C-017

クラスタ化アーキテクチャにおける非重複分散レジスタファイルの評価
Evaluation of Non-consistent Register Files on Clustered Architecture

佐藤 幸紀† 鈴木 健一† 中村 維男†
Yukinori Sato Ken-ich Suzuki Tadao Nakamura

1. まえがき

半導体加工技術の微細化に伴い、1 チップのマイクロプロセッサ上に 10 億個以上のトランジスタが集積可能であると予測されている。これら豊富なハードウェア資源を処理時間短縮のために有効活用するマイクロプロセッサアーキテクチャが望まれている。一方で、半導体プロセスの微細化は、チップ上の長距離配線の遅延を増加させ、結果的にマイクロプロセッサの性能を制限するという問題を引き起こす。それゆえ、プログラム中に内在する処理の局所性を利用し、チップ上において局所的処理を行うことにより、配線遅延の影響を最小化しつつ並列実行を行う必要がある。

スーパースカラ方式の広域的かつ複雑な構造は、将来的に性能向上の妨げとなると予想される。そこで、スーパースカラ方式の単一レジスタファイルや自由度の高い演算資源間ネットワーク等の構造を局所化された単純な処理要素（PE）に分割するクラスタ化アーキテクチャが提案されている [1][2]。クラスタ化アーキテクチャは、プログラムに内在する処理の局所性を PE の局所性に対応させながら処理を進行するため、並列性と逐次性 [3] のバランスのとれた将来的に有望なプロセッサアーキテクチャの一つであるといえる。

クラスタ化アーキテクチャは、演算において必要なオペランドを各 PE に分割されたレジスタファイルに保持する。先行する研究の多くは、分散された各レジスタファイルに同一の内容を重複させて保持するという方法を採用している [2]。文献 [1] においては、一部の内容のみ重複させるという方式を用いているが、レジスタファイルを一部重複させることの評価および必要となるレジスタ数の議論がなされていない。分散された各レジスタファイルの内容が不必要に重複されると、レジスタリネーミングの際に必要な利用可能なリネーミングレジスタの数を十分確保できないため、各クラスタに分散された物理レジスタを有効に利用しているとはいえない。

本報告では、クラスタ化アーキテクチャにおいて効率的にレジスタファイルを活用するために、非重複分散レジスタファイル（Non-consistent RF）を用いる。レジスタファイルを非重複分散させることにより、重複したデータを PE 間にまたがって保持しないため、プログラムの局所性をさらに有効活用することができ、配線遅延の影響を緩和することが可能である。また、不必要に重複されたレジスタファイルをレジスタリネーミングに有効活用することにより、リネームレジスタが欠乏することを避けることが可能である。しかし一方で、PE 毎のレジスタ利用のばらつきにより、特定の PE において空きリネームレジスタが足りなくなるために、その PE への割付けが行えず、空きリネームレジスタを持つ PE へ再割付けが必要になる場合がある。PE の再割付けが行われた場合、PE 間の通信が必要となり、IPC（Instruction Per Clock）を低下させるおそれがある。本報告では、非重複分散レジスタファイルを用いたクラスタ化アーキテクチャの IPC を見積もることにより、非重複分散レジスタファイルの有効性を明らかにする。

†東北大学大学院情報科学研究科

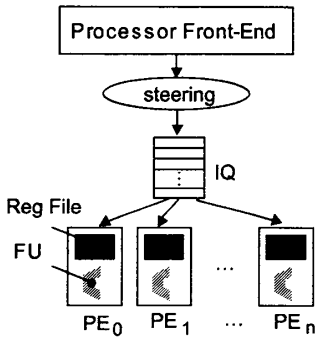

図 1: クラスタ化アーキテクチャの概要

本論文は以下のように構成される。2 節では本論文において想定するデータ依存に基づくクラスタ化アーキテクチャの概要と非重複分散レジスタファイルを実現するために必要となる拡張について述べる。3 節では非重複分散レジスタファイルの有効性を確認するための評価実験を行う。4 節で本論文をまとめる。

2. クラスタ化アーキテクチャ

図 1 に本論文で想定するデータ依存に基づくクラスタ化アーキテクチャを示す。クラスタ化アーキテクチャにおいては、従来のスーパースカラ方式の備える単一のマルチポートレジスタファイルや自由度が高く複雑なフォワーディング回路は、配線遅延の影響を避けるために各 PE に分割される。データ依存関係のある命令列を同一 PE 内で処理することにより処理の局所化を図りつつ、依存のない命令列を異なる PE で並行に処理することにより並列処理を行う。

図 2（a）に本論文で想定するパイプラインのステージを示す。これは、Alpha21264 [4] のパイプラインの構成に基づいている。プロセッサのフロントエンド（IF,ID）は Alpha21264 と同様に複数の命令を同時にフェッチし、続いてデコードする。MAP ステージにおいては、ステアリング機構がそれぞれの命令に対して、どの PE で実行を行うかの割付けを決定する。レジスタリネーミング機構が命令ステアリング機構による命令の割付けを踏まえて演算結果の出力先のレジスタを割付けた後、命令は命令キュー（IQ）に格納される。ISSUE ステージでは、IQ に格納されている命令のオペランドが利用可能かどうかが常に監視される。オペランドが利用可能であれば命令は wakeup され、対応する PE の演算資源が利用可

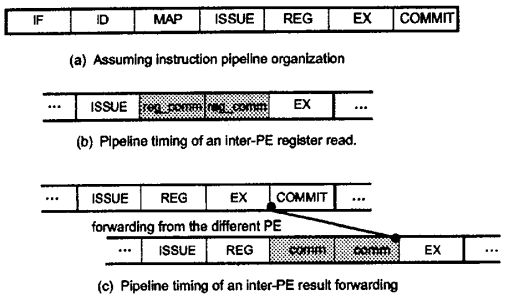

図 2: パイプラインのタイミング

能かどうかが判定される。演算資源が利用可能である場合、命令は select され、REG ステージにおいてレジスタが読み込まれ、EX ステージにおいて実行される。その後、COMMIT ステージにおいてコミットされる。

本論文において、図 1 に示すように、全ての PE が 1 つの IQ を共有すると想定した。これは、本論文の目的である非重複分散レジスタファイルの影響に集中するためである。IQ を PE 毎に分散する場合、各キューの利用状況を考慮に入れる必要がある。分割 IQ の影響の評価は今後の課題である。

本論文では、各 PE のレジスタファイルのレジスタアドレス空間を重複させずに管理することにより、各 PE に分散された物理レジスタを有効活用することを目指す。命令セットにおいて単一のレジスタ空間により定義されているレジスタは、非重複分散レジスタファイルにおいては各 PE に物理的に分散され独立した構成をとる。このため、レジスタリネーミングを行う際に論理レジスタをどの PE のどの物理レジスタにマッピングするかということを管理する必要がある。そこで、本研究では既存の Alpha21264 において使用されているレジスタリネーミング手法を、各 PE が独立して利用可能なリネームレジスタのリスト（free register list）を用いてレジスタを管理するというように変更した。また、特定の PE のレジスタに対してレジスタプレッシャが高くなることを防ぐために、コミット後の値を保持するアーキテクチャレジスタの数を各 PE に等しく分配し、各 PE は等しい数だけリネームレジスタを持つとした。また、レジスタリネーミングは、1 つの論理レジスタが 1 つの物理レジスタに対応するマップテーブルにより記録されるとした。

命令ステアリング機構は、依存のある命令列に対する処理が同一 PE 内において完結するように命令を PE に割付ける。依存のある先行命令が既に実行されている場合、命令の実行に必要な入力オペランドは分散されたレジスタファイルのいずれかに既に存在している。重複分散レジスタファイル方式においては、結果の書き込みが既に行われている解決済みのレジスタは常にどの PE においても利用可能である。しかしながら、非重複分散レジスタファイル方式においては、割付けられた PE 内のレジスタファイルに命令の実行に必要な入力オペランドが存在しない場合がある。この場合、他 PE のレジスタファイルとの通信を必要とする。本論文では、この通信にかかる遅延を 1 サイクルと想定した。図 2（b）は他 PE からのレジスタ読み込みのための PE 間通信のタイミングを示す。

入力オペランドがまだ利用可能でない場合、同一 PE 内では入力オペランドが利用可能になった直後のサイクルにおいてフォワーディングロジックにより結果を利用できるため、逐次的に処理を進行することが出来る。しかし、異なる PE からの結果を必要とするオペランドを持つ場合は、同一 PE 内でのフォワーディングと比べて図 2（c）に示すように 2 サイクル余分に必要と想定した。重複分散方式と非重複分散方式のどちらの構成においても、未解決オペランドの結果のフォワーディングに対する遅延は等しくなる。

非重複分散方式は、重複分散レジスタファイルと比較して、他 PE のレジスタファイルへのアクセスが必要になるが、各 PE に分散された物理レジスタを有効活用することが可能となる。32 個のアーキテクチャレジスタと 48 個のリネームレジスタを持つ場合、8PE 構成のとき重複分散方式は全てを各 PE において重複するために、各 PE に 80 個、全体で 640 個の物理レジスタが必要となる。しかし、非重複分散方式は各 PE が個別の内容を保持するため、各 PE に 10 個、全体で 80 個の物理レジスタで済む。

命令レベル並列性を利用するプロセッサにおいて、レジスタリネーミングのためのリネームレジスタが十分に確保されていない場合、空きリネームレジスタが欠乏し性能が低下する場合がある。不必要にレジスタを重複させる重複分散方式においてリネームレジスタを増加させることは、PE 数に比例した物理レジスタの増加を伴うため、リネームレジスタを十分に確保することは困難となる。非重複分散方式においては、不必要にレジスタを重複させるのではなく、物理レジスタを個別のリネームレジスタとして有効活用することにより、リネームレジスタを十分に確保でき、リネームレジスタの不足に起因する性能低下を防ぐことが可能となる。例えば 32 個のアーキテクチャレジスタと 96 個のリネームレジスタを必要とする構成が、各 PE に 16 個のレジスタにより実現される。必要な物理レジスタの個数が少ないとレジスタファイルを構成するために必要な面積も減少し、レジスタファイルへのアクセス速度も向上する。よって、非重複分散方式の方が優れたレジスタファイルの構成であると予想されるが、適切な PE への命令の割付けが行われない場合、PE 間通信の遅延のため、性能が低下する恐れがある。

本研究では、命令割付を行うステアリング機構として以下の 4 つを使用する。データ依存に基づく方式（dep_based）はデータ依存に基づくクラスタ化アーキテクチャの最も基本的な命令ステアリング方式であり、ある命令のオペランドに依存がある場合、そのオペランドが未解決であろうと解決済であろうと依存のある先行命令が割付けられている PE に命令を割付ける。この方式では、PE 間通信を最小化できるが、負荷集中により性能が低下する。

dep_based 方式に負荷分散を組み込んだ方式（Advanced_RMBS）[1] は、DCOUNT という各 PE にディスパッチされた命令の差分を示すカウンタを用いて負荷の状況を監視し、負荷の集中が起こったとみなした際に負荷が最小の PE に命令を割付けることにより負荷分散

を行う。しかし、負荷集中の閾値を超えると常にPE間通信が必要となるため、不必要なPE間通信がIPCを低下させてしまう。

依存のある命令間の発行時間差に基づき命令ステアリングを行う方式（ldist）[2]は、発行時間差を算出するために、ステアリングされた命令に対してPE毎に通し番号を付ける。この番号の差をLocal distance（ldist）と呼ぶ。未解決のオペランドとその依存する先行命令の間のldistが閾値を越えない場合、dep_based方式と同様のステアリングを行う。その他の場合は、その命令はPE間通信を挿入しても性能には影響がないと判断され、負荷が最小のPEに割付けられる。負荷の状況はPE内の未発行命令数を用いて監視される。ldist方式は優れた命令ステアリング方式であるが、番号の生成や記憶、距離の算出、比較にハードウェアを追加する必要がある。

未解決オペランドを持つ命令をデータ依存に基づきステアリングを行う!ready（not ready）方式は、ldistの距離の閾値を無限大とした場合と同等である。この方式は解決済の命令を負荷最小のPEに割付けるので、dep_based方式と比較して負荷集中が起こりにくい。また、未解決オペランドを持つ命令が常に優先されるとするので、ldist方式のようにldistの算出等に必要なハードウェアを追加する必要がない。

非重複分散レジスタファイルを持つクラスタ化アーキテクチャにおいては、各PEにおいて独立してリネームレジスタが管理されるため、各PEの利用可能なリネームレジスタの数は実行中に変動する。利用可能なリネームレジスタがなくなったPEが存在する場合、そのPEへの割付けは行えない。そこで、我々は空きリネームレジスタのないPEへの割付けを行おうとする場合に、最も空きリネームレジスタの数が多いPEに再割付けを行うということをそれぞれの命令ステアリング方式に追加した。このPEの再割付けが行われる場合、PE間に通信が発生するために重複分散レジスタファイルを用いた場合と比べて性能が低下する可能性がある。

3. 性能評価

3.1 実験条件

クラスタ化アーキテクチャにおける非重複分散レジスタファイルを評価するために、我々はサイクル精度の実行駆動型シミュレータを開発した。ベースとしたシミュレータはsim-alpha[5]というSimpleScalarツールセットの拡張版である。sim-alphaは2個の整数PE（cluster）から構成されるクラスタ化アーキテクチャであるalpha21264のマイクロアーキテクチャの詳細をモデル化している。我々はsim-alphaを改変し、8つの同質な整数PEから構成される8-wayのクラスタ化アーキテクチャをモデル化した。整数PEはそれぞれクロック毎に1つの命令を実行する。主要なアーキテクチャパラメタを表1に示す。残りの構成や、キャッシュ、機能ユニットの遅延はalpha21264と同様とした。

MediaBenchより4つのベンチマーク（djpeg, cjpeg, rawdaudio, rawcaudio）を、SPEC2000CPUintより7つのベンチマークを（gzip, vpr, gcc, mcf, perlbmk, bzip, twolf）を選び実験を行った。MediaBenchは商用のメディアアプリケーションによるベンチマークであり、高い並列性を抽出可能である。SPEC2000は汎用のPC向けベンチマークである。全てのベンチマークはCompaq C compiler v6.5により-O4 -fast -non_sharedオプションを用いてコンパイルされた。MediaBenchの各プログラムは終了するまで命令の実行を行った。SPEC2000intの各プログラムは1G命令フォワードした後の100M命令の実行を行った。

表1: 主要なアーキテクチャパラメタ

Fetch and decode	8 instructions
Branch predictor	Tournament branch predictor
IQ, FQ, LQ, SQ size	64
ROB size	256
Functional Units	1 ALU + 1 MUL per int. PE
Issue width	1 inst per PE
The number of PEs	8 int + 1 fp
Icache	128kB, 2way
Dcache	128kB, 2way

3.2 実験結果

図3と図4に各PEに16個ずつレジスタを持つ構成（reg16）と、PEにつき40個のレジスタを持つ構成（reg40）において異なる命令ステアリング方式を用いた場合のIPCをそれぞれ示す。reg16構成よりもreg40構成の方が平均してIPCが高いことが分かる。ldistの場合、MediaBenchを通しての平均で22%ほどreg40構成の方がIPCが高い。スーパースカラ方式のように単一のレジスタファイルを使用する場合、8-way構成で64個のIQを持つときレジスタ数は128個でIPCが飽和するが[6]、非重複分散レジスタファイルを持つクラスタ化アーキテクチャにおいては128個のレジスタを持つreg16構成でもIPCが飽和しないことが分かった。これは、クラスタ化アーキテクチャにおける各PEのレジスタファイルに対するレジスタプレッシャが異なるためである。リネームレジスタは各PEで独立して管理されるため、実行中に利用可能なリネームレジスタの数はPE毎に異なる。このため、特定のPEでin-flightの状態にある命令数が多い場合に、そのPEのリネームレジスタが欠乏する可能性が高くなる。空きリネームレジスタのないPEへの割付けが起こった場合、最も空きリネームレジスタを多く持つPEに再割付けが行われるため、余分なPE間通信が発生しIPCが低下する。それゆえ、reg16構成のIPCはreg40構成より全体的に低い。

命令ステアリング方式については、ldist方式を用いた場合が常にIPCが高いことが分かる。しかしながら、reg16構成においては!ready方式とldist方式のIPCを比べた場合、ほとんど差が見られなかった。これは、非重複分散レジスタファイルを用いることにより、特定のPEのレジスタプレッシャが増加し、リネームレジスタの不足が起こったためにPEの再割付けが行われたからである。reg16構成の場合は、命令ステアリング方式よりもリネームレジスタ数の不足がIPCを低下させる原因であるといえる。そのため、reg16構成においては特別なハードウェアを必要としない!ready方式が最良の選択であるといえる。

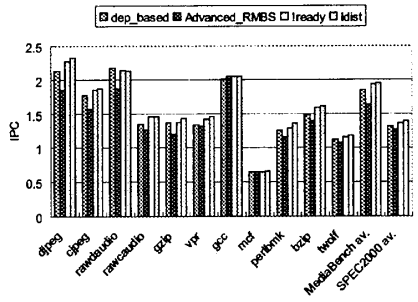

図 3: reg16 構成の IPC

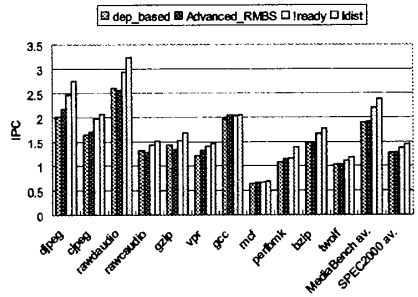

図 4: reg40 構成の IPC

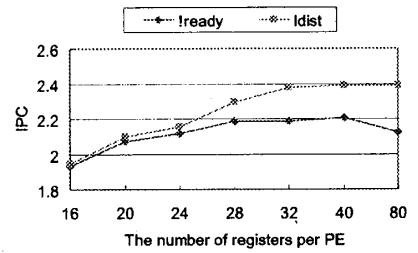

図 5: レジスタ数を変化させた場合の IPC

適切な命令ステアリング方式に加えて、適切なレジスタ数の選択がクラスタ化アーキテクチャにおけるプログラムの処理時間短縮のためには必要である。図 5 に PE 毎のレジスタ数を変化させた場合の MediaBench の平均 IPC を示す。レジスタ数を増加させた場合、IPC が増加していくことが分かる。さらにレジスタ数を増加させていくと、リネームレジスタの不足による PE の再割付けが起こらなくなくなる時点で、IPC が飽和することが分かる。物理レジスタ数を増加させた場合、IPC は増加するが、レジスタファイルへのアクセス時間やハードウェアコストも増加するため、ハードウェアコストも含め、レジスタファイルの適切な構成を探っていく必要がある。また、各 PE のレジスタプレッシャを低減することが可能であれば、レジスタの数を増加させることなく IPC を増加させることができる。Virtual physical renaming[7] のようにリネームレジスタの生存期間を短くする技術によりレジスタプレッシャは低減可能であると考えられる。

4. まとめ

本論文では、クラスタ化アーキテクチャにおけるレジスタファイルの非重複分散の影響について評価を行った。既存の研究では、処理要素に分散されたレジスタファイルに同一のデータを重複させつつ保持させているため、物理的に存在するレジスタを有効活用していない状況にあった。

既存のレジスタの管理機構と命令ステアリング機構に変更を加え、非重複分散レジスタファイルを持つクラスタ化アーキテクチャをモデル化し、クロックサイクル精度の実行駆動シミュレーションを行った。その結果、各処理要素におけるレジスタの利用状況のばらつきのため生じる空きリネームレジスタの不足に対して、不必要にレジスタを重複しないために利用可能となる物理レジスタを用いてレジスタ数を適切に設定することにより、レジスタを有効活用しつつ、PE 間の不必要な通信を回避することが可能であることが分かった。このため、重複分散レジスタファイルと非重複分散レジスタファイルを比較すると、非重複分散レジスタファイルは圧倒的に必要となるレジスタ数が少ないため非常に有効であるといえる。

今後の課題として、重複分散レジスタファイルや部分的に重複を許容する分散レジスタファイルを持つクラスタ化アーキテクチャとの IPC の比較、PE 間の通信経路とレジスタのポート数を踏まえたハードウェア的側面の評価が挙げられる。

参考文献

[1] Parcerisa, J.-M. and Gonzalez, A.: Reducing wire delay penalty through value prediction, in *Proceedings of the 33rd annual international symposium on Microarchitecture*, pp. 317–326 (2000).

[2] 服部直也, 高田正法, 岡部淳, 入江英嗣, 坂井修一, 田中英彦：発行時間差に基づいた命令ステアリング方式, 情報処理学会論文誌コンピューティングシステム, Vol. 45, No. SIG11, pp. 80–93 (2004).

[3] 佐藤幸紀, 鈴木健一, 中村維男：プログラムにおける命令の並列性と逐次性について, 情報処理学会研究会報告 2004-ARC-157, pp. 121–132 (2004).

[4] Kessler, R. E.: The Alpha 21264 Microprocessor, *IEEE Micro*, Vol. 19, No. 2, pp. 24–36 (1999).

[5] Desikan, R., Burger, D. and Keckler, S. W.: Measuring Experimental Error in Microprocessor Simulation, in *Proceedings of the 28th annual international symposium on Computer architecture*, pp. 266–277 (2001).

[6] Farkas, K., Jouppi, N. and Chow, P.: Register file design considerations in dynamically scheduled processors, in *High-Performance Computer Architecture*, pp. 40–51 (1995).

[7] Monreal, T., Gonzalez, A., Valero, M., Gonzalez, J. and Vinals, V.: Delaying physical register allocation through virtual-physical registers, in *Proceedings of the 32nd annual international symposium on Microarchitecture*, pp. 186–192 (1999).