天津大學

数字逻辑与数字系统

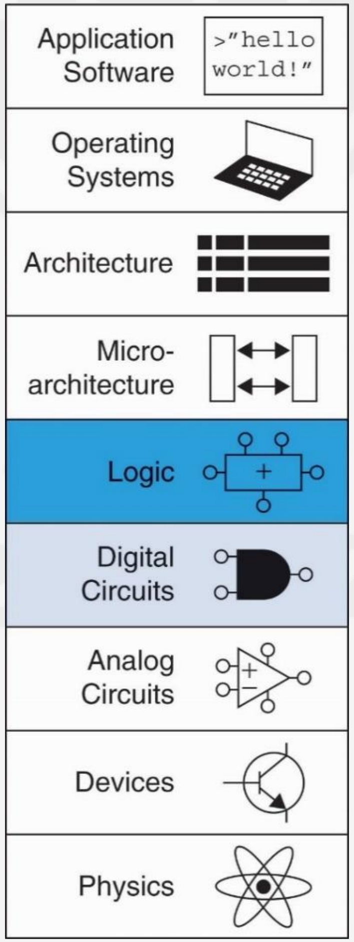
2 组合逻辑设计

Combinational Logic Design

本章内容

Topic

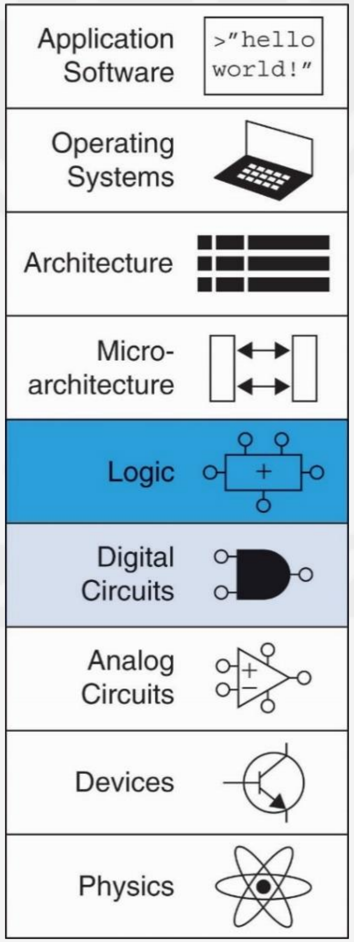
- 引言
- 布尔代数
- 卡诺图
- 从逻辑到门
- 多级组合逻辑
- X和Z
- 组合逻辑电路设计方法
- 组合逻辑中的时序问题
- 组合逻辑模块



本章内容

Topic

- 引言
- 布尔代数
- 卡诺图
- 从逻辑到门
- 多级组合逻辑
- X和Z
- 组合逻辑电路设计方法
- 组合逻辑中的时序问题
- 组合逻辑模块

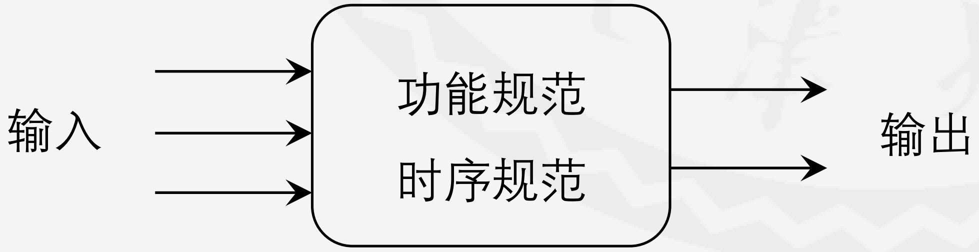


引言

Introduction

数字逻辑电路

- 数字逻辑电路（logic circuit）是一个可以处理离散值变量的网络
- 其中包括:
  - 一个或多个离散值输入端
  - 一个或多个离散值输出端
  - 描述输入和输出关系的功能规范
  - 描述当输入改变时输出响应延迟的时序规范

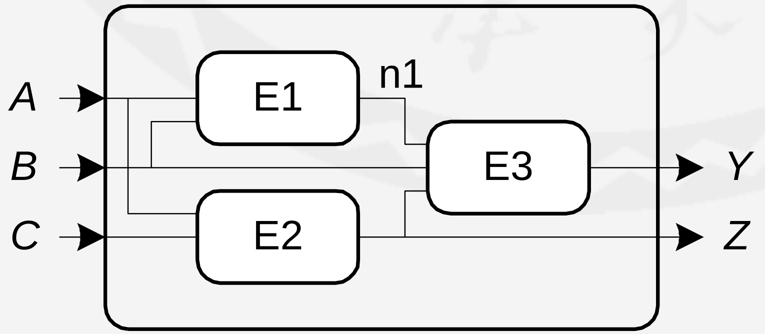


引言

Introduction

结点和模块

- 电路由结点（nodes）和模块（elements）组成
- 结点是一段导线，通过电压传递离散值变量
  - 输入结点：接收外部的值（图中的A，B，C）
  - 输出结点：输出值到外部（图中的Y，Z）
  - 内部结点：不属于以上两者的结点（图中的n1）
- 模块本身是一个带有输入、输出、功能规范和时序规范的电路
  - 每一个模块本身都是一个电路
  - 图中的E1，E2，E3

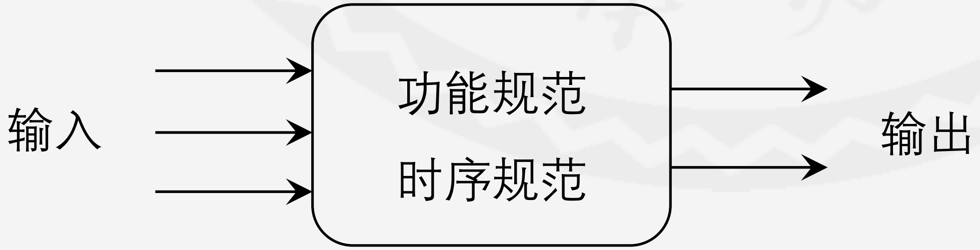


引言

Introduction

数字逻辑电路的分类

- 组合逻辑电路（combinational logic ）
  - 任一时刻的输出仅由该时刻的输入信号决定
  - 无记忆的，与电路状态无关
- 时序逻辑电路（sequential logic）
  - 任一时刻的输出由该时刻的输入和电路该时刻的状态共同决定
  - 有记忆的，与电路状态有关

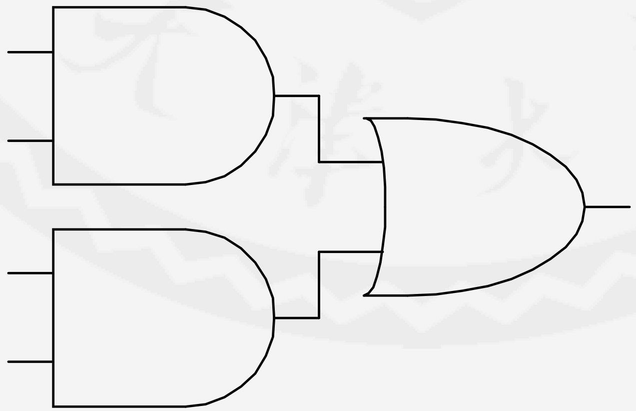


引言

Introduction

组合逻辑电路

- 每个电路模块都是一个组合逻辑电路
- 每个电路结点：
  - 或者是电路的输入
  - 或者是只连接电路模块的一个输出端
- 电路中不包含回路

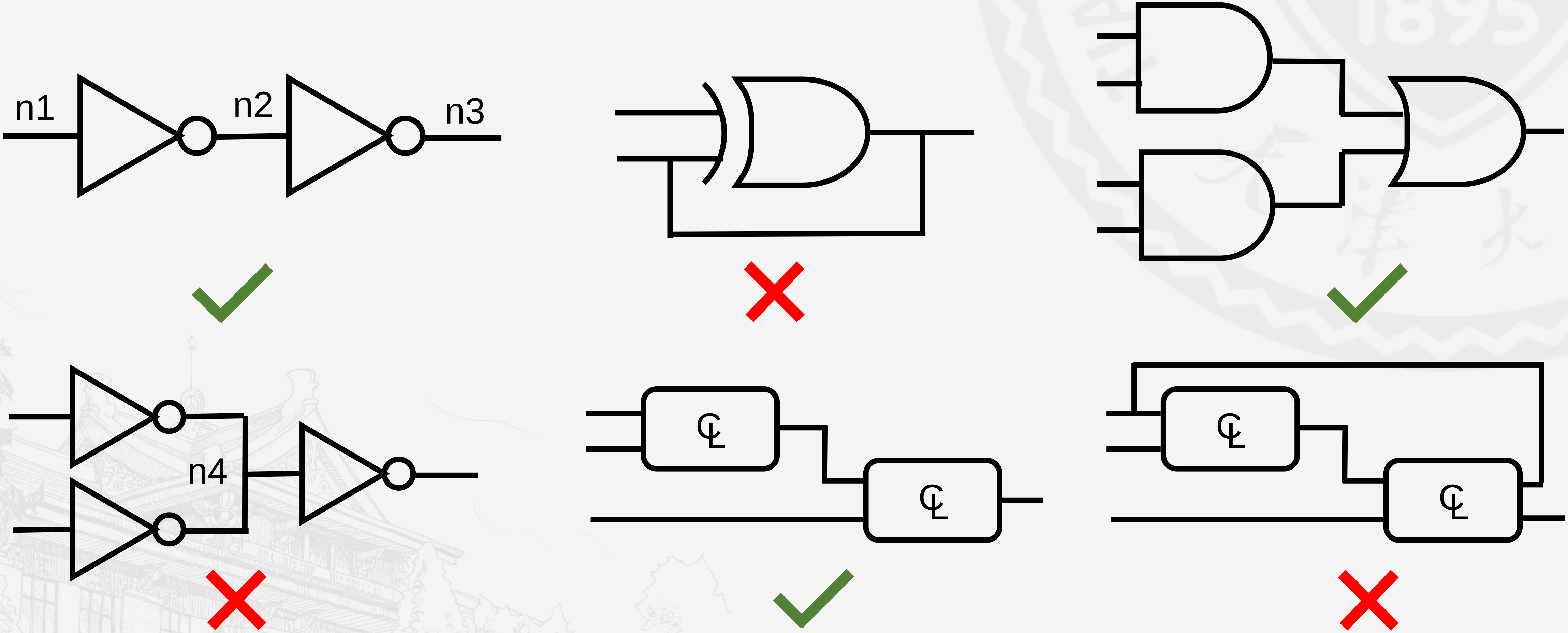


在本课程中我们使用 $\mathcal{C}\!\!\!\!\mathcal{L}$ 符号表示组合逻辑

引言

Introduction

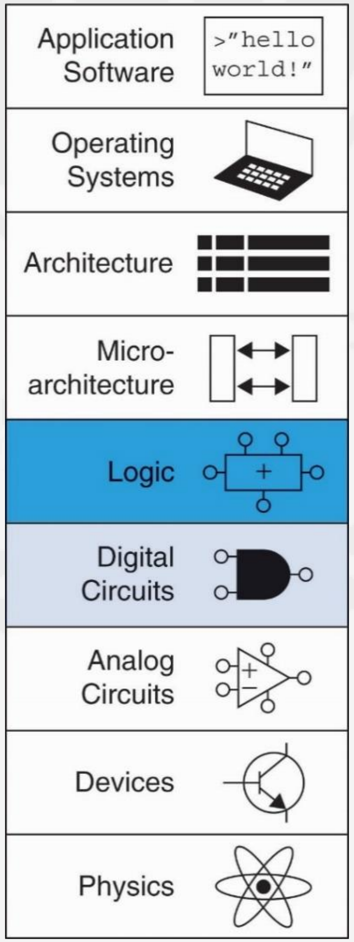
思考：下列哪些电路是组合逻辑电路?



本章内容

Topic

- 引言
- 布尔代数
- 卡诺图
- 从逻辑到门
- 多级组合逻辑
- X和Z
- 组合逻辑电路设计方法
- 组合逻辑中的时序问题
- 组合逻辑模块



布尔代数

Boolean Algebra

①基本概念

②公理

③定理

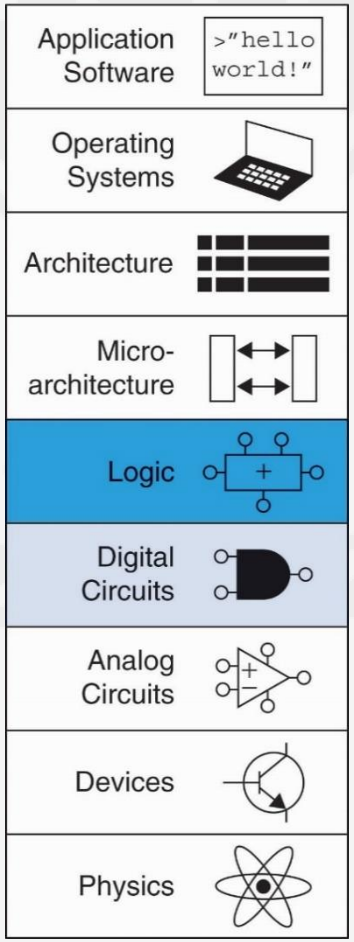
④最小项

⑤最大项

⑥标准与或式和标准或与式

⑦布尔表达式与真值表的转换

⑧使用定理化简表达式

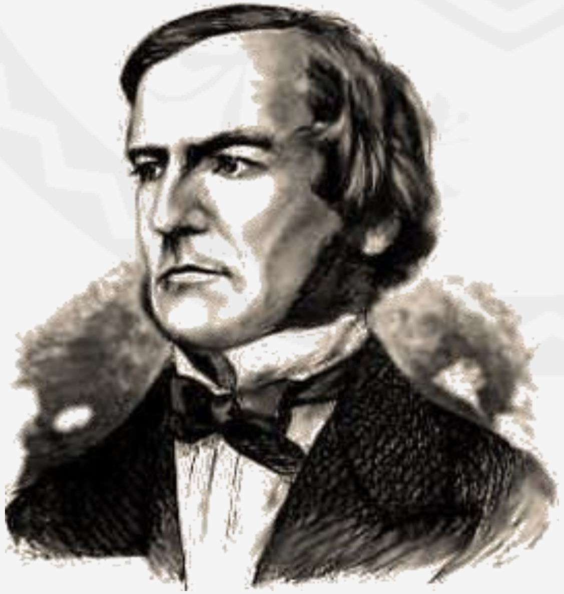


基本概念

Basic Concepts

乔治 · 布尔

- 乔治· 布尔（George Boole，1815~1864）
- 19世纪最重要的数学家之一
- 1854年，出版了 *The Laws of Thought* 一书
- 在这本书中，对布尔代数进行了全面的介绍
- 布尔代数是研究数字电路的数学基础



布尔代数的定义

- 布尔代数中的变量取值只能为“真”(TRUE) 或“假”(FALSE)
- “1” 表示真，“0”表示假
- 三种基本逻辑运算：
  - “与”，运算符 ·，例：$A \cdot B$ 或 AB
  - “或”，运算符 +，例：$A + B$
  - “非”，运算符 —，例：$\bar{A}$

基本概念

- 变量：可以使用A、B、C …… 或 a、b、c …… 来表示，取值只能为0或1
- 反变量（变量的非，Complement）：变量上面有一条横线

  $\bar{A}$, $\bar{B}$, $\bar{C}$

- 项（Literal）：变量或它的反变量

  A, $\bar{A}$, B, $\bar{B}$, C, $\bar{C}$

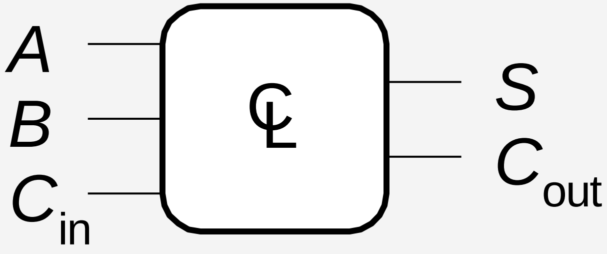
基本概念

Basic Concepts

布尔表达式

- 适用于描述组合逻辑电路中**输入与输出间的功能规范**
- 例如：$S = F(A, B, C_{in})$

  $C_{out} = F(A, B, C_{in})$



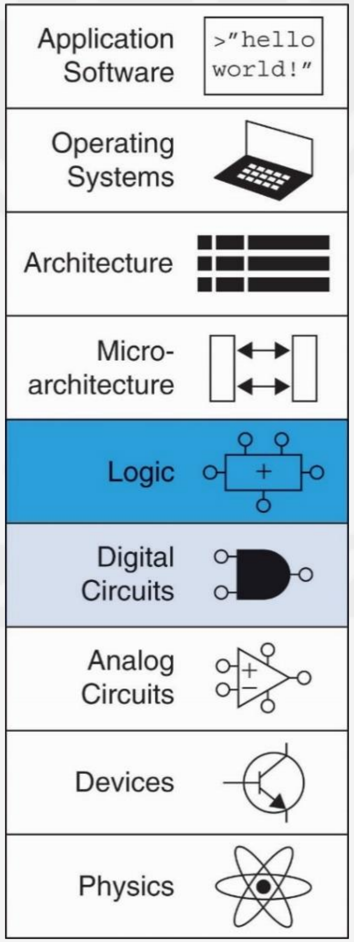
$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

布尔代数

Boolean Algebra

①基本概念
②公理
③定理
④最小项
⑤最大项
⑥标准与或式和标准或与式
⑦布尔表达式与真值表的转换
⑧使用定理化简表达式



公理

Axioms

布尔代数的公理

	公理		对偶公理	名称
A1	$B=0$ 如果 B $\neq 1$	A1'	$B=1$ 如果 B $\neq 0$	二进制量
A2	$\bar{0}=1$	A2'	$\bar{1}=0$	NOT
A3	$0 \cdot 0=0$	A3'	$1+1=1$	AND/OR
A4	$1 \cdot 1=1$	A4'	$0+0=0$	AND/OR
A5	$0 \cdot 1=1 \cdot 0=0$	A5'	$1+0=0+1=1$	AND/OR

对偶规则

- 对偶（Duality）
  - 设F为任意逻辑表达式,若将F中所有运算符和常量作如下变换

$$\begin{array}{cccc} \cdot & + & 0 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ + & \cdot & 1 & 0 \end{array}$$

例：$F = A\bar{B} + C\bar{D}$

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

  - 则所得新的表达式为F的对偶式 F'
- 对偶是相互的， F和 F'互为对偶式
- 对偶规则：两个逻辑表达式F和G相等，则对偶式F'和G'也相等

$$A(B + C) = AB + AC \xrightarrow{\text{对偶关系}} A + BC = (A + B)(A + C)$$

布尔代数

Boolean Algebra

① 基本概念

② 公理

③ 定理

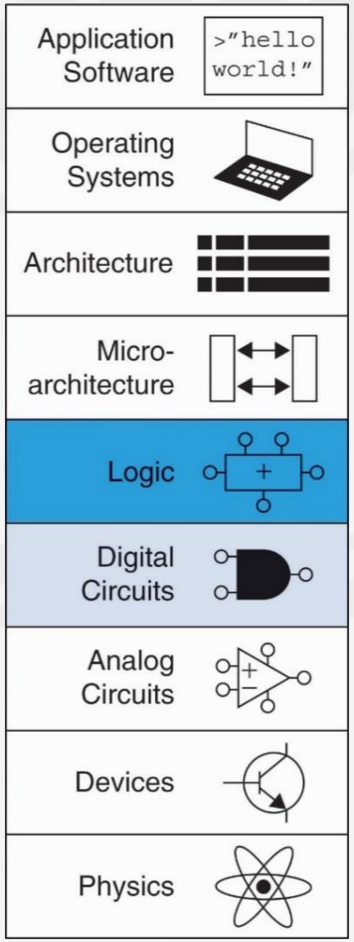
④ 最小项

⑤ 最大项

⑥ 标准与或式和标准或与式

⑦ 布尔表达式与真值表的转换

⑧ 使用定理化简表达式



定理

Theorems

单变量定理

	定理		对偶定理	名称
T1	$B \cdot 1 = B$	T1'	$B + 0 = B$	同一性
T2	$B \cdot 0 = 0$	T2'	$B + 1 = 1$	零元
T3	$B \cdot B = B$	T3'	$B + B = B$	重叠
T4		$\overline{\overline{B}} = B$		回旋
T5	$B \cdot \overline{B} = 0$	T5'	$B + \overline{B} = 1$	互补

定理

Theorems

多变量定理

	定理		对偶定理	名称
T6	$B \cdot C = C \cdot B$	T6’	$B + C = C + B$	交换律
T7	$(B \cdot C) \cdot D = B \cdot (C \cdot D)$	T7’	$(B + C) + D = B + (C + D)$	结合律
T8	$(B \cdot C) + (B \cdot D) = B \cdot (C + D)$	T8’	$(B + C) \cdot (B + D) = B + (C \cdot D)$	分配律
T9	$B \cdot (B + C) = B$	T9’	$B + (B \cdot C) = B$	吸收律
T10	$(B \cdot C) + (B \cdot \overline{C}) = B$	T10’	$(B + C) \cdot (B + \overline{C}) = B$	合并律
T11	$(B \cdot C) + (\overline{B} \cdot D) + (C \cdot D) = B \cdot C + \overline{B} \cdot D$	T11’	$(B + C) \cdot (\overline{B} + D) \cdot (C + D)$ $= (B + C) \cdot (\overline{B} + D)$	一致律
T12	$\overline{B_0 \cdot B_1 \cdot B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	T12’	$\overline{B_0 + B_1 + B_2 \ldots} = \overline{B_0} \cdot \overline{B_1} \cdot \overline{B_2} \ldots$	德•摩根定律

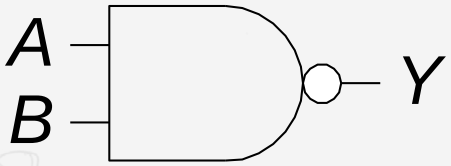
定理

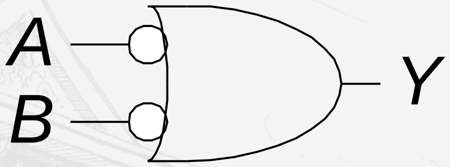
Theorems

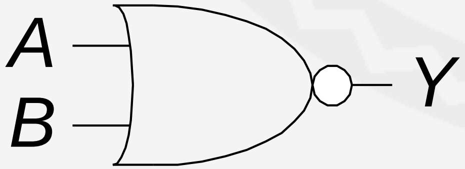
德 · 摩根定律

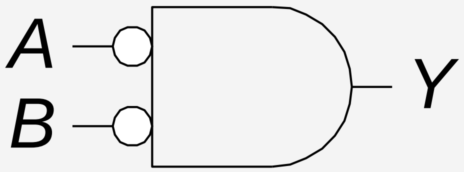
$Y=\overline{A\cdot B}=\overline{A}+\overline{B}$

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









定理

Theorems

定理的证明

■ 主要思想：完全归纳法

证明在变量所有的可能取值的组合下，定理都能够成立

等号左右两端的表达式所对应的真值表相等

■ 真值表

表征逻辑表达式输入和输出之间全部状态的表格

与逻辑真值表

A	B	F=A·B
0	0	0
0	1	0
1	0	0
1	1	1

布尔代数

Boolean Algebra

① 基本概念

② 公理

③ 定理

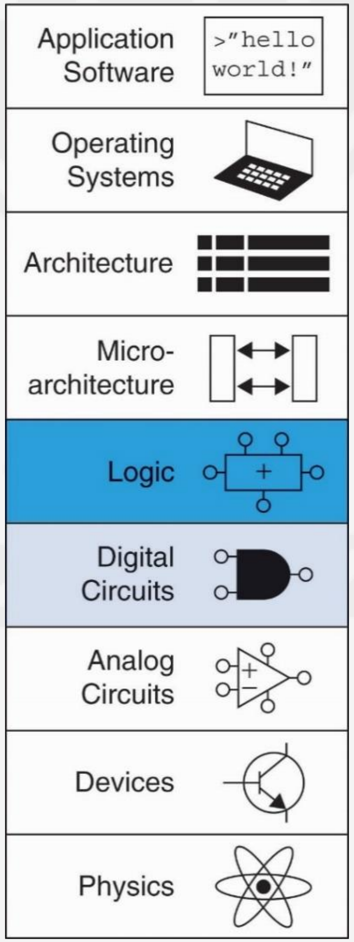
④ 最小项

⑤ 最大项

⑥ 标准与或式和标准或与式

⑦ 布尔表达式与真值表的转换

⑧ 使用定理化简表达式



最小项

Miniterm

基本概念

- 蕴含项（Implicant）：项的乘积

  $AB\bar{C}$，$\bar{A}C$，BC

- 最小项（Miniterm）：包含全部输入变量的乘积项

  $AB\bar{C}$，$A\bar{B}\bar{C}$，ABC

- 最大项（Maxterm）：包含全部输入变量的求和项（Sum）

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

最小项

- 最小项是一种特殊的乘积项（“与”项）
- 最小项特点
  - n个变量逻辑函数的每个最小项，一定是包含n个因子的乘积项
  - 在各个最小项中，每个变量必须以原变量或反变量形式作为因子出现一次，而且仅出现一次

例：包含A、B两变量的最小项共有四项(2^2)

$\bar{A}\,\bar{B}$　　$\bar{A}\,B$　　$A\,\bar{B}$　　$A\,B$

例：包含A、B、C三变量的最小项共有八项(2^3)

$\bar{A}\,\bar{B}\,\bar{C}$　　$\bar{A}\,\bar{B}\,C$　　$\bar{A}\,B\,\bar{C}$　　$\bar{A}\,B\,C$　　$A\,\bar{B}\,\bar{C}$　　$A\,\bar{B}\,C$　　$A\,B\,\bar{C}$　　$A\,B\,C$

最小项

Miniterm

最小项的编号

- 最小项用 m_i 表示
  - m表示最小项
  - 下标 i 为使该最小项为1的变量取值所对应的等效十进制数

- 例： 最小项 $\bar{A}BC$

要使该最小项为1，A、B、C的取值应为0、1、1

二进制数 011所等效的十进制数为 3，所以$\bar{A}BC = m_3$

最小项

Miniterm

三变量最小项编号表

最小项	使最小项为1的变量取值			对应的十进制数	编号
	A	B	C		
$\bar{A}\,\bar{B}\,\bar{C}$	0	0	0	0	m_0
$\bar{A}\,\bar{B}\,C$	0	0	1	1	m_1
$\bar{A}\,B\,\bar{C}$	0	1	0	2	m_2
$\bar{A}\,B\,C$	0	1	1	3	m_3
$A\,\bar{B}\,\bar{C}$	1	0	0	4	m_4
$A\,\bar{B}\,C$	1	0	1	5	m_5
$A\,B\,\bar{C}$	1	1	0	6	m_6
$A\,B\,C$	1	1	1	7	m_7

最小项

Miniterm

三变量最小项真值表

A	B	C	m_0 $\bar{A}\,\bar{B}\,\bar{C}$	m_1 $\bar{A}\,\bar{B}\,C$	m_2 $\bar{A}\,B\,\bar{C}$	m_3 $\bar{A}\,B\,C$	m_4 $A\,\bar{B}\,\bar{C}$	m_5 $A\,\bar{B}\,C$	m_6 $A\,B\,\bar{C}$	m_7 $A\,B\,C$
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

从表中可以看出，每个最小项只有一组变量取值能使其值为1,而其他各组取值该最小项皆为0。由这种“与”函数真值表中1的个数最少，而得名“最小项”

最小项

Miniterm

最小项的性质

① 变量任取一组值，仅有一个最小项为1，其他最小项为0

例：变量ABC的值为010，只有最小项 $\bar{A}B\bar{C}$（m_2）能使其为1

$$\bar{A}\,\bar{B}\,\bar{C}=\bar{A}\,\bar{B}\,C=\bar{A}\,B\,C=A\,\bar{B}\,\bar{C}=A\,\bar{B}\,C=A\,B\,\bar{C}=A\,B\,C=0$$

② n变量的全体最小项(共有2^n个)之和恒为1

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

③ n个变量任意两个不同的最小项相与，结果恒为0

最小项

Miniterm

最小项的性质（cont.）

④ 两最小项相邻，相邻最小项相“或”，可以合并成一项，并可以消去一个变量因子

- 相邻：两最小项如仅有一个变量因子不同，其他变量均相同，则称这两个最小项相邻
- 例：$\mathrm{ABC} + AB\bar{C} = AB$
- 任一n变量的最小项，必定和其他n个不同最小项相邻

（每一变量取反都是相邻项）

- 例：ABC与$\bar{A}BC$、$A\bar{B}C$、$AB\bar{C}$ 相邻

布尔代数

Boolean Algebra

① 基本概念

② 公理和定理

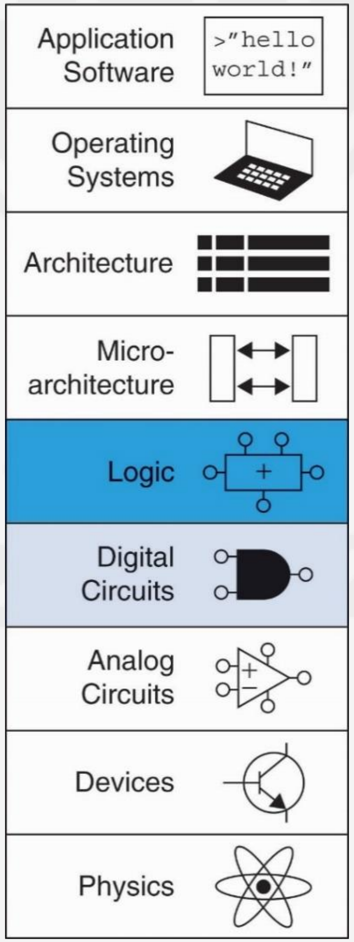
③ 最小项

④ 最大项

⑤ 标准与或式和标准或与式

⑥ 布尔表达式与真值表的转换

⑦ 使用定理化简表达式



最大项

Maxterm

最大项

- 最大项是一种特殊的和项（“或”项）
- 最大项特点
  - n个变量构成的每个最大项,一定是包含n个因子的“或”项;
  - 在各个最大项中，每个变量必须以原变量或反变量形式作为因子出现一次,而且仅出现一次

例：包含A、B两变量的最大项共有四项(2^2)

$\bar{A}+\bar{B}$　$\bar{A}+B$　$A+\bar{B}$　$A+B$

例：包含A、B、C三变量的最小项共有八项(2^3)

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

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

最大项编号

- 最大项用 M_i 表示
  - M 表示最大项
  - 下标 i 为使该最大项为0的变量取值所对应的等效十进制数

- 例： 最大项 $\overline{A}+B+C$

要使该最大项为 0， A、B、C的取值应为 1 、 0、 0

二进制数 100 所等效的十进制数为 4， 所以$\overline{A}+B+C=M_4$

最大项

Maxterm

三变量最大项编号表

最大项	使最大项为0的变量取值			对应的十进制数	编号
	A	B	C		
$A+B+C$	0	0	0	0	M_0
$A+B+\bar{C}$	0	0	1	1	M_1
$A+\bar{B}+C$	0	1	0	2	M_2
$A+\bar{B}+\bar{C}$	0	1	1	3	M_3
$\bar{A}+B+C$	1	0	0	4	M_4
$\bar{A}+B+\bar{C}$	1	0	1	5	M_5
$\bar{A}+\bar{B}+C$	1	1	0	6	M_6
$\bar{A}+\bar{B}+\bar{C}$	1	1	1	7	M_7

最大项

Maxterm

三变量最大项真值表

A B C	M_0 $A+B+C$	M_1 $A+B+\bar{C}$	M_2 $A+\bar{B}+C$	M_3 $A+\bar{B}+\bar{C}$	M_4 $\bar{A}+B+C$	M_5 $\bar{A}+B+\bar{C}$	M_6 $\bar{A}+\bar{B}+C$	M_7 $\bar{A}+\bar{B}+\bar{C}$
0 0 0	0	1	1	1	1	1	1	1
0 0 1	1	0	1	1	1	1	1	1
0 1 0	1	1	0	1	1	1	1	1
0 1 1	1	1	1	0	1	1	1	1
1 0 0	1	1	1	1	0	1	1	1
1 0 1	1	1	1	1	1	0	1	1
1 1 0	1	1	1	1	1	1	0	1
1 1 1	1	1	1	1	1	1	1	0

每个最大项只有对应的1组变量取值能使其值为0，正因为这种“或”函数真值表中1的个数最多，所以得名“最大项”

最大项

Maxterm

最大项的性质

① 变量任取一组值,仅有一个最大项为0,其它最大项为1

② n变量的全体最大项之积为 0

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

③ 不同的最大项相或，结果为 1

最大项的性质（cont.）

④ 两相邻的最大项相“与”，可以合并成一项（等于相同因子之和），并可消去一个变量因子

- 相邻：两最大项如仅有一个变量因子不同，其他变量均相同，则称这两个最大项相邻
- 例：$(A+B+C)\left(A+B+\bar{C}\right)=A+B$
- 证明：$=(A+B)+(A+B)\bar{C}+(A+B)+(A+B)C$

$$=(A+B)+(A+B)\left(\bar{C}+C\right)$$

$$=(A+B)+(A+B)\cdot 1=A+B$$

- 任一n变量的最大项，必定和其他n个不同的最大项相邻

最大项

Maxterm

最小项和最大项的关系

- 编号下标相同的最小项和最大项互为反函数
- 即 $M_i = \overline{m_i}$ 或 $m_i = \overline{M_i}$
- 例 $\mathrm{m}_0 = \bar{A}\,\bar{B}\,\bar{C} = \overline{A + B + C} = \overline{M_0}$

$$M_0 = A + B + C = \overline{\bar{A} + \bar{B} + \bar{C}} = \overline{m_0}$$

布尔代数

Boolean Algebra

① 基本概念

② 公理

③ 定理

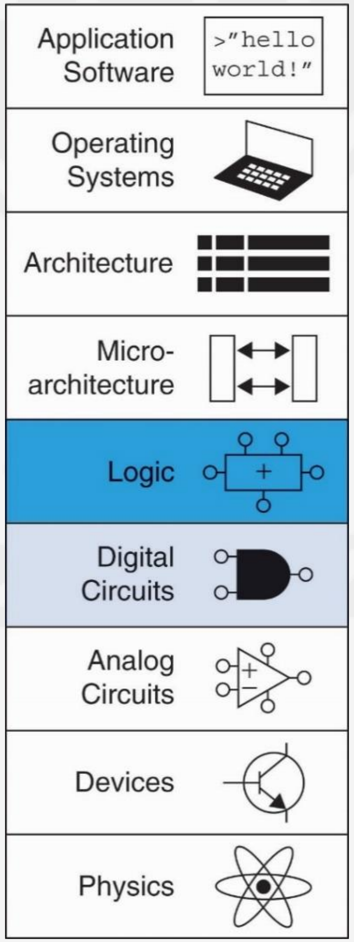
④ 最小项

⑤ 最大项

⑥ 标准与或式和标准或与式

⑦ 布尔表达式与真值表的转换

⑧ 使用定理化简表达式



标准与或式和标准或与式

Canonical SOP & Canonical POS

标准与或式（sum-of-products）

- 由最小项之和构成的逻辑表达式
- 例： $F(A,B,C) = \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,\bar{C}$

  $= \sum(m_2,\ m_4\ ,\ m_6)$

  $= \sum(2,\ 4,\ 6)$
- 每个最小项都对应真值表中值为1的一行
- 标准与或式是最小项之间的或运算
- 标准与或式与真值表间一一对应
- 因此，从标准与或式中可以直接判断哪些变量取值可以是表达式为1

A	B	C	F(A,B,C)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

标准与或式和标准或与式

Canonical SOP & Canonical POS

标准与或式具有唯一性

- 任一逻辑函数都可以表达为最小项之和的形式,而且是唯一的
- 例：$F(A,B,C) = AB + \bar{A}C$

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

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

$$= m_6 + m_7 + m_1 + m_3$$

$$= \sum(1,\ 3,\ 6,\ 7)$$

标准与或式和标准或与式

Canonical SOP & Canonical POS

标准或与式（product-of-sums）

- 最大项之积的构成的逻辑表达式
- 例： $F(A,B,C)=(A+B+C)(A+\bar{B}+C)(\bar{A}+B+C)$

$$=\prod(M_0,\ M_2\ ,\ M_4)$$

$$=\prod(0,\ 2,\ 4)$$

- 任一逻辑函数都可以表达为最大项之积的形式，而且是唯一的

标准与或式和标准或与式

Canonical SOP & Canonical POS

标准与或式和标准或与式的关系

$$F=\sum_{i} m_i=\prod_{j\neq i} M_j$$

■ 推导:

$$F=\overline{\sum_{j\neq i} m_j}$$ （根据最小项的性质，当$m_i=1$时，其它最小项都是0）

$$=\prod_{j\neq i}\overline{m_j}$$ （德 · 摩根定律）

$$=\prod_{j\neq i} M_j$$ （最小项和最大项的关系，$M_i=\overline{m_i}$）

布尔代数

Boolean Algebra

① 基本概念

② 公理

③ 定理

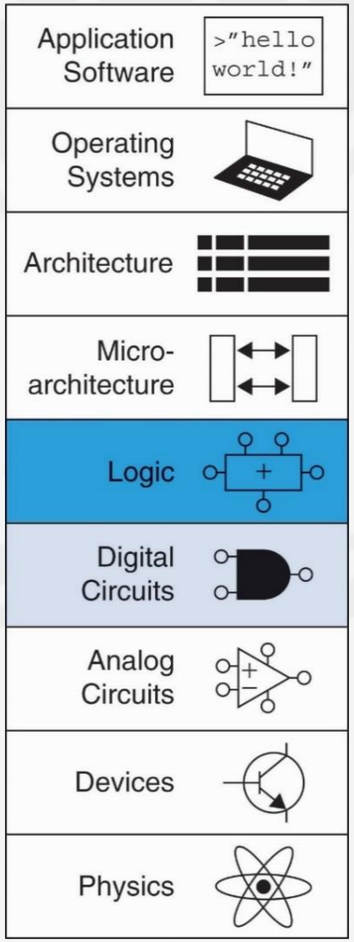
④ 最小项

⑤ 最大项

⑥ 标准与或式和标准或与式

⑦ 布尔表达式与真值表的转换

⑧ 使用定理化简表达式



布尔表达式与真值表的转换

Conversion Between Boolean Equation & Truth Table

布尔表达式→真值表

① 将变量的组合所有取值组合一一代入表达式进行计算得到

② 将表达式转化为标准与或式

例：$F(A,B,C) = AB + BC$

$$= AB(C+\bar{C}) + (A+\bar{A})BC$$

$$= ABC + AB\bar{C} + \bar{A}BC$$

$$= \sum m(3,6,7)$$

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

布尔表达式与真值表的转换

Conversion Between Boolean Equation & Truth Table

布尔表达式→真值表（cont.）

③ 根据函数式的逻辑含义，直接填表

$F(A,B,C) = AB + BC$ 表示的逻辑含义为：

1. A 和 B 同时为 "1"，即 $AB = 1$时，$F = 1$
2. B 和 C 同时为 "1"，即 $BC = 1$ 时，$F = 1$
3. 当不满足上面两种情况时，$F = 0$

	A	B	C	F
	0	0	0	0
	0	0	1	0
	0	1	0	0
BC同时为1	0	1	1	1
	1	0	0	0
	1	0	1	0
	1	1	0	1
AB同时为1	1	1	1	1

布尔表达式与真值表的转换

Conversion Between Boolean Equation & Truth Table

真值表→布尔表达式

- 根据最小项的性质，直接从真值表写出标准与或式

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

- 也可根据最大项的性质，直接写出标准或与式

$$F(A,B,C)=\prod M(0,1,2,4)$$

A	B	C	F
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

布尔代数

Boolean Algebra

① 基本概念

② 公理

③ 定理

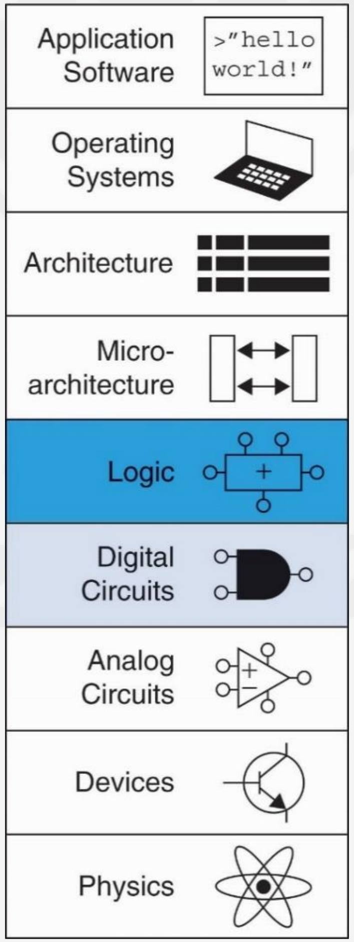
④ 最小项

⑤ 最大项

⑥ 标准与或式和标准或与式

⑦ 布尔表达式与真值表的转换

⑧ 使用定理化简表达式



使用定理化简表达式

例 1:

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

$$= B\left(A + \overline{A}\right) \qquad \text{T8} \qquad (B \cdot C) + (B \cdot D) = B \cdot (C + D)$$

$$= B(1) \qquad \text{T5'} \qquad B + \overline{B} = 1$$

$$= B \qquad \text{T1} \qquad B \cdot 1 = B$$

使用定理化简表达式（cont.）

■ 例 2:

$Y = A(AB + ABC)$

$= A(AB(1 + C))$ T8 $(B \cdot C) + (B \cdot D) = B \cdot (C + D)$

$= A(AB(1))$ T2’ $B + 1 = 1$

$= A(AB)$ T1 $B \cdot 1 = B$

$= (AA)B$ T7 $(B \cdot C) \cdot D = B \cdot (C \cdot D)$

$= AB$ T3 $B \cdot B = B$

使用定理化简表达式

Simplifying Equations

化简时的注意事项

- 借助布尔代数的公理、定理，对复杂的布尔表达式推导、变换和化简是逻辑设计的重要工作
- 其中必须注意逻辑代数与普通代数的区别：
  - 不存在指数、系数、减法和除法
  - 等式两边的相同项不能随便消去

$A+A=A$ 不能得到 $A+A=2A$ 系数

$A\cdot A=A$ 不能得到 $A\cdot A=A^2$ 指数

$A+\bar{A}=1$ 不能得到 $A=1-\bar{A}$ 消项

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

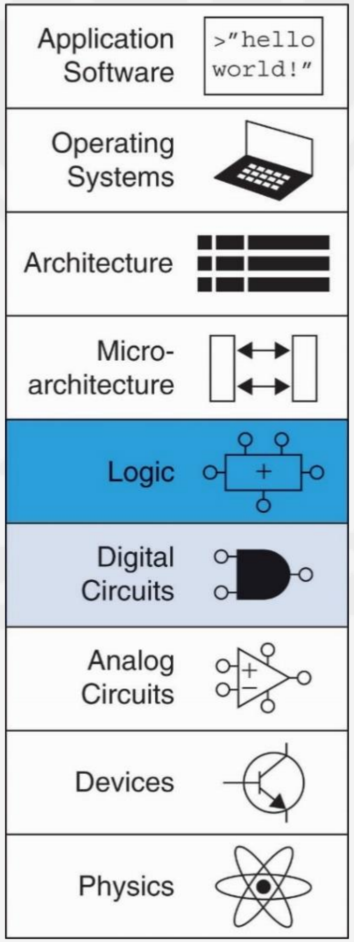
不能得到 $A\bar{B}+\bar{A}B=A+B$ 消项

$A(A+B)=A$ 不能得到 $A+B=1$ 消项

本章内容

Topic

- 引言
- 布尔代数
- 卡诺图
- 从逻辑到门
- 多级组合逻辑
- X和Z
- 组合逻辑电路设计方法
- 组合逻辑中的时序问题
- 组合逻辑模块



卡诺图

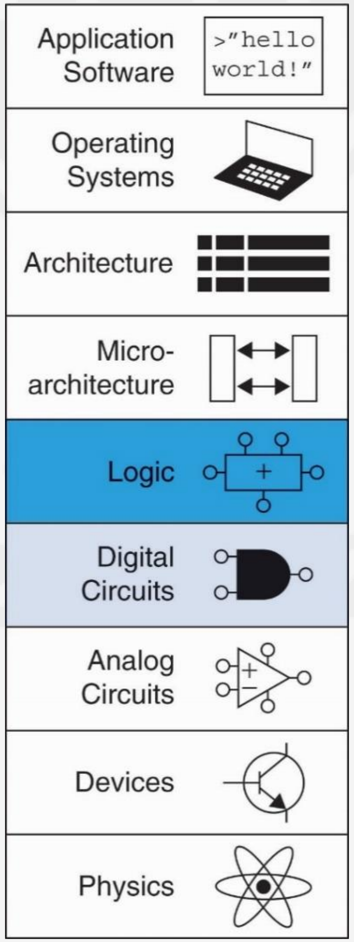
Karnaugh Maps

① 基本概念

② 在卡诺图中合并最小项

③ 使用卡诺图化简表达式

④ 使用无关项化简表达式



使用卡诺图化简布尔表达式

- 通过合并项可以实现布尔表达式的化简
- $PA + P\bar{A} = P$
- 卡诺图化简法是将逻辑函数用一种称为“卡诺图”的图形来表示,然后在卡诺图上进行函数化简的方法。

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

基本概念

Basic Concepts

卡诺图的构成

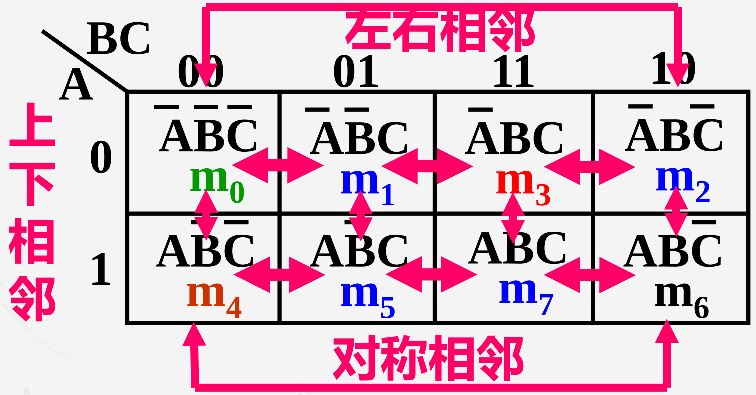
- 卡诺图是一种包含一些小方块的几何图形
- 图中每个小方块称为一个单元，每个单元对应一个最小项
- 两个相邻的最小项在卡诺图中也必须是相邻的。卡诺图中相邻的含义:

① 几何相邻性,即几何位置上相邻,也就是左右紧挨着或者上下相接

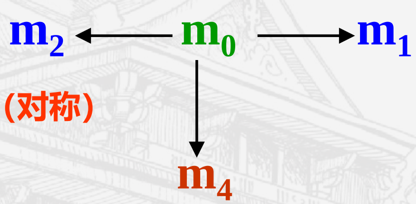
② 对称相邻性,即图形中位于边缘的单元与对称位置的单元是相邻的

Y \ C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

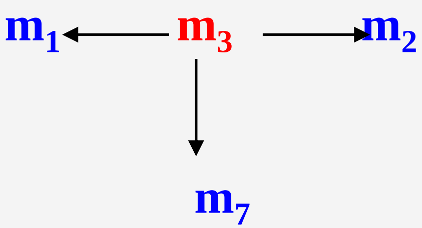
三变量卡诺图



相邻性规则



几何相邻性规则



对称相邻：

- m_0 与 m_2
- m_4 与 m_6

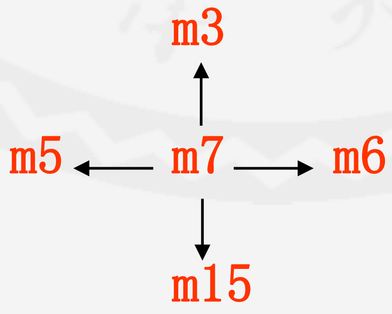
■ 特点：任何两组相邻，只有1位变量取值不同。即符合循环码排列规则。如BC的取值00,01,11,10→00…

二变量、四变量卡诺图

A\B	0	1
0	0	1
1	2	3

AB\CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

相邻性规则



基本概念

Basic Concepts

五变量卡诺图

取值排列符合相邻性规则

AB \ CDE	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

用卡诺图表示逻辑函数

- 把各组变量值所对应的逻辑函数的值，填在卡诺图对应的小方格中
- 卡诺图是真值表的一种变形
- 例：$\mathrm{F}(A,\ B,\ C)=\bar{A}BC+A\bar{B}C+ABC$

用卡诺图表示为：

A \ BC	00	01	11	10
0	0	0	1 m_3	0
1	0	1 m_5	1 m_7	0

卡诺图

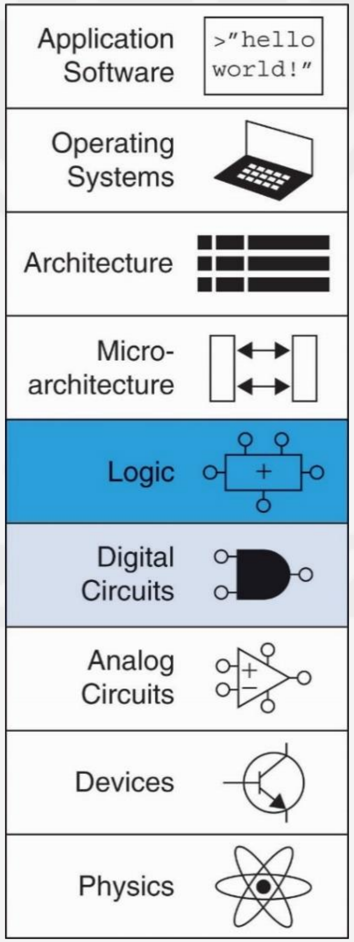
Karnaugh Maps

① 基本概念

② 在卡诺图中合并最小项

③ 使用卡诺图化简表达式

④ 使用无关项化简表达式



在卡诺图中合并最小项

Merge Miniterms in Karnaugh Map

卡诺图上合并最小项的规则

- 当卡诺图中有最小项相邻时（即：有标1的方格相邻），可利用最小项相邻的性质，对最小项合并

- 规则一：

卡诺图上任何两个标1的方格相邻，可以合为1项，并可消去1个（取值相反的）变量。

在卡诺图中合并最小项

Merge Miniterms in Karnaugh Map

例：三变量卡诺图，消去1个变量

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

取值：011　　111　　101

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

$\bar{A}BC + ABC = BC$

消去取值相反的变量

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

化简结果：$F = AC + BC$

在卡诺图中合并最小项

Merge Miniterms in Karnaugh Map

例：四变量卡诺图，消去1个变量

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

$\bar{A}\bar{B}\bar{D}$　消去C变量

ABD　消去C变量

在卡诺图中合并最小项

Merge Miniterms in Karnaugh Map

合并最小项的规则(cont.)

- 规则二:

  卡诺图上四个标1方格相邻，可合并为一项，并可消去2个变量

- 四个标1方格相邻的特点:
  - 同在一行或一列
  - 同在一个田字格中

在卡诺图中合并最小项

Merge Miniterms in Karnaugh Map

例：四变量卡诺图，消去2个变量

①同在一行或一列

$\overline{A}\overline{B}$ 消去CD

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

CD 消去AB

②同在一个田字格中

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

$\overline{B}\overline{D}$

BD

横看消去C，纵看消去A

在卡诺图中合并最小项

Merge Miniterms in Karnaugh Map

思考题

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

F = ?

$\bar{A}\bar{B}$ BC

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

F = ?

$B\bar{D}$ $\bar{B}C$

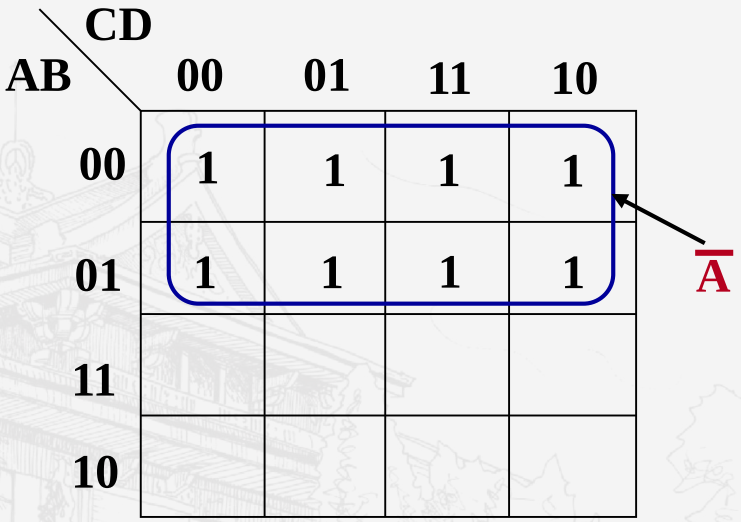
在卡诺图中合并最小项

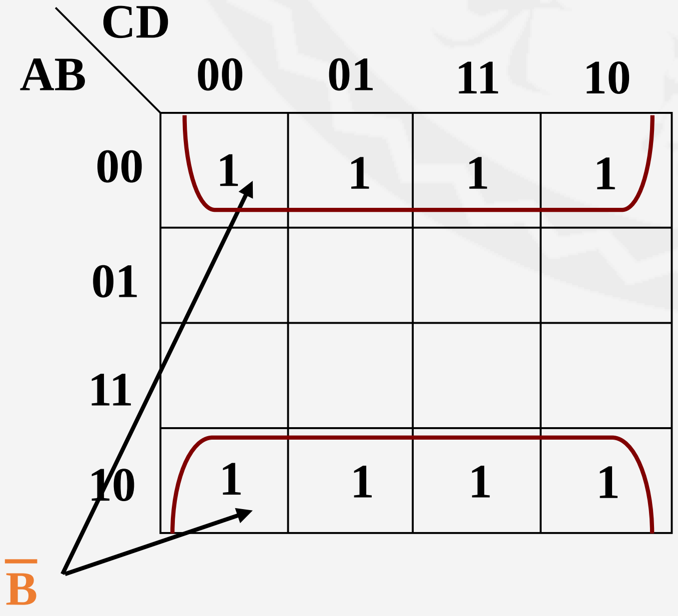
Merge Miniterms in Karnaugh Map

合并最小项的规则(cont.)

- 规则三:

卡诺图上八个标1方格相邻，可以并为一项，并可消去3个变量





在卡诺图中合并最小项

Merge Miniterms in Karnaugh Map

思考题

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

在卡诺图中合并最小项

Merge Miniterms in Karnaugh Map

合并规则总结

- 综上，在 n 个变量的卡诺图中:
  - 只有 2 的 i 次方个相邻的 标1 方格（必须排列成方形格或矩形格的形状）才能圈在一起，合并为一项
  - 该项保留了原来各项中 n-i 个相同的变量
  - 消去 i 个不同变量

如：n=4，i=3 （4个变量，如：ABCD，$2^i=2^3=8$）

则:保留1个相同值变量,消去3个不同值变量

卡诺图

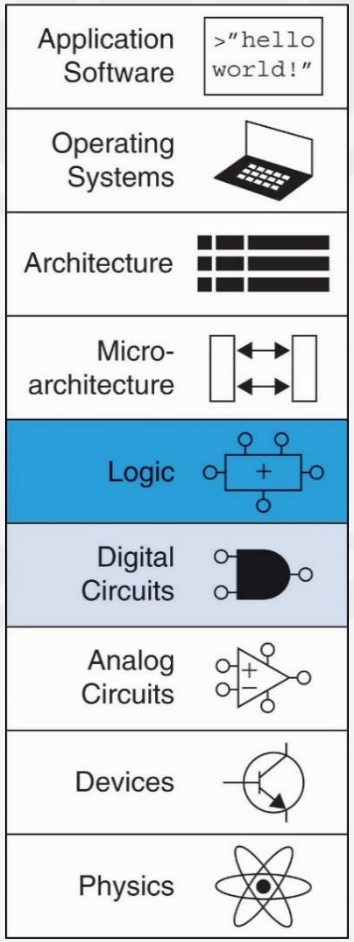
Karnaugh Maps

① 基本概念

② 在卡诺图中合并最小项

③ 使用卡诺图化简表达式

④ 使用无关项化简表达式



用卡诺图化简表达式

- 基本概念：
  - 卡诺图上的每一个圈都代表一个蕴含项
  - 主蕴含项：扩展到最大的蕴含项
  - 奇异“1”单元：卡诺图中仅能被单一主蕴含项覆盖的方格
  - 质主蕴含项：包含着一或多个的奇异“1”单元的主蕴含项
- 化简目标：最简与或式
- 最简标准
  - 项数最少，意味着卡诺图中圈数最少
  - 每项中的变量数最少，意味着卡诺图中的圈尽可能大

使用卡诺图化简逻辑表达式

Simplifying Equations Using Karnaugh Maps

- 例：将$F(A, B, C) = \sum m(3, 4, 5, 6, 7)$

  化为最简与或式

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

$F = A + BC$ （最简）

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

$$
\begin{aligned}
F &= A\bar{B} + BC + AB\bar{C} \text{（非最简）} \\
&= A(\bar{B} + B\bar{C}) + BC \\
&= A(\bar{B} + \bar{C}) + BC \\
&= A\overline{BC} + BC \\
&= A + BC
\end{aligned}
$$

使用卡诺图化简逻辑表达式

Simplifying Equations Using Karnaugh Maps

化简步骤（结合实例说明）

- 例：将 $F(A, B, C, D) = \sum m(0, 1, 3, 7, 8, 10, 13)$ 化为最简与或式
- 解：

  ① 由表达式填卡诺图

  ② 圈出孤立的标1方格

  （质主蕴含项）

AB \ CD	00	01	11	10
00	1	1	1	
01			1	
11		$\boxed{1}$		
10	1			1

m13

使用卡诺图化简逻辑表达式

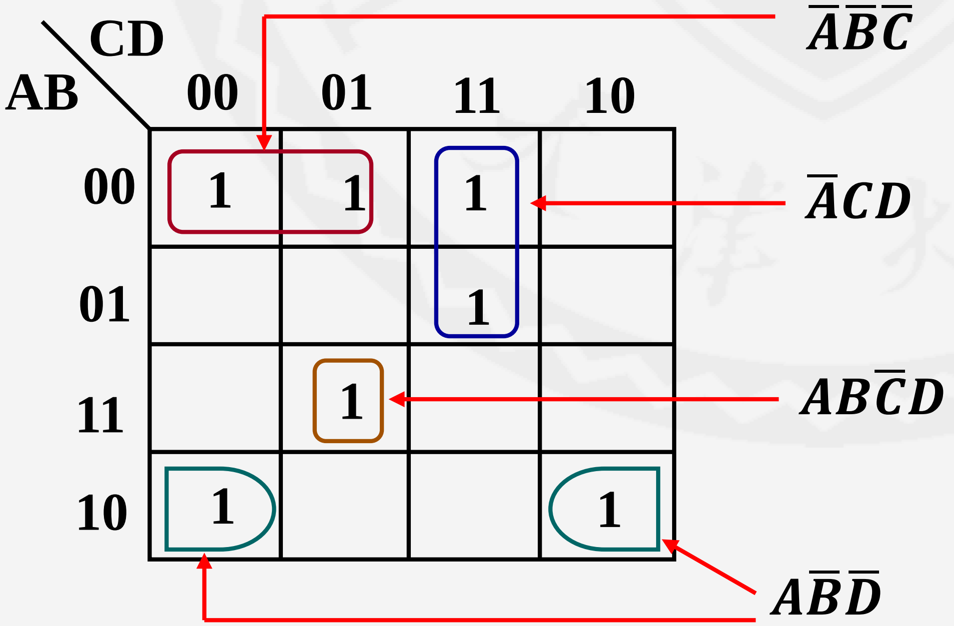
Simplifying Equations Using Karnaugh Maps

③ 找出只被一个最大的圈所覆盖的标1方格，并圈出覆盖该标1方格的最大圈（质主蕴含项） m_3m_7，m_8m_{10}

④ 将剩余的相邻标1方格,圈成尽可能少，而且尽可能大的圈 m_3m_7

⑤ 将各个对应的乘积项相加，写出最简与或式

$$F(A,B,C,D)=AB\bar{C}D+\bar{A}CD+A\bar{B}\bar{D}+\bar{A}\bar{B}\bar{C}$$



使用卡诺图化简逻辑表达式

Simplifying Equations Using Karnaugh Maps

一种特殊情况

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

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

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

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

得到两种化简结果，都是最简的

使用卡诺图化简逻辑表达式

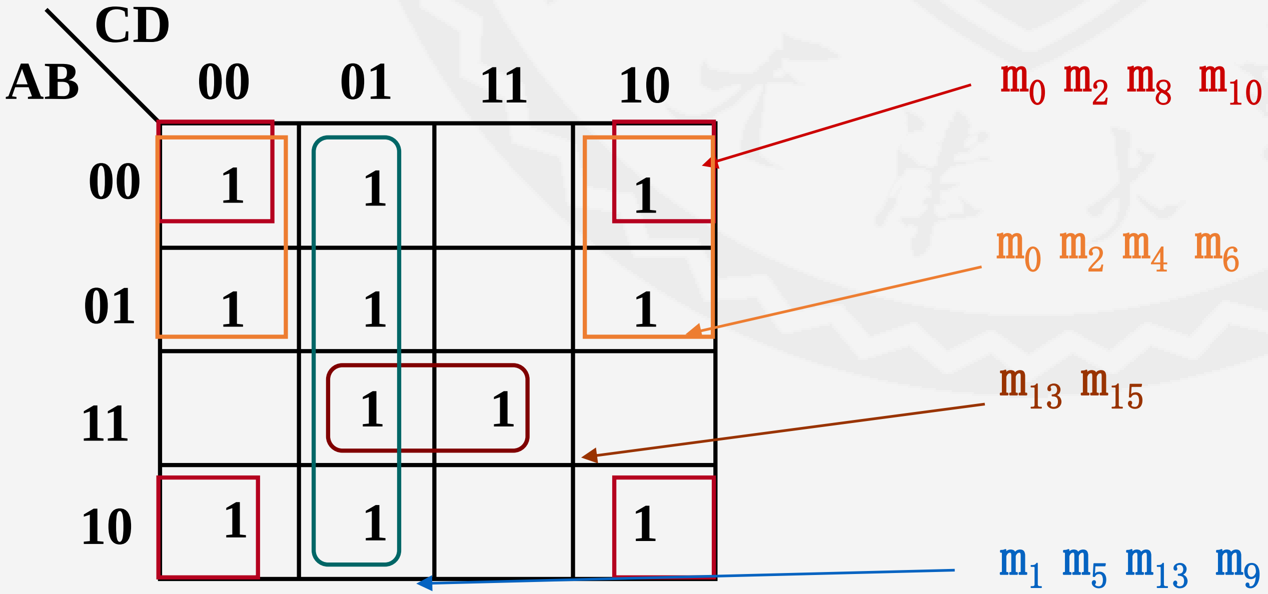
Simplifying Equations Using Karnaugh Maps

化简中注意的问题

① 每一个标1的方格必须至少被圈一次

② 每个圈中包含的相邻小方格数,必须为2的整数次幂

③ 为了得到尽可能大的圈,圈与圈之间可以重叠



使用卡诺图化简逻辑表达式

Simplifying Equations Using Karnaugh Maps

化简中注意的问题（cont.）

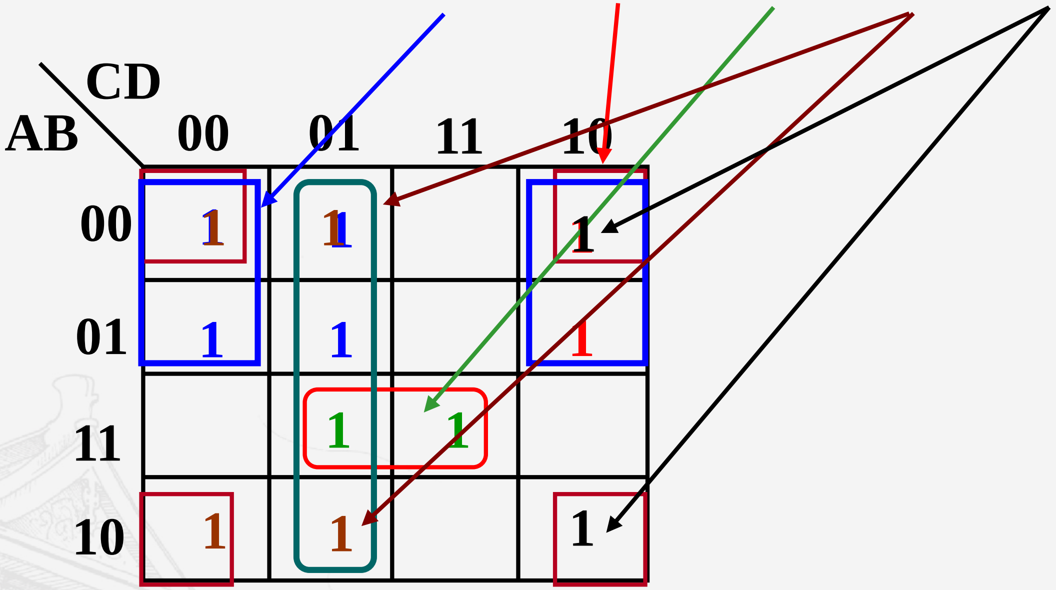
④ 若某个圈中的所有标1方格,已经完全被其它圈所覆盖,则该圈为多余的。即每个圈中至少应有1个标1方格未被其他圈覆盖

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

图中蓝色的圈是多余的

课堂练习

$$F(A,B,C,D)=\bar{A}\bar{C}+\bar{A}C\bar{D}+ABD+\bar{B}\bar{C}+\bar{B}C\bar{D}$$



$$F(A,B,C,D)=ABD+\bar{B}\bar{D}+\bar{A}\bar{D}+\bar{C}D$$

卡诺图

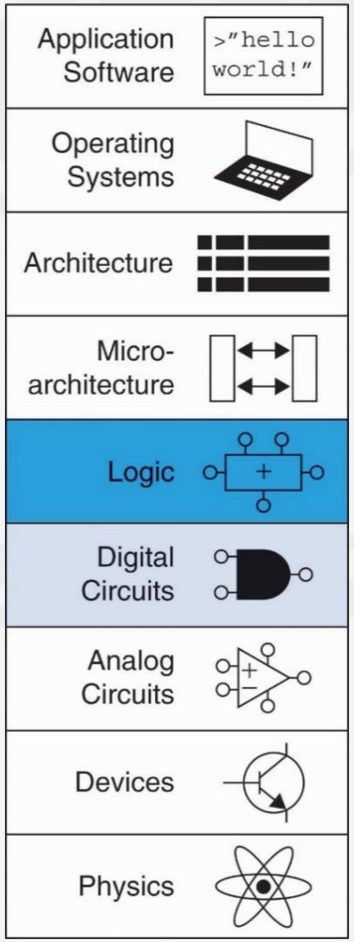
Karnaugh Maps

① 卡诺图的定义

② 在卡诺图中合并最小项

③ 使用卡诺图化简表达式

④ 使用无关项化简表达式



使用无关项化简表达式

Simplifying Equations With Don't Cares

无关项的化简

- 真值表的输出x
  - 当输出的值不重要或者相对应的输入组合从不出现时
  - 可以由设计者决定这些输出是0还是1
- 充分利用无关项，可以进一步化简逻辑表达式

使用无关项化简表达式

Simplifying Equations With Don't Cares

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

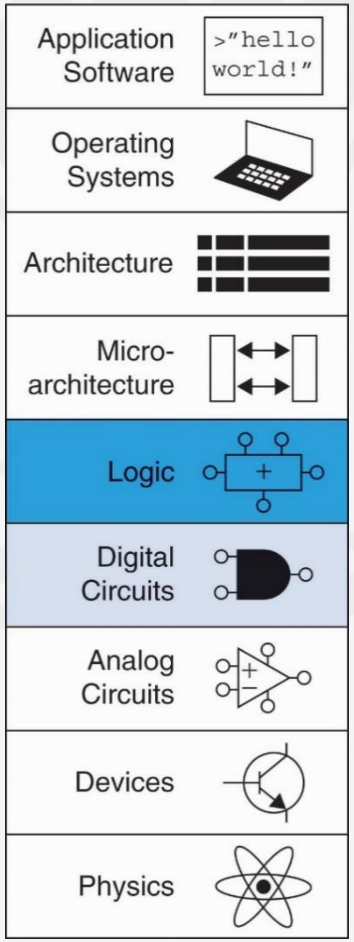
CD \ AB	00	01	11	10
00	1		X	1
01		X	X	1
11	1	1	X	X
10	1	1	X	X

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

本章内容

Topic

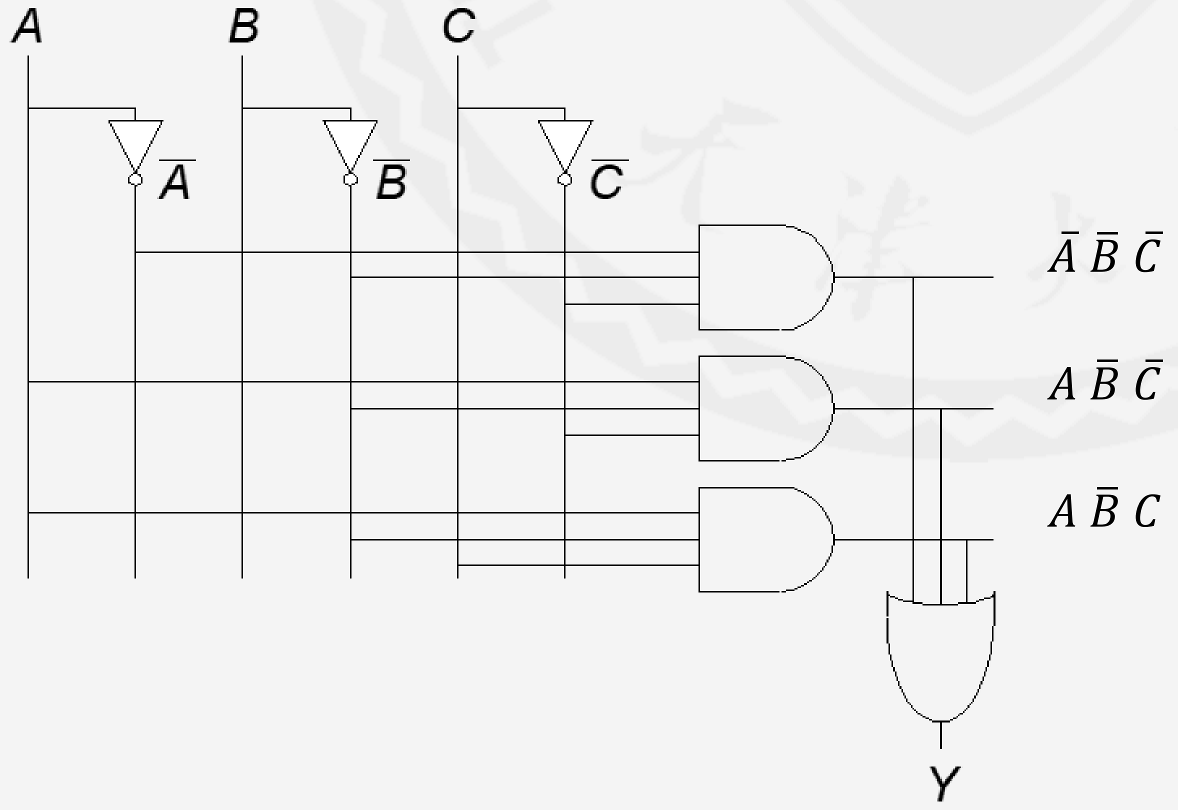
- 引言
- 布尔代数
- 卡诺图
- 从逻辑到门
- 多级组合逻辑
- X和Z
- 组合逻辑电路设计方法
- 组合逻辑中的时序问题
- 组合逻辑模块



由布尔表达式绘制原理图

- 与或式可以使用两级门电路来实现
  - 第一级：与门
  - 第二级：或门
- 例：

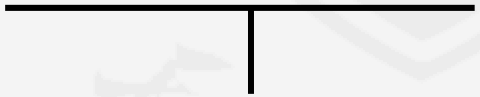
$$Y = \bar{A}\,\bar{B}\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,\bar{B}\,C$$



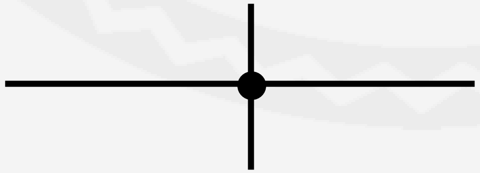
电路原理图绘制原则

- 原理图绘制需要遵循一致的风格，以易于阅读和检查错误
- 绘制原则如下：
  - 输入在原理图的左边（或顶部）
  - 输出在原理图的右边（或底部）
  - 门电路流应从左至右（或从上至下）
  - 尽量使用直线连接
  - T型接头表示两条线有连接
  - 两条线交叉的地方有一个点，表示有连接
  - 两条线交叉的地方没有点，表示没有连接

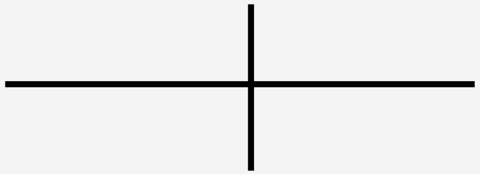
T型接头连接



交叉的地方有点
表示连接



交叉的地方没有点
表示无连接

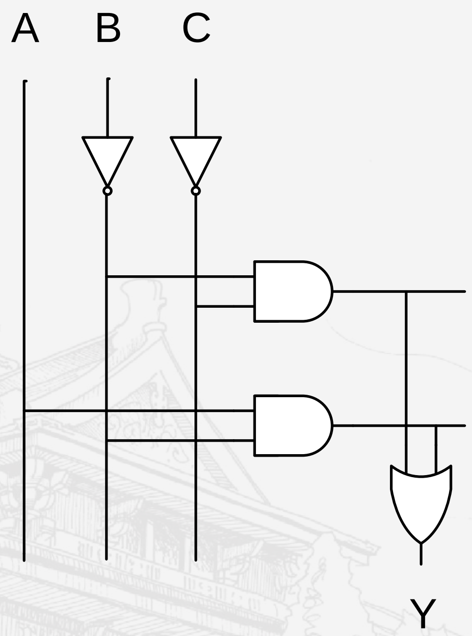


从逻辑到门

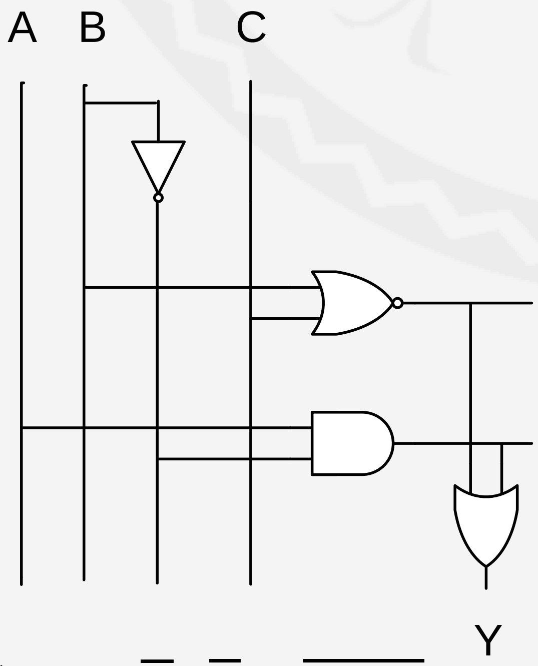
From Logic To Gates

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

原理图



使用更少的门

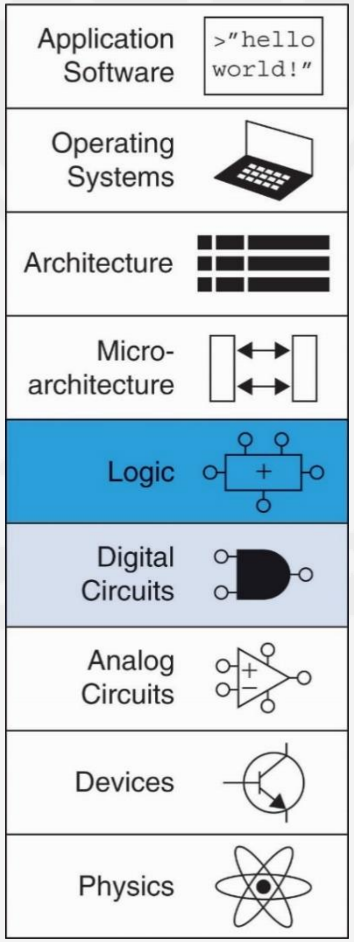


德•摩根定理 $\overline{B} \cdot \overline{C} = \overline{B + C}$

本章内容

Topic

- 引言
- 布尔代数
- 卡诺图
- 从逻辑到门
- **多级组合逻辑**
- X和Z
- 组合逻辑电路设计方法
- 组合逻辑中的时序问题
- 组合逻辑模块

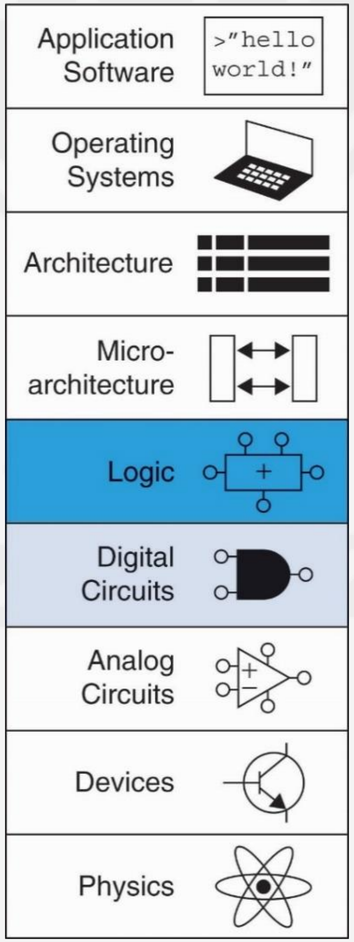


多级组合逻辑

Multilevel Combinational Logic

① 减少硬件

② 推气泡



减少硬件

Hardware Reduction

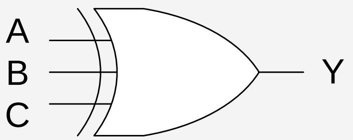
减少硬件的目的

- 所有的逻辑表达式都可以转化为与或式
- 理论上，与或式可以使用两级门电路来实现（先与后或）
- 使用二级逻辑可能带来更高的成本
- 在工程上，门电路的扇入数不可能无限制的增加
  - 受工艺、成本等方面的制约
- 采用多级逻辑
  - 可以减少门电路的数量
  - 可以减少扇入数

- 扇入（fan-in）：单个逻辑门能够接受的数字信号最大输入数

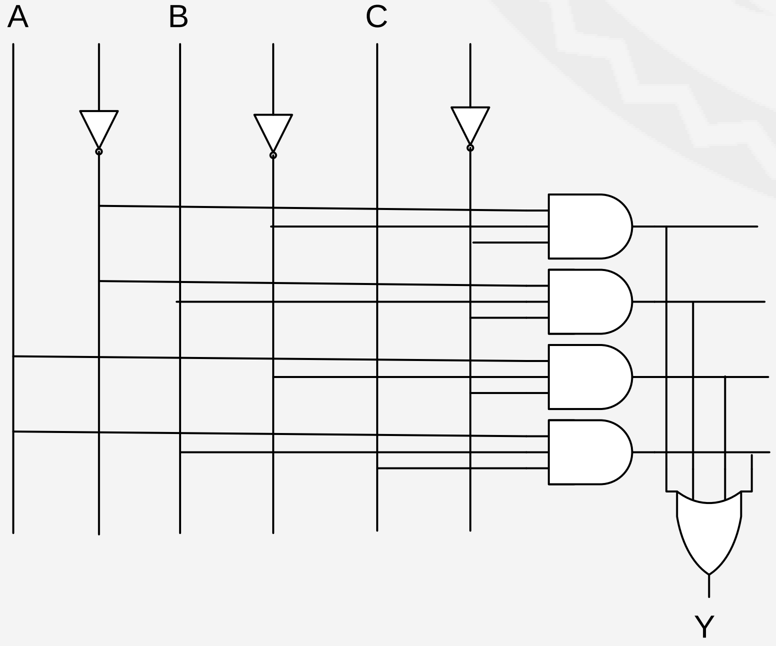
扇入数：2　　扇入数：5

3输入异或门的实现



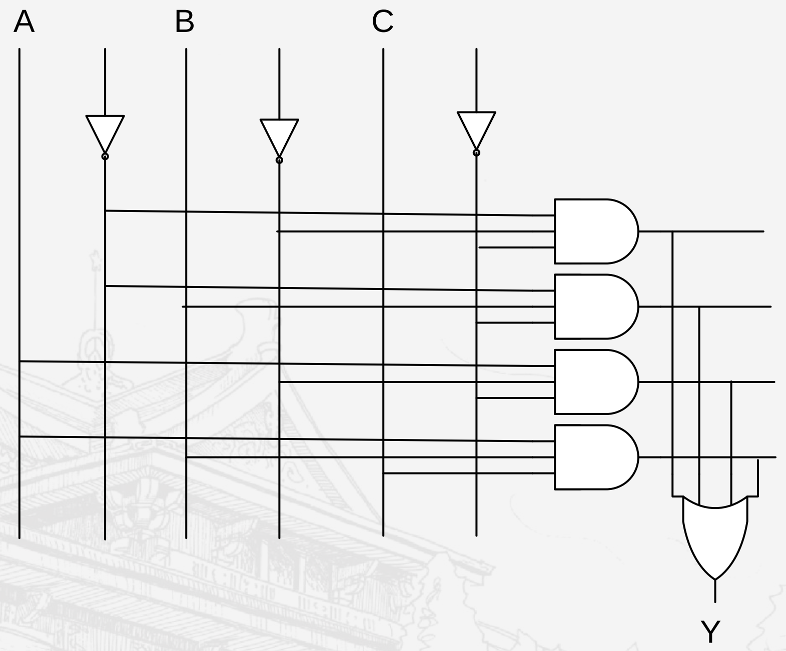
$$Y = A \oplus B \oplus C = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

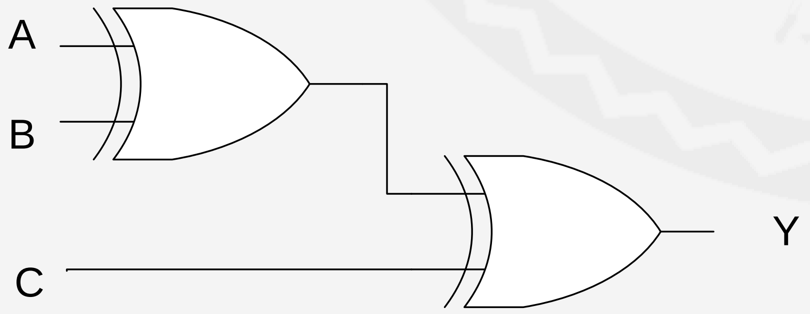
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



使用更少的门电路

$$Y = A \oplus B \oplus C = (A \oplus B) \oplus C$$





使用两个2输入异或门

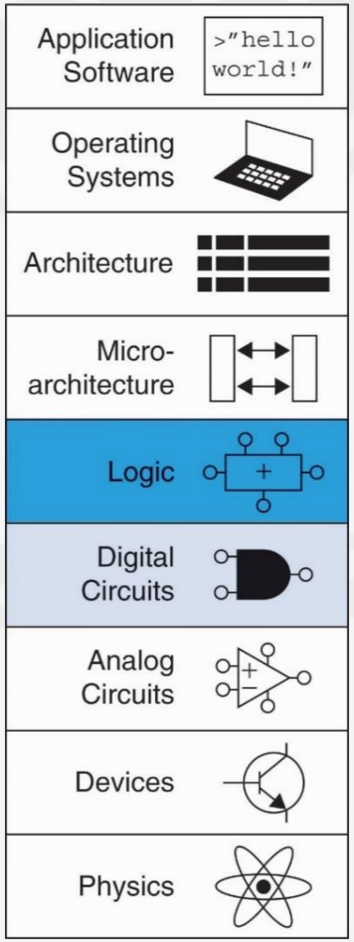
构造一个3输入异或门

多级组合逻辑

Multilevel Combinational Logic

① 减少硬件

② 推气泡



推气泡

Bubble Pushing

推气泡

- CMOS电路中经常使用与非门和或非门
- 不易直接根据电路推导出表达式
- 推气泡可以帮助我们重画电路，更容易确定逻辑功能

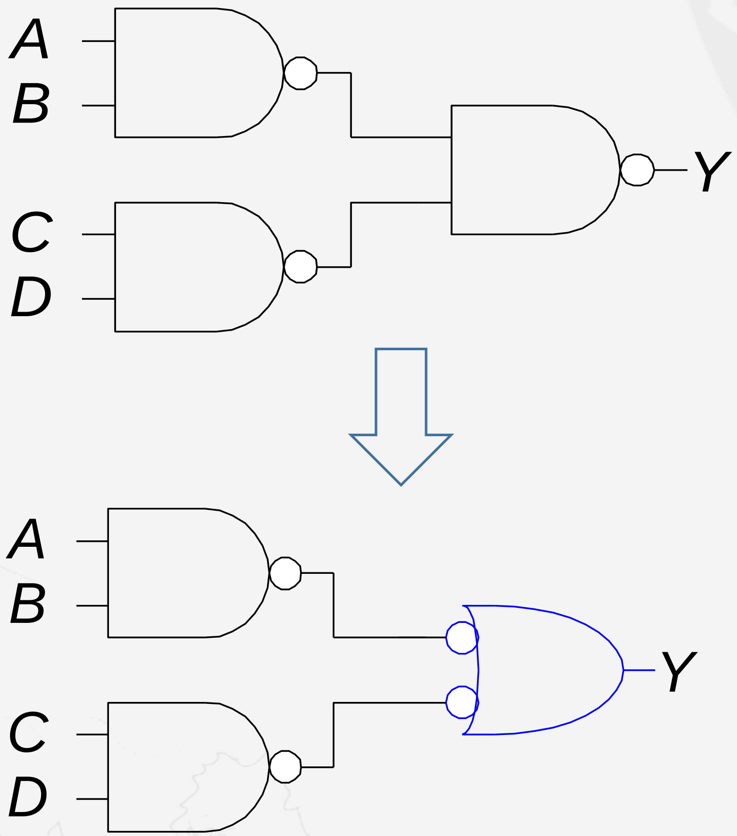
- 向后推
  - 电路符号变化
  - 将气泡加在输入端

- 向前推
  - 电路符号变化
  - 将气泡加在输出端

推气泡

Bubble Pushing

写出电路的逻辑表达式



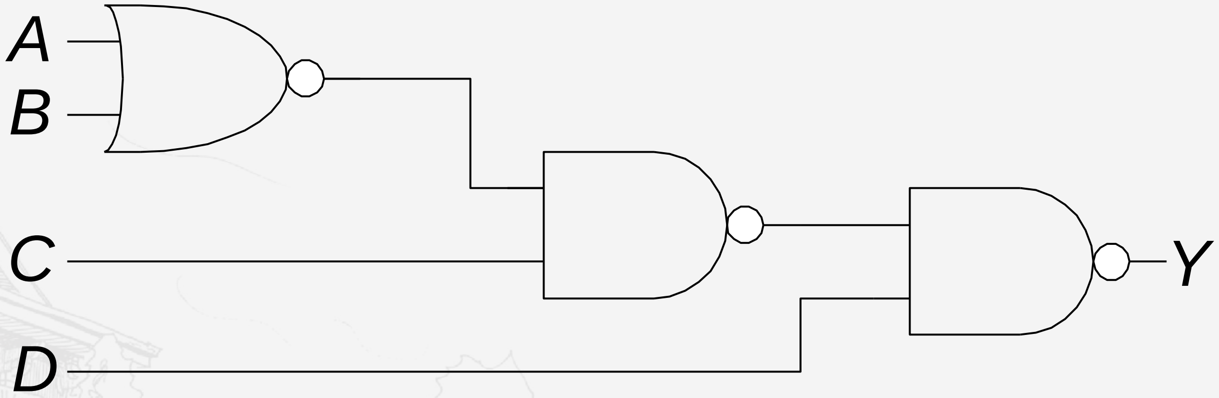
$$Y = AB + CD$$

推气泡

Bubble Pushing

推气泡的方法

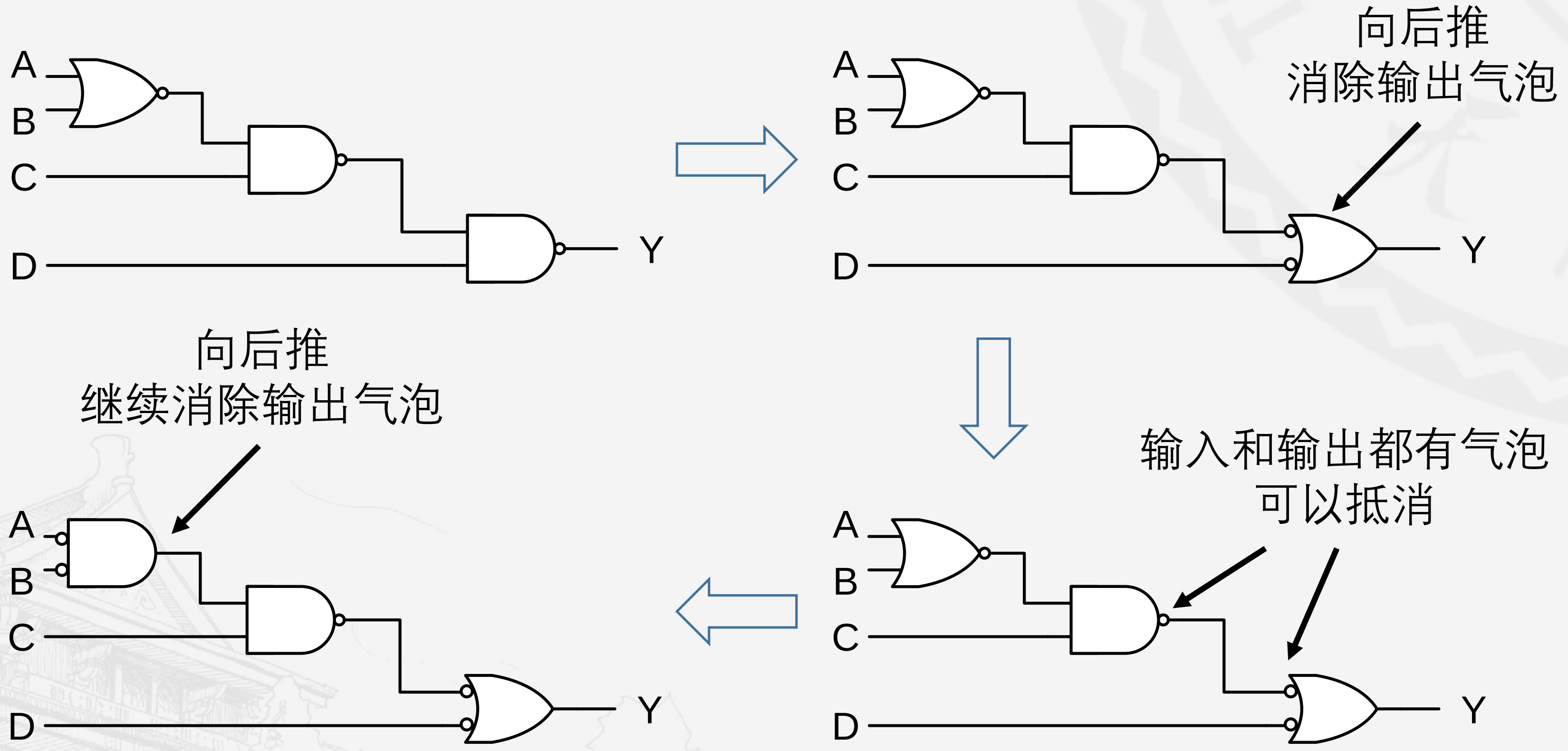
- 从输出端向输入端推
- 将气泡从电路最后的输出端开始推
- 如果当前门有一个输入气泡，则消除该气泡，并在其上一级门的输出加上气泡



推气泡

Bubble Pushing

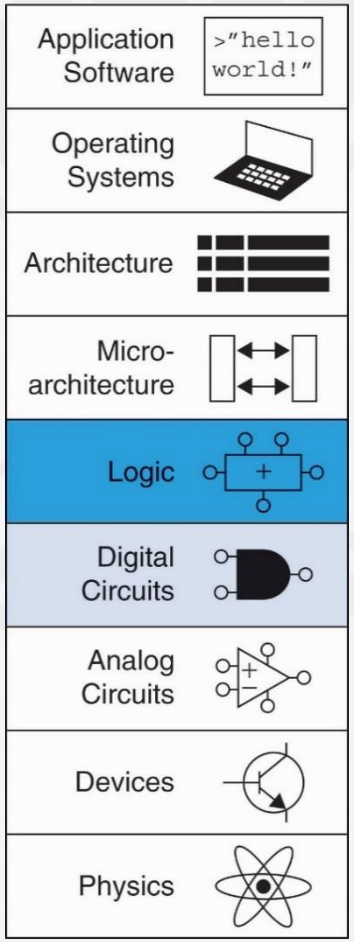
推气泡的例子



本章内容

Topic

- 引言
- 布尔代数
- 卡诺图
- 从逻辑到门
- 多级组合逻辑
- X和Z
- 组合逻辑电路设计方法
- 组合逻辑中的时序问题
- 组合逻辑模块



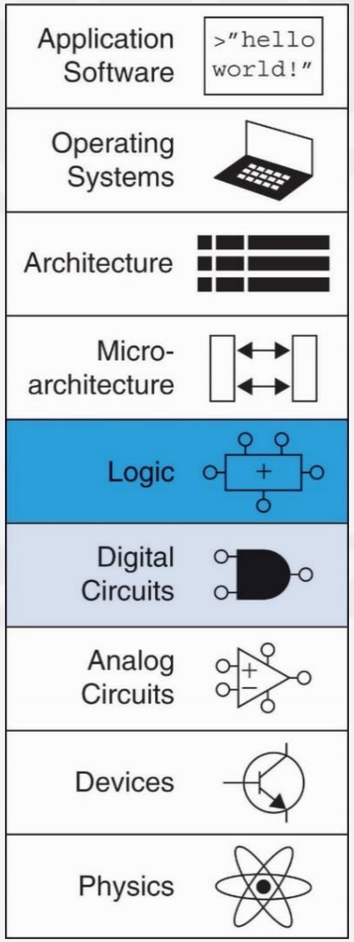
X和Z

X and Z

① 非法值

② 无关项

③ 浮空值

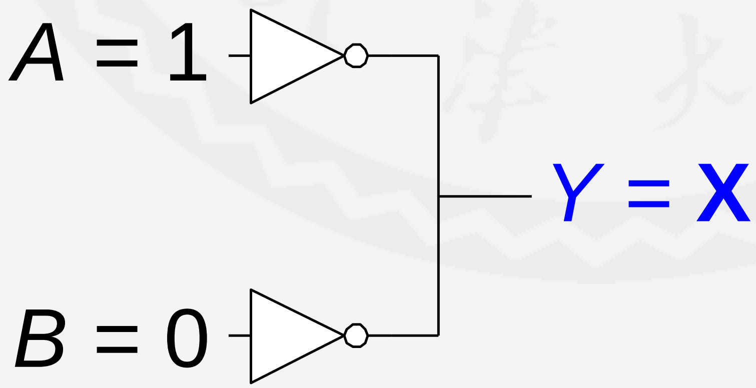


非法值

Illegal

非法值：X

- 竞争（Contention）：电路结点同时被0和1驱动
  - 电压值可能介于 $0 \sim V_{DD}$ 之间
  - 可能是0，可能是1，也可能处于禁止区域内
  - 导致电路的功耗变大、电路发热，并导致损坏
- 注意：竞争通常是由于电路设计缺陷引起的



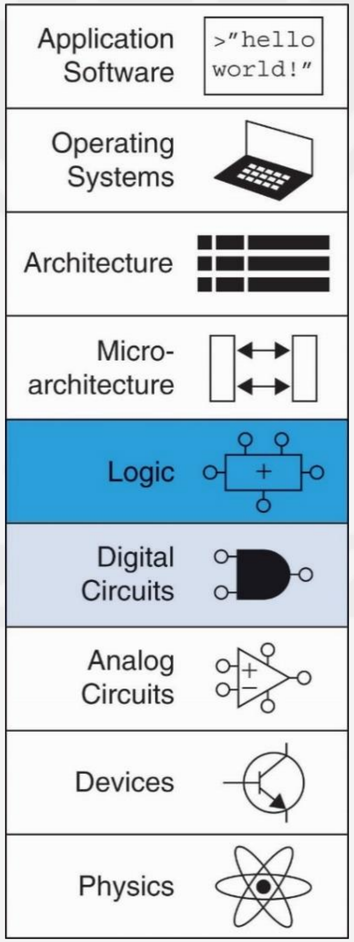
X和Z

X and Z

① 非法值

② 无关项

③ 浮空值



无关项：X

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

- 在优先级电路中：
  - 如果A_3输入为TRUE，则输出不用考虑其他的输入量
- 用符号X表示输出不需要考虑的输入（Don’t Care ）

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0

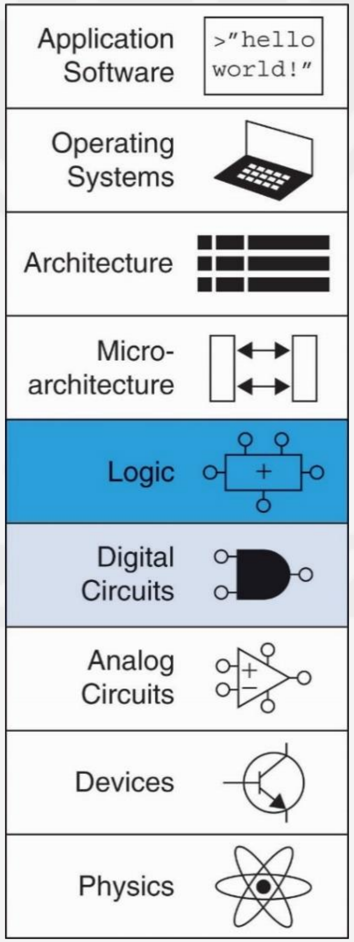
X和Z

X and Z

① 非法值

② 无关项

③ 浮空值



浮空值

Floating

浮空值（floating）：Z

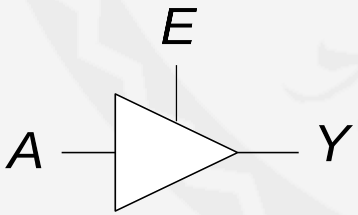
- 浮空也称为悬空、高阻态（High impedance）、高Z态、开路、断路
- 浮空不等于逻辑0
  - 使用电压表并不能判断哪个电路结点处于浮空状态
  - 测量断路结点的电压和接地点的电压，在电压表上的读数都为0
- 当电路的输入结点浮空时，输出不确定
  - 可能为0，可能为1，也可能为某个中间电压（处于禁止区）
- 产生浮空结点常见的原因是忘记将电压连接到输入端
- 但浮空结点并不意味着电路一定出错

浮空值

Floating

三态缓冲器 (tristate buffer)

- 浮空可以用来防止结点处于竞争状态
- 三态缓冲器
  - 有3种可能输出状态
  - 高电平、低电平和浮空
  - 输入端 A、输出端 Y、使能端 E



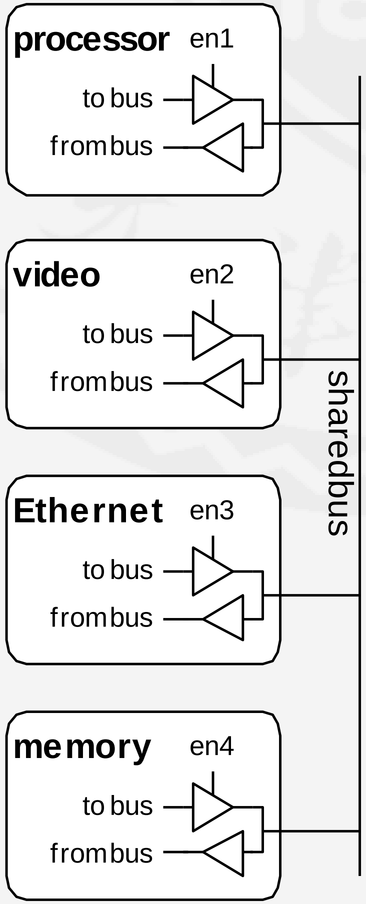
该图表示 E为高电平有效使能

E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

使能端为0时，输出端浮空
使能端为1时，Y跟随A变化

三态缓冲器的应用

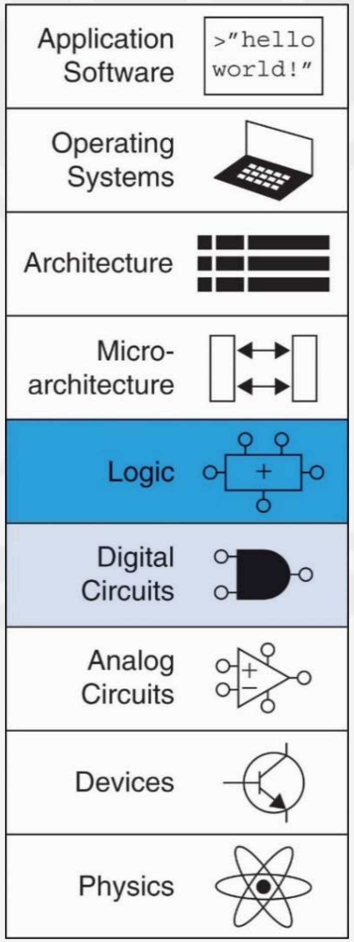
- 当一个结点同时连接n个输出时，若其中n-1个输出处于浮空状态，则当前结点的值等于驱动正常电平输出端的值
- 在连接多个芯片的总线上使用
  - 许多不同的设备同时连接在同一总线上
  - 在某一个时刻只允许一个芯片的使能信号有效，并向总线输出数据
  - 其它芯片的输出必须浮空，以防止总线竞争
  - 任何芯片在任何时刻都可以通过总线读取信息



本章内容

Topic

- □ 引言
- □ 布尔代数
- □ 卡诺图
- □ 从逻辑到门
- □ 多级组合逻辑
- □ X和Z
- ■ 组合逻辑电路设计方法
- □ 组合逻辑中的时序问题
- □ 组合逻辑模块



组合逻辑电路设计方法

The Design Method of Combinatorial Logic Circuit

设计思路

① 对实际逻辑问题进行抽象，定义输入和输出逻辑变量

② 由实际逻辑问题列出真值表

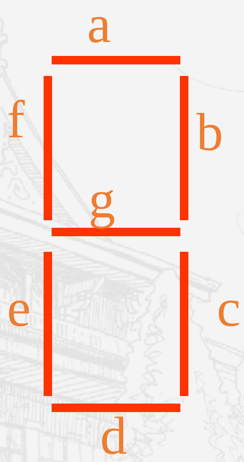
③ 由真值表写出表达式

④ 化简表达式

⑤ 画出原理图

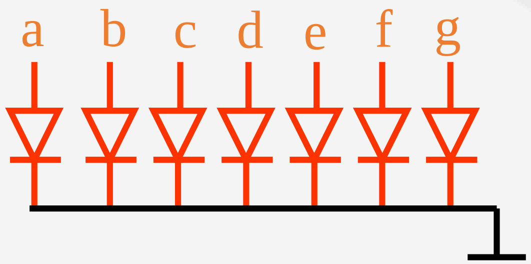
7段数码管驱动电路

- 7段数码管驱动电路
  - 4位输入数据，输入一个十进制数字
    （4位二进制数可表示一位十进制数）
  - 7位输出控制发光管显示数字 0 – 9



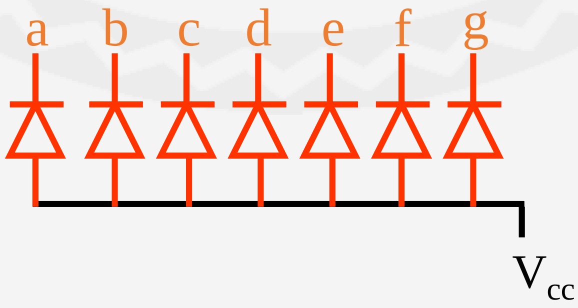
七段数码管的两种连接方法

① 共阴极



对应字段高电平时点亮

② 共阳极



对应字段低电平时点亮

组合逻辑电路设计方法

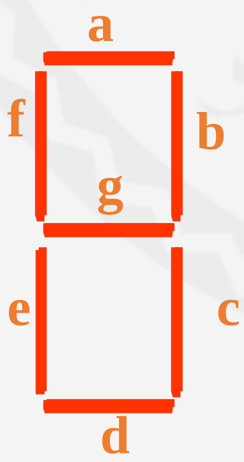
The Design Method of Combinatorial Logic Circuit

■在这里，我们讨论共阴极7段管显示译码器的设计

① 对实际逻辑问题进行抽象，定义输入和输出逻辑变量

设：4位输入数据为D_3、D_2、D_1、D_0，对应输入的十进制数字

7位输出数据为S_a，S_b，S_c，S_d，S_e，S_f，S_g，输出为1时，点亮对应字段的数码管



0： abcdef

1： bc

2： abdeg

3： abcdg

4： bcfg

5： acdfg

6： acdefg

7： abc

8： abcdefg

9： abcdfg

组合逻辑电路设计方法

The Design Method of Combinatorial Logic Circuit

② 由实际逻辑问题列出真值表

D_3	D_2	D_1	D_0	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
其他				0	0	0	0	0	0	0

组合逻辑电路设计方法

The Design Method of Combinatorial Logic Circuit

③ 由真值表写出表达式

④ 化简表达式

$$S_a = \sum m(0,2,3,5,6,7,8,9)$$

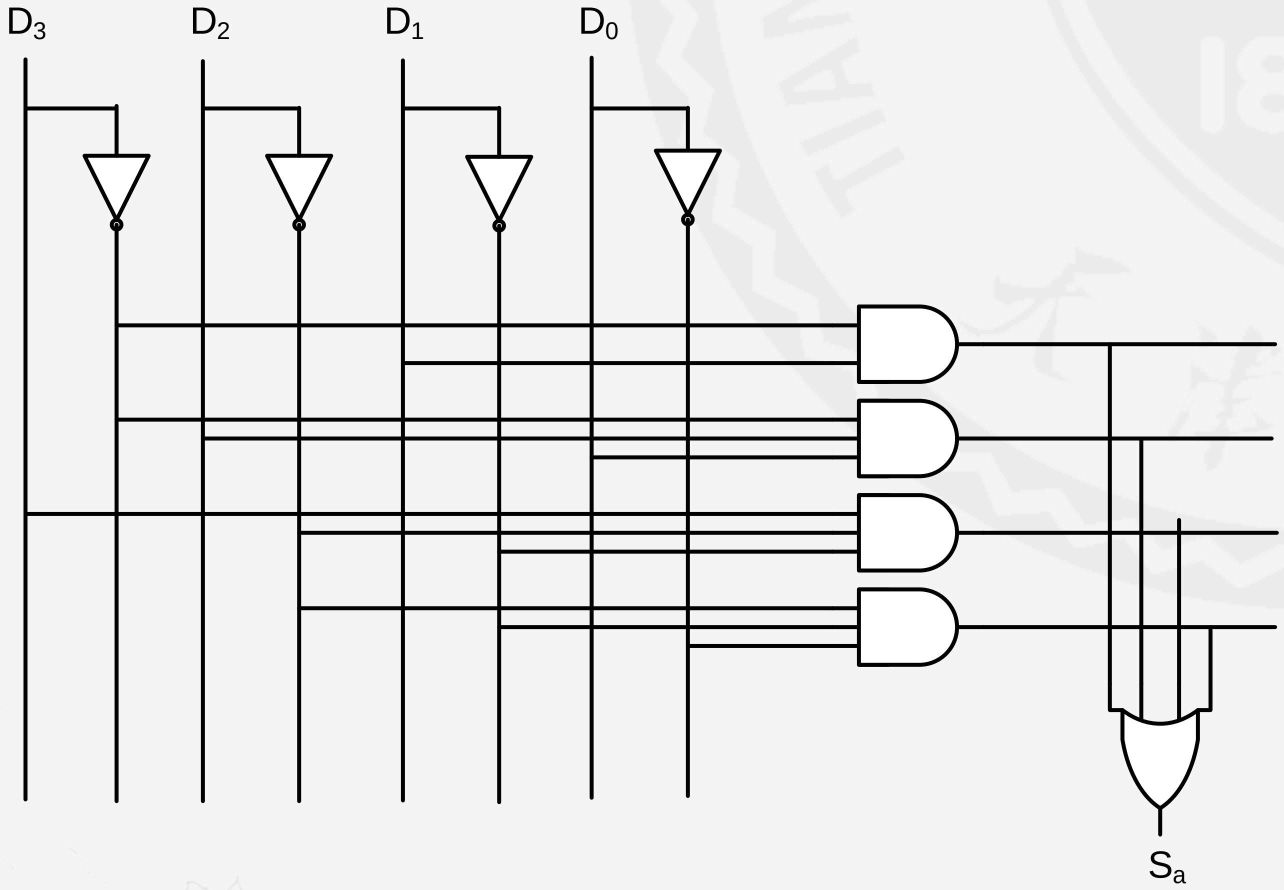
D_1D_0 \ D_3D_2	00	01	11	10
00	1			1
01		1		1
11	1	1		
10	1	1		

$$S_a = \overline{D_3}D_1 + \overline{D_3}D_2D_0 + D_3\overline{D_2}\,\overline{D_1} + \overline{D_2}\,\overline{D_1}\,\overline{D_0}$$

组合逻辑电路设计方法

The Design Method of Combinatorial Logic Circuit

⑤ 画出原理图



S_b，S_c，S_d，S_e，S_f，S_g 的设计略

考虑无关项

D_3	D_2	D_1	D_0	S_a
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

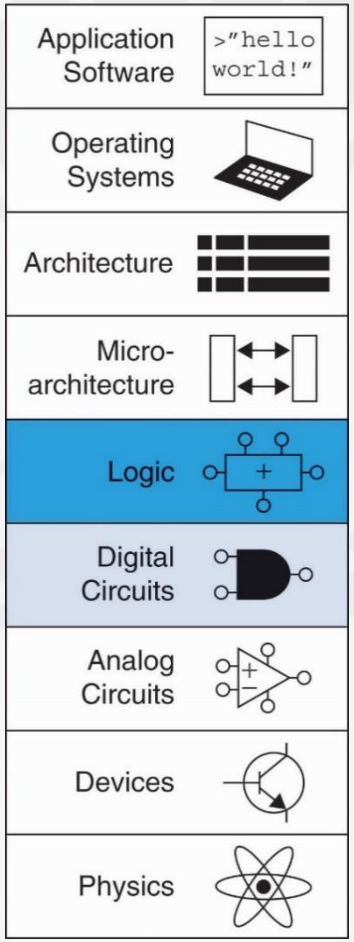
D_1D_0 \ D_3D_2	00	01	11	10
00	1		X	1
01		1	X	1
11	1	1	X	X
10	1	1	X	X

$$S_a = D_3 + D_2D_0 + \overline{D_2}\,\overline{D_0} + D_1$$

本章内容

Topic

- 引言
- 布尔代数
- 卡诺图
- 从逻辑到门
- 多级组合逻辑
- X和Z
- 组合逻辑电路设计方法
- **组合逻辑中的时序问题**
- 组合逻辑模块

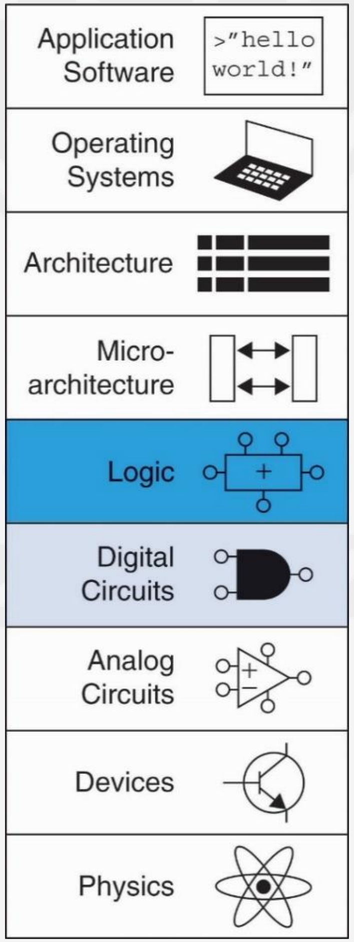


组合逻辑中的时序问题

Timing in Combinational Logic

① 传播延迟和最小延迟

② “毛刺”

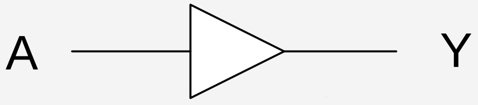


传播延迟和最小延迟

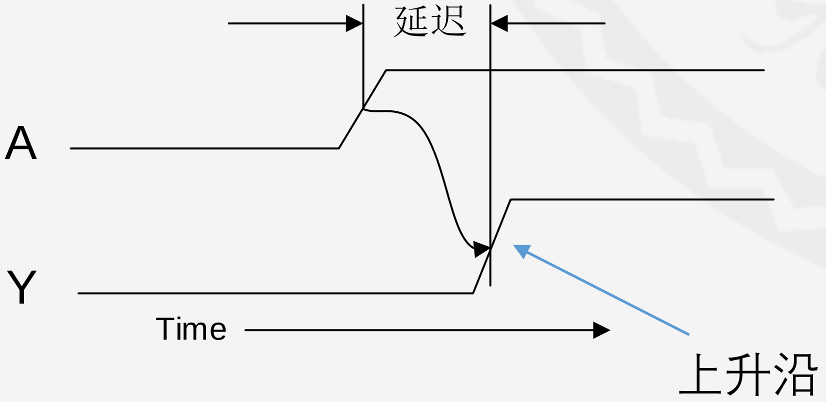
Propagation & Contamination Delay

时序

- 在实际电路中，输出响应输入的改变需要一定的时间



缓冲器（buffer）



缓冲器时序图

- 电路设计中最具有挑战性的问题是时序

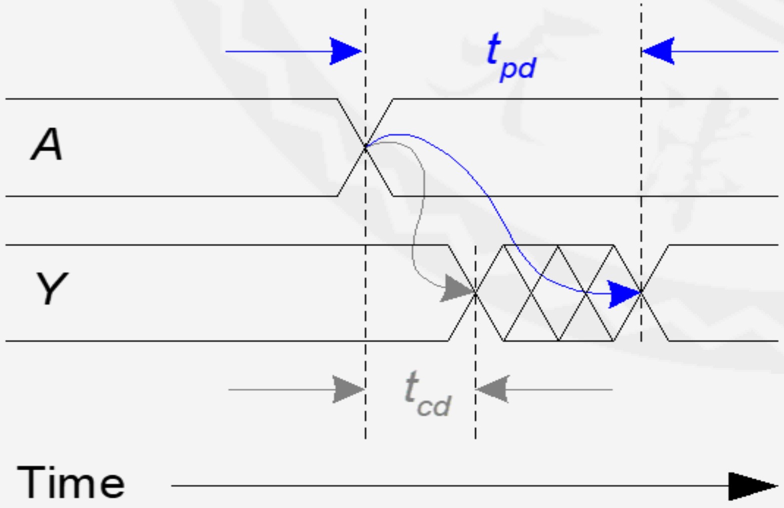
如何使电路运行得最快?

传播延迟和最小延迟

Propagation & Contamination Delay

定义

- 传播延迟（propagation delay）： t_{pd}
  - 输入改变直到一个或多个输出改变为最终值所经历的最长时间延迟
- 最小延迟（contamination delay）： t_{cd}
  - 输入发生变化直到任何一个输出开始改变的最短时间



传播延迟和最小延迟

Propagation & Contamination Delay

产生原因

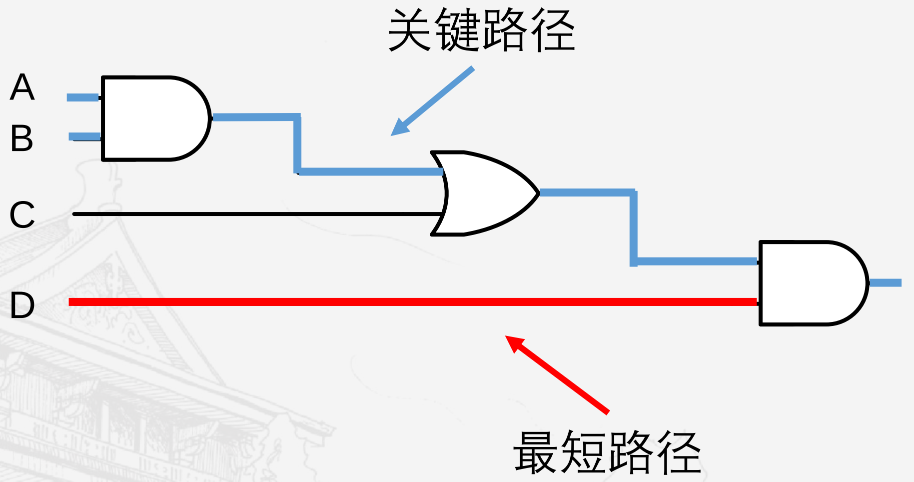
- 产生延迟的原因包括:
  - 电路中的电阻和电容的充放电
  - 光速的上限
- t_{pd}和t_{cd}的值可能不同
  - 上升沿与下降沿的延迟可能不同
  - 电路存在多个输入和输出时，不同输出的延迟可能不同
  - 电路对温度敏感，电路较热时速度会变慢

传播延迟和最小延迟

Propagation & Contamination Delay

关键路径与最短路径

- 关键路径（critical path）：信号传输最慢的一条路径
- 最短路径（short path）：信号通过最快的一条路径



关键路径：$t_{pd} = 2t_{pd\_AND} + t_{pd\_OR}$

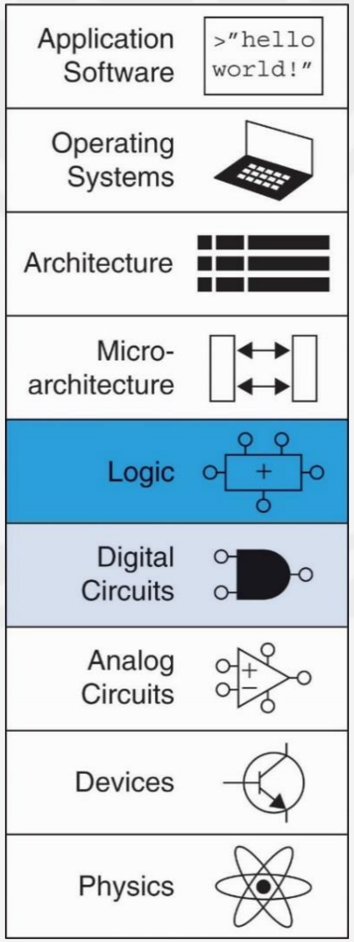
最短路径：$t_{cd} = t_{cd\_AND}$

组合逻辑中的时序问题

Timing in Combinational Logic

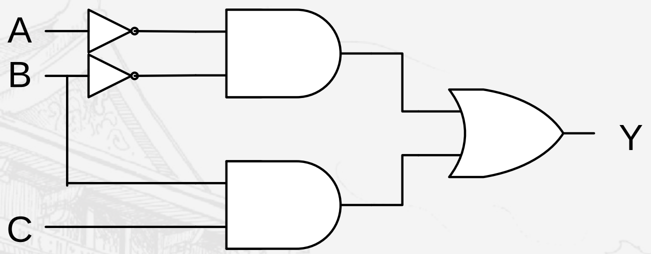
① 传播延迟和最小延迟

② “毛刺”



“毛刺”的产生

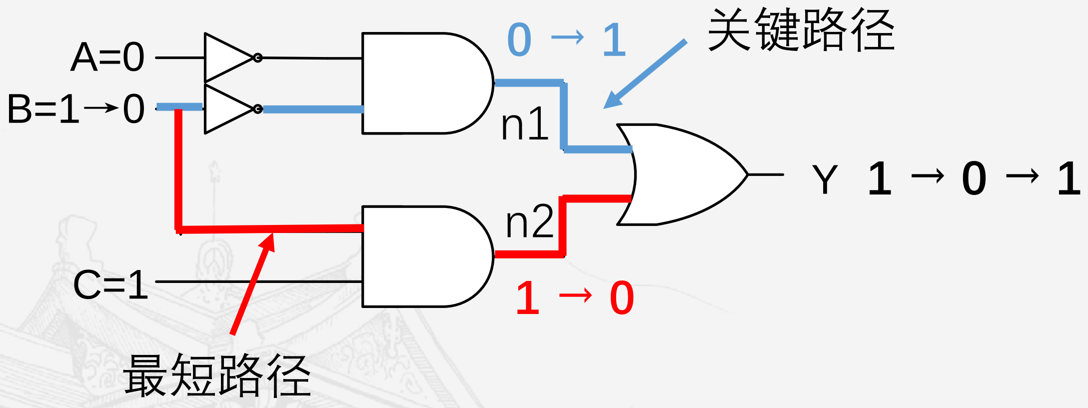
- 一个输入改变引起输出的多次变化
- 也称为“冒险”（hazard）
- 观察A = 0，C = 1时，B由1变0的瞬间发生了什么？

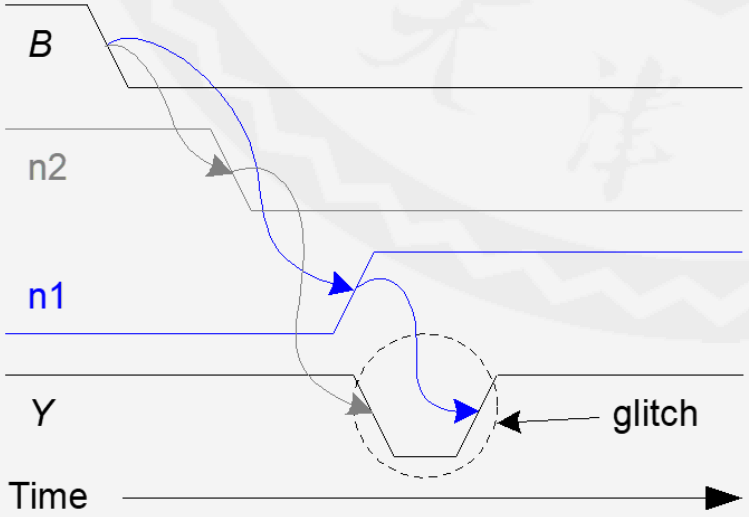


C \ AB (Y)	00	01	11	10
0	1	0	0	0
1	1	1	1	0

$$Y = \overline{A}\,\overline{B} + BC$$

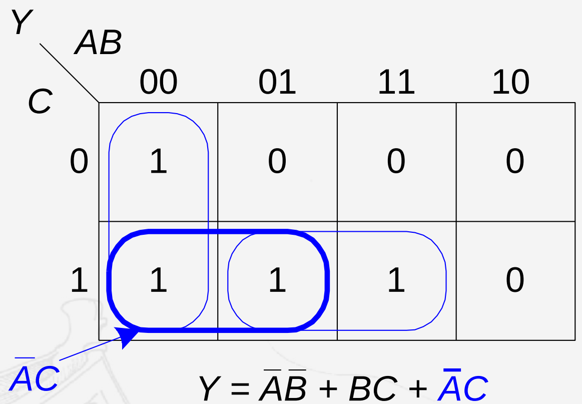
“毛刺” 产生的分析



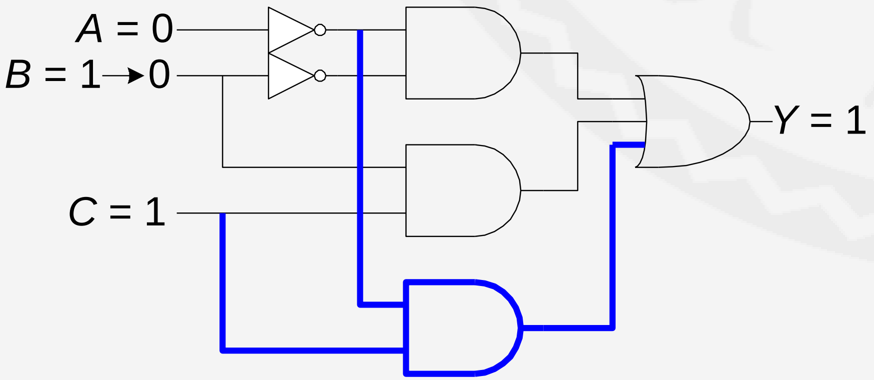


消除“毛刺”

- 当信号的变化在卡诺图中穿越2个主蕴含项的边缘时会出现“毛刺”
- 通过在卡诺图中增加多余的蕴含项来盖住这些边缘以避免毛刺



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

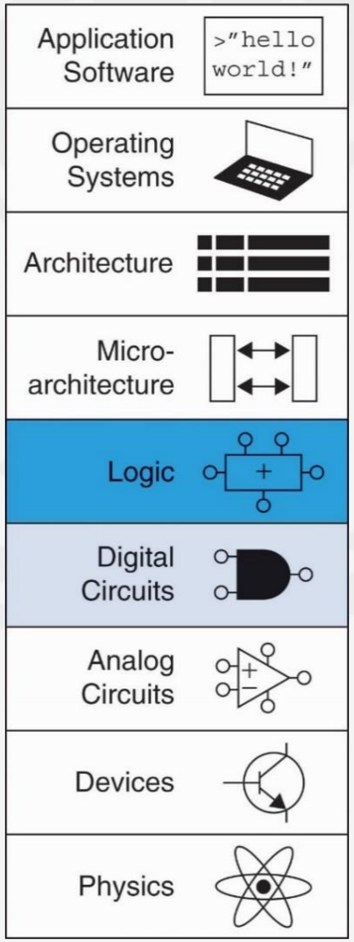


- 多个输入（几乎）同时变化也会产生“毛刺”
  - 这些不能通过增加硬件来避免
- 毛刺在大多数电路中都存在

本章内容

Topic

- 引言
- 布尔代数
- 卡诺图
- 从逻辑到门
- 多级组合逻辑
- X和Z
- 组合逻辑电路设计方法
- 组合逻辑中的时序问题
- 组合逻辑模块



组合逻辑模块

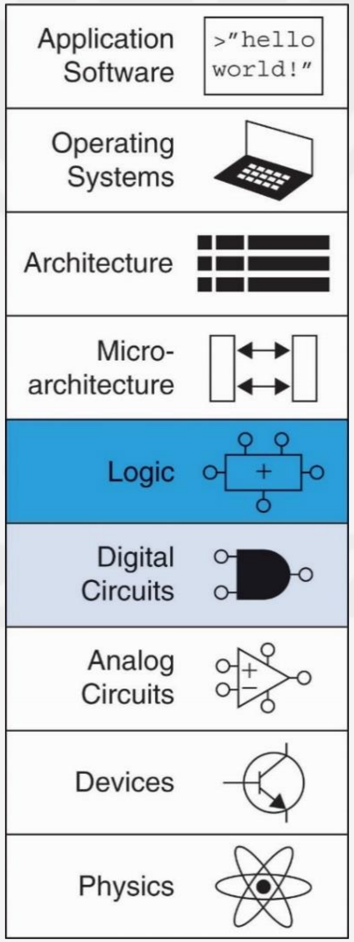
Combinational Building Blocks

① 编码器

② 译码器

③ 多路选择器

④ 算术电路



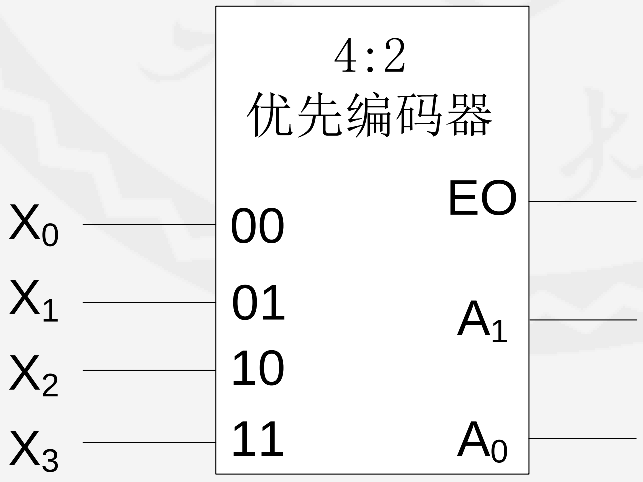
编码器

- 用n位二进制代码对 $N=2^n$ 个特定信息进行编码的逻辑电路
- 例：设计一个具有 4路信号输入的优先级编码器

  输入：X_0、X_1、X_2、X_3 (高电平有效)

  输出：A_1、A_0、EO（用于判定是否存在有效输入）

  功能：将4个输入信号进行二进制编码（4线-2线编码器）



编码器

Encoder

带输出使能的优先级编码器

■ 优先级编码

当有多个信号同时输入时,只对优先权高的一个信号进行编码

■ 输出使能端

用于判别电路是否有信号输入。用EO表示， EO=0有信号输入；EO=1无信号输入

输入	A_1	A_0	EO
无有效输入	0	0	1
X_0 有效且 X_1、X_2、X_3无效	0	0	0
X_1有效且 X_2、X_3无效	0	1	0
X_2有效且 X_3无效	1	0	0
X_3	1	1	0

编码器

Encoder

输入	A_1	A_0	EO
无有效输入	0	0	1
X_0 有效且 X_1、X_2、X_3无效	0	0	0
X_1有效且 X_2、X_3无效	0	1	0
X_2有效且 X_3无效	1	0	0
X_3	1	1	0

真值表

X_3	X_2	X_1	X_0	A_1	A_0	EO
0	0	0	0	0	0	1
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	1	1	0
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	1	1	0
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	0

使用X简化真值表

X_3	X_2	X_1	X_0	A_1	A_0	EO
0	0	0	0	0	0	1
0	0	0	1	0	0	0
0	0	1	X	0	1	0
0	1	X	X	1	0	0
1	X	X	X	1	1	0

编码器

Encoder

X_3	X_2	X_1	X_0	A_1	A_0	EO
0	0	0	0	0	0	1
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	1	1	0
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	1	1	0
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	0

$$EO = \overline{X_0}\,\overline{X_1}\,\overline{X_2}\,\overline{X_3} = \overline{X_0 + X_1 + X_2 + X_3}$$

$$A_0 = X_1\overline{X_2} + X_3$$

$$A_1 = X_2 + X_3$$

A_0

X_3X_2 \ X_1X_0	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

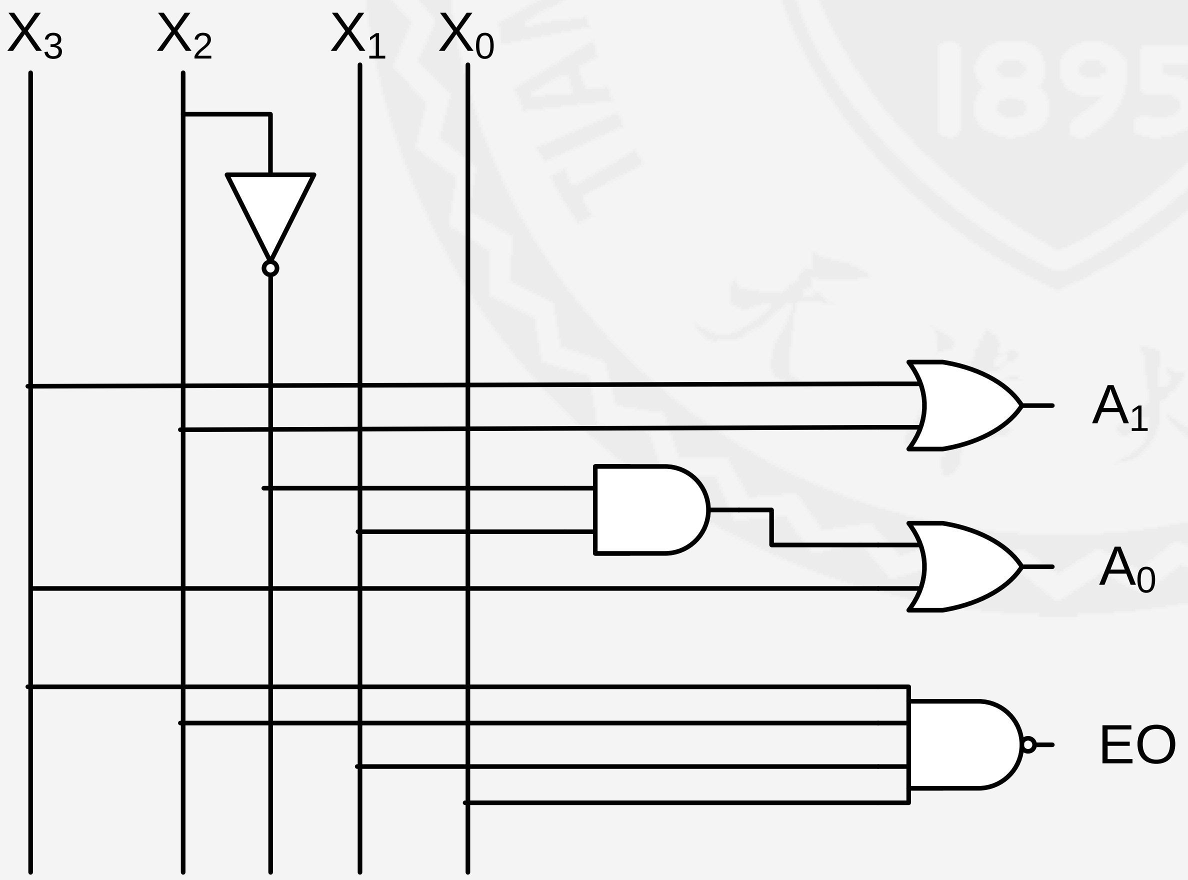
A_1

X_3X_2 \ X_1X_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

编码器

Encoder

$$\begin{cases} EO = \overline{X_0 + X_1 + X_2 + X_3} \\ A_0 = X_1\overline{X_2} + X_3 \\ A_1 = X_2 + X_3 \end{cases}$$



组合逻辑模块

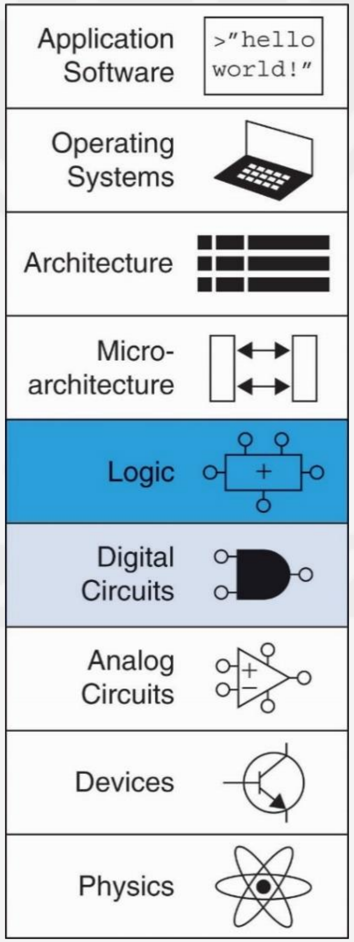
Combinational Building Blocks

① 编码器

② 译码器

③ 多路选择器

④ 算术电路

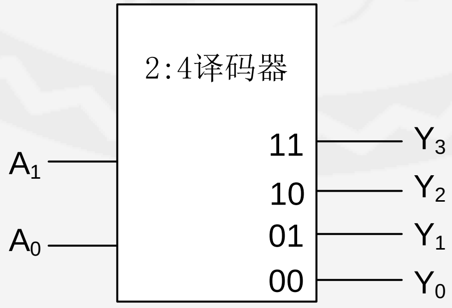


译码器

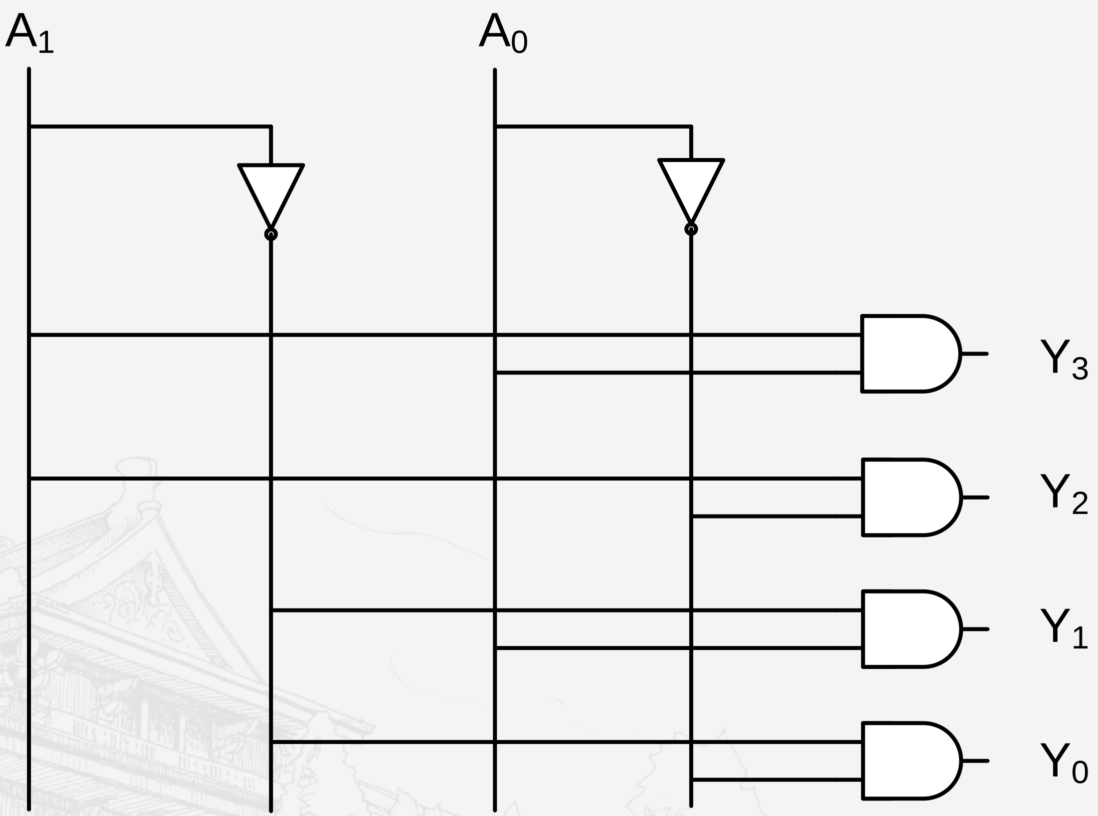
- 译码是编码的逆过程，有n个输入，2^n个输出
- 每一种输入的组合对应使能某个特定的输出信号
- 输出是独热（one-hot，互斥）的，同一时刻只能输出一个有效信号

2线-4线译码器（高电平为有效信号）

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



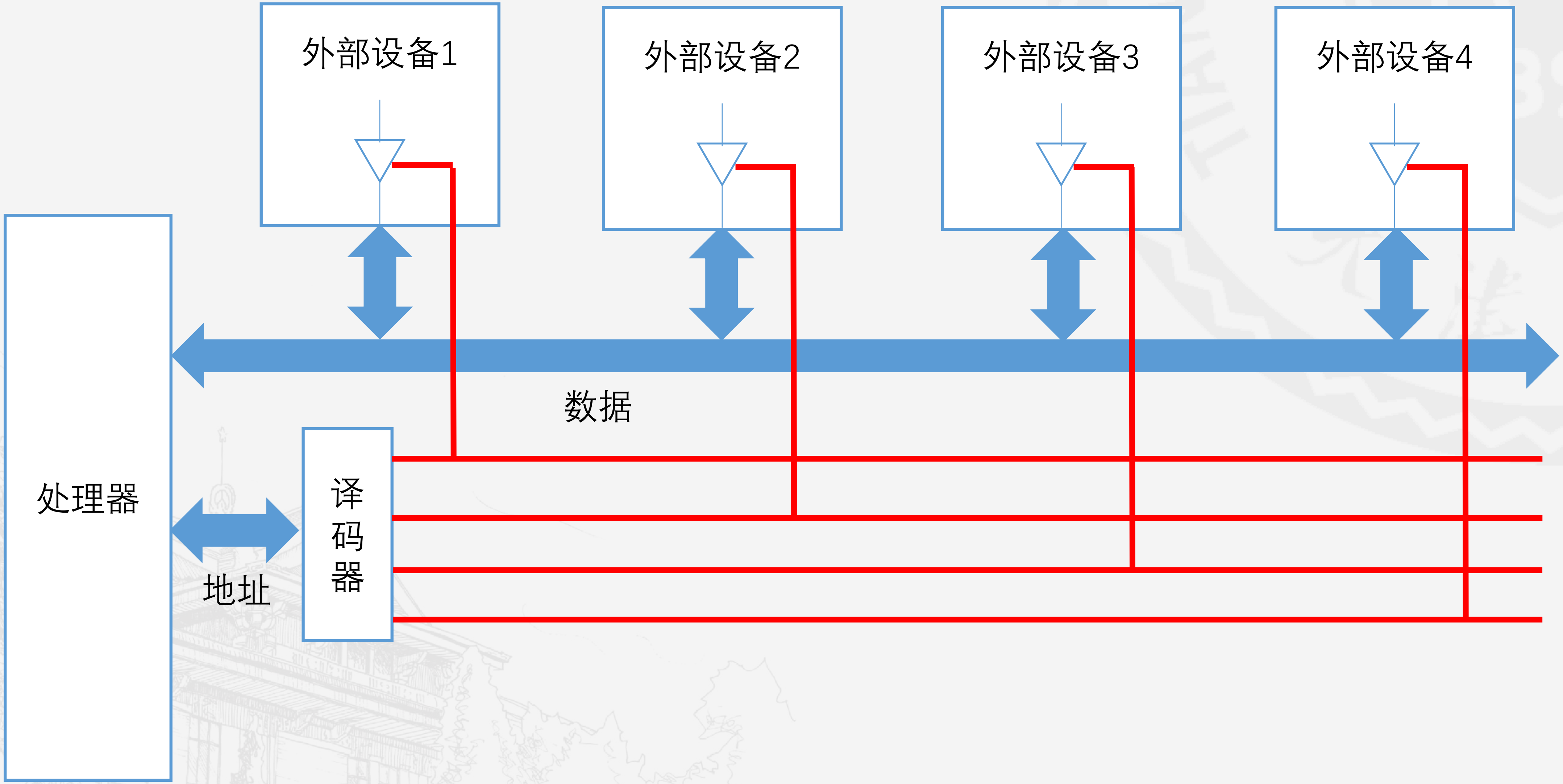
译码器电路的实现



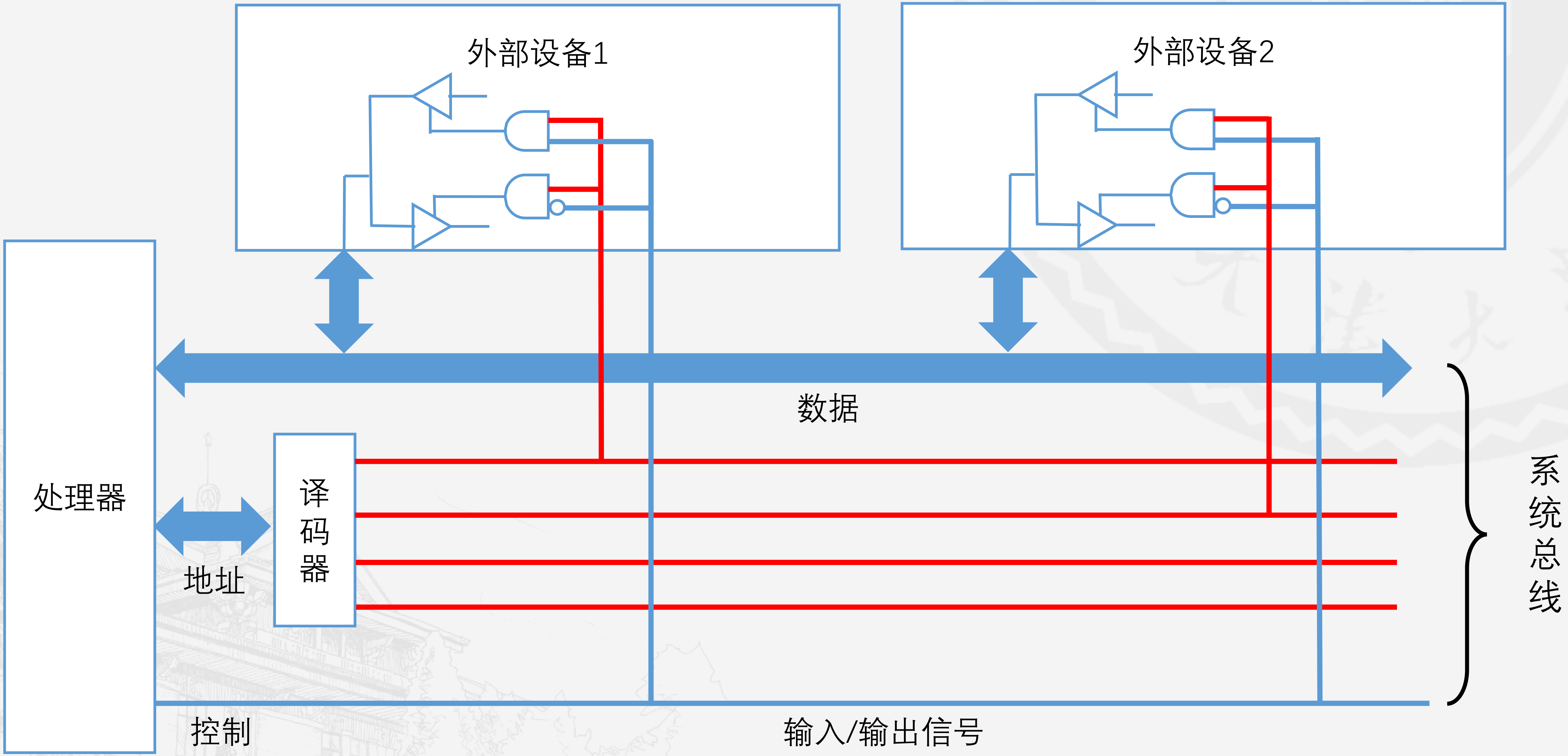
Y_0，Y_1，Y_2，Y_3分别对应

m_0，m_1，m_2，m_3四个最小项

译码器的应用

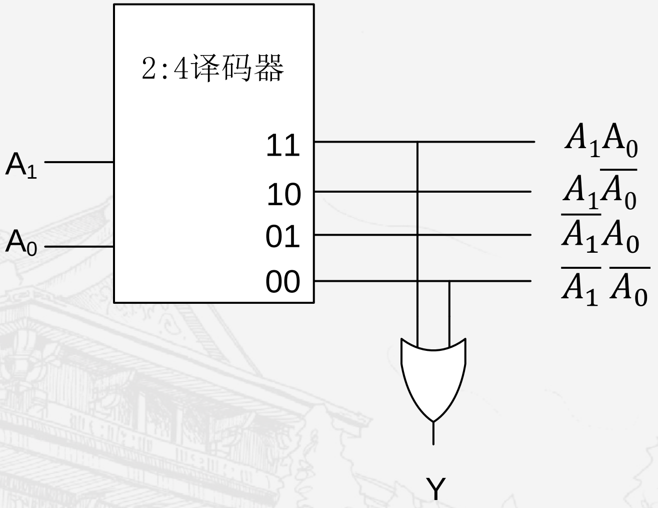


译码器的应用（cont.）



使用译码器实现复杂逻辑

- 译码器每个输出都对应一个最小项
- 使用译码器+或门可以构造出更加复杂的表达式



$$Y = A_1A_0 + \overline{A_1}\,\overline{A_0} = \overline{A_1 \oplus A_0} \quad （同或）$$

组合逻辑模块

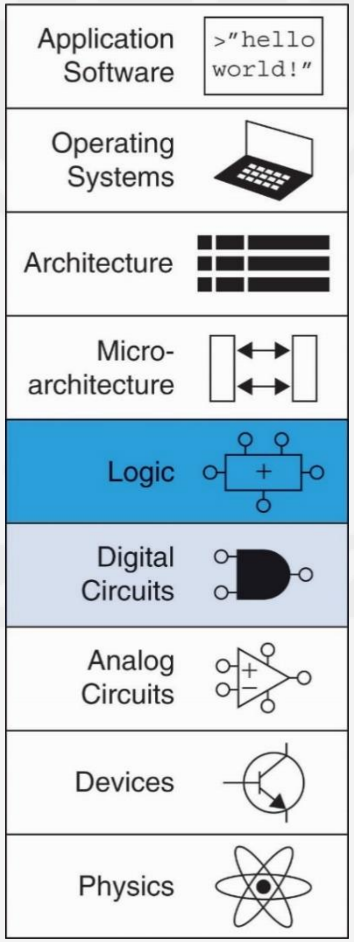
Combinational Building Blocks

① 编码器

② 译码器

③ 多路选择器

④ 算术电路

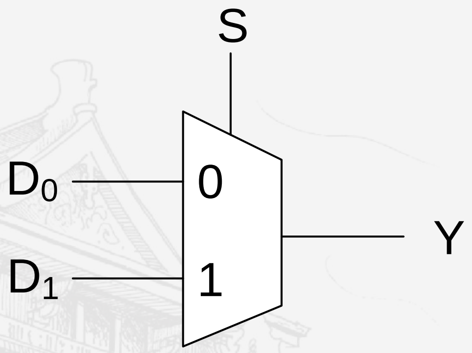


多路选择器

Multiplexers

多路选择器

- 根据选择信号的值从 N 个可能的输入中选择一个作为输出
- 需要使用$log_2 N$位选择信号作为输入，控制输入信号的选择



二选一电路

S	Y
0	D_0
1	D_1

S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

多路选择器

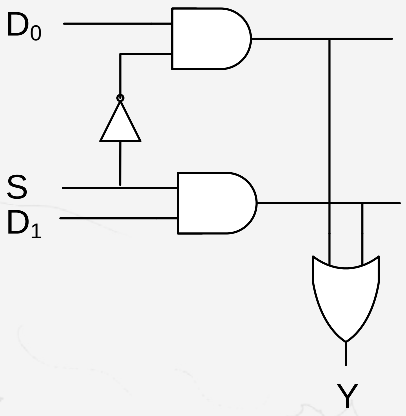
Multiplexers

多路选择器的实现

使用门电路实现

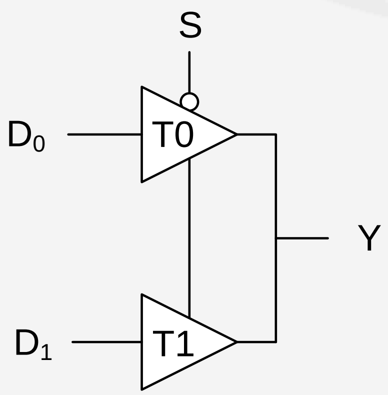
A \ D_0D_1	00	01	11	10
0			1	1
1		1	1	

$$Y = D_0\bar{S} + D_1S$$



使用三态门实现

- N输入的选择器，使用N个三态门
- 对选择信号进行译码以使能对应的三态门进行输出

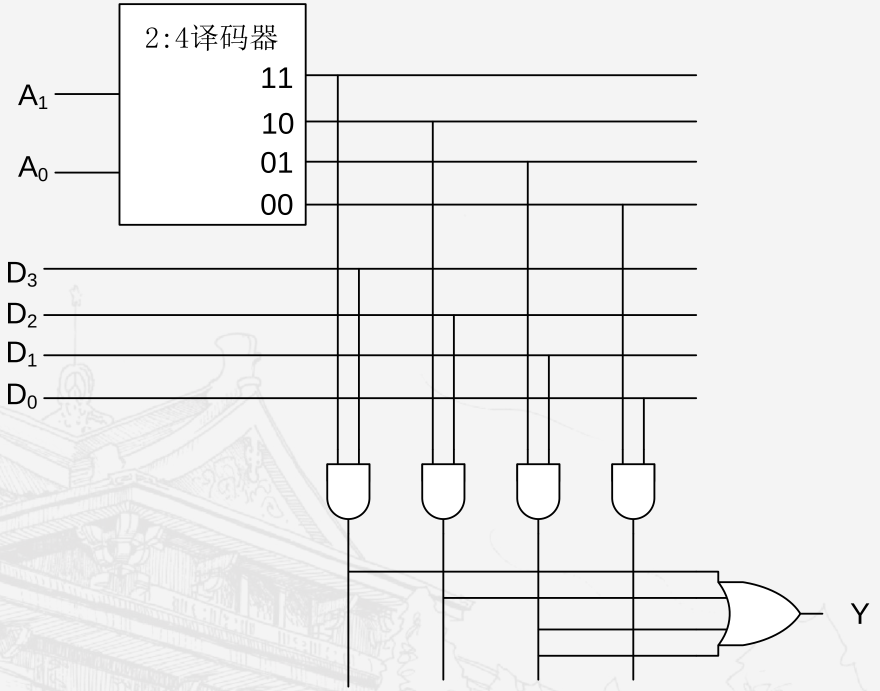


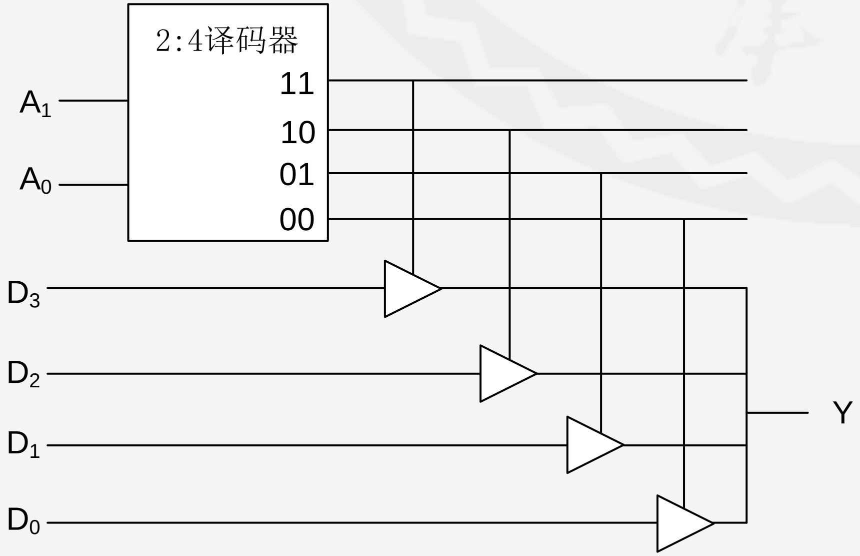
多路选择器

Multiplexers

更多输入的多路选择器

- 可使用译码器实现





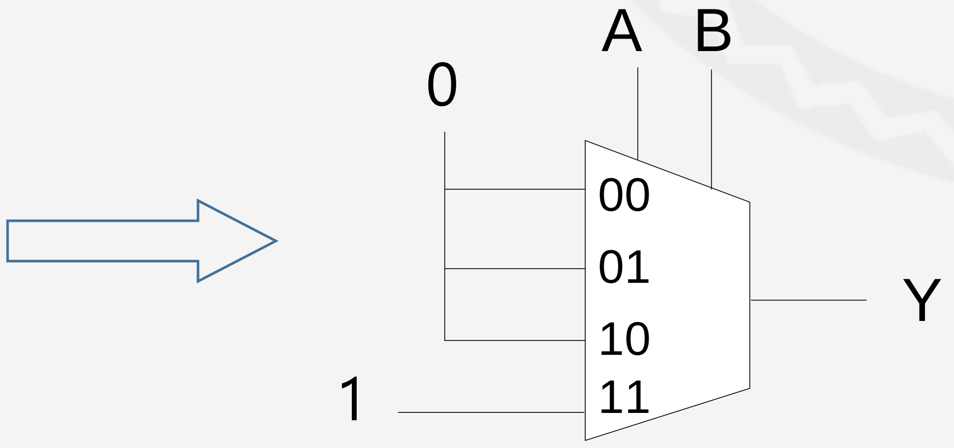
多路选择器

Multiplexers

使用多路选择器实现复杂逻辑

- 可以将多路选择器看做一个查找表

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



组合逻辑模块

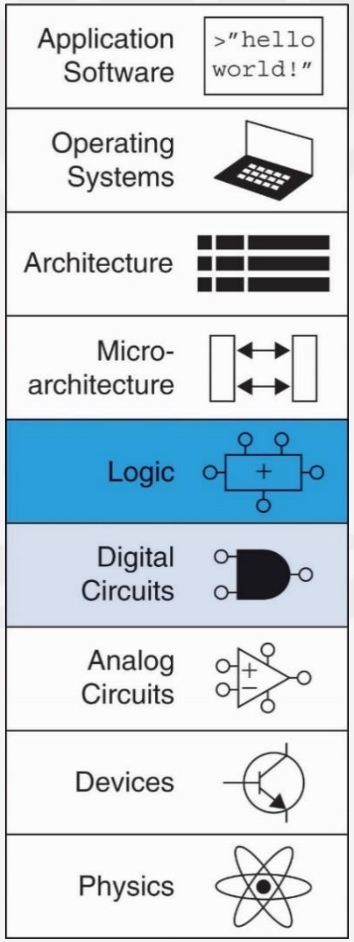
Combinational Building Blocks

① 编码器

② 译码器

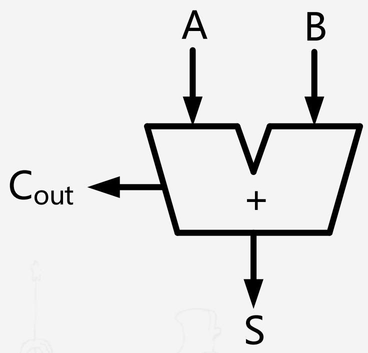
③ 多路选择器

④ 算术电路



1位加法器 — — 半加器（half-adder）

■ 半加器（half-adder）有两个输入A和B，两个输出S和C_{out}。S是A和B之和，C_{out}为进位。



A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

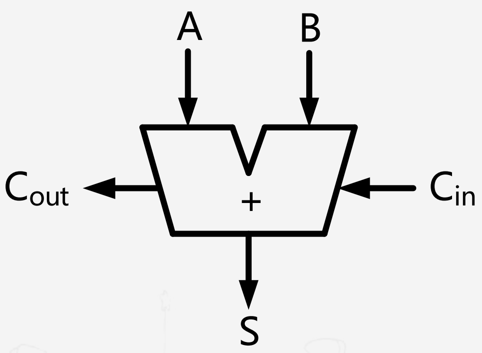
$S = A \oplus B$

$C_{out} = A \,\&\, B$

■ 如果用1位半加器设计多位加法器，则半加器存在一个问题，即它缺少一个进位输入C_{in}来接收前一个半加器的进位输出C_{out}。

1位加法器 —— 全加器（full-adder）

■ 全加器（full-adder）在半加器的基础之上增加了一个进位输入C_{in}。



A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

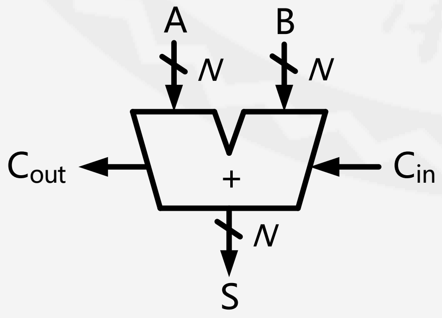
$S = A \oplus B \oplus C_{in}$

$C_{out} = A \ \& \ B+(A \oplus B) \ \& \ C_{in}$

多位加法器（CPAs）

■ 1个N位加法器将两个N位输入（A和B）与一个进位C_{in}相加，产生一个N位结果S和一个输出进位C_{out}。因为在N位加法器内部，1位进位将传播到下一位，所以这种加法器通常称为**进位传播加法器（Carry Propagate Adder, CPA）**。

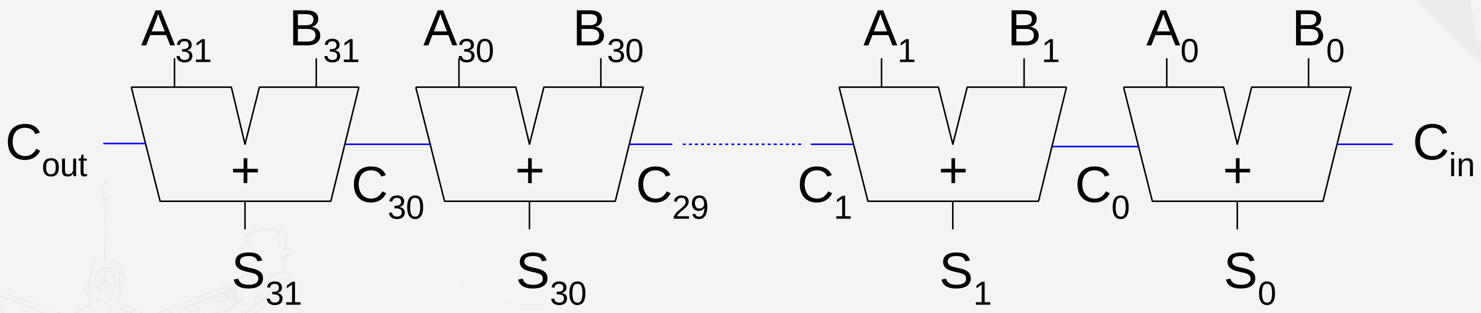
- ▶ 行波进位加法器（慢速）
- ▶ 先行进位加法器（快速）
- ▶ 前缀加法器（更快速）



■ 对于高位宽的加法器，先行进位和前缀更具优势，但需要消耗更多的硬件资源（**请大家时刻谨记：任何工程设计都体现了折中的思想（Tradoff）！**）。

CPAs —— 行波进位加法器

- 最简单的CPA就是将N个全加器串联，称为行波进位加法器（ripple-carry adder）,每级的C_{out}就是下一级的C_{in}，则所有进位构成的通路称为**进位链**。



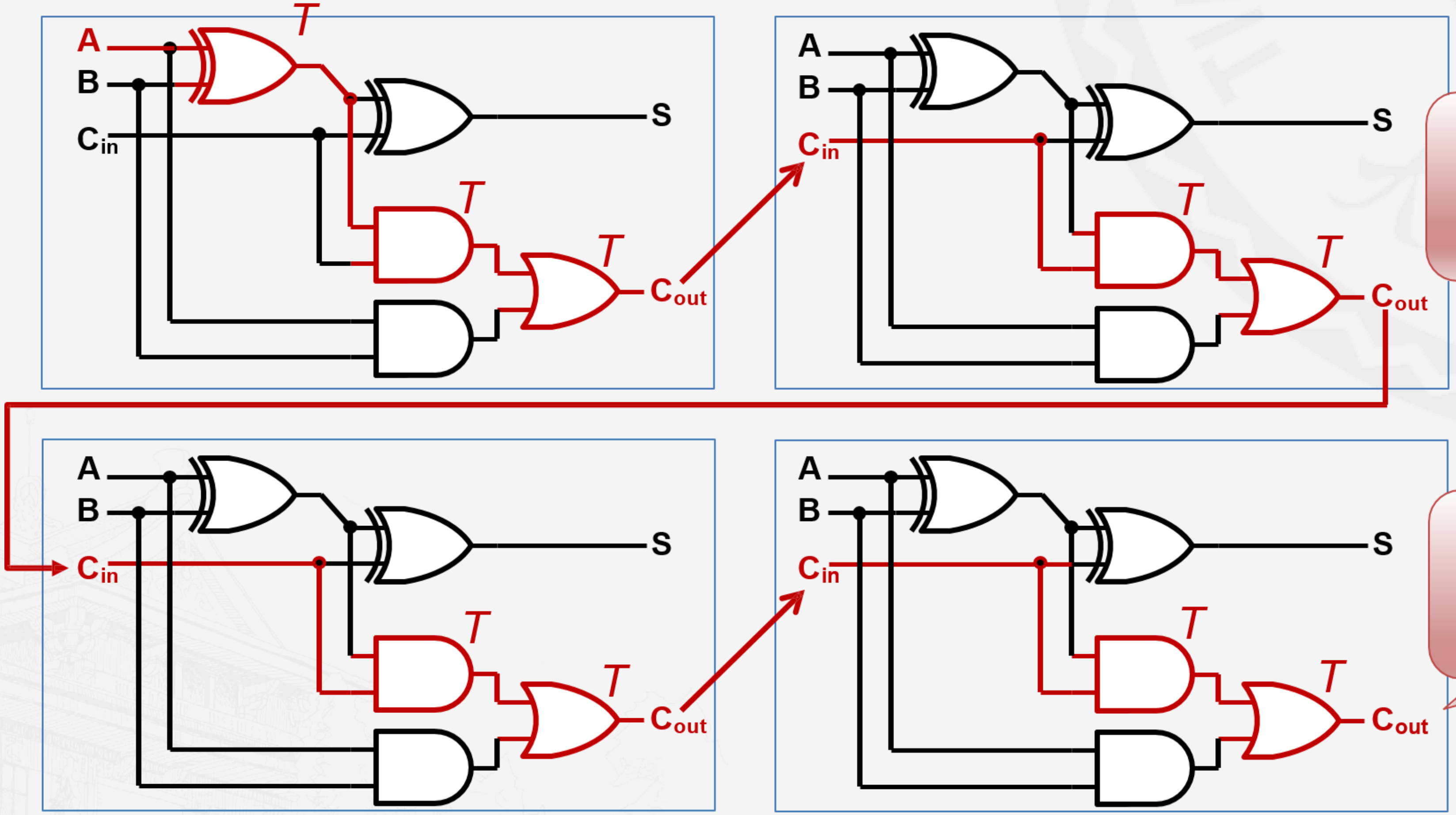
这是一个近似延迟，请问精确延迟是多少?

$t_{ripple} = N \cdot t_{FA}$

t_{FA}是全加器的延迟

- 这种加法器最大的缺点：由于进位是一级一级的从低位传输到高位，当N较大时，计算延迟也较大，运算速度较慢，即延迟t_{ripple}随位数增加而增加。

CPAs — — 行波进位加法器（cont.）



总延迟时间
$(T+T)\times n+T$
$=(2n+1)T$

总延迟时间
$(T+T)\times 4+T$
$=9T$

CPAs —— 先行进位加法器

■ **先行进位加法器（Carry-lookahead Adder, CLA）**是一种**快速**的进位传播加法器。它把加法器分解成若干块，当每块一有进位时就快速确定此块的输出进位。因此，它不需要等待进位通过一块内的所有加法器，而是直接先行通过该块。

■ 先考虑每一位（一列）的进位输出如何确定。使用产生（G）和传播（P）两个信号来描述。

▶ 第 i 位**产生**一个进位，如果A_i和B_i均为“1”，即$G_i = A_iB_i$；

▶ 第 i 位**传播**一个进位，如果有进位输入，并且A_i或B_i为“1”，即$P_i = A_i \oplus B_i$；

▶ 综上定义，加法器第i位产生进位输出的表达式为：$C_i = A_iB_i + (A_i \oplus B_i)C_{i-1} = G_i + P_iC_{i-1}$；

CPAs — — 先行进位加法器（cont.）

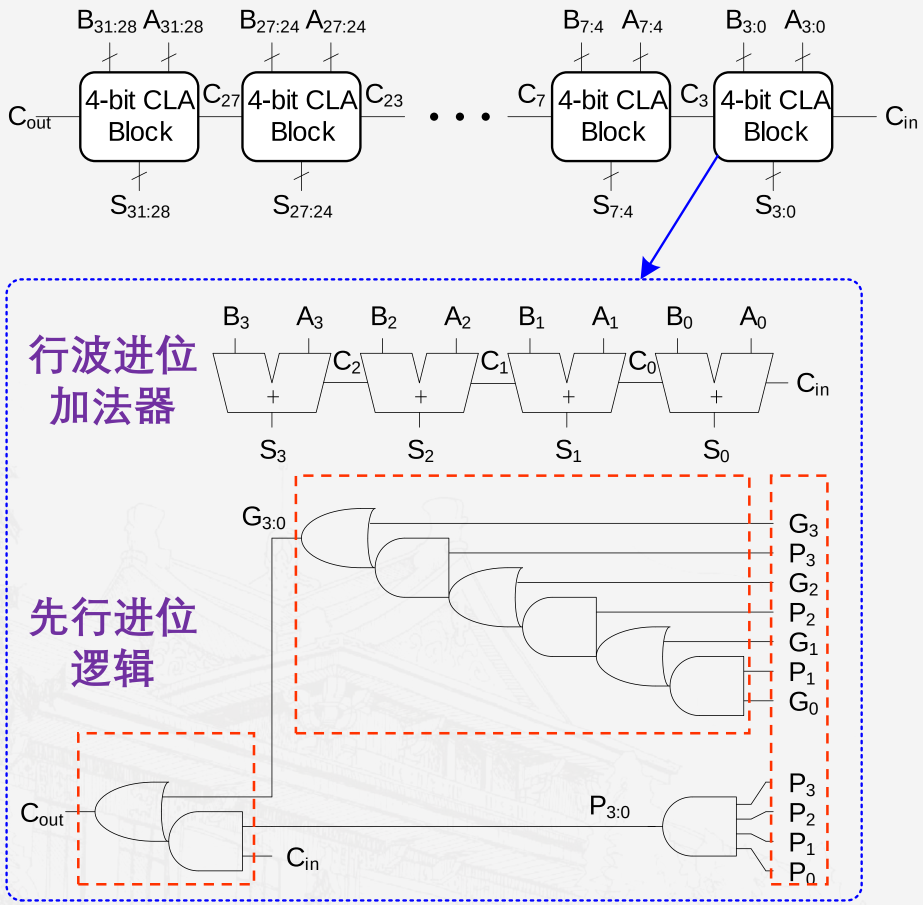
■ N位加法器按每k位分为一块，可将产生和传播信号的定义扩展至该多位块。假设**每块4位**，$G_{i,j}$ 和$P_{i,j}$表示从第i位到第j位这一块的产生信号和传播信号。

$$G_{3:0} = G_3 + P_3(G_2 + P_2(G_1 + P_1G_0))$$
$$P_{3:0} = P_3P_2P_1P_0$$
$$\rightarrow \quad G_{i:j} = G_i + P_i(G_{i-1} + P_{i-1}(G_{i-2} + P_{i-2}G_j))$$
$$P_{i:j} = P_iP_{i-1}P_{i-2}P_j$$
$$\rightarrow \quad C_i = G_{i:j} + P_{i:j}C_j$$

■ 结论：**所有的G和P都可并行的计算得到**，只要某一块获得了上一级的进位就可以快速确定它的输出进位，块间进位的生成速度大幅提升。块内进位如下所示，可见**块内部分和**也可并行计算。

$$C_0 = G_0 + P_0C_{in}$$
$$C_1 = G_1 + P_1C_0 = G_1 + P_1(G_0 + P_0C_{in}) = G_1 + P_1G_0 + P_1P_0C_{in}$$
$$C_2 = G_2 + P_2C_1 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_{in}$$

CPAs —— 先行进位加法器（cont.）



N位先行进位加法器，按k位分块，其延迟为：

$$t_{CLA} = t_{pg} + t_{pg\_block} + (N/k - 1)t_{AND\_OR} + kt_{FA}$$

t_{pg}：产生所有P_i和G_i的延迟（单个AND或OR门）

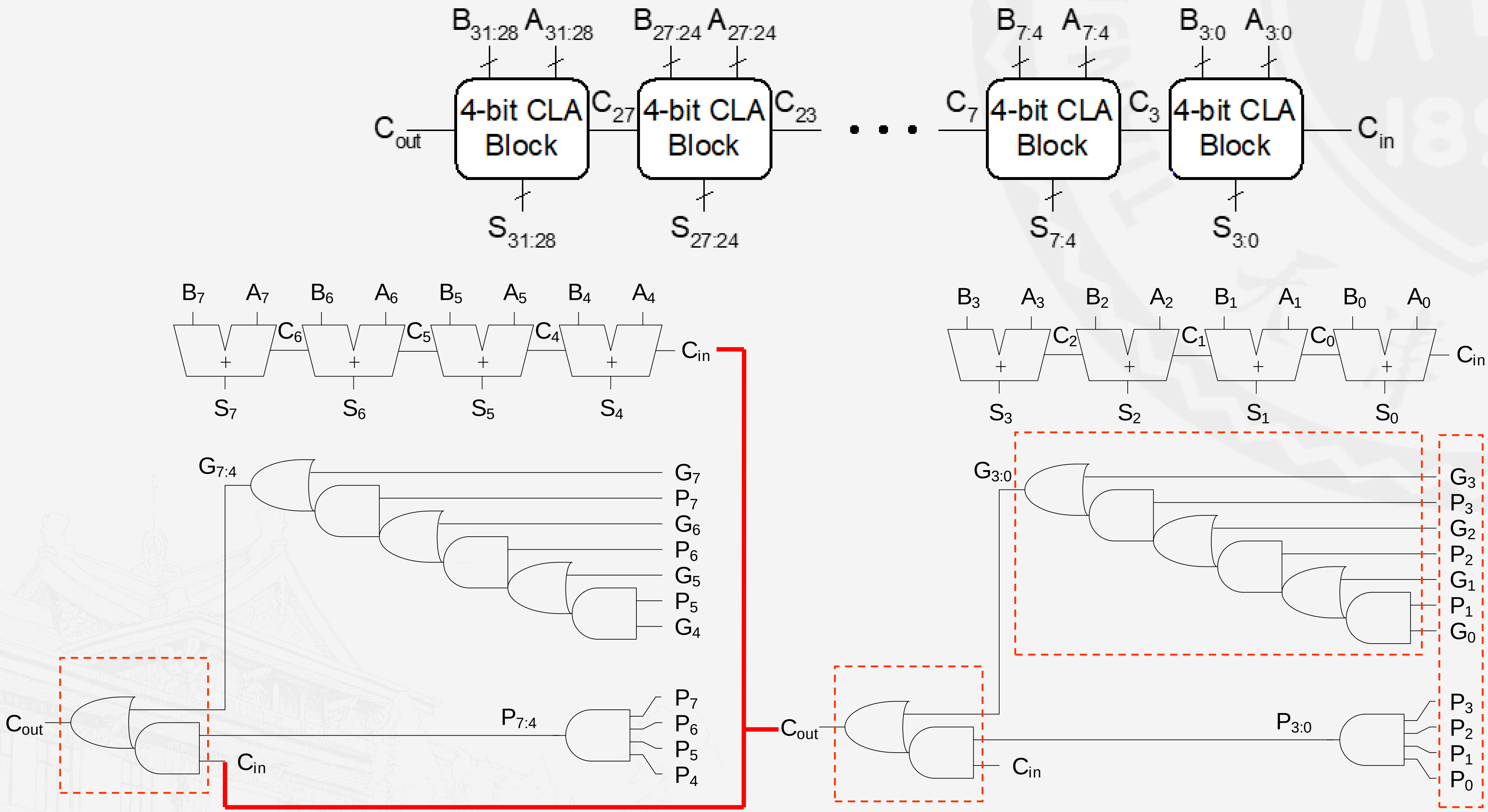
t_{pg_block}：产生所有$P_{i:j}$和$G_{i:j}$的延迟

t_{AND_OR}：连通（从C_{in}到C_{out}）所有k位CLA最后与门/或门的延迟

算术电路

Arithmetic circuits

$$t_{CLA} = t_{pg} + t_{pg\_block} + (N/k - 1)t_{AND\_OR} + kt_{FA}$$



加法器的延迟比较

- 对于32位行波进位加法器和4位块组成的32位先行进位加法器的延迟。假设每个两输入门电路的延迟为100ps，全加器的延迟是300ps。

32位行波进位加法器的传播延迟： $t_{ripple} = Nt_{FA} = 32 \times 300ps = 9.6ns$

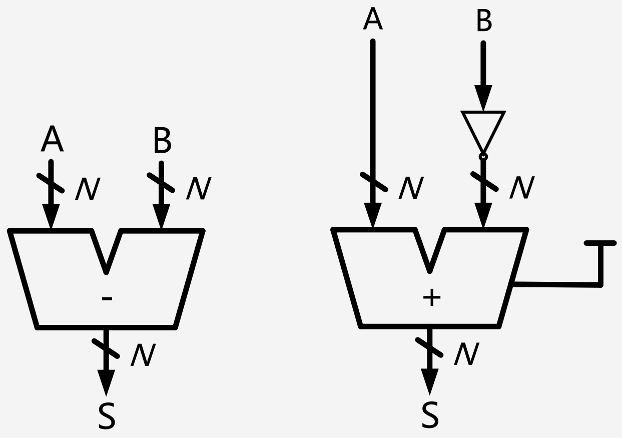
32位先行进位加法器的传播延迟： $t_{CLA} = t_{pg} + t_{pg\_block} + (N/k - 1)t_{AND\_OR} + kt_{FA}$

$= (100 + 600 + 7 \times 200 + 4 \times 300)\ ps$

$= 3.3ns$

减法器

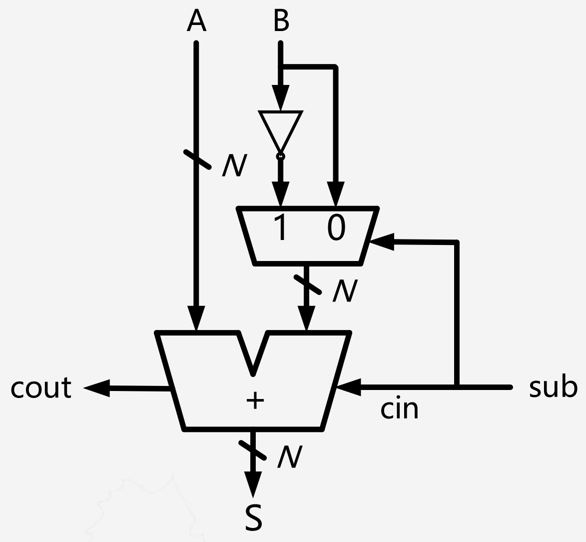
- 补码的减法可以表示为“$[A - B]_{补} = [A + (-B)]_{补} = [A]_{补} + [-B]_{补} = [A]_{补} + \overline{[B]}_{补} + 1$”，由此可见，减法器可以由加法器进行实现。



符号　　实现

减法器（cont.）

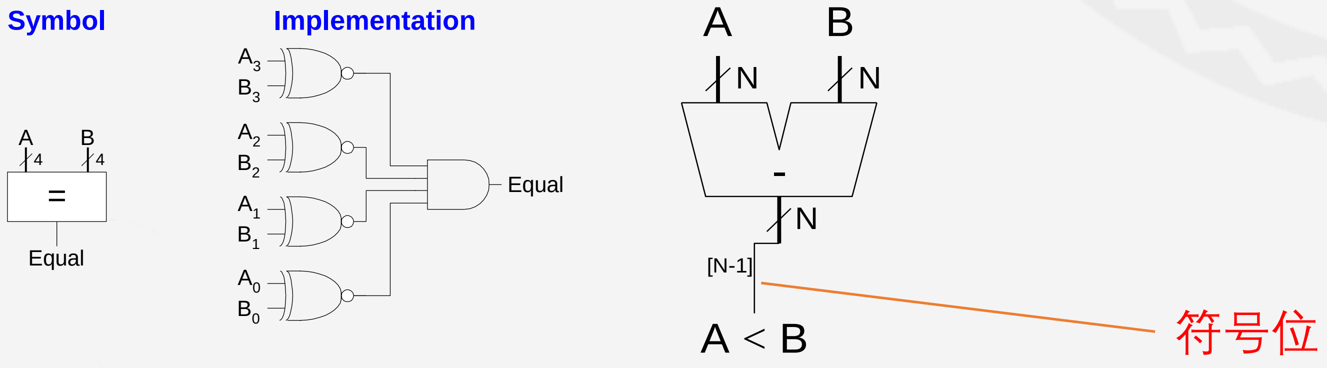
- 在计算机中所有的数据都用补码表示，因此实际计算机中是没有减法器的，加法和减法都通过加法器实现，如下所示



比较器

■ 比较器是判断两个N位二进制数A和B是否相等，或者一个比另一个大还是小。常见有两种类型：

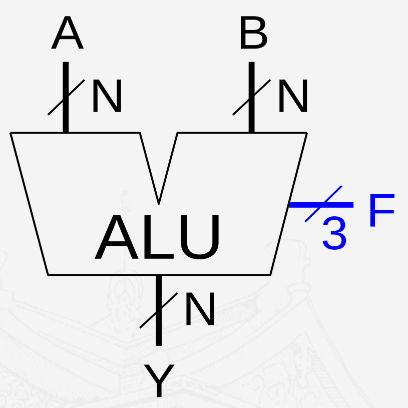
- ▶ 相等比较器，产生一个输出，表示A是否等于B（A == B）。
- ▶ 数值比较器，产生一个或多个输出，表示A和B的关系（>，<等）。



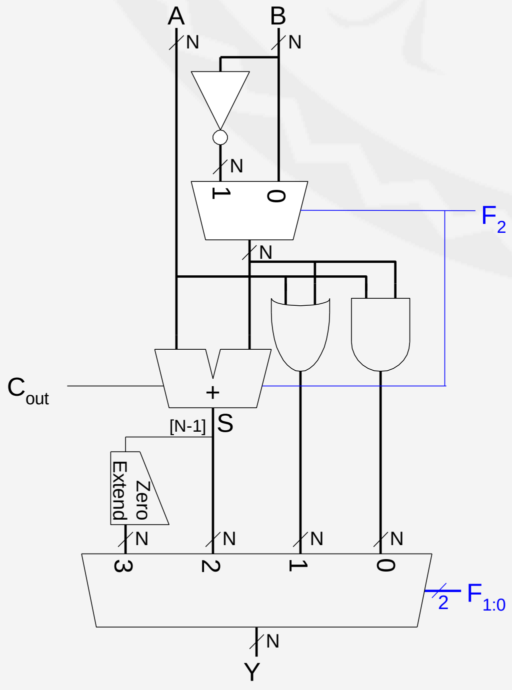
相等比较器　　数值比较器

算术逻辑单元

■ 算术逻辑单元（Arithmetic and Logical Unit, ALU）将多种算术和逻辑运算组合到一个单元模块中。典型的ALU可以执行加法、减法、量值比较、逻辑运算等。ALU是大多数计算机的核心。



$F_{2:0}$	功能
000	A & B
001	A | B
010	A + B
011	not used
100	A & ~B
101	A | ~B
110	A – B
111	SLT



SLT（小于则置位）操作，当A<B时，Y = 1；否则Y = 0.

移位器和循环移位器

- 逻辑移位器：将数据向左（LSL）或向右（LSR）移动指定位数，空出的位置补“0”。

  11001 >> 2 = 00110

  11001 << 2 = 00100

- 算术移位器：算术左移（ASL）和LSL相同，算术右移（ASR）时使用原数据的最高位填充空位。

  11001 >> 2 = 11110

  11001 << 2 = 00100

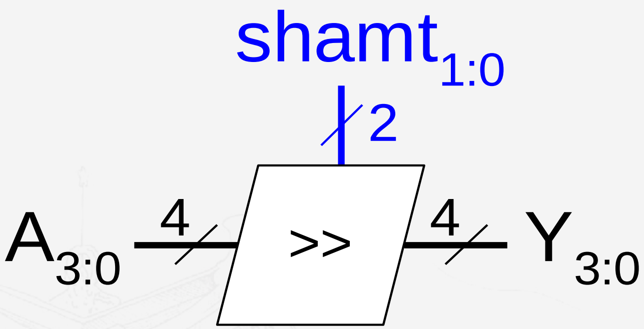
- 循环移位器：循环移动数据，从一端移走位重新填充到另一端的空位上。

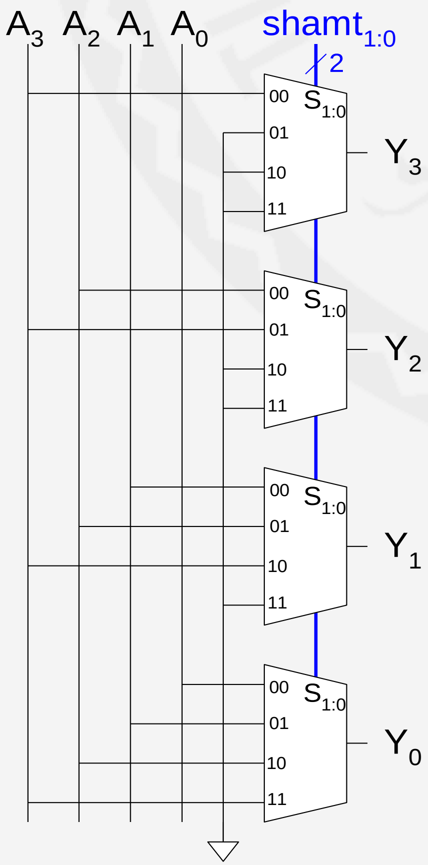
  11001 ROR 2 = 01110

  11001 ROL 2 = 00111

移位器和循环移位器（cont.）

■ N位移位器可以用N个N:1多路选择器构成。





乘法器

- **无符号二进制数**的乘法和十进制的乘法很相似，可通过移位和加法实现。

Decimal		Binary
230	multiplicand	0101
x 42	multiplier	x 0111
460	partial products	0101
+ 920		0101
		0101
		+ 0000
9660	result	0100011

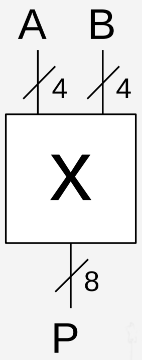
230 x 42 = 9660   5 x 7 = 35

有符号二进制数的乘法应如何计算?

- **两个N位二进制数相乘，产生一个2N位的结果**，其部分积要么是被乘数，要么全部是“0”。1位二进制数乘法相当于AND运算，所以可以使用AND门电路产生部分积。

无符号二进制数乘法器

■ **无符号二进制数**的乘法和十进制的乘法很相似，可通过移位和加法实现。



					A_3	A_2	A_1	A_0
x					B_3	B_2	B_1	B_0
					A_3B_0	A_2B_0	A_1B_0	A_0B_0
				A_3B_1	A_2B_1	A_1B_1	A_0B_1	
			A_3B_2	A_2B_2	A_1B_2	A_0B_2		
+		A_3B_3	A_2B_3	A_1B_3	A_0B_3			
	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0