社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－ＡＲＣ－168 (19)
2006／6／9

オンチップトーラス網における仮想チャネルフリーなマッピング手法

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上でのネックとなる．しかし，wrap-around チャネルを有するトーラス網においては，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に，マッピングによる解決策を提案する．本手法では，1) 各 wrap-around チャネルを動的に有効／無効化できるトーラスを提案し，2) デッドロックおよび性能低下を引き起こさず，かつ，できるだけ多くの wrap-around チャネルを有効化できるようにタスクをマッピングする．評価では，提案手法を 17 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，17 個中 10 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 52.4%まで抑えることができた．

A Virtual-Channel Free Mapping 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. For tori providing wrap-around channels, however, a virtual-channel mechanism is usually required to avoid deadlocks with dimension-order routing that exploits the regularity of the topology. In this paper, we propose a scheme to remove virtual channels in tori by accomplishing the following steps: 1) providing a mechanism which allows wrap-around channels to be individually disabled in each router, 2) a task mapping strategy that carefully assigns tasks to a tori, so that as many wrap-around channels as possible are exploited without introducing deadlocks or performance degradation. For the evaluations, the proposed strategy was applied to 17 real application traces. Although the proposed strategy does not use virtual channels, it achieves almost the same performance as a virtual-channel router on tori in ten traces. Moreover, the amount of hardware for a router used in the proposed strategy can be decreased to 52.4% 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，タイル同士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)~5)] や 2 次元トーラス[2),6)] 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[7)] が挙げられる．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[8)]．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[7)]．

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[8),9)]．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，仮想チャネル割り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．例えば，文献 8) のパイプライン化例では，ヘッ

† 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みをノードと呼ぶ．

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ため，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．なお，仮想チャネル機構には head-of-line（HOL）ブロッキングを緩和するという効果もあるが，NoC で想定されるような小規模ネットワーク，かつ，単純な固定型ルーティングにおいてはその効果はきわめて小さい．したがって本論文では HOL ブロッキングについては考慮しない．

近年の組込み機器設計では，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マッピング手法を提案する．この手法では，1) 各 wrap-around チャネルを動的に有効／無効にできるトーラスを提案し，2) デッドロックおよび性能低下を引き起こさず，かつ，できるだけ多くの wrap-around チャネルを有効にできるようタスクを配置する．

まず，本論文の 2 章で既存の仮想チャネルフリー技術を紹介する．3 章でリコンフィギャラブルトーラスと呼ばれる wrap-around チャネルを一部無効化できるトーラスを説明する．その上で仮想チャネルフリーを実現するためのマッピングアルゴリズムを 4 章で提案する．5 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 17 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．

2. 既存の仮想チャネルフリー技術

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[10] が並列計算機向けに提案されている．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．あるパケットがサイクルを形成する可能性があるとき，そのパケットをルータのチャネルバッファに一時的に溜め込むことでサイクルが形成されるのを防ぐ．この場合，パケット全体を丸々格納できるだけのバッファがルータの各チャネルに必要となるため，ハードウェアコストが問題となる NoC には不向きである．

別の解決策としては，パケットを中継ノードで一度吸収した後，もとの宛先に向けて再注入することで循環依存を取り除く方法がある[11]．この方法は Myrinet のような仮想チャネルを持たない SAN において性能向上を目的に提案された．ところが，NoC のようなチップ内通信では通信遅延が極めて小さいため，中継ノードにおけるパケットの吸収／再注入の遅延が性能に大きな影響を及ぼす．したがって，NoC における仮想チャネルフリーのためにこの方法を直接利用することはできない．

3. リコンフィギャラブルトーラス

本章ではリコンフィギャラブルトーラスの定義を述べる．k-ary n-cube における各ノードを N_i と表記する（ただし，$i = \{0, ..., k^n - 1\}$）．図 1 は 4×4 トーラスである．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．例えば，図 1 におけるリング R_0^{x+} は単方向サイクル $N_0 - N_1 - N_2 - N_3 - N_0$ を表す．

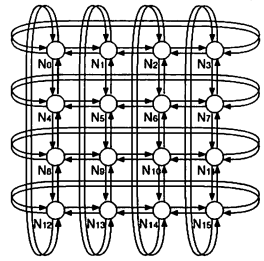

図 1 2-D トーラスの例．

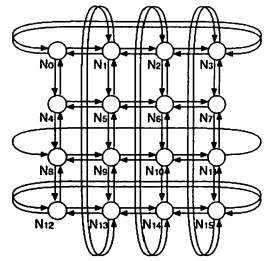

図 2 2-D R トーラスの例．

すべての wrap-around チャネルを無効にすればデッドロックを回避できるが，トポロジ的にはメッシュと等価になるため，ネットワークのスループットはトーラスに比べ大幅に低下する．一方，すべての wrap-around チャネルを有効にできるようタスクをマッピングしても，平均ホップ数が増えてしまってはネットワークの帯域を無駄に消費してしまうため性能向上は見込めない．そこで，本研究では wrap-around チャネルを部分的に無効化できるようにし，この機能を用いて，性能低下を抑えつつ，ほとんどの wrap-around チャネルを活用できるようにタスクを割り当てる．以降，各 wrap-around チャネルを有効／無効にできるトーラスをリコンフィギャラブルトーラス（**R トーラス**）と呼ぶ．なお，無効化した wrap-around チャネルは電気的に切断されるのではなく，一時的に使用不可になるだけである．つまり，無効化した wrap-around チャネルに応じてルータのルーティング関数が変更される．

図 2 に 4×4 の R トーラスの例を示す．この例では，リング R_4^{x+}，R_4^{x-}，R_0^{y+}，R_0^{y-}，R_8^{x-} に対応する各 wrap-around チャネルが無効化されている．R トーラスにおける各ルータは，それぞれの方向の wrap-around チャネルの設定（有効／無効）を記憶するためのレジスタを持つ．そして，同一リング上のルータは，そのリングに対応する wrap-around チャネルに対し，同一の設定を持つ．例えば，ノード N_4，N_5，N_6，N_7 では，$x+$ と $x-$ 方向の wrap-around チャネルが無効化されている．ある一方向の wrap-around チャネルが無効化されると，対応するルータのルーティング関数は，メッシュにおける次元順ルーティングと等価になり，仮想チャネルは必要ない．ルータにおいて，ある方向の wrap-around チャネルを使うか使わないかを記憶するために，各方向ごとに 1-bit のレジスタのみが必要となる．そのため，R トーラスを実現するための面積オーバヘッドは小さいと言える．R トーラス向けルータのハードウェア量については 5.1 節にて示す．この方法によって，動作中に wrap-around チャネルを有効／無効化することができる．

4. マッピング手法

4.1 準　備

近年の組込み機器設計では，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．本マッピングアルゴリズムでは，この解析された通信パターンを用いる．ここで，タスク T_s から T_d へのデータ転送量の合計を $D_{(s,d)}$ と表記する．$D_{(s,d)}$ が 0 の場合，通信パス T_s-T_d は使われておら

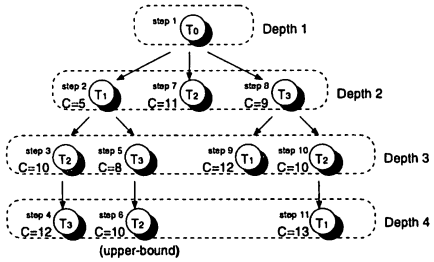

図 3 マッピングのための探索木の例.

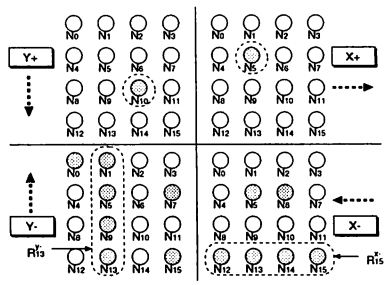

図 4 サイクル検出のためのビットマップ（$x+, x-, y+, y-$）.

ず，この通信がデッドロックを引き起こすことはない．つまり，有効な通信パス（$D_{(s,d)} > 0$）がリング上にサイクルを形成しないようにタスクをノードに割り当てれば，仮想チャネル無しでもデッドロックは起きない．

可能なすべてのマッピングは図 3 のような木構造で表すことができる．この図では 4 個のノードに対し，4 個のタスクを割り当てている．この探索木における各マッピングは，ルートノードから任意のノードへの線形リストで表現される．図 3 のマッピング $M = T_0 - T_1 - T_3 - T_2$ は，タスク T_0，T_1，T_3，T_2 がノード N_0，N_1，N_2，N_3 にそれぞれマッピングされるという意味である．また，マッピング $M' = T_0 - T_2$ は，タスク T_0 と T_2 がノード N_0 と N_1 にそれぞれマッピングされ，タスク T_1 と T_3 はまだどのノードにもマッピングされていないという意味である．このように未配置のタスクを含むマッピングを “不完全マッピング” と呼び，すべてのタスクが配置済みのマッピングを “完全マッピング” と呼ぶ．

4.2 マッピングの探索アルゴリズム

本マッピング手法では，デッドロックフリーを満たし，かつ，性能低下が小さいマッピングをすべての可能なマッピングから全数探索によって得る．それぞれのマッピング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 はタスク数，$D_{(s,d)}$ は タスク T_s から T_d へのデータ転送量の合計，$H_{(s,d)}$ は タスク T_s から T_d へのホップ数とする．不完全マッピングにおいては，すべてのタスクの座標が定まっていないため，ホップ数 H が不明な場合もある．この場合はホップ数 H を 1 と仮定する．このコスト関数を用いて，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探索する．

ここで全数探索の計算コストについて考察すると，可能なすべてのマッピングの組み合わせが膨大な数になることがわかる⋆．そこで，本マッピング手法では，現実的な時間で最適解を得るために分枝限定法を用いて探索空間の枝刈りを行う．

図 3 を用いて，マッピング探索の流れを説明する．図中の step 1 から順に探索を開始し，step 6 で マッピング $T_0 - T_1 - T_3 - T_2$ が得られる．このときのコストは 10 である．一方，step 7 の不完全マッピング $T_0 - T_2$ のコストは 11 である．このとき，マッピング $T_0 - T_2$ 以深のすべてのマッピングにおいてコストは 11 以上となる．マッピング $T_0 - T_2$ 以深で 10 より小さいコストのマッピングを見つけることはできないため $T_0 - T_2$ で枝刈りが起きる．このように枝を刈ることで，無駄な探索を行わずに済み，最適解を見つけるための計算量を大幅に減らすことができる．

4.3 サイクル検出アルゴリズム

ここで，R トーラス T においてマッピング M がサイクルフリーか判定するアルゴリズムを説明する．

(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) （ルーティング）送信元タスク T_s と宛先タスク T_d のすべての組み合わせについて：

- (a) データ転送量 $D_{(s,d)}$ が 0 のとき，または，タスク T_s と T_d の両方が未配置のときは (2) に戻る．
- (b) R トーラス T において，N_s から N_d への経路 $P_{(s,d)}$ を次元順ルーティングを用いて計算する．例えば，$P_{(4,14)} = \{N_4, N_5, N_6, N_{10}, N_{14}\}$ となる．
- (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）R トーラス T 上のすべてのリングについて：

- (a) リング上のすべてのノードがマークされていない場合，そのリングはサイクルフリーである（これは文献 12) で証明されている）．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）R 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．

通信パターンを予め解析できる場合，本手法に従って R トーラス上にタスクをマッピングすれば，仮想チャネルを用いなくとも次元順ルーティングでデッドロックフリーを実現できる．

4.4 動的トラフィックへの対処

NoC の主用途である組込み機器においては，アプリケーションの通信パターンは設計時のシミュレーションにより解析可能である．ところが，動的に発生する制御パケットなど，一部の通信についてはアプリケーション設計時に解析できない場合も少なからず存在する．本手法において，このような動的トラフィックは，少量であれ，サイクルを形成しデッドロックを引き起こす可能性がある．そこで，2 章で紹介したパケットの中継ノードでの吸収／再注入[11] を，一部の動的トラフィックに対して行うことでこの問題を解決する．

この方法を用いると，通常の通信遅延に加え，パケットの吸収

⋆ n ノードのネットワークには $n!$ パターンのマッピングが存在する．探索空間のすべての枝を辿るのに 13 ノードで 5 分，14 ノードで 74 分，15 ノードで 1,110 分を要す（AMD Athlon XP 2.08GHz での実行時）．

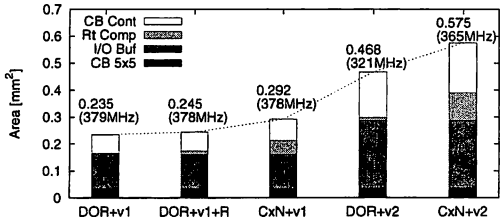

図 5 DOR+v1+R および他のルータの面積.

および再注入のために大きな遅延が発生する．NoC のようなチップ内通信では，そもそも通信遅延が極めて小さいため，パケットの吸収／再注入による影響は無視できない．その上，パケットの吸収／再注入中はその中継ノードはパケットの送受信ができないため，頻繁に吸収／再注入が起こるとスループットが低下する．本研究では次の方法で吸収／再注入が必要なパケットを識別する．

(1) 前節までで説明したマッピング手法は予め解析された静的トラフィックにのみ適用される．これにより，リング上に最低 1 個以上，静的トラフィックが通過しないチャネル (non-static-channel) が存在し，静的トラフィックによってサイクルが形成されないことが保証される．

(2) この non-static-channel を通過するトラフィックはすべて動的トラフィックである．これらの動的パケットを non-static-channel がつながるノードで吸収／再注入する．

上記のステップ (1), (2) を用いることで，静的および動的トラフィックが混在した状況でもデッドロックフリーを満足できる．

すべてのノードは何らかのかたちでパケットを生成，解体するためのバッファを持っており，パケットの吸収／再注入にこれらのバッファを流用できる．このようなノードであれば追加ハードウェア無しに中継ノードになることができる．

5. 評　　価

仮想チャネルの実装コストは文献 9) でも議論されているが，ここでは R トーラス向けに仮想チャネルを持たないルータを実装し，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とも仮想チャネルルータに近い性能が出ることを示す．

5.1 ルータの面積

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** で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，仮想チャネルは持たない．**DOR+v1+R** は R 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（“+R” は Reconfigurable 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．これらのルータのデータ幅（フリット幅）は 32-bit とした．

上記の 5 種類のルータを 0.18μm スタンダードセルライブラリを用いて合成した．合成後の面積および動作速度を図 5 に示す．

R トーラス向けに仮想チャネルを持たないルータ (DOR+v1+R) 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+R 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 52.4% まで抑えることができた．DOR+v1+R 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 4.3%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．

5.2 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．**RTorus+v1** は R トーラス向けに仮想チャネルを持たないルータである．

5.2.1 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．ところが，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[13),14)]) なものが多く，かつ，通信パターンも単純なため，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)[15)] プログラムから得られた通信パターンを用いる．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 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．これら 5 種類の NPB プログラムから得られたトレース 17 種類を評価に用いる．

9, 16, 32, 36, 64 ノードの各プログラムは，それぞれ 3×3, 4×4, 6×6, 6×6, 8×8 の 2 次元ネットワークにマッピングされる．仮想チャネルを持たない R トーラス向けルータ (RTorus+v1) では，デッドロックおよび性能低下を引き起こさないように本マッピング手法によってタスクを割り当てる．一方，従来の 2 次元メッシュ (Mesh+v1) と仮想チャネルを 2 個持つ 2 次元トーラス (Torus+v2) は次元順ルーティングにおいてすでにデッドロックフリーである．これらのマッピングでは，式 1 にもとづいて多量のデータが転送される通信対が近隣に配置されるようにマッピングする．RTorus+v1, Mesh+v1, Torus+v2 の各マッピングを公平に比較するため，マッピングの計算時間をそれぞれ最大 60 分とした．

4.4 節で述べたとおり，一部の通信についてはアプリケーション設計時に解析できない場合もあり，このような環境においても本手法を適用できることを示す必要がある．そこで，全体の 30%のトラフィックを動的トラフィックとしてランダムに生成させた場合も評価した．この結果を **RTorus+v1+d** と表記し，本章の最後で RTorus+v1 の結果と比較する．

5.2.2 シミュレーション結果

上記の 17 種類のアプリケーションを用いて，Mesh+v1, Torus+v2, RTorus+v1 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 6-図 9 に 9, 16, 16, 36, 64 ノードで BT トレースを用いた際のスループット (accepted traffic) とレイテンシのグラフ

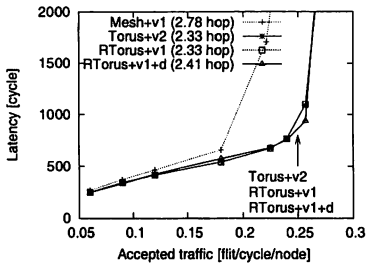

図 6 BT トラフィック（9 ノード）

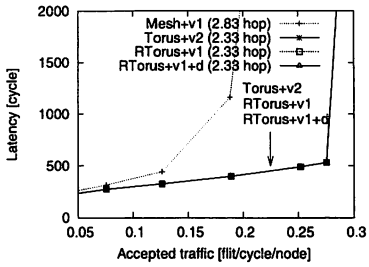

図 7 BT トラフィック（16 ノード）

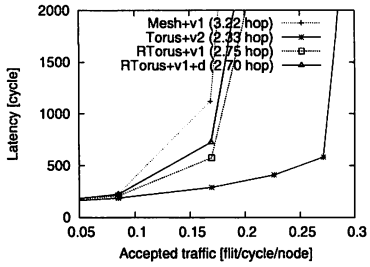

図 8 BT トラフィック（36 ノード）

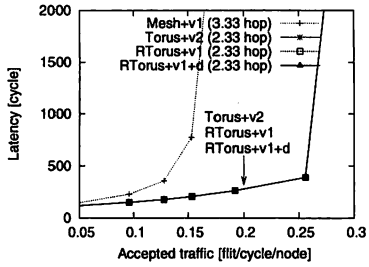

図 9 BT トラフィック（64 ノード）

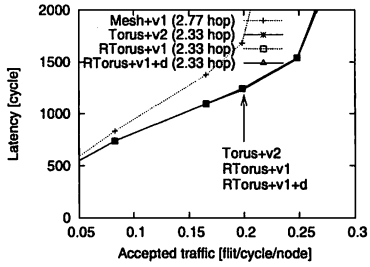

図 10 SP トラフィック（9 ノード）

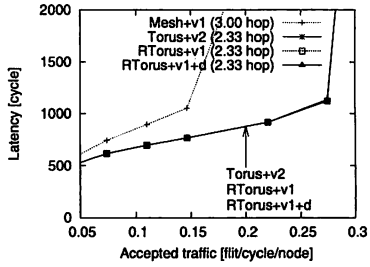

図 11 SP トラフィック（16 ノード）

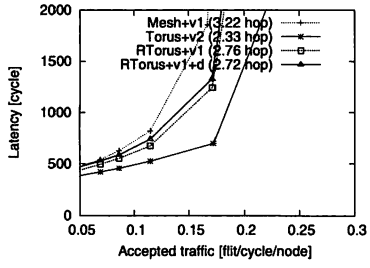

図 12 SP トラフィック（36 ノード）

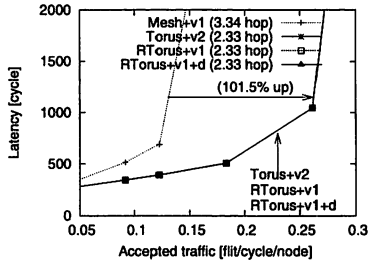

図 13 SP トラフィック（64 ノード）

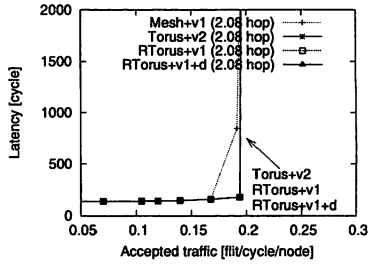

図 14 CG トラフィック（16 ノード）

を示す．それぞれの平均ホップ数は括弧内に示してある．9 ノードでの結果（図 6）では，提案手法によって 3 × 3 R トーラスのすべての wrap-around チャネルを有効化するようにマッピングでき，RTorus+v1 のスループットは Torus+v2 と同等となった．同様に 16，64 ノードの場合においても，RTorus+v1 は Torus+v2 と同様の性能を実現できた．対照的に，36 ノードの場合，RTorus+v1 で 24 本中 12 本の wrap-around チャネルを無効化する必要があり，平均ホップ数も Torus+v2 より増加した．このとき RTorus+v1 の性能は Mesh+v1 より優れるものの Torus+v2 ほどの性能は出ていない．

図 10-図 13 は 9，16，36，64 ノードで SP トラフィックを用いた際の結果である．SP と BT は類似したアルゴリズムをもとにしている[15] 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 20-図 22 に 16，32，64 ノードでの IS トラフィックの結果を示す．IS トレースでは all-to-all 通信が大部分を占めている．64 ノードでは，RTorus+v1 においてすべての wrap-around チャネルを無効化する必要があり，R トーラスは論理的に同サイズのメッシュと等価となってしまった．そのため，RTorus+v1 は Mesh+v1 と同等の性能しか出ていない．一方，16 ノードでは all-to-all 通信が用いられているにも関わらず，サイクルが形成されないようにタスクを割り当てることで，R トーラス上のすべての wrap-around チャネルを有効化できた．

以上の評価により，17 個中 10 個のアプリケーション・トレースにおいて，R トーラス向けルータは従来の仮想チャネルルータと同等の性能を実現できた．

最後に，動的トラフィックが含まれる環境においても本マッピング手法を適用できることを示す．ここでは RTorus+v1 と 30%の動的トラフィックを含む RTorus+v1+d の結果を比べる．BT と SP の結果を見ると，RTorus+v1 と RTorus+v1+d の差異は小さく，動的トラフィックの影響が小さいことがわかる．これは，これらの通信パターンの平均ホップ数が比較的小さく，中継ノードで吸収／再注入されるパケットの数が少なかったためである．一方，IS では，吸収／再注入が頻繁に発生した影響で RTorus+v1+d の性能が大きく低下している．ただし，実際 NoC に適用されるようなストリーム処理では隣接間ノード間通信が多く，IS のような通信パターンはまれであると考えられる．

6. 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に 1) 各 wrap-around チャネルを動的に有効／無効にできる R トーラスを提案し，2) デッドロックおよび性能低下を引き起こさず，かつ，できるだけ多くの wrap-around チャネルを有効にできるようなタスクマッピング手法を示した．

R トーラス向けに仮想チャネルを持たないルータは，2 次元トーラス向けに仮想チャネルを 2 個持つ従来の仮想チャネルルータの 52.4% の面積まで抑えることができた．さらに，17 個中 10

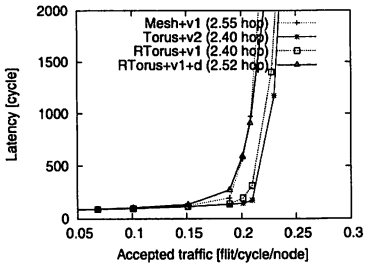

図 15 CG トラフィック (32 ノード)

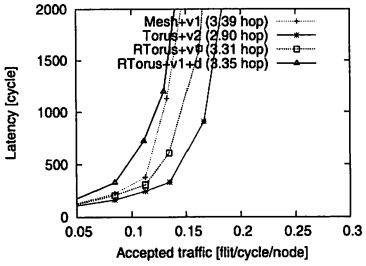

図 16 CG トラフィック (64 ノード)

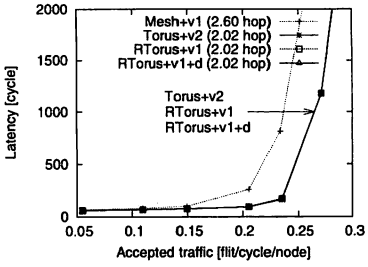

図 17 MG トラフィック (16 ノード)

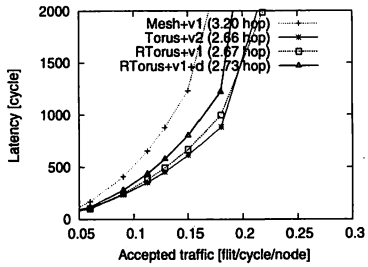

図 18 MG トラフィック (32 ノード)

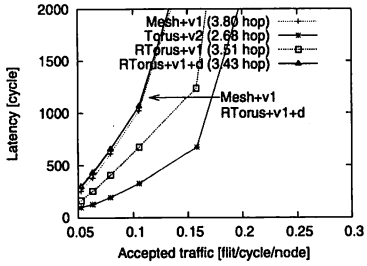

図 19 MG トラフィック (64 ノード)

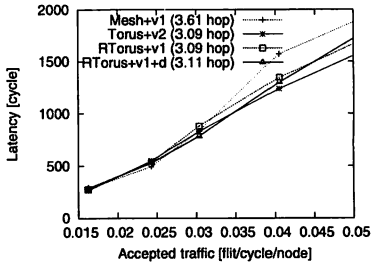

図 20 IS トラフィック (16 ノード)

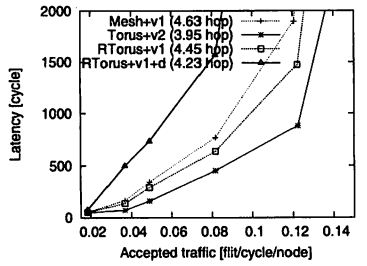

図 21 IS トラフィック (32 ノード)

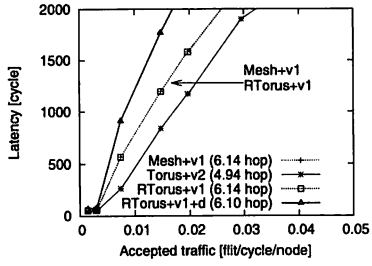

図 22 IS トラフィック (64 ノード)

個のアプリケーションで R トーラス向けルータはトーラス上の従来の仮想チャネルルータ並みの性能を実現できた．NoC の主用途である組込み分野では，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し，タスクを R トーラス上にマッピングすることで，仮想チャネルを用いない低コストルータにおい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わかった．

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, *Proceedings of the 38th Design Automation Conference*, pp. 684–689 (2001).
3) Koibuchi, M., Anjo, K., Yamada, Y., Jouraku, A. and Amano, H.: A Simple Data-Transfer Technique using Local Address for Networks-on-Chips, *IEEE Transactions on Parallel and Distributed Systems* (to be appeared).
4) Hu, J. and Marculescu, R.: Energy- and Performance-Aware Mapping for Regular NoC Architectures, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 4, pp. 551–562 (2005).
5) Taylor, M. B. and 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).
6) Marescaux, T. and et. al.: Interconnection Networks Enable Fine-Grain Dynamic Multi-Tasking on FPGAs, *Proceedings of the Field-Programmable Logic and Applications (FPL)*, pp. 795–805 (2002).
7) Dally, W. J. and Seitz, C. L.: Deadlock-Free Message Routing in Multiprocessor Interconnection Networks, *IEEE Transaction on Computers*, Vol. 36, No. 5, pp. 547–553 (1987).
8) Dally, W. J. and Towles, B.: *Principles and Practices of Interconnection Networks*, Morgan Kaufmann (2004).
9) Chien, A. A.: A Cost and Speed Model for k-ary n-Cube Wormhole Routers, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 2, pp. 150–162 (1998).
10) 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, *Proceedings of the 1999 International Conference on Parallel Processing*, pp. 58–67 (1999).
11) Flich, J., Lopez, P., Malumbres, M. P. and Duato, J.: Boosting the Performance of Myrinet Networks, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 7, pp. 693–709 (2002).
12) 松谷宏紀, 鯉渕道紘, 天野英晴: 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, 先進的計算基盤システムシンポジウム (SACSIS2006) 論文集 (2006). (to be appeared).
13) Liang, J., Laffely, A., Srinivasan, S. and Tessier, R.: An Architecture and Compiler for Scalable On-Chip Communication, *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 12, No. 7, pp. 711–726 (2004).
14) 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).
15) Bailey, D., Harris, T., Saphir, W., Wijngaart, R., A.Woo and M.Yarrow: The NAS Parallel Benchmarks 2.0, *NAS Technical Report, NAS-95-020* (1995).