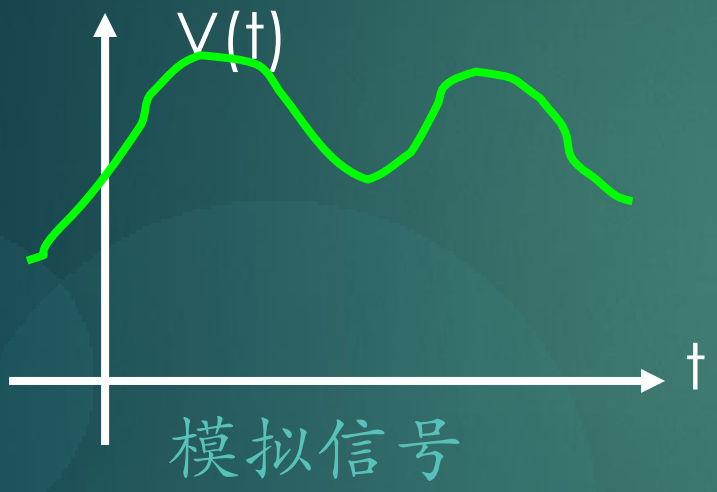
《数字逻辑》
Digital Logic

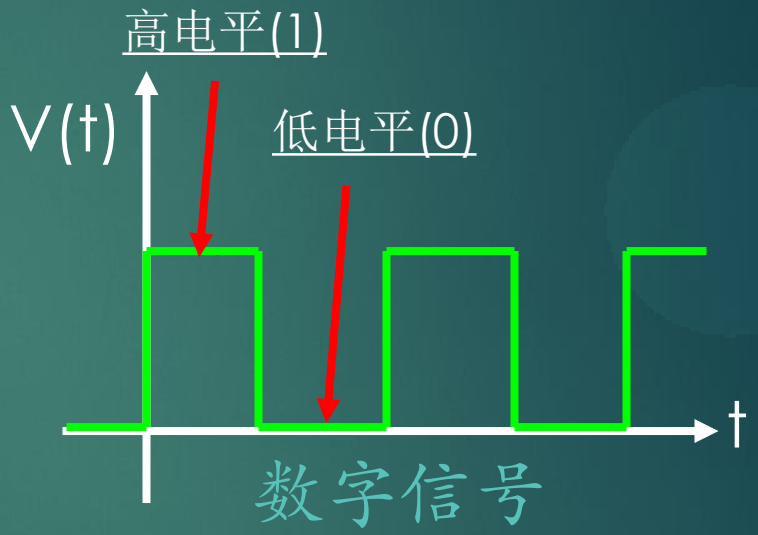
习题课

北京工业大学软件学院
王晓懿

信息的逻辑表示

- 数字信号和模拟信号的区别



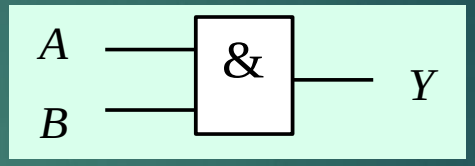


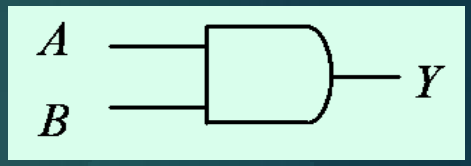
- 156个状态至少需要多少位二进制数码？

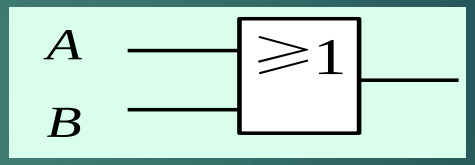
基本逻辑的电路表示

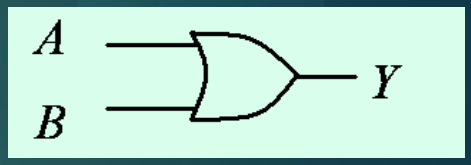
- 基本门电路的逻辑电路符号

  不同的符号表示



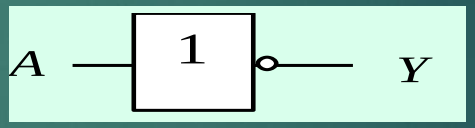




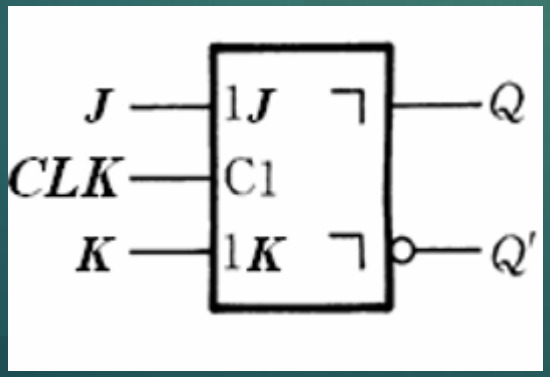


- 悬空的输入

  悬空为高“1”



例如：



逻辑代数

二、逻辑函数表示方法

常用逻辑函数的表示方法有：逻辑真值表（真值表）、逻辑函数式（逻辑式或函数式）、逻辑图、波形图、卡诺图及硬件描述语言。它们之间可以相互转换。

逻辑代数

- 对偶函数与反函数
  - 对偶函数

$$Y = A(B+C)+CD \quad \Rightarrow \quad Y^D = (A+BC)(C+D)$$

  - 反函数

$$Y = A(B+C)+CD \quad \Rightarrow \quad \bar{Y} = (\bar{A}+\bar{B}\bar{C})(\bar{C}+\bar{D})$$

最小项(MinTerm)

逻辑函数有n个变量，由它们组成的具有n个变量的乘积项中，每个变量以原变量或反变量的形式出现且仅出现一次，这个乘积项为最小项。N个变量有2^n个最小项。

例如：n=3，对A、B、C，有8个最小项

$\bar{A}\bar{B}\bar{C}$　$A\bar{B}\bar{C}$　$\bar{A}B\bar{C}$　$AB\bar{C}$

$\bar{A}\bar{B}C$　$A\bar{B}C$　$\bar{A}BC$　ABC

最小项(续)

- 对任意最小项，只有一组变量取值使它的值为1，其他取值使该最小项为0
- 为方便起见，将最小项表示为m_i

  n=3的8个最小项为：

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

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

最小项(续)

- 任何逻辑函数均可表示为唯一的一组最小项之和的形式，称为标准的与或表达式
- 某一最小项不是包含在F的原函数中，就是包含在F的反函数中
- 例：

$$F = \overline{A}B + BC + A\overline{B}\overline{C}$$

$$= \overline{A}B(C + \overline{C}) + (A + \overline{A})BC + A\overline{B}\overline{C}$$

$$= \overline{A}BC + \overline{A}B\overline{C} + ABC + A\overline{B}\overline{C}$$

$$= m_6 + m_2 + m_7 + m_1$$

$$= \sum m^3(1,2,6,7)$$

BACK

最大项(MaxTerm)

- n个变量组成的或项，每个变量以原变量或反变量的形式出现且仅出现一次，则称这个或项为最大项

  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(续)

- 对任意一个最大项，只有一组变量取值使它的值为0，而变量的其他取值使该项为1
- 将最大项记作M_i
- 任何一个逻辑函数均可表示为唯一的一组最大项之积，称为标准的或与表达式
- n个变量全体最大项之积必为“0”
- 某个最大项不是含在F的原函数中，就是在F的反函数中

最大项(续)

例如：

$$F = (A + B) \bullet (\overline{A} + B + C)$$

$$= [A + B + (C \bullet \overline{C})] \bullet (\overline{A} + B + C)$$

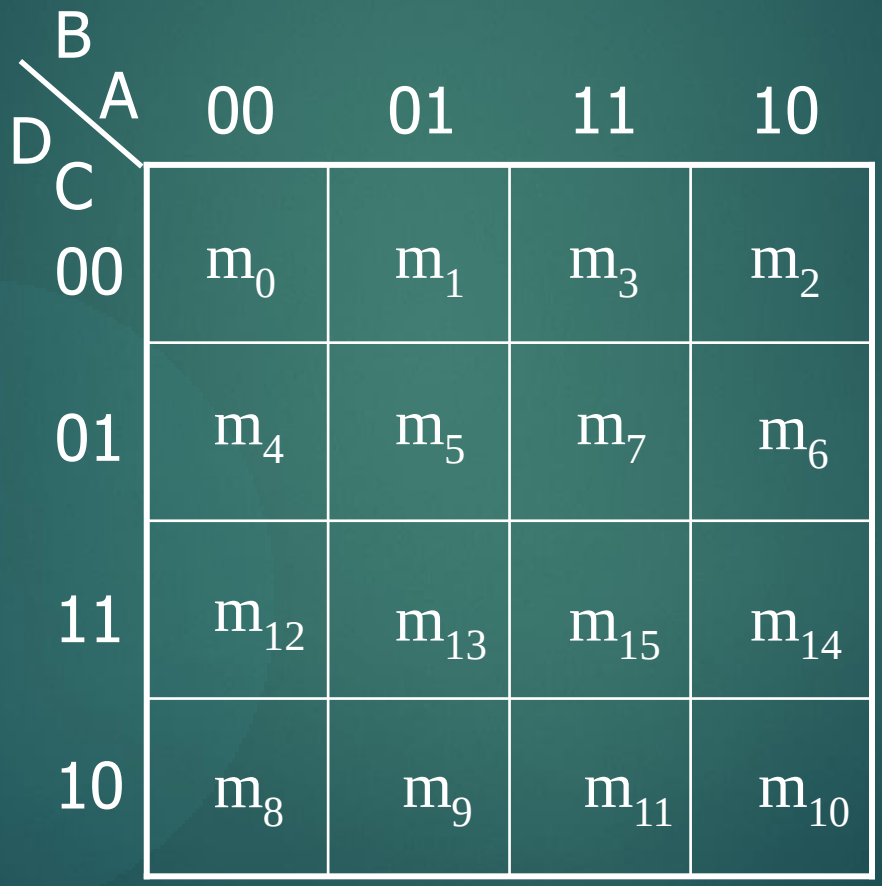
$$= (A + B + C) \bullet (A + B + \overline{C}) \bullet (\overline{A} + B + C)$$

$$= M_0 + M_4 + M_1$$

$$= \prod M^4(0,1,4)$$

BACK

4变量Karnaugh Map



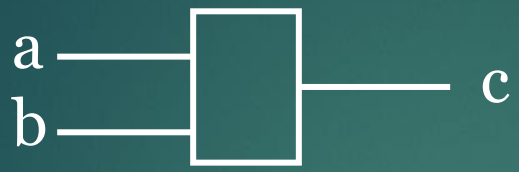
数字逻辑电路的电气特性

物理上的

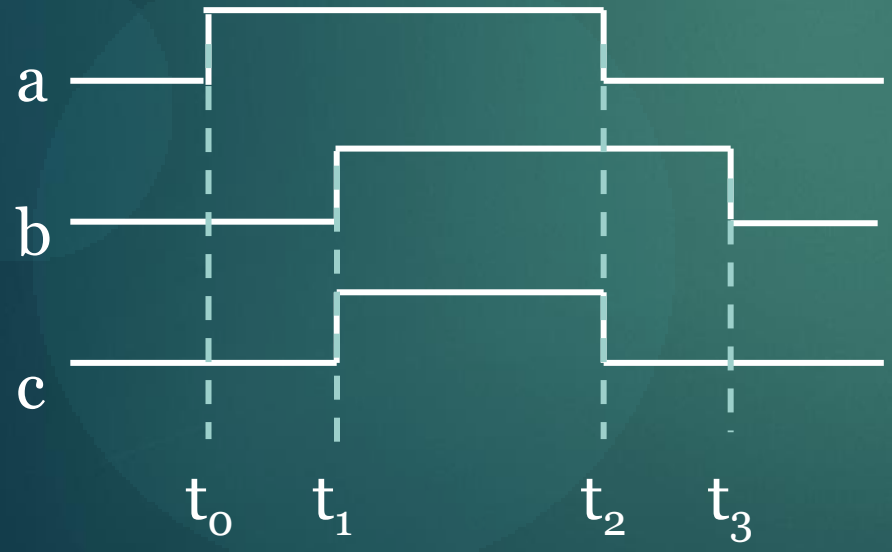
而不是逻辑上的

- 逻辑电压电平
- 直流噪声容限
- 扇入/扇出
- **延时/速度**
- **功耗**
- **噪声**
- 漏极开路输出、三态输出

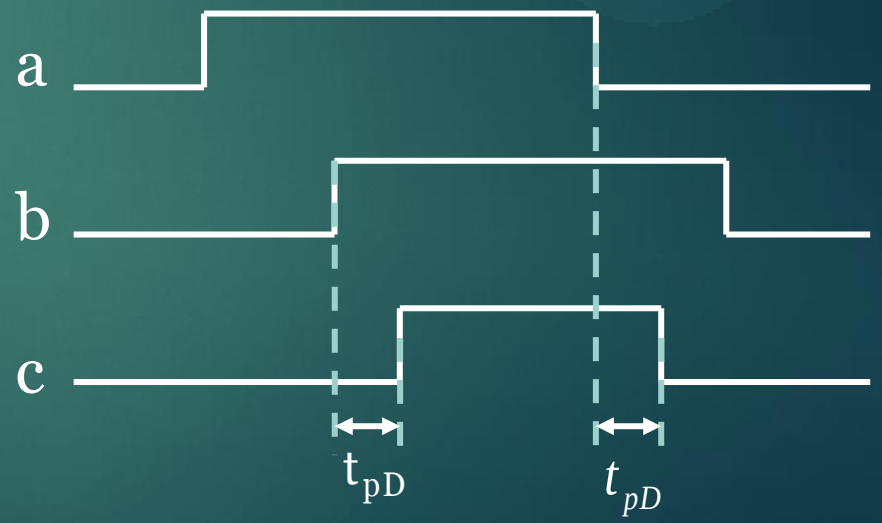
延迟（延时，时延，Delay）



理想情况：门电路没有延迟



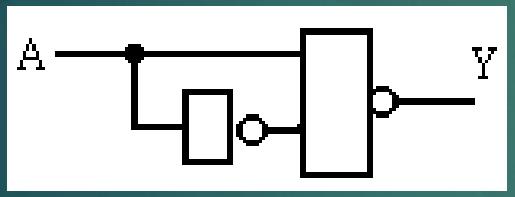
实际情况：门电路存在延迟



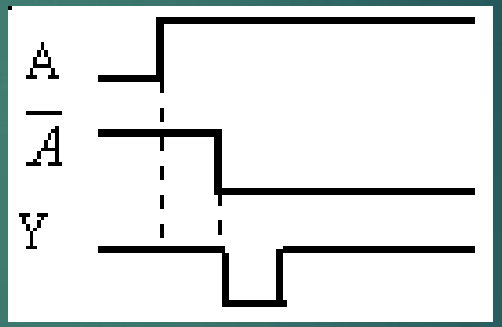
竞争冒险问题

门电路的传输延迟造成竞争冒险问题。

二输入AND门（OR门）的输入为A和 $\overline{A}$时，$\overline{A}$滞后于A，则Y会出现尖峰信号。



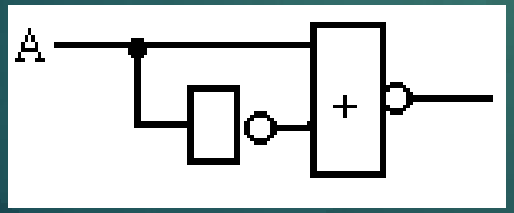
与非门上升沿有尖峰



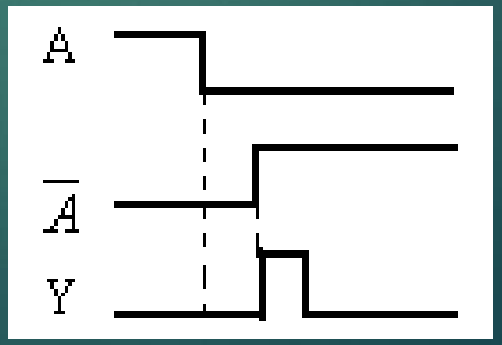
理想情况：

$Y = A\overline{A} = 1$

负向尖峰



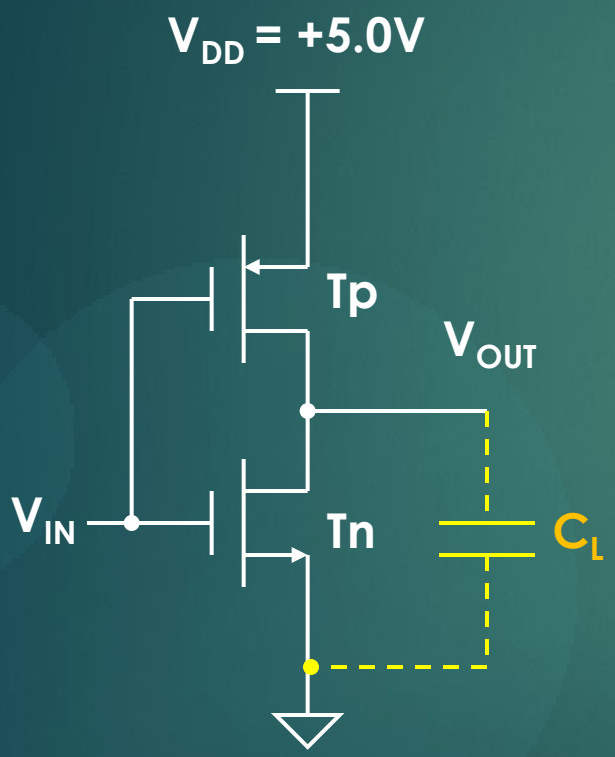
或非门下降沿有尖峰



正向尖峰

功率损耗

分为：静态功耗、动态功耗



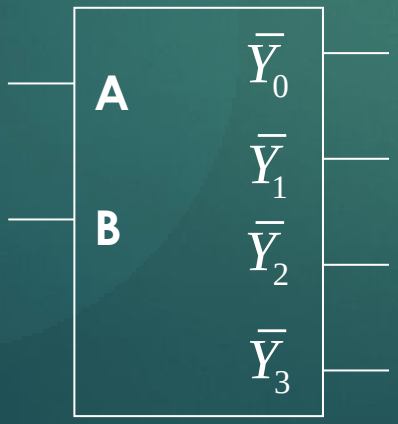
动态功耗的来源：

- 两个管子瞬间同时导通产生的功耗 P_T
- 对负载电容充、放电所产生的功耗 P_L

组合逻辑电路

- 组合逻辑电路功能分析方法
- 组合逻辑电路功能设计方法
- 常见组合逻辑器件的表示
  - 符号友好的表示法
- 译码器

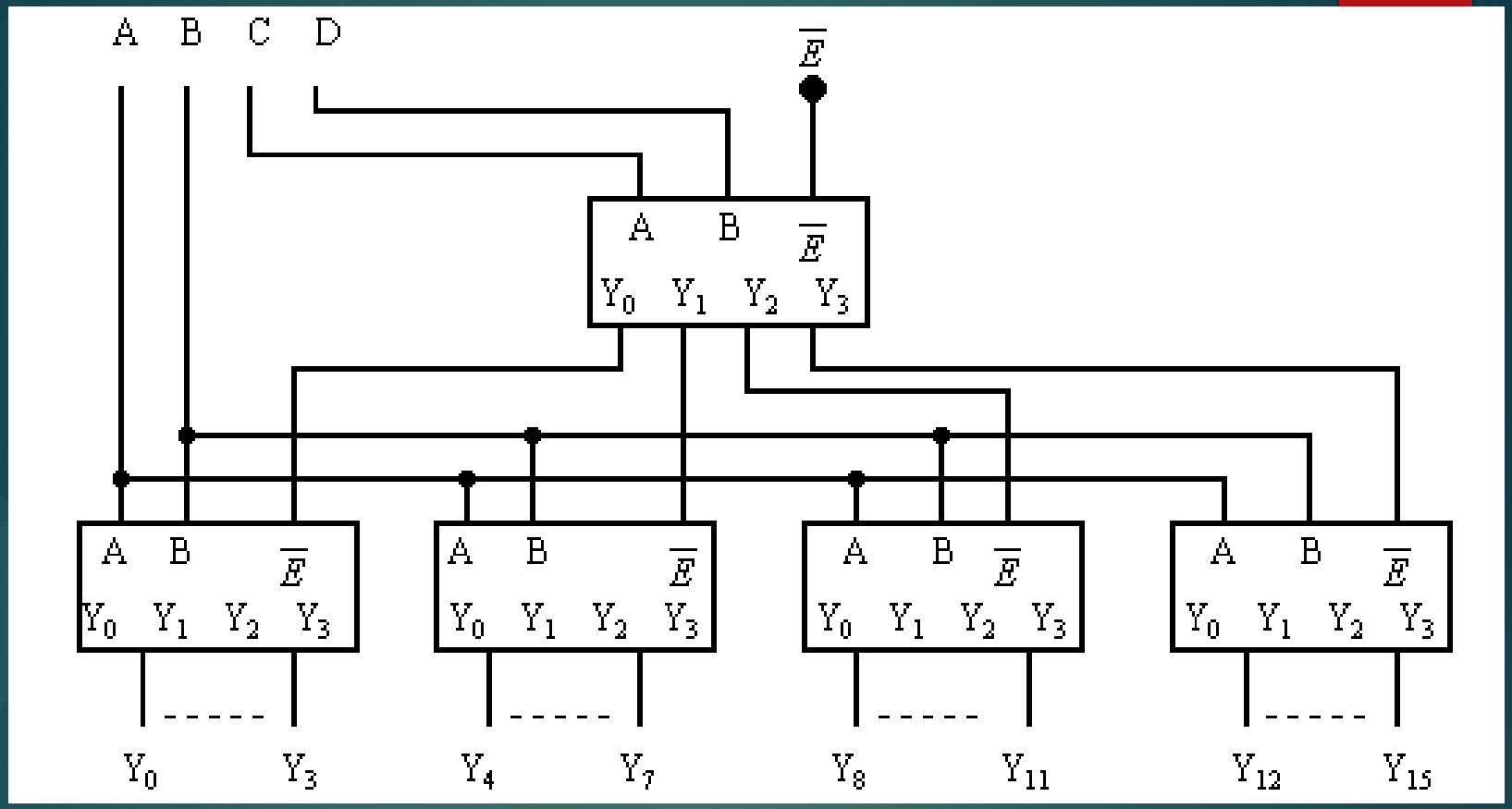
符号友好的表示法



输出表达式

$$\begin{cases} \bar{Y}_0 = \overline{\overline{A}\,\overline{B}} \\ \bar{Y}_1 = \overline{\overline{A}B} \\ \bar{Y}_2 = \overline{A\overline{B}} \\ \bar{Y}_3 = \overline{AB} \end{cases}$$

“小模块组成大模块”

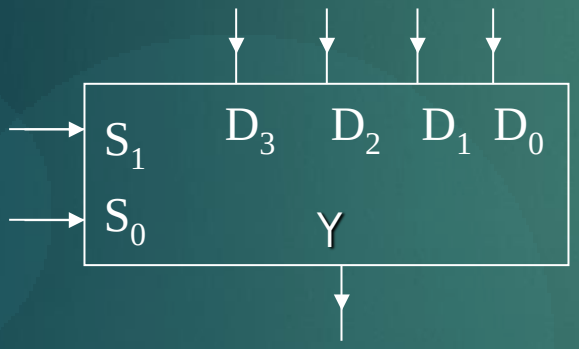


5片2－4译码器构成4－16译码器。第一层的一个译码器用作选片。$\overline{E}$=0时，C D＝00时选中左边一片，译出Y_0…Y_3;依此类推。

组合逻辑电路

- 数据选择器

逻辑框图



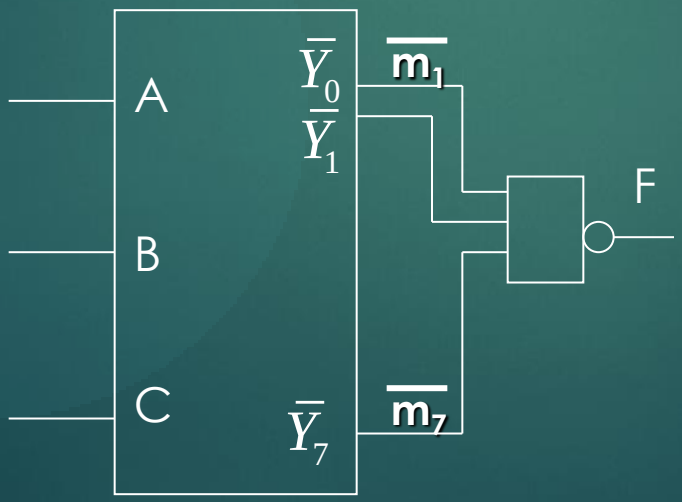
真值表

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

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

译码器实现逻辑函数

- **译码器输出可以看成是N个输入变量组成的2^N个最小项,再经一级与非门,组成“与非-与非”逻辑，既可表达“与-或”表达式。**
- 例如：$F=\overline{A}\,\overline{B}C+\overline{A}B\overline{C}+ABC=m_1+m_2+m_7$

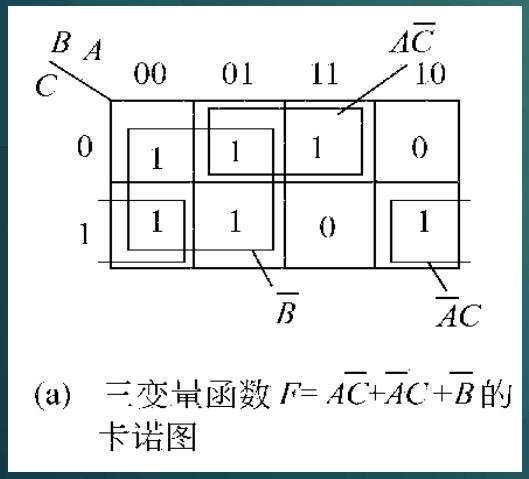


数据选择器实现逻辑函数

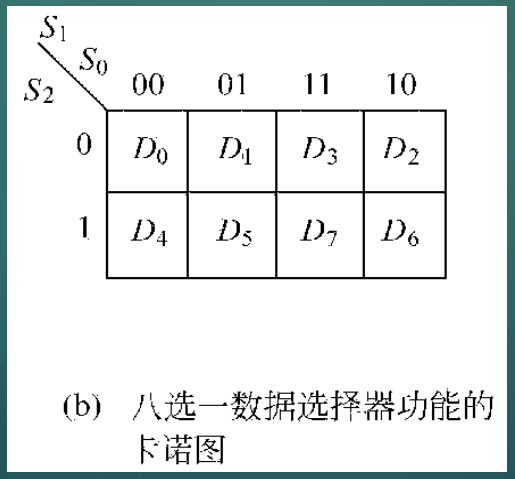
数据选择器: 逻辑结构就是与-或表达式。

数据选择器可以看成是N个控制端选择2^N个最小项组成的“与-或”表达式。选择某些输入为“1”，就是选中这些最小项组成逻辑函数。逻辑变量接到选择控制端，逻辑表达式中包含的最小项取“1”，其余取“0”。

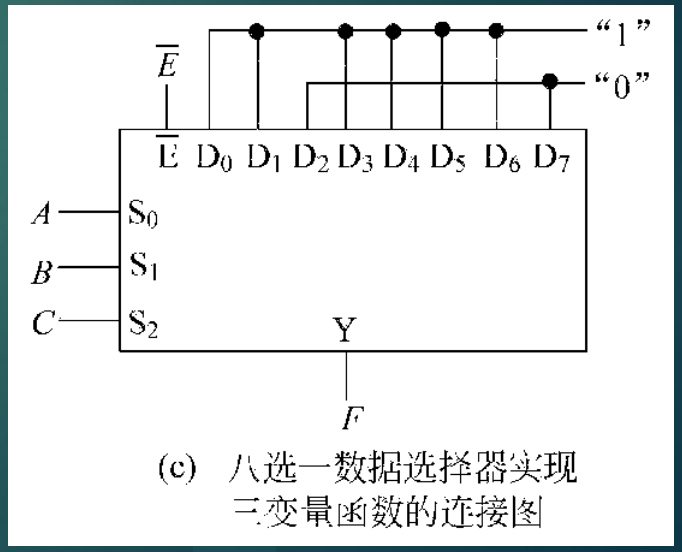
例如，用八选一数据选择器实现函数: $F = \overline{B} + \overline{A}C + A\overline{C}$



(a) 三变量函数 $F = A\overline{C} + \overline{A}C + \overline{B}$ 的卡诺图



(b) 八选一数据选择器功能的卡诺图



(c) 八选一数据选择器实现三变量函数的连接图

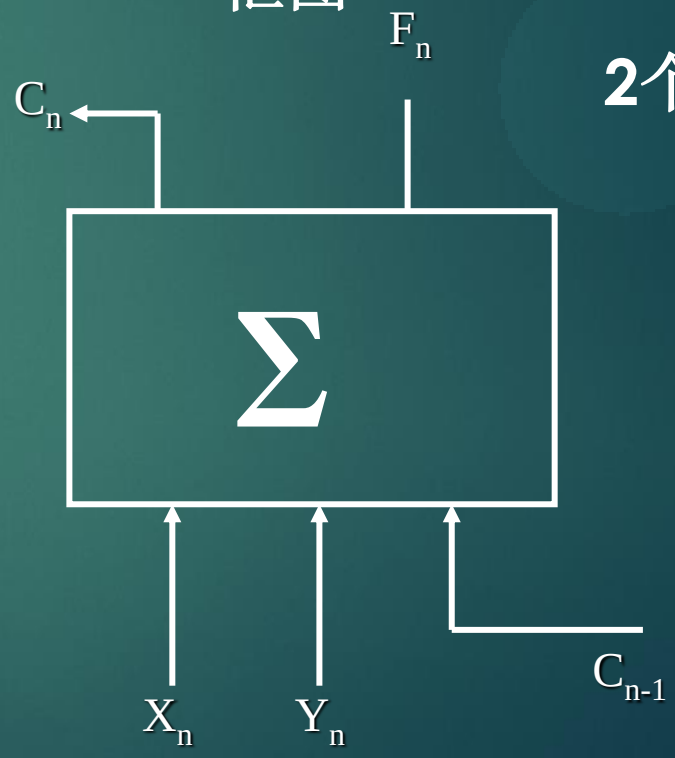
组合逻辑电路

- 全加器 （全减器？）

真值表

输入			输出	
X_n	Y_n	C_{n-1}	F_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

框图



3个输入

2个输出

全加器的逻辑表达式

C_{n-1} \ X_nY_n	00	01	11	10
0	0	1	0	1
1	1	0	1	0

F_n

C_{n-1} \ X_nY_n	00	01	11	10
0	0	0	1	0
1	0	1	1	1

C_n

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

$$F_n = \overline{X}_n\overline{Y}_nC_{n-1} + \overline{X}_nY_n\overline{C}_{n-1} + X_n\overline{Y}_n\overline{C}_{n-1} + X_nY_nC_{n-1}$$

$$= \overline{C}_{n-1}(\overline{X}_nY_n + X_n\overline{Y}_n) + C_{n-1}(\overline{X}_n\overline{Y}_n + X_nY_n)$$

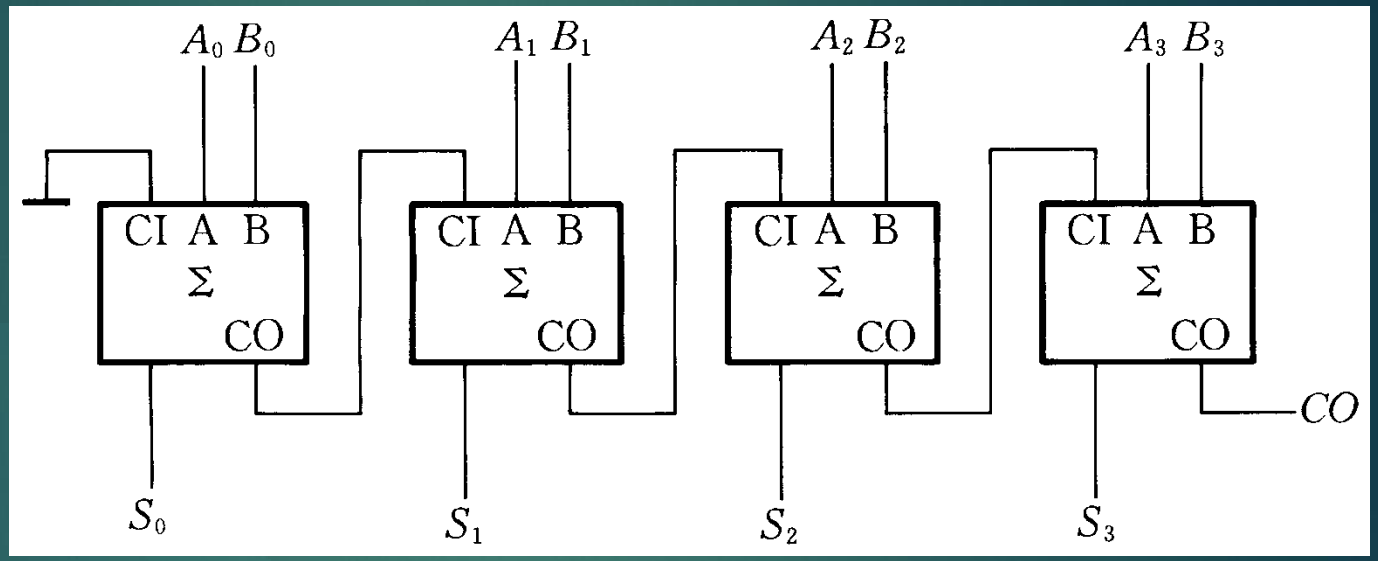
$$= C_{n-1} \oplus (X_n \oplus Y_n)$$

多位加法器

1. 4位串行进位加法器

优点：简单

缺点：慢



$$(CI)_i = (CO)_{i-1}$$

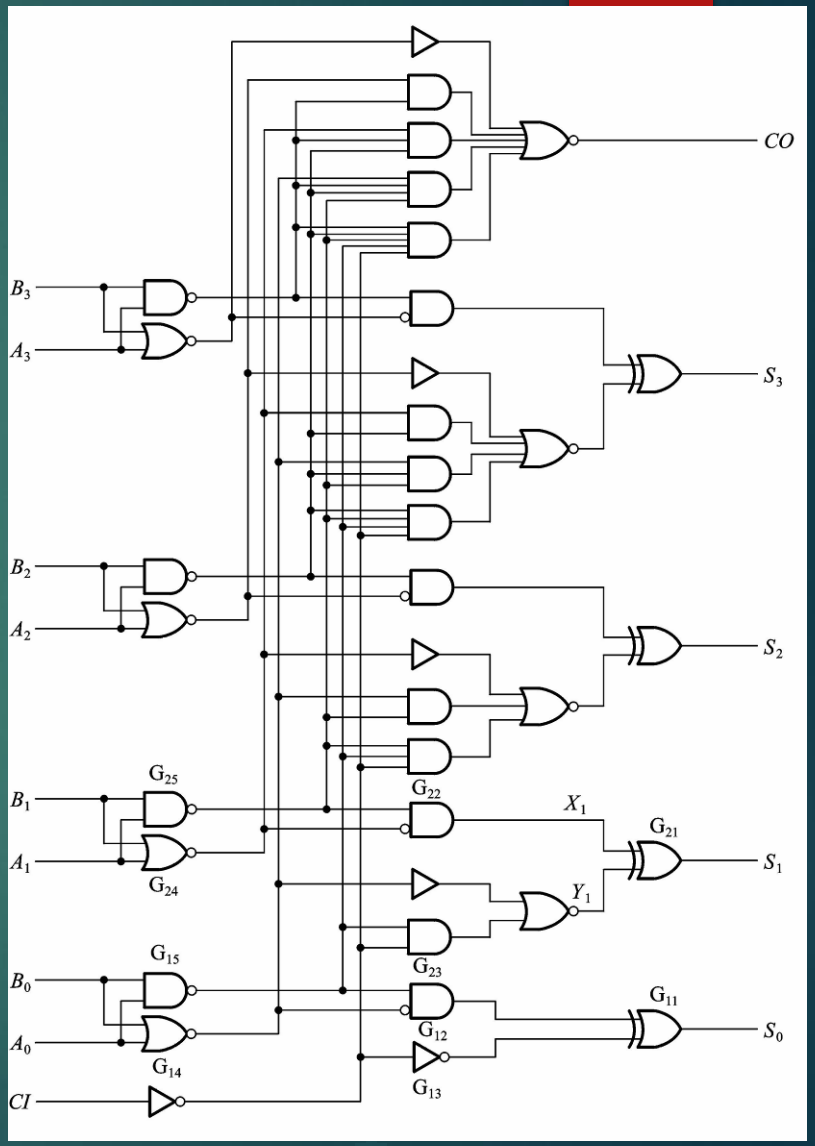
$$S_i = A_i \oplus B_i \oplus (CI)_i$$

$$(CO)_i = A_i B_i + (A_i + B_i)(CI)_i$$

2. 并行进位（超前进位)加法器

优点：快，每1位的和及最后的进位基本同时产生。

缺点：电路复杂。



连续赋值

```
assign R = X | (Y & ~Z);

assign r = &X;

assign R = ( a == 1'b0) ? X : Y;

assign P = 8'hff;

assign P = X * Y;

assign P[7:0] = { 4{X[3]}, X[3:0] };

assign {cout, R} = X + Y + cin;

assign Y = A << 2;

assign Y = {A[1], A[0], 1'b0, 1'b0};
```

位操作符

reduction操作符

算术操作符

扩展操作符和拼接操作符

移位操作符

Verilog中常用运算符

Symbol	Operation	Symbol	Operation
+	binary addition		
–	binary subtraction		
&	bitwise AND	&&	logical AND
|	bitwise OR	||	logical OR
^	bitwise XOR		
~	bitwise NOT	!	logical NOT
= =	equality		
>	greater than		
<	less than		
{}	concatenation		
?:	conditional		

“always”语句回顾

- always 语句
  - 总是等待触发信号的变化
  - 触发信号变化即开始执行

```
module and_gate (output reg out,
                 input in1, in2);
  reg result;
  always @(in1, in2) begin
      result = in1 & in2;
      out = result;
  end
endmodule
```

always块中左端项必须为reg类型，但是并不一定就是寄存器！

声明了触发信号，即语句执行的时机

always触发事件（敏感列表）

- 边沿触发posedge, negedge表明always语句描述的是时序逻辑；
- 与组合逻辑不同，时序逻辑中敏感列表会影响电路综合结果；
- 一个变量只在一个always块中设置，避免竞争冒险；

同步重置的D触发器

```
module D_FF (output reg Q,
    input D, clk, set, rst);
  always @ ( posedge clk)
    if (rst)
      Q <= 1'b0;
    else if (set)
      Q <= 1'b1;
    else
      Q <= D;
endmodule
```

异步重置的D触发器

```
module D_FF (output reg Q,
    input D, clk, set, rst);
  always @ ( posedge clk, posedge set,
    negedge rst)
    if (!rst)
      Q <= 1'b0;
    else if (set)
      Q <= 1'b1;
    else
      Q <= D;
endmodule
```

reg类型

- always块中赋值的左端必须是reg类型
- always块中赋值的输出变量也必须是reg类型
- 在always触发之前reg类型数据保持不变

```
module and_gate (output reg out,
                 input in1, in2);
  reg result;
  always @(in1, in2) begin
      result = in1 & in2;
      out = result;
  end
endmodule
```

不一定是寄存器！

阻塞式赋值和非阻塞式赋值

- always块中赋值分为阻塞式赋值和非阻塞式赋值

- 阻塞式赋值：=

赋值完成前不往下执行

```
always @(*) begin
    x = a | b;
    y = a ^ b ^ c;
    z = b & ~c;
end
```

- 非阻塞式赋值：=

右端表达式同时开始求值，时间步长结束后同时赋值

```
always @(*) begin
    x <= a | b;
    y <= a ^ b ^ c;
    z <= b & ~c;
    //时间步长结束，才开始赋值
end
```

有时候两种赋值逻辑综合得到的结果完全不同！

触发器

- JK触发器
- D触发器
- 下降沿触发，上升沿触发
- Set和Reset信号
- 时序电路的输出波形
  - 一次触发

触发器

- JK触发器和D触发器

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

CLK	J	K	Q^*
×	×	×	Q
↓	0	0	Q
↓	0	1	0
↓	1	0	1
↓	1	1	Q'

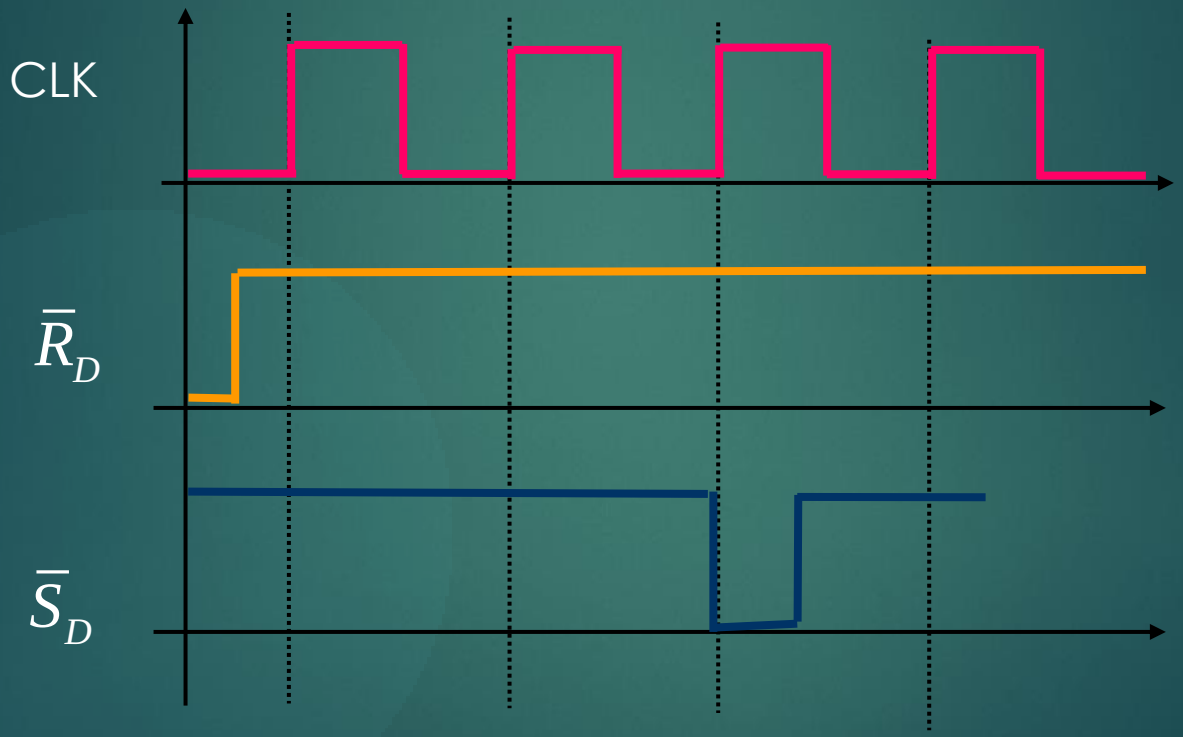
Q^*=D

特性表

CLK	D	Q^*
0	×	Q
↑	0	0
↑	1	1

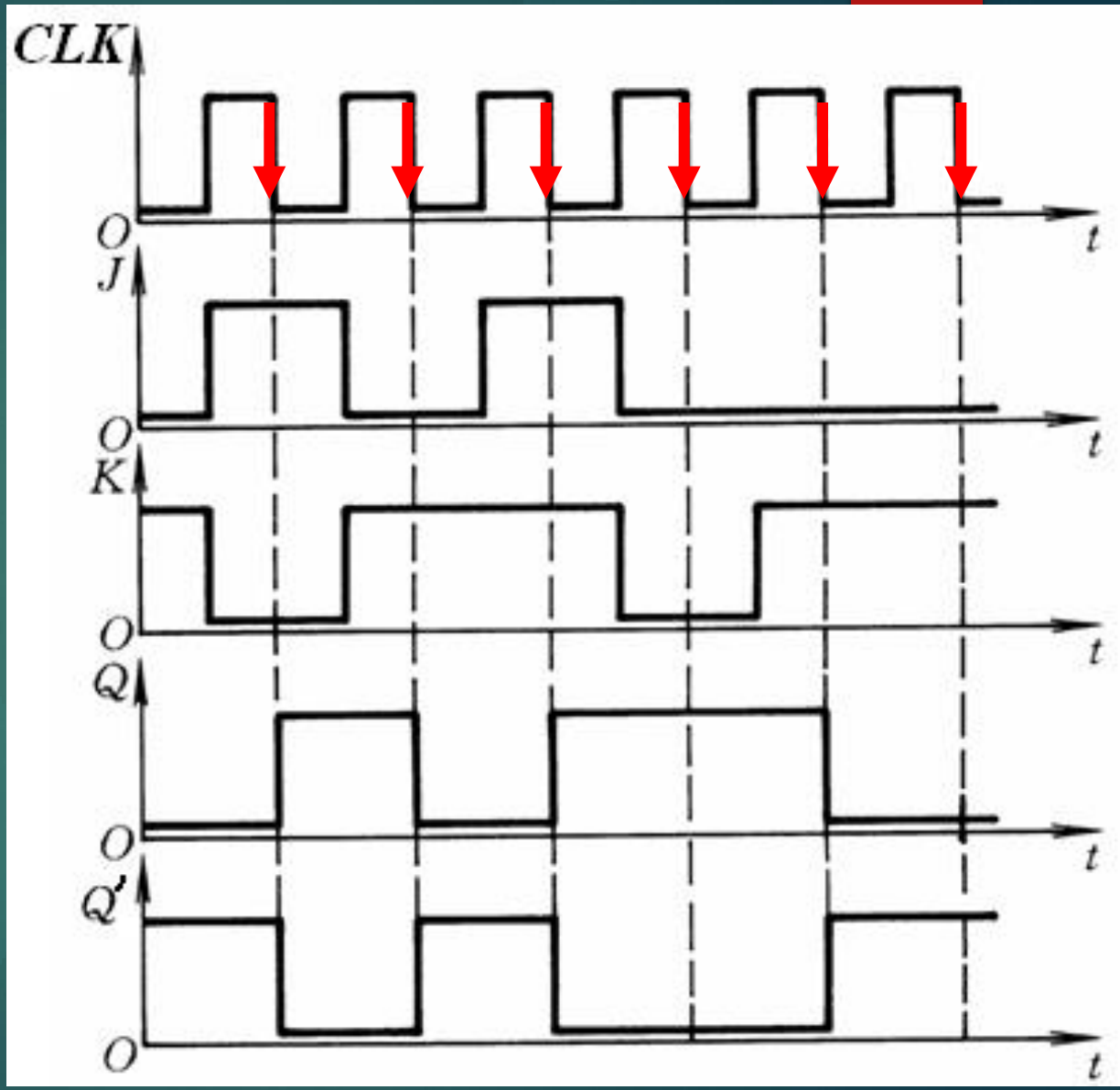
Reset和Set

1. 异步置0和置1
2. 为“0”时置0或者置1



JK触发器

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



Verilog描述触发器

```
module D_latch (output reg Q,
                input D, enable);
  always @ (enable or D)
    if (enable) Q <= D;
endmodule
```

```
module D_FF (output reg Q,
             input D, Clk);
  always @ ( posedge Clk)
    Q <= D;
endmodule
```

```
module JKFF ( output reg Q,
      input J, K, Clk, rst);
  wire JK;
  assign JK = (J & ~Q) | (~K & Q);
  DFF JK1 (Q, JK, Clk, rst);
endmodule
```

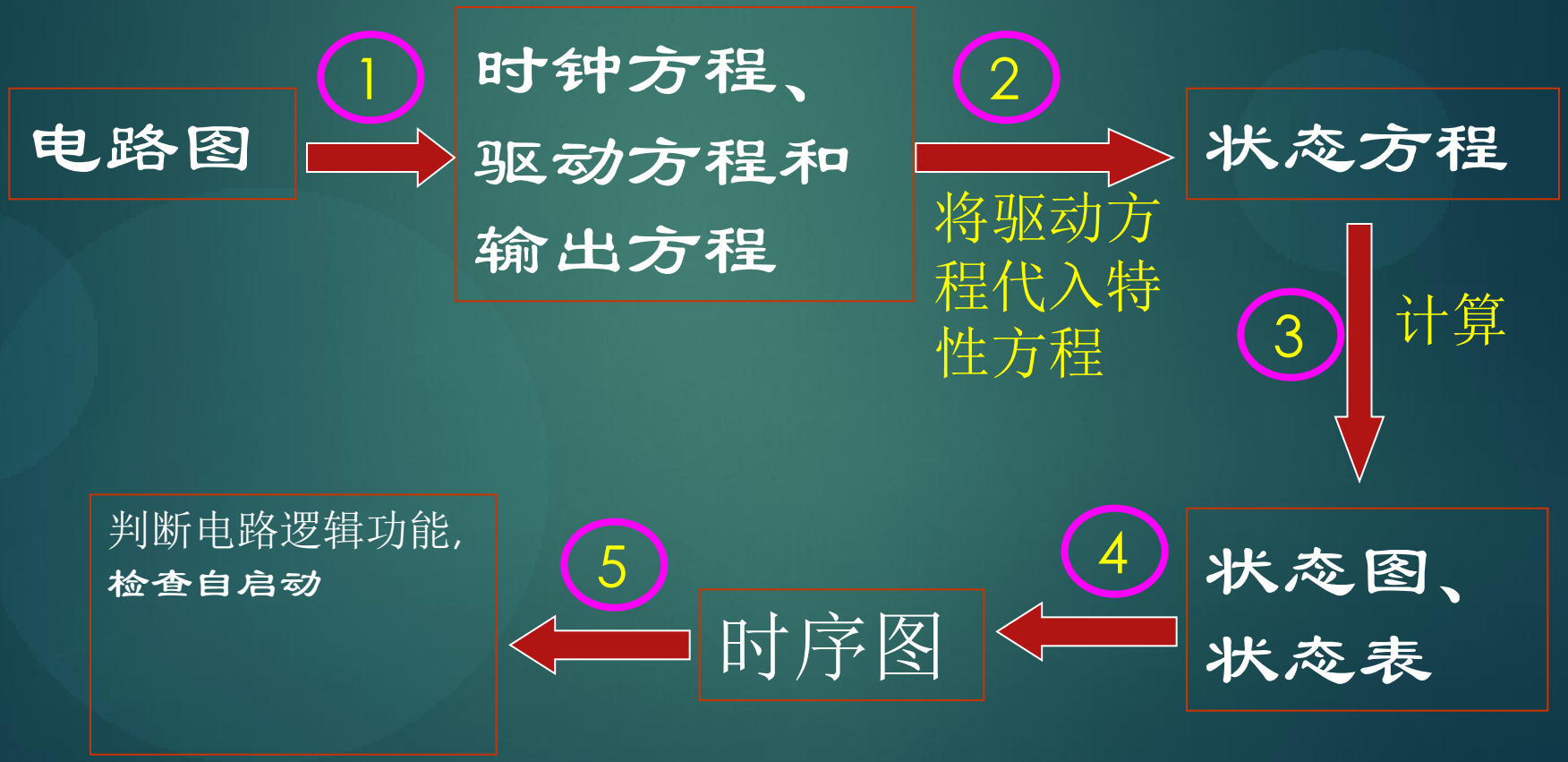
```
module JK_FF (output reg Q, output Q_b,
                        input J, K, Clk);
  assign Q_b = ~ Q ;
  always @ ( posedge Clk)
    case ({J,K})
      2'b00: Q <= Q;
      2'b01: Q <= 1'b0;
      2'b10: Q <= 1'b1;
      2'b11: Q <= !Q;
    endcase
endmodule
```

时序逻辑电路

- 同步时序电路和异步时序电路对比
- 时序逻辑电路功能分析方法
- 时序逻辑电路的设计方法
- 计数器
  - 74LS161和74LS160
  - 重置法，置数法
  - 使用N位计数器组成M位计数器

时序逻辑电路的分析方法

时序电路的分析步骤：



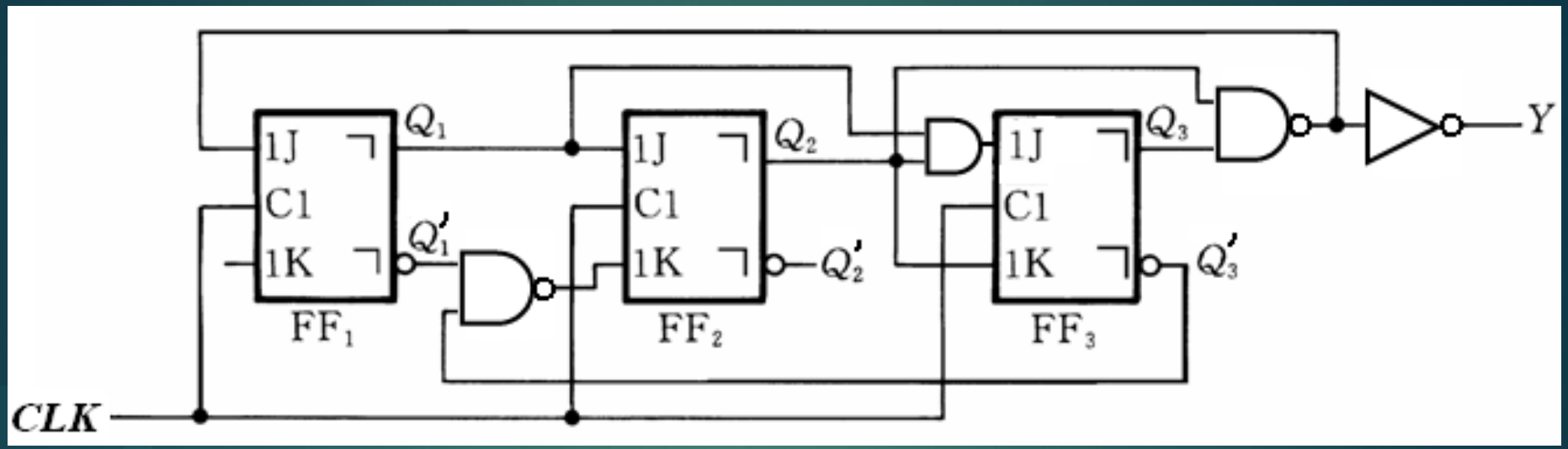
几个概念

有效状态：在时序电路中，凡是被利用了的状态。
有效循环：有效状态构成的循环。

无效状态：在时序电路中，凡是没有被利用的状态。
无效循环：无效状态若形成循环，则称为无效循环。

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

例



解: ①写方程组

驱动方程

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

同步时序电路，时钟方程省去。

输出方程 $Y = Q_2 \cdot Q_3$

②求状态方程

将驱动方程代入JK触发器的特性方程

中得电路的状态方程:

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

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

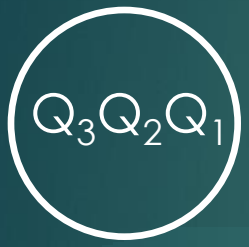
③计算、列状态转换表

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

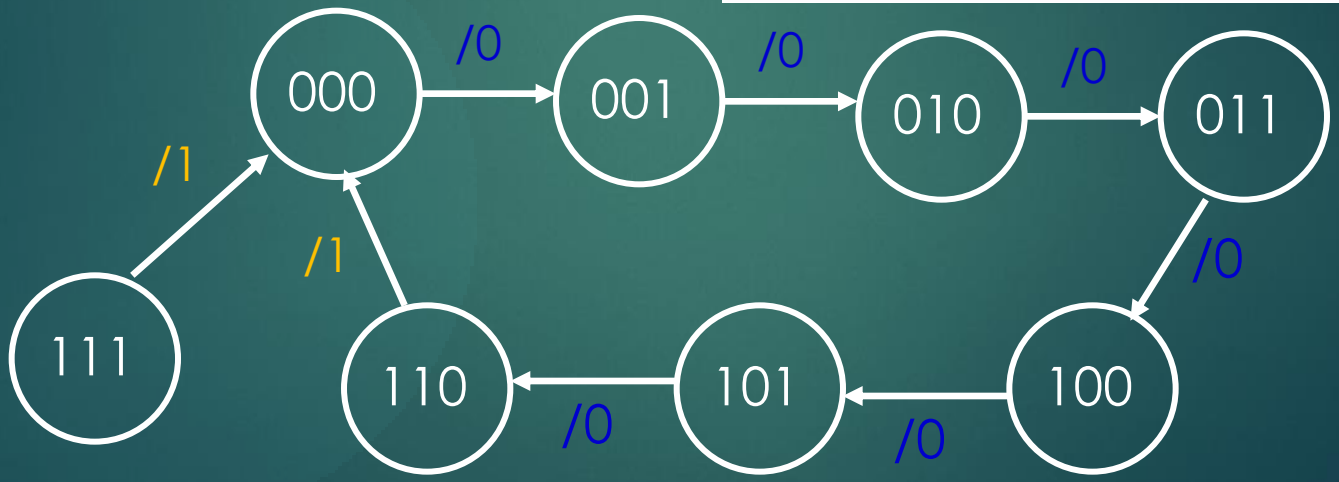
现态			次态			输出
Q_3	Q_2	Q_1	Q_3^*	Q_2^*	Q_1^*	Y
0	0	0	0	0	1	0
0	0	1	0	1	0	0
0	1	0	0	1	1	0
0	1	1	1	0	0	0
1	0	0	1	0	1	0
					0	0
					0	1
					0	1

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

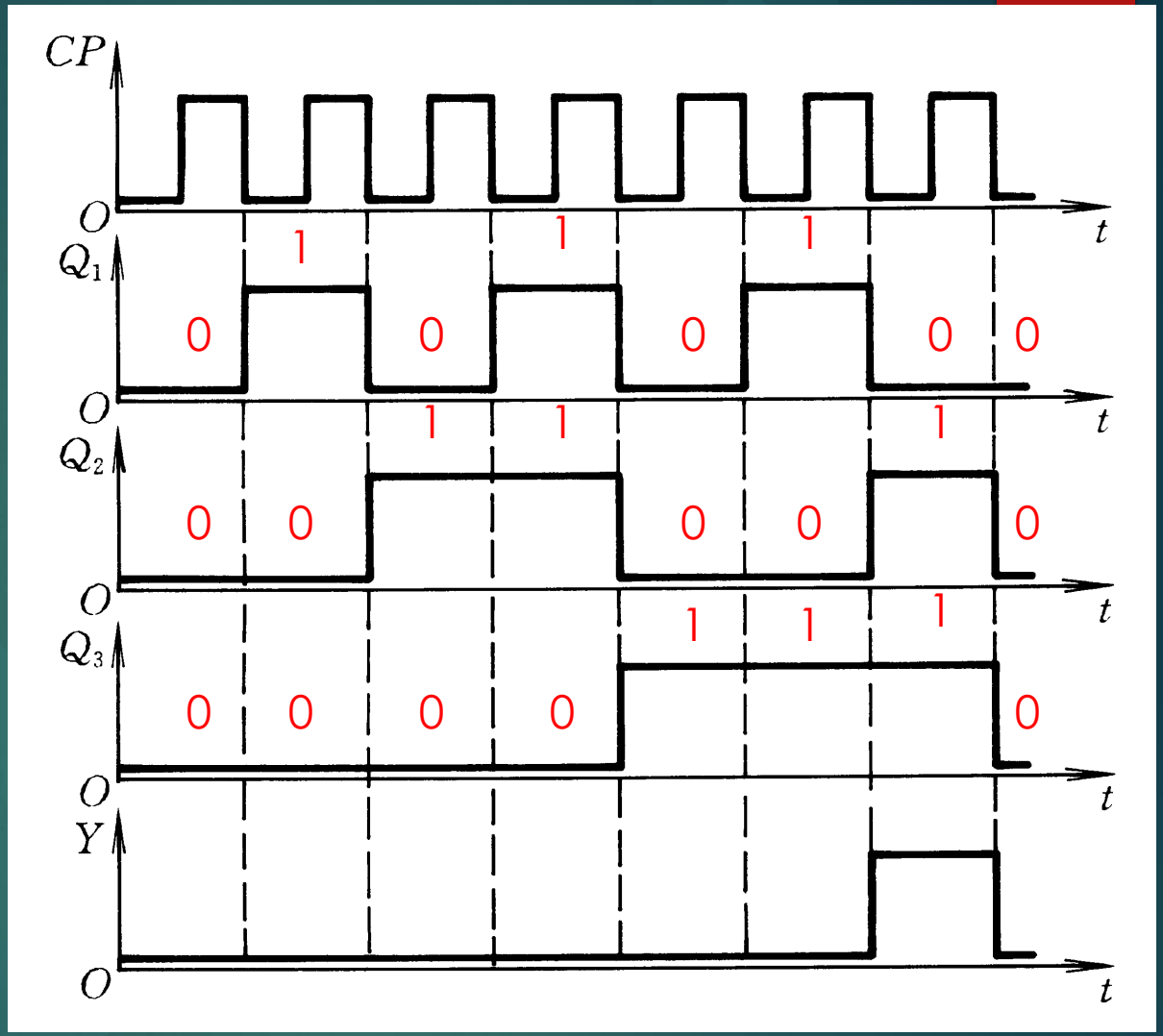
画状态转换图



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



④作时序图



⑤说明电路功能

这是一个同步七进制加法计数器，能自启动。

时序逻辑电路

- 计数器
  - 74LS161（2^N进制）和74LS160 （10进制）
  - 置零法，置数法
  - 使用N位计数器组成M位计数器

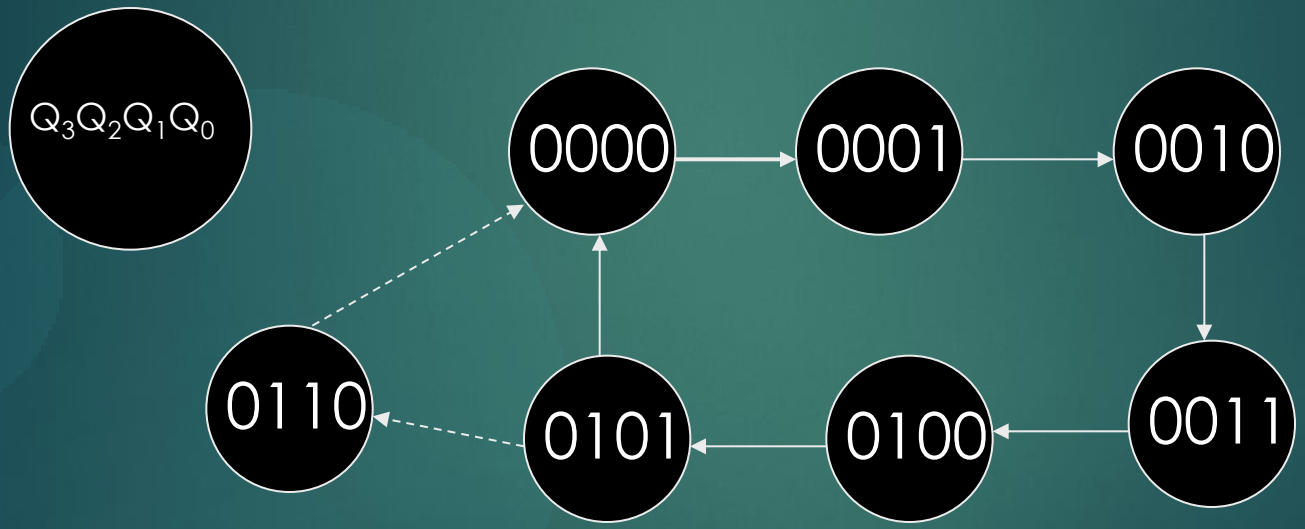
当M<N时，一片N进制计数器即可实现

例：利用同步十进制计数器构成同步六进制计数器

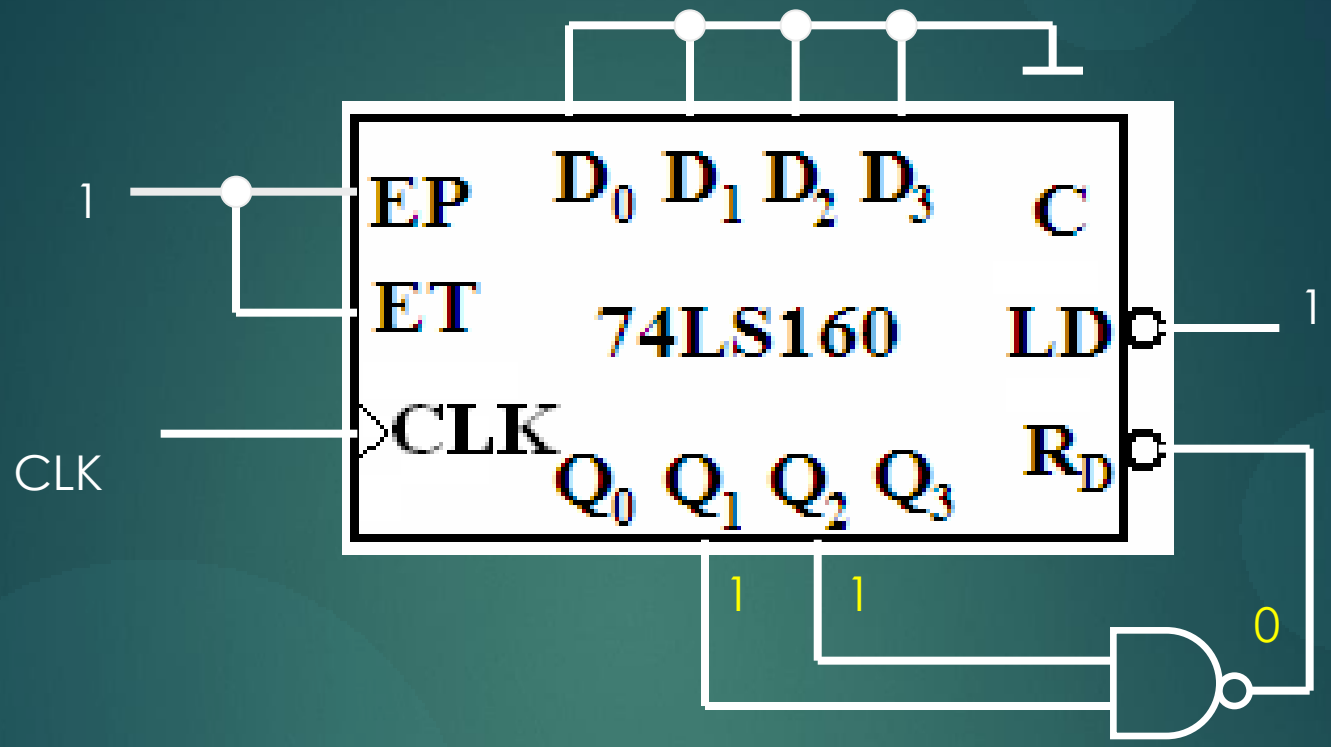
解：

置零法

74LS160具有异步清零功能

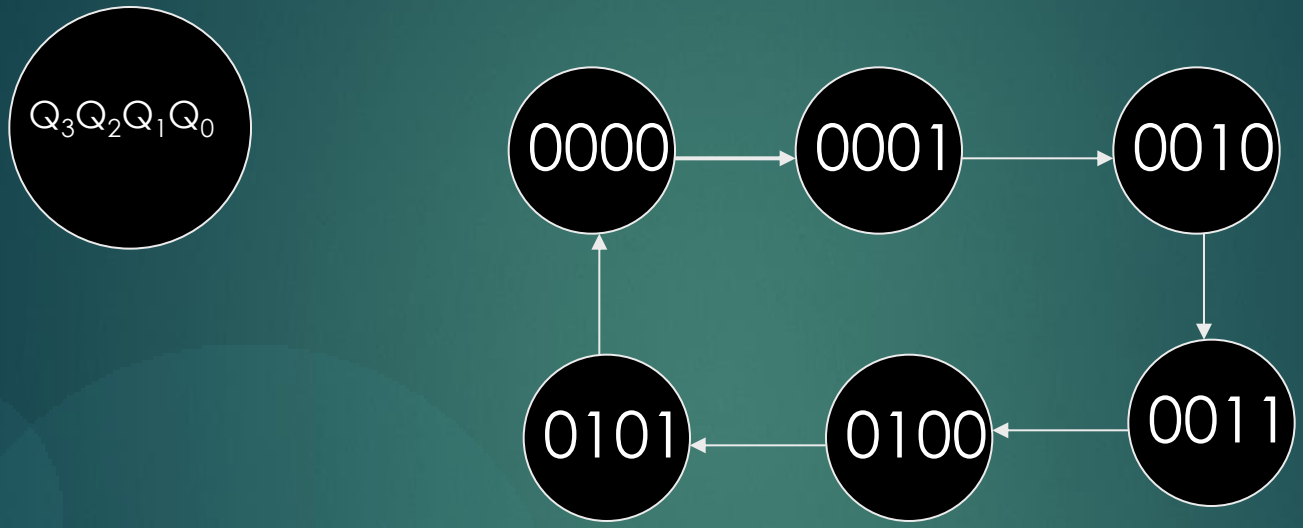


$$R'_D = (Q_2 \cdot Q_1)'$$

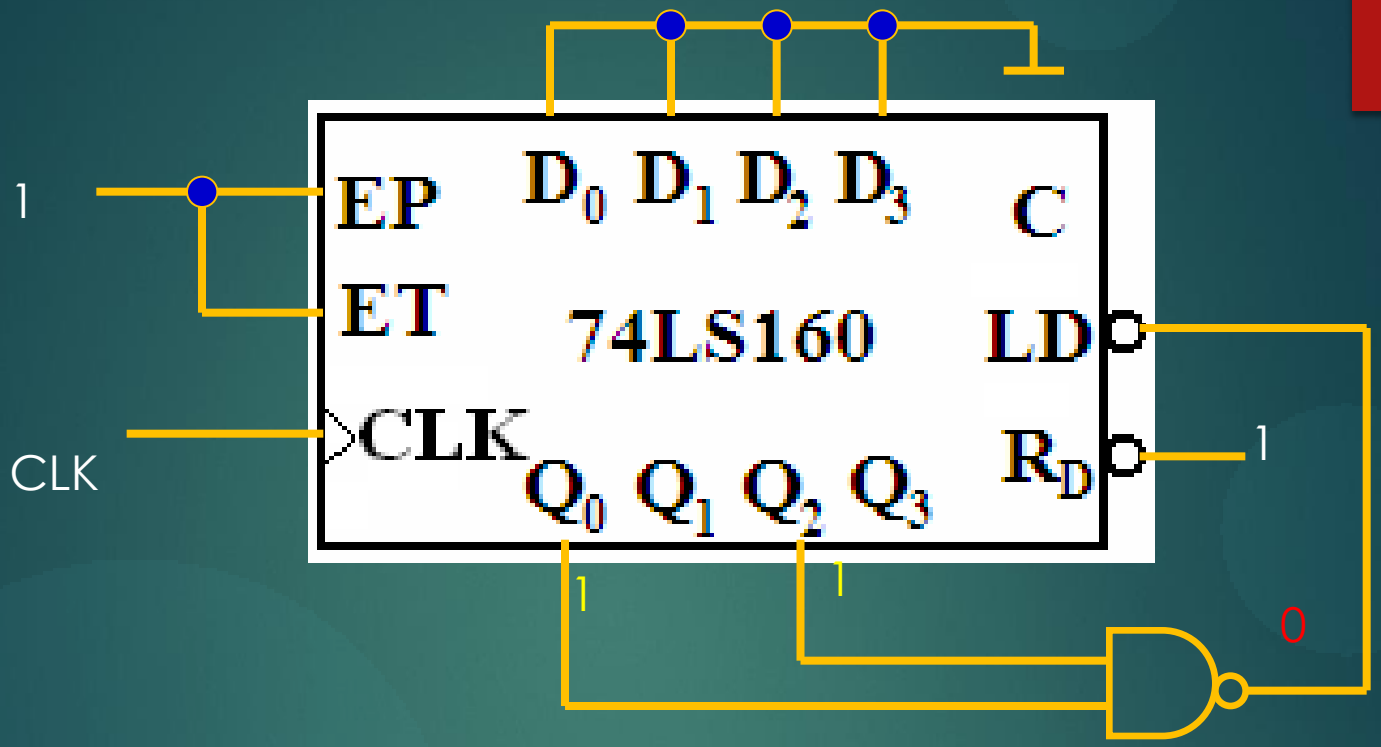

1
EP
ET
CLK
CLK
D0 D1 D2 D3
74LS160
C
LD
RD
Q0 Q1 Q2 Q3
1
1
1
0

置数法

74LS160具有同步置数功能

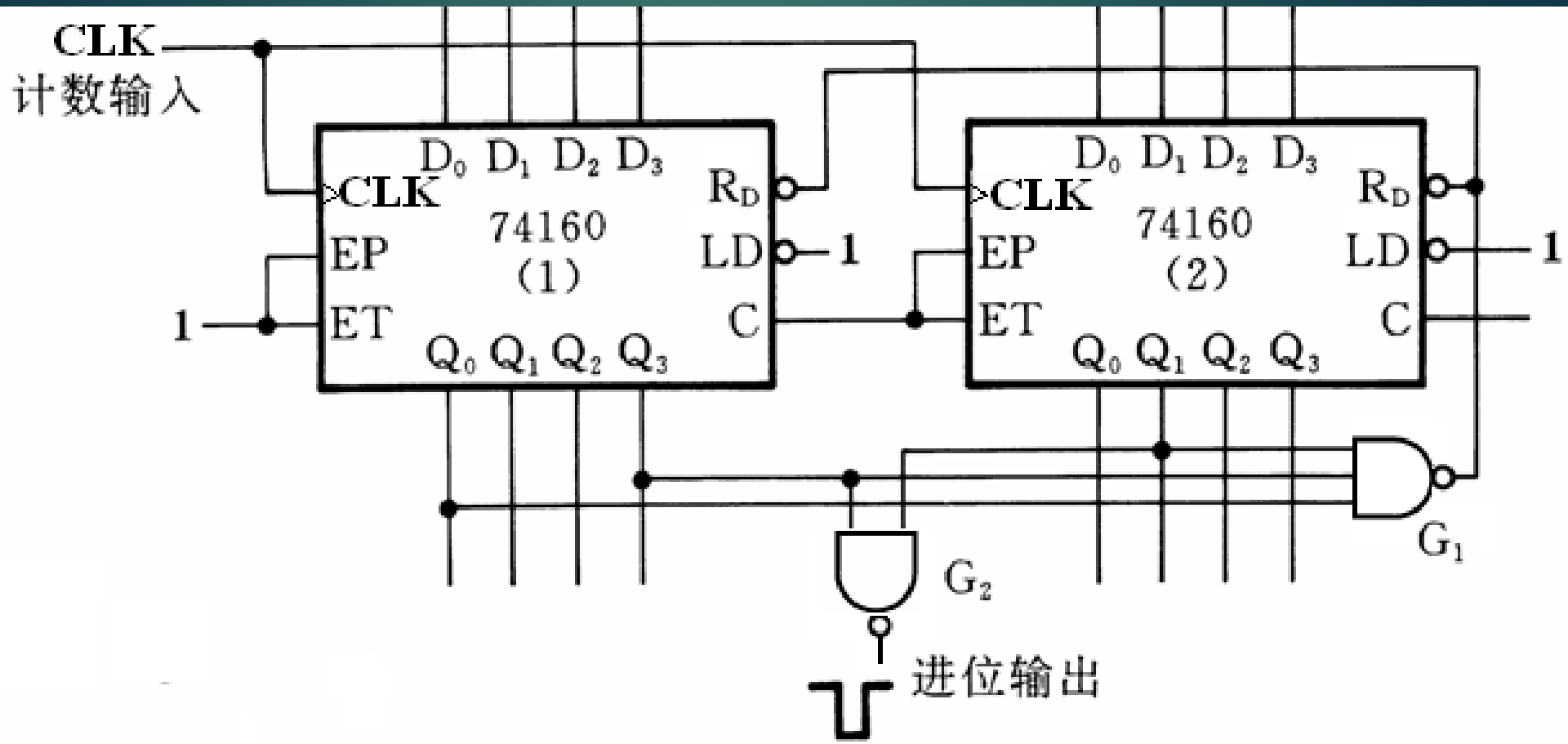


$$LD' = (Q_2 \cdot Q_0)'$$

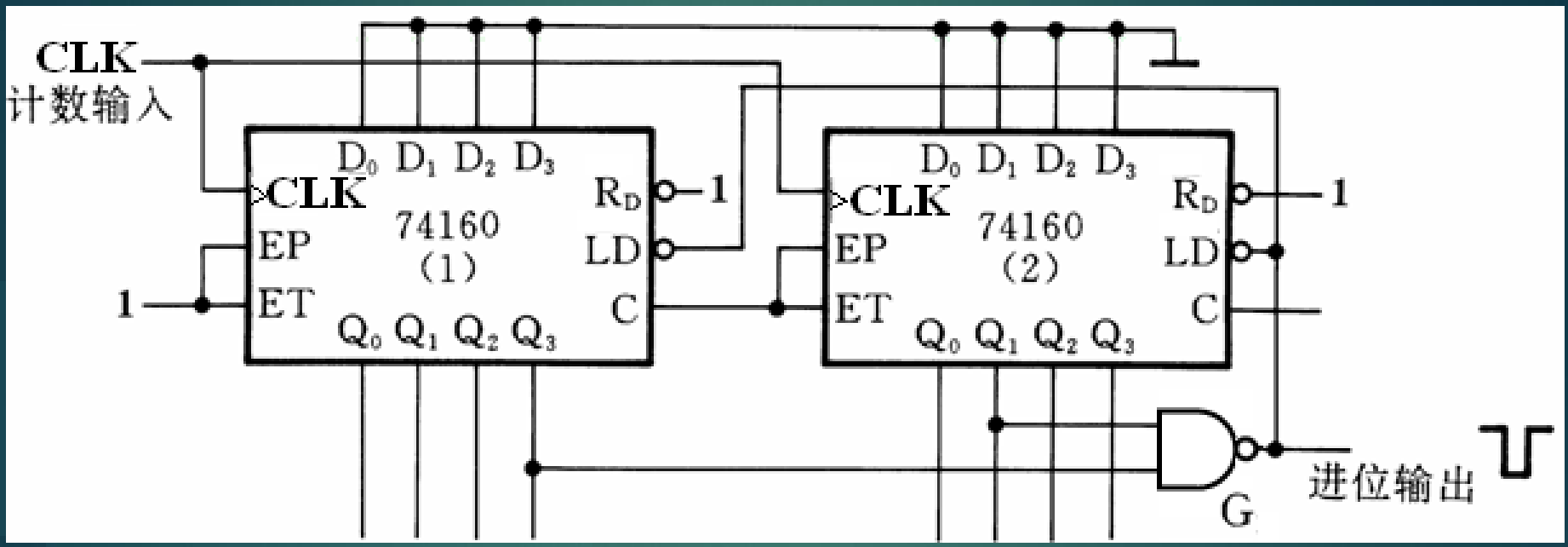

1
EP
ET
CLK
CLK
D0 D1 D2 D3
74LS160
C
LD
RD
Q0 Q1 Q2 Q3
1
1
1
0

用两片74LS160接成二十九进制计数器.

解：　　①整体置零方式

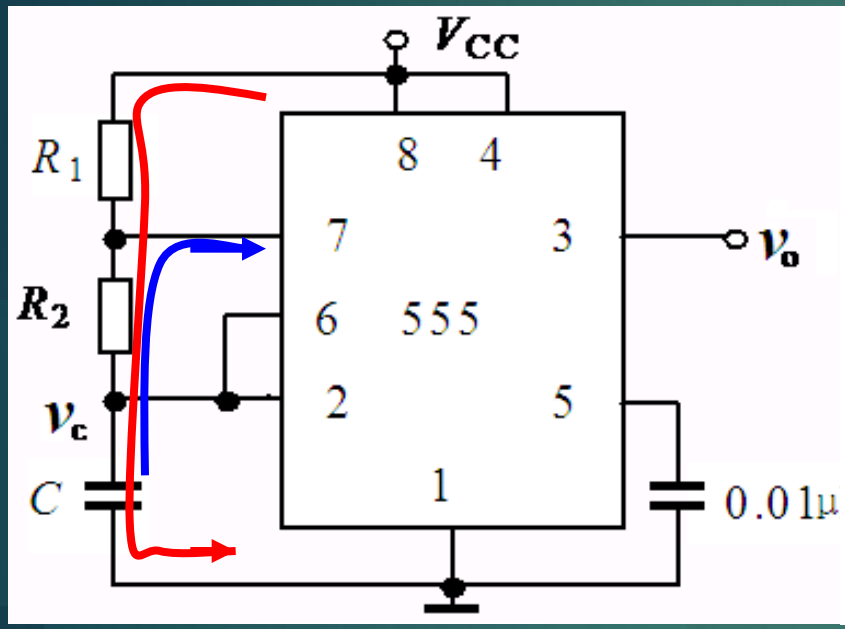


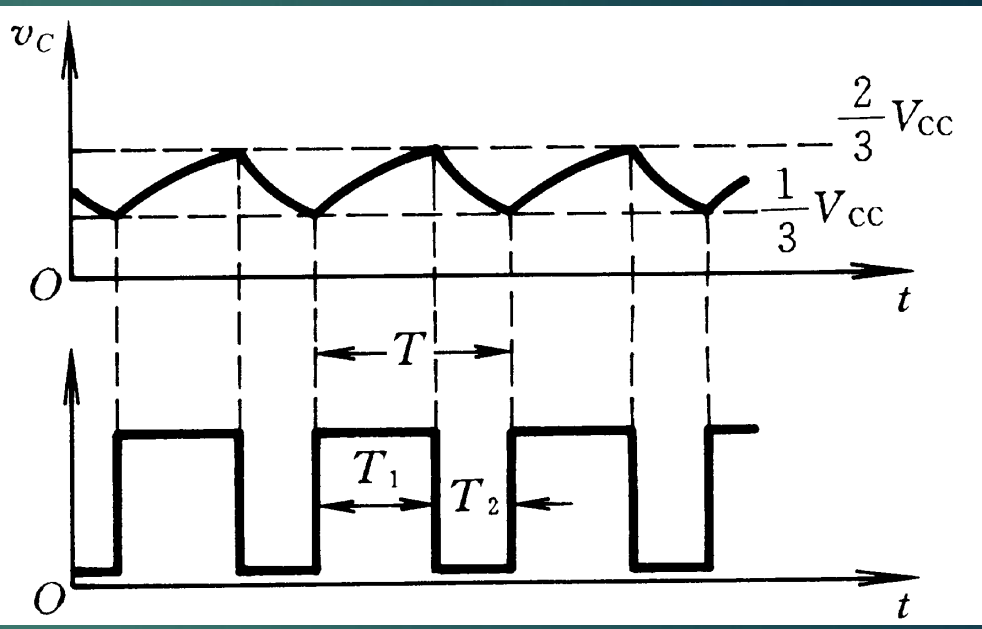
②整体置数方式


CLK
计数输入
D0 D1 D2 D3
CLK
EP
ET
74160
(1)
RD
LD
C
Q0 Q1 Q2 Q3
1
1
D0 D1 D2 D3
CLK
EP
ET
74160
(2)
RD
LD
C
Q0 Q1 Q2 Q3
1
G
进位输出

555定时器

- 555定时器接成多谐振荡器





充 振荡周期：$T=0.69(R_1+2R_2)C$ $+R_2)C\ln 2$

输出脉冲占空比：$q=\dfrac{R_1+R_2}{R_1+2R_2}$

放

存储器与可编程逻辑器件

- 任意逻辑函数的表示
  - 译码器
  - 数据选择器
  - ROM
  - 可编程逻辑器件

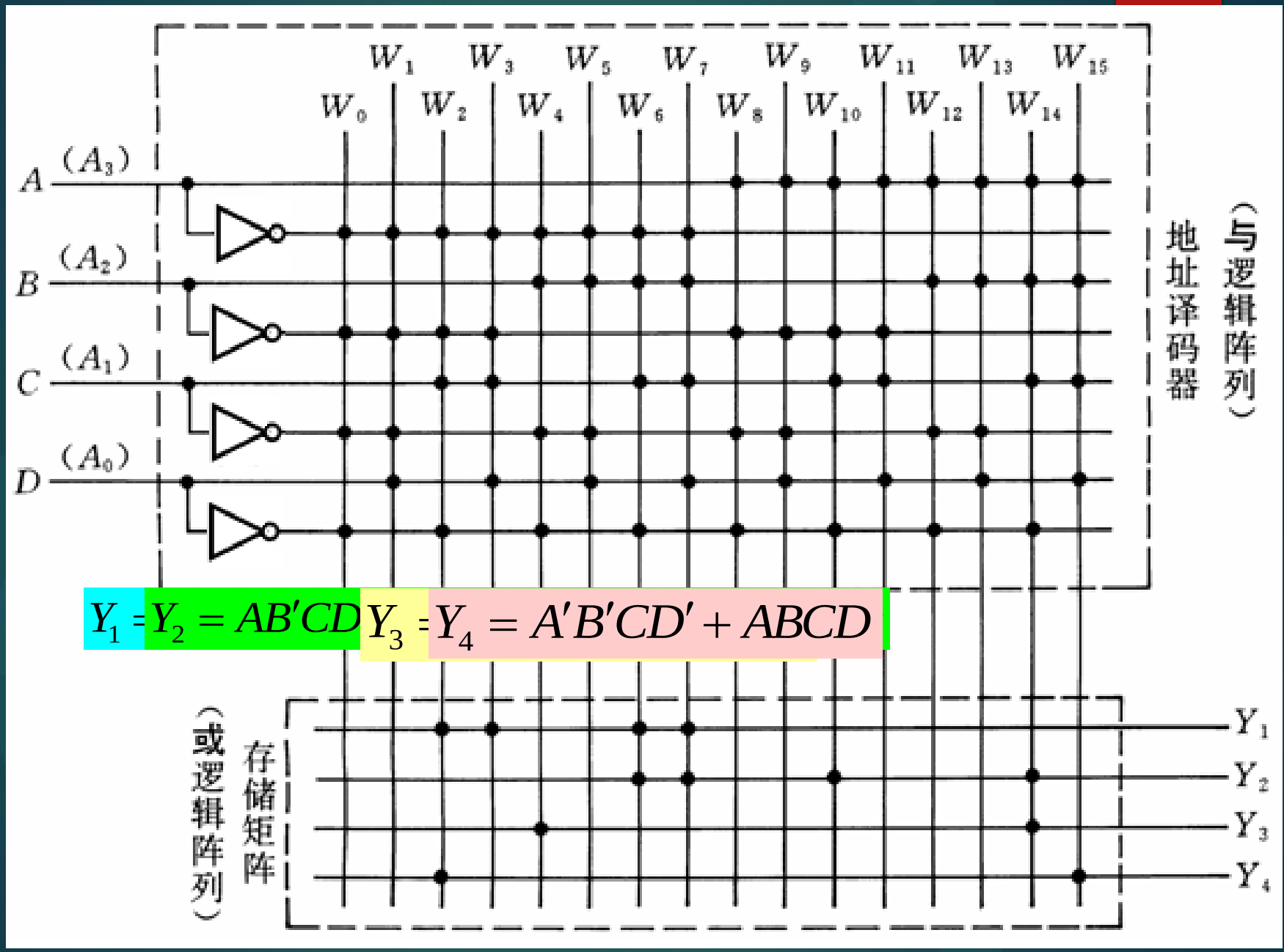
用存储器实现组合逻辑函数

例　试用ROM产生如下一组多输出逻辑函数

$$\begin{cases} Y_1 = A'BC + A'B'C \\ Y_2 = AB'CD' + BCD' + A'BCD \\ Y_3 = ABCD' + A'BC'D' \\ Y_4 = A'B'CD' + ABCD \end{cases}$$

解：化为最小项之和的形式：

$$\begin{cases} Y_1 = A'BCD' + A'BCD + A'B'CD' + A'B'CD \\ Y_2 = AB'CD' + A'BCD' + ABCD' + A'BCD \\ Y_3 = ABCD' + A'BC'D' \\ Y_4 = A'B'CD' + ABCD \end{cases}$$


W_0 W_1 W_2 W_3 W_4 W_5 W_6 W_7 W_8 W_9 W_{10} W_{11} W_{12} W_{13} W_{14} W_{15}
A (A_3)
B (A_2)
C (A_1)
D (A_0)
地址译码器（与逻辑阵列）
$Y_1 = AB'CD$
$Y_2 = AB'CD$
$Y_3 = A'B'CD' + ABCD$
$Y_4 = A'B'CD' + ABCD$
存储矩阵（或逻辑阵列）
Y_1
Y_2
Y_3
Y_4