第三章　运算方法与运算器

- 运算器用于数值运算及加工处理数据
- 它是由CPU中的算术逻辑单元(ALU)、通用寄存器(GR)等部件构成
- 运算器的结构取决于指令系统、数据的表示方法、运算方法及所选用的硬件。

3.1 定点数运算

3.1.1 加减运算

1. 加减运算方法

- 补码加法

  补码加法的运算法则为：

  $$[X + Y]_{补} = [X]_{补} + [Y]_{补}$$

- 补码减法

  补码减法的运算法则为：

  $$[X - Y]_{补} = [X]_{补} + [-Y]_{补} = [X]_{补} + [[Y]_{补}]_{求补}$$

复习：

- **求补运算**

$[X]_{补}$ ⟷（符号位和数值位，按位取反，末位加一）$[-X]_{补}$

- **补码与真值、原码之间的相互转换**

当真值 X<0 时：真值 X ⟷（符号“－”⟷ 1，数值按位取反，末位加一）$[X]_{补}$

当真值 X<0 时：$[X]_{原}$ ⟷（符号位“1”不变，数值：按位取反，末位加一）$[X]_{补}$

定点数加减运算

例3.1 若两个定点整数63和35，利用补码加法求 $63 + 35 = ?$

解：根据题意，用8位二进制补码表示63和35：

$$[63]_{补} = 00111111$$

$$[35]_{补} = 00100011$$

则 $[63 + 35]_{补} = 01100010$

定点数加减运算

例3.2 若两个定点整数-63和-35，利用补码加法求-63 + (-35) = ？

解：根据题意，用8位二进制补码表示-63和-35：

$$[-63]_{补} = 11000001$$

$$[-35]_{补} = 11011101$$

则 $[-63 + (-35)]_{补} = 10011110$

定点数加减运算

例3.3 若两个定点整数63和35，利用补码减法求63 − 35 = ?

解：根据题意，用8位二进制补码表示63和35:

$[63]_{补} = 00111111$

$[35]_{补} = 00100011$

而$[63 - 35]_{补} = [63]_{补} + [-35]_{补}$；

同时，$[-35]_{补} = [[35]_{补}]_{求补} = 11011101$，从而求出：

$$\begin{array}{r} 00111111 \\ +\ 11011101 \\ \hline 100011100 \end{array}$$

得到$[63 - 35]_{补} = 00011100$。

- 补码加减运算规则

① 参加运算的操作数用补码表示；

② 符号位参加运算；

③ 若进行相加，则两个数的补码直接相加；

若进行相减运算，则对减数求补（连同符号位一起变反加1）后与被减数相加；

④ 运算结果用补码表示。

2. 溢出判断

- **溢出**

例3.4 若两个定点整数63和85，利用补码加法求63 + 85 = ?

解：根据题意，若用8位二进制补码表示63和85:

$[63]_{补}$ = 00111111

$[85]_{补}$ = 01010101

```
  00111111
+ 01010101
----------
  10010100
```

- 两个正数（63和85）相加的结果变成一个负数（符号位为1）。出现这种错误结果是由于在相加的过程中产生了溢出。
- 原因就在于运算的结果超出了所规定的数值范围。

例3.6 设负整数X= -1111000，Y= -10010，若用8位补码表示，则$[X]_{补}$=10001000，$[Y]_{补}$=11101110，求$[X+Y]_{补}$。

解：计算$[X]_{补}+[Y]_{补}$

$$\begin{array}{r} 1\ 0001000 \\ +\ \ 1\ 1101110 \\ \hline 0\ 1110110 \end{array}$$

两个负数相加，结果为一个正数，显然也是错误的。

- 只有当两个同符号的数相加（或者是相异符号数相减）时，运算结果才有可能溢出。而在异符号的数相加（或者是同符号数相减）时，永远不会产生溢出。

- 只要运算结果超出所能表示的数据范围，就会发生溢出。发生溢出时，运算结果肯定是错误的。只要发现运算结果产生溢出，就必须采取措施防止溢出发生。最简单有效的方法就是增加补码的二进制编码长度。

● 溢出的判定

(1) 双符号位判决法

- ✓ 若补码采用两位表示符号，即00表示正号、11表示负号，一旦发生溢出，则两个符号位就一定不一致，利用判别两个符号位是否一致便可以判定是否发生了溢出。
- ✓ 若运算结果两符号分别用S_2S_1表示，则判别溢出的逻辑表示式为：

$$VF = S_2 \oplus S_1$$

例 设两正整数X=+1000001，Y= +1000011，若用双符号位的8位补码表示，则 $[X]_{补}$=00 1000001，$[Y]_{补}$=00 1000011，求$[X+Y]_{补}$。

解：计算$[X]_{补}+[Y]_{补}$

$$\begin{array}{r} 00\ 1000001 \\ +\ 00\ 1000011 \\ \hline 01\ 0000100 \end{array}$$

式中，由于结果的S2和S1 不一致，

VF＝S2 ⊕ S1＝1，溢出发生。

例 $x=+0.1100,\ y=+0.1000,$ 求 $x+y$。

解:

$$[x]_{补}=00.1100,\quad [y]_{补}=00.1000$$

$$\begin{array}{ll} & [x]_{补} \quad 00.1100 \\ + & [y]_{补} \quad 00.1000 \\ \hline & \quad 01.0100 \end{array}$$

两个符号位出现“01”，表示已溢出，即结果大于 +1。

例 $x=-0.1100,\ y=-0.1000,$ 求 $x+y$。

解:

$$[x]_{补}=11.0100,\quad [y]_{补}=11.1000$$

$$\begin{array}{rl} & [x]_{补} \quad 11.0100 \\ + & [y]_{补} \quad 11.1000 \\ \hline & \quad 10.1100 \end{array}$$

两个符号位出现“10”，表示已溢出，即结果小于−1。

- 溢出的判定

（2）进位判决法

若C_{n-1}为最高数值位向最高位（符号位）的进位，C_n表示符号位的进位，则判别溢出的逻辑表示式为：

$$VF = C_{n-1} \oplus C_n$$

（3）根据运算结果的符号位和进位标志判别

该方法适用于两同号数求和或异号数求差时判别溢出。溢出的逻辑表达式为：

$$VF = SF \oplus CF$$

例 设两正整数X=+1000001，Y= +1000011，采用8位补码表示，则 $[X]_{补}$=0 1000001，$[Y]_{补}$=0 1000011，求$[X+Y]_{补}$。

解：计算$[X]_{补}+[Y]_{补}$

$$\begin{array}{r} 0\ 1000001 \\ +\ 0\ 1000011 \\ \hline 1\ 0000100 \end{array}$$

式中，由于$C_{n-1}=1$，$C_n=0$，

$VF=C_{n-1}\oplus C_n=1$，溢出发生。

- 溢出的判定

（4）根据运算前后的符号位进行判别

若用Xs、Ys、Zs分别表示两个操作数及运算结果的符号位，当两同号数求和或异号数求差时，就有可能发生溢出。溢出是否发生可根据运算前后的符号位进行判别，其逻辑表达式为：

$$VF = Xs \cdot Ys \cdot \overline{Zs} + \overline{Xs} \cdot \overline{Ys} \cdot Zs$$

3. 一位全加器

设：一位全加器的输入为X_i和Y_i
低一位对该位的进位为C_i
全加器的结果和进位用Z_i和C_{i+1}表示

一位全加器逻辑表达式：

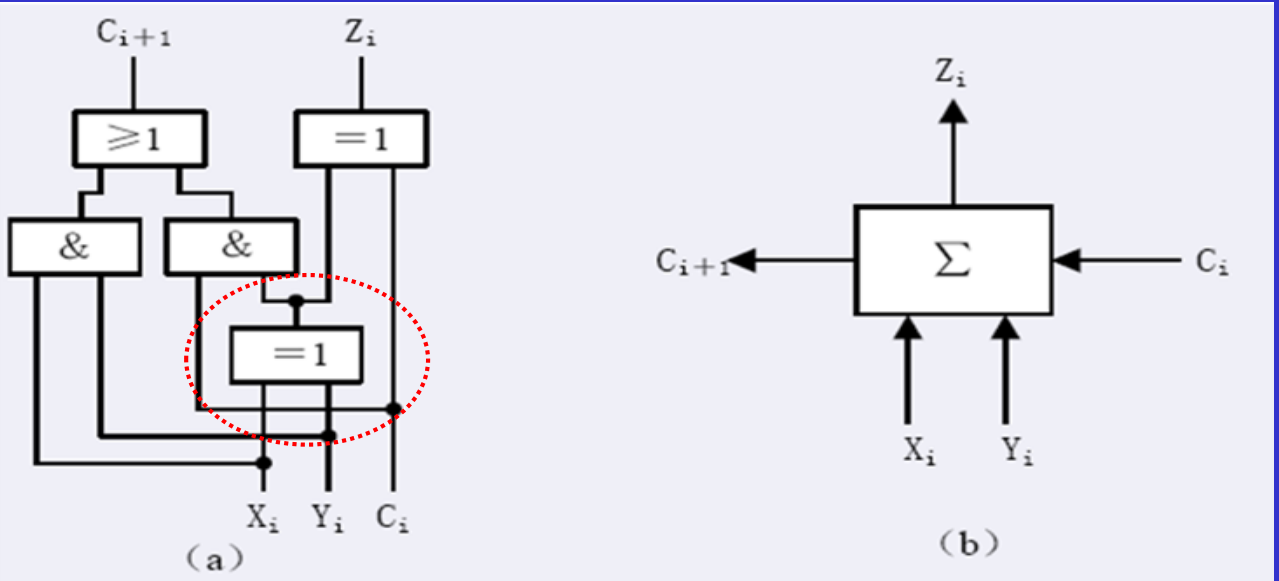
$$Z_i = X_i \oplus Y_i \oplus C_i$$

$$C_{i+1} = (X_i \cdot Y_i) + (X_i + Y_i) \cdot C_i$$

若令$G_i = X_i \cdot Y_i$，$P_i = X_i + Y_i$，则可写为：

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

本位进位函数　　进位传递函数



一位全加器逻辑及其框图

$$Z_i = X_i \oplus Y_i \oplus C_i$$

$$C_{i+1} = (X_i \cdot Y_i) + (X_i + Y_i) \cdot C_i$$

若令$G_i = X_i \cdot Y_i$，$P_i = X_i + Y_i$，则可写为：

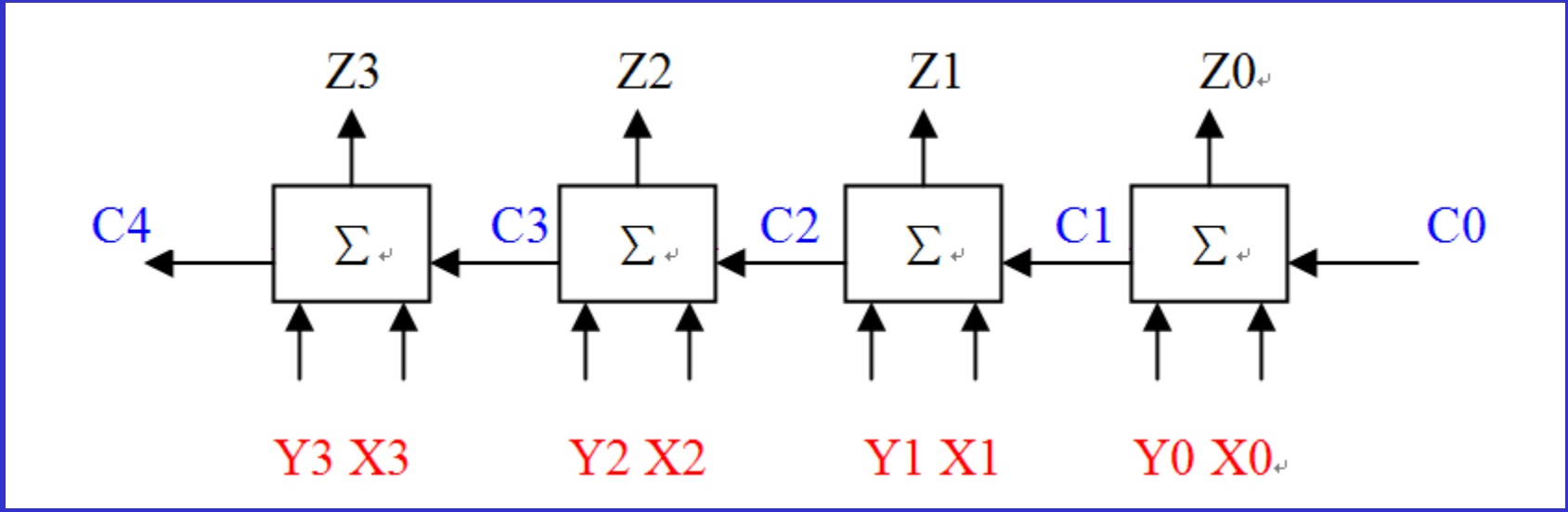
$$C_{i+1} = G_i + P_i \cdot C_i$$

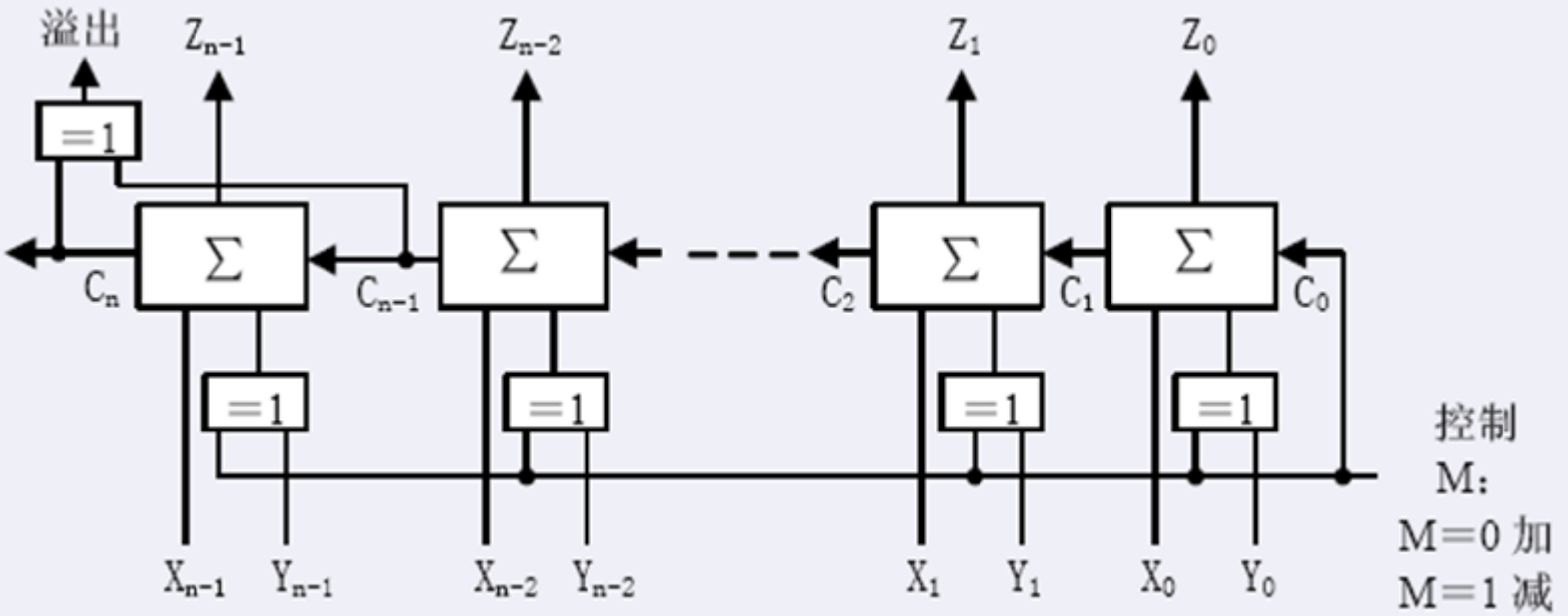
本位进位函数　　进位传递函数

Generation　　Propagate

4. n位加法器

（1）行波进位（串行进位）加法器





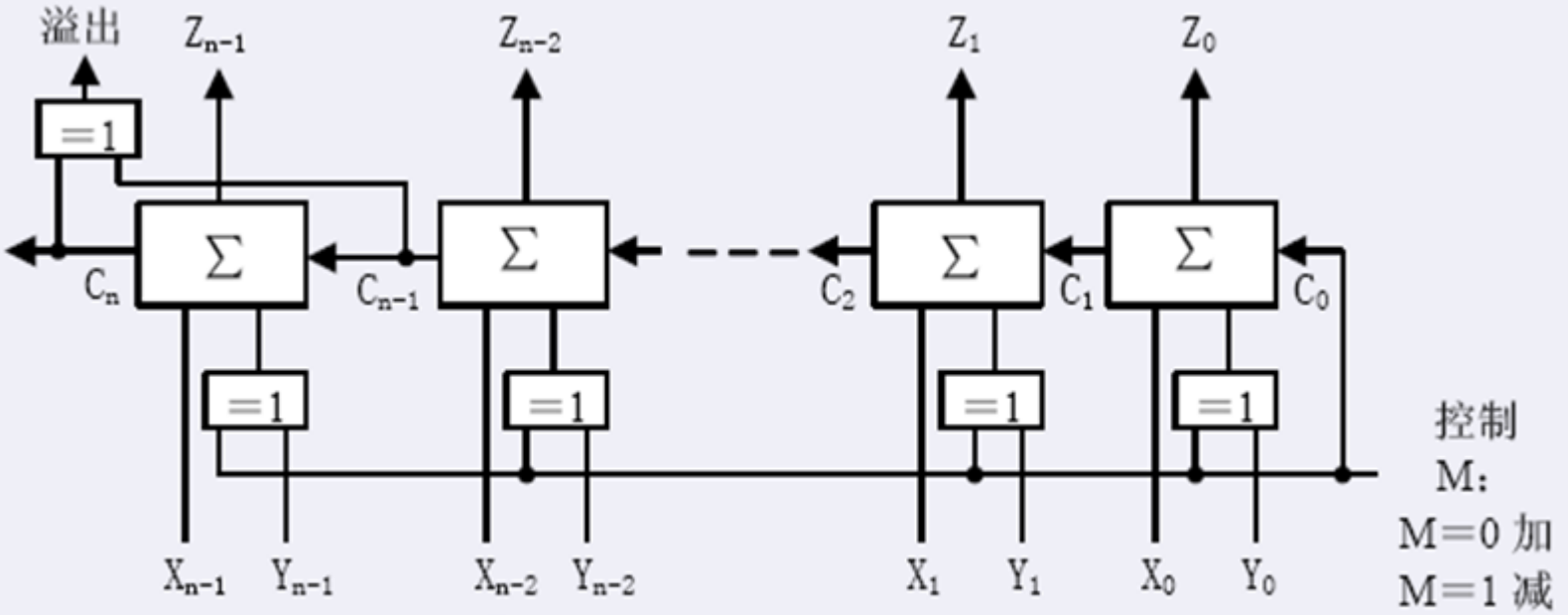
行波进位的n位加法/减法器

分析：　当 M=0 时，$Z = X + Y$

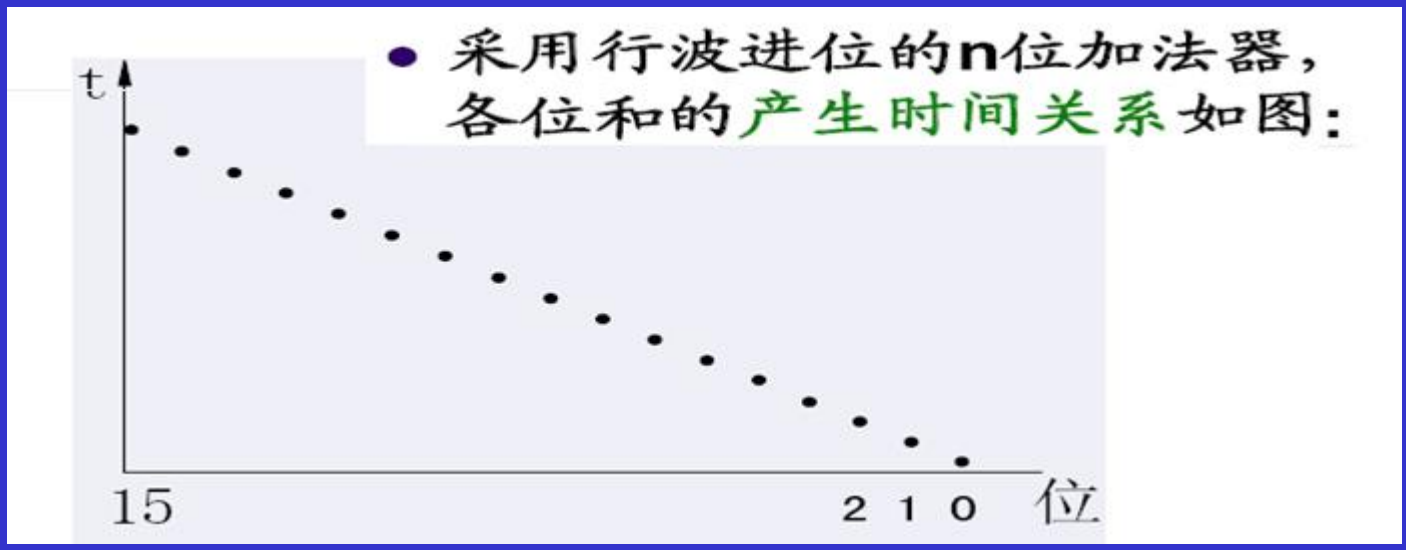
当 M=1 时，$Z = X + (\overline{Y} + 1)$

$= X - Y$

功能：　加/减法器



行波进位的n位加法/减法器



- 采用行波进位的n位加法器，各位和的产生时间关系如图：

(2) 先行进位加法器

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

从式中可知，只要有输入Xi和Yi就能求出G_i和P_i，在已知输入C_i的情况下，便可以获得C_{i+1}

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

$$C_{i+2} = G_{i+1} + P_{i+1} C_{i+1} = G_{i+1} + P_{i+1} G_i + P_{i+1} P_i C_i$$

$$C_{i+3} = G_{i+2} + P_{i+2} C_{i+2} = G_{i+2} + P_{i+2} G_{i+1} + P_{i+2} P_{i+1} G_i + P_{i+2} P_{i+1} P_i C_i$$

$$C_{i+4} = G_{i+3} + P_{i+3} C_{i+3}$$

$$= G_{i+3} + P_{i+3} G_{i+2} + P_{i+3} P_{i+2} G_{i+1} + P_{i+3} P_{i+2} P_{i+1} G_i + P_{i+3} P_{i+2} P_{i+1} P_i C_i$$

$$= G^*_{i+3} + P^*_{i+3} C_i$$

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

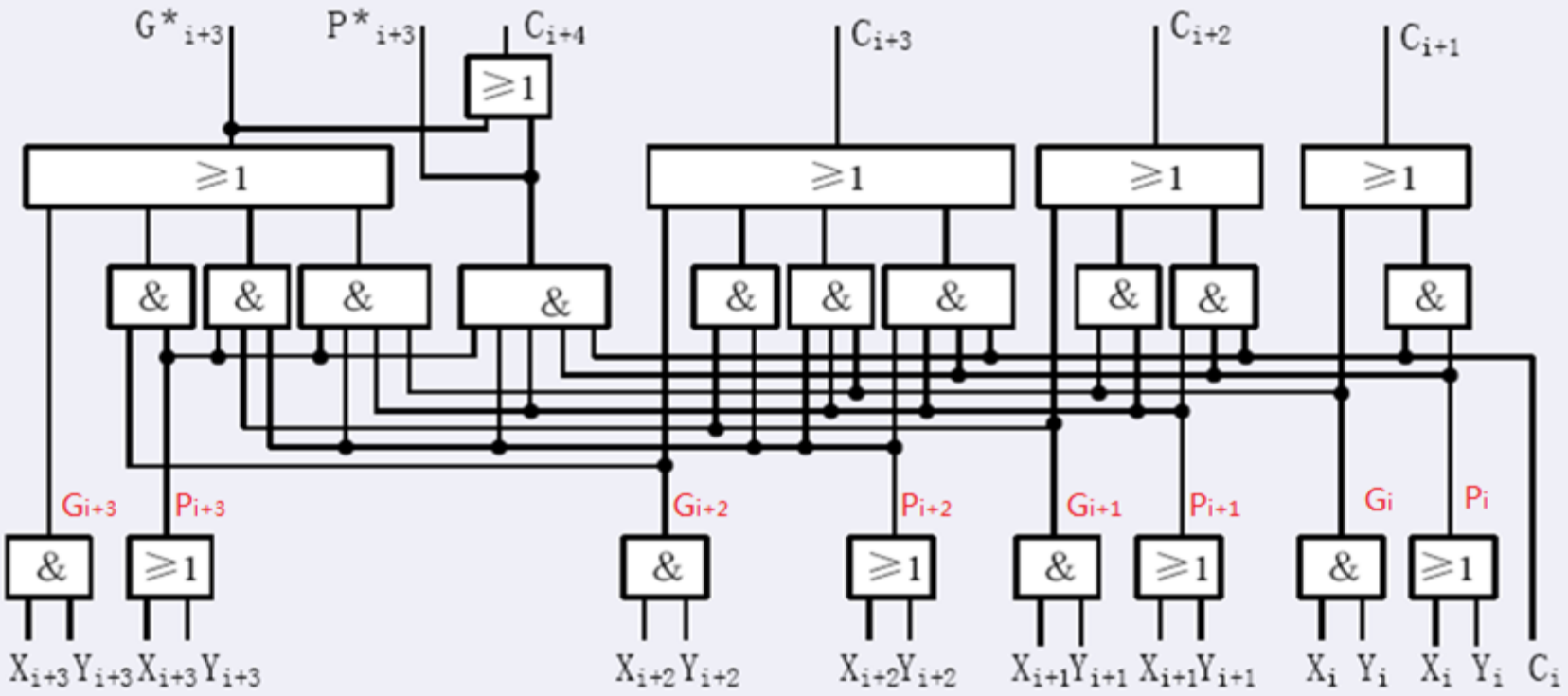
$$C_{i+2} = G_{i+1} + P_{i+1} C_{i+1} = G_{i+1} + P_{i+1} G_i + P_{i+1} P_i C_i$$

$$C_{i+3} = G_{i+2} + P_{i+2} C_{i+2} = G_{i+2} + P_{i+2} G_{i+1} + P_{i+2} P_{i+1} G_i + P_{i+2} P_{i+1} P_i C_i$$

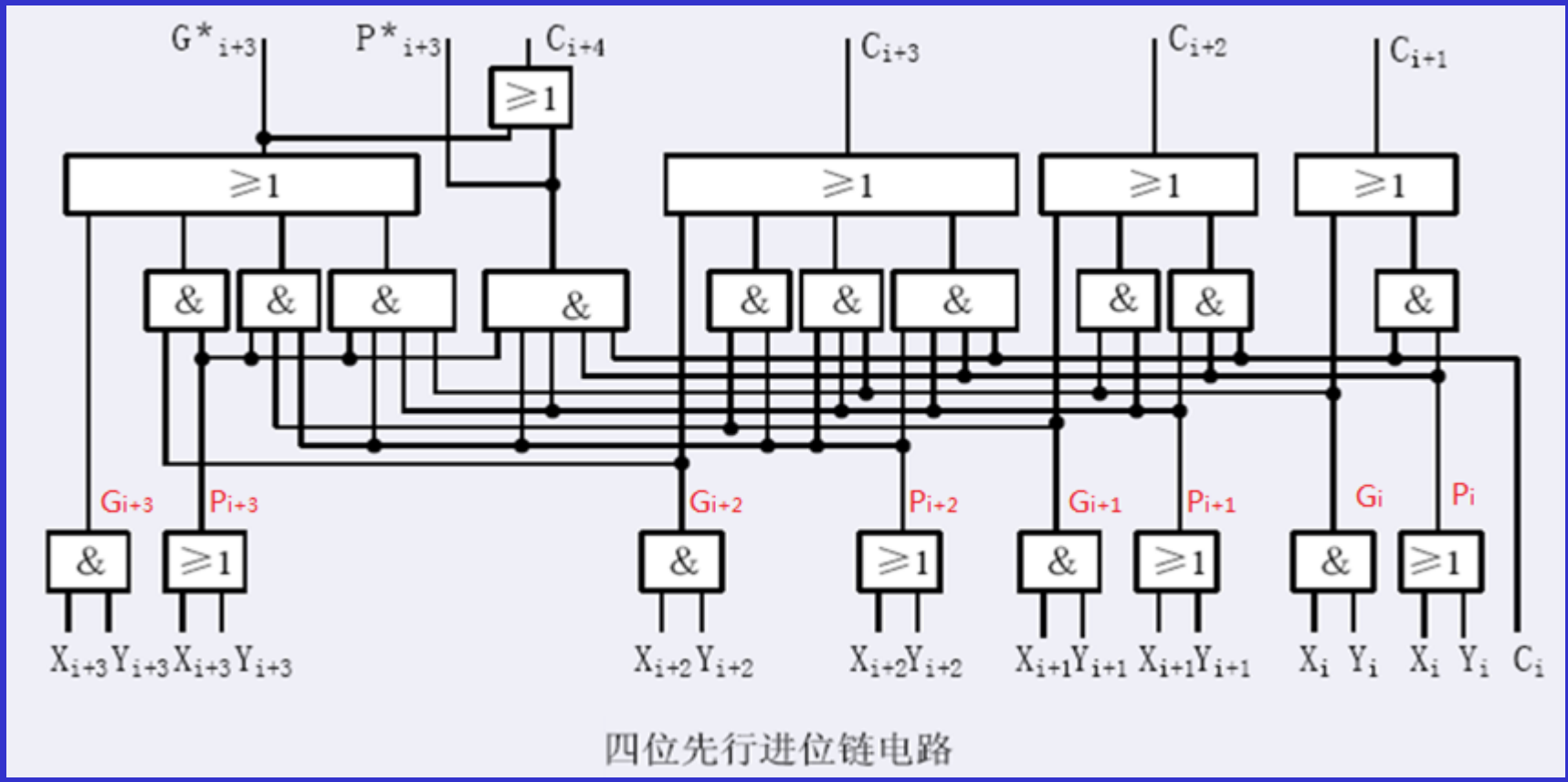
$$C_{i+4} = G_{i+3} + P_{i+3} C_{i+3}$$

$$= G_{i+3} + P_{i+3} G_{i+2} + P_{i+3} P_{i+2} G_{i+1} + P_{i+3} P_{i+2} P_{i+1} G_i + P_{i+3} P_{i+2} P_{i+1} P_i C_i$$

$$= G^*_{i+3} + P^*_{i+3} C_i$$

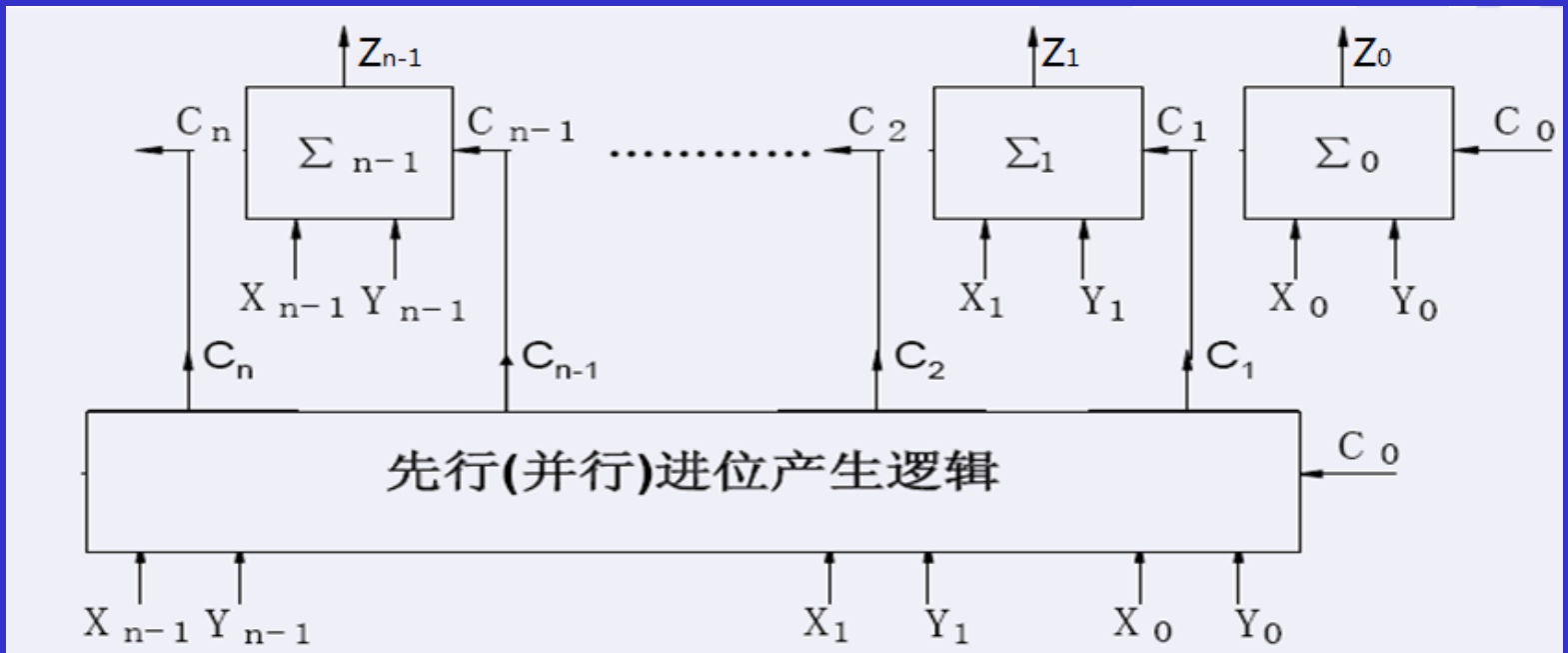


四位先行进位链电路



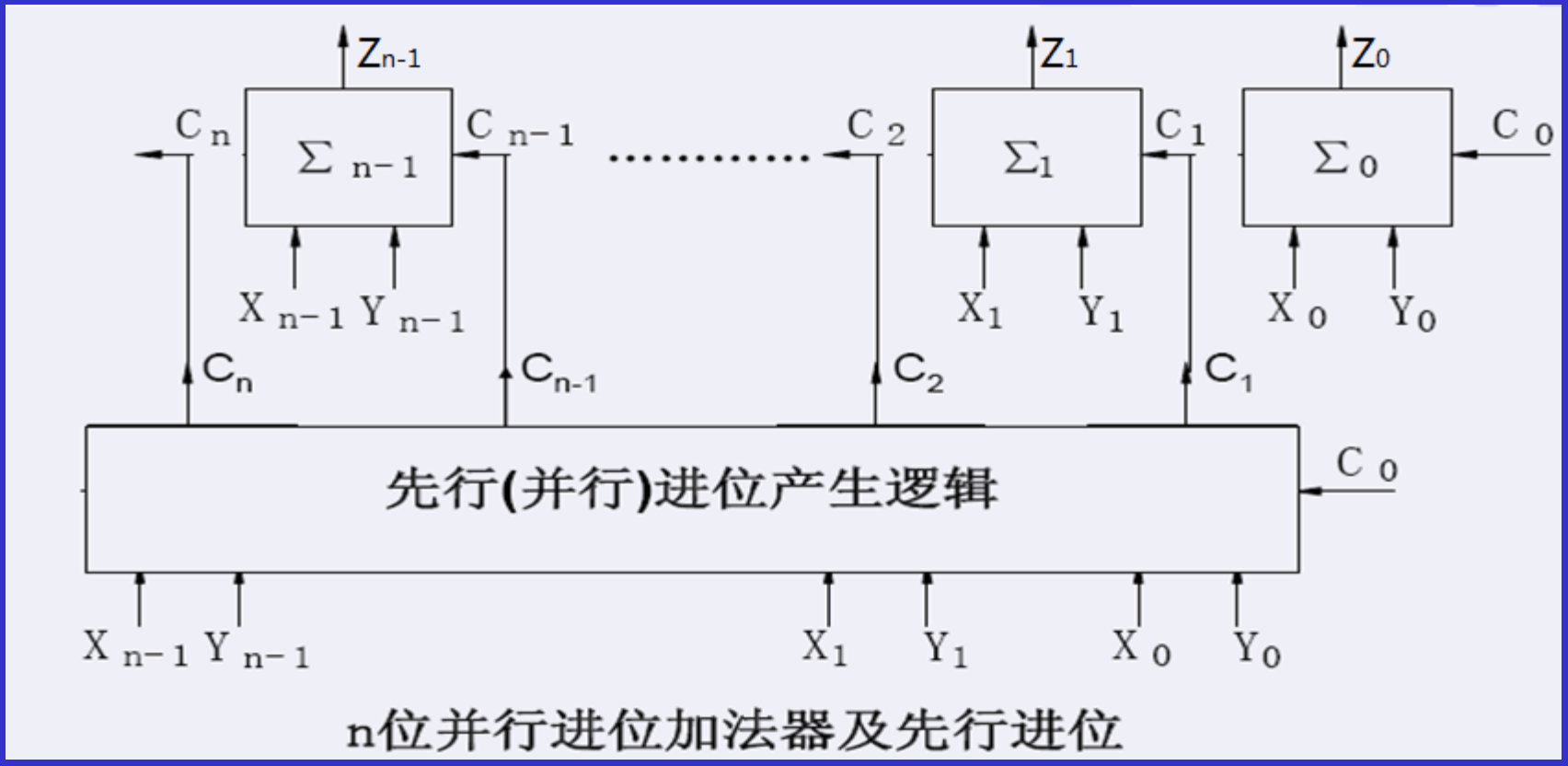
四位先行进位链电路

- 利用输入信号X_i、X_{i+1}、X_{i+2}、X_{i+3}和Y_i、Y_{i+1}、Y_{i+2}、Y_{i+3}以及C_i，通过与或逻辑电路的组合就可以同时将上面四个进位信号产生出来。



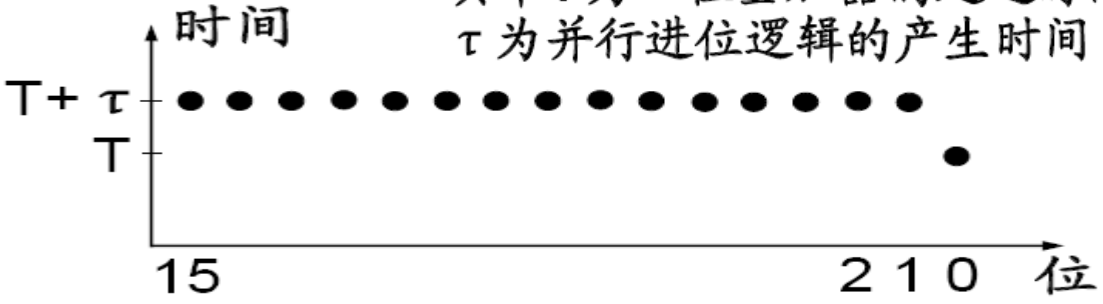
n位并行进位加法器及先行进位

- 将这些先行形成的进位信号并行地加到加法器上，则加法器就不必等待进位产生，从而大大地提高了加法器的速度。也就是说在进行加法运算之前，各位加法器所需要的进位已经产生出来。这将加快加法运算的速度，这就是先行进位的来由。

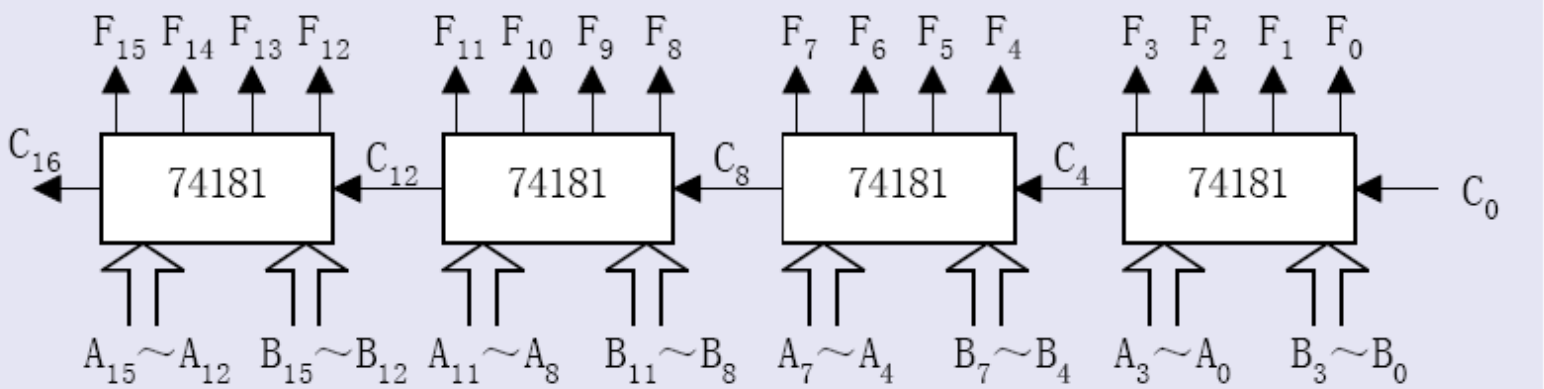


n位并行进位加法器及先行进位

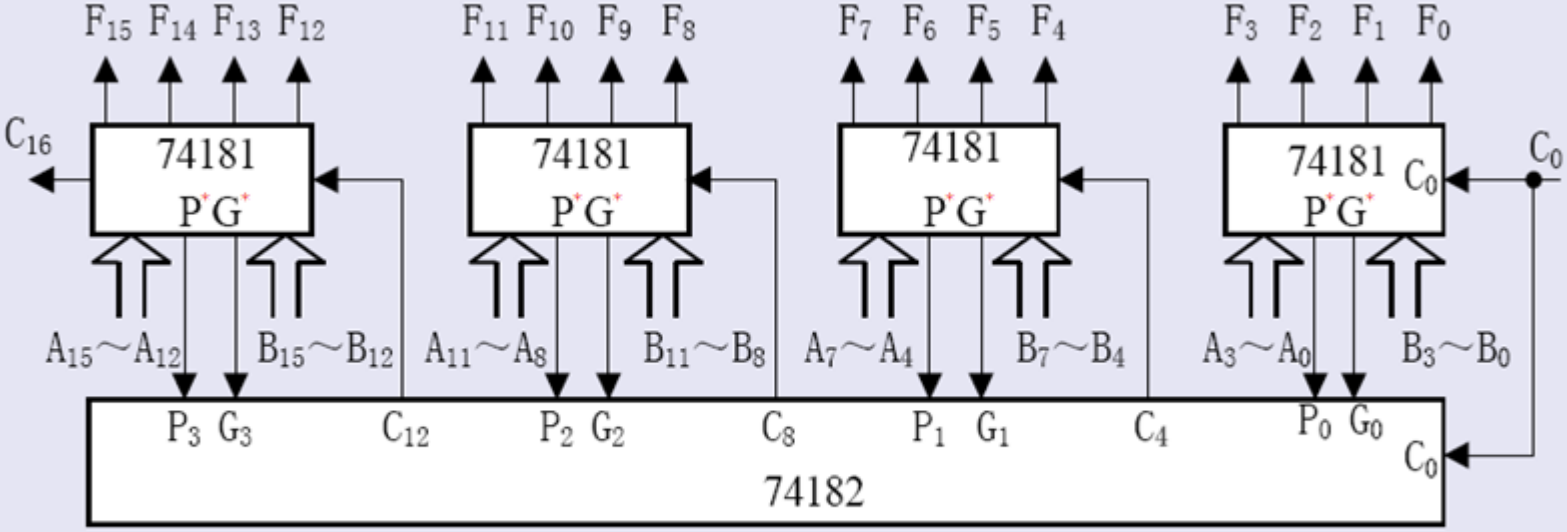
16位并行加法器各位和的产生时间如图
其中T为一位全加器的延迟时间，
τ为并行进位逻辑的产生时间



(3) 先行进位加法器的级联



由74181构成组间串行进位的16位ALU



由74181和74182构成组内组间均并行进位的16位ALU

5. 移码加减运算

$$[X]_{移} \pm [Y]_{移} = (2^{n-1}+X) \pm (2^{n-1}+Y) = 2^n + (X \pm Y)$$

$$= [X \pm Y]_{补} \longleftrightarrow [X \pm Y]_{移} \quad \text{MOD } 2^n$$

符号位取反

定点整数移码的加减运算法则：

①对两移码求和差时，首先对该两移码求和差；

②然后，对结果进行修正——将结果的符号取反。这样，就可以得到正确的结果。

6. BCD数加法器（课后学习）

- **8421 BCD码**

  8421 BCD码只利用了四位二进制编码的0000到1001这十种来表示十进制数的0到9。剩余的六种：1010，1011，1100，1101，1110，1111对用于表示十进制数来说是非法的。一旦在定义的BCD运算中出现这六种编码，结果一定是错误的。

- 8421 BCD码加法运算也可以用多位全加器实现，在运算过程中有可能产生上面提到的错误结果。

a:
$$\begin{array}{r} 0110 \\ +\ 0010 \\ \hline 1000 \end{array}$$

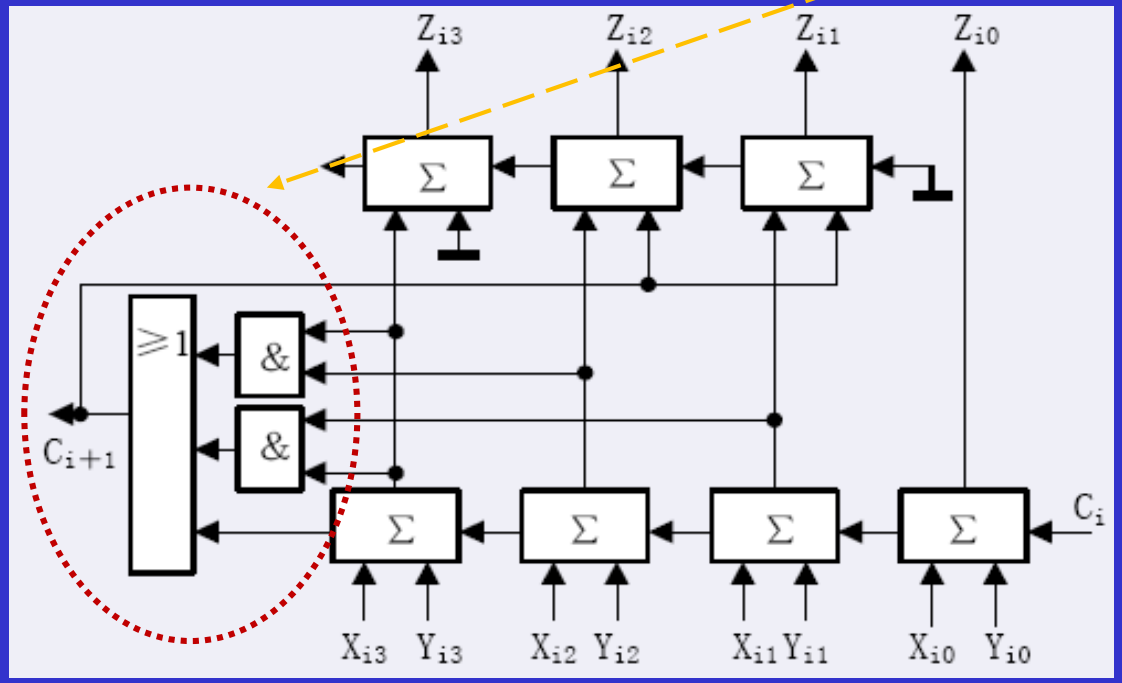
b:
$$\begin{array}{r} 0110 \\ +\ 0111 \\ \hline 1101 \end{array}$$

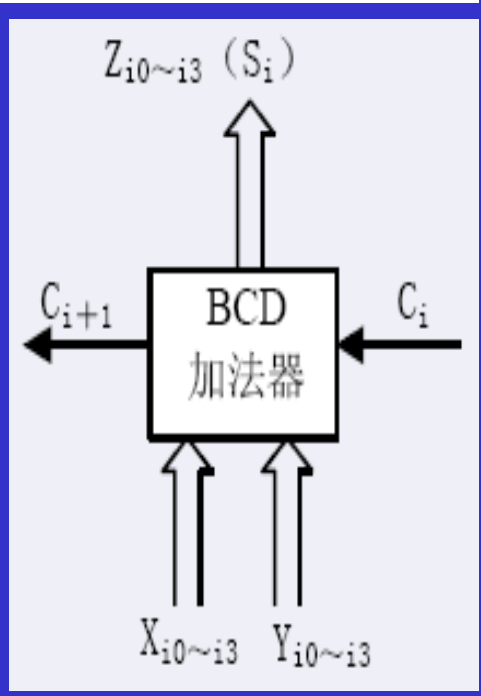
- 8421 BCD码加法运算也可以用多位全加器实现，在运算过程中有可能产生上面提到的错误结果。

a:	0110	b:	0110
	+ 0010		+ 0111
	1000		1101

- 校正

运算中低四位相加的结果＞9或有由bit3向bit4的进位，则结果加06H;





本章作业-1

第17（1）（2）、18（2）、19（2）题

注：本次作业与下次作业一起交