中国科学院大学
University of Chinese Academy of Sciences
B0911006Y-01 2024-2025学年春季学期

计算机组成原理
Principles of Computer Organization

计算机中数的运算

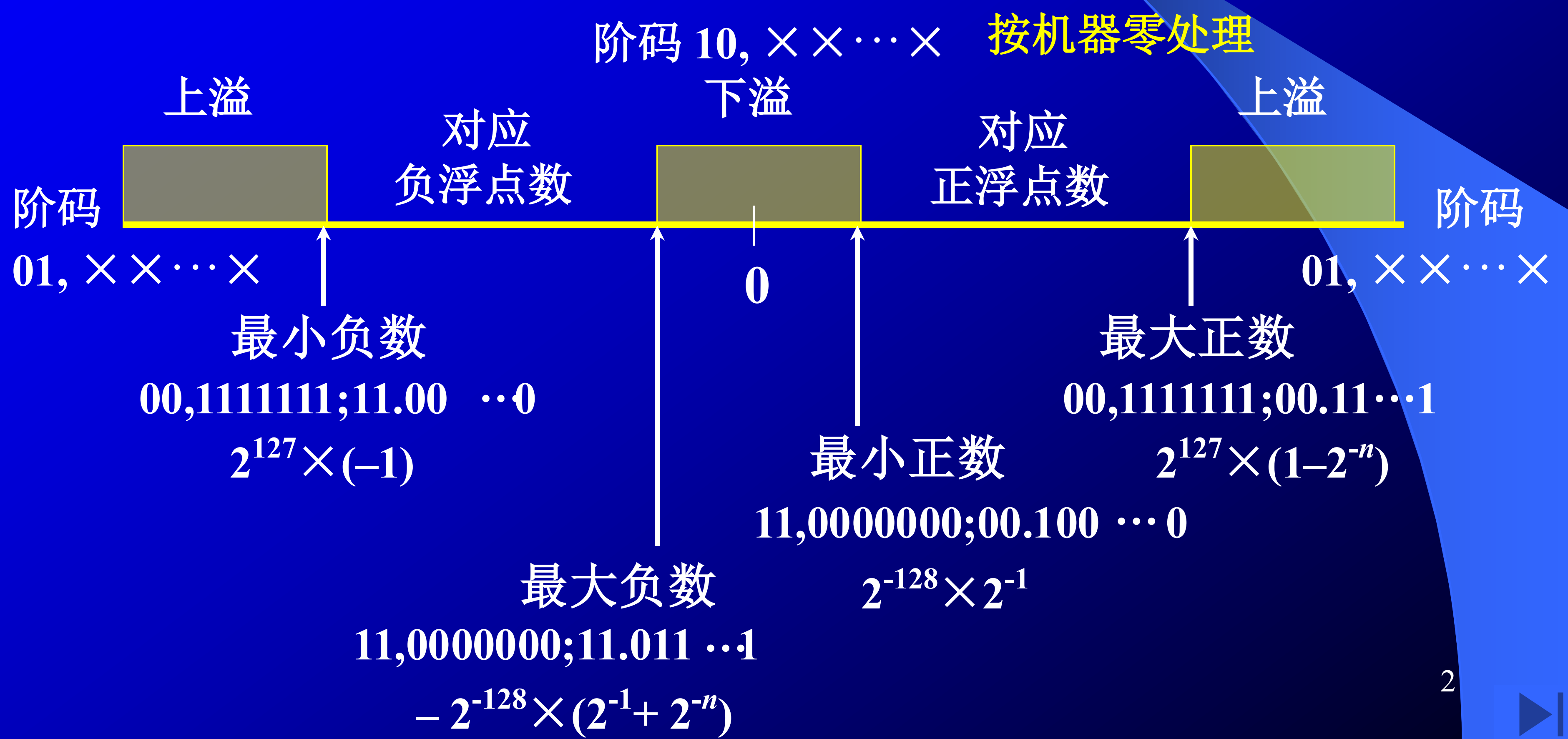
如何判断浮点运算的溢出？如何实现高效的加法进位？

主讲教师：石 侃
shikan@ict.ac.cn

2025年4月21日

5. 溢出判断

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

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

1. 乘法

$$x \cdot y = (S_x \cdot S_y) \times 2^{j_x + j_y}$$

2. 除法

$$\frac{x}{y} = \frac{S_x}{S_y} \times 2^{j_x - j_y}$$

3. 步骤

(1) 阶码采用 补码定点加（乘法）减（除法）运算

(2) 尾数乘除同 定点 运算

(3) 规格化

4. 浮点运算部件

阶码运算部件，尾数运算部件

引申：浮点数的精度问题

- 1991年2月25日，海湾战争中，美国在沙特阿拉伯达摩地区设置的爱国者导弹拦截伊拉克的飞毛腿导弹失败，致使飞毛腿导弹击中了一个美军军营，杀死了美军28名士兵。其原因是由于爱国者导弹系统时钟内的一个软件错误造成的，引起这个软件错误的原因是浮点数的精度问题。
- 爱国者导弹系统中有一内置时钟，用计数器实现，每隔0.1秒硬件计数一次。程序用0.1的一个24位定点二进制小数x来乘以计数值作为以秒为单位的时间
- 这个x的机器数是多少呢?
- 0.1的二进制表示是一个无限循环序列：0.0001100 [1100]...，x=0.000 1100 1100 1100 1100 1100B。显然，x是0.1的近似表示，0.1-x

  = 0.000 1100 1100 1100 1100 1100 [1100]... -

  0.000 1100 1100 1100 1100 1100B，即为:

  = 0.000 0000 0000 0000 0000 0000 1100 [1100]...B (橙色字是0.1,前面有20个0)

  $=2^{-20}\times 0.1 \approx 9.54\times 10^{-8}$ 这就是机器值与真值之间的误差!

举例：爱国者导弹定位错误

已知在爱国者导弹准备拦截飞毛腿导弹之前，已经连续工作了100小时，飞毛腿的速度大约为2000米/秒，则由于时钟计算误差而导致的距离误差是多少?

100小时相当于计数了$100\times60\times60\times10=36\times10^{5}$次，因而导弹的时钟已经偏差了$9.54\times10^{-8}\times36\times10^{5}\approx0.343$秒

因此，距离误差是2000×0.343秒≈687米

实际上，以色列方面已经发现了这个问题并于1991年2月11日知会了美国陆军及爱国者计划办公室（软件制造商）。以色列方面建议重新启动爱国者系统的电脑作为暂时解决方案，可是美国陆军方面却不知道每次需要间隔多少时间重新启动系统一次。1991年2月16日，制造商向美国陆军提供了更新软件，但这个软件最终却在飞毛腿导弹击中军营后的一天才运抵部队。

如果你是软件制造商，你想怎么解决这个bug?

举例：爱国者导弹定位错误

- 若x用float型表示，则x的机器数是什么？0.1与x的偏差是多少？系统运行100小时后的时钟偏差是多少？在飞毛腿速度为2000米/秒的情况下，预测的距离偏差为多少？
  - 0.1= 0.0 0011[0011]B=+1.1 0011 0011 0011 0011 0011 00B×2^{-4}，故x的机器数为0 011 1101 1 100 1100 1100 1100 1100 1100
  - Float型仅24位有效位数，后面的有效位全被截断，故x与0.1之间的误差为：|x–0.1|=0.000 0000 0000 0000 0000 0000 0000 1100 [1100]...B。这个值等于2^{-24}×0.1 ≈ 5.96×10^{-9}。100小时后时钟偏差5.96×10^{-9}×36×10^{5} ≈ 0.0215秒。距离偏差0.0215×2000≈43米。比爱国者导弹系统精确约16倍。

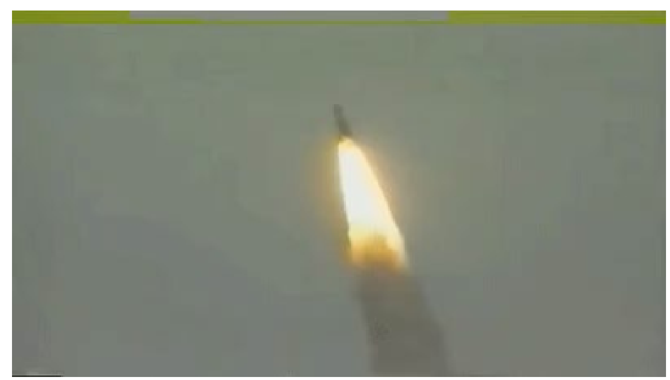
举例：爱国者导弹定位错误

- 若用32位二进制定点小数x=0.000 1100 1100 1100 1100 1100 1100 1101 B表示0.1，则误差比用float表示误差更大还是更小？
  - 当x=0.000 1100 1100 1100 1100 1100 1100 1101 B时，与0.1之间的误差约为：|x–0.1|=0.000 0000 0000 0000 0000 0000 0000 0000 00 1100 [1100]…B。这个值等于$2^{-30} \times 0.1 \approx 9.31 \times 10^{-11}$。100小时后时钟偏差$9.31 \times 10^{-11} \times 36 \times 10^{5} \approx 0.000335$秒。预测的距离偏差仅为$0.000335 \times 2000 \approx 0.67$米。

浮点数的精度问题举例：欧洲阿丽亚娜-5火箭

- 1996年6月4日，Ariane 5火箭测试发射，仅37秒钟后，偏离了飞行路线，然后解体爆炸，火箭上载有价值5亿美元的通信卫星
- 原因是在将一个64位浮点数转换为16位带符号整数时，产生了溢出异常。溢出的值是火箭的水平速率，这比原来的Ariane 4火箭所能达到的速率高出了5倍。在设计Ariane 4火箭软件时，设计者确认水平速率决不会超出一个16位的整数；但在设计Ariane 5时，他们没有重新检查这部分，而是直接重用了原来的软件代码

```
double d_bh;
short s_bh;
sense_horizontal_velocity(&d_bh);
s_bh = d_bh; //OPERAND ERROR
```



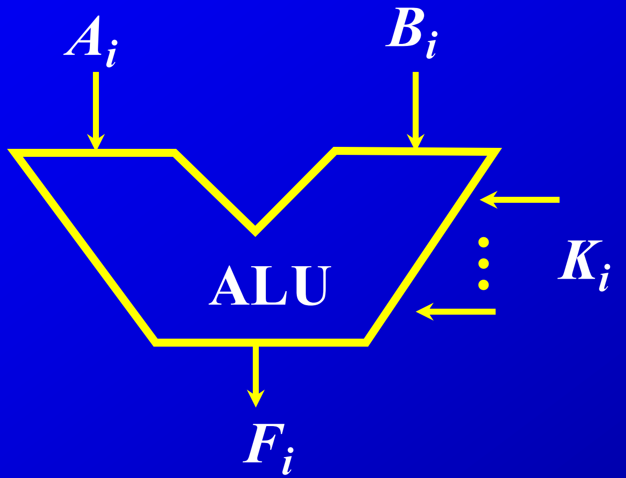
- 在不同数据类型之间转换时，往往隐藏着一些不容易被察觉的错误，这种错误有时会带来重大损失，因此，编程时要非常小心！

小结：浮点数运算的精度问题

- 从爱国者导弹的例子可以看出：
  - 用32位定点小数表示0.1，比采用float精度高64倍
  - 用float表示在计算速度上更慢，必须先把计数值转换为IEEE 754格式浮点数，然后再对两个IEEE 754格式的数相乘，故采用float比直接将两个二进制数相乘要慢
- Ariane 5火箭和爱国者导弹等真实案例带来的启示
  - ✓程序员应对底层机器级数据的表示和运算有深刻理解
  - ✓计算机世界里，经常是“差之毫厘，失之千里”，需要细心再细心，精确再精确
  - ✓不能遇到小数就用浮点数表示，有些情况下（如需要将一个整数变量乘以一个确定的小数常量），可先用一个确定的定点整数与整数变量相乘，然后再通过移位运算来确定小数点

6.5 算术逻辑单元

一、一位ALU 电路



组合逻辑电路

K_i 不同取值

F_i 不同

四位 ALU　74181

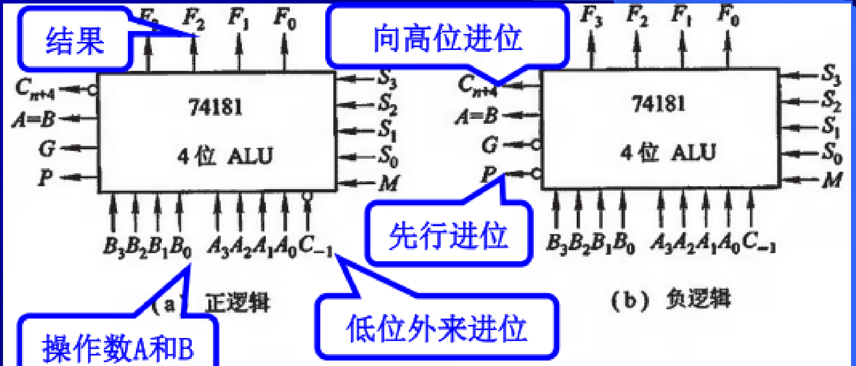


图 6.17　74181 外特性示意图

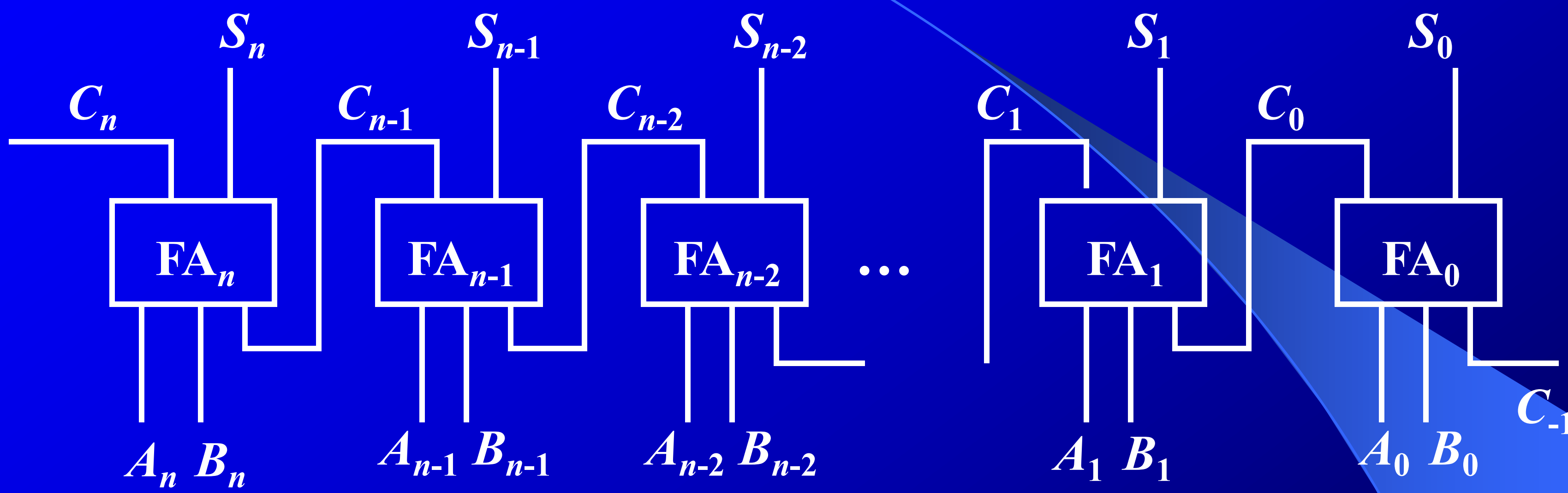
$M = 0$　　算术运算

$M = 1$　　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

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

6.5

进位链　　　　传送进位的电路

串行进位链　　进位串行传送

以 4 位全加器为例，每一位的进位表达式为

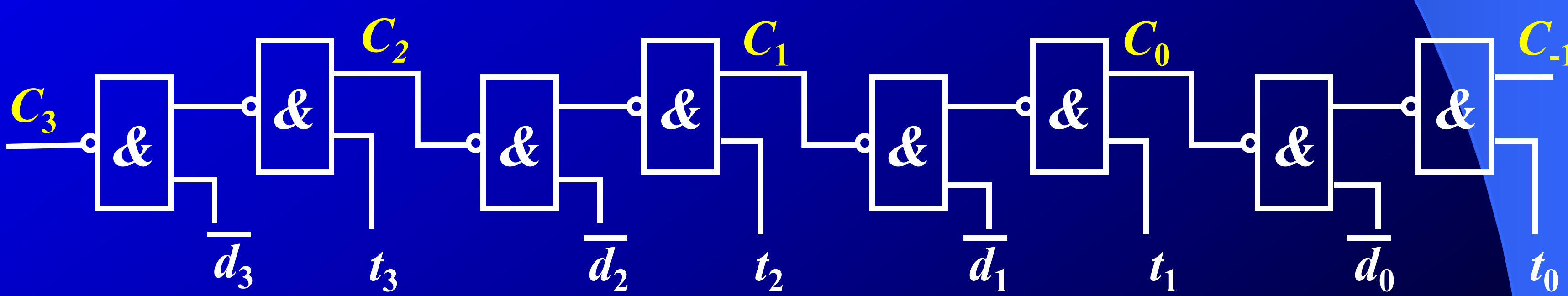
$$C_0 = d_0 + t_0 C_{-1} = \overline{\overline{d_0} \cdot \overline{t_0 C_{-1}}}$$

$$C_1 = d_1 + t_1 C_0$$

$$C_2 = d_2 + t_2 C_1$$

$$C_3 = d_3 + t_3 C_2$$

设与非门的级延迟时间为 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）

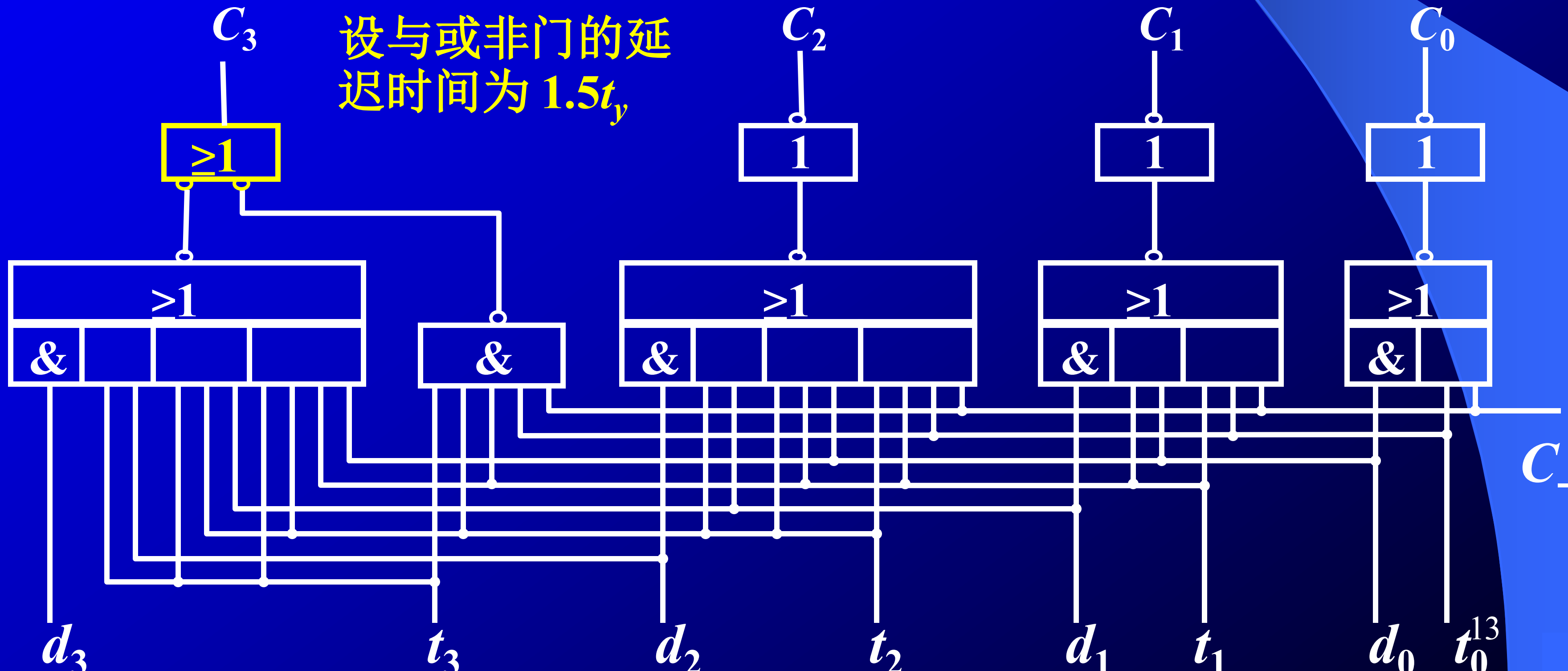
6.5

n 位加法器的进位同时产生　　以 4 位加法器为例

$$C_0 = d_0 + t_0 C_{-1}$$
$$C_1 = d_1 + t_1 C_0 = d_1 + t_1 d_0 + t_1 t_0 C_{-1}$$
$$C_2 = d_2 + t_2 C_1 = d_2 + t_2 d_1 + t_2 t_1 d_0 + t_2 t_1 t_0 C_{-1}$$
$$C_3 = d_3 + t_3 C_2 = d_3 + t_3 d_2 + t_3 t_2 d_1 + t_3 t_2 t_1 d_0 + t_3 t_2 t_1 t_0 C_{-1}$$

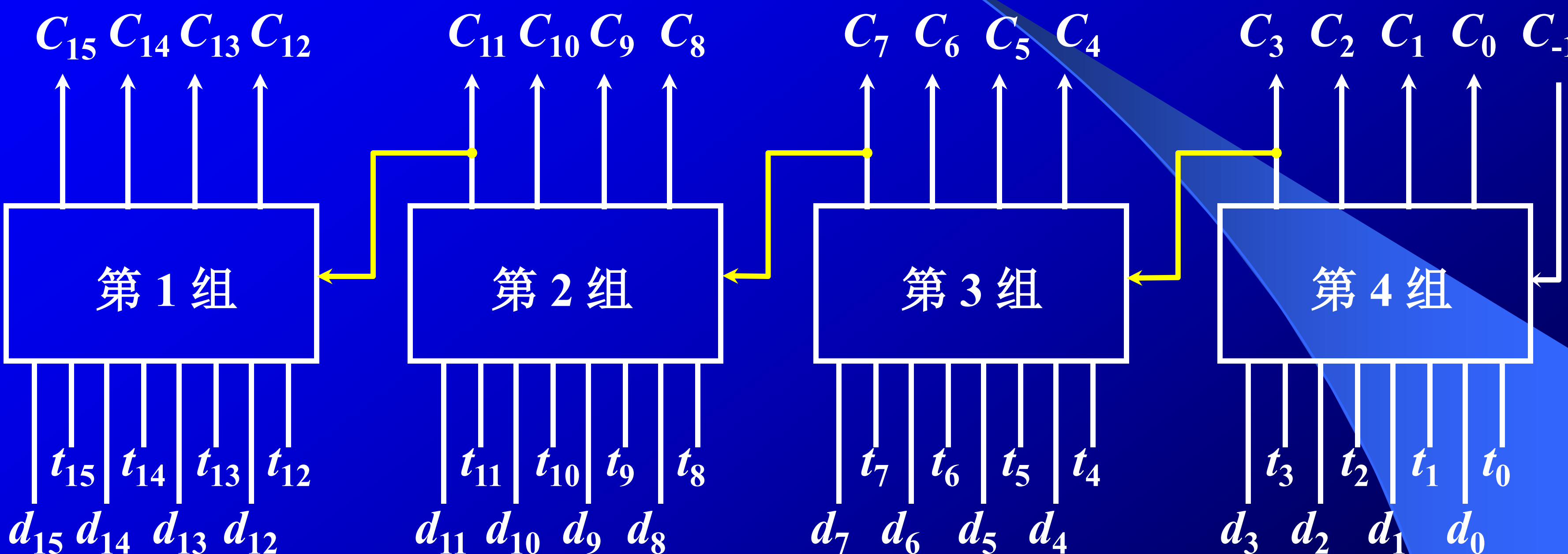
当 $d_i t_i$ 形成后，只需 $2.5t_y$ 产生全部进位

设与或非门的延迟时间为 $1.5t_y$



(1) 单重分组跳跃进位链

n 位全加器分若干小组，小组内的进位同时产生，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 $C_7 \sim C_4$

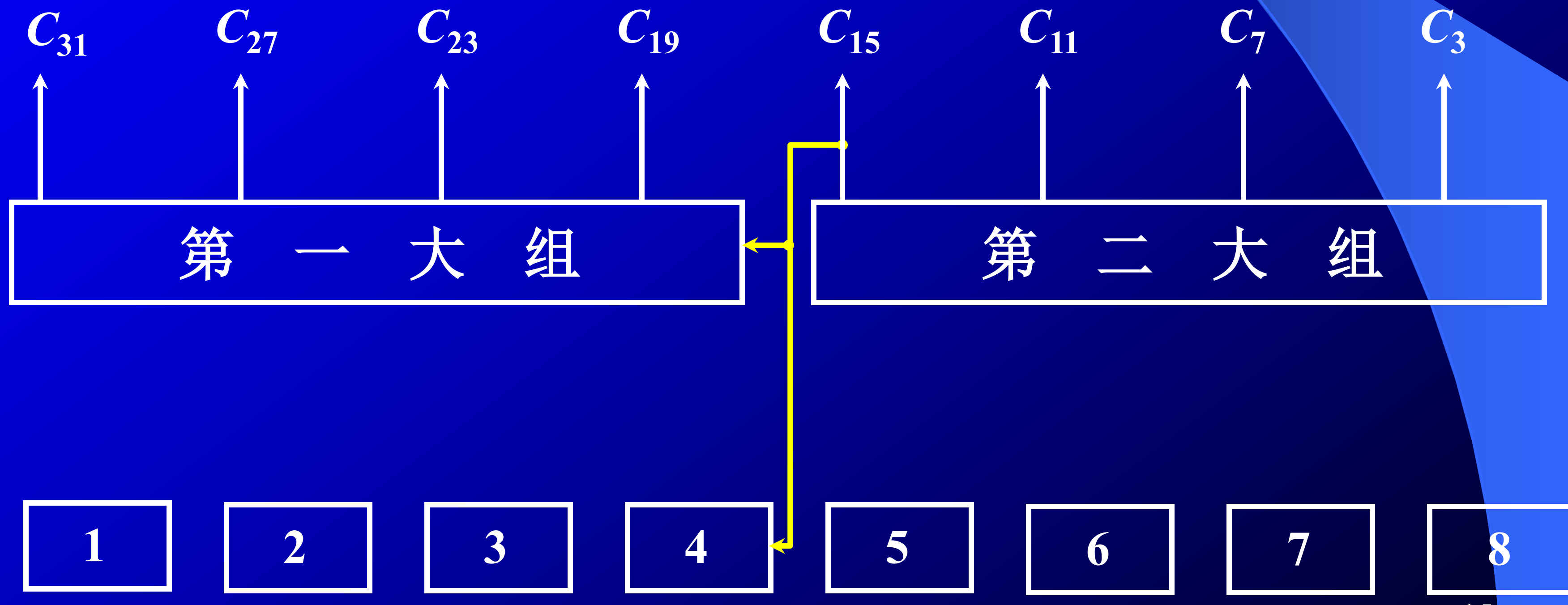
$7.5\ t_y$　产生 $C_{11} \sim C_8$

$10\ t_y$　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

以第 8 小组为例

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

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

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

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

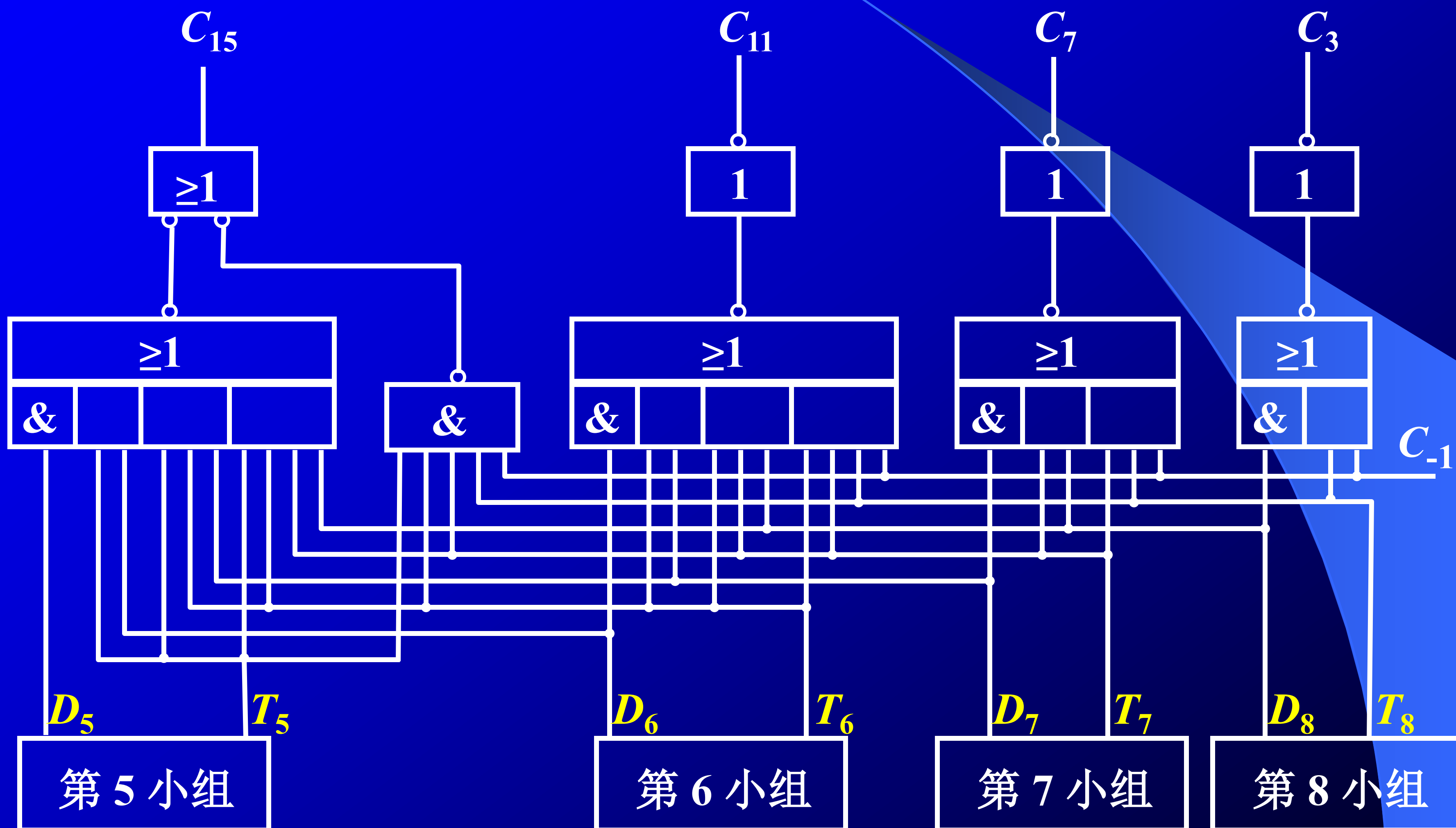
$$C_7 = D_7 + T_7C_3 = D_7 + T_7D_8 + T_7T_8C_{-1}$$

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

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

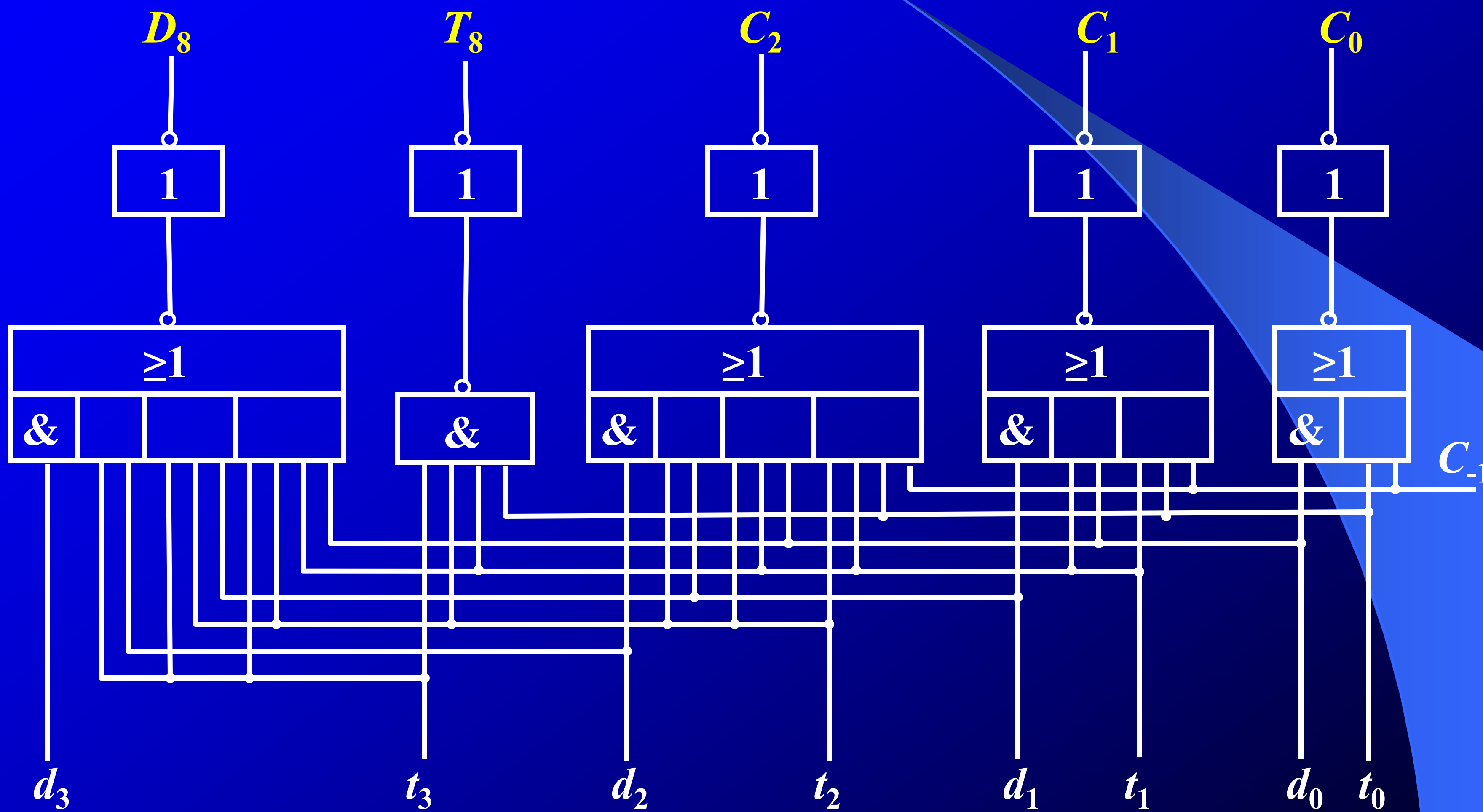
(4) 双重分组跳跃进位链的 大组 进位线路 6.5

以第 2 大组为例



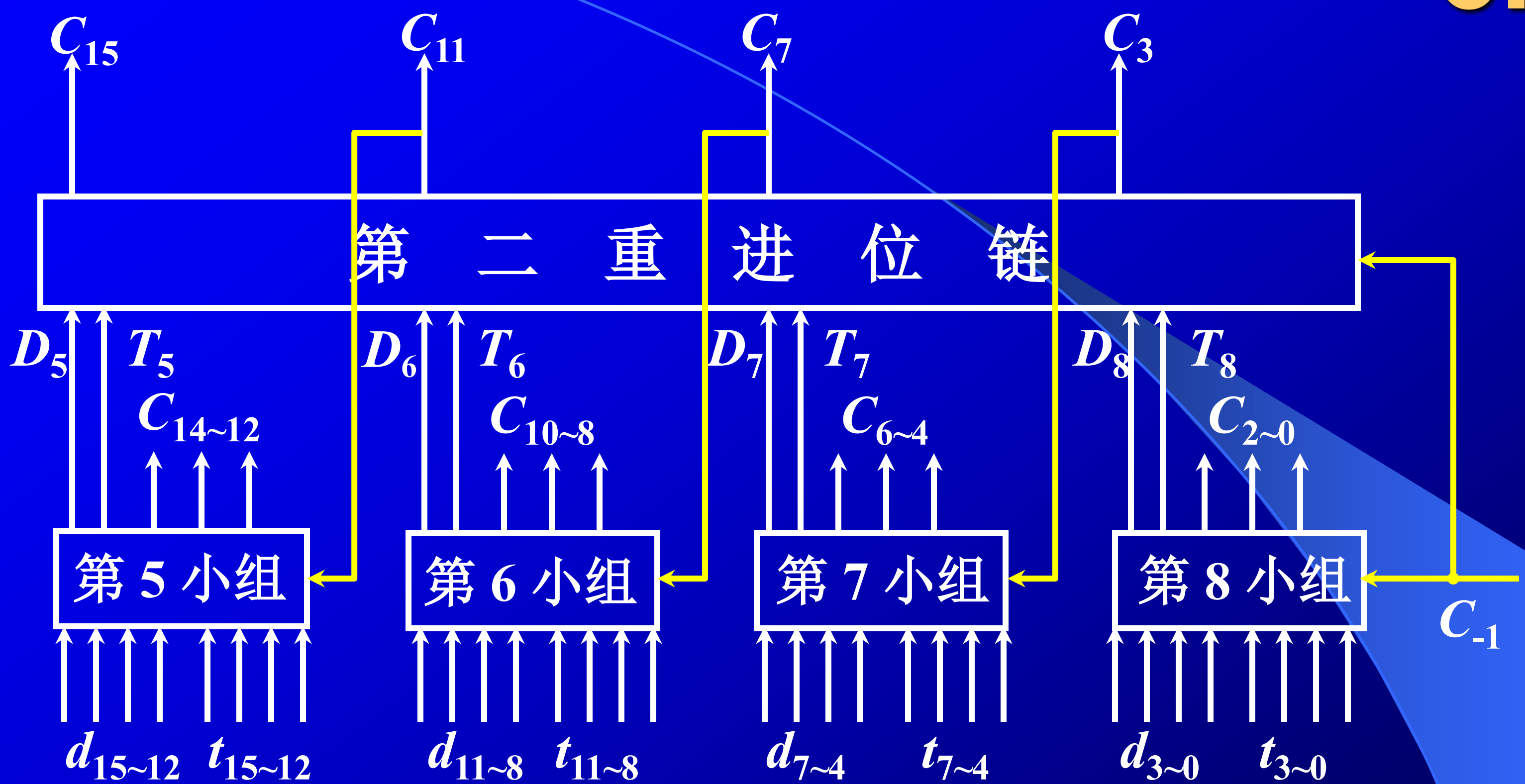
(5) 双重分组跳跃进位链的 小组 进位线路 6.5

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、$D_5 \sim D_8$、$T_5 \sim T_8$

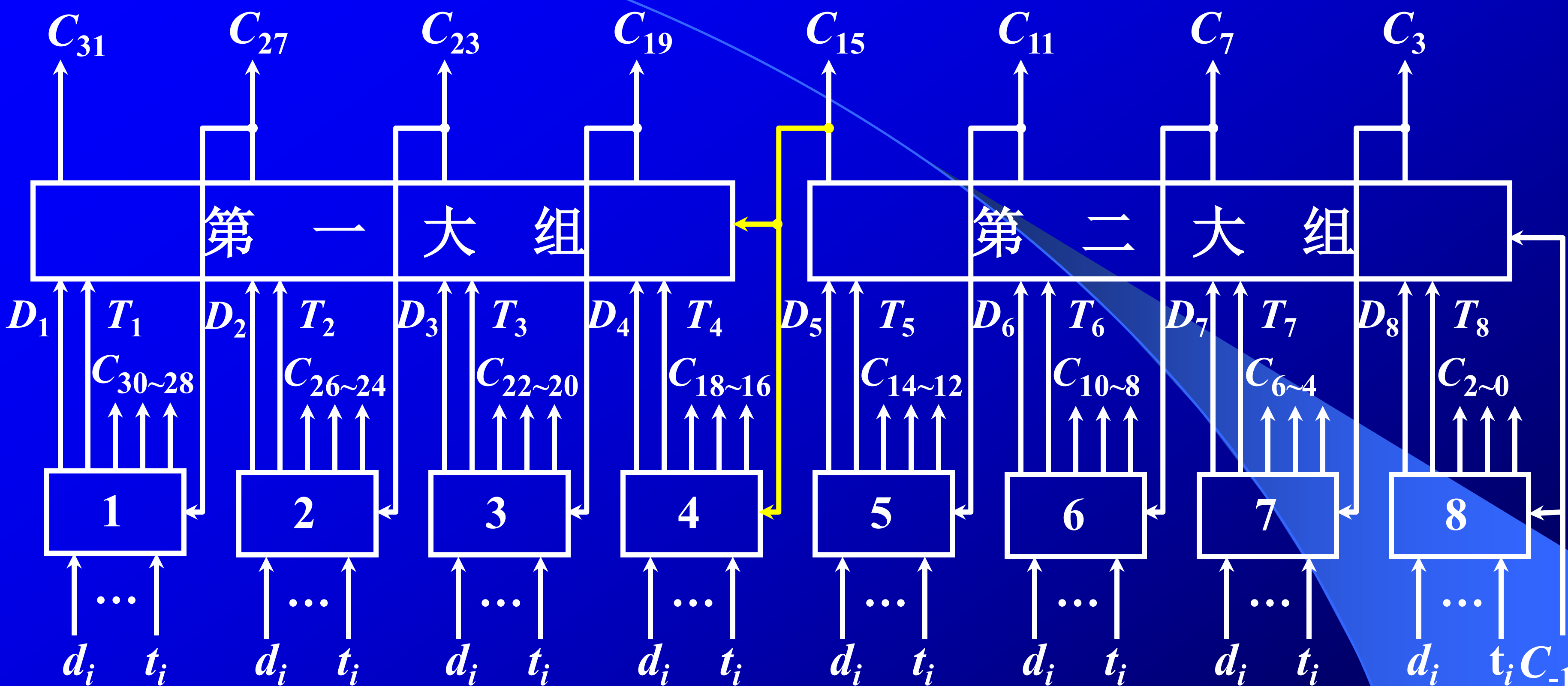
经 5 t_y 产生 C_{15}、C_{11}、C_7、C_3

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 3 2 t_y 产生 全部进位

单重分组跳跃进位链 经 1 0 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}

1 0 t_y　产生 $C_{30}\sim C_{28}$、$C_{26}\sim C_{24}$、$C_{22}\sim C_{20}$

小结：

算术逻辑单元ALU：
实现基本的加减运算和逻辑运算

1. 加法运算时所有定点和浮点运算（加减乘除）的基础，加法速度至关重要
2. 进位方式是影响加法速度的重要因素
3. 并行进位方式能加快加法速度：串行进位加法器（RCA） vs 超前进位加法器（CLA、单重分组、双重分组）
4. 通过“本地进位生成”和“进位传递”函数，让各进位独立、并行产生

作业

- 习题：6.27，6.31

- 提交截止时间：4月28日