

# 省電力 NoC 向けの動的リンク分割を用いた ハイブリッドスイッチング手法

和 遠<sup>1,a)</sup> 近藤 正章<sup>1</sup>

**概要:** 近年の NoC では Virtual Channel 方式によるフロー制御が一般的であるが、ルータ上のバッファによる消費電力増大が問題となる。本稿では、アプリケーションのトラフィックに合わせて、NoC ルータ、およびルータ間のリンクを仮想的に分割し、それぞれを Circuit Switching (CS) と Virtual Channel (VC) 方式によるフロー制御に分けて利用するハイブリッドスイッチング手法を提案する。本手法では、動的にプロファイリングを行いつつ、頻繁に使用される経路をある期間固定的に CS のデータパスを構築し転送する。CS フロー制御のチャンネルでは、VC 方式で必要となるバッファアクセスの必要がなく、ルータ上の経路制御も単純になることから、転送レイテンシの削減も期待され、高性能と省電力化を達成することが可能となる。シミュレーションによる評価の結果、VC 方式のみに比べ、提案手法では転送するフリットあたりの消費エネルギーを平均で 64%程度、レイテンシを平均で 32%程度削減できることがわかった。

## 1. はじめに

プロセッサチップのコア数は増加の一途を辿っており、コア間を接続するネットワークオンチップ (Network-on-chip: NoC) の重要性は高まっている。近年の NoC では、コア間のリンクバンド幅を有効活用し高いスループットを達成するため、フロー制御手法として Virtual Channel (VC) 方式が用いられることが一般的である。一方で、VC 方式では各オンチップルータにおいてフリットをバッファリングする必要があることから、バッファアクセスのダイナミック電力、およびバッファ自体のスタティック電力増加が問題となる。さらに、ルータのパイプライン構成にも依存するが、バッファアクセスの時間は無視できず、ホップあたりのレイテンシ増加も問題となり得る。これら問題へ対処のため、従来から様々な研究が行われている [2], [5], [9], [17], [18], [19], [21]。

これらの従来研究とは別に、バッファの利用、すなわち VC 方式の利用を自体を再考する研究もある。一般的に、NoC はバンド幅の要求が非常に高い場合でも動作するように設計されているが、アプリケーションによって NoC への要求は大きく異なり [10]、そこまで高いバンド幅は必要がないことが多い [8], [9], [22]。従って、VC 方式は多くの場合有効利用されていない。そこで、VC と Circuit-Switching (CS) フロー制御方式を組み合わせるハイブリッド

スイッチング (HS) 手法が提案されている [3], [12], [24]。

CS 方式では、あらかじめパケットの送信元から宛先までの経路をルータ上に設定することで通信を行い、バッファリングの必要がないために VC に比べてレイテンシが短く、また省電力となる。しかし、経路設定に要する遅延が大きく、設定中は経路上のルータのポート、およびリンクが予約され、それらを利用した他の通信が行えないことからスループットが制限されてしまう。一方、VC 方式ではリンクやルータのポートを複数のパケットで共有できるため、高いスループットが達成可能である。

CS と VS 手法の両者の利点を活用する目的で、本稿では NoC をアプリケーションの特徴に応じて複数の論理チャンネルに適応的に分割し、そのうちの 1 つを VC に、残りを複数の CS チャンネルとして利用する手法を提案する。分割においては、ルータ間、およびルータ内の物理的なリンクのビット幅を複数に分割し各フロー制御に割り当てる。例えばリンクのビット幅が 128bit でありそれを 2 分割する場合は、64bit を VC フロー制御に、残りの 64bit を CS フロー制御に用いてアプリケーションを実行する。

CS フロー制御の利点を十分に生かすためには、パケット転送リクエストの度に CS パスを構築するのではなく、CS 方式に利用可能リンク上なるべく多くの CS パスを構築し、そのパスを何度も再利用して転送することが重要である。そこで、提案手法では、トラフィックの規則性に着目し、動的にプロファイリングを行いつつ頻繁に使用される経路のいくつかを CS パスとして割り当てる。我々の予備評価によると、多くのアプリケーションにおいて、半分以上の転送が送信元・送信先の全組み合わせのうちの

<sup>1</sup> 東京大学 大学院情報理工学系研究科  
Graduate School of Information Science and Technology,  
The University of Tokyo

<sup>a)</sup> he@hal.ipc.i.u-tokyo.ac.jp

5%のみのパスで発生することがわかっている。このように、NoC上のトラフィックには規則性があるため、頻出パスを抽出し、それらをCS方式で転送することで、多くのトラフィックがCSフロー制御の利点を楽しみ得ると思われる。一方で、CSチャネルを利用できないトラフィックについては、VCフロー制御方式により転送することで、その都度CSパスを設定することなく、全体のデータ転送を正しく行うことが可能である。

本提案手法には、以下の利点がある。1)CSチャネルを利用するトラフィックに関して、バッファアクセスが必要ないためダイナミック電力を削減でき、またCSパスに設定した経路上のバッファをパワーゲーティングすることでスタティック電力も削減可能である。2)CSチャネル上の転送では、ホップあたりのレイテンシが短くなり、性能向上が期待できる。3)本提案手法は近年のNoCにおけるルータの設計に対して、わずかな拡張で実装可能である。本稿では、提案手法のアーキテクチャ、およびCSパスの設定方法について述べた後、いくつかの並列ベンチマークプログラムを用いたネットワークのレイテンシの削減効果と消費エネルギー削減効果の評価結果を示す。

## 2. NoCのフロー制御

NoCのフロー制御手法は、もともとバッファを用いないものから、バッファを利用したデザインへと進化してきた。最も一般的なバッファを用いないフロー制御手法はCircuit-Switching (CS) [4]である。CSは実装が容易であり消費電力も小さいが、各転送のための経路設定に要する時間的なコストが大きく、チャネル利用効率の面で効率が悪い。バッファを導入することで、チャネル割り当てを待つ間、バッファにパケットをストアできることから、予め隣り合うルータ間のチャネルの結合を設定する必要がなく、チャネルバンド幅をより効率的に利用することができる。

バッファを用いるフロー制御手法としては、Store-and-Forward, Cut-Through, Wormhole, および Virtual Channel (VC) などがある。近年のNoCではVCフロー制御方式がもっとも一般的になっている。VC方式では、ポート競合などでブロックされたパケットをいったんバッファに保存することで、他のパケットが当該パケットを追い越して転送することができ、各チャネル効果的に使用可能である。1つの物理チャネルに対し、より多くの仮想チャネルを割り当てることで、より多くのパケットのブロックに対応することができるが、バッファのコストとハードウェアの複雑度が増大する。

CSからVC方式へとアーキテクチャが進化するにともない、スループットが向上する一方、消費電力が増加してしまう。さらに、バッファへのアクセスや複雑な資源割り当てを行う必要があることから、ホップ毎のレイテンシも増加してしまう。図1はCS方式、CSから経路設定のオーバーヘッドを除いた場合、およびVC方式のそれぞれについて、フリット注入レートを増加させた際のネットワーク

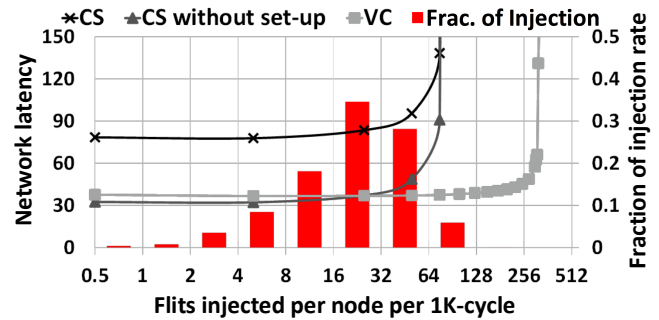


図1 フリット注入レートを増加させた際のレイテンシ変化とアプリケーションの実際の注入レート

レイテンシを示したものである。図に示すように、フリット注入レートを増加させると、CSのバンド幅はVCに比べて早く飽和してしまう。また、経路設定オーバーヘッドのため、VCに対してレイテンシも大きい。一方で、経路設定オーバーヘッドを除いた場合には、ネットワークが飽和していない領域ではVCに比べてレイテンシが小さいこともわかる。

文献[10]によると、NoCの各チャネルの平均負荷とピーク負荷には大きな差があると報告されている。これは、ある時刻に着目すると、一部のチャネルのみが他に比べ負荷が大きい場合があることを意味している。ほとんどの時間は、ネットワークの大部分は負荷が低く、ハードウェアが有するバンド幅が有効利用されていない。図1には、“ocean non-contiguous”ベンチマークを実行した際の実際のフリット注入レートの割合をヒストグラムで示しているが、大部分で注入レートがノードあたり64フリット/1K-cycle以下であることがわかる。この場合はCSフロー制御でも十分なバンド幅を確保できる。しかし、64フリット/1K-cycle以上の場合もわずかながら存在し、これらのトラフィックの場合はCSフロー制御では実行不可能になってしまう。そのため、VC方式の性能が一部では必須である。

## 3. 動的リンク分割および割り当て手法

### 3.1 ハイブリッドスイッチング

一つのNoC内でVCおよびCSフロー制御方式の両者を実現するために、本稿では空間分割多重方式 (Space-Division Multiplexing: SDM) のコンセプトを利用する。ここでは、ルータ間、およびルータ内の物理的なリンクのビット線を論理的に複数のビット線に分割し、それぞれに異なるフロー制御手法を適用する。例えば、リンクのビット幅が128bitであり、それを2分割する場合は、64bitのVCチャネルと64bitのCSチャネルの2つのチャネルを持つNoCとなる(図2参照)。

CSパスは、あらかじめ設定された経路に従ってデータを転送するため、決められた送信元と宛先の場合のみ、そのパスを使用してパケット送信を行うことができる。CSパスは、リンクと経路上のルータのポートが競合しない限り、同時に複数設定することもできるが、送信元と宛先の

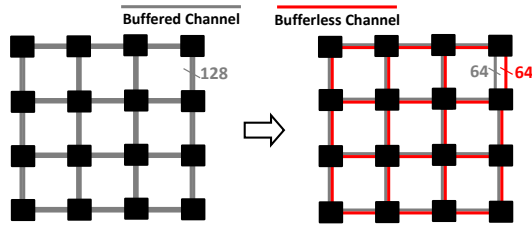


図 2 リンクビット幅の分割の例

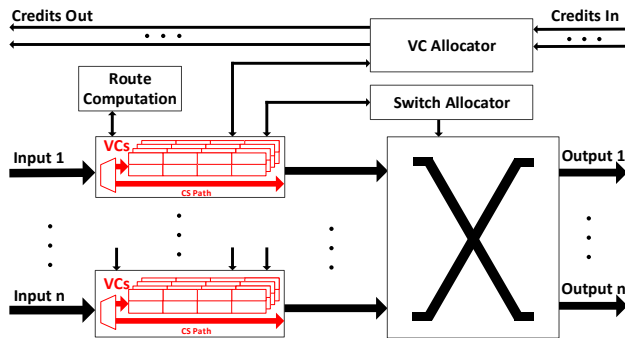


図 3 ハイブリッドスイッチング向けのルータ拡張

組み合わせの全パターンを同時に設定することは不可能である。そこで、CS データパスを使用できないパケットは、VC チャンネルを利用して転送を行う。従来の CS 方式では、CS パスが使用できないパケットに対して新たに経路を設定した後に転送を行う必要があり、そのレイテンシ、さらにその間は他のパケット送受信ができないことが問題となるが、提案手法はレイテンシの影響等を受けずに全てのパケットを転送することができる。

CS データパスを用いたパケットの転送では、バッファリングを行う必要がなく、ルータ内でのパイプライン化も必要ないため、省電力化と当該パケットの転送レイテンシ削減が期待される。そのためには、なるべく多くのフリットを CS チャンネル上で転送することが重要となる。リンクを分割する際に分割数を増やし、1つのチャンネルを VC に、残りのチャンネルを CS に割り当てることで、CS 方式を利用するフリット数の増加が期待できる。一方で、リンクビット幅が小さくなることで、より多くのフリットに分割して転送を行う必要があることから、パケット全体の伝達レイテンシ (最後のフリットが宛先に届くまでの時間) が長くなり、性能に悪影響を及ぼすことも考えられる。物理リンクをどのように分割し、どのように CS データパスを設定するかについては 3.2 節で述べる。

SDM によるハイブリッドスイッチング手法を実現するためのルータの拡張を図 3 に示す。CS として割り当てたチャンネルでは、ルーティングの計算処理やクロスバーの設定を省くことができ、ルータでのパケット処理は 1 サイクルで完了することが可能となる (なお、ルータ間のリンク上の転送にも 1 サイクル必要である)。例えば、CS 上で 3 ホップの転送を行う場合は、3 サイクルがルータで、4 サイクルがリンク上の転送に必要なことから、送信元から送信先までの通信に合計で 7 サイクルかかる。本ルータでは、



図 4 エポックベースの適応的 CS パス割り当て

分割をせずに全ビットを VC フロー制御に用いることも想定し、全ビット幅分のバッファを用意するが、CS フロー制御に用いる場合は当該ビット部分のバッファをパワーゲーティングする。

### 3.2 適応的な CS チャンネルの設定

CS データパスの設定にあたっては、経路設定遅延のオーバーヘッドを最小限に抑えつつ効果的に CS フロー制御を用いて通信を行う必要がある。また、1 節でも述べたように、アプリケーションの実行には特徴の異なるいくつかのフェーズで成り立っていることが多い。そこで、アプリケーションの実行を一定サイクル (エポック) 毎に分割し、最も頻繁に転送に使用すると予想されるいくつかのパスを CS の経路として設定する。同一のエポック内では再度の経路設定は行わない。そのため、エポックの期間がある程度長ければ、オーバーヘッドの影響はほとんど無視できる。

図 4 にエポックベースの CS データパスの経路設定法の概要を示す。エポック毎にそれぞれのコア (送信元) において、どの送信先へどれだけの転送があったかの統計情報を取得する。そして、各エポックの最後で統計情報をもとに頻出の送信元・送信先のペアから CS データパスを割り当てる。なお、この CS データパスの抽出は greedy に行い、最もトラフィック量の多いペアの経路から順に CS パスを割り当てていき、これ以上割り当てられなくなったら終了とする。なお、既に割り当てた CS データパスと新たに割り当て用とするパス間で競合が発生する場合は、当該パスは割り当てない。次に、選択したパスの経路設定を行い、次のエポックでは、当該パスに一致するフリットは設定したチャンネルを用いて CS フロー制御に基づき転送が行われる。一方で、CS チャンネルが設定されていないフリットに関しては、VC チャンネルへフリットを送出し、VC フロー制御に基づいて転送が行われる。

この適応的な経路設定制御を実現するためには、統計情報から最適な CS データパスを抽出し CS データパスを設定するためのコントローラが必要となる。これらはソフトウェアで行うことを想定する。一方で、データパス設定中は CS チャンネルを利用することはできず、(ビット幅が減少した) VC 方式のチャンネルのみで転送を行う必要があることから、性能が低下する。これが本提案手法のオーバーヘッドとなる。

### 3.3 CS チャンネル利用の拡張

例えば、4 メモリチャンネルを持つ 16 コアの CMP では、

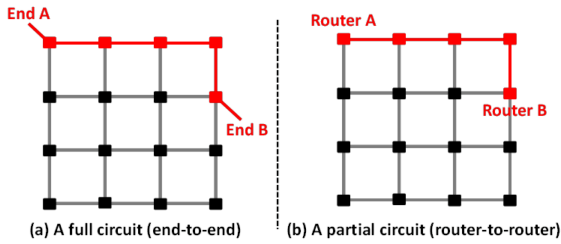


図 5 CS チャンネル利用の拡張方式の例

表 1 評価におけるパラメータ

|                      |                                               |
|----------------------|-----------------------------------------------|
| Number of cores:     | 16                                            |
| Topology:            | 4 × 4 mesh                                    |
| Processor:           | 4 GHz, In-order                               |
| L1 I/D cache:        | 32 KB per Processor,<br>4-way set associative |
| L2 cache:            | 256 KB per Bank,<br>16-way set associative    |
| Cache line:          | 64 Bytes                                      |
| Main memory:         | 4 GB                                          |
| Main memory latency: | 160 cycles                                    |
| Coherence protocol:  | MOESI, Directory                              |
| Link:                | 128-bit, 1 cycle traversal                    |
| Packet:              | 128-bit control, 640-bit data                 |
| Router:              | 1 GHz, 4-cycle virtual channel<br>router      |
| Virtual channel:     | 4 per Virtual network                         |
| Virtual network:     | 3 per Physical link                           |
| Routing algorithm:   | X-Y routing                                   |
| Process technology:  | 32 nm                                         |
| Vdd:                 | 1 V                                           |

コアで 16, 各コアにある L2 キャッシュで 16, メモリコントローラで 4 つの計 36 個のインタフェースを持つことになる。36 個の送信先・送信元の組み合わせで CS チャンネルの設定を考えると、組み合わせの数も大きく、当該ペアの組み合わせのみでしか CS チャンネルを利用できないとすると効率的でない。本稿での提案手法は、なるべく多くのトラフィックを CS チャンネルで転送するのがエネルギー効率、および性能においても有効であるため、CS チャンネル利用の拡張手法を検討する。

拡張手法では、各インターフェース単位の組み合わせではなく、図 5 に示すようにインタフェースが属するルータ単位の組み合わせで CS チャンネルの経路設定を行う。送信元と送信先のタイルがあるインタフェースからルータへ、またその逆の転送の際には調停が必要となることから、インターフェース単位で経路設定をする場合に比べてそれぞれで 2 サイクル余分にレイテンシがかかることになる。しかし、より多くのフリットを CS チャンネルで転送できることから、ペナルティに比べても利点が大きいと考えられる。

#### 4. 評価手法

提案手法の評価のために、本稿では GEMS [16], および Simics [15] をベースの評価環境として用いる。また、

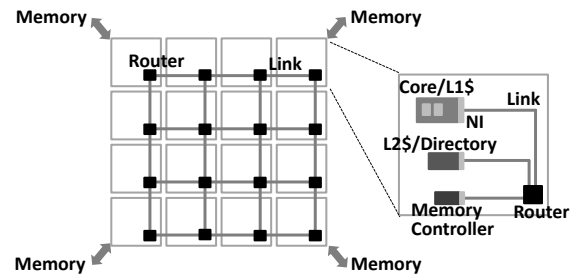


図 6 評価で用いる 16-tile CMP の概要

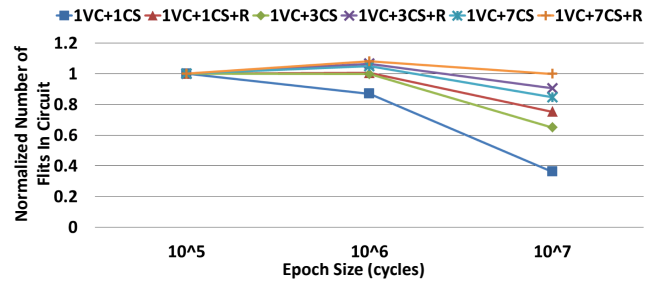


図 7 エポック (cycle) を変えた場合の CS チャンネルで転送したフリット数 (アプリケーション平均)

Garnet [1] によりネットワークモデルを拡張し、また電力評価のために McPAT [14] も用いる。なお、サイクルレベルで提案手法のアーキテクチャが評価できるよう、GEMS および Garnet のコードを変更した。CS フロー制御手法を適用する際のパワーゲーティングに関するパラメータは文献 [11], [17] を参考にした。

本評価では、16-tile をメッシュネットワークで繋いだ CMP を想定し、NoC リンクのビット幅は 128 ビットを仮定する。図 6 に評価で用いる CMP の概要を示す。各タイルはインオーダーのプロセッサコア、L2 キャッシュ・directory を持つ。加えて、4 隅のタイルはメモリコントローラも搭載される。キャッシュミスが発生し、オフチップメモリへアクセスが発生する場合は、アドレスに応じてそれらのメモリコントローラを持つタイルを経由してメモリアクセスが発行される。プロセッサコア、L2 キャッシュ (directory), メモリコントローラは個別に当該ノードのルータへと接続される。プロセッサコア、およびネットワークのパラメータを表 1 に示す。評価に用いるベンチマークは、NPB-3 [20], および SPLASH-2 [23] の中からいくつかのプログラムを利用する。

#### 5. 評価結果と考察

図 7 に、エポックサイクル数を変化させた場合の提案手法における CS チャンネルで転送したフリット数 (評価アプリケーション平均) を示す。なお、図の値は 10<sup>5</sup> サイクルの結果に対する相対値である。また、ビット幅の分割数が 2, 3, 8 個の場合を評価しており、VC 方式のチャンネル数は 1 に固定している。また、“+R” は節で述べた拡張手法を意味する。図より、エポックサイズが 10<sup>6</sup> サイクルの場合、



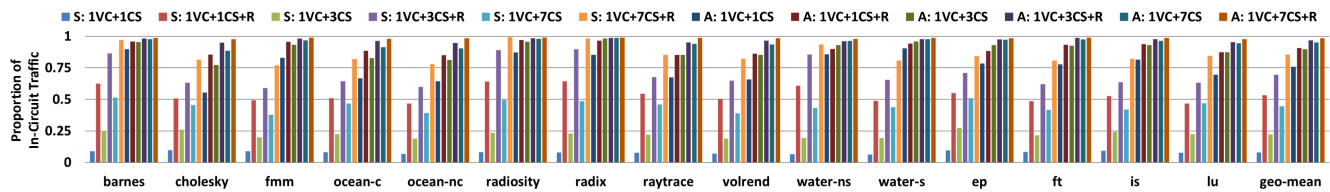


図 8 CS チャンネルで転送したトラフィックの割合

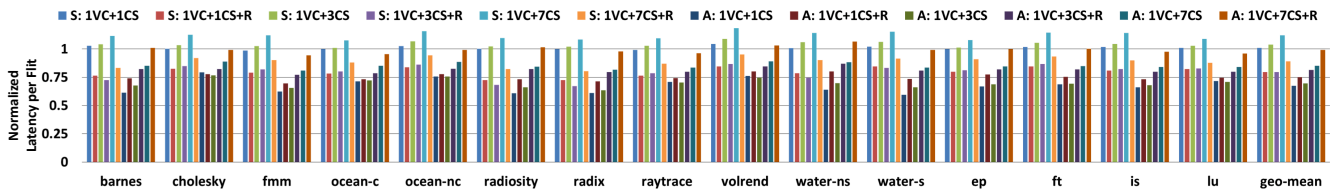


図 9 フリットあたりの相対転送レイテンシ

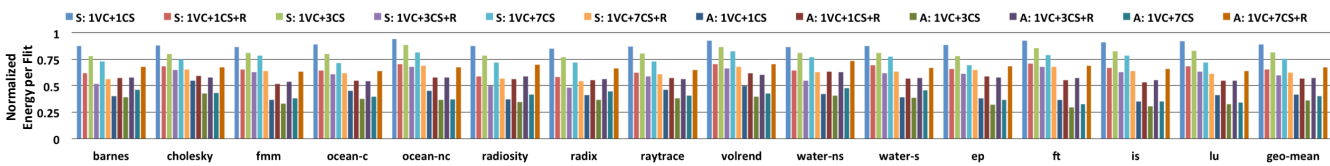


図 10 フリットあたりの相対消費エネルギー

また拡張手法を用いた場合に最も CS チャンネルを用いた転送が行えていることがわかる。そこで、以降の評価では、 $10^6$  サイクルを用いた場合の結果を示す。

図 8 はアプリケーション毎に、全トラフィックに対する CS チャンネルを用いて転送したトラフィックの割合を示したものである。図の“S:”と“A:”は、それぞれアプリケーションを通して静的に CS チャンネルを割り当てた場合とエポックベースで適応的に割り当てた場合を表している。提案手法では、最悪の場合でも 75% ものフリットが CS 方式で転送されている。CS チャンネルが増加するにつれ、また拡張手法を用いることで、より多くのトラフィックを CS チャンネル上で転送することが可能となる。また、多くの場合で、適応的に CS チャンネルを割り当てることで CS チャンネル上での転送量が増えることもわかる。一方で、barnes, radiosity, radix などのアプリケーションでは静的に割り当てた場合でも、7CS チャンネルおよび拡張手法を適用することで、ほとんどのフリットを CS チャンネル上で転送できている。このことは、これらのアプリケーションではトラフィックの規則性が非常に高いことを意味している。

図 9 に、フリットあたりの相対転送レイテンシを示す。1CS, あるいは 3CS チャンネルで拡張手法を用いない場合に、VC 方式に比べておよそ 32% のレイテンシ削減と最も良い結果が得られている。いくつかの場合では、拡張手法ありで適応的に CS チャンネルを割り当てるとかえってレイテンシが悪化している。これは、各エポックの終了時に CS チャンネルを再割り当てする際のオーバーヘッドのためである。また、分割数を増やすとレイテンシが悪化する傾向があるが、これは分割数を増やすに従い、1 サイクルで転送できるビット量が削減され、フリットの全データが送信先に

到達するまでの伝達レイテンシが長くなるためである。そのため、提案手法を用いたにも関わらず、“1VC+7CS+R”では VC フロー制御手法とほとんど同じレイテンシとなってしまう。

図 10 に、フリットあたりの相対消費エネルギーを示す。図より、提案手法を用いることで、CS チャンネル上で転送したトラフィックについてはバッファアクセスの電力が削減され、またスタティック電力も抑えられることから大幅な省エネルギー化が達成されている。VC 方式に比べ、“1VC+3CS”の場合に平均でおよそ 64% もの消費エネルギー削減効果が得られている。一方で、拡張手法は静的割り当てには有効であるが、適応的割り当ての場合ではエネルギーが増大してしまうことがわかる。これは、静的割り当ての場合では拡張手法により CS チャンネル上のトラフィックを増やすことが有効であるが、適応的割り当てではもともと多くのトラフィックを CS チャンネルに割り当てることができている反面、拡張手法により送信元・送信先タイル上でルータ・インターフェース間のバッファリングが必要となり、そこで余分なエネルギーが消費されるためである。

## 6. 関連研究

電力・エネルギー効率は、コンピュータシステムを設計する上での最も重要な指標の一つとして認識されているが、ダークシリコン問題が顕在化するにつれ [7], 近年ではその重要性はさらに増している。プロセッサチップにおいて NoC の電力も無視できず、その性能、およびエネルギー効率向上も重要な研究課題である。

NoC の電力・エネルギー効率向上のための手法がこれ

までに多く研究されてきた。ルータのスタティック電力削減を目的したものとしては、パワーゲーティング [17] や、トラフィックの負荷に応じて電源を供給する手法 [6] などがある。さらに、電力・エネルギー効率以外にも、ルータのレイテンシ削減を目的とした研究も多く行われている [8], [9], [13], [18], [19], [21]。これらの手法は、それぞれの目的において有効ではあるが、フロー制御とリンクのビット幅の割り当てに着目してエネルギー効率の向上を図る手法はこれまでに提案されていない。

CS フロー制御と VC フロー制御の両者を用いるネットワークに関する研究としては、空間分割多重方式によりネットワークのハードウェアを共有しつつ異なるフロー制御を組み合わせる手法 [12] がまず提案された。その後、時分割多重方式 (Time Division Multiplexing) を用い、CS 方式と VC 方式の両者を組み合わせる手法 [3], [24] も提案されている。本稿での提案手法も、文献 [12] と同様に空間分割多重方式をベースとしているが、両者では CS のための経路設定の方法が大きく異なる。従来手法では、ハードウェアとしても軽量の経路設定専用のネットワークを利用し、全パケットを対象として CS フロー制御を利用することを想定している。一方で、本提案手法では、専用ネットワークは持たず、プロファイルにより頻出するパスを抽出しエポックベースで CS データパスを設定することで、従来の NoC とほぼ変わらないハードウェアコストで CS フロー制御を有効に用いることを目的としている。この点で従来手法とは異なるものである。

## 7. おわりに

NoC において、フロー制御手法はネットワークの性能と消費エネルギーに大きく影響する重要な技術要素である。本稿では、VC フロー制御のネットワークにおいて、物理的なリンクのビット線を論理的に複数のビット線に分割し、その一部を CS フロー制御のチャネルとして用い、レイテンシと消費エネルギーを削減する手法を提案した。エポックベースの手法により適応的に CS チャネルに割り当てるパスを選択することで、大幅な消費エネルギー削減と、レイテンシ短縮効果があることがわかった。今後の課題としては、提案手法のためのルータを設計することでハードウェアオーバーヘッドの評価をすること、また、CS の経路設定アルゴリズムを改良し、より性能を向上させることなどがあげられる。

**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) の研究プロジェクト「ポストパタスケールシステムのための電力マネジメントフレームワークの開発」の助成により行われたものである。

## 参考文献

- [1] Agarwal, N. et al.: GARNET: a detailed on-chip network model inside a full-system simulator, *Proc. ISPASS'09*, pp. 33–42 (2009).
- [2] Chen, X. et al.: In-network Monitoring and Control Policy for DVFS of CMP Networks-on-Chip and Last Level Caches, *Proc. 6th NoCS*, pp. 43–50 (2012).
- [3] Cong, J., Gill, M., Hao, Y., Reinman, G. and Yuan, B.: On-chip Interconnection Network for Accelerator-rich Architectures, *Proc. of 52nd DAC*, pp. 8:1–8:6 (2015).
- [4] Dally, W. et al.: *Principles and Practices of Interconnection Networks*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2003).
- [5] Das, R. et al.: Performance and Power Optimization Through Data Compression in Network-on-Chip Architectures, *Proc. 14th HPCA*, pp. 215–225 (2008).
- [6] Das, R. et al.: Catnap: Energy Proportional Multiple Network-on-chip, *Proc. 40th ISCA*, pp. 320–331 (2013).
- [7] Esmaeilzadeh, H. et al.: Dark Silicon and the End of Multicore Scaling, *Proc. 38th ISCA*, pp. 365–376 (2011).
- [8] Hayenga, M. et al.: The NoX router, *Proc. 44th MICRO*, pp. 36–46 (2011).
- [9] He, Y. et al.: McRouter: Multicast Within a Router for High Performance Network-on-chips, *Proc. 22nd PACT*, pp. 319–330 (2013).
- [10] Hesse, R. et al.: Fine-Grained Bandwidth Adaptivity in Networks-on-Chip Using Bidirectional Channels, *Proc. 6th NoCS*, pp. 132–141 (2012).
- [11] Hu, Z. et al.: Microarchitectural techniques for power gating of execution units, *Proc. ISLPED'04*, pp. 32–37 (2004).
- [12] Jerger, N. D. E. et al.: Circuit-Switched Coherence, *Proc. 2nd NoCS*, pp. 193–202 (2008).
- [13] Kumar, A. et al.: Express virtual channels: towards the ideal interconnection fabric, *Proc. 34th ISCA*, pp. 150–161 (2007).
- [14] Li, S. et al.: McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures, *Proc. 42nd MICRO*, pp. 469–480 (2009).
- [15] Magnusson, P. et al.: Simics: a full system simulation platform, *IEEE Computer*, Vol. 35, No. 2, pp. 50–58 (2002).
- [16] Martin, M. M. K. et al.: Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset, *SIGARCH CAN*, Vol. 33, No. 4 (2005).
- [17] Matsutani, H. et al.: Ultra Fine-Grained Run-Time Power Gating of On-chip Routers for CMPs, *Proc. 4th NoCS*, pp. 61–68 (2010).
- [18] Matsutani, H. et al.: Prediction Router: a low-latency on-chip router architecture with multiple predictors, *IEEE TC*, Vol. 60, No. 6, pp. 783–799 (2011).
- [19] Mullins, R. et al.: Low-latency virtual-channel routers for on-chip networks, *Proc. 31st ISCA*, pp. 188–197 (2004).
- [20] NASA Advanced Supercomputing Division: *NAS parallel benchmarks 3.3*, <http://www.nas.nasa.gov/>.
- [21] Peh, L.-S. et al.: A Delay Model and Speculative Architecture for Pipelined Routers, *Proc. 7th HPCA*, pp. 255–266 (2001).
- [22] Sanchez, D. et al.: An analysis of on-chip interconnection networks for large-scale chip multiprocessors, *ACM TACO*, Vol. 7, No. 1 (2010).
- [23] Woo, S. et al.: The SPLASH-2 programs: characterization and methodological considerations, *Proc. 22nd ISCA*, pp. 24–36 (1995).
- [24] Yin, J. et al.: Energy-Efficient Time-Division Multiplexed Hybrid-Switched NoC for Heterogeneous Multi-core Systems, *Proc. 28th IPDPS*, pp. 293–303 (2014).