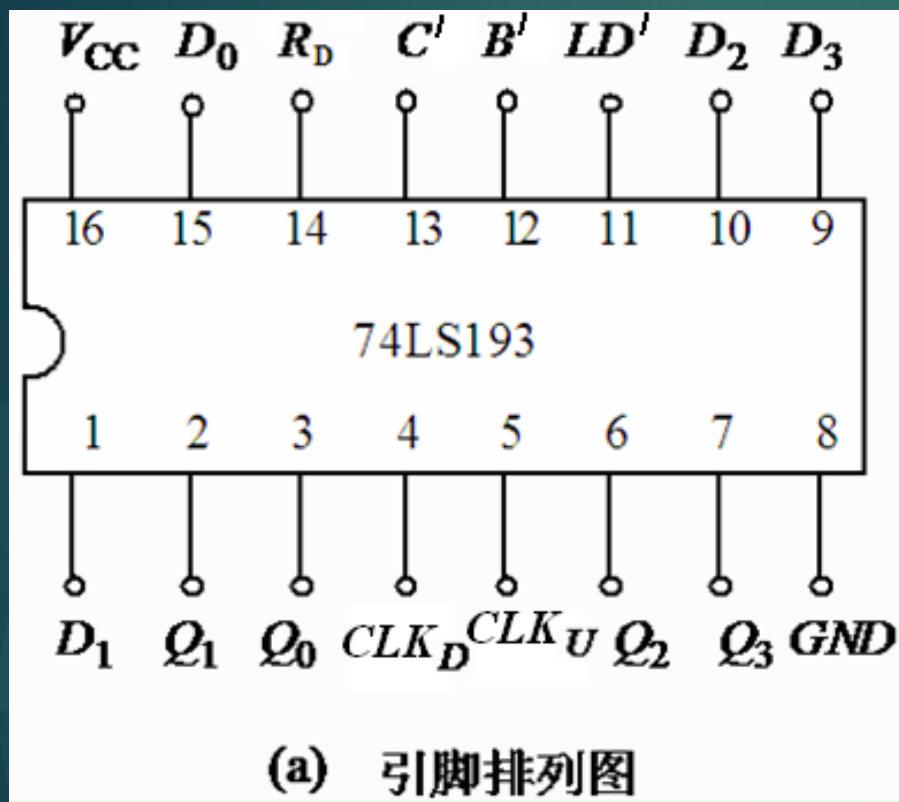
4位同步二进制可逆计数器74LS191功能表

CLK_I	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

74LS191具有异步置数功能.



双时钟加/减计数器74LS193



74LS193具有异步清零和异步置数功能.

2、同步十进制计数器

同步十进制加法计数器:在同步二进制加法计数器基础上修改而来.

同步十进制加法计数器74LS160与74LS161逻辑图和功能表均相同,所不同的是74LS160是十进制而74LS161是十六进制。

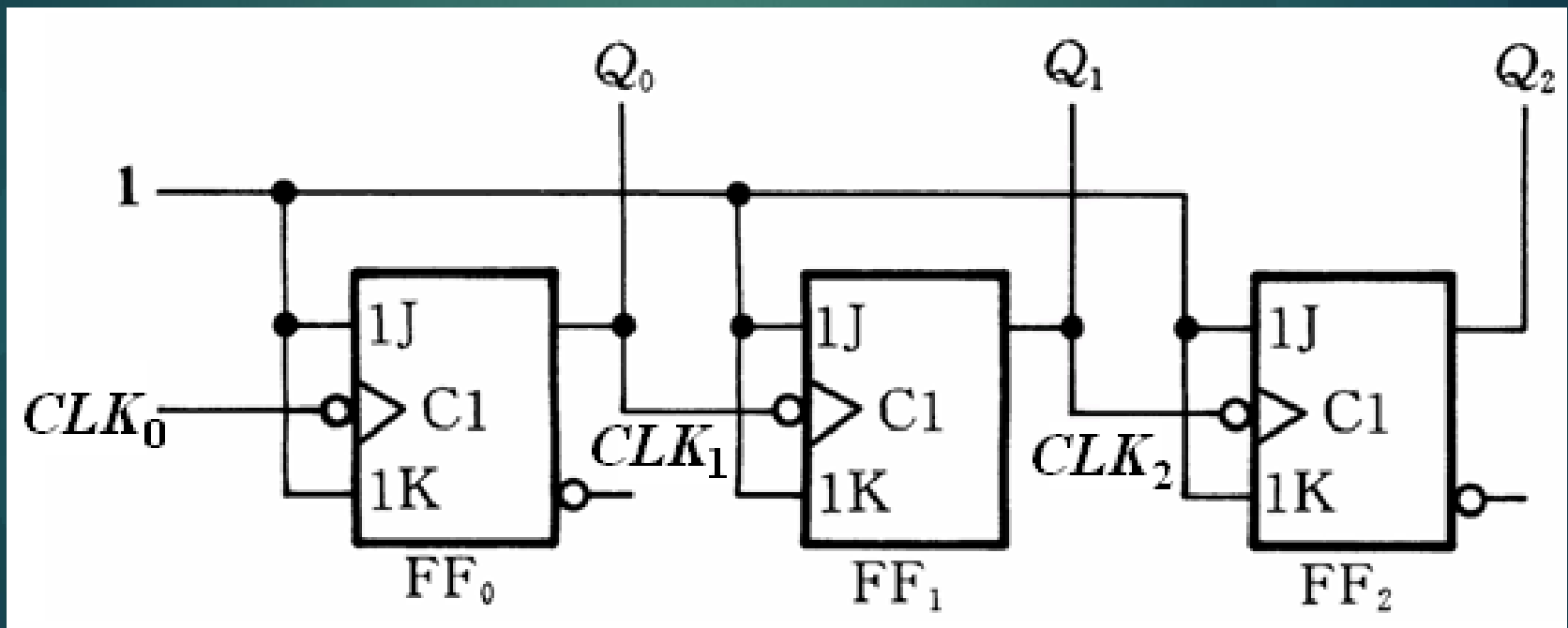
同步十进制可逆计数器也有单时钟和双时钟两种结构形式。属于单时钟的有74LS190等，属于双时钟的有74LS192等。

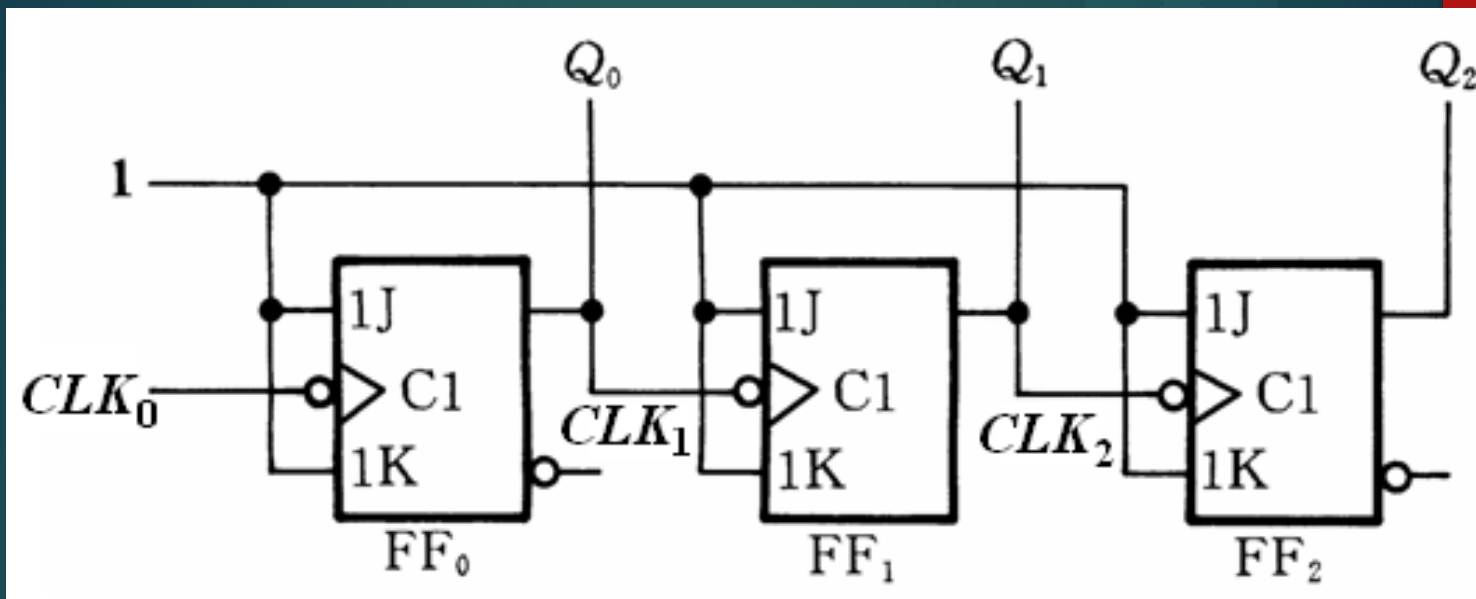
74LS190与74LS191逻辑图和功能表均相同；
74LS192与74LS193逻辑图和功能表均相同。

二、异步计数器

1、异步二进制计数器

3位异步二进制加法计数器

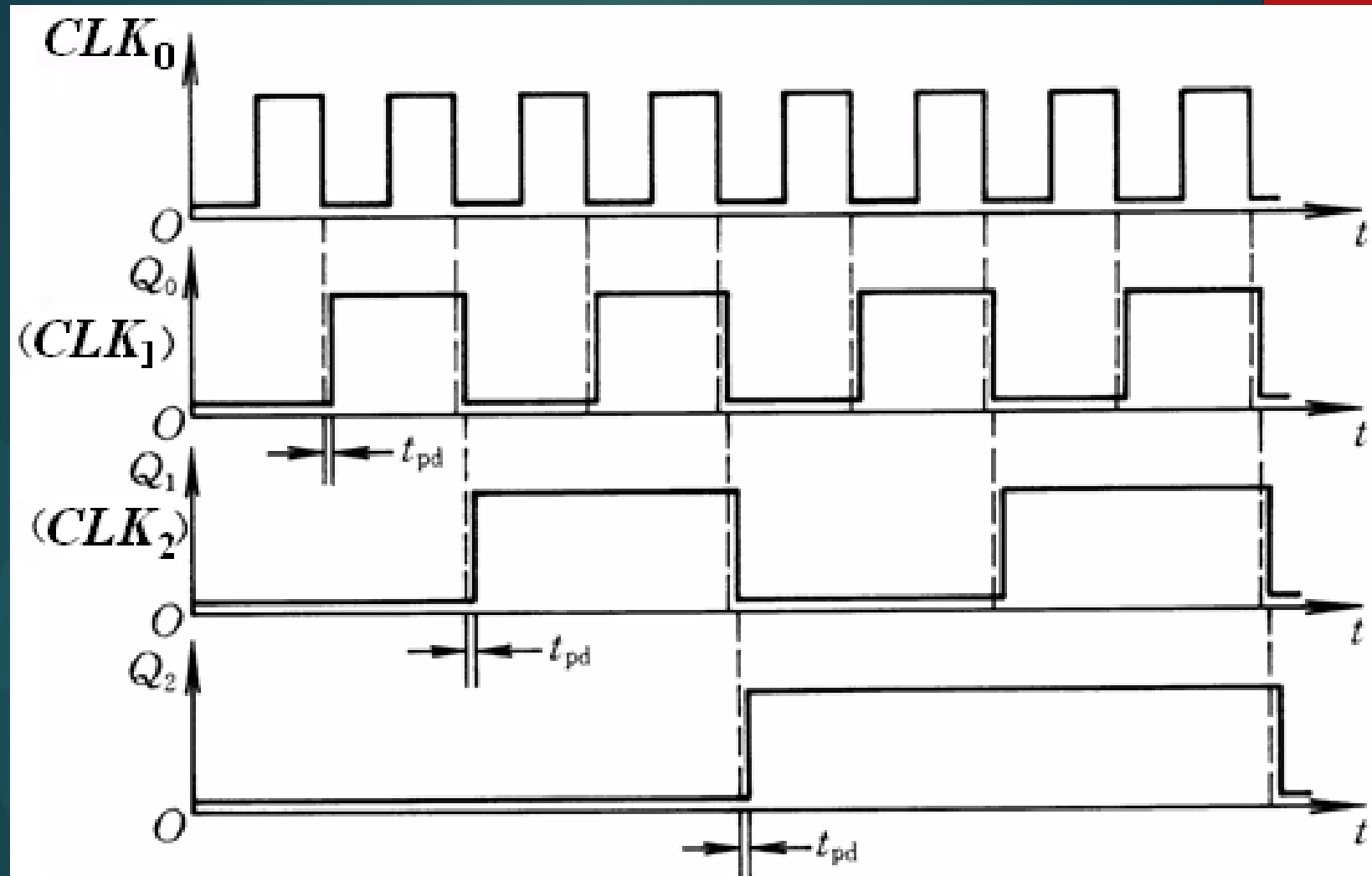




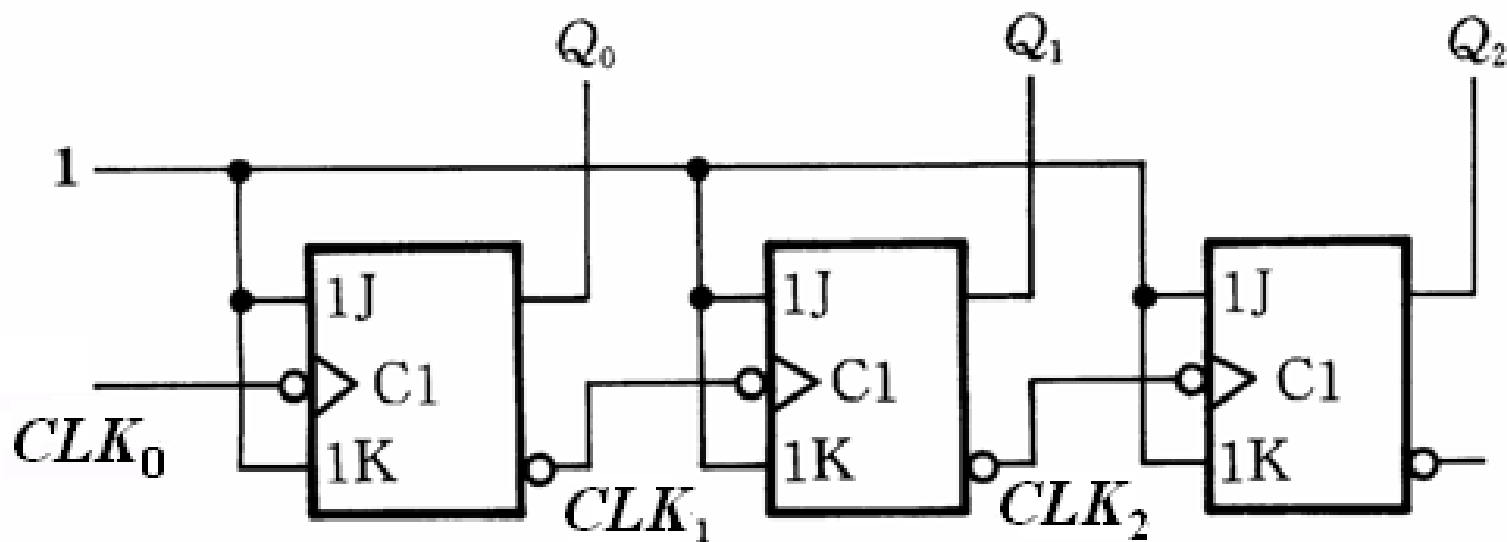
$$\left\{ \begin{array}{l} J_0 = K_0 = 1 \\ J_1 = K_1 = 1 \\ J_2 = K_2 = 1 \end{array} \right.$$

触发器为下降沿触发， Q_0 接
 CLK_1 , Q_1 接 CLK_2 。

若上升沿触发，则应 Q_0' 接
 CLK_1 , Q_1' 接 CLK_2 。

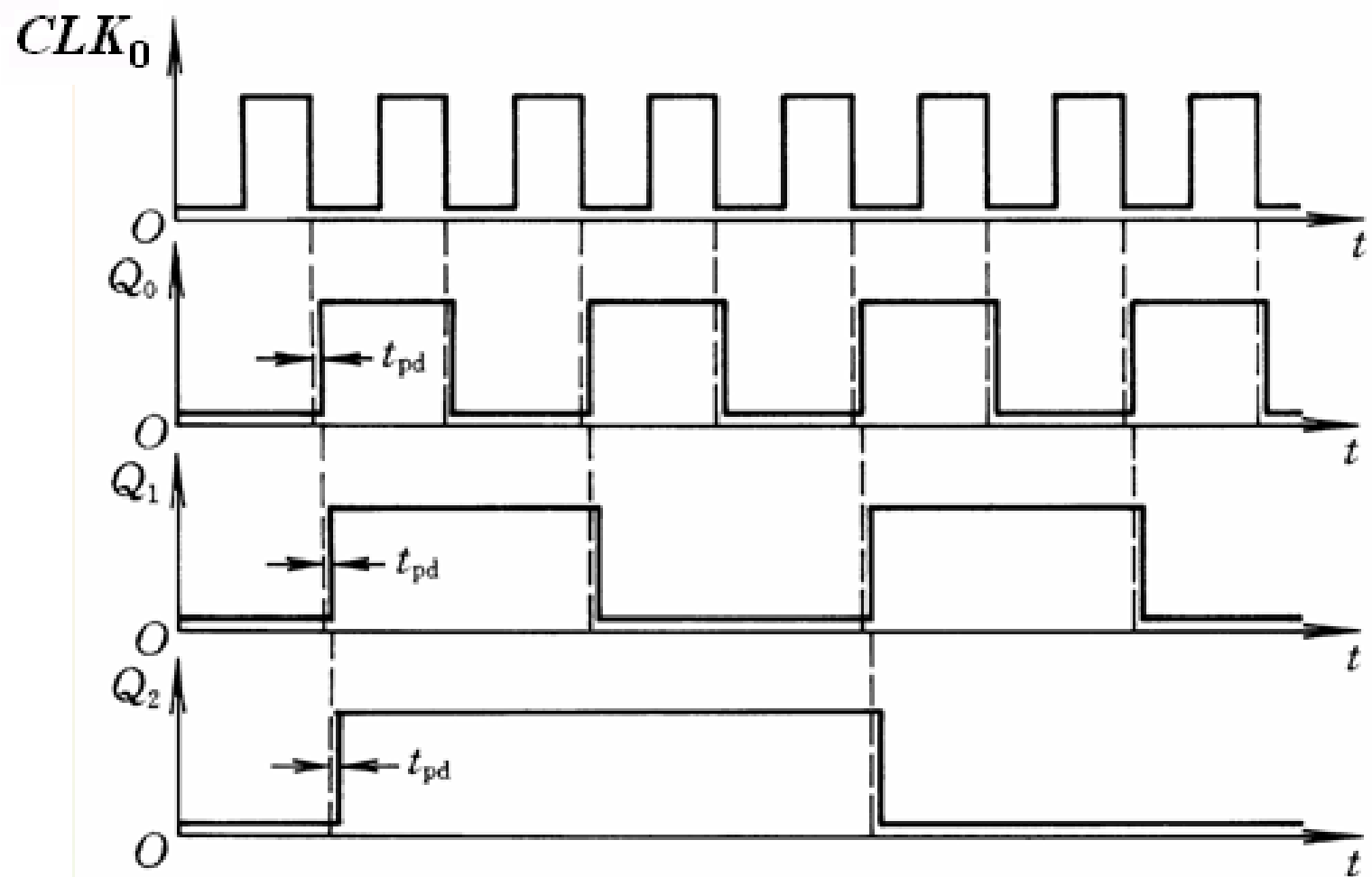


3位异步二进制减法计数器



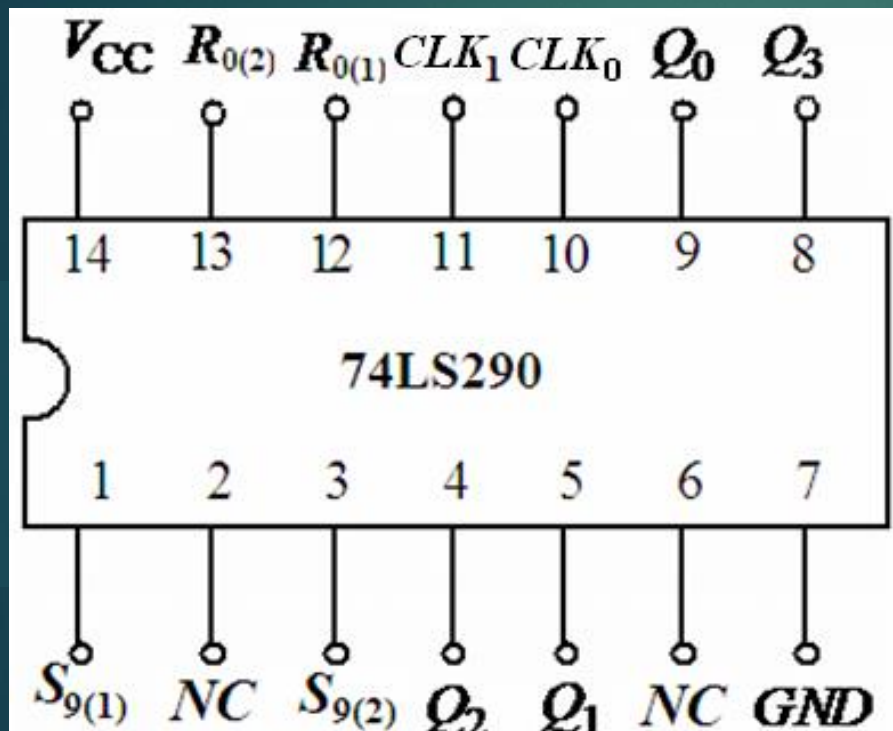
$$\begin{cases} J_0 = K_0 = 1 \\ J_1 = K_1 = 1 \\ J_2 = K_2 = 1 \end{cases}$$

触发器为下降沿触发， Q_0' 接 CLK_1 ， Q_1' 接 CLK_2 。
若上升沿触发，则应 Q_0 接 CLK_1 ， Q_1 接 CLK_2 。



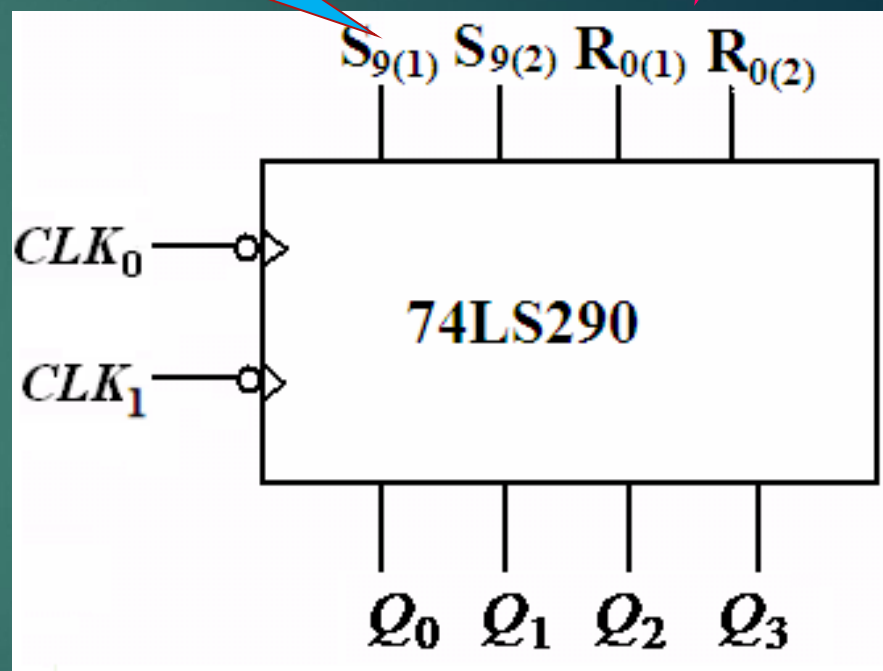
2、异步十进制计数器

异步二一五一十进制计数器74LS290

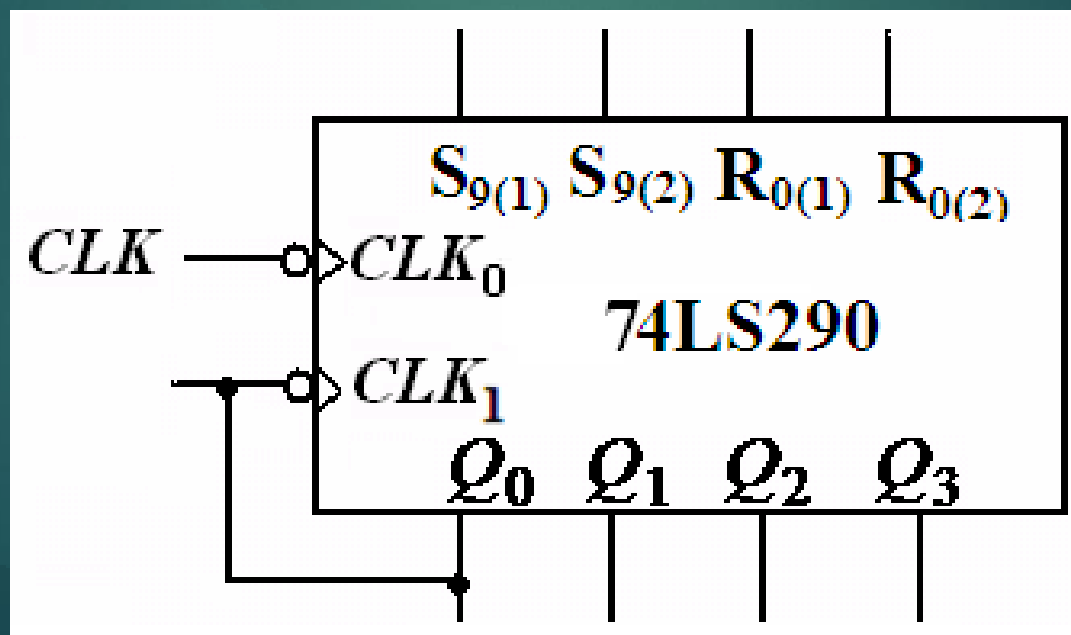


置9端

置0端



若计数脉冲由 CLK_0 端输入，输出由 Q_0 端引出，即得到二进制计数器；
若计数脉冲由 CLK_1 端输入，输出由 $Q_1 \sim Q_3$ 引出，即是五进制计数器；若将 CLK_1 与 Q_0 相连，同时以 CLK_0 为输入端，输出由 $Q_0 \sim Q_3$ 引出，则得到8421码十进制计数器。



74LS290功能表

输 入				输 出			
$R_{0(1)} \cdot R_{0(2)}$	$S_{9(1)} \cdot S_{9(2)}$	CLK_0	CLK_1	Q_3	Q_2	Q_1	Q_0
1	0	×	×	0	0	0	0
×	1	×	×	1	0	0	1
0	0	CLK	0	二进制计数			
0	0	0	CLK	五进制计数			
0	0	CLK	Q_0	8421码十进制计数			

异步计数器特点

优点：结构简单

缺点：（1）工作频率较低；
（2）在电路状态译码时存在竞争一冒险现象。

三、任意进制计数器的构成方法

利用现有的N进制计数器构成任意进制（M）计数器时，如果 $M < N$ ，则只需一片N进制计数器；如果 $M > N$ ，则要多片N进制计数器。

实现方法

置零法（复位法）

置数法（置位法）

置零法：适用于有清零输入端的集成计数器。原理是不管输出处于哪一状态，只要在清零输入端加一有效电平电压，输出会立即从那个状态回到0000状态，清零信号消失后，计数器又可以从0000开始重新计数。

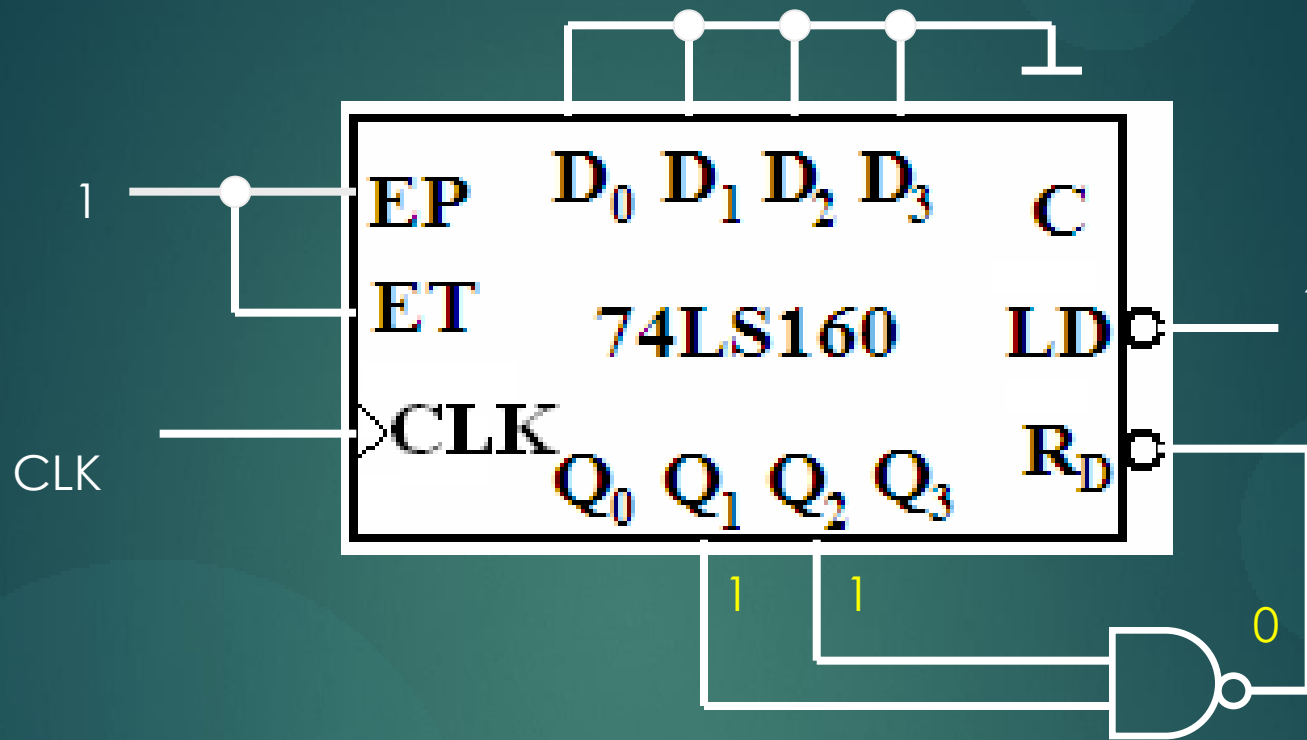
置数法：适用于具有预置功能的集成计数器。对于具有预置数功能的计数器而言，在其计数过程中，可以将它输出的任意一个状态通过译码，产生一个预置数控制信号反馈至预置数控制端，在下一个CLK脉冲作用后，计数器会把预置数输入端 $D_0D_1D_2D_3$ 的状态置入输出端。预置数控制信号消失后，计数器就从被置入的状态开始重新计数。



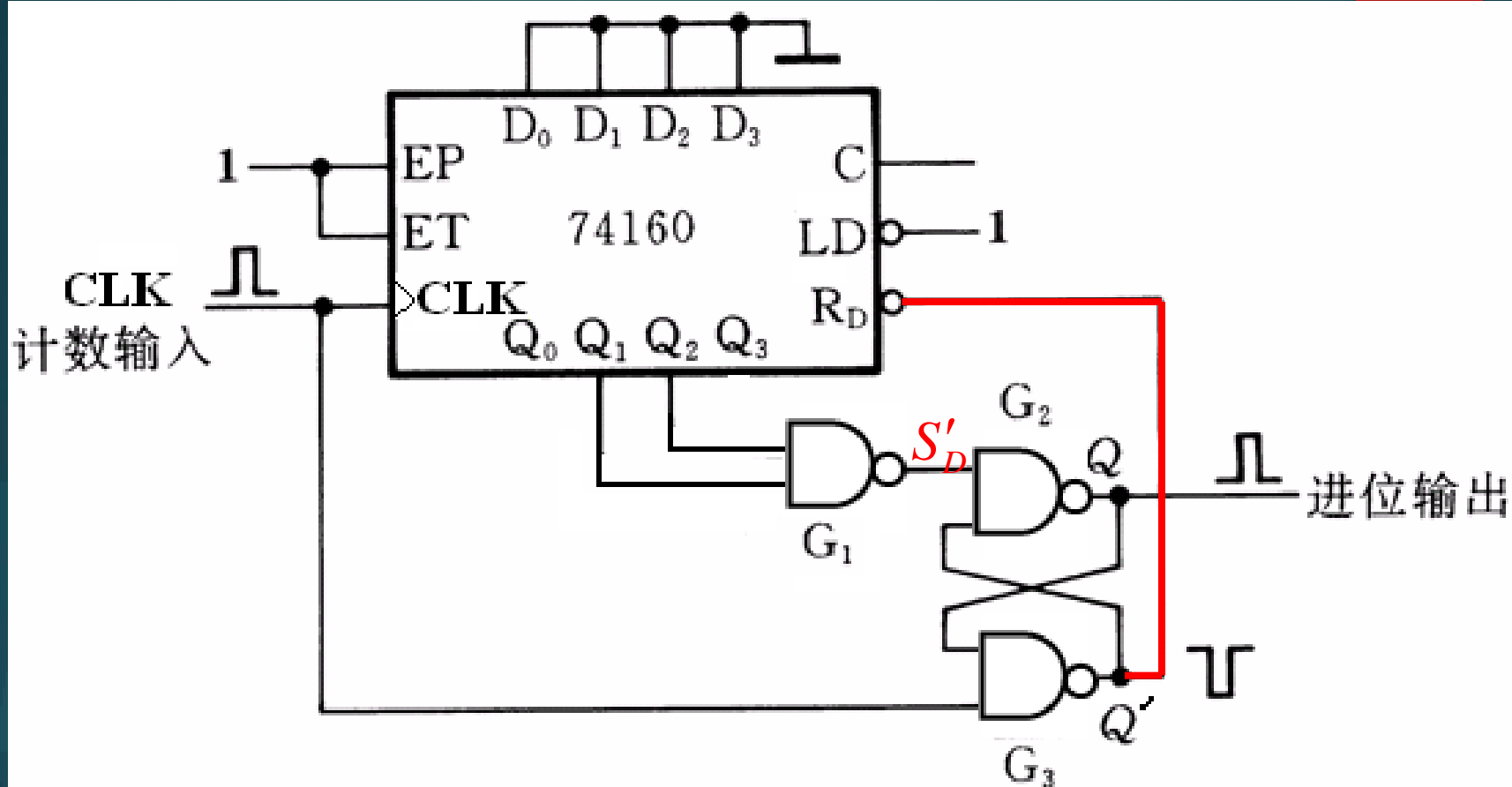
解：

74LS160具有异步清零功能





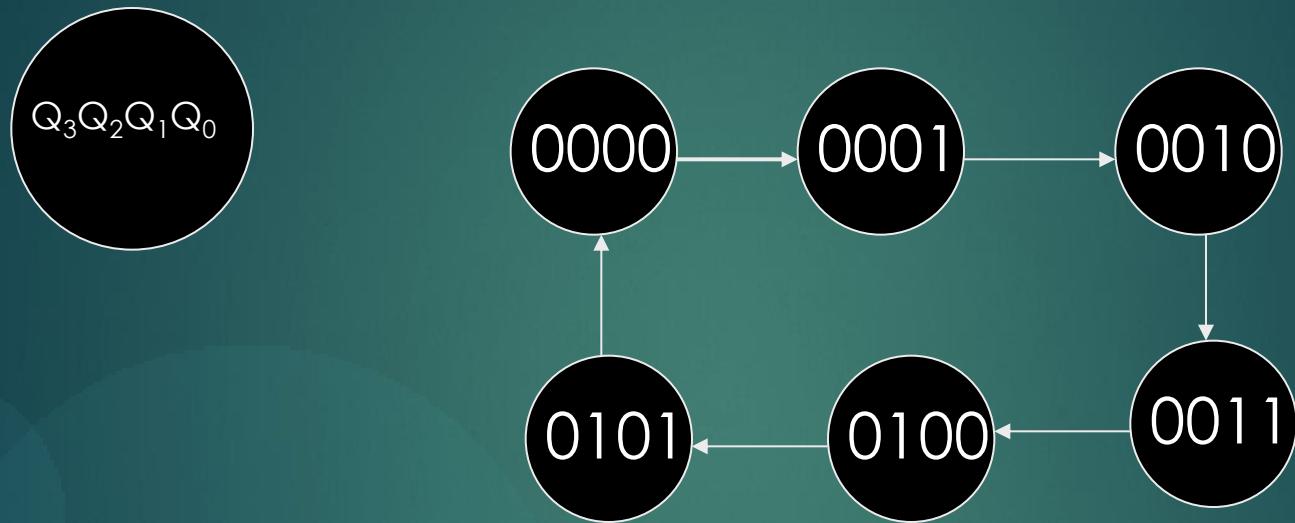
当计数器记成 $Q_3Q_2Q_1Q_0=0110$ 时，与非门输出低电平信号给 R'_D 端，将计数器置零。置零信号不是一个稳定的状态，持续时间很短，有可能导致电路误动作。



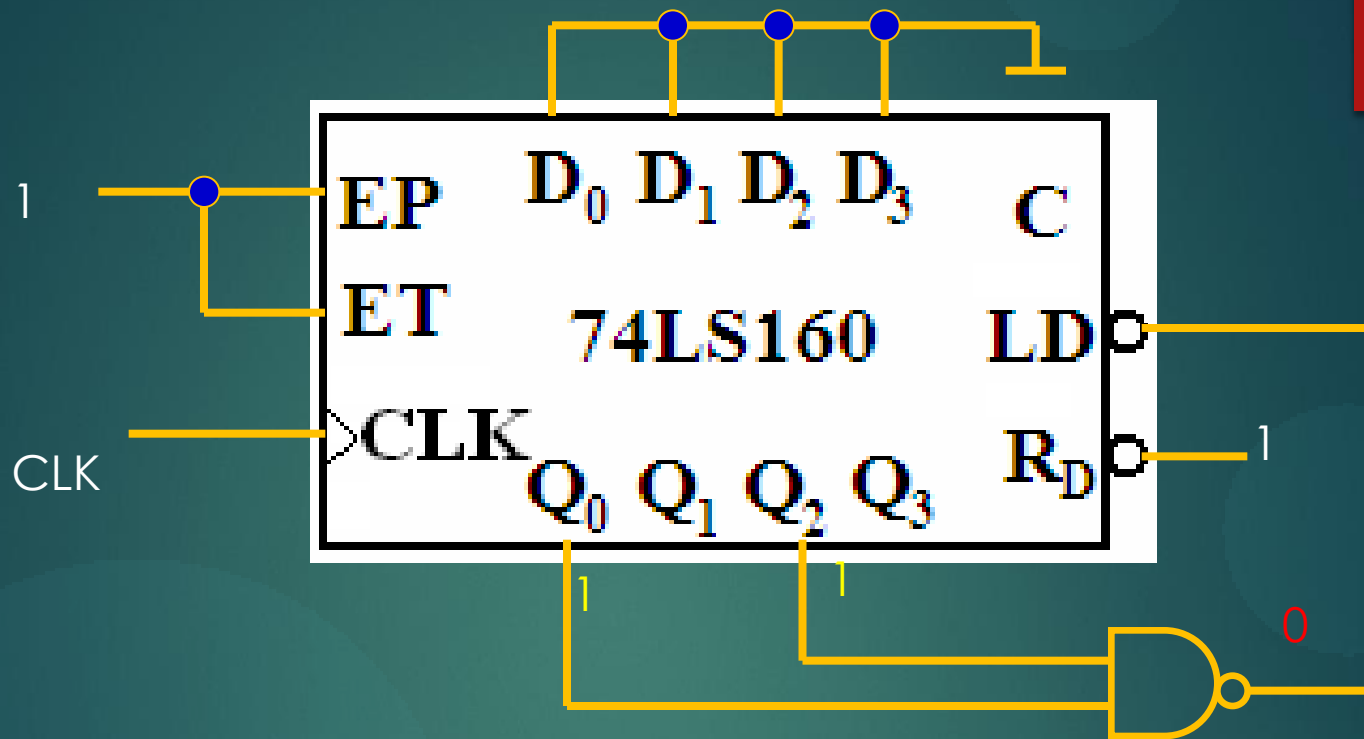
改进电路

置数法

74LS160具有同步置数功能

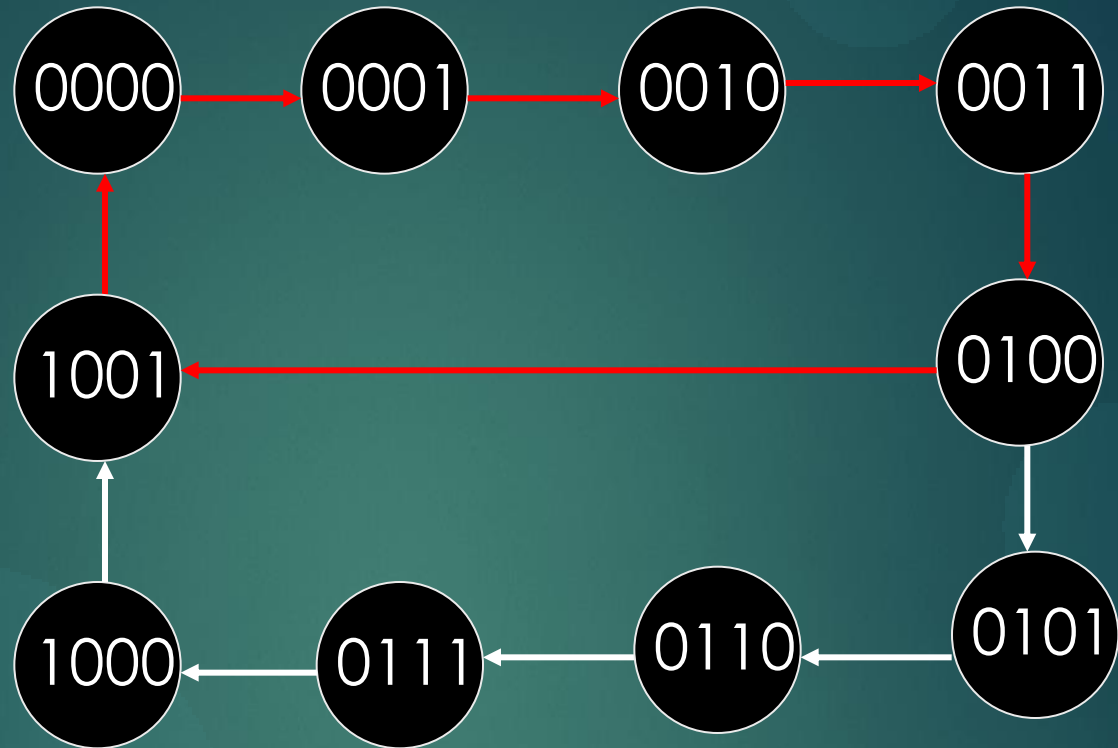


$$LD' = (Q_2 \cdot Q_0)'$$

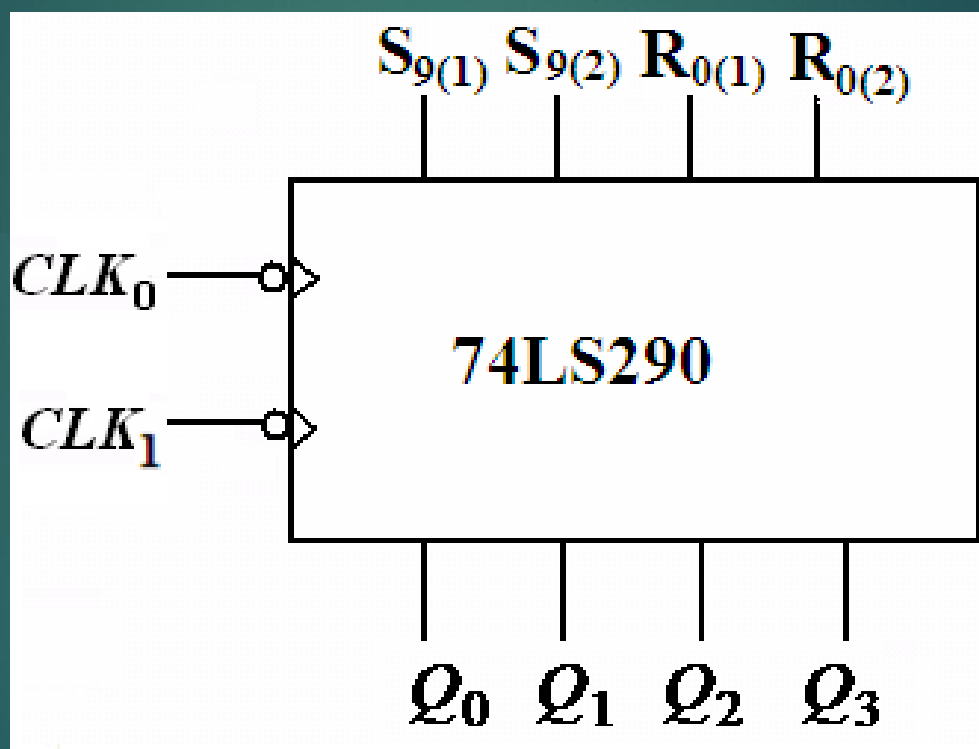


LD'=0后，还要等下一个CLK信号到来时才置入数据，而这时LD'=0的信号以稳定地建立了，提高了可靠性。

$Q_3Q_2Q_1Q_0$



用集成**异步**二—五—十进制计数器74LS290接成六进制计数器（模六）。（不用其他元件）。已知74LS290的逻辑示意图和功能表。

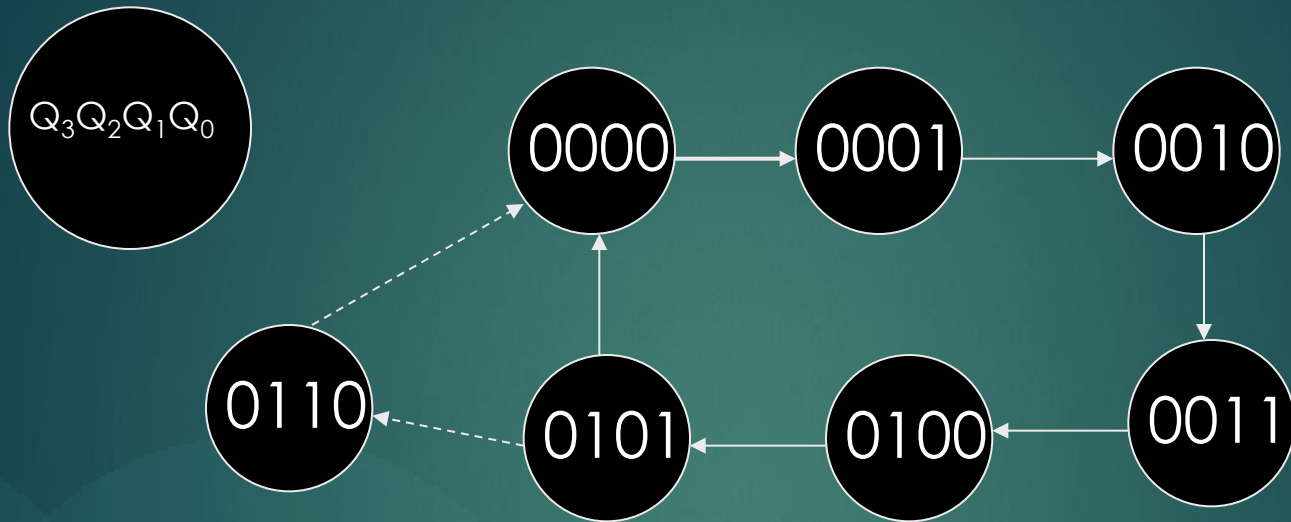


74LS290功能表

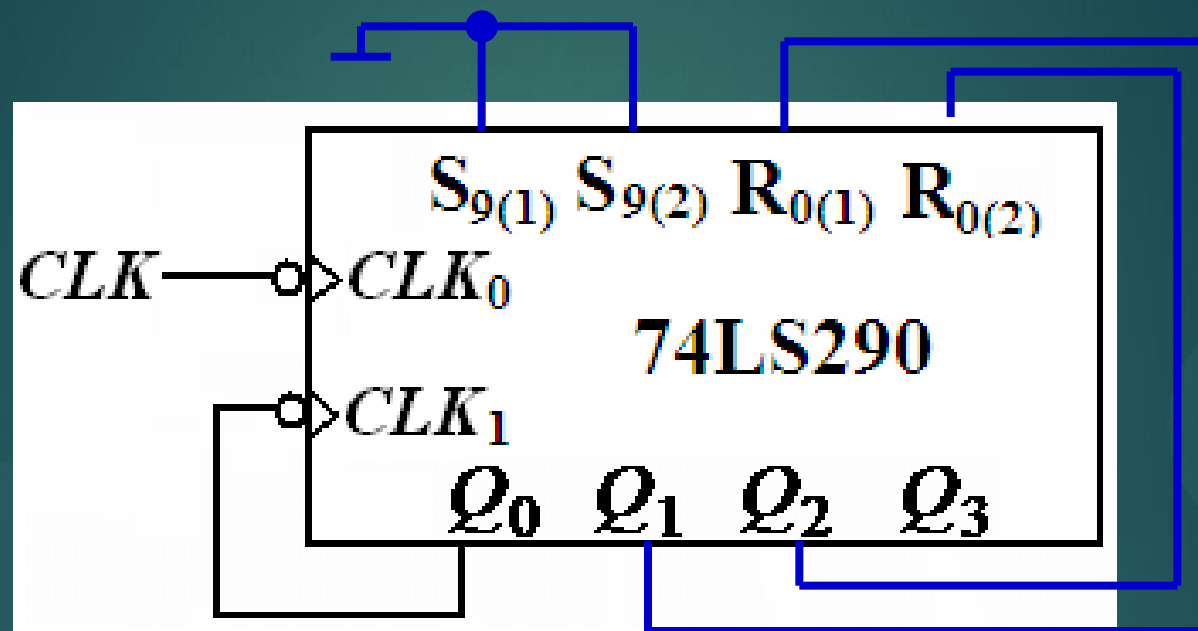
输 入				输 出			
$R_{0(1)} \cdot R_{0(2)}$	$S_{9(1)} \cdot S_{9(2)}$	CLK_0	CLK_1	Q_3	Q_2	Q_1	Q_0
1	0	×	×	0	0	0	0
×	1	×	×	1	0	0	1
0	0	CLK	0	二进制计数			
0	0	0	CLK	五进制计数			
0	0	CLK	Q_0	8421码十进制计数			

置零法构成六进制

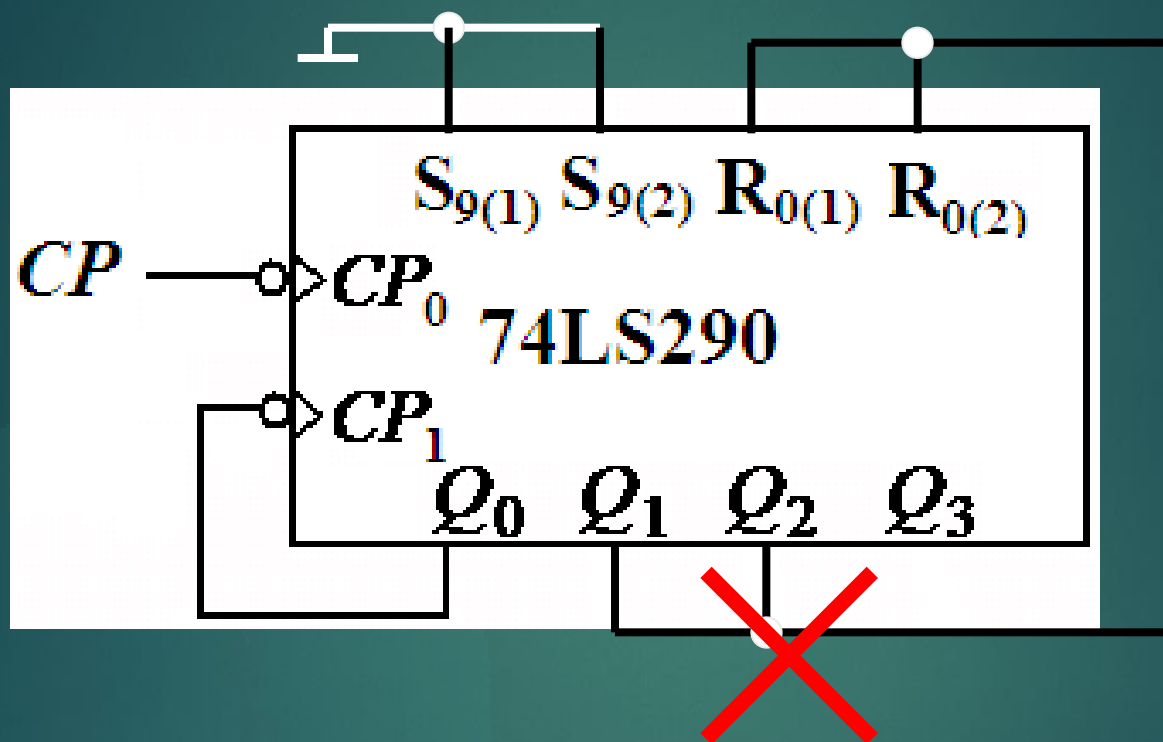
74LS290具有异步清零功能



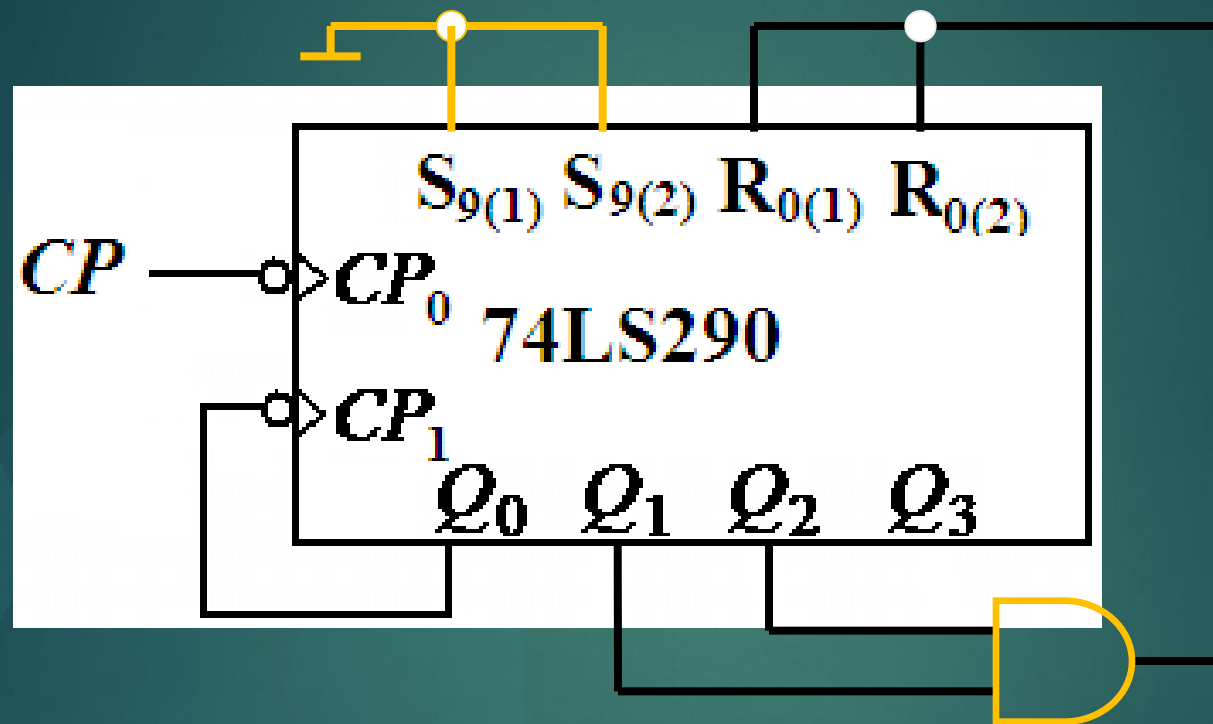
首先将74LS290接成8421BCD码的十进制计数器，即将 CLK_1 与 Q_0 相连， CLK_0 作为外部计数脉冲CLK。



以下电路连接是否正确？



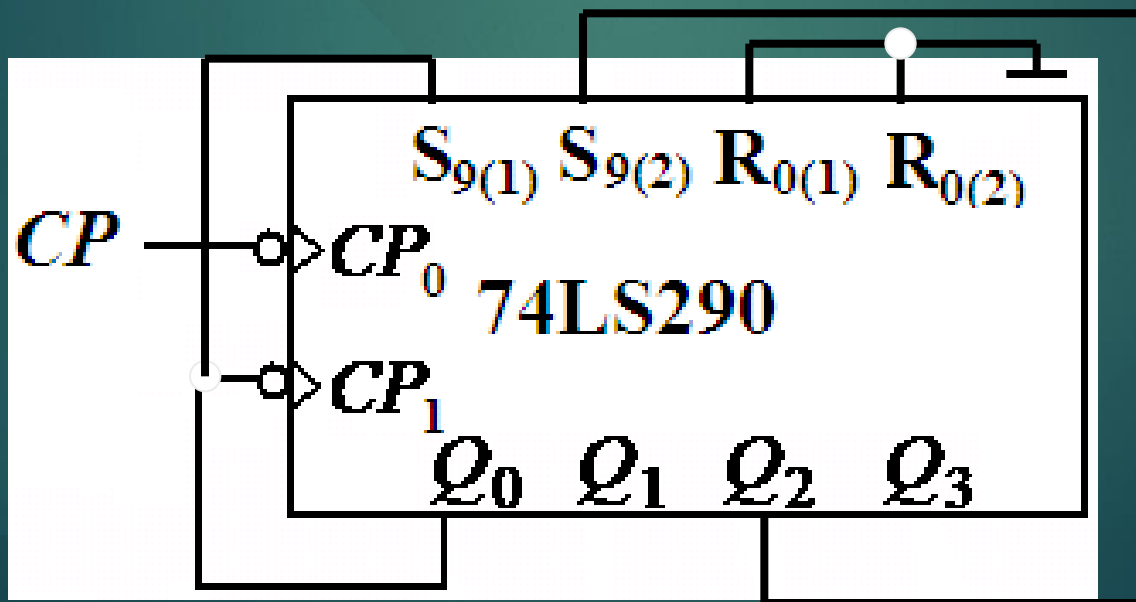
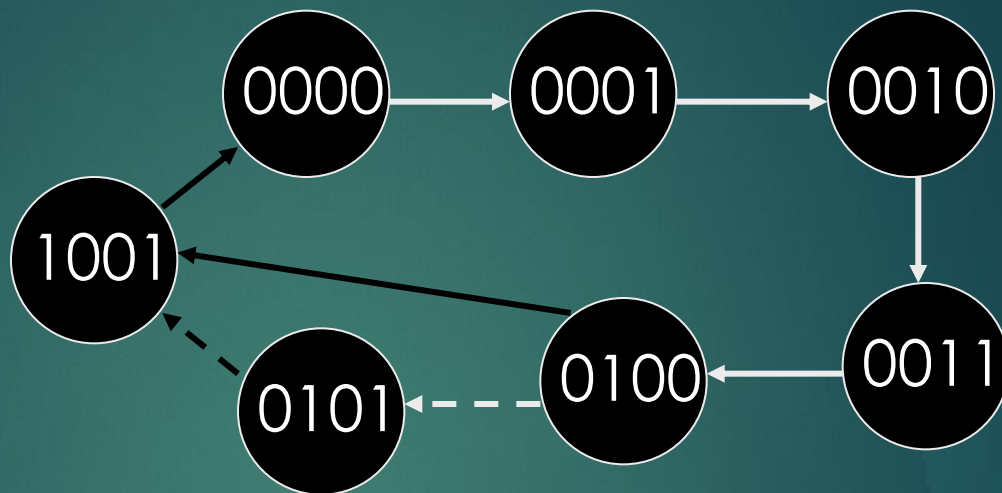
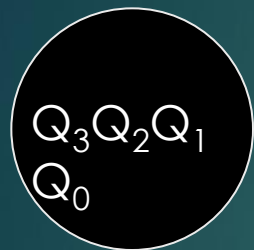
警告:切不可将输出端相互短路！！



这样接是正确的。

置9法构成六进制

74LS290具有异步置9功能



当 $M > N$ 时，需用多片 N 进制计数器组合实现

若 M 可分解为 $M = N_1 \times N_2$ (N_1 、 N_2 均小于 N)，可采用连接方式有：

串行进位方式、并行进位方式、
整体置零方式、整体置数方式

若 M 为大于 N 的素数，不可分解，则其连接方式只有：
整体置零方式、整体置数方式

串行进位方式：以低位片的进位信号作为高位片的时钟输入信号。

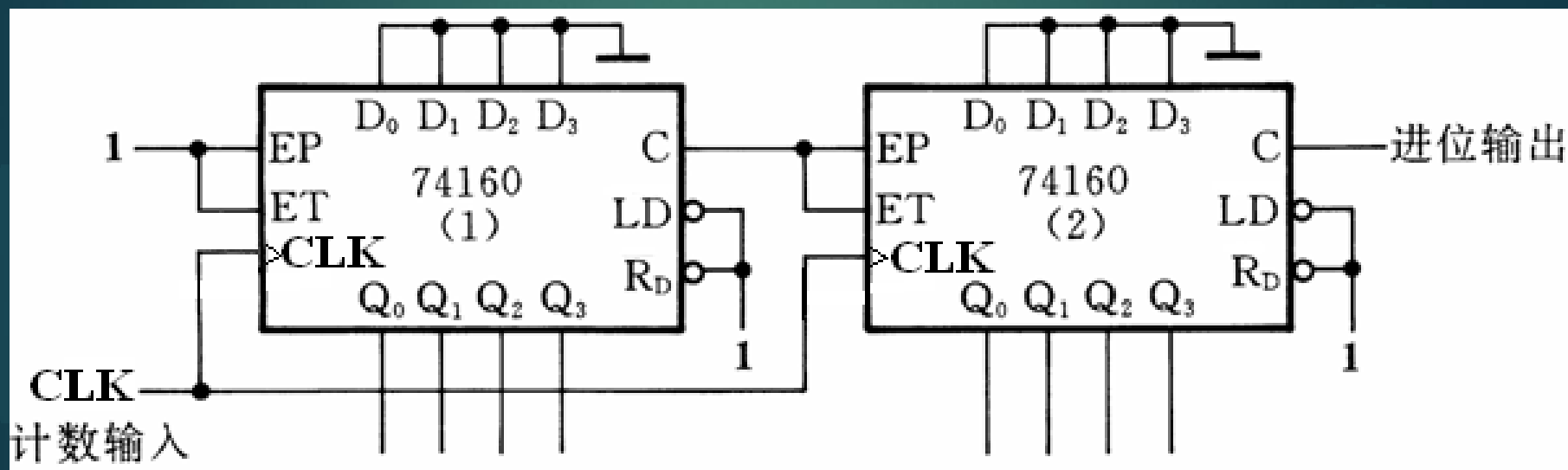
并行进位方式：以低位片的进位信号作为高位片的工作状态控制信号。

整体置零方式：首先将两片N进制计数器按最简单的方式接成一个大于M进制的计数器，然后在计数器记为M状态时使 $R_D'=0$ ，将两片计数器同时置零。

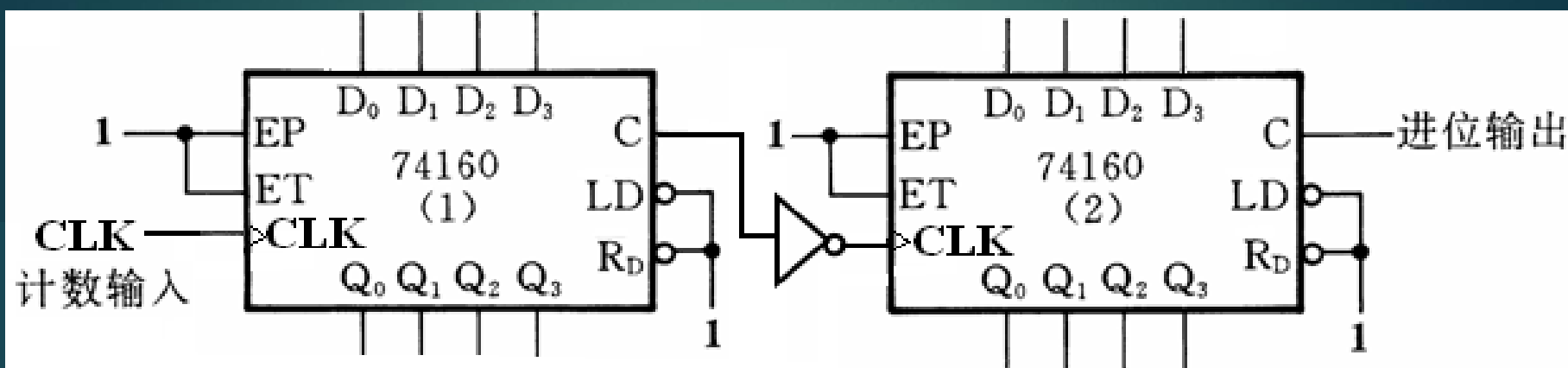
整体置数方式：首先将两片N进制计数器按最简单的方式接成一个大于M进制的计数器，然后在某一状态下使 $LD'=0$ ，将两片计数器同时置数成适当的状态，获得M进制计数器。

用两片同步十进制计数器接成百进制计数器.

解: ①并行进位方式

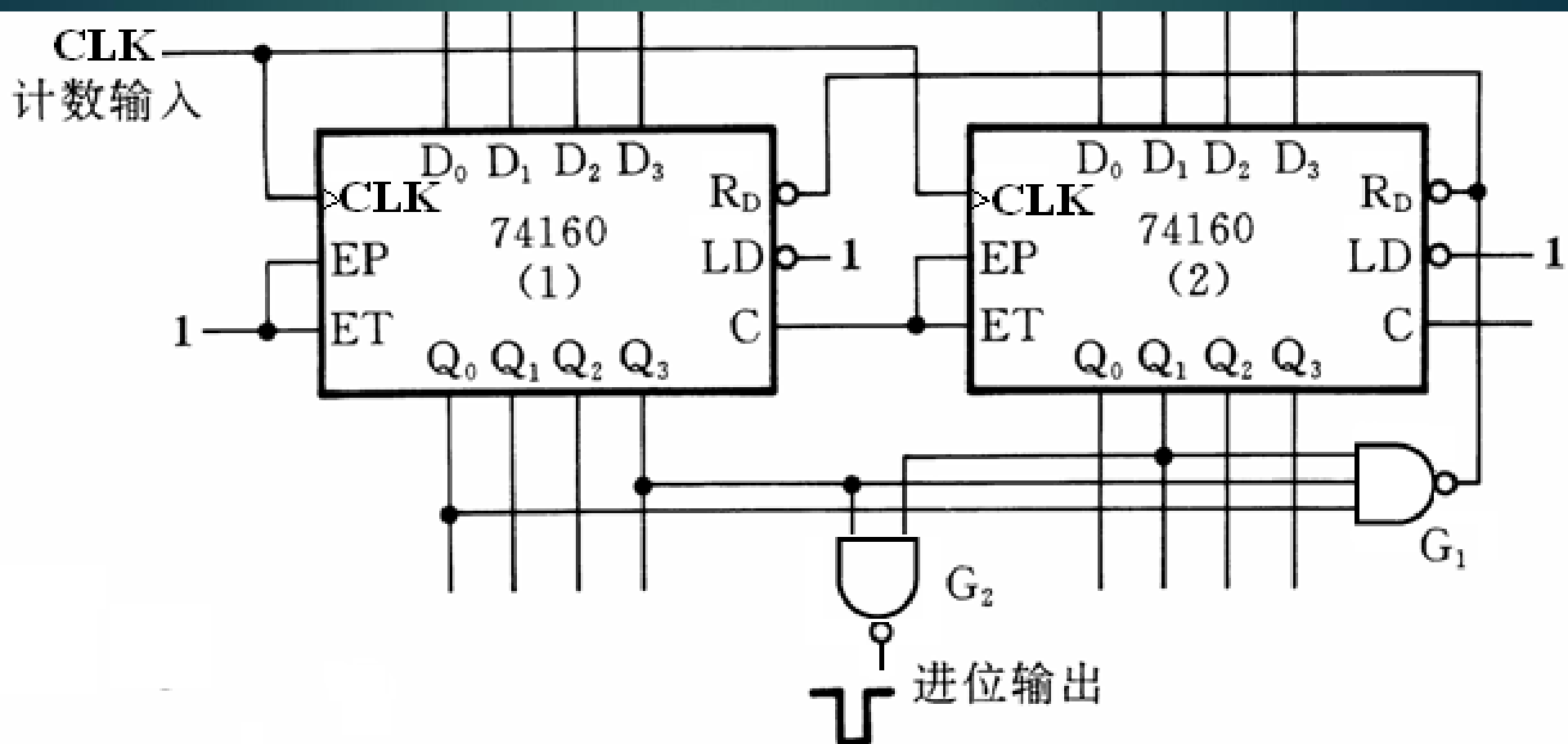


②串行进位方式

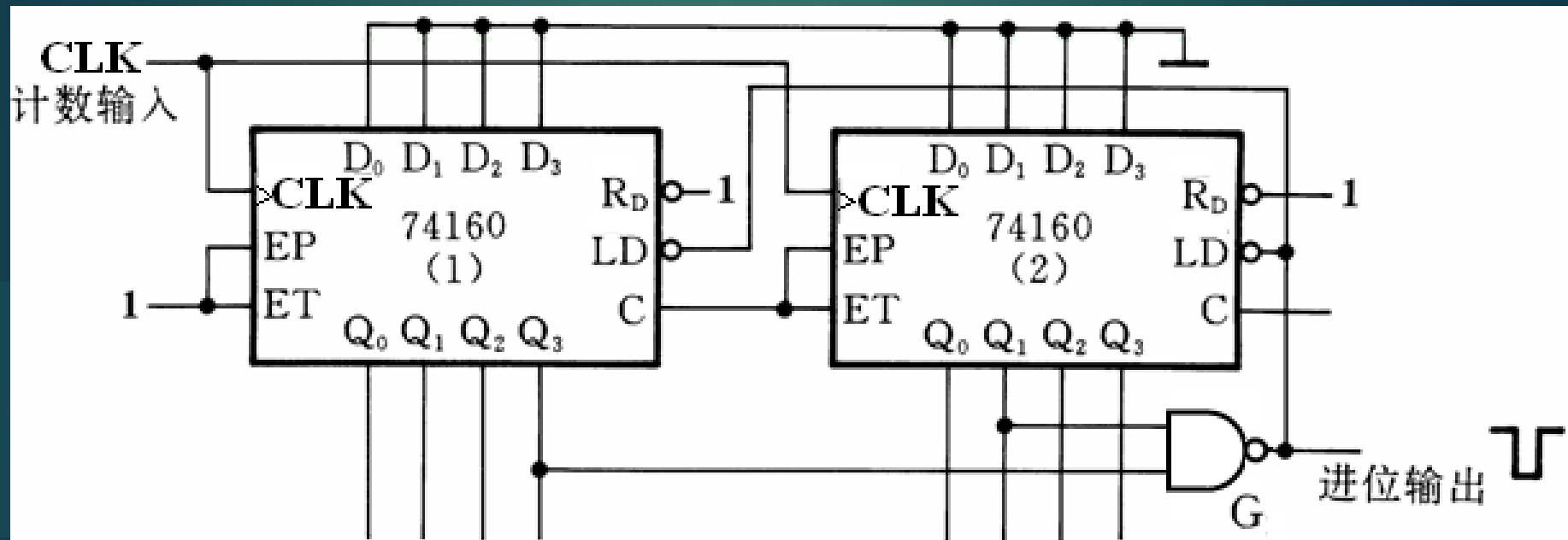


用两片74LS160接成二十九进制计数器。

解： ①整体置零方式

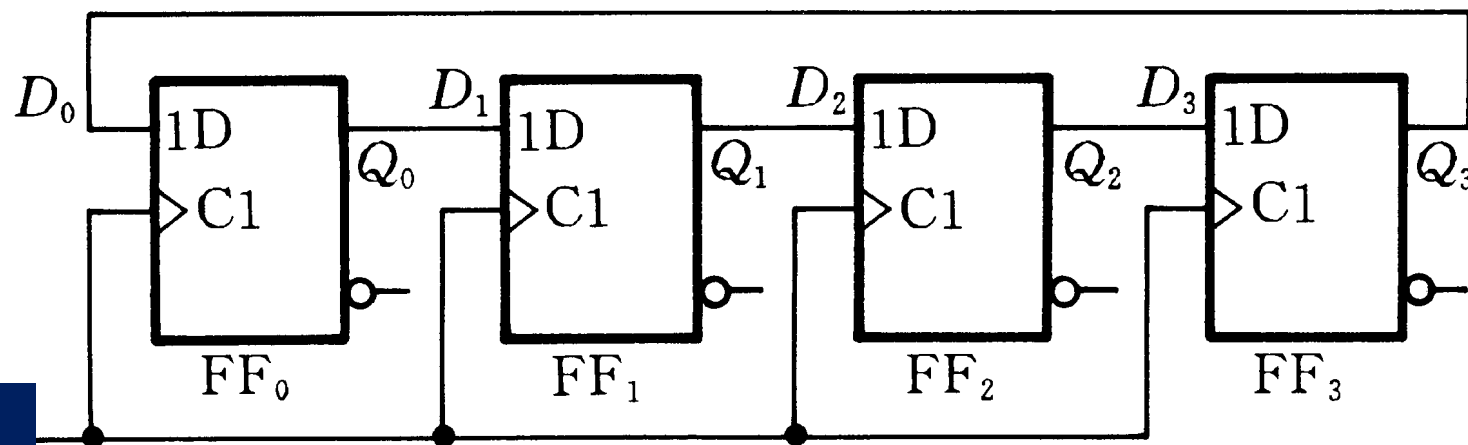


②整体置数方式



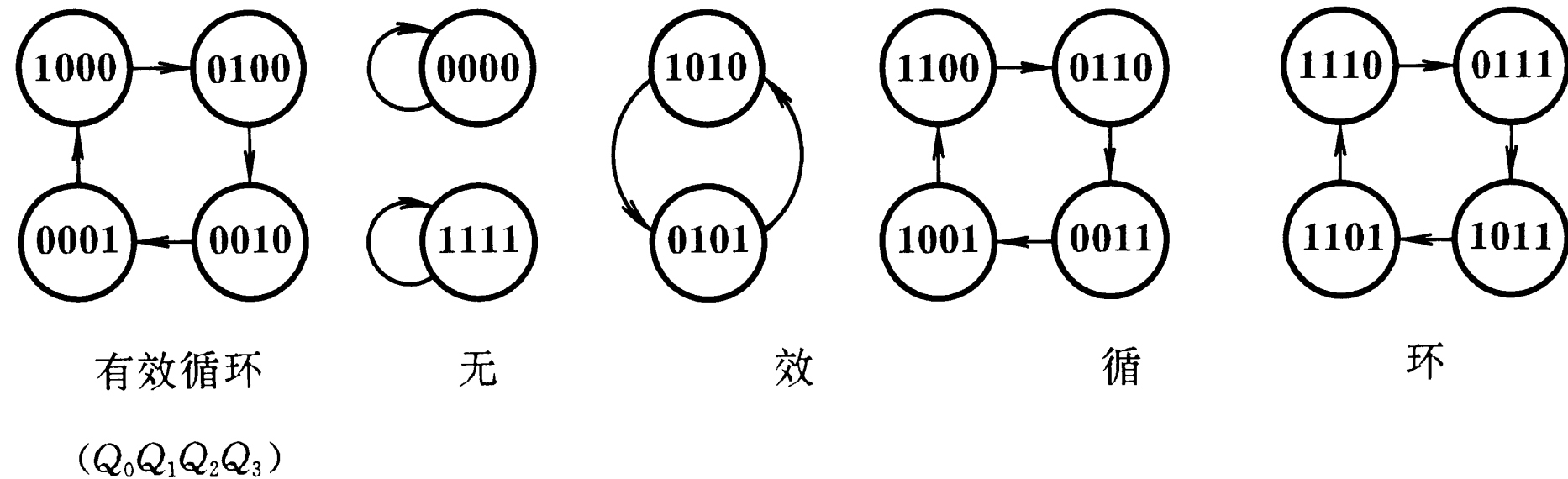
四、移位寄存器型计数器

环形计数器



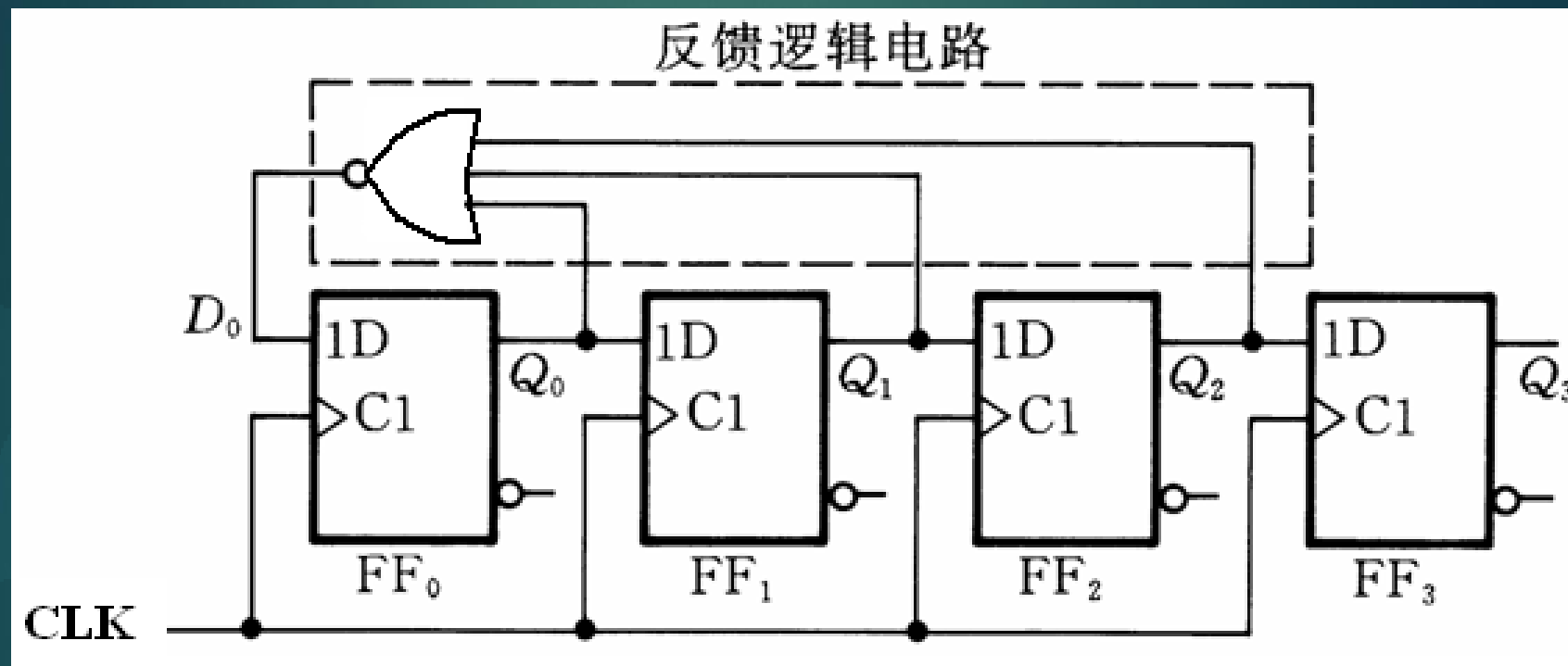
结构特点: $D_0 = Q_3$

状态转换图:

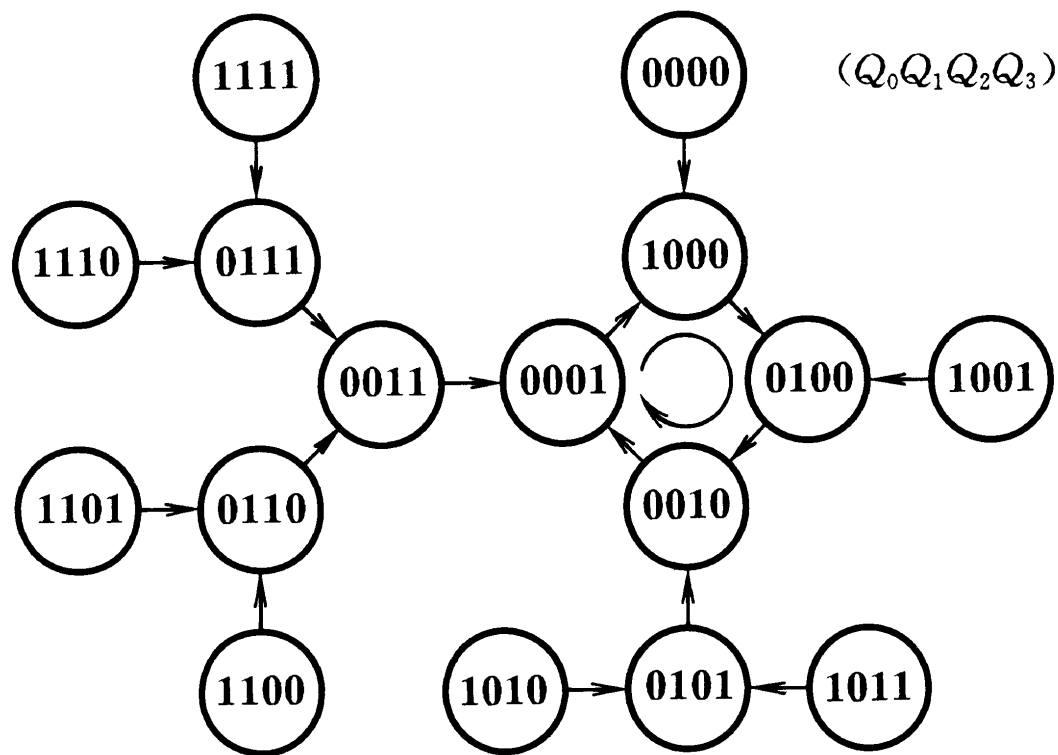


构成四进制计数器,不能自启动.

能自启动的环形计数器:

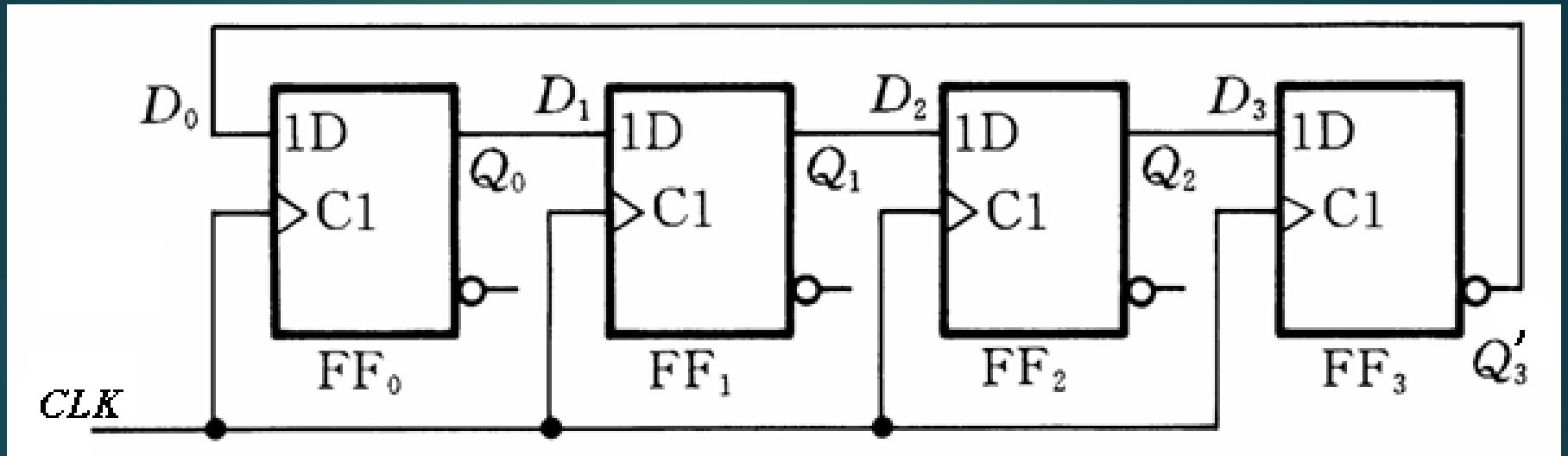


状态转换图：



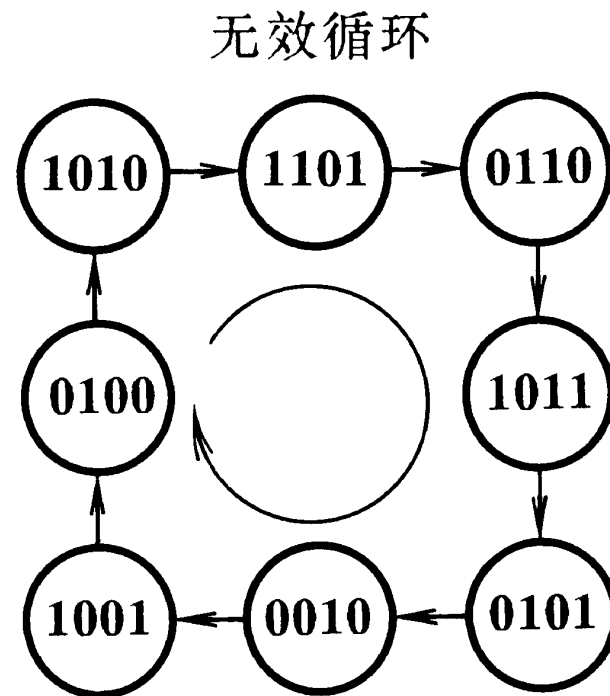
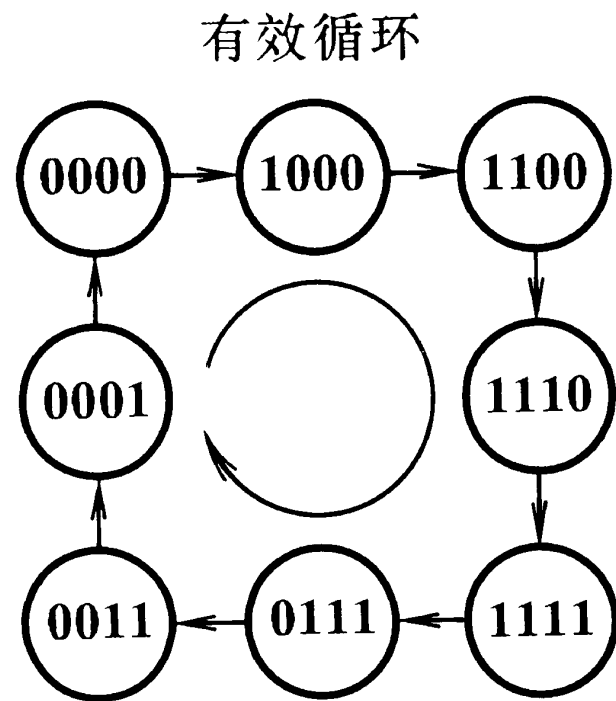
n 位移位寄存器构成的环形计数器只有 n 个有效状态，有 $2^n - n$ 个无效状态。

扭环形计数器



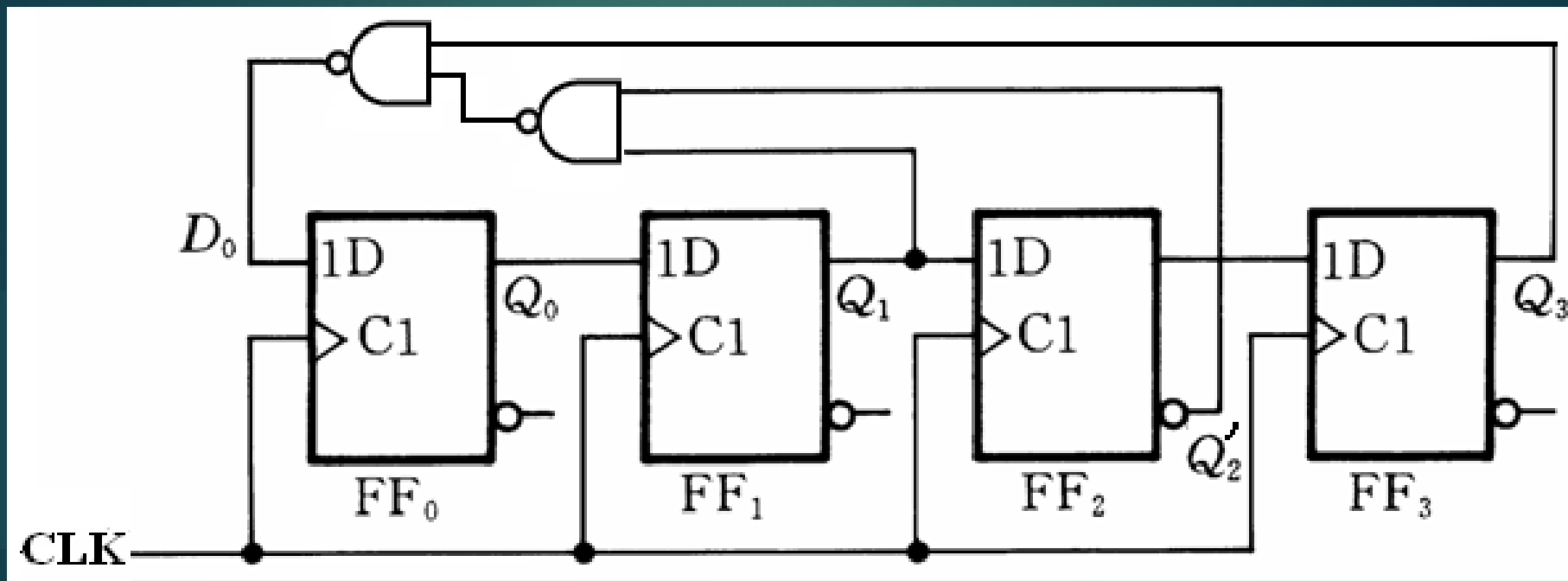
结构特点: $D_0 = Q'_{n-1}$

状态转换图：

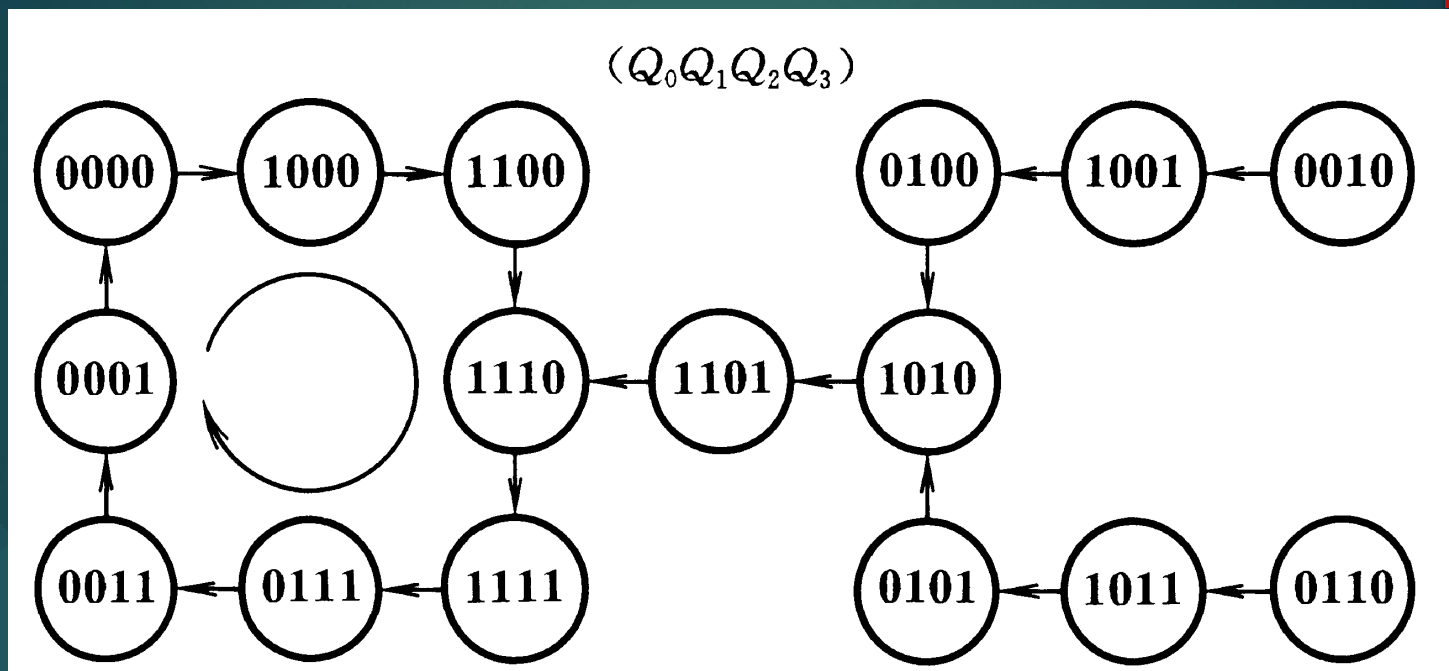


$(Q_0Q_1Q_2Q_3)$

能自启动的扭环形计数器:



状态转换图：



n 位移位寄存器构成的扭环形计数器有 $2n$ 个有效状态，有 $2^n - 2n$ 个无效状态。

时序逻辑电路的设计方法

设计步骤:



例 设计一个带有进位输出端的十三进制计数器。

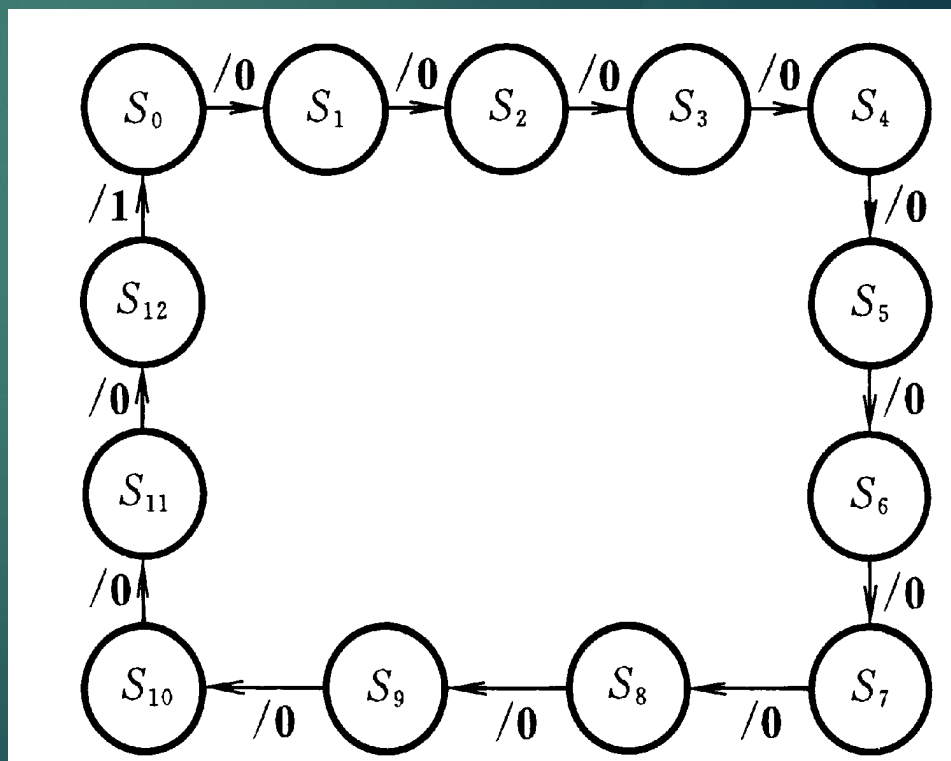
解:

1

建立原始状态图

该电路不需输入端,有进位输出用C表示,规定有进位输出时 $C=1$,无进位输出时 $C=0$ 。

十三进制计数器应该有十三个有效状态,分别用 S_0 、 S_1 、... S_{12} 表示。画出其状态转换图:



2 状态化简

状态转换图不需化简。

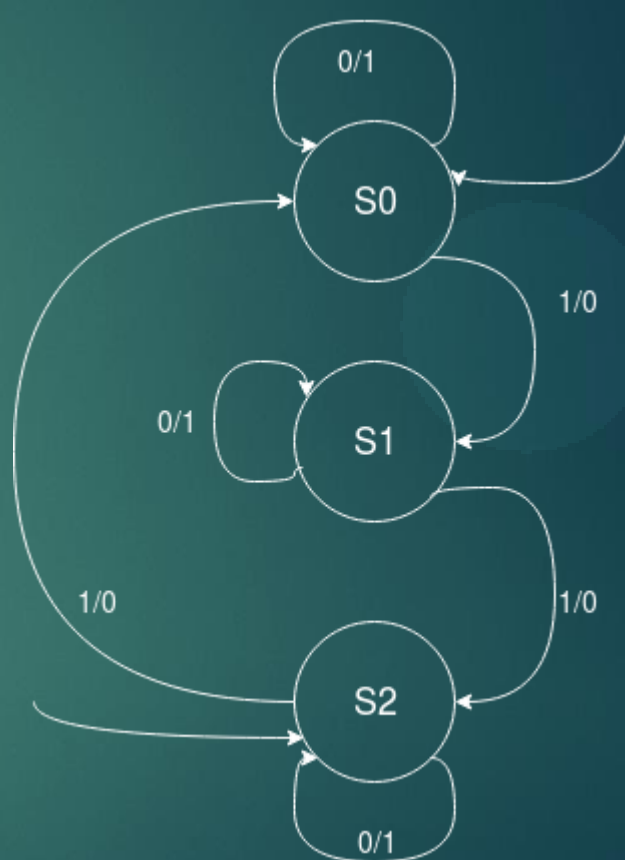
3 状态分配

因为 $2^3 < 13 < 2^4$ ，因此取触发器位数 $n=4$ 。对状态进行编码，得到状态转化表如下：

状态变化顺序	状态编码				进位输出 C
	Q_3	Q_2	Q_1	Q_0	
S_0	0	0	0	0	0
S_1	0	0	0	1	0
S_2	0	0	1	0	0
S_3	0	0	1	1	0
S_4	0	1	0	0	0
S_5	0	1	0	1	0
S_6	0	1	1	0	0
S_7	0	1	1	1	0
S_8	1	0	0	0	0
S_9	1	0	0	1	0
S_{10}	1	0	1	0	0
S_{11}	1	0	1	1	0
S_{12}	1	1	0	0	1
S_0	0	0	0	0	0

状态图化简

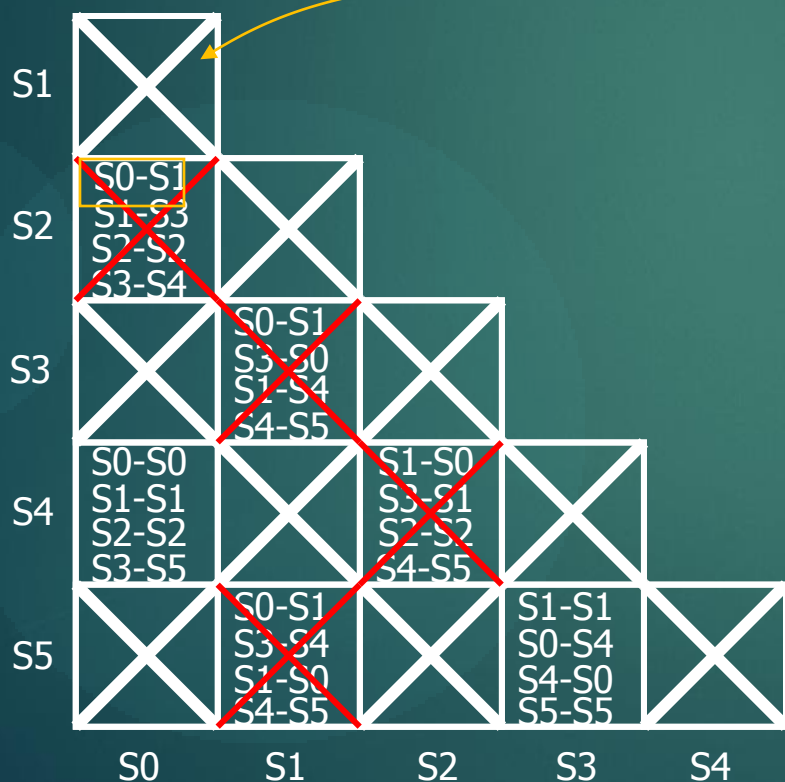
- ▶ 基于等价类划分的方法：
 - ▶ 寻找等价状态对
 - ▶ 寻找最大等价类
 - ▶ 确定所有等价类集
 - ▶ 将等价类指定为新状态



状态图化简

► Implication Chart Method 蕴涵图法

- 寻找全部等价状态对
- 首先去除掉所有输出不等价的状态对，标记输出和次态完全等价的状态对
- 列出状态对的所有次态
- 去除掉所有次态已知不等价的状态对



present state	next state				output
	00	01	10	11	
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S4	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

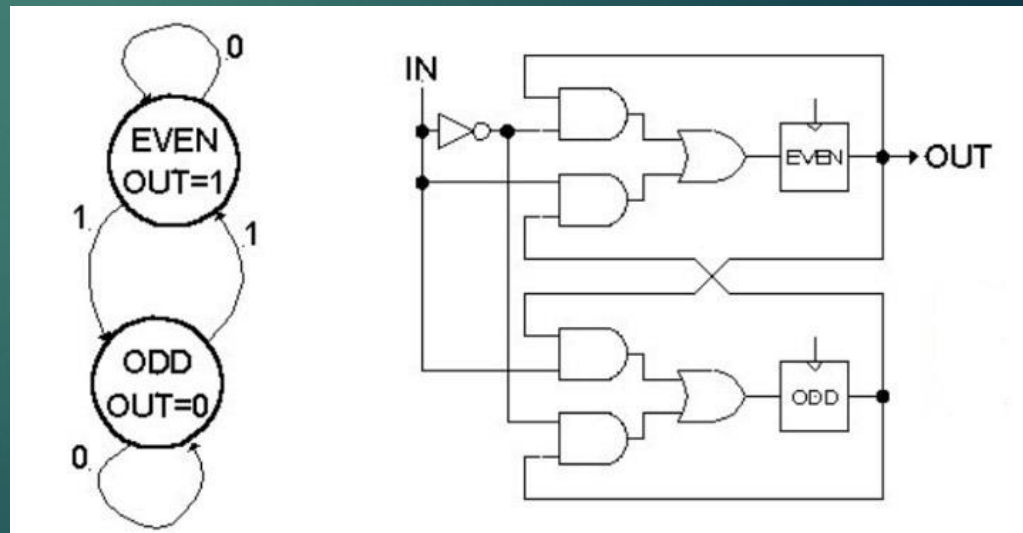
present state	next state				output
	00	01	10	11	
S0'	S0'	S1	S2	S3'	1
S1	S0'	S3'	S1	S3'	0
S2	S1	S3'	S2	S0'	1
S3'	S1	S0'	S0'	S3'	0

minimized state table
(S0==S4) (S3==S5)

状态编码

- ▶ n 位编码（触发器个数）可以表示 2^n 个状态
- ▶ 但也可以使用更多位的编码（更多触发器）用来简化逻辑
- ▶ One-Hot编码：与状态图直接对应

State	Assignment 1, Binary	Assignment 2, Gray Code	Assignment 3, One-Hot
<i>a</i>	000	000	00001
<i>b</i>	001	001	00010
<i>c</i>	010	011	00100
<i>d</i>	011	010	01000
<i>e</i>	100	110	10000



4

选触发器，求时钟、输出、状态、驱动方程

$Q_3Q_2 \backslash Q_1Q_0$					
		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	

状态变化顺序	状态编码				进位输出 C
	Q_3	Q_2	Q_1	Q_0	
S_0	0	0	0	0	0
S_1	0	0	0	1	0
S_2	0	0	1	0	0
S_3	0	0	1	1	0
S_4	0	1	0	0	0
S_5	0	1	0	1	0
S_6	0	1	1	0	0
S_7	0	1	1	1	0
S_8	1	0	0	0	0
S_9	1	0	0	1	0
S_{10}	1	0	1	0	0
S_{11}	1	0	1	1	0
S_{12}	1	1	0	0	1
S_0	0	0	0	0	0

电路次态/输出 ($Q_3^*Q_2^*Q_1^*Q_0^*/C$) 的卡诺图

状态方程:

$Q_3Q_2 \backslash Q_1Q_0$		00	01	11	10
		00	01	11	10
00	0	0	0	0	0
01	0	0	1	0	0
11	0	×	×	×	×
10	1	1	1	1	1

$$Q_3^* = Q_3 Q_2' + Q_2 Q_1 Q_0$$

$Q_3Q_2 \backslash Q_1Q_0$		00	01	11	10
		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	XXXX/X	XXXX/X	XXXX/X	
10	1001/0	1010/0	1100/0	1011/0	

z_2 z_3 z_2 z_1 z_3 z_2 z_0 z_2 z_1 z_0

$Q_3Q_2 \backslash Q_1Q_0$		00	01	11	10
		00	01	11	10
00	0	1	0	1	
01	0	1	0	1	
11	0	×	×	×	
10	0	1	0	1	

$$Q_1^* = Q_1' Q_0 + Q_1 Q_0'$$

$Q_3Q_2 \backslash Q_1Q_0$		00	01	11	10
		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	XXXX/X	XXXX/X	XXXX/X	
10	1001/0	1010/0	1100/0	1011/0	

若选用4个JK触发器，需将状态方程变换成JK触发器特性方程的标准形式，即 $Q^* = JQ' + K'Q$ ，找出驱动方程。

$$Q_3^* = Q_3 Q_2' + Q_2 Q_1 Q_0 (Q_3 + Q_3') = (Q_2 Q_1 Q_0) Q_3' + Q_2' Q_3$$

$$Q_2^* = (Q_1 Q_0) Q_2' + (Q_3' Q_1' + Q_3' Q_0') Q_2 = (Q_1 Q_0) Q_2' + Q_3' (Q_1 Q_0)' Q_2$$

$$Q_1^* = Q_0 Q_1' + Q_0' Q_1$$

$$Q_0^* = Q_3' Q_0' + Q_2' Q_0' = (Q_3 Q_2)' Q_0' + 1' Q_0$$

$$Q_3^* = Q_3 Q_2' + Q_2 Q_1 Q_0 (Q_3 + Q_3') = (Q_2 Q_1 Q_0) Q_3' + Q_2' Q_3$$

$$Q_2^* = (Q_1 Q_0) Q_2' + (Q_3' Q_1' + Q_3' Q_0') Q_2 = (Q_1 Q_0) Q_2' + Q_3' (Q_1 Q_0)' Q_2$$

$$Q_1^* = Q_0 Q_1' + Q_0' Q_1$$

$$Q_0^* = Q_3' Q_0' + Q_2' Q_0' = (Q_3 Q_2)' Q_0' + 1' Q_0$$

比较得到触发器的
驱动方程：

$$J_3 = Q_2 Q_1 Q_0$$

$$K_3 = Q_2$$

$$J_2 = Q_1 Q_0$$

$$K_2 = (Q_3' (Q_1 Q_0)')'$$

$$J_1 = Q_0$$

$$K_1 = Q_0$$

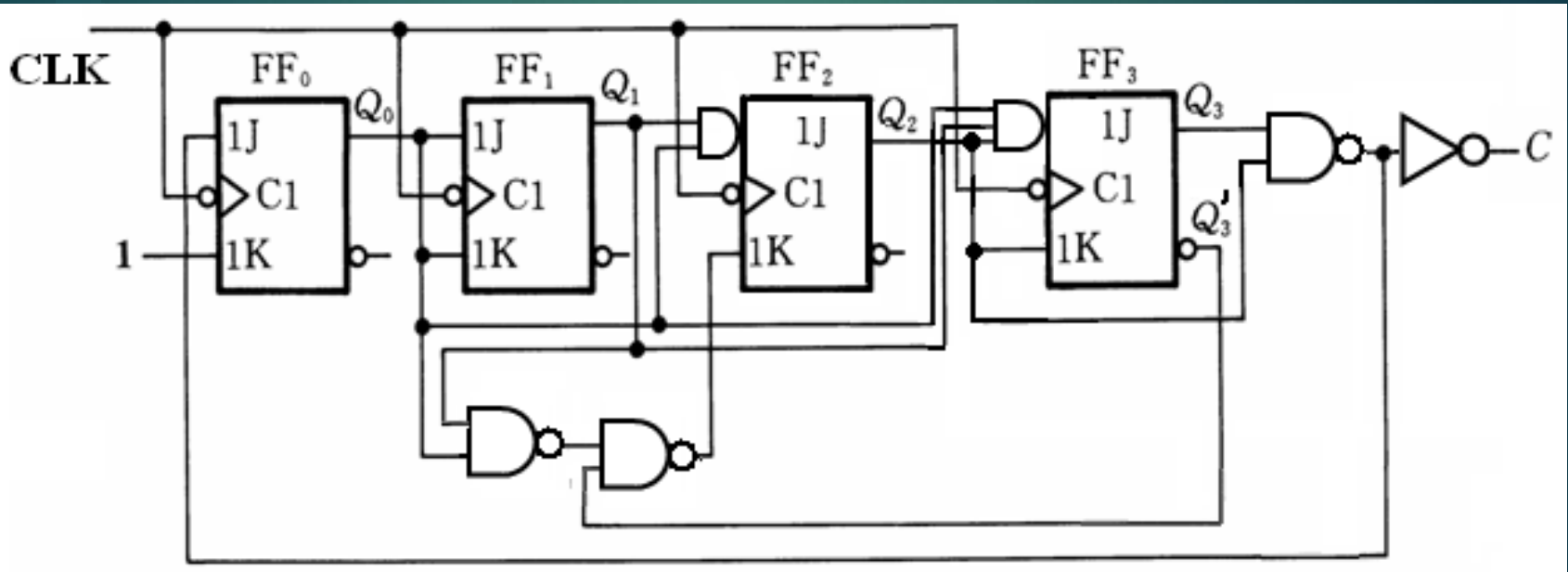
$$J_0 = (Q_3 Q_2)'$$

$$K_0 = 1$$

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

5

画电路图



6

检查电路能否自启动

将0000作为初始状态代

$$Q_3^* = (Q_2 Q_1 Q_0) Q_3' + Q_2' Q_3$$

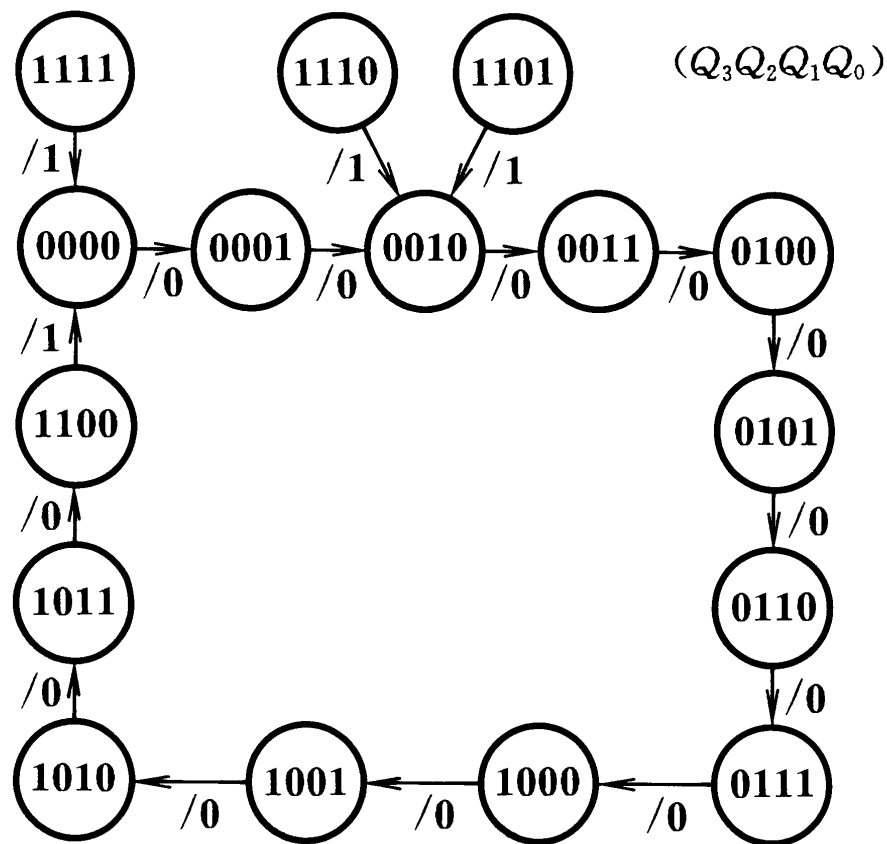
状态转换图，与状态转换表

$$Q_2^* = (Q_1 Q_0) Q_2' + Q_3' (Q_1 Q_0)' Q_2$$

否自启动。

$$Q_1^* = Q_0 Q_1' + Q_0' Q_1$$

$$Q_0^* = (Q_3 Q_2)' Q_0'$$



由状态转换图可知该电路能够自启动。

例 连续输入三个及以上1输出1，否则输出0

解：

1 建立原始状态图

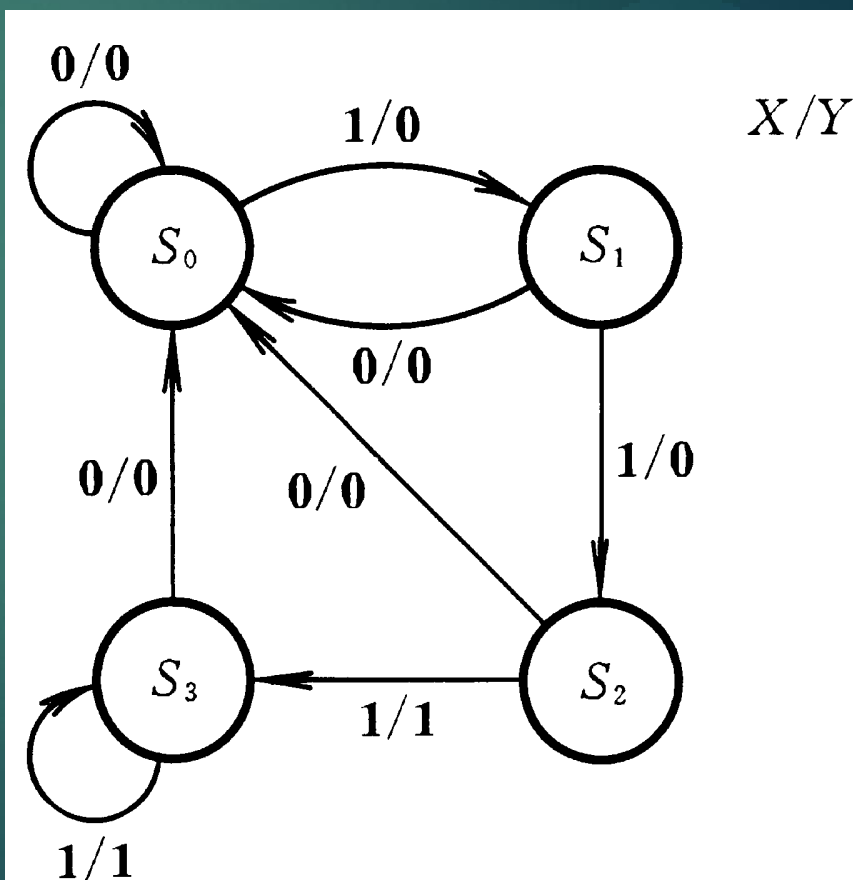
输入数据作为输入变量，用X表示；检测结果为输出变量，用Y表示。例如：

输入X 101100111011110

输入Y 000000001000110

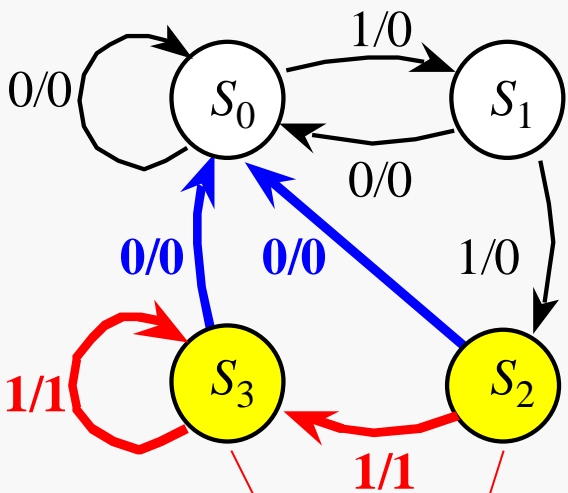
设电路没有输入1以前的状态为 S_0 ，输入一个1状态为 S_1 ，连续输入两个1后的状态为 S_2 ，连续输入3个1以后的状态为 S_3 。

画状态转换图

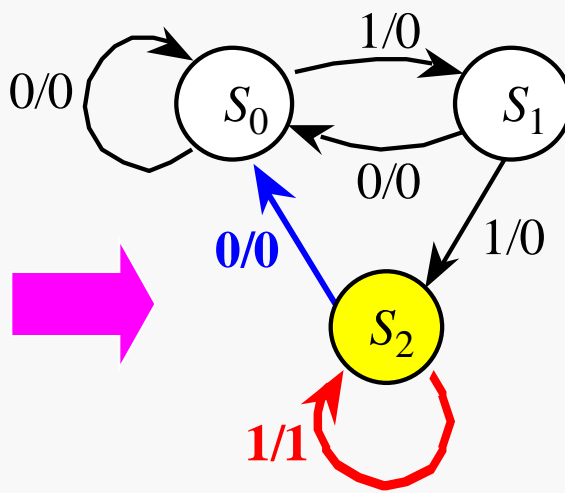


2 状态化简

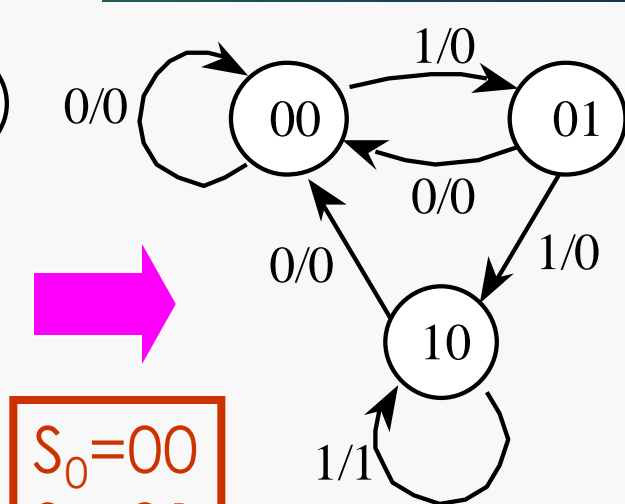
3 状态分配



(a) 原始状态图



(b) 简化状态图



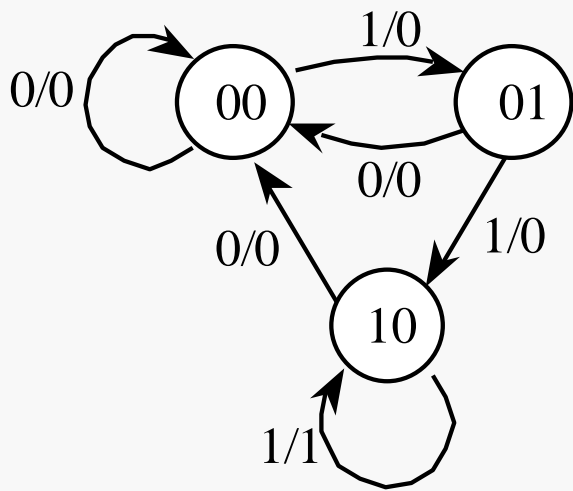
(c) 二进制状态图

$s_0=00$
 $s_1=01$
 $s_2=10$

两个状态等价

4

选触发器，求时钟、输出、状态、驱动方程



(c) 二进制状态图

X	Q_1	Q_0	Q_1^*	Q_0^*	Y
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	×	×	×
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	×	×	×

X	$Q_1 Q_0$			
	00	01	11	10
0	00/0	00/0	××/×	00/0
1	01/0	10/0	××/×	10/1

 $(Q_1^* Q_0^* / Y)$

卡诺图

将卡诺图分解,求状态方程和输出方程, 并得到驱动方程

$X \backslash Q_1 Q_0$		00	01	11	10
		0	0	×	0
1	0	0	1	×	1

(a) Q_1^*

$X \backslash Q_1 Q_0$		00	01	11	10
		0	0	×	0
0	1	0	0	×	0
1	1	1	0	×	0

(b) Q_0^*

$X \backslash Q_1 Q_0$		00	01	11	10
		0	0	×	0
1	0	0	0	×	1

(c) Y

$X \backslash Q_1 Q_0$		00	01	11	10
		00/0	00/0	× × / ×	00/0
0					
1		01/0	10/0	× × / ×	10/1

$$\begin{cases} J_1 = XQ_0 \\ K_1 = X' \end{cases}$$

$$\begin{cases} J_0 = XQ_1' \\ K_0 = 1 \end{cases}$$

输出方程: $Y = XQ_1$

5

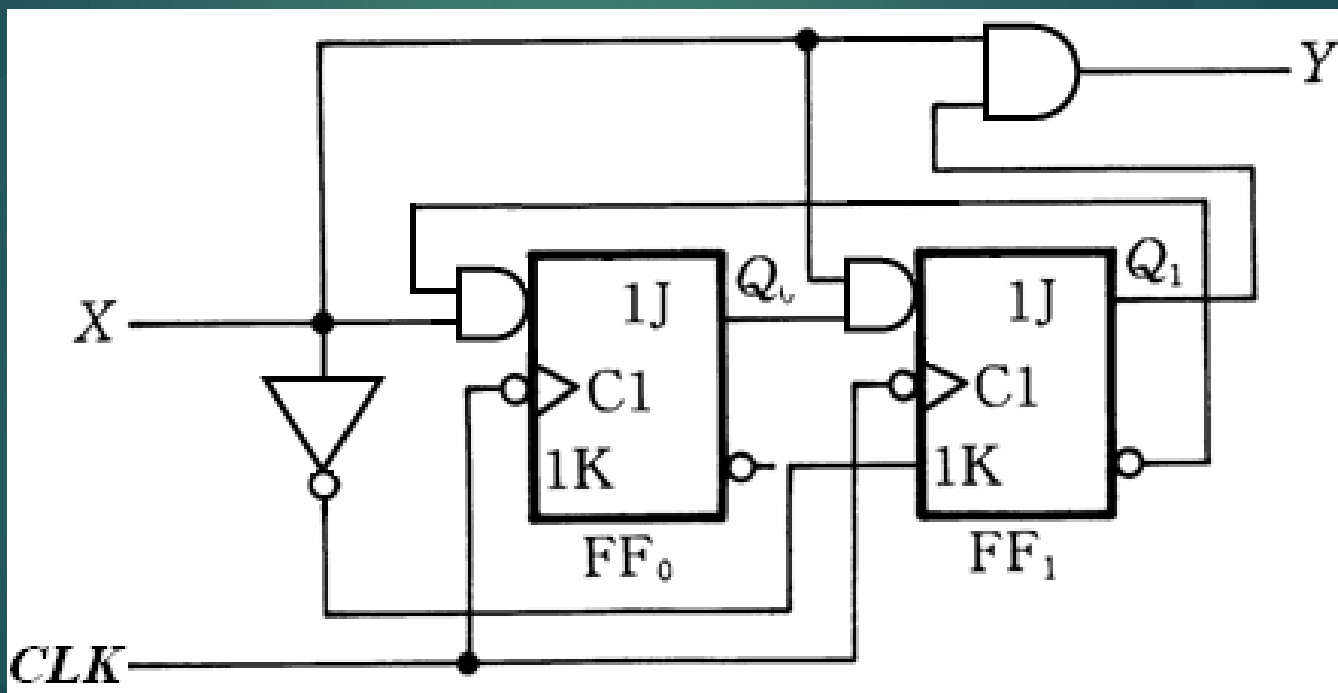
画电路图

$$\begin{cases} J_1 = XQ_0 \\ K_1 = X' \end{cases}$$

$$\begin{cases} J_0 = XQ_1' \\ K_0 = 1 \end{cases}$$

输出方程:

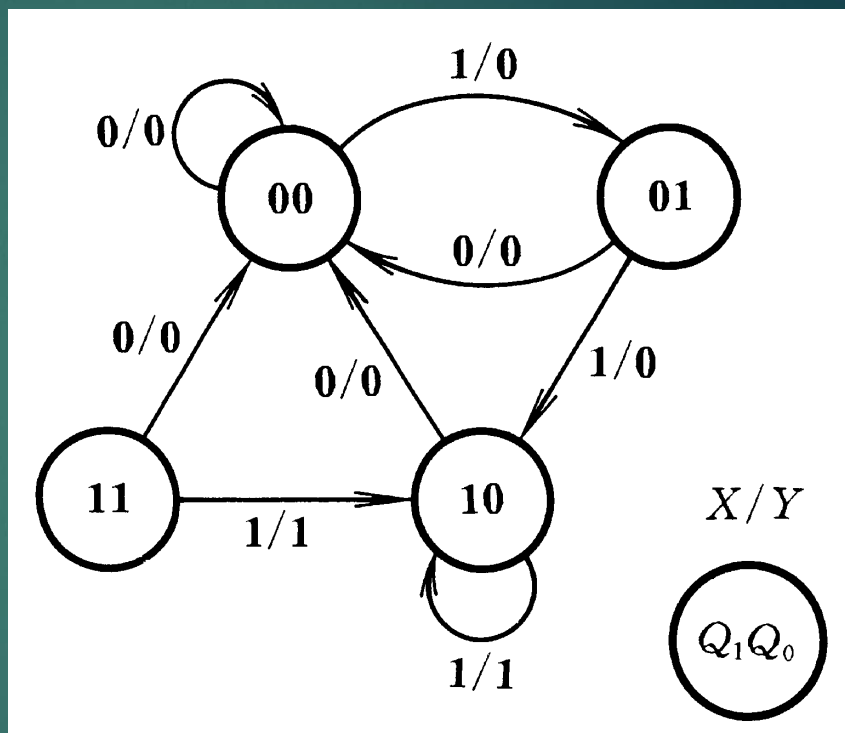
$$Y = XQ_1$$



6

检查电路能否自启动

$$\begin{cases} Q_1^* = XQ_0 + XQ_1 \\ Q_0^* = XQ_1' Q_0' \end{cases}$$



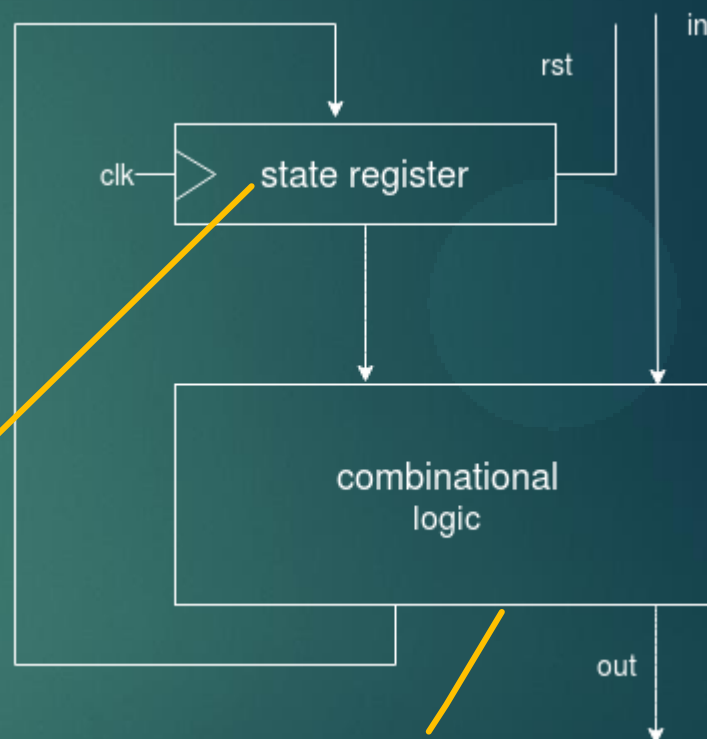
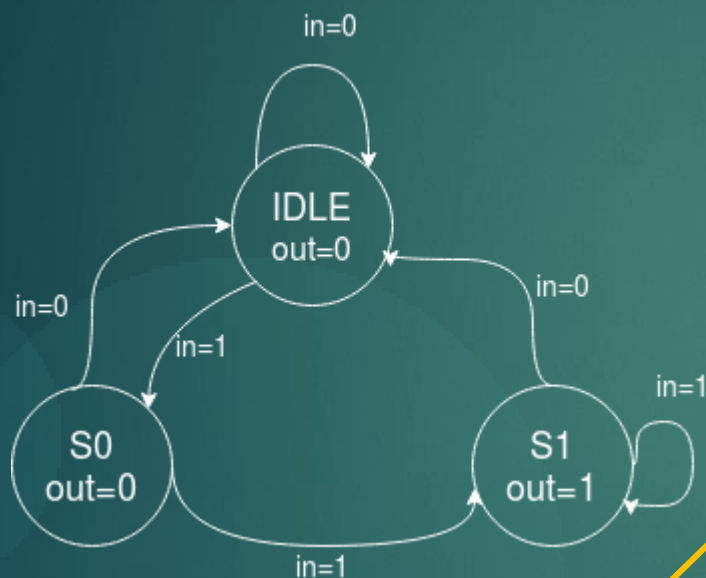
由状态转换图可知该电路能够自启动.

时序逻辑电路的设计方法

设计步骤:



使用Verilog实现时序逻辑状态机

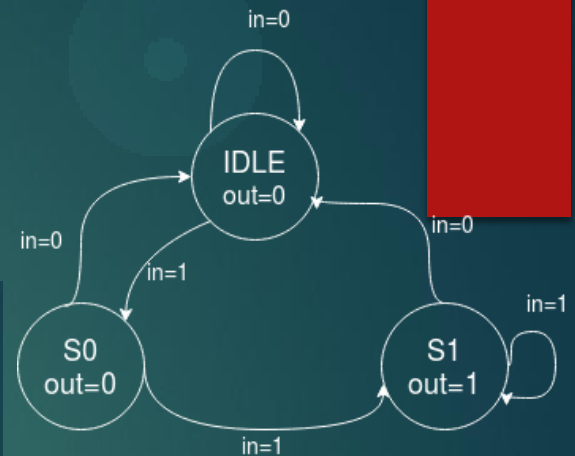


通过寄存器变量保存状态机所处状态（现态）

用于根据现态和输入计算次态和输出的组合逻辑

状态机的Verilog描述

```
module FSM1(output reg out,  
            input in, clk, rst);  
  
    // 状态编码  
    parameter IDLE = 2'b00;  
    parameter S0 = 2'b01;  
    parameter S1 = 2'b10;  
    reg [1:0] present_state, next_state;  
  
    // 状态转移的 always 块  
    always @(posedge clk)  
        if(rst) present_state <= IDLE;  
        else present_state <= next_state;
```



必须有，以便于初始化系统状态

用于定义状态编码的
模块参数

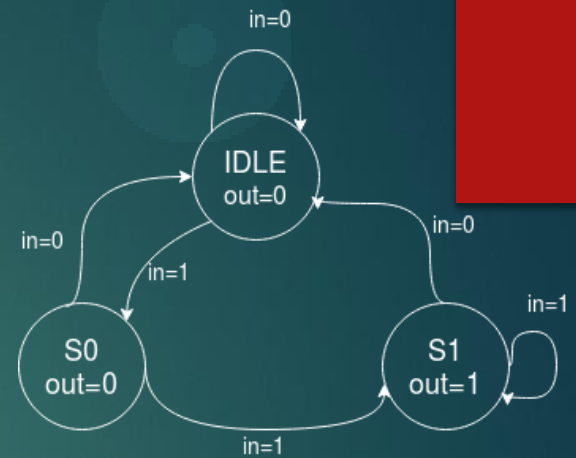
现态

次态，组合
逻辑的输出

使用另外一个always块来描述次态计算的组合逻辑。这两个always块是并行执行的。

状态机的Verilog描述

```
// 组合逻辑的always块
always @(present_state, in)
case (present_state)
// 对每一个现态定义不同输入下的次态和输出
IDLE : begin out = 1'b0;
        if (in == 1'b1) next_state = S0;
        else next_state = IDLE;
    end
S0 : begin out = 1'b0;
        if (in == 1'b1) next_state = S1;
        else next_state = IDLE;
    end
S1 : begin out = 1'b1;
        if (in == 1'b1) next_state = S1;
        else next_state = IDLE;
    end
default : begin
        out = 1'b0;
        next_state = IDLE;
    end
endcase
endmodule
```



每个状态，对应一个case分支

对每个现态，给出其在不同输入下的输出和次态

使用default语句来处理不完备（不关心）的状态转移，一般转向重置后状态

同步计数器的Verilog描述

```
module Binary_Counter_4_Par_Load (  
    output reg [3: 0] A_count, // 输出和  
    output C_out, // 输出进位  
    input [3: 0] Data_in, // 数据输入  
    input Count, // 计数  
        Load, // 置数  
        CLK, // 时钟  
    Clear_b // 清零  
);  
assign C_out = Count && (~Load) && (A_count == 4'b1111);  
always @ ( posedge CLK, negedge Clear_b)  
    if (~Clear_b) A_count <= 4'b0000;  
    else if (Load) A_count <= Data_in;  
    else if (Count) A_count <= A_count + 1'b1;  
    else A_count <= A_count;  
endmodule
```


移位寄存器的Verilog描述

```
module Shift_Register_4_beh (
    output reg [3: 0] A_par, // 寄存器输出
    input [3: 0] I_par, // 寄存器输入
    input s1, s0, // 控制信号
    MSB_in, LSB_in, // 串行输入
    CLK, Clear_b
);
always @ (posedge CLK, negedge Clear_b)
    if (Clear_b == 0) A_par <= 4'b0000;
    else
        case ({s1, s0})
            2'b00: A_par <= A_par; // 不移位
            2'b01: A_par <= {MSB_in, A_par[3: 1]}; // 右移
            2'b10: A_par <= {A_par[2: 0], LSB_in}; // 左移
            2'b11: A_par <= I_par; // 并行置数
        endcase
endmodule
```