面積効率を指向するプロセッサの設計と実装

堀尾 一生† 塩谷 亮太† 五島 正裕† 坂井 修一†
†東京大学大学院 情報理工学系研究科

1 はじめに

スーパスカラ・プロセッサの性能——IPCを向上させる一次的な方法は，そのウェイ数を増やすことである．しかし，IPCはウェイ数に比例して増加する訳ではない．その一方で，標準的なスーパスカラ・プロセッサのデザインでは，各部の回路規模はウェイ数のほぼ3乗に比例して増大してしまう．これは各部の中心となるハードウェアが多ポートのRAMで構成されており，RAMの回路面積はポート数の2乗に比例するからである．そのため，ウェイ数を無闇に増加させると，回路面積は現実的な線を超えてしまう．したがって最近では，各コアのウェイ数は2程度に抑え，コアを多数並べることによって性能向上を目指すことが現実的な解だと認識されている．

これに対し本研究室では，回路規模がウェイ数の3乗には比例しないような技術をいくつも提案してきた．発行幅を増やさずに命令のスループットを増やせるツインテール・アーキテクチャ，スケジューリング・ロジックに対してはマトリックス・スケジューラ[2]，リネーミング・ロジックに対してはリネームド・トレース・キャッシュ[3]，レジスタ・ファイルに対しては非レイテンシ指向レジスタ・キャッシュ・システム[4]を提案した．

これらの技術はスーパスカラ・プロセッサの各部に対応している．これらを一つのプロセッサの中に全て組み合わせることができれば，比較的ウェイ数の大きなスーパスカラ・プロセッサを現実的な面積で実現することが可能になる．しかしこれらの技術はそれぞれ異なるスーパスカラ・プロセッサの構成方式を想定しているため，単純に組み合わせようとすると矛盾が生じる．

本研究が提案するのは，これらの技術を一つに組み合わせられる特殊な構成のスーパスカラ・プロセッサである．

2 要素技術

リネームド・トレース・キャッシュ　スーパスカラ・プロセッサでは同じPCの命令であっても，それが新たにフェッチされる度にリネームを行う必要がある．

リネームド・トレース・キャッシュ(以下，RTC)は特殊なリネームを行うことでリネーム結果を再利用可能にする．そしてリネーム済みの命令をトレース・キャッシュに保存することで，同じ命令に対して毎回リネームを行う必要がなくなる．

1度命令をリネームしてしまえば以降リネームする必要がないので，リネームのスループットを落としても全体のパフォーマンスに与える影響はほとんどなくなる．そのためRMTの規模を削減することができる．

再利用可能なリネーム結果とは，コンシューマとプロデューサ間の距離(命令数)の情報である．コンシューマとプロデューサ間の距離はプロセッサの状態によって変化しないので，物理レジスタ番号と違って再利用可能である．

マトリックス・スケジューラ　ウェイク・アップは命令ウィンドウに格納された命令の中から，プロデューサに依存する命令を探し出す操作である．通常これはプロデューサの実行結果のタグと，全ての命令のソース・オペランドのタグを比較することによって実現される．

マトリックス・スケジューラ(以下，MXS)はコンシューマを探し出す操作を，シンプルな依存行列の読み出しによって実現する．依存行列はその列がプロデューサの，行がコンシューマの，それぞれ格納されたエントリに対応する．行と列の交差点のビットを立てることで依存関係を表現する．

依存行列を用いれば，プロデューサは自分の列を読み出すだけで，コンシューマの場所が分かるようになる．これによりタグの比較は必要なくなり，ウェイク・アップ・ロジックの回路面積が削減される．

非レイテンシ指向レジスタ・キャッシュ　従来のレジスタ・キャッシュ・システムは，レジスタ・キャッシュのヒットを仮定したパイプライン構成をとることで，アクセス・レイテンシを短縮していた．しかしこれはミス時にパイプラインを乱し，性能低下を招く．アクセス・レイテンシの短縮が性能に影響するのは分岐予測ミス時だが，レジスタ・キャッシュ・ミスの方が頻度がずっと高いため，トータルの性能は低下してしまう．

非レイテンシ指向レジスタ・キャッシュ(以下，NORCS)はレジスタ・キャッシュのミスを仮定したパイプライン構成をとる．すなわちレジスタ・キャッシュは初めからヒットしないものとして，後続の命令を発行する．アクセス・レイテンシは短縮されないが，ミスによってパイプラインが乱れることもない．従ってNORCSの導入は性能にほとんど影響を与えない．

レジスタ・キャッシュはレジスタ・ファイルへのアクセスのフィルタとして働くため，レジスタ・ファイルのポー

Design and Implementation of Area-efficient Processor
†Kazuo Horio †Ryota Shioya †Masahiro Goshima †Shuichi Sakai
†Graduate School of Information Science and Technology, The University of Tokyo

ト数は大きく削減できる．さらにバイパスの必要な期間を短縮してバイパス・ネットワークの簡単化を達成できる．NORCSは性能に悪影響を与えずにこの恩恵に預かることができる．

ツインテール・アーキテクチャ　スーパスカラ・プロセッサの命令ウィンドウの回路面積は，その発行幅の3乗に比例する．演算器そのものは小さいので多数並べることができるが，発行幅を増やすことは難しい．命令の実行のスループットは発行幅によって制限されてしまう．

ツインテール・アーキテクチャ(以下，TTA)では命令ウィンドウと並列に，演算器を行列状に配置する．この演算器には命令を，命令ウィンドウのようなバッファを介さず直接投入する．これによりバッファのポート数が実行のスループットを制限することがない．従って発行幅に制限されずに，命令の実行のスループットを向上させられる．

命令ウィンドウの外で命令を実行するためには，命令ウィンドウの外でソース・オペランドを得る必要がある．このためTTAはフロントエンドでレジスタ読み出しを行うスーパスカラ・プロセッサの構成を前提としている．

3　組み合わせるための構成

スーパスカラ・プロセッサの基本的な構成方式は，フロントエンドでレジスタ・ファイルを読み出す方式と，バックエンドでレジスタ・ファイルを読み出す方式の2種類に大別できる．どちらの方式もそのままでは全ての技術を組み込むことはできない．

フロントエンド方式　フロントエンド方式はフロントエンドでレジスタ・ファイルから読み出したレジスタ値を，バックエンドで実行に用いる．命令はバックエンドに送られた後，スケジューリングを経て実行されるが，この間バックエンドでレジスタ値を保持するバッファ，すなわちリザベーション・ステーション(以下，RS)が必要である

RSはレジスタ・ファイル同様，読み出されて命令にソース・オペランドを供給し，さらにその実行結果を書き込まれる．これらの機能を実現するのに，レジスタ・ファイルと同じ数のポートが必要である．だがRSの場合，これに加えてフロントエンドから送られてくるレジスタ値を受け取らなくてはいけない．そのためにレジスタ・ファイルより多くの書き込みポートを必要とする．

我々はレジスタ・ファイルの面積削減に対して有効な，NORCSという技術を持っている．しかしNORCSは書き込みポートの数を減らすことはできないので，RSに対してはレジスタ・ファイルほど有効ではない．

バックエンド方式　TTAを適用するためにはフロントエンドにオペランドを供給する必要がある．フロントエンド方式はこれを自然に達成できているが，バックエンド方式にはそのような機能は元々備わっていない．

TTAを適用するためだけにフロントエンドにレジスタ・ファイルを配置するのではハードウェア量の収支が合わない．しかしレジスタ・ファイルをそのまま配置する必要はない．

TTAは命令ウィンドウの内と外に2つの実行系を持つ．命令ウィンドウの外で実行できなかった命令を，命令ウィンドウの内で実行する仕組みになっている．これにより，命令ウィンドウの外では全ての命令を実行できる必要はない．

全ての命令を実行できる必要がないので，全てのオペランドを供給する必要がない．最も参照される可能性が高いと思われる一部のオペランドに絞り込んで供給するようにすれば，ハードウェア・コストを削減できる．すなわちレジスタ・ファイルではなくレジスタ・キャッシュを用いることで，低コストでバックエンド方式にTTAを導入できる．

従ってバックエンド方式をベースに，フロントエンドにレジスタ・キャッシュを配置した構成をとることで，全ての技術を組み合わせることが可能となる．

4　まとめ

本研究は面積効率の高いスーパスカラ・プロセッサの設計と実装を行うものである．スーパスカラ・プロセッサの各部の面積効率を高めるアーキテクチャ技術は既に提案されているが，これらはそれぞれ異なるスーパスカラ・プロセッサの構成方式を想定しているため，一つのプロセッサの中に組み合わせることは困難である．

本研究ではこれらの技術を組み合わせるためのスーパスカラ・プロセッサの構成方式の提案を行った．現在，実際にこの構成方式に基づくスーパスカラ・プロセッサのFPGA上への実装を進めている．実際にチップの形にすることを最終目標とし，そのファースト・ステップとしてプロトタイプをFPGA上に実装する．

参考文献

[1] 堀尾一生，平井遥，五島正裕，坂井修一: ツインテール・アーキテクチャ，先進的計算基盤システムシンポジウム *SACSIS2007*, pp.303-311, May, 2007

[2] M. Goshima, K. Nishino, Y. Nakashima, S. Mori, T. Kitamura, and S. Tomita: *A High-Speed Instruction Scheduling Scheme for Superscalar Processors*, *MICRO-34*, pp. 225-236 (2001)

[3] 一林 宏憲, 塩谷 亮太, 入江 英嗣, 五島 正裕, 坂井 修一: 逆 *Dualflow* アーキテクチャ，先進的計算基盤システムシンポジウム *SACSIS2008*, pp.245-254

[4] 塩谷 亮太, 入江 英嗣, 五島 正裕, 坂井 修一: 回路面積指向レジスタ・キャッシュ，先進的計算基盤システムシンポジウム *SACSIS2008*, pp.229-236