計算機アーキテクチャ　109－11
（１９９４．１２．１３）

グローバル通信を用いた並列FFTアルゴリズムと超並列計算機への実装

武田利浩*，丹野州宣*，堀口進**

*山形大学工学部電子情報工学科
**北陸先端科学技術大学院大学

本文ではグローバルネットワークを用いた基数４のバタフライ演算を用いた並列FFTアルゴリズムとその商用超並列計算機への実装について述べる．Stage-by-stage法とMulti-stage法をグローバルネットワークを用いて実現し，その通信コストを比較する．グローバルネットワークを用いた場合に，提供される通信がatomic typeのみであるか，structやarray全体を一括して送ることが出来るのかが重要なことを示す．また，CM-5およびnCUBEへの実装結果についても述べる．

A Parallel FFT Algorithms with Global Communication Networks and Implementation to Massively Parallel Computers

Toshihiro TAKETA*, Kuninobu TANNO*, Susumu HORIGUCHI**

*Department of Electrical and Information Engineering, Yamagata University
Yonezawa, Yamagata 992, Japan
**Japan Advanced Institute of Science and Technology
15 Asahidai, Tatunokuti, Noumi, Ishikawa 923-12, Japan

We describe parallel FFT algorithms on massively parallel processor arrays with global interconnection networks. These algorithms adopt two method, Stage-by-stage method and Multi-stage method. The algorithms are implemented on commercial massively parallel computera the CM-5 and the nCUBE, and their communication costs are evaluated.

1．まえがき

高速フーリエ変換（FFT）[1]はディジタル信号処理の基本技術としてスペクトル解析，ディジタルフィルタ，音声認識，画像処理などに広く利用され，その応用分野はますます広がるばかりでなく，処理されるデータ量も増大の一途をたどっており，その高速処理が強く望まれている[2]．FFTを高速に処理するための一つの方法は，FFTに内包されている並列性を巧みに利用し，超並列計算機で処理することである[3]．従来から各種の並列アルゴリズムやそれらのプロセッサ・アレイあるいはマルチプロセッサへの実装が提案されている．

著者らは，すでに8隣接格子型プロセッサアレイ上でローカルネットワークを用いた基数4の並列ＦＦＴアルゴリズムの提案とその評価を行なっている[4]．高速なローカルネットワークを利用したアルゴリズムに比べ，グローバルネットワークを利用したアルゴリズムは，ネットワークの形態によってアルゴリズムを変更する必要が無いため，汎用性を有する．商用機においても，接続の形態に依存するローカルネットワークと共にグローバルネットワークを用意していたり，グローバルネットワークのみをユーザに提供するものもある．したがって，グローバルネットワークを用いたアルゴリズムについても，十分に検討する必要がある．

本文では，グローバル通信を利用した並列FFTアルゴリズムとその商用超並列計算機への実装について述べる．Stage-by-stage法とMulti-stage法をグローバルネットワークを用いて実現し，その通信コストを比較する．また，CM-5およびnCUBEへの実装結果についても述べる．

2．基数4のＦＦＴアルゴリズム

N個の複素入力データ列をg(n)，$0 \leqq n \leqq N-1$で表す．$N=4^m$の場合，nを4進m桁で表示すればg(n)は$g(n_{m-1},...,n_1,n_0)$と表すことができる．ただし，$0 \leqq n_i \leqq 3$である．また，g(n)のフーリエ係数をG(n)とすれば，G(n)も$G(n_{m-1},...,n_1,n_0)$と表示することができる．このとき，g(n)のフーリエ係数G(n)は，$W=\exp(-2\pi j/N)$として，

$$G(n)=\frac{1}{N}\sum_{k=0}^{N-1} g(k)W^{nk}$$

である（以下，１／Ｎを省略）．$G(n_{m-1},..,n_1,n_0)$は，

$$\begin{aligned}&G(n_{m-1},...,n_1,n_0)\\&=\sum_{k_{m-1}=0}^{3}\cdots\sum_{k_1=0}^{3}\sum_{k_0=0}^{3} g(k_{m-1},...,k_1,k_0)W^{n(4^{m-1}k_{(m-1)}+\cdots+4k_1+k_0)}\\&=\sum_{k_0=0}^{3}\sum_{k_1=0}^{3}\cdots\left\{\sum_{k_{m-1}=0}^{3} g(k_{m-1},\cdots,k_1,k_0)W^{4^{m-1}nk_{m-1}}\right\}W^{4^{m-2}nk_{m-2}}\cdots W^{4nk_1}W^{nk_0} \quad (2)\end{aligned}$$

のように表される[6]．以下の説明において，データ数4^i（$1 \leqq i \leqq m$）に対する回転子を$W_i=\exp(-2\pi j/4^i)$で表わす．また，$n_{i-1}=0$，１，２，３をパラメータとして$g_i(n_0,n_1,..,n_{i-1},k_{m-i-1},...,k_0)$はＦＦＴの第i段目の出力を与える．さらに，$g_0(k_{m-1},..,k_1,k_0)$は初期データ$g(k_{m-1},...,k_1,k_0)$を表す．以下，時間間引き型のFFTについて考察するが，周波数間引き型についても同様に議論できる．

時間間引き型では式（２）において，まずk_{m-1}に対しΣ操作を施す．｛｝内を第１段目として展開すると，次式のような４点のDFTの式が得られる．

$$\begin{aligned}&g_1(n_0,k_{m-2},...,k_1,k_0)\\&\qquad=\sum_{k_{m-1}=0}^{3} g(k_{m-1},...,k_1,k_0)W_1^{n_0k_{m-1}}\end{aligned}$$

式（３）の右辺のこの形は，基数４の基本演算となっており，第１段目は直接４点のフーリエ変換として与えられる．一般に，k_{m-i}（$2 \leqq i \leqq m$）に対してΣ操作を施す第i段目は次式のようになる．

$$\begin{aligned}&g_i(n_0,n_1,...,n_{i-1},k_{m-i-1},...,k_1,k_0)\\&=\sum_{k_{m-i}=0}^{3}\left\{g_{i-1}(n_0,...,n_{i-2},k_{m-i},...,k_1,k_0)W_i^{n_ik_{m-i}}W_{i-1}^{n_{i-1}k_{m-i}}\cdots W_2^{n_2k_{m-i}}\right\}W_1^{n_1k_{m-i}}\\&=\sum_{k_{m-i}=0}^{3}\left\{g_{i-1}(n_0,...,n_{i-2},k_{m-i},...,k_1,k_0)W^{bk_{m-i}}\right\}W_1^{n_1k_{m-i}} \quad (4)\end{aligned}$$

ただし，回転子W^{hkm-i}のhの値は

$$h=(4^{m-i}n_0+4^{m-i+1}n_1+\cdots+4^{m-2}n_{i-2}) \quad (5)$$

より求まる．なお，式(3)からわかるようにi＝1のとき，hは0である．W_1はデータ数4に対する回転子であるから，式（4）は，｛｝内のデータに対する4点のDFTの式に相当する．以上のようにして，第i段目の出力と回転子W^{hkm-i}のhの値が得られる．したがって，第m段目（最終段）まで計算することにより，フーリエ係数G（n）$=g_m(n_0,n_1,\ldots,n_{m-2},n_{m-1})$が求められる．

ところで，$W_1^0=1$，$W_1^1=-j$，$W_1^2=-1$，$W_1^3=j$であることから，式（3），（4）は次のような基数4のバタフライ演算と呼ばれる加減算

$$\begin{aligned}
g_i(0)&= g_{i-1}(0)v_0 + g_{i-1}(1)v_1 + g_{i-1}(2)v_2 + g_{i-1}(3)v_3\\
g_i(1)&= g_{i-1}(0)v_0 - j\,g_{i-1}(1)v_1 - g_{i-1}(2)v_2 + j\,g_{i-1}(3)v_3\\
g_i(2)&= g_{i-1}(0)v_0 - g_{i-1}(1)v_1 + g_{i-1}(2)v_2 - g_{i-1}(3)v_3\\
g_i(3)&= g_{i-1}(0)v_0 + j\,g_{i-1}(1)v_1 - g_{i-1}(2)v_2 - j\,g_{i-1}(3)v_3
\end{aligned} \quad (6)$$

により求められる[7]．ここで，$g_i(n_{i-1})=g_i(n_0,n_1,\ldots,n_{i-1},k_{m-i-1},\ldots,k_1,k_0)$で，$n_{i-1}=0$，1，2，3である．式（4）より，$v_i=W^{h\cdot i}$，i＝0，1，2，3である．ところで，$v_0$は常に1であるので，式（6）は3回の乗算と8回の加算で求まる．

3．　8－隣接プロセッサ・アレイ

3．1．　8－隣接プロセッサ・アレイの構成

本文では図1に示す$P=P_r\times P_r$個の8－隣接プロセッサ・アレイから成るSIMD型並列計算機を仮定する．ここで，$P_r=2^q$であり，またx，y＝0，1，・・・，P_r-1をそれぞれ行，列の番号とし，各プロセッサをPE(x,y)あるいはシャフル行主体添字付けの一連番号z＝0，1，・・・，P－1を用いてPE(z)と表す．

各PEはFFTを計算するための演算機能および計算に必要なデータを格納するためのローカルメモリを備えているものとする．

3．2．　通信ネットワーク

本プロセッサにおけるPE間通信は隣接PE間通信を介して通信するローカルネットワークと任意のPE間で自由に通信可能なグローバルネットワークにより行われる．ローカルネットワークは図2に示すようにパケット交換により中間のPEで中継されながら，8方向のリンク上のPEと通信可能で各リンク間で1つのデータを転送するのに要する通信時間はt_rである．したがって，dリンク離れたPE間通信に要する時間はdt_r+t_uとなる．ただし，t_uはスタートアップ時間である．グローバルネットワークに関しては任意のPE間を接続するスイッチ（回線交換）を仮定し，PE間の距離には無関係に1つのデータを転送するのに要する通信時間はt_{gr}（回線接続待ち時間も含むものとする）とし，スタートアップ時間はt_{gu}とする．ただ

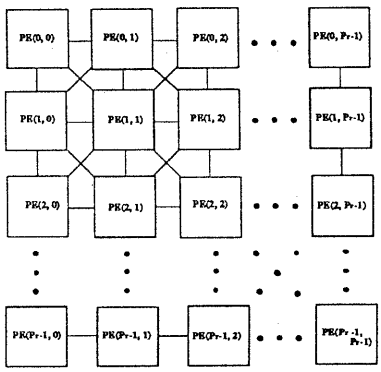


図1　8隣接プロセッサ・アレイの構成

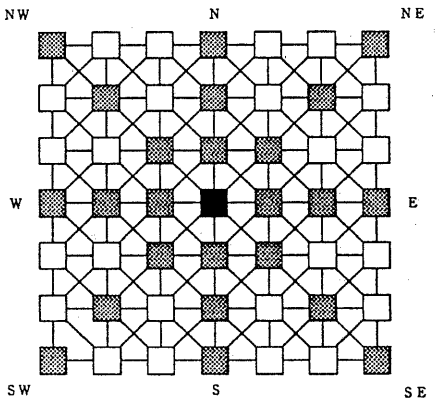


図2 ローカルネットワークによる通信

し，各リンク（回線）のt_{gr}の値はスイッチの接続状態により変化する．

各リンクの通信機能としては，最近の商用超並列計算機の仕様を参考にし[5][8]~[10]，双方向同時通信と双方向交互通信の場合について，また各PEの転送要求を持つリンクがすべて同時に並列に転送する全ポート通信と各リンクが順々に連続して転送する1ポート通信[11]も考慮し，それらの組み合わせたものを仮定する．

4．並列FFTアルゴリズム

4．1．　データ割付法

1個のPEに4の冪乗個のデータを割り付けることにより基数4の並列FFTを実現する．データの割付法としては相似割付法と重畳割付法と呼ぶ2通りの方法を考える．データ番号も4進数で表現し，データを$g(n_{m-1},n_{m-2},...,n_1,n_0)$で与える．また，PEをシャフル行主体順序で添字付し，$PE(z_{q-1},z_{q-2},...,z_1,z_0)$で表す．ただし，$m \geqq q+1$である．

2つのデータ割付法を説明するが，データは$N_r \times N_r$の2次元配列にシャフル行主体順序で並んでいるものとする．相似割付法はこの配列をq回再帰的に4分割し（図3参照），各領域にシャフル行主体順序で添字を付け，各領域の添字に対応した添字をもつPEにその領域に含まれるデータを格納する方法である．すなわち，データ$g(n_{m-1},n_{m-2},...,n_1,n_0)$を$PE(n_{m-1},n_{m-2},...,n_{m-q})$に割り付ける方法である．一方，重畳割付法は再帰的に（m−q）回4分割し，下位q桁$n_{q-1},n_{q-2},...,n_1,n_0$が等しいデータを同じ番号を持つPEに重ねて格納する方法である．すなわち，

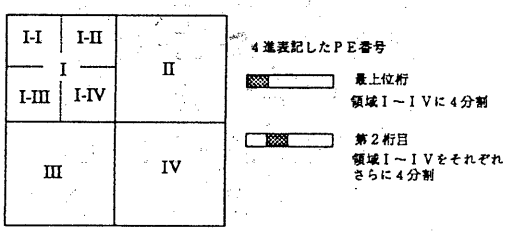


図3　シャフル行主体順序の性質

データ$g(n_{m-1},n_{m-2},...,n_1,n_0)$を$PE(n_{q-1},n_{q-2},...,n_1,n_0)$に割り付ける方法である．

4．2．　アルゴリズム

2つのデータ割付法を説明したが，アルゴリズムの並列度を考慮すると重畳割付法が相似割付法より優れているので，以下，重畳割付法について考える．

4．2．1．　Stage-by-stage法

各PEが4^{m-q}のデータを格納しているので（m−q）段バタフライ演算を行えるが，本方法は第1段～第q段までは1段毎のバタフライ演算とデータの転送を繰り返すことにより，処理を実行する．そのアルゴリズムは次のようになる．

アルゴリズム1

1）データを重畳割付けする．

2）i＝1～qまで繰り返す．

(a) 各PEはデータ$g_{i-1}(n_0,n_1,...,n_{i-2},k_{m-i},...,k_1,k_0)$に対し，式（6）にしたがって第i段目のバタフライ演算を行う．各PEの演算回数は4^{m-q-1}回である．ただし，$g_0=g$である．

(b) 結果$g_i(n_0,n_1,...,n_{i-1},k_{m-i-1},...,k_1,k_0)$を$PE(n_0,...,n_{i-1}(=0,1,2,3),k_{q-i-1},...,k_0)$に転送する．

3）i＝（q＋1）～mまで繰り返す．

各$PE(n_0,n_1,...,n_{q-1})$はデータ$g_{i-1}(n_0,n_1,...,n_{i-1},k_{m-i-1},...,k_1,k_0)$を式（6）にしたがってi段目のバタフライ演算を行う．

説明　1）は重畳法による初期データの割り付けを表す．そのためには，おのおの$n_{q-1},...,n_1,n_0$が等しい4^{m-q}個のデータ$g(n_{m-1},n_{m-2},...,n_q,n_{q-1},...,n_1,n_0)$を1つの$PE(n_{q-1},....,n_1,n_0)$に割り当てる．次いで，2）によりバタフライ演算とデータの転送を繰り返しながら1～q段までのバタフライ演算を行う．q段のバタフライ演算後のデータは$g_q(n_0,n_1,...,n_{q-1},k_{m-q-1},...,k_1,k_0)$と表されるので，以後3）にしたがい各PE内で（q＋1）～m段までのバタフライ演算が実行される．

4．2．2．　Multi-stages 法

アルゴリズム3では1～q段まではバタフライ演算とデータ転送を1段毎に繰り返すが，各PEが4^{m-q}のデータを格納しているので（m－q）段バタフライ演算を行ない，その結果$g_i(n_0,n_1,..,n_{m-q-1},k_{q-1},..,k_1,k_0)$を$PE(n_0,..,n_{n-q-1},k_{2q-m-1},..,k_0)$に転送するようなアルゴリズムも考えられる．そのアルゴリズムは次のように記述される．

アルゴリズム2

1）データを重畳割付けする．

2）i＝1～fまで繰り返すことにより，1～(m-q)f段までのバタフライ演算を行う．ただし，f=ceil($\frac{q}{m-q}$)で，ceil()はシーリング関数である．

(a) 各PEはデータ$g_{(m-q)(i-1)}(n_0,n_1,..,n_{(m-q)i-1},k_{m-(m-q)i-1},..,k_1,k_0)$に対し，式（6）にしたがって第(m-q)(i-1)+1～(m-q)i段目までのバタフライ演算を行う．各PEの演算回数は4^{m-q-1}(=m-q)回である．ただし，g_0＝gである．

(b) 結果$g_{(m-q)i}(n_0,n_1,..,n_{(m-q)(i+1)-1},k_{m-(m-q)(i+1)-1},..,k_1,k_0)$を$PE(n_0,..,n_{(m-q)(i-1)},..,n_{(m-q)i-1},k_{q-i-1},..,k_0)$に転送する．

3）i＝(m-q)f+1～mまで繰り返すことにより最終結果を得る．

各$PE(n_0,n_1,..,n_{q-1})$はデータ$g_{(m-q)(i-1)}(n_0,n_1,..,n_{(m-q)i-1},k_{m-(m-q)i-1},..,k_1,k_0)$を式（6）にしたがってi段目のバタフライ演算を行う．

説明　1）は重畳法による初期データの割り付けを表す．次いで，2）によりバタフライ演算とデータの転送を繰り返しながら1～(m-q)f段までのバタフライ演算を行う．(m-q)f段のバタフライ演算後のデータは$g_{(m-q)f}(n_0,n_1,..,n_{(m-q)(f+1)-1},k_{m-(m-q)(f+1)-1},..,k_1,k_0)$と表されるので，以後3）にしたがい各PE内で(m-q)f+1段～m段までのバタフライ演算が実行され，式（1）の結果が得られる．

5．並列計算機への実装

アルゴリズム1と2を並列計算機へ実装する場合はローカルネットワークを使用するかグローバルネットワークを使用するかでその実装方法が異なる．ここでは，グローバルネットワークを用いた場合を考える．

5．1．　アルゴリズム1の実装

アルゴリズム1を実装する場合，細部において幾つかのバリエーションが考えられる．アルゴリズム1の骨格部分は，図4のように書ける．アルゴリズム1は，ＰＥに割り付けられた4の冪乗個のデータを使い1段バタフライ演算を済ませてから，データの転送を行う方法である．ここで，対象とする並列計算機の提供する通信方式ががatomic typeのみなのか，structやar-

```
// Algorithm 1 Stage-by-stage法

// 重畳割付
SuperpositionMapping();

// FFT と 再配置
for (i=1; i<=q; i++) {
    FFT( i, 1);
    ReMapping( i, 1);
}
// FFT (m-q)段
for ( k=1; i<=m; i++, k++) {
    FFT( i, k);
}

// ＦＦＴを実行する関数
// FFT( i, k);
// i段目のFFTを配列の添え字の
// k番目をパラメータにして行う

// データの再配置を実行する関数
// ReMapping( i, l);
// i段目のデータ再配置を
// l段目のFFTを行ったものとして行う
```

図4 アルゴリズム1：Stage-by-stage法

```
// Algorithm 2 Multi-stage

// 重畳割付
SuperpositionMapping();

// FFT と 再配置
for (j=0, f=ceil(q/(m-q)); j<f; j++) {
    for (k=1; k<=(m-q); k++) {
        i= j*f+k;
        FFT( i, k);
    }
    ReMapping( j, (m-q));
}
// FFT (m-(m-q)*f)段 : 最大で(m-q)段
for ( k=m-(m-q)*f; i<=m; i++, k++) {
    FFT( i, k);
}
```

図5 アルゴリズム２：Multi-stage法

ray全体を一括して送れる機能があるのかも重要である．それぞれを区別する際は，1aと1bと呼ぶ．一括してデータを通信出来れば，同一のＰＥにデータをまとめて転送することができる．

５．２．　アルゴリズム２の実装

アルゴリズム２を実装する場合は，アルゴリズムの骨格部分は，図５のようになる．ローカルネットワークを用いた場合は，アルゴリズム１に比べＦＦＴを多段行うために転送する際に送り先のＰＥが増え複雑になる．しかし，グローバルネットワークを用いる場合，相手ＰＥを指定するだけであるから，その複雑さはない．

５．３．通信時間の評価

Ｌ段のＦＦＴの後，転送を要するデータの数Nt(L)は，各ＰＥ内に保持するデータから自ＰＥ内に残る分を除外すればよく，

$$Nt(L)=4^{(m-q)}\times((4^{L}-1)/4^{L}) \quad (7)$$

と書くことが出来る．また，これらのデータを送る，相手ＰＥの種類Na(L)は，

$$Na(L)=(4^{L}-1) \quad (8)$$

と書くことができる．

ここで，式（７）と式（８）を用いて，N=4096(=4⁶)，P=256(=4⁴)とP=64(=4³)の場合の通信回数を見積もる．各アルゴリズムについてまとめると表１のようになる．この結果は，転送がatomic typeで行われるのかそうでないのかが重要なポイントとなり atomic typeしかサポートしていなければ，Multi-satage方が，structやarray全体を一括して転送する機能をサポートしていれば，Stage-by-stage法が有利であることを示している．

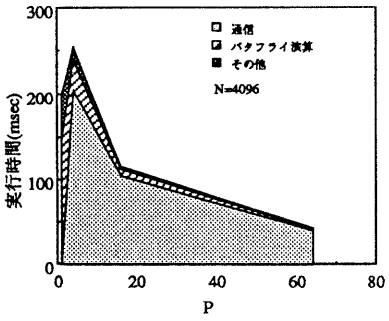
アルゴリズム		Nt(L)	Na(L)	転送回数	通信回数
Algorithm 1	a	$4^1-1=3$	$4^2\times(4^1-1)/4^1=12$	4	12×4=48
	b				3×4=12
	a	$4^1-1=3$	$4^3\times(4^1-1)/4^1=48$	3	48×3=144
	b				3×3=9
Algorithm 2		$4^2-1=15$	$4^2\times(4^2-1)/4^2=15$	2	15×2=30
		$4^3-1=63$	$4^3\times(4^3-1)/4^3=63$	1	63×1=63

表１ Nt(L)およびNa(L)と通信回数との関係

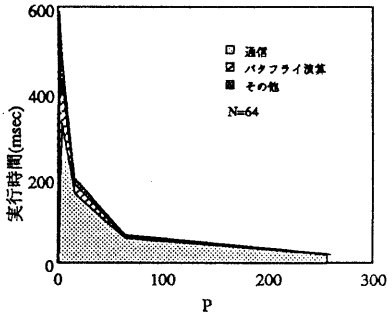
５．４．処理時間の実測値

ここでは，本アルゴリズムをCM-5とnCUBEに実装し，処理時間を実測した結果について述べる．使用するCM-5とnCUBEのＰＥ数はそれぞれ，６４個と２５６個である．

アルゴリズム１の１aおよび１bについて，データ数は4096個，ＰＥ数はCM-5が1,4,15,64個，nCUBEが1,4,15,64,256個としたときの実測値を図にすると，それぞれ，図６と図７のようになる．図６に示すように，アルゴリズム1aのようにatomic typeのみの通信を用いる場合，CM-5，nCUBEともに，その処理時間のほとんどを通信時間が占めることになってしまうことが分かる．特にCM-5の場合は，１個だけで実行したときに約200msecであった実行時間が，４個で実行した時に約254msecに増加してしまっている．一方，図７に示すように，アルゴリズム１bでは，データを一括して転送する事で，通信時間は減少し，大幅な速度向上が得られることが分かる．これは先の表１により得られた結果と一致する．

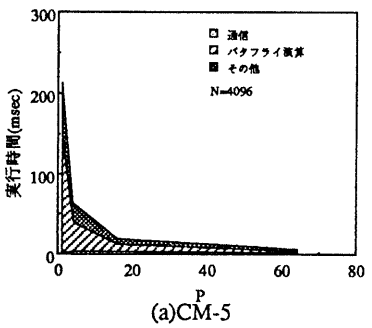


(a)CM-5

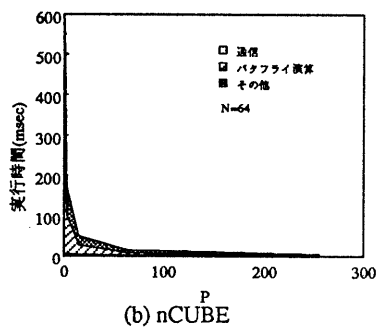


(b) nCUBE

図6 アルゴリズム1aの実行時間
(a)CM-5, (b)nCUBE.



(a)CM-5



(b) nCUBE

図7 アルゴリズム1bの実行時間
(a)CM-5, (b)nCUBE.

6. むすび

グローバルネットワークを用いた基数4の並列ＦＦＴアルゴリズムを説明し，その実装についても述べた．本アルゴリズムを実装する際には，転送がatomic typeで行われるのかそうでないのかが重要なポイントとなる．もし，atomic typeしかサポートしていなければ，Multi-satage方が，structやarray全体を一括して転送する機能をサポートしていれば，Stage-by-stage法が有利であることを示した．また，CM-5とnCUBEへ実装したときの実測値も示した．

参考文献

(1)J.W.Cooley and J.W.Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Math.Comput., Vol.19, pp.297-301(1965).

(2)川又，樋口，”多次元ディジタル信号処理の基礎（VI・完）”，電情通学誌，Vol.74, No. 10, pp. 1098-1108(1991).

(3)馬場敬信，”超並列マシンへの道”，情報処理，Vol.32, No. 4, pp.348-364 (1991).

(4)武田，丹野，堀口，"並列ＦＦＴと通信時間の評価"，情処研報 94-ARC-104， pp.57-64 (1994

(5))"MasPar Parallel Application Language (MPL) Reference Manual," MasPar Computer Corporation, Document No. 9302-0000(1991).

(6)伊達玄（訳）／A. V. Oppenhein and R. W. Schafer著，”ディジタル信号処理”，コロナ社(1978).

(7)佐川，本間（訳）／HJ.Nussabaumer著，”高速フーリエ変換のアルゴリズム”，科学術出版社(1989).

(8)山田実，”CM-5”，並列処理シンポジュウムJSPP'92論文集，pp.39-43(1992).

(9)"PARAGON Supercomputers," Intel Corporation, No. 203/6/92/10k/GA(1992).

(10)"nCUBE Supercomputers," NCUBE Corporation, No. G2-0100(1989).

(11)梅尾博司，”超並列計算機アーキテクチャとそのアルゴリズム”，共立出版(1991).