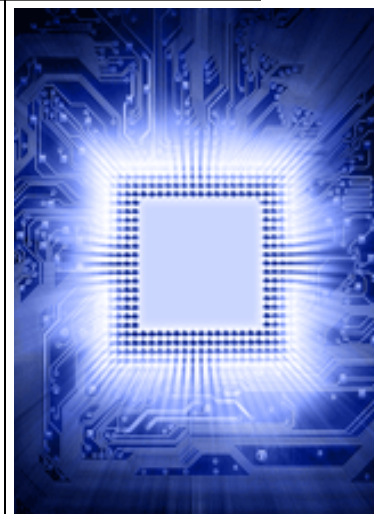
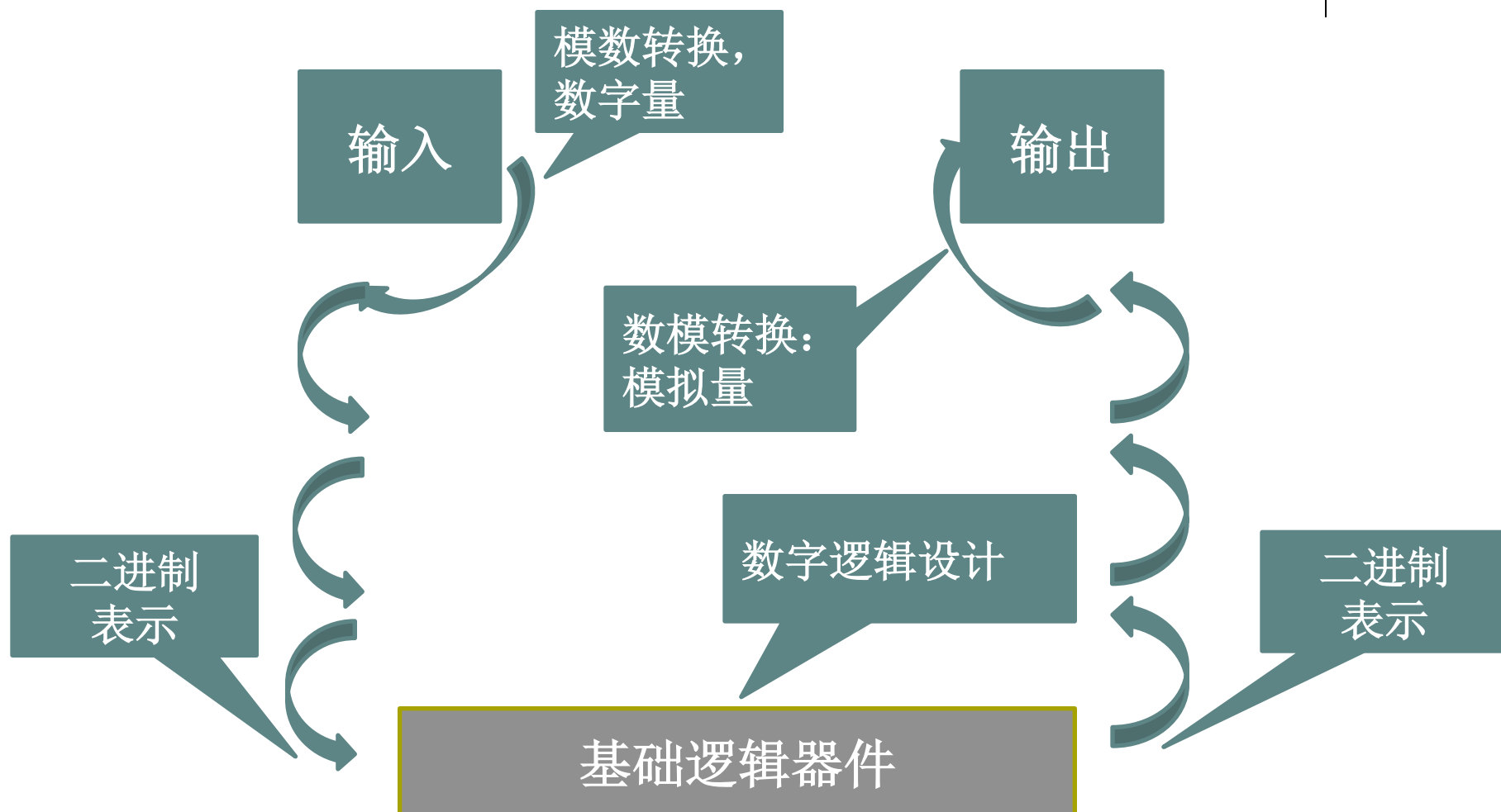
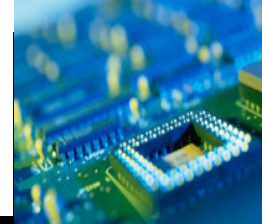


第4章

组合逻辑设计原理

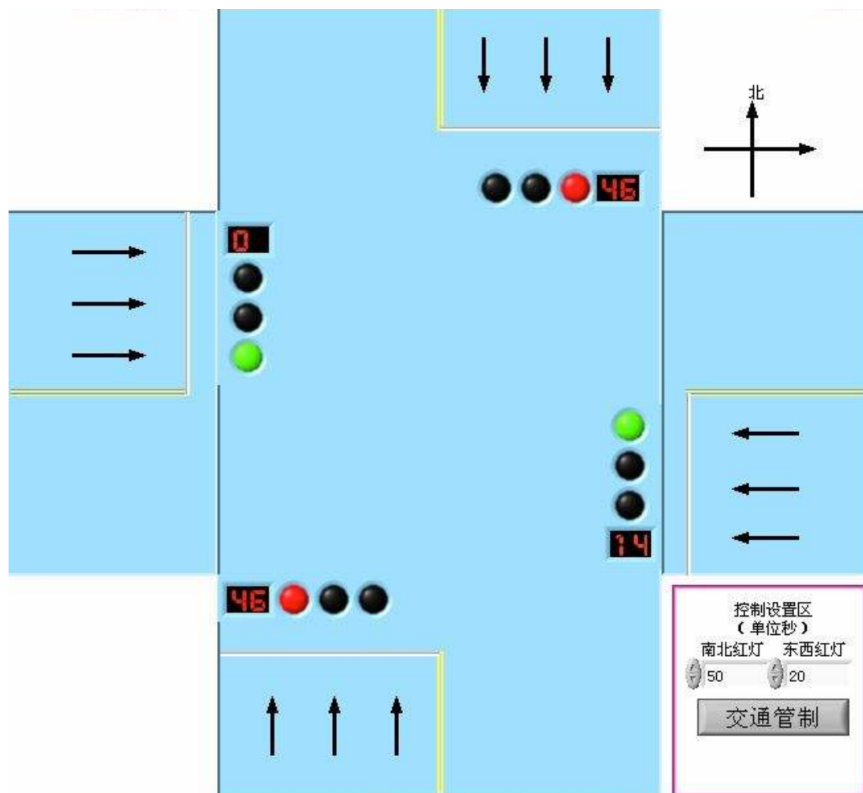
南京大学人工智能学院
2018-2019 春季







例子：交通灯控制



状态	功能
000	初态
001	南北
010	南北等
011	东西
100	东西等

现态	次态/ 控制
000	001/010010
001	010/100001
010	011/010001
011	100/001100
100	000/001010

要推导出输入到输出的函数。
要用电路实现函数。



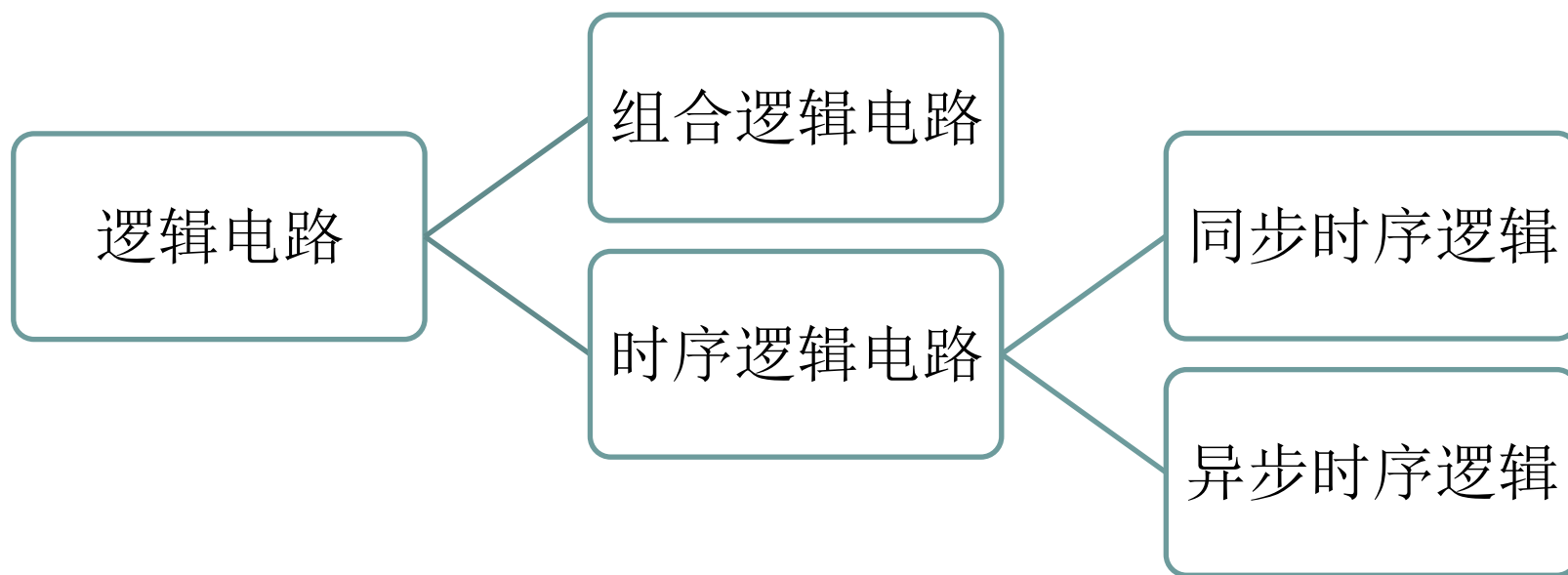
数字逻辑设计



- 逻辑设计问题：
 - 输入：M个二进位
 - 输出：N个二进位
 - 处理：输入到输出的映射
- 映射关系的实现——逻辑电路
 - 二进制的运算
 - 二进制编码转换成检错码、纠错码
 - 指令到动作
 - 现态到次态
- 需求的抽象、归纳、总结

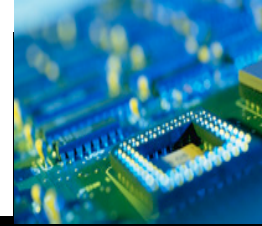


课程导引



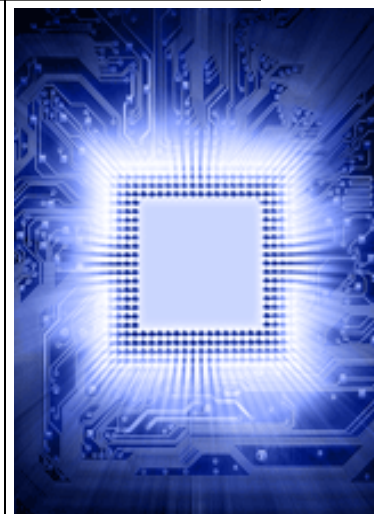


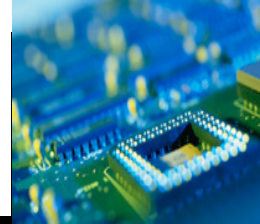
主要内容



- 开关代数
- 组合电路分析
- 组合电路综合
- 定时冒险

1 开关代数





- 1854年, George Boole, *An Investigation of the Laws of Thought, on which are Founded the Mathematical Theories of Logic and Probabilities*
 - 提出了布尔代数, “基于人类逻辑思考的本性”, 将思想翻译成符号。并且指出, 这些符号只需要两个值, 即0和1
- 1938年, Claude E. Shannon, *A Symbolic Analysis of Relay and Switching Circuits*, 硕士论文
 - 提出将布尔代数用于分析和优化继电器逻辑电路



公理 (Axiom)



- 公理 这些公理已经完备地描述了布尔代数。

(A1) 如果 $X \neq 1$, 则 $X=0$; (A1') 如果 $X \neq 0$, 则 $X=1$

(A2) 如果 $X=0$, 则 $X'=1$; (A2') 如果 $X=1$, 则 $X'=0$

(A3) $0 \cdot 0 = 0$

(A3') $1 + 1 = 1$

(A4) $1 \cdot 1 = 1$

(A4') $0 + 0 = 0$

(A5) $0 \cdot 1 = 1 \cdot 0 = 0$

(A5') $0 + 1 = 1 + 0 = 1$

与门表示法: 逻辑乘, 符号 “.”、“ \wedge ”、“&”、“and”

或门表示法: 逻辑加, 符号 “+”、“ \vee ”、“|”、“or”

非门表示法: 取反, 符号 “'”、“ \neg ”、“ \sim ”、“!”、“not”

运算优先级: 取反、先乘后加

Verilog HDL

VHDL



单变量定理



(T1)	$X + 0 = X$	(T1')	$X \cdot 1 = X$	自等律
(T2)	$X + 1 = 1$	(T2')	$X \cdot 0 = 0$	0-1律
(T3)	$X + X = X$	(T3')	$X \cdot X = X$	同一律
(T4)	$(X')' = X$			还原律
(T5)	$X + X' = 1$	(T5')	$X \cdot X' = 0$	互补律

- 完备归纳法证明
 - 证明 $x=0$ 和 $x=1$ 时，定理正确



二变量和三变量定理



$$(T6) \quad X + Y = Y + X$$

$$(T6') \quad X \cdot Y = Y \cdot X$$

交换律

$$(T7) \quad (X + Y) + Z = X + (Y + Z)$$

$$(T7') \quad (X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$$

结合律

$$(T8) \quad X \cdot Y + X \cdot Z = X \cdot (Y + Z)$$

$$(T8') \quad (X + Y) \cdot (X + Z) = X + Y \cdot Z$$

分配律

$$(T9) \quad X + X \cdot Y = X$$

吸收律其它形式:

$$X + X' \cdot Y = X + Y$$

$$X \cdot (X' + Y) = X \cdot Y$$

$$(T9') \quad X \cdot (X + Y) = X$$

吸收律

$$(T10) \quad X \cdot Y + X \cdot Y' = X$$

$$(T10') \quad (X + Y) \cdot (X + Y') = X$$

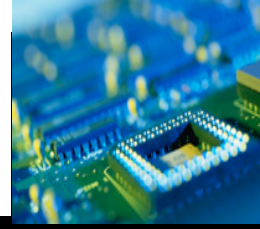
组合律

$$(T11) \quad X \cdot Y + X' \cdot Z + \boxed{Y \cdot Z} = X \cdot Y + X' \cdot Z$$

$Y \cdot Z$ 称为一致项，若 $Y \cdot Z$ 为1，
则 $X \cdot Y$ 和 $X' \cdot Z$ 必有一个为1

一致律

$$(T11') \quad (X + Y) \cdot (X' + Z) \cdot (Y + Z) = (X + Y) \cdot (X' + Z)$$



$$(T12) \quad X + X + \dots + X = X$$

广义同一律

$$(T12') \quad X \cdot X \cdot \dots \cdot X = X$$

$$(T13) \quad (X_1 \cdot X_2 \cdot \dots \cdot X_n)' = X_1' + X_2' + \dots + X_n'$$

德·摩根定理

$$(T13') \quad (X_1 + X_2 + \dots + X_n)' = X_1' \cdot X_2' \cdot \dots \cdot X_n'$$

$$(T14) \quad [F(X_1, X_2, \dots, X_n, +, \cdot)]' = F(X_1', X_2', \dots, X_n', \cdot, +)$$

广义德·摩根定理

$$(T15) \quad F(X_1, X_2, \dots, X_n) = X_1 \cdot F(1, X_2, \dots, X_n) + X_1' \cdot F(0, X_2, \dots, X_n)$$

香农展开定理

$$(T15') \quad F(X_1, X_2, \dots, X_n) = [X_1 + F(0, X_2, \dots, X_n)] \cdot [X_1' + F(1, X_2, \dots, X_n)]$$

● 使用归纳法证明

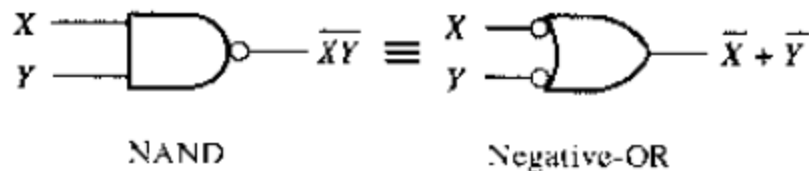


德·摩根定理

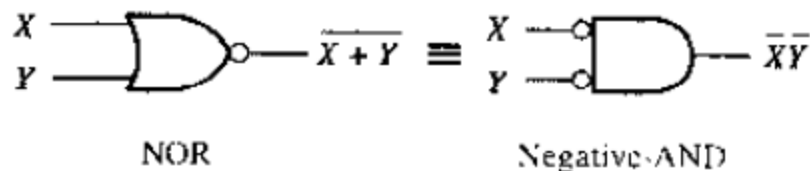


- Augustus De Morgan (1806 - 1871)
 - 和George Boole一起，是符号逻辑(Symbolic Logic)的奠基人
- 德·摩根定理
 - 变量乘积取反等于将每个变量取反，然后再求其和
 - 变量求和取反等于将每个变量取反，然后再求其积

$$\overline{X \cdot Y} = \overline{X} + \overline{Y}$$



$$\overline{X + Y} = \overline{X} \cdot \overline{Y}$$

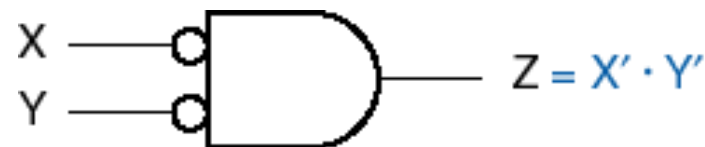
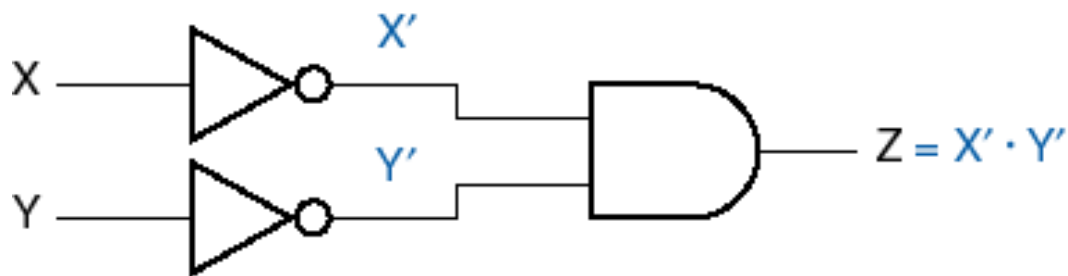
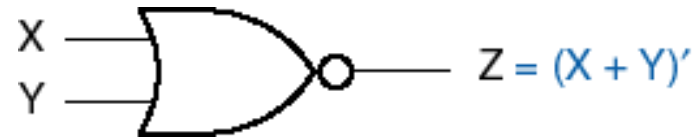
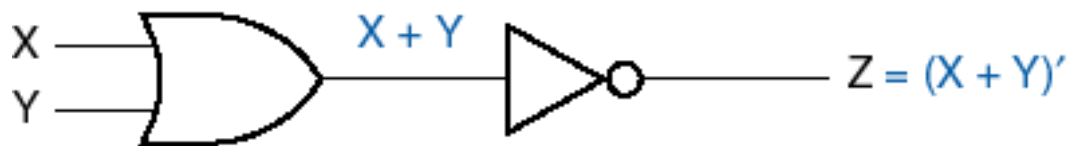
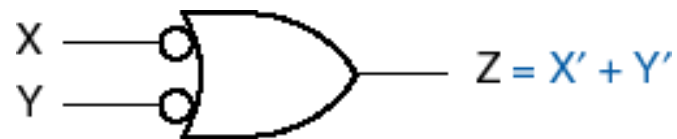
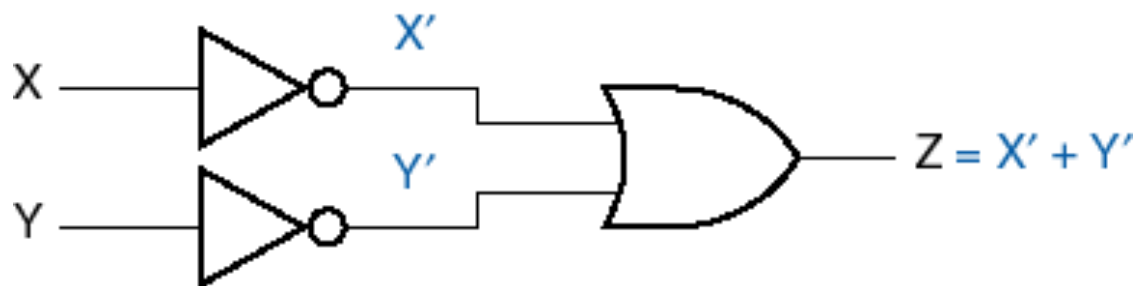
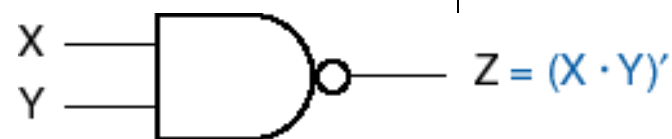
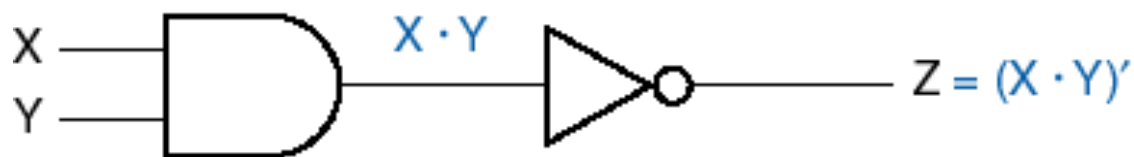




德·摩根定理



与非门或非门等效电路





德·摩根定理的应用



- **Examples:**

- $(a + b \cdot c)' = (a + (b \cdot c))'$

$$= a' \cdot (b \cdot c)' \quad [T13']$$

$$= a' \cdot (b' + c') \quad [T13]$$

$$= a' \cdot b' + a' \cdot c' \quad [T8]$$

- $(a \cdot (b + z \cdot (x + a')))' = a' + (b + z \cdot (x + a'))'$

[T13']

$$= a' + b' \cdot (z \cdot (x + a'))'$$

[T13]

$$= a' + b' \cdot (z' + (x + a'))'$$

[T13]

$$= a' + b' \cdot (z' + x' \cdot (a'))'$$

[T13']

$$= a' + b' \cdot (z' + x' \cdot a)$$

[T4]

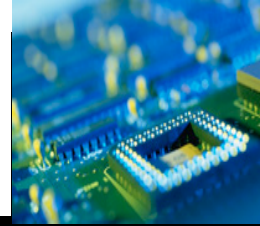
$$= a' + b' \cdot z' + b' \cdot x' \cdot a$$

[T8]

$$= a' + b' \cdot z' + b' \cdot x'$$

[T9]

三个基本定律



1. 代入定理

- 所谓代入定理，是指在逻辑等式中任何一个变量A，都可以用任意逻辑表达式代入，则等式仍然成立。

2. 反演定理

- 所谓反演定理，是指对于任意一个逻辑式Y，若将其中的所有“ \cdot ”与“ $+$ ”互换，“0”和“1”互换，原变量与反变量互换，则得到的结果就是原函数的反函数 Y' 。
 - 需遵守“先括号，然后与，最后或”的运算优先次序。
 - 不属于单个变量上的反号应保留不变。



三个基本定律



3. 对偶定理

- 若两逻辑式相等，则它们的对偶式也相等，这就是对偶定理。
- 所谓对偶式，即：对于任何一个逻辑式Y，若将其中的“.”与“+”互换，“0”和“1”互换，则得到Y的对偶式Y^D，或者Y与Y^D互为对偶式。
- 保持运算优先次序不变

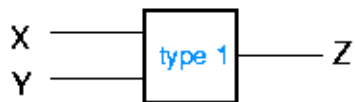
$$X + X \cdot Y = X$$

对偶式

$$X \cdot (X + Y) = X$$

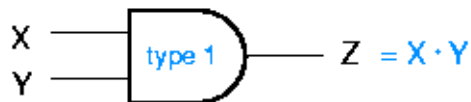


对偶式的应用



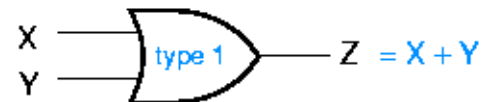
X	Y	Z
LOW	LOW	LOW
LOW	HIGH	LOW
HIGH	LOW	LOW
HIGH	HIGH	HIGH

Electric function



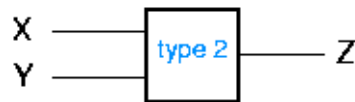
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Positive-logic

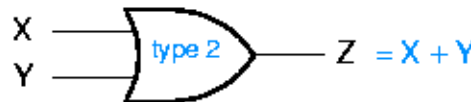


X	Y	Z
1	1	1
1	0	1
0	1	1
0	0	0

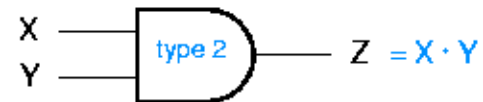
Negative-logic



X	Y	Z
LOW	LOW	LOW
LOW	HIGH	HIGH
HIGH	LOW	HIGH
HIGH	HIGH	HIGH



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1



X	Y	Z
1	1	1
1	0	0
0	1	0
0	0	0



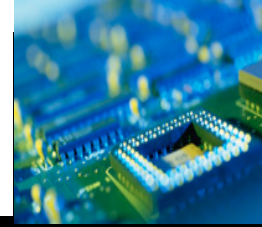
逻辑函数化简基本方法



- 比较常用的办法：
 - 并项法: $AB + A'B = B$
 - 吸收法: $AB + B = B$
 - 消去法: $A + A'B = A + B$
 - 配项法: $AB + A'C + BC = AB + A'C$
- 摩根定理

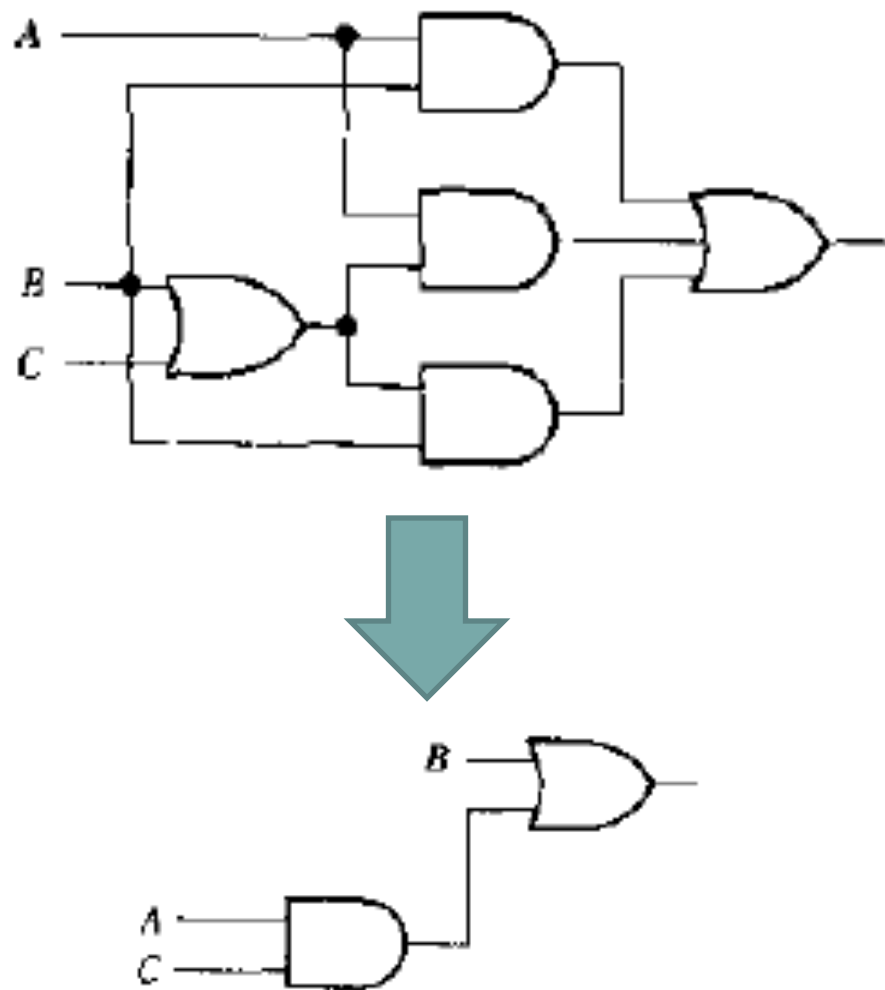


逻辑表达式的简化



- 化简如下表达式

$$\begin{aligned} & A \cdot B + A \cdot (B + C) + B \cdot (B + C) \\ &= A \cdot B + A \cdot B + A \cdot C + B \cdot B + B \cdot C \\ &= A \cdot B + A \cdot C + B + B \cdot C \\ &= B \cdot (A + 1 + C) + A \cdot C \\ &= B + A \cdot C \end{aligned}$$





4.1.6 逻辑函数的标准表示法

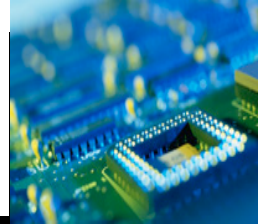


- 术语

- 文字(literal): 一个自变量或反变量。 X, X'
- 乘积项(product term): 单个文字或2个以上文字的逻辑积。如: $X, X \cdot Y \cdot Z', W' \cdot Y' \cdot Z$;
- 积之和(sum of product, SOP): 是乘积项的逻辑和, 如 $X \cdot Y \cdot Z' + W' \cdot Y' \cdot Z$
- 求和项(sum of term): 单个文字或2个以上文字的逻辑和。如: $X, X + Y + Z'$
- 和之积(product of sum, POS): 求和项的逻辑积, 如 $(X + Y + Z') \cdot (W' + Y' + Z)$
- 标准项(normal term): 是一个乘积项或求和项, 每个变量当且仅当只出现一次。



逻辑函数的标准表示法



- **n变量最小项(minterm):** 具有n个文字的标准乘积项。
 - 只有一个输入组合可以使最小项的值为**1**。
 - 该组合的二进制值就是最小项的编号。
 - 例如：对于4变量的逻辑函数，W、X、Y、Z四个逻辑变量，有**16**个最小项。
 - $W'X'Y'Z'$ 只有在各个变量分别等于0000时才为**1**，因此其编号是0，记为 m_0
 - $WXYZ$ 只有在各个变量分别为1111时才为1，因此其编号为 $(1111)_2$ ，即15，记为 m_{15}
 - 简单的编号方法：**原变量取1，反变量取0**，即可得到编号
 - 例如： $WX'Y'Z$ 的编号是 $(1001)_2$ ，因此是 m_9



逻辑函数的标准表示法

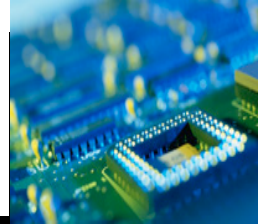


Minterm	Minterm Code	Minterm Number
$A'B'C'$	000	m_0
$A'B'C$	001	m_1
$A'BC'$	010	m_2
$A'BC$	011	m_3
$AB'C'$	100	m_4
$AB'C$	101	m_5
ABC'	110	m_6
ABC	111	m_7

$$\sum_{i=0}^{2^n-1} m_i = 1$$



逻辑函数的标准表示法



- **n变量最大项(maxterm):** 具有n个文字的标准求和项。
 - 只有一个输入组合可以使最大项的值为**0**。
 - 该组合的二进制值就是最大项的编号。
 - 例如：对于4变量的逻辑函数，W、X、Y、Z四个逻辑变量，有16个最大项
 - $(W'+X'+Y'+Z')$ 只有在各个变量分别等于1111时才为**0**，因此其编号是 $(1111)_2$ ，即15，记为 M_{15}
 - $(W+X+Y+Z)$ 只有在各个变量分别为0000时才为0，因此其编号为 $(0000)_2$ ，记为 M_0
 - 简单的编号方法：**原变量取0，反变量取1**，即可得到编号
 - 例如： $W+X'+Y'+Z$ 的编号是 $(0110)_2$ ，因此是 M_6



逻辑函数的标准表示法



Maxterm	Maxterm Code	Maxterm Number
$A+B+C$	000	M_0
$A+B+C'$	001	M_1
$A+B'+C$	010	M_2
$A+B'+C'$	011	M_3
$A'+B+C$	100	M_4
$A'+B+C'$	101	M_5
$A'+B'+C$	110	M_6
$A'+B'+C'$	111	M_7

$$\prod_{i=0}^{2^n-1} M_i = 0$$



逻辑函数的标准表示法



- 相同变量相同编号的最小项和最大项互为反函数。
- $(m_i)' = M_i$
- 反演规则
- 标准规范式：函数全部由最小项或最大项组成的表示式。
- $f(A,B,C) = AB + AC' + A'C$
 - $AB = ABC' + ABC = m_6 + m_7$
 - $AC' = AB'C' + ABC' = m_4 + m_6$
 - $A'C = A'B'C + A'BC = m_1 + m_3$
 - Therefore,
$$f(A,B,C) = (m_6 + m_7) + (m_4 + m_6) + (m_1 + m_3) = \Sigma m(1, 3, 4, 6, 7)$$



逻辑函数的标准表示法



- 标准规范式：函数全部由最小项或最大项组成的表示式。
 - $f(A,B,C) = AB + AC' + A'C \rightarrow$ 和之积形态
 - $f(A,B,C) = \sum m(1, 3, 4, 6, 7)$
 - $f(A,B,C) = \prod M(0, 2, 5)$
- 函数积之和与和之积表示形态之间的关系：
 - $f(A,B,C) = \sum m(0,2,4,6) = \prod M(1,3,5,7)$
 - $f'(A,B,C) = \sum m(1,3,5,7) = \prod M(0,2,4,6)$



逻辑函数的标准表示法

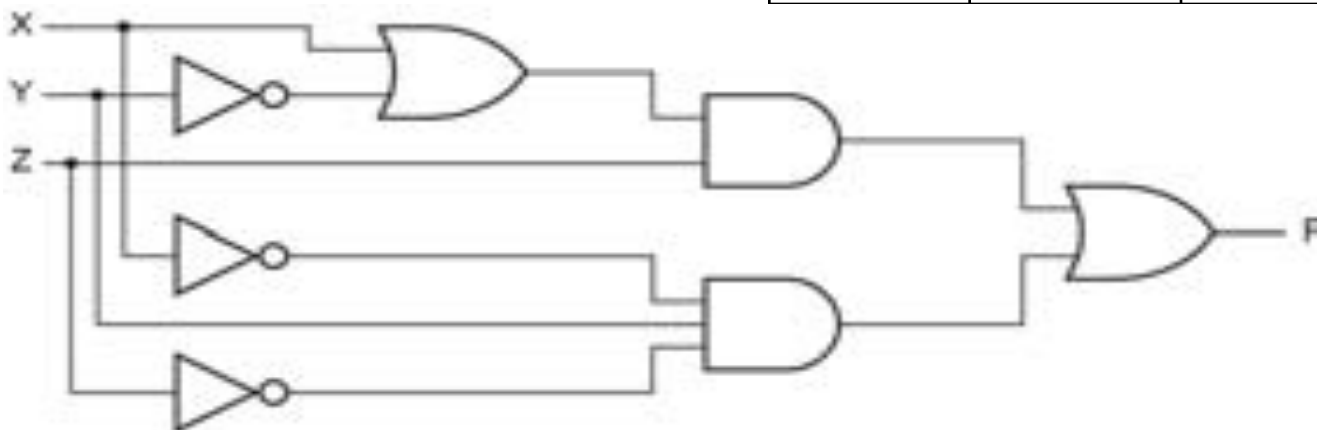


- 逻辑函数的其它表示方法:

- 真值表

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

- 电路图



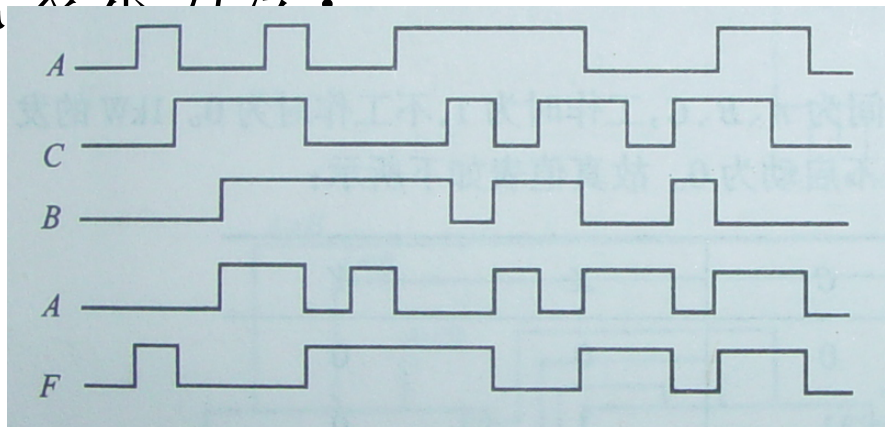


逻辑函数的标准表示法

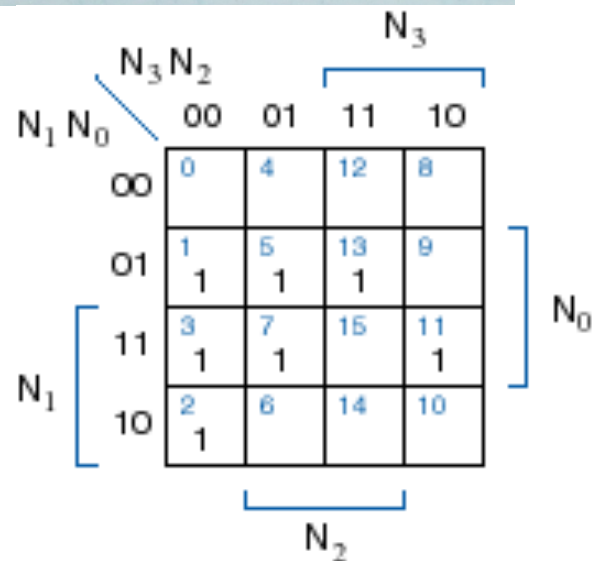


- 逻辑函数的其它表示方法.

- 真值表
- 电路图
- 波形图



- 卡诺图



$$F = \sum_{N_3, N_2, N_1, N_0} (1, 2, 3, 5, 7, 11, 13)$$



逻辑函数的标准表示法

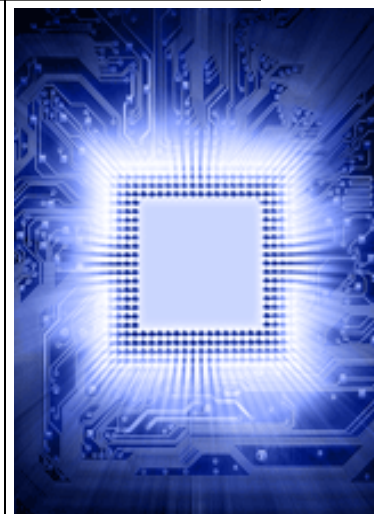


- 逻辑函数的其它表示方法:

- 真值表
- 电路图
- 波形图
- 卡诺图
- HDL描述

```
module MUX2_1 (out, a, b, sel) ;  
    output out ;  
    input a, b, sel ;  
        not #1 not1( sel_, sel);  
        and #2 and1( a1, a, sel_);  
        and #2 and2( b1, b, sel);  
        or #1 or1( out, a1, b1);  
endmodule
```

2 组合逻辑电路分析





组合逻辑电路分析与设计



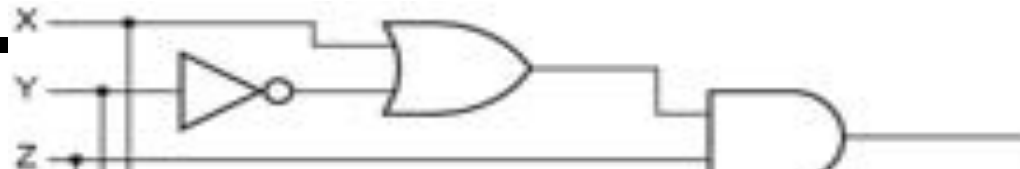
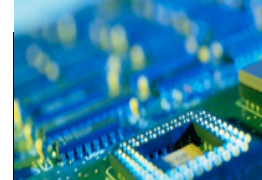
分析Analysis

- 电路描述与设计
 - 文字描述
 - 分析因果关系，确定输入输出量，定义逻辑状态
 - 罗列真值表(或时序图)
- 逻辑演算
 - 逻辑函数形式(**SOP**或**POS**)
 - 逻辑化简
- 逻辑电路和逻辑图

综合Synthesis



组合电路分析



注意：

通常情况下需要对电路的功能进行描述。

如：***投票电路，***校验电路，**加法器等

补例

X	Y	Z	F
0	1	0	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Z'

积之和的形式：

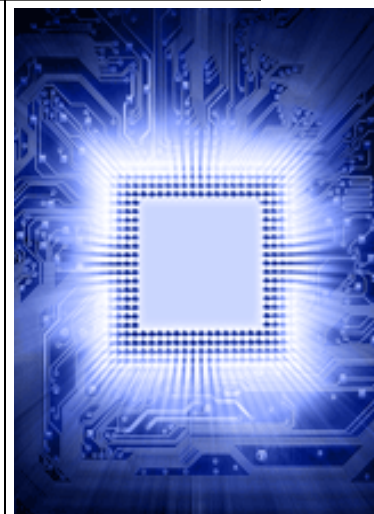
$$F = X \cdot Z + Y' \cdot Z + X' \cdot Y \cdot Z'$$

和之积的形式：

$$F = (X + Y' + Z') \cdot (X' + Z) \cdot (Y + Z)$$

穷举法： $F = X' \cdot Y' \cdot Z + X' \cdot Y \cdot Z' + X \cdot Y' \cdot Z + X \cdot Y \cdot Z$

3 组合电路的综合





组合逻辑电路分析与设计



分析Analysis

- 电路描述与设计
 - 文字描述
 - 分析因果关系，确定输入输出量，定义逻辑状态
 - 罗列真值表(或时序图)
- 逻辑演算
 - 逻辑函数形式(**SOP或POS**)
 - 逻辑化简
- 逻辑电路和逻辑图

综合Synthesis



组合逻辑电路设计

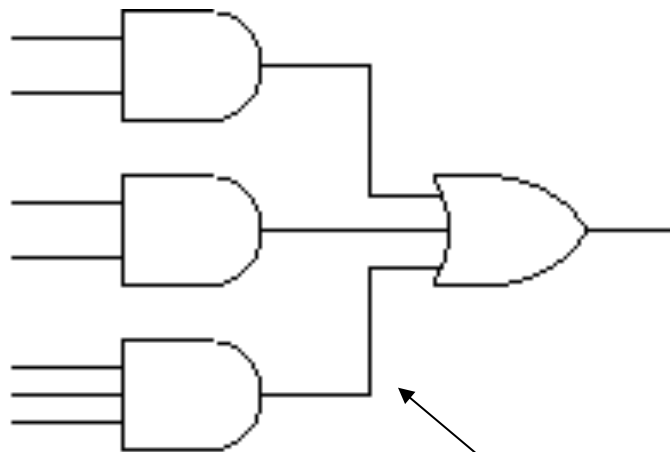
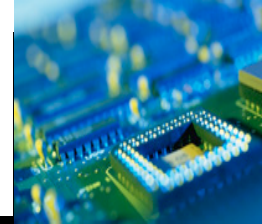


- 一般采用两级电路(Two-level Circuits)
 - 输入信号通过两级门到达输出信号;
 - 采用多于两级是基于扇入(Fan-in)限制或电路速度方面的考虑。
- 逻辑化简
 - AND-OR门网络, 表达式用SOP形式
 - OR-AND门网络, 表达式用POS形式
- 在大多数系统中, 与非门和或非门的速度比与门和或门速度快。

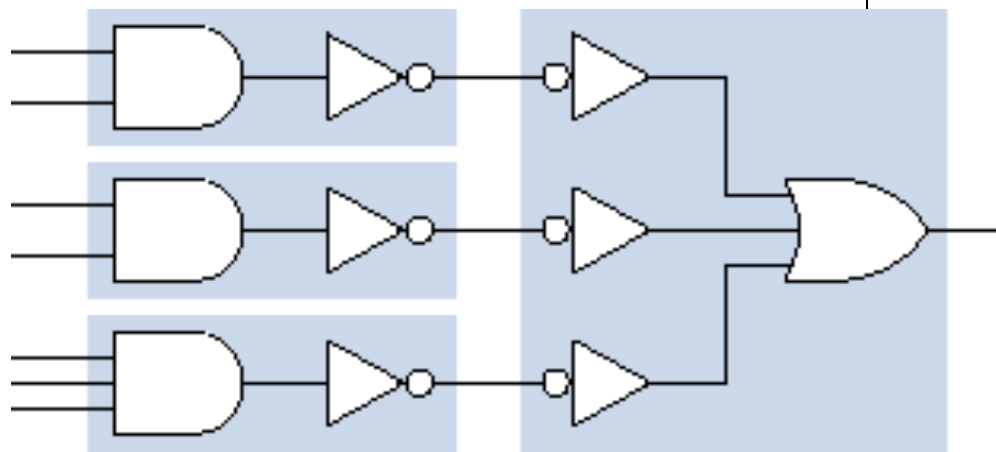


积之和

(举例)



AND-OR



NAND-NAND

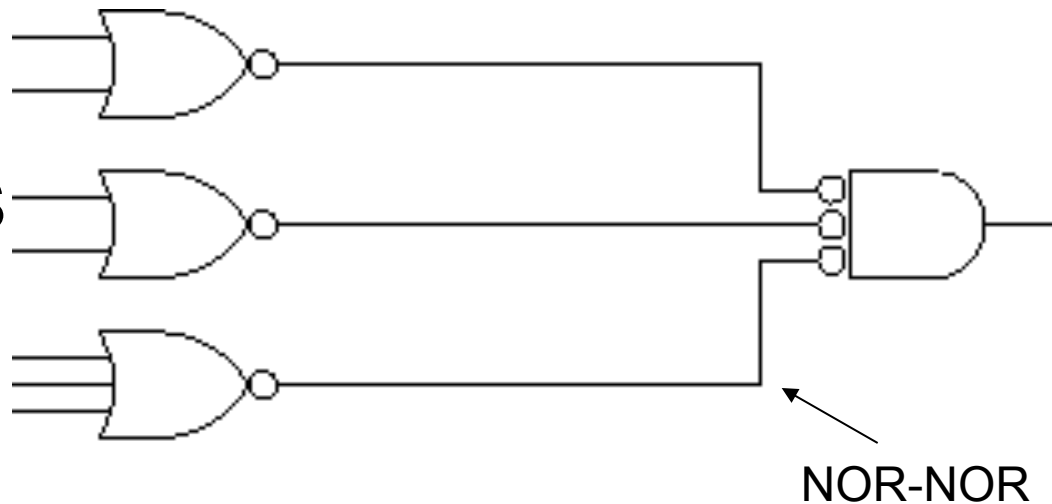
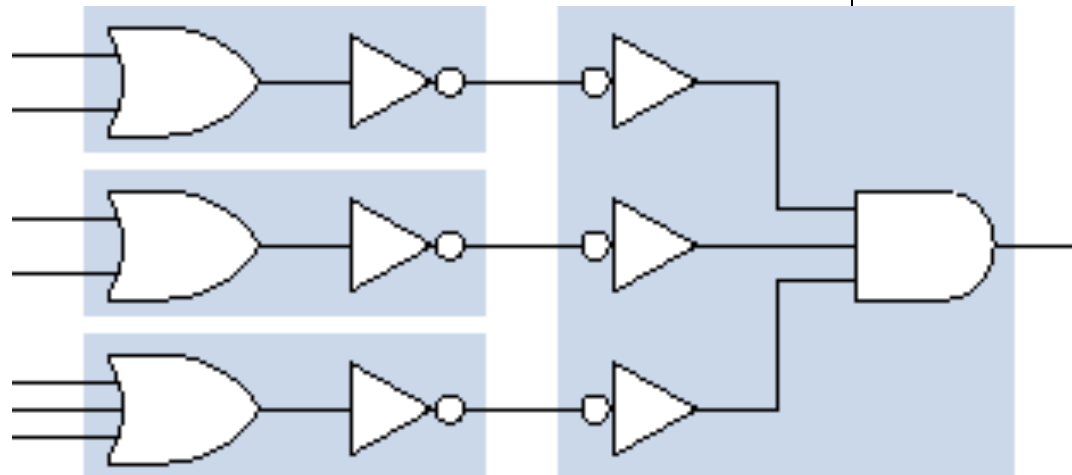
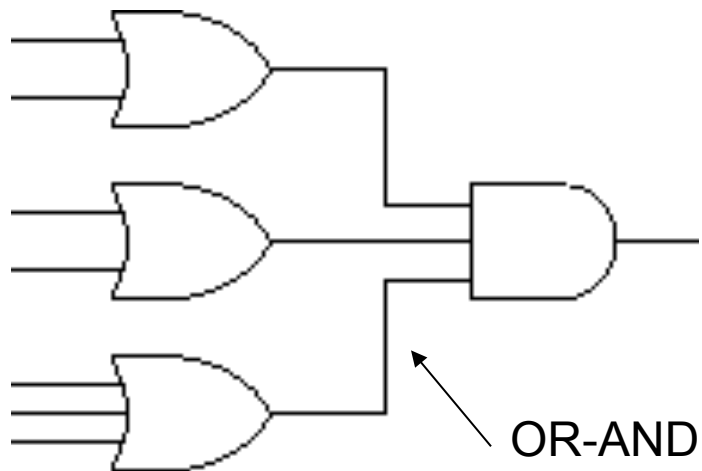
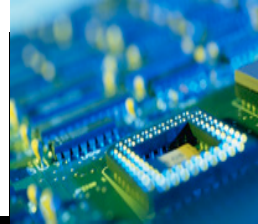
积之和

- Sum of Product: SOP
- 一系列乘积的和式
- 可以用与非门实现
- 有多种不同形式实现



和之积

(举例)



和之积

- Product of Sum: POS
- 一系列和的乘积
- 可以用或非门实现
- 有多种不同形式实现



强力设计 Brute-force design



- 真值表设计：
 - 列出所有的输入输出项
- 例如：素数检测器
 - 4-bit input, $N_3N_2N_1N_0$

$F =$

$\Sigma_{N_3N_2N_1N_0}(1,2,3,5,7,11,13)$

设计一个投票电路

row	N_3	N_2	N_1	N_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	0	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1



组合电路最小化



- **ASIC设计和PLD设计中，化简都很重要**
 - 门数多，芯片面积大
 - 门输入端数多，芯片面积大
 - 最小化可以降低成本
- 最小化两级“与-或”、“或-与”、“与非-与非”或“或非-或非”电路的方法：
 - 最小化第一级门的数目。
 - 最小化每个第一级门的输入端数目。
 - 最小化第二级门的输入数目。
- 化简时不考虑输入反相器成本，一般输入变量及其反码都是现成的——尤其对于**PLD设计**



电路最小化方法



- 观察法
 - 布尔代数 **Boolean simplification**
 - 卡诺图 **K-maps simplification**
- 程序自动 **Automating simplification**
 - 奎因-穆克鲁斯基算法 **Quine-McCluskey Algorithm**
 - 探测法，如 **Espresso Method**



代数化简



- 利用结合律T10, 减少逻辑门数和输入端数。

$$F = \Sigma_{N_3 N_2 N_1 N_0} (1, 3, 5, 7, 2, 11, 13)$$

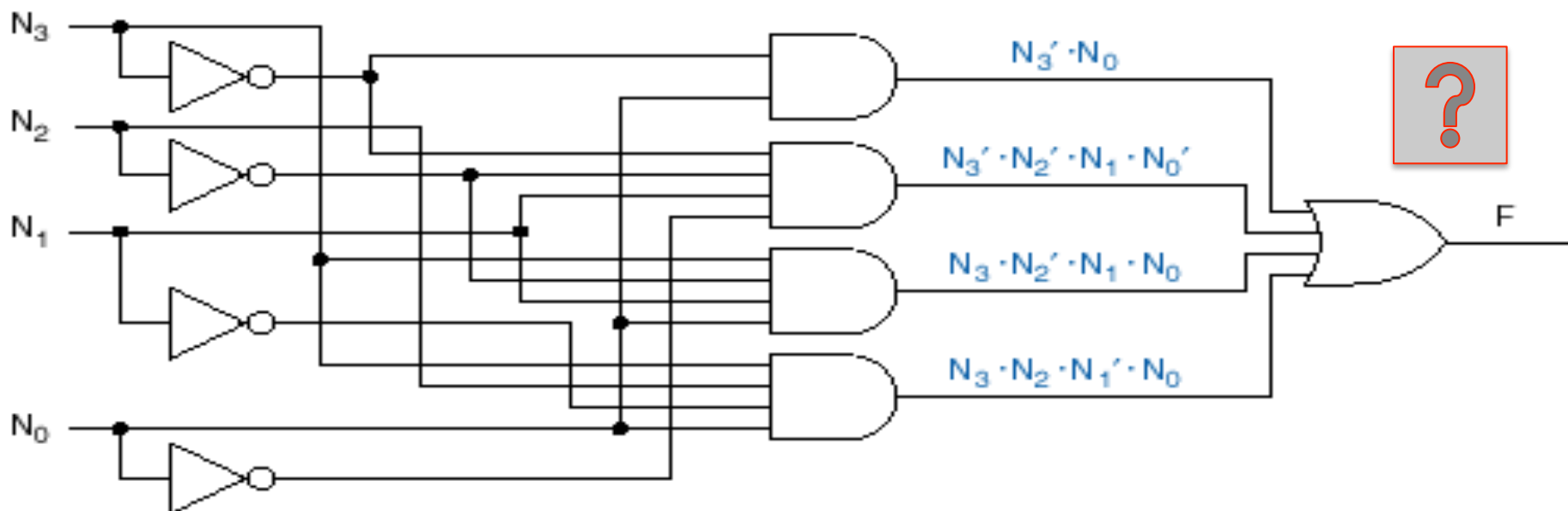
$$X \cdot Y + X \cdot Y' = X$$

$$(X + Y) \cdot (X + Y') = X$$

$$= N_3' \cdot N_2' \cdot N_1' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0 + N_3' \cdot N_2 \cdot N_1' \cdot N_0 + N_3' \cdot N_2 \cdot N_1 \cdot N_0 + \dots$$

$$= (N_3' \cdot N_2' \cdot N_1' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0) + (N_3' \cdot N_2 \cdot N_1' \cdot N_0 + N_3' \cdot N_2 \cdot N_1 \cdot N_0) + \dots$$

$$= N_3' \cdot N_2' \cdot N_0 + N_3' \cdot N_2 \cdot N_0 + \dots$$





卡诺图 (Karnaugh Map: K-MAP)



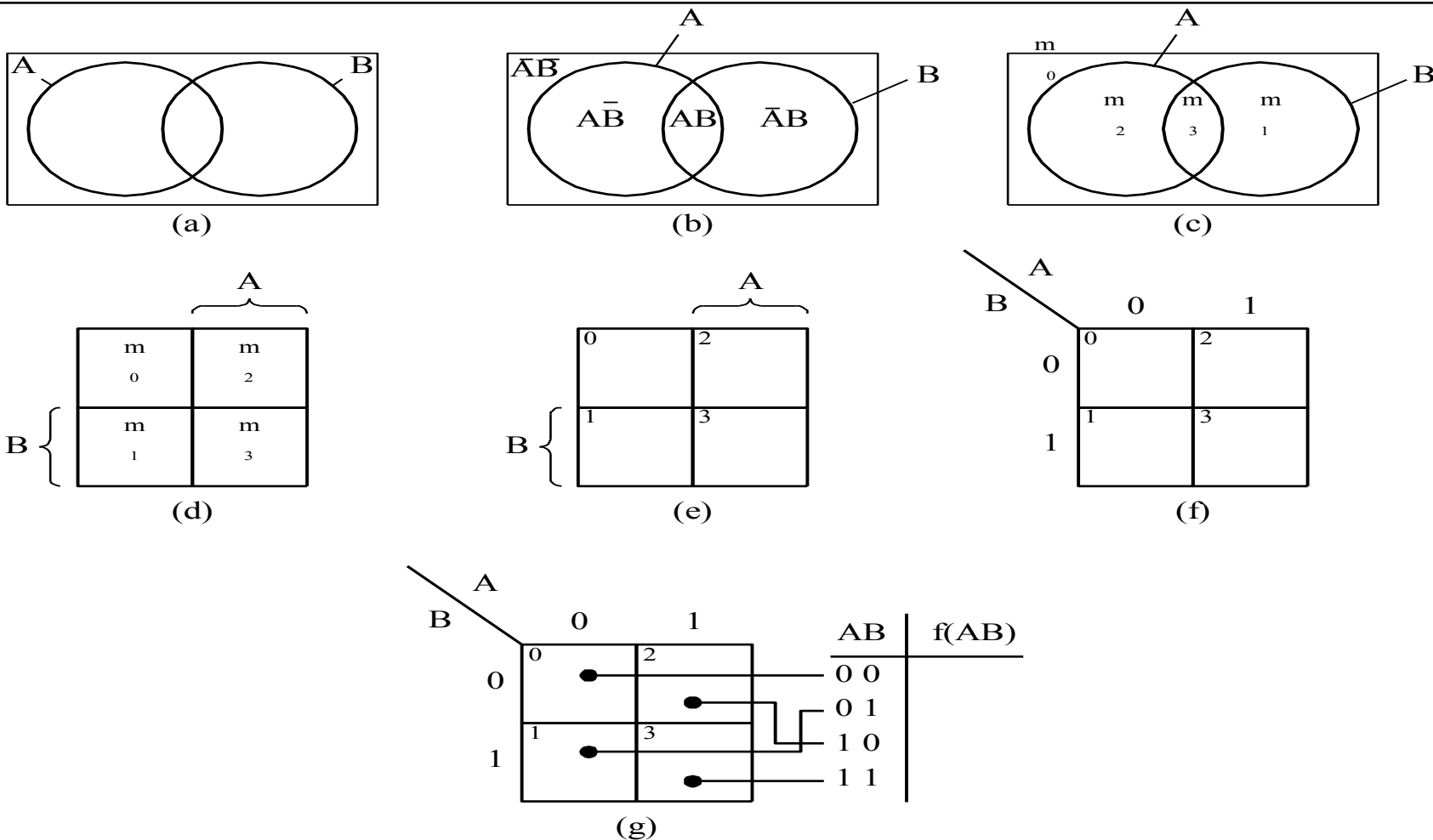
- 卡诺图是逻辑函数真值表的图形化，每一项对应真值表中的一行。
- n 输入的卡诺图是一个含有 2^n 个单元的矩阵图，每个单元代表在一个输入组合或最小项；
- 卡诺图的行和列都做了标记，行/列的标头确定该单元对应的输入组合；
- 单元中的小数字表示真值表中**最小项**的编号；
- 相邻的单元只有**一个**变量不同；
- 可以很方便地写出函数的**最小项和**或**最大项积**的形式。



二变量卡诺图



- 2变量的逻辑函数 $F(X,Y)$ 和卡诺图的关系

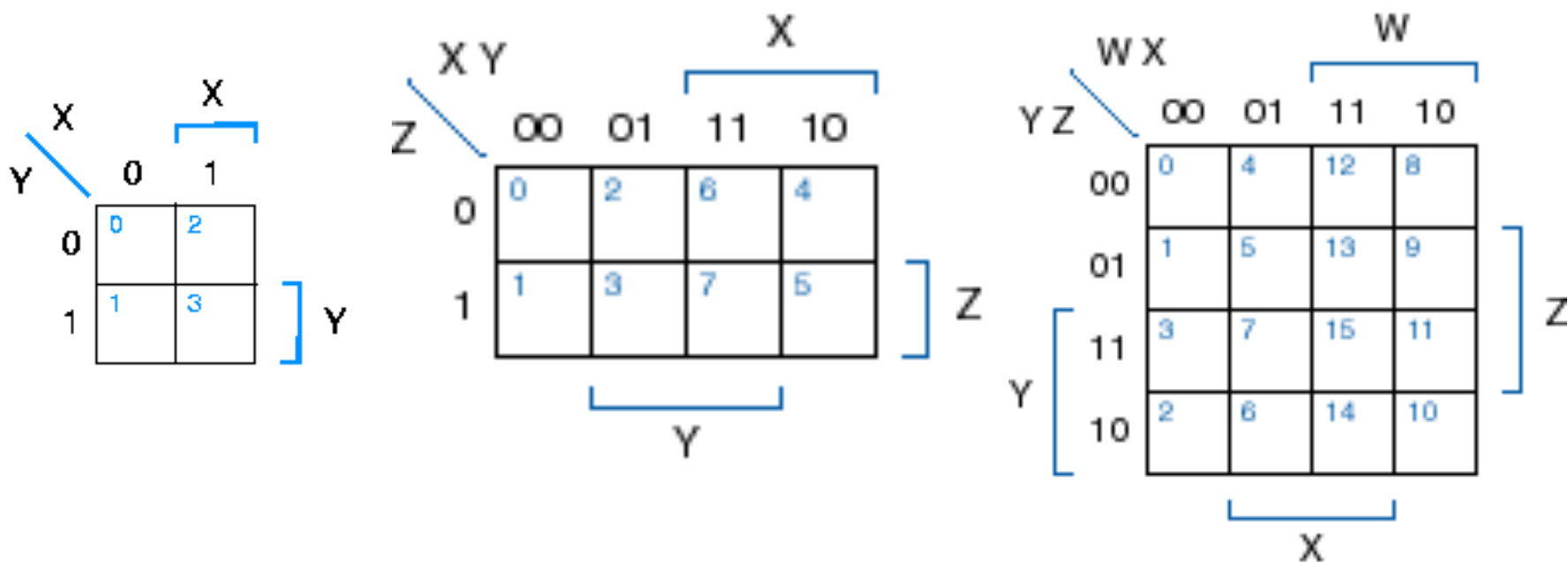




卡诺图



- 2、3、4变量的卡诺图结构





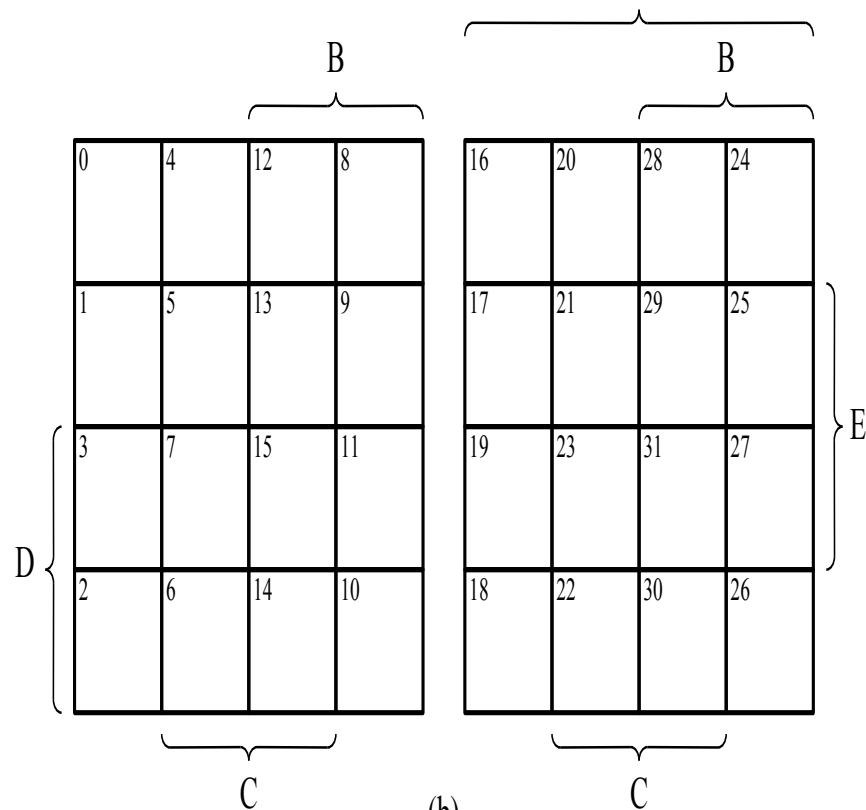
卡诺图的结构



- 5变量卡诺图的结构

ABC DE		000				001				011				010				100				101				111				110			
		000				001				011				010				100				101				111				110			
00		0				4				12				8				16				20				28				24			
01		1				5				13				9				17				21				29				25			
11		3				7				15				11				19				23								27			
10		2				6				14				10				18				22				30				26			

(a)



(b)



卡诺图的结构

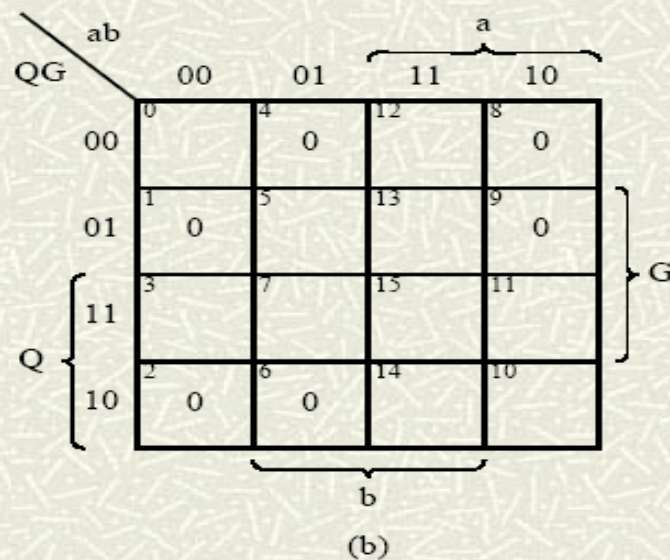
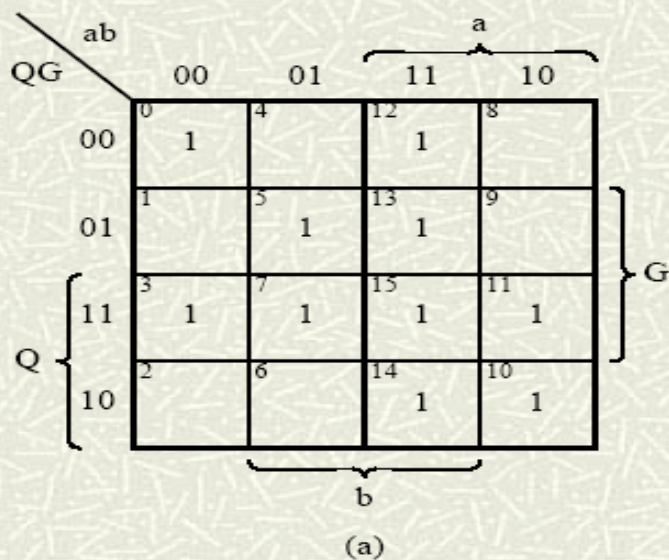


- 卡诺图的画法

(a) 最小项形式

(b) 最大项形式

$$f(a,b,Q,G) = \sum m(0,3,5,7,10,11,12,13,14,15) = \prod M(1,2,4,6,8,9)$$





利用卡诺图简化逻辑函数



- 将临近的“1”圈起来，每个圈中包含 2^i 个1， $i \geq 0$
- 每个圈代表一个乘积项，但不一定是标准项
- 圈所覆盖的区域：
 - 如果变量取值是1，则包含原变量；
 - 如果是0，则包含反变量；
 - 如果既有0，又有1，则不包含该变量

X \ Y	0	1
0	0	1
1	0	1



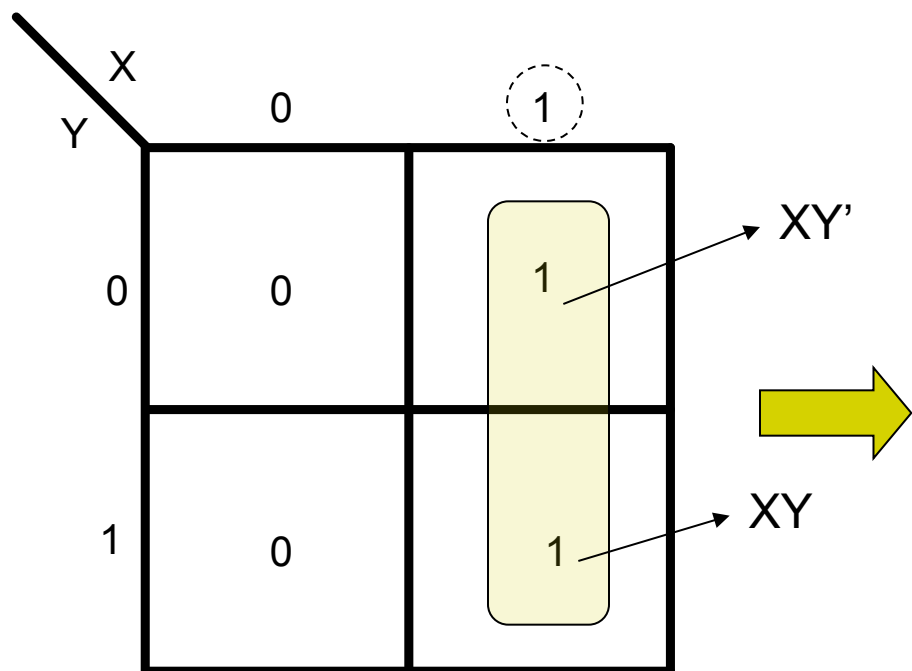
利用卡诺图简化逻辑函数



例: $F(X, Y) = XY' + XY = m_2 + m_3$

$$\begin{aligned} \square F &= XY + XY' \\ &= X(Y + Y') \\ &= X \end{aligned}$$

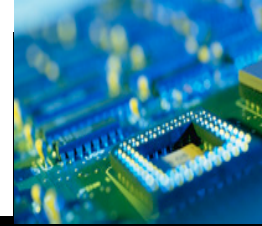
- 相邻的码字只有一位不同



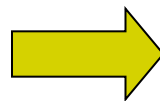
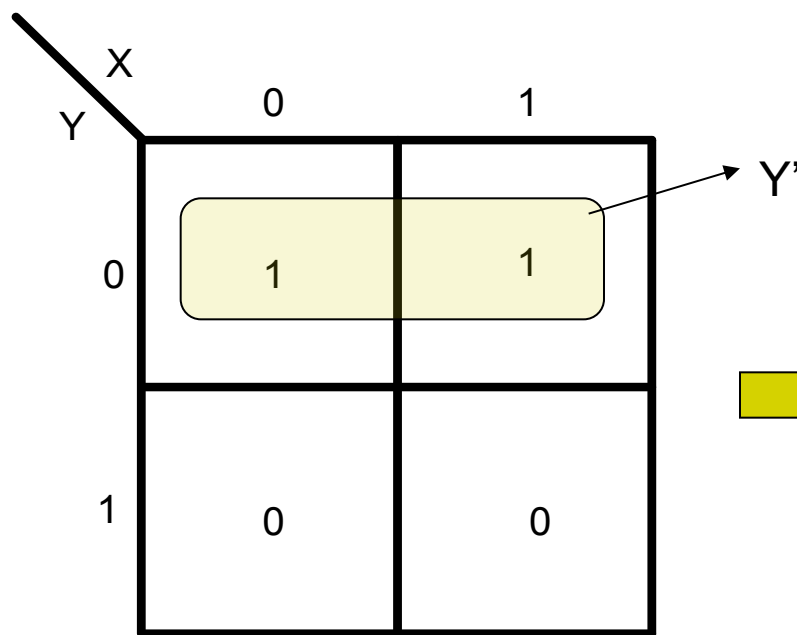
$$F = X$$



利用卡诺图简化逻辑函数



- $F = m_0 + m_2 = X'Y' + XY'$



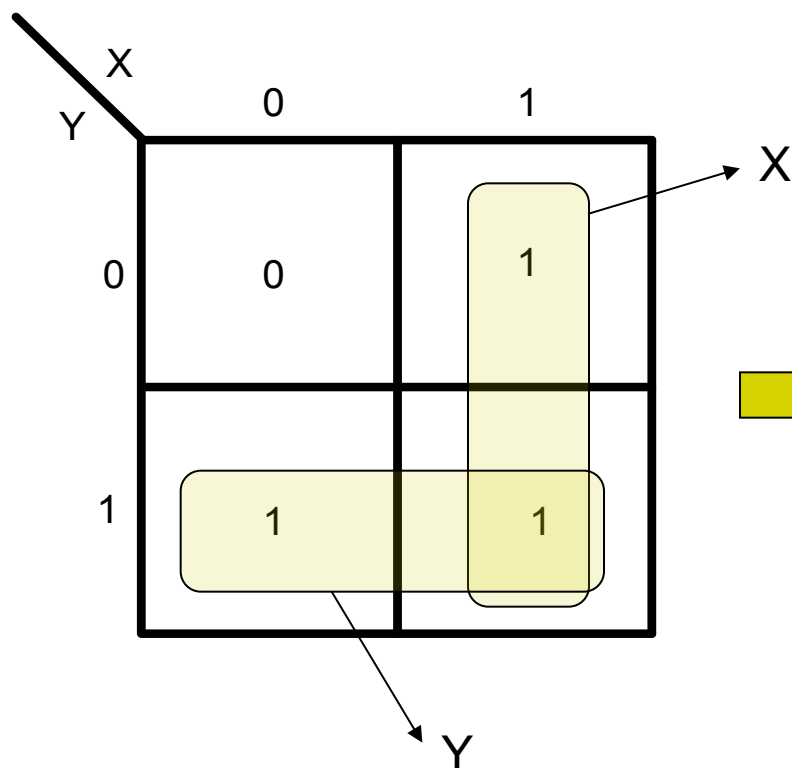
$$F = Y'$$



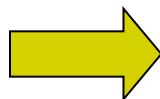
利用卡诺图简化逻辑函数



- $F = \sum (1,2,3) = XY + XY' + X'Y$



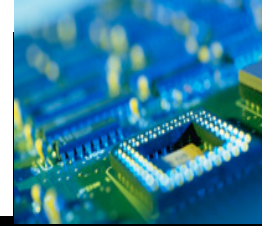
- $$\begin{aligned} F &= XY + XY' + X'Y \\ &= (XY + XY) + XY' + X'Y \\ &= X(Y + Y') + Y(X + X') \\ &= X + Y \end{aligned}$$



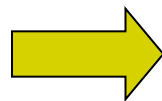
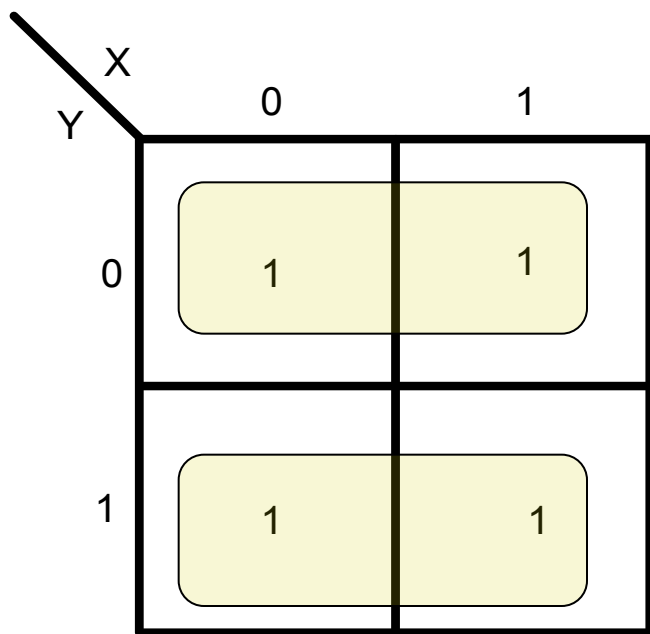
$$F = X + Y$$



利用卡诺图简化逻辑函数



- $F = \sum (1, 2, 3, 4) = XY + XY' + X'Y + X'Y'$



$$F = Y + Y'$$

$$F = 1$$

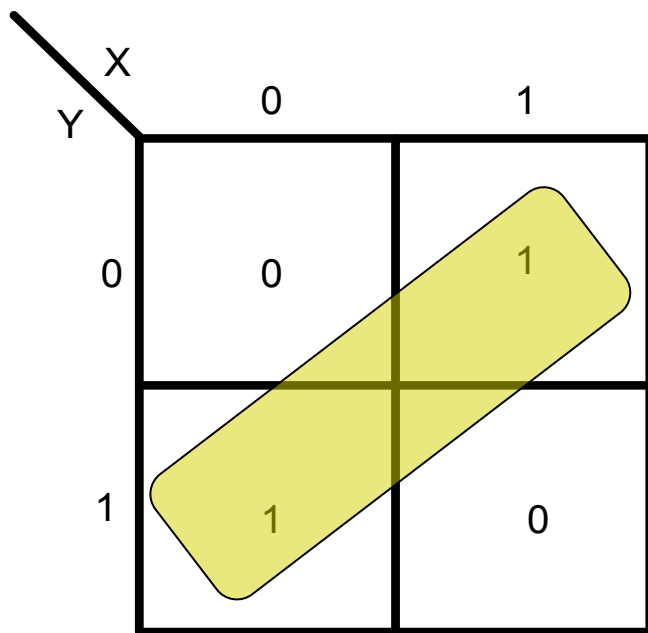
注意：每次都试图用最大的圈，圈最多的1



利用卡诺图简化逻辑函数

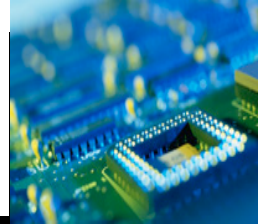


- $F = \sum (1, 2) = XY' + X'Y$





卡诺图化简：相关术语



- 如何化简逻辑函数——最小化(Minimize)
 - (主蕴涵项)定理：最简“积之和”是主蕴涵项之和
- 蕴涵项(Implicant)
 - 任何积项都称为蕴涵项，与卡诺图中的圈对应
- 主蕴涵项(Prime implicant)
 - 也称“本原蕴涵项”或“素项”
 - 定义若逻辑函数的积项 P 再也不能同其它积项合并以组成变量个数更少的积项，则称 P 为主蕴涵项
 - 对应卡诺图中最大的圈



相关术语



- **质主蕴涵项(Essential prime implicant)**
 - 定义：不能被其它蕴涵项代替的主蕴涵项；至少包含一个不能被其它任何主蕴涵项所覆盖的最小项
 - 也称“必要素项”，对应卡诺图中必不可少的最大圈
- **覆盖(Cover)**
 - 若逻辑函数的所有最小项被1组蕴涵项所包含，则该组蕴涵项称为函数的1个覆盖
- **最小覆盖(Minimal cover)**
 - 是1个包含最少主蕴涵项和最少符号数的覆盖



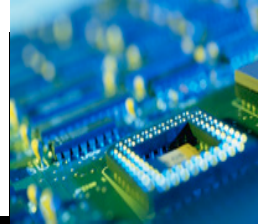
最小化积之和(MSOP)



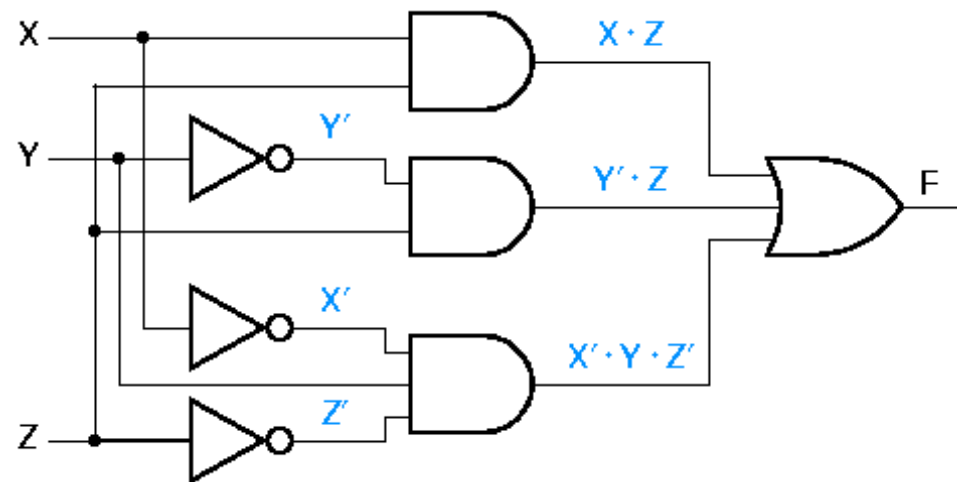
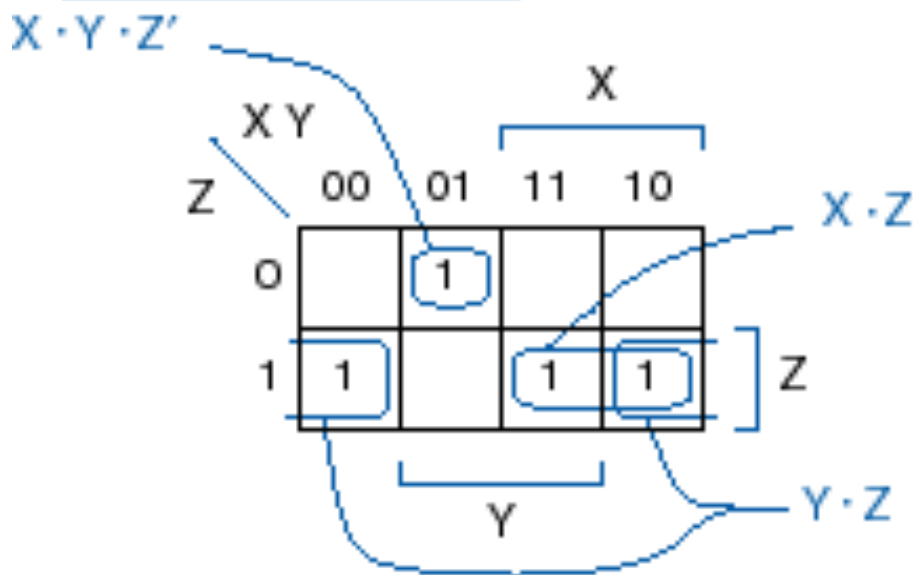
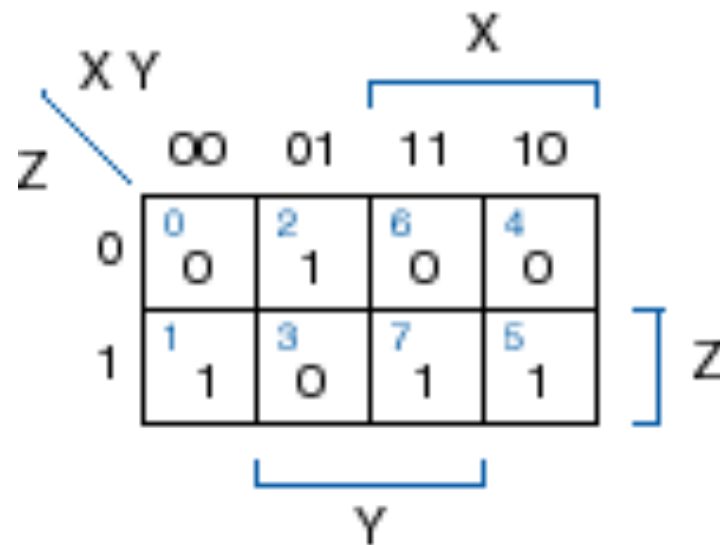
- Minimum sum of products
- 算法A(卡诺图)
 1. 计算其中每个最小项的相邻单元数
 2. 从未被覆盖的具有**最小**相邻数的最小项(从最孤立的1)开始；若存在多种选择，任选其一
 3. 生成这个最小项的1个主蕴涵项并将它放入覆盖中；若该最小项被其它多个主蕴涵项覆盖，选择1个覆盖最多的主蕴涵项
 4. 回到第2步，直到所有最小项被覆盖



Example: $F = \Sigma(1,2,5,7)$

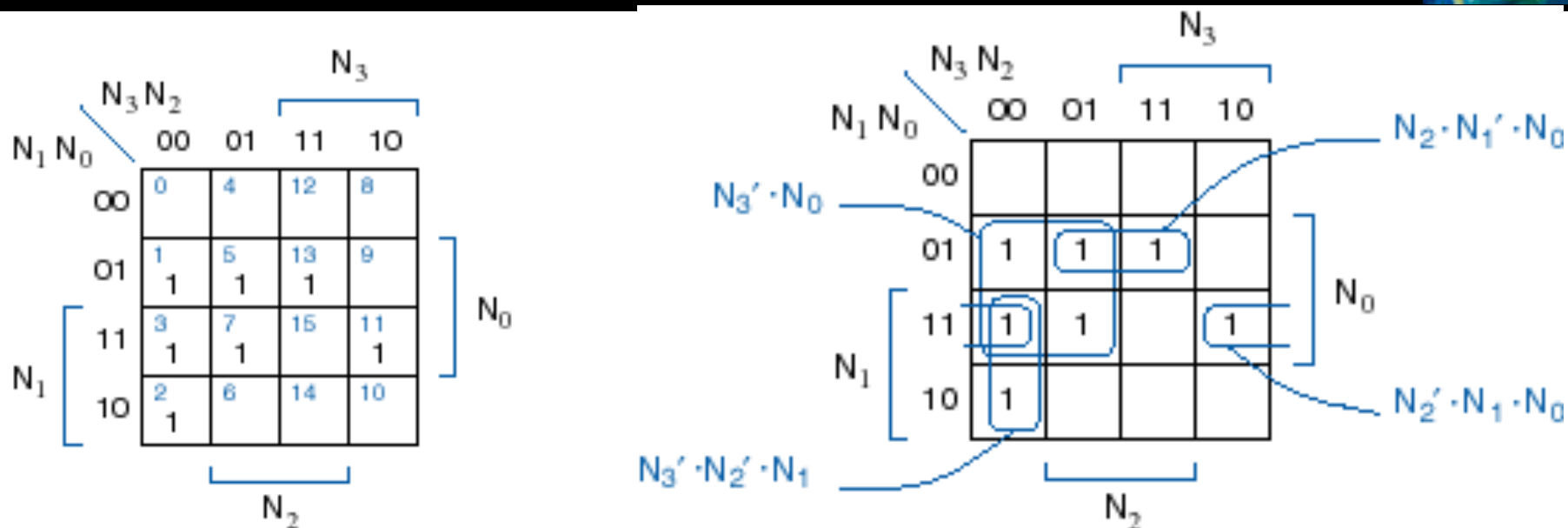
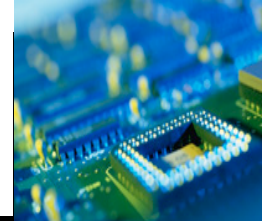


X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



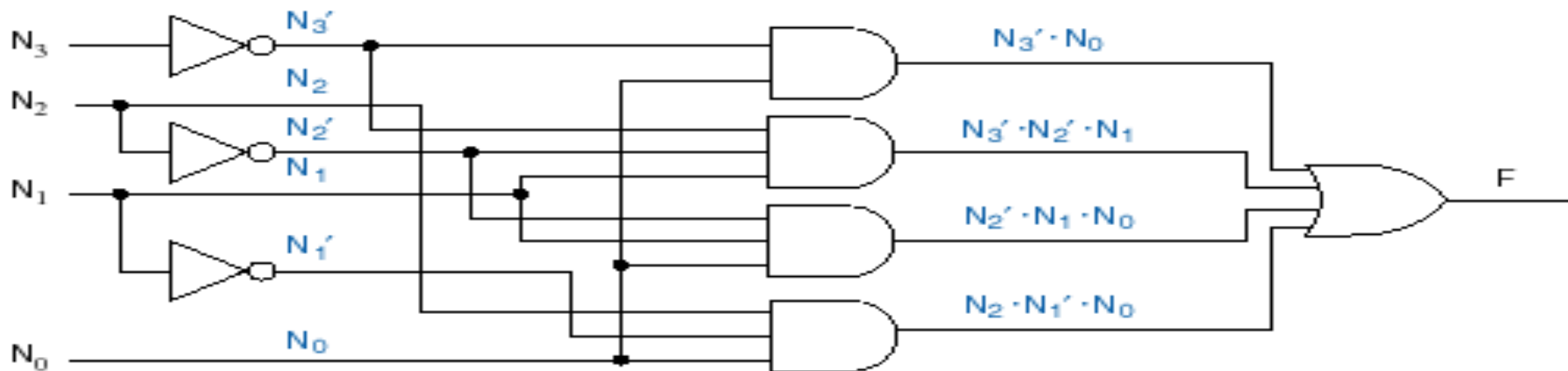


素数检测器



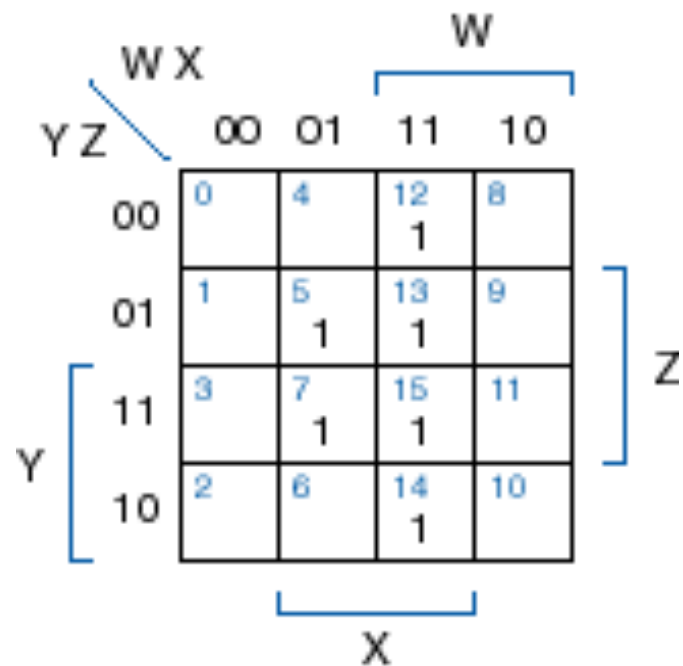
$$F = \sum_{N_3, N_2, N_1, N_0} (1, 2, 3, 5, 7, 11, 13)$$

$$F = N_3' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 + N_2' \cdot N_1 \cdot N_0 + N_2 \cdot N_1' \cdot N_0$$

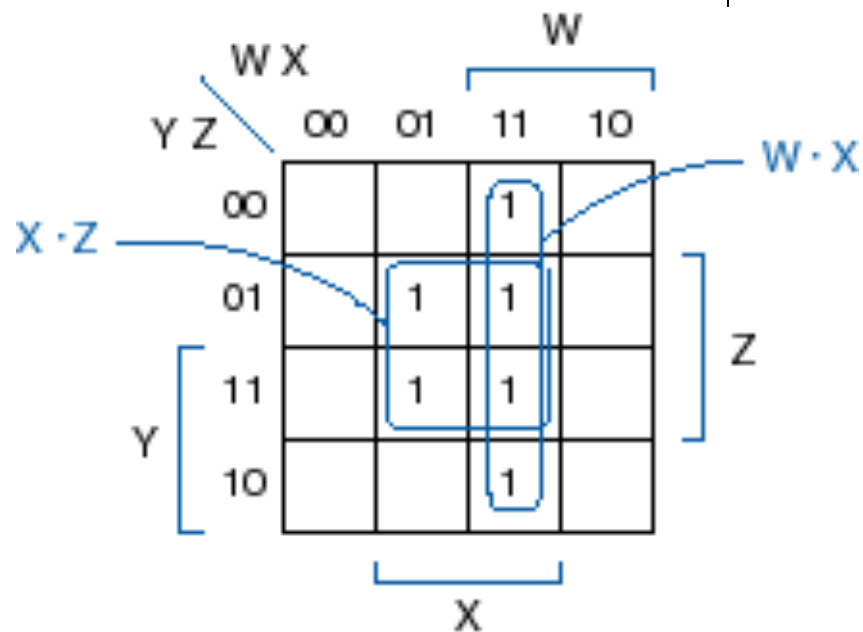




主蕴涵项



$$F = \Sigma_{W,X,Y,Z}(5,7,12,13,14,15)$$



$$F = X \cdot Z + W \cdot X$$



选择质主蕴涵项



W X		W			
		00	01	11	10
Y Z	00	0	4 1	12 1	8
	01	1 1	5 1	13 1	9 1
	11	3 1	7	15 1	11 1
	10	2	6	14 1	10

Y

X

Z

$$F = \sum_{W,X,Y,Z}(1,3,4,5,9,11,12,13,14,15)$$

W X		W			
		00	01	11	10
Y Z	00		1	1	
	01	1	1	1	1
	11	1		1	1
	10			1	

Y

X

Z

$X \cdot Y'$

$W \cdot Z$

$Y' \cdot Z$

$X' \cdot Z$

$W \cdot X$

- 完全和：逻辑函数的所有主蕴涵项之和；

- 完全和不总是最小；

$$F = X \cdot Y' + W \cdot Z + X' \cdot Z + W \cdot X + Y' \cdot Z$$

- 如何选择：选择质主蕴涵项

$$F = X \cdot Y' + X' \cdot Z + W \cdot X$$



多重覆盖的选择



W X		W			
		00	01	11	10
Y Z	00	0 1	4 1	12 	8
	01	1 1	5 1	13 	9
	11	3 1	7 1	15 1	11
	10	2 1	6 	14 1	10

W X		W			
		00	01	11	10
Y Z	00	1	1		
	01	1	1		
	11	1	1	1	1
	10	1		1	1

W X		W			
		00	01	11	10
Y Z	00				
	01				
	11		1		
	10				

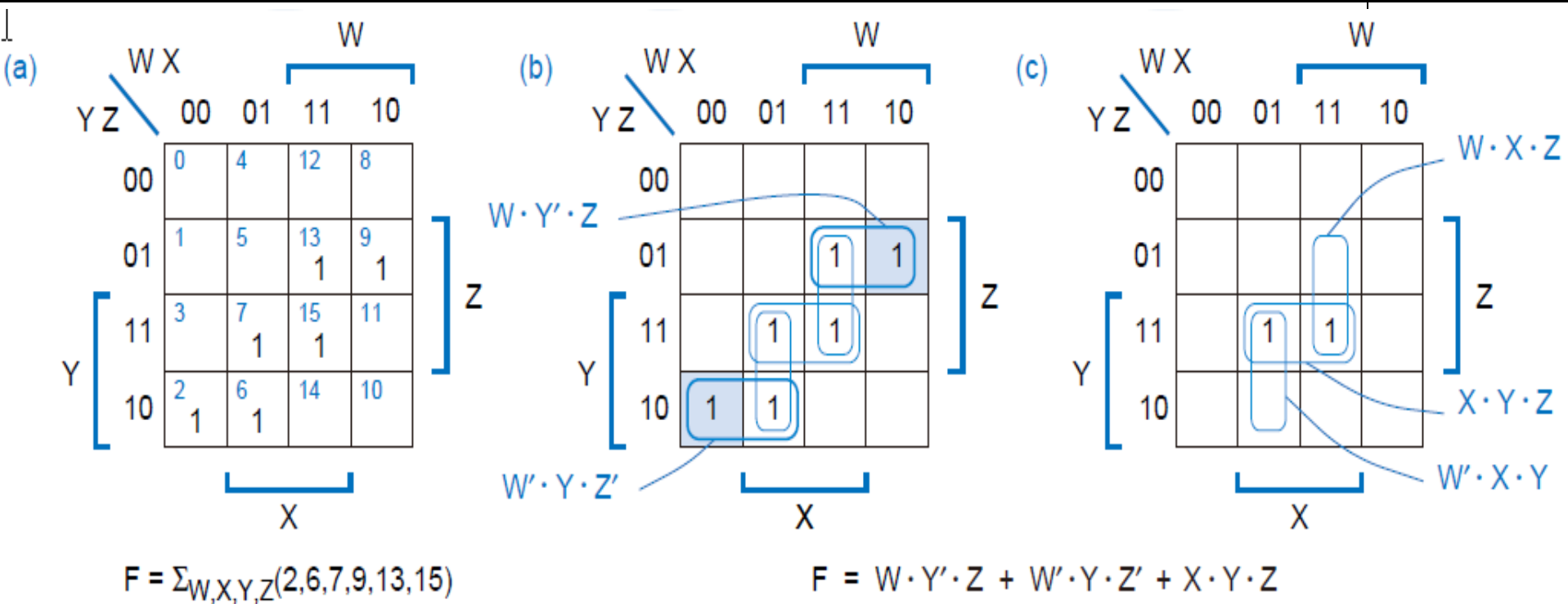
$$F = \Sigma_{W,X,Y,Z}(0,1,2,3,4,5,7,14,15)$$

$$F = W' \cdot Y' + W \cdot X' + W \cdot X \cdot Y + W' \cdot Z$$

- 并非所有的1都被质主蕴涵项所覆盖
- 如何选择：选择输入端数较少（覆盖面大）



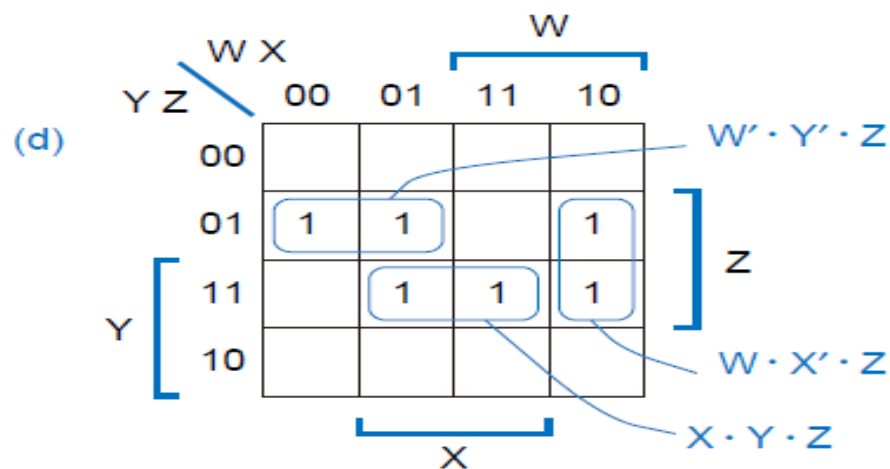
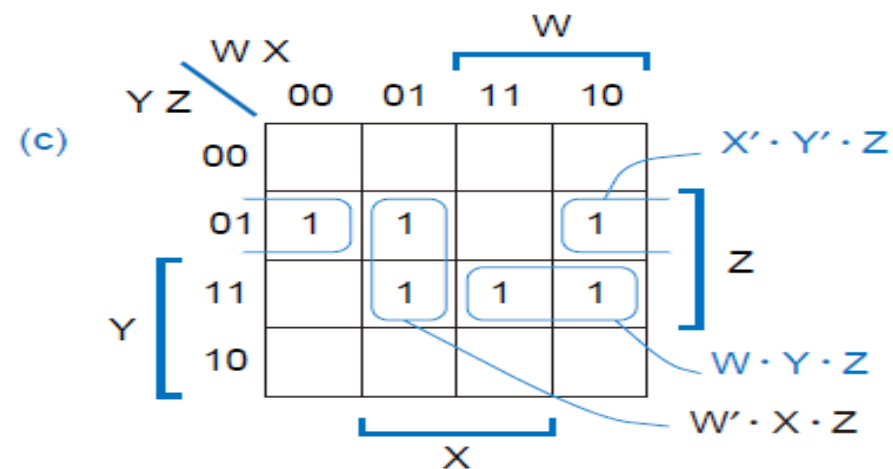
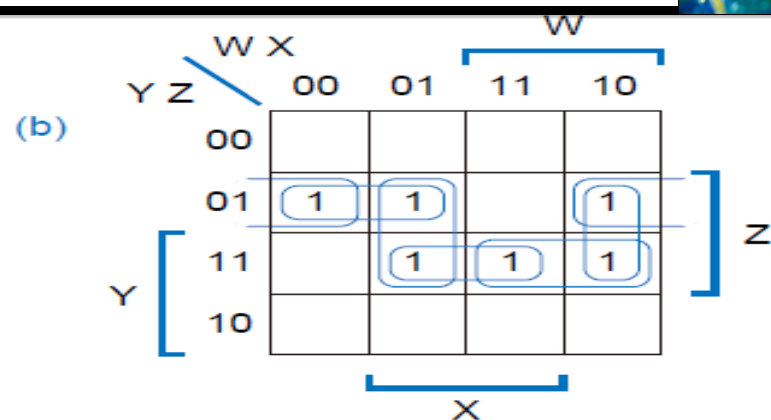
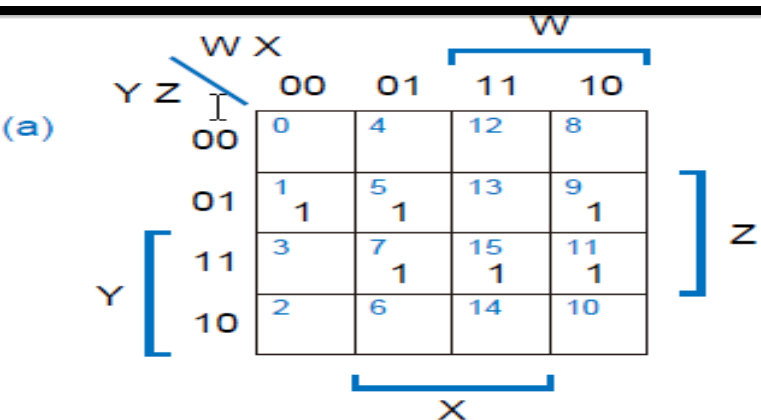
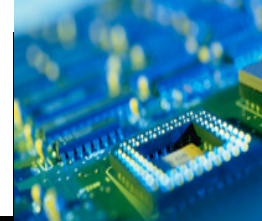
多重覆盖的选择



- 已经被覆盖的主蕴涵项可不考虑，如 $X \cdot Y \cdot Z$



多重覆盖的选择



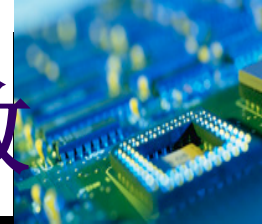
$$F = W' \cdot X \cdot Z + W \cdot Y \cdot Z + X' \cdot Y' \cdot Z$$

$$F = X \cdot Y \cdot Z + W \cdot X' + W' \cdot Y' \cdot Z$$

- 可构建两个不同的最小和

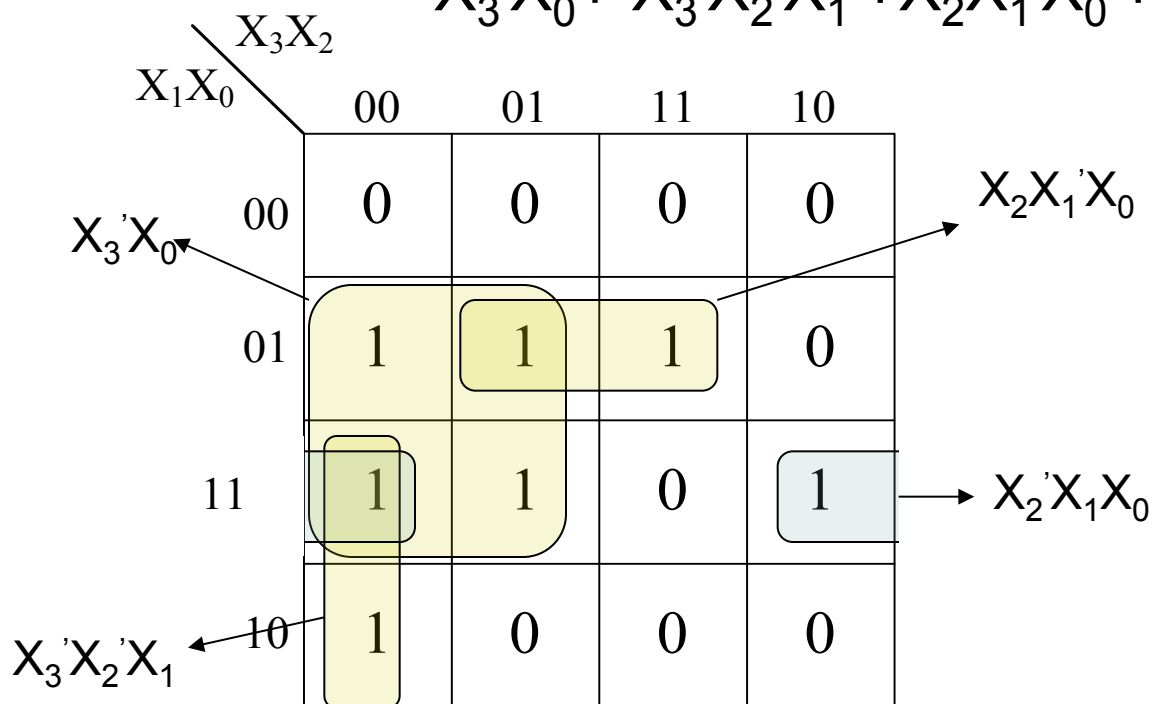


利用卡诺图化简四变量逻辑函数



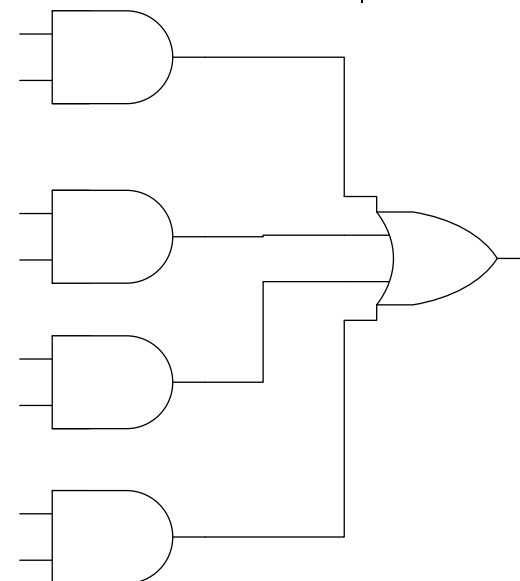
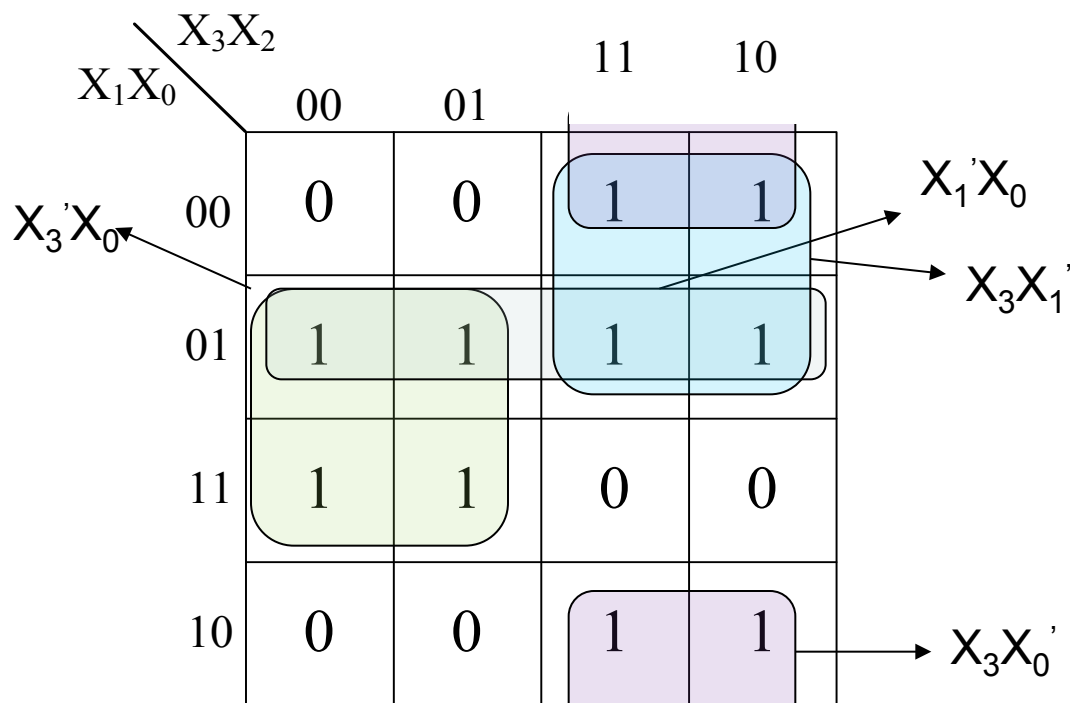
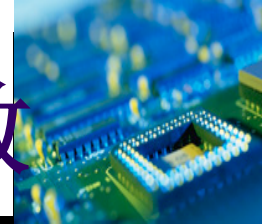
$$F(X_3X_2X_1X_0) = \sum (1, 2, 3, 5, 7, 11, 13)$$

$$= X_3'X_0 + X_3'X_2'X_1 + X_2X_1'X_0 + X_2'X_1X_0$$



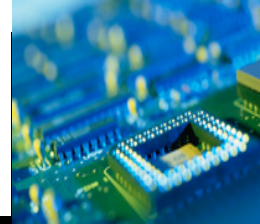


利用卡诺图化简四变量逻辑函数



$$F(X_3X_2X_1X_0) = \Sigma (1, 3, 5, 7, 8, 9, 10, 12, 13, 14)$$

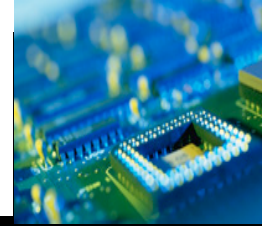
$$??? = X_3'X_0 + X_3X_1' + X_1'X_0 + X_3X_0'$$



- 按照逻辑函数的最小项表达式或者真值表画出卡诺图
- 在图中圈出最大数量的连续1单元
 - 1单元个数必须满足 2^i
 - 可以跨越边界
- 每个圈对应一个乘积项
 - 变量是1，使用原变量；否则使用反变量
 - 变量既有1又有0，则不包含该变量
- 化简的目的是最小化门的数目和门输入的数目



用卡诺图化简为和之积的形式



- 圈卡诺图上的0单元
- 每个圈是一个求和项
- 圈所覆盖的区域：
 - 如果对应着0，取原变量，
 - 如果对应着1，取反变量
 - 如果既有0又有1，则不包含



用卡诺图化简为和之积的形式



- $F = (W+Y) \cdot (W+X+Z') \cdot (W'+Z) \cdot (W'+X+Y')$

- $F = (W+Y) \cdot (W'+Z) \cdot (X+Y'+Z')$

YZ \ WX		WX			
		00	01	11	10
00	0	0	0	0	0
01	0	0	1	1	1
11	0	1	1	0	0
10	1	1	0	0	0

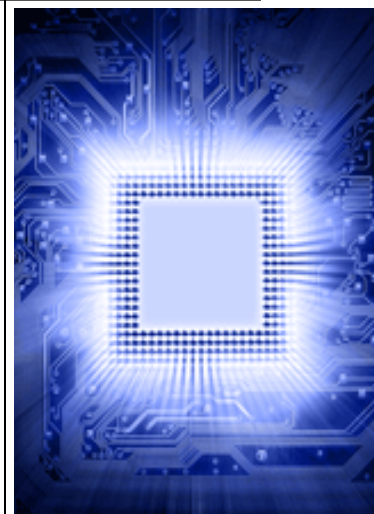


考虑特殊问题的逻辑函数设计



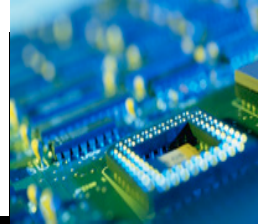
- 逻辑函数变换
 - 与非门实现
 - 或非门实现
 - 与或非门实现
- 其它问题
 - 多输出问题;
 - 电路级数;
 - 输入变量的具有约束关系
- 举例

4.4 定时冒险





冒险或险象



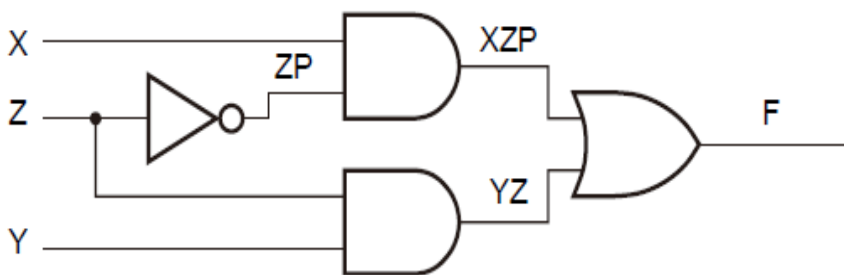
- 逻辑电路的瞬态特性可能和稳态特性的预期不一致
- 由于电路延迟，输出端可能产生短脉冲，可能出现尖峰/毛刺/闪烁(glitch)。
- 输入信号经不同路径因而经历的延迟不同，若各输入信号的变化不能同时传递到输出级，就可能产生真值表以外的冒险信号。
- 一个信号，以两种形式出现在输出端，因传输时间不同，使二者某段时间不具有相应逻辑关系，造成错误输出，称为冒险或险象（ Hazard ）
- 竞争： 门电路的两个输入端同时向相反的逻辑电平跳变。



静态冒险



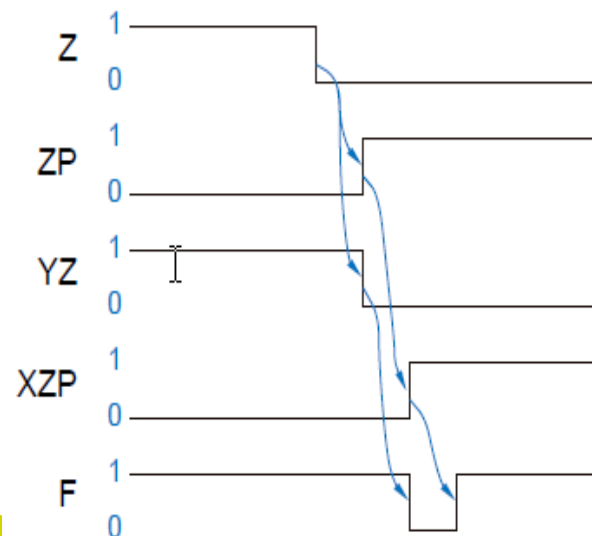
- 静态冒险(*Static hazard*)：一个周期内，输出只出现一次瞬时改变。
- 静态-1冒险(*Static -1 hazard*)：在输出1的过程中，出现0尖峰。
 - 定义：若一个逻辑器件存在两种输入组合，它们a)只有一个变量的值不同；b)这两种输入组合情况下，逻辑器件的输出都为1。在不同输入量发生改变期间，就有可能发生短暂的0输出。



$$F = X \cdot Z' + Y \cdot Z$$

假设每个门电路延迟相同为1个时间单位

通常产生于最小项生成电路中 (POS)

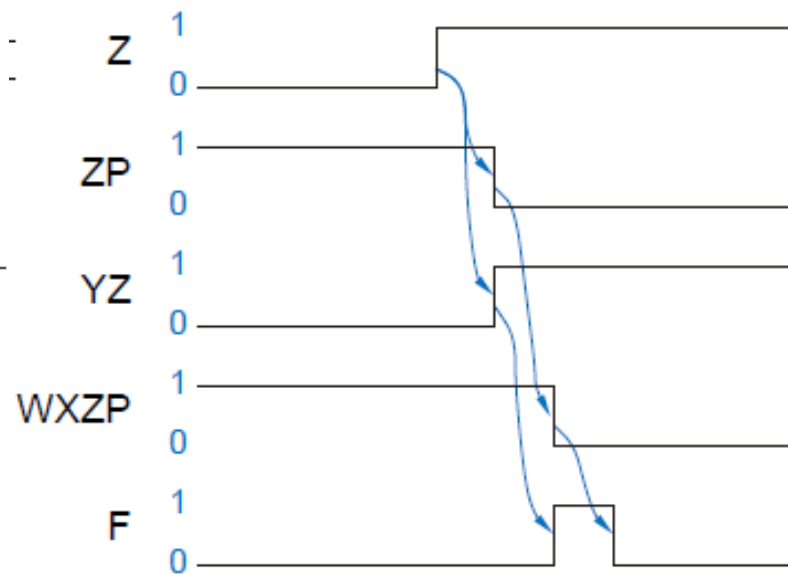
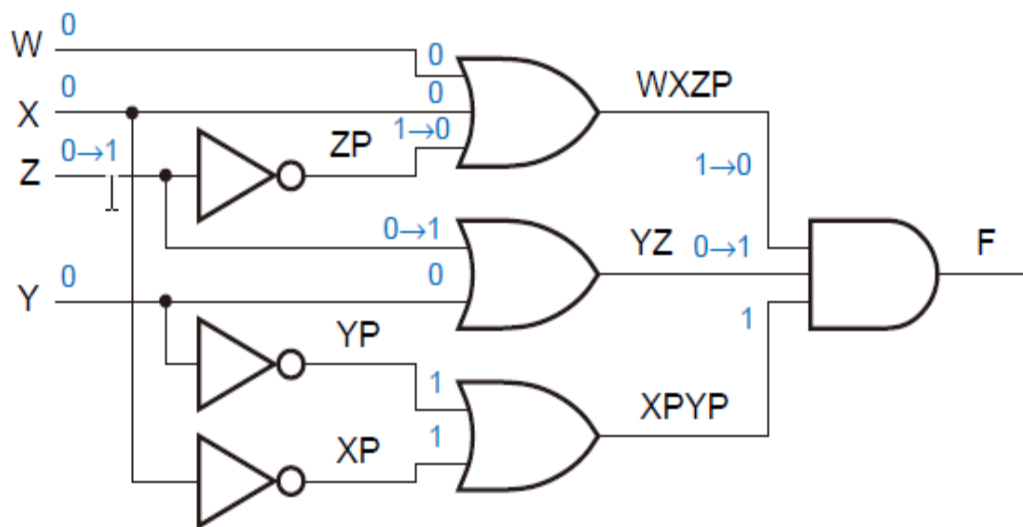




静态冒险



- 静态-0冒险(Static -0 hazard)：在输出0的过程中，出现1尖峰。
- 定义：一个输入组合对，它们a)只有一个变量不同；b)两种输入组合都输出0。在不同输入量发生改变期间，就有可能发生短暂的1输出。



$$F = (W + X + Z') \cdot (Z + Y) \cdot (X' + Y')$$

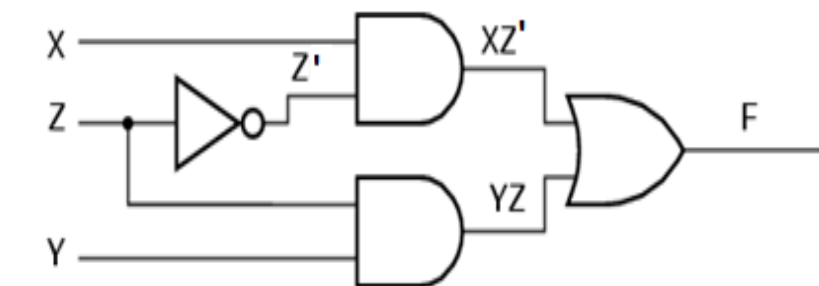
通常产生于最大项生成电路中 (SOP)



静态冒险的检测

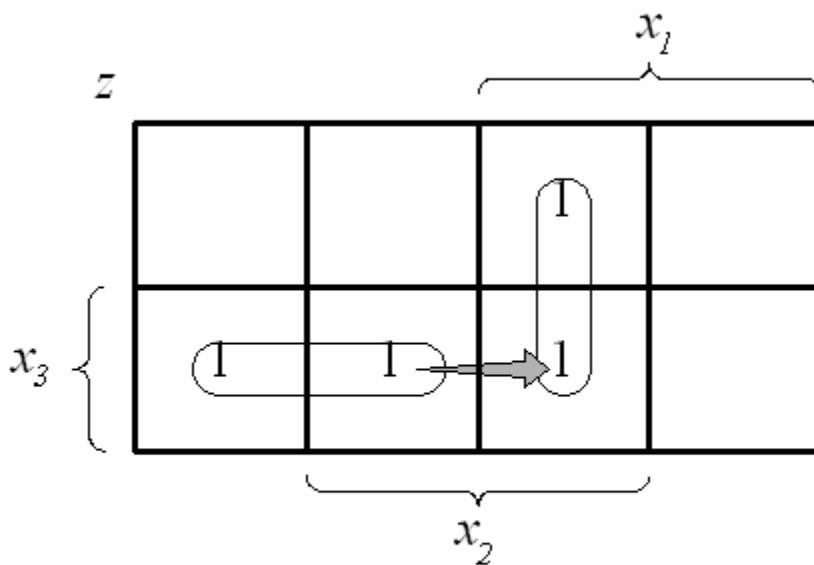


- 卡诺图检测：在卡诺图中存在两个质主蕴涵项相切，当从一个质主蕴涵项向另一个转换时，一旦有传递延迟，则产生险态。



XY \ Z	00	01	11	10	
0	0	0	1	1	XZ'
1	0	1	1	0	YZ

$$F = XZ' + YZ$$

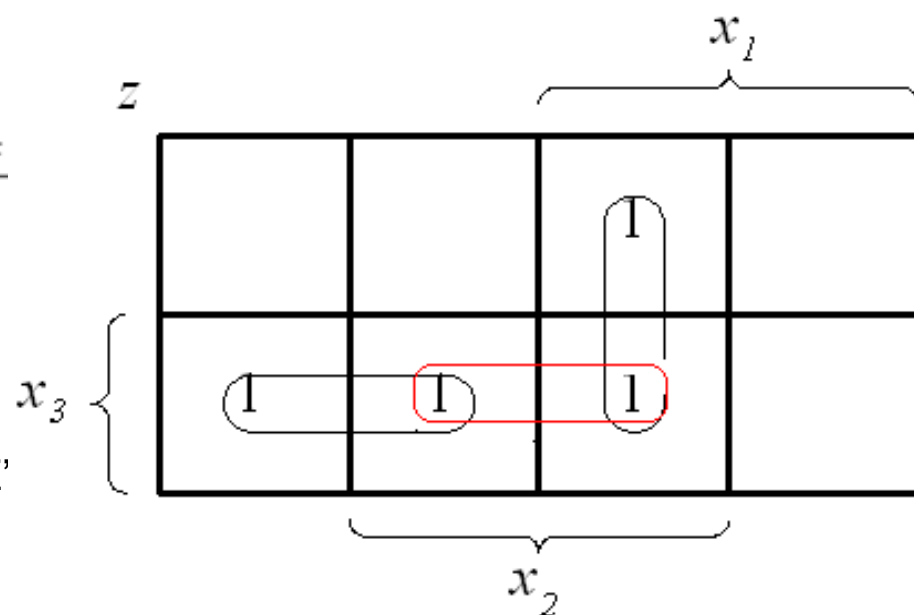
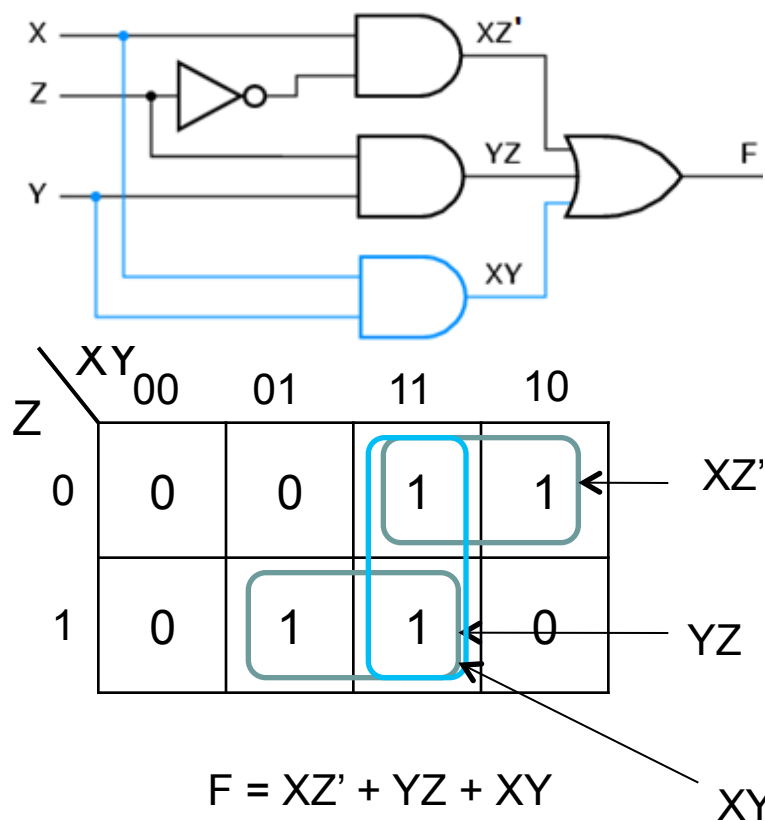




静态冒险的消除

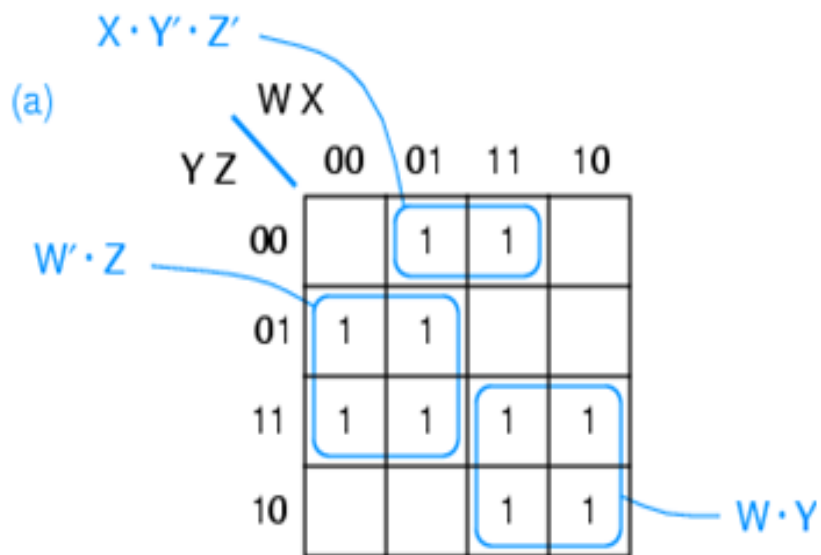


- 添加一致项consensus:增加新的主蕴涵项，覆盖相切的两个质主蕴涵。



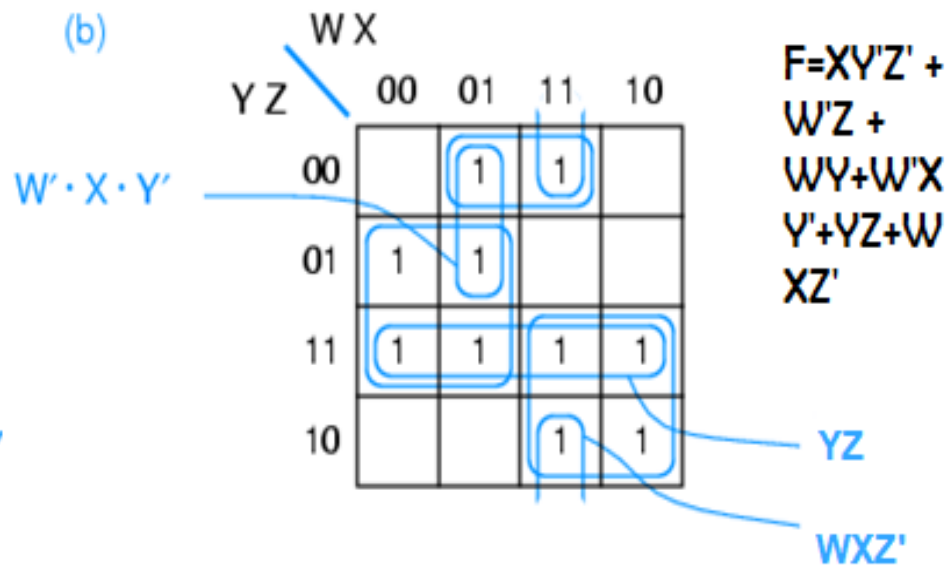


另一个例子



$$F = X \cdot Y' \cdot Z' + W' \cdot Z + W \cdot Y$$

消除冒险之前



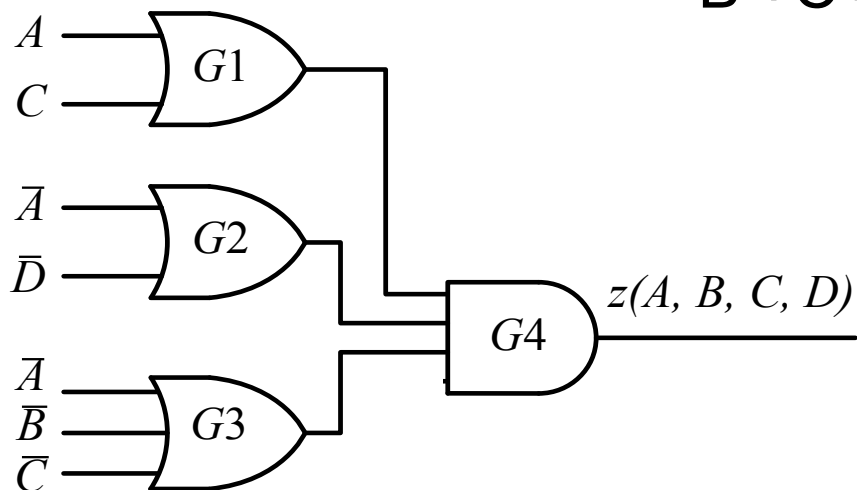
消除冒险之后



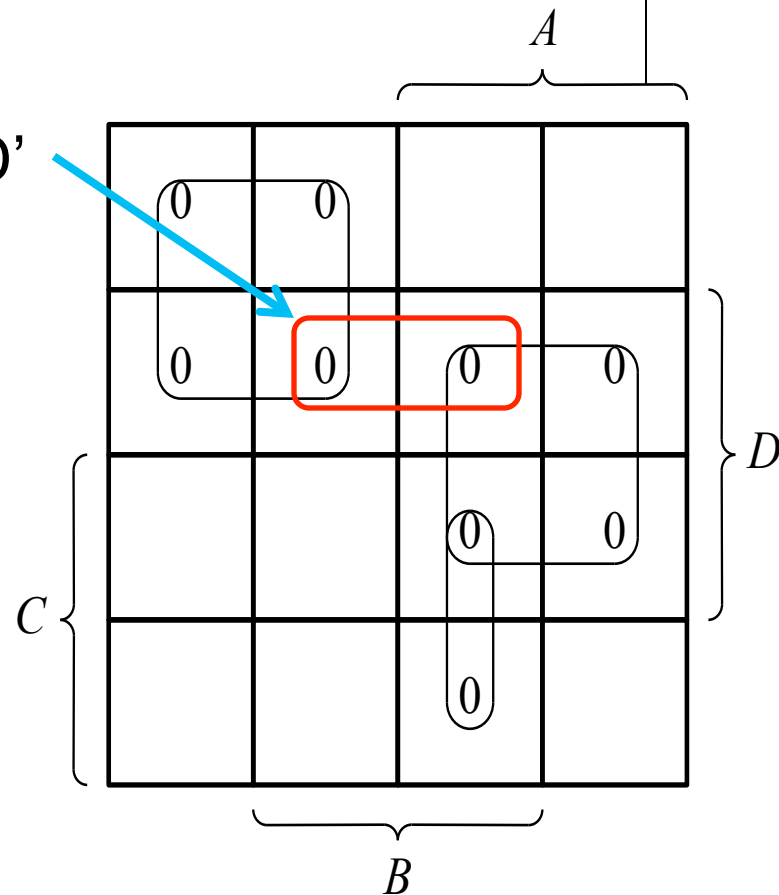
静态冒险的消除



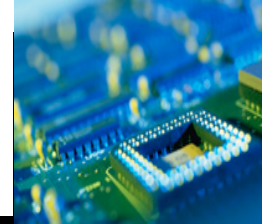
● 静0冒险消除



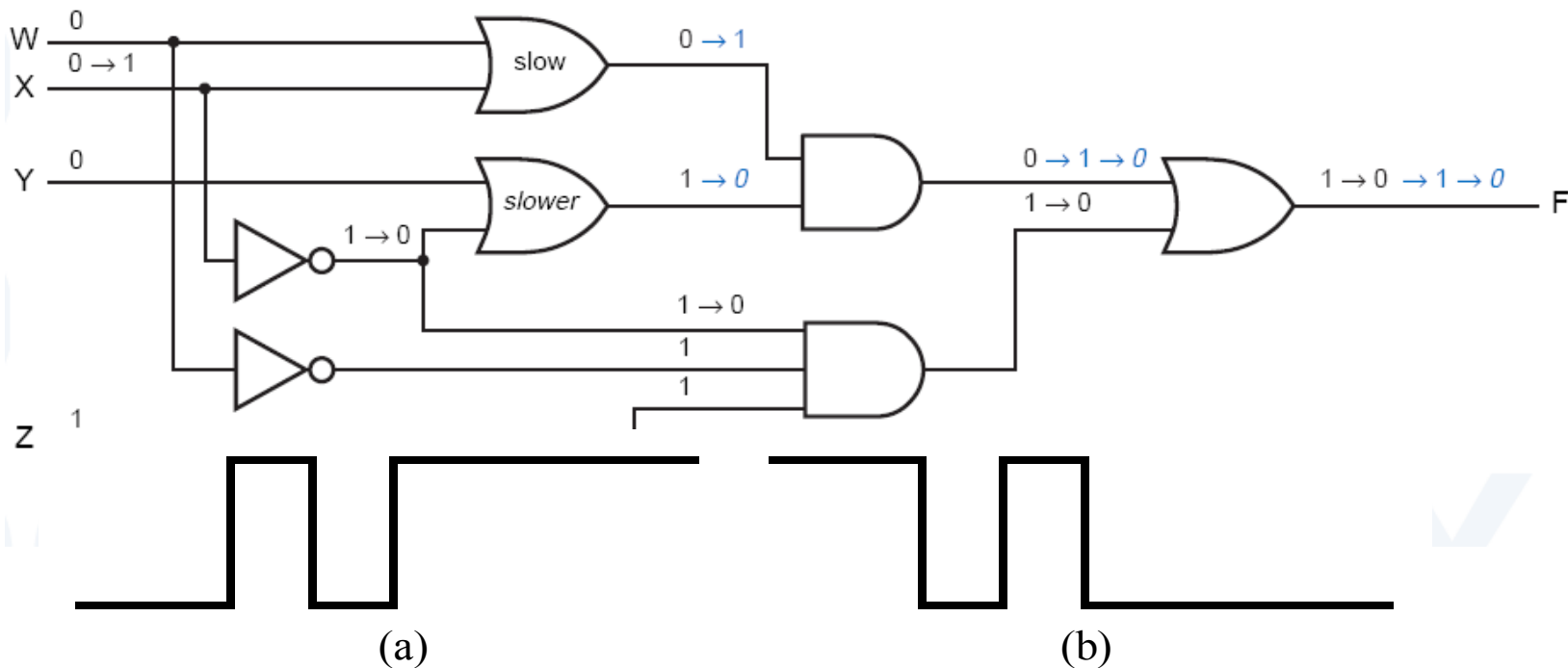
$B' + C + D'$



动态冒险



- 动态冒险 *Dynamic* : 一个输入转变一次而引起输出变化多次。由于多个不同的延迟路径所产生。

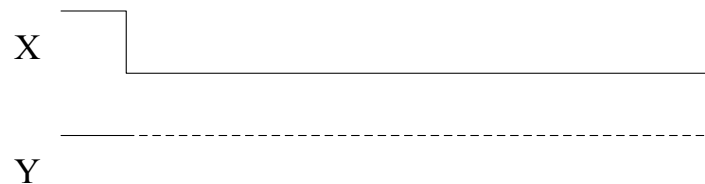
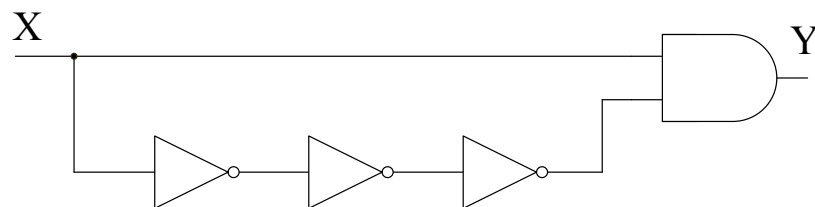
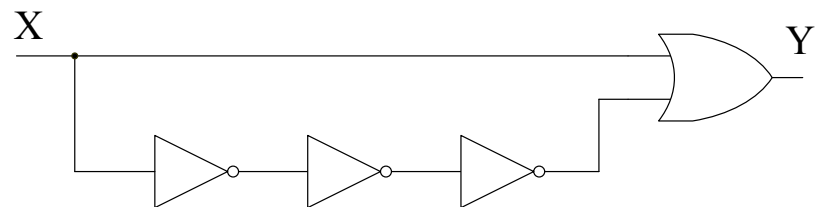


(a) Dynamic hazard on 0 to 1

(b) dynamic hazard on 1 to 0

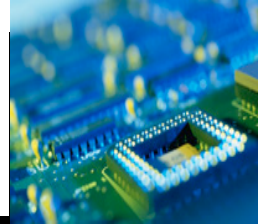


不是所有的竞争都会导致冒险





冒险小结



- 险象的判别

- 代数法：函数式中，当变量同时以原变量和反变量的形式出现在函数式中，则具备存在险象的条件。

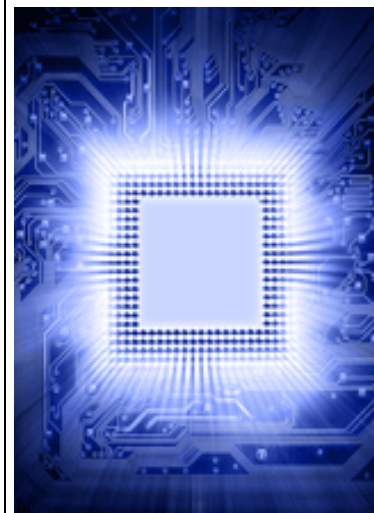
如： $A+A'$ （静1）， $A \cdot A'$ （静0）

- 卡诺图：如两个卡诺圈存在部分的相切，而这个相切的部分没有被另外的卡诺圈所包围，则该电路必然存在险象。

- 险象的消除

- 增加一致项(代数法/卡诺图)：用多余的卡诺圈将相切的部分连接起来。 $F=A \cdot B+A' \cdot C+B \cdot C$ （也有称冗余项）
- 使用选通脉冲/使用RC电路，低通滤波。

无关项处理





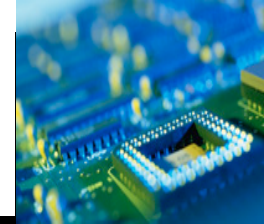
“Don't Care” Input



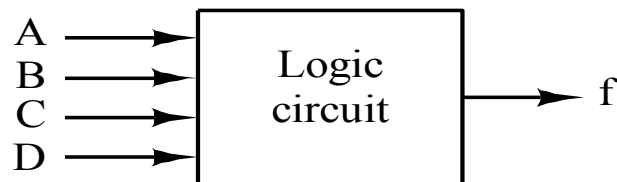
- 无关项/不确定项/禁止态：电路非法状态
 - 干扰导致错码引起
 - 电路启动时，各部件不能同时进入工作状态
- 禁止态出现方式
 - 瞬态方式出现
 - 稳态方式出现
- 克服措施
 - 人工干预或自恢复电路
- 禁止态检测电路通常是必须的，**若不影响电路运行**，此时可不必刻意区分禁止态和其它状态，可以处理成0，也可以处理成1，按照化简的需要酌情确定。



“Don’t Care” Input



- BCD码检测器：BCD digits ≥ 5 from those < 5 .



(a)

ABCD	Minterm	f(A, B, C, D)
0000	0	0
0001	1	0
0010	2	0
0011	3	0
0100	4	0
0101	5	1
0110	6	1
0111	7	1
1000	8	1
1001	9	1
1010	10	d
1011	11	d
1100	12	d
1101	13	d
1110	14	d
1111	15	d

(b)



“Don't Care” Input

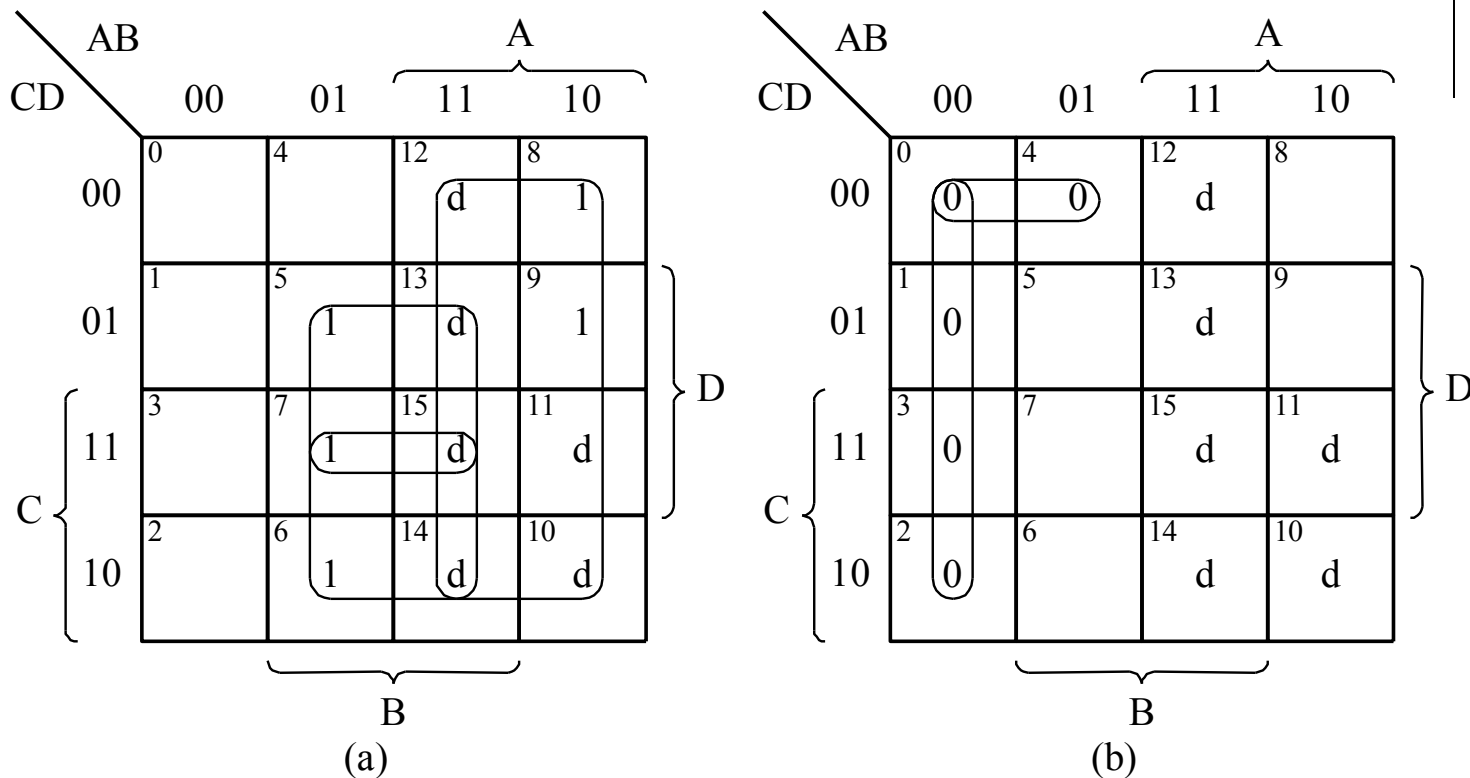


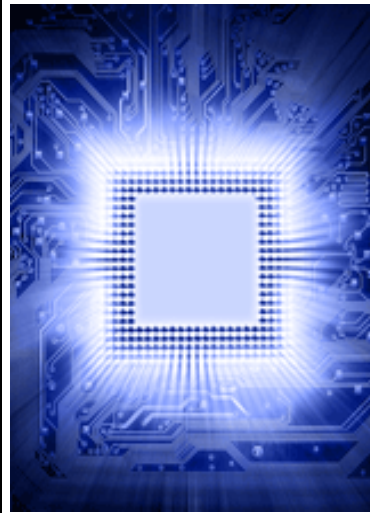
Figure Use of don't cares for SOP and POS forms.

$$f(A,B,C,D) = A + BD + BC;$$

$$f(A,B,C,D) = (A + B)(A + C + D)$$

奎因-穆克鲁斯算法

Quine-McCluskey algorithm





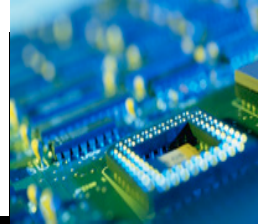
Quine-McCluskey法



- Q-M方法的优点
 - 直接，系统的计算方法
 - 可以处理多于六个变量的函数
 - 可以处理多输出函数
- Overview of the method
 - Given the minterms of a function 得到函数的最小项
 - Find all prime implicants (steps 1 and 2) 查找出所有的质蕴含。
 - Partition minterms into groups according to the number of 1's 根据最小项中所含1的个数分组。只有相邻的两个组，才可能消去一个变量。
 - Exhaustively search for prime implicants。 尽可能查找出所有的质蕴含。
 - Find a minimum prime implicant cover (steps 3 and 4) 查找最小的质蕴含覆盖。
 - Construct a prime implicant chart 建立质蕴含表
 - Select the minimum number of prime implicants 选择最小的质蕴含集合



Q—M选择质蕴含



- 条件：函数表示成最小项表达。
- 步骤：
 - 1、按每个最小项中1的个数，重新排列分组，使得每个组中的最小项含有相同的1的数目。这样能够消去变量的最小项组合，只能来自相邻的两个组。
 - 2、生成一个第n组和n+1组中，只有一个位置不一样的最小项组合的新表。在不相同的位置上用短划线“—”表示，并在原表中的行后面打“√”。重复这一步骤，直到，没有相邻两组可以合并为止。那些不能合并的最小项组，则是质蕴含。
 - 3、建立质蕴含表。在只被一个质蕴含所包含的最小项成在的列上划圆圈。则该质蕴含为必要质蕴含。
 - 4、除去必要质蕴含，再选择最小数目的质蕴含，以覆盖整个函数。



Quine-McCluskey法



Example -- Use the Q-M method to find the MSOP of the function
 $f(A,B,C,D) = \sum m(2,4,6,8,9,10,12,13,15)$

Group	Minterm	ABCD	
1	2	0010	✓
	4	0100	✓
	8	1000	✓
2	6	0110	✓
	9	1001	✓
	10	1010	✓
	12	1100	✓
3	13	1101	✓
4	15	1111	✓

	Minterms	ABCD	
1	2,6	0-10	PI2
	2,10	-010	PI3
	4,6	01-0	PI4
	4,12	-100	PI5
	8,9	100-	✓
	8,10	10-0	PI6
	8,12	1-00	✓
2	9,13	1-01	✓
	12,13	110-	✓
3	13,15	11-1	PI7

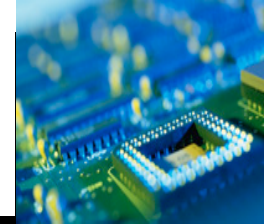
	Minterms	ABCD	
1	8,9,12,13 8,12,9,13	1-0- 1-0-	PI1

Step1 分组

Step2 查找质蕴含



Quine-McCluskey法



				✓	✓		✓	✓	✓
质蕴含	2	4	6	8	9	10	12	13	15
PI1(8,9,12,13)				×	⊗		×	×	
PI2(2,6)	×		×						
PI3(2,10)	×					×			
PI4(4,6)		×	×						
PI5(4,12)		×					×		
PI6(8,10)				×		×			
PI7(13,15)								×	⊗

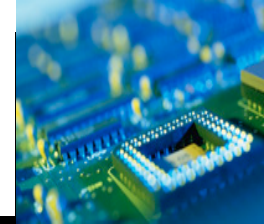
只被一个质蕴含包含的最小项是m9,m15

对应的质蕴含（PI1, PI7）为必要质蕴含。所包含的最小项为m8,m9,m12,m13,m15。

Step 3 -- Prime Implicants Chart



Quine-McCluskey法



选择覆盖所有最小项的最少的质蕴含组合(PI3,PI4)。

	✓	✓	✓	✓
	2	4	6	10
PI2(2,6)	×		×	
PI3(2,10)	×			×
PI4(4,6)		×	×	
PI5(4,12)		×		
PI6(8,10)				×

Step 4 -- Reduced Prime Implicant Chart



Quine-McCluskey法



The Resulting Minimal Realization of f

$$f(A, B, C, D) = \text{PI}_1 + \text{PI}_3 + \text{PI}_4 + \text{PI}_7$$

$$= 1-0- + -010 + 01-0 + 11-1$$

$$= A\boxed{?}C + \boxed{?}B C\boxed{?}D + \boxed{?}A B\boxed{?}D +$$

$$ABD$$



Quine-McCluskey法



How the Q-M Results Look on a K-map

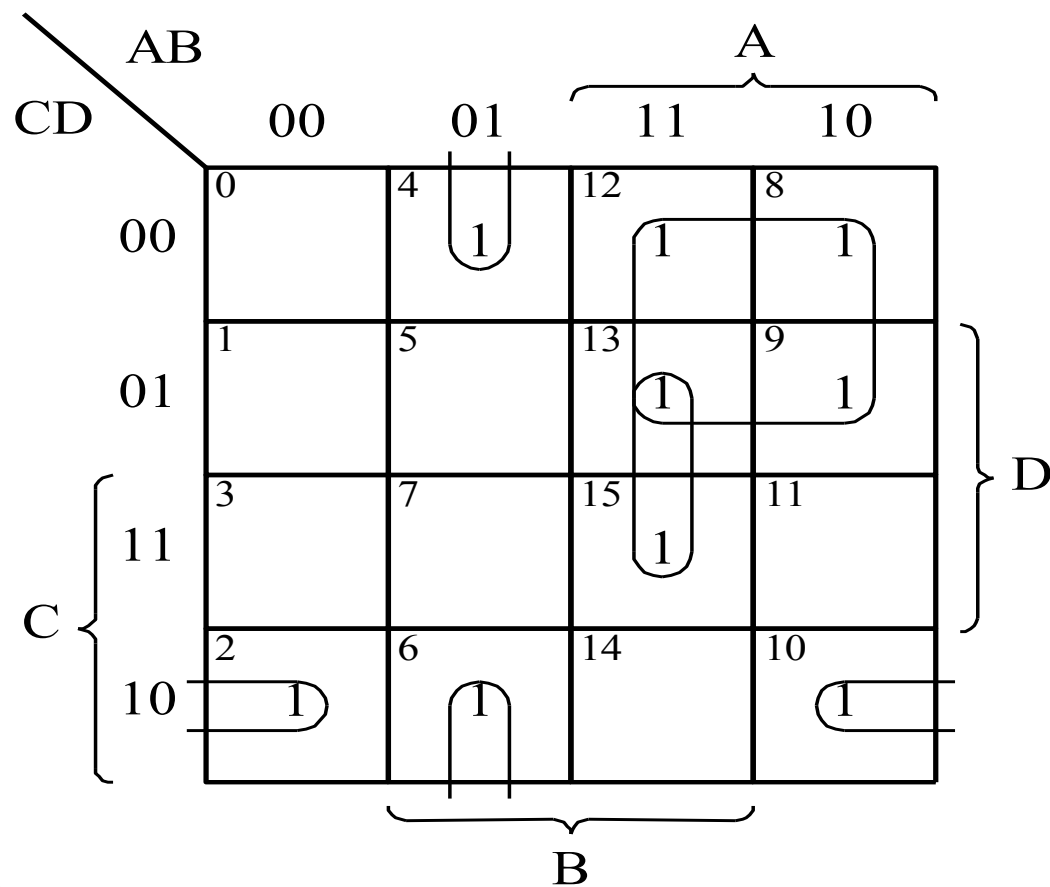
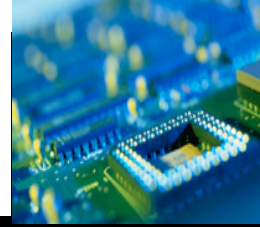


Figure Grouping of terms.



- p160页，
 - 第5、6、
 - 7 (a,c,e,g,i) 、 14 (b,d,f) 、
 - 19 (a,b,c,e) 、
 - 28、35、54 (或-异或)
 - 58 (a,e) 、 61