



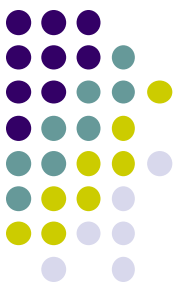
# 计算机组成与系统结构

## 第二章 运算方法和运算器 (1)

吕昕晨

[lvxinchen@bupt.edu.cn](mailto:lvxinchen@bupt.edu.cn)

网络空间安全学院



# 任意进制：r进制计数法

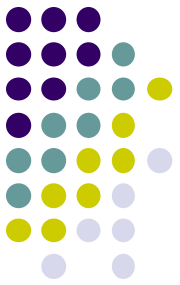
r 进制：  $K_n K_{n-1} \dots K_2 K_1 K_0 K_{-1} K_{-2} \dots K_{-m}$

位权

$$= K_n \times r^n + K_{n-1} \times r^{n-1} + \dots + K_2 \times r^2 + K_1 \times r^1 + K_0 \times r^0 \\ + K_{-1} \times r^{-1} + K_{-2} \times r^{-2} + \dots + K_{-m} \times r^{-m}$$

**基数**：每个数码位所用到的不同符号的个数，r 进制的基数为 r

- 常见基数选择
  - R=2（二进制）、8（八进制）、10（十进制）、16（十六进制）
  - 二进制：冯诺依曼体系结构要求，方便逻辑电路运算



# 常见计数法

二进制: 0,1

八进制: 0,1,2,3,4,5,6,7

十进制: 0,1,2,3,4,5,6,7,8,9

十六进制: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

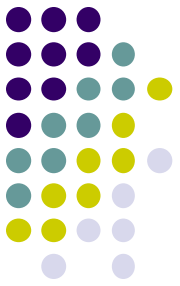
实数**5.5**多进制表示

二进制:  $101.1 \rightarrow 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} = 5.5$

八进制:  $5.4 \rightarrow 5 \times 8^0 + 4 \times 8^{-1} = 5.5$

十进制:  $5.5 \rightarrow 5 \times 10^0 + 5 \times 10^{-1} = 5.5$

十六进制:  $5.8 \rightarrow 5 \times 16^0 + 8 \times 16^{-1} = 5.5$



# 常见计数法表示

二进制——  $(1010001010010)_2$

1010001010010**B**

八进制——  $(1652)_8$

1652**O**

十六进制——  $(1652)_{16}$

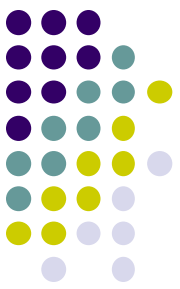
1652**H**

**0x**1652

十进制——  $(1652)_{10}$

1652**D**

- H: Hexadecimal (十六进制)、D: Decimal (十进制)
- O: Octal (八进制)、B: Binary (二进制)



# 任意进制→十进制

- 基于R进制数值公式

r 进制:  $K_n K_{n-1} \dots K_2 K_1 K_0 K_{-1} K_{-2} \dots K_{-m}$  位权

$$= K_n \times r^n + K_{n-1} \times r^{n-1} + \dots + K_2 \times r^2 + K_1 \times r^1 + K_0 \times r^0 \\ + K_{-1} \times r^{-1} + K_{-2} \times r^{-2} + \dots + K_{-m} \times r^{-m}$$

- 常见转换示例

二进制: 10010010.110

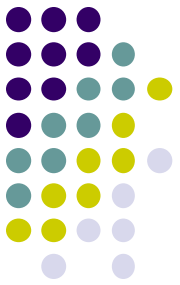
$$1 * 2^7 + 1 * 2^4 + 1 * 2^1 + 1 * 2^{-1} + 1 * 2^{-2} = 146.75$$

八进制: 251.5

$$2 * 8^2 + 5 * 8^1 + 1 * 8^0 + 5 * 8^{-1} = 168.625$$

十六进制: AE86.1

$$10 * 16^3 + 14 * 16^2 + 8 * 16^1 + 6 * 16^0 + 1 * 16^{-1} = 44678.0625$$



# 二进制 $\leftrightarrow$ 八/十六进制 (1)

- 二进制 $\leftrightarrow$ 八进制：1位8进制=3位2进制

二进制  $\rightarrow$  八进制

3位一组，每组转换成对应的八进制符号

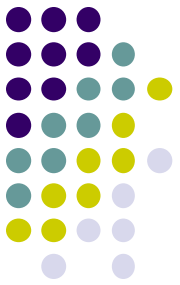
001 111 000 010 . 011 010

1    7    0    2    .    3    2    八进制

八进制 $\rightarrow$  二进制

每位八进制对应的3位二进制

$(251.5)_8 \rightarrow (010\ 101\ 001.101)_2$



# 二进制 $\leftrightarrow$ 八/十六进制 (2)

- 二进制 $\leftrightarrow$ 十六进制：1位16进制=4位2进制

二进制  $\rightarrow$  十六进制

4位一组，每组转换成对应的十六进制符号

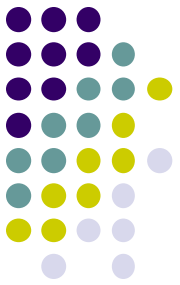
0011 1100 0010 . 0110 1000

3      C      2      .      6      8      十六进制

十六进制 $\rightarrow$  二进制

每位十六进制对应的4位二进制

$(AE86.1)_{16} \rightarrow (1010\ 1110\ 0110.0001)_2$



# 十进制→任意进制（整数部分）

- 十进制：整数部分+小数部分，分别转换
- 如：75.3转换为2进制
- 整数部分75转换：**除基取余**

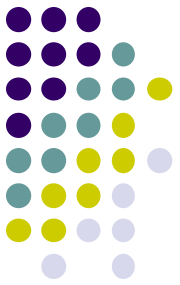
除基	取余
2   75	1
2   37	1
2   18	0
2   9	1
2   4	0
2   2	0
2   1	1
0	

↑ 低位

↓ 高位

$$(75)_{10} = (1001011)_2$$





# 十进制→任意进制 (小数部分)

- 如：75.3转换为2进制
- 小数部分0.3转换：乘基取整

$$0.3 \times 2 = 0.6 = \boxed{0} + \boxed{0.6} \quad K_{-1}$$

$$0.6 \times 2 = 1.2 = \boxed{1} + \boxed{0.2} \quad K_{-2}$$

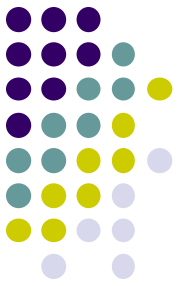
$$0.2 \times 2 = 0.4 = \boxed{0} + \boxed{0.4} \quad K_{-3}$$

$$0.4 \times 2 = 0.8 = \boxed{0} + \boxed{0.8} \quad K_{-4}$$

$$0.8 \times 2 = 1.6 = \boxed{1} + \boxed{0.6} \quad K_{-5}$$

循环

$$0.3D = 0.01001... B$$



# 多进制数间比较大小

- 直观方式：均转换成十进制

题目：

比较以下四个数的大小，最大的是（ ）

A. 1010B

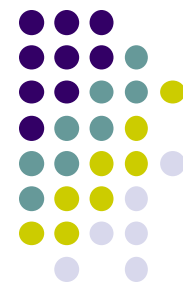
B. 26O

C. 10D

D. 0x1B

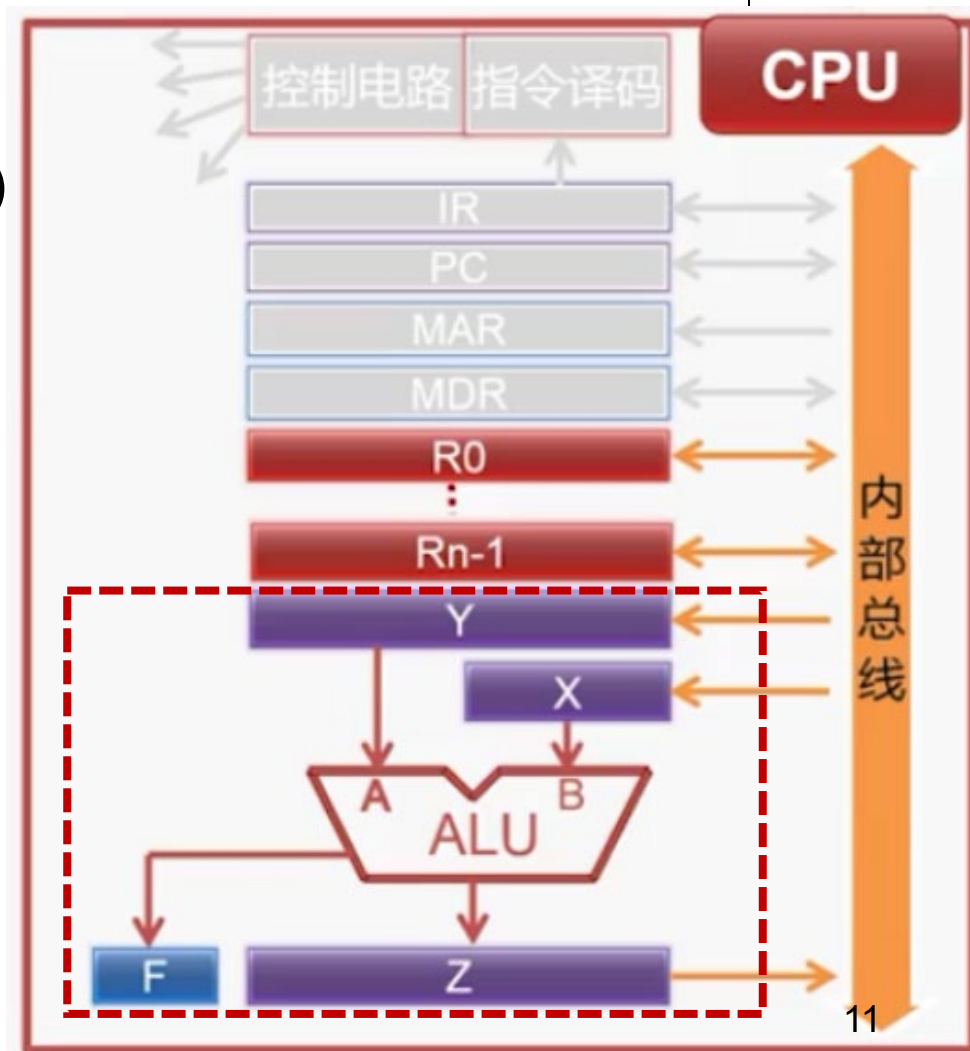
- A: 10, B: 22, C: 10, D: 27

答案：D



# 模型机——运算器 (第二章)

- 运算器核心部件
  - ALU (算数逻辑单元)
- 算数运算
  - 加、减、乘、除
- 逻辑运算
  - 非、与、或
- 数表示方式
  - 定点数
  - 浮点数





# 运算方法与运算器



# 本周教学安排

## ——运算方法和运算器

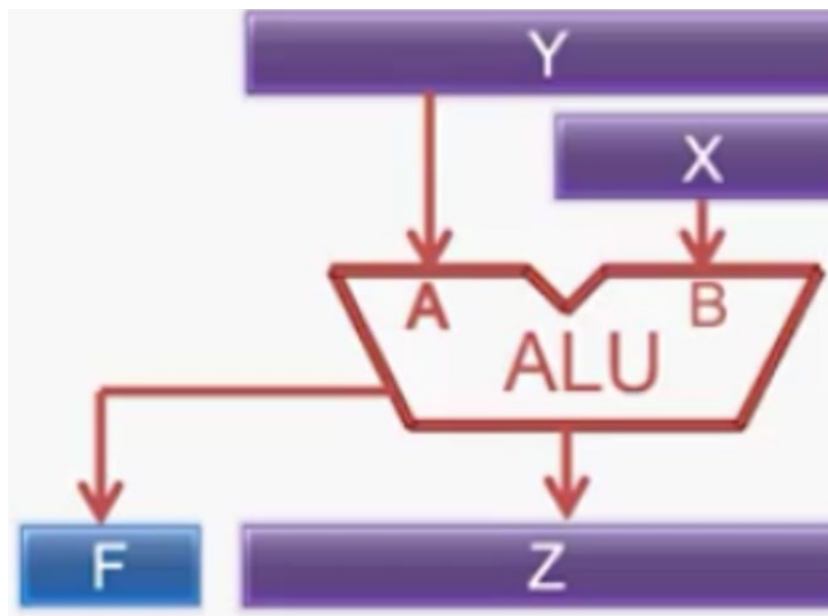


- **硬件部分** (如何设计基本ALU)

- 逻辑运算
  - 非、与、或
- 算数运算
  - 加、减法

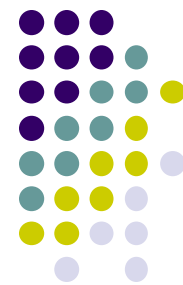
- **理论部分**

- 数的表示方法
  - 定点、浮点数、原、反、补码
  - 字符串与汉字等
- 逻辑、加减运算

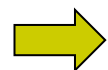


# 第二章 运算方法和运算器

## ——如何设计基本ALU



- 逻辑运算



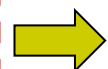
逻辑运算器

- 加法运算

- 行波进位加法器

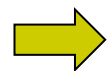
- 超前进位加法器

- 减法运算

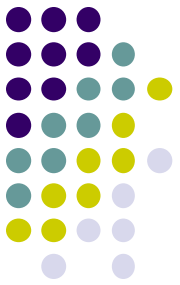


加减法运算器

- 算数逻辑单元实现

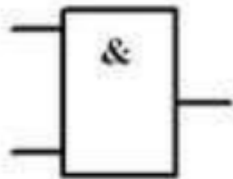
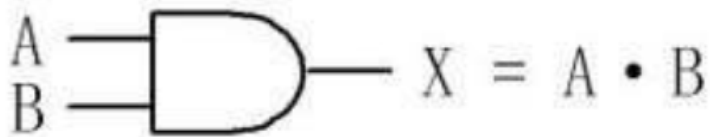


算数+逻辑



# 各类逻辑运算与逻辑门 (1)

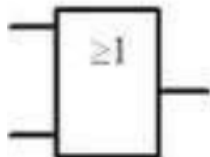
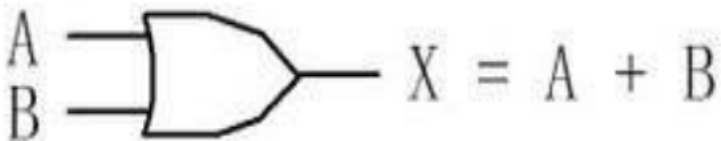
- 与门



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

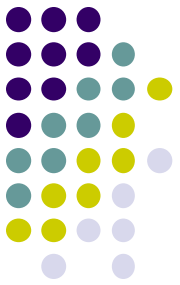
与操作真值表

- 或门



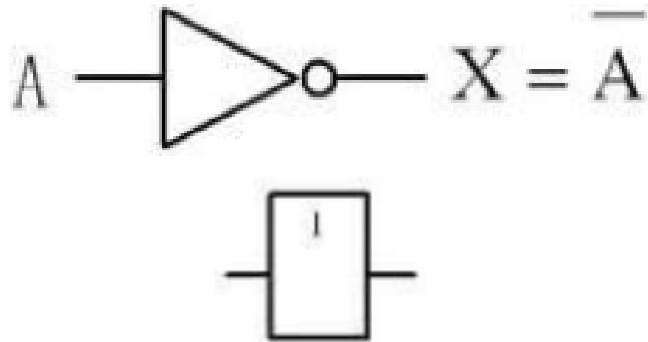
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

或操作真值表



## 各类逻辑运算与逻辑门 (2)

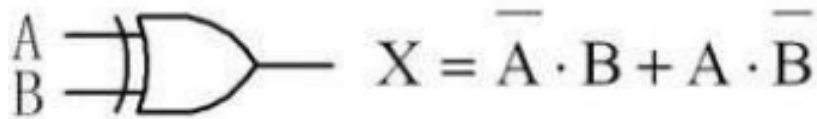
- 非门



A	X
0	1
1	0

非操作真值表

- 异或门

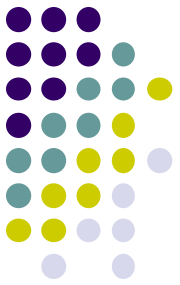


简单异或加密 (simple XOR cipher)  
 $A \text{ xor } B \text{ xor } A = B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

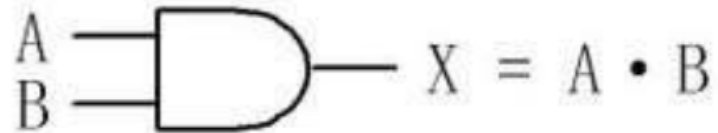
异或操作真值表

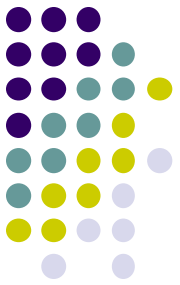




# 运算器中逻辑运算构建

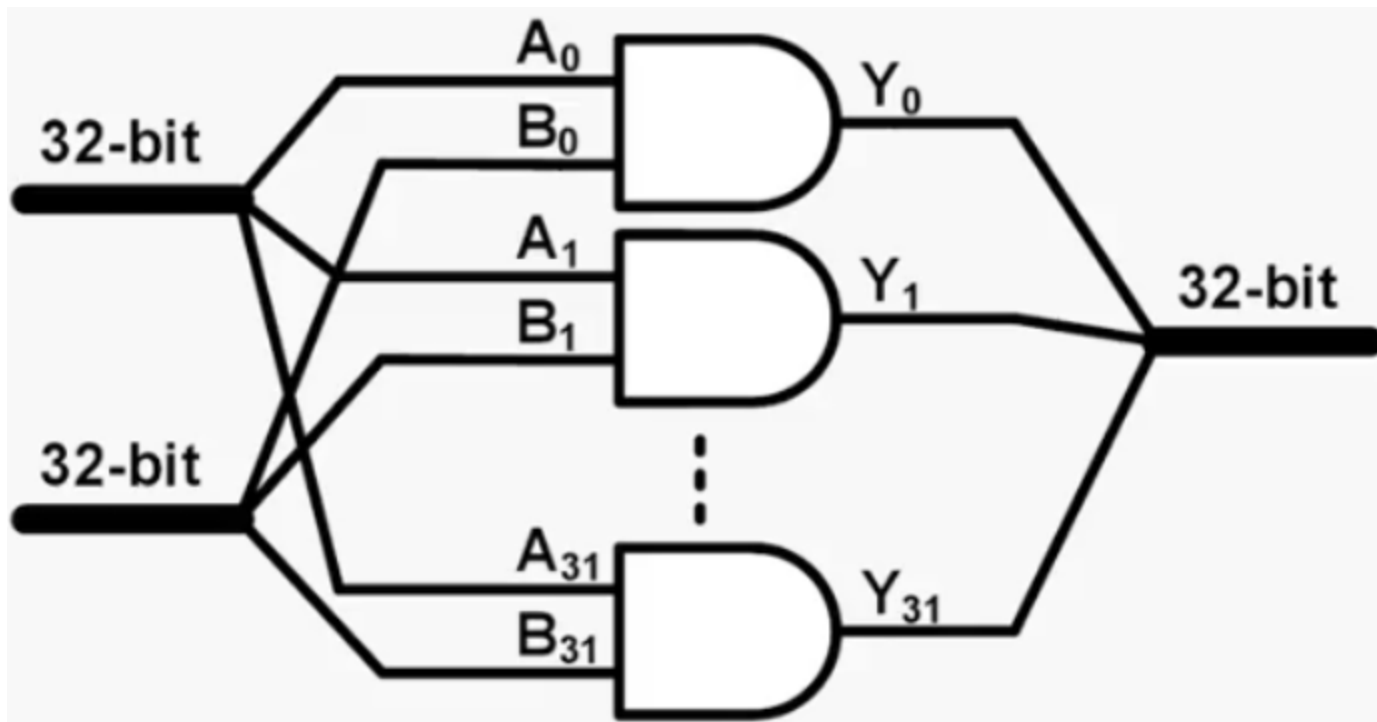
- 逻辑门——1位二进制数，单种逻辑运算
- 运算器逻辑运算指令
  - 与：AND A, B
  - 或：OR A, B
  - 非：NOT A
  - 异或：XOR A, B
  - 上述A/B寄存器为多位（例如，80386为32位）
- 问题
  - 1) 将1位逻辑门扩展为32位逻辑运算电路
  - 2) 如何实现多种逻辑运算

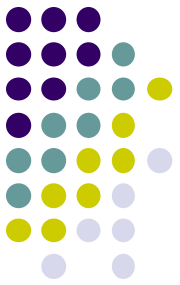




# 问题1：与运算的位扩展

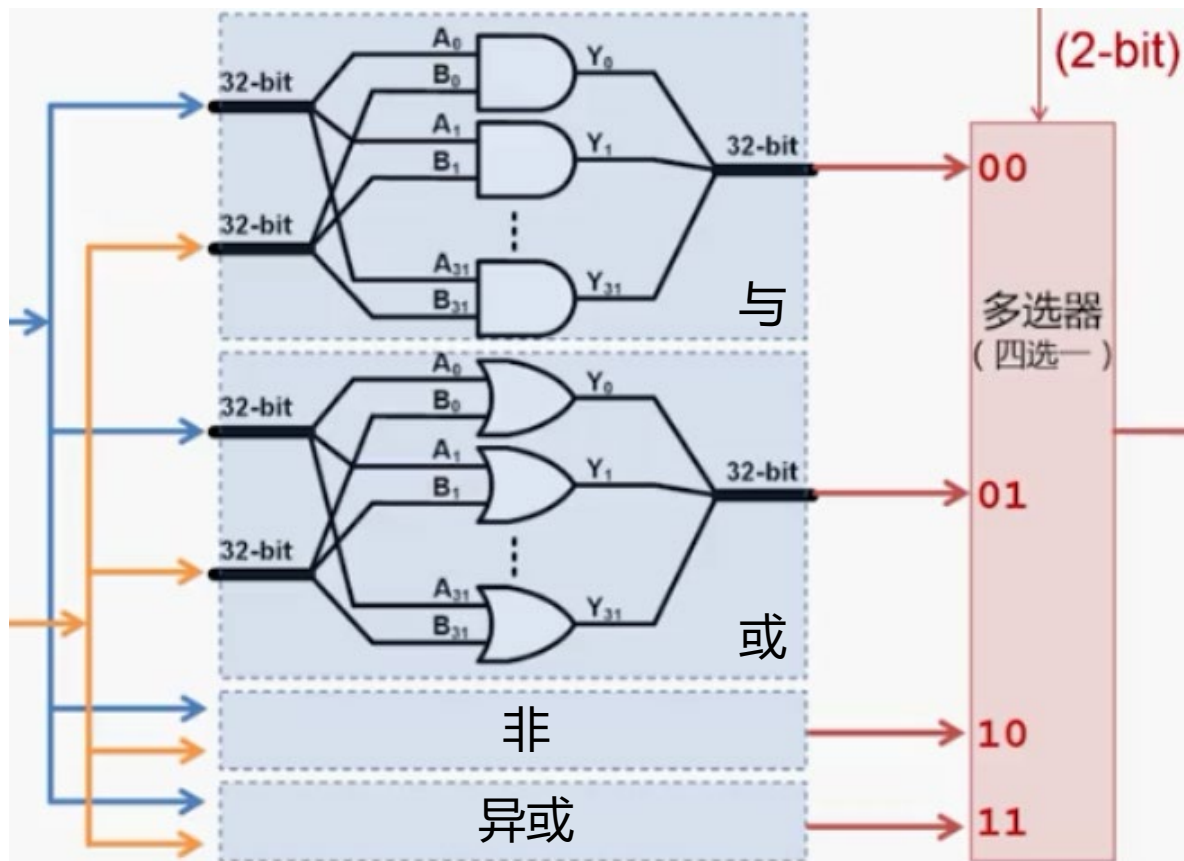
- 逻辑运算特性——按位运算，各位之间独立
- 位扩展方式：32个与门分别连接

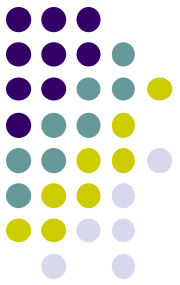




## 问题2：多种逻辑运算

- 逻辑运算单元：包含多种逻辑功能——**如何选择**
- 扩展方式：多种逻辑功能单元+**多**选器+**选**通信号



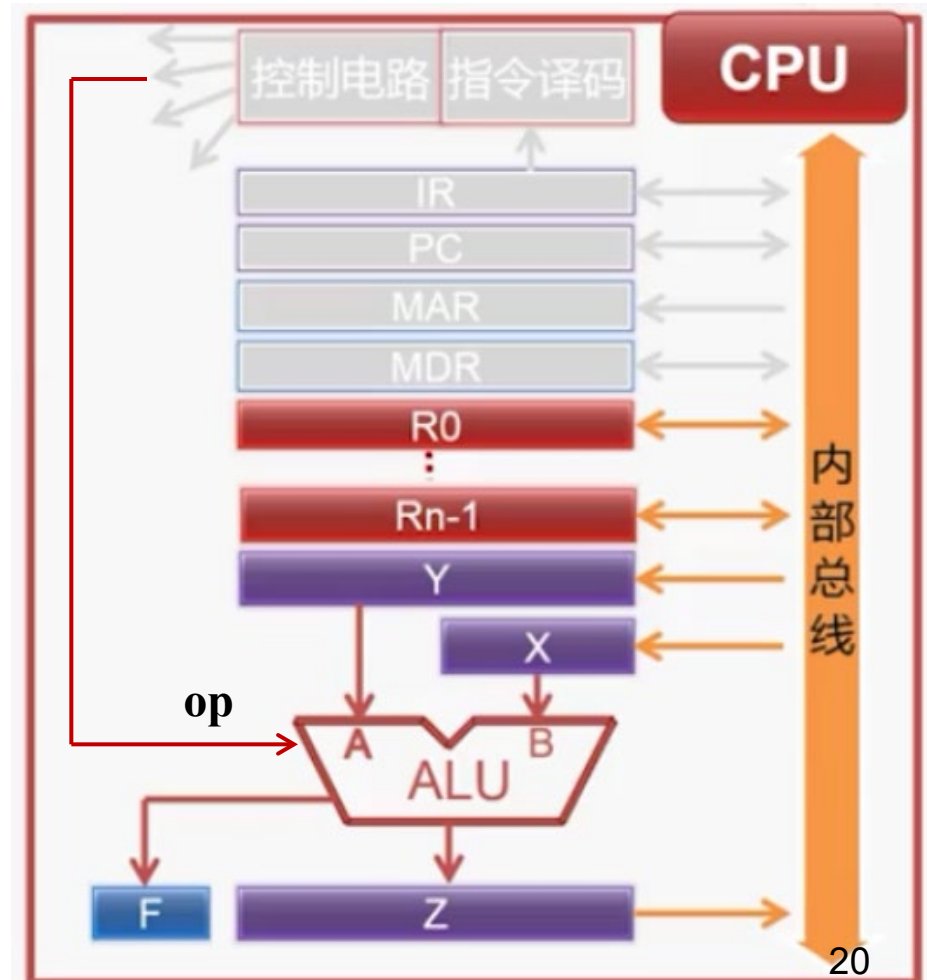


# 逻辑运算单元

- 逻辑运算单元→ALU
- 选通控制信号
  - 两位二进制: op

op1	op2	功能
0	0	与
0	1	或
1	0	非
1	1	异或

- 在ALU新增op信号输入端
  - 控制ALU逻辑运算功能



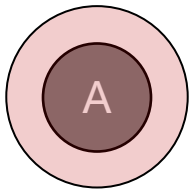
## 单选题 1分



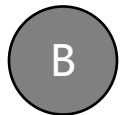
已知  $A=10010010$ ,  $B=00101111$

1) AND A, B

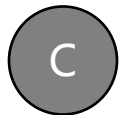
2) XOR A, B



1) 00000010; 2) 10111101



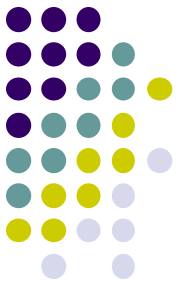
1) 01000010; 2) 10100101



1) 00001110; 2) 10111001



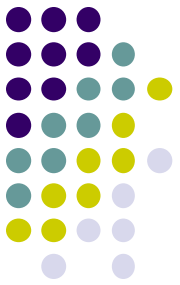
1) 00010010; 2) 10100101



# 第二章 运算方法和运算器

## ——如何设计基本ALU

- 逻辑运算 → 逻辑运算器
- 加法运算
- 行波进位加法器
- 超前进位加法器 → 加减法运算器
- 减法运算
- 算数逻辑单元实现 → 算数+逻辑



# 加减法运算器整体设计思路

- 4位二进制加法—— $A+B$
- 示例： $A=1101$ ,  $B=0101$

$$\begin{array}{r} A \quad \quad 1\ 1\ 0\ 1 \\ +\ B \quad 0\ 1\ 0\ 1 \\ \hline \end{array}$$

$$A+B \quad 1\ 0\ 0\ 1\ 0$$

- 类比逻辑运算设计思路——扩展至加减法
  - 问题1：设计1位二进制加法器（逻辑门组成）
  - 问题2：设计多位加法器（进位问题）
  - 问题3：实现多种算数功能（加/减法+控制信号）

# 第二章 运算方法和运算器

## ——如何设计基本ALU



- 逻辑运算

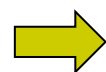
- 加法运算

- 行波进位加法器

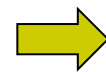
- 超前进位加法器

- 减法运算

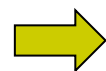
- 算术逻辑单元实现



问题1：逻辑门——1位加法



问题2：两种进位方法



问题3：统一加减法



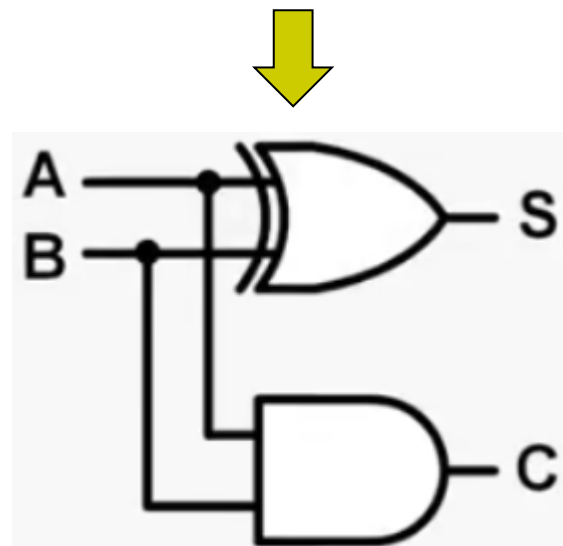
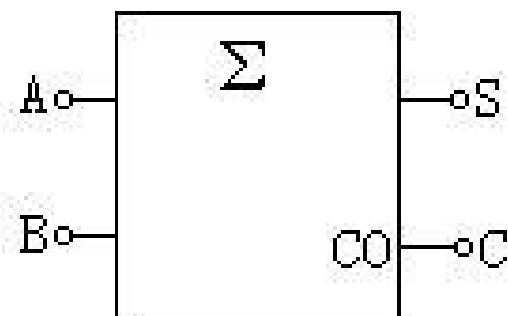
# 半加器

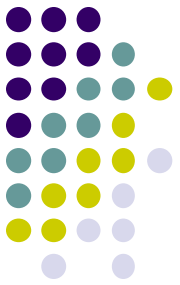
- 半加器 (HA) 功能：两个1位二进制数相加
  - 输入端口：A、B
  - 输出端口：S (和)、C (进位)

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

异或

与





# 全加器

- 全加器 (FA)
- 功能：三个1位二进制数相加
  - 输入：A、B、 $C_{in}$  (进位输入)
  - 输出：S、 $C_{out}$  (进位输出)

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

逻辑函数

A	B	$C_{in}$	S	$C_{out}$
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

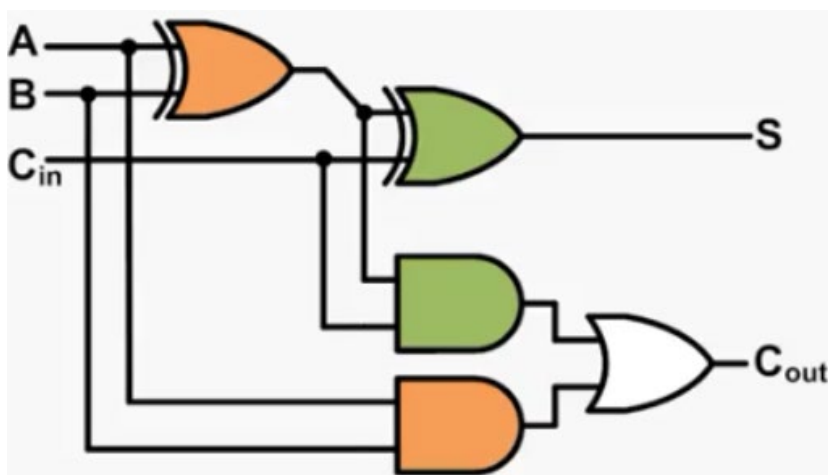
# 全加器实现



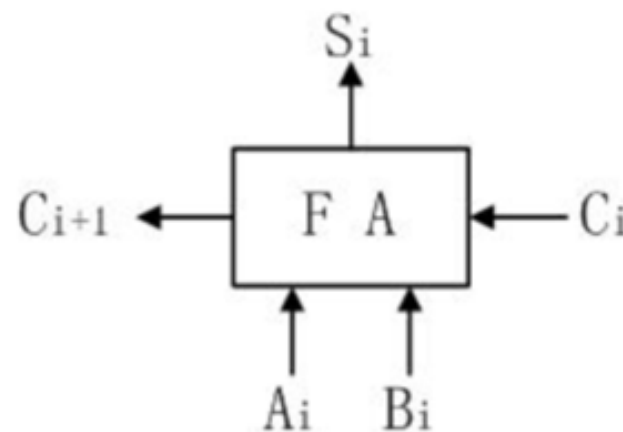
$$\begin{aligned} S_i &= A_i \oplus B_i \oplus C_i \\ C_{i+1} &= A_i B_i + A_i C_i + B_i C_i \\ &= A_i B_i + (A_i \oplus B_i) C_i \end{aligned}$$

(a) 逻辑函数

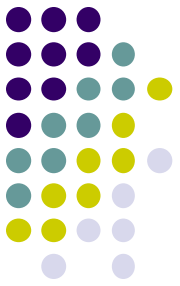
$$\begin{aligned} C_{i+1} &= A_i \cdot B_i + A_i \cdot C_i + B_i \cdot C_i \\ &= A_i \cdot B_i + (A_i + B_i) C_i \end{aligned}$$



(b) 全加器门电路

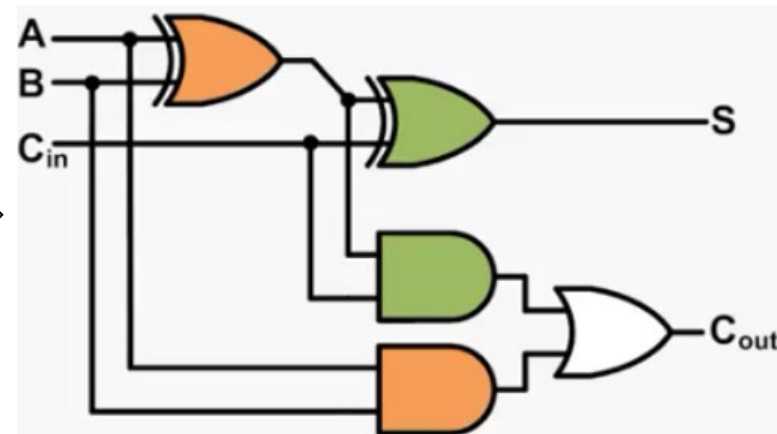
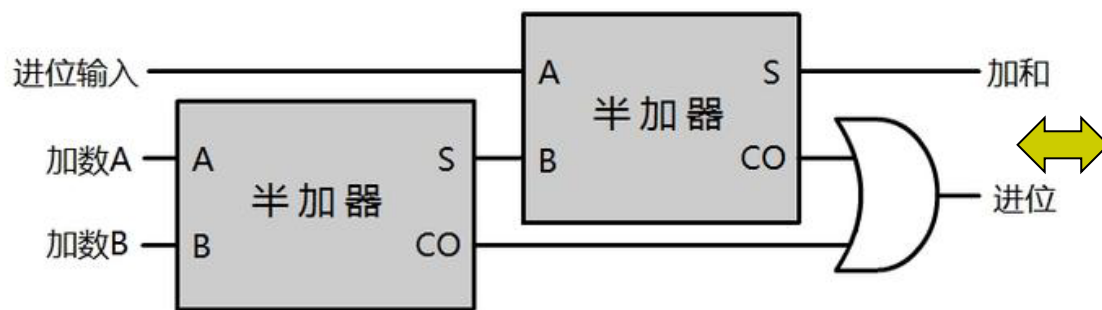


(c) 全加器框图



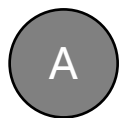
# 全加器与半加器关联

- 半加器 (HA)
  - 功能: 两个1位二进制数相加 ( $A$ 、 $B$ )
- 全加器 (HA)
  - 功能: 三个1位二进制数相加 ( $A$ 、 $B$ 、 $C_{in}$ )
- 全加器可通过两个半加器级联得到



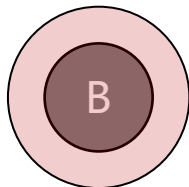


全加器逻辑函数可表示为：



$$S = AB + BC_{in}$$

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

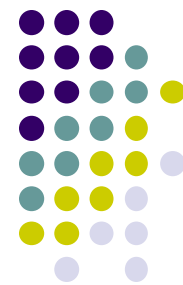


$$S = A \oplus B \oplus C_{in}$$

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

# 第二章 运算方法和运算器

## ——如何设计基本ALU



- 逻辑运算

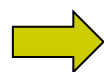
- 加法运算

- 行波进位加法器

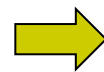
- 超前进位加法器

- 减法运算

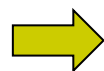
- 算数逻辑单元实现



问题1：逻辑门—1位加法

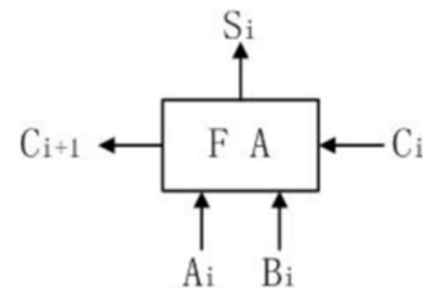


问题2：两种进位方法

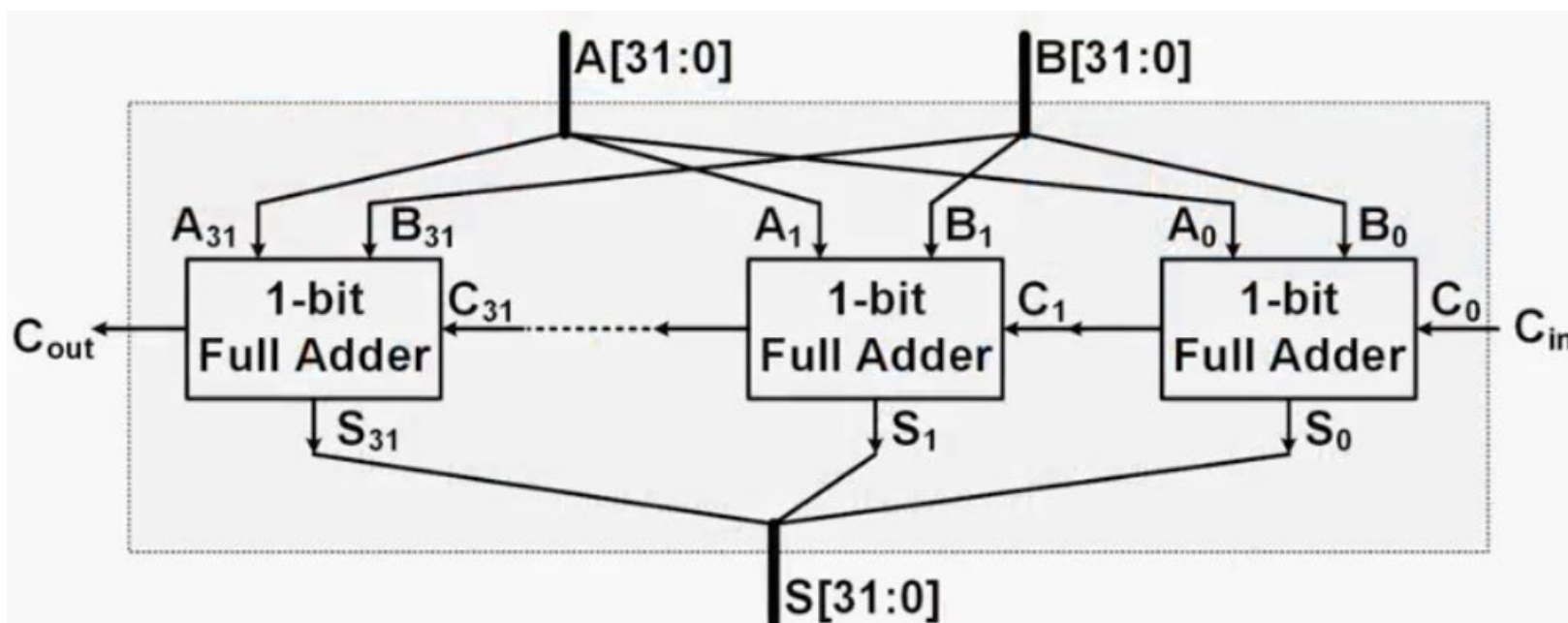


问题3：统一加减法

## 问题2：设计多位加法器



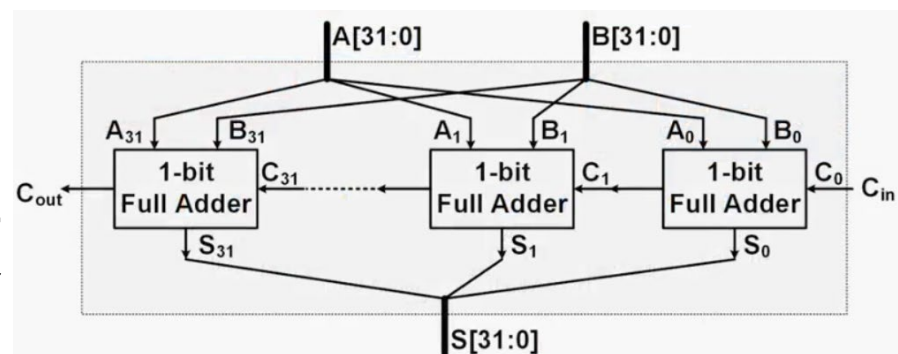
- 问题：加法前后依赖，如何利用全加器实现多位加法
- 扩展方式：全加器级联扩展
- 将前一个全加器的 $C_{out}$ 连接到后一个全加器 $C_{in}$



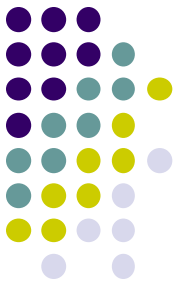


# 行波进位加法器分析

- 结构特点
  - 级联：前一个全加器进位输出→后一个全加器进位输入
- 优点
  - 电路布局简单，设计方便
- 缺点
  - 后一级全加器输入需等待前一级全加器输出
  - 延迟时间长，且随加法位数递增
- 问题：如何分析行波进位加法器时延？

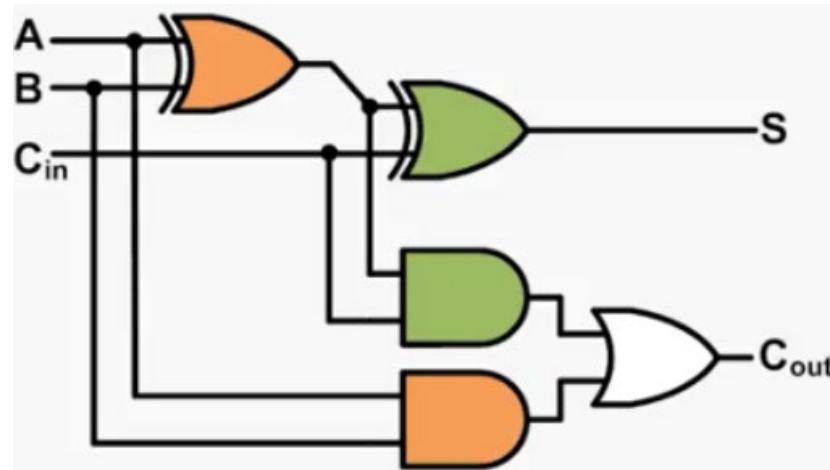


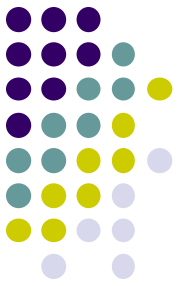




# 问题：延迟分析——单级全加器

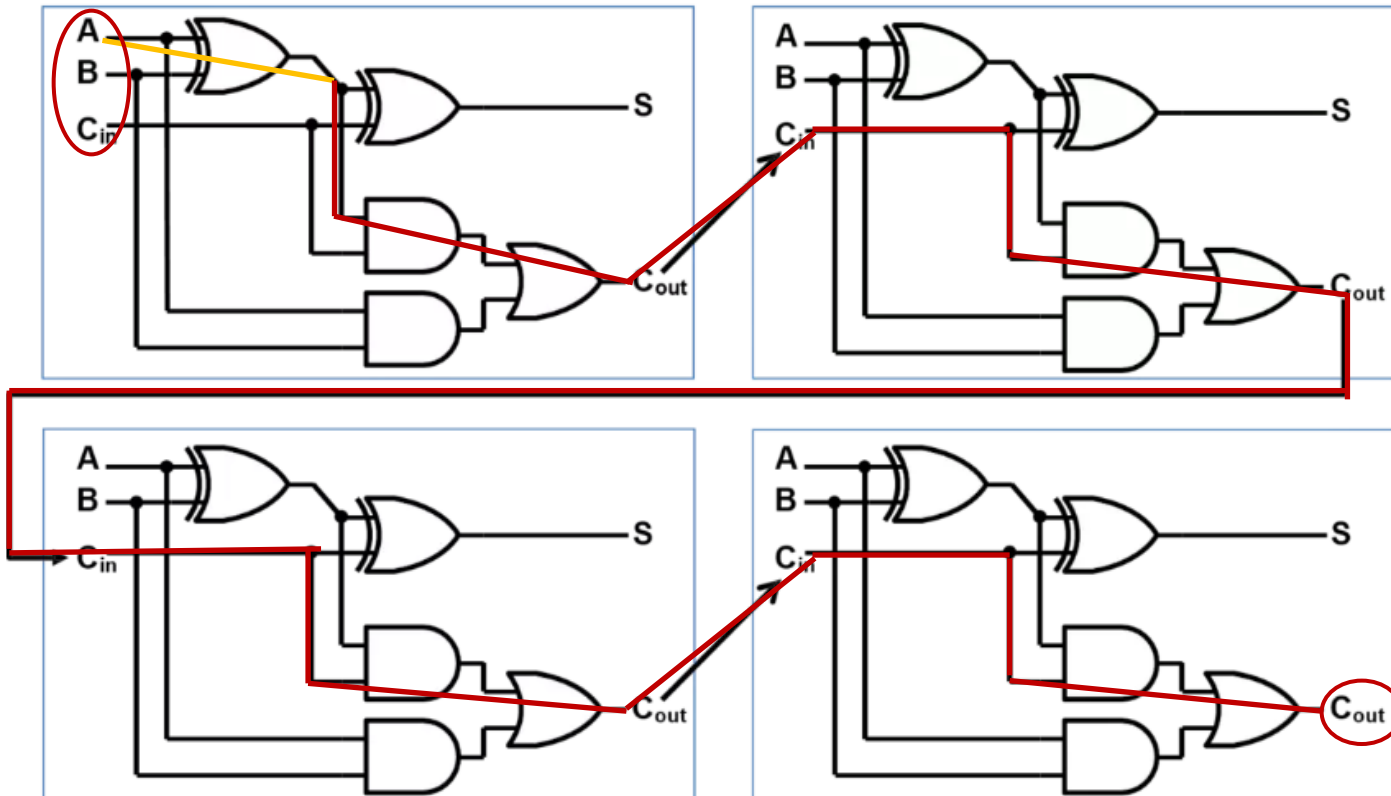
- 基本门电路延迟
  - 与门/或门延迟： $T$
  - 异或门延迟： $3T$
- 每级全加器
  - 输入A、B → 输出S： $6T$
  - 输入 $C_{in}$  → 输出 $C_{out}$ ： $2T$

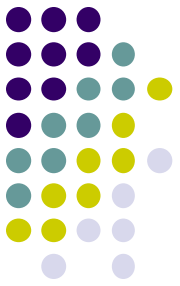




# 问题：延迟分析——关键路径

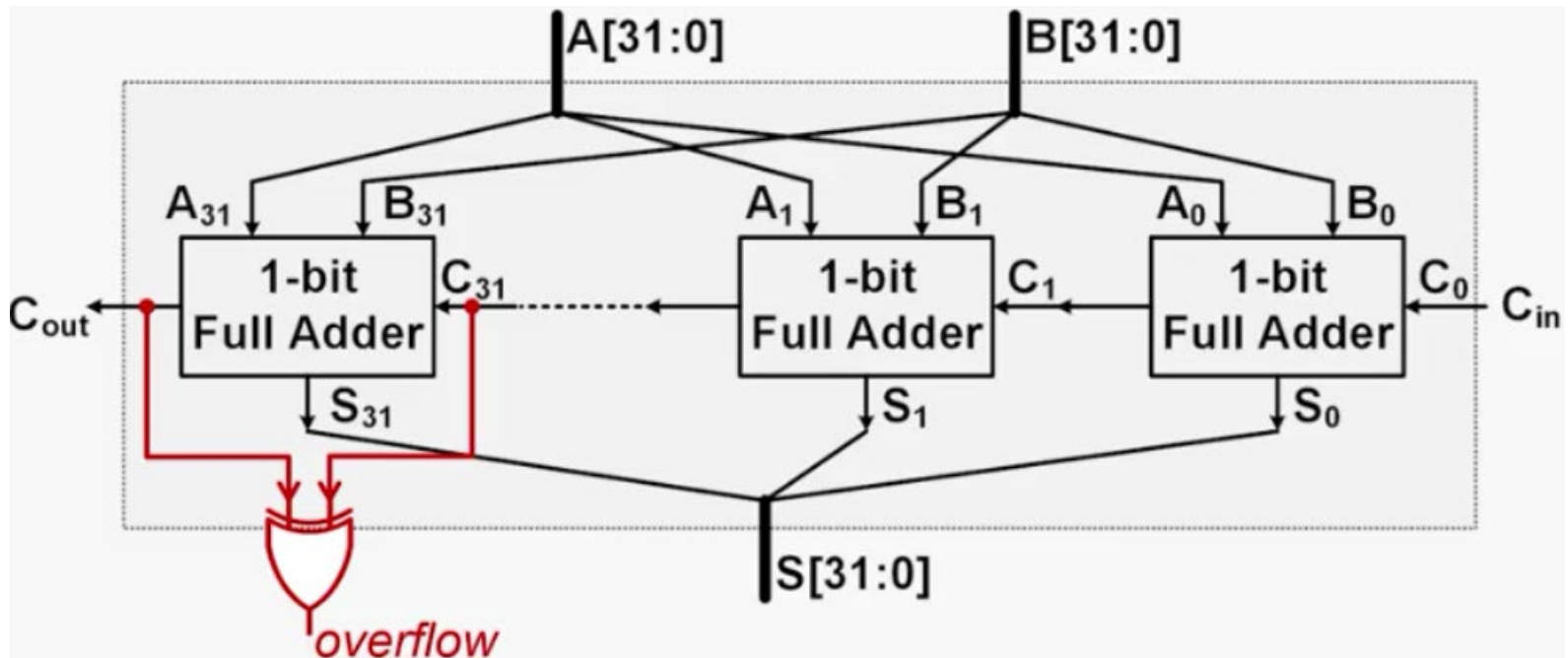
- n位Cout延迟：  $t_a = 3T + 2T * n$ 
  - 第一级异或门：  $3T$
  - 每级进位输入输出：  $2T$





# 行波进位加法器——溢出判断

- 溢出判断：最高位进位输入不等于其进位输出
  - 有符号数 v.s. 无符号数
  - 溢出判断分析（后续内容）
  - 溢出判断延迟：  $t_a = 3T + 2T * n + 3T$



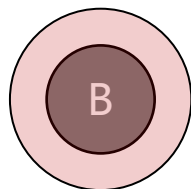


N位行波进位加法器中，最高位进位输出Cout与溢出判断延迟分别为？



$$2T * N + 3T$$

$$2T * N + 4T$$



$$2T * N + 3T$$

$$2T * N + 6T$$

# 第二章 运算方法和运算器

## ——如何设计基本ALU



- 逻辑运算

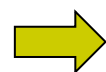
- 加法运算

- 行波进位加法器

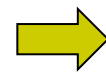
- 超前进位加法器

- 减法运算

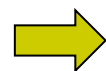
- 算数逻辑单元实现



问题1：逻辑门—1位加法

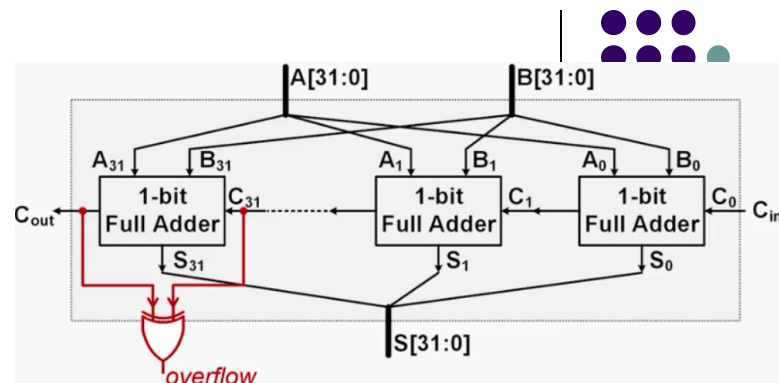


问题2：两种进位方法

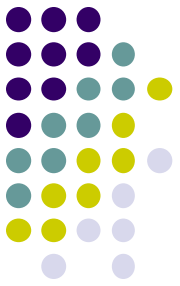


问题3：统一加减法

# 行波进位加法器优化



- 主要问题
  - 高位运算需等待低位运算结果
- 如何避免行波进位加法器延迟？
- 优化方法
  - 提前计算进位信号
  - 超前（先行）进位加法器
- 超前进位信号发生器设计
  - 分析进位逻辑表达式
  - 表达式化简→计算超前进位表达式

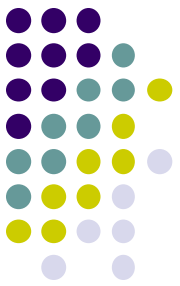


# 进位信号分析

$$\begin{aligned}C_{i+1} &= A_i \cdot B_i + A_i \cdot C_i + B_i \cdot C_i \\&= A_i \cdot B_i + (A_i + B_i) C_i\end{aligned}$$

- 超前进位
  - 避免 $C_i$ 与 $C_{i+1}$ 间依赖
- 令
  - 生成信号:  $G_i = A_i \cdot B_i$
  - 传播信号:  $P_i = A_i + B_i$
- 简化进位信号为

$$C_{i+1} = G_i + P_i \cdot C_i$$



# 进位信号计算

$$C_1 = G_0 + P_0 \cdot C_0$$

$$C_2 = G_1 + P_1 \cdot C_1$$

$$= G_1 + P_1 \cdot (G_0 + P_0 \cdot C_0)$$

$$= \boxed{G_1} + \boxed{P_1} \cdot \boxed{G_0} + \boxed{P_1} \cdot \boxed{P_0} \cdot \boxed{C_0}$$

$$C_3 = G_2 + P_2 \cdot C_2$$

$$= G_2 + P_2 \cdot (G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0)$$

$$= \boxed{G_2} + \boxed{P_2} \cdot \boxed{G_1} + \boxed{P_2} \cdot \boxed{P_1} \cdot \boxed{G_0} + \boxed{P_2} \cdot \boxed{P_1} \cdot \boxed{P_0} \cdot \boxed{C_0}$$

$$C_4 = G_3 + P_3 \cdot C_3$$

$$= G_3 + P_3 \cdot (G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0)$$

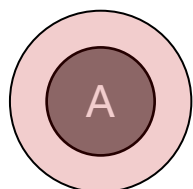
$$= \boxed{G_3} + \boxed{P_3} \cdot \boxed{G_2} + \boxed{P_3} \cdot \boxed{P_2} \cdot \boxed{G_1} + \boxed{P_3} \cdot \boxed{P_2} \cdot \boxed{P_1} \cdot \boxed{G_0} + \boxed{P_3} \cdot \boxed{P_2} \cdot \boxed{P_1} \cdot \boxed{P_0} \cdot \boxed{C_0}$$

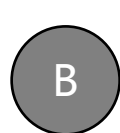
$$\boxed{C_{i+1} = G_i + P_i \cdot C_i}$$

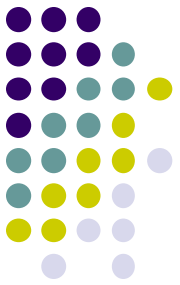




超前进位信号发生器中，进位信号 $C_5$ 表达式为

 
$$G_4 + P_4 \cdot G_3 + P_4 \cdot P_3 \cdot G_2 + P_4 \cdot P_3 \cdot P_2 \cdot G_1 + P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

 
$$G_4 + P_4 \cdot G_3 + P_4 \cdot P_3 \cdot G_2 + P_4 \cdot P_3 \cdot P_2 \cdot G_1 + P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot G_0$$



# 进位信号分析 (1)

- 延迟分析:  $3T$  (与级数无关)

- 生成信号/传播信号延迟:  $T$  (与/或门)

- 进位信号:  $2T$  (与门+或门)

$$G_i = A_i \cdot B_i$$

$$P_i = A_i + B_i$$

$$C_1 = G_0 + P_0 \cdot C_0$$

$$C_2 = G_1 + P_1 \cdot C_1$$

$$= G_1 + P_1 \cdot (G_0 + P_0 \cdot C_0)$$

$$= G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0$$

$$C_{i+1} = G_i + P_i \cdot C_i$$

$$C_3 = G_2 + P_2 \cdot C_2$$

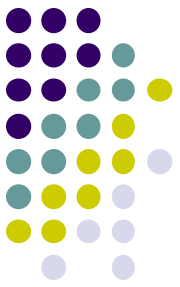
$$= G_2 + P_2 \cdot (G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0)$$

$$= G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

$$C_4 = G_3 + P_3 \cdot C_3$$

$$= G_3 + P_3 \cdot (G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0)$$

$$= G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0$$



## 进位信号分析 (2)

$$\underline{C_1 = G_0 + P_0 \cdot C_0}$$

2项

$$C_2 = G_1 + P_1 \cdot C_1$$

$$= G_1 + P_1 \cdot (G_0 + P_0 \cdot C_0)$$

$$\underline{= G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0}$$

3项

$$C_3 = G_2 + P_2 \cdot C_2$$

$$= G_2 + P_2 \cdot (G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0)$$

$$\underline{= G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0}$$

4项

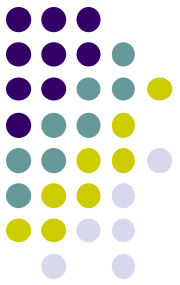
$$C_4 = G_3 + P_3 \cdot C_3$$

$$= G_3 + P_3 \cdot (G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0)$$

$$\underline{= G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0}$$

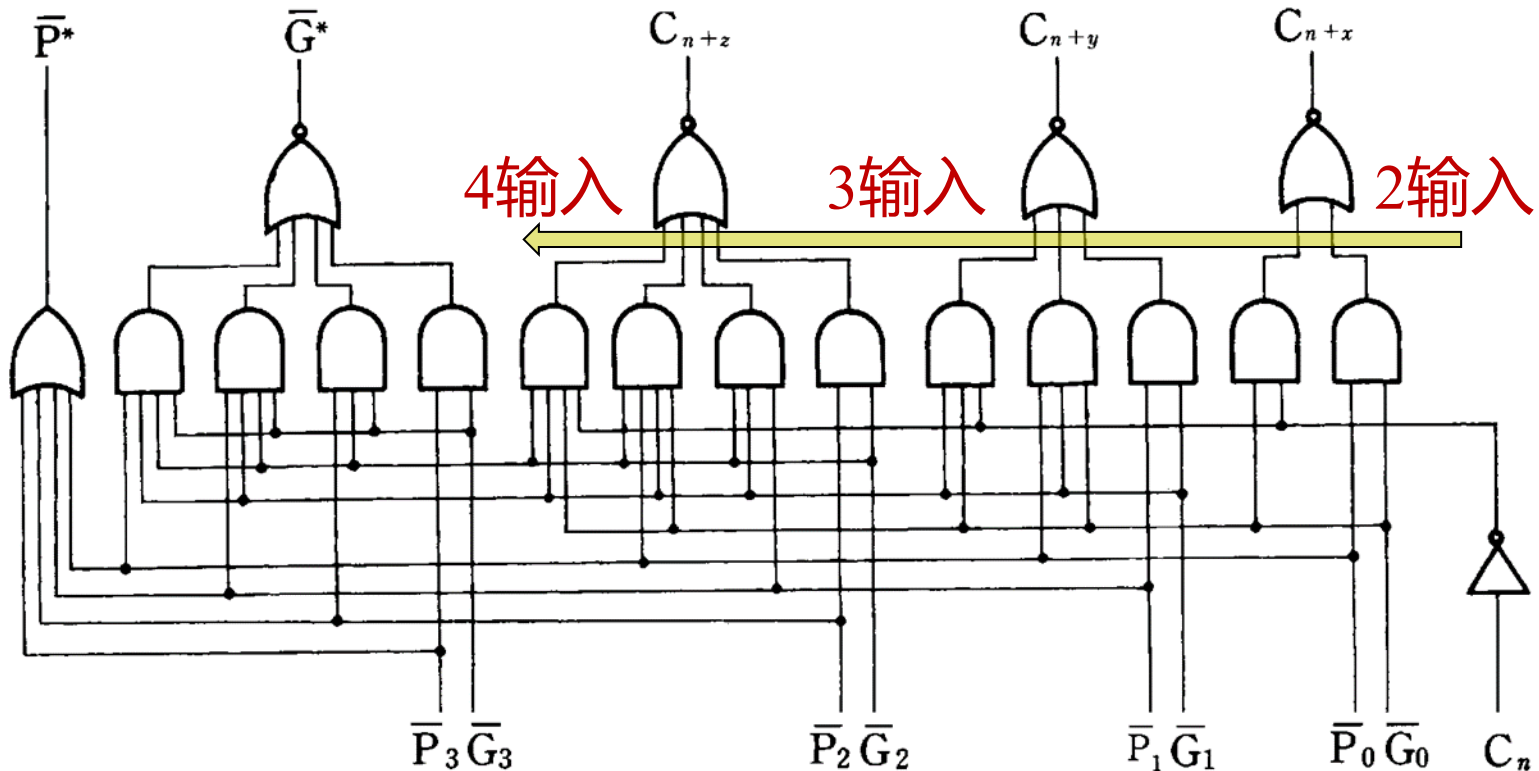
5项

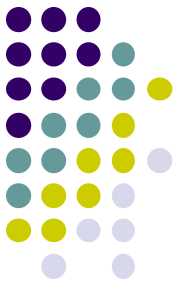
$$\boxed{C_{i+1} = G_i + P_i \cdot C_i}$$



# 超前进位信号发生器 (74182)

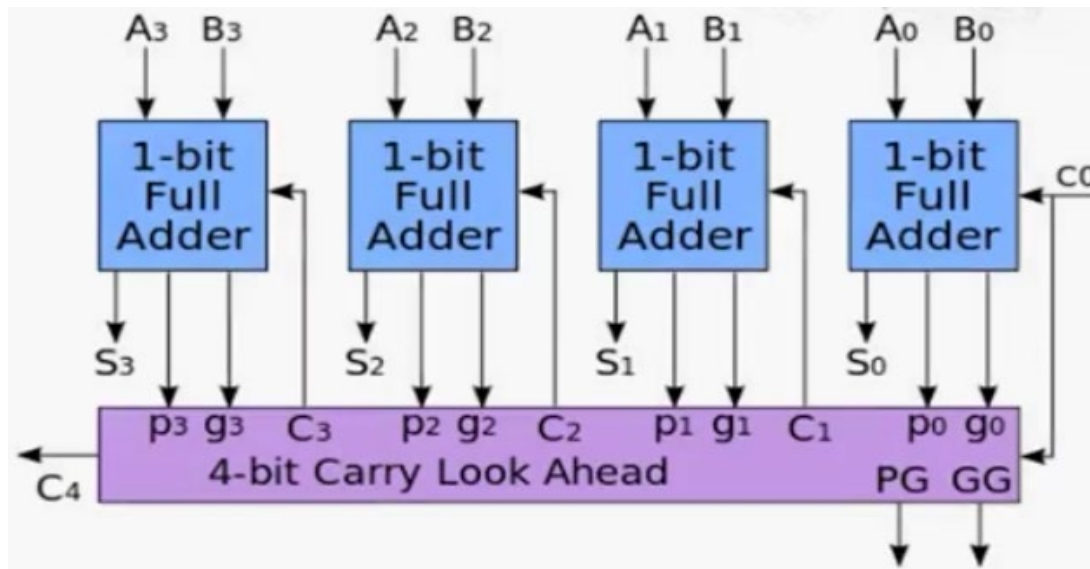
- 超前进位信号发生器实现 (74182) :
  - 对照进位信号分析结果
- 最高位保留进位与生成信号方便级联

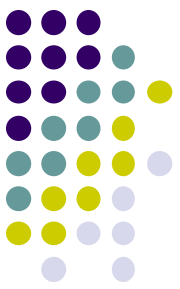




# 超前进位加法器实现

- 超前进位加法器
  - Carry-Lookahead Adder, CLA
- 实现方法
  - 并联加法器
  - 超前进位信号发生器





# 超前进位加法器分析

- 结构特点
  - 并联方法：统一进位信号发生器
- 优点：
  - 后一级输入不依赖前一级输出，电路延迟低
- 缺点：
  - 进位信号发生器实现复杂，并随位数增加，难以实现
- 实际加法器实现
  - 混合方式：行波进位（片间）+超前进位（片内）
  - 32位加法器：4个8位超前进位加法器

# 第二章 运算方法和运算器

## ——如何设计基本ALU



- 逻辑运算

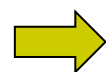
- 加法运算

- 行波进位加法器

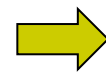
- 超前进位加法器

- 减法运算

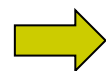
- 算数逻辑单元实现



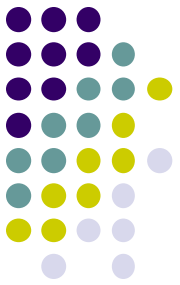
问题1：逻辑门—1位加法



问题2：两种进位方法



问题3：统一加减法



# 有符号数存储方式——补码

- 减法运算可转换为加法运算

$$A - B = A + (-B)$$

- 转换规则：

- 负号：**按位取反，末位加一**

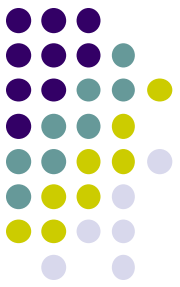
- [-1] 的原/反/补码

$$= [10000001]_{\text{原}} = [11111110]_{\text{反}} = [11111111]_{\text{补}}$$

- 加减法转换

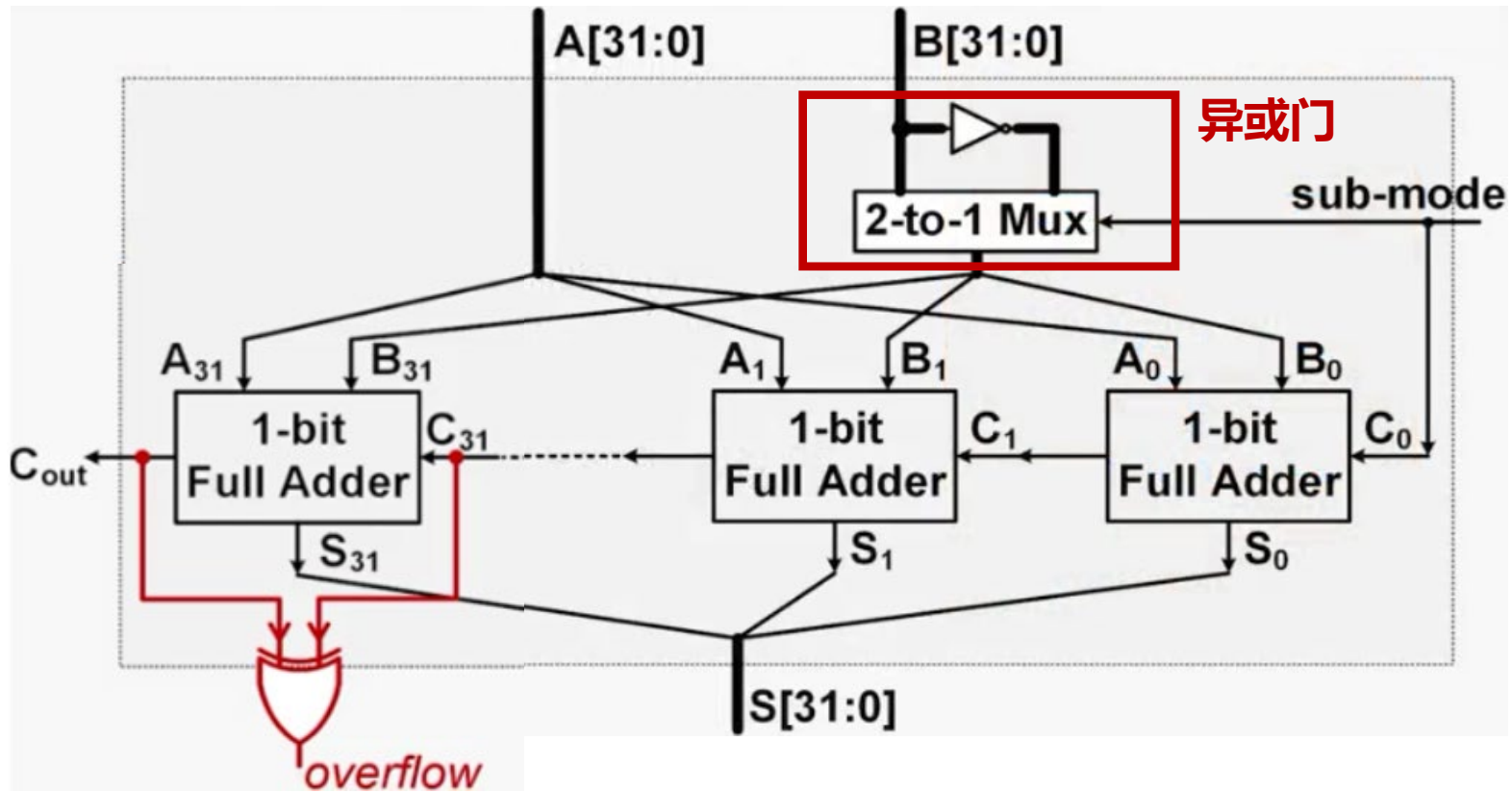
$$[A - B]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}}$$





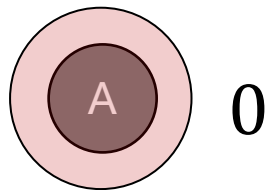
# 加减法运算实现

- 实现思路：加法器基础上进行修正
  - B输入端加入多选器
  - sub-mode 选择信号：0 / 1 **按位取反，末位加一**

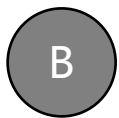




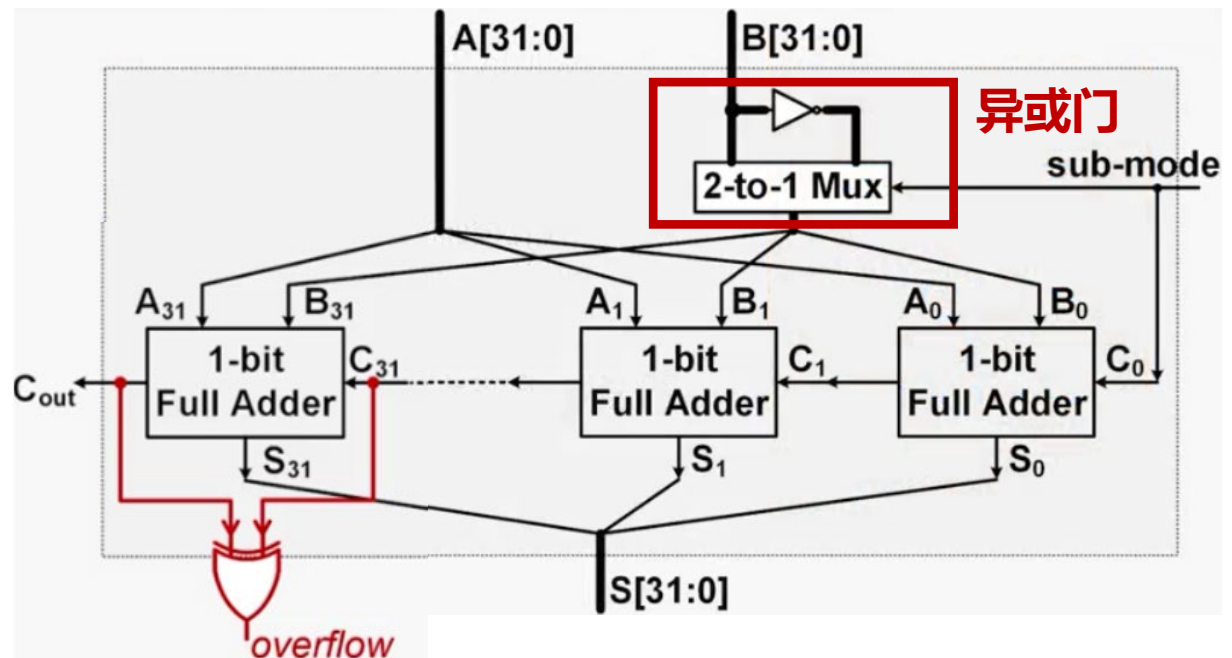
下图所示电路中，若要实现加法运算，此时，sub-mode信号应为



0



1



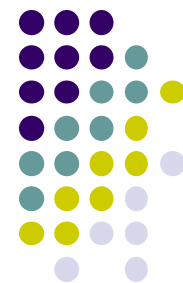


# 第二章 运算方法和运算器

## ——如何设计基本ALU

- 逻辑运算 → 逻辑运算器
- 加法运算
- 行波进位加法器
- 超前进位加法器 → 加减法运算器
- 减法运算
- 算数逻辑单元实现 → 算数+逻辑

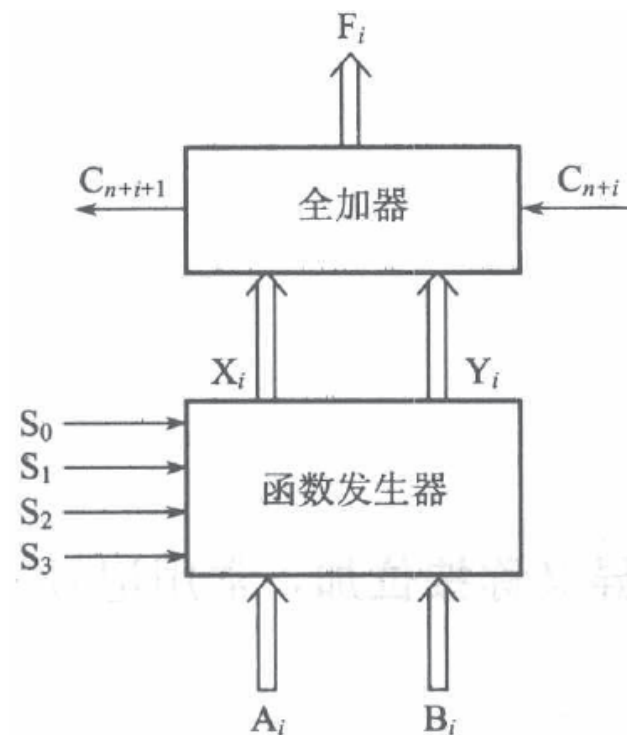
# 多功能算数逻辑单元 (ALU)



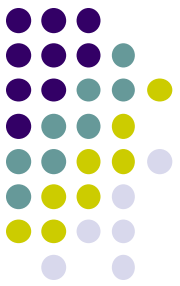
- 目标：
  - 实现多功能算数逻辑运算
  - 算数：加、减
  - 逻辑运算：与、或、非、异或等
- 方法：
  - 新增控制输入端：S0~S4
  - 配合函数发生器，实现不同功能
- 全加器 (FA) 输入为函数发生器输出

$$F_i = X_i \oplus Y_i \oplus C_{n+1}$$

$$C_{n+i+1} = X_i Y_i + Y_i C_{n+1} + C_{n+1} X_i$$



一位ALU逻辑图



# 函数发生器功能

- 输入输出对应关系

- S0~S1控制Yi

- S2~S3控制Xi

$$\xrightarrow{\quad} (A_i/B_i \xrightarrow[S_0 \sim S_3]{\quad} X_i/Y_i)$$

- 函数关系表

S <sub>0</sub> S <sub>1</sub>	Y <sub>i</sub>	S <sub>2</sub> S <sub>3</sub>	X <sub>i</sub>
0 0	$\overline{A_i}$	0 0	1
0 1	$\overline{A_i}B_i$	0 1	$\overline{A_i} + \overline{B_i}$
1 0	$\overline{A_i}\overline{B_i}$	1 0	$\overline{A_i} + B_i$
1 1	0	1 1	$\overline{A_i}$

- 函数表达式

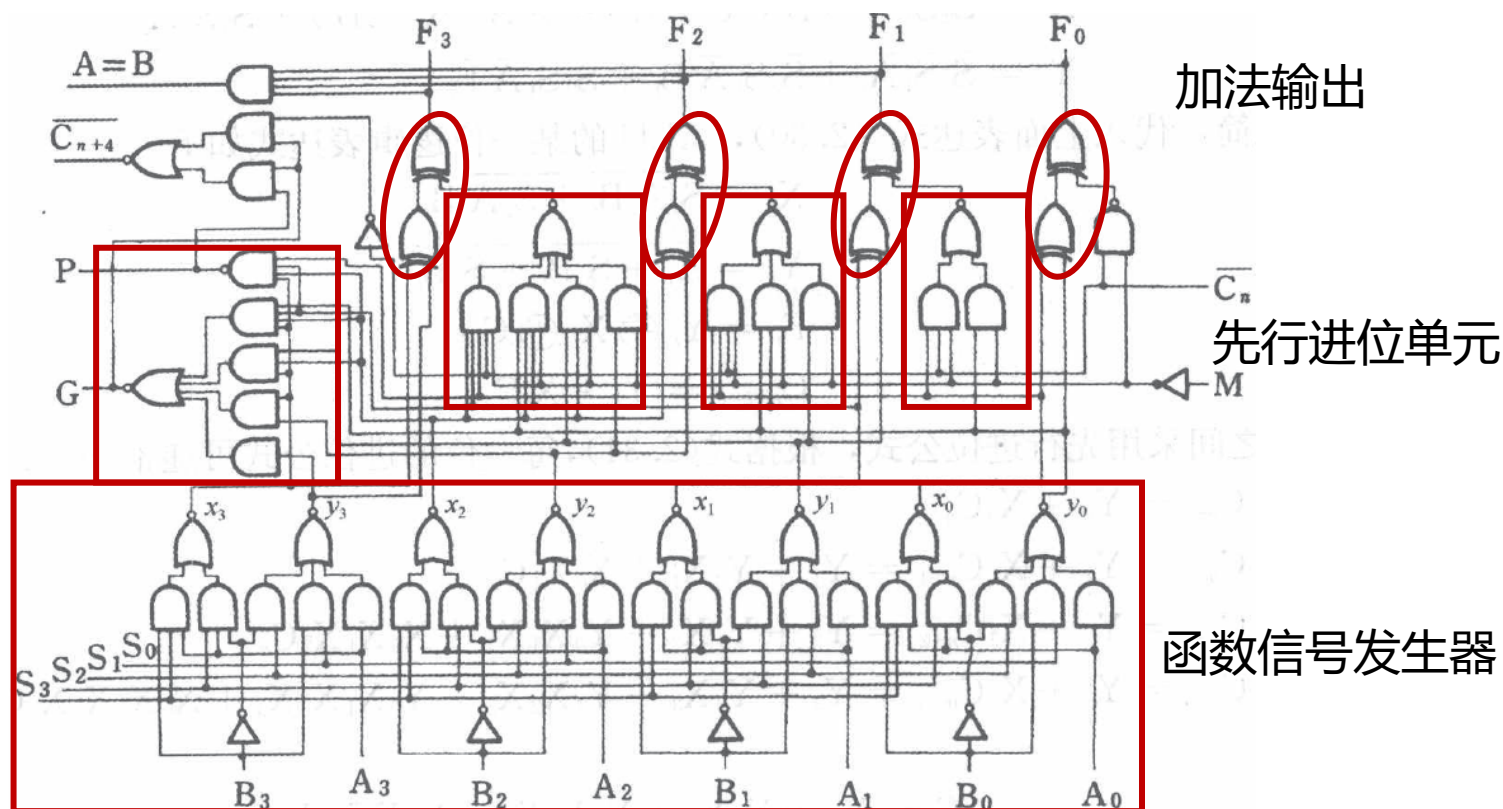
$$X_i = \overline{S_3 A_i B_i + S_2 A_i \overline{B_i}}$$

$$Y_i = \overline{A_i + S_0 B_i + S_1 \overline{B_i}}$$

# 多功能算数逻辑单元 (ALU)



- 4位ALU芯片74181逻辑电路图 (正逻辑)

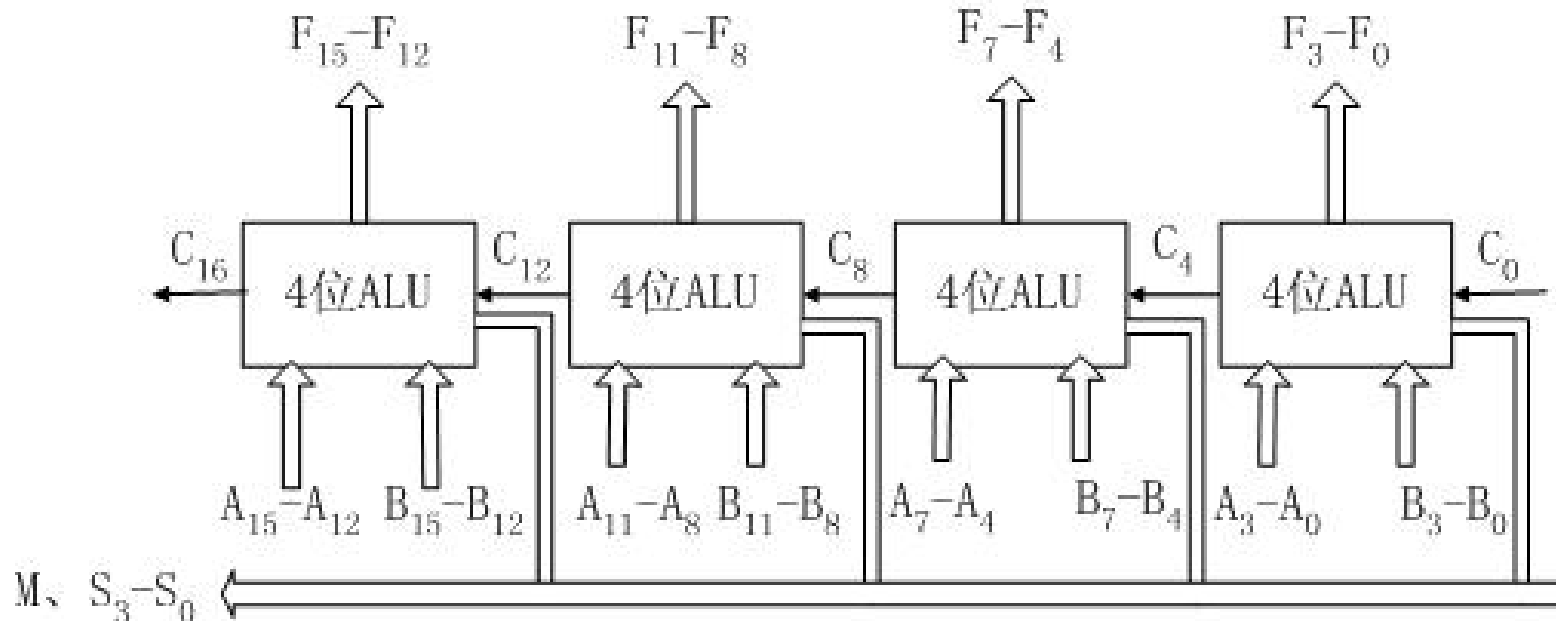


- $M=L/0$ 时, 进位信号无影响, 算数运算
- $M=H/1$ 时, 进位门被封锁, 无进位, 逻辑运算



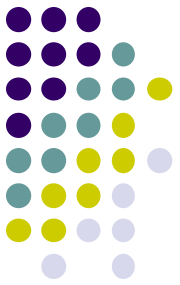
# 多功能算数逻辑单元的级联

- 设计16位ALU
  - 4片74181 (4位ALU) 级联



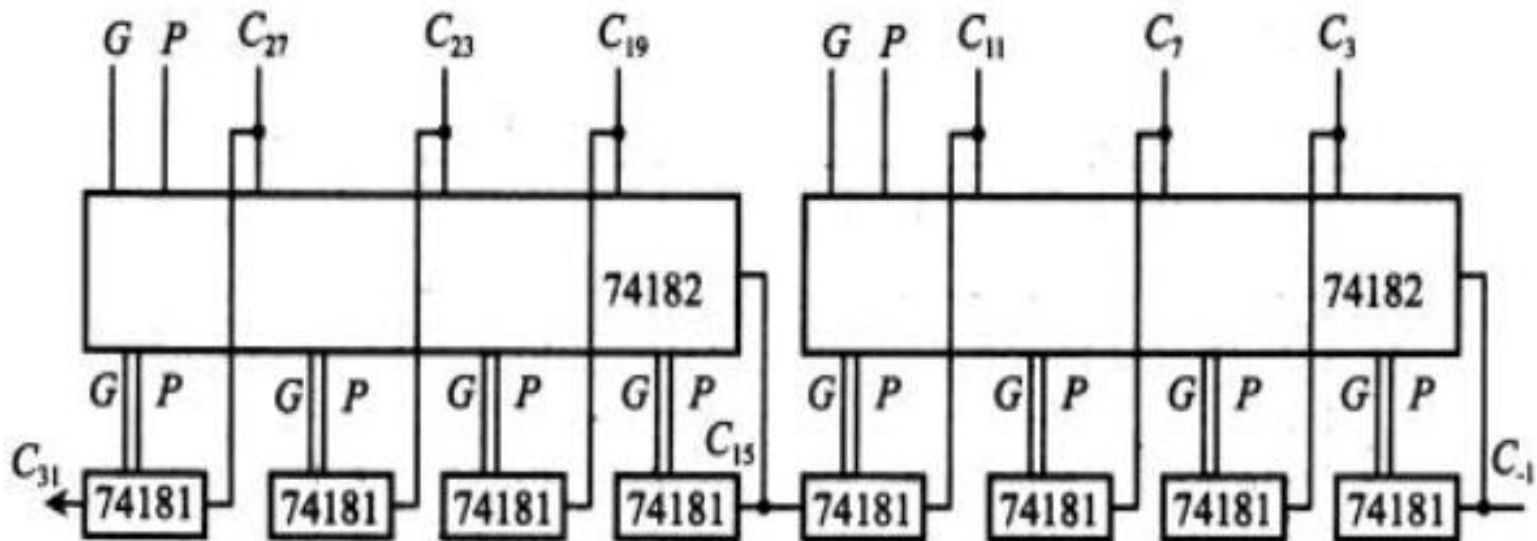
- 片内超前（先行）进位，片间行波进位





# 多功能算数逻辑单元的级联

- 设计32位ALU
  - 8片74181 (4位ALU)
  - 2片超前进位信号发生器74182



- 16位内超前进位, 16位间行波进位



# 总结



## 运算器 (定点)

### 逻辑运算器

- 门电路: 与、或、非、异或
- 两类扩展: 位扩展、功能合并

### 加减运算器

#### 加减运算器设计

1位运算器: HA、FA (逻辑表达式)

位扩展 (前后依赖)

行波 (串行) 进位

逻辑框图

时延分析: 关键路径

超前 (并行) 进位

超前信号分析: 规律、基础公式

时延分析:  $3T$

74182超前进位信号发生器

溢出判断 (符号补码)

最高FA的Cin与Cout异或

功能合并: 加减法统一

补码 (按位取反、末位加一)

异或门电路

### ALU (基础)

思路: 函数信号发生器+算数逻辑单元 (节约硬件)

基本框图与74181算数逻辑芯片对应

扩展方式

片内超前、片间行波

片内超前、片间超前