

# 数字逻辑

## 补码加减法运算

2019年3月18日

# 补码加减法运算

## 1. 原码加/减法运算

### 加法规则：

先判符号位，若相同，绝对值相加，结果符号不变； 若不同，则作减法，  $|大| - |小|$ ，结果符号与 $|大|$ 相同。

### 减法规则：

两个原码表示的数相减，首先将减数符号取反，然后将被减数与符号取反后的减数按原码加法进行运算。

## 2. 补码加法运算

补码加法的公式：

$$[x]_{\text{补}} + [y]_{\text{补}} = [x+y]_{\text{补}} \quad (\text{mod } 2)$$

**特点：**不需要事先判断符号，符号位与码值位一起参加运算。

符号位相加后若有进位，则舍去该进位数字。

在模2意义下，任意两数的补码之和等于该两数之和的补码。

这是补码加法的理论基础。

补码加法的特点：

- (1) 符号位要作为数的一部分一起参加运算；
- (2) 在模2的意义下相加，即大于2的进位要丢掉。

其结论也适用于定点整数。

例:  $x=0.1001$ ,  $y=0.0101$ , 求  $x+y$ 。

解:  $[x]_{\text{补}}=0.1001$ ,  $[y]_{\text{补}}=0.0101$

$$\begin{array}{r} [x]_{\text{补}} \quad 0.1001 \\ + [y]_{\text{补}} \quad 0.0101 \\ \hline [x+y]_{\text{补}} \quad 0.1110 \end{array}$$

所以  $x+y=+0.1110$

例:  $x=+0.1011$ ,  $y=-0.0101$ , 求  $x+y$ 。

解:  $[x]_{\text{补}}=0.1011$ ,  $[y]_{\text{补}}=1.1011$

$$\begin{array}{r} [x]_{\text{补}} \quad 0.1011 \\ + [y]_{\text{补}} \quad 1.1011 \\ \hline [x+y]_{\text{补}} \quad 1.0110 \end{array}$$

所以  $x+y=0.0110$

### 3. 补码减法

补码减法运算的公式：

$$[x - y]_{\text{补}} = [x]_{\text{补}} - [y]_{\text{补}} = [x]_{\text{补}} + [-y]_{\text{补}}$$

两数差的补码等于两数补码之差

公式证明： 只要证明 $[-y]_{\text{补}} = -[y]_{\text{补}}$ ，上式即得证。

证明：

$$\because [x + y]_{\text{补}} = [x]_{\text{补}} + [y]_{\text{补}} \quad (\text{mod } 2)$$

$$\text{令 } y = -x$$

$$\therefore [0]_{\text{补}} = [x]_{\text{补}} + [-x]_{\text{补}}$$

$$\text{故 } [-x]_{\text{补}} = -[x]_{\text{补}} \quad (\text{mod } 2)$$

减法运算化为加法完成。关键是求 $[-Y]_{\text{补}}$

例:  $x = +0.1101$ ,  $y = +0.0110$ , 求  $x - y$ 。

解:  $[x]_{\text{补}} = 0.1101$

$[y]_{\text{补}} = 0.0110$   $[-y]_{\text{补}} = 1.1010$

$$\begin{array}{r}
 [x]_{\text{补}} \quad \quad \quad 0.1 \ 1 \ 0 \ 1 \\
 + [-y]_{\text{补}} \quad \quad 1.1 \ 0 \ 1 \ 0 \\
 \hline
 [x-y]_{\text{补}} \quad \quad 1 \ 0.0 \ 1 \ 1 \ 1
 \end{array}$$

$\therefore x - y = +0.0111$

例:  $x = -0.1101$ ,  $y = -0.0110$ , 求  $x - y = ?$

解:  $[x]_{\text{补}} = 1.0011$

$[y]_{\text{补}} = 1.1010$

$[-y]_{\text{补}} = 0.0110$

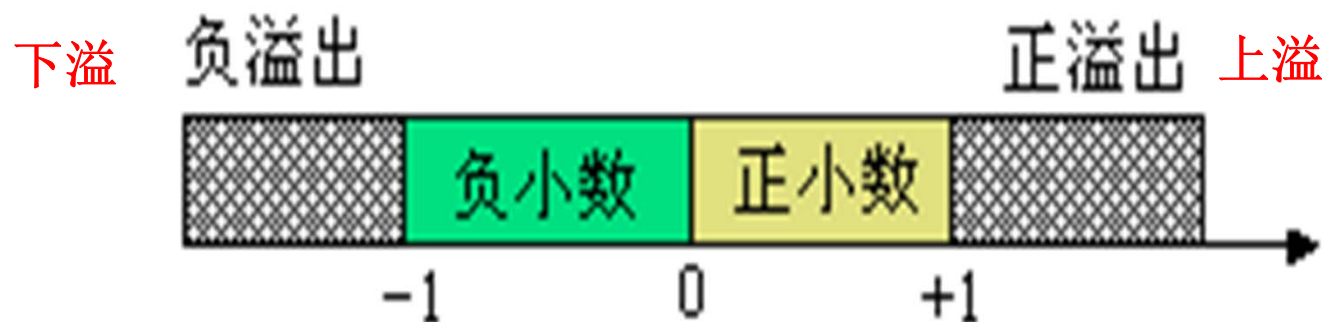
$$\begin{array}{r}
 [x]_{\text{补}} \quad \quad \quad 1.0 \ 0 \ 1 \ 1 \\
 + [-y]_{\text{补}} \quad \quad 0.0 \ 1 \ 1 \ 0 \\
 \hline
 [x-y]_{\text{补}} \quad \quad 1.1 \ 0 \ 0 \ 1
 \end{array}$$

$\therefore x - y = -0.0111$

# 溢出及与检测方法

## 1. 概念

在定点小数机器中, 数的表示范围为  $|x| < 1$ 。在运算过程中如出现大于1的现象, 称为“溢出”。



机器定点小数表示

发生溢出的原因, 是因为运算结果超出编码所能表示的数字大小。  
两个正数相加: 结果大于机器所能表示的最大正数, 称为上溢;  
两个负数相加: 结果小于机器所能表示的最小负数, 称为下溢。

例:  $x=+0.1011$ ,  $y=+0.1001$ , 求 $x+y$ 。

解:

$$\begin{array}{r}
 [x]_{\text{补}} = 0.1011 \qquad [y]_{\text{补}} = 0.1001 \\
 \begin{array}{r}
 [x]_{\text{补}} \\
 + [y]_{\text{补}} \\
 \hline
 [x+y]_{\text{补}}
 \end{array}
 \begin{array}{r}
 0.1011 \\
 0.1001 \\
 \hline
 1.0100
 \end{array}
 \end{array}$$

$$\begin{array}{r}
 0.10101 \\
 + 0.01000 \\
 \hline
 0.11101
 \end{array}$$

正常结果

两个正数相加的结果成为负数，这显然是错误的。

例:  $x=-0.1101$ ,  $y=-0.1011$ , 求 $x+y$ 。

解:

$$\begin{array}{r}
 [x]_{\text{补}} = 1.0011 \qquad [y]_{\text{补}} = 1.0101 \\
 \begin{array}{r}
 [x]_{\text{补}} \\
 + [y]_{\text{补}} \\
 \hline
 [x+y]_{\text{补}}
 \end{array}
 \begin{array}{r}
 1.0011 \\
 1.0101 \\
 \hline
 0.1000
 \end{array}
 \end{array}$$

$$\begin{array}{r}
 1.10101 \\
 + 1.11000 \\
 \hline
 1.11011
 \end{array}$$

正常结果

两个负数相加的结果成为正数，这同样是错误的。



## 2. 溢出的检测方法

$$\begin{array}{r} [x]_{\text{补}} \quad 0.1011 \\ + [y]_{\text{补}} \quad 0.1001 \\ \hline [x+y]_{\text{补}} \quad 1.0100 \end{array}$$

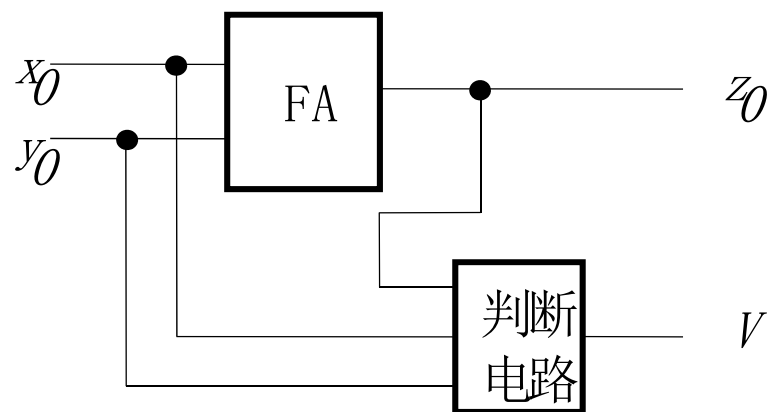
$$\begin{array}{r} [x]_{\text{补}} \quad 1.0011 \\ + [y]_{\text{补}} \quad 1.0101 \\ \hline [x+y]_{\text{补}} \quad 0.1000 \end{array}$$

### (1) 单符号位检测方法1

设两数符号位分别为  $S_1$ 、 $S_2$   
和数符号位  $S_c$

溢出逻辑表达式为：

$$V = \bar{S}_1 \bar{S}_2 S_c + S_1 S_2 \bar{S}_c$$



判断电路

## (2) 单符号位检测方法2

符号位进位 $C_f$ ，最高位进位 $C_n$

$$\begin{array}{r} 0.10101 \\ + 0.01000 \\ \hline \end{array}$$

0.11101

$C_f = 0, C_n = 0$

$$\begin{array}{r} 0.\textcolor{teal}{1}0101 \\ + 0.\textcolor{teal}{1}1000 \\ \hline \end{array}$$

$\boxed{1}.01101$

$C_f = 0, C_n = 1$

$$\begin{array}{r} 1.\textcolor{teal}{1}0101 \\ + 1.\textcolor{teal}{1}1000 \\ \hline \end{array}$$

$\boxed{1}1.01101$

$C_f = 1, C_n = 1$

$$\begin{array}{r} 1.00101 \\ + 1.11000 \\ \hline \end{array}$$

$\boxed{1}\boxed{0}.11101$

$C_f = 1, C_n = 0$

从上面例中看到：

当最高有效位有进位而符号位无进位时，产生上溢；

当最高有效位无进位而符号位有进位时，产生下溢。

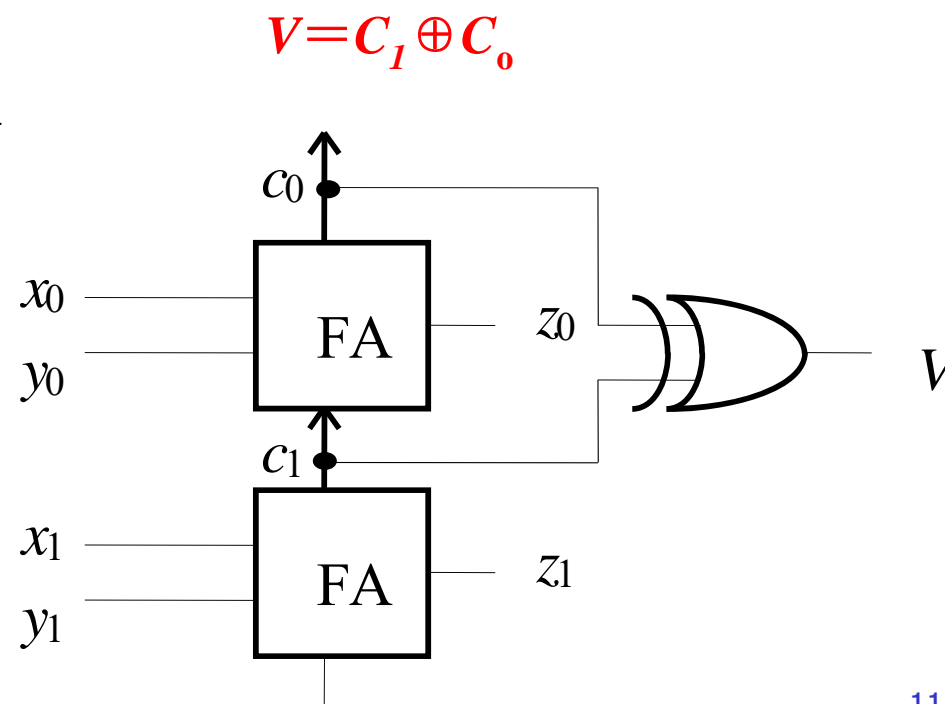
（简单地说是正数相加为负数或负数相加为正数则产生溢出）

故溢出逻辑表达式为： $V = C_f \oplus C_o$

其中 $C_f$ 为符号位产生的进位， $C_o$ 为最高有效位产生的进位。

此逻辑表达式也可用异或门实现。

判断电路



### (3) 双符号位法

一个符号位只能表示正、负两种情况，当产生溢出时，符号位的含义就会发生混乱。如果将符号位扩充为两位( $S_{f1}$ 、 $S_{f2}$ )，其所能表示的信息量将随之扩大，既能判别是否溢出，又能指出结果的符号。

双符号位法也称为“变形补码”或“模4补码”。

定点小数变形补码定义：

$$[x]_{\text{补}} = \begin{cases} x & 0 \leq x < 1 \\ 4+x & -1 \leq x < 0 \end{cases} \quad (\text{mod } 4)$$

字长 $n+2$ 定点整数，变形补码定义：

$$[x]_{\text{补}} = \begin{cases} x & 0 \leq x < 2^n \\ 2^{n+2} + x & -2^n \leq x < 0 \end{cases} \quad (\text{mod } 2^{n+2})$$

采用变形补码后数的表示：

- 任何小于1的正数：两个符号位都是“0”，即  $00.x_1x_2\cdots x_n$ ；
- 任何大于-1的负数：两个符号位都是“1”，即  $11.x_1x_2\cdots x_n$

模4补码加法公式： $[x]_{\text{补}} + [y]_{\text{补}} = [x+y]_{\text{补}} \pmod{4}$

两数变形补码之和等于两数和的变形补码，要求：

- 两个符号位都看做数码一样参加运算；
- 两数进行以4为模的加法，即最高符号位上产生的进位要丢掉。

## 双符号数溢出检测

$$\begin{array}{r} 00.10101 \\ + 00.01000 \\ \hline 00.11101 \end{array}$$

正常结果

$$\begin{array}{r} 00.10101 \\ + 00.11000 \\ \hline 01.01101 \end{array}$$

非正常符号位，溢出

$$\begin{array}{r} 11.10101 \\ + 11.11000 \\ \hline 111.01101 \end{array}$$

符号位进位舍去，正常结果

$$\begin{array}{r} 11.00101 \\ + 11.11000 \\ \hline 110.11101 \end{array}$$

非正常符号位，溢出

双符号位的含义如下：

$S_{f1}S_{f2} = 00$       结果为正数，无溢出

01                      结果正溢

10                      结果负溢

11                      结果为负数，无溢出

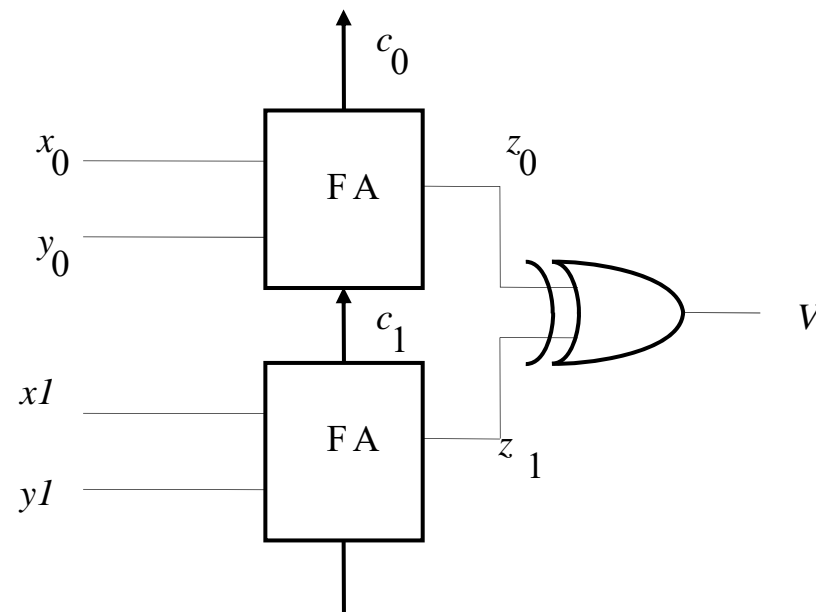
即：结果的两个符号位的代码不一致时，表示溢出；

两个符号位的代码一致时，表示没有溢出。

不管溢出与否，最高符号位永远表示结果的正确符号。

溢出逻辑表达式为： $V = S_{f1} \oplus S_{f2}$

式中： $S_{f1}$ 和 $S_{f2}$ 分别为最高符号位和第二符号位，此逻辑表达式可用异或门实现。



例  $x = +0.1100$ ,  $y = +0.1000$ , 求  $x+y$ 。

解:

$$\begin{array}{r}
 [x]_{\text{补}} = 00.1100 \qquad [y]_{\text{补}} = 00.1000 \\
 \begin{array}{r}
 [x]_{\text{补}} \quad 0 \ 0. \ 1 \ 1 \ 0 \ 0 \\
 + \quad [y]_{\text{补}} \quad 0 \ 0. \ 1 \ 0 \ 0 \ 0 \\
 \hline
 0 \ 1. \ 0 \ 1 \ 0 \ 0
 \end{array}
 \end{array}$$

符号位出现“01”，表示已溢出，正溢。即结果大于+1

例  $x = -0.1100$ ,  $y = -0.1000$ , 求  $x+y$ 。

解:

$$\begin{array}{r}
 [x]_{\text{补}} = 11.0100 \qquad [y]_{\text{补}} = 11.1000 \\
 \begin{array}{r}
 [x]_{\text{补}} \quad 1 \ 1. \ 0 \ 1 \ 0 \ 0 \\
 + \quad [y]_{\text{补}} \quad 1 \ 1. \ 1 \ 0 \ 0 \ 0 \\
 \hline
 1 \ 0. \ 1 \ 1 \ 0 \ 0
 \end{array}
 \end{array}$$

符号位出现“10”，表示已溢出，负溢出。即结果小于-1



# 基本的二进制加法/减法器

## 1. 一位全加器

逻辑方程

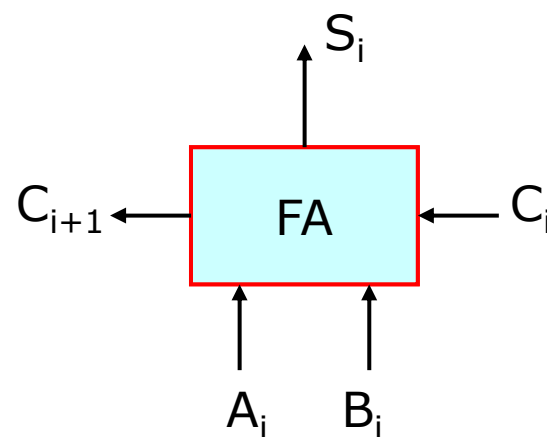
$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_{i+1} = A_i B_i + (A_i \oplus B_i) C_i$$

一位全加器真值表

输入			输出	
$A_i$	$B_i$	$C_i$	$S_i$	$C_{i+1}$
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

⇒

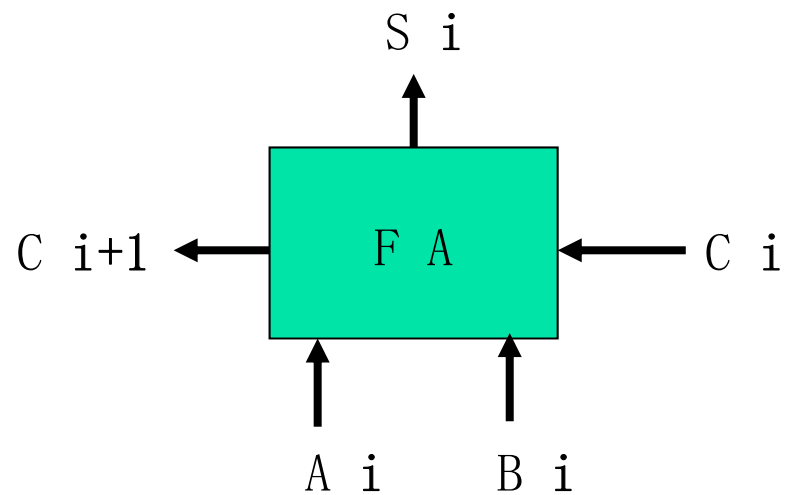


一位全加器

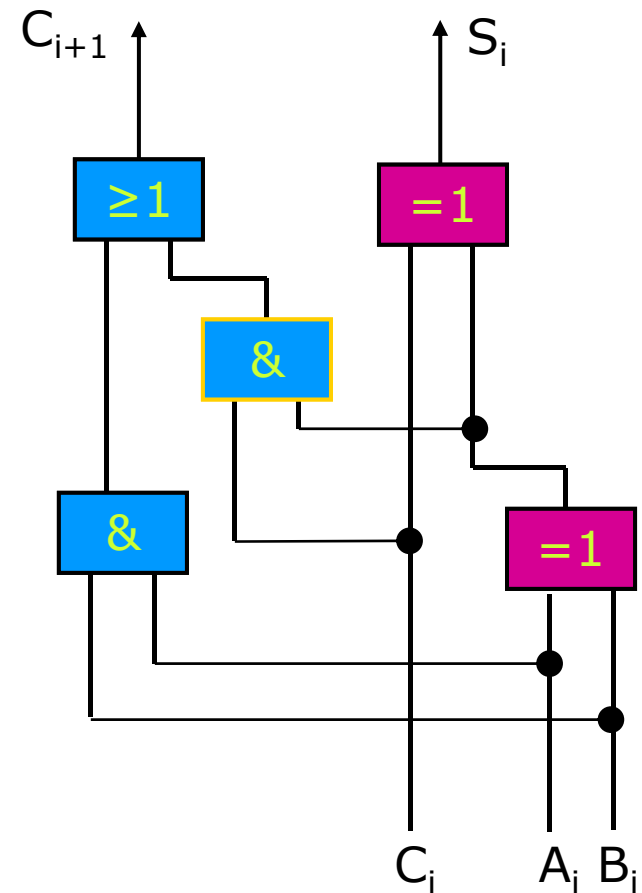
## 逻辑方程

$$S_i = A_i \oplus B_i \oplus C_i$$

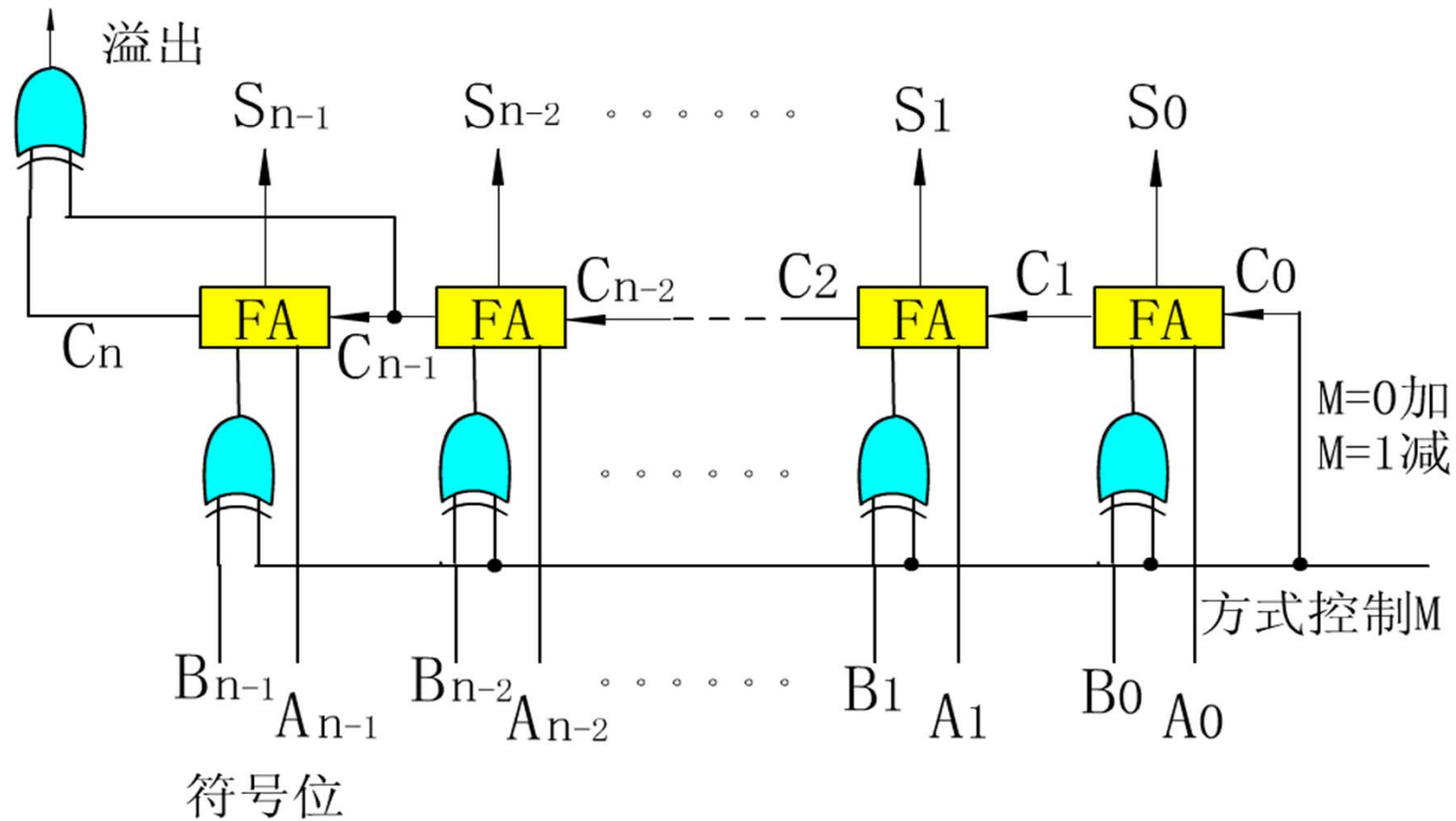
$$C_{i+1} = A_i B_i + (A_i \oplus B_i) C_i$$



逻辑符号






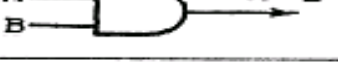
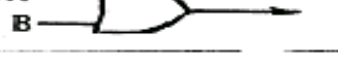
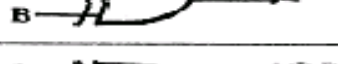
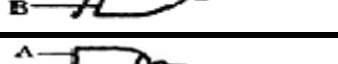
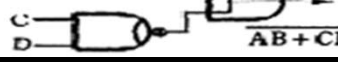
## 2. $n$ 位的行波进位加减器



$n$ 个1位的全加器 (FA) 可级联成一个  $n$  位的行波进位加减器。

### 3. n位的行波进位加法器的问题

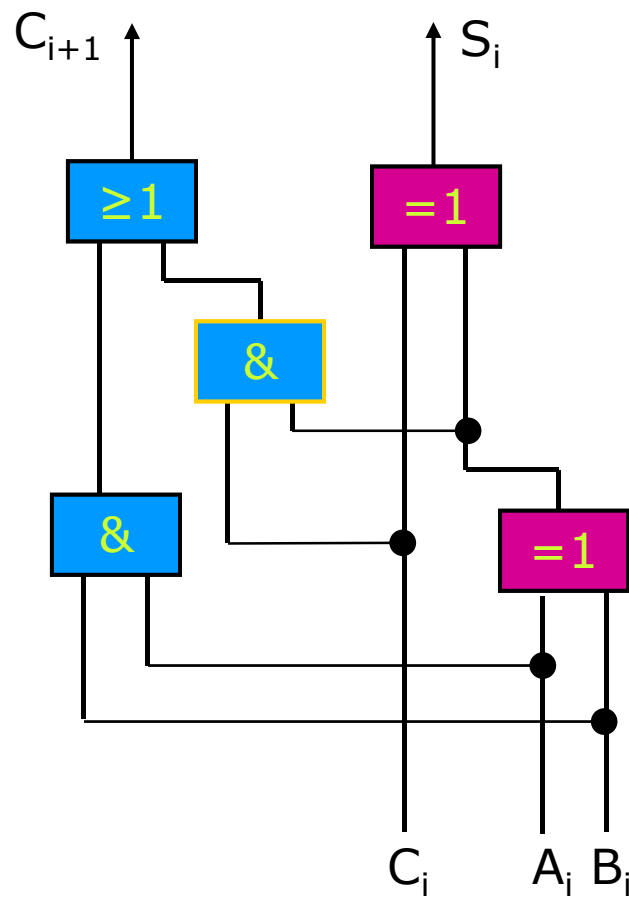
#### 典型门电路的逻辑符号和延迟时间

门的名称	门的功能	逻辑符号（正逻辑）	时间延迟
与非	NAND		T
或非	NOR		T
非	NOT		T
与	AND		2T
或	OR		2T
异或	XOR		3T
异或非	XNOR		3T
接线逻辑 (与或非)	AOI		$T + T_{RC}$

T被定义为相应于单级逻辑电路的单位门延迟。

T通常采用一个“与非”门或一个“或非”门的时间延迟来作为度量单位。

(1) 对一位全加器 (FA) 来说,  $S_i$  的时间延迟为  $6T$  (每级异或门延迟  $3T$ );  
 $C_{i+1}$  的时间延迟为  $5T$ 。



(2)  $n$ 位行波进位加法器的延迟时间  $t_a$  为:

考虑溢出检测时, 有:  $t_a = n \cdot 2T + 9T = (2n + 9)T$

- $9T$  为最低位上的两极“异或”门再加上溢出“异或”门的总时间;
- $2T$  为每级进位链的延迟时间。

当不考虑溢出检测时, 有:  $t_a = (n - 1) \cdot 2T + 9T$

$t_a$  为在加法器的输入端输入加数和被加数后, 在最坏的情况下加法器输出端得到稳定的求和输出所需要的最长时间。

$t_a$  越小越好。

由一位全加器 (FA) 构成的行波进位加法器:

缺点:

- (1) 串行进位, 它的运算时间长;
- (2) 只能完成加法和减法两种操作而不能完成逻辑操作。

能否提前产生各位的进位输入?

使得各位的加法运算能并行起来, 即可提高多位加法器运算速度

## 并行加法器进位链

- $S_i = A_i \oplus B_i \oplus C_{i-1}$
- $C_i = A_i B_i + (A_i \oplus B_i) C_{i-1}$
- $G_i = A_i B_i$   $G_i$  进位生成函数 Generate
- $P_i = A_i \oplus B_i$   $P_i$  进位传递函数 Propagate
- $C_i = G_i + P_i C_{i-1}$
- $C_n = A_n B_n + (A_n \oplus B_n) C_{n-1} = G_n + P_n C_{n-1}$
- $C_{n-1} = A_{n-1} B_{n-1} + (A_{n-1} \oplus B_{n-1}) C_{n-2} = G_{n-1} + P_{n-1} C_{n-2}$
- .....
- $C_1 = A_1 B_1 + (A_1 \oplus B_1) C_0 = G_1 + P_1 C_0$
- 高位的运算依赖于低位运算的进位输入 计算不能并行
- 能否提前得到当前位的进位输入??

## 并行加法器进位链

$$C_1 = A_1B_1 + (A_1 \oplus B_1)C_0 = G_1 + P_1C_0$$

$$\begin{aligned} C_2 &= A_2B_2 + (A_2 \oplus B_2)C_1 = G_2 + P_2C_1 \\ &= G_2 + P_2(G_1 + P_1C_0) \\ &= G_2 + P_2G_1 + P_2P_1C_0 \end{aligned}$$

$$\begin{aligned} C_3 &= A_3B_3 + (A_3 \oplus B_3)C_2 = G_3 + P_3C_2 \\ &= G_3 + P_3(G_2 + P_2G_1 + P_2P_1C_0) \\ &= G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1C_0 \end{aligned}$$

$$C_{n-1} = G_{n-1} + P_{n-1}G_{n-2} + P_{n-1}P_{n-2}G_{n-3} \cdots + P_n P_{n-1} \cdots P_1 C_0$$

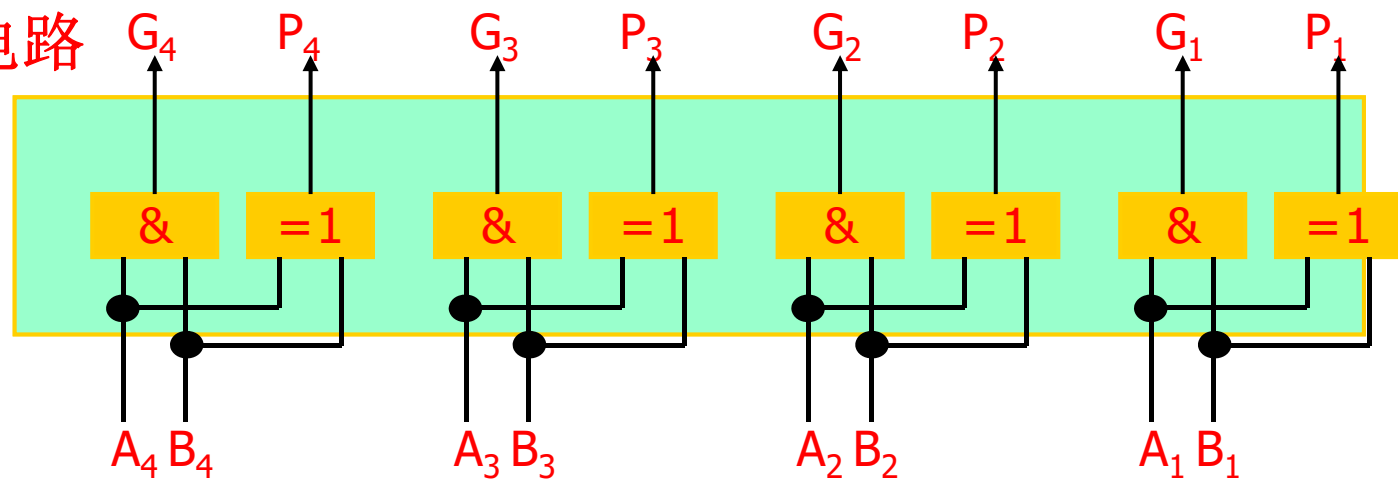
.....

$$C_n = G_n + P_n G_{n-1} + P_n P_{n-1} G_{n-2} + P_n P_{n-1} P_{n-2} G_{n-3} \cdots + P_n P_{n-1} P_{n-2} \cdots P_1 C_0$$

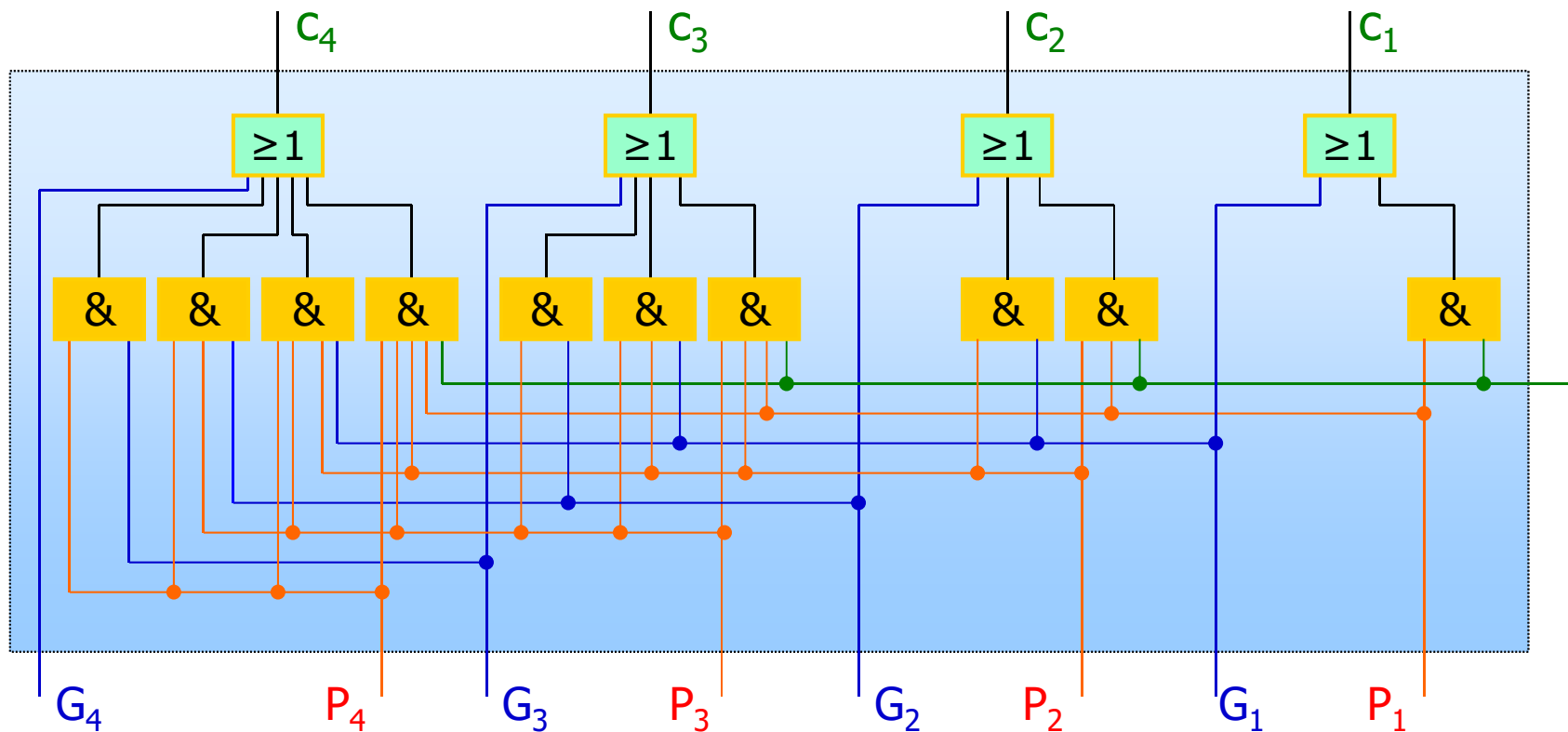
位数越长，进位链电路复杂度越高  
通常按照**4**位一组进行分组运算



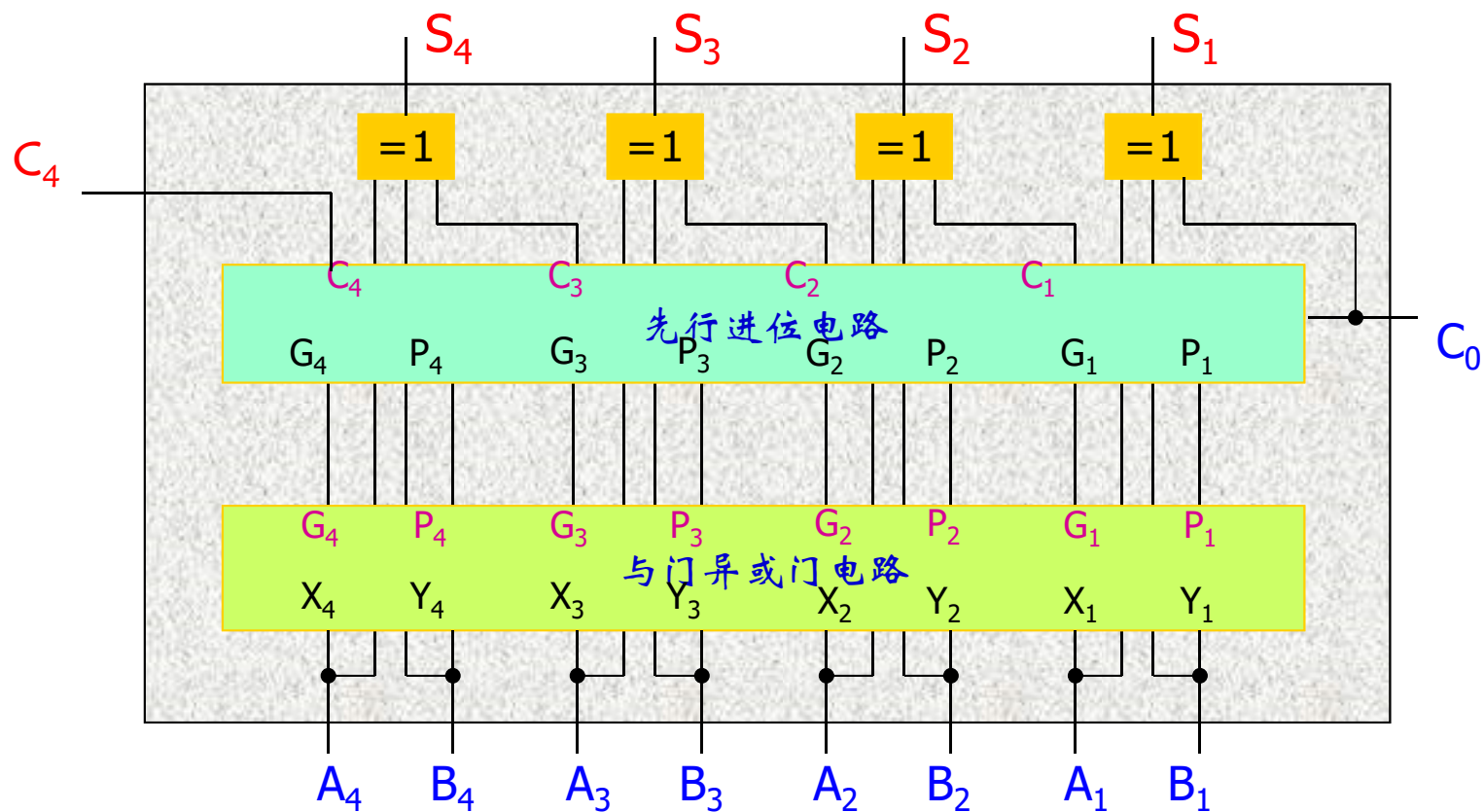
# 与门异或门电路



## 先行进位电路



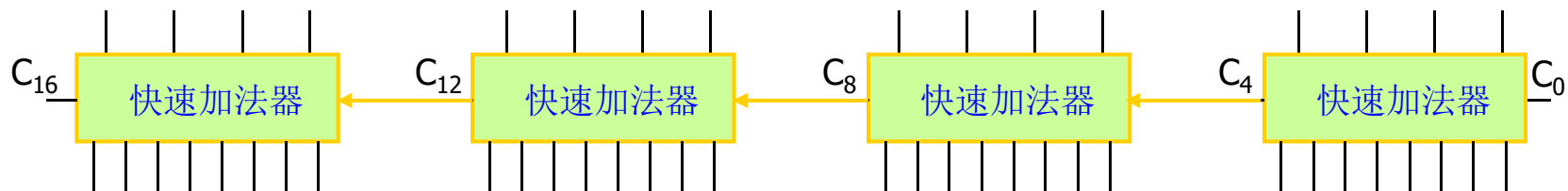
## 四位快速加法器



# 16位加法器



- 组内先行进位
- 组间串行进位
- 可否组间并行？

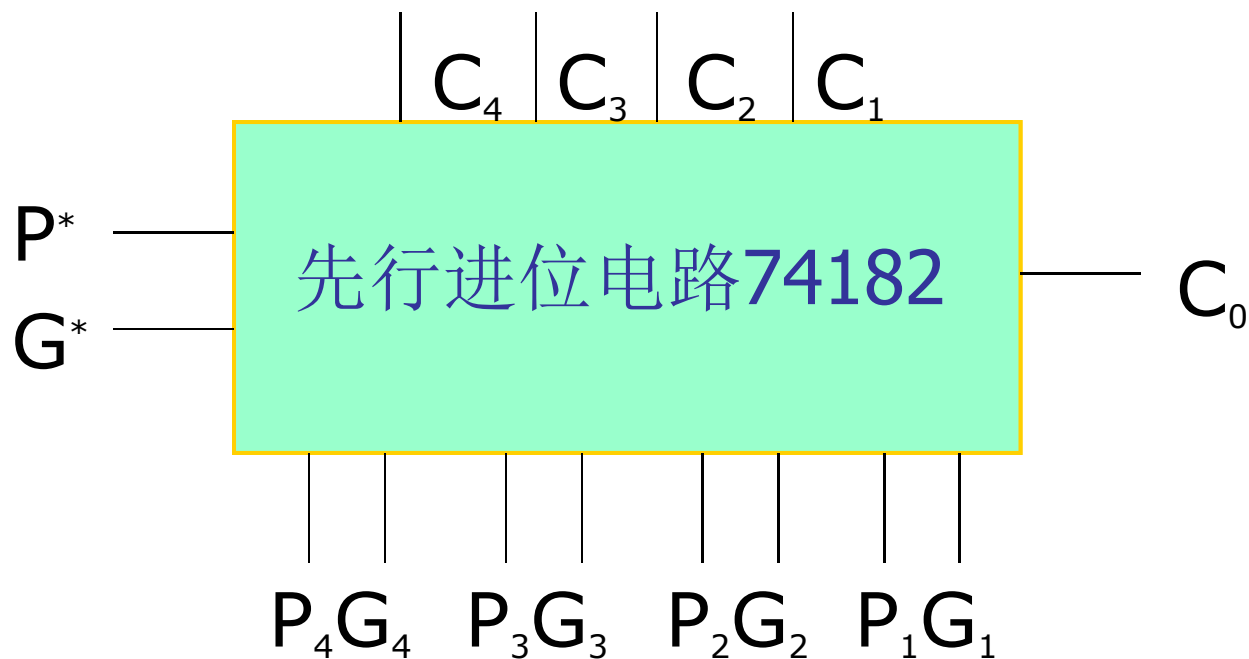


# 成组进位

- $C_4 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_0$
- $G_4^* = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1$  成组进位发生输出
- $P_4^* = P_4 P_3 P_2 P_1$  成组进位传递函数
- $C_4 = G_4^* + P_4^* C_0$
- $C_1 = G_1 + P_1 C_0$  比较原相邻位进位公式
- $C_4 = G_4^* + P_4^* C_0$
- $C_8 = G_8^* + P_8^* (G_4^* + P_4^* C_4)$   
 $= G_8^* + P_8^* G_4^* + P_8^* P_4^* C_0$
- $C_{16} = G_{16}^* + P_{16}^* G_{12}^* + P_{16}^* P_{12}^* G_8^* + P_{16}^* P_{12}^* P_8^* G_4^* + P_{16}^* P_{12}^* P_8^* P_4^* C_0$
- 用4组  $P^* G^*$  作输入, 即可复用原先行进位电路
- 产生组间先行进位信号

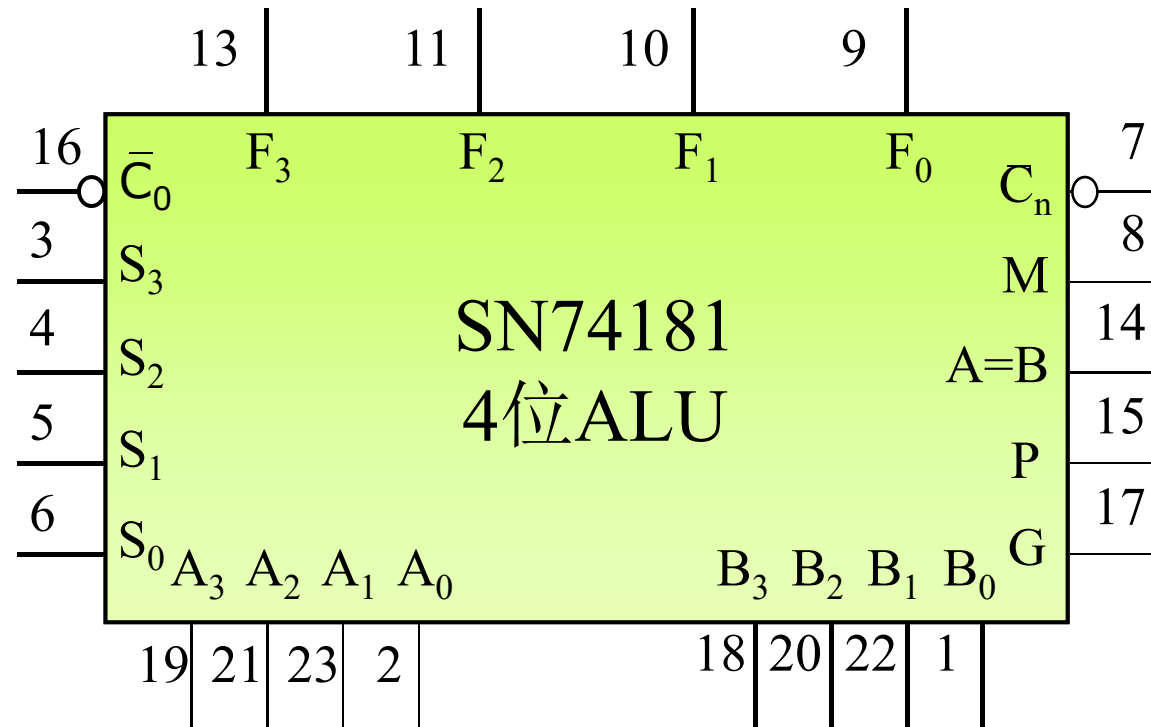
## 先行进位电路74182

- 输入:  $P_4G_4 \ P_3G_3 \ P_2G_2 \ P_1G_1 \ C_0$
- 输出: 先行进位输出  $C_4 \ C_3 \ C_2 \ C_1$   
成组进位传送输出  $P^*$   
成组进位发生输出  $G^*$
- $C_n = G_n + P_n G_{n-1} + P_n P_{n-1} G_{n-2} + P_n P_{n-1} P_{n-2} G_{n-3} \dots + P_n P_{n-1} \dots P_1 C_0$
- $G_i = X_i Y_i \quad P_i = X_i \oplus Y_i$



# ALU74181

- 先行进位的多功能算术/逻辑运算单元



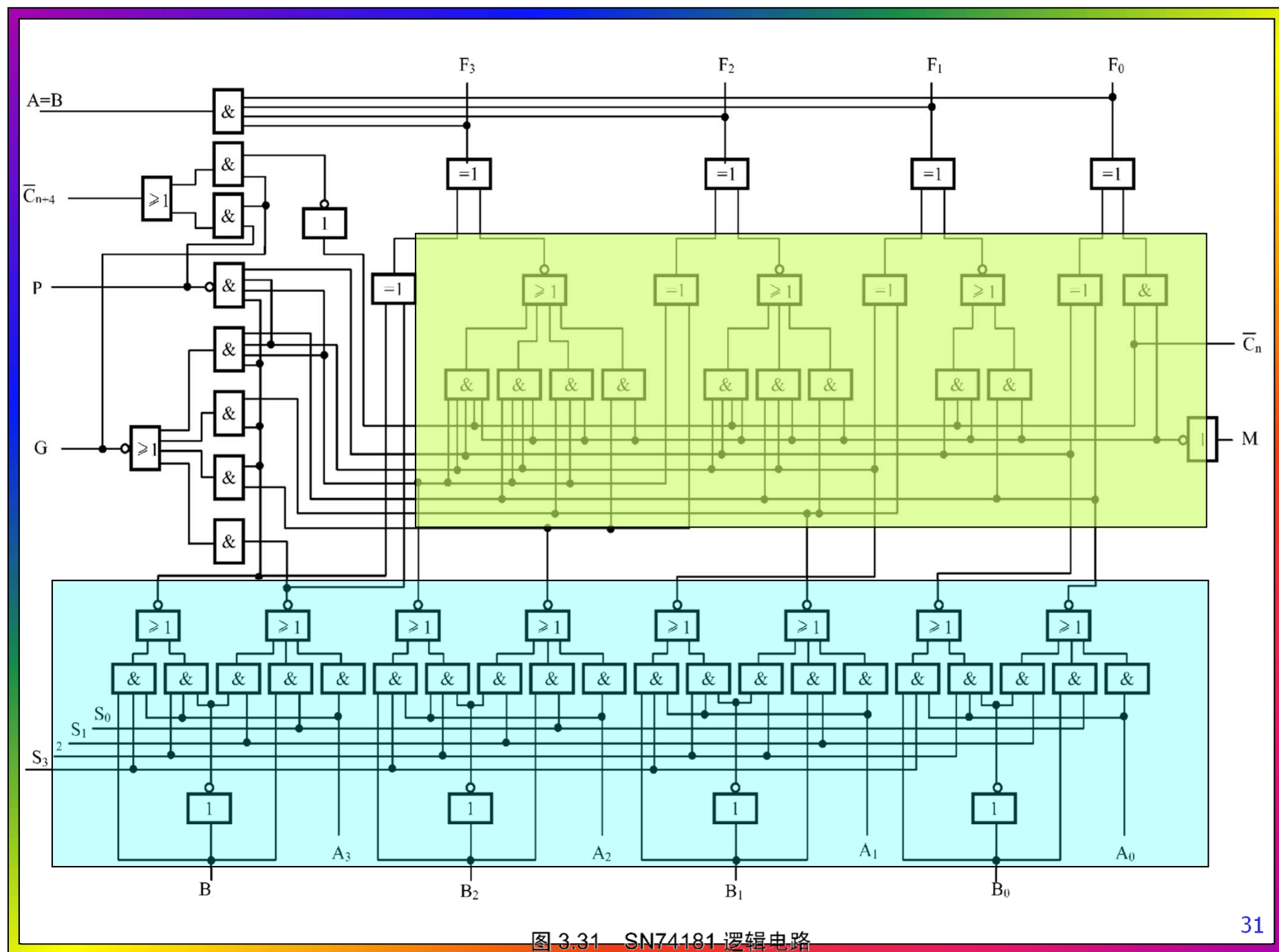
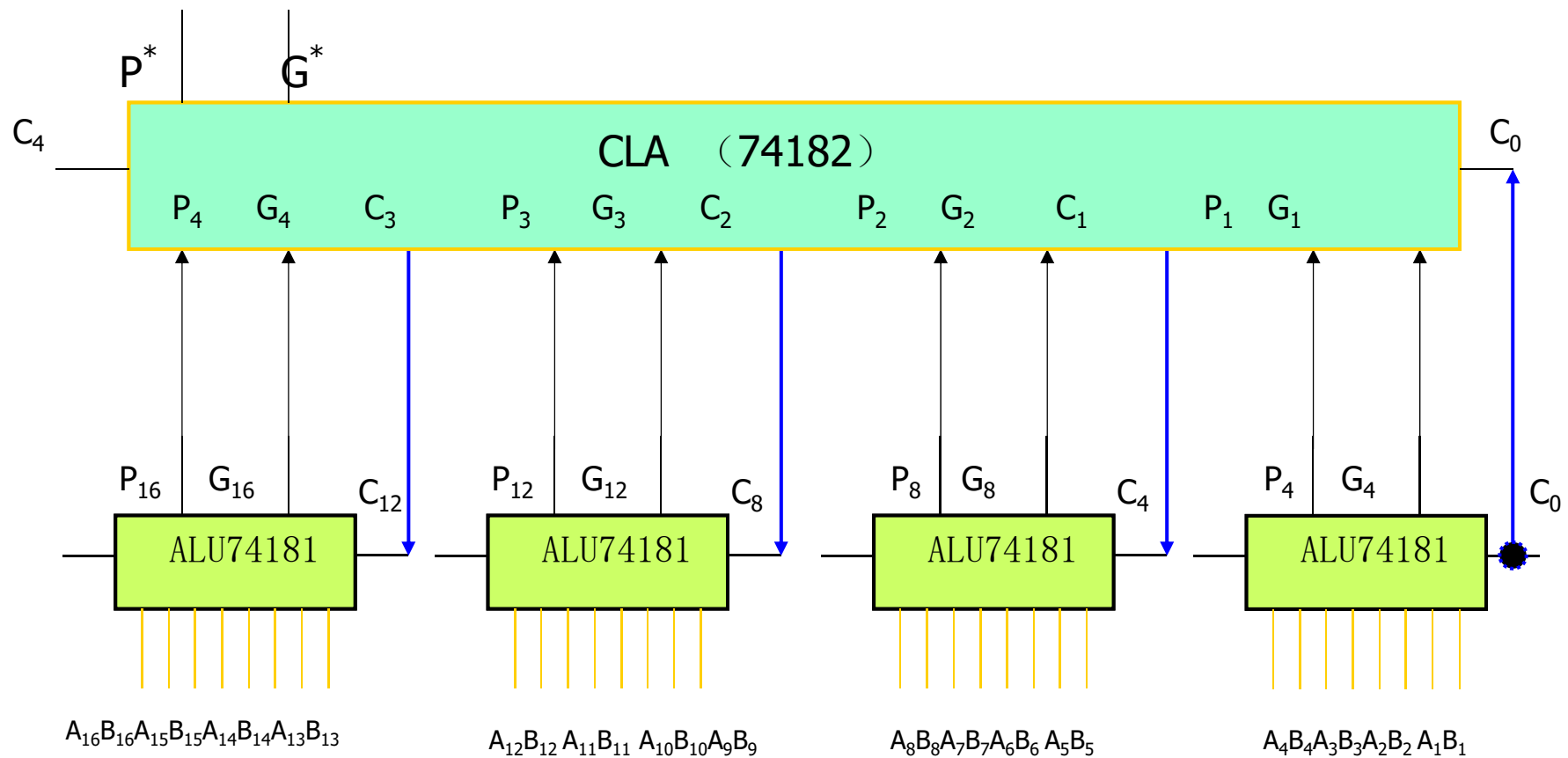


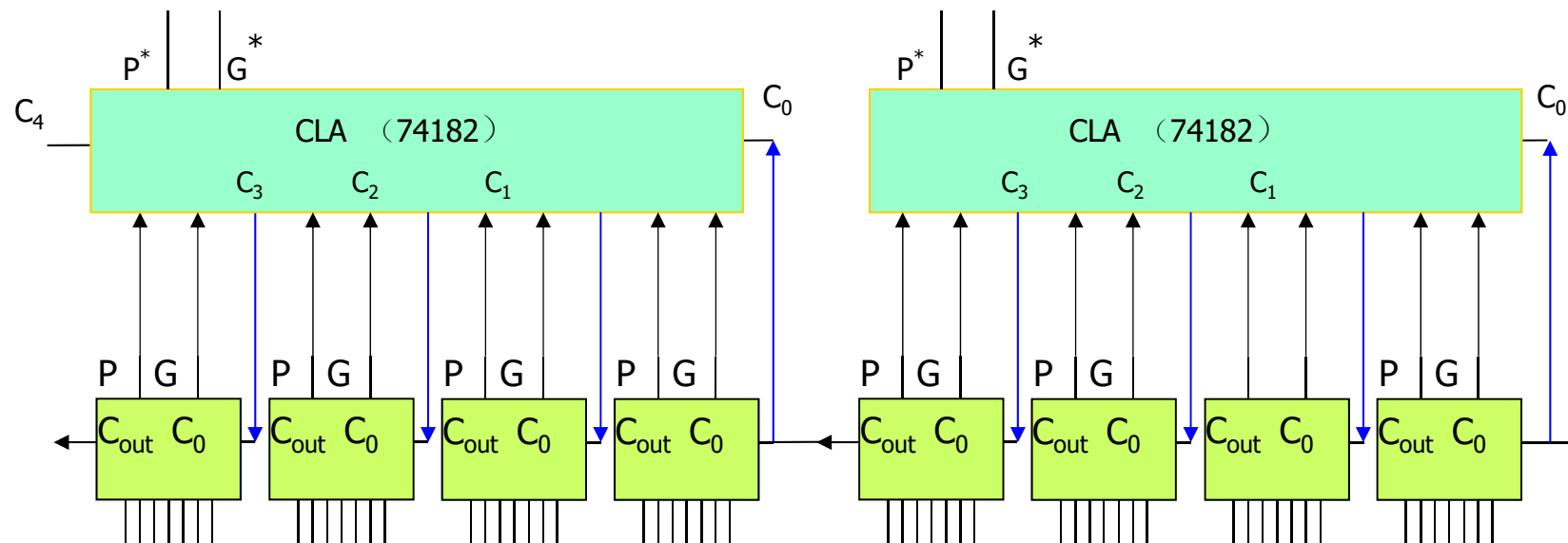
图 3.31 SN74181 逻辑电路

## 16位组内先行进位，组间先行进位

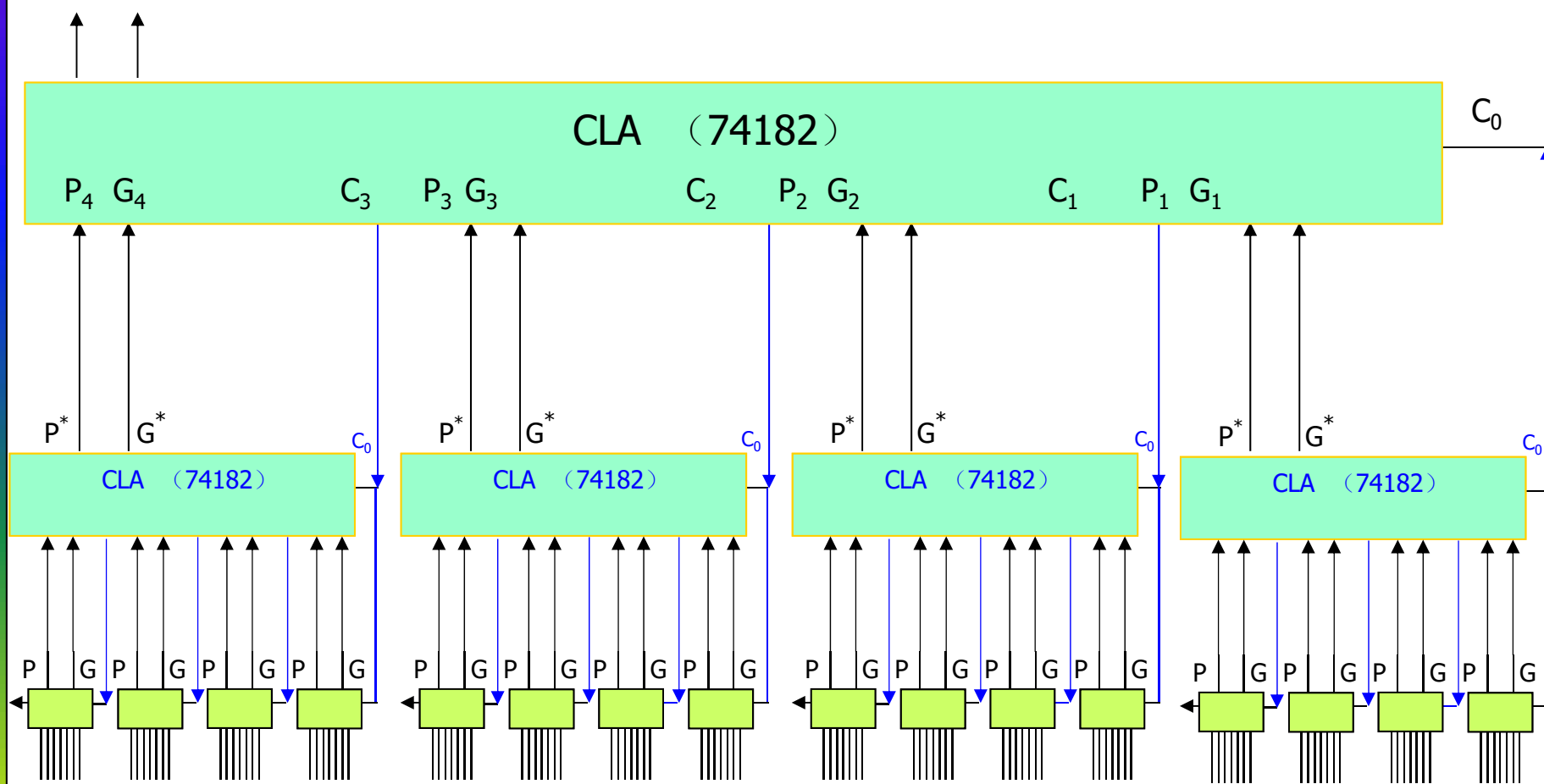




## 32位先行进位系统



# 64位先行进位系统



## 先行进位电路时间延迟分析

$$C_n = G_n + P_n G_{n-1} + P_n P_{n-1} G_{n-2} \dots + P_n P_{n-1} \dots P_1 C_0$$

假设所有门电路均按照2输入

$G_n$	需要1个门电路延迟
$P_n G_{n-1}$	需要2个门电路延迟
$P_n P_{n-1} G_{n-2}$	需要3个门电路延迟
$P_n P_{n-1} \dots P_1 C_0$	需要n+1个门电路延迟

考虑并发，时间延迟级别  $\lceil \log_2(2n+1) \rceil + 1$

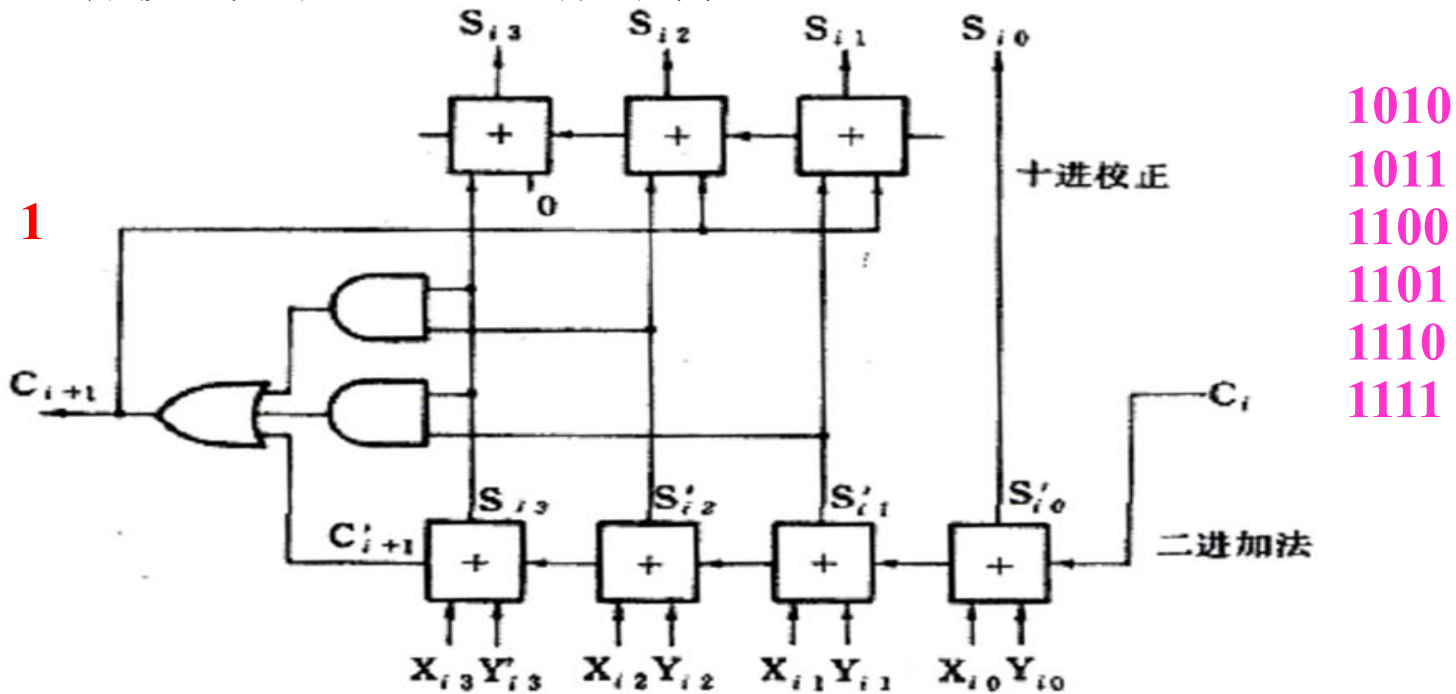
# 十进制加法器

十进制加法器可由BCD码(二一十进制码)来设计,它可以在二进制加法器的基础上加上适当的“校正”逻辑来实现。

<p><math>X+Y+C &lt; 10</math> 不调整</p>	$\begin{array}{r} 3 \\ + 5 \\ \hline 8 \end{array}$	$\longrightarrow$	$\begin{array}{r} 0011 \\ + 0101 \\ \hline 1000 \end{array}$	
<hr/>				
<p><math>X+Y+C &gt; 10</math> 调整</p>	$\begin{array}{r} 7 \\ + 6 \\ \hline 13 \end{array}$	$\longrightarrow$	$\begin{array}{r} 0111 \\ + 0110 \\ \hline 1101 \\ + 0110 \\ \hline 10011 \end{array}$	<p>(= D) (= 13)</p>
<p>和数(4位) 有进位 调整</p>	$\begin{array}{r} 28 \\ + 9 \\ \hline 37 \end{array}$	$\longrightarrow$	$\begin{array}{r} 0010\ 1000 \\ + 0000\ 1001 \\ \hline 0011\ 0001 \\ + 0000\ 0110 \\ \hline 0011\ 0111 \end{array}$	<p>(=31) (=37)</p>

故： 1. 和为10~15时，加6校正； 2. 和数有进位时，加6校正。

### 一位BCD码行波式进位加法器一般结构:



### n位BCD码行波式进位加法器一般结构:

