2004－ARC－158　(4)
2004／5／14

クラスタ化スーパスカラ・プロセッサにおける レジスタ・ファイルの階層化と選択的広域通信制御

望月厚志[†]　嶋田創[†]
安藤秀樹[†]　島田俊夫[†]

近年のプロセッサでは，機能ユニット数や物理レジスタ数はプロセッサの世代ごとに増加する傾向にある．しかし，機能ユニット数や物理レジスタ数の増加は配線遅延の増加を招き，クロック周波数に悪影響を与えることになる．この配線遅延を削減する方法として，機能ユニット，レジスタ・ファイルなどを複数の塊に分けるクラスタ化が有望である．しかし，クラスタ化を行っても，レジスタ・ファイルのエントリ数とポート数は依然として大きく，配線遅延は削減されないため，クロック周波数に悪影響を与えると考えられる．このレジスタ・ファイルのエントリ数とポート数を削減する手法として，レジスタ・ファイルの階層化がある．

本論文では，クラスタ化プロセッサに階層化レジスタ・ファイルを組み込み，上位のレジスタ・ファイルの汚染を防ぐため，上位レジスタ・ファイルへの書き込みを制限することを提案する．評価の結果，すべての結果を上位レジスタ・ファイルに書き込む単純な方法と比較して，IPC は平均で 8.9%向上するという結果を得た．

Global Transmission Control in a Clustered Superscalar Processor with Multi-Level Register Files

Atsushi Mochizuki,[†] Hajime Shimada,[†] Hideki Ando[†]
and Toshio Shimada[†]

In recent processors, functional units and physical registers are increasing in proportion to the processor generation. But increasing functional units and physical registers increases the wiring delay, and it makes negative impact on the clock frequency. To reduce the wiring delay, a clustered procesor which divides functional units and a resister file to some clusters is becoming hopeful technique. But even if we crustered the processor, entries and ports of the register file are still large, and it remains negative impact on the clock frequency. To reduce entries and ports of the register file, there's a multi-level register file technique.

In this paper, we implemented the multi-level register file in the clustered processor, and we propose techniques which limit writing to the high-level register file to prevent the high-level register file from pollution. Our evaluation results show that our technique improves IPC by an average of 8.9% than a easy technique which writes all result in the high-level register file.

1. はじめに

近年のプロセッサの性能向上は，命令の並列性の利用とクロック周波数の向上により達成されている．プロセッサの世代が進むに伴い，命令の並列性をより多く抽出するため，機能ユニット数や物理レジスタ数は増加している．しかし，近年のプロセス技術では配線遅延が問題となるため，長いバイパス経路が必要となる機能ユニット数の増加や，長い配線が必要となる巨大なレジスタ・ファイルはクロック周波数を制限する要素となる．

このバイパス経路の配線を短くし，配線遅延を低減するため，文献 1) では機能ユニット，レジスタ・ファイル，命令ウィンドウをいくつかの塊に分けるクラスタ化が提案されている．クラスタ化を行うと各クラスタ内の機能ユニット数が減少するので，バイパスの配線長は短くなり，バイパス論理も簡単になる．これによりバイパス回路の遅延時間は減少する．

一方，クラスタ化を行っても，レジスタ・ファイルのアクセス時間はあまり短縮できない．その理由を以下に述べる．まず，レジスタ・ファイルのアクセス時間は，レジスタ・ファイルの面積に依存し，その面積は，レジスタ・ファイルのエントリ数とポート数の 2 乗にほぼ比例する．そして，従来のクラスタ化プロセッサは，各クラスタにクラスタ化前と同じエントリ数のレジスタ・ファイルを持つ．さらに，読み出しポートは各クラスタで 1 サイクル中に発行される命令が必要とする数まで減少するが，書き込みポートはすべてのクラスタの結果を書き込む必要があるために減少しない．ゆえに，エントリ数と書き込みポート数が不変のレジスタ・ファイルを持つクラスタ化プロセッサでは，レジスタ・ファイルのアクセス時間をあまり短縮できない．また，エントリ数やポート数が大きなレジスタ・ファイルをクラスタと同数持つという構成は，チップ上の大きな面積を占有するという点や，チップの消費電力を増加させる

† 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University

という点からも好ましくない．

このレジスタ・ファイルのエントリ数を削減し，レジスタ・ファイルのアクセス時間を向上させる手法として，文献 2) ではレジスタ・ファイルの階層化が提案されている．これは，レジスタ・ファイルを小容量で高速な上位レジスタ・ファイルと，大容量で低速な下位レジスタ・ファイルの 2 階層に分割する方法である．この上位レジスタ・ファイルを**レジスタ・キャッシュ(RC)** と呼び，下位レジスタ・ファイルを**メイン・レジスタ・ファイル (MRF)** と呼ぶ．命令が階層化レジスタ・ファイルから値を得る時は，まず RC にアクセスする．もし，必要な値が RC にあれば，1 サイクル後に値を得ることができる．一方，RC に値が無い場合は，MRF から複数サイクルかけて値を得ることになる．階層化レジスタ・ファイルの利点は 3 点ある．第 1 の点は，RC のアクセス時間が短いため，クロック周波数の向上が望める点である．第 2 の点は，RC のヒット率が高ければ，階層化されていない 1 サイクルでアクセスできるレジスタ・ファイルとほぼ同等の IPC を維持できる点である．第 3 の点は，すべての値を MRF から読み出さないので，MRF のポート数を削減できる点である．

本研究ではクラスタ化プロセッサのレジスタ・ファイルのエントリ数を削減するため，クラスタ化プロセッサのレジスタ・ファイルの階層化を行う．クラスタ化プロセッサに階層化レジスタ・ファイルを組み込む場合，RC は各クラスタに 1 つずつ置き，MRF は全体に 1 つ置くという構成を取る．

従来の階層化レジスタ・ファイルでは，結果は全て RC に書き込むという手法を取る．しかし，ある命令の実行結果に着目した場合，クラスタ化プロセッサでは，クラスタによって結果を用いるクラスタと用いないクラスタが存在する．全ての結果を書き込むというポリシーでは，そのクラスタで不要な結果を書き込むことがある．この場合，そのクラスタの RC を不要な値で汚染することになる．もし，結果を用いるクラスタの RC へのみ結果を書き込むよう，選択的に書き込みを行うことができるならば，不要な値による RC の汚染を防ぎ，RC のヒット率を向上させることが可能となる．本論文では，on-the-fly と広域通信予測という，RC への 2 つの書き込み制御手法を提案する．提案する方法を評価した結果，従来の全ての結果を書き込む方法と比較して，IPC は on-the-fly を用いることによって 8.9%，広域通信予測を用いることによって 6.8%向上するという結果を得た．

2 節では，階層化レジスタ・ファイルを組み込んだクラスタ化プロセッサの構造を説明し，3 節では，提案する RC への書き込み制御方式を述べる．そして，4 節では評価環境を述べ，5 節では評価結果を示す．最後に，6 節で本論文をまとめる．

2. 階層化レジスタ・ファイルを組み込んだクラスタ化スーパスカラ・プロセッサと選択的広域通信制御

階層化レジスタ・ファイルを組み込んだクラスタ化スーパスカラ・プロセッサは，従来では各クラスタに置かれていたレジスタ・ファイルの代わりに，各クラスタに RC を置く．そして，MRF は全体に 1 つだけ置き，すべてのレジスタ値を保持する．構成を**図 1** に示す．

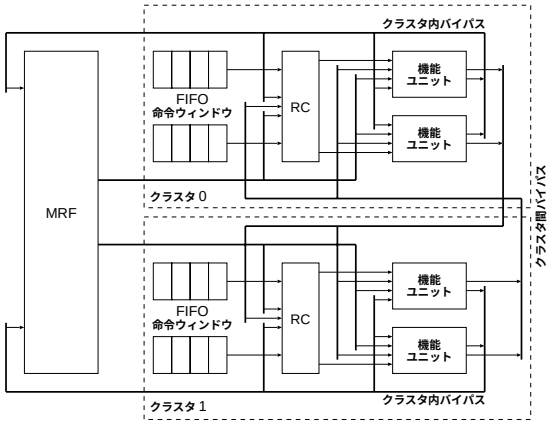

図 1 階層化レジスタ・ファイルを組み込んだクラスタ化スーパスカラ・プロセッサ

結果が出力されたときは，必ず MRF にその結果を書き込むが，RC には後述するように，様々な書き込みポリシーのもとに書き込むことになる．この RC への書き込みは，結果を出した命令の属するクラスタの RC への書き込みと，それとは異なるクラスタの RC への書き込みに分けることができる．後者のように，値を出力したクラスタから別のクラスタに放送することを以下，**広域通信**と呼ぶことにする．広域通信は，クラスタ間の長い配線による遅延のため，数サイクルを要する．

前提としているクラスタ化プロセッサの命令ウィンドウは，回路の複雑さを軽減するため，完全連想の命令ウィンドウではなく FIFO(First In First Out) キューで実現する[1)]．FIFO を用いる命令ウィンドウの特徴は，FIFO 内の命令が，常に直前の命令に依存していることである．命令を FIFO に登録するときは，まず，命令の第 1 ソース・オペランド (src1) と，FIFO の末尾の命令のデスティネーション・オペランドを比較する．一致した場合，つまり，依存関係がある場合は，その依存している命令の FIFO に登録する．依存関係がなかった場合は，第 2 ソース・オペランド (src2) で同様のことを行う．src1,src2 共に FIFO の末尾の命令に依存していない場合や，依存関係があっても FIFO が一杯で命令を登録できないときは，空の FIFO に登録する．もし，空の FIFO が 1 つもないときは命令の登録を停止し，FIFO に空きができるのを待つ．

3. 選択的広域通信制御

RC の容量は小さいため，有効な値を保持しなければ RC ミスが増加し，性能低下を引き起こす．したがって，どの値を RC に書き込むかが重要になる．

最も単純な書き込みポリシーは，すべての結果をすべての RC に書き込むものである．ただし，この方法は不必要な値も書き込むために，必要な値を RC から失う確率が高くなり，RC ミスの増加によって性能が低下してしまう．そこで，本論文では広域通信による書き込みに注目し，結果の中から，あるクラスタで生成される値が，別のクラス

タで使用される場合を見つけ，その値のみを広域通信する方法を提案する．以後，この方法を**選択的広域通信**と呼ぶ．

3.1 on-the-fly

on-the-fly による選択的広域通信とは，命令の結果が出る前に，その結果に依存している命令が別のクラスタに存在するか調査し，存在する場合は，そのクラスタの RC に結果を広域通信するという方法である．この方法は，今後，確実に結果を使用するクラスタの RC のみに広域通信するため，不要な値で RC が汚染されることを防ぎ，効率よく RC を使用することができる．

on-the-fly の実現方法として，request&write と check&write の 2 種類の方法を提案する．以下，それぞれの方法について説明する．

3.1.1 request&write

request&write では，on-the-fly を実現するために，まず，物理レジスタを索引とするクラスタ番号表を準備する．クラスタ番号表の各エントリは，結果が出力されるクラスタ番号からなる．また，各クラスタの FIFO の各エントリに広域通信先を示すフラグを追加する．これは，クラスタ番号に対応するビットベクタである．さらに，広域通信先フラグの更新のため，FIFO 中のデスティネーション物理レジスタ番号のエントリを CAM にする．

クラスタ番号表と広域通信先フラグを用いて on-the-fly により選択的広域通信する手順を，**図 2** を用いて示す．まず，図 2(a) に示すように，命令を FIFO に登録するとき，その命令のデスティネーション物理レジスタ番号 (dst) に対応するクラスタ番号表のエントリに命令が登録されるクラスタ番号を書き，FIFO の広域通信先フラグをすべて 0 とする．同時に，図 2(b) に示すように，ソース物理レジスタ番号 (src) でクラスタ番号表を引き，読み出されたクラスタ番号と当該命令が登録されるクラスタ番号を比較する．もし，クラスタ番号が一致しなければ，先行命令の実行結果を当該命令が登録されるクラスタへ広域通信する必要があると判断し，FIFO 中の先行命令の広域通信先フラグのうち，当該命令のクラスタに対応するフラグを立てる．このとき，FIFO 中の先行命令を探すために，当該命令の src で FIFO 中の dst を連想検索し，一致するエントリの広域通信先フラグを適切にセットする．命令は発行される時に FIFO 中の広域通信先フラグを参照し，フラグの立っているクラスタへ結果を広域通信する．

クラスタ番号表への登録はレジスタ・リネーミング後に，命令の FIFO への登録と同時に行うことになる．また，広域通信先フラグの参照は命令の発行と同時に行われる．よって，クラスタ番号表への登録と広域通信フラグの参照のどちらにおいてもタイミング上の問題はなく，この方法によるクロックサイクル時間への影響はない．

3.1.2 check&write

check&write では，on-the-fly を実現するために，各クラスタに RC 書き込み許可表を設ける．これは，各物理レジスタに対して，そのクラスタの RC への書き込み許可を示す表である．表のインデックスは物理レジスタ番号，エントリの内容は書き込み許可を示す 1 ビットのフラグである．

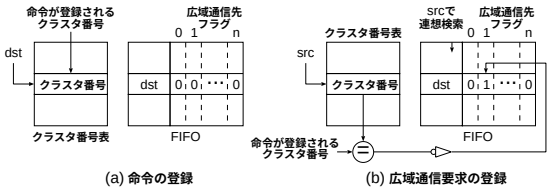

図 2 request&write の動作

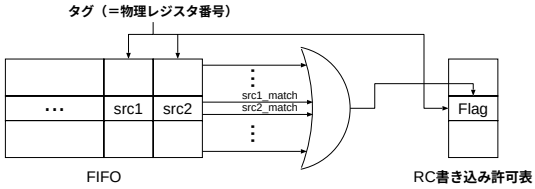

図 3 check&write の動作

この RC 書き込み許可表を用いて，on-the-fly により選択的広域通信する手順を，**図 3** を用いて示す．まず，他のクラスタの命令の実行結果が送られてくる 1 サイクル以上前に，その結果が使用可能となることを示すタグが送られている．このタグは実行結果が格納される物理レジスタ番号であり，タグは FIFO に放送され，src と比較される．もし，その結果を用いる命令があるならば，タグは一致し，対応する ready ビットをセットすることになる．逆にいえば，FIFO 内でタグの一致がなければ，その FIFO 内には結果を必要とする命令が存在しないことになる．よって，FIFO 内でタグの一致があった場合はそのクラスタの RC に結果を書き込み，一致がなければ書き込まないようにすれば，on-the-fly を実現できる．RC 書き込み許可表のタグに対応するエントリには，全 FIFO のエントリのタグの比較結果の OR を取った結果を書き込む．他のクラスタから放送された結果はこの RC 書き込み許可表のフラグを参照し，そのクラスタの RC への書き込みを決定する．

タグが，あるクラスタの FIFO に到着する時刻は，結果がそのクラスタに到着する時刻よりも 1 サイクル以上前であるため，RC 書き込み許可の決定が結果到着より遅くなることはない．また，RC への書き込み前に，RC 書き込み許可表を参照する必要がある．この RC への書き込み許可は，書き込むエントリの選択終了までに判明すればよいため，RC 書き込み許可表の読み出しは LRU で書き込むエントリを選択することと並行して行うことができる．よって，RC 書き込み許可表への登録と参照のどちらもにおいてもタイミング上の問題はなく，この方法によるステージの追加はない．

3.2 広域通信予測

広域通信予測とは，ある命令の実行結果が，他のクラスタから参照された回数を数え，その回数より，次に同一の命令が実行されるときに広域通信するかどうか判断する方法である．

まず，**図 4**(a) に示すように，どの命令を広域通信するか示す広域通信判定表を用意する．この表は，命令アドレスの下位ビットを索引とし，命令識別のためのタグ，広域通信の可否を判断するための 2 ビット飽和カウンタ (2bc)，

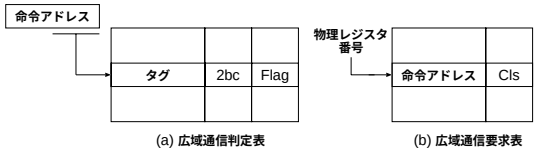

図 4 広域通信予測器

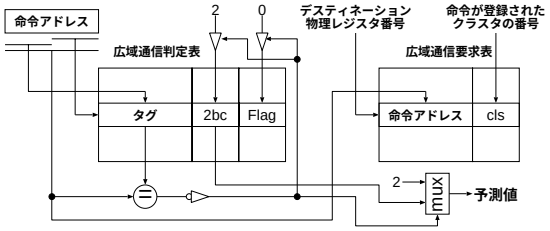

図 5 命令の FIFO 登録時

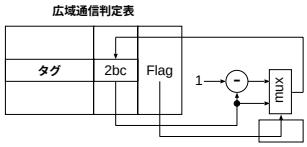

図 6 2bc の更新

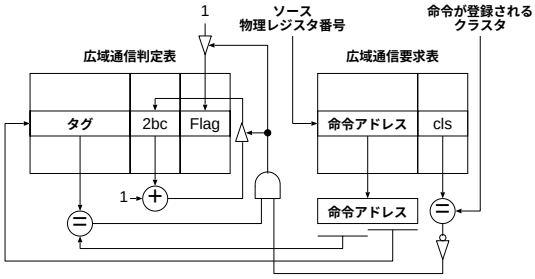

図 7 広域通信要求の登録

前回の命令の FIFO 登録後に 2bc の更新があったかどうかを示す更新フラグ (Flag) を持つ．命令識別のためのタグは命令アドレスのうち，索引として用いなかった上位ビットを使用する．また，図 4(b) に示すように，ある物理レジスタに対する広域通信要求を広域通信判定表に反映するために，広域通信要求表を用意する．この表は，物理レジスタ番号を索引とし，命令アドレス，クラスタ番号 (Cls) を登録する．

以下，広域通信の判断方法，広域通信判定表と広域通信要求表の更新手順を説明する．

(1) 命令の FIFO 登録時 (**図 5**)

- 命令アドレスの下位で広域通信判定表を引き，タグを比較する．
  - タグが一致： すでに命令は広域通信判定表に登録されている
    - 前回の命令の登録後に広域通信要求があったかどうかにより，2bc を更新する．(**図 6**)
      - Flag が 0：
        前回の命令の FIFO 登録後に広域通信要求はなかったので，2bc をデクリメントする．
      - Flag が 1：
        前回の命令の FIFO 登録後に広域通信要求があったので，2bc の値は変化させない．
    - 2bc の値を用いて広域通信予測を行う．
    - Flag を 0 にする．
  - タグが不一致：
    - 広域通信判定表にタグ，2bc の初期値 2(広域通信すると弱く予測)，Flag の初期値 0 を書き込む．
    - 2bc の初期値 2 に準じ，広域通信は行うものとする．
- 広域通信要求表のデスティネーション物理レジスタ番号に対応するエントリに，命令アドレスと命令が登録されるクラスタ番号を書く．

(2) 後続命令による広域通信要求の登録 (**図 7**)

- ソース物理レジスタ番号で広域通信要求表を引き，表から得られた命令アドレスで広域通信判定表を引き，タグを比較する．また，表から得られたクラスタ番号と命令が登録されるクラスタ番号を比較する．
  - タグは一致しクラスタ番号は異なる：
    先行する命令に依存があり，なおかつ，その結果を出力されるクラスタが異なるため，広域通信判定表の 2bc をインクリメントする．また，Flag に 1 をセットする．
  - 上記以外：
    依存元が表に登録されていないか，依存があっても結果が同一クラスタで生成されるので，広域通信判定表の更新は行わない．

表 1 ベンチマーク

ベンチマーク	入力	実行命令数
compress95	bigtest.in	95M
gcc	genoutput.i	84M
go	2stone9.in	75M
ijpeg	specmun.ppm	450M
li	train.lsp	183M
m88ksim	ctl.in	100M
perl	scrabbl.in	80M
vortex	vortex.in	80M

4. 評価環境

評価には，SimpleScalar Tool Set[3] 中のスーパスカラ・プロセッサ用のシミュレータに，提案機構を組み込んだものを用いた．命令セットは SimpleScalar/PISA である．ベンチマーク・プログラムは，SPECint95 の全 8 種類を使用した．ベンチマーク・プログラムのバイナリは，gcc ver. 2.7.2.3，コンパイル・オプション -O6 -funroll-loops を用いて作成した．ベンチマークと実行命令数を**表 1** に示す．

表 2 にプロセッサのモデルを示す．このプロセッサでは，RC ミス後に MRF に 3 サイクルかけてアクセスするため，RC ミス時には 4 サイクル後に値を利用できることになる．また，広域通信に要するサイクル数は 2 サイクルとした．

また，文献 2) に提案されているプリフェッチ機構を改良したものを用い，RC ミス率を改善している．文献 2) では，FIFO の先頭にある命令が発行されたときに，FIFO の先頭から 2 番目の src を MRF にアクセスし，読み出す方法が提案されている．しかし，この方法の欠点は，MRF の

表 2　プロセッサモデル

命令発行幅	16 命令
リオーダ・バッファ	256 エントリ
ロード/ストア・キュー	128 エントリ
RC(int,fp 共に)	16 エントリ, 完全連想, LRU 置き換え
MRF(int,fp 共に)	256 エントリ
クラスタ数	8
1 クラスタあたりの FIFO 数	2
1FIFO のエントリ数	8
機能ユニット	16 iALU, 16 iMULT/DIV, 16 Ld/St, 16 fpALU, 16 fpMULT/DIV/SQRT
分岐予測	履歴長 11bit/インデクス長 16bit の gshare, 1K エントリ/4-way の BTB, 32 エントリ RAS
命令キャッシュ	完全
データ・キャッシュ	32KB,32 バイト・ライン, 2-way,LRU 置き換え, ヒット・レイテンシ 1 サイクル, ミス・レイテンシ 6 サイクル
MRF アクセス・サイクル数	3 サイクル
広域通信サイクル数	2 サイクル
広域通信判定表	8K エントリ

アクセス・レイテンシが長い場合，プリフェッチが完了する前に依存命令が発行されてしまう点である．そこで，本論文では命令の FIFO 登録時にプリフェッチを行い，MRF から早めに値を得る方法に変更した．この方法の利点は，命令の FIFO 登録時に MRF から値を得るので，命令が発行される前に値が RC に書き込まれる可能性が高くなる点である．一方，欠点は，値を読み出してから命令が発行されるまでの時間が長い場合，別の値により RC の当該エントリが上書きされてしまう確率が高くなる点である．

5. 評価結果

階層化レジスタ・ファイルを組み込んだクラスタ化プロセッサで，広域通信をしないモデル，すべての結果を広域通信するモデル，on-the-fly で選択的広域通信するモデル，広域通信予測で選択的広域通信するモデル，理想モデルについて評価した．

5.1 広域通信精度

on-the-fly と広域通信予測のそれぞれの場合で，広域通信がどの程度正しく行われているか評価した．

まず，**図 8** に on-the-fly の通信精度を示す．グラフの縦軸は on-the-fly が成功した割合と失敗した割合を示し，横軸は各ベンチマークを表す．各ベンチマークごとの積層棒グラフは，下より以下の割合を示している．

(1)　on-the-fly で結果の必要なクラスタに広域通信を行えた場合
(2)　広域通信する必要がなかった場合
(3)　広域通信が必要であったが，on-the-fly では広域通信を行えなかった場合

(1)(2) の和が on-the-fly で行う広域通信が成功した割合となり，(3) が on-the-fly で行う広域通信が失敗した割合となる．なお，上記の割合は，実際に RC にヒットしたかどうかまでは調べていない．よって，広域通信された値を後続命令が用いる前に，他の結果によって広域通信された値が上書きされても，on-the-fly による広域通信は成功したものとしてカウントしてある．これは，後述する広域通信予測の予測精度ついても同様である．

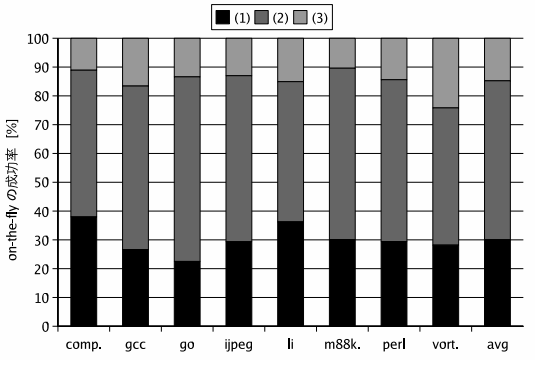

図 8　on-the-fly の成功率

比較対象である，すべての結果を広域通信が必要と予測した場合，広域通信の成功率，つまり，広域通信した値が使われた確率は 44.7%であった．これに対し，on-the-fly の広域通信の成功率は平均 85.5%であり，すべての結果を広域通信する場合と比較して，高い通信精度を達成した．よって，on-the-fly は広域通信が必要かどうかを有効に判断できているといえる．

図 9 に広域通信予測の予測精度を示す．縦軸は予測が成功した割合と失敗した割合を示し，横軸はベンチマークを表す．各ベンチマークの積層棒グラフは，下より以下の割合を示している．

(1)　広域通信が必要と予測し，実際に広域通信が必要であった場合
(2)　広域通信が不要と予測し，実際に広域通信が不要であった場合
(3)　広域通信が必要と予測したが，実際は広域通信が不要であった場合
(4)　広域通信が不要と予測したが，実際は広域通信が必要であった場合の割合

(1)(2) の和が広域通信予測が成功した割合となり，(3)(4) の和が広域通信予測が失敗した割合となる．

前述したようにすべての結果を広域通信が必要と予測した場合，広域通信の成功率は 44.7%である．これに対し，広域通信予測ではヒット率が 84.4%であり，高い予測精度を達成した．よって，広域通信予測は広域通信が必要かどうかを有効に予測できているといえる．

5.2 RC ミス率

測定により得られた RC ミス率を**図 10** に示す．図の縦軸は RC ミス率を示し，横軸はベンチマークの種類を表す．そして，ベンチマーク毎の 5 本の棒グラフは，左から広域通信をしないモデル，すべての結果を広域通信するモデル，on-the-fly で選択的広域通信するモデル，広域通信予測で選択的広域通信を行うモデルを示してある．

広域通信をしないモデルでは RC ミス率は平均で 57.1%であり，RC 読み出しのうち半分以上がミスするという結果になった．それに対し，全ての結果を広域通信するモデルでは，RC ミス率は平均で 39.5%となっており，広域通

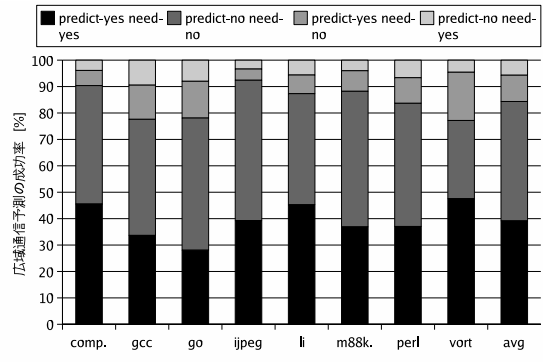

図 9 広域通信予測の成功率

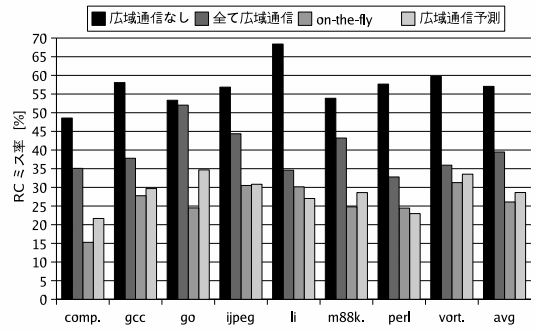

図 10 RC ミス率の変化

信をしないモデルよりも RC ミスが 17.6%ポイント少ない．全く広域通信をしないモデルよりは広域通信をした方が性能が良いため，広域通信予測の初期値は広域通信を行なうことにしている．

提案する on-the-fly，広域通信予測を用いたモデルでは，RC ミス率はそれぞれ平均で 26.1%，28.6%であり，それぞれ全ての結果を広域通信するモデルと比較して 13.4%ポイント，10.9%ポイント削減できるという結果になった．図 8 と図 9 より，on-the-fly と広域通信予測の成功率の平均の差は 1%ポイントだが，RC のミス率では 2.5%ポイントの差と，より大きな差となっていることがわかる．これは，広域通信予測では全てのクラスタに広域通信するか否かしか選択できないが，on-the-fly ではクラスタ単位で広域通信の有無を選択できるという点が RC の汚染を防ぎ，RC ミス率の差として現れたと考えられる．

5.3 IPC

測定により得られた IPC を図 11 に示す．図の縦軸は IPC を示し，横軸はベンチマークの種類を表す．そして，ベンチマーク毎の 4 本の棒グラフは，左から広域通信をしないモデル，すべての結果を広域通信するモデル，on-the-fly で選択的広域通信するモデル，広域通信予測で選択的広域通信を行うモデルを示している．

全く広域通信をしないモデルと全ての結果を広域通信するモデルでは，全ての結果を広域通信するモデルの方が IPC は平均で 34.2%高い．そして，提案する on-the-fly，広域通信予測を用いたモデルでは，全ての結果を広域通信するモデルよりもそれぞれ IPC は平均で 8.9%，6.8%向上

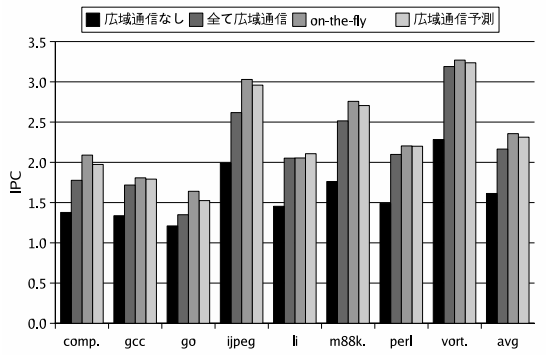

図 11 IPC の変化

している．on-the-fly と広域通信予測の IPC の差は，RC ミス率の項で述べたように，同じ広域通信の成功率ならば，クラスタ単位で広域通信の有無を決めることができる on-the-fly の方が，RC の汚染が少なくなるためである．

6. ま と め

プロセスが微細化しても配線遅延はあまり減少しないので，将来のプロセス技術では配線遅延が重要になる．配線遅延が支配的な回路にバイパス論理があり，その配線長を短くし，配線遅延を低減する方法にクラスタ化がある．しかし，クラスタ化プロセッサは，各クラスタにエントリ数とポート数の多いレジスタ・ファイルを持つため，レジスタ・ファイル・アクセス時間が大きく，クロック周波数の向上が妨げられてしまう．

レジスタ・ファイルのアクセス時間を削減する方法に，レジスタ・ファイルの階層化がある．本論文では，クラスタ化プロセッサのレジスタ・ファイルを階層化し，RC のヒット率に影響を与える広域通信による RC への結果の書き込み方法を研究し，on-the-fly と広域通信予測を提案した．

評価の結果，すべての結果を広域通信する場合に対し，on-the-fly，広域通信予測を用いるモデルは，IPC はそれぞれ平均で 8.9%，6.8%向上するという結果を得た．広域通信の成功率はどちらの方法もほぼ同じであったが，クラスタ単位で広域通信の有無を決めることができる on-the-fly の方が，RC の汚染が少なくなるため，より高い IPC を達成できるという結果になった．

参 考 文 献

1) Palacharla, S., Jouppi, N. P. and Smith, J. E.: Complexity-Effective Superscalar Processors, *Proc. of 24th Int. Symp. on Computer Architecture*, pp. 206–218 (1997).
2) Cruz, J.-L., González, A. and Valero, M.: Multiple-Banked Register File Architectures, *Proc. of 27th Int. Symp. on Computer Architecture*, pp. 316–325 (2000).
3) Burger, D. and Austin, T. M.: The SimpleScalar Tool Set, Version 2.0, Technical Report CS-TR-97-1342, University of Wisconsin-Madison Computer Sciences Dept. (1997).