低遅延オンチップネットワークのための予測ルータの評価

松谷宏紀[†] 鯉渕道紘[††]
天野英晴[†] 吉永努[†††]

近年のメニーコア・アーキテクチャでは，コア間の通信遅延がアプリケーションに与える影響が益々大きくなってきている．コア間の通信遅延を減らすために我々は予測機構を用いた低遅延ルータを提案してきた．予測ルータでは，次のパケット転送で使われるであろう出力チャネルを予測し，パケット到着前にアービトレーションを完了させておく．予測が当たれば，最短 1 サイクルでフリットを転送できるため，ヒット率の高い予測アルゴリズムを選択することが低遅延化の鍵である．

予測ルータでは複数の予測アルゴリズムを装備し，通信パターンに応じて予測アルゴリズムを選択することで，幅広いアプリケーションの低遅延化を狙う．本論文では，ネットワーク環境ごとにどのような予測アルゴリズムが適しているかを示すため，予測ルータを 4 種類のメニーコア・アーキテクチャに適用する．4 種類の case study ごとに予測ルータを 65nm プロセスを用いて実装し，面積，消費エネルギー，予測アルゴリズムごとの予測ヒット率，通信遅延を評価することで，予測ルータによる低遅延化とそのオーバヘッドを明らかにする．

Evalutions of Prediction Router for Low-Latency On-Chip Networks

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

To reduce the communication latency on recent many-core architectures, we have proposed a low-latency router architecture that predicts an output channel being used by the next packet transfer and speculatively completes the switch arbitration. In the prediction routers, incoming packets are transferred without waiting the routing computation and switch arbitration if the prediction hits. Thus, the primary concern for reducing the communication latency is to select the prediction algorithm that offers a high hit rate in a given network.

The prediction routers support multiple prediction algorithms and select one of them in response to the traffic pattern in order to accelerate wider range of applications. To investigate the optimal prediction algorithm for a given network, the prediction router architecture is applied to four many-core architectures. For each case study, the prediction routers are designed with a 65nm CMOS process and evaluated in terms of the area, energy, prediction hit rate, and communication latency. This paper shows their latency reduction and their area- and energy-overhead.

1. はじめに

近年のメニーコア・アーキテクチャでは，コアの数は増加の一途を辿っており，コア間の通信遅延がアプリケーションに与える影響が益々大きくなってきている．コア間の通信にはネットワーク構造（Network-on-Chip，NoC）[1),2)] が広く用いられるため，通信遅延の小さいオンチップルータの開発が急務である．

このような状況を背景に，これまでに様々な低遅延ルータが提案されてきた[3)~10)]．例えば，文献 6) ではチップマルチプロセッサの L2 キャッシュバンクを接続する 8×8 のネットワーク向けにルータの低遅延化を行っており，その結果，SPLASH-2 ベンチマークの実行時間を平均約 20% 削減，FFT プログラムにおいては最大 30% 削減した．また，文献 7) においても，同様のネットワークを低遅延化することによって SPLASH-2 ベンチマークの実行時間を大幅に削減している．

さらに，コア数の増加に伴いルータの低遅延化がアプリケーションに与える影響がさらに大きくなると考えられる．例えば，8×8 mesh の平均ホップ数は 16/3，16×16 mesh では 32/3 である☆．ルータが 1-flit 転送するのに 3 サイクルかかるとすると，各ネットワークにおける 1-flit の平均転送遅延は 16 サイクル，32 サイクルとなる☆☆．L2 キャッ

† 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/総合研究大学院大学
National Institute of Informatics / The Graduate University for Advanced Studies
††† 電気通信大学 大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications

☆ $k\times k$ mesh の平均ホップ数は $2k/3$ で計算できる[4)]．

☆☆ ここでは議論を簡単にするために 1-packet = 1-flit としている．なお，wormhole ルータにおいて，L-flit から成るパケットの通信遅延は式 1 で計算できる．

シュアクセス遅延は 6 サイクル程度であるため[6),7)]，ネットワークが大きくなるにしたがい L2 アクセスに占める通信遅延の影響，つまり，ルータの低遅延化の効果が大きくなる．

我々はこれまでに予測機構を用いた低遅延ルータを提案してきた[11)~13)]．予測ルータでは，次のパケット転送で使われるであろう出力チャネルを予測し，パケット到着前にアービトレーションを完了させておく．予測が当たれば，文献 6)，7) のように 1 サイクルでフリットを転送できるため，その分アプリケーションを高速化できる．したがって，予測ルータにおいてはヒット率が高い予測アルゴリズムを選択することが低遅延化の鍵となる．

これまでに我々が行った理論的な予測ヒット率やパケット転送遅延の解析[11),12)] によって，効果的な予測アルゴリズムは通信パターンなどのネットワーク環境に大きく依存することが分かっている．そこで，本論文では，ルータに複数の予測アルゴリズムを装備させ，通信パターンに合わせて使用する予測アルゴリズムを選択する．ネットワーク環境ごとにどのような予測アルゴリズムが適しているかを示すために，4 種類のメニーコア・アーキテクチャに対し，予測ルータを 65nm プロセスを用いて実装し，予測アルゴリズムごとの予測ヒット率，通信遅延，面積，消費エネルギーについて評価する．予測ルータの定量的な評価は文献 13) でも行われたが，予測ルータの評価が主題ではないため，1 つ目の case study を除き，十分なデータを示すことができなかった．本論文では，4 種類の case study を通して，予測ルータによる低遅延化とそのオーバヘッドについて詳細に評価する．

本論文の構成は次のとおりである．2 章でこれまでに提案された低遅延ルータアーキテクチャをサーベイし，3 章で予測ルータについて説明する．4 章では 4 種類の case study をとおして予測ルータを評価し，5 章で本論文をまとめる．

2. 低遅延オンチップルータのサーベイ

本研究の目的は，ルータがヘッダフリットを転送するのに要すサイクル数を減らすことである．ルータの低遅延化のために様々な手法が提案されてきたが，それらは speculative ルータ，look-ahead ルータ，bypassing ルータの 3 種類に大別できる．まずベースとなるルータを説明したうえで，既存の低遅延化手法を紹介する．

2.1 4 サイクルルータ

図 1(a) に典型的な wormhole ルータ（4 サイクル）のパイプライン構造を示す．この例では，ルータに入力されたフリットがルータから出力されるのに最低 4 サイクルかかる．具体的には，まず，宛先アドレスより出力物理チャネルの計算（Routing computation，RC）と出力仮想チャネルの割り当て（Virtual channel allocation，VA）をそれぞれ 1 サイクルかけて行う．そして，クロスバスイッチのアービトレーション（Switch allocation，SA）を済ませ，フリットをクロスバスイッチ上を通過させる（Switch traversal，ST）．

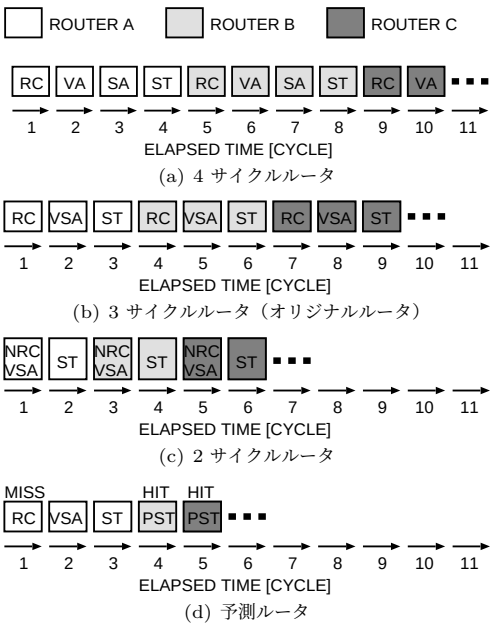


図 1 各種ルータのパイプライン構造

2.2 Speculative ルータ（オリジナルルータ）

speculative ルータでは，複数のパイプラインステージを同時に（並列に）実行することでパイプライン段数を減らす[3)~5)]．最も一般的な speculative ルータは VA と SA を並列して実行するタイプであり，本論文ではこのような 3 サイクルルータを “オリジナルルータ” と呼ぶ．**図 1(b)** にあるように，ここでは VA と SA を統合して VSA ステージとする．この場合，VA および SA 処理に成功すればヘッダが VSA ステージを通過できるが，どちらか片方でも失敗するともう一度 VSA ステージをやり直す必要が生じる．

なお，この方法で RC と VSA を並列化すること（double speculation）もできるが，そのためには RC の結果を待たずに正しい出力チャネルに対して VA および SA を成功させなければならない．当然，やり直しが多発し性能が悪化するため double speculation はあまり採用されていない．

2.3 Look-Ahead ルータ

look-ahead ルータでは，RC と VSA 間の依存関係を断ち切ることで両者を並列に実行できるようにする．このために，各ルータは自ルータの RC ではなく，次ホップのルータの RC を実行（Next routing computation，NRC）する．NRC の結果は次ホップのルータで使われ，自ルータの VSA に影響を与えないため，NRC と VSA を並列に実行できる[4)]．speculation と look-ahead を併用して 2 サイクル化を実現したルータのパイプライン構造を **図 1(c)** に示す．

ただし，NRC/VSA の結果を受けて ST が実行されるため，この方法を用いても NRC/VSA および ST を並列に実行することはできない．NRC/VSA および ST を直列に実行すれば 1 サイクル化を実現できるが，この場合は，ルータの動作周波数に影響が生じる．

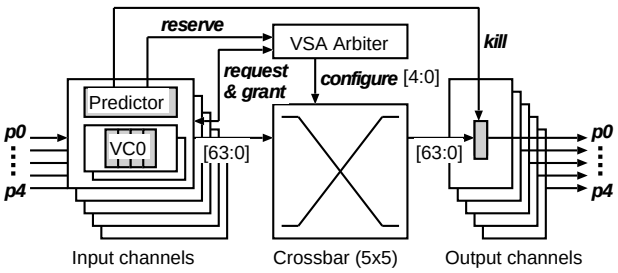


図 2 予測ルータアーキテクチャ（5 ポートの場合）

2.4 Bypassing ルータ

bypassing ルータでは，特定の経路に対して予めクロスバスイッチを割り当てておく，もしくは，専用のデータパスを用意しておく．そうすることで，このような “特定の経路” が使われたときのみ，RC，VA，SA ステージをバイパスして ST のみの 1 サイクル転送を行う．バイパス経路の選び方は以下のとおりである．

- **Frequently Used：** 頻繁に使われる経路に対しクロスバを割り当てるなどして 1 サイクル転送する[6]．
- **Static Straight (SS)：** パケットが同一次元上を直進すると仮定して 1 サイクル転送する[8]．もしくは，文献 7) のように，同一次元上の非隣接ノード間に仮想的なバイパス経路を張り，バイパス経路上の中間ノードで 1 サイクル転送を行う．
- **Custom：** 1 サイクル転送する経路をユーザが設定できるようにする[9),10)]．

ただし，通信パターンに隣接間通信が多いと，文献 7) や文献 8) のバイパス方法では 1 サイクル転送できるチャンスが限られてしまう．また，Custom ではトラフィックの変化に応じて低遅延化する経路を動的に切り替えることはできない．このように，どの方法にもそれぞれ得手不得手がある．

文献 6)∼10) では単一のポリシに基づいてバイパス経路を選択しているが，1 サイクル転送のチャンスを最大化できるようなバイパス経路の選び方はアプリケーションやネットワーク環境（トポロジ，ルーティング）ごとに異なる．つまり，幅広い用途に適用しようとすると複数のバイパス方法をあわせ持ち，状況に応じて切り替える必要がある．

3. 予測ルータ

3.1 予測ルータアーキテクチャ

図 2 に予測ルータのアーキテクチャを示す．予測ルータでは入力チャネルごとに予測器を持ち，パケットが来ていないときに次のパケット転送で使われるであろう出力チャネルを予測，パケット到着前にアービトレーションを完了させておく．この状態を出力チャネルの仮予約と呼び，入力パケットはこの仮予約中の出力チャネルへ投機的に転送される．ただし，他の入力チャネルによってこの仮予約中の出力チャネルが実際に使われると仮予約は解消される．

予測が当たれば RC および VSA を省略できるので ST のみの 1 サイクル転送ができる（**図 1(d)** のルータ B および C を参照）．予測による投機的な ST を，ここでは通常の ST と区別して Predictive switch traversal（PST）と呼ぶ．一方，予測が外れると通常どおり RC，VSA，ST を実行するため 3 サイクル転送となる（図 1(d) のルータ A を参照）．

予測ミス時に，間違った出力チャネルに転送されてしまったフリットを dead flit と呼ぶ．図 1(d) の予測ルータでは 1 サイクル目に予測失敗が検出され，dead flit が転送されてしまった出力チャネルに対し kill 信号が送られる（図 2）．kill 信号を受信した出力チャネルは受信したフリットを無効化する．そのため，dead flit は出力チャネルまで転送されてバッファリングされるが，出力チャネルでマスクされてルータの外に伝搬することはない．消去された dead flit はオリジナルルータと同じタイミングで正しい出力チャネルへ再送されるため，予測が外れた場合はオリジナルルータと同じ挙動をする．

予測ルータにおける dead flit の無効化は，フリットが出力チャネルで一度ラッチされることを利用している．出力チャネルにラッチを持たないルータの実装も可能ではあるが，この場合，ルータ内部の遅延とルータ-ルータ間の配線遅延を分離できない．つまり，クロスバスイッチとリンクの遅延を合わせたものがルータの最大遅延となってしまう．近年のプロセスでは配線遅延が益々深刻化している[14] ため，出力チャネルにラッチを設ける実装がより一般的になると考えられる．

3.2 予測アルゴリズム

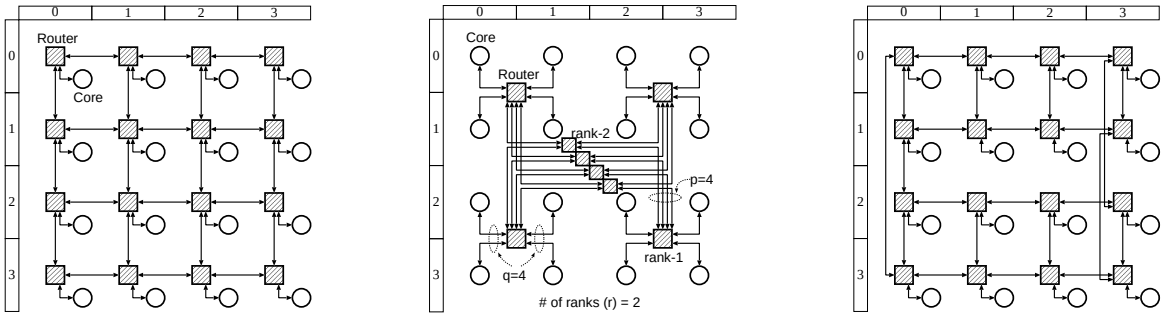
予測ルータでは予測が当たるか外れるかによって転送サイクル数が変化するため，ヒット率が高い予測アルゴリズムを選択することが低遅延化の鍵である．予測ルータでは，複数の予測アルゴリズムを装備し，ネットワーク環境に応じて予測アルゴリズムを切り替えることで幅広い用途において 1 サイクル転送ができるようにする．

予測ルータがサポートする予測アルゴリズムは以下のとおりである．

- **Random:** 次のパケット転送で使われるであろう出力チャネルをランダムに予測する．
- **Static Straight（SS）：** パケットが同一次元上を直進すると仮定して出力チャネルを予測する．文献 8) の方法と等価.
- **Custom:** 予測する出力チャネル（preferred channel と呼ぶ）をユーザが設定できる．文献 9) の方法と等価.
- **Latest Port Matching（LP）：** 前回のパケット転送で使われた出力チャネルが次回も使われると予測する．
- **Finite Context Method（FCM）：** 過去の履歴から次に出現する値を予測する．具体的には，ある値に続く n 個の値の出現パターンごとに出現頻度のカウンタを持ち，その中で最も出現頻度が高い値を予測値とする[15]．$n = 0$ のときは単に最も頻繁に使われた出力チャネルが予測値となる．本論文では $n = 0$ の FCM

表 1 4 種類の case study で用いた NoC の概要

	Case study 1	Case study 2	Case study 3	Case study 4
Topology	2-D mesh	2-D mesh	Fat tree (4,4,r)	Spidergon
# of cores	16-256 cores	64 cores	16-256 cores	16-256 cores
Core distance	0.75mm	1.50mm	0.75mm	0.75mm
Traffic	Uniform	7 NPB programs + 4 synthetic patterns	Uniform	Uniform
Routing	Dimension-order (deterministic)	Dimension-order (deterministic)	up*/down* (adaptive)	Across-first (deterministic)
Switching	Wormhole; no VC	Wormhole; 2 VCs	Wormhole; no VC	Wormhole; 2 VCs
Piplie structure	[RC][SA][ST]	[RC][VSA][ST][LT]	[RC][SA][ST]	[RC][VSA][ST]
Packet size	4-flit (1-flit=64-bit)	4-flit (1-flit=64-bit)	4-flit (1-flit=64-bit)	4-flit (1-flit=64-bit)
Input buffer	4-flit FIFO	4-flit FIFO	4-flit FIFO	4-flit FIFO
Predictor(s)	SS, LP, FCM	SS, Custom, LP, FCM Random, SPM	LRU, LRU+LP	SS, LP, FCM



(a) 2-D mesh (b) Fat tree (4,4,2) (c) Spidergon

図 3 本論文で対象とするネットワークトポロジ（16 コアの例）

を実装したが，$n = 1$ や $n = 2$ の FCM に拡張することも考えている．

- **Sampled Pattern Matching（SPM）：** パターンマッチングに基づく予測アルゴリズムに，系列の長さ制限等の制約条件を付けた方法[16]．過去の通信履歴から繰り返しパターンを検索するため，通信の規則性を抽出しやすい．

Custom における preferred channel の選び方には様々な方法が考えられるが，ここではアプリケーションの通信パターンが既知であると仮定して，通信パターンを事前に解析することで preferred channel を選ぶ．具体的には，各ルータの各入力チャネルにおいて，アプリケーション実行中に最も利用された出力チャネルを調べ，それをその入力チャネルにおける preferred channel として設定する．

4.2 節の case study 2 では，特徴的な通信特性を持つ 11 種類のトラフィックパターンを用いて上記の 6 種類の予測アルゴリズムのヒット率を評価する．

3.3 予測アルゴリズムの選択

理論的な予測ヒット率やパケット転送遅延の解析[11),12)] によって，効果的な予測アルゴリズムはトポロジ，ルーティング，通信パターンなどに大きく依存することが分かっている．

ネットワークトポロジはチップの製造時に固定され，また，ルーティングアルゴリズム関してもトポロジに合わせて選択されるため動作時に変化することは多くない．一方，通信パターンはアプリケーションごとに変化するため，それに合わせて予測アルゴリズムを切り替える必要がある．本論文では，使用するトポロジとルーティングに合わせて予測アルゴリズムを複数セット装備し，実行するアプリケーションの通信特性に合わせてそのうち 1 つを選択するという使い方を想定する．

このために，入力チャネルごとに予測アルゴリズムを切り替えるための信号線を用意した．プロセッサコアからこの信号線を操作することで，実行するアプリケーションに合わせて 1 サイクルで予測アルゴリズムを切り替えることができる．本論文では，アプリケーション毎に予測アルゴリズムを選択するという使い方を想定しているが，予測アルゴリズムを 1 サイクルで切替えることができるため，アプリケーションの実行中に予測アルゴリズムを動的に切替えるという使い方も可能である．ただし，アプリケーション実行中の予測アルゴリズムの動的な切替えは今後の課題とする．

4. 評　価

本章では，ネットワーク環境ごとにどのような予測アルゴリズムが適しているかを示すため，予測ルータを**表 1** に示す 4 種類のメニーコア・アーキテクチャに適用する．各 case study の概要と目的は以下のとおりである．

- **Case study 1：** 2-D mesh トポロジ（**図 3(a)**）にシンプルな予測ルータを適用し，スループット性能，予測ヒット率，無負荷時の通信遅延，ハードウェア量，ルータのクリティカルパス，フリット転送エネルギーについて広範囲に評価する．そのうえで，予測ルータが少

ないオーバヘッドで通信遅延を削減できることを示す．

- **Case study 2：** 様々なアプリケーションのトラフィックパターンを用いて予測アルゴリズムのヒット率を評価し，予測アルゴリズムごとの得手不得手を明らかにする☆．また，このうちいくつかの予測アルゴリズムをオンチップルータ回路に実装し，ハードウェア量と消費エネルギーの評価も行う．これにより，予測ヒット率とハードウェアオーバヘッドのトレードオフを示す．
- **Case study 3 と 4：** 予測ルータが 2-D mesh 以外のトポロジにも有効であることを示すため，fat tree トポロジ（**図 3(b)**）および Spidergon トポロジ（**図 3(c)**）を想定した評価を行う．case study 2 に準じて，予測ヒット率，ハードウェア量，消費エネルギー，さらに無負荷時の通信遅延について評価するが，case study 4 についてはルータの構造が 2-D mesh のものと近いため，ハードウェアオーバヘッドに関する評価は省略する．

4.1 Case Study 1: 2-D Mesh ネットワーク（細粒度）

case study 1 では，予測ルータを網羅的に評価するためにシンプルかつ一般的な構成の NoC を選択する．そこで，細粒度なオペランドネットワークを想定し，NoC で最も一般的なトポロジである 2-D mesh に予測ルータを適用する．

4.1.1 ルータアーキテクチャ

ここでは，以下のオリジナルルータと予測ルータを比較する．

- **オリジナル:** シンプルな wormhole ルータをオリジナルルータとする．ルーティングとして 2-D mesh で最も一般的な次元順ルーティングを用い，各入力チャネルごとに 4-flit 分のバッファを持たせた．仮想チャネルを装備しないため，RC，SA，ST ステージの 3 サイクル転送となる．
- **予測ルータ:** 予測アルゴリズムとして SS，LP，FCM（$n = 0$）を用いる．なお，コアからの入力チャネル（local channel）では SS を利用できないため，SS においても local channel だけは部分的に LP を用いるものとした．予測が当たれば 1 サイクル転送，外れれば 3 サイクル転送となる．

4.1.2 スループット性能

本節では，ルータのパイプライン段数がスループット性能に与える影響を調べる．文献 13) で使用したフリットレベル・ネットワークシミュレータを用いて 1〜4 サイクルルータ☆☆，SS を用いた予測ルータ（Pred(SS)）のスループット性能を測定した．ネットワークサイズは 16×16 mesh とした．通信パターンとしては，ランダムに宛先ノードを選択する uniform トラフィック[4] を用いた．それ以外のシミュレーションパラメータは表 1 のとおりである．

図 4 に評価結果を示す．4.1.3 節で述べるように，16×16 mesh における SS の予測ヒット率は約 80%である．このとき予測ルータは 1.4（$1 \times 0.8 + 3 \times 0.2$）サイクルルータとみなすことができ，評価結果が示すように予測ルータのスループットは 1 サイクルと 2 サイクルルータの間となっている．

4.1.3 予測ヒット率

本節では，ネットワークサイズ（$k \times k$ mesh の k）を変えながら，SS，LP，FCM の予測ヒット率をシミュレーションにより求めた．

図 5 に評価結果を示す．ネットワークサイズが大きくなるにしたがい全体的に予測ヒット率が向上している．SS は小さなネットワークでは効果が小さいが，規模が大きくなるにしたがい効果が大きくなり，$k = 14$ 以上では最もヒット率が高い．$k = 16$ のとき SS のヒット率は 80.5% となった．

4.1.4 無負荷時の通信遅延

本節では，パケットが衝突しないときの通信遅延を求める．

L-flit から成るパケットが h-hop 移動するとき，オリジナルルータの通信遅延 T_0^{orig} は以下の式で計算できる．

$$T_0^{orig} = (T_{rc} + T_{vsa} + T_{st})h + L/\mathrm{BW} \quad (1)$$

ただし，T_{rc}，T_{vsa}，T_{st} はそれぞれのステージのサイクル数であり，今回の場合すべて 1 サイクルとなる．

一方，予測ヒット率を P_{hit} としたときの予測ルータの通信遅延 T_0^{pred} は以下の式で計算できる．

$$T_0^{pred} = (T_{pst})hP_{hit} + (T_{rc} + T_{vsa} + T_{st})h(1 - P_{hit}) + L/\mathrm{BW} \quad (2)$$

図 6 において，Orig はオリジナルルータの通信遅延，Pred(SS) は SS を用いた予測ルータの通信遅延，Ideal は予測が 100% ヒットする場合の通信遅延である．この結果より，16×16 mesh において，SS は無負荷時の通信遅延を 48.2% 削減できることが分かった．

4.1.5 ハードウェア量

本節では，予測ルータのハードウェア量（ゲート数）を求め，オリジナルルータと比較する．

Verilog-HDL を用いて，SS ベースの予測ルータとオリジナルルータを設計し，ローパワー向け 65nm CMOS プロセスを用いて Synopsys Design Compiler で合成した．これらのルータは 500MHz での合成後シミュレーションで動作を確認している．

図 7 に評価結果を示す．予測ルータ（Pred(SS)）の面積オーバヘッドは高々 10.1% である．この増分は，各入力チャネルに実装された予測器，dead flit を消去するための kill 機構，仮予約を実現するためのアービタの改良によるものである．

4.1.6 クリティカルパス遅延

図 8 に，オリジナルルータと予測ルータの各パイプライン

☆ case study 2 ではコア数を 8×8 に固定している．無負荷時の通信遅延は予測ヒット率から容易に計算できるため，case study 2 では予測ヒット率をもとに予測アルゴリズムの得手不得手を議論する．

☆☆ ここでは，図 1(c) の 2 サイクルルータの 2 つのパイプラインステージを直列に 1 サイクルで実行するものを 1 サイクルルータと呼ぶ．

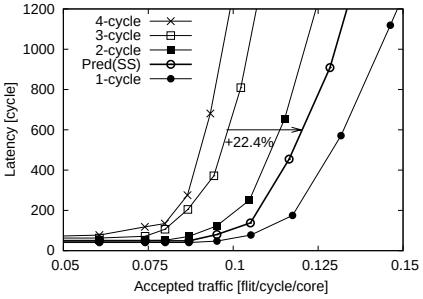


図 4 スループット性能（case study 1）

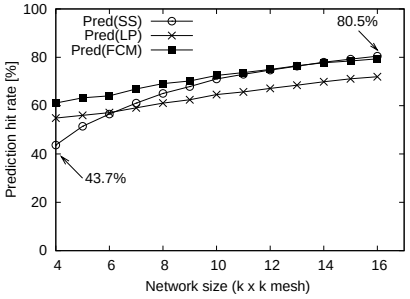


図 5 予測ヒット率（case study 1）

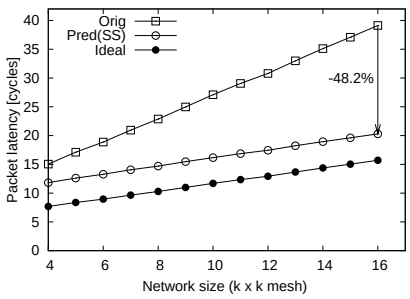


図 6 無負荷時の通信遅延（case study 1）

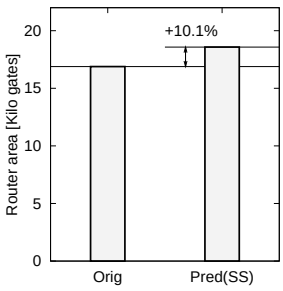


図 7 ルータのハードウェア量（case study 1）

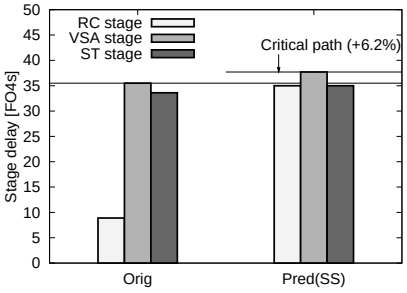


図 8 ルータの遅延（case study 1）

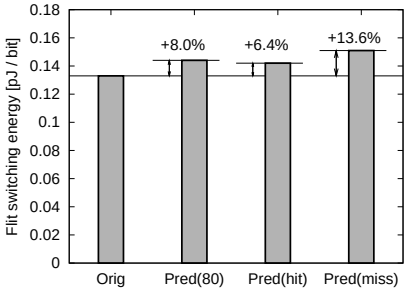


図 9 フリット転送エネルギー E_{sw}（case study 1）

段におけるクリティカルパス遅延（単位は FO4☆）を示す．

予測ルータでは予測がヒットすれば最初のステージ（RC）からでも PST を実行できるため，RC ステージの遅延が ST ステージ並となっている．それでも，オリジナルルータにおいて最も遅延の大きい VSA ステージの遅延は予測機構を持たせても大きくは増加していない．そのため，予測ルータにおける遅延の増加はオリジナルルータと比べて高々 6.2% である．

4.1.7 消費エネルギー

1-flit 分のデータを送信元から宛先まで転送するのに要す平均エネルギー E_{flit} は以下のように表記できる[17]．

$$E_{flit} = wH_{ave}(E_{sw} + E_{link}) \quad (3)$$

ただし，w はフリット幅，H_{ave} は平均ホップ数，E_{sw} はルータが 1-bit を転送するのに要すエネルギー，E_{link} は 1-bit がリンク上を伝搬するのに要すエネルギーとする．

本節では，オリジナルルータと予測ルータを E_{sw} について比較する．両者をローパワー向け 65nm CMOS プロセスを用いて Synopsys Design Compiler で合成，Synopsys Astro で配置配線した．配置配線後ネットリストを 500MHz のクロック，1.20V の供給電圧で配置配線後シミュレーションし，switching activity interchange format （SAIF）を生成した．この SAIF から Synopsys Power Compiler を用いて E_{sw} を求めた．

図 9 において Orig はオリジナルルータの E_{sw} を示す．Pred(hit) および Pred(miss) は，SS ベースの予測ルータで予測がヒットしたとき，ミスしたときの E_{sw} である．予測ルータでは，予測処理が必要となるため，予測の成否に関係なく消費エネルギーが増える．加えて，予測がミスすると dead flit の転送と消去が生じるため☆☆，Pred(miss) の E_{sw} は Pred(hit) よりも大きくなる．それでも，16×16 mesh における SS の予測ヒット率は 80% 前後であるため，予想される E_{sw}（Pred(80)）のオーバヘッドは高々 8.0% である．

4.2 Case Study 2: 2-D Mesh ネットワーク（粗粒度）

case study 2 では，様々なアプリケーションのトラフィックを用いて予測アルゴリズムごとの得手不得手を明らかにする．そこで，チップマルチプロセッサを想定し，8×8 mesh に予測ルータを適用する．ルータの構成は case study 1 に準じるが，様々なアプリケーションの実行を想定し，ルータに仮想チャネルを持たせることで転送能力を向上させる．

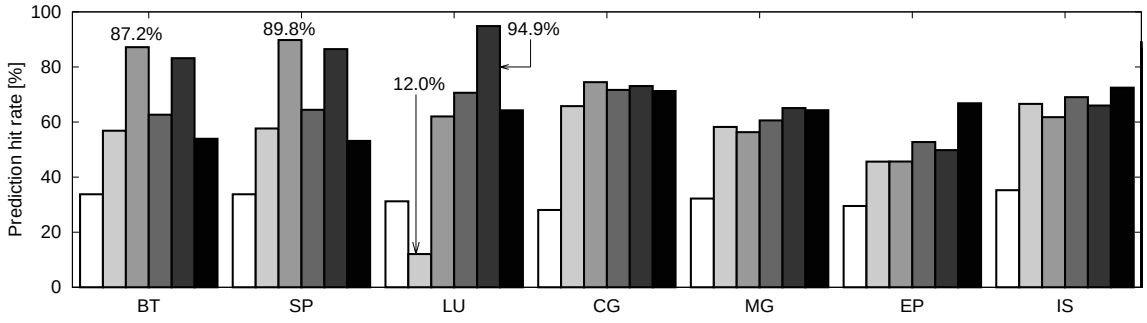
使用するトラフィックとしては，NAS parallel benchmark（NPB）[18] より BT，SP，LU，CG，MG，EP，IS の 7 種類のトラフィック，合成トラフィック（特定の規則に基づいて人工的に生成したトラフィックパターン）として bitcomp，bitrev，transpose，uniform[4] の 4 種類を選んだ．

4.2.1 ルータアーキテクチャ

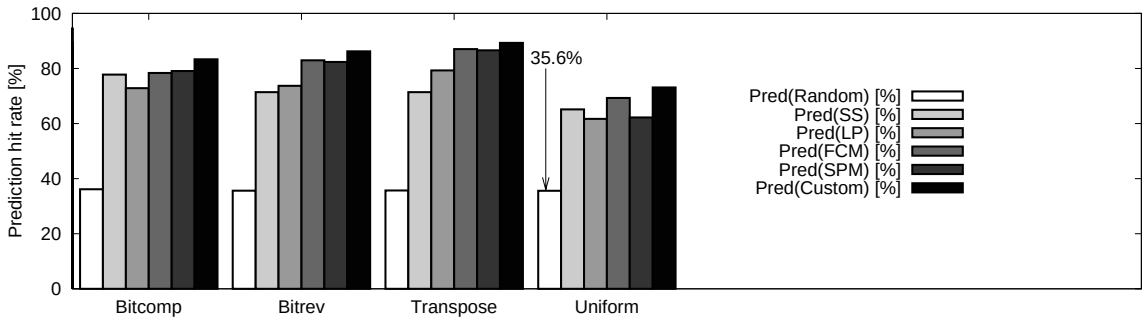
- **オリジナル:** 仮想チャネルを 2 本持った wormhole ルータをベースとする．ルータ間距離が比較的長いため，リンク上をデータが移動（Link traversal，LT）するのに 1 サイクルを割り当てた．結果として RC，VSA，ST，LT ステージから成る 4 サイクル転送となった．

☆ ここでは，あるインバータゲートが同一のインバータゲート 4 個を駆動する際のゲート遅延を Fan-Out 4（FO4）インバータ遅延と呼ぶ．FO4 は半導体プロセスのゲート遅延を表す単位としてしばしば用いられる．

☆☆ 間違った出力チャネルに転送された dead flit は出力チャネルで消去され，オリジナルルータと同じタイミングで正しい出力チャネルへフリットが再送される．そのため予測ミス時には二重のフリット転送が生じてしまう．



(a) NPB トラフィック（64-core）



(b) 合成トラフィック（64-core）

図 10　予測ヒット率（case study 2）

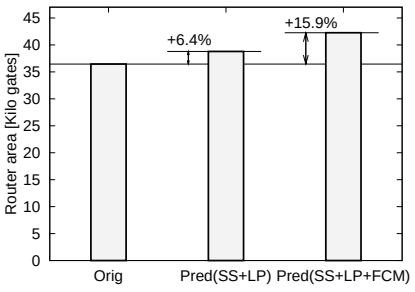


図 11　ルータのハードウェア量（case study 2）

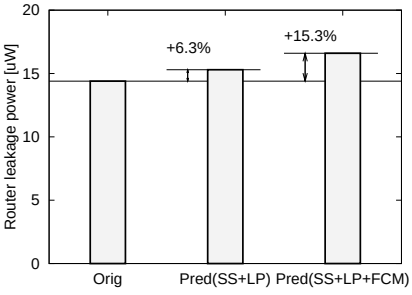


図 12　ルータのリーク電力（case study 2）

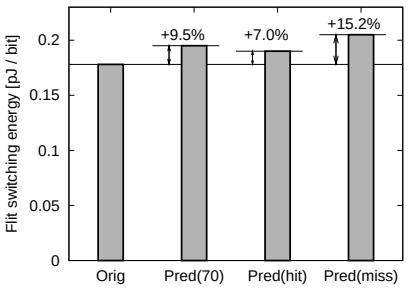


図 13　フリット転送エネルギー E_{sw}（case study 2）

- **予測ルータ:**　予測アルゴリズムとして SS，Custom，LP，FCM，Random，SPM をシミュレーションする．このうち，実装が比較的容易な SS，LP，FCM はルータのハードウェアにも実装する．予測が当たれば 2 サイクル転送，外れれば 4 サイクル転送となる．

4.2.2　予測ヒット率

本節では，8×8 mesh において上記の 11 種類のトラフィックを用いて予測ヒット率をシミュレーションにより求めた．**図 10(a)** に NPB トラフィックの結果，**図 10(b)** に合成トラフィックの結果をそれぞれ示す．合成トラフィックのほうが実アプリケーションよりも通信パターンが単純であるため，全体として予測ヒット率が高くなった．

Random のヒット率はどのトラフィックにおいても 35% 前後☆と非常に低く，実用的とは言えない．

SS は case study 1 で示したとおり，ネットワークサイズが大きくなるにしたがいヒット率が高くなる．16×16 mesh において SS は 80% 以上のヒット率を実現したが，8×8 ではそれほど高くない．また，SS は LU トラフィックでのヒット率が極端に低い（12.0%）．これは，LU には多量の 1-hop 通信が含まれており，直進予測が当たる機会が限られてしまったためである．

LP，FCM，SPM は多くのアプリケーションにおいて SS と同じかそれ以上のヒット率を実現している．LP は繰り返しの短距離通信においてヒット率がとくに高くなる．実際，繰り返しの短距離通信を多く含む BT と SP において LP は最大 89.8% の予測がヒットしている．SPM は予測にパ

☆ 2-D torus で次元順ルーティングを用いる場合，X+ の入力チャネルにおいて転送可能な出力チャネルは X-，Y+，Y-，local channel であるため，25% の確率で予測が成功する．一方，Y+ の入力チャネルに対して可能な出力チャネルは Y- と local channel であるため，50% の確率で予測が成功する．このようにして 5 個の入力チャネルの予測ヒット率の平均を取ると 35% となる．ただし，これは 2-D torus の場合の結果であり，非対称な構造を持つ 2-D mesh ではヒット率が若干変化する．

ターンマッチングを用いるため，通信パターンに規則性がある場合にヒット率が高い．とくに LU では 94.9% の予測がヒットしている．ただし，SPM では過去の履歴を持つ必要があるためハードウェア量のオーバヘッドが大きいと考えられる．

Custom は全体的に高いヒット率を実現できたが，これは通信パターンを事前に解析して最もヒット率が高くなるように preferred channel を選んだためである．通信パターンが動的に変化するなど事前に通信パターンの情報が得られない状況ではこの方法は利用できない．

以上の結果より，通信パターンの事前解析が必要な Custom を除くと，SS，LP，FCM はヒット率が高く，実装も容易であると考えられる．そこで，以降では SS，LP，FCM をハードウェアに実装し，ハードウェア量と消費エネルギーについて評価する．

4.2.3 ハードウェア量

ここでは，2 種類の予測ルータを実装してゲート数を求めた．Pred(SS+LP) は予測アルゴリズムとして SS と LP が実装された予測ルータで，ネットワーク環境に応じて 1 サイクルでアルゴリズムを切り替えることができる．これに FCM を加えた予測ルータが Pred(SS+LP+FCM) である．

図 11 に Orig，Pred(SS+LP)，Pred(SS+LP+FCM) のゲート数を示す．このグラフより，Pred(SS+LP) および Pred(SS+LP+FCM) の面積オーバヘッドはそれぞれ 6.4% と 15.9% である．FCM では各出力チャネルの使用回数をカウントする 4-bit カウンタが必要となるためハードウェア量が多い．その一方で，**図 10** で見たように FCM は幅広いアプリケーションにおいて安定して高いヒット率を実現している．このように，Pred(SS+LP) と Pred(SS+LP+FCM) には面積と予測ヒット率（通信遅延の削減量）のトレードオフが生じている．

4.2.4 リーク電力

図 12 に，温度を 25°C としたときの Orig，Pred(SS+LP)，Pred(SS+LP+FCM) ルータのリーク電力を示す．リーク電力はデバイスエリアにほぼ比例するため，予測ルータのハードウェア量（図 11）に比例してリーク電力も増加している．

4.2.5 消費エネルギー

図 13 に Orig と Pred(SS+LP+FCM) の E_{sw} を示す．図 10 で示したように，MG と EP を除き，ほとんどのアプリケーションの予測ヒット率は 70% を越えている．4.1.7 節で述べたとおり，予測ルータでは予測が成否に応じて E_{sw} の値が異なるが，ここでは 70% の予測が当たると仮定する．その結果，予想される E_{sw} のオーバヘッドは 9.5% となった．

4.3 Case Study 3: Fat Tree ネットワーク

case study 3 では，予測ルータが 2-D mesh 以外のトポロジにも有効であることを示すため fat tree トポロジを用いる．

fat tree には様々な構成があるが，ここでは fat tree を (p,q,r) の 3 つのパラメータで表記する．p は上向きのリンク数，q は下向きのリンク数，r はランク数とする．したがって，図 3(b) は fat tree (4,4,2) と表記される．ここでは SPIN[19] で用いられた $p = 4$ かつ $q = 4$ の fat tree を評価に用いる．

ツリー型トポロジで最も一般的な up*/down* ルーティングと呼ばれる適応型ルーティングを用いる．up*/down* ルーティングでは，パケットは送信元と宛先の共通の祖先（least common ancestor，LCA）までルート方向へ進み（up 方向の転送），LCA から宛先までリーフ方向へ進む（down 方向の転送）．$p > 1$ のとき up 方向の転送には複数の代替経路があるが，down 方向の転送には単一の経路しかない．つまり，up 方向は適応型ルーティングとなるが，down 方向は常に固定型ルーティングである．

4.3.1 ルータアーキテクチャ

- **オリジナル:** ここでは $p = 4$ かつ $q = 4$ の fat tree を仮定する．そのため各ルータは下側にチャネル 4 個，上側にチャネル 4 個を持つ．それ以外は case study 1 のルータと同じである．
- **予測ルータ:** fat tree の下側チャネルでは，SS を改良した LRU という予測アルゴリズムを用いる．LRU を用いる下側チャネルは，パケットが直進すると予測し，上側チャネルのいずれかが使われると予測する．このとき，複数の上側チャネルのうち最近最も使われていない上側チャネルを選び，投機的にパケットを転送する．こうすることで負荷を複数のツリーに分散させることができる．一方，上側チャネルにとっては，正しい出力チャネルは 1 つしかないため，LRU をそのまま利用すると q 回に 1 回の割合でしかヒットしない．

  ここでは fat tree 向けに 2 種類の予測ルータを実装した．Pred(LRU) では下側チャネルは LRU を用い，上側チャネルは予測しない．一方，Pred(LRU+LP) では下側チャネルは LRU，上側チャネルは LP を用いる．後者のほうが予測ミスの回数が多くなるが，1 サイクル転送できるチャンスが増えるため通信遅延は減る．

4.3.2 予測ヒット率・無負荷時の通信遅延

図 14 および **図 15** に，オリジナルルータ（Orig）と 2 種類の予測ルータの予測ヒット率，無負荷時の通信遅延を示す．

図 14 より，Pred(LRU+LP) のほうが Pred(LRU) よりヒット率が高い．前者のヒット率は 256-core の fat tree において 55.8% であり，このときの通信遅延は Orig と比べて 30.7% 小さい．

4.3.3 ハードウェア量

図 16 に Orig と Pred(LRU+LP) のゲート数を示す．Pred(LRU+LP) の面積オーバヘッドは高々 7.8% となった．

4.3.4 消費エネルギー

4.3.2 節の結果をもとに Pred(LRU+LP) の予測ヒット率を 55% と仮定する．**図 17** に示すように，このときの Pred(LRU+LP) の E_{sw} に関するオーバヘッドは 9.0% と

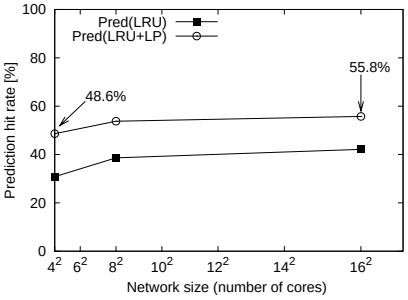


図 14 予測ヒット率（case study 3）

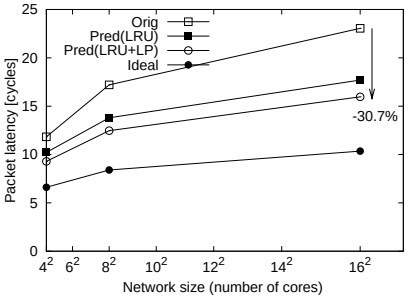


図 15 無負荷時の通信遅延（case study 3）

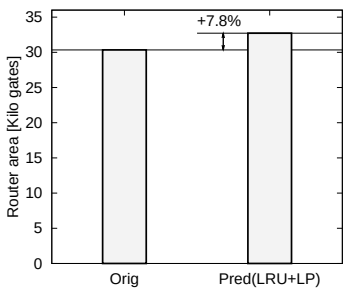


図 16 ルータのハードウェア量（case study 3）

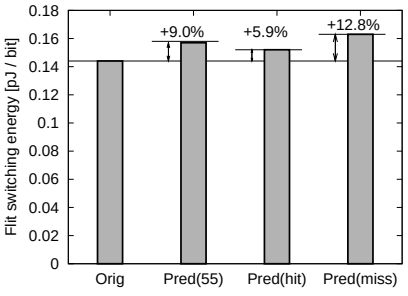


図 17 フリット転送エネルギー E_{sw}（case study 3）

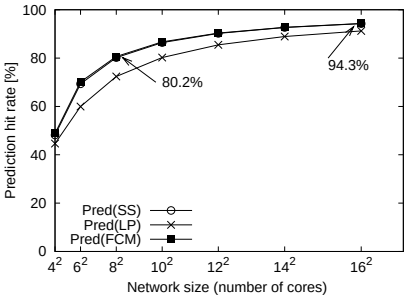


図 18 予測ヒット率（case study 4）

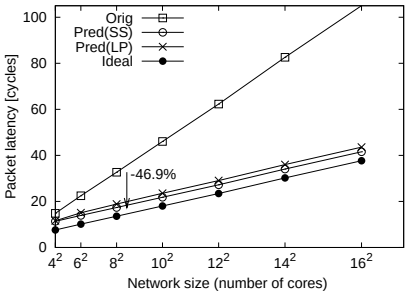


図 19 無負荷時の通信遅延（case study 4）

なった．

4.4 Case Study 4: Spidergon ネットワーク

case study 4 では予測ルータが Spidergon[20),21)] においても有効であることを示す．Spidergon は STMicroelectronics の NoC 向けに提案されたトポロジであり[20)]，リングトポロジの各ノードに対し，対角線上のノードと直接通信するためのリンク（across link）を付加した構成をとる．2-D mesh よりも node degree が低く，コスト効率が良いトポロジとして近年注目されている．Spidergon はリングを拡張したトポロジであるにも関わらず，図 3(c) のような mesh-like なレイアウト方法も考案されている．

Spidergon では across-first ルーティング[21)] と呼ばれる最短ルーティングを用いる．across-first ルーティングでは，送信元ノードでのみ across link を利用でき，それ以降は宛先に到達するまで clockwise 方向，もしくは，anticlockwise 方向のどちらか一方のチャネルを使い続ける．

4.4.1 ルータアーキテクチャ

- **オリジナル:** 各ルータは across 方向，clockwise 方向，anticlockwise 方向，コアとの接続のために 4 個のチャネルを持つ．デッドロックフリーのため仮想チャネルを 2 本使用する．
- **予測ルータ:** 予測アルゴリズムとして SS，LP，FCM を用いる．なお，コアからの入力チャネル（local channel），および，across link からの入力チャネルでは SS がヒットしない．そこで，SS においてもこれらのチャネルに関しては部分的に LP を用いるものとした．

4.4.2 予測ヒット率

図 18 に 3 種類の予測ルータの予測ヒット率を示す．SS および FCM のほうが LP よりヒット率が高い．Spidergon では，clockwise 方向から入力されたパケットは anticlockwise 方向へ出力されるというように直進するパスの利用頻度が高い．そのため，直進予測の SS と利用頻度を基にした FCM でほぼ同じ予測ヒット率を示した．

Spidergon は 2-D mesh と比べて node degree が小さいため，2-D mesh よりも経路の多様性が低い．その分，予測が当たる可能性も高くなっている．実際，64-core の Spidergon において Pred(SS) の予測ヒット率は 80% を越えており，256-core では 94% 以上ヒットしている．

4.4.3 無負荷時の通信遅延

4.4.2 節の結果より，無負荷時の通信遅延を見積もった．**図 19** に示すように，64-core の Spidergon において Pred(SS) の無負荷時の通信遅延は Orig よりも 46.9% 小さい．このように Spidergon においても予測機構によって通信遅延が大幅に削減できることを確認した．

5. ま と め

予測ルータでは，ヒット率の高い予測アルゴリズムを用いることが低遅延化の鍵である．ネットワーク環境ごとにどのような予測アルゴリズムが適しているかを示すため，予測ルータを 4 種類のメニーコア・アーキテクチャに適用した．

- **Case study 1：** オペランドネットワークを想定した 2-D mesh にシンプルな予測ルータを適用し，65nm CMOS プロセスを用いてハードウェア量，消費エネルギー，通信遅延の削減量について網羅的に評価した．その結果，ゲート数および消費エネルギーはそれぞれ 10.1% および 8.0% 増加したが，通信遅延は 16×16 mesh において最大 48.2% 削減できた．
- **Case study 2：** チップマルチプロセッサ向けの仮

想チャネルルータを想定し，11 種類のトラフィックパターンを用いて予測アルゴリズム SS，Custom，LP，FCM，Random，SPM のヒット率を評価した．
SS はネットワークサイズが大きくなるにしたがいヒット率が高くなるが，1-hop 通信ではほとんどヒットしなかった．Random のヒット率は 35% 前後と低く，実用的ではなかった．LP は繰り返しの短距離通信において高いヒット率を達成した．SPM は通信パターンの規則性を利用でき，最大 94.9% 予測がヒットしたが，通信履歴の管理が必要となる．その点，SS，LP，Custom は比較的シンプルな回路で高いヒット率を実現できるが，Custom は事前に通信パターンを解析し，preferred channel を選択しておかなければならない．

- **Case study 3：** 予測ルータが 2-D mesh 以外のトポロジにも有効であることを示すため，fat tree ネットワークに予測ルータを適用した．fat tree では LRU+LP のほうが LRU よりヒット率が高く，LRU+LP は通信遅延を最大 30.7% 削減できた．
- **Case study 4：** 低コストな NoC 用のトポロジとして注目されている Spidergon に予測ルータを適用した．Spidergon では SS と FCM のヒット率がほぼ同じとなり，これらは LP よりヒット率が高い．64-core の Spidergon において SS は最大 46.9% の通信遅延を削減できた．

以上より，本論文では 1) 予測ルータは様々なネットワーク環境（トポロジ，ルーティング）に適用できること．2) 予測ルータは現実的なオーバヘッドで通信遅延を大幅に削減できること．3) これまでの単一ポリシに基づいた低遅延ルータ[6)~10)] ではなく，予測ルータのように複数の予測アルゴリズムを装備しネットワーク環境に応じて適切な予測アルゴリズムを選択することが通信遅延削減のために重要であること確認した．

謝 辞 本研究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，株式会社半導体理工学研究センター，(株) イー・シャトル，富士通株式会社の協力で行われた．

参 考 文 献

1) Dally, W. J. and Towles, B.: Route Packets, Not Wires: On-Chip Interconnection Networks, *Proceedings of the Design Automation Conference (DAC'01)*, pp. 684–689 (2001).
2) Benini, L. and Micheli, G. D.: *Networks on Chips: Technology And Tools*, Morgan Kaufmann (2006).
3) Peh, L.-S. and Dally, W. J.: A Delay Model and Speculative Architecture for Pipelined Routers, *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA'01)*, pp. 255–266 (2001).
4) Dally, W. J. and Towles, B.: *Principles and Practices of Interconnection Networks*, Morgan Kaufmann (2004).
5) Kim, J., Nicopoulos, C., Park, D., Narayanan, V., Yousif, M. S. and Das, C. R.: A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks, *Proceedings of the International Symposium on Computer Architecture (ISCA'06)*, pp. 14–15 (2006).
6) Park, D., Das, R., Nicopoulos, C., Kim, J., Vijaykrishnan, N., Iyer, R. and Das, C.: Design of a Dynamic Priority-Based Fast Path Architecture for On-Chip Interconnects, *Proceedings of the IEEE Symposium on High-Performance Interconnects (HOTI'07)*, pp. 15–20 (2007).
7) Kumar, A., Peh, L.-S., Kundu, P. and Jha, N. K.: Express Virtual Channels: Towards the Ideal Interconnection Fabric, *Proceedings of the International Symposium on Computer Architecture (ISCA'07)*, pp. 150–161 (2007).
8) Izu, C., Beivide, R. and Jesshope, C.: Mad-Postman: A Look-Ahead Message Propagation Method for Static Bidimensional Meshes, *Proceedings of the Euromicro Workshop on Parallel and Distributed Processing (PDP'94)*, pp. 117–124 (1994).
9) Michelogiannakis, G., Pnevmatikatos, D. N. and Katevenis, M.: Approaching Ideal NoC Latency with Pre-Configured Routes, *Proceedings of the International Symposium on Networks-on-Chip (NOCS'07)*, pp. 153–162 (2007).
10) Koibuchi, M., Matsutani, H., Amano, H. and Pinkston, T. M.: A Lightweight Fault-tolerant Mechanism for Network-on-Chip, *Proceedings of the International Symposium on Networks-on-Chip (NOCS'08)* , pp. 13–22 (2008).
11) 吉永 努, 村上 弘和, 鯉渕 道紘: 2-D トーラスネットワークにおける動的通信予測による低遅延化, 情報処理学会論文誌コンピューティングシステム, Vol. 1, No. 1, pp. 28–39 (2008).
12) 鯉渕 道紘, 吉永 努, 村上 弘和, 松谷 宏紀, 天野 英晴: 予測機構を持つルータを用いた低遅延チップ内ネットワークに関する研究, 情報処理学会論文誌コンピューティングシステム, Vol. 1, No. 2, pp. 59–69 (2008).
13) Matsutani, H., Koibuchi, M., Amano, H. and Yoshinaga, T.: Prediction Router: Yet Another Low Latency On-Chip Router Architecture, *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA'09)* , pp. 367–378 (2009).
14) Ho, R., Mai, K. W. and Horowitz, M. A.: The Future of Wires, *Proceedings of the IEEE*, Vol. 89, No. 4, pp. 490–504 (2001).
15) Sazeides, Y. and Smith, J. E.: The Predictability of Data Values, *Proceedings of the International Symposium on Microarchitecture (MICRO'97)*, pp. 248–258 (1997).
16) Jacquet, P., Szpankowski, W. and Apostol, I.: A Universal Predictor Based on Pattern Matching, *IEEE Transactions on Information Theory*, Vol.48, No.6, pp.1462–1472 (2002).
17) Wang, H., Peh, L.-S. and Malik, S.: A Technology-Aware and Energy-Oriented Topology Exploration for On-Chip Networks, *Proceedings of the Design, Automation and Test in Europe Conference (DATE'05)*, pp. 1238–1243 (2005).
18) Bailey, D., Harris, T., Saphir, W., van der Wijngaart, R., Woo, A. and Yarrow, M.: The NAS Parallel Benchmarks 2.0, *NAS Technical Reports NAS-95-020* (1995).
19) Andriahantenaina, A. and Greiner, A.: Micro-network for SoC: Implementation of a 32-port SPIN network, *Proceedings of the Design Automation and Test in Europe Conference (DATE'03)*, pp. 1128–1129 (2003).
20) Coppola, M., Locatelli, R., Maruccia, G., Pieralisi, L. and Scandurra, A.: Spidergon: A Novel On-Chip Communication Network, *Proceedings of the International Symposium on System-on-Chip (ISSOC'04)*, p. 15 (2004).
21) Bononi, L. and Concer, N.: Simulation and Analysis of Network on Chip Architectures:. Ring, Spidergon and 2D Mesh, *Proceedings of the Design, Automation, and Test in Europe Conference (DATE'06)*, pp. 154–159 (2006).