

逻辑函数标准式

- 用逻辑函数标准式表示任意逻辑函数具有唯一性
- 两种标准式
 - 标准与或式：逻辑函数表示成最小项之和形式，又称最小项表达式
 - 标准或与式：逻辑函数表示成最大项之积形式，又称最大项表达式

最小项定义

- 包含全部输入变量的乘积项，每个变量均以原变量或反变量的形式出现一次

例，对于3变量逻辑函数 $Y(A,B,C)$

$\overline{A}B\overline{C}$, $CA\overline{B}$, $\overline{B}A\overline{C}$ 是最小项

$AC + \overline{B}$, $\overline{A}B$, $\overline{B}A\overline{C}$ 不是最小项

- 对于n变量逻辑函数，共有 2^n 个最小项，每个最小项简记为 m_i ，其中下标i为最小项的编号
 - 编号方法：原变量和反变量分别取1和0构成的二进制数对应的十进制数（注意变量排列次序）
- 上例中， $\overline{A}B\overline{C}$, $CA\overline{B}$, $\overline{B}A\overline{C}$ 分别记作 m_2 , m_5 , m_4

最小项性质

- 对于任意一个最小项

- 有且仅有一组变量取值使其为1

- 对于任意一组变量取值

- 有且仅有一个最小项为1
- 全体最小项的和恒为1
- 任意两个不同最小项乘积均为0

2变量全部最小项真值表

A B	$\bar{A}\bar{B}$ m_0	$\bar{A}B$ m_1	$A\bar{B}$ m_2	AB m_3
0 0	1	0	0	0
0 1	0	1	0	0
1 0	0	0	1	0
1 1	0	0	0	1

示例一逻辑函数标准与或式

- 求逻辑函数的标准与或式

$$\begin{aligned}Y(A,B,C) &= \overline{A}B + A\overline{C} \\&= \overline{A}B(C + \overline{C}) + A(B + \overline{B})\overline{C} \\&= \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}\overline{C} + A\overline{B}C \\&= m_3 + m_2 + m_6 + m_4 \\&= \sum m(2,3,4,6)\end{aligned}$$

$$\begin{aligned}Y(A,B,C) &= \overline{\overline{A}B(B+C)} \\&= AB + AC + \overline{B}C \\&= \sum m(1,5,6,7)\end{aligned}$$

真值表

	A	B	C	Y
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

最大项定义

- 包含全部输入变量的或项，每个变量均以原变量或反变量的形式出现一次

例，对于3变量逻辑函数 $Y(A,B,C)$

$\bar{A} + B + \bar{C}$, $C + A + \bar{B}$, $\bar{B} + A + \bar{C}$ 是最大项

- 对于n变量逻辑函数，共有 2^n 个最大项，每个最大项记作 M_i ，其中下标i为最大项的编号
 - 编号方法：原变量和反变量分别取0和1构成的二进制数对应的十进制数（注意变量排列次序）

上例中， $\bar{A} + B + \bar{C}$, $C + A + \bar{B}$, $\bar{B} + A + \bar{C}$ 分别记作 M_5 , M_2 , M_3

最大项性质

- 对于任意一个最大项
 - 有且仅有一组变量取值使其为0
- 对于任意一组变量取值
 - 有且仅有一个最大项为0
 - 全体最大项的积恒为0
 - 任意两个不同最大项的和均为1

2变量全部最大项真值表

A B	A+B M₀	A+\bar{B} M₁	\bar{A}+B M₂	\bar{A}+\bar{B} M₃
0 0	0	1	1	1
0 1	1	0	1	1
1 0	1	1	0	1
1 1	1	1	1	0

最大项和最小项之间关系

$$M_i = \overline{m_i} \quad \text{或者} \quad m_i = \overline{M_i}$$

示例一逻辑函数标准或与式

- 求逻辑函数的标准或与式

$$\begin{aligned}
 Y(A, B) &= \sum m(1, 2) \\
 &= \overline{A}B + A\overline{B} = \overline{\overline{\overline{\overline{A}B} + \overline{A\overline{B}}}} \\
 &= \overline{(A + \overline{B})(\overline{A} + B)} \\
 &= \overline{\overline{\overline{A}B} + \overline{AB}} = (A + B)(\overline{A} + \overline{B}) \\
 &= M_0 M_3 = \prod M(0, 3) \\
 L(A, B, C) &= \sum m(1, 5, 6, 7) \\
 &= \prod M(0, 2, 3, 4)
 \end{aligned}$$

真值表

	A	B	C	L	
0	0	0	0	0	M_0
1	0	0	1	1	m_1
2	0	1	0	0	M_2
3	0	1	1	0	M_3
4	1	0	0	0	M_4
5	1	0	1	1	m_5
6	1	1	0	1	m_6
7	1	1	1	1	m_7

逻辑函数的化简

- 逻辑函数的最简与或式
 - 乘积项（与项）最少
 - 每个乘积项的因子也最少
- 逻辑函数化简的主要方法
 - 代数法（公式法）
 - 卡诺图法（图解法）

代数法化简

- 反复应用逻辑代数基本定律，消去多余的因子和乘积项
- 常用公式

$$\textcircled{1} \quad \mathbf{AB + A\overline{B} = A}$$

$$\textcircled{2} \quad \mathbf{A + AB = A}$$

$$\textcircled{3} \quad \mathbf{A + \overline{A}B = A + B}$$

$$\textcircled{4} \quad \mathbf{AB + \overline{A}C + BC = AB + \overline{A}C}$$

证明③式： $\mathbf{A + \overline{A}B}$ 用②代入 A

$$\begin{aligned} &= \mathbf{A + AB + \overline{A}B} \\ &= \mathbf{A + (A + \overline{A})B} \\ &= \mathbf{A + B} \end{aligned}$$

证明④式： $\mathbf{AB + \overline{A}C + BC}$

$$\begin{aligned} &= \mathbf{AB + \overline{A}C + (A + \overline{A})BC} \\ &= \mathbf{\underline{AB} + \underline{\overline{A}C} + \underline{ABC} + \underline{\overline{A}BC}} \\ &= \mathbf{AB + \overline{A}C} \end{aligned}$$

②吸收律

示例一代数法化简

$$Y = \underline{\overline{A}B\overline{C}} + \underline{\overline{A}BC} + A\overline{B}C + \underline{AB\overline{C}} + \underline{ABC} \quad (1)$$

$$= \underline{\overline{A}B} + A\overline{B}C + \underline{AB} \quad (1)$$

$$= \underline{B} + A\underline{\overline{B}C} \quad (3)$$

$$= B + AC$$

$$Y = AB + \underline{\overline{A}C} + \underline{\overline{B}C}$$

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

$$= \underline{AB} + \underline{\overline{A}B}C \quad (3)$$

$$= AB + C$$

$$Y = AC + A\overline{B} + \overline{B + C}$$

$$= \underline{AC} + \underline{A\overline{B}} + \underline{\overline{B}C} \quad (4)$$

$$= AC + \overline{B}C$$

卡诺图

- 一般用于**5**变量以内逻辑函数的图解法化简
- **n**变量卡诺图：将每个最小项用一个小方块表示，全部 2^n 个最小项按逻辑相邻（只有一个变量不同）在几何位置上也相邻的方式排列起来，所构成的阵列图
 - 每个最小项都有n个与其逻辑相邻的最小项

两变量卡诺图

左右相邻
上下相邻
对角不相邻

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

简画为

		B	
		0	1
A	0	m_0	m_1
	1	m_2	m_3

三变量和四变量卡诺图

三变量卡诺图

		BC			
		00	01	11	10
A	0	m_0	m_1	m_3	m_2
	1	m_4	m_5	m_7	m_6

循环码

四变量卡诺图

		CD			
		00	01	11	10
AB	00	m_0	m_1	m_3	m_2
	01	m_4	m_5	m_7	m_6
	11	m_{12}	m_{13}	m_{15}	m_{14}
	10	m_8	m_9	m_{11}	m_{10}

左右相邻，上下相邻

循环相邻：左边右边相邻，上边下边相邻

对角不相邻

逻辑函数用卡诺图表示

- 方法一：首先将逻辑函数式转化为最小项表达式，然后在卡诺图上与这些最小项对应的方格添1，其余方格添0（有时也可用空格表示）
- 方法二：根据逻辑函数式各乘积项含义直接填写卡诺图

$$\begin{aligned} Y(A,B,C) &= \overline{A}B + A\overline{C} \\ &= \overline{A}BC + \overline{A}B\overline{C} + A\overline{B}C + A\overline{B}\overline{C} \\ &= \sum m(2,3,4,6) \end{aligned}$$

BC		00	01	11	10
A	0			1	1
	1	1			1

利用卡诺图化简逻辑函数

- 将 2^i 个相邻的最小项合并，得到一个乘积项
 - 该项保留各项中相同的变量，消去了 i 个不同的变量

AB \ CD				
	00	01	11	10
00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$	$\bar{A}\bar{B}C\bar{D}$
01	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
11	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$	$ABC\bar{D}$
10	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$

$$\bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D} = \bar{A}\bar{B}C$$

$$\bar{A}BCD + \bar{A}BC\bar{D} = \bar{A}BC$$

$$\bar{A}\bar{B}C + \bar{A}BC = \bar{A}C$$

$$ABC + A\bar{B}C = AC$$

$$\bar{A}C + AC = C$$

卡诺图法化简步骤

- 根据逻辑函数式或真值表填写卡诺图
 - 将逻辑函数中存在的或真值表中为1的最小项对应的方格填1，其它填0（或空白）
- 用尽可能少的包围圈将所有填1方格圈起来
- 每个包围圈写出一个乘积项
 - 保留各项中相同的变量，消去不同的变量
- 将全部乘积项相加，即得逻辑函数的最简与或式

画包围圈原则

- 包围圈中的方格都是填1的方格
- 包围圈中的方格数要求尽可能多，一定是 2^i 个，而且所有方格可以构成矩形
- 同一方格可以被不同的包围圈重复包围，但新增的包围圈中一定要有新的未被包围的方格
- 有时1方格较多时，也可先圈0，求反函数，然后再求原函数

示例—卡诺图化简(1)

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

BC					
		00	01	11	10
A	0	0	1	1	1
	1	1	1	0	1

$$Y = A\overline{B} + \overline{A}C + B\overline{C}$$

BC					
		00	01	11	10
A	0	0	1	1	1
	1	1	1	0	1

$$Y = A\overline{C} + \overline{B}C + \overline{A}B$$

逻辑函数的化简结果可能不唯一

示例—卡诺图化简(2)

$$Y(A,B,C,D) = \Sigma (m_0, m_2, m_4, m_6, m_8 \sim m_{15})$$

AB \ CD		CD			
		00	01	11	10
00	AB	1	0	0	1
		1	0	0	1
11	AB	1	1	1	1
		1	1	1	1

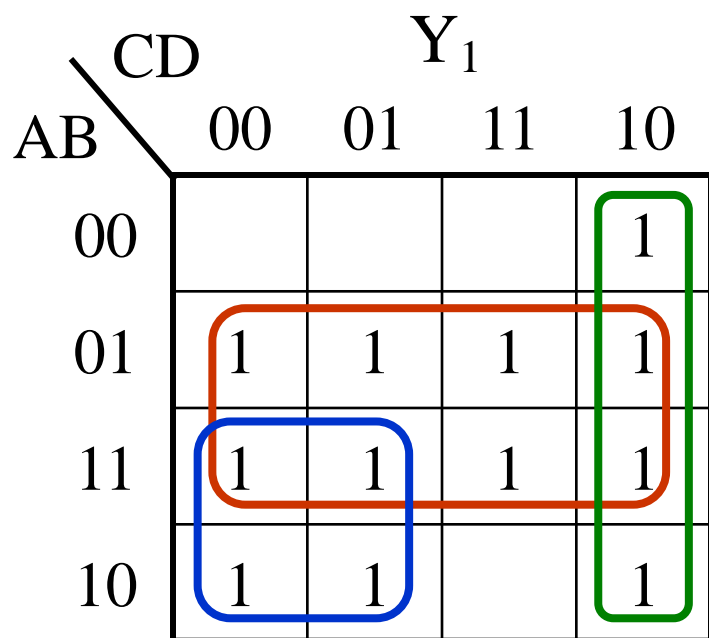
AB \ CD		CD			
		00	01	11	10
00	AB	1	0	0	1
		1	0	0	1
11	AB	1	1	1	1
		1	1	1	1

$$Y = A + \overline{D}$$

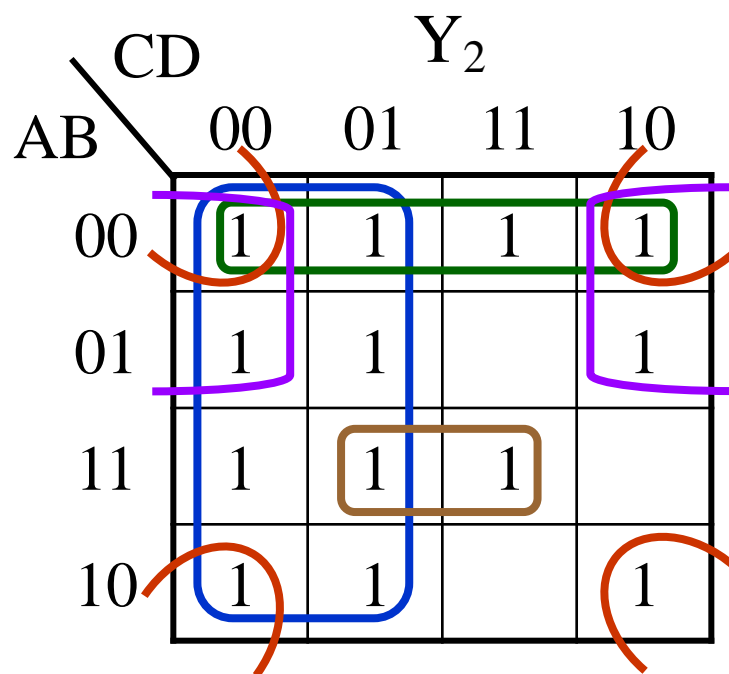
$$\overline{Y} = \overline{A}D$$

$$Y = \overline{\overline{A}D} = A + \overline{D}$$

示例—卡诺图化简(3)



$$Y_1 = B + A\bar{C} + C\bar{D}$$



$$Y_2 = ABD + \bar{B}\bar{D} + \bar{A}\bar{D} + \bar{A}\bar{B} + \bar{C}$$

含无关项的逻辑函数化简

- 无关项（或者任意项）
 - 不允许或不可能出现的最小项
 - 对应的函数值可以是任意的最小项
- 填卡诺图时在无关项对应的格内填符号“x”、或“ Φ ”
- 化简时可根据需要视无关项为1或0
- 含有无关项的函数的表示形式

$$Y = \sum m(\dots) + \sum d(\dots) \text{ 或者}$$

$$Y = \sum m(\dots), \quad \sum m(\dots) = 0$$

示例—含无关项的函数化简

$$Y(A,B,C,D) = \Sigma m(0, 2\sim 4, 6, 8, 10) \\ + \Sigma d(11, 12, 14, 15)$$

不考虑无关项时:

$$Y = \underline{\overline{A} \overline{D}} + \underline{\overline{B} \overline{D}} + \underline{\overline{A} \overline{B} C}$$

CD \ AB	00	01	11	10
00	1		1	1
01	1			1
11	x		x	x
10	1		x	1

示例—含无关项的函数化简

$$Y(A,B,C,D) = \Sigma m(0, 2\sim 4, 6, 8, 10) \\ + \Sigma d(11, 12, 14, 15)$$

不考虑无关项时:

$$Y = \overline{A} \overline{D} + \overline{B} \overline{D} + \overline{A} \overline{B} C$$

考虑无关项时:

$$Y = \underline{\overline{D}} + \underline{\overline{B} C}$$

CD \ AB	00	01	11	10
00	1		1	1
01	1			1
11	x		x	x
10	1		x	1

逻辑函数不同最简形式变换

- 与或 \rightarrow 与非-与非 $Y = \overline{A}B + A\overline{C}$ 与或式
– 两次取反，摩根展开一次 $= \overline{\overline{\overline{A}B} \overline{\overline{A}C}}$ 与非-与非
- 与或 \rightarrow 与或非 $= \overline{\overline{A} \overline{B} + A\overline{C}}$ 与或非式
– 先展开，再忽略 $\overline{B}C$ 项
- 与或非 \rightarrow 或与 $= (A + B)(\overline{A} + \overline{C})$ 或与式
– 摩根展开一次
- 或与 \rightarrow 或非-或非 $= \overline{\overline{A + B} \overline{A + C}}$ 或非-或非式
– 两次取反，摩根展开一次

示例—给定电路分析功能 (续)

$$X = A$$

$$Y = A \oplus B$$

$$Z = A \oplus C$$

- 列写真值表
- 确定电路逻辑功能

电路实现功能:

将三位二进制原码转换为三位二进制反码

真值表

A	B	C	X	Y	Z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	0	0

组合逻辑电路设计

- 根据实际逻辑问题，求出实现所要求逻辑功能的最简逻辑电路

逻辑功能 \Rightarrow 逻辑图

- 设计步骤
 - 分析实际逻辑问题的因果关系，确定输入/输出变量，定义逻辑状态含义，列出真值表
 - 由真值表写出逻辑函数式
 - 根据选用器件类型，化简和变换逻辑函数式
 - 画出逻辑电路图

示例—给定逻辑问题设计电路

- 设计三人多数表决电路

假设输入变量A、B、C

1--赞成, 0--否决

输出变量Y

1--通过, 0--未通过

Y	BC				
		00	01	11	10
A	0	0	0	1	0
	1	0	1	1	1

真值表

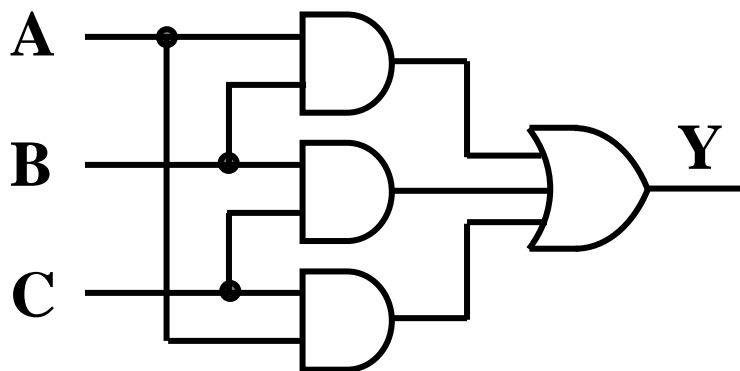
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$Y = AB + AC + BC$$

$$Y(A, B, C) = \sum m(3, 5, 6, 7)$$

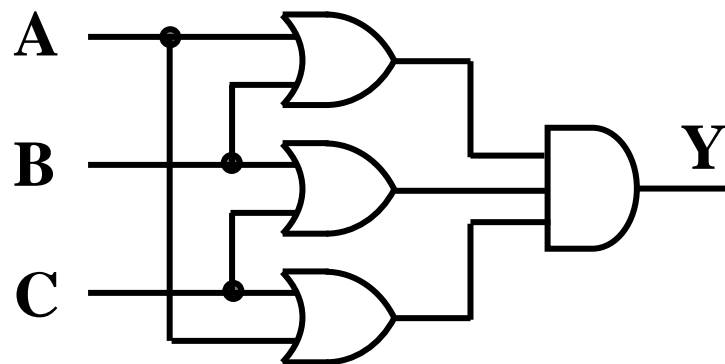
$$Y = (A + B)(A + C)(B + C)$$

示例一给定逻辑问题设计电路(续)



与或式实现

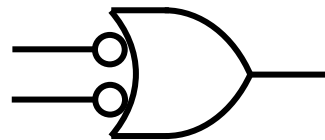
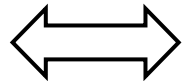
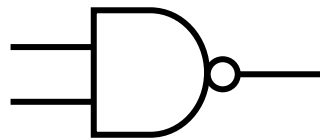
$$Y = AB + AC + BC$$



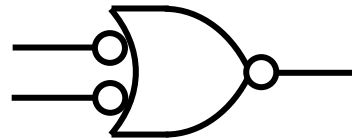
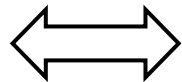
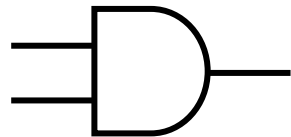
或与式实现

$$Y = (A + B)(A + C)(B + C)$$

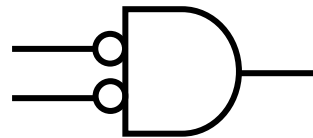
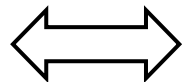
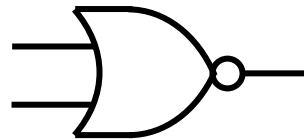
逻辑门等效符号



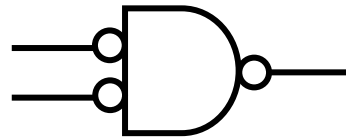
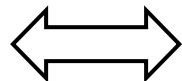
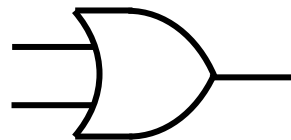
$$\overline{A \cdot B} = \overline{A} + \overline{B}$$



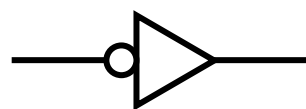
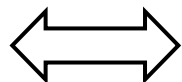
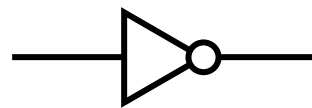
$$A \cdot B = \overline{\overline{A} + \overline{B}}$$



$$\overline{A+B} = \overline{A} \cdot \overline{B}$$



$$A+B = \overline{\overline{A} \cdot \overline{B}}$$



$$\overline{A} = \overline{A}$$

重点回顾

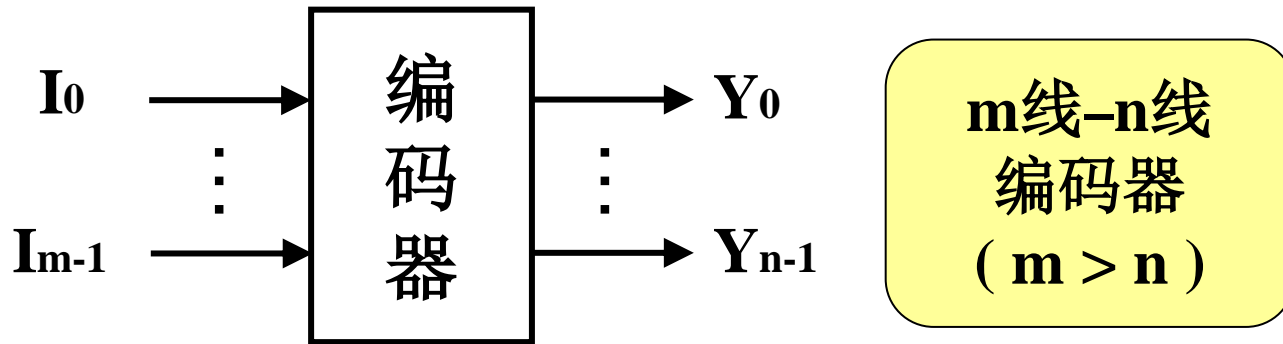
- 什么是组合逻辑
 - 无记忆、无反馈
- 组合逻辑电路分析过程
 - 逻辑图->逻辑函数式（化简）->真值表->逻辑功能
- 组合逻辑电路设计过程
 - 分析逻辑问题->真值表/卡诺图->逻辑函数式（化简）->逻辑图
 - 逻辑门等效：线两端非非、变量+门一起非

内容提纲

- 常用组合逻辑电路
 - 编码器
 - 译码器

编码器

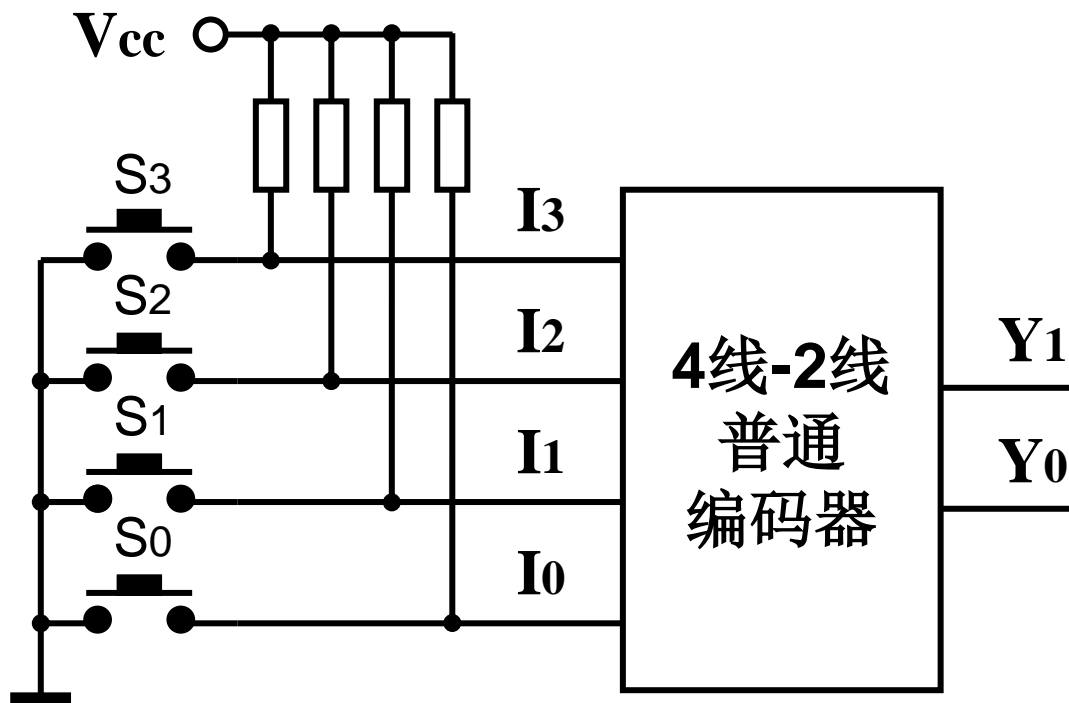
- 编码器：将输入信号的状态，转换成二进制代码输出



- 编码器分类：普通编码器和优先编码器
- 常见编码器：8线-3线编码器74x148，10线-4线(BCD)编码器74x147等

普通编码器

- 任何时刻只允许有一个输入信号处于有效状态
- 设计4线-2线普通编码器（设输入采取低电平为有效状态）



编 码 表

输入有效	$Y_1 Y_0$	
I_0	0	0
I_1	0	1
I_2	1	0
I_3	1	1

4线-2线普通编码器

真值表

I ₃	I ₂	I ₁	I ₀	Y ₁	Y ₀
1	1	1	0	0	0
1	1	0	1	0	1
1	0	1	1	1	0
0	1	1	1	1	1
其 他				x	x

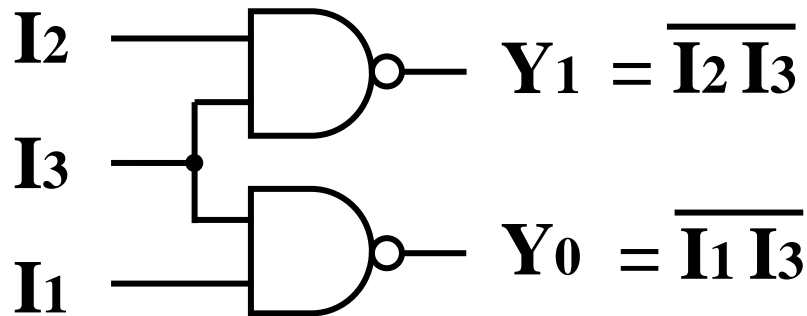
$$Y_1 = \overline{I_2} + \overline{I_3} = \overline{I_2 I_3}$$

$$Y_0 = \overline{I_1} + \overline{I_3} = \overline{I_1 I_3}$$

$Y_1 \quad I_1 I_0$ $I_3 I_2$		$I_1 I_0$			
		00	01	11	10
00		x	x	x	x
01		x	x	1	x
11		x	0	x	0
10		x	x	1	x

Y_0	$I_1 I_0$				
	$I_3 I_2$	00	01	11	10
00		x	x	x	x
01		x	x	1	x
11		x	1	x	0
10		x	x	0	x

4线-2线普通编码器(续1)



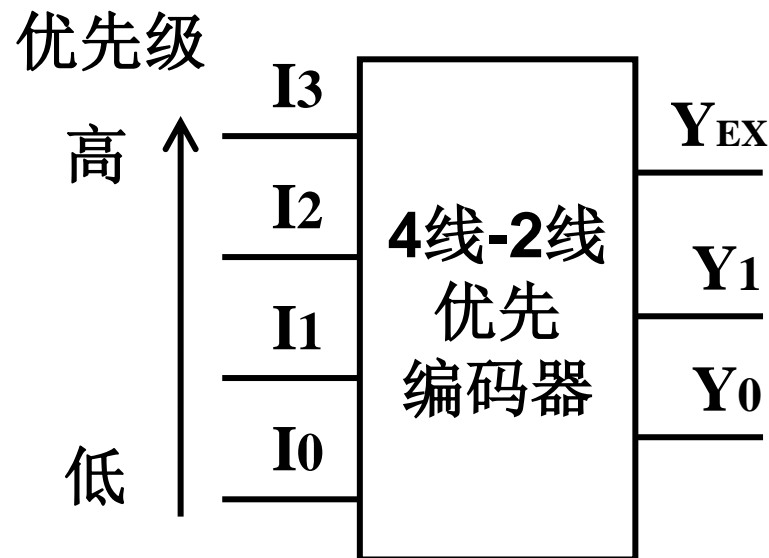
• 编码器的限制

- 不能区分输入 I_0 有效与无输入有效
- 不能处理多输入同时有效

I_3	I_2	I_1	I_0	Y_1	Y_0
0	0	0	0	1	1
0	0	0	1	1	1
0	0	1	0	1	1
0	0	1	1	1	1
0	1	0	0	1	1
0	1	0	1	1	1
0	1	1	0	1	1
0	1	1	1	1	1
1	0	0	0	1	1
1	0	0	1	1	1
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	0
1	1	1	1	0	0

优先编码器

- 允许两个以上输入信号同时有效，并对其中优先级最高的一个进行编码
- 设计4线-2线优先编码器 (输入和输出信号均为低电平有效)



真值表

I_3	I_2	I_1	I_0	Y_{EX}	Y_1	Y_0
1	1	1	1	1	x	x
0	x	x	x	0	0	0
1	0	x	x	0	0	1
1	1	0	x	0	1	0
1	1	1	0	0	1	1

4线-2线优先编码器

$I_3 I_2 \backslash I_1 I_0$		00	01	11	10
		00	01	11	10
Y_{EX}	00	0	0	0	0
	01	0	0	0	0
	11	0	0	1	0
	10	0	0	0	0

$I_3 I_2 \backslash I_1 I_0$		00	01	11	10
		00	01	11	10
Y_1	00	0	0	0	0
	01	0	0	0	0
	11	1	1	x	1
	10	0	0	0	0

$$Y_{EX} = I_0 I_1 I_2 I_3$$

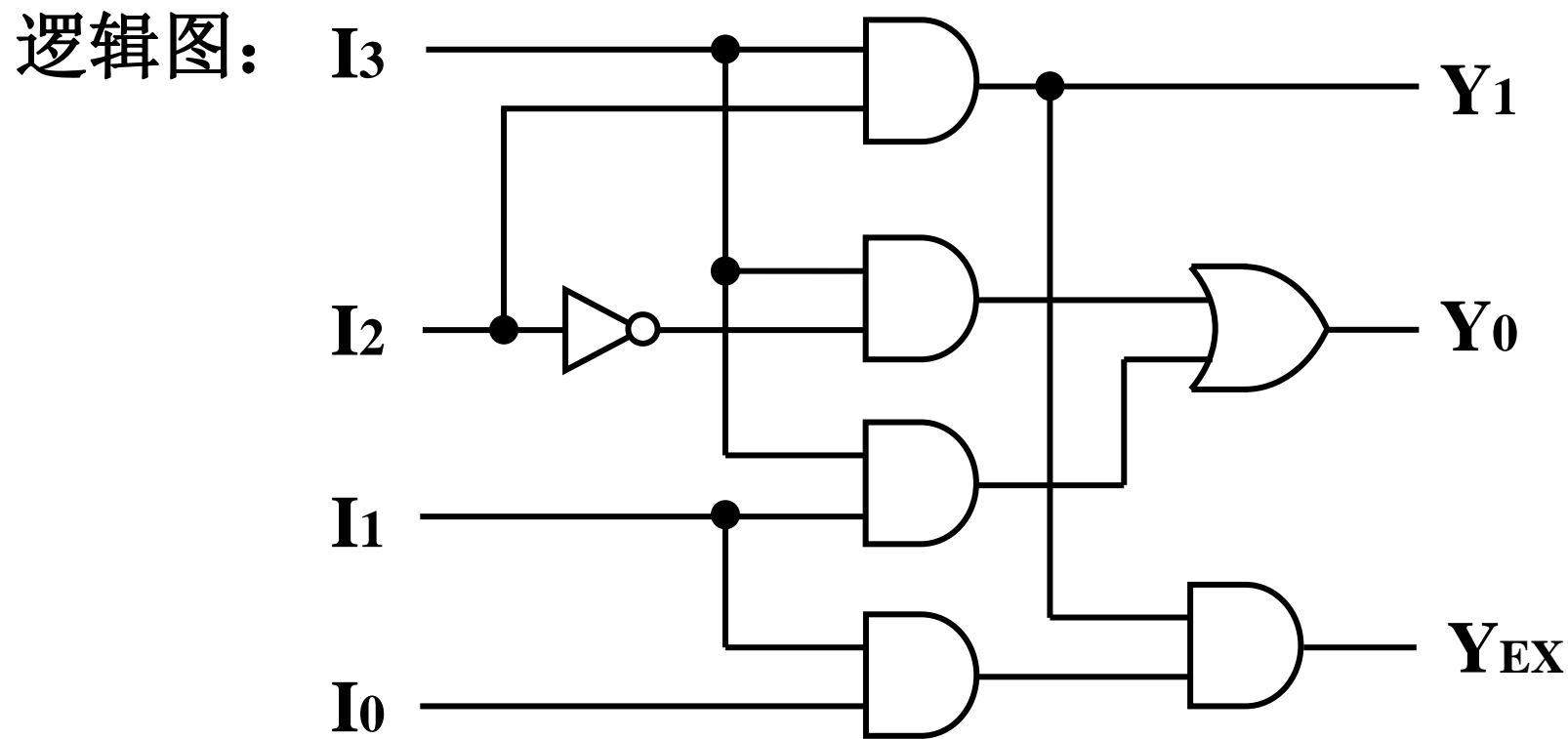
$$Y_1 = I_3 I_2$$

$$Y_0 = I_3 \bar{I}_2 + I_3 I_1$$

$I_3 I_2 \backslash I_1 I_0$		00	01	11	10
		00	01	11	10
Y_0	00	0	0	0	0
	01	0	0	0	0
	11	0	0	x	1
	10	1	1	1	1

4线-2线优先编码器(续)

逻辑式: $Y_{EX} = I_0 I_1 I_2 I_3$ $Y_1 = I_3 I_2$ $Y_0 = I_3 \bar{I}_2 + I_3 I_1$



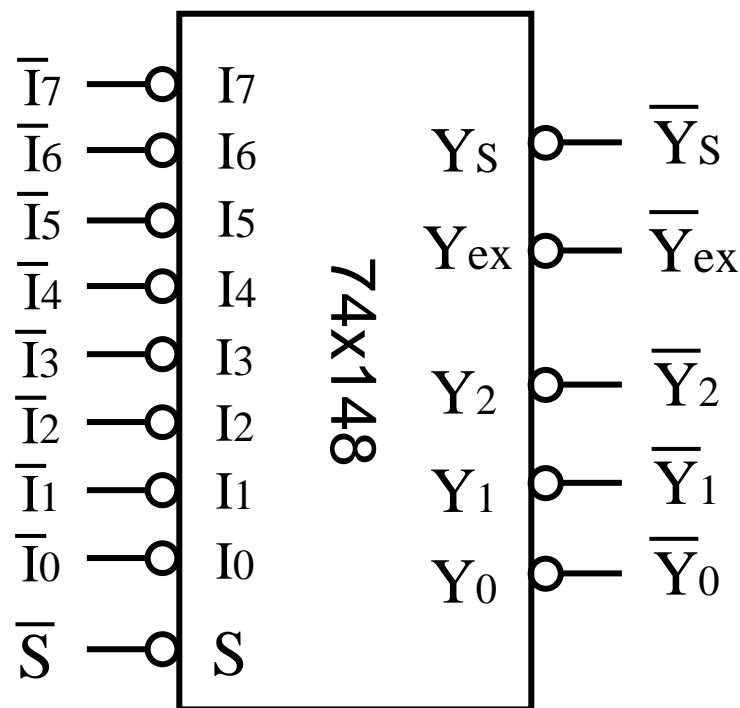
8线-3线优先编码器74x148

- 对**8**个输入信号按优先级编码，输出**3**位代码
- 通过多片级联，对多于**8**个输入信号进行编码

- $\overline{I_7} \sim \overline{I_0}$: 待编码输入信号，低电平有效，优先级递减，即 $\overline{I_7}$ 优先级最高， $\overline{I_0}$ 最低

- $\overline{Y_2} \sim \overline{Y_0}$: 二进制负有效输出， $\overline{Y_2}$ 为最高位

- $\overline{Y_{ex}}$: 扩展输出
 - \overline{S} : 使能输入
 - $\overline{Y_S}$: 使能输出
- } 低电平有效



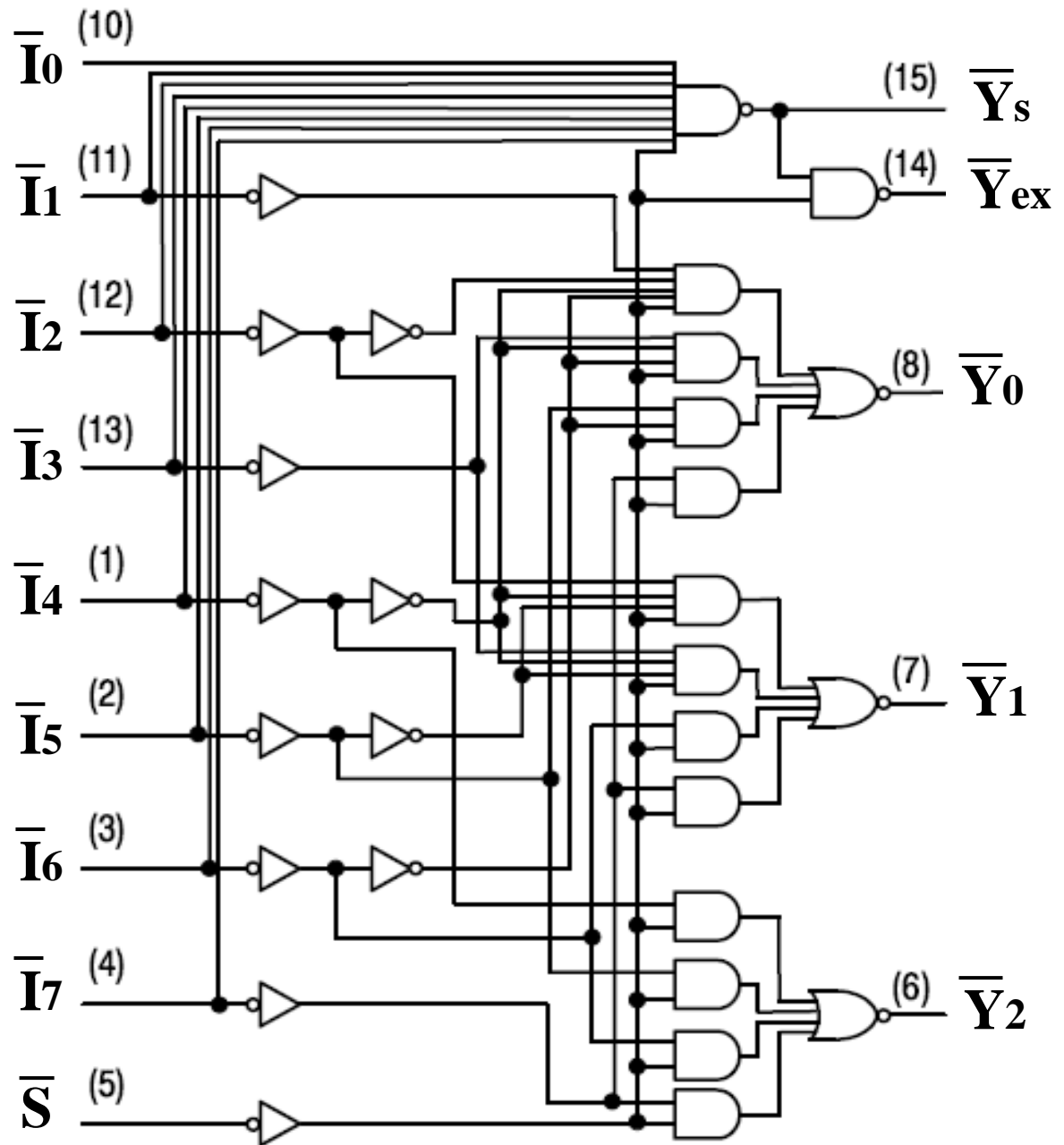
逻辑符号

74x148真值表

\bar{S}	\bar{I}_0	\bar{I}_1	\bar{I}_2	\bar{I}_3	\bar{I}_4	\bar{I}_5	\bar{I}_6	\bar{I}_7	\bar{Y}_2	\bar{Y}_1	\bar{Y}_0	\bar{Y}_{ex}	\bar{Y}_s
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	x	x	x	x	x	x	x	0	0	0	0	0	1
0	x	x	x	x	x	x	0	1	0	0	1	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	0	1	1	0	1
0	x	x	x	0	1	1	1	1	1	0	0	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	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

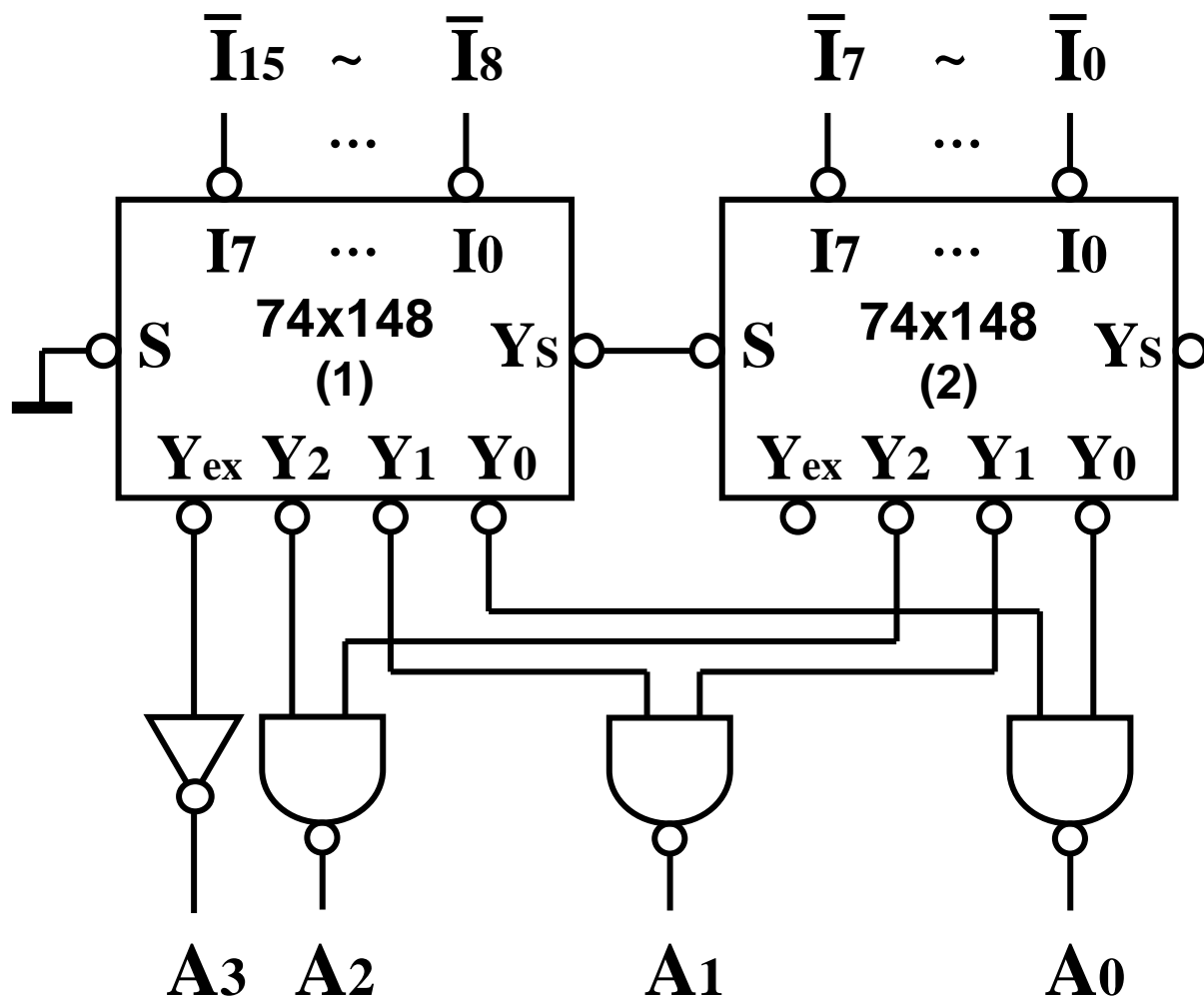
74x148逻辑图

- 若 \overline{S} 无效，则所有输出均无效
- 若 \overline{S} 有效且 $\overline{I}_0 \sim \overline{I}_7$ 均无效，则仅 \overline{Y}_s 有效
- 若 \overline{S} 有效且 $\overline{I}_0 \sim \overline{I}_7$ 中存在有效，则 \overline{Y}_s 无效， \overline{Y}_{ex} 和 $\overline{Y}_2 \sim \overline{Y}_0$ 有效



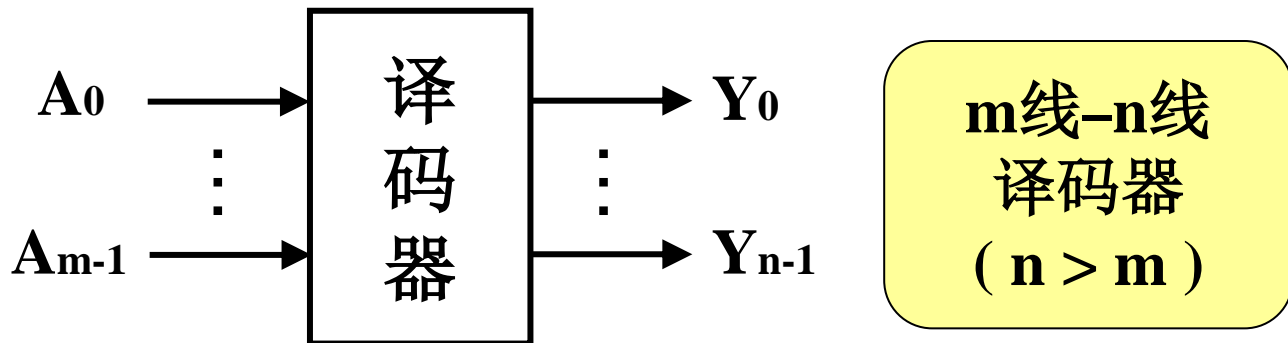
74x148扩展应用

- 16线-4线优先编码器
- 16个输入信号 $\overline{I}_{15} \sim \overline{I}_0$ ，低电平有效， \overline{I}_{15} 优先级最高
- 输出4位二进制正码 $A_3 \sim A_0$ ， A_3 为最高位



译码器

- 译码是编码的逆过程，将输入的二进制代码转换成高、低电平组合状态输出



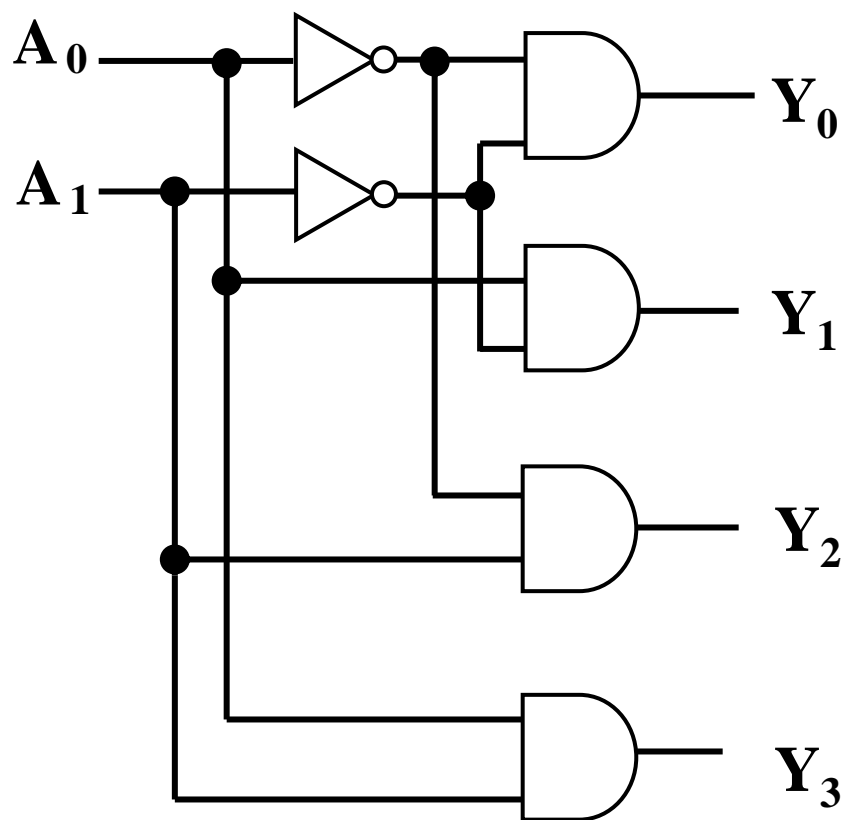
- 常见译码器
 - 二进制译码器74x138、74x139
 - 二-十进制译码器74x42
 - 显示译码器74x47等

2线-4线译码器

- 设输出高电平有效

A_1	A_0	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$$\left. \begin{aligned} Y_0 &= \overline{A_1} \overline{A_0} = m_0 \\ Y_1 &= \overline{A_1} A_0 = m_1 \\ Y_2 &= A_1 \overline{A_0} = m_2 \\ Y_3 &= A_1 A_0 = m_3 \end{aligned} \right\} \text{最小项译码}$$



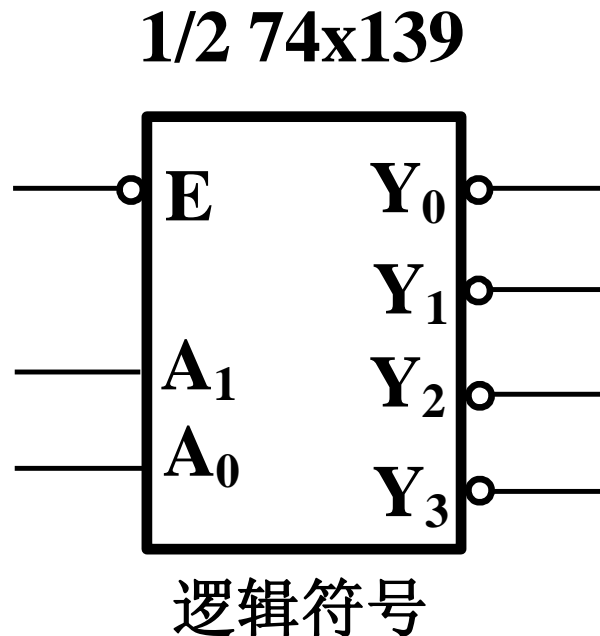
若输出低电平有效?

双2线-4线译码器74x139

- 简称2-4译码器

- \overline{E} : 使能输入, 低电平有效
- $\overline{Y}_0 \sim \overline{Y}_3$: 译码输出, 低电平有效

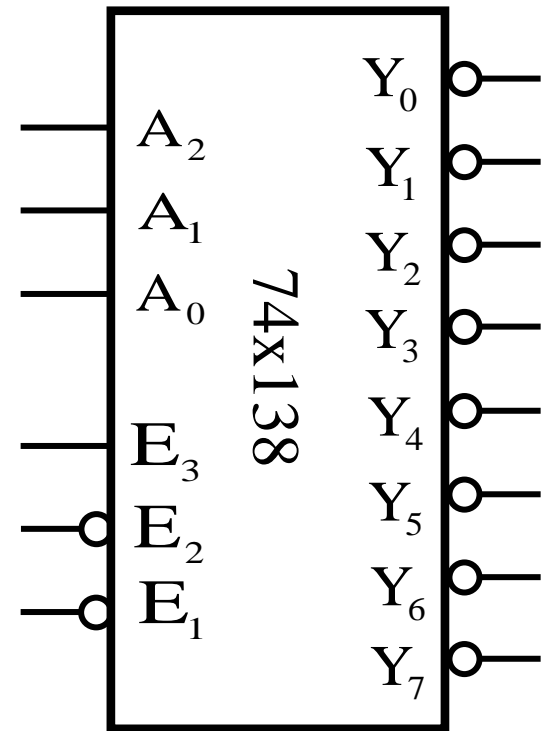
真值表



\overline{E}	A_1	A_0	\overline{Y}_0	\overline{Y}_1	\overline{Y}_2	\overline{Y}_3
1	x	x	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

3线-8线译码器74x138

- 简称3-8译码器
- $A_2 \sim A_0$: 待译码输入信号
 - A_2 为最高位, A_0 为最低位
- $\overline{Y}_0 \sim \overline{Y}_7$: 译码输出信号
 - 低电平有效
 - 最多只有一个有效
- $E_3, \overline{E}_2, \overline{E}_1$: 使能输入信号
 - 同时有效时才译码
 - E_3 高电平有效
 - \overline{E}_1 、 \overline{E}_2 低电平有效



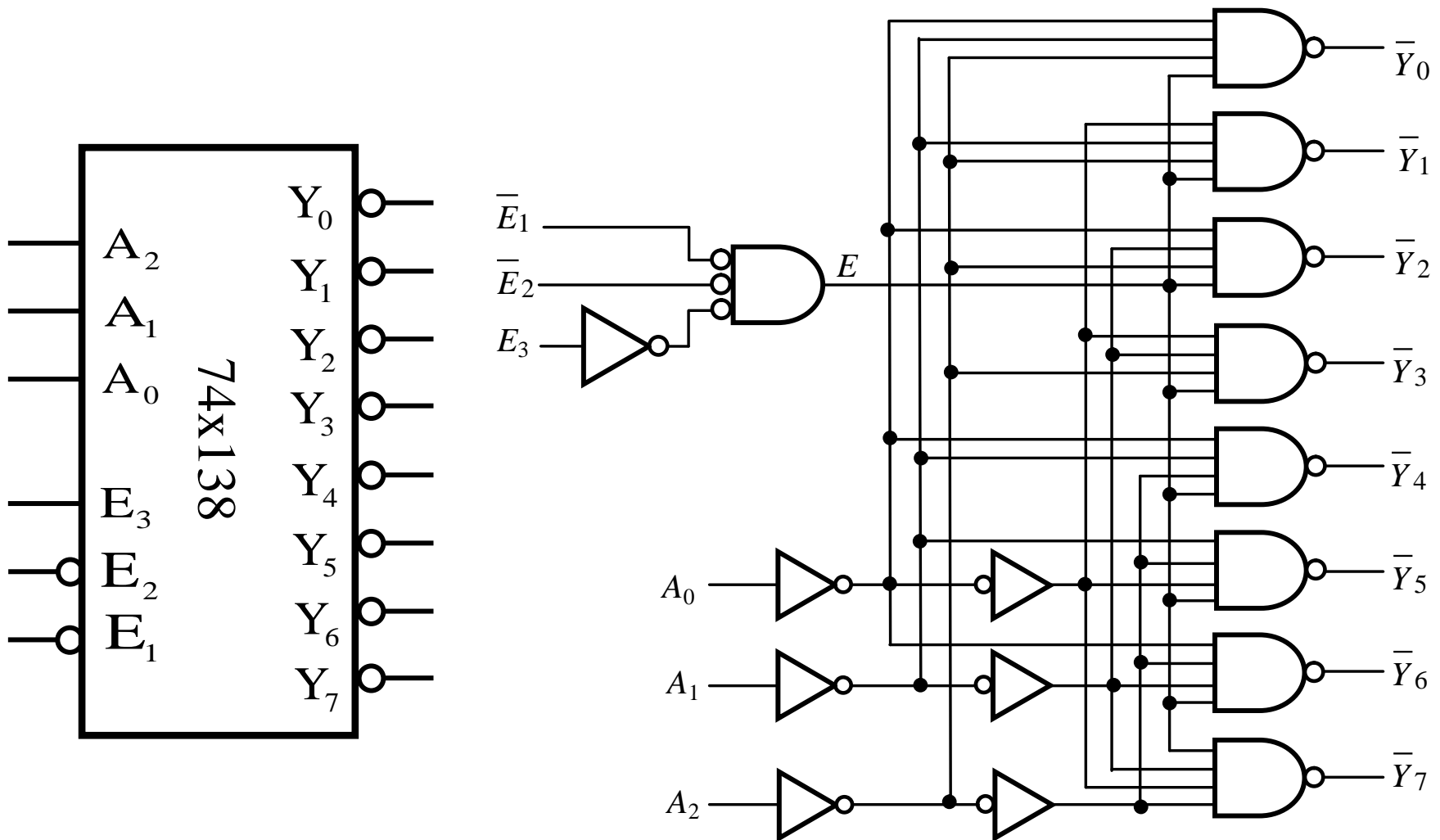
逻辑符号

74x138真值表

E_3	\overline{E}_2	\overline{E}_1	A_2	A_1	A_0	\overline{Y}_0	\overline{Y}_1	\overline{Y}_2	\overline{Y}_3	\overline{Y}_4	\overline{Y}_5	\overline{Y}_6	\overline{Y}_7
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

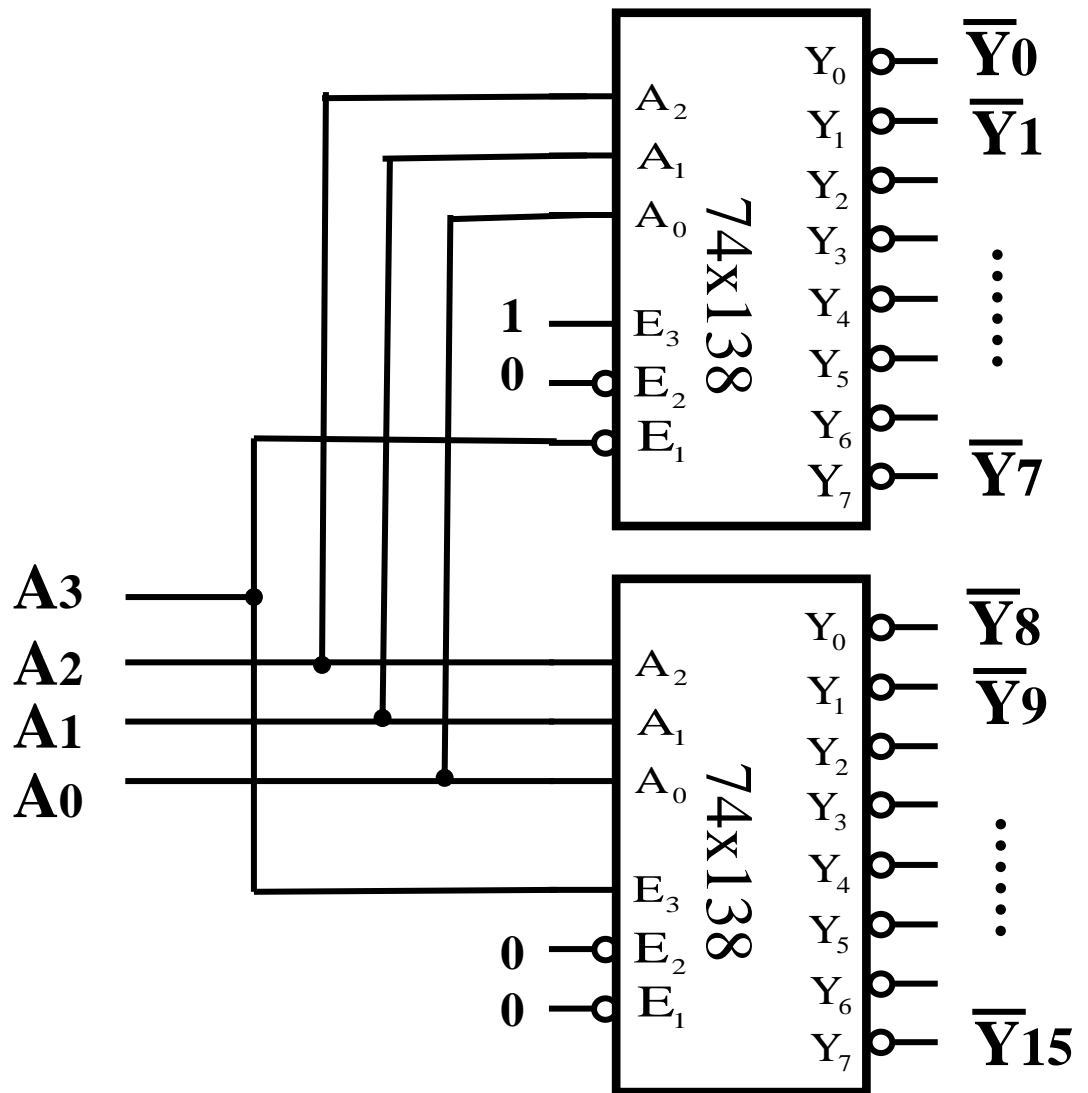
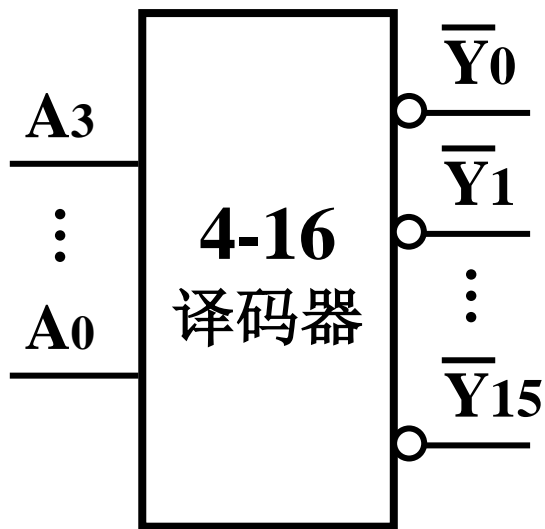
$$\overline{Y}_i = E_3 \overline{E}_2 \overline{E}_1 m_i \quad \text{当 } E_3 \overline{E}_2 \overline{E}_1 = 100 \text{ 时,} \quad \overline{Y}_i = \overline{m}_i$$

74x138逻辑图



译码器应用

- 利用74x138设计
4线-16线译码器



译码器实现组合逻辑函数

- 用**74x138**可以实现任意三变量的逻辑函数

$$E_3 \bar{E}_2 \bar{E}_1 = 100$$

$$A_2 A_1 A_0 = ABC$$

$$\bar{Y}_i = \bar{m}_i$$

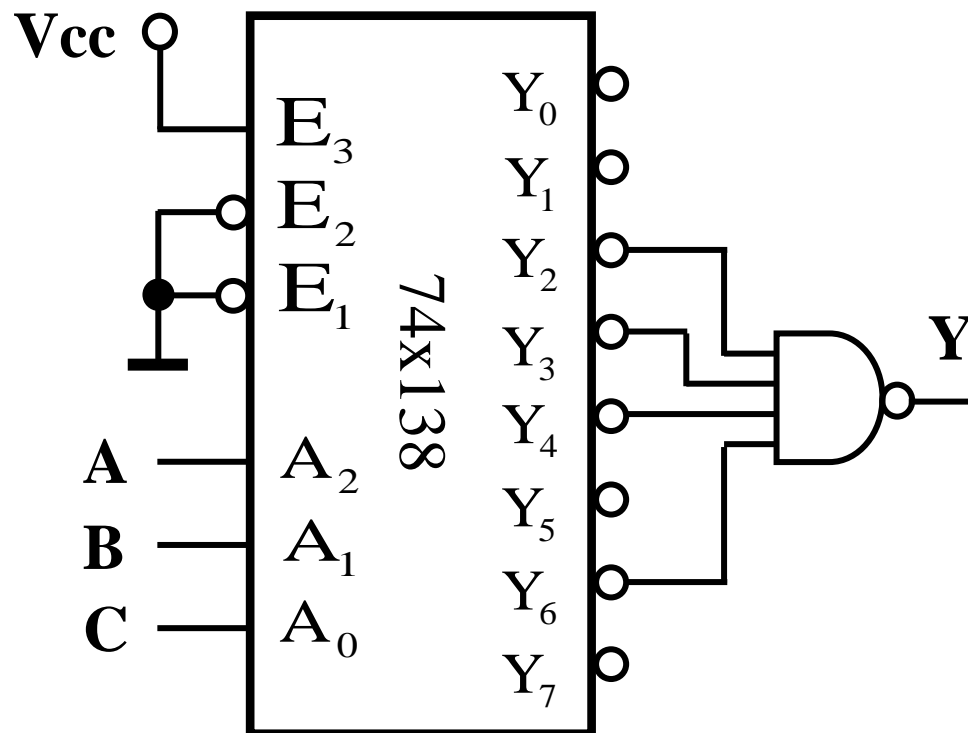
$$Y(A, B, C) = \sum m_i$$

$$= \overline{\prod \bar{m}_i} = \overline{\prod \bar{Y}_i}$$

$$Y(A, B, C) = \bar{A}B + A\bar{C}$$

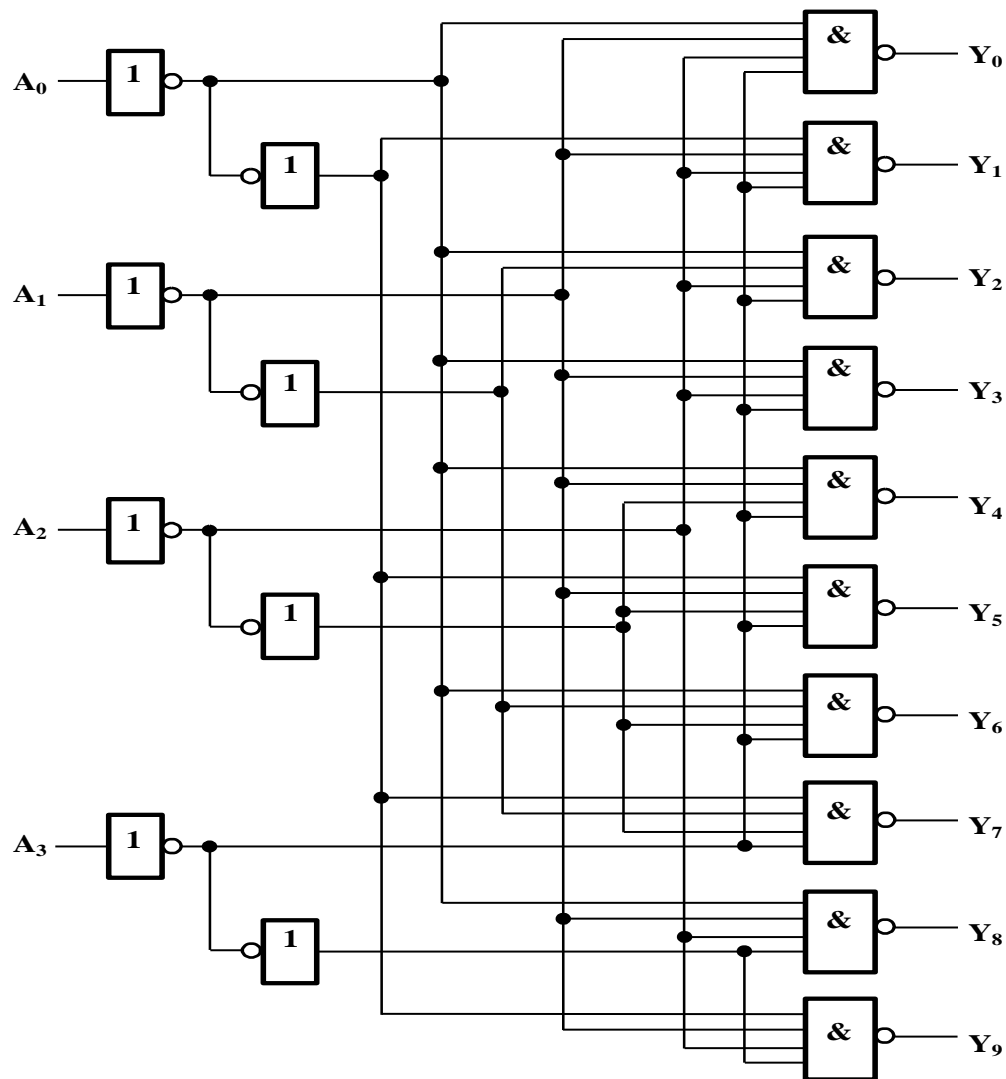
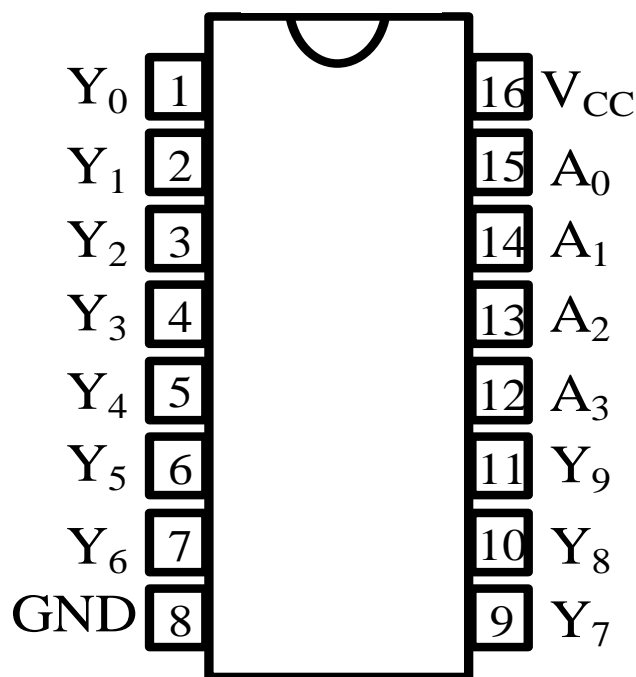
$$= m_2 + m_3 + m_4 + m_6$$

$$= \overline{\bar{Y}_2 \bar{Y}_3 \bar{Y}_4 \bar{Y}_6}$$



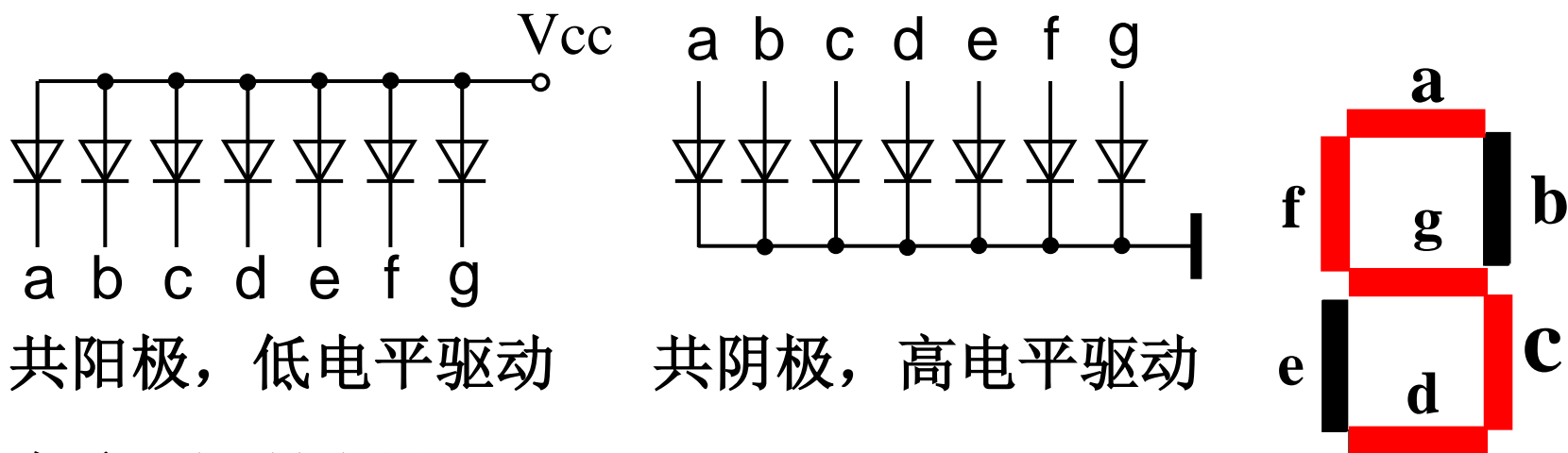
二-十进制译码器74x42

- 将8421BCD码译成10个状态输出



显示译码器

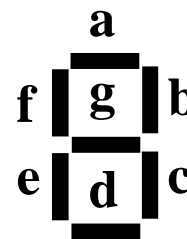
- 七段数码管：由七个发光二极管组成的显示器件



- 七段显示译码器

- 将BCD代码翻译成数码管所需的驱动信号
- 常见有74x48, 74x248, 74x4511等

七段显示译码器真值表




数字	A ₃	A ₂	A ₁	A ₀	a	b	c	d	e	f	g	字形
0	0	0	0	0	1	1	1	1	1	1	0	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	1	0	1	1	

74x4511功能表

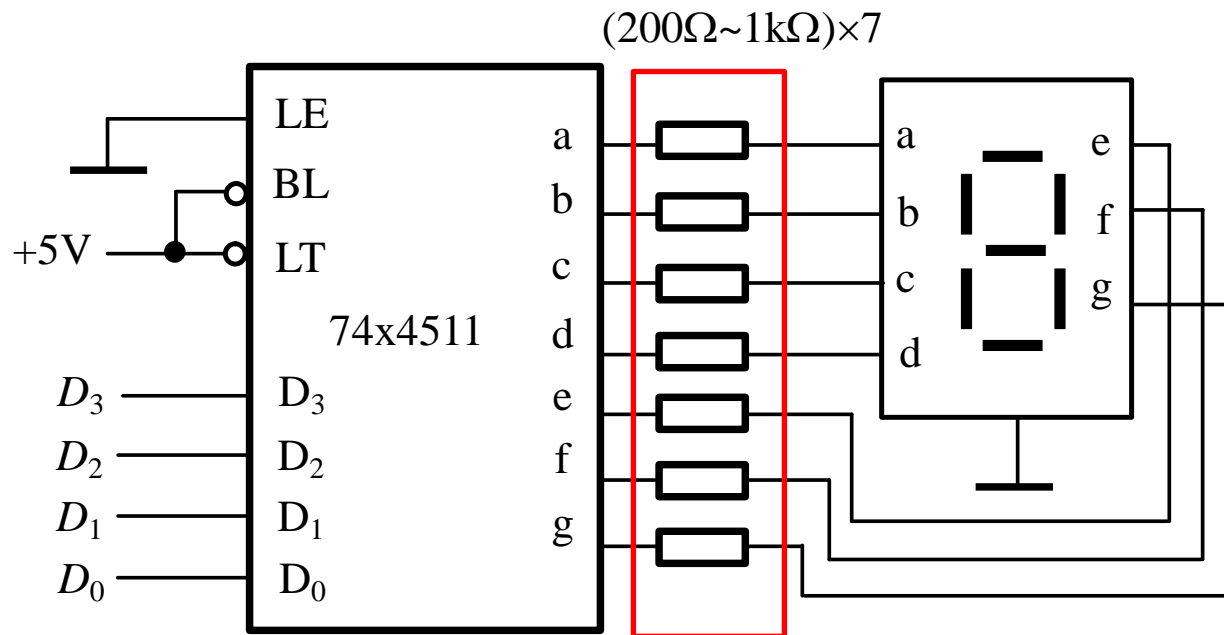
十进制 或功能	输 入							输 出							字形
	LE	\overline{BL}	\overline{LT}	D_3	D_2	D_1	D_0	a	b	c	d	e	f	g	
0	0	1	1	0	0	0	0	1	1	1	1	1	1	0	0
1	0	1	1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	1	1	0	0	1	0	1	1	0	1	1	0	1	2
3	0	1	1	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	1	0	1	0	0	0	1	1	0	0	1	1	4
5	0	1	1	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	1	0	0	0	1	1	1	1	1	6
7	0	1	1	0	1	1	1	1	1	1	0	0	0	0	7
8	0	1	1	1	0	0	0	1	1	1	1	1	1	1	8
9	0	1	1	1	0	0	1	1	1	1	1	0	1	1	9

74x4511功能表(续)

十进制 或功能	输 入							输 出							字形
	LE	\overline{BL}	\overline{LT}	D_3	D_2	D_1	D_0	a	b	c	d	e	f	g	
10	0	1	1	1	0	1	0	0	0	0	0	0	0	0	熄灭
11	0	1	1	1	0	1	1	0	0	0	0	0	0	0	熄灭
12	0	1	1	1	1	0	0	0	0	0	0	0	0	0	熄灭
13	0	1	1	1	1	0	1	0	0	0	0	0	0	0	熄灭
14	0	1	1	1	1	1	0	0	0	0	0	0	0	0	熄灭
15	0	1	1	1	1	1	1	0	0	0	0	0	0	0	熄灭
灯 测 试	×	×	0	×	×	×	×	1	1	1	1	1	1	1	
灭 灯	×	0	1	×	×	×	×	0	0	0	0	0	0	0	熄灭
锁 存	1	1	1	×	×	×	×	*							*

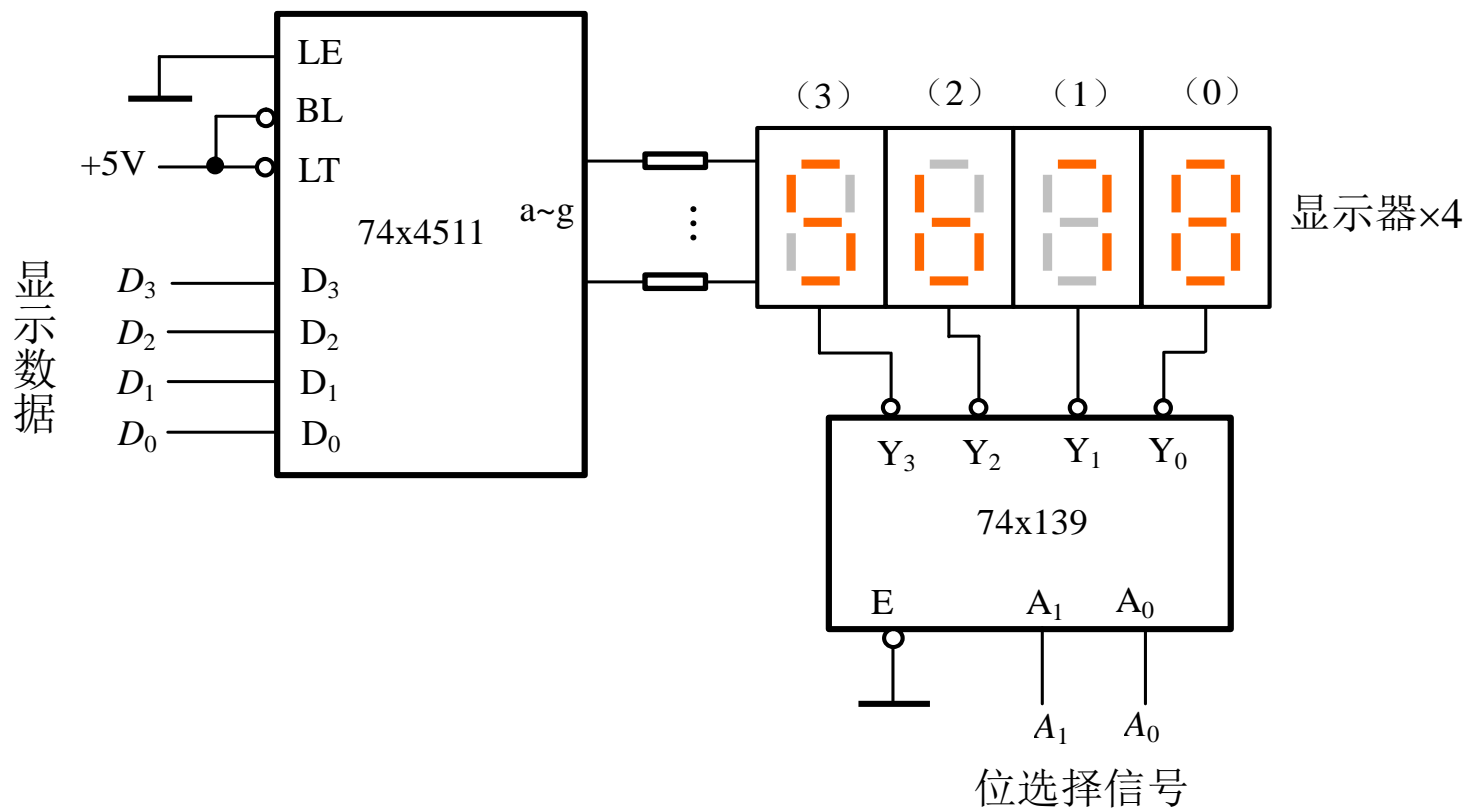
74x4511应用(1)

- 显示译码器与7段数码管的连接方式



请在讲完模电二极管之后思考这些电阻的作用

74x4511应用(2)



位选择信号 A_1 、 A_0 控制 $\overline{Y_3} \sim \overline{Y_0}$ 依次产生低电平，使4个显示器轮流显示。利用人的视觉暂留时间，可以看到稳定的数字。

$$25\text{Hz} < f_C < 100\text{Hz}$$

重点回顾

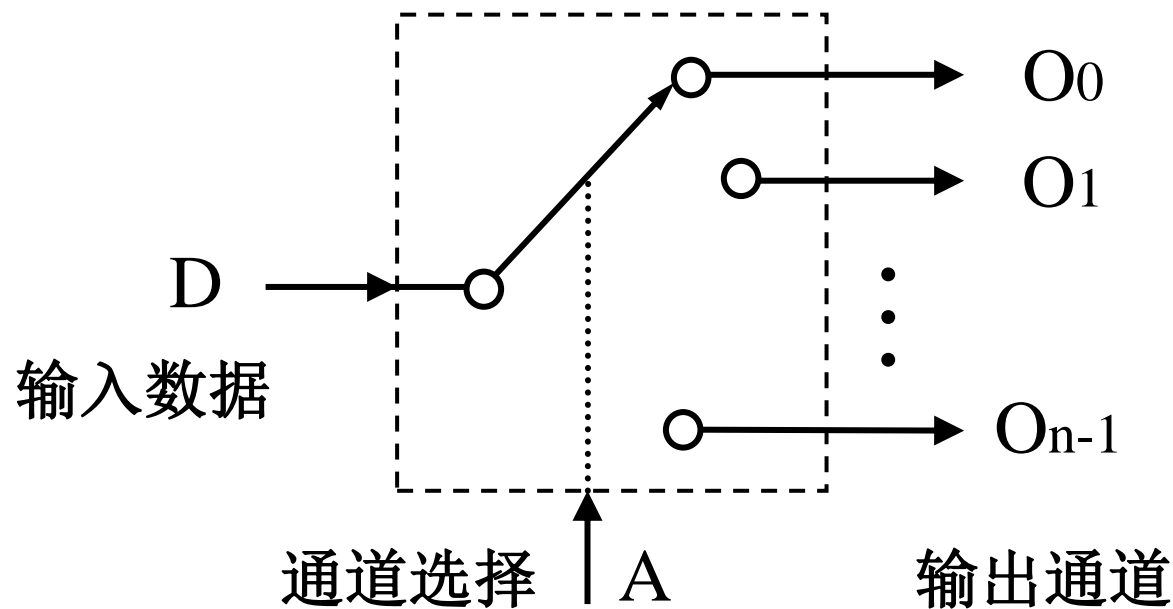
- **编码器**
 - 普通编码器（电路设计、好处、缺点）
 - 优先编码器（电路设计、好处、缺点）
 - 74x148：结构、扩展原理
- **译码器**
 - 电路设计
 - 74x139/74x138：结构、扩展、任意三变量逻辑函数实现方法
 - 七段显示译码器：原理、应用（视觉暂留）

内容提纲

- 数据分配器
- 数据选择器

数据分配器

- 将一路输入数据分配到多路输出端的电路
 - 在通道选择信号控制下，可以将输入数据传送到多个输出通道中的任何一个通道输出



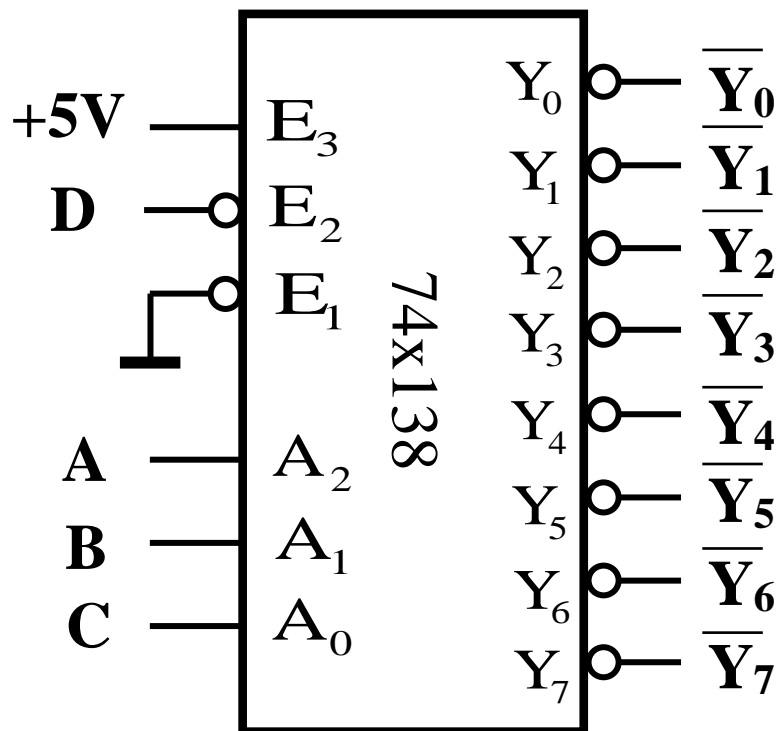
译码器实现数据分配器

$$\overline{Y_i} = \overline{E_3} \overline{E_2} \overline{E_1} m_i$$

$$\overline{Y_i} = \overline{D} m_i$$

当 $ABC = 000$ 时

$$\overline{Y_i} = \begin{cases} D & , i = 0 \\ 1 & , i = 1 \sim 7 \end{cases}$$



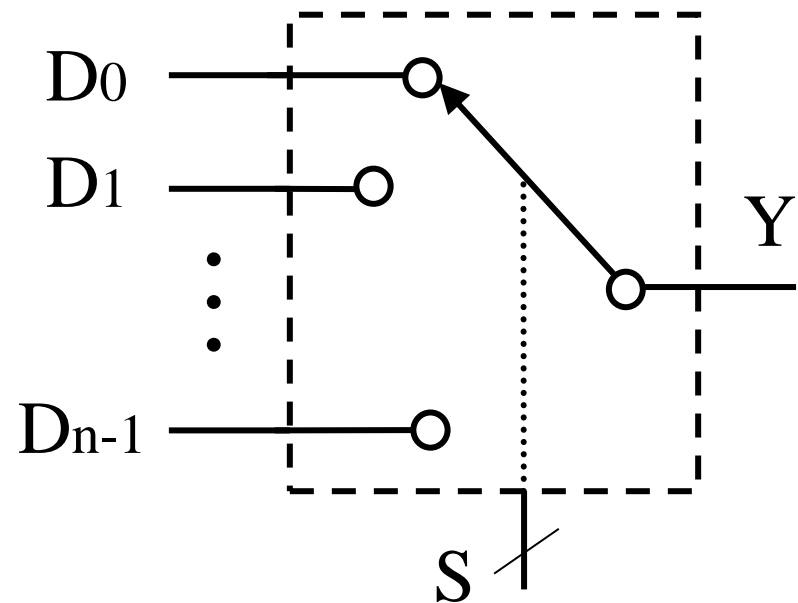
按照通道地址 $A_2A_1A_0$ 的取值，将输入数据D从相应的输出通道 $\overline{Y_i}$ 输出，其他输出通道保持高电平

译码器实现数据分配器

输 入						输 出							
E_3	\overline{E}_2	\overline{E}_1	A_2	A_1	A_0	\overline{Y}_0	\overline{Y}_1	\overline{Y}_2	\overline{Y}_3	\overline{Y}_4	\overline{Y}_5	\overline{Y}_6	\overline{Y}_7
0	x	0	x	x	x	1	1	1	1	1	1	1	1
1	D	0	0	0	0	D	1	1	1	1	1	1	1
1	D	0	0	0	1	1	D	1	1	1	1	1	1
1	D	0	0	1	0	1	1	D	1	1	1	1	1
1	D	0	0	1	1	1	1	1	D	1	1	1	1
1	D	0	1	0	0	1	1	1	1	D	1	1	1
1	D	0	1	0	1	1	1	1	1	1	D	1	1
1	D	0	1	1	0	1	1	1	1	1	1	D	1
1	D	0	1	1	1	1	1	1	1	1	1	1	D

数据选择器

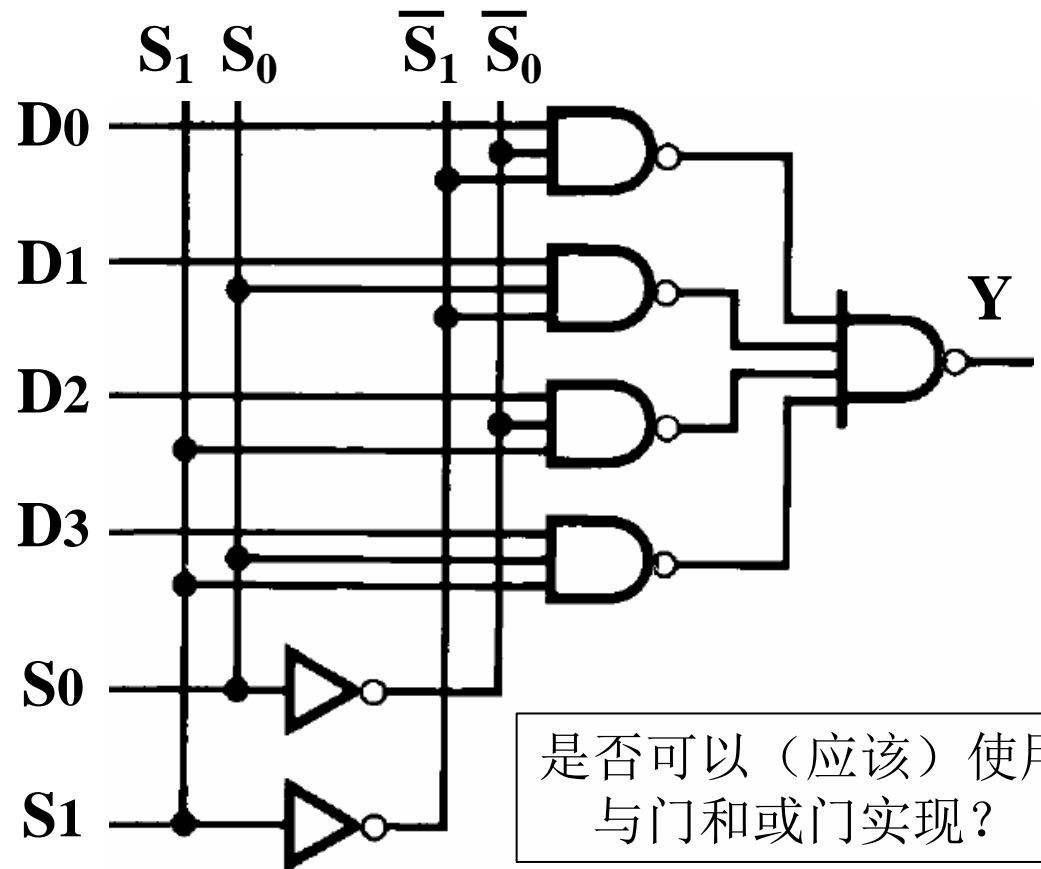
- 数据选择器(Multiplexer, 简称MUX): 根据通道选择信号, 从多个通道(路)输入数据中选择一个通道数据输出, 也称多路选择器
- 常见集成数据选择器
 - 2选1(74x157)
 - 4选1(74x153)
 - 8选1(74x151)
 - 16选1(74x150)等



设计4选1数据选择器

功能表

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



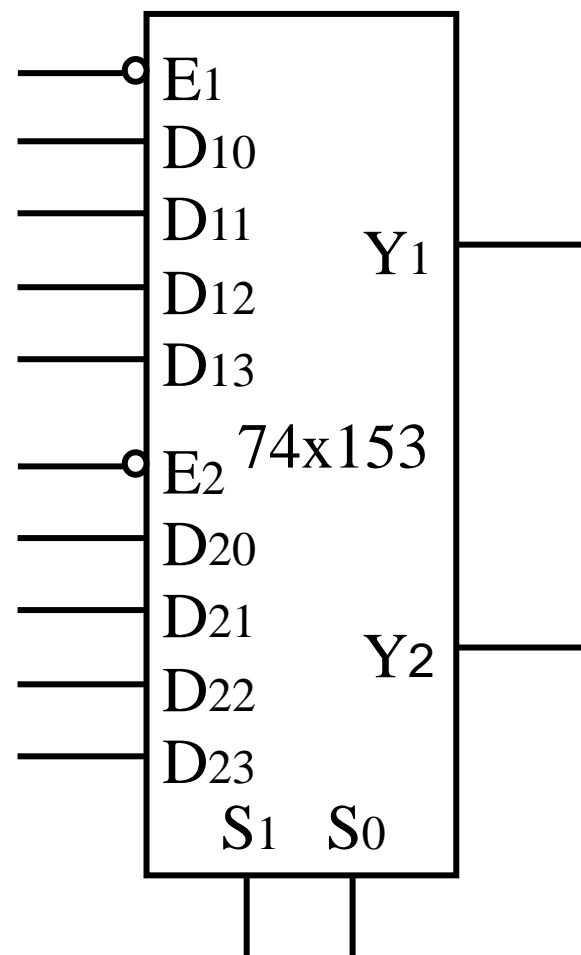
$$Y = \bar{S}_1 \bar{S}_0 D_0 + \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 + S_1 S_0 D_3 = \sum_{i=0}^3 m_i D_i$$

双4选1数据选择器74x153

- 公用通道选择控制
- 独立选通(使能)控制

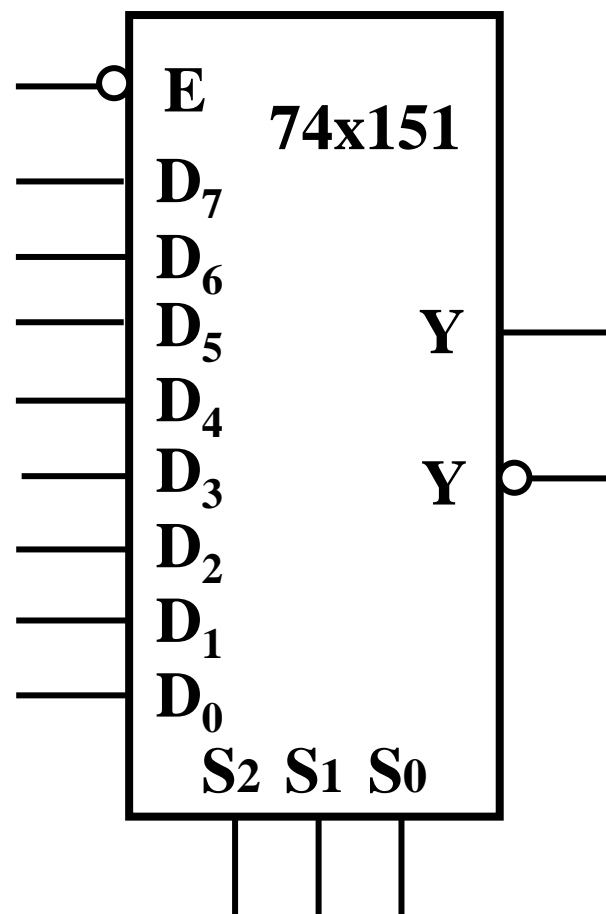
功能表

\overline{E}	S_1	S_0	Y
1	x	x	0
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3



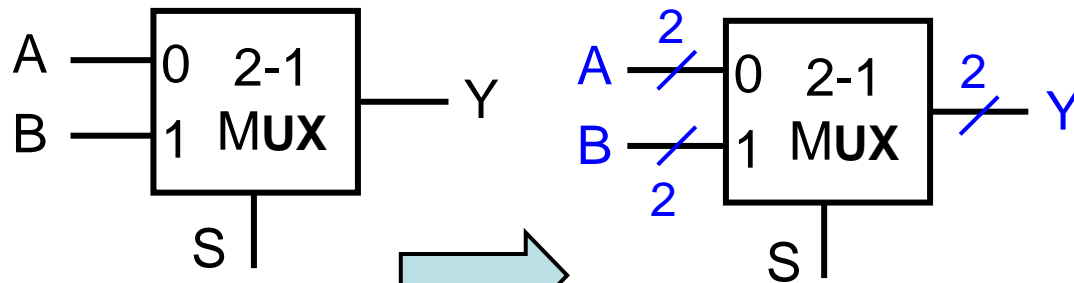
8选1数据选择器74x151

- 带使能和互补输出的8通道数据选择器
- $D_0 \sim D_7$: 8路数据输入
- Y 、 \overline{Y} : 互补输出
- $S_2 \sim S_0$: 通道选择输入, S_2 为最高位
- \overline{E} : 使能输入, 低电平有效
 - $\overline{E}=0$ 时, $Y=D_i$, $\overline{Y}=\overline{D_i}$
 - $\overline{E}=1$ 时, $Y=0$, $\overline{Y}=1$

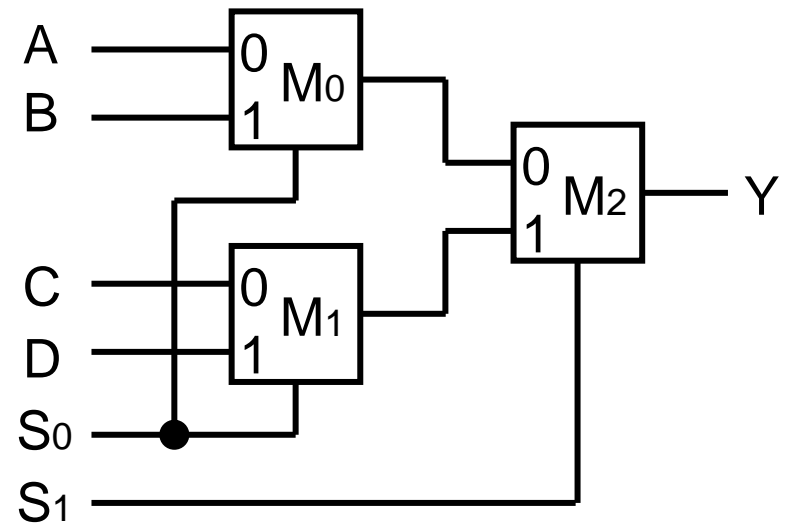
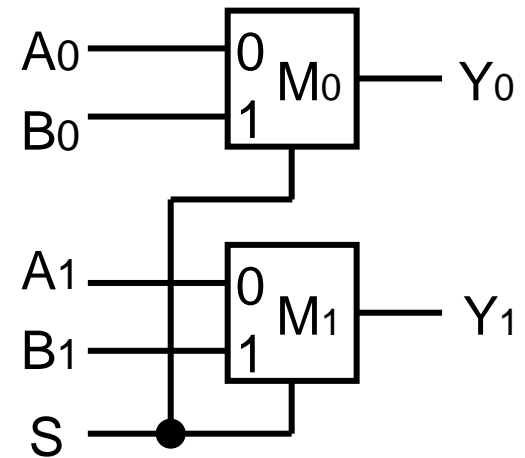
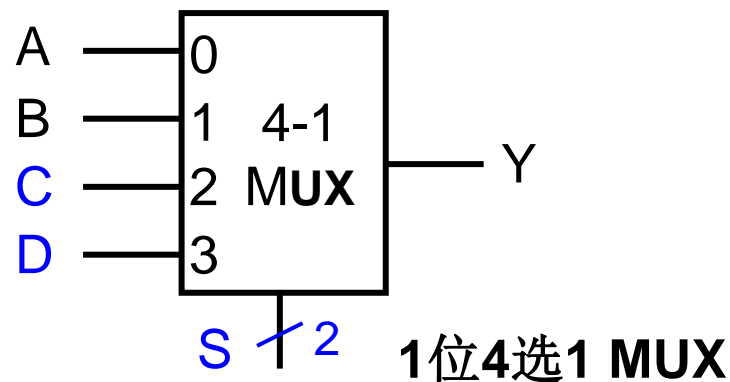


数据选择器的扩展

- 用1位2选1 MUX进行扩展设计

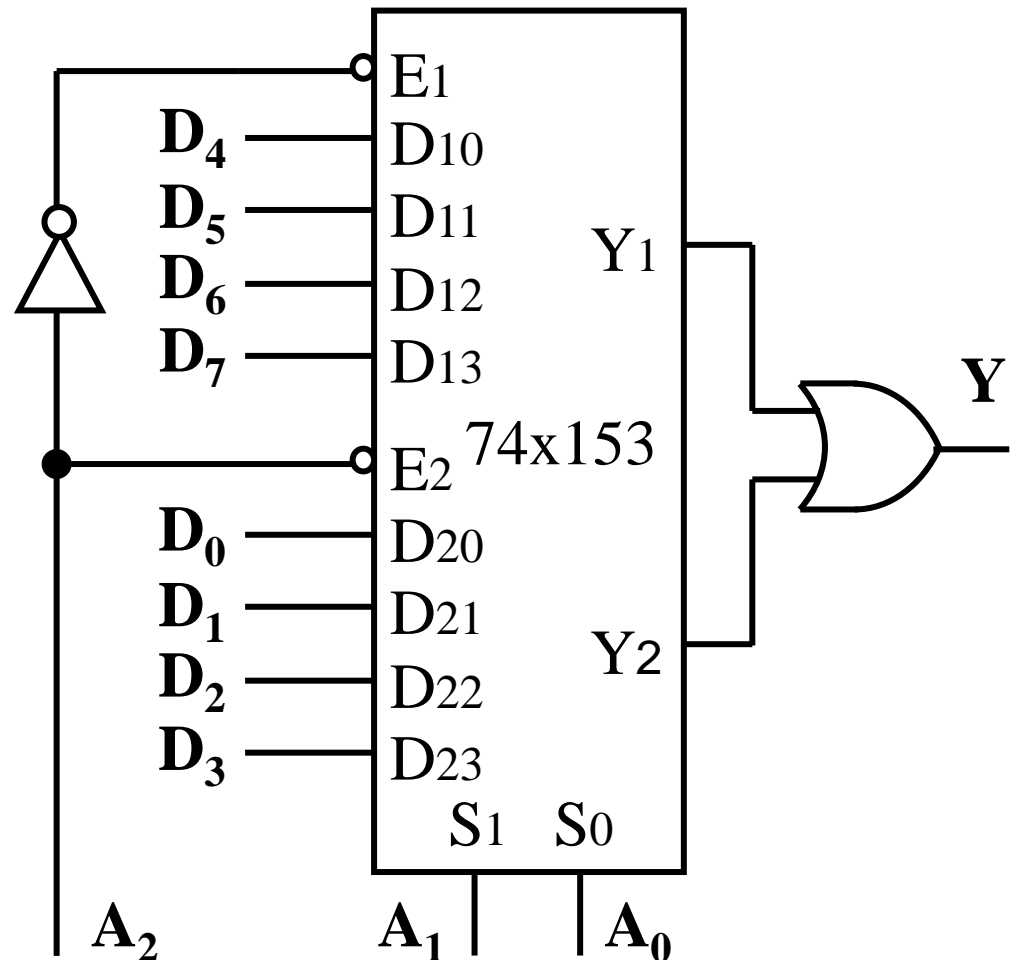
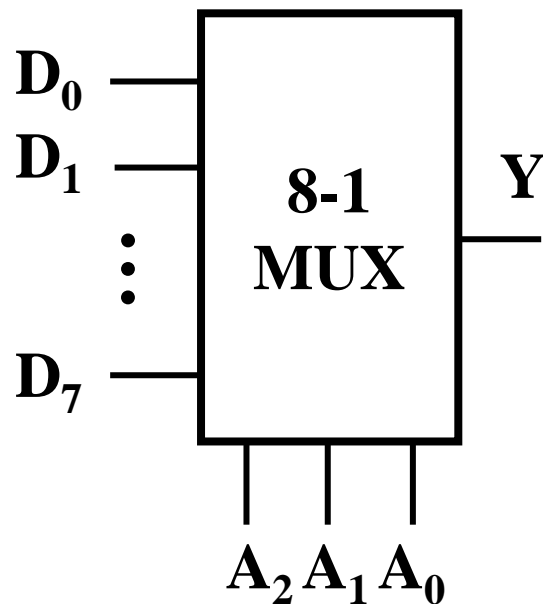


字扩展



数据选择器的扩展(续)

- 用1片74x153构成8选1数据选择器



数据选择器实现组合逻辑函数

- 用1片74x153和非门实现

$$Y_1 = \overline{A}B + A\overline{B}$$

$$= 1 \cdot (m_1 + m_2) + 0 \cdot (m_0 + m_3)$$

$$Y_2 = \overline{A}C + \overline{B}\overline{C}$$

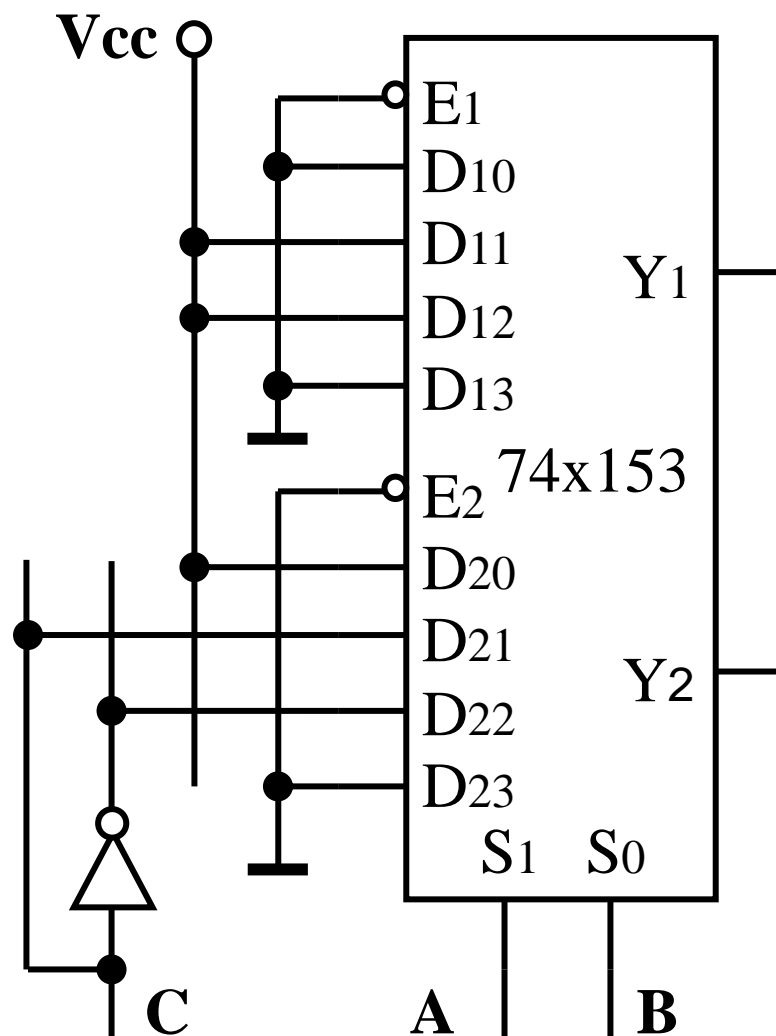
$$= 1 \cdot m_0 + C \cdot m_1 + \overline{C} \cdot m_2 + 0 \cdot m_3$$

		AB			
		00	01	11	10
Y ₂	C				
0		1	0	0	1
1		1	1	0	0

1*m₀

C*m₁

/C*m₂



重点回顾

- **数据分配器**
 - 按照地址线，将输入送到指定道上输出
 - 可用译码器74x138实现
- **数据选择器**
 - 按照地址线，将指定道上的输入送到输出
 - 扩展方法（字扩展，位扩展）
 - 数据选择器实现组合逻辑

内容提纲

- 比较器
- 加法器

数值比较器

- 判断两个二进制数大小关系的逻辑电路
- 设计一位数值比较器
 - 输入：2个1位数A和B
 - 输出： $F_{A>B}$ 、 $F_{A<B}$ 、 $F_{A=B}$

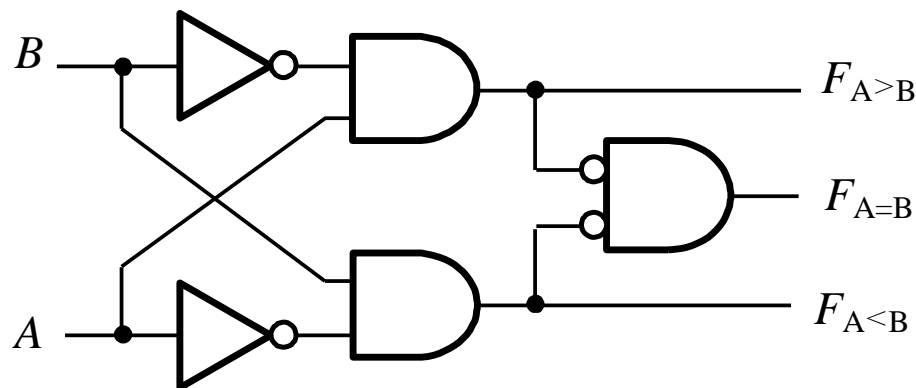
$$F_{A>B} = A\bar{B}$$

$$F_{A<B} = \bar{A}B$$

$$F_{A=B} = \bar{A}\bar{B} + AB$$

真值表

A	B	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1



逻辑图

两位数值比较器

- 利用一位数值比较器设计两位数值比较器
- 待比较的两个2位二进制数： $A = A_1A_0$ ， $B = B_1B_0$
 - 先比较高位 A_1 和 B_1
 - 只有高位相等，才比较低位 A_0 和 B_0

真值表

A_1 B_1	A_0 B_0	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
$A_1>B_1$	x	1	0	0
$A_1<B_1$	x	0	1	0
$A_1=B_1$	$A_0>B_0$	1	0	0
	$A_0<B_0$	0	1	0
	$A_0=B_0$	0	0	1

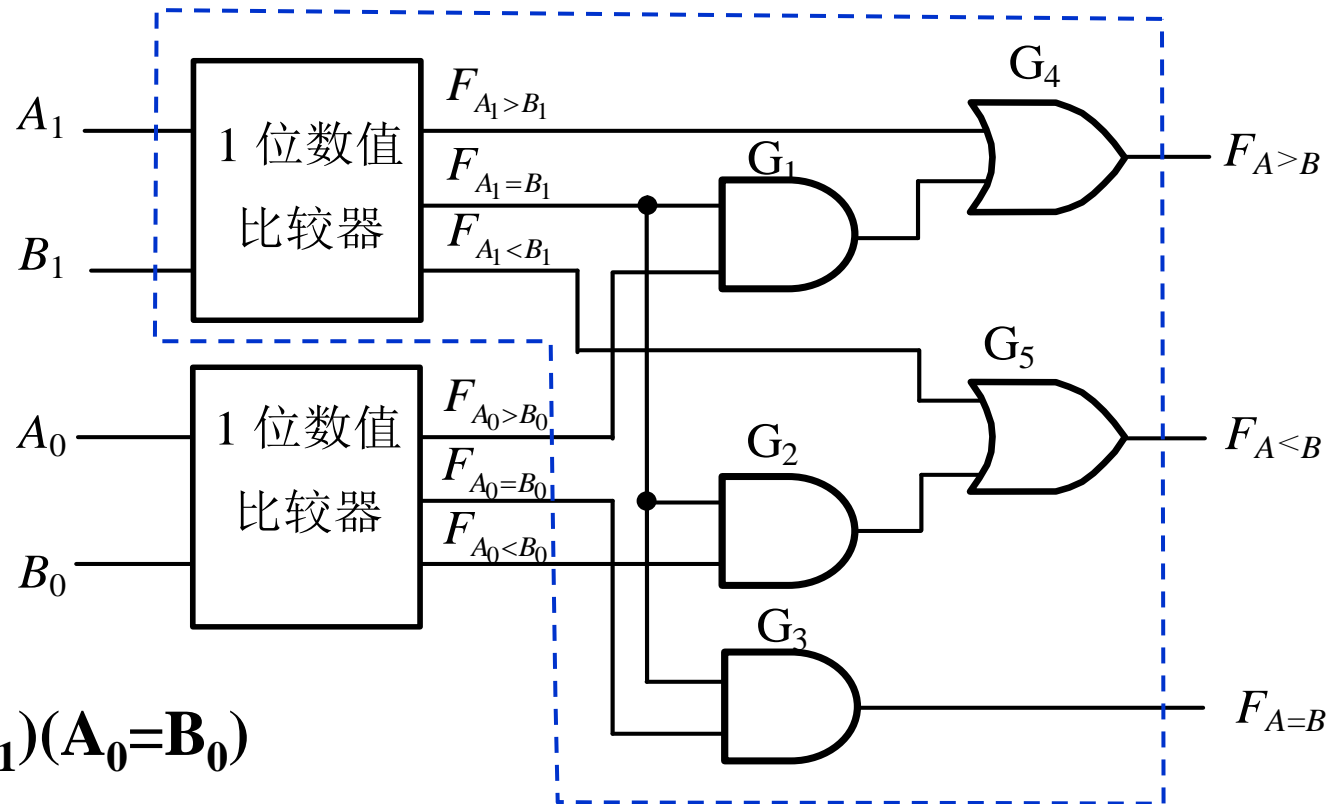
$$F_{A>B} = (A_1>B_1) + (A_1=B_1)(A_0>B_0)$$

$$F_{A<B} = (A_1<B_1) + (A_1=B_1)(A_0<B_0)$$

$$F_{A=B} = (A_1=B_1)(A_0=B_0)$$

两位数值比较器(续)

- 逻辑图



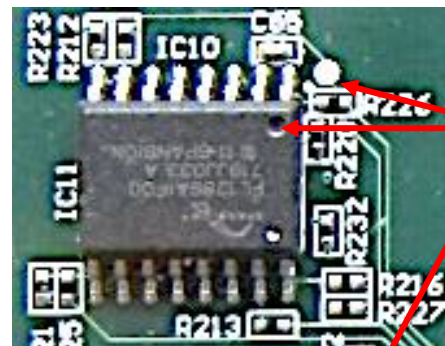
$$F_{A=B} = (A_1=B_1)(A_0=B_0)$$

$$F_{A>B} = (A_1>B_1) + (A_1=B_1)(A_0>B_0)$$

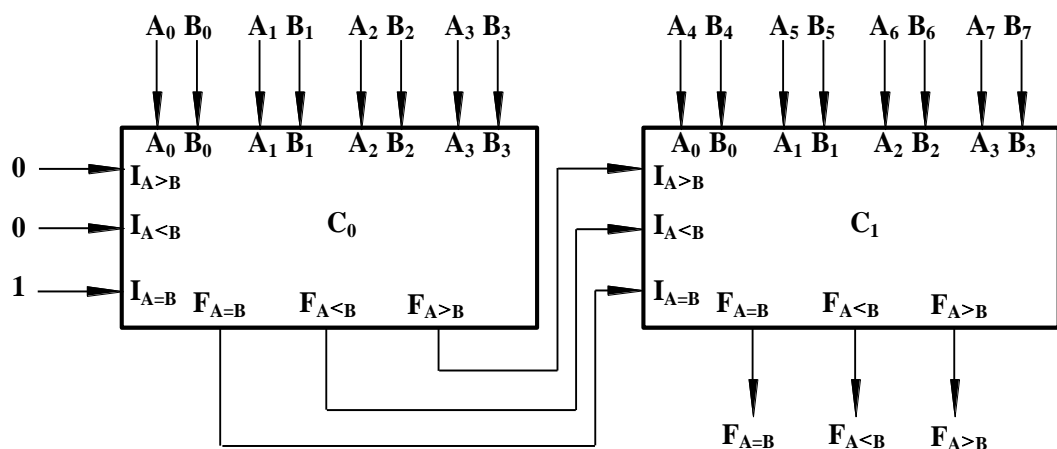
$$F_{A<B} = (A_1<B_1) + (A_1=B_1)(A_0<B_0)$$

四位数值比较器74x85

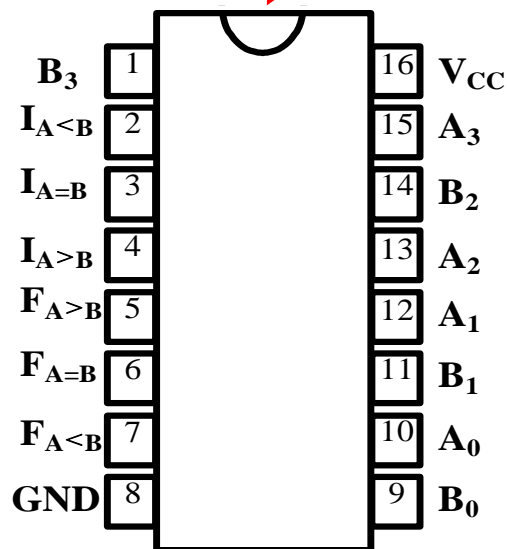
- 工作原理和两位数值比较器相同
- 提供附加输入端 $I_{A<B}$ 、 $I_{A=B}$ 和 $I_{A>B}$ ，便于扩展应用
 - 从高位组比起，若不等，出结果，否则还需比较次高位组



1
脚
标
志



八位串联数值比较器



引脚图

加法器

- 加法器是算术运算(加、减、乘、除)电路的基本单元
- 1位加法器
 - 1位半加器
 - 1位全加器
- 由1位加法器构成多位加法器
 - 串行进位加法器
 - 超前进位加法器

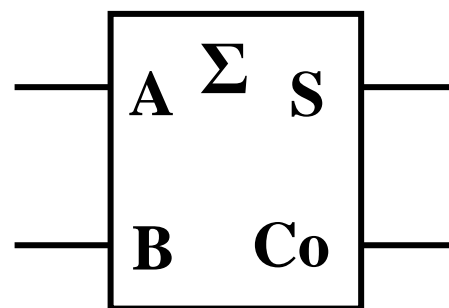
1位半加器

- 将两个1位数相加，产生1位和、1位进位
– $\{C_o, S\} = A + B$

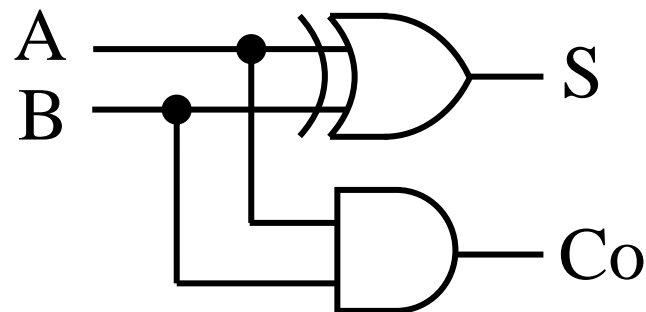
A	B	C _o	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = \bar{A}B + A\bar{B} = A \oplus B$$

$$C_o = AB$$



逻辑符号

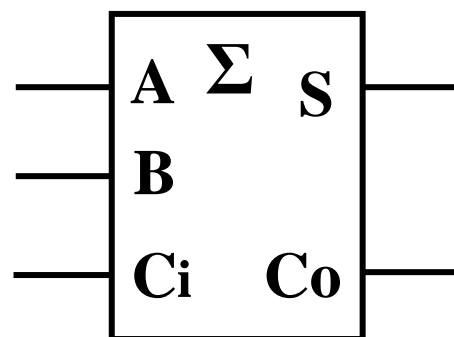


1位全加器

- 将两个1位数与来自低位的进位相加，产生1位的和，以及向高位的进位

$$- \{C_o, S\} = A + B + C_i$$

C_i	A	B	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



逻辑符号

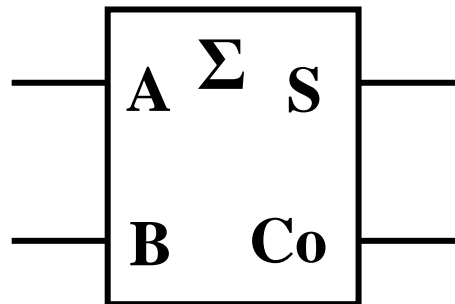
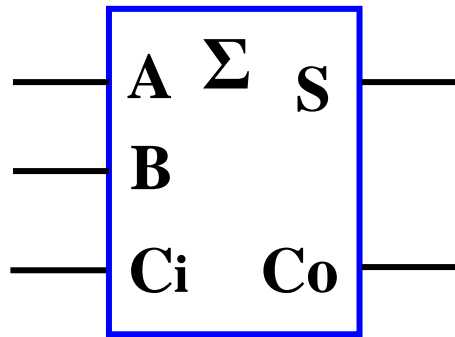
$$S = \overline{A}B\overline{C}_i + A\overline{B}\overline{C}_i + \overline{A}\overline{B}C_i + ABC_i$$

$$= A \oplus B \oplus C_i$$

$$C_o = \underbrace{AB\overline{C}_i}_{\text{blue}} + \underbrace{\overline{A}BC_i}_{\text{red}} + \underbrace{A\overline{B}C_i}_{\text{red}} + \underbrace{ABC_i}_{\text{blue}}$$

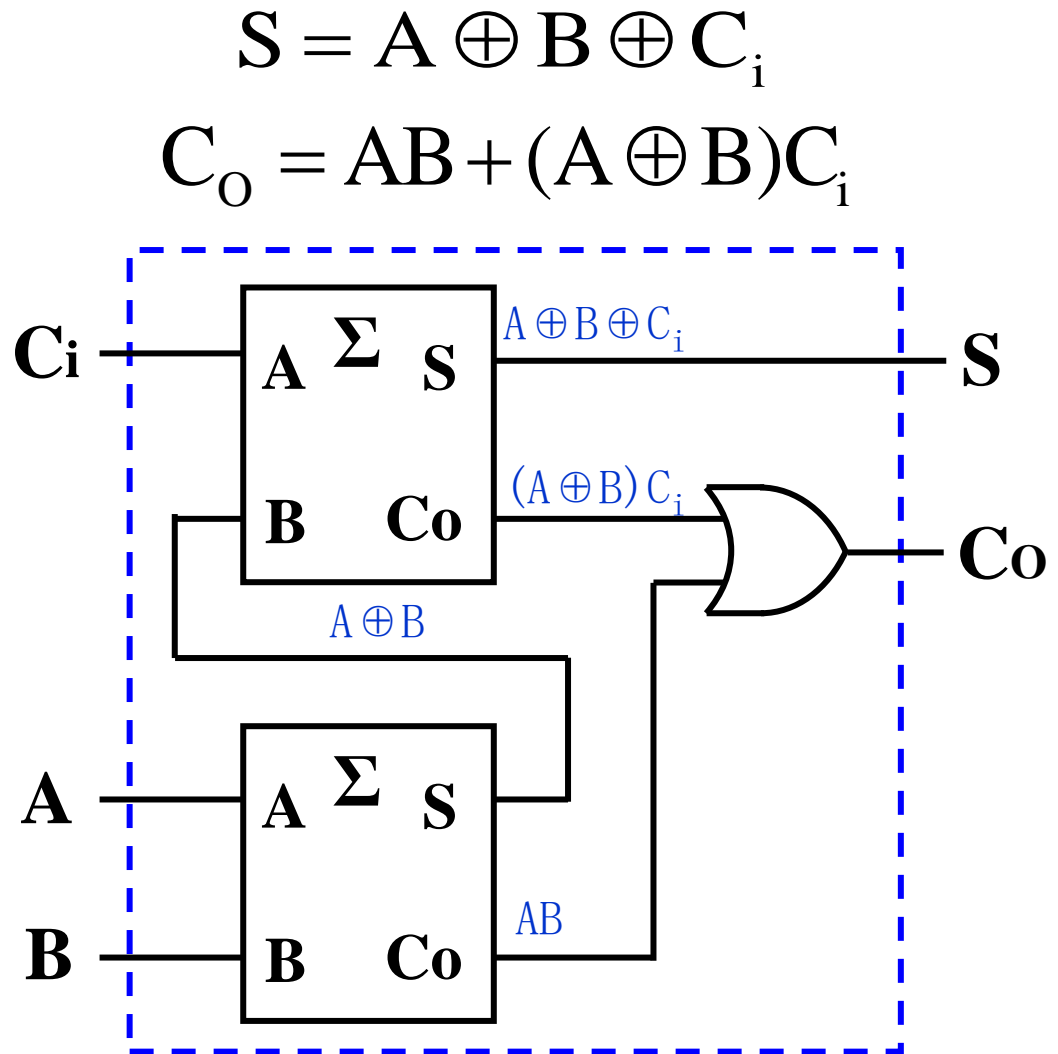
$$= AB + (A \oplus B)C_i$$

1位全加器逻辑图



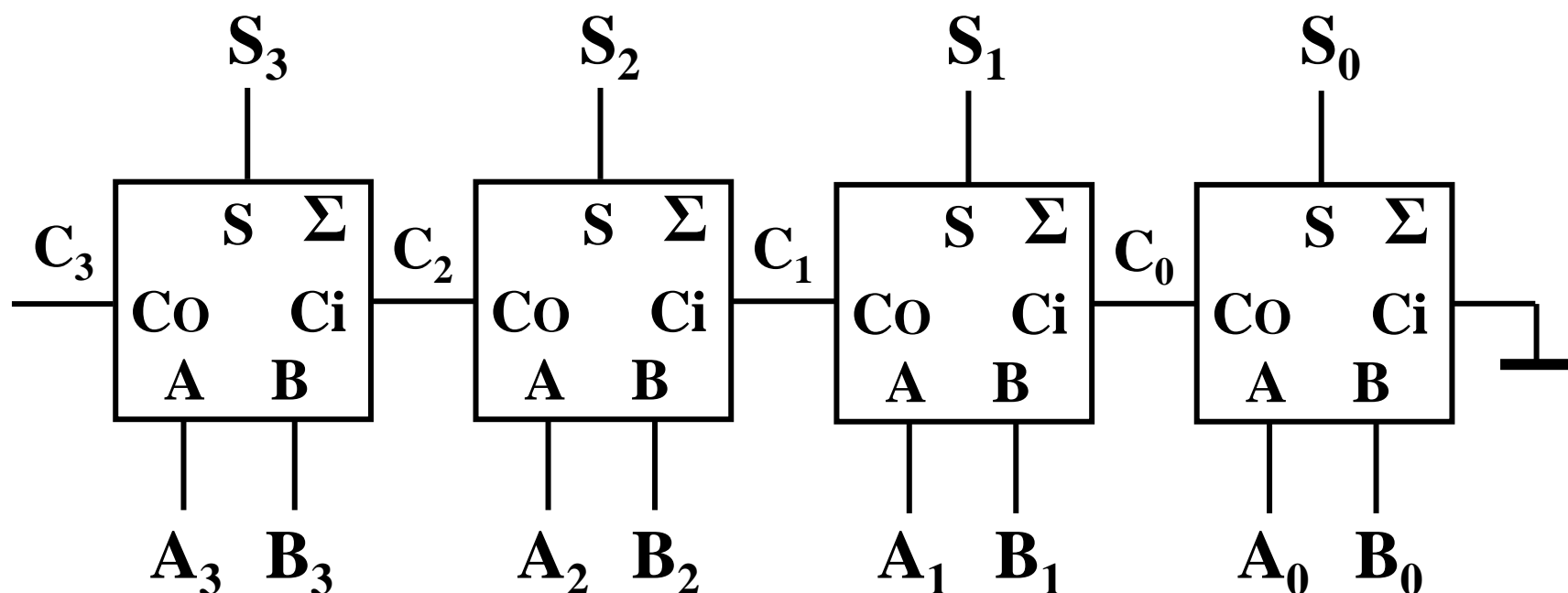
$$S = A \oplus B$$

$$C_o = A B$$



串行进位加法器

- 用1位全加器构造4位加法器



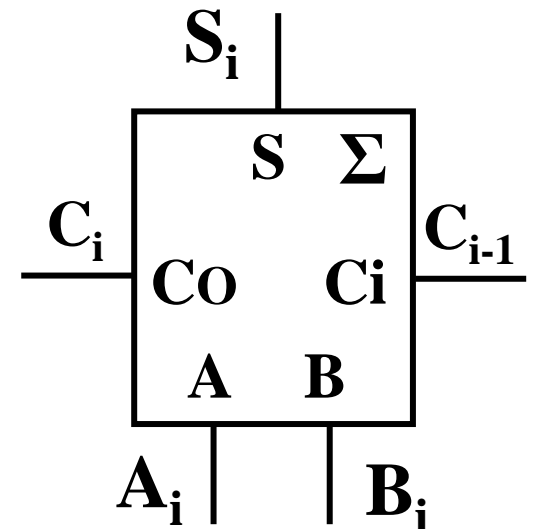
优点：简单，易于扩展；缺点：速度慢

超前进位加法器

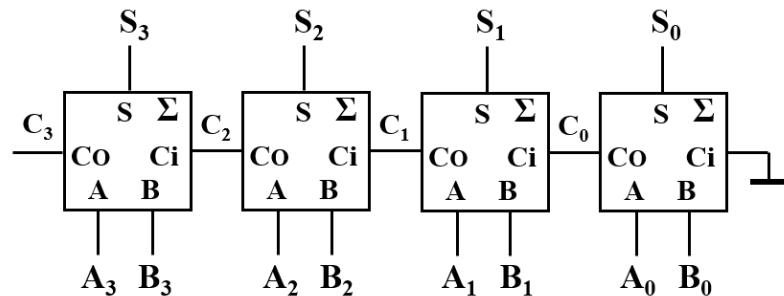
- 基本原理

- C_{i-1} 是 $A_{i-1} \sim A_0$ 和 $B_{i-1} \sim B_0$ 的函数
 - 设计每位进位信号产生电路：根据输入加数和被加数，同时获得该位全加的进位信号，无需等待最低位的进位信号

- 优点：速度快
- 缺点：电路复杂
- 4位超前进位加法器74x283



进位信号超前产生



$$C_i = A_i B_i + (A_i \oplus B_i) C_{i-1} \quad S_i = A_i \oplus B_i \oplus C_{i-1}$$

令 $G_i = A_i B_i$, $P_i = (A_i \oplus B_i)$ 则 $C_i = G_i + P_i C_{i-1}$

$$C_0 = \underline{G_0 + P_0 C_{-1}}$$

$$S_i = P_i \oplus C_{i-1}$$

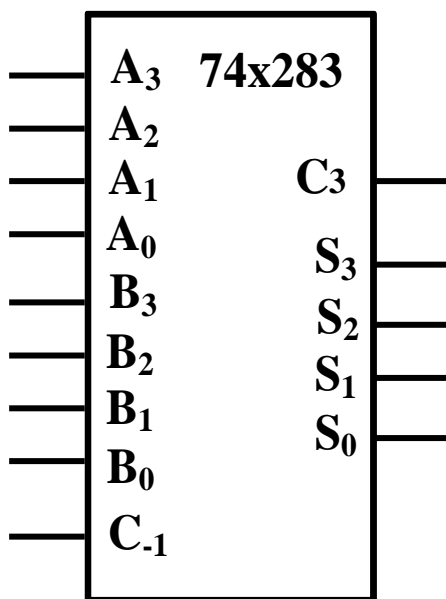
$$C_1 = G_1 + P_1 C_0 = \underline{G_1 + P_1 G_0 + P_1 P_0 C_{-1}}$$

$$C_i = \sum_{j=0}^i \prod_{k=j+1}^i P_k G_j + \prod_{k=0}^i P_k C_{-1}$$

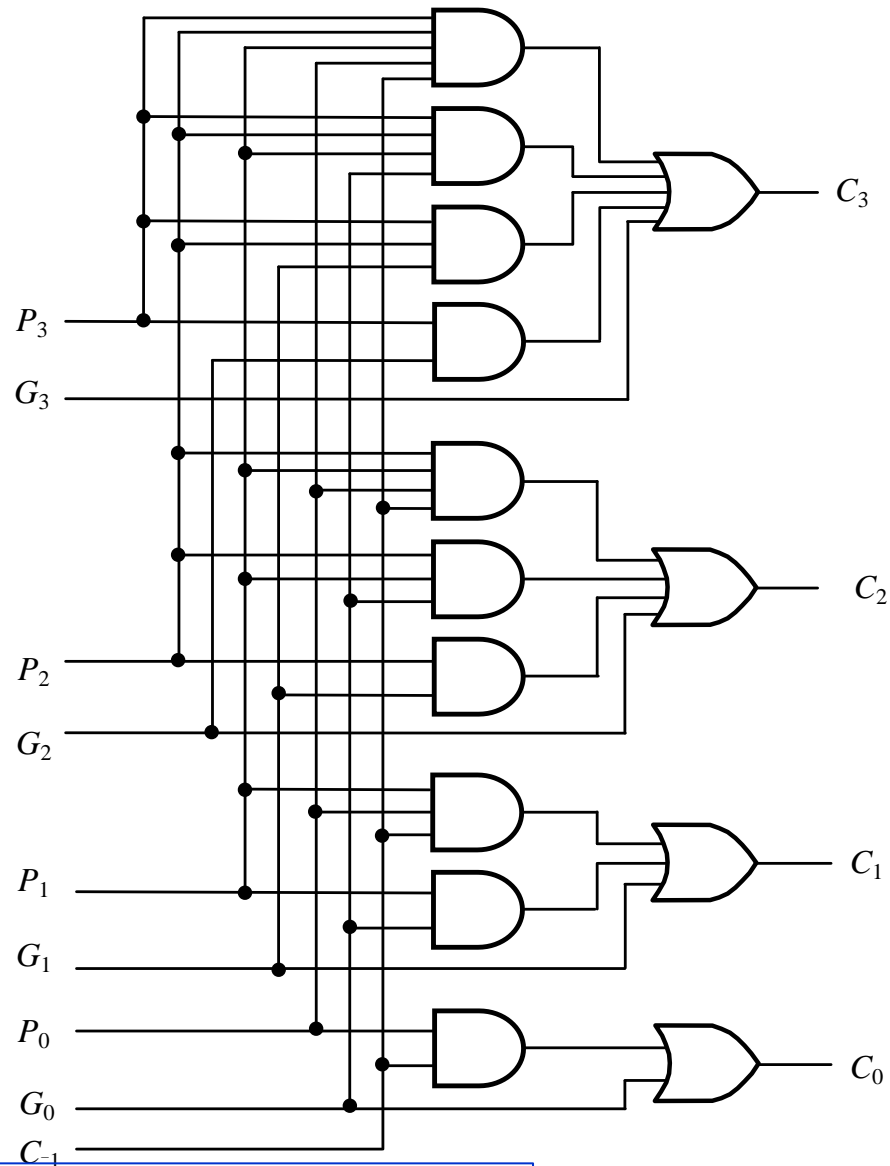
$$C_2 = G_2 + P_2 C_1 = \underline{G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{-1}}$$

$$C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 \\ + \underline{P_3 P_2 P_1 P_0 C_{-1}}$$

4位超前进位 加法器74x283



逻辑符号



$$P_i = A_i \oplus B_i \quad G_i = A_i B_i$$

$$C_i = \sum_{j=0}^i \prod_{k=j+1}^i P_k G_j + \prod_{k=0}^i P_k C_{-1}$$

$$S_i = P_i \oplus C_{i-1}$$

门延迟 (gate delay)

In [electronics](#), [digital circuits](#) and [digital electronics](#), the propagation delay, or **gate delay**, is the length of time which starts when the input to a [logic gate](#) becomes stable and valid to change, to the time that the output of that logic gate is stable and valid to change. Often on manufacturers' datasheets this refers to the time required for the output to reach 50% of its final output level when the input changes to 50% of its final input level.

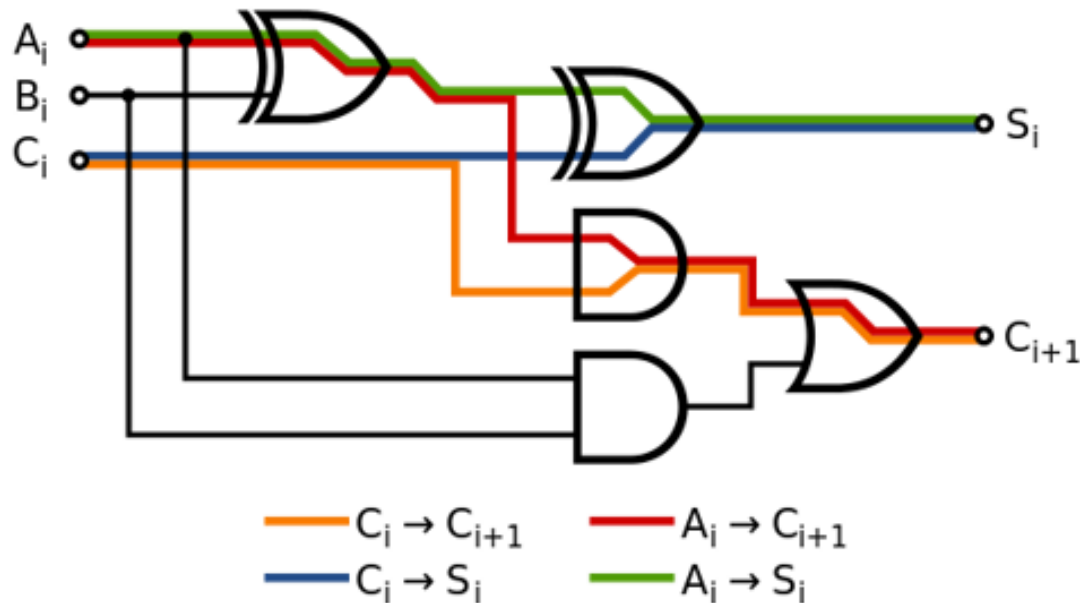
SN54147, SN74147 switching characteristics, $V_{CC} = 5\text{ V}$, $T_A = 25^\circ\text{C}$ (see Figure 1)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	WAVEFORM	TEST CONDITIONS	MIN	TYP	MAX	UNIT
tPLH	Any	Any	In-phase output	C _L = 15 pF, R _L = 400 Ω		9	14	ns
tPHL						7	11	
tPLH	Any	Any	Out-of-phase output			13	19	ns
tPHL						12	19	

https://en.wikipedia.org/wiki/Propagation_delay

门延迟 (gate delay)

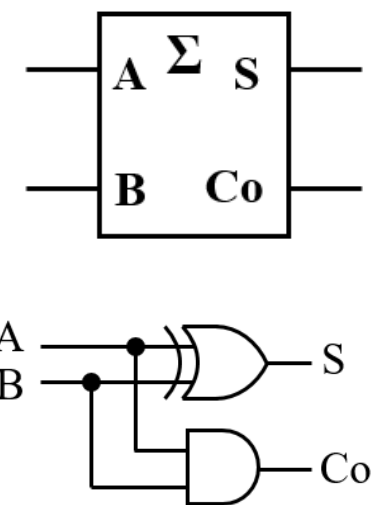
Reducing gate delays in [digital circuits](https://en.wikipedia.org/wiki/Digital_circuits) allows them to process data at a faster rate and improve overall performance. The determination of the propagation delay of a combined circuit requires identifying the longest path of propagation delays from input to output and by adding each tpd time along this path.



https://en.wikipedia.org/wiki/Propagation_delay

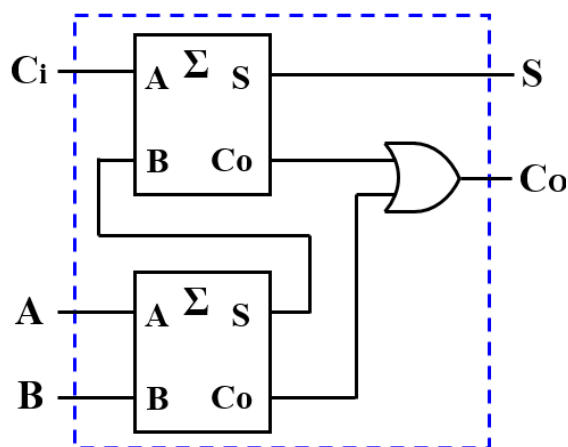
延迟分析

- 使用1位全加器实现4位全加器



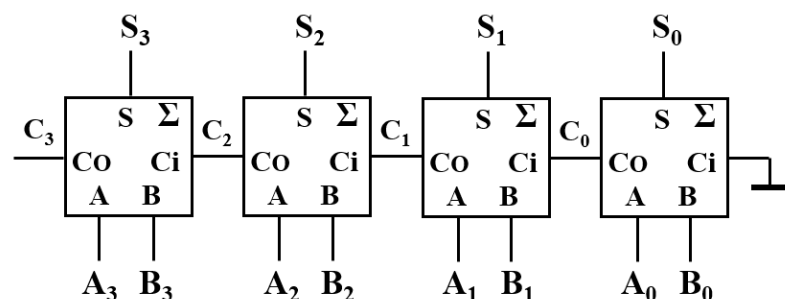
1位半加器

使用 2个门
S门延迟 1
Co门延迟 1



1位全加器

使用 $2*2+1 = 5$ 个门
S门延迟 2 (Ci→S,1; A/B→S,2)
Co门延迟 3 (Ci→Co,2; A/B→Co,3)



4位全加器

使用 $5*4 = 20$ 个门

S0门延迟 2, Co门延迟 3

S1门延迟 $3+1=4$, C1门延迟 $3+2=5$

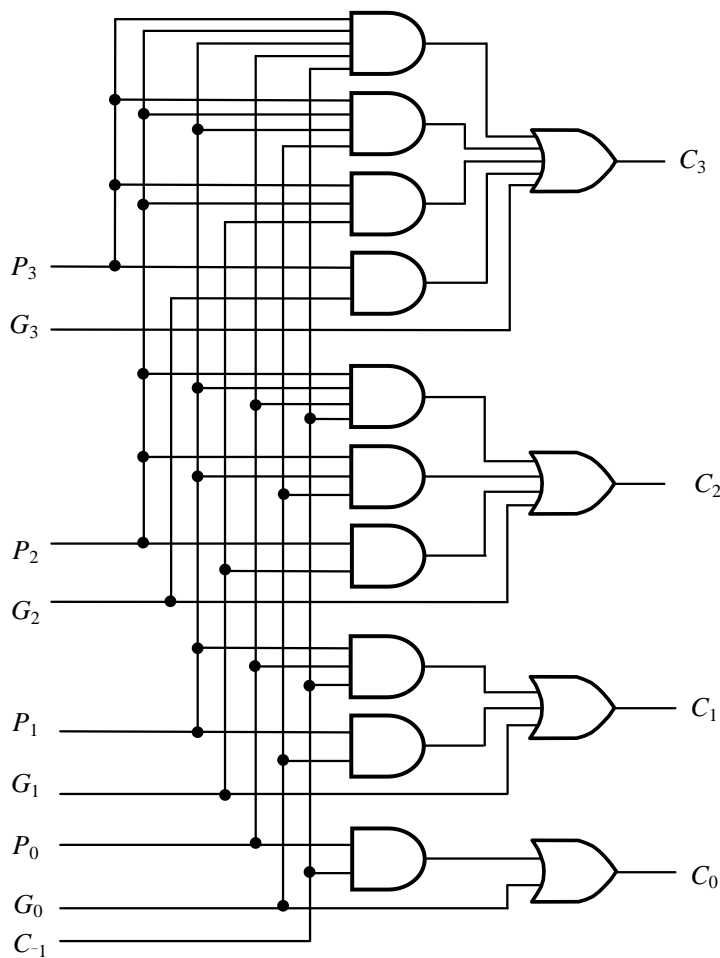
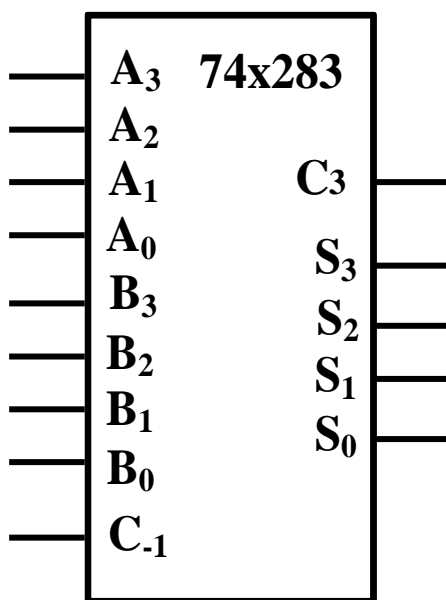
S2门延迟 $5+1=6$, C2门延迟 $5+2=7$

S3门延迟 $7+1=8$, **C3门延迟 $7+2=9$**

总共使用门数

$$4+4+14+4=30$$

P_i G_i C_i S_i



$$P_i = A_i \oplus B_i \quad G_i = A_i B_i$$

$$C_i = \sum_{j=0}^i \prod_{k=j+1}^i P_k G_j + \prod_{k=0}^i P_j C_{-1} \quad S_i = P_i \oplus C_{i-1}$$

P_0, P_1, P_2, P_3 门延迟均为1

G_0, G_1, G_2, G_3 门延迟均为1

C_0, C_1, C_2, C_3 门延迟均为1+2=3

S_0, S_1, S_2, S_3 门延迟均为 3+1=4

且不随着位数增加而增加!

延迟分析

- **4位全加器**

- 1位全加器串联:

- 使用20个门, 最大门延迟 $3+2*3=9$

- 超前进位:

- 使用30个门, 最大门延迟4

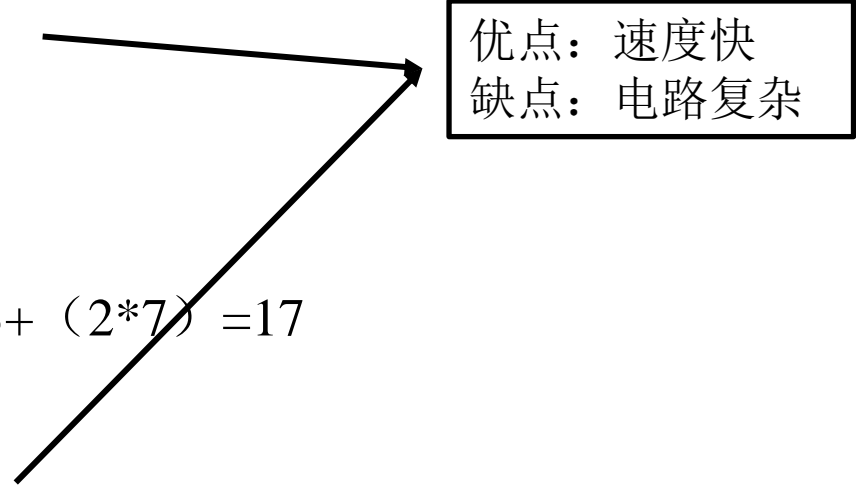
- **8位全加器**

- 1位全加器串联:

- 使用40个门, 最大门延迟 $3+(2*7)=17$

- 超前进位:

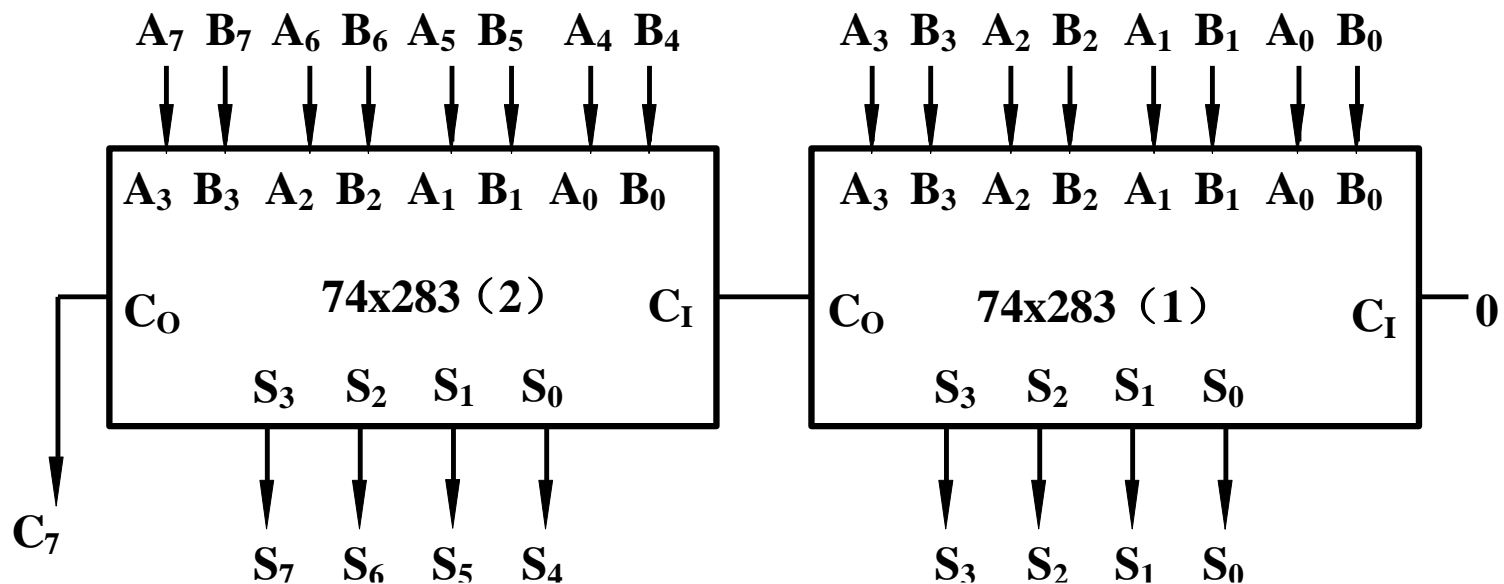
- 使用60个门, 最大门延迟4



优点: 速度快
缺点: 电路复杂

74x283应用 (1)

- 8位二进制数加法器
 - 片内超前进位，片间串行进位



74x283应用 (2)

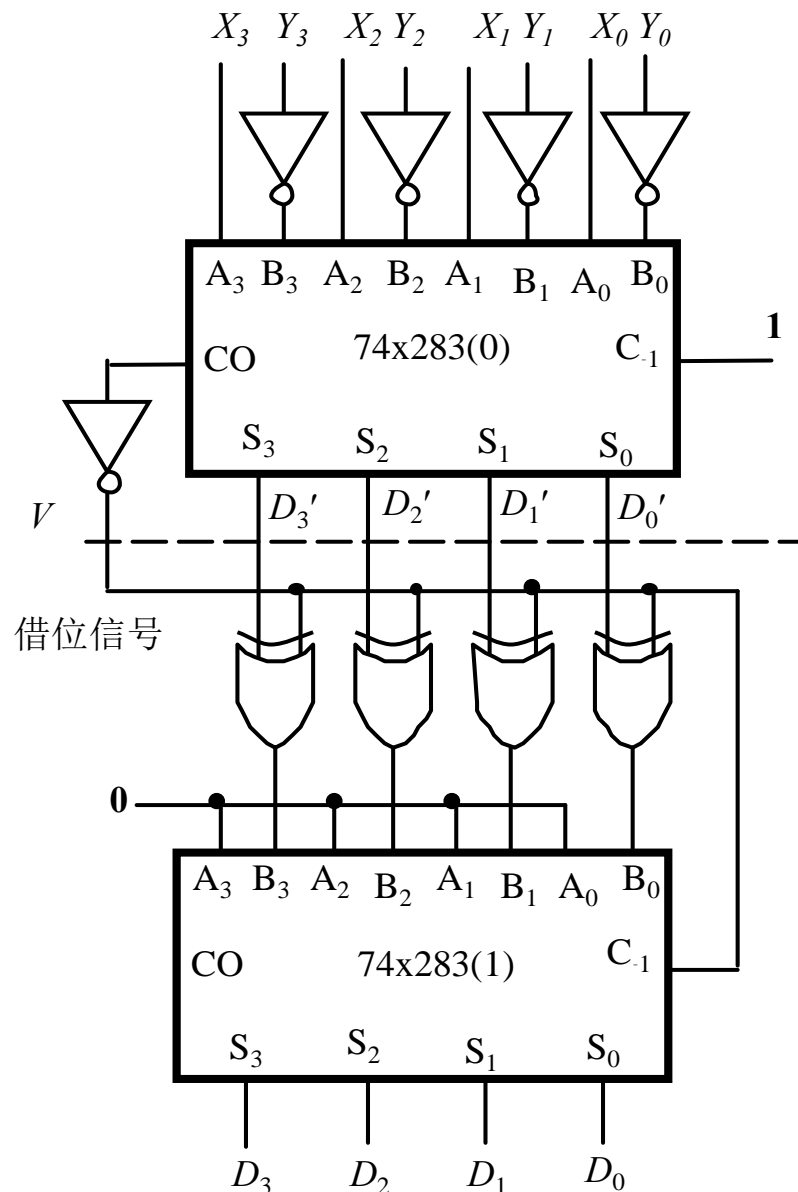
• 74x283(0)

- 按补码执行 $D' = X - Y$ 运算
- $X \geq Y$: 无借位, $V = 0$
- $X < Y$: 有借位, $V = 1$

• 74x283(1)

- $V = 0$: $D = 0 + D' = D$
- $V = 1$: $D = 0 - D' = -D$

即 $D = |X - Y|$



隐藏条件

X和Y都大于零或者都小于零时才成立

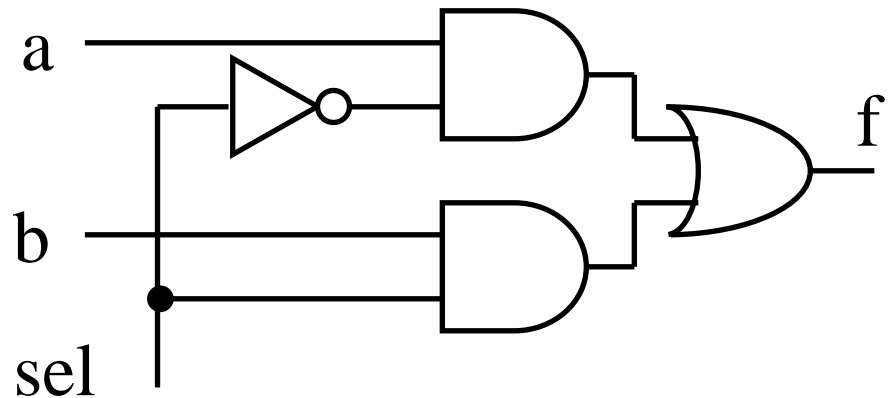
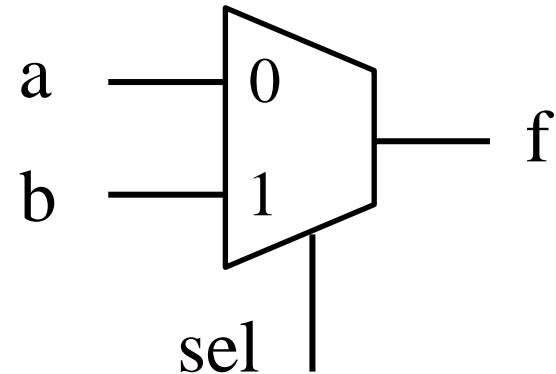
输入		对 74x283(0)						对 74x283(0)				
X	Y	X 补	Y 补	/Y 补	X 补+/Y 补+1	Co	D'	V=/Co	B= V*/D'+/V*D'	D=0+B+V	D 原	X-Y
1	2	0001	0010	1101	01111	0	1111	1	0000	0001	1	-1
	-2		1110	0001	00011	0	0011	1	1100	1101	-3	3
-1	2	1111	0010	1101	11101	1	1101	0	1101	1101	-3	-3
	-2		1110	0001	10001	1	0001	0	0001	0001	1	1
2	1	0010	0001	1110	10001	1	0001	0	0001	0001	1	1
	-1		1111	0000	00011	0	0011	1	1100	1101	-3	3
-2	1	1110	0001	1110	11101	1	1101	0	1101	1101	-3	-3
	-1		1111	0000	01111	0	1111	1	0000	0001	1	-1

示例— Mux with Assign

```
module mux2_1(  
    output f,  
    input a, b, sel );
```

```
    assign f = (a & ~sel) | (b & sel);
```

```
endmodule
```



HDL主要特征

- HDL语言既包含一些高级程序设计语言的结构形式，同时也兼顾描述硬件线路连接的具体构件
- 通过使用结构级或行为级描述可以在不同的抽象层次描述设计
 - 五个抽象层次：系统级、算法级、寄存器传输级、逻辑（门）级、电路（开关）级
- HDL语言是**并发**的，即具有在同一时刻执行多个任务的能力
- HDL语言有**时序**的概念

Verilog HDL与 C语言的比较

- 虽然Verilog的某些语法与C语言接近，但存在本质上的区别
 - Verilog是一种硬件语言，最终是为了产生实际的硬件电路或对硬件电路进行仿真
 - C语言是一种软件语言，是控制硬件来实现某些功能
- 利用Verilog编程时，要时刻记着：Verilog是硬件描述语言，要将其与**硬件电路对应**起来

C	Verilog
procedures	modules
variables	wires/regs
parameters	ports
control (if,case,?:)	control (if,case,?:)

Verilog HDL基本语法

- 空白符（间隔符）
 - 主要起分隔文本的作用，可以使文本错落有致，便于阅读与修改
 - 包括空格、制表符、换行符及换页符
- 注释符
 - 改善程序的可读性，在编译时不起作用
 - 多行注释符：以/*开始到*/结束
 - 单行注释符：以//开始到行尾结束

Verilog HDL基本语法 (续1)

- 关键字

- Verilog语言内部已经使用的词，例如，module、endmodule、input、output、wire、reg、and等
- 其中的字母都是小写，例如Input不是关键字

- 标识符

- 用于对象（如模块名、电路的输入与输出端口、变量等）命名
- 以字母或下划线 “_”开始，字母、下划线、数字等的组合
- 字母大小敏感，如in，IN是不同的标识符
- 不能与关键词相同

Verilog HDL基本语法 (续2)

- 逻辑值集合
 - 0: 低电平、逻辑0或“假”
 - 1: 高电平、逻辑1或“真”
 - x/X: 不确定的值（未知状态）
 - z/Z: 高阻态
- 常量与符号常量

常量

- 整数型

- 十进制数形式表示，例如，30、-2
- 带基数形式表示，格式为：

〈+/-〉 〈位宽〉' 〈基数符号〉 〈数值〉

基数符号：十进制 D/d，二进制 B/b，八进制 O/o，十六进制 H/h

例如， - 8' d101、5' o37、8' HeD, 8' b1001_001x

- 实数型常量

- 十进制记数法，例如，0.1、2.0、5.67
- 科学记数法，例如，23.1e2、5E-4

符号常量

- 用参数定义语句定义一个标识符来代表一个常量
 - 常用来定义变量的位宽及延时等
- 定义格式

parameter 参数名1=常量表达式1, 参数名2=常量表达式2,;

例如: parameter BIT=1, BYTE=8, PI=3.14;

变量数据类型

- **线网(net)型**：表示元件之间的物理连线
 - 输出值紧随输入值的变化而变化
 - 最常用类型是wire
- **寄存器(register)型**：表示抽象存储元件
 - 在赋新值以前保持原值
 - 只能在initial或always语句中被赋值
 - 最常用类型是reg

- **定义格式**

wire/reg [MSB:LSB] 变量名1, ..., 变量名n;

例如: wire a, b; reg[3:0] state;

Verilog HDL程序基本结构

- 由实现特定功能的模块构成

module 模块名 (端口名1, 端口名2, ...);

端口类型说明(input, output, inout);

参数定义(可选);

数据类型定义(wire, reg等);

} 说明部分

实例化低层模块和基本门级元件;

连续赋值语句 (assign) ;

过程块结构 (initial和always)

行为描述语句;

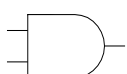




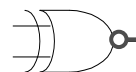
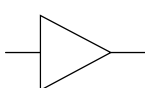
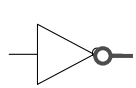
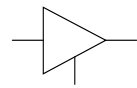
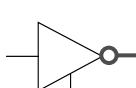
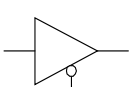
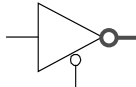
} 功能描述部分
顺序是任意的

endmodule

VerilogHDL描述组合逻辑电路

- 组合逻辑电路的门级描述
 - 使用内置的基本门级元件描述
- 组合逻辑电路的数据流描述
 - 使用连续赋值assign语句描述
- 组合逻辑电路的行为级描述
 - 使用always结构描述

基本门级元件

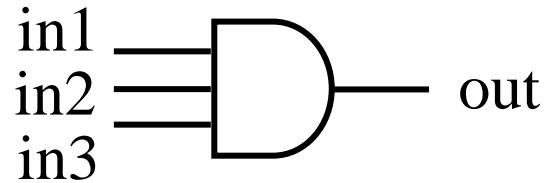
元件符号	功能说明	元件符号	功能说明
and 	多输入端与门	nand 	多输入端与非门
or 	多输入端或门	nor 	多输入端或非门
xor 	多输入端异或门	xnor 	多输入端异或非门
buf 	多输出端缓冲器	not 	多输出端反相器
bufif1 	高电平有效三态缓冲器	notif1 	高电平有效的三态反相器
bufif0 	低电平有效三态缓冲器	notif0 	低电平有效的三态反相器

多输入门和多输出门

- 多输入门：允许多个输入，但只有一个输出

– and, or, xor, nand, nor, xnor

实例名可忽略

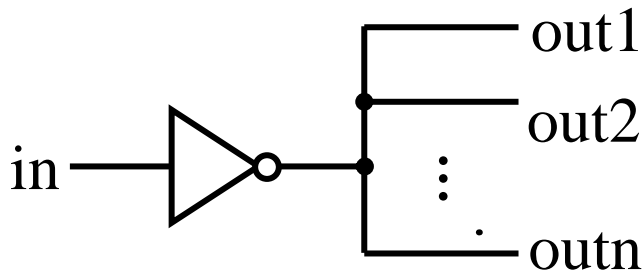


and A1(out, in1, in2, in3);

- 多输出门：允许有多个输出，但只有一个输入

– not, buf

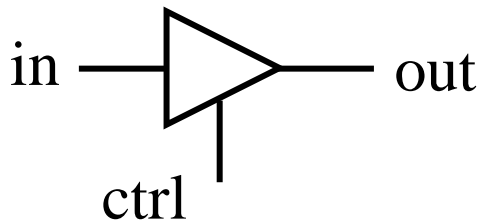
实例名可忽略



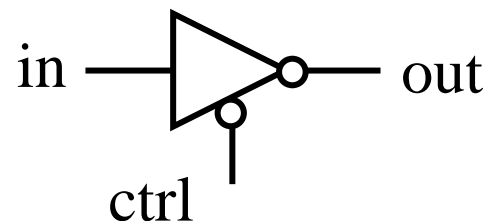
not B1(out1, out2, ..., in);

三态门

- 一个输出、一个数据输入和一个控制输入
 - notif0, notif1, bufif0, bufif1



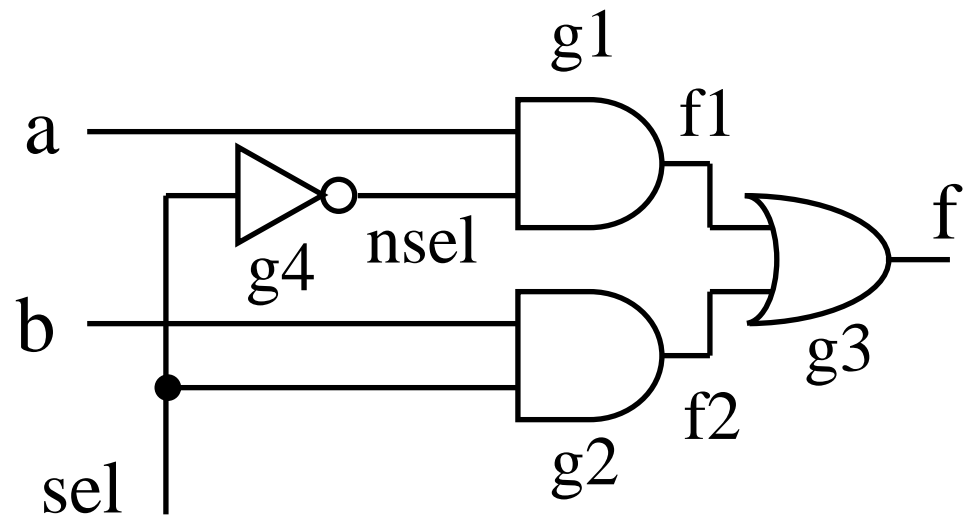
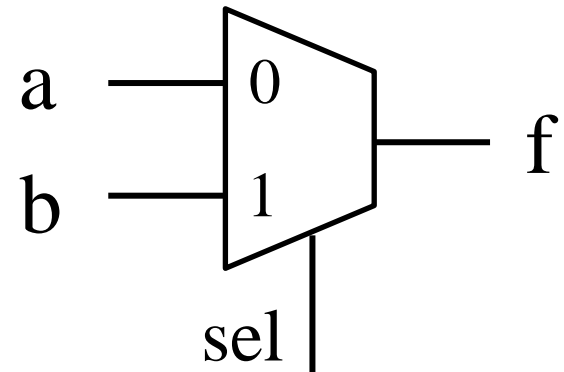
bufif1 B1(out, in, ctrl);



notif0 N1(out, in, ctrl);

示例— Mux with Primitives

```
module mux2_1( f, a, b, sel );  
    output f;  
    input a, b, sel;  
  
    and g1(f1, a, nsel),  
        g2(f2, b, sel);  
    or  g3(f, f1, f2);  
    not g4(nsel, sel);  
  
endmodule
```



赋值语句

- 连续赋值语句

assign 变量名= 赋值表达式

- 只能对线网型变量进行赋值，不能对寄存器型变量进行赋值
- 仅用于描述组合逻辑

- 过程赋值语句

变量名= 赋值表达式

- 只能对寄存器数据类型的变量赋值
- 在always和initial语句内的赋值
- 可用于描述组合和时序逻辑

Verilog HDL运算符

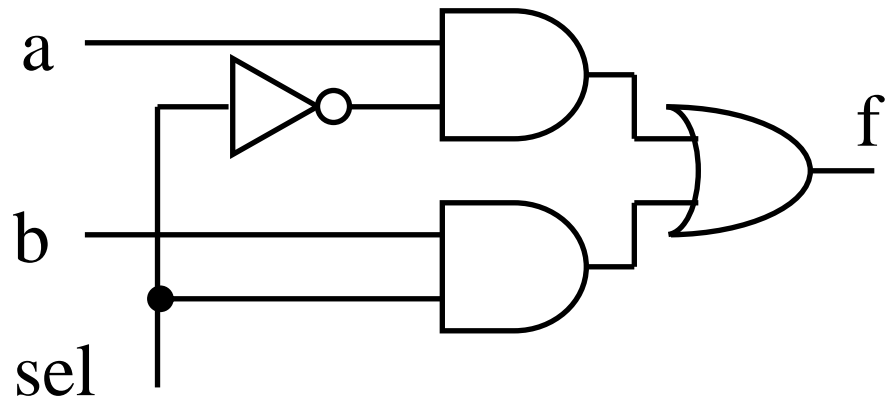
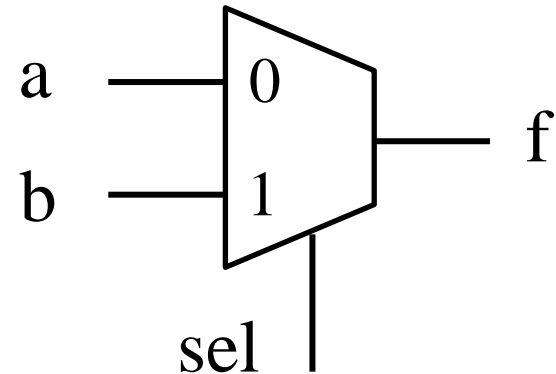
类型	符号	功能说明	类型	符号	功能说明
算术运算符	+ - * / %	二进制加 二进制减 二进制乘 二进制除 求模	关系运算符	> < >= <= == !=	大于 小于 大于或等于 小于或等于 等于 不等于
位运算符	~ & ^ ^~ 或 ~^	按位取反 按位与 按位或 按位异或 按位同或	缩位运算符	& ~& ~ ^ ^~ 或 ~^	缩位与 缩位与非 缩位或 缩位或非 缩位异或 缩位同或
逻辑运算符	! && 	逻辑非 逻辑与 逻辑或	移位运算符	>> <<	右移 左移

示例— Mux with Assign

```
module mux2_1(  
    output f,  
    input a, b, sel );
```

```
    assign f = (a & ~sel) | (b & sel);
```

```
endmodule
```



if条件语句

- 表达式一般为逻辑表达式或关系表达式
 - 对表达式的值进行判断，若为0，x，z，按假处理；若为1，则按真处理，执行指定语句
- if和else后可包含单个或多个语句，多句时用begin-end块语句括起来
- if语句嵌套使用时，注意if与else的配对关系

```
if (表达式) 语句1;  
if (表达式) 语句1;  
    else 语句2;  
if (表达式1) 语句1;  
    else if (表达式2) 语句2;  
    else if (表达式3) 语句3;  
        .....  
    else if (表达式n) 语句n;  
    else 语句n+1;
```

Case条件语句

case (敏感表达式)

值1: 语句1;

值2: 语句2;

.....

值n: 语句n;

default: 语句n+1;

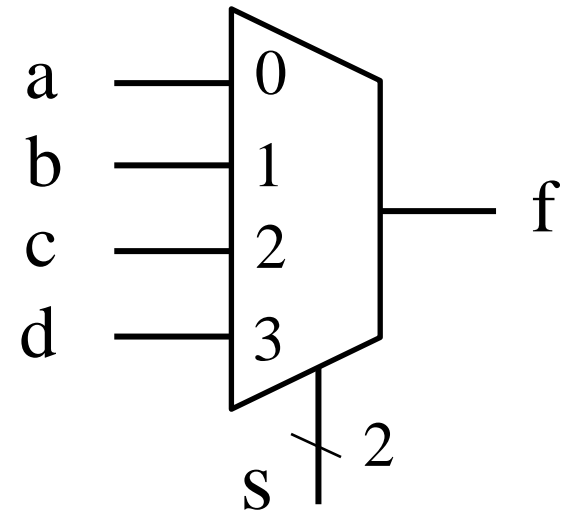
endcase

条件语句使用要点

- 描述组合电路时，应注意列出所有条件分支，否则编译器认为条件不满足时，会引进一个记忆单元（锁存器）来保持原值，从而产生时序电路而非组合电路
- 由于每个变量有4种取值，为包含所有分支，可在if语句后加上 **else**；在 **case**语句后加上 **default**

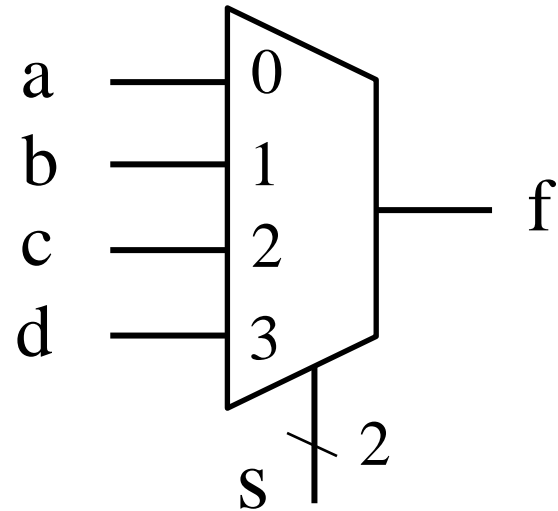
示例— Mux with Always (If)

```
module mux4_1(  
    output reg f,  
    input a, b, c, d,  
    input [1:0] s);  
  
    always @(a, b, c, d, s)  
        if (s == 2'b00) f = a;  
        else if (s == 2'b01) f = b;  
        else if (s == 2'b10) f = c;  
        else f = d;  
endmodule
```



示例— Mux with Always(Case)

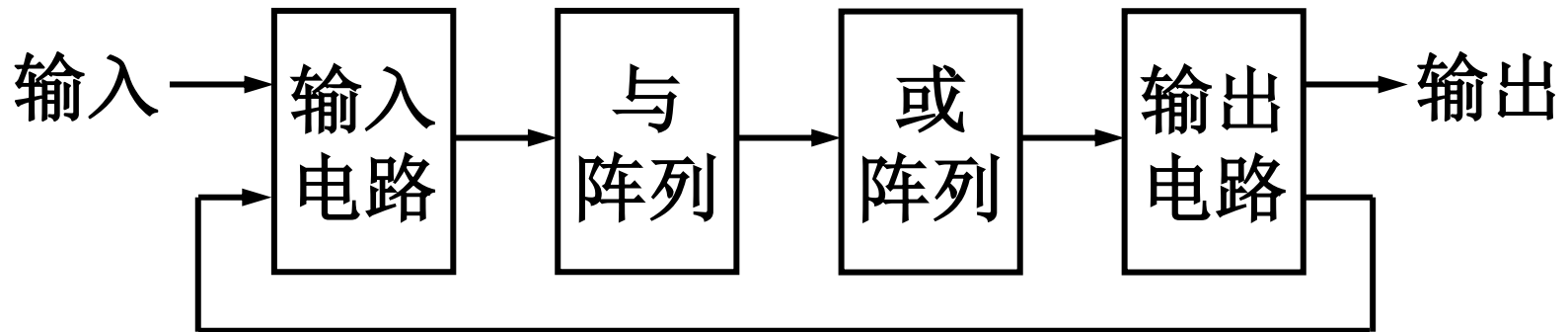
```
module mux4_1(  
    input a, b, c, d,  
    input [1:0] s,  
    output reg f );  
    always @(a, b, c, d, s)  
        case (s)  
            2'b00: f = a;  
            2'b01: f = b;  
            2'b10: f = c;  
            default: f = d;  
        endcase  
endmodule
```



可编程逻辑器件


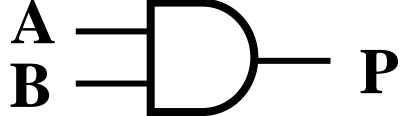
- 可编程逻辑器件（**Programmable Logic Device**, 简称**PLD**）是一种可以由用户定义和设置逻辑功能的器件
 - 与中小规模通用逻辑器件相比，具有集成度高、速度快、功耗低、可靠性高等优点
 - 与其他专用集成电路相比，具有产品开发周期短、用户投资风险小、小批量生产成本低等优势
- 按集成度**PLD**可分为
 - 低密度PLD: PROM、PLA、PAL、GAL
 - 高密度PLD: CPLD、FPGA

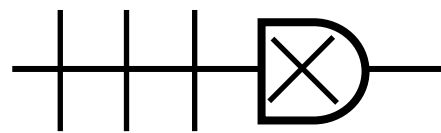
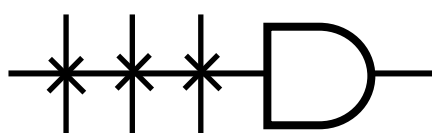
PLD基本结构

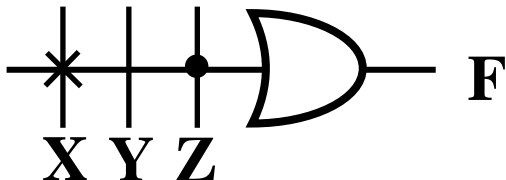
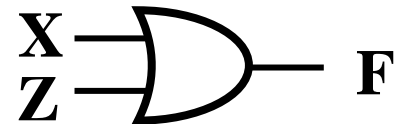


- 输入电路
 - 提供互补输入变量 (原变量和反变量)
- 与阵列
 - 产生逻辑函数所需的乘积项
- 或阵列
 - 选择乘积项，形成与或式，实现逻辑函数
- 输出电路
 - 提供不同的输出方式

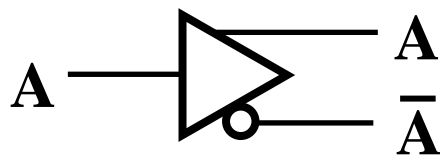
PLD中逻辑符号表示

• 与门:  = 

 = 

• 或门:  = 

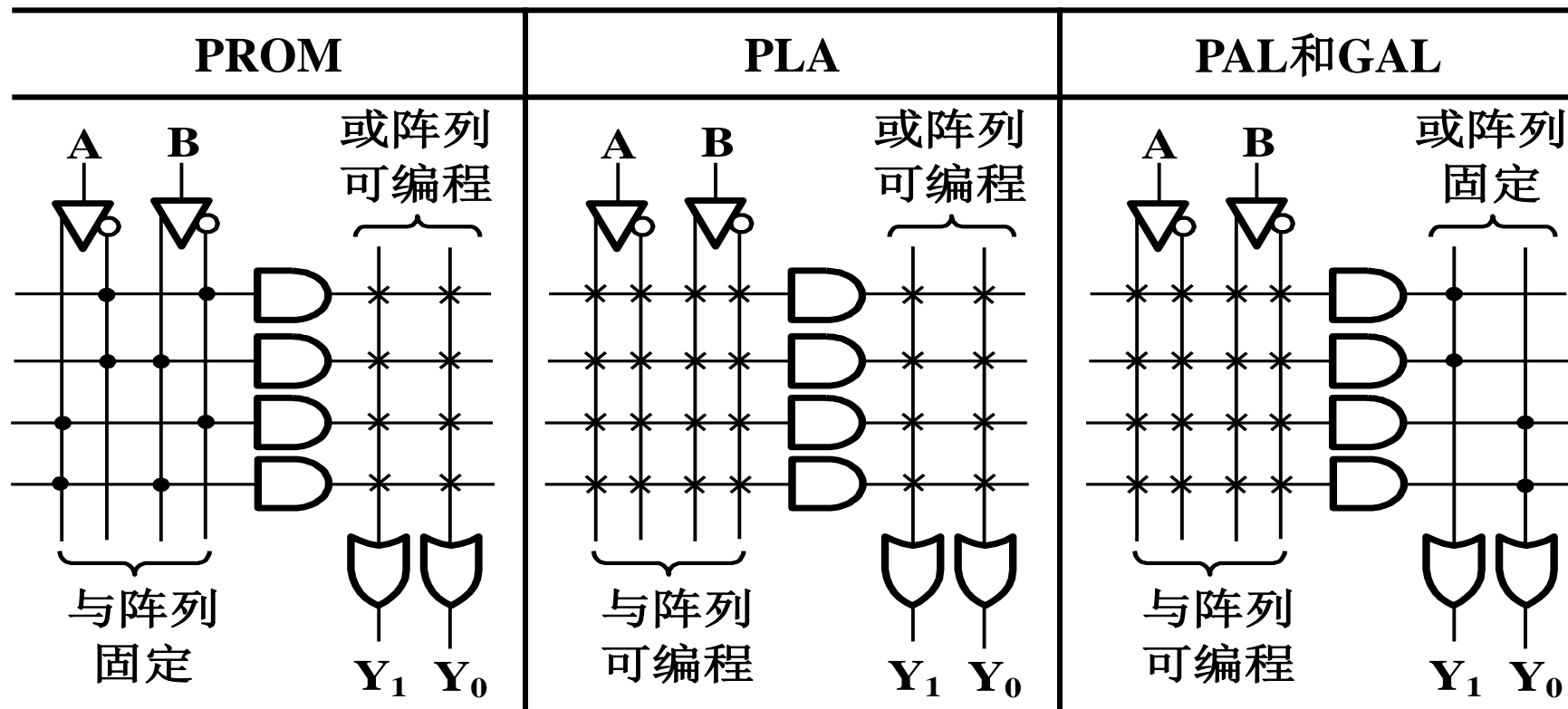
• 互补输入缓冲器:



标记	连接方式
●	固定连接
×	编程连接
空	未连接

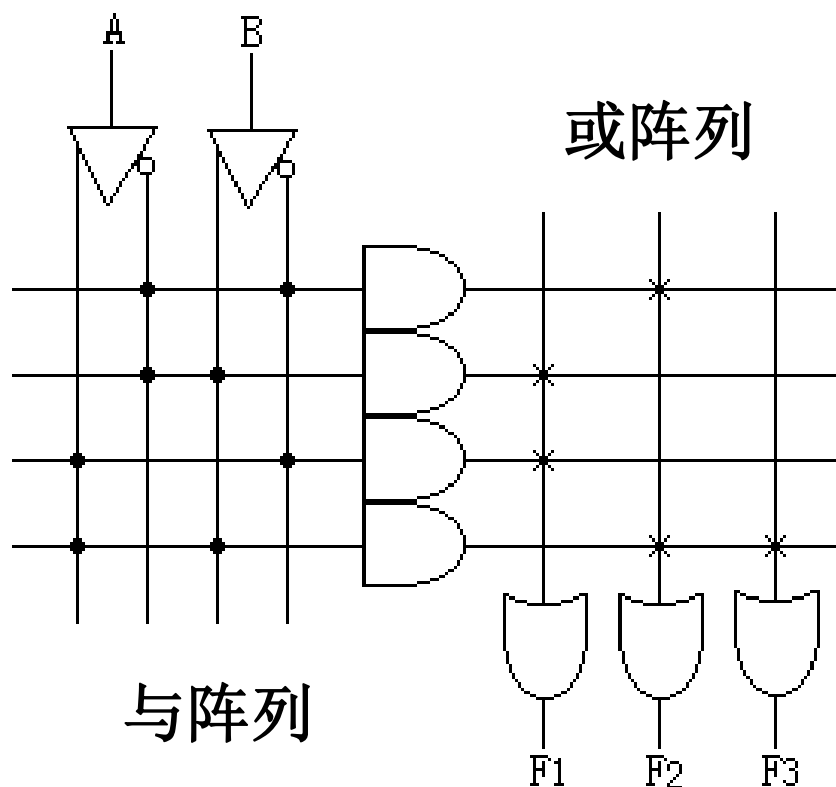
低密度PLD的与、或阵列结构

- PROM: Programmable Read Only Memory, 可编程只读存储器
- PLA: Programmable Logic Array, 可编程逻辑阵列
- PAL: Programmable Array Logic, 可编程阵列逻辑
- GAL: Gate Array Logic, 门阵列逻辑

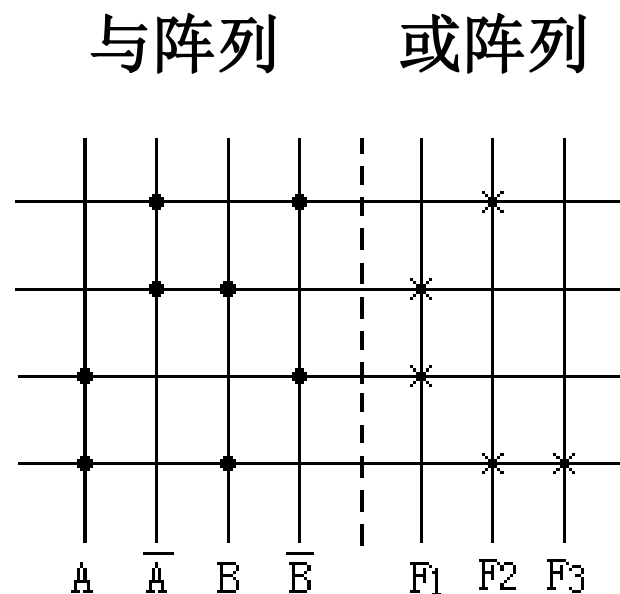


示例—PROM实现组合逻辑

$$F1 = \bar{A} \cdot B + A \cdot \bar{B} \quad F2 = \bar{A} \cdot \bar{B} + A \cdot B \quad F3 = A \cdot B$$

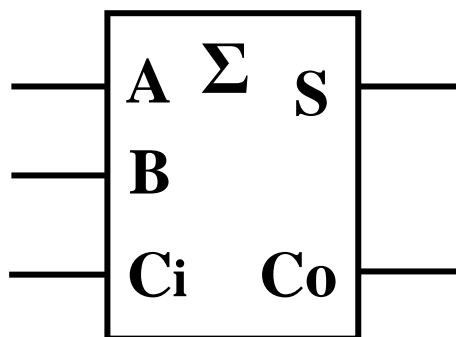


逻辑图



简化逻辑图

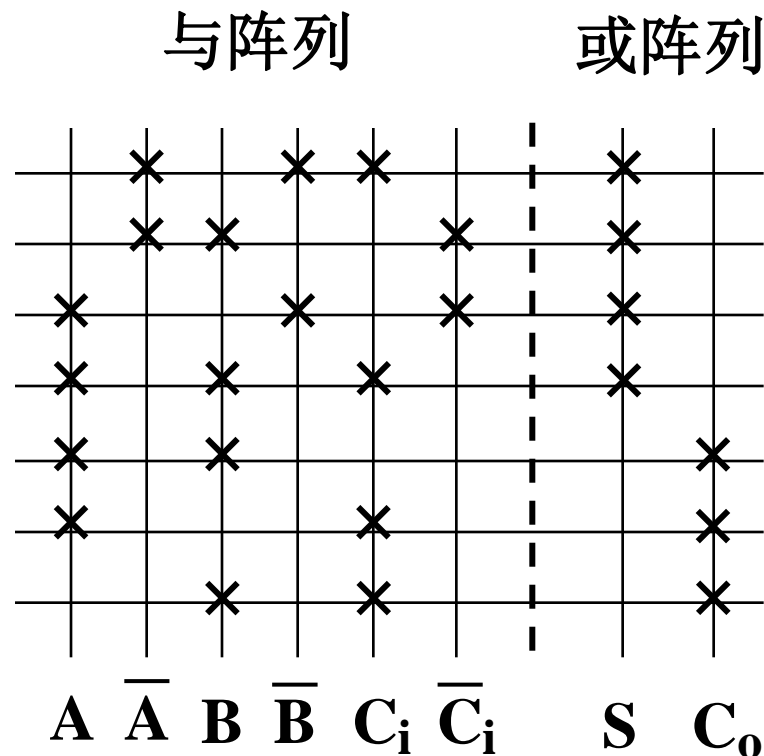
示例—PLA实现一位全加器



$$C_o = AB + (A \oplus B)C_i$$

$$= AB + AC_i + BC_i$$

$$S = A \oplus B \oplus C_i = \overline{A}\overline{B}C_i + \overline{A}B\overline{C_i} + A\overline{B}\overline{C_i} + ABC_i$$



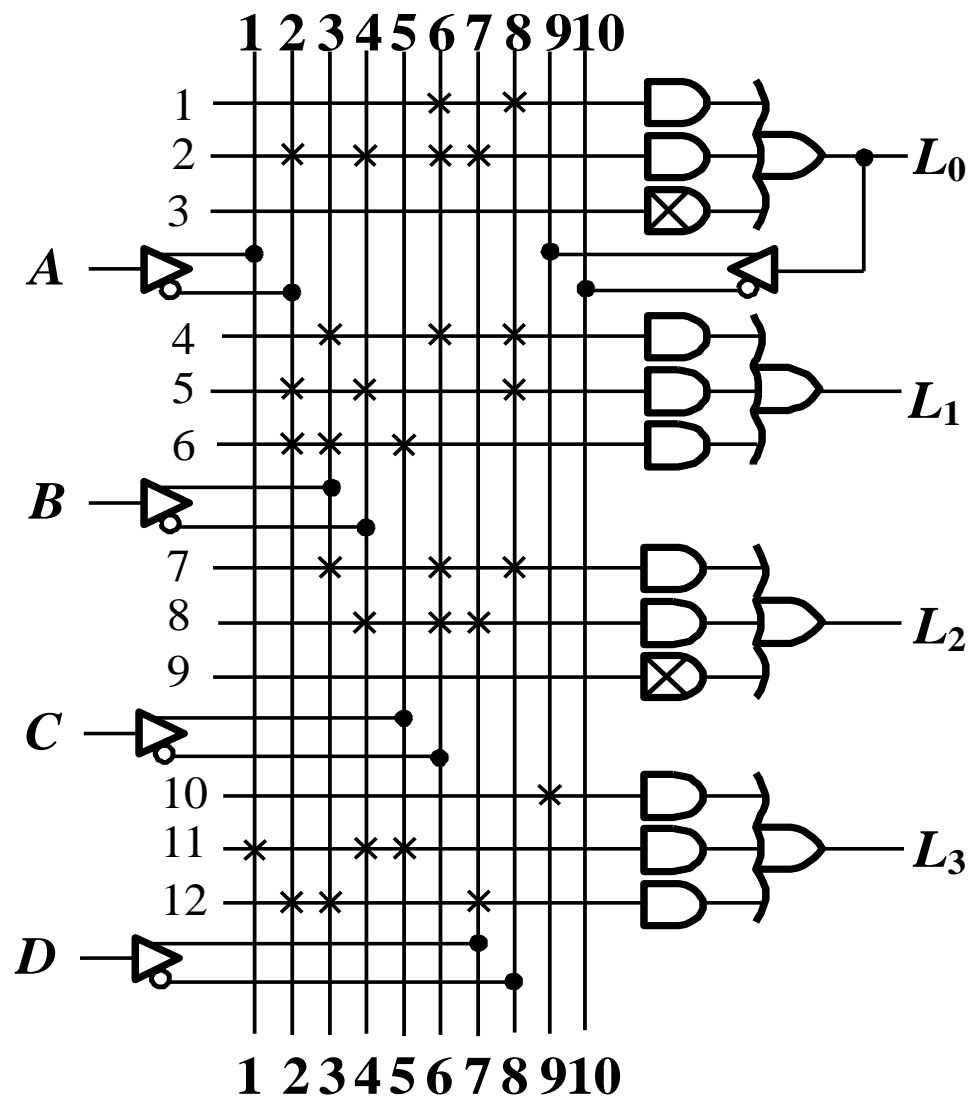
示例—PAL实现组合逻辑

$$L_0 = \overline{C}\overline{D} + \overline{A}B\overline{C}D$$

$$L_1 = \overline{B}\overline{C}D + \overline{A}B\overline{C}D + \overline{A}BC$$

$$L_2 = \overline{B}\overline{C}D + \overline{B}C\overline{D}$$

$$\begin{aligned} L_3 &= L_0 + \overline{A}B\overline{C} + \overline{A}BD \\ &= \overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C} + \overline{A}BD \end{aligned}$$

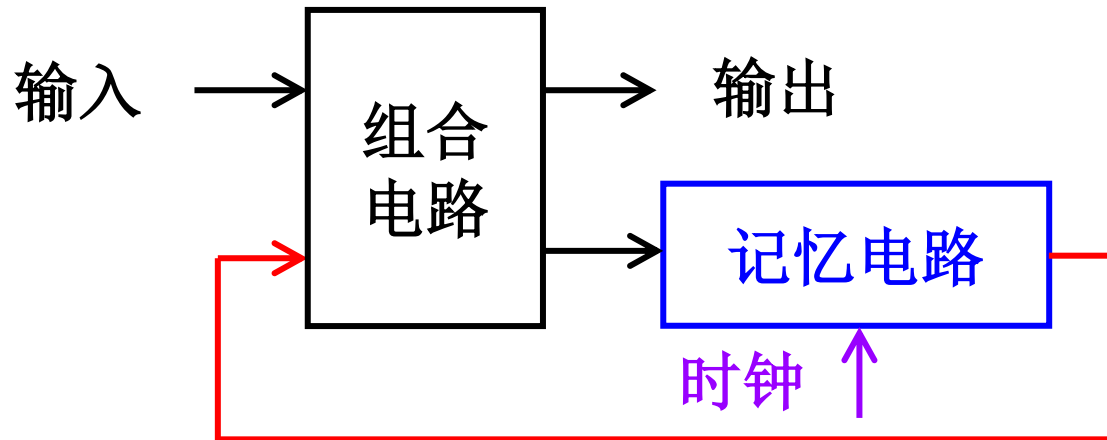


内容提纲

- 时序逻辑电路特点
- 锁存器
 - 基本SR锁存器
 - 门控SR锁存器
 - D锁存器

时序逻辑电路

- 任意时刻电路的输出不仅与该时刻的输入有关，还与之前的输入有关
- 时序电路结构特点：含有记忆电路和反馈路径



- 记忆单元电路：锁存器和触发器

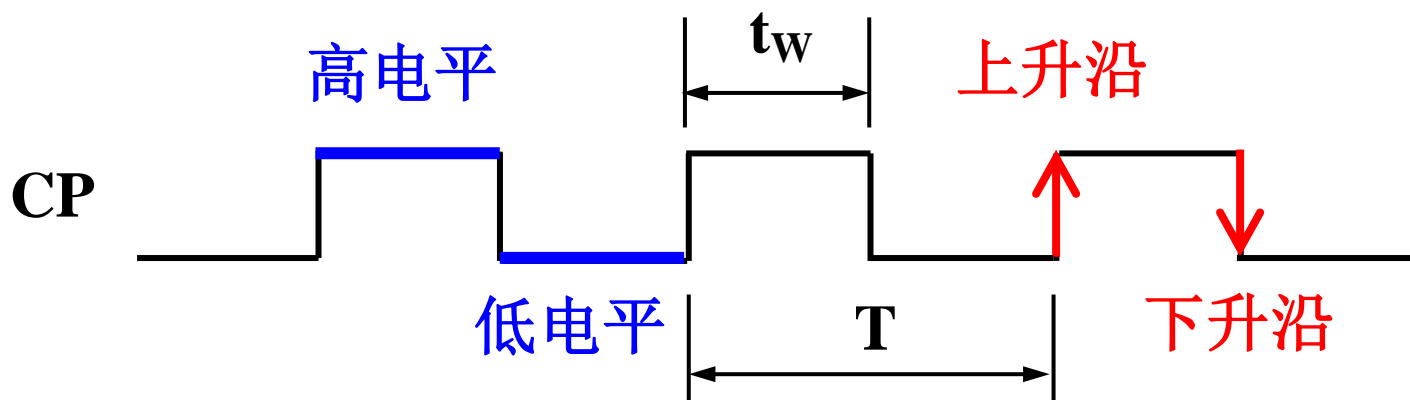
时钟信号

- 周期性的脉冲信号，也称时钟脉冲(CP)，简称时钟，用于控制记忆单元状态更新的时机

– 参数：周期 T ，频率 f ，脉冲宽度 t_w ，占空比 q

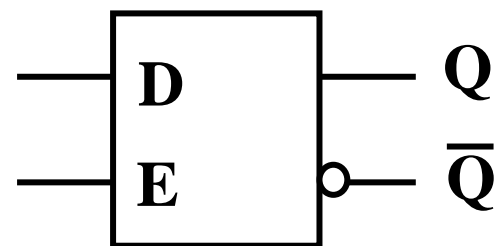
$$f = \frac{1}{T} \qquad q = \frac{t_w}{T} \times 100\%$$

– 有效时机：高电平、低电平、上升沿或下降沿

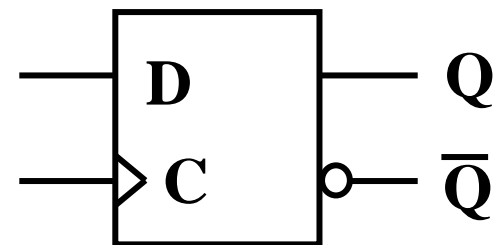


锁存器和触发器

- 具有记忆功能的逻辑单元电路，是构成时序电路的基石，又称为记忆单元、存储单元或状态单元
- 两者共同点：记忆功能
 - 具有两个能自行保持的稳定状态，可用来存储一位二值信息
 - 在输入信号作用下可更新状态
- 两者不同点：更新时机
 - 锁存器对时钟的电平敏感，在有效电平期间更新状态
 - 触发器对时钟的边沿敏感，在有效边沿瞬间更新状态



D锁存器



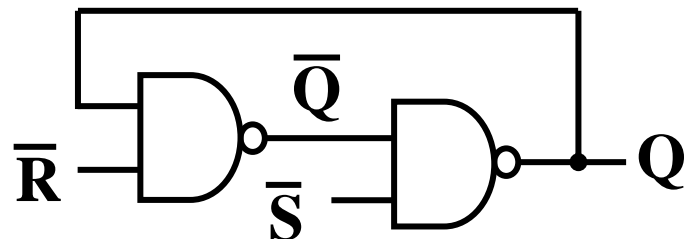
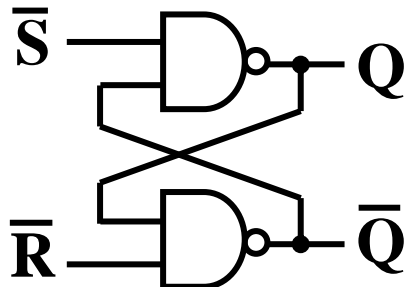
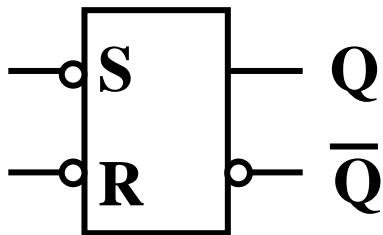
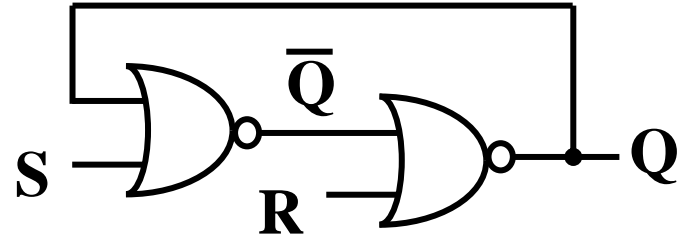
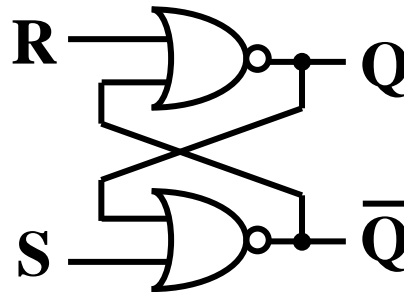
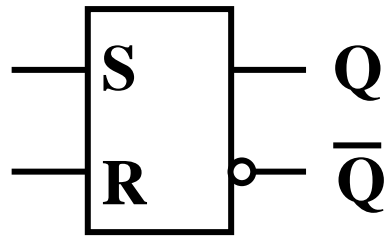
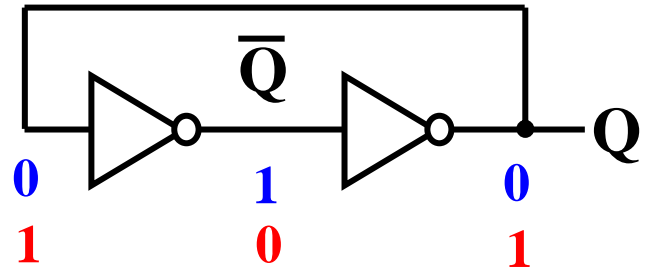
D触发器

基本SR锁存器(Latch)

- 电路结构

- 利用反馈实现记忆
- 利用R、S更新状态

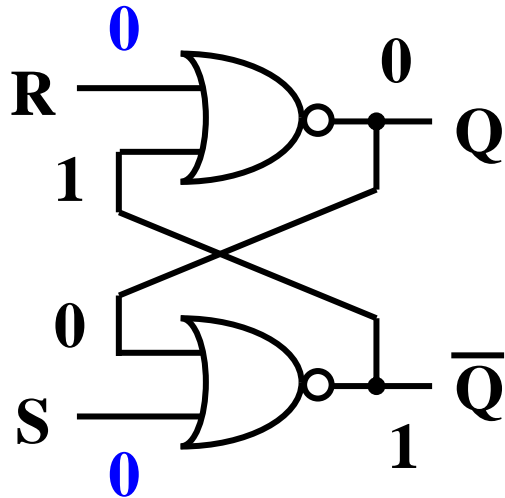
- 逻辑符号



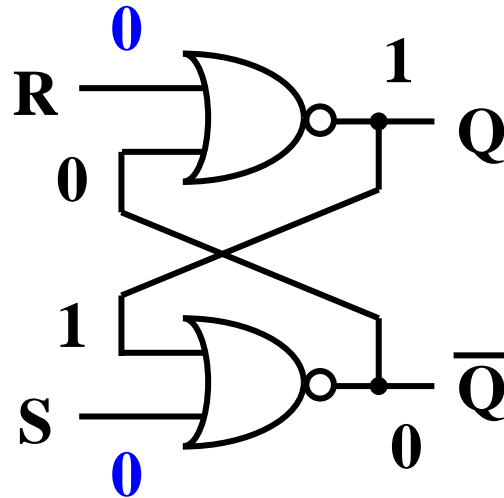
基本SR锁存器工作原理

- 现态：R、S作用前Q端的状态，记为 Q^n
- 次态：R、S作用后Q端的状态，记为 Q^{n+1}

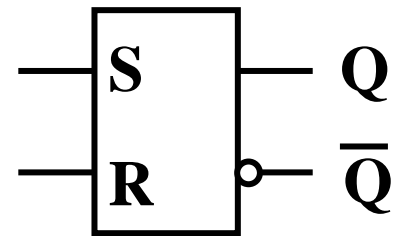
当 $R=0$ ， $S=0$ 时， $Q^{n+1} = Q^n$ (状态不变)



$$Q^n = 0, Q^{n+1} = 0$$



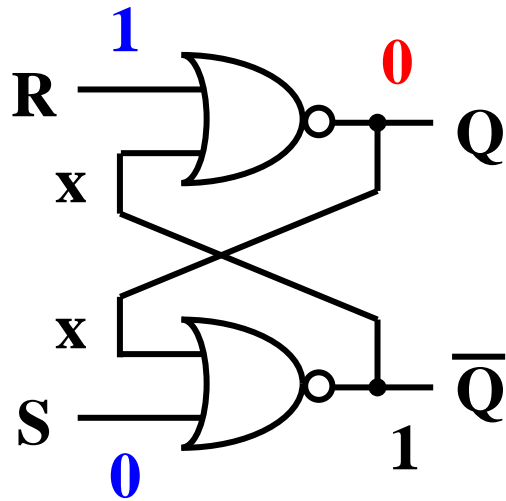
$$Q^n = 1, Q^{n+1} = 1$$



基本SR锁存器工作原理 (续1)

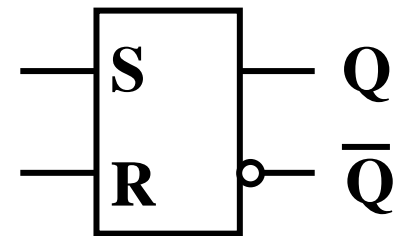
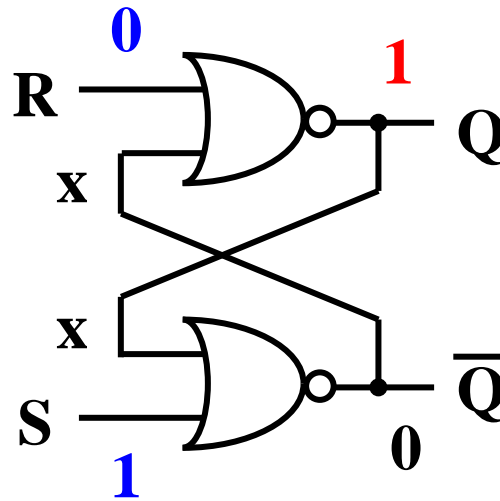
当 $R=1$, $S=0$ 时

$Q^{n+1} = 0$ (清0)



当 $R=0$, $S=1$ 时

$Q^{n+1} = 1$ (置1)

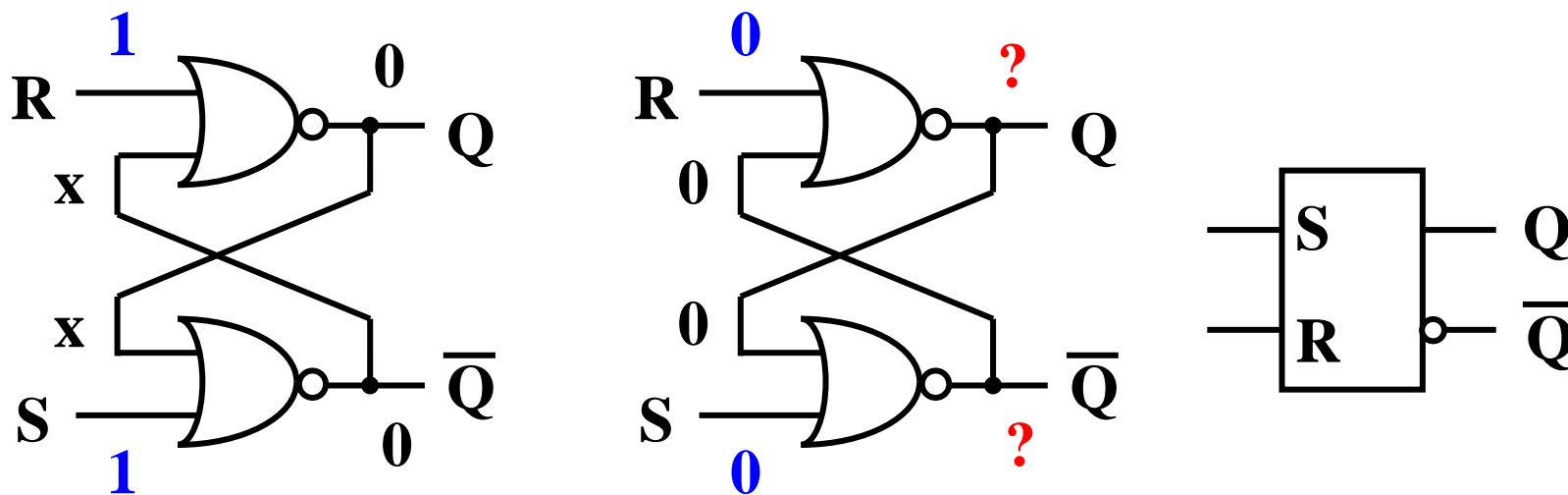


当 R 、 S 都恢复到0后，锁存器新的状态保持不变

基本SR锁存器工作原理 (续2)

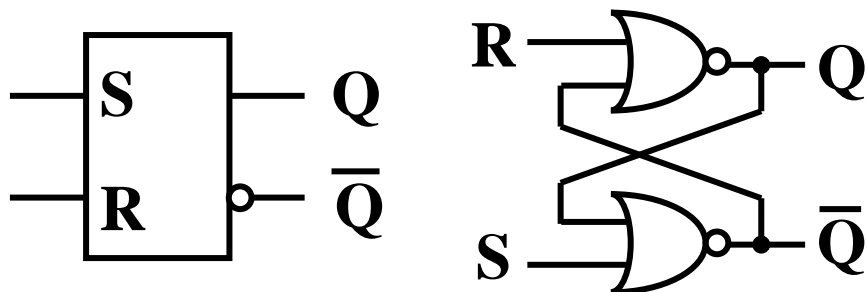
当 $R=1$, $S=1$ 时, $Q^{n+1} = \overline{Q}^{n+1} = 0$

当 R 、 S 同时回到0后, 锁存器最终状态不能确定



在实际应用中, 应避免 R 和 S 同时为1, 即要求满足约束条件: $SR = 0$

基本SR锁存器特性

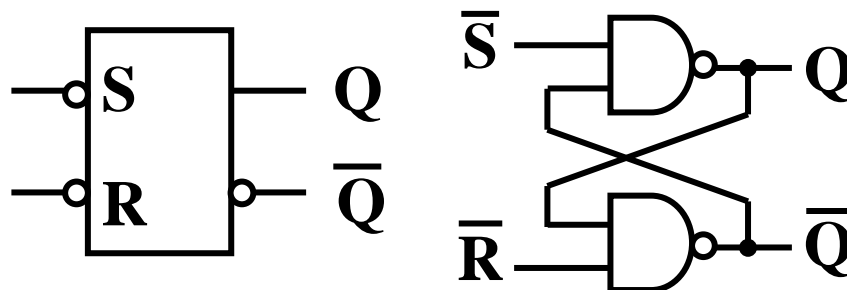


特性表

S	R	Q^{n+1}	说明
0	0	Q^n	保持
0	1	0	清0
1	0	1	置1
1	1	0*	禁止

特性表

\bar{S}	\bar{R}	Q^{n+1}	说明
0	0	1*	禁止
0	1	1	置1
1	0	0	清0
1	1	Q^n	保持



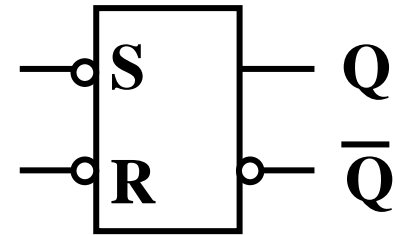
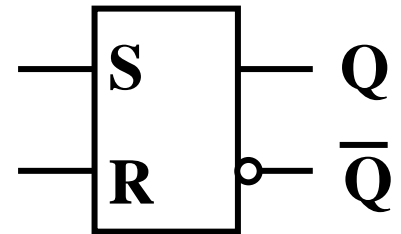
基本SR锁存器特性(续)

- 两个激励输入端

- R: Reset, 复位/置0/清0, 有效时, $Q=0$, $\overline{Q}=1$
- S: Set, 置位/置1, 有效时, $Q=1$, $\overline{Q}=0$
- 存在约束条件, 要求R和S不能同时有效
- 对于或非门实现的SR锁存器, 高电平有效, $SR = 0$
- 对于与非门实现的SR锁存器, 低电平有效的, $\overline{R} + \overline{S} = 1$

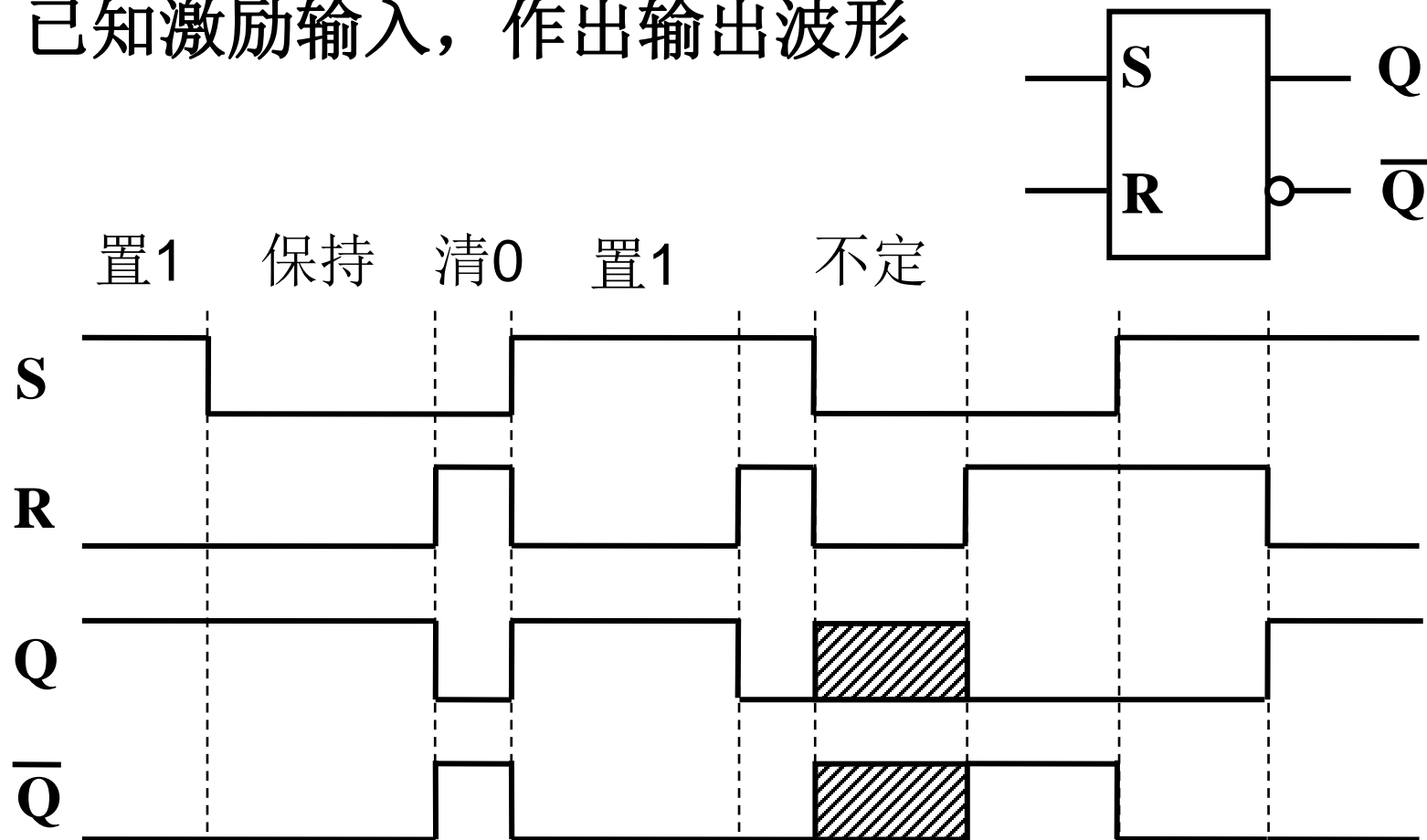
- 锁存器状态更新不受时间控制

- 激励输入可以随时更新状态



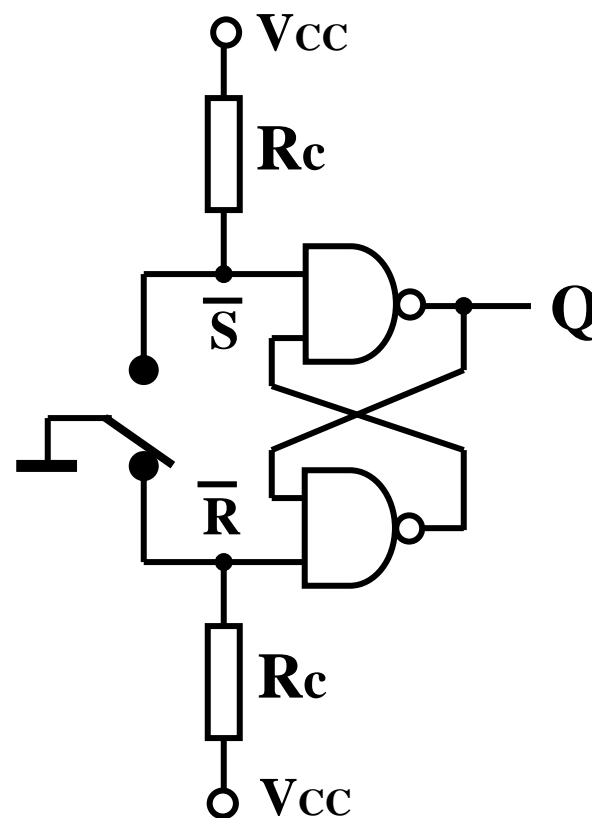
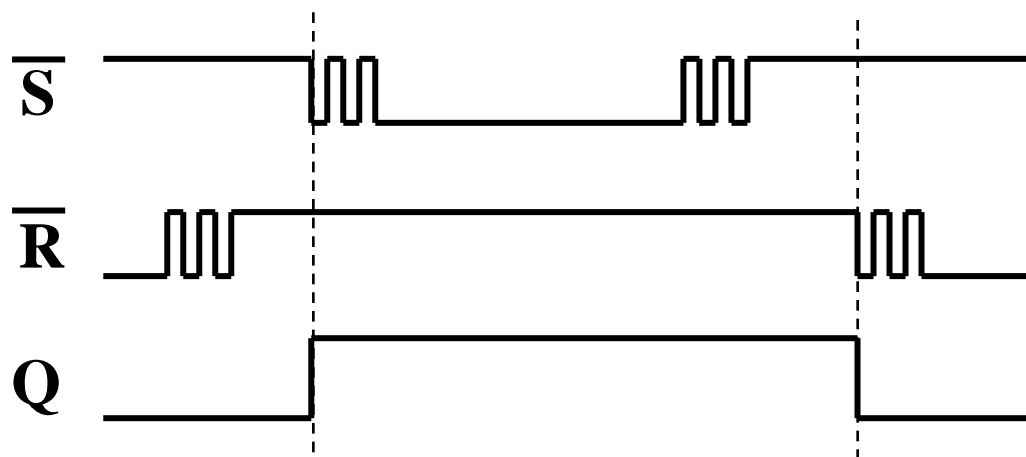
示例—基本SR锁存器波形图

- 已知激励输入，作出输出波形



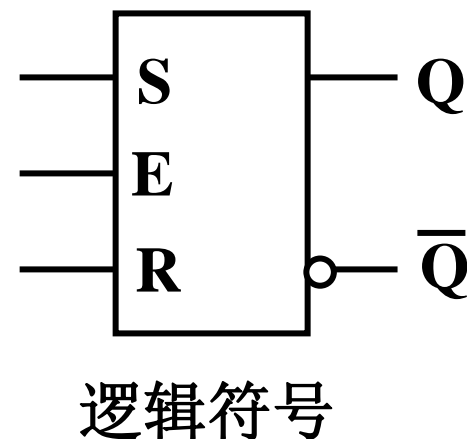
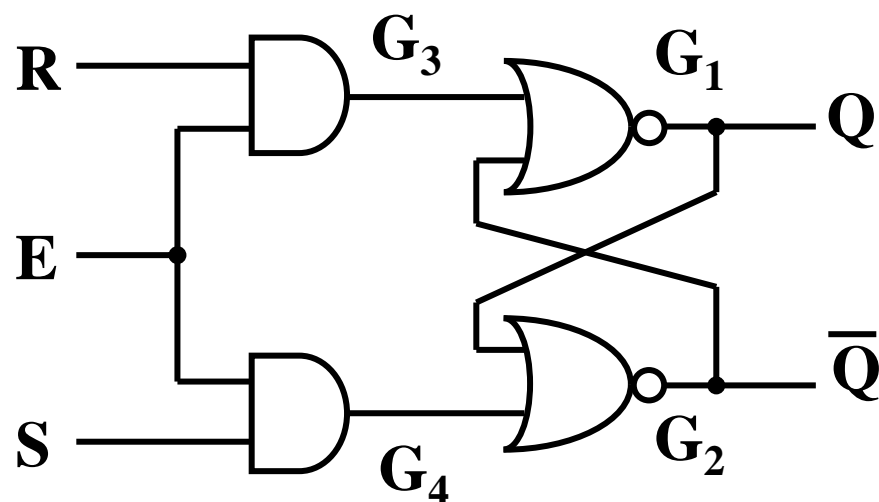
示例—基本SR锁存器应用

- 开关去抖动电路
 - 运用基本SR锁存器，消除因机械开关触点抖动所引起的干扰脉冲的输出



门控SR锁存器

- 增加门控(Gated)信号，使得激励输入信号更新锁存器状态的时机可以受控

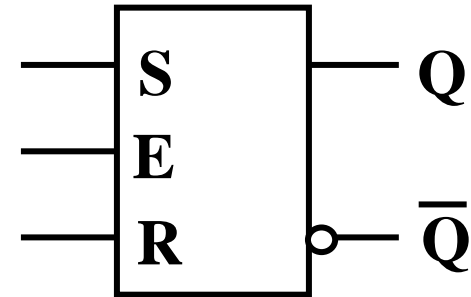


当E=0时：锁存器状态保持不变，不受R、S影响
当E=1时：与基本SR锁存器功能相同

门控SR锁存器特性

特性表

E	S	R	Q^{n+1}	说明
0	x	x	Q^n	保持
1	0	0	Q^n	保持
1	0	1	0	清0
1	1	0	1	置1
1	1	1	x	禁止

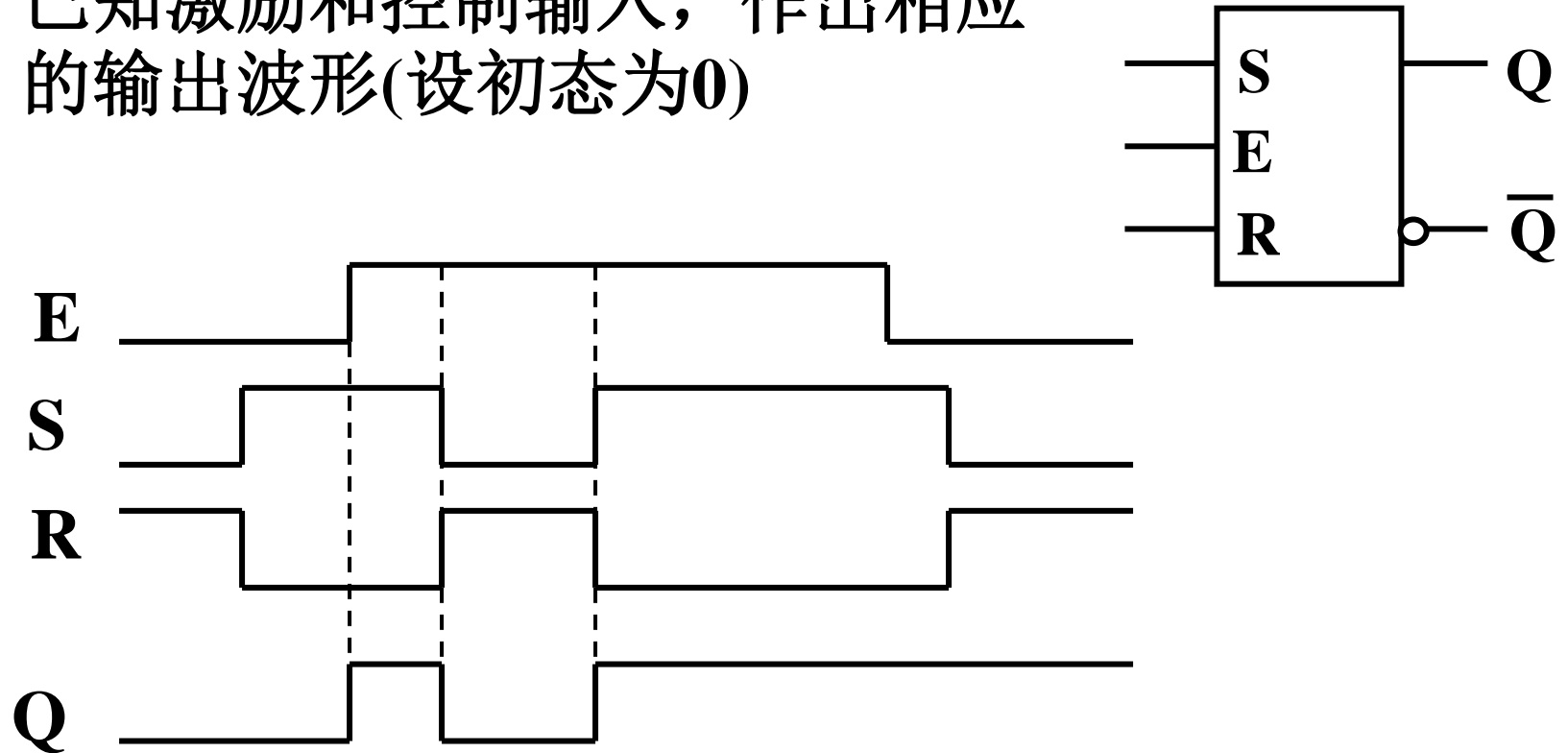


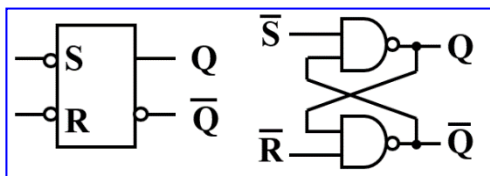
- 锁存器状态可以随激励输入变化发生多次翻转
 - 在E有效(高电平)期间，R和S的变化将引起输出状态的变化

- 激励输入约束条件($SR = 0$)仍然存在

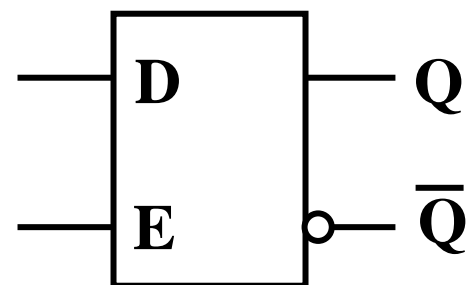
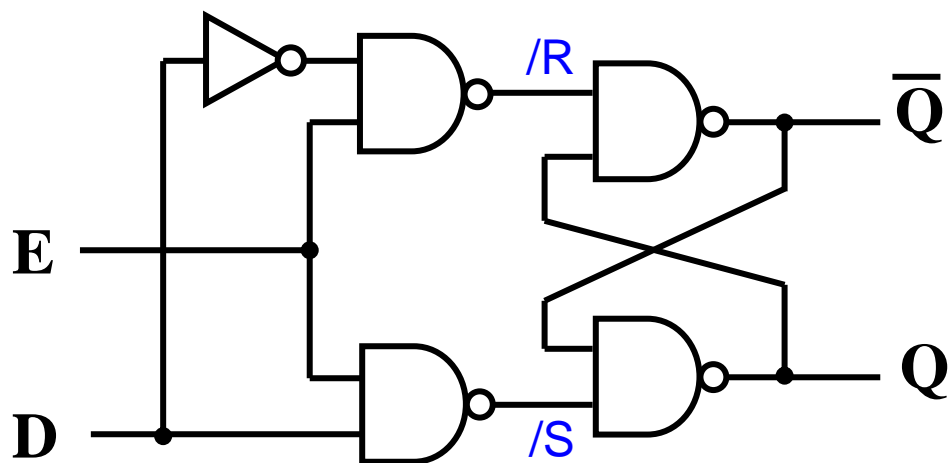
示例—门控SR锁存器波形图

- 已知激励和控制输入，作出相应的输出波形(设初态为0)





D锁存器



逻辑符号

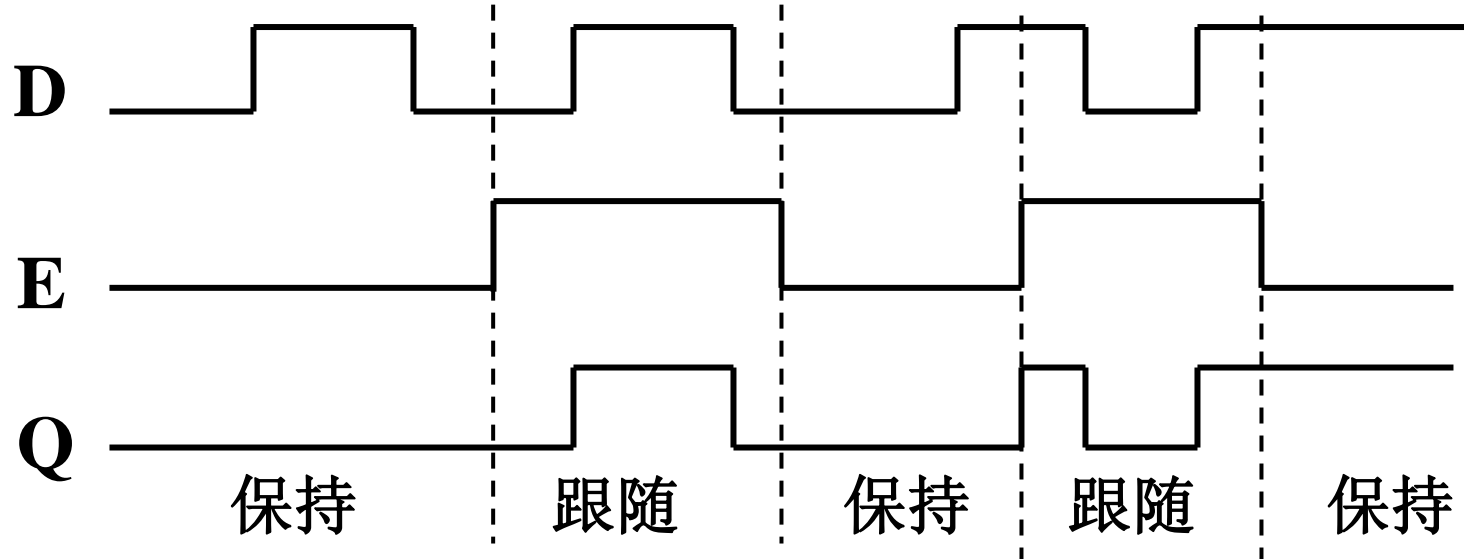
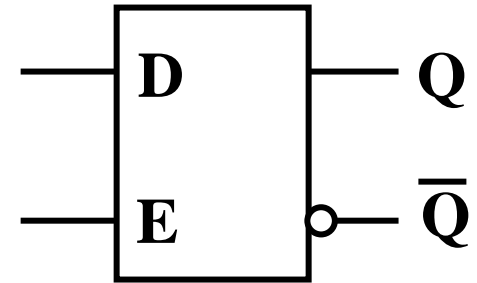
特性表

E	D	Q^{n+1}	说明
0	x	Q^n	保持
1	x	D	跟随

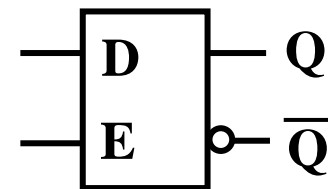
- 在E有效(电平有效)期间，D的变化将引起锁存器状态多次翻转

示例—D锁存器波形图

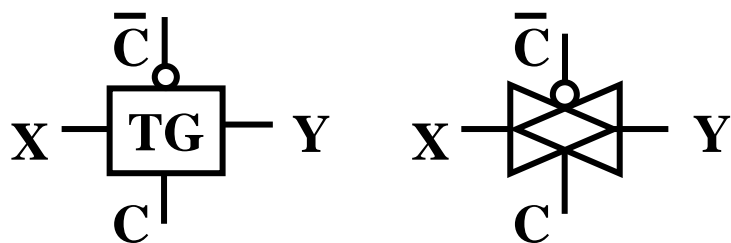
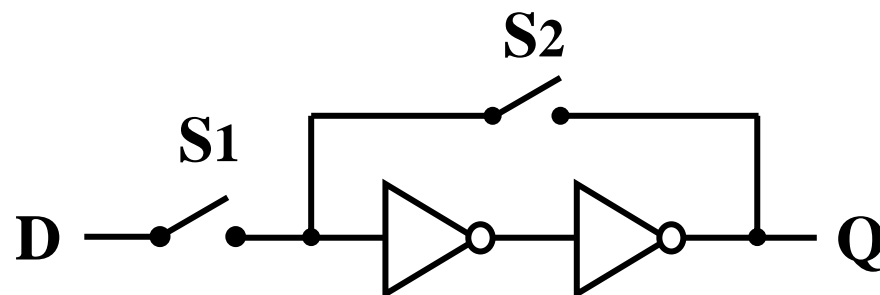
- 已知激励和控制输入，作出相应输出波形（设初态为0）



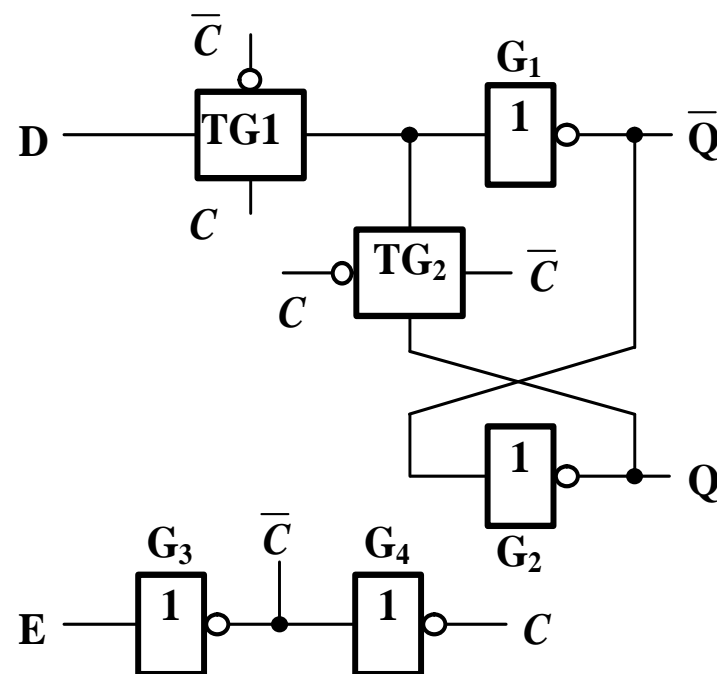
D锁存器的传输门实现



- 当S1合上，S2断开时， $Q=D$ ，跟随
- 当S2合上，S1断开时， Q 保持
- 传输门相当于受控的双向开关



- 当 $C=1$ ， $\bar{C}=0$ ：Y与X连通
- 当 $C=0$ ， $\bar{C}=1$ ：Y与X断开

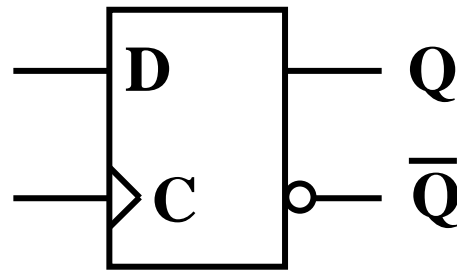


内容提纲

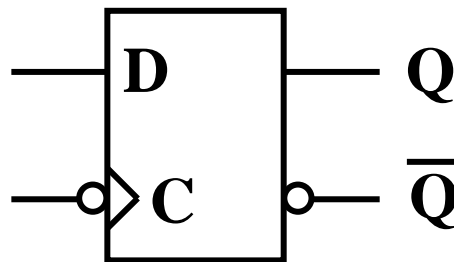
- 触发器的实现和功能描述
- 触发器的逻辑功能转换
- 锁存器和触发器的动态特性

D触发器(Flip-Flop)

- 只在时钟信号的上升沿或下降沿变化瞬间，根据输入信号更新状态
 - 最多一次翻转
- 在其他时间状态保持不变



(a)



(b)

逻辑符号

特性表(a)

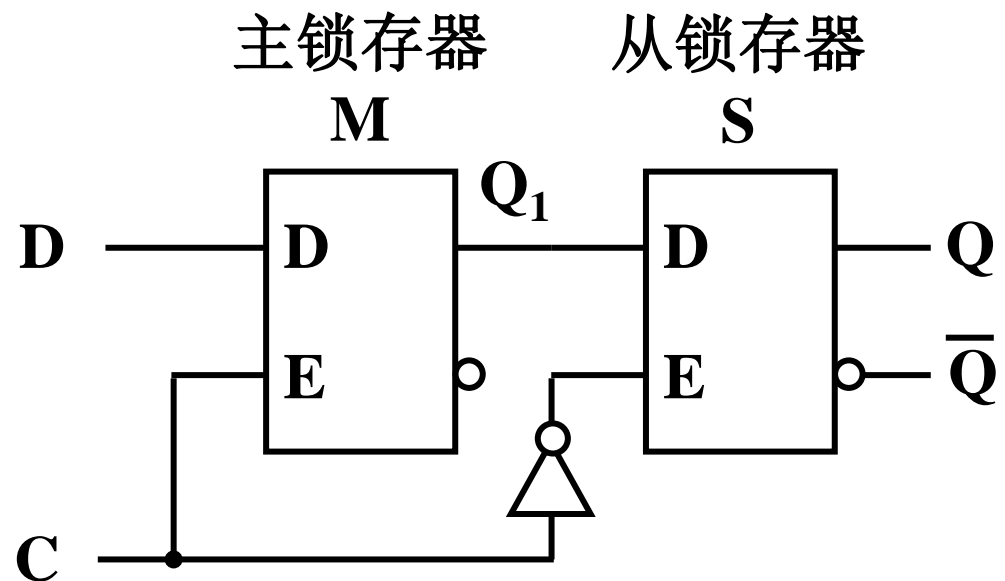
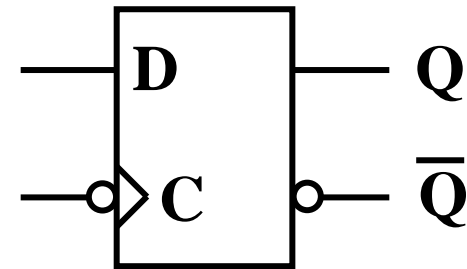
C	D	Q^{n+1}
\uparrow	x	D
x	x	Q^n

特性表(b)

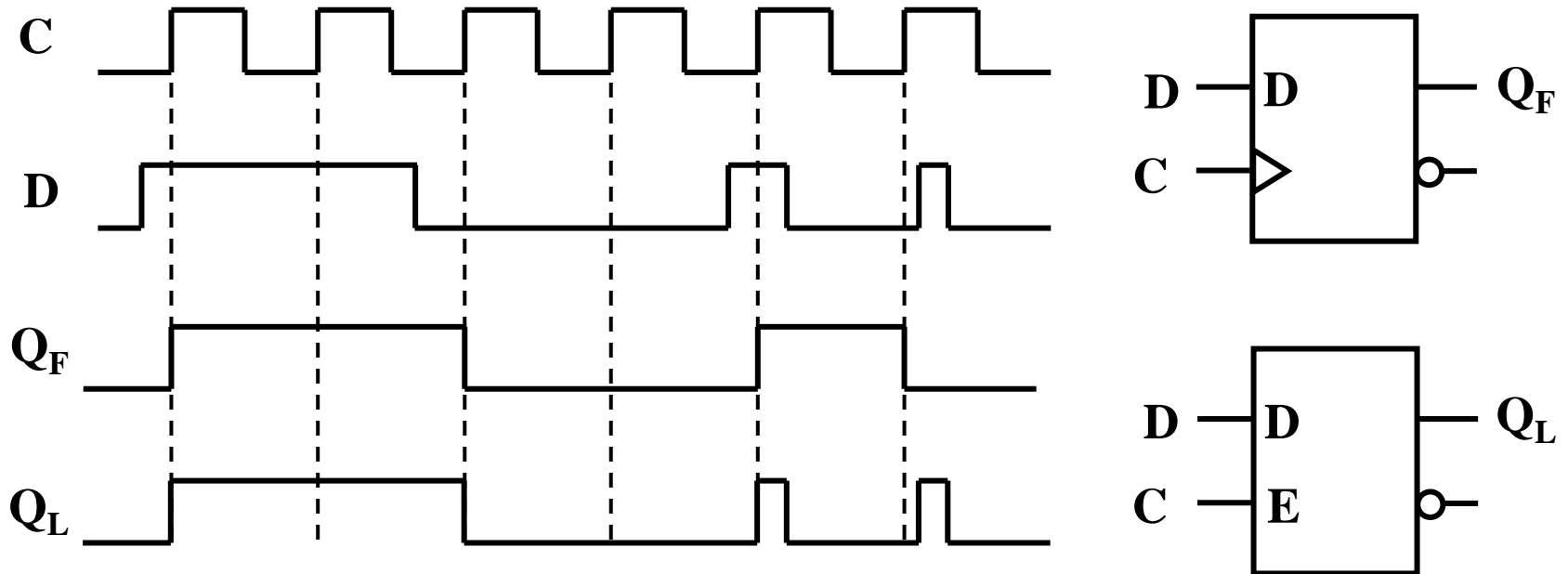
C	D	Q^{n+1}
\downarrow	x	D
x	x	Q^n

D触发器—主从结构

- 当 $C=1$ 时，主锁存器 M 随 D 变化而变化，从锁存器 S 保持原先状态不变
- 当 C 从 1 变为 0 (下降沿) 时，主锁存器 M 锁存此时 D 的值，从锁存器 S 按 M 状态更新



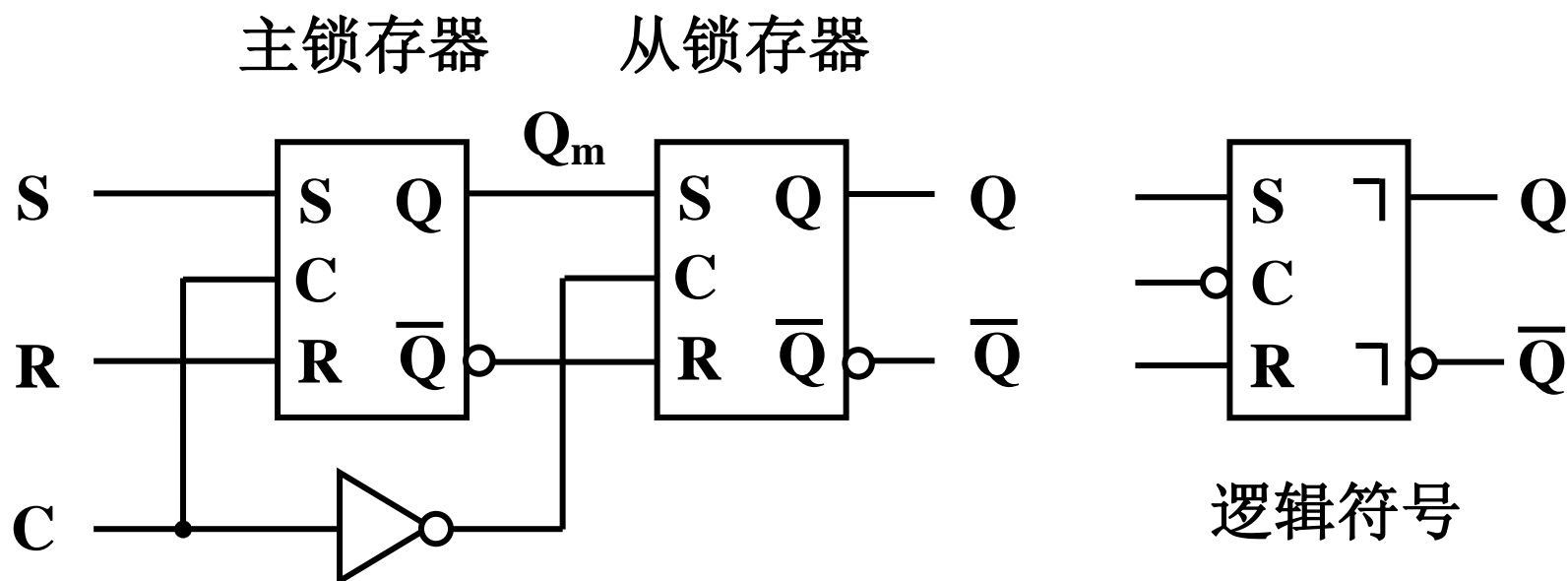
示例—D触发器波形图



如果在**C**高电平期间**D**不变化，则两者行为相同

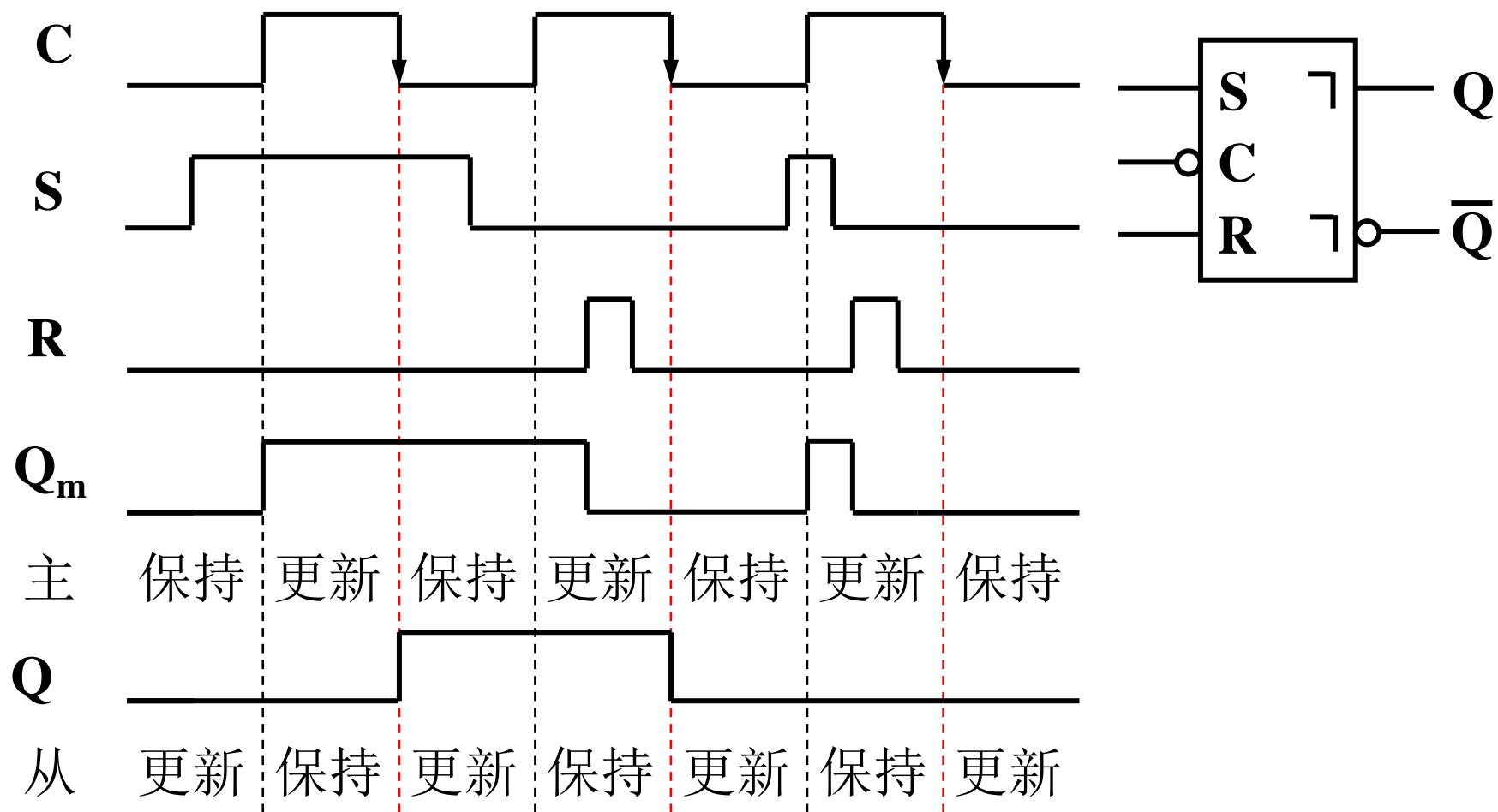
如果改成低电平和下降沿有效的器件，波形如何？

主从结构SR触发器



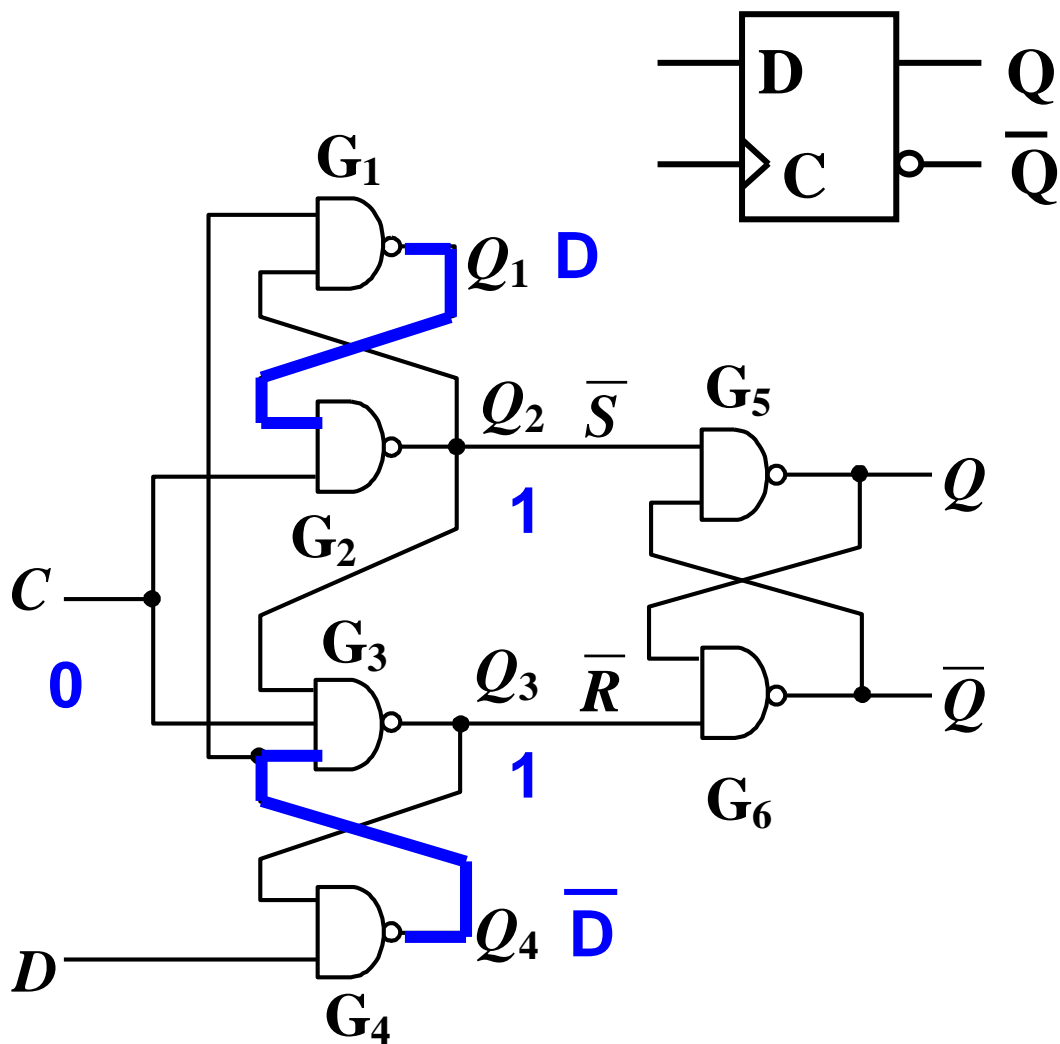
- 主锁存器按输入信号S、R更新时，从锁存器保持
- 主锁存器保持时，从锁存器按主锁存器状态更新
 - 主锁存器可能翻转多次，但从锁存器只能翻转一次

主从结构SR触发器波形图



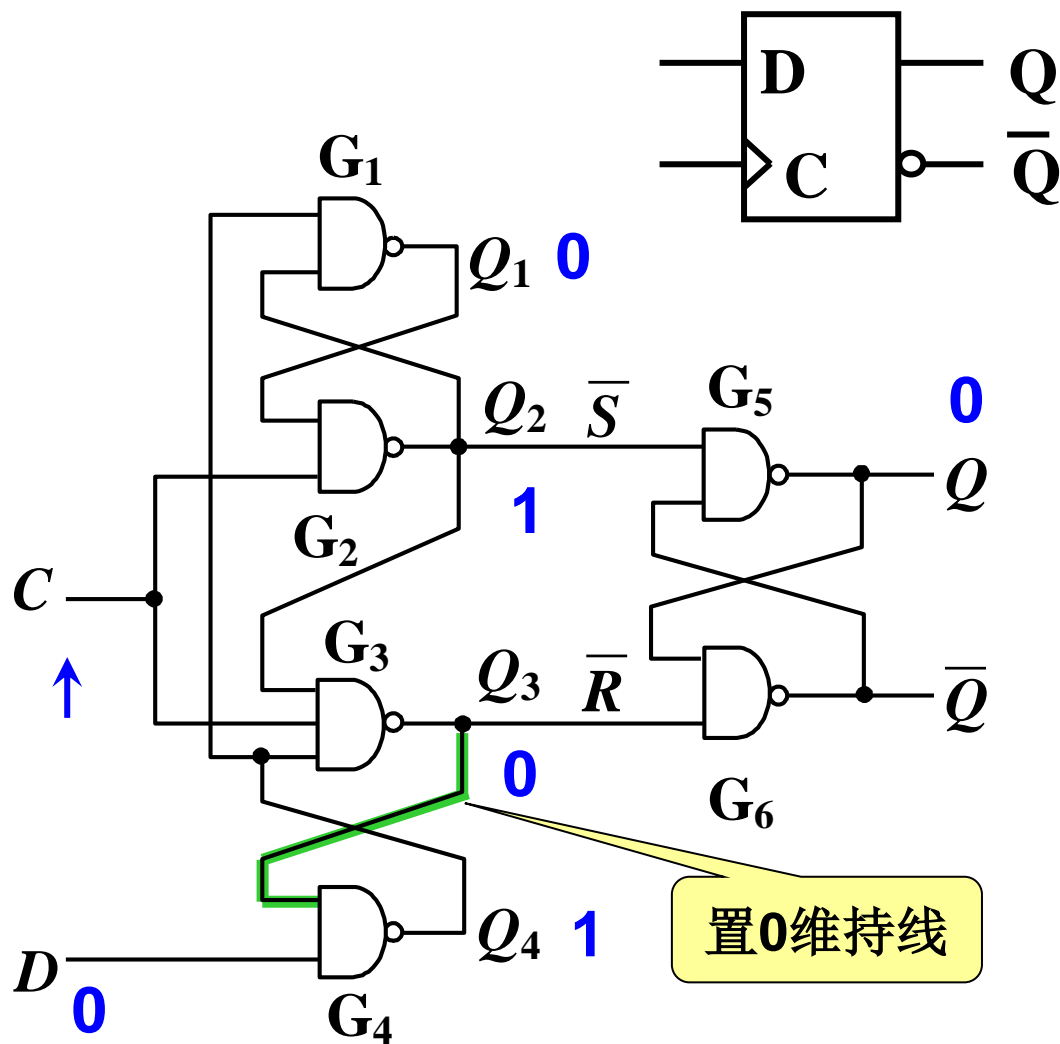
维持阻塞结构D触发器

- 当 $C=0$ 时
 - $Q_2=1$, $Q_3=1$, Q 、 \bar{Q} 保持
 - $Q_4=\bar{D}$, $Q_1=D$
- 触发器为状态更新作好准备



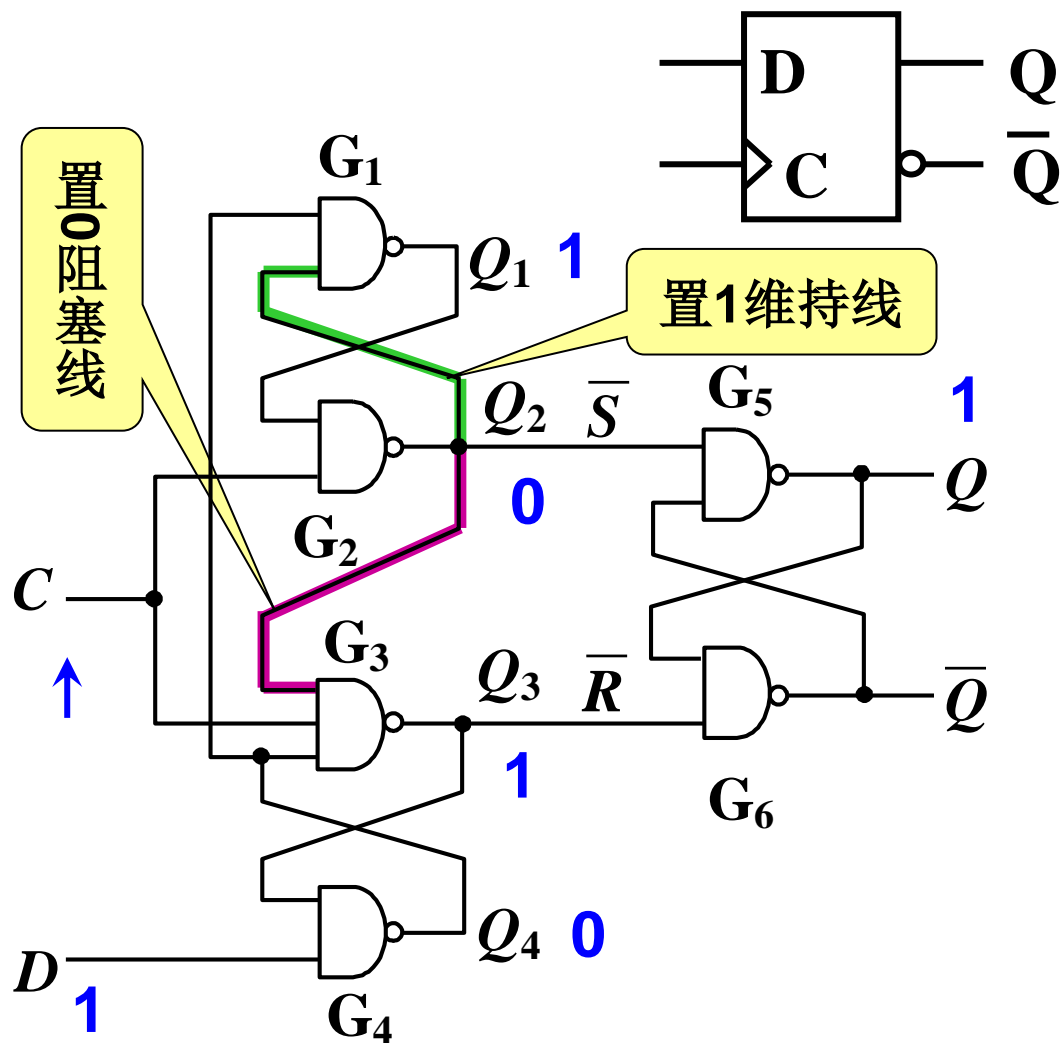
维持阻塞结构D触发器(续1)

- C由0变为1时
- 若 $Q_4=1$, $Q_1=0$, 即 $D=0$, 则
 - $Q_3=0$, $Q_2=1$
 - $Q_3=0$, 将Q置0, 封锁G4, 随后D的变化不影响Q



维持阻塞结构D触发器(续2)

- C由0变为1时
- 若 $Q_4=0$, $Q_1=1$, 即 $D=1$, 则
 - $Q_3=1$, $Q_2=0$
 - $Q_2=0$, 将Q置1, 封锁G1、G3, 维持Q, 阻止随后D的变化影响Q



维持阻塞结构D触发器(续2)

● 综合：

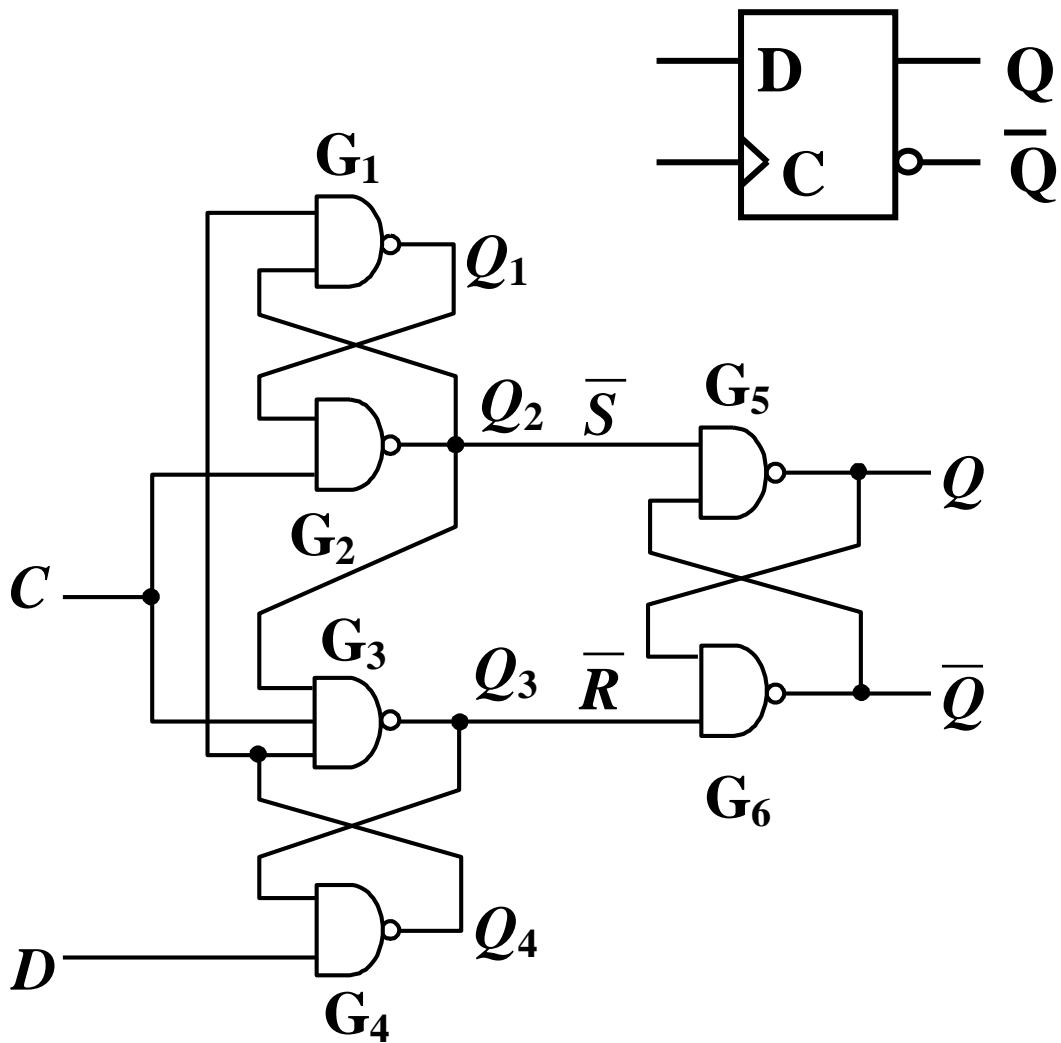
— 当 $C=0$ 时：

- 触发器保持原先状态，为更新做好准备

— 当C从0变1时:

- Q变为D值，且随后封锁输入端的更新，保持这个D值不变

➔ 上升沿有效D触发器



触发器逻辑功能描述

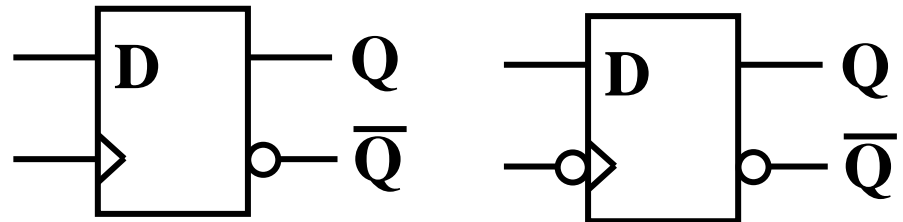
- 在时钟信号有效边沿(上升沿或下降沿), 触发器根据输入信号更新状态
 - 现态: 时钟信号有效边沿前触发器的状态, 记为 Q^n
 - 次态: 时钟信号有效边沿后触发器的状态, 记为 Q^{n+1}
- 触发器的逻辑功能是指次态与现态和输入信号之间的逻辑关系
 - 描述方法: 特性表(真值表)、特性方程(逻辑表达式)、状态图(状态转换图)、硬件描述语言 (HDL)
- 触发器按逻辑功能分类: **D**触发器、**SR**触发器、**T**触发器、**JK**触发器

D触发器

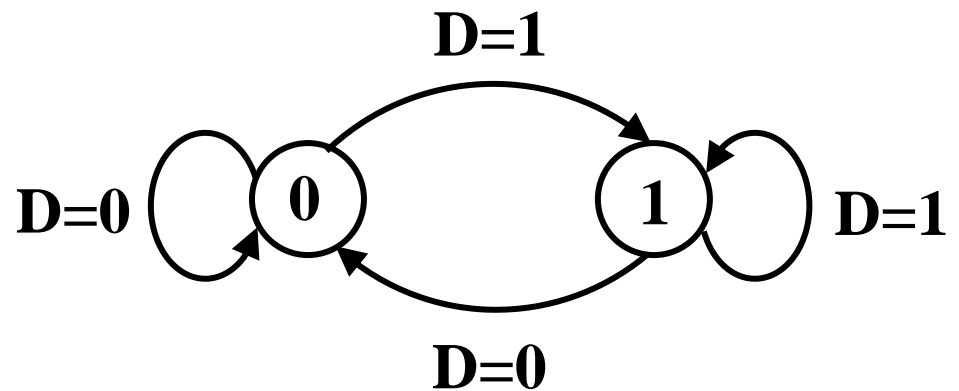
特性表

D	Q^n	Q^{n+1}	说明
0	0	0	清0
0	1	0	
1	0	1	置1
1	1	1	

特性方程 $Q^{n+1} = D$



逻辑符号



状态图

T触发器

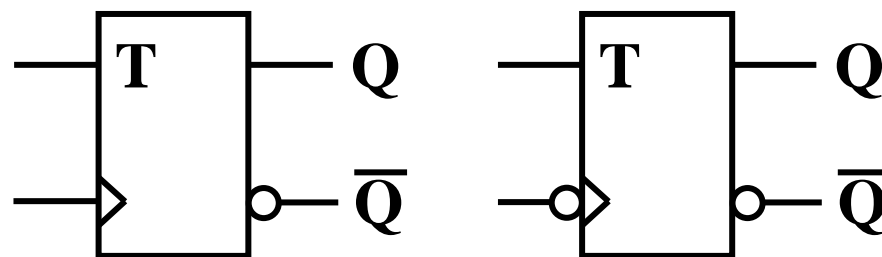
特性表

T	Q^n	Q^{n+1}	说明
0	0	0	保持
0	1	1	
1	0	1	翻转
1	1	0	

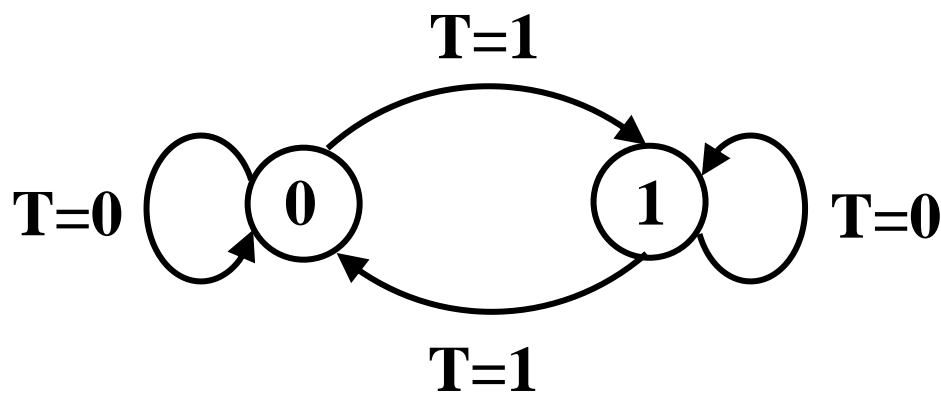
特性方程

$$Q^{n+1} = \bar{T}Q^n + T\bar{Q}^n$$

$$= T \oplus Q^n$$



逻辑符号

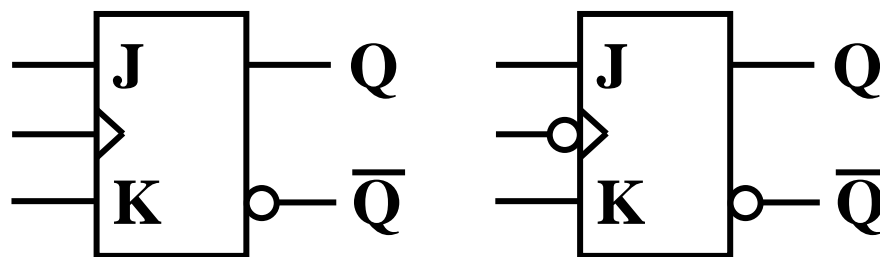


状态图

JK触发器

特性表

J	K	Q^n	Q^{n+1}	说明
0	0	0	0	保持
0	0	1	1	
0	1	0	0	清0
0	1	1	0	
1	0	0	1	置1
1	0	1	1	
1	1	0	1	翻转
1	1	1	0	



逻辑符号

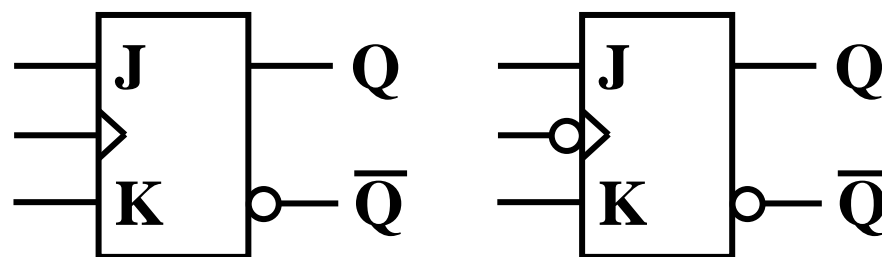
JK		00	01	11	10
Q^n	0	0	0	1	1
	1	1	0	0	1

特性方程 $Q^{n+1} = J \bar{Q}^n + \bar{K} Q^n$

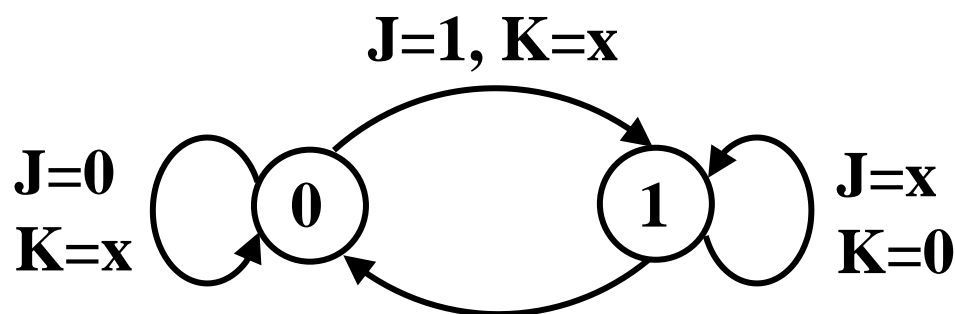
JK触发器 (续)

特性表

J	K	Q^n	Q^{n+1}	说明
0	0	0	0	保持
0	0	1	1	
0	1	0	0	清0
0	1	1	0	
1	0	0	1	置1
1	0	1	1	
1	1	0	1	翻转
1	1	1	0	



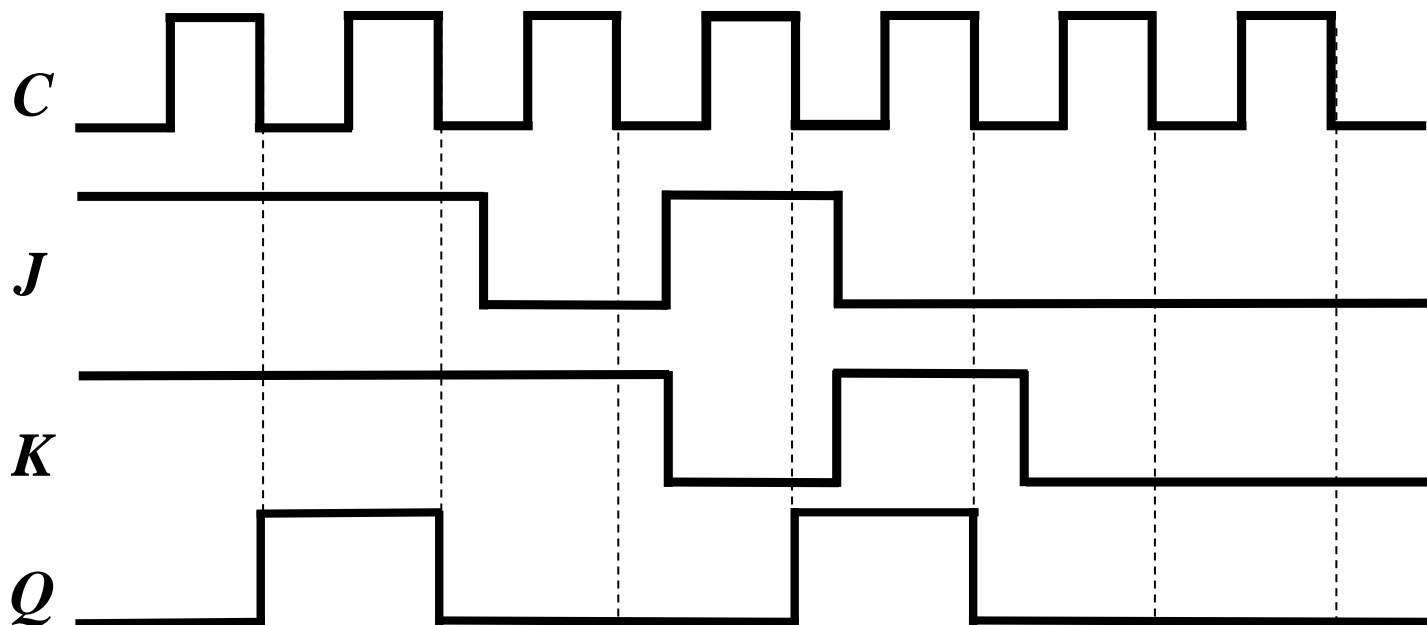
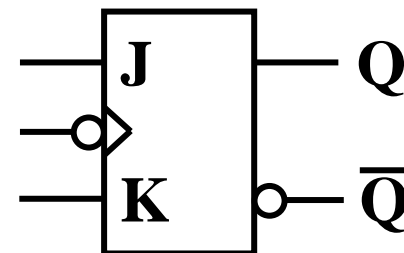
逻辑符号



状态图

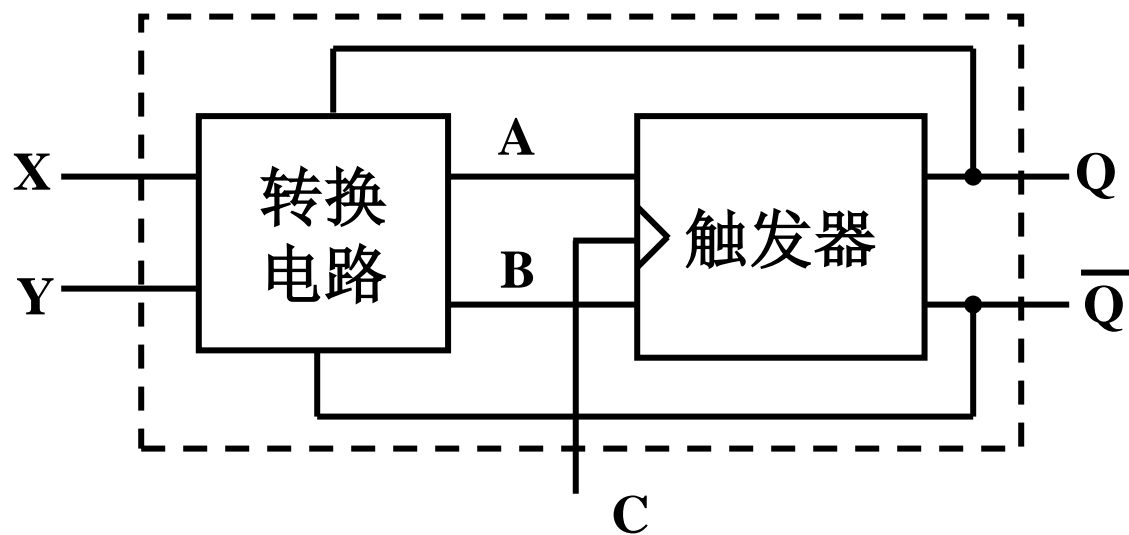
示例—JK触发器波形图

- 画出输出波形(设初态为0)

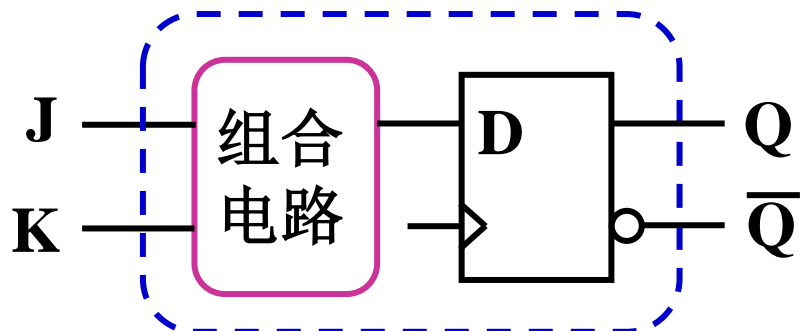


触发器逻辑功能转换

- 利用某种功能触发器来构造不同功能的触发器
 - 根据两种触发器的特性方程，求解转换电路的逻辑函数式



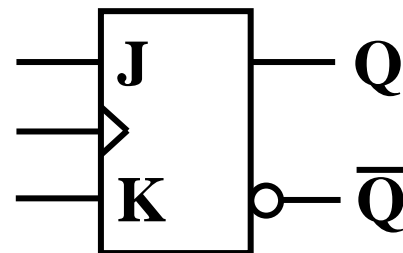
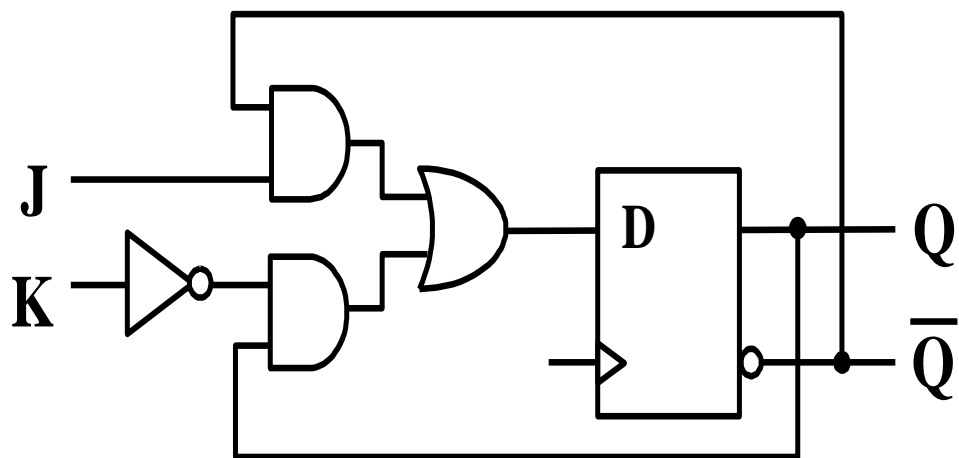
D触发器构成JK触发器



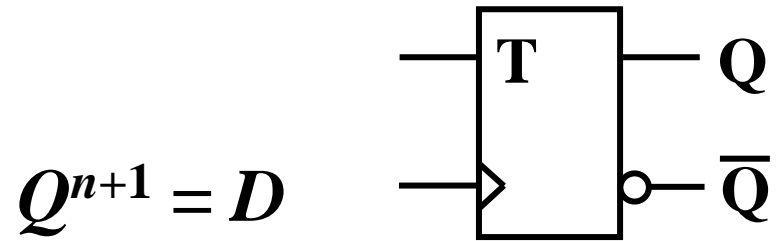
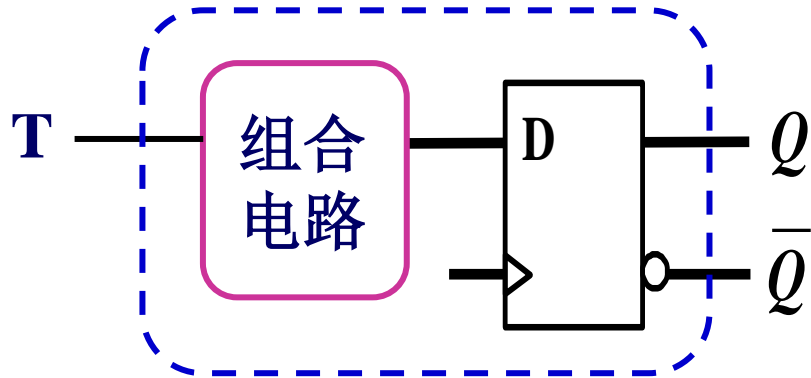
$$Q^{n+1} = D$$

$$Q^{n+1} = J \bar{Q}^n + \bar{K} Q^n$$

$$D = J \bar{Q}^n + \bar{K} Q^n$$



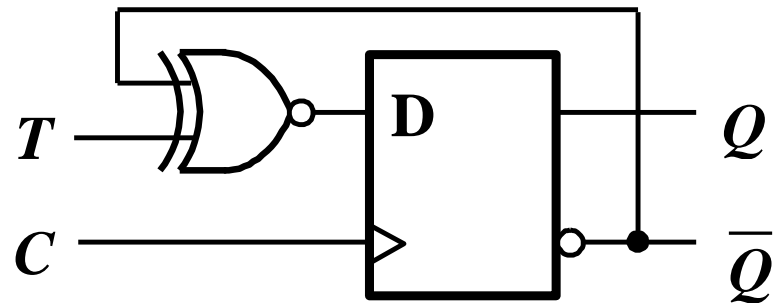
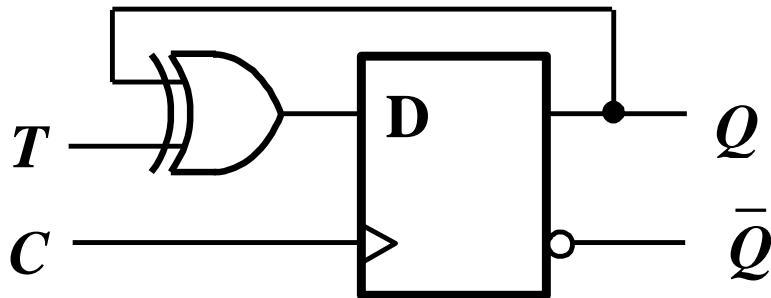
D触发器构成T触发器



$$Q^{n+1} = D$$

$$Q^{n+1} = T\bar{Q}^n + \bar{T}Q^n$$

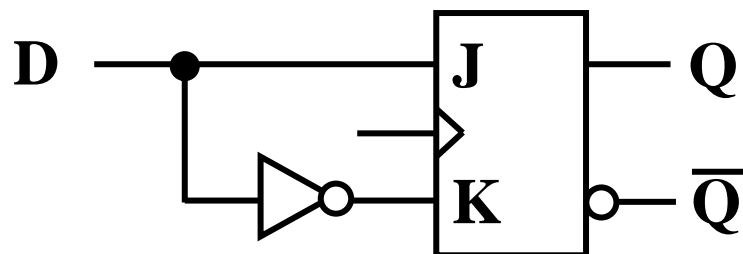
$$D = T\bar{Q}^n + \bar{T}Q^n = T \oplus Q^n$$



JK触发器构成其他触发器

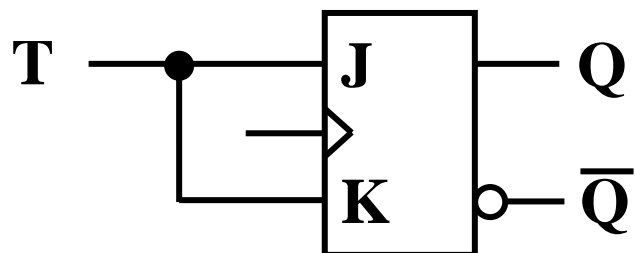
- 构成D触发器

– $J=D, K=\bar{D} \quad Q^{n+1} = D$



- 构成T触发器

– $J=K=T \quad Q^{n+1} = T \oplus Q^n$



JK触发器特性表

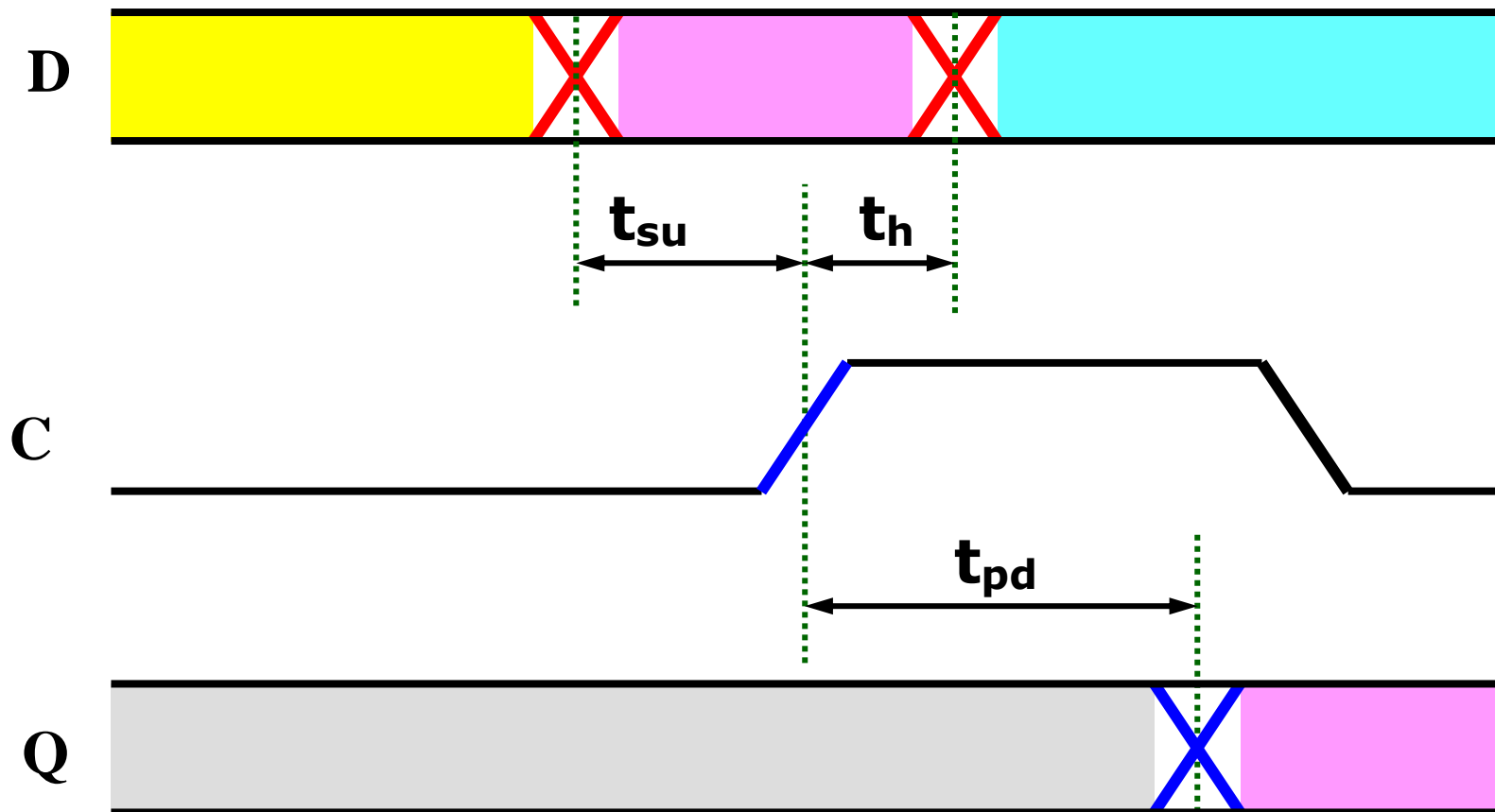
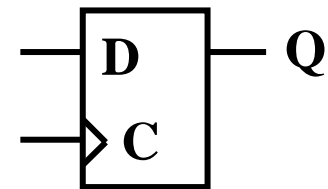
J	K	Q^n	Q^{n+1}	说明
0	0	0	0	保持
0	0	1	1	
0	1	0	0	清0
0	1	1	0	
1	0	0	1	置1
1	0	1	1	
1	1	0	1	翻转
1	1	1	0	

$$Q^{n+1} = J \bar{Q}^n + \bar{K} Q^n$$

锁存器和触发器的动态特性

- 保证锁存器和触发器可靠地更新状态，输入信号与时钟信号之间的时间要求
 - 建立时间 $t_{su}(\text{setup})$ ：要求输入信号在时钟有效边沿到来之前，提前一段时间做好准备
 - 保持时间 $t_h(\text{hold})$ ：在时钟有效边沿到达后，需要输入信号再保持一段时间
- 锁存器和触发器输出信号对时钟信号响应的延迟时间
 - 传输延迟时间 $t_{pd}(\text{propagation delay})$

D触发器定时波形图



示例—分析D触发器动态参数

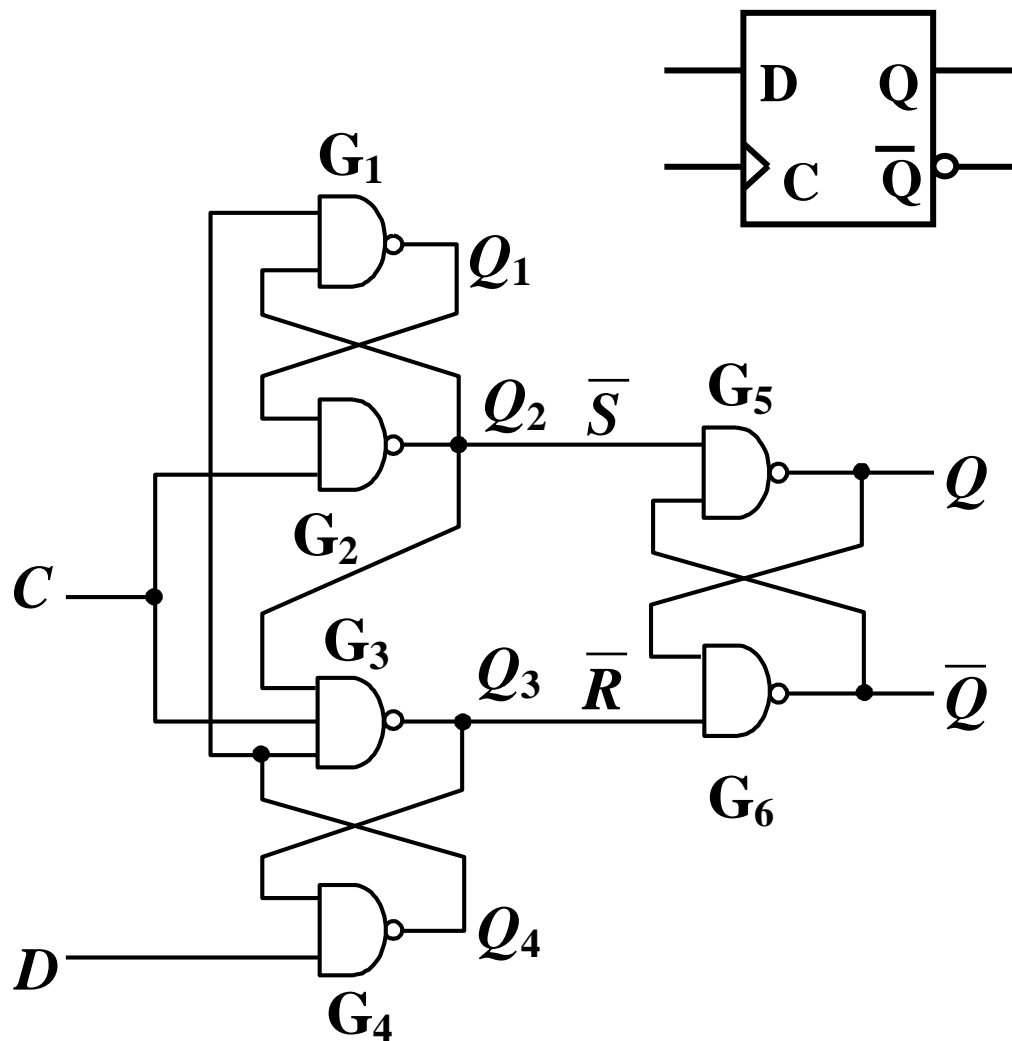
- 设所有门的传输延迟都等于 t_d ,
忽略导线延迟,
求:

t_{su} , t_h , t_{pd}

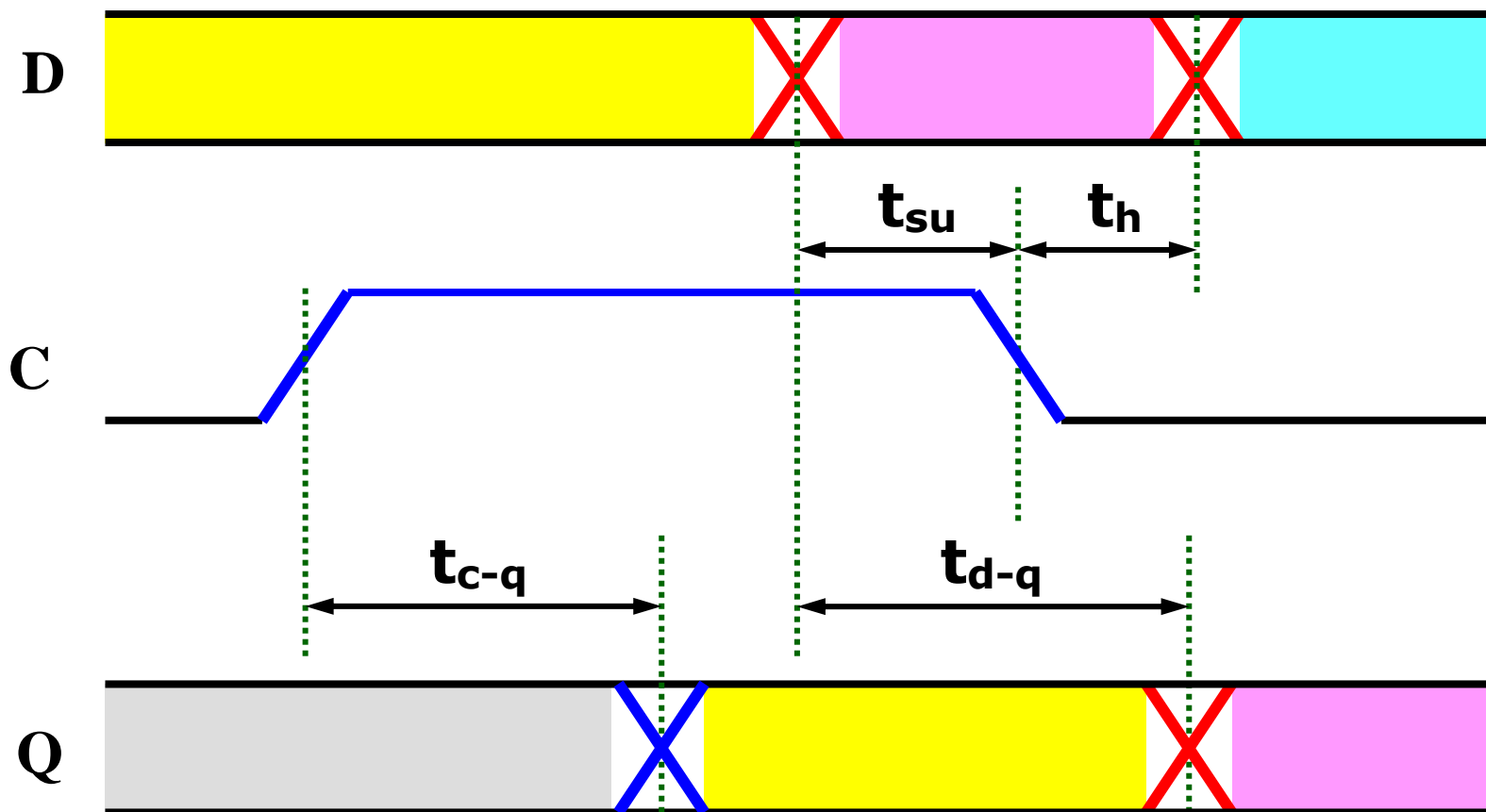
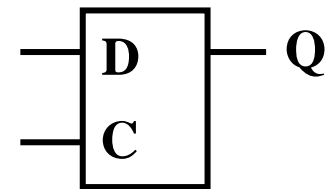
$$t_{su} = 2t_d$$

$$t_h = t_d$$

$$t_{pd} = 3t_d$$



D锁存器定时波形图

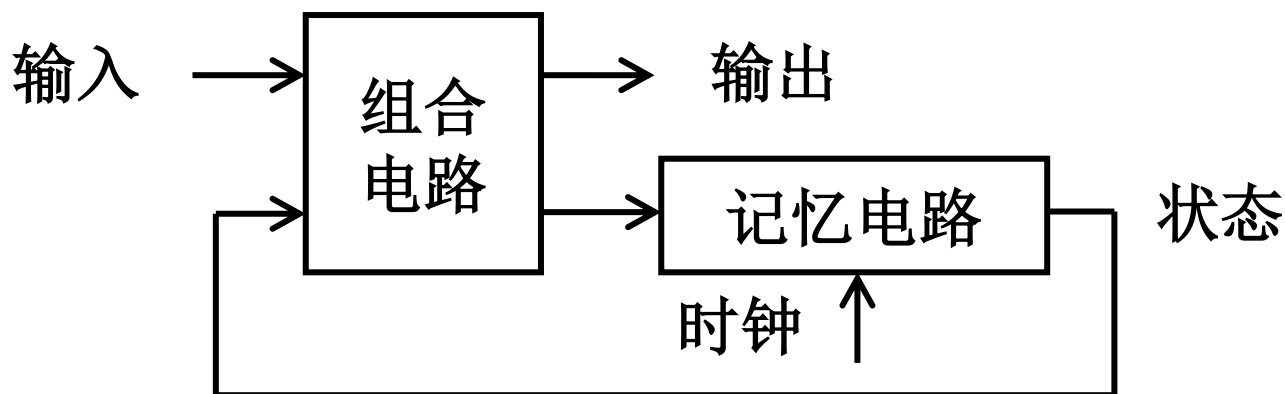


内容提纲

- 时序电路的分类
- 时序电路的描述方式
- 同步时序电路的分析方法

时序电路的分类

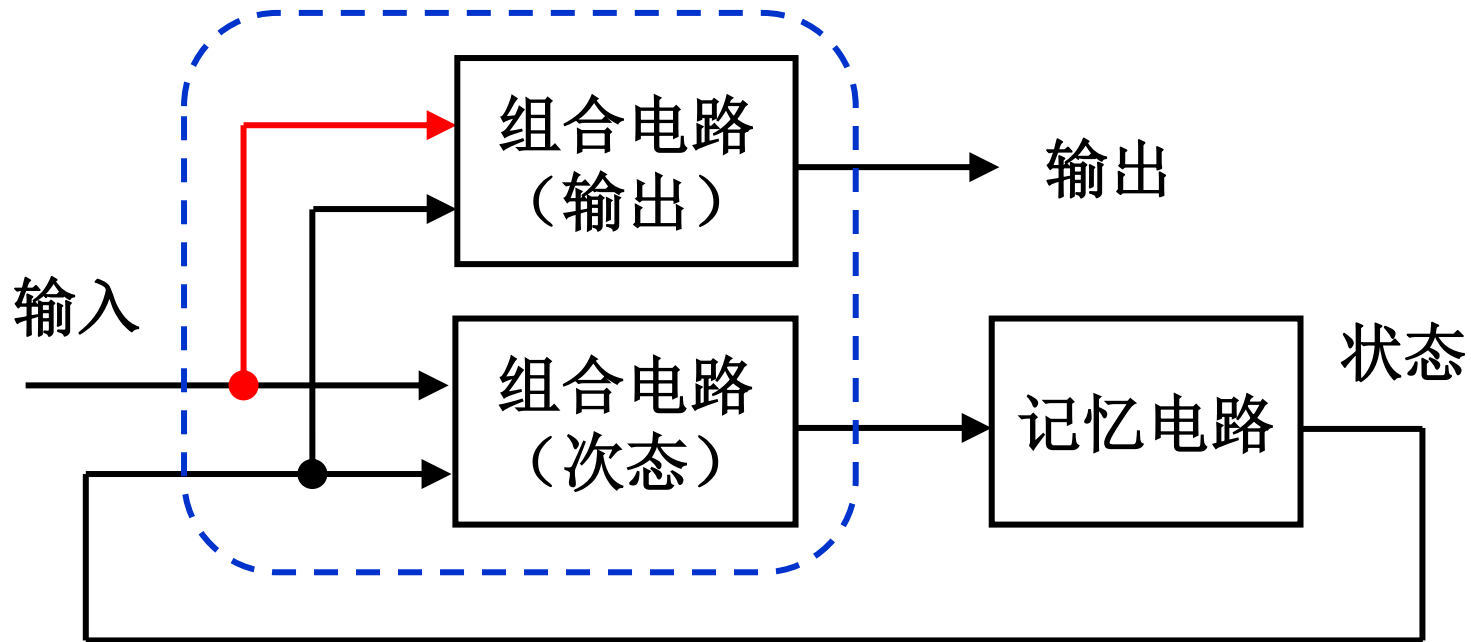
- 根据记忆电路状态更新的特点，时序电路分为同步时序电路和异步时序电路



- 同步时序电路：所有记忆电路由统一的时钟信号控制，它们的状态在同一时刻更新
- 异步时序电路：没有统一的时钟信号或没有时钟信号，记忆电路的状态更新不是同时发生的

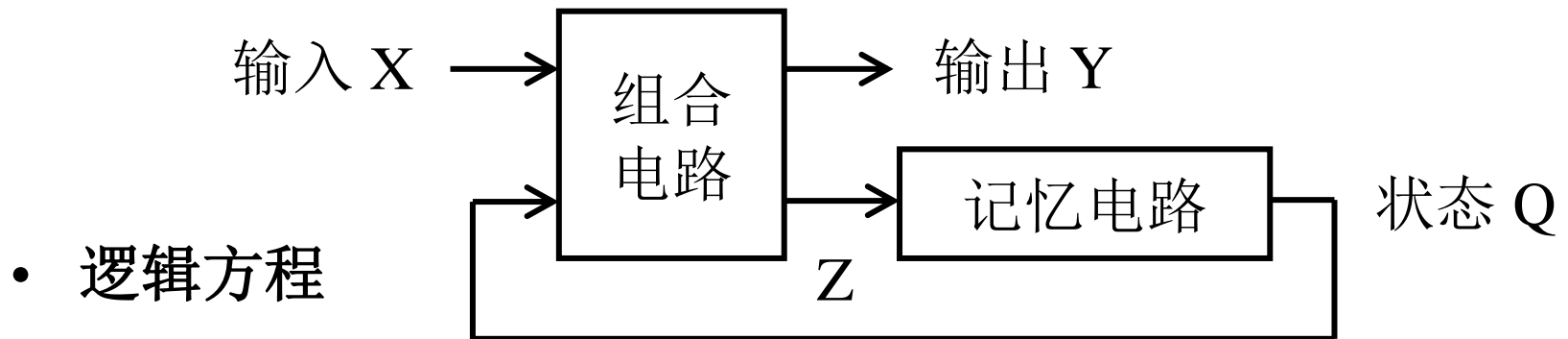
米里型和摩尔型时序电路

- 米利 (Mealy)型：输出是输入和状态的函数
- 穆尔 (Moore)型：输出仅是状态的函数



时序电路的描述方式

- 逻辑方程、状态转换表（状态表）、状态转换图（状态图）、时序波形图（时序图）、**HDL描述**
 - 不同描述方式是等价的，可以相互转换



激励方程: $Z = f_1(X, Q)$

状态方程: $Q^{n+1} = f_2(Z, Q^n)$

输出方程: $Y = f_3(X, Q)$ ---- Mealy型

$Y = f_4(Q)$ ---- Moore型

逻辑方程和状态表

激励方程: $Z = f_1(X, Q)$

状态方程: $Q^{n+1} = f_2(Z, Q^n)$

输出方程: $Y = f_3(X, Q)$ ---- Mealy型

$Y = f_4(Q)$ ---- Moore型

状态表(Mealy型)

Q^n	Q^{n+1}/Y	
	$X=i$	$X=j$
...	.../...	.../...

状态表(Moore型)

Q^n	Q^{n+1}		Y
	$X=i$	$X=j$	
...

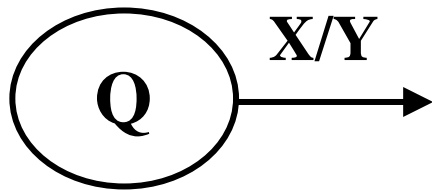
状态图

激励方程: $Z = f_1(X, Q)$

状态方程: $Q^{n+1} = f_2(Z, Q^n)$

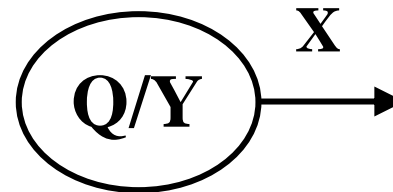
输出方程: $Y = f_3(X, Q)$ ---- Mealy型

$Y = f_4(Q)$ ---- Moore型



状态图(米利型)

状态Q转换到另一个, 同时输出Y变化, 两个都受到X直接影响



状态图(穆尔型)

状态Q转换到另一个, 在某个状态下Q值确定, 与当前输入X无关, 只有Q直接受到X影响

示例1—时序电路描述方式

- 逻辑方程

输出方程

$$Y = (Q_0 + Q_1)\bar{X}$$

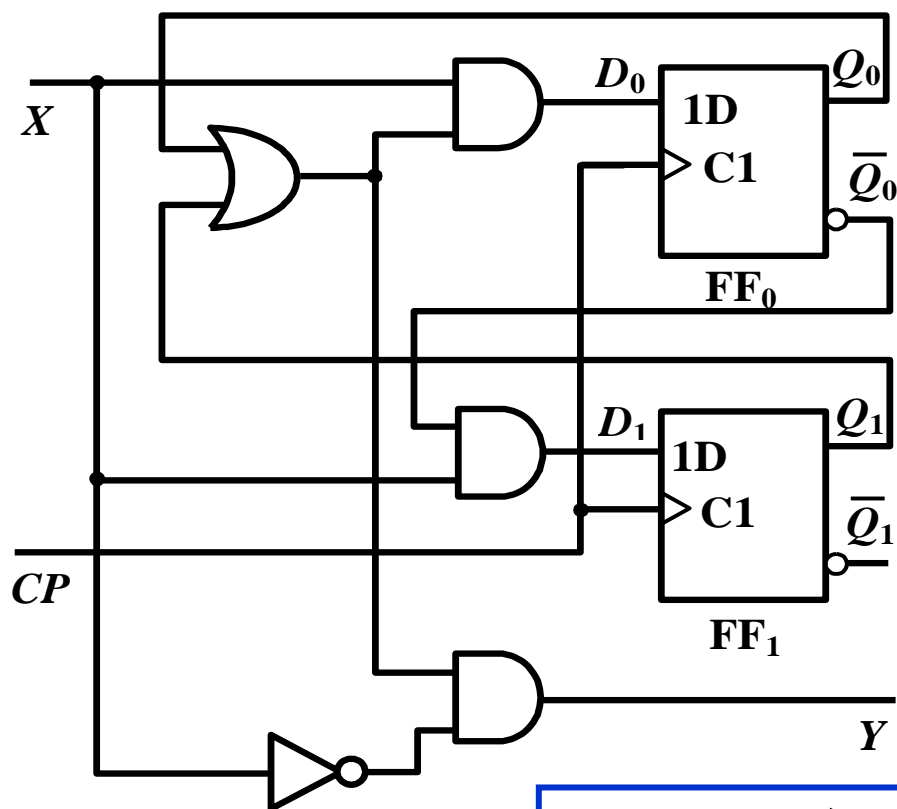
激励方程

$$D_1 = \bar{Q}_0 X$$

$$D_0 = (Q_0 + Q_1)X$$

状态方程

$$Q_1^{n+1} = \bar{Q}_0^n X, \quad Q_0^{n+1} = (Q_0^n + Q_1^n)X$$



Mealy型
时序电路

示例1—时序电路描述方式(续1)

- 状态表

$Q_1^n Q_0^n$	$Q_1^{n+1} Q_0^{n+1} / Y$	
	X=0	X=1
0 0	0 0 / 0	1 0 / 0
0 1	0 0 / 1	0 1 / 0
1 0	0 0 / 1	1 1 / 0
1 1	0 0 / 1	0 1 / 0

$$Q_1^{n+1} = \overline{Q_0^n} X$$

$$Q_0^{n+1} = (Q_0^n + Q_1^n) X$$

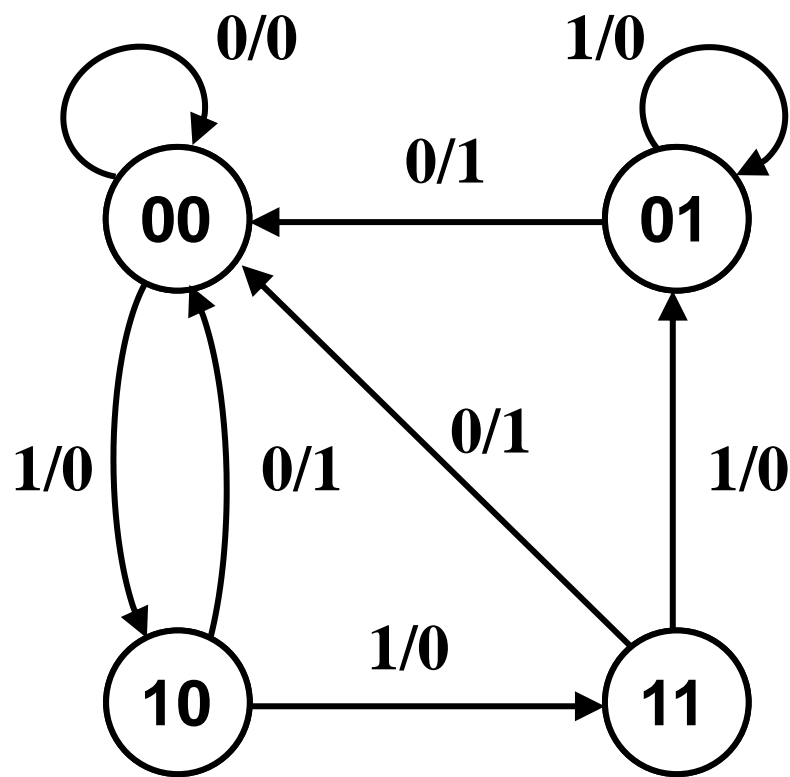
$$Y = (Q_0 + Q_1) \overline{X}$$

Mealy型
时序电路

看当前的 Q_1^n, Q_0^n 和变化后的X来确定输出Y

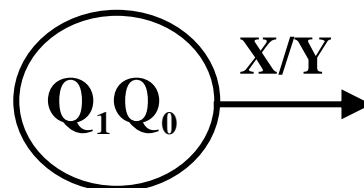
示例1—时序电路描述方式(续2)

- 状态图



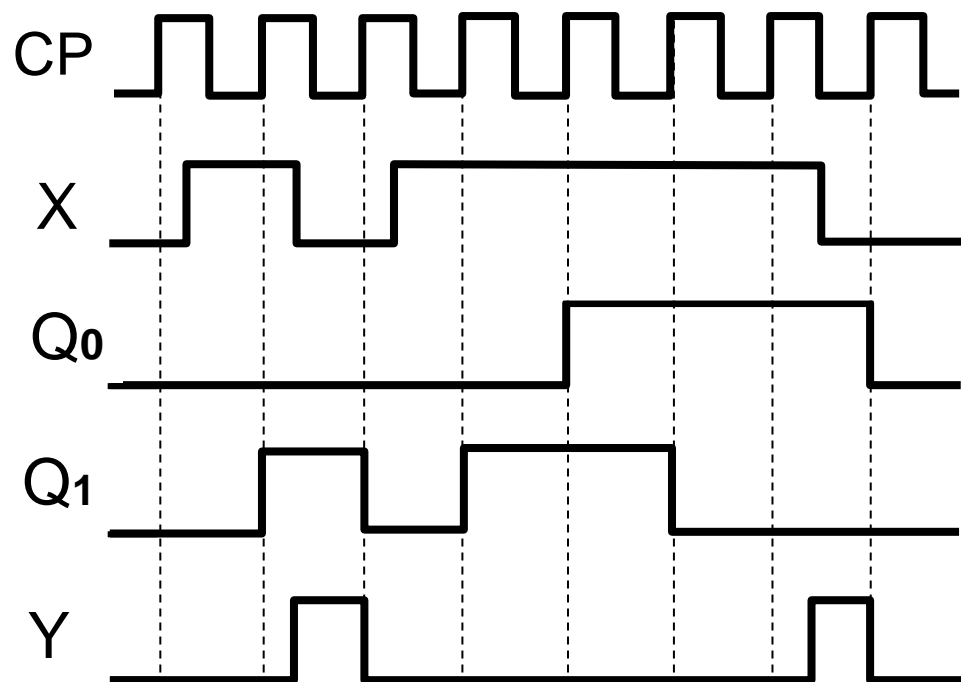
状态表

$Q_1^n Q_0^n$	$Q_1^{n+1} Q_0^{n+1} / Y$	
	X=0	X=1
0 0	0 0 / 0	1 0 / 0
0 1	0 0 / 1	0 1 / 0
1 0	0 0 / 1	1 1 / 0
1 1	0 0 / 1	0 1 / 0



示例1—时序电路描述方式(续3)

• 时序图



Q0,Q1在时钟沿上根据X和当前的Q0,Q1更新,
Y随时随刻根据Q0,Q1和X更新

状态表

$Q_1^n Q_0^n$	$Q_1^{n+1} Q_0^{n+1} / Y$	
	X=0	X=1
0 0	0 0 / 0	1 0 / 0
0 1	0 0 / 1	0 1 / 0
1 0	0 0 / 1	1 1 / 0
1 1	0 0 / 1	0 1 / 0

$$Y = (Q_0 + Q_1) \bar{X}$$

同步时序电路的分析

- 已知逻辑电路图，确定其逻辑功能
- 一般分析步骤
 - 根据逻辑图，写出逻辑方程
 - 输出方程
 - 激励方程：每个触发器的输入驱动方程
 - 状态方程：将激励方程代入触发器特性方程得到
 - 列出状态表、画出状态图或时序图
 - 确定电路的逻辑功能

示例2—分析时序电路

- 写出逻辑方程

Moore型时序电路

输出方程

$$Y = Q_2 Q_1$$

激励方程

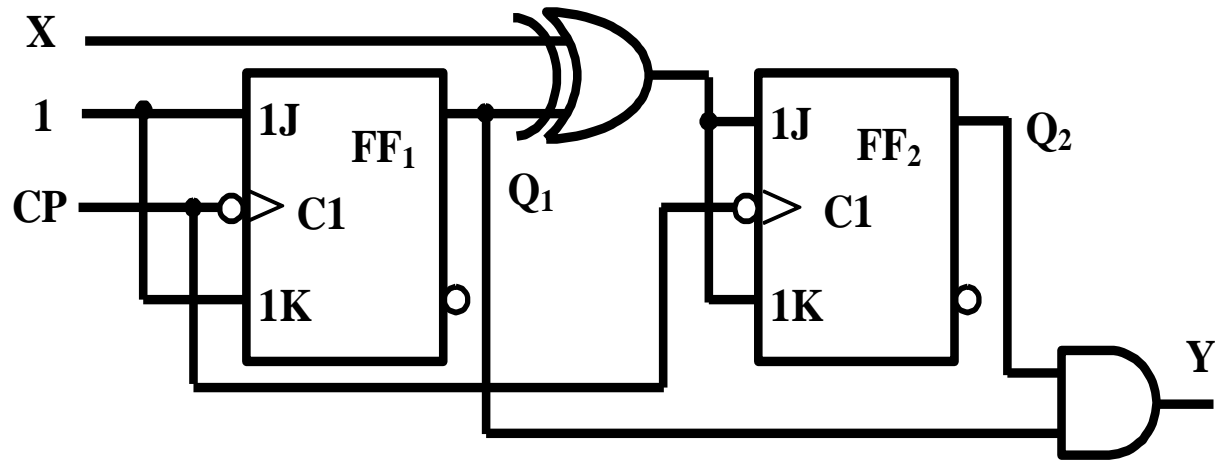
$$J_1 = K_1 = 1$$

$$J_2 = K_2 = X \oplus Q_1$$

状态方程

$$Q_1^{n+1} = \overline{Q_1^n}$$

$$Q_2^{n+1} = X \oplus Q_1^n \oplus Q_2^n$$



$$[Q^{n+1} = J \overline{Q}^n + \overline{K} Q^n]$$

示例2—分析时序电路(续1)

- 列出状态表

输出方程

$$Y = Q_2 Q_1$$

状态方程

$$Q_1^{n+1} = \overline{Q_1^n}$$

$$Q_2^{n+1} = \underline{X \oplus Q_1^n \oplus Q_2^n}$$

有1个1或3个1时输出为1，否则为0

状态表

$Q_2^n Q_1^n$	$Q_2^{n+1} Q_1^{n+1}$		Y
	$X=0$	$X=1$	
0 0	0 1	1 1	0
0 1	1 0	0 0	0
1 0	1 1	0 1	0
1 1	0 0	1 0	1

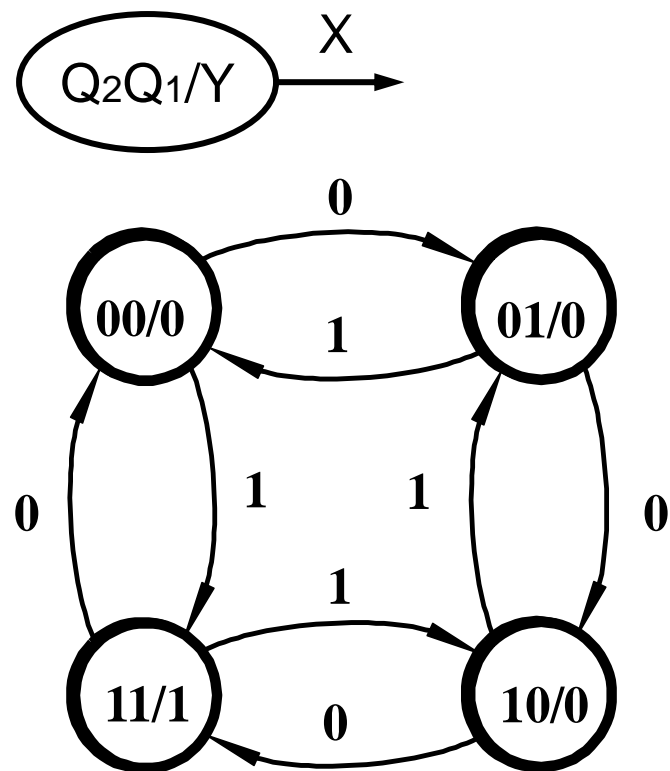
看当前的Q2,Q1和变化后的X来确定输出Y

示例2—分析时序电路(续2)

- 画出状态图

状态表

$Q_2^n Q_1^n$	$Q_2^{n+1} Q_1^{n+1}$		Y
	$X=0$	$X=1$	
0 0	0 1	1 1	0
0 1	1 0	0 0	0
1 0	1 1	0 1	0
1 1	0 0	1 0	1



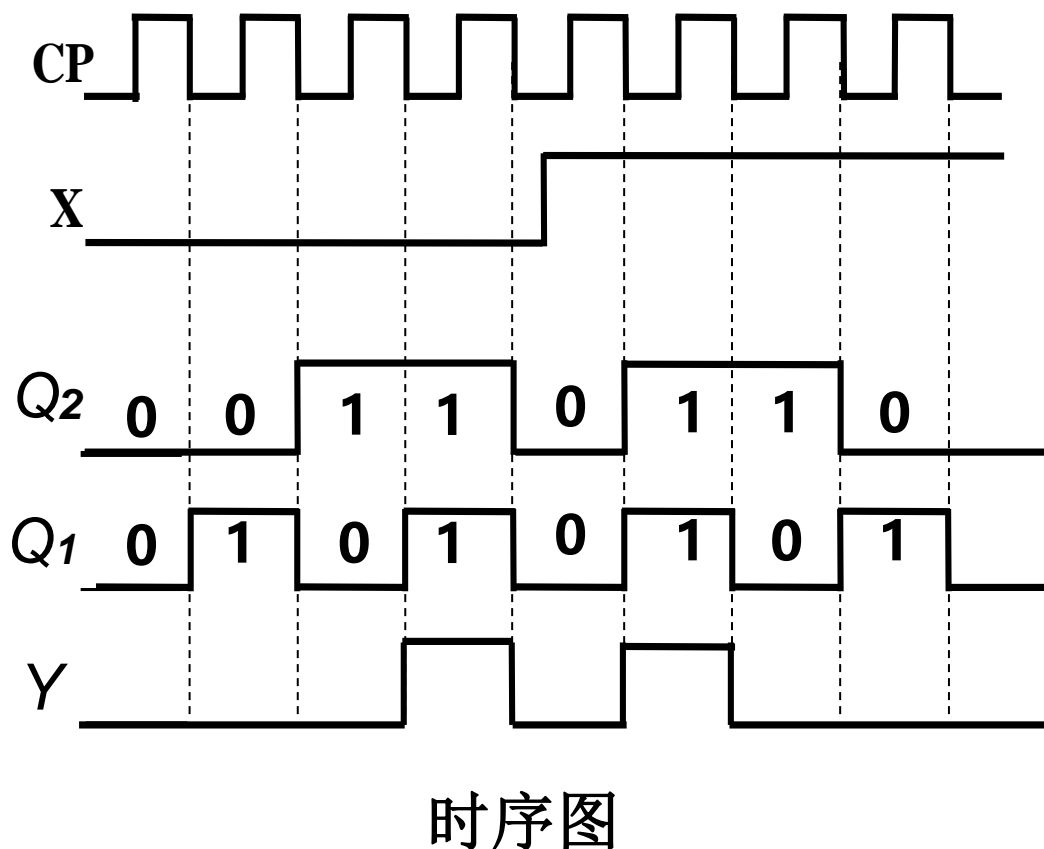
状态图

示例2—分析时序电路(续3)

- 画出时序图

状态表

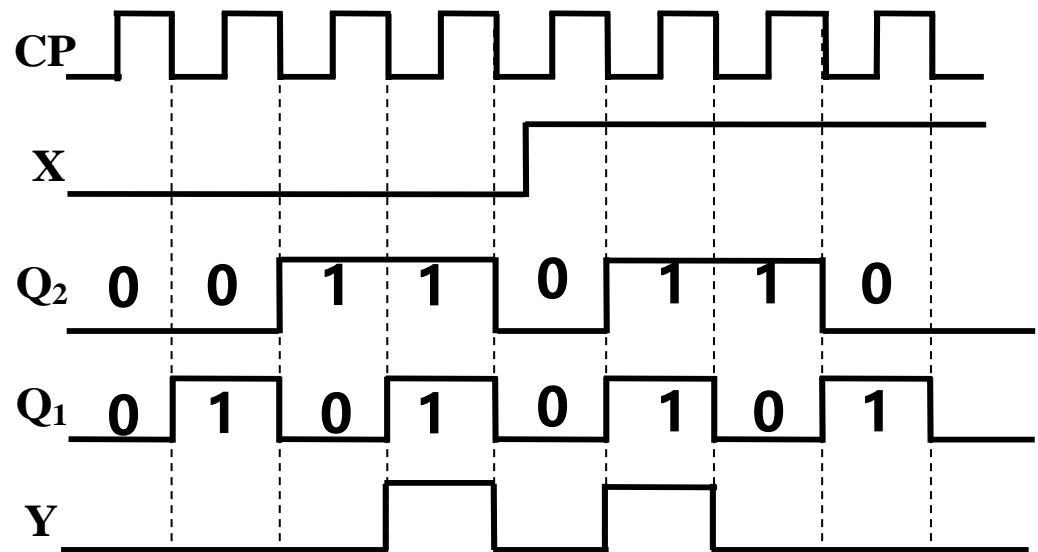
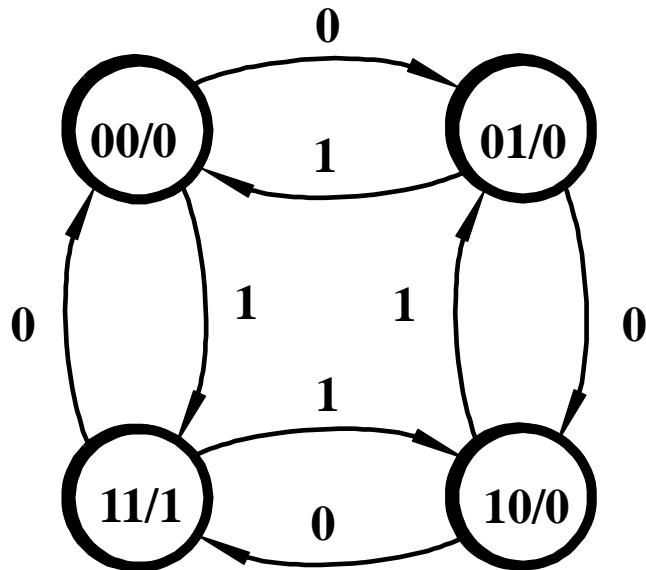
$Q_2^n Q_1^n$	$Q_2^{n+1} Q_1^{n+1}$		Y
	$X=0$	$X=1$	
0 0	0 1	1 1	0
0 1	1 0	0 0	0
1 0	1 1	0 1	0
1 1	0 0	1 0	1



Q1,Q2在时钟沿上根据X和当前的Q1,Q2更新, Y随时随刻根据Q0,Q1和X更新

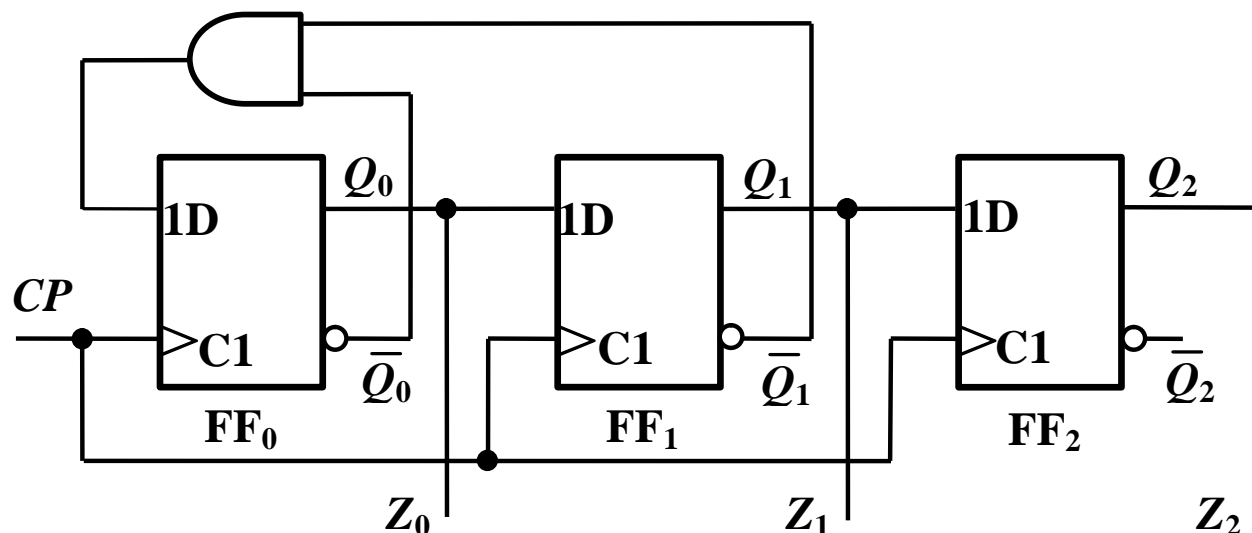
示例2—分析时序电路(续4)

- 确定电路逻辑功能：可逆计数器
 - 当 $X=0$ 时，电路进行加1计数
 - 当 $X=1$ 时，电路进行减1计数
 - Y 可理解为进位或借位



示例3—分析时序电路

- 写出逻辑方程



输出方程

$$Z_0 = Q_0$$

$$Z_1 = Q_1$$

$$Z_2 = Q_2$$

激励方程

$$D_0 = \overline{Q_1^n} \overline{Q_0^n}$$

$$D_1 = Q_0^n$$

$$D_2 = Q_1^n$$

状态方程 [$Q^{n+1} = D$]

$$Q_0^{n+1} = \overline{Q_1^n} \overline{Q_0^n}$$

$$Q_1^{n+1} = Q_0^n$$

$$Q_2^{n+1} = Q_1^n$$

示例3—分析时序电路(续1)

- 列出状态表

$$Q_0^{n+1} = \overline{Q_1^n} \overline{Q_0^n} = \overline{Q_1^n + Q_0^n}$$

$$Q_1^{n+1} = Q_0^n$$

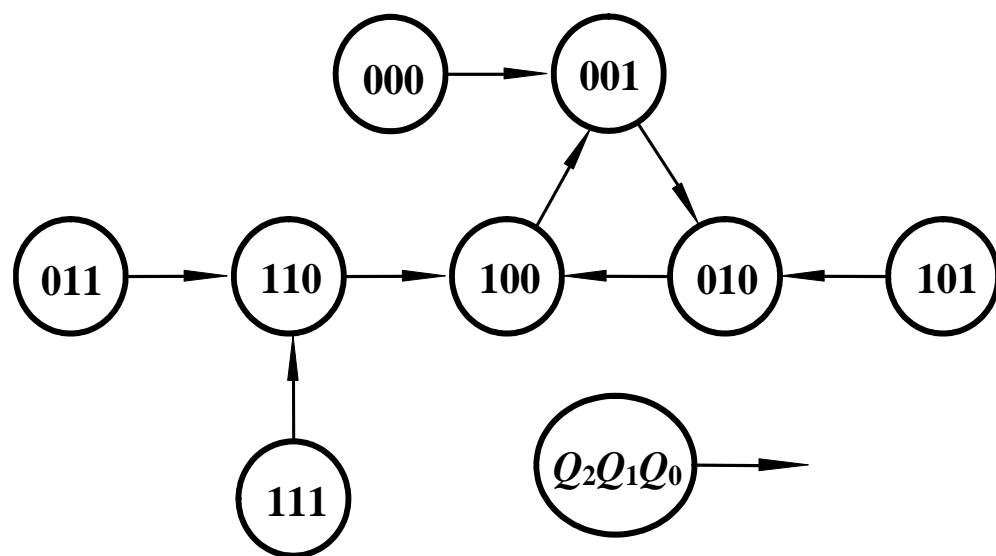
$$Q_2^{n+1} = Q_1^n$$

状态表

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	1 0 0
0 1 1	1 1 0
1 0 0	0 0 1
1 0 1	0 1 0
1 1 0	1 0 0
1 1 1	1 1 0

示例3—分析时序电路(续2)

- 画出状态图



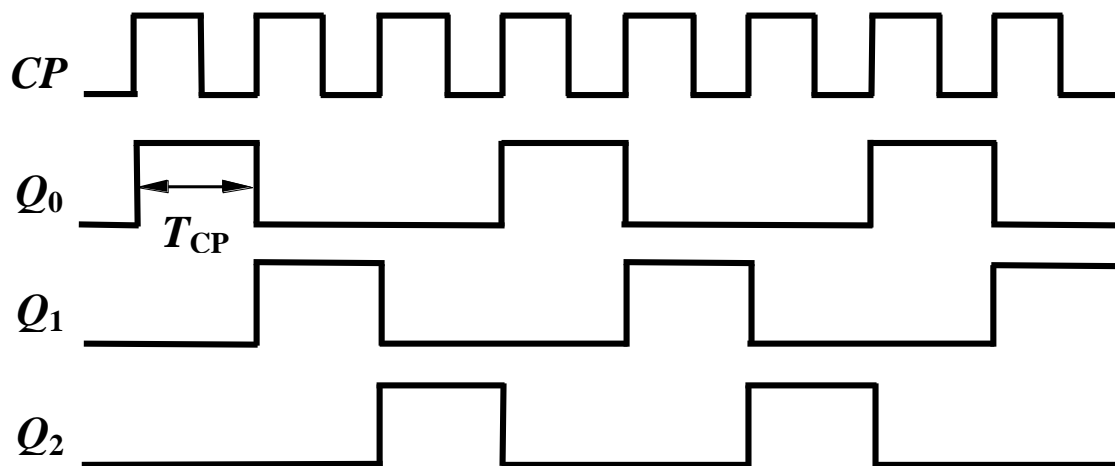
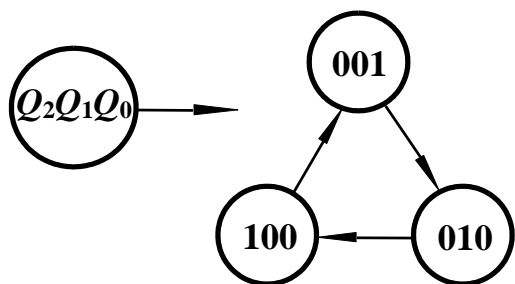
状态图

状态表

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$		
0 0 0	0	0	1
0 0 1	0	1	0
0 1 0	1	0	0
0 1 1	1	1	0
1 0 0	0	0	1
1 0 1	0	1	0
1 1 0	1	0	0
1 1 1	1	1	0

示例3—分析时序电路(续3)

- 画出时序图



- 由状态图可见，电路有3个循环的有效状态
- 从时序图可看出，电路正常工作时，各触发器的 Q 端轮流出现一个宽度为一个时钟周期(T_{CP})的脉冲信号，循环周期为 $3T_{CP}$
- 电路的功能为脉冲分配器或节拍脉冲产生器

模拟与数字电路

Analog and Digital Circuits

14_时序逻辑电路(2)

(数电P289-P302)

内容提纲

- 同步时序电路的设计
- 示例1 — 序列检测器
- 示例2 — 可逆六进制计数器

同步时序电路的设计

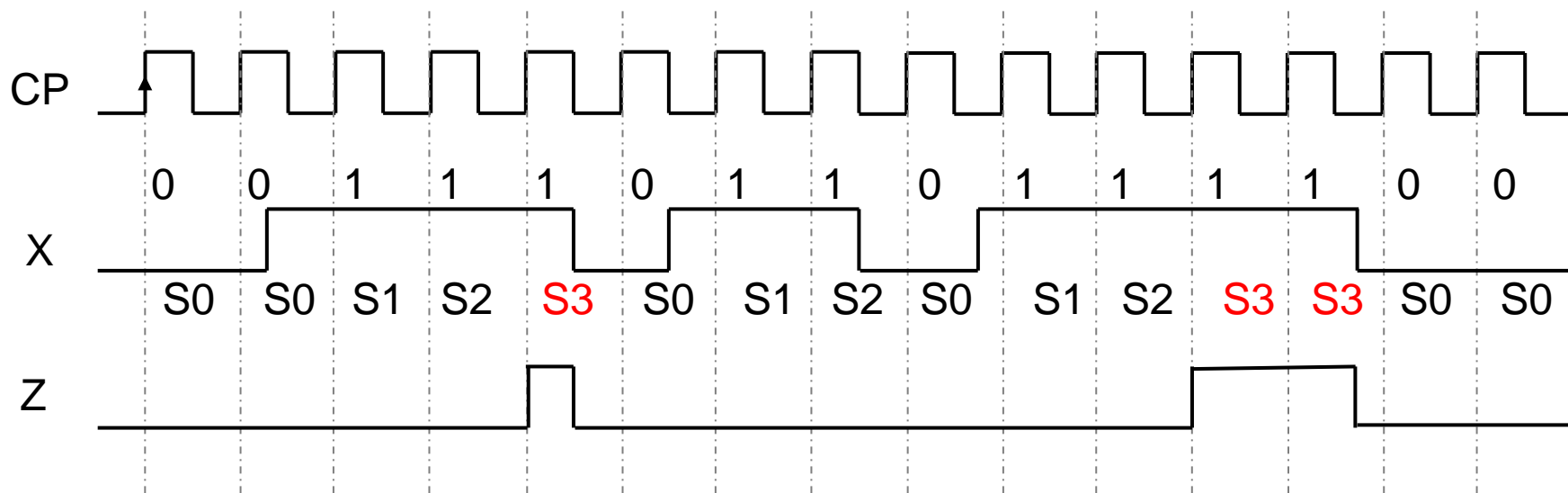
- 给定逻辑功能的要求，求相应的逻辑电路
- 设计的一般步骤
 - 建立原始状态图和原始状态表
 - 状态化简
 - 状态编码
 - 求状态方程和输出方程
 - 检查自启动
 - 选择触发器类型，求激励方程
 - 画出逻辑图

示例1—序列检测器

- 检测“111”序列，当连续输入三个“1”时，输出为“1”，否则输出为“0”

输入： 0 0 1 1 1 0 1 1 0 1 1 1 1 0

输出： 0 0 0 0 1 0 0 0 0 0 0 1 1 0



按Mealy型时序电路分析

(1) 建立原始状态图和原始状态表

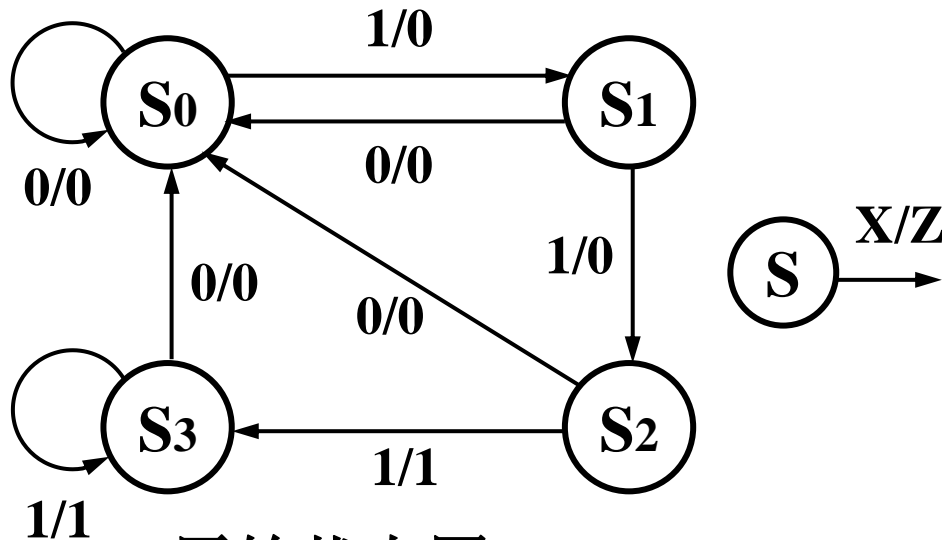
设输入、输出变量分别为 X 和 Z ，定义电路状态

S_0 : 输入 “0”

S_2 : 连续输入 “11”

S_1 : 输入 “1”

S_3 : 连续输入 “111”

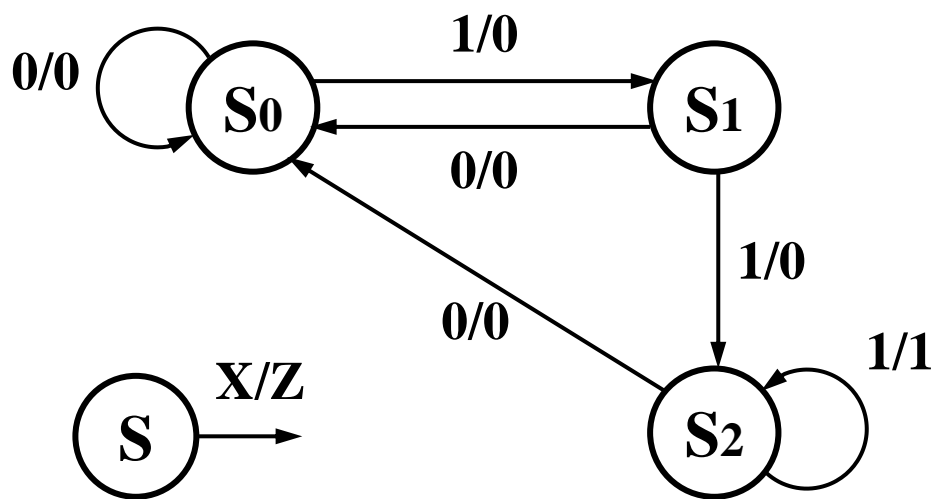


原始状态图

原始状态表

S^n	S^{n+1}/Z	
	$X=0$	$X=1$
S_0	$S_0/0$	$S_1/0$
S_1	$S_0/0$	$S_2/0$
S_2	$S_0/0$	$S_3/1$
S_3	$S_0/0$	$S_3/1$

(2) 状态化简



化简后状态图

化简后状态表

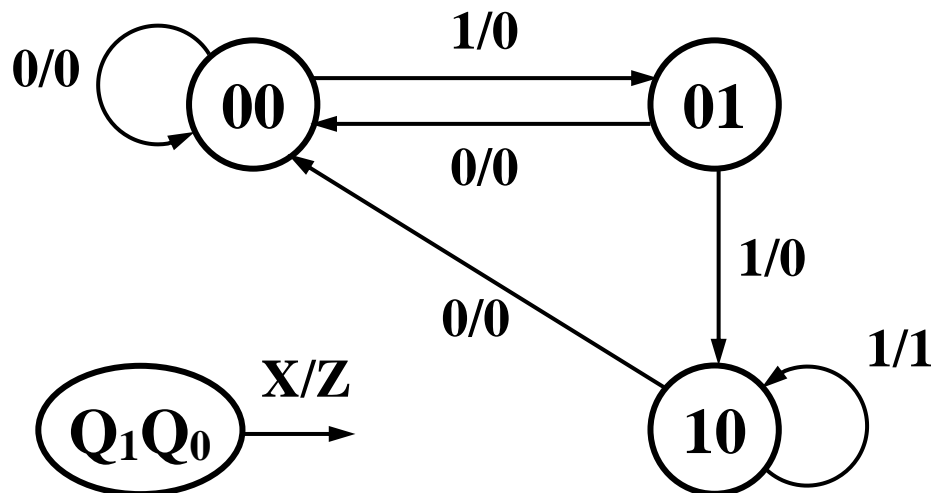
S^n	S^{n+1}/Z	
	$X=0$	$X=1$
S_0	$S_0/0$	$S_1/0$
S_1	$S_0/0$	$S_2/0$
S_2	$S_0/0$	$S_2/1$

(3) 状态编码

采用顺序码，设 $S_0=00$ ， $S_1=01$ ， $S_2=10$

状态表

$Q_1^n Q_0^n$	$Q_1^{n+1} Q_0^{n+1} / Z$	
	X=0	X=1
00	00/0	01/0
01	00/0	10/0
10	00/0	10/1
11	xx/x	xx/x



状态图

(4) 求出状态方程和输出方程

状态表

$Q_1^n Q_0^n$	$Q_1^{n+1} Q_0^{n+1} / Z$	
	$X=0$	$X=1$
00	00/0	01/0
01	00/0	10/0
10	00/0	10/1
11	xx/x	xx/x

$Q_1^n Q_0^n$	X	
	0	1
00	0	0
01	0	1
11	×	×
10	0	1

$Q_1^n Q_0^n$	X	
	0	1
00	0	1
01	0	0
11	×	×
10	0	0

$Q_1^n Q_0^n$	X	
	0	1
00	0	0
01	0	0
11	×	×
10	0	1

$$Q_1^{n+1} = X(Q_0^n + Q_1^n)$$

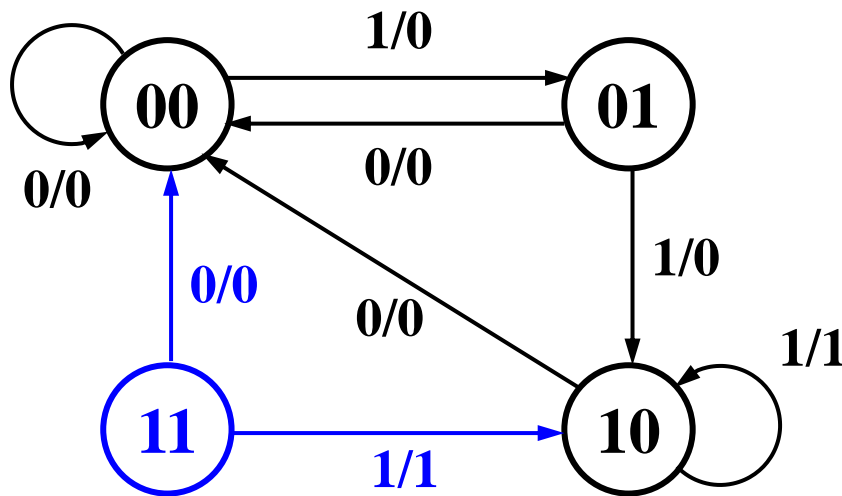
$$Q_0^{n+1} = X \bar{Q}_0^n \bar{Q}_1^n$$

$$Z = X Q_1^n$$

状态表

$Q_1^n Q_0^n$	$Q_1^{n+1} Q_0^{n+1} / Z$	
	X=0	X=1
00	00/0	01/0
01	00/0	10/0
10	00/0	10/1
11	00/0	10/1

具有自启动能力



状态图



(5) 检查自启动

$$Q_1^{n+1} = X(Q_0^n + Q_1^n)$$

$$Q_0^{n+1} = X \overline{Q_0^n} \overline{Q_1^n}$$

$$Z = XQ_1^n$$

(6) 选择触发器，求激励方程和输出方程

$$Q_1^{n+1} = X(Q_0^n + Q_1^n)$$

$$Q_0^{n+1} = X \overline{Q_0}^n \overline{Q_1}^n$$

$$Z = XQ_1^n$$

D触发器



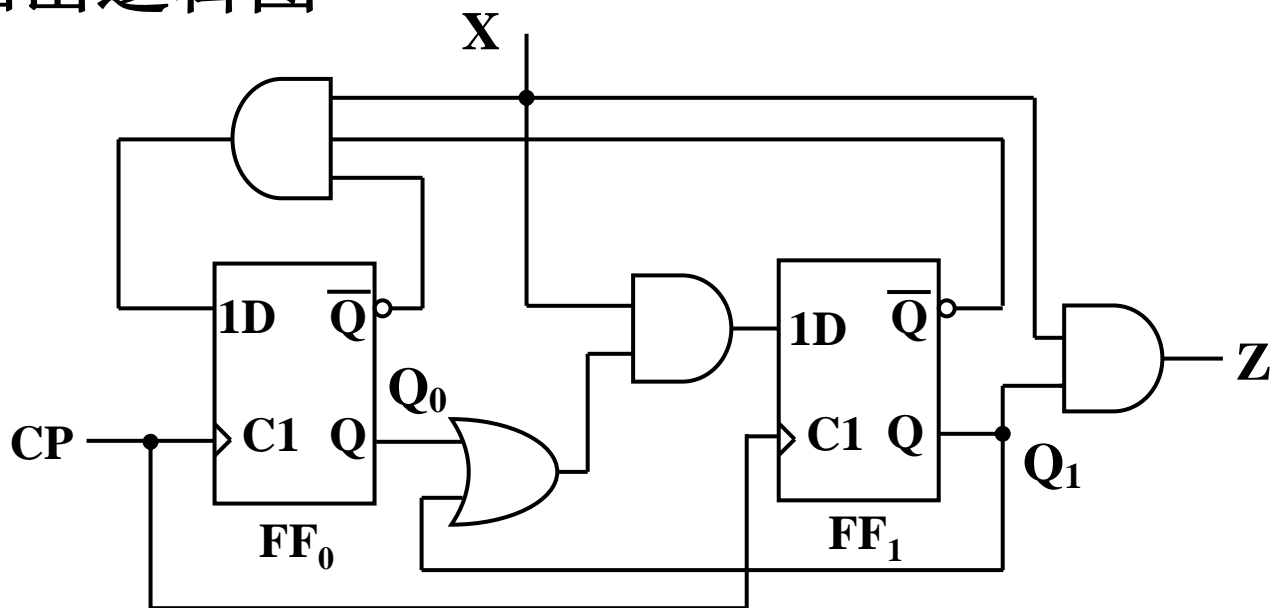
$$Q^{n+1} = D$$

$$D_1 = X(Q_0 + Q_1)$$

$$D_0 = X \overline{Q_0} \overline{Q_1}$$

$$Z = XQ_1$$

(7) 画出逻辑图



设计的一般步骤 (1)

- 建立原始状态图和原始状态表
 - 确定输入/输出变量、电路状态数
 - 定义输入/输出逻辑状态以及每个电路状态的含意
 - 按设计要求画出状态转换图，或列出状态转换表
- 状态化简
 - 求出最简状态图(表)，以便用最少的触发器实现电路
 - 合并等价状态，消去多余状态
 - 等价状态：在相同的输入下有相同的输出，且转换到相同的次态

设计的一般步骤 (2)

- 状态编码：给每个状态赋予不同的二进制代码
 - 根据状态数(M)，确定触发器的数目(n)
 - n的最小值满足： $2^{n-1} < M \leq 2^n$ ，即 $n = \lceil \log_2 M \rceil$
- 常用编码方法
 - 顺序编码：状态从0至M-1编号，将编号用等值的二进制数码表示
 - 循环码：相邻代码只有1位不同
 - 独热(One-hot)码：n=M，任意状态的代码中只有1位为1，其余位都是为0

设计的一般步骤 (3)

- 求状态方程和输出方程
 - 将状态代码代入状态表，得到状态变量和输出变量的真值表
 - 根据真值表，求出简化的状态函数和输出函数
- 检查自启动
 - 画出全部状态图
 - 检查是否存在无效状态之间的循环
 - 若没有，称电路具有自启动(也称自校正)能力
 - 否则，重新定义无关项，以便消除无效循环，并求状态方程和输出方程

设计的一般步骤 (4)

- 选择触发器类型，求激励方程
 - 根据选择触发器的特征方程和待实现的状态方程，求触发器的激励方程

例如，要实现状态方程： $Q_0^{n+1} = Q_0^n + Q_1^n$

– 选用D触发器：特征方程 $Q^{n+1} = D \Rightarrow D_0 = Q_0^n + Q_1^n$

– 选用JK触发器：特征方程 $Q^{n+1} = J\overline{Q}^n + \overline{K}Q^n$

$$\begin{aligned} Q_0^{n+1} &= Q_0^n + Q_1^n = Q_0^n + Q_1^n(Q_0^n + \overline{Q_0^n}) \\ &= Q_0^n + Q_1^n \overline{Q_0^n} \Rightarrow J = Q_1^n, K = 0 \end{aligned}$$

选用T触发器，如何？

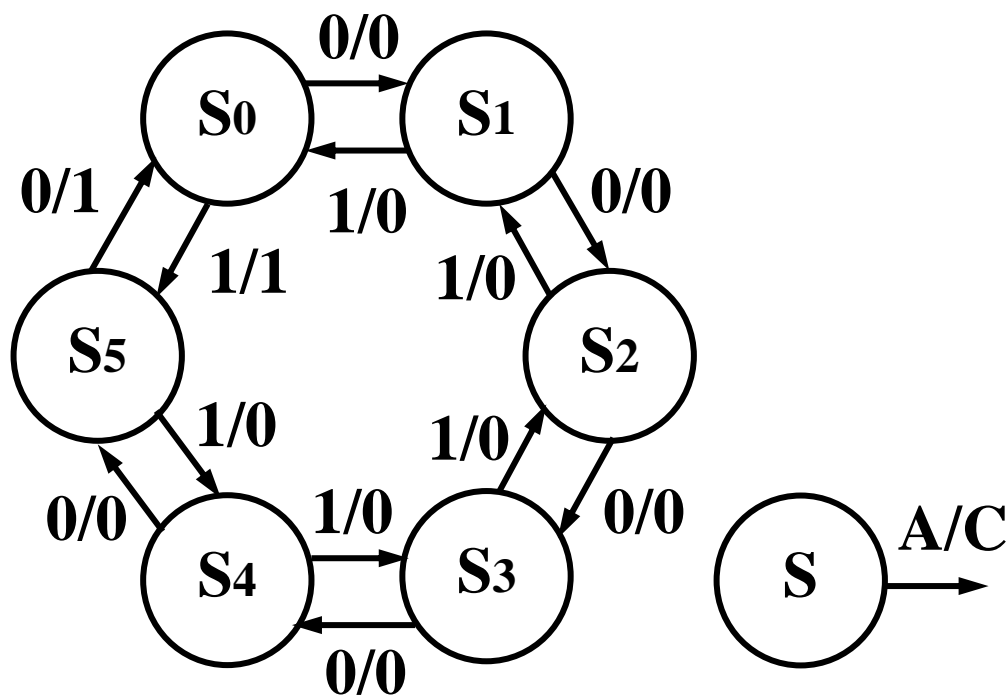
示例2—可逆六进制计数器

- 建立状态图和状态表

- 设A=0加法，A=1减法；C为进位或借位

状态表

S^n	S^{n+1}/C	
	A=0	A=1
S_0	$S_1/0$	$S_5/1$
S_1	$S_2/0$	$S_0/0$
S_2	$S_3/0$	$S_1/0$
S_3	$S_4/0$	$S_2/0$
S_4	$S_5/0$	$S_3/0$
S_5	$S_0/1$	$S_4/0$



状态图

• 状态编码

- 至少需要 $\lceil \log_2 6 \rceil = 3$ 位
- 顺序码：000~101
- 独热码需要6位
- 循环码（如左图）

Q ₁ Q ₀				
Q ₂	00	01	11	10
0	S ₀	S ₁	x	S ₅
1	x	S ₂	S ₃	S ₄

Q ₁ Q ₀				
Q ₂	00	01	11	10
0	S ₀	S ₁	S ₂	S ₃
1	S ₅	x	x	S ₄

状态表

Q ₂ ⁿ Q ₁ ⁿ Q ₀ ⁿ	Q ₂ ⁿ⁺¹ Q ₁ ⁿ⁺¹ Q ₀ ⁿ⁺¹ /C	
	A=0	A=1
0 0 0	001/0	010/1
0 0 1	101/0	000/0
1 0 1	111/0	001/0
1 1 1	110/0	101/0
1 1 0	010/0	111/0
0 1 0	000/1	110/0
0 1 1	xxx/x	xxx/x
1 0 0	xxx/x	xxx/x

- 求状态方程和输出方程

Q_2Q_1	$Q_2^+ \quad Q_0A$			
	00	01	11	10
00	0	0	0	1
01	0	1	x	x
11	0	1	1	1
10	x	x	0	1

$$Q_2^{n+1} = Q_1^n A + Q_0^n \bar{A}$$

Q_2Q_1	$Q_1^+ \quad Q_0A$			
	00	01	11	10
00	0	1	0	0
01	0	1	x	x
11	1	1	0	1
10	x	x	0	1

$$Q_1^{n+1} = Q_2^n \bar{A} + \bar{Q}_0^n A$$

Q_2Q_1	$Q_0^+ \quad Q_0A$			
	00	01	11	10
00	1	0	0	1
01	0	0	x	x
11	0	1	1	0
10	x	x	1	1

$$Q_0^{n+1} = Q_2^n A + \bar{Q}_1^n \bar{A}$$

Q_2Q_1	$C \quad Q_0A$			
	00	01	11	10
00	0	1	0	0
01	1	0	x	x
11	0	0	0	0
10	x	x	0	0

$$C = \bar{Q}_2^n \bar{Q}_0^n (A \oplus Q_1^n)$$

状态表

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1} / C$	
	A=0	A=1
0 0 0	001/0	010/1
0 0 1	101/0	000/0
1 0 1	111/0	001/0
1 1 1	110/0	101/0
1 1 0	010/0	111/0
0 1 0	000/1	110/0
0 1 1	xxx/x	xxx/x
1 0 0	xxx/x	xxx/x

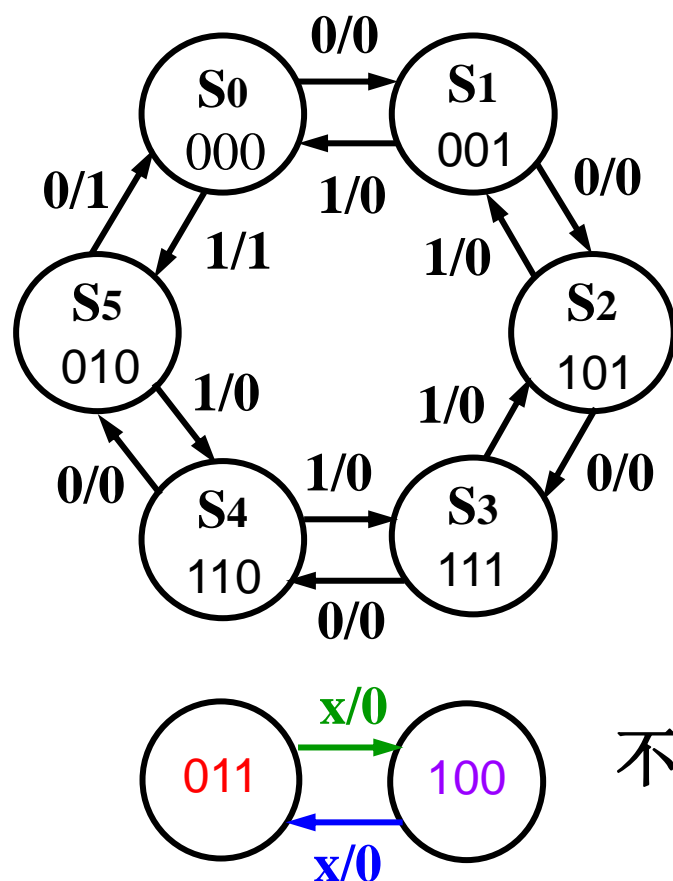
• 检查自启动

$$Q_2^{n+1} = Q_1^n A + Q_0^n \bar{A}$$

$$C = \bar{Q}_2^n \bar{Q}_0^n (A \oplus Q_1^n)$$

$$Q_1^{n+1} = Q_2^n \bar{A} + \bar{Q}_0^n A$$

$$Q_0^{n+1} = Q_2^n A + \bar{Q}_1^n \bar{A}$$



不能自启动!

状态表

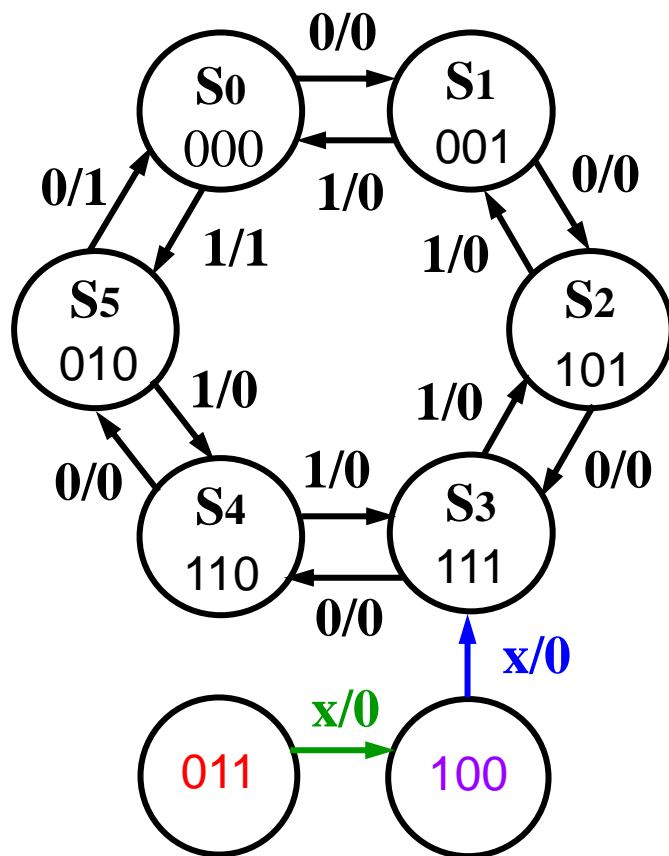
$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1} / C$	
	A=0	A=1
0 0 0	001/0	010/1
0 0 1	101/0	000/0
1 0 1	111/0	001/0
1 1 1	110/0	101/0
1 1 0	010/0	111/0
0 1 0	000/1	110/0
0 1 1	100/0	100/0
1 0 0	011/0	011/0

- 修改设计，使其能够自启动

$$Q_2^{n+1} = Q_1^n A + Q_0^n \bar{A}$$

$$Q_1^{n+1} = Q_2^n \bar{A} + \bar{Q}_0^n A$$

$$Q_0^{n+1} = Q_2^n A + \bar{Q}_1^n \bar{A}$$



状态表

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1} / C$	
	A=0	A=1
0 0 0	001/0	010/1
0 0 1	101/0	000/0
1 0 1	111/0	001/0
1 1 1	110/0	101/0
1 1 0	010/0	111/0
0 1 0	000/1	110/0
0 1 1	100/x	100/x
1 0 0	111/x	111/x
1 0 0	011/x	011/x

- 修改设计，使其能够自启动

这里采用了011->100->111，
也可以采用100->011->101/110

~~$$Q_2^{n+1} = Q_1^n A + Q_0^n \bar{A}$$~~

$$Q_1^{n+1} = Q_2^n \bar{A} + \bar{Q}_0^n A$$

$$Q_0^{n+1} = Q_2^n A + \bar{Q}_1^n \bar{A}$$

~~| | Q_2^+ | | $Q_0 A$ | |
|-----------|---------|----|---------|----|
| $Q_2 Q_1$ | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 1 |
| 01 | 0 | 1 | X | X |
| 11 | 0 | 1 | 1 | 1 |
| 10 | X | X | 0 | 1 |~~

	Q_2^+		$Q_0 A$	
$Q_2 Q_1$	00	01	11	10
00	0	0	0	1
01	0	1	1	1
11	0	1	1	1
10	1	1	0	1

$$Q_2^{n+1} = Q_1^n A + Q_0^n \bar{A} + Q_2^n \bar{Q}_1^n \bar{Q}_0^n$$

状态表

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1} / C$	
	A=0	A=1
0 0 0	001/0	010/1
0 0 1	101/0	000/0
1 0 1	111/0	001/0
1 1 1	110/0	101/0
1 1 0	010/0	111/0
0 1 0	000/1	110/0
0 1 1	100/x	100/x
1 0 0	111/x	111/x
1 0 0	011/x	011/x

- 修改设计，使其能够自启动

$$Q_2^{n+1} = Q_1^n A + Q_0^n \bar{A} + Q_2^n \bar{Q}_1^n \bar{Q}_0^n$$

$$Q_1^{n+1} = Q_2^n \bar{A} + \bar{Q}_0^n A \quad \text{保持不变}$$

$$Q_0^{n+1} = Q_2^n A + \bar{Q}_1^n \bar{A}$$

	Q_1^{n+1}		$Q_0 A$	
$Q_2 Q_1$	00	01	11	10
00	0	1	0	0
01	0	1	x	x
11	1	1	0	1
10	x	x	0	1

	Q_1^{n+1}		$Q_0 A$	
$Q_2 Q_1$	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	1	1	0	1
10	1	1	0	1

$Q_1^{n+1} = Q_2^n \bar{A} + \bar{Q}_0^n A$

状态表

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1} / C$	
	A=0	A=1
0 0 0	001/0	010/1
0 0 1	101/0	000/0
1 0 1	111/0	001/0
1 1 1	110/0	101/0
1 1 0	010/0	111/0
0 1 0	000/1	110/0
0 1 1	100/x	100/x
1 0 0	111/x	111/x
1 0 0	011/x	011/x

- 修改设计，使其能够自启动

$$Q_2^{n+1} = Q_1^n A + Q_0^n \bar{A} + Q_2^n \bar{Q}_1^n \bar{Q}_0^n$$

$$Q_1^{n+1} = Q_2^n \bar{A} + \bar{Q}_0^n A \quad \text{保持不变}$$

$$Q_0^{n+1} = Q_2^n A + \bar{Q}_1^n \bar{A} \quad \text{保持不变}$$

	Q_0^+		$Q_0 A$	
$Q_2 Q_1$	00	01	11	10
00	1	0	0	1
01	0	0	x	x
11	0	1	1	0
10	x	x	1	1

	Q_0^+		$Q_0 A$	
$Q_2 Q_1$	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	1	1	0
10	1	1	1	1

$$Q_0^{n+1} = Q_2^n A + \bar{Q}_1^n \bar{A}$$

状态表

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1} / C$	
	A=0	A=1
0 0 0	001/0	010/1
0 0 1	101/0	000/0
1 0 1	111/0	001/0
1 1 1	110/0	101/0
1 1 0	010/0	111/0
0 1 0	000/1	110/0
0 1 1	100/x	100/x
1 0 0	111/x	111/x
1 0 0	011/x	011/x

- 求出对应的输出

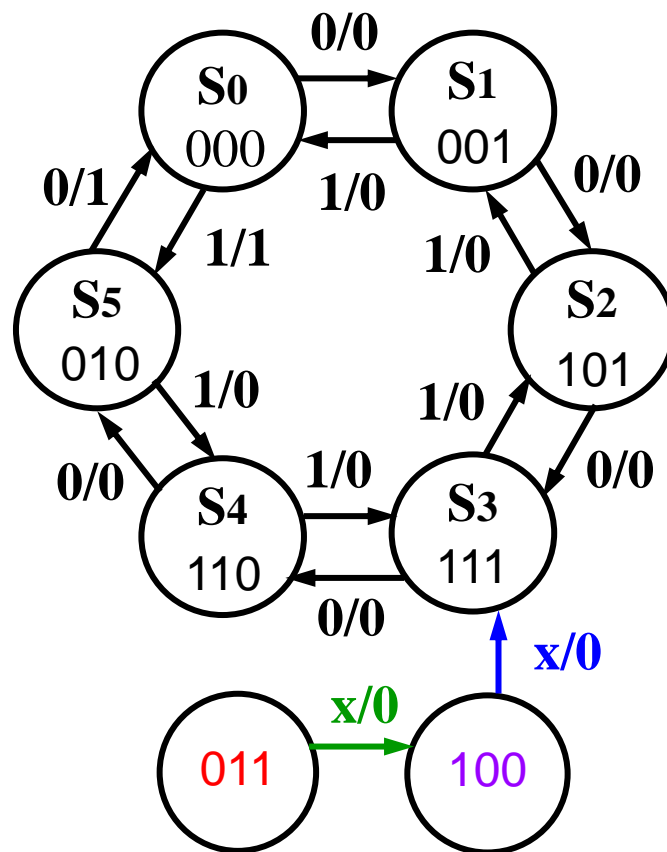
		C		Q ₀ A	
		00	01	11	10
Q ₂ Q ₁	00	0	1	0	0
	01	1	0	x	x
	11	0	0	0	0
	10	x	x	0	0

$$C = \overline{Q_2}^n \overline{Q_0}^n (A \oplus Q_1^n)$$

或(不严格条件下):

$$\mathbf{C} = \overline{\mathbf{Q}_2}^n \mathbf{Q}_1^n \overline{\mathbf{A}} + \mathbf{Q}_2^n \overline{\mathbf{Q}_0}^n \mathbf{A}$$

在011和100两种状态下的C输出虽然已经固定，但是我们不用太过关心，因为这两个状态最迟在上电后两个时钟周期就进入到正常S₀-S₅中



- 选用JK触发器，求出激励方程

$$Q_2^{n+1} = Q_1^n A + Q_0^n \bar{A} + Q_2^n \bar{Q}_1^n \bar{Q}_0^n$$

$$Q_1^{n+1} = Q_2^n \bar{A} + \bar{Q}_0^n A$$

$$Q_0^{n+1} = Q_2^n A + \bar{Q}_1^n \bar{A}$$

JK触发器特征方程：

$$Q^{n+1} = J \bar{Q}^n + \bar{K} Q^n$$

$$\begin{aligned} Q_0^{n+1} &= Q_2^n A + \bar{Q}_1^n \bar{A} \\ &= (Q_2^n A + \bar{Q}_1^n \bar{A}) \bar{Q}_0^n + (Q_2^n A + \bar{Q}_1^n \bar{A}) Q_0^n \end{aligned}$$

$$\Rightarrow J_0 = Q_2^n A + \bar{Q}_1^n \bar{A} \quad K_0 = \bar{J}_0$$

$$Q_1^{n+1} = Q_2^n \bar{A} + \bar{Q}_0^n A \Rightarrow J_1 = Q_2^n \bar{A} + \bar{Q}_0^n A \quad K_1 = \bar{J}_1$$

$$Q_2^{n+1} = Q_1^n A + Q_0^n \bar{A} + Q_2^n \bar{Q}_1^n \bar{Q}_0^n$$

$$\Rightarrow J_2 = Q_1^n A + Q_0^n \bar{A} \quad K_2 = \overline{J_2 + \bar{Q}_1^n \bar{Q}_0^n}$$

- 画出逻辑图（略）

求激励方程的另一种方法

- 利用激励表和状态表求各触发器的激励方程

D触发器激励表

Q^n	Q^{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

JK触发器激励表

Q^n	Q^{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

T触发器激励表

Q^n	Q^{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

RS触发器激励表

Q^n	Q^{n+1}	R	S
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

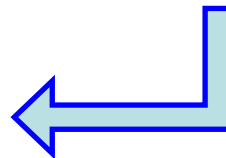
激励表和状态表求激励方程

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$	
	A=0	A=1
000	001	010
001	101	000
101	101	001
111	110	101
110	010	111
010	000	110
011	100	100
100	111	111

状态表

JK触发器激励表

Q^n	Q^{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0



由Q的变化情况，结合激励表，推导出各自的变化条件

画出卡诺图求JK (J_2K_2)

条件填入 $Q_2^n Q_1^n Q_0^n A$

J_2	$Q_0 A$			
$Q_2 Q_1$	00	01	11	10
00	0	0	0	1
01	0	1	1	1
11	x	x	x	x
10	x	x	x	x

K_2	$Q_0 A$			
$Q_2 Q_1$	00	01	11	10
00	x	x	x	x
01	x	x	x	x
11	1	0	0	0
10	0	0	1	0

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$		$J_2 K_2 J_1 K_1 J_0 K_0$	
	A=0	A=1	A=0	A=1
000	001	010	0x0x1x	0x1x0x
001	101	000	1x0xx0	0x0xx1
101	101	001	x01xx0	x10xx0
111	110	101	x0x0x1	x0x1x0
110	010	111	x1x00x	x0x01x
010	000	110	0xx10x	1xx00x
011	100	100	1xx1x1	1xx1x1
100	111	111	x01x1x	x01x1x

$$J_2 = Q_1^n A + Q_0^n \bar{A}$$

$$K_2 = Q_1^n \bar{Q}_0^n \bar{A} + \bar{Q}_1^n Q_0^n A$$

和第一种方法推导的结果做对比：

J_2	$Q_0 A$			
$Q_2 Q_1$	00	01	11	10
00	0	0	0	1
01	0	1	1	1
11	0	1	1	1
10	0	0	0	1

K_2	$Q_0 A$			
$Q_2 Q_1$	00	01	11	10
00	0	0	1	0
01	1	0	0	0
11	1	0	0	0
10	0	0	1	0

$$J_2 = Q_1^n A + Q_0^n \bar{A}$$

$$K_2 = \overline{J_2 + \bar{Q}_1^n \bar{Q}_0^n}$$

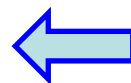
达到功能一致，
表达式一致

画出卡诺图求JK ($J_1 K_1$)

条件填入 $Q_2^n Q_1^n Q_0^n A$

J_1	$Q_0 A$			
$Q_2 Q_1$	00	01	11	10
00	0	1	0	0
01	x	x	x	x
11	x	x	x	x
10	1	1	0	1

K_1	$Q_0 A$			
$Q_2 Q_1$	00	01	11	10
00	x	x	x	x
01	1	0	1	1
11	0	0	1	0
10	x	x	x	x



$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$		$J_2 K_2 J_1 K_1 J_0 K_0$	
	A=0	A=1	A=0	A=1
000	001	010	0x0x1x	0x1x0x
001	101	000	1x0xx0	0x0xx1
101	101	001	x01xx0	x10xx0
111	110	101	x0x0x1	x0x1x0
110	010	111	x1x00x	x0x01x
010	000	110	0xx10x	1xx00x
011	100	100	1xx1x1	1xx1x1
100	111	111	x01x1x	x01x1x

$$J_1 = Q_2^n \bar{A} + \bar{Q}_0^n A$$

$$K_1 = \bar{Q}_2^n \bar{A} + Q_0^n A$$

和第一种方法推导的结果做对比：

J_1	$Q_0 A$			
$Q_2 Q_1$	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	1	1	0	1
10	1	1	0	1

K_1	$Q_0 A$			
$Q_2 Q_1$	00	01	11	10
00	1	0	1	1
01	1	0	1	1
11	0	0	1	0
10	0	0	1	0

$$J_1 = Q_2^n \bar{A} + \bar{Q}_0^n A$$

$$K_1 = \bar{J}_1$$

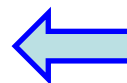
达到功能一致，
表达式一致

画出卡诺图求JK (J_0K_0)

条件填入 $Q_2^n Q_1^n Q_0^n A$

J_0	$Q_0 A$			
$Q_2 Q_1$	00	01	11	10
00	1	0	x	x
01	0	0	x	x
11	0	1	x	x
10	1	1	x	x

K_0	$Q_0 A$			
$Q_2 Q_1$	00	01	11	10
00	x	x	1	0
01	x	x	1	1
11	x	x	0	1
10	x	x	0	0



$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$		$J_2 K_2 \ J_1 K_1 \ J_0 K_0$	
	A=0	A=1	A=0	A=1
000	001	010	0x0x1x	0x1x0x
001	101	000	1x0xx0	0x0xx1
101	101	001	x01xx0	x10xx0
111	110	101	x0x0x1	x0x1x0
110	010	111	x1x00x	x0x01x
010	000	110	0xx10x	1xx00x
011	100	100	1xx1x1	1xx1x1
100	111	111	x01x1x	x01x1x

$$J_0 = Q_2^n A + \overline{Q_1^n} \overline{A}$$

$$K_0 = Q_1^n \overline{A} + \overline{Q_2^n} A$$

和第一种方法推导的结果做对比：

J_0	$Q_0 A$			
$Q_2 Q_1$	00	01	11	10
00	1	0	x	x
01	0	0	x	x
11	0	1	x	x
10	1	1	x	x

K_0	$Q_0 A$			
$Q_2 Q_1$	00	01	11	10
00	x	x	1	0
01	x	x	1	1
11	x	x	0	1
10	x	x	0	0

$$J_0 = Q_2^n A + \overline{Q_1^n} \overline{A}$$

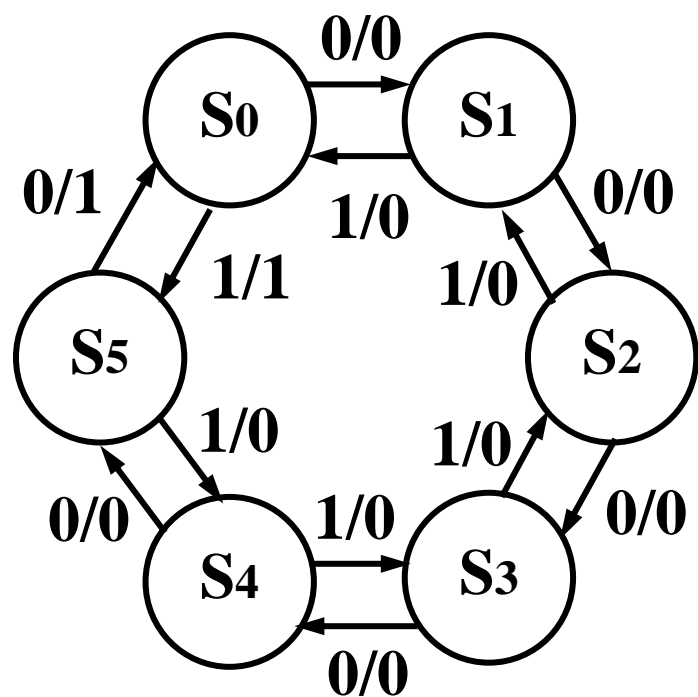
$$K_0 = \overline{J_0}$$

达到功能一致，
表达式一致

示例2—可逆六进制计数器

• 状态编码

- 二进制码：000 ~ 101
- T触发器实现



状态图

T触发器激励表

Q^n	Q^{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$		$T_2 T_1 T_0$	
	A=0	A=1	A=0	A=1
S0:000	001	101	001	101
S1:001	010	000	011	001
S2:010	011	001	001	011
S3:011	100	010	111	001
S4:100	101	011	001	111
S5:101	000	100	101	001
S6:110	xxx	xxx	xxx	xxx
S7:111	xxx	xxx	xxx	xxx

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$		$T_2 T_1 T_0$	
	A=0	A=1	A=0	A=1
S0:000	001	101	001	101
S1:001	010	000	011	001
S2:010	011	001	001	011
S3:011	100	010	111	001
S4:100	101	011	001	111
S5:101	000	100	101	001
S6:110	xxx	xxx	xxx	xxx
S7:111	xxx	xxx	xxx	xxx

画出卡诺图求T

$Q_2 Q_1$	$T_2 \quad Q_0 A$			
	00	01	11	10
00	0	1	0	0
01	0	0	0	1
11	x	x	x	x → 1
10	0	1	0	1

$$T_2 = \overline{Q_1}^n \overline{Q_0}^n A + (Q_2^n + Q_1^n) \overline{A}$$

$Q_2 Q_1$	$T_1 \quad Q_0 A$			
	00	01	11	10
00	0	0	0	1
01	0	1	0	1
11	x	x → 1	x	x
10	0	1	0	0

$$T_1 = \overline{Q_2}^n Q_0^n \overline{A} + (Q_2^n + Q_1^n) \overline{Q_0}^n A$$

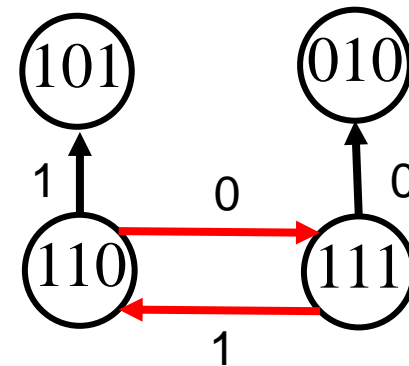
$Q_2 Q_1$	$T_1 \quad Q_0 A$			
	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	x	x	x	x → 1
10	1	1	1	1

$$T_0 = 1$$

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$		$T_2 T_1 T_0$	
	A=0	A=1	A=0	A=1
S0:000	001	101	001	101
S1:001	010	000	011	001
S2:010	011	001	001	011
S3:011	100	010	111	001
S4:100	101	011	001	111
S5:101	000	100	101	001
S6:110	111	101	001	011
S7:111	010	110	101	001

验证自启动:

先写出 $T_2 T_1 T_0$, (圈中 $x \rightarrow 1$, 否则 $x \rightarrow 0$)
再根据 $Q_2^n Q_1^n Q_0^n$ 求出 $Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$
可能上电后:



$Q_2 Q_1$	$T_2 \quad Q_0 A$			
	00	01	11	10
00	0	1	0	0
01	0	0	0	1
11	x	x	x	x $\rightarrow 1$
10	0	1	0	1

$$T_2 = \overline{Q_1^n} \overline{Q_0^n} A + (Q_2^n + Q_1^n) \overline{A}$$

$Q_2 Q_1$	$T_1 \quad Q_0 A$			
	00	01	11	10
00	0	0	0	1
01	0	1	0	1
11	x	x $\rightarrow 1$	x	x
10	0	1	0	0

$$T_1 = \overline{Q_2^n} \overline{Q_0^n} A + (Q_2^n + Q_1^n) \overline{Q_0^n} A$$

$Q_2 Q_1$	$T_0 \quad Q_0 A$			
	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	x	x	x	x $\rightarrow 1$
10	1	1	1	1

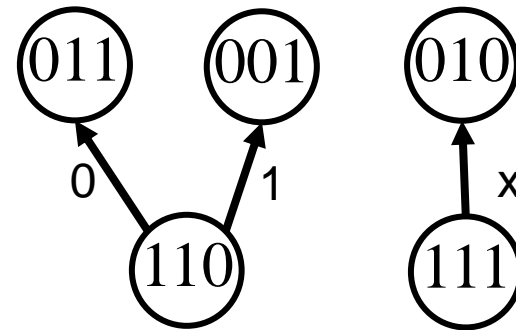
$$T_0 = 1$$

$Q_2^n Q_1^n Q_0^n$	$Q_2^{n+1} Q_1^{n+1} Q_0^{n+1}$		$T_2 T_1 T_0$	
	A=0	A=1	A=0	A=1
S0:000	001	101	001	101
S1:001	010	000	011	001
S2:010	011	001	001	011
S3:011	100	010	111	001
S4:100	101	011	001	111
S5:101	000	100	101	001
S6:110	011	001	101	111
S7:111	010	010	101	101

验证自启动:

$$\text{扩展 } T_2 \quad T_2 = \overline{Q_1^n} \overline{Q_0^n} A + (Q_2^n + Q_1^n) \overline{A} + Q_2^n Q_1^n$$

求出输出C, 画出逻辑图 (略)



$Q_2 Q_1$	$T_2 \quad Q_0 A$			
	00	01	11	10
00	0	1	0	0
01	0	0	0	1
11	x	x	x	x
10	0	1	0	1

$Q_2 Q_1$	$T_1 \quad Q_0 A$			
	00	01	11	10
00	0	0	0	1
01	0	1	0	1
11	x	x	x	x
10	0	1	0	0

$Q_2 Q_1$	$T_0 \quad Q_0 A$			
	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	x	x	x	x
10	1	1	1	1

$$T_1 = \overline{Q_2^n} Q_0^n \overline{A} + (Q_2^n + Q_1^n) \overline{Q_0^n} A$$

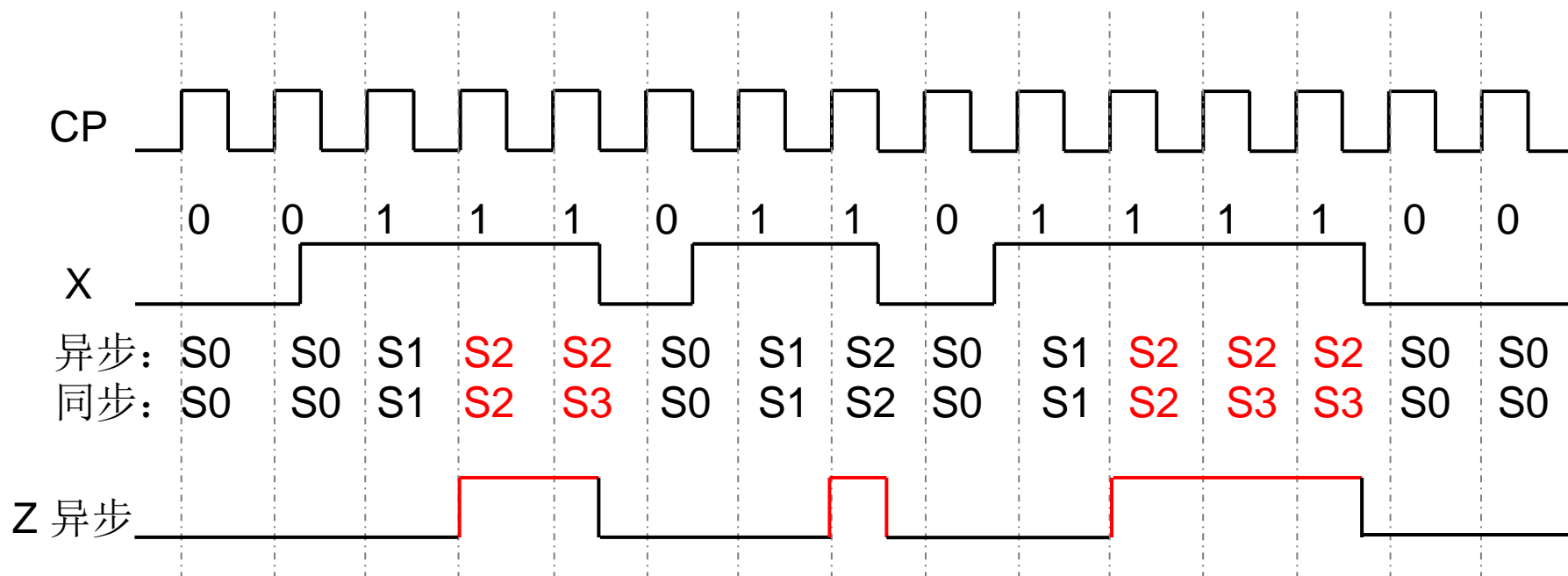
$$T_0 = 1$$

作业

- 电子技术基础-数字部分
- **P355-356**（同步逻辑设计）：
 - 6.3.2, 6.3.4, 6.3.5, 6.3.6, 6.3.7

The End

彩蛋时间：Mealy实现 vs Moore实现



(1) 建立原始状态图和原始状态表

设输入、输出变量分别为 X 和 Z ，定义电路状态

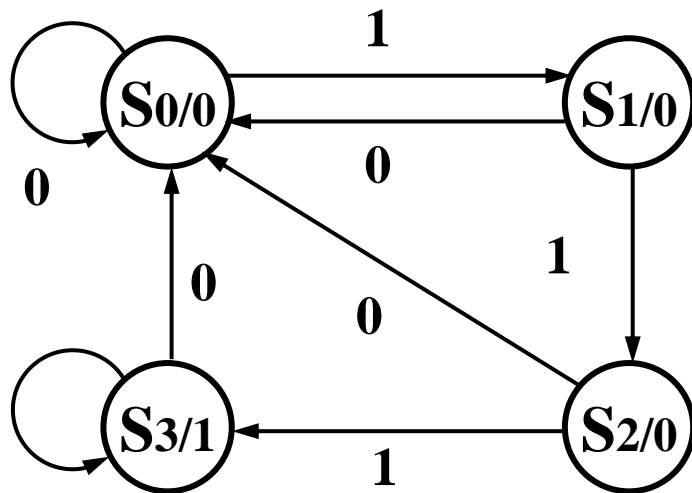
S_0 : 输入 “0”

S_2 : 连续输入 “11”

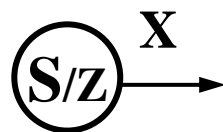
S_1 : 输入 “1”

S_3 : 连续输入 “111”

原始状态表



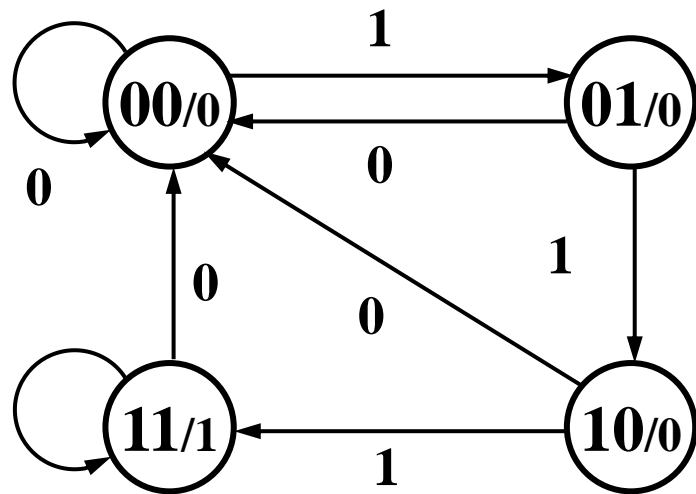
原始状态图



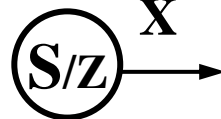
S^n	S^{n+1}		Z
	$X=0$	$X=1$	
S_0	S_0	S_1	0
S_1	S_0	S_2	0
S_2	S_0	S_3	0
S_3	S_0	S_3	1

(2) 状态化简：已经是最简

(3) 状态编码：顺序码



状态图



状态表

S^n	S^{n+1}		Z
	$X=0$	$X=1$	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

(4) 求出状态方程和输出方程
 $S=[Q_1, Q_0]$

状态表			
S^n	S^{n+1}		Z
	$X=0$	$X=1$	
00	00	01	0
01	00	10	0
10	00	11	0
11	00	11	1

Q_1^{n+1} $Q_1^n Q_0^n$	X	
	0	1
00	0	0
01	0	1
11	0	1
10	0	1

Q_0^{n+1} $Q_1^n Q_0^n$	X	
	0	1
00	0	1
01	0	0
11	0	1
10	0	1

Z $Q_1^n Q_0^n$		
	0	1
00	0	
01	0	
11	1	
10	0	

$$Q_1^{n+1} = X(Q_0^n + Q_1^n)$$

$$Q_0^{n+1} = X(\bar{Q}_0^n + Q_1^n)$$

$$Z = Q_0^n Q_1^n$$

(6) 选择触发器，求激励方程

$$Q_1^{n+1} = X(Q_0^n + Q_1^n)$$

$$Q_0^{n+1} = X(\bar{Q}_0^n + Q_1^n)$$

$$Z = Q_0 Q_1$$

D触发器



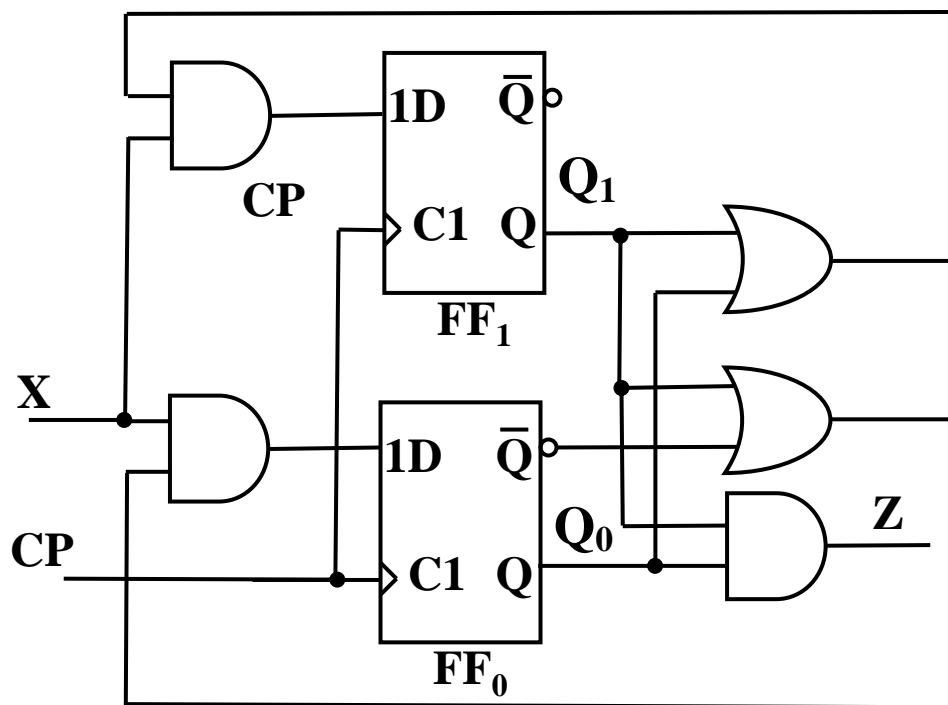
$$Q^{n+1} = D$$

$$D_1 = X(Q_0 + Q_1)$$

$$D_0 = X(\bar{Q}_0 + Q_1)$$

$$Z = Q_0 Q_1$$

(7) 画出逻辑图



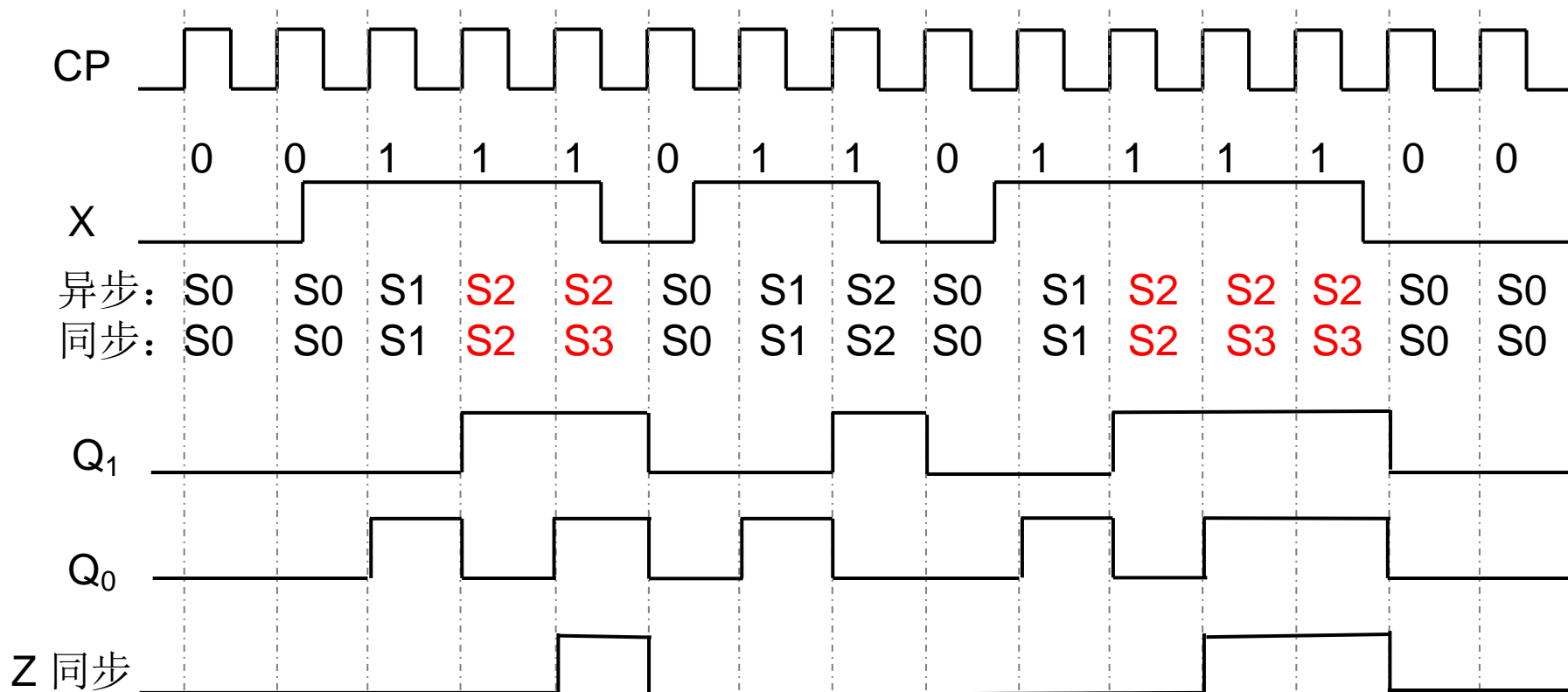
检查时序图

无误！

$$Q_1^{n+1} = X(Q_0^n + Q_1^n)$$

$$Q_0^{n+1} = X(\overline{Q_0^n} + Q_1^n)$$

$$Z = Q_0 Q_1$$



- Moore型，仅在时钟边沿考察
- 输出与时钟严格同步，至少持续一个周期，~~不会产生“毛刺”~~