Vol.2011-ARC-196 No.4
2011/7/27

Distributed Shared-Buffer NoC ルータのためのパイプラインバイパス手法の改良

佐藤真平[†1,†2] 笹河良介[†1] 吉瀬謙二[†1]

近年プロセッサアーキテクチャはコア数の増加の一途を辿っている．コア数の増加に伴い，チップ内ネットワークのレイテンシ，スループットがアプリケーションに与える影響が大きくなってきている．Network on Chip (NoC) においては，低レイテンシ，高スループットなネットワークの開発が要求される．高スループットなルータとして，Distributed Shared-Buffer (DSB) NoC ルータが提案されている．このルータは，高スループットではあるが，パイプライン段数が多いため，通信遅延が大きい．本稿では，我々の提案している DSB ルータの遅延削減を目的としたパイプラインバイパス方式を改良し，さらなる遅延の削減を実現する手法を提案する．サイクルアキュレートなフリットレベルシミュレータを用いた評価から，提案手法は Zero Load Latency において DSB ルータに対して最大で 37.1%，平均で 36.7% のレイテンシの削減を達成した．

Enhancement of Pipeline Bypassing Method for Distributed Shared-Buffer NoC Router

Shimpei Sato,[†1,†2] Ryosuke Sasakawa[†1]
and Kenji Kise[†1]

†1 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology
†2 日本学術振興会 特別研究員（DC2）
Research Fellow of the Japan Society for the Promotion of Science (DC2)

1. はじめに

プロセス技術の進展により，1 チップに複数のコアが集積されるようになった．設計プロセスは依然として微細化傾向にあり，数 10 から 数 100 ものコアが搭載されるメニーコアプロセッサが実現可能になると考えられる．メニーコアプロセッサにおいて，コア間の通信は Network on Chip (NoC) と呼ばれる通信路を用いておこなわれる[1)]．NoC の設計では，レイテンシとスループットが重要なパラメタである．コア数の増加に伴い，ネットワークの性能がアプリケーションに与える影響が大きくなり，低レイテンシ，高スループットな NoC の需要が高まると考えられる．

NoC ルータアーキテクチャは，レイテンシ，スループットのどちらにも大きな影響を与え，そのネットワークの性能を決定する重要な役割を持つ．高スループットな NoC ルータとして Distributed shared-buffer (DSB) ルータ[2)] が提案されている．DSB ルータは，一般的な入力バッファの他に 2 つのクロスバーにはさまれたミドルメモリと呼ばれるバッファを搭載している．このミドルメモリを利用するために一般的なルータと比べてクロスバーが 1 つ追加された構成になっている．入力バッファにあるフリットは 1 つめのクロスバーを通り一度ミドルメモリに格納される，そしてミドルメモリから読み出されたフリットは 2 つめのクロスバーを通り隣接するルータに転送される．このミドルメモリを利用することで出力ポートにおけるフリットの衝突を回避し，高いスループットを実現できる．しかし，ミドルメモリの搭載とそれに伴うクロスバーの追加により制御が複雑になり，パイプライン段数が 5 段に増加しレイテンシが大きくなっている．

我々は，DSB ルータのパイプラインをバイパスし，レイテンシを削減する手法を提案している[3)]．この手法は，出力ポートにおけるフリットの衝突がない場合，1 つめのクロスバーとミドルメモリを使用せず，入力バッファから直接 2 つめのクロスバーにフリットを送る．これにより，パイプライン段数を 4 段にすることができる．また，バイパスが不可能な場合は 5 段のパイプラインを利用した転送になるため，従来の DSB ルータと比較してスループットの低下はない．

本稿では，これまでのパイプライン段数を 1 段削減するバイパス手法を改良し，2 段削減する手法を提案する．提案手法により，バイパス可能な場合はパイプライン段数を 3 段にすることができる．また，従来手法同様にパイプラインが不可能な場合のペナルティはなく，従来の DSB ルータと比較してスループットが低下しない．評価にはフリットレベルのサイクルアキュレートシミュレータを用い，提案手法の有効性を示す．

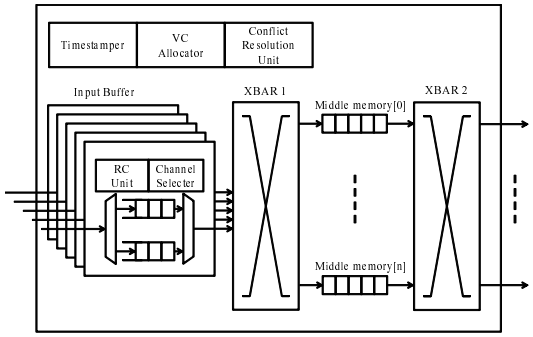

図 1 DSB ルータのアーキテクチャ

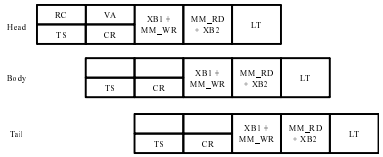

図 2 ルータのパイプライン

本稿の構成を以下に示す．2 章では DSB ルータについて説明する．3 章では従来のパイプラインを 1 段スキップするバイパス方式について説明する．4 章では提案手法であるパイプラインを 2 段スキップするバイパス方式について述べる．5 章で提案手法についてレイテンシ，スループットの評価をおこなう．6 章で関連研究について紹介する．最後に，7 章でまとめる．

2. Distributed Shared-buffer NoC ルータ

2.1 概　　要

高スループットな NoC ルータとして，Distributed Shared-buffer (DSB) ルータが提案されている[2)]．**図 1** に DSB ルータマイクロアーキテクチャを示す．入力チャネルごとに入力バッファが設けられている．フリットは入力バッファから読み出され，一旦ミドルメモリと呼ばれるバッファに格納される．ミドルメモリから読み出されたフリットは，第 2 クロスバーを経由し隣接するルータに転送される．ミドルメモリは 1 ポートのメモリで，1 サイクルに 1 フリットの書き込みと 1 フリットの読み出しをおこなう，このメモリを利用するために 2 つのクロスバーが必要になる．フリットを入力バッファからミドルメモリに格納することで，入力バッファの解放を早め，高いスループットを達成する．

図 2 に DSB ルータのパイプラインを示す．ミドルメモリとクロスバーの制御のためにパイプラインは 5 ステージ構成になっている．ルータに入力されたフリットは入力バッファに格納され，以下の処理を経て隣接するルータに転送される．

Route Computation (RC)

DSB ルータの RC ステージは Look-ahead ルーティングのため，出力先のルータの出力方向を計算する．一般的な Look-ahead ルーティングと同様の構成である．

Timestamping (TS)

DSB ルータの特徴的なステージである．このステージでは，各入力ポートからチャネルを 1 つ選択し，そのフリットがミドルメモリから読み出される予定時刻（タイムスタンプ）を計算し，付与する．フリットはこのタイムスタンプに従ってミドルメモリから読み出され，隣接するルータに転送される．異なる入力ポートでタイムスタンプを付与するフリットが同じ出力方向の場合は，予定時刻をインクリメントしながらタイムスタンプを付与する．この処理により，同じ時刻に同じ出力ポートを使用するフリットが存在しないことを保証する．このステージは DSB ルータの特徴的なステージで，パイプラインバイパス手法に関係するため詳細を後述する．

Virtual Channel Allocation (VA)

パケットに出力先の仮想チャネルを割り当てる．前のステージで選択されたチャネルを対象に，文献 4) で提案されている仮想チャネル割り当と同様の方式を用いる．

Conflict Resolution (CR)

ミドルメモリは 1 ポートのメモリであるため，同じタイムスタンプを付与されたフリットが同一のメモリに格納されることを回避する必要がある．このステージでは，ミドルメモリに書き込まれるフリットにタイムスタンプの衝突がないように適切にミドルメモリを割り当てる．

First crossbar traversal (XB1) + Middle memory write (MM_WR)

このステージで，フリットは 1 つめのクロスバー 1 を経由し，中間メモリに格納される．

Middle memory read (MM_RD) + Second crossbar traversal (XB2)

このステージで，フリットはミドルメモリから読み出され，2 つめのクロスバーを通過し，各出力ポートに出力される．

Link Traversal (LT)

隣接するルータにフリットを転送する．通常の Link Traversal と同様である．

パイプラインの 1 つめのステージで RC ステージと TS ステージ，2 つめのステージで VA ステージと CR ステージが同時に処理される．したがって，パイプラインは 5 段の構成になる．RC ステージおよび VA ステージはヘッドフリットの場合のみ処理される．

2.2 DSB ルータの Timestamping (TS) ステージ

TS ステージでは，フリットがミドルメモリから読み出され，第 2 クロスバーを通過する

時刻を計算し，フリットに付与する．この処理により，それぞれの出力ポートをフリットが通過する時刻を一意に決定し，出力ポートにおけるフリットの衝突を回避する．このステージは以下のような流れで処理される．

- 各入力ポートにおいてタイムスタンプを付与するチャネルを選択する．チャネルは LRU (Least Recently Used) により決定する．ただし，次のステージにおいて仮想チャネルの割り当てが失敗する可能性のあるチャネルは除外する．
- 選択されたチャネルの入力バッファの先頭にあるフリットに計算されたタイムスタンプを付与する．もし，先頭のフリットが前のサイクルにタイムスタンプを付与されたフリットであるならば，2 番目のフリットにタイムスタンプを付与する．
- 同じ出力ポートに転送される複数のフリット（入力ポートが異なる）にタイムスタンプを付与する場合，タイムスタンプをインクリメントし，それぞれのフリットに付与する．このときの優先度は固定で，ポート番号順にタイムスタンプが付与される．
- タイムスタンプを付与したフリットが次のステージにおいて，仮想チャネルの割り当て，もしくはミドルメモリの割り当てに失敗した場合，そのフリットは再び TS ステージを実行し，新たなタイムスタンプを付与する．入力バッファの先頭のフリットが第 2 ステージにおける割り当てに失敗し，そのときバッファの 2 番目のフリットが TS ステージを実行していた場合は，フリットの順序を維持するためにその処理をキャンセルする．

タイムスタンプの計算は次のようにおこなわれる．まず，出力ポート p に転送されるフリットがミドルメモリに格納されていない場合を考える．このとき，出力ポート p に転送されるフリットに新たに付与するタイムスタンプ $Timestamp$ は次の式で計算される．

$$Timestamp = Current\_Time + 3 \quad (1)$$

出力ポート p に転送されるミドルメモリにフリットが格納されていない場合，TS ステージを処理中の出力ポート p へ転送されるフリットがミドルメモリから読み出される時刻は 3 サイクル後なので，現在時刻（$Current\_Time$）に 3 を加えている．

次に，ミドルメモリに出力ポート p を使用するフリットが格納されている場合を考える．DSB ルータでは，出力ポートごとに最後に付与されたタイムスタンプの値が保持されている．出力ポート p の最後に付与されたタイムスタンプを $LAT[p]$ とあらわす．現在，TS ステージを処理中のフリットには，出力ポートの最後に付与されたタイムスタンプより遅い時刻を付与しなければならない．したがって，出力ポート p の最も早いタイムスタンプ $Timestamp$ は式 1 とあわせて次の式で計算される．

$$Timestamp = max(LAT[p] + 1, Current\_Time + 3) \quad (2)$$

同じ出力ポートを使用する複数のフリットに同時にタイムスタンプを付与する場合，入力ポート順の固定優先度に従ってタイムスタンプを付与する．この，出力ポート p を使用する複数のフリットに対して同時にタイムスタンプを付与する場合を考える．簡単のために，ミドルメモリには出力ポート p を使用するフリットが格納されていないと仮定する．同じ出力ポートに転送される複数のフリットには異なるタイムスタンプを付与する必要がある．DSB ルータでは，入力ポート i に対して付与するタイムスタンプについてオフセット値 $offset[i]$ を設定する．タイムスタンプのオフセット値は，同じ出力ポートを要求している入力ポートに対して固定優先度に従って連続した値である．一番優先度の高いフリットのオフセット値は 0 で，優先度順にインクリメントしていく．したがって，入力ポート i に付与されるタイムスタンプ $TS[i]$ は，次の式で表される．

$$TS[i] = Current\_Time + 3 + offset[i] \quad (3)$$

$offset[i]$ は，入力ポート i に設定されたオフセット値を表す．たとえば，入力ポート 4 のフリットが出力ポート p を要求し，他に出力ポート p を要求するフリットが存在しない場合，$offset[4]$ は 0 となる．また，入力ポート 3 のフリットが出力ポート p を要求し，他に入力ポート 0 と 1 のフリットが出力ポート p を要求している場合，$offset[3]$ は 2 となる．

これらをまとめて，出力ポート p を要求している入力ポート i のフリットに付与するタイムスタンプ $TS[i]$ は次の式で表すことができる．

$$TS[i] = max(LAT[p] + 1, Current\_Time + 3) + offset[i] \quad (4)$$

3. 従来の DSB ルータのパイプラインバイパス方式

3.1 提案手法

DSB ルータは NoC では典型的な Input Buffer Router (IBR) と比較して，高スループットであるがレイテンシの点で劣っている．レイテンシは，ルータのパイプライン段数に大きく影響されるため，パイプライン段数を削減することでレイテンシの改善が見込まれる．我々は，DSB ルータのパイプラインをバイパスすることでパイプライン段数を 1 段削減する手法を提案している[3]．

この手法は，DSB ルータの第 3 ステージである第 1 クロスバーの通過とミドルメモリへの書き込みをおこなわず，フリットを第 4 ステージである第 2 クロスバーの通過へ直に出力する．**図 3** に本手法を適用した DSB ルータのマイクロアーキテクチャを示す．入力ポートごとに第 1 クロスバーの手前から第 2 クロスバーの手前（ミドルメモリからの出力）

Vol.2011-ARC-196 No.4
2011/7/27

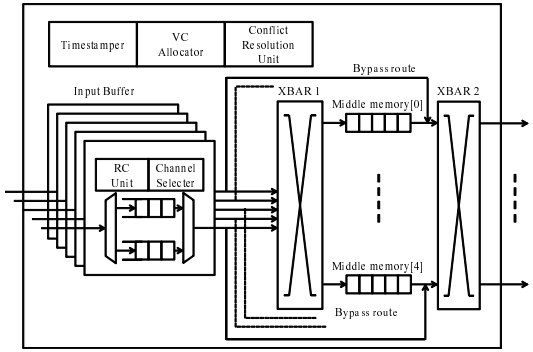

図 3 パイプラインをバイパスする DSB ルータのアーキテクチャ

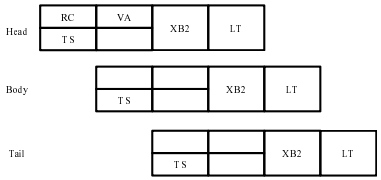

図 4 パイプラインをバイパスする DSB ルータのパイプライン

にフリットを転送するバイパス回路が設けられている．バイパス回路は各入力ポートに 1 つしかない．例えば，入力ポート 0 のバイパス回路は，入力バッファの出力から，Middle memory[0] の出力に接続されている．

ここで，ミドルメモリの数について制約を設ける．我々の提案手法は，各入力ポートにミドルメモリからの出力へのバイパス回路が追加されるため，ミドルメモリは入力ポート数分が必要である．例えば，メッシュネットワークの場合の入出力ポート数は 5 であるためミドルメモリは 5 個必要となる．文献 2) の評価において ミドルメモリの個数が 5 の場合と 10 の場合が比較されている．この評価から，ミドルメモリが 5 の場合と 10 の場合の性能差がほとんどないことが報告されている．以降は，簡単のためメッシュネットワークに限定して議論を進める．したがって，ミドルメモリの個数は 5 とする．

図 4 に，この手法によるパイプラインのバイパスが成功した場合のパイプラインの構成を示す．第 1 ステージの段階でフリットはバイパス可能かどうか判定され，バイパス回路を経由することでミドルメモリを利用せずに直に第 2 クロスバーに送られる．フリットがパイプラインをバイパスする場合，TS ステージでは必ず $Current\_Time + 2$ の値がタイムスタンプとして与えられる．このタイムスタンプにより第 3 ステージで第 1 クロスバーに出力するかバイパス回路に出力するかを決定する．

以上より，従来のパイプラインから 1 段削減することができ，隣接するルータにフリットを 4 サイクルで転送することができる．バイパス不可能な場合は，5 段パイプラインの DSB ルータと同様の転送手順となる．したがって，バイパスができない場合のペナルティは無く，DSB ルータと比較して性能の低下はない．

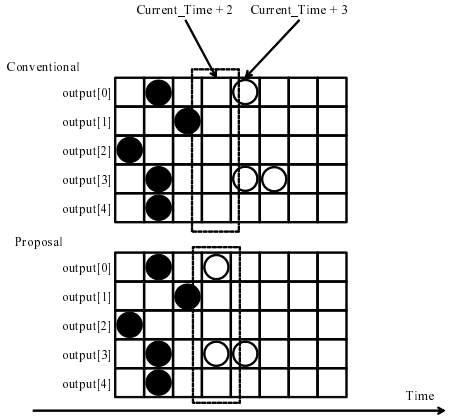

図 5 バイパス可能かどうかの判定例（成功）

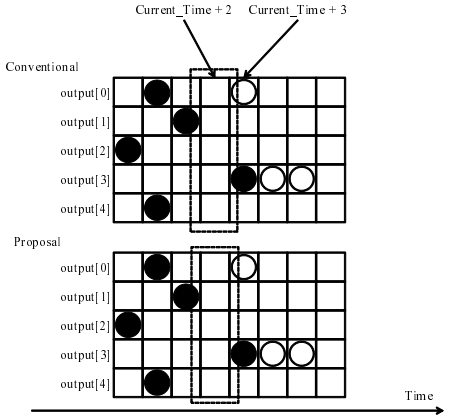

図 6 バイパス可能かどうかの判定例（失敗）

3.2 バイパス可否の判定と TS ステージへの変更

バイパス可能かどうかの判定は第 1 ステージでおこなう．フリットをバイパスする場合，そのフリットがミドルメモリから出力されるフリットと衝突しないことを保証する必要がある．フリットが格納されるミドルメモリが決定するのは第 2 ステージの ConflictReolution (CR) の処理である．したがって，第 1 ステージの段階であるフリットがバイパス可能かどうかの判定には，先行するフリットがどのミドルメモリに格納されるかが不明である．したがって，保守的な方針である $Current\_Time + 2$ のタイムスタンプを付与されたフリットが存在するかどうか，という条件を用いる．

$Current\_Time + 2$ のタイムスタンプを付与されたフリットが存在するかを判定するために，出力ポートごとに保持されている最後に付与されたタイムスタンプ（LAT）の値を利用する．この値より小さいタイムスタンプは発行された可能性があり，フリットが存在する可能性がある．逆に，この値より大きなタイムスタンプは発行されておらず，$Current\_Time + 2$ の方が大きい値である場合，バイパス可能である．

図 5, **図 6** を用いて，具体的に説明する．図 5, 図 6 は出力ポートごとに発行されたタイムスタンプの表である．図の黒丸はある出力ポート p（$output[p]$）における $LAT[p]$ を示しており，白丸が新たに発行されるタイムスタンプを示している．図の右にいくにつれてタイムスタンプの値は大きくなる．Conventional は DSB ルータにおける タイムスタンプ，Proposal はバイパス手法における タイムスタンプの様子である．図の例では，$output[0]$

を要求するフリットが 1 つ，$output[3]$ を要求するフリットが 2 つ存在している．

図 5 において DSB ルータ（Conventional）では，いずれの $LAT[p]$ よりも $Current\_Time$+3 の方が大きいため，新たに発行されるタイムスタンプは $Current\_Time$+3 以上の値となっている．これに対して，バイパスする DSB ルータ（Proposal）では，すべての $LAT[p]$ が $Current\_Time+2$ より小さいためバイパス可能と判定され，2 つのフリットに対して $Current\_Time+2$ のタイムスタンプが発行されている．

DSB ルータではタイムスタンプを発行されるフリットは異なる入力バッファのフリットであるため，バイパス回路の競合は発生しない．また，バイパス回路から直接第 2 クロスバーにフリットが転送されてもその時刻にミドルメモリから読み出されるフリットは存在せず，出力ポートの競合は発生しない．

図 6 は，バイパス不可能な場合の例である．$LAT[3]$ の値が $Current\_Time+2$ よりも大きな値となっている．この場合，$Current\_Time+2$ を付与されたフリットが存在し，フリットをバイパスしてしまうと同じタイムスタンプのフリットがミドルメモリからの出力で衝突するする可能性がある．したがって，このような状況ではバイパス不可能と判断され，従来の DSB ルータと同様のタイムスタンプが発行される．

3.3 パイプラインバイパスのために追加するハードウェア

パイプラインをバイパスするためのハードウェアとしては，第 2 ステージから第 4 ステージにフリットを転送する回路およびマルチプレクサ，デマルチプレクサがある．これらの追加によるハードウェアの増加は少ない．最もハードウェアの変更を必要とするのは TS ステージである．

前節より，パイプラインのバイパスを実現するには以下の 2 点が必要であることがわかった．

- バイパス可能かどうかは，すべての $LAT[p]$ が $Current\_Time+2$ より小さいかどうかで判定する．
- バイパス可能な場合は従来の DSB において発行されたタイムスタンプすべてを -1 した値をタイムスタンプとして発行する．

図 7 に，上に挙げた条件を実現する TS ステージのブロック図を示す．図中の点線で囲まれた範囲は従来の DSB ルータにおける TS ステージである．全ての $LAT[p]$（出力ポート p における最後に付与されたタイムスタンプ）と $Current\_Time+2$ の大小を比較器により比較し，その結果からバイパスするかどうかを決定する．バイパスする場合はデクリメンタにより計算された $TS[i]$（入力ポート i に付与されるタイムスタンプ）がすべて -1 さ

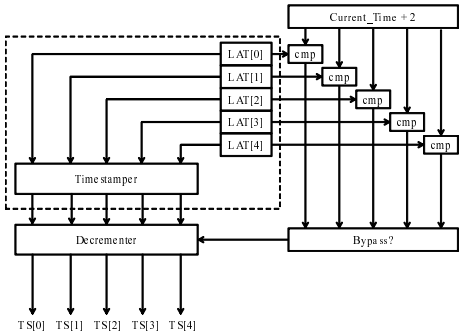

図 7 バイパスのために変更した TS ステージ

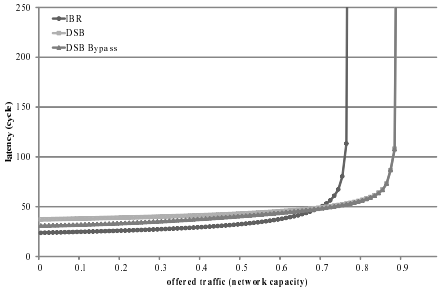

図 8 IBR と DSB ルータの性能比較（ランダム通信）

れる．

3.4 従来の DSB ルータのパイプラインバイパス方式の評価

IBR, DSB ルータ, 従来手法のパイプラインをバイパスする DSB ルータの性能を比較する．**図 8** は，それぞれのルータにおいて，パケットの注入レートを変化させたときのレイテンシの変化を示している．シミュレーションは，8×8 の 2 次元メッシュネットワークで，XY 次元順ルーティング，通信パターンは uniform[⋆1]，パケット長は 4 フリットの環境でおこなった．IBR は Look-ahead ルーティングと投機的なスイッチ割り当てをおこなう 3 段パイプラインの構成で，仮想チャネル数は 8，各入力バッファのサイズは 5 フリットとする．DSB ルータの仮想チャネル数は 5, 各入力バッファのサイズは 4 フリット，ミドルメモリは 5 個で，メモリのサイズは 20 フリットとする．IBR と DSB ルータのバッファサイズは合計で 200 フリットである．これらのパラメタは比較のため文献 2) の評価と同じ値を用いている．

注入レートが低い場合は DSB ルータに比べて IBR のレイテンシが小さい．注入レートが高い場合は，DSB ルータのスループットが IBR を上回っている．DSB ルータが IBR と比較して高いスループットを達成するが，レイテンシについては劣っていることが確認できる．パイプラインバイパスについては，注入レートが低い場合はフリットをバイパスし，レイテンシが削減でき，注入レートが高い場合は DSB ルータの高スループットを維持しており，期待する効果を達成している．

⋆1 各ノードがランダムな宛先のパケットを発行するパターン，ランダム通信．

4. DSB ルータのパイプラインバイパス方式の改良

4.1 概　　要

ルータのパイプライン段数を削減することで NoC におけるレイテンシの改善が見込まれる．本稿では，DSB ルータのパイプラインをバイパスすることで 1 段削減する従来手法を改良し，パイプライン段数を 2 段削減する手法を提案する．

従来手法において，フリットをバイパスする場合，そのフリットは選択されたチャネルのバッファの先頭に格納されている．あるサイクルにおいて，バッファの先頭にあるフリットをバイパス回路に送るか，クロスバーに送るかは，フリットに付与されているタイムスタンプが $Current\_Time + 1$ かどうかで判断する．したがって，もし第 1 ステージでバッファの先頭にあるフリットが選択され，$Current\_Time + 1$ のタイムスタンプを付与することができれば，そのフリットは従来手法のバイパス回路のままパイプラインを 2 段スキップできる．

従来手法では，図 4 からわかるように，ヘッドフリット以外は第 2 ステージの処理がなく潜在的にパイプラインを 2 段スキップすることができる．すべてのフリットについてパイプラインを 2 段スキップさせるには，ヘッドフリットの第 2 ステージで処理されている仮想チャネルの割り当て（VA）を第 1 ステージでおこなうことができればよい．

以上から，パイプラインを 2 段スキップさせるバイパス手法の実現には次の 2 点についてハードウェア変更が必要となる．

- 仮想チャネル割り当てを第 1 ステージでおこなう．
- タイムスタンプの発行において，バイパス可能な場合は $Current\_Time + 1$ の値をフリットに付与する．

4.2 仮想チャネル割り当ての変更

DSB ルータの第 1 ステージでは，はじめに入力ポートごとにタイムスタンプを付与するチャネルを選択する．選択したチャネルが第 2 ステージの処理に失敗するとそのチャネルのパイプラインはロールバックが発生し，発行したタイムスタンプが無駄になる．DSB ルータは高いスループットを維持するためにタイムスタンプを継続的に発行し，パイプラインのロールバックをできるだけ少なくする必要がある．

チャネルを選択するアービタにリクエストする際に，それが仮想チャネル割り当てが必要なフリットである場合は，そのフリットが要求する出力方向の割り当て可能な仮想チャネル数や，すでに他の入力ポートにおいて選択されたチャネルのうち，同じ出力方向を要求しているチャネルの数などの情報を元にそのフリットが次のステージでチャネル割り当てが可能かどうか判定し，アービタにリクエストするかどうかを決定する．これにより，第 1 ステージで選択されたチャネルは，必ず第 2 ステージにおける仮想チャネル割り当てが成功するようになっている．

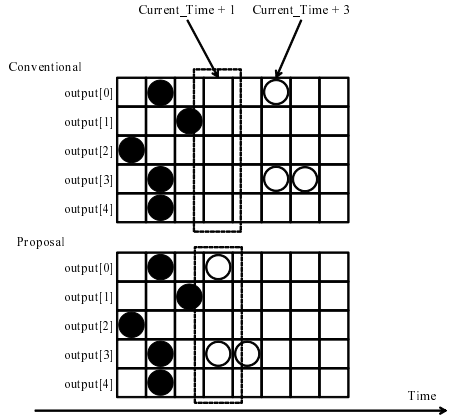

図 9　バイパス可能かどうかの判定例

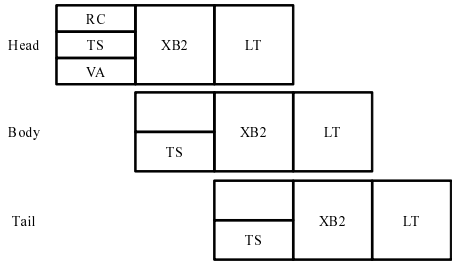

図 10　改良したパイプラインをバイパスする DSB ルータのパイプラインステージ

このため，DSB ルータの仮想チャネル割り当ては簡素化されており，第 1 ステージのチャネル選択後のタイムスタンプを計算する処理と並行して，仮想チャネル割り当てが可能であると考えられる．

4.3 バイパス可否の判定

仮想チャネルが第 1 ステージでおこなわれるようになると，TS ステージにおいて適切にタイムスタンプを発行することで，パイプラインを 2 段スキップすることができる．

第 1 ステージにおいて，あるフリットがバイパス可能かどうかの判定には，先行するフリットがどのミドルメモリに格納されるかが不明なため，$Current\_Time + 1$ のタイムスタンプを付与されたフリットが存在するかどうかという条件を用いる．従来手法と同様に，すべての $LAT[p]$ と $Current\_Time + 1$ を比較し，バイパス可能かどうかを判定する．TS ステージは図 7 に示した従来手法におけると同様の構成で，バイパスが可能な場合は，発行されるタイムスタンプ（$Current\_Time + 3$）を -2 する点が異なる．

図 9 を用いて，具体的に説明する．図 9 は出力ポートごとに発行されたタイムスタンプの表である．図の黒丸はある出力ポート（$output[p]$）における $LAT[p]$ を示しており，白

丸が新たに発行されるタイムスタンプを示している．図の右にいくにつれてタイムスタンプの値は大きくなる．図の例では，$output[0]$ を要求するフリットが 1 つ，$output[3]$ を要求するフリットが 2 つ存在している．Conventional は従来の DSB ルータにおける タイムスタンプ，Proposal は改良したバイパス手法における タイムスタンプの様子である．

図 9 において従来の DSB ルータではいずれの $LAT[p]$ よりも $Current\_Time + 3$ の方が大きいため，新たに発行されるタイムスタンプは $Current\_Time + 3$ 以上の値となっている．これに対して，バイパスする DSB ルータではすべての $LAT[p]$ が $Current\_Time + 1$ より小さいためバイパス可能と判定され，2 つのフリットに対して $Current\_Time + 1$ のタイムスタンプが発行されている．ここで，$output[3]$ を要求する 2 つめのフリットは $Current\_Time + 2$ のタイムスタンプが付与されている．

4.4 バイパス方式の改良のまとめ

改良手法と従来手法の異なる点は，まとめると次の 2 点である．

- 仮想チャネル割り当てを第 1 ステージでおこなう．
- TS ステージで $Current\_Time + 1$ のタイムスタンプを付与する．

バイパスは従来手法と同様に 1 つめのクロスバーの手前から，2 つめのクロスバーの手前までの回路を利用する．したがって，図 3 で示したアーキテクチャは従来手法と改良手法で同じである．

図 10 に，仮想チャネル割り当てを第 1 ステージに変更したパイプラインの構成を示す．第 1 ステージの段階でフリットはバイパス可能かどうか判定され，バイパス回路を経由することでミドルメモリを利用せずに直に第 2 クロスバーに送られる．フリットがパイプラインをバイパスする場合，TS ステージでは $Current\_Time + 1$ の値がタイムスタンプとして与えられる．このタイムスタンプにより第 2 ステージでバイパス回路を使用するかを決定する．これにより，DSB ルータのパイプラインを 2 段削減することができ，隣接するルータにフリットを 3 サイクルで転送することができる．

同じ出力ポートを要求する 2 つのフリットに同時にタイムスタンプが発行され，かつバイパス可能である場合は $Current\_Time + 1$ と $Current\_Time + 2$ のタイムスタンプがフリットに付与される．このとき $Current\_Time + 2$ が付与されたフリットは，パイプラインの第 2 ステージでは何も処理されずに第 3 ステージでバイパスの回路に転送される．バイパスのための経路が従来手法と同じなため，4 サイクルで隣接するルータにフリットを転送することができる．

バイパス不可能な場合も従来手法と同様に 5 段パイプラインの DSB ルータとして動作し，高いスループットは維持される．

表 1 チャネル数とバッファサイズ

Config.	Input buffer (per port)		Middle memory buffers		Total buffer
	#VCs	Flits/VC	MM	Flits/MM	
Input-buffered router	8	5	-	-	200
DSB router	5	4	5	20	200

5. 評　　価

5.1 評価方法

評価には，独自に開発したフリットレベルのサイクルアキュレートなソフトウェアシミュレータを用いる．このシミュレータは，文献 5) において開発されたシミュレータのネットワーク部分をベースに，パケットジェネレータを追加し，任意の通信パターンによる評価を可能にしたものである．シミュレータに 3 段パイプラインの IBR，DSB ルータ，パイプラインをバイパスする DSB ルータ，改良したパイプラインをバイパスする DSB ルータを実装し，スループットとレイテンシの比較から提案手法の有効性を示す．

評価環境は，8×8 の 2 次元メッシュトポロジ，XY 次元順ルーティングのネットワークを用い，パケット長は 4 フリットとする．**表 1** に，IBR および DSB ルータの 仮想チャネル数，バッファ量のパラメタをまとめる．DSB ルータと提案手法を適用した DSB ルータは同じパラメタを用いる．通信パターンは，代表的な uniform, complement, tornado の 3 種を用いる．シミュレーションは，ウォームアップとして 10,000 サイクルを実行した後に計測を開始し，100,000 サイクルまで実行する．これらのパラメタや評価方法は，文献 2) における評価方法に合わせている．

5.2 スループットとレイテンシの評価

図 11, **図 12** , **図 13** にパケットの注入レートを変化させたときの，IBR, DSB ルータ，従来のバイパスする DSB ルータ，バイパスを改良した DSB ルータのレイテンシを示す．バイパスを改良した DSB ルータのレイテンシは，注入レートが低い領域では，3 段パイプラインの IBR と同等のレイテンシを達成しており，バイパスの効果が確認できる．注入レートが高い領域では，DSB ルータとスループットは同じになり，バイパス回路を追加したことによるペナルティがないことが確認できる．

表 2 にバイパスを改良した DSB ルータの DSB ルータに対するレイテンシの削減率を示す．Zero load latency における削減率は，最大で 37.1%，平均で 36.7% を達成している．

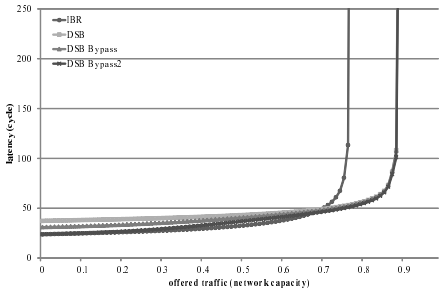

図 11 Uniform パターンの場合のレイテンシ

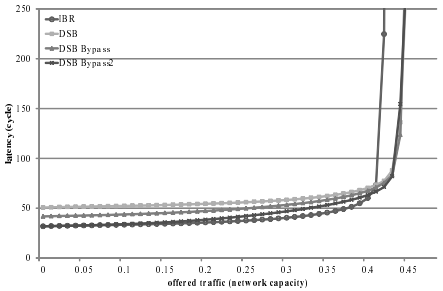

図 12 Complement パターンの場合のレイテンシ

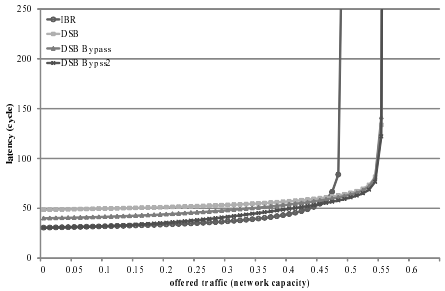

図 13 Tornado パターンの場合のレイテンシ

表 2 Zero Load Latency におけるレイテンシの削減率

traffic pattern	rate(%)
uniform	36.1
complement	37.1
tornado	37.0

6. 関 連 研 究

NoC ルータのパイプライン段数を削減し，低レイテンシを実現する研究は盛んに行われている．IBR における Look-ahead ルーティング[6]，および Speculative Switch Allocation[7] は，共にパイプライン段数を削減する技術である．DSB ルータの RC ステージは Look-ahead ルーティングである．また，CR ステージは IBR における SA ステージと同様の機能であり，DSB ルータでは Speculative Switch Allocation の技術も利用されているといえる．我々の提案するバイパス手法は，バイパスする場合は CR ステージの処理をおこなわないが，バイパス不可能な場合は CR ステージの処理をおこなうため，併用可能な技術である．

予測ルータ[8] は，入力されるパケットの出力方向を予測し，通常の経路計算や仮想チャネルの割り当てなどをすべてスキップして転送を行う．ルータにおける通常の処理とは別に予測による転送が行われるため，バイパスする DSB ルータにおいても併用可能な技術である．

Express VC[9] は，2 ホップ先までの方向が決定できる場合に，隣接しないルータ間で仮想的にバイパス経路を構成することで中継するルータにおける VA と SA ステージを削減する方式である．ただし，局所性を持つ通信パターンにおいてはレイテンシ削減の効果がない．我々の提案手法は経由する全てのルータにおいてレイテンシ削減の可能性がある点で異なる．

Token flow Control[10] は，近隣のルータ間でリソース情報を共有し，混雑がない場合にバイパス経路を形成し隣接ルータへの転送レイテンシを削減する．この方式は常に近隣のルータ間でリソース情報を共有する必要がある．我々の提案方式は，そのような情報を必要とせずにバイパスによりレイテンシを削減できる点で異なる．

Mad-postman[11] は，次元順ルーティングの規則性に着目し，ビットシリアル転送を行う場合にパケットヘッダが全て到着する前に隣接するルータに転送を開始することでレイテンシを削減する．文献 12) は，通信経路の使用頻度に着目し，頻繁に使われる経路にバイパスを構成する．文献 13) は，あらかじめ経路を固定した上でデータを転送する．NoC ルータを利用してサーキットスイッチング方式の転送を仮想的に構成することでレイテンシを削減する．これらのバイパス方式は転送ルートやルーティング方式などに依存するが，我々の提案するバイパス方式は転送ルートやルーティング方式に依存せずに適用できる点で異なる．

IBM Colony router[14] は，NoC ルータではないが DSB ルータと似たアーキテクチャであり，バイパス回路を持つ．このルータは共有メモリを搭載し，入力されたパケットは出力されるまでに 2 つのクロスバーを通過する．バイパス回路は，共有メモリへの読み書きを回避するように設けられている．パケットをバイパスする場合は，このバイパス回路上にあるクロスバーを経由して直接出力ポートに転送される．我々の提案方式はバイパスするための新たなクロスバーを必要としないため，ハードウェア量に制限のある NoC ルータに向けたバイパス回路になっている．

7. ま と め

プロセッサのコア数の増加に伴い，チップ内のネットワーク性能がアプリケーションに与える影響が大きくなると予想される．Network on Chip においては低レイテンシ，高スループットなネットワークの開発が求められている．レイテンシやスループットは NoC ルータのアーキテクチャが大きく影響している．典型的な NoC ルータとして Input-Buffered

ルータ（IBR）が広く知られている．この IBR より高いスループットを達成するルータとして Distributed Shared-Buffer ルータがある．しかし，DSB ルータのパイプラインは 5 段の構成なっており IBR に比べてレイテンシが大きいという欠点がある．

我々は，DSB ルータのパイプラインをバイパスし 1 段スキップすることでレイテンシを削減する手法を提案している．この手法では，トラフィックが低いときは 4 段パイプラインとして動作し，レイテンシの削減を達成する．本稿では，従来手法を改良し，パイプラインを 2 段スキップすることでレイテンシを削減する手法を提案した．改良手法は従来手法と同様にトラフィックが低いときはレイテンシを削減し，トラフィックが高いときは DSB ルータの高いスループットを維持する．DSB ルータと比較して，性能を低下させることなくバイパスのための要素を追加することができる手法である．

ソフトウェアシミュレータを用いた評価により，トラフィックが低いときは 3 段パイプラインとして動作し，トラフィックが高いときは，従来の 5 段パイプラインとして動作することを確認した．Zero Load Latency では DSB ルータに対して最大で 37.1%，平均で 36.7%のレイテンシ削減を達成した．

今後の課題としては次のようなものが挙げられる．通信パターンによる性能評価だけではなく，ベンチマークソフトを用いた性能評価をおこなう．提案手法におけるバイパス可否の判定は保守的なものとなっているが，ミドルメモリへフリットを格納するポリシーを変更し，より多くバイパスすることで性能を向上させる．ハードウェア記述言語を用いて提案手法を実装し，ハードウェア規模などを評価する．

参 考 文 献

1) Dally, W. J. and Towles, B.: Route packets, not wires: on-chip interconnection networks, *Proceedings of the Design Automation Conference*, pp.684–689 (2001).
2) Ramanujam, R. S., Soteriou, V., Lin, B. and Peh, L.-S.: Design of a High-Throughput Distributed Shared-Buffer NoC Router, *Proceedings of the Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pp. 69–78 (2010).
3) 姜　軒，佐藤真平，吉瀬謙二：Distributed Shared-buffer ルータの遅延を削減するパイプラインバイパス方式，情報処理学会研究会報告, 2011-ARC-194，No.13, pp.1–10 (2011).
4) Kumar, A., Kundu, P., Singh, A.P., shiuan Peh, L. and Jha, N.K.: A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS, *International Conference on Computer Design*, pp.63–70 (2007).
5) 植原　昂，佐藤真平，吉瀬謙二：メニーコアプロセッサの研究・教育を支援する実用的な基盤環境 (教育システム,¡特集¿システム開発論文)，電子情報通信学会論文誌. D, 情報・システム， Vol.93, No.10, pp.2042–2057 (2010-10-01).
6) Dally, W.J. and Towles, B.: Principles and Practices of Interconnection Networks (2004).
7) Peh, L.-S. and Dally, W.: A delay model and speculative architecture for pipelined routers, *Proceedings of the 7th International Symposium on High-Performance Computer Architecture (HPCA)*, pp.255–266 (2001).
8) 松谷宏紀，鯉渕道紘，天野英晴，吉永　努：低遅延オンチップネットワークのための予測ルータの評価 (組込みシステムプラットフォーム)，情報処理学会研究報告. EMB, 組込みシステム， Vol.2009, No.1, pp.1–6 (2009-01-06).
9) Kumar, A., shiuan Peh, L., Kundu, P. and Jha, N.K.: Express Virtual Channels: Towards the Ideal Interconnection Fabric, *in In Proceedings of 34th International Symposium on Computer Architecture (ISCA)*, pp.150–161 (2007).
10) Kumar, A., Peh, L.-S. and Jha, N.: Token flow control, *Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on*, pp.342–353 (2008).
11) Izu, C., Beivide, R. and Jesshope, C.: Mad-postman : A Look-ahead Message Propagation Method For Static Bidimensional Meshes, *Parallel and Distributed Processing, 1994. Proceedings. Second Euromicro Workshop on*, pp.117–124 (1994).
12) Park, D., Das, R., Nicopoulos, C., Kim, J., Vijaykrishnan, N., Iyer, R. and Das, C.: Design of a Dynamic Priority-Based Fast Path Architecture for On-Chip Interconnects, *High-Performance Interconnects, 2007. HOTI 2007. 15th Annual IEEE Symposium on*, pp.15–20 (2007).
13) Michelogiannakis, G., Pnevmatikatos, D. and Katevenis, M.: Approaching Ideal NoC Latency with Pre-Configured Routes, *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, pp.153–162 (2007).
14) Stunkel, C., Herring, J., Abali, B. and Sivaram, R.: A New Switch Chip for IBM RS/6000 SP Systems, *Supercomputing, ACM/IEEE 1999 Conference*, p.16 (1999).