表 4.2　自然顺序与倒位序二进制数对照表及相应的十进制数

自然顺序十进制数(I)	自然顺序二进制数	倒位序二进制数	倒位序后十进制数(J)
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

I,J 都是从 0 开始。若已知某个倒位序数 J，要求下一个倒位序数，则应先判断 J 的最高位是否为 0，这可与 $K=N/2$ 相比较，因为 $N/2$ 总是等于 100…的。如果 $K>J$ 则 J 的最高位为 0，只要把该位变为 1（J 与 $K=N/2$ 相加即可），就得到了下一个倒位序数；如果 $K\leqslant J$，则 J 的最高位为 1，可将最高位变为 0（J 减去 $N/2$ 即可）。然后还需判断次高位，这可与 $K=N/4$ 相比较，若次高位是 0，则将它改成 1（J 与 $N/4$ 相加即可），其他位不变，即得到下一个倒位序数；若次高位是 1，则需将它也变为 0（J 减去 $N/4$ 即可）。然后还需判断再下一位，这可与 $K=N/8$ 相比较……依次进行，总会碰到某位为 0（除非最后一个数 $N-1$ 的各位全是 1，而这个数不需倒位序，不属于倒位序程序内的数），这时把这个 0 改成 1，就得到下一个倒位序数。求出新的倒位序数 J 以后，当 $I<J$ 时，进行变址交换。注意，在倒位序中，$x(0)$ 和 $x(N-1)$ 总是不需要交换的，因为 0 与 $N-1$ 的倒位序数与原自然顺序数是一样的。

4.8　N 为复合数的 FFT 算法——混合基（多基多进制）FFT 算法

学习要点

1. 若序列长度不满足 $N=2^L$，前面说过，可以补零值点来满足这一要求，但有可能增加的计算量太长，例 $N=280$ 点，则需补到 $2^8=512$ 点共须补 232 个零值点。

2. 若要求准确的 N 点，则只能采用直接 DFT 方法，或采用后面将介绍的 CZT（Chirp-z 变换）算法。

3. 若 N 是一个组合数，可以分解为一些因子的乘积，则可采用混合基 FFT 算法，或称多基多进制 FFT 算法。

4. 二进制、多进制（r 进制）及混合基（或称多基多进制）FFT 算法

(1) 二进制 FFT。当 $N=2^L$，则任一个 $n<N$ 的正整数 n 可以表示成以 2 为基数的二进制形式

$$(n)_2=(n_{L-1}n_{L-2}\cdots n_1 n_0),\quad n_i=0\text{ 或 }1,\quad i=0,1,\cdots,L-1 \tag{4.8.1}$$

此二进制数所代表的数值为

$$(n)_{10}=n_{L-1}2^{L-1}+n_{L-2}2^{L-2}+\cdots+n_1 2+n_0 \tag{4.8.2}$$

此二进制倒位序后为 $(\overline{n})_2=(n_0\, n_1\cdots n_{L-2}n_{L-1})$，它所代表的数值为

$$(\bar{n})_{10} = n_0 2^{L-1} + n_1 2^{L-2} + \cdots + n_{L-2} 2 + n_{L-1} \tag{4.8.3}$$

(2) r 进制（多进制）FFT。当 $N=r^L$，r、L 皆为大于 1 的正整数，任一个 $n<N$ 的正整数 n 都可表示为以 r 为基数的 r 进制形式

$$(n)_r = (n_{L-1} n_{L-2} \cdots n_1 n_0), \quad n_i = 0,1,\cdots,r-1, \quad i = 0,1,\cdots,L-1 \tag{4.8.4}$$

此 r 进制数所代表的数值为

$$(n)_{10} = n_{L-1} r^{L-1} + n_{L-2} r^{L-2} + \cdots + n_1 r + n_0 \tag{4.8.5}$$

此 r 进制倒位序后为 $(\bar{n})_r=(n_0,n_1,\cdots,n_{L-2},n_{L-1})$，它们代表的数值为

$$(\bar{n})_{10} = n_0 r^{L-1} + n_1 r^{L-2} + \cdots + n_{L-2} r + n_{L-1} \tag{4.8.6}$$

当 $r=2$ 时，就是二进制，$r=4$ 时，就是四进制，等等。

(3) 多基多进制 FFT，即混合基 FFT。

当 $N=r_0 r_1 \cdots r_{L-1}$，各 $r_i(i=0,1,\cdots,L-1)$ 为大于 1 的正整数，则任一个 $n<N$ 的正整数 n，可以表示为多基多进形式

$$(n)_{r_0 r_1 \cdots r_{L-1}} = (n_{L-1} n_{L-2} \cdots n_1 n_0) \tag{4.8.7}$$

$$(n)_{10} = (r_0 r_1 \cdots r_{L-2}) n_{L-1} + (r_0 r_1 \cdots r_{L-3}) n_{L-2} + \cdots + (r_0 r_1) n_2 + r_0 n_1 + n_0 \tag{4.8.8}$$

其倒位序后为

$$(\bar{n})_{r_0 r_1 \cdots r_{L-1}} = (n_0 n_1 \cdots n_{L-2} n_{L-1}) \tag{4.8.9}$$

$$(\bar{n})_{10} = (r_1 r_2 \cdots r_{L-1}) n_0 + (r_2 r_3 \cdots r_{L-1}) n_1 + \cdots + (r_{L-2} r_{L-1}) n_{L-3} + r_{L-1} n_{L-2} + n_{L-1} \tag{4.8.10}$$

在 $(n)_{10}$ 及 $(\bar{n})_{10}$ 的表示式中[即(4.8.8)式及(4.8.10)式]，各 n_i 的取值范围为

$$\begin{aligned} n_{L-1} &= 0,1,\cdots,r_{L-1}-1 \\ n_{L-2} &= 0,1,\cdots,r_{L-2}-1 \\ &\vdots \\ n_1 &= 0,1,\cdots,r_1-1 \\ n_0 &= 0,1,\cdots,r_0-1 \end{aligned} \tag{4.8.11}$$

可记为 $n_i=0,1,\cdots,r_i-1, \quad i=0,1,\cdots,L-1$

这种对序列进行抽取的办法，因为是使其 FFT 算法仍能保持原位（同址）运算的特点，将这种抽取造成的“混序”组合数情况称为广义倒位序。

* 多基多进制的广义倒位序与基-2 或基-r 倒位序的区别在于，当低位与高位颠倒交换时，其相应的加权系数也产生变化，而基-2（或基-r）在某一位处，其加权系数是固定的（例如基-2，在从左到右第二位，加权系数就是 2^{L-2}，不管是 n 或是 $\bar{n}$，处于此位的加权系数都是这一数值）。多基多进制正序排列从左到右第一位 n_{L-1} 的加权系数是 $r_0 r_1 \cdots r_{L-2}$，而倒位序排列同样的第一位 n_0 的加权系数则是 $r_1 r_2 \cdots r_{L-1}$。

* 由于以上组合数做分解时可以有多种方式，例如可有 $N=r_0 r_1 \cdots r_{L-1}$，$N=r_{L-1} r_{L-2} \cdots r_0$ 等等，故混合基 FFT 算法可有多种形式流图，多种形式算法。

【例 4.1】 试用混合基 FFT 算法求 $N=30$ 的结果，并画出流图。

解 (1) 这里采用 $N=r_0 r_1 r_2=5\times2\times3$，即 $r_0=5$，$r_1=2$，$r_2=3$ 的混合基 FFT 算法，采用输入 n 按正序排列，输出 k 按倒位序排列的办法，则有

$L=3$

例 4×4=16.

$$\left.\begin{aligned} n &= r_0r_1n_2 + r_0n_1 + n_0 = 10n_2 + 5n_1 + n_0 \\ \bar{k} &= r_1r_2k_0 + r_2k_1 + k_2 = 6k_0 + 3k_1 + k_2 \end{aligned}\right\} \tag{4.8.12}$$

其中各 n_i, k_i 的取值范围为

$$\left.\begin{aligned} n_0, k_0 &= 0,1,\cdots,r_0-1 = 0,1,2,3,4 \\ n_1, k_1 &= 0,1,\cdots,r_1-1 = 0,1 \\ n_2, k_2 &= 0,1,\cdots,r_2-1 = 0,1,2 \end{aligned}\right\} \tag{4.8.13}$$

(2) 列出混合基运算的表达式$[x(n)=x(10n_2+5n_1+n_0)]$

$$\begin{aligned} X(k) &= \sum_{n=0}^{29} x(n)W_{30}^{nk} = \sum_{n_0=0}^{4}\sum_{n_1=0}^{1}\sum_{n_2=0}^{2} x(n_2,n_1,n_0)W_{30}^{(10n_2+5n_1+n_0)(6k_0+3k_1+k_2)} \\ &= \sum_{n_0=0}^{4}\sum_{n_1=0}^{1}\sum_{n_2=0}^{2} x(n_2,n_1,n_0)W_{30}^{10n_2k_2}W_{30}^{15n_1k_1}W_{30}^{5n_1k_2}W_{30}^{6n_0k_0}W_{30}^{3n_0k_1}W_{30}^{n_0k_2} \end{aligned}$$

在以上推导中，已应用到 $W_{30}^{60n_2k_0}=W_{30}^{30n_1k_0}=1$，再考虑到 $W_N^{N_1n_ik_i}=W_{N/N_1}^{n_ik_i}$（$N/N_1$＝整数时）则有

$$\begin{aligned} X(k) &= \sum_{n_0=0}^{4}\sum_{n_1=0}^{1}\left[\sum_{n_2=0}^{2} x(n_2,n_1,n_0)W_3^{n_2k_2}\right]W_{30}^{(5n_1+n_0)k_2}W_2^{n_1k_1}W_{30}^{3n_0k_1}W_5^{n_0k_0} \\ &= \sum_{n_0=0}^{4}\sum_{n_1=0}^{1}\left[X_1(k_2,n_1,n_0)W_{30}^{(5n_1+n_0)k_2}\right]W_2^{n_1k_1}W_{10}^{n_0k_1}W_5^{n_0k_0} \\ &= \sum_{n_0=0}^{4}\left[\sum_{n_1=0}^{1} X_1'(k_2,n_1,n_0)W_2^{n_1k_1}\right]W_{10}^{n_0k_1}W_5^{n_0k_0} \\ &= \sum_{n_0=0}^{4}\left[X_2(k_2,k_1,n_0)W_{10}^{n_0k_1}\right]W_5^{n_0k_0} \\ &= \sum_{n_0=0}^{4} X_2'(k_2,k_1,n_0)W_5^{n_0k_0} \\ &= X(k_2,k_1,k_0) \end{aligned} \tag{4.8.14}$$

其中

$$X_1(k_2,n_1,n_0) = \sum_{n_2=0}^{2} x(n_2,n_1,n_0)W_3^{n_2k_2} \tag{4.8.15}$$

$$X_1'(k_2,n_1,n_0) = X_1(k_2,n_1,n_0)W_{30}^{(5n_1+n_0)k_2} \tag{4.8.16}$$

$$X_2(k_2,k_1,n_0) = \sum_{n_1=0}^{1} X_1'(k_2,n_1,n_0)W_2^{n_1k_1} \tag{4.8.17}$$

$$X_2'(k_2,k_1,n_0) = X_2(k_2,k_1,n_0)W_{10}^{n_0k_1} \tag{4.8.18}$$

$$X(k_2,k_1,k_0) = \sum_{n_0=0}^{4} X_2'(k_2,k_1,k_0)W_5^{n_0k_0} = X(k) = X(6k_0+3k_1+k_2) \tag{4.8.19}$$

这里看出有三级 DFT 运算，一个是(4.8.15)式的(n_2,k_2)变量的 3 点 DFT 共有 10 个，一个是(4.8.17)式的(n_1,k_1)变量的 2 点 DFT，共有 15 个，一个是(4.8.19)式的(n_0,k_0)变量的 5 点 DFT，共有 6 个，此外还有第一级 3 点 DFT 运算之后的乘 $W_{30}^{(5n_1+n_0)k}$ 运算，见(4.8.16)式，以及第二级 2 点 DFT 运算之后的乘 $W_{10}^{n_0k_1}$ 运算，见(4.8.18)式，这两个乘因子都只影响输出序列的相位，不影响幅度，故被称为旋转因子。有时，将这种算法称为旋转因子算法，对于

基-2 FFT 是混合基算法的特定情况，因而也是旋转因子算法。

按以上算式可画出 $N=5\times2\times3=30$ 的混合基 FFT 流图（输入正序、输出倒位序）如图 4.23 所示，图中第一级的 10 个 3 点 DFT 只画出了$[x(0),x(10),x(20)]$及$[x(4),x(14),x(24)]$两个。

0,1,2 间隔10 间隔5,1 0~4

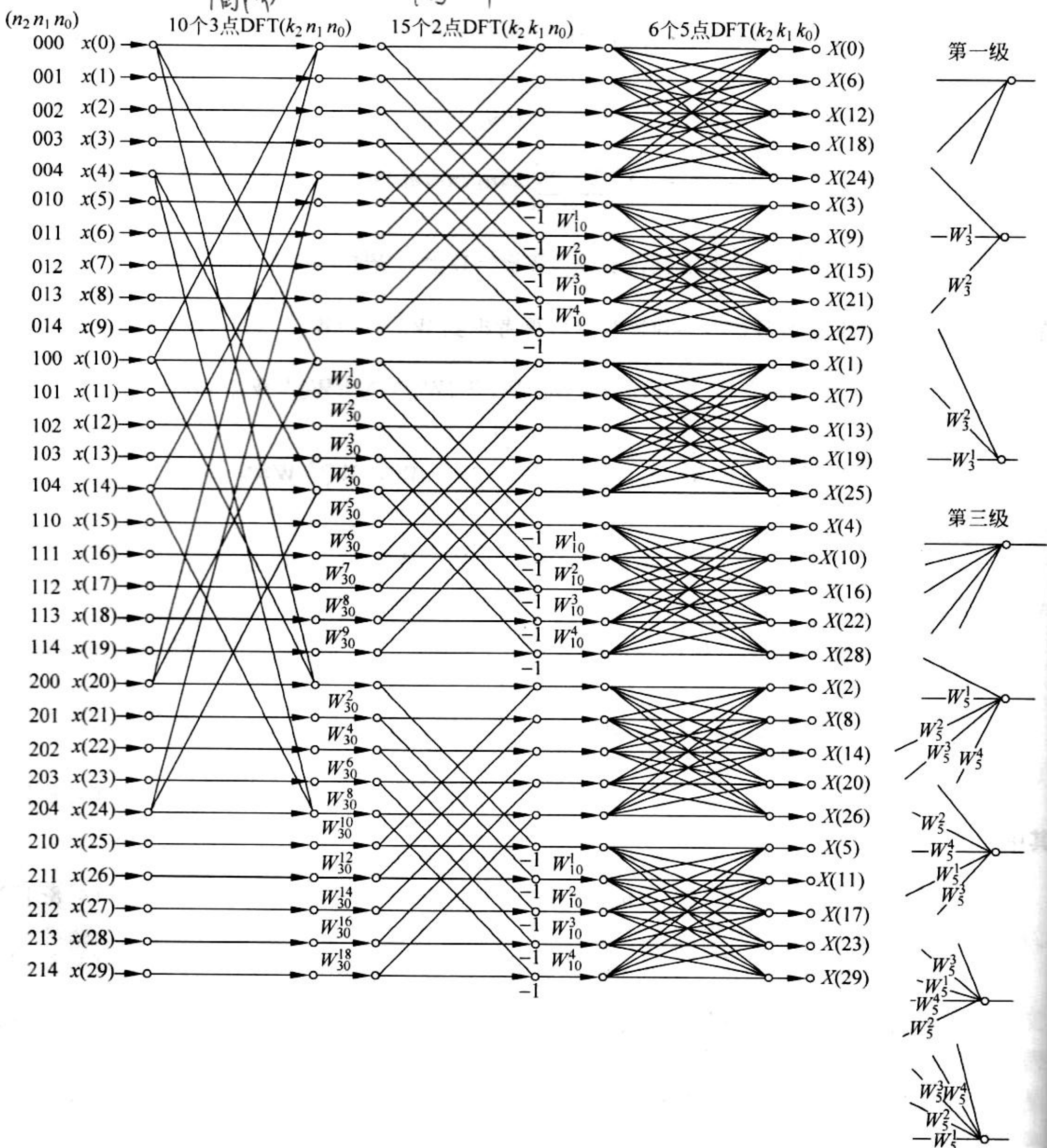

图 4.23 $N=5\times2\times3=30$ 的混合基 FFT 流图，输入正序、输出倒位序 $X(k)=X(6k_0+3k_1+k_2)$

5. N 为复合数时 FFT 运算量的估计。

当 $N=r_0r_1$ 时，如果不算倒位序的工作量，其运算量为

$$\text{直接法求 } r_1 \text{ 个 } r_0 \text{ 点 DFT：}\begin{cases}\text{复数乘法} & r_1r_0^2\\ \text{复数加法} & r_1r_0(r_0-1)\end{cases}$$

乘 N 个旋转因子：复数乘法 N

直接法求 r_0 个 r_1 点 DFT：$\begin{cases} 复数乘法 & r_0 r_1^2 \\ 复数加法 & r_0 r_1 (r_1-1) \end{cases}$

总计：

$$\left.\begin{aligned} &复数乘法 \quad r_1 r_0^2 + N + r_0 r_1^2 = N(r_0+r_1+1) \\ &复数加法 \quad r_1 r_0(r_0-1) + r_0 r_1(r_1-1) = N(r_0+r_1-2) \end{aligned}\right\} \tag{4.8.20}$$

而直接计算一个 N 点 DFT 的运算量为

复数乘法 $\qquad N^2$

复数加法 $\qquad N(N-1)$

因而混合基算法可节省的运算量倍数为

$$\left.\begin{aligned} &乘法 \quad R_\times = \frac{N^2}{N(r_0+r_1+1)} = \frac{N}{r_0+r_1+1} \\ &加法 \quad R_+ = \frac{N(N-1)}{N(r_0+r_1-2)} = \frac{N-1}{r_0+r_1-2} \end{aligned}\right\} \tag{4.8.21}$$

例如，当 $N=r_0 r_1=5\times 7=35$ 时

$$R_\times = \frac{35}{15} = 2.6$$

直接 DFT 算法等于混合基算法的 2.6 倍工作量。

同样，当 $N=r_0 r_1 r_2$ 时，一定有 $r_1 r_2$ 个 r_0 点 DFT，$r_0 r_2$ 个 r_1 点 DFT，$r_0 r_1$ 个 r_2 点 DFT，加上两次乘旋转因子，因而总乘法次数为 $N(r_0+r_1+r_2+2)$。

这样可以推算出，当 $N=r_0 r_1 \cdots r_{L-1}$ 时，采用混合基算法所需总乘法次数为

$$N\left[\left(\sum_{i=0}^{L-1} r_i\right) + L - 1\right] \tag{4.8.22}$$

则直接计算 DFT 与之相比，运算量之比为

$$R_\times = \frac{N^2}{N\left[\left(\sum\limits_{i=0}^{L-1} r_i\right) + L - 1\right]} = \frac{N}{L-1+\sum\limits_{i=0}^{L-1} r_i} \tag{4.8.23}$$

注意(4.8.22)式用于每个 r_i 均为素数（但 $\neq 2$）的情况是精确的，此时可将 r_i 点变换看成乘法次数为 r_i^2 是对的，但是当 r_i 不是素数或 $r_i=2$ 时，就不一定对了，例如：$r_i=2$ 时，是两点变换不带有乘法运算，对 $r_i=4$ 也是这样，对 $r_i=8$，则所需运算比 64 次乘法少得多，所以分解成 r_i 为 2，4，8，将使上面公式几乎失效，也就是说 $N=2^L$ 时，上面(4.8.23)式完全不适用。

4.9 线性调频 z 变换（Chirp-z 变换或 CZT）算法

DFT 运算的局限性是：①z 平面单位圆上的 N 个抽样点必须均匀等间隔地分布在[0，2π)范围中，且输入、输出序列长度都必须是相同的 N 点。若只需要计算某一频段的频谱值，例如对窄带信号，且希望提高计算的分辨率，即在窄的频带内，有更密集的抽样点，例如，