数字逻辑电路 & 实验

经*12-计18* 张诗颖 *2021011056*

逻辑代数基础

1. **数制**

   一个进位计数包含两个基本因素：基数、位权

   - 基数：数制中数码的个数，如基数为R的数制（R进制）包含0~$R-1$的数码
   - 位权：R进制中处于不同位数的数码代表的权值

   R进制数N的表示方法（其中K_i表示不同数位的数码）

   - 并列表示：$(N)_R=(K_{n-1}K_{n-2}\dots K_1K_0K_{-1}K_{-2}\dots K_{-m})_R$
   - 多项式标识方式：$(N)_R=(\sum_{i=-m}^{n-1}K_i\times R^i)_R$

   常用的数制转换

   - $2^{15}=32,768;\ \ 2^{16}=65,536;\ \ 2^{31}=2,147,483,648$

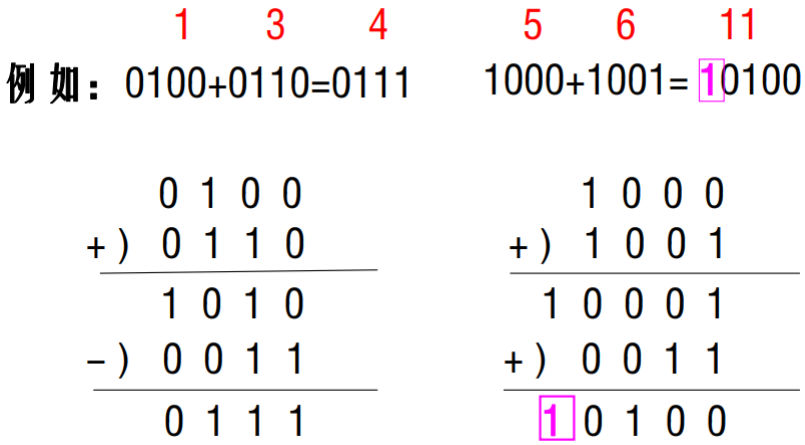
2. **码制**

   - 十进制数的二进制编码

     8421码（BCD码 / binary-code-decimal）：有权码

     5421码、2421码

     余3码（无权码）：8421BCD码+0011，当两个十进制数的和是9时，相应的余3码的和正好是15，可以自动产生进位信号；但对其它"和"需要进行修正（如两个余3码相加没有进位，则和数要减3，否则和数要加3）

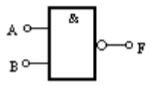
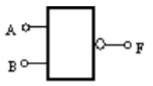
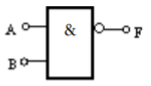
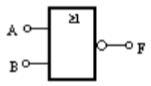
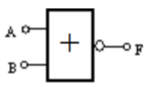
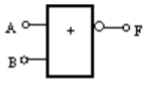
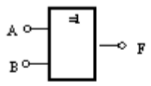
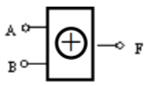
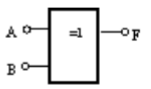
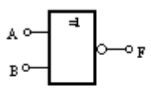
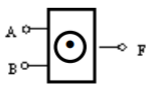
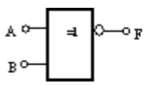
   - 格雷码（无权码）：任何相邻的十进制数的格雷码仅有一位不同

十进制数	8421码	格雷码1	格雷码2	典型格雷码	修改格雷码
0	0000	0000	0000	0000	0010
1	0001	0001	0001	0001	0110
2	0010	0011	0011	0011	0111
3	0011	0010	0010	0010	0101
4	0100	0110	0110	0110	0100
5	0101	1110	0111	0111	1100
6	0110	1010	0101	0101	1101
7	0111	1011	0100	0100	1111
8	1000	1001	1100	1100	1110
9	1001	1000	1000	1101	1010

典型格雷码编码规则：$G_i = B_{i+1} \oplus B_i$

- 七位ASCII编码：高三位区分，低四位编码

3. **常用逻辑门**

符号名称	国标符号	常用符号	国际流行符号	IEEE逻辑符号
与非门				
或非门				
异或门				
同或门				

4. **逻辑代数化简**

- 最小项表示

应用举例：n=3时8个最小项为（C为高位，A低位）

$$m_0 = \overline{A}\,\overline{B}\,\overline{C} \quad m_1 = A\overline{B}\,\overline{C} \quad m_2 = \overline{A}B\overline{C} \quad m_3 = AB\overline{C}$$

$$m_4 = \overline{A}\,\overline{B}C \quad m_5 = A\overline{B}C \quad m_6 = \overline{A}BC \quad m_7 = ABC$$

- 最大项表示

例如：n=3的最大项为

$$M_0 = A + B + C \quad M_1 = \overline{A} + B + C$$

$$M_2 = A + \overline{B} + C \quad M_3 = \overline{A} + \overline{B} + C$$

$$M_4 = A + B + \overline{C} \quad M_5 = \overline{A} + B + \overline{C}$$

$$M_6 = A + \overline{B} + \overline{C} \quad M_7 = \overline{A} + \overline{B} + \overline{C}$$

- 最大项与最小项的关系

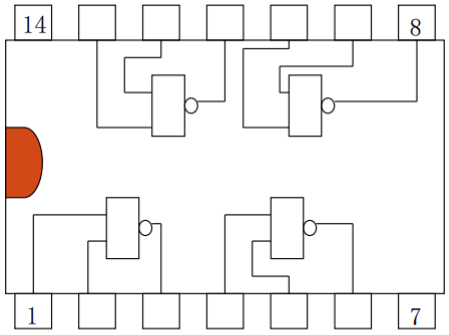
对于所有的i，相同i的最大项与最小项**互补**：$M_i=\overline{m_i}$，$m_i=\overline{M_i}$
对于所有的i，m_i和M_i互为**反函数**（反演规则下，函数F的反函数记作$\overline{F}$）
对于所有的i，m_i和M_{2^n-1-i}互为**对偶式**（对偶规则：当某个逻辑恒等式成立时，则该恒等式两侧的对偶式也相等）；F的对偶式记作F'
——或与表达式和与或表达式的转换：求两次对偶（注意！求对偶式的时候变量不用取反！）

- 逻辑代数化简、卡诺图化简、Q-M表格法化简
  吸收律：$A+\overline{A}B=A+B$
  包含律：$AB+\overline{A}C+BC=AB+\overline{A}C$，
  $(A+B)(\overline{A}+C)(B+C)=(A+B)(\overline{A}+C)$

1 组合逻辑电路 - 与非门

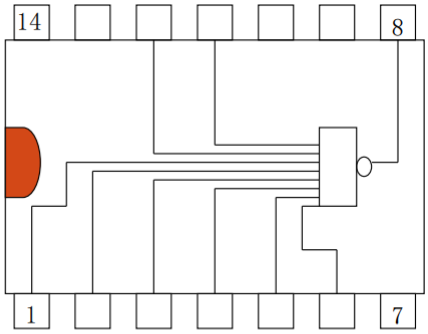
1. 实际的与非门器件

> 74LS00、74LS30（芯片 / 小规模集成电路）
> `54`开头的表示军用，$-45℃$~$105℃$，`74`开头的表示民用，$0℃$~$45℃$
> `LS`表示`TTL`
> `74LS138`：三个使能端的3-8译码器



74LS00

2输入4与非门



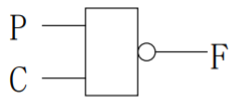
74LS30

8输入与非门

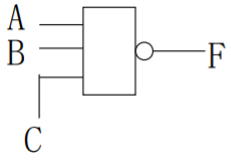
> DIP管脚：18个管脚
> PGA（`74163`）：16个管脚
> 其它：BGA, PLCC, QFP

2. 与非门的应用

1. 与非门实现**封锁电路**（由C控制）

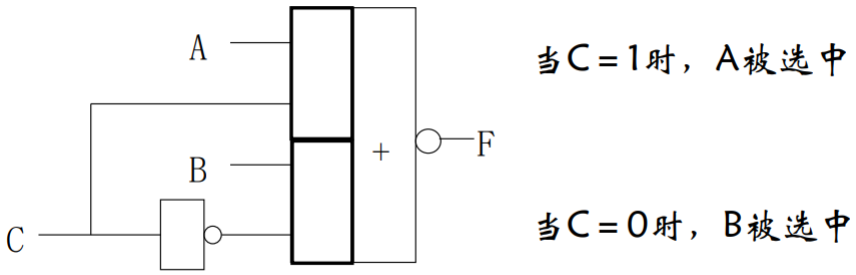


$$\begin{cases} C=1 & F=\overline{P} \\ C=0 & F=1 \end{cases}$$



$$\begin{cases} C=1 & F=\overline{AB} \\ C=0 & F=1 \end{cases}$$

2. 与非门实现**数据选择**：$AC+B\overline{C}$



3. 与非门的逻辑表示、门电路概述

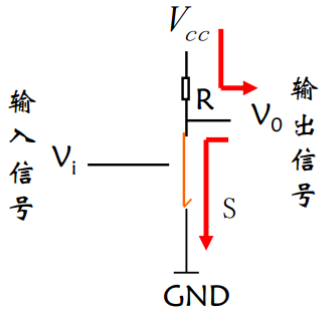
1. **正逻辑**：$H=1, L=0$，即高电平表示1，低电平表示0
2. 两大类**工艺技术**：**TTL (Transistor-Transistor/晶体管 Logic)**（速度快、功耗大、集成度低），**MOS（金属氧化物半导体）**（速度慢，工号小，集成度高）
   目前常用器件的工艺：**CMOS**（互补金属氧化物半导体）
3. **集成电路**发展：SSI（小规模IC），MSI（中规模IC），LSI（大规模IC），VLSI（超大规模IC），ULSI（超大规模IC）

1.1 晶体管的工作状态

1. 二极管

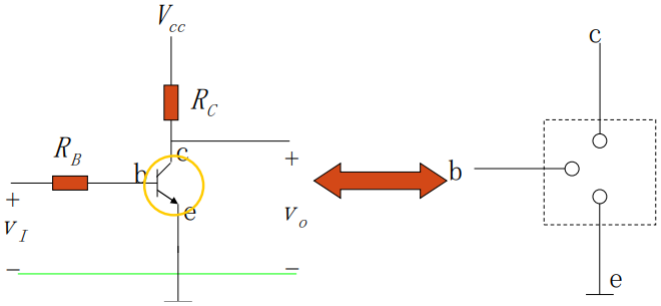
1. 最简单的二值逻辑：开关

> $V_i=0$：电键闭合，输出高电平（H）
> $V_i=1$：电键打开，输出低电平（L）



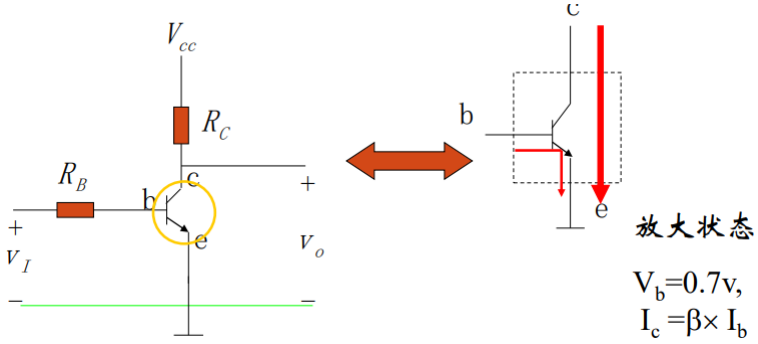
2. 截止状态

> 条件：$V_b<0.7V$
> 结果：$I_b=0, Ic=0, V_o=V_cc$
> 相当于断路



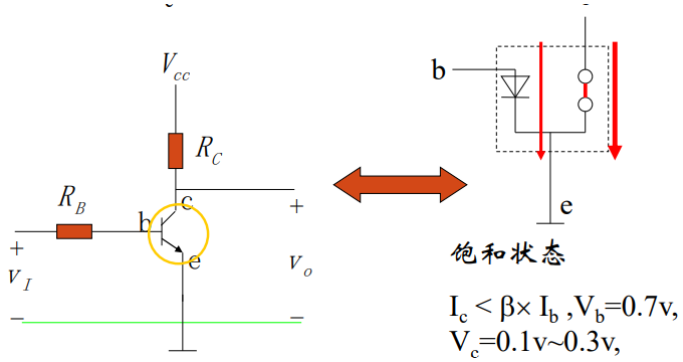
3. 放大状态

> 条件：$V_b = 0.7V$，I_b较小，I_c足够大
> 结果：$I_c = \beta \times I_b$，放大系数β：20~100



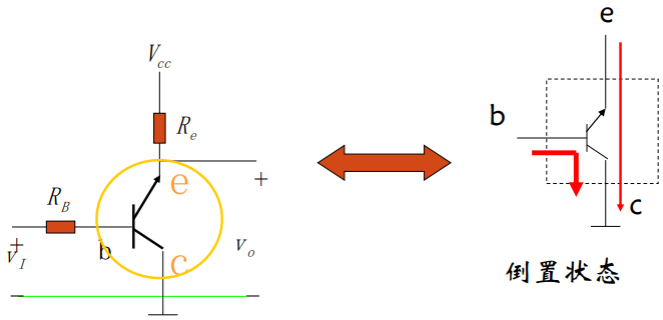
4. 饱和状态

> 条件：$V_b = 0.7V$，I_b较大，$I_c < \beta \times I_b$
> 结果：$V_c = 0.1V$~$0.3V$



5. 倒置状态

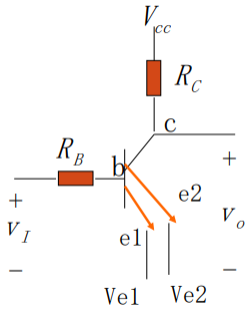
> 条件：$V_b = 0.7V$，$V_e > V_c$
> 结果：$I_e = \beta' \times I_b$，其中β≈0.5



2. 多发射极三极管

高电压：$3.6V$~$5V$
低电压：$0.3V$以下

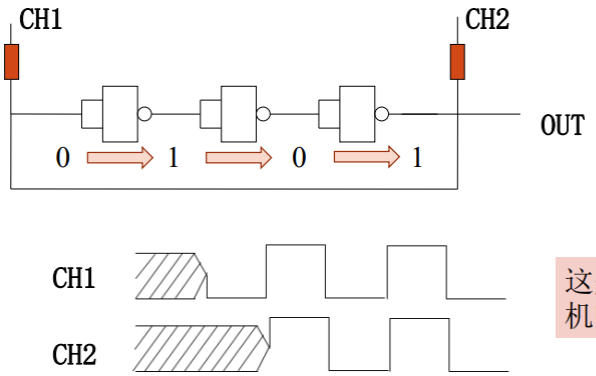
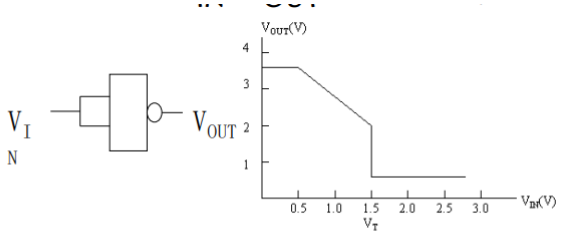
> 双发射极只要有一个为**低**，当$V_1 - V_{e1} > 0.7V$，整个三极管出于导通状态
> 若两级V_{e1}和V_{e2}都很高，则三极管处于截止状态



1.2 典型五管TTL“与非门”

主要技术参数:

1. **扇入**（Fan-in）：一个门的可用输入数目
2. **扇出**（Fan-out）：一个门的输出可以驱动的标准门个数
3. **传输延迟**（Propagation Delay）：从输入传输到输出所需要的时间（以高低中间点为参考点）
4. **功耗**（Power Dissipation）：逻辑门消耗的能量

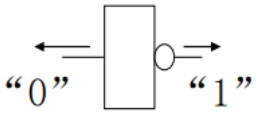
> - **自激振荡电路**
>
> 
>
>
> 这是永动机吗？
>
> - **转移特性**：门电路中输出电压随输入电压的变化特性（$V_{IN}-V_{OUT}$关系曲线）
>
> 
>
>
> 在曲线上，V_{OUT}急剧下降时的V_{IN}称：阈值电压V_T，或称门槛电压

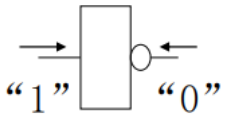
直流参数:

“0”输入电流 $I_{IL}<=1.6mA$
“1”输出电流 $I_{oH}<=0.4mA=400uA$
“1”输出电压 V_{oh} >=3^V (10个负载)

“1”输入电流 $I_{IH}<=40uA$
“0”输出电流 $I_{oL}<=16mA$
“0”输出电压 $V_{oL}<=0.35^V$ (10个负载)

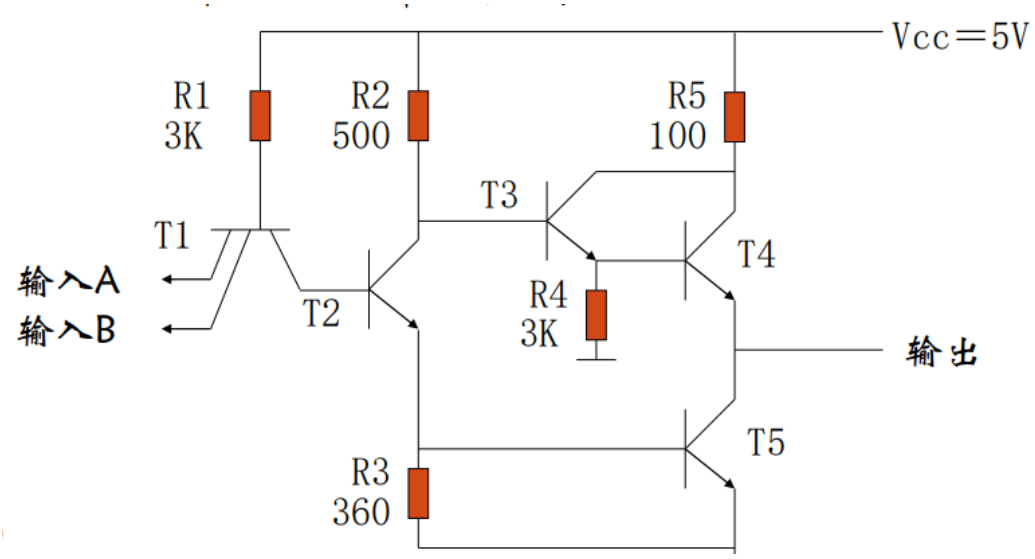
> 下图所示的箭头表示电流/电压方向



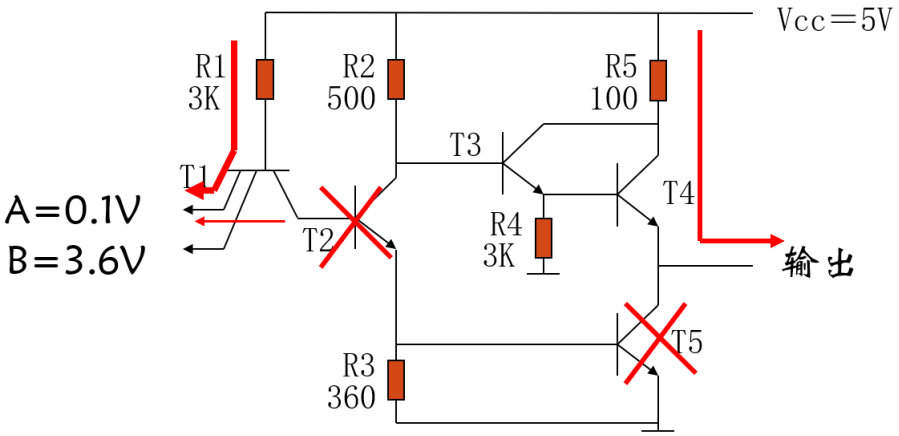


门电路级联的上限：负载能力Max = **10**
负载大于与非门承受能力时，低电平变高，高电平变低，会导致整个逻辑电路不能工作

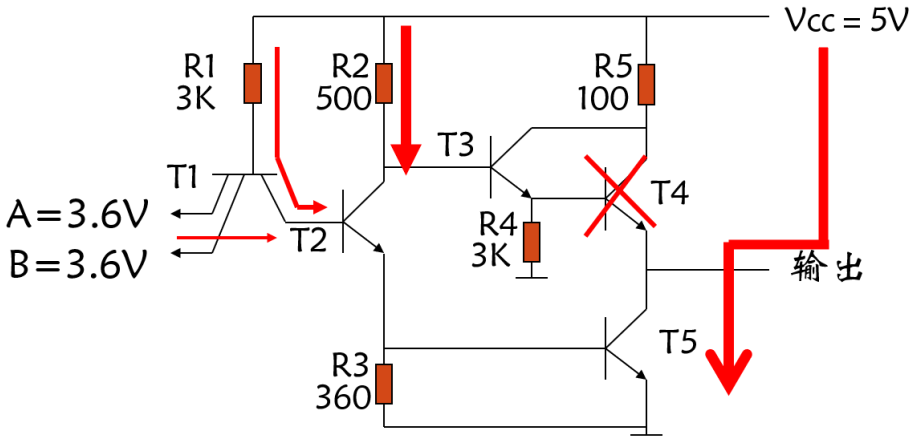
1. 图腾柱（Totem）结构



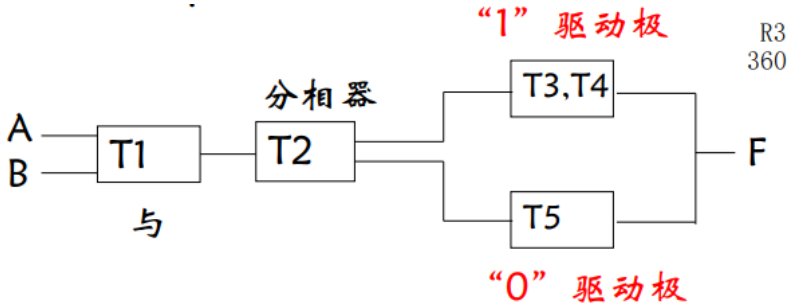
- **输入有低**：此时，T_1深饱和，T_2和T_5截止，T_3和T_4导通，输出电压约为$3.6V$



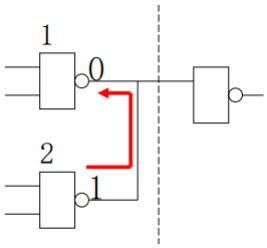
- **输入均为高**：此时，T_1倒置，T_2饱和，T_3导通，T_4截止，T_5饱和，输出电压约为$0.3V$



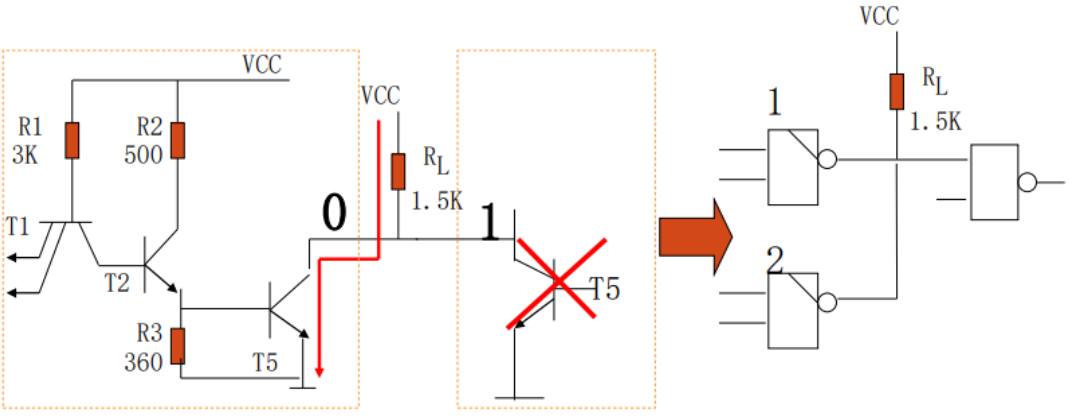
这种典型的与非门结构称为**图腾柱结构（Totem）**



2. 集电极开路（OC）结构

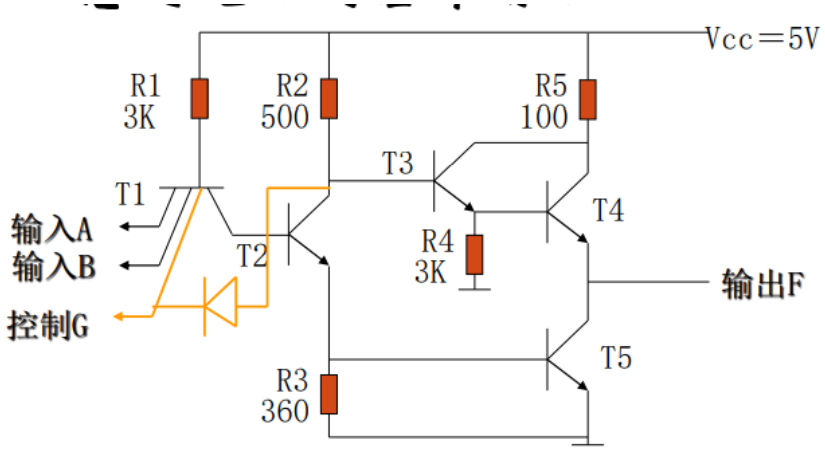
> 普通与非门输出实现“线与”时的问题：
>
> 图腾输出结构的电路，是不能把它们的输出线与在一起的。否则，当一门电路的输出为“H”，另一为 “L”时，有大电流从“H”端流向“L”端， 电流太大，会烧坏与非门
>
> 
>
>
> **线与**：如果把驱动电路*A*、*B*、*C*……的输出直接挂向总线，要求当某一驱动器向总线 发送数据*D* 时，其余驱动器输出均为 *“1”*。这样，总线状态为各驱动器输出状态之“与”，即*D·1·1·……=D*， 把这种与连接称为“线与”（*Wired AND*）

OC电路线与：



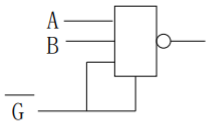
特点：上升延迟很大（T_5退饱和很慢），适合速度较慢的电路（对于速度要求较快如CPU数据总线）不能使用OC门；同时，OC门也不能和普通与非门线与

3. 三态门（Tri-State Circuit）



- 当控制G=1时，电路是一个图腾结构的与非门
- 当G=0，T3、T4、T5均截止，与非门输出F=Z（高阻态）

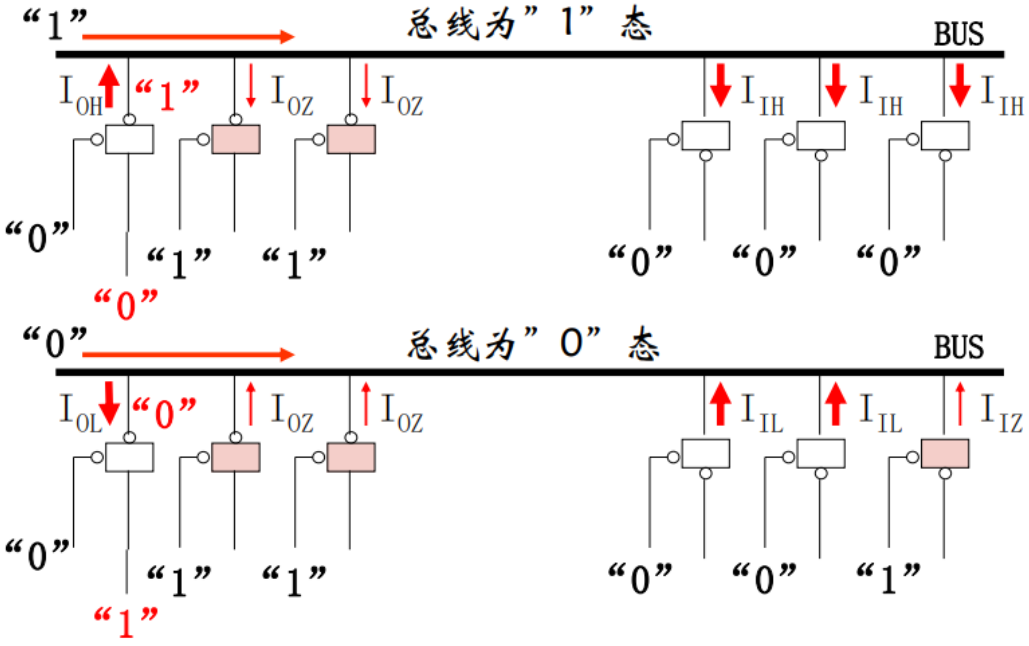
功能表：（其中，**高阻态**相当于隔断状态（电阻很大，可以理解为开路）
注意：下面这张图的$\overline{G}$和普遍认知的有一个反的关系！



功能表

A	B	$\overline{G}$	F	
X	X	0	Z	高阻态
0	0	1	1	正常态
1	0	1	1	
0	1	1	1	
1	1	1	0	

> 两个三态门和总线“线与”的时候，**只能有一个处于正常态**（另一个需要处于高阻态）
>
> OH, IH, OL, IZ依次表示：output high, input high, output low, output 高阻态
> $ouput$：向总线输出，接电输入电平
> $input$：从总线输入，由此读数据
>
> 下图中红色箭头表示**电流流向**而不是**数据流向**，数据永远从输入流向输出
>
> 控制端的**电流流向**：指向“0”



2 常用中规模组合逻辑电路

译码器的分类：

- 变量译码器：用来表示输入变量状态的全部组合，一般为N位输入，2^N位输出
- 码制译码器：如8421码变换为循环码等
- 显示译码器：控制数码管显示

2.1 译码器

1. 2-4变量译码器

- 定义：对任意两个输入，对应一个四输出组合，唯一值有一个输出为“0”
- 输出表达式：根据真值表写出输出表达式

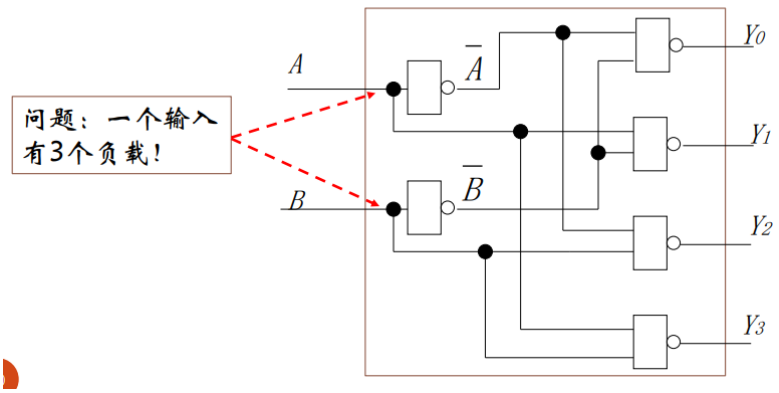
真值表

输入		输出			
A	B	Y_0	Y_1	Y_2	Y_3
0	0	0	1	1	1
1	0	1	0	1	1
0	1	1	1	0	1
1	1	1	1	1	0

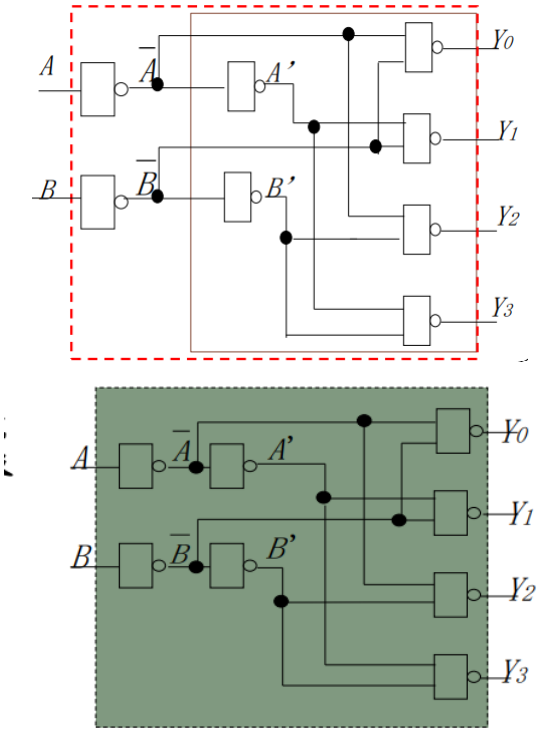
输出表达式

$$\begin{cases} Y_0 = \overline{\overline{A}\overline{B}} \\ Y_1 = \overline{A\overline{B}} \\ Y_2 = \overline{\overline{A}B} \\ Y_3 = \overline{AB} \end{cases}$$

- 逻辑图：根据输出表达式画出逻辑图

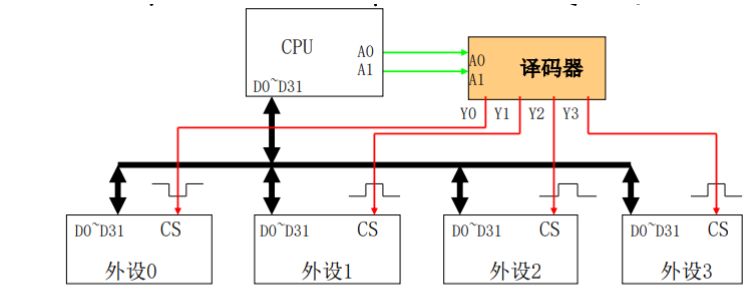


- 检查：**一个输入只能按照一个输入负载计算**，修正设计——增加一级输入缓冲



输入缓冲电路　译码逻辑电路

应用举例：CPU控制四个设备



功能级设计要求：
A0=0，A1=0时，外设0工作
A0=1，A1=0时，外设1工作
A0=0，A1=1时，外设2工作
A0=1，A1=1时，外设3工作

信号级设计要求：
A0=0，A1=0时，Y0=0，Y1,Y2,Y3=1
A0=1，A1=0时，Y1=0，Y0,Y2,Y3=1
A0=0，A1=1时，Y2=0，Y0,Y1,Y3=1
A0=1，A1=1时，Y3=0，Y0,Y1,Y2=1

14

使能（Enable）

可以使得集成电路更加**灵活**（用于**拓展**）、**可靠**（用于**选通**）

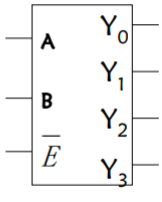
> 有使能端$\overline{E}$的2-4译码器：
> 当$\overline{E}=0$，译码器使能
> 当$\overline{E}=1$，译码器禁止

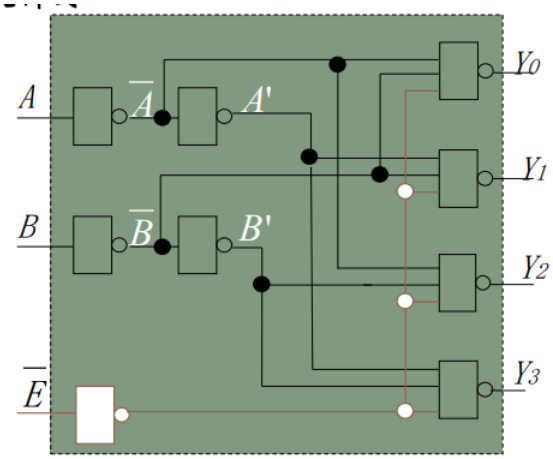
功能表

$\overline{E}$ A B	Y_0 Y_1 Y_2 Y_3
1 X X	1 1 1 1
0 0 0	0 1 1 1
0 1 0	1 0 1 1
0 0 1	1 1 0 1
0 1 1	1 1 1 0

$$\begin{cases} Y_0 = \overline{\overline{E}AB} \\ Y_1 = \overline{\overline{E}A\overline{B}} \\ Y_2 = \overline{\overline{E}\overline{A}B} \\ Y_3 = \overline{\overline{E}\overline{A}\overline{B}} \end{cases}$$

逻辑示意图

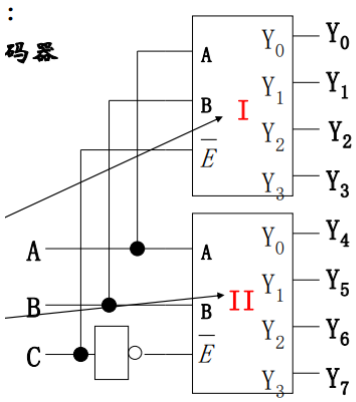




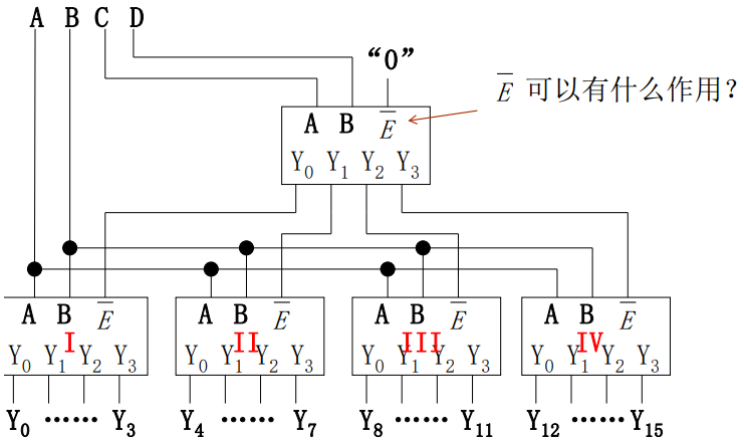
3-8译码器中设计了$\overline{E}_1$、$\overline{E}_{2A}$、$\overline{E}_{2B}$3个使能端

1. 用2片带使能端的2-4译码器组成3-8译码器

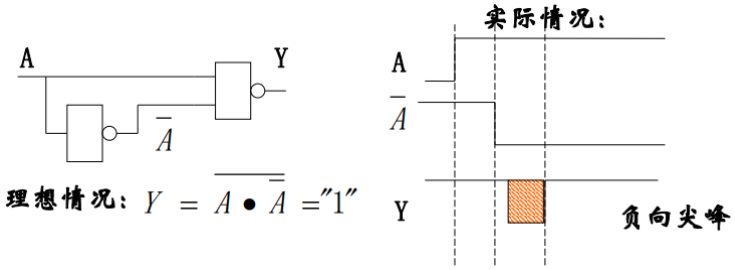
> $C=0$选中片1，$C=1$选中片2



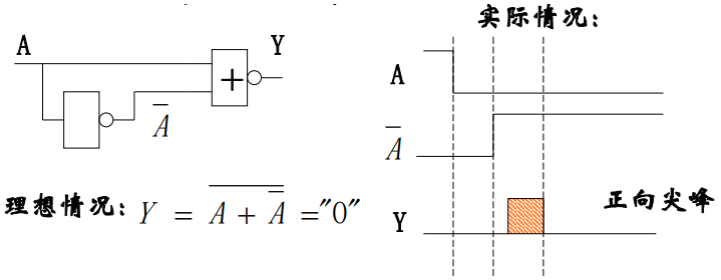
> 用5片带使能端的2-4译码器组成4-16译码器



2. 使能端$\overline{E}$用作选通
   - 二输入**与非门**输入为A和$\overline{A}$时，$\overline{A}$滞后于A，则输出端Y出现尖峰信号（**与非门**上升沿有负向尖峰）



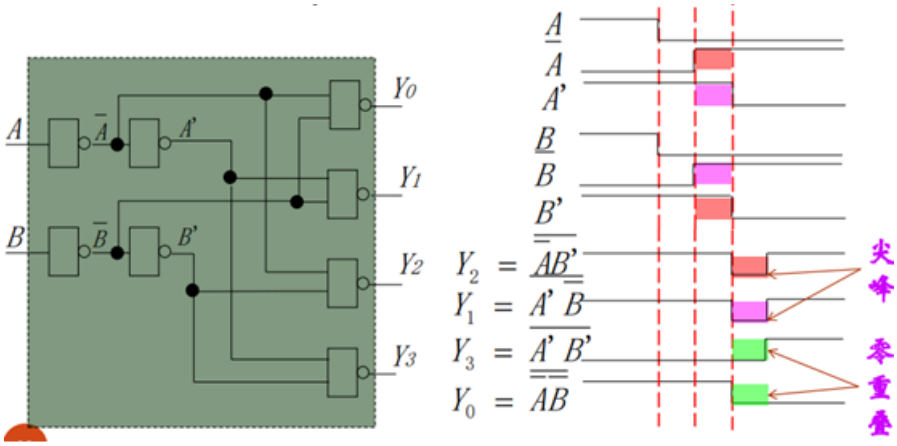
- 二输入**或非门**输入为A和$\overline{A}$时，$\overline{A}$滞后于A，则输出端Y出现尖峰信号（**或非门**下降沿有正向尖峰）



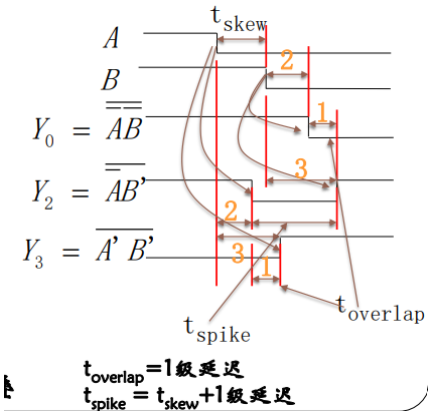
<u>若没有使能端，延迟产生**尖峰**和**零重叠问题**</u>

> 假设A、B从“11”变到“00”

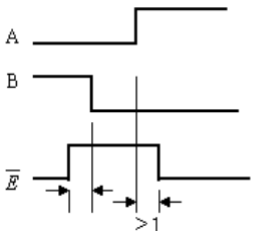
- 若A、B同时到来，无偏移Skew



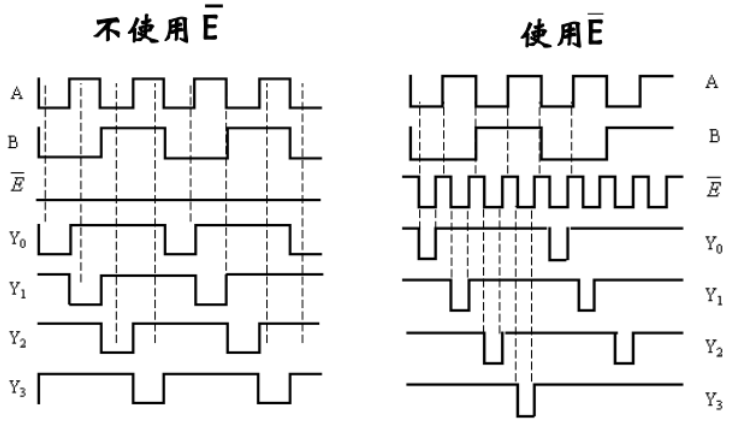
- 若A、B不能同时到来，存在偏移Skew，导致尖峰信号更宽



> 在A、B变化期间，输出是不稳定的，可能会出现尖峰信号。加一个能覆盖输入变化的正脉冲($\overline{E}=1$)，使得A、B变化期间强制$Y0 - Y3 = 1$，即可消除输出端的干扰。



抑制尖峰和零重叠的使能正信号应先于（或同时）译码器的变量输入变化前到来，正信号撤除应滞后于变量输入的变化(至少滞后1级缓冲的延迟)。
但也不能太宽，否则速度会慢。



2. 3-8变量译码器

1. 真值表和逻辑表达式、输出表达式

➢ **真值表和逻辑表达式**

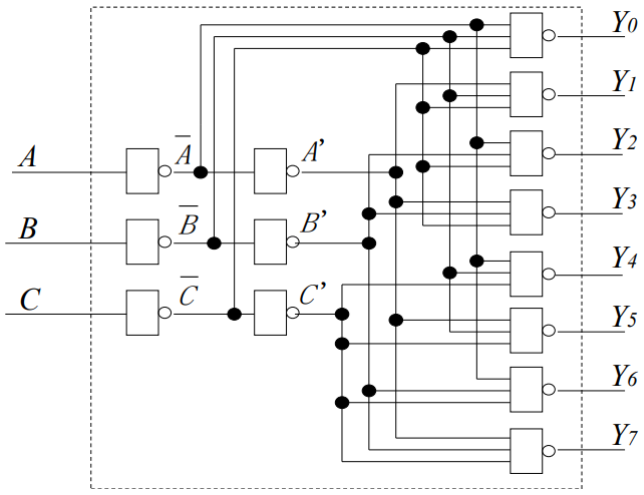
真　值　表

输　入	输　　出
A B C	Y_0 Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7
0 0 0	0 1 1 1 1 1 1 1
1 0 0	1 0 1 1 1 1 1 1
0 1 0	1 1 0 1 1 1 1 1
1 1 0	1 1 1 0 1 1 1 1
0 0 1	1 1 1 1 0 1 1 1
1 0 1	1 1 1 1 1 0 1 1
0 1 1	1 1 1 1 1 1 0 1
1 1 1	1 1 1 1 1 1 1 0

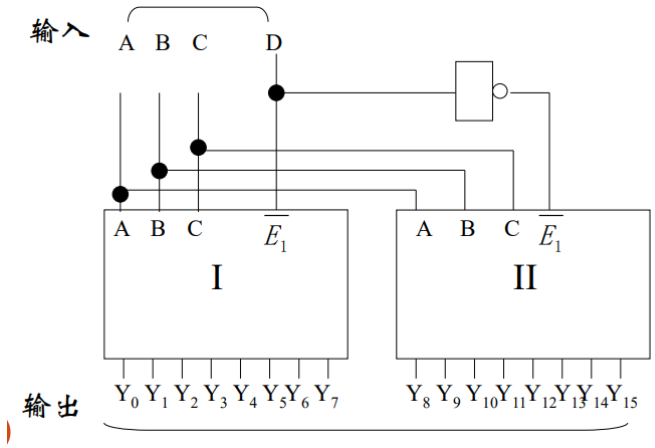
输出表达式

$$\begin{cases} Y_0 = \overline{\overline{A}\overline{B}\overline{C}} \\ Y_1 = \overline{\overline{A}\overline{B}C} \\ Y_2 = \overline{\overline{A}B\overline{C}} \\ Y_3 = \overline{\overline{A}BC} \\ Y_4 = \overline{A\overline{B}\overline{C}} \\ Y_5 = \overline{A\overline{B}C} \\ Y_6 = \overline{AB\overline{C}} \\ Y_7 = \overline{ABC} \end{cases}$$

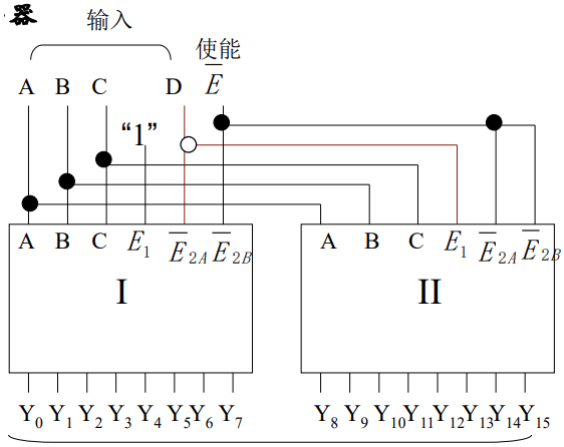
2. 按照输出表达式画出3-8译码器的逻辑图



3. 用3-8译码器扩展成4-16译码器



> 使能端E用作拓展



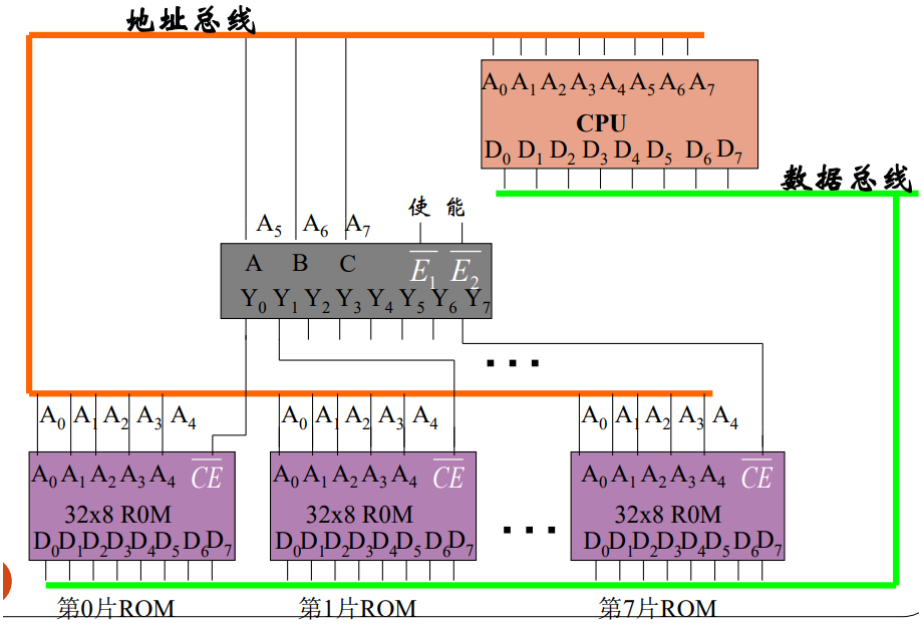
3. 变量译码器的应用

(1) 多个使能端译码器件——典型器件

- 器件一：74LS(HCT, HC) 139，功能：双2-4译码器
- 器件二：84LS(HCT, HC) 138，功能：3-8译码器（3个使能端）
- 器件三：74LS（HCT, HC) 154，功能：4-16译码器（2个使能端）

(2) 用3-8译码器分配地址区

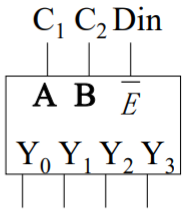
> CPU的地址空间：A_7~A_0共有256个地址空间，每个ROM有32个地址空间
>
> CPU地址空间的高三位$A_7A_6A_5$作为3-8译码器的输入，输出为8片ROM的$\overline{CE}$，以及CPU地址空间的第五位$A_4A_3A_2A_1A_0$



(3) 用作数据分配器 Demultiplexer

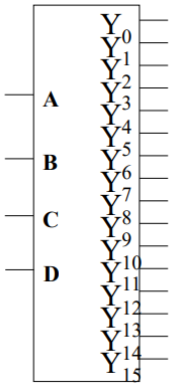
> 数据分配：将输入数据在地址控制下连接到多个输出通道
>
> 数据分配器将输入数据端Din根据地址码C_1，C_2分配到输出端Y_0~Y_3中的相应一个中去

Din	C_1	C_2	Y_0	Y_1	Y_2	Y_3
0/1	0	0	0/1	1	1	1
0/1	1	0	1	0/1	1	1
0/1	0	1	1	1	0/1	1
0/1	1	1	1	1	1	0/1



4. 4-16变量译码器

> 逻辑示意图：（D为最高位，A为最低位，含有2个使能端）



存在的问题：

- 缓冲门的负载较大
  第一级缓冲门（反变量）有**9**个负载，第二级缓冲门（原变量）有**8**个负债
  使能端与门的负载有**16**个
- 当输入变量N增大时
  译码部分与非门的输入端数会增多：输入端数为$N+1$（使能端）个
  负载：第一级为$2^{N-1}+1$，第二级为2^{N-1}，使能端为2^N
- **采用多级译码技术可以减少负载**：用在大容量存储器片内的译码结构

多级译码与二级译码

令$E=\overline{AB}, F=\overline{AB}, G=\overline{AB}, H=AB$

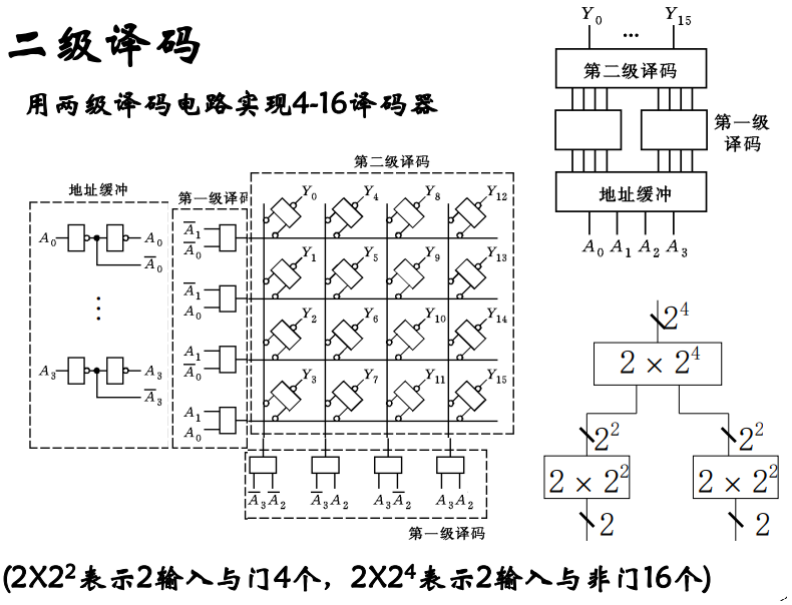
$W=\overline{CD}, X=C\overline{D}, Y=\overline{C}D, Z=CD$，则有：

$Y_0=\overline{\overline{ABCD}}=\overline{EW}$	$Y_4=\overline{\overline{ABCD}}=\overline{EX}$	$Y_8=\overline{\overline{ABCD}}=\overline{EY}$	$Y_{12}=\overline{\overline{ABCD}}=\overline{EZ}$
$Y_1=\overline{\overline{ABCD}}=\overline{FW}$	$Y_5=\overline{\overline{ABCD}}=\overline{FX}$	$Y_9=\overline{\overline{ABCD}}=\overline{FY}$	$Y_{13}=\overline{\overline{ABCD}}=\overline{FZ}$
$Y_2=\overline{\overline{ABCD}}=\overline{GW}$	$Y_6=\overline{\overline{ABCD}}=\overline{GX}$	$Y_{10}=\overline{\overline{ABCD}}=\overline{GY}$	$Y_{14}=\overline{\overline{ABCD}}=\overline{GZ}$
$Y_3=\overline{\overline{ABCD}}=\overline{HW}$	$Y_7=\overline{\overline{ABCD}}=\overline{HX}$	$Y_{11}=\overline{\overline{ABCD}}=\overline{HY}$	$Y_{15}=\overline{\overline{ABCD}}=\overline{HZ}$

✓A,B,C,D和它们的反变量，都出现8次，说明它们共有8个负载。

✓考察E,F,G,H,W,X,Y,Z，每个出现4次，意味着它们共有4个负载。

✓说明经过变换后，负载数降低了一半。



(2X2²表示2输入与门4个，2X2⁴表示2输入与非门16个)

5. 码制译码器

> 将一种编码转换为另一种编码的逻辑电路

二-十进制译码器

BCD编码：Binary-Coded to Decimal，二进制编码的十进制数

- 不完全译码的BCD译码器

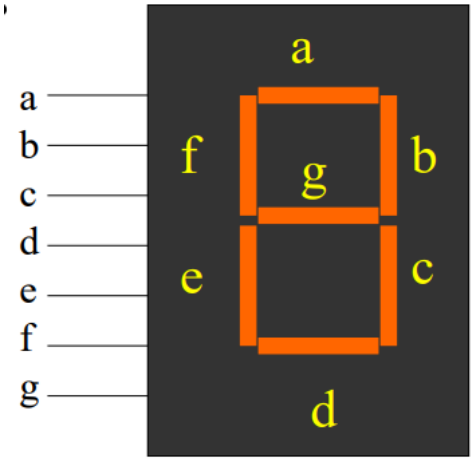
	A	B	C	D	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	0	1	1	1	1	1	1	1	1
2	0	1	0	0	1	1	0	1	1	1	1	1	1	1
3	1	1	0	0	1	1	1	0	1	1	1	1	1	1
4	0	0	1	0	1	1	1	1	0	1	1	1	1	1
5	1	0	1	0	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	1	1	1	0	1	1	1	1	1	1	1	0	1	1
8	0	0	0	1	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0

- 完全译码的BCD译码器：更好地debug

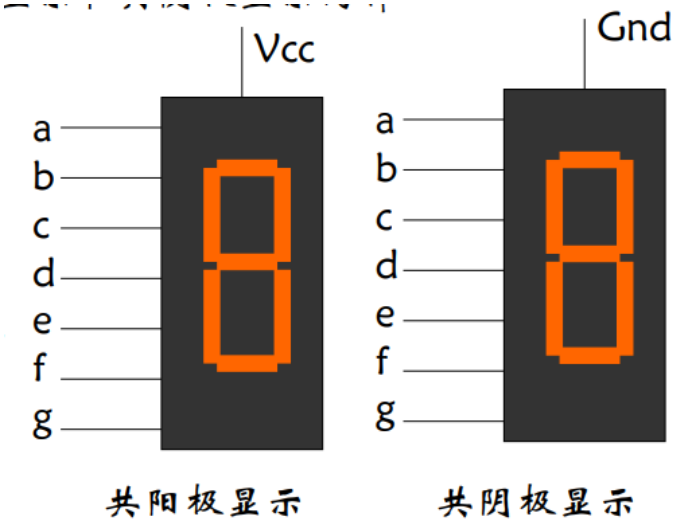
6. 显示译码器

七段数码管

> 一个七段数码管有7个控制端输入$a\sim g$，分别 对应与数码管的7段。有些数码管是8个输入， 在右下方有一个小数点。



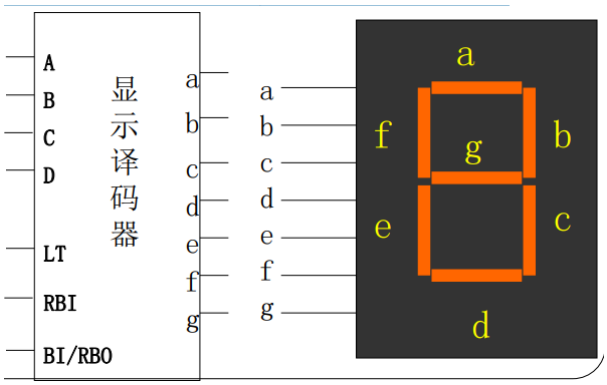
- 共阳极显示：高灭低亮（以下例子均为共阳极显示）
- 共阴极显示：高亮低灭

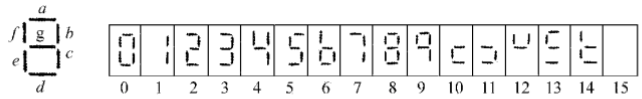


显示0~9中的任何一个数字分别对应于$a\sim g$的一组编码

显示译码器

> 在BCD码和7段数码管编码之间需要一个**显示译码器**
>
> - 将BCD码转换为数码管对应的编码
> - 显示译码器实质是码制译码器





显示数字	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	1	0	0	0	1	0	0	1	1	1	1
2	0	1	0	0	0	0	1	0	0	1	0
3	1	1	0	0	0	0	0	0	1	1	0
4	0	0	1	0	1	0	0	1	1	0	0
5	1	0	1	0	0	1	0	0	1	0	0
6	0	1	1	0	1	1	0	0	0	0	0
7	1	1	1	0	0	0	0	1	1	1	1
8	0	0	0	1	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	1	1	0	0
A	0	1	0	1	1	1	1	0	0	1	0
B	1	1	0	1	1	1	0	0	1	1	0
C	0	0	1	1	1	0	1	1	1	0	0
D	1	0	1	1	0	1	1	0	1	0	0
E	0	1	1	1	1	1	1	0	0	0	0
F	1	1	1	1	1	1	1	1	1	1	1

各个变量的逻辑表达式：

$$
\begin{aligned}
a &= BD + \overline{A}C + A\overline{B}\,\overline{C}\,\overline{D} \\
b &= B\overline{D} + A\overline{B}C + \overline{A}BC \\
c &= A\overline{B}\,\overline{C} + C\overline{D} \\
d &= A\overline{B}C + \overline{A}B\overline{C} + ABC \\
e &= A + \overline{B}C \\
f &= A\overline{C}\,\overline{D} + AB + B\overline{C} \\
g &= \overline{B}C\overline{D} + ABC
\end{aligned}
$$

控制功能输入：$LT, RBI, BI/RBO$（主要用于对数码管的测试和其他一些应用）

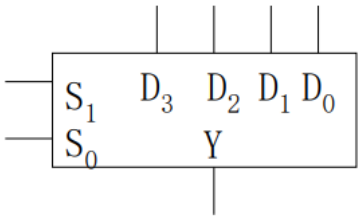
2.2 数据选择器

> 在选择控制的信号作用下，能从多个输入数据中选择一个或多个作为输出。
>
> - 多输入单输出数据选择器
> - 多输入多输出数据选择器

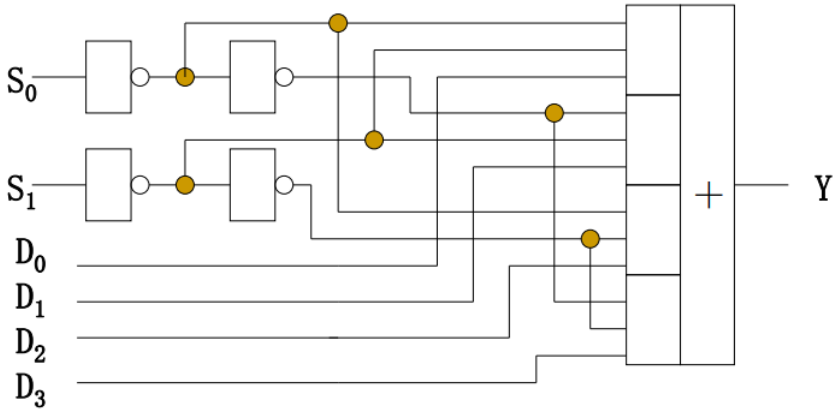
1. 4选1数据选择器

> 4输入，1输出，2个选择控制

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

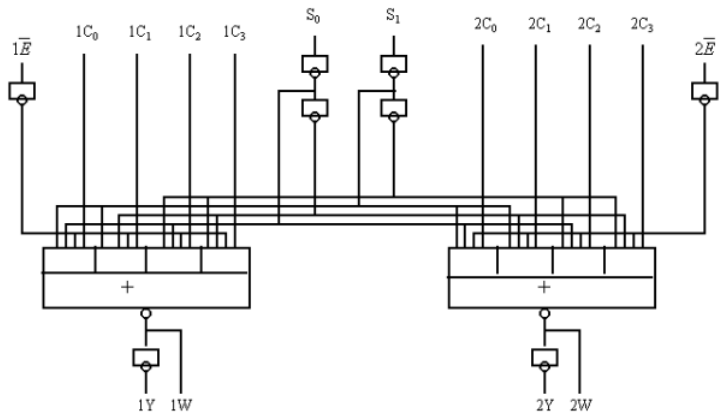


$$Y = \overline{S_0}\,\overline{S_1}D_0 + S_0\overline{S_1}D_1 + \overline{S_0}S_1D_2 + S_0S_1D_3$$



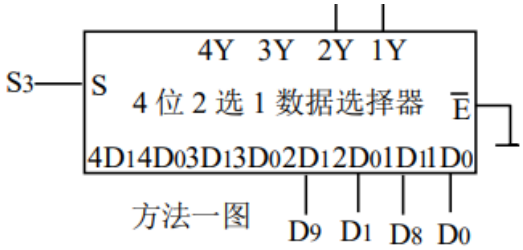
2. 有使能端的双4选1数据选择器

> 提供1正1反两个输出



3. 4位2选1数据选择器

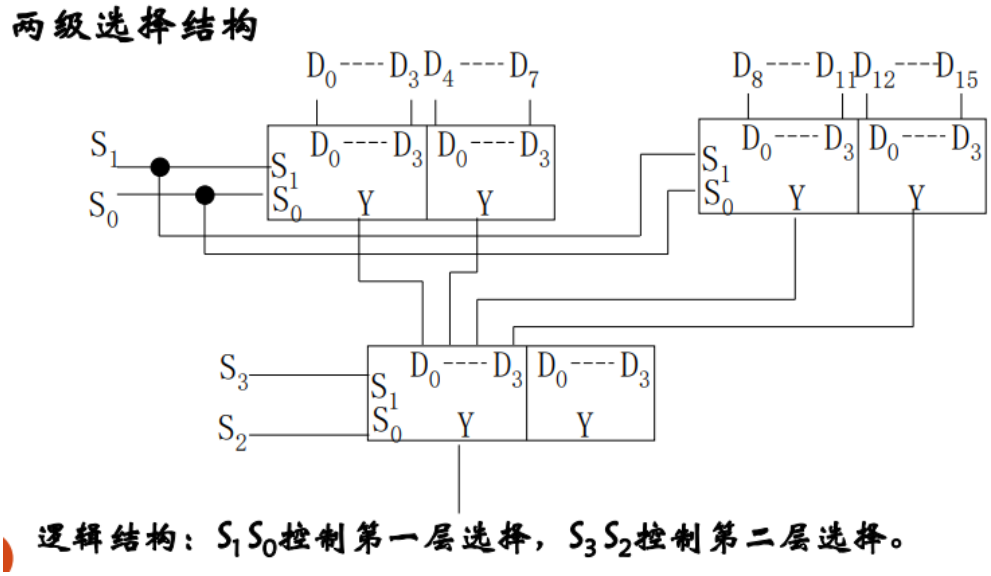
> 4个二选一选择器，共用选择输入S_3



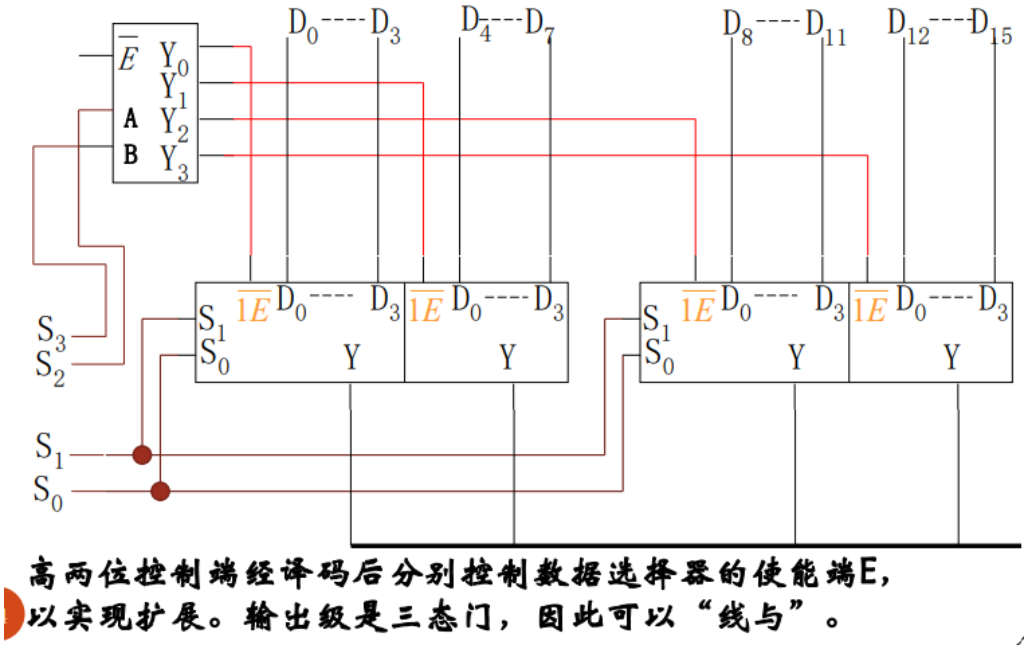
4. 16选1选择器

> 选择器扩展

- 用无使能端的双4选1选择器

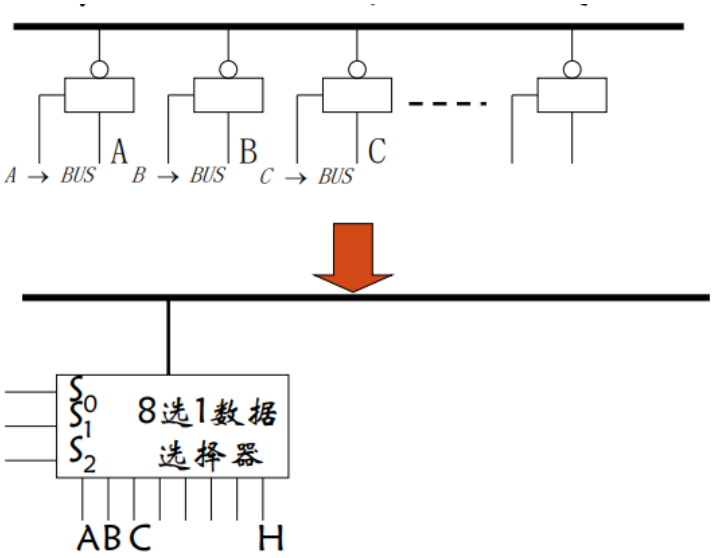


- 用有使能端的的双4选1选择器



应用

总线发送控制



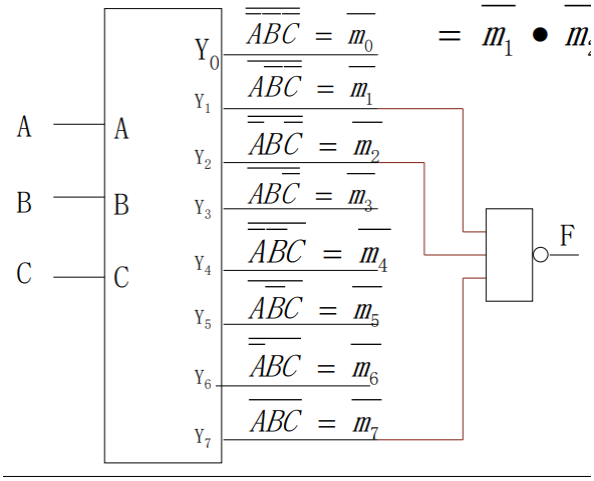
实现逻辑函数

1. 译码器实现逻辑函数

> 译码器：可以看成是N个输入变量组成的2^N个最小项
> 再加一级与非门，可以组成“与非-与非”逻辑，也可以表达“与-或”逻辑
> 因此，可以用译码器实现**“与-或”逻辑函数**

- 举例

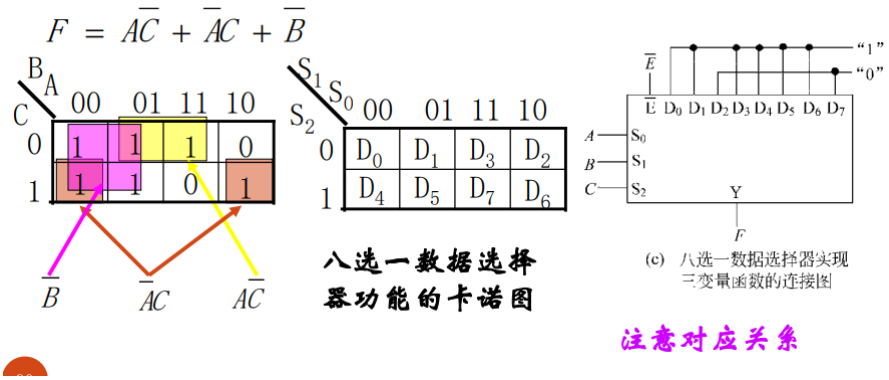
$$F = A\overline{BC} + \overline{A}B\overline{C} + ABC = m_1 + m_2 + m_7 = \overline{\overline{m_1} \cdot \overline{m_2} \cdot \overline{m_7}}$$



2. 数据选择器实现逻辑函数

> 数据选择器：逻辑结构就是“与-或”表达式
> 可以看成是N个控制端的2^N个最小项和2^N个输入组成的“与-或”表达式
> 选择某些输入为"1"，就是选中这些最小项组成的逻辑函数

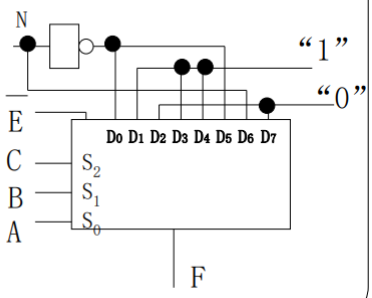
- 举例（4选1选择器）：$Y = \overline{S_0 S_1} D_0 + S_o \overline{S_1} D_1 + \overline{S_0} S_1 D_2 + S_0 S_1 D_3$
- 举例：用八选一数据选择器实现3变量函数



- 举例：用八选一数据选择器实现4变量函数

$$F = A\overline{C} + \overline{BN} + \overline{A}CN$$

S_0 S_1 S_2	输入	函数F的值
0 0 0	D_0	$F = A\overline{C} + \overline{B}\,\overline{N} + \overline{A}CN = \overline{N}$
1 0 0	D_1	$F = A\overline{C} + \overline{B}\,\overline{N} + \overline{A}CN = 1$
0 1 0	D_2	$F = A\overline{C} + \overline{B}\,\overline{N} + \overline{A}CN = 0$
1 1 0	D_3	$F = A\overline{C} + \overline{B}\,\overline{N} + \overline{A}CN = 1$
0 0 1	D_4	$F = A\overline{C} + \overline{B}\,\overline{N} + \overline{A}CN = 1$
1 0 1	D_5	$F = A\overline{C} + \overline{B}\,\overline{N} + \overline{A}CN = \overline{N}$
0 1 1	D_6	$F = A\overline{C} + \overline{B}\,\overline{N} + \overline{A}CN = N$
1 1 1	D_7	$F = A\overline{C} + \overline{B}\,\overline{N} + \overline{A}CN = 0$



2.3 编码器

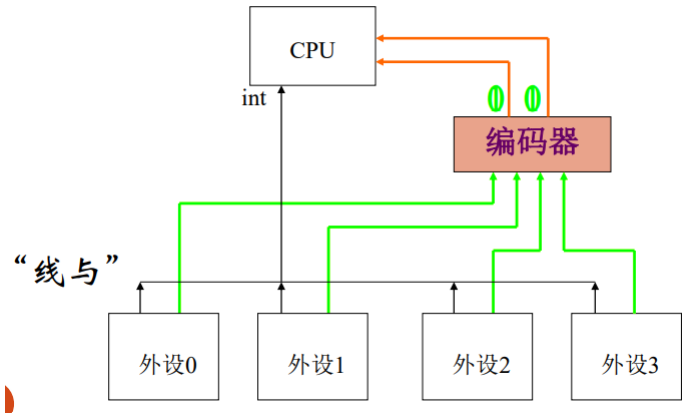
1. 编码器原理

> 将译码器反过来，对应输入的每 一个状态，输出一个编码。

常用编码器：

- 4-2编码：将输入的4个状态编成2位二进制数码
- 8-3编码：将输入的8个状态编成3位二进制数码
- BCD编码：将10个输入编成BCD码

编码器的应用：



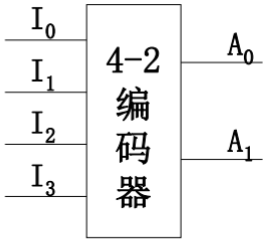
编码器的注意事项：

部分编码器设计（如下面的编码器）只有在**互斥输入**的时候才能使用。

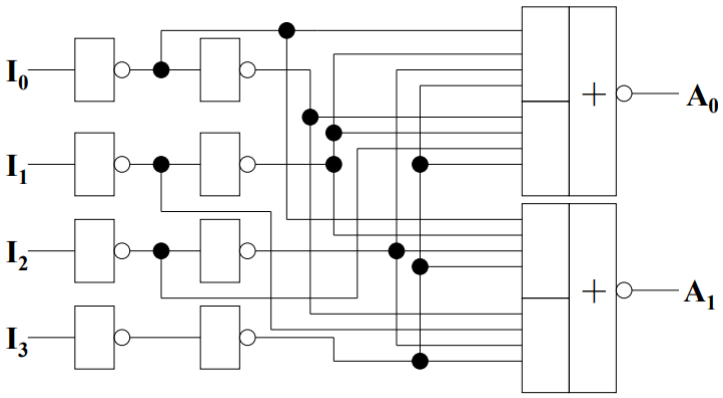
4-2编码器

功能表

I_0	I_1	I_2	I_3	A_0	A_1
0	1	1	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	0	1	1

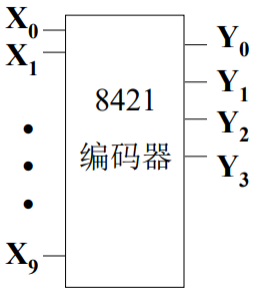


$$\begin{cases} A_0 = I_0\overline{I_1}I_2I_3 + I_0I_1I_2\overline{I_3} = \overline{\overline{I_0}I_1I_2I_3 + I_0I_1\overline{I_2}I_3} \\ A_1 = I_0I_1\overline{I_2}I_3 + I_0I_1I_2\overline{I_3} = \overline{\overline{I_0}I_1I_2I_3 + I_0\overline{I_1}I_2I_3} \end{cases}$$



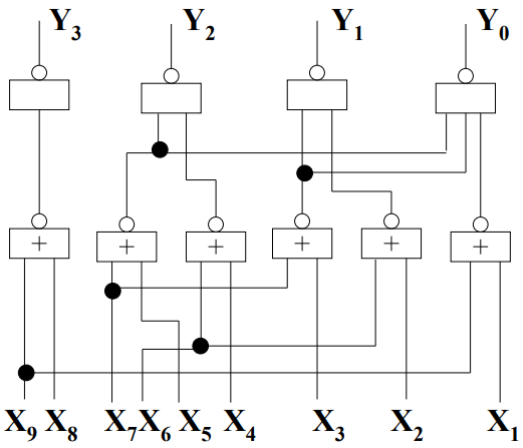
8421编码器

> 将输入信号编码成8421码



X_9	X_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	1

$$Y_3 = X_8 + X_9$$
$$Y_2 = X_4 + X_5 + X_6 + X_7$$
$$Y_1 = X_2 + X_3 + X_6 + X_7$$
$$Y_0 = X_1 + X_3 + X_5 + X_7 + X_9$$



2. 优先编码器

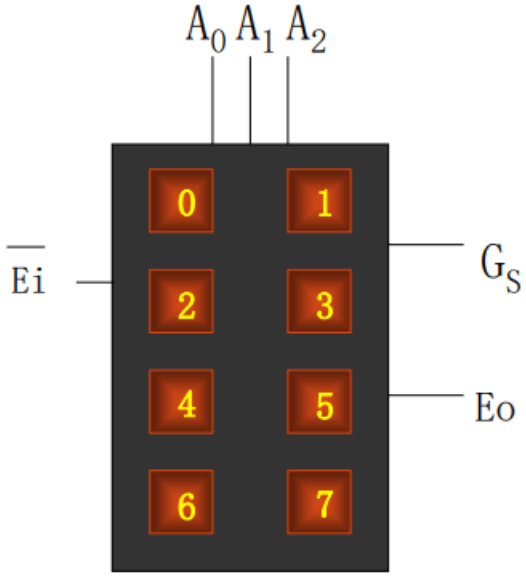
> 当两条或两条以上线为“0”时，优先按输入编号大的编码，称优先编码器 (Priority Encoder) 。

应用:

- 设备按照优先等级编码，用于中断响应
- 键盘输入的读取

8-3优先编码器

- 对各键进行编码：A_2, A_1, A_0
- 输入使能$\overline{E_i}$
- 设定输出的编码是否有效指示G_s
- 设定是否允许编码级联输出E_0



1. 8-3优先编码功能表

$\overline{Ei}$	0	1	2	3	4	5	6	7	A_0	A_1	A_2	G_S	Eo
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	1	0	0	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	1	1	0	0	1
0	X	X	X	0	1	1	1	1	0	0	1	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
1	X	X	X	X	X	X	X	X	1	1	1	1	1

(A_2,A_1,A_0用反码编码，Gs为编码输出有效，$\overline{Ei}$为使能输入，Eo为允许级联输出)

$\overline{Ei}$	0	1	2	3	4	5	6	7	A_0	A_1	A_2	G_S	Eo
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	1	0	0	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	1	1	0	0	1
0	X	X	X	0	1	1	1	1	0	0	1	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
1	X	X	X	X	X	X	X	X	1	1	1	1	1

$$\overline{A_0} = \overline{7} + 76\overline{5} + 765\overline{4} + 7654\overline{3} + 76543\overline{2} + 765432\overline{1} = \overline{7} + 6\overline{5} + 64\overline{3} + 642\overline{1}$$

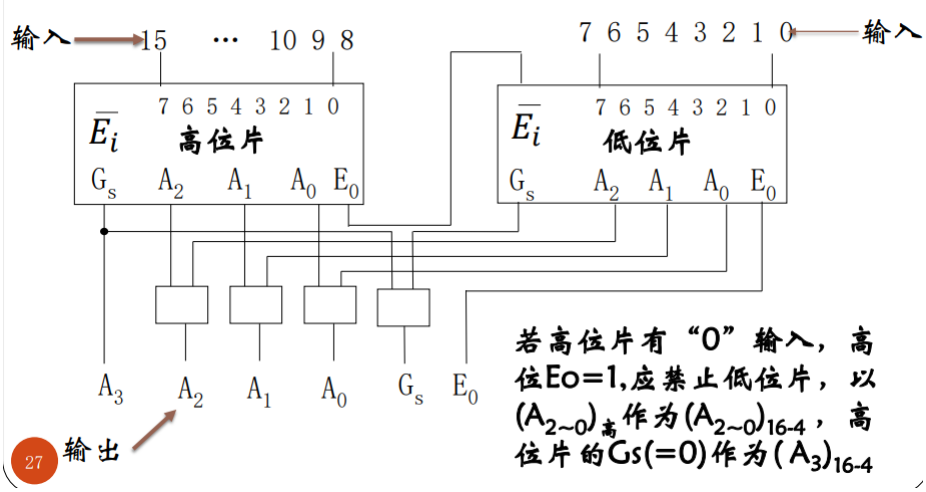
类似方法可以得到：

$$\overline{A_0} = \overline{7} + 6\overline{5} + 64\overline{3} + 642\overline{1}$$
$$\overline{A_1} = \overline{7} + \overline{6} + 54\overline{3} + 54\overline{2}$$
$$\overline{A_2} = \overline{7} + \overline{6} + \overline{5} + \overline{4}$$
$$E_0 = \overline{\overline{\overline{E_i}} \cdot 76543210}$$
$$G_s = \overline{E_o \cdot \overline{\overline{E_i}}}$$

如果$\overline{E_i} = 0$，输入有“0”，则$G_s = 0$，$E_o = 1$，编码有效，级联禁止
如果$\overline{E_i} = 0$，输入全为“1”，则$G_s = 1$，$E_o = 0$，编码无效，级联有效
如果$\overline{E_i} = 1$, 无论输入为何值，$G_s = 1$，$E_o = 1$，编码无效，级联禁止
输出编码为**反码**，即用“000”表示7，用“111”表示0

4-16优先编码器

> 将8-3优先编码器扩展为16-4优先编码器



若高位片有“0”输入，高位$E_o=1$，应禁止低位片，以$(A_{2-0})_{高}$作为$(A_{2-0})_{16-4}$，高位片的$Gs(=0)$作为$(A_3)_{16-4}$

若高位片无“0”输入，高位$Eo=0$，低位片工作，以$(A_{2-0})_{低}$作为$(A_{2-0})_{16-4}$，$(A_3)_{16-4}$为“1”

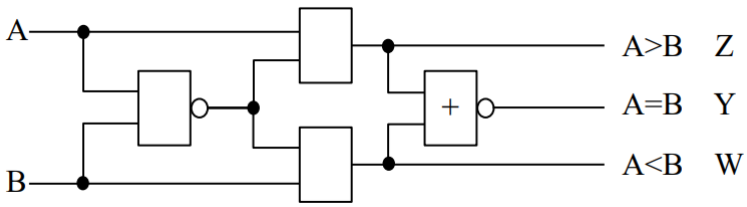
2.4 数据比较器

> 定义：数字系统中能够完成数据比较功能的部件
> 功能：比较A、B两数，判断A>B、A<B、A=B
> $A_3, A_2, A_1, A_0, B_3, B_2, B_1, B_0$
> 从高位开始比较，若$A_3 > B_3$，则$A > B$
> 若$A_3 < B_3$，则$A < B$
> 若$A_3 = B_3$，则再比较低位

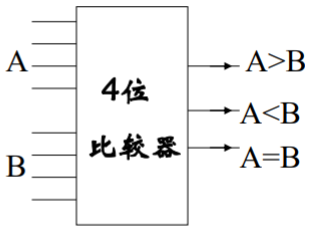
$A_i>B_i$的条件：$A_i=1$，$B_i=0$；即 $A_i \cdot \overline{B_i} = 1$或$Z = A_i \cdot \overline{A_i B_i} = 1$

$A_i<B_i$的条件：$A_i=0$，$B_i=1$；即 $\overline{A_i} \cdot B_i = 1$或$W = B_i \cdot \overline{A_i B_i} = 1$

$A_i=B_i$的条件：$\overline{A_i \oplus B_i} = 1$ 或 $Y = \overline{(A_i \cdot \overline{A_i} \cdot \overline{B_i}) + (\overline{A_i} \cdot \overline{B_i} \cdot B_i)} = 1$



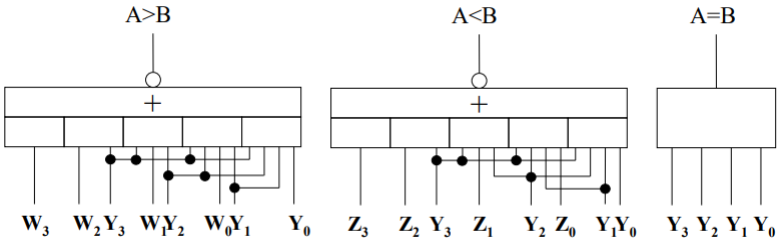
A_3 B_3	A_2 B_2	A_1 B_1	A_0 B_0	A>B	A<B	A=B
$A_3 > B_3$	X	X	X	1	0	0
$A_3 < B_3$	X	X	X	0	1	0
$A_3 = B_3$	$A_2 > B_2$	X	X	1	0	0
A3= B3	A2 <B2	X	X	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	0	0	1



$$W_i = [A_i < B_i] = B_i \cdot \overline{A_i B_i}$$

$$Y_i = [A_i = B_i] = \overline{(A_i \cdot \overline{A_i B_i}) + (\overline{A_i B_i}) \cdot B_i}$$

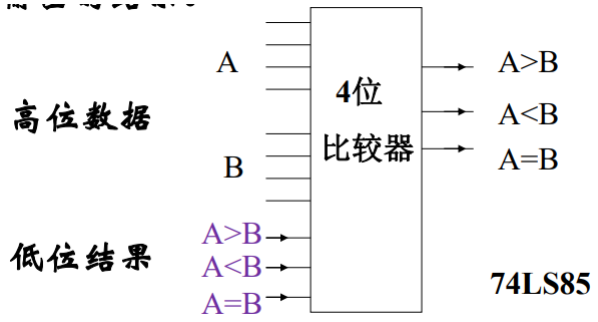
$$Z_i = [A_i > B_i] = A_i \cdot \overline{A_i Bi}$$



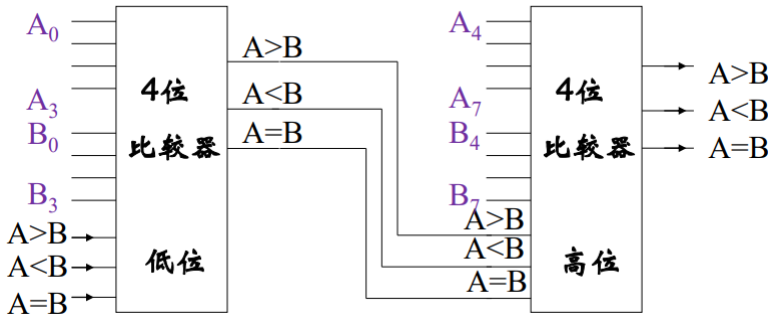
- **分段比较**：多片比较器构成更长位数的方法
- 比较器不仅输出比较结果，还要能接受其它片输出的结果

1. 4位比较器

举例：74LS85



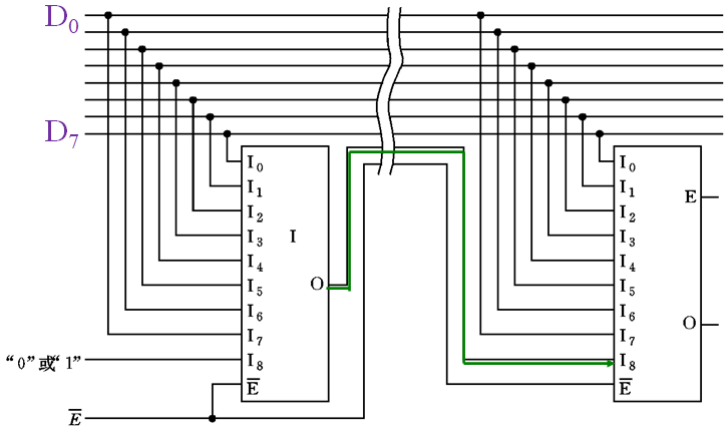
2. 8位比较器



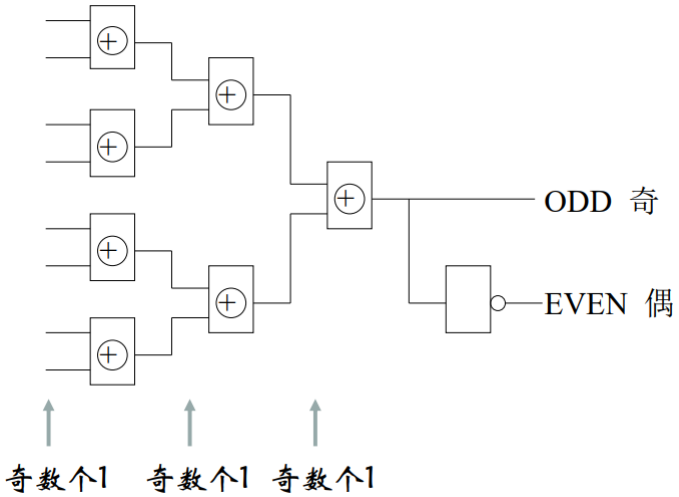
2.5 奇偶校验器

> **奇偶检验**指检验数据中包含奇数个“1”还是偶数个“1”

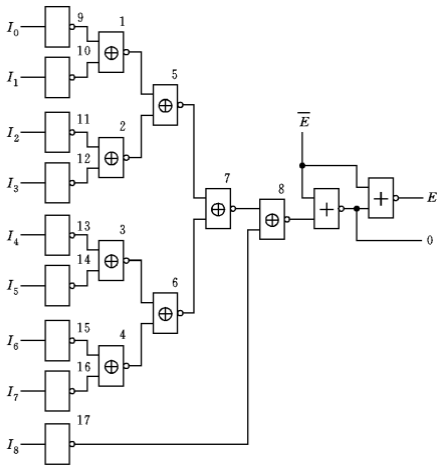
用于发送和接收的奇偶校验逻辑结构图：



异或门构成八位奇偶校验位产生电路：



九位奇偶检验电路：



2.6 可编程逻辑器件

硬件描述语言（HDL）：用户可编程，设计方便，易于实现

可编程逻辑器件（PLD / Programmable Logic Device）：属于中小规模可编程逻辑器件

ROM (Read-Only Memory)：只读存储器

PLA (Programmable Logic Array)：可编程逻辑阵列

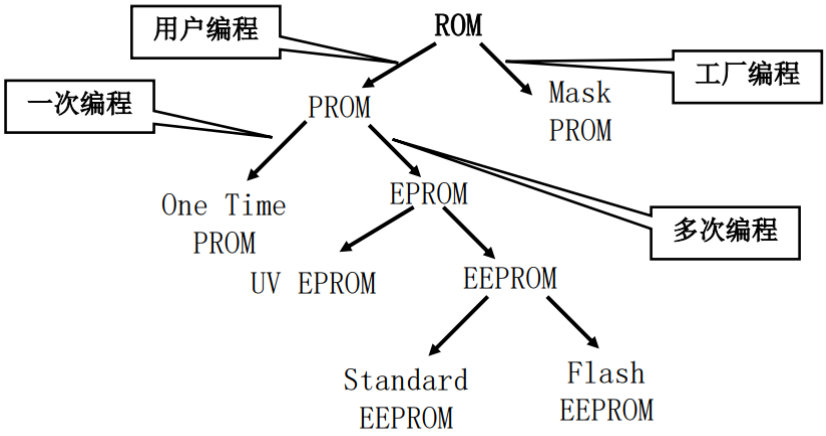
PAL (Programmable Array Logic)：可编程阵列逻辑

GAL (General Array logic)：通用阵列逻辑

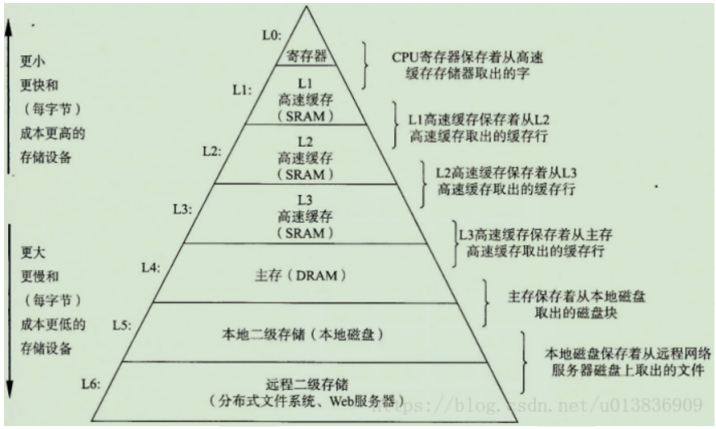
1. ROM 只读存储器

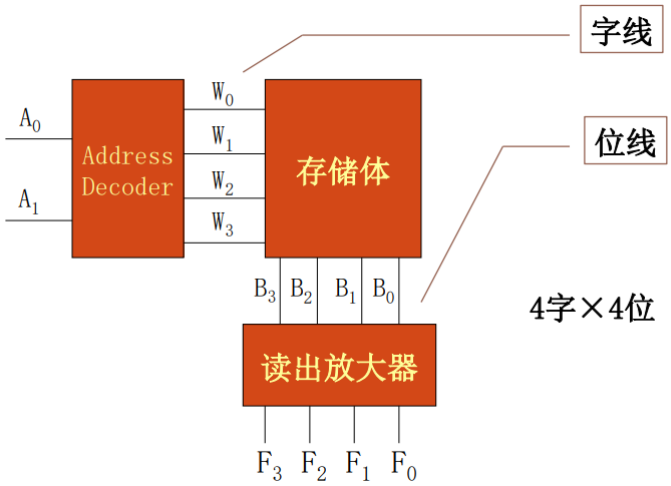
两大类存储器 (Memory)：

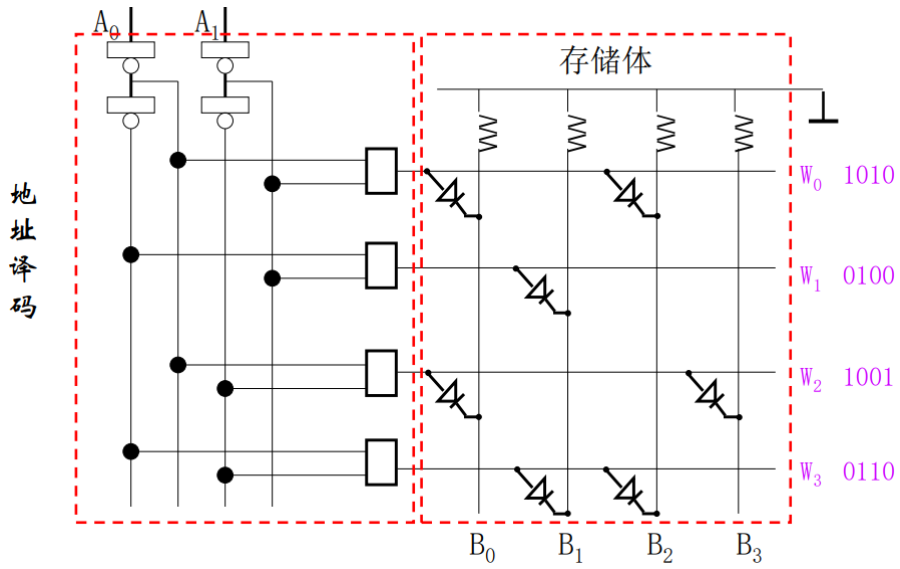
- ROM / Read-Only Memory：一旦信息写入，在机器上只读
  存放固定信息：程序、常数、指令...
  信息非易失" (Nonvolatile)；结构简单、规律性强、容量大
  - 掩模型ROM (Mask ROM)：工厂编程
    用户提交码点，在工厂编程
  - 可编程ROM (PROM)：用户一次编程
    出厂保留全部熔丝，用户可编程但不可改写
  - 可改写ROM (EPROM)：用户多次编程
    光可改写：UV EPROM
    电可改写：E^2PROM



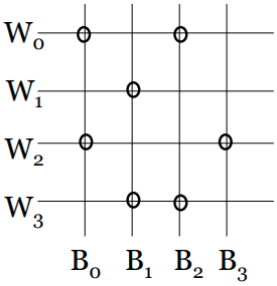
- RAM / Random-Access Memory：随机存储器，在运行状态可读可写
  对比磁盘：块粒度访问较小（ROM请求64B可能返回4KB，RAM则只会返回64B），因此RAM的访存速度更快

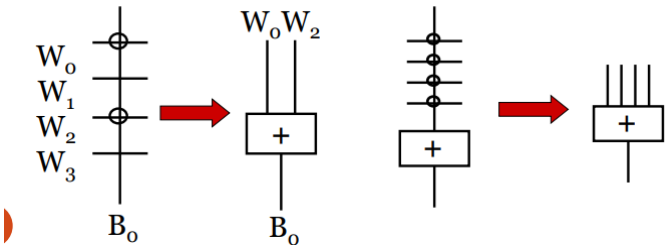






> 其中，存储体的一列表示“二极管或门”，因此，存储体可以画成**或阵列**

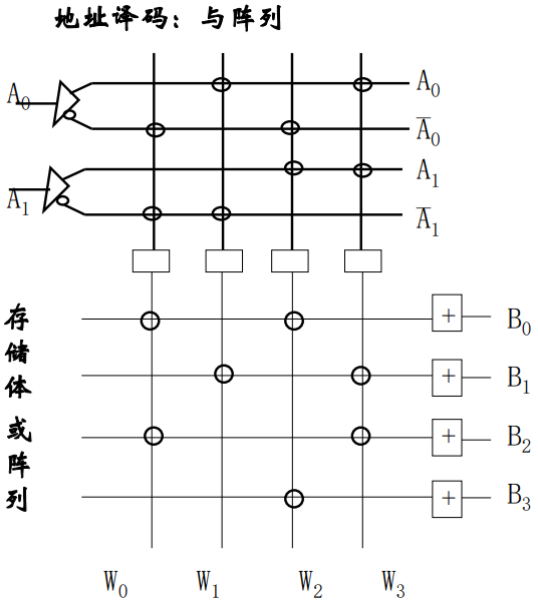




ROM的特点：

- 输入地址和存储信息一一对应
- 与阵列是译码器，包括全部最小项，信息表完全

ROM存储器逻辑结构

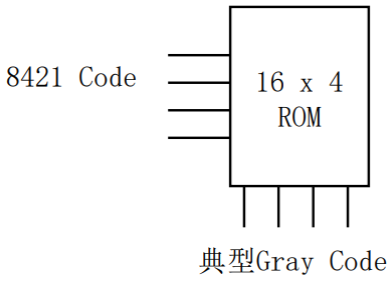


> 圆圈节点：固定连接 / 已变成连接
> 叉叉节点：可编程连接

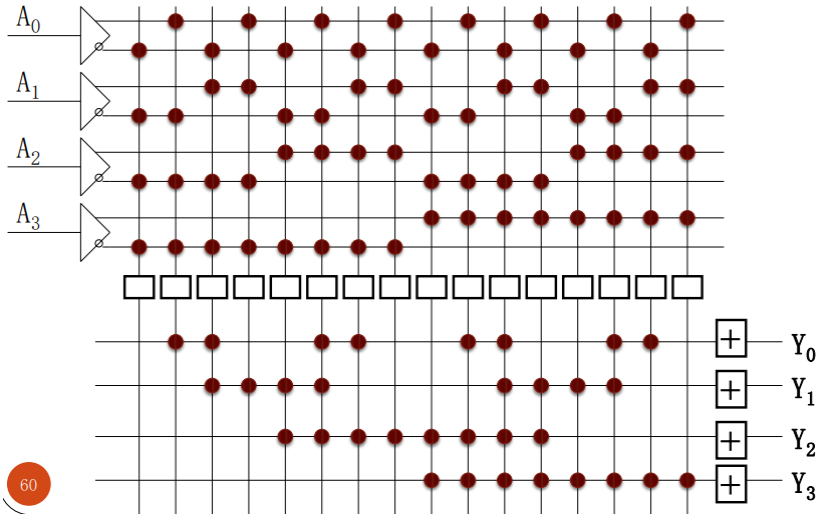
2. ROM的应用

码制变换

> 8421 => 典型的Gray Code
> ROM中存储真值表（或阵列）

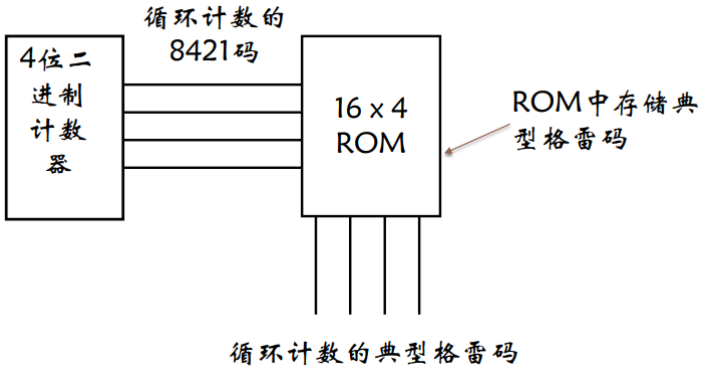


十进制数	8421码	典型格雷码
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111



其中，与阵列为8×16=128，或阵列4×16=64，总点数为192

- 用4为二进制计数器 + ROM实现典型格雷码计数器



三位二进制数的平方

> ROM是用与门构成的变量译码器及或门组合而成
> 将真值表存储在ROM里实现组合函数的功能
> 可以用软件方式来建立布尔函数的信息

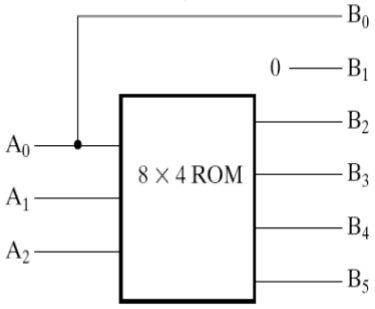
Inputs			Outputs						
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

B0与A0相同

B1为恒”0”

可选择8×4bit的ROM

ROM输出对应平方的输出值



A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

3. 可编程逻辑阵列 PLA

> PLA针对ROM的逻辑压缩：ROM的与阵列不一定产生所有的最小项，只需产生逻辑函数所需的乘积项即可
>
> **PROM：与阵列固定、或阵列可编程**
> **PLA：与、或阵列均可编程**
> **PAL：与阵列可编程、或阵列固定**

- PLA实现组合逻辑时，将逻辑表达式写成最简与-或表达式，在用与项的“或”操作构成或阵列
- 【例：用PLA实现8421码到格雷码】

□ 8421码表示为$B_3B_2B_1B_0$，格雷码表示为$G_3G_2G_1G_0$。

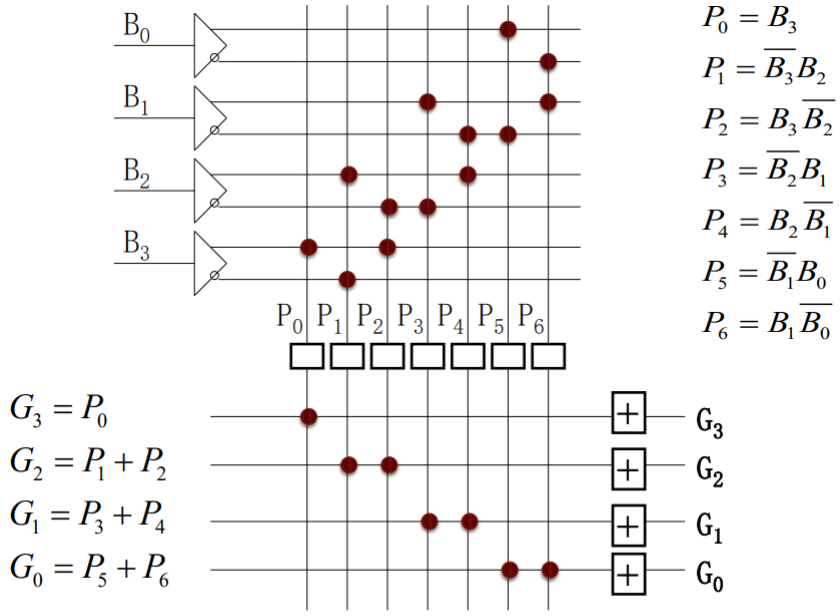
□ 格雷码和8421码的转换关系如下：

$$G_3 = B_3$$
$$G_2 = B_3 \oplus B_2 = \overline{B_3}B_2 + B_3\overline{B_2}$$
$$G_1 = B_2 \oplus B_1 = \overline{B_2}B_1 + B_2\overline{B_1}$$
$$G_0 = B_1 \oplus B_0 = \overline{B_1}B_0 + B_1\overline{B_0}$$

$$P_0 = B_3$$
$$P_1 = \overline{B_3}B_2$$
$$P_2 = B_3\overline{B_2}$$
$$P_3 = \overline{B_2}B_1$$
$$P_4 = B_2\overline{B_1}$$
$$P_5 = \overline{B_1}B_0$$
$$P_6 = B_1\overline{B_0}$$

将每个不同的与项用乘积项Pi表示

> P项用与阵列实现，G_i用或阵列实现



与阵列8×7=56，或阵列4×7=28，总点数84

- PLA与ROM的区别
  ROM的信息表原封不动，全译码
  PLA作了逻辑压缩，信息表改动很大，但逻辑上等价

PROM,PLA,PAL对比

- PROM
  - 与阵列固定，或阵列可编程，实现组合逻辑
  - 实现时序逻辑电路需要外加触发器
- PLA
  - 与或阵列均可编程，实现组合逻辑
  - 实现时序逻辑电路需要外加触发器
- PAL
  - 与阵列可编程，或阵列固定
  - 不同的芯片可实现不同的逻辑，有些PAL只能实现组合逻辑电路，有些只能实现时序逻辑电路。
  - 一次性编程

PROM,PLA,PAL对比

- PROM、PLA、PAL共同存在的问题：
  - 不存在只用一种芯片，即可以实现组合逻辑电路，又可以实现时序逻辑电路
- GAL是为解决这一问题而产生的芯片
  - GAL:通用阵列逻辑（General Array Logic）

- PLA的容量表示：(输入数)×(P项数)×(输出数)
- 可编程期间工艺演化过程

可编程器件工艺演化过程

- PROM → PLA
  - 或阵列可编程
  - 与、或阵列都可编程，灵活，节省码点
- PLA → PAL
  - 工艺：简化工艺,降低成本(熔丝工艺,一次编程)
  - 结构：输入/输出公用
  - PAL是专用词，MMI公司的产品
- PAL → GAL
  - 工艺:电可擦除,多次编程.(Lattice公司1985年专利)
  - 结构：输出宏单元，更通用，
  - 或阵列不可编程

2.7 运算器（算术逻辑单元ALU）

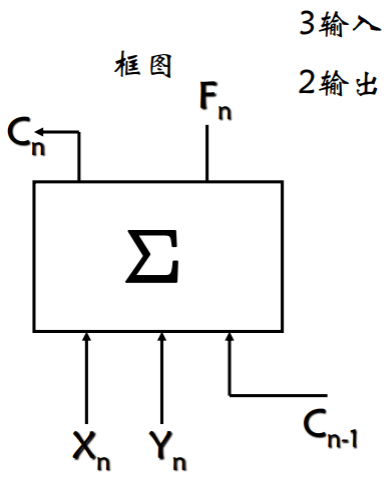
1. 加法器 Adder

采用与或非门实现：

- 三级门延迟

真值表

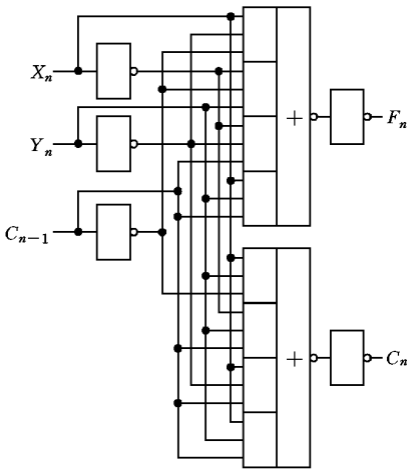
X_n	Y_n	C_{n-1}	F_n	C_n
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1



$$F_n = X_n\overline{Y_n}\overline{C_{n-1}} + \overline{X_n}Y_n\overline{C_{n-1}} + \overline{X_n}\overline{Y_n}C_{n-1} + X_nY_nC_{n-1}$$

$$C_n = X_nY_n\overline{C_{n-1}} + X_n\overline{Y_n}C_{n-1} + \overline{X_n}Y_nC_{n-1} + X_nY_nC_{n-1}$$

不化简，用全部最小项实现，需要3级门。



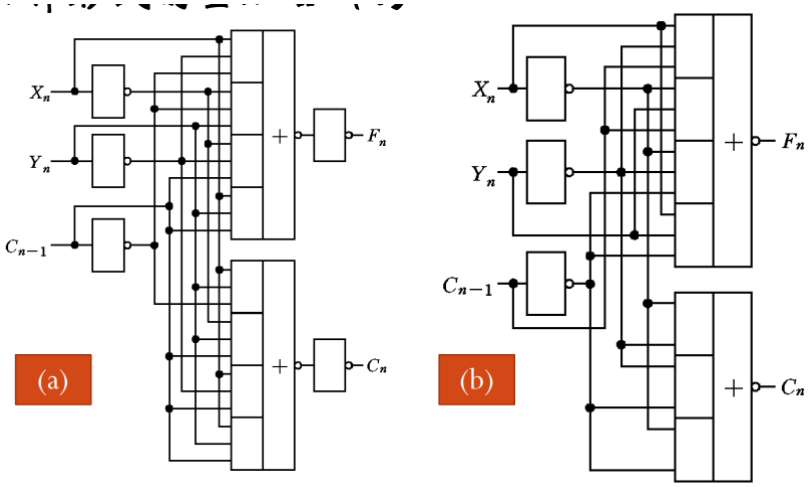
- 二级门延迟

$$\overline{F} = \overline{X_n Y_n C_{n-1}} + \overline{X_n} Y_n C_{n-1} + X_n \overline{Y_n} C_{n-1} + X_n Y_n \overline{C_{n-1}}$$

$$F = \overline{\overline{F}} = \overline{\overline{X_n Y_n C_{n-1}} + \overline{X_n} Y_n C_{n-1} + X_n \overline{Y_n} C_{n-1} + X_n Y_n \overline{C_{n-1}}}$$

$$\overline{C_n} = \overline{X_n Y_n} + \overline{X_n C_{n-1}} + \overline{Y_n C_{n-1}}$$

$$C_n = \overline{\overline{C_n}} = \overline{\overline{X_n Y_n} + \overline{X_n C_{n-1}} + \overline{Y_n C_{n-1}}}$$



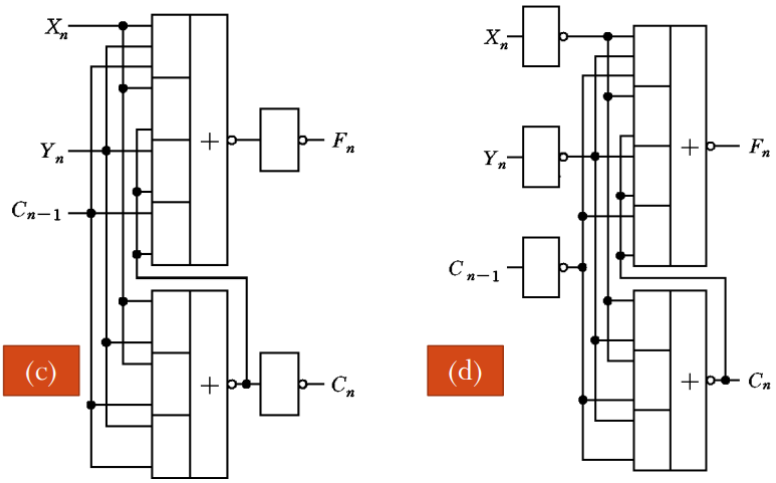
- 其它实现方式：C_n需要二级门延迟，F_n需要三级门延迟

$$F_n = X_n Y_n C_{n-1} + X_n \overline{C_n} + Y_n \overline{C_n} + C_{n-1} \overline{C_n}$$

$$C_n = X_n Y_n + (X_n + Y_n) C_{n-1}$$

$$F_n = \overline{\overline{X_n}\ \overline{Y_n}\ \overline{C_{n-1}} + \overline{X_n} C_n + \overline{Y_n} C_n + \overline{C_{n-1}} C_n}$$

$$C_n = \overline{\overline{X_n}\ \overline{Y_n} + (\overline{X_n} + \overline{Y_n}) \overline{C_{n-1}}}$$



> 组合使用(c)和(d)可以增加进位的效率

半加器 Half Adder：不考虑低位进位输入和向高位的进位输出，两数码X_n和Y_n相加。

一位全加器 Full Adder

真值表达式：

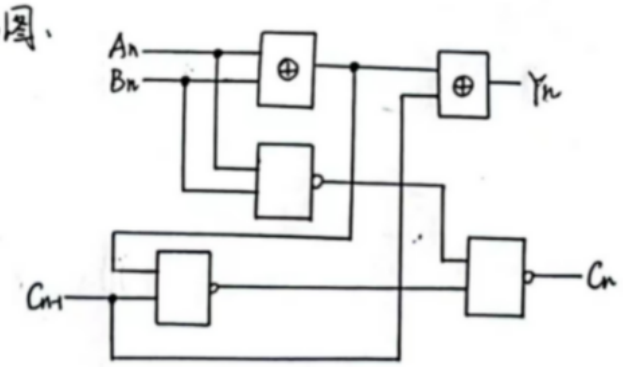
A_n	B_n	C_{n-1}	Y_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

逻辑表达式：

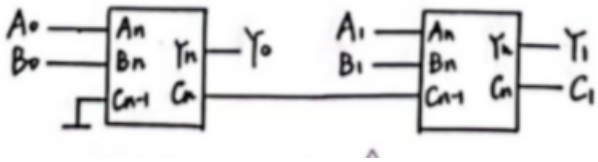
$$Y_n = A_n \oplus B_n \oplus C_{n-1}$$

$$C_n = A_nB_n + C_{n-1}(A_n \oplus B_n) = \overline{\overline{A_nB_n} \cdot \overline{C_{n-1}(A_n \oplus B_n)}}$$

电路图：



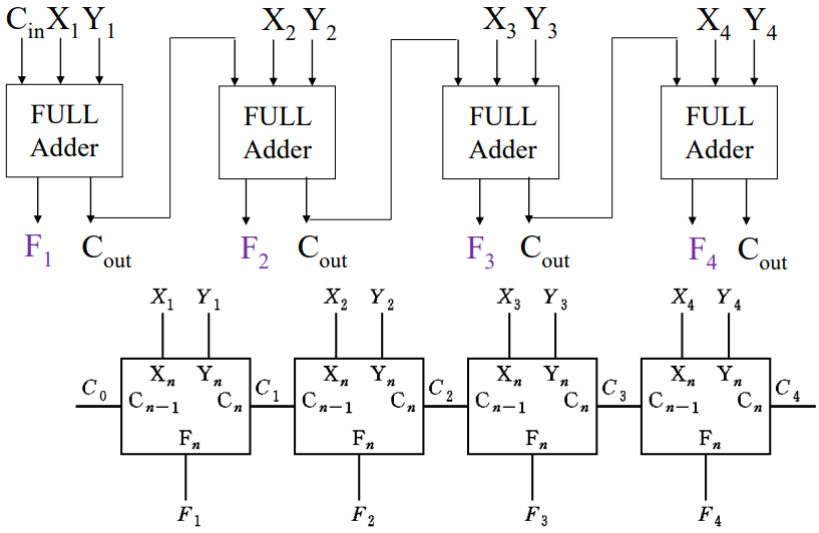
二位全加器 Full Adder



1. 四位串行进位加法器（Ripple Adder）

   计算速度受到进位的传递瓶颈。

   - 使用上述(b)型加法器需要8级门延迟



   - 使用上述(c)型和(d)型交叉串联：F_4需要5级门延迟，C_4需要4级门延迟

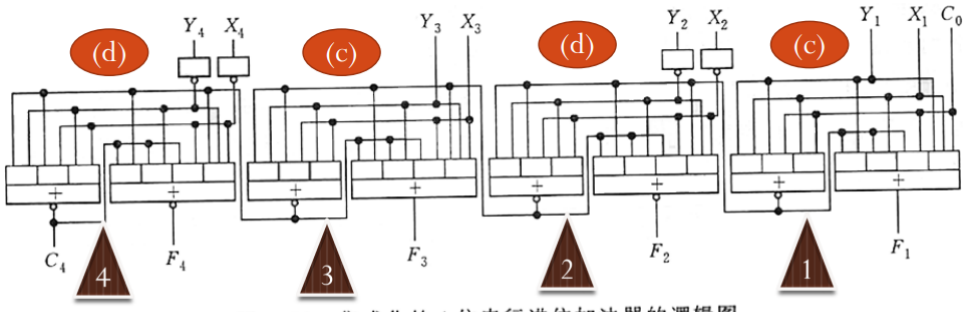


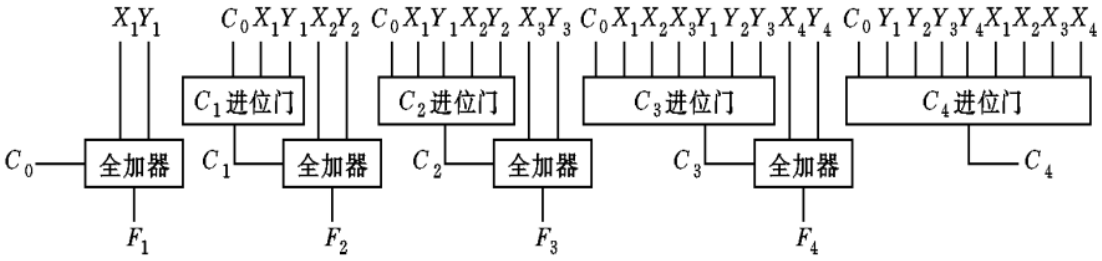
图 4-51　集成化的 4 位串行进位加法器的逻辑图

- F_1需要3级，C_1需要2级，F_2需要3级，C_2需要2级，
- F_3需要5级，C_3需要4级，F_4需要5级，C_4需要4级。

21

2. 四位并行进位加法器（Look-ahead Adder）

> 进位输入是由专门的“进位门”总和所有低位的加数、被加数及最低位进入输入后提供的
>
> 又称超前进位加法器或快速加法器



$$C_1 = X_1Y_1 + (X_1 + Y_1)C_0$$

$$C_2 = X_2Y_2 + (X_2 + Y_2)X_1Y_1 + (X_2 + Y_2)(X_1 + Y_1)C_0$$

$$C_3 = X_3Y_3 + (X_3 + Y_3)X_2Y_2 + (X_3 + Y_3)(X_2 + Y_2)X_1Y_1 + (X_3 + Y_3)(X_2 + Y_2)(X_1 + Y_1)C_0$$

$$C_4 = X_4Y_4 + (X_4 + Y_4)X_3Y_3 + (X_4 + Y_4)(X_3 + Y_3)X_2Y_2 + (X_4 + Y_4)(X_3 + Y_3)(X_2 + Y_2)X_1Y_1 + (X_4 + Y_4)(X_3 + Y_3)(X_2 + Y_2)(X_1 + Y_1)C_0$$

$G_i = X_iY_i$ 称为产生进位函数

$P_i = X_i + Y_i$ 称为传递进位函数

$$C_1 = X_1Y_1 + (X_1 + Y_1)C_0 = G_1 + P_1C_0$$

化简：

$$\overline{C_1} = (\overline{X_1} + \overline{Y_1})(\overline{X_1}\,\overline{Y_1} + \overline{C_0}) = (\overline{X_1} + \overline{Y_1})\overline{C_0} + \overline{X_1}\,\overline{Y_1}$$

得到$C_1 = \overline{\overline{X_1}\,\overline{Y_1} + (\overline{X_1} + \overline{Y_1})\overline{C_0}}$

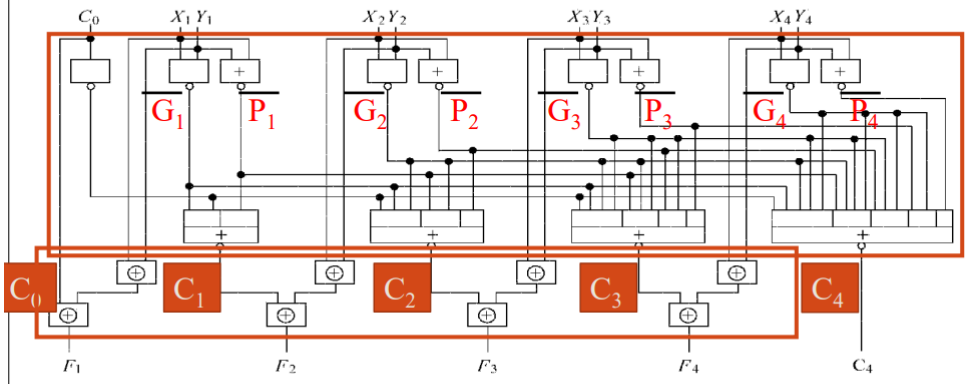
改写为：$C_i = \overline{\overline{X_1 + Y_1} + \overline{X_1Y_1}\,\overline{C_0}} = \overline{\overline{P_1} + \overline{G_1}\,\overline{C_o}}$

$$C_1 = \overline{\overline{X_1 + Y_1 + X_1Y_1C_0}} = \overline{\overline{P_1 + G_1C_0}}$$

$$C_2 = \overline{\overline{X_2 + Y_2 + X_2Y_2(X_1 + Y_1) + X_2Y_2X_1Y_1C_0}} = \overline{\overline{P_2 + G_2P_1 + G_2G_1C_0}}$$

$$C_3 = \overline{\overline{X_3 + Y_3 + X_3Y_3(X_2 + Y_2) + X_3Y_3X_2Y_2(X_1 + Y_1) + X_3Y_3X_2Y_2X_1Y_1C_0}}$$

$$C_4 = \overline{\overline{X_4 + Y_4 + X_4Y_4(X_3 + Y_3) + X_4Y_4X_3Y_3(X_2 + Y_2) + X_4Y_4X_3Y_3X_2Y_2(X_1 + Y_1) + X_4Y_4X_3Y_3X_2Y_2X_1Y_1C_0}}$$



30 C_i延迟级数与位数无关：都是2级；F_i基本都是3级

C_i延迟级数与位数无关：都是2级
F_i延迟级数基本都是3级

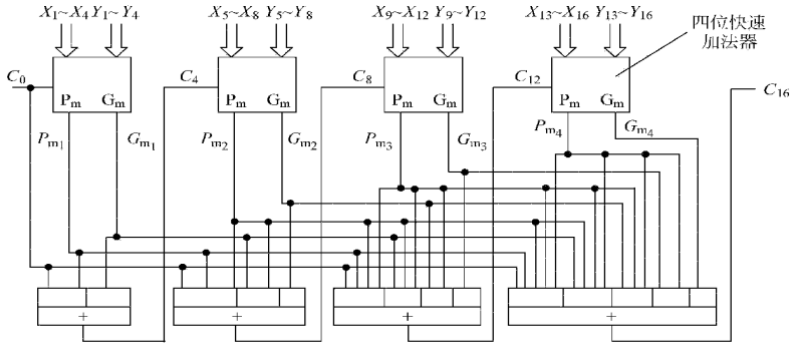
3. 16位加法器
   - 16为串行加法器

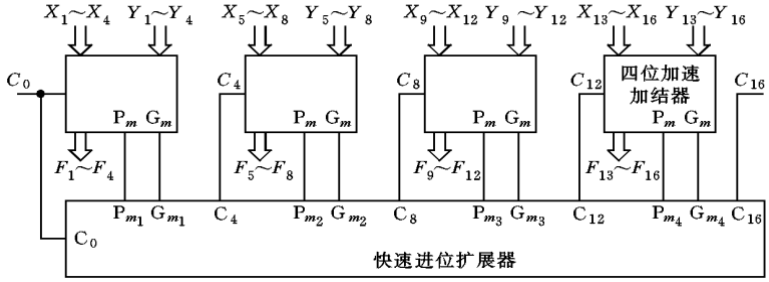
     > 用4片四位并行加法器串行组成16为快速加法器

     此时，C_{16}为8级门延迟，F_{13}~F_{16}需要9级门延迟，$C_n = \frac{n}{2}$级
   - 16为并行加法器

     > 用类似四位并行加法器中C_1、C_2、C_3、C_4形成的原理，去形成片间快速进位C_4、C_8、C_{12}、C_{16}

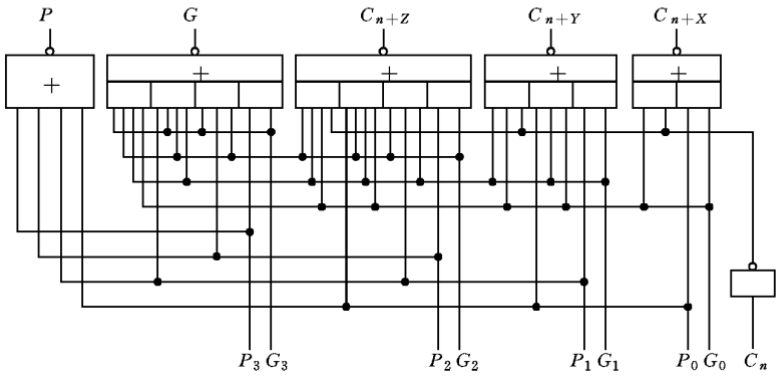
     超前进位扩展器：使得C_4、C_8、C_{12}、C_{16}同时产生



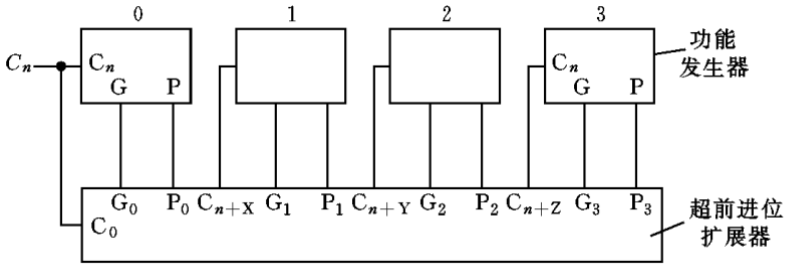


四位并行加法器的输出提供P_m、G_m需2级延迟。
产生C_4，C_8，C_{12}，C_{16}的延迟3级，F_1 ~ F_4要3级，F_5 ~ F_{16}要6级。

和4位加法器（或4位ALU）连用的超前进位扩展器专用器件(SN74182)（P146）

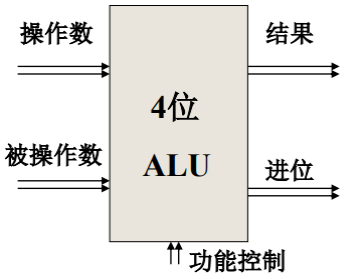


4位ALU和超前进位扩展器组成16位快速运算单元（P146）



2. 算术运算逻辑单元（ALU）

> 以4为算术逻辑运算单元为例



4位ALU的实现

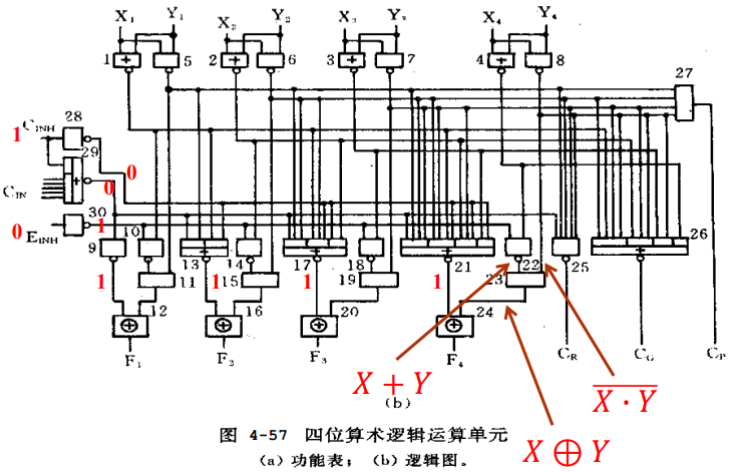
1. 控制4位加法器中的进位逻辑，获得多种运算能力

    - 简单、运算种类较少
    - 在控制信号C_{INH}、E_{INH}的作用下，可以完成4位数的加、比较和逻辑乘（与） 运算

功能表

C_{INH}	E_{INH}	功　能
0	0	加
0	1	不用
1	0	$\overline{X_n \oplus Y_n}$
1	1	$X_n \cdot Y_n$

(a)



(b)

图 4-57　四位算术逻辑运算单元

(a) 功能表；(b) 逻辑图。

电路1: 功能控制加在进位门和半加器

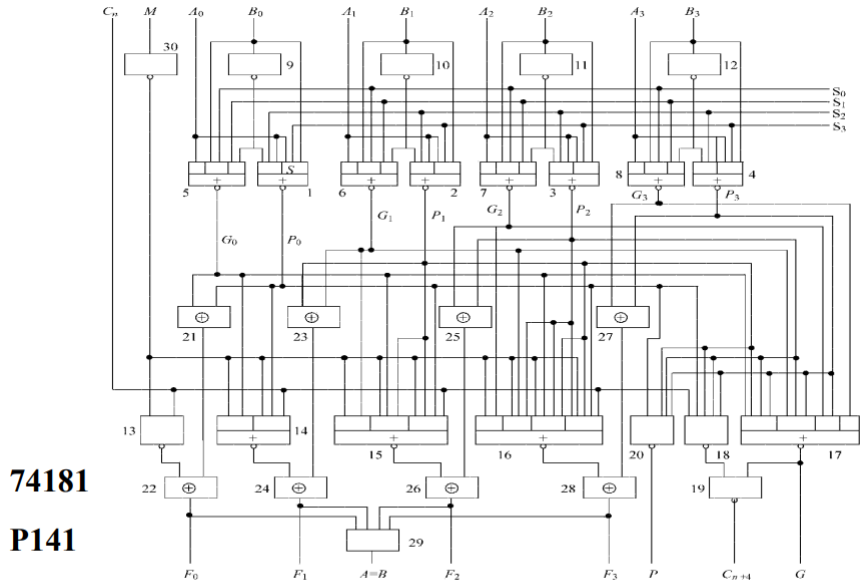
2. 改变加法器中的G_i和P_i来获得多种运算能力

- 运算种类多，但运算单元结构较复杂
- 能执行16中算术运算和16种逻辑运算

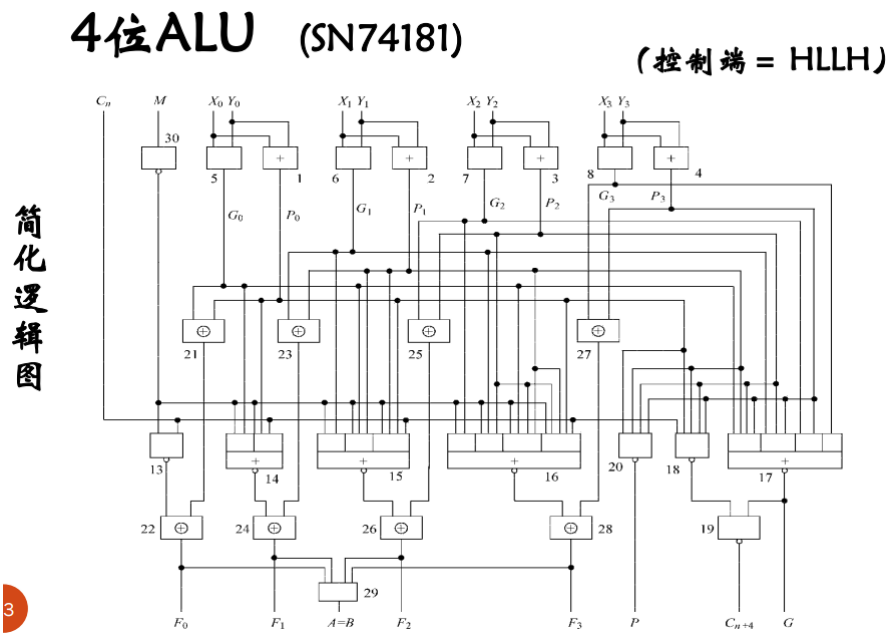
4位ALU功能表 (SN74181)

M是状态控制端，M＝”H”，执行逻辑运算，反之，执行算术运算。

S_3 S_2 S_1 S_0	正逻辑 M=H 逻辑运算	正逻辑 M=L C_n=1	正逻辑 M=L C_n=0	负逻辑 M=H 逻辑运算	负逻辑 M=L C_n=1	负逻辑 M=L C_n=0
L L L L	$\overline{A}$	A	A 加 1	$\overline{A}$	A 减 1	A
L L L H	$\overline{A+B}$	$A+B$	$(A+B)$ 加 1	$\overline{A \cdot B}$	$(A \cdot B)$ 减 1	$A \cdot B$
L L H L	$\overline{A} \cdot B$	$A+\overline{B}$	$(A+\overline{B})$ 加 1	$\overline{A}+B$	$(A \cdot \overline{B})$ 减 1	$A \cdot \overline{B}$
L L H H	“0”	减 1	0	1	减 1	0
L H L L	$\overline{A \cdot B}$	A 加 $(A \cdot \overline{B})$	A 加 $(A \cdot \overline{B})$ 加 1	$\overline{A+B}$	A 加 $(A+\overline{B})$	A 加 $(A+\overline{B}$ 加 1
L H L H	$\overline{B}$	$(A \cdot \overline{B})$ 加 $(A+B)$	$(A \cdot \overline{B})$ 加 $(A+B)$ 加 1	$\overline{B}$	$(A \cdot \overline{B})$ 加 $(A+\overline{B})$	$(A \cdot \overline{B})$ 加 $(A+\overline{B})$ 加 1
L H H L	$A \oplus B$	A 减 B 减 1	A 减 B	$\overline{A \oplus B}$	A 减 B 减 1	A 减 B
L H H H	$A \cdot \overline{B}$	$(A \cdot \overline{B})$ 减 1	$A \cdot \overline{B}$	$A+\overline{B}$	$A+\overline{B}$	$(A+\overline{B})$ 加 1
H L L L	$\overline{A}+B$	A 加 $(A \cdot B)$	A 加 $(A \cdot B)$ 加 1	$\overline{A} \cdot B$	A 加 $(A+B)$	A 加 $(A+B)$ 加 1
H L L H	$\overline{A \oplus B}$	A 加 B	A 加 B 加 1	$A \oplus B$	A 加 B	A 加 B 加 1
H L H L	B	$(A \cdot \overline{B})$ 加 $(A+B)$	$(A \cdot \overline{B})$ 加 $(A+B)$ 加 1	B	$(A \cdot \overline{B})$ 加 $(A+B)$	$(A \cdot \overline{B})$ 加 $(A+B)$ 加 1
H L H H	$A \cdot B$	$(A \cdot B)$ 减 1	$A \cdot B$	$A+B$	$A+B$	$(A+B)$ 加 1
H H L L	1	A 加 A	A 加 A 加 1	0	A 加 A	A 加 A 加 1
H H L H	$A+\overline{B}$	A 加 $(A+B)$	A 加 $(A+B)$ 加 1	$A \cdot \overline{B}$	A 加 $(A \cdot B)$	A 加 $(A \cdot B)$ 加 1
H H H L	$A+B$	A 加 $(A+\overline{B})$	A 加 $(A+\overline{B})$ 加 1	$A \cdot B$	A 加 $(A \cdot B)$	A 加 $(A \cdot B)$ 加 1
H H H H	A	A 减 1	A	A	A	A 加 1



电路2: 功能控制加在输入端,改变进位产生函数G_i和进位传递函数P_i



4位ALU的功能

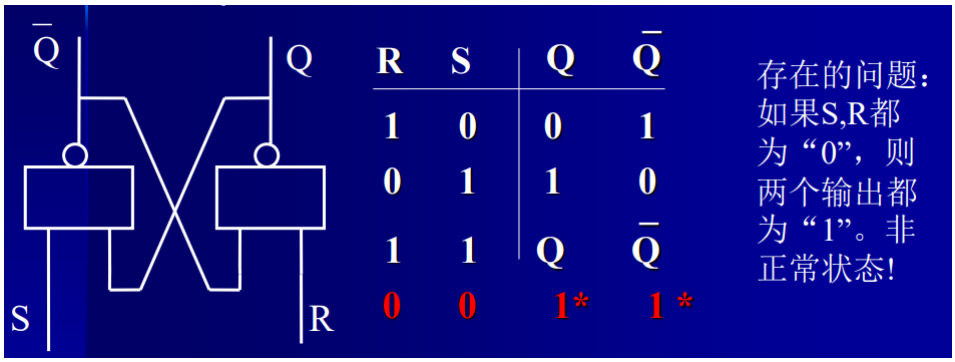
3 同步时序电路

组合逻辑电路某一时刻的输出只取决于此时刻的输入。
时序逻辑电路某一时刻的稳定输出不仅取决于当时的输入，还取决于过去的输入（历史状态）。

记忆元件（Memory Devices）是时序逻辑电路的基本原件
计算机中的实现记忆存储功能的元件有多种：磁存储、 光存储、半导体存储（电存储）
时序逻辑电路中的记忆元件一定要是可以修改的，也可以控制的，这种元件称为**“触发器”**。

1. 反馈：把输出接到输入
    - 与门的“反馈功能”：一旦输入为`0`，输出永远为`0`；可以记住“过去曾经”输入为`0`
    - 或门的”反馈功能”：一旦输入为`1`，输出永远为`1`；可以记住“过去曾经”输入为`1`

2. 交叉耦合电路——**RS基本触发器**
   可以任意修改，输出有正负两种状态；当Input撤除时，触发器保持状态（记忆功能）
   `R`表示Reset，`S`表示Set，RS基本触发器可以设置一个比特的状态



> `R = 1, S = 1`：保持（存）原来的状态
> `R, S = {1, 0}`：写入（写）状态

3.1 触发器

定义：能存储一位二进制数的记忆元件，英文缩写FF (Flip-Flop)

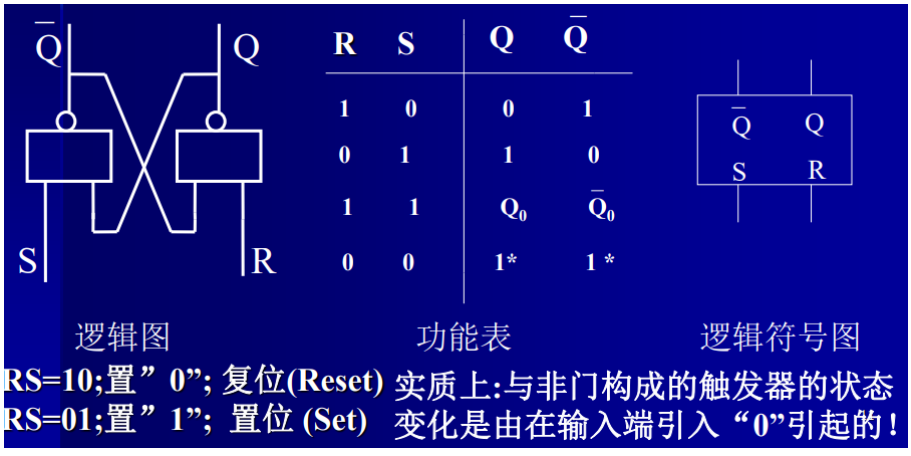
分类：
（*）重点 = 边沿触发方式的D触发器 + JK触发器

- 按时钟（Clock Pulse）控制方式分类：
  电位触发方式FF（Level Trigger）
  边沿触发方式FF（Edge-Trigger）
  主-从触发方式FF（Master-Slave 或 Pulse-Trigger）
- 按功能来分类：
  D触发器（Delay）
  R-S触发器（Set-Reset）
  J-K触发器
  T触发器（Toggle）

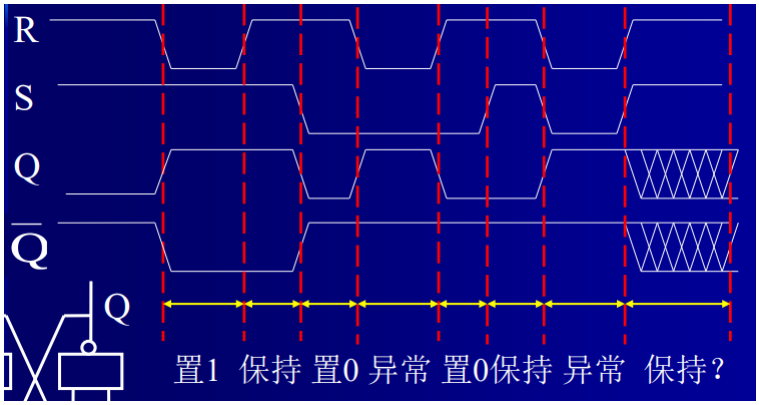
1. 基本R-S触发器

> 与非门构成的“直接置位-复位型R-S触发器”
>
> （*）逻辑符号图：从下往上，从左往右 = 输入到输出

有两个稳定的状态，可以存储一位二进制数，因此叫**“双稳态”（Bi-stable）触发器**



- **R-S基本触发器时序图（Timing Diagram）**



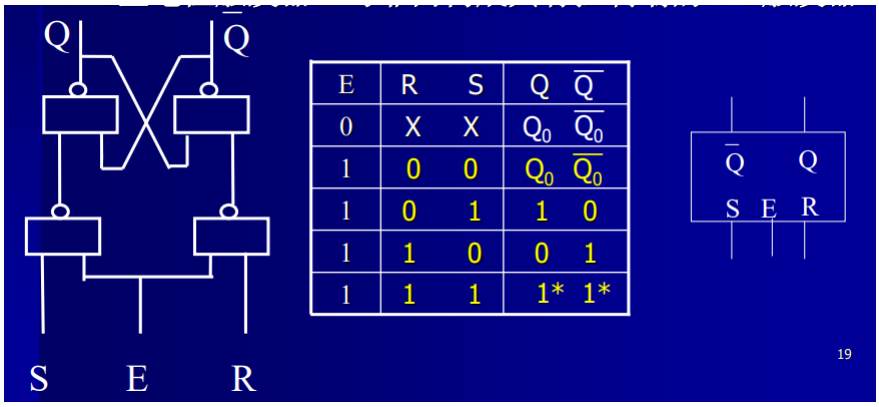
RS由`00`变为`11`，下一状态不定（取决于`S`和`R`对应门延迟的大小），因此称之为“异常”
由或非门和与或非门组成的R-S基本触发器同样存在这一问题

2. 电位触发方式的触发器

> 电位触发方式FF（Level Trigger）

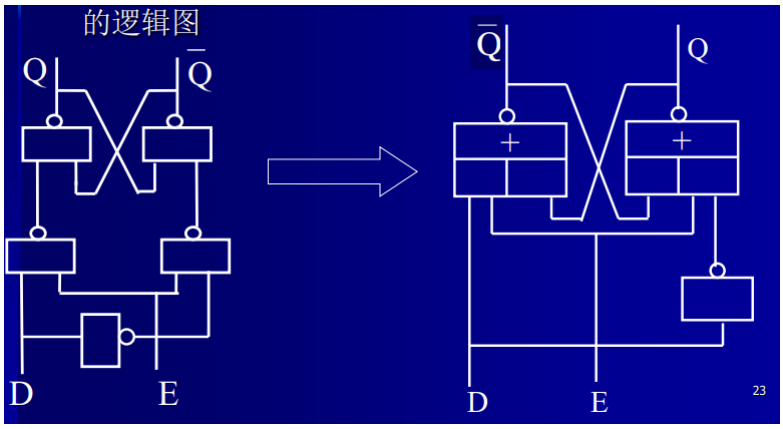
- **与非门构成具有控制端的R-S触发器**

  当触发器的同步控制信号`E`为`1`时，触发器接收数据，输入数据的变化会在输出端得到反映
  当`E`为`0`时，触发器的状态保持不变
  （*）采用与非门实现`E`（而不是或门）是为了保持结构的一致性（*R-S*基本触发器由与非门构成）



问题：`E=1，R-S="11"`的时候，同样存在不定状态

- **输入由`R，S`双端输入改为单端输入：电位型D触发器**
  控制端控制状态保持，输入端保证为`01`或`10`
  电位触发方式：在控制电位`E`的控制下接收数据（将`E`的地方加反相器可以更改约定状态）

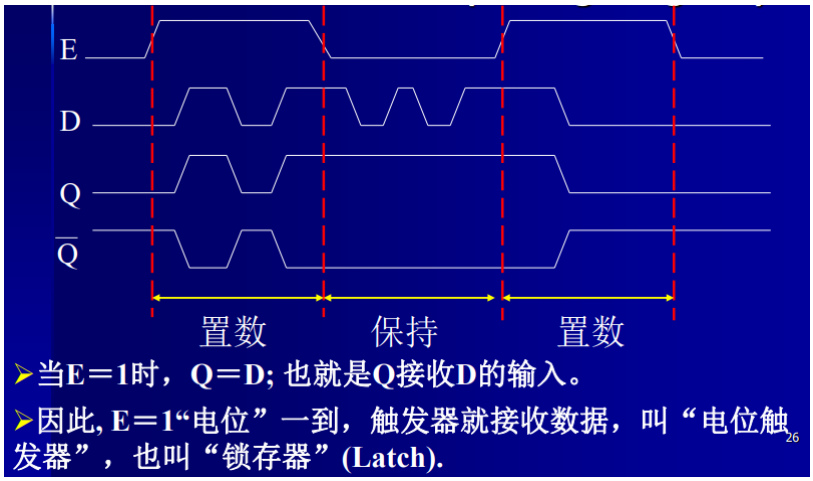


E	D	Q	$\overline{Q}$
1	D	D	$\overline{D}$
0	X	Q_0	$\overline{Q_0}$

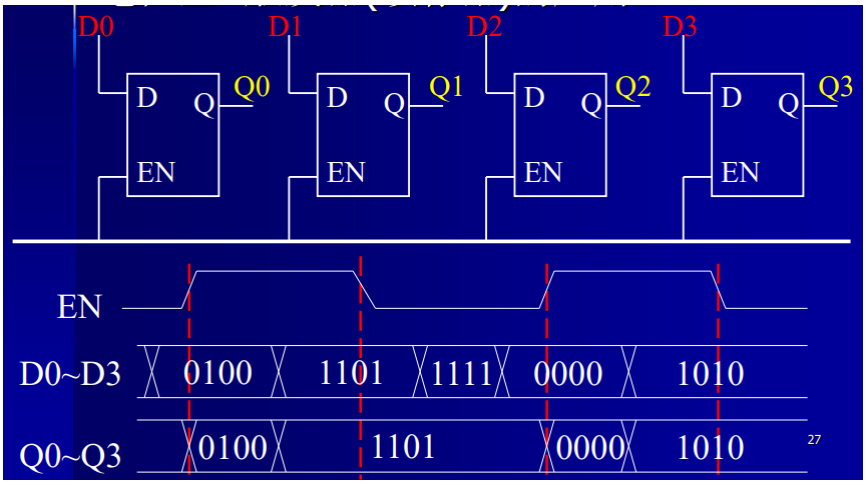
E=1，D以互补形式进入，Store（置数）

E=0，D被封锁，Hold（保持）

- 电位型D触发器时序图（Timing Diagram）

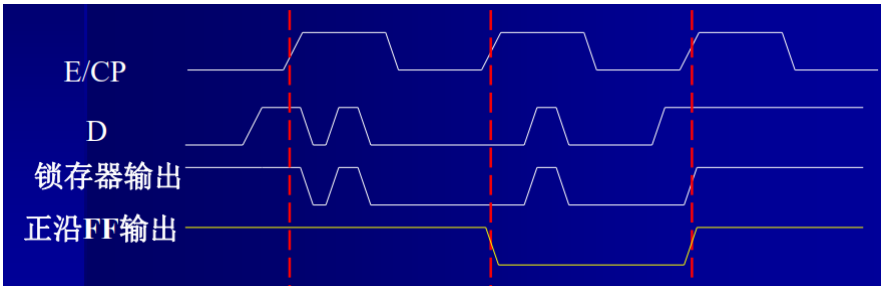


- 电位型D触发器（锁存器）的应用



电位型D触发器（锁存器）存在的问题：

1. 抗干扰能力较差
2. 期望：接收使能（时钟）脉冲某一跳变来到时，输出才变化为输入的值
   时钟周期 = 上升沿 + 高电平 + 下降沿 + 低电平



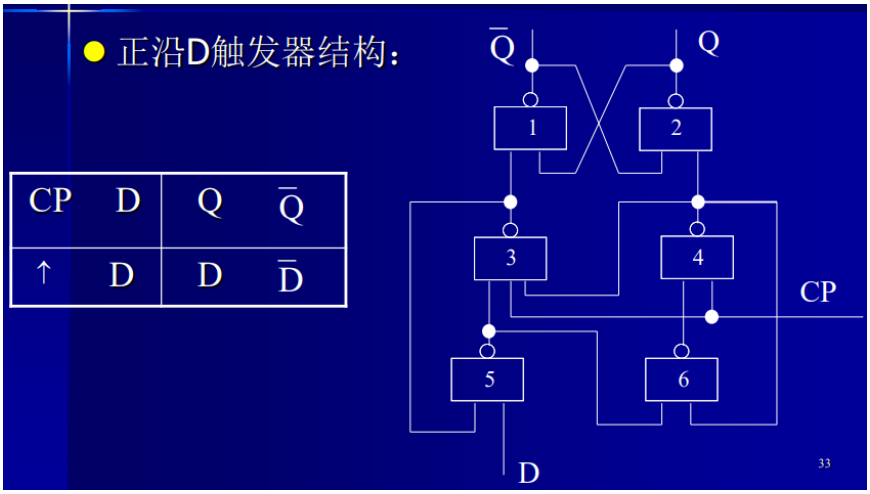
3. 边沿触发方式的触发器（正边沿D触发器）

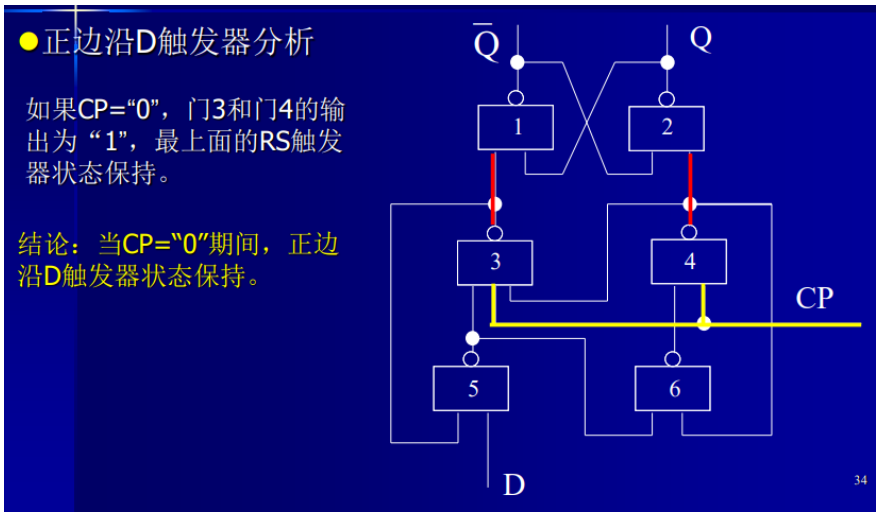
> 正边沿：时钟周期中的上升沿

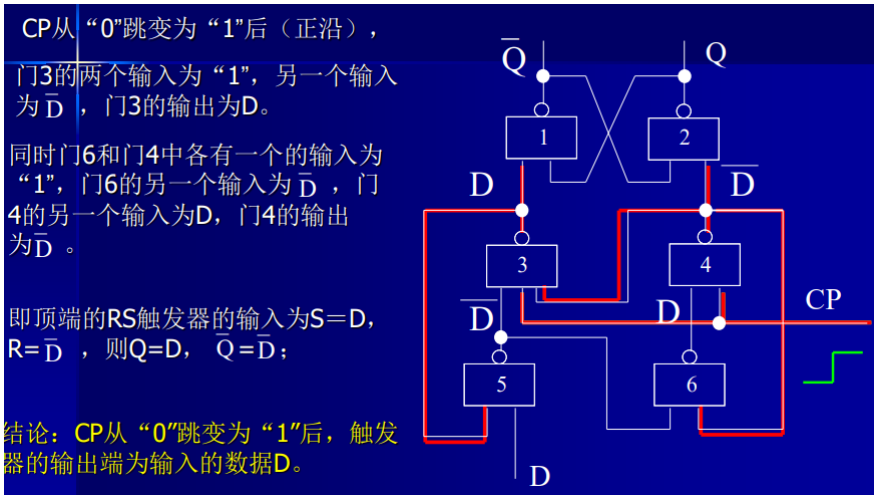
- **边沿触发方式FF（Edge-Trigger）**

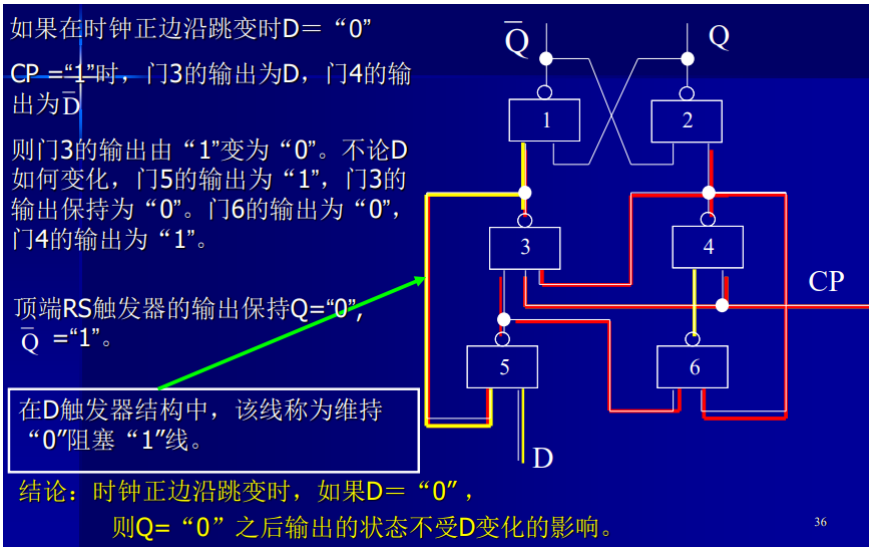
  触发器只有在时钟输入`CP`的某一约定跳变（正跳变或负跳变）到来时，才接收输入数据
  当`CP`没有跳变（`CP=0`或`CP=1`）期间，输入数据的变化不会引起触发器输出状态的变化
  当`CP`的非约定跳变到来时，触发器也不会接收输入数据

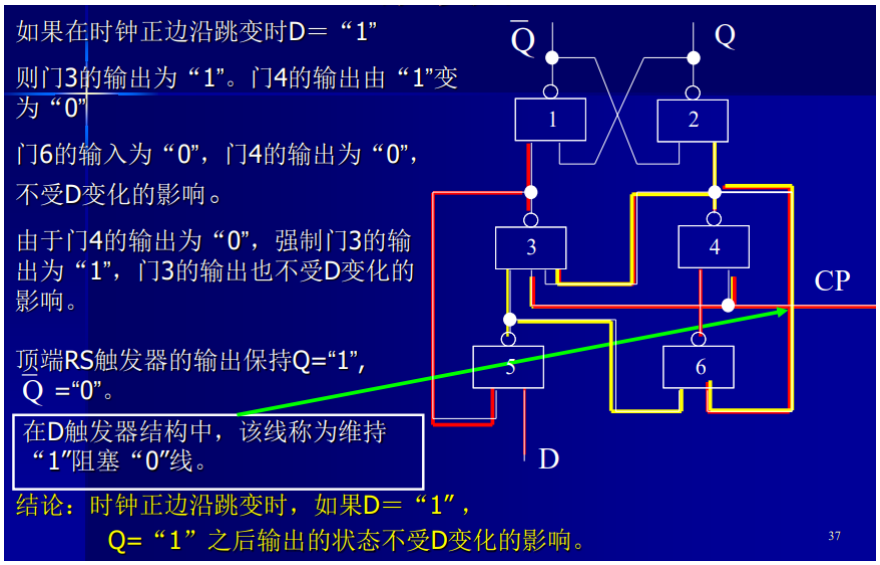
  > 抗干扰的核心逻辑：**维持“0”阻塞“1”线，维持“1”阻塞“0”线**

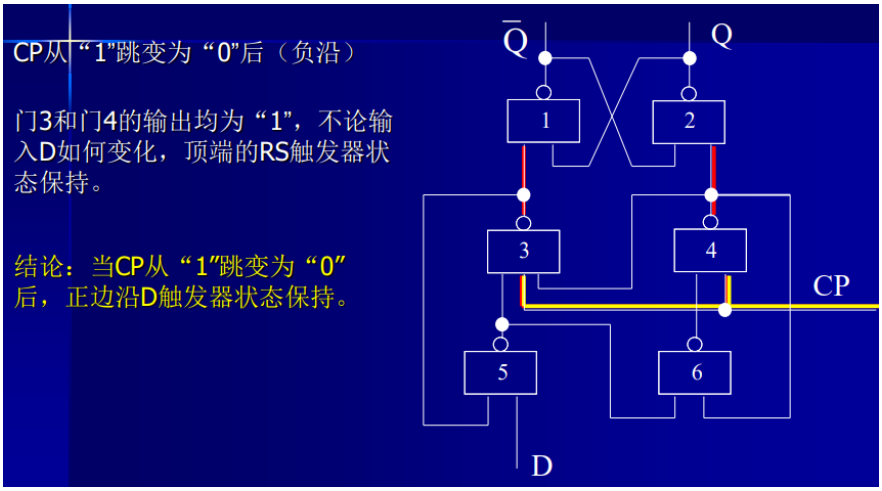












- **正沿D触发器的开关特性**
  - **数据建立时间t_{su}（set up）**：在时钟的上升沿之前多长时间必须给出数，
    $t_{su} \geq$ 5门延迟 + 6门延迟（两级门延迟）
    必须在时钟的低电平时间！
  - **数据保持时间t_h（hold）**：数来了之后需要保持的时间，
    $t_h \geq MAX(t_{pd3}, t_{pd4})$（一级门延迟）
    必须在时钟的高电平时间！

- **传输延迟参数**t_{pd}：time propagation delay

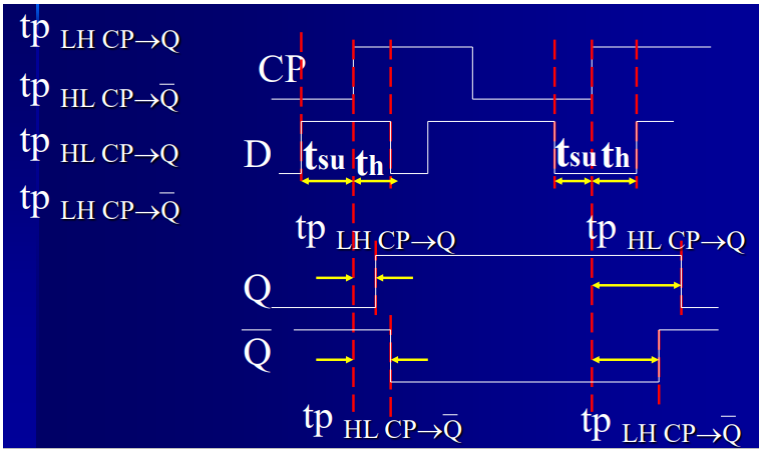
  $t_{pLH:CP\to Q}$：约定的时钟到来时Q从低电平（L）到高电平（H）跳变的时间

  $t_{pHL:CP\to \overline{Q}}$：约定的时钟到来时$\overline{Q}$从高电平（H）到低电平（L）跳变的时间

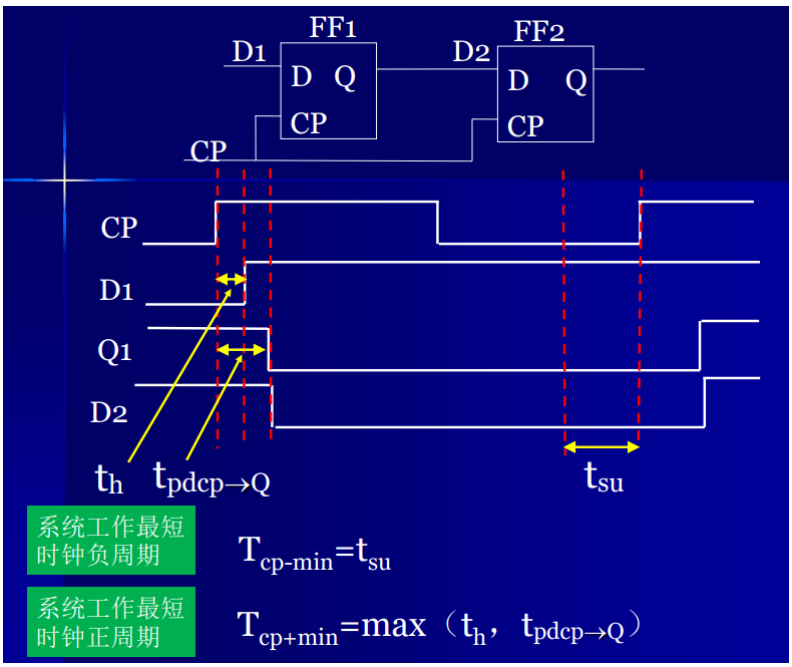
  $t_{pHL:CP\to Q}$：约定的时钟到来时Q从高电平（H）到低电平（L）跳变的时间

  $t_{pLH:CP\to \overline{Q}}$：约定的时钟到来时Q从低电平（L）到高电平（H）跳变的时间

  $t_{pd:CP\to Q}$必须在时钟的高电平时间！



- **多个触发器组成的电路最高时钟频率**

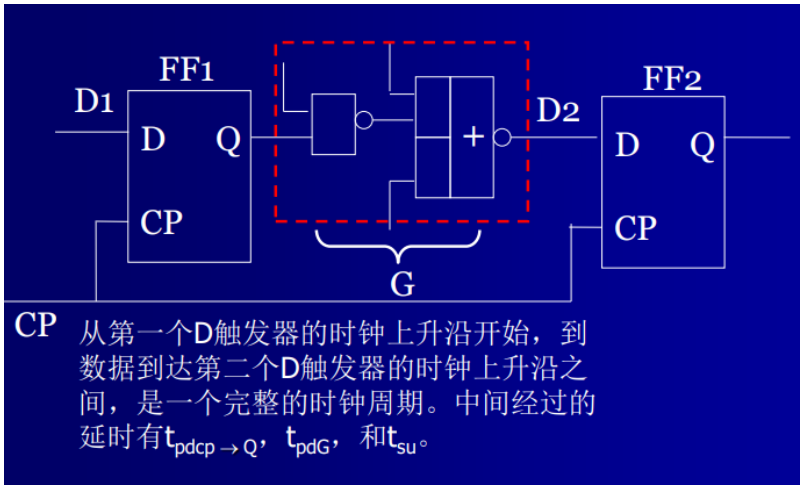


  系统工作最短时钟负周期：$T_{cp\_min}=t_{su}$

  系统工作最短时钟正周期：$T_{cp\_min}=max(t_h,t_{pdcp\to Q})$

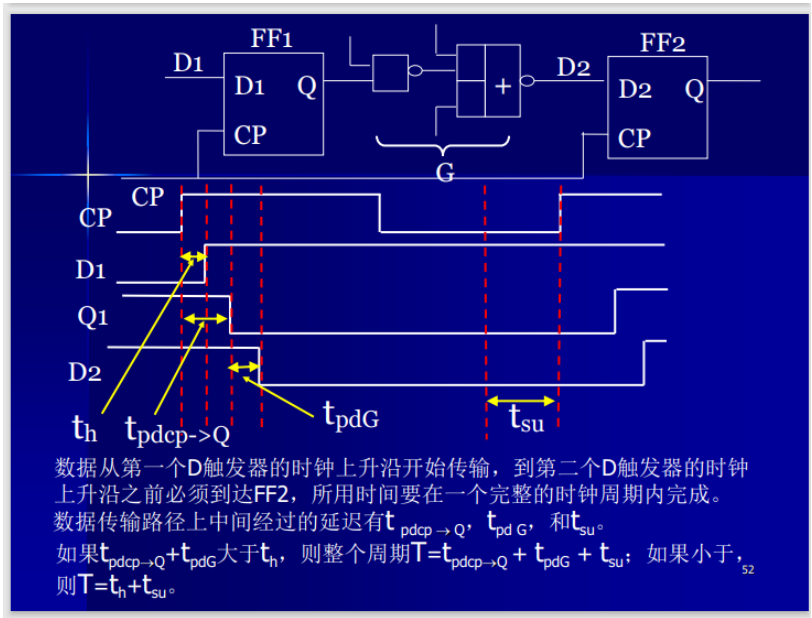
  系统工作最快时钟频率：$f_{cp\_max}=\frac{1}{t_{su}+max(t_h,t_{pdcp\to Q})}$

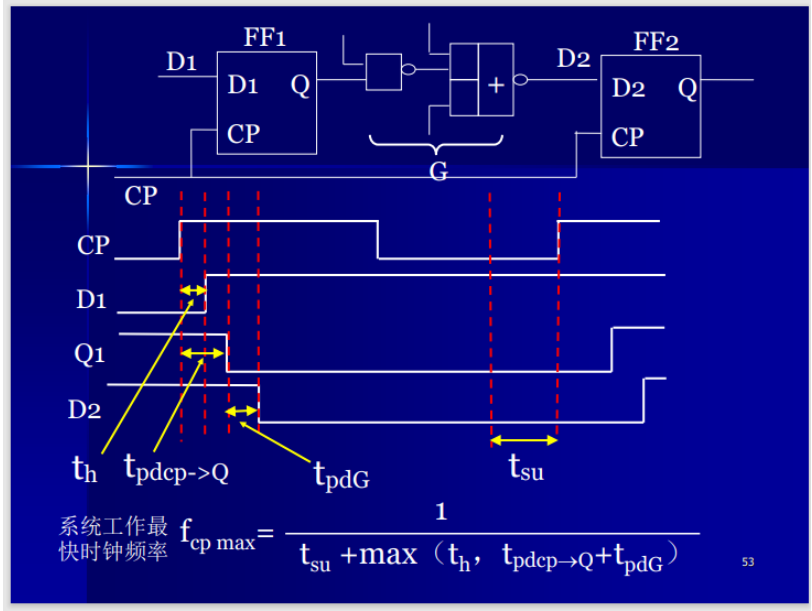
- **多个触发器组成的电路最高时钟频率**



分析：

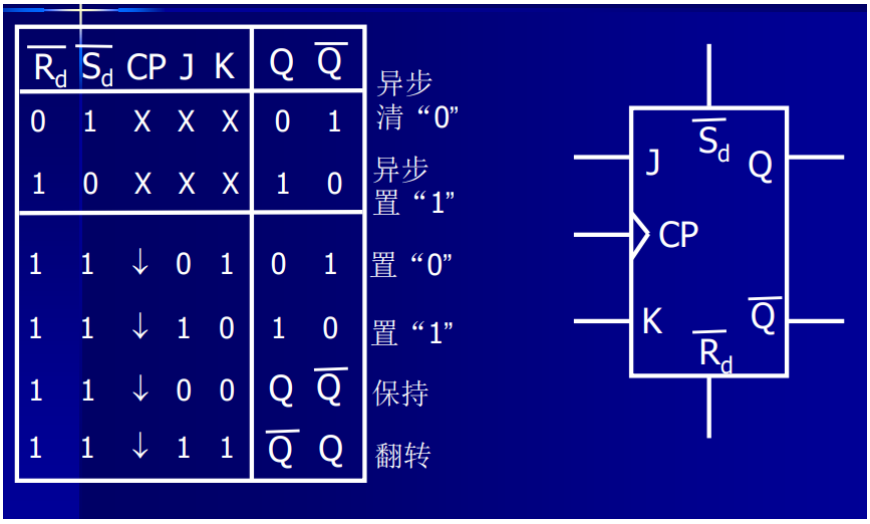
如下图分析，可以想办法把t_{pdG}分配在高/低电平，这样可以使得取max的时候不占时间，下图中分配在了高电平，因此最快时钟频率表达式中没有t_{su}



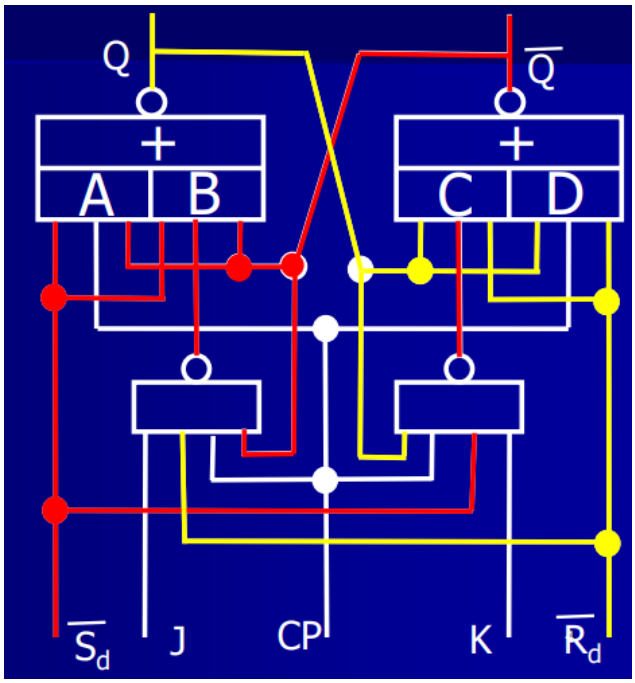


4. 负边沿JK触发器

该结构的基本要求：$t_{pd与非门} > 2\times t_{pd与或非门}$，$t_{set\_up} > t_{pd与非门}$

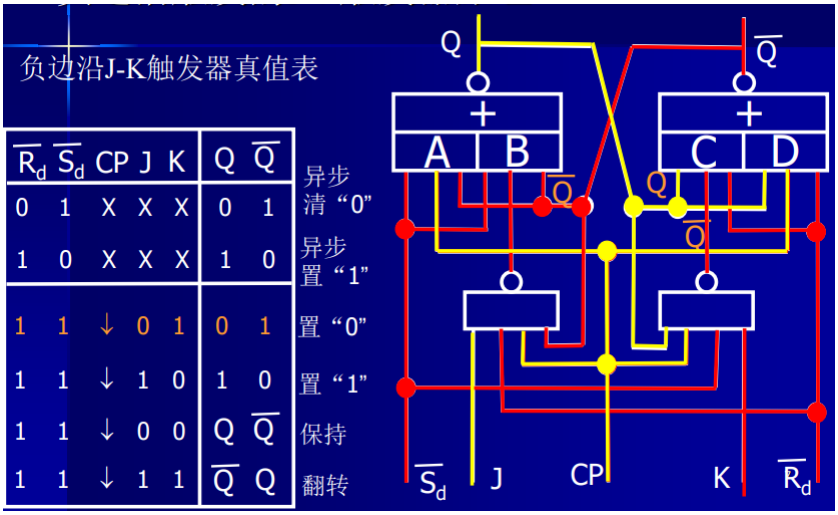


异步：跟时钟没有关系

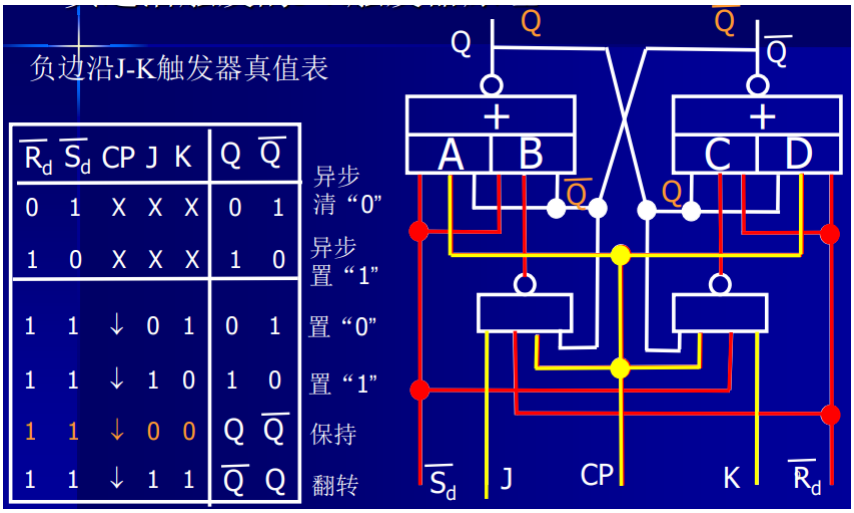


该图为$\overline{R_d}=0,\overline{S_d}=1,CP=J=K=X,Q=0,\overline{Q}=1$的情形，红色为高电平*1*，黄色为低电平*0*，另一个异步情况同理

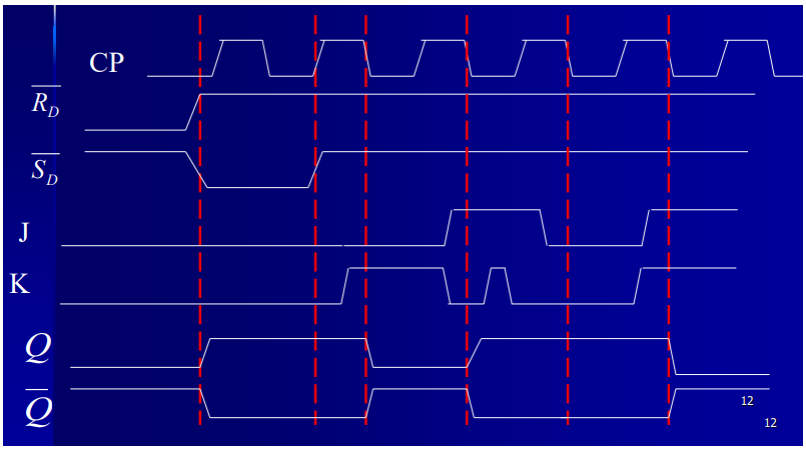
1. $\overline{R_d}=1,\overline{S_d}=1,CP=\downarrow,J=0,K=1,Q=0,\overline{Q}=1$：置`0`；置`1`的情况类似
   此时，$t_{hold}=0$即可



2. $\overline{R_d}=1,\overline{S_d}=1,CP=\downarrow,J=0,K=0,Q=Q_0,\overline{Q}=\overline{Q_0}$：保持；翻转情况类似



3. 时序图



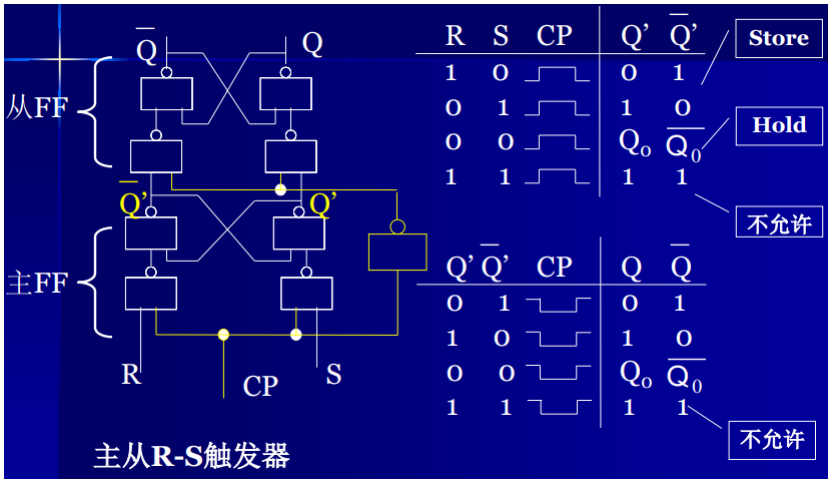
5. 主-从触发方式的触发器

由两级电位触发器（主触发器和从触发器）串连而成
`CP=1`期间，主触发器接收数据，从触发器封锁
在负跳变到来时，主触发器封锁，从触发器将接收CP负跳变时主触发器的状态。

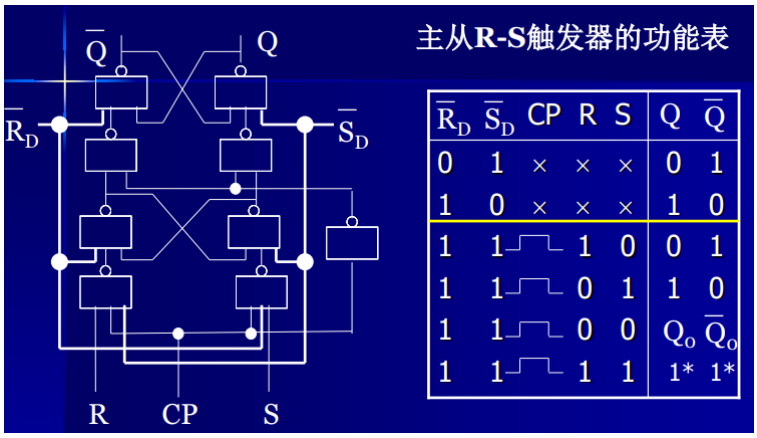
常用的有：主从R-S触发器，主从J-K触发器

> U盘的四根针：VCC, GND, TX（读）, RX（写）——只能用电位触发，没有时钟

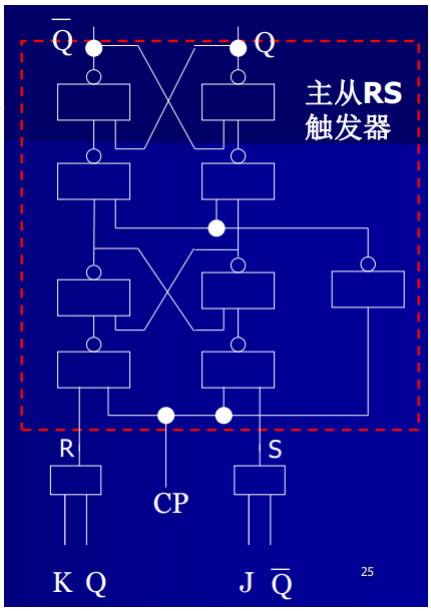
- **主从R-S触发器**
  由两个R-S电位触发器组成
  正脉冲期间主触发器接受输入，从触发器关闭
  负脉冲期间主触发器状态打入从触发器，从而保证稳定接受

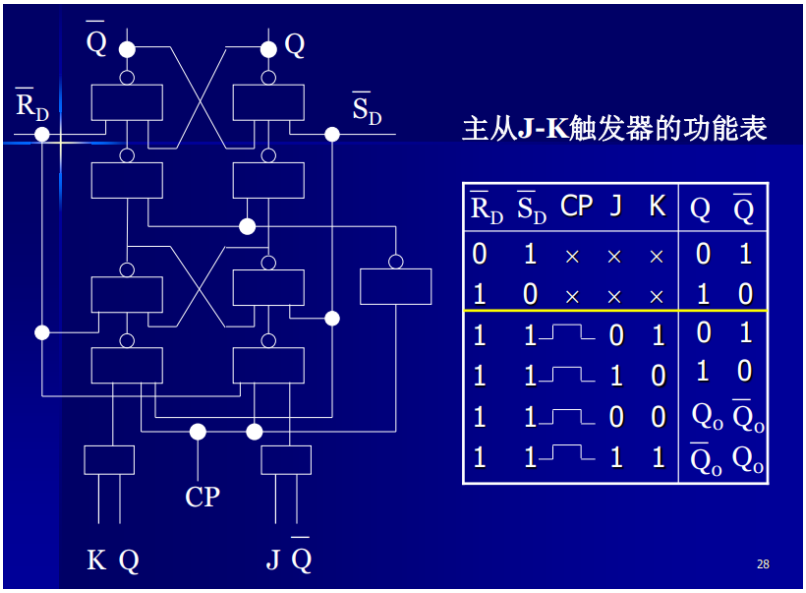


主从R-S触发器异步置位/复位端：



- **主从J-K触发器**
  将Q和Q非馈接到输入
  在正脉冲期间，主触发器接收是JQ，KQ
  负脉冲期间，主触发器状态打入从触发器，又称“脉冲触发器”





主从J-K触发器的功能表

$\overline{R}_D$	$\overline{S}_D$	CP	J	K	Q	$\overline{Q}$
0	1	×	×	×	0	1
1	0	×	×	×	1	0
1	1	⊓	0	1	0	1
1	1	⊓	1	0	1	0
1	1	⊓	0	0	Q_o	$\overline{Q}_o$
1	1	⊓	1	1	$\overline{Q}_o$	Q_o

主从R-S触发器的功能表

$\overline{R}_D$	$\overline{S}_D$	CP	R	S	Q	$\overline{Q}$
0	1	×	×	×	0	1
1	0	×	×	×	1	0
1	1	⊓	1	0	0	1
1	1	⊓	0	1	1	0
1	1	⊓	0	0	Q_o	$\overline{Q}_o$
1	1	⊓	1	1	1*	1*

主从J-K触发器的功能表

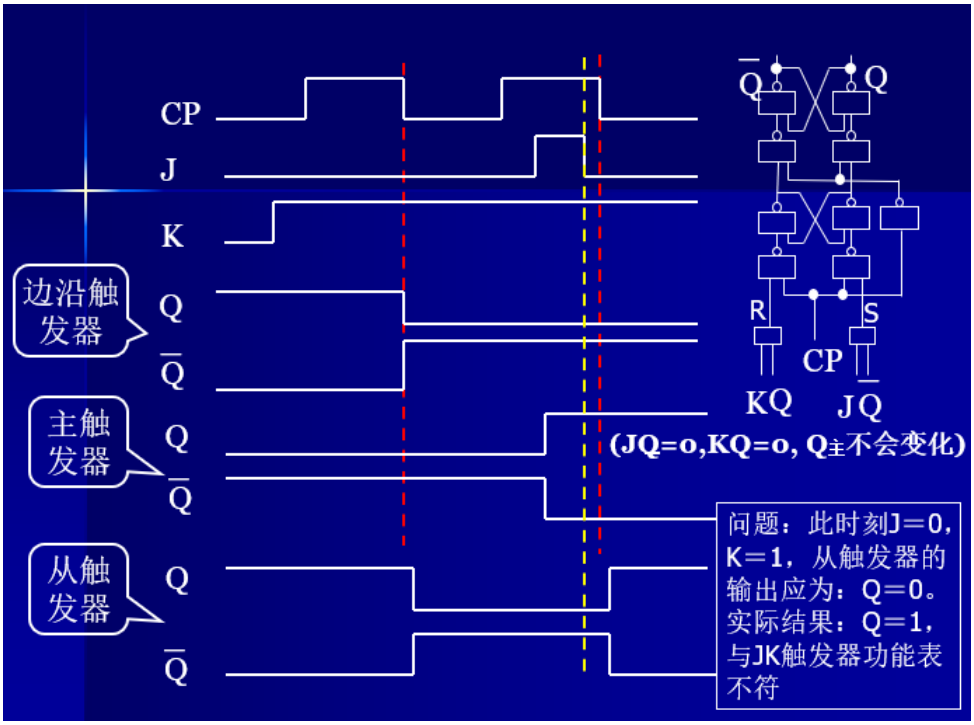
$\overline{R}_D$	$\overline{S}_D$	CP	J	K	Q	$\overline{Q}$
0	1	×	×	×	0	1
1	0	×	×	×	1	0
1	1	⊓	0	1	0	1
1	1	⊓	1	0	1	0
1	1	⊓	0	0		
1	1	⊓	1	1		

主从J-K触发器功能表的前提：只是在`CP=1`期间，`J`，`K`不发生变化的前提下给出的（此时，主从J-K触发器与复变艳J-K触发器相似）

- **J-K边沿触发器 vs 主从J-K触发器**

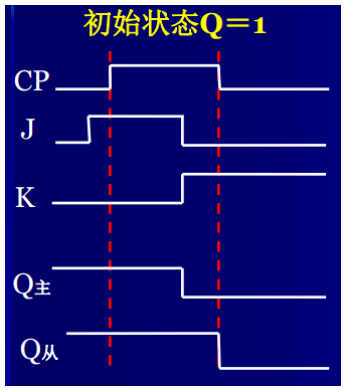
  主触发器在`CP=1`期间接收数据，在时钟的负跳变到来时，从触发器接收主触发器的状态。主从触发器的输出Q和状态在CP的负跳变时才发生变化，似乎是负边沿触发器

  边沿触发器在约定时钟跳变到来时，输出反映的是触发器的输入数据
  主从触发器反映的是从触发器的状态，不一定是输入`J`，`K`的状态

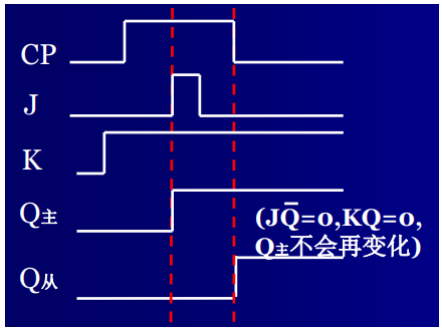


——主从J-K触发器抗干扰能力差

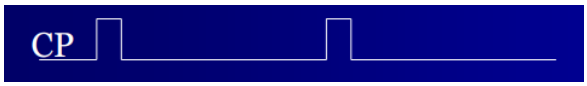
- **主从J-K触发器对 `CP=1` 器件 `JK` 变化的波形分析**
  - `CP=1` 期间，`JK` 不发生变化：触发器功能正常
  - `CP=1` 期间，`JK` 变化一次：可能会影响触发器状态（跟初始状态有关）



  - `CP=1` 期间，`JK` 变化两次：触发器状态不满足功能表描述

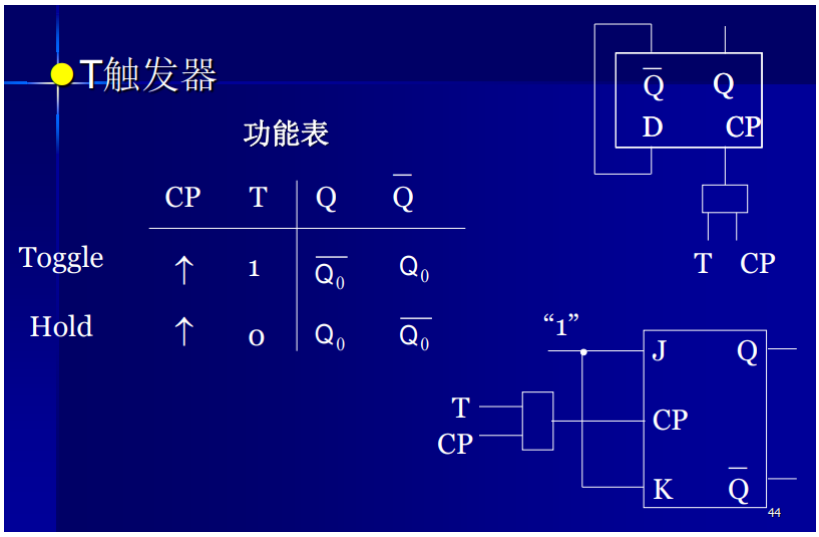


因此，主从J-K触发器的时钟配和方式：使用窄脉宽的CP信号，高电平时间缩短，可以保证主从J-K触发器正常工作

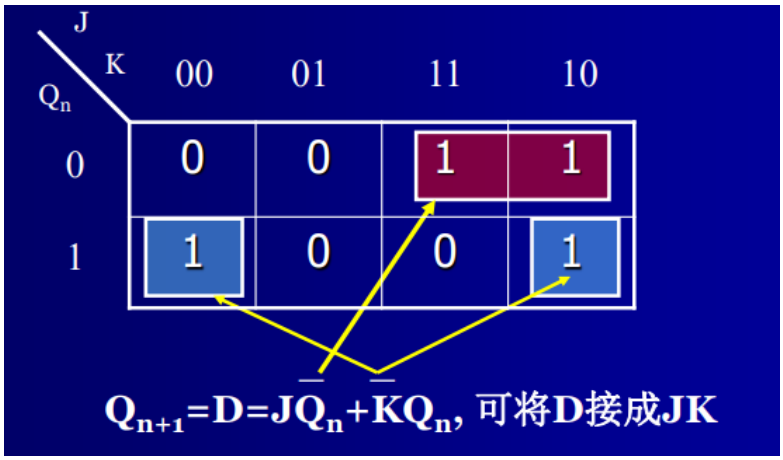


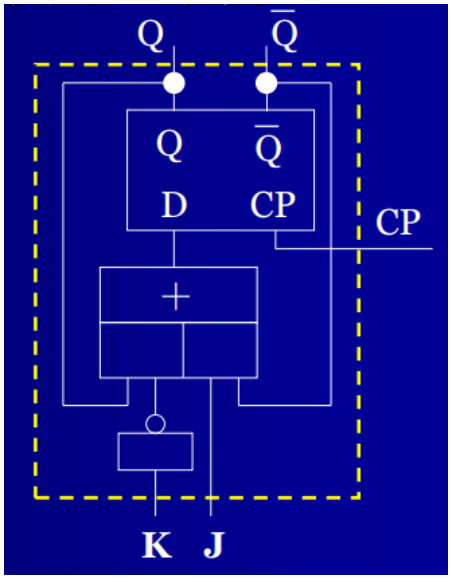
6. T触发器

不设数据输入端，只要来一个时钟脉冲，触发器就翻转一次
T触发器可由D触发器或J-K触发器等构成
T触发器一般都是边沿触发器

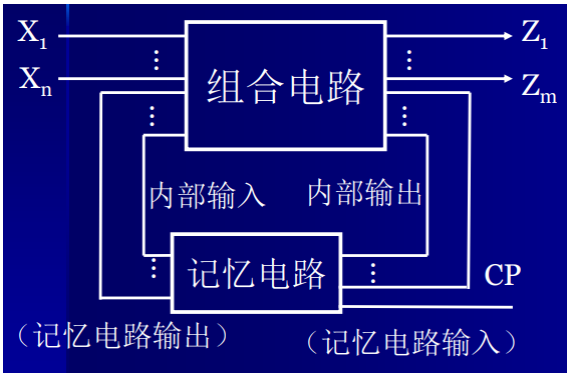


- **用D触发器转换成J-K触发器**





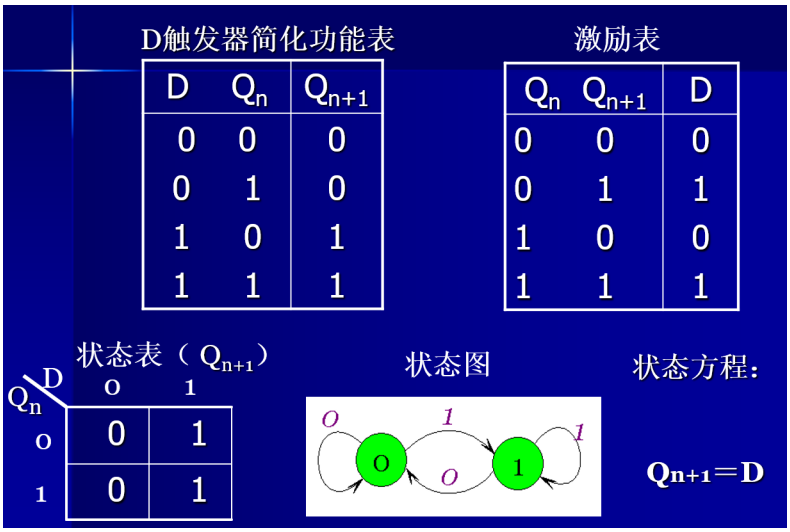
3.2 同步时序电路的分析与设计方法

> Synchronous Sequential Logic Circuit
>
> 时序电路的一般结构：
>
> 
>
>
> 同步时序电路的分析工具：状态图（State Diagram），状态表（State Table）

同步时序电路：各触发器使用**同一个CP**；只有约定时钟到来，电路状态才能改变；一个脉冲只能改变一次状态

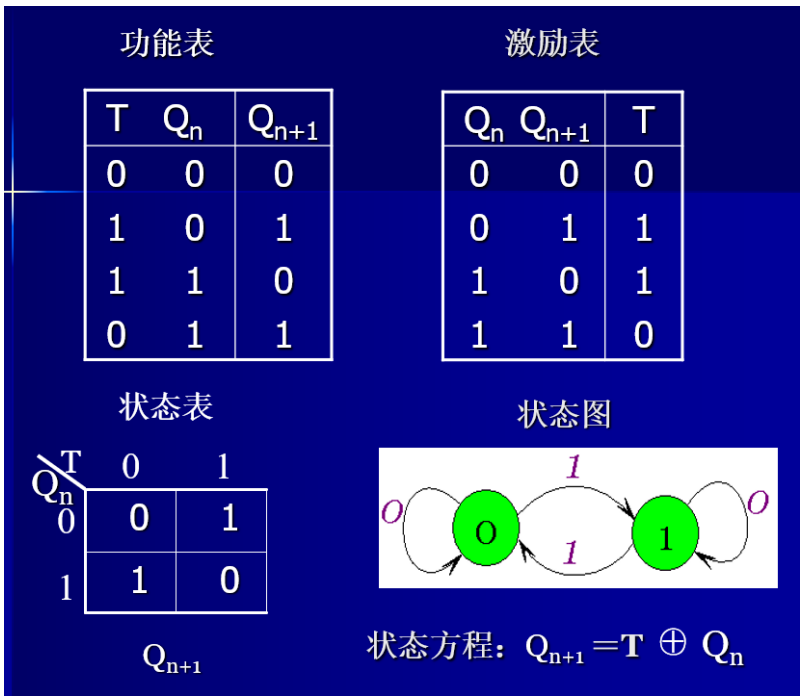
功能表：描述电路输入输出关系
现态Q_n：约定时钟跳变到来之前电路的状态
次态Q_{n+1}：约定时钟跳变到来之后电路的状态
状态表与状态图：反映输入与状态转换的关系
状态方程：状态转换的表达式
激励表：从现态转变到次态，对触发器激励的要求

- **D触发器**



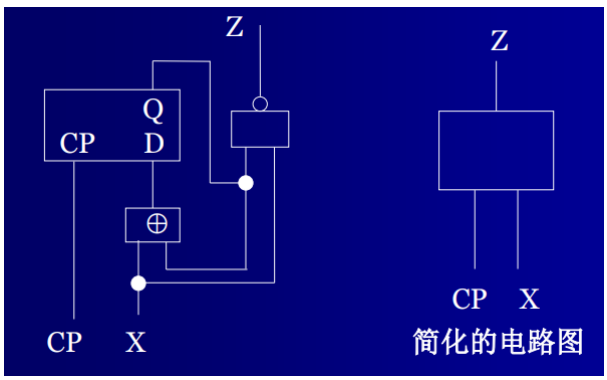
- **J-K触发器**

- **T触发器**



1. 同步时序电路的分析

Example：



- 写出激励函数D表达式，电路输出函数Z表达式
  $Q_{n+1}=D=X\oplus Q_n$，$Z=\overline{X\cdot Q_n}$
- 计算状态表，假设初始状态$Q_N=0$

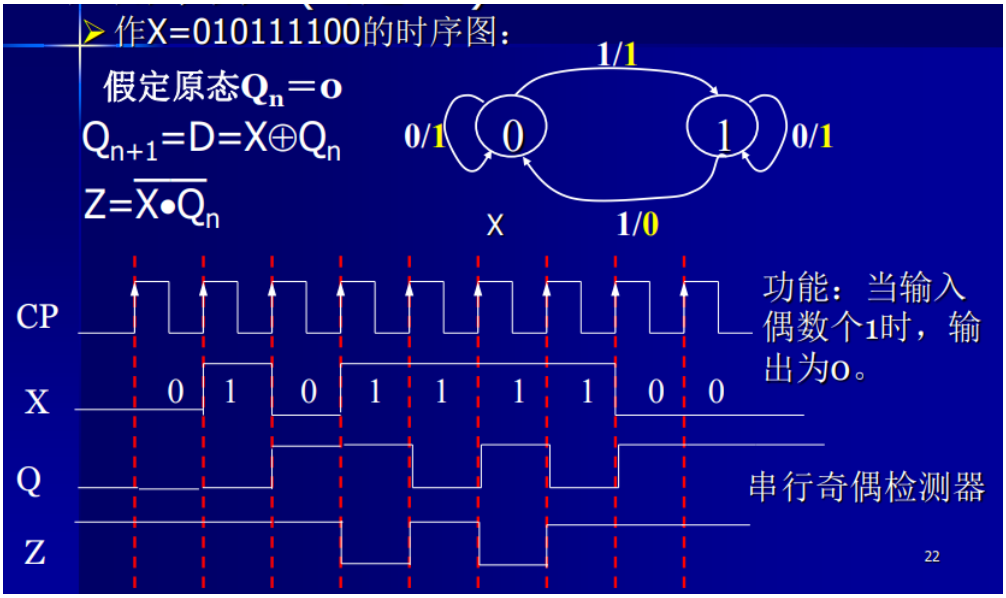
状态表

X	Q_n	D	Q_{n+1}	Z
0	0	0	0	1
1	0	1	1	1
0	1	1	1	1
1	1	0	0	0

输入 现态 激励 次态 输出

20

- 画出状态图和时序图，假定原态$Q_n = 0$



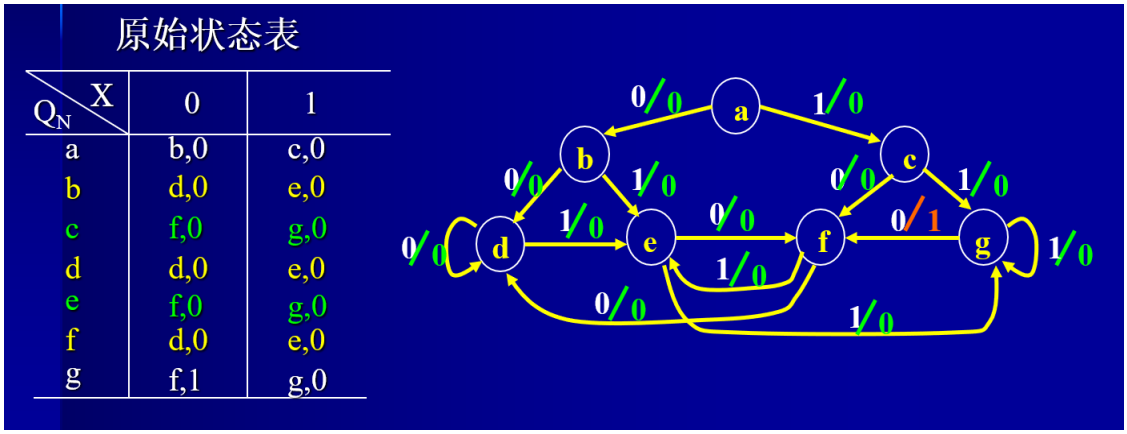
2. 同步时序电路的设计

文字描述 → 状态图 → 逻辑图

Example：

> 用与非门和D触发器设计一个同步时序逻辑电路，以检测输入信号序列是否为连续的`110`

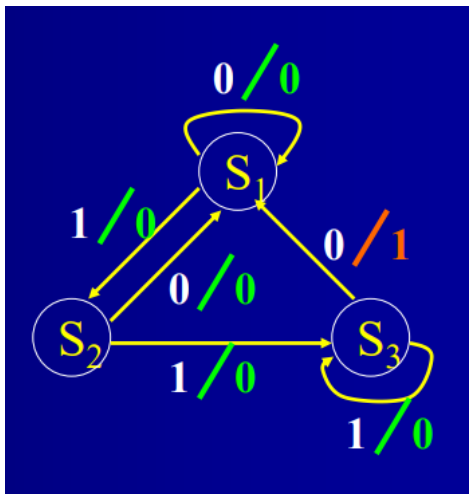
1. 确定输入变量（`x`，一个二进制序列）和输出变量（`z`，检测信号）
2. 做出原始状态表



根据分析做出中间状态表

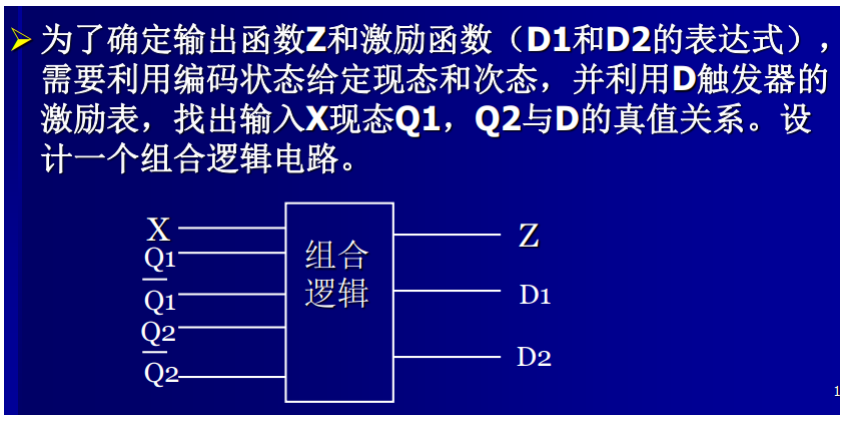
Q_N \ X	0	1
a	s_1, 0	s_2, 0
s_1	s_1, 0	s_2, 0
s_2	s_1, 0	g, 0
g	s_1, 1	g, 0

3. 状态表化简（如上图可以发现`a`和`s1`可以合并）——最简状态表



Q_N \ X	0	1
S_1	S_1, 0	S_2, 0
S_2	S_1, 0	S_3, 0
S_3	S_1, 1	S_3, 0

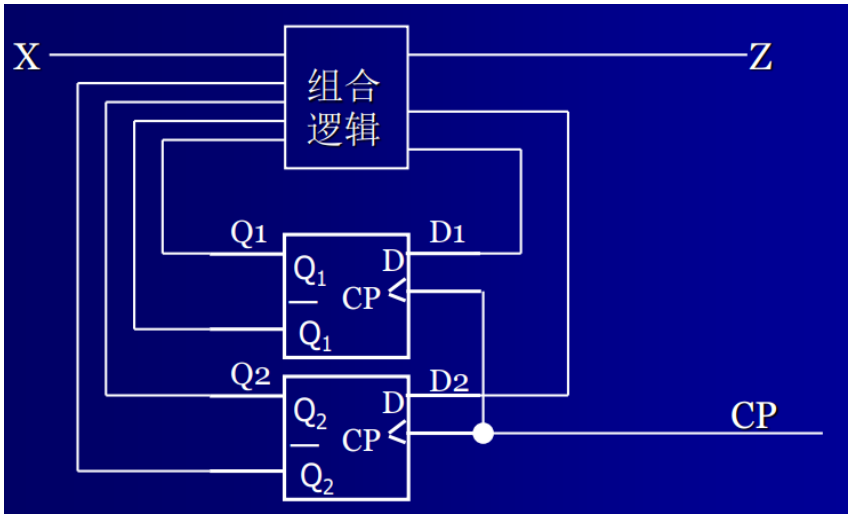
4. 状态分配：给确定的状态S_1， S_2，S_3二进制编码
   采用两个触发器的输出Q_1，Q_2实现，分别用`00`（$\overline{Q_2}\ \overline{Q_1}$），`10`（$Q_2\ \overline{Q_1}$），`11`（$Q_2Q_1$）表示三种状态
5. 求出激励函数和输出函数

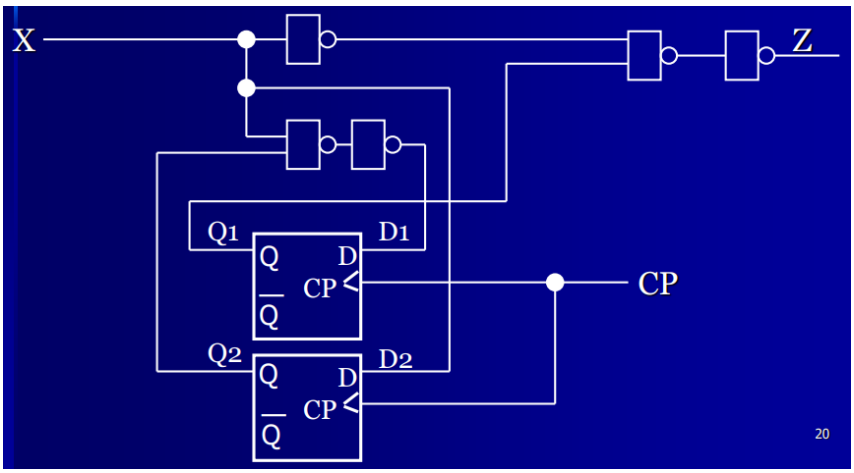


X	Q_{2N}	Q_{1N}	$Q_{2(N+1)}$	$Q_{1(N+1)}$	D2	D1	Z
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	1
1	0	0	1	0	1	0	0
1	1	0	1	1	1	1	0
1	1	1	1	1	1	1	0
0	0	1	×	×	×	×	×
1	0	1	×	×	×	×	×

$D2=X$，$D1=XQ_{2N}$，$Z=\overline{X}\,Q_{1N}$

6. 画出逻辑图（D触发器）





7. 改用J-K触发器进行设计

功能表

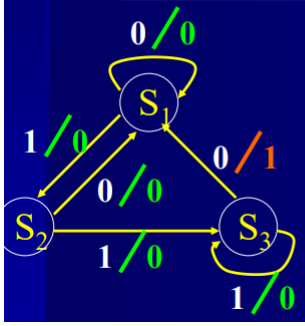
J	K	Q_n	Q_{n+1}	
0	0	0	0	Hold
0	0	1	1	
0	1	0	0	Store 0
0	1	1	0	
1	0	0	1	Store 1
1	0	1	1	
1	1	0	1	Count
1	1	1	0	

激励表

Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

22

X	Q_{2N}	Q_{1N}	$Q_{2(N+1)}$	$Q_{1(N+1)}$	J2	K2	J1	K1	Z
0	0	0	0	0	0	×	0	×	0
0	1	0	0	0	×	1	0	×	0
0	1	1	0	0	×	1	×	1	1
1	0	0	1	0	1	×	0	×	0
1	1	0	1	1	×	0	1	×	0
1	1	1	1	1	×	0	×	0	0
0	0	1	×	×	×	×	×	×	×
1	0	1	×	×	×	×	×	×	×



$J2=X$，$K2=\overline{X}$，

$J1=XQ_{2N}$，$K1=\overline{X}$

$Z=\overline{X}\,Q_{1N}$

23

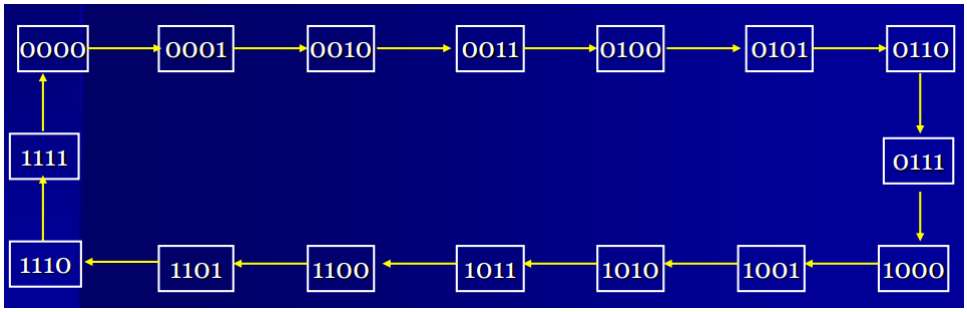
3.3 计数器

功能：对CP脉冲计数，一个脉冲变化一次状态

分类：同步计数器+异步计数器，加法计数器+减法计数器+可逆计数器，进制计数器，环形计数器、扭环计数器……

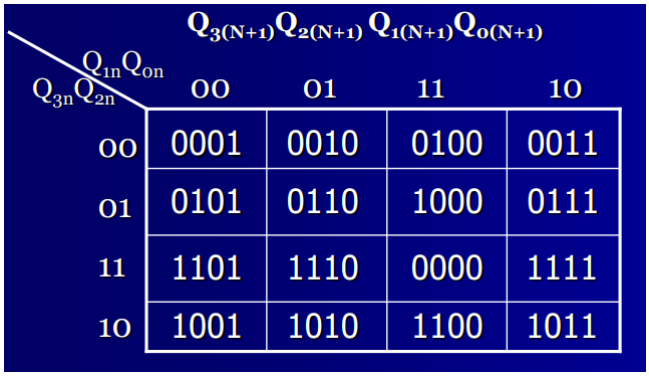
1. 同步二进制计数器设计

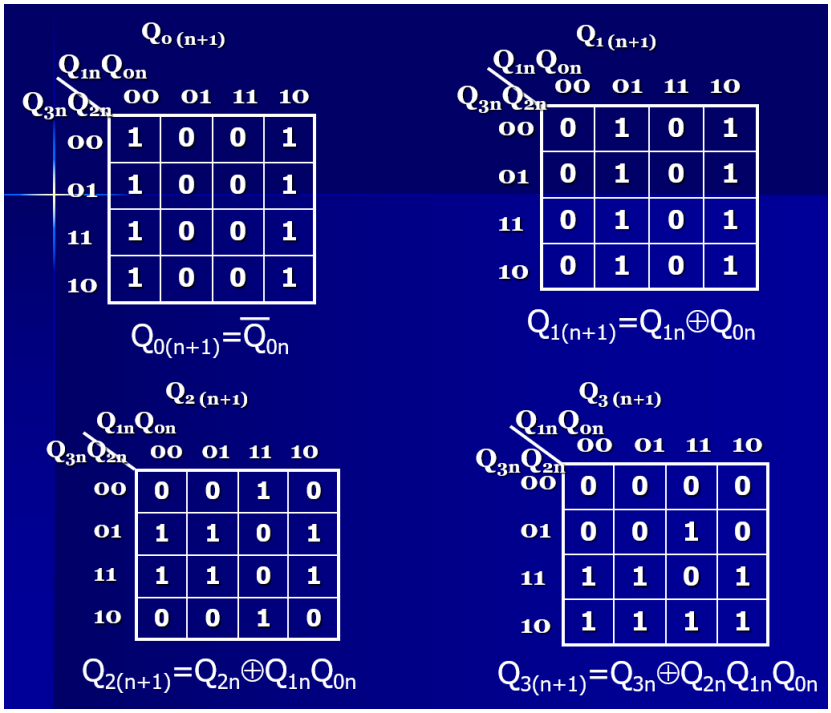
- 画出状态图和状态表



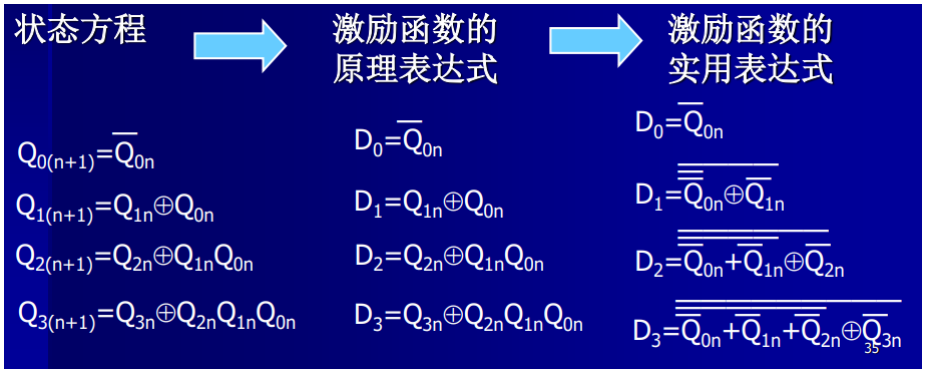
N	Q_{3n}	Q_{2n}	Q_{1n}	Q_{0n}	Q_{3N+1}	Q_{2N+1}	Q_{1N+1}	Q_{0N+1}
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	0
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	1	0	0
4	0	1	0	0	0	1	0	1
5	0	1	0	1	0	1	1	0
6	0	1	1	0	0	1	1	1
7	0	1	1	1	1	0	0	0
8	1	0	0	0	1	0	0	1
9	1	0	0	1	1	0	1	0
10	1	0	1	0	1	0	1	1
11	1	0	1	1	1	1	0	0
12	1	1	0	0	1	1	0	1
13	1	1	0	1	1	1	1	0
14	1	1	1	0	1	1	1	1
15	1	1	1	1	0	0	0	0

- 利用状态表求次态表达式

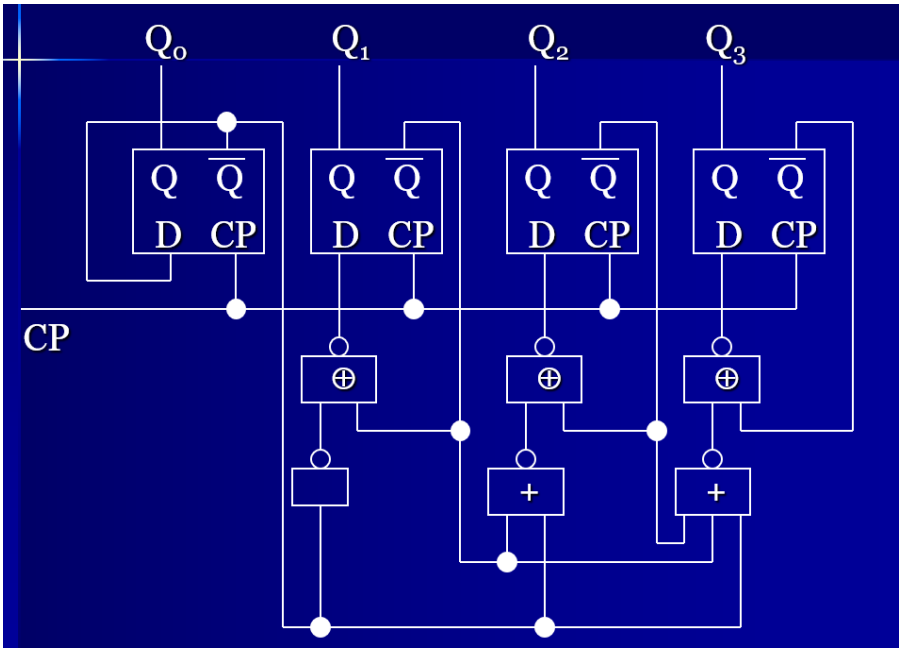


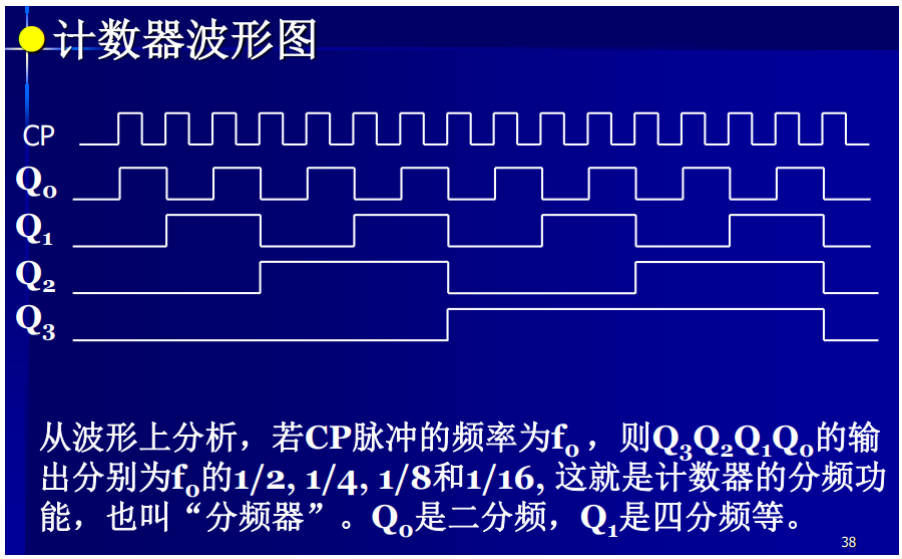


- 求激励函数表达式（D触发器）

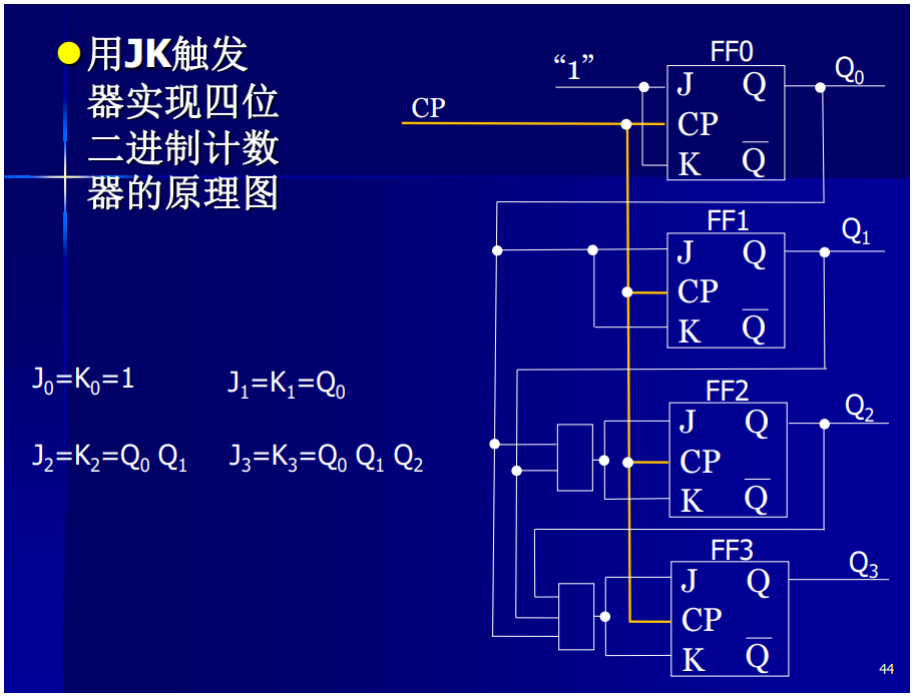


- 画出逻辑图和波形图，并进行分析



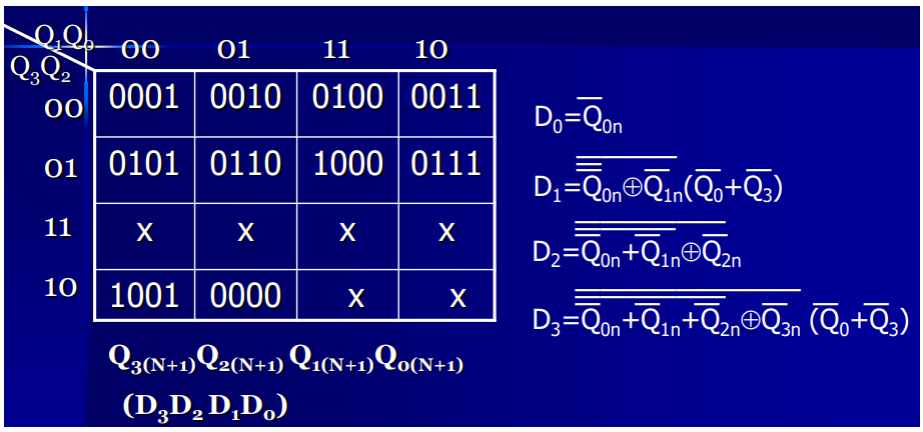


- 改用J-K触发器设计



2. 十进制计数器设计

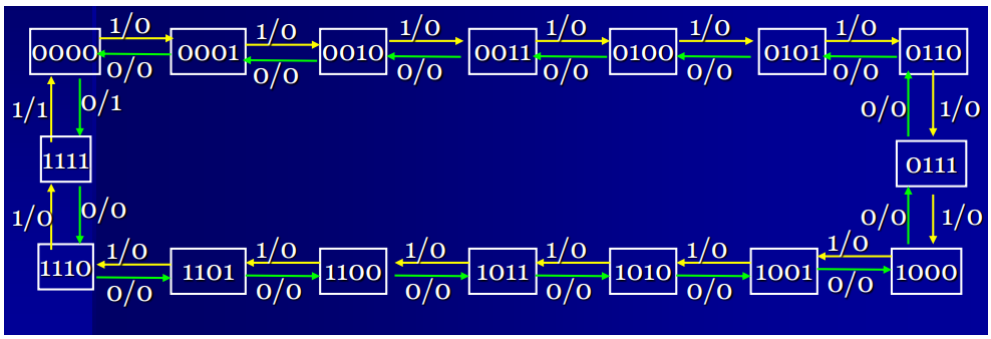
- D触发器



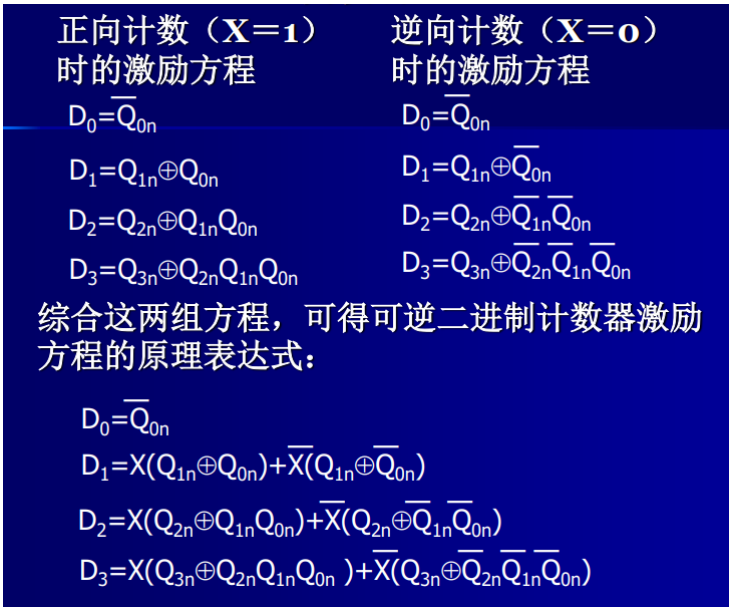
问题：有可能不能自启动（解决方法：将X变为确定的状态）

3. 可逆计数器设计

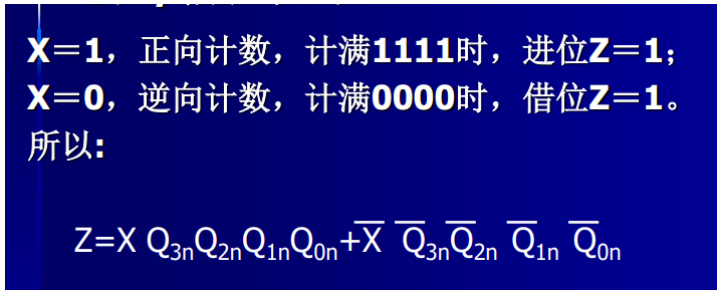
> 分类：单时钟可逆计数器和双时钟可逆计数器
> 用D触发器设计单时钟四位二进制可逆计数器：
> X=1，正项计数，计满1111时进位Z=1
> X=0，逆向计数，计满0000时借位Z=1



- 激励设计

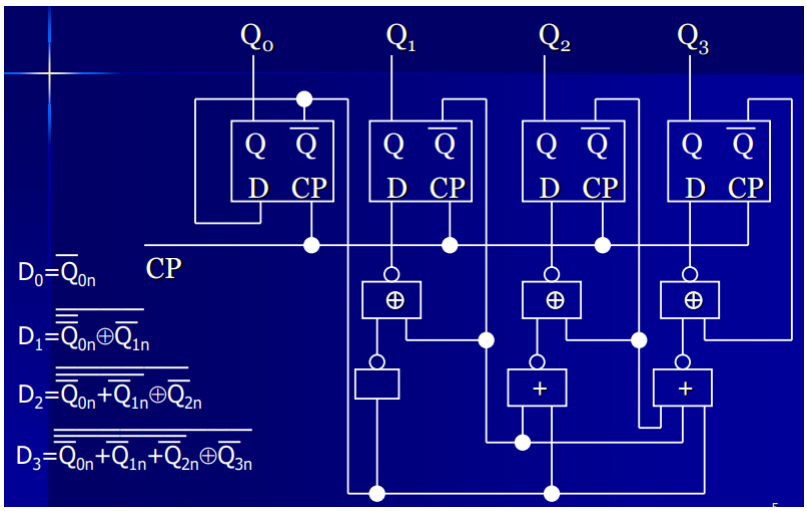


进位 / 借位设计：



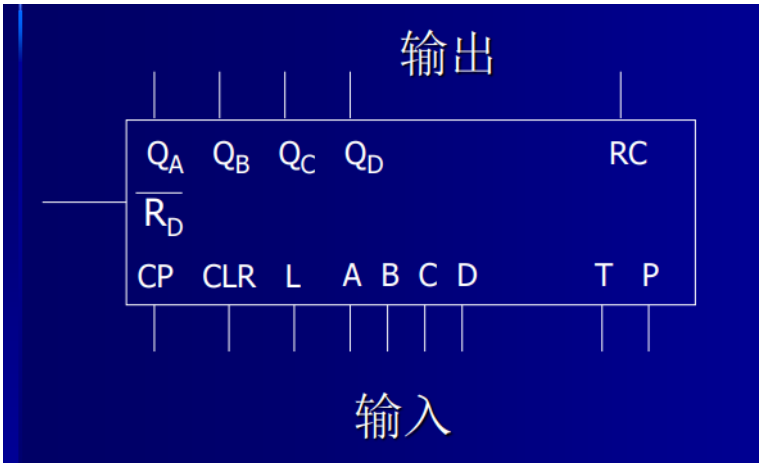
3.4 集成化的4位二进制计数器

> 4位二进制计数器的状态方程和激励函数：
>
> 状态方程：
>
> $$Q_{0,n+1}=\overline{Q_{0,n}}$$
> $$Q_{1,n+1}=Q_{1,n}\oplus Q_{0,n}$$
> $$Q_{2,n+1}=Q_{2,n}\oplus Q_{1,n}Q_{0,n}$$
> $$Q_{3,n+1}=Q_{3,n}\oplus Q_{2,n}Q_{1,n}Q_{0,n}$$
>
> 激励函数的原理表达式：
>
> $$D_{0,n+1}=\overline{Q_{0,n}}$$
> $$D_{1,n+1}=Q_{1,n}\oplus Q_{0,n}$$
> $$D_{2,n+1}=Q_{2,n}\oplus Q_{1,n}Q_{0,n}$$
> $$D_{3,n+1}=Q_{3,n}\oplus Q_{2,n}Q_{1,n}Q_{0,n}$$
>
> 激励函数的使用表达式：
>
> $$D_{0,n+1}=\overline{Q_{0,n}}$$
> $$D_{1,n+1}=\overline{\overline{Q_{0,n}}\oplus\overline{Q_{0,n}}}$$
> $$D_{2,n+1}=\overline{\overline{Q_{0,n}}+\overline{Q_{1,n}}\oplus\overline{Q_{2,n}}}$$
> $$D_{3,n+1}=\overline{\overline{Q_{0,n}}+\overline{Q_{1,n}}+\overline{Q_{2,n}}\oplus\overline{Q_{3,n}}}$$



- **集成化的计数器具有的基本功能**

  同步 / 异步清零：初始化
  同步置数：开始计数的初值
  计数：正常工作
  保持：计数到某个特定的计数值保持
  扩展性：相同的计数器级连更多位数的计数器
  进位传递：扩展时进位传递到高位片



> RC：进位输出
> R_D：异步清零
> CLR：同步清零
> L：同步置数，置为$ABCD$
> T, P：用于进位传递和拓展，计数使能端

➢ 用负边沿D触发器构成的4位二进制计数器

R_D	CLR	L	T	P	CP	功能
0	X	X	X	X	X	清零(异步)
1	0	X	X	X	↓	清零(同步)
1	1	0	X	X	↓	Load(并行数据输入)
1	1	1	1	1	↓	Count(计数)
1	1	1	0	X	X	FF Hold, RC=0
1	1	1	1	0	X	Hold(保持)

1. 异步清零

用$\overline{R_D}$进行异步置零：触发器提供的异步清零方式
在输出用组合逻辑，如$\overline{R_d} \cdot Q$，这种方法并不本质，要慎重

2. 同步清零

激励函数的原理表达式：

$$D_A = CLR \cdot \overline{Q_A}$$
$$D_B = CLR \cdot (Q_B \oplus Q_A)$$
$$D_C = CLR \cdot (Q_C \oplus Q_B Q_A)$$
$$D_D = CLR \cdot (Q_D \oplus Q_C Q_B Q_A)$$

3. 并行置数

$L\ (Load)$
当L=0时，触发器接收已准备好的数据A~D
当L = 1时，触发器接收进位信号（原激励不变）

从本质上来看，用L对激励做数据选择（二选一）：

激励函数的原理表达式：

$$D_A = CLR \cdot (\overline{L} \cdot A + L \cdot \overline{Q_A})$$
$$D_B = CLR \cdot (\overline{L} \cdot B + L \cdot (Q_B \oplus Q_A))$$
$$D_C = CLR \cdot (\overline{L} \cdot C + L \cdot (Q_C \oplus Q_B Q_C))$$
$$D_D = CLR \cdot (\overline{L} \cdot D + L \cdot (Q_D \oplus Q_C Q_B Q_A))$$

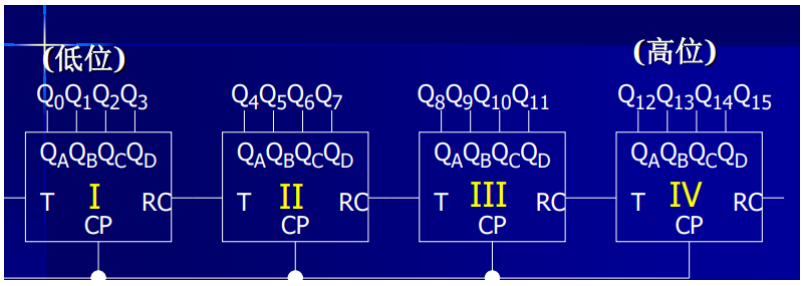
4. 扩展功能

基本方法

扩展应用：多片计数器扩展
扩展规则：① 低位片未计满，则高位片不能计数；② 低位片计满后来CP，两片一起动

设置进位标志：$RC\ (RippleCarry)$
二进制计数器：$RC = Q_A Q_B Q_C Q_D$
十进制计数器：$RC = Q_A \overline{Q_B}\ \overline{Q_C} Q_D$

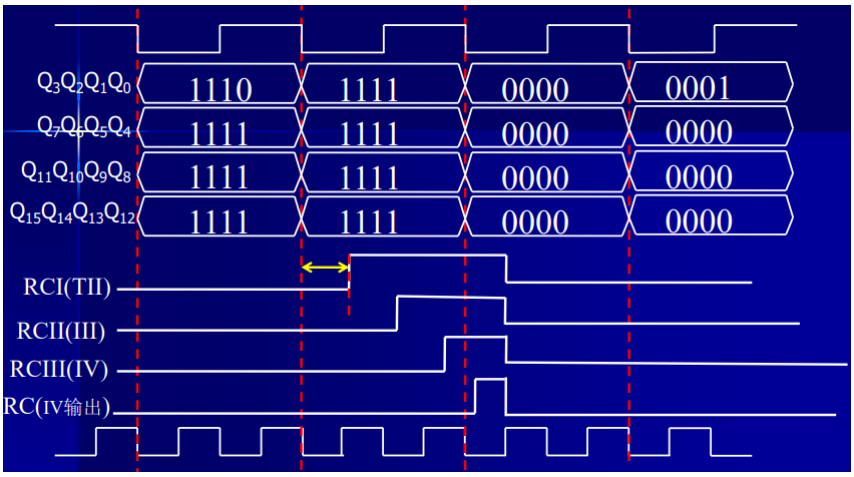
允许（禁止）端：
$T(Trickle)$：串行控制；当T = 0时，触发器保持；当T = 1时，允许计数
$P(Parallel)$：并行控制



激励函数的原理表达式：

$$D_A = CLR \cdot (\overline{L} \cdot A + L \cdot (\overline{T} \cdot Q_A + T \cdot \overline{Q_A}))$$
$$D_B = CLR \cdot (\overline{L} \cdot B + L \cdot (\overline{T} \cdot Q_B + T \cdot (Q_B \oplus Q_A)))$$
$$D_C = CLR \cdot (\overline{L} \cdot C + L \cdot (\overline{T} \cdot Q_C + T \cdot (Q_C \oplus Q_B Q_C)))$$
$$D_D = CLR \cdot (\overline{L} \cdot D + L \cdot (\overline{T} \cdot Q_D + T \cdot (Q_D \oplus Q_C Q_B Q_A)))$$
$$RC = Q_D Q_C Q_B Q_A T$$

问题：片间进位信号RC逐片传递，造成计数频率不能很高



该图中，再扩展一片，就不行了

快速扩展

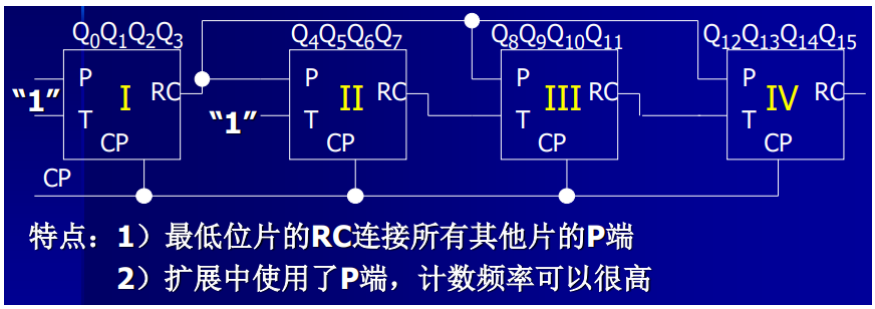
分析产生进位的条件：

1 当所有低位片的输出都为“1111”且在来一个约定时钟时， 本片才计数一次。

2 要想大家同时进位，计数器需要再增加一个控制端P，配合T一齐使用，只有当$P=1$且$T=1$时，本片才进位。

3 最低位片的RC和P相连，当最低位片有进位且本片的上 一个RC也为1时，本片进位。

换而言之，仅低12位$=1$即可满足最高位片$RC=1$



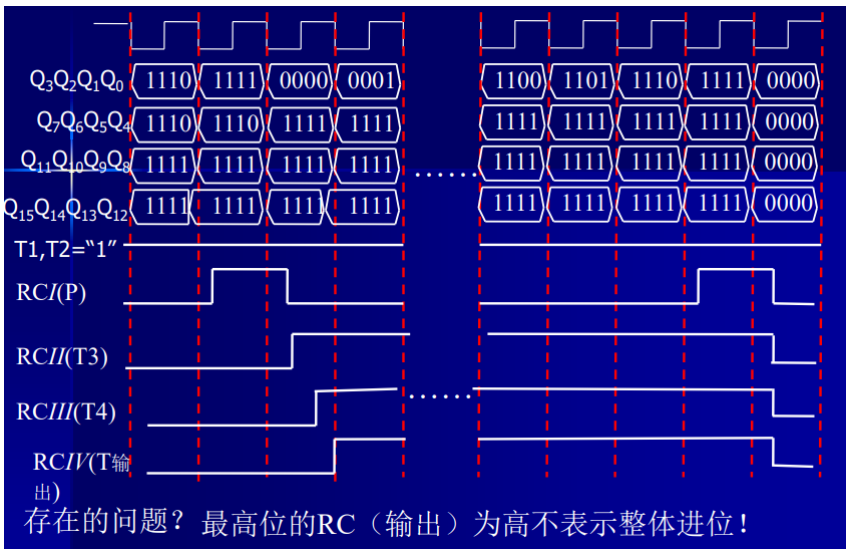
T、P有一个是0时，触发器保持；

$T=P=1$时，才允许计数

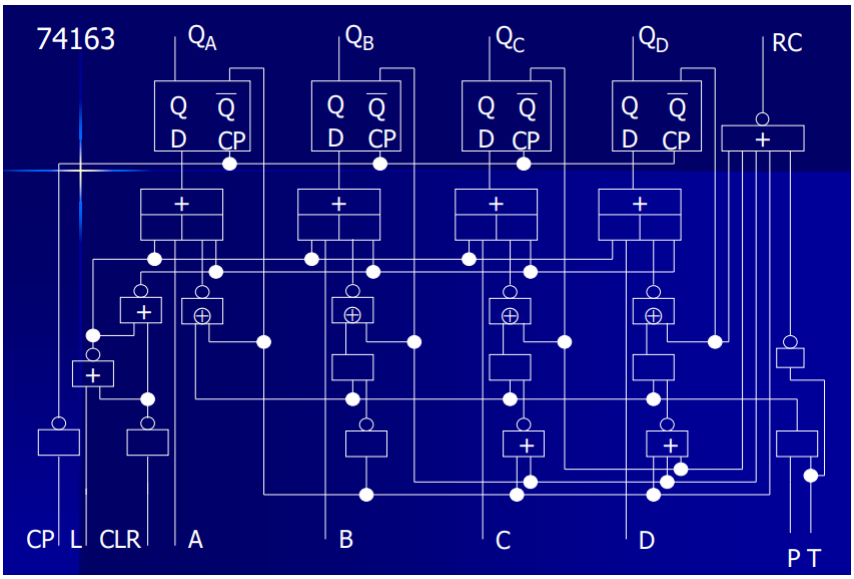
激励函数的原理表达式：

$$D_A=CLR\cdot(\overline{L}\cdot A+L\cdot(\overline{T\cdot P}\cdot Q_A+T\cdot P\,\overline{Q_A}))$$
$$D_B=CLR\cdot(\overline{L}\cdot B+L\cdot(\overline{T\cdot P}\cdot Q_B+T\cdot P\,(Q_B\oplus Q_A)))$$
$$D_C=CLR\cdot(\overline{L}\cdot C+L\cdot(\overline{T\cdot P}\cdot Q_C+T\cdot P\,(Q_C\oplus Q_BQ_C)))$$
$$D_D=CLR\cdot(\overline{L}\cdot D+L\cdot(\overline{T\cdot P}\cdot Q_D+T\cdot P\,(Q_D\oplus Q_CQ_BQ_A)))$$
$$RC=Q_DQ_CQ_BQ_AT$$

问题：



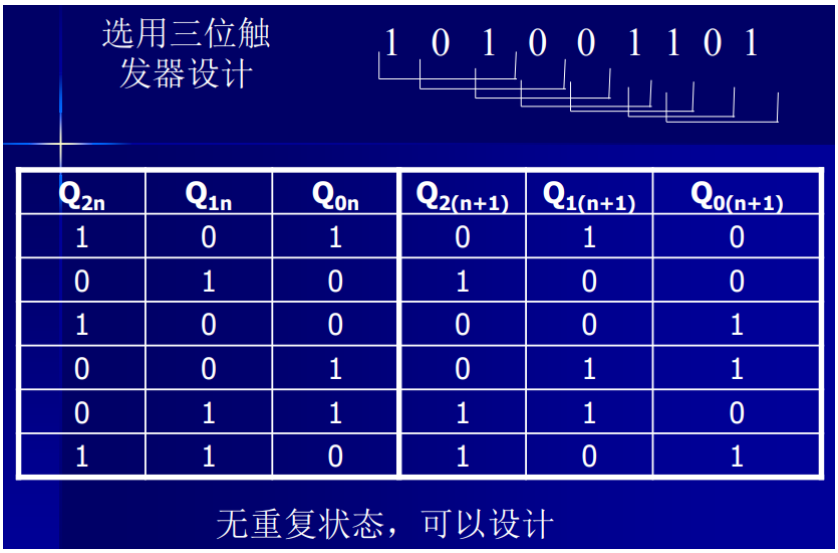
5. 74163 逻辑图



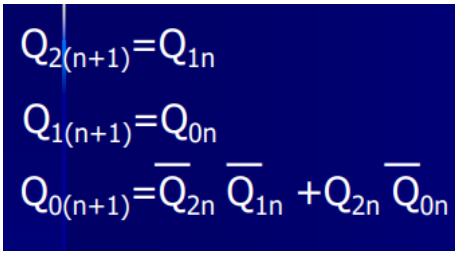
6. 脉冲序列信号发生器的设计

设计脉冲序列信号发生器：*101001 101001 101001...*
（三种设计方式：*74163*，*D/JK*触发器，循环移位方式）

- **循环移位**



Q_{2n}	Q_{1n}	Q_{0n}	$Q_{2(n+1)}$	$Q_{1(n+1)}$	$Q_{0(n+1)}$
1	0	1	0	1	0
0	1	0	1	0	0
1	0	0	0	0	1
0	0	1	0	1	1
0	1	1	1	1	0
1	1	0	1	0	1

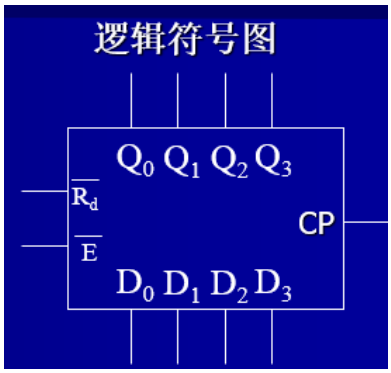


3.5 寄存器

> 在计算机中用于存储指令、数据、运算结果
> 寄存器的数量多少，是计算机结构的重要区别之一
> 外存、内存、缓存、寄存四类中，寄存器速度最快，但容量最小

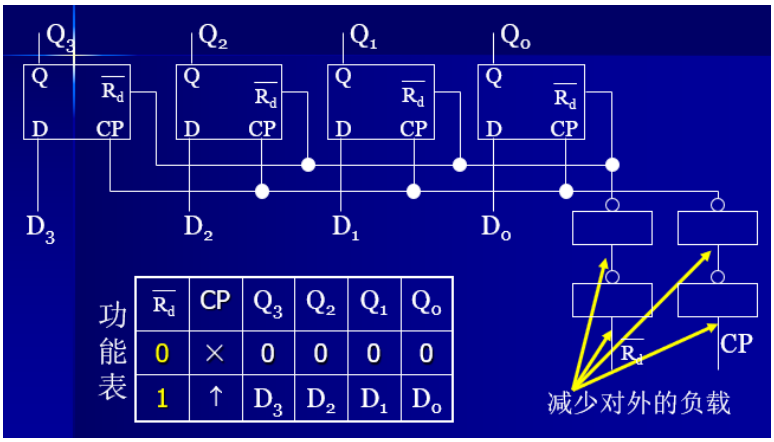
1. 基本寄存器

写、存、读（用于计算加减乘除的结果）
$\overline{R_d}=0$的时候异步置零，$\overline{R_d}=1$的时候置数（$Q_3Q_2Q_1Q_0=D_3D_2D_1D_0$）



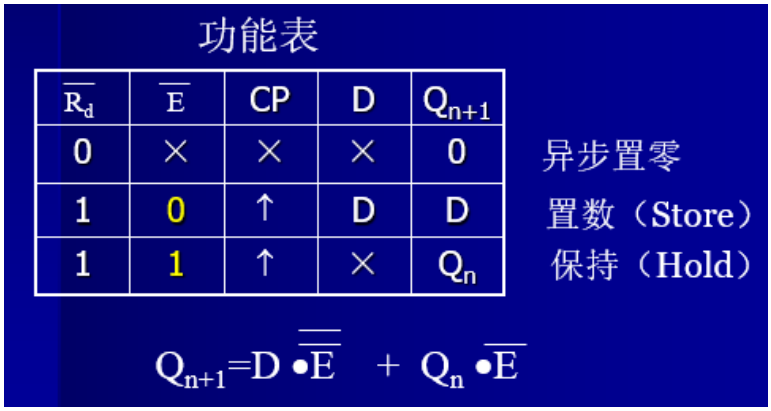
- **写：异布置零、置数**

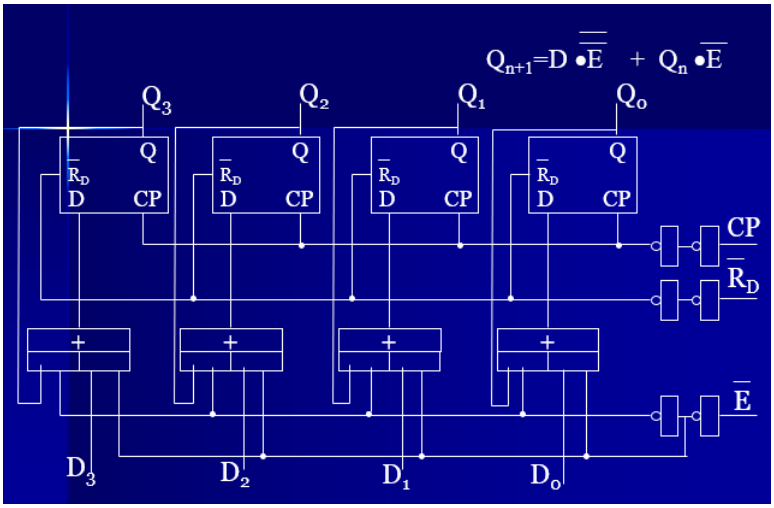
  寄存器的基本功能是存储；如图为4个D触发器构成的寄存器



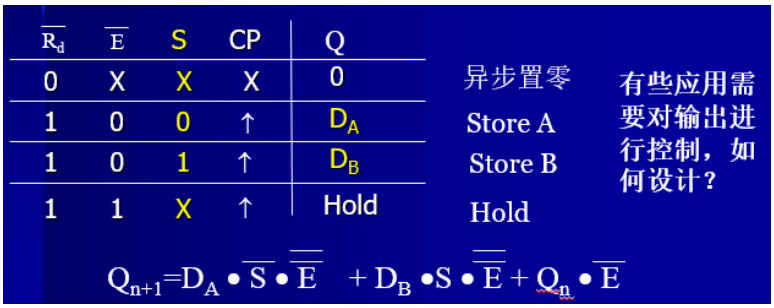
- **存：保持功能**

  用4D触发器构成的寄存器存在的问题：数据撤销，输出不能保持数据。
  设计具有保持（Hold）功能的4D寄存器：



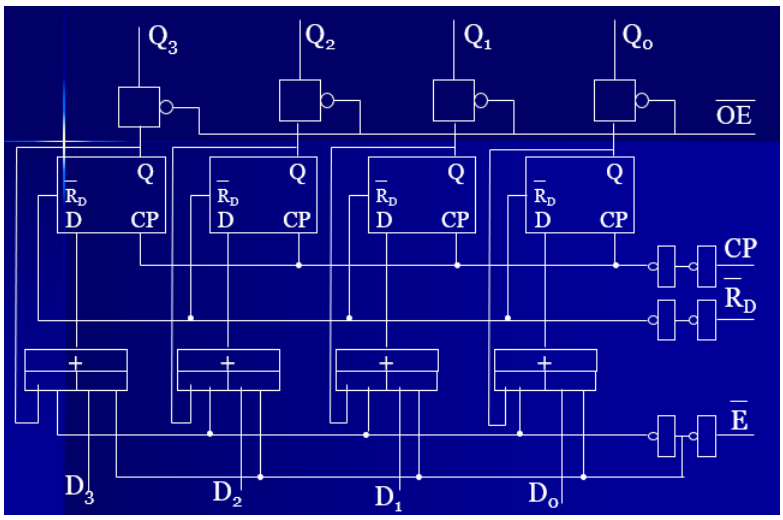


选择功能（每个输出是两个输入之一，需要对输入进行控制）：

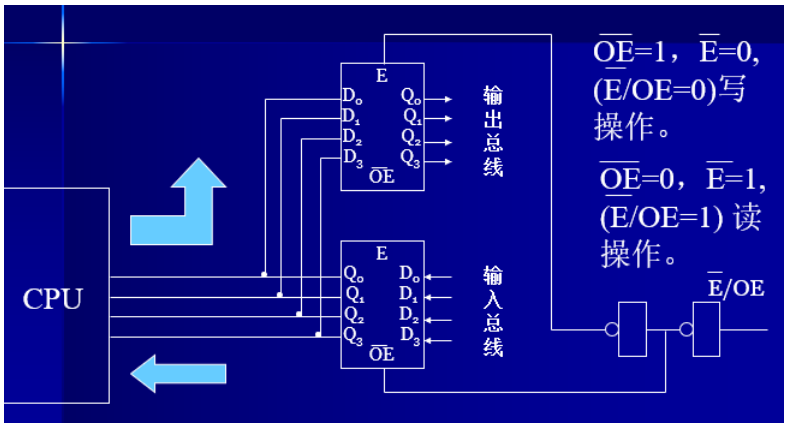


- **读：输入输出使能**

  使用使能端$\overline{OE}$：$\overline{OE}=1$（高阻态）表示伴随关系（即多个输出通过三态门接在一起，只能有一个输出为`1`）



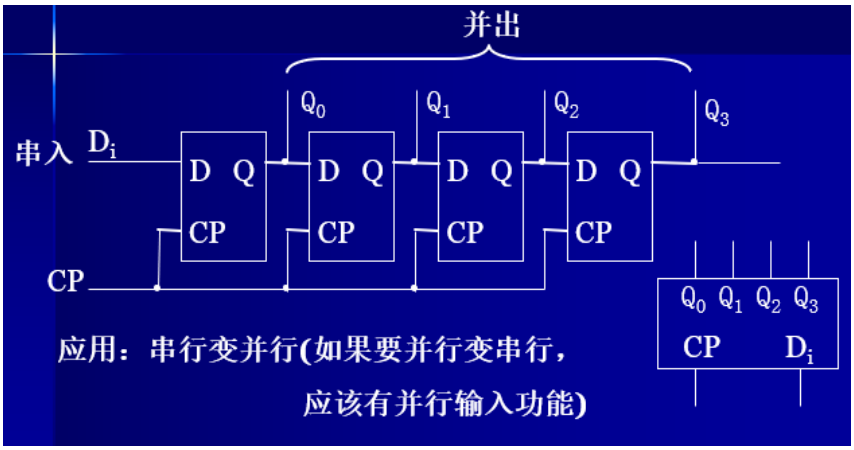
应用举例：用有输入输出使能的4D寄存器组成双向总线寄存器

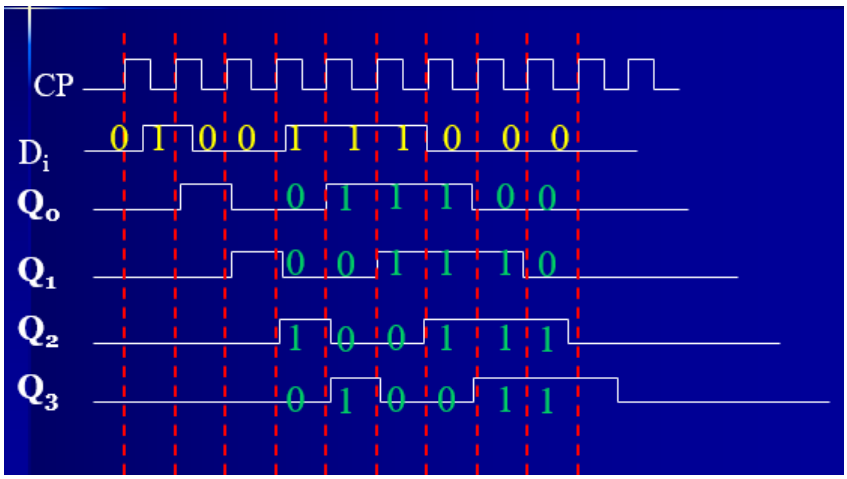


2. 移位寄存器

移位寄存器定义：具有移位功能的寄存器。
设计移位寄存器的要求：每来一个时钟脉冲（移位命令），寄存器中的数据就顺序左（右）移动一位，所以必须采用边沿触发器或主从触发器。

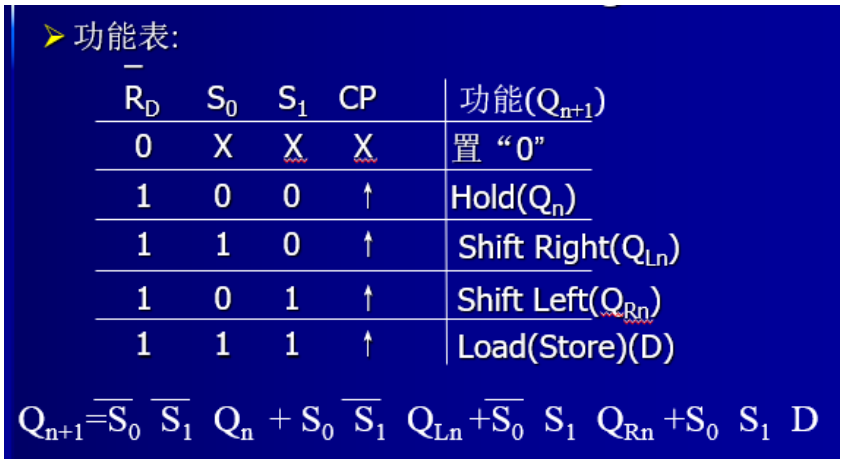
- **串入 / 并出的移位寄存器（右移）**



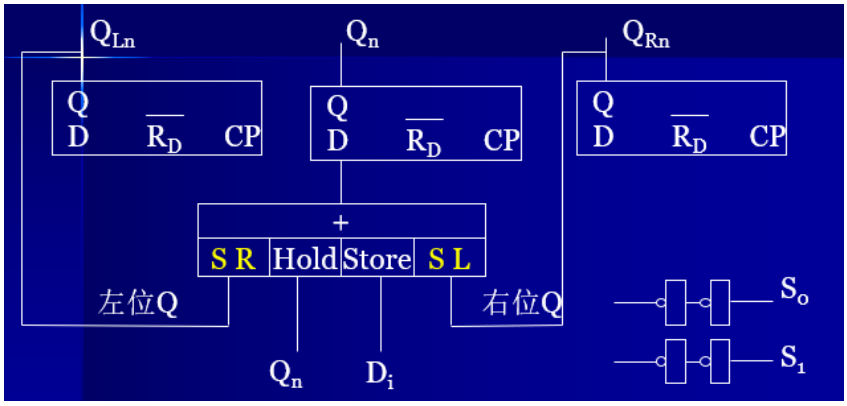


每四个时钟周期读一次：0100 => 1110

- **并入 / 并出的双向移位寄存器**

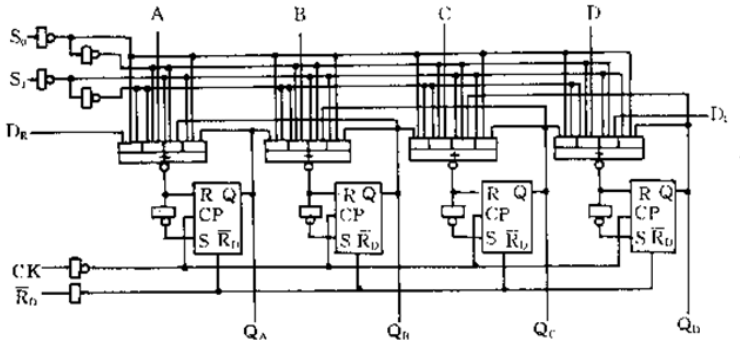


简化结构图：



集成化双向移位寄存器：

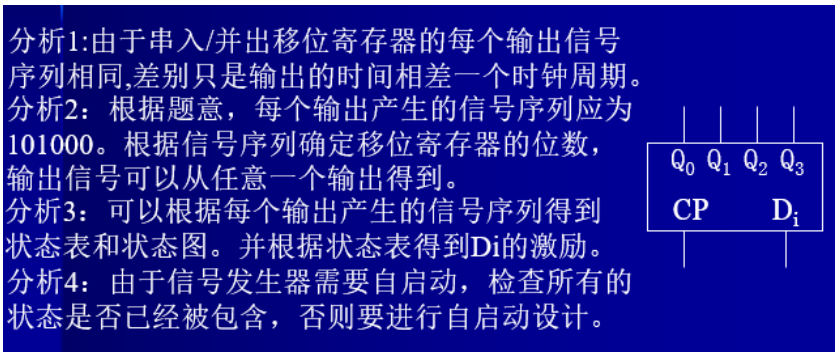
其中，D_r，D_L表示右移 / 左移的补位数码或者循环，下图中是悬空的



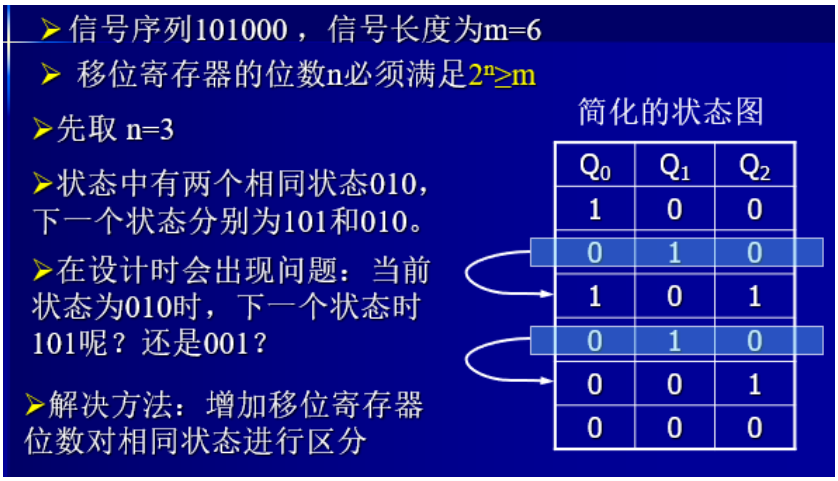
$\overline{R}_D$	S_0	S_1	CK	功 能
0	×	×	×	置"0"
1	0	0	↑	保持
1	1	0	↑	左移
1	0	1	↑	右移
1	1	1	↑	并行输入

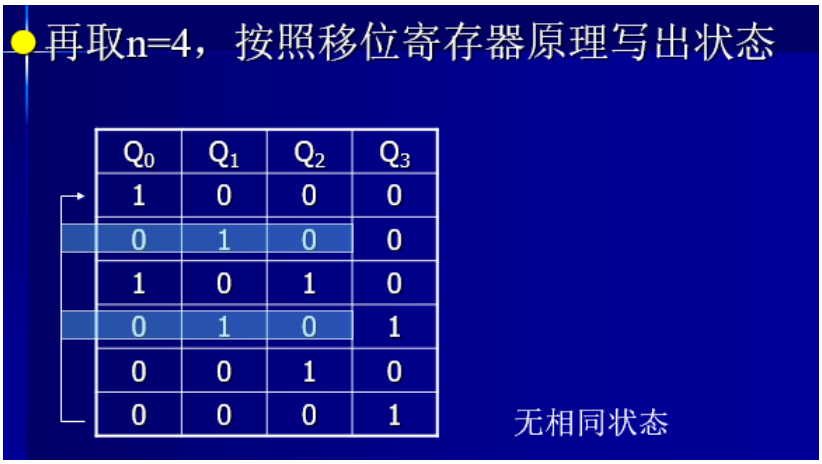
图 5-46　集成化双向移位寄存器之二

举例：用串入/并出的移位寄存器（右移）设计信号序列发生器，产生如下信号序列：101000,101000...

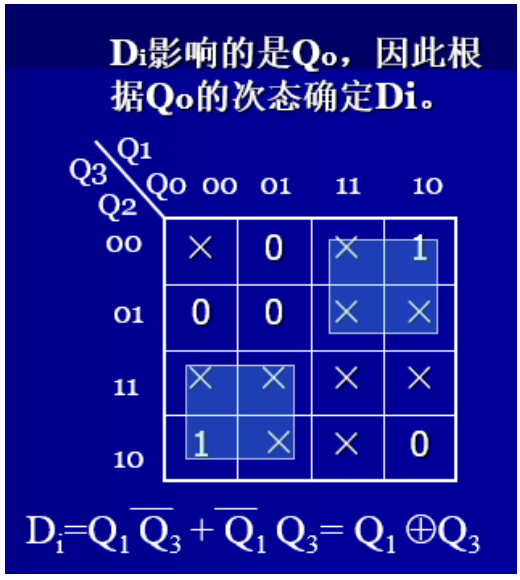


1. 确定移位寄存器的位数n

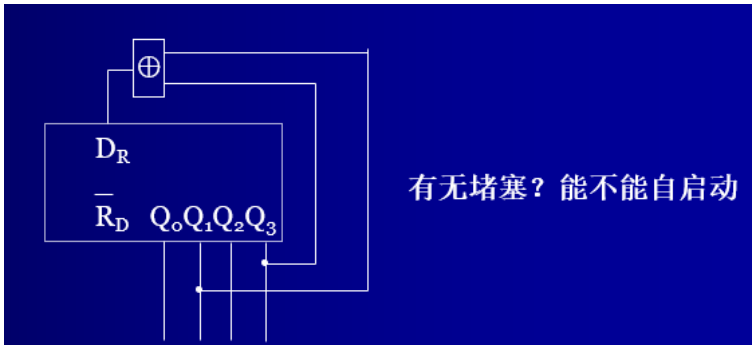




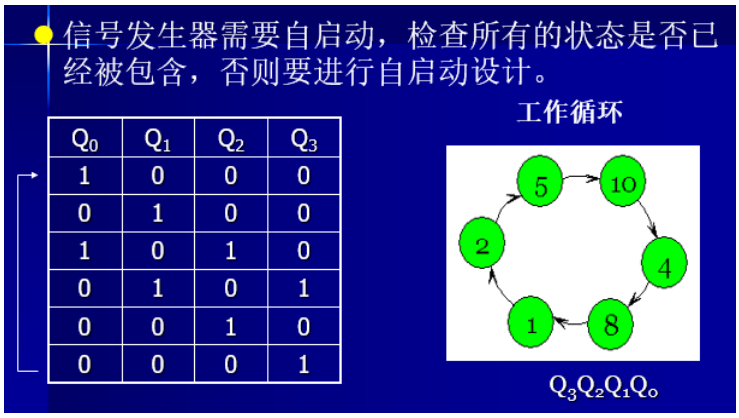
2. 设计D_i：Q_0的次态：



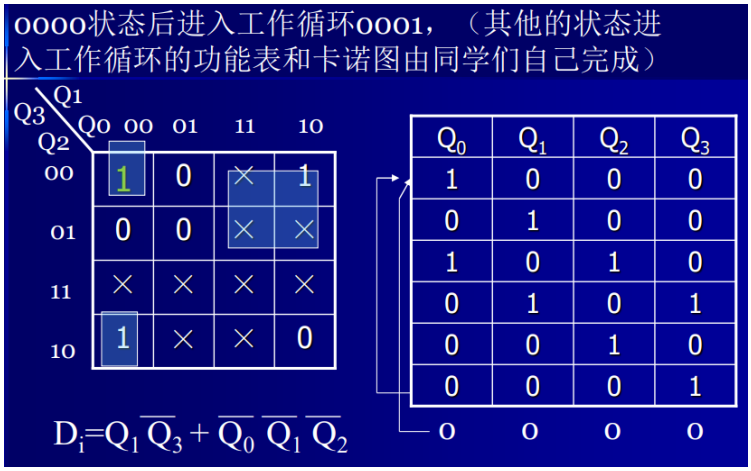
3. 得到逻辑图：



4. 检查是否可以自启动



这里的例子存在三个非工作循环，需要更改卡诺图中的×来进行改进



4 可编程逻辑电路

硬件描述语言（HDL）

可编程逻辑器件PLD (Programmable Logic Device)是一大类器件的总称，包括：

ROM (Read-Only Memory)：只读存储器（PROM：与阵列固定、或阵列可编程）

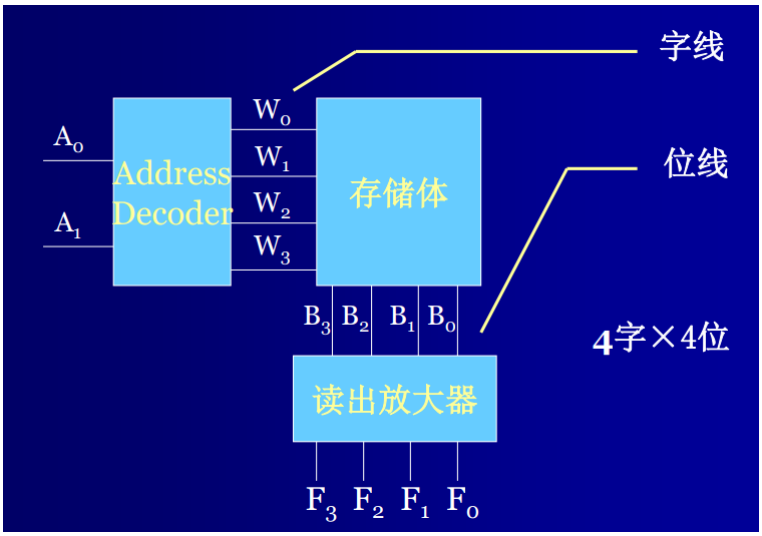
PLA (Programmable Logic Array)：可编程逻辑阵列（与、或阵列均可编程）

PAL (Programmable Array Logic)：可编程阵列逻辑（与阵列可编程，或阵列固定）

GAL (General Array Logic)：通用阵列逻辑

4.1 只读存储器（ROM）

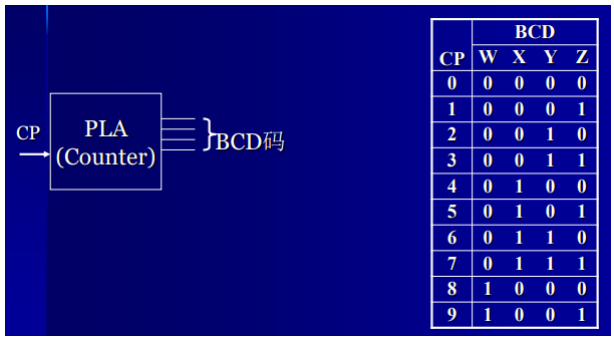
输入地址和存储信息一一对应
与阵列是译码器，包括全部最小项，信息表完全



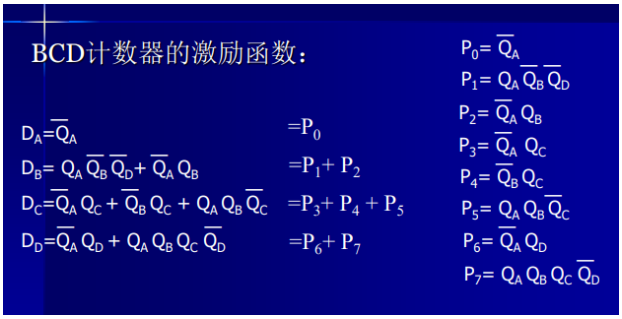
4.2 可编程逻辑阵列（PLA）

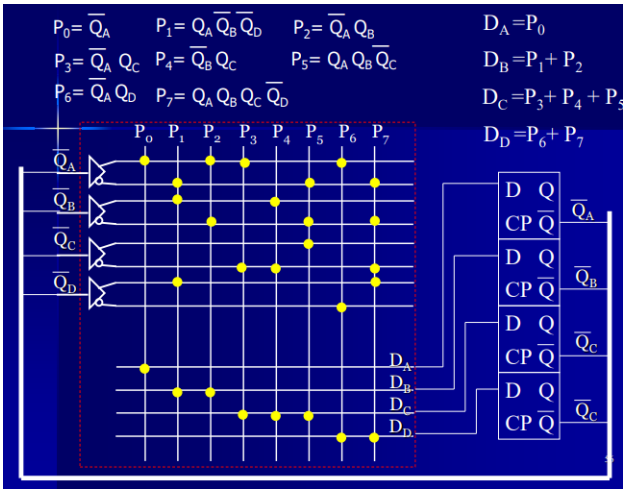
为了提高芯片的利用率，与阵列不一定产生所有的最小项，只需产生逻辑函数所需的乘积项即可
与阵列可编：形成P项（不是最小项）；或阵列可编；一组地址可选中多个P项
PLA的容量表示：（输入数）X（P项数）X（输出数）

举例：用PLA和D触发器组成BCD计数器

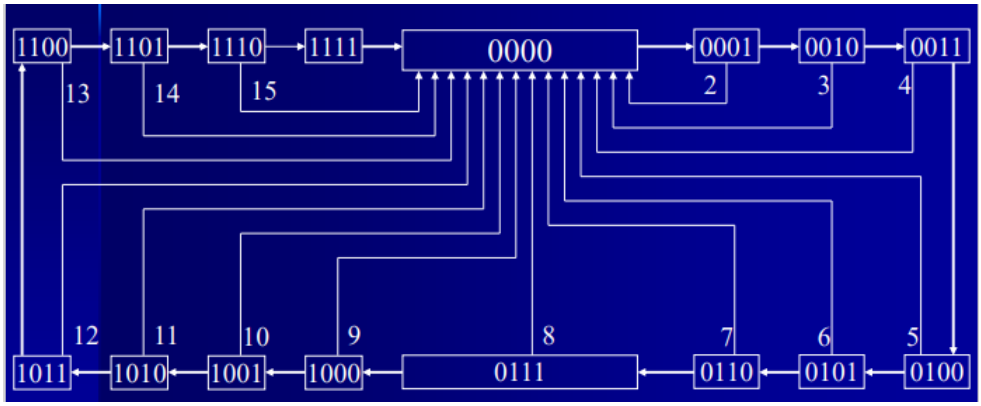


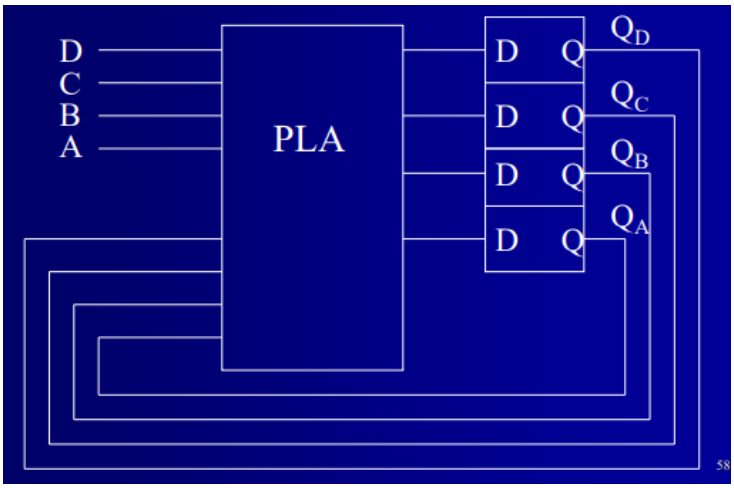
	BCD			
CP	W	X	Y	Z
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1





举例：用PLA实现4为可变模数计数器





4位可变模数计数器激励方程

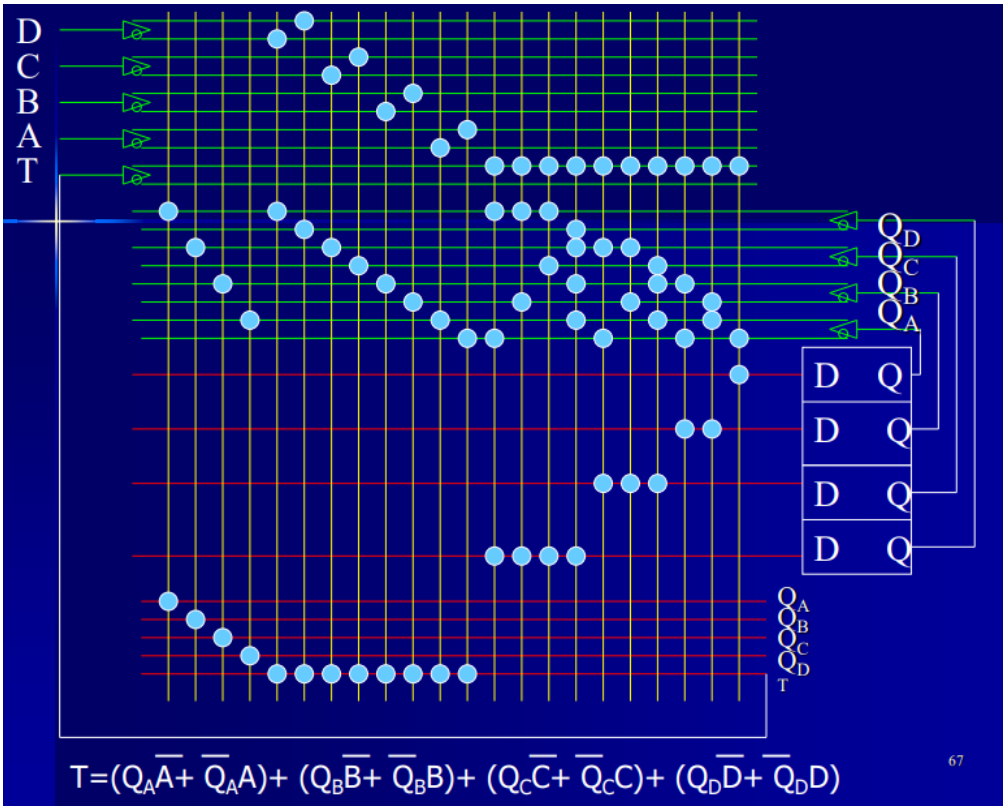
$D_A=\overline{Q_A}$ $D_A=\overline{Q_A}T$

$D_B=Q_B\oplus Q_A$ $D_B=(Q_B\oplus Q_A)T$

$D_C=Q_C\oplus Q_BQ_A$ $D_C=(Q_C\oplus Q_BQ_A)T$

$D_D=Q_D\oplus Q_CQ_BQ_A$ $D_D=(Q_D\oplus Q_CQ_BQ_A)T$

$T=(Q_A\oplus A)+(Q_B\oplus B)+(Q_C\oplus C)+(Q_D\oplus D)$



4.3 可编程阵列逻辑（PAL）

与阵列可编程、或阵列固定
PAL是专用词，MMI公司的产品
不同的芯片可实现不同的逻辑，有些PAL只能实现组合逻辑电路，有些只能实现时序逻辑电路。
工艺：简化工艺，降低成本（熔丝工艺，一次编程）

4.4 通用阵列逻辑（GAL）

ROM、PLA、PAL共同存在的问题：不存在只用一种芯片，即可以实现组合逻辑电路，又可以实现时序逻辑电路；GAL是为解决这一问题而产生的芯片

工艺：电可擦除，多次编程（Lattice公司1985年专利）
结构：输出宏单元，更通用
与阵列可编程、或阵列固定
增加了输出控制（OLMC结构）、反馈结构、多次可编程、集成度提高

基本逻辑结构：输入、输出、与阵列、输出逻辑宏单元（OLMC）

1. 输出逻辑宏单元 / OLMC

（或阵列 + 触发器）的二选一