ディスパッチト・イメージ・キャッシュ

伊達 三雄[†]　倉田 成己[††]　伊藤 悠二[††]　塩谷 亮太[††]　五島 正裕[††]
坂井 修一[††]

[†] 東京大学 工学部
[††] 東京大学大学院 情報理工学系研究科

1　はじめに

一般にスーパースカラ・プロセッサにおいてはウェイ数を増加させることで性能の上げることができる．ウェイ数とは各構成ユニットで同時に処理することのできる命令数を指す．しかし一般的なスーパースカラ・プロセッサの制御部 (非演算器) の面積は，ウェイ数に対して２乗から３乗に比例する大きさで増加する [2]．これは，スーパースカラ・プロセッサの制御部の大部分が RAM や CAM で構成されており，それらの面積がポート数の２乗に比例するためである．ウェイ数を増やすと各 RAM・CAM のポート数も比例して増加し，回路面積が２乗に比例して増大する．単純にウェイ数を増加させると性能以上に回路面積が増大してしまう．

これに対し本研究室では，回路規模を最小限に保ちながら性能を引き出す技術を提案してきた．

本研究では，このようなスーパースカラ・プロセッサの制御部の中で**ディスパッチ**のロジックに着目する．ディスパッチとは，各命令を，対応する命令ウィンドウに格納することである．一般的なスーパースカラ・プロセッサではこの命令ウィンドウを整数，ロード/ストア，浮動小数点といった命令の系統ごとに分割し, スケジューリングを行う．

同時にディスパッチする命令数 (**ディスパッチ幅**) が増加すると，各命令を対応する命令ウィンドウに配分するための配線網は複雑化する．同時に RAM で構成される命令ウィンドウの書き込みポートも増加する．このためディスパッチの配線面積もディスパッチ幅の２乗に比例増加することとなる．

この複雑なロジックを省略する手法として**ディスパッチト・イメージ・キャッシュ** を提案する．ディスパッチト・イメージ・キャッシュでは命令間の依存関係を解析する**レジスタ・リネーミング**を終え，ディスパッチされた後の命令列を，キャッシュに格納する．

レジスタ・リネーミング済みの命令をキャッシュし，再利用する手法として**リネームド・トレース・キャッシュ**が提案されている [1].

リネームド・トレース・キャッシュでは後続の命令が，どの命令の実行結果をソース・オペランドとして使用するかを指定する形式に命令を変換する．何命令前の実行結果に依存しているか，という相対的な差分情報を得ることによって依存を解析する．毎回異なる物理レジスタ値を割り当てる通常のレジスタ・リネーミングと異なり，この手法では変換情報を再利用できる．

ディスパッチト・イメージ・キャッシュでも，リネームド・トレース・キャッシュと同様の形式でリネーミングを行うことによって，依存解析結果の再利用を可能とする．さらに，ディスパッチ済みの命令列をキャッシュすることで，命令ウィンドウの割り当て情報も再利用する．キャッシュ・ヒット時にはディスパッチト・イメージ・キャッシュからフェッチすると同時に命令ウィンドウに格納できる．ミス時にのみ少数の命令を，レジスタ・リネーミング及びディスパッチ行う．これにより，制御部の回路面積を最小限に保ちながら，ウェイ数が大きいときの性能を得られる．

2　リネームド・トレース・キャッシュ

リネームド・トレース・キャッシュの基本的な考え方は次の通りである:

- 命令の実行結果を格納するサイクリックなバッファである**物理レジスタ・ファイル**を用意し，in-order にエントリを割り当て，out-of-order で実行結果を格納する
- 物理レジスタ・ファイルとは別に，論理レジスタの値を保持する**論理レジスタ・ファイル**を用意し，命令の実行結果を in-order に格納する

Dispatched Image Cache
Date Mitsuo[†], Naruki Kurata[††], Yuji Ito[††], Ryota Shioya[††], Masahiro Goshima[††], Syuichi Sakai[††]
[†]Faculty of Engineering, The University of Tokyo
[††]Graduate School of Information Science and Technology, The University of Tokyo

- 初回の実行時に命令を変換し，ソース・オペランドを「n 命令前」の実行結果として物理レジスタ・ファイル上での変位で指定するようにする．リタイア済みであり実行結果が論理レジスタ・ファイルに格納されていることが保証できる命令の実行結果を使用する場合には，ソース・オペランドを論理レジスタ番号で指定する
- 変換した命令をトレース・キャッシュにキャッシュし，再利用する

キャッシュミス時のみ変換を行えば良いので，一度に変換できる命令の数は少なくとも性能に与える影響は小さい．レジスタ・リネーミングを行う際に用いる RMT (Register Map Table) を RAM で構成する場合，1 命令に対して 4 ポート必要である (ソース・オペランド 2 つ，デスティネーション・オペランド 1 つのとき)．従ってリネームド・トレース・キャッシュによって RMT の面積を大幅に縮小できる．

同一の命令に対して「n 命令前」が常に同じ命令ではないことに注意する必要がある．同一の命令列でも，その命令までの実行履歴が異なれば，違うトレースとしてキャッシュする必要がある．

3　ディスパッチト・イメージ・キャッシュ

ディスパッチト・イメージ・キャッシュはリネームド・トレース・キャッシュと同様の形式に変換され，更にディスパッチされた命令列をキャッシュする．ディスパッチト・イメージ・キャッシュは各命令ウィンドウに対応するように，キャッシュのバンク分けを行い，それぞれのバンクには対応する種類の命令のみを格納するようにする．

このようなバンク分けを行うことでディスパッチ情報の再利用が可能である．キャッシュラインの形成方法には自由度があるが，今回は**図 1** のような手法を実装した．要点は以下の通りである．

- 各バンク 1 ライン 2 命令まで格納できるようにする
- ラインをまたいで命令を追い越さない．すなわち，1 つのバンクが埋まれば，同一ラインの他のバンクが空いていてもラインを形成する
- 分岐命令の次の命令は，次のラインに格納する

図 1 の左側の命令列が上から順に出現したときに，i7 は i2 の隣などに詰めたりせず，i9 は i8 の隣に詰めないという意味である．

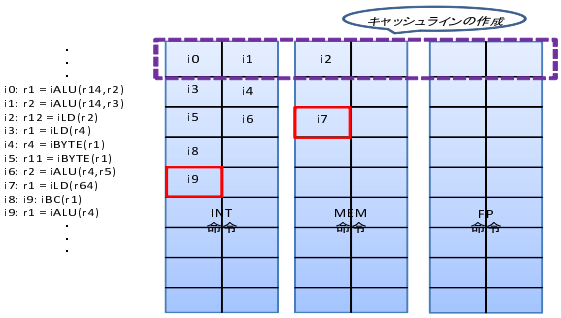

図 1: ディスパッチト・イメージ・キャッシュの格納法

ロジックを簡略にし，リネーミング情報の再利用を簡単に行うためにこのような構成にした．命令の出現には偏りがあるため，同一種類の命令が連続することも少なくない．このモデルでフェッチできる平均の命令数は整数演算系のベンチマークで 2.4 程度である．

ディスパッチ・ロジックは，キャッシュミス時に流れる少数の命令に対してのみ行えばよいので，その面積を省略することができる．ヒット時には，フェッチと同時にディスパッチが完了しているため，パイプラインの段数もその分短くなる．

4　まとめ

本研究は面積効率の高いスーパースカラ・プロセッサの実現手段として，ディスパッチト・イメージ・キャッシュを提案した．今後は実装と評価を進める．また，よりキャッシュ利用効率の高い実現手法を求める予定である．

参考文献

[1] 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

[2] 五島正裕: *Out-of-order ILP* プロセッサにおける命令スケジューリングの高速化の研究, 京都大学 *(*博士論文*)*, March, 2004