续表

被除数(余数)	商	说　明
00.0101	10	$[R]_{补}$与$[S_y]_{补}$异号,上商“0”
00.1010	10	←1 位
+ 11.0011		$+[S_y]_{补}$
11.1101	101	$[R]_{补}$与$[S_y]_{补}$同号,上商“1”
11.1010	101	←1 位
+ 00.1101		$+[-S_y]_{补}$
00.0111	1010	$[R]_{补}$与$[S_y]_{补}$异号,上商“0”
+ 00.1110	1010$\underline{1}$	←1 位,末位商恒置“1”

所以
$$\left[\frac{S_x}{S_y}\right]_{补}=1.0101$$

③ 规格化:

尾数相除结果已为规格化数。

所以
$$\left[\frac{x}{y}\right]_{补}=00,010;\ 11.0101$$

则
$$\left[\frac{x}{y}\right]=2^{010}\times(-0.1011)=\left[2^{2}\times\left(-\frac{11}{16}\right)\right]$$

6.4.3　浮点运算所需的硬件配置

由于浮点运算分阶码和尾数两部分,因此浮点运算器的硬件配置比定点运算器的复杂。分析浮点四则运算发现,对于阶码只有加减运算,对于尾数则有加、减、乘、除四种运算。可见浮点运算器主要由两个定点运算部件组成。一个是阶码运算部件,用来完成阶码加、减,以及控制对阶时小阶的尾数右移次数和规格化时对阶码的调整;另一个是尾数运算部件,用来完成尾数的四则运算以及判断尾数是否已规格化,此外,还需有判断运算结果是否溢出的电路等。

现代计算机可把浮点运算部件做成独立的选件,或称协处理器,用户可根据需要选择,不用选件的机器,也可用编程的方法来完成浮点运算,不过这将会影响机器的运算速度。

例如,Intel 80287 是浮点协处理器,它可与 Intel 80286 或 80386 微处理器配合处理浮点数的算术运算和多种函数计算。

6.5　算术逻辑单元

针对每一种算术运算,都必须有一个相对应的基本硬件配置,其核心部件是加法器和寄存

器。当需要完成逻辑运算时，势必需要配置相应的逻辑电路，而 ALU 电路是既能完成算术运算又能完成逻辑运算的部件。

6.5.1　ALU 电路

图 6.16 所示是 ALU 框图。图中 A_i 和 B_i 为输入变量；k_i 为控制信号，k_i 的不同取值可决定该电路做哪一种算术运算或哪一种逻辑运算；F_i 是输出函数。

现在 ALU 电路已制成集成电路芯片，例如，74181 是能完成 4 位二进制代码的算逻运算部件，外特性如图 6.17 所示。

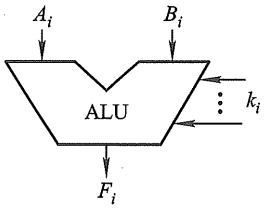

图 6.16　ALU 框图

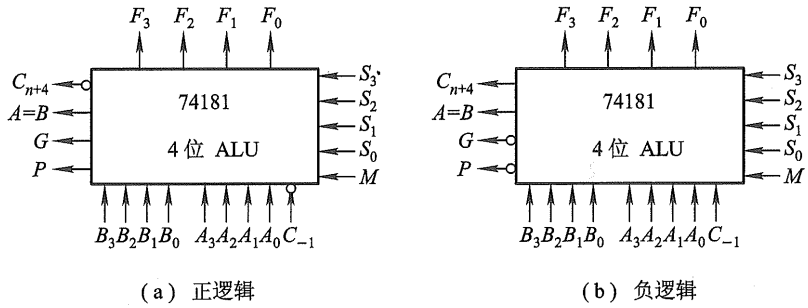

图 6.17　74181 外特性示意图

74181 有两种工作方式，即正逻辑和负逻辑，分别如图 6.17(a)和图 6.17(b)所示。表 6.32 列出了其算术/逻辑运算功能，逻辑电路参见附录 6C 的图 6.30。

表 6.32　74181 ALU 的算术/逻辑运算功能表

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 $(M=0)(C_{-1}=1)$
0 0 0 0	$\overline{A}$	A 减 1	$\overline{A}$	A
0 0 0 1	$\overline{AB}$	AB 减 1	$\overline{A+B}$	$A+B$
0 0 1 0	$\overline{A}+B$	$A\overline{B}$ 减 1	$\overline{A}B$	$A+\overline{B}$
0 0 1 1	逻辑 1	减 1	逻辑 0	减 1
0 1 0 0	$\overline{A+B}$	A 加$(A+\overline{B})$	$\overline{AB}$	A 加 $A\overline{B}$
0 1 0 1	$\overline{B}$	AB 加$(A+\overline{B})$	$\overline{B}$	$(A+B)$加 $A\overline{B}$
0 1 1 0	$\overline{A\oplus B}$	A 减 B 减 1	$A\oplus B$	A 减 B 减 1

续表

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 ($M=0$)($C_{-1}=1$)
0 1 1 1	$A+\overline{B}$	$A+\overline{B}$	$A\overline{B}$	$A\overline{B}$ 减 1
1 0 0 0	$\overline{A}B$	A 加($A+B$)	$\overline{A}+B$	A 加 AB
1 0 0 1	$A\oplus B$	A 加 B	$\overline{A\oplus B}$	A 加 B
1 0 1 0	B	$A\overline{B}$ 加($A+B$)	B	($A+\overline{B}$)加 AB
1 0 1 1	$A+B$	$A+B$	AB	AB 减 1
1 1 0 0	逻辑 0	A 加 A^*	逻辑 1	A 加 A^*
1 1 0 1	$A\overline{B}$	AB 加 A	$A+\overline{B}$	($A+B$)加 A
1 1 1 0	AB	$A\overline{B}$ 加 A	$A+B$	($A+\overline{B}$)加 A
1 1 1 1	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 $A^*=2A$。

以正逻辑为例,$B_3\sim B_0$ 和 $A_3\sim A_0$ 是两个操作数,$F_3\sim F_0$ 为输出结果。C_{-1}表示最低位的外来进位,C_{n+4}是 74181 向高位的进位;P、G 可供先行进位使用(有关 P、G 的具体含义参见 6.5.2 节)。M 用于区别算术运算还是逻辑运算;$S_3\sim S_0$ 的不同取值可实现不同的运算。例如,当 $M=1$,$S_3\sim S_0=0110$ 时,74181 做逻辑运算 $A\oplus B$;当 $M=0$,$S_3\sim S_0=0110$ 时,74181 做算术运算。由表 6.32 可见,在正逻辑条件下,$M=0$,$S_3\sim S_0=0110$,且 $C_{-1}=1$ 时,完成 A 减 B 减 1 的操作。若想完成 A 减 B 运算,可使 $C_{-1}=0$。注意,74181 的算术运算是用补码实现的,其中减数的反码是由内部电路形成的,而末位加“1”,则通过 $C_{-1}=0$ 来体现(图 6.17(a)中 C_{-1}输入端处有一个小圈,意味着 $C_{-1}=0$ 反相后为 1)。尤其要注意的是,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。现在有的芯片将寄存器和 ALU 电路集成在一个芯片内,如 29C101,如图 6.18 所示(图中 ALU 的控制端 $I_8\sim I_0$ 未画出)。

该芯片的核心部件是一个容量为 16 字的双端口 RAM 和一个高速 ALU 电路。

RAM 可视为由 16 个寄存器组成的寄存器堆。只要给出 A_i 口或 B_i 口的 4 位地址,就可以从 A_o 出口或 B_o 出口读出对应于口地址的存储单元内容。写入时,只能写入由 B_i 口指定的那个单元内。参与操作的两个数分别由 RAM 的 A_o、B_o 出口输出至两个锁存器中。

ALU 受 $I_8\sim I_0$ 控制,I_2、I_1、I_0 控制 ALU 的数据源;I_5、I_4、I_3 控制 ALU 所能完成的 3 种算术运算和 5 种逻辑运算;$I_8\sim I_6$ 用来控制 RAM 和 Q 移位器,决定是否移位以及 Y 口输出是来自 RAM 的 A 出口还是 ALU 的 F 出口。

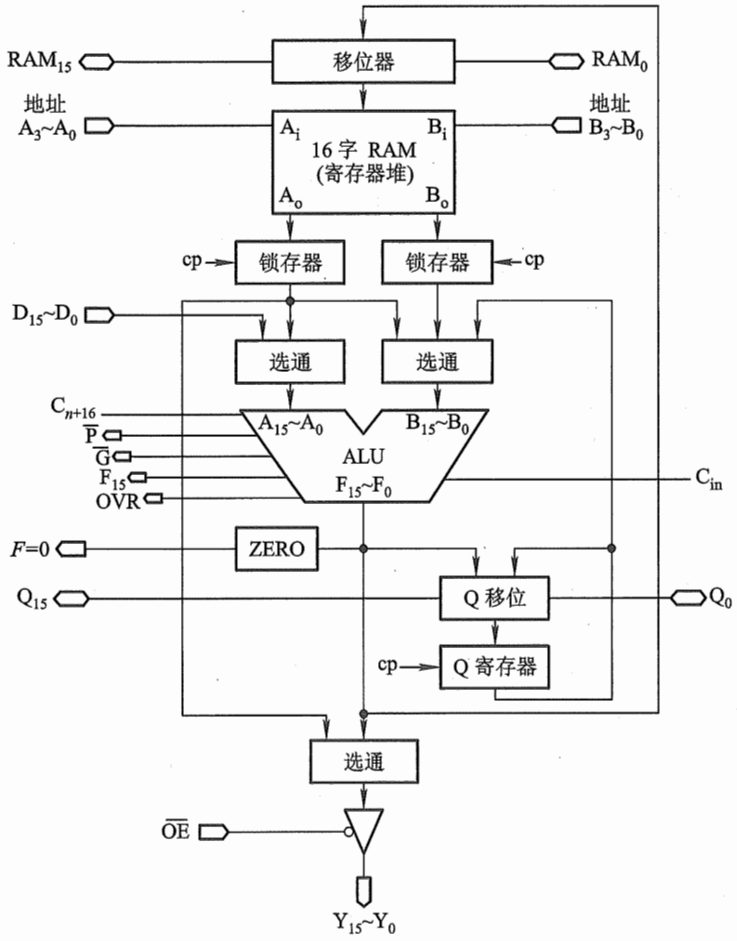

图 6.18　29C101 框图

ALU 的 C_{in}为低位来的外来进位，C_{n+16}为向高位的进位，可供 29C101 级联时用。ALU 结果为 0 时，F=0 可直接输出，OVR 为溢出标记。而 $\overline{P}$、$\overline{G}$ 与 74181 的P、G含义相同，它们可供先行进位方式时使用。ALU 的输出可直接通过移位器存入 RAM，也可通过选通门在 $\overline{OE}$ 有效时从Y_{15}~Y_0 输出。Q 寄存器主要为乘法和除法服务，D_{15}~D_0 为 16 位立即数的输入口。

6.5.2　快速进位链

随着操作数位数的增加，电路中进位的速度对运算时间的影响也越来越大，为了提高运算速度，本节将通过对进位过程的分析设计快速进位链。

1. 并行加法器

并行加法器由若干个全加器组成，如图 6.19 所示。$n+1$ 个全加器级联就组成了一个 $n+1$ 位的并行加法器。

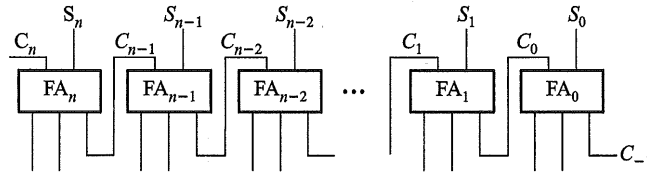

图 6.19　并行加法器示意图

由于每位全加器的进位输出是高一位全加器的进位输入，因此当全加器有进位时，这种一级一级传递进位的过程将会大大影响运算速度。

由全加器的逻辑表达式可知：

和　$S_i=\overline{A}_i\,\overline{B}_iC_{i-1}+\overline{A}_iB_i\,\overline{C}_{i-1}+A_i\,\overline{B}_i\,\overline{C}_{i-1}+A_iB_iC_{i-1}$

进位$C_i=\overline{A}_iB_iC_{i-1}+A_i\,\overline{B}_iC_{i-1}+A_iB_i\,\overline{C}_{i-1}+A_iB_iC_{i-1}$

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

可见，C_i 进位由两部分组成：本地进位 A_iB_i，可记作 d_i，与低位无关；传递进位 $(A_i+B_i)C_{i-1}$，与低位有关，可称 A_i+B_i 为传递条件，记作 t_i，则

$$C_i=d_i+t_iC_{i-1}$$

由 C_i 的组成可以将逐级传递进位的结构转换为以进位链的方式实现快速进位。目前进位链通常采用串行和并行两种。

2. 串行进位链

串行进位链是指并行加法器中的进位信号采用串行传递，图 6.19 所示就是一个典型的串行进位的并行加法器。

以四位并行加法器为例，每一位的进位表达式可表示为

$$\left.\begin{aligned}C_0&=d_0+t_0C_{-1}\\C_1&=d_1+t_1C_0\\C_2&=d_2+t_2C_1\\C_3&=d_3+t_3C_2\end{aligned}\right\}\tag{6.22}$$

由式(6.22)可见，采用与非逻辑电路可方便地实现进位传递，如图 6.20 所示。

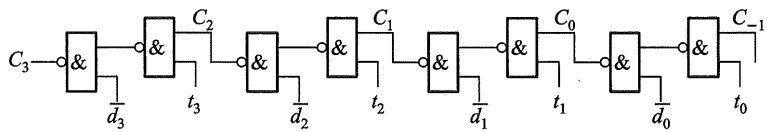

图 6.20　四位串行进位链

若设与非门的级延迟时间为 t_y，那么当 d_i、t_i 形成后，共需 $8t_y$ 便可产生最高位的进位。实际上每增加一位全加器，进位时间就会增加 $2t_y$。n 位全加器的最长进位时间为 $2nt_y$。

3. 并行进位链

并行进位链是指并行加法器中的进位信号是同时产生的，又称先行进位、跳跃进位等。理想的并行进位链是 n 位全加器的 n 位进位同时产生，但实际实现有困难。通常并行进位链有单重分组和双重分组两种实现方案。

(1) 单重分组跳跃进位

单重分组跳跃进位就是将 n 位全加器分成若干小组，小组内的进位同时产生，小组与小组之间采用串行进位，这种进位又有组内并行、组间串行之称。

以四位并行加法器为例，对式(6.22)稍做变换，便可获得并行进位表达式：

$$\left.\begin{aligned} C_0 &= d_0 + t_0 C_{-1} \\ C_1 &= d_1 + t_1 C_0 = d_1 + t_1 d_0 + t_1 t_0 C_{-1} \\ C_2 &= d_2 + t_2 C_1 = d_2 + t_2 d_1 + t_2 t_1 d_0 + t_2 t_1 t_0 C_{-1} \\ C_3 &= d_3 + t_3 C_2 = d_3 + t_3 d_2 + t_3 t_2 d_1 + t_3 t_2 t_1 d_0 + t_3 t_2 t_1 t_0 C_{-1} \end{aligned}\right\} \tag{6.23}$$

按式(6.23)可得与其对应的逻辑图，如图 6.21 所示。

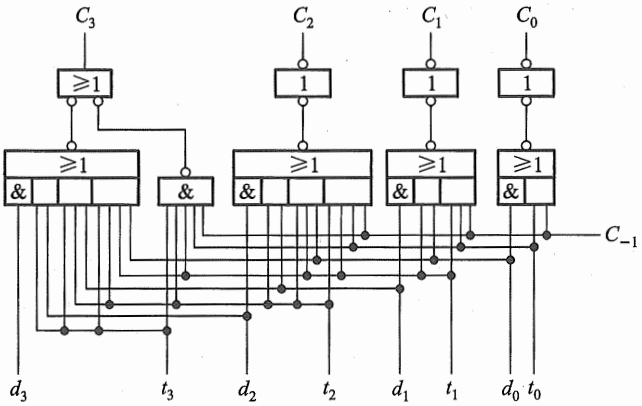

图 6.21　四位一组并行进位链

设与或非门的级延迟时间为 $1.5t_y$，与非门的级延迟时间仍为 $1t_y$，则 d_i、t_i 形成后，只需 $2.5t_y$ 就可产生全部进位。

如果将 16 位的全加器按 4 位一组分组，便可得单重分组跳跃进位链框图，如图 6.22 所示。

不难理解在 d_i、t_i 形成后，经 $2.5t_y$ 可产生 C_3、C_2、C_1、C_0 这 4 个进位信息，经 $10t_y$ 就可产生全部进位，而 $n=16$ 的串行进位链的全部进位时间为 $32t_y$，可见单重分组方案进位时间仅为串行进位链的 1/3。但随着 n 的增大，其优势便很快减弱。例如，当 $n=64$ 时，按 4 位分组，共为 16 组，组间有 16 位串行进位，在 d_i、t_i 形成后，还需经 $40t_y$ 才能产生全部进位，显然进位时间太长。如果

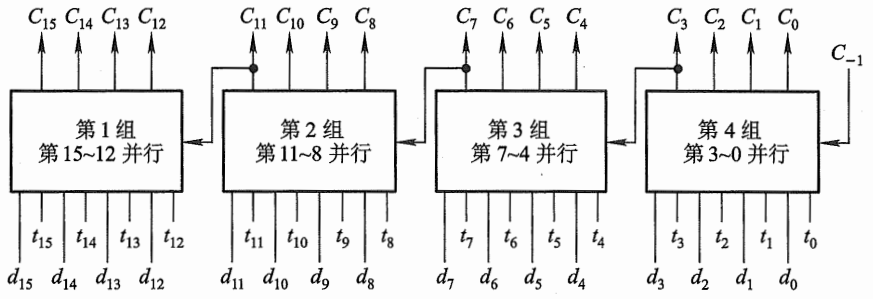

图 6.22　单重分组跳跃进位链框图

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。

（2）双重分组跳跃进位

双重分组跳跃进位就是将 n 位全加器分成若干大组，每个大组中又包含若干小组，而每个大组内所包含的各个小组的最高位进位是同时产生的，大组与大组间采用串行进位。因各小组最高位进位是同时形成的，小组内的其他进位也是同时形成的（注意：小组内的其他进位与小组的最高位进位并不是同时产生的），故又有组（小组）内并行、组（小组）间并行之称。图 6.23 是一个 32 位并行加法器双重分组跳跃进位链的框图。

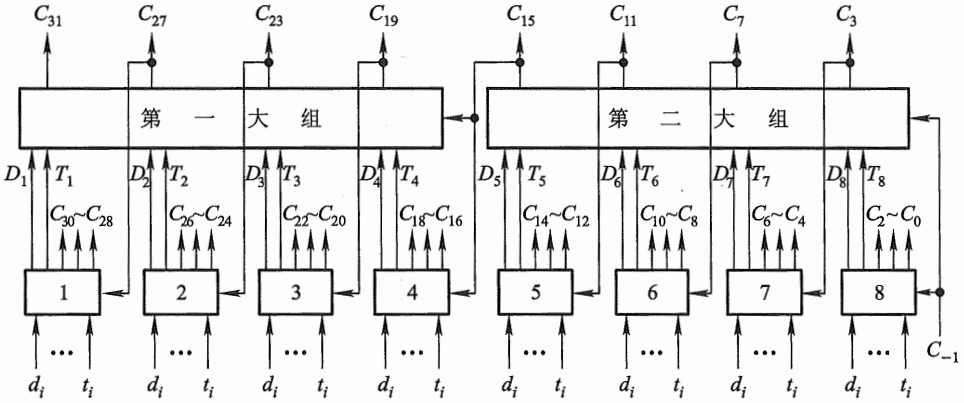

图 6.23　32 位并行加法器双重分组跳跃进位链框图

图中共分两大组，每个大组内包含 4 个小组，第一大组内的 4 个小组的最高位进位 C_{31}、C_{27}、C_{23}、C_{19}是同时产生的；第二大组内 4 个小组的最高位进位 C_{15}、C_{11}、C_7、C_3 也是同时产生的，而第二大组向第一大组的进位 C_{15}采用串行进位方式。

以第二大组为例，分析各进位的逻辑关系。

按式(6.23)，可写出第八小组的最高位进位表达式

$$C_3 = d_3 + t_3C_2 = d_3 + t_3d_2 + t_3t_2d_1 + t_3t_2t_1d_0 + t_3t_2t_1t_0C_{-1}$$
$$= D_8 + T_8C_{-1}$$

式中，$D_8 = d_3 + t_3d_2 + t_3t_2d_1 + t_3t_2t_1d_0$，仅与本小组内的 d_i、t_i 有关，不依赖外来进位 C_{-1}，故称 D_8 为第八小组的本地进位；$T_8 = t_3t_2t_1t_0$，是将低位进位 C_{-1} 传到高位小组的条件，故称 T_8 为第八小组的传送条件。

同理，可写出第五、六、七小组的最高位进位表达式：

$$\left.\begin{aligned}
&\text{第七小组 } C_7 = d_7 + t_7d_6 + t_7t_6d_5 + t_7t_6t_5d_4 + t_7t_6t_5t_4C_3 \\
&\qquad = D_7 + T_7C_3 \\
&\text{第六小组 } C_{11} = d_{11} + t_{11}d_{10} + t_{11}t_{10}d_9 + t_{11}t_{10}t_9d_8 + t_{11}t_{10}t_9t_8C_7 \\
&\qquad = D_6 + T_6C_7 \\
&\text{第五小组 } C_{15} = d_{15} + t_{15}d_{14} + t_{15}t_{14}d_{13} + t_{15}t_{14}t_{13}d_{12} + t_{15}t_{14}t_{13}t_{12}C_{11} \\
&\qquad = D_5 + T_5C_{11}
\end{aligned}\right\} \quad (6.24)$$

进一步展开又得

$$\left.\begin{aligned}
&C_3 = D_8 + T_8C_{-1} \\
&C_7 = D_7 + T_7C_3 = D_7 + T_7D_8 + T_7T_8C_{-1} \\
&C_{11} = D_6 + T_6C_7 = D_6 + T_6D_7 + T_6T_7D_8 + T_6T_7T_8C_{-1} \\
&C_{15} = D_5 + T_5C_{11} = D_5 + T_5D_6 + T_5T_6D_7 + T_5T_6T_7D_8 + T_5T_6T_7T_8C_{-1}
\end{aligned}\right\} \quad (6.25)$$

可见，式(6.25)和式(6.23)极为相似，因此，只需将图 6.21 中的 d_0、d_1、d_2、d_3 改为 D_8、D_7、D_6、D_5，又将 t_0、t_1、t_2、t_3 改为 T_8、T_7、T_6、T_5 便可构成第二重跳跃进位链，即大组跳跃进位链，如图 6.24所示。

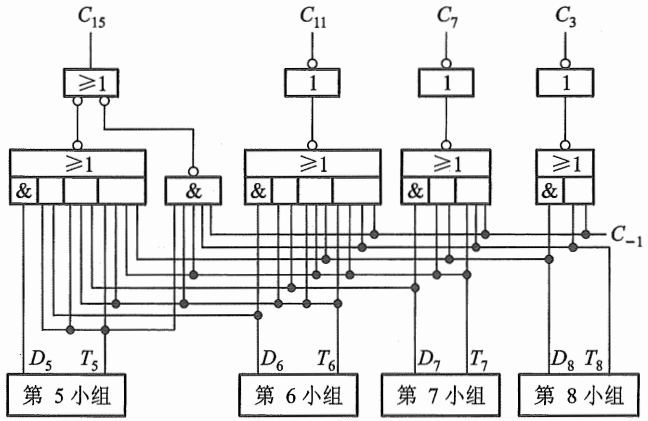

图 6.24　双重分组跳跃进位链的大组进位线路

由图可见，当 D_i、T_i($i=5\sim8$)及外来进位 C_{-1}形成后，再经过 $2.5t_y$ 便可同时产生 C_{15}、C_{11}、C_7、C_3。至于 D_i 和 T_i 可由式(6.24)求得，它们都是由小组产生的，按其逻辑表达式可画出相应的电路。实际上只需对图 6.21 略做修改便可得双重分组进位链中的小组进位链线路，该线路能产生 D_i 和 T_i，如图 6.25 所示。

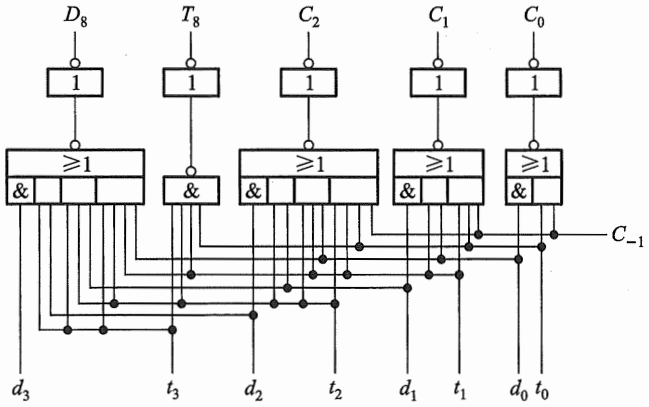

图 6.25　双重分组跳跃进位链的小组进位线路

可见，每小组可产生本小组的本地进位 D_i 和传送条件 T_i 以及组内的各低位进位，但不能产生组内最高位进位，即第五组形成 D_5、T_5、C_{14}、C_{13}、C_{12}，不产生 C_{15}；第六组形成 D_6、T_6、C_{10}、C_9、C_8，不产生 C_{11}；第七组形成 D_7、T_7、C_6、C_5、C_4，不产生 C_7；第八组形成 D_8、T_8、C_2、C_1、C_0，不产生 C_3。

图 6.24 和图 6.25 两种类型的线路可构成 16 位加法器的双重分组跳跃进位链框图，如图 6.26所示。

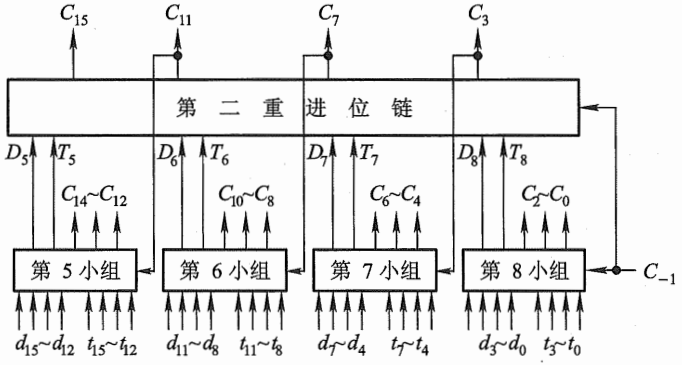

图 6.26　16 位并行加法器的双重分组跳跃进位链框图

由图 6.24、图 6.25 和图 6.26 可计算出从 d_i、t_i 及 C_{-1}(外来进位)形成后开始，经 $2.5t_y$ 形成 C_2、C_1、C_0 和全部 D_i、T_i；再经 $2.5t_y$ 形成大组内的 4 个进位 C_{15}、C_{11}、C_7、C_3；再经过 $2.5t_y$ 形成第

五、六、七小组的其余进位 C_{14}、C_{13}、C_{12}、C_{10}、C_9、C_8、C_6、C_5、C_4。可见，按双重分组设计 $n=16$ 的进位链，最长进位时间为 $7.5t_y$，比单重分组进位链又省了 $2.5t_y$。

对应图 6.23 所示的 32 位加法器的双重分组进位链，不难理解从 d_i、t_i、C_{-1}形成后算起，经 $2.5t_y$ 产生 C_2、C_1、C_0 及 $D_1 \sim D_8$、$T_1 \sim T_8$；再经 $2.5t_y$ 后产生 C_{15}、C_{11}、C_7、C_3；再经 $2.5t_y$ 后产生 $C_{18} \sim C_{16}$、$C_{14} \sim C_{12}$、$C_{10} \sim C_8$、$C_6 \sim C_4$ 及 C_{31}、C_{27}、C_{23}、C_{19}；最后经 $2.5t_y$ 产生 $C_{30} \sim C_{28}$、$C_{26} \sim C_{24}$、$C_{22} \sim C_{20}$。由此可见，产生全部进位的最长时间为 $10t_y$。若采用单重分组进位链，仍以 4 位一组分组，则产生全部进位时间为 $20t_y$，比双重分组多一倍。显然，随着 n 的增大，双重分组的优越性显得格外突出。

机器究竟采用哪种方案，每个小组内应包含几位，应根据运算速度指标及所选元件等诸方面因素综合考虑。

由上述分析可知，D_i 和 T_i 均是由小组进位链产生的，它们与低位进位无关。而 D_i 和 T_i 又是大组进位链的输入，因此，引入 D_i 和 T_i 可采用双重分组进位链，大大提高了运算速度。

6.5.1 节介绍的 74181 芯片是 4 位 ALU 电路，其 4 位进位是同时产生的，多片 74181 级联就犹如本节介绍的单重分组跳跃进位，即组内（74181 片内）并行，组间（74181 片间）串行。74181 芯片的 G、P 输出就如本节介绍的 D、T。当需要进一步提高进位速度时，将 74181 与 74182 芯片配合，就可组成双重分组跳跃进位链，如图 6.27 所示。

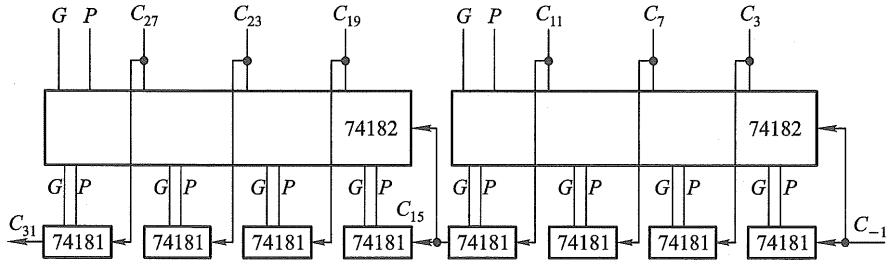

图 6.27　由 74181 和 74182 组成双重分组跳跃进位链

图中 74182 为先行进位部件，两片 74182 和 8 片 74181 组成 32 位 ALU 电路，该电路采用双重分组先行进位方案，原理与图 6.23 类似，不同点是 74182 还提供了大组的本地进位 G 和大组的传送条件 P。

思考题与习题

6.1　最少用几位二进制数即可表示任一 5 位长的十进制正整数？

6.2　已知 $X=0.a_1a_2a_3a_4a_5a_6$（a_i 为 0 或 1），讨论下列几种情况时 a_i 各取何值。

（1）$X>\dfrac{1}{2}$

(2) $X \geqslant \frac{1}{8}$

(3) $\frac{1}{4} \geqslant X > \frac{1}{16}$

6.3 设 x 为整数，$[x]_{补}=1, x_1x_2x_3x_4x_5$，若要求 $x<-16$，试问 $x_1 \sim x_5$ 应取何值？

6.4 设机器数字长为8位(含1位符号位在内)，写出对应下列各真值的原码、补码和反码。

$$-\frac{13}{64}, \frac{29}{128}, 100, -87$$

6.5 已知 $[x]_{补}$，求 $[x]_{原}$ 和 x。

$[x]_{补}=1.1100$；$[x]_{补}=1.1001$；$[x]_{补}=0.1110$；$[x]_{补}=1.0000$；

$[x]_{补}=1,0101$；$[x]_{补}=1,1100$；$[x]_{补}=0,0111$；$[x]_{补}=1,0000$。

6.6 设机器数字长为8位(含1位符号位在内)，分整数和小数两种情况讨论真值 x 为何值时，$[x]_{补}=[x]_{原}$ 成立。

6.7 设 x 为真值，x^* 为绝对值，说明 $[-x^*]_{补}=[-x]_{补}$ 能否成立。

6.8 若 $[x]_{补}>[y]_{补}$，讨论是否有 $x>y$。

6.9 当十六进制数9BH和FFH分别表示为原码、补码、反码、移码和无符号数时，所对应的十进制数各为多少(设机器数采用1位符号位)？

6.10 在整数定点机中，设机器数采用1位符号位，写出±0的原码、补码、反码和移码，得出什么结论？

6.11 已知机器数字长为4位(含1位符号位)，写出整数定点机和小数定点机中原码、补码和反码的全部形式，并注明其对应的十进制真值。

6.12 设浮点数格式为：阶码5位(含1位阶符)，尾数11位(含1位数符)。写出 $\frac{51}{128}$、$-\frac{27}{1024}$、7.375、-86.5所对应的机器数。要求如下：

(1) 阶码和尾数均为原码。

(2) 阶码和尾数均为补码。

(3) 阶码为移码，尾数为补码。

6.13 浮点数格式同上题，当阶码基值分别取2和16时：

(1) 说明2和16在浮点数中如何表示。

(2) 基值不同对浮点数什么有影响？

(3) 当阶码和尾数均用补码表示，且尾数采用规格化形式，给出两种情况下所能表示的最大正数和非零最小正数真值。

6.14 设浮点数字长为32位，欲表示±6万间的十进制数，在保证数的最大精度条件下，除阶符、数符各取1位外，阶码和尾数各取几位？按这样分配，该浮点数溢出的条件是什么？

6.15 什么是机器零？若要求全0表示机器零，浮点数的阶码和尾数应采用什么机器数形式？

6.16 设机器数字长为16位，写出下列各种情况下它能表示的数的范围。设机器数采用1位符号位，答案均用十进制数表示。

(1) 无符号数。

(2) 原码表示的定点小数。

(3) 补码表示的定点小数。

(4) 补码表示的定点整数。

(5) 原码表示的定点整数。

(6) 浮点数的格式为:阶码 6 位(含 1 位阶符),尾数 10 位(含 1 位数符)。分别写出正数和负数的表示范围。

(7) 浮点数格式同(6),机器数采用补码规格化形式,分别写出其对应的正数和负数的真值范围。

6.17 设机器数字长为 8 位(含 1 位符号位),对下列各机器数进行算术左移一位、两位,算术右移一位、两位,讨论结果是否正确。

$[x]_{原}=0.0011010;[x]_{补}=0.1010100;[x]_{反}=1.0101111;$

$[x]_{原}=1.1101000;[x]_{补}=1.1101000;[x]_{反}=1.1101000;$

$[x]_{原}=1.0011001;[x]_{补}=1.0011001;[x]_{反}=1.0011001。$

6.18 试比较逻辑移位和算术移位。

6.19 设机器数字长为 8 位(含 1 位符号位),用补码运算规则计算下列各题。

(1) $A=\frac{9}{64},B=-\frac{13}{32}$,求 $A+B$。

(2) $A=\frac{19}{32},B=-\frac{17}{128}$,求 $A-B$。

(3) $A=-\frac{3}{16},B=\frac{9}{32}$,求 $A+B$。

(4) $A=-87,B=53$,求 $A-B$。

(5) $A=115,B=-24$,求 $A+B$。

6.20 用原码一位乘、两位乘和补码一位乘(Booth 算法)、两位乘计算 $x\cdot y$。

(1) $x=0.110111,y=-0.101110$。

(2) $x=-0.010111,y=-0.010101$。

(3) $x=19,y=35$。

(4) $x=0.11011,y=-0.11101$。

6.21 用原码加减交替法和补码加减交替法计算 $x\div y$。

(1) $x=0.100111,y=0.101011$。

(2) $x=-0.10101,y=0.11011$。

(3) $x=0.10100,y=-0.10001$。

(4) $x=\frac{13}{32},y=-\frac{27}{32}$。

6.22 设机器数字长为 16 位(含 1 位符号位),若一次移位需 1 μs,一次加法需 1 μs,试问原码一位乘、补码一位乘、原码加减交替除和补码加减交替法最多各需多少时间?

6.23 画出实现 Booth 算法的运算器框图。要求如下:

(1) 寄存器和全加器均用方框表示,指出寄存器和全加器的位数。

(2) 说明加和移位的次数。

(3) 详细画出最低位全加器的输入电路。

(4) 描述 Booth 算法重复加和移位的过程。

6.24 画出实现补码加减交替法的运算器框图。要求如下:

(1) 寄存器和全加器均用方框表示,指出寄存器和全加器的位数。

(2) 说明加和移位的次数。

(3) 详细画出第5位(设 n 为最低位)全加器的输入电路。

(4) 画出上商的输入电路。

(5) 描述商符的形成过程。

6.25 对于尾数为40位的浮点数(不包括符号位在内),若采用不同的机器数表示,试问当尾数左规或右规时,最多移位次数各为多少?

6.26 按机器补码浮点运算步骤计算 $[x\pm y]_{补}$。

(1) $x=2^{-011}\times 0.101100$, $y=2^{-010}\times(-0.011100)$。

(2) $x=2^{-011}\times(-0.100010)$, $y=2^{-010}\times(-0.011111)$。

(3) $x=2^{101}\times(-0.100101)$, $y=2^{100}\times(-0.001111)$。

6.27 假设阶码取3位,尾数取6位(均不包括符号位),计算下列各题。

(1) $[2^{5}\times\frac{11}{16}]+[2^{4}\times(-\frac{9}{16})]$。

(2) $[2^{-3}\times\frac{13}{16}]-[2^{-4}\times(-\frac{5}{8})]$。

(3) $[2^{3}\times\frac{13}{16}]\times[2^{4}\times(-\frac{9}{16})]$。

(4) $[2^{6}\times(-\frac{11}{16})]\div[2^{3}\times(-\frac{15}{16})]$。

(5) $[2^{3}\times(-1)]\times[2^{-2}\times\frac{57}{64}]$。

(6) $[2^{-6}\times(-1)]\div[2^{7}\times(-\frac{1}{2})]$。

(7) 3.3125+6.125。

(8) 14.75−2.4375。

6.28 如何判断定点和浮点补码加减运算结果是否溢出,如何判断原码和补码定点除法运算结果是否溢出?

6.29 设浮点数阶码取3位,尾数取6位(均不包括符号位),要求阶码用移码运算,尾数用补码运算,计算 $x\cdot y$,且结果保留1倍字长。

(1) $x=2^{-100}\times 0.101101$, $y=2^{-011}\times(-0.110101)$。

(2) $x=2^{-011}\times(-0.100111)$, $y=2^{101}\times(-0.101011)$。

6.30 机器数格式同上题,要求阶码用移码运算,尾数用补码运算,计算 $x\div y$。

(1) $x=2^{101}\times 0.100111$, $y=2^{011}\times(-0.101011)$。

(2) $x=2^{110}\times(-0.101101)$, $y=2^{011}\times(-0.111100)$。

6.31 设机器字长为32位,用与非门和与或非门设计一个并行加法器(假设与非门的延迟时间为30 ns,与或非门的延迟时间为45 ns),要求完成32位加法时间不得超过0.6 μs。画出进位链及加法器逻辑框图。

6.32 设机器字长为16位,分别按4、4、4、4和5、5、3、3分组后,要求:

(1) 画出两种分组方案的单重分组并行进位链框图,并比较哪种方案运算速度快。

(2) 画出两种分组方案的双重分组并行进位链,并对这两种方案进行比较。

(3) 用 74181 和 74182 画出单重和双重分组的并行进位链框图。

附录 6A　各种进位制

6A.1　各种进位制的对应关系

表 6.33 是十进制数、二进制数、十六进制数对照表。书写时,可在十六进制数后面加上“H”,如 17DBH 或(17DB)$_{十六}$;若在数的后面加上“B”,如 10101100B,即表示此数为二进制数,或写成(10101100)$_{二}$。

表 6.33　十进制数、二进制数、十六进制数对照表

十进制数	二进制数	十六进制数	十进制数	二进制数	十六进制数
0	00000	0	16	10000	10
1	00001	1	17	10001	11
2	00010	2	18	10010	12
3	00011	3	19	10011	13
4	00100	4	20	10100	14
5	00101	5	21	10101	15
6	00110	6	22	10110	16
7	00111	7	23	10111	17
8	01000	8	24	11000	18
9	01001	9	25	11001	19
10	01010	A	26	11010	1A
11	01011	B	27	11011	1B
12	01100	C	28	11100	1C
13	01101	D	29	11101	1D
14	01110	E	30	11110	1E
15	01111	F	31	11111	1F

6A.2　各种进位制的转换

任意一个数 N 可用下式表示:

$$\begin{aligned} N &= (d_{n-1}d_{n-2}\cdots d_1 d_0 . d_{-1}d_{-2}\cdots d_{-m})_r \\ &= d_{n-1}r^{n-1}+d_{n-2}r^{n-2}+\cdots+d_1 r^1+d_0 r^0+d_{-1}r^{-1}+\cdots+d_m r^{-m} \\ &= \sum_{i=-m}^{n-1} d_i r^i \end{aligned}$$

其中,r 为基值;n、m 为正整数,分别代表整数位和小数位的位数;d_i 为系数,代表第 i 位的一个数码,可以是 0~(r−1)数码中的任意一个;r^i 为第 i 位的权数。

1. 二进制数转换成十进制数

(1) 按“权”展开法

例如$(11011.1)_{二}=1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0+1\times2^{-1}$

$=27.5$

(2) 按基值重复相乘(除)法

- 整数部分采用基值重复相乘法

例如$(101001)_{二}=((((1\times2+0)\times2+1)\times2+0)\times2+0)\times2+1=41$

- 小数部分采用基值重复相除

例如$(0.1011)_{二}=((((1\div2)+1)\div2+0)\div2+1)\div2+0=0.6895$

2. 十进制数转换成二进制数

(1) 重复相除(乘)法

这种方法的规则是整数部分除以2取余数,直到商为0止;小数部分乘以2取整数,直到小数部分为0止(或按精度要求确定位数)。

例　将十进制数123.6875转换成二进制数。

解:整数部分

重复除以2	得商	取余数	
123÷2	61	1	最低位
61÷2	30	1	
30÷2	15	0	
15÷2	7	1	
7÷2	3	1	
3÷2	1	1	
1÷2	0	1	最高位

故整数部分$(123)_{十}=(1111011)_{二}$。

小数部分

重复乘以2	得小数部分	取整数	
0.6875×2	0.3750	1	最高位
0.3750×2	0.7500	0	
0.7500×2	0.5000	1	
0.5000×2	0.0000	1	最低位