2014 年　度

博　　士　　論　　文

バンク型マルチポートメモリを用いた NoC ルータ回路におけるリンク間共有法に関する研究

学籍番号　12T2501

氏　　名　深瀬　尚久

提出月日　2014 年 9 月

指導教員　三浦　康之

湘南工科大学 大学院 博士後期課程 電気情報工学専攻

目次

第1章 緒言

1.1. 背景

近年，計算機の性能は向上しており，かつてはできなかった様々な計算を高速に行えるようになってきている．しかし，物理学のシミュレーションなどのいくつかの分野ではより高速な計算機が絶えず必要とされている．計算処理の中心的な役割を果たすプロセッサ内には実際に処理を行うコアというものがある．コアの高速化は，計算機の高速化の一分野として以前から研究され，実現されてきた．しかし，単一コアによる性能向上は，クロック周波数向上の頭打ちや消費電力，熱量の増加などの問題によって限界が来ている．そのため，近年は複数のコアを使用した並列処理によって処理性能の向上を図っている．また，近年，半導体の集積技術の向上により，単一チップ内に多くの IP コアや回路ブロックを集積することが可能となっている．現在製品化されている例としてはプレイステーション 3 の Cell BE や GeForce8800 などがある．

プロセッサの分野では，intel が 80 個のコアを持つ CPU を試作するなどコア数が増加しており，未来的には数百のコアを持つメニーコアプロセッサなども計画されている．コア数の増加に伴い，図 1-1 のように各コア間の接続方法も以前のようなバス網ではなく，各コアをネットワークによって接続するネットワークオンチップ(NoC)［1］が用いられるようになってきている．

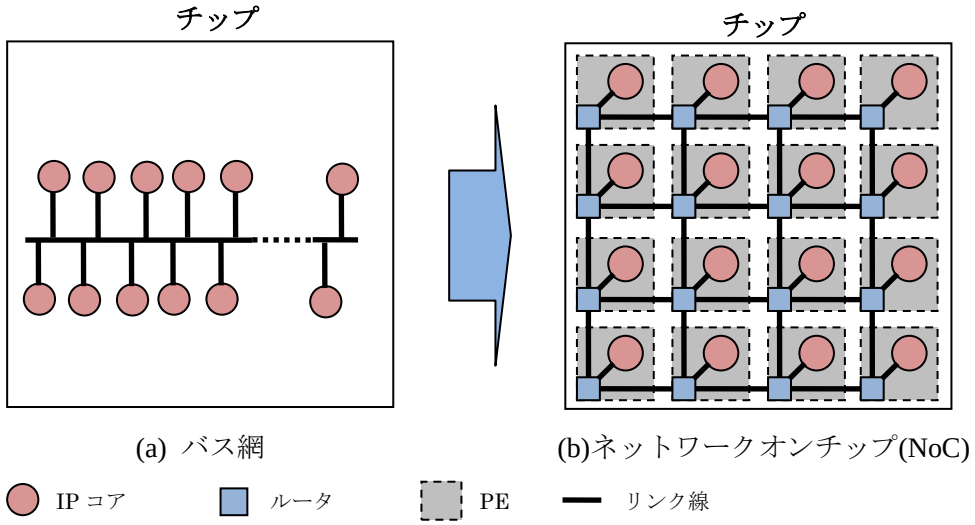


図 1-1 バス網とネットワークオンチップ

直接結合網を用いた NoC は，IP コアとルータから一つの PE が構成され，PE 同士が相互に接続される構造になっている．一般的な LAN などのネットワークなどと異なり，NoC ではルータ間の距離が極端に短いため，ルータの処理時間が通信時間の大部分を占める．また，通信の粒度が小さく頻度も高いため，NoC で使用されるルータは高性能であること

が求められる．一方で，チップ上ではチップ面積や消費電力などに制約があるため，NoC に使用されるルータは，高性能であると同時に底面積，低電力であることも求められる．

図 1-2 のようにルータの内部には，フリットを一時的に格納するバッファが取り付けられている．NoC 全体の性能は，バッファが大容量であるほど高くなる．これは，パケットの進行がブロックされたとき，一つのパケットがまたぐルータ数が減り，パケット同士の衝突が減るためである．しかしながら，バッファの大容量化は面積と消費電力の増大を招くという問題がある．そのため，NoC ルータでは少量のバッファを有効に利用することが重要である．

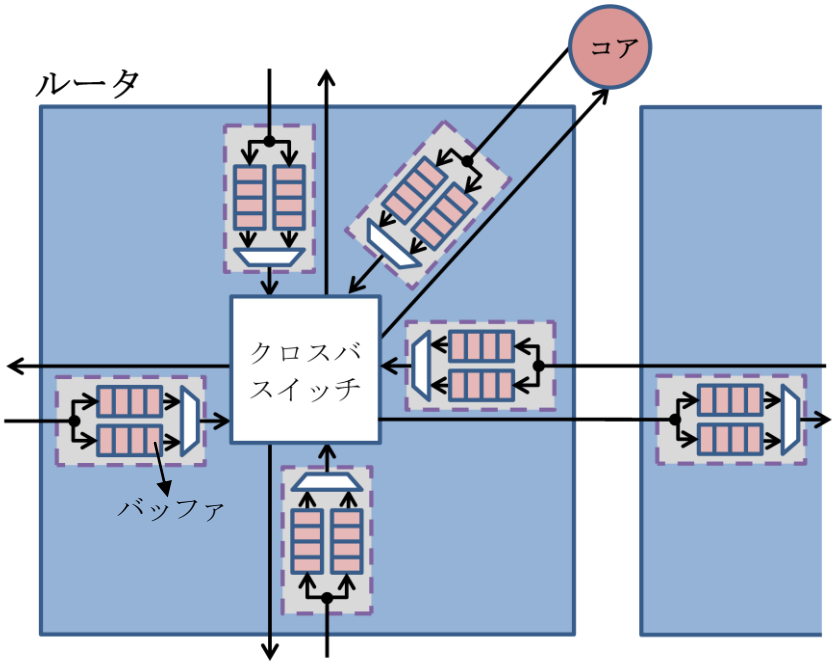


図 1-2 一般的なルータ回路

一般的な NoC ルータでは，各チャネルに個別のバッファが割り当てられている[1][2][3][4][5][6][7][8]．しかし，このような構造の場合，チャネルの未使用時などにそのチャネルに割り当てられたバッファが使用できず，非効率であるという問題がある．この問題を解決するため，従来から各チャネルのバッファを共有するさまざまな手法が提案されている．しかし，複数の物理リンクにまたがるバッファの共有法は，一般的にあまり用いられない．これは，複数の物理リンクから同時に入出力が行われるため，1 ポートのメモリでは同時アクセスの問題が生じるためである．また，この対策としてマルチポートメモリを使用する方法も考えられたが，通常のマルチポートメモリはポート数の 2 乗に比例したハードウェアコストを要するため，NoC の分野での使用は現実的ではなかった．また，ハードウェアコストを抑えた実装法もいくつか提案されているが，それらは，処理の複雑化からパイプラインステージが増加してしまうという問題があった．

1.2. 本論文の目的

ルータ回路における従来のバッファ共有法では，ハードウェアコストやパイプラインの段数の増加という問題がある．近年の LSI は，搭載可能なコア数が増加する傾向にあるためこれらの問題はより大きなものとなると考えられる．この問題を解決するため，本研究では，パイプライン段数の増加なしに，ハードウェアコストの増加を最小限に抑えたバッファの共有手法を提案し，通信性能とハードウェアコストの評価，実装に関する考察などを行う．

1.3. 本論文の構成

図 1-3 に，本論文の構成を示す．本論文は次の通りに構成される．まず 2 章では，NoC の関連知識と関連研究の例を紹介する．3 章では，提案手法の説明と通信性能の評価を行う．4 章で実装やその改善方法などを説明し，ハードウェアコストを評価する．そして，5 章で通信性能を多角的に評価する．6 章では，さらなるハードウェアコスト削減手法として二リンク共有について検討し，最後に 7 章で本研究の結論と今後の予定について述べる．

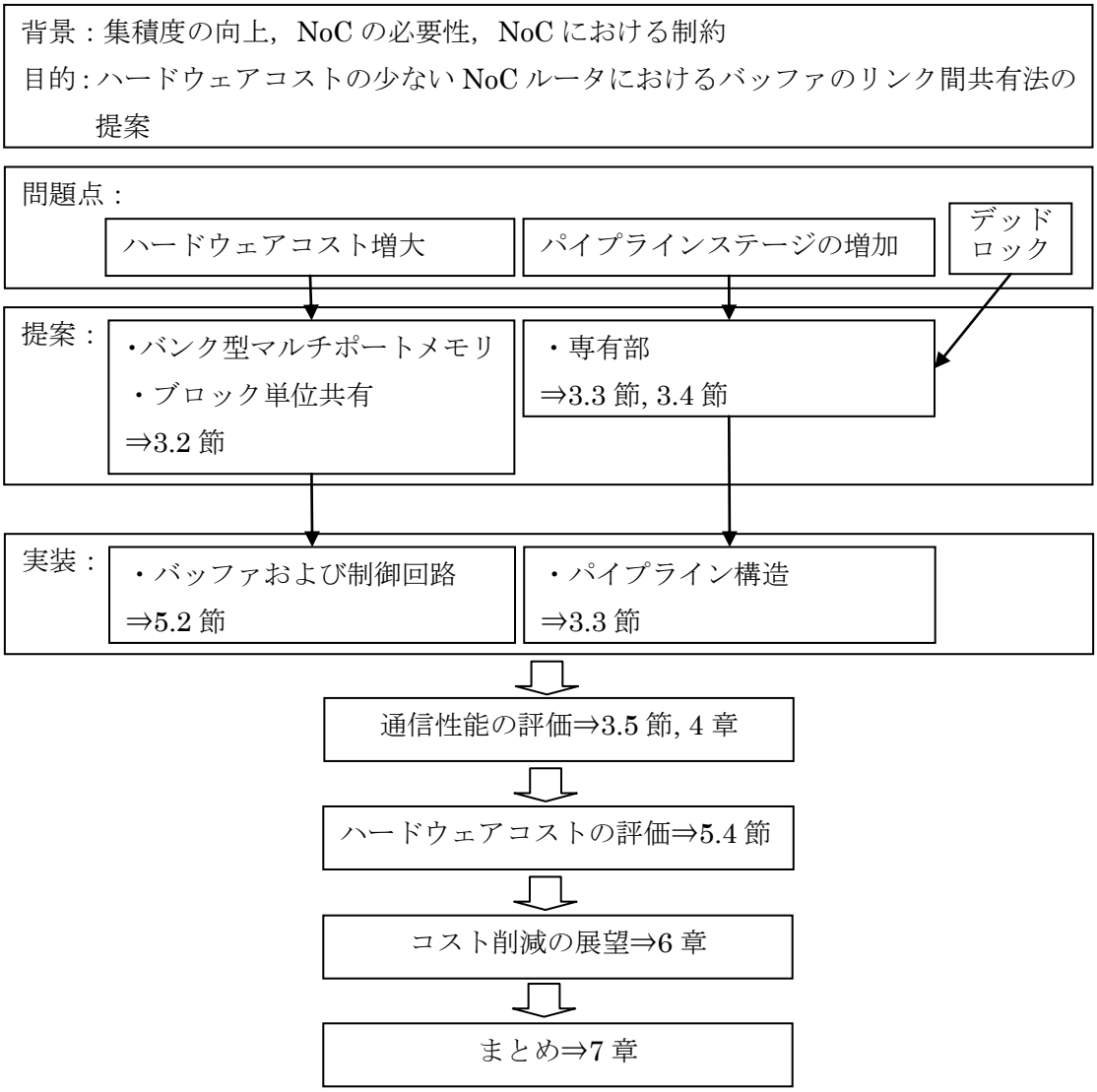

背景：集積度の向上，NoC の必要性，NoC における制約
目的：ハードウェアコストの少ない NoC ルータにおけるバッファのリンク間共有法の提案
問題点：
ハードウェアコスト増大
パイプラインステージの増加
デッドロック
提案：
・バンク型マルチポートメモリ
・ブロック単位共有
⇒3.2 節
・専有部
⇒3.3 節, 3.4 節
実装：
・バッファおよび制御回路
⇒5.2 節
・パイプライン構造
⇒3.3 節
通信性能の評価⇒3.5 節, 4 章
ハードウェアコストの評価⇒5.4 節
コスト削減の展望⇒6 章
まとめ⇒7 章

図 1-3 本論文の構成

第2章 ネットワークオンチップ(NoC)

2.1. はじめに

ネットワークオンチップ(NoC)は, チップ内に実装された IP コアや回路ブロックなどをネットワークで接続し，パケット交換方式で通信を行う手法である．図 2-1 にコア数が 16 の場合の NoC の構成例(メッシュ網)を示す．図 2-1 のように，直接結合網を用いた NoC は，IP コアとルータから一つの PE が構成され, PE 同士が相互に接続される構造になっている．従来，チップ内のコア間の接続にはバス網が主に使用されていた．バスには，構造が単純で低コストであるという利点があるが，チップ内に実装されるコア数の増大により，遅延の増加，配線やタイミング制御などの複雑化などの問題が大きくなった．それに対して，NoC はコア数が増加しても遅延が小さく，配線も単純でスケーラビリティに優れているため，バスに変わるチップ内の接続手法として注目され，様々な応用で実用化されるようになっている．

本章では，NoC に関連した知識と以前に提案されたバッファの共有法について説明する．

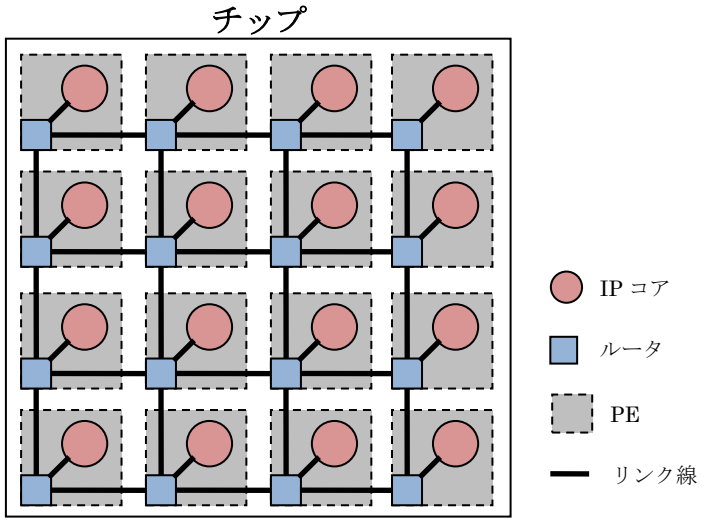


図 2-1 ネットワークオンチップの構成例

2.2. トポロジ

2.2.1. NoC のトポロジ

トポロジとは，IP コア同士がどのように接続されているかを表す用語である．トポロジには，直接結合網と間接結合網がある．直接結合網は，コアとルータから一つの PE が構成され PE 間が相互に接続される．それに対して，間接結合網はコア同士がいくつかのルータを介して接続される構造となっている．本研究では，直接結合網を用いた NoC について述べる．直接結合網の NoC ではメッシュ網とトーラス網が主に使用される．これは，ルーテ

ィングやレイアウトが容易で，コアをタイル状に配置するタイルアーキテクチャに対応しやすいためである．

2.2.2. メッシュ網

図 2-1 で使用されている結合網が 2 次元メッシュ網である．メッシュ網は図 2-1 のように，各 PE がタイル状にはいちされ，端の PE 以外は，上下左右の 4 つの PE と接続され，左端と右端，最上と最下のそれぞれの PE は接続されない構造をしている．レイアウトやルーティングが単純で，配線長が一定で短いため，NoC において良く使用されている．

2.2.3. トーラス網

図 2-2 に 2 次元トーラス網を使用した NoC の例を示す．図 2-2 の赤線のように，2 次元トーラス網は，2 次元メッシュ網の全ての PE に接続される PE の数が等しくなるように端の PE 間を接続したトポロジである．2 次元トーラス網は，同サイズのメッシュ網に比べ，2 倍の帯域を持ち平均ホップ数も少ないという利点がある．しかしその反面，各次元をリング状に接続するため循環依存によりデッドロックが発生するほか，配線長が長くなるという問題もある．

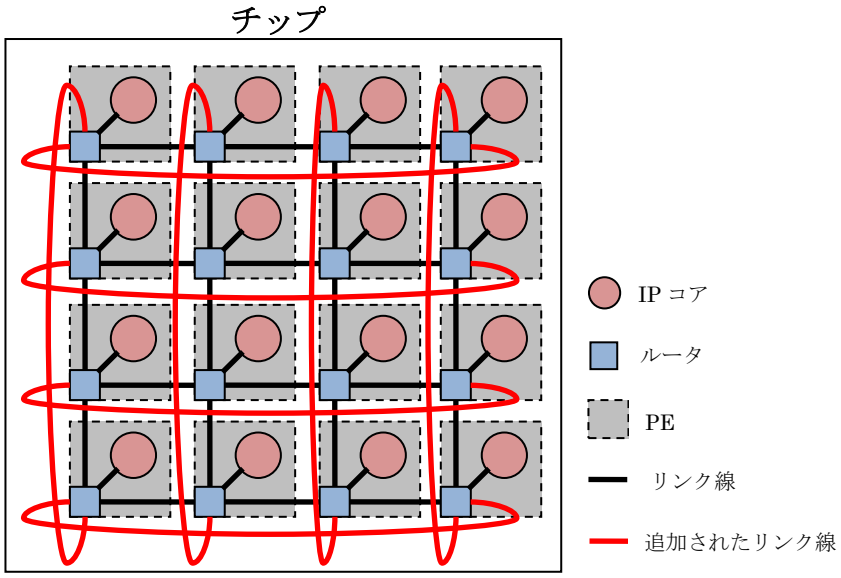


図 2-2　トーラス網を使用した NoC

2.3. ルーティングアルゴリズム

2.3.1. NoC のルーティングアルゴリズム

ルーティングアルゴリズムとは，与えられたトポロジにおいてパケットを目的の PE に送る経路を決定する決まりである．ルーティングアルゴリズムには，特定の PE から特定の PE への経路が一定である固定型ルーティングと，ネットワークの状態によって，動的に経

路を変更する適応型ルーティングがある．NoC では，制御が簡単で実装コストが低い固定型ルーティングが主に使用されるが，ターンモデルなどを用いた適応型ルーティング等についての研究もおこなわれている．本研究では，固定型ルーティングの一種である次元順ルーティング[9]を用いる．

2.3.2. 次元順ルーティング

次元順ルーティングは，固定型ルーティングの一種で次元順にパケット転送を行う手法である．2 次元メッシュ網での例を図 2-3 に示す．図 2-3 に示すように，送信元(図 2-3 では PE0)で生成されたパケットは最初に X 次元(横軸)方向を移動し，送信先(図 2-3 では PE 14)と同じ列にある PE(図 2-3 では PE2)まで移動する(図 2-3 ①)．その後 Y 次元(縦軸)方向に移動し送信先の PE に到達する(図 2-3 ②)．X 次元，Y 次元の順番は逆の場合もある．構造が単純で，ハードウェア量も少ないことからメッシュ網やトーラス網を用いた NoC で広く使用されている．ただし，トーラス網においては，各次元方向に循環依存が発生するので，デッドロックを回避するために 2 本以上の仮想チャネルを必要とする．

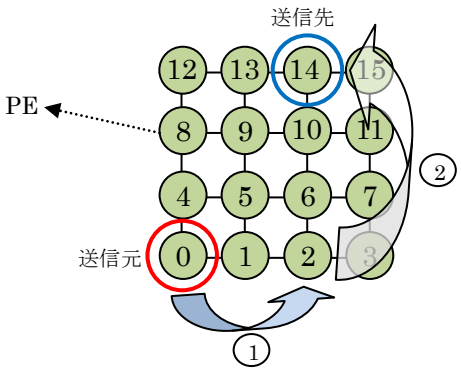


図 2-3　2 次元メッシュ網での次元順ルーティング

2.4. データ交換方式

2.4.1. パケット交換方式

パケット交換方式は，送信データをパケットという単位に分割し，通信を行う方式である．各パケットには，送信先などの情報(ヘッダー)が付加され，通信路上のルータがその情報とルーティングアルゴリズムによってパケットを中継することにより，パケットを目的地に転送する．この方式では，複数のパケットが回線を時分割で共有できるため，次のような場合に有利である．

- 通信の頻度が多い
- 送信データのサイズ(粒度)が小さい
- PE 数が多い

・送信先が頻繁に変更される

データの交換方式には，パケット交換方式などのほかに送受信を行うノードの間で回線を占有する回線交換方式などがあるが，近年のチップ内のようにノード数が多い環境では，実装面積や遅延の関係で使用はされなくなってきている．

パケット交換方式には大きく分けて，ストアアンドフォワード方式，ワームホール方式[6][10]，バーチャルカットスルー方式の 3 種がある．NoC では，低ハードウェアコスト，高スループットであることが求められるため，主にワームホール方式が使用される．以下にパケットの構造と各方式について述べる．

2.4.2. パケットの構造

パケットの構造を図 2-4 に示す. 図 2-4 のようにパケットは, 通信に使用するヘッダ部と，送信データであるボディ部から構成されている．ヘッダには，送信先のアドレスや送信元のアドレス，パケット長などが格納される．また，図 2-4 のようにパケットは，より細かなフリットという単位に分割できる．フリットは，1 サイクルで並列転送可能なデータの単位で，サイズは 8bit～256bit 程度である．

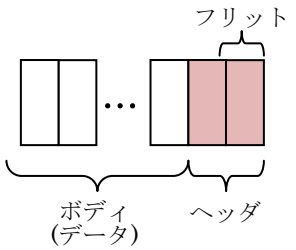


図 2-4　パケットの構造

2.4.3. ストアアンドフォワード方式

図 2-5 にストアアンドフォワード方式の動作を示す．図 2-5 のように，ストアアンドフォワード方式では，各ルータがパケット全体を格納できるバッファを持ち，各ルータはパケット全体を格納するまで次のノードへの送信を行わず(図 2-5(2)～(4)), 全てのパケットを受信してから送信を行う．(図 2-5(5)).

バッファのサイズが大きいためハードウェアコストが大きくなり，通信遅延も大きいことから，NoC ではほとんど使用されない方式である．しかし，パケット全体を保存できるため，誤り訂正などの処理や入出力側の回線速度が異なる場合への対応などが可能であるため，インターネットなどでは広く使用されている．

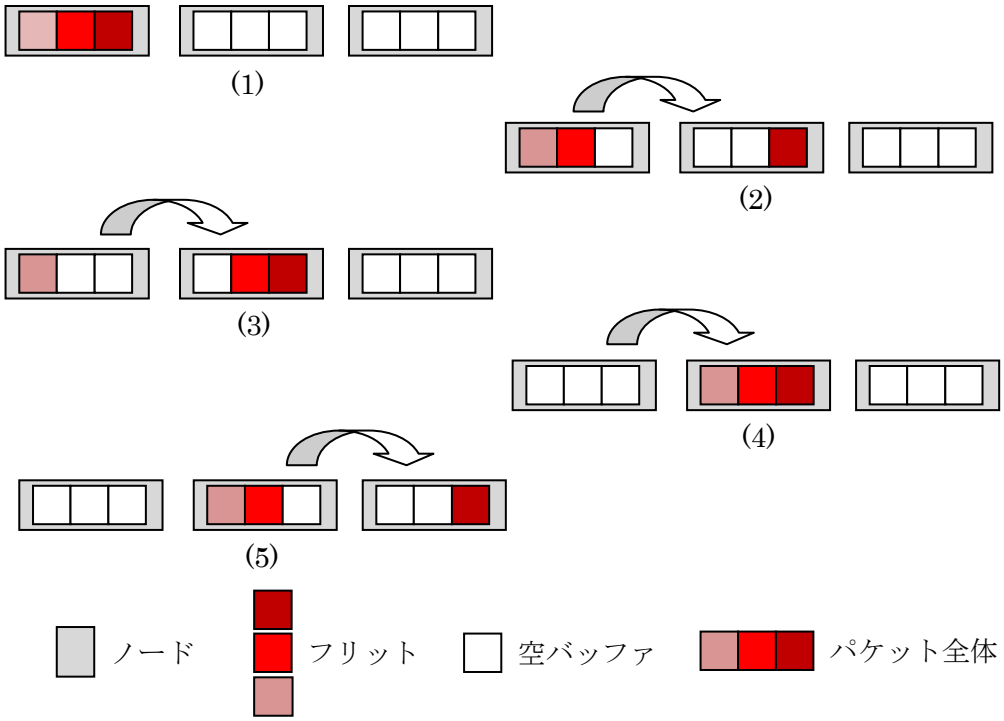


図 2-5　ストアアンドフォワード方式の動作

2.4.4. ワームホール方式

ワームホール方式の動作を図 2-6 に示す．図 2-6 のように，ワームホール方式は各ノードがパケット長よりも小さなバッファをもち，フリットを受信するたびにそのフリットを次のノードに転送する．

パケット全体を保持するバッファが必要ないため，他の方式に比べ少ないハードウェアコストで実装でき，パケット長が十分に長ければ転送時間がノード数に影響されず高速であるため，NoC では主にこの方式が使用される．

ただし，パケットが複数のノードにまたがって存在するため，パケット同士の衝突が起きやすく，後述するデッドロックやヘッドオブラインブロッキング等が起きやいという問題もある．

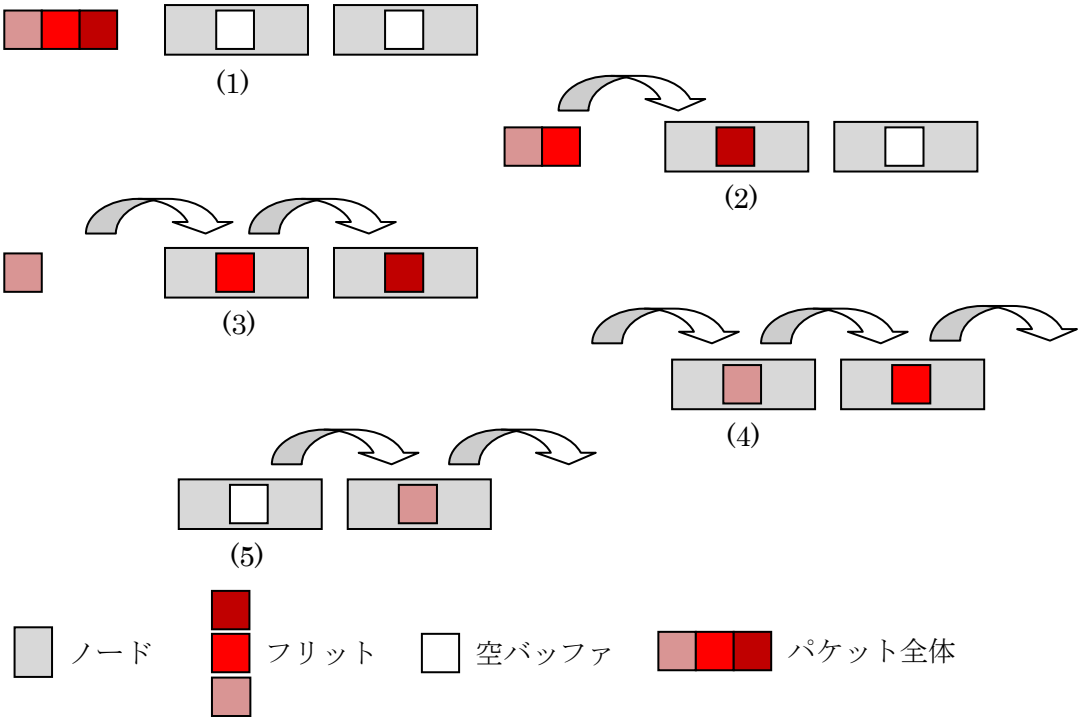


図 2-6　ワームホール方式の動作

2.4.5. バーチャルカットスルー方式

ワームホール方式と同様に，フリットを受信するたびにそのフリットをでき，ストアアンドフォワード方式と同じく，パケット全体を保持するバッファを持つ方式である．各ルータがパケット全体を保持するバッファを持つため，パケットの進行がブロックされた場合でも，後続フリットが先頭フリットのあるルータのバッファにすべて格納できる．これにより，ワームホール方式のように進行のブロック時に複数のルータをまたいでバッファを専有することがなくなり，パケット同士の衝突を軽減することができる．ストアアンドフォワードと同様にバッファの容量が膨大になるため，NoC で使用されることはほとんどない方式である．しかし，遅延が小さいため，ハードウェアコストの制約が小さい場合などで使用される．

2.5. デッドロックとヘッドオブラインブロッキング

2.5.1. デッドロック

パケット交換方式において，各パケットはリソース(資源)としてバッファとチャネルを取得し，使用する．この時，あるパケット 1 が取得しようとしたリソースがほかのパケット 2 に使用されていた場合，そのリソースを取得しているパケット 2 がそれを解放するまでパ

ケット 1 は待機する．デッドロック[11]とは，この状態が複数のパケットで循環するようにおき，それ以上転送を行うことができなくなる状態である．図 2-7 にデッドロックの例を示す．図 2-7 では，パケット 1 がパケット 2 の取得しているリソースを待ち，パケット 2 がパケット 3 の取得しているリソースを待ち，パケット 3 がパケット 1 の取得しているリソースを待っており，これ以上転送が行えなくなっている．

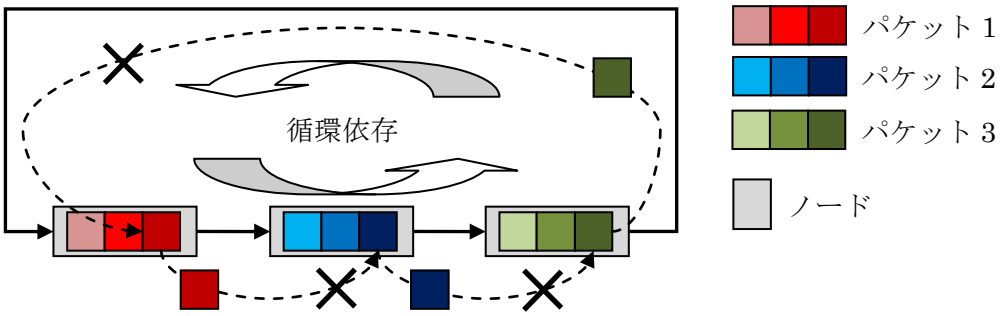


図 2-7　デッドロックの例

2.5.2. デッドロックの回避

パケット通信におけるデッドロックの回避法には，主に次の 2 つがある．

一つは，デッドロックの発生を検知し，デッドロックを起こしたパケットを破棄，再送させる．あるいは，パケットを破棄せずに，リソースを割り当てなおす方法である．デッドロックの検知には，タイムアウト機構などが使用される．デッドロックの発生率が低いときは有効であるが，高い場合には通信性能が大きく低下するという問題がある．そのため，デッドロックの発生率が高いワームホール方式を用いることが多く，遅延に対する要求が厳しい NoC ではほとんど使用されない．

2 つ目は，デッドロックの原因である図 2-7 のような循環(循環依存)を，ルーティングの工夫やリソース(チャネルやバッファ)追加によって断ちきる方法である．ルーティングの工夫による方法には turn モデルなどがある．図 2-8 に turn モデルの概念を示す．図 2-8(1)のような循環依存が経路上に存在する場合，デッドロックが発生するため，turn モデルでは図 2-8(2)の点線のように特定方向への曲がりを禁止することで循環依存が起こらないようにしている．リソースを追加する方法では，各ノードが持つチャネルとバッファを増やし，それらをうまく使用することによってデッドロックを回避できる．図 2-9 にリソースを追加した際のデッドロック回避の例を示す．図 2-9 では，各ノードが持つチャネルの数を一つ増やし，通常は上のチャネルを，右端のノードを通ったパケットは下のチャネルを使用することで循環依存が起こらないようにしている．

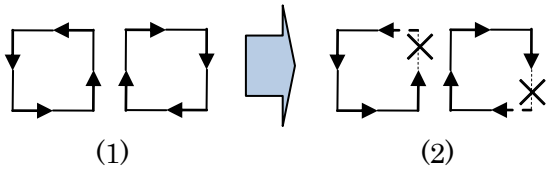


図 2-8　turn モデル

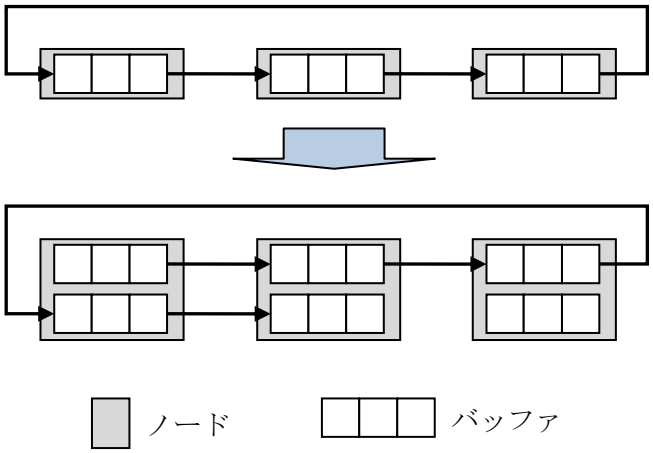


図 2-9　デッドロック回避例

2.5.3. チャネル依存グラフ

チャネル依存グラフは，ネットワーク内の各チャネル間の関係を表したグラフで，デッドロックの確認や照明のために使用される．このグラフ作成は，まず各チャネルに番号付けを行いノードとする．そして，各チャネルの後に選ばれる可能性能あるチャネルに有効枝をつないでいくことで作成できる．例として図 2-10 に，図 2-9 のリソース追加前と追加後のネットワークのチャネル依存グラフを示す．図 2-10 より，リソース追加前のチャネル依存グラフ(図 2-10(1))には循環依存が存在し，デッドロックが発生することが分かる．対して，リソース追加後のチャネル依存グラフ(図 2-10(2))には，循環依存が存在しておらず，デッドロックが発生しないことが確認出来る．

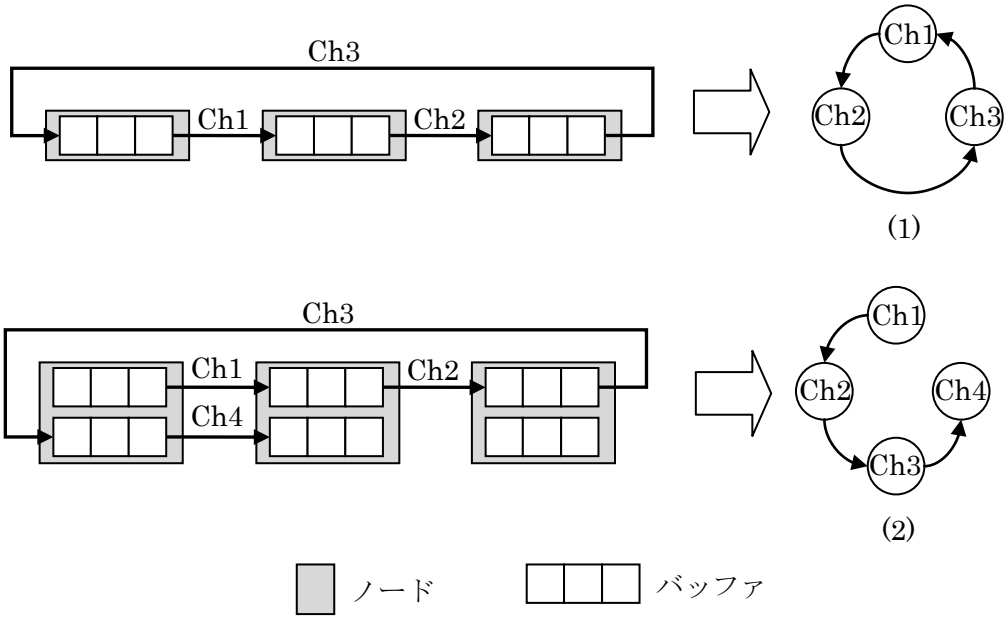


図 2-10　チャネル依存グラフの例

2.5.4. ヘッドオブラインブロッキング

ヘッドオブラインブロッキングの例を図 2-11 に示す．図 2-11 では，パケット 1 によってパケット 2 の進行がブロックされ，パケット 3 は目的地がノード D であるにもかかわらずパケット 1 の転送が終了し，その後のパケット 2 がノード C に転送されるまで転送を行えない状態になっている．このように，ヘッドオブラインブロッキングとは，あるパケット(図 2-11 パケット 1)がブロックされることによって，後続のほかのパケット(図 2-11 パケット 2)もブロックされてしまい，本来ならば最初に停止したパケットを待たなくてもよいパケット(図 2-11 パケット 3)も待たなければならなくなる状態である．この現象は，パケット同士の衝突率の高いワームホール方式で特におこりやすく，性能を低下させる要因となっている．

ヘッドオブラインブロッキングによる性能低下を軽減するには，デッドロック対策で述べたリソースの追加が有効である．図 2-12 に図 2-11 のノード B のリソースを 2 つにした例を示す．図 2-12 では，パケット 2 の通信がブロックされていることに変化はないが，ノード B のリソースが二つあるため，パケット 3 はパケット 1，2 の送信を待たず，ノード D に送信できる．

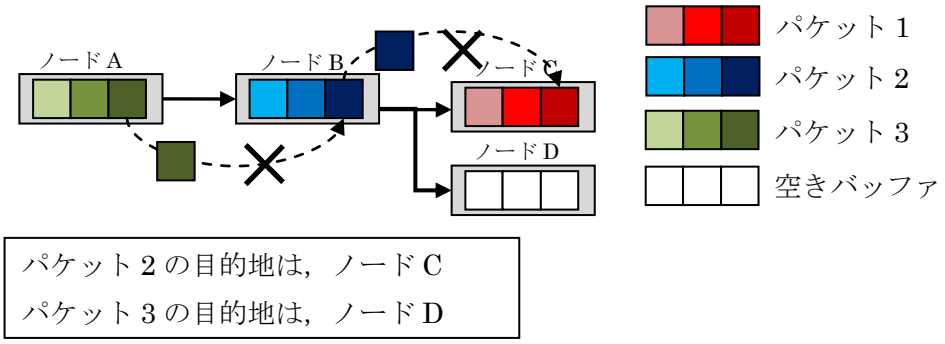


図 2-11　ヘッドオブラインブロッキングの例

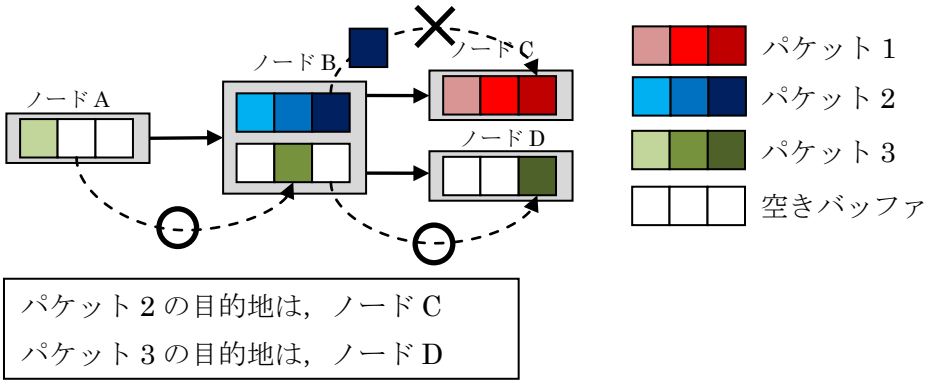


図 2-12　ヘッドオブラインブロッキングの軽減例

2.5.5. 仮想チャネル

デッドロックの回避やヘッドオブラインブロッキングの軽減などで使用したリソースの追加において，物理的にチャネル(リンクとバッファの組)の数を増やすことはハードウェアの増加を招くこととなる．そのため，実際には一つのリンクに複数のバッファを設置することで仮想的に複数のチャネルがあるように見せかける仮想チャネルが使用される[12]．仮想チャネルの例を図 2-13 に示す．図 2-13 のデマルチプレクサは，一つの入力と複数の出力を持ち，制御信号によって出力を選択する回路である．そして，マルチプレクサは，複数の入力と一つの出力を持ち，制御信号によって入力を選択する回路である．仮想チャネルはどの手法にも使用できるが，仮想チャネルごとにバッファを用意しなければならないため，各仮想チャネルに追加するバッファの容量が他の方式に比べ少ないワームホール方式に有効な方法である．

基本的に，仮想チャネルの数が多いほどパケット同士の衝突確率が少なくなり，リンクの利用効率は向上する．しかし，一つのリンクを時分割で使用するため，仮想チャネルが多いほどレイテンシが下がる可能性があり，利用効率もある程度仮想チャネルが増えると上がりにくくなる．

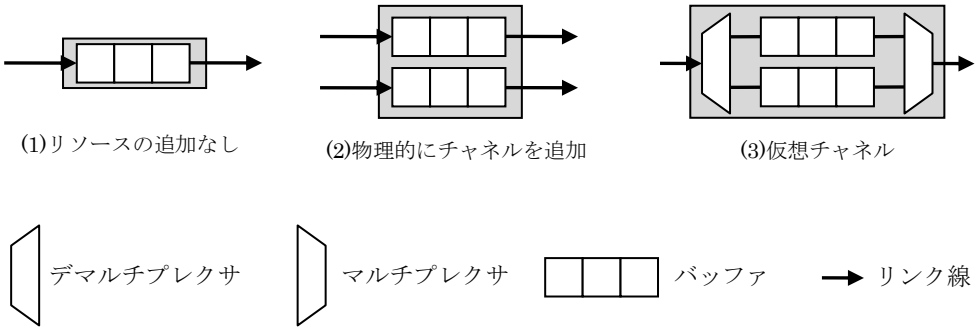


図 2-13　仮想チャネル

2.6. ルータ回路

2.6.1. NoC のルータ回路

ルータ回路は，通信に関する処理を行う回路で，入力ポートに入力されたパケット(フリット)をヘッダの情報とルーティングアルゴリズムに従って，適切な出力ポートに出力する．NoC において，ルータの性能はシステム全体の性能に大きく影響する．これは，ルータ間の距離が極端に短いため，通信時間におけるルータの処理時間の割合が大きいためである．また，チップ内の通信では通信の粒度が小さく，頻度も高いことから，NoC のルータはより高性能であることが求められる．一方で，チップ上ではチップ面積や消費電力などに制約があるため，NoC に使用されるルータは，高性能であると同時に底面積，低電力であることも求められる．これらの制約のため NoC では前述したとおり，ルータのサイズが小さく高速なワームホール方式が使用される．

2.6.2. ワームホールルータ

図 2-14 に一般的なワームホールルータのアーキテクチャを示す．また，仮想チャネルを使用する場合の各リンクの構造を図 2-15 に示す．図 2-14 の制御回路には，経路を計算するロジックやクロスバスイッチを制御するアービタ，仮想チャネルを使用する場合は仮想チャネルを割り当てるアロケータなどが含まれる．

動作として，まず入力ポートから入力されたフリットは入力側のバッファに格納される．そして，ルート計算などの処理後に適切な出力チャネルに転送される．

図 2-14 では，入出力側の両方にバッファが設置されているが，入出力側の一方にバッファを設置する実装もある．

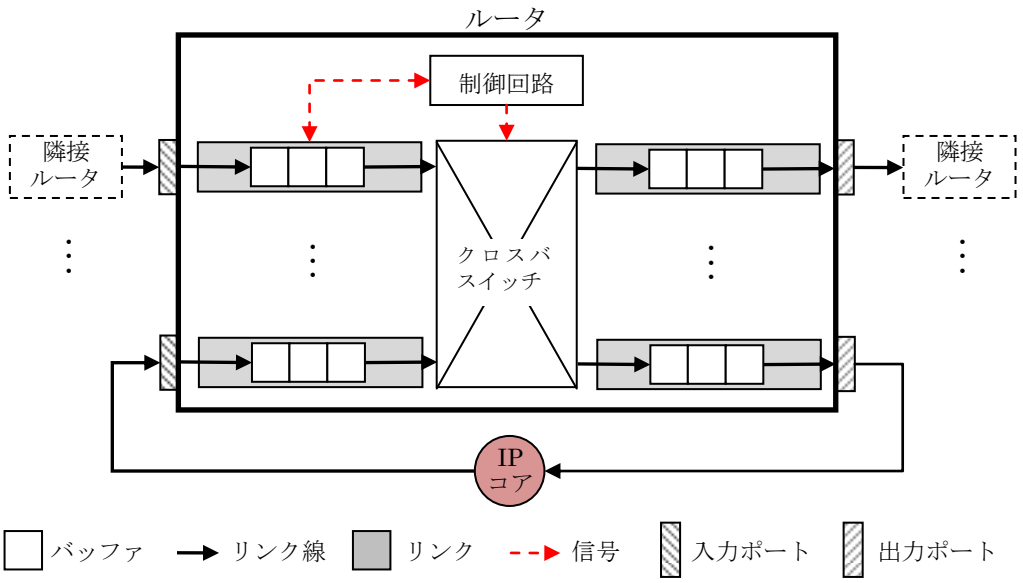


図 2-14　ワームホールルータのアーキテクチャ

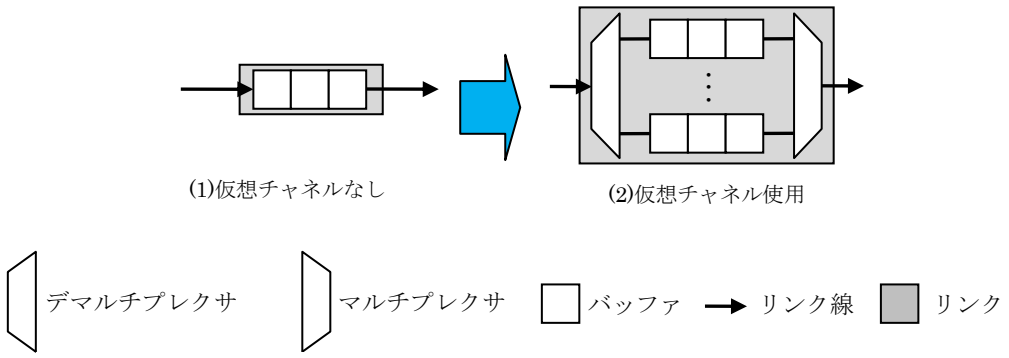


図 2-15　仮想チャネル実装時のリンク構造

2.6.3. バッファ

図 2-14 のようにルータの内部には，バッファが取り付けられている．これは，ルート計算などの処理や通信のブロック時にフリットを一時的に格納しなければならないためである．ルータ内のバッファは FIFO で構成されており，先に入力されたフリットが先に出力されるようになっている．

NoC 全体の性能は，バッファが大容量であるほど高くなる．これは，パケットの進行がブロックされたとき，一つのパケットがまたぐルータ数が減り，パケット同士の衝突が減るためである．しかしながら，バッファの大容量化は面積と消費電力の増大を招くという問題がある．そのため，NoC ルータでは少量のバッファを有効に利用することが重要である．

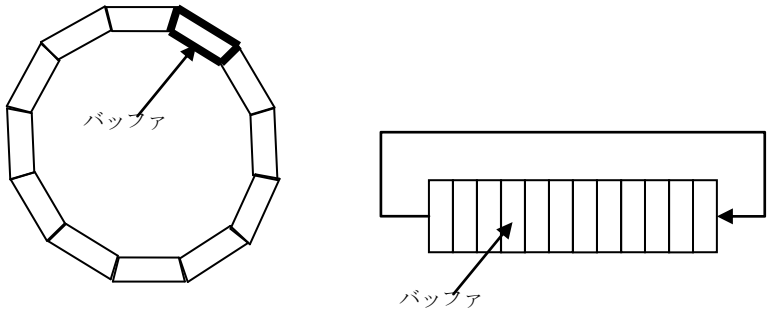
2.6.4. リングバッファ型 FIFO

ルータのバッファに使用される FIFO は，リングバッファ型という法式である．リングバッファ型の FIFO とは, データを記録する RAM の部分がリングバッファで構成されている FIFO のことである．リングバッファとは，図 2-16(a)のようにリング状に配置されたバッファである．ただし，実際にバッファをリング状に配置する事はできないので，実際には図 2-16(b)のように直線状に配置したバッファの両端を論理的につなげる事によってリングバッファを構成している．

FIFO を構成するにはバッファ本体である RAM と入力場所を示す edp と出力場所を示す stp という 2 種類のポインタ, 空きのあるなしを表すビット empty, full が必要になる. FIFO 各部にかかるビット数を図 2-17 に，FIFO の動作について図 2-18 に示す.

FIFO のサイズを S，RAM の記憶領域ひとつのワード長を W(bits)とすると図 2-17 のように RAM 本体は(S×W)bit，edp，stp は RAM の容量分の領域を識別すればよいので(log(S))bits となる．empty，full はそれぞれ「空かそうでないか」,「満タンかそうでないか」を表現できればよいため 1bit となる．

図 2-18(1)は空の状態を表しており図にはないが full=0, empty=1 となっている. 図 2-18(1)の状態でデータが入力されると edp が指している RAM の領域にデータが入力される．その後 edp が+1 され，空でなくなるため empty が 0 となる(図 2-18(2))．この時，edp の値が RAM の最上位のアドレス(図 2-18 の RAM における一番上のアドレス)を越えていた場合は，RAM の最下位のアドレス(図 2-18 の RAM における一番下のアドレス)を指すようにする．図 2-18(2)以降も入力が続き，図 2-18(3)のように edp = stp になった時に入力が行われると，RAM は満タンになり full が 1 となる(図 2-18(4))．図 2-18(4)の状態で，出力が行われると stp が＋1 され，空きができるため full が 0 となる．その後，stp が指している RAM の領域のデータが出力され，図 2-18(5)の状態になる．この時，stp の値が RAM の最上位のアドレスを越えていた場合，RAM の最下位のアドレスを指すようにする．図 2-18(5)以降も出力が続き，図 2-18(6)のように stp = edp－2 になった時出力が行われると RAM は空となり，empty が 1 になる(図 2-18(1)).



(a)理論上のリングバッファ　　(b)実際リングバッファ

図 2-16　リングバッファ

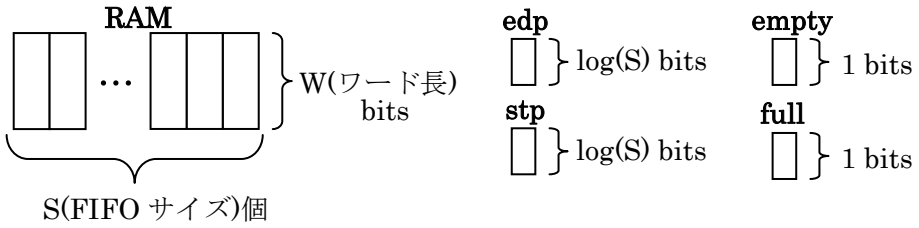


図 2-17　FIFO の各部ビット数

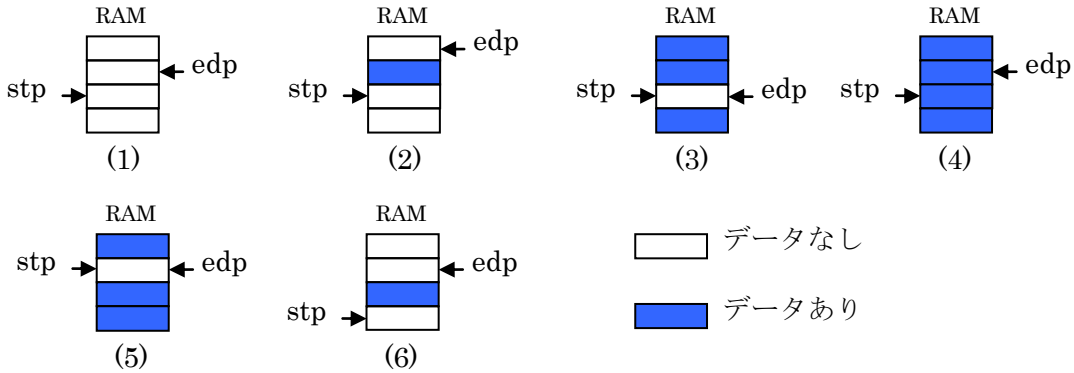


図 2-18　FIFO の動作

2.6.5. クロスバスイッチ

図 2-19 にクロスバスイッチの構成を示す．図 2-19 のようにクロスバスイッチとは縦方向と横方向の通信路を用意し，それを格子状に配置したものである．その交点にはクロスポイントというスイッチが取り付けられており，それを制御することによって，垂直方向の通信路同士を動的に接続する．

クロスバスイッチは，入出力が交点一つで接続され，複数の入力が同時に同じ出力にアクセスしなければ衝突が起こらないためるため遅延が小さいという利点がある．しかし，入力数を n，出力数を m とするとサイズが m×n となり，入出力の数が多い場合はハードウェアコストが大きくなるという問題もある．

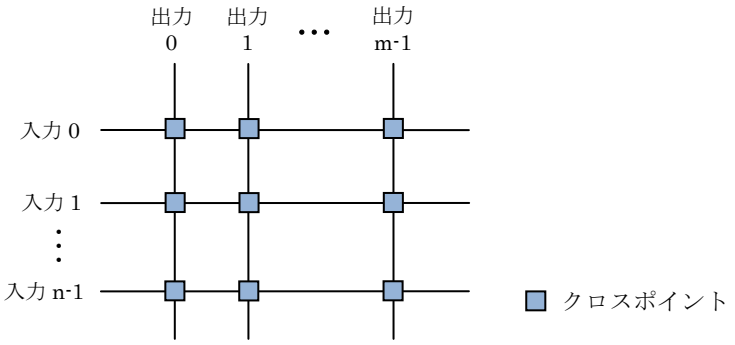


図 2-19　クロスバスイッチ

2.7. 関連研究

2.7.1. NoC ルータの研究

前述したとおり NoC ルータは高い通信性能を有し，少ない実装面積，低消費電力で実装可能であることが求められる．直接結合網を用いた NoC は IP コアとルータで一つの PE を構成するため，IP コア数が増加するとともにルータの数も増加する．そのため，今後予想される IP コア数の増加により，ルータに求められる各要求はより厳しいものになると考えられる．このような理由から NoC ルータは，様々な手法が提案されている．たとえば，消費電力の分野では，リーク電流を抑えるために使用していない部位の電力供給を遮断する手法などが提案されている．通信性能の向上手法としては，ルータ内で経路予測を行い，処理を省略することで遅延を削減する手法やバッファの共有などにより少ないバッファ容量で多くのフリットの格納を可能にする手法[13][14][15]などが提案されている．次節以降は，提案手法と類似した手法として後者の手法を紹介する．

2.7.2. 仮想チャネル間の共有

図 2-20 に，通常の仮想チャネルを用いたリンクとチャネル間の共有を行った場合のリンクの構成を示す．前述したように，各リンクにはデッドロックの回避やヘッドオブラインブロッキングの軽減などのために，仮想チャネルが用いられることが多い．このような構造では，図 2-20(a)のように各仮想チャネルに個別のバッファを取り付けることが普通である．しかしこのような構造は，チャネルが使用されていない場合にそのチャネルに割り当てられたバッファが活用されないという問題がある．そのことから，バッファを効率的に使用するため，各物理リンクにおける複数の仮想チャネルで 1 個のバッファを共有して使用する方法が提案されている[13][14][15] (図 2-20(b))．この手法では，フリット入力時に物理リンクごとに設けられた共有メモリから 1 フリット分のメモリ領域を動的に割り当て，その割り当てられた領域にフリットを格納する．その際， 図 2-20(b)にあるように割り当てられたメモリの制御を行うため，各リンクにコントローラが必要になる．

図 2-21 に従来法の効果例を示す．図 2-21 のようにこの手法ではバッファの容量を増やした場合と同様に，パケットの進行がブロックされた時に一つのパケットがまたぐルータが減ることでパケット同士の衝突率が下がり，通信性能が向上する．図 2-21(a)では，パケット 1 が PE1 でブロックされ，PE1 と PE2 にまたがって存在している．これにより，PE2 の上のチャネルを取得しようとしたパケット 2 が待たされている．対して，図 2-21(b)では，バッファの共有によりもう片方のチャネル分のバッファを使用できるため，PE1 のみにパケット 1 が存在している．これにより，パケット 2 は PE2 の上のチャネルを取得でき，通信を続けることができる．

図 2-20　チャネル間共有のリンク構造

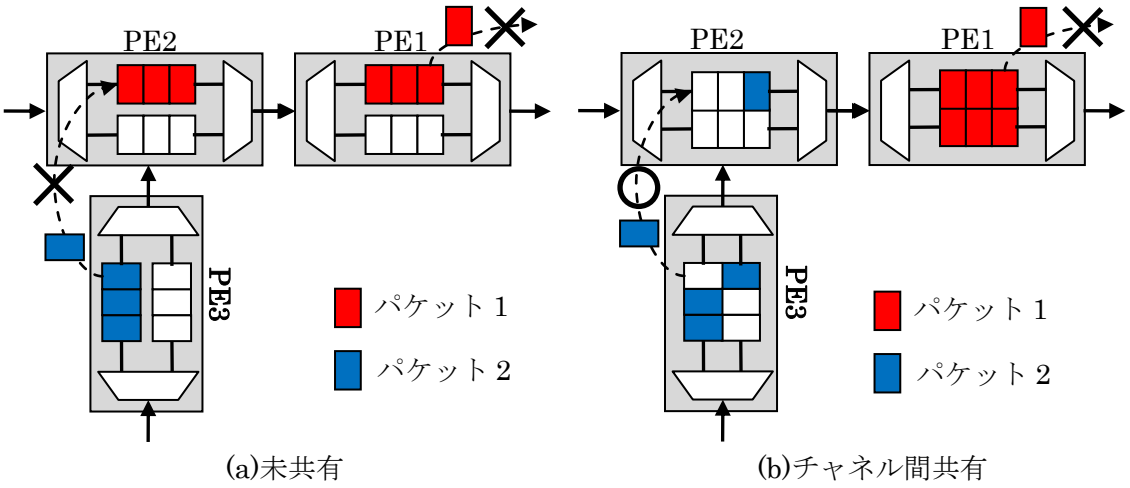


図 2-21　バッファ共有の効果

図 2-20(b)にあるように，この手法では共有メモリを管理するため，各リンクにコントローラが必要になる．従来法の全体像と構造を図 2-22，2-23 に示す．図 2-23 の Free_Pool は，どのチャネルにも取得されていない空きメモリ領域のポインタを格納する FIFO である．Block_Info は各パケットが取得したメモリ領域のポインタを格納するものである．

本論文では説明を簡単にするため以後，従来法のような物理リンク内の仮想チャネルで一つのバッファを使用する手法を「チャネル間共有」，フリット入力時にフリット一つ分のメモリ領域を割り当てる手法を「フリット単位共有」，両方を行うものを「従来法」と呼ぶこととする．また，Block_Info と Free_Pool を合わせて「制御用 FIFO」と呼ぶこととする．

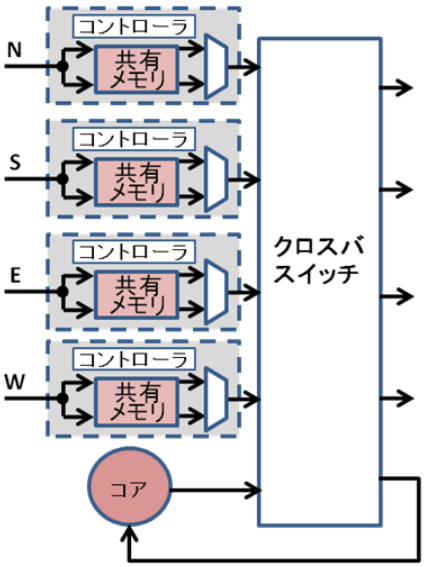


図 2-22　従来法の全体像

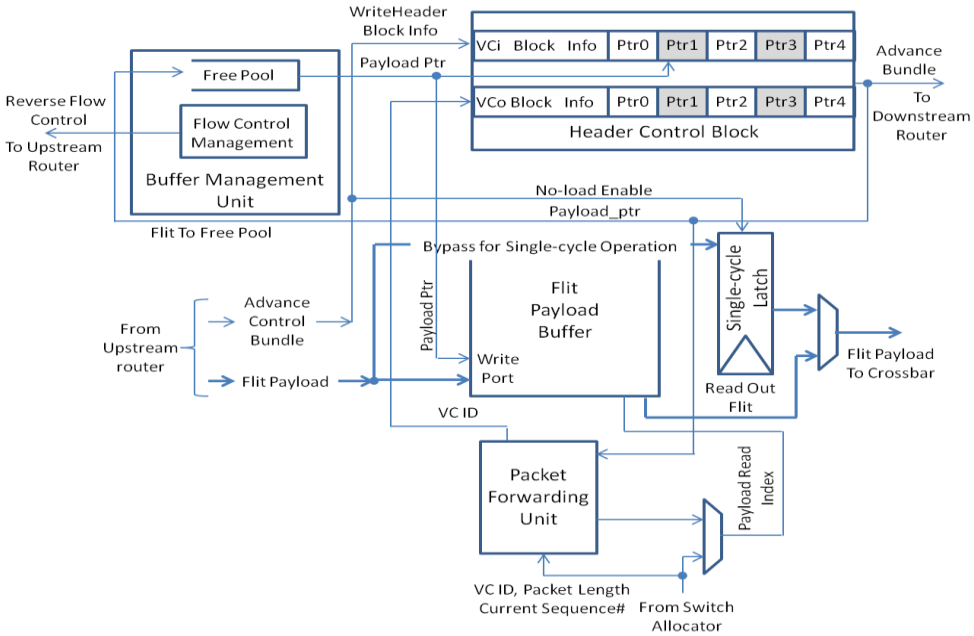


図 2-23　従来法の構造

2.7.3. CBDA(centrally-buffed,dynamically-Allocated)方式

提案手法と同様に物理リンク間でバッファを共有する方式に CBDA 方式[15]がある．この手法は理想状態を示す指標として紹介されたが，実装向きではないためこれまで用いられることはあまりなかった．それは，下記の理由による．

・1 個の物理リンク内の複数の仮想チャネルにまたがる共有では，同時に入出力が行われるフリットの数は 1 個ずつであるため，1 ポートのメモリ 1 個を複数のチャネルで

共有することができる. これに対して, 複数の物理リンクにまたがる共有を行う場合, 各物理リンクが同時にフリットの入出力を行うため，１ポートのメモリでは同時アクセスの問題が生じる.

・上記の問題の解決のために，マルチポートのメモリを用いる方法が考えられるが, 通常のマルチポートメモリを使用した場合，ポート数の２乗に比例したハードウェアコストを要するため，ハードウェアコストが膨大なものとなる.

2.7.4. Distributed Shared-Buffer Router

図 2-24 に DSB ルータの構造を示す[16]. 図 2-24(a)にあるように DSB ルータは，バッファが二つのクロスバスイッチの間に配置される構造となっている．これにより，バッファの利用効率が向上し，スループットを改善することに成功している．一方で図 2-24(b)にあるように，処理数の増加によりパイプラインステージが増加してしまい，レイテンシ自体は低下してしまうという問題もある.

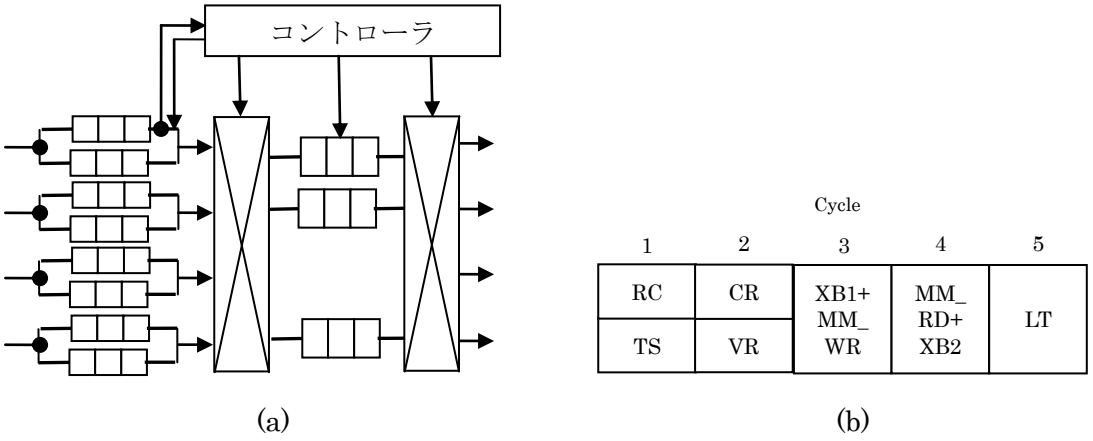


図 2-24　DSB ルータの構造とパイプライン構造

2.7.5. リングバッファを用いた共有法

提案手法と類似した手法にリングバッファを用いたものがある[17]. 図 2-25 にリングバッファを用いた手法の構造を示す．この手法も DSB ルータと同様に，バッファの利用効率を向上し，スループットを改善することに成功している．しかし，ハードウェアコストとレイテンシ増加が問題となる.

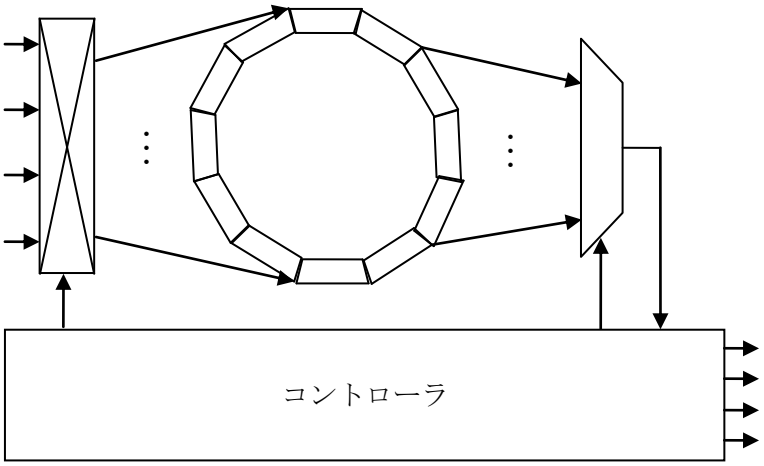


図 2-25　DSB ルータの構造とパイプライン構造

2.8. まとめ

本章では, NoC やルータ技術に関係する知識や, 一般的ルータや従来法について紹介した. 一般的なルータには，クロスバスイッチの入力側の各チャネルに等量のバッファが取り付けられており，バッファの容量が大きいほどネットワークが混雑に強くなることを述べた. また，NoC ルータではバッファを効率的に使用することが重要で，一般的ルータではバッファの使用効率が良くないことも説明した.

ルータ内のバッファを効率的に使用するために以前に研究されていた関連研究について紹介した. そして，関連研究の概要と問題点について簡単に説明した.

3 章では，これらの技術を元に我々が提案するリンク間共有法について説明する.

第3章 提案手法：リンク間共有法

3.1. はじめに

2 章で述べたとおり，NoC ルータにおいて少量のバッファを有効に使用することは，高い通信性能と低ハードウェアコストを両立するうえで重要である．バッファを有効利用するため従来、物理リンク内の仮想チャネル間や全物理リンク間でバッファを共有する手法が提案されている。その概念図を図 3-1 に示す。図 3-1(a)は共有を行わない一般的な手法、(b)は物理リンク内で共有を行う手法、そして(c)がリンク間で共有を行う手法である。(c)のような手法は過去に文献[15]において，CBDA(centrally-buffed, dynamically-Allocated)方式として，理想状態を示す指標として紹介されることはあったがこれまで用いられることはあまりなかった．これは，複数リンクによる同時入出力への対応が困難なためである．仮想チャネルは，時分割で 1 本の物理リンクを共有するため，同時に同一リンク内の仮想チャネルに対するパケットの入出力が生じない．そのため，(b)の手法におけるメモリの割り当てや解放などの処理はリンクごとに行えばよく，各共有メモリへのアクセスも一つのチャネルからのみ行われる．それに対して(c)の手法では、複数のリンクから同時入出力に対応しなければならないため，メモリ割り当てに調停などの処理が必要になり、共有メモリにもマルチポートメモリを使用する必要がある．しかし，一般的なマルチポートメモリであるマルチポートセル型のメモリは，ポート数の 2 乗に比例してハードウェアコストが増加するため，ハードウェアコストへの制約が厳しい NoC での使用は困難である．加えて，メモリの共有には，割り当て情報の管理のために制御用のメモリ要素を使用するが，リンク間の共有により管理対象となるメモリ領域が，(b)の手法の L 倍(L はリンク数)となり，容量が大幅に増大してしまうという問題もある．また、(c)の手法には他にも、処理の複雑化によるパイプライン段数の増加や特有のデッドロックなどの問題もある。

そこで我々は、(c)のように複数の物理リンクにより 1 個のメモリを共有し，上記の問題を解決した手法を提案する[18][19][20][21][22][23]。提案手法のルータ構造を図 3-2 に示す。図 3-2 のように提案手法のルータは，全ての物理リンクで 1 個の共有メモリ(Shared Memory)を持ち，隣接ルータからの入力ポートと各チャネル専用のバッファである専有部(Private Buffer)の間に配置する。

本章では提案手法の概要を説明し、通信性能について評価する。2 節では、共有メモリについて説明し、3 節でパイプラインについて説明する。そして、4 節でデッドロックについて説明し、最後に通信性能の評価について述べる。

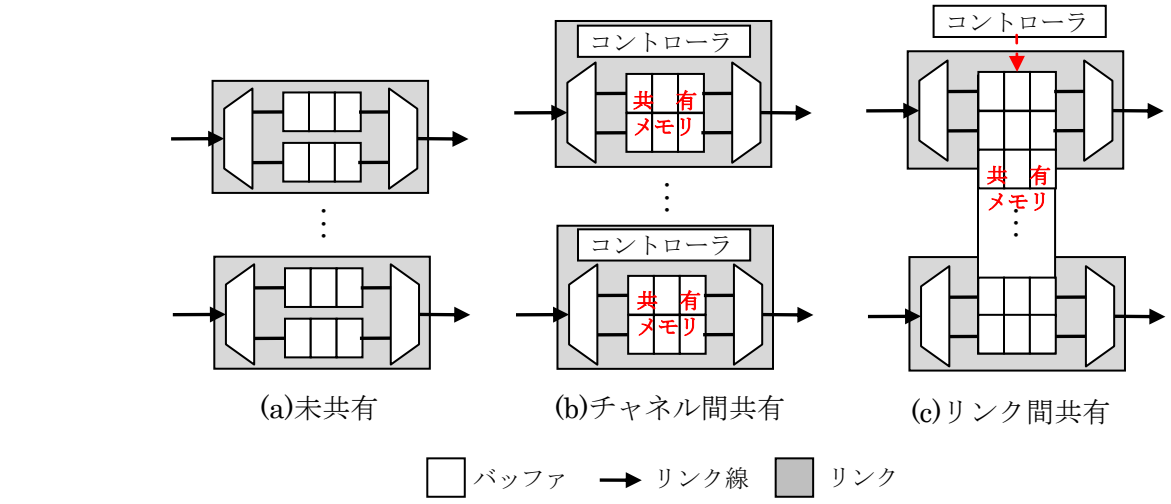


図 3-1　リンク間共有の概念図と比較

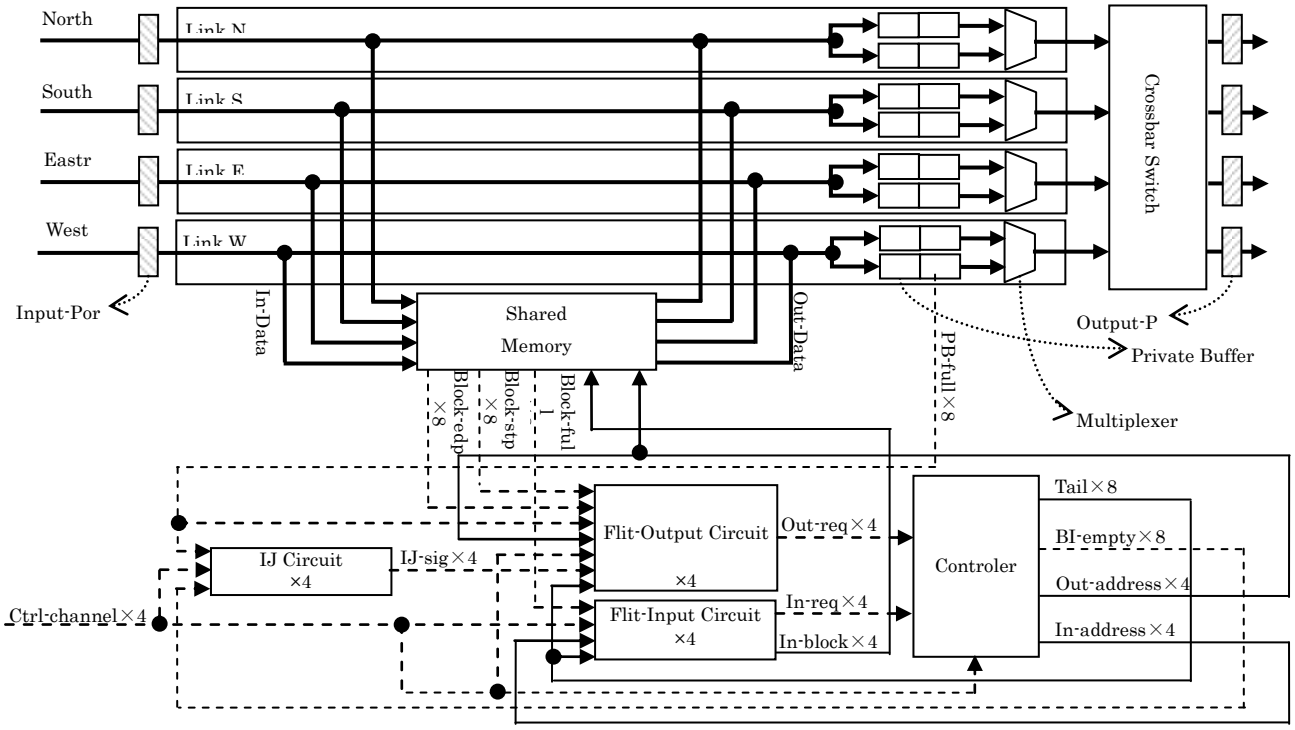


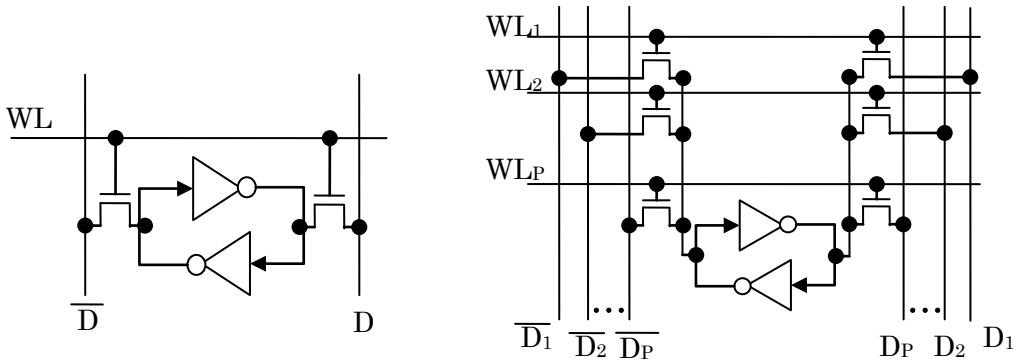
図 3-2　提案手法の構造

3.2. 共有メモリ：マルチポートメモリ

3.2.1. マルチポートセル型マルチポートメモリ

図 3-3 に一般的な 6 トランジスタ型とマルチポートセル型マルチポートメモリの SRAM セルを示す．図 3-3 中の D はデータの転送に使われるデータ線，WL は読み書きの制御信号を流すワード線，P はポート数である．図 3-3(a)にあるように，6 トランジスタ型 SRAM セルは，記録部である二つのインバータ(二つのトランジスタから構成される)と読み書きを制御する二つのトランジスタから構成されている．記録部は，二つの安定状態があり，それを 0，1 に対応させて情報を記録する．

図 3-3(b)にあるように，マルチポートセル型のマルチポートメモリは，各メモリセルをマ



(a)1 ポート SRAM セル　　(b)P ポート SRAM セル

図 3-3 マルチポートセル型マルチポートメモリの SRAM セル

ルチポート化する手法である．この手法には複数の入力ポートから同一セルにアクセスがない限り衝突が起きないという特徴がある．しかし，チップ面積がポート数の 2 乗に比例して増加するため，少ないポート数であってもハードウェアコストが大きく増大してしまうという問題がある．このことから，ハードウェアコストへの制約が厳しい NoC での使用は困難である．

3.2.2. バンク型マルチポートメモリ

リンク間共有法の共有メモリにはマルチポートメモリを使用するが，前述のように通常のマルチポートメモリを使用した場合，ハードウェアコストが膨大なものとなるため，バンク型マルチポートメモリ[24][25][26][27][28]を使用することによりハードウェアコストの増大を抑える。バンク型マルチポートメモリの構造を図 3-4 に示す。バンク型マルチポートメモリは図 3-4 に示した通り，入出力ポートと 1 ポートメモリであるバンクメモリをクロスバスイッチ等の結合網によって接続することで構成されるマルチポートメモリである．マルチポートセル型と異なり，各メモリセルをマルチポートにしないので，比較的少ないハードウェアコストで実装できる．例えば，８ポート３２バンクのバンク型マルチポート

メモリにおいて，データの入出力に要する回路のハードウェアコストが，メモリセル自体に要するコストと同等程度以下であることが知られており[26]，複数本の物理リンクにまたがる手法の実用的な実装形態の実現が可能となる。しかし，通常のマルチポートメモリと異なり，バンク型マルチポートメモリは同じバンクのメモリ領域に対して同時にアクセスすることができないという制限がある。そのため、従来法のようなフリット単位のメモリ領域を一つずつ割り当てる手法では，アクセスの競合が発生する。これについて図 3-5 に示す．図 3-5 のバンク内の数字はその領域を取得したポート番号とする。図 3-5 では，ポート 0 と 1 がそれぞれバンク 0 のメモリ領域にアクセスすることにより，衝突が起きている．このような場合，一般的なクロスバスイッチでは調停処理を行い，どちらかの要求を優先する．しかし，このような処理をルータ内のバッファで行ってしまうと通信性能の低下を招く恐れがある．

また，マルチバンク型には，クロスバスイッチのような閉塞網以外に非閉塞網を用いる実装法もある．非閉塞網を用いた実装はハードウェアコストが少ないという利点もあるが，閉塞網とちがい，複数のポートが異なるバンクにアクセスする場合にも結合網内で衝突起きるため，ルータ内のバッファに使用すると通信性能の低下を招く恐れがある．

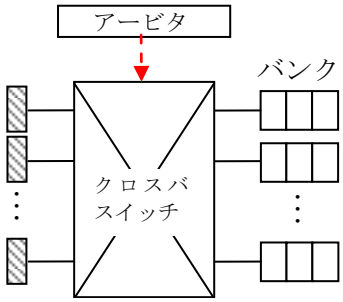


図 3-4　バンク型マルチポートメモリの構造

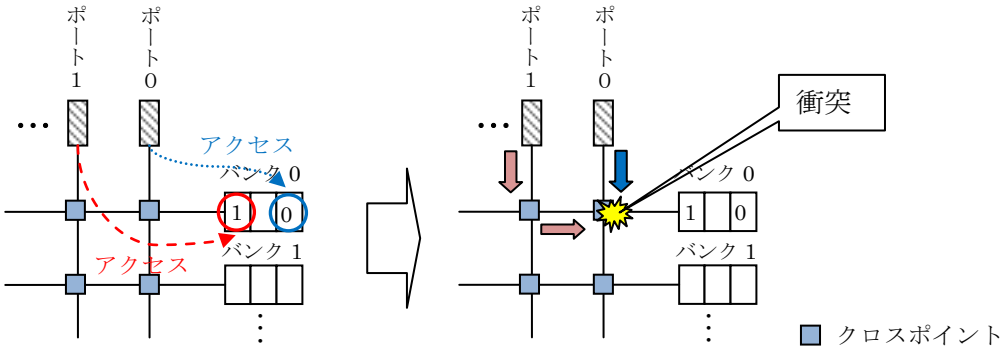


図 3-5 バンク型マルチポートメモリの同時アクセス問題

3.2.3. ブロック単位共有

バンク型マルチポートメモリの同時アクセス問題を解決するため，提案手法ではブロック単位共有を提案している．この手法は各バンクをブロックという単位とし，その単位でブロックの割り当てや開放を行うものである．図 3-6 に従来法などで使用されていたフリット単位の共有とブロック単位共有の違いを示す．図 3-6 では，両手法のメモリ割り当て時の動作について示したものである．図 3-6(a)のように，フリット単位共有では，フリット入力時にフリット一つ分のメモリ領域を確保し，取得したメモリ領域にフリットを格納する．対してブロック単位共有法は図 3-6(b)のように，フリット入力時に取得するブロックに含まれる全てのメモリ領域を取得し，その先頭にフリットを格納する．こうすることにより，各バンクへの入出力がそのバンク(ブロック)を取得したチャネルからのみ行われるようになり，同時アクセスの問題が解決される．またこの手法では，各バンクへの同時アクセスが起こらないため，バンク型マルチポートメモリ内のクロスバスイッチに調停処理を行うアービタを設ける必要がないという利点もある．

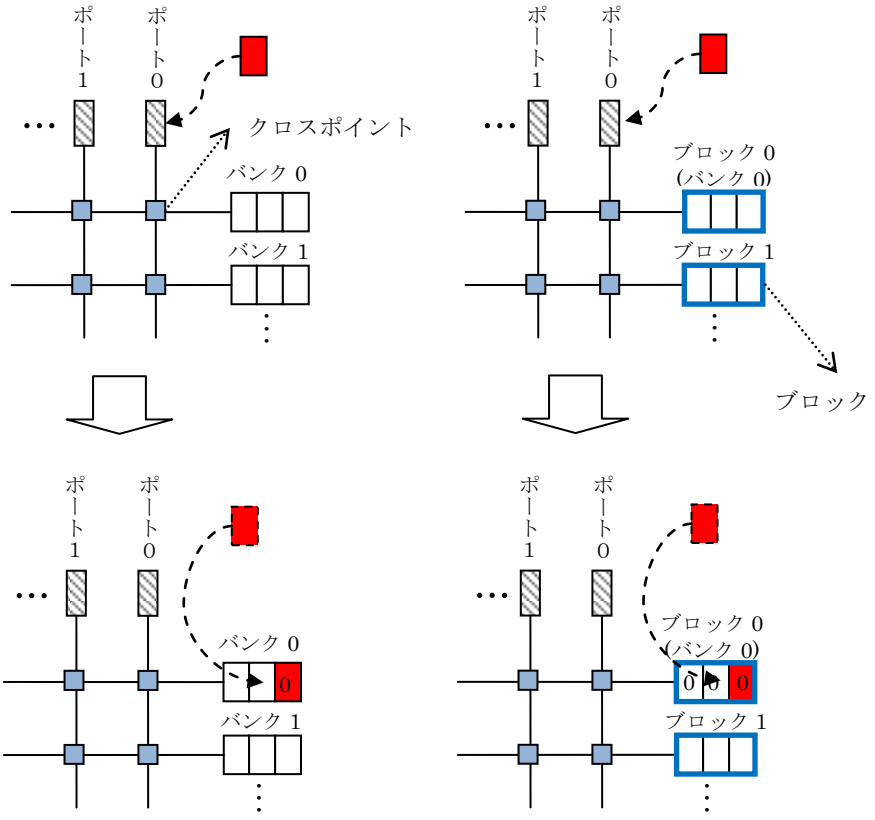
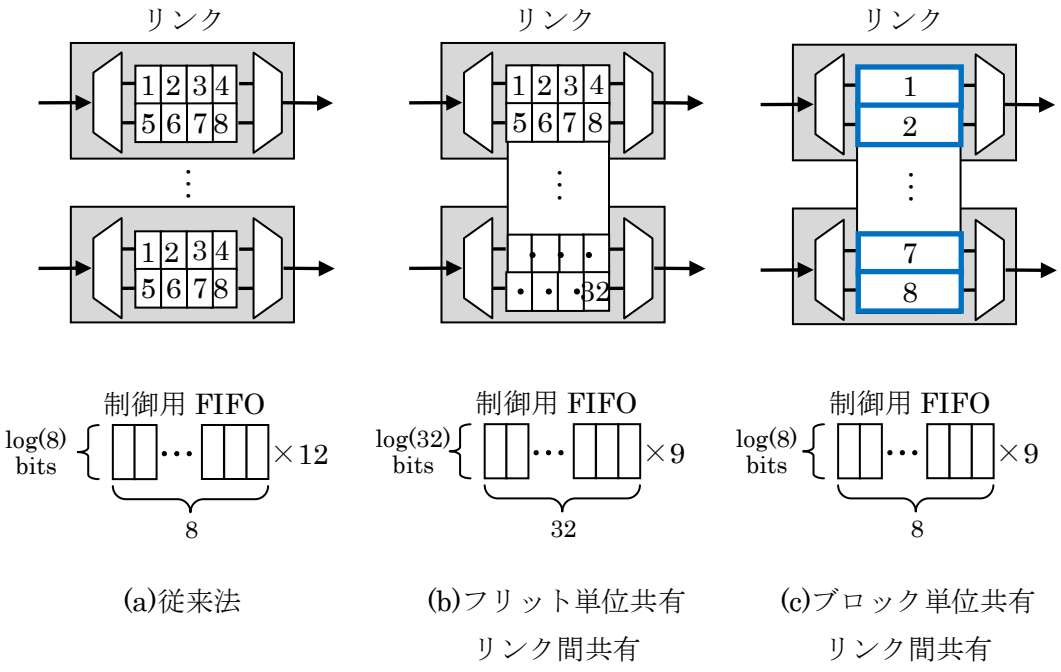


図 3-6　フリット単位共有とブロック単位共有

3.2.4. 制御用 FIFO の容量問題とブロック単位共有

提案手法にはマルチポートメモリ以外にも，制御用 FIFO の容量増加という大きな問題がある．提案手法では共有メモリを管理するため，従来法で使用された Block_Info などの制御用 FIFO を使用するが，単純にフリット単位共有を行うとその管理対象(フリット単位のメモリ領域)の数は従来法の L 倍(L はリンク数)となり，各制御用 FIFO の容量が膨大になる．それに対してブロック単位共有では，管理対象がブロック単位でよいため，その数はフリット単位共有の 1/B(B はブロックサイズとする)となる．

各実装法における容量増減の例として図 3-7 にリンク数 4，リンクごとのチャネル数 2，バッファ総量 32，ブロック数 8 のときの制御用 FIFO 本体(RAM)の容量を示す．図 3-7(a)のように，従来法は管理対象がリンクごとに割り振られたメモリの領域数なので，サイズが 8，ワード長が log(8)bits の FIFO を用いる．そして，チャネル間共有では，制御用 FIFO が合計で 12 個(Free_Pool:4，Block_Info:8)となるので，合計ビット数は 288bits となる．続いて図 3-7(b)にあるように，フリット単位共有を用いてのリンク間共有(以後、「フリット単位リンク間共有」とする) は，管理対象が全リンクの合計メモリ領域数なので，サイズが 32，ワード長が log(32) の FIFO を使用する必要がある．そして，リンク間共有では制御用 FIFO が合計 9 個(Free_Pool:1，Block_Info:8)となるので，合計ビット数は 1440bits となる．最後にブロック単位共有は管理対象がブロック数であるため，図 3-7(c)のようにサイズが 8，ワード長が log(8)bits の FIFO を用いる．そして，リンク間共有なので制御用 FIFO の合計は 9 個となり，合計ビット数は 216bits となる．この結果を見てみる



(a)従来法　　(b)フリット単位共有 リンク間共有　　(c)ブロック単位共有 リンク間共有

図 3-7　制御用 FIFO 本体の容量

と，フリット単位リンク間共有は，制御用 FIFO 本体の容量が従来法の 5 倍以上となっており，その実装が現実的でないことが確認出来る．対してブロック単位共有は管理対象がブロックのため，ビット数が低く抑えられており，この場合では従来法よりも少ないビット数で制御用 FIFO を実装できることが分かる．また，FIFO には前述したとおり，読み書きのためもポインタなどもあり，その容量もこの手法によって削減できるため削減幅はより大きいものとなる．ここで用いた評価は一例であり，より詳しい評価とその方法は，後述する「4.1. ハードウェアコストの評価」で説明する．

3.2.5. 提案手法のバンク型マルチポートメモリ

図 3-8 に使用する提案手法に使用されるバンク型マルチポートメモリを示す．図 3-8 のように，提案手法におけるバンクは入出力が同時に行える 2 ポートの FIFO で構成されており，クロスバスイッチも入力用と出力用の二つが必要になる．入力用クロスバスイッチは，各リンクと各ブロックを接続するため L×B のサイズをもち，入力として「入力データ」と「入力ブロック信号」を持つ．「入力データ」は，各リンクから入力される実際のデータ(フリット)でリンク数分用意される．「入力ブロック信号」は，各リンクがフリットを入力するブロックを示しており，この信号により入力用クロスバスイッチは，「入力データ」と各バンクを適切に接続する．そして出力用クロスバスイッチは，各ブロックと各リンクを接続するため B×L のサイズをもち，出力として「出力データ」，入力として「出力ブロック信号」を持つ．「出力データ」は，各リンクへ出力される実際のデータでリンク数分用意される．「出力ブロック信号」は，各リンクへフリットを出力するブロックを示しており，この信号により出力用クロスバスイッチは，各バンクを適切な「出力データ」に接続する．

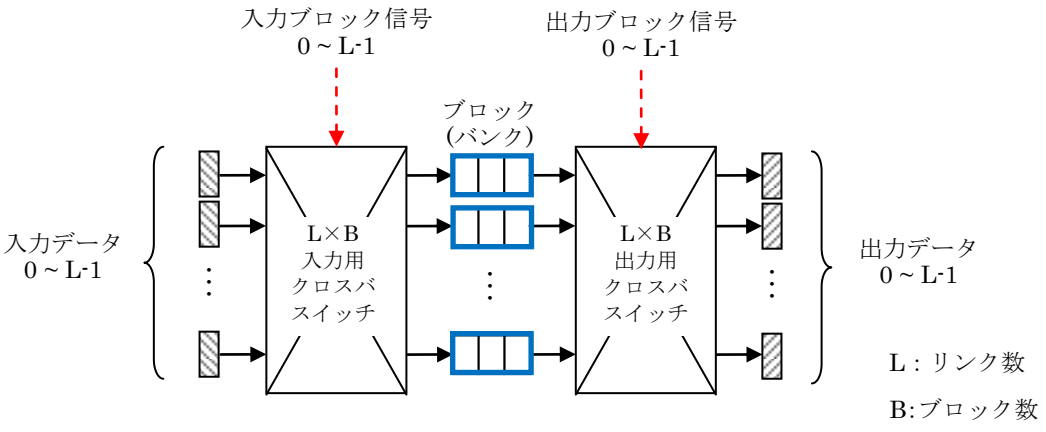


図 3-8　提案手法のバンク型マルチポートメモリの構造

3.3. パイプライン構造

一般的に使用されているルータのパイプラインを図 3-9 に示す[31]。図 3-9 のように一般的なパイプラインは 3 つのステージで構成され，次の 4 つの処理を 3 段のパイプラインにより実行する[31]。

1) Routing Computation(RC)
ヘッダフリットの情報から出力リンクを決定する。

2) Virtual Channel Allocation(VA)
出力する仮想チャネルを割り当てる。

3) Switch Allocation(SA)
クロスバスイッチのアービトレーションと設定を行う。

4) Switch Traversal(ST)
フリットがクロスバスイッチを通過する。

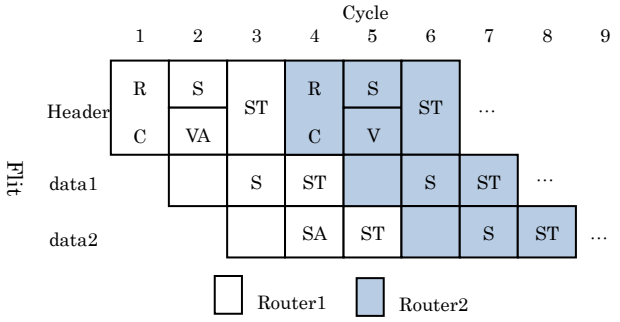


図 3-9　一般的ルータのパイプライン構造

続いて，提案手法のパイプラインを説明する。前述したとおり提案手法において，ルータに入力されたフリットは状況によって次の二つの経路のいずれかを通る。

・経路 1：入力ポート⇒専有部⇒出力ポート

・経路 2：入力ポート⇒共有メモリ⇒専有部⇒出力ポート

経路 1 は，通信のブロックが発生せず，専有部のみで通信が可能である場合に通る経路である。経路 1 のパイプラインを図 3-10 に示す。経路 1 は，一般的なルータと同様に 3 ステージのパイプラインから構成される。ただし，図 3-10 にあるように 1 ステージ目で In-jugde(IJ)ステップを行う。IJ は，「共有メモリを使用するか否か」と「新たにブロックを取得するか否か」の判定を行うステップである。パケットの出力リンクは，共有メモリを使用するか否かとは無関係に決まるので，RC と IJ は並列に処理できる。

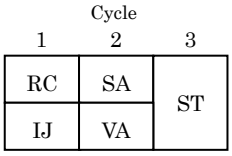


図 3-10　経路 1 のパイプライン

経路 2 は通信のブロックが発生し，IJ ステップで共有メモリを使用する必要があると判定された場合に通る経路である。経路 2 のパイプラインを図 3-11 に示す。提案手法において共有メモリを使用する場合に，必要になるステップは次のようになる。

1)IJ(In-Judge)

入力フリットが共有メモリを使用する必要があるかを判定する。この判定は入力チャネルの専有部の full ビットと Block Info の empty ビットを用いて行われる。同時に，共有メモリを使用する場合，当該チャネルが新たにブロックを取得する必要があるか否かの判定を行う。

2)SiA(Switch-i Allocation)

Block Info，Free Pool の更新を行うと同時に，バンク型マルチポートメモリの入力用クロスバスイッチの設定を行う。

3)SiT(Switch-i Traversal)

SiA ステップに成功した場合，フリットをバンク型マルチポートメモリの入力用クロスバスイッチを通過させて，共有メモリに格納する。

4)SoA(Switch-o Allocation)

バンク型マルチポートメモリの出力用クロスバスイッチの設定，およびブロック解放処理を同時に行う。

5)SoT(Switch-o Traversal)

SoA ステップに成功した場合，バンク型マルチポートメモリの出力用クロスバスイッチにフリットを通過させ，専有部に格納する。

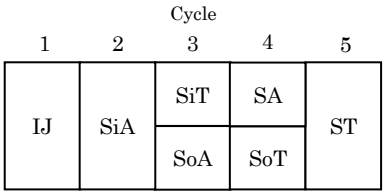


図 3-11　経路 2 のパイプライン構造

図 3-11 のように，経路 2 は経路 1 に比べてパイプラインが 2 ステージ増加する。しかし，以下の理由により，経路 2 のパイプラインによる遅延は隠ぺいされる。

・ネットワークが混雑しておらず，専有部に空きがある場合，3 段パイプラインに従っ

て処理されるため，従来法と同じ量の遅延となる。

・ネットワークが混雑するに従って，専有部に空きが少なくなると共有部が使用される。専有部からクロスバスイッチに送られるパケットがブロッキングされることにより，専有部中のフリットが増えることになるが，そのようなブロッキングを 1 回まで許容できるように専有部を設計すれば(専有部のフリット数を 2 以上にすれば)，共有部を使用したフリットのパイプラインがスムーズに動作する。

提案するパイプラインの動作例を図 3-12 に示す。図 3-12 では，先頭のフリットがブロックされたことにより，3 つ目に入力されたフリットが経路 2 を使用する場合である。
これらのうち，従来型のルータ[13]-[15]と大きく異なる処理は，IJ と SoA におけるブロック解放の処理となる。うち IJ は，「共有メモリを使用するか否かの判定」と「新たにブロックを取得するか否かの判定」となる。IJ の 2 つの処理は，それぞれ 4.2.3 節で紹介する回路によって行われる。これらのうち最も多くのゲートを通るパスは，4 リンク，2 仮想チャンネルのルータにおいては，2 入力のマルチプレクサ一つと複数の論理ゲートを通るのみであるため，パイプライン処理におけるクロック周波数の低下にはつながりにくいと考えられる。

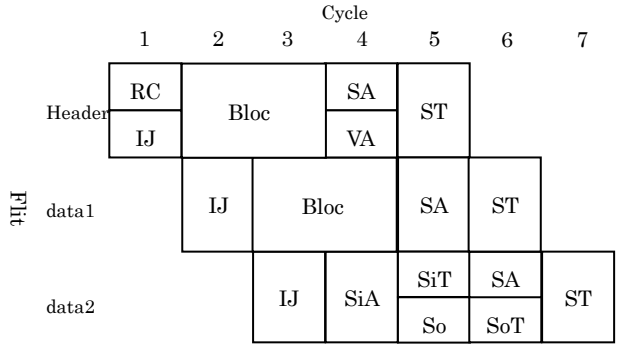


図 3-12　提案手法のパイプライン処理例

3.4. デッドロックの回避

3.4.1. チャネル間共有のデッドロックフリー

通信網においてデッドロックは大きな問題である。そこでデッドロックを回避するため，さまざまな方法が提案されている[9][10][11][29][30]。本稿では，k-ary h-cube において最もハードウェアコストが少ない手法である次元順ルーティング[9]を想定して議論する。各リンク内で共有を行うチャネル間共有では、同じ入力リンク内の仮想チャネルを使用している複数のパケットが全て同じ出力チャネルを要求し，その出力チャネルを使用しているパケット以外のパケットが使用している仮想チャネルが全てのバッファを取得してしまった場合にデッドロックが発生する．図 3-13 にその例を示す．図 3-13 では，パケット 1 が IP コアに向かうチャネルと PE2 のチャネル 1 (図中の ch1)，PE1 の ch1 を取得している．

そして，パケット 2 が PE2 の ch2 を取得しており，コアに向かうチャネルの解法を待っている．図 3-13 ではこのような状況でパケット 2 が取得している PE2 の ch2 が PE2 の全共有メモリを取得してしまったために，パケット 1 は，PE3 にある後続フリットを待ち，PE1 の後続フリットは，PE2 の ch2 がメモリを開放するのを待ち，PE2 の ch2 を取得しているパケット 2 は，パケット 1 がコアに向かうチャネルを解放するのを待っている状態となり，デッドロックを起こしている．

チャネル間共有ではバッファの共有によるデッドロックを防止するため，チャネルを取得した各パケットの最後尾のフリットが取得チャネルを通過するまで，最低でも 1 フリット分のメモリ領域を確保し続けるようになっている．チャネル間共有ではこうすることで，チャネルを取得した各パケットがチャネルを解放するまで，そのチャネルにバッファがなくなることを防ぎ，デッドロックを防止している．

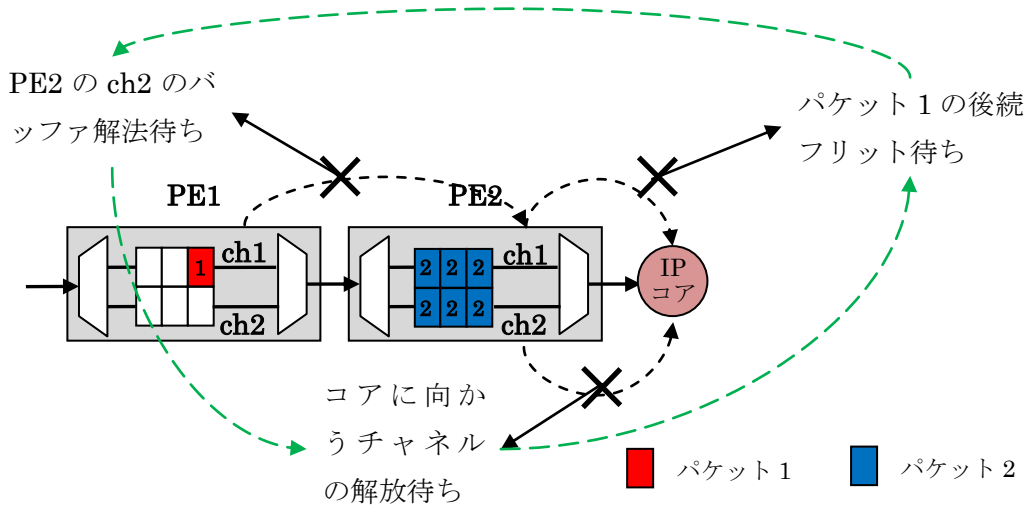


図 3-13　チャネル間共有のデッドロック

3.4.2. リンク間の共有におけるデッドロック

リンク間の共有では，共有メモリの全領域が使用され，空きメモリ領域がなくなることによってメモリを取得できないリンクが発生する可能性がある．もし，そのようなリンクを使用するパケットが存在した場合，そのパケットは他のリンクが共有メモリを解放するまで通信がブロックされることになる．これにより，従来は存在しなかった循環依存が形成され，デッドロックが発生するようになる．デッドロックの発生例を図 3-14 に示す。図 3-14 のように，1 個のバッファを複数のリンクで共有する手法では，バッファ全体が 1 個のパケットで満たされると，双方が進路を妨害する「デッドロック」が発生する。従来法のようなチャネル間共有でも，メモリ取得不能による通信のブロックは，同一リンク内の仮想

チャネル間で発生する．しかし，同一リンク内であるためそれらの間に循環依存が発生せず，デッドロックは起こらない．ただし，トーラス網などのように仮想チャネルを用いてデッドロックを防ぐ結合網では，仮想チャネルにバッファが存在しなくなるだけで循環依存が発生するため，チャネル間共有であってもデッドロックが発生する．また，このデッドロックはチャネル取得時に起こるため，チャネルの取得後から解放までの転送を保証する従来法のデッドロックフリー手法では防ぐことができない．

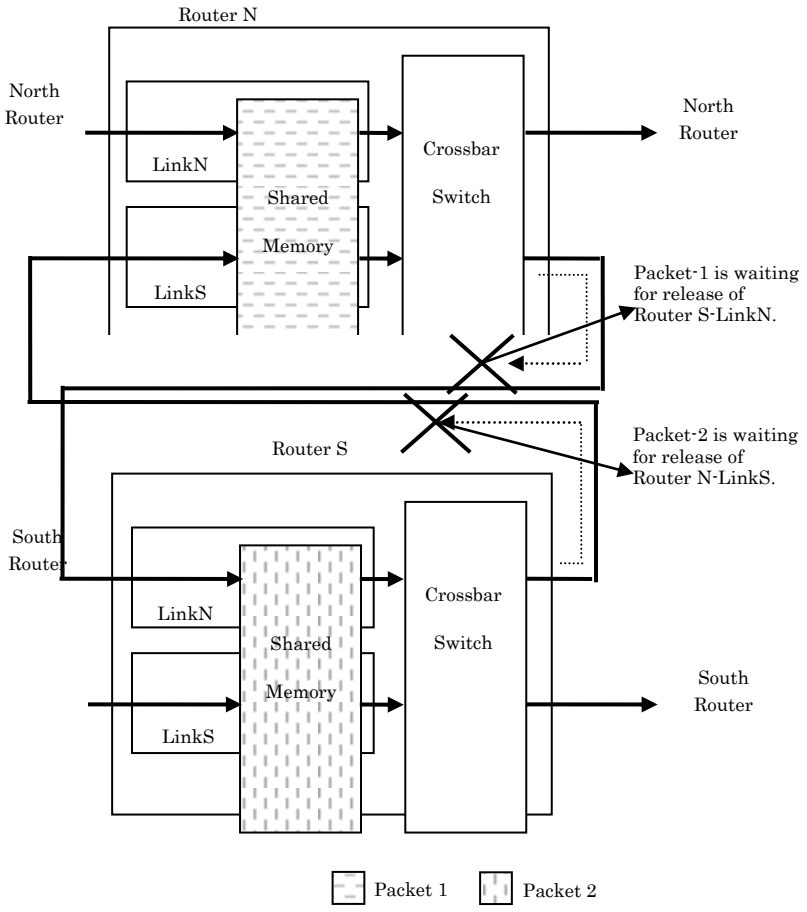


図 3-14　デッドロックの例

3.4.3. 専有部を用いたデッドロックフリー

リンク間共有によるデッドロックは，チャネルやリンクにバッファが存在しなくなることが原因となっている．そのため，本提案手法では各チャネルに専有部という専用バッファを設けることでそれを防いでいる．専有部は、FIFO で構成され，ヘッダフリットを受け取れる最低限の容量が割り振られている．これにより，全チャネルは共有メモリの取得が不可能な場合でも最低限のバッファを持つこととなり，メモリ取得失敗による通信のブロックが起こらなくなる．提案手法は，図 3-2 のように，専有部の手前に共有メモリを挟む構造となっているため，共有メモリを考慮に入れず，専有部をチャネルバッファ本体と見做せ

ば，従来型のルータの構造と同等となる。従って，専有部同士によって結合された相互結合網がデッドロック・フリーであれば，本稿の手法により共有メモリを含むルータによって構成された相互結合網もデッドロック・フリーとなる。この手法は，共有メモリの取得失敗によりチャネルが使用不能になることがないので，仮想チャネルの使用不能によるトーラス網などのデッドロックも防ぐことができ，メッシュ網などにおいてもヘッドオブラインブロッキングの影響を軽減できる可能性がある．

拡張チャネル依存グラフ[30]の考え方に従うと，専有部から隣接ルータの専有部に直接パケットが送られる関係を「直接依存関係」，専有部から共有メモリを介して隣接ルータの専有部にパケットが送られる関係を「間接依存関係」と考えることができる。このような関係を図 3-15 に示す。本稿の手法では，共有メモリは専有部の間にのみ位置するため，同じ組み合わせのチャネル(今回の場合，専有部がチャネルに該当する)間に対して直接依存関係と間接依存関係の双方が成立する。このような依存関係を辺とし，チャネルをノードとした有向グラフを作成し，グラフが閉路を持たなければデッドロック・フリーが証明される[30]

一例として，3×3 メッシュ網において X-Y ルーティングを行う際のチャネル依存グラフを図 3-16 に示す。図 3-16 の灰色の大きな四角がルータ，小さな四角が専有部，矢印がリンクを示している。また，黒色の四角がノード(チャネル)，黒い矢印が依存関係を示していて，同じノード間に対して直接依存関係と間接依存関係の双方が成立する。図 3-16 のグラフは閉路を持たないことから，デッドロック・フリーが示される。

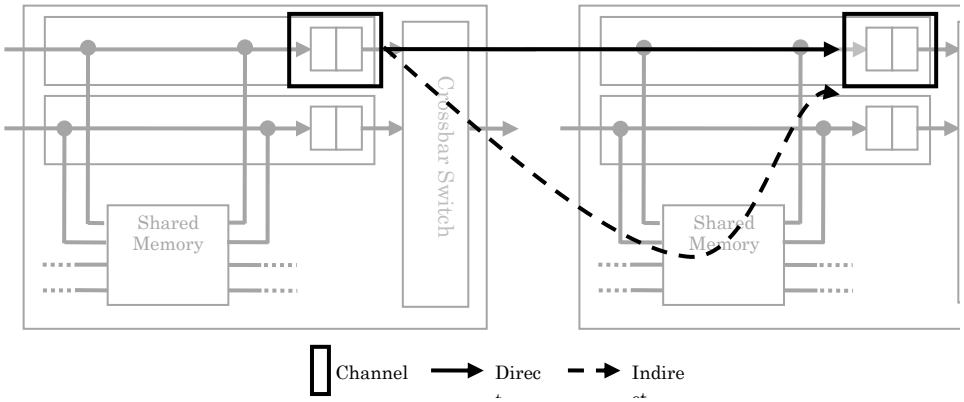


図 3-15 直接依存関係と間接依存関係

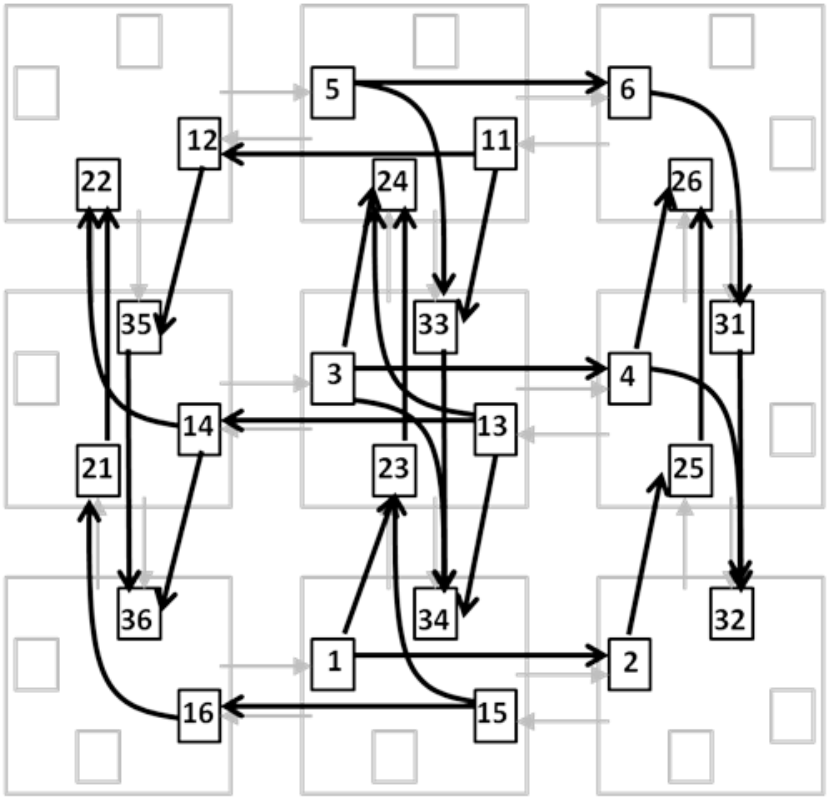


図 3-16 3×3 メッシュ網に次元順ルーティングを用いた場合の
チャネル依存グラフ

3.4.4. 専有部を用いた場合の制御

専有部を含めたメモリの共有を行う場合，はじめに専有部に対するパケットの入力を処理し，専有部が一杯になった場合に，共有メモリ領域の取得を行う。すなわち，提案手法においてチャネルに対してフリットの入力が行われる 際には，下記のルールで入力先が選ばれる。

- 専有部に空きがあり，かつ共有メモリにフリットがないときには，専有部に対してフリットが入力される。
- 上記以外のときには，共有メモリから当該チャネルに割り当てられた最後尾のブロックに対してフリットが入力される。
- フリットの出力は，専有部からのみ行われる。

チャネルが共有メモリからブロックを取得している状態で，専有部からフリットが出力されて専有部に空きができた時には，専有部の後続ブロック(取得された共有メモリの先頭のブロック)から，フリットを専有部に移動する。最後尾のブロックがフリットで一杯になっている状態の時に当該チャネルにフリットが入力されると，ブロック取得の処理が行われる。また，フリットの出力によりブロックが空になると，ブロック解放の処理が行われる。

提案手法では， ⅰ)バッファ本体と周辺回路のほかにⅱ)バッファ制御情報を保持するメモリ要素， ⅲ)ブロック制御用の論理回路を必要とする。

3.5. 動的通信性能

各実装形態における動的通信性能をソフトウェアシミュレータによって比較し，評価する。このシミュレーションは，各 PE で 1 サイクルごとに指定された確率でパケットを発生し，ランダムに選択した他の PE に送信する。これらの処理を 200000 サイクル行い，平均転送時間と平均スループットを記録する。そして同条件にてこのようなシミュレーションを 10 回行った平均値をグラフにプロットする。これらの処理をパケットの発生確率を変化させて実行し，グラフを作成する。このグラフは，横軸が平均スループット，縦軸が平均転送時間としている。

比較対象とパラメータをそれぞれ表 3-1、3-2 に示す。このとき，「By-block and channel-sharing」のブロック数は物理リンクごとに 2,「By-block and link-sharing」のブロック数は全物理リンクで 8 としている。

シミュレーションの結果を図 3-17 から 3-22 に示す。結果より，物理リンク内で複数のチャネルによるメモリ共有を行った場合より，提案手法により全物理リンクによるメモリ共有を行ったほうが性能が向上することが確認できた。また，共有範囲に関らずフリット単位の共有と，ブロック単位の共有で性能に大きな差がないことが分かった。今回の実験では，通信先をランダムとした「一様な」通信を行っているものの，実際には，ラップアラウンドチャネルとその他のチャネルではチャネルの使用法が異なる。また，双方向トーラス網では,パケットがチャネルを使用する頻度が+方向チャネルと-方向チャネルで異なる場合がある。そのため，次元順ルーティングにおいてはチャネルによってパケットが使用する頻度が異なる。その結果として，特定のチャネルがメモリを取得することが多くなるため，チャネルがまとめてメモリを取得する「ブロック単位共有」と，チャネルが少しずつメモリを取得する「フリット単位共有」で通信性能に大きな差が現われないものと考えられる。

平均スループットに対する，共有メモリの利用率を図 3-23 に示す。図 3-23 の横軸は，図 3-17 から 3-22 の実験と同じ条件により計算された平均スループットである。縦軸は，それぞれの条件において，ルータに滞在中のパケットが共有メモリを使用する比率を示している。なお，実験の条件は図 3-22 と同条件としている。

平均スループットが低い場合（ネットワークが混雑していない場合），専有部のみを使用した転送が多いため，従来手法と提案手法のふるまいに大きな差が見られない。平均スループットが高くなる（ネットワークが混雑する）に従って，共有部を用いた転送の割合がゆるやかに増加し，スループットが 0.2 flit/Cycle・PE を超えて従来法と提案手法の違いが現われるあたりで，共有部の割合の増加が激しくなる。ただしその場合も，3.3 節において述べたように，共有部の通過に伴う遅延はほぼ隠蔽される（前方のフリットのブロッキングによって隠される）ため，遅延時間の増加による通信性能のロスは目立たない。

表 3-1 比較する実装形態

	Sharing unit	Sharing Range
No-sharing	don't sharing	
By-flit and channel-sharing	by flit	channel
By-block and channel-sharing	by block	channel
By-flit and link-sharing	by flit	link
By-block and link-sharing	by block	link

表 3-2 シミュレーションのパラメータ

Topology	torus
Communication method	wormhole routing
Routing algorithm	Dimension-order routing
Number of PE	16, 64
Number of physical link	4
Number of virtual channel	2
Total buffer	64
Packet size	16, 32, 64

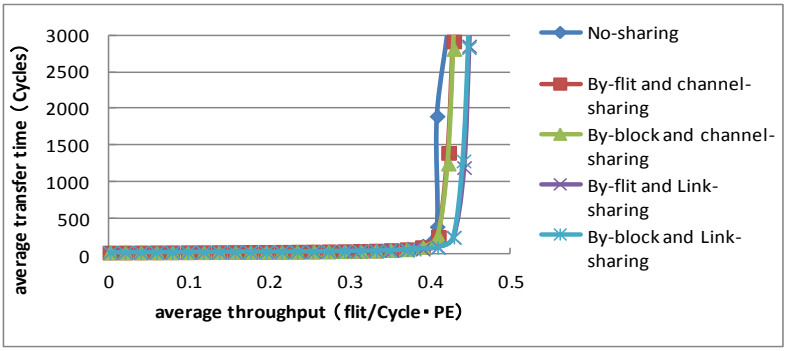


図 3-17 シミュレーション結果

(PE No:16, total buffer:64, packet size:16)

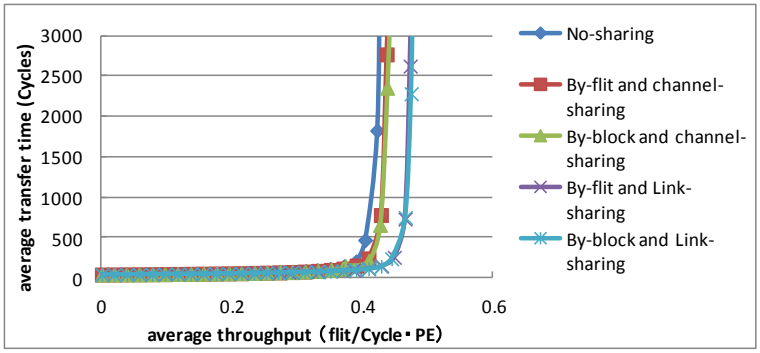


図 3-18 シミュレーション結果

(PE No:16, total buffer:64, packet size:32)

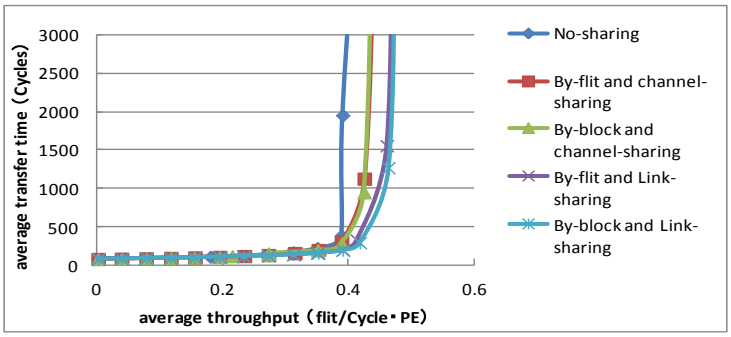


図 3-19 シミュレーション結果

(PE No:16, total buffer:64, packet size:64)

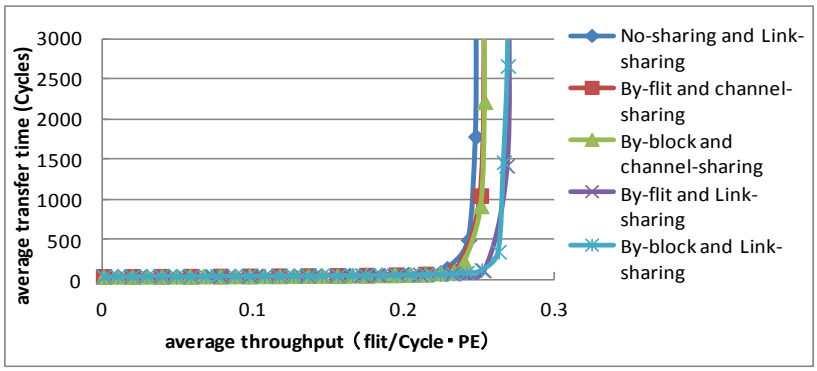


図 3-20 シミュレーション結果

(PE No:64, total buffer:64, packet size:16)

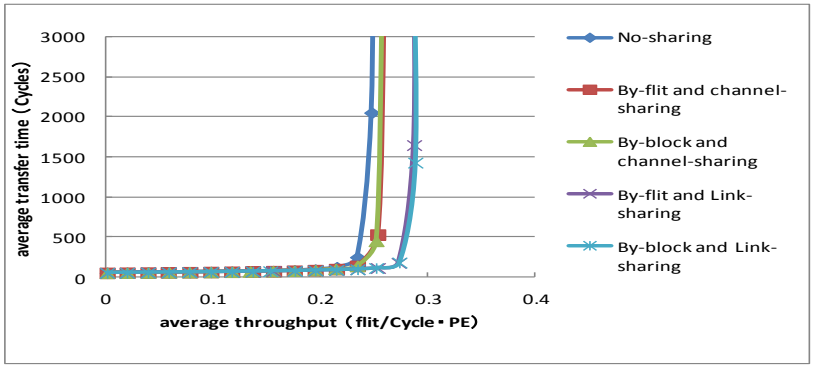


図 3-21　シミュレーション結果

(PE No:64,　total buffer:64, packet size:32)

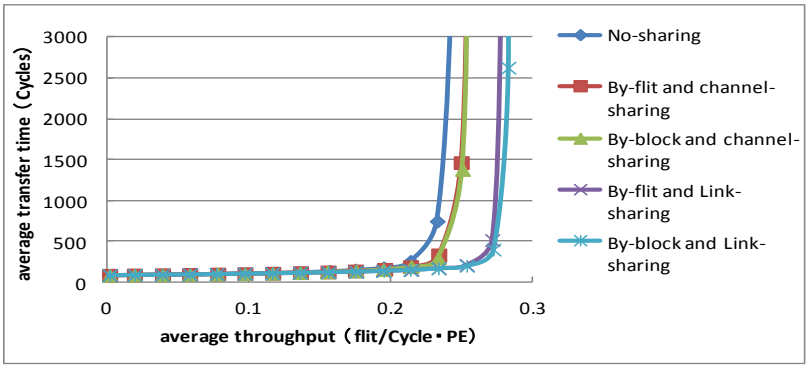


図 3-22 シミュレーション結果

(PE No:64,　total buffer:64, packet size:64)

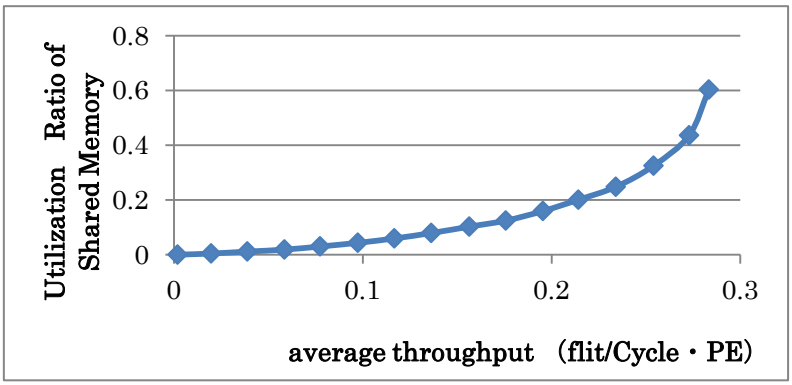


図 3-23　共有メモリの利用率

(PE No:64, total buffer:64, packet size:64)

3.6. まとめ

本章では提案手法の概要を説明し、通信性能について評価した。2 節では、提案手法の概要について述べ、ハードウェアコストを削減する工夫について簡単に説明した。3 節では、提案手法はフリットが通る経路が二つあり、それぞれ 3、5 段のパイプラインになることを述べた。そして、経路間の遅延は隠ぺい可能であることを述べた。4 節では、提案手法におけるデッドロックについて説明し、各チャネル専用のバッファである専有部を設けることでそれを防ぐことができることを説明した。最後に 5 節でが通信性能の評価を行い、提案手法を用いることで性能が大きく向上することを述べた。

第4章 通信性能の評価

4.1. はじめに

本節では，提案手法の通信性能を評価する．評価はソフトウェアシミュレータを用いて行い，これによって各実装形態の結果を比較する．まず，評価 1 ではブロック数による提案手法の性能の変化を確認し，基準となるブロック数を決定する．評価 2 では，様々な条件を変更し，提案手法の性能を多面的に評価する．

通信性能の評価は，三浦らにより作成されたソフトウェアシミュレータ[33][34]によって行う．このシミュレータは，C 言語で作成されており、ルータ回路の動作をブロック単位で再現したもので、多くの研究で用いられている[35]-[40]．本シミュレータでは，各サイクルで全ての PE が指定された確率でパケットを生成し，ランダムに選ばれたほかの PE にパケットを送る．これらの処理を 20000 サイクル実行し，平均転送時間と平均スループットを記録する．そして，同条件にてこのようなシミュレーションを 10 回行った平均値をグラフにプロットする．これらのグラフは，横軸が平均スループット，縦軸が平均転送時間となっている．各シミュレーションの条件は表 4-1 のようになる．表中のバッファ総量とパケット長のかっこは単位である．

表 4-1 シミュレーションのパラメータ

トポロジ	2 次元メッシュ，2 次元トーラス
通信方式	ワームホールルーティング
ルーティングアルゴリズム	次元順ルーティング
PE 数	16(4×4)， 64(8×8)
物理リンク数	4
仮想チャネル数/物理リンク	2
バッファ総量(フリット/ルータ)	32，64
パケット長(フリット/パケット)	16， 32， 64

4.2. 評価 1：ブロック数と通信性能の関係

評価 1 では，ブロック数によって提案手法の性能にどのような影響を与えるかを評価する．提案手法では，ブロック数が少ないほどハードウェアコストは大幅に小さくなる．これは，バンク型マルチポートメモリ内のスイッチや，共有メモリを管理するメモリ要素のサイズなどがブロック数によって決定するためである．しかし，ブロック数が少ないほどメモリの利用効率は落ちるため，通信性能は落ちる可能性がある．図 4-1～4-12 にトーラス網とメッシュ網のシミュレーション結果を示す．図中の上のグラフはトーラス網，下のグラフは

メッシュ網の結果である．この評価において我々は以下の実装法を用いて比較を行う．

・no-sharing　　：共有を行わない手法

・by-flit-link：ブロック単位共有を使用しないリンク間共有

・B2, B4, B8　　：提案手法．各ブロック数はそれぞれ 2 (B2), 4 (B4), 8 (B8)となっている．

結果よりブロック数が 2(B2)，4(B4) の場合は by-flit-link より性能が低下する場合が多いことがわかる．一方でブロック数が 8(B8)の提案手法は by-flit-link と比較しても性能に大きな違いがないことがわかる．この結果より 2D メッシュおよびトーラスにおいて，性能上の最適ブロック数は 8 個であることがわかる．以後，ブロック数 8 を提案手法の基本値として扱うものとする．

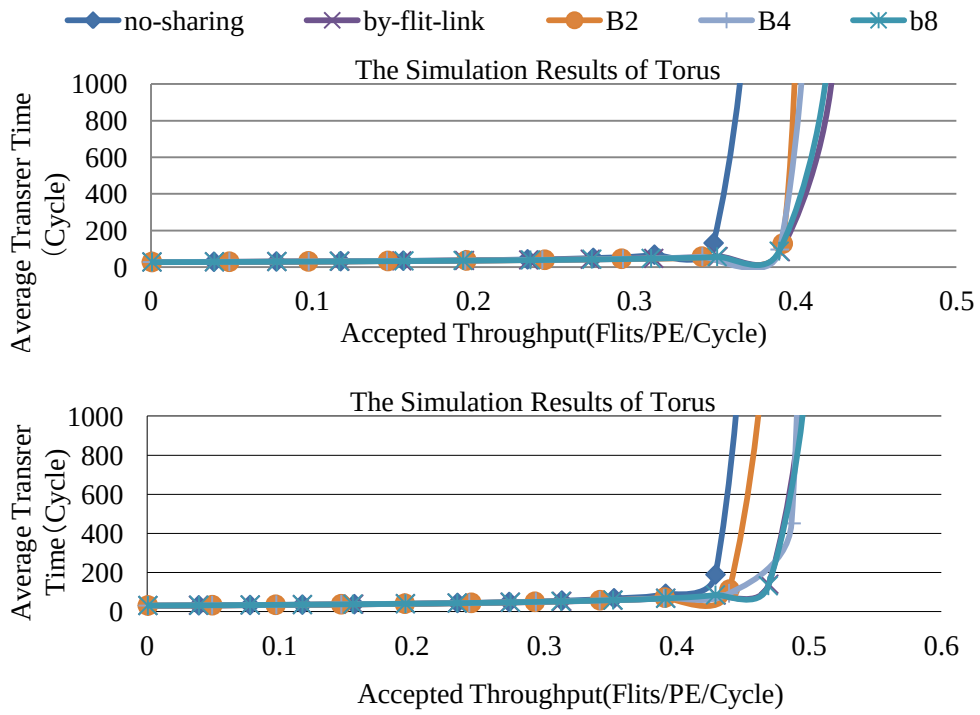


図 4-1 評価 1 のシミュレーション結果(16 PE, 32 Buffer, and 16 Flits/Packet)

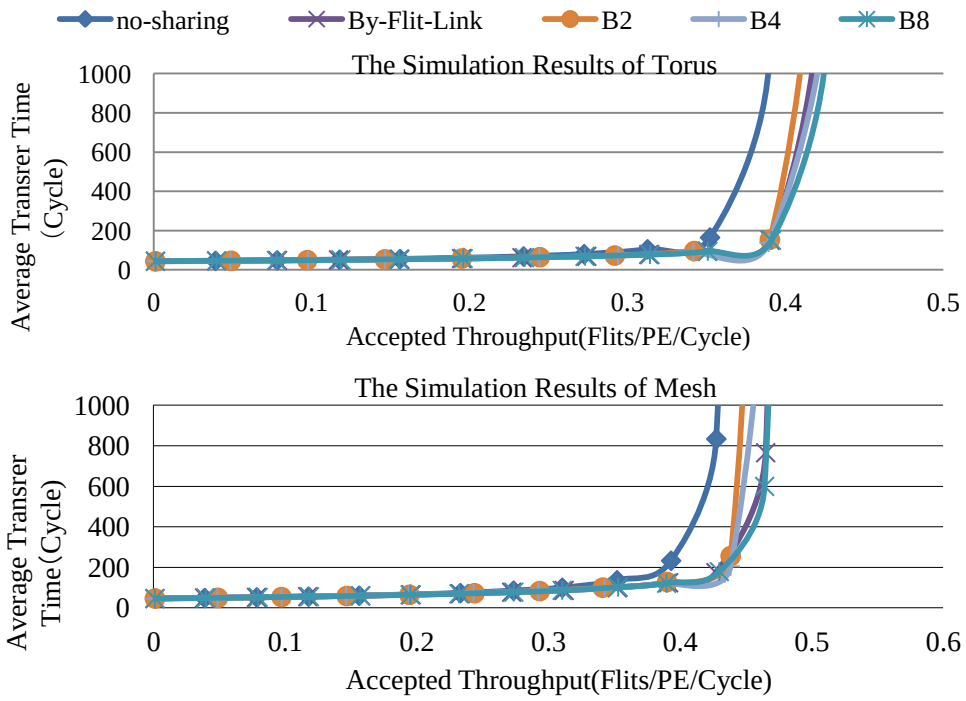


図 4-2 評価 1 のシミュレーション結果(16 PE, 32 Buffer, and 32 Flits/Packet)

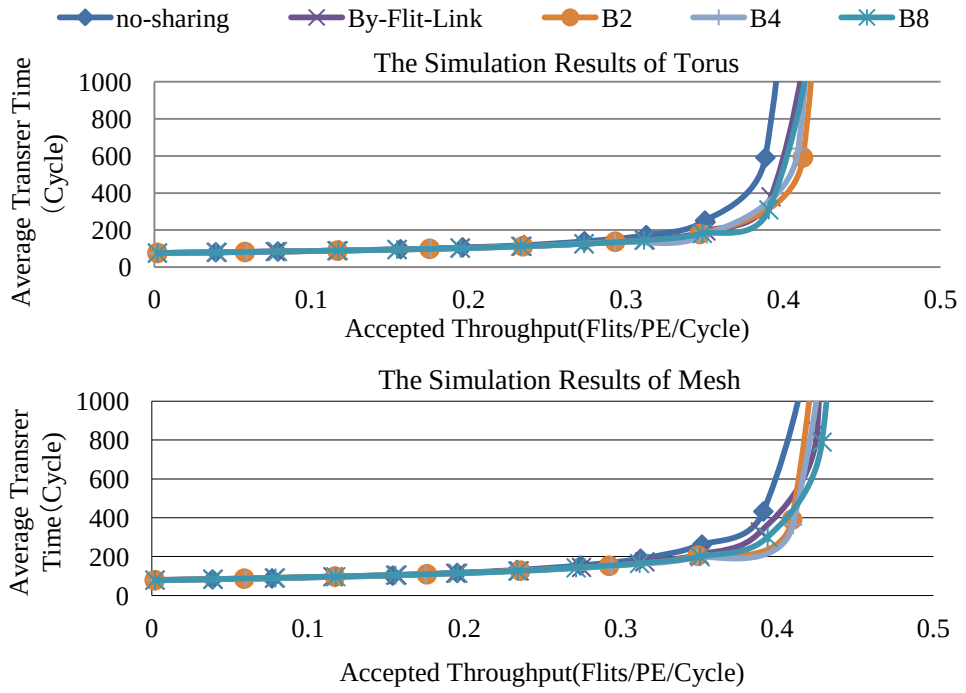


図 4-3 評価 1 のシミュレーション結果(16 PE, 32 Buffer, and 64 Flits/Packet)

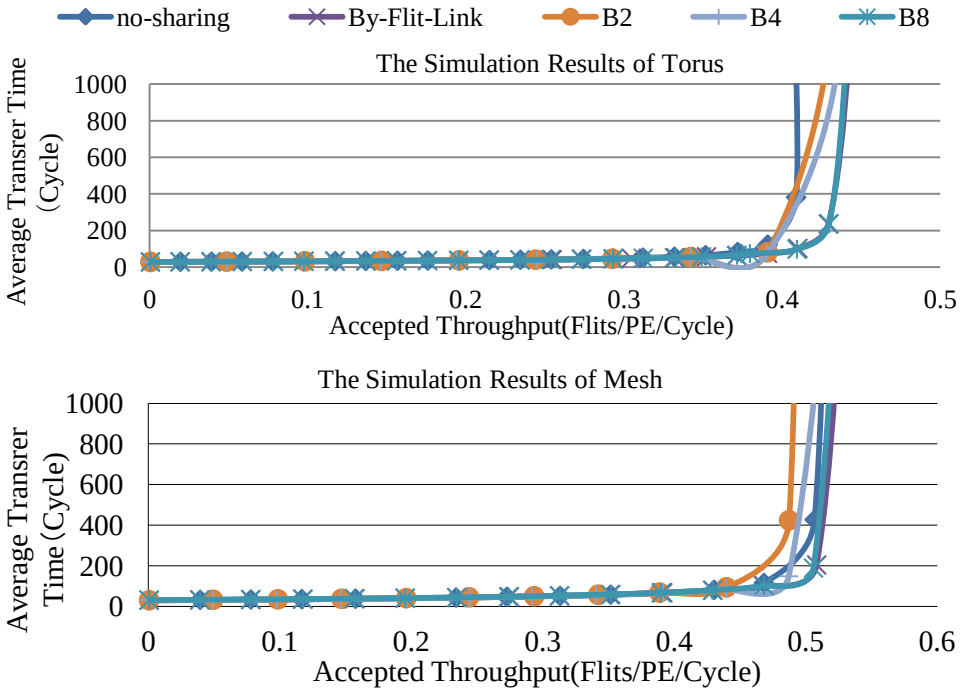


図 4-4 評価 1 のシミュレーション結果(16 PE, 64 Buffer, and 16 Flits/Packet)

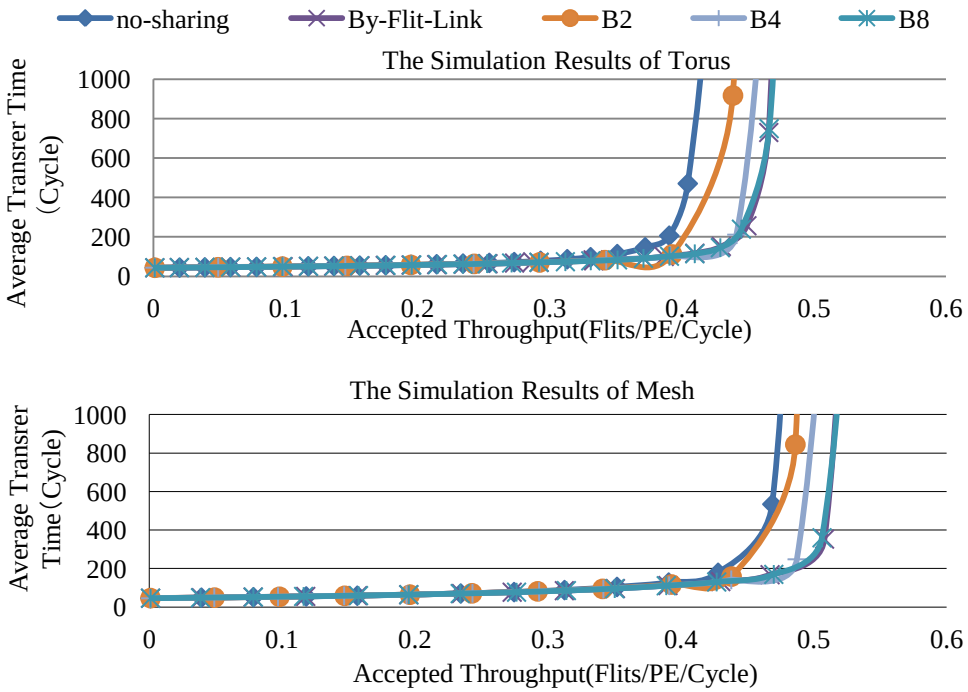


図 4-5 評価 1 のシミュレーション結果(16 PE, 64 Buffer, and 32 Flits/Packet)

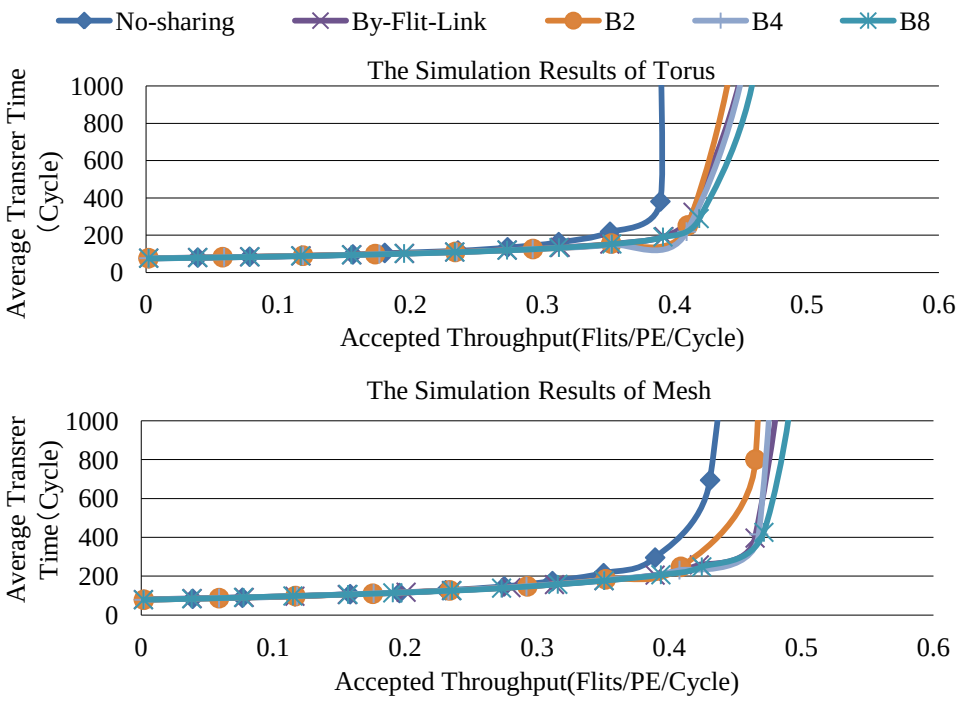


図 4-6 評価 1 のシミュレーション結果(16 PE, 64 Buffer, and 64 Flits/Packet)

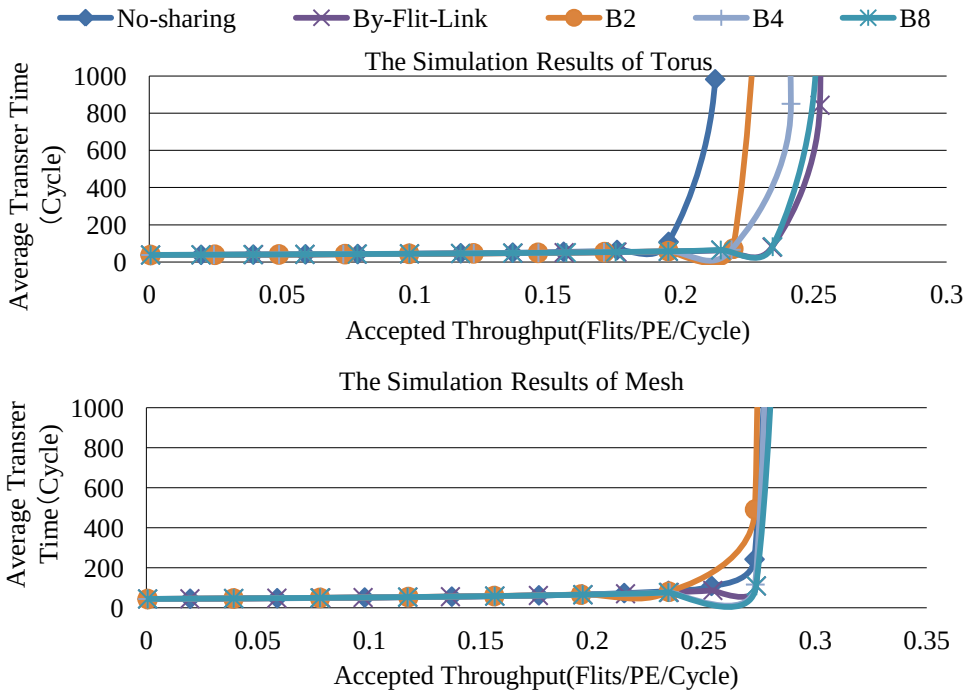


図 4-7 評価 1 のシミュレーション結果(64 PE, 32 Buffer, and 16Flits/Packet)

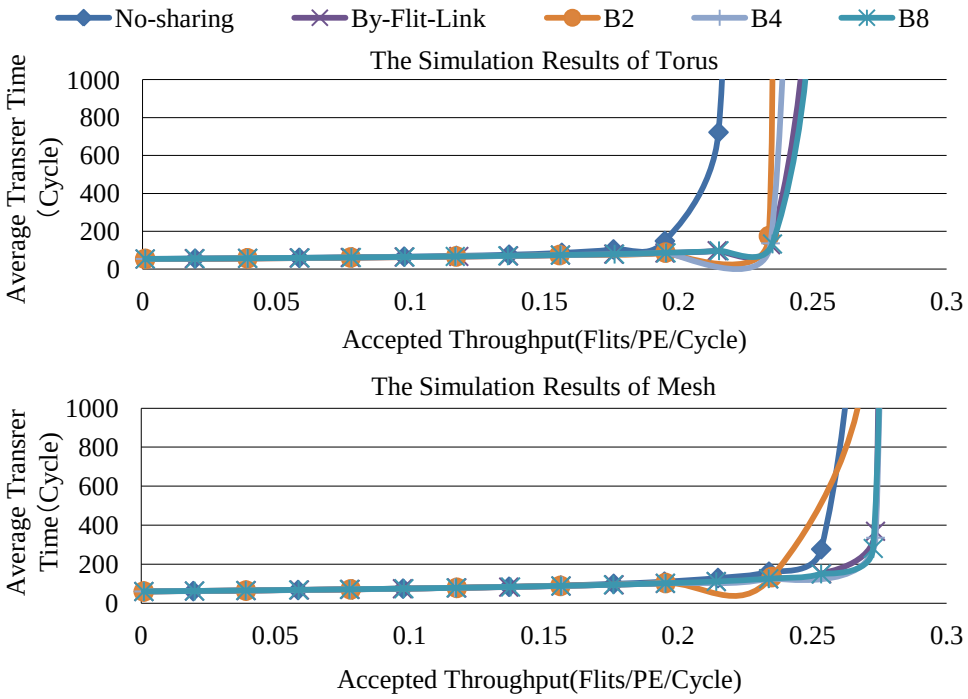


図 4-8 評価 1 のシミュレーション結果(64 PE, 32 Buffer, and 32Flits/Packet)

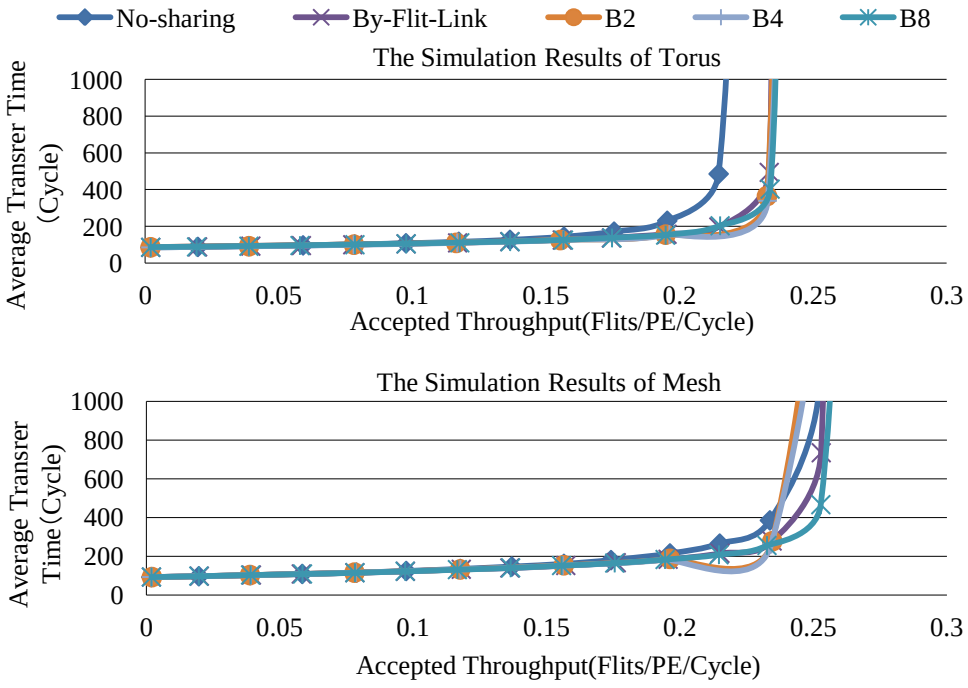


図 4-9 評価 1 のシミュレーション結果(64 PE, 32 Buffer, and 64Flits/Packet)

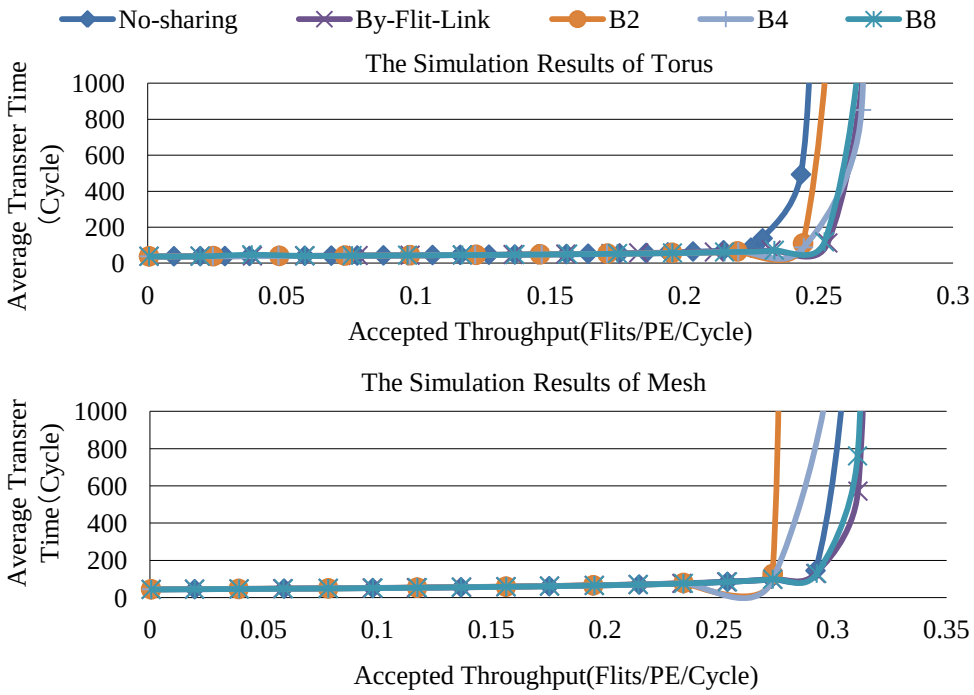


図 4-10 評価 1 のシミュレーション結果(64 PE, 64 Buffer, and 16Flits/Packet)

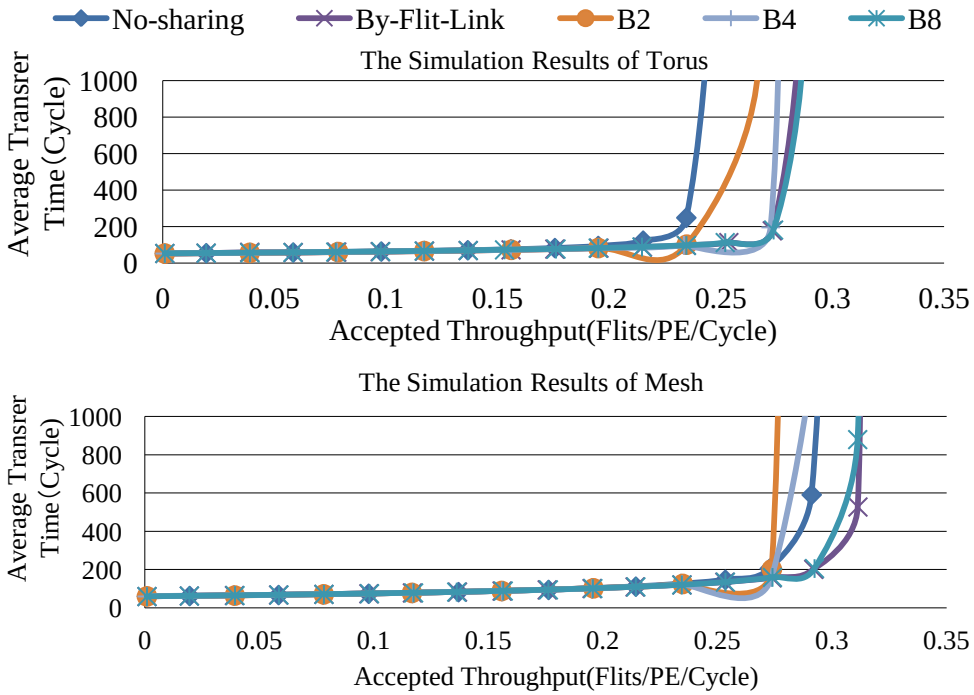


図 4-11 評価 1 のシミュレーション結果(64 PE, 64 Buffer, and 32Flits/Packet)

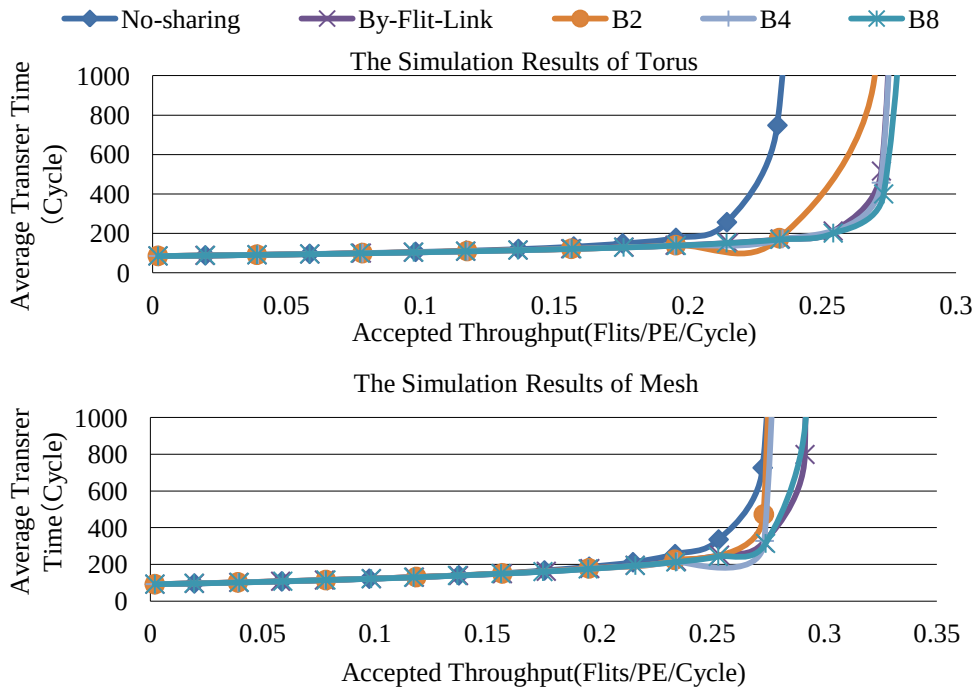


図 4-12 評価 1 のシミュレーション結果(64 PE, 64 Buffer, and 64Flits/Packet)

4.3. 評価 2：トポロジ，PE 数，バッファ容量，パケット長と通信性能の関係

本節において我々は，トポロジ，PE 数，バッファ容量，パケット長などの様々な条件を用いてシミュレーションを行い，提案手法の性能を多面的に評価し，結果について考察する．この評価において我々は以下の実装法を用いて比較を行う．

・no-sharing　：共有を行わない手法

・by-flit-link：ブロック単位共有を使用しないリンク間共有

・channel-sharing：各物理リンク間の仮想チャネル間でメモリを共有する手法

・proposed methods：評価 1 の結果よりブロック数を 8 個とした提案手法

図 4-13 から 4-24 にトーラス網とメッシュ網のシミュレーション結果を示す．このグラフは，上のグラフがトーラス網，下がメッシュ網の結果となっている．図より，提案手法の性能は，未共有やチャネル間共有に比べて高くなることがわかる．また評価 1 と同様に，提案手法とフリット単位共有に大きな差がないことも確認できる．

未共有に対する提案手法の性能増加率を表 4-2 に示す．表 4-2 より，合計バッファ(フリット/ルータ)とパケット長(フリット/パケット)が近いとき，性能が上がる傾向にあることが分かる．一方でパケット長が合計バッファより大幅に大きい，あるいはパケット長が合計バ

ッファより非常に小さい場合，性能の向上幅は小さい傾向にあることも確認できる．前者は，パケットが大きすぎるために常に共有メモリが使い切られてしまい，メモリが必要なチャネルに共有メモリを割り振ることができないことが多いためだと考えられる．後者は，パケットが小さいため共有メモリが使用される割合が小さく，従来法の動作と変わらないことが多いためであると考えられる．

また，PE 数の多いほどメッシュ網の向上率がトーラス網に比べて小さいこともわかる．メッシュ網では，メッシュ網の端の PE に使用されていないリンクがあり，そこに割り当てられているバッファが有効に使用されていない．そのため PE の少ないメッシュ網は提案手法によって性能が向上しやすい．たいして，より多くの PE を持つメッシュネットワークでは， 端の PE の割合が全体に対して少ないため，効果が小さいと考えられる．さらにトーラス網と異なり，メッシュ網はデッドロックの回避をチャネルひとつで行えるため，二つのチャネルを自由に使うことができる．このため，提案手法と従来法で性能差小さいと考えられる．

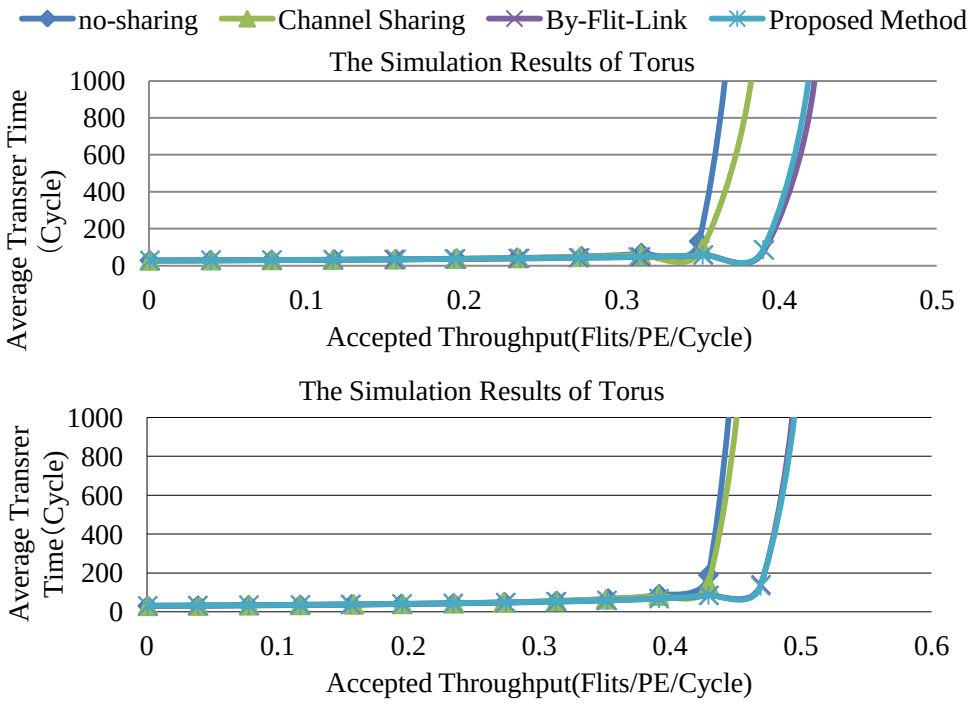


図 4-13 評価 2 のシミュレーション結果(16 PE, 32 Buffer, and 16 Flits/Packet)

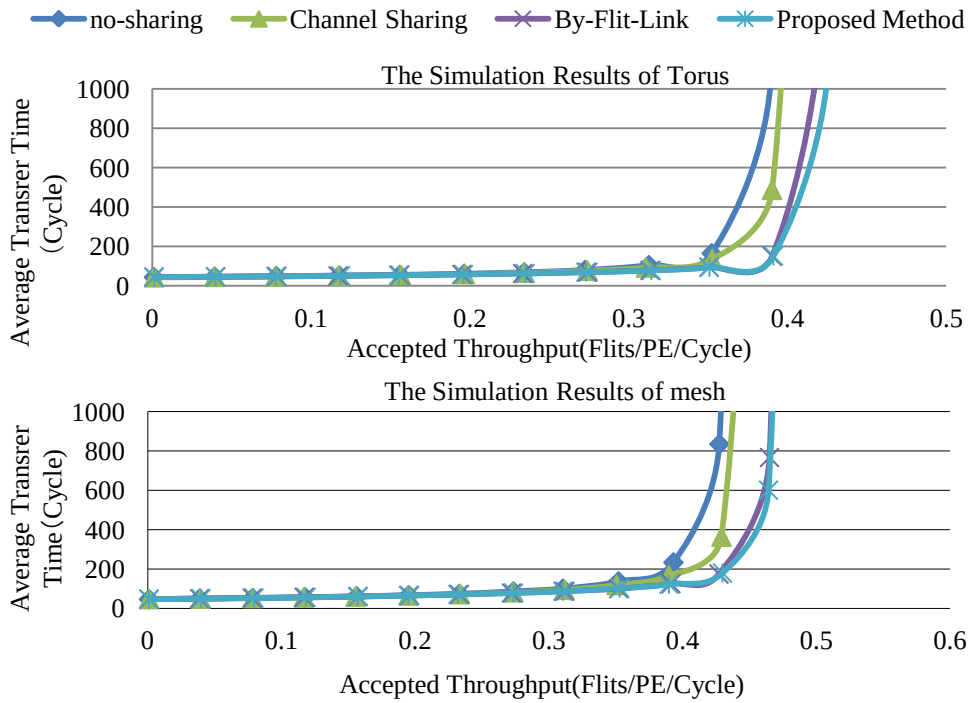


図 4-14 評価 2 のシミュレーション結果(16 PE, 32 Buffer, and 32 Flits/Packet)

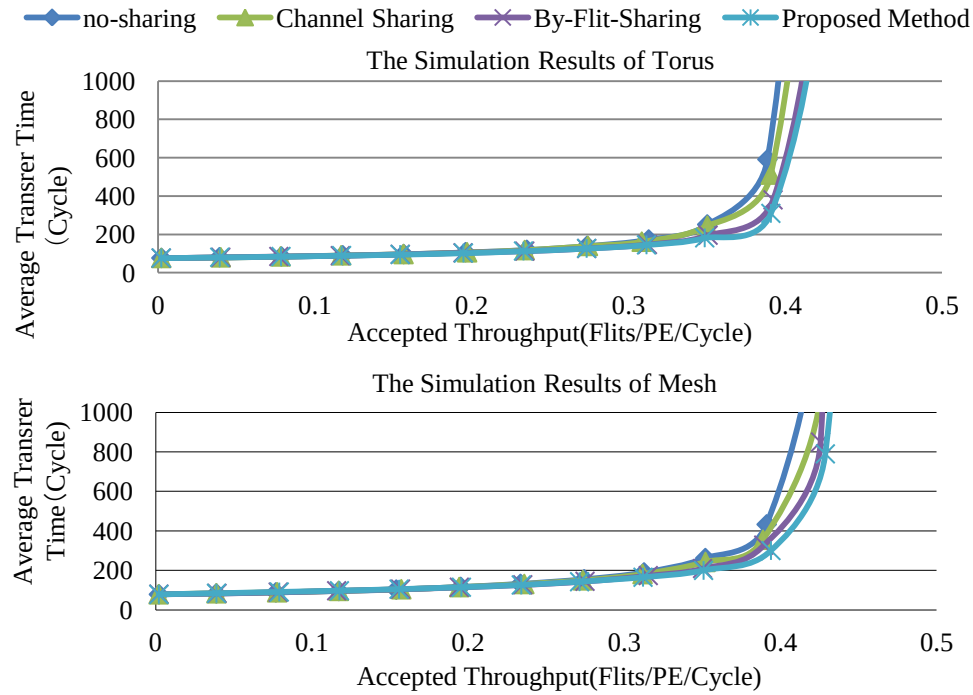


図 4-15 評価 2 のシミュレーション結果(16 PE, 32 Buffer, and 64 Flits/Packet)

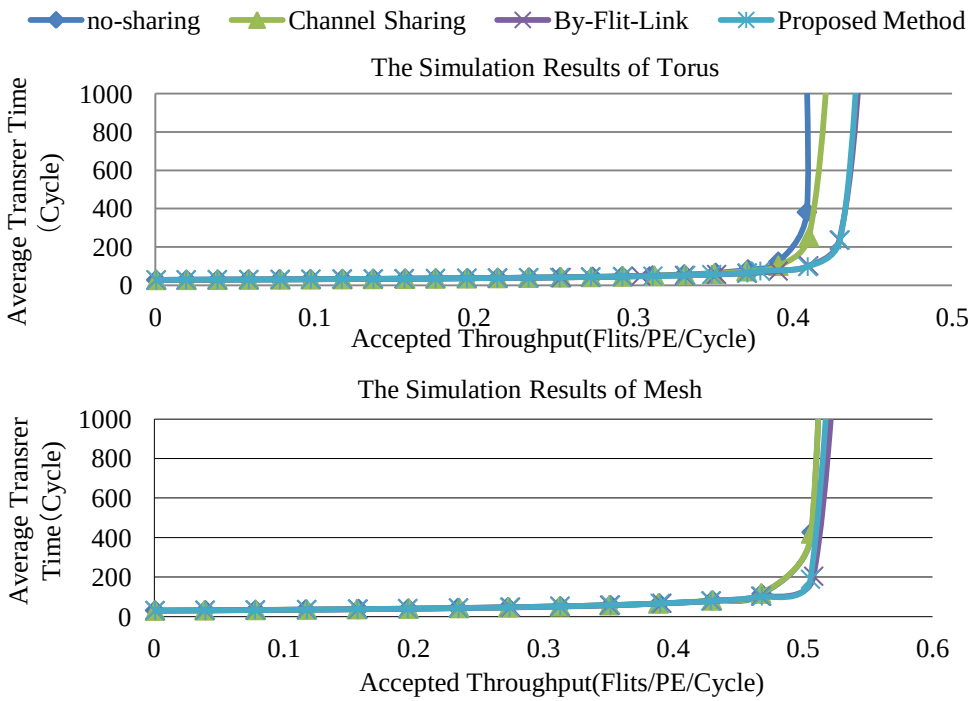


図 4-16 評価 2 のシミュレーション結果(16 PE, 64 Buffer, and 16 Flits/Packet)

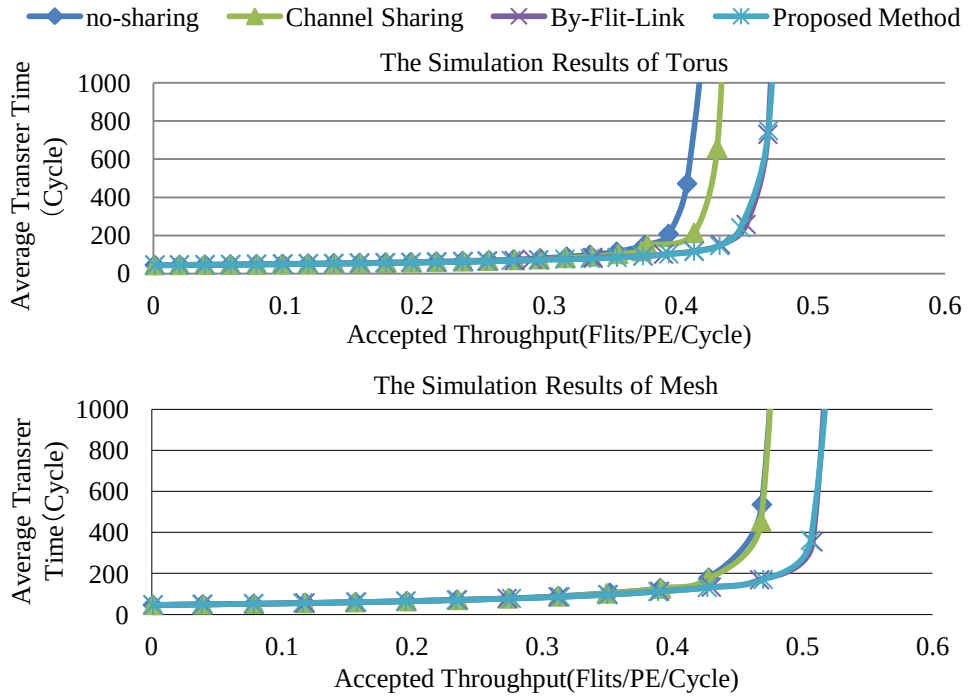


図 4-17 評価 2 のシミュレーション結果(16 PE, 64 Buffer, and 32 Flits/Packet)

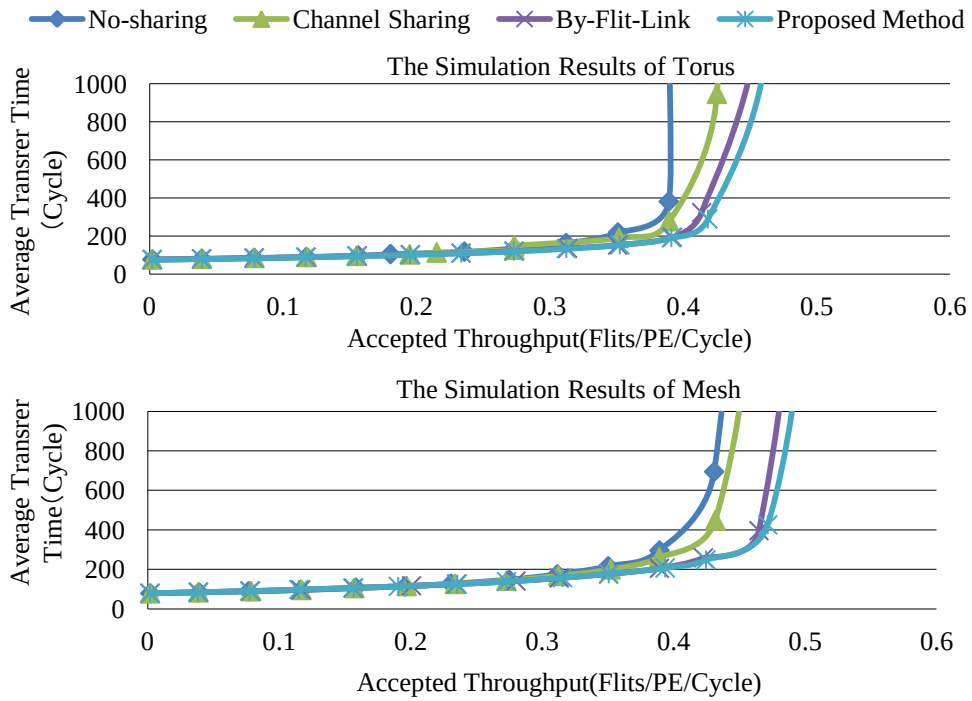


図 4-18 評価 2 のシミュレーション結果(16 PE, 64 Buffer, and 64 Flits/Packet)

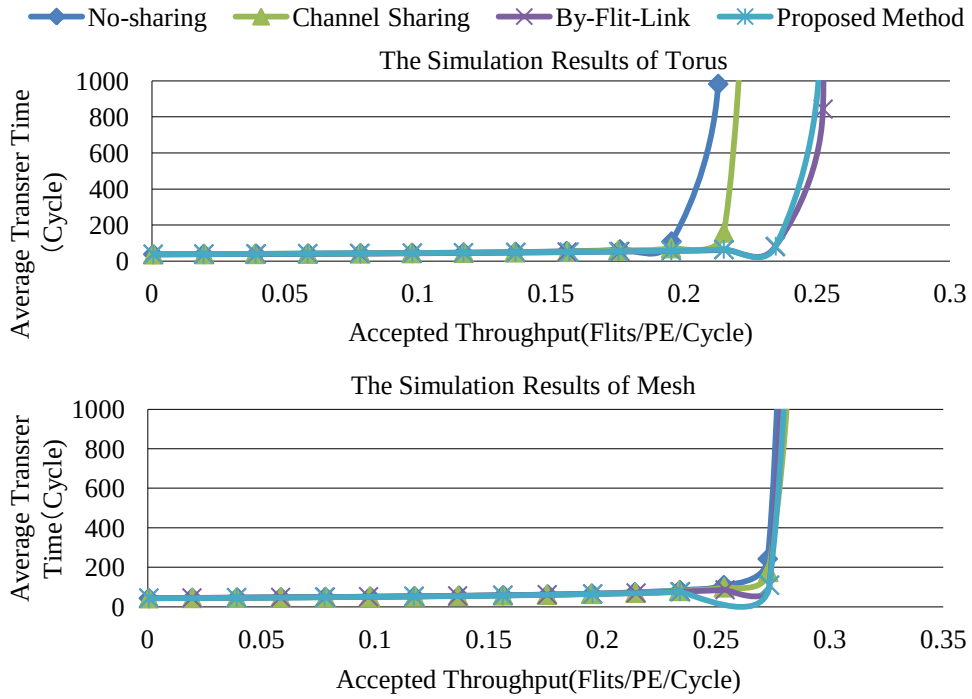


図 4-19 評価 2 のシミュレーション結果(64 PE, 32 Buffer, and 16Flits/Packet)

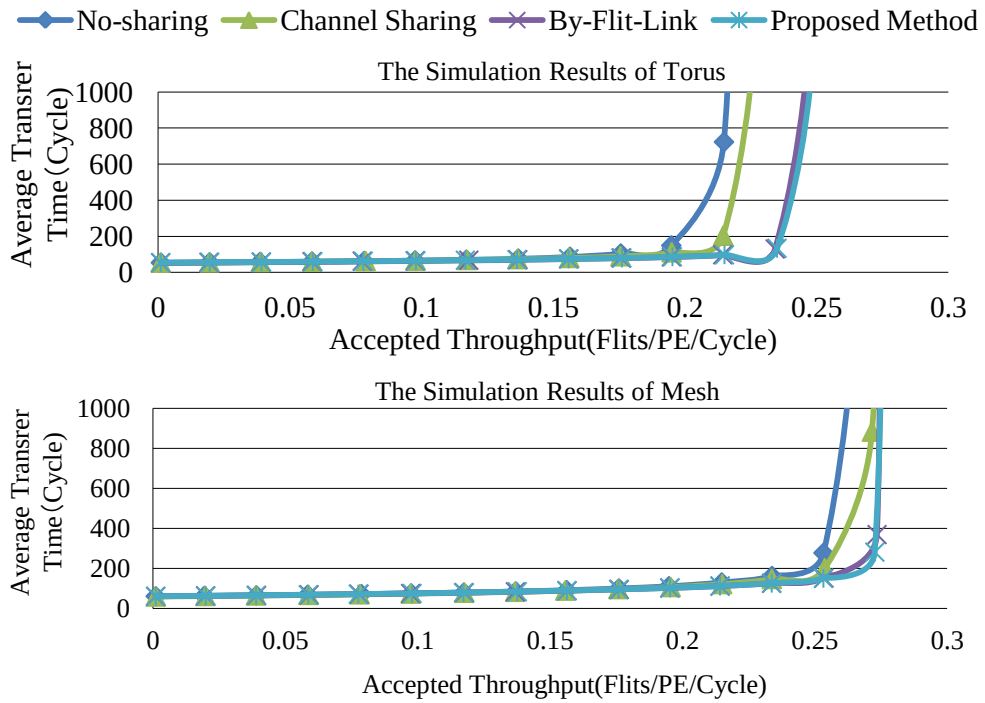


図 4-20 評価 2 のシミュレーション結果(64 PE, 32 Buffer, and 32Flits/Packet)

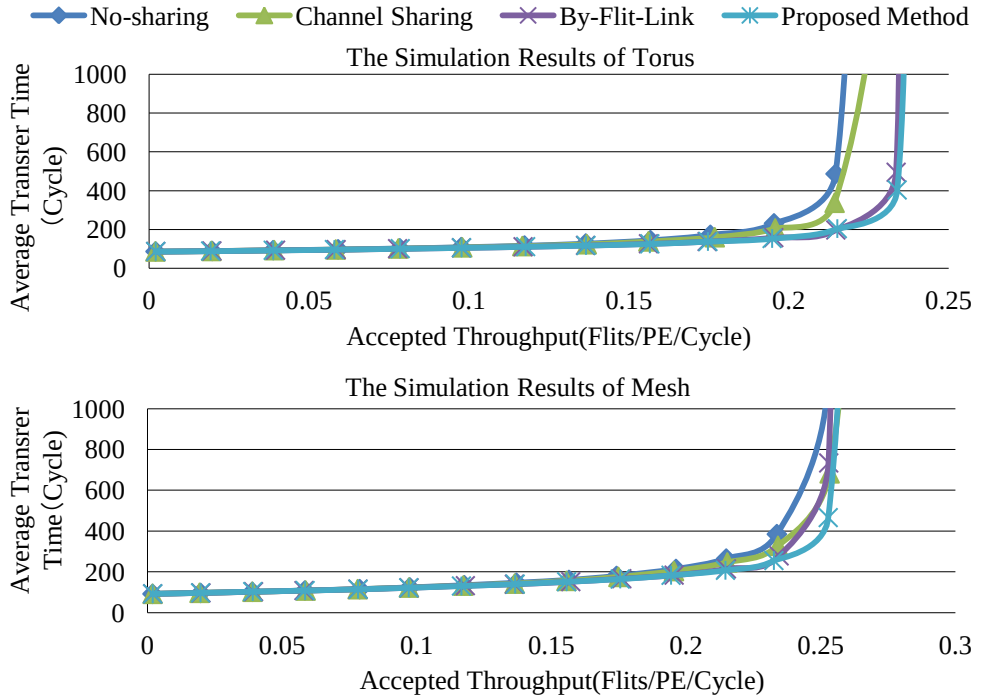


図 4-21 評価 2 のシミュレーション結果(64 PE, 32 Buffer, and 64Flits/Packet)

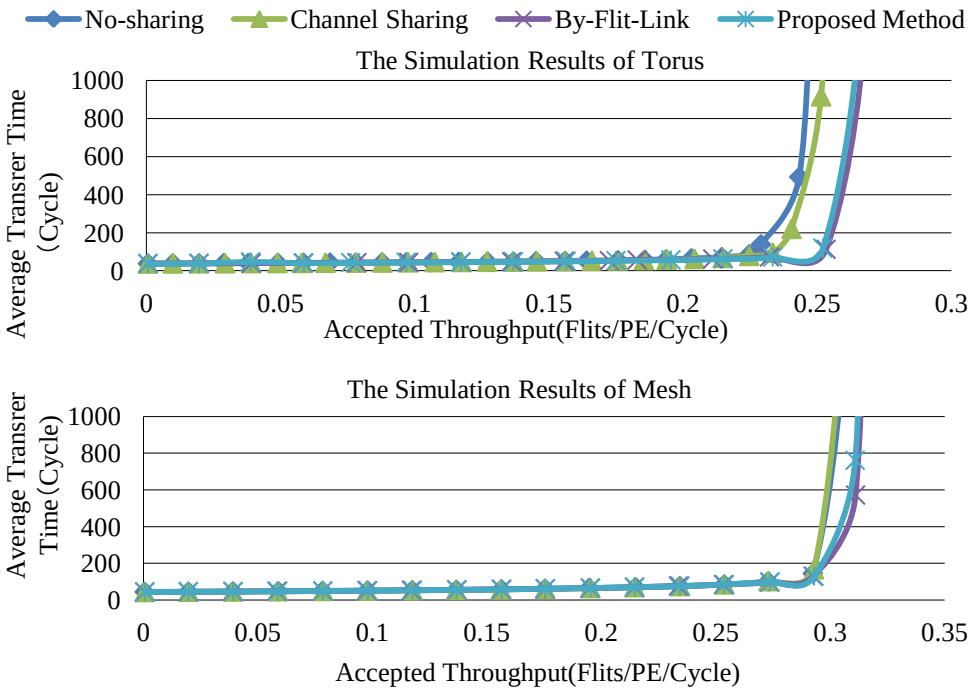


図 4-22 評価 2 のシミュレーション結果(64 PE, 64 Buffer, and 16Flits/Packet)

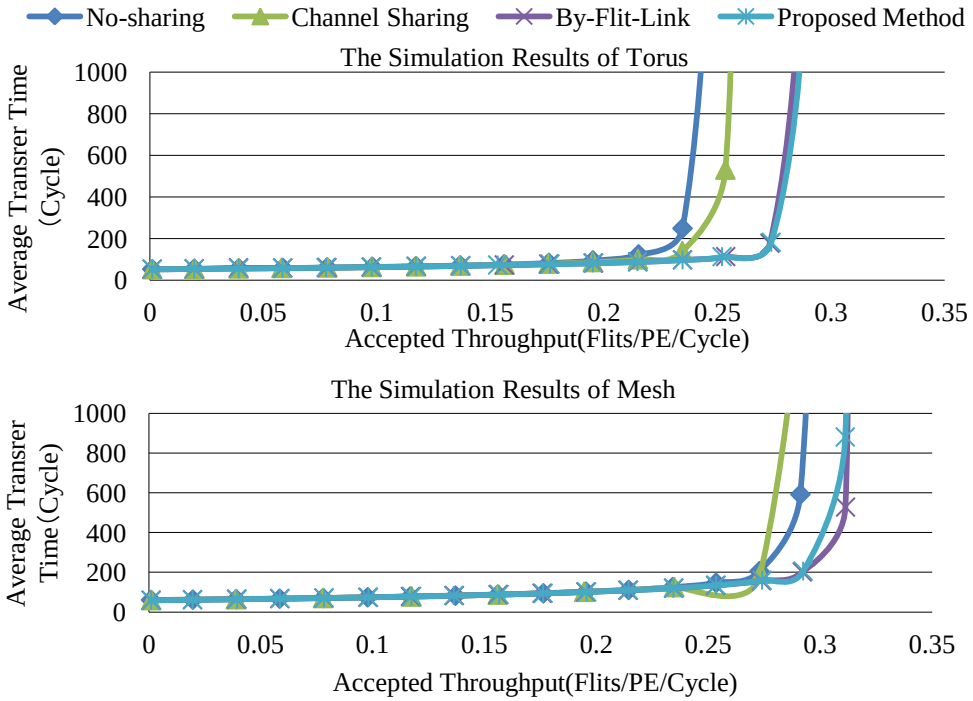


図 4-23 評価 2 のシミュレーション結果(64 PE, 64 Buffer, and 32Flits/Packet)

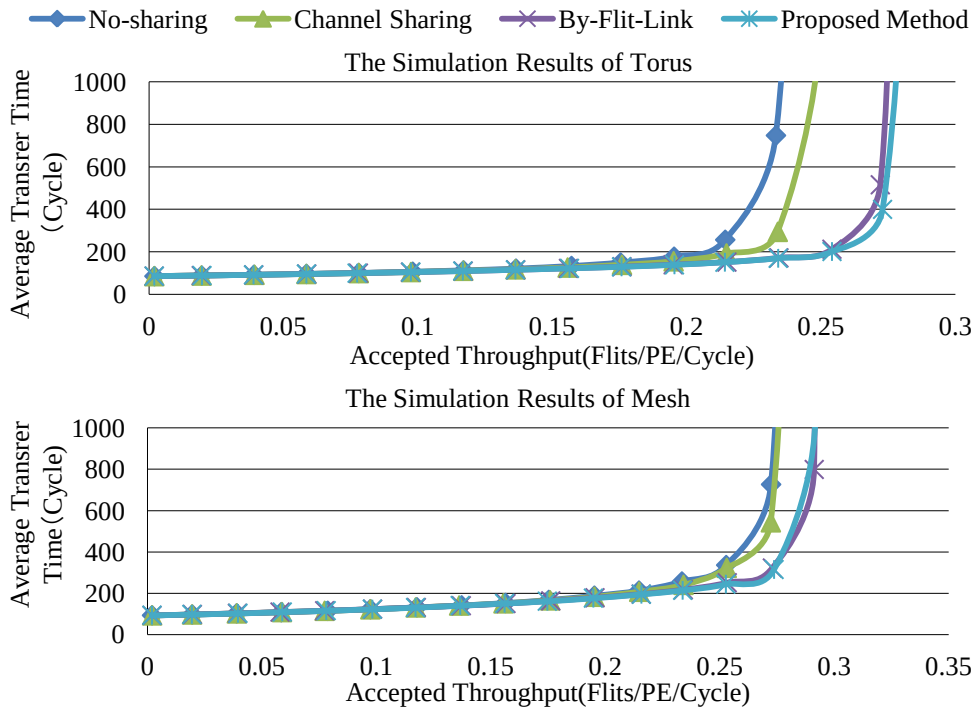


図 4-24 評価 2 のシミュレーション結果(64 PE, 64 Buffer, and 64Flits/Packet)

表 4-2　提案手法の性能上率

Topology	Number of PE	Total Buffer	Packet Length	**Progress Ratio (%)**
Torus	16	32	16	**11.5**
			32	**9.5**
			64	**2**
		64	16	**9.7**
			32	**12.4**
			64	**18.6**
	64	32	16	**17.9**
			32	**16.4**
			64	**9**
		64	16	**7.5**
			32	**16.4**
			64	**21.5**
Mesh	16	32	16	**9.5**
			32	**8.6**
			64	**6**
		64	16	**4.9**
			32	**8**
			64	**9.6**
	64	32	16	**2.5**
			32	**1.1**
			64	**4.2**
		64	16	**1.1**
			32	**6.9**
			64	**7.1**

4.4. まとめ

この章では，提案手法の通信性能をソフトウェアのシミュレータを使用して評価した．そしてブロック数に関する評価を行った結果，2 次元メッシュ，トーラスの場合，ブロック数が 8 以上であればブロック単位共有を行わなかった場合と同様の性能を持つことを確認した．様々な条件を用いた多面的な評価では，未共有やチャネル間共有に比べて性能が高く

なることが確認できた．また，次のような傾向があることも確認できた．

提案手法は，合計バッファ(フリット/ルータ)とパケット長(フリット/パケット)が近いとき，性能が上がる傾向にある．一方でパケット長が合計バッファより大幅に大きい，あるいはパケット長が合計バッファより非常に小さい場合，性能の向上幅は小さい傾向にある．また，PE 数の多いほどメッシュ網の向上率がトーラス網に比べて小さいことも確認できた．

第5章 ハードウェア構造とコスト

5.1. はじめに

この章ではまず，2 節で提案手法のハードウェア構造について紹介する．そして，3 節でトランジスタ数の算出式し，ハードウェアコストの評価を行う．また 4 節では，回路の一部である FreePool 回路の回路削減法について紹介する．

5.2. 各部の構造

5.2.1. バッファ本体，および周辺回路

物理リンク数が 4，ブロック数が 8 の場合のリンク間共有法におけるバンク型マルチポートメモリの構造を図 5-1 に示す．図 5-1 のように，バンク型マルチポートメモリは，メモリ本体であるブロック(バンク)と 2 つのクロスバスイッチから構成されている．ブロックは，入力と出力が同時に行える FIFO で構成されている. 図 5-1 の左のクロスバスイッチは入力データである In-data を各チャネルの入力先ブロックを記す In-block に示されたブロックと接続する．対して，右のクロスバスイッチは，各ブロックの出力をそれぞれの Out-block に示された各出力ポート Out-data と接続する．Out-block と Out-data はそれぞれ，各チャネルの出力ブロック，フリットの出力部となっている．

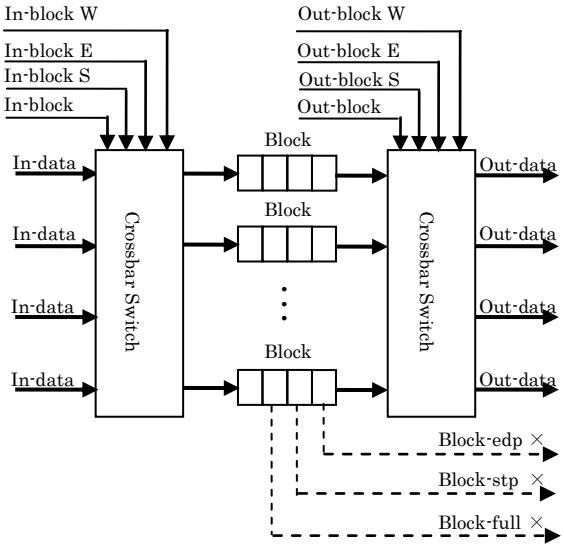


図 5-1 バンク型マルチポートメモリの構造

5.2.2. 制御情報を保持するメモリ要素

物理リンク数が 4，物理リンクごとの仮想チャネル数が 2 の場合の提案手法における，メモリの制御に必要な情報を保存するコントローラ内のポインタ類の構造を図 5-2 に示す. 提案手法のコントローラはメモリブロックの開放や取得を制御する回路で，図 5-2 のように，

全物理リンクで共通した一つの Free Pool と，チャネルごとに用意された Block Info，そして物理リンクごとに用意された In-req Control と Out-req Control によって構成されている．

図 5-2 の回路の動作は以下のようになる．ブロックが取得される際は，まず In-req Control にメモリ取得要求(In-req)と各リンクにおける各サイクルの動作仮想チャネル(Ctrl-channel)が入力され，各キューにメモリ取得のための信号が出力される．それによって Free Pool は先頭に格納されているポインタを動作仮想チャネルの Block Info に出力し，対象チャネルの Block Info はそのポインタを最後尾に格納する．逆にメモリ解放の際は，Out-req Control にメモリ解放要求(Out-req)と Ctrl-channel が入力され，各キューにメモリ解放のための信号が出力される．それによって対象チャネルの Block Info は先頭に格納されているポインタを出力し，Free Pool はそのポインタを最後尾に格納する．また，ブロック制御のため，各 Block Info の先頭と最後尾のブロックを表す Out-block，Tail と各物理リンクで次に取得されるメモリを表す In-address，そして動作仮想チャネルがブロックを有するかどうかを示す BI-empty を出力する．

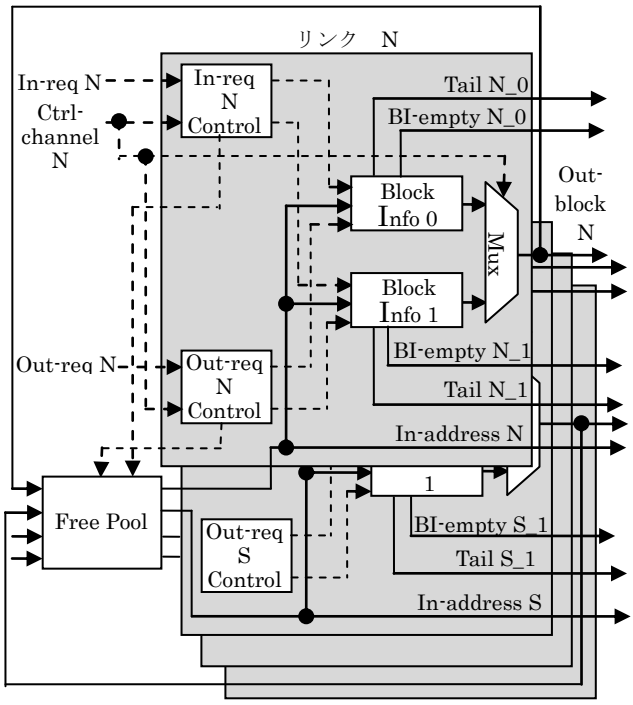


図 5-2 提案手法におけるコントローラの構造

リンク間共有法には，共有する物理リンク数の増大に伴って共有するメモリのサイズも増加するため，Free Pool や Block Info に要するハードウェアコストが増大するという問題もある．しかしながら，この問題におけるハードウェアコストの増大は，前述したブロック単位によるメモリ管理を行うことで，抑えることができる．これは，Free Pool や Block Info に要するキューのエントリ数をフリットサイズのメモリ数からブロック数まで削減することが可能になるためである．

5.2.3. ブロック制御用の論理回路

図 5-3 に，物理リンク 1 つ辺り(図 5-3 ではリンク N)の IJ ステージ回路を示す．IJ ステージは，入力フリットが共有メモリを使用する必要があるか否かを判定する．入力フリットが共有メモリに入力されるのは，専有部に空きがあり，かつ入力チャネルがブロックを取得していないとき以外である．この判定には，専有部に空きがあるかを示す PB-full と入力チャネルがブロックを取得しているかを示す Block Info の empty ビット(BI-empty)を用いる．図 5-3 の回路では，入力チャネルの PB-full と BI-empty を各リンクにおいてこのサイクルに動作する仮想チャネルを示す Ctrl-channel で選択し, 判定を行う. この回路では，PB-full が 0，BI-empty が 1 のとき(専有部に行く場合)のみ 0 が出力され，それ以外のとき(共有メモリに入力される場合)は 1 が出力される．

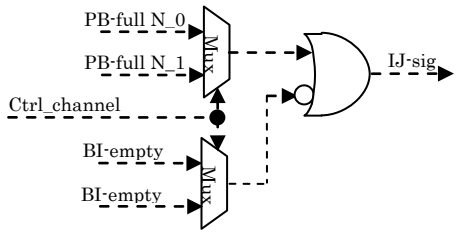


図 5-3 IJ 回路の構造

物理リンク一つ辺りのブロックへのフリット入力回路の構成を図 5-4 に示す．図 5-4 の回路は，フリットの入力に対し，ブロックの取得を行うか，またはブロック内にそのままフリットを入力するかを判定するためものである．

図 5-4 の左下のマルチプレクサは，入力処理を行う仮想チャネルが取得している最後尾のブロックを選択するものである．動作としては，物理リンクに含まれる各仮想チャネルの最後尾ブロックを指す Tail を， Ctrl-channel によって選択し，選択された Tail が指すブロックを示す Tail-block を出力する．

上部のマルチプレクサは，選択されたブロックに空きがあるかどうかを確認するものである．動作としては，各ブロックに空きがあるかを示す信号 Block-full を，左下のマルチプレクサの出力 Tail-block によって選択し，Tail-block が示すブロックに空きがあるかを示す Tail-full を出力する．この時，対象のブロックに空きがなければ，Tail-full をブロック取得要求 In-req として，図 5-2 の回路に送る．

右下のマルチプレクサは，フリットを入力するブロックを決定する部分である．動作としては，上部の出力 Tail-full によって，入力可能かを判定し，Tail-block あるいは Free Pool から出力される In-address に示されたブロックへのアドレスを出力し，In-block とする．In-address はこのサイクル内にブロックが取得された場合の取得ブロックを示している．

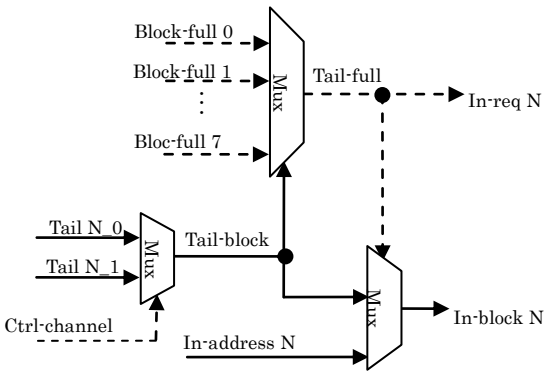


図 5-4 フリット入力回路の構造

物理リンク 1 つ辺りのブロック解放判定回路の構成を図 5-5 に示す．図 5-5 の回路は，チャネルが取得している先頭ブロックが各サイクルで空になるかを判定する回路である．ブロックのような FIFO が空になる条件は次のようなものになる．

1)ブロック内のフリットが残り一つである．

2)出力が行われる．

3)入力が行われない．

提案手法において 1)の条件は, 先頭ブロック内の入力アドレスを示すポインタ(Block-edp)と出力アドレスを示すポインタ(Block-stp)の位置関係によって判定される. 図 5-5 においてこの処理は上部の回路で行う. この回路は, 入力である各ブロックの Block-edp と Block-stp を，先頭ブロックを示す Out-block を用いて選択し，選択された情報により，先頭ブロック内のフリット数が 1 であるかを判定する．

先頭ブロックからのフリットの移動は，専有部に空きがある場合に行われるため，2)の条件は，ブロックを取得しているチャネルの専有部に空きがある場合に満たされる．図 5-5 においてこの処理は下部のマルチプレクサで行われる．このマルチプレクサは，各チャネルの専有部に空きがないかを示す PB-full を Ctrl-channel を用いて選択し，出力するものである．

先頭ブロックへのフリットの入力は，先頭ブロックが最後尾のブロックであり，かつ共有メモリへの入力がある場合に行われる．前者の条件は，先頭ブロックを示す Out- block と最後尾ブロックを示す Tail の値を比較することによって判定できる．後者の条件は， IJ ステップを行う回路(図 5-3)の IJ-sig を使用することで判定できる．3)の条件は，前述した場合以外であれば満たすことができる．図 5-5 においてこの処理は，中央のマルチプレクサと XNOR，NAND ゲートで行われる．これらの回路は，Ctrl-channel によって最後尾ブロックを示す Tail を選択し，XNOR によって選択した Tail と先頭ブロックを表す Out-block が同値であるかを判定する．そして，その判定結果と共有メモリへの入力信号である IJ-sig を NAND に入力することで 3)の条件を判定している．

図 5-5 の回路では上記した 3 つの条件の判定結果を，AND ゲートを通すことでブロック

を解放するかどうかを判定し，図 5-2 の回路に Out-req として出力する.

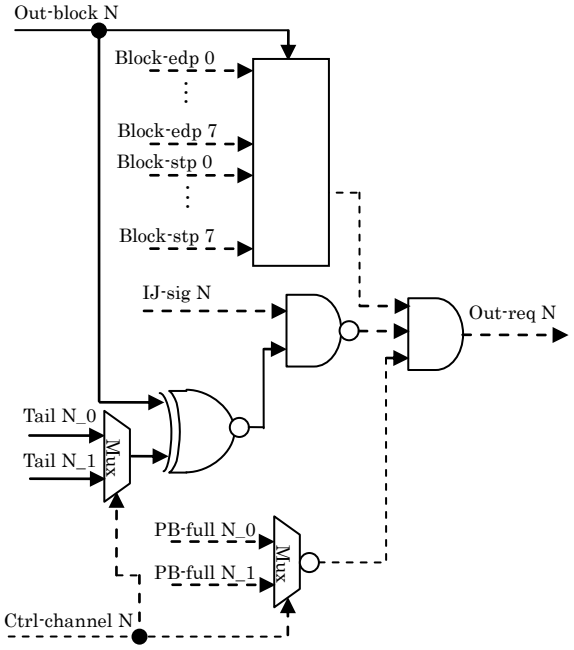


図 5-5 フリット出力回路の構造

5.3. パイプライン構造を加えたハードウェア構造

パイプライン構造を加えた提案手法のブロック図を図 5-6 に示す. 図 5-6 に示したように, 提案手法は 5 段のパイプラインステージを持つ. 各ステージは破線に囲まれたエリアです.

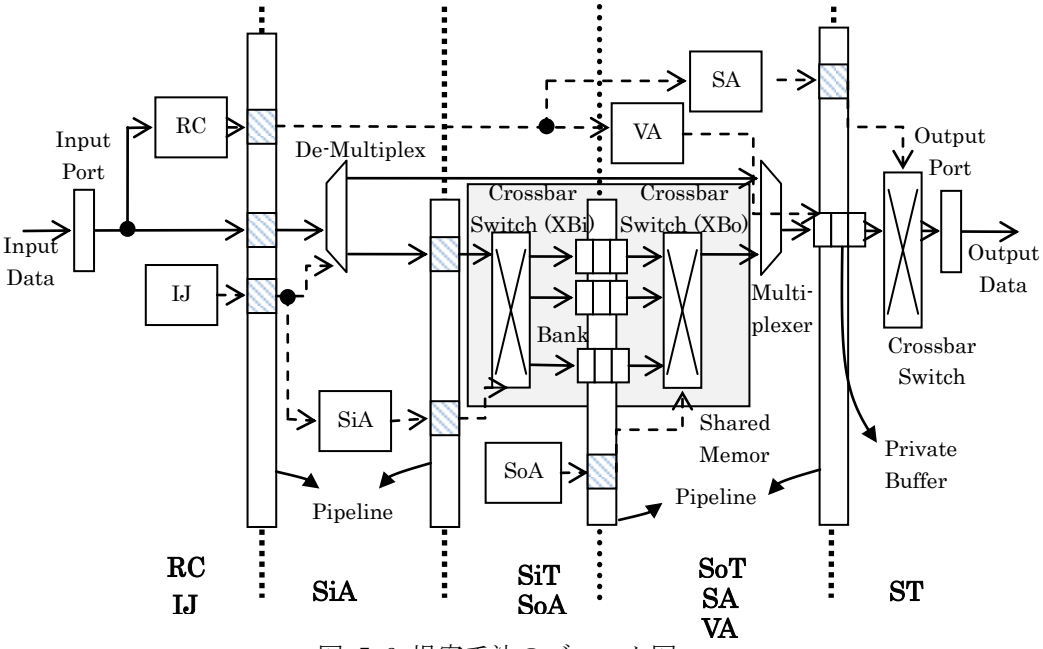


図 5-6 提案手法のブロック図

各ステージは，図中の長方形で示されるパイプラインレジスタや共有メモリと専有部のようなバッファによって区切られている．.経路 2 は図 5-6 の全てのステージを使用する．そして, 経路 1 を通るパケットは, 2, 3 個目のステージを除いた 3 つのステージを使用する．

提案手法の制御回路の構造を図 5-7 に示す．この手法では，メモリはフリープールとブロックインフォの 2 種類のキューによって制御されている．フリープールはチャネルに使用されていないブロックのポインタを格納するキューです．ブロックインフォは各チャネルに用意されます．これは，各チャネルに取得されたメモリのポインタを格納するキューです．チャネルがメモリを取得したとき，フリープールの先頭に格納されたポインタがその茶円るのブロックインフォのているに入力される．チャネルがメモリを解放したとき，Block info の先頭に格納されているポインタが Free Pool の最後尾に入力される．

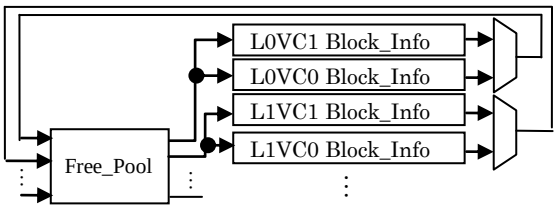


図 5-7 コントローラの簡略図

5.4. ハードウェアコストの算出と評価

以下に，提案手法の実装に要するハードウェアコストを，トランジスタ数を基準とした概算で見積もる．従来法においては，クロスバスイッチやルーティングの制御回路を除いた物理リンク本体に要するハードウェアの大部分が

a) バッファ本体

となる．また，物理リンク内の仮想チャネルでメモリを共有する手法(以後，これを従来法とする)と提案手法の双方で必要になるものとして,

b) 制御に必要な情報を保持するメモリ要素

が考えられる．上記の b)は，従来法と提案手法の双方で必要となるが，提案手法の実装方法次第で大幅に増大することが考えられる．さらに上記に加え，本手法により追加で必要になるものは,

c) ブロック制御用の論理回路

d) マルチポートメモリのポート周辺の回路

となる．上記 a)～d)により，物理リンク本体のハードウェアコストを，おおまかに見積もることができる．これらのうち a) d)は〈3･3･1〉節における「バッファ本体，および-周辺回路」，b)は〈3･3･2〉節における「制御情報を保持するメモリ要素」，c)は〈3･3･3〉節における「ブロック制御用の論理回路」に相当する．以降，これらについて順に，おおまか

な見積もりを述べる．

なお，本評価において，B, C, L, F, W をそれぞれ以下のように仮定する．

−B: 全物理リンクのメモリブロックの合計数

−C: 全物理リンクのチャネルの合計数

−L: 物理リンク数

−F: 1 ブロックあたりに入るフリットの数

−W: 1 フリット辺りのデータのビット数

このようにすると，1 リンクあたりのチャネル数は，C/L 個，リンク内共有(従来法)における，1 リンクあたりのメモリブロックの数は，B/L 個となる．また，フリット単位の共有法における見積もりを行う場合，F=1 とする．ハードウェアコストの評価に際しては，実装に要するトランジスタ数を想定し，メモリ要素は SRAM を想定して 6，n 入力 NAND(NOR)ゲートは 2n，インバータは 2，クロスポイントのスイッチはトライステートインバータを想定して 6 個のトランジスタを用いると仮定した．

5.4.1. バッファ本体

a)のバッファ本体に要するハードウェアコストは，各手法ともに同じで，

$$M_{buf} = B \times F \times W \tag{1}$$

bit となる．

5.4.2. 制御用のメモリ要素

b)の制御用のメモリのハードウェアコストは以下のようになる．

(ｉ) 従来法　　従来法は，Free Pool と Block Info という制御用のキューと，フリットを格納するメモリブロックに使用されるキューの 3 種類のキューから構成される．

Free Pool には，記憶部本体と，制御に必要な edp, stp，empty ビットが必要になる．記憶部本体は，1 リンク当たりのブロック数である B/L ブロックを表現し，B/L 個のエントリを持つため，Free Pool に必要なメモリ要素の数は

$$M_{tr,FP} = \frac{B}{L}\log\frac{B}{L} + 2\log\frac{B}{L} + 1 \tag{2}$$

bit となる．右辺第一項から，それぞれ，記憶部本体，edp と stp，empty ビットに要するメモリ要素の数である．

Block Info は，Free Pool と同様に計算できる．1 リンク当たりにチャネル数分(C/L 個)必要になるため，メモリ要素の数は以下のようになる．

$$M_{tr,BI} = \left(\frac{B}{L}\log\frac{B}{L} + 2\log\frac{B}{L} + 1\right)\frac{C}{L} \tag{3}$$

メモリブロックには，入出力場所を示すポインタである edp， stp とブロックに空きがあるかどうかを判断する full ビットが必要になる．edp， stp はブロックのエントリ数である 1 ブロック辺りのフリットの数を表現するため，それぞれ$\log F$ビットとなり，full ビットは 1 ビットとなる．1 リンク当たりにB/L個のブロックが必要になるため合計すると以下のようになる．

$$M_{tr,MB} = \frac{B}{L}\left(2\log F + 1\right) \tag{4}$$

上記より，従来法の制御用バッファの容量は合計して次のようになる．

$$\begin{aligned} M_{tr} &= \left(M_{tr,FP} + M_{tr,BI} + M_{tr,MB}\right)L \\ &= \left\{\begin{array}{l} \dfrac{B}{L}\log\dfrac{B}{L} + 2\log\dfrac{B}{L} + 1 \\ + \left(\dfrac{B}{L}\log\dfrac{B}{L} + 2\log\dfrac{B}{L} + 1\right)\dfrac{C}{L} \\ + \dfrac{B}{L}\left(2\log\ F + 1\right) \end{array}\right\} \times L \\ &= \left(\frac{B}{L} + 2\right)(L + C)\log\frac{B}{L} + 2B\log F + B + C + L \end{aligned} \tag{5}$$

ただし，従来手法の条件である$F = 1$のときにはメモリブロックに対するポインタを必要としないため，上式は，

$$\begin{aligned} M_{tr} &= \left(\frac{B}{L} + 2\right)(L + C)\log\frac{B}{L} \\ &\quad + 2B'\log F + B' + C + L \end{aligned} \tag{6}$$

となる．ただし，

$$B' = \begin{cases} 0 & (F = 1) \\ B & (F \neq 1) \end{cases}$$

とする．従来法では$F = 1$なので，$B' = 0$となる．

（ⅱ） 提案手法　　提案手法では，従来法と同様に Free Pool，Block Info，メモリブロックの 3 種類のキューが必要になる．

Free Pool の実装は，従来法と同様だが，B個のブロックを管理するキューが，全物理リンクに対して 1 個だけ必要となるので，Free Pool に必要なメモリ要素の数は以下のようになる．

$$M_{pr,FP} = B\log B + 2\log B + 1 \tag{7}$$

Block Info も同様であり，全物理リンクのチャネル数の合計分必要になるため，Block Info に必要なメモリ要素の数は以下のようになる．

$$M_{pr,BI} = \left(B\log B + 2\log B + 1\right)C \tag{8}$$

同様に，メモリブロックに必要なメモリ要素の数は以下のようになる．

$$M_{pr,MB} = B\left(2\log F + 1\right) \tag{9}$$

上記(7)-(9)を合計すると，全体に必要なメモリ要素の数は以下のようになる．

$$\begin{aligned} M_{pr} &= M_{pr,FP} + M_{pr,BI} + M_{pr,MB} \\ &= B\log B + 2\log B + 1 + \left(B\log B + 2\log B + 1\right)C + \left(2\log F + 1\right)B \\ &= \left(B+2\right)\left(C+1\right)\log B + 2B\log F + B + C + 1 \end{aligned} \tag{10}$$

(i)のケースと同様，B’を用いて，上式は，

$$\begin{aligned} M_{pr} = &\left(B+2\right)\left(C+1\right)\log B \\ &+ 2B'\log F + B' + C + 1 \end{aligned} \tag{11}$$

となる．

5.4.3. ブロック制御用の論理回路

c)のブロック制御用の論理回路は，以下のようになる．

1) Flit-Input Circuit(図 5-4)

2) Flit-Output Circuit(図 5-5)

3) IJ Circuit(図 5-3)

4)Free Pool 制御用のクロスバスイッチ(図 5-1)

となる．以後の評価において次のことを仮定する．

・2 入力マルチプレクサは 2 入力の NAND3 個とインバータによって構成される．ここでは，簡単のため 2 入力マルチプレクサを組み合わせて多入力マルチプレクサを構成する．

・AND(OR)ゲートは NAND(NOR)ゲートとインバータによって構成する．

・XOR(XNOR)は，2 入力の NAND3 個とインバータ 2 個によって構成し，b ビット XOR(XNOR)は b 個の 2 入力 XOR(XNOR)と b 入力の AND ゲートによって構成する．

各回路のトランジスタ数は，次のようになる．

1) Flit-Input Circuit に必要な回路とトランジスタ数の算出式はそれぞれ次のようになる．

・B入力 1 ビットマルチプレクサ

・C/L 入力 log B ビットマルチプレクサ

・2 入力 log B ビットマルチプレクサ

$$\begin{aligned} T_{FI} &= 14\left(B-1\right) + 14\left(\frac{C}{L} - 1\right)\log B + 14\log B \\ &= 14\left\{\left(B-1\right) + \frac{C}{L}\log B\right\} \end{aligned} \tag{12}$$

変形前の式(12)における第 1 項は「B入力 1 ビットマルチプレクサ」，第 2 項は「C/L 入力 log B ビットマルチプレクサ」，第 3 項は「2 入力 log B ビットマルチプレクサ」を示している．

2) Flit-Output Circuit に必要な回路は，以下のようになる．

・ブロック内の残りフリットが 1 かどうか確認する回路

－B 入力 logF ビットマルチプレクサ 2 個

－2 入力 logF ビット XNOR ゲート 3 個

－2 入力 1 ビット XOR ゲート logF – 1 個

－logF – 1，logF - 2 … 0 入力の AND ゲートが各 1 個

－インバータが 1 個

－2 入力 AND ゲート

－2 入力 OR ゲート

・C/L 入力 logB ビットマルチプレクサ 1 個

・2 入力 logB ビット XNOR ゲート 1 個

・2 入力 NAND ゲート 1 個

・C/L 入力 1 ビットマルチプレクサ 1 個

・3 入力 AND ゲート 1 個

したがって，必要なトランジスタ数の算出式は，次のようになる．

$$\begin{aligned} T_{FO} &= 28(B-1)\log F + 54\log F + 6 \\ &\quad + 16\log F - 16 + \sum_{i=2}^{\log F - 1}(2i+2) + 2 + 12 \\ &\quad + 14\left(\frac{C}{L} - 1\right)\log B + 18\log B + 2 + 4 \\ &\quad + 14\left(\frac{C}{L} - 1\right) + 8 \end{aligned} \tag{13}$$

$$\begin{aligned} &= 14(2B+3)\log F + 2(\log B + 1)\left(\frac{7C}{L} + 2\right) \\ &\quad + \sum_{i=2}^{\log F - 1}(2i+2) \end{aligned}$$

変形前の式(13)の第 1 項は「B 入力 logF ビットマルチプレクサ 2 個」，第 2，3 項は「2 入力 logF ビット XNOR ゲート 3 個」，第 4，5 項は「2 入力 1 ビット XOR ゲート logF – 1 個」，第 6 項は「logF – 1，logF - 2 … 0 入力の AND ゲートが各 1 個」，第 7 項は「インバータ」，第 8 項は「2 入力 AND ゲートと 2 入力 OR ゲート」，第 9 項は「C/L 入力 logB ビットマルチプレクサ 1 個」，第 10，11 項は「2 入力 logB ビット XNOR ゲート 1 個」，第 12 項は「2 入力 NAND ゲート 1 個」，第 13 項は「C/L 入力 1 ビットマルチプレクサ 1 個」，第 14 項は「3 入力 AND ゲート 1 個」を示している．ただし，第 6 項は F≦4 のときは使用しないため計算から除外する．

3) IJ Circuit に必要な回路とトランジスタ数の算出式はそれぞれ次のようになる．

・C/L 入力 1 ビットマルチプレクサ 2 個

・インバータ

・2 入力 OR ゲート

$$T_{IJ} = 14\left(\frac{C}{L} - 1\right) \times 2 + 2 + 6 = 4\left(\frac{7C}{\mathrm{L}} - 5\right) \tag{14}$$

変形前の式(14)の第 1 項は「C/L 入力 1 ビットマルチプレクサ 2 個」，第 2 項は「インバータ」，第 3 項は「2 入力 OR ゲート」を示している．

4)Free Pool 制御用のクロスバスイッチに必要な回路とトランジスタ数の算出式はそれぞれ次のようになる．

・logB ビット幅のクロスポイントが L×B 個

・デコーダが L 個

－$\log B$入力の AND ゲートがB個

－インバータが $\mathrm{logB} \times 2^{\mathrm{logB}}/2$ 個

$$T_{FPX} = \left\{ BL\log B \times 6 + \left(\begin{array}{c} (2\log B + 2)B \\ + \dfrac{2^{\log B}\log B}{2} \times 2 \end{array} \right) \times L \right\} \times 2 \tag{15}$$

$$= \left\{ \left(8B + 2^{\log B}\right)\log B + 2B \right\} \times 2L$$

変形前の式(15)の第 1 項は「logB ビット幅のクロスポイントが L×B 個」，第 2，3 項はそれぞれデコーダ内の AND ゲートとインバータを示している．

上記した結果より，ブロック制御用の組み合わせ論理回路の合計トランジスタ数は次のようになる．

$$T_{ctl} = \left(T_{FI} + T_{FO} + T_{IJ}\right)L + T_{FPX} \tag{16}$$

ただし 1)と 2)の回路は，F=1 の場合は使用しないのでその場合の合計トランジスタ数は次のようになる．

$$T_{ctl} = \left(T_{IJ}\right)L + T_{FPX} \tag{17}$$

5.4.4. マルチポートメモリの周辺回路

マルチポートメモリ周辺には，入力用と出力用に，それぞれL対BのWビット幅クロスバスイッチを要する．これに要するクロスポイントの数は，$L \times B \times W$個となる．また，交点スイッチを制御するために，入力用と出力用のそれぞれに対して，$\log B$ビットデコーダをL個必要とする．デコーダとして$\log B$入力の AND ゲートがB個，インバータが $\mathrm{logB} \times 2^{\mathrm{logB}}/2$ 個必要となる．したがって合計トランジスタ数は，およそ

$$T_{Mctl} = \left\{ 6LBW + \left(\left(2\log B + 2\right) B + \frac{2^{\log B}\log B}{2} \times 2 \right) \times L \right\} \times 2$$

$$= \begin{pmatrix} 6BW + 2B\left(\log B + 1\right) \\ + 2^{\log B}\log \mathrm{B} \end{pmatrix} \times 2\mathrm{L} \tag{18}$$

となる．

5.4.5. ハードウェアコストの評価結果

以上の結果より従来法と提案手法の合計トランジスタ数はそれぞれ式(19)，式(20)のようになる．

$$T_{trtotal} = 6\left(M_{buf} + M_{tr}\right) \tag{19}$$

$$T_{prtotal} = 6\left(M_{buf} + M_{pr}\right) + T_{ctl} + T_{Mctl} \tag{20}$$

(19)および(20)により計算される，L，C，B，F，W を変化させた時の，各実装形態のおおよそのハードウェアコストの合計を表 5-1，5-2 に示す．表中に示される数字はトランジスタ数である．なお，本評価では，リング網，および 2 次元と 3 次元のトーラス網の三種類の結合網を評価の対象とした．

表 5-1 は，従来法(Conventional Method)と，提案手法をフリット単位の共有法により実装したとき(By Flit and Link Sharing)の(すなわち，$F = 1$とした場合の)比較である．表 5-1 に示すように，従来法と提案手法のハードウェアコストの比は 5.001~14.3 と非常に大きなものとなる．この比率は，物理リンク数の多い結合網ほど(Lが大きいほど)大きなものとなる．また，バッファに格納できるフリット数が多い(Bが大きい)場合，若干高くなる．Lが大きいほどハードウェアコストが増大する理由は，Lが増えるに従って共有メモリの管理情報を保持する記憶素子の量や，マルチポートメモリを実装するためのクロスバスイッチなどの回路量が増大するためである．

表 5-1　提案手法の実装コスト(従来の実装法)

W	Topology	L	C	B	F	Conventional Method			By Flit and Link Sharing					
						Buffer	Control Memory	Total	Buffer	Control Memory	Control Logic	Memory Surround	Total	/Conventional Method
64	Ring	2	4	4	1	1536	180	1716	1536	390	392	6272	8590	5.006
				16		6144	1116	7260	6144	2190	2504	25472	36310	5.001
				64		24576	6156	30732	24576	11910	14408	103424	154318	5.021
	2D torus	4	8	8	1	3072	360	3432	3072	1674	2000	25280	32026	9.332
				16		6144	936	7080	6144	3942	5008	50944	66038	9.327
				64		24576	5256	29832	24576	21438	28816	206848	281678	9.442
	3D torus	6	12	24	1	9216	1404	10620	9216	10218	14232	115968	149634	14.09
				48		18432	3348	21780	18432	23478	33624	233856	309390	14.21
				96		36864	7884	44748	36864	53586	77784	471552	639786	14.3
128	Ring	2	4	4	1	3072	180	3252	3072	390	392	12416	16270	5.003
				16		12288	1116	13404	12288	2190	2504	50048	67030	5.001
				64		49152	6156	55308	49152	11910	14408	201728	277198	5.012
	2Dtorus	4	8	8	1	6144	360	6504	6144	1674	2000	49856	59674	9.175
				16		12288	936	13224	12288	3942	5008	100096	121334	9.175
				64		49152	5256	54408	49152	21438	28816	403456	502862	9.242
	3Dtorus	6	12	24	1	18432	1404	19836	18432	10218	14232	226560	269442	13.58
				48		36864	3348	40212	36864	23478	33624	455040	549006	13.65
				96		73728	7884	81612	73728	53586	77784	913920	1119018	13.71

バッファの量を同じ条件にして($B \times F = 64$に固定して)，ブロックの数を変えた(Bの値を変えた)時のハードウェアコストの比較を表 5-2 に示す．従来法を用いた場合，および提案手法をフリット単位の共有法により実装した場合のハードウェアコストを，表 5-2 中の「Conventional Method」および「By Flit and Link Sharing」の欄に示した．これらはいずれも，$F = 1$の場合の値となる．表 5-2 に示すように，提案手法のハードウェアコストは，ブロックの数を少なくしてバッファを少数ブロックにまとめるに従って(Bの値を小さくするに従って)ハードウェアコストが少なくなり，フリット単位で実装した場合(F=1)に比べて大幅に削減できることが分かる．これは表 5-1 でブロック数の合計が 64，96 個(B=64，96)の場合との比較からわかるように，「Control Memory」，「Control Logic」，「Memory

Surround」の各追加回路のハードウェアコストを，前述した各手法により削減できるためである．このうち，Memory Surround」の削減はバンク型マルチポートメモリ，「Control Memory」，「Control Logic」の削減はブロック単位共有によるものである．ブロック単位の共有法では，ブロック制御用の組み合わせ論理回路を必要とするものの，制御用のメモリや，マルチポートメモリのポート周辺回路のハードウェアコスト削減効果がそれを大きく上回る．今回の試算では，例えばチャネル数と同数以下のブロックで実装を行った場合，提案手法は従来法の 2 倍程度のコストで実装可能となる見積もりとなった．以上のように，バッファをブロックの単位で共有することにより，提案手法のハードウェアコストを大きく抑えることができる．

本評価では，提案手法のハードウェアコストの概算を，トランジスタ数により評価した．文献[16]では，バッファの入出力双方にクロスバスイッチを挟む構造を持つルータのレイアウト面積の評価をしており，実装コストが従来法の約 1.5 倍程度であったことから，本手法のレイアウト面積も今回の評価と大きく異なることはないと考えられるものの，より正確な評価のためには，実装を行い，それに伴うレイアウト面積の評価が必要になる．この点に関しては後日の課題となる．

文献[32]ではクロスバスイッチなどを使用したバンク型マルチポートメモリのトランジスタ数の評価を行っており，クロスバ方式はその中でもハードウェアコストが多い手法である．そのため今後，状況次第ではクロスバスイッチ以外を使用したバンク型マルチポートメモリについても検討を行う必要があると考えられる．また，クロスバ方式においても本手法の共有メモリに使用されるクロスバスイッチでは，調停処理の回路やバンク内アドレスのためのスイッチなどいくつかの回路を省略できる可能性があるため，ハードウェアコストについてはまだ改善の余地が残っていると考えられる．

表 5-2 提案手法の実装コスト（ブロック単位の実装）

W	Topology	L	C	B	F	Conventional Method	By Flit and Link Sharing	By block and Link Sharing(Proposed Method)					
								Buffer	Control Memory	Control Logic	Memory Surround	Total	/Conventional Method
64	Ring	2	4	16	4	30732	154318	24576	2670	5428	25472	58146	1.89203
				8	8			24576	1266	3228	12640	41710	1.35722
				4	16			24576	606	2040	6272	33494	1.08987
	2D torus	4	8	32	2	29832	281678	24576	9810	18992	102656	156034	5.23042
				16	4			24576	4422	10856	50944	90798	3.04364
				8	8			24576	2010	6456	25280	58322	1.95501
	3D torus	6	12	48	2	44748	639786	36864	24342	48240	233856	343302	7.6719
				24	4			36864	10938	26724	115968	190494	4.25704
				12	8			36864	4950	15276	57504	114594	2.56087
128	Ring	2	4	16	4	55308	277198	49152	2670	5428	50048	107298	1.94001
				8	8			49152	1266	3228	24928	78574	1.42066
				4	16			49152	606	2040	12416	64214	1.16103
	2D torus	4	8	32	2	54408	502862	49152	9810	18992	200960	278914	5.12634
				16	4			49152	4422	10856	100096	164526	3.02393
				8	8			49152	2010	6456	49856	107474	1.97533
	3D torus	6	12	48	2	81612	1119018	73728	24342	48240	455040	601350	7.3684
				24	4			73728	10938	26724	226560	337950	4.14094
				12	8			73728	4950	15276	112800	206754	2.53338

5.5. ハードウェアコスト削減手法

5.5.1. 制御回路

図 5-8 に提案手法における割り当て情報を記録する回路を示す．図 5-8 のようにこの回路は，一つの Free Pool と仮想チャネル数分の Block Info から構成されている．Block_Info は，各仮想チャネルに用意される FIFO で，対応した仮想チャネルに割り当てられたブロックのポインタを格納する．Free_Pool はすべての物理リンクで一つ用意される FIFO で，どのチャネルにも割り当てられていないブロックのポインタを格納する．

Free_Pool は複数の物理リンクから同時アクセスされる可能性があるため，マルチポート

FIFO によって構成する必要がある．しかし前述したようにマルチポートメモリはハードウェアコストが膨大である．そこで，共有メモリと同様に Free_Pool も複数の通常の FIFO をスイッチで接続して構成する．このとき，使用する FIFO の数は Free_Pool への同時アクセスの最大数である物理リンク数となる．

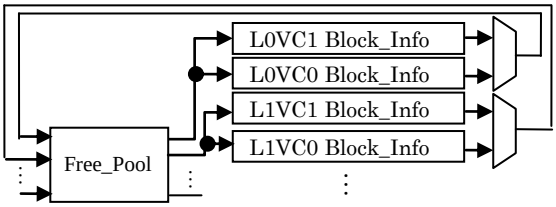


図 5-8 The Controller of Proposed Method

5.5.2. Free Pool のハードウェアコストの評価

表 5-3 に Free_Pool のトランジスタ数を示す．表 5-3 において，リンク数は 4，仮想チャネル数は 2，フリット長 8 ビット，そして共有メモリの容量は 64 となっている．また，削減効果を評価するため，制御回路内のほかのハードウェアコストも表 5-3 の最後の列に示す．表 5-3 の最初の列はブロック数である．

表 5-3 にあるようにブロック数が 8，16 の時，提案手法のトランジスタ数は単純にマルチポートメモリを使用した場合より小さくなることが分かる．また，この効果はブロックの数が大きいほど大きくなることも確認できる．

表 5-3 The number of transistor for Free_Pool

	Multi-port FIFO	Proposed Method	Others
Block4	512	512	512
Block8	1312	704	1536
Block16	3200	1088	4096

5.6. まとめ

この章では，提案手法の構造について紹介した．そして，それをもとにトランジスタ数を算出する式を計算し，ハードウェアコストを大まかに見積もった．その結果，チャネル数と同数以下のブロックで実装を行った場合，提案手法は従来法の 2 倍程度のコストで実装可能となる見積もりとなった．また，回路の一つである Free Pool 回路の削減策についても紹介した．その結果，コストを 1/2～1/3 に削減できることを示した．

第6章 より現実的な実装法に関する考察

6.1. はじめに

これまで，我々が提案した手法は，図 3-2 のようにすべての物理リンクで一つのメモリを共有する構成であった．このような手法を以後は「全リンク共有」とする．全リンク共有は，各種ハードウェアコスト削減手法を用いてその増大を抑えているが，実用にはまだ十分とは言えなかった．全リンク共有のような構成は，全ての回路や制御情報で全ての物理リンクを識別し，状態を保存する必要があるため，回路や処理の複雑化が大きな問題となる．それを軽減するために本章では，共有する範囲を二リンクずつとした実装について検討する．このような実装法を以後は「二リンク共有」とし，その構成を図 6-1 に示す．図 6-1 にあるように今回使用する二リンク共有においては，E と W，N と S のリンクでそれぞれ一つのメモリを共有する構成とする．これは次元順ルーティングにおいて，共有したリンクにそれぞれ入力されたパケットが同じリンクを待つパターンが若干ながら少ないためである．二リンク共有では，各リンクが管理するブロック数を減らすことができるため，制御情報のための記録素子やスイッチのサイズなどを削減でき，ハードウェアコストを大幅に抑えることができると考えられる．しかし，メモリの利用率は低下するため，通信性能の低下を招く可能性がある．

2 節では，二リンク共有のハードウェアコストを評価し，全リンク共有と比較する．3 節では，二リンク共有を用いることによる性能の低下を，全リンク共有と比較することで確認する．

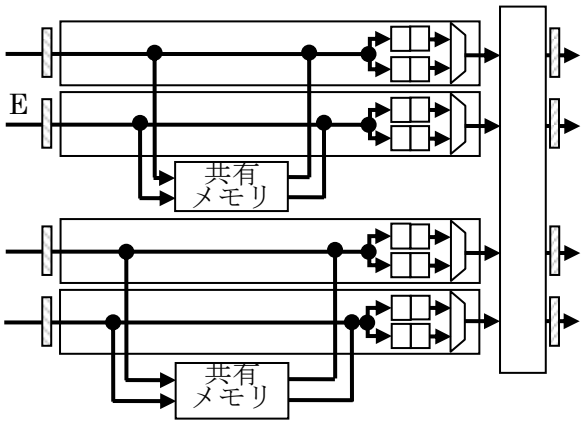


図 6-1 二リンク共有のルータ構造

6.2. ハードウェアコストの評価

本節では，以前の研究で提案手法のハードウェアコストを見積もるために使用した式[3]を用いて各実装のトランジスタ数を求め，評価する．このとき二リンク共有は，物理リン

ク数が 2，仮想チャネル数が 2 の場合の提案手法を 2 つ用意するものとして計算している．表 6-1 に結果を示す．表中の記号はそれぞれ以下の意味を表す．

−B: 全物理リンクのブロックの合計数

−C: 全物理リンクの仮想チャネルの合計数

−L:　物理リンク数

−F: 1 ブロックあたりに入るフリットの数

−W: 1 フリット辺りのデータのビット数

また表中の「バッファ本体」は実際にパケットを保存するバッファ，「制御用メモリ」は制御に必要な情報を保持するメモリ要素，「制御用論理」はブロック割り当て制御などに必要になる論理回路，「メモリ周辺」はマルチポートメモリ周辺の回路のトランジスタ数を表している．そして，最後の列の「増加率」に各実装形態の未共有と比較した場合のトランジスタ数の増加率を示している．また，二リンク共有のブロック数(B)の項目のかっこ内の数字は，ルータ全体の合計ブロック数で，非かっこの数値は二リンク共有の各共有部のブロック数となっている．

全リンクの部分では，ブロック数が減るほどにハードウェアコストが大きく減少しており，ブロック数が 8 以下ならば 2 倍以下で実装可能であることが分かる．二リンク共有では，全リンク共有にくらべてすべての項目でトランジスタ数が大きく減少しており，ブロック数が 4 つの時などでは，全リンク共有のブロック数が 2 の場合とほぼ同じハードウェアコストで実装できることが分かる．

表 6-1 各実装形態のトランジスタ数

W	手法	L	C	B	F	バッファ本体	制御用メモリ	制御用論理	メモリ周辺	合計	未共有	増加率
64	二リンク	4	8	8(16)	4	24576	2340	5368	25280	57564	29832	1.93
64		4	8	4(8)	8	24576	1116	3432	12544	41668	29832	1.40
64		4	8	2(4)	16	24576	516	2368	6224	33684	29832	1.13
64	全リンク	4	8	16	4	24576	4422	10856	50944	90798	29832	3.04
64		4	8	8	8	24576	2010	6456	25280	58322	29832	1.96
64		4	8	4	16	24576	918	4080	12544	42118	29832	1.41
64		4	8	2	32	24576	402	2800	6224	34002	29832	1.14
128	二リンク	4	8	8(16)	4	49152	2340	5368	49856	106716	54408	1.96
128		4	8	4(8)	8	49152	1116	3432	24832	78532	54408	1.44
128		4	8	2(4)	16	49152	516	2368	12368	64404	54408	1.18
128	全リンク	4	8	16	4	49152	4422	10856	100096	164526	54408	3.02
128		4	8	8	8	49152	2010	6456	49856	107474	54408	1.98
128		4	8	4	16	49152	918	4080	24832	78982	54408	1.45
128		4	8	2	32	49152	402	2800	12368	64722	54408	1.19

6.3. 通信性能の評価

評価にはソフトウェアシミュレータを使用する．評価では，通信パターンにユニフォームトラフィック，ルーティングアルゴリズムが次元順ルーティングである 2 次元トーラス網を使用する．

図 6-2 に全リンク共有におけるブロック数ごとの結果を示す．図 6-2 は PE 数が 64，バッファ総量が 32，パケット長が 16 ビットの場合の結果である．図中の「未共有」は共有を行わなかった結果，「フリット単位」は，全リンク共有は行うがブロック単位の共有を行わずに，メモリの割り当てをフリット 1 つ分のメモリ領域で行った場合の結果，B2，4，8 はそれぞれブロック数が 2，4，8 である提案手法の結果となっている．この結果より，ブロック数が 8 の提案手法は，ブロック単位共有を行わなかった場合と同等の性能を持つことが分かる．たいして，ブロック数が 2，4 のものは性能が若干落ちることが分かる．そのことから全リンクを共有した提案手法における性能上の最適なブロック数は 8 個であることが分かる．

続いて，二リンク共有におけるシミュレーション結果を図 6-3，6-4 および表 2 に示す．図と表中の「2 リンク B2，4，8」はそれぞれ二リンク共有の各共有部分のブロック数が 2，4，8 の場合の結果である．すなわち，ルータ全体でブロック数が 4，8，16 となる．また，

表 2 の各数値は，未共有と比較した場合の増加率で，単位はパーセントとなっている．結果より，二リンク共有ではシミュレーションに使用した範囲ではブロック数による性能の変化が小さいことが分かる．また，2 リンク B2，4 は B8 と比べると若干ながら性能が低下するが，B2，4 と比べると同等か若干高い性能を持つことが分かる．

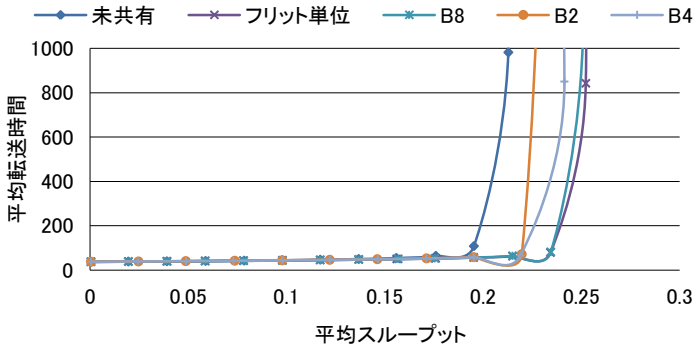


図 6-2 リンク共有のシミュレーション結果

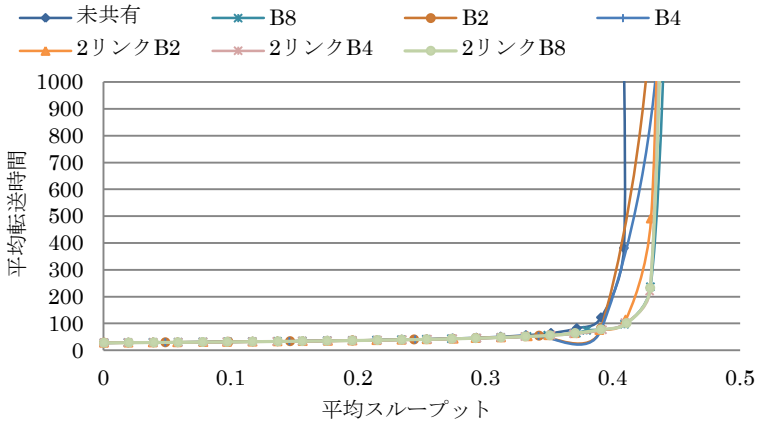


図 6-3 通信性能のシミュレーション結果（PE-No:16, Buffer:64, Packet:16）

表 6-2 各実装形態のトランジスタ数

W	手法	L	C	B	F	バッファ本体	制御用メモリ	制御用論理	メモリ周辺	合計	未共有	増加率
64	二リンク	4	8	8(16)	4	24576	2340	5368	25280	57564	29832	1.93
64		4	8	4(8)	8	24576	1116	3432	12544	41668	29832	1.40
64		4	8	2(4)	16	24576	516	2368	6224	33684	29832	1.13
64	全リンク	4	8	16	4	24576	4422	10856	50944	90798	29832	3.04
64		4	8	8	8	24576	2010	6456	25280	58322	29832	1.96
64		4	8	4	16	24576	918	4080	12544	42118	29832	1.41
64		4	8	2	32	24576	402	2800	6224	34002	29832	1.14
128	二リンク	4	8	8(16)	4	49152	2340	5368	49856	106716	54408	1.96
128		4	8	4(8)	8	49152	1116	3432	24832	78532	54408	1.44
128		4	8	2(4)	16	49152	516	2368	12368	64404	54408	1.18
128	全リンク	4	8	16	4	49152	4422	10856	100096	164526	54408	3.02
128		4	8	8	8	49152	2010	6456	49856	107474	54408	1.98
128		4	8	4	16	49152	918	4080	24832	78982	54408	1.45
128		4	8	2	32	49152	402	2800	12368	64722	54408	1.19

6.4. まとめ

本章では，提案手法のハードウェアコストを削減するため，共有を行う物理リンクを 2 リンクずつとした 2 リンク共有について検討した．その結果，2 リンク共有は，性能の若干の低下はあるものの，ハードウェアコストを大きく削減できることを示した．

第7章 まとめ

近年各回路をネットワークによって接続するネットワークオンチップ(NoC)が用いられるようになってきている．NoC で使用されるルータは，高性能であると同時に底面積，低電力であることも求められている．しかし，一般的なルータはメモリの使用効率が悪くなる可能性がある．従来から各チャネルのバッファを共有するさまざまな手法が提案されてきた．しかし，従来のバッファ共有法はハードウェアコストやパイプラインの段数の増加という問題があった．そこで本研究では，パイプライン段数の増加なしに，ハードウェアコストの増加を最小限に抑えたバッファの共有手法を提案し，通信性能とハードウェアコストの評価，実装に関する考察などを行った．

2 章では, NoC やルータ技術に関係する知識や, 一般的ルータや従来法について紹介した．そして，一般的ルータや関連研究の概要と問題点について簡単に説明した．

3 章では，提案手法の概要について解説し，ハードウェアコスト削減手法，パイプライン構造やデッドロック回避などについて説明した．ハードウェアコスト削減では，共有メモリにバンク型のマルチポートメモリを使用することで共有メモリのハードウェアコストを削減した．そして，複数のメモリ領域のまとまりであるバンク 1 つをブロックという単位とし，この単位で制御することで制御回路のコストを大幅に削減した．パイプライン構造では提案手法に 2 つの経路があり，共有メモリを使わない経路は 3 段，使う経路は 5 段となることを示した．そして，共有メモリを使う場合が専有部に空きがなくなったときであるため，専有部の容量が 2 回の通信ブロックを許容できる容量であれば遅延を隠ぺいできることも示した．リンク間の共有では共有メモリが全て他のチャネルに使われることでメモリを持てないチャネルが発生する可能性があった．そのため，チャネルを 2 つ以上使うことでデッドロックを防いでいるトーラス網などではデッドロックが問題となった．そこで，提案手法では各チャネルに専用のバッファである専有部を残すことによってチャネルが使用不能になることを防いだ．

4 章では，提案手法の回路について紹介しトランジスタ数を算出する式を計算し，ハードウェアコストを大まかに見積もった．その結果，チャネル数と同数以下のブロックで実装を行った場合, 提案手法は従来法の 2 倍程度のコストで実装可能となる見積もりとなった．また，回路の一つである Free Pool 回路の削減策についても紹介した．

5 章では，提案手法の通信性能をソフトウェアのシミュレータを使用して評価した．そしてブロック数に関する評価を行った結果，2 次元メッシュ，トーラスの場合，ブロック数が 8 以上であればブロック単位共有を行わなかった場合と同様の性能を持つことを確認した．また様々な条件を用いた多面的な評価では，未共有やチャネル間共有に比べて性能が高くなることが確認できた．また，次のような傾向があることも確認できた．提案手法は，合計バッファ(フリット/ルータ)とパケット長(フリット/パケット)が近いとき, 性能が上がる傾向にある．一方でパケット長が合計バッファより大幅に大きい，あるいはパケット長が合

計バッファより非常に小さい場合，性能の向上幅は小さい傾向にある．また，PE 数の多いほどメッシュ網の向上率がトーラス網に比べて小さいことも確認できた．

6 章では, 提案手法のハードウェア削減手法として部分的リンク共有法について紹介した．そして評価した結果，2 リンク共有は，性能の若干の低下はあるものの，ハードウェアコストを大きく削減できることを示した．

今後の展望として，ハードウェアについてはより詳しい回路を検討し，実装面積や消費電力などの評価を行うとともに，さらなるハードウェアコスト削減手法についても検討していく．通信性能については，シミュレータでなく数学的な評価も行う．

現在，必要な回路の大部分を VHDL ソースコードにより完成させている．これが全て完成すると提案手法を備えたルータ回路が出来上がり，実システムへの搭載が可能となる．将来，LSI の集積度が向上して，現在のものよりも，多数の PE を搭載したメニーコアプロセッサが登場した場合，従来のバス構造では通信能力の限界が来ることが予想される．その場合，よりスケーラビリティに優れた NoC が普及することが期待される．その際，本手法，およびこれまで作成した回路を用いることにより，NoC 向けルータ回路の通信キャパシティの向上が可能になり，当該領域の発展に寄与することが期待できると思われる．

謝辞

本研究の方針など，多大なご指導をいただきました三浦康之准教授に感謝いたします．また，論文やプレゼンテーションの資料などにおいて様々なご助言をいただきました渡辺重佳教授，ご多忙の中審査に立ち会っていただいた小林学教授，二宮洋教授に心よりお礼申し上げます.

参考文献

[1] 小柳光正, 杉村武昭, 福島誉史, 田中徹, 「3 次元集積化技術とリコンフィギャラブル 3D-Soc」, 電子情報通信学会技術研究報告, RECONF, リコンフィギャラブルシステム 106(49), pp.13-18, May. 2006.

[2] Michihiro Koibuchi, Kenichiro Anjo, Yutaka Yamada, Akiya Jouraku and Hideharu Amano, “A Simple Data Transfer Technique Using Local Address for Networks-on-Chips”, IEEE Transaction on Parallel and Distributed Systems, vol.17, No.12, pp.1425-1437, Dec. 2006.

[3] Yasushi Kanoh, Masaaki Nakamura, Tetsuya Hirose, Takeo Hosomi, Hirokazu Takayama and Toshiyuki Nakata, “Message Passing Communication in a Parallel Computer Cenju-4”, Proc. of 2nd International Symposium on High Performance Computing, pp.55-70, May. 1999.

[4] Yasuyuki Miura, Masahiro Kaneko and Susumu Horiguchi, “Examination of Hardware Implementation on Adaptive Routing for Hierarchical Interconnection Network TESH”, Proc. of International Workshop on High Performance and Highly Survivable Routers and Networks (HPSRN 2008), Mar. 2008.

[5] 金子昌弘，茂手木貴彦，三浦康之，渡辺重佳，「階層型相互結合網 TESH における適応型ルーティングのハードウェアコストに関する検討」，情報科学技術フォーラム講演論文集 8(1), pp.151-156, RC-006, Aug. 2009.

[6] Lionel M. Ni and Philip K. McKinley, “A Survey of Wormhole Routing Techniques in Direct Networks”, Computer, Vol. 26, No. 2, pp. 62-76, Feb. 1993.

[7] Yasuyuki Miura, Masahiro Kaneko, Shigeyoshi Watanabe, “Adaptive Routing Algorithms and Implementation for Interconnection Network TESH for Parallel

Processing", The 35th IEEE Conference on Local Computer Networks (LCN), pp.308-311, Oct. 2010.
[8] So Tran Cong, Shigeru Oyanagi, Katsuhiro Yamazaki, "Speculative Selection in Adaptive Routing on Interconnection Networks", 情報処理学会論文誌.コンピューティングシステム,Vol.44, pp.147-156, Aug. 2003.
[9] William J. Dally and Charles L. Seitz. "Deadlock-Free Message Rouring in Multiprocessor interconnection Networks". IEEE Transactions on Computers, Vol. C-36, No.5, pp.547-553, May. 1987.
[10] William J. Dally and Hiromichi Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels", IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No4, pp. 466-475, Apr. 1993.
[11] Eric Fleury and Pierre Fraigniaud, "A General Theory for Deadlock Avoidance in Wormhole-Routing Networks", IEEE Transaction Parallel and Distributed Systems, Vol. 9, No. 7, pp. 626-638, July 1998.
[12] W.J.Dally, "Virtual-Channel Flow Control", IEEE Transactions on Parallel and Destributed Systems, vol.3, No.2, pp.194-205, Mar. 1992.
[13] Amit Kumary, Partha Kunduz, Arvind P. Singh, Li-Shiuan Peh, Niraj K Jha, "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS", 25th International Conference on Computer Design(ICCD 2007), pp.63-70, Oct. 2007.
[14] Gregory L. Frazier, Yuval Tamir, "The design and implementation of a multiqueue buffer for VLSI communication switches", Proc. of the International Conference on Computer Design, pp.466-471, Oct.1989.
[15] Yuval Tamir, Gregory L. Frazier, "Dynamically-Allocated Multi-Queue Buffers for VLSI Communication Switches", IEEE Transactions on Computers, Vol.41, No.6, pp.725-737, Jun. 1992.
[16] R.S. Ramanujam, V. Soteriou, B. Lin and Li-Shiuan Peh, "Extending the Effective Throughput of NoCs with Distributed Shared-Buffer Routers", IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, vol.30, No.4, pp.548-561, Apr. 2011.
[17] Ali Ahmadinia and Alireza Shahrabi, "A Highly Adaptive and Efficient Router Architecture for Network-on- Chip", The Computer Journal, Vol.54, No.8, pp.1295-1307, Aug. 2011.
[18] 深瀬尚久，三浦康之，「直接結合ネットワークのルータ回路におけるバッファの有効利用」，全国大会講演論文集　第 72 回平成 22 年(1)，pp."1-165"-"1-166", 2M-2, Mar. 2010.
[19] 深瀬尚久，三浦康之，渡辺重佳，「直接結合ネットワークにおけるバッファのリンク単位共有法」，情報処理学会第 73 回全国大会，6H-1，Mar, 2011.
[20]Naohisa Fukase，Yasuyuki Miura，Shigeyoshi Watanabe，"Link-Sharing Method of Buffer in Direct-Connection Network"，The 2011 IEEE Pacific Rim Conference on

Communications, Computers and Signal Processing, pp.208-213, 2011.08.
[21] 深瀬尚久, 三浦康之, 渡辺重佳,「NoC ルータにおけるリンク間共有法の通信性能の評価」, 情報処理学会第 74 回全国大会, 5K-6, Mar. 2012.
[22] Naohisa Fukase, Yasuyuki Miura, Shigeyosi Watanabe, “The Hardware Cost Reduction Method of Control Circuit for Link-Sharing Method of Buffer in NoC Router”, 2013 RISP International Workshop on Nonlinear Circuits, Communications and Signal Processing, Mar. 2013.
[23] Naohisa Fukase, Yasuyuki Miura, Shigeyosi Watanabe, “The Proposal of Link-Sharing Method of Buffer in NoC Router : Its Implementation and Communication Performance”, Jounal of Basic and Applied Physics, Vol. 3, pp. 68-75, Feb. 2014.
[24] Y.Tatsumi, H.J.Mattausch “Fast quadratic increase of multiport-storage-cell area with port number”, Electronics Letters, Vol.35, No.25, pp.2185-2187, Dec. 1999.
[25] Michael Golden, Hamid Partovi, “A 500MHz write-bypassed, 88-entry, 90bit register file,” Proc. of Symposium on VLSI Technology, Session C11-1, 1999.
[26] H.J Mattausch, Koji Kishi and Takayuki Gyohten, “Area-efficient multi-port SRAMs for on-chip data-storage with high random-access bandwidth and large storage capacity” , IEICE Transaction Electronics, Vol.E84-C, No.3, p410-417, Mar. 2001.
[27] 井上他, 「K 出力可能な閉そく網と非閉そく網を階層的に用いたバンク型マルチポートメモリの構成と評価」, 電子情報通信学会論文誌 A, 基礎・境界, Vol.J89-A, No.10, pp.774-789, Oct. 2006.
[28] 佐々木他, 「オンチップマルチプロセッサ用共有キャッシュの実現方式の検討とその性能面積評価」, 電子情報通信学会論文誌 D-I, 情報・システム, I-情報処理, Vol.J87-D-I, No.3, pp.350-363, Mar. 2004.
[29]M.P.Merlin and J.P.Schweitzer, “Deadlock Avoidance in Store-and-Forward Networks-1: Store and Forward Deadlock”, IEEE Transactions on Communications, Vol. 28, No.3, pp.345-354, Mar. 1980.
[30]J.Duato, “A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks”, IEEE Transaction on Parallel and Distributed Systems, Vol.4, No.12, pp.1320-1331, Dec. 1993.
[31] W. J. Dally and B. Towles, Principles and Practices of InterconnectionNetworks, Jan. 2004.
[32]井上他,「閉そく網を用いたオンチップバンク型多ポートメモリの検討と回路規模の評価」, 電子情報通信学会論文誌 A, 基礎・境界, Vol.J88-A, No.4, pp.498-510, Apr. 2005.
[33] 三浦康之, 阿部亨, 堀口進, ワームホールルーティングにおける仮想チャネルフロー制御, 情報処理学会研究報告(98-HPC-74-11), pp.59-64, 1998.12.
[34] 三浦康之, 優先順位に基づく仮想チャネルフロー制御に関する研究 , 北陸先端科学技術大学院大学修士論文, 1999.03.

[35] 三浦康之，堀口進，Vijay K Jain，階層型ネットワーク TESH におけるデッドロック・フリー・ルーティング，情報処理学会論文誌，Vol.41，No.5，pp1370-1378，2000.5.
[36] Susumu Horiguchi, Yasuyuki Miura, Performance of Deadlock-Free Adaptive Routing for Hierarchical Interconnection Network TESH, Proc. of 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp.275-283, 2002.11.
[37] 三浦康之，堀口進，福士将，細粒度並列処理向け相互結合網 TESH における適応型ルーチングアルゴリズム，電子情報通信学会論文誌, Vol.J91-D, No.5, pp.1202-1215, 2008.5.
[38] M.M. Hafizur Rahman, Yasushi Inoguchi, Yukinori Sato, Yasuyuki Miura, Susumu Horiguchi, Dynamic Communication Performance of the TESH Network under Nonuniform Traffic, Journal of Networks, Vol.4, No.10, 2009.12, pp.941-951.
[39] Md Rabiul Awal, M. M. Hafizur Rahman, Rizal Bin Mohd Nor, Tengku Mohd Bin Tengku Sembok, Yasuyuki Miura and Yasushi Inoguchi, Wire Length of Midimew-connected Mesh Network, Proc. of the 11th IFIP International Conference on Network and Parallel Computing (NPC 2014), 2014.09.
[40] Yasuyuki Miura, Kentaro Shimozono, Kazuya Matoyama, Naohisa Fukase, and Shigeyoshi Watanabe, An Adaptive Routing Algorithm of 2-D Torus Network Based on Turn Model: The Communication Performance, International Journal of Networking and Computing (IJNC), Vol.5, No.1, pp.223-238, 2015.01.

研究業績

・論文

1) 深瀬尚久，三浦康之，渡辺重佳，直接結合網のルータ回路におけるバッファのリンク間共有法の提案，電気学会論文誌 C 分冊，Vol. 132(2012)，No. 10，pp.1675-1688, 2012.08., (3 章， 5.2, 5.4)

2) Naohisa Fukase, Yasuyuki Miura, M.M.Hafizur Rahman, and Shigeyoshi Watanabe, The Communication Performance of Link-Sharing Method of Buffer in NoC Router -The relation between the communication performance and the number of banks -, Transactions on Networks and Communications, Vol.1, No.1, pp.1-13, 2013.12., (4 章)

3) Naohisa Fukase, Yasuyuki Miura, and Shigeyoshi Watanabe, The Proposal of Link-Sharing Method of Buffer in NoC Router : Implementation and Communication Performance, Journal of Basic and Applied Physics (JBAP), Vol.3, No.1, pp.67-75, 2014.02. , (5.3, 5.5)

・国際会議

1) Naohisa Fukase, Yasuyuki Miura, and Shigeyoshi Watanabe, Link-Sharing Method of Buffer in Direct-Connection Network, Proc. of the 2011 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM 2011), 2011.08., (3.2, 5.2, 5.4)

2) Naohisa Fukase, Yasuyuki Miura, M.M.Hafizur Rahman, and Shigeyoshi Watanabe, The Performance Evaluation of Link-Sharing Method of Buffer in NoC Router, Proc. of 4th International Workshop on Advances in Networking and Computing, pp.567-571, 2013.12., (4 章)

3) Naohisa Fukase, Yasuyuki Miura, M.M.Hafizur Rahman, and Shigeyoshi Watanabe, The Proposal of Partial Sharing for Link-Sharing Method of Buffer in NoC Router, Proc. of 5th International Workshop on Advances in Networking and Computing, pp.567-571, 2014.12., (6 章)

・その他の論文

1) Yasuyuki Miura, Kentaro Shimozono, Kazuya Matoyama, Naohisa Fukase, and Shigeyoshi Watanabe, An Adaptive Routing Algorithm of 2-D Torus Network Based on Turn Model: The Communication Performance, International Journal of Networking and Computing (IJNC), pp.223-238, 2015.01.

・口頭発表

1) 深瀬尚久, 三浦康之, 直接結合ネットワークのルータ回路におけるバッファの有効利用, 情報処理学会創立 50 周年記念（第 72 回）全国大会， 2010.03

2) 深瀬尚久，三浦康之，渡辺重佳，直接結合ネットワークにおけるバッファのリンク単位共有法，第 73 回情報処理学会全国大会，2011.03.

3) 深瀬尚久, 三浦康之, 渡辺重佳, NoC ルータにおけるリンク間共有法の通信性能の評価, 第 74 回情報処理学会全国大会，2012.03.

4) 深瀬尚久，三浦康之，渡辺重佳，NoC ルータのためのリンク間共有法におけるパイプライン・ステージの検討，第 11 回情報科学技術フォーラム(FIT2012)，2012.09.

5) 深瀬尚久，三浦康之，渡辺重佳，ルータ回路のバッファ共有法における部分的共有法の性能評価，第 13 回情報科学技術フォーラム(FIT2014)，C-008，2014.09.