计算机组成原理

翁睿

哈尔滨工业大学

Part 3

第 6 章　计算机的运算方法

6.1 无符号数和有符号数

6.2 数的定点表示和浮点表示

6.3 定点运算

6.4 浮点四则运算

6.5 算术逻辑单元

乘法运算的实现方法

6.3

- **执行乘法运算子程序，实现乘法运算**
  - 零成本、Intel 8008/8080、RISCV32-I指令集
- **利用加法器多次累加，实现乘法运算**
  - 原码一位乘
  - 补码一位乘（Booth算法）
  - 成本低
- **设置专用乘法器硬件**
  - 成本高 （原码、补码乘法器）

乘法器性能提升小结

6.3

- 乘法的核心算法：**n个部分积累加**
- **Booth一位乘 → Booth两位乘**
  - 一位乘法：n个全加器，n^2个全加器时延，面积小 (Intel 8086)
  - 两位乘法： 减少部分积的个数，速度更快，增加额外电路
- **斜向进位阵列乘法器 → 华莱士树**
  - 斜向进位：（n^2-n）个全加器，n级全加器时延，面积大
  - 华莱士树： 更多全加器并行运算，$\log_2 n$ 级全加器时延，面积更大
- **主流乘法器**
  - **二位booth算法 + 华莱士树 + 流水**

特例：变量与常数的乘法

- 整数乘法比移位和加法运算慢很多
- 编译器在处理变量与常数相乘时，可能会用其它快速运算指令代替乘法运算，例如：

  x*20 → (x<<4)+(x<<2)　　乘法转换成了2次移位和1次加法

  x*15 → (x<<4) – x　　乘法转换成了1次移位和1次减法

- 移位加减组合运算和直接相乘结果一样的（包括溢出）
- 是否优化取决于组合运算周期数是否小于乘法开销

3. 原码除法

以小数为例

商的符号位单独处理 $x_0 \oplus y_0$

数值部分为绝对值相除 $\frac{x^*}{y^*}$

约定

小数定点除法 $x^* < y^*$　　整数定点除法 $x^* > y^*$

被除数不等于 0

除数不能为 0

(1) 恢复余数法

符号位单独处理，数值位按无符号数计算：

尝试用被除数(移位后的余数)减除数

余数为正　上商 1

余数为负　上商 0，恢复余数

左移 1 位　形成新的余数

特点　上商 n + 1 次

第一次上商判溢出

移 n 次

(2) 不恢复余数法（加减交替法）

- 恢复余数法运算规则

余数 $R_i>0$ 上商 “1”，$2R_i-y^*$

余数 $R_i<0$ 上商 “0”， R_i+y^* 恢复余数

下次上商： $2(R_i+y^*)-y^*=2R_i+y^*$

- 不恢复余数法运算规则

余数 $R_i>0$ 上商 “1” $2R_i-y^*$

余数 $R_i<0$ 上商 “0” $2R_i+y^*$

加减交替

(2) 不恢复余数法（加减交替法）

符号位单独处理，数值位按无符号数计算：

尝试用被除数(移位后的余数)减除数

余数为正　上商 1，余数左移1位，$-y^*$

余数为负　上商 0，余数左移1位，$+y^*$

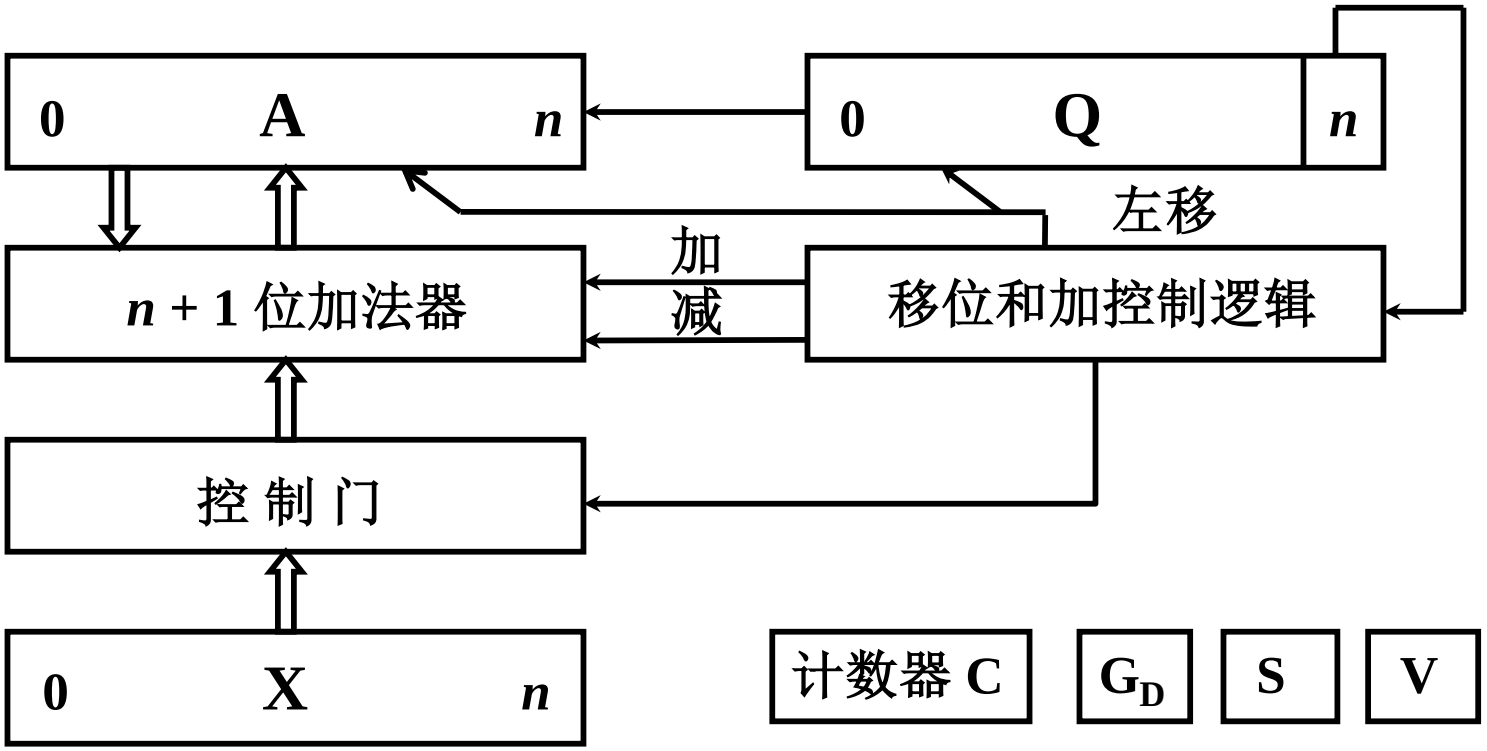
特点　上商 $n+1$ 次

第一次上商判溢出

移 n 次，加 $n+1$ 次

用移位的次数判断除法是否结束

(3) 原码加减交替除法硬件配置



A、X、Q 均 $n+1$ 位

用 Q_n 控制加减交替

6.4 浮点四则运算

一、浮点加减运算

$$x = S_x \cdot 2^{j_x} \qquad y = S_y \cdot 2^{j_y}$$

1. 对阶

(1) 求阶差

$$\Delta j = j_x - j_y = \begin{cases} = 0 & j_x = j_y \quad \text{已对齐} \\ > 0 & j_x > j_y \begin{cases} x \text{ 向 } y \text{ 看齐} & S_x \leftarrow 1,\ j_x - 1 \\ y \text{ 向 } x \text{ 看齐} & \checkmark\ S_y \rightarrow 1,\ j_y + 1 \end{cases} \\ < 0 & j_x < j_y \begin{cases} x \text{ 向 } y \text{ 看齐} & \checkmark\ S_x \rightarrow 1,\ j_x + 1 \\ y \text{ 向 } x \text{ 看齐} & S_y \leftarrow 1,\ j_y - 1 \end{cases} \end{cases}$$

(2) 对阶原则

小阶向大阶看齐

6.4 浮点四则运算

6.4

一、浮点加减运算

1. 对阶

① 求阶差　$[\Delta j]_{补} = [j_x]_{补} - [j_y]_{补}$

② 对阶　小阶向大阶看齐

2. 尾数求和

按补码规则计算

3. 规格化

左归 / 右归

3. 规格化

(1) 规格化数的定义

$$r = 2 \qquad \frac{1}{2} \leq |S| < 1$$

(2) 规格化数的判断

原码　不论正数、负数，第一数位为1

补码　符号位和第 一数位不同

(3) 左规

尾数左移一位，阶码减 1，

直到数符和第一数位不同为止

(4) 右规

当 尾数溢出（ >1）时，需 右规

即尾数出现 01.××…×或 10.××…×时

尾数右移一位，阶码加 1

4. 舍入

在 对阶 和 右规 过程中，可能出现 尾数末位丢失 引起误差，需考虑舍入。

常见的方法：

(1) 0 舍 1 入法

结果精确，但可能增加加法和右归的次数

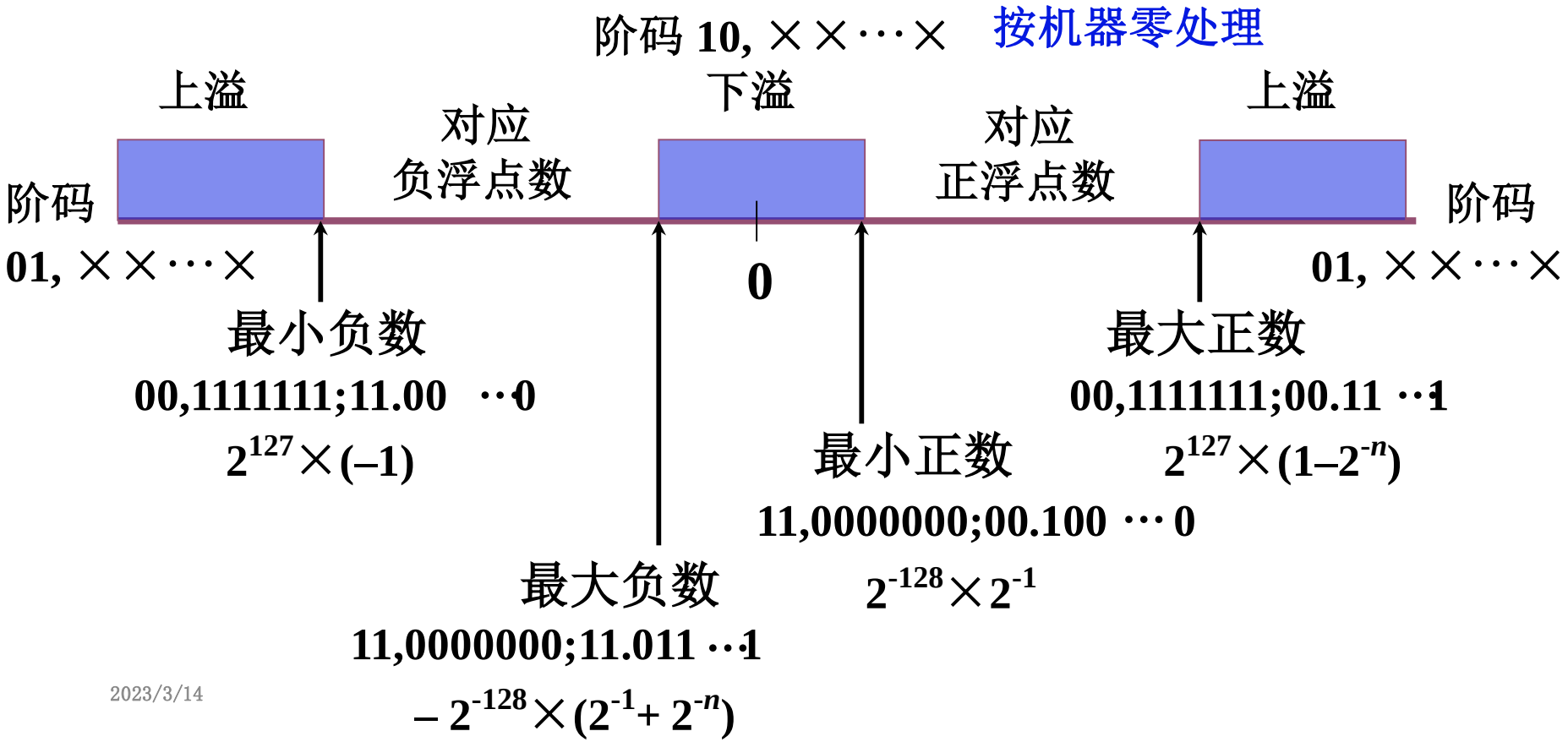
(2) 恒置 “1” 法

误差略大，胜在不需要任何额外的运算

5. 溢出判断

6.4

设机器数为补码，尾数为 规格化形式，并假设阶符取 2 位，阶码的数值部分取 7 位，数符取 2 位，尾数取 n 位，则该 补码 在数轴上的表示为



二、浮点乘除运算

1. 浮点数乘法运算

如：$x = S_x \times 2^{jx}$　　$y = S_y \times 2^{jy}$

$$x \times y = (S_x \times 2^{jx}) \times (S_y \times 2^{jy}) = (S_x \times S_y) \times 2^{(jx + jy)}$$

① 阶码相加

阶码相加可能产生溢出，要进行溢出判断，如溢出计算机要进行处理

② 尾数相乘

尾数相乘可得积的尾数，可按定点乘法运算方法运算

③ 结果规格化

可按浮点加/减法运算规格化方式处理，舍入方式也相同

二、浮点乘除运算

2. 浮点数除法运算

如：$x = S_x \times 2^{jx}$　　$y = S_y \times 2^{jy}$

$$x \div y = (S_x \times 2^{jx}) \div (S_y \times 2^{jy}) = (S_x \div S_y) \times 2^{(jx - jy)}$$

① 尾数调整

如被除数尾数大于除数尾数 (绝对值)，则将被除数尾数右移一位，阶码 +1

② 阶码相减

商的阶码等于被除数的阶码减去除数的阶码

③ 尾数相除

以被除数的尾数除以除数的尾数以获得商的尾数，与定点除法运算相同

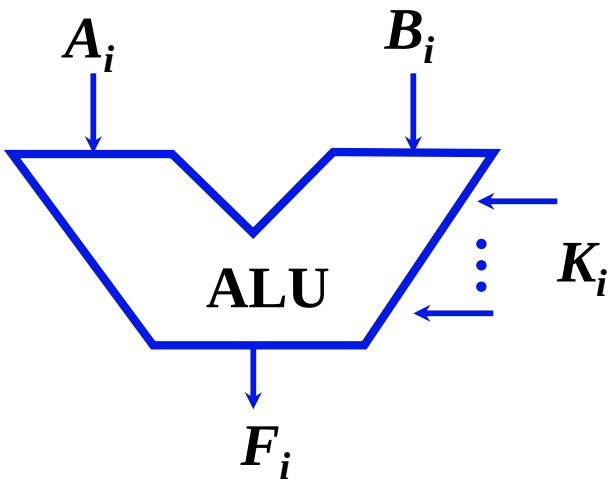
浮点四则运算小结

6.3

- 加减法运算
  - ① 对阶 ➔ 先求阶差，然后小阶向大阶看齐
  - ② 尾数求和
  - ③ 规格化
- 乘法运算
  - ① 阶码相加
  - ② 尾数相乘
  - ③ 结果规格化
- 除法运算
  - ① 尾数调整 ➔ 调整被除数
  - ② 阶码相减
  - ③ 尾数相除

6.5 算术逻辑单元

一、ALU 电路



组合逻辑电路

K_i 不同取值

F_i 不同

四位 ALU　74181

$M = 0$　　算术运算

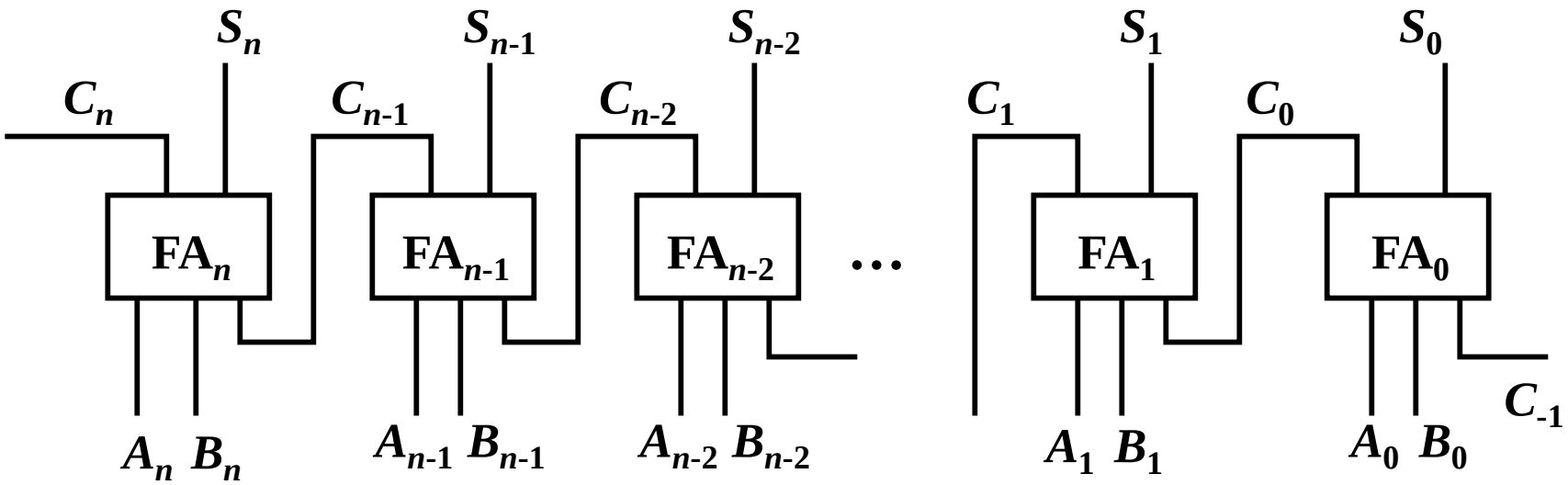
$M = 1$　　逻辑运算

$S_3 \sim S_0$　　不同取值，可做不同运算

二、快速进位链

6.5

1. 并行加法器



$$S_i = \overline{A}_i\,\overline{B}_i\,C_{i-1} + \overline{A}_i\,B_i\,\overline{C}_{i-1} + A_i\,\overline{B}_i\,\overline{C}_{i-1} + A_i\,B_i\,C_{i-1}$$

$$C_i = \overline{A}_i\,B_i\,C_{i-1} + A_i\,\overline{B}_i\,C_{i-1} + A_i\,B_i\,\overline{C}_{i-1} + A_i\,B_i\,C_{i-1}$$

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

$d_i = A_i\,B_i$　本地进位　　$t_i = A_i + B_i$　传送条件

则 $C_i = d_i + t_i C_{i-1}$

2. 串行进位链

进位链　　传送进位的电路

串行进位链　　进位串行传送 (逐位进位)

以 4 位全加器为例，每一位的进位表达式为

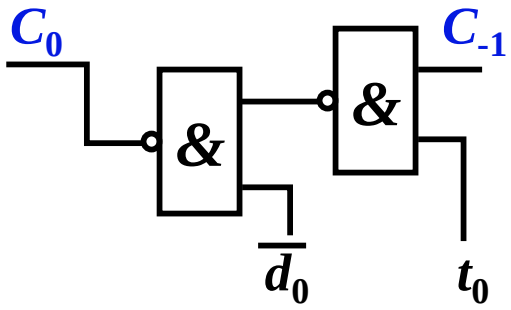
$$C_0 = d_0 + t_0 C_{-1} = \overline{\overline{d_0} \cdot \overline{t_0 C_{-1}}}$$

$$C_1 = d_1 + t_1 C_0$$

$$C_2 = d_2 + t_2 C_1$$

$$C_3 = d_3 + t_3 C_2$$

设单个与非门的延迟时间为t_y



4位 全加器产生进位的全部时间为 $8t_y$

n 位全加器产生进位的全部时间为 $2nt_y$

3. 并行进位链（先行进位，跳跃进位）

6.5

n 位加法器的进位同时产生　　以 4 位加法器为例

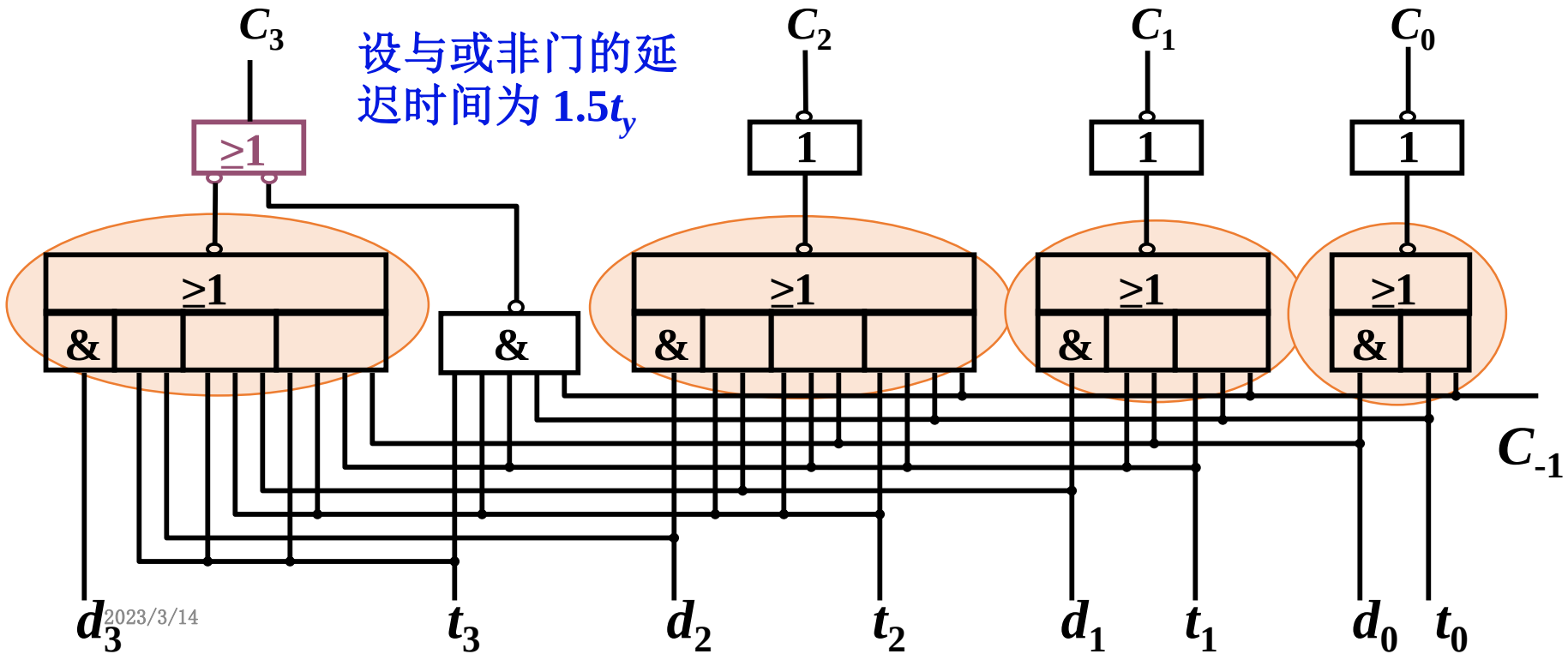
$$C_0 = d_0 + t_0 C_{-1}$$

$$C_1 = d_1 + t_1 C_0 = d_1 + t_1 d_0 + t_1 t_0 C_{-1}$$

$$C_2 = d_2 + t_2 C_1 = d_2 + t_2 d_1 + t_2 t_1 d_0 + t_2 t_1 t_0 C_{-1}$$

$$C_3 = d_3 + t_3 C_2 = d_3 + t_3 d_2 + t_3 t_2 d_1 + t_3 t_2 t_1 d_0 + t_3 t_2 t_1 t_0 C_{-1}$$

当 d_i t_i 形成后，只需 $2.5t_y$ 产生全部进位



3. 并行进位链（先行进位，跳跃进位）

n 位加法器的进位同时产生　　以 4 位加法器为例

$$C_0 = d_0 + t_0 C_{-1}$$

$$C_1 = d_1 + t_1 C_0 = d_1 + t_1 d_0 + t_1 t_0 C_{-1}$$

$$C_2 = d_2 + t_2 C_1 = d_2 + t_2 d_1 + t_2 t_1 d_0 + t_2 t_1 t_0 C_{-1}$$

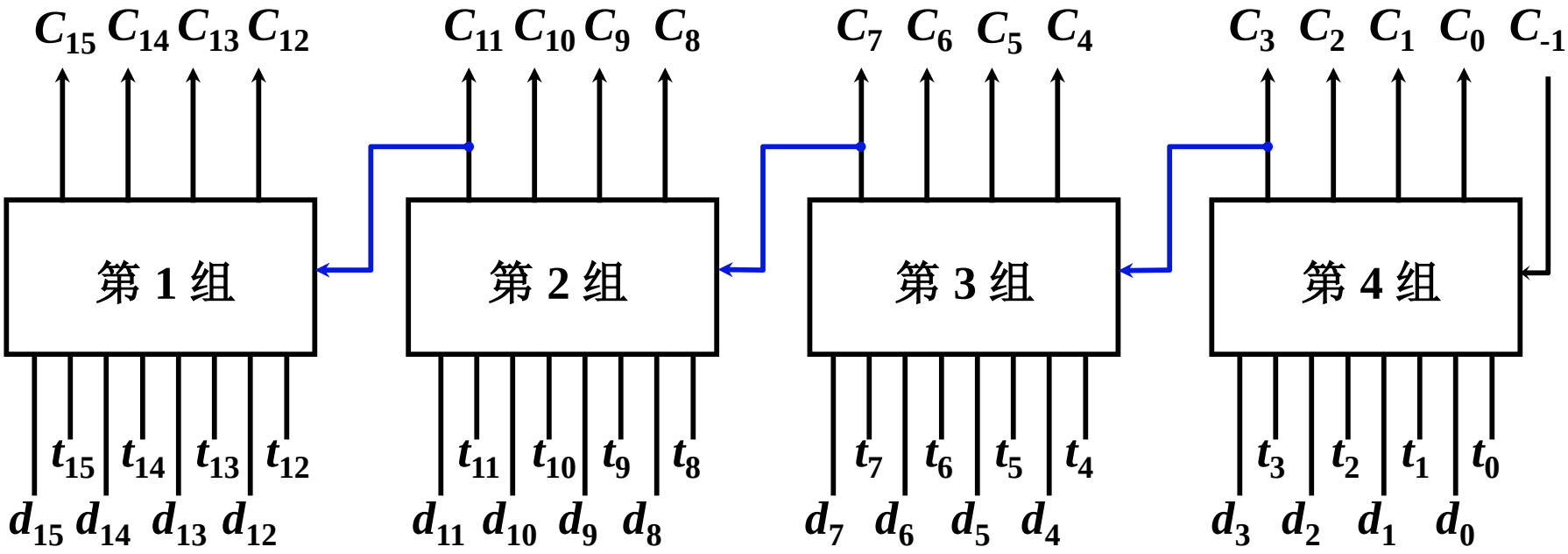
$$C_3 = d_3 + t_3 C_2 = d_3 + t_3 d_2 + t_3 t_2 d_1 + t_3 t_2 t_1 d_0 + t_3 t_2 t_1 t_0 C_{-1}$$

并行进位链的特点：

- ① 进位输出仅与最低位进位输入C_{-1}有关
- ② 位数越长，进位链电路复杂度越高
- ③ 通常按照 4位一组 进行分组运算

(1) 单重分组跳跃进位链

n 位全加器分若干小组，小组中的进位同时产生，
小组与小组之间采用串行进位　　以 *n* = 16 为例



当 d_i、t_i 形成后　经 2.5 t_y　产生 $C_3 \sim C_0$

5 t_y　产生 $C_7 \sim C_4$

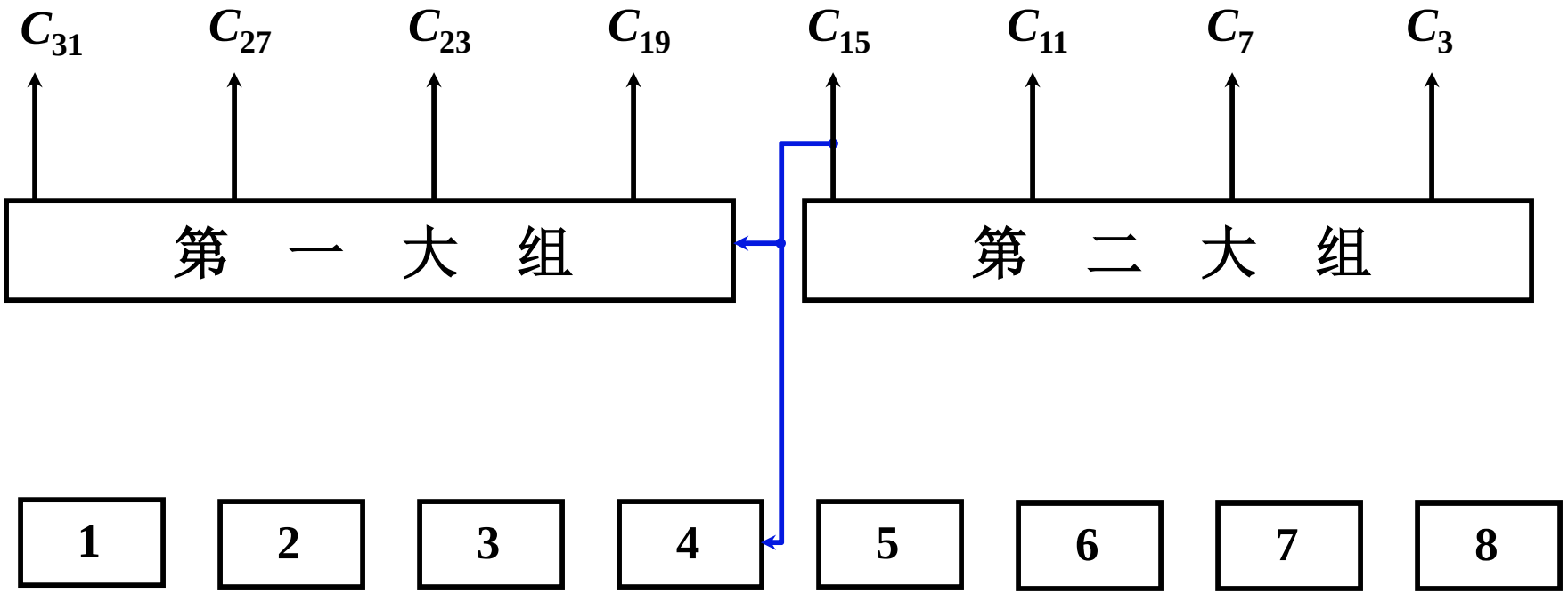
7.5 t_y　产生 $C_{11} \sim C_8$

1 0 t_y　产生 $C_{15} \sim C_{12}$

(2) 双重分组跳跃进位链

n 位全加器分若干大组，大组中又包含若干小组。每个大组中小组的最高位进位同时产生。大组与大组之间采用串行进位。

以 $n = 32$ 为例



(3) 双重分组跳跃进位链 大组进位分析

6.5

以第 8 小组为例

$$C_3 = d_3 + t_3C_2 = \underbrace{d_3 + t_3d_2 + t_3t_2d_1 + t_3t_2t_1d_0}_{D_8} + \underbrace{t_3t_2t_1t_0}_{T_8}C_{-1}$$

D_8 小组的本地进位　与外来进位无关

T_8 小组的传送条件　与外来进位无关　传递外来进位

同理　第 7 小组　$C_7 = D_7 + T_7C_3$

第 6 小组　$C_{11} = D_6 + T_6C_7$

第 5 小组　$C_{15} = D_5 + T_5C_{11}$

进一步展开得

$$C_3 = D_8 + T_8C_{-1}$$

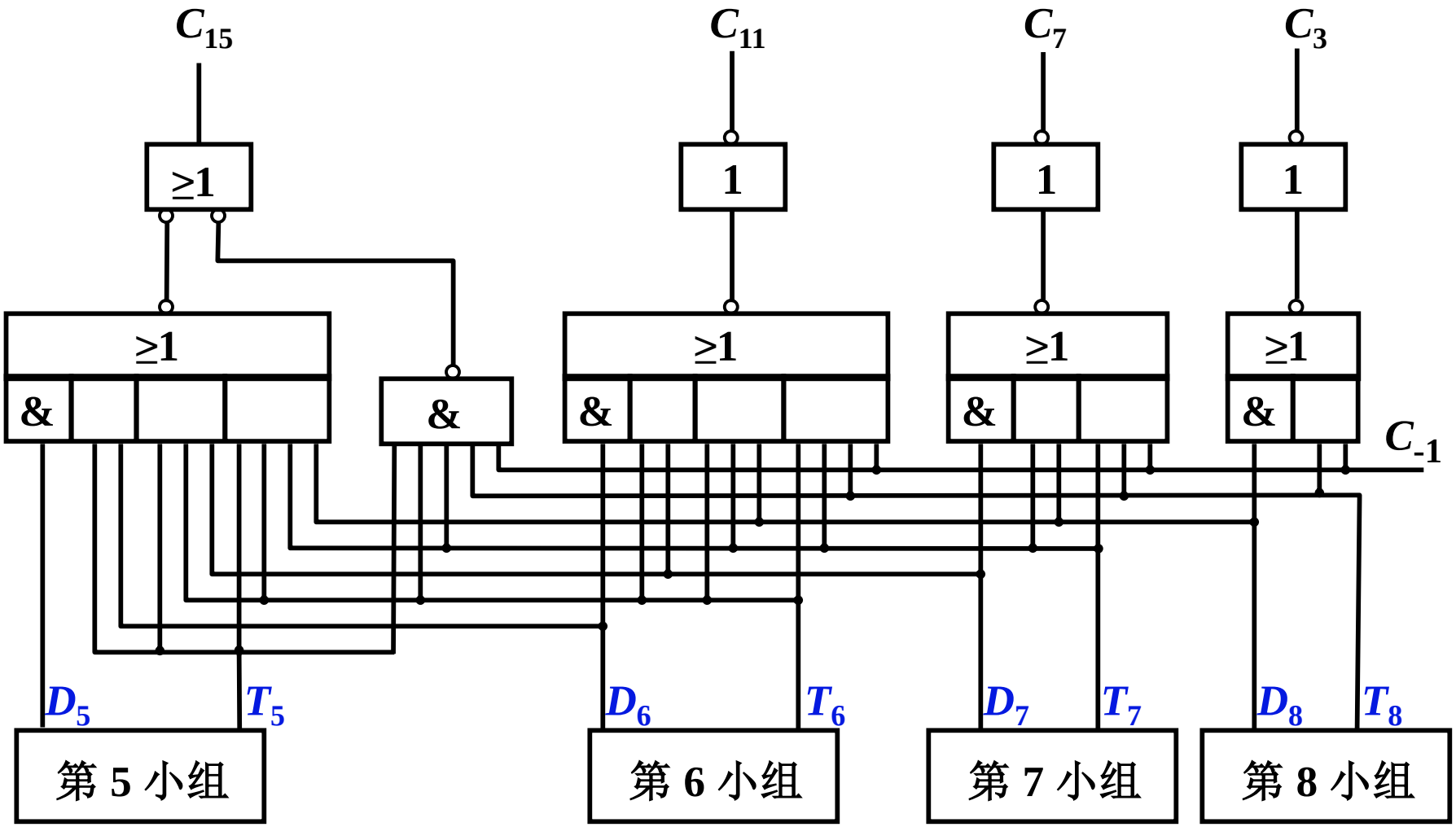
$$C_7 = D_7 + T_7C_3 = D_7 + T_7D_8 + T_7T_8C_{-1}$$

$$C_{11} = D_6 + T_6C_7 = D_6 + T_6D_7 + T_6T_7D_8 + T_6T_7T_8C_{-1}$$

$$C_{15} = D_5 + T_5C_{11} = D_5 + T_5D_6 + T_5T_6D_7 + T_5T_6T_7D_8 + T_5T_6T_7T_8C_{-1}$$

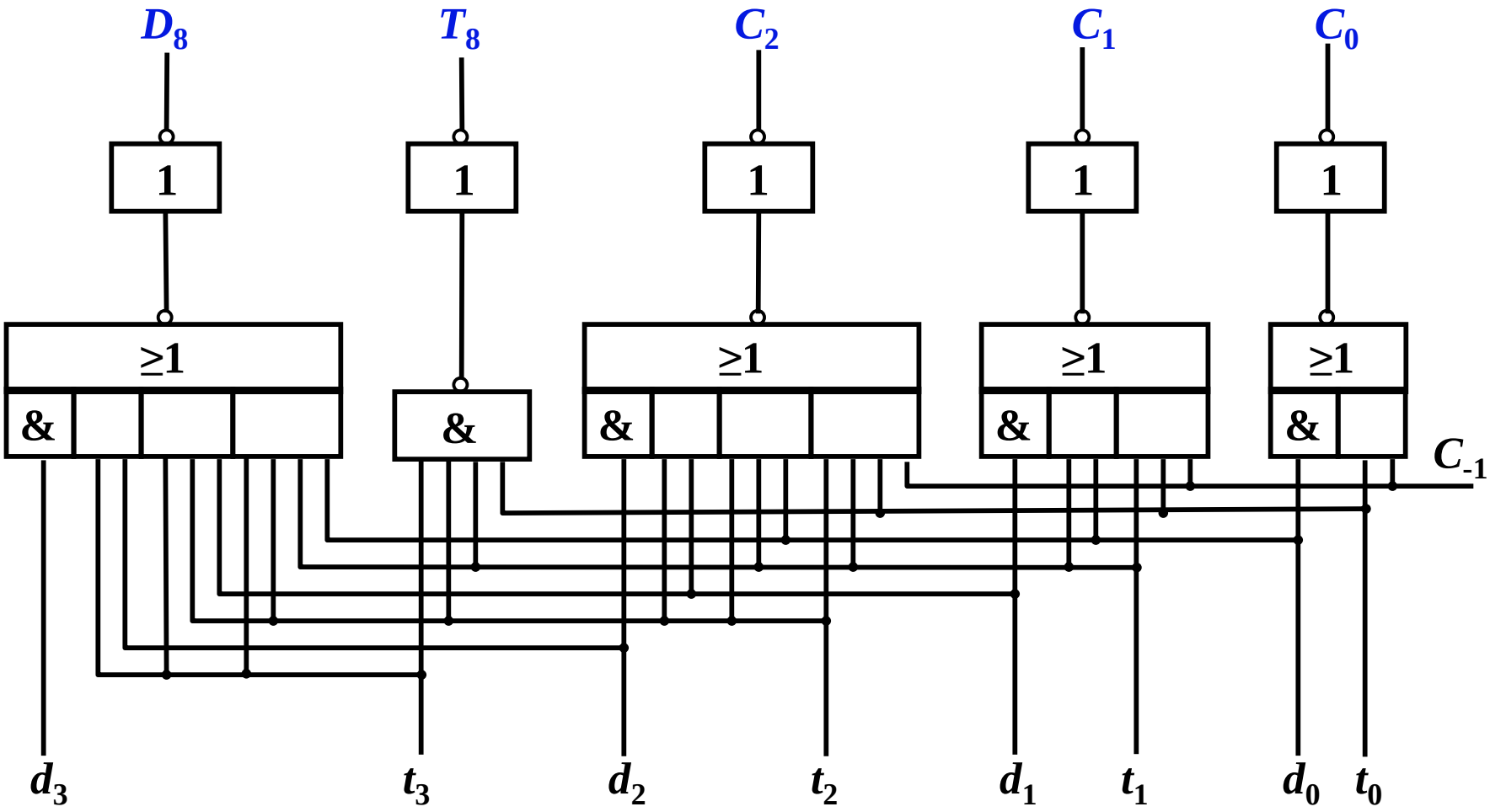
(4) 双重分组跳跃进位链的 大组 进位线路 6.5

以第 2 大组为例



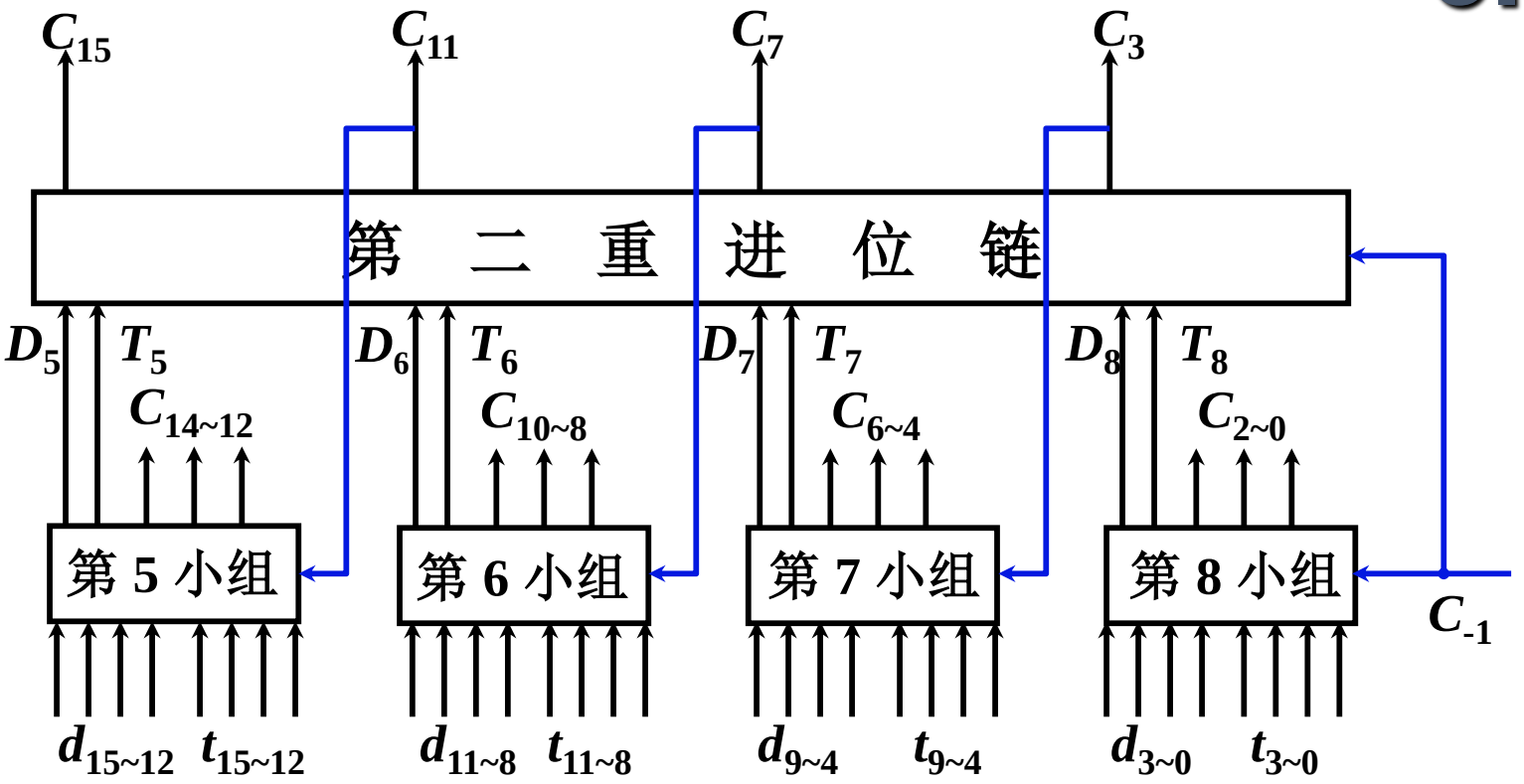
(5) 双重分组跳跃进位链的 小组 进位线路 6.5

以第 8 小组为例　只产生 低 3 位 的进位和 本小组的 D_8 T_8



(6) n =16 双重分组跳跃进位链

6.5



当 d_i、t_i 和 C_{-1} 形成后　经 2.5 t_y　产生 C_2、C_1、C_0、$D_5 \sim D_8$、$T_5 \sim T_8$

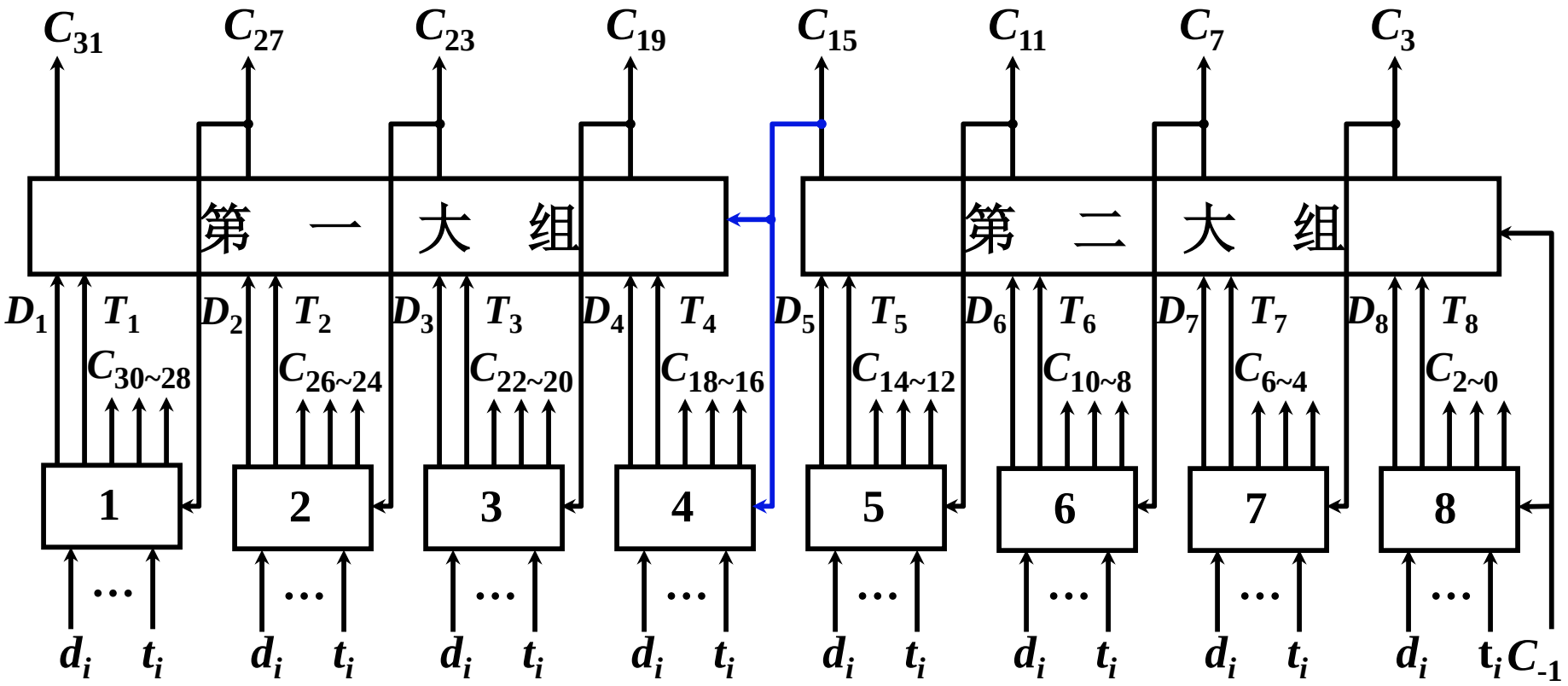
经 5 t_y　产生 C_{15}、C_{11}、C_7、C_3

经 7.5 t_y　产生 $C_{14}\sim C_{12}$、$C_{10}\sim C_8$、$C_6\sim C_4$

串行进位链　经 32 t_y　产生　全部进位

单重分组跳跃进位链　经 10 t_y　产生　全部进位

(7) n =32 双重分组跳跃进位链



当 d_i、t_i 形成后 经 2.5 t_y 产生 C_2、C_1、C_0、$D_1 \sim D_8$、$T_1 \sim T_8$

5 t_y 产生 C_{15}、C_{11}、C_7、C_3

7.5 t_y 产生 $C_{18} \sim C_{16}$、$C_{14} \sim C_{12}$、$C_{10} \sim C_8$、$C_6 \sim C_4$
C_{31}、C_{27}、C_{23}、C_{19}

10 t_y 产生 $C_{30} \sim C_{28}$、$C_{26} \sim C_{24}$、$C_{22} \sim C_{20}$

快速进位链小结

- 串行进位链
  - ① 电路简单 ➔ 进位信号逐位传递
  - ② 速度慢 ➔ 多个串联的全加器延迟叠加
- 并行进位链
  - ① 通过增加组合逻辑电路并行计算各进位信号
  - ② 进位输出仅与最低位进位输入C_{-1}有关
  - ③ 位数越长，进位链电路复杂度越高
  - ④ 通常按照 4位一组 进行分组运算