

数字逻辑设计及应用

General Number Systems

- 10进制: decimal
- 2进制: binary
- 8进制: octal
- 16进制: hexadecimal

1	2	3	4	5	6	7	8	9	10
2	4	8	16	32	64	128	256	512	1024

变位看小数点位置，小数点左边的从小数点向左数n位变1位，小数点右边的从小数点向右数n位变1位

radix: **基数** 通式: $d_{p-1}d_{p-2}\cdots d_1d_0 \cdot d_{-1}d_{-2}\cdots d_{-n}$ $D = \sum_{i=-n}^{p-1} d_i \cdot r^i$

- Number-System Conversion进制变换
- 整数部分除的余数下方为高位；小数部分乘的整部分上方为高位；
- 小数部分乘以 r(radix)，直到小数部分为 0.0 或所需的精度(一般: $x^{-n} < 0.01$)

二进制的原补反码

Signed-magnitude原码

标识: $(X)_{sm}$ $X > 0$, 前面加0, $X < 0$, 前面加 Sign bit 符号位, **未带标识符前n-1位, 带标识符后一共n位**

$$(X)_{sm} = \begin{cases} 0 X, & X = \text{Positive} \\ 1 X, & X = \text{Negative} \end{cases}$$

范围: $-(2^{n-1} - 1) \sim +(2^{n-1} - 1)$.

正0和负0都是0 (10000000, 00000000) 所以0不能直接表示

Ones' Complement反码

标识: $(X)_{1s'c}$ **正数的反码与原码一致**；负数的反码是对原码按位取反(每位d变为 $r - 1 - d$)，只是**最高位(符号位)不变**。

$$(X)_{1s'c} = \begin{cases} X & X = \text{Positive} \\ (2^n - 1) + X & X = \text{Negative} \end{cases}$$

范围: $-(2^{n-1} - 1) \sim +(2^{n-1} - 1)$.

正0和负0都是0 (10000000, 00000000) 所以0不能直接表示

Two's Complement补码

标识 $(X)_{2s'c}$ **正数的补码与原码一致**；负数的补码是该数的**反码加1**（带小数则在小数的末位加1）

$$(X)_{2s'c} = \begin{cases} X & X = \text{Positive} \\ 2^n + X & X = \text{Negative} \end{cases} \pmod{2^n}$$

范围: **$-2^{n-1} \sim +(2^{n-1} - 1)$** 负0为 -2^{n-1}

正0就是0 (00000000) 所以0可以表示

格雷码的特点及优势： 相邻两个数值之间只有一位发生变化，提高了系统的抗干扰能力，而且在计数时，各个输出的门电路翻转次数要远远小于二进制计数器，从而可以大幅度的降低系统的功耗

Parity Code奇偶校验码（产生）

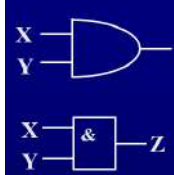
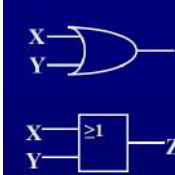
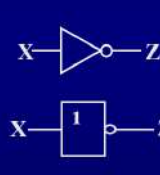
本质：加上校验码之后形成奇数或偶数个1 ***-parity transceiver system （产生检验位电路，要多一位）

Odd-parity 奇校验 原码有奇数个1前面加0，偶数个1前面加1

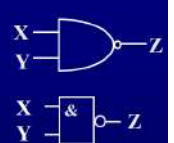
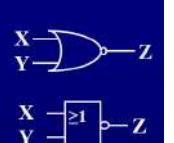
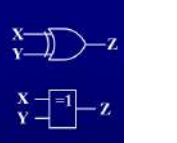
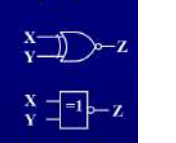
Even-parity 偶校验 原码有奇数个1前面加1，偶数个1前面加0（0算偶数）

Digital Circuits

正逻辑(positive logic)，开关代数的基本概念之一，与门和或门的概念是相对，用1表示高电位，用0表示低电位；

AND Gate	OR Gate	NOT Gate (Inverter)																																				
<div><div>Logic symbol</div><div></div><div>Truth table</div><table><tr><th>X</th><th>Y</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></div>	X	Y	Z	0	0	0	0	1	0	1	0	0	1	1	1	<div><div>Logic symbol</div><div></div><div>Truth table</div><table><tr><th>X</th><th>Y</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></div>	X	Y	Z	0	0	0	0	1	1	1	0	1	1	1	1	<div><div>Logic symbol</div><div></div><div>Truth table</div><table><tr><th>X</th><th>Z</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table></div>	X	Z	0	1	1	0
X	Y	Z																																				
0	0	0																																				
0	1	0																																				
1	0	0																																				
1	1	1																																				
X	Y	Z																																				
0	0	0																																				
0	1	1																																				
1	0	1																																				
1	1	1																																				
X	Z																																					
0	1																																					
1	0																																					
$Z = X \cdot Y$	$Z = X + Y$	$Z = X'$																																				

XOR 的对偶为 XNOR

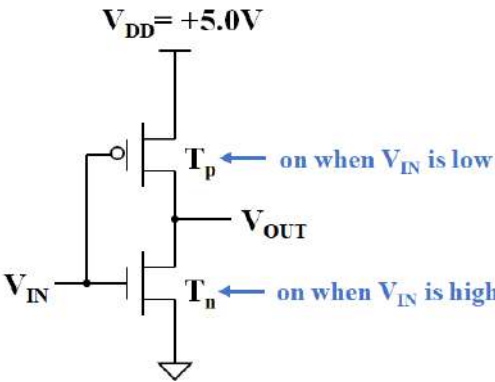
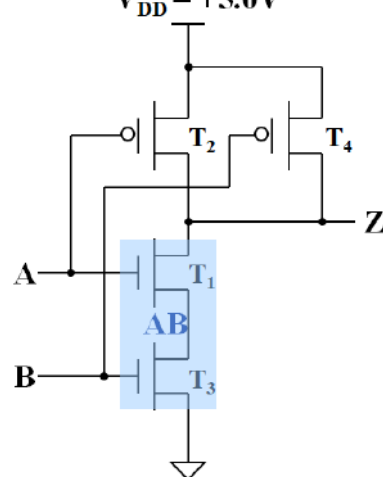
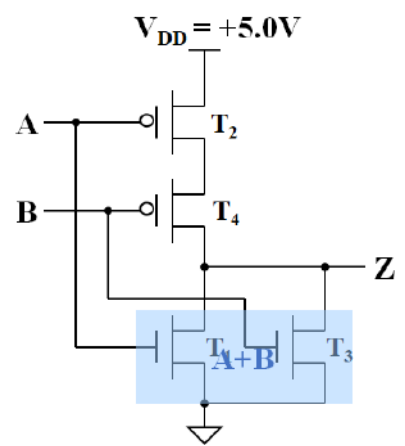
NAND Gate	NOR Gate	XOR(Exclusive OR) Gate	XNOR(Exclusive NOR) Gate																																																												
<div><div>Truth table</div><table><tr><th>X</th><th>Y</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></div> <div><div>Logic symbol</div><div></div></div>	X	Y	Z	0	0	1	0	1	1	1	0	1	1	1	0	<div><div>Truth Table</div><table><tr><th>X</th><th>Y</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></div> <div><div>Logic Symbol</div><div></div></div>	X	Y	Z	0	0	1	0	1	0	1	0	0	1	1	0	<div><div>Truth Table</div><table><tr><th>X</th><th>Y</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></div> <div><div>Logic Symbol</div><div></div></div>	X	Y	Z	0	0	0	0	1	1	1	0	1	1	1	0	<div><div>Truth Table</div><table><tr><th>X</th><th>Y</th><th>Z</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></div> <div><div>Logic Symbol</div><div></div></div>	X	Y	Z	0	0	1	0	1	0	1	0	0	1	1	1
X	Y	Z																																																													
0	0	1																																																													
0	1	1																																																													
1	0	1																																																													
1	1	0																																																													
X	Y	Z																																																													
0	0	1																																																													
0	1	0																																																													
1	0	0																																																													
1	1	0																																																													
X	Y	Z																																																													
0	0	0																																																													
0	1	1																																																													
1	0	1																																																													
1	1	0																																																													
X	Y	Z																																																													
0	0	1																																																													
0	1	0																																																													
1	0	0																																																													
1	1	1																																																													
$Z = (X \cdot Y)'$	$Z = (X + Y)'$	$Z = X \oplus Y = X'Y + XY'$	$Z = X \odot Y = XY + X'Y' = (X \oplus Y)'$																																																												

CMOS门电路

MOS管g端带反向为p型，不带为n型

在互补CMOS电路中，与非门是PMOS管并联，NMOS管串联，而或非门正好相反，NMOS导通电阻小，所以NAND gate 比NOR gate 快

与关系n串p并，或关系n并p串；（主要看高电平下方是否导通接地）

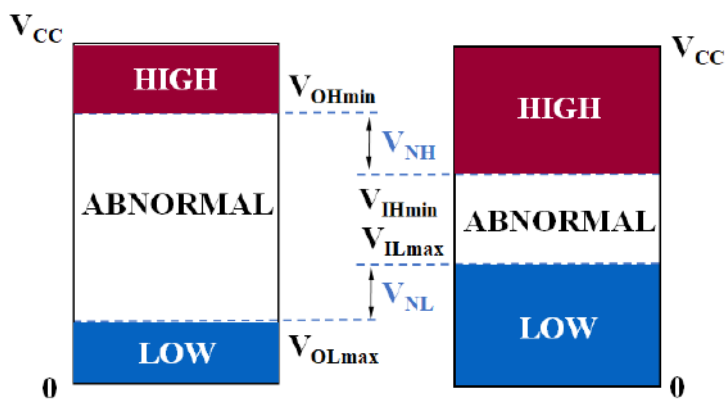
CMOS Inverter 反相器	CMOS NAND Gates 与非门	CMOS NOR Gates 或非门
<div></div>	<div></div>	<div></div>

Fanin扇入

门电路所具有的输入端的数目，称为该逻辑系列的扇入

一个n输入门就具有n个串联和n个并联晶体管

V_{NH}, V_{NL} 为Noise Margins（噪声容限）
 $V_{NH} = V_{OHmin} - V_{IHmin}$
 $V_{NL} = V_{ILmax} - V_{OLmax}$



Fanout 散出

两个器件连接，前一器件的out为后一器件的in

HIGH - state fanout $N_H = \left| \frac{I_{OHmax}}{I_{IHmax}} \right|$

LOW - state fanout $N_L = \left| \frac{I_{OLmax}}{I_{ILmax}} \right|$

Overall fanout $N = \min(N_H, N_L)$

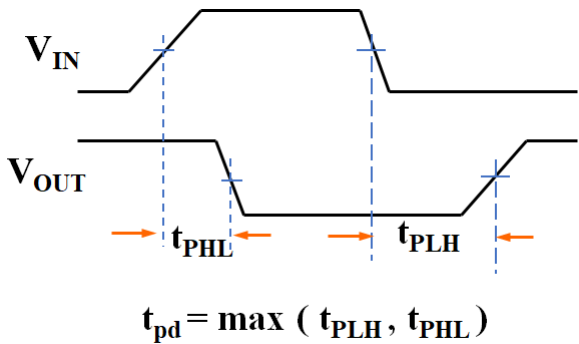
Excess driving capability 剩余驱动能力: $\begin{cases} \text{high : } |I_{OHmax}| - N |I_{IHmax}| \\ \text{low : } |I_{OLmax}| - N |I_{ILmax}| \end{cases}$

传播延迟propagation delay

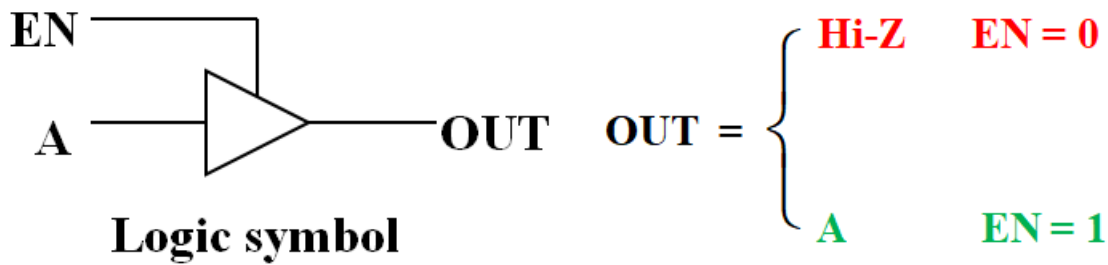
一般写为 t_{tp}

Rise time t_r 上升时间 $t_r \approx 4R_{p(on)}C_L$

Fall time t_f 下降时间 $t_f \approx 4R_{n(on)}C_L$



Three-State Output



The output states of an open-drain gate without the pull-up resistor are Hi-z and Low

Hi-Z: high-impedance state（高阻抗状态）

To implement a wired AND function, (open-drain) gates should be used

Open-drain CMOS NAND gate with pull-up resistor: output are H and L(when A,B=L)

Combinational Logic Design Principles

常用的与或关系式：

$X + 0 = X \quad X \cdot 0 = 0 \quad X + 1 = 1 \quad X \cdot 1 = X \quad X \cdot X = X \quad X \cdot X' = 0 \quad X + X = X \quad X + X' = 1$

$XY + XY' = X \quad (X + Y)(X + Y') = X$

$X + XY = X \quad X + X'Y = X + Y \quad X(X + Y) = X \quad X(X' + Y) = XY$

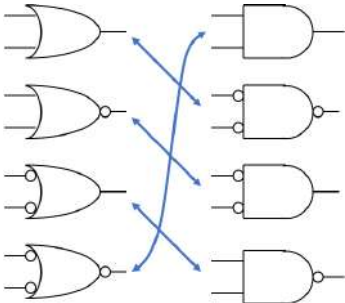
$XY + X'Z + YZ + \cdots = XY + X'Z \quad (X + Y)(X' + Z)(Y + Z \cdots) = (X + Y)(X' + Z)$

异或关系式： $X \oplus Y = (X + Y) \oplus XY \quad X \oplus 0 = X \quad X \oplus 1 = X' \quad X \odot 1 = X \quad X \odot 0 = X'$

异或传递： $A \oplus B = C \rightarrow A \oplus C = B$

德摩根定理： $(X_1 \cdot X_2 \cdots X_n)' = X_1' + X_2' + \cdots + X_n' \quad (X_1 + X_2 + \cdots + X_n)' = X_1' \cdot X_2' \cdots X_n'$

德摩根等效门：或变与，三个位置圈圈(非)取反



Duality theorem 对偶定理	Generalized DeMorgan's theorem 德摩根定理
交换所有的0和1以及"+"和"."，规则对倒	交换所有的0和1以及"+"和"."，变量取反
$F^D(X_1, X_2, \cdots, X_n, 0, 1, +, \cdot) = F(X_1, X_2, \cdots, X_n, 1, 0, \cdot, +)$	$F'(X_1, X_2, \cdots, X_n, 0, 1, +, \cdot) = F(X_1', X_2', \cdots, X_n', 1, 0, \cdot, +)$
<div><div>0 ↔ 1</div><div>x ↔ x'</div><div>• ↔ +</div><div>() ⇒ • ⇒ +</div></div>	<div><div>0 ↔ 1</div><div>x ↔ x'</div><div>• ↔ +</div><div>() ⇒ • ⇒ +</div></div>

$F(A,B,C,D) = \{(A+C)'[(BC)'+D]\}'/(B+C)+AD$

举例： $F^D(A,B,C,D)= \{ \{ (AC)' + [(B+C)'D] \}' + BC \} (A+D)$

$F'(A,B,C,D)=\{ \{ (A'C')' + [(B'+C')D'] \}' + B'C' \} (A'+D')$

香农展开定理Shannon's expansion theorem

“与-或”式 $F(X_1, \cdots, X_i, \cdots, X_n) = X_i \cdot F(X_1, \cdots, 1, \cdots, X_n) + X_i' \cdot F(X_1, \cdots, 0, \cdots, X_n)$

“或-与”式 $F(X_1, \cdots, X_i, \cdots, X_n) = [X_i + F(X_1, \cdots, 0, \cdots, X_n)] \cdot [X_i' + F(X_1, \cdots, 1, \cdots, X_n)]$

Standard Representations

最大项、最小项都有 2^n 个， m_i, M_i 为最大最小表达式的值

最小项列表 (Sum-of-products expression(SOP)) \sum 若i的某位二进制值为0，则相应的变量取反， $\sum m_i = 1$

最大项列表 (Product-of-sums expression(POS)) \prod 若i的某位二进制值为1，则相应的变量取反， $\prod M_i = 0$

Minterms vs Maxterms					
A	B	C	Minterm	Number	Maxterm
0	0	0	A'B'C'	m ₀	A+B+C
0	0	1	A'B'C	m ₁	A+B+C'
0	1	0	A'BC'	m ₂	A+B'+C
0	1	1	A'BC	m ₃	A+B'+C'
1	0	0	AB'C'	m ₄	A'+B+C
1	0	1	AB'C	m ₅	A'+B+C'
1	1	0	ABC'	m ₆	A'+B'+C
1	1	1	ABC	m ₇	A'+B'+C'

最小项最大项列表关系式：（注意对偶转换，求反变换）

对偶变换是将m变成M且下标求补，求反变换将m变成M下标不变

$$m_i \cdot M_i = 0$$

$$m_i' = M_i$$

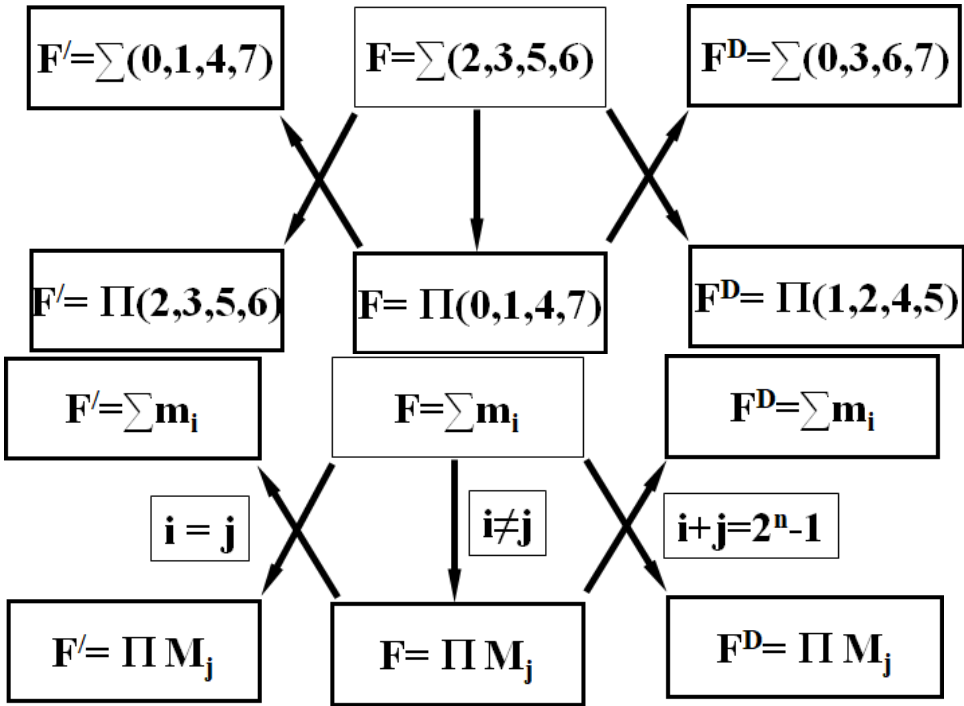
$$m_i^D = M_{n-1-i}$$

$$m_i + M_i = 1$$

$$M_i' = m_i$$

$$M_i^D = m_{n-1-i}$$

转化关系



Karnaugh Map

真值表的图形表示，两列变量以格雷码形式变换，两个相邻单元之间只有一个变量值不同

特征：逻辑相邻(Logic Adjacency)

例： $F(A, B, C, D) = AC' + BCD' + B = AC' + B$ 如果1的位置一个变量有0有1即消除

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

AC'

B

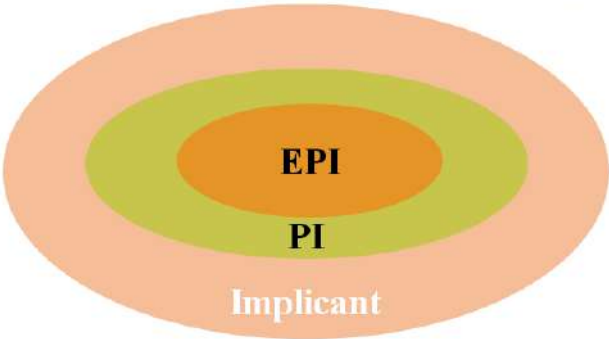
最简原则：

- 圆圈的数量必须最少
- 每个圆（由2ⁱ个相邻单元组成）尽可能大
- 可以选择重叠的圆，但这些圆中至少有一个单元格不被任何其他圆环绕
- 所有1或0的单元格都必须圈起来

Circle 1's → Minimal sum (最小和)	1就写原变量，见0就写反变量，每个圈构成一个与项
Circle 0's → Minimal product(最小积)	0就写原变量，见1就写反变量，每个圈构成一个或项

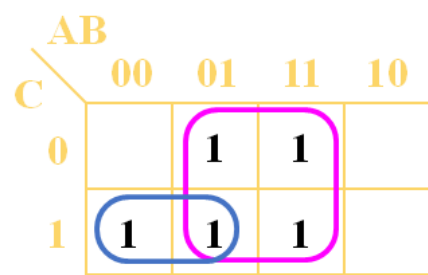
逻辑函数蕴涵关系

Relationship of Implicant(蕴含), PI(主蕴含项) and EPI(质主蕴含项)



Prime Implicants(PI, 主蕴含项)

- 主蕴含项是指不包含在任何其他蕴含项中的蕴含项，即它不能扩展为更大的蕴含项。
- 主蕴含项是卡诺图上可以圈出的最大的相邻1的单元集
- 主蕴含项可能不止一个，各个主蕴含项包含的变量个数也可能不一样



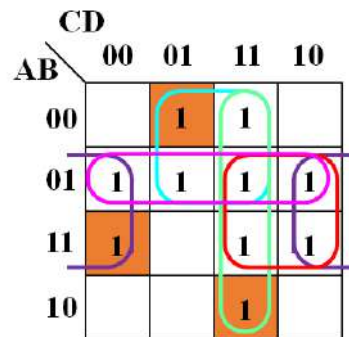
There are 2 prime implicants:

PI1: 1 Group of four minterms: { B }

PI2: 1 Group of two minterms: { A/C }

Distinguished 1-cell(奇异1单元): 只被一个主蕴含项覆盖一次的单元, 即在卡诺图上只被一个群包围一次

Essential Prime Implicant(EPI, 质主蕴含项): 质主蕴含项是指包含一个或多个奇异1单元的主蕴含项



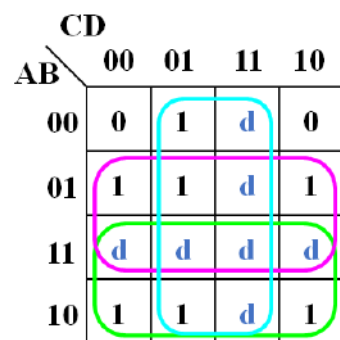
5 PI: { BD', A/B, A/D, CD, BC }

3 distinguished 1-cells: { 1, 11, 12 }

3 EPI: { BD', A/D, CD }

Don't-Care" Input Combination 无关项: 所有无效的BCD代码都被称为无关项(d-set)

例题:



$$F = \sum_{A,B,C,D} (1,4,5,6,8,9,10) + d(3,7,11,12,13,14,15)$$

$$F_{\text{minimal-sum}} = A + B + D$$

最小化

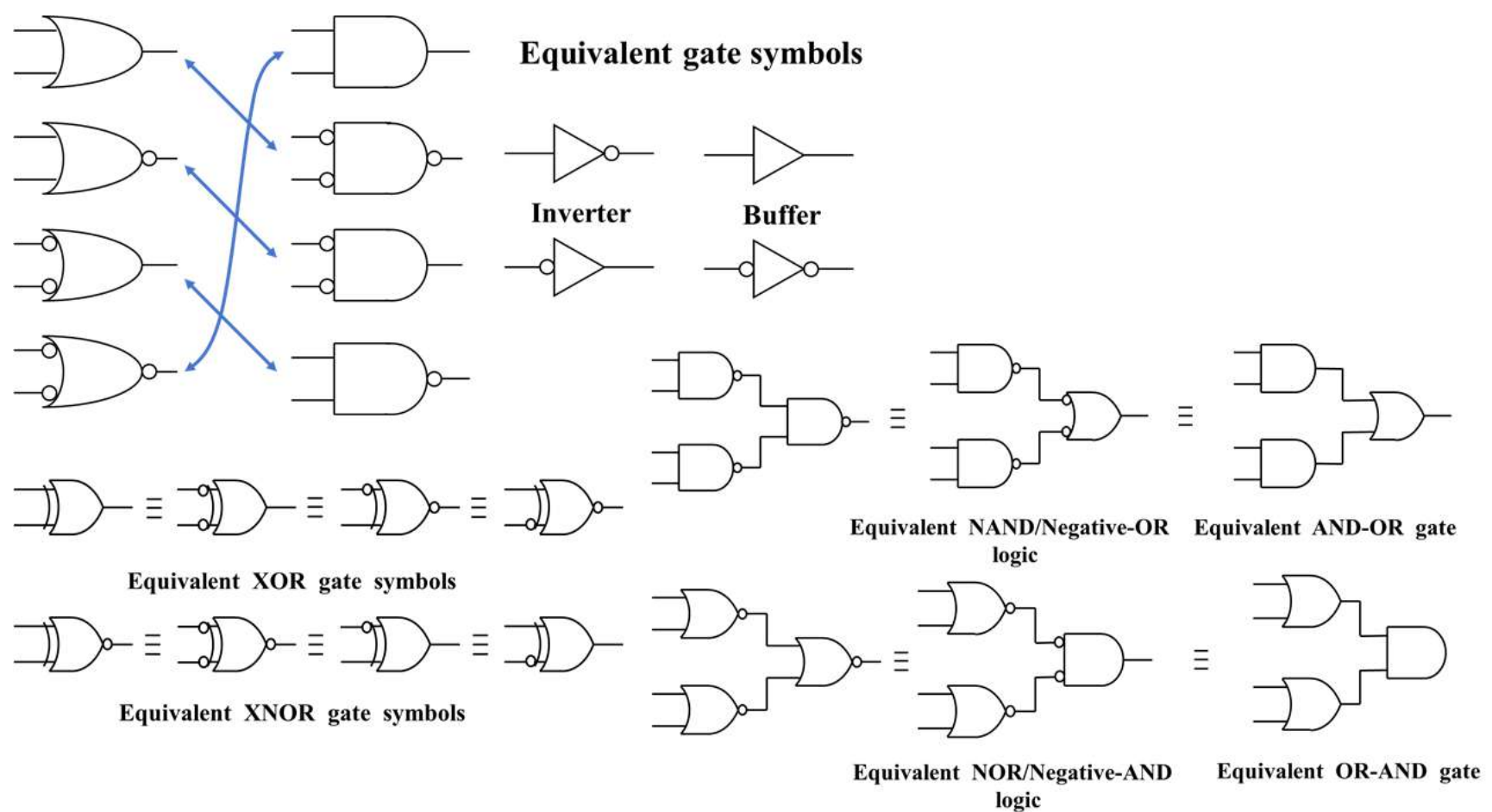
Multiple-Output Minimization 多输出最小化	MinimizationBased on Shared Term 基于共享项的最小化
<p>$F = XY + YZ, \quad G = X'Y' + X'Z$</p>	<p>$F = XY + X'YZ, \quad G = X'Y' + X'YZ$</p>

组合逻辑电路分析

组合逻辑电路

- 没有存储元件和反馈回路
- 输出仅取决于其当前输入

等效门(关注非的圈圈):



Timing Hazards

具有不同延迟的信号路径的过渡行为可能会导致电路的输出产生短脉冲（short pulse），通常称为小毛刺（glitch）

静态冒险

Static-1 Hazard：逻辑表达式恒为1转化产生0 glitch

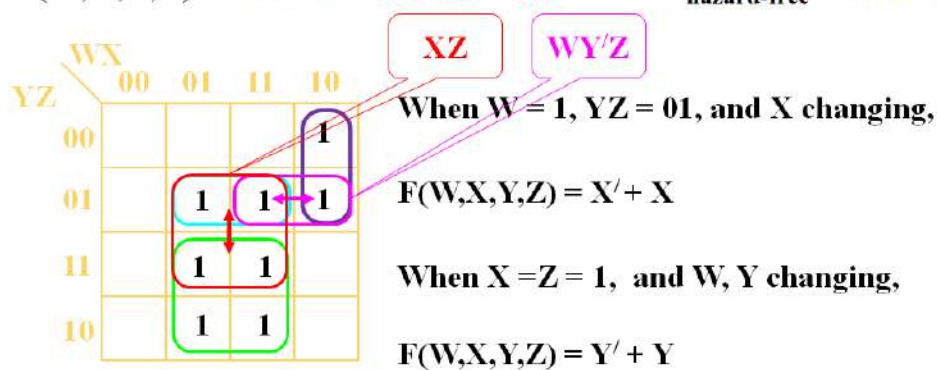
Static-0 Hazard：逻辑表达式恒为0转化产生1 glitch

静态1型冒险：“1”单元画的圈存在相切的地方则转化时候可能产生

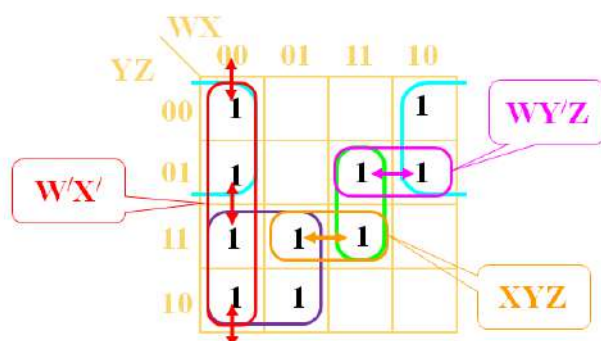
消除方法是在相切部分重新画圈或增加冗余项；

例题：(最小和)

$$F(W,X,Y,Z) = WX/Y' + XY'/Z + XY \rightarrow F_{\text{hazard-free}} = WX/Y' + XY + WY'/Z + XZ$$



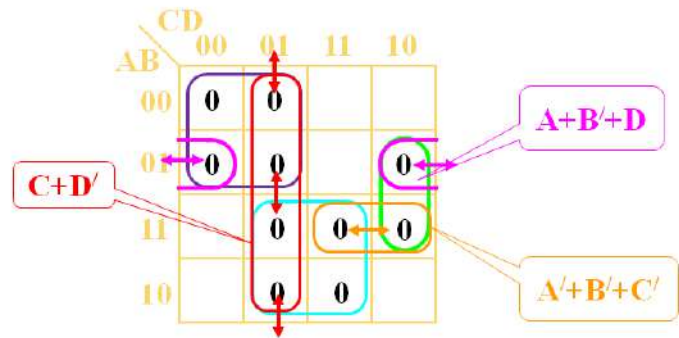
$$F(W,X,Y,Z) = W/Y + X/Y' + WXZ$$



$$F_{\text{hazard-free}} = W/Y + X/Y' + WXZ + WY'/Z + XYZ + W/X'$$

$$F(A,B,C,D) = (A+C) (A'+D') (B'+C'+D)$$

(最小积)1写成0相互累与:



$$F_{\text{hazard-free}} = (A+C)(A'+D') (B'+C'+D) (C+D')(A+B'+D)(A'+B'+C')$$

Combinational Logic Design Practices

Decoders 解码器

二进制解码器具有 n 位二进制输入代码和 1-out-of- 2^n 输出代码。

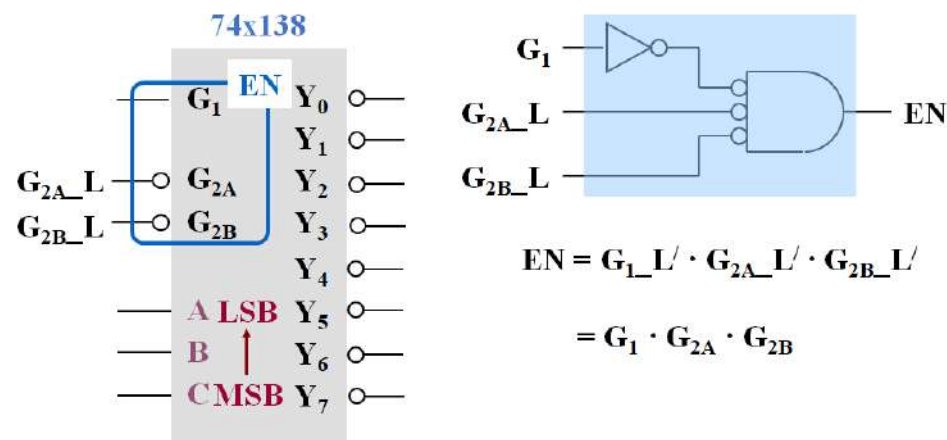
2-to-4 decoder: (74x139)

		Truth table							
		EN	I ₁	I ₀	Y ₃	Y ₂	Y ₁	Y ₀	
<div> <div>2-to-4 decoder</div> <div> I₀ → Y₀ I₁ → Y₁ EN → Y₂ Y₃ </div> </div>		0	x	x	0	0	0	0	
		1	0	0	0	0	0	1	
		1	0	1	0	0	1	0	
		1	1	0	0	1	0	0	
		1	1	1	1	0	0	0	

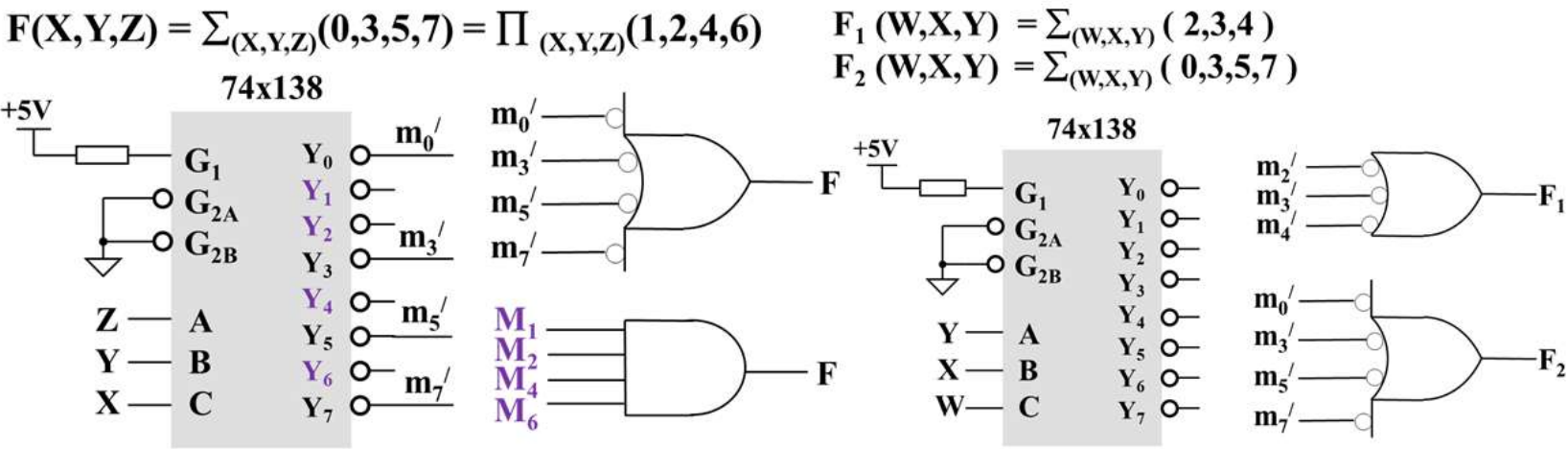
74x139 Dual 2-to-4 Decoder (注意, 非对应则正常输出为1, 对应数字输出为0)

		½ 74x139 truth table							
		G _L	B	A	Y _{3_L}	Y _{2_L}	Y _{1_L}	Y _{0_L}	
<div> <div>74x139</div> <div> 1G → 1Y₀ 1Y₁ 1ALS → 1Y₂ 1BMS → 1Y₃ </div> </div>		1	x	x	1	1	1	1	
		0	0	0	1	1	1	0	
		0	0	1	1	1	0	1	
		0	1	0	1	0	1	1	
		0	1	1	0	1	1	1	

3-to-8 Decoder: (74x138) (注意输出)

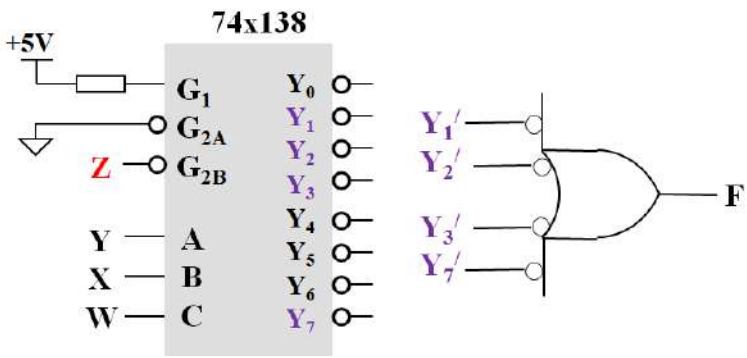


例题: 这里输出是Y_L 对应数输出低电平, 非对应数输出高电平 (或非门对应最小项)

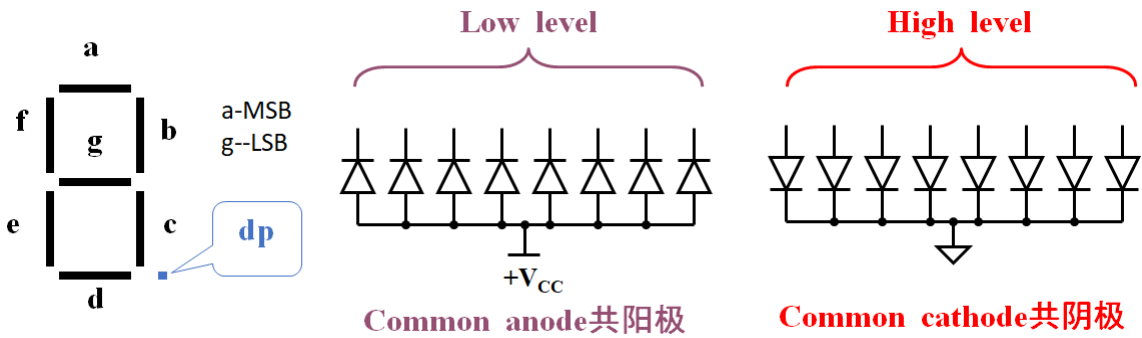


提取公因式（观察最小项列表）：

$F(W,X,Y,Z) = \sum_{(W,X,Y,Z)}(2,4,6,14) = Z/\sum_{(W,X,Y)}(1,2,3,7)$



Seven-Segment Display



如果有小数点（dp），则小数点为LSB（低有效）（正常都是高有效）

两个0的区别在于BI_L是0 还是1

74x49 truth table

BI_L	D C B A	a b c d e f g	Display
0	x x x x	0 0 0 0 0 0 0	
1	0 0 0 0	1 1 1 1 1 1 0	0
1	0 0 0 1	0 1 1 0 0 0 0	1
1	0 0 1 0	1 1 0 1 1 0 1	2
1	0 0 1 1	1 1 1 1 0 0 1	3
1	0 1 0 0	0 1 1 0 0 1 1	4
1	0 1 0 1	1 0 1 1 0 1 1	5
1	0 1 1 0	0 0 1 1 1 1 1	6
1	0 1 1 1	1 1 1 0 0 0 0	7
1	1 0 0 0	1 1 1 1 1 1 1	8
1	1 0 0 1	1 1 1 0 0 1 1	9
1	1 0 1 0	0 0 0 1 1 0 1	
1	1 0 1 1	0 0 1 1 0 0 1	
1	1 1 0 0	0 1 0 0 0 1 1	
1	1 1 0 1	1 0 0 1 0 1 1	
1	1 1 1 0	0 0 0 1 1 1 1	
1	1 1 1 1	0 0 0 0 0 0 0	

Encoders 译码器

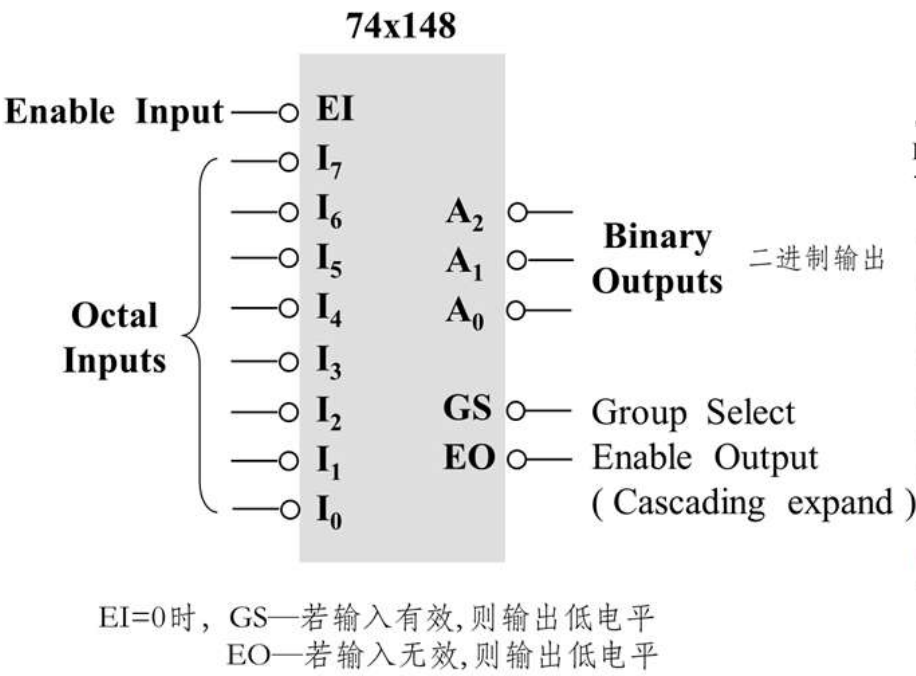
解码器的反向行为

Truth table for common encoder

H ₇	H ₆	H ₅	H ₄	H ₃	H ₂	H ₁	H ₀	A ₂	A ₁	A ₀	IDLE
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1	1	1	0
0	1	0	0	0	0	0	0	1	1	0	0
0	0	1	0	0	0	0	0	1	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Octal-to-Binary Priority Encoder (74x148)

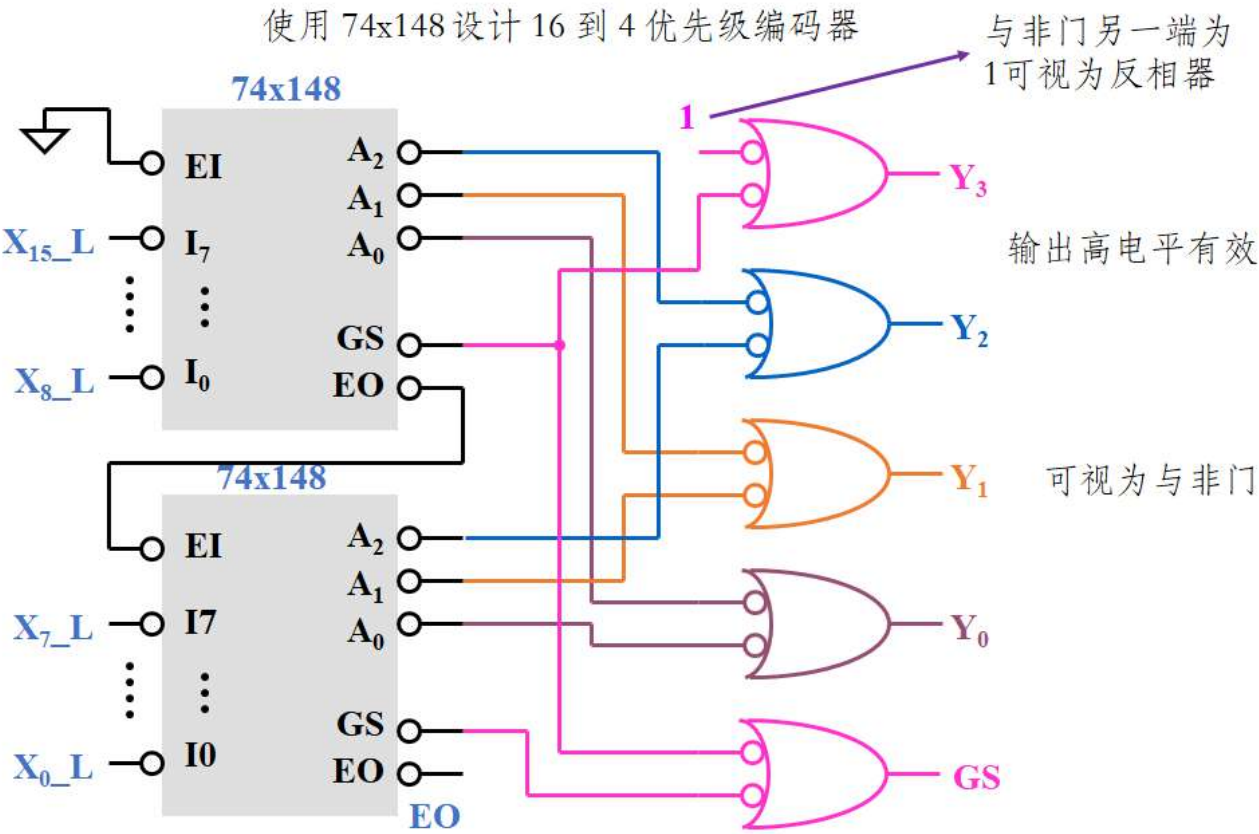
(with active low inputs) 低电平有效，输出也得反一下找最高的低输入，输出对应数的取反；



Truth table for a 74x148 priority encoder

E ₁ L	I ₇ L	I ₆ L	I ₅ L	I ₄ L	I ₃ L	I ₂ L	I ₁ L	I ₀ L	A ₂ L	A ₁ L	A ₀ L	GS ₁ L	EO ₁ L
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	0	x	x	x	x	x	x	x	0	0	0	0	1
0	1	0	x	x	x	x	x	x	0	0	1	0	1
0	1	1	0	x	x	x	x	x	0	1	0	0	1
0	1	1	1	0	x	x	x	x	0	1	1	0	1
0	1	1	1	1	0	x	x	x	1	0	0	0	1
0	1	1	1	1	1	0	x	x	1	0	1	0	1
0	1	1	1	1	1	1	0	x	1	1	0	0	1
0	1	1	1	1	1	1	1	0	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0

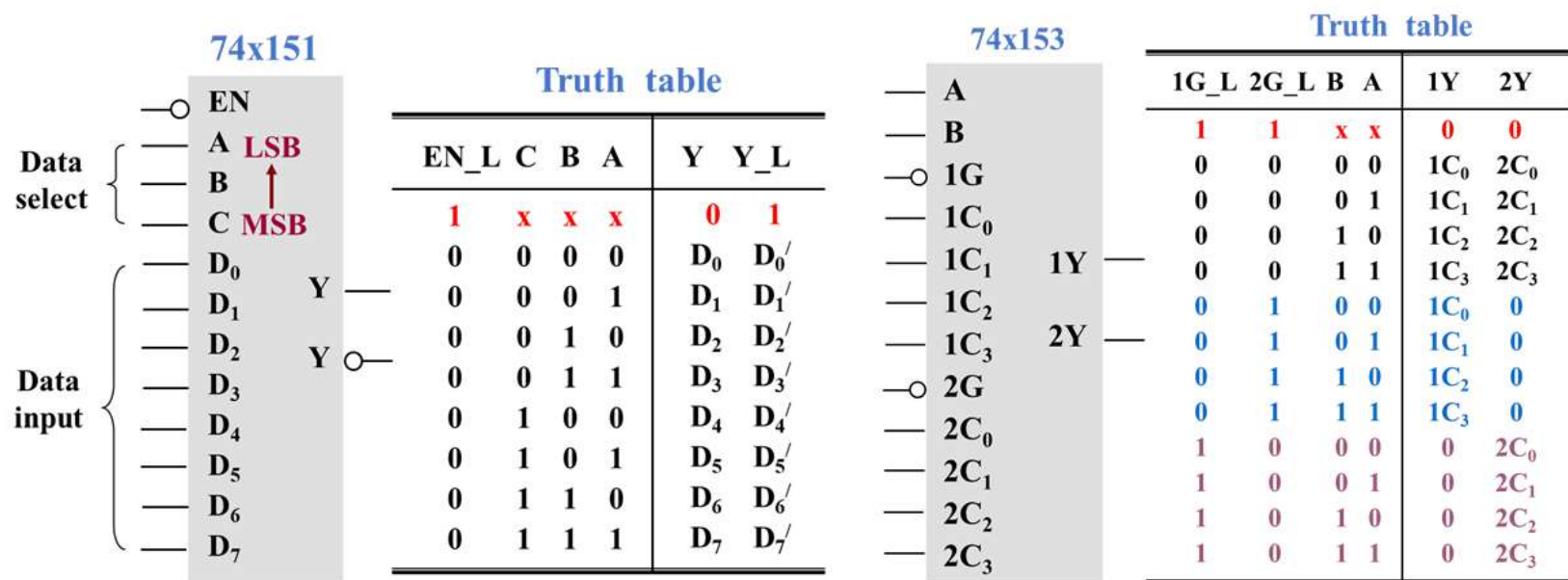
例题：16-to-4 Priority Encoder Using 74x148 (>=8时上面那个74x148有效，否则无效)



Multiplexers 多路复用器

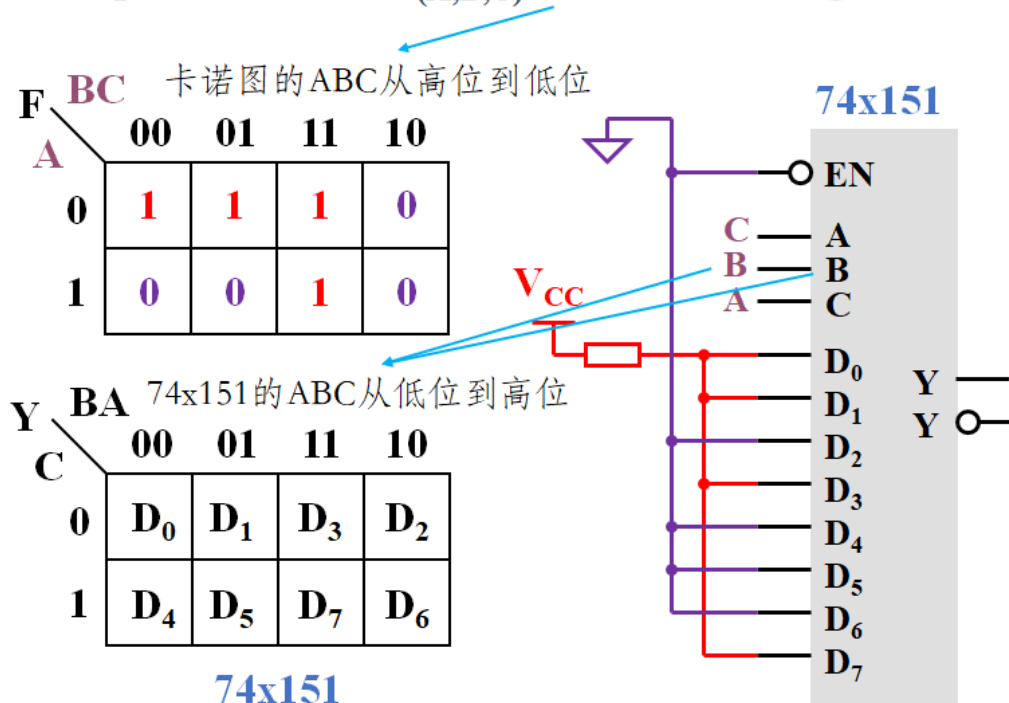
74x151: 8 输入、1 位多路复用器；74x153: 4 输入、2 位多路复用器

看内部CBA (BA) 输入啥决定输出从D₀(C₀)到D₇(C₃)取哪个



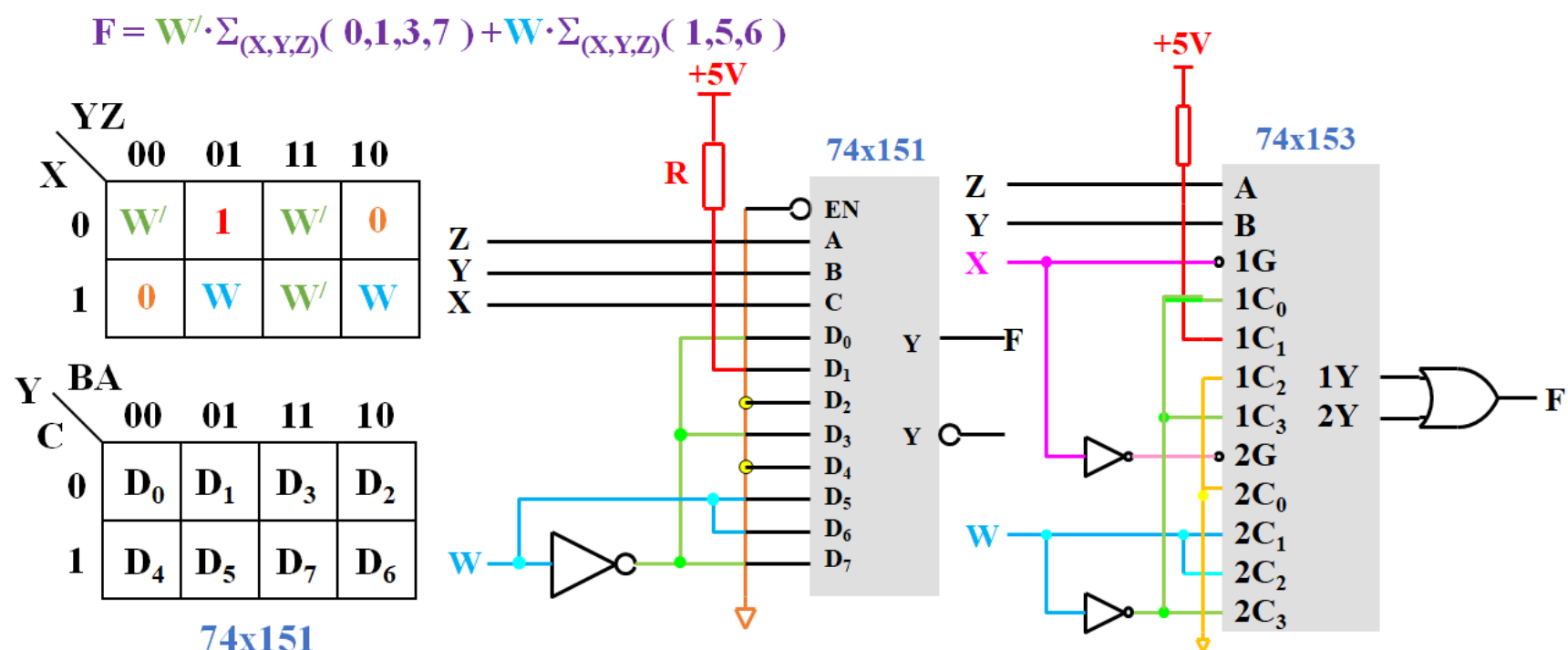
实现逻辑功能:

Implement $F = \sum_{(A,B,C)} (0,1,3,7)$ using a 74x151



Implement $F = \sum_{(w,x,y,z)} (0, 1, 3, 7, 9, 13, 14)$ using a 74x151 or 74x153

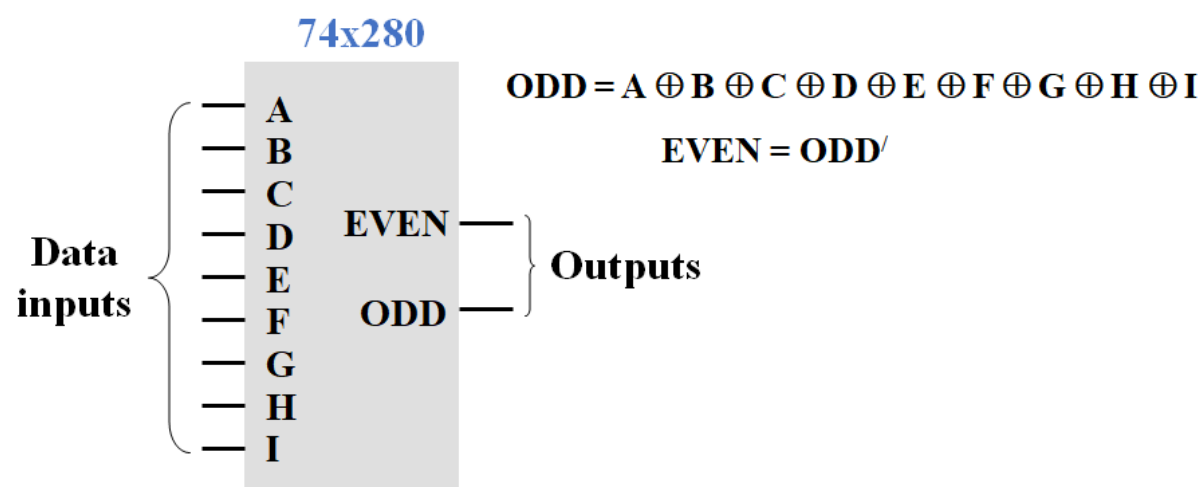
153再将 $X = 0/1$ 的情况分开



Parity Circuits 奇偶校验

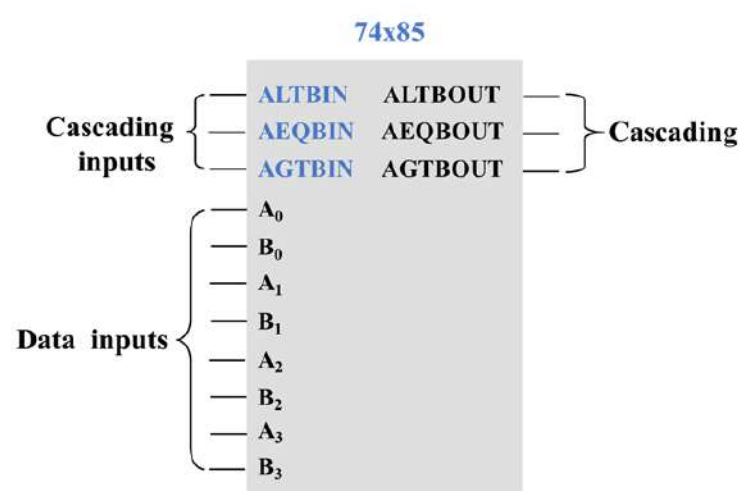
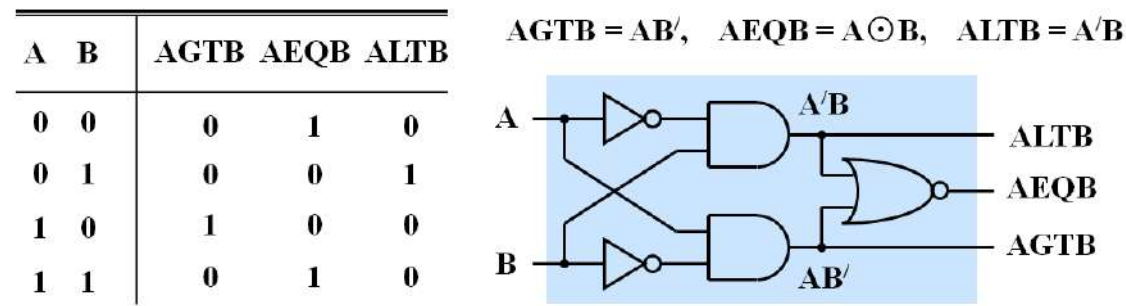
$$\text{ODD} = I_0 \oplus I_1 \oplus \dots \oplus I_N = \begin{cases} 1 & \text{Odd number of 1 s} \\ 0 & \text{Even number of 1 s} \end{cases}$$

$$\text{EVEN} = \text{ODD}' = \begin{cases} 0 & \text{Odd number of 1 s} \\ 1 & \text{Even number of 1 s} \end{cases}$$



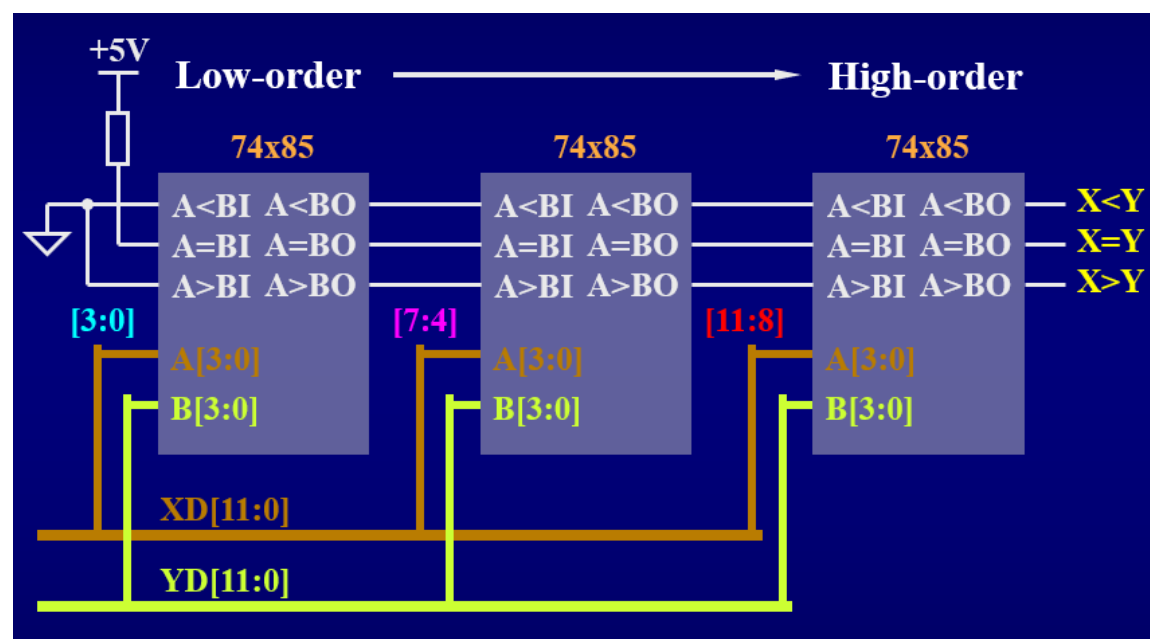
Comparator 比较器

1-bit Magnitude Comparator



级联输入用于实现超过4位

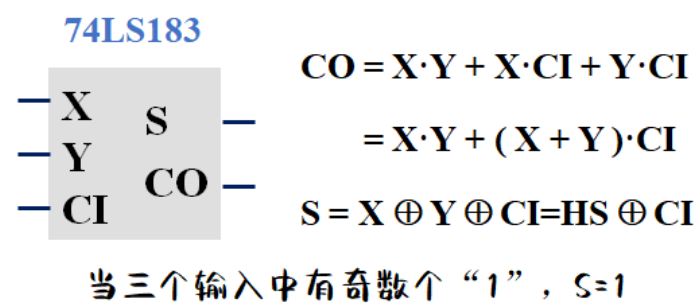
Data inputs				Cascading inputs			Outputs		
A ₃ B ₃	A ₂ B ₂	A ₁ B ₁	A ₀ B ₀	ALTBIN	AEQBIN	AGTBIN	ALTBOUT	AEQBOUT	AGTBOUT
A ₃ < B ₃	x x	x x	x x	x	x	x	1	0	0
A ₃ > B ₃	x x	x x	x x	x	x	x	0	0	1
A ₃ = B ₃	A ₂ < B ₂	x x	x x	x	x	x	1	0	0
	A ₂ > B ₂	x x	x x	x	x	x	0	0	1
	A ₁ < B ₁	x x	x x	x	x	x	1	0	0
		x x	x x	x	x	x	0	0	1
	A ₂ = B ₂	A ₀ < B ₀	x	x	x	1	0	0	
		A ₀ > B ₀	x	x	x	0	0	1	
	A ₁ = B ₁	A ₀ = B ₀	1	0	0	1	0	0	
			0	1	0	0	1	0	
			0	0	1	0	0	1	



Adders 加法器

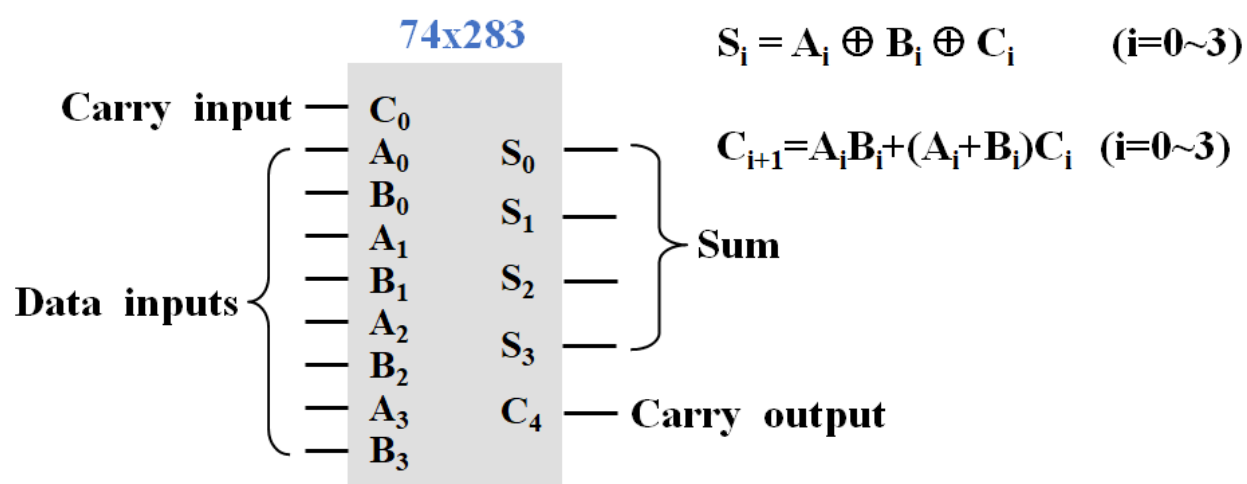
Half Adders--不包含进位位; Full Adders--包含进位位 (当有奇数个1输入时, S输出为1)

74x183 1-bit full adder

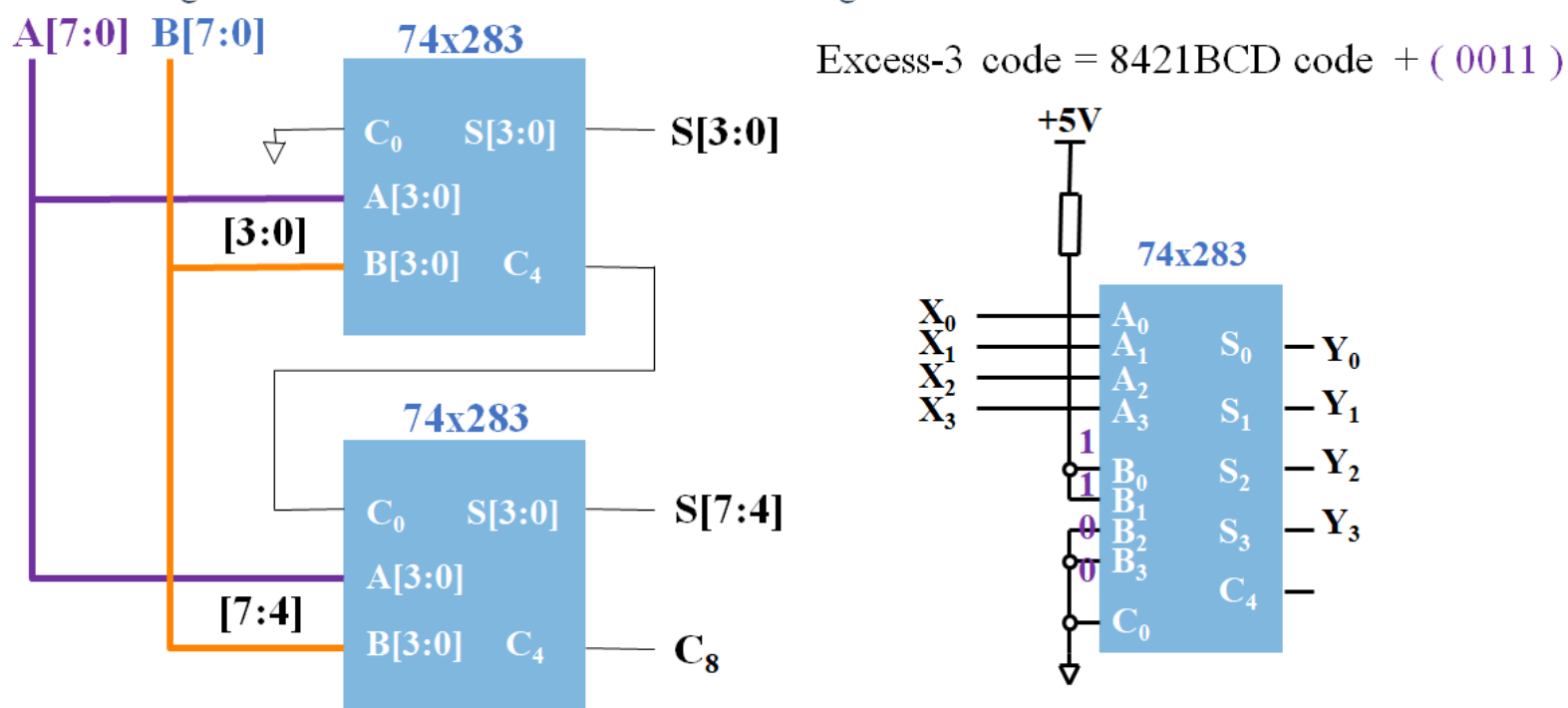


74x283 4-bit Binary Adder (当有奇数个1输入时，S输出为1)

输出就是A+B，如果要 $A - x$ 可以让B为 $(16 - x)$



Cascading two 74x283s to an 8-bit adder Design an 8421BCD to Excess-3 Code Converter



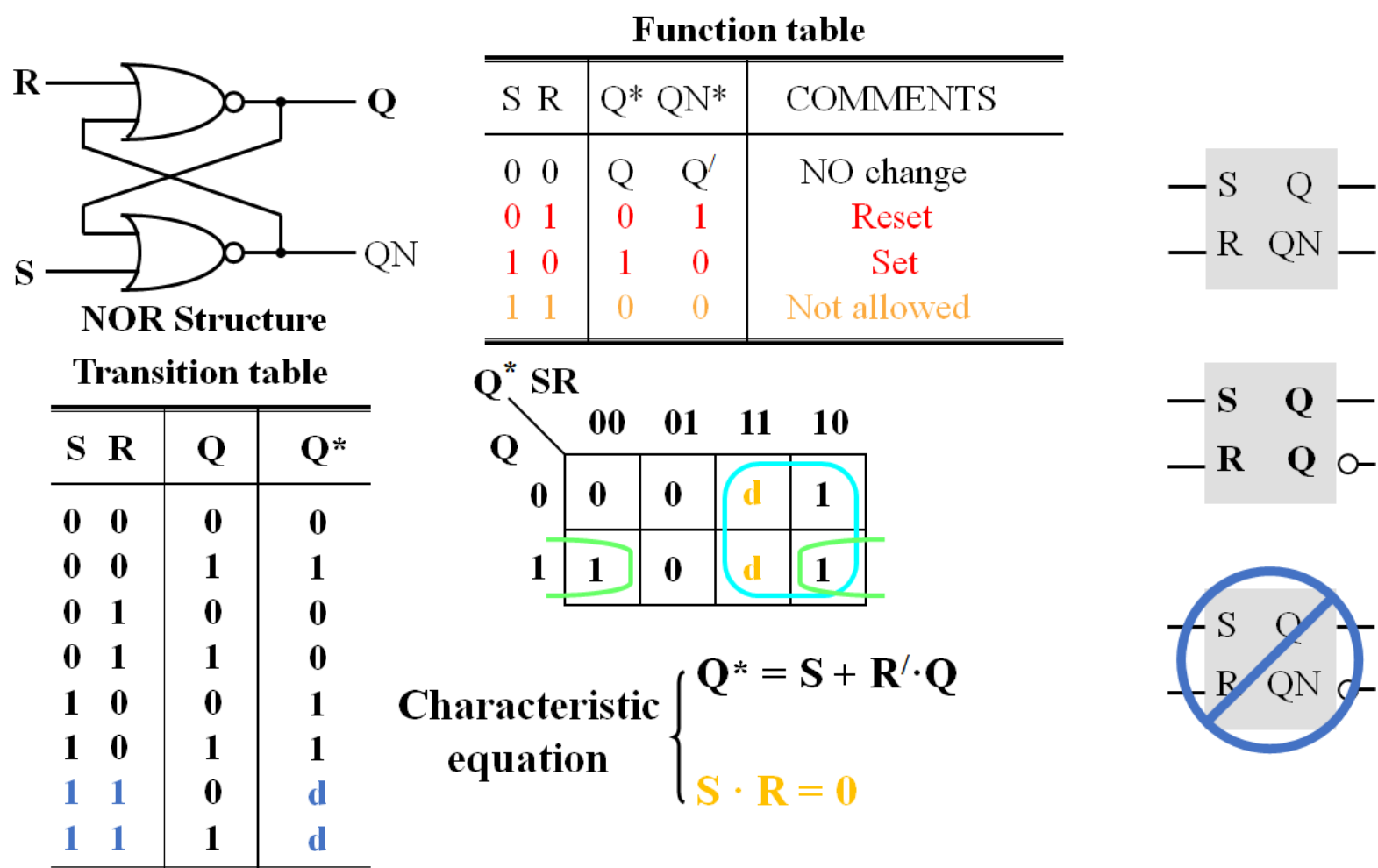
Sequential Logic Design Principles

主要画的都是简化器件

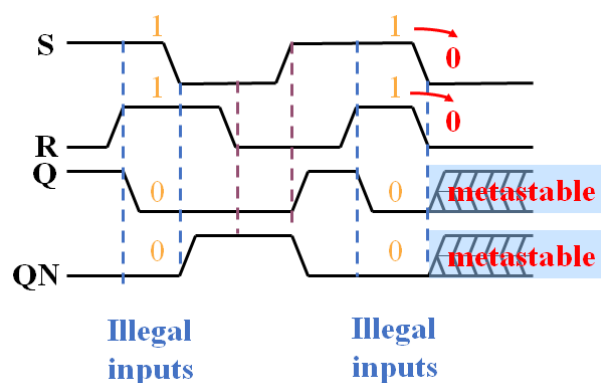
Q^* 是Q的下一状态，S-R类器件均已被淘汰；QN本质上是 Q' ，所以从相同状态突然将启动置0会产生亚稳态

Latches 锁存器

S-R锁存器

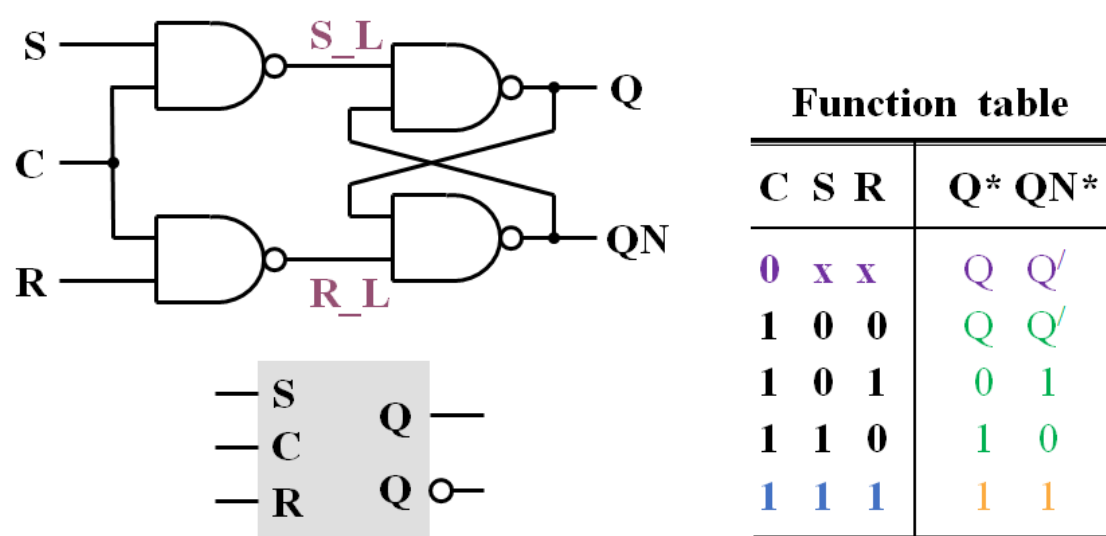


亚稳状态：当 $S = R = 1$ 时，如果 S 和 R 同时变为“0”，则下一个状态将是振荡或亚稳态

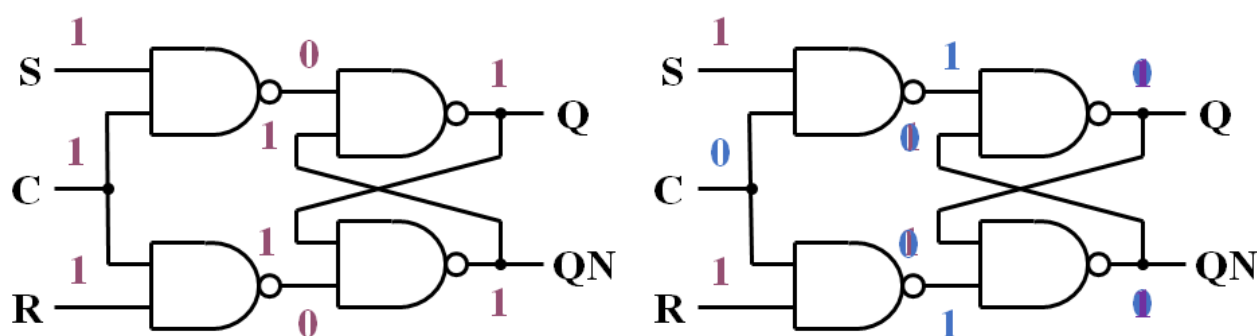


门控 S-R 锁存器

synchronous

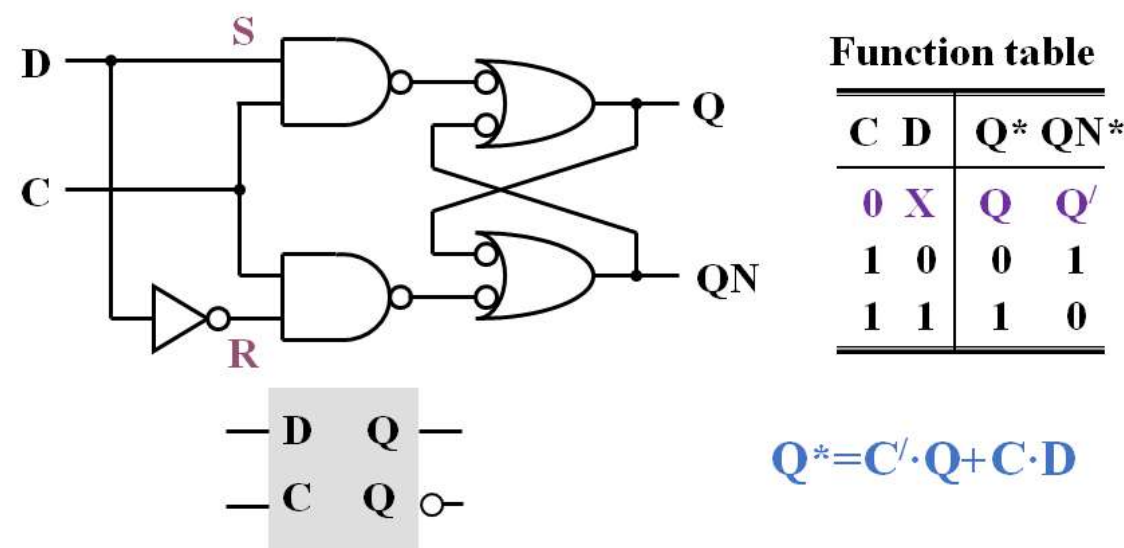


在 $S = R = 1$ 下，如果 C 从 1 变为 0，则下一个状态将是振荡或亚稳态



D（延迟）锁存器

C 低电平锁存，根据D输入改变Q*的值



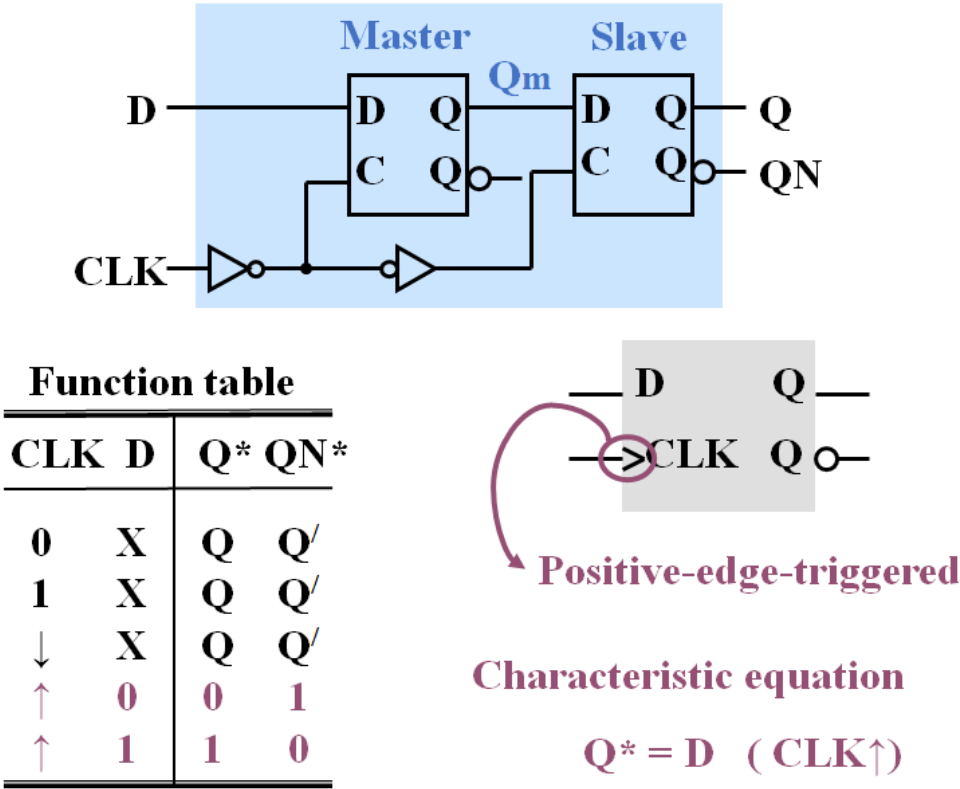
Edge-Triggered Flip-Flop

边沿触发器

D f-f	J-K f-f	T f-f
$Q^* = D$	$Q^* = JQ' + K'Q$	$Q^* = Q'$ 带使能: $Q^* = EN \oplus Q$

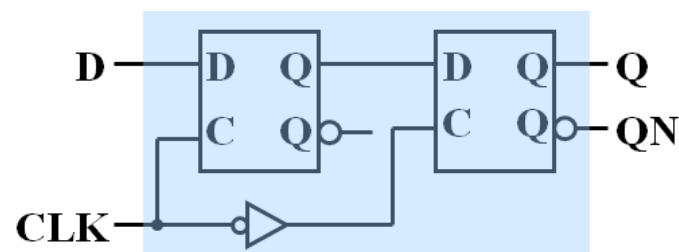
D Flip-Flop

也有可能进入亚稳态: metastable status



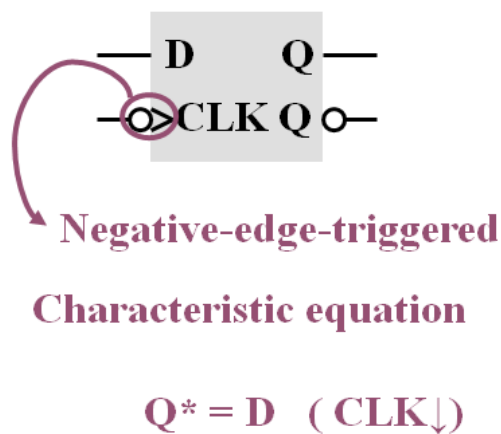
正边沿触发：下一个状态 Q* 取决于时钟脉冲 CLK 上升沿之前的输入 D；状态只能在时钟脉冲CLK的上升沿改变。

下面是负边沿触发的



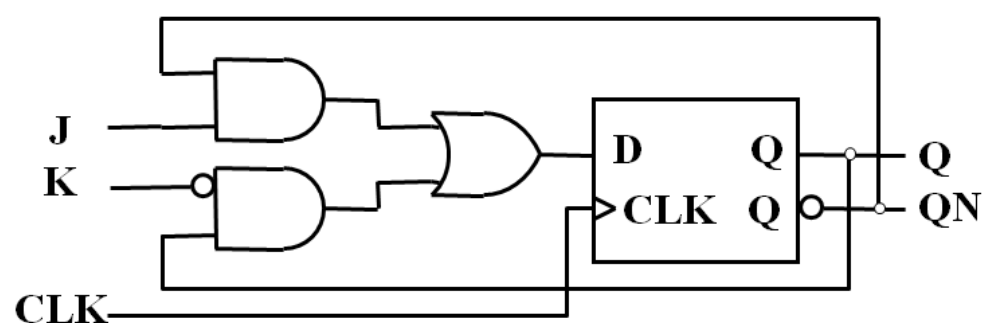
Function table

CLK	D	Q*	QN*
0	X	Q	Q'
1	X	Q	Q'
↑	X	Q	Q'
↓	0	0	1
↓	1	1	0

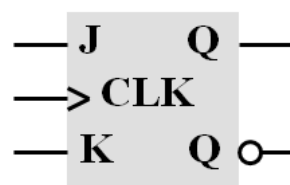


J-K Flip-Flop

j=k=1时会不断上下振荡



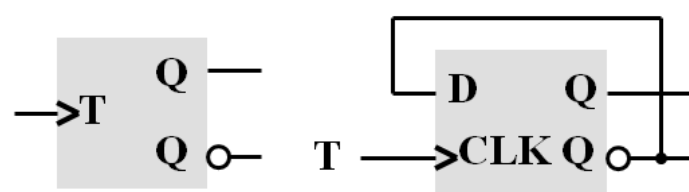
$$Q^* = J \cdot Q' + K' \cdot Q \text{ (CLK}\uparrow\text{)}$$



Function table

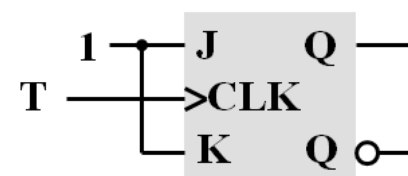
J	K	CLK	Q*	QN*
X	X	0	Q	Q'
X	X	1	Q	Q'
X	X	↓	Q	Q'
0	0	↑	Q	Q'
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	Q'	Q

T(Toggle) Flip-Flop



$$Q^* = Q' \text{ (T}\uparrow\text{)}$$

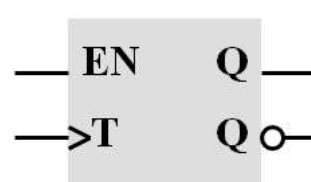
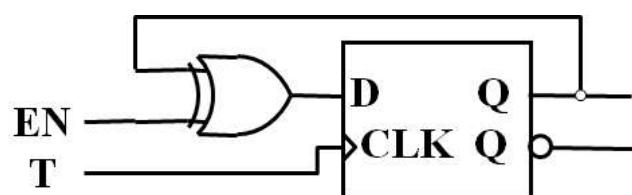
$$Q^* = D = Q' \text{ (CLK}\uparrow\text{)}$$



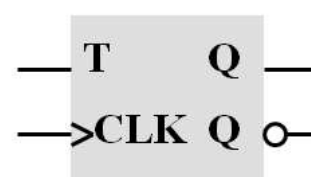
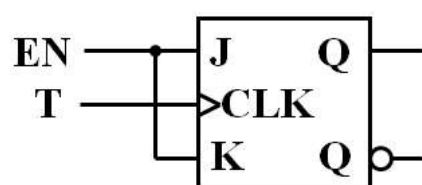
$$Q^* = J \cdot Q' + K' \cdot Q = Q' \text{ (CLK}\uparrow\text{)}$$

特征方程： $Q^* = Q' \text{ (T}\uparrow\text{)}$

带使能的 T 触发器：



$$Q^* = EN \oplus Q \text{ (T}\uparrow\text{)}$$

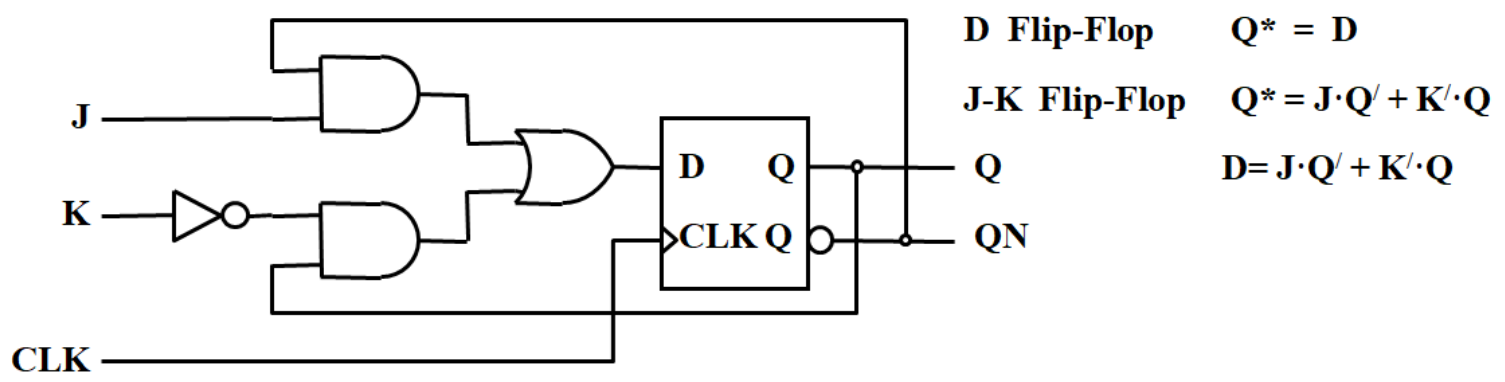


$$Q^* = T \oplus Q \text{ (CLK}\uparrow\text{)}$$

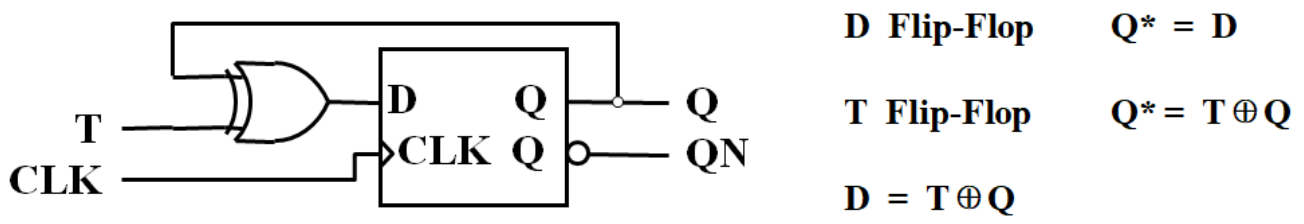
Flip-Flops Conversion

写出表达式然后建立等式关系，建立D与输入的关系

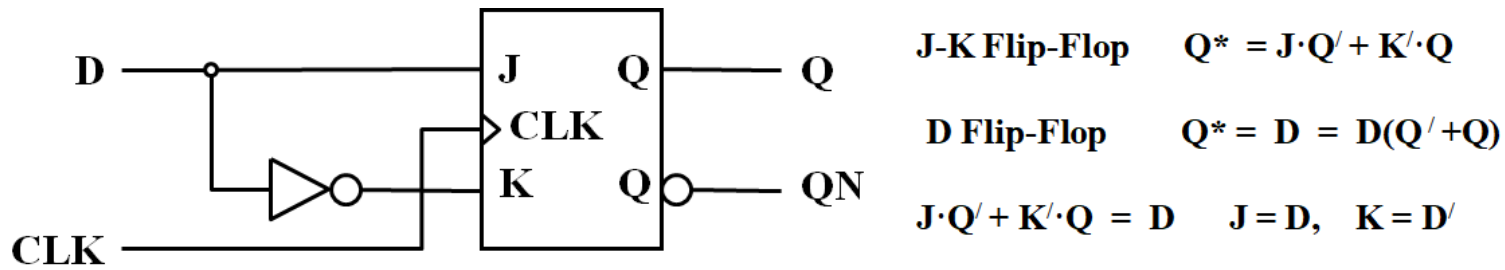
D 触发器转换为 J-K 触发器



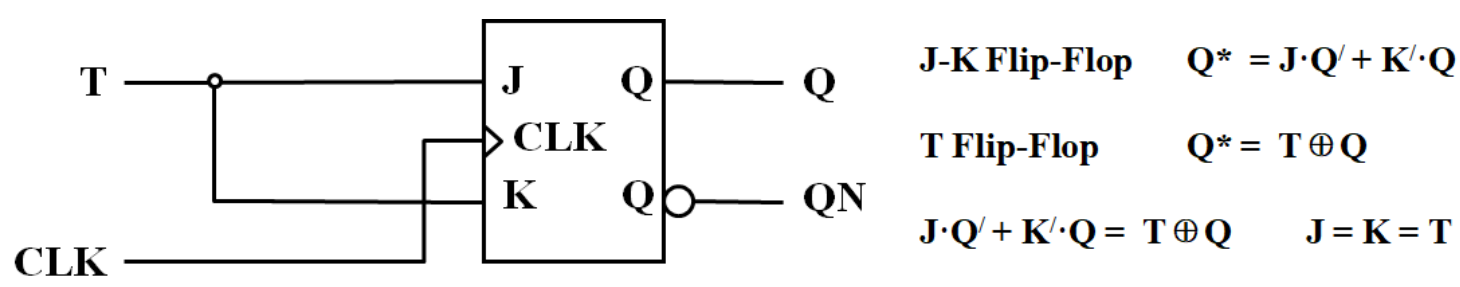
D 触发器转换为 T 触发器



J-K 触发器转换为 D 触发器



J-K 触发器转换为 T 触发器



Sequential Logic Circuit

Excitation equations: 激励方程（触发器D的表达式）

Transition/State equations: 过渡/状态方程（Q*的表达式）

Output equation: 输出方程（最终输出F输出的表达式）

State diagram 状态图

左侧和右侧分别是输入和输出（输入状态后的输出）

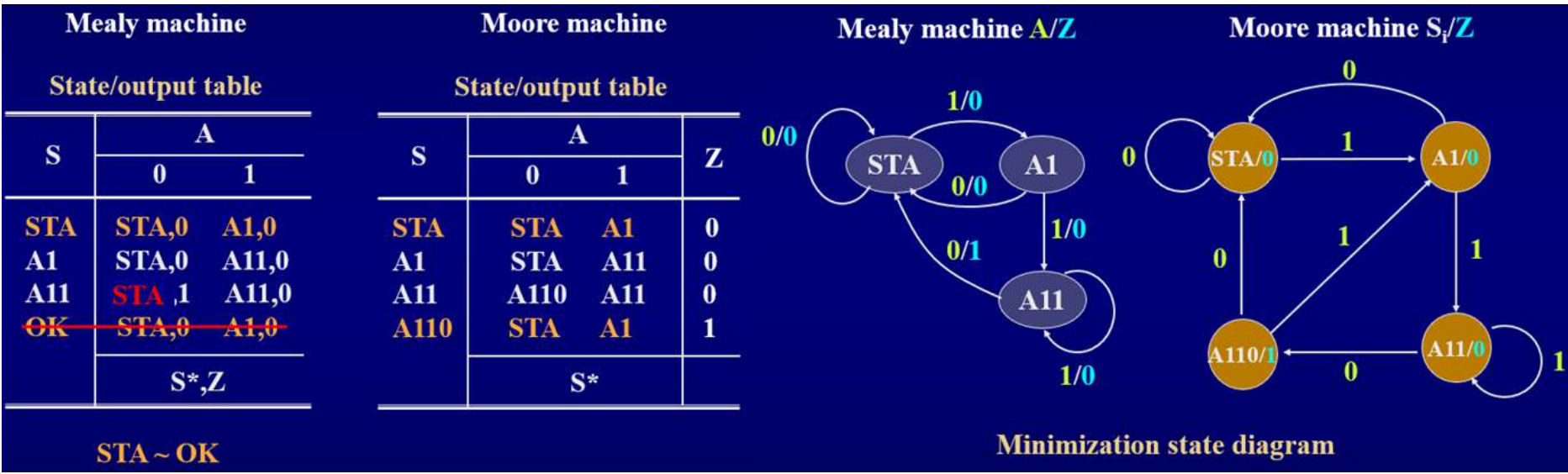
Mutual exclusion: 离开状态的弧上每对可能的转换表达式的逻辑积为 0

All inclusion: 所有离开状态的弧上的转移表达式的逻辑和为 1

Mealy Machine: 米利机（输出与外输入，与现态有关） **Moore Machine:** 摩尔机（输出与外输入无关，与现态有关）

STA: 初始状态，在要求的序列中没有得到正确的输入

ambiguous states 指不同输入类型相或≠1（指出箭头



Counter 计数器

QDQCQBQA=Q3Q2Q1Q0 与移位寄存器不一样

Modules——模数：循环中的状态数

Binary Counter：二进制计数器

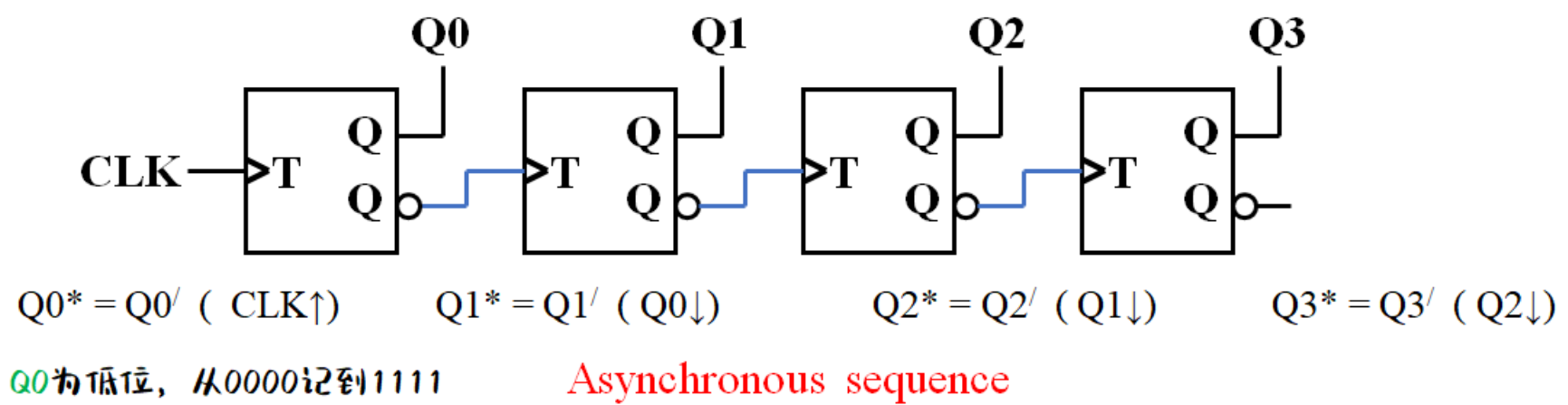
n flip-flops and 2ⁿ states 编码为相应 n 位二进制整数的状态

计数器类型

Clock mode: Synchronous, Asynchronous (同步，异步)

Count mode: Up, Down, Up/Down. (计数模式)

Encoding mode: Binary, BCD, Gray, etc. (编码方式)



74x163 4-bit binary counter (LD CLR ENP ENT 接高电平)

控制CLR来实现在某些位置清0，控制LD在某些位置装载数字

1111自动clear;

74x163

Synchronous clear

Synchronous load

Enable input {

Data input {

CLK

CLR

LD

ENP

ENT

A

B

C

D

LSB

MSB

QA

QB

QC

QD

RCO

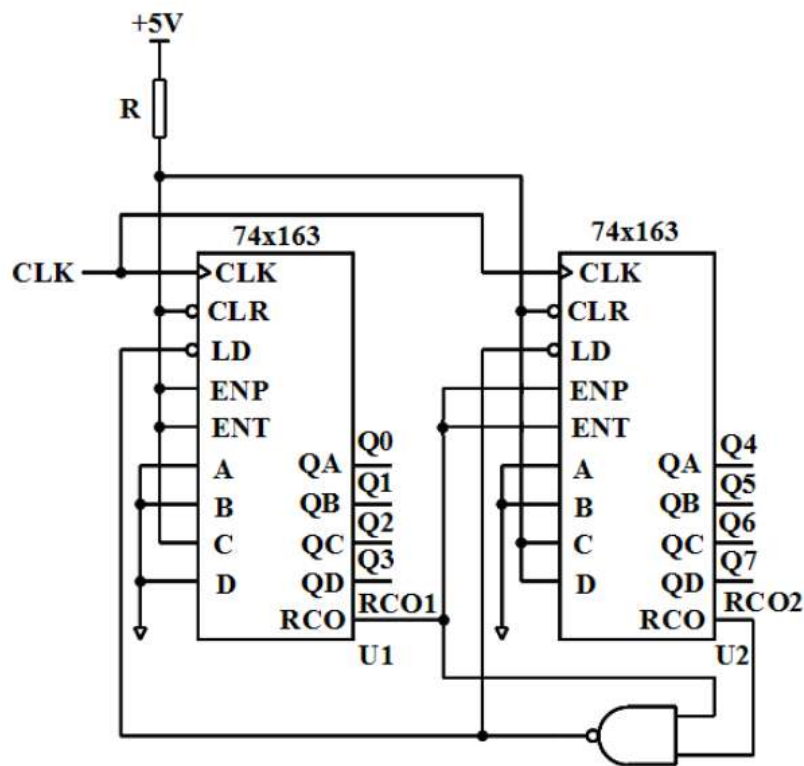
Data output

Ripple carry out

The 74x163 function table

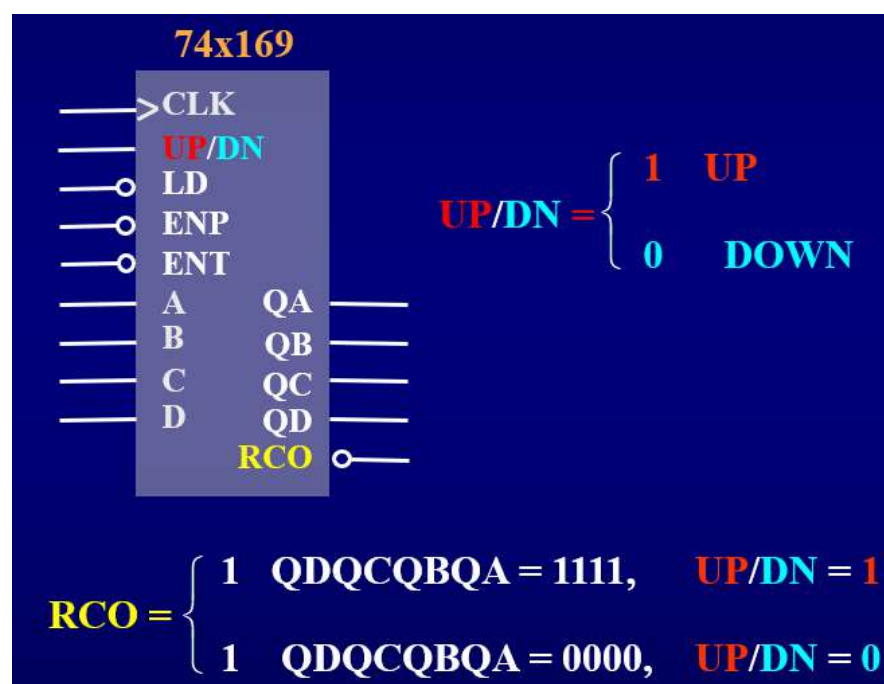
CLK	CLR_L	LD_L	ENP	ENT	Mode
	0	x	x	x	Synchronous clear
	1	0	x	x	Synchronous load
x	1	1	0 or 0		Hold
	1	1	1	1	Count

163实现8位计数器：下图序列是从1100 0100 到1111 1111 （左边低4位到1111 后RCO置1，让右边的4位加1，直到两边都加到1111 1111 后重新LD ）



The function table for 74x163

74x169 Up/Down Counter



Any Modulus Counter

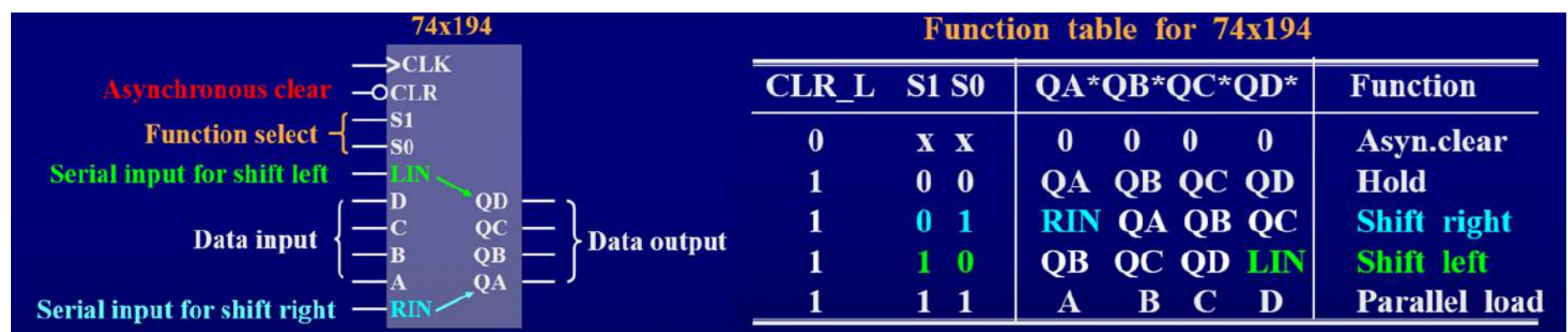
控制CLR_L=0来实现在某些位置清0，控制LD_L=0在某些位置装载数字

Shift Register 移位寄存器

Serial：串；**Parallel**：并

74x194 4-bit Universal Shift Register 正常顺序了，QA就是高位 QAQBQCQD=Q3Q2Q1Q0

S1 左 S0 右 都1则装载

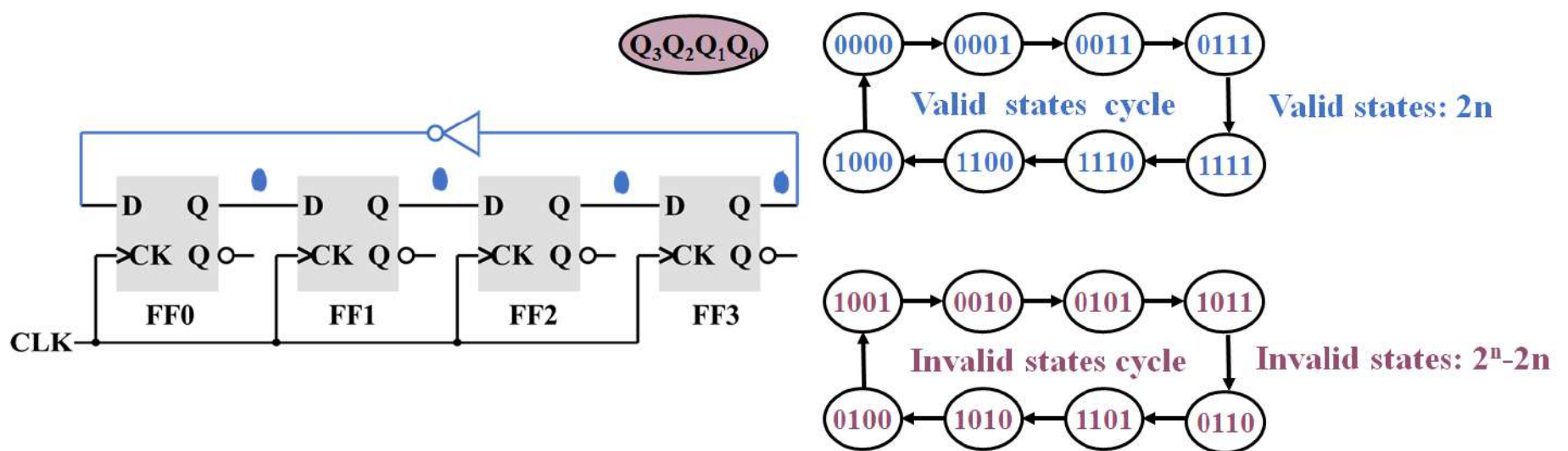


考试需记住的两种计数器

Ring Counters：n位环行计数器模为n such as 0001 0010 0100 1000 0001

Twisted-Ring/Johnson Counters扭环/约翰逊计数器

Johnson counter 用n位记2n个数，有 $2^n - 2n$ 数无用 (abnormal state)



Linear Feedback Shift-Register (LFSR) Counters

线性反馈移位寄存器 (LFSR) 计数器:

一个 n bitz LFSR 计数器在单个周期内包含 $2^n - 1$ 个有效状态, 无自校正 (**without self-correcting**) (全0会死循环)

have self-correcting 则 2^n 个有效状态

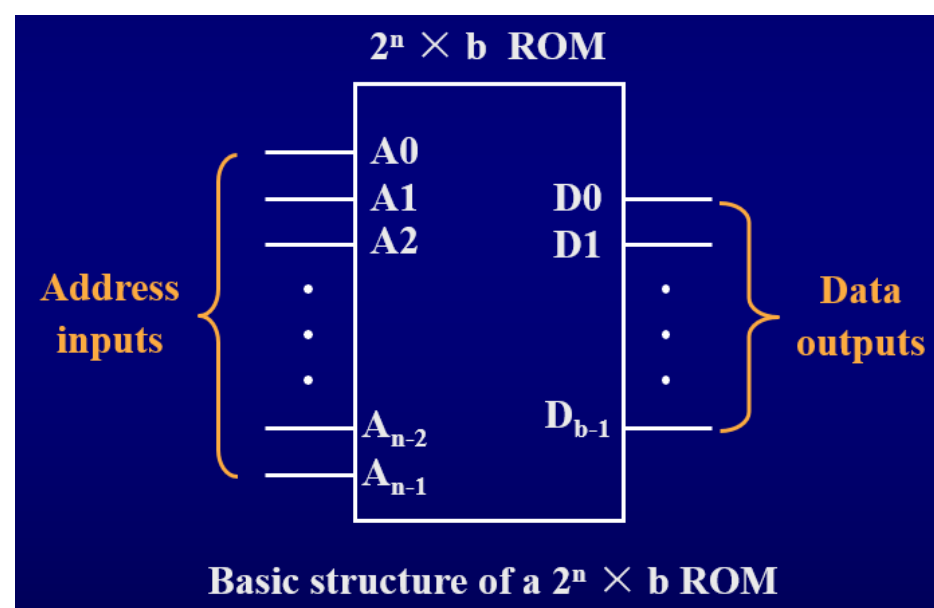
触发器的数量

题目只说filp则: $2^n \geq L$ 提到using shift register则需考虑多少位不会发生重复

ROM

Memory capacity: 下图乘积的单位为bits

A ROM has 2^n words



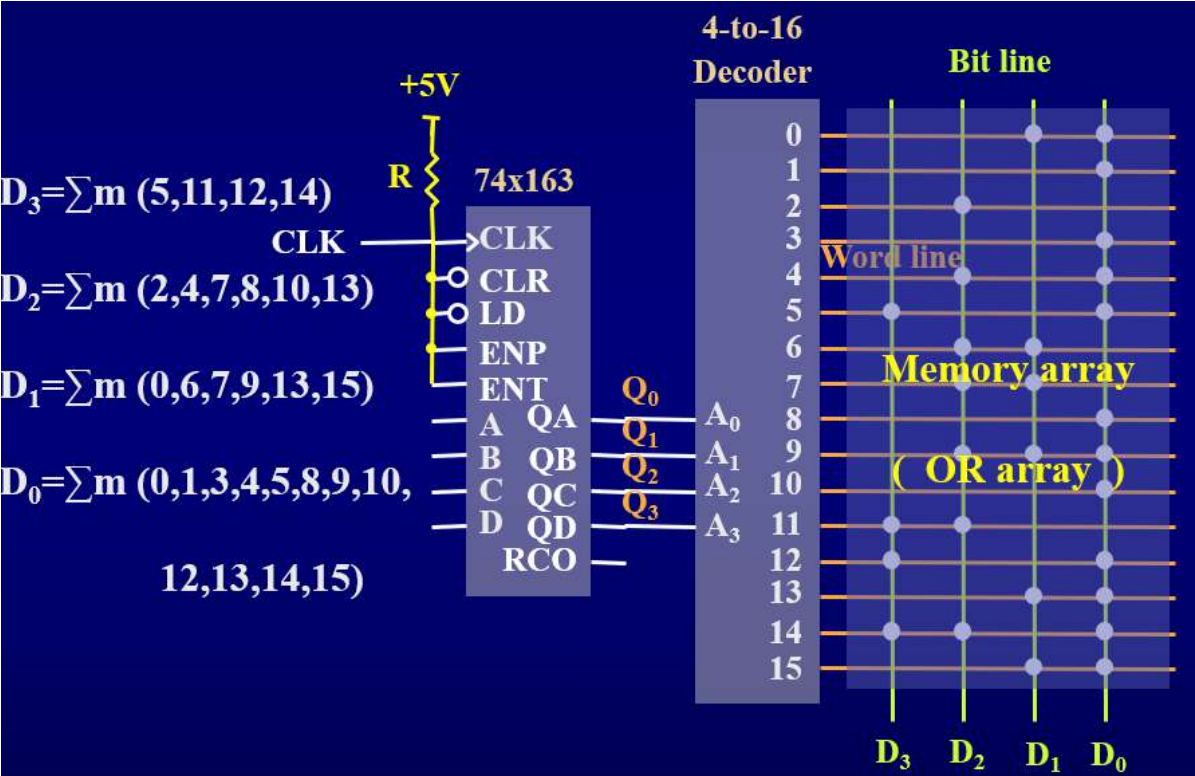
1byte = 8 bits, (一字节等于8位) 1KB = 2^{10} bytes = 1024 bytes (1K字节等于1024字节)

1MB = 210KB = 1024 KB, 1GB = 210 MB = 1024 MB

有几个output就可以表示多少个组合逻辑函数

标准 EPROM 的逻辑符号 8Kx8 前面8K表示有 $2^3 \times 2^{10}$ 个存储单元 后面的8代表每个存储单元只能存储8个二进制

ROM MOS管导通则输出1



DAC and ADC

DAC的模拟输出电压的通式： $V_O = K \times \sum_{i=0}^{n-1} (D_i \times 2^i)$ 最小是0000 0001 (8 bits)

D/A转换器分辨率Resolution： $\text{Resolution} = \frac{1}{2^n - 1} \times 100\%$

voltage range 电压范围

考试

英语单词

asynchronous: 异步 dector: 检测 activated: 被激活

even-parity: 偶校验 偶数个0输出1;

modulus=范围+1 (如 196~255 则为255-196=59 +1 =60)

ascending 上升 descending 下降

大题:

Transition/output table: 转化/输出表格, 需写出Q 和 Q*

右移, QA变成RIN

QCQBQA	QC*QB*QA*	RIN
110	101	1
101	010	0
010	100	0
100	001	1
001	011	1
011	110	0
000	001	1
111	110	0

先在第一列
写出序列

最后不在序
列中的找变
化最小的变
化, 把剩下
的情况补全

Excitation equations: 激励方程 (触发器D的表达式)

Transition/State equations: 过渡/状态方程 (Q*的表达式)

Output equation: 输出方程 (最终输出F输出的表达式)

Logic diagram: 逻辑框图 Logic expression : 逻辑表达式 Timing diagram: 时序关系图

indicate the modulus: 计算模数

题目

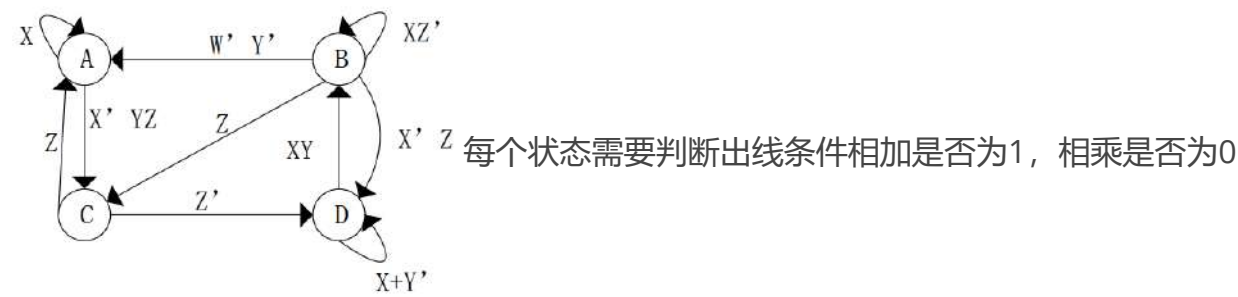
To design a "110111" serial sequence generator, (3) flip-flops are need at least （没有提到移位寄存器） $2^n > \text{modulus}$

如果提到 by shift registers 需要根据是否重复来判断需要几个

A ROM with 1024 bits capacity has 256 words. It can be used to implement a combinational circuit with (8) inputs and (4) outputs （n inputs 有 2^n words ）

A 16Kx8 ROM, which can implement a combinational circuit with (14) inputs and(8) outputs at most, can be built by (4) 8Kx4 ROMs . （因为16K是 $2^4 \times 2^{10} = 2^{14}$, 8K是 $2^3 \times 2^{10} = 2^{13}$ ）

the state diagram shown in Fig. 5 has (C) ambiguous states



To build a modulus -191 counter, it requires at least (3) 74x162s （因为162是模10 的计数器，需要 $10^n > 191$ 则n=3）