計算機アーキテクチャ 109－4
（１９９４．１２．１３）

マルチポートメモリを用いたハイパーキューブ結合マルチプロセッサの通信性能

井口 寧　　堀口 進

北陸先端科学技術大学院大学 情報科学研究科

近年のコンピュータの高速処理への要求に対する１つの解決方法として多数のMPUを用いた並列処理があげられる．並列処理を行なう場合プロセッサ間通信が処理能力全体を規定する重要な要素なる．従来のハンドシェイクによるプロセッサ間結合では通信を行なっている間プロセッサは通信に占有されるが，複数の同時入出力可能なマルチポートメモリを通信に用いることにより，通信とプロセッサによる演算が並行に実行でき，システム全体の処理能力が向上する．本稿では高い通信能力を持つハイパーキューブ型マルチプロセッサにマルチポートメモリを用いることによりプロセッサ間通信能力を高めたシステムを提案し，試作システムを構築した．試作システムを用いて通信能力を評価した結果，従来のハンドシェイクを用いた通信方式より通信効率が向上することが分かった．

Communication Performance of Hypercube Multiprocessor System with Multiport Memory

Yasushi Inoguchi　　Susumu Horiguchi

Department of Information Science,
Japan Advanced Institute of Science and Technology

There are many types of interconnection networks for multiprocessor systems. Most of multiprocessor systems adopt the message passing scheme for processor communication. However the message passing scheme has a heavy overhead in processor communication, because every processor has to move data in their own memory to network interface. It is very important for massively parallel system to reduce the communication overhead.

We propose a hypercube multiprocessor using a multiport memory which is able to reduce the communication overhead. A prototype multiprocessor system has been constructed to evaluate the communication performance of a multiport memory scheme. The communication performance are also evaluated by implementing parallel FFT on the parallel machine.

1 はじめに

近年 VLSI 技術はめざましく発展し, 多数のプロセッサを用いたマルチプロセッサ・システムが盛んに研究されている. 一方で, 自然科学分野におけるシミュレーション, VLSI の論理シミュレーションなど, 大規模科学技術計算の需要は年々増大しており, これらの多くの分野で各種のアルゴリズムが本質的に並列実行が可能であることから, マルチプロセッサシステムによる高速処理に期待が寄せられている.

しかしながら, これまでの並列処理に関する研究の過程で並列計算機の処理性能は, 計算量複雑度 (Computational Complexity) よりも通信複雑度 (Communicational Complexity) に支配されることが分かってきた [1]. 従って効率良く並列処理を行なうためには, プロセッサ間の通信を効率良く行なわなければならない. 一方近年では, スレーブプロセッサに用いられるマイクロプロセッサ (Micro Processing Unit; 以下 MPU) は RISC 化により大幅に処理性能が向上したため, メモリ及びプロセッサ間通信ネットワークインターフェース (Network Interface; 以下 NI) の動作速度と MPU の処理速度の乖離が大きく, システムの内部バス速度がシステム全体の処理性能を決定する大きな要因となってきている. これに対し, プロセッサ間通信速度は, 光ファイバ接続などの採用により, 更に向上する余地が大きい. このため, プロセッサ間通信の効率を向上するためには, プロセッサ要素の内部バス上データ移動を減らすことが重要である.

スレーブプロセッサ内システムバス上のデータ移動を減らす手段は大別して MPU に直接 NI を接続する方式とメモリに NI を接続する方式の 2 方式になる. MPU に直接 NI を接続する方式は, 例えばトランスピュータなどで実現されており, MPU で計算された結果を即座に別のプロセッサに送る場合, システムバスをアクセスすることなしに通信可能であるため, 細粒度の並列処理に効果的である. メモリに NI を接続する方式は, メモリブロック内の大量データ転送時に内部バスをアクセスする必要がないため, 大規模行列計算などの大規模粗粒度並列処理に有効である. しかしながら, この方式で高効率な処理を行なうためには, メモリの NI に直結されるポートとシステムバスに接続されるポートが独立に動作する必要がある. 現在, 2 つの入出力を同じメモリサイクル内で実行可能なマルチポートメモリが開発されている [2] [3]. マルチポートメモリを NI に用いることにより, MPU によるプロセッサ要素の内部バス経由アクセスとプロセッサ通信によるアクセスを同時に行なうことができ, プロセッサ間通信のオーバーヘッドを軽減することができる. しかしながら, 現在までマルチポートメモリを用いた通信によって並列プログラムを実際に実行できるシステムにより, 通信性能を評価した例はそれほど多くない.

本稿では, ネットワーク構造に優れ高い通信性能を持つハイパーキューブ型マルチプロセッサに, マルチポートメモリを用いてプロセッサ間通信能力を高めたシステムを提案し, 試作システムを構築した. ハイパーキューブ結合は, 既に試作機で多く用いられ [4] [5], 商業機としても NCUBE/2, CM-2 [6] [7] などで採用されているが, これらのマルチプロッセッサ・システムでは I/O インターフェースにより結合されているため, メッセージ交換時 MPU とネットワークインターフェースとの競合により, 通信オーバーヘッドが無視できない. マルチポートメモリを使った通信は, この競合がないので, オーバーヘッドの大幅な削減が期待できる. そこで, 試作したシステムを用いて, マルチポートメモリを用いたプロセッサ間通信の基礎的な通信性能, 及び FFT 処理での性能について実験的検討を行なう [8].

2 プロセッサ間通信方式

2.1 ハンドシェイク方式

図 1 (a) に, ハンドシェイクを用いてプロセッサ間通信を行なう方式 (ハンドシェイク方式) のスレーブプロセッサ内データ移動の様子を示す. ハンドシェイク方式は, 実装が容易であり, CM-5[9] などの多くのマルチプロセッサシステムで使用されている. ローカルメモリ, MPU, 及び NI すべてがプロセッサのバスに接続される. ハンドシェイク方式によってプロセッサ間通信を行なう場合, プロセッサ自身がローカルメモリから NI に通信データを転送しなければならない. このために, プロセッサはまずメモリからデータを読みだし (i), 次に NI に書き出す (ii) ので, 最低 2 回のバスサイクルが必要である. また転送を実行するための命令をプロセッサがフェッチする必要があるので, 更に 1 回以上のバスサイクルが必要

である．従って通信に要する時間を次のように定めれば，

m	通信メッセージの平均パケット数
l	1パケットのワード数
T_r	ルーティングなど通信開始に要する時間
T_n	1パケット分の調停時間
t_m	主記憶のサイクル時間
t_i	NIのアクセス時間
t_α	MPUがデータ転送命令を読み出す時間

ハンドシェイクを用いたプロセッサ間通信に要する時間全体T_tは，

$$T_t = T_r + m(T_n + l(t_m + t_i + t_\alpha)) \qquad (1)$$

となる．

2.2 マルチポートメモリを用いたプロセッサ間通信

マルチポートメモリは，複数の同時動作可能な入出力端子をもつ．本稿で議論するマルチポートメモリの入出力は，1パラレル，1シリアルの2つの入出力端子を用いて行なわれる．内部には通常の任意アクセス可能なメモリ回路に加え，シリアル入出力用のデータレジスタ及びシリアルデータセレクタ (SDS) を持つ．

シリアル出力を行なう際には，ロウ (row) アドレスを指定する．すると，指定された256 × 4bitが，メモリセルから1メモリサイクルでトランスファ・ゲートを通じてSDSに転送される．この後，外部から転送クロックを与えると，1×4bitづつデータがシリアルI/Oから出力される．シリアル入力を行なう場合，まず1 × 4bitのデータを転送クロックと同期してシリアルI/O端子に与える．すると，転送クロックごとにデータがSDSに格納される．データが256 × 4bit分格納されたあと，格納するロウを指定することによりトランスファ・ゲートを通じてメモリセルに転送される．シリアル入出力を行なう際のパラレル入出力との競合は，データセレクタとのデータ転送を行なう時のロウアドレスの指定時に限定される．また，SDSに対する入出力 (シリアル入出力) は，メモリセルに対するアクセスよりもかなり高速なので，転送クロックはメモリサイクルタイムよりも早い周期のクロックが入力可能である．従って，マルチポートメモリの最大の特徴は，シリアル入出力を行なう際にロウ全体が1メモリサイクルで転送され，かつSDSのアクセスとパラレル入出力のアクセスは並行に動作可能であるので，パラレル入出力のみのメモリに比べ2倍以上のバンド幅が得られることである．

マルチプロセッサシステムのメモリにマルチポートメモリを用いて通信を行なう方式 (マルチポートメモリ方式) は，高効率なプロセッサ間通信が可能である．図1 (b) に，マルチポートメモリを用いてプロセッサ間通信を行なう方式 (マルチポートメモリ方式) のデータ移動の様子を示す．

マルチポートメモリ方式によるプロセッサ間通信では，通信を行なうデータはマルチポートメモリに格納される．マルチポートメモリのパラレル入出力はプロセッサバスに，シリアル入出力はNIに接続される．通信を行なう場合，MPUはマルチポートメモリの通信データが格納されているロウアドレスを指定する．すると，データはマルチポートメモリ内のSDSに1メモリサイクル時間で移され，その後転送クロックに従い1bitづつNIに転送される．この転送の間，MPUはシステムバスを使用することができるので，プロセッサの実行がブロックされることがない．また，マルチポートメモリのメモリセルもアクセス可能なので，次の転送データの用意をすることもできる．従ってマルチポートメモリ方式では，プロセッサとNIのバス競合は，ロウアドレスの指定時のみである．通信に要する時間をハンドシェイク方式と同様に定義すれば，マルチポートメモリ方式のプロセッサ間通信時間は，

$$T_t = T_r + m(T_n + t_m + l \times t_i) \qquad (2)$$

となる．lはSDSの大きさによって制限されるので，SDSが小容量である場合及び通信パケット長が短い場合は相対的にT_n, T_rが占める時間が大きくなり効率が低下する．もし，

$$t_i = t_\alpha = t_m \qquad (3)$$

と仮定するならば，メッセージ長がT_n, T_rに対し十分長い場合，マルチポートメモリ方式はハンドシェイク方式の約3倍の通信性能を得ることができる．まとめると，マルチポートメモリ方式の通信は次の特徴を持つ．

1. パラレル入出力のバンド幅を損なうことなくシリアル入出力が可能である．SDSは1メモ

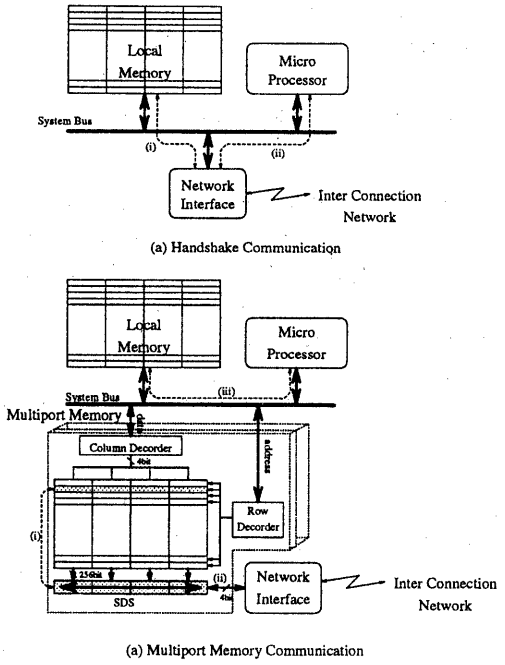

図 1: プロセッサ間通信方式

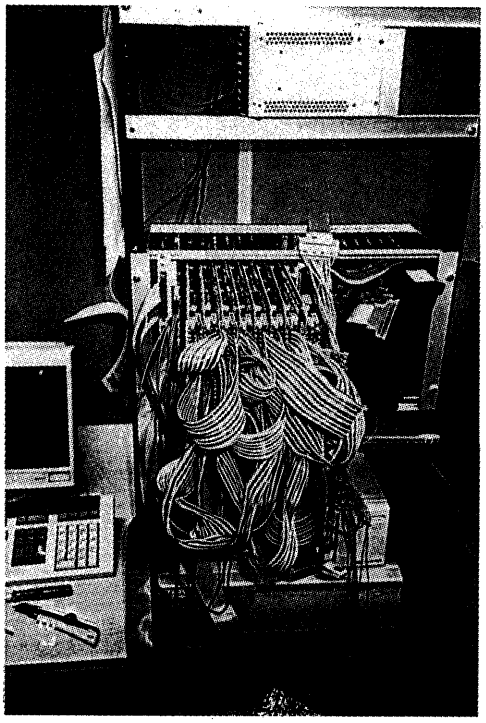

図 2: 8プロセッサ構成の試作システム

リサイクルで全データをメモリセルと交換可能なので, シリアル I/O によるパラレル I/O のメモリバンド幅への影響が少ない.

2. MPU はデータ転送に際しデータ入出力のロウの指定のみを行えば良いので データ転送における MPU の負荷が大変少ない.

3. メモリのブロック転送を効率良く行なうことができる. SDS の容量単位でメモリの入出力を行なうので, メッセージ長が SDS 容量の整数倍のとき最も効率が良い.

3 試作ハイパーキューブ型マルチプロセッサシステム

3.1 試作システムの構成

実際にマルチポートメモリを通信チャネルとして使用した場合の性能を評価するために, マルチポートメモリとして現在入手可能なデュアルポートメモリを用いた試作システムを試作した. 試作システムの写真を図 2 に示す.

本マルチプロセッサシステムは 8 台のプロセッサからなるハイパーキューブ結合方式を採用している. ハイパーキューブシステムは既にいくつか発表されている. それらと比べ本システムの特徴はプロセッサ間通信にマルチポートメモリを用い, 通信によるオーバーヘッド時間の削減を図ったことと, 遠隔プロセッサに対し途中のパケット転送を介することなく直接通信可能としたことである. 試作システムの構成概念図を図 3に示す. 全体のシステム構成は, ホストシステム及び 8 台のスレーブプロセッサより成る. スレーブプロセッサの MPU に MC68000, 浮動少数点プロセッサとして MC68882 を用いている. スレーブプロセッサはローカルメモリ, マルチポートメモリ, ハンドシェイク回路及び通信ゲート回路を持つ. マルチポートメモリは 256 × 256 × 4bit 構成であり, パケット長を 256 ワードまで大きくすることができる. 通信ゲート回路は, プロセッサとハイパーキューブ網の間, 及びハイパーキューブ網のリン

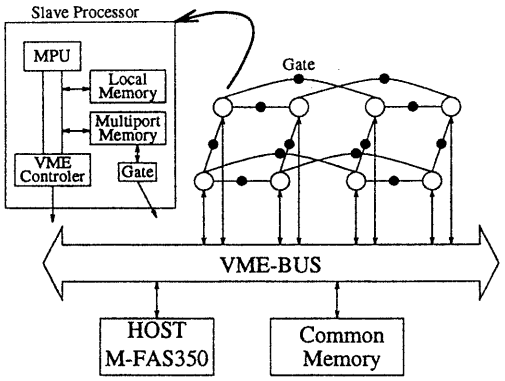

図 3: 試作ハイパーキューブ型マルチプロセッサシステムの構成概念図

クの接続または切り離しを行なう. プロセッサとハイパーキューブ網間のゲートはプロセッサの切り離しを可能とするので, 中継プロセッサはリンクの確保だけを行えば良く中継処理の負荷が少ない. ハイパーキューブ網のリンク上にあるゲートはノードに位置するプロセッサから制御可能であり, ハイパーキューブ網上で直接通信可能な任意の経路を形成することができる. このため, 遠隔プロセッサ間でプロセッサが直接通信でき, 通信遅延が軽減される.

スレーブプロセッサはハイパーキューブ・ネットワークの他に VME バスにより全プロセッサがホストコンピュータと接続されている. VME バス上の共有メモリを使用し, ホストコンピュータとのデータ転送, 及び I/O による割り込みを行なうことにより, ホストコンピュータと協調して処理を行なうことができる. スレーブプロセッサのプログラムはホストコンピュータ上でコンパイルされる.

3.2 通信モニタシステム

マルチポートメモリ方式による通信と演算の並行実行を可能とするため, 通信モニタシステムは次の要件を満たす必要がある.

- 応用プログラムと独立したプロセスとして動作すること.
- 応用プログラムによる非同期通信が可能であること.

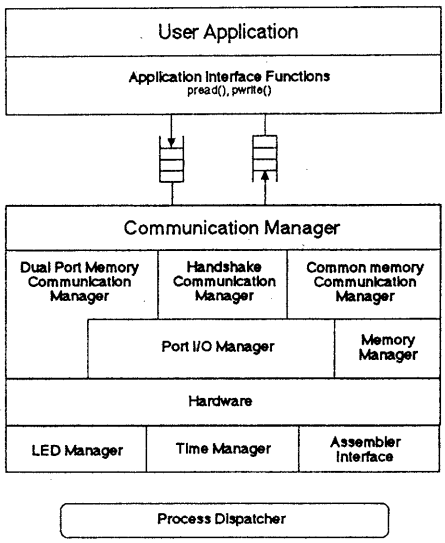

図 4: 通信モニタシステムの構成図

試作システムのスレーブプロセッサ上のソフトウェア環境は, 通信を管理する通信マネージャ(Communication Manager), 通信マネージャを呼び出すための通信ライブラリ, およびプロセスを管理するプロセスディスパッチャから構成される. 本通信モニタシステムの構成を図 4 に示す. 通信マネージャはアプリケーションプログラムとは独立に動作し, 常時通信チャネルおよび通信キューを監視する. 通信チャネルにメッセージが到着した場合, 受信処理または中継処理を行なう. アプリケーションプログラムによって通信キューにメッセージが入れられた場合は, 送信処理を開始する. 通信モニタシステムは, 機能ごとにモジュール化されており, 通信時のプロセス切り替えや通信方法の変更などが容易である.

本システムが応用プログラムに対して提供するプロセッサ間通信機能は以下の関数群である.

pread()	データ受信
pwrite()	データ送信
pwait()	通信の待ち合わせ

pread,pwrite 関数は同期通信と非同期通信が可能である. 同期通信では, 通信メッセージは即座に通信マ

ネージャに渡され, 通信が行なわれる. アプリケーションプログラムの実行は, データ転送が完了するまでブロックされる. このため同期通信は各プロセッサの処理時間がそろっている応用に対して, プログラムが容易であるという利点を持つ.

一方非同期転送では, 送信側はデータが用意できた時点でプロセッサ間通信の通信キューに登録し, 送信可能になると通信マネージャによってバックグラウンドで送信される. 受信側は同様にデータを受信すべき変数が用意できた時点で通信キューに登録し, バックグラウンドで受信を行う. 通信キューに登録した変数は通信が完了するまでデータの変更が制限されるが, プロセッサ間通信と MPU による演算が同時に実行でき, 並列処理の効率が高い. 通信キューに登録した変数の変更が必要なときには, pwait 関数を用いて同期が行われる.

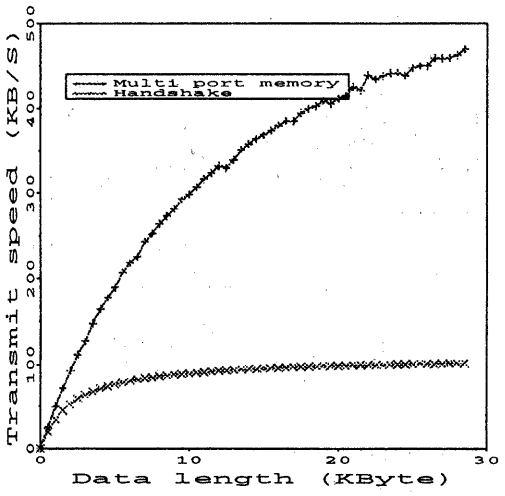

図 5: プロセッサ間通信速度

4 試作システムの通信性能評価

4.1 隣接プロセッサ間通信速度

試作システムを用いて, マルチポートメモリを用いた通信方式とハンドシェイクによる通信方式の通信性能を評価した. マルチポートメモリ方式と比較検討するための疑似ハンドシェイク方式による通信を行なう. ハンドシェイク方式はマルチポートメモリ方式に比べ, プロセッサが通信データをメモリから NI に転送する必要がある. そこで次のような疑似ハンドシェイク方式を考える. 疑似ハンドシェイク方式では通信データはローカルメモリに置く. 通信時に, ハンドシェイク方式で必要なメモリと NI 間のデータ転送をシミュレートするために, ローカルメモリからマルチポートメモリを介してデータ転送を行なう. すなわち, マルチポートメモリ方式では通信データがマルチポートメモリに格納されるのに対し, 疑似ハンドシェイク方式ではローカルメモリに格納される. このような疑似ハンドシェイク方式を用いると, ハンドシェイク方式とマルチポートメモリ方式の通信効率を, 完全に同条件で比較することができる.

疑似ハンドシェイク方式とマルチポートメモリ方式のプロセッサ間通信能力について, 試作システムを用いて測定した. 測定は送信側スレーブプロセッサに接続したロジックアナライザによって行なった. 通信に要した時間は, アプリケーションプログラムが pwrite 関数を呼び出だす直前から, pwait 関数終了直後までの時間である. pwrite 関数と pwait 関数は連続して呼び出されるので, アプリケーションプログラムが通信だけを行なう時の通信時間が得られる.

マルチポートメモリ方式及び疑似ハンドシェイク方式の通信時間を次に示す.

マルチポートメモリ方式:

$$time = 1.52 \times KByte + 19.0(mS) \qquad (4)$$

疑似ハンドシェイク方式:

$$time = 9.32 \times KByte + 20.4(mS) \qquad (5)$$

疑似ハンドシェイク方式とマルチポートメモリ方式の通信速度を図 5 に示す. 疑似ハンドシェイク方式ではローカルメモリと NI 間のデータ転送のため, 通信データ長が大きくなっても通信速度はあまり改善されない. このため, 疑似ハンドシェイク方式の最大通信速度は約 107.3KByte/s である. 一方, マルチポートメモリ方式では, ローカルメモリと NI 間のデータ転送のオーバーヘッドが削減された分, 高い通信速度が得られた. 両方式の通信速度は約 6 倍異なる. この原因は, 通信データの転送ループにおいてジャンプ命令のプリフェッチミスのため, 式 (3) の T_α が比較的大きくなるためである.

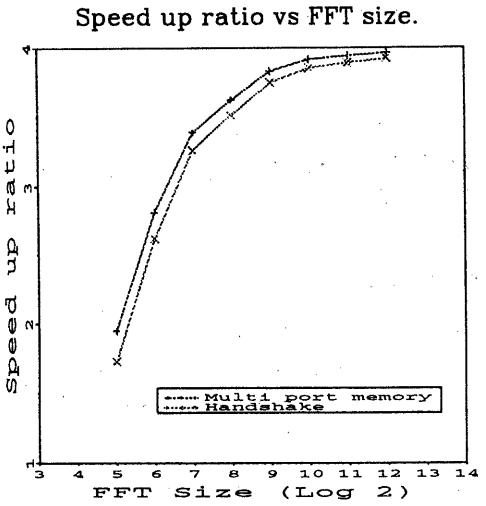

図 6: FFT 処理における速度向上比

4.2 並列 FFT における通信性能

実用的なアプリケーションの例として, 並列複素 FFT 処理を行ない, 処理性能を評価した. ハイパーキューブシステムで FFT 処理を行なう場合, 対象となるデータ数 N の配列を, プロセッサ台数分に均等に分割する. 次に分割した部分配列を順にプロセッサに割り当てる. すると, 各プロセッサは遠隔プロセッサ間通信を行なうことなく, 隣接プロセッサ間の通信だけで FFT 処理を行なうことができる.

このような方法で, 4 プロセッサからなる試作システム上で通信実験を行なった場合の速度向上比を図 6 に示す. この測定では, スレーブプロセッサ上で実行可能な浮動小数点演算ライブラリが用意できないので, 実際の FFT 処理は行なっていない. 演算部分は, あらかじめホストコンピュータ上で FFT の部分演算を行なうことにより, 実行時間とを決定し, スレーブプロセッサ上では決定した実行時間分だけの待ちループを行なった後, プロセッサの担当する FFT の部分演算の結果を通信する. 本節で議論する FFT 処理時間は, FFT 演算と実際のデータ通信を含めた時間を測定している.

図 6から分かるように, マルチポートメモリ方式では疑似ハンドシェイク方式よりも高い速度向上比を示す. 4 プロセッサ構成のシステム上で, FFT のサイズが 2^{12} の時, 擬似ハンドシェイク方式では 3.92 の速度向上比が得られるのに対し, マルチポートメモリ方式では 3.97 の速度向上比が得られた. FFT のサイズが十分大きい場合には, いずれの通信方式を用いてもほぼ 4 倍の実行速度が得られ, 効率が高い. FFT のサイズが小さい場合には, 両方式とも処理効率が低くなるが, 疑似ハンドシェイク方式に比べマルチポートメモリ方式の方が効率低下の度合いが少い. FFT のサイズが小さい場合に高速化できない原因は, 式 (2) で論じたように, パケット長が短いとマルチポートメモリ中の SDS の利用効率が低下するためである. 試作システムでは複素浮動少数点変数が 8Byte 長であり, パケットサイズが 512Byte 単位なので, FFT のサイズが 2^6未満ではパケットがデータで満たされず, パケットの利用効率が低下する. FFT のサイズが大きくなると, 1 プロセッサ当たりの演算量は $n \log n$ で増加するのに対し通信量は n で増加するので, 処理全体に占める通信の割合が減少する. このため, FFT のサイズが大きくなると, マルチポートメモリ方式と擬似ハンドシェイク方式の速度向上比の差は少くなる.

マルチポートメモリ方式による通信の実装上における興味ある問題として, (i) 通信と演算の並行実行が可能であるが, 通信と並行に実行可能な演算があるのかどうか, (ii) 並行実行の際に通信ルーチンから演算ルーチンへのコンテキストスイッチが行なわれるが, コンテキストスイッチの負荷は通信演算の並行実行可能な時間に比べ十分小さいかどうか, がある.

問題 (i) について, 応用プログラムとしての並列 FFT 処理の性質を考える. 並列 FFT 処理は 1 段演算するごとにハイパーキューブ上の隣合うプロセッサ間で通信を行なう. プロセッサを先行グループと後行グループに分けることによって, 先行グループは終了した演算を後行グループに送りながら次段の演算にとりかかることが可能である. 次の段では先行後行が入れかわる. 問題 (ii) つまりコンテキストスイッチの時間は, 試作システムではジョブ制御を含めたプロセス切替えには約 $300\mu S$, 割り込み処理には最低 $10\mu S$必要であり, プロセス切替えを行なう場合, 1 パケットの通信時間 $128\mu S$を越える. 従ってプロセス切替えを伴った通信は, 通信パケット長がかなり長くないと効率的でない. しかしながら, 通信処理自体が通信と演算の並行動作可能部分を含んでおり, かつ通信処理は割り込みを用いてプロセス制

御が可能であるため, 通信処理に必要な演算とマルチポートメモリによる通信は並行に実行できる. 試作システムでは並行動作可能な時間 $128\mu S$のうちおよそ半分を次のパケットの準備で消費している.

5 まとめ

現在までに多くのマルチプロセッサシステムが構築されてきたが, 従来のシステムでは, プロセッサ間通信を行なうネットワークインターフェース (NI) にハンドシェイクによるI/Oインターフェースを用いており, 通信オーバーヘッドが大きいという問題点があった. このため, 本稿では, マルチポートメモリを用いたハイパーキューブ型マルチプロセッサシステムを提案した. また, このマルチプロセッサシステムの通信性能について検討し, 実際に8台のスレーブプロセッサからなるシステムを試作し, 試作システムによる通信効率の測定を行なった.

基本通信性能の評価では, マルチポートメモリとハンドシェイク方式を等しい条件で比較するため, 疑似ハンドシェイク方式とマルチポートメモリ方式の通信性能評価を行なった. その結果, 疑似ハンドシェイク方式はローカルメモリとNI間のデータ転送のため, 通信性能がかなり低くなることが分かり, これに対しマルチポートメモリ方式では通信性能は通信同期処理によって制限される速度まで向上することが分かった. FFT処理では, 実際にFFTで用いられる通信を行ない性能評価を行なった. この結果, データサイズが通信パケットサイズよりも十分大きければ効率良く処理できることが分かった. データサイズが小さくなるとFFT処理の効率は低くなるが, データサイズが小さくなると, マルチポートメモリ方式と疑似ハンドシェイク方式の処理性能の差は大きくなり, マルチポートメモリ方式はFFT処理の効率低下を軽減することが分かった. またマルチポートメモリにより可能となった通信と演算の並行実行可能な時間のうち, 約50%が有効に利用できることが分かった.

今後の課題は, 通信メッセージ長によって通信実行中のMPUによる処理を, 割り込みによる通信マネージャとコンテキストスイッチを伴うプロセス切替えによる処理に, 自動切替え可能であり, 通信メッセージの動的スケジューリング対応可能な超並列システム向けの並列処理オペレーディングシステムの開発である.

謝辞 本研究の一部は (財) 日本科学協会研究助成及び文部省科学研究費助成を用いて行なわれた. 関係各位に感謝する.

参考文献

[1] S. Horiguchi, Y.Kawazoe, H.Nara: "An Algorithm of Parallel Processing for Integration of Ordinary Differential Equations", Trans. of IEICE, Vol.E70 pp.49-55 (1987)

[2] 松見一誠, 倉知 郁生: "1M ビット マルチポートメモリの開発" 沖電気研究開発 第138号 Vol.55 pp.3-8 (1988)

[3] "MB81461-12/15" 富士通半導体デバイス ED12-05111-7 pp.621-643

[4] 石川 勉: "通信特性向上のためのバス結合付加型ハイパーキューブアーキテクチャ" 信学論 D-I Vol.J73-D-I pp.415-423 (Apr. 1990)

[5] 酒居 敬一, 園田 幸生, 佐伯嘉崇, 阿江 忠: "マルチポートメモリ結合を用いる並列プロセッサMC1の試作例" IPSJ93-ARC-107-6, (Jly. 1994)

[6] Hayes,J.P.and Mudge,T.: "Hypercube Supercomputers," Proc. IEEE, Vol.77, No.12, pp.1929-1841 (Dec. 1989)

[7] 小柳 滋, 田邉 昇: "超並列マシンの実装技術" 情報処理 Vol.32 No.4 pp365-374 (Apr. 1991)

[8] "マルチポートメモリを用いたハイパーキューブ型マルチプロセッサの通信性能" 情報処理学会第49回全国大会講演論文集 (6) 2L-4, pp. 6.61-6.62

[9] "CM5" Thinking Machines Corporation