1993年9月 重庆大学学报 Vol. 16, №. 5
第16卷第5期 JOURNAL OF CHONGQING UNIVERSITY Sep. 1993

一维离散 Fourier 变换的阵列协处理器设计*

The Design of Array Co-Processors for One-Dimensional Fourier Transform

周六丁 程代杰 邹华
Zhou Liuding Cheng Daijie Zhou Hua
(重庆大学电子信息学院,重庆,630044)

摘 要 提出了一种用短DFT芯片构成长序列DFT的阵列协处理器的有效方法。此法易用2^t片、2^t长DFT芯片及2^{2t}个乘法器构成2^{2t}长的DFT阵列协处理器,也易用2^{t+1}片、2^t长DFT芯片,2^{2t+1}个乘法器及2^{2t}个蝶形运算单元构成2^{2t+1}长的DFT阵列协处理器。文中给出了它们的并行结构及分析结果。它们具有并行计算度高,芯片需求少,简单易实现的特点。

关键词 Fourier变换;并行计算;体系结构
中国图书资料分类法分类号 TP302.8

ABSTRACT In this paper, an efficient method for constructing array co-processors for one-dimensional DFT was proposed. With it, we can construct an array co-processor for 2^{2t} points DFT with 2^t chips (2^t points per chip) and 2^{2t} multiplication units; and an array co-processor for 2^{2t+1} points DFT with 2^{2t+1} chips (2^t points per chip), 2^{2t+1} multiplication units and 2^{2t} butterfly operation units.

KEYWORDS fourier transform; parallel computing; systematical structure

0 引 言

离散Fourier变换(即DFT)在众多领域中起着极重要作用,其高速计算是一重要研究课题。人们已提出许多快速算法[1]、并行算法[2,3],DFT专用硬件[4],阵列机[5]和宏流水向量机[5]。

因多维DFT可归为一维DFT计算,故DFT的快速计算常归为一维DFT快速计算的研究,从实用角度看,Cooley-Tukey的基2FFT因结构简单,适用面宽等优点而仍广泛采用。在硬件及专用机方面,协处理器(如TMS32020[1])是性能价格比较优的选择。因此本文基于Cooley-Tukey FFT研究一维DFT阵列型协处理器的设计。由于Cooley-Tukey FFT不具备任务级并行性(即不易用VLSI芯片实现),故文中先给出一种新的计算1-DFT的"长化短"算法(及修正本)。它(们)与Cooley-Tukey FFT(简称普通FFT)同样快,但具有下述两个特点。

* 收文日期 1992-11-22
受国家自然科学基金和重庆博士基金资助

1) 规整性　即它(们)能将 2^{2t} 长(或 2^{2t+1} 长)DFT 化为 2^{2t+1}(或 2^{t+2}) 个 2^t 长的计算。

2) 对称性　它(们)的计算流程的两阶段相同。　利用 1) 则可用 2^t 长芯片构成 2^{2t} 长芯片构成 2^{2t}(或 2^{2t+1}) 长 DFT 阵列协处理器。利用 2) 可使芯片需求量减少一半。

文中随后给出了 2^{2t} 长及 2^{2t+1} 长的 DFT 阵列协处理器的并行结构及时耗分析结果。它们具有并行计算度高，芯片需求量少，数据传输率高，易于实现四方面优点。

1　一维 DFT 的两种快速算法

不失一般性，仅考虑下面一维 DFT 的计算。

$$F(k)=\sum_{m=0}^{M-1} f(m)\cdot W^{k\cdot m}\cdot m \qquad (1)$$

$$m=0$$

$$(k=0,1,\cdots,M-1;M=2^n;n\in Z^+)$$

其中 $W_N=e^{-j2\pi/N}$

1.1　一维 DFT 的普通 FFT 算法(Cooley-Tukey FFT)

因读者已对 Cooley-Tukey FFT 很熟悉[1]，这里仅引用有关结论。

结论1　Cooley-Tukey FFT 完成 $M(=2^n)$点复序列 DFT 所需复数乘，加次数 $M_c(M)$，$A_c(M)$分别约为：

$$M_c(M)=\frac{M}{2}\cdot\log_2 M;\qquad A_c(M)=M\cdot\log_2 M$$

1.2　一维 DFT 的"长化短"FFT 算法

因 $M=2^n(n\in Z^+)$，则 $M=M_0\cdot M_1$，$M_0=2^{[n/2]}$，$M_1=2^{n-[n/2]}$。定义两个 $M_0\times M_1$ 的二维数组 $\{y(m_0,m_1)\}$，$\{Y(k_0,k_1)\}$。若将序列$\{f(m)\}$ 按列序依次放入$\{y(m_0,m_1)\}$，则建立起 $f(m)$ 与 $y(m_0,m_1)$ 间一一对应$(0\leqslant m\leqslant M-1;0\leqslant m_0\leqslant M_0-1;0\leqslant m_1\leqslant M_0-1)$。如将 $\{Y(k_0,k_1)\}$ 按行序依次放入$\{F(k)\}$ 中，则建立起$\{Y(k_0,k_1)\}$ 与 $F(k)$ 间一一对应$(0\leqslant k\leqslant M-1;0\leqslant k_0\leqslant M_0-1;0\leqslant k_1\leqslant M_0-1)$。两种情形如图 1,2 所示。

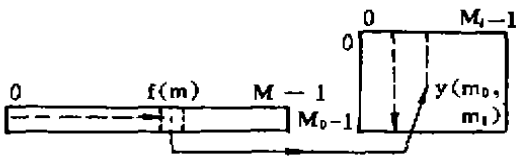


图 1　$f(m)$ 与 $y(m_0,m_1)$ 间对应关系，其中有

$$\begin{cases} f(m)\Leftrightarrow y(m_0,m_1)\\ m=m_1\cdot M_0+m_0\\ (0\leqslant m\leqslant M-1;0\leqslant m_0\leqslant M_0-1;\\ 0\leqslant m_1\leqslant M_1-1) \end{cases}$$

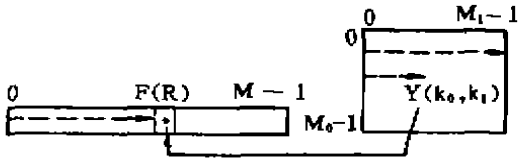


图 2　$F(k)$ 与 $Y(k_0,k_1)$ 间的对应关系，其中有

$$\begin{cases} f(k)\Leftrightarrow y(k_0,k_1)\\ k=k_0\cdot M_1+k_1\\ (0\leqslant k\leqslant M-1;0\leqslant k_0\leqslant M_0-1;\\ 0\leqslant k_1\leqslant M_1-1) \end{cases}$$

定理 1　令　$M=M_0\cdot M_1(M=2^n;M_0=2^{[n/2]};M_1=2^{n-[n/2]})$，则式(1) 等价于下三步计算。

(i) 将序列$\{f(m)\}$ 按列序依次放入二维数组 $y(m_0,m_1)$ 中(方法见图 1)。

(ii) 计算：

$$Y(k_0,k_1)=\sum_{m_0=0}^{M_0-1}\sum_{m_1=0}^{M_1-1}y(m_0,m_1)\cdot W_{M_1}^{m_1\cdot k_1}\cdot W_{M_0}^{m_0\cdot k_0}\cdot W_M^{m_0\cdot k_1} \tag{2}$$

$$(0\leqslant k_0\leqslant M_0-1;0\leqslant k_1\leqslant M_1-1)$$

(*iii*) 将二维数组$\{Y(k_0,k_1)\}$按行序依次放入$\{F(k)\}$中(方法见图 2).

证明：见文献[3].

定理1提供了另一种计算1－DFT 的方法。注意到式(2)的计算等价于下三步的计算。

$$Z(m_0,k_1)=\sum_{m_1=0}^{M_1-1}y(m_0,m_1)\cdot W_{M_1}^{m_1\cdot k_1} \tag{3}$$

$$(0\leqslant m_0\leqslant M_0-1;0\leqslant k_1\leqslant M_1-1)$$

$$Z'(m_0,k_1)=Z(m_0,k_1)\cdot W_M^{m_0\cdot k_1} \tag{4}$$

$$(0\leqslant m_0\leqslant M_0-1;0\leqslant k_1\leqslant M_1-1)$$

$$Y(k_0\cdot k_1)=\sum_{m_0=0}^{M_0-1}Z'(m_0,k_1)\cdot W_{M_0}^{m_0\cdot k_0} \tag{5}$$

$$(0\leqslant k_0\leqslant M_0-1;0\leqslant k_1\leqslant M_1-1)$$

再注意到数组 Y,Z,Z' ,Y 可公用 y,f 及 F 可公用 f,则有下面算法。

Algorithml. 计算一维 DFT(式(1))的"长化短"FFT 算法。

Procedure FFT1 (f,M)(输入时 f 为 $M=2^n$ 点序列，输出时 f 放结果).

1) 由 M 计算 M_0、M_1 并定义 $y(0:M_0-1;0:M_1-1)$;

2) 将 f 数组按列序依次放入 y 中(见图 1);

3) 对 y 的每一行，用普通 FFT 计算 M_1 点 DFT，其结果放回原行;

4) 对 y 中每一元 $y(m_0,k_1)$ 乘上因子 $W_M^{m_0\cdot k_1}$;

5) 对 y 的每一列，用普通 FFT 计算 M_0 点 DFT，其结果放回原列;

6) 将数组按行序依次放入 f 数组中(见图 2).

定理 2　Algorthml 能正确计算 1 － DFT，且完成 M 点 DFT 需 $\frac{M}{2}\cdot\log_2M+M$ 次复乘及 $M\cdot\log_2M$ 次复加。

证：见文献[3]

2　一维 DFT 阵列协处理器的设计与分析

由于协处理器具有较高性能价格比且现有 DFT 芯片变换长度一般较短，因而研究 DFT 的 VISI 的阵列型协处理器具有理论和实用价值。

2.1　$M=2^{2t}$ 长的阵列协处理器的设计及分析

当变换长度 $M=2^{2t}$ 时，令 $M'=2^t$，Algorithml 等价于下面算法。

Alogrithm 2. $M=2^{2t}$ 时计算 1 － DFT 的"长化短"FFT 算法

Procedure FFT2(f,M),(输入时 f 为 $M=2^{2t}$ 点序列，输出时为结果)

1) 由 M 计算 M' ($M=2^{2t}$;$M'=2^t$)，定义 $y(0:M'-1;0:M'-1)$;

2) 将 f 数组按行序依次放入 y 中，将 y 转置;

3) 对 y 的每一行，用普通 FFT 算法计算 M' 点 DFT，其结果放回原行;

4) 对 y 的每一元 $y(m_0,k_1)$ 乘因子 $W_M^{m_0\cdot k_1}$，然后将其转置；

5) 对 y 的每一行，用普通 FFT 算法计算 M' 点 DFT，其结果放回原行；

6) 将 y 转置，将 y 按行放入 f 中。

Alogrithm 2 的计算流程(以 $2\cdot 2$ 点为例) 如图 3 所示，它具有“规整性”和“对称性”两特点。

利用 lgorithm 2 的计算流程，则可设计出计算 2^{2t} 点 DFT 的 VISI 阵列协处理器。图 4 给出了 $2^{2\times 2}=16$ 点 DFT 阵列协处理器结构。其中每位移 / 设置寄存器的逻辑图及功能表如图 5 所示。

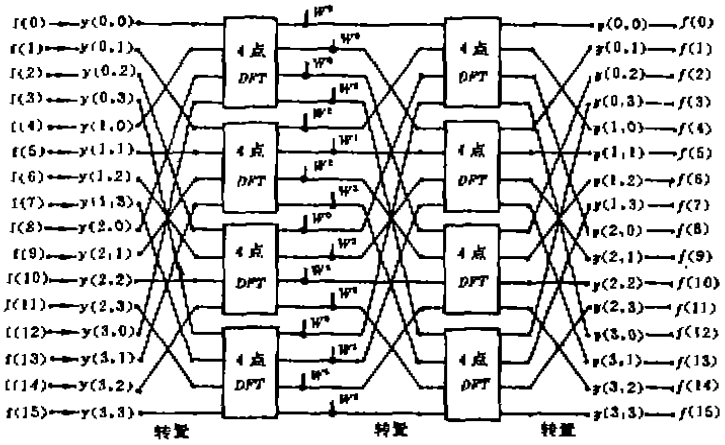


图 3 Algorithm 2 计算 $2^{2t}(t=2)$ 点 DFT 流程

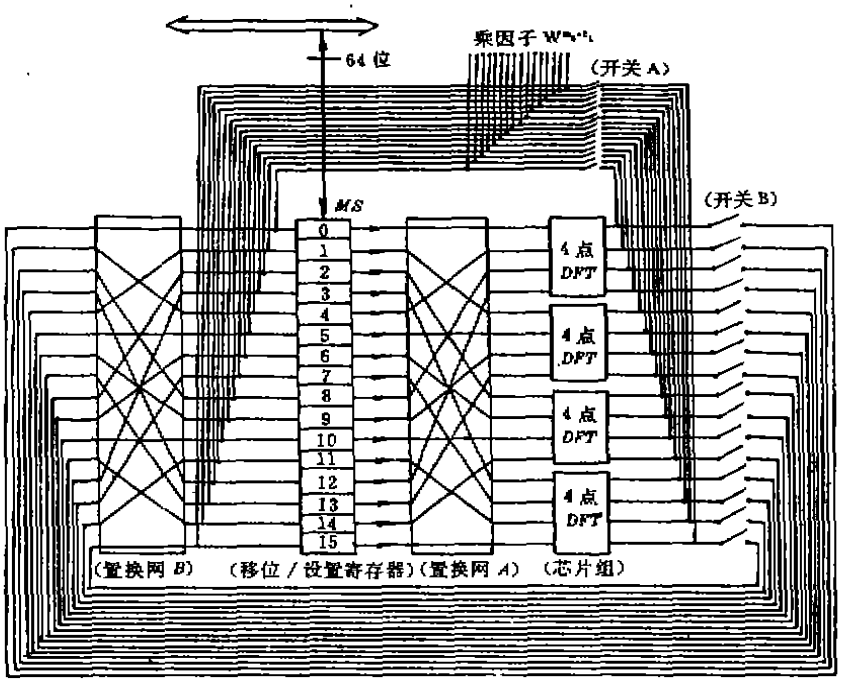
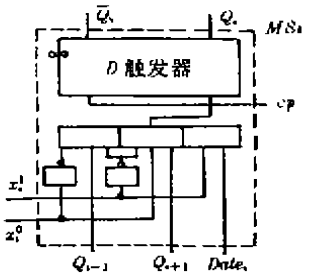


图 4 $2^t\cdot 2^t(t=2)$ 点 DFT 阵列协处理器的结构



MSi 功能表

x_i^1	x_i^0	
0	0	右移
0	1	左移
1	X	设置

图 5　移位 / 设置寄存器(一位)的逻辑图与功能表

2^{2r} 长 DFT 阵列协处理器由 $2^{2t} \times 64$ 个移位 / 设置寄存器(2^{2t} 点，每点 64 位)，两个 2^{2t} 点输入 2^{2t} 点输出的置换网，2^t 片 2^t 点 DFT 芯片，2^{2t} 个复数乘法器，两组三态开关 A 及 B(每组 $2^{2t} \times 64$) 以及一条 64 位宽的数据传输线构成。

其工作过程如下。中央处理机首先以 DMA 方式将 2^{2t} 点序列从主存传送到移位 / 设置寄存器(即 MS) 中。数据通过置换网 A 到达芯片组的输入端，这时 2^t 片 2^t 长 DFT 芯片并行完成第一次计算，其结果通过开关 A，并行乘法网后又置入 MS 中。其后数据又通过置换网 A 到达芯片组输入端，这时 2^t 片 2^t 点 DFT 芯片并行完成第二次计算，其结果通过开关 B，置换网 B 置入 MS 中。最后中央处理机以 DMA 方式将 MS 中的变换结果传回主存。

2^{2t} 点 DFT 阵列协处理器完成 $M = 2^{2r}$ 点 DFT 所需时间 $T_{co\text{-}p}(2^{2t})$ 分为两部分。一是主存与移位 / 设置寄存器组 MS 间数据传输的时间(一次从主存到 MS，一次从 MS 到主存)。该时耗为 $2 \cdot T_{DMA}(2^{2t}) = 2 \cdot C_{DMA} \cdot 2^{2t}$，其中 $T_{DMA}(x)$ 表示用 DMA 方式在主存与 MS 间传输 x 点复数数据的时耗，而 C_{DMA} 为这种情形下传输一点数据的平均时耗。由于移位 / 设置寄存器的翻转周期可与主存的存取周期 t_m 相匹配，则有 $C_{DMA} = 2 \cdot t_m$，因而 $2 \cdot T_{DMA}(2^{2t}) = 4 \cdot t_m \cdot 2^{2t}$.

$T_{co\text{-}p}(2^{2t})$ 的第二部分是 2^t 片 2^t 长 DFT 芯片完成 2^t 点 DFT 所需时间的两倍，即约等于 $2T_{DFTC}(2^t)$，因此有：

$$T_{co\text{-}p}(2^{2t}) = 4 \cdot t_m \cdot 2^{2t} + 2 \cdot T_{DFTC}(2^t)$$

由结论 1 知道普通 FFT 完成 $M = 2^{2t}$ 点 DFT 约需 $(M/2) \cdot \log_2 M$ 次复乘，$M \cdot \log_2 M$ 次复加。即需 $4 \cdot 2^{2t} \cdot t$ 次实数乘法，$6 \cdot 2^{2t} \cdot t$ 次实数加法。通常用指令实现时，一次实数乘法约需 30 个主存周期，一次实数加法约需 10 个主存周期，因此普通 FFT 完成 2^{2t} 点 DFT 所需时间约为：

$$T_{FFT}(2^{2t}) > 30 \cdot t_m 4 \cdot 2^{2t} \cdot t + 10t_m \cdot 6 \cdot 2^{2t} \cdot t = 180t_m \cdot 2^{2t} \cdot t$$

则用阵列协处理器计算 $M = 2^{2t}$ 点 DFT 相比用普通 FFT 算法计算的加速比为：

$$S_p(2^{2t}) = \frac{T_{FFT}(2^{2t})}{T_{co\text{-}p}(2^{2t})} \geqslant \frac{180t_m \cdot 2^{2r} \cdot t}{4 \cdot t_m \cdot 2^{2r} + 2 \cdot T_{DFTC}(2^t)}$$

若假定 $2 \cdot T_{DFTC}(2^t) < 4 \cdot t_m \cdot 2^{2t}$(即计算时间小于数传时间)，则：

$$S_p(2^{2t}) > 22.5 \times t; \text{当 } t = 6 \text{ 时 } S_p(4096) > 135(\text{倍})$$

这说明，在合理假定下，用阵列协处理器完成 4096 点 DFT 相比用串行算法快 135 倍左右。

2) $M = 2^{2t+1}$ 长的阵列协处理器的设计及分析

容易推知，一个 $M = 2^{2t+1}$ 点 DFT

$$F(k) = \sum_{m=0}^{M-1} f(m) \cdot W_M^{k \cdot m} \tag{6}$$

$$(k = 0, 1, \cdots\cdots, M-1; M = 2^{2t+1}; W_M = e^{-j2\pi/M})$$

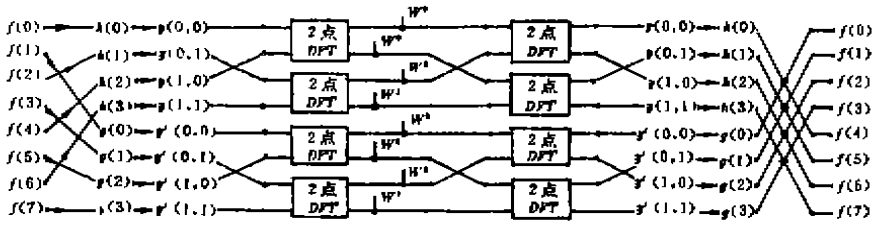


图 6 *Algorithm*3计算$2^{2t+1}=8$点 *DFT* 的流程

可由下两式计算，即：

$$F(k) = F_{even}(k) + F_{odd}(k) \cdot W_M^k \tag{7}$$

$$F(k + M/2) = F_{even}(k) - F_{odd}(k) \cdot W_M^k \tag{8}$$

$$(k = 0, 1, \cdots\cdots, M/2 - 1; M/2 = 2^{2t})$$

其中

$$F_{even}(k) = \sum_{m=0}^{M/2-1} f(2m) \cdot W_{M/2}^{k \cdot m} \tag{9}$$

$$F_{odd}(k) = \sum_{m=0}^{M/2-1} f(2m+1) \cdot W_{M/2}^{k \cdot m} \tag{10}$$

$$(k = 0, 1, \cdots\cdots, M/2 - 1; M/2 = 2^{2t})$$

注意式(9)、(10)是 2^{2t} 点 DFT，可用 Algorithm 2 计算，则有算法：

Algorithm 3 $M = 2^{2t+1}$ 的计算 M 点 DFT 的“长化短”算法

Procedure FFT3(f, M)(输入 f 为 $M = 2^{2t+1}$ 点复序列，输出为结果)

1) 定义两个 $M/2$ 点的一维数组 h 和 g；

2) 令 h 中放置 f 中具有偶下标的抽取点；g 中放置 f 中具有奇下标的抽取点；

3) CALL FFT2(h, $M/2$)；CALL FFT2(g, $M/2$)；(用 Algorithm 2 计算 $M/2$ 点 DFT)；

4) *for* $k \leftarrow 0$ *to* $M/2 - 1$ *do*

$\{t \leftarrow g(k) \cdot W_M^k; f(k) \leftarrow h(k) + t; f(k + M/2) \leftarrow h(k) - t;\}$

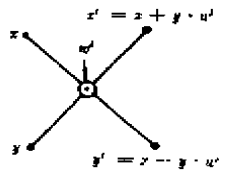


图 7 蝶形运算功能图

根据该算法及 Algorithm 2 则可得到计算 $M = 2^{2t+1}$ 点 DFT 的计算流程。图 6 给出 $2^{2\times1+1} = 8$ 点 DFT 的流程，其中每个蝶形运算是图 7 中省略 W_M^k 的形式。

利用 Algorithm3 及其流程，可设计出 2^{2t+1} 点 DFT 的阵列协处理器。例如，图 8 给出了 8 点 DFT 的阵列协处理器结构。

类似于 2^{2t} 阵列协处理器的讨论，我们可以给出 2^{2t+1} 点 DFT 阵列协处理器的组成，工作过程及性能分析。有关问题可参见文献[8].

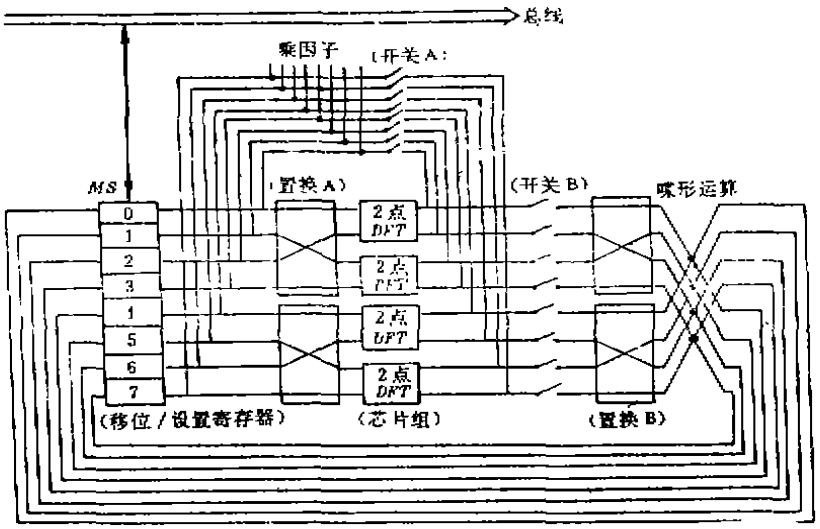


图 8　$2^{2t+1}(t=1)$ 点 DFT 阵列协处理器

3　结束语

1）因变换长度 M 几乎总为 $2^n(n \in Z^+)$ 形式，则 M 总可表示为 $M=2^{2t}$ 或 2^{2t+1} 的形式。因而总可用短 DFT 芯片构成所需变换长度的阵列协处理器。但从实用角度看，$M=2^{2t}$ 型阵列协处理器比较易于实现。

2）本文核心是利用 DFT 可"规整分解"且分解后的计算具有"对称性"两特点，构成能减少一半芯片量的阵列协处理器，并采用移位/设置寄存器组作为数据交换的接口，较好解决数传瓶颈。

3）应注意文中方法具有递归性质，这种递归性质用于设计 DFT 专用芯片时可望使布线规整并能较大幅度减少硬件复杂度。

4）文中设计 DFT 阵列协处理器的方法可用于 Walsh 变换阵列协处理器设计[7]。

参 考 文 献

1　Nussbaumer H J. Fast Fourier Transform and Convolution Algorithms, Springer-Verlag Berlin Heidelberg New York, 1981

2　A Norton, etal. Parallelization and Performance Analysis of the Cooly-Tukey FFT Algorithm for shared-Memory Architectures, IEEE Trans. on Computer, 1987, 581～591

3　周六丁等. 一种计算一维 Fourier 变换的 MIMD 并行算法. 计算机学报, 1991(7): 548～552

4　汪亚南等. TMS320系列高速单片计算机原理与应用. 成都：电子科学大学出版社, 1991

5　K. Huang, etal. Computer Architecture and Parallel Processing, McGraw-Hill Book Company, 1984, 325～327

6　张德富等. 构造宏流水线并行算法的一种有效方法. 计算机学报, 1989(7): 617～625

7　周六丁等. 一维 Walsh 变换的阵列协处理器的设计. 计算机学报, 1993(1): 59～64

8　周六丁. 正交变换，卷积及组合问题快速计算. 重庆大学博士论文, 1991, 5