第7卷 第1期 电子科学学刊 Vol. 7, No. 1
1985 年 1 月 JOURNAL OF ELECTRONICS Jan., 1985

树形分解 FFT 算法*

汪凯仁
(复旦大学)

提 要

本文对 K. Nakayama 提出的时间-频率混合抽选 FFT 算法作了简化和深化，提出了树形分解 FFT 算法，其实数乘法次数与 K. Nakayama 的方法相比，由 $(3/2)N\log_2 N - 7N + 10N^{1/2} - 4$ 减少到约为 $(65/64)N\log_2 N - 3N - 4$. 该算法并未改变 FFT 算法的基本结构，用软件和硬件实现不致有太大的变化.

一、引 言

自从 1965 年 J. Cooley 和 J. Tukey[1] 提出 FFT 算法以来，已出现了多种多样的具体结构，基本上可分成两大类：时间抽选型（DIT）和频率抽选型（DIF）. 利用旋转因子 w_N^k 中 $w_N^0 = 1$、$w_N^{N/4} = -j$ 和 $w_N^{N/8} = \frac{\sqrt{2}}{2}(1-j)$ 等几个特殊复数值，可以节省很多实数乘法运算次数，而且基底 4 的算法比起基底 2 的算法来可以节省更多. 在 DIT 型算法中，第 1、第 2 轮蝶形运算实际上不需要做乘法，以后各轮实质性的乘法运算逐步增加，到最后一轮最多. 而在 DIF 型算法中情况恰恰相反，第 1 轮应做最多的乘法，到最后两轮则没有乘法. 1979年，日本的 K. Nakayama(中山谦二)把 DIT 和 DIF 组合起来，提出了一种更有效的混合抽选 FFT 算法 (MDFFT)[2]. 对于 $N = 2^L$，L 为偶数的情况，使前 $L/2$ 轮蝶形运算按 DIT 进行，继而乘上一组旋转因子，然后使后 $L/2$ 轮

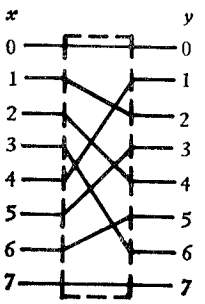

图 1 时间抽选重排 (TDS)

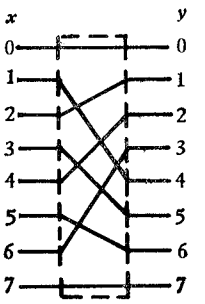

图 2 频率抽选重排 (FDS)

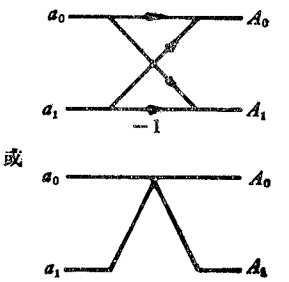

图 3 蝶形运算

* 1982 年 9 月 28 日收到，1984 年 8 月 27 日修改定稿.

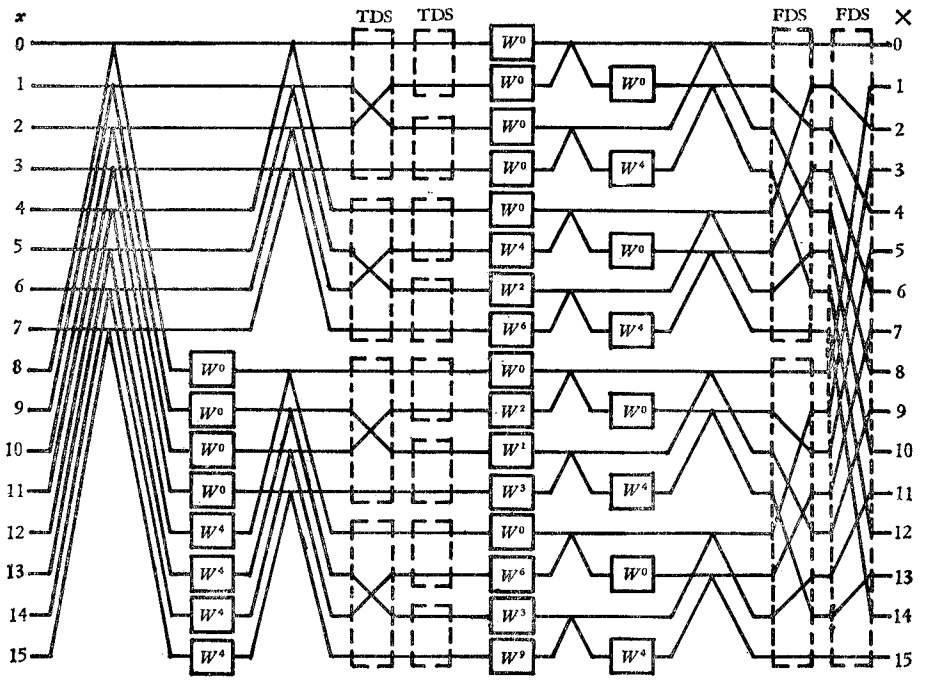

图 4　16 点 MDFFT ($w = w_{16}$)

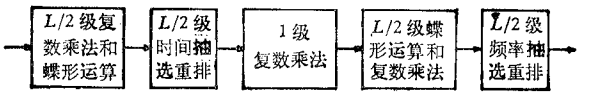

图 5　MDFFT 的结构

按 DIF 进行，得以节省乘法运算次数．中间还有一些为 DIT 和 DIF 过程所必须的数据重排过程（Shuffle），见图 1—5．其实数乘法运算次数为 $\frac{3}{2}N\log_2 N-7N+10N^{1/2}-4$．在文献 [3] 中用类似于文献 [1] 的方法给出了这个算法的一种具体推导．

二、混合抽选 FFT 算法的简化解释

众所周知，FFT 的核心思想是利用 $N = P \cdot Q$ 时，N 点 DFT 可以用抽选的方法缩减运算次数．就是在

$$X(m) = \sum_{k=0}^{N-1} x(k) w_N^{km}, \qquad 0 \leqslant m < N \tag{1}$$

中，取

$$\begin{aligned} m &= m_1 P + m_0, \quad 0 \leqslant m_1 < Q, \quad 0 \leqslant m_0 < P; \\ k &= k_1 Q + k_0, \quad 0 \leqslant k_1 < P, \quad 0 \leqslant k_0 < Q; \end{aligned} \tag{2}$$

则有

$$\begin{aligned} X(m) &= X(m_1P + m_0) \\ &= \sum_{k_0=0}^{Q-1} \sum_{k_1=0}^{P-1} x(k_1Q + k_0) w_N^{(m_1P+m_0)(k_1Q+k_0)} \\ &= \sum_{k_0=0}^{Q-1} \left[w_N^{m_0k_0} \sum_{k_1=0}^{P-1} x(k_1Q + k_0) w_P^{m_0k_1} \right] w_Q^{m_1k_0}. \end{aligned} \tag{3}$$

因此N点 DFT 的计算可以分解为Q个P点 DFT，继而乘旋转因子 $w_N^{m_0k_0}(0 \leqslant m_0 < P, 0 \leqslant k_0 < Q)$，然后是$P$个$Q$点 DFT. 这里$m$实际上是用一个特殊的二位数$m = (m_1m_0)$表示，其低位$m_0$是$P$进制的，高位$m_1$是$Q$进制的；$k$也用一个二位数$k = (k_1k_0)$表示，但低位$k_0$是$Q$进制的，高位$k_1$是$P$进制的. 如果$N$个输入数据顺序存放，所有的$P$点和$Q$点 DFT 运算都将计算结果顺序存放于原处(in-place)，则在第 $k = (k_1k_0) = k_1Q + k_0$ 单元(存一个复数的单元)处求得的结果是 $X(m_0Q + m_1)$，与所需的 $X(m_1P + m_0)$ 正好相差一个广义的反序过程.

通常的基底 2 的 FFT 算法用 $2^L = 2^{L-1} \cdot 2$，$2^{L-1} = 2^{L-2} \cdot 2 \cdots$的方法逐步分解，可以得到 DIT 型 FFT 算法(即 Cooley-Tukey 算法)[1]. 在L轮蝶形运算中，旋转因子乘法中可节省的部分逐轮减少. 如果用 $2^L = 2 \cdot 2^{L-1}$，$2^{L-1} = 2 \cdot 2^{L-2} \cdots$的方法逐步分解，则得到 DIF 型 FFT 算法(即 Sande-Tukey 算法)[4]，旋转因子乘法中可节省的部分逐轮增加.

现在来考虑一般的 $N = P \cdot Q$ 的场合. 当P和Q都是 4 的倍数时，我们把N的旋转因子排成矩阵形式如 (4) 式，

$$\begin{bmatrix} w^0 & w^0 & \cdots & w^0 & \cdots & w^0 & \cdots & w^0 & \cdots & w^0 \\ w^0 & w^1 & \cdots & w^{\frac{P}{4}} & \cdots & w^{\frac{P}{2}} & \cdots & w^{\frac{3P}{4}} & \cdots & w^{P-1} \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ w^0 & w^{\frac{Q}{4}} & \cdots & w^{\frac{P}{4}\cdot\frac{Q}{4}} & \cdots & w^{\frac{P}{2}\cdot\frac{Q}{4}} & \cdots & w^{\frac{3P}{4}\cdot\frac{Q}{4}} & \cdots & w^{(P-1)\frac{Q}{4}} \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ w^0 & w^{\frac{Q}{2}} & \cdots & w^{\frac{P}{4}\cdot\frac{Q}{2}} & \cdots & w^{\frac{P}{2}\cdot\frac{Q}{2}} & \cdots & w^{\frac{3P}{4}\cdot\frac{Q}{2}} & \cdots & w^{(P-1)\frac{Q}{2}} \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ w^0 & w^{\frac{3Q}{4}} & \cdots & w^{\frac{P}{4}\cdot\frac{3Q}{4}} & \cdots & w^{\frac{P}{2}\cdot\frac{3Q}{4}} & \cdots & w^{\frac{3P}{4}\cdot\frac{3Q}{4}} & \cdots & w^{(P-1)\frac{3Q}{4}} \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ w^0 & w^{Q-1} & \cdots & w^{\frac{P}{4}(Q-1)} & \cdots & w^{\frac{P}{2}(Q-1)} & \cdots & w^{\frac{3P}{4}(Q-1)} & \cdots & w^{(P-1)(Q-1)} \end{bmatrix} \tag{4}$$

可见其中有 $P + Q - 1$ 项为 $w_N^0 = 1$，不必做复数乘法；一项为 $w_N^{\frac{P}{2}\cdot\frac{Q}{2}} = -j$，也不必做乘法；四项为

$$w_N^{\frac{P}{4}\cdot\frac{Q}{4}} = w_N^{\frac{P}{4}\cdot\frac{Q}{2}} = \frac{\sqrt{2}}{2}(1 - j),$$

$$w_N^{\frac{P}{2}\cdot\frac{3Q}{4}} = w_N^{\frac{3P}{4}\cdot\frac{Q}{2}} = \frac{\sqrt{2}}{2}(-1 - j),$$

每当乘这种旋转因子只要做 2 次实数乘法和 2 次实数加法;其余的 $PQ-P-Q-4$ 个旋转因子乘法以每次复数乘法需要 3 次实数乘法和 3 次实数加法计（见附录 1). 所以，乘上全部旋转因子所需要的实数乘法次数和实数加法次数是

$$3(PQ-P-Q-4)+2\cdot 4=3N-3P-3Q-4. \tag{5}$$

当 P、Q 中有一个或两个仅是 2 的倍数而不是 4 的倍数时，运算次数仍有所节省. 对于恰有一个仅是 2 的倍数的场合，类似地可得实数乘法次数和实数加法次数都是

$$3(PQ-P-Q-2)+2\cdot 2=3N-3P-3Q-2. \tag{6}$$

当 $P=Q=N^{1/2}$ 是 4 的倍数，所有 $2N^{1/2}$ 个 $N^{1/2}$ 点 FFT 都采用基底 2 的 FFT 算法执行时，我们来计算总的运算次数. 已知M点 FFT 需要实数乘法 $\frac{3}{2}M\log_2 M-5M+8$ 次，实数加法次数为实数乘法次数加上 $2M\log_2 M$，(见附录 2). 这时总的实数乘法次数为

$$\begin{aligned}&(3N-3N^{1/2}-3N^{1/2}-4)+2N^{1/2}\left(\frac{3}{2}N^{1/2}\log_2 N^{1/2}-5N^{1/2}+8\right)\\&=\frac{3}{2}N\log_2 N-7N+10N^{1/2}-4.\end{aligned} \tag{7}$$

实数加法次数为

$$\begin{aligned}&(3N-3N^{1/2}-3N^{1/2}-4)+2N^{1/2}\left(\frac{7}{2}N^{1/2}\log_2 N^{1/2}-5N^{1/2}+8\right)\\&=\frac{7}{2}N\log_2 N-7N+10N^{1/2}-4,\end{aligned} \tag{8}$$

仍是实数乘法次数加上 $2N\log_2 N$.

上面算出的实数乘法次数跟 Nakayama 的结果完全一致. 但必须指出，这里不仅大大简化了 Nakayama 的原来工作中的繁琐推导，而且实际上也已改变了他所提出的“时间-频率混合抽选”的结构. 因为在我们的推导中，在乘一组旋转因子的前后各有 $N^{\frac{1}{2}}=2^{\frac{L}{2}}$ 个 $2^{\frac{L}{2}}$ 点的 DFT，可用任何基底 2 的 FFT 实现，并未要求前 $2^{\frac{L}{2}}$ 个 FFT 用时间抽选型，后面 $2^{\frac{L}{2}}$个 FFT 用频率抽选型，如同图 5 和图 4 要求的那样. 也就是说，例如在图 4 中，左边一列中的 0、4、8、12 这 4 点需作 4 点 DFT，不是非用时间抽选型不可，也可用频率抽选型，并不影响总的实数乘法次数.

另一方面，在 Nakayama 所提出的算法中两组数据重排 (Shuffle) 过程也不是这类方法的内在需要. 因为作一次 $N=N^{\frac{1}{2}}\cdot N^{\frac{1}{2}}$ 的分解，如果每个 $N^{\frac{1}{2}}$ 点 DFT 的结果顺序存放在原处，则最后结果与所要求的N点顺序 DFT 值相差一个二位 $N^{\frac{1}{2}}$ 进制数的反序. 如果 $N=2^L$，L为偶数，N的二进制表示为 $(n_{L-1}\cdots n_{L/2}\ n_{(L/2)-1}\cdots n_0)$，则这一反序实际上是

$$(\underbrace{n_{L-1}\cdots n_{L/2}}_{L/2\text{ 位}}\ \underbrace{n_{(L/2)-1}\cdots n_0}_{L/2\text{ 位}})\longleftrightarrow(\underbrace{n_{(L/2)-1}\cdots n_0}_{L/2\text{ 位}}\ \underbrace{n_{L-1}\cdots n_{L/2}}_{L/2\text{ 位}}). \tag{9}$$

如果每个 $N^{\frac{1}{2}}$ 点 DFT 用 FFT 的蝶形运算流程计算，不作任何二进制反序，则所得结果是按二进制反序存放的，即 (9) 式右端的两组 $L/2$ 位都作二进制反序，所得结果为

$(n_0\cdots n_{(L/2)-1}n_{L/2}\cdots n_{L-1})$，就是 $(n_{L-1}\cdots n_{L/2}n_{(L/2)-1}\cdots n_0)$ 的二进制反序. 如同通常的基底 2 的 FFT 算法一样，所需的二进制反序过程可以放在输入后或输出前. 当然也可以用稍复杂的重排过程嵌在中间步骤中，Nakayama 所提出的一组时间抽选重排 (Time Decimation Shuffle)和一组频率抽选重排 (Frequency Decimation Shuffle) 就是一个特例，见图1，2，4，5. 如果这类不同的安排对硬件或集成电路实现有好处的话，应是很有选择余地的.

三、树形分解方法

显然，如果上述的 $N^{\frac{1}{2}}$ 点 DFT 不是直接采用常规 FFT 算法，而继续采取 $N^{\frac{1}{2}}=N^{\frac{1}{4}}\cdot N^{\frac{1}{4}}$ 分解的话，乘法次数应有进一步的节省. 一般地，可考虑 $N=2^{2^r}$ 的场合，逐次分解直至 $16=4\cdot4$，至于 $4=2\cdot2$ 用不用都可以，因为已无乘法. 这样可以获得尽可能多的节省，利用 (5) 式知，总的实数乘法次数为

$$(3N-6N^{\frac{1}{2}}-4)+(2N^{\frac{1}{2}})(3N^{\frac{1}{2}}-6N^{\frac{1}{4}}-4)+(2N^{\frac{1}{2}})(2N^{\frac{1}{4}})(3N^{\frac{1}{4}}-6N^{\frac{1}{8}}-4)$$
$$+\cdots+(2N^{\frac{1}{2}})(2N^{\frac{1}{4}})\cdots(2\cdot16)(3\cdot16-6\cdot4-4)$$
$$=\sum_{j=2}^{r}2^{r-j}2^{(2^r-2^j)}(3\cdot2^{2^j}-6\cdot2^{2^{j-1}}-4). \tag{10}$$

当 $r\geqslant3$ 即 $N\geqslant256$ 时，可算出总的实数乘法次数为等于或略少于

$$\frac{65}{64}N\log_2N-3N-4. \tag{11}$$

至于实数加法次数，因为在旋转因子乘法中实数加法次数与实数乘法次数相等，其余的加法仅出现在

$$(2N^{\frac{1}{2}})(2N^{\frac{1}{4}})\cdot\cdots\cdot(2\cdot16)(2\cdot4)=\frac{1}{8}N\log_2N \tag{12}$$

次 4 点 DFT 中，现在每个 4 点 DFT 需要 16 次实数加法，所以总的实数加法次数仍是总的实数乘法次数加上 $2N\log_2N$.

然而，对 $N=2^{2^r}$ 当 $r=2,3,4$ 时取值为 16、256、65536，接着迅速增长成为天文数字. 看来有实用意义的仅有 256 和 65536 两个，常用的 1024，2048，4096 等都不在此列. 但仍可用相同的思想构造算法以压缩运算次数. 例如对 $N=2048$，按图 6(a) 所示的二叉树逐步分解，可算出实数乘法次数为 16836；按图 6(b) 所示的二叉树逐步分解，实数乘法次数为 18104. 顺便指出，图 6(c) 所示的二叉树对应于 DIT 型 FFT 算法；图 6(d) 则对应于 DIF 型 FFT 算法.

一般，对 $L=\log_2N=l_{r-1}2^{r-1}+l_{r-2}2^{r-2}+\cdots+l_1\cdot2+l_0$，可采用从二进制高位向低位的分解方法，如图 7 所示. 其中当 $l_j=0$ 时，$2^{l_j2^j}$ 处的子树实际上并不存在. 这时要推导实数乘法次数的一般公式是极其麻烦的，但当 N 为 16 的倍数，即 $l_1=l_0=0$ 时，可用归纳法证明乘法次数仍然不超过 $\frac{65}{64}N\log_2N-3N-4$. 对于其他情况，往往稍有超过.

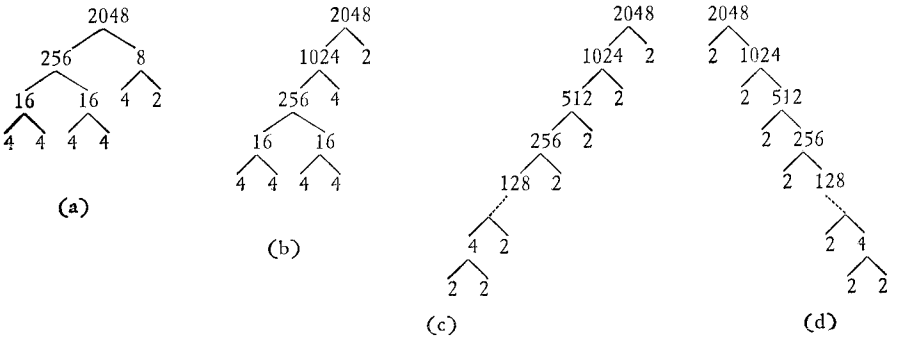

图 6　2048 因数分解的二叉树表示

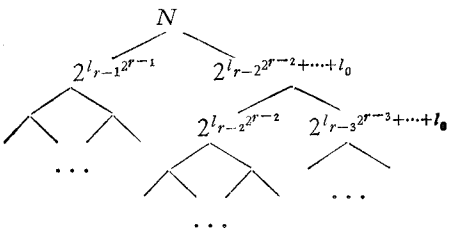

图 7　$N=2^L$ 因数分解的一种二叉树表示

表 1　各种 FFT 算法实数乘法次数的比较

算法	用于减少乘法的旋转因子			实数乘法次数			
				N 点	256点	2048点	65536点
	w_N^0	$w_N^{N/4}$	$w_N^{N/8}$				
基底 2	✓			$\frac{3}{2}N\log_2 N-3N+3$	2307	27651	1376259
	✓	✓		$\frac{3}{2}N\log_2 N-\frac{9}{2}N+6$	1926	24582	1277958
	✓	✓	✓	$\frac{3}{2}N\log_2 N-5N+8$	1800	23560	1245192
基底 4	✓			$\frac{9}{8}N\log_2 N-3N+3$	1539	—	983043
	✓	✓		$\frac{9}{8}N\log_2 N-\frac{13}{4}N+4$	1476	—	966660
	✓	✓	✓	$\frac{9}{8}N\log_2 N-\frac{43}{12}N+\frac{16}{3}$	1392	—	944812
混合抽选	✓	✓	✓	$\frac{3}{2}N\log_2 N-7N+10\sqrt{N}-4$	1436	—	1116668
树形分解 $N=2^{2^r}$	✓	✓	✓	$\leqslant\frac{65}{64}N\log_2 N-3N-4$	1308	—	864764
树形分解 $N=2^L$				—	—	18104* 16836**	—

* 按图 6(b) 分解，** 按图 6(a) 分解.

四、各种算法的比较

树形分解方法的实质在于充分利用 w_N^0, $w_N^{N/4}$, $w_N^{N/8}$ 可以节省运算的特点，尽可能使它们多出现在旋转因子中. 如果不顾这几个旋转因子的特殊性的话，总的实数乘法次数还是 $\frac{3}{2}N\log_2 N$, 一点没有占到便宜. 如果仅利用 $w_N^0=1$，则运算次数与基底 4 的 FFT 相同. 表 1 列出了一系列算法的实数乘法次数，再加上 $2N\log_2 N$ 就都是实数加法次数.

另一方面，按文献 [4] 的分析，基数愈大愈可节省运算，当然结构复杂到了不实用的地步. 树形分解方法可以说是"穷尽"了其中的潜力，但看来其结构还是可实现的，这可结合硬件或线路作进一步的研究.

五、结 束 语

本文将 K. Nakayama 提出的时间-频率混合抽选 FFT 算法推广为树形分解 FFT 算法，它的实数乘法次数降低到约为 $\frac{65}{64}N\log_2 N-3N-4$，但基本结构仍属FFT类型，用软件、硬件或集成电路实现不致有太大的变化.

本文受美国 Princeton 大学刘必治教授讲学的启发而写就，并承刘教授慨允复制文献 [3] 中的附图(本文的图 1—5)，谨致感谢.

附录 1 关于复数乘法

如果 $a+jb$ 是已知常数，则直接计算

$$u+jv=(a+jb)(x+jy)=(ax-by)+j(ay+bx)$$

需要 4 次实数乘法、2 次实数加法. 也可由

$$m_1=a(x+y),\quad m_2=(a+b)y,\quad m_3=(a-b)x,$$
$$u=m_1-m_2,\quad v=m_1-m_3$$

算出，共需要 3 次实数乘法，3 次实数加法. 但这时要有 3 个单元存储常数 a, $a+b$, $a-b$.

附录 2 关于基底 2 的 FFT 算法的实数运算次数 ($N=2^L$)

现仅对 DIT 型的场合计算，对 DIF 型有相同的结论. 这时第 1、第 2 轮蝶形运算没有实质性的实数乘法，从第 3 轮开始，第 j 轮的旋转因子重复 2^{L-j} 组，每组 2^{j-1} 个旋转因子中有 $w_N^0=1$, $w_N^{N/4}=-j$, $w_N^{N/8}=\frac{\sqrt{2}}{2}(1-j)$, $w_N^{3N/8}=\frac{\sqrt{2}}{2}(-1-j)$ 等 4 个特殊值，总的实数乘法次数为

$$\sum_{j=3}^{L} 2^{L-j}[(2^{j-1}\cdot 4)\cdot 3+2\cdot 2]=\frac{3}{2}N\log_2 N-5N+8.$$

在旋转因子乘法中涉及的实数加法次数同样是 $\frac{3}{2}N\log_2 N-5N+8$，加上 $L=\log_2 N$ 轮蝶形运算中的 $2N\log_2 N$ 次实数加法，总的实数加法次数为 $\frac{7}{2}N\log_2 N-5N+8$.

参　考　文　献

[1] J. Cooley and J. Tukey, *Mathematics of Computation,* 19(1965), 297.

[2] K. Nakayama, Fast Fourier Transform Using Mixed Frequency and Time Decimation, IECE of Japan, Report of Technical Group on Circuit Syst., Vol. CAS 79—94, pp. 49—54, Oct. 1979.

[3] C. Caraiscos and B. Liu, Two Dimensional DFT Using Mixed Time and Frequency Decimation, Proc. Int. Conf. Acoust., Speech, Signal Processing, pp. 24—27, Paris, May 1982.

[4] E. O. Brigham, The Fast Fourier Transform, Prentice-Hall, 1974.

TREE DECOMPOSITION FFT ALGORITHM

Wang Kairen

(Fudan University, Shanghai)

The Mixed Time and Frequency Decimation FFT Algorithm (MDFFT) proposed by K. Nakayama is simplified and deepened, then a new FFT algorithm based on tree decomposition process is developed. The number of real multiplications of the new algorithm is about $\frac{65}{64}N\log_2 N-3N-4$ which is less than $\frac{3}{2}N\log_2 N-7N+10\sqrt{N}-4$ of MDFFT.