



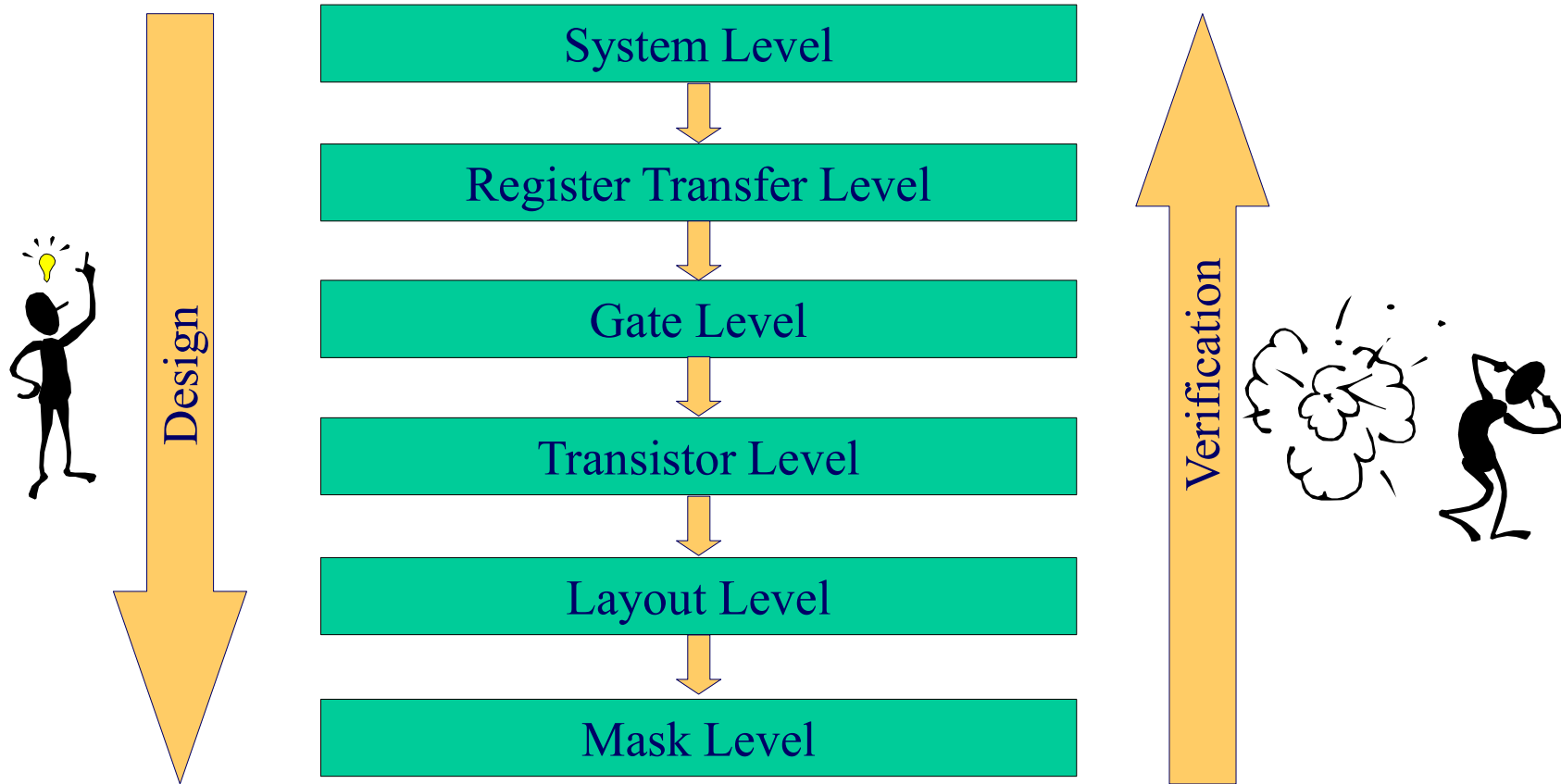
## 第四章 逻辑综合

(Logic Synthesis)

- 4.1 逻辑综合的内容和方法
- 4.2 布尔函数的立方体表示法
- 4.3 立方体运算
- 4.4 多输出函数与单输出函数的阵列变换
- 4.5 单输出函数质立方体的计算
- 4.6 单输出函数的综合
- 4.7 多输出函数的综合
- 4.8 组合逻辑电路的变换
- 4.9 时序逻辑电路的综合



# Design of Integrated Systems





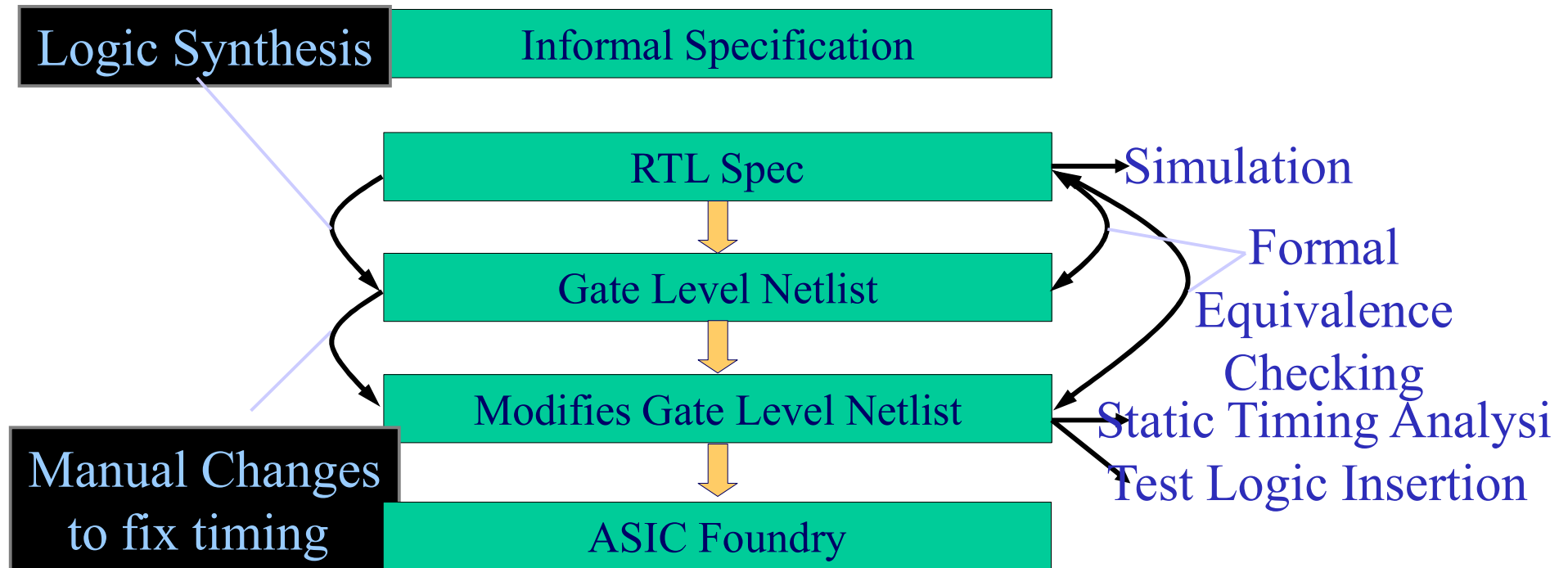
# ASIC Design Flow

- ◆ **Application: general IC market**
  - peripheral chips in PCs, toys, handheld devices etc.
- ◆ **Target: small to medium markets, tight design schedules**
  - e.g. consumer electronics
- ◆ **Complexity of design: standard design style, quite predictable**
  - standard flows, standard off-the-shelf tools
- ◆ **Role of Logic Synthesis:**
  - used on large fraction of design except for special blocks such as RAM's, ROM's, analog components



# ASIC Design Flow

## ◆ Incomplete picture:





## 综合的目标函数

# Objective Function for Synthesis

- ◆ **Minimize area (面积)**
  - in terms of literal count, cell count, register count, etc.
- ◆ **Minimize power(功耗)**
  - in terms of switching activity in individual gates, deactivated circuit blocks, etc.
- ◆ **Maximize performance (性能)**
  - in terms of maximal clock frequency of synchronous systems, throughput for asynchronous systems
- ◆ **Any combination of the above (组合目标)**
  - combined with different weights
  - formulated as a constraint problem
    - “minimize area for a clock speed > 300MHz”
- ◆ **More global objectives (其他总体目标)**
  - feedback from layout
    - actual physical sizes, delays, placement and routing



# 综合约束

## Constraints on Synthesis

- ◆ **Given implementation style: (实现方式)**
  - two-level implementation (PLA, CAMs)
  - multi-level logic
  - FPGAs
- ◆ **Given performance requirements (性能要求)**
  - minimal clock speed requirement
  - minimal latency, throughput
- ◆ **Given cell library (指定单元库)**
  - set of cells in standard cell library
  - fan-out constraints (maximum number of gates connected to another gate)
  - cell generators



## 逻辑综合的历史

- ◆ 1960s: 最先使用布尔推理实现测试马自动生成 (automatic test pattern generation)
  - D-Algorithm
- ◆ 1978: IBM在设计大型计算机过程中引入形式化等价性检验 (Formal Equivalence checking)
  - SAS tool based on the DBA algorithm
- ◆ 1979: IBM 引入门阵列逻辑综合( logic synthesis for gate array )
  - LSS, next generation is BooleDozer
- ◆ End 1986: Synopsys 成立
  - 第1个产品 “remapper”, 基于标准单元库 (standard cell libraries)
  - 后来扩展到RTL综合
- ◆ 1990s other synthesis companies enter the marker
  - Ambit, Compass, Synplicity. Magma, Monterey, ...



# 逻辑综合内容

- ◆ 布尔函数表示和基本算法
  - 表示法: Boolean functions, formulas, circuits, cube representations, BDDs
  - 数据结构及算法
  - 可满足性问题 (SAT)
  
- ◆ 组合电路的功能优化 (Functional optimization of combinational circuits)
  - 2级电路 (two-level circuits)
    - Quine McCluskey
    - Espresso
  - 多级电路 (multi-level circuits)
    - algebraic methods
    - structural transformation-based methods
    - technology mapping





# 逻辑综合内容

- ◆ **时延 (Timing)**
  - timing models and timing analysis
  - timing optimization
- ◆ **时序电路的功能优化 (Functional Optimization of Sequential Circuits)**
  - clock skew optimization
  - retiming
  - synchronous versus asynchronous circuits
  - state assignment and state minimization
  - reachability analysis
- ◆ **低功耗综合 (Low-power Synthesis)**
  - power analysis
  - low-power synthesis



## 4.1 逻辑综合的内容和方法

### ◆ 手工设计:

真值表 → 布尔表达式 → 逻辑电路

### ◆ 综合——自动设计:

- 功能描述 → 结构描述
- 高层次描述 → 低层次描述
- 同层次描述 → 优化(不同的实现方法)

### ◆ 约束条件:

- 造价
- 速度
- 功耗 .....



# 手工设计方法回顾

## 真值表卡诺图和布尔表达式

输 入			输 出	
$x_1$	$x_2$	$x_3$	$y^1$	$y^2$
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	d	d
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

$$y_1 = \overline{x_1} \overline{x_2} + x_1 x_2 x_3$$

$$y_2 = \overline{x_1} x_3 + \overline{x_2} x_3 + x_1 x_2 \overline{x_3}$$

		$x_2 x_3$			
		00	01	11	10
$x_1$	0	1	1	0	0
	1	d	0	1	0

		$x_2 x_3$			
		00	01	11	10
$x_1$	0	0	1	1	0
	1	d	1	0	1



## 手工设计方法回顾

$x_1$	$x_2$	$x_3$	$f^1$	$f^2$	$f^3$
0	0	0	1	1	0
0	0	1	0	1	d
0	1	0	1	1	1
0	1	1	1	0	1
1	0	0	1	0	1
1	0	1	0	1	1
1	1	0	0	0	1
1	1	1	0	0	0

真值表

$x_1, x_2$		00	01	11	10
$x_3$	$f^1$	0	1		1
	1		1		

$x_1, x_2$		00	01	11	10
$x_3$	$f^2$	0	1	1	
	1	1			1

$x_1, x_2$		00	01	11	10
$x_3$	$f^3$	0		1	1
	1	d	1		1

卡诺图

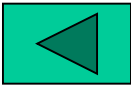




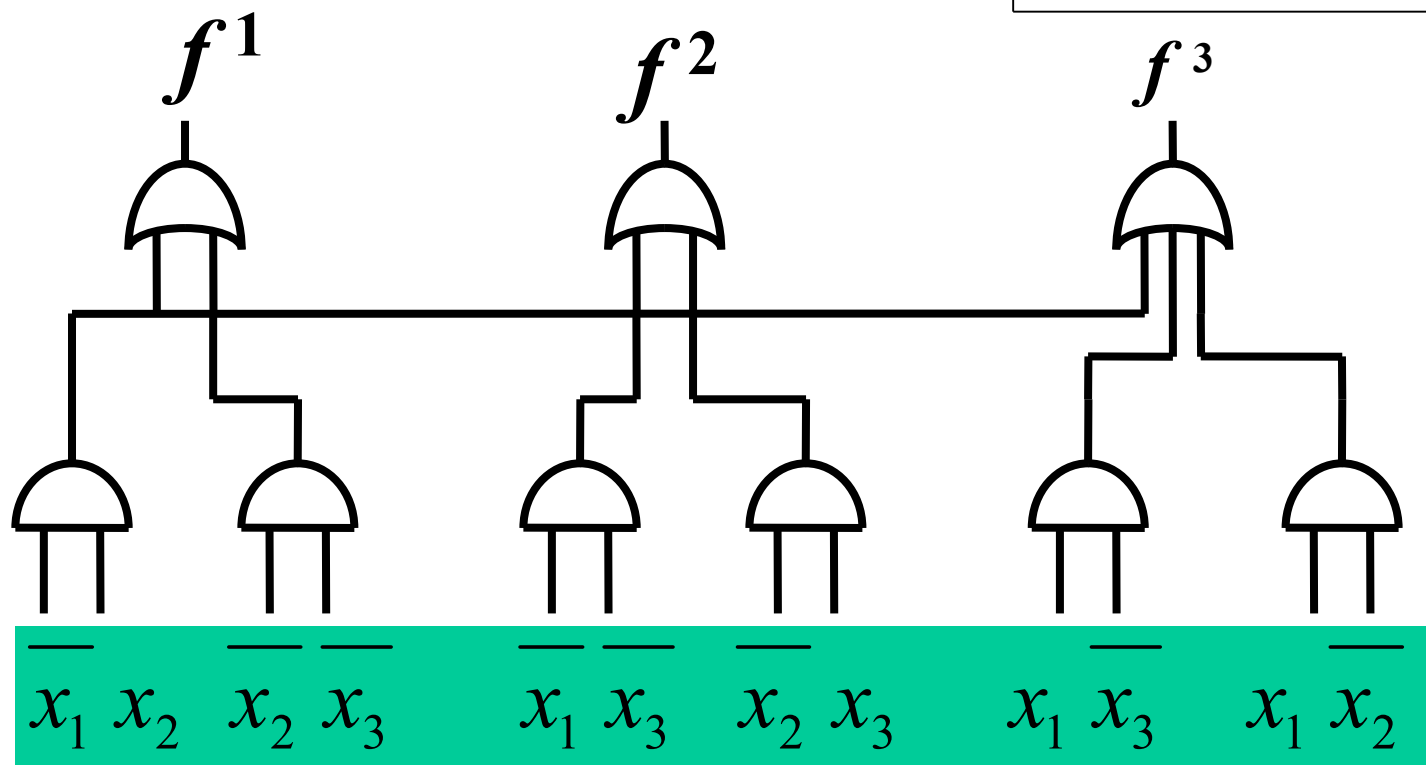
## 手工设计结果

门数=9

输入端数=19

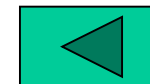


$x_1, x_2$		00	01	11	10
$x_3$	0	1	1		1
	1		1		





## 设计优化

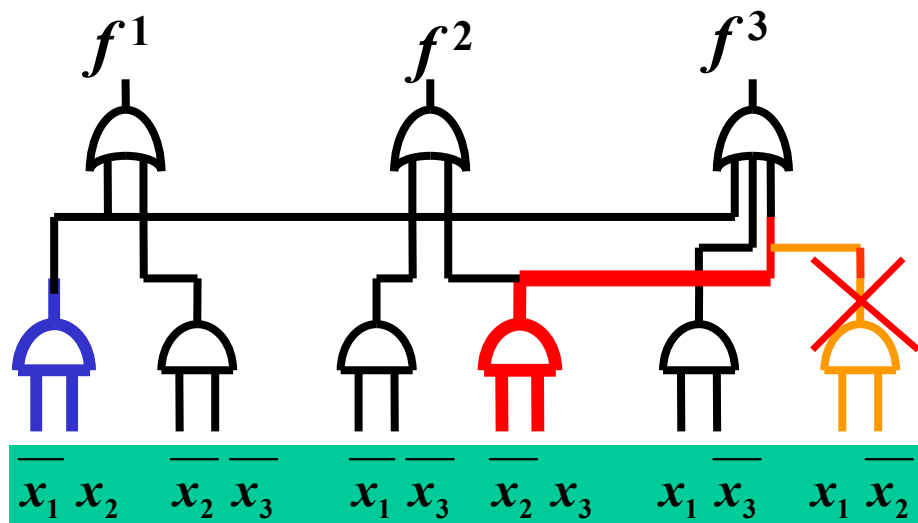
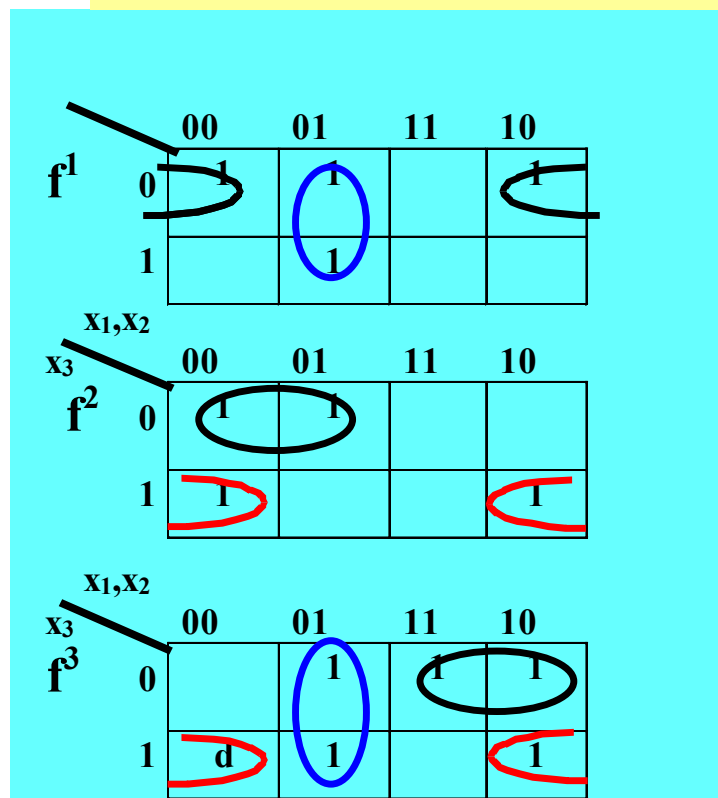


项数  $\Leftrightarrow$  与门个数

因子  $\Leftrightarrow$  与门输入端个数

门数尽量少, 门尽量公用

门的输入端尽量少

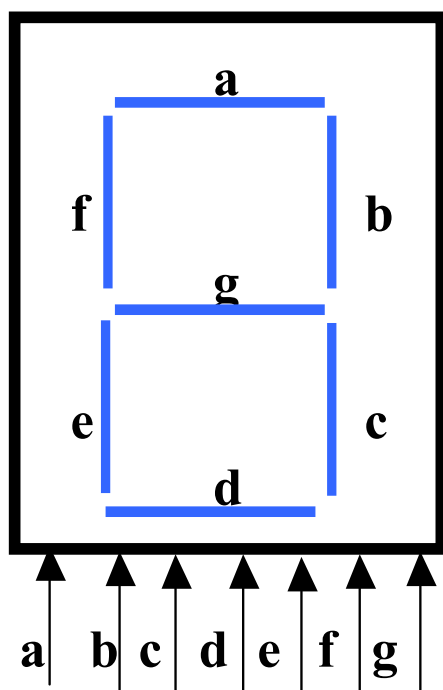


门数 = 8    输入端数 = 17

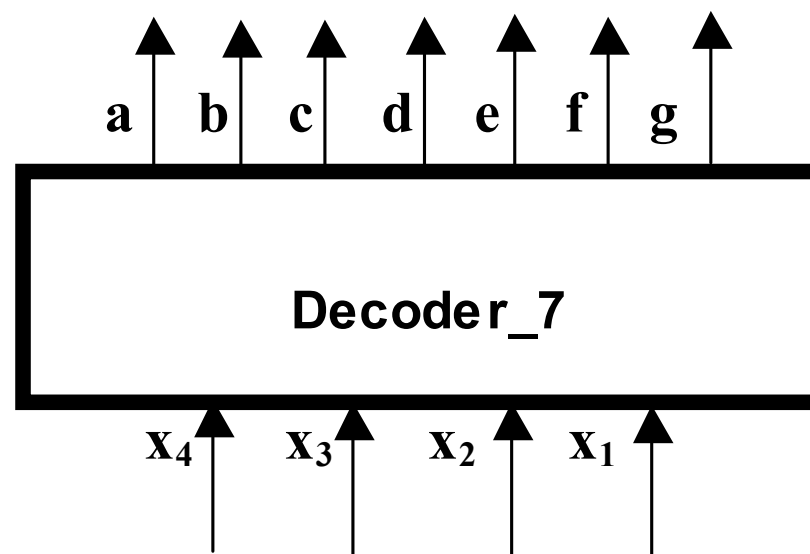


## 实例 -- 7段译码器设计

7 段数码管:



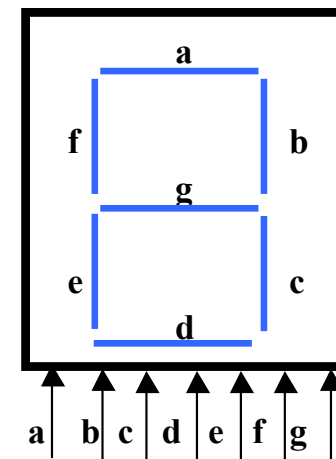
7 段译码器:





## 实例 -- 7段译码器设计(续)

步骤1 列真值表:



7 段译码器真值表

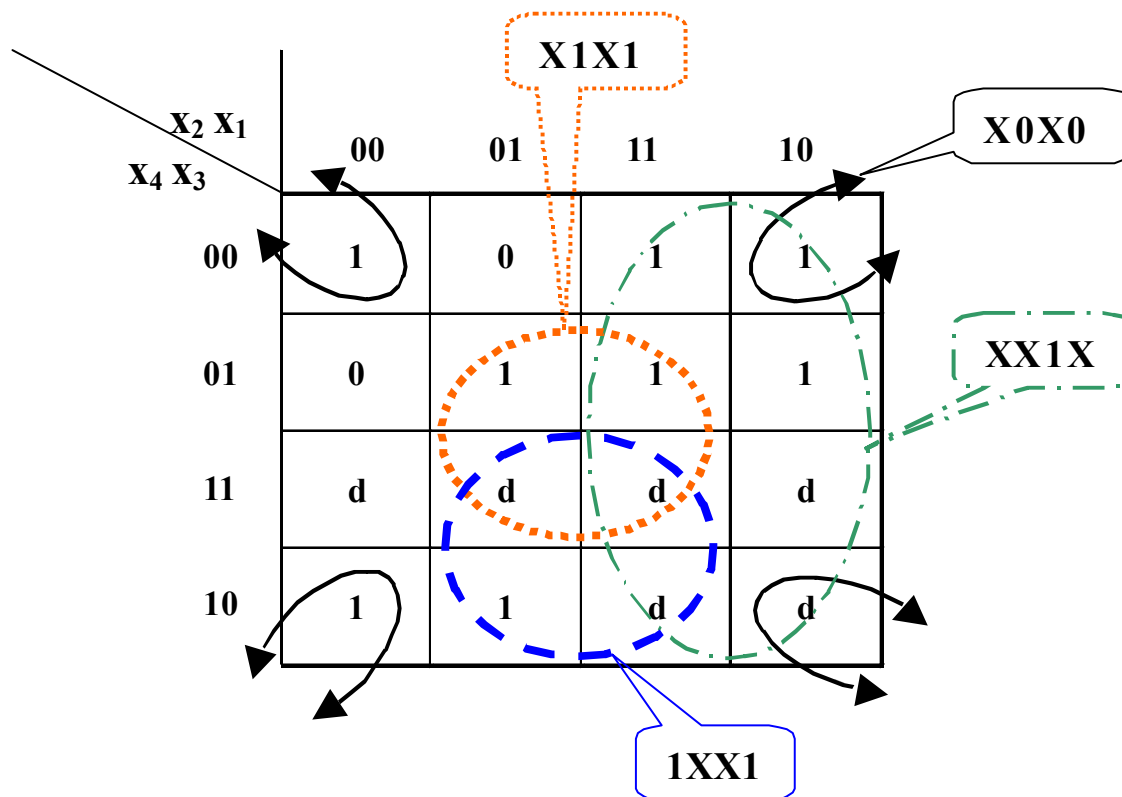
$x_4x_3x_2x_1$	a b c d e f g	$x_4x_3x_2x_1$	a b c d e f g
0 0 0 0	1 1 1 1 1 1 0	1 0 0 0	1 1 1 1 1 1 1
0 0 0 1	0 1 1 0 0 0 0	1 0 0 1	1 1 1 1 0 1 1
0 0 1 0	1 1 0 1 1 0 1	1 0 1 0	d d d d d d d
0 0 1 1	1 1 1 1 0 0 1	1 0 1 1	d d d d d d d
0 1 0 0	0 1 1 0 0 1 1	1 1 0 0	d d d d d d d
0 1 0 1	1 0 1 1 0 1 1	1 1 0 1	d d d d d d d
0 1 1 0	1 0 1 1 1 1 1	1 1 1 0	d d d d d d d
0 1 1 1	1 1 1 0 0 0 0	1 1 1 1	d d d d d d d





## 实例 -- 7段译码器设计(续)

步骤 2: 写出布尔表达式并化简:

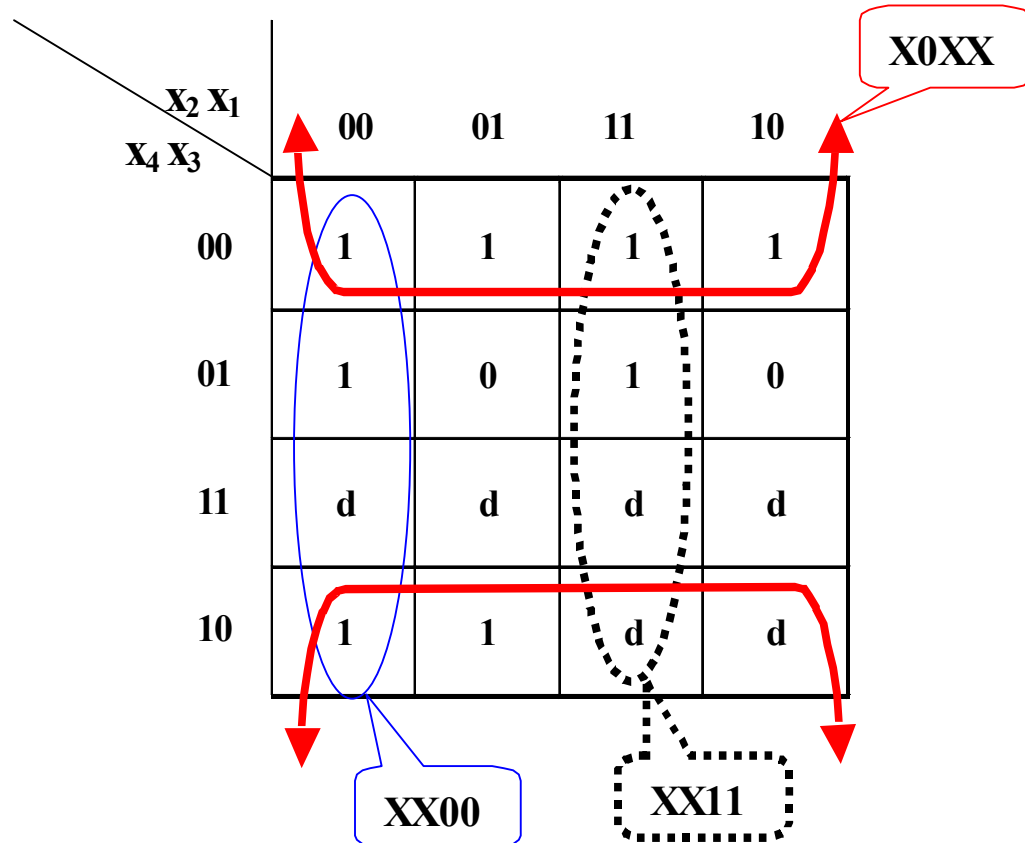


$$a = x_2 + x_3x_1 + x_4x_1 + \bar{x}_3\bar{x}_1$$



## 实例 -- 7段译码器设计(续)

步骤 2: 写出布尔表达式并化简:



$$b = \bar{x}_3 + \bar{x}_2 \bar{x}_1 + x_2 x_1$$



## 实例 -- 7段译码器设计(续)

步骤 2: 写出布尔表达式并化简:

.....

$$g = x_4 + \bar{x}_3 x_2 + x_3 \bar{x}_1 + x_3 \bar{x}_2 x_1$$

步骤 3: 根据化简了的布尔表达式, 画出7段译码器的逻辑图。

思考: 怎样进一步化简? ➡ 乘积项共享



## 实例 -- 7段译码器设计(续)

$X_4 X_3 X_2 X_1$	a b c d e f g
x 0 x x	x 1 x x x x x
x 1 x 0	x x 1 x x 1 1
x 1 0 1	1 x x 1 x 1 1
x x 0 x	x x 1 x x x x
x x 1 1	1 1 1 x x x x
x 0 x 0	1 x x 1 1 x x
x x 1 0	1 x x 1 1 x x
x x 0 0	x 1 x x x 1 x
1 x x x	1 x x 1 x 1 1
x 0 1 x	x x x 1 x x 1

PRODUCT NUMBER=10  
TOTAL COST=44

左边为EDA工具的综合结果:

- **PRODUCT NUMBER = 10**  
表示乘积项个数为10;
- **TOTAL COST = 44**  
表示取值为0或1的文字  
总数为44。
- 乘积项中文字个数  $\geq 2$  时,  
电路中需要一个与门。 →
- 与门个数 =  $10 - 3 = 7$ ;
- 输入端总数 =  $44 - 3 = 41$



## 布尔函数的各种表示方法

- ◆ 真值表；
- ◆ 布尔表达式；
- ◆ 卡诺图；
- ◆ 覆盖表  $\Leftrightarrow$  立方体集合；
- ◆ 二叉判决图 (Binary Decision Diagram, BDD)



## 真值表、卡诺图和布尔表达式

输 入			输 出	
$x_1$	$x_2$	$x_3$	$y^1$	$y^2$
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	d	d
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

$$y_1 = \overline{x_1} \overline{x_2} + x_1 x_2 x_3$$

$$y_2 = \overline{x_1} x_3 + \overline{x_2} x_3 + x_1 x_2 \overline{x_3}$$

		$x_2 x_3$			
		00	01	11	10
$y_1:$	$x_1$	0	1	0	0
	1	d	0	1	0

		$x_2 x_3$			
		00	01	11	10
$y_2:$	$x_1$	0	1	1	0
	1	d	1	0	1



## 立方体集合 (覆盖)

◆ 函数 $y^1$

$$ON = \left\{ \begin{matrix} 000 \\ 001 \\ 111 \end{matrix} \right\} = \left\{ \begin{matrix} 00X \\ 111 \end{matrix} \right\}$$

$$OFF = \left\{ \begin{matrix} 011 \\ 010 \\ 101 \\ 110 \end{matrix} \right\} = \left\{ \begin{matrix} 01X \\ 101 \\ 111 \end{matrix} \right\}$$

$$DC = \{100\}$$

		$x_2 \ x_3$				
		$x_1$	00	01	11	10
$y_1$ :	0	1	1	0	0	
	1	d	0	1	0	



## 立方体集合 (覆盖)

◆ 函数  $y^2$

$$ON = \begin{Bmatrix} 001 \\ 011 \\ 101 \\ 110 \end{Bmatrix} = \begin{Bmatrix} X01 \\ 0X1 \\ 110 \end{Bmatrix}$$

$$OFF = \begin{Bmatrix} 000 \\ 010 \\ 111 \end{Bmatrix} = \begin{Bmatrix} 0X0 \\ 111 \end{Bmatrix}$$

$$DC = \{100\}$$

		x2 x3		00	01	11	10
y2:	x1	0	0	1	1	0	
	1	d	1	0	1		

$$DC = U_n - \{ON \cup OFF\}$$





## 覆盖表

$x_1x_2x_3$	$y^1y^2$	$x_1x_2x_3$	$y^1y^2$
00X	1u	X01	u1
111	1u	0X1	u1
01X	0u	110	u1
101	0u	0X0	u0
111	00		

### ◆ 覆盖表

- 布尔函数的另一种表示方法；
- 真值表的扩展

- X 表示无该因子；
- u 表示该乘积项与相应函数之间的关系未定义（未定义并不表示没有关系）。



## 立方体的术语 (1)

- ◆ 顶点: 最小方格  $(001|1)$ ,  $(100|u)$ ,  $(000|0)$ ,
- ◆ 立方体的维数
  - 0维立方体  $\Leftrightarrow$  顶点 — 无X, 1个方格
  - 1维立方体  $\Leftrightarrow$  棱 — 1个X, 2个方格,  $(01X|0)$
  - 2维立方体  $\Leftrightarrow$  面 — 2个X, 4个方格,  $(1XX|1)$
  - n维立方体 — n个X,  $2^n$ 个方格
- ◆ 蕴涵项——取值为1的立方体,
- ◆ 最小项——取值为1的顶点。

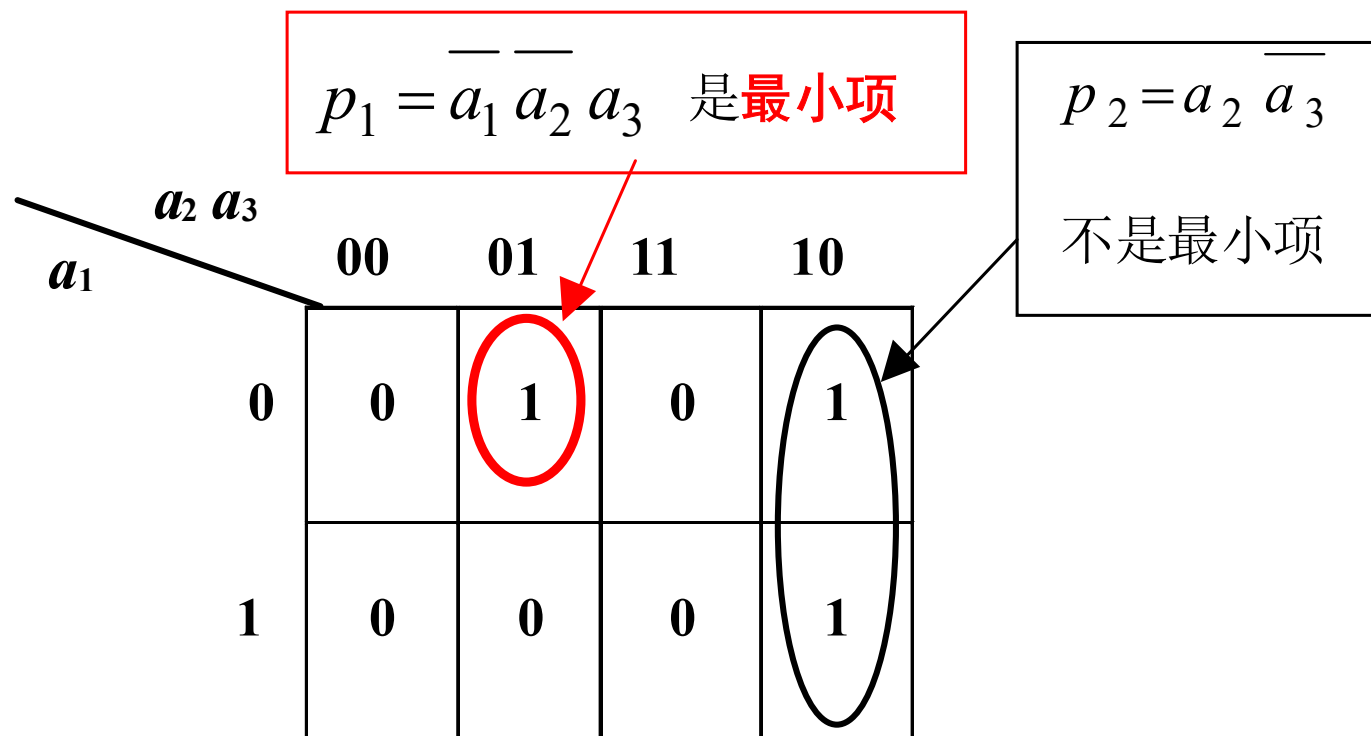


## 立方体术语 (2)

- ◆ 质立方体——质蕴涵项
- ◆ 必要质立方体——必要质蕴涵项
- ◆ 特征顶点——特征最小项
- ◆ 多输出函数的顶点:
  - 输入部分无X，输出部分只有一个0或1  
(101|1uu)
- ◆ 覆盖: 表示一个布尔函数的立方体集合



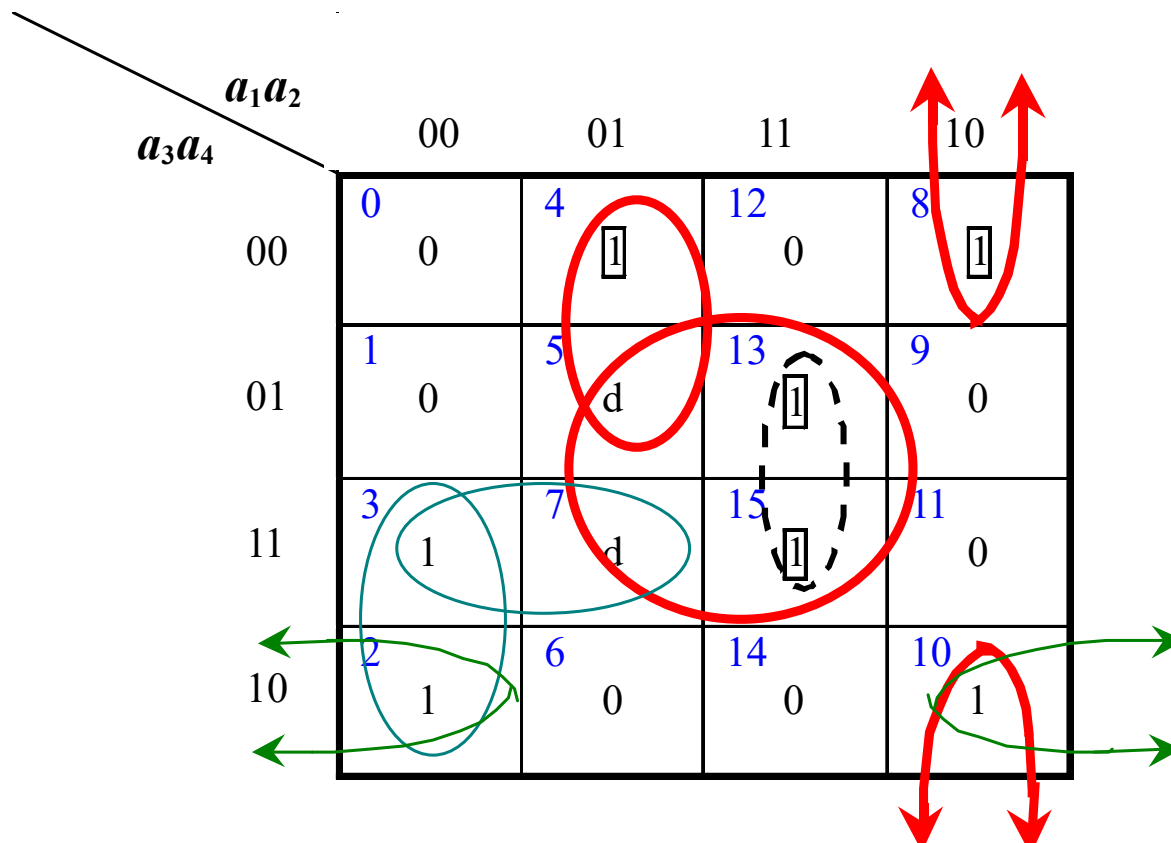
## 4.2 布尔函数的立方体表示



最小项在卡诺图中的表示



## 质蕴涵项和必要质蕴涵项在卡诺图中的表示



1: 特征顶点

⋯: 非质立方体 (蕴涵项)

○: 必要质立方体 (蕴涵项)    ○: 质立方体 (蕴涵项)



# 卡诺图和立方体 (Cube) 的对应关系

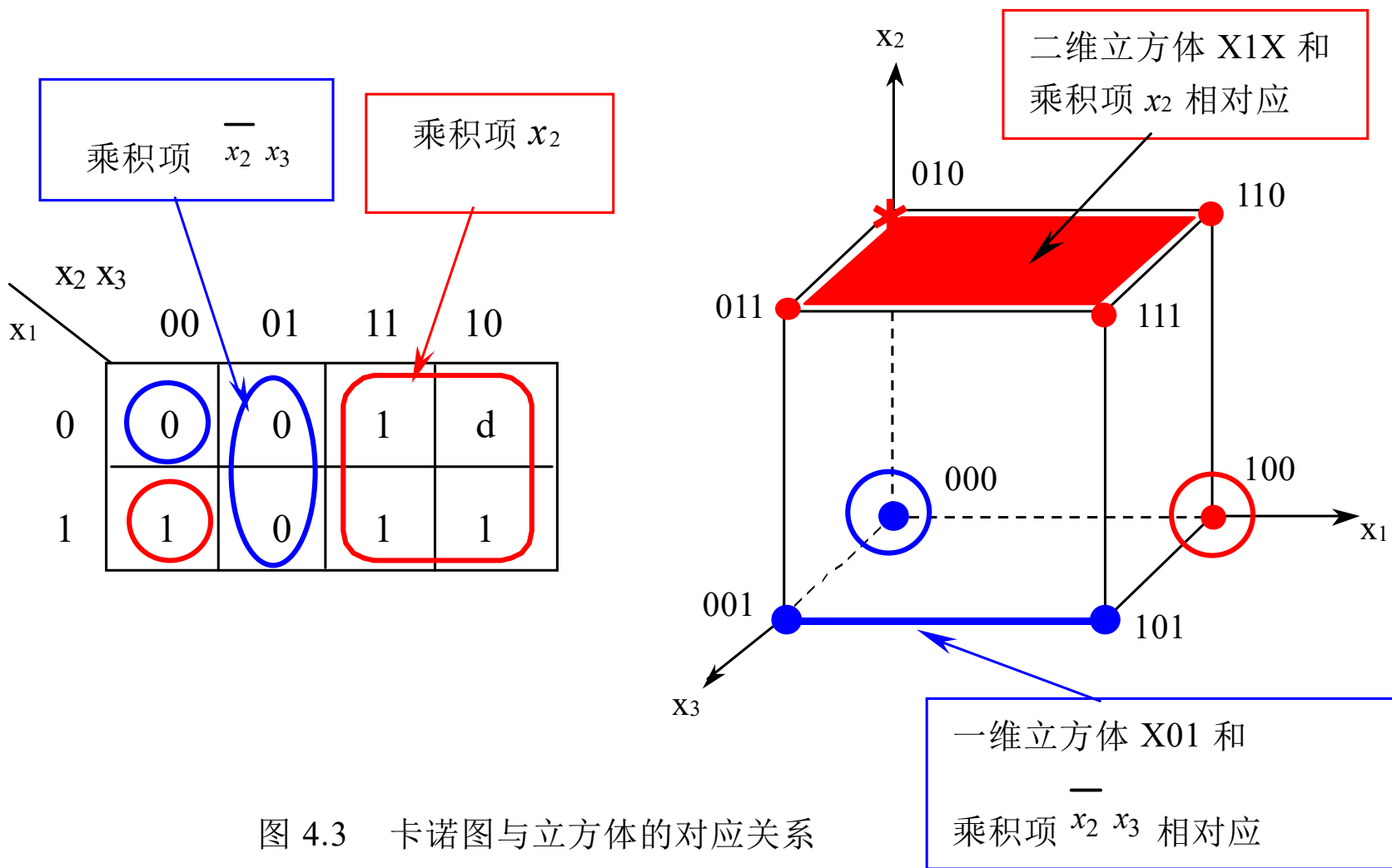


图 4.3 卡诺图与立方体的对应关系



## 立方体在计算机中的内部表示

- ◆ 用字符串表示立方体；
- ◆ 用二进制数表示立方体：

- 二进制数  $a$ :  $a_1 a_2 a_3 a_4 \dots a_{i-1} a_i \dots a_{n-1} a_n$

立方体  $x$ :  $\underbrace{\quad} x_1 \quad \underbrace{\quad} x_2 \quad \underbrace{\quad} x_k \quad \underbrace{\quad} x_m$

- 二进制数  $i$ :  $i_1, i_2, \dots i_k \dots i_n$

二进制数  $j$ :  $j_1, j_2, \dots j_k \dots j_n$

立方体  $x$ :  $x_1, x_2, \dots x_k \dots x_n$

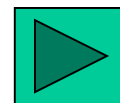


## 立方体在计算机中的内部表示

表 4.3 用 二 进 制 数 表 示 立 方 体

二进制数第 $k$ 位的取值	$i_k$	0	0	1	1
	$j_k$	0	1	0	1
立方体表示法中对应的变量 $c_i$ 的取值	方案之一	0	X	/	1
	方案之二	/	0	1	X

说明：还可以有许多别的方案，这里只列举了2个。



1



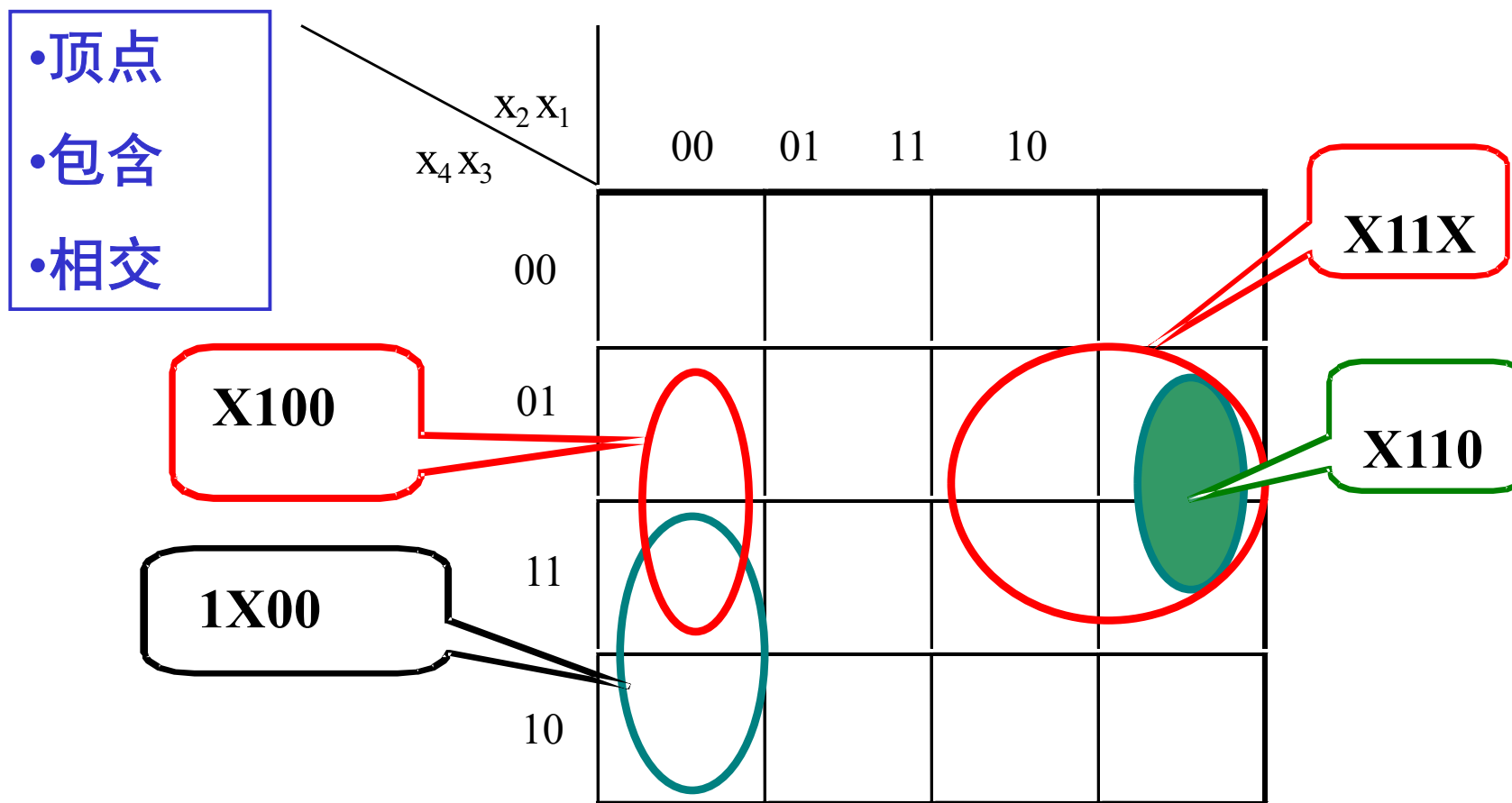
2





## 4.3 立方体运算

### -- 4.3.1 基本概念 --





## 立方体的包含判断

输入变量的包含判断

$C_i \subseteq a_i$ $C_i$	$a_i$			
		0	1	X
0		y	n	y
1		n	y	y
X		n	n	y

输出变量的包含判断

$d^j \subseteq b^j$ $d^j$	$b^j$			
		0	1	u
0		y	n	n
1		n	y	n
u		y	y	y



## 立方体的包含判断（续）

例1  $a|b$ : 10 1 | 1 u

$\subseteq$ )  $c|d$ : 10X | 1 1

---

y y y | y y

$\therefore a|b \subseteq c|d$

例2  $a|b$ : 10 1 1 | u 0 1

$\subseteq$ )  $c|d$ : 1XX1 | 1 0 u

---

y y y y | y y n

$\therefore a|b \not\subseteq c|d$

◆ 立方体间的包含关系是  
可传递的，即：

如果：  $c|d \subseteq a|b$

且：  $e|f \subseteq c|d$

则：  $e|f \subseteq a|b$

◆ 立方体间的包含关系是  
不可逆的，即：

如果：  $c|d \subseteq a|b$

但  $a|b \subseteq c|d$

不一定成立。



## 立方体的相交运算

- 单输出函数的立方体  $a$  和立方体  $c$  的“交”是它们的公共顶点所组成的立方体。
- 多输出函数的立方体  $a|b$  和立方体  $c|d$  的“交”是单输出函数“交”运算的扩展。
- 若两立方体之间无公共顶点，则说它们是分离的。

$$a \cap b = \phi$$



## 立方体的相交运算 (续)

表 4.6 相交运算表 A

$a_i$ $a_i \cap c_i$ $c_i$	0	1	X
0	0	q	0
1	q	1	1
X	0	1	X

表 4.7 相交运算表 B

$b^j$ $b^j \cap d^j$ $d^j$	0	1	u
0	0	q	0
1	q	1	1
u	0	1	u



## 立方体的相交运算（举例）

### ◆ 相交运算举例：

例1  $a|b$ : 10 1 | 1 u

$\cap$ )  $c|d$ : 10X | 1 1

---

1 0 1 | 1 1

例2  $a|b$ : 10 1X | u 0 0

$\cap$ )  $c|d$ : 10 X0 | 1 0 u

---

1 0 1 0 | 1 0 0

例3  $a|b$ : 1X 1 1 | 1 u

$\cap$ )  $c|d$ : 0 1X 1 | 1 0

---

q 1 1 1 | 1 0

➡ 相交结果为空

◆ 相交运算可以用于一致性判断：

见下页



## 立方体的一致性判断

- ◆ 用立方体定义函数时，立方体之间必须具备一致性。
- ◆ 两个立方体作相交运算时，若其输出部分出现 $q$ 而其输入部分不出现 $q$ 的话，则称此二立方体是不一致的，反之，它们是一致的。
- ◆ 在作逻辑综合之前，首先应对原始数据作一致性检查。若发现不一致性，则应纠正原始数据中的错误。



## 立方体的一致性判断（举例）

◆ 例1

$$\cap) \quad \begin{array}{cccc|ccc} 1 & X & 1 & 1 & 1 & u & 1 \\ 0 & 1 & X & 1 & u & 1 & u \end{array}$$


---

$$q \quad \begin{array}{cccc|ccc} 1 & 1 & 1 & & 1 & 1 & 1 \end{array}$$

→ 相交结果为空立方体，具备一致性。

◆ 例2

$$\cap) \quad \begin{array}{cccc|ccc} 0 & X & 0 & X & 1 & u & 1 \\ X & X & 0 & 1 & 0 & 1 & u \end{array}$$


---

$$\begin{array}{cccc|ccc} 0 & X & 0 & 1 & q & 1 & 1 \end{array}$$

→ 输入部分无q、输出部分有q，不具备一致性。

◆ 对于函数  $f^1$  而言，顶点（最小项）**0001**的取值到底是**0**还是**1**？





## 立方体的一致性判断（举例）

### ◆ 例3

$$\cap) \quad \begin{array}{cccc|ccc} 1 & 1 & X & 1 & 0 & u & 1 \\ 1 & X & 1 & 0 & 1 & 0 & u \end{array}$$

---

$$\begin{array}{cccc|ccc} 1 & 1 & 1 & q & q & 0 & 1 \end{array}$$

➔ 输入部分有q，输出部分也有q，是空立方体，具备一致性。



## 覆盖和函数(cover and function)

- ◆ 输入、输出变量相同的，逐对一致的非退化立方体集合称为覆盖 $C$ 。
- ◆ 覆盖 $C$ 中每一立方体所包含的每一顶点都表示输入变量和输出变量的对应关系，因此整个覆盖 $C$ 就代表了函数 $f$ 。记作：

$$f = F(C)$$



## 覆盖运算：吸收，并，交

### ◆ 吸收 $C'=S[C]$ :

- 去掉被包含的立方体
- 去掉输出全为u的立方体

吸收运算的目的：  
在不改变覆盖所描述的函数性质的前提下减少覆盖中元素的个数。

### ◆ 并：合并再吸收

设  $A = \{ a^1, a^2, \dots, a^i \}$

$B = \{ b^1, b^2, \dots, b^s \}$

则  $A \cup B = S\{ a^1, a^2, \dots, a^i, b^1, b^2, \dots, b^s \}$



## 覆盖的“交”运算

- ◆ 交运算：两覆盖的立方体逐对相交，运算结果还是一个覆盖。

$$b \cap A = \cup (b \cap a) \quad \text{其中 } a \in A$$

$$B \cap A = \cup (b \cap A) \quad \text{其中 } b \in B$$

说明：大写字母表示覆盖，小写字母表示立方体。



## 面与余面

### ◆ 面(face):

- 若  $a \subseteq b$ , 则称  $a$  为  $b$  的面。
- 求立方体  $b$  的面:
  - 输入:  $x$  改 0 或 1;
  - 输出: 0 或 1 改  $u$

### ◆ 余面(coface):

- 对  $a \in C$ ,  $\exists e$  ( $e$  可不属于  $C$ ),  $a \subseteq e$ , 且  $e$  与  $C$  中各立方体均一致, 称  $e$  为  $a$  的相对于  $C$  的一个余面。
- 可见  $e$  不仅和  $a$  有关; 还和  $C$  有关。



## 求余面的方法

- (1) 从覆盖 $C$ 中选定某一立方体 $c$ 。
- (2) 把 $c$ 的某一个或几个输入变量的取值由原来的 $\{0或1\}$ 改为 $X$ 。
- (3) 把 $c$ 的某一个或几个输出变量的取值由原来的 $u$ 改为 $\{0或1\}$ 。
- (4) 把新形成的立方体 $e$ 和覆盖 $C$ 作相交运算，并判断 $e$ 是否和 $C$ 中每一立方体相一致。若全部一致，则 $e$ 是一个余面。

➔ 从定义可知，余面必须和初始覆盖 $C$ 一致，但并不要求余面和余面一致。



## 求余面举例

C为初始覆盖，对其中的立方体求余面：

$$C = \left\{ \begin{array}{cc|c} 1 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right\} \Rightarrow \left\{ \begin{array}{cc|c} 1x & & 0 \\ x1 & & 0 \\ & 0x & 1 \\ & x0 & 1 \end{array} \right\}$$

◆ 注意：形成的余面之间可以不一致

1x|0 与 x0|1 不一致  
x1|0 与 0x|1 不一致



## 4.3.2 相交和包含判断的具体实现

- ◆ 设有两个立方体:

$$a = a_1 a_2 \dots a_k \dots a_n$$

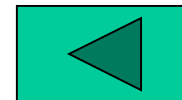
$$b = b_1 b_2 \dots b_k \dots b_n$$

其运算结果为立方体c:

$$c = c_1 c_2 \dots c_k \dots c_n$$

讲解具体实现的目的：  
实例引导

- ◆ 我们用2个二进制数合起来代表1个立方体。并且采用表4.3中的方案一（具体实现必须和编码有关）；
  - $(a_i, a_j)$ 代表立方体a,
  - $(b_i, b_j)$ 代表立方体b,
  - $(c_i, c_j)$ 代表立方体c:







## 相交运算的具体实现 (续)

(1) 若:  $(a_i \text{ XOR } b_i) \text{ AND } (a_j \text{ XOR } b_j) = 0$

则立方体 $a$ 和 $b$ 相交。

$$c = a \cap b$$

(2) 若条件(1)不满足, 则立方体 $a$ 和 $b$ 不相交;

$$c = \phi$$

说明: 具体实现的算法和编码有关, 这里只是举例。



## 包含判断的具体实现（续）

若下述两个条件

$$\begin{cases} a_i \text{ AND } b_i = b_i \\ a_j \text{ AND } b_j = a_j \end{cases}$$

成立，则：

$$a \subseteq b$$

说明：具体实现的算法和编码有关，这里只是举例。



## 相交和包含判断的具体实现 (续)

表 4.8 相交运算和包含判断(方案一)

立方体表示	$a_k$	0 1	XX	0 1 0 1 X
	$b_k$	1 0	0 1	0 1 XXX
对应的 二进制数表示	$ai_k$	0 1	0 0	0 1 0 1 0
	$aj_k$	0 1	1 1	0 1 0 1 1
	$bi_k$	1 0	0 1	0 1 0 0 0
	$bj_k$	1 0	0 1	0 1 1 1 1

前面的公式是由  
这个表导出

$$\begin{array}{cc}
 \underbrace{\hspace{10em}} & \underbrace{\hspace{10em}} \\
 a_k \subseteq b_k = n & a_k \subseteq b_k = y \\
 \underbrace{\hspace{10em}} & \underbrace{\hspace{10em}} \\
 a_k \cap b_k = q & a_k \cap b_k \neq q
 \end{array}$$



# 思考：

## 立方体运算要解决哪些问题？

冗余立方体

	$x_2 x_1$	00	01	11	10
$x_4 x_3$	00	0	1	0	0
	01	0	d	1	1
	11	1	1	1	0
	10	0	0	1	0

最小化覆盖中的立方体：

- 必须是质立方体；
- 立方体总数尽量少；
- 立方体维数尽量大；



- 首先求得全部质立方体集合 $Z$ ；
- 从 $Z$ 中挑选一个适当的子集；
  - 怎样挑选？先挑维数大的吗？

• 覆盖 $C$ ：

• 立方复合体 $K$ ：

$$\left\{ \begin{array}{l} (1) \text{ 覆盖 } C \text{ 中的每一个立方体都属于 } K; \\ (2) \text{ 设 } e \in K, \text{ 则 } e \text{ 的面以及 } e \text{ 相对于 } C \text{ 的余面都属于 } K. \end{array} \right.$$

• 质立方体：  $z \in K$ ，设 $z$ 不是 $K$ 中任何其他立方体的面，则 $z$ 是 $C$ 的质立方体。

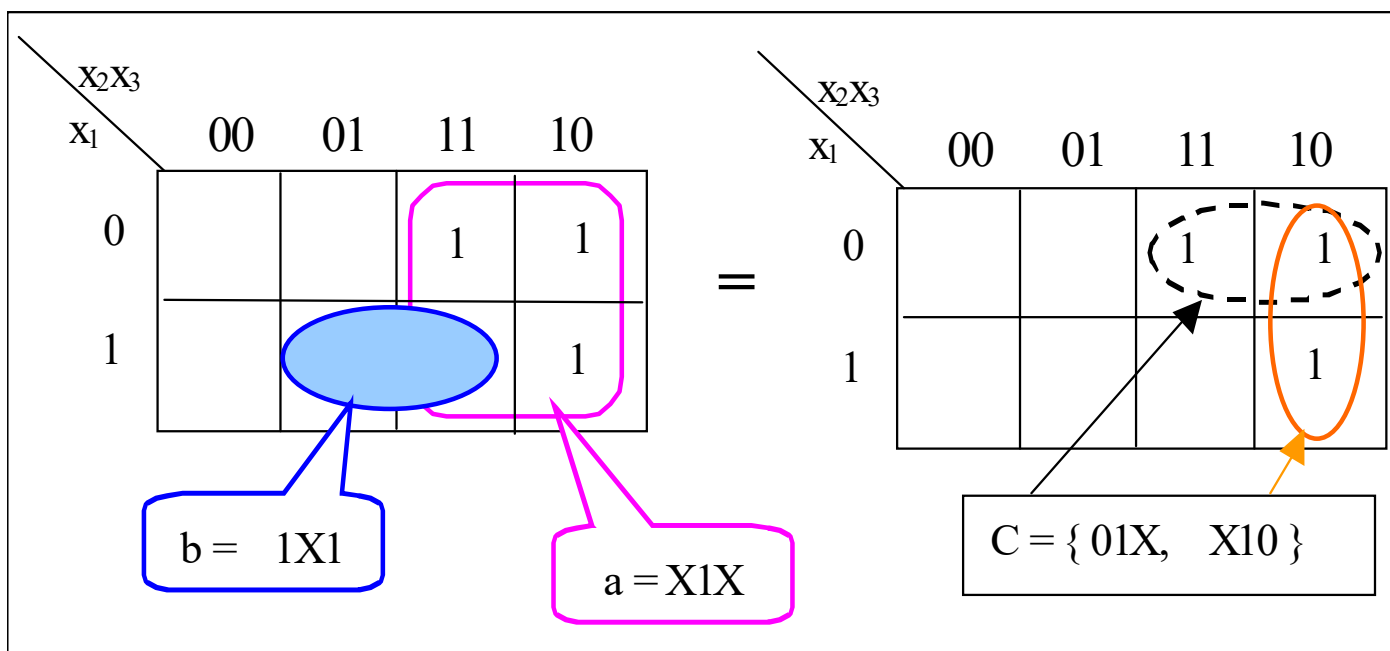


## 4.3.3 锐积运算 (sharp product)

### (1) 单输出函数的锐积 (#)

◆ 概念:

- $C = a \# b$  (C是立方体集合)
- 由顶点观点, 顶点集合求差
- 剩余顶点形成维数尽量大的立方体集合

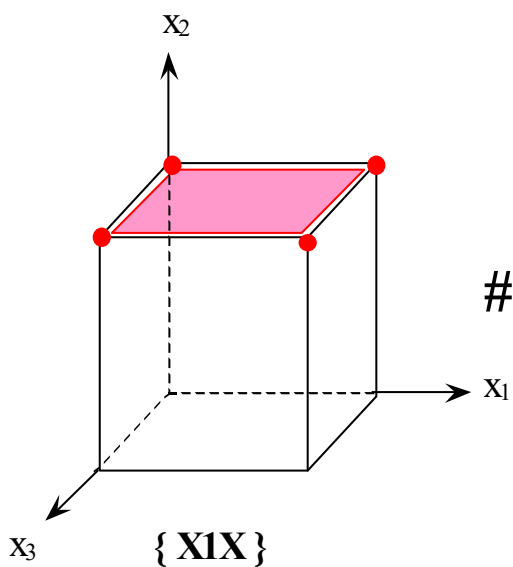




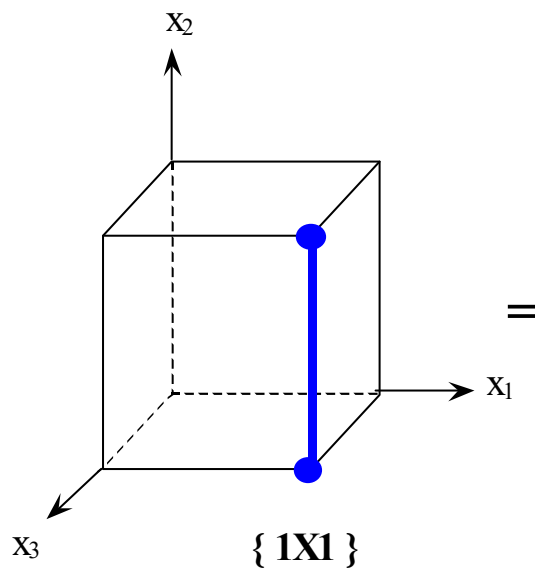
$X_2X_3$		$X_1$			
		00	01	11	10
$X_1$	0			1	1
	1			1	1

$X_2X_3$		$X_1$			
		00	01	11	10
$X_1$	0				
	1		1	1	

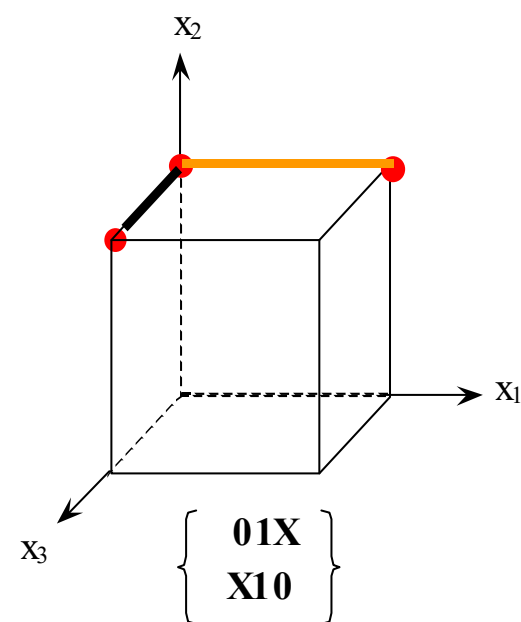
$X_2X_3$		$X_1$			
		00	01	11	10
$X_1$	0			1	1
	1				1



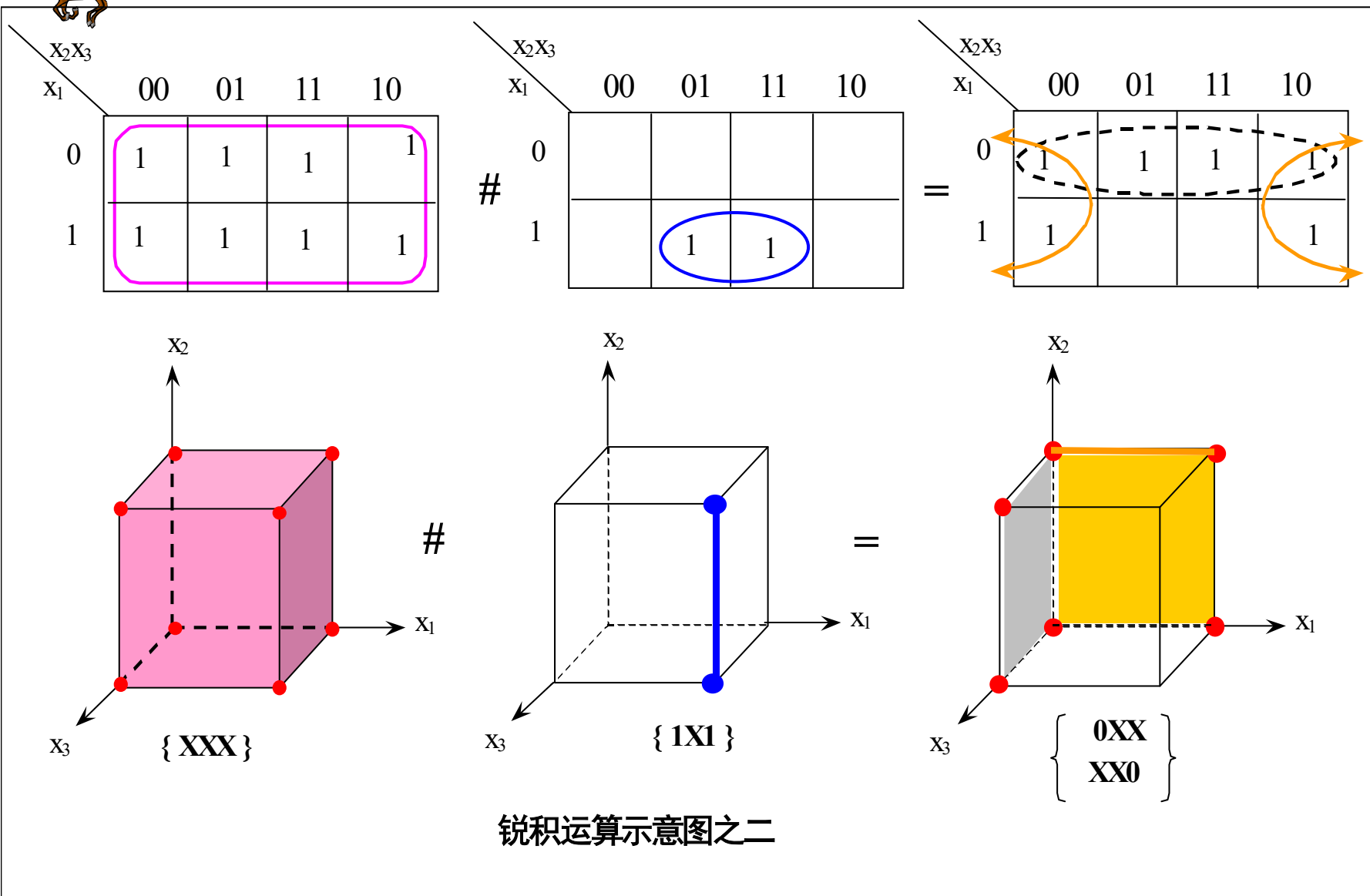
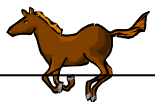
#



=



锐积运算示意图之一





## 单输出函数的锐积 (续)

- $a \# b$  结果是一个立方体的集合  $C$  ;  
由  $a$  包含的顶点减去  $a \cap b$  包含的顶点所形成的维数最大的立方体都是  $C$  的元素。

$$K^0(C) = K^0(a) - K^0(a \cap b)$$

- 当  $a$  和  $b$  不相交时:

$$K^0(a \cap b) = \phi \text{ (空集)}$$

$$K^0(C) = K^0(a)$$

$$\Rightarrow a \# b = a$$

- 当  $b$  包含  $a$  时:

$$K^0(a \cap b) = K^0(a)$$

$$K^0(C) = \phi \text{ (空集)}$$

$$\Rightarrow a \# b = \phi$$

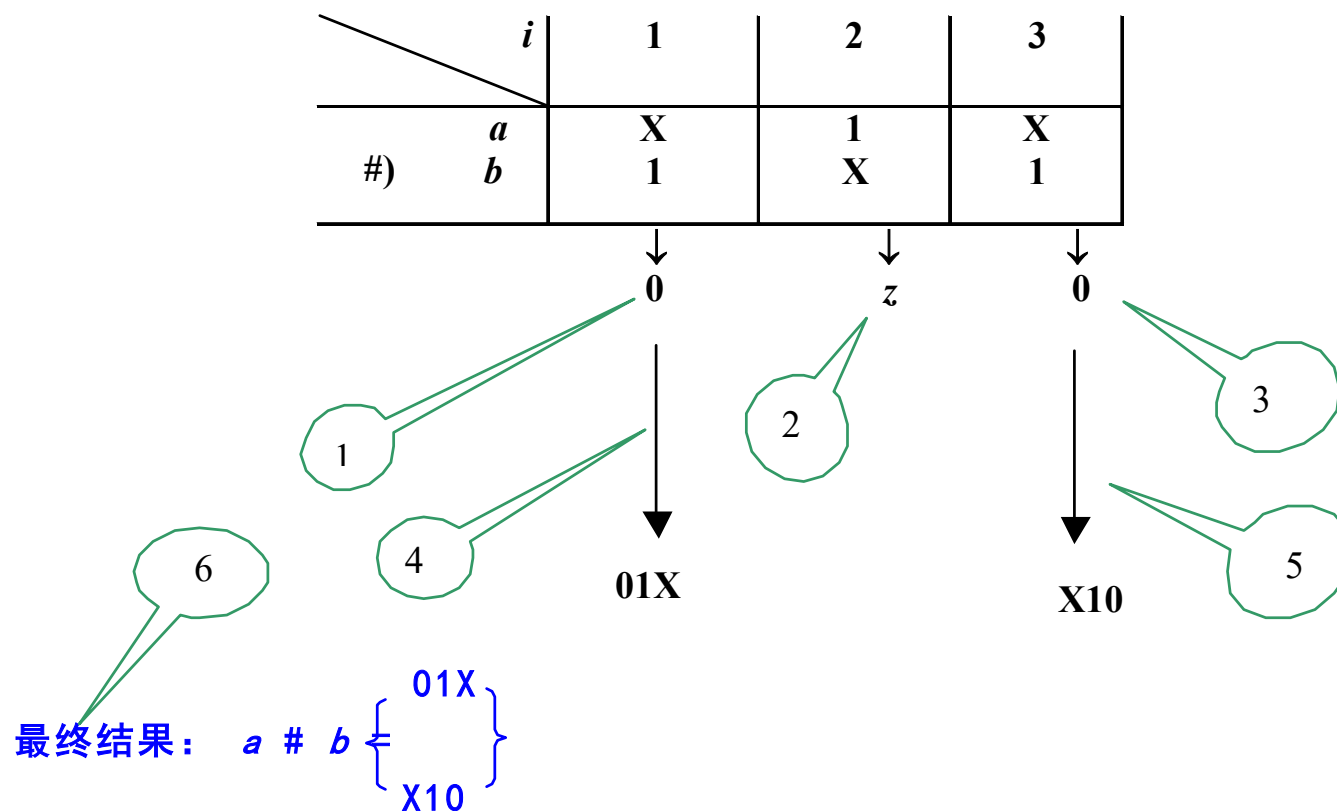
$K^0(a)$ 表示立方体 $a$   
所包含的顶点集合





## 单输出函数的锐积 (续)

- 若  $b$  包含  $a$  的一部分，按位找出剩余，与其余位组合。
- 举例： 设  $a = X1X$   $b = 1X1$  求  $a \# b$





小结： 锐积运算可由表 4. 10 和下述规则来定义。

表 4. 10 锐积运算表

$a_i \# b_i$	$b_i$			
	$a_i$	0	1	X
0		$z$	$y$	$z$
1		$y$	$z$	$z$
X		1	0	$z$

(1) 若存在一个  $i$  ( $i$  从 1 到  $n$ ) 能使  $a_i \# b_i = y$  成立, 则

$$a \# b = a$$

(2) 若对所有的  $i$  ( $i$  从 1 到  $n$ ) 都使  $a_i \# b_i = z$  成立, 则

$$a \# b = \phi$$

(3) 不满足以上条件时, 若有:

$$a_i \# b_i = \alpha_i \in \{0, 1\}$$

则对应于一个立方体  $f^i$ :

$$a \# b = \bigcup_i \{f^i\}$$

$$a \# b = \bigcup_i \{a_1 a_2 \cdots a_{i-1} \alpha_i a_{i+1} \cdots a_n\}$$



## 立方体锐积的具体实现

假定用二进制数表示立方体并采用表4.3中的方案二 (p. 24):



- (1) 令结果  $C$  为空。
- (2) 作:  $e = ai \text{ AND } bi$                        $d = aj \text{ AND } bj$
- (3) 若:  $e = ai$  且  $d = aj$               则运算结束。
- (4) 若;  $e \text{ OR } d \neq 11...1...1$       则  $C = a$ , 运算结束。
- (5) 若条件(3)和条件(4)都不满足, 作:

$$l = (ai \text{ AND } aj) \text{ AND } (bi \text{ XOR } bj)$$

重复以下步骤  $n$  次(  $k$  从 1 到  $n$  ):

若  $l_k = 1$ , 则:

$$fi = ai_1 ai_2 \cdots ai_{k-1} bj_k ai_{k+1} \cdots ai_n$$

$$fj = aj_1 aj_2 \cdots aj_{k-1} bi_k aj_{k+1} \cdots aj_n$$

( $f_i, f_j$ ) 代表一个立方体  $f$ 。

$$C = C \cup f$$





## 锐积的用途

### ◆ 判断包含：

若  $a \# b = \phi$  则  $a \subseteq b$

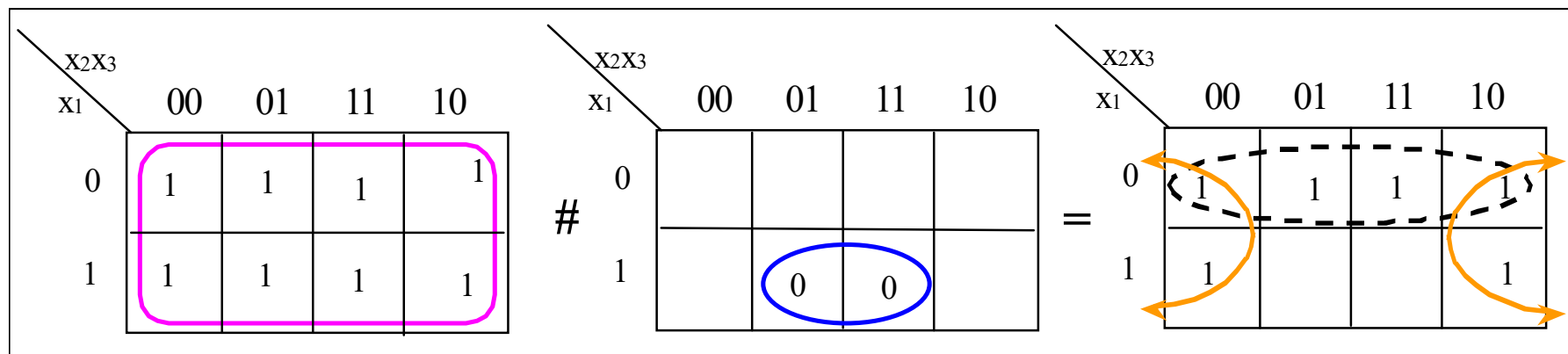
### ◆ 求质立方体集合Z:

$$Z = S[U_n \# \text{OFF}]$$

$U_n$ 代表全集合；

例：

这不是严格的定义！





## (2) 多输出函数锐积

$$\underline{p = a|b \# c|d = ?}$$

### ◆ 原始想法：

- 多输出函数  $\Rightarrow$  单输出函数；
- 按照单输出函数的规则作锐积；
- 将单输出函数的锐积结果合并  $\Rightarrow$  多输出函数形式。

### ◆ 改进：

- 上述方法完全可行；但是如果能用多输出函数形式直接作锐积，则有利于编程；
- 下面是直接用多输出函数形式作锐积的算法描述；
  - 理解稍有困难，但描述简练，有利于编程。



## (2) 多输出函数锐积

$$\underline{p = a|b \# c|d = ?}$$

1. 若 $a|b$  和  $c|d$  相分离, 则  $a|b \# c|d = a|b$  。
2. 当 $a \subseteq c$ , 则  $p = \{ a|e \}$   
若  $b^j = \beta \in \{0, 1\}$  且  $d^j = u$  ( $j$ 从1到 $m$ )  
则  $e^j = b^j$ , 否则  $e^j = u$
3. 若不满足上述条件,其锐积由下列立方体集合:  
 $\{ p1, P2 \}$  组成:
  - $p1 = \{ a|e \}$ ,  
若  $b^j = \beta \in \{0, 1\}$  且  $d^j = u$  则  $e^j = b^j$ , 否则  $e^j = u$
  - $P2 = \cup (f^i | b)$ ,  $f^i$  与单输出( $a \# c$ ) 相同(p.51)。





## 多输出函数锐积举例

例:  $1001|u10 \# 0x11|10u = 1001|u10$  (分离)

例: 

x011	1u1
xx1x	11u
<u>     </u>	<u>     </u>
<b>zzzz</b>	uu1

 $\Rightarrow \{ x011 | uu1 \}$  (包含)

例  $\begin{array}{c} x0x1 | 1u1 \\ 1x01 | 11u \\ \hline 0z1z | uu1 \end{array} \Rightarrow \left\{ \begin{array}{l} x0x1 | uu1 \\ 00x1 | 1u1 \\ x011 | 1u1 \end{array} \right\}$

例  $\begin{array}{c} x0x1 | 1u10 \\ 1x01 | 11uu \\ \hline 0z1z | uu10 \end{array} \Rightarrow \left\{ \begin{array}{l} x0x1 | uu10 \\ 00x1 | 1u10 \\ x011 | 1u10 \end{array} \right\}$



## 覆盖的锐积运算

- ◆ 设覆盖A和覆盖B表示为：

$$A = \{a^1, a^2, \dots, a^i, \dots, a^m\}$$

$$B = \{b^1, b^2, \dots, b^i, \dots, b^w\}$$

其中  $a^i$  是覆盖A中的立方体；  $b^i$  是覆盖B中的立方体。

$$A \# b = (a^1 \# b) \cup (a^2 \# b) \cup (a^3 \# b) \cdots \cup (a^m \# b) = \bigcup_{a^i \in A} (a^i \# b)$$

$$\begin{aligned} a \# B &= \{[(a \# b^1) \# b^2] \# \cdots \# b^w\} \\ &= \{(a \# b^1) \cap (a \# b^2) \cap \cdots \cap (a \# b^w)\} \end{aligned}$$

$$A \# B = \bigcup_{a^i \in A} (a^i \# B)$$





## 不相交锐积 $\textcircled{\#}$ 与普通锐积 $\#$ 的比较

$$\underline{a\textcircled{\#}b = D, a\#b = C}$$

### ◆ 共同点:

- C, D均为立方体集合
- 其顶点均为集合求差, 所包含的顶点相同。

### ◆ 不同点:

- C要求维数最大(质立方体); D要求立方体互不相交。
- C结果唯一, D结果不唯一。



## 不相交锐积计算举例

◆ 实例:

$a = X0XX$     $b = 1X01$    求  $a \# b$  和  $a \oplus b$

$X0XX \dots a$

#)  $1X01 \dots b$

0z10



00XX

X01X

X0X0

$X0XX \dots a$

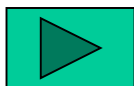
(#)  $1X01 \dots b$

0z10

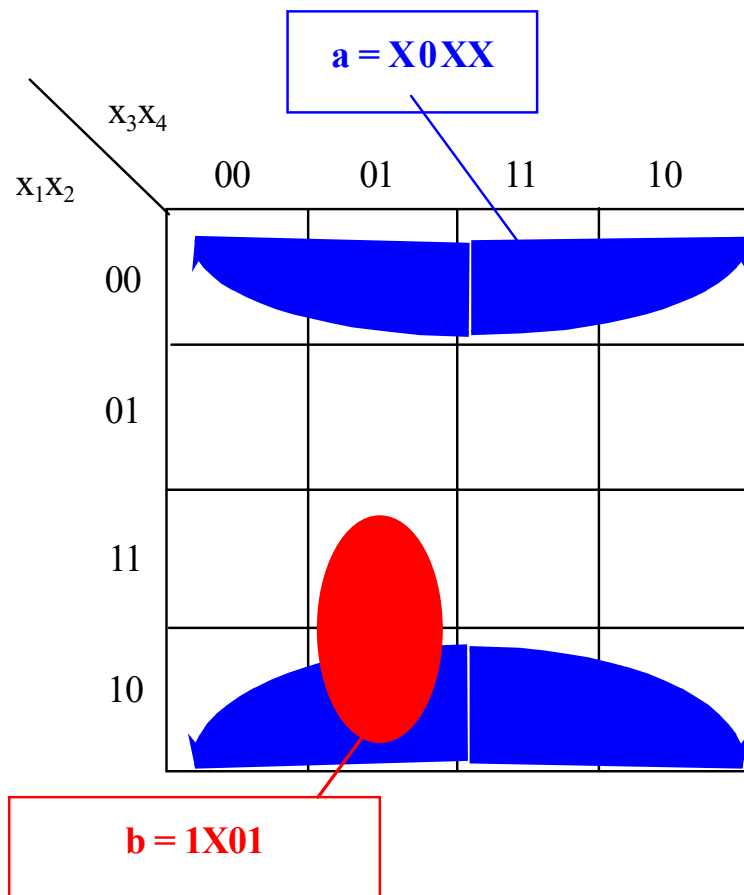
00XX

101X

1000



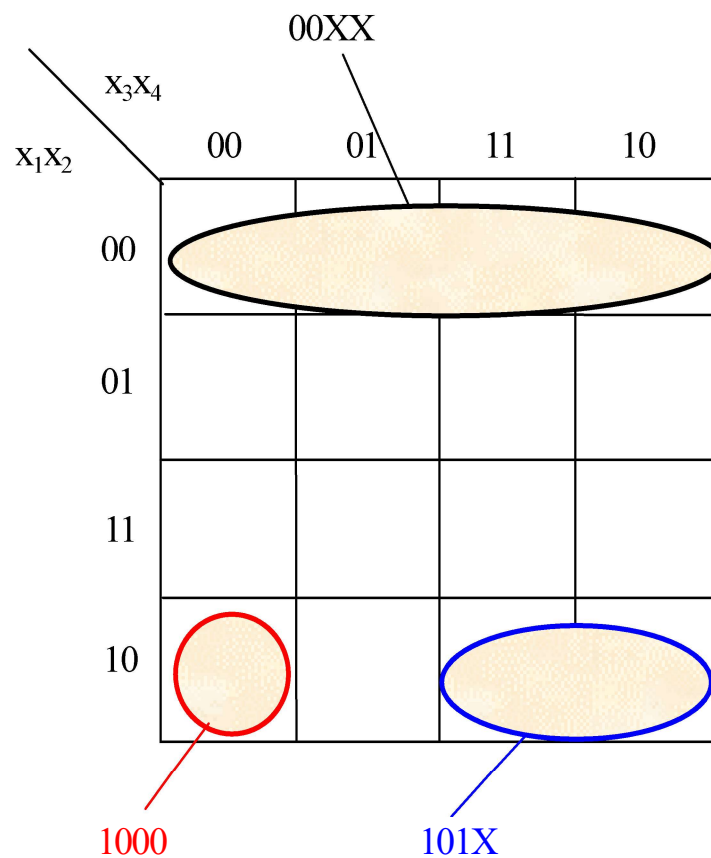
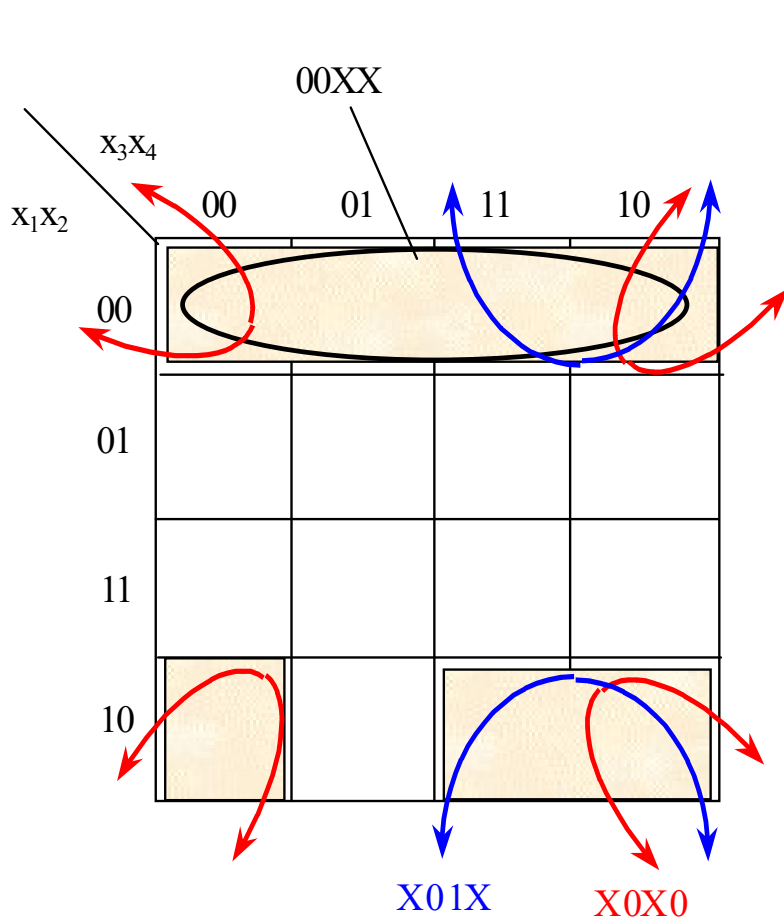
◆ 卡诺图:





## 不相交锐积计算举例

◆ 举例：





## 单输出函数不相交锐积运算规则：

表 4.10 锐积运算表

$a_i \# b_i$	$b_i$			
	$a_i$	0	1	X
0		$z$	$y$	$z$
1		$y$	$z$	$z$
X		1	0	$z$

(1) 若存在一个  $i$  ( $i$  从 1 到  $n$ ) 能使  $a_i \# b_i = y$  成立，则

$$a \# b = a$$

(2) 若对所有的  $i$  ( $i$  从 1 到  $n$ ) 都使  $a_i \# b_i = z$  成立，则

$$a \# b = \phi$$

(3) 不满足以上条件时，若有：

$$a_i \# b_i = \alpha_i \in \{0, 1\}$$

则对应于一个立方体  $f^i$ ：

$$a \# b = \bigcup_i \{f^i\}$$

$$a \# b = \bigcup_i \left\{ (a_1 \cdot b_1)(a_2 \cdot b_2) \cdots (a_{i-1} \cdot b_{i-1}) \underline{\alpha_i} a_{i+1} \cdots a_n \right\}$$



## 多输出函数不相交锐积计算规则

规则1 若  $(a \mid b)$  和  $(c \mid d)$  是分离的, 则  $(a \mid b) \oplus (c \mid d) = (a \mid b)$

规则2 若  $a \subseteq c$  则  $(a \mid b) \oplus (c \mid d) = (a \mid e)$ 。

若:  $b^j = \beta \in \{0, 1\}$  且  $d^j = u$  ( $j$ 从1到 $m$ ) 则:  $e^j = b^j$   
 否则:  $e^j = u$

规则3 若上述规则都不满足, 则  $(a \mid b) \oplus (c \mid d)$  的结果由下列立方体组成:

(1)  $a \mid g$

若:  $d^j = u$  ( $j$ 从1到  $m$ ) 则:  $g^j = b^j$

否则:  $g^j = u$

(2)  $\bigcup_i f^i \mid h$

若:  $a_i \oplus c_i = \alpha_i \in \{0, 1\}$  则对应于一个立方体  $f^i$ :

$$f^i = (a_1 \cdot b_1)(a_2 \cdot b_2) \cdots (a_{i-1} \cdot b_{i-1}) \underline{\alpha_i} a_{i+1} \cdots a_n$$

$$\text{而: } h^j = \begin{cases} u & \text{若 } d^j = u \\ b^j & \text{若 } d^j \neq u \end{cases}$$



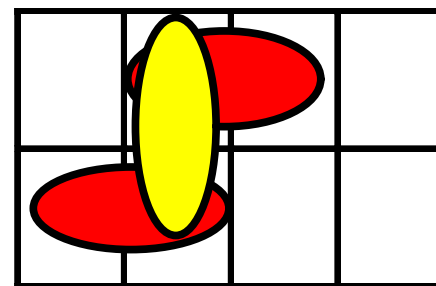
## 4.3.4 星积 \* (star product)运算

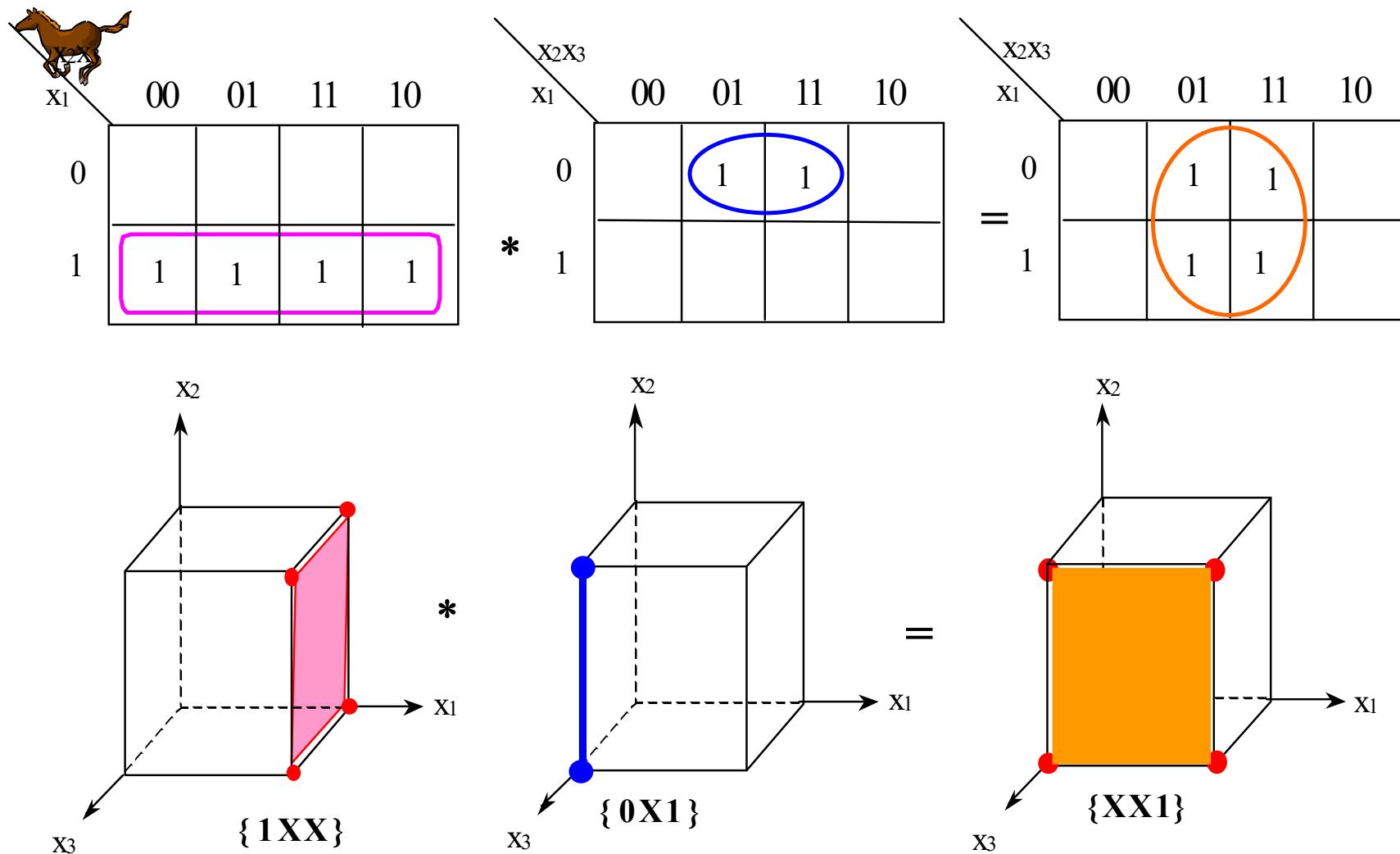
### (相容运算)

◆ 两个立方体星积运算后生成新的最大维数立方体

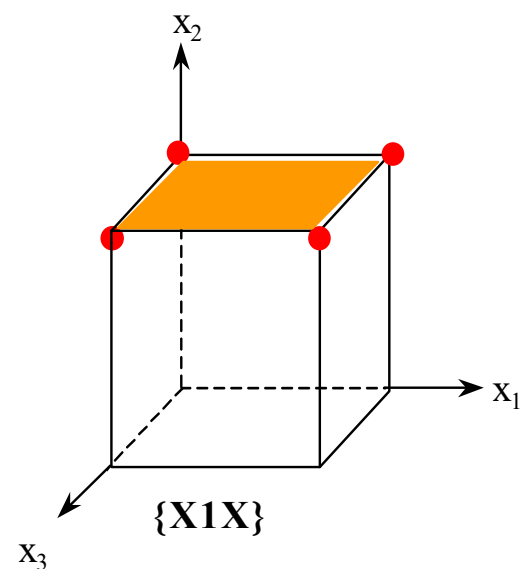
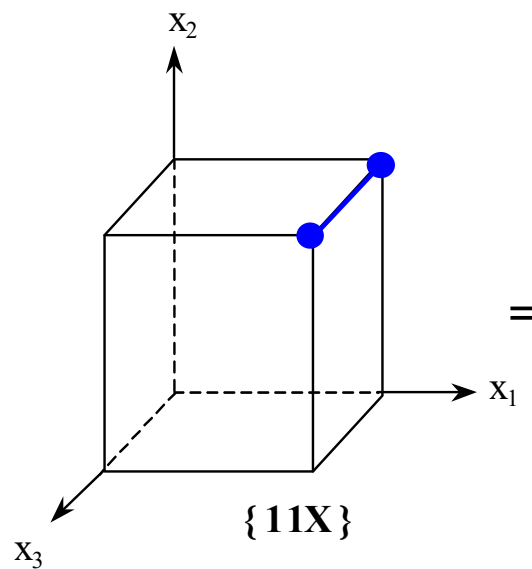
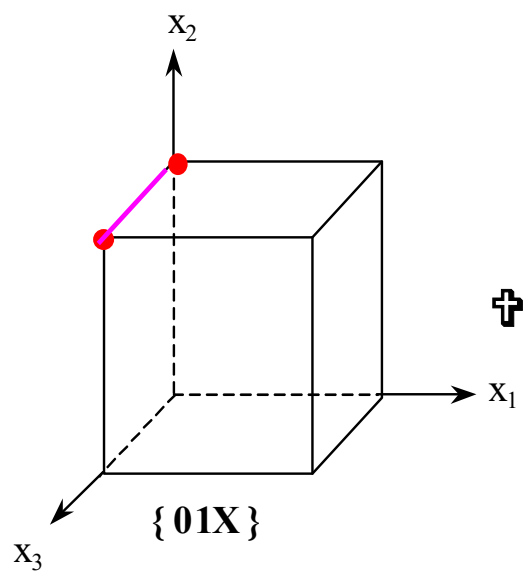
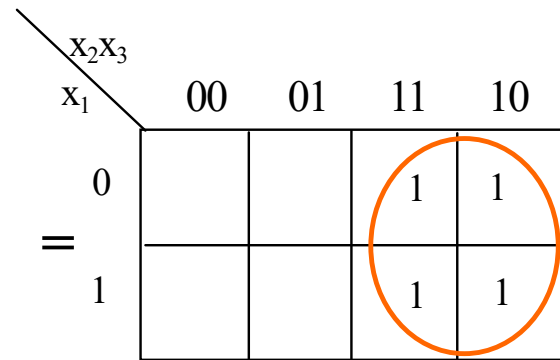
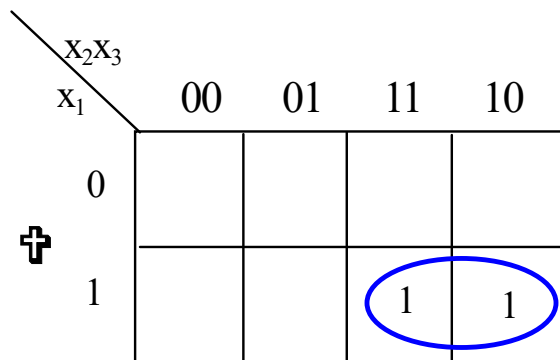
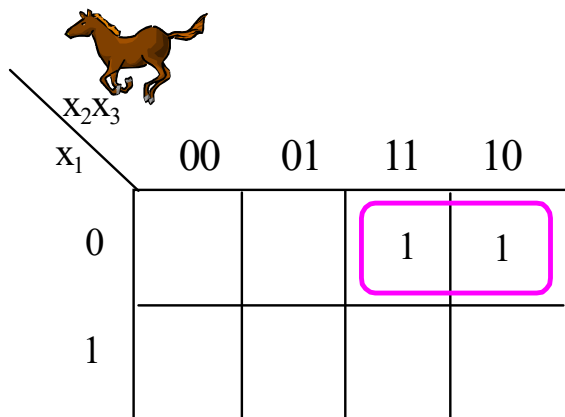
◆ 特点:  $c = a * b$

1.  $c \not\subseteq a$ , 且  $c \not\subseteq b$ , 但  $c \subseteq a \cup b$
2. 仅当  $a \cap b = \emptyset$ , 且  $a$  与  $b$  相邻时,  
 $c \neq \emptyset$
3.  $c$  维数最大, 为唯一立方体。





星积运算示意图



星积运算示意图





## 星积运算规则

星积运算表:

$a_i$ $a_i * b_i$ $b_i$			
	0	1	X
0	0	q	0
1	q	1	1
X	0	1	X

(1) 如果  $a_i * b_i$  ( $i = 1, 2, \dots, n$ )  
中有一个且仅有一个等于  $q$  时, 则

$$c = a * b$$

其中:

$$a_i * b_i = q \text{ 时 } c_i = X$$

$$a_i * b_i \neq q \text{ 时 } c_i = a_i * b_i$$

(2) 不满足条件(1)时:

$$a_i * b_i = \emptyset \quad (\text{空立方体})$$



## 星积运算举例

例1:  $a = 1\ X\ X$   $b = 0\ X\ 1$

求  $a * b$

		$i$		
		1	2	3
*)	$a$	1	X	X
	$b$	0	X	1
		<b>q</b>	X	1

上述按位运算结果中只有一个 **q**, 用 X 去置换 **q**, 得:

$$c = a * b = \text{XX1}$$

例2:  $a = X\ X\ 1$   $b = 0\ X\ 1$

求  $a * b$

		$i$		
		1	2	3
*)	$a$	X	X	1
	$b$	0	X	1
		0	X	1

上述按位运算结果中 **q** 的个数为 0, 故:

$$c = a * b = \emptyset$$



## 星积运算的具体实现

◆ 用两个二进制数代表一个立方体，并采用表4.3中方案二 (p. 24)。

(1) 作:  $w = (ai \text{ XOR } bi) \text{ AND } (aj \text{ XOR } bj)$

(2) 若:  $w = 0$  则  $c = \emptyset$ , 算法结束。

(解释:  $w = 0$  意味着  $q$  的个数为 0)

(3) 若:  $(w) \text{ AND } (w - 1) = 0$  则:

(解释: 此条件意味着  $q$  的个数为 1)

$$ci = (ai \text{ AND } bi) \text{ OR } (w)$$

$$cj = (aj \text{ AND } bj) \text{ OR } (w)$$

否则:  $c = \emptyset$  (解释:  $q$  的个数大于 1)



## 4.4 多输出函数与单输出函数的阵列变换

- ◆ 多个单输出函数描述  $\Rightarrow$  一个多输出函数描述;
- ◆ 一个多输出函数描述  $\Rightarrow$  多个单输出函数描述;



## 多个单输出函数描述 $\Rightarrow$ 一个多输出函数描述

### ◆ 单输出函数描述:

$$CON^1 = \{0XX\}$$

$$COFF^1 = \left\{ \begin{array}{l} 10X \\ 110 \end{array} \right\}$$

$$CON^2 = \left\{ \begin{array}{l} 0X1 \\ X01 \end{array} \right\}$$

$$COFF^2 = \{11X\}$$

$$CON^3 = \left\{ \begin{array}{l} 000 \\ 110 \end{array} \right\}$$

$$COFF^3 = \{1X1\}$$

### ◆ 多输出函数描述:

提示:

可以用  
交运算  
和吸收  
运算把  
C合并  
化简

$$C = \left\{ \begin{array}{l|l} 0XX & 1uu \\ 10X & 0uu \\ 110 & 0uu \\ 0X1 & u1u \\ X01 & u1u \\ 11X & u0u \\ 000 & uu1 \\ 110 & uu1 \\ 1X1 & uu0 \end{array} \right\}$$



## 一个多输出函数描述 $\Rightarrow$ 多个单输出函数描述

---

◆  $j$ 从1到 $m$ 重复以下步骤 $m$ 次:

(1) 令 $CON^j = \emptyset$ ,  $COFF^j = \emptyset$ ,

(2) 扫描 $C$ 中每一立方体,

- 若  $y^j$  取值为1, 则令该立方体的输入部分加入  $CON^j$ ,
- 若  $y^j$  取值为0, 则令该立方体的输入部分加入  $COFF^j$ 。





## 4.5.1 锐积求质立方体集合Z

### ◆ 步骤:

$$T = S (U_n \# Coff)$$

$$-- Coff = S [U_n \textcircled{\#} Con \textcircled{\#} Cdc]$$

$$Z = \{ t \mid t \in T, \text{ 且 } t \cap Con \neq \emptyset \}$$



### ◆ 思考:

1. 为什么用  $U_n \# Coff$ ? 用不相交锐积可以吗?
2.  $S$  的含义?
3. 为什么求  $Coff$  用不相交锐积? 用普通锐积可以吗?
4. 第二个式子的意义?





## 4.5.2 迭代星积求质立方体集合Z

### ◆ 步骤：

1.  $C = S[Con \cup Cdc]$

2. 重复做：  $C = S[C \cup (C * C)]$ , 直到  $C * C = \emptyset$  ;

其中  $C * C = \{ c^i * c^j \mid i < j \}$

3.  $Z = \{ c \mid c \in C \text{ 且 } c \cap Con \neq \emptyset \}$

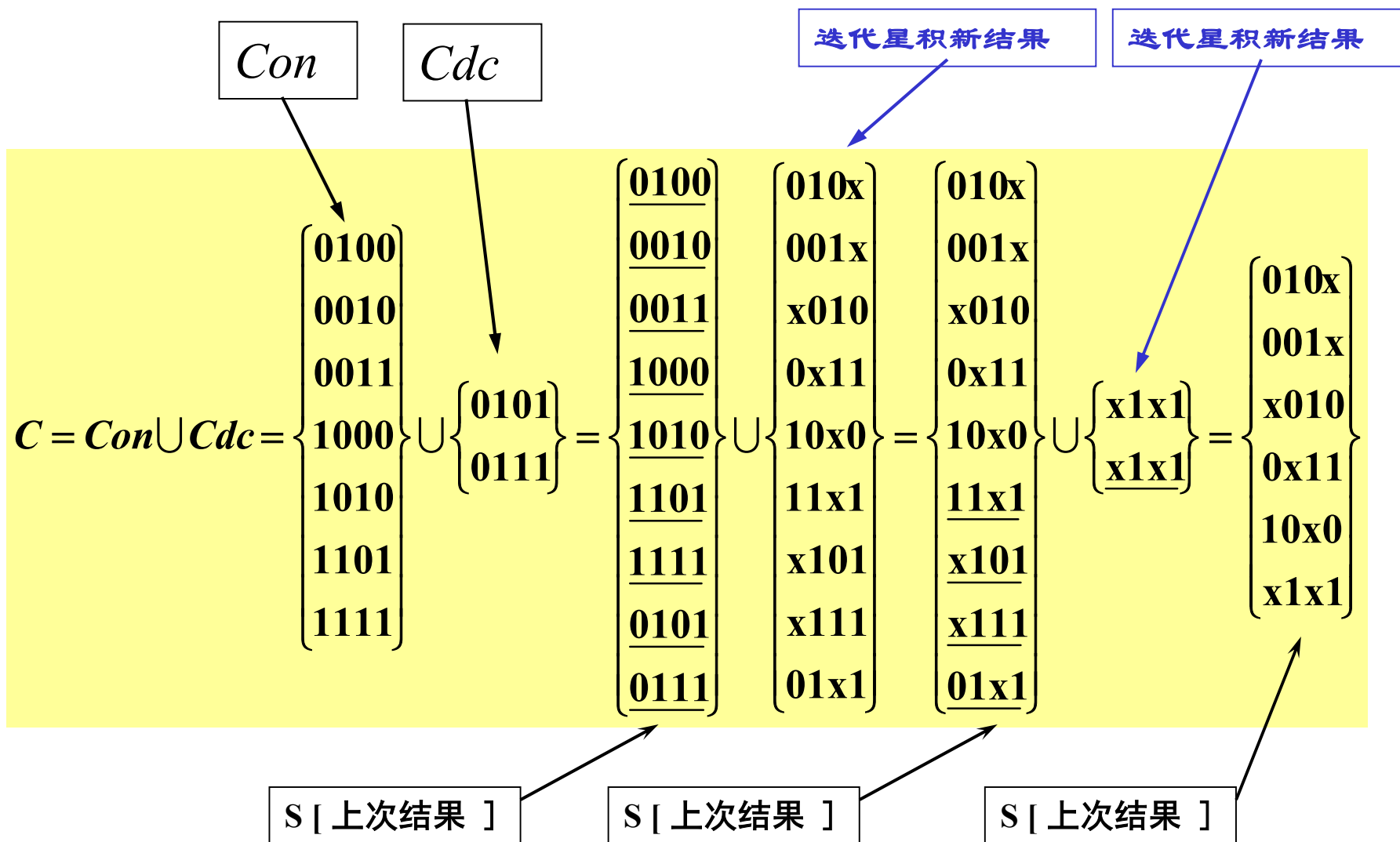
### ◆ 思考：

1.  $S$  的含义？

2. 每一个步骤的含义？



## 迭代星积法例





### 4.5.3 广义星积求质立方体集合Z

◆ 思路：把立方体的星积运算化为各变量之间的交运算（按位做）。

◆ 求星积实际是求交：

$$\begin{array}{rcl} & 11x & x1x \\ *) & 011 & \Rightarrow \cap) \quad x11 \\ \hline & q11 \Rightarrow x11 & \hline & x11 & \end{array}$$

如果有且仅有一个变量分别取值为 1 和 0



## 广义星积算法是对迭代星积算法的改进

前页的例子:	11x		x1x
	*) 011	=>	∩) x11
	q11 => x11		x11

- ◆ 每次沿着一个坐标(即变量)作星积。所以重复 $n$ 次( $n$ 为变量个数)即可。
- ◆ 把 $C$  中的立方体沿坐标  $i$  分为3类:
  - 变量 $x_i$ 取值为 0 的归入 $B_0$ ; => 并且将0改为X
  - 变量 $x_i$ 取值为 1 的归入 $B_1$ , => 并且将0改为X
  - 取值为 X 的留下不用。
- ◆ 若从 $B_0$ 中取任一立方体和 $B_1$ 中任一立方体作星积运算, 则在坐标 $i$ 上一定得q。
  - ➔ 可把星积运算化作交运算作!



## 广义星积求质立方体集合Z

### ◆ 步骤:

1.  $C = S[ \text{Con} \cup \text{Cdc} ]$

2.  $i$  从 1 到  $n$  重复以下步骤  $n$  次:

(1) 令  $B_0 = \emptyset, B_1 = \emptyset,$

(2) 扫描  $C$  中每一立方体:

若变量  $x_i = 0$ , 则令  $x_i = X$ , 其它变量取值不变, 使之加入  $B_0$ ;

若变量  $x_i = 1$ , 则令  $x_i = X$ , 其它变量取值不变, 使之加入  $B_1$ 。

(3) 令 
$$C = S \{ C \cup S( B_0 \cap B_1 ) \}$$

3.  $Z = \{ c \mid c \in C \text{ 且 } c \cap \text{Con} \neq \emptyset \}$



# 广义星积法例

$$C = \begin{Bmatrix} 0100 \\ 0010 \\ 0011 \\ 1000 \\ 1010 \\ 1101 \\ 1111 \\ 0101 \\ 0111 \\ x010 \\ x101 \\ x111 \end{Bmatrix} = \begin{Bmatrix} 0100 \\ 0011 \\ 1000 \\ x010 \\ x101 \\ x111 \\ 0x11 \end{Bmatrix} = \begin{Bmatrix} 0100 \\ 1000 \\ x010 \\ x101 \\ x111 \\ 0x11 \\ 10x0 \\ x1x1 \\ 01x1 \end{Bmatrix} = \begin{Bmatrix} 0100 \\ x010 \\ 0x11 \\ 10x0 \\ x1x1 \\ 010x \\ 001x \end{Bmatrix} = \begin{Bmatrix} x010 \\ 0x11 \\ 10x0 \\ x101 \\ 010x \\ 001x \end{Bmatrix}$$

$$B_0 = \begin{Bmatrix} x100 \\ x010 \\ x011 \\ x101 \\ x111 \end{Bmatrix} \quad \begin{Bmatrix} 0x11 \\ 1x00 \\ xx10 \end{Bmatrix} \quad \begin{Bmatrix} 01x0 \\ 10x0 \\ x1x1 \end{Bmatrix} \quad \begin{Bmatrix} 010x \\ x01x \\ 10xx \end{Bmatrix}$$

$$B_1 = \begin{Bmatrix} x000 \\ x010 \\ x101 \\ x111 \end{Bmatrix} \quad \begin{Bmatrix} 0x00 \\ xx01 \\ xx11 \end{Bmatrix} \quad \begin{Bmatrix} x0x0 \\ x1x1 \\ 0xx1 \end{Bmatrix} \quad \begin{Bmatrix} 0x1x \\ x1xx \end{Bmatrix}$$

$$B_0 \cap B_1 = :$$

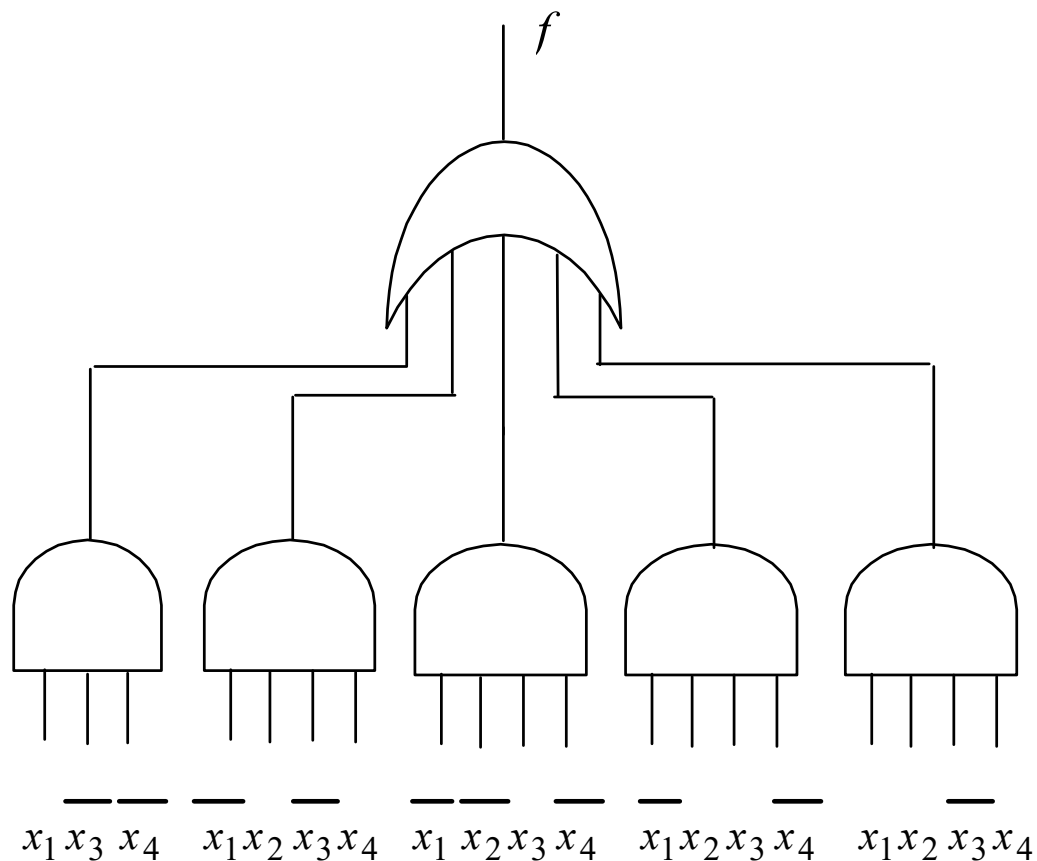
$$\begin{Bmatrix} X010 \\ X101 \\ X111 \end{Bmatrix} \quad \{0X11\} \quad \begin{Bmatrix} 10X0 \\ X1X1 \\ 01X1 \end{Bmatrix} \quad \begin{Bmatrix} 010X \\ 001X \end{Bmatrix}$$



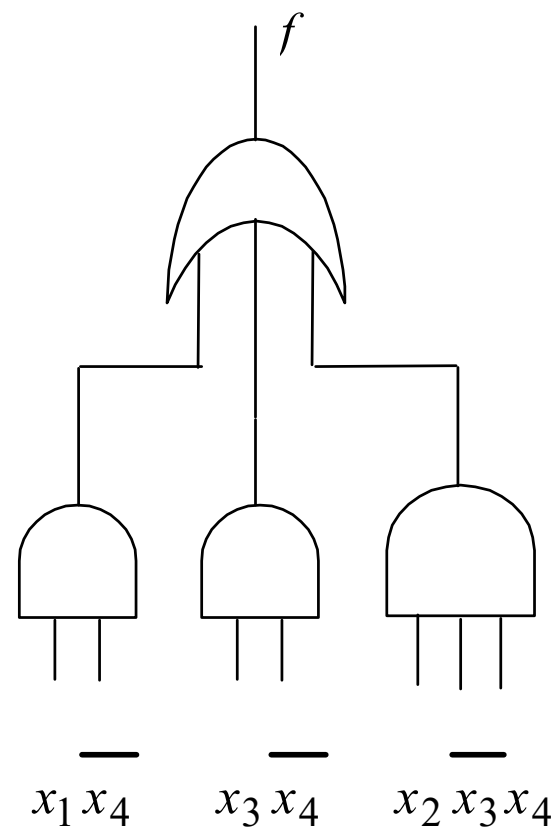
## 4.6

# 单输出函数的综合和优化

-- 举例



(a) 化简前



(b) 化简后



## 基本概念

### 最小化覆盖M:

- 1. 逻辑等价
- 2. 成本最低

### 无冗余覆盖N:

- 1. 逻辑等价
- 2. 接近最小化
  - $\forall n_i \in N, (n_i \# (N - n_i)) \cap \text{Con} \neq \emptyset$
  - $n_i$  是质立方体

---

• 逻辑等价:  $M \cap \text{Coff} = \emptyset, \quad \text{Con} \# M = \emptyset$

• 与门个数 = 立方体个数 **尽量少**;

$$CS(c) = \sum_{i=1}^k (n - r_i + 1)$$

• 输入端总数 **尽量少**:

k: 立方体个数    n: 输入变量个数

$r_i$ : 立方体  $c_i$  的维数 (X的个数)





## 基本算法

◆ 选拔法求最小化覆盖

◆ 收缩法求无冗余覆盖



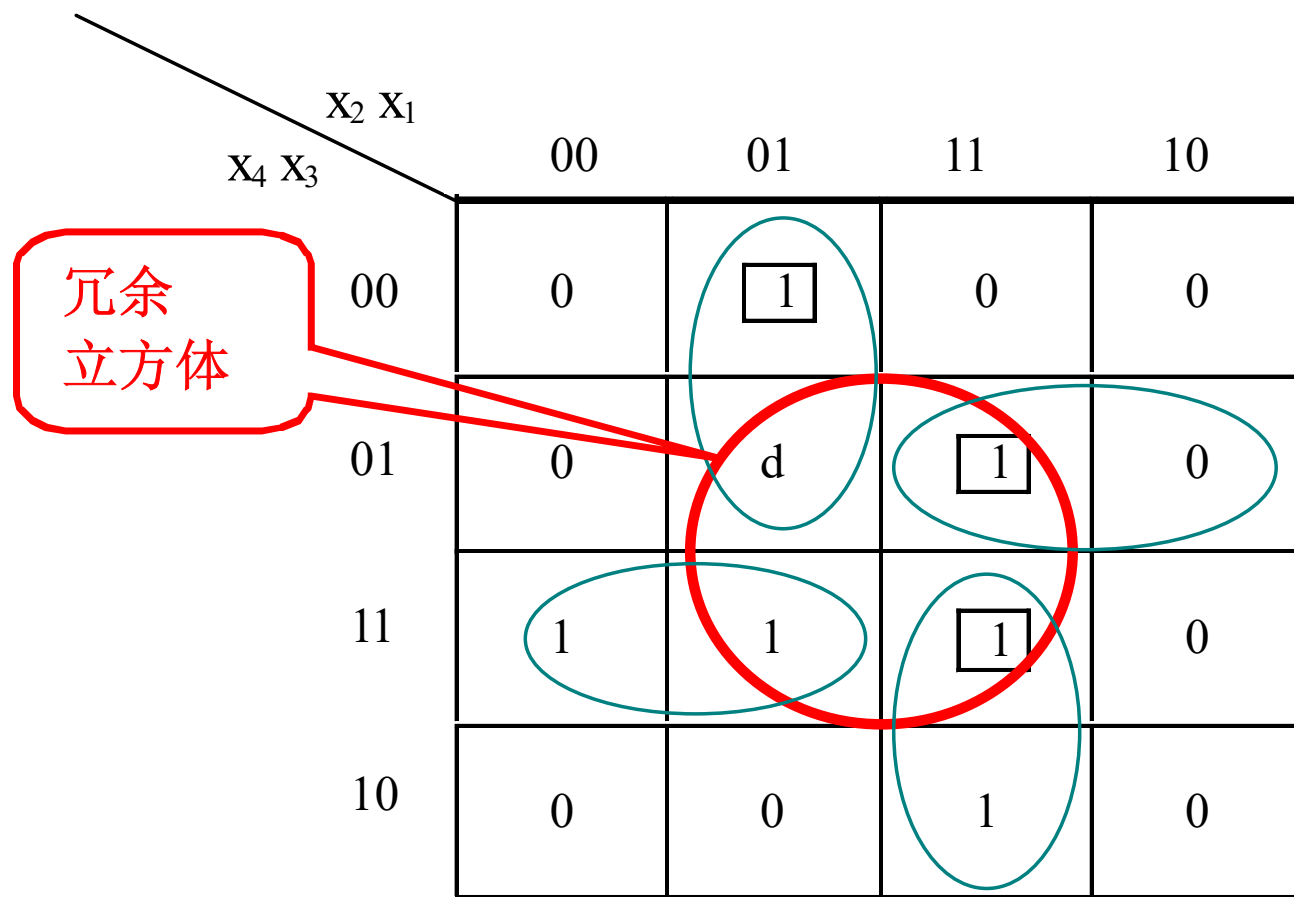
## 4.6.1 选拔法求最小化覆盖

- ◆ 初始覆盖：假定已化为质覆盖 $Z$ ；  
( $Z$ 中每一个元素都是质立方体)
- ◆ 选拔的过程： 反复进行
  - 删劣；
  - 选优；
  - 直到所有真值顶点全部被包含为止。



## 值得思考的问题

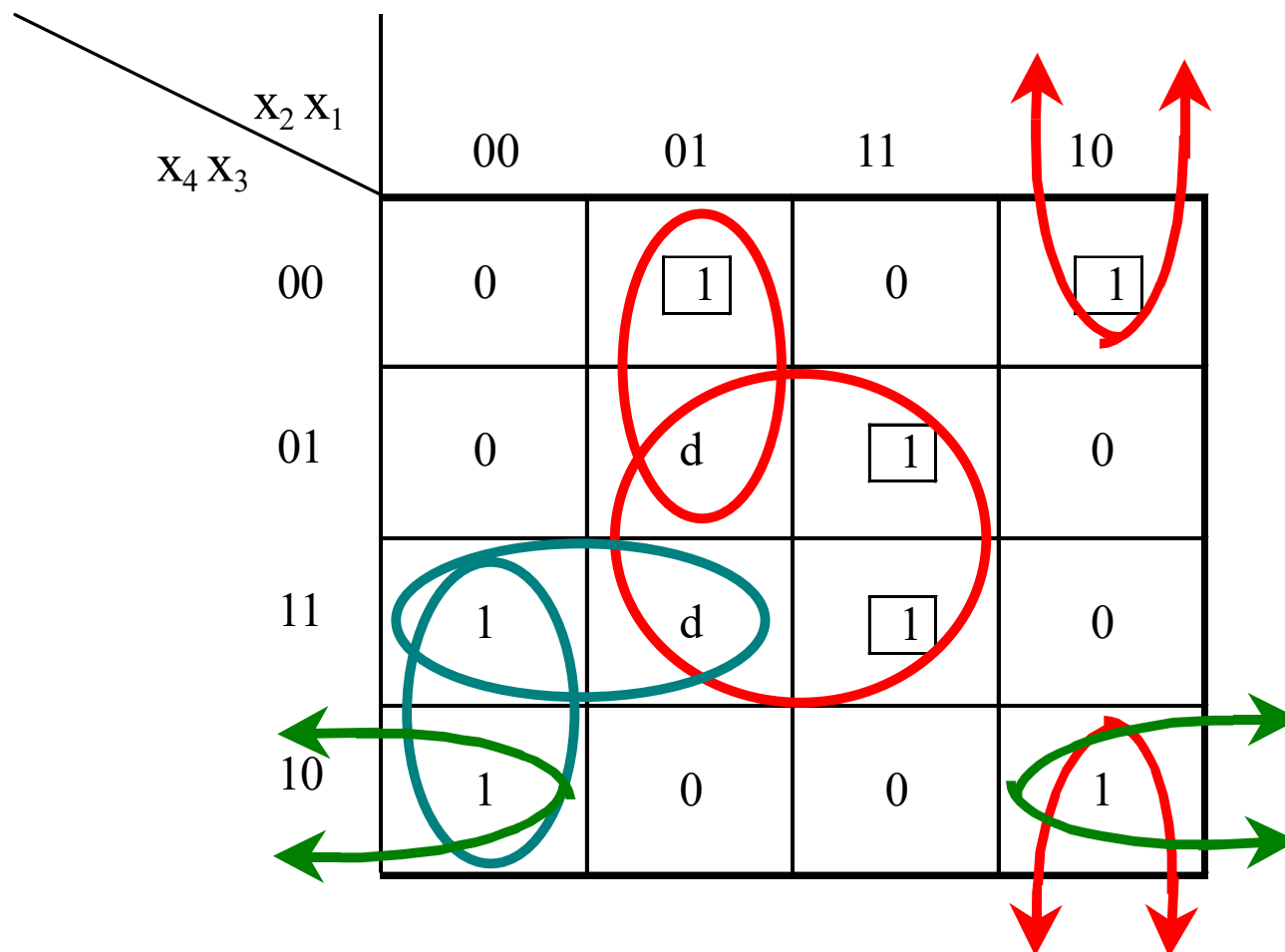
◆ 示意图:





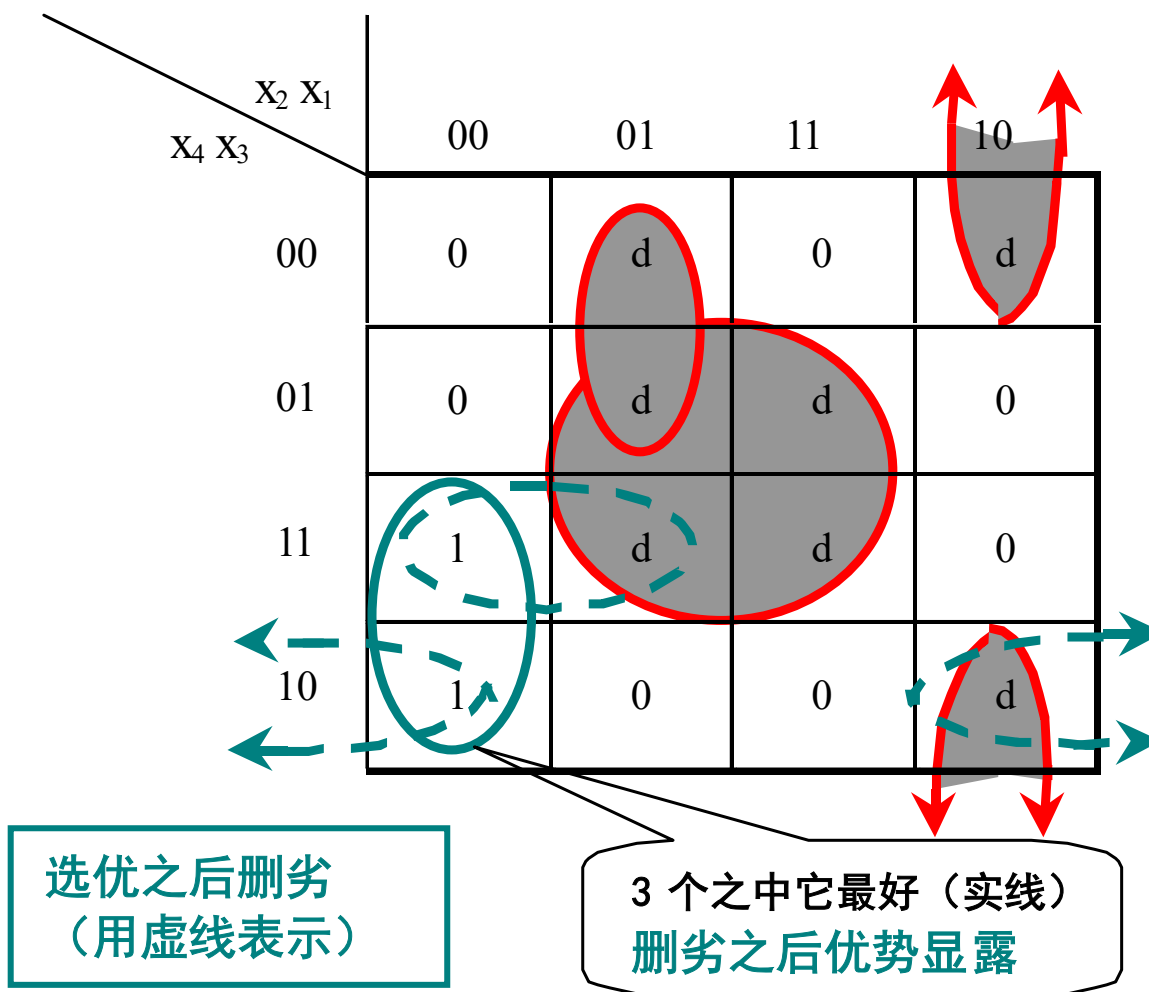
选优:

## 实例之一





## 实例之一（删劣→选优）





## 选拔法关键步骤：选优/删劣

Z动态变化记作SZ；

Con动态变化记作CC；

### ◆ 选优：选极值

对SZ的每一元素e, 若有  $(e \# (SZ - e)) \cap CC \neq \emptyset$ , 则e为极值。

-- 其中CC为待覆盖真值顶点集合

### ◆ 删劣：将成本大的无用立方体删除。

• 删劣算法：(小于运算)  $SZ = Lt(SZ)$

SZ中立方体两两比较，对 $a^i, a^j$

若 (1)  $CS(a^i) \geq CS(a^j)$  (成本)

(2)  $(a^i \# a^j) \cap CC = \emptyset$ , 则 $a^i$ 劣于 $a^j$ , 删去 $a^i$

( $a^i$  所贡献的顶点均包含于  $a^j$  )



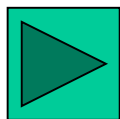
## 选拔法算法描述

1. 初值:  $CC=Con$ ,  $SZ=Z$ ,  $M=\emptyset$
2. 循环选优删劣:
  - (1) 删劣:  $SZ=lt(SZ)$
  - (2) 选优: 得极值集合  $E$ , 修改 $SZ$ ,  $CC$ ,  $M=M\cup E$
  - (3) 若 $CC=\emptyset$ , 则 $M$ 为最小化覆盖, 结束。
  - (4) 若 $E\neq\emptyset$ , 执行下一循环, 删劣/选优; 否则执行3.
3. 分支处理。选一个立方体  $p\in SZ$ , 分两支并行处理:
  - (1)  $M=M\cup\{p\}$ , 修改 $SZ$ ,  $CC$ , 转2(3)
  - (2)  $SZ=SZ-\{p\}$ , 转2(2)。选取其中总成本小者作为最终解。

$Z$ 动态变化记作 $SZ$  ;  
 $Con$ 动态变化记作 $CC$  ;  
选拔的结果记入 $M$



## 选拔法实例







## 选拔法求布尔函数的最小化

择优：  $M = \{0X0X1\}$

待覆盖 真值 顶点	0	0	0	0	0	1	1	1	1	1	1	选 中
质立方												
1) 10X01						√	√					
2) 0X0X1	√			√								√
3) 101XX							√	√	√			
4) 1X11X								√	√		√	
5) X01X0			√					√				
6) 11X10										√	√	
7) X1010					√					√		
8) 0X01X		√			√							
9) 00XX0		√	√									
10) 00X1X		√	√									
11) X011X			√					√	√			
12) 000XX	√	√										
13) X0001	√					√						
总 计	3	4	4	1	2	2	2	4	3	2	2	

### 择优结果：

. 真值顶点01001只被0X0X1所包含，称01001为特征真值顶点。**0X0X1**被选中，它既是极值项，又是必要质蕴涵项（必要质立方体）。

被0X0X1包含的真值顶点（00001和01001）应从表中删去。



## 删劣

待覆盖 真值 顶点	0	0	0	1	1	1	1	1	1	选中
	0	0	1	0	0	0	0	1	1	
	0	1	0	0	1	1	1	0	1	
	1	1	1	0	0	1	1	1	1	
	0	0	0	1	1	0	1	0	0	
质立方										
1) 10X01				√	√					
3) 101XX					√	√	√			
4) 1X11X						√	√		√	
5) X01X0		√				√				
6) 11X10								√	√	
7) X1010			√					√		
8) 0X01X	√		√							
9) 00XX0	√	√								
10) 00X1X	√	√								
11) X011X		√				√	√			
12) 000XX	√									
13) X0001				√						
总 计	4	4	2	2	2	4	3	2	2	

### 删劣结果:

- 9 与 10 相比: 10 被删除。
- 9 与 12 相比: 12 被删除。
- 1 与 13 相比: 13 被删除。



## 选优

选优:  $M = \begin{Bmatrix} 0X0X1 \\ 10X01 \end{Bmatrix}$

待覆盖 真值 顶点	0 0 0 1 0	0 0 1 1 0	0 1 0 1 0	1 0 0 0 1	1 0 1 0 1	1 0 1 1 0	1 0 1 1 1	1 1 0 1 0	1 1 1 1 0	选    中
质立方										
1) 10X01				√	√					√
3) 101XX					√	√	√			
4) 1X11X						√	√		√	
5) X01X0		√				√				
6) 11X10								√	√	
7) X1010			√					√		
8) 0X01X	√		√							
9) 00XX0	√	√								
11) X011X		√				√	√			
总 计	2	3	2	1	2	4	3	2	2	

选优结果:

1) 10001 是特征真值顶点, 10X01 取极值, 被选中。

注意, 它不是必要质立方体。

2) 待覆盖真值顶点 10001 及 10101 被 10X01 覆盖, 因而被删除。



## 删劣

待覆盖 真值 顶点  质立方	0	0	0	1	1	1	1	选 中
	0	0	1	0	0	1	1	
	0	1	0	1	1	0	1	
	1	1	1	1	1	1	1	
	0	0	0	0	1	0	0	
3) 101XX				v	v			
4) 1X11X				v	v		v	
5) X01X0		v		v				
6) 11X10						v	v	
7) X1010			v			v		
8) 0X01X	v		v					
9) 00XX0	v	v						
11) X011X		v		v	v			
总 计	2	3	2	4	3	2	2	

删劣结果:

- 1) 4 与 3 相比, 3 被删除。
- 2) 5 与 11 相比, 5 被删除。



## 进入循环状态

待覆盖 真值 顶点 质立方	0 0 0 1 0	0 0 1 1 0	0 1 0 1 0	1 0 1 1 0	1 0 1 1 1	1 1 0 1 0	1 1 1 1 0	选   中
4) 1X11X				√	√		√	
6) 11X10						√	√	
7) X1010			√			√		
8) 0X01X	√		√					
9) 00XX0	√	√						
11) X011X		√		√	√			
总 计	2	2	2	2	2	2	2	

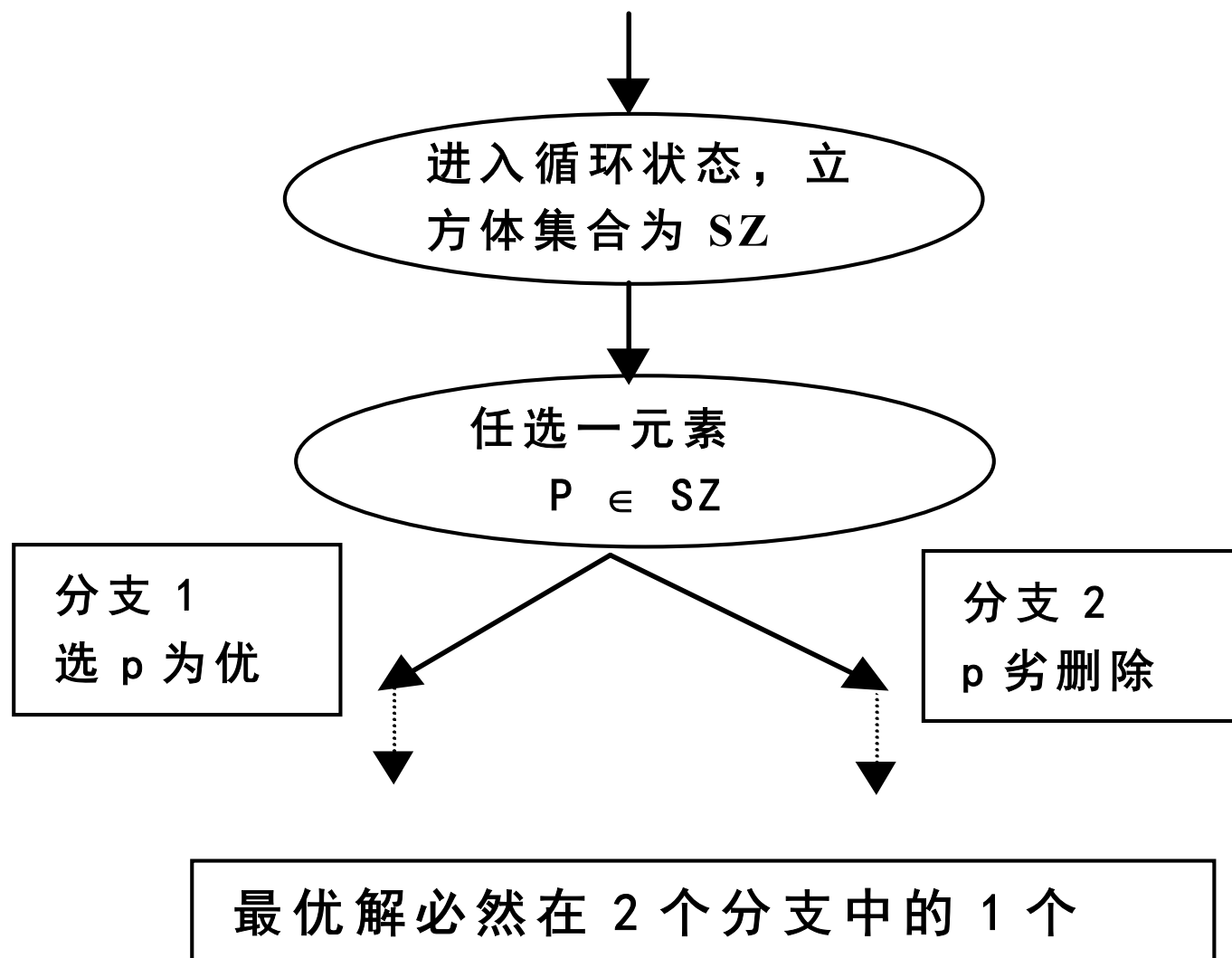
现在进入循环状态，即：

- 删劣删不掉；
- 选优选不出；

本例选用分支法打破循环状态



## 分支法示意图





## 分支法 第一分支：

$$\text{选优结果: } M = \begin{Bmatrix} 0 & X & 0 & X & 1 \\ 1 & 0 & X & 0 & 1 \\ 1 & X & 1 & 1 & X \end{Bmatrix}$$

以前的结果：

待覆盖 真值 顶点	0	0	0	1	1	1	1	选
质立方	0	0	0	0	1	0	0	中
4) 1X11X				v	v		v	
6) 11X10						v	v	
7) X1010			v			v		
8) 0X01X	v		v					
9) 00XX0	v	v						
11) X011X		v		v	v			
总 计	2	2	2	2	2	2	2	

把**1X11X**当作优项，被选中。⇒

待覆盖 真值 顶点	0	0	0	1	1	1	1	选
质立方	0	0	0	0	1	0	0	中
4) 1X11X				v	v		v	v
6) 11X10						v	v	
7) X1010			v			v		
8) 0X01X	v		v					
9) 00XX0	v	v						
11) X011X		v		v	v			
总 计	2	2	2	2	2	2	2	



## 第一分支后续操作：删劣

<div style="display: inline-block; transform: rotate(-45deg); text-align: center;"> 待覆盖 真值 顶点  质立方 </div>	0	0	0	1	选  中
	0	0	1	1	
	0	1	0	0	
	1	1	1	1	
	0	0	0	0	
6) 11X10				√	
7) X1010			√	√	
8) 0X01X	√		√		
9) 00XX0	√	√			
11) X011X		√			
总 计	2	2	2	2	

删劣结果：

- 1) 6 与 7 相比，6 被删除。
- 2) 9 与 11 相比，11 被删除。





## 第一分支后续操作：选优

待覆盖 真值 顶点	0	0	0	1	选 中
	0	0	1	1	
	0	1	0	0	
	1	1	1	1	
	0	0	0	0	
质立方					
7) X1010			√	√	√
8) 0X01X	√		√		
9) 00XX0	√	√			√
总计	2	1	2	1	

选优结果：

1) **X1010** 取极值，被选中。被它覆盖的真值顶点 11010 及 01010 被删除。

2) **00XX0** 取极值，被选中。被它覆盖的真值顶点 00110 及 00010 被删除。

至此，待覆盖真值顶点全部被覆盖，任务完成。

第一分支的最终结果为：

$\left\{ \begin{array}{l} 0X0X1 \\ 10X01 \\ 1X11X \\ X1010 \\ 00XX0 \end{array} \right\}$



## 第一分支后续操作：选优

待覆盖 真值 顶点	0	0	0	1	选中
	0	0	1	1	
	0	1	0	0	
	1	1	1	1	
	0	0	0	0	
质立方					
7) X1010			v	v	v
8) 0X01X	v		v		
9) 00XX0	v	v			v
总 计	2	1	2	1	

最终结果：

$$M = \left\{ \begin{array}{l} 0 X 0 X 1 \\ 1 0 X 0 1 \\ 1 X 1 1 X \\ X 1 0 1 0 \\ 0 0 X X 0 \end{array} \right\}$$

选优结果：

- 1) **X1010** 取极值，被选中。被它覆盖的真值顶点 11010 及 01010 被删除。
- 2) **00XX0** 取极值，被选中。被它覆盖的真值顶点 00110 及 00010 被删除。



## 分支法 第二分支：

⇒ 把1X11X当作劣项，被删除。 ⇒

待覆盖 真值 顶点 质立方	0 0 0 1 0	0 0 1 1 0	0 1 0 1 0	1 0 1 1 0	1 0 1 1 1	1 1 0 1 0	1 1 1 1 0	选中
6) 11X10						√	√	
7) X1010			√			√		
8) 0X01X	√		√					
9) 00XX0	√	√						
11) X011X		√		√	√			
总 计	2	2	2	1	1	2	1	

由于1X11X 被删除，循环状态被打破。  
极值项有可能出现，进入下一轮选优。

以前的结果：

待覆盖 真值 顶点 质立方	0 0 0 1 0	0 0 1 1 0	0 1 0 1 0	1 0 1 1 0	1 0 1 1 1	1 1 0 1 0	1 1 1 1 0	选中
4) 1X11X				√	√		√	
6) 11X10						√	√	
7) X1010			√			√		
8) 0X01X	√		√					
9) 00XX0	√	√						
11) X011X		√		√	√			
总 计	2	2	2	2	2	2	2	

$$M = \left\{ \begin{array}{l} 0X0X1 \\ 10X01 \end{array} \right\}$$



## 第二分支后续操作：选优

待覆盖 真值 顶点	0	0	0	1	1	1	1	选 中
	0	0	1	0	0	1	1	
	0	1	0	1	1	0	1	
	1	1	1	1	1	1	1	
	0	0	0	0	1	0	0	
	0	0	0	0	1	0	0	
质立方								
6) 11X10						√	√	
7) X1010			√			√		
8) 0X01X	√		√					
9) 00XX0	√	√						
11) X011X		√		√	√			
总 计	2	2	2	1	1	2	1	

$$M = \left\{ \begin{array}{l} 0X0X1 \\ 10X01 \\ 11X10 \\ X011X \end{array} \right\}$$

选优结果:

- 1) 11X10 取极值, 被选中。被它覆盖的真值顶点 11010 及 11110 被删除。
- 2) X011X 取极值, 被选中。被它覆盖的真值顶点 00110、10110 及 10111 被删除。



## 第二分支后续操作：删劣 → 选优

待覆盖 真值 顶点	0	0	选
	0	1	
	0	0	
	1	1	中
	0	0	
质立方			
7) X1010		√	
8) 0X01X	√	√	√
9) 00XX0	√		
总 计	2	2	

删劣结果：

1) 7 与 8 相比，7 被删除。

2) 9 与 8 相比，9 被删除。

至此，唯一可选的质立方体是：0X01X

最终解：

$$M = \left\{ \begin{array}{l} 0X0X1 \\ 10X01 \\ 11X10 \\ X011X \\ 0X01X \end{array} \right\}$$

◆ 最后，在2个分支的解中择一优者；

◆ 对于本例，2个解的造价相同，可任选其一。



## 对本例的评述

- ◆ 本例中CC以真值顶点集合形式表示，有利于表述；
- ◆ 选拔法求最小化覆盖算法（p.86，复制于右）描述中对真值项集合CC无此限制。
  - CC是立方体集合；
  - CC包含所有待覆盖的真值顶点。

1. 初值：  $CC=Con, SZ=Z, M=\emptyset$
2. 循环选优删劣：
  - (1) 删劣：  $SZ=lt(SZ)$
  - (2) 选优：得极值集合  $E$ , 修改  $SZ, CC, M=M\cup E$
  - (3) 若  $CC=\emptyset$ , 则  $M$  为最小化覆盖，结束。
  - (4) 若  $E\neq\emptyset$ , 执行下一循环，删劣/选优；否则执行3；
3. 分支处理。选一个立方体  $p\in SZ$ , 分两支并行处理：
  - (1)  $M=M\cup\{p\}$ , 修改  $SZ, CC$ , 转2(3)
  - (2)  $SZ=SZ-\{p\}$ , 转2(2)。选取其中总成本小者作为最终解。



## 4.6.2 收缩法求无冗余覆盖

### 基本思路：

(1) 对每一个立方体求余面 (→ 吸收)，得质立方体集合  $N$ 。

$$N \cap \mathbf{Coff} = \emptyset; \quad \mathbf{Con} \# N = \emptyset;$$

- $n_i \in N$  是质立方体；
- $N$  中元素个数尽可能少，且单调下降；

(2) 删去  $N$  中冗余立方体 → 无冗余

$$\forall n_i \in N, (n_i \# (N - n_i)) \cap \mathbf{Con} \neq \emptyset$$



## 收缩法算法描述

1.  $N = S[Con]$  (思考：是否需要考虑Cdc?)

吸收运算

2. 生成质立方体：

对N中每一元素e

{ 对  $i = 1$  to  $n$  ( $n$ 为输入变量个数)

令  $p = e$ , 设  $p = x_1 x_2 \dots x_n$ , 若  $x_i \neq X$ , 则令  $x_i = X$

若  $p \cap Coff = \emptyset$ , (不包含假值顶点), 则令  $e = p$ ,

并删去被  $e$  包含的立方体 (吸收)。

}

3. 删去N中冗余立方体： (本步骤的必要性?)

对每一立方体  $e \in N$ , 若  $(e \# (N - e)) \cap Con = \emptyset$ , 则删去  $e$ 。





# 收缩法例1

$$\text{Con} = \begin{Bmatrix} 0010 \\ 0011 \\ 0100 \\ 1000 \\ 1010 \\ 1101 \\ 1111 \end{Bmatrix} \quad \text{Cdc} = \begin{Bmatrix} 0101 \\ 0111 \end{Bmatrix} \quad \text{Coff} = \begin{Bmatrix} 0000 \\ 0001 \\ 0110 \\ 1001 \\ 1011 \\ 1100 \\ 1110 \end{Bmatrix}$$

- 标有双下划线者表示被选定待操作的立方体；
- 标有双上划线者表示所求余面；
- 标有单下划线者表示执行收缩算法应被删除的立方体；

$$N = \begin{Bmatrix} \underline{\underline{0010}} \\ 0011 \\ 0100 \\ 1000 \\ \underline{1010} \\ 1101 \\ 1111 \end{Bmatrix} = \begin{Bmatrix} \overline{\overline{x010}} \\ \underline{\underline{0011}} \\ 0100 \\ 1000 \\ 1101 \\ 1111 \end{Bmatrix} = \begin{Bmatrix} x010 \\ \underline{\underline{0x11}} \\ 0100 \\ 1000 \\ 1101 \\ 1111 \end{Bmatrix} = \begin{Bmatrix} x010 \\ 0x11 \\ \underline{\underline{010x}} \\ 1000 \\ 1101 \\ 1111 \end{Bmatrix} = \begin{Bmatrix} x010 \\ 0x11 \\ 010x \\ \underline{\underline{10x0}} \\ \underline{1101} \\ \underline{1111} \end{Bmatrix} = \begin{Bmatrix} x010 \\ 0x11 \\ 010x \\ 10x0 \\ \underline{\underline{x1x1}} \end{Bmatrix}$$



## 收缩法例2

$$\text{Con} = \begin{Bmatrix} 0101 \\ 0111 \\ 1100 \\ 1101 \\ 1011 \\ 0110 \\ 0001 \end{Bmatrix} \quad \text{Cdc} = \{1111\} \quad \text{Coff} = \begin{Bmatrix} 0000 \\ 0100 \\ 0010 \\ 0011 \\ 1000 \\ 1001 \\ 1010 \\ 1110 \end{Bmatrix}$$

		$X_4 \ X_3$		$X_2 \ X_1$	
		00	01	11	10
00		0	0	1	0
01		1	1	1	0
10		0	1	d	1
11		0	1	0	0

冗余立方体  
删除！

$$N = \begin{Bmatrix} \overline{0101} \\ \overline{0111} \\ 1100 \\ \overline{1101} \\ 1011 \\ 0110 \\ 0001 \end{Bmatrix} = \begin{Bmatrix} \overline{\overline{x1x1}} \\ \overline{\overline{1100}} \\ 1011 \\ 0110 \\ 0001 \end{Bmatrix} = \begin{Bmatrix} \overline{\overline{x1x1}} \\ \overline{\overline{110x}} \\ \overline{\overline{1011}} \\ 0110 \\ 0001 \end{Bmatrix} = \begin{Bmatrix} \overline{\overline{x1x1}} \\ \overline{\overline{110x}} \\ \overline{\overline{1x11}} \\ \overline{\overline{0110}} \\ \overline{\overline{0001}} \end{Bmatrix} = \begin{Bmatrix} \overline{\overline{x1x1}} \\ \overline{\overline{110x}} \\ \overline{\overline{1x11}} \\ \overline{\overline{011x}} \\ \overline{\overline{0001}} \end{Bmatrix} = \begin{Bmatrix} \overline{\overline{x1x1}} \\ \overline{\overline{110x}} \\ \overline{\overline{1x11}} \\ \overline{\overline{011x}} \\ \overline{\overline{0x01}} \end{Bmatrix} = \begin{Bmatrix} 110x \\ 1x11 \\ 011x \\ 0x01 \end{Bmatrix}$$



## 4.7 多输出函数的综合

◆ 复习：多输出函数  $\Leftrightarrow$  单输出函数：

例如，右边是多输出函数表示，  
下面是其单输出函数表示：

$$Con^1 = \left\{ \begin{array}{l} 000 \\ 001 \\ 111 \end{array} \right\} \quad Coff^1 = \left\{ \begin{array}{l} 010 \\ 011 \\ 10X \end{array} \right\}$$

$$Con^2 = \left\{ \begin{array}{l} 001 \\ 011 \\ 10X \end{array} \right\} \quad Coff^2 = \left\{ \begin{array}{l} 000 \\ 010 \\ 111 \end{array} \right\}$$

<u>输 入</u>			<u>输 出</u>	
$x_1$	$x_2$	$x_3$	$y^1$	$y^2$
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	d	d
1	0	x	0	1
1	1	1	1	0



## 4.7 多输出函数的综合

### ◆成本计算公式:

$$CS(C) = \sum_{i=1}^k (n - r_i + t_i)$$

- $k$ : 立方体个数 (7) ;
- $n$ : 输入变量个数 (3) ;
- $r_i$ : 输入部分维数( $X$ 个数);
- $t_i$ : 输出部分取值为1的个数。

### ◆ 由多输出函数的初始描述C求Con, Coff:

Con: 输出部分 0 变为u,做吸收;

Coff: 输出部分1变为u, 0变为1, 做吸收;

### ◆ 各种运算按多输出运算规则进行;

### ◆ 得到结果后删除多余连线;

输 入			输 出	
$x_1$	$x_2$	$x_3$	$y^1$	$y^2$
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	d	d
1	0	x	0	1
1	1	1	1	0



## 4.7.1 收缩算法求无冗余覆盖

1.  $N=S[Con] \rightarrow$  余面法做质立方体变换  $\rightarrow$  吸收。

吸收运算

- \* 将立方体扩展；
- \* 扩展后的立方体和Coff相交为空。

2. 重复执行下列操作：

(1) 删去冗余立方体：

对每一立方体  $e \in N$ , 若  $(e \# (N-e)) \cap Con = \emptyset$ , 则删去  $e$

(2) 删去多余连线，若无多余连线，结束。

(3) 在删除多余连线中变换的立方体进行质立方体变换。

◆ 结束条件：

(1) 无多余连线

(2) 最后步骤的  $N$  已出现重复



## 多输出收缩法例

原始描述为C； 由C导出Con：

Con: 输出部分 0 变为u → 做吸收。

$$C = \left\{ \begin{array}{l|l} 000 & 110 \\ 001 & 01u \\ 010 & 111 \\ 011 & 101 \\ 100 & 101 \\ 101 & 011 \\ 110 & 001 \\ 111 & 000 \end{array} \right\} \Rightarrow \text{Con} = \left\{ \begin{array}{l|l} 000 & 11u \\ 001 & u1u \\ 010 & 111 \\ 011 & 1u1 \\ 100 & 1u1 \\ 101 & u11 \\ 110 & uu1 \end{array} \right\}$$



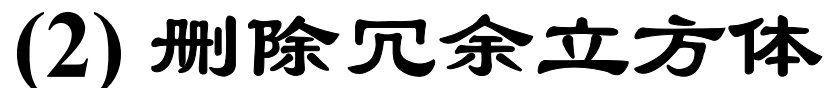
# (1)余面法求质立方体

$N = S\{\text{Con}\} \Rightarrow$  求余面：

$$C = \left\{ \begin{array}{l} 000 | 110 \\ 001 | 01u \\ 010 | 111 \\ 011 | 101 \\ 100 | 101 \\ 101 | 011 \\ 110 | 001 \\ 111 | 000 \end{array} \right\}$$

$$N = \left\{ \begin{array}{l} \underline{\underline{000 | 11u}} \\ 001 | u1u \\ 010 | 111 \\ 011 | 1u1 \\ 100 | 1u1 \\ 101 | u11 \\ 110 | uu1 \end{array} \right\} = \left\{ \begin{array}{l} \overline{\overline{0x0 | 11u}} \\ \overline{\overline{001 | u1u}} \\ 010 | 111 \\ 011 | 1u1 \\ 100 | 1u1 \\ 101 | u11 \\ 110 | uu1 \end{array} \right\} = \left\{ \begin{array}{l} 0x0 | 11u \\ \underline{\underline{x01 | u11}} \\ 010 | 111 \\ 011 | 1u1 \\ \underline{\underline{100 | 1u1}} \\ 110 | uu1 \end{array} \right\} = \left\{ \begin{array}{l} 0x0 | 11u \\ x01 | u11 \\ 010 | 111 \\ \underline{\underline{01x | 1u1}} \\ 100 | 1u1 \\ \underline{\underline{110 | uu1}} \end{array} \right\} = \left\{ \begin{array}{l} 0x0 | 11u \\ x01 | u11 \\ 010 | 111 \\ 01x | 1u1 \\ 100 | 1u1 \\ \underline{\underline{1x0 | uu1}} \end{array} \right\}$$

标有双下划线者表示被选定待操作的立方体；  
 标有双上划线者表示所求余面；  
 标有单下划线者表示执行收缩算法应被删除的立方体；



$$N = \left\{ \begin{array}{l} \overline{0x0} | 11u \\ \overline{x01} | u11 \\ \overline{010} | 111 \\ \overline{01x} | 1u1 \\ \overline{100} | 1u1 \\ \overline{1x0} | uu1 \end{array} \right\} = \left\{ \begin{array}{l} 0x0 | 11u \\ x01 | u11 \\ 01x | 1u1 \\ 100 | 1u1 \\ 1x0 | uu1 \end{array} \right\}$$

The diagram shows a logic circuit where three functions  $f^1, f^2, f^3$  are connected to a common bus. The bus feeds into six AND gates. The third AND gate, which takes inputs  $x_1, x_2, x_3$ , is highlighted in red. A red arrow points from this gate to the set  $N$  on the left.

**{010|111}为冗余立方体，删除后造价降低。**





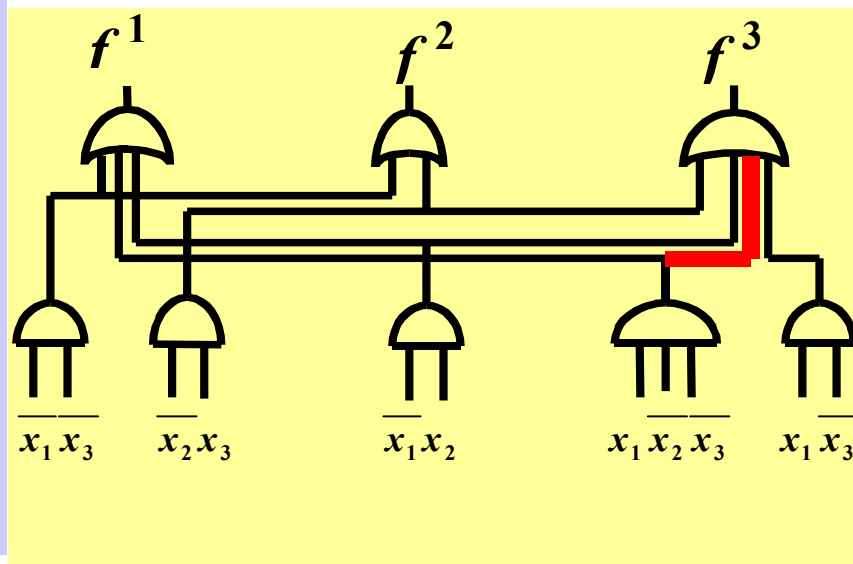
### (3) 删多余连线

- ◆ 对输出第j位（本例是 $f^3$ ），若某一立方体为冗余立方体，则将该位1改为u。

本例：  $100 | 1u1 \rightarrow 100 | 1uu$  （理由见下页）

修改后原理图中红线被删除，造价降低。

$$N = \left\{ \begin{array}{l} 0x0 | 11u \\ x01 | u11 \\ 01x | 1u1 \\ \underline{\underline{100 | 1u1}} \\ 1x0 | uu1 \end{array} \right\} = \left\{ \begin{array}{l} 0x0 | 11u \\ x01 | u11 \\ 01x | 1u1 \\ \underline{\underline{100 | 1uu}} \\ 1x0 | uu1 \end{array} \right\}$$





为什么  $100 \mid 1u1 \rightarrow 100 \mid 1uu$  ?

- ◆ 对 $f^3$ 而言：立方体 $100$ 是冗余立方体（被 $1X0$ 包含）

$$N = \left\{ \begin{array}{l} 0x0 \mid 11u \\ x01 \mid u11 \\ 01x \mid 1u1 \\ \underline{\underline{100 \mid 1u1}} \\ \underline{1x0 \mid uu1} \end{array} \right\} \quad Con^3 = \left\{ \begin{array}{l} x01 \\ 01x \\ \underline{\underline{100}} \\ \underline{1x0} \end{array} \right\} = \left\{ \begin{array}{l} x01 \\ 01x \\ 1x0 \end{array} \right\}$$

- 思考：把质立方体化为非质立方体成本反而降低！  
➔ 存在进一步降低成本的可能性。

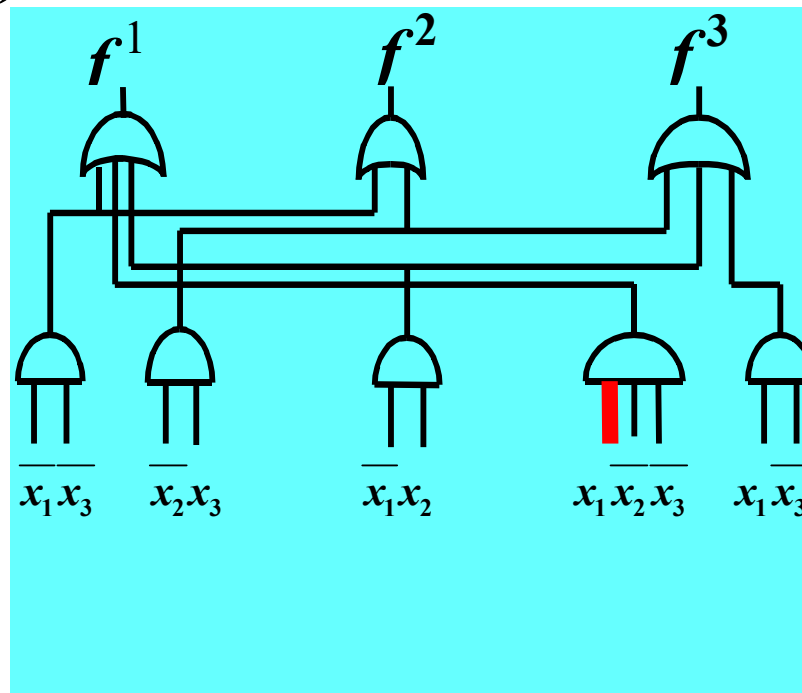


## (4) 再变成质立方体

◆ 输入部分1或0变x, 形成质立方体

$(100|1uu \Rightarrow \mathbf{x00|1uu}) \Rightarrow$  造价降低

$$N = \left\{ \begin{array}{l} 0x0|11u \\ x01|u11 \\ 01x|1u1 \\ \underline{\underline{100|1uu}} \\ 1x0|uu1 \end{array} \right\} = \left\{ \begin{array}{l} 0x0|11u \\ x01|u11 \\ 01x|1u1 \\ \underline{\underline{x00|1uu}} \\ 1x0|uu1 \end{array} \right\}$$

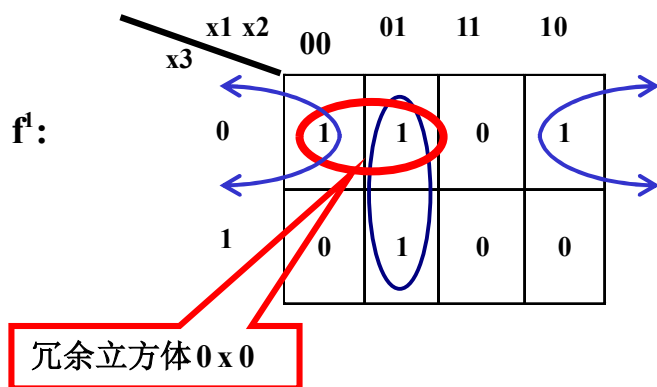
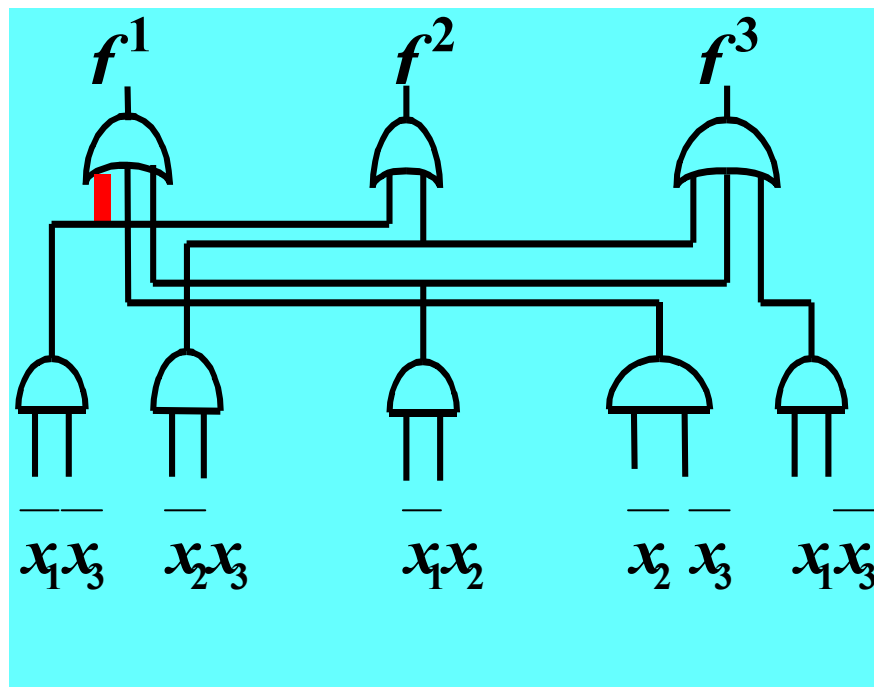


标有双下划线者表示被选定待操作的立方体；  
标有双上划线者表示所求余面；



## (5) 再删除多余连线

$$N = \left\{ \begin{array}{l} \overline{0x0|11u} \\ \overline{x01|u11} \\ \overline{01x|1u1} \\ \overline{x00|1uu} \\ \overline{1x0|uu1} \end{array} \right\} = \left\{ \begin{array}{l} \overline{0x0|u1u} \\ \overline{x01|u11} \\ \overline{01x|1u1} \\ \overline{x00|1uu} \\ \overline{1x0|uu1} \end{array} \right\}$$



对  $f^1$  而言：立方体  $0x0$  是冗余立方体，因而导致：

$$0x0|11u \rightarrow 0x0|u1u$$

原理图中红色连线被删除。

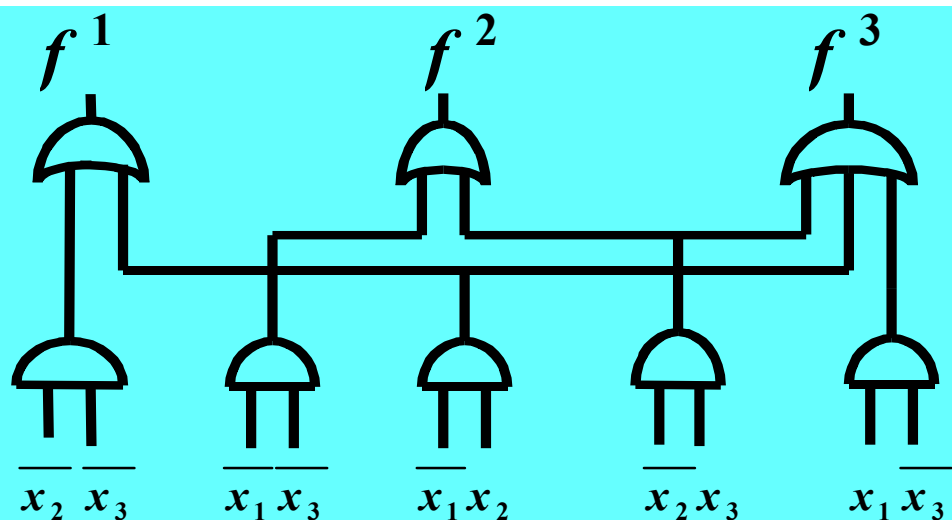


## (6) 结果

$$N = \left\{ \begin{array}{l} 0x0 \mid u1u \\ x01 \mid u11 \\ 01x \mid 1u1 \\ x00 \mid 1uu \\ 1x0 \mid uu1 \end{array} \right\}$$

与精心手工综合结果相同。

提示：只是在变量个数很少的情况下，手工设计才可行。



$$\begin{array}{c|cccc} & \text{00} & \text{01} & \text{11} & \text{10} \\ \hline x_3 \backslash F^1 & & & & \\ 0 & 1 & 1 & & 1 \\ 1 & & 1 & & \end{array}$$

$$\begin{array}{c|cccc} & \text{00} & \text{01} & \text{11} & \text{10} \\ \hline x_3 \backslash F^2 & & & & \\ 0 & 1 & 1 & & \\ 1 & 1 & & & 1 \end{array}$$

$$\begin{array}{c|cccc} & \text{00} & \text{01} & \text{11} & \text{10} \\ \hline x_3 \backslash F^3 & & & & \\ 0 & & 1 & 1 & 1 \\ 1 & 1 & 1 & & 1 \end{array}$$



## 4.7.2 选拔法求最小化覆盖

### ◆ 算法：

1. 求质立方体集合Z；

2. 选优、删劣

（概念与单输出相同）；

3. 删除多余连线；

- 删除多余连线过程与收缩法相同。

### ◆ 若出现循环状态，进行分枝处理

- 每个分支都要作删除多余连线的操作；

输 入			输 出	
$x_1$	$x_2$	$x_3$	$y^1$	$y^2$
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	d	d
1	0	x	0	1
1	1	1	1	0



## 如何求质立方体集合 $Z$ (多输出)

### 方法1：锐积求质立方体：

- ◆ 从原始描述覆盖 $C$ 中分离出 $Coff$ ,
  - 把输出部分取值为1者一律改为 $u$ ;
  - 取值为0者一律改为1;
  - 删去输出部分取值全为 $u$ 的立方体。
- ◆ 令全立方体 $Q_n$ 为,

$$Q_n = XXX...X...X \mid 11...1...1$$

- ◆  $Z = Q_n \# Coff$ 
  - 每一步锐积运算之后都要作吸收运算

输 入			输 出	
$x_1$	$x_2$	$x_3$	$y^1$	$y^2$
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	d	d
1	0	x	0	1
1	1	1	1	0



## 如何求质立方体集合 $Z$ (多输出)

### 方法2: 多输出 $\rightarrow$ 单输出 $\rightarrow$ 多输出

◆  $j$ 从1到 $m$ 重复以下步骤 $m$ 次:

1) 用阵列分离法从初始覆盖 $C$ 中函数 $y^j$ 的 $Con^j$ 和 $Coff^j$ ;

(2) 形成 $y^j$ 的全部质立方体集合 $Z^j$ ;

◆ 用阵列合并法把 $Z^1 Z^2 \dots Z^j \dots Z^m$ 合并成 $Z$ ;  
重复以下步骤:

(1) 令  $E = \emptyset$  ;

(2) 令  $e = e_i \cap e_j$  ( $e_i, e_j \in Z$ , 且  $i \neq j$ ) ;

若 $e$ 不是 $\{Z \cup E\}$ 中任何元素的面, 则  $E = E \cup e$

(3) 若 $E \neq \emptyset$ , 则令 $Z = S(Z \cup E)$ , 转入(1)。否则, 结束运算。

输 入			输 出	
$x_1$	$x_2$	$x_3$	$y^1$	$y^2$
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	d	d
1	0	x	0	1
1	1	1	1	0





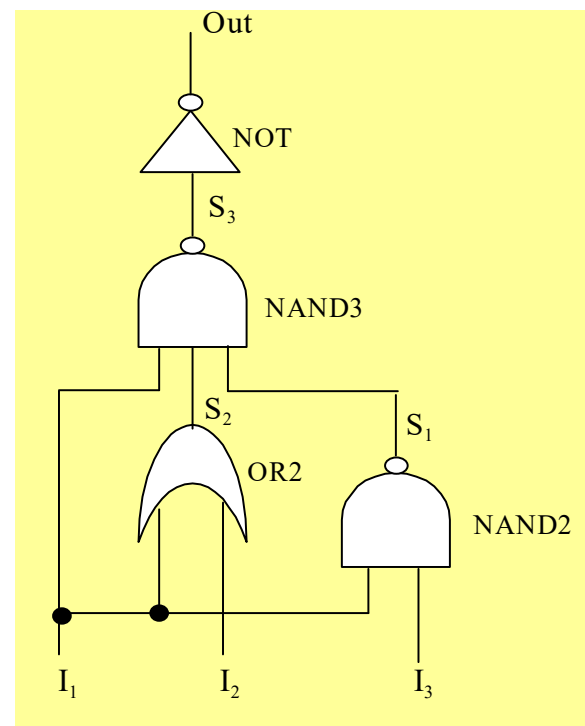
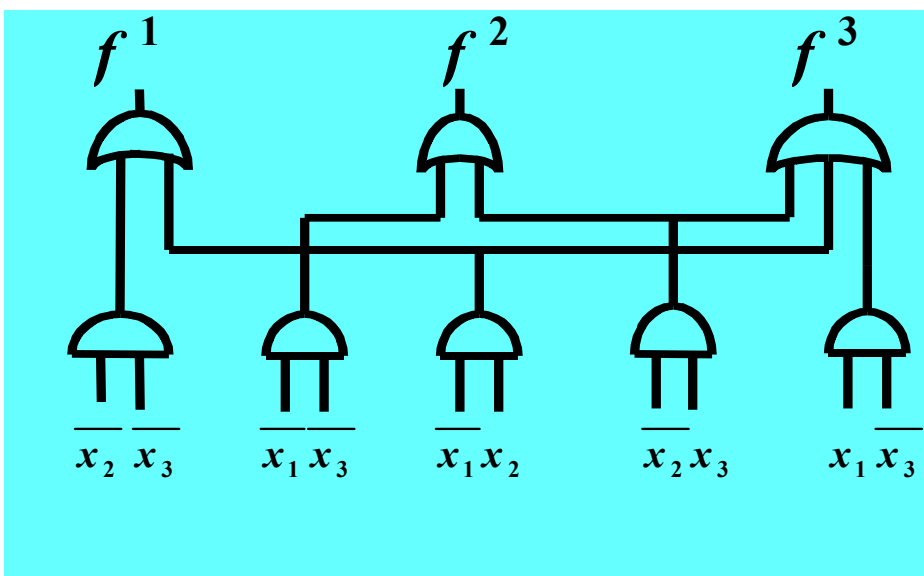
### 4.7.3 判别质蕴涵项的E算法

- ◆ 在无需求出 $Z$ 的情况下就可以断定一个质蕴涵项  $p$  是否是必要质蕴涵项;
- ◆ 在计算机资源需求量方面接近于收缩算法, 而解题精度方面接近于精选法;
- ◆ 不属于基本内容, 有兴趣者可自学;



## 4.8 组合逻辑电路的变换

- ◆ 多级逻辑电路 → 二级逻辑电路;
- ◆ 二级逻辑电路 → 多级逻辑电路;
- ◆ 二级与多级逻辑电路举例:



## 4.8.1 多级逻辑电路 → 二级逻辑电路

- ◆ 准备工作：对每一个基本元件建立其 ON-Cover 和 OFF-Cover，例如：

	ON - Cover	OFF - Cover
NAND2	0X X0	11
NOR3	000	1XX X1X XX1
NOT	0	1



## 多级逻辑电路 → 二级逻辑电路 (续)

准备工作:

◆ 信号分为3类:

- 原始输入信号  $I_1, I_2, \dots, I_n$ ;
- 中间信号  $S_1, S_2, \dots, S_k$ ;
- 最终输出信号  $Out$ ;

◆ 对信号排序, 假定为:

{ 输出信号, 中间信号, 输入信号 }



## 多级 $\rightarrow$ 二级 (算法描述)

Step1 建立输出信号的覆盖C:

- 对于Out:  $C = \{ 1XX \dots X \};$
- 对于 $\overline{Out}$  :  $C = \{ 0XX \dots X \};$

Step2 对  $c \in C$  作以下操作, 直到C中的输出及中间信号的取值全部变为X为止:

(1) 依次扫描  $c$  的输出信号及中间信号的取值,

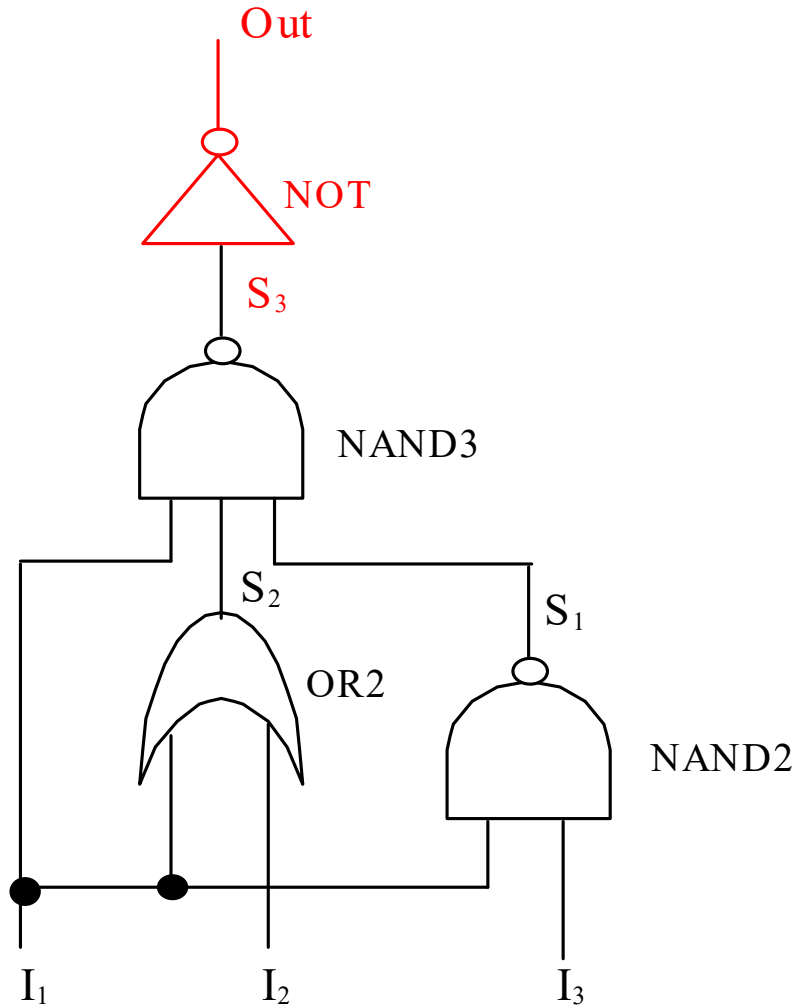
- 若取值为0或1, 将其改为 X 后用 d 表示,
- 若取值为1, 令D代表其 ON-Cover;
- 若取值为0, 令D代表其 OFF-Cover;

$$(2) C = S[(C - c) \cup (D \cap d)]$$

吸收运算



## 多级 → 二级 (实例)



准备：信号排序

Out,  $S_3$ ,  $S_2$ ,  $S_1$ ,  $I_1$ ,  $I_2$ ,  $I_3$

Step1  $C = \{ 1XXXXXXXX \}$

Step2 C中只有1个元素 c :

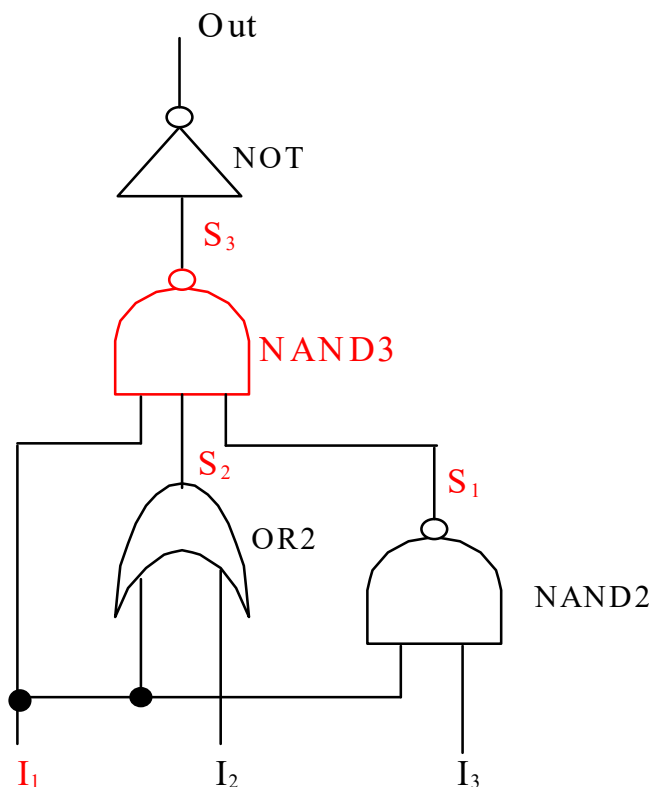
(1)  $d = \{ XXXXXXXX \}$ ;

$D = \{ X0XXXXXX \}$ ;

(2)  $C = S[ (C - c) \cup (D \cap d) ]$   
 $= \{ X0XXXXXX \}$ ;



## 多级 → 二级 (实例 -- 续)



信号排序:

Out, S<sub>3</sub>, S<sub>2</sub>, S<sub>1</sub>, I<sub>1</sub>, I<sub>2</sub>, I<sub>3</sub>

前一步结果:

$C = \{ X0XXXXX \};$

重复Step2:

Step2' C中只有1个元素 c :

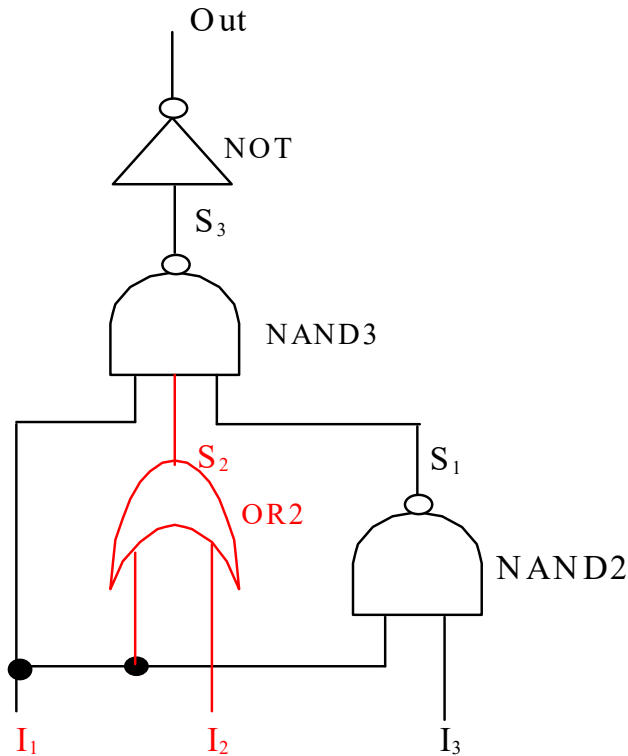
(1)  $d = \{ XXXXXXX \};$

$D = \{ XX11XX \};$

(2)  $C = S[ (C - c) \cup (D \cap d) ]$   
 $= \{ XX11XX \};$



## 多级 → 二级 (实例 -- 续)



信号排序:

Out, S<sub>3</sub>, S<sub>2</sub>, S<sub>1</sub>, I<sub>1</sub>, I<sub>2</sub>, I<sub>3</sub>

前一步结果:

$C = \{ \text{XX} \color{red}{11} \text{XX} \};$

重复Step2:

Step2” C中只有1个元素 c :

(1)  $d = \{ \text{XX} \color{blue}{X} 11 \text{XX} \};$

$$D = \left\{ \begin{array}{cc} \text{XXXX} & \underline{\underline{1XX}} \\ \text{XXXX} & \underline{\underline{X1X}} \end{array} \right\}$$

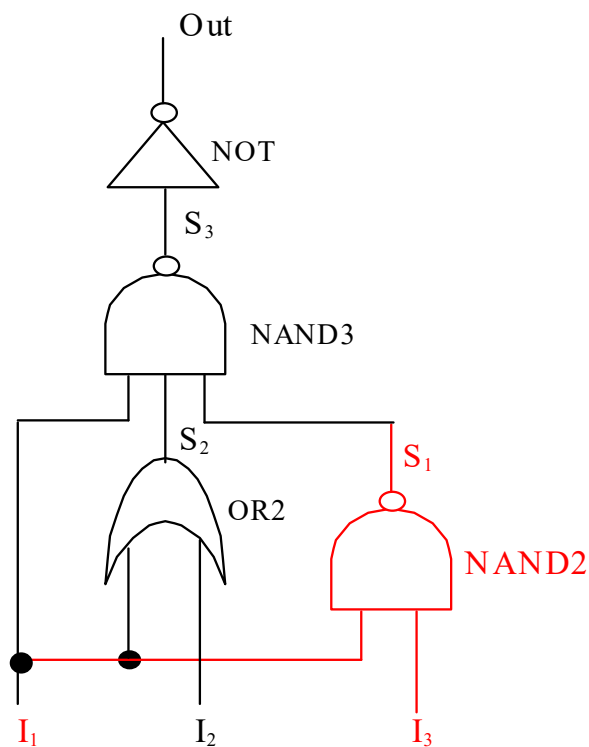
(2)  $C = S[ (C - c) \cup (D \cap d) ]$

$$\begin{aligned} C &= S \left\{ \begin{array}{c} \text{XXX11XX} \\ \text{XXX111X} \end{array} \right\} \\ &= \{ \text{XXX11XX} \} \end{aligned}$$





## 多级 → 二级 (实例 -- 续)



信号排序:

Out, S<sub>3</sub>, S<sub>2</sub>, S<sub>1</sub>, I<sub>1</sub>, I<sub>2</sub>, I<sub>3</sub>

前一步结果:

$C = \{ XXX1XX \};$

重复Step2:

Step2''' C中只有1个元素 c :

(1)  $d = \{ XXXX1XX \};$

$$D = \left\{ \begin{array}{ccc} XXXX & 0 & X \\ & = & = \\ XXXX & X & X \\ & = & = \\ & & 0 \end{array} \right\}$$

(2)  $C = S[ (C - c) \cup (D \cap d) ]$

$$\underline{C = \{ XXXX1X0 \}}$$

-- 最终解

(输出信号及中间信号取值均为X)

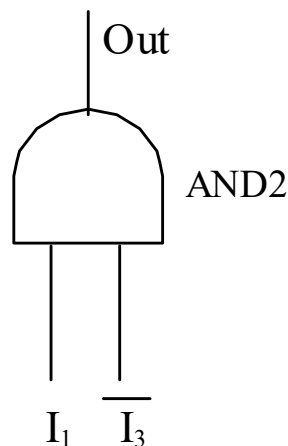


## 多级 → 二级 (实例结果分析)

前一步结果 (最终解) :  $C = \{ XXXX1X0 \}$ ;

◆ C所对应的布尔表达式及对应的逻辑图:

$$Out = I_1 \overline{I_3}$$



说明:

本例的多级逻辑电路造价较高; 经过运算得到化简;  
一般情况下, 多级逻辑电路造价较低, 本例是一种特殊情况。

## 多级 $\rightarrow$ 二级 (算法描述 -- 重复 p.123)

Step1 建立输出信号的覆盖C:

- 对于Out:  $C = \{ 1XX \dots X \};$
- 对于 $\overline{Out}$  :  $C = \{ 0XX \dots X \};$

Step2 对  $c \in C$  作以下操作, 直到C中的输出及中间信号的取值全部变为X为止:

(1) 依次扫描  $c$  的输出信号及中间信号的取值,

- 若取值为0或1, 将其改为 X 后用  $d$  表示,
- 若取值为1, 令D代表其 ON-Cover;
- 若取值为0, 令D代表其 OFF-Cover;

$$(2) C = S[(C - c) \cup (D \cap d)]$$

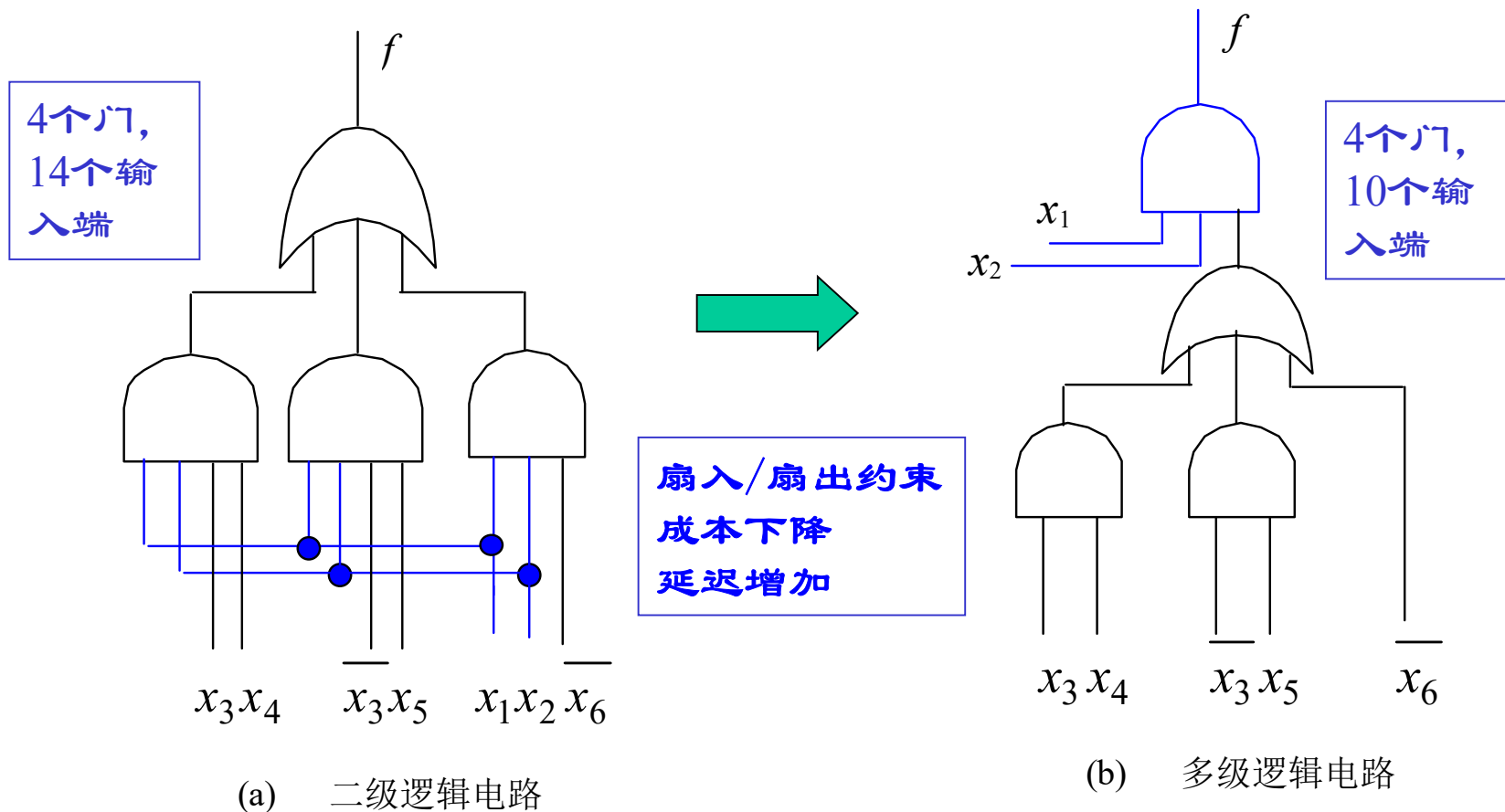
吸收运算

## 4.8.2 二级逻辑电路 → 多级逻辑电路

举例：

$$f = x_1 x_2 x_3 x_4 + x_1 x_2 \bar{x}_3 x_5 + x_1 x_2 \bar{x}_6$$

$$f = x_1 x_2 (x_3 x_4 + \bar{x}_3 x_5 + \bar{x}_6)$$





## 二级 → 多级 (提取公因子效果分析)

- ◆ 提取公因子前:

$$f = x_1 x_2 x_3 x_4 + x_1 x_2 \overline{x_3} x_5 + x_1 x_2 \overline{x_6}$$

对应的二级电路共需4个门, 14个输入端。

- ◆ 提取公因子后:

$$f = x_1 x_2 (x_3 x_4 + \overline{x_3} x_5 + \overline{x_6})$$

对应的多级电路共需4个门, 10个输入端。

- ◆ 提取公因子后可导致:

- (1) 元件的输入端减少, 可解决扇入FI所带来的限制。同时也缓解了扇出FO所带来的限制。
- (2) 成本有可能降低
- (3) 逻辑级数有可能增加, 即电路的总延迟时间可能增加。



## 二级 → 多级 (续)

提取公因子方法举例:

◆ 原始描述:

$$f = x_1 x_2 x_3 x_4 + x_1 x_2 \overline{x_3} x_5 + x_1 \overline{x_2} \overline{x_6}$$

◆ 宽度优先 (w = 2, h = 2) :

$$f = x_1 x_2 (x_3 x_4 + \overline{x_3} x_5) + x_1 \overline{x_2} \overline{x_6}$$

◆ 高度优先 (w = 1, h = 3) :

$$f = x_1 (x_2 x_3 x_4 + x_2 \overline{x_3} x_5 + \overline{x_2} \overline{x_6})$$

基本方法:

◆ 提取公因子:

- W算法 (宽度优先)
- H算法 (高度优先)
- G算法 (宽度与高度乘积优先)

◆ 先农展开定理

◆ 二叉判决图

(Binary Decision Diagram, BDD)



## 提取公因子的立方体表示

◆ 把函数  $f$  表示为它的最小化覆盖  $C$  :

$$C = \left\{ \begin{array}{l} 1 \ 1 \ 1 \ 1 \ X \ X \\ 1 \ 1 \ 0 \ X \ 1 \ X \\ 1 \ 1 \ X \ X \ X \ 0 \end{array} \right\} \begin{array}{l} \dots\dots c_1 \\ \dots\dots c_2 \\ \dots\dots c_3 \end{array}$$

◆  $\gamma = 111XXX$  是  $c_1$  的因子。  $w=3, h=1$ 。  
 $\gamma = 1111XX$  也是  $c_1$  的因子。  $w=4, h=1$ 。

◆  $\gamma = 11XXXX$  是  $c_1, c_2, c_3$  的公因子。  $w=2, h=3$ 。



## 提取公因子的立方体表示 (续)

- ◆ 把函数  $f$  表示为它的最小化覆盖  $C$ :

$$C = \left\{ \begin{array}{cccccc} 1 & 1 & 1 & 1 & X & X \\ 1 & 1 & 0 & X & 1 & X \\ 1 & 1 & X & X & X & 0 \end{array} \right\}$$

- ◆ 改写作:

$$C = \gamma \cap D$$

$$\gamma = 11XXXX$$

- 其中  $\gamma$  是公因子;
- $D$  是从  $C$  中提取公因子  $\gamma$  后的剩余部分。

$$D = \left\{ \begin{array}{c} XX11XX \\ XX0X1X \\ XXXXX0 \end{array} \right\}$$





## 提取公因子（算法描述）

◆ 立方体表示：

$$\gamma = \gamma_1 \gamma_2 \cdots \gamma_i \cdots \gamma_n$$

$$c = c_1 c_2 \cdots c_i \cdots c_n$$

$$d = d_1 d_2 \cdots d_i \cdots d_n$$

若  $c \subseteq \gamma$  则称 $\gamma$ 是 $c$ 的因子；

若  $c \subseteq \gamma$  且  $d \subseteq \gamma$  则称 $\gamma$  是 $c$ 和 $d$ 的公因子。

◆ 公因子 $\gamma$ 的高度 $h(\gamma)$ ：设覆盖 $C$ 中元素的个数为 $m$ ， $\gamma$ 是其中 $h$ 个元素的公因子，则 $h(\gamma) = h$

◆ 公因子 $\gamma$ 的宽度： $w(\gamma) = cs(\gamma)$

= (变量个数 $n$  -  $\gamma$ 中取值为 $X$ 的个数)



## 提取公因子 (m积)

◆ **m积** (mask product): 设

$$a = c_1 c_2 \dots c_i \dots c_n \mid d^1 d^2 \dots d^j \dots d^m$$

$$b = e_1 e_2 \dots e_i \dots e_n \mid f^1 f^2 \dots f^j \dots f^m$$

◆ **定义:**

(1) 若  $c_i = e_i \neq X$

$$\text{则} \quad (c_i) \text{ m } (e_i) = c_i$$

$$\text{否则} \quad (c_i) \text{ m } (e_i) = X$$

(2) 若  $d^j = f^j \neq u$

$$\text{则} \quad (d^j) \text{ m } (f^j) = 1$$

$$\text{否则} \quad (d^j) \text{ m } (f^j) = u$$

例1

$$\begin{array}{r} 10X11 \mid 1u1u \dots a \\ m) \quad 101X0 \mid 111u \dots b \end{array}$$


---

$$10XXX \mid 1u1u \dots \gamma$$

例2

$$\begin{array}{r} 11011XX \mid 1 \dots a \\ m) \quad 1111000 \mid 1 \dots b \end{array}$$


---

$$11X1XXX \mid 1 \dots \gamma$$



## W算法

### 以宽度最大为目标

- (1) 从覆盖 $C$ 中选一宽度最大的元素 $c$ ;
- (2)  $c$ 和 $C$ 中其它任一元素作 $m$ 积, 从诸 $m$ 积中选一宽度最大者作为本次提出的公因子 $\gamma$ ;
- (3) 记下公因子 $\gamma$  和形成 $\gamma$  的两个元素;
- (4) 从覆盖 $C$ 中删去形成 $\gamma$  的两个元素, 以 $\gamma$  代替它们;
- (5) 若覆盖 $C$  中提不出公因子时停止, 否则转入(1)。



(1) 立方体8的宽度最大。

(3) 记下公因子 $\gamma$  和形成 $\gamma$  的两个立方体:

1 XXXX

X X X X X 0 0 X ...148

前页结果

## W算法 --实例

(4)

0 0 0 0 1 X X X  
 X X X X X X 1 1  
 X X X X X 0 0 X

C {
 

a	b	c	d	e	f	g	h
1	1	1	X	1	0	0	X
1	1	1	X	1	X	0	1
1	1	0	0	X	X	1	1
		1	1	0	0	X	1 X 1
0	0	X	1	0	X	1	0
0	0	X	1	0	0	1	X
0	0	0	0	1	X	X	X

(1) 立方体6的宽度最大。

(2) 立方体6和C中其它元素作m积，选其中宽度最大者为本次提取的公因子：

$$\gamma = 5 \cdot 6 = 00X10X1X$$

.....1  
 .....2  
 .....3  
 .....4  
 .....5  
 .....6  
 .....7 • 8

(5) C中元素个数 > 1，转入(1)。重复以上步骤，进一步提取公因子。



## W算法 --实例

前页结果

- (1) 立方体6的宽度最大。
- (2) 立方体6和C中其它元素作m积，选其中宽度最大者为本次提取的公因子：  
 $\gamma = 5 \cdot 6 = 00X10X1X$

(3)

$$\begin{array}{c} 00X10X1X \\ \left\{ \begin{array}{l} XXXXX0 \\ XXXX0XX \end{array} \right. \end{array}$$

(4)

$$C = \left\{ \begin{array}{cccccc} a & b & c & d & e & f & g & h \\ 1 & 1 & 1 & X & 1 & 0 & 0 & X \\ 1 & 1 & 1 & X & 1 & X & 0 & 1 \\ 1 & 1 & 0 & 0 & X & X & 1 & 1 \\ 1 & 1 & 0 & 0 & X & 1 & X & 1 \\ 0 & 0 & X & 1 & 0 & X & 1 & X \\ 0 & 0 & 0 & 0 & 1 & X & X & X \end{array} \right\} \begin{array}{l} \dots\dots 1 \\ \dots\dots 2 \\ \dots\dots 3 \\ \dots\dots 4 \\ \dots\dots 5 \cdot 6 \\ \dots\dots 7 \cdot 8 \end{array}$$



## W算法 --实例

(续前)

(5)  $C$  中元素个数  $> 1$ , 转入 (1) 进入下一轮。本次所得结果为:

$$\gamma = 1 \cdot 2 = 1 \ 1 \ 1 \ X \ 1 \ X \ 0 \ X$$

$$1 \ 1 \ 1 \ X \ 1 \ X \ 0 \ X \quad \left\{ \begin{array}{l} X \ X \ X \ X \ X \ 0 \ X \ X \\ X \ X \ X \ X \ X \ X \ X \ 1 \end{array} \right.$$

$$C = \left\{ \begin{array}{l} a \ b \ c \ d \ e \ f \ g \ h \\ 1 \ 1 \ 1 \ X \ 1 \ X \ 0 \ X \\ 1 \ 1 \ 0 \ 0 \ X \ X \ 1 \ 1 \\ 1 \ 1 \ 0 \ 0 \ X \ 1 \ X \ 1 \\ 0 \ 0 \ X \ 1 \ 0 \ X \ 1 \ X \\ 0 \ 0 \ 0 \ 0 \ 1 \ X \ X \ X \end{array} \right\} \begin{array}{l} \dots\dots 1 \cdot 2 \\ \dots\dots 3 \\ \dots\dots 4 \\ \dots\dots 5 \cdot 6 \\ \dots\dots 7 \cdot 8 \end{array}$$



## W算法 --实例

(续前) 下一轮结果为:

$$\gamma = 3 \cdot 4 = 1\ 1\ 0\ 0\ X\ X\ X\ 1$$

$$1\ 1\ 0\ 0\ X\ X\ X\ 1 \quad \left\{ \begin{array}{l} X\ X\ X\ X\ X\ X\ 1\ X \\ X\ X\ X\ X\ X\ 1\ X\ X \end{array} \right.$$

$$C = \left\{ \begin{array}{l} a\ b\ c\ d\ e\ f\ g\ h \\ 1\ 1\ 1\ X\ 1\ X\ 0\ X \\ 1\ 1\ 0\ 0\ X\ X\ X\ 1 \\ 0\ 0\ X\ 1\ 0\ X\ 1\ X \\ 0\ 0\ 0\ 0\ 1\ X\ X\ X \end{array} \right\} \begin{array}{l} \dots\dots 1 \cdot 2 \\ \dots\dots 3 \cdot 4 \\ \dots\dots 5 \cdot 6 \\ \dots\dots 7 \cdot 8 \end{array}$$





## W算法 --实例

(续前) 下一轮结果为:

$$\gamma = 12 \cdot 34 = 11XXXXXX$$

$$\begin{array}{c}
 11XXXXXX \\
 \left\{ \begin{array}{l}
 XX1X1X0X \\
 \\
 XX00XXXX1
 \end{array} \right. \left\{ \begin{array}{l}
 XXXXX0XX \\
 \\
 XXXXXXX1 \\
 XXXXXXX1X \\
 \\
 XXXXX1XX
 \end{array} \right.
 \end{array}$$

$$C = \left\{ \begin{array}{l}
 abcdefgh \\
 11XXXXXX \\
 00X10X1X \\
 00001XXX
 \end{array} \right\} \begin{array}{l}
 \dots\dots 12 \cdot 34 \\
 \dots\dots 5 \cdot 6 \\
 \dots\dots 7 \cdot 8
 \end{array}$$



# W算法 --实例

前页结果

(续前)

下一轮结果为:

11XXXXXX

XX1X1X0X

XX00XXX1

XXXXXX0XX

XXXXXXXXX1

XXXXXXXXX1X

XXXXXX1XX

$$\gamma = 56 \cdot 78 = 00XXXXXX$$

00XXXXXX

XXX10X1X

XX001XXXX

XXXXXXXXX0

XXXXXXXX0XX

XXXXXXXXX11

XXXXXXXX00X

$$C = \left\{ \begin{array}{c} a \ b \ c \ d \ e \ f \ g \ h \\ 1 \ 1 \ X \ X \ X \ X \ X \ X \\ 0 \ 0 \ X \ X \ X \ X \ X \ X \end{array} \right\} \begin{array}{l} \dots\dots 12 \cdot 34 \\ \dots\dots 56 \cdot 78 \end{array}$$

⇒

$$f = ab \{ c e \bar{g} (\bar{f} + h) + \bar{c} \bar{d} h (g + f) \} + \bar{a} \bar{b} \{ d \bar{e} g (\bar{h} + \bar{f}) + \bar{c} \bar{d} e (gh + \bar{f} \bar{g}) \}$$

此表达式对应于多级逻辑



### 以宽度最大为目标

- (1) 从覆盖 $C$ 中选一宽度最大的元素 $c$ ;
- (2)  $c$ 和 $C$ 中其它任一元素作 $m$ 积, 从诸 $m$ 积中选一宽度最大者作为本次提出的公因子 $\gamma$ ;
- (3) 记下公因子 $\gamma$  和形成 $\gamma$  的两个元素;
- (4) 从覆盖 $C$ 中删去形成 $\gamma$  的两个元素, 以 $\gamma$  代替它们;
- (5) 若覆盖 $C$  中提不出公因子时停止, 否则转入(1)。



## H算法

◆ 以高度  $h(\gamma)$  最大为目标，在覆盖  $C$  中立方体之间求一公因子  $\gamma$ 。

◆ 求出  $\gamma$  之后，把覆盖  $C$  分解为两个子集： $S$  和  $N$

$$S = ( e \mid e \in C \quad \text{且} \quad e \subseteq \gamma );$$

$$N = C - S;$$

•  $S = \gamma \cap D$       —  $\gamma$  是  $S$  中每一元素的因子

•  $N$  是没有提取公因子的剩余部分；

◆ 继续对  $D$  和  $N$  提取公因子，直到提不出时为止。



## H算法

### 算法描述:

(1) 形成覆盖  $c$  的列高度函数:

$h(0) = a_1 a_2 \dots a_i \dots a_n$        $a_i$  代表覆盖  $C$  中变量  $x_i$  取值为 0 的个数;

$h(1) = b_1 b_2 \dots b_i \dots b_n$        $b_i$  代表  $C$  中变量  $x_i$  取值为 1 的个数。

(2) 求出最大列高度:

$$a_k = \text{MAX}(a_1 a_2 \dots a_i \dots a_n) \quad k \in (1, 2, \dots, n)$$

$$b_j = \text{MAX}(b_1 b_2 \dots b_i \dots b_n) \quad j \in (1, 2, \dots, n)$$

(3) 若  $a_k < 2$  且  $b_j < 2$  则停止。

(4) 若  $a_k > b_j$  则  $\gamma_k = 0$  且  $\gamma_i = X$  ( $i \neq k$ );

否则  $\gamma_j = 1$  且  $\gamma_i = X$  ( $i \neq j$ )。



## H算法 --实例

设:

$$C = \begin{Bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ 1 & 1 & X & 1 & 0 & X \\ 1 & 0 & 0 & X & 1 & 1 \\ 1 & 0 & X & 1 & X & 1 \\ 0 & 0 & 0 & 1 & X & X \end{Bmatrix}$$

$$h(0) = 1 \ 3 \ 2 \ 0 \ 1 \ 0$$

$$h(1) = 3 \ 1 \ 0 \ 3 \ 1 \ 2$$

最大列高度

$$a_2 = 3$$

$$b_1 = 3$$

提取公因子

$$\gamma = 1 \ X \ X \ X \ X \ X$$

把C分解为:

$$S = \begin{Bmatrix} 11X10X \\ 100X11 \\ 10X1X1 \end{Bmatrix}$$

$$N = \{0 \ 0 \ 0 \ 1 \ X \ X\}$$

对S提出公因子  $\gamma$  后, 剩余部分为 **D**:

$$S = \gamma \cap D$$

$$D = \begin{Bmatrix} X1X10X \\ X00X11 \\ X0X1X1 \end{Bmatrix}$$



## H算法 --实例（续）

◆ 分别对 $D$ 和 $N$ 进一步提取公因子：

- $N$ 中只有一个元素，不可能再提公因子；
- 对 $D$ 提公因子。令  $C = D$ ，重复上述过程：

$$C = \begin{Bmatrix} X1X10X \\ X00X11 \\ X0X1X1 \end{Bmatrix}$$

$$h(0) = 0\ 2\ 1\ 0\ 1\ 0$$

$$h(1) = 0\ 1\ 0\ 2\ 1\ 2$$

最大列高度：

$$a_2 = 2$$

$$b_4 = 2$$

◆ 提取公因子  $\gamma = XXX1XX$

◆ 把 $C$ 分解为：

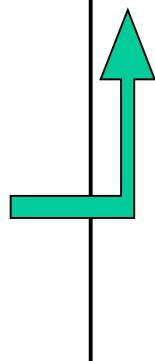
$$S = \begin{Bmatrix} X1X10X \\ X0X1X1 \end{Bmatrix}$$

$$N = \{X00X11\}$$

◆ 对 $S$ 提取公因子 $\gamma$ 后，剩余部分为 $D$ ：

$$D = \begin{Bmatrix} X1XX0X \\ X0XXX1 \end{Bmatrix}$$

◆  $N$ 和 $D$ 都不存在公因子，结束。





## H算法 --实例（续）

◆ 结果：

$$C = (1XXXXX) \cap \left\{ \left[ (XXX1XX) \cap \left\{ \begin{matrix} X1XX0X \\ X0XXX1 \end{matrix} \right\} \right] \cup (X00X11) \right\} \cup (0001XX)$$

对应的布尔表达式为：

$$f = x_1 (x_4 (\overline{x_2 x_5} + \overline{x_2 x_6}) + \overline{x_2 x_3 x_5 x_6}) + \overline{x_1 x_2 x_3 x_4}$$

◆ 此表达式对应于多级逻辑电路。 





## H算法描述

-- 重复p. 147

(1) 形成覆盖  $c$  的列高度函数：

$h(0) = a_1 a_2 \dots a_i \dots a_n$        $a_i$  代表覆盖  $C$  中变量  $x_i$  取值为 0 的个数；

$h(1) = b_1 b_2 \dots b_i \dots b_n$        $b_i$  代表  $C$  中变量  $x_i$  取值为 1 的个数。

(2) 求出最大列高度：

$$a_k = \text{MAX}(a_1 a_2 \dots a_i \dots a_n) \quad k \in (1, 2, \dots, n)$$

$$b_j = \text{MAX}(b_1 b_2 \dots b_i \dots b_n) \quad j \in (1, 2, \dots, n)$$

(3) 若  $a_k < 2$  且  $b_j < 2$  则停止。

(4) 若  $a_k > b_j$  则  $\gamma_k = 0$  且  $\gamma_i = X \quad (i \neq k)$  ；

否则  $\gamma_j = 1$  且  $\gamma_i = X \quad (i \neq j)$ 。



## G算法

- ◆ G算法以 $M(\gamma)$ 最大为目标。在覆盖 $C$ 中立方体之间求一公因子 $\gamma$ 。
- ◆  $M(\gamma)$ 的定义：

$$M(\gamma) = h(\gamma) \cdot w(\gamma)$$



## G算法 (续)

### ◆ 算法描述:

(1) 覆盖 $C$ 中每一立方体的初始高度 $h$ 都为1, 标明其高度 $h$ 和优值 $M$ 。

(2) 令  $\gamma = \mathbf{XX} \dots \mathbf{X} \dots \mathbf{X}$   $M(\gamma) = 0$

(3) 从覆盖 $C$ 中选一优值 $M$ 最大的元素 $c$ 。

(4)  $c$ 和覆盖 $C$ 中其它任一元素作 $\mathbf{m}$ 积, 并标明其高度 $h$ 和优值 $M$ 。

(5) 从诸 $\mathbf{m}$ 积中选出一优值最大者 $\beta$ 。

(6) 用 $\beta$ 代替覆盖 $C$ 中由 $\mathbf{m}$ 积形成 $\beta$ 的两个元素。

(7) 若  $M(\gamma) < M(\beta)$

则令  $M(\gamma) = M(\beta)$   $\gamma = \beta$

(8) 若覆盖 $C$ 被简化到只有一个立方体时则停止, 否则转入(3)。

◆ 把 $C$ 分解为两个子集 $S$ 和 $N$ , 并进一步提取公因子.....

(和H算法相同)



## 4.8.3 先农展开定理及BDD表示法

### ◆ 先农展开定理 (Shannon's Expansion Theorem) :

$$f(x_1, x_2, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) = x_i f_1(x_1, x_2, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) \\ + \overline{x_i} f_0(x_1, x_2, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$$

### ◆ 通过真值表理解先农展开定理:

$x_1$	$x_2$	$x_3$	$f$
<b>0</b>	0	0	1
	0	1	0
	1	0	1
	1	1	1

$x_1$	$x_2$	$x_3$	$f$
<b>1</b>	0	0	0
	0	1	0
	1	0	1
	1	1	0



## 先农展开定理及BDD表示法（续）

◆ 先农展开定理（Shannon's Expansion Theorem）：

$$f(x_1, x_2, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) = x_i f_1(x_1, x_2, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) \\ + \overline{x_i} f_0(x_1, x_2, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$$

◆ 效果：1个布尔函数分解为2个布尔函数  $\Rightarrow$  变量个数减少 1  $\Rightarrow$  复杂度降低。

◆ 适合于手工作业。

◆ 先农展开定理  $\Rightarrow$  BDD表示  $\Rightarrow$  适合于计算机操作。



## 二叉判决图BDD

- ◆ **BDD** ( Binary Decision Diagram )  
Akers 1978 年 在 提 出 ；  
是基于先农展开的图形表示；是一个有向无环图。
- ◆ **OBDD** Ordered BDD 变量次序排定；
- ◆ **ROBDD** Reduced ROBDD 简化的OBDD  
R.E. Bryant1986年提出，1992年充实改进。  
并且证明**ROBDD** 的正则性。
- ◆ **ROBDD**适合于计算机作业。 R.E. Bryant建立  
了对**ROBDD**的一系列运算规则（函数）。
- ◆ 已经应用于多级逻辑综合和形式验证。
- ◆ 许多人简称**ROBDD**为**BDD**。

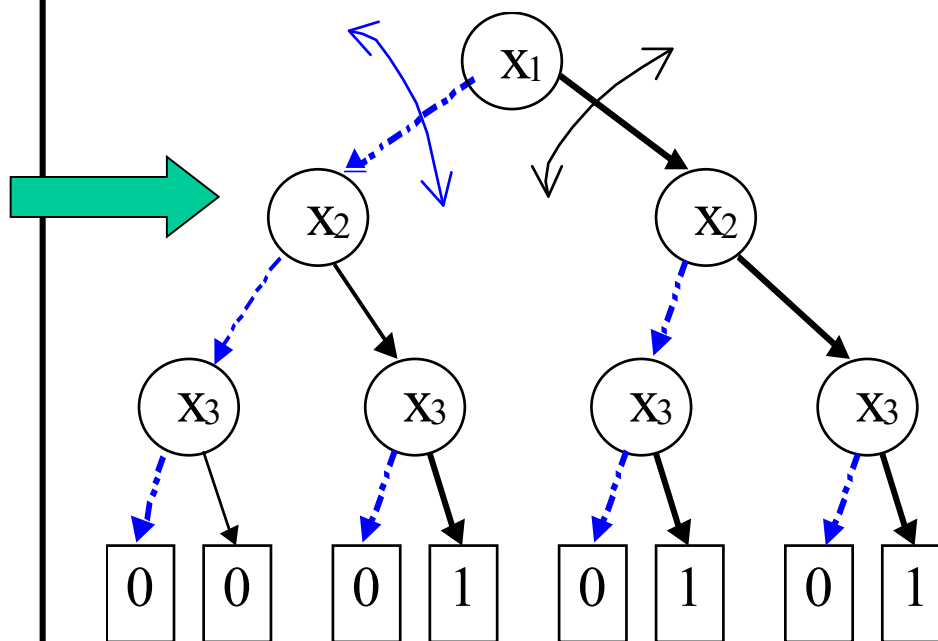


## 二叉判决图 (续)

- ◆ 函数  $f(x) = x_1x_3 + x_2x_3$  的真值表

$x_1$	$x_2$	$x_3$	$f$
0	0	0	0
		1	0
	1	0	0
		1	1
1	0	0	0
		1	1
	1	0	0
		1	1

- ◆ 二叉判决图:



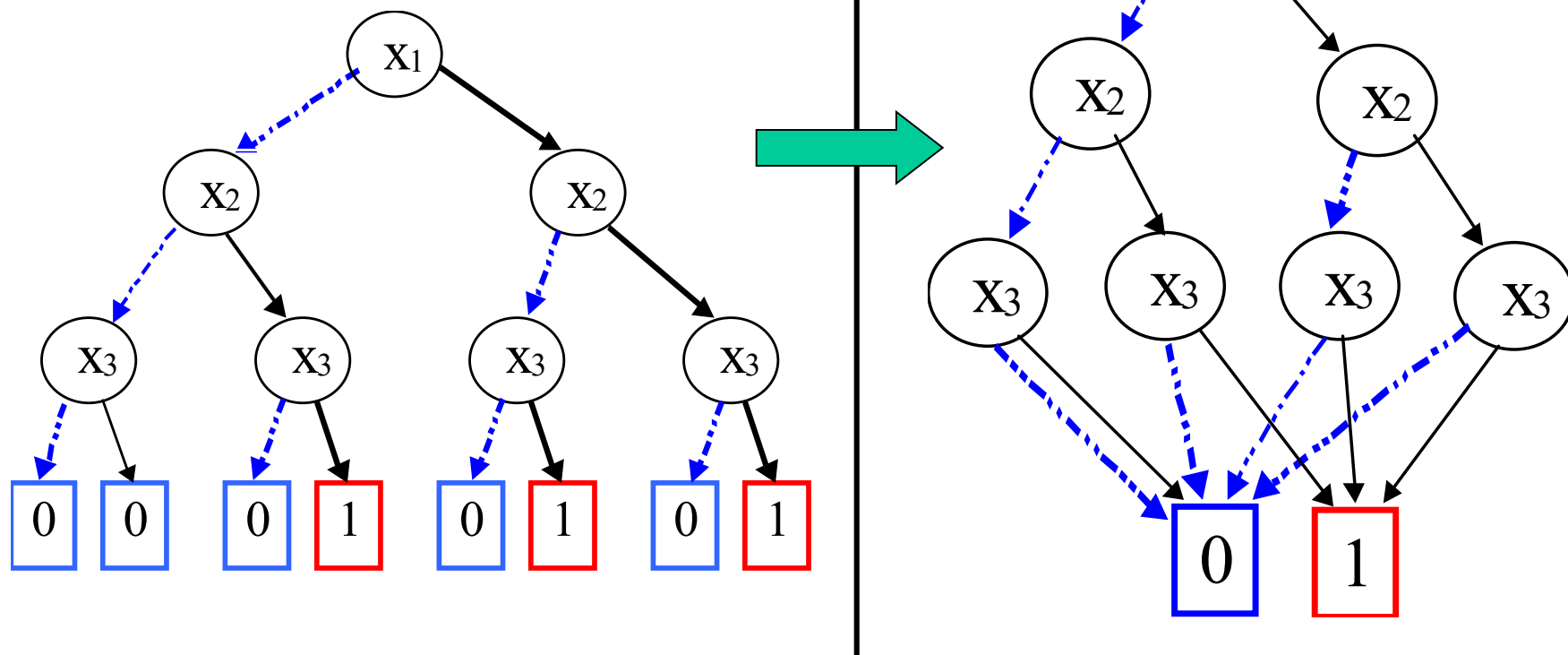
$$f(x_1, x_2, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) = x_i f_1(x_1, x_2, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) \\ + \overline{x_i} f_0(x_1, x_2, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$$



## 二叉判决图的化简 —— 规则1

### 删除重复的端节点:

- 对于属性值相同的端节点，只保留其中一个，删除其余属性值相同的端节点；
- 把指向已删除端节点的弧指向保留的那个端节点。





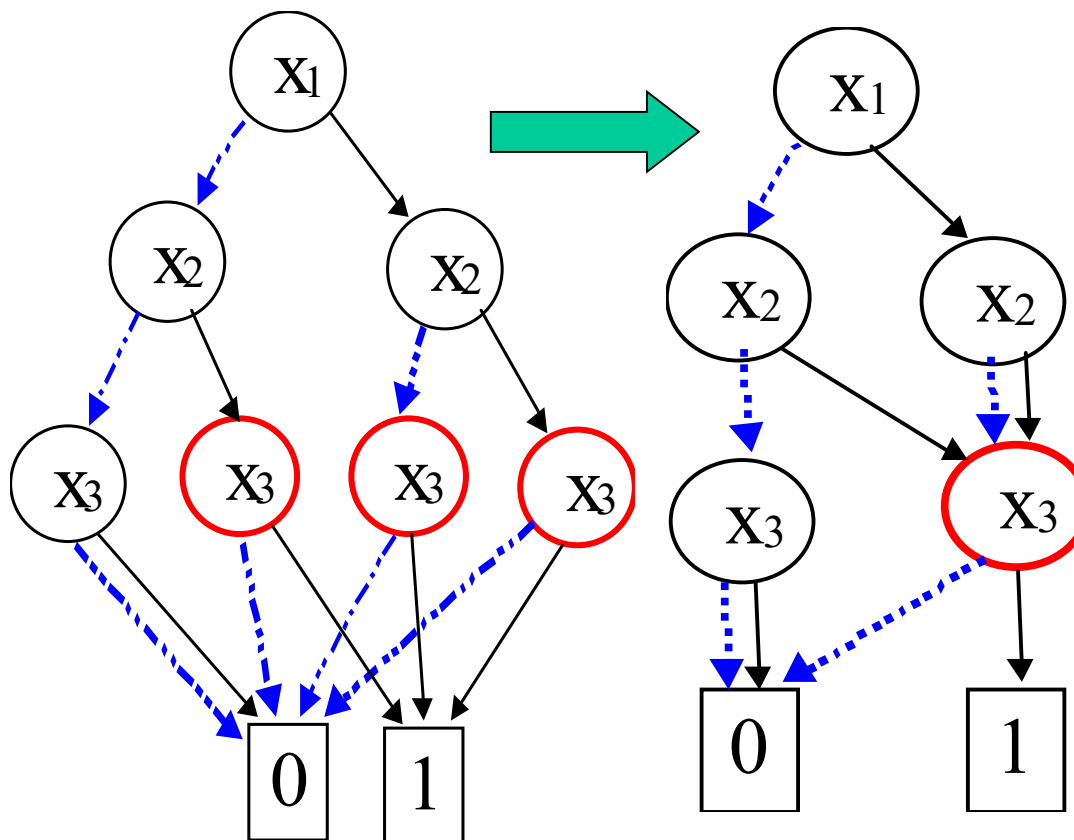


## 二叉判决图的化简

### — 规则2

#### ◆ 删除重复的非端节点:

- 对于非端节点 $u$ 和 $v$ , 如果 $index(u)=index(v)$ ,  $low(u)=low(v)$ ,  $high(u)=high(v)$ , 则删除两个节点中的一个;
- 并把指向已删除节点的弧指向未删除的那个节点。



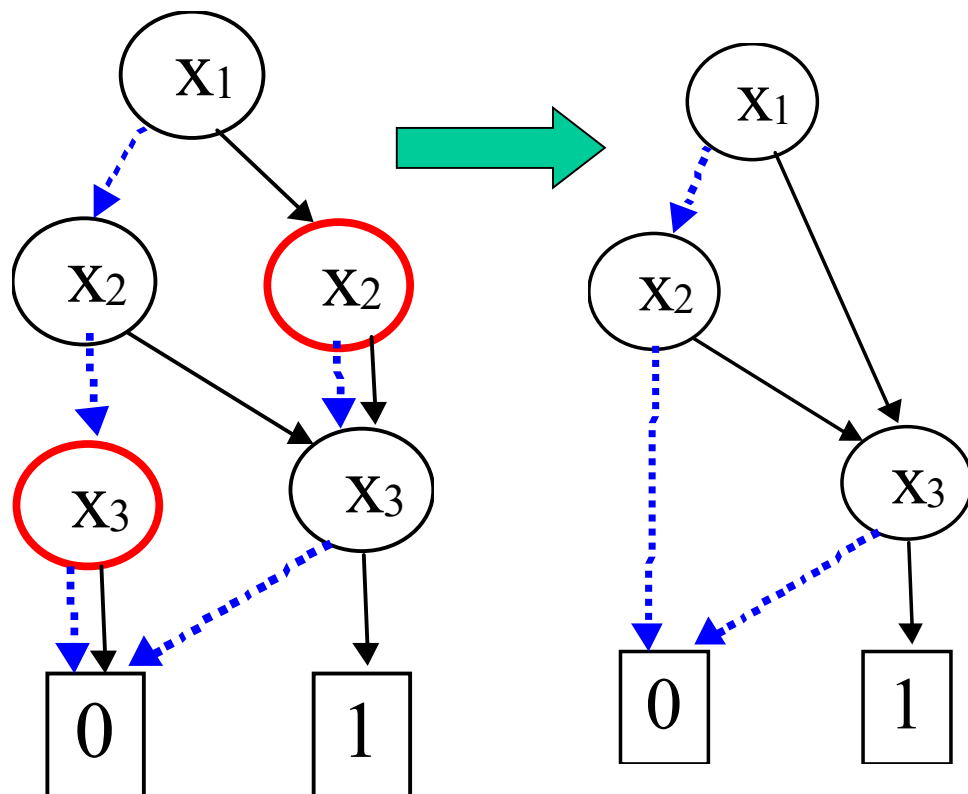


## 二叉判决图的化简

### ——规则3

#### ◆ 删除冗余的非端节点:

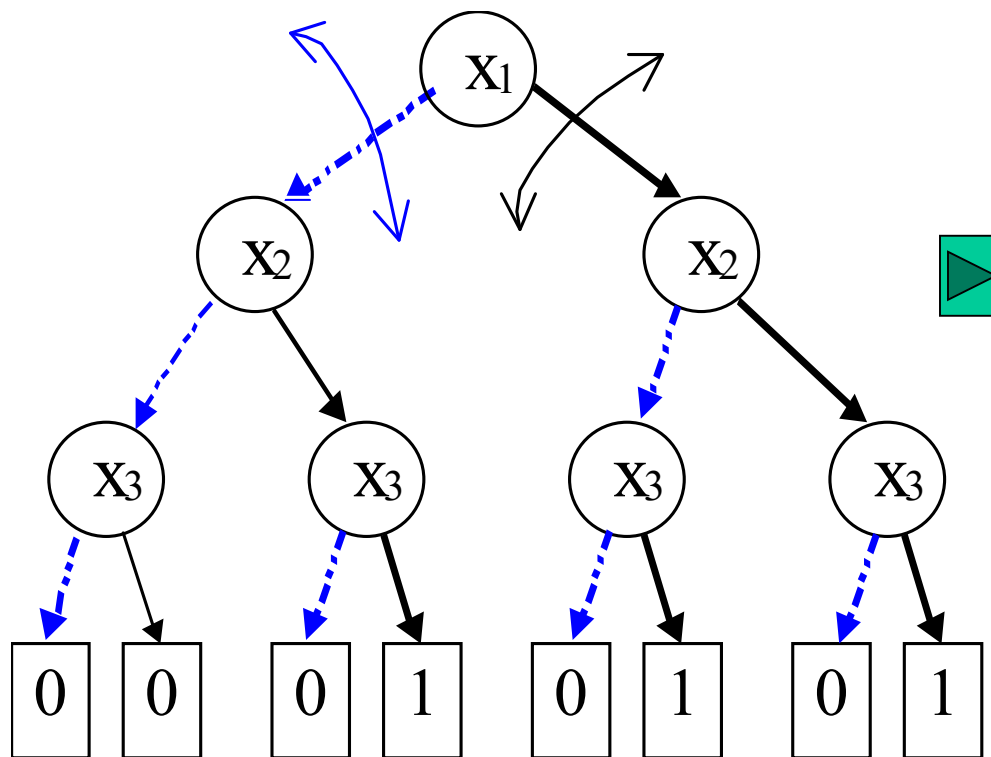
- 对于非端节点 $u$ , 如果 $low(u)=high(u)$ , 则删除 $u$ 节点;
- 并把指向 $u$ 节点的弧指向 $low(u)$ 。





## BDD的应用

- ◆ 应用于多级逻辑综合：
  - 在适当的地方**切割**，子图可看作新的**Sub\_BDD**；
  - 子图的**规模**已经**减小**！
  - 多次切割所得必然是**多级逻辑**，也可以适合于**查找表形式**
- ◆ 逻辑等价性形式验证：
  - 利用**ROBDD**的**正则性**；



$$f(x_1, x_2, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) = x_i f_1(x_1, x_2, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) + \overline{x_i} f_0(x_1, x_2, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$$



## BDD应用于PLD -- 应用实例

- ◆ 宏单元的组成：  
组合逻辑电路 + 触发器
- ◆ 组合逻辑电路规模(输入变量个数)：
  - CPLD: 8 ~ 10
  - FPGA: 3 ~ 5
- ◆ 宏单元中组合逻辑的实现方法：
  - CPLD: 乘积项之和(用与-或阵列实现)；
  - FPGA: 查找表(用SRAM/ROM实现)；

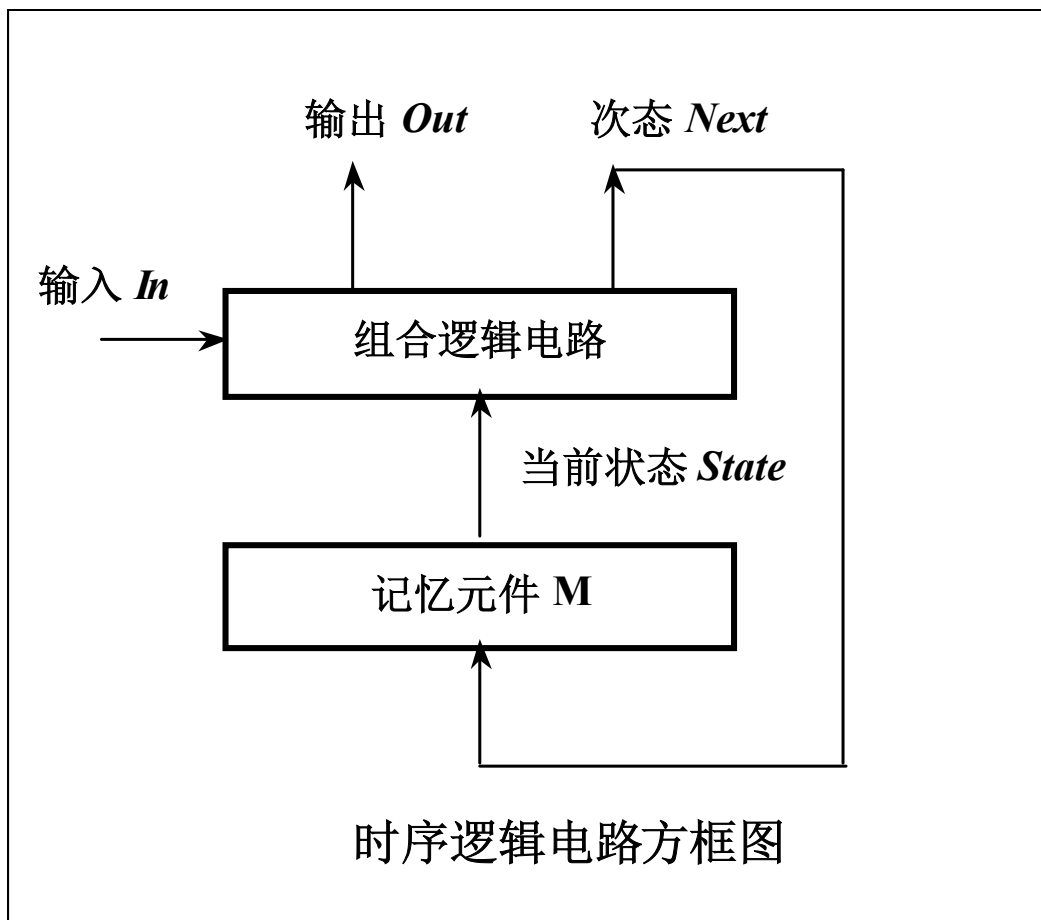
- ◆ 用BDD 将复杂的组合逻辑电路分解为由多级宏单元实现；
  - 对于CPLD: SoP;
  - 对于FPGA: ROM;





## 4.9 时序逻辑电路的综合

- ◆ 时序机的数学模型
- ◆ 完全规定时序机的状态最小化
- ◆ 不完全规定时序机的状态化简
- ◆ 时序机的状态分配: 状态编码





## 4.9.1 时序机的数学模型

◆ 有限状态机**FSM** (Finite State Machine)

◆  $m=(i, z, s, \delta, \lambda)$

$i$  —— 输入

$z$  —— 输出

$s$  —— 状态

$\delta: i \times s \rightarrow s$  —— 状态迁移函数  $s(t+1)=n(s(t), i(t))$

$\lambda: i \times s \rightarrow z$  —— 输出函数 —— **Mealy模型**

或  $\lambda: s \rightarrow z$  —— 输出函数 —— **Moore模型**



# 时序机的描述

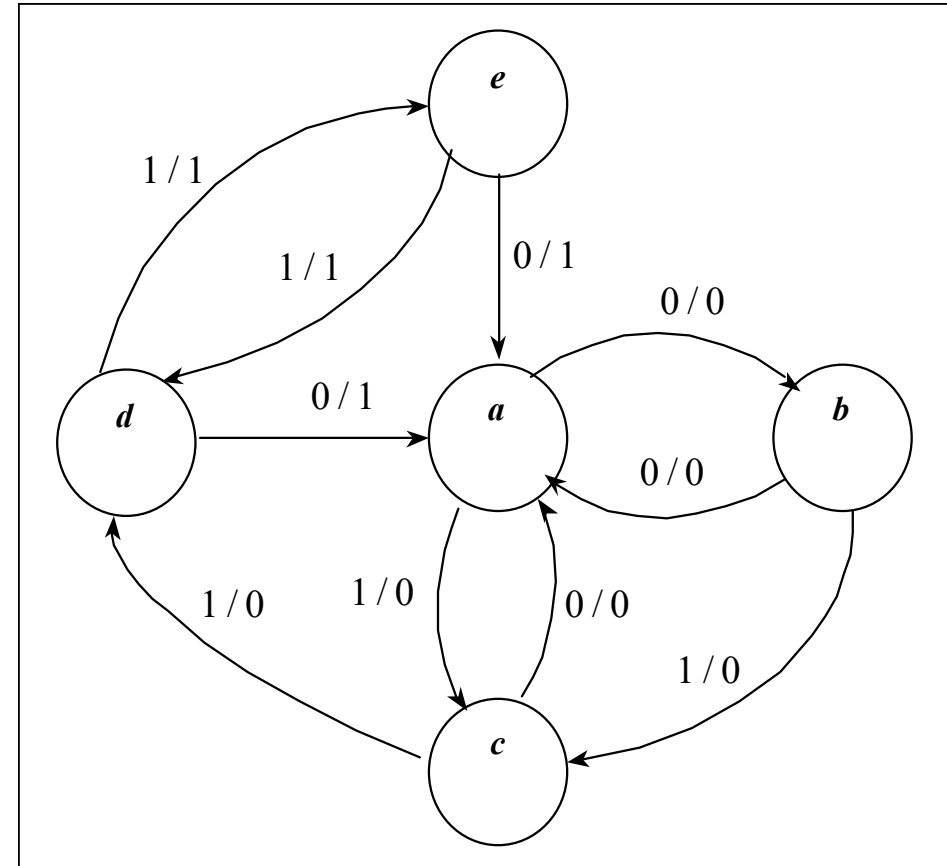
◆ 状态表:



◆ 状态图:

完全规定时序机 (Mealy 模型)

当前状态 <i>State</i>	输入信号 <i>In</i>	后继状态 <i>Next</i>	输出信号 <i>Out</i>
<i>a</i>	0	<i>b</i>	0
<i>a</i>	1	<i>c</i>	0
<i>b</i>	0	<i>a</i>	0
<i>b</i>	1	<i>c</i>	0
<i>c</i>	0	<i>a</i>	0
<i>c</i>	1	<i>d</i>	0
<i>d</i>	0	<i>a</i>	1
<i>d</i>	1	<i>e</i>	1
<i>e</i>	0	<i>a</i>	1
<i>e</i>	1	<i>d</i>	1





## 完全规定 $\longleftrightarrow$ 不完全规定

- ◆ 完全规定时序机：  
状态表中每一输入组合情况下，其次态和输出唯一确定；
- ◆ 不完全规定时序机：  
状态表中有不确定的次态  $\emptyset$  或未明确指定的输出u。





## 时序机的综合

- (1) 建立原始描述：状态迁移图或状态表；
- (2) 状态化简——去冗余，状态合并， $\Rightarrow$  状态最少；
- (3) 状态分配——状态变量编码  $\Rightarrow$  成本最低；
- (4) 组合电路部分综合，实现状态迁移函数和输出函数。



## 状态最小化

### ◆ 完全规定时序机

- 目标：时序机  $m \Rightarrow m'$ ;

- $m'$ 与 $m$ 等价;

- $m'$ 状态总数最小。

- 方法：合并等价状态

最大等价类

### ◆ 不完全规定时序机的状态化简

- 允许对未指定的状态和值任意指定：

增加难度，提高质量

- 方法：对完全规定时序机算法的扩展：

等价      相容



## 4.9.2 完全规定时序机状态最小化

### -- 等价状态 --

#### ◆ 等价状态对:

设状态 $s_i, s_j$  是某时序机状态集合 $S$ 中的两个状态, 分别对其施加各种可能的输入组合 $I_p$ , 若同时满足以下条件:

(1) 输出完全相同, 即

$$Z(s_i, I_p) = Z(s_j, I_p) \quad (I_p \in I, s_i, s_j \in S)$$

(“=”表示相同)

(2) 后继状态等价, 即

$$N(s_i, I_p) = N(s_j, I_p) \quad (I_p \in I, s_i, s_j \in S)$$

(“=”表示等价)

则称 $s_i$  与 $s_j$  等价, 它们是等价“状态对”。



## 等价状态类

### ◆ 等价类：

状态集合的一个子集  $S_k = \{ s_0, s_1, \dots, s_i, \dots, s_t \}$

若初始状态处于其中任何一个，在任意可能的输入序列的作用下，其输出序列都相同，则

$\{ s_0, s_1, \dots, s_i, \dots, s_t \}$  是一个等价类。

### ◆ 等价状态的性质：

(1) 可传递性：若  $s_i = s_j$  且  $s_j = s_k$

则  $s_i = s_k$

(2) 若  $s_0, s_1, \dots, s_i, \dots, s_t$  都等价于  $s_i$ ，  
则  $\{ s_0, s_1, \dots, s_i, \dots, s_t \}$  是一个等价类。

### ◆ 目标：寻找最大等价类。

- 不被其它等价类所包含的等价类称为最大等价类；
- 一个等价类可合并为一个状态。



## 状态划分

### ◆ 状态划分的定义：

设  $S_1, S_2, \dots, S_i, \dots, S_m$  都是状态集合  $S$  的子集，若同时满足以下条件：

$$S_i \cap S_j = \emptyset \quad (i \neq j)$$

$$S_1 \cup S_2 \cup \dots \cup S_i \dots \cup S_m = S$$

则集合  $\pi_k = \{S_1, S_2, \dots, S_i, \dots, S_m\}$  是  $S$  的一个划分。并称  $S_i$  是  $\pi_k$  的一个分组或类（Class, Block）。

例：

$s = \{a, b, c\}$  的可能划分：

$$\pi_1 = \{(a), (b), (c)\} \quad \text{0划分}$$

$$\pi_2 = \{(a, b), (c)\}$$

$$\pi_3 = \{(a), (b, c)\}$$

$$\pi_4 = \{(a, c), (b)\}$$

$$\pi_5 = \{(a, b, c)\}$$

单位划分I

$\pi_1, \pi_5$  为无效划分



## 置换性划分

### ◆ 定义：

对状态集 $S$ 的某一个划分 $\pi$ ,

若其任一个类 $s_i$ 中所有状态在输入 $i$ 取值的任一个组合作用下的后继状态集合 $s_t$ 被 $\pi$ 中某一个类所包含,

则称 $\pi$ 是一个置换性划分。

(闭合性)

### ◆ 对本例：

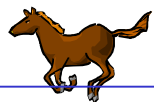
$$S=\{a, b, c, d, e\}$$

$$\pi_1=\{(a, b), c, (d, e)\}$$

$$=\{s_1, s_2, s_3\}$$

### ◆ 是置换性划分？ 是！（见下页）

state	in	next	out
a	0	b	0
a	1	c	0
b	0	a	0
b	1	c	0
c	0	a	0
c	1	d	0
d	0	a	1
d	1	e	1
e	0	a	1
e	1	d	1

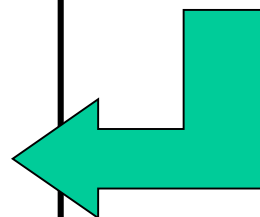


## 置换性划分例1: $s=\{a, b, c, d, e\}$

$\pi_1 = \{(a, b), c, (d, e)\}$   
=  $\{s_1, s_2, s_3\}$  是置换性划分? 是!

当前状态 state		输入信 号 in	后继状态 next	
s1	a	0	b	s1
	b		a	
s1	a	1	c	s2
	b		c	
s2	c	0	a	s1
s2	c	1	d	s3
s3	d	0	a	s1
	e		a	
s3	d	1	e	s3
	e		d	

原始描述			
state	in	next	out
a	0	b	0
a	1	c	0
b	0	a	0
b	1	c	0
c	0	a	0
c	1	d	0
d	0	a	1
d	1	e	1
e	0	a	1
e	1	d	1



次态落入同一类中!



## 置换性划分例2: $s=\{a, b, c, d, e\}$

◆  $\pi_2=\{(a, b, c), (d, e)\}=\{s_1, s_2\}$  不是置换性划分。

当前状态 state		输入 in	次态 next	
s1	a	1	c	s1
	b	1	c	
	c	1	d	s2

状态s1在输入信号 in 的作用下，其次态没有落入同一个类中！

原始描述			
state	in	next	out
a	0	b	0
a	1	c	0
b	0	a	0
b	1	c	0
c	0	a	0
c	1	d	0
d	0	a	1
d	1	e	1
e	0	a	1
e	1	d	1





## 输出一致的划分

### ◆ 输出一致的划分：

对状态集 $S$ 的某一个划分 $\pi$ ，若其任一个类 $S_i$ 中所有状态在输入 $I$ 取值的任一个组合 $I_p$ 作用下的输出都相同，则称 $\pi$ 是一个输出一致的划分。



## 输出一致的划分举例

◆  $\pi_1 = \{(a, b), (c), (d, e)\}$   
 $= \{s_1, s_2, s_3\}$

当前状态 state		输入信 号 in	输出信 号 out
s1	a	0	0
	b	0	
s1	a	1	0
	b	1	
s2	c	0	0
s2	c	1	0
s3	d	0	1
	e	0	
s3	d	1	1
	e	1	

•  $\pi_2 = \{(a, b, c), (d, e)\}$   
 $= \{s_1, s_2\}$

当前状态 state		输入信 号 in	输出信 号 out
s1	a	0	0
	b		
	c		
s1	a	1	0
	b		
	c		
s2	d	0	1
	e		
s2	d	1	1
	e		



## 输出一致的置换性划分

- ◆ 输出一致的置换性划分：
  - 既是置换性划分（闭合性）；
  - 又是输出一致的划分。
- ◆ 输出一致的置换性划分中的类是等价类，可以合并。



## 等价类最小划分的逐次分割法

### -- k等价划分法 --

1. 令  $K := 1$ , 按**输出一致**的要求进行**划分**, 形成  $\pi_k$  (即  $\pi_1$ ) ;
2. 令  $K := K + 1$  ;  $\pi_k := \pi_{k-1}$  ;  
按置换性要求对  $\pi_k$  的各个类进行**(闭合性)**划分, 对每一个类  $S_i \in \pi_k$

施加每一个可能的输入组合  $I_p$ , 若其次态落入  $\pi_k$  同一类者形成一个新类。用这些新类置换原来的类, 形成新的划分  $\pi_k$ 。

3. 若  $\pi_k = \pi_{k-1}$ , 则  $\pi_k$  即为所求结果, 结束。否则, 转2。

## 例：对 $s=\{a, b, c, d, e\}$ 作 $k$ 等价划分

原始描述			
state	in	next	out
a	0	b	0
a	1	c	0
b	0	a	0
b	1	c	0
c	0	a	0
c	1	d	0
d	0	a	1
d	1	e	1
e	0	a	1
e	1	d	1

- $\pi_1 = \{(a, b, c), (d, e)\}$  (输出一致)
- 对  $\pi_1$  中的各个类进行(闭合性)划分：
  - 划分(a, b, c):
    - $n(a,0)=b, n(b,0)=n(c,0)=a$ :  
落入  $\pi_1$  中的(a, b, c),
    - $n(a,1)=c, n(b,1)=c$ : 落入 (a, b, c),
    - $n(c,1)=d$ : 落入 (d, e)。
  - $\therefore (a, b, c)$  的次态没有落入同一组,
    - (a, b) 的次态落入同一组;
    - (c) 的次态落入同一组;
  - $\therefore (a, b, c) \Rightarrow (a, b), (c)$ ;
- 划分(d, e):
  - $n(d,0)=n(e,0)=a$ : 落入  $\pi_1$  中的(a, b, c)
  - $n(d,1)=e, n(e,1)=d$ : 落入  $\pi_1$  中的(d, e)
  - $\therefore (d, e)$  满足置换性要求, 不再划分;
- $\therefore \pi_2 = \{(a, b), (c), (d, e)\}$



3.

$\therefore \pi_2 \neq \pi_1$ ,

继续检查划分  $\pi_2 = \{(a, b), (c), (d, e)\}$  中的各个类:

✓ 划分(a, b):

$n(a, 0) = b, n(b, 0) = a$ : 落入 (a, b),

$n(a, 1) = c, n(b, 1) = c$ : 落入 (c),

(a, b) 满足置换性要求, 不再划分;

✓ 划分(c) 只有一个状态, 不需划分;

✓ 划分(d, e):

$n(d, 0) = n(e, 0) = a$ : 落入  $\pi_2$  中的(a, b)

$n(d, 1) = e, n(e, 1) = d$ : 落入  $\pi_2$  中的(d, e)

(d, e) 满足置换性要求, 不再划分;

$\therefore \pi_3 = \pi_2 \quad \therefore \pi_2$  为等价性划分。

原始描述			
state	in	next	out
a	0	b	0
a	1	c	0
b	0	a	0
b	1	c	0
c	0	a	0
c	1	d	0
d	0	a	1
d	1	e	1
e	0	a	1
e	1	d	1



### 4.9.3 不完全规定时序机的状态化简

- ◆ 次态出现未明确指定的状态( $\emptyset$ ), 或输出出现未明确指定的值(u)。
- ◆ 允许综合时任意指定其状态和值。
  - 增加难度
  - 有可能提高质量
- ◆ 方法：对完全规定时序机的算法的扩展：  
等价 $\Rightarrow$ 相容



## 相容状态的概念 (compatible state)

◆ 定义:

- $s_i$ 与 $s_j$ 相容, 当且仅当对每一个可能的输入序列 $I_p$ , 无论该时序机的初始状态为 $s_i$ 或 $s_j$ , 在状态表上输出规定明确的地方都产生相同的输出序列。





## 相容类术语

- ◆ 相容状态对 ( $s_1, s_2$ ): 任意可能输入组合下:
  - 输出相容: 在状态表规定明确的地方输出相同;
  - 次态相容。
- ◆ 相容类: 状态集合子集, 其中任意2个状态皆为相容对。
- ◆ 最大相容类: 不被其他相容类包含的相容类。



## 相容类术语(续)

- 不相容状态对：不满足相容条件的状态对。
- 不相容类：不相容状态集合的子集。其中任意2个状态皆为不相容对。
- 最大不相容类：不被任何其他不相容类所包含。



## 相容状态对的产生

- ◆ 状态对集合  $S = \{ \text{所有可能的状态对} \}$   
 $= \{ C \cup D \}$

其中C为相容状态对集合；D为不相容状态对集合

- ◆ 方法：挑出不相容状态对，剩下为相容状态对。
- ◆ 算法：
  1. 检查每个状态对，将输出相容对放在集合 C 中，输出不相容对放在集合 D 中；
  2. 检查 C 中的每个状态对，将次态不相容者移入D；



## 求相容状态对举例

1. 按**输出一致**检查每个状态对:

$C = \{(s1, s2), (s1, s3), (s1, s4), (s1, s5),$   
 $(s2, s3), (s2, s4), (s3, s4), (s4, s5)\}$

$D = \{(s2, s5), (s3, s5)\},$

2. 检查C 中的每个状态对, 将**次态不相容** (次态对已在D中) **者移入D**:

对于状态对 **(s2, s4)**:

$n(s2, 0) = s5,$

$n(s4, 0) = s3,$

$\therefore (s3, s5)$  不相容,

$\therefore D = \{(s2, s5), (s3, s5), (s2, s4)\}$

$\begin{array}{c} x \\ s \end{array}$	0	1
s1	s4/u	s1/u
s2	s5/0	s1/u
s3	s4/0	s2/u
s4	s3/u	s3/u
s5	s3/1	s2/u



## 相容关系的性质

### ◆ 分析：

- (s1,s3) 是相容状态对；
- (s1, s5) 是相容状态对；

### ◆ 但是

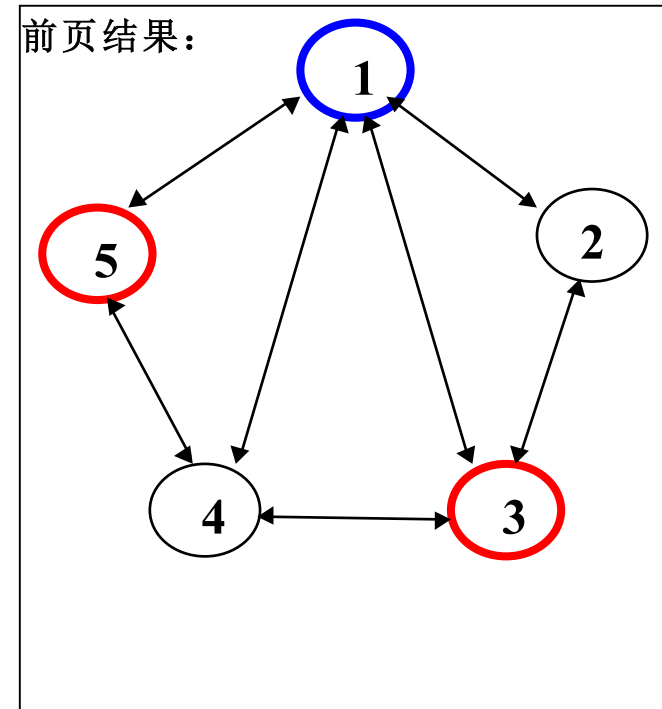
- (s3, s5)是不相容状态对；

### ◆ 相容关系不具有可传递性！

### ◆ → 相容类划分：

$$\pi_1 = \{ (s1, s2, s3 ), ( s4, s5) \}$$

$$\pi_2 = \{ ( s2, s3 ), ( s1, s4, s5) \}$$





## 用团划分法使不完全规定时序机 状态最小化

1. 最大不相容类中的每一个状态都是一个**生长点**；
2. 以各**生长点**为**核心**，以满足**相容**为原则进行**结群**，形成**团**；
3. 团必须满足**封闭性**（对于每一个团，在每一输入组合下的次态必须落在某一个团中）；否则返回2；
4. 若还存在没有被包含进来的状态，则从剩余状态中找出**新的生长点**，继续结群（**相容性+封闭性**）。
5. 重复执行，直到所有状态均被结群。



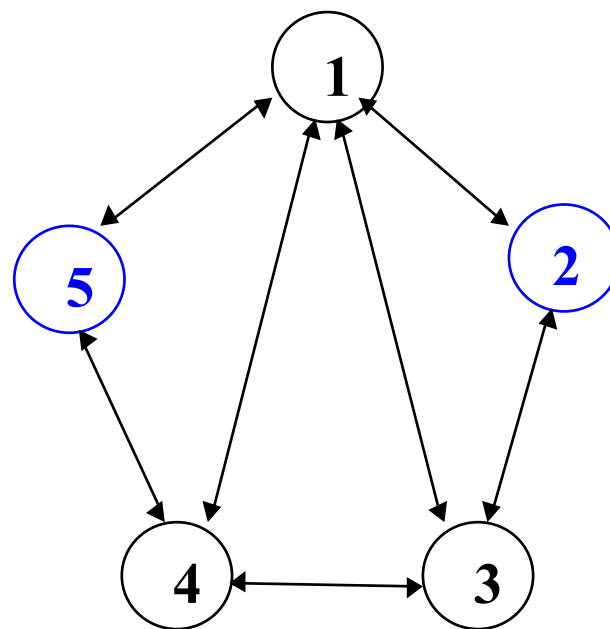
## 团划分算法举例

### 1. 求出最大不形容类：

任选不相容状态对 (s2, s5),  
 $C1=\{s2\}$ ,  $C2=\{s5\}$ , 构成二个生长点。

- 另一不相容状态对(s2, s4),  
因s4与s5相容, 不构成生长点。
- 另一不相容状态对(s3, s5),  
因s3与s2相容, 不构成生长点。

$\therefore \pi = \{C1, C2\} = \{(s2), (s5)\}$ , 是最大不相容类;  
剩余状态有s1, s3, s4  $\rightarrow$  结群





## 团划分算法举例（续）

前页结果，生长点为：  $C1 = \{s2\}$ ,  $C2 = \{s5\}$

### 2. 生长：对各剩余状态：

$s1$ : 与  $C1$  相容,  $C1 = \{s1, s2\}$ ,

$s3$ : 与  $C1$  相容,  $C1 = \{s1, s2, s3\}$

$s4$ : 与  $C2$  相容,  $C2 = \{s4, s5\}$

剩余状态为空,

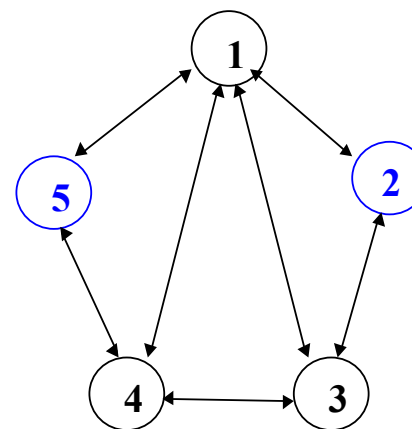
$\therefore \pi = \{C1, C2\} = \{(s1, s2, s3), (s4, s5)\}$

→ 检查封闭性：满足，

→ 结群过程结束；

### 3. 合并状态：

$\pi = \{C1, C2\} = \{(s1), (s4)\}$



$s \backslash x$	0	1
s1	s4/u	s1/u
s2	s5/0	s1/u
s3	s4/0	s2/u
s4	s3/u	s3/u
s5	s3/1	s2/u



## 4.9.4 时序机的状态分配: 状态编码



- ◆ 要实现一个时序机，必须给状态编码。同时希望成本低。
- ◆ 使成本低的思路：寄存器长度最短，组合逻辑成本最低。  
(有时，寄存器加长，组合逻辑电路成本更低)。
- ◆ 相邻编码法（启发性算法）：
  1. 某些状态在同一输入条件下次态相同，将这些状态分配以相邻的编码。
    - 何谓相邻？ 2个代码只在1位上取值不同。
  - 即若 $n(s1, i) = n(s2, i)$ , 则 $s1 \text{ adj } s2$ .
  2. 某一状态的逻辑相邻的输入对应的次态分配以相邻的编码。
    - 即设 $n(s1, i1) = s3$ ,  $n(s1, i2) = s4$ ,
    - 若 $i1 \text{ adj } i2$ , 则 $s3 \text{ adj } s4$ .



## 状态分配举例

1. 初始：将状态1的编码分配为00。
2.  $n(1, 0) = n(4, 0) = 3$ ,  $n(1, 1) = n(4, 1) = 4$ ,  
 $\therefore 1 \text{ adj } 4$ ,  
 $\rightarrow$  分配状态4的编码为01 (规则1)
3. 以状态4为参考状态, 其后继状态分别为3和4,  
 即  $3 \text{ adj } 4$ ,  $\therefore$  分配状态3的编码为11, 与状态4的编码01 相邻。(规则2)
4.  $n(2, 1) = n(3, 1) = 1$   
 $\therefore 2 \text{ adj } 3$   
 $\rightarrow$  分配状态2的编码为10 (规则1), 与状态3的编码11 相邻。

$\begin{smallmatrix} & x \\ s & \end{smallmatrix}$	0	1
1	3	4
2	1	1
3	2	1
4	3	4



$\begin{smallmatrix} & x \\ s & \end{smallmatrix}$	0	1
1	00	01
2	10	00
3	11	00
4	01	01



# 状态分配的启发式算法

## -- 按权分配法 --

### ◆ 原则:

- 根据相邻关系规定状态对 $(s_i, s_j)$ 的权 $w(s_i, s_j)$ ;
- 权值大者优先分配相邻的编码。

### ◆ 权值的规定: $w(s_i, s_j) = w_1 + w_2 + w_3 + w_4$ (状态总数= $n$ )

1. 若有 $p_1$ 个输入组合其次态相同, 则  $w_1 = p_1 \cdot \lceil \log n \rceil$
2. 若有 $p_2$ 个输入组合其输出相同, 则  $w_2 = p_2$
3. 若有 $p_3$ 个状态, 其中每一个在两个相邻输入组合下其次态分别为 $s_i, s_j$ , 则  $w_3 = p_3$
4. 若有 $p_4$ 个输入组合, 其次态仍然落在该两个状态中, 则  $w_4 = p_4 \cdot (\lceil \log n \rceil - 1)$





## 按权分配法举例

S	X		Z
	0	1	
s1	s1	s2	0
s2	s3	s4	0
s3	s2	s1	1
s4	s1	s4	1

状态对	w1	w2	w3	w4	w
s1, s2	0	2	2	0	4
s1, s3	0	0	0	0	0
s1, s4	2	0	1	0	3
s2, s3	0	0	0	1	1
s2, s4	2	0	0	0	2
s3, s4	0	2	1	0	3

优先级: (s1, s2), (s1, s4), (s3, s4),  
(s2, s4), (s2, s3), (s1, s3)

状 态 分 配 :

1. (s1, s2): s1: 00, s2: 01
2. (s1, s4): s4: 10
3. s3: 11

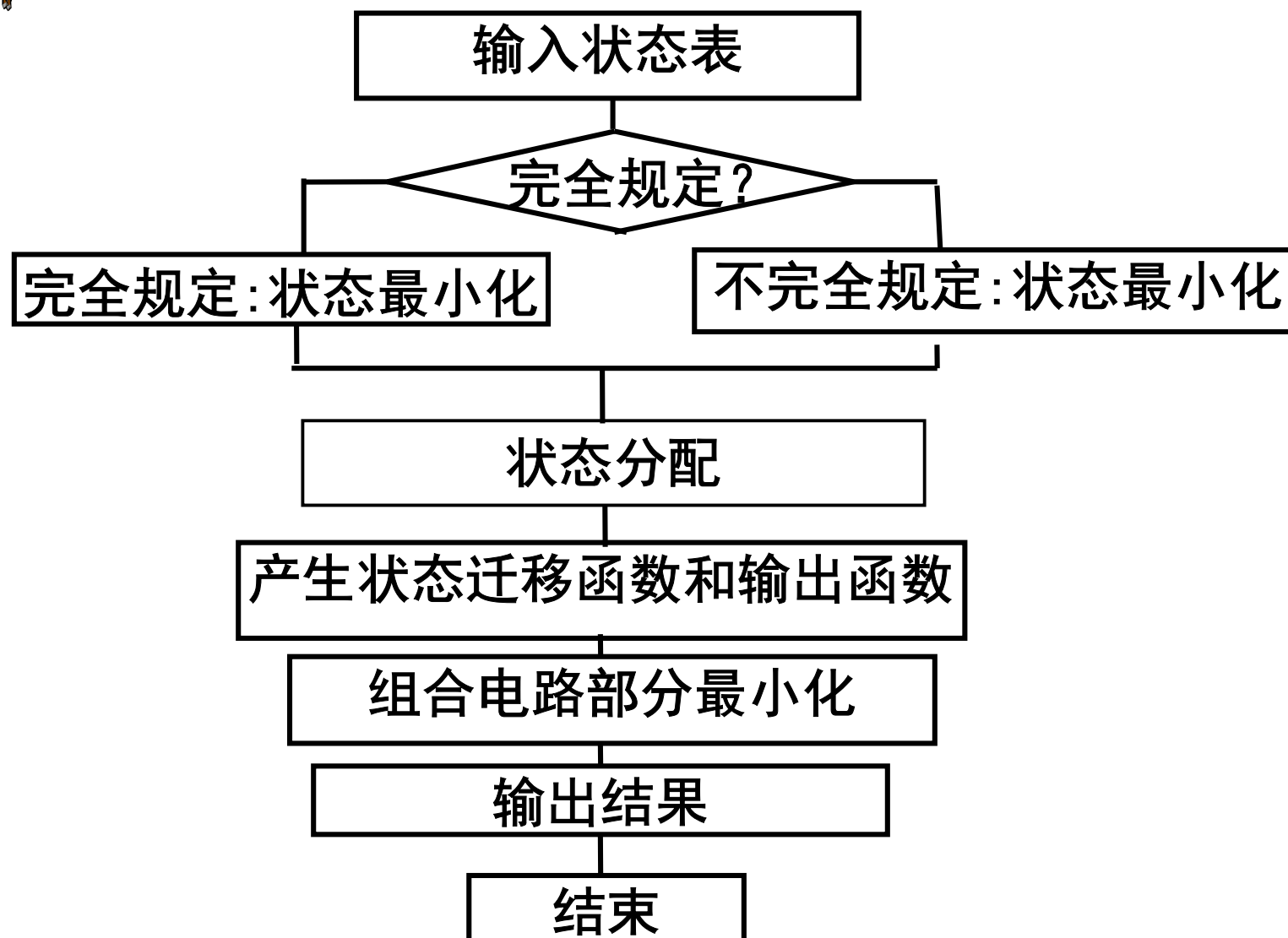
S1	00
S2	01
S3	11
S4	10





# 小结

--时序逻辑电路自动综合过程

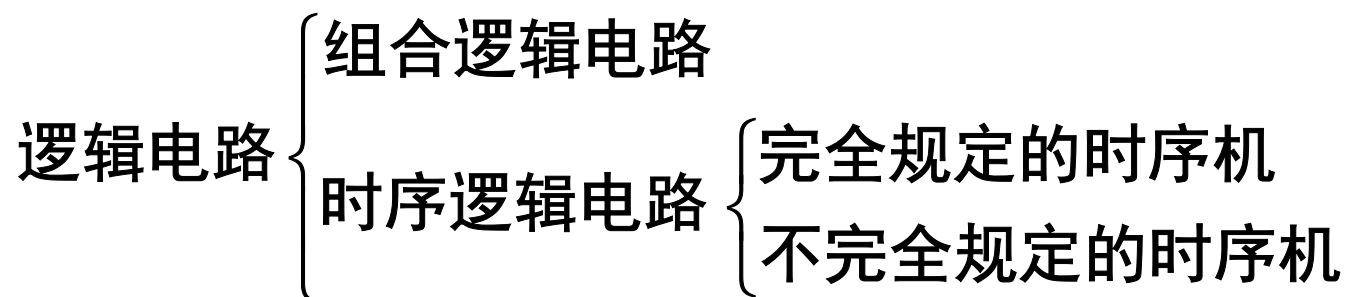




## 本章小结

- ◆ 本章的主要目标是研究如何开发逻辑综合软件。
  - 逻辑综合软件的主要任务是：根据设计者的逻辑功能描述及约束条件（速度、功耗、成本、器件类型...），导出满足上述要求的逻辑电路。

- ◆ 逻辑电路：



- ◆ 时序电路由记忆元件和组合逻辑电路组成，所以组合逻辑电路的综合是整个逻辑综合的基础。



## 本章小结（续）

### ◆ 本章的主要内容：。

- 逻辑综合的基本概念。
- 布尔函数的立方体表示法及立方体的各种运算。
- 单输出（以及多输出）函数质立方体的计算。
- 单输出函数的综合（求最小化覆盖）。
- 多输出函数的综合
- 时序电路的逻辑综合