並列FFTアルゴリズムと並列計算機への実装

5F－8

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

１．はじめに

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]．

本文では，基数４のFFTを実現する２つの並列ＦＦＴアルゴリズム，Stage-by-stage法とMulti-stage法について述べ，Stage-by-stage法をCM-5およびnCUBEへ実装した結果についても示す．

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．

ここで，$n_{i-1}=0,1,2,3$をパラメータとして，$g_i(n_{i-1})=g_i(n_0,n_1,...,n_{i-1},k_{m-i-1},...,k_0)$はＦＦＴの第i段目の出力を表し，$g_0(k_{m-1},...,k_1,k_0)$は初期データ$g(k_{m-1},...,k_1,k_0)$を表すとすれば，フーリエ係数$G(n)=g_m(n_0,n_1,...,n_{m-2},n_{m-1})$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を第m段目（最終段）まで計算することにより求められる．

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

ここで，$v_i=W^{h\cdot i}$，i=0,1,2,3であり，$j=\sqrt{-1}$である．ただし，$h=(4^{m-i}n_0+4^{m-i+1}n_1+\cdots+4^{m-2}n_{i-2})$である．

Parallel FFT Algorithms and Implementation to Massively Parallel Computers

*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

３．　プロセッサアレイの構成

本文ではP=4^q個の演算要素（ＰＥ）から成るSIMD型並列計算機を仮定する．各ＰＥは，$PE(z_{q-1},z_{q-2},...,z_1,z_0)$と表す．ただし，$m \geqq q+1$である．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．通信ネットワークは，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を持つものとする．通信において，データがをatomic type単位で行う場合とstructやarray全体を一括して転送する場合を考え，それぞれatomic転送，一括転送と呼ぶことにする．

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．相似割付法は，上位q桁が等しいデータを同じPEに格納する方法であり，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．一方，重畳割付法は，下位q桁が等しいデータを同じPEに重ねて格納する方法であり，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．

４．２．　Stage-by-stage法 - アルゴリズム１

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図１のようになる．ここで，FFT_stage_i()は，i段目のFFTを行う関数である．ReMapping()は，データを再配列する関数である．この時，転送の方式でアルゴリズムを分け，atomic転送を用いたアルゴリズムをアルゴリズム１a，一括転送を用いたアルゴリズムをアルゴリズム１bと呼ぶこととする．

４．３．　Multi-stage法 - アルゴリズム２

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

ズムは図２のように記述される．

５．通信コストの評価

ここでは，本アルゴリズムの通信コストを評価する．P=42,N-43としたときの，再割付回数とグローバルネットワークによる通信回数を表１に示す．Multi-stage法は，再割付回数は少なくなるが，通信回数については一括転送を採用したStage-by-stage法が最も少なくなる．

次に，アルゴリズム１の並列計算機への実装ついて述べる．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それぞれ，図３と図４に示す．図３に示すように，アルゴリズム1aのようにatomic転送を用いる場合，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４に示すように，アルゴリズム１bでは，同一ＰＥへのデータを一括転送する事で，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．

６．おわりに

２つの基数４の並列ＦＦＴアルゴリズム，Stage-by-stage法とMulti-satage法を説明した．さらに，Stage-by-stage法をCM-5およびnCUBEへ実装したときの実測値も示した．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転送しかサポートしていなければ，Multi-satage法が，一括転送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．

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).

```
// Algorithm 1 Stage-by-stage

// 重畳割付
SuperpositionMapping();

// i段目のFFT と 再割付
for (i=1; i<=q; i++) {
    FFT_stage_i( i, 1);
    ReMapping( i, 1);
}
// (q+1)～m段目のFFT :(m-q)段
for ( k=1; i<=m; i++, k++) {
    FFT_stage_i( i, k);
}
```

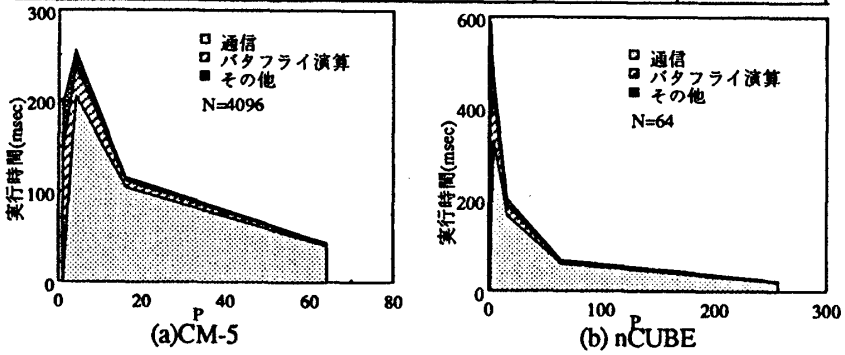
図１ アルゴリズム１

```
// Algorithm 2 Multi-stage

// 重畳割付
SuperpositionMapping();

// j×f～(j×f+(m-q))段目のFFT
// と 再割付
for (j=0, f=ceil(q/(m-q)); j<f; j++) {
    for (k=1; k<=(m-q); k++) {
        i= j*f+k;
        FFT_stage_i( i, k);
    }
    ReMapping( j, (m-q));
}
// FFT (m-(m-q)*f)段 : 最大(m-q)段
for ( k=m-(m-q)*f; i<=m; i++, k++) {
    FFT_stage_i( i, k);
}
```

図２ アルゴリズム２

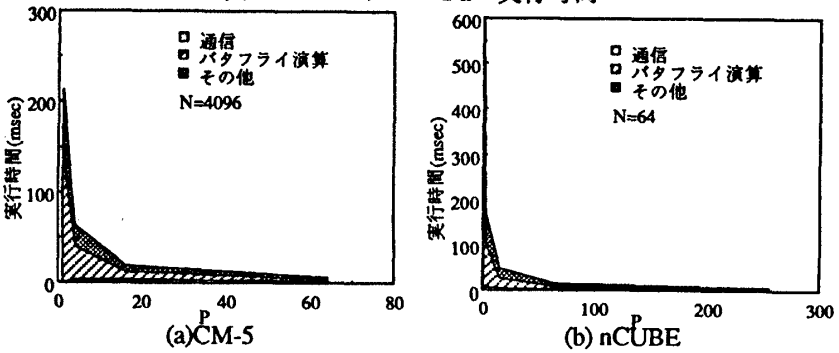
表１ 再割付回数と通信回数

アルゴリズム	転送先数	要転送データ数	再割付回数	通信回数
Algorithm 1a	4^1-1=3	$4^3\times(4^1-1)/4^1$=48	3	48×3=144
Algorithm 1b	4^1-1=3	$4^3\times(4^1-1)/4^1$=48	3	3×3=9
Algorithm 2	4^3-1=63	$4^3\times(4^3-1)/4^3$=63	1	63×1=63



(a)CM-5　(b) nCUBE

図３ アルゴリズム１aの実行時間



(a)CM-5　(b) nCUBE

図４ アルゴリズム１bの実行時間