计算机组成原理

阵列乘法器、阵列除法器、定点运算器

王浩宇,教授

haoyuwang@hust.edu.cn

https://howiepku.github.io/

本节内容

- **阵列乘法器**
  - 不带符号阵列乘法器
  - 带符号阵列乘法器（间接补码乘法电路）
- **阵列除法器**
  - 可控加法/减法(CAS)单元
  - 不恢复余数的阵列除法器
- **定点运算器的组成**

阵列乘法器

- **在早期的计算机中为了简化硬件结构，采用串行的1位乘法方案，即多次执行“加法-移位”操作来实现。**
  - 这种方法不需要很多器件
  - 但串行方法太慢，不能满足对高速乘法的要求
- **阵列乘法器**
  - 乘法运算大概占全部算数运算的1/3
  - 高速乘法部件在速度和效率上都是十分必要

原码并行乘法

- **不带符号的阵列乘法器**
  - 设有两个不带符号的二进制整数：

    $A=a_{m-1}\cdots a_1a_0$ ；　　$B=b_{n-1}\cdots b_1b_0$

  - 它们的数值分别为a和b，即

$$a=\sum_{i=0}^{m-1}a_i2^i \qquad b=\sum_{j=0}^{n-1}b_j2^j$$

  - A、B相乘，产生m＋n位乘积P：

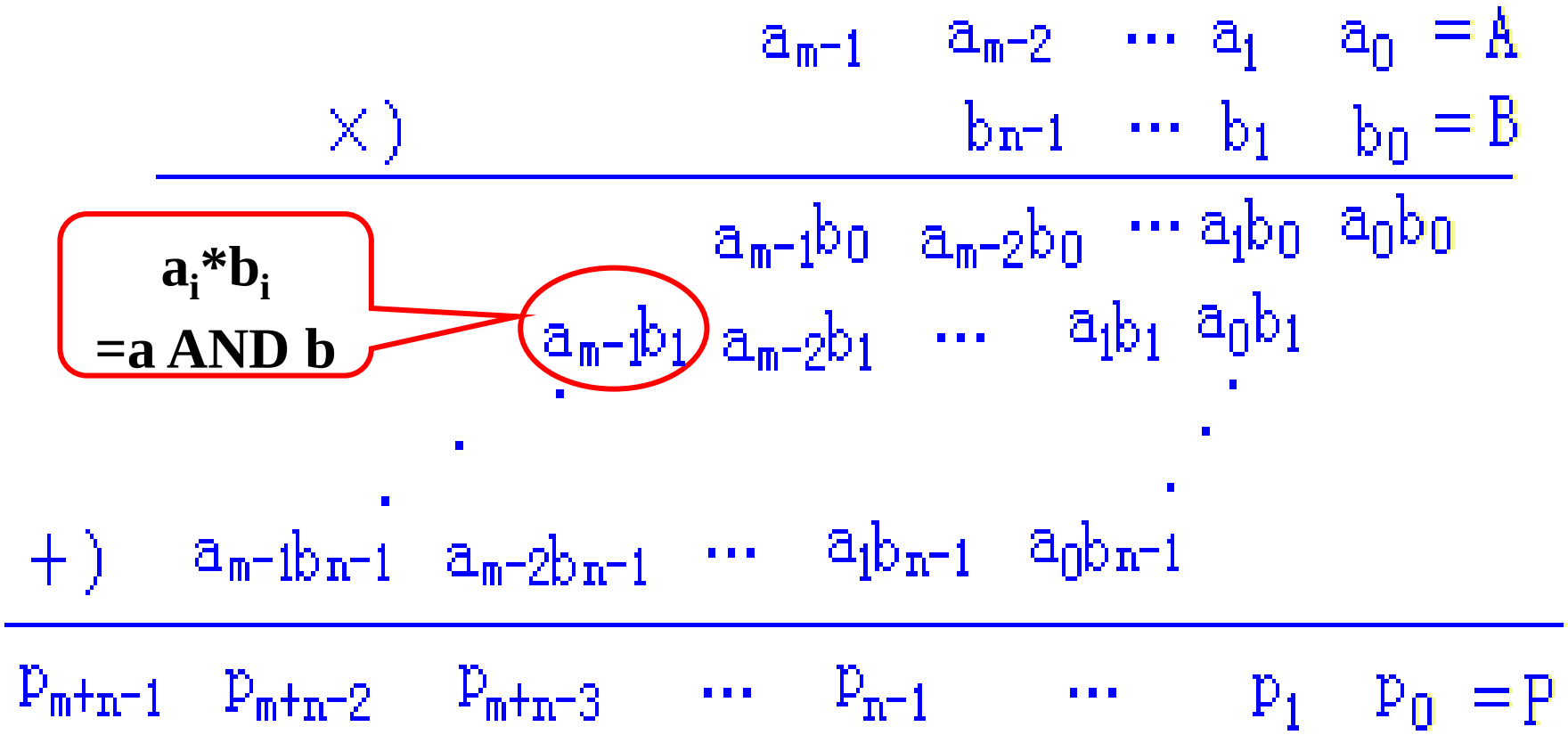
    $P=p_{m+n-1}\cdots p_1p_0$

  - 乘积P 的数值为

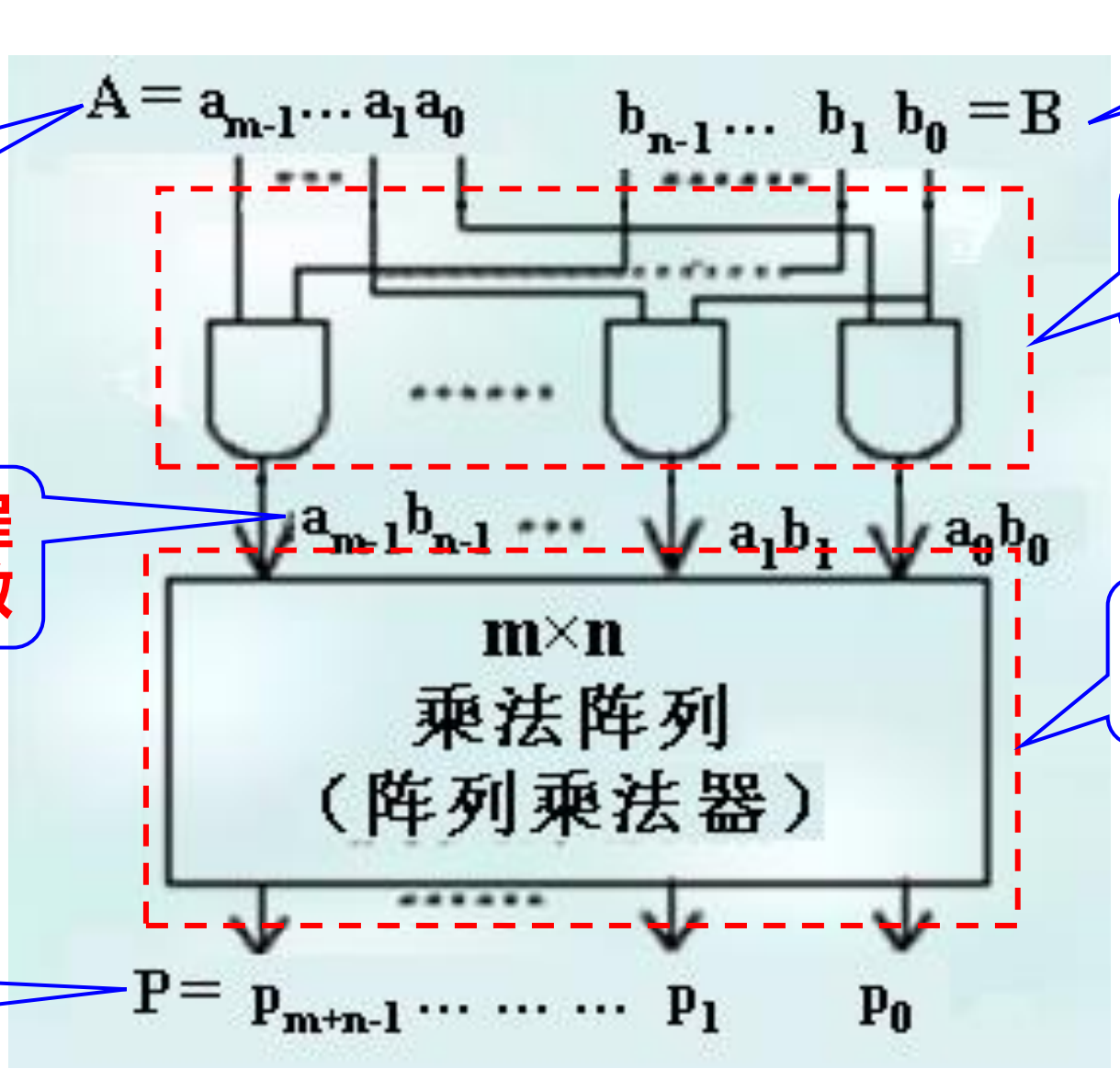
$$p=ab=(\sum_{i=0}^{m-1}a_i2^i)(\sum_{j=0}^{n-1}b_j2^j)=\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(a_ib_j)2^{i+j}=\sum_{k=0}^{m+n-1}p_k2^k$$

两个无符号数据的并行乘法电路

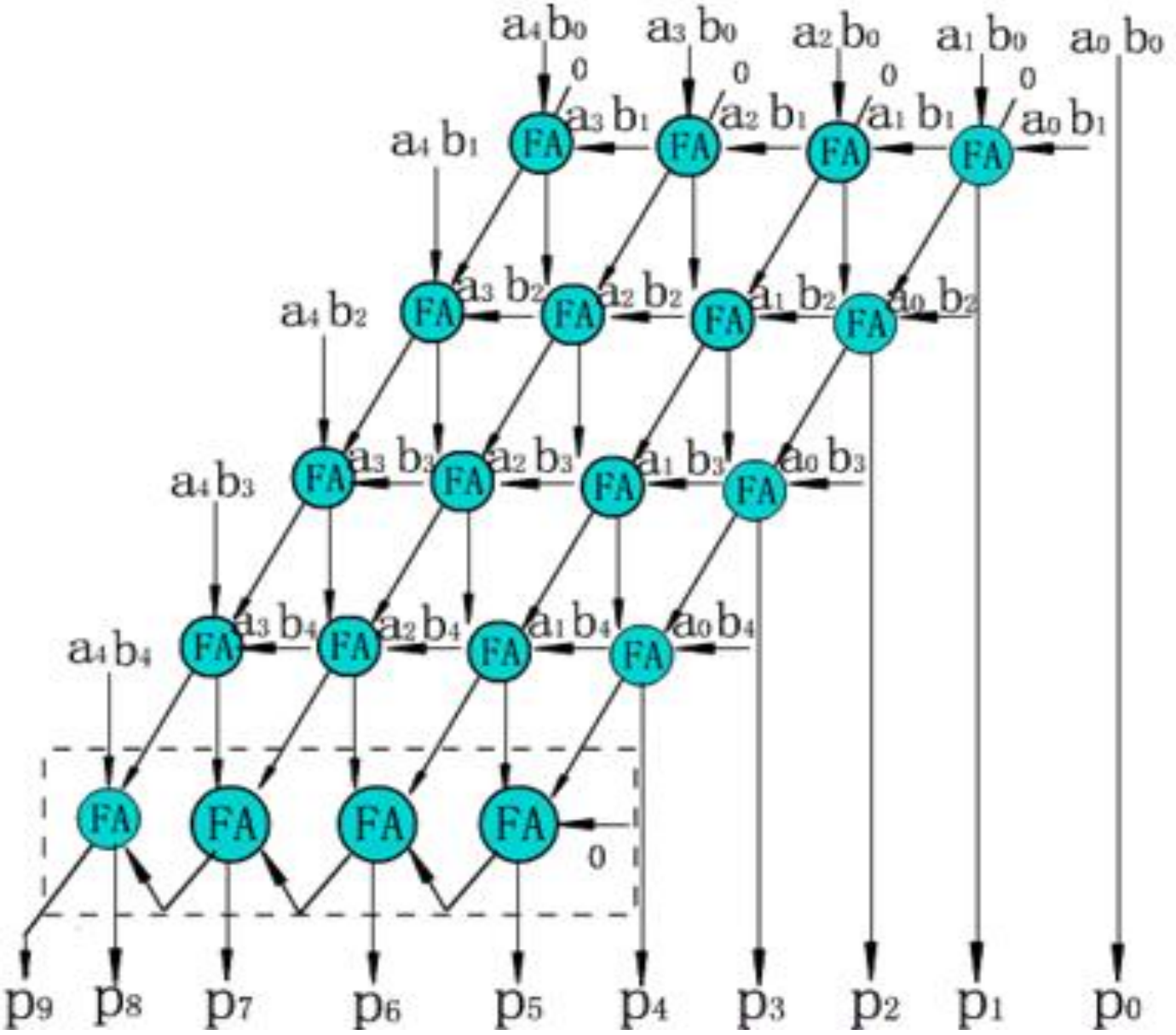
二进制数乘法的运算过程

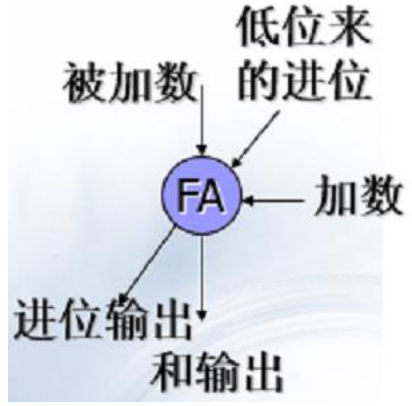


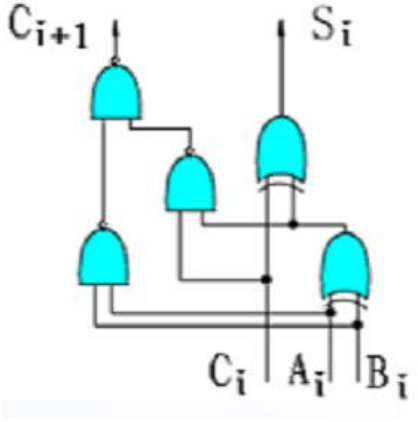
m×n位不带符号的阵列乘法器逻辑框图



5×5位不带符号阵列乘法器电路



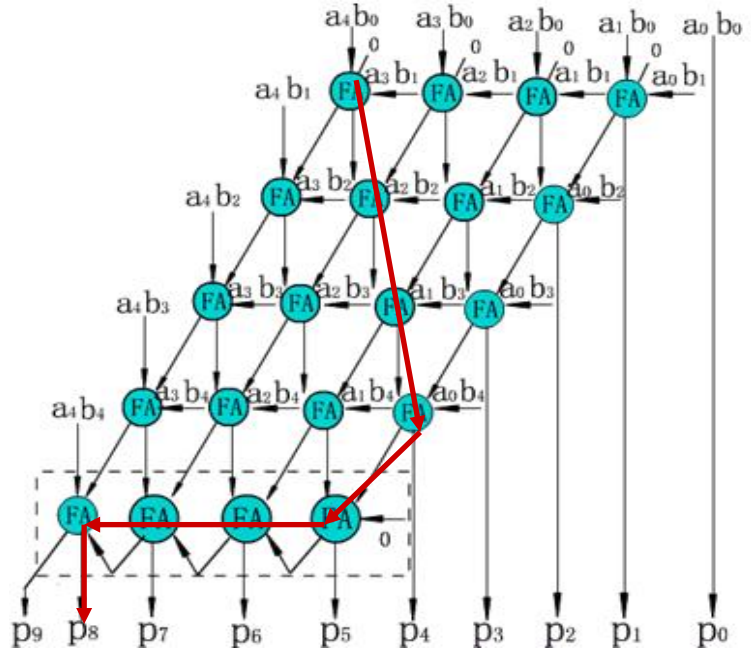




不带符号阵列乘法器电路

- N*N位阵列乘法器，需要？个全加器和？个与门

$n(n-1)$个全加器和n^2个与门



- N*N位阵列乘法器，总的乘法时间为？

$$T_m = Ta+(n-2)*6T + 5T + (n-2)*2T + 3T = (8n-6)T$$

最长延迟路径?

带符号的阵列乘法器

- **带符号原码的乘法：**
  - 设有两个n+1位带符号的二进制整数：

    $A=a_n\cdots a_1a_0$；　　$B=b_n\cdots b_1b_0$
  - 乘积$A*B=(an\oplus bn)(a_{n-1}\cdots a_1a_0 * \ b_{n-1}\cdots b_1b_0)$
- **带符号数的阵列乘法器电路：**
  - 在无符号数的阵列乘法器的基础上完成
  - 使用3个求补器，分别用对应的符号位控制
    - 2个算前求补器：将乘数真值的绝对值送入运算器
    - 1个算后求补器：将负的乘积的真值求出补码表示
  - 称为间接补码乘法电路

对2求补电路的原理

- 设有两个n+1位带符号补码：

  $A = a_n \cdots a_1 a_0$ ； $B = b_n \cdots b_1 b_0$

- 采用原码乘法电路计算A*B
  - 符号位不参与运算，只用于确定结果的符号；
  - 数据位要取绝对值参与运算：

    正数——不变；负数——各位取反，末位加1

- 对2求补电路：
  - 数据举例：

    $[-2]_{补}=1110$ $[-3]_{补}=1101$ $[-6]_{补}=1010$

    $[+2]_{补}=0010$ $[+3]_{补}=0011$ $[+6]_{补}=0110$

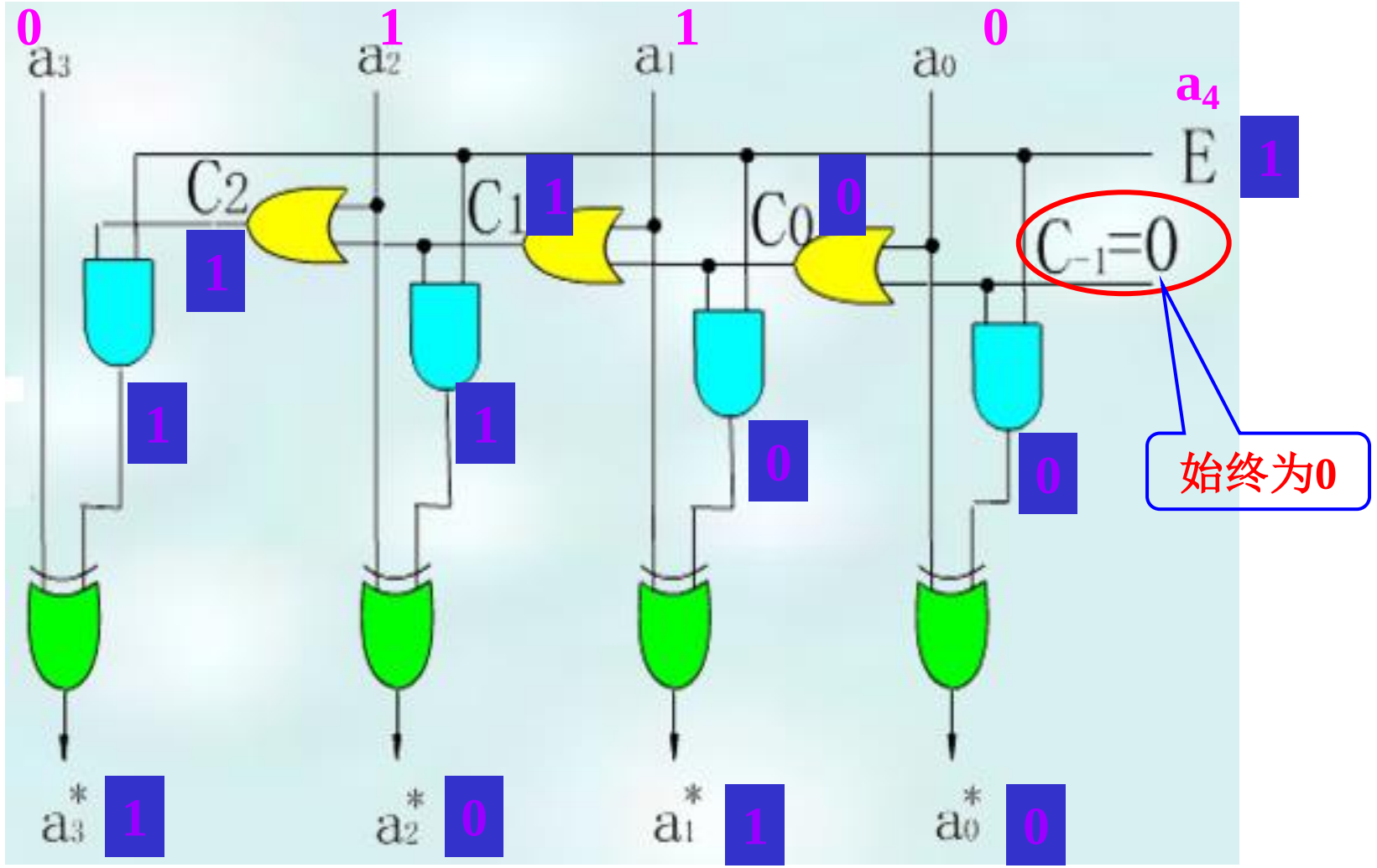
  - 两相反数的补码特征：

    自右向左，第一个“1”的右侧所有数据位，均相同

    左侧所有数据位，均相反

对2求补器电路逻辑

- 采用**按位扫描技术**来执行求补操作
- E为控制信号线，可由**数据a的符号位**来控制



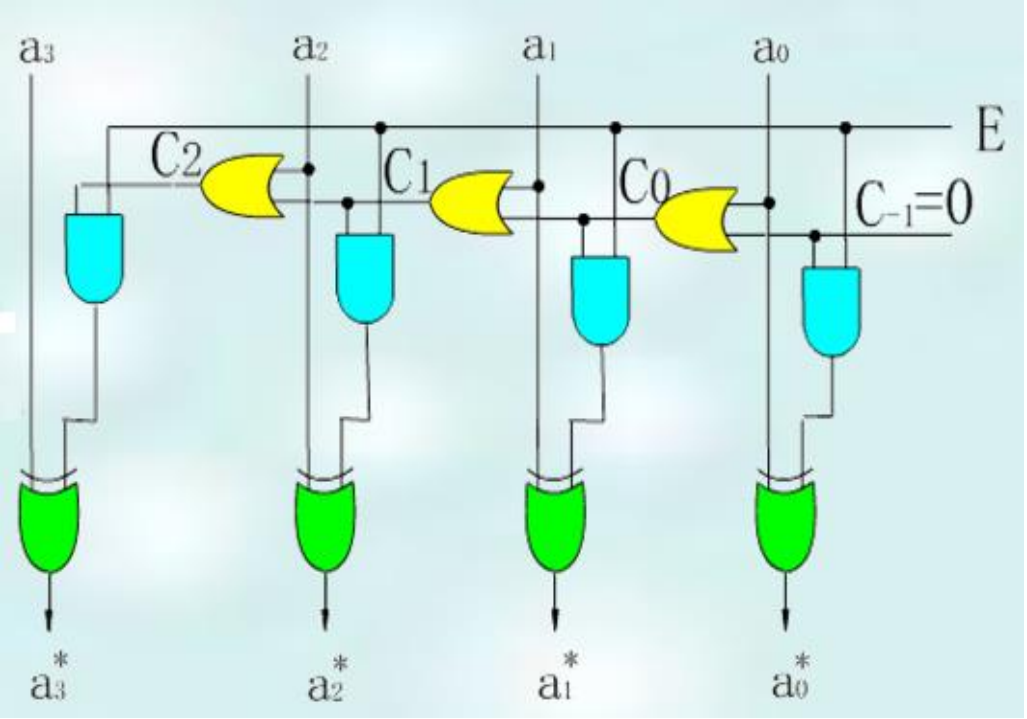
对二求补器电路

逻辑表达式

$C_{-1}=0,$

$C_i=a_i+C_{i-1}$

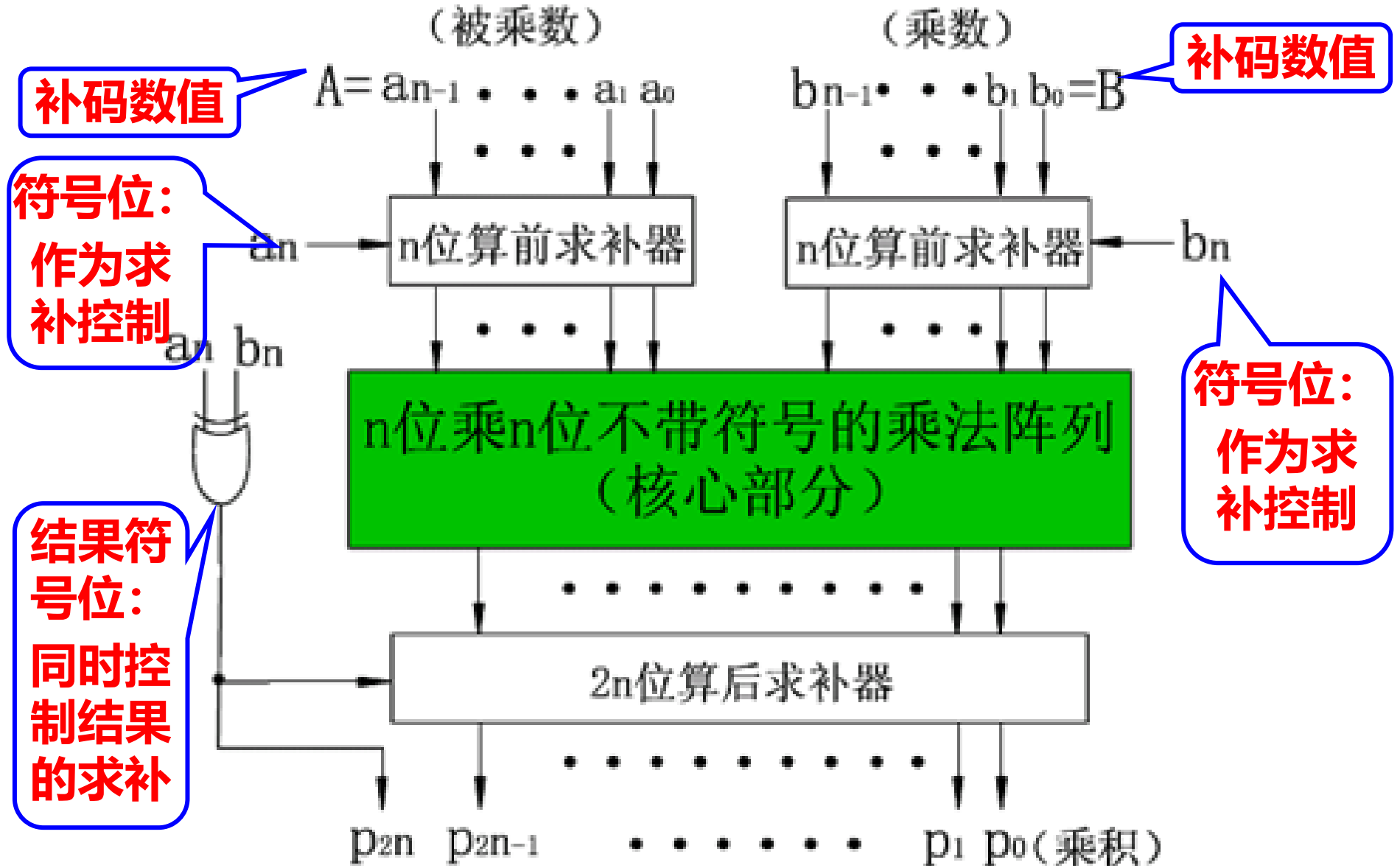
$a_i^*=a_i\oplus EC_{i-1},\ 0\le i\le n$



- 利用符号位来作为控制信号
  - 当控制信号线E为“1”时,启动对2求补的操作
  - 当控制信号线E为“0”时,输出将和输入相等
- 用这种对2求补器来转换一个(n + 1)位带符号的数,所需的总时间延迟为?

$$t_{TC} = n\cdot 2T + 5T = (2n + 5)T$$

间接补码乘法电路


（被乘数）
（乘数）
补码数值
A= an-1 • • • a1 a0
bn-1 • • • b1 b0=B
补码数值
符号位：
作为求
补控制
an
n位算前求补器
n位算前求补器
bn
an bn
n位乘n位不带符号的乘法阵列
（核心部分）
符号位：
作为求
补控制
结果符
号位：
同时控
制结果
的求补
2n位算后求补器
p2n p2n-1 • • • • • • p1 p0（乘积）

例 设 x ＝＋15， y ＝－13，用带求补器的补码阵列乘法器求出乘积 x·y ＝？

设最高位为符号位，输入数据用5位补码表示：

$[x]_{补}$=01111　　$[y]_{补}$=10011

符号位单独运算：$x_f \oplus y_f = 0 \oplus 1 = 1$

算前求补器输出：　|x|=1111 ，|y|=1101

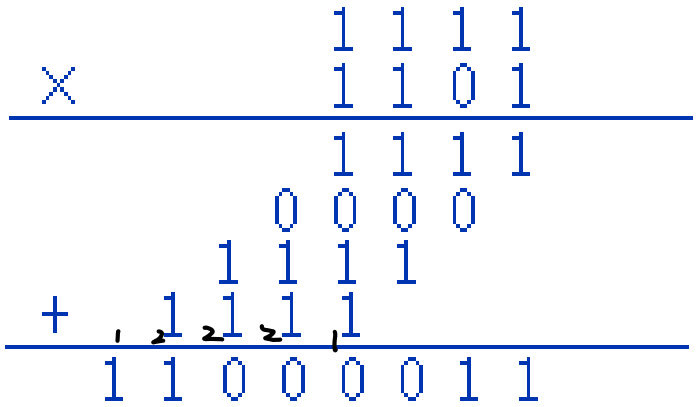
乘法阵列输出：

|x|×|y|=1100 0011

乘积符号位为1，则算后求补器输出为00111101

所以，$[x \times y]_{补}$=1 00111101，　$x \times y = (-195)_{10}$

十进制数验证：15×（−13）$=(-195)_{10}$



[例]设 x ＝－15, y ＝－13, 用带求补器的补码阵列乘法器求出乘积 x × y ＝?

设最高位为符号位，输入数据用5位补码表示：

$[x]_{补}$=10001　$[y]_{补}$=10011

符号位单独运算：$x_f \oplus y_f = 1 \oplus 1 = 0$

算前求补器输出：　|x|=1111 ，|y|=1101

乘法阵列输出：

|x| × |y| =1100　0011

乘积符号位为0，则算后求补器输出为11000011

所以，$[x \times y]_{补}$=0 11000011，$x \times y = (+195)_{10}$

十进制数验证：　(−15) × （−13）$= (+195)_{10}$

```
          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
```

本节内容

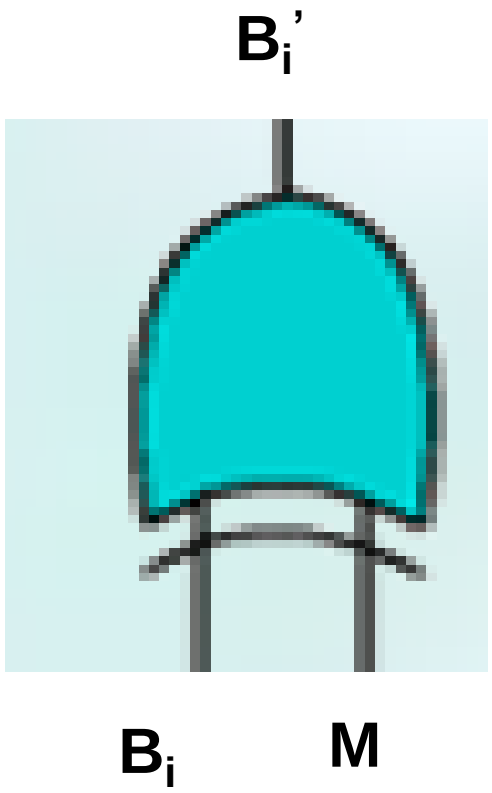
- 阵列乘法器
  - 不带符号阵列乘法器
  - 带符号阵列乘法器（间接补码乘法电路）
- **阵列除法器**
- 定点运算器的组成

并行除法器

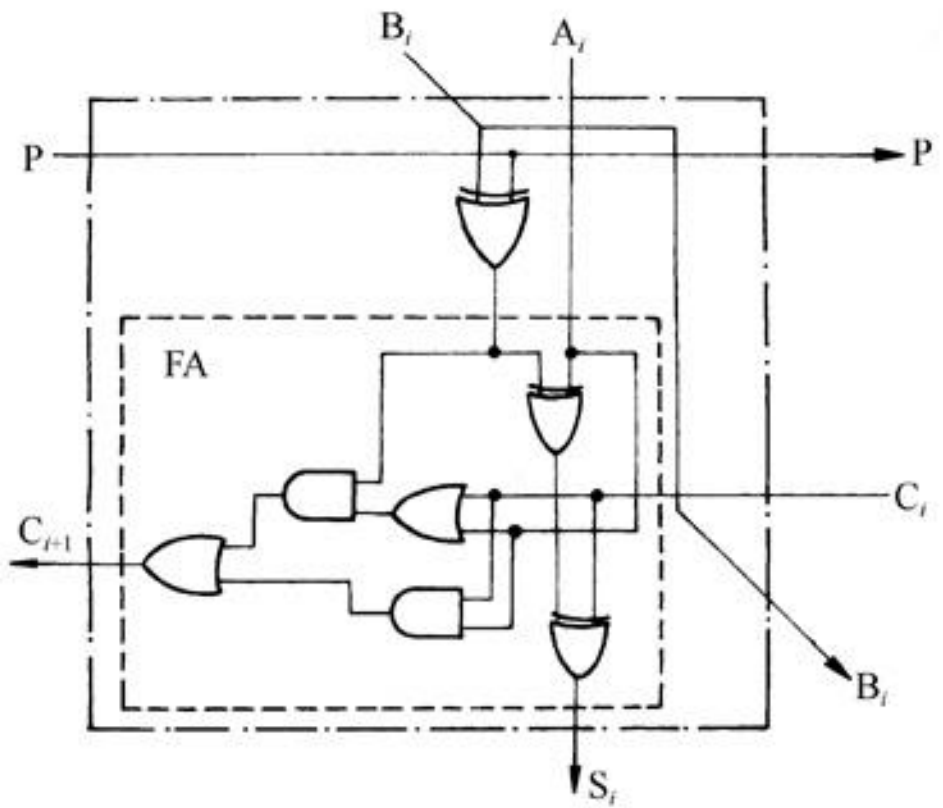
- 并行除法器也称阵列除法器
- 并行除法器的类型
  - 恢复余数阵列除法器
  - 不恢复余数阵列除法器（采用加减交替法原理）
    - 基本元件：可控加法/减法(CAS)单元
  - 补码阵列除法器

回顾：多位二进制数据加法/减法器

- 将减法转换成加法
  - $[A]_{补}-[B]_{补} = [A]_{补}+[-B]_{补}$
- 由$[B]_{补}$ 求$[-B]_{补}$
  - $[B]_{补}$ 求各位取反，末位加1；
- 将加减法电路合二为一
  - 使用异或运算；
  - 当M=0时，$B_i' = B_i$
  - 当M=1时，$B_i' = \neg B_i$ ；



可控加法/减法(CAS)单元



(a) 可控加法/减法(CAS)单元的逻辑图

- 用于加减交替法的除法器中；
- 由控制端P选择运算类型：
  - P=0，作加法运算
  - P=1，作减法运算
- 核心：全加器
  - 四个输入
    - 被加/减数A_i、加/减数B_i、低位进/借位C_i、控制端P
  - 四个输出
    - 加/减数B_i、当位和/差S_i、向高位的进/借位C_{i+1}、控制端P

可控加法/减法(CAS)单元

CAS单元的输入与输出的关系可用如下一组逻辑方程来表示：

$$S_i = A_i \oplus (B_i \oplus P) \oplus C_i$$

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

- 当P＝0时，即是我们熟悉的一位全加器(FA)的公式：

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

$$C_{i+1} = A_i B_i + B_i C_i + A_i C_i$$

- 当P＝1时，则得求差公式：

$$S_i = A_i \oplus \overline{B}_i \oplus C_i$$

$$C_{i+1} = A_i \overline{B}_i + \overline{B}_i C_i + A_i C_i$$

其中$\overline{B}_i = B_i \oplus 1$。

在减法情况下：

输入C_i称为借位输入，而C_{i+1}称为借位输出。

不恢复余数的阵列除法器

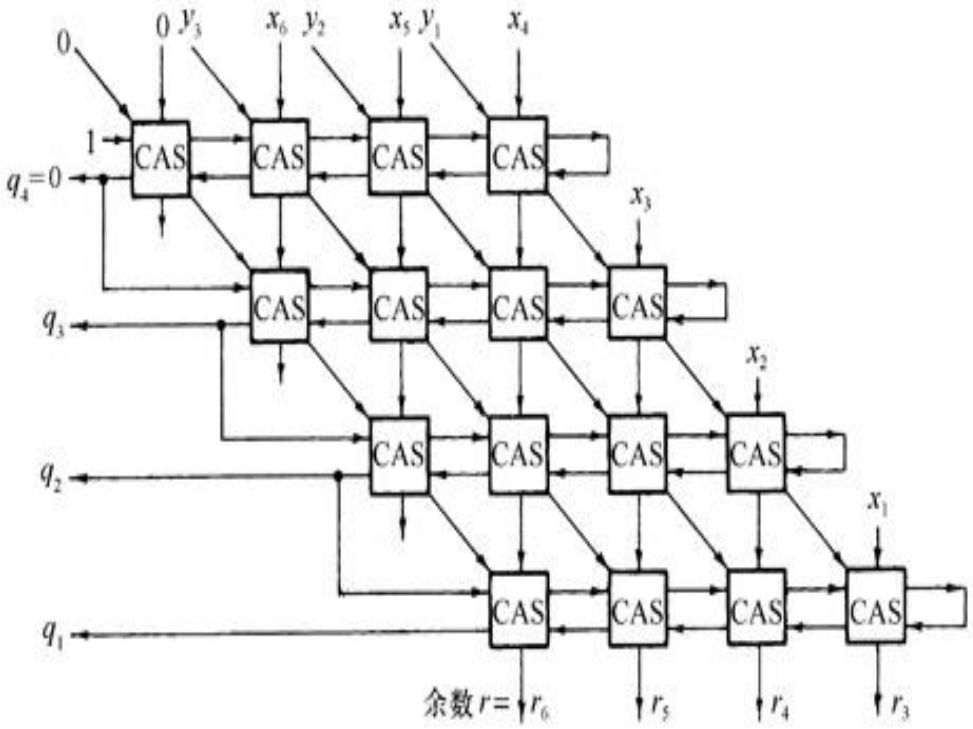
在不恢复余数的除法阵列中：

- 当余数为正时（$r_i \geq 0$），商“1”，下次做减法运算，减法是用2的补码运算来实现的，此时

$$[x-y]_{补}=[x]_{补}+[-y]_{补};$$

- 当余数为负时（$r_i < 0$），商“0”，下次做加法运算；
- 每次运算完成后要将余数左移一位，再与除数做加或减运算；
- 商的符号由两数的符号按位相加求得。

不恢复余数的阵列除法器



(b) 4 位除 4 位阵列除法器

- 被除数 $x=0.x_6x_5x_4x_3x_2x_1$
  - 通过顶部一行和最右边对角线输入
- 除数 $y=0.y_3y_2y_1$
  - 沿对角线方向输入
  - 逐行右移
- 商 $q=0.q_3q_2q_1$
  - 每次的加/减运算中产生
- 余数 $r=0.00r_6r_5r_4r_3$
  - 由最后一行产生

本节内容

- 阵列乘法器
  - 不带符号阵列乘法器
  - 带符号阵列乘法器（间接补码乘法电路）
- 阵列除法器
- **定点运算器的组成**

定点运算器的组成

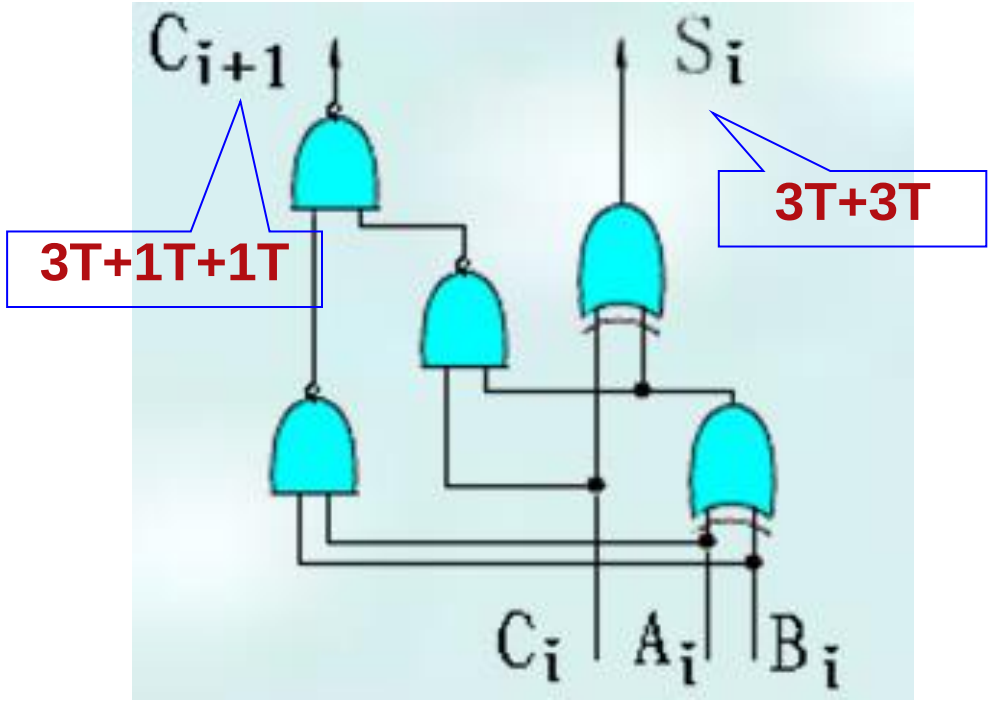
- 逻辑运算
- 多功能算术/逻辑运算单元ALU
- 先行进位ALU
- 内部总线
- 定点运算器的基本结构

定点运算器的组成

- **基本组成包括：**
  - 算术逻辑运算单元ALU：核心部件
  - 数据缓冲寄存器：用来存放参与计算的数据及运算结果，只对硬件设计者可见，即只被控制器硬件逻辑控制或微程序所访问
  - 通用寄存器堆：用于存放程序中用到的数据，它可以被软件设计者所访问。
  - 内部总线：用于连接各个部件的信息通道。
  - 其他可选电路
- **设计定点运算器，如何确定各部件的功能和组织方式是关键**
  - 指令系统
  - 机器字长
  - 机器数及其运算原理
  - 体系结构

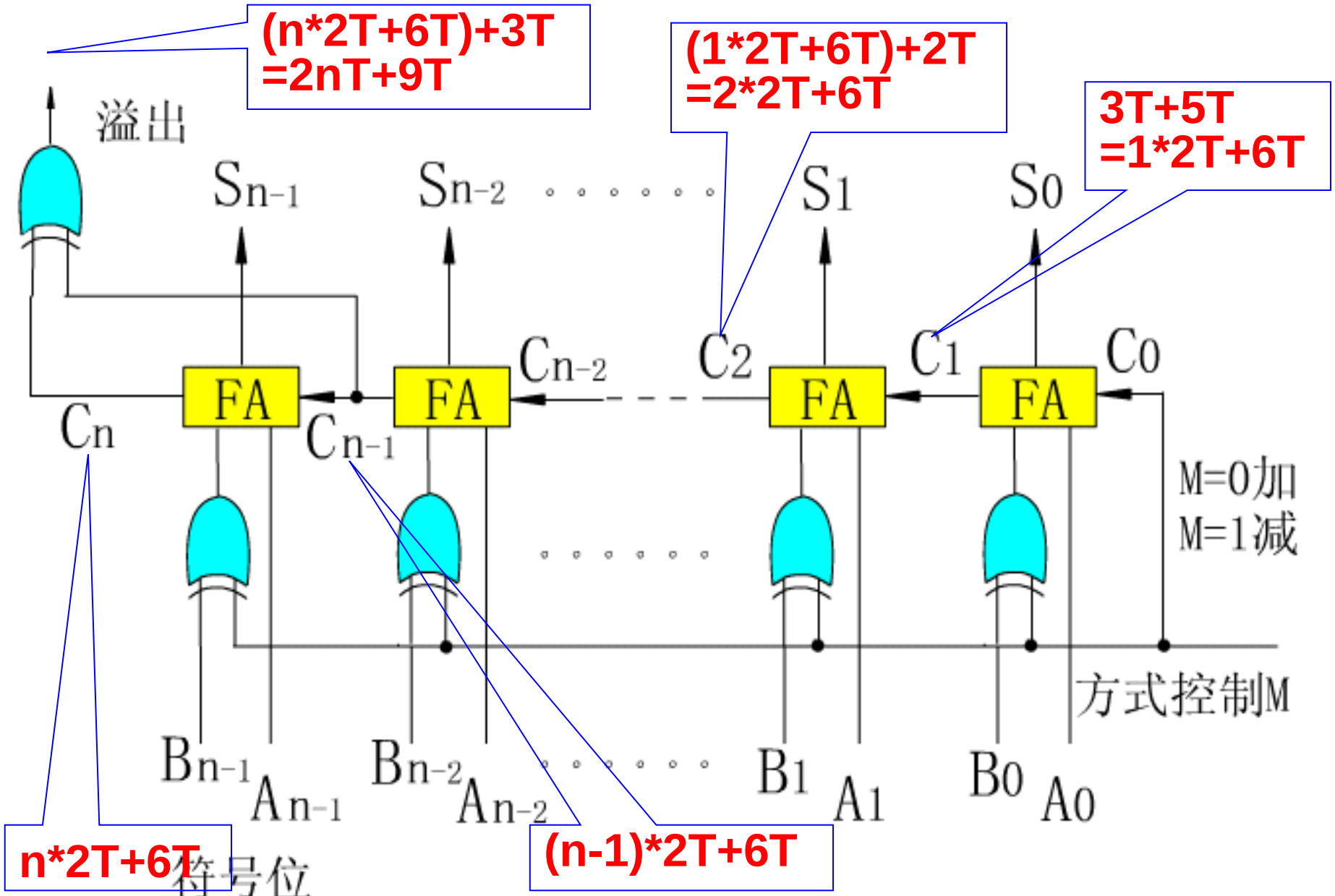
一位二进制数据的全加器的逻辑结构（回顾）

- 全加运算的真值表如右所示：
- 两个输出端的逻辑表达式
  - $S_i = A_i \oplus B_i \oplus C_i$
  - $C_{i+1} = A_iB_i + B_iC_i + C_iA_i = A_iB_i + (A_i \oplus B_i)C_i$
  - 全加器逻辑结构：



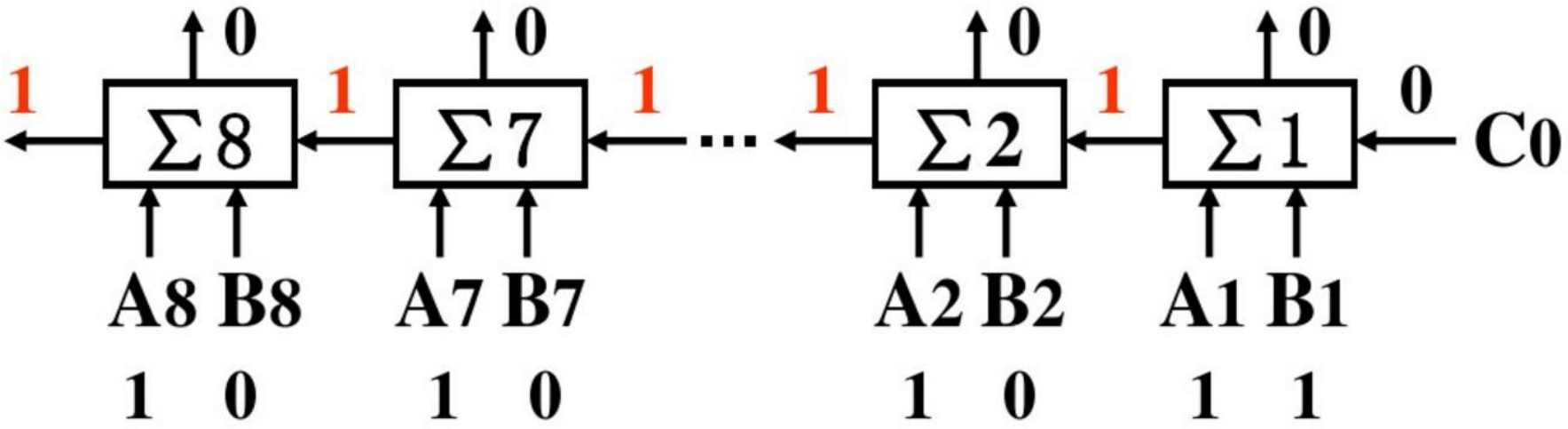
输入			输出	
A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

行波进位加法器（回顾）


(n*2T+6T)+3T
=2nT+9T
(1*2T+6T)+2T
=2*2T+6T
3T+5T
=1*2T+6T
溢出
Sn-1
Sn-2
S1
S0
FA
FA
FA
FA
Cn
Cn-1
Cn-2
C2
C1
C0
M=0加
M=1减
方式控制M
Bn-1
An-1
Bn-2
An-2
B1
A1
B0
A0
n*2T+6T
(n-1)*2T+6T
符号位

行波进位加法器

- 特点：各位同时相加
- 串行进位



- 影响速度的主要因素：进位信号的传递
- 如何改进串行进位延迟长的缺点？

行波进位加法器

- 只能完成加法和减法两种操作，不能完成逻辑操作

- 如何以加法器为核心，通过输入选择逻辑扩展为具有多种算术、逻辑功能的ALU?

多功能算术/逻辑运算单元 ALU （arithmetic logic unit）

- 多功能算术/逻辑运算单元ALU，本节介绍74LS181
- **74LS181： The first complete ALU on a single chip, it was used as the arithmetic/logic core in the CPUs of many historically significant minicomputers and other devices.**
- **Many computer CPUs and subsystems were based on the 74181**
  - NOVA - First widely available 16-bit minicomputer
  - Xerox Alto - The first computer to use the desktop metaphor and graphical user interface (GUI)

多功能算术/逻辑运算单元 ALU

- 如何改进串行进位延迟长的缺点？

  先行进位加法器

- 如何以加法器为核心，通过输入选择逻辑扩展为具有多种算术、逻辑功能的ALU？

  函数发生器，通过不同的控制参数可以得到不同的组合函数

多功能算术/逻辑运算单元ALU

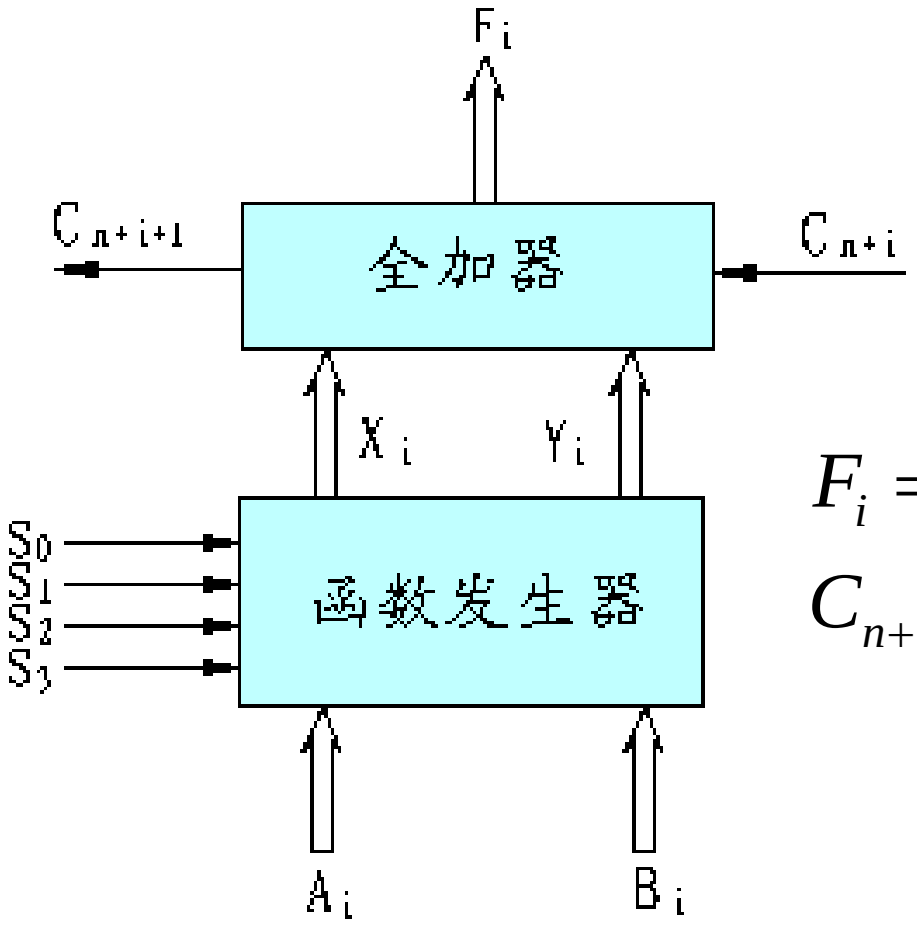
- 通过改变其输入端Ai和Bi来实现算术运算和逻辑运算功能。怎样实现？
  - 基本思想：一位全加器FA的逻辑表达式：

$$F_i = A_i \oplus B_i \oplus C_{n+i}$$

$$C_{n+i+1} = A_i B_i + A_i C_{n+i} + B_i C_{n+i}$$

- 为了实现多种算术逻辑运算，可将Ai和Bi输入一个函数发生器（进位传递函数和进位产生函数）得到输出Xi和Yi，作为一位全加器的输入

ALU的逻辑图与逻辑表达式



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

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

不同的控制参数可以得到不同的组合函数，因而能够实现多种算术运算和逻辑运算

多功能算术/逻辑运算单元ALU

XiYi 与控制参数和输入量的关系构造如下真值表

S_0 S_1	Y_i	S_2 S_3	X_i
0 0	$\overline{A_i}$	0 0	1
0 1	$\overline{A_i}B_i$	0 1	$\overline{A_i}+B_i$
1 0	$\overline{A_i}\,\overline{B_i}$	1 0	$\overline{A_i}$
1 1	0	1 1	$\overline{A_i}+\overline{B_i}$

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

$$X_i=\overline{S_2}\,\overline{S_3}+\overline{S_2}S_3(\overline{A_i}+\overline{B_i})+\overline{S_2}S_3(\overline{A_i}+B_i)+S_2S_3\overline{A_i}$$

多功能算术/逻辑运算单元ALU

进一步化简得到下式

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

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

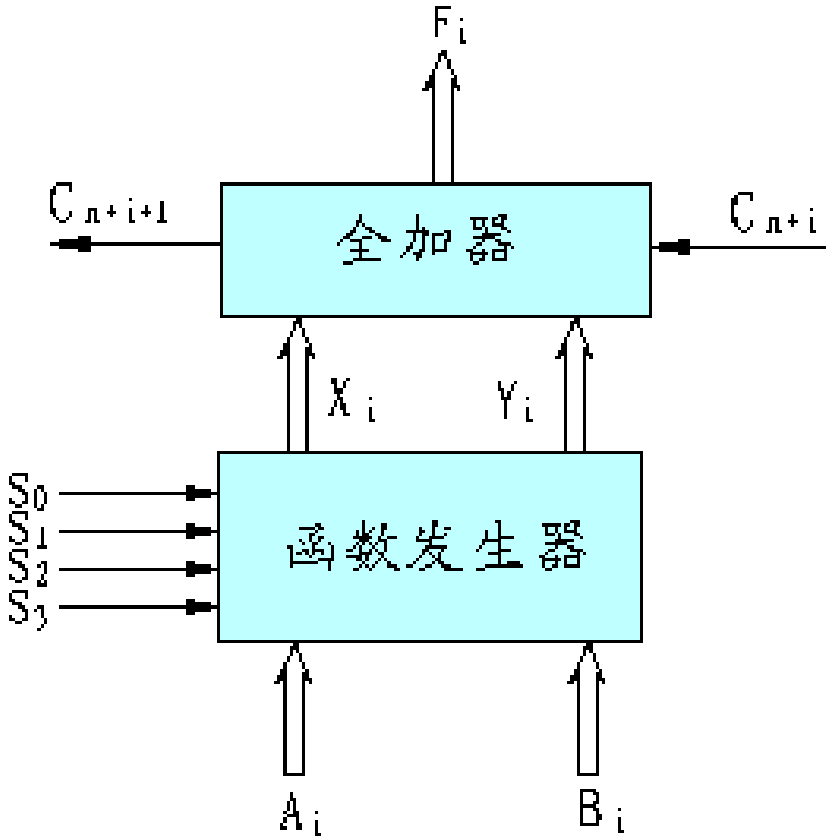
可以证明：**Xi+Yi=Xi**　　**XiYi=Yi**

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

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

$$\therefore C_{n+i+1} = X_i Y_i + C_{n+1}(Y_i + X_i) = Y_i + C_{n+1} X_i$$

多功能算术/逻辑运算单元ALU



例如：

$S_3S_2S_1S_0$＝0000

代入：

$$X_i = \overline{S_3 A_i B_i + S_2 A_i \overline{B_i}} = \overline{0+0} = 1$$

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

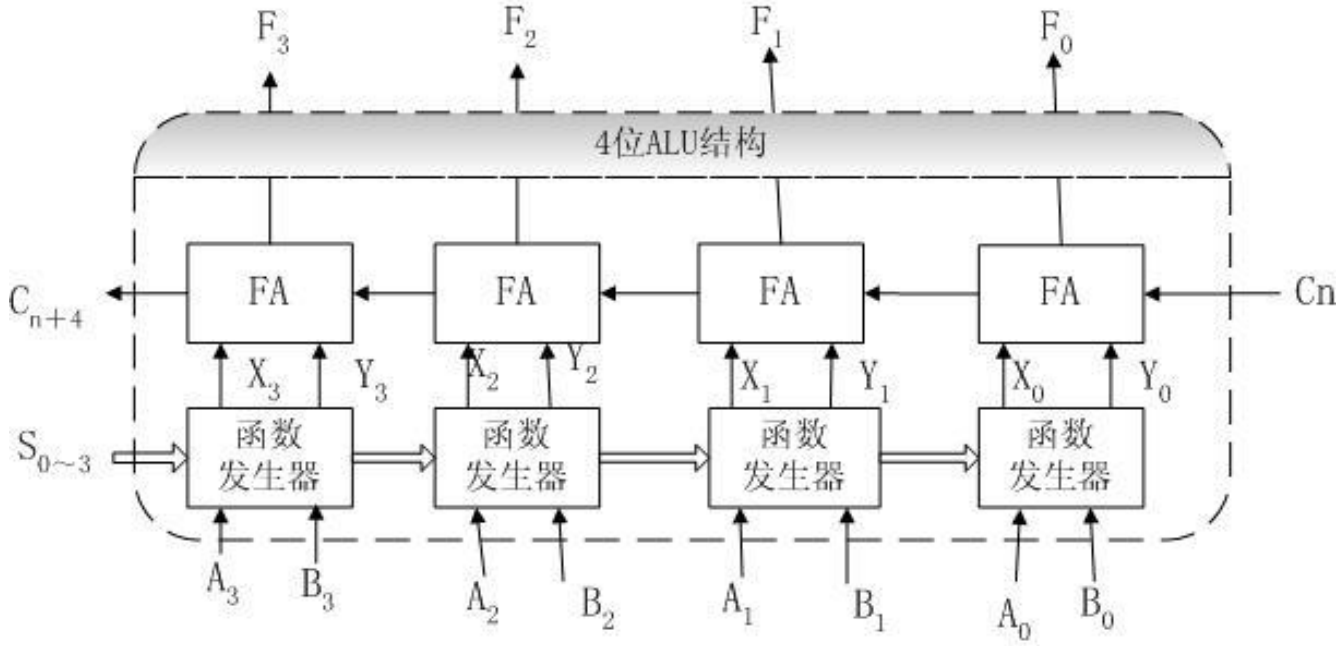
$$F_i = Y_i \oplus X_i \oplus C_{n+1} = \overline{A_i} \oplus 1 \oplus C_{n+1} = A_i \text{（设}C_{n+1}=0\text{）}$$

所以在S_3～S_0＝0000的时候，输出结果为A_i

- 每种运算只针对1位二进制编码
- 思考：如何设计4位ALU？16位呢？

多功能算术/逻辑运算单元ALU

- 4位ALU



- **片内是串行进位还是并行进位？**

由上图结构中可以看出

$$C_{n+1}=Y_0+X_0C_n$$
$$C_{n+2}=Y_1+X_1C_{n+1}$$
$$C_{n+3}=Y_2+X_2C_{n+2}$$
$$C_{n+4}=Y_3+X_3C_{n+3}$$

是串行进位，速度慢，为实现快速ALU，需加以改进

多功能算术/逻辑运算单元ALU

上述片内进位采用串行，具有延时长的缺点如何改进？

C_{n+i}与X、Y有关，而每一位中X、Y的产生是不是同时的？

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}

第0位向第1位的进位公式为

$$C_{n+1}=Y_0+X_0C_n \qquad (1)$$

其中C_n是向第0位（末位）的进位。

第1位向第2位的进位公式为

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

（C_{n+1}用（1）式代入）

第2位向第3位的进位公式为

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

第3位的进位输出（即整个4位运算进位输出）公式为

$$C_{n+4}=Y_3+X_3C_{n+3}=Y_3+Y_2X_3+Y_1X_2X_3+Y_0X_1X_2X_3+X_0X_1X_2X_3C_n$$

多功能算术/逻辑运算单元ALU

- 为了实现多片（组）ALU之间的先行进位，需要配合电路，称为先行进位发生器（CLA），所以定义G、P两个信号
- 令$G=Y_3+Y_2X_3+Y_1X_2X_3+Y_0X_1X_2X_3$

  $P=X_0X_1X_2X_3$

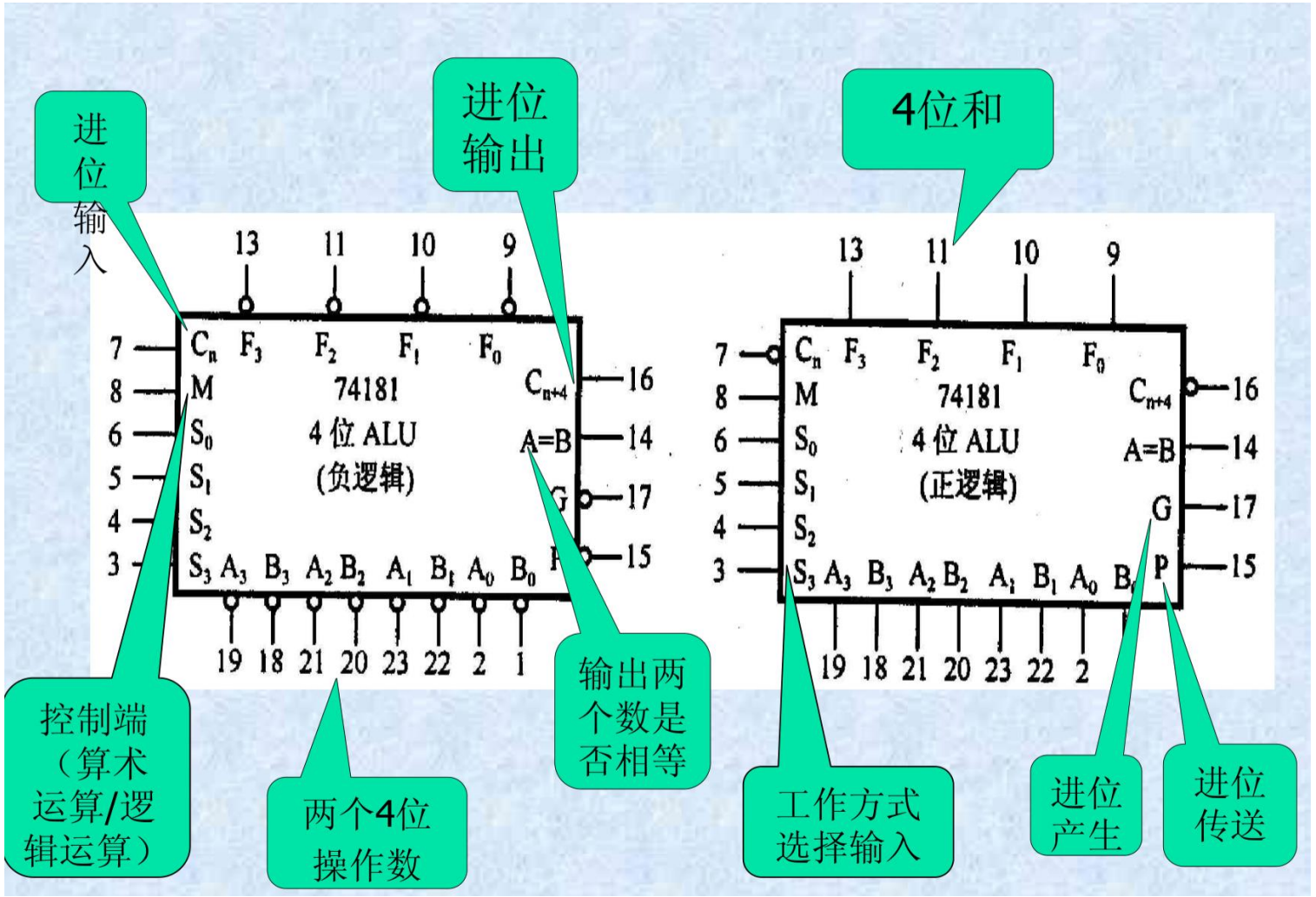
  G为进位发生输出

  P为进位传送输出

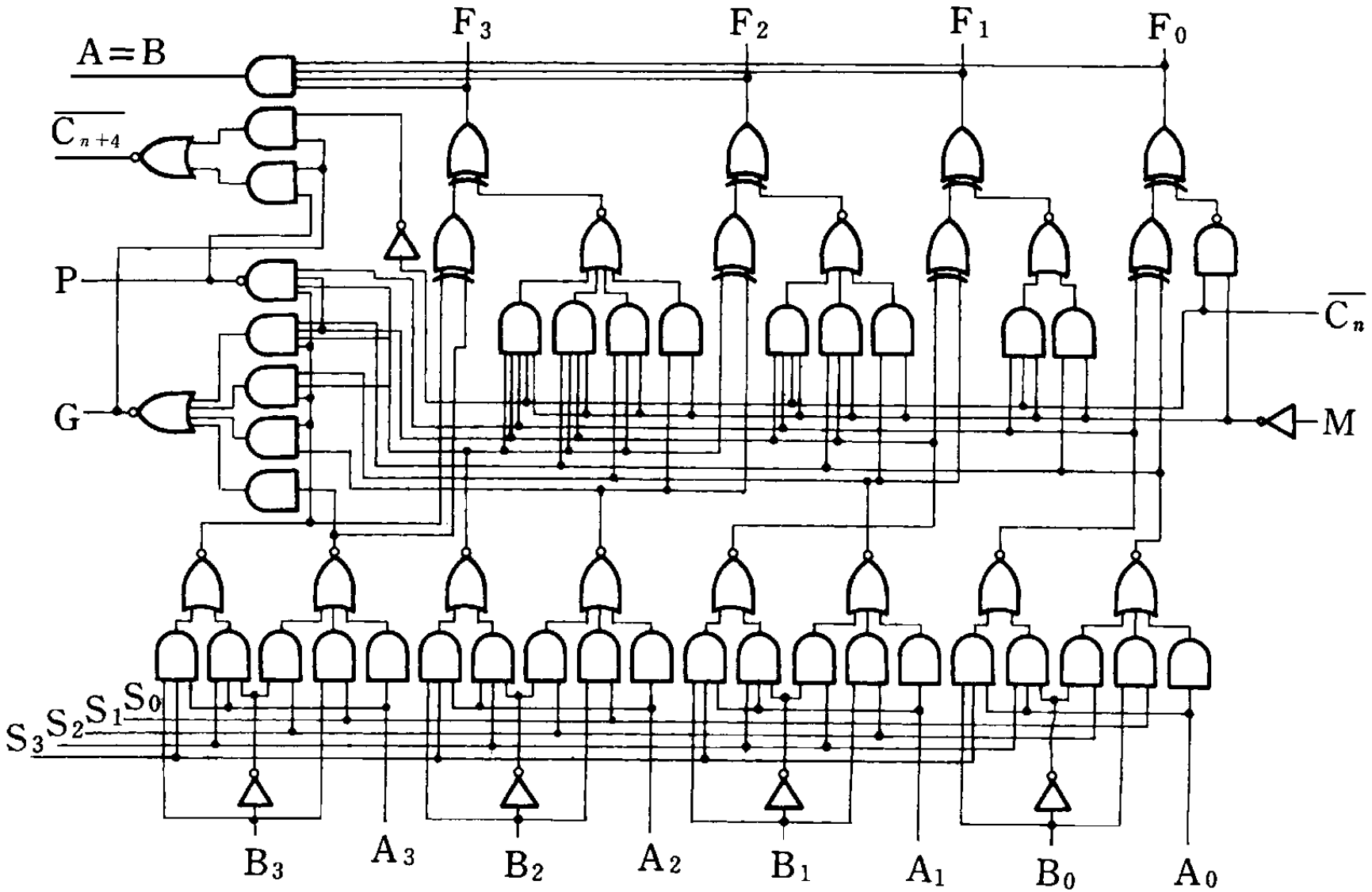
  器件： 74181

多功能算术/逻辑运算单元ALU

- 具有正逻辑和负逻辑两种



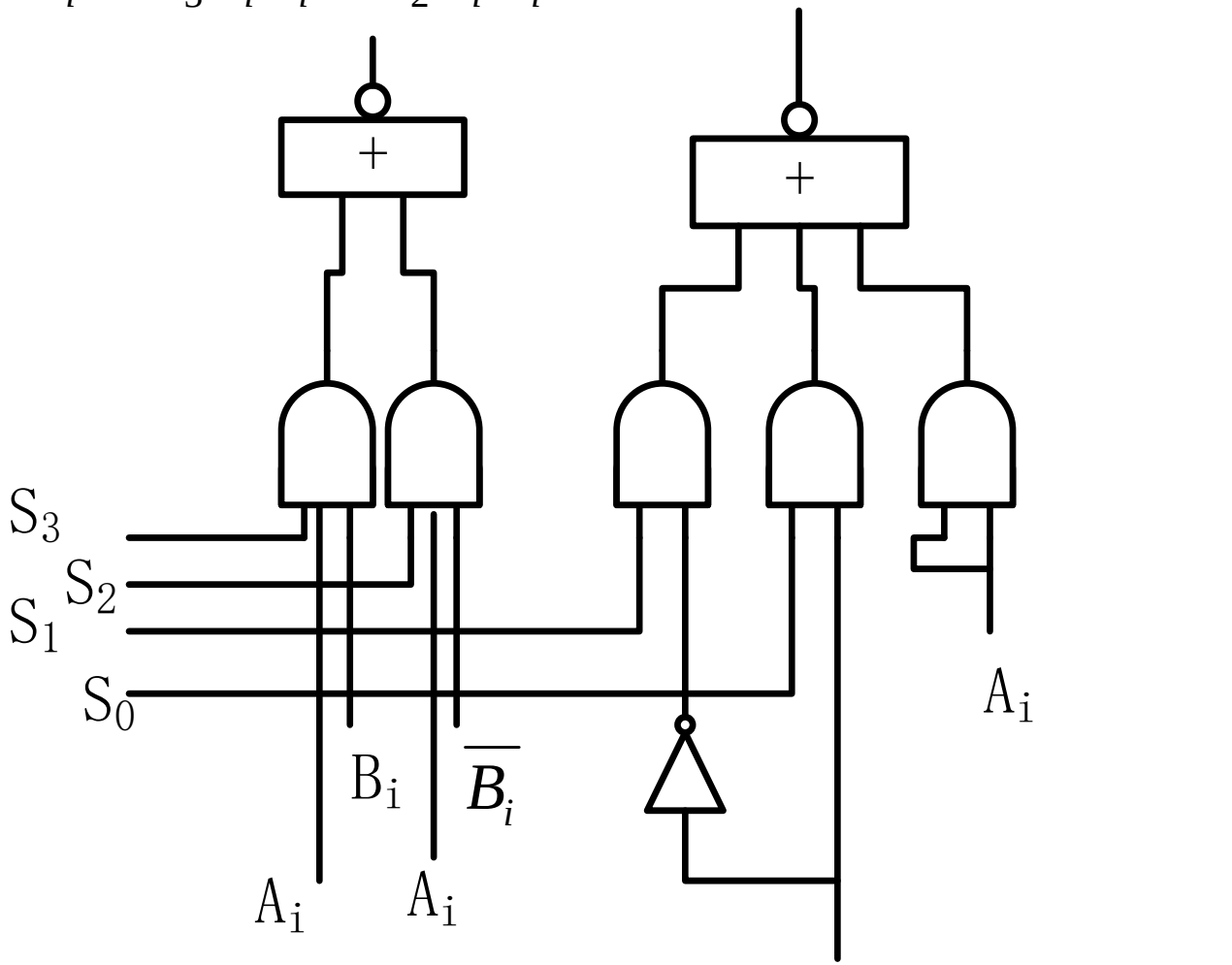
74181ALU逻辑图（总体）



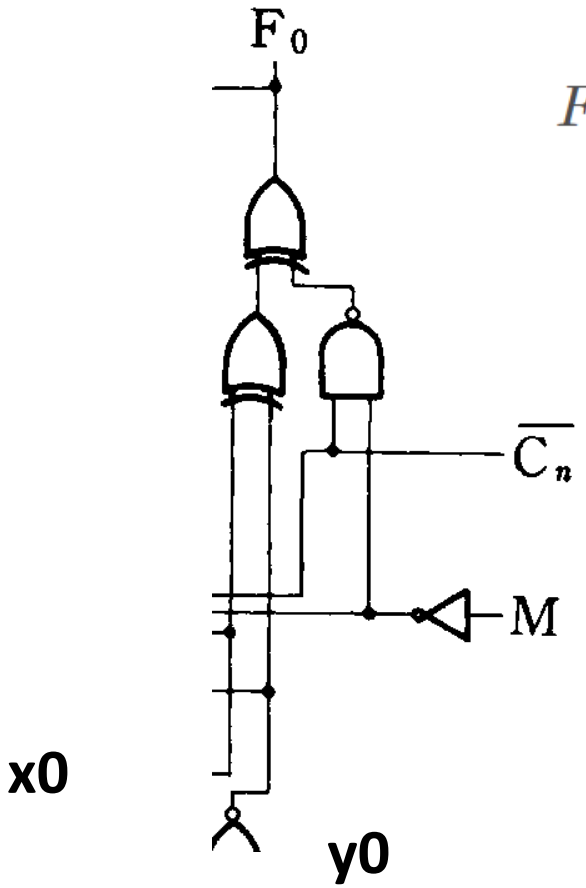
74181ALU逻辑图（1）

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

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



74181ALU逻辑图（2）

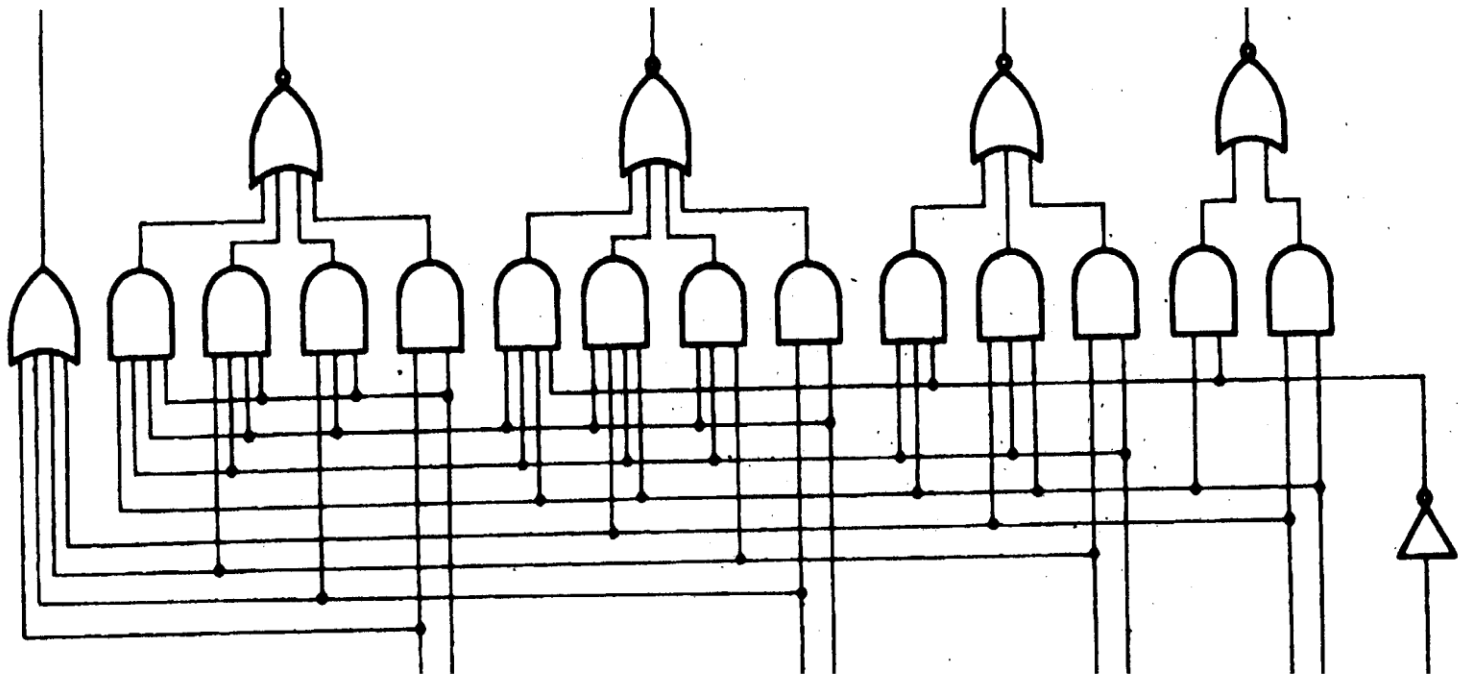


$$F_0 = Y_0 \otimes X_0 \otimes C_n$$

74181ALU逻辑图（3）

$$G = Y_3 + Y_2X_3 + Y_1X_2X_3 + Y_0X_1X_2X_3$$

$$P = X_0X_1X_2X_3$$



74181—ALU集成电路芯片

功能表				
工作方式选择输入 $S_3S_2S_1S_0$	负逻辑输入或输出		正逻辑输入或输出	
	逻辑运算 (M=1)	算术运算 (M=0) (C_{-1}=0)	逻辑运算 (M=1)	算术运算 (M=0) (C_{-1}=1)
0000	$\overline{A}$	A 减 1	$\overline{A}$	A
0001	$\overline{AB}$	AB 减 1	$\overline{A+B}$	$A+B$
0010	$\overline{A}+B$	$A\overline{B}$ 减 1	$\overline{A}B$	$A+\overline{B}$
0011	逻辑 1	减 1	逻辑 0	减 1
0100	$\overline{A+B}$	A 加 $(A+\overline{B})$	$\overline{AB}$	A 加 $A\overline{B}$
0101	$\overline{B}$	AB 加 $(A+\overline{B})$	$\overline{B}$	$(A+B)$ 加 $A\overline{B}$
0110	$\overline{A\oplus B}$	A 减 B 减 1	$A\oplus B$	A 减 B 减 1
0111	$A+\overline{B}$	$A+\overline{B}$	$A\overline{B}$	$A\overline{B}$ 减 1
1000	$\overline{A}B$	A 加 $(A+B)$	$\overline{A}+B$	A 加 AB
1001	$A\oplus B$	A 加 B	$\overline{A\oplus B}$	A 加 B
1010	B	$A\overline{B}$ 加 $(A+B)$	B	$(A+\overline{B})$ 加 AB
1011	$A+B$	$A+B$	AB	AB 减 1
1100	逻辑 0	A 加 A^*	逻辑 1	A 加 A^*
1101	$A\overline{B}$	AB 加 A	$A+\overline{B}$	$(A+B)$ 加 A
1110	AB	$A\overline{B}$ 加 A	$A+B$	$(A+\overline{B})$ 加 A
1111	A	A	A	A 减 1

(1) 1=高电平；0=低电平；(2) *表示每一位均移到下一个更高位，即 $A^*=2A$

注意：ALU为组合逻辑电路，因此实际应用ALU时，其输入端口A和B必须与锁存器相连，而且在运算的过程中锁存器的内容是不变的。其输出也必须送至寄存器中保存。

多功能算术/逻辑运算单元ALU

- **算术逻辑运算的实现（74181）**
  - M=L时，对进位信号没有影响，做算术运算
  - M=H时，进位门被封锁，做逻辑运算
- **说明：**
  - 74181执行正逻辑输入/输出方式的一组算术运算和逻辑运算和负逻辑输入/输出方式的一组算术运算和逻辑运算是等效的
  - A=B端可以判断两个数是否相等

多功能算术/逻辑运算单元ALU

- 设计16位ALU

- $C_{n+x}=G_0+P_0C_n$ $C_{n+y}=G_1+P_1C_{n+x}$
- $C_{n+z}=G_2+P_2C_{n+y}$ $C_{n+4}=G_3+P_3C_{n+z}$
- 片内先行进位,片间串行进位

多功能算术/逻辑运算单元ALU

- 随着n的增大，其优势便很快减弱
  - 例如，n=64，4位分组，共为16组
  - 组间有16位串行进位，每组延时为6T，所以还需经16*6T才能产生全部进位，显然进位时间太长

- **如果能使组间进位也同时产生，必然会更大地提高进位速度，这就是组内、组间均为并行进位的方案**

先行进位 ALU

- **两级先行进位的ALU**

  4片（组）的先行进位逻辑

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

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

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

  $=G_2+G_1P_2+G_0P_1P_2+P_0P_1P_2C_n$

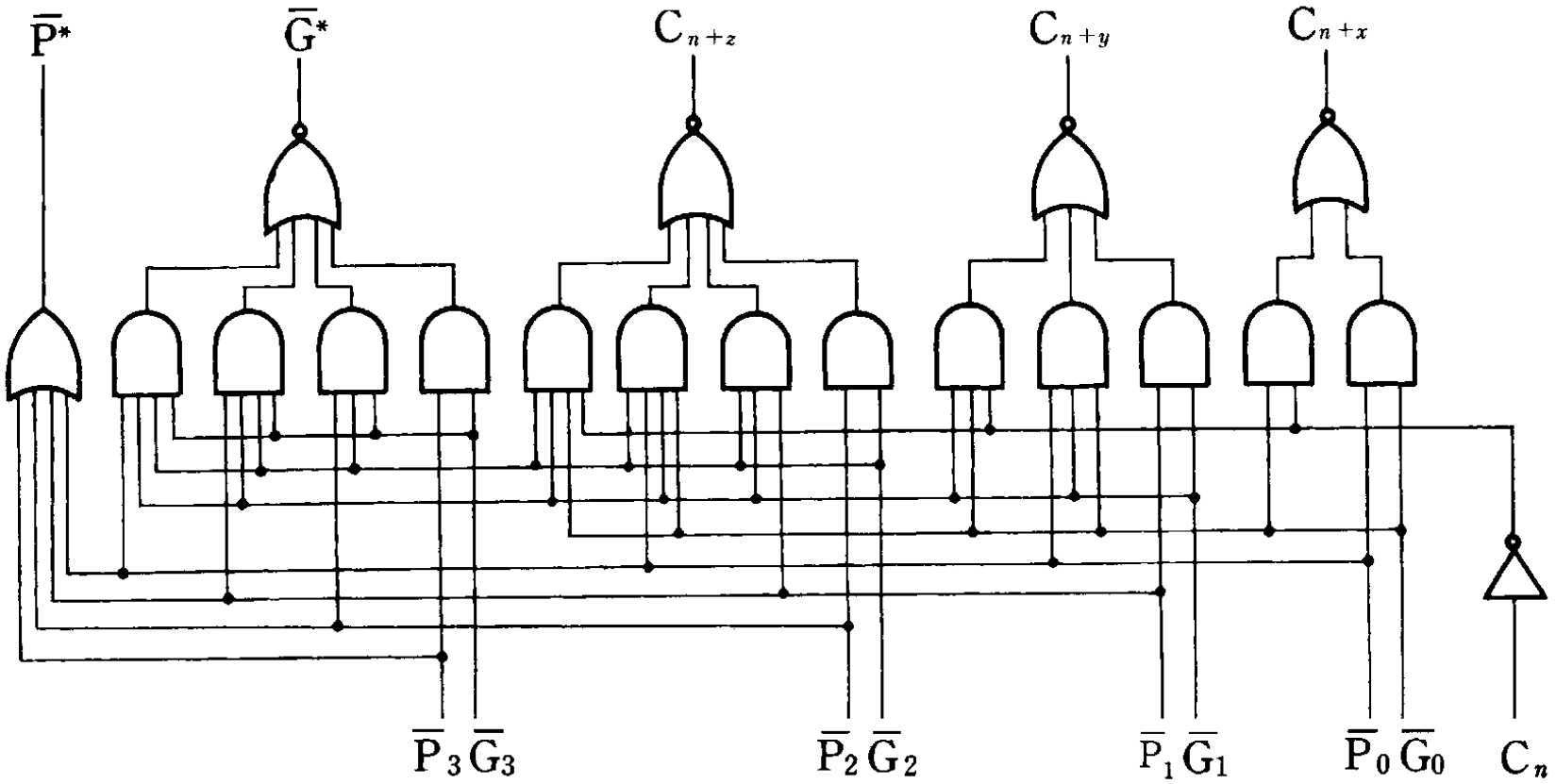
  $C_{n+4}^{*}=G_3+P_3C_{n+z}$

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

  $=G^*+P^*C_n$

  - G*为成组先行进位发生输出
  - P*为成组先行进位传送输出

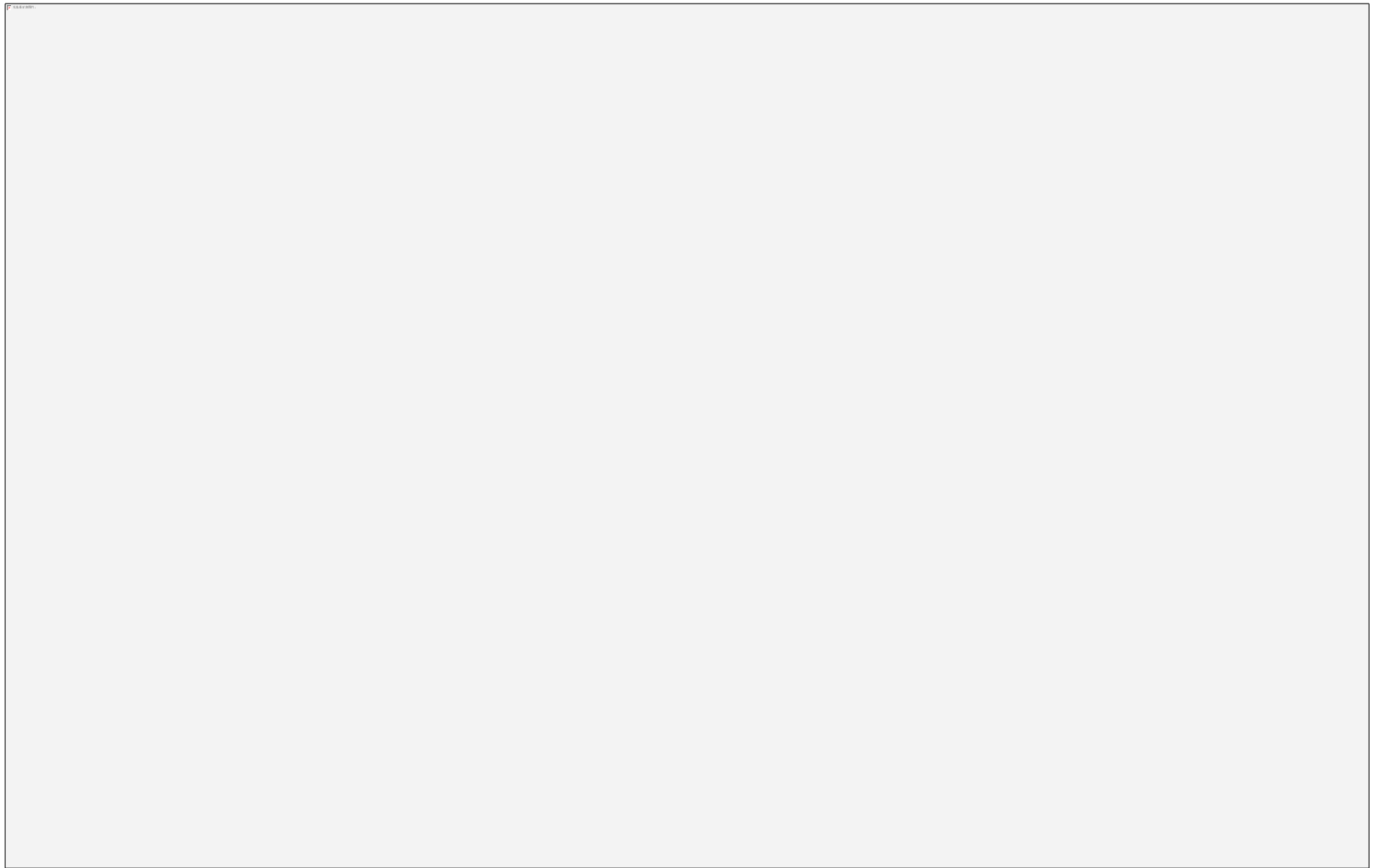
成组先行进位部件CLA的逻辑图74182



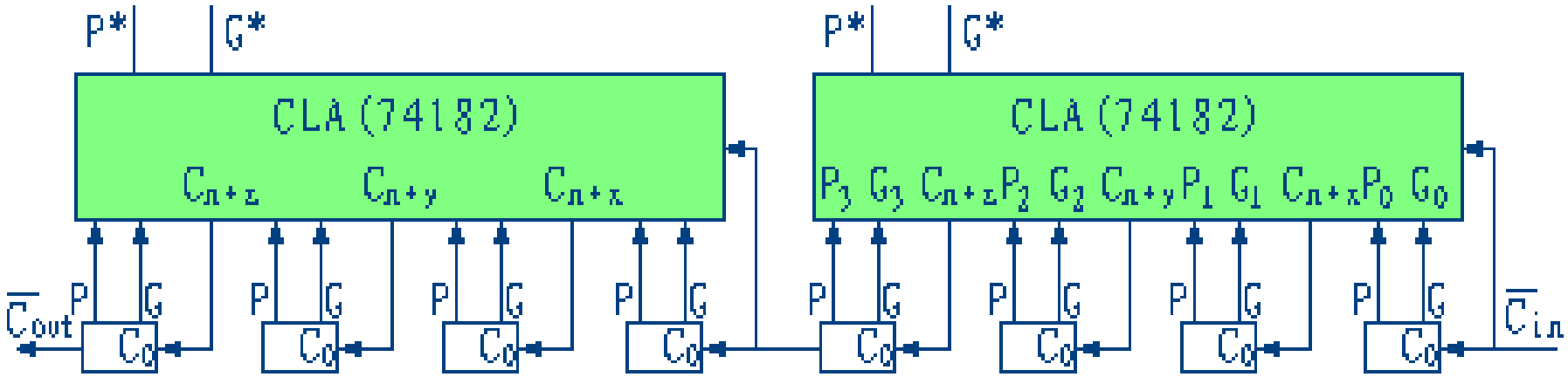
先行进位ALU

设计16位先行进位ALU

内先行进位，片间先行进位.



32位ALU逻辑方框图

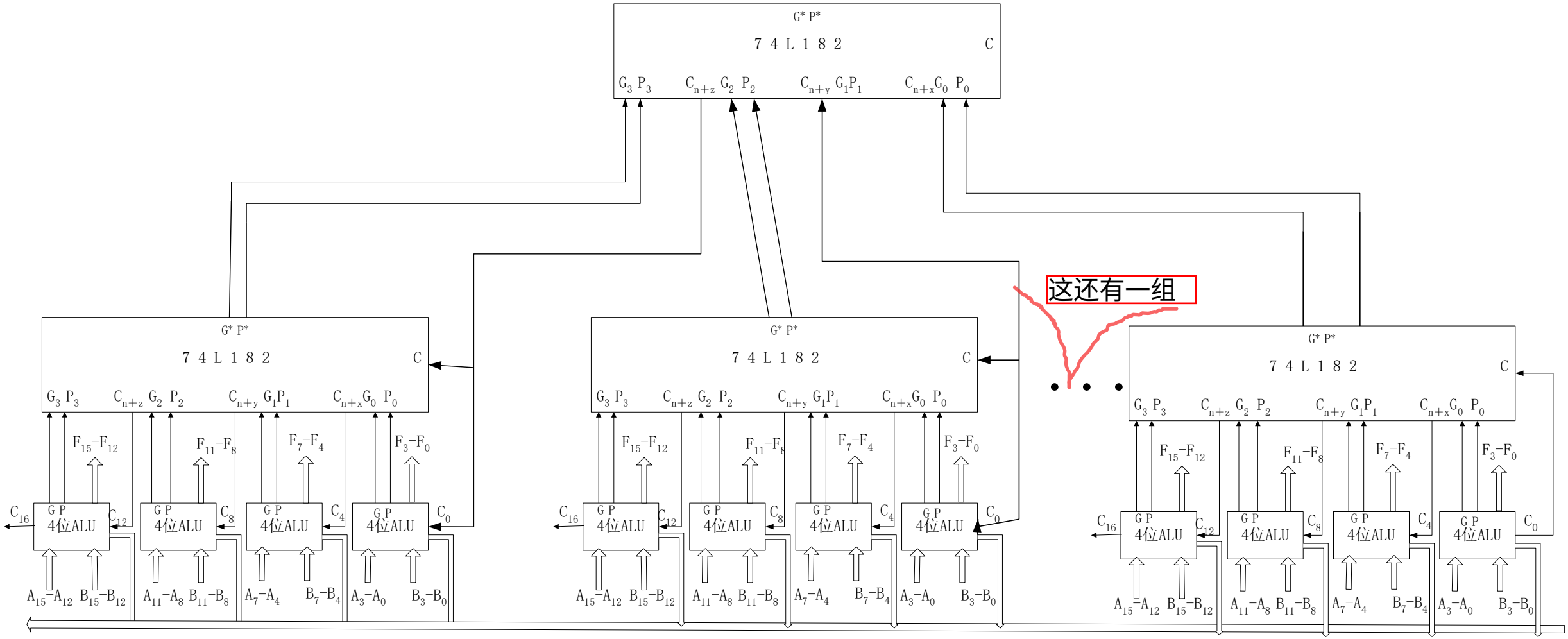


2个74L182

8个4位ALU74L181

组内并行进位，组间串行进位

64位具有全先行进位的ALU



两级CLA（74182）级联组成

课堂提问

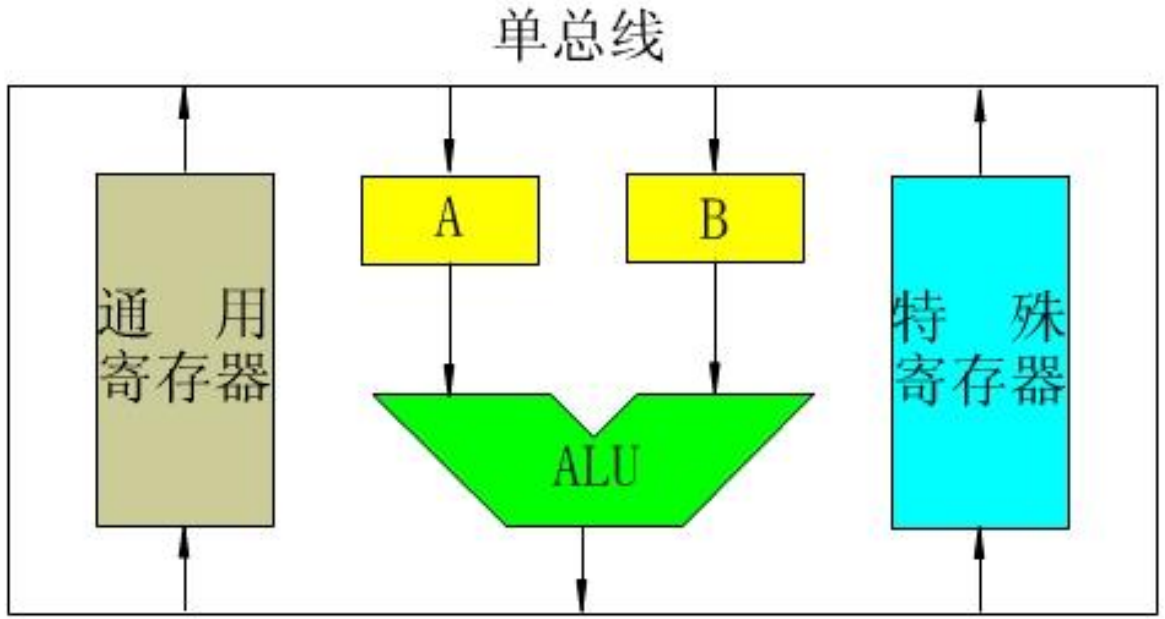
- 构造32位全先行进位的ALU，需要74181ALU和74182CLA各多少片？
- 构造64位全先行进位的ALU，需要74181ALU和74182CLA各多少片？
- 构造64位组内先行进位，组间串行进位的ALU，需要74181ALU和74182CLA各多少片？
- 构造64位片间串行进位的ALU，需要74181ALU和74182CLA各多少片？

内部总线

- **内部总线**
  - 机器内部各部份数据传送频繁，可以把寄存器间的数据传送通路加以归并，组成总线结构。
  - 分类
    - 所处位置
      - 内部总线（CPU内）
      - 外部总线（系统总线）
    - 逻辑结构
      - 单向传送总线
      - 双向传送总线

定点运算器的基本结构

单总线结构的运算器



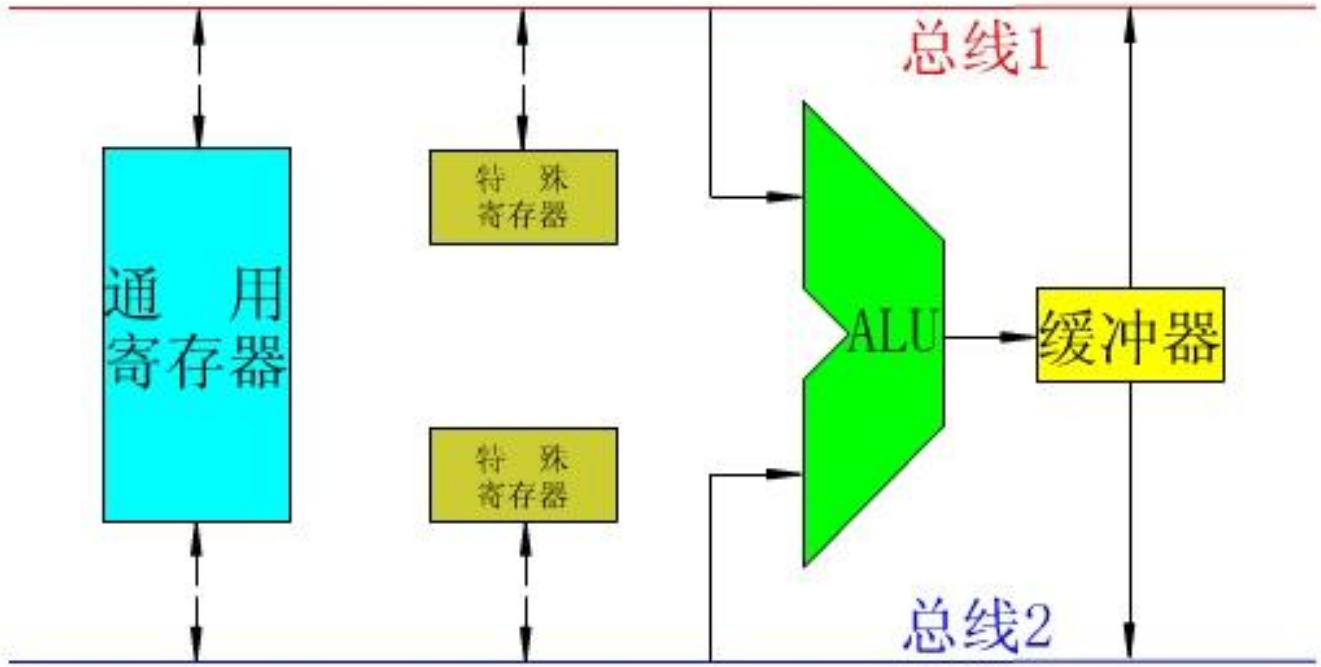
（a）单总线结构的运算器

定点运算器的基本结构

- 单总线结构的ALU
  - 两个操作数分时经总线进入锁存器A和B，运算结果也通过单总线送回
  - 特点：控制电路简单，操作速度比较慢

定点运算器的基本结构

双总线结构的运算器



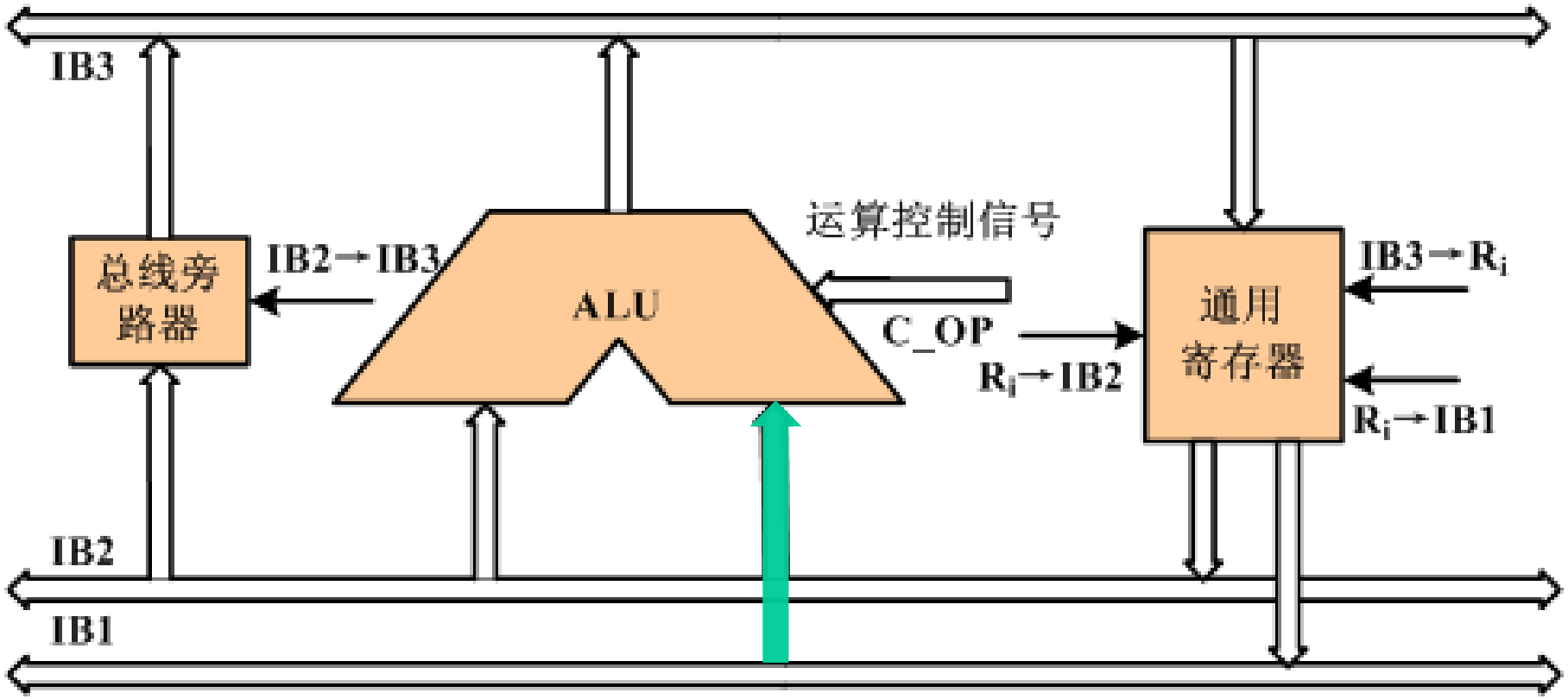
（b）双总线结构的运算器

定点运算器的基本结构

- 双总线结构的ALU
  - 两个操作数通过各自的总线送加法器运算，运算结果通过其中一总线送回。
  - 特点：操作速度较快。
    - 例如：加法操作

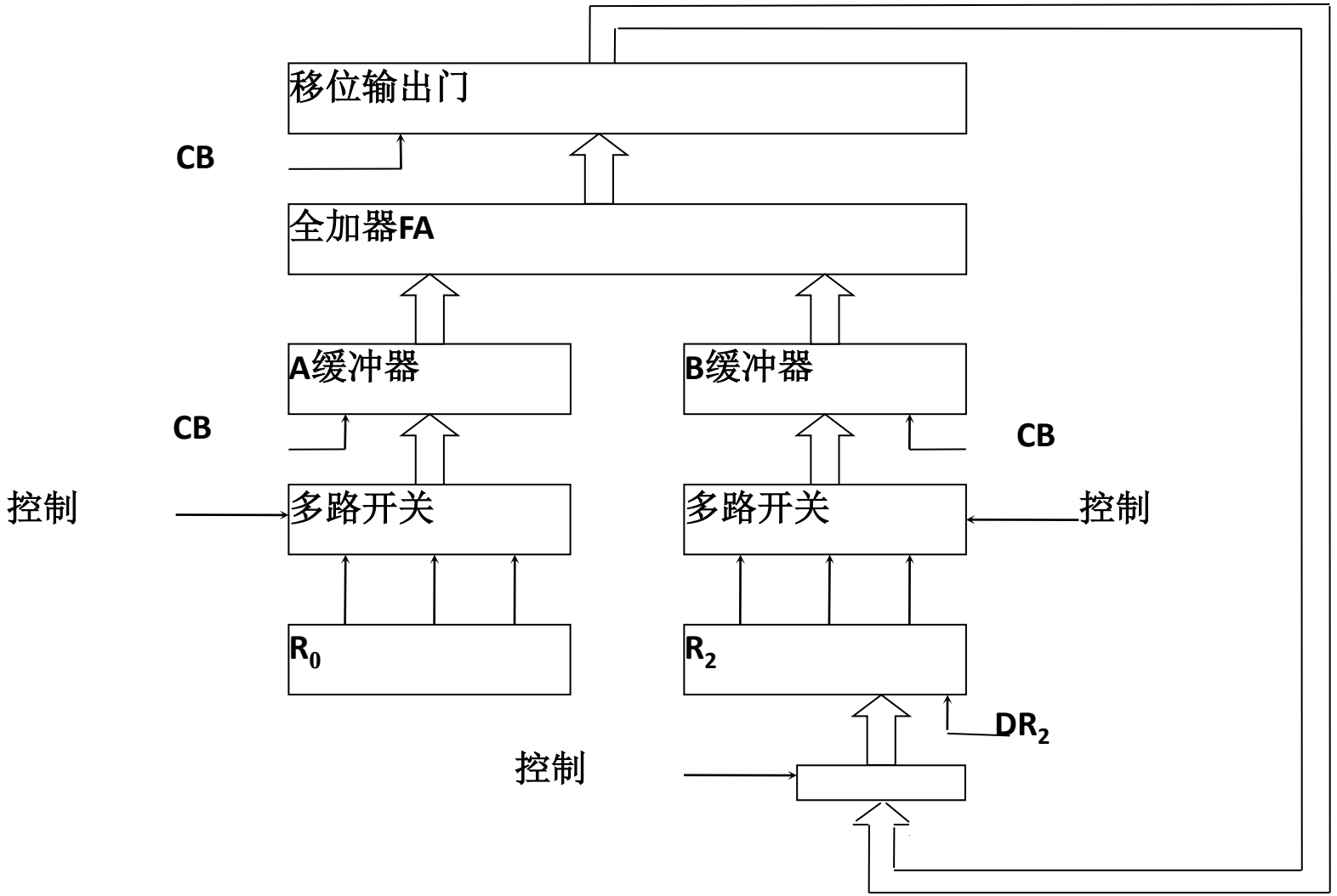
定点运算器的基本结构

三总线结构的运算器



定点运算器的基本结构

- **三总线结构的运算器**
  - 两个操作数和操作结果通过各自的总线传送
    - 例如：加法操作
  - 总线旁路器：使传送操作经它直接输出，速度快
  - 特点:操作速度快,但所需总线多



数据通路示意图