

计算机组成

# 从门电路到运算

(2025秋)

高小鹏

北京航空航天大学计算机学院

## 同步数字电路分类

- 组合逻辑 (CL, Combinational Logic)
  - ◆ 输出只是输入的函数, 与历史状态无关
  - ◆ 例如: 加法器
- 时序逻辑 (SL, Sequential Logic)
  - ◆ 电路能存储信息
  - ◆ 例如: 存储器、寄存器

synchronous/同步 memory/存储器 register/寄存器

## 逻辑门<sup>1/2</sup>

- 名字、符号、功能
- 真值表：描述电路输入与输出之间的对应关系

圆圈：代表NOT

NOT

```
graph LR; a --- NOT1[NOT]; NOT1 --- bubble(( )); bubble --- c;
```

| a | c |
|---|---|
| 0 | 1 |
| 1 | 0 |

AND

```
graph LR; a --- AND1[AND]; b --- AND1; AND1 --- c;
```

| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR

```
graph LR; a --- OR1[OR]; b --- OR1; OR1 --- c;
```

| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |


3


 北京航空航天大学计算机学院  
 School of Computer Science and Engineering, Beihang University

## 逻辑门<sup>2/2</sup>


- 名字、符号、功能
- 真值表：描述电路输入与输出之间的对应关系

NAND




| a | b | c |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NOR



| a | b | c |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

XOR



| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

4

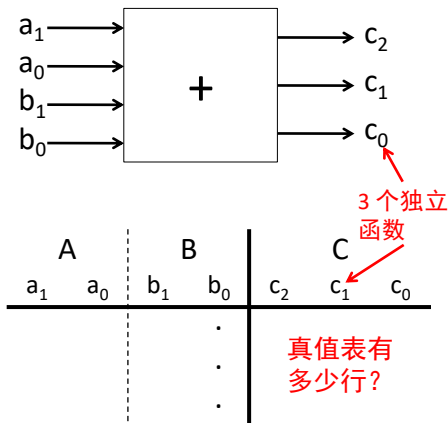

 北京航空航天大学计算机学院  
 School of Computer Science and Engineering, Beihang University

## 更复杂的真值表

□ 3输入：表决器

□ 2位加法器

| a | b | c | y |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |



5

## 从真值表到布尔表达式

□ 根据变量取值确定表达式中的变量表示

- ◆ 变量值为1：变量名
- ◆ 变量值为0：变量名的NOT

□ 方法1：积项求和（SoP, Sum of Products）

- ◆ 输出值为1的行产生积项；然后把所有项OR起来
- ◆  $c = \bar{a}b + a\bar{b}$

□ 方法2：和项求积（PoS, Product of Sums）

- ◆ 输出值为0的行产生和项；然后把所有项AND起来
- ◆  $c = (\bar{a} + \bar{b}) \cdot (a + b)$

| a | b | c |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

6

## 布尔代数的基本定律

□ 这些基本定律常用于布尔表达式的化简

|                                            |                                            |         |
|--------------------------------------------|--------------------------------------------|---------|
| $x \cdot \bar{x} = 0$                      | $x + \bar{x} = 1$                          | 互补性     |
| $x \cdot 0 = 0$                            | $x + 1 = 1$                                | 0-1律    |
| $x \cdot 1 = x$                            | $x + 0 = x$                                | 重言律     |
| $x \cdot x = x$                            | $x + x = x$                                | 幂等律     |
| $x \cdot y = y \cdot x$                    | $x + y = y + x$                            | 交换律     |
| $(xy)z = x(yz)$                            | $(x + y) + z = x + (y + z)$                | 结合律     |
| $x(y + z) = xy + xz$                       | $x + yz = (x + y)(x + z)$                  | 分配律     |
| $xy + x = x$                               | $(x + y)x = x$                             | 吸收律     |
| $\bar{x}y + x = x + y$                     | $(\bar{x} + y)x = xy$                      | 吸收律 v.2 |
| $\overline{x \cdot y} = \bar{x} + \bar{y}$ | $\overline{x + y} = \bar{x} \cdot \bar{y}$ | 德·摩根律   |

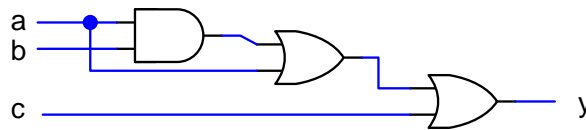
7



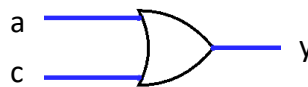
北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University

## 布尔表达式化简示例

□  $y = ab + a + c$



$$\begin{aligned}
 y &= ab + a + c \\
 &= a(b + 1) + c \\
 &= a(1) + c \\
 &= a + c
 \end{aligned}$$

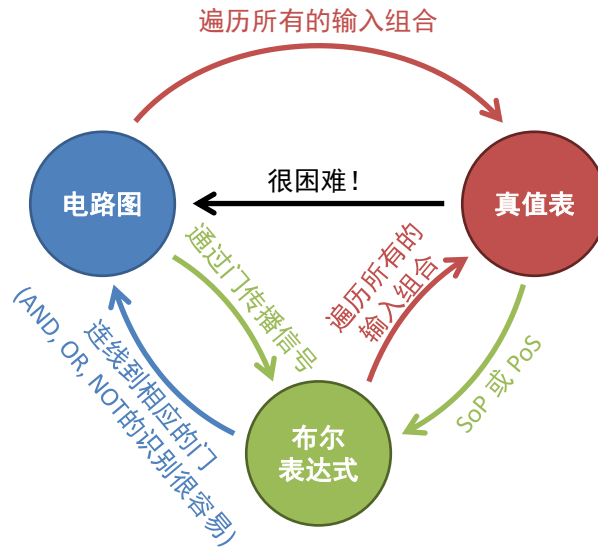


8



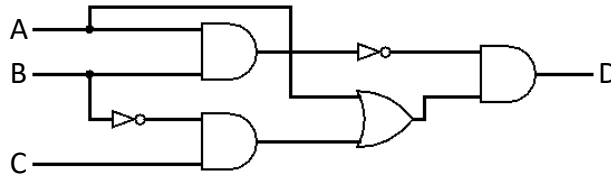
北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University

## 组合逻辑的转换



9

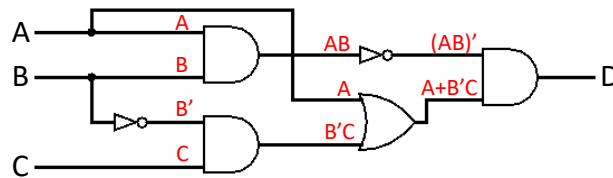
## 电路化简示例<sup>1/4</sup>



- 方法1: 遍历所有输入组合建立真值表, 然后运用SoP或PoS化简
- 方法2: 根据电路写出对应的表达式, 然后运用公式化简
  - ◆ 下面展示方法2

10

## 电路化简示例<sup>2/4</sup>



- 按从输入到输出的方向，逐层建立表达式
- $D = \overline{AB} \cdot (A + \bar{B}C)$

11

## 电路化简示例<sup>3/4</sup>

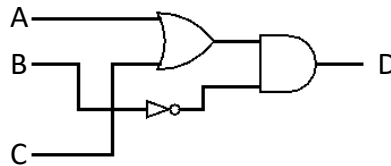
$$\begin{aligned}
 D &= (AB)'(A + B'C) \\
 &= (A' + B')(A + B'C) && \text{德摩根律} \\
 &= A'A + A'B'C + B'A + B'B'C && \text{分配律} \\
 &= 0 + A'B'C + B'A + B'B'C && \text{互补律} \\
 &= A'B'C + B'A + B'C && \text{幂等律} \\
 &= (A' + 1)B'C + AB' && \text{分配律} \\
 &= B'C + AB' && \text{1律} \\
 &= B'(A + C) && \text{分配律}
 \end{aligned}$$

12

## 电路化简示例<sup>4/4</sup>

- $D = (AB)'(A + B'C)$ 
  - ◆ Q: 多少个门?
- $D = B'C + AB' = B'(A + C)$ 

4
3
- 画出电路



13

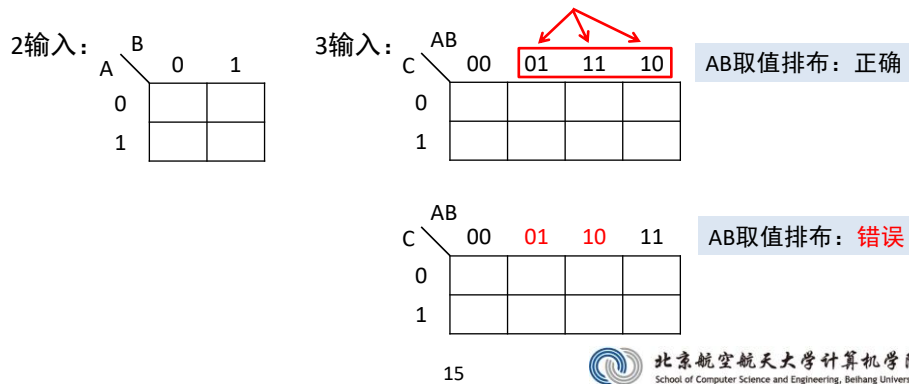
## 卡诺图

- 卡诺图 (Karnaugh Maps) 是另一种简化布尔表达式的方法
  - ◆ 适用于求解输入总数比较少的表达式, 例如输入为5个以内
  - ◆ 更多信息请参考: [http://en.wikipedia.org/wiki/Karnaugh\\_map](http://en.wikipedia.org/wiki/Karnaugh_map)
- 卡诺图背后的思路
  - ◆ 使用SoP把相邻项化简掉1个输入
    - 相邻项: 只有1个输入信号不同
    - 例如:  $ab + a'b = b$ ,  $a'bc + a'bc' = a'b$

14

## 卡诺图的基本构图方式

- 把输入信号尽可能分成均匀的2组（一组n个信号，一组m个信号）
  - ◆ 例如：4输入分成2-2，5输入分成2-3
- 构造一个 $2^n \times 2^m$ 的矩阵
  - ◆ n个信号有 $2^n$ 个取值组合；m个信号有 $2^m$ 个取值组合
- 组合值排布的规则：相邻的2个组合只有1个信号的值有变化



北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University

## 卡诺图化简示例：3输入表决器<sup>1/2</sup>

- 第1步：把真值表填入卡诺图
  - ◆ 真值表的行：对应卡诺图的单元格
  - ◆ 填表时要仔细，否则在从011到100这样的地方就容易犯错误

| a | b | c | y |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

|   |    |    |    |    |    |
|---|----|----|----|----|----|
|   | ab | 00 | 01 | 11 | 10 |
| c | 0  | 0  | 0  | 1  | 0  |
| 1 |    | 0  | 1  | 1  | 1  |



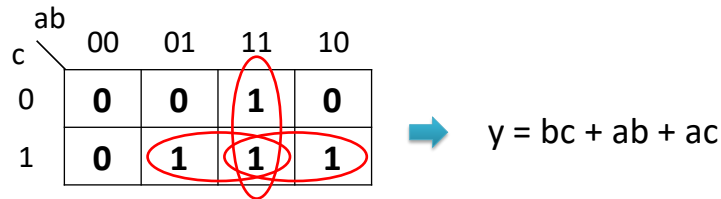
北京航空航天大学计算机学院  
School of Computer Science and Engineering, Beihang University



## 卡诺图化简示例：3输入表决器<sup>2/2</sup>

- 相邻的邻居如果均为1，则可以化简：将变化的信号化简

- ◆ 例如：110和111，其对应的表达式为 $abc' + abc$ ，可以优化为 $ab$



- 约束：邻居的个数必须是2的整数幂

- ◆ 例如：右半部分4个单元格，由于100不是1，因此就不满足约束

- 组越大，则最终表达式约简单

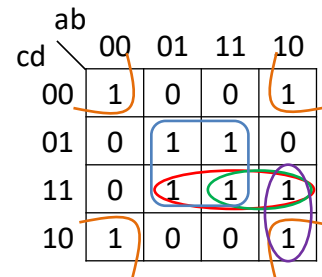
- ◆ 例如：如果最下边4个都是1，则化简为 $c$

Q  
111项被多次使用。为什么这是正确的？

17

## 卡诺图使用的一般性规则

- 组内有效单元的个数必须是2的整数幂
- 应该选择尽可能大的组
- 注意4角的单元
- 避免出现单个单元



- 1) 不合法
- 2)  $bd$ ：有效
- 3)  $b'd'$ ：角
- 4) 1011有多种选择

答案1:  $y = bd + b'd' + acd$

答案2:  $y = bd + b'd' + ab'c$

18

## 提纲

- 门电路
- 运算
  - 加法、减法、乘法、除法

北京航空航天大学计算机学院

## 1位加法器

- 利用真值表构造1位加法器的表达式

|   |       |       |       |       |
|---|-------|-------|-------|-------|
|   | $a_3$ | $a_2$ | $a_1$ | $a_0$ |
| + | $b_3$ | $b_2$ | $b_1$ | $b_0$ |
|   | $s_3$ | $s_2$ | $s_1$ | $s_0$ |

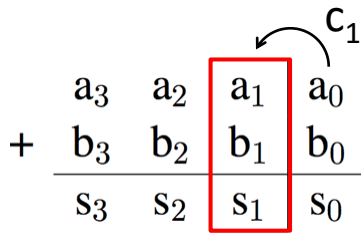
|       |       | 进位输出  |       |
|-------|-------|-------|-------|
| $a_0$ | $b_0$ | $s_0$ | $c_1$ |
| 0     | 0     | 0     | 0     |
| 0     | 1     | 1     | 0     |
| 1     | 0     | 1     | 0     |
| 1     | 1     | 0     | 1     |

$$s_0 = \overline{a_0}b_0 + a_0\overline{b_0}$$

$$c_1 = a_0b_0$$



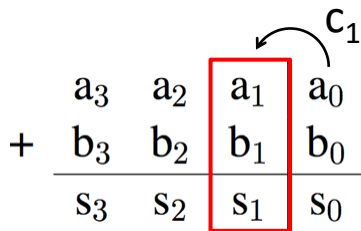
## 1位加法器



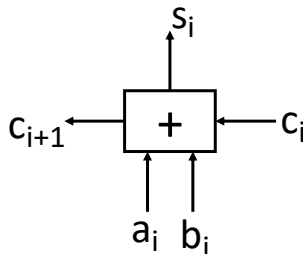
| $a_i$ | $b_i$ | $c_i$ | $s_i$ | $c_{i+1}$ |
|-------|-------|-------|-------|-----------|
| 0     | 0     | 0     | 0     | 0         |
| 0     | 0     | 1     | 1     | 0         |
| 0     | 1     | 0     | 1     | 0         |
| 0     | 1     | 1     | 0     | 1         |
| 1     | 0     | 0     | 1     | 0         |
| 1     | 0     | 1     | 0     | 1         |
| 1     | 1     | 0     | 0     | 1         |
| 1     | 1     | 1     | 1     | 1         |

$$\begin{aligned} S_i &= \bar{A}_i \bar{B}_i C_i + \bar{A}_i B_i \bar{C}_i + A_i \bar{B}_i \bar{C}_i + A_i B_i C_i \\ C_{i+1} &= A_i B_i + A_i C_i + B_i C_i \end{aligned}$$

## 1位加法器



| $a_i$ | $b_i$ | $c_i$ | $s_i$ | $c_{i+1}$ |
|-------|-------|-------|-------|-----------|
| 0     | 0     | 0     | 0     | 0         |
| 0     | 0     | 1     | 1     | 0         |
| 0     | 1     | 0     | 1     | 0         |
| 0     | 1     | 1     | 0     | 1         |
| 1     | 0     | 0     | 1     | 0         |
| 1     | 0     | 1     | 0     | 1         |
| 1     | 1     | 0     | 0     | 1         |
| 1     | 1     | 1     | 1     | 1         |



$$\begin{aligned} S_i &= \overline{A_i} \overline{B_i} C_i + \overline{A_i} B_i \overline{C_i} + A_i \overline{B_i} \overline{C_i} + A_i B_i C_i \\ C_{i+1} &= A_i B_i + A_i C_i + B_i C_i \end{aligned}$$

## 用N个1位加法器构造N位加法器

□ 把 $C_i$ 与 $C_{i-1}$ 连接，形成进位链

- ◆ 串行加法器；行波进位加法器

Q  
如何处理 $C_0$ ?

