ネットワークオンチップにおける仮想チャネル利用法の再考と評価

笹河良介[†] 藤枝直輝[†,☆]
高前田（山崎）伸也[†,††] 吉瀬謙二[†]

単一プロセッサ内のコア数は増加の一途を辿っている．メニーコアプロセッサでのコア間の通信には効率的なネットワークオンチップ (NoC) のデザインが必要である．ネットワークオンチップにおける高性能へのアプローチとしてネットワークトポロジやフロー制御，ルータアーキテクチャ，そしてルーティングがある．特にルーティングは以前から研究がおこなわれている．ネットワークオンチップでのルーティングにおいて，デッドロックフリーであることは重要であり，仮想チャネルをクラス分けすることによってデッドロック回避を提供する手法がある．本稿ではその手法に対して，さらに仮想チャネルを効率的に活用し，性能を上げることが可能な拡張的な手法を提案する．サイクルアキュレイトなフリットレベルシミュレータを用いた評価から，提案手法で性能が向上することが確認された．

Rethinking utilization of virtual channel for deadlock avoidance in Network-on-Chip

Ryosuke Sasakawa,[†] Naoki Fujieda,[†,☆]
Shinya Takamaeda-Yamazaki[†,††] and Kenji Kise[†]

The number of cores in a single microprocessor chip is increasing. Many-core processors need efficient Network-on-Chip (NoC) design for interconnection. In NoC, we can choose some approaches, such as network topology, flow control, router architecture, and routing algorithms, to improve the network performance. Routing algorithms have been especially researched for a long time. An important requirement of routing algorithm is the freedom from deadlock in NoC. For implementing deadlock-free routing algorithms, there is a method which divides virtual channels into classes. In this paper, we propose extended methods which additionally provide high performance and utilization of virtual channels. By using cycle-accurate NoC simulator, we confirmed that proposed methods improve performance.

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，数百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．その様な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．コア間の通信を担う NoC に対して，高いネットワーク性能の要求は高まると考えられる．

† 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
†† 日本学術振興会 特別研究員（DC1）
JSPS Research Fellow
☆ 現在，豊橋技術科学大学 電気・電子情報工学系
Presently with Department of Electrical and Electronic Information Engineering, Toyohashi University of Technology

性能に対するアプローチとしては，トポロジ，ルータアーキテクチャ，ルーティングなどが考えられる．その中において，ルーティングは以前から多くの研究がなされており，特にデッドロックフリーであることは重要である．しかし，適応型ルーティングといった高性能のルーティングはデッドロック回避を施さない場合，デッドロックを引き起こす可能性がある．**図 1** で示されている (a) がこのようなデッドロック回避が必要なルーティングである．

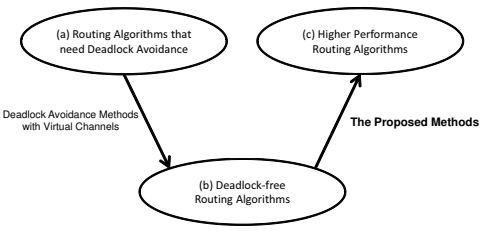

図 1 提案手法の立ち位置

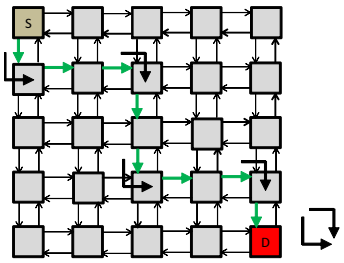

図 2 Minimal fully adaptive routing の例

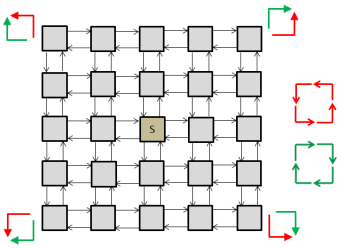

図 3 各方向の受信ノードへ転送するために必要なターン

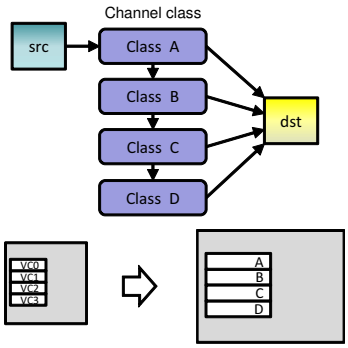

図 4 複数クラスを用いた時のパケットの転送イメージ

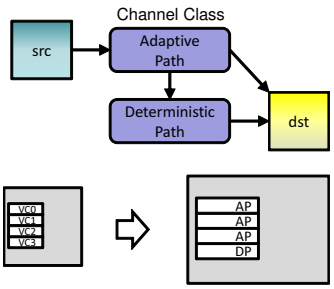

図 5 Deterministic Path を用いた時のパケットの転送イメージ

デッドロックとは，ネットワークに存在する各パケットが，ルーティングにより決定されたパスを通るためにリソースの解放を待つ時に，循環依存を形成し，永遠にパケットの転送がおこなわれない状態を指す．デッドロックフリーなルーティングを実現するにあたり，仮想チャネルを用いてデッドロック回避を提供できることが知られている．その中で，仮想チャネルをクラス分けして，デッドロック回避を提供する手法がある．図 1 では，デッドロック回避手法を (a) から (b) への矢印として，デッドロック回避によってデッドロックフリーとなったルーティングを (b) として示している．

我々はその手法に対して，さらに仮想チャネルを効率的に活用し，性能を上げることが可能な拡張的な手法を提案する．我々の提案手法は，図 1 では，(b) から (c) への矢印として表現している．本稿では，サイクルアキュレイトなネットワークシミュレータを用い，代表的な通信パターンにおけるレイテンシ，スループットの評価から提案手法の有効性を示す．

本稿の構成を以下に示す．2 章では，従来手法である仮想チャネルをクラス分けしてデッドロック回避を提供する手法を簡単な例を挙げて説明する．そして，3 章で提案手法を述べ，4 章で評価をおこない，5 章で考察をおこない，最後に 6 章でまとめる．

2. クラス分けされた仮想チャネルによるデッドロック回避

システムのルーティングによってはデッドロックが起きてしまうものがある．そういったルーティングに対して，仮想チャネルを利用してデッドロックを回避する手法が提案された[1]．さらに，仮想チャネルに対してラベル付けやクラス分けをおこない，各ラベルやクラスに応じてパケットのチャネルアロケーションやルーティングなどの扱いを変えることで，デッドロックを回避できる．この手法を利用，またはこの手法に帰着できるものを利用した研究は数多く発表されている．

本稿では，パケットの進行方向が西から北へと変化した時の 90 度ターンを，W-N と表記する．同様に E-N，W-S，E-S，N-W，N-E，S-W，S-E を定義する．N，S，W，E はそれぞれ北，南，西，東を表すとする．

Minimal fully adaptive routing とは，メッシュ型のトポロジにおいて，送信ノードから受信ノードまでの最短距離となる全てのルートがルーティングによって選択されうる適応型ルーティングである．Minimal fully adaptive routing でのパケットの転送例を **図 2** に示す．

図 2 において，送信ノードから受信ノードまでの最短距離となるルートは全部で 70 ある．Minimal fully adaptive routing によって，パケットはこの 70 のルートの内，どれか 1 つのルートで転送される．また，図 2 の黒の太い矢印で示すように南東方向にパケットを送信する場合，このルートは S-E と E-S の 2 種類のターンをとりうる．

同様に，北西方向にパケットを送信するためには N-W と W-N のターンが，北東方向にパケットを送信するためには N-E と E-N のターンが，南西方向にパケットを送信するためには S-W と W-S のターンが必要になる．これらのターンをまとめたものが **図 3** である． 図 3 に示す通り，パケットが全ての 90 度ターンを取りうるので，それぞれのターンによって，ループが描け，循環依存が発生してしまう．したがって，Minimal fully adaptive routing はデッドロックが起こりうるルーティングである．

この章では，クラス分けされた仮想チャネルを利用したデッドロック回避手法が，実際にどのように利用できるのか，または利用されているのかを Minimal fully adaptive routing を用いて紹介する．

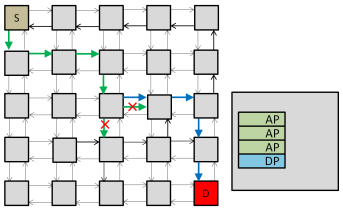

図 6 Deterministic Path を用いた Minimal fully adaptive routing

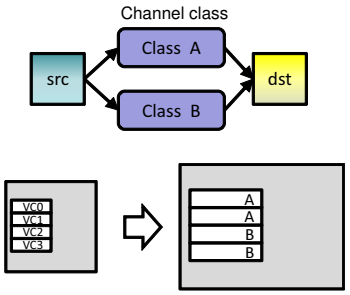

図 7 仮想チャネルの完全分割を用いた時のパケットの転送イメージ

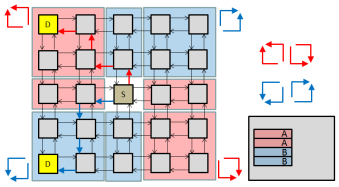

図 8 仮想チャネルの完全分割を用いた時のパケットのルーティングイメージ

2.1 Case A:複数のクラス分けを用いたデッドロック回避

まず始めに，複数のクラス分けを用いたデッドロック回避について紹介する．このデッドロック回避方法は，Escape Path[2),3)] や，dimension reversal routing[4)]，Duan らの研究[5)6)]，mad-y[7)] そして LEAR[8)] などで利用されている．複数のクラス分けを用いた時のパケットの転送イメージは **図 4** である． 図 4 では 4 本の仮想チャネルは Class A，Class B，Class C，Class D の 4 つのクラスに 1 本ずつ振り分けられている．

今回は，dimension reversal routing のように Deterministic Path を用いて，先ほど紹介した Minimal fully adaptive routing をデッドロックフリーなルーティングにする．Deterministic Path を用いた時のパケットの転送イメージは **図 5** である．

Deterministic Path を用いる場合は **図 6** に示すように，各物理チャネルの仮想チャネルは Adaptive Path に 3 本，Deterministic Path に 1 本振り分ける．パケットは最初， 図 6 にあるように，Adaptive Path の仮想チャネルを使って Minimal fully adaptive routing に沿ったルーティングがおこなわれる．Adaptive Path を利用している間のルーティングはデッドロックフリーである必要はない．そして，目的の方向の Adaptive Path の仮想チャネルが他のパケットにより全て占有されて，ブロックされた場合，今度は Deterministic Path を用いて受信ノードへ転送される．しかし，Deterministic Path を用いる際には，図のように XY 次元順ルーティングのようなデッドロックフリーなルーティングを用いて受信ノードへ転送される．また，このルーティングでは一度 Deterministic Path を利用したパケットは，二度と Adaptive Path を利用することはできない．以上のように，Deterministic Path を用いて Minimal fully adaptive routing は実現される．

2.2 Case B:仮想チャネルの完全分割によるデッドロック回避

次に，仮想チャネルの完全分割を用いてデッドロックフリーな Minimal fully adaptive routing を実現する．仮想チャネルの完全分割を用いた時のパケットの転送イメージは **図 7** のようになる．また，ルーティングのイメージを **図 8** に示す．

まず，各物理チャネルの仮想チャネルを Class A と Class B とに分ける．それぞれのクラスには 図 8 のように 2 本ずつ仮想チャネルを割り当てる．パケットの受信ノードの位置が送信ノードから見て，北西，西，東，南東の方角の場合，つまり 図 8 において赤で囲まれたノードにパケットを送信する場合，そのパケットは Class A の仮想チャネルのみを用いて転送される．そして，パケットの受信ノードの位置が送信ノードから見て，北東，北，南，南西の方角の場合，つまり 図 8 において青で囲まれたノードにパケットを送信する場合，そのパケットは Class B の仮想チャネルのみを用いて転送される．

Class A での仮想チャネル内でおこなわれるターンは 図 8 で示されているように，N-W，W-N，S-E，E-S の 4 つのターンのみであり，デッドロックを引き起こさない．また，Class B での仮想チャネル内でおこなわれるターンは，N-E，E-N，S-W，W-S の 4 つのターンのみであり，同じくデッドロックを引き起こさない．Class A と Class B とでルーティングを分けているので，仮想チャネルの完全分割を用いて Minimal fully adaptive routing は実現される．

このように仮想チャネルの完全分割を用いた研究には，O1TURN[9)] やフォールトトレラントルーティング[10)] が存在する．

3. 提 案 手 法

前章に 2 つの例を示したが，それぞれ共通しているのが,「あるパケットが，あるクラスの仮想チャネルを利用している間は,（次のクラスの仮想チャネルに移動する場合を除き）そのクラスに属する仮想チャネルしか利用しない」という制約である．

これは，仮想チャネルをクラス分けすることで，ネットワーク全体をサブネットワークに分け，全体としてデッドロックフリーなネットワークを提供する従来手法の本質である．しかし，我々はこのデッドロック回避手法に対し,「あるパケットが，あるクラスの仮想チャネルを利用している間は，そのクラスに属する仮想チャネルしか利用しない」という制約を変えることで，仮想チャネルをより効率的に利用でき，性能向上が可能な拡張的な手法を 2 つ提案する．

1 つは共有クラスを用いる方法．もう 1 つはクラス間の階層的な関係を用いる方法である．

それぞれの方法を前章で示された 2 つの例に用いて説明する．

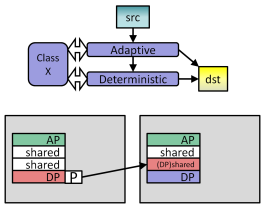

図 9 共有クラスを用いた Deterministic Path

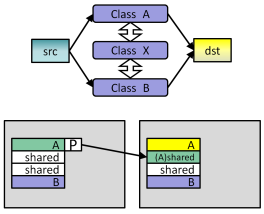

図 10 共有クラスを用いた仮想チャネルの完全分割

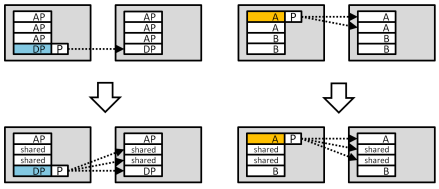

図 11 共有クラスを用いた時の仮想チャネルアロケーションリクエスト

3.1 拡張的な手法の提案

共有クラスを用いる手法

まず初めに提案手法の 1 つとして，各クラスに加え，共有クラス Class X を用意する手法を説明する．この手法を適用した例を **図 9** と **図 10** に示す．

この共有クラスは，仮想的にあらゆるクラスの振る舞いができ，どのクラスのパケットでも受け入れることができるクラスである．つまりパケットからすれば，どのクラスのパケットでも，共有クラスの空いている仮想チャネルを，一時的に現在利用している同じクラスの仮想チャネルとして利用できる．

共有クラスは Adaptive Path としても Deterministic Path としても振る舞えるので， 図 9 のように，Deterministic Path を利用していたパケットが共有クラスの仮想チャネルを Deterministic Path の仮想チャネルとして利用することができるようになる．また，図 10 では，共有クラスは Class A としても Class B としても振る舞えるので，Class A を利用していたパケットが，共有クラスの仮想チャネルを Class A の仮想チャネルとして利用できるようになっている．

以上の手法を用いれば，従来では全仮想チャネルをいずれかのクラスに分けていたのが，各クラスに振り分ける仮想チャネルを最小限に抑え，残った仮想チャネルを共有クラスに振り分けるというクラスの分け方ができる．

この手法による，チャネルアロケーションリクエストの変化を **図 11** に示す．図 11 に示したとおり，Deterministic Path を利用していたパケットが，以前より 2 本多く仮想チャネルアロケーションリクエストが出せるようになり，Class A を利用していたパケットが，以前より 1 本多く仮想チャネルアロケーションリクエストが出せるようになっている．以上のように，全ての仮想チャネルが 1 つのクラスの振る舞いしかできなかった従来手法と比べ，複数のクラスの振る舞いができる共有クラスがある分，パケットの仮想チャネルに対する選択の自由度が上がる．

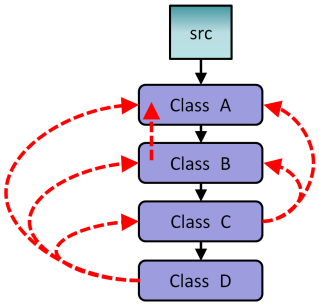

図 12 クラスが 4 つある時の階層的な関係

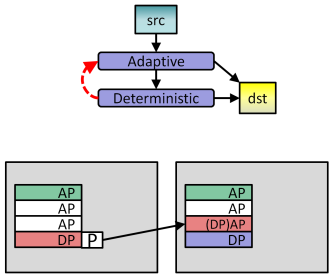

図 13 階層的な関係を用いた Deterministic Path

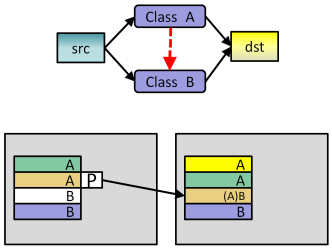

図 14 階層的な関係を用いた仮想チャネルの完全分割

階層的な関係を用いる手法

2 つ目の提案手法は，階層的な関係を用いる手法であり，新しくクラスを追加しなくてもよい手法である．

図 4 では，Class A が Class B，Class C，Class D よりも先に利用される立場にあり，Class B が Class C，Class D よりも先に利用される立場にあり，Class C が Class D よりも先に利用される立場にある．我々はこういった階層的な関係に注目した．

本稿では，クラスに対してランクを定義する．あるクラス M が他のあるクラス N より先に利用される立場のクラスであった場合，そのクラス M はクラス N よりランクの高いクラスと表現する．逆に，あるクラス M が他のあるクラス N より後に利用される立場のクラスであった場合，そのクラス M はクラス N よりランクの低いクラスと表現する．

したがって， 図 4 では，Class A が最もランクの高いクラスであり，Class D が最もランクが低いクラスである．

そして，我々はランクの高いクラスの仮想チャネル

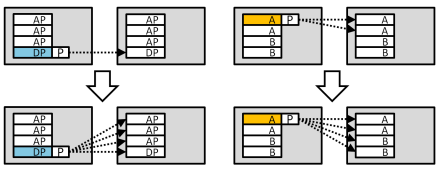

図 15 階層的な関係を用いた時の仮想チャネルアロケーションリクエスト

が，ランクの低いクラスの仮想チャネルとして振る舞うことができるという手法を提案する．つまり，あるクラスを利用しているパケットは，そのクラスよりランクの高いクラスの仮想チャネルを，そのクラスの仮想チャネルとして利用できる手法である．

図 12 を用いて説明する． 図 12 において，破線の矢印の根のクラスのパケットは，破線の矢印の先のクラスの仮想チャネルを，現在利用しているクラスの仮想チャネルとして利用できることを示している．したがって， 図 12 にあるように，Class D を利用しているパケットは，Class A，Class B，Class C の仮想チャネルを Class D の仮想チャネルとして利用でき，Class C を利用しているパケットは，Class A，Class B の仮想チャネルを Class C の仮想チャネルとして利用でき，Class B を利用しているパケットは，Class A の仮想チャネルを Class B の仮想チャネルとして利用できる．

この手法を適用した例を **図 13** と **図 14** に示す．図 13 のように，Deterministic Path を利用していたパケットが Adaptive Path の仮想チャネルを Deterministic Path の仮想チャネルとして利用ができるようになる．また， 図 7 のように，階層的な関係が存在しない場合でも，各クラスのランクをこちらで定義すれば，この手法を適用することができる．たとえば，Class B が Class A よりもランクが高いクラスであると定義すれば， 図 14 のように，Class A を利用していたパケットが，Class B の仮想チャネルを Class A の仮想チャネルとして利用することができる．

この手法による，チャネルアロケーションリクエストの変化を **図 15** に示す． 図 15 に示されているように，この階層的な関係を用いる提案手法によって，Deterministic Path を利用していたパケットが，以前より 3 本多く仮想チャネルアロケーションリクエストが出せるようになり，Class A を利用していたパケットが，以前より 2 本多く仮想チャネルアロケーションリクエストが出せるようになる．以上のように，全ての仮想チャネルが 1 つのクラスの振る舞いしかできなかった従来と比べ，パケットの仮想チャネルに対する選択の自由度が上がる．

3.2 デッドロック回避の問題

我々が提案する手法は，デッドロック回避が必要なルーティングに対し，従来の手法でデッドロックフリーなルーティングを提供し，さらに仮想チャネルを効率的に利用できるルーティングを提供するものである．この節では，提案手法を用いた後もデッドロックフリーが保たれているのかを議論したい．

共有クラスを用いる手法

まず，共有クラスを用いる場合を考える．共有クラスの仮想チャネルが一時的に別のクラスとして振る舞うことがない限り，つまり，あるパケットが共有クラスの仮想チャネルを利用しない限りは，元々のデッドロックフリーなルーティングと相違ない．よって，共有クラスの仮想チャネルが一時的に別のクラスとして振る舞い，あるパケットが共有クラスの仮想チャネルを一時的に利用している場合を考える．

この場合，あるパケットが共有クラスの仮想チャネルを利用しても，それはパケットが利用しているクラスの仮想チャネルが一時的にただ増えたことと同じことである．つまり，共有クラスの仮想チャネルを利用してもそのパケットは転送がおこなわれる．したがって，そのパケットが利用した共有クラスの仮想チャネルの資源は有限時間内に解放されるので，元のデッドロックフリーなルーティングと同じ状況に戻る．

よって，共有クラスを用いる手法を利用しても，デッドロックフリーであることが保たれる．

階層的な関係を用いる手法

次に，階層的な関係を用いる場合を考える．あるクラスの仮想チャネルが一時的に別のクラスとして振る舞うことがない，つまり，あるパケットが今いるクラスとは別のクラスの仮想チャネルを利用しない限りは，元々デッドロックフリーなルーティングと相違ない．よって，あるクラスの仮想チャネルが一時的に別のクラスとして振る舞い，あるパケットが今いるクラスとは別のそのクラスを一時的に利用している場合を考える．

具体的に 図 4 の環境において，ある物理チャネルの Class B に振り分けられたある仮想チャネル c_b が，Class C の仮想チャネルを利用していたパケット p_0 を受け入れるために，一時的に Class C として振る舞い，そのパケット p_0 を受け入れた時を考える．この場合，一時的に Class C の仮想チャネルが増え，Class B の仮想チャネルが減った状態になるとみなせる．仮想チャネル c_b が Class C として振る舞ったことで，ネットワークによっては，その物理チャネル内に Class B として振る舞う仮想チャネルが無くなった可能性がでてくる．したがって， Class B の仮想チャネルを利用していた他のパケットが一時的に詰まってしまう可能性がでてくる．当然 Class A の仮想チャネルを利用しているパケットも，後に Class B の仮想チャネルを利用しうるので，この詰まりの影響を受ける．しかし，逆に Class C は仮想チャネルが増えた状況となっているので，パケット p_0 を含めた Class C の仮想チャネルを利用していた他のパケットはスムーズに転送がおこなわれる．Class D についても同様である．つまり，パケット p_0 自体は転送がおこなわれるので，有限時間内で仮想チャネル c_b はパケット p_0 の占有からは解放される．そして，仮想チャネル c_b はパケット p_0 から解放されることで，元の Class B の仮想チャネルとして振る舞うようになる．したがって，ネットワークは元の状態に戻り，元々デッドロックフリーなルーティングと相違ないものとなる．

よって，階層的な関係を用いる手法を利用しても，デッドロックフリーであることが保たれる．

以上により，提案手法を用いてもデッドロックフリーであることが保たれる．

4. 評　　価

4.1 提案手法の適用

2.1 の Case A で実現させた Minimal fully adaptive routing に対して，共有クラスを用いる方法と，階層的な関係を用いる方法の両方を適用し，さらに 2.2 の Case B で実現させた Minimal fully adaptive routing に対して，共有クラスを用いる方法と，階層的な関係を用いる方法の両方を適用し，評価をおこなう．

Case A で実現させた Minimal fully adaptive routing （A とする）と，共有クラスを用いる手法を適用したもの（A_shared とする）と，階層的な関係を用いる手法を適用したもの（A_hierar とする）において，各クラスに 4 本の仮想チャネルを何本振り分けたのかを **表 1** に示す．また Case A において，Adaptive Path のクラスは高位，Deterministic Path のクラスは低位のクラスであるため，A_hierar においては，Deterministic Path を利用しているパケットは，Adaptive Path の仮想チャネルが Deterministic Path として振る舞えるので，Adaptive Path の仮想チャネルが利用できるものとする．したがって，Deterministic Path を利用しているパケットは合計 4 本の仮想チャネルが利用できる．

Case B で実現させた Minimal fully adaptive routing （B とする）と，共有クラスを用いる手法を適用したもの（B_shared とする）と，階層的な関係を用いる手法を適用したもの（B_hierar とする）において，各クラスに 4 本の仮想チャネルを何本振り分けたのかを **表 2** に示す．また B_hierar において，Class A および Class B には元々ランクは存在しないが，Class B をランクの高い，Class A をランクの低いクラスとした．Class A を利用しているパケットは，Class B の仮想チャネルが Class A として振る舞えるので，Class B の仮想チャネルが利用できるものとする．したがって，Class A を利用しているパケットは合計 4 本の仮想チャネルが利用できる．

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イトなソフトウェアシミュレータを用いる．評価環境を **表 3** に示す．8×8 ノードの 2 次元メッシュ型のトポロジで，ワームホール方式のフロー制御をおこなう 5 段パイプラインの Input-buffered ルータを用いたネットワークとする．また，パケット長は 8 フリットとし， 表 1 および 表 2 にあるように 1 本の物理チャネルにつき 4 本の仮想チャネルを用意，また 1 本の仮想チャネルにつき 4 つのフリットバッファを持つとする．

通信パターンは，今回は Uniform， Tornado, Hotspot, Transposed の 4 パターンを用いる．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．

4.2 スループットとレイテンシの評価

図 16 から **図 23** に，パケットの注入レートを変化させた時の，A，A_shared，A_hierar，B，B_shared，B_hierar のレイテンシを示す．

Uniform，Tornado，Hotspot のどの通信パターンにおいても，全ての場合で元の構成よりも提案手法の方が，通信が飽和した時の注入レートの値が大きくなっておりスループットが高くなっている．

しかし，Transposed の通信パターンにおいては，**図 22** で示される A_shared，A_hierar は，数値上では確かに良い方に転じているが，元となる A からはほとんど変わっていないと言える． **図 24** は，外周上のノードを除いた時の各ノードでの，各通信パターンにおける，元の A での Deterministic Path の利用率を示したものである．外周上のノードを除いたので，北から，南から，西から，東からの全ての物理チャネルがそれぞれ他のノードと繋がっている．さらに各物理チャネルはそれぞれ 1 本の Deterministic Path を持っているので，1 ノード当たり 4 本の仮想チャネルを持っている．ただし， 表 3 で示した Local 方向へのチャネルは無視する．したがって， 図 24 で示される仮想チャネルの利用率の最大値は 4 となる．

図 24 と各通信でのレイテンシとを見比べると，元となる A の時点で Deterministic Path をより多く利用する通信において，提案手法がより性能向上を果たしていることが分かる．これは，提案手法によって Deterministic Path を利用しているパケットが Adaptive Path の仮想チャネルを利用できるようになるので，元となる A の時点で Deterministic Path をより多く利用する通信の方が，提案手法の恩恵を受けやすいためと考えられる．

しかし，逆に 図 23 で示される B_shared，B_hierar は大きな性能向上を果たしている． **図 25** は， 図 24 のように，外周上のノードを除いた時の各ノードの，Transposed における仮想チャネルの利用率を示したものである．各物理チャネルはそれぞれ 4 本の仮想チャネルを持っているので，1 ノード当たり 16 本の仮想チャネルを持っている．ただし， 表 3 で示した Local 方向へのチャネルは無視する．したがって， 図 25 で示される仮想チャネルの利用率の最大値は 16 となる．

元々低かった仮想チャネルの利用率が，提案手法によって高くなったことが， 図 25 から分かる．特に，B_hierar に至っては，元の利用率から倍の利用率を達成している．このように，仮想チャネルの利用率を改善したことで， 図 23 のように性能向上が達成できることが示された．

5. 考　　察

今まではデッドロック回避手法の拡張方法として，提案手法を扱ってきたが，ここでは新しいルーティング生成手法として考察する．

例えば，XY ルーティングと YX ルーティングを同時に扱えるルーティングを考える．仮想チャネルが 4 本あった時，最も簡単なやり方が，2 本を XY 用，もう 2 本を YX 用とする方法である．しかし，今回の提案手法を用いれば，より高性能のルーティングを作り出すことができる．

図 26 と **図 27** を用いて説明する．まず， 図 26 のように，XY なら X 軸方向に，YX なら Y 軸方向

表 1 Case A で各クラスに振り分けた仮想チャネルの本数

	Adaptive	Deterministic	Shared Class
A	3	1	—
A_shared	1	1	2
A_hierar	3	1	—

表 2 Case B で各クラスに振り分けた仮想チャネルの本数

	Class A	Class B	Shared Class
B	2	2	—
B_shared	1	1	2
B_hierar	1	3	—

表 3 評価環境

Network Size	8 × 8, 64 nodes
Traffic Pattern	Uniform, Tornado, Hotspot, Tranposed
# Directions	5-input/output (North, East, West, South, Local)
# Virtual Channels	4
Pipeline	5-Stage Pipeline RC: Routing Computation, VA: Virtual Channel Allocation, SA: Switch Allocation, ST: Switch Traversal, LT: Link Traversal
Packet Length	8
Input Buffer Size	4
Flow Control, Transfer	Credit-Base Flow Control, Warmhole
# Simulation Cycles	Warm-up: 10,000 cycles, Simulation: 100,000 cycles

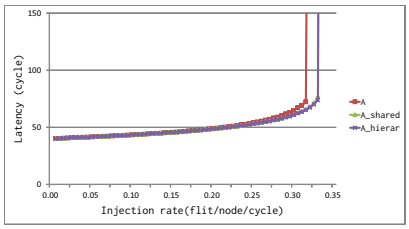

図 16 uniform でのレイテンシ (A, A_shared, and A_hierar)

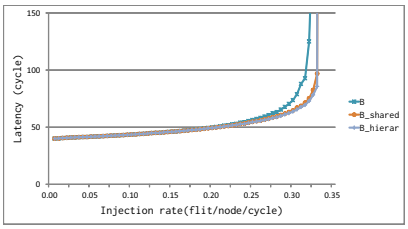

図 17 uniform でのレイテンシ (B, B_shared, and B_hierar)

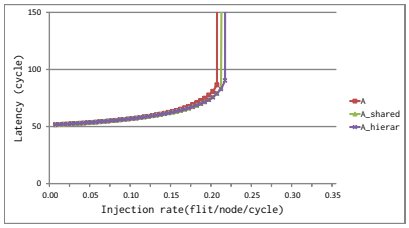

図 18 tornado でのレイテンシ (A, A_shared, and A_hierar)

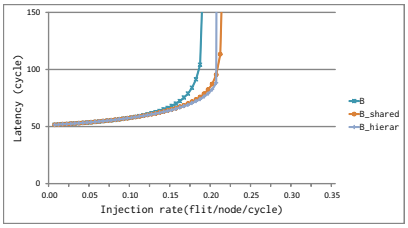

図 19 tornado でのレイテンシ (B, B_shared, and B_hierar)

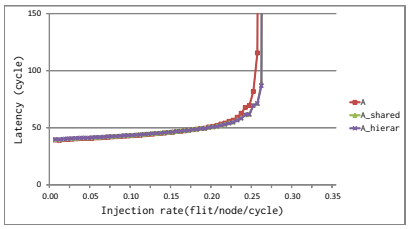

図 20 hotspot でのレイテンシ (A, A_shared, and A_hierar)

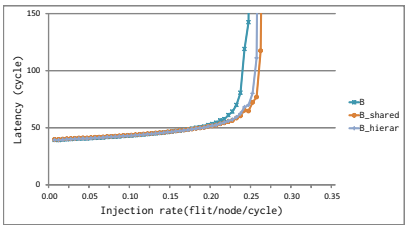

図 21 hotspot でのレイテンシ (B, B_shared, and B_hierar)

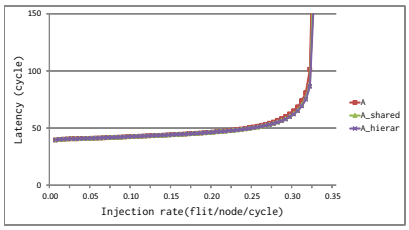

図 22 tranposed でのレイテンシ (A, A_shared, and A_hierary)

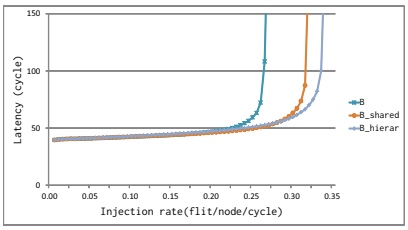

図 23 tranposed でのレイテンシ (B, B_shared, and B_hierar)

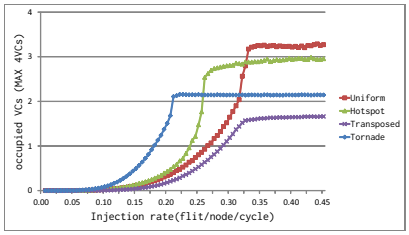

図 24 仮想チャネルの利用率 (Deterministic Path)

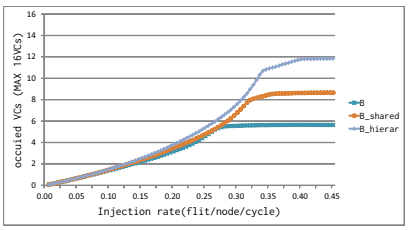

図 25 仮想チャネルの利用率 (transposed)

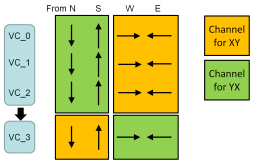

図 26 XY と YX とを同時に扱えるルーティング

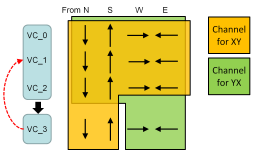

図 27 提案手法を利用したルーティング

に移動するときは，4 本ある仮想チャネルの VC_0 から VC_2 までの 3 本を利用できるとする．次に，XY なら Y 軸方向に，YX なら X 軸方向に移動するときは，VC_3 だけ利用できるとする．すると，VC_0 から VC_2 までのクラスと，VC_3 のクラスに分かれる．ターンする時にクラスを移動するので，各クラス内ではターンは存在せず，デッドロックは発生しない．ここで，提案手法の階層的な関係を用いる方法を利用すれば，XY と YX は共に，ターンした後も VC_0 から VC_2 までの 3 本を利用できるようになる．したがって，図 27 のように，XY は X 軸方向に移動するときは VC_0 から VC_2 までの 3 本を利用でき，Y 軸方向に移動するときは全ての仮想チャネルが利用でき，YX は Y 軸方向に移動するときは VC_0 から VC_2 までの 3 本を利用でき，X 軸方向に移動するときは全ての仮想チャネルが利用できるルーティングが完成する．

このルーティングは従来のデッドロック回避方法だけでは生成することのできないルーティングである．このように，提案手法をルーティング生成法として利用することで，今までの方法では導かれることのなかった新しいルーティングを生成することが可能である．

6. 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 における高性能へのアプローチとしてルーティングがある．ルーティングにおいてデッドロックフリーであることは重要であり，以前より仮想チャネルを用いてデッドロック回避が提供できることが知られている．その中で，仮想チャネルをクラス分けして，デッドロック回避を提供する手法がある．

我々はその手法に対して，さらに仮想チャネルを効率的に活用し，性能を上げることが可能な拡張的な手法を提案した．

ソフトウェアシミュレータを用いた評価により，例として示したデッドロックフリーな Minimal fully adaptive routing に対し提案手法を適用した時，スループットが向上することを確認した．

謝辞 本研究の一部は，科学技術振興機構・戦略的創造研究推進事業 (CREST)「ディペンダブルネットワークオンチッププラットフォームの構築」の支援による．

参 考 文 献

1) Dally, W. J. et al.: Deadlock-Free Message Routing in Multiprocessor Interconnection Networks, *IEEE Trans. Comput.* (1987).
2) Duato, J.: A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks, *IEEE Trans. Parallel Distrib. Syst.* (1993).
3) Duato, J.: A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks, *IEEE Trans. Parallel Distrib. Syst.* (1995).
4) Dally, W. J. et al.: Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels, *IEEE Trans. Parallel Distrib. Syst.* (1993).
5) Duan, X. et al.: Fault-Tolerant Routing Schemes for Wormhole Mesh, *Parallel and Distributed Processing with Applications, 2009 IEEE International Symposium on* (2009).
6) Duan, X. et al.: A Multiphase Routing Scheme in Irregular Mesh-Based NoCs, *Proceedings of the 2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming*, PAAP '11 (2011).
7) Glass, C. J. et al.: Maximally Fully Adaptive Routing in 2D Meshes, *In International Conference on Parallel Processing, volume I* (1992).
8) Ebrahimi, M. et al.: LEAR – A Low-Weight and Highly Adaptive Routing Method for Distributing Congestions in On-chip Networks, *Parallel, Distributed and Network-Based Processing (PDP), 2012 20th Euromicro International Conference on* (2012).
9) Seo, D. et al.: Near-Optimal Worst-Case Throughput Routing for Two-Dimensional

Mesh Networks, *Proceedings of the 32nd annual international symposium on Computer Architecture*, ISCA '05 (2005).
10) Borhani, A. et al.: A new deterministic fault tolerant wormhole routing strategy for k-ary 2-cubes, *Computational Intelligence and Computing Research (ICCIC), 2010 IEEE International Conference on* (2010).