オンチップトーラス網における仮想チャネルフリールーティング

松谷宏紀[†] 鯉渕道紘[††] 天野英晴[†]

Networks-on-Chip（NoC）ではルータのハードウェア量を必要最低限に抑える必要があり，ルータのパイプライン構造を複雑化させる仮想チャネル機構はルータを小規模化するうえでのネックとなる．しかし，ネットワークトポロジとして広く普及しているトーラス網において，次元順ルーティングなどトポロジの規則性を利用するルーティングを用いる場合，デッドロック回避のために仮想チャネル機構が必要となる．そこで，本論文では，トーラス網において仮想チャネルなしでデッドロックフリーを保証する次元順非最短ルーティング手法を提案する．本手法では，パケットの転送方向を一部 180 度変更することで，仮想チャネルなしで対象の通信パターンにおける循環依存を取り除く．評価では，提案手法を viterbi デコーダを含む 18 種類のアプリケーション・トレースを用いてシミュレーションした．その結果，仮想チャネルを用いない提案手法は，18 個中 11 個のトレースで従来の仮想チャネルルータと同等の性能を達成した．また，提案手法によってルータから仮想チャネル機構を完全に取り除くことができ，ルータのハードウェア量を仮想チャネルルータの 50.7%まで抑えることができた．

A Virtual-channel Free Routing Strategy for On-chip Torus Networks

Hiroki Matsutani,[†] Michihiro Koibuchi[††] and Hideharu Amano[†]

Networks-on-Chips (NoCs) have been employed light-weight routers compared with those in parallel computers, and a virtual-channel mechanism, which requires additional logic and pipeline stages, is one of the crucial factors for a low cost implementation of an NoC router. Although a torus is widely used as a regular network topology, a virtual-channel mechanism is usually required to avoid deadlocks in tori with dimension-order routing that exploits the regularity of the topology. In this paper, we propose a non-minimal strategy for dimension-order routing, in order to guarantee deadlock-freedom without any virtual channels in tori. By inverting the traveling direction of a part of routing paths, the proposed strategy can remove cycle dependancies in the target communications without a virtual-channel mechanism. For the evaluations, the proposed strategy was applied to 18 real application traces including a viterbi decoder. Although the proposed strategy does not use any virtual channels, it achieves almost the same performance as a virtual-channel router on tori in eleven traces. Moreover, the amount of hardware for a router used in the proposed strategy can be decreased to 50.7% of a conventional router providing two virtual channels for tori.

1. はじめに

半導体技術の進歩により，単一チップ上にプロセッサやメモリ，I/O など複数の設計モジュールをタイル状に実装できるようになった．このようなタイルアーキテクチャにおいて，タイルどうしを結合するチップ内結合網はアプリケーションの性能とハードウェア量を決定付ける一要素であり，チップ内ネットワーク（Networks-on-Chip，NoC）[1),2)] が用いられる．NoC は従来チップ内結合網として広く用いられたバスよりも通信帯域に優れ，リンクの距離が限定されるため最近のプロセスで深刻になりつつある配線遅延の問題も解決できる．

タイルアーキテクチャの主要なアプリケーションは組み込み機器であり，メディア処理や情報家電などへの応用が期待される．このような場合，各ノード☆が持つルータの面積が増えると，アプリケーションを実行する計算タイルの面積が圧迫されるので，ルータの面積は極力小さくする必要がある．

タイルアーキテクチャのトポロジとしては，2 次元の平面実装に適した 2 次元メッシュ[3)〜7)] や 2 次元トー

† 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University
†† 国立情報学研究所
National Institute of Informatics

☆ 本論文では，計算タイルとルータの組をノードと呼ぶ．

ラス[2),8)] が代表的である．このようなネットワークで広範囲に利用されている固定型ルーティングとして，次元順ルーティング[9)] があげられる．次元順ルーティングは 2 次元メッシュやトーラスにおいて，まず，送信元ノードから x 軸方向のチャネルを使って移動し，次に y 軸方向のチャネルを使って宛先ノードに到達する．次元順ルーティングは単純な論理回路で実現できるため，経路情報を保持するためのルーティングテーブルをルータに持たせる必要がなく，ハードウェア量の面で有利である．さらに，ユニフォームトラフィックにおいては最短経路を均等に分散させることができ，高い性能を得られる．

さて，トポロジに関して検討すると，トーラスはメッシュに比べて 2 倍の bisection bandwidth を持ち，平均ホップ数の点でも有利である[10)]．しかし，トーラスで次元順ルーティングを用いる場合は，メッシュと異なりデッドロック回避のために仮想チャネル機構が必要となる[9)]．

仮想チャネル機構はルータの設計に大きな影響を与える．仮想チャネルを実装する場合，1 サイクルで同一物理チャネルの複数仮想チャネルから入力されたフリットを切り替えるためには，仮想チャネルごとにバッファが必要となる[10)]．一般的に，ルータはフリットレベルでパイプライン化され，仮想チャネル割当てのアービトレーションのためにパイプラインステージを新設することがある．たとえば，文献 10) のパイプライン化例では，ヘッダフリットは routing computation，virtual-channel allocation，crossbar allocation，crossbar traversal の各ステージを通過する．仮想チャネル機構のためにパイプライン段数が増えると，入力バッファもそれに従い深くなり，ネットワーク遅延も増加する．NoC 向けの単純なルータにおいてはバッファ領域が大きな面積を占めることから，仮想チャネル機構をルータから取り除くことができれば，ルータのハードウェアをさらに小型化することができる．

近年の組み込み機器設計では，対象アプリケーションは SystemC などのシステムレベル言語で記述され，設計の初期段階からシミュレーションされる．そのため，対象アプリケーションごとにノード間の通信パターンを解析することが可能である．本論文では，この解析された通信パターンを利用し，トーラスで次元順ルーティングを用いるルータから仮想チャネル機構を完全に取り除く手法を提案する．この手法は，1) トーラスにおける次元順ルーティングに非最短経路を導入し，2) 通信パターンを解析し，仮想チャネルなしでデッドロックフリーを実現できる経路集合を生成する．

本論文の構成は以下のとおりである．2 章において，非最短経路を含む次元順ルーティングと経路生成アルゴリズムを説明する．3 章の評価では，まず，提案するルータのハードウェア量が従来の仮想チャネルルータより小さいことを示す．次に，提案手法を 18 種類のアプリケーション・トレースを用いてシミュレーションし，仮想チャネルルータに近い性能が出ることを示す．4 章で関連研究を紹介し，5 章で本論文をまとめる．

2. 仮想チャネルフリールーティング

2.1 次元順非最短ルーティング

メッシュでは次元順ルーティングがとりうる経路は一意に定まる．ところが，トーラスでは wrap-around チャネルがあるため，同一次元上の任意の宛先に行く場合，wrap-around チャネルを使う経路と使わない経路の 2 つが存在する．すなわち，n 次元トーラスには最大 2^n 個の代替経路が存在する．**図 1** は，4×4 トーラスにおけるノード $N_{(0,2)}$ から $N_{(2,1)}$ への次元順ルーティングに従った 4 種類の経路を表している．(a) は $x+$ 方向に進んだ後，$y-$ 方向に進み宛先に到達している．同様に，(b) は $x-$ から $y-$，(c) は $x+$ から $y+$，(d) は $x-$ から $y+$ の経路をとる．この例では，(a) と (b) は最短経路，(c) と (d) は非最短経路

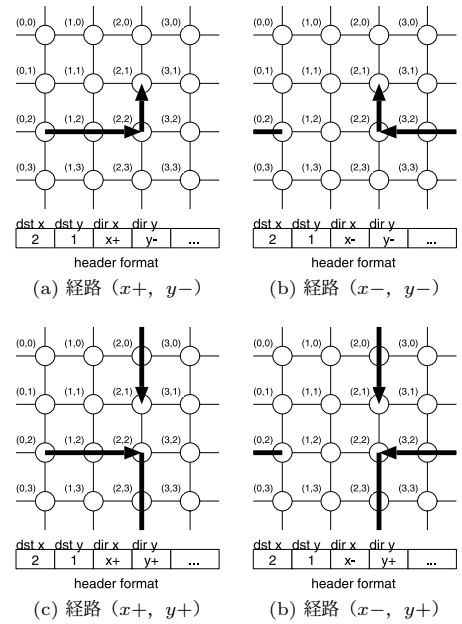

図 1 4×4 トーラスにおける次元順非最短ルーティングの経路．ヘッダ中の dst は宛先ノードの座標を表し，dir は各次元ごとの進行方向を表す

Fig. 1 Routing paths provided by dimension-order non-minimal routing on a 4×4 torus.

である．一般的に非最短経路は性能低下を引き起こすため，次元順ルーティングでは最短経路のみを用い，最短経路の代替経路が複数ある場合はどれか 1 つのみを用いる，もしくは，ランダムに選択する[10]．

次元順ルーティングを用いることで，次元間をまたぐサイクルを断ち切ることができる．ところが，トーラスにおいては同一次元内にサイクルが形成される可能性があり，これを断ち切るために仮想チャネルが利用されてきた．ここで，本論文では仮想チャネルを用いずにこの問題の解決を試みる．本来，次元順ルーティングではすべて最短経路となるが，本論文では，組み込み用途においてあらかじめ通信パターンが分かっていることを前提として，一部の通信に非最短経路を割り当てることで，仮想チャネルなしでデッドロックしない通信パターンを作り出す．この経路生成アルゴリズムは 2.2 節で述べる．このように非最短経路もとりうる次元順ルーティングを，従来の最短経路のみの次元順ルーティングと区別して，**次元順非最短ルーティング**と呼ぶことにする．

次元順非最短ルーティングの実装方法はいくつか考えられるが，ここでは 2 次元トーラスにおけるその一例を紹介する．図 1 に示すように，2 次元トーラスにおいて次元順非最短ルーティングがとりうる経路は最大 4 種類あり，これらは x 次元の進行方向（$x+$ または $x-$）と y 次元の進行方法（$y+$ または $y-$）で区別できる．この各次元ごとの進行方向をヘッダフリットにあらかじめ格納しておき，次元順にこの進行方向に従ってルーティングすることで，次元順非最短ルーティングを実現できる．この方法で次元順非最短ルーティングが実装されたルータのハードウェア量を 3 章で示す．

2.2 経路生成アルゴリズム

次のポリシに基づき経路集合を生成する．

(1) トーラス上で仮想チャネルを用いなくてもデッドロックしないこと．
(2) 非最短経路を導入することによる性能低下を最小限に抑えること．
(3) 現実的な時間（数秒以内）で経路集合を生成できること．

経路集合の生成では，可能なすべての代替経路の組合せの中から最も性能低下が小さいものを全数探索により求める．2.2.2 項で，分枝限定法を用いた経路集合の探索アルゴリズムを説明する．2.2.3 項で，与えられた経路集合がトーラス上でデッドロックフリーか判定するアルゴリズムを示す．

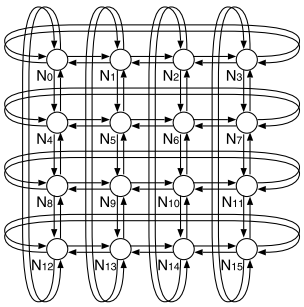

図 2 2 次元トーラスの例
Fig. 2 An example of a 2-D torus.

2.2.1 用　　語

k-ary n-cube における各ノードを N_i と表記する．ただし，$i = \{0, \ldots, k^n - 1\}$ とする．トーラス網において，ある一方向のリンクの集合は単方向サイクルを形成する．このような単方向サイクルをリングと呼び，$x+$ 方向に対しノード N_i を含むリングをリング R_i^{x+} と表記する．同様に，$x-$，$y+$，$y-$ 方向の単方向サイクルについても，それぞれリング R_i^{x-}，R_i^{y+}，R_i^{y-} と表記する．たとえば，**図 2** のトーラスにおいてリング R_0^{x+} は単方向サイクル $N_0 - N_1 - N_2 - N_3 - N_0$ を表す．

2.2.2 経路集合の探索アルゴリズム

与えられた通信パターンにおいて，次元順非最短ルーティングでとりうる代替経路の組合せの中からデッドロックしない経路集合を探索する．このとき，すべての通信で wrap-around チャネルを用いない代替経路を選択すれば通信パターンはメッシュと等価となり，この経路集合は仮想チャネルなしでデッドロックフリーを満たす．したがって，いかなる通信パターンにおいても，デッドロックフリーな解は 1 つ以上存在する．もちろん，wrap-around チャネルの利用を制限すれば性能は低下してしまうため，本手法では性能低下が最小限に抑えられるように一部のソース–ディスティネーション・ペアのみに遠回りな非最短経路を割り当てる．このとき，非最短経路化するソース–ディスティネーション・ペアは注意深く選択しなければならない．文献 11) で議論されているように，多量のデータが流れるソース–ディスティネーション・ペアに非最短経路を割り当てると，ネットワークの帯域を無駄に消費し大幅な性能低下を引き起こしかねない．一方で，少量のデータが流れる通信に非最短経路を割り当てても，ネットワークに及ぼす影響は大きくないと考えられる．そこで，文献 11) で得られた知見をもとに，本論文ではデータ転送量に応じた非最短経路の割当てを行う．

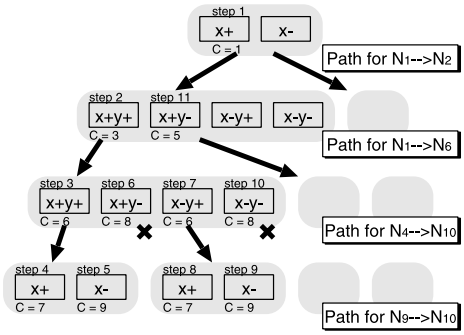

図 3 最適な経路集合を発見するための探索木．4 個のソース–ディスティネーション・ペアのそれぞれに対し，代替経路のいずれかを割り当てている

Fig. 3 An example of a search tree for finding the best set of routing paths.

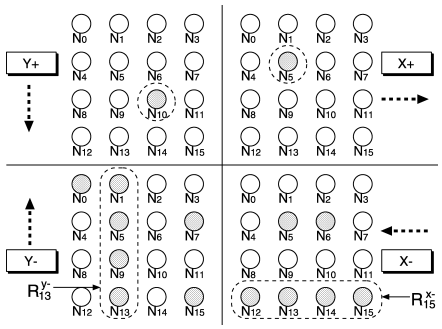

図 4 サイクル検出のためのビットマップ（$x+$，$x-$，$y+$，$y-$）

Fig. 4 Bitmap $x+$, $x-$, $y+$ and $y-$ for the cycle detection.

それぞれの経路集合は下記のコスト関数によって評価される．

$$Cost = \sum_{s=0}^{n-1} \sum_{d=0}^{n-1} H_{(s,d)} \times D_{(s,d)} \qquad (1)$$

ただし，n はノード数，$D_{(s,d)}$ はノード N_s から N_d へのデータ転送量の合計，$H_{(s,d)}$ はノード N_s から N_d へのホップ数とする．このコスト関数によって，データ転送量 D の多いソース–ディスティネーション・ペアほど非最短経路をとりにくく，データ転送量 D の少ない通信ほど非最短経路をとりやすくなる．コストが最小，かつ，デッドロックフリーを満たす経路集合を最適解とする．与えられた経路集合がデッドロックフリーか判定する方法は 2.2.3 項で説明する．

k-ary n-cube で all-to-all 通信を含む場合，とりうる経路集合の組合せは最大 $2^{nk^n(k^n-1)}$ 通り☆となり，現実的な時間で最適解を求めることは難しい．そこで，本論文では分枝限定法を用いて探索空間を枝刈りする．2 次元トーラスを例に枝刈りの様子を**図 3** を用いて説明する．図 3 ではソース–ディスティネーション・ペア $N_1 \mapsto N_2$，$N_1 \mapsto N_6$，$N_4 \mapsto N_{10}$，$N_9 \mapsto N_{10}$ のそれぞれに対し最良な代替経路を選択する．代替経路の数は，各ソース–ディスティネーション・ペアごとに最大 4 種類（$x+y+$，$x+y-$，$x-y+$，$x-y-$）だが，$N_1 \mapsto N_2$ と $N_9 \mapsto N_{10}$ は y 次元上の座標が同じため代替経路は $x+$ と $x-$ の 2 種類だけとなっている．図中の C は式 (1) によって求めた経路集合のコストである．ここでは単純化のため，データ転送量 D は一律 1 とした．図中の step 1 から順に探索を開始し，step 4 でペア $N_1 \mapsto N_2$ に経路（$x+$），$N_1 \mapsto N_6$ に経路（$x+y+$），$N_4 \mapsto N_{10}$ に経路（$x+y+$），$N_9 \mapsto N_{10}$ に経路（$x+$）が割り当てられる．このときのコストは 7 である．この後，step 6 でコストが 8 となるが，現状の最適解のコストは 7 なのでこれ以上探索しても最適解は得られない．よって step 6 で枝刈りが起きる．また，探索の途中でデッドロックが検出された場合もこれ以深の探索は行わない．このように探索木の枝を刈ることで計算時間を大幅に減らすことができる．なお，経路集合の探索に要す時間については 3 章で評価する．

☆ すべてのソース–ディスティネーション・ペアが 2^n 個の代替経路を持つ場合．ある次元上で送信元ノードと宛先ノードの座標が同じときは，代替経路の数が $2^{(n-1)}$ に減るので，実際の探索空間はこれよりも小さい．

2.2.3 サイクル検出アルゴリズム

経路集合 P によってサイクルが形成されるか判定する．

(1) （ビットマップ初期化）各方向（$x+$，$x-$，$y+$，$y-$）ごとに，ノード数分のビットマップを用意する（**図 4**）．

(2) （マーク付け）送信元ノード N_s と宛先ノード N_d のすべての組について：

(a) データ転送量 $D_{(s,d)}$ が 0 のときは (2) に戻る．

(b) N_s から N_d への経路 $P_{(s,d)}$ を求める．たとえば $P_{(4,14)} = \{N_4, N_5, N_6, N_{10}, N_{14}\}$ とする．

(c) 経路 $P_{(s,d)}$ から，送信元ノード，宛先ノード，ターンの支点ノードを取り除く．経路 $P_{(4,14)}$ の場合，送信元ノード N_4，宛先ノード N_{14}，ターンの支点ノード N_6 が取り除かれ，$P'_{(4,14)} = \{N_5, N_{10}\}$ となる．

(d) $P'_{(s,d)}$ のノードごとに，各ノードの進行方向に対応するビットマップにマークを

付ける．経路 $P'_{(4,14)}$ の場合，$x+$ ビットマップのノード N_5，$y+$ ビットマップのノード N_6 がマーク付けされる（図 4）．

(3) （サイクル検出）各リングごとに，リング上のすべてのノードがマークされているか調べる．マークが揃わない場合は，そのリングはサイクルフリーである．これに関しては以下で証明する．図 4 では，リング R_{15}^{x-} と R_{13}^{y-} でサイクルが形成されている．

(4) （デッドロックフリー判定）トーラス上のすべてのリングでサイクルが形成されないとき，経路集合 P はデッドロックフリーである．

定理 あるリング上のすべてのノードがマークされないとき，そのリング上でサイクルは形成されない．

証明 この定理をリング R_i^{x+} で証明する．他のリングについても同じ方法で証明できる．送信元ノード，宛先ノード，ターン支点ノードはビットマップ上にマーク付けされない．そのため，$x+$ ビットマップ上のノード N_i がマークされるとき，N_i の $x+$ チャネルが開放されるまで，N_i の $x-$ チャネルを占有するパケットが 1 つ以上存在する．リング上のあるノードがマークされないとき，そのノードの $x+$ チャネルの開放を待つパケットは存在しない．このようなチャネルによってリング上のサイクルは断たれるので，マークが揃わないリング上ではサイクルは形成されない．

∎

本手法に従って次元順ルーティングに非最短経路を導入することで，デッドロックフリーな経路集合を仮想チャネルなしで得られる．

2.2.4 探索アルゴリズムの高速化

経路生成をさらに高速化するために以下の手法を併用した．

- 通信量の多いソース–ディスティネーション・ペアを探索木のルート付近に集め，探索の早い段階で枝刈りが起きやすくする．
- 通信量がとくに多いソース–ディスティネーション・ペアに非最短経路を割り当てないようにし，探索空間を縮小させる．なお，非最短経路の割当てをどの程度制限するかは，経路生成のために割ける時間に応じて可変にした．

3. 評　　価

次元順非最短ルーティングに対応したルータを実装し，仮想チャネルを持つ従来のルータよりハードウェア量の点で有利なことを示す．次に，提案手法によって仮想チャネルを用いなくても仮想チャネルルータに近い性能が出ることを示す．

3.1 ルータの面積

2 次元トーラス向けに仮想チャネルを 2 個持つルータ，仮想チャネルを持たないが非最短経路を利用できるルータ，仮想チャネルを持たないメッシュ向けルータの面積をそれぞれ見積もった．これらのルータには次元順ルーティングが専用ロジックとして実装されており，ルーティングテーブルは持たない．一方，比較のためルーティングテーブルを持つルータも実装した．各入力ポートからのルーティングテーブル参照を独立に処理するために，ポートごとに経路情報を分割して持たせる．このような実装は文献 10) のルーティング関係式 $C \times N \mapsto C$ に対応している．

3.1.1 ルータの構造

図 5 に今回実装した wormhole ルータの構造を示す．このルータは 1 つのクロスバ（CB5x5）と 5 つの物理ポートから構成される．1 つのポートは計算タイルとの接続用に，残りのポート $x+$，$x-$，$y+$，$y-$ は隣接ルータとの接続に用いられる．仮想チャネルの実装では，p をポート数，v を仮想チャネル数としたとき，$pv \times pv$ フルクロスバを用いることもある．しかし，$pv \times pv$ クロスバはポート数が増えるとハードウェア量が大きくなるにもかかわらず，データ転送レートは物理チャネルの帯域によって頭打ちになるので性能向上も限定的であることが報告されている[12]．そこで，本論文ではバッファのみ仮想チャネル分持たせ，クロスバは小型な $p \times p$ を採用して公平な比較を行った．

このルータでは仮想チャネル数は可変とした．以降，仮想チャネルを 2 個持つ設計を “v2”，仮想チャネルを持たない設計を “v1” と表記する．各ポートは入力バッファ（InBuf），ルーティング回路（RtComp），クロスバコントローラ（CbCont）から構成される．ヘッダフリットに格納されている宛先アドレスに応じて，

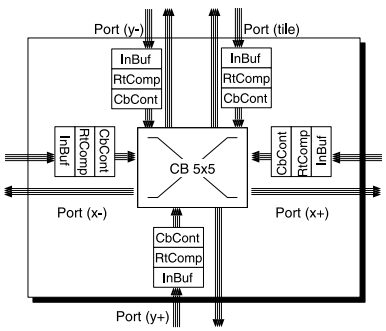

図 5 実装した wormhole ルータ（5 ポート）
Fig. 5 A wormhole router with 5 ports.

RtComp が出力ポートを計算する．CbCont が出力ポートのアービトレーションを行い，クロスバスイッチを切り替えてデータを転送する．仮想チャネルを持つ v2 ルータは，routing computation，virtual-channel allocation，crossbar allocation，crossbar traversal の 4 段パイプライン構成をとる．一方，仮想チャネルを持たない v1 ルータは，virtual-channel allocation ステージが必要ないので 3 段パイプライン構成とした．フリットのデータ幅は 32-bit とし，ルータの各チャネルには 1-flit 分のバッファを持たせた．

3.1.2 合成結果

本論文では wormhole 方式を用いる固定型ルーティングを対象とする．virtual-cut through（VCT）方式のほうが wormhole より高性能ではあるが，VCT では各ルータに 1 パケット分のデータが丸々収まるバッファが必要となる．wormhole ルータと，1 パケットを 16-flit として実装した VCT ルータの面積を参考までに **図 6** に示す☆．このグラフより，VCT ルータの面積は wormhole ルータの面積よりもかなり大きいことが分かる．また，適応型ルーティングではパケット転送の FIFO 性を保証できないため，VCT ルータよりさらに大きなバッファ領域がパケットのリオーダリング処理のために必要となることがある[5]．そのため，以降の評価では wormhole 方式および固定型ルーティングの使用を前提とする．

本研究で提案したルータを含む以下の wormhole ルータの面積を比較する：**DOR+v1** はメッシュ向けに次元順ルーティングがロジックとして実装されたルータで仮想チャネルは持たない．**DOR+v1+N** はトーラス向けに次元順非最短ルーティングが実装され，仮想チャネルは持たない（“+N” は Non-minimal の略）．**DOR+v2** は 2 次元トーラス向けに次元順ルーティングが実装され，仮想チャネルを 2 個持つ．**CxN+v1** は $C \times N$ 型ルーティングテーブルを持つルータで，**CxN+v2** ではさらに仮想チャネルを 2 個持つ．以上に加え，文献 11) の *flee* ルーティング法に対応したルータもそれぞれ **Flee+v1**，**Flee+v2** として実装した．*flee* では経路情報をルータに格納する必要があり，ルーティングの実装にはソースルーティングを用いた．

上記の 7 種類のルータの面積を**図 7** に示す．これらは可能な限り面積が小さくなるように設計してある．面積の見積りには 0.35 μm ASIC ライブラリを用い，

☆ 0.35 μm ASIC ライブラリを使用．VCT ルータの I/O バッファには RAM マクロの使用を仮定した．

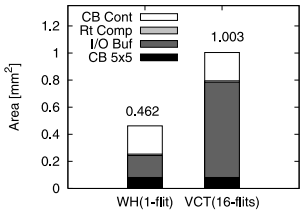

図 6 wormhole と VCT ルータの面積
Fig. 6 Hardware amount of wormhole and VCT routers.

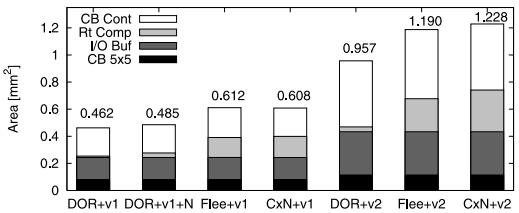

図 7 v1 および v2 の各種ルータの面積
Fig. 7 Hardware amount of v1 and v2 routers.

動作速度は最大で 96 MHz 程度となった．DOR ルータはルーティングテーブルを持たないが，CxN ルータは経路情報を保持するためのレジスタファイルを持つため，CxN+v1 ルータは DOR+v1 より 31.6%面積が増加した．同様に，Flee+v1 ルータの面積も DOR+v1 より増大した．

次に，DOR+v1 と DOR+v2 の面積を比較し，仮想チャネル機構の実装コストについて考察する．グラフに示すように，DOR+v2 ルータの面積は DOR+v1 の約 2 倍まで増加した．面積増加分の内訳を見ると，virtual-channel allocation のための新たなパイプラインステージと，仮想チャネルごとのバッファ領域が大半を占めていた．このように，ルータから仮想チャネル機構を取り除くことで，パイプライン構造を単純化でき，ルータの面積を大幅に小型化できることが分かった．

仮想チャネルを持たない次元順非最短ルータ（DOR+v1+N）の面積を，従来の仮想チャネルルータと比較する．グラフに示すとおり，DOR+v1+N の面積は 2 次元トーラス向けに仮想チャネルを 2 個持つルータの 50.7%まで抑えることができた．DOR+v1+N の面積は，メッシュ向けに仮想チャネルを持たないルータ（DOR+v1）より約 5%増えたものの，仮想チャネル機構の追加に比べると面積の増分ははるかに小さい．なお，本論文の DOR+v1+N は，進行方向を指定しない従来のルーティングモードと，ヘッダフリットに格納された進行方向に従う非最短モードの両方に対応している．

3.1.3 タイル面積との比較

本項では実アプリケーションを用いて，タイルの面積とルータの面積を比較する．ここでは比較的単純なストリーム処理の一例として viterbi デコーダを用いる．この viterbi デコーダは，NEC エレクトロニクス製の動的リコンフィギャラブルプロセッサ DRP [13] 向けに実装されたもので，各タスクの配置は**図 8** のようになっている．

ルータ同様 0.35 μm ASIC ライブラリを用いて viterbi デコーダの各タイルの面積を見積もった（**表 1**）．ここで，ルータとタイルを合わせた面積をノードの面積とし，ノード全体の面積に占めるルータ面積の割合を**ルータ占有率**と定義する．表 1 に示すとおりタイルの面積はそれぞれ異なるが，DOR+v2 ルータを用いるとルータ占有率は平均で 53.1% となる．一方，本研究で提案した DOR+v1+N ルータでは，仮想チャネル機構を取り除くことで，ルータ占有率の平均値を 36.5%まで引き下げることができた．

プロセスが微細化すれば，同一エリア上により多くの回路を配置できるようになる．したがって，ルータの機能が同一であれば，プロセスの微細化にともないルータ占有率は相対的に低下する．12 mm 角のチップで 0.10 μm プロセスを用いた場合，ルータ占有率は 6.6%まで抑えられるとの報告もある [2]．これほど大規模なアプリケーションに至っては，仮想チャネルフリーの利点は薄れ，仮想チャネルを要求する通常の次元順ルーティングがコストパフォーマンスの点で優位に立つ可能性もある．しかし，今回の viterbi デコーダのような小規模アプリケーションにおいては，ルータ占有率の高さゆえに，仮想チャネルの実装コストを無視できない．

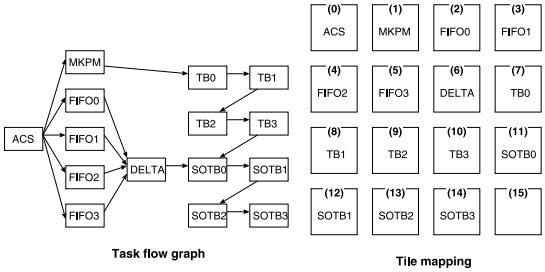

図 8 Viterbi デコーダのタスクの流れとノードへの割当て．入力データは Add-Compare-Select（ACS）回路に転送される．ACS は Branch Metric を生成し FIFO に格納する．ACS はさらに Path Metric を生成する．Make Path Metric（MKPM）で最も確からしい状態が選ばれ，DELTA で最も確からしい Branch Metric が選ばれる．最後に，Trace Back（TB）と Soft Output Trace Back（SOTB）で復号結果を生成する

Fig. 8 Task flow graph and mapping result for viterbi decoder.

表 1 Viterbi デコーダにおける各タイルの面積（mm^2）

Table 1 Hardware amount of each tile on viterbi decoder (mm^2).

Tile	Area	Tile	Area	Tile	Area
ACS	1.180	MKPM	0.300	DELTA	0.039
TB 0	0.372	FIFO 0	2.168	SOTB 0	0.247
TB 1	0.372	FIFO 1	2.168	SOTB 1	0.247
TB 2	0.372	FIFO 2	2.168	SOTB 2	0.247
TB 3	0.372	FIFO 3	2.168	SOTB 3	0.247

3.2 スループット

提案手法を含む以下のルータごとにスループットを比較する：**Mesh+v1** はメッシュ向けに仮想チャネルを持たないルータで，**Torus+v2** は 2 次元トーラス向けに仮想チャネルを 2 個持つ．**Torus+v1+N** はトーラス向けだが仮想チャネルは持たず，次元順非最短ルーティングに対応する．

3.2.1 シミュレーション環境

上記のルーティング環境ごとに，デッドロックフリーを確認しスループットを測定するため C++言語で記述されたフリットレベル・シミュレータを用いる．シミュレータでは，各ルータは 5 つのポートを持ち，1 つは計算タイルとの接続に，残りは隣接ルータとの接続に使用する．また，各ルータのスイッチング機構として，I/O バッファ，クロスバ，クロスバコントローラを単純化したモデルを採用し，ヘッダフリットが隣接ルータや計算タイルに転送されるのに 3 サイクルかかるものとする．パケット転送方式として wormhole 方式を用い，パケット長はヘッダの 1-flit 分を含め 16-flit とした．

シミュレーションには実アプリケーションから得られた通信パターンを用いる．NoC の主要なアプリケーションの 1 つにストリーム処理がある．本評価では，典型的なストリーム処理として 3.1.3 項で紹介した viterbi デコーダの通信パターンを用いる．しかし，現状ではこのようなストリーム処理は比較的小規模（16 ノードなど [6),11)]）なものが多く，かつ，通信パターンも単純なため，ここにあげたアプリケーションだけでは提案手法の網羅的な評価はできない．そこで，本評価では NAS Parallel Benchmark（NPB）[14] プログラムから得られた通信パターンも用いる．NPB の各プログラムは数値計算を扱う並列アプリケーションであるが，これらの通信パターンにはストリーム処理で頻出する fork/join に似た通信パターンが含まれる．今回，NPB から次のプログラムを用いる：Block Tridiagonal solver（BT），Scalar Pentadiagonal solver（SP），Conjugate Gradient（CG），Multi-Grid solver（MG），large Integer

表 2 経路集合の計算時間（秒）（@Intel Pentium4 2.6 GHz）
Table 2 Calculation time for finding the routing path-set (seconds).

Application	Time	Application	Time
Viterbi (16 node)	0.005	BT (9 node)	0.014
BT (16 node)	0.030	BT (36 node)	4.798
BT (64 node)	1.018	SP (9 node)	0.015
SP (16 node)	0.011	SP (36 node)	5.404
SP (64 node)	0.130	CG (16 node)	0.011
CG (32 node)	0.196	CG (64 node)	0.298
MG (16 node)	0.011	MG (32 node)	5.354
MG (64 node)	0.427	IS (16 node)	0.865
IS (32 node)	19.901	IS (64 node)	3,600.000

Sort（IS）．プログラムのクラスは “W” とし，ノード数は 9，16，32，36，64 とした．評価に用いるトレースは，5 種類の NPB プログラムから得られたトレース 17 種類に viterbi デコーダのトレースを加え合計 18 種類となった．

以上のトレースデータの時間軸を変化させることで，様々な注入データレートに対するスループットとレイテンシの評価を行う．これは，ノード内の処理時間をネットワークに対して相対的に変化させた場合の評価にあたる．ノード内の処理時間には，アプリケーションの計算時間のほかに，パケットの生成，解体によるオーバヘッドも含まれる．

Torus+v1+N 向けには，2.2 節で示した方法を用いて，仮想チャネルなしでデッドロックしない経路集合を生成する．経路集合の生成に要した計算時間を**表 2** に示す．all-to-all 通信を含む IS を除き，計算時間は無視できるほど小さい．

3.2.2 シミュレーション結果

上記の 18 種類のアプリケーション・トレースを用いて，Mesh+v1，Torus+v2，Torus+v1+N の各ルーティング環境におけるスループットとレイテンシを測定した．

図 9 に viterbi トレースを用いた際の，Mesh+v1，Torus+v2，Torus+v1+N におけるスループット（accepted traffic）とレイテンシのグラフを示す．このトレースは文献 11) で *flee* の評価に用いた viterbi デコーダをもとに生成した．そこで，仮想チャネルを用いず up*/down*ルール[15] をもとにした *flee*（Torus+v1+F）と，ターンモデル・ベース[16] で仮想チャネルを用いる *flee*（Torus+v2+F）の結果もグラフに加えた．それぞれの平均ホップ数は括弧内に示してある．図 9 に示すとおり，Torus+v2 は wrap-around チャネルを使うことで Mesh+v1 よりも性能的に優れる．Torus+v1+N は wrap-around チャネルを活用するが，非最短経路を導入しなくても仮想チャネルなしでデッドロックフリーを実現でき，Torus+v2 と同等の性能が得られた．この viterbi デコーダにおいては通信距離が十分小さくなるようにマッピングされているため隣接間通信が多い．隣接間通信はホップ数が 1 なので，サイクルの形成に影響を与えない．以上の理由から，この viterbi トレースでは非最短経路を導入しなくてもサイクルが形成されなかった．なお，*flee* に関しては Torus+v2+F が高い性能を実現している一方，Torus+v1+F の性能は Torus+v1+N に劣っている．

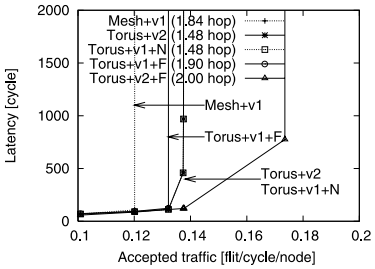

図 9 Viterbi トラフィック（16 ノード）
Fig. 9 Viterbi traffic (16-nodes).

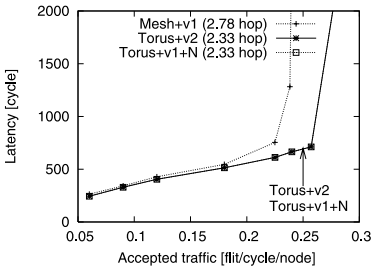

図 10 BT トラフィック（9 ノード）
Fig. 10 BT traffic (9-nodes).

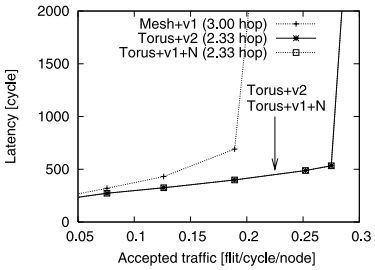

図 11 BT トラフィック（16 ノード）
Fig. 11 BT traffic (16-nodes).

次に NPB トレースでの評価に移る．**図 10**，**図 11**，**図 12**，**図 13** に 9，16，36，64 ノードで BT トレースを用いた際の結果を示す．この BT トレースでは，平均ホップ数はネットワークの規模によらず比較的小さい．そのため，9，16，64 ノードでは非最短経路を導入しなくても，Torus+v1+N でデッドロックフリー

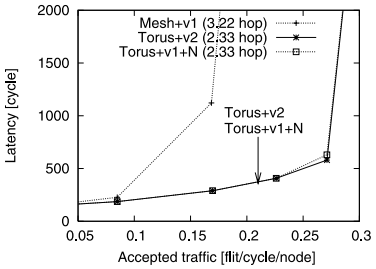

図 12 BT トラフィック（36 ノード）
Fig. 12 BT traffic (36-nodes).

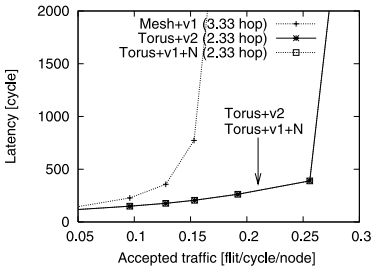

図 13 BT トラフィック（64 ノード）
Fig. 13 BT traffic (64-nodes).

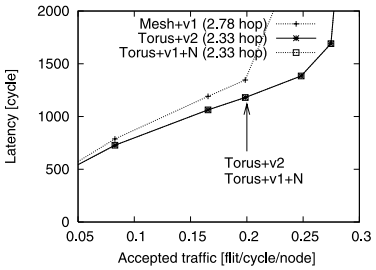

図 14 SP トラフィック（9 ノード）
Fig. 14 SP traffic (9-nodes).

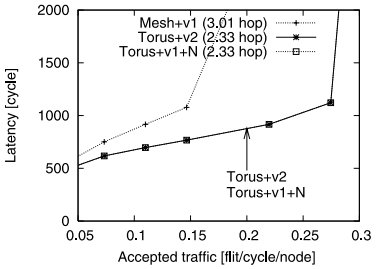

図 15 SP トラフィック（16 ノード）
Fig. 15 SP traffic (16-nodes).

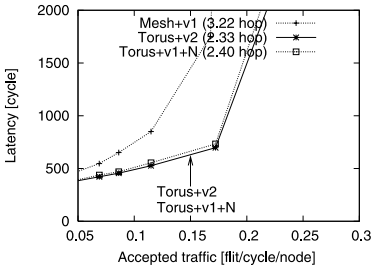

図 16 SP トラフィック（36 ノード）
Fig. 16 SP traffic (36-nodes).

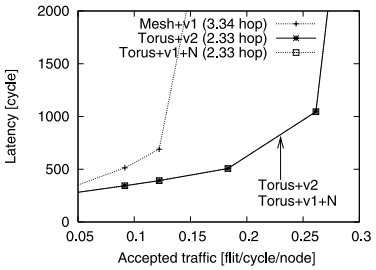

図 17 SP トラフィック（64 ノード）
Fig. 17 SP traffic (64-nodes).

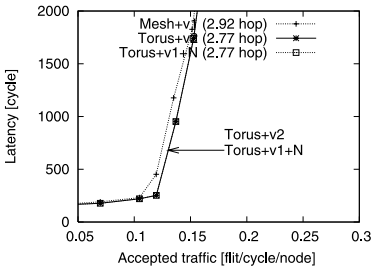

図 18 CG トラフィック（16 ノード）
Fig. 18 CG traffic (16-nodes).

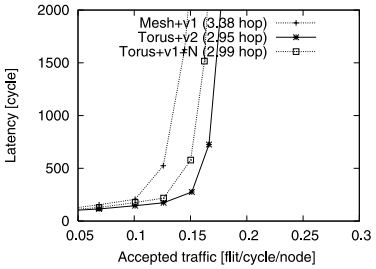

図 19 CG トラフィック（32 ノード）
Fig. 19 CG traffic (32-nodes).

を実現でき，性能は Torus+v2 と同等となった．一方，36 ノードの Torus+v1+N では，次元順ルーティングの進行方向を一部変更することでデッドロックを回避している．それでも，Torus+v1+N と Torus+v2 の性能差はほとんどない．

図 14，**図 15**，**図 16**，**図 17** は 9，16，36，64 ノードでの SP トラフィックの結果である．SP と BT は類似したアルゴリズムをもとにしている[14] ため通信パターンが似ており，SP の結果には BT と同じ傾向が見られた．

図 18，**図 19**，**図 20** に 16，32，64 ノードでの CG トラフィックの結果を示す．この CG では，

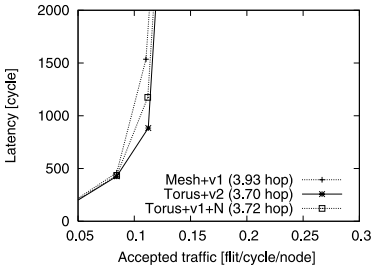

図 20 CG トラフィック（64 ノード）
Fig. 20 CG traffic (64-nodes).

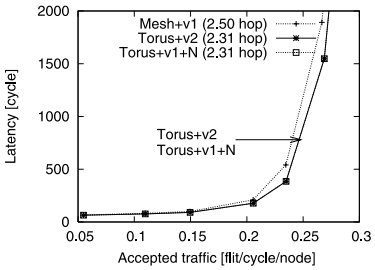

図 21 MG トラフィック（16 ノード）
Fig. 21 MG traffic (16-nodes).

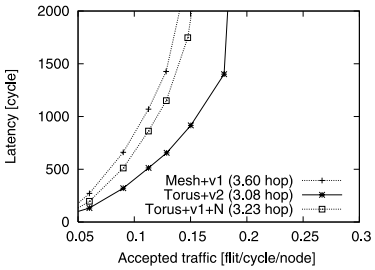

図 22 MG トラフィック（32 ノード）
Fig. 22 MG traffic (32-nodes).

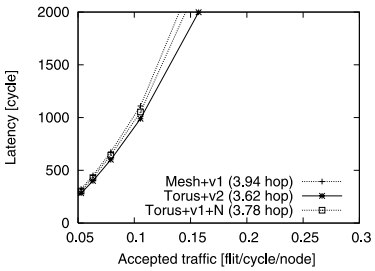

図 23 MG トラフィック（64 ノード）
Fig. 23 MG traffic (64-nodes).

Torus+v2 においても wrap-around チャネルがあまり活用されておらず，Mesh+v1 と Torus+v2 の平均ホップ数の差は小さい．よって，Mesh+v1，Torus+v2，Torus+v1+N の性能差は限定的となった．

図 21，**図 22**，**図 23** に 16，32，64 ノードでの MG トラフィックの結果を示す．紙面の都合から詳細は省略するが，CG トラフィックに似た傾向が見られた．

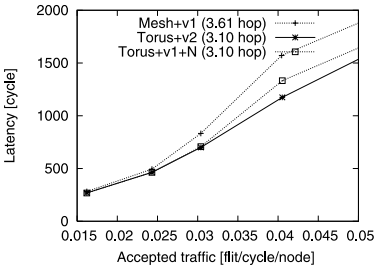

図 24 IS トラフィック（16 ノード）
Fig. 24 IS traffic (16-nodes).

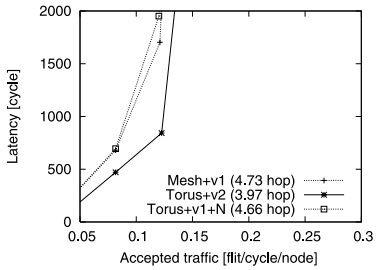

図 25 IS トラフィック（32 ノード）
Fig. 25 IS traffic (32-nodes).

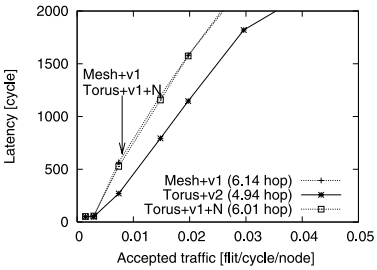

図 26 IS トラフィック（64 ノード）
Fig. 26 IS traffic (64-nodes).

図 24，**図 25**，**図 26** に 16，32，64 ノードでの IS トラフィックの結果を示す．IS トレースには all-to-all 通信が含まれており，非最短経路を多数導入しない限り，Torus+v1+N でデッドロックを回避することは難しい．実際，32 ノードと 64 ノードの結果では，Torus+v1+N の平均ホップ数は Mesh+v1 に近いものとなり，Mesh+v1 並みの性能しか得られていない．

一方，16 ノードでは Torus+v1+N の性能は Torus+v2 にやや劣るものの平均ホップ数は同じとなった．以下，この理由を**図 27** を用いて説明する．この図は，4×4 トーラスのリング R_0^{x+} における経路割当ての例である．$N_0 \mapsto N_1$，$N_0 \mapsto N_2$，$N_1 \mapsto N_2$，$N_1 \mapsto N_3$，$N_2 \mapsto N_3$，$N_3 \mapsto N_0$ の各経路が図示されている．これらの経路はどれも最短経路だが図 27 でサイクルは形成されていない．この例では同一ホッ

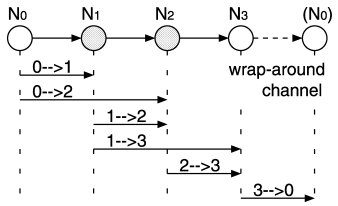

図 27 4 × 4 トーラスのリング R_0^{x+} における経路集合の例
Fig. 27 Routing paths for ring R_0^{x+} on a 4 × 4 torus.

プ数で複数の代替経路が存在するときに，サイクルが形成されない組合せで経路を選ぶことでデッドロックを回避している．このように経路を選択することで，Torus+v1+N は非最短経路を導入しなくてもデッドロックフリーを実現できた．

以上の評価をとおして，Torus+v1+N の性能は Mesh+v1 以上であり，18 個中 11 個のアプリケーション・トレースで Torus+v2 と同等の性能を達成した．NoC の主用途である組み込み分野では，動作させるアプリケーションは設計時に決まっており，アプリケーションの通信パターンを解析したうえで経路を選択できる．本研究で提案した手法を用いてアプリケーションを解析すれば，仮想チャネルを用いなくても，仮想チャネルルータに匹敵する性能を実現できることが分かった．

4. 関 連 研 究

トーラス上で次元順ルーティングを用いるルータから仮想チャネルを取り除く方法として bubble flow control [17] が提案されている．bubble flow control はフロー制御によるパケットの注入制限の一種である．リング上に最低 1 個のパケットを格納できるだけのバッファ（bubble）を空けておくことで，リング上にサイクルが形成されるのを防ぐ．この bubble を作るためにルータはパケット全体を溜め込む必要があるので，この方法は VCT 方式で利用される．VCT ではパケット全体を格納するバッファが必要となり，ルータの面積は増大する．一方，本研究の対象は組み込み用途に応用される NoC なので，VCT ではなく軽量な wormhole 方式を想定し，また，設計時に得られる通信パターンを活用してデッドロックフリーな通信パターンを生成するという点で NoC に特化している．

仮想チャネルが不要という点では，不規則ネットワーク向けに提案された up*/down*ルーティングを利用できる．ただし，up*/down*ルーティングではスパニング木のルート付近にトラフィックが集中するという問題をかかえている．文献 18) はスパニング木を複数持たせることでこの問題を緩和することを提案しているが，スパニング木ごとに別々の仮想チャネルを割り当てる必要があるため NoC における仮想チャネル問題の解決にはならない．しかも，up*/down*ルーティングの実装にはルーティングテーブルが要る．

本論文の次元順非最短ルーティングは，ソースルーティング実装の応用によって，進行方向を指定できる点を利用して容易に実装できる．トラフィック分散のための経路設定手法として Valiant アルゴリズム [10] が提案されている．Valiant アルゴリズムでは，まず，送信元ノードからあらかじめ選択しておいた中継ノードまでパケットを転送し，次に，この中継ノードを起点に宛先ノードまで転送する．本提案手法は中継ノードを設定せず，次元順にパケットを転送する規則に従いながら経路を柔軟に設定している点で，Valiant アルゴリズムとは異なる．また，本論文では，デッドロックフリーな通信パターンを仮想チャネルなしで得るためにパケットの進行方向を制御している点で，そもそも目的が異なる．

NoC において，設計時に得られる通信パターンを解析することで，アプリケーションに適したトポロジを生成する研究 [19]，マッピングを生成する研究 [5]，ルーティングを最適化する研究 [3),11)] が行われてきた．とくに文献 11) の *flee* ルーティング法では，通信データ量に応じた非最短経路の割当てによって，局所性の強いストリーム処理で次元順ルーティングを上回る性能を実現した．本論文の仮想チャネルフリールーティングでは，この *flee* の研究で得られた知見を活かし，仮想チャネルなしでデッドロックしない通信パターンを生成している．

トーラスにおいて仮想チャネルが利用できなければ，次元順ルーティングを内包するターンモデルなどトーラス構造により適したルーティングは利用できない．それでも，up*/down*ルーティングなどの不規則ネットワーク向けルーティングをもとに *flee* を用いて経路を生成すれば，*flee* でトーラスにおける仮想チャネルフリーを実現することもできる．ただし，3.2.2 項で示したとおり，up*/down*ルール・ベースの *flee* はターンモデル・ベースのものより性能が出ないため，*flee* の活用法としては適していない．さらに，3.1.2 項で示したとおり，*flee* ではルータに経路情報を持たせる必要があり，ハードウェア量の点でも不利となる．一方，本提案はこれらの問題点を解決し，さらには既存のメッシュ/トーラスを採用した NoC [7),8)] で次元順ルーティングが利用されている点に注目し，この既存の枠組みで活用できる現実的なアプローチをとった点も 1 つの特徴といえる．

5. まとめ

本論文では，トーラス上で次元順ルーティングを用いるルータから仮想チャネル機構を完全に取り除くための手法を提案した．この手法は，1) 次元順ルーティングに非最短経路を導入し，2) 通信パターンを解析し，仮想チャネルなしでデッドロックフリーを実現できる経路集合を生成する．

1) ではルーティングテーブルが不要な次元順ルーティングを維持しつつ非最短経路を導入するために，各次元ごとに進行方向を指定できるようにした．仮想チャネル機構の代わりに非最短経路に対応したルータを実装し，従来の仮想チャネルルータの 50.7%の面積で実現できることを示した．

非最短経路化するソース–ディスティネーション・ペアは 2) の手法を用いて選択する．得られた経路集合の性能をシミュレーションしたところ，18 個中 11 個のアプリケーション・トレースで従来の仮想チャネルルータと同等の性能が得られた．また，現状の NoC で用いられる 16 ノード程度の小規模な構成では，all-to-all 通信が含まれていても非最短経路なしでサイクルフリーな通信パターンを生成できることを確認した．さらに，経路集合の探索時間はほとんどのアプリケーションで無視できるほど小さく，本手法を容易に適用できることを示した．

謝辞 本研究の一部は，国立情報学研究所共同研究「ネットワークオンチップのアーキテクチャに関する研究」による．本研究の一部は，東京大学大規模集積システム設計教育研究センターを通し，シノプシス株式会社の協力で行われたものである．本論文で用いた viterbi デコーダの実装を提供してくださったノキア・ジャパン株式会社金子直人氏に感謝いたします．

参考文献

1) Benini, L. and Micheli, G.D.: Networks on Chips: A New SoC Paradigm, *IEEE Computer*, Vol.35, No.1, pp.70–78 (2002).
2) Dally, W.J. and Towles, B.: Route Packets, Not Wires: On-Chip Interconnection Networks, *Proc. 38th Design Automation Conference*, pp.684–689 (2001).
3) Anjo, K., Yamada, Y., Koibuchi, M., Jouraku, A. and Amano, H.: BLACK-BUS: A New Data-Transfer Technique using Local Address on Networks-on-Chips, *Proc. IEEE International Parallel and Distributed Processing Symposium*, p.10a (2004).
4) Burger, D., Keckler, S.W., McKinley, K.S., Dahlin, M., John, L.K., Lin, C., Moore, C.R., Burrill, J., McDonald, R.G., Yoder, W. and the TRIPSTeam: Scaling to the End of Silicon with EDGE Architectures, *IEEE Computer*, Vol.37, No.7, pp.44–55 (2004).
5) Hu, J. and Marculescu, R.: Energy- and Performance-Aware Mapping for Regular NoC Architectures, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.24, No.4, pp.551–562 (2005).
6) Liang, J., Laffely, A., Srinivasan, S. and Tessier, R.: An Architecture and Compiler for Scalable On-Chip Communication, *IEEE Trans. Very Large Scale Integration Systems*, Vol.12, No.7, pp.711–726 (2004).
7) Taylor, M.B., et al.: The Raw Microprocessor: A Computational Fabric for Software Circuits and General Purpose Programs, *IEEE Micro*, Vol.22, No.2, pp.25–35 (2002).
8) Marescaux, T., Bartic, A., Verkest, D., Vernalde, S. and Lauwereins, R.: Interconnection Networks Enable Fine-Grain Dynamic Multi-Tasking on FPGAs, *Proc. Field-Programmable Logic and Applications* (*FPL*), pp.795–805 (2002).
9) Dally, W.J. and Seitz, C.L.: Deadlock-Free Message Routing in Multiprocessor Interconnection Networks, *IEEE Trans. Computers*, Vol.36, No.5, pp.547–553 (1987).
10) Dally, W.J. and Towles, B.: *Principles and Practices of Interconnection Networks*, Morgan Kaufmann (2004).
11) 松谷宏紀，鯉渕道紘，山田 裕，上樂明也，天野英晴：非最短経路を用いたチップ内ネットワーク向け経路設定手法，情報処理学会論文誌：コンピューティングシステム，Vol.46, No.SIG 12, pp.73–83 (2005).
12) Dally, W.J.: Virtual-Channel Flow Control, *IEEE Trans. Parallel and Distributed Systems*, Vol.3, No.2, pp.194–205 (1992).
13) NEC エレクトロニクス：動的再構成プロセッサ Dynamically Reconfigurable Processor. http://www.necel.com/ja/techhighlights/drp/
14) Bailey, D., Harris, T., Saphir, W., Wijngaart, R., Woo, A. and Yarrow, M.: The NAS Parallel Benchmarks 2.0, NAS Technical Report, NAS-95-020 (1995).
15) Schroeder, M.D., Birrell, A.D., Burrows, M., Murray, H., Needham, R.M. and Rodeheffer, T.L.: Autonet: A High-speed, Self-configuring Local Area Network Using Point-to-point Links, *IEEE Journal on Selected Areas in Communications*, Vol.9, pp.1318–1335 (1991).
16) Glass, C.J. and Ni, L.M.: The Turn Model

for Adaptive Routing, *Proc. International Symposium on Computer Architecture*, pp.278–287 (1992).
17) Puente, V., Beivide, R., Gregorio, J.A., Prellezo, J.M., Duato, J. and Izu, C.: Adaptive Bubble Router: A Design to Improve Performance in Torus Networks, *Proc. 1999 International Conference on Parallel Processing*, pp.58–67 (1999).
18) Lysne, O. and Skeie, T.: Load Balancing of Irregular System Area Networks Through Multiple Roots, *Proc. 2nd International Conference on Communications in Computing*, pp.165–171 (2001).
19) Ho, W.H. and Pinkston, T.M.: A Design Methodology for Efficient Application-Specific On-Chip Interconnects, *IEEE Trans. Parallel and Distributed Systems*, Vol.17, No.2, pp.174–190 (2006).

(平成 18 年 1 月 27 日受付)
(平成 18 年 5 月 24 日採録)

松谷　宏紀（学生会員）

平成 16 年慶應義塾大学環境情報学部卒業．平成 18 年同大学大学院理工学研究科開放環境科学専攻修士課程修了．現在，同大学院理工学研究科開放環境科学専攻博士課程．平成 18 年度より日本学術振興会特別研究員．チップ内ネットワークの研究に従事．

鯉渕　道紘（正会員）

平成 12 年慶應義塾大学理工学部情報工学科卒業．平成 15 年同大学大学院理工学研究科開放環境科学専攻博士課程修了．博士（工学）．平成 14 年度から平成 16 年度まで日本学術振興会特別研究員．現在，国立情報学研究所助手．マルチプロセッサシステムの結合網に関する研究に従事．

天野　英晴（正会員）

昭和 56 年慶應義塾大学理工学部電気工学科卒業．昭和 61 年同大学大学院理工学研究科電気工学専攻博士課程修了．現在，慶應義塾大学理工学部情報工学科教授．工学博士．計算機アーキテクチャの研究に従事．