河海大学

第2章 运算方法和运算器

- 1、若x=103，y=-25，则下列表示式采用8位定点补码运算实现时，会发生溢出的是（　　）。

A. x+y

B. -x+y

C. x-y

D. -x-y

[答案]C

- 2、某字长为8位的计算机中，已知整型变量x、y的机器数分别为$[x]_{补}$=1111 0100，$[y]_{补}$=1011 0000。若整型变量z=2*x+y/2，则z的机器数为（　　）。

A. 11000000

B. 00100100

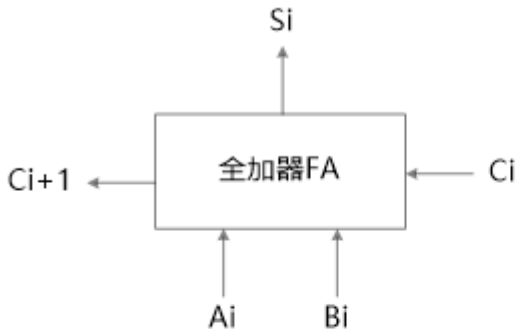
C. 10101010

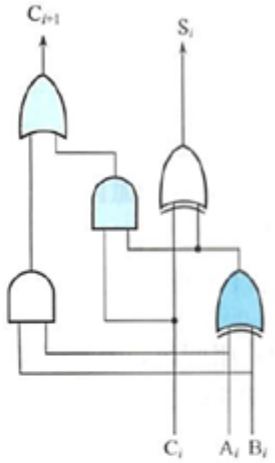
D. 溢出

[答案]A

2.2.4 基本的二进制加法/减法器

- 基本加法单元（全加器）





$S_i = (A_i \oplus B_i) \oplus C_i$ 奇数个1

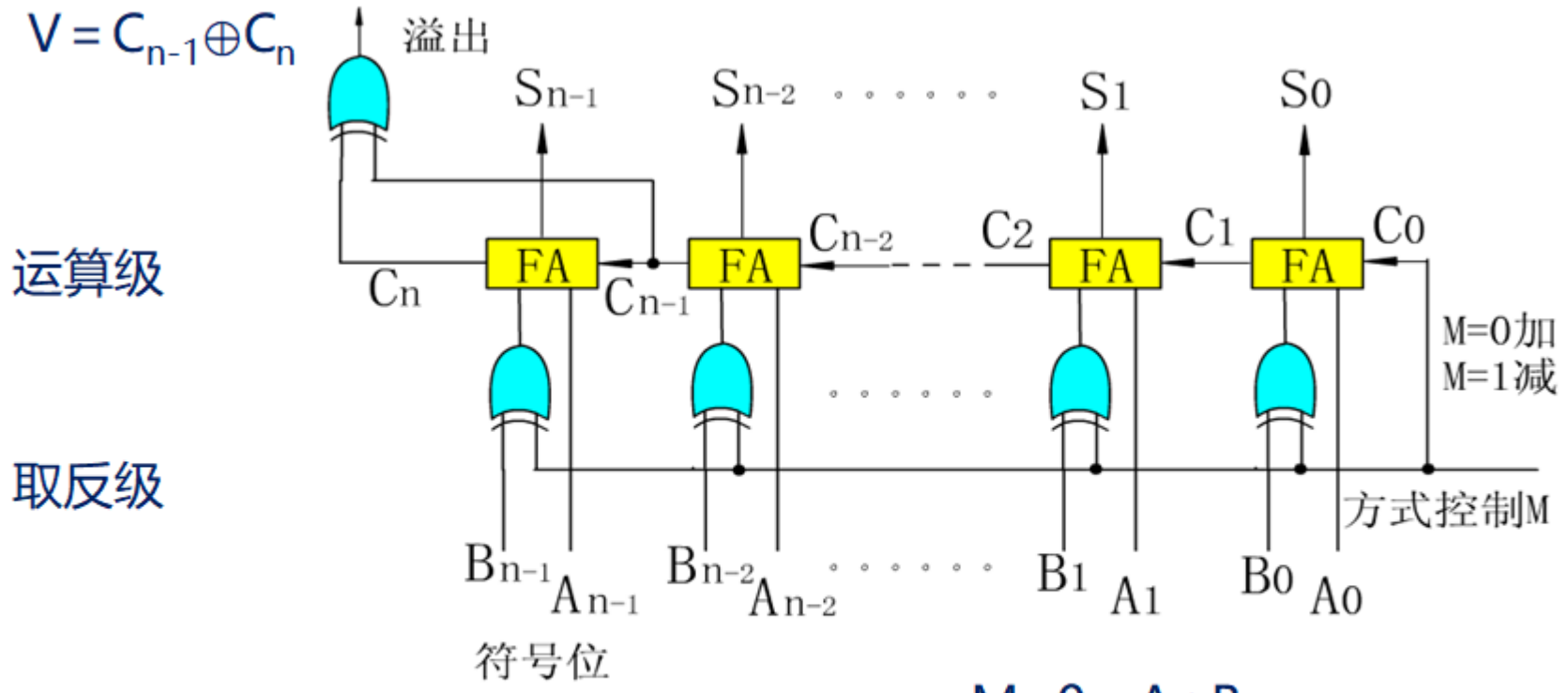
$C_{i+1} = A_iB_i + B_iC_i + C_iA_i$ 两个以上1

$= AiBi + (Ai \oplus Bi)\, Ci$

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

2.2.4 基本的二进制加法/减法器

- 行波（串行）进位的加减法器



- 某加法器进位链小组信号为$C_4C_3C_2C_1$，低位来的信号为C_0，请分别按下述两种方式写出$C_4C_3C_2C_1$的逻辑表达式。

（1） 串行进位方式

（2） 并行进位方式

分析：

- 加法器的进位链的基本逻辑关系

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

- 令 $G_i = A_iB_i$ 为**进位产生函数（本地进位）**

  $P_i = A_i \oplus B_i$ 为**进位传递函数**

  P_iC_i 为**传送进位**

  于是 $C_{i+1} = G_i + P_iC_i$

- 只有$A_i = B_i = 1$时，本位才向高位进位
- 当$A_i \neq B_i$时，低一位的进位将向更高位传送
- 传送进位和本地进位不可能同时为1

4位先行进位（超前进位）方式

$$C_1 = G_0 + P_0C_0$$
$$C_2 = G_1 + P_1G_0 + P_1P_0C_0$$
$$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$
$$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$$

CLA：Carry Look Ahead

2.3 定点乘法运算

1. 完全软件实现

   不设乘除法运算的硬件电路，而是由软件利用运算器中的加法和移位操作编程进行乘除法运算

2. 加法器增加硬件辅助电路实现

   利用运算器中的加法器硬件电路和移位电路，再设计必要的扩展电路，用硬件通过加法和移位操作实现乘除法运算

3. 专用乘除法器实现

   在运算器中除了设置加法器之外，再增加硬件电路设置高速乘除法部件，直接完成乘除法运算

原码一位乘法运算

令$[X]_{原} = X_f.X_1\ X_2\ X_3\cdots\cdots X_n$

$[Y]_{原} = Y_f.Y_1\ Y_2\ Y_3\cdots\cdots Y_n$

则

$$[X\times Y]_{原} = (X_f \oplus Y_f) + (|X|\times|Y|)$$

$$= (X_f \oplus Y_f) + (0.X_1X_2\cdots X_n \times 0.Y_1Y_2\cdots Y_n)$$

积的符号：被乘数与乘数二符号的异或值

积的数值：被乘数与乘数二数的绝对值之积

2.3 定点乘法运算

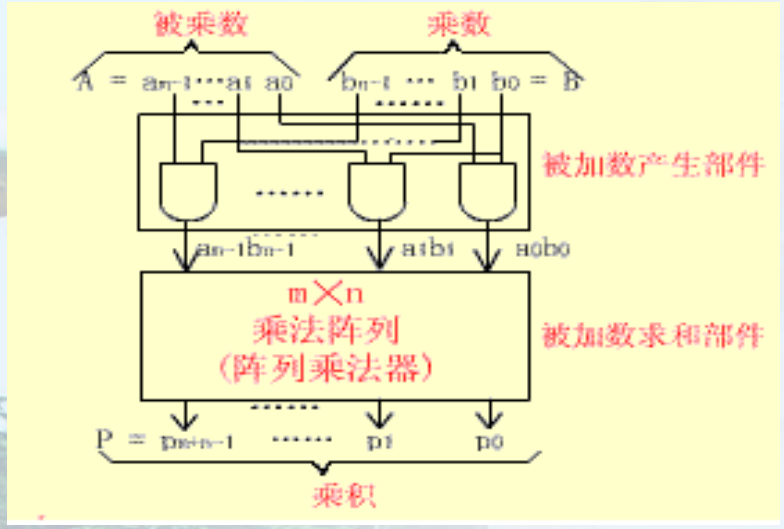
- 人工算法
  - 被乘数 $[x]_{原}=x_f \quad x_{n-1}\cdots x_1x_0$
  - 乘数 $[y]_{原}=y_f \quad y_{n-1}\cdots y_1y_0$
  - 乘积 $[z]_{原}=(x_f \oplus y_f) + (x_{n-1}\cdots x_1x_0) \times (y_{n-1}\cdots y_1y_0)$

```
                1 1 0 1 (x)
×               1 0 1 1 (y)
-----------------------------
                1 1 0 1
              1 1 0 1
            0 0 0 0
+       1 1 0 1                .
-----------------------------
    1 0 0 0 1 1 1 1 (z)
```

2.3 定点乘法运算

- 不带符号的阵列乘法器

$$
\begin{array}{rccccccc}
 & a_{m-1} & a_{m-2} & \cdots & a_1 & a_0 & = A \\
\times) & & b_{n-1} & \cdots & b_1 & b_0 & = B \\
\hline
 & a_{m-1}b_0 & a_{m-2}b_0 & \cdots & a_1b_0 & a_0b_0 & \\
 & a_{m-1}b_1 & a_{m-2}b_1 & \cdots & a_1b_1 & a_0b_1 & \\
 & \vdots & \vdots & & \vdots & \vdots & \\
+) & a_{m-1}b_{n-1} & a_{m-2}b_{n-1} & \cdots & a_1b_{n-1} & a_0b_{n-1} & \\
\hline
p_{m+n-1} & p_{m+n-2} & p_{m+n-3} & \cdots & p_{n-1} & \cdots & p_1 \quad p_0 = P
\end{array}
$$



- 可以用m×n个与门一次性并行（同时）产生所有的a_ib_j，关键是如何累加？

2.3 定点乘法运算

- 不带符号的阵列乘法器

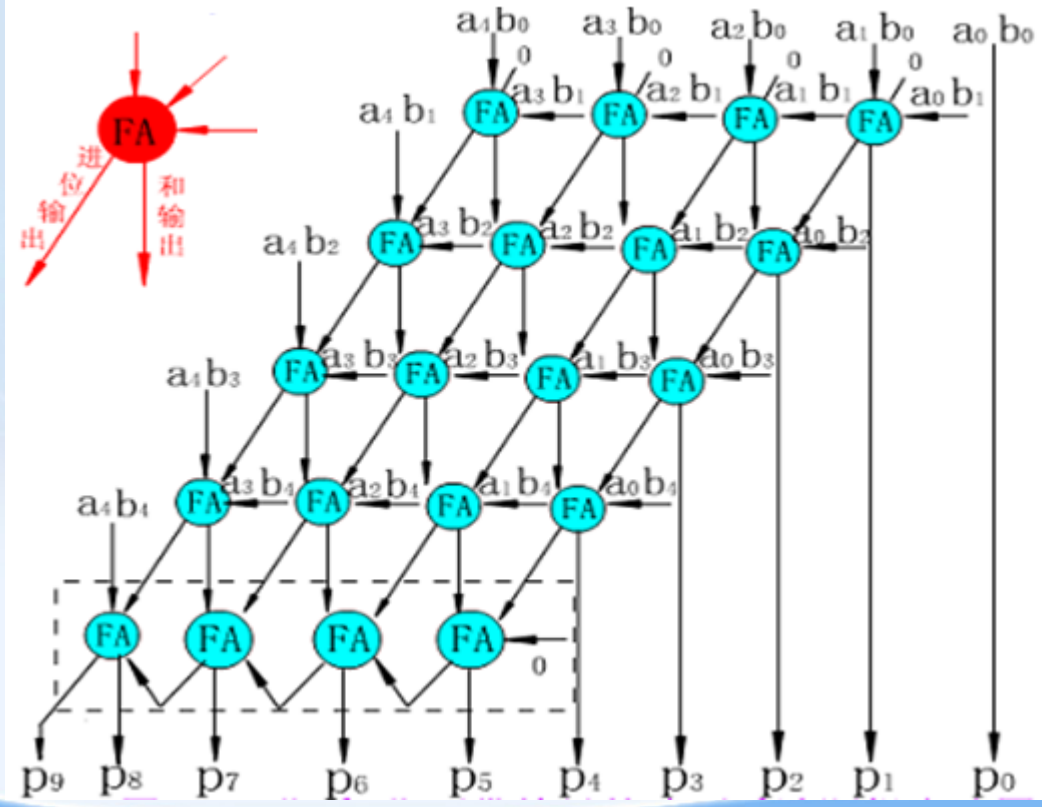
5bits * 5bits的乘法过程描述（A4A3A2A1A0 * B4B3B2B1B0）：

					A4B0	A3B0	A2B0	A1B0	A0B0
				A4B1	A3B1	A2B1	A1B1	A0B1	
			A4B2	A3B2	A2B2	A1B2	A0B2		
		A4B3	A3B3	A2B3	A1B3	A0B3			
	A4B4	A3B4	A2B4	A1B4	A0B4				
+	进位	进位	进位	进位	进位				
	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

可以这样操作：第1行+第2行、然后加第3行、加第4行、加第5行、加进位行。

2.3 定点乘法运算

- 不带符号的阵列乘法器



第0步：5X5个与门同时工作求a_ib_j；

第1步：第1排4个全加器同时工作，初始进位都为0，全加器的进位到第2排；

第2步：第2排4个全加器同时工作，全加器的进位到第3排；

第3步：第3排4个全加器同时工作，全加器的进位到第4排；

第4步：第4排4个全加器同时工作，全加器的进位到第5排；

第5步：第5排全加器串行工作，其中第1个的加数为0，串行传递进位。

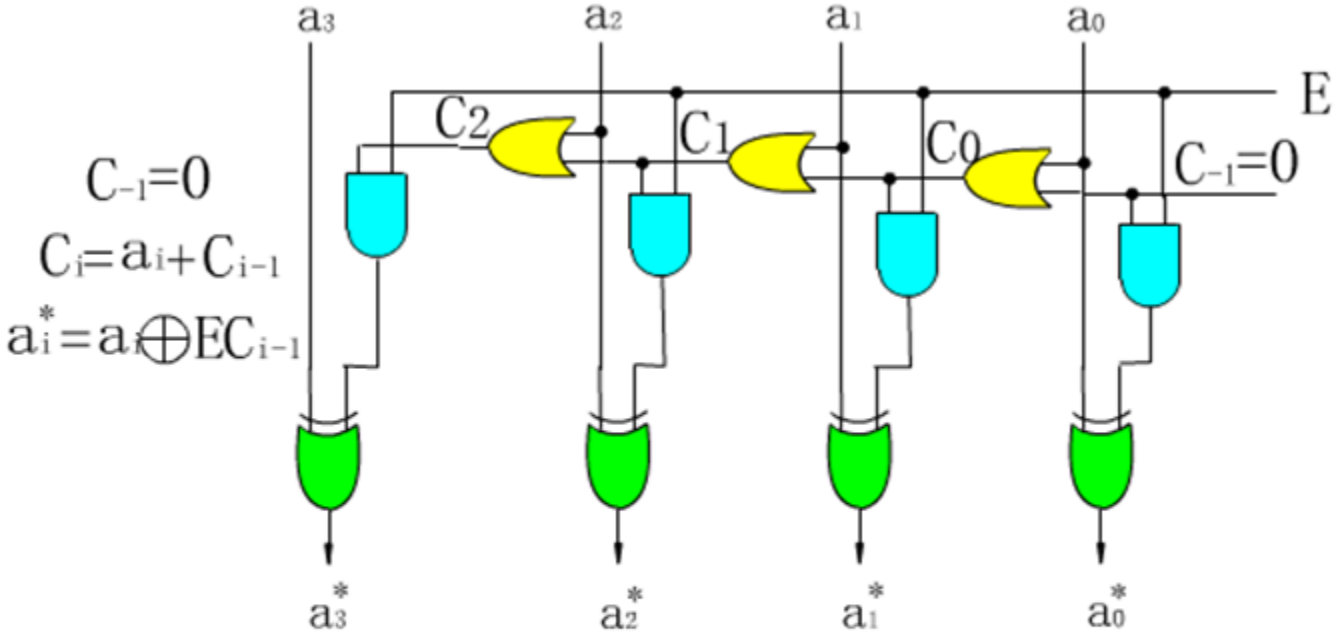
不带符号的阵列乘法器

2.3 定点乘法运算

- (n+1)×(n+1)带符号的原码阵列乘法器
  - 符号位进行异或运算，就是乘积的符号位
  - 尾数部分用“n × n不带符号的阵列乘法器”实现

2.3 定点乘法运算

- 求补电路



从低位往高位找到第一个1，该1通过或门往高位传递，按位取反。

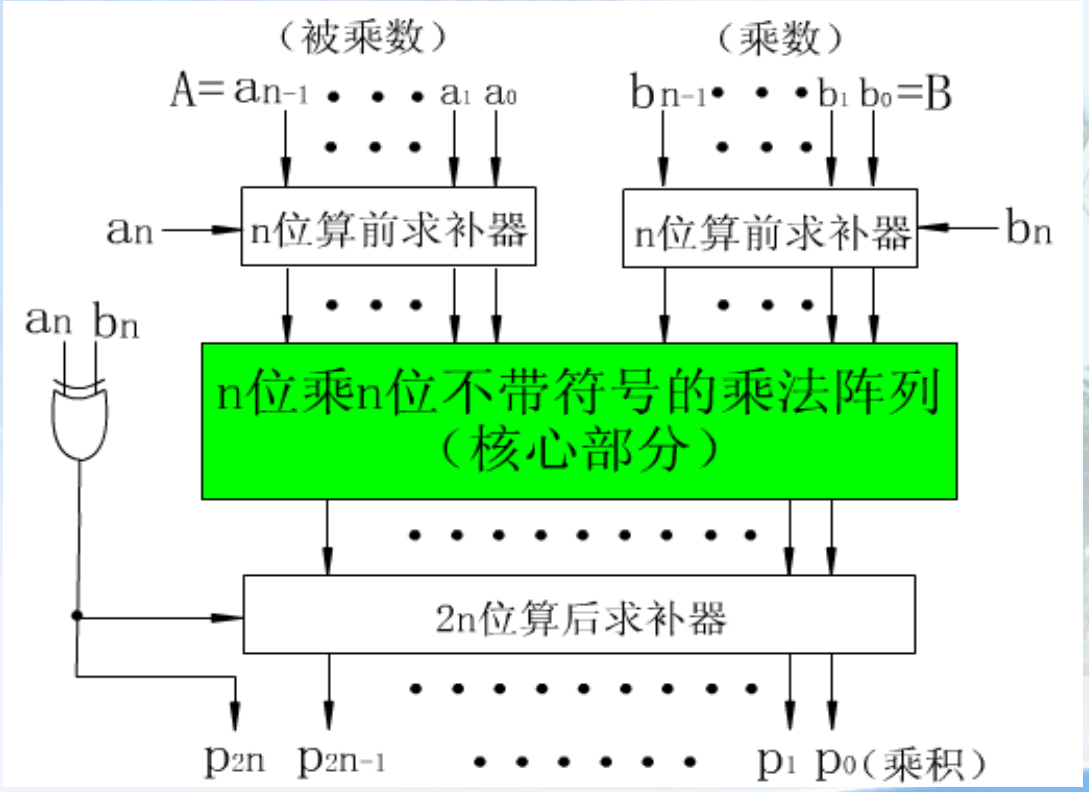
E：数的符号位；

E=0正数，所有位原样输出，$Ea_3a_2a_1a_0$= 0 1010，则$a_3{*}a_2{*}a_1{*}a_0{*}$= 1010

E=1负数，求补， Ea3a2a1a0= 1 1010，则a3*a2*a1*a0*= 0110

2.3 定点乘法运算

- (n+1)×(n+1)带符号的补码阵列乘法器



被乘数、乘数的补码尾数转换为原码尾数

原码乘积转换为补码尾数。

[例23] 设x=+15，y=-13，用带求补器的原码阵列乘法器求出乘积x·y=?

解：$[x]_{原}$=01111，$[y]_{原}$=11101，|x|=1111，|y|=1101

符号位运算：0⊕1=1

```
          1 1 1 1
×         1 1 0 1
-------------------
          1 1 1 1
        0 0 0 0
      1 1 1 1
+   1 1 1 1
-------------------
  1 1 0 0 0 0 1 1
```

被乘数和乘数都是原码时：求补操作不执行，只将去掉符号的数值部分原样输出。

乘积符号为1，算后求补器输出11000011，

$[x \times y]_{原}$=111000011

换算成二进制数真值是 $x \cdot y = (-1100011)_2 = (-195)_{10}$

[例24] 设x=-15，y=-13，用带求补器的补码阵列乘法器求出乘积x·y=? 并用十进制数乘法进行验证。

解：$[x]_{补}$=10001，$[y]_{补}$=10011，乘积符号位运算：1⊕1=0

尾数部分算前求补器输出 |x|=1111，|y|=1101

```
          1 1 1 1
×         1 1 0 1
-------------------
          1 1 1 1
        0 0 0 0
      1 1 1 1
+   1 1 1 1
-------------------
  1 1 0 0 0 0 1 1
```

被乘数和乘数都是负数补码时：去掉符号位，其余各位求反加1，即完成求补过程得到原码。

乘积符号为0，算后求补器输出11000011，$[x \times y]_{补}$=011000011

补码二进制数真值 $x \cdot y = 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^1 + 1 \times 2^0 = (+195)_{10}$

十进制数乘法验证 x·y = (-15)×(-13) = +195

河海大学

2.4 定点除法运算

- 原码除法算法原理(2.4.1)
- 并行除法器(2.4.2)

2.4.1 原码除法算法原理

1. 分析笔算除法

$x = -0.1011$　$y = 0.1101$　求 $x \div y$

```
                0.1101
          ------------
0.1101 ) 0.10110
         0.01101
         -------
         0.010010
         0.001101
         --------
         0.0001010 0
         0.0000110 1
         -----------
         0.0000011 1
```

✓商符单独处理

？心算上商

？余数不动低位补“0”减右移一位的除数

$x \div y = -0.1101$　商符心算求得

余数 -0.0000111

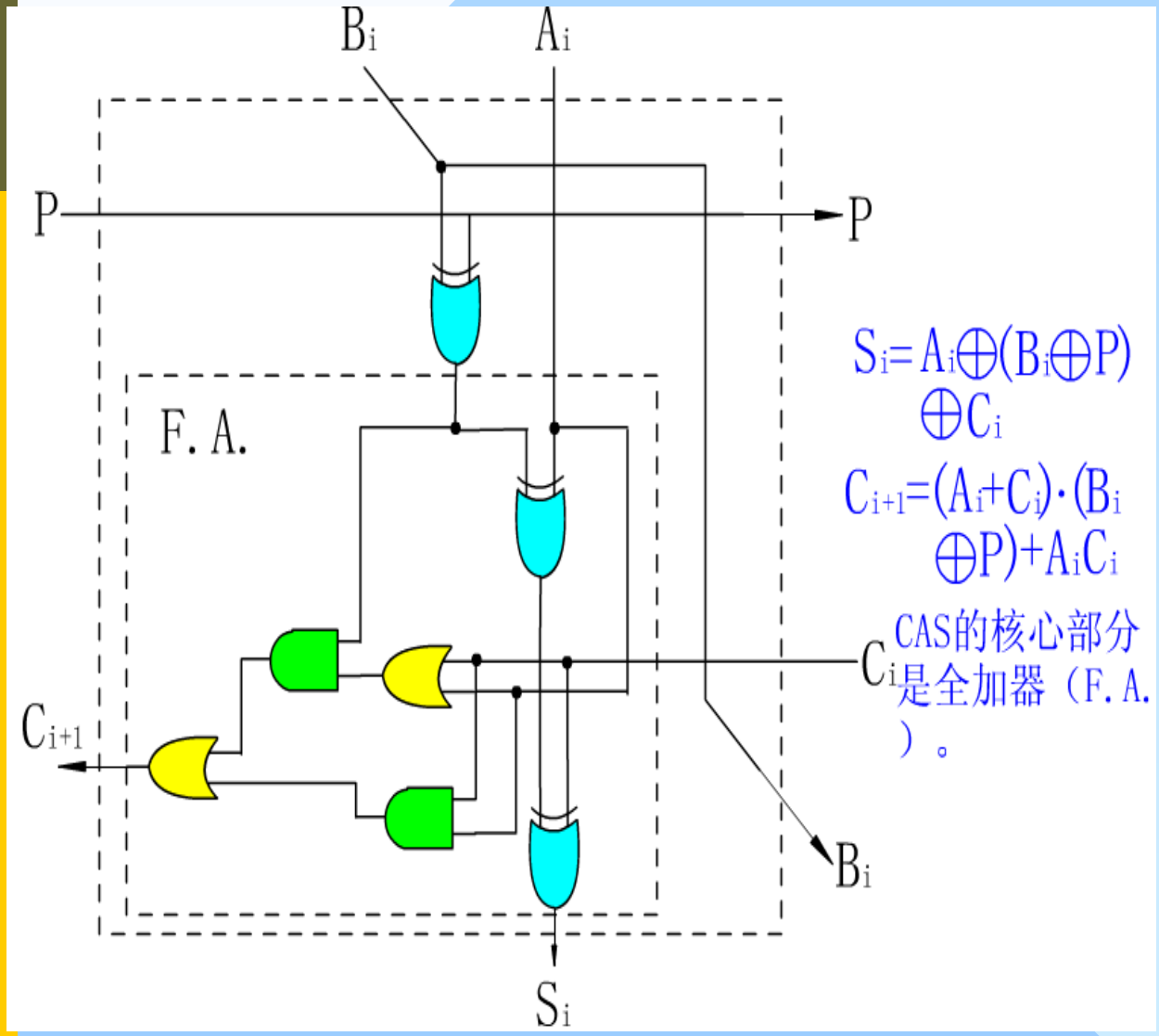
2、不恢复余数的除法

- 人工除法时，人可以比较被除数（余数）和除数的大小来确定商1（够减）或商0（不够减）
- 机器除法时，余数为正表示够减，余数为负表示不够减。不够减时必须恢复原来余数，才能继续向下运算。这种方法叫恢复余数法，控制比较复杂。
- 不恢复余数法（加减交替法）
  余数为正，商1，下次除数右移（或部分余数左移）做减法；
  余数为负，商0，下次除数右移（或部分余数左移）做加法。
  控制简单，有规律。

2.4.2 并行除法器

- 和阵列乘法器非常相似,阵列式除法器也是一种并行运算部件,采用大规模集成电路制造.与早期的串行除法器相比,阵列除法器不仅所需的控制线路少,而且能提供令人满意的高速运算速度。
  - 1.可控加法/减法(CAS)单元
  - 2.不恢复余数的阵列除法器

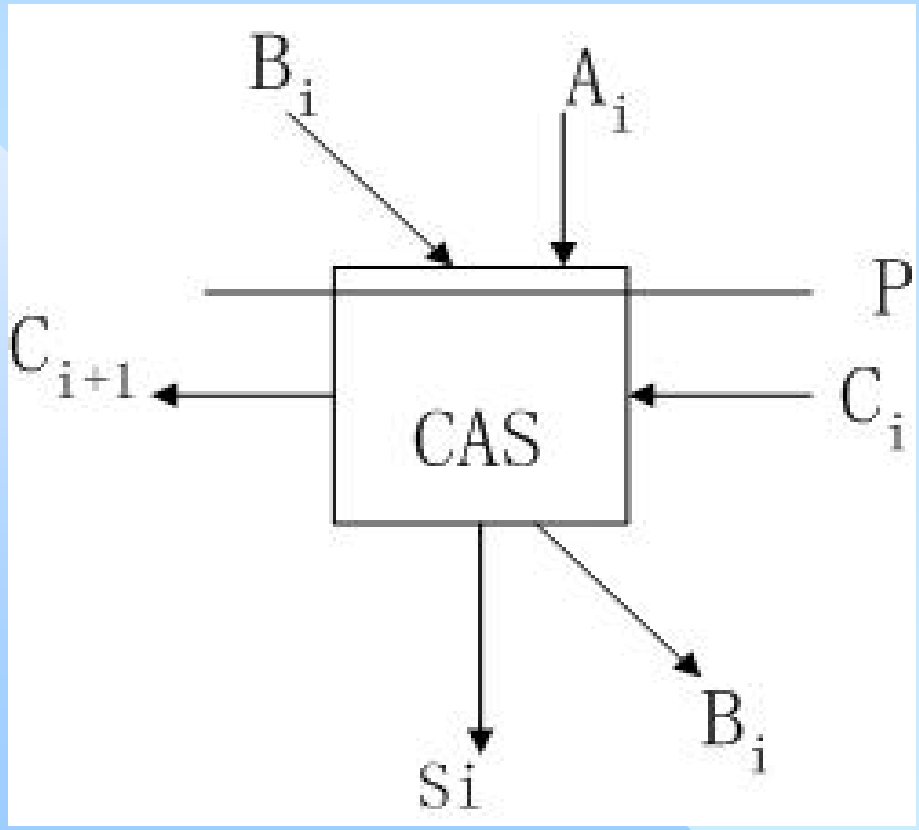
1.可控加法/减法(CAS)单元



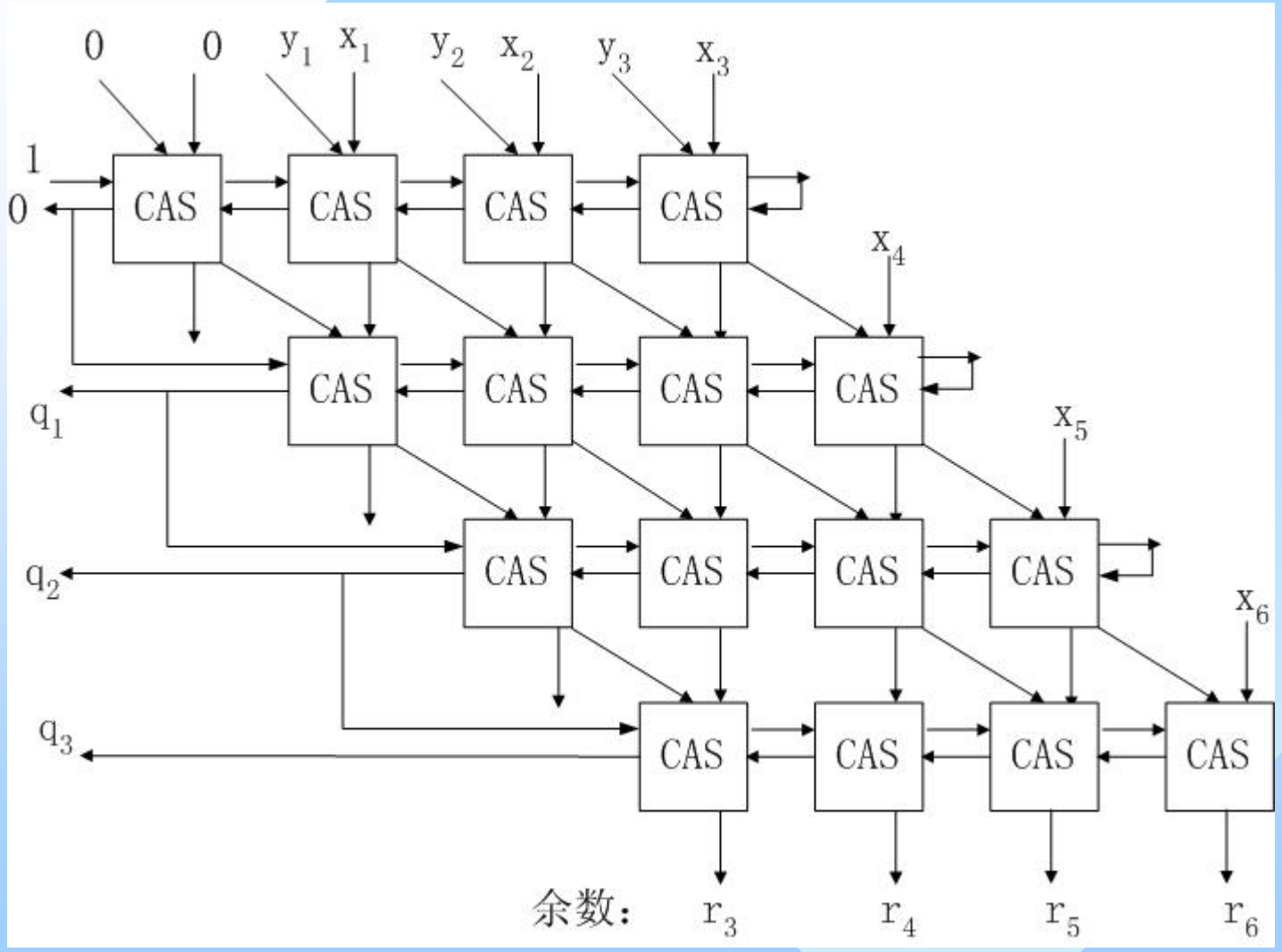
当输入线P＝0时,CAS作加法运算;

当P＝1时,CAS作减法运算。

–四个输入端、四个输出端



2.不恢复余数的阵列除法器



上级商，作为下一级的P（控制端）输入。

2.不恢复余数的阵列除法器

- 在不恢复余数的除法阵列中,每一行所执行的操作究竟是加法还是减法,取决于前一行输出的符号与被除数的符号是否一致。
  - 当出现不够减时,部分余数相对于被除数来说要改变符号。这时应该产生一个**商位“0”,除数首先沿对角线右移**,然后加到下一行的部分余数上。
  - 当部分余数不改变它的符号时,即产生**商位“1”,下一行的操作应该是减法**。

2.不恢复余数的阵列除法器

- 注意：
  - 最后如果余数为负，需校正
  - 余数符号和被除数相同

举例

- $x=0.101001$, $y=0.111$, 求 $x \div y$。

2.4.2 并行除法器

[例] $x=0.101001$，$y=0.111$，求 $x \div y$。

[解:] $[|x|]_{补}$=0.101001 , $[|y|]_{补}$=0.111 , $[-|y|]_{补}$=1.001

	0.1 0 1 0 0 1		;被除数
＋$[-y]_{补}$	1.0 0 1		;第一步减除数y
	1.1 1 0 0 0 1	<0 q_0=0	;余数为负,商0
＋$[y]_{补}\rightarrow$	0.0 1 1 1		;除数右移1位加
	0.0 0 1 1 0 1	>0 q_1=1	;余数为正,商1
＋$[-y]_{补}\rightarrow$	1.1 1 0 0 1		;除数右移2位减
	1.1 1 1 1 1 1	<0 q_2=0	;余数为负,商0
＋$[y]_{补}\rightarrow$	0.0 0 0 1 1 1		;除数右移3位加
	0.0 0 0 1 1 0	>0 q_3=1	;余数为正,商1

商$q=q_0.q_1q_2q_3$=0.101 , 余数$r=(0.00r_3r_4r_5r_6)$=0.000110

河海大学

2.5 定点运算器的组成

- 逻辑运算（2.5.1）
- 多功能算术/逻辑运算单元（2.5.2）
- 内部总线（2.5.3）
- 定点运算器的基本结构（2.5.4）

2.5.1 逻辑运算

- 逻辑运算指令
  - 与运算、或运算、非运算、异或运算
- 逻辑运算特点
  - 按位运算
  - 没有进位概念

2.5.1 逻辑运算

- 逻辑非运算

【例】x_1=0100 1011，x_2=1111 0000，求$\overline{x}_1$，$\overline{x}_2$

【解】非运算

$$\overline{x_1} = 1011\ \ 0100$$

$$\overline{x_2} = 0000\ \ 1111$$

2.5.1 逻辑运算

- 逻辑或运算

【例】 x =1010 0001， y =1001 1011,求 x + y

【解】或运算

$$\begin{array}{r} 1010\ \ 0001 \\ +\ \ 1001\ \ 1011 \\ \hline 1011\ \ 1011 \end{array}$$

2.5.1 逻辑运算

- 逻辑与运算

【例】 x =1011 1001， y =1111 0011，求x . y

【解】与运算

```
  1011  1001
. 1111  0011
-------------
  1011  0001
```

2.5.1 逻辑运算

- 逻辑异或运算

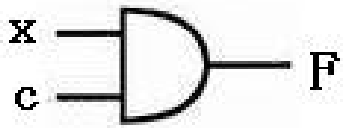
【例】 x =1010 1011， y =1100 1100，求 x ⊕ y

【解】异或运算

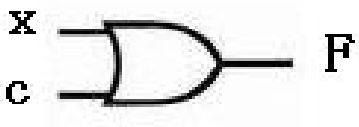
$$\begin{array}{rr} & 1010\ \ 1011 \\ \oplus & 1100\ \ 1100 \\ \hline & 0110\ \ 0111 \end{array}$$

2.5.1 逻辑运算

1011 and 0000 = 0000	1011 or 0000 = 1011	1011 xor 0000 = 1011
1011 and 1111 = 1011	1011 or 1111 = 1111	1011 xor 1111 = 0100
全0屏蔽，全1保留。	全1屏蔽，全0保留。	遇0保留，遇1求反。



c=1　F=x



c=0　F=x

c=1　$F=\overline{x}$

c=0　F=x

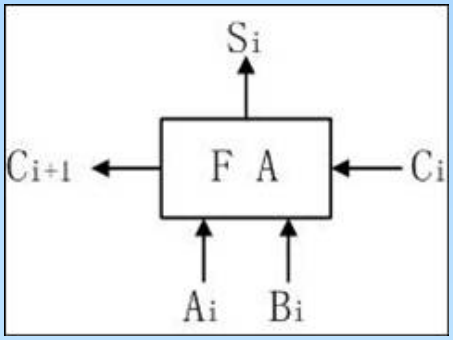
2.5.2 多功能算术/逻辑运算单元(ALU)

- 全加器(FA)构成的行波进位加法器,它可以实现补码数的加法运算和减法运算。但是这种加法/减法器存在两个问题:
  - 一是由于串行进位,它的运算时间很长。
  - 二是就行波进位加法器本身来说,它只能完成加法和减法两种操作而不能完成逻辑操作。

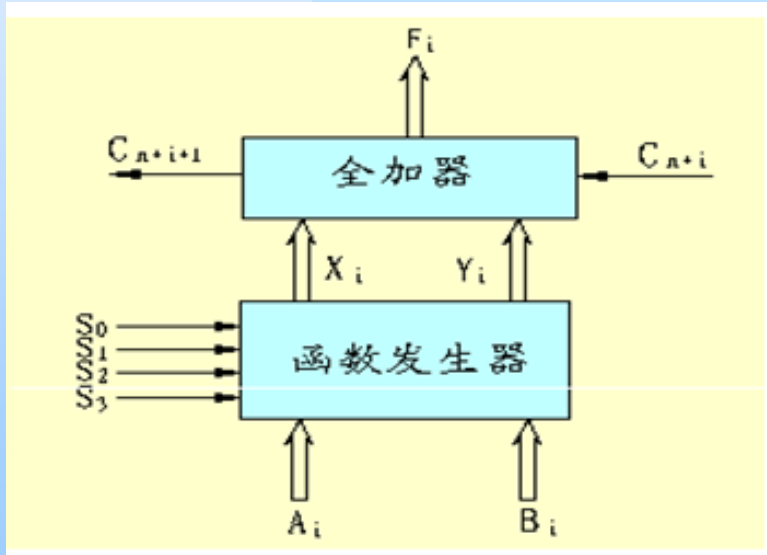
- 创新点
  - 实现并行进位(先行进位)
  - 实现16种算术运算，16种逻辑运算

1.基本思想

- 一位全加器(FA)的逻辑表达式为
  - $F_i = A_i \oplus B_i \oplus C_i$
  - $C_{i+1} = A_iB_i + B_iC_i + C_iA_i$



- 改进思路



n: 片组号

i: 片组内第i位

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

$C_{n+i+1} = X_iY_i + Y_iC_{n+i} + X_iC_{n+i}$

Xi=f(Ai,Bi,s0,s1,s2,s3)

Yi=f(Ai,Bi,s0,s1,s2,s3)

4位一片，i=0,1,2,3

2.5.2 多功能算术/逻辑运算单元

- 逻辑表达式

X_i, Y_i与控制参数和输入量的关系

S_0 S_1	Y_i	S_2 S_3	X_i
0 0	$\overline{A}i$	0 0	1
0 1	$\overline{A}_iB_i$	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	$\overline{A}_i$

$$X_i=\overline{S}_2\overline{S}_3+\overline{S}_2S_3\ (\overline{A}_i+\overline{B}_i)\ +S_2\overline{S}_3\ (\overline{A}_i+B_i)\ +S_2S_3\overline{A}_i$$

$$Y_i=\overline{S}_0\overline{S}_1\overline{A}_i+\overline{S}_0S_1\overline{A}_iB_i+S_0\overline{S}_1\overline{A}_i\overline{B}_i$$

$$X_i=\overline{S_3A_iB_i+S_2A_i\overline{B}_i}$$

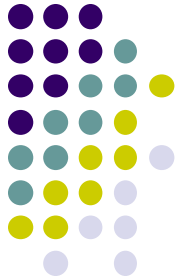
$$Y_i=\overline{A_i+S_0B_i+S_1\overline{B}_i}$$

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

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

Xi、Yi只与Si、Ai、Bi有关，可以根据A、B、S直接求得。

2.5.2 多功能算术/逻辑运算单元ALU



- 如何实现先行进位？

答：由于每一位中X、Y的产生是同时的，则可以由下面方法算出并行进位的C_{n+4}

$$C_{n+1}=Y_0+X_0C_n$$

$$C_{n+2}=Y_1+X_1C_{n+1}=Y_1+Y_0X_1+X_0X_1C_n$$

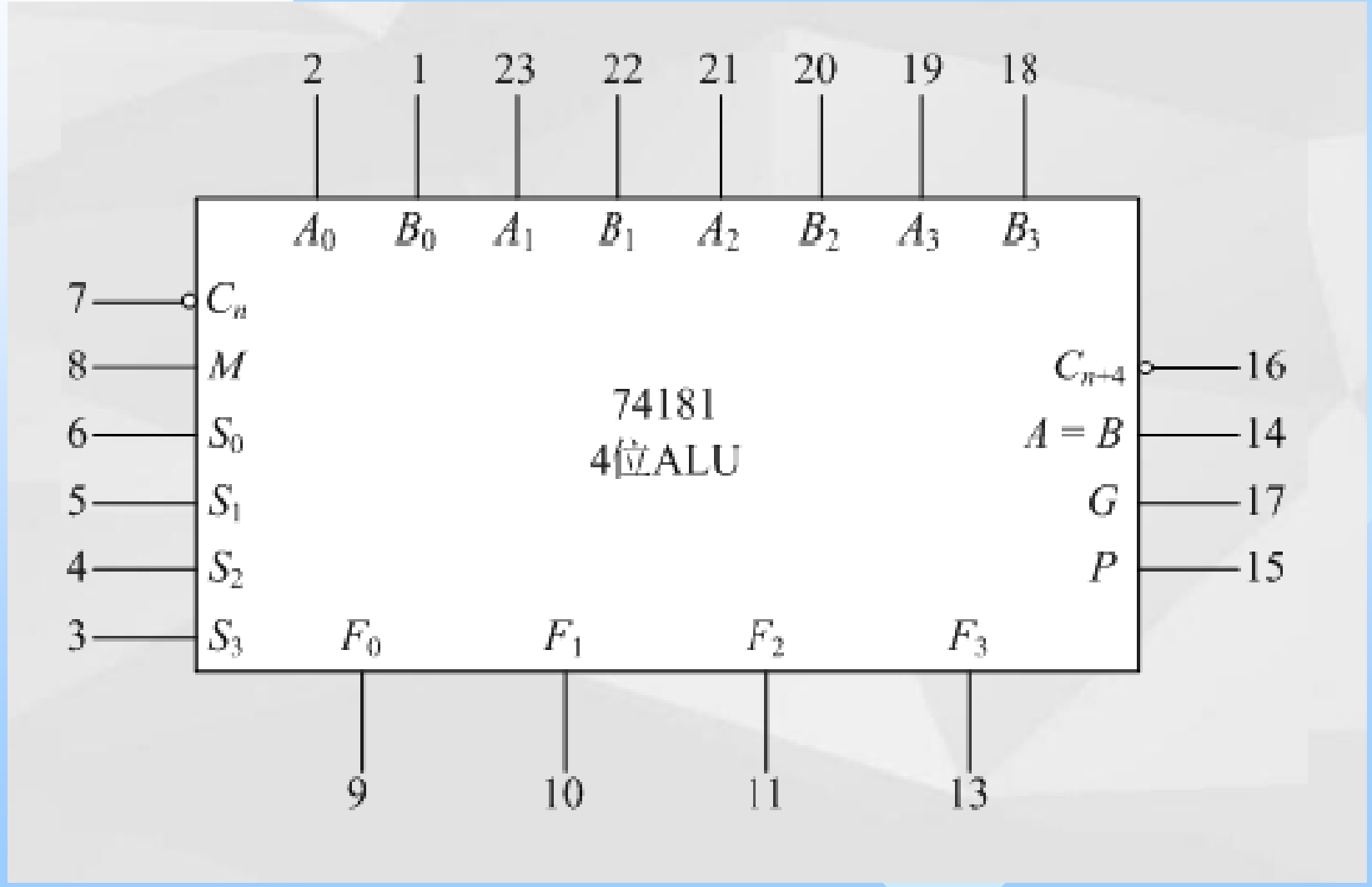
$$C_{n+3}=Y_2+X_2C_{n+2}=Y_2+Y_1X_1+Y_0X_1X_2+X_0X_1X_2C_n$$

$$C_{n+4}=Y_3+X_3C_{n+3}=\underline{Y_3+Y_2X_3+Y_1X_2X_3+Y_0X_1X_2X_3}+\underline{X_0X_1X_2X_3}C_n$$

G进位发生输出　　P进位传送输出

- 器件： 74181

74181芯片

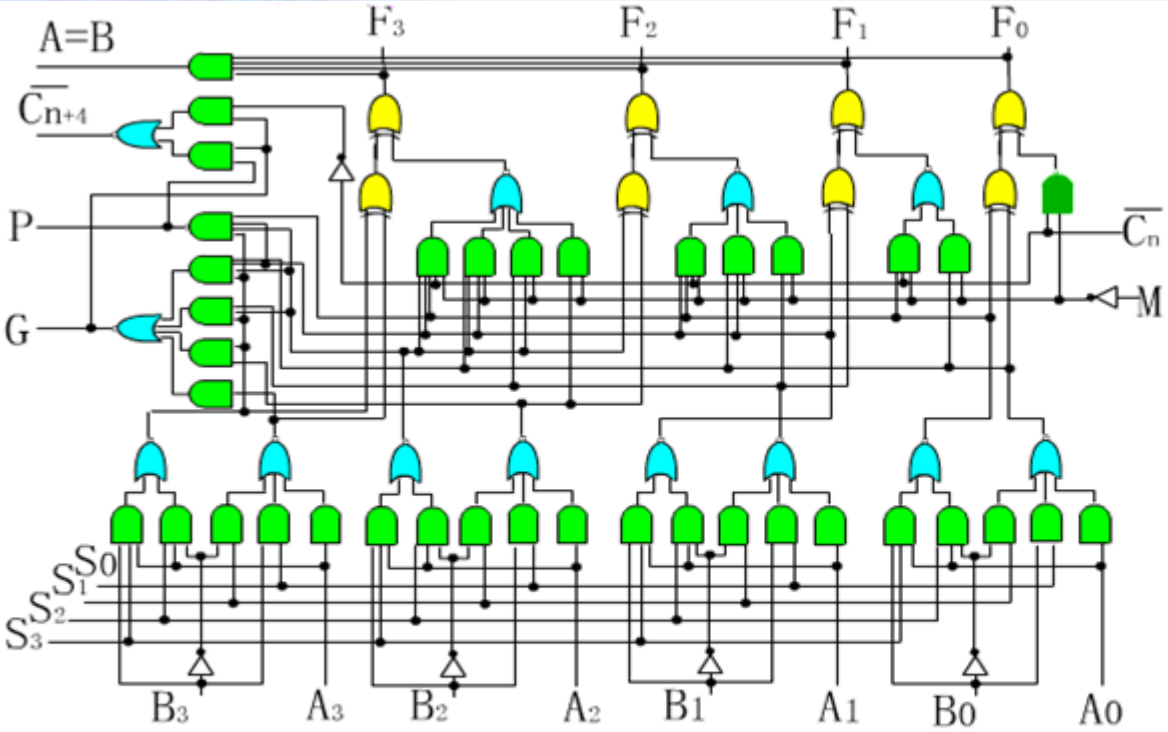

2
1
23
22
21
20
19
18
A_0
B_0
A_1
B_1
A_2
B_2
A_3
B_3
7
C_n
8
M
6
S_0
5
S_1
4
S_2
3
S_3
74181
4位ALU
C_{n+4}
16
$A = B$
14
G
17
P
15
F_0
F_1
F_2
F_3
9
10
11
13

74181ALU功能表

操作选择	运 算 功 能		
S3S2S1S0	M=H 逻辑功能	M=L 算术功能	
		Cn'=H(无进位)	Cn=L(有进位)
0 0 0 0	F=A'	F=A	F=A加1
0 0 0 1	F=(A+B)'	F=A+B	F=(A+B)加1
0 0 1 0	F=A'B	F=A+B'	F=(A+B')加1
0 0 1 1	F=0	F=减 1(2的补码)	F=0
0 1 0 0	F=(AB)'	F=A加AB'	F=A加AB'加1
0 1 0 1	F=B'	F=(A+B)加AB'	F=(A+B)加AB'加1
0 1 1 0	F=A⊕B	F=A减B减1	F= A减B
0 1 1 1	F=AB'	F=AB'减1	F= AB'
1 0 0 0	F=A'+B	F=A加AB	F=A加AB加1
1 0 0 1	F=A⊙B	F=A加B	F=A加B加1
1 0 1 0	F=B	F=(A+B')加AB	F=(A+B')加AB'加1
1 0 1 1	F=AB	F=AB减1	F=AB
1 1 0 0	F=1	F=A加A	F=A加A加1
1 1 0 1	F=A+B'	F=(A+B)加A	F=(A+B)加A加1
1 1 1 0	F=A+B	F=(A+B')加A	F=(A+B')加A加1
1 1 1 1	F=A	F=A减1	F=A

2.5.2 多功能算术/逻辑运算单元

- 算术/逻辑运算单元74181ALU

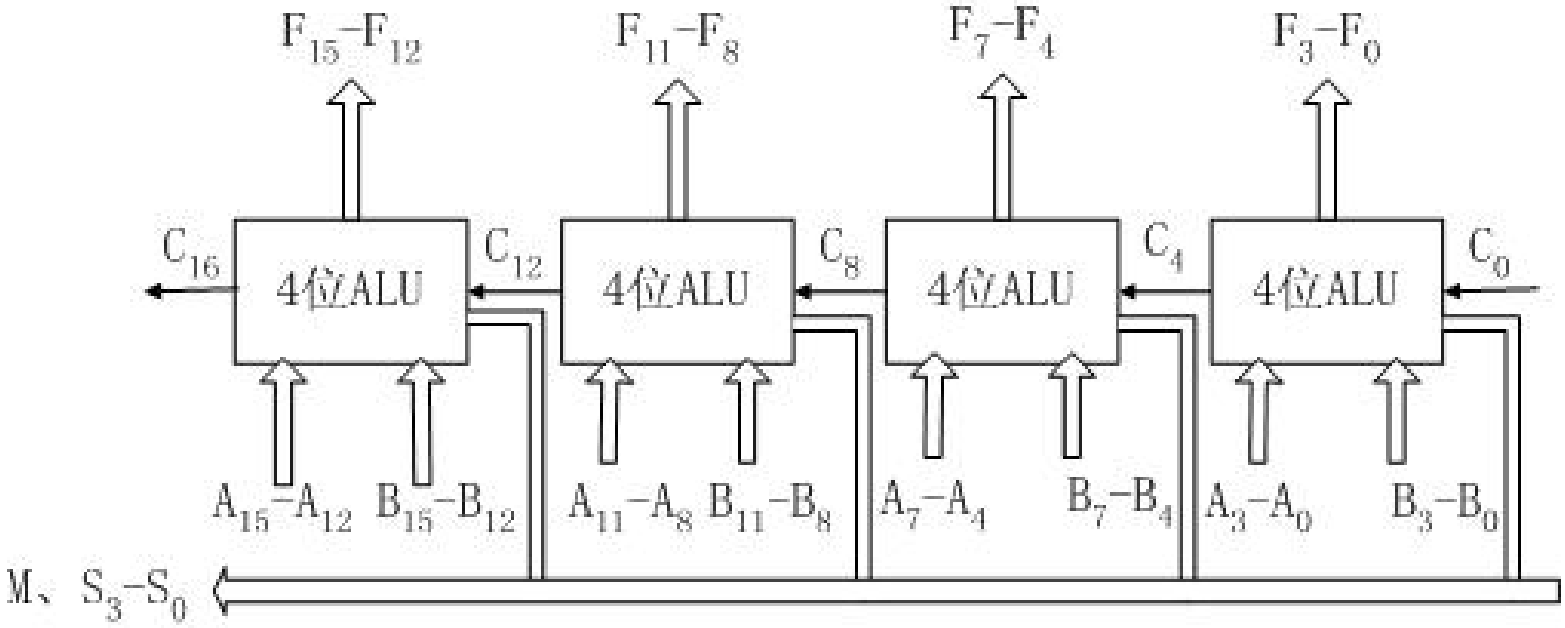


M=1，Fi=Xi ⊕Yi，屏蔽了进位，可变换为各种逻辑运算。

M=0，Fi=Xi ⊕Yi ⊕Ci，可变换为各种算术运算。

74181ALU

- 设计16位ALU(组间行波进位)

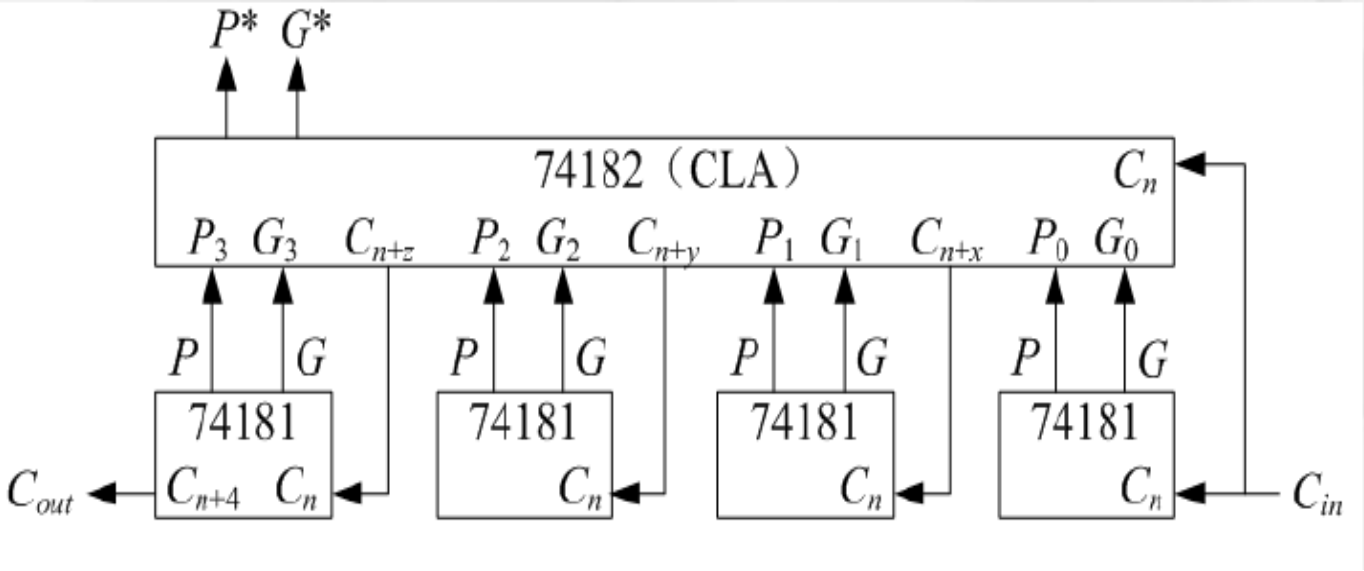


4. 两级先行进位的ALU

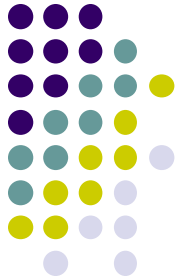
- 74181ALU设置了P和G两个本组先行进位输出端。
- 如果将四片74181的P,G输出端送入到74182先行进位部件（CLA）,又可实现第二级的先行进位,即组与组之间的先行进位。
- 74181的先行进位输出依次P_0,G_0,P_1,G_1,P_2,G_2,P_3,G_3, 先行进位部件74182CLA

74182芯片

74182芯片是专门与74181ALU芯片配套使用的先行进位部件(CLA)，可以为4个74181ALU芯片提供芯片间的先行进位支持，从而构成一个16位全先行进位ALU。



2.5.2 多功能算术/逻辑运算单元ALU



- 4、两级先行进位的ALU

  4片（组）的先行进位逻辑

  $C_{n+x}=G_0+P_0C_n$

  $C_{n+y}=G_1+P_1C_{n+x}=G_1+G_0P_1+P_0P_1C_n$

  $C_{n+z}=G_2+P_2C_{n+y}$

  $=G_2+G_1P_2+G_0P_1P_2+P_0P_1P_2C_n$

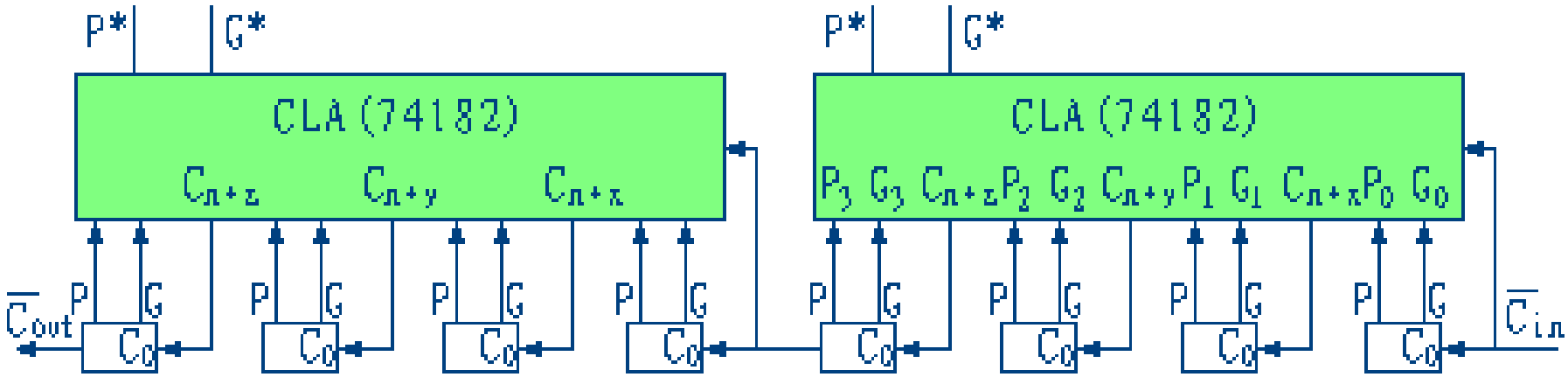
  $C_{n+4}=G_3+P_3C_{n+z}$

  $=G_3+G_2P_3+G_1P_2P_3+G_0P_1P_2P_3+P_0P_1P_2P_3C_n$

  $=G^*+P^*C_n$

  - G*为成组先行进位发生输出
  - P*为成组先行进位传送输出

2.5.2 多功能算术/逻辑运算单元ALU



2个74L182

8个4位ALU74L181

图2.14 用两个16位全先行进位逻辑级联组成的32位ALU

- **算术 / 逻辑运算单元74181ALU可完成_____。**
- A　16种算术运算功能
- B　16种逻辑运算功能
- C　16种算术运算功能和16种逻辑运算功能
- D　4位乘法运算和除法运算功能

- **四片74181和1片74182器件相配合，具有如下进位传递功能（　）。**

A. 串行进位

B. 组内先行进位，组间先行进位

C. 组内先行进位，组间串行进位

D. 组内串行进位，组间先行进位

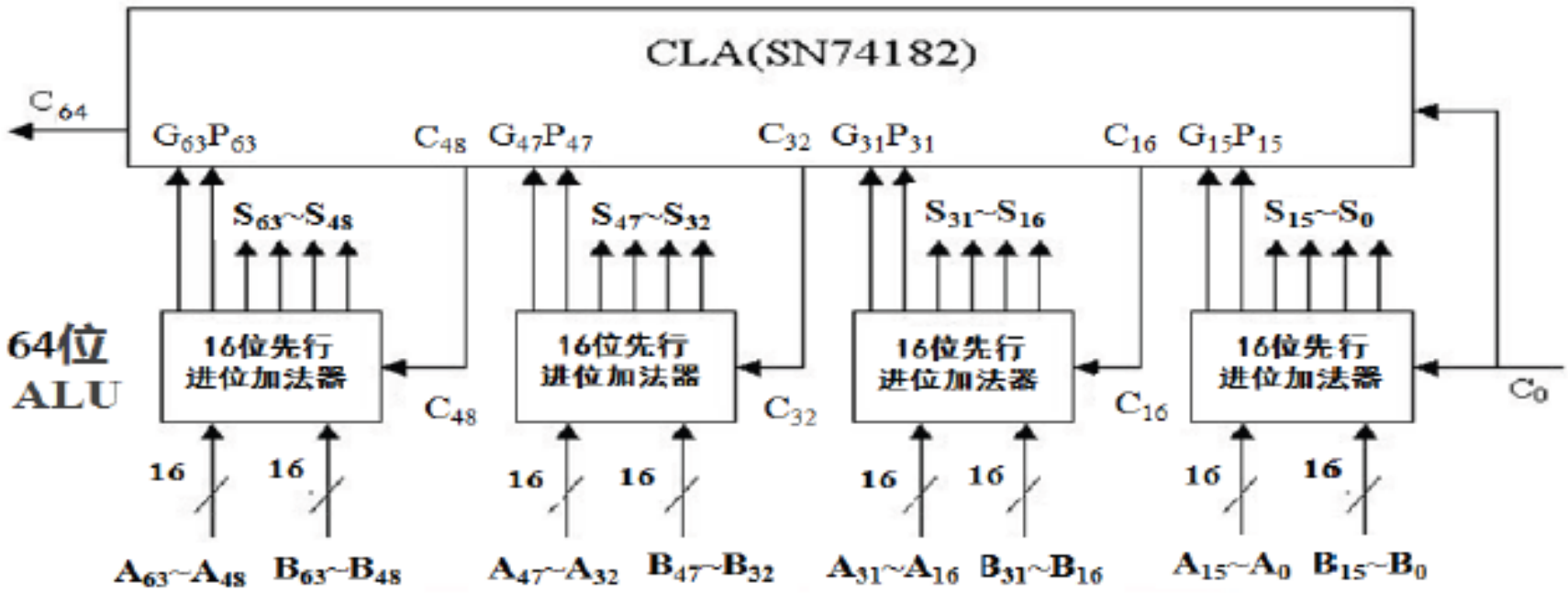
- 已知**SN74181**和**SN74182**芯片分别是**4**位**ALU**部件和**4**位**CLA**部件，用它们构成**64**位快速**ALU**时，所需的**SN74181**和**SN74182**的芯片数分别是（　）

A. 8，2

B. 8，3

C. 16，4

D. 16，5



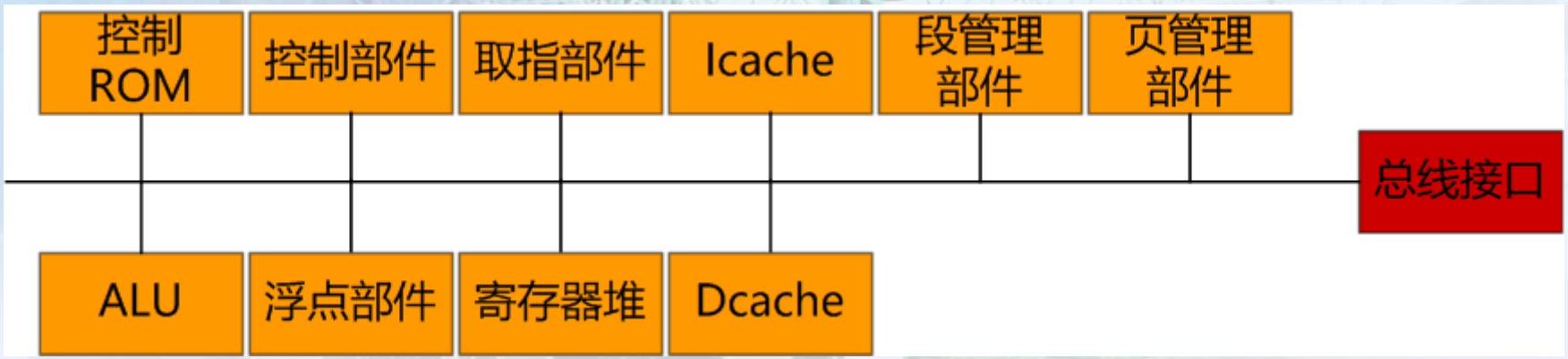
64位三级先行进位ALU

2.5.3 内部总线

- 总线结构
  - 共享性：总线所连接的所有部件都可以通过它传送信息。
  - 分时性：在某一个时间范围内，总线只允许一个部件发送信息到总线上。显然，共享是分时实现的。
  - 总线协议：总线不仅是一组传输线路，同时连接到总线上的所有部件都必须共同遵守一组规则和约定，称为总线协议（Protocol）。它一般包括信号线定义、数据格式、时序关系、信号电平、控制逻辑等。
  - 易扩展性。增加、移除部件方便。

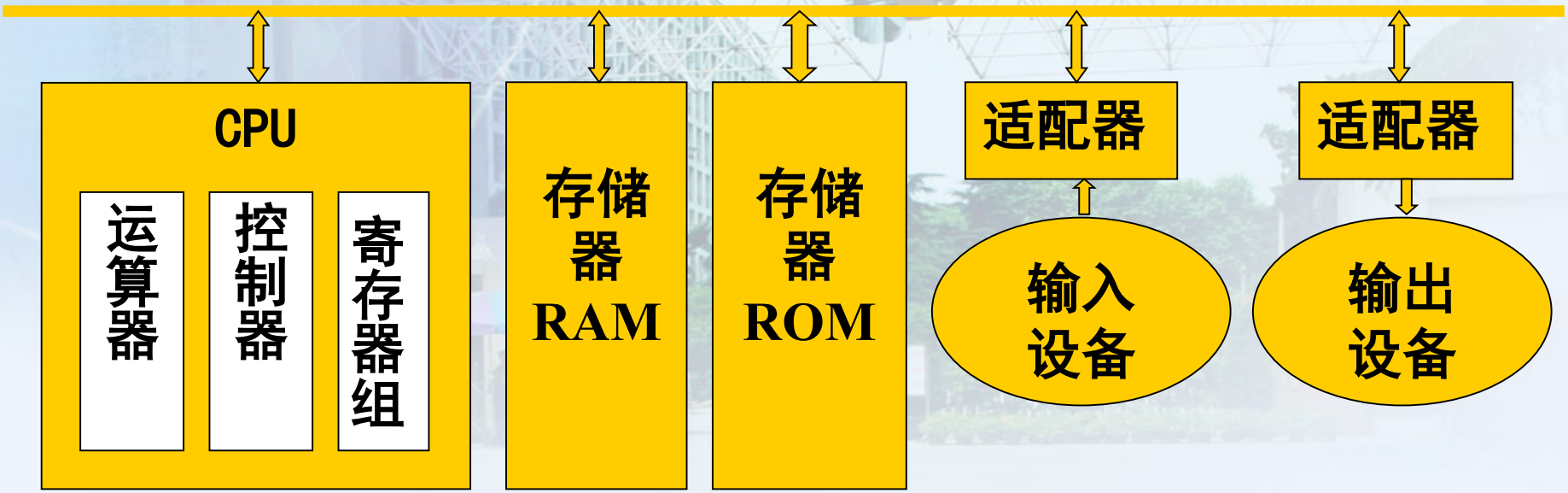
2.5.3 内部总线

- 内部总线
  - CPU内部各部件（算术/逻辑运算单元ALU、控制器、寄存器）之间的连线，即内部信息交换通路。



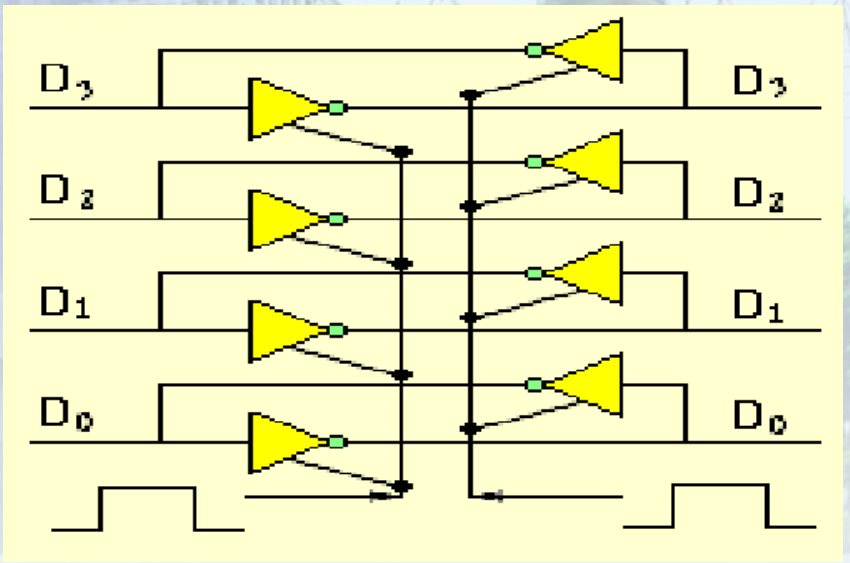
2.5.3 内部总线

- 外部总线
  - CPU与存储器、输入设备、输出设备之间的连线，即外部信息交换通路。



2.5.3 内部总线

- 总线的内部形态
  - 并行传送的一组线、同时传送若干位，每根线传送一个位。
  - 总线上数据的发送、接收是可控的。



发送选通信号

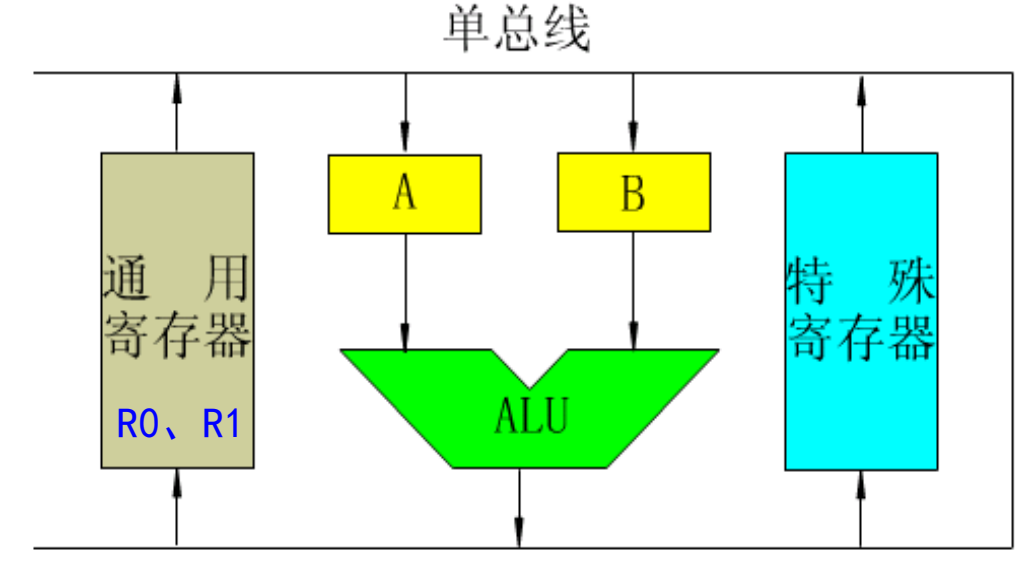
接收选通信号

2.5.4 定点运算器的基本结构

- 运算器包括
  - **ALU**
  - 阵列乘除器
  - 寄存器
  - 多路开关
  - 三态缓冲器
  - 数据总线等逻辑部件

2.5.4 定点运算器的基本结构

- 单总线结构的运算器



指令：ADD R0，R1

含义：R0+R1→R0

执行过程（三步）：

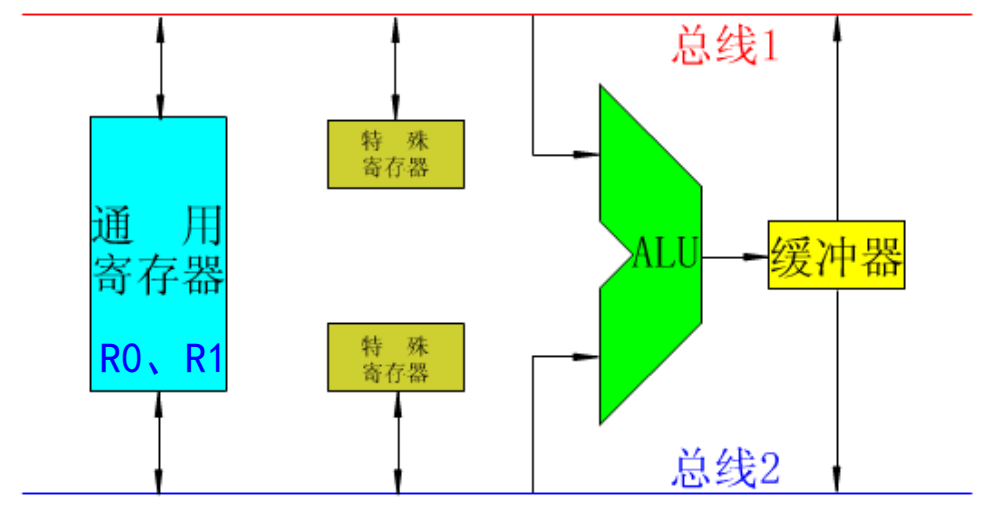
R0 →A

R1 →B

A+B→R0

2.5.4 定点运算器的基本结构

- 双总线结构的运算器



指令：ADD R0，R1

含义：R0+R1→R0

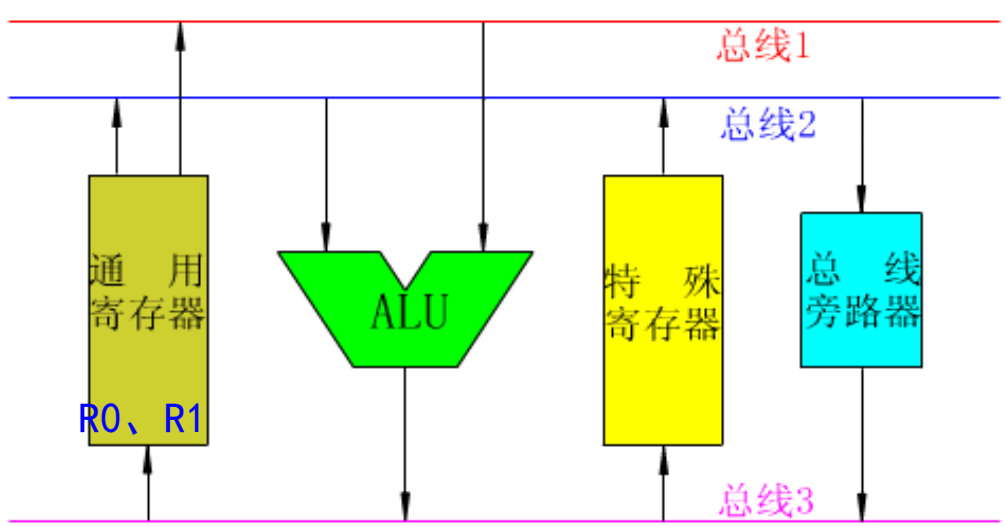
执行过程（二步）：

R0→总线1→ALU，R1→总线2→ALU；

ALU运算→缓冲器→总线1→R0

2.5.4 定点运算器的基本结构

- 三总线结构的运算器



指令：ADD R0，R1

含义：R0+R1→R0

执行过程（二步）：

R0→总线1→ALU，R1→总线2→ALU；

ALU运算→总线3→R0

2.6.1 浮点加法、减法运算

- 浮点加法、减法运算过程

设有两个浮点数 x 和 y ,它们分别为:

$$x = Mx \cdot 2^{Ex}$$

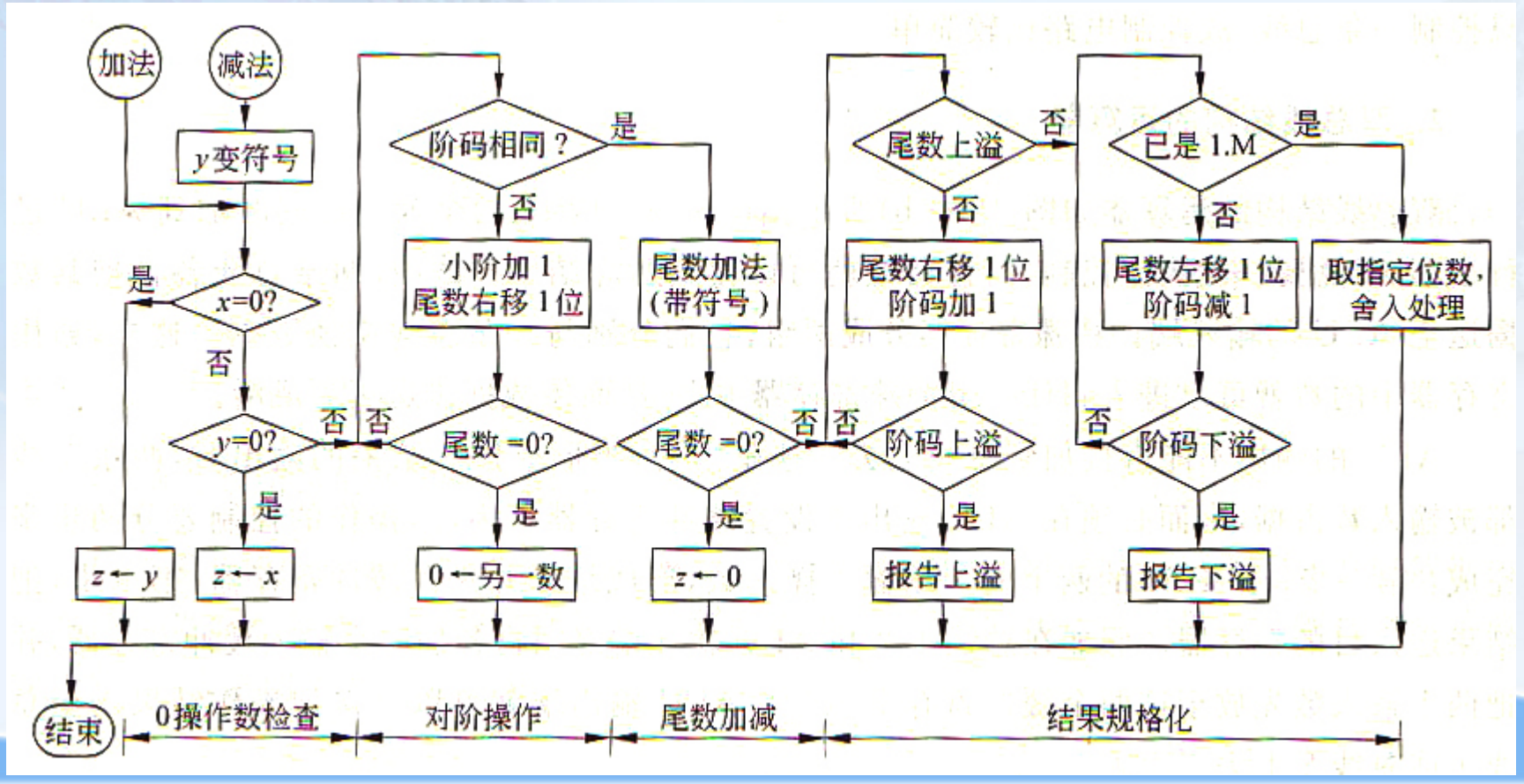
$$y = My \cdot 2^{Ey}$$

则浮点加法、减法运算结果为:

$$z=x\pm y=(Mx\cdot 2^{Ex-Ey}\pm My)\cdot 2^{Ey} \quad Ex \le Ey$$

2.6.1 浮点加法、减法运算

- 浮点加法、减法运算过程：多阶段（并行）



9-1、$x=2^{-011}\times0.100101$，$y=2^{-010}\times(-0.011110)$，求[x+y]

- 设尾数阶码均使用双符号位的补码表示

$[x]_{浮}$＝ 11 101，00.100101　　$[y]_{浮}$＝ 11 110，11.100010

1)求阶差并对阶

$[\triangle E]_{补}=[Ex-Ey]_{补}=[Ex]_{补}+[-Ey]_{补}$=11 101 ＋00 010＝11 111

对阶后的x表示为：

$[x]_{浮}$＝ 11 110，00.010010(1)

2)尾数求和

$[M_S]_{补}=[Mx+My]_{补}$ =11.110100(1)

$$\begin{array}{r} 00.0\,1\,0\,0\,1\,0\ (1) \\ +\quad 11.1\,0\,0\,0\,1\,0 \\ \hline 11.1\,1\,0\,1\,0\,0\,(1) \end{array}$$

3)规格化处理

执行2次左规处理， M_S= 11. 010010 (0)， E_S= 11 100

4)舍入处理　　采用0舍1入法处理，则舍去0

5)判溢出　　阶码符号位为11，不溢出

故得最终结果为 $x+y=2^{-100}\times(-0.101110)$

9-1、$x=2^{-011}\times0.100101$，$y=2^{-010}\times(-0.011110)$，求[x-y]

- 设尾数阶码均使用双符号位的补码表示

$[x]_{浮}$＝ 11 101，00.100101　　$[y]_{浮}$＝ 11 110，11.100010

1)求阶差并对阶

$[\triangle E]_{补}=[Ex]_{补}+[-Ey]_{补}=11\ 101+00\ 010=11\ 111$

对阶后的x表示为：

$[x]_{浮}$＝ 11 110，00.010010 (1)

2)尾数求差

$$\begin{array}{rl} & 00.0\,1\,0\,0\,1\,0(1) \quad [M_x]_{补} \\ + & 00.0\,1\,1\,1\,1\,0 \quad [-M_y]_{补} \\ \hline & 00.1\,1\,0\,0\,0\,0(1) \end{array}$$

$[M_S]_{补}=[Mx-My]_{补}=00.110000\ (1)$

3)规格化处理　不需规格化

4)舍入处理　采用0舍1入法处理，则进位，M_S= 00.110001

5)判溢出　阶码符号位为11，不溢出

故得最终结果为 $x-y=2^{-010}\times0.110001$

- **若浮点数用补码表示，则判断是否为规格化数的方法是_____。**

A. 阶符与数符相同为规格化数

B. 阶符与数符相异为规格化数

C. 数符与尾数小数点后第一位数字相异为规格化数

D. 数符与尾数小数点后第一位数字相同为规格化数

- **在二进制浮点数的规格化操作中，当尾数绝对值小于1/2时，应该将（　）。**

A. 尾数左移，阶码减小

B. 尾数右移，阶码增大

C. 尾数左移，阶码增大

D. 尾数右移，阶码减小

本次作业

P69习题：7(1)、9(2)

本章重点

- 原码、补码、反码、移码的求法及表示范围
- 补码加减法运算、溢出检测方法
- 并行加法器的进位方法及逻辑表达式
- 阵列乘法、陈列除法运算
- 浮点加减法运算

河海大學

Q&A