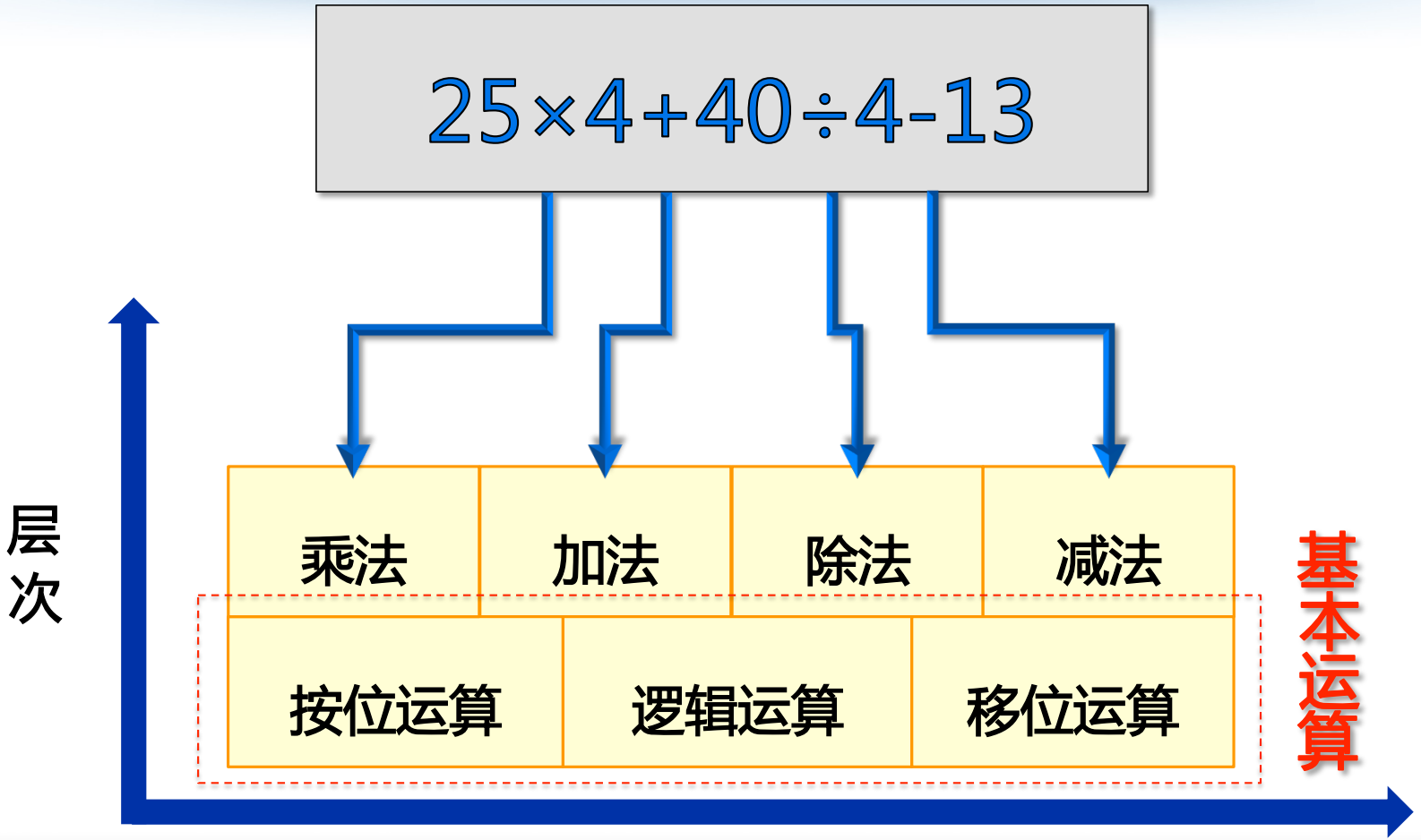
3.2 加法和减法运算

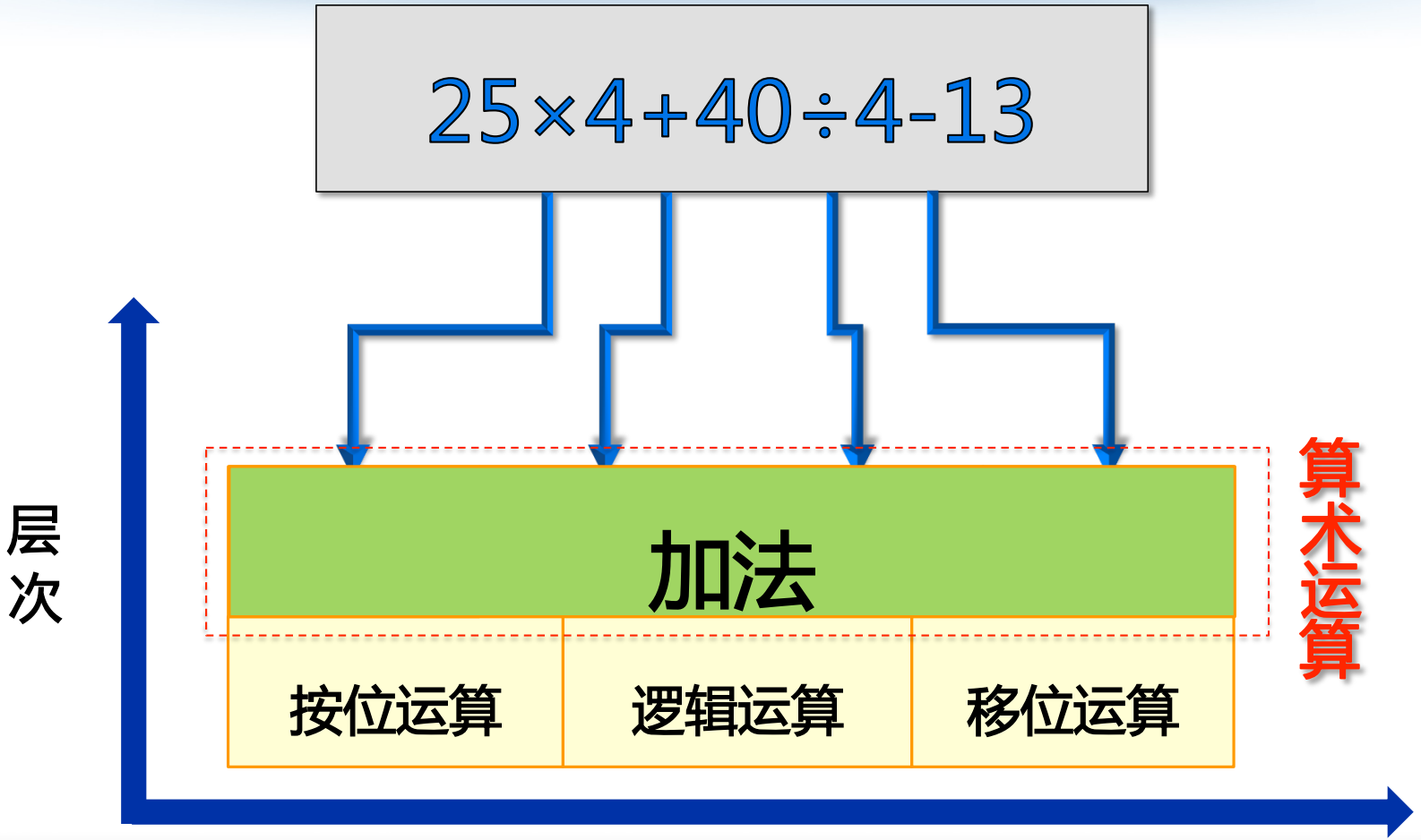
刘 芳 副教授

国防科学技术大学计算机学院

从一个算式说起



从一个算式说起



从十进制加法谈起

❑十进制加法，例：

$$\begin{array}{rr} & 245 \\ + & 673 \\ \hline & 918 \end{array}$$

❑加法表

+	0	1	2	3	4	5	6	7	8	9
0	0									
1	1	2								
2	2	3	4							
3	3	4	5	6						
4	4	5	6	7	8					
5	5	6	7	8	9	10				
6	6	7	8	9	10	11	12			
7	7	8	9	10	11	12	13	14		
8	8	9	10	11	12	13	14	15	16	
9	9	10	11	12	13	14	15	16	17	18

从十进制加法谈起

- 十进制加法，例：

$$\begin{array}{rr} & 245 \\ + & 673 \\ \hline & 918 \end{array}$$

- 加法表
  - 加法位表
  - 进位位表

加法位表

+	0	1	2	3	4	5	6	7	8	9
0	0									
1	1	2								
2	2	3	4							
3	3	4	5	6						
4	4	5	6	7	8					
5	5	6	7	8	9	0				
6	6	7	8	9	0	1	2			
7	7	8	9	0	1	2	3	4		
8	8	9	0	1	2	3	4	5	6	
9	9	0	1	2	3	4	5	6	7	8

进位位表

+	0	1	2	3	4	5	6	7	8	9
0	0									
1	0	0								
2	0	0	0							
3	0	0	0	0						
4	0	0	0	0	0					
5	0	0	0	0	0	1				
6	0	0	0	0	1	1	1			
7	0	0	0	1	1	1	1	1		
8	0	0	1	1	1	1	1	1	1	
9	0	1	1	1	1	1	1	1	1	1

由十进制加法到二进制加法

❑二进制加法表(真值表)

0	00	01
1	01	10
+	0	1

0	0	1
1	1	0
+	0	1

加法位表

0	0	0
1	0	1
+	0	1

进位位表

❑最简单的二进制加法示例：

无符号数加法

	1	0	0	1	0	0	1	0	1
+	~~1~~	1	0	1	1	0	1	1	0
	?	?	?	?	?	?	?	?	?	?

有符号怎么办？

有符号原码加法

❑同号

```
    1  0  0  1  0  0  1  0  1
+   1  1  0  1  1  0  1  1  0
?
```

2. 取同号

1. 数值位相加

❑异号

```
    1  0  0  1  0  0  1  0  1
+   0  1  0  1  1  0  1  1  0
?   0
```

1. 负数取补码

产生进位

2. 数值位相加

3. 结果为正

最高位没有进位怎么办？

有符号原码加法

❑异号(最高位有进位)

	1	0	0	1	0	0	1	0	1
	1	1	1	0	1	1	0	1	1
+	0	1	0	1	1	0	1	1	0
	0	1	0	0	1	0	0	0	1

❑异号(最高位无进位)

	1	1	0	1	1	0	1	1	0
+	0	0	0	1	0	0	1	0	1
	1								

1. 负数取补码

2. 数值位相加

无进位

3. 结果为负

4. 取补得到原码

原码二进制加法规则

- 加法规则：符号位和数值部分分别处理
  - 同号：数值位相加，若最高位产生进位，则结果溢出。和的符号取被加数的符号
  - 异号：负数取补码，与正数相加，分二种情况讨论：

    a)最高数值位产生进位，符号位为0，表明加法结果为正，所得数值位正确。

    b)最高数值位没有产生进位，符号位为1，表明加法结果为负，得到的是数值位的补码形式，需对结果求补，得到原码结果

原码二进制加法规则

- 加法规则：符号位和数值部分分别处理
  - 同号：数值位相加，若最高位产生进位，则结果溢出。和的符号取被加数的符号
  - 异号：负数取补码，与正数相加，分二种情况讨论：

    a)最高数值位产生进位，符号位为0，表明加法结果为正，所得数值位正确。

    b)最高数值位没有产生进位，符号位为1，表明加法结果为负，得到的是数值位的补码形式，需对结果求补，得到原码结果

仅对数值部分进行加减运算，符号位起判断和控制作用，复杂！

补码

- 补码的表示
  - 符号部分同原码
    - 数的最高位为符号位，0表示正数，1表示负数
  - 数字部分与它的符号位有关
    - 对于正数，补码数值部分与原码数值部分相同
    - 对于负数，补码数值部分是将原码数值部分按位取反再加1，即在反码数值部分基础上加1

$X = +1101$	$Y = -1110$
$[X]_{原} = 00001101$	$[Y]_{原} = 10001110$
$[X]_{补} = 00001101$	$[Y]_{补} = 11110010$

补码加法

❑补码加法：符号位参与运算

$$[A + B]_{补} = [A]_{补} + [B]_{补} \quad (mod2)$$

例1：A= 1011， B= -1110， 求$[A+B]_{补}$

解：$[A]_{补}$=0 1011，$[B]_{补}$=1 0010

$[A+B]_{补}$=1 1101

$$\begin{array}{rr} & 0\ 1011 \\ + & 1\ 0010 \\ \hline & 1\ 1101 \end{array}$$

补码减法如何实现？

补码减法

❑补码表示法可以简化加法运算，并且可以将减法变成加法

$$[A - B]_{补} = [A]_{补} + [-B]_{补} \ (mod2)$$

例2： A= 1011， B= - 0010， 求$[A-B]_{补}$

解：$[A]_{补}$=0 1011， $[-B]_{补}$=0 0010

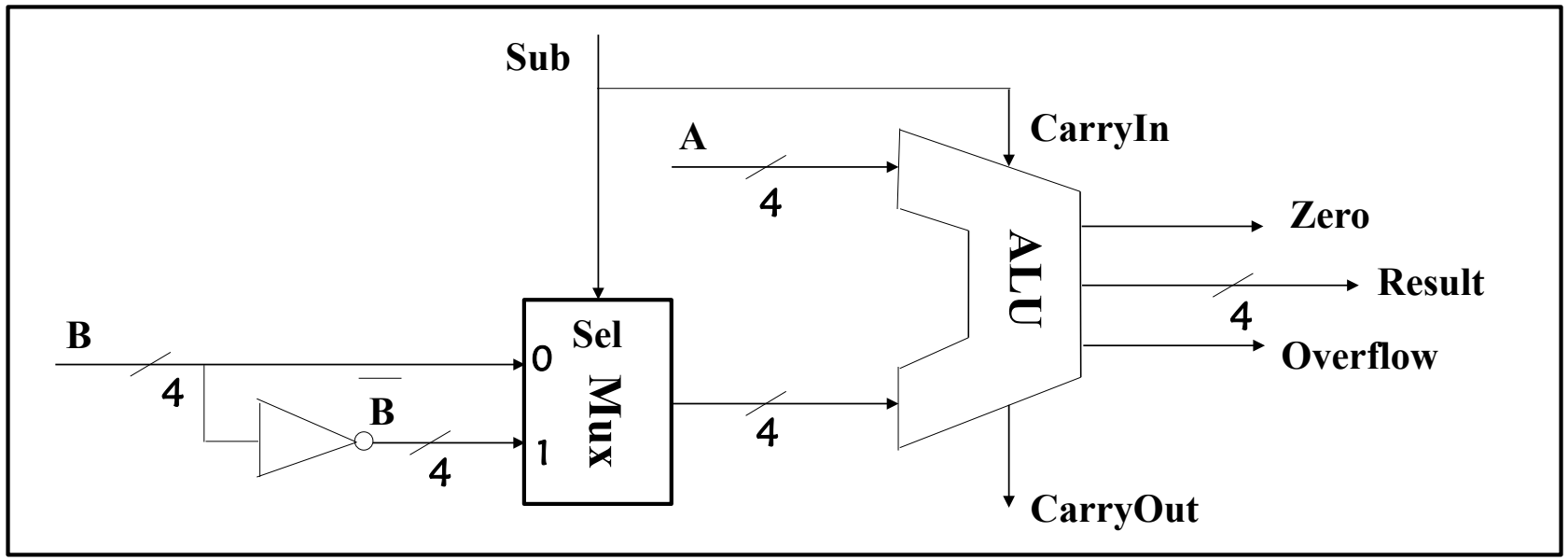
$[A-B]_{补}$=0 1101

$$\begin{array}{rr} & 0\ 1011 \\ + & 0\ 0010 \\ \hline & 0\ 1101 \end{array}$$

采用补码进行加减法运算，在计算机中只需要一套实现加法运算的电路，从而简化了计算机内部硬件电路的结构

补码加法器的基本实现

■加法运算和减法运算使用同一个加法电路，简化了运算器的设计



补码示例

❑补码运算（用5位二进制补码表示数）

A=+1111, B= - 1101, 求A+B

$[A+B]_{补}= [A]_{补} + [B]_{补}$

$[A]_{补}=01111, [B]_{补}=10011,$

```
   0 1 1 1 1
+) 1 0 0 1 1
-------------
 1 0 0 0 1 0
```

进位标志位CF=1
溢出标志位OF=0

$[A+B]_{补}=00010$

补码示例

❑补码运算（用5位二进制补码表示数）

A= -1111，B= - 1101，　求A+B

$[A+B]_{补}= [A]_{补} + [B]_{补}$

$[A]_{补}$=10001, $[B]_{补}$=10011,

```
    1 0 0 0 1
+)  1 0 0 1 1
-------------
  1 0 0 1 0 0
```

进位标志位CF=1
溢出标志位OF=1

$[A+B]_{补}$ 溢出！

溢出

❑补码加法

■ 需要考虑溢出问题，即运算结果超出了机器能表示数的范围

A1 = + 1101，B2 = + 1001

A1 = -1011，B2 = -1100

求A1 + B2 ， A1 + B2

$[A_1]_{补} + [B_2]_{补} = 01101 + 01001 = 10110$

正溢出

$[A_1]_{补} + [B_2]_{补} = 10101 + 10100 = 01001$

负溢出

如何判断溢出？

溢出检测

- 检测方法：
  - 补码中采用两位符号位(变形补码)： $[x]_{补}=4+x \pmod 4$(模4补码)
  - 符号位仍然参与运算，结果中的两位符号标示溢出状态

符号位	相同	01	10
状态表示	正常	正溢出	负溢出

A= - 1101
B= - 1010
求 $[A+B]_{补}$

$[A]_{补}$=11 0011
$[B]_{补}$=11 0110

11 0011
+ 11 0110
10 1001

结果为负溢出

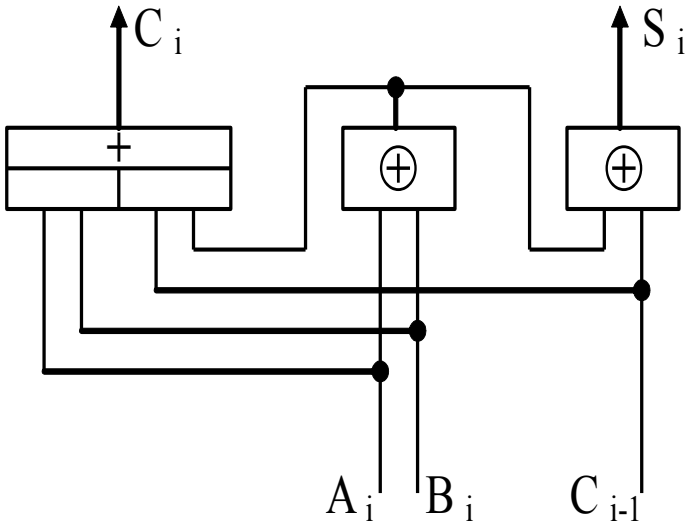
A= 1101
B= - 1010
求 $[A - B]_{补}$

$[A]_{补}$= 00 1101
$[-B]_{补}$= 00 1010

00 1101
+ 00 1010
01 0111

结果为正溢出

1位全加器



A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

$$C_i = (A_i+B_i)C_{i-1}+A_iB_i$$

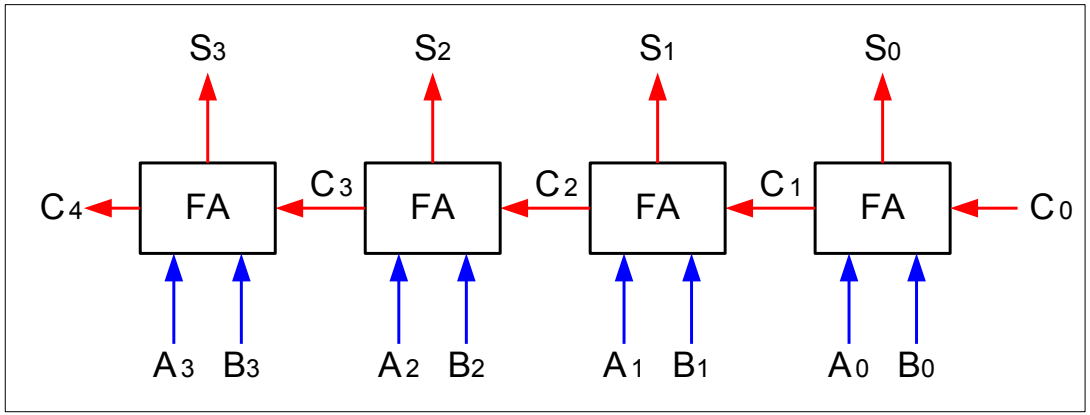
$$= (A_i \oplus B_i)C_{i-1}+A_iB_i$$

n位加法器如何实现？

串行进位加法器

- n位串行进位加法器：分n步实现，每步只求一位和

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



n位串行进位加法器从C_0到C_n的延迟时间为多少？

2n级门延迟！

并行进位加法器

C_{in}	A_i	B_i	C_{out}
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

能否提前获得进位？

逻辑合并

C_{in}	A_i	B_i	C_{out}
X	0	0	0
C_{in}	0	1	C_{in}
C_{in}	1	0	C_{in}
X	1	1	1

如果$A_i=B_i=1$，则$C_{out}=1$与C_{in}无关

如果$A_i=B_i=0$，则$C_{out}=0$与C_{in}无关

如果$A_i+B_i=1$，则C_{out}与C_{in}相同

进位生成函数：$G_i=A_iB_i$

进位传递函数：$P_i=A_i+B_i=A_i \oplus B_i$

传递进位：P_iC_{i-1}

$$C_i=G_i+P_iC_{i-1}$$

$$C_i=A_iB_i+(A_i+B_i)C_{i-1}$$

各进位信号都是独自形成

并行进位(或先行进位)加法器

进位生成函数：$G_i=A_iB_i$

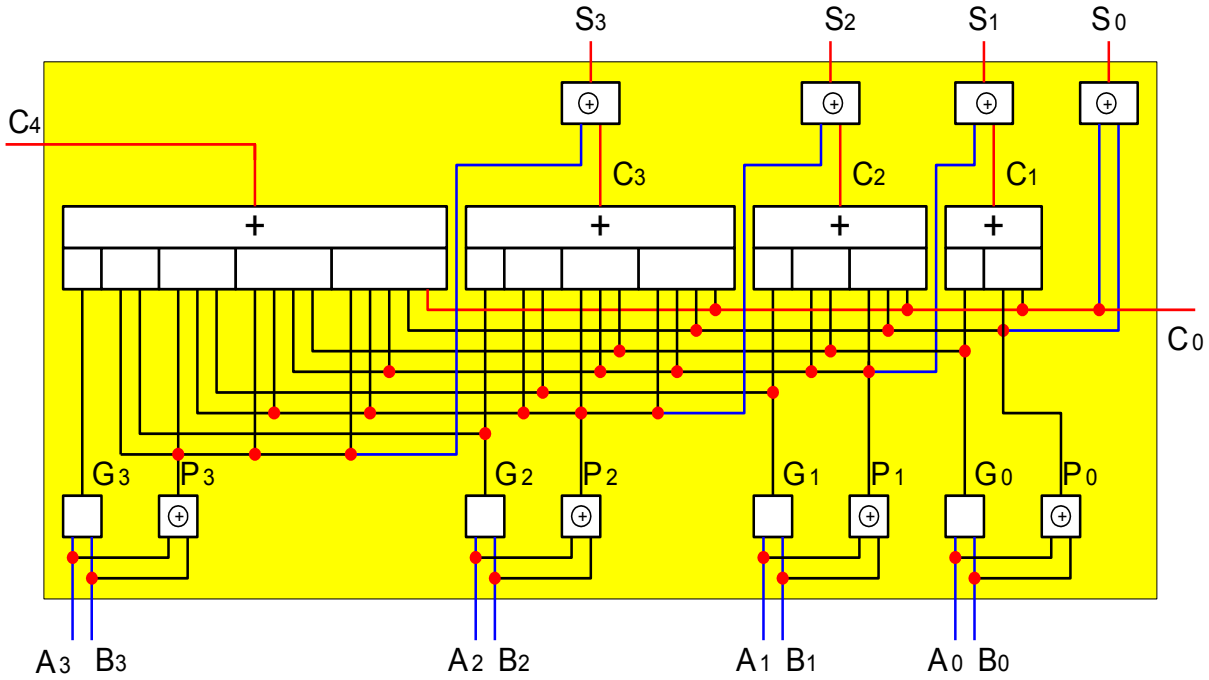
进位传递函数：$P_i=A_i+B_i$

$=A_i \oplus B_i$

传递进位：P_iC_{i-1}

$C_i=G_i+P_iC_{i-1}$

$C_i=A_iB_i+(A_i+B_i)C_{i-1}$

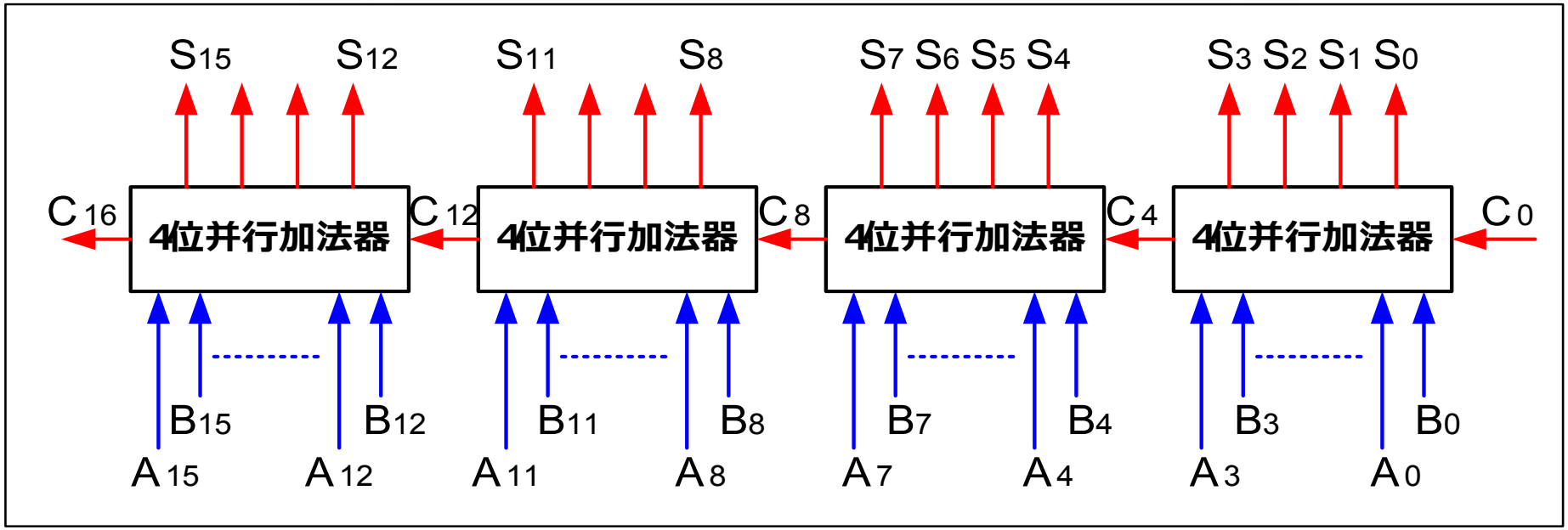


可以实现大规模完全并行进位吗？

并行进位加法器

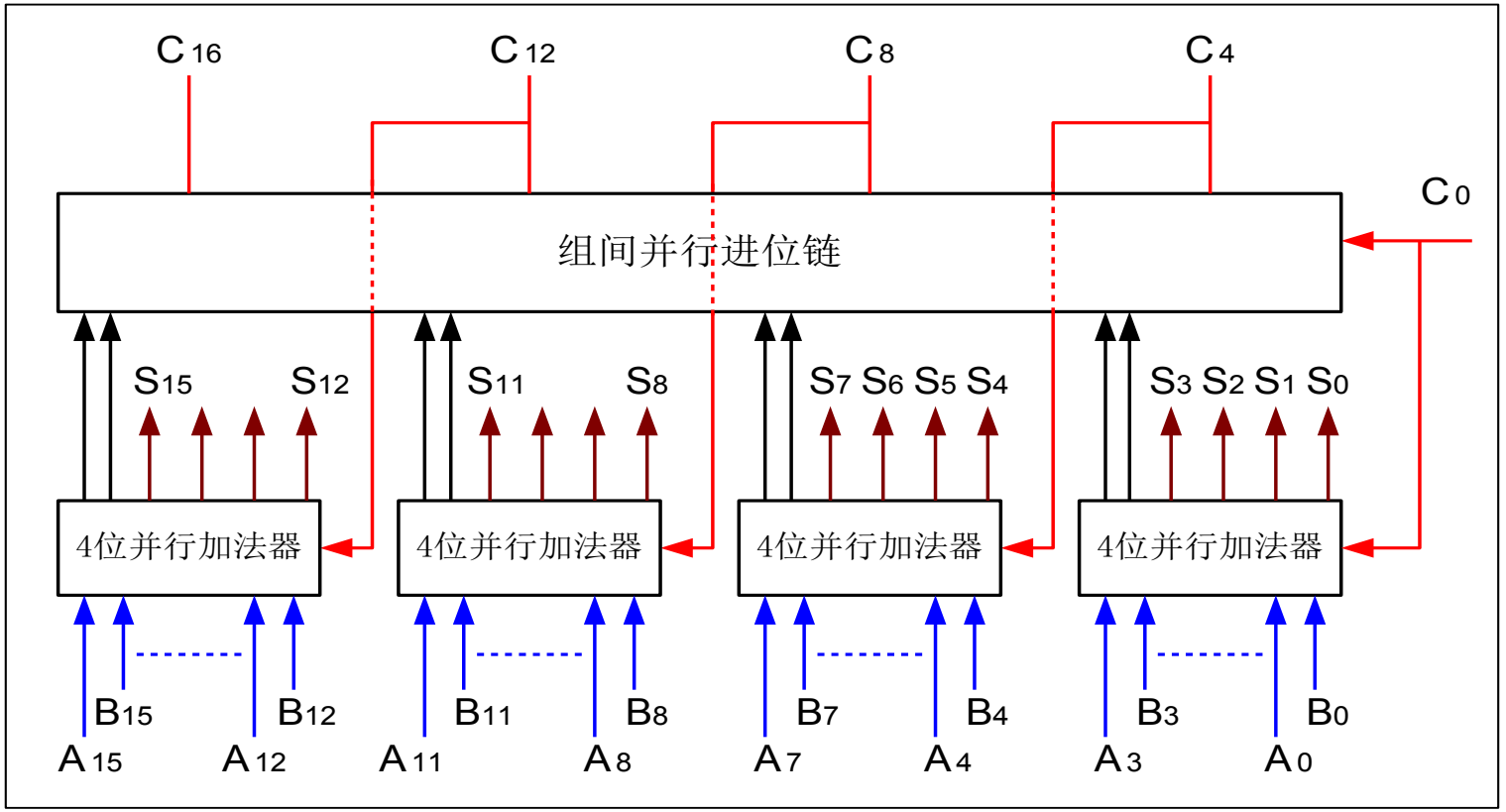
- 完全大规模并行进位实现困难
  - 高位的进位形成逻辑涉及输入变量过多，将受到器件扇入系数的限制
- 位数较多的加法器，常采用分级、分组的进位链结构
  - 分组：组内并行，组间或串或并行(4/8位一组)

分组并行进位加法器



组内并行，组间传递

分组并行进位加法器



组内并行，组间并行