スーパースケーラのための高速な動的命令スケジューリング方式の改良

西野賢悟[†]　五島正裕[†]　中島康彦[†]
森眞一郎[†]　北村俊明[†]　富田眞治[†]

スーパースケーラは，動的スケジューリングのため，命令の実行に必要なデータの有効性を追跡する *wakeup* と呼ぶロジックを持つ．従来の *wakeup* は CAM によって実装されるため，LSI の微細化，パイプラインの深化に伴ってクリティカルになっていく．そこで我々は，小容量の RAM の読み出しによって *wakeup* を実現する方式を示した．ただし以前に示した方法では，この RAM の更新が read-modify-write となっていた．本稿では，この read-modify-write を避ける方法について述べる．

Improvement of a high-speed dynamic instruction scheduling scheme for superscalars

Kengo Nishino,[†] Msasahiro Goshima,[†] Yasuhiko Nakashima,[†]
Shin-ichiro Mori,[†] Toshiaki Kitamura[†] and Shinji Tomita[†]

A superscalar has *wakeup* logic which manages availability of the data for dynamic instruction scheduling. Since usual *wakeup* are implemented by a CAM, the delay time of it will be critical with smaller feature sizes and deeper pipelines. We proposed a new scheme which realizes *wakeup* by reading a small RAM. The scheme, however, needs read-modify-write to update the RAM. In this paper, we describe an improvement to avoid the read-modify-write.

1. はじめに

スーパースケーラの構成要素のうち，***wakeup*** と呼ぶ処理の遅延が，将来クロック速度を制限するものの 1 つになると予測されている．*wakeup* は，動的命令スケジューリングのために，命令の発行に必要なデータの有効性を追跡する処理である．

従来の *wakeup* は，データに割り当てられたタグによる連想処理に基づくもので，RAM を読み出した結果で CAM をアクセスするという構造を持つ．これらのメモリは，配線遅延に支配されるため，LSI の微細化の恩恵を受けにくい．また *wakeup* は，他の多くの構成要素とは異なり，複数のステージに分割することができない．そのため *wakeup* の遅延は，LSI の微細化，パイプラインの深化にともなっていっそうクリティカルになっていくと予測されている．

このような背景から我々は，*wakeup* を高速化する新しいスケジューリングの方式を提案した[1]．この方式は，タグによる連想処理ではなく，命令間の依存関係を表す行列によって *wakeup* の処理を行う．行列は，**依存行列テーブル** (dependence matrix table:**DMT**) と呼ぶ小容量の RAM によって実装される．DMT の更新はフロントエンドにおいて実行しておくことができ，*wakeup* は DMT の読み出しによって実現できる．

しかし，文献 1) に示した方式では，DMT の更新処理が，原理的には，read-modify-write（以下**RMW**）となっていた．そこで本稿では，この RMW を回避する方法を示す．以下，まず 2 章では 1) の方式についてまとめ，3 章でその改良の方法について述べる．

† 京都大学　Kyoto University

2. DMT

図 1 に，DMT の概念図を示す．DMT は，基本的には，左/右のソース・オペランドごとに用意される依存行列である．エントリ ID $= p$ の命令 I_p が生産するデータを， ID $= c$ の命令 I_c がその左（右）オペランドとして消費する場合，左（右）オペランド用の行列の p 行 c 列の要素は “1”，それ以外は “0” とする．図 1 に示す例では，連続する 4 つの命令が ID = 1，2，3，4 のエントリに順に格納された場合を表している．この場合，例えば，ID = 1 の命令が生産する r1 を ID = 2 の命令が左オペランドとして消費するから，左の行列の 1 行 2 列の要素は “1” となる．その他の要素も同様に求められる．

wakeup は，具体的には，左/右オペランドが利用可能かどうかを表すフラグ rdyL/R を更新する処理である．提案方式では，実行される命令に対応する複数行の bitwise-OR を求めれば，定義により，セットすべき rdyL/R を表す行ベクトルを求めることができる．

DMT を実装するには，図 1 に示した行列をそのまま 1-read の RAM とすればよい．DMT では，複数行の bitwise-OR を求める必要があるが，そのための特別なロジックを用意する必要はない．DMT では，通常の RAM と異なり，同時に実行される命令に対応して，同時に複数のワード線がアサートされることになる．RAM のリードポートは，もともと各行に対する wired-OR として実現されているため，同時に複数のワード線がアサートすることによって，対応する複数行の bitwise-OR を読み出すことができるのである．

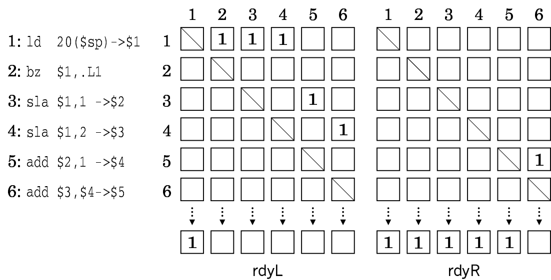

図 1 依存行列テーブル (dependence matrix table:DMT)

DMT の更新

DMT の更新は，以下のように行われる．今デコードされている I_c が ID $= c$ のエントリに格納されるとしよう．DMT の更新処理は，(1) I_c の依存元の命令 I_p が格納されているエントリの ID $= p$ を求める，(2) DMT の p 行 c 列を "1" にセットする の 2 つフェーズからなる．

レジスタ・リネーミングでは，I_p に割り当てられたタグを求める処理が行われる．(1) I_p が格納されているエントリの ID $= p$ は，それと全く同じ方法，同じタイミングで求めることができる．

(2) では，DMT に，p をアドレス，c をデコードしたものを内容として与えることになる．このときに，c をデコードしたものを上書きすることはできない．更新の対象となる p 行には，一般に，すでに "1" となっているビットがある．したがって，通常の RAM では，エントリ p を読み出し，c 列のビットをセットし，その結果を書き戻す，すなわち，RMW が必要となる．

文献 1) では，実際に RMW が行われるを避けるため，通常の RAM では相補的に用意される書き込みポートの一方を敢えて省略する方法を示した．この回路では，RMW が避けられる代わりに，各行は使用に先だってリセットする必要がある．リセットのため，パイプラインの構成によっては，命令ウィンドウのエントリが解放された直後のサイクルには当該エントリを使用することができなくなる．ただし，このことよる IPC の低下は，MIPS R10000 の構成においてSPECint95 のプログラムを実行した場合，平均で 0.37%，最悪で 1.12%に過ぎない．

3. DMT の改良

RMW が必要となるのは，今デコードされている I_c が格納されるのは ID $= c$ のエントリであるのに，c 行ではなく，p 行が更新されるためである．そのため p 行は，互いに異なる複数の命令が格納される際に更新の対象となり得る．

この状況は，図 **2** に示すように，図 1 に示した DMT を，単に転置することで回避することができる．転置した DMT では，各行は，I_p ではなく，I_c に対応する．なお，I_c の左/右のソースに対してその I_p はただ 1 つであるから，c 行では p 列だけが "1" となる．

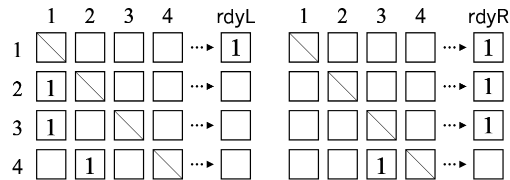

図 2 転置 DMT

DMT の更新では，(1) の p を求める部分は変わらない．一方 (2) では，転置前の DMT では p 行 c 列をセットしたが，転置後では，原理的には，c 行 p 列をセットすることになる．

転置後では，I_c を格納する際に更新の対象となるのは，p 行から c 行に変わる．格納される命令と更新される行が 1 対 1 に対応するため，各行が再び更新の対象となることはない．したがって，c 行には p をデコードしたものを単に上書きすればよく，RMW を回避することができる．

なお，DMT の読み出し，すなわち，*wakeup* では，行ベクトルではなく，列ベクトルを読み出すことになる．したがって転置後の DMT では，RAM としての行と列，すなわち，ワードとビットの関係が，書き込みと読み出しで入れ替わる．

4. おわりに

本稿では，DMT の更新における RMW を回避する方法について述べた．ただしそれによる IPC の向上は，2 章で述べたように，1%程度に過ぎず，ハードウェアも目立って簡単化される訳でもない．しかし，単に転置することで RMW が避けられるという事実には興味深いものがある．

もともと DMT は，データフロー計算機のように，生産側の命令が消費側の命令を直接指定すれば，連想検索を避けられるという発想から生まれたものである．そのため転置前の DMT では，生産者を行に対応させたのである．しかし，転置した DMT ではむしろ，消費側が生産側を直接指定しているように見える．

転置後の DMT の方がすっきりとしているのは，1 個の命令の結果を複数の命令が消費するという計算の自然な性質を反映しているためといえるかも知れない．

謝辞

本研究の一部は文部省科学研究費補助金，基盤研究 (B)(2) #12480072，同#12558027 による．

参考文献

1) 五島正裕, 西野賢悟他: スーパースケーラのための高速な動的命令スケジューリング方式, 情処研報 2001–ARC–142 (HOKKE-2001) (2001).