計算機アーキテクチャ　99－19
（1993. 3. 12）

相互結合網におけるフロー制御方式の通信パターンに基づく性能評価

船越 誠司†　　久我 守弘‡　　末吉 敏則†‡

† 九州工業大学 情報工学部 知能情報工学科

‡ 九州工業大学 マイクロ化総合技術センター

funakosi@mickey.ai.kyutech.ac.jp
kuga@cms.kyutech.ac.jp
sueyoshi@ai.kyutech.ac.jp

あらまし

ハードウェア記述言語による相互結合網のモデル化と，このモデルを用いて行ったシミュレーションによるフロー制御方式の通信パターンに基づく性能評価について述べる．相互結合網のモデル化にハードウェア記述言語を用いることにより，プロセッサ要素間のデータ転送動作を実際の回路に即して把握することができる．また，本シミュレーションでは実際の並列アプリケーションから取得した通信パターンを利用できるため，理論的な近似解析等と比較してより詳細な性能評価を行うことが可能である．本稿では，ハードウェア記述言語による相互結合網のモデル化の詳細について述べた後，代表的な3種類のフロー制御方式についてその違いが相互結合網の性能にどの様な影響を及ぼすかを報告する．

和文キーワード　相互結合網，フロー制御，通信パターン，性能評価，ハードウェア記述言語

Performance Evaluation of Flow Control Methods based on Traffic Patterns in Interconnection Network

Seiji Funakoshi†　　Morihiro Kuga‡　　Toshinori Sueyoshi†‡

† Department of Artificial Intelligence
Kyushu Institute of Technology
Iizuka, 820 Japan

‡ Center for Microelectronic Systems
Kyushu Institute of Technology
Iizuka, 820 Japan

funakosi@mickey.ai.kyutech.ac.jp
kuga@cms.kyutech.ac.jp
sueyoshi@ai.kyutech.ac.jp

Abstract

This paper presents the modeling of the interconnection network with Hardware Description Language (HDL) and the performance evaluation of the flow control methods based on traffic patterns utilizing this model. With HDL, the data transfer between processor elements in the simulation is executed in the same manner as in actual circuits. With traffic patterns obtained from the actual parallel applications, we can perform more detailed performance evaluation of the flow control methods rather than theoretical approximation analysis.

英文 key words　interconnection network, flow control, traffic patterns, performance evaluation, hardware description language

1 はじめに

マルチプロセッサのような並列計算機システムにおいて，プロセッサ要素間を相互に結合しその間の通信を可能にする相互結合網は，システムの性能を左右する重要な構成要素といえる．相互結合網を設計する場合，設計者はプロセッサ要素間の結合形態を決定するトポロジ，プロセッサ要素間でデータの転送経路を決定するルーティング方式，転送するデータに対してトラフィックを制御し，通信資源の割当てを行うフロー制御方式の3つの要素について考慮しなければならない．これら3つの要素を適切に選ぶとき，並列計算機システムにおいて高い並列処理効率を期待できる [1]．

現在，我々の研究室では上記の3つの要素をもとに様々な相互結合網について性能評価を行っているが [2][3]，今回はこのうちフロー制御方式に着目し，フロー制御方式の違いが相互結合網の性能に与える影響を調べた．ここでは，ハードウェア記述言語を用いて相互結合網のモデル化を行い，代表的なフロー制御方式である store and forward, wormhole, virtual cut-through の3種類についてシミュレーションによる性能評価を行った．相互結合網のモデル化にハードウェア記述言語を用いることにより，プロセッサ要素間のデータ転送動作を実際の回路に即して把握することができるので，理論的な近似解析等と比較して各フロー制御方式について詳細な性能評価を行うことができる．

また，並列計算機システムの並列処理効率を左右する要因として，上記の3つの要素に加えてプロセッサ要素間の通信パターンがある．通信パターンは実行するアプリケーションに依存し，相互結合網の通信遅延時間に影響を及ぼすため，相互結合網の性能を評価する上で考慮する必要がある．最近，相互結合網の性能評価のために理論解析やシミュレーションが行われているが，様々な通信パターンを踏まえた性能評価はほとんど行われていない [2][4][5]．これに対して，本シミュレーションでは，通信パターンとして実際の並列アプリケーションの通信パターンを利用でき，相互結合網の実効性能を様々な状況のもとで評価することができる．

本稿では，ハードウェア記述言語による相互結合網のモデル化，シミュレーション概要，およびシミュレーション結果に基づくフロー制御方式の性能評価について述べる．以下，第2章では相互結合網のモデル化の概要，モデル化に用いたハードウェア記述言語の特徴および作成したモデルについて述べる．第3章では，このモデルを利用したシミュレーションの概要について述べる．また，第4章では，規則的な通信パターンを用いたシミュレーションに対する性能評価について述べる．さらに第5章では，フロー制御方式の詳細な性能評価のために行った，実際の並列アプリケーションの通信パターンを用いたシミュレーションによる性能評価について述べる．

2 相互結合網のモデル化

相互結合網には大きく分けて静的網と動的網があり，フロー制御方式の性能評価を行う場合，モデル化およびシミュレーションはこの両方について行うことが望ましい．しかしながら，メッシュやトーラスなどの静的網の場合，フロー制御方式の他にルーティング方式も考慮しなければならない．つまり，静的網の場合，相互結合網の性能はトポロジが決まっていてもフロー制御方式とルーティング方式の両者から影響を受けることが予想できる．

一方，動的網のうち代表的な多段結合網であるオメガ網は，プロセッサ要素間に複数段の交換スイッチ群を配置し，その間をシャッフル置換で結合したものである．このオメガ網は送信側と受信側のプロセッサ要素を決定すると，その間の経路が一意に決定できるという特徴を持っている．したがって，オメガ網の場合にはフロー制御方式の違いによる性能への影響のみを見ることができる．そこで，今回は動的網のうち代表的な多段結合網であるオメガ網についてモデル化を行った．

2.1 ハードウェア記述言語

オメガ網のモデル化には，現在ハードウェア設計で広く使われている，Verilog HDL と呼ばれるハードウェア記述言語を用いた．Verilog HDL は回路の動作を C や Pascal のような手続き型言語に類似した言語で記述するもので，ゲートおよびスイッチレベルだけでなく動作レベルで相互結合網の動作を記述することができる．また，回路の動作タイミングや内部遅延の設定および変更が容易であり，これらを適度に設定することにより，実際の相互結合網で実行する場合に即したシミュレーション結果を得ることができる．さらに，Verilog-XL と呼ばれる強力なシミュレーション環境を利用して回路の動作を波形等で確認できるので，詳細な性能評価を行うことができ，記述した回路は最終的にハードウェア化もできるという特徴を有する [6]．

2.2 オメガ網のモデル化

オメガ網をモデル化するにあたり，プロセッサ要素間の結合方式としては密結合型，通信方式としてはパケット交換方式を想定した．モデル化では，まず図1に示す構成の交換スイッチをVerilog HDLで記述した．プロセッサ要素間でパケットを転送する場合，パケットをデータ線幅ごと（フリット）に分割して転送する．交換スイッチに入力されたフリットは，交換スイッチ内のバッファに保管される．制御回路はパケットの先頭フリットから転送先のアドレスを読み出し，適切なクロスバスイッチの状態(直行または交換)を要求する．この要求はアービタで調停され，要求に応じてクロスバスイッチの状態が設定される．今回行ったモデル化では，パケットの優先度を設けていない．このため，2つの制御回路が同時にアービタに対して要求を行った場合，アービタは両方の要求を交互に満たすように設計している．制御回路はクロスバスイッチが要求する状態になると，次の段の交換スイッチに対してフリットを転送する．

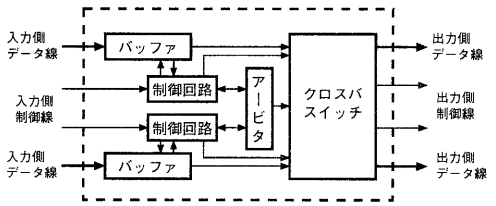


図 1. 交換スイッチの構成

store and forwardでは交換スイッチ内に1パケット分のバッファを持ち，バッファへのデータの入出力はパケット単位で行われる．wormholeでは交換スイッチ内に1フリット分のバッファを持ち，バッファへのデータの入出力はフリット単位で行われる．store and forwardと比較してハードウェア量は小さく，バッファへのデータの入出力がパイプライン的に行われるため転送遅延も短い．しかしながら，交換スイッチ内でパケットが競合して一方が待たされた場合，それに続く経路上のデータ転送はすべて待たされてしまう．これに対して，virtual cut-throughではstore and forwardと同じく交換スイッチ内に1パケット分のバッファを持ち，バッファへのデータの入出力はwormholeと同じくフリット単位でパイプライン的に行われる．

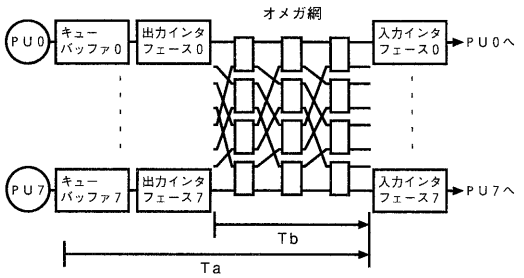


図 2. オメガ網のモデル化(プロセッサ要素数:8)

モデル化を行った交換スイッチは，動作クロックを20MHz，フリットサイズ（データ線幅）を16bitとした場合，交換スイッチ当たり40MByte/sec程度のデータ転送能力を持つ．この動作クロックやパケットのフリットサイズ等の値は容易に変更することが可能である．オメガ網のモデル化は，図1の交換スイッチ，プロセッサユニット(PU)および入出力インタフェース等の構成要素を図2のように接続して行った．

ここで，交換スイッチ以外の構成要素もすべてVerilog HDLで記述している．PUは記述された通信パターンにしたがってパケットを転送する．このモデルでは，PUが自分宛に送られてきたパケットの到着を知ることができるので，実際のアプリケーションで行われるパケットに対する応答やプロセスの同期などを行うことが可能となっている．

3 シミュレーションの概要

3.1 オメガ網の仕様

プロセッサ要素数が8のオメガ網についてシミュレーションを行う．このオメガ網の仕様を表1に示す．

表 1. オメガ網の仕様

動作クロック	20MHz	
フリットサイズ (データ線幅)	16bit	
パケットサイズ	128bit(8フリット)固定長	
バッファ長	wormhole	1フリット分
	virtual cut-through store and forward	1パケット分

3.2 通信パターン

シミュレーションでは，モデル化を行ったオメガ網に対して様々な通信パターンのパケットを転送し，wormhole, virtual cut-through および store and forward の 3 つのフロー制御方式について，それぞれの転送遅延の測定結果をもとに性能評価を行う．通信パターンとしては，C プログラムによって自動的に生成した規則的な通信パターンと，実際の並列アプリケーションから得た通信パターンの両方を用意した．

3.2.1 規則的通信パターン

規則的な通信パターンでは，ある時間間隔でパケットを生成して宛先に転送する．あるパケットが生成されてから次のパケットが生成されるまでの時間は，上限および下限付きのランダムで決定される．パケットの宛先としては，(a) 送信したプロセッサ要素を宛先として交換スイッチ内でのパケットの衝突を起こさないようにしたもの (contention free)，(b) ランダムなもの (random) および (c) 全てのパケットが同じ宛先であるもの (hot spot) の 3 種類を用意した．また，転送するパケット数はプロセッサ要素当たり 100 パケットとする．各 PU で生成されたパケットは，図 2に示す出力インタフェース側のキューバッファにキューイングされ，それぞれの宛先に転送される．

3.2.2 並列アプリケーションの通信パターン

実際の並列アプリケーションの通信パターンは，我々の研究室で構築した分散スーパーコンピューティング環境 (Distributed Supercomputing Environment, DSE) を利用して取得した [7][8]．DSE は，分散共有メモリ型並列計算機と等価な機能を分散システム上に実現したものであり，複数のワークステーションによって構成される (図 3)．DSE での分散共有メモリ型並列計算機の実現は，各ワークステーションにプロセッサ要素を割り当て，各プロセッサ要素間の結合形態を定義する接続情報ファイルを用いて意図する相互結合網の結合形態を仮想的に実現することで行われる．DSE は，実行した並列アプリケーションについて各プロセッサ要素での入出力メッセージの履歴をとることができるので，これを利用することにより，並列計算機システム上で実際の並列アプリケーションを実行した場合の通信パターンを取得することが可能である．

並列アプリケーションの通信パターンは，Sun microsystems 社の SPARC Station 2 を 8 台使用して取得した．取得した通信パターンは，図 4に示すように Verilog HDL で記述したプロセッサ要素モジュールに変換し，オメガ網のモジュールと組み合わせてシミュレーションを行う．

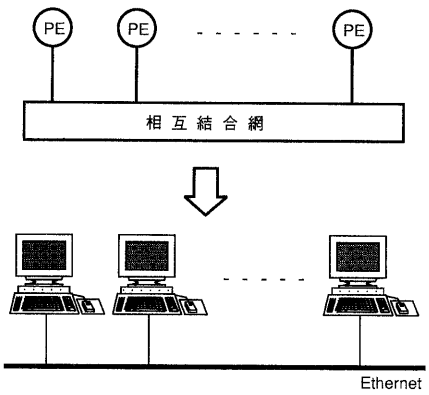


図 3. 分散スーパーコンピューティング環境 (DSE)

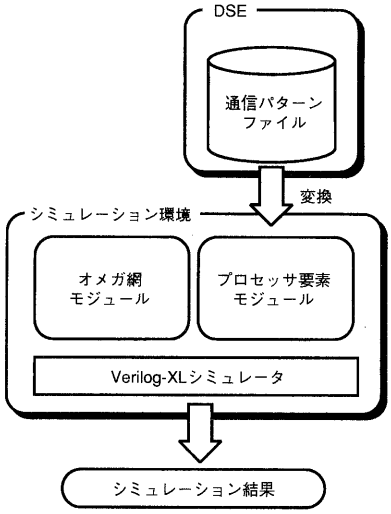


図 4. シミュレーション環境の構成

3.3 評価指標

シミュレーションにおける性能評価は，パケットの転送遅延の平均値をもとに行う．測定する転送遅延としては，PU がパケット生成を行ってからそのパケットが宛先に到

着するまでの時間 (図 2の T_a) と，オメガ網に対して入力インタフェースがパケットの送信を開始してからそのパケットが宛先に到着するまでの時間 (図 2の T_b) の 2 種類を調べた．

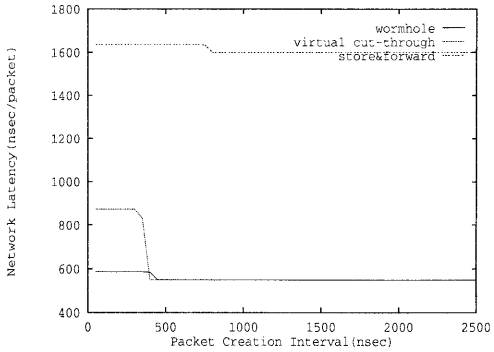
4 規則的な通信パターンを用いた性能評価

規則的な通信パターンを用いたシミュレーションの結果を図 5に示す．グラフは，横軸が各 PU でのパケット生成間隔，縦軸がパケットの転送遅延である．ここでは，紙面の都合により図 2の T_bについての測定結果のみを示す．

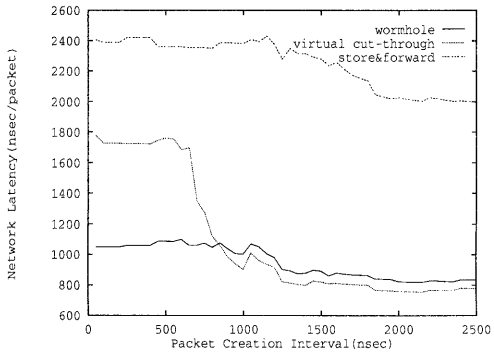
4.1 contention free

図 5(a) は contention free の場合の結果である．どのフロー制御方式の場合も，各 PU でのパケット生成間隔が大きい間，転送遅延はある一定値をとる．これは，パケット生成間隔がオメガ網のパケット転送遅延よりも十分大きいために，転送中のフリットが交換スイッチ内のバッファで待たされることなく次の交換スイッチに送られていることを示す．このとき，wormhole と virtual cut-through は全く同じ転送遅延になるが，これは 2 つのフロー制御方式がフリットをパイプライン的に転送するため，この状態のとき両者は性能的に全く同じものと見なせるからである．

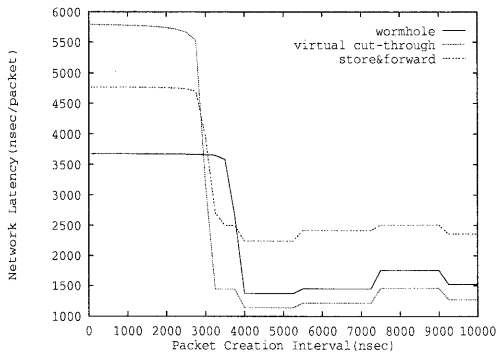
この状態からパケットの生成間隔を徐々に小さくしていくと，ある点で転送遅延が急に大きくなる．これは，この上昇点でパケット生成間隔がオメガ網のパケット転送遅延よりも小さくなったために，経路上の交換スイッチ内のバッファが一杯になり，転送中のフリットが待たされる状況が起きているためである．したがって，経路上のバッファが一杯になった後は再び転送遅延は一定値を採る．このとき，上昇の前後で virtual cut-through の転送遅延だけが大きく変化する．これは，パケット生成間隔が十分大きいときにはフリットがオメガ網の入力から出力まで通過するのに要するステップ数が 3 であるのに対して，経路上のバッファが一杯になった状態ではステップ数が 24(=8×3) となるためであると考えられる．これに対して，wormhole と store and forward では経路上のバッファが一杯になる前後でステップ数はそれぞれ 3，24 と変わらない．



(a) contention free



(b) random



(c) hot spot

図 5. 規則的な通信パターンに基づくシミュレーション結果

4.2 random

図 5(b) は random の場合の結果である．図 5(a) の場合と同様に，パケット生成間隔をだんだん小さくしていくとある点で転送遅延が大きくなる．この場合も上昇点の前後で転送遅延はほぼ一定の値を採るが，図 5(a) の場合とは違い，完全な一定値にはなっていない．これはパケットの宛先がランダムであるために交換スイッチ内でのパケット (フリット) の衝突が起こり，ネットワークのトラフィックが一定にならないためである．

また，パケット生成間隔が十分大きい間 wormhole と virtual cut-through はほぼ同じ転送遅延を示すが，図 5(a) のように全く同じ値にはならない．これは，パケット生成間隔が十分大きな場合にも交換スイッチ内でのパケットの衝突が起こるためである．つまり，交換スイッチ内でパケットの衝突が起こった場合，virtual cut-through は交換スイッチ内に 1 パケット分のバッファを持っているので，その交換スイッチに送られてくるフリットをバッファリングすることができるが，wormhole は交換スイッチ内に 1 フリット分のバッファしか持たないので，その交換スイッチに送られてくるフリットはすべて待たされてしまう．そのため，wormhole の方が多少転送遅延が大きくなってしまうと考えられる．

さらに，図 5(b) の場合，すべてのフロー制御方式において上昇の前後での転送遅延の差が図 5(a) の場合よりも大きくなっている．これは，パケット生成間隔が短くなったために経路上のバッファが一杯になったことに加えて，交換スイッチ内でのパケットの衝突によりフリットがバッファで待たされているためであると考えられる．

4.3 hot spot

図 5(c) は hot spot の場合の結果である．このグラフでも他の場合と同様に，パケット生成間隔を短くしていくとある点で転送遅延が急に大きくなる．このグラフでは，パケット生成間隔が十分大きい間は転送遅延にばらつきがあるのに対して，経路上のバッファが一杯になった後では転送遅延は一定値を採る．これは，図 5(b) の場合と同様に，パケットの生成間隔が十分大きい間でも交換スイッチ内でのパケットの衝突が起こっているため，転送遅延にばらつきが生じるが，経路上のバッファが一杯になった後では，この衝突が常に起こっているために転送遅延は一定値になると考えられる．ここで，上昇の前後で転送遅延の差が大きいのは図 5(b) の場合と同じ理由であるが，この差が図 5(b) よりも大きいのは，すべてのパケットが同じ宛先に集中するためにバッファ内で待たされる時間がはるかに大きくなっているためであると考えられる．

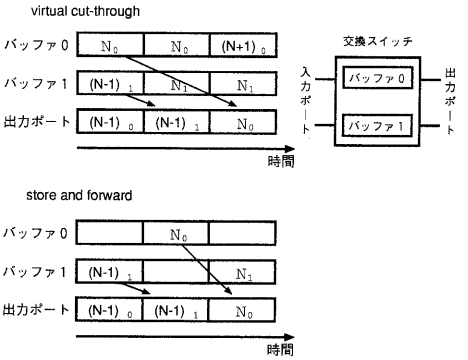


図 6. hot spot における交換スイッチ内のバッファ

図 5(c) のグラフでは他の 2 つの場合とは異なり，store and forward と virtual cut-through がほぼ同じところで上昇している．これは，hot spot の場合，パケットは経路上のすべての交換スイッチで他のパケットと衝突するので，交換スイッチ内のバッファ数が等しい store and forward と virtual cut-through がほぼ同じ性能になっていると考えられる．

また，経路上のバッファが一杯になったときに virtual cut-through の転送遅延がもっとも大きくなっているが，これは以下の理由による．ここで，hot spot において既に経路上のバッファが一杯になっているとする．図 6はこの状況での経路上のある交換スイッチとその交換スイッチ内のバッファの状態を示す．交換スイッチの出力ポートからバッファ0 にあるパケット $(N-1)_0$が次の交換スイッチに送り出すときに，virtual cut-through ではバッファへのデータの入出力がパイプライン的に行われるので，同時にバッファ0 にパケット N_0を取り込む．これに対して，store and forward ではパケットの転送と取り込みを同時に行えないので次のステップでパケット N_0を取り込む．この交換スイッチではバッファからのパケットの送り出しを交互に行うようになっているので，バッファ0 に保持したパケット N_0が次の交換スイッチに送り出されるのはバッファ1 に取り込まれたパケット $(N-1)_1$を送り出した後である．したがって，virtual cut-through ではパケットを取り込んでから送り出すまで2 ステップ要するが，store and forward では取り込んだパケットを次のステップには送り出すことができる．2 つの転送遅延の差はこのことが

原因であると考えられる．

このシミュレーション結果より，virtual cut-through はパケット生成間隔が十分大きい間はもっとも良い性能を示すが，トラフィックが大きくなると急に性能が悪化することが分かる．それに対して，wormhole は他の2つに比べて交換スイッチ内のバッファが一杯になる前後で性能が安定している．しかしながら，転送遅延を図 2の T_a で見た場合には，wormhole よりも virtual cut-through の方が性能が良い．これらについてはさらに様々な状況のもとでの比較が必要であると考えられる．

5 並列アプリケーションに基づく性能評価

並列アプリケーションの通信パターンを用いたシミュレーションを行うために，ここでは偏微分方程式，レイ・トレーシングおよび行列演算の3種類のアプリケーションを用意した．それぞれのアプリケーションでのトラフィックの様子を図 7に示す．このグラフでは横軸を時間とする一方で，縦軸を生成されているが未だ宛先に到着していないパケットの数とする．

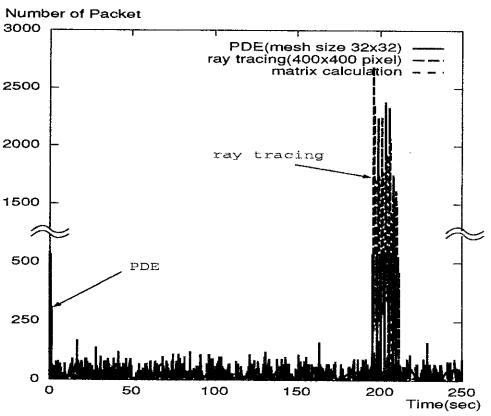


図 7. 並列アプリケーションにおけるトラフィック

5.1 通信パターンの特徴

(1) 偏微分方程式

メッシュサイズを 32×32 とした．プロセッサ要素数は8であるのでメッシュに分割した領域のうち横4列分を各プロセッサ要素に割り当てる．このとき，図 7の時間0付近に見られるように，大量のパケットが各プロセッサ要素に転送される．アルゴリズム的には通信はメッシュに分割した各領域から隣接する上下左右の領域に対して行われるが，この場合隣接する左右の領域は同じプロセッサ要素内に存在するので，実際にはプロセッサ要素をリング上に配置したときに隣合うプロセッサ要素間でのみ行われる．

(2) レイ・トレーシング

ピクセル数を 400×400 とし，プロセッサ要素へのデータの割り当てはラインごとに動的に行う．画像情報は各プロセッサ要素のローカルメモリに存在し，各プロセッサ要素は計算時にローカルメモリから必要な画像情報を取り出す．このとき，各ピクセルの計算は独立に行われるので他のプロセッサ要素との通信を必要としないが，自分がどのピクセルを計算するかを他のプロセッサ要素に通知するための通信が必要となる．この通信は次に処理されるべきデータの情報を管理しているプロセッサ要素に対して送信される．また，計算終了後結果を共有メモリに書き込むために，大量のパケットが1つのプロセッサ要素に対して転送される．

(3) 行列演算

8×8 行列の乗算 A × B → A を 20 回繰り返し実行する．各プロセッサ要素の共有メモリには対応する被乗数の行と乗数の列を割り当てる．計算時には乗数の列データを得るために各プロセッサ要素間で通信が起こる．したがって，通信パターンは一定量のパケットが各プロセッサ要素間を平均的に流れているものになる．

5.2 評価結果

シミュレーション結果を表 2に示す．表の数値は図 2の T_b の算術平均値である．偏微分方程式とレイ・トレーシングの結果を比較すると，レイ・トレーシングの方がやや転送遅延が大きい．これは，通信量的には偏微分方程式の方が大きいにも関わらず，レイ・トレーシングでは処理するデータを確保するための通信が1つのプロセッサ要素に集中する上に，結果の書き戻しのときに1つのプロセッサ要素に大量のパケットが転送されるため，転送遅延の平均値はレイ・トレーシングの方が大きくなっていると考えられる．このとき，他の2つのフロー制御方式と比べて virtual cut-through の転送遅延だけその差が大きくなっている．このことから，レイ・トレーシングの場合の平均転送遅延はちょうど図 5の結果における上昇点程度であると予想できる．

表 2. 並列アプリケーションに基づくシミュレーション結果

	wormhole	virtual cut-through	store and forward
偏微分方程式 (mesh size 32×32)	1160	1343	3274
レイ・トレーシング (400×400 pixel)	1169	1713	3269
行列演算 (8×8 行列乗算)	1176	1240	3312

(結果は図 2の T_bの算術平均値, 単位は nsec)

行列演算の場合, 他の 2 つのアプリケーションと比較して wormhole および store and forward の転送遅延が大きい. これは, 行列の各要素を計算するときに一定量のパケットがプロセッサ要素間を平均的に流れるため, 交換スイッチ内でのパケットの衝突による待ち時間が大きくなっていると考えられる. しかしながら, virtual cut-through だけは他の 2 つのアプリケーションよりも転送遅延が小さい. 図 7からわかるように, 偏微分方程式とレイ・トレーシングの場合にはそれぞれ最初, 最後に大量のパケットが転送される. トラフィックが大きい場合 virtual cut-through の性能は大きく悪化するので, 平均値としては一定量の通信しか行われない行列演算の方が小さくなっていると考えられる.

6 おわりに

本稿では, ハードウェア記述言語を用いてオメガ網のモデル化を行い, フロー制御方式の違いが与える性能への影響を評価するために, 規則的な通信パターンと実際の並列アプリケーションにおける通信パターンの両方を用いてシミュレーションを行った. シミュレーションにハードウェア記述言語を利用することにより, フロー制御方式が性能に与える影響を詳細に把握することができた. 現在, 様々な条件での評価を行うために, 格子網など静的網のモデル化や, 様々な並列アプリケーションを用いたシミュレーションを行っている.

謝辞

並列アプリケーションの通信パターンを DSE を用いて取得していただいた本学情報工学研究科院生の手塚忠則氏, 波内良樹氏ならびに知能情報工学科学部生の岡雅樹氏に感謝の意を表す. また, 本研究に対し日頃から貴重な御助言, 御討論いただく本学情報工学部有田・末吉研究室の諸氏に感謝する. なお, 本研究の一部は文部省科学研究費 (重点領域研究 (1) 課題番号 04235103「超並列ハードウェア・アーキテクチャの研究」) による.

参考文献

[1] William J. Dally : Network and Processor Architecture for Message-Driven Computers, *VLSI and PARALLEL COMPUTATION, Edited by Robert Suaya and Graham Birtwistle*, pp. 140-222(Chapter 3), 1990.

[2] 船越 誠司, 久我 守弘, 末吉 敏則 : ハードウェア記述言語による相互結合網のモデル化と性能評価, 第 45 回情報処理学会全国大会講演論文集 7L-05, 1992 年 10 月.

[3] 柴村 英智, 久我 守弘, 末吉 敏則 : 超並列計算機のための相互結合網シミュレータの開発, 情報処理学会研究報告, 92-ARC-97, pp. 121-128, 1992 年 12 月.

[4] William J. Dally : Performance Analysis of k-ary n-cube Interconnection Networks, *IEEE Transactions on Computers.*, vol. 39, no. 6, pp. 775-785, June 1990.

[5] Kirk L. Johnson : The Impact of Communication Locality on Large-Scale Multiprocessor Performance, *Proceedings of 19th Annual International Symposium on Computer Architecture*, pp. 392-402, May 1992.

[6] Cadence Design Systems, Inc. : Verilog-XL Reference Manual, 1991.

[7] T. Tezuka, K. Ryokai, B. O. Apduhan and T. Sueyoshi : Implementation and Evaluation of a Distributed Supercomputing Environment on a Cluster of Workstations, *Proceedings of 1992 International Conference on Parallel And Distributed Systems*, pp. 58-65, December 1992.

[8] B. O. Apduhan, T. Sueyoshi, Y. Namiuchi, T. Tezuka and I. Arita : Experiments of A Reconfigurable Multiprocessor Simulation on A Distributed Environment, *Proceedings of 1992 IEEE International Phoenix Conference on Computers and Communication*, pp. 539-546, April 1992.

盛光印刷所