命令ウィンドウの動的最適化

吉瀬 謙二，中村 友洋，辻 秀典，安島 雄一郎，高峰信，坂井修一，田中 英彦
東京大学大学院工学系研究科

1　はじめに

プロセッサが抽出できる並列性は、データ依存関係が解消され機能ユニットに発行できる命令を解析する能力(先見能力) に依存する。さらにプロセッサの先見能力を決めるパラメタは命令ウィンドウとリオーダ・バッファのエントリ数である。

本稿では、フェッチ機構、オペランド供給、結果バスといったハードウェア資源を理想化したプロセッサ・モデルを定義する。その上で先見能力の制約を変化させ、得られる並列性について検討する。

2　命令ウィンドウ

命令をアウト・オブ・オーダに発行するためには、デコード・ステージと実行ステージを分離し、両者の間で実行可能な命令を動的に抽出し、実行待ちの命令をバッファする機構が必要である。この目的で設けられるバッファを命令ウィンドウと呼ぶ。フェッチ、ディスパッチ、発行、実行という処理の流れを図 1に示した。命令ウィンドウから機能ユニットに命令を発行するためには当該命令で必要となるオペランドが利用可能かつ、必要なハードウェア資源が利用可能となっている必要がある。

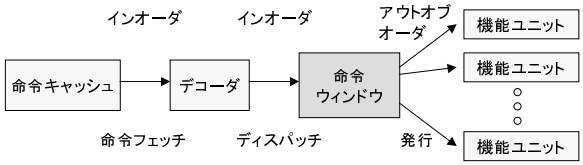

図 1: アウト・オブ・オーダ発行のためのウィンドウ

3　命令ウィンドウとプロセッサ・モデル

命令ウィンドウを含むプロセッサ・モデルについてはさまざまな方式が提案されているが、本稿では図 2 に示したモデルを検討する。

多くのスーパースカラ・プロセッサではそれぞれの機能ユニット毎に分割されたウィンドウ（リザベーション・ステーション）を用いている。別の選択は集中ウィンドウを用いる方法で、ロジックが複雑になるという欠点があるものの、エントリ数を有効に利用できるという利点がある。この利点より、本モデルでは集中ウィンドウを採用する。

幾つかのモデルにおいては命令ウィンドウにイン・オーダの情報も格納する場合がある。これらのモデルは、ハードウェア量の節約という点から重要であるが、ここでは、命令を発行するためのバッファとしての命令ウィンドウとイン・オーダの情報を保つためのバッファ(リオーダ・バッファ) を分けて議論を進める。これより集中ウィンドウに格納されていた命令が発行された時点で当該エントリを開放し、それらのエントリには任意の命令を格納することができるようになる。

プロセッサの正確な割込みを実現するために用いるリオーダ・バッファにはディスパッチされてからリタイアされるまでの全命令に対して、イン・オーダにエントリが割り付けられる。これよりイン・オーダなレジスタ・ファイルへの書き込み (リタイア) を実現できる。

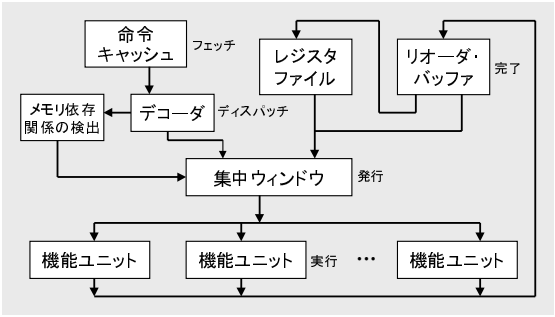

図 2: 評価に用いたプロセッサ・モデル

正確な割込みをサポートする場合、命令ウィンドウのエントリ数だけでなく、リオーダ・バッファのエントリ数が先見能力に影響を与える。図 3 はこの様子を説明するものである。アウト・オブ・オーダに発行された命令は、命令ウィンドウのエントリから削除されるため、命令ウィンドウのエントリには連続した命令が格納されているとは限らない。また、命令ウィンドウのエントリに空きがあったとしても、リオーダ・バッファのエントリに空きがなければディスパッチがストールしてしまう。図 3から明らかなように、リオーダ・バッファのエントリ数以上の命令ウィンドウのエントリ数は必要ない。

ロード命令に関しては、リオーダ・バッファ内のストア命令との依存関係を検出できると想定する。これより、ロード命令は依存関係のないストア命令を追い越して発

Dynamic Optimization of Instruction Window
Kenji KISE, Tomohiro NAKAMURA, Hidenori TSUJI, Yuichiro AJIMA, Makoto TAKAMINE, Shuichi SAKAI, Hidehiko TANAKA
Graduate school of Engineering, The University of Tokyo

行できるようになる。

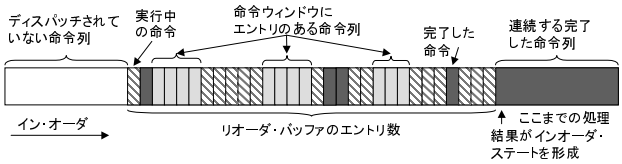

図 3: あるプロセッサ状態における先見能力

4 評価環境

3章で定義したプロセッサ・モデルに従い、命令ウィンドウとリオーダ・バッファのエントリ数を変化させた場合の IPC (1 サイクルにリタイアした平均命令数) を測定する。ディスパッチ、発行、リタイアの幅はそれぞれ 10 命令とする。ロード・ユニット 2 つ、ストア・ユニット 2 つ、分岐ユニット 2 つ、それ以外の命令を処理する ALU を 6 つという構成 (IPC10 を可能とする構成) とする。フォワーディング・パスと結果バスの数は制限していない。理想的な命令フェッチ機構を仮定した。2 ポートのミスをおこさないデータ・キャッシュを仮定する。

アーキテクチャは SPARC architecture version 8 を用いる。リオーダ・バッファ、命令ウィンドウのエントリ数ともに 32 から 256 まで変化させる。解析時間の都合上、SPEC95 から cc1 に関する 4700 万命令の実行トレースを解析する。

5 評価結果と検討

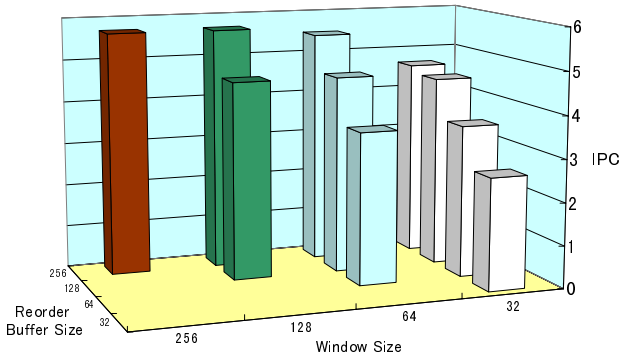

図 4: エントリ数を変化させた場合の IPC (cc1)

cc1 の評価結果を図 4 に示した。右下の共に 32 の設定が標準的なプロセッサの先見能力を想定しており、この時の IPC は 2.6 である。一方、先見能力を最も高く設定した (図の左上で、エントリ数を共に 256 とした) 場合の IPC は 5.7 という結果を得た。先見能力の向上により 2.2 倍の IPC が利用可能となることがわかる。

次に、図 4からエントリ数と IPC の関係を検討する。ウィンドウのエントリ数 32 の時には、リオーダ・バッファのエントリ数が 128 を越えた所で IPC が飽和していることがわかる。一方、ウィンドウのエントリ数 64 以上の時にはリオーダ・バッファのエントリ数を 256 まで増加することに意味がある。ただし、リオーダ・バッファのエントリ数 256 を想定した場合には、命令ウィンドウのエントリ数が 64 を越えたところで IPC が飽和する。以上の検討より、命令ウィンドウとリオーダ・バッファのエントリ数を 32, 128 または 64, 256 とした場合が IPC を飽和させないためバッファ構成であり、命令ウィンドウのエントリ数の 4 倍がリオーダ・バッファのエントリ数の目安となることがわかった。

時間の都合上、cc1 の解析結果しか検討することができなかっが、現在いくつかのプログラムを解析中である。

6 関連研究

命令ウィンドウのエントリ数を変化させた時の性能は文献 [1] に詳しいが、(1) 分岐ユニットとロード/ストア・ユニットがそれぞれ一つしか存在しない。(2) 分岐命令、ロード/ストア命令はインオーダで発行する。という制約により速度向上比はそれほど得られていない。より多くの IPC をターゲットとした場合には、先の制約は厳しすぎる。汎用アプリケーションにおける分岐命令の実行頻度は 20% 以上となることが多く、1 サイクルに 1 つ以上の分岐命令を処理する必要がある。また、ロード・ストア命令をインオーダに処理するという制約が並列性抽出のネックとなる可能性がある。

7 まとめと今後の課題

命令ウィンドウ、リオーダ・バッファと先見能力の関係をまとめ、それらのエントリ数を変化させた場合の並列性 (IPC) を検討した。

評価において、データ・フォワーディングのパスや、結果バス、レジスタ・ファイル等のポート数は制限していない。本稿で示した IPC を達成するプロセッサを設計しようとした場合には、文献 [3] で指摘されているようにデータ・フォワーディングのパスが複雑になるなど、多くの問題点が発生する。本稿で示した IPC を実現するためのアーキテクチャの提案は今後の課題である。

本稿では考慮しなかったが、文献 [2] で提案されているように参照データの値予測をおこなえば真のデータ依存関係さえ解消できる場合がある。本評価で得た並列性には、まだ改善できる余地が残っている。

参考文献

[1] マイク・ジョンソン. スーパースカラ・プロセッサ. 日経 BP センター.

[2] Mikko H. Lipasti, Christopher B. Wilkerson, and John Paul Shen. Value Locality and Load Value Prediction. ASPLOS VII, pages 138–147, 1996.

[3] Subbarao Palacharla, Norman P. Jouppi, and J. E. Smith. Complexity-Effective Superscalar Processors. 24th Annual Int. Symp. on Computer Architecture, pages 206–218, 1997.