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

2009-ARC-181／2009-EMB-11 (1)
2009/1/13

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†††

† 慶應義塾大学大学院 理工学研究科 〒223-8522 神奈川県横浜市港北区日吉 3-14-1
†† 国立情報学研究所 〒101-8430 東京都千代田区一ツ橋 2-1-2
††† 電気通信大学 大学院情報システム学研究科 〒182-8585 東京都調布市調布ヶ丘 1-5-1
E-mail: †{matutani,hunga}@am.ics.keio.ac.jp, ††koibuchi@nii.ac.jp, †††yosinaga@is.uec.ac.jp

あらまし 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．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，トポロジ，ルーティング，通信パターンに応じて，予測アルゴリズムを積極的に切替える．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．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，それぞれのオーバヘッドや予測アルゴリズムの得手不得手を明らかにした．

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

Hiroki MATSUTANI†, Michihiro KOIBUCHI††, Hideharu AMANO†, and Tsutomu YOSHINAGA†††

† Graduate School of Science and Technology, Keio University 3-14-1, Hiyoshi, Yokohama, JAPAN 223-8522
†† National Institute of Informatics 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, JAPAN 101-8430
††† Graduate School of Information Systems, The University of Electro-Communications 1-5-1, Chofugaoka, Chofu-shi, Tokyo, JAPAN 182-8585
E-mail: †{matutani,hunga}@am.ics.keio.ac.jp, ††koibuchi@nii.ac.jp, †††yosinaga@is.uec.ac.jp

Abstract 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 for a given network environment. In this paper, the prediction routers support multiple prediction algorithms and select one of them in response to the network topology, routing algorithm, and traffic pattern. To investigate optimal prediction algorithm for a given network, the prediction router architecture is applied to four many-core architectures. For each case study, the router is designed with a 65nm CMOS process and evaluated in terms of the area, energy, prediction hit rate, and latency. This paper shows their area- and energy-overhead, and the pros and cons of each prediction algorithm.

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] 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，我々は予測機構を用いた低遅延ルータを提案してきた [2] [3] [4]．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．

これまでの我々の研究によって，効果的な予測アルゴリズムはネットワーク環境ごとに異なることが分っている [2] [3] [4]．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，トポロジ，ルーティング，通信パターンに応じて，予測アルゴリズムを積極的に切替えることを考える．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成功率，通信遅延，面積，消費エネルギーについて評価した．

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

本論文の目的は，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, look-ahead, bypassing 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 Original ルータ (4-stage)

図 1 の上部に典型的な wormhole ルータ（Original）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ポートの計算（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調停（Switch allocation, SA）を行い，フリットがスイッチ上を通過する（Switch traversal, ST）．

2.2 Speculative ルータ (3-stage)

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 [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る．この場合，VA および SA 処理に成功すればヘッダが VA/SA（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 VSA ステージをやり直す必要が生じる．

この方法で RC と VSA を並列化すること（double speculation）もできるが，やり直しが多発するため性能が悪化する．

2.3 Look-Ahead ルータ (2-stage)

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 [5].

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両者を並列に実行することはできない．

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，ST のみの 1-cycle 転送を行う．バイパス経路の選び方は以下のとおり．

- **Frequently Used:** 最も頻繁に使われる経路を 1-cycle 転送する [6] [7].
- **Static Straight (SS):** パケットが同一次元上を直進すると仮定して 1-cycle 転送する [8].
- **Custom:** 1-cycle 転送する経路を設定可能 [9] [10].

ただし，通信パターンに隣接間通信が多いと，文献 [6] や文献 [8] の方法では 1-cycle 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．

文献 [6] [7] [8] [9] [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-cycle 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ごとに異なる．

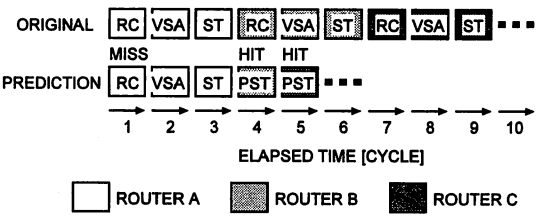


図 1 通常のルータ（上）と予測ルータ（下）のパイプライン構造．

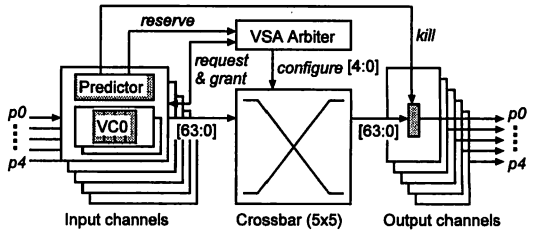


図 2 予測ルータアーキテクチャ．

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．

予測が当たれば RC, VSA を省略できるので ST（通常の ST と区別して，Predictive switch traversal もしくは PST と呼ぶ）のみの 1-cycle 転送ができる（図 1 下部のルータ B および C）．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-cycle 転送となる（図 1 下部のルータ A）．

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 の予測ルータでは 1-cycle 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）．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．

なお，予測ルータの詳細については文献 [4] を参照されたい．

3.2 予測アルゴリズム

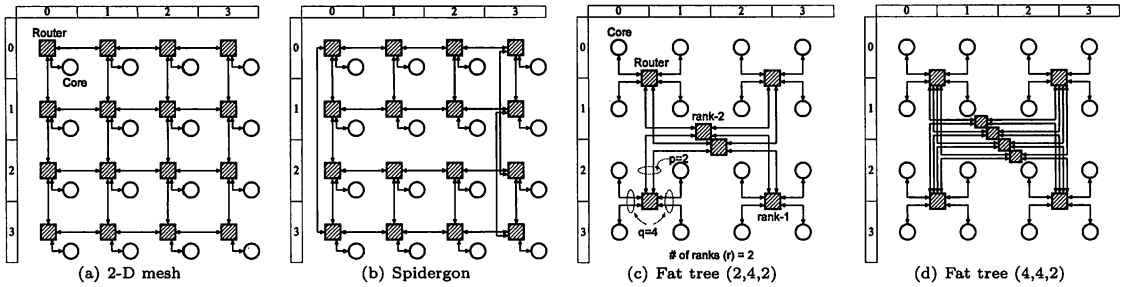
予測ルータでは予測が当たるか外れるかによって転送サイクル数が変化するため，ヒット率が高い予測アルゴリズムを選択することが低遅延化の鍵である．2.4 節で述べたように，適切な予測アルゴリズムはネットワーク環境（トポロジ，ルーティング，通信パターン）ごとに異なる．予測ルータでは，複数の予測アルゴリズムを装備し，ネットワーク環境に応じて予測アルゴリズムを切替えることで幅広い用途において 1-cycle 転送ができるようにする．

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．

- **Random:** 次のパケット転送で使われる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] と等価．
- **Custom:** 予測する出力チャネルを設定可能 [9].
- **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．

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 (4,4,4)	Spidergon
# of cores	16×16 cores	8×8 cores	256 cores	8×8
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][SA][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, LP, FCM	LRU(SS) + LP	SS, LP, FCM



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

図 3　本論文で対象とするネットワークトポロジ.

- **Finite Context Method (FCM):**　直前 n 個のコンテキストの後に最も頻繁に使われた出力チャネルが次のパケット転送で使われると予測する [11]. $n = 0$ 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.
- **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 [12]. 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.

4. 評　価

3.2 節では 6 種類の予測アルゴリズムを示したが，ヒット率を最大化できる予測アルゴリズムはネットワーク環境（トポロジ，ルーティング，通信パターン）ごとに異なる.

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 4 種類のメニーコア・アーキテクチャ（表 1 を参照）に適用して評価する.

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

case study 1 では，細粒度なオペランドネットワークを想定し，16 × 16 mesh ネットワーク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 ルータをベースラインとする. ルーティングとして次元順ルーティングを用い，各入力チャネルごとに 4-flit 分のバッファを持つ. 仮想チャネルを装備しないため，RC，SA，ST の 3-cycle 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，例外的に SS においても LP を用いるものとした. 予測が当たれば 1-cycle 転送，外れれば 3-cycle 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 [4] 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 cycle 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. 他のシミュレーションパラメータは表 1 に示したとおりである.

図 4(a) に評価結果を示す. 16 × 16 mesh における SS の予測ヒット率は 80% 前後である. このとき予測ルータは 1.4 cycle ルータみなすことができ，評価結果が示すように予測ルータのスループットは 1-cycle ルータと 2-cycle ルータの間となる.

4.1.3 予測ヒット率

2-D mesh のサイズ（k-ary 2-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 4(b) に評価結果を示す. ネットワークサイズが大きくなるに従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.

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.

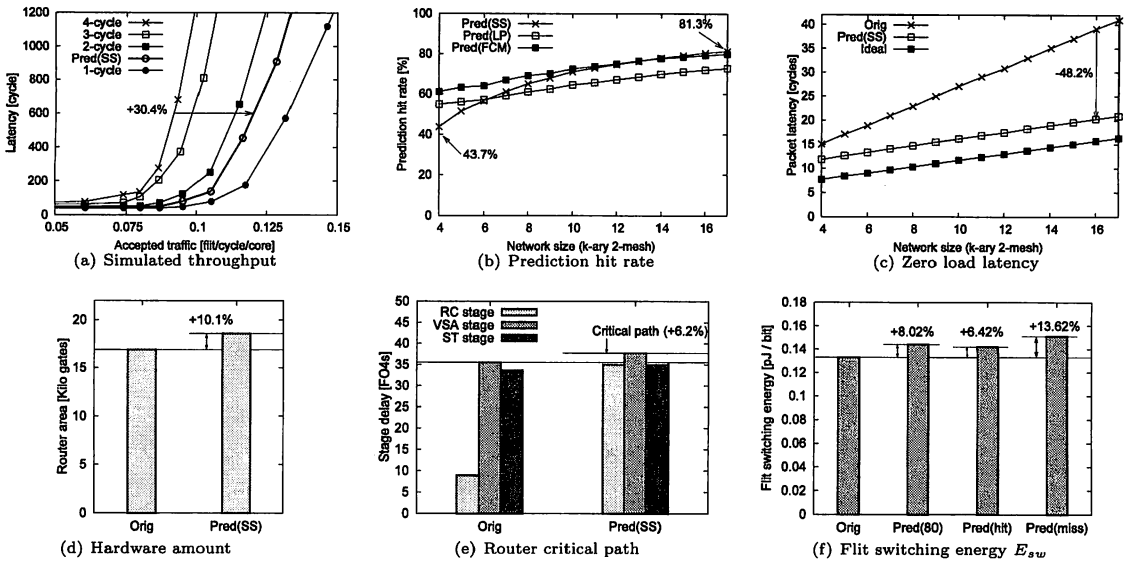
L-flit から成るパケットが h-hop 移動するとき，オリジナルルータの通信遅延 T_0^{orig} は以下の式で計算できる.

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

ただし，T_{rc}，T_{vsa}，T_{st}，T_{lt} はそれぞれのステージのサイクル数であり，T_{lt}（Link traversal）は 0，それ以外は 1 とした.

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.

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



(a) Simulated throughput (b) Prediction hit rate (c) Zero load latency

(d) Hardware amount (e) Router critical path (f) Flit switching energy E_{sw}

図 4 case study 1 の評価結果.

図 4(c) 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 は通信遅延を 48.2% 削減できた．

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，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 4(d) 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，アービタの改良 [4] によるものである．

4.1.6 クリティカルパス遅延

図 4(e) に，オリジナルルータと予測ルータの各パイプライン段におけるクリティカルパス遅延 [FO4] を示す．予測機構を持たせたことによる遅延の増加は高々 6.2% である．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 の遅延が ST ステージ並となっている．

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．

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

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 65nm CMOS プロセスを用いて合成，配置配線した．配線配線後ネットリストを 500MHz, 1.2V で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 4(f) 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．予測がミスすると二重のフリット転送が生じるため(注1)，Pred(miss) の E_{sw} は Pred(hit) よりも大きい．以上より，予測成功率が 80% のとき，予想される E_{sw} 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，チップマルチプロセッサを想定した 8 × 8 mesh に仮想チャネルを持った予測ルータを適用する．ここでは SS, LP, FCM の各予測アルゴリズムに対し，様々なトラフィックを与え，各アルゴリズムの得手不得手を明らかにする．

使用するトラフィックとして，NAS parallel benchmark (NPB) [13] より BT, SP, LU, CG, MG, EP, IS の 7 種類のトラフィックと，合成系として bitcomp, bitrev, transpose, uniform [5] の 4 種類のトラフィックを選んだ．

4.2.1 ルータアーキテクチャ

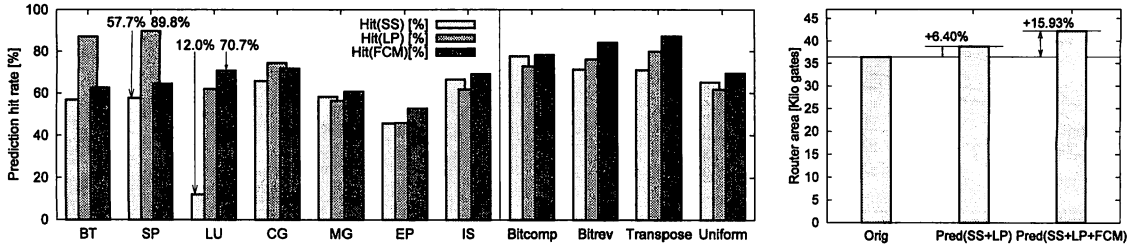
- **オリジナル:** 仮想チャネルを 2 本持った wormhole ルータをベースとする．ルータ間距離が比較的長いため，リンク上をデータが移動する（Link traversal）のに 1-cycle を割り当てた．結果として RC, VSA, ST, LT の 4-cycle 転送となった．
- **予測ルータ:** 予測アルゴリズムとして SS, LP, FCM を用いる．予測が当たれば 2-cycle 転送，外れれば 4-cycle 転送．

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 5(a) に評価結果を示す．合成系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．

(注1)：間違った出力チャネルに転送された dead flit は消去され，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転送されるため．



(a) Prediction hit rate (b) Hardware amount

図 5 case study 2 の評価結果.

SS は case study 1 (16 × 16 mesh) のように大きなネットワークで 80% 以上のヒット率を実現したが, 8 × 8 ではヒット率はそれほど高くない. とりわけ, 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るためである.

LP と FCM は多くのアプリケーションにおいて SS よりヒット率が高い. とりわけ, SP と BT トラフィックにおいて LP は最大 89.8% の予測がヒットしている. SP や BT には繰り返しの短距離通信が多量に含まれるため, LP がとくに良く効いた.

このように予測アルゴリズムには得手不得手があり, ネットワーク環境に応じて適切なアルゴリズムを選択する必要がある.

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-cycle 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 5(b) 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 5(a) 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. そのため, 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) のトレードオフがある.

4.2.4 消費エネルギー

本節では Orig と Pred(SS+LP+FCM) の E_{sw} を比較する. 図 5(a) 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. 予測ルータの場合, 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と E_{sw} のオーバヘッドは高々9.5% である.

4.3 Case Study 3: Fat Tree ネットワーク

case study 3 では, 256 個の ALU を fat tree トポロジで接続する細粒度オペランドネットワークを想定する.

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(c) は fat tree (2,4,2), 図 3(d) は fat tree (4,4,2) である.

fat tree では up*/down* ルーティングと呼ばれる適応型ルーティングが用いられる. これは, 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), LCA から宛先までリーフ方向へ進む (down). $p > 1$ のとき up 方向の転送には複数の代替経路がある (適応型ルーティング) が, down 方向の転送には単一の経路しかない (固定型).

4.3.1 ルータアーキテクチャ

- **オリジナル:** ここでは $p = 4$ 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, 最近最も使われていない上側チャネルに投機的にパケットを転送する. 一方, 上側チャネルにとっては, 正しい出力チャネルは 1 つしかないため予測が当たりにくい.

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が多いが, 1-cycle 転送できるチャンスも多い.

4.3.2 無負荷時の通信遅延

オリジナルルータ (Orig) と 2 種類の予測ルータの無負荷時通信遅延を求めた (図 6(a)). グラフより, 256-core の fat tree において, Pred(LRU+LP) は Orig と比べて通信遅延を 30.7% 削減できた.

4.3.3 ハードウェア量

図 6(b) 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% である.

4.3.4 消費エネルギー

ここでは 55%の予測が当たると仮定すると, Pred(LRU+LP) の E_{sw} に関するオーバヘッドは高々 9.0%である.

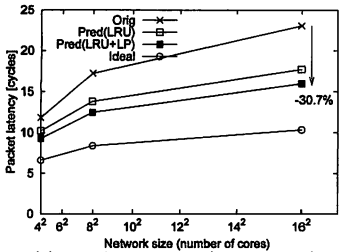
4.4 Case Study 4: Spidergon ネットワーク

case study 4 では Spidergon [14] を想定する. Spidergon は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) を付加したトポロジである. 図 3(b) に示す mesh-like なレイアウト方法が考案されている.

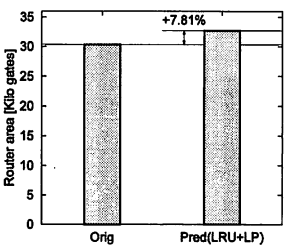
ここでは across-first ルーティング [14] を用いる. これは, 送信元ノードでのみ across link を利用でき, 以降は clockwise 方向, もしくは, anticlockwise 方向のどちらか一方を使う.

4.4.1 ルータアーキテクチャ

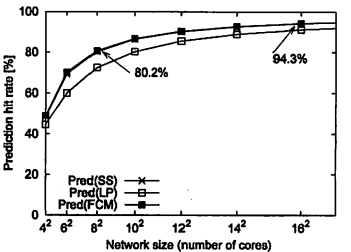
- **オリジナル:** 各ルータは across 方向, clockwise 方向, anticlockwise 方向, コアとの接続のために 4 個のチャネルを持つ. デッドロックフリーのため仮想チャネルを 2 本使用する.
- **予測ルータ:** 予測アルゴリズムとして SS, LP, FCM を用いる. なお, コアからの入力チャネル, および, across link からの入力チャネルでは SS がヒットしないため, SS において



(a) Zero load latency (case study 3)



(b) Hardware amount (case study 3)



(c) Prediction hit rate (case study 4)

図 6 case study 3 と 4 の評価結果.

も部分的に LP を用いるものとした.

4.4.2 予測ヒット率

図 6(c) に Orig と 3 種類の予測ルータの予測ヒット率を示す. Spidergon は 2-D mesh と比べて node degree が小さいため, 経路の多様性が mesh より低く予測ヒット率も高い. 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 95% 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. その結果, 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% 小さく, 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 16×16 mesh ネットワーク向けに 65nm CMOS プロセスを用いて予測ルータを実装した. 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 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に対し, 複数個の予測アルゴリズム (SS, LP, FCM) を実装し, これらを 1-cycle で切り替えられるようにした. これらを 11 種類のトラフィックパターンで評価したところ, 予測ヒット率は 12.0% から 89.8% までの開きがあった.
- **Case study 3:** 適応型ルーティングを用いる 256-core の fat tree ネットワークに予測ルータを適用し, 最大 30.7% の通信遅延を削減した.
- **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, 64-core の Spidergon で最大 46.9% の通信遅延を削減した.

以上より, 1) 予測ルータは様々なネットワーク環境に適用でき, 2) 少ないオーバヘッドで通信遅延を削減できること. また, これまでの単一ポリシに基づいた低遅延ルータ [6] [7] [8] [9] [10] ではなく, 3) 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.

文献

[1] W. J. Dally and B. Towles: "Route Packets, Not Wires: On-Chip Interconnection Networks", Proceedings of the Design Automation Conference (DAC'01), pp. 684–689 (2001).

[2] 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, 1, 1, pp. 28–39 (2008).

[3] 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, 1, 2, pp. 59–69 (2008).

[4] H. Matsutani, M. Koibuchi, H. Amano and T. Yoshinaga: "Prediction Router: Yet Another Low Latency On-Chip Router Architecture", Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA'09) (2009). (to appear).

[5] W. J. Dally and B. Towles: "Principles and Practices of Interconnection Networks", Morgan Kaufmann (2004).

[6] A. Kumar, L.-S. Peh, P. Kundu and N. K. Jha: "Express Virtual Channels: Towards the Ideal Interconnection Fabric", Proceedings of the International Symposium on Computer Architecture (ISCA'07), pp. 150–161 (2007).

[7] D. Park, R. Das, C. Nicopoulos, J. Kim, N. Vijaykrishnan, R. Iyer and C. Das: "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).

[8] C. Izu, R. Beivide and C. Jesshope: "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] G. Michelogiannakis, D. N. Pnevmatikatos and M. Katevenis: "Approaching Ideal NoC Latency with Pre-Configured Routes", Proceedings of the International Symposium on Networks-on-Chip (NOCS'07), pp. 153–162 (2007).

[10] M. Koibuchi, H. Matsutani, H. Amano and T. M. Pinkston: "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] M. Burtscher and B. G. Zorn: "Hybrid Load-Value Predictors", IEEE Transactions on Computers, **51**, 7, pp. 759–774 (2002).

[12] P. Jacquet, W. Szpankowski and I. Apostol: "A Universal Predictor Based on Pattern Matching", IEEE Transactions on Information Theory, **48**, 6, pp. 1462–1472 (2002).

[13] D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, A. Woo and M. Yarrow: "The NAS Parallel Benchmarks 2.0", NAS Technical Reports NAS-95-020 (1995).

[14] L. Bononi and N. Concer: "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).