計算機アーキテクチャ 142−21
ハイパフォーマンス 85−21
コンピューティング
（2001. 3. 9）

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

五島正裕[†] 西野賢悟[†] グェンハイハー[†]
縣亮慶[†] 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，動的命令スケジューリングのため，オペランドの有効性を追跡する *wakeup* と呼ぶロジックを持つ．本稿では，従来のタグに基づく連想処理ではなく，命令間の依存関係を直接的に表現するRAMを用読み出すことで *wakeup* を実現する方式と，その遅延をIPCに対するペナルティに転化する手法を示す．実在する.18μm CMOSプロセスのデザイン・ルールに基づいてこのロジックを設計し，回路の面積を求め，Hspiceによって遅延を測定した．また，シミュレーションによって，ペナルティを測定した．その結果，3%以下のペナルティを代償に，2GHzを越える最高動作周波数を達成できることが分かった．

A high-speed dynamic instruction scheduling scheme for superscalars

Masahiro Goshima,[†] Kengo Nishino,[†] Nguyen Hai Ha,[†]
Akiyoshi Agata,[†] 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. This paper describes a new scheduling scheme which substitutes association of tags by reading a small RAM which directly represents depndence between instructions, and changes the delay of the logic into IPC penalty. We actually designed the logic guided by a design rule of a real .18μm CMOS process, measured the areas, and calculated the delays by Hspcie. And we also evaluated the IPC penalty by simulation. The evaluation result shows that this scheme achieves over 2GHz clock speed with the IPC penalty less than 3%.

1. はじめに

現在のスーパースケーラでは，クロック速度が命令発行幅 (***IW***:Issue Width) とウィンドウ・サイズ (***WS***) を制限する主因となりつつある．スーパースケーラの構成要素のうち，***wakeup*** と呼ぶロジックが，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[1]．*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ロジックである．

従来の *wakeup* は，命令が使用する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，複数のパイプライン・ステージに分割することができない．以上の理由により *wakeup* は，LSIの微細化，パイプラインの深化にともなってクリティカルになっていくと予測されるのである．

このような背景から我々は，*wakeup* を高速化する全く新しい動的命令スケジューリング方式を提案するした[2]．本方式では，タグによる連想処理ではなく，命令間の依存関係を直接的に表現する行列を用いて *wakeup* を実現するため，小型のRAMを読み出す程度の遅延で *wakeup* を実行することができる．本稿では，この方式を更に高速化する手法と，それらの定量評価の結果を示す．

以下，まず2章では従来の動的命令スケジューリング方式の一般的な構成法を紹介し，3章で提案する方式について詳しく述べる．そして4章で，提案方式の定量的評価を行う．

† 京都大学 情報学研究科
Gradudate School of Informatics, Kyoto University
†† 京都大学 経済学研究科
Gradudate School of Economics, Kyoto University
††† 京都大学 総合情報メディアセンター
Center for Information & Multimedia Studies, Kyoto University

2. 従来の動的命令スケジューリング方式

スーパースケーラを構成するの基本構造のうち，演算器それ自体以外のほとんど全ての遅延は IW, WS の増加関数で与えられる．ただしそれらの遅延は，パイプライニングやクラスタリングなどの技術によって，1サイクルに終えなければならない処理の遅延を大幅に短縮できる．しかし動的命令スケジューリングを行うロジックに対しては，このような技術は効果的ではない．本章では，その理由について詳しく述べる．以下まず2.1節においてスーパースケーラの動的命令スケジューリングの原理についてまとめ，2.2節でスケジューリングの処理と命令パイプラインの関係について説明する．

2.1 従来の動的命令スケジューリング方式

Out-of-order スーパースケーラは，論理的なレジスタとは別に，各命令の実行結果を一時的に保存するバッファを用いる．このバッファの構成方式には，リオーダ・バッファを用いる方式と，物理レジスタを用いる方式がある．スーパースケーラにおける動的命令スケジューリングは，これらのバッファのエントリを用いて，局所的にデータ駆動型の計算を行うこととみなすことができる．命令ウィンドウ内で，命令 I_p が生産するデータを命令 I_c が消費する場合を考えよう．バッファのエントリを介して I_p から I_c にデータが渡されることに着目すると，スケジューリングの処理は以下のように説明できる：

(1) *rename* 命令がフェッチされると，論理レジスタ番号から**タグ**への変換が行われる．

I_p には，バッファの1エントリが割り当てられ，エントリはデータが『ない』状態に初期化される．このエントリのIDがタグである．I_p に割り当てられたタグを特にtagDということにする．

I_c は，左/右のソースの論理レジスタ番号から，依存する I_p に割り当てられたtagDを得る．これをtagL/Rということにする．I_c は，tagL/Rで示されるエントリにデータが書き込まれるのを待つ．

(2) *wakeup* I_p の実行にともなって，I_c が実行可能になることを検出する．

I_p が実行されると，その結果はtagDで示されるエントリに書き込まれ，エントリはデータが『ある』状態に遷移する．

I_c は，tagL/Rで示されるエントリにデータが『ある』のを見て，発行可能になる．

(3) *select* 発行可能な命令から，実際に発行するものを選択し，発行する．

2.2 命令スケジューリングのパイプライン化

次に，*rename*，*wakeup*，*select* の各処理をパイプライン化することを考えよう．命令パイプライン中のステージの違いから，*rename* と (*wakeup*+*select*) とに分けて考える必要がある．

rename は，必要ならば，パイプライン化することでクリティカル・パスから外すことができる．実際現存するスーパースケーラでは，*rename* の遅延のため，デコード・ステージに複数サイクルを充てることが普通である．ただしもちろん，その分だけ分岐予測ミス・ペナルティが増加することになる．

wakeup* と *select は，*rename* とは異なり，パイプライン化することができない．*wakeup* と *select* のそれぞれに1サイクルかけた場合，先行する命令の結果を消費する命令は，先行する命令に引き続くサイクルに実行することができない．このことは，レイテンシが1サイクルである演算器 ——— 通常の構成ではALU ——— からはオペランド・バイパスを行わないことと等価である．詳細は4章で述べるが，それによるIPCの悪化は30%程度にもなり，クロック速度の向上に見合わない可能性が高い．このような観点から，*wakeup* と *select* は，実際上，合わせて1サイクルで実行しなければならないとできる．

2.3 Superscalar の *wakeup*

wakeup は，実行される I_p のtagDをRAMから読み出し，それをキーとしてCAMにアクセスするという処理によって実現される．

前述したように，*wakeup* と *select* と合わせて1サイクルで実行する必要がある．これらのうち，*select* の遅延は専らゲート遅延からなるため，LSIの微細化に伴って順調に短縮されていくと予測される．一方，*wakeup* の遅延はRAMとCAMのワード線，ビット線などの配線遅延からなるため，LSIの微細化の恩恵を受けにくい．以上の理由により *wakeup* 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るのである[1]．

命令ウィンドウは，実際には，演算器（のクラス）ごとに設けられたリザベーション・ステーション，あるいは，命令キューによって実現されることが多い．例えばMIPS R10000では，ロード/ストア，整数演算，浮動小数点演算のそれぞれに16エントリの命令キューを用意している[3]．しかし複数のキューによって実現される場合にも，それらの基本的な構成は変わらず，命令キューの命令発行幅を $IW_q (< IW)$，サイズを $WS_q (< WS)$ とすると，各キューにおけるRAM，CAMのワード数は WS から WS_q に，また，RAMの読み出しポート数は IW から IW_q に，それぞれ縮小することができる．しかしCAMの比較入力ポート数は，IW のまま縮小することができない．

3. 提案する動的命令スケジューリング方式

提案するスケジューリング方式は，従来のタグによる連想処理に基づくものとは根本的に異なり，各命令間のデータ依存関係を直接的に表すデータ構造を用い

てスケジューリングを行う．本章では，提案方式について詳しく述べる．

3.1 DMT

提案方式では，命令ウィンドウ中の各命令間のデータ依存関係を直接的に表す**依存行列テーブル** (dependence matrix table:**DMT**) が，スケジューリングにおける中心的な役割を果たす．

3.1.1 DMT の概要

図1 に，DMT の概念図を示す．DMT は，基本的には，rdyL/R 用に各1つづつの WS 行 WS 列の RAM である（ただし，対角要素は使用しない）．それぞれの RAM の各行は I_p に，各列は I_c に対応する．各要素は，対応する I_p と I_c の間の依存関係を表す．すなわち，p 行 c 列の要素は，命令ウィンドウ内のエントリ ID=p の I_p の実行結果を ID=c の I_c が消費するなら "1"，そうでなければ "0" とする．図1の例は，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．図の例では，ID=1 の命令が生産する r1 を ID=2，3 の命令がそれぞれの左オペランドとして消費している．したがって DMT では，1行目の 2，3 列目が "1" となる．その他の要素も同様に求められる．

DMT の更新は，従来のスーパースケーラの *rename* ステージにおいて実行しておくことができる．本稿では提案手法のポイントとなる *wakeup* の処理について述べる．

wakeup ステージにおいては，実行される最大 IW 個の I_p に対応する IW 行の OR を求めれば，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．例えば，図1に示した状態でエントリ ID=1 の命令が発行された場合を考えよう．この命令の実行結果は，ID=2，3 の命令が左オペランドとして消費する．自明ではあるが，DMT の第1行は ID=1 の命令が実行される時にセットすべき rdyL/R を表している．実際に ID=1 の命令が実行されると，ID=2，3 の rdyL がセットされ，次のサイクルにはその2つの命令が実行可能になる．実際にその ID=2，3 の命令が選択され同時に実行される場合には，第2行と第3行の OR を求めればよい．結果，ID=4 の命令の rdyL/R がそれぞれによってセットされる．ただし実際には，OR

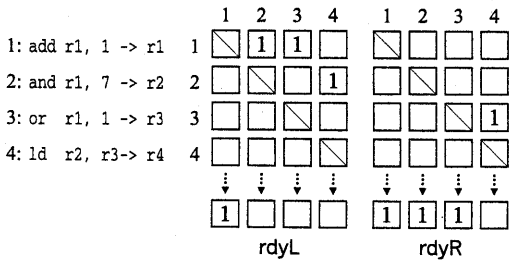

図1 依存行列テーブル (dependence matrix table:DMT)

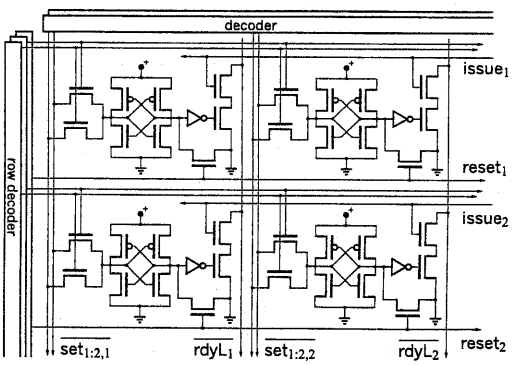

図2 依存行列テーブルのロジック（rdyL 用）

を求めると言っても，各列で "1" である要素ははたかだか1つである；なぜなら，1つのソース・オペランドを生産する I_p はただ1つだからである．

命令ウィンドウが q 本の命令キューによって実装される場合には，生産側と消費側に対応して，$q \times q$ 個の DMT を用意すればよい．それぞれの DMT は，WS b×WS word から，WS_q b×WS_q word に縮小される．

3.1.2 DMT の実装

DMT を実装するにあたっては，単に 1-read の RAM を用いればよい．複数行の OR を求めるための特別なロジックは必要ない．

図2 に，DMT の回路図を示す．同図には，左上の2行2列分のセルが示してある．

各セルの中央にある 4T セルの左側は，書き込みポートである．DMT の更新は，上書きではなく，ビット・セットを行う必要がある．そのため書き込みポートは，single-bitline となっている．bitline が low であればセットされるが，high であってもリセットされない．

その一方で DMT のエントリは，使用に先だってエントリごとリセットしておく必要が生じる．そのため，4T セルの右下にリセット用のポートが用意されている．このリセットのため，エントリの解放後再利用するためには，実装によっては1サイクルの空きが生じることがある．

wakeup に関連するのは，4T セルの右側にある読み出しポートである．図からも明らかなように，この読み出しポート部は，single-bitline の 1-read の RAM と構造上は全く変わらない．

ただし，通常の RAM では同時にはたかだか1つのワード線しかアサートされないのに対して，DMT では毎サイクル実行される I_p に対応する（最大）IW 本のワード線 issue が同時にアサートされる点が異なる．各ビット線 $\overline{\text{rdyL}}$ には，アサートされた issue に対応する（最大）IW 個のセルが接続され，いずれかのセルの出力が low であれば pull-down される．すなわち，単に複数のワード線を同時にアサートすることによって，対応する行の OR を読み出すことができる．

3.2 DMTの縮小

依存するI_pとI_cの間の距離は短い場合が多く，32命令以下の場合が90%程度以上を占めることが分かっている[4]．本節では，この性質を利用して，前節で述べた$wakeup$を更に高速化する手法について述べる．

前節で述べたDMTは$(WS-1)$ b×WS wordであったが，このビット数を$w(0 \leq w < WS-1)$ bに削減することによって$wakeup$の遅延を更に短縮することを考える．このwを**DMTの幅**と呼ぶことにする．依存する命令間の距離がDMTの幅以下の場合には，DMTによってrdyL/Rを更新する．幅を越えていた場合には，クロック速度に影響を及ぼさない別の方法を用い，IPCに対するペナルティに転化するのである．依存する命令間の距離は短い場合が多いので，ペナルティの影響は小さいと予想される．

命令間の距離がDMTの幅を越える場合への対応としては，以下の2つの方法が考えられる：

dealy 幅$w(w < WS-1)$と$WS-1$の2種のDMTを用意し，後者には1サイクル余分にかける．
距離が幅wを越えている場合にはrdyL/Rの更新が1サイクル遅れるため，I_cはI_pに引き続くサイクルには発行されない．

stall 依存するI_pまでの距離がDMTの幅を越えている場合には，I_cのデコードの段階でフロント・エンドをストールさせる．
I_pの実行が終了した後にデコードを再開すれば，rdyL/Rは1に初期化される．

DMTの縮小の方法

図3に，DMTの縮小の様子を示す．左は元々の，右が縮小後のDMTである．同図では，$WS=8$，$w=4$である．元々のDMTから削除されるセルは，左図中で薄く示した．必要なセルを（矩形領域に）集める方法には任意性があるが，よりクリティカルであるビット線の長さを短縮することを優先して，図3右のようにするとよいであろう．

図3右から明らかなように，縮小されたDMTのワード線，ビット線は，それぞれw個のセルにしか接続されておらず，それぞれの長さはWSとは無関係となっている．したがって$wakeup$の遅延は，WSとは独立に，wによって決まる．

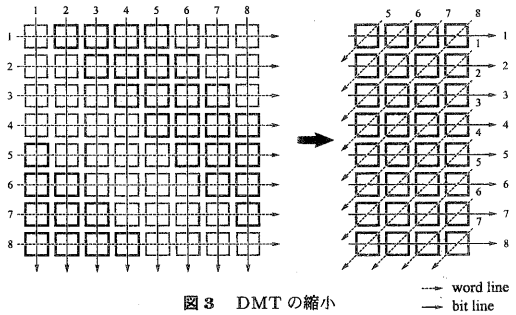

図3 DMTの縮小

ウィンドウ・エントリの使用に対する制限

上では，命令間の距離という言葉を曖昧に用いていたが，DMTによってrdyL/Rが更新できるのは，正確には，命令間の距離ではなく，ウィンドウ内のエントリ間の距離がDMTの幅以下の場合である．したがって，命令流上での距離とウィンドウ内での距離が（ある程度）一致するように制御する必要がある．

このことは特に，命令ウィンドウを複数の命令キューによって構成する場合に問題となる．エントリをサイクリックに使用する場合，異なる命令キュー間では，パディングによってエントリの消費の歩調を揃える必要がある．例えばR10000の命令キューの構成では，各キューのサイズは同じであるから，例えばあるサイクルに2つの整数命令が整数命令キューに格納されるとすると，他のキューでも1つのエントリを無駄に消費する必要がある．

4. 評　　価

従来方式と提案方式との定量的な比較を行う．比較項目は，3.2節で述べたDMTの縮小に対するペナルティと，回路の面積/遅延である．前者は4.1節で，後者は4.2節で，それぞれ述べる．

4.1 IPCの評価

測定条件

SimpleScalarツールセット (ver.2.0) に対して，3.2節で述べたDMTの縮小を実装し，SPECベンチマークを用いてDMTの幅に対するIPCの変化を測定した．

ベース・モデルとしては，MIPS R10000[3]のマシン構成を用いた．R10000は，ロード/ストア，整数演算，浮動小数点演算のそれぞれに命令キューを持ち，$(IW_q, IW, WS_q, TW) = (2, 4, 16, 6)$である．1次キャッシュは，命令/データ，それぞれ，容量32KB，ライン・サイズ32Bである．2次キャッシュは命令/データ共有で，容量1MB，ライン・サイズ64Bである．レイテンシは，1次は1サイクル，2次は6サイクルである．2次キャッシュ・ミス時は，最初のデータが得られるまでが18サイクルで，後続データへのアクセスには2サイクルが必要である．分岐予測には，ツールセットに用意されている2b飽和型カウンタによるもの (bimod) を用いた．これを，構成R10K×1と呼ぶことにする．また，キャッシュ－メモリ以外のすべての資源を2倍，すなわち，$(IW_q, IW, WS_q, TW) = (4, 8, 32, 7)$としたものも合わせて測定した．これを構成R10K×2と呼ぶ．

測定結果

まず，DMTの縮小の前に，前章で述べたDMTエントリのリセットとパディングの影響を調べたところ，それらによるIPCの低下率は平均で.59%，最悪でも

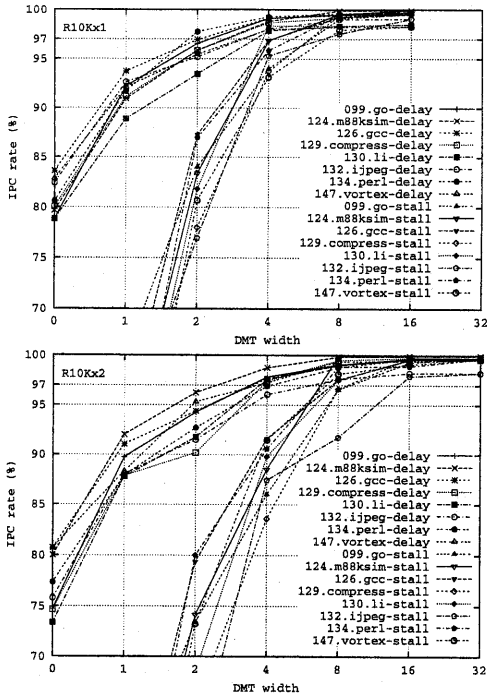

図4 DMT の幅に対する IPC

1.54%であった．後述する DMT の縮小による IPC の低下率は数%程度はあるから，リセットとパディングによる影響はそれに比べてほとんど無視できる．

次に，DMT の縮小を行った場合の，従来方式に対する IPC の低下率を**図4**に示す．なお提案方式には，リセットを組み込んである．また，DMT の幅が WS_q の場合でもパディングを行っている．

結果からは，DMT の幅は，delay ならば WS_q の 1/4，stall でも 1/2 あれば，従来方式に対する IPC の低下は 3%以下に抑えられることが分かる．

ちなみに，DMT の幅を 0 とした場合の delay の値は，ちょうど，2.2 節で述べた，*wakeup* と *select* にそれぞれ 1 サイクルかけ，ALU からはバイパスを行わない場合に相当する．その場合の IPC は低下は，30%程度になる．

4.2 回路の評価

富士通株式会社から提供された，.18μm CMOS プロセスのデザイン・ルールに基づいて，従来方式と提案方式の主要な回路のレイアウト設計をスクラッチから行った．得られたレイアウトから，回路面積を求めた．また，プロセス・パラメータに基づいて RC データを抽出し，Hspice シミュレーションによって遅延を求めた．

図5に，従来方式の tagD 用 RAM，CAM のセルを示す．DMT のセルの構成は，既に図2に示した．

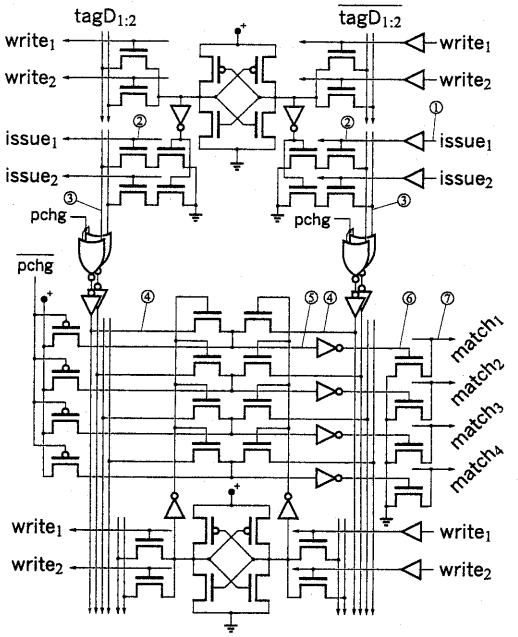

図5 従来方式の RAM セル（上）と CAM セル（下）

面　積

各回路の面積を**表1**に示す．表中，×1 の列は，各回路の 1 つ分の面積を表す．×2 の列は，命令キューが 2 本の場合にシステム全体で必要となる，RAM/CAM の 2 つ分，および，DMT の 4 つ分を表す．

提案方式においても，単に物理レジスタ・ファイルにアクセスするために，その番号 ——— すなわち，タグを記憶するため，従来方式の RAM と同じものが必要である．したがって，従来方式における CAM と，DMT を比較することが適当であろう．

DMT の縮小の方式として delay を採用した場合には，幅 WS_q，すなわち，各列で最大の DMT が必要であり，提案方式の面積は従来方式のそれを越える．一方 stall を採用した場合には，DMT の幅によって従来方式の 1〜数分の 1 程度となる．

いずれにしても，面積は 0.1mm^2 のオーダであり，チップ面積に占める割合は無視できるほど小さい．

表1 CAM と DMT の面積 ($\times 10^3 \mu m^2$)

構成		R10K×1		R10K×2	
		×1	×2	×1	×2
RAM		7.351	14.702	26.273	52.546
CAM		11.843	23.686	71.520	143.039
DMT	2	1.168	4.673	2.875	11.502
	4	2.336	8.945	5.751	23.003
	8	4.673	18.690	11.502	46.006
	16	9.354	37.416	23.003	92.013
	32	—	—	46.006	184.025

遅　　延

遅延の測定結果を，**図 6** に示す．測定条件は，電源電圧 1.8V，温度 85°C (typical) である．同図中の ①～⑦ は，図 5 中の点 ①～⑦ に対応する．⑦ より右は，IW 本のマッチ線の OR をとる部分の遅延である．

また，16→1，16→2，および，16→4，32→4 の *select* の遅延も同じ方法で測定した；ただし，$WS_q \rightarrow IW_q$ は，WS_q 個の命令から IW_q 個の命令を選択することを意味する．

これらのうち，前 2 つ（$IW_q = 1, 2$）には，通常利用される，1 命令を選択するアービタをカスケード接続した方式[1]を用いた．グラフにも示されているように，この方式の遅延は IW_q に比例するため，$IW_q \geq 3$ での利用は難しい．

そこで後 2 つ（$IW_q = 4$）には，発行要求の prefix-sum を計算する方式を用いた．紙面の都合上詳しくは述べられないが，prefix-sum 方式では，全体をたかだか $\log_2 WS_q + 1\ (= 5\sim6)$ 段のドミノ・ゲートで構成することができるため，カスケード方式を越える高速性を達成することができた．prefix-sum 方式の 16→4 の遅延は，カスケード方式の 16→1 のそれに比肩する．

MIPS R10000 は，*wakeup* は R10K×1，*select* は 16→2 にあたる．その場合，*wakeup* と *select* の遅延の合計は 986.4ps である．この部分が決める動作周波数の上限は 1.01GHz となり，現存する .18μm の 4-way スーパースケーラの仕様とおおよそ符合する．

DMT の縮小を行わない場合には，構成 R10K×1/R10K×2 では，DMT の幅は 16/32 となる．すなわち，従来方式の 2 本のバーと DMT の一番下の 2 本のバーをそれぞれ比較すればよい．DMT の縮小を行う場合には，従来方式の構成に関わらず，DMT のその幅のバーと比較すればよい．

構成 R10K×1 の場合，*wakeup*+*select* の遅延は従来方式の 68.0%，動作周波数の上限は 1.93GHz となる．DMT の幅を 4 まで縮小すると，前節の結果から delay 方式ならば IPC は 3%程度低下するが，遅延は 375.0ps，動作周波数の上限は 2.7GHz に達する．

同様に R10K×2 の場合では，*wakeup*+*select* の遅延は，従来方式の 75.7%，動作周波数の上限は 1.29GHz となる．DMT の幅を 8 まで縮小すると IPC はやはり 3%程度低下するが，遅延は 495.8ps，動作周波数の上限は 2.02GHz となる．

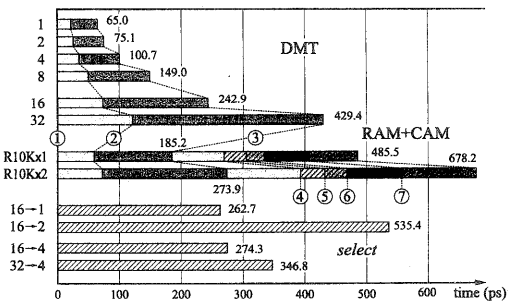

図 6　各回路の遅延

5. おわりに

本稿では，命令間の依存関係を直接的に表すテーブル DMT を用いた動的命令スケジューリング方式について述べた．本方式では，小容量の RAM を読み出す程度の処理で *wakeup* を実現することができる．

富士通株式会社から提供された .18μm CMOS プロセスのデザイン・ルールに基づいてレイアウト設計を行い，Hspice を用いて遅延を計測した．その結果，MIPS R10000 と同様の構成において，ロジックの遅延は 516.3ps となり，従来方式の 68.0%にまで削減された．その場合の動作周波数の上限は 1.93GHz となり，動的命令スケジューリングがクリティカルとなる可能性は極めて低いと言える．

加えて本稿では，DMT を縮小することによってその遅延を IPC のペナルティに転化する方法ついても述べた．シミュレーションによりペナルティを測定したところ，DMT をウィンドウ・サイズの 1/4 にまで縮小しても，IPC の悪化は 3%以下であることが分かった．DMT の縮小によって，R10000 の 2 倍の計算資源を持つ仮想的な 8-way の構成においても，2.02GHz の動作周波数の上限を達成することができる．

以上から，スーパースケーラにおいて，その動的命令スケジューリング・ロジックがクロック速度を規制する可能性は極めて低くなくなったと結論づけることができよう．

謝辞

富士通株式会社には，LSI の設計情報をご提供頂いたのでここに深甚なる謝意を表したい．

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) Palacharla, S., Jouppi, N. P. and Smith, J. E.: Quantifying the Complexity of Superscalar Processors, Technical report, Univ. of Wisconsin-Madison (1996).

2) 縣亮慶, グェンハイハー, 五島正裕, 中島康彦, 森眞一郎, 富田眞治: Superscalar における低遅延な命令発行機構, 情処研報 2000–ARC–139, pp. 109–114 (2000).

3) Yeager, K. C.: The MIPS R10000 Superscalar Microprocessor, *IEEE Micro*, No. 4, pp. 28–40 (1996).

4) 五島正裕, グェンハイハー, 森眞一郎, 富田眞治: Dual-Flow: 制御駆動とデータ駆動を融合したプロセッサ・アーキテクチャ, 情処研報 98–ARC–130, pp. 115–120 (1998).