第10卷 第2期 电 子 科 学 学 刊 Vol. 10, No. 2
1988 年 3 月 JOURNAL OF ELECTRONICS Mar., 1988

时间抽取情况下的旋转因子合并 FFT 算法和 TMFFT 的软件实现*

许 蔚 陈宗骞
(中国科学院电子学研究所,北京)

摘要 马滕斯 (Martens) 提出了一种效率高 (可与 WFTA 法和 PFA 法相比拟)、结构简单 (与 FFT 法相似)的 DFT 计算方法——RCFA. 作者已经证明, 在基 2 的情况下, RCFA 与旋转因子合并的频率抽取 FFT 算法是完全等价的. 本文给出了旋转因子合并的时间抽取 FFT 算法,从而使得在任何条件下,目前使用的 FFT 算法都可以用外部特性完全相同、内部结构基本相同的高效算法——旋转因子合并 FFT 算法来代替. 本文还给出了实现旋转因子合并 FFT 算法的软件.

关键词 FFT 算法;旋转因子合并 FFT 算法;软件实现

一、引 言

马滕斯利用多项式代数理论,提出了一种 DFT 的新算法——递归割圆因式分解算法 (RCFA)[1], 它的计算效率可与 WFTA 和 PFA 相比拟, 而结构与 FFT 没有多大差别,也具有递归形式,因而结构简单,实现方便,是一种比较理想的 DFT 计算方法. 作者已经证明,在基 2 的情况下, 将频率抽取 FFT 进行两级旋转因子合并, 得到的结果与 RCFA 完全相同[2]. 因此,凡是使用基 2 频率抽取 FFT 的信号处理系统,都可以在完全不改变 DFT 变换器外部特性和很少改变 DFT 变换器本身结构的条件下,用更高效的算法来代替原来的 FFT 算法. 那么,效率与频率抽取 FFT 完全相同的时间抽取 FFT 能否被同样高效的算法来代替呢? 本文的回答是肯定的. 这样, 无论是基 2 频率抽取 FFT 算法,还是基 2 时间抽取 FFT 算法,都可以在外部特性完全不变、内部结构很少改变的条件下,用更高效的旋转因子合并 FFT 算法——TMFFT (Twiddle factor Merged FFT) 来代替.

在实际应用中用软件实现 TMFFT 时,只要对 FFT 的实现软件作一些修改, 就可得到 TMFFT 的实现软件.

二、时间抽取的 TMFFT 算法

N点复数序列 $\{x(n)|n=0,1,\cdots,N-1\}$ 的 DFT $\{X(k)|k=0,1,\cdots,N-1\}$

* 1986 年 11 月 22 日收到,1987 年 8 月 28 日修改定稿.

定义为[3]:

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk},\quad k=0,1,\cdots,N-1 \tag{1}$$

其中

$$W_N=e^{-j\frac{2\pi}{N}} \tag{2}$$

类似于频率抽取的情况,在奇偶时分解过程中,对偶时部分只需进行复数加减法,而对奇时部分则还要进行复数乘法. 对奇时部分进行第二级时间抽取后的乘法运算单元的一般形式如图 1 所示.

图 1 所表示的数学关系为:

$$X_1^*(k)=W_N^k[X_{10}(k)+W_{N/2}^kX_{11}(k)] \tag{3}$$

$$X_1^*(N/4+k)=W_N^{N/4+k}[X_{10}(k)-W_{N/2}^kX_{11}(k)] \tag{4}$$

经过旋转因子合并,(3)式和(4)式变为:

$$X_1^*(k)=W_N^kX_{10}(k)+W_N^{3k}X_{11}(k) \tag{5}$$

$$X_1^*(N/4+k)=-j[W_N^kX_{10}(k)-W_N^{3k}X_{11}(k)] \tag{6}$$

这时的复乘运算单元如图 2 所示.

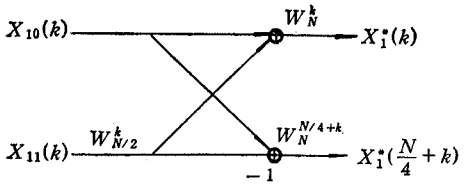

图 1 两级时间抽取中的复乘运算单元

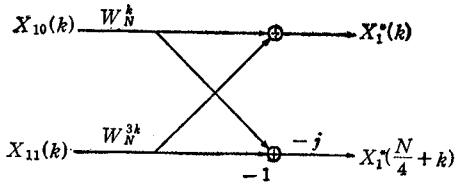

图 2 两级旋转因子合并后的时间抽取复乘运算单元

依次分解下去,一直分解到 2 点和 1 点 DFT,我们就得到了递归形式的时间抽取 FFT 的改进算法——旋转因子合并的时间抽取 FFT 算法. 它与旋转因子合并的频率抽取 FFT 算法一起构成了完整的旋转因子合并 FFT 算法 (TMFFT).

三、时间抽取 TMFFT 的性质

与频率抽取 TMFFT 一样[2],时间抽取 TMFFT 的复乘法次数 $M(t)$ 也满足:

$$M(t)=\frac{1}{3}\cdot 2^t\left(t-\frac{8}{3}\right)+1+\frac{1}{9},\quad t\text{为奇数} \tag{7}$$

$$M(t)=\frac{1}{3}\cdot 2^t\left(t-\frac{8}{3}\right)+1-\frac{1}{9},\quad t\text{为偶数} \tag{8}$$

其中 $N=2^t$ 为 DFT 的点数.

在数字信号处理中,有时只需要计算输入信号的部分频谱,那就需要对 FFT 进行输出修枝[4]. 这时使用时间抽取 FFT 算法就很方便,而使用频率抽取 FFT 算法则不便于进行修枝. 在这种情况下,时间抽取 TMFFT 就特别适用. 反之,如果输入信号序列

只有部分值不为零，那就需要对 FFT 进行输入修枝[5]，这时频率抽取 TMFFT 就特别适用.

四、频率抽取 TMFFT 的软件实现

频率抽取 TMFFT 中的蝶形运算单元可以分为两类：第一类只有复数加减法，没有复数乘法；第二类除了复数加减法之外，在输出端还有复数乘法.

第一类蝶形运算单元的一般数学表示为：

$$x_k(N_k^l+n)=x_{k-1}(N_k^l+n)+x_{k-1}(N_k^l+N_k/2+n),$$
$$n=0,1,\cdots,\ N_k/2-1 \tag{9}$$

$$x_k(N_k^l+N_k/2+n)=x_{k-1}(N_k^l+n)-x_{k-1}(N_k^l+N_k/2+n),$$
$$n=0,1,\cdots,N_k/2-1 \tag{10}$$

其中 N_k^l 表示在第 k 级第 l 段中蝶形运算单元的起始位置，N_k 则表示在第 k 级中每段蝶形运算单元所涉及的数据点总数. 我们将数据按实部与虚部分开表示：

$$x_k(n)=x_{R,k}(n)+jx_{I,k}(n) \tag{11}$$

可以得到第一类蝶形运算单元的实现过程如图 3 所示.

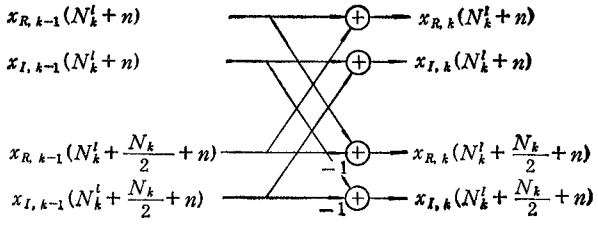

图 3　频率抽取时的第一类蝶形运算单元

第二类蝶形运算单元的一般数学表示为：

$$x_k(N_k^l+n)=[x_{k-1}(N_k^l+n)-jx_{k-1}(N_k^l+N_k/2+n)]W_{N_{k-1}}^{n},$$
$$n=0,1,\cdots,\ N_k/2-1 \tag{12}$$

$$x_k(N_k^l+N_k/2+n)=[x_{k-1}(N_k^l+n)+jx_{k-1}(N_k^l+N_k/2+n)]W_{N_{k-1}}^{3n},$$
$$n=0,1,\cdots,\ N_k/2-1 \tag{13}$$

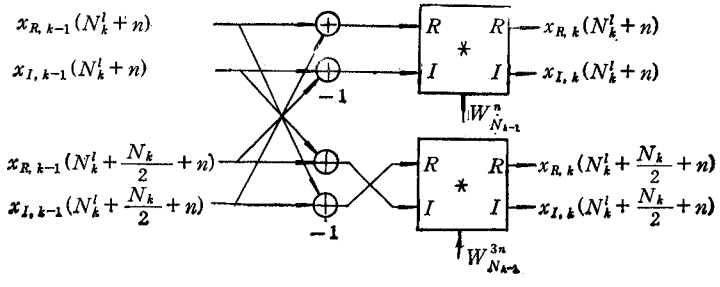

图 4　频率抽取的第二类蝶形运算单元

将实部与虚部分开后，得到第二类蝶形运算单元的实现过程如图 4 所示.

记第 k 级蝶形运算单元的段数为 $L_p(k)$，则:

$$L_p(k)=2^{k-1},\quad k=1,2,\cdots,t \tag{14}$$

故

$$N_k=\frac{N}{L_p(k)}=2^{t+1-k},\quad k=1,2,\cdots,t \tag{15}$$

$$N_k^l=N_k(l-1)=(l-1)2^{t+1-k},\quad k=1,2,\cdots,t,$$
$$l=1,2,\cdots,L_p(k) \tag{16}$$

五、两类蝶形运算单元的类别确定

当 k 与 l 给定后，如何判断所需要进行的蝶形运算单元是第一类还是第二类呢？下面我们给出确定类别的方法.

因为总共只有两类蝶形运算单元，所以我们只需确定所有第二类蝶形运算单元的位置. 记第 k 级中第二类蝶形运算单元的段数为 $LK(k)$，而段号分别为 $NK(k,l)$，$l=1,2,\cdots,LK(k)$，显然有:

$$NK(k,l)\leqslant L_p(k) \tag{17}$$

在第一级中，有

$$L_p(1)=1 \tag{18}$$

这唯一的一段蝶形运算单元是第一类的，故

$$LK(1)=0 \tag{19}$$

在第二级中，它的上半部分是一个长度减半的 DFT，因此这一半的蝶形运算单元分类情况与第一级相同(只不过每段点数减半，而这不影响分类)，它的下半部分是一段第二类蝶形运算单元，故

$$LK(2)=LK(1)+1=1 \tag{20}$$

$$NK(2,l)=NK(1,l)\quad l=1,2,\cdots,LK(1) \tag{21}$$

$$NK(2,LK(1)+1)=L_p(1)+1=2 \tag{22}$$

虽然(21)式是没有意义的(由(19)式可知)，但它很容易推广到后面各级的情况.

在第三级和以后各级中，上面的一半就是前一级的情况，而下面的一半则是两个长度为 $N/4$ 的 DFT，它们的分类情况分别是再前面一级的分类情况的重复. 故对 $k\geqslant 3$ 有

$$L_p(k)=2L_p(k-1) \tag{23}$$

$$LK(k)=LK(k-1)+2LK(k-2) \tag{24}$$

$$NK(k,l)=NK(k-1,l),\quad l=1,2,\cdots,LK(k-1) \tag{25}$$

$$NK(k,LK(k-1)+l)=L_p(k-1)+NK(k-2,l),$$
$$l=1,2,\cdots,LK(k-2) \tag{26}$$

$$NK(k,LK(k-1)+LK(k-2)+l)=L_p(k-1)+L_p(k-2)$$
$$+NK(k-2,l),\quad l=1,2,\cdots,LK(k-2) \tag{27}$$

由(25)式可见，在不同的 k 值时，除了 $NK(k,l)$ 数组的长度不同外，在有值的长度内，$NK(k,l)$ 的数值是完全相同的. 而 $NK(k,l)$ 的长度又已由(24)式确定. 所以对于第二类蝶形运算单元的段号数组 $NK(k,l)$，我们可以不区分其级号 k，统一用 $NK(l)$ 来表示. 这样，(25)、(26)和(27)式变为

$$NK(LK(k-1)+l)=L_p(k-1)+NK(l),\quad l=1,2,\cdots,LK(k-2) \tag{28}$$

$$NK(LK(k-1)+LK(k-2)+l)=L_p(k-1)+L_p(k-2)+NK(l),\quad l=1,2,\cdots,LK(k-2) \tag{29}$$

于是，全部第二类蝶形运算单元的段号可以由图 5 的流程确定.

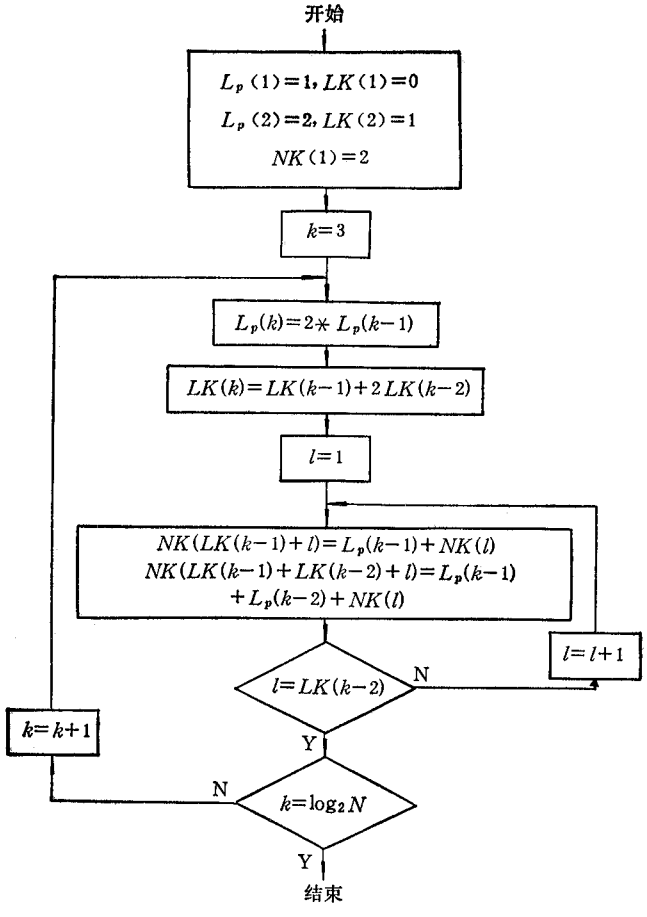

图 5　确定第二类蝶形运算单元段号的流程

确定第二类蝶形运算单元段号的过程可以离线进行，此时 N 可按最大的可能取值. 表 1 给出了当 N 为 $2^{13}=8192$ 时第二类蝶形运算单元段数 $LK(k)$ 的取值. 表 2 给出了 $NK(l)$ 的取值(由于篇幅有限，我们只给出了当 N 为 $2^7=128$ 时的数据).

从表 1 我们可以看出 $LK(k)$ 满足如下关系：

$$LK(k)=2LK(k-1)-1,\ k\text{为奇数} \tag{30}$$

表1 第二类蝶形运算单元的段数 $LK(k)$

k	1	2	3	4	5	6	7	8	9	10	11	12	13
$LK(k)$	0	1	1	3	5	11	21	43	85	171	341	683	1365

表2 第二类蝶形运算单元的段号 $NK(l)$

l	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$LK(l)$	2	6	8	10	14	18	22	24	26	30	32	34	38	40	42	46	50	54	56	58	62

$$LK(k) = 2LK(k-1) + 1,\ k\text{为偶数} \tag{31}$$

这可以用数学归纳法给以严格证明(见附录).

有了这一结论,图5中 $LK(k)$ 的计算可以简化,这里不再详述. 根据上述结论,我们还可以推测 $LK(k)$ 基本上是 2^k 的形式. 事实上,我们还可以用数学归纳法进一步证明(见附录):

$$LK(k) = \frac{1}{3}(2^{k-1} - 1),\ k\text{为奇数} \tag{32}$$

$$LK(k) = \frac{1}{3}(2^{k-1} + 1),\ k\text{为偶数} \tag{33}$$

六、时间抽取 TMFFT 的软件实现

时间抽取 TMFFT 与频率抽取 TMFFT 的分解过程非常相似,差别在于它们的运算分级次序正好相反和它们的第二类蝶形运算单元的结构方向相反.

第一类蝶形运算单元的一般数学表示为:

$$x_k(N_k^l + n) = x_{k-1}(N_k^l + n) + x_{k-1}(N_k^l + N_k/2 + n),$$
$$n = 0,1,\cdots,\ N_k/2 - 1 \tag{34}$$

$$x_k(N_k^l + N_k/2 + n) = x_{k-1}(N_k^l + n) - x_{k-1}(N_k^l + N_k/2 + n),$$
$$n = 0,1,\cdots,\ N_k/2 - 1 \tag{35}$$

它与频率抽取情况下的第一类蝶形运算单元在形式上是相同的,但是两者的 N_k 与 N_k^l 是不同的. 把数据分为实部和虚部后,得到它的运算过程如图6所示.

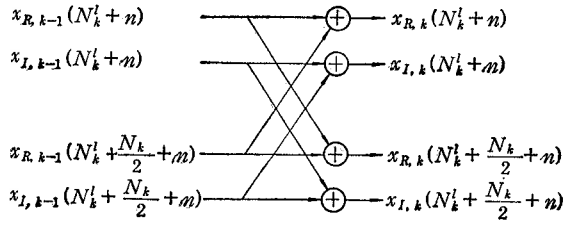

图6 时间抽取时的第一类蝶形运算单元

第二类蝶形运算单元的一般数学表示为:

$$x_k(N_k^l + n) = W_{N_{k+1}}^{n} x_{k-1}(N_k^l + n) + W_{N_{k+1}}^{3n} x_{k-1}(N_k^l + N_k/2 + n),$$

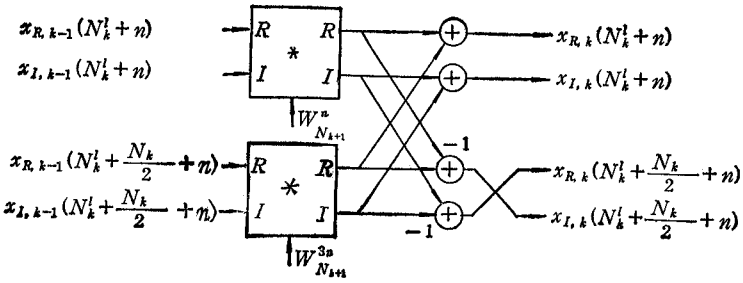

图 7　时间抽取时的第二类蝶形运算单元

$$n = 0,1,\cdots,\ N_k/2 - 1 \tag{36}$$

$$x_k(N_k^l + N_k/2 + n) = -j[W^n_{N_{k+1}} x_{k-1}(N_k^l + n) - W^{3n}_{N_{k+1}} x_{k-1}(N_k^l + N_k/2 + n)],$$

$$n = 0,1,\cdots,N_k/2 - 1 \tag{37}$$

把数据分为实部和虚部后，得到它的运算过程如图 7 所示。

在时间抽取情况下，

$$L(k) = N/2^k = 2^{t-k},$$
$$k = 1,2,\cdots,t \tag{38}$$

$$N_k = N/L_p(k) = 2^k,$$
$$k = 1,2,\cdots,t \tag{39}$$

$$N_k^l = N_k(l-1) = (l-1)2^k,$$
$$k = 1,2,\cdots,t,$$
$$l = 1,2,\cdots L_p(k) \tag{40}$$

由于时间抽取的分级次序与频率抽取的分级次序正好相反，故第 k 级中第二类蝶形运算单元的段数为 $LK(t+1-k)$，LK 数组与频率抽取时相同，而第二类蝶形运算单元的段号 $NK(l)$ 与 k 无关，故它与频率抽取时完全相同。

七、TMFFT 算法的一般流程

不论是频率抽取还是时间抽取的 TMFFT 算法，如果不考虑序列的倒序问题，都可以用图 8 的流程来实现。

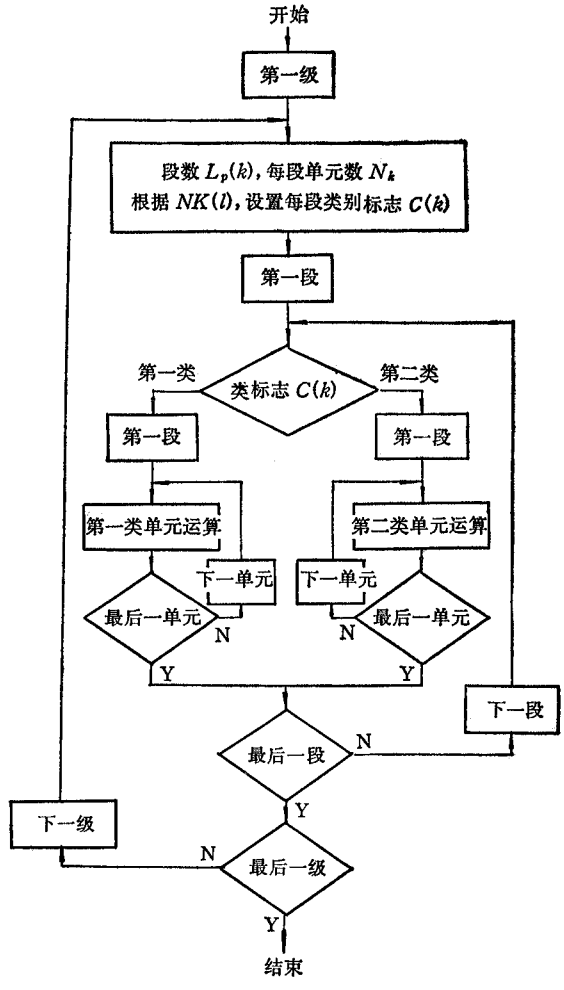

图 8　不包括序列倒序的 TMFFT 流程

八、结 论

不论是频率抽取还是时间抽取的FFT，算法都可以通过两级旋转因子合并得到更高效的 TMFFT 算法，其外部特性完全不变，可以在现有的信号处理系统中直接取代 FFT. 而且对进行输入或输出修枝的情况也适用. 对 FFT 算法的实现方法作一些修改，就得到 TMFFT 的实现方法.

附录 第二类蝶形运算单元段数公式的证明

I. (30),(31)式的证明

1. 当 $k=2$ 时，k 为偶数

$$2LK(k-1)+1=2LK(1)+1=1=LK(2)=LK(k) \tag{A-1}$$

结论成立.

2. 假设 $k=k_1$ 时，结论成立

(1) 若 k_1 为奇数，则 k_1+1 为偶数，根据(30)式得：

$$2LK(k_1-1)=LK(k_1)+1 \tag{A-2}$$

又根据(24)式得：

$$LK(k_1+1)=LK(k_1)+2LK(k_1-1) \tag{A-3}$$

将 (A-3) 式代入 (A-2) 式即得：

$$LK(k_1+1)=2LK(k_1)+1 \tag{A-4}$$

即 $k=k_1+1$ 时结论也成立.

(2) 若 k_1 为偶数，则 k_1+1 为奇数. 同理，根据(24),(31)式可得：

$$LK(k_1+1)=2LK(k_1)-1 \tag{A-5}$$

即 $k=k+1$ 时结论也成立.

3. 根据数学归纳原理，对任意 $k\geqslant 2$，结论(30),(31)式成立.

II. (32),(33)式的证明

1. 当 $k=1$ 时，k 为奇数

$$\frac{1}{3}(2^{k-1}-1)=0=LK(1) \tag{A-6}$$

结论成立.

2. 假设 $k=k_1$ 时，结论成立

(1) 若 k_1 为奇数，则 k_1+1 为偶数. 根据(31),(32)式可得：

$$LK(k_1+1)=\frac{1}{3}(2^{k_1}+1) \tag{A-7}$$

即 $k=k_1+1$ 时，结论也成立.

(2) 若 k_1 为偶数，则 k_1+1 为奇数. 根据(30),(33)式可得：

$$LK(k_1+1)=\frac{1}{3}(2^{k_1}-1) \tag{A-8}$$

即 $k=k_1+1$ 时，结论也成立

3. 根据数学归纳原理，对任意 k 为正整数，结论(32)，(33)式成立.

参　考　文　献

[1] J. B. Martens, *IEEE Trans.* on ASSP, **ASSP-32**(1984), 750.
[2] 许蔚，陈宗骞，电子科学学刊，9(1987)，229—234.
[3] A. V. Oppenheim, R. W. Schafer, Digital Signalnal Processing, Prentice-Hall Inc., 1975; 董世嘉，杨耀增译，数字信号处理，科学出版社，1980.
[4] 许蔚，合成孔径雷达信号数字处理，中国科学技术大学六系硕士论文，1984 年 6 月.
[5] T. V. Sreenivas, P. V. S. Rao, *IEEE Trans.* on ASSP, **ASSP-28**(1980), 254.

TWIDDLE FACTOR MERGED TIME-DECIMAL FFT ALGORITHM AND THE SOFTWARE IMPLEMENTATION FOR TMFFT

Xu Wei　Chen Zongzhi

(Institute of Electronics, Academia Sinica, Beijing)

ABSTRACT Martens (1984) propesed a high efficient and simple formed DFT algorithm-RCFA,, whose efficiency can be compared with that of WFTA or that of PFA, and whose structure is similar to that of FFT. The authors have proved that, in the case of radix 2, the RCFA is exactly equivalent to the twiddle factor merged frequency-decial FFT algorithm. The twiddle factor merged time-decimal FFT algoorithm is provided in this paper. Thus, in any case, the FFT algorithm used currently can be replaced by the more efficient algorithm—— the twiddle factor merged FFT algorithm, with exactly the same external property and the similar internal structure. In addition, the software for implementing the twiddle factor merged FFT algorithm (TMFFT) is also provided.

KEY WORDS FFT algorithm, TMFFT; Software implementation