リザーベーションステーションと物理レジスタ・ファイルを併用するスーパースケーラ・プロセッサ

小西将人[†]　五島正裕[†]
森眞一郎[†]　富田眞治[†]

近年では，命令が発行されてから実行されるまでのレイテンシ，発行レイテンシの増加と，小容量化に伴う1次キャッシュ・ミス率の悪化のため，投機的スケジューリング・ミスによる性能低下が問題となりつつある．本稿では，最近多くの高性能なプロセッサが採用している物理レジスタ・ファイルを用いたout-of-order実行方式に対して，リザベーション・ステーションを併用する技術を提案する．この方式では，レジスタ読み出しをバックエンドではなく，フロントエンドで行うことにより，発行レイテンシを半減することができる．シミュレーションの結果，SPEC95 ベンチマークでは平均で 20.0%の性能向上を確認した．

A Superscalar Processor Using Reservation Station and Physical Register File Together

MASAHITO KONISHI,[†] MASAHIRO GOSHIMA,[†] SHIN-ICHIRO MORI[†]
and SHINJI TOMITA[†]

Recently, the increase of *issue latency* is becoming a problem. Issue latency is cycles from the instruction scheduling to actual execution. The increase of the issue latency increases miss-penalty of instruction scheduling and degrades the performance. When a line buffer is used to reduce load latency, scheduling misses will occurs with high frequency and the issue latency problem will become apparent. This paper introduces yet another design of a superscalar processor using a reservation station and a physical register file together. Since the register file read is performed in the front-end, it can drastically reduce the issue latency. Evaluation result shows it achieves an average speed-up of 20% for the SPEC95 benchmark.

1. はじめに

LSIの微細化に伴い，スーパースカラ・プロセッサの設計のバランスも変化する．LSIの微細化に対して，ゲート遅延は順調に短縮されるが，配線遅延はほとんど短縮されない．キャッシュやレジスタ・ファイル，命令ウィンドウなど，演算器以外のほとんどの部分は配線遅延成分を相対的に多く含むため，それらの遅延が相対的に増大することになる．そのため，LSIの微細化はより深いパイプラインの採用を促すことになる[1)]

深いパイプラインでは，各部のレイテンシの増加が性能に悪影響を与えるが，分岐予測をはじめとする投機技術によって，性能低下は最小限に抑えられている．

しかし近年では，投機的命令スケジューリングのミス・コストの増加が，新たに問題となりつつある．

従来は，ロード命令がすべて1次キャッシュにヒットすると予測して，それに依存する命令を投機的にスケジューリングしていた．この投機のミス・ペナルティは，命令のスケジューリングから実行までのレイテンシ，すなわち，**発行レイテンシ**によって決まる．また，ミス率は，1次キャッシュ・ミス率に等しい．

近年では，これらの両方が悪化しつつある．発行レイテンシは，パイプラインの深化の一環として増大する傾向にある．また，レイテンシの短縮のために1次キャッシュは徐々に小容量化されており，それに伴って1次キャッシュ・ミス率も悪化しつつある．

そのため最近では，スケジューリング・ミス率の低減を目的として，1次キャッシュのヒット/ミスを予測する手法が提案されている[2),3)]．これらの手法では，1次キャッシュ・ミス率とは独立に，スケジューリング・ミス率を低減することができる．

それに対して本稿では，スーパースカラ・プロセッサの基本構造を見直して，発行レイテンシを半減する手法を提案する．以下2章で投機的スケジューリングについて説明した後，3章で提案方式について詳しく説明する．最後に4章でシミュレーションによるIPCの評価結果を示す．

† 京都大学　Kyoto University

2. 投機的命令スケジューリング

発行レイテンシは，各命令の**実行レイテンシ**が静的に定まっている場合にはいくら長くても問題にならない．しかしロード命令の実行レイテンシはキャッシュのヒット/ミスによって動的に変化してしまう．そのため発行レイテンシは，キャッシュ・ヒット/ミス予測のミス・ペナルティとして性能に悪影響を与えることになる．

命令の実行レイテンシが静的に定まっている場合には，この命令が発行された時点でその結果が得られるサイクルが決定されるため，これに依存する命令をペナルティなく発行することができる．**図 1 (a)** は，実行レイテンシが 1 である先行命令と，これに依存する後続命令が実行される様子を示している．同図では，発行レイテンシを 2 サイクルとしている．先行命令が発行された後，その命令の実行レイテンシ（図では 1 サイクル）だけ遅れて後続命令を発行すると，2 つの命令は back-to-back に実行される．

しかし先行命令がロードの場合，その実行レイテンシがキャッシュのヒット/ミスに依存して動的に変化するため，問題は複雑になる．

図 1 **(b)** に，先行命令がロードの場合の様子を示す．同図では，ロード命令が 1次キャッシュ（以下 **L1C**）にヒットすると予測して，ロード命令が発行されてから，L1Cヒット時の実行レイテンシ（図では 2 サイクル）だけ遅れて，後続命令を発行している．実際に L1C にヒットした場合には，2 命令は back-to-back に実行できる．

しかし L1C にミスした場合，後続命令はこのタイミングでは実行することができない．図 1 **(c)** に，その場合の様子を示す．同図では，後続命令は一旦キャンセルされ，ロード・データが LIC にリフィルされた後に再発行されており，2 つの命令は back-to-back に実行できていない．back-to-back に実行できた場合に対する後続命令の実行の遅れが，ミス・ペナルティである．図から分かるように，ペナルティはほぼ発行レイテンシによって決まる．

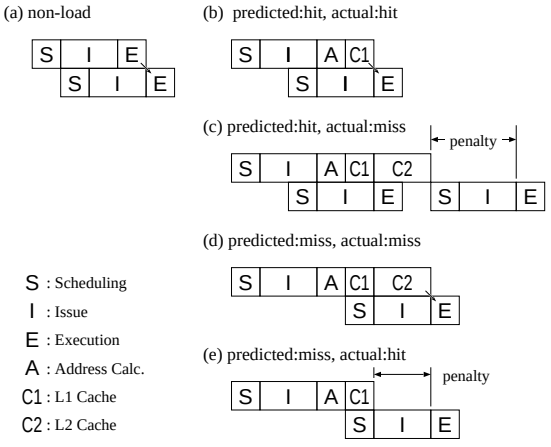

図 1 1次キャッシュ・ヒット/ミス予測による投機的スケジューリング

従来では，ロード命令がすべて L1C にヒットすると予測しても，スケジューリング・ミスが性能に与える影響は軽微であった．ペナルティを決定する発行レイテンシがそもそも 1 サイクル程度と短かく，スケジューリング・ミス率，すなわち，L1C ミス率もまた低かったためである．

しかし近年では，発行レイテンシが増大するのに加えて，L1C ミス率も悪化する傾向にある．L1C のミス率が悪化するのは，そのレイテンシの増大に対処するため，小容量化される傾向にあるからである．スケジューリング・ミスの影響が無視できなくなりつつあるのは，発行レイテンシの増大と L1Cミス率の悪化の相乗的な作用による．

そのため最近では，L1C のヒット/ミスを予測することにより，スケジューリング・ミス率を低減する手法が提案されている．Alpha 21264 は，L1Cヒット/ミス予測器を搭載している[2]．また福田らは，分岐予測に用いられる gshare 予測器を用いて**ライン・バッファ**のヒット/ミス予測を行う手法を提案している[3]．ライン・バッファとは，ロード/ストア・ユニットに付随するごく小容量のキャッシュであり[4]，小容量ゆえにスケジューリング・ミスの問題もいっそう深刻化する．

これらの手法では，予測器がミスと予測した場合には，次の階層のキャッシュにはヒットすると予測して，そこからデータが届くタイミングに合わせて，後続命令をスケジューリングする．図 1 **(d)** は，予測どおりにミスした場合の様子を示しており，2 つの命令は back-to-back に実行でき，ペナルティは生じない．一方，図 1 **(e)** は，L1Cミスと予測したが，実際にはヒットであった場合を示している．この場合，再スケジューリングの必要はないが，実行タイミングは L1Cミスの場合と同じになり，L1C の効果が得られないことになる．

3. 提 案 方 式

スケジューリング・ミスの問題に対処する方法としては，前章で述べた L1Cヒット/ミス予測などによってスケジューリングのミス率を改善する方法と，ミス・ペナルティを決定する発行レイテンシを短縮する方法がある．本稿では，スーパースカラ・プロセッサの基本構造を見直して，発行レイテンシを短縮することを考える．

さて，out-of-orderスーパースカラ・プロセッサは，**リザベーション・ステーション**（以下，**RS** と略）と**リオーダ・バッファ**（同，**ROB**）を組み合わせる方式[5),6)] と，大容量の**物理レジスタ・ファイル**（同，**PRF**）に対してレジスタ・リネーミングを施す方式に大別される．以降では，前者を **RS+ROB 方式**，後者を **PRF 方式**と呼ぶ．PRF 方式では，演算器から PRF を直接読み書きすることができ，RS+ROB 方式に比べ，データ・パスを大

幅に簡略化することができる．PA-8000[7]，R10000[8]，21264[2]，Pentium 4，Athlon/Opteron など，最近の高速なスーパースカラ・プロセッサのほとんどは PRF 方式を採用している．

しかし PRF 方式は，RS+ROB 方式より発行レイテンシが長いという欠点を持つ．RS+ROB 方式では，ソース・オペランドの読み出しをディスパッチ前，すなわち**フロントエンド**で行う．対して PRF 方式では，命令発行後，すなわち**バックエンド**で行う．そのため PRF 方式では，RS+ROB 方式に対して，発行レイテンシが倍増してしまうのである．前章で述べたように，長い発行レイテンシは，投機的スケジューリング・ミスによる性能低下を深刻にしてしまう．

そこで本稿では，PRF 方式に対して RS を併用することを提案する．この構成では，RS+ROB 方式と同様，ソース・オペランド読み出しをフロントエンドで行うことができ，従来の PRF 方式に比べ発行レイテンシを半減することができる．以下ではまず 3.1 節で従来の out-of-order 実行方式についてまとめた後，3.2 節以降で提案方式について述べる．

3.1 従来方式

スーパースカラ・プロセッサの命令パイプラインは，おおよそ以下のフェーズからなる．なお，各フェーズは，1 つまたは複数のステージからなる：

F 命令がフェッチ，デコードされる．
D その後命令は，命令ウィンドウを構成する**ペイロード RAM** にディスパッチされる．
S 実行可能な命令が検出され (*wakeup*)，その中から実際に実行される命令が選択される (*select*)．
I スケジューリングされた命令の情報がペイロードから読み出され，実行ユニットに送られる．
E 各命令に対応した実行ユニットにおいて，命令の実行が行われる．

実行フェーズ**E** の前後には，ソース・オペランドを読み出す処理 **R** と，実行結果を書き戻す処理 **W** がそれぞれ必要であるが，以下で述べるように，それらは方式によってパイプライン内での位置が異なる．

3.1.1 RS+ROB 方式

図 2 (a) に，RS+ROB 方式のブロック図を示す．この方式では，ソース・オペランドの読み出し R はフロントエンドで行われる．各フェーズは以下のように進む：

F 命令がフェッチ，デコードされる．
R ROB は，ソース・オペランドの論理レジスタ番号をキーに，プログラム・オーダ上で最近書き込まれた値を連想検索する．
D 命令がディスパッチされる時に，ROB から読み出されたソース・オペランドは RS に書き込まれる．
S 命令の実行が決まり，
I 命令が発行される時に，ソース・オペランドは RS から読み出される．
E 命令が実行されると，
W 実行結果は，ROB に書き込まれると同時に，この結果を利用する命令の RS エントリにフォワードされる．

このようにペイロード RAM，RS の読み出しは並列になされるので，発行レイテンシはこれらの読み出しレイテンシの大きい方で与えられる．

3.1.2 PRF 方式

図 2 **(b)** に，PRF 方式のブロック図を示す．この方式では，ソース・オペランドの読み出し R はバックエンドで行われる：

F 命令がフェッチ，デコードされる．
N レジスタ・リネーミングにより，ソース・オペランドに対応する物理レジスタの番号が求められる．
D 物理レジスタ番号は，命令の情報の一部として，ペイロードに書き込まれる．
S 命令の実行が決まると，
I 命令が発行される時に，物理レジスタ番号がペイロードから読み出され，
R その番号で PRF がアクセスされ，ソース・オペランドが読み出される．
E 命令が実行され，
W 結果は，PRF に書き戻される．

このように，I と R は逐次的に行われる必要がある．したがって，ペイロードの読み出しレイテンシと PRF の読み出しレイテンシの和が，発行レイテンシを与える．3.3 節で詳述するように，PRF の規模はペイロードに比べて何倍も大きく，発行レイテンシのかなりの部分を PRF の読み出しレイテンシが占めることとなる．

冒頭で LSI の微細化にともなって演算器以外の各部の遅延が相対的に増大すると述べたが，ペイロードと PRF も例外ではない．そのため，投機的スケジューリングの問題は PRF 方式でより顕著に現れることになる．

3.2 提案方式

提案方式では，PRF と RS を組み合わせる．図 2 **(c)** に，提案方式のブロック図を示す．Out-of-order スーパースカラ・プロセッサは，out-of-order 実行のための以下の基本的な機能を持つ：

(1) **レジスタ・リネーミング**　各命令の論理レジスタ番号を物理的なロケーションの番号に付け換える．
(2) **マシン状態の管理**　割り込みや投機ミスに対して，適切な状態を回復する．

提案方式では，これらの基本機能は，PRF 方式と同様，主に大容量の PRF によって提供される．それに対して RS は，命令ウィンドウの一部として，PRF から読み出されたソース・オペランドのコピーを一旦保持する役割を担う．

提案方式では，命令は以下のように実行される：

F 命令がフェッチ，デコードされる．
N PRF 方式と同様，レジスタ・リネーミングにより，ソース・オペランドに対応する物理レジスタの番号が求められる．

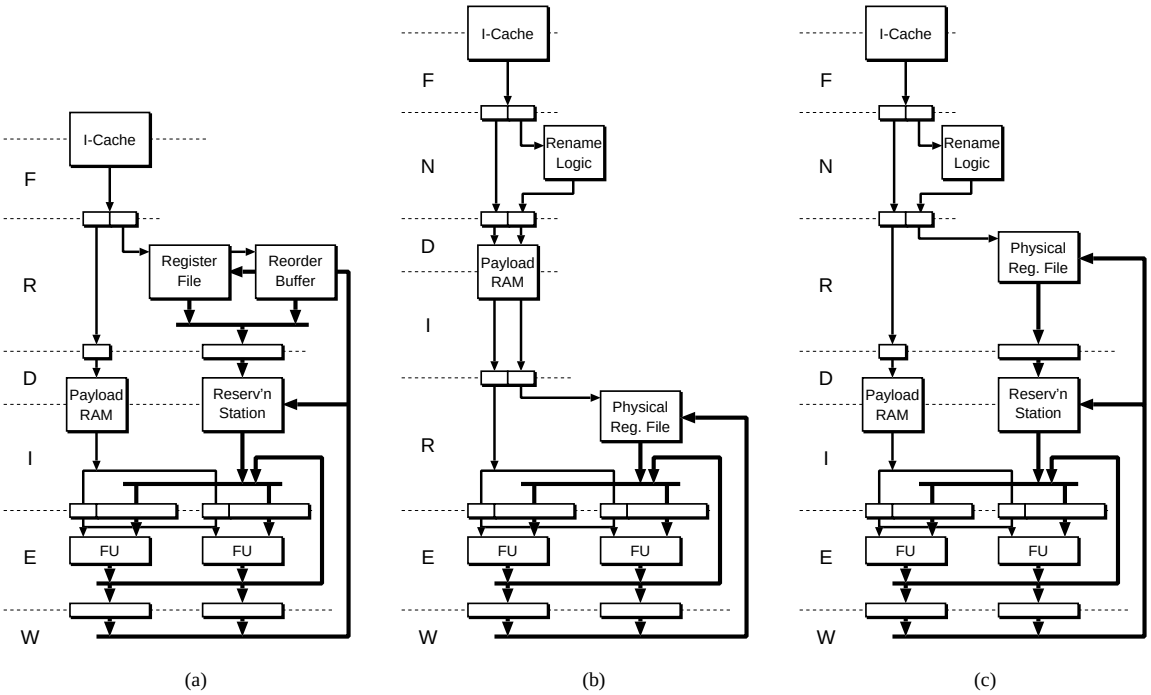

図 2　(a) RS+ROB 方式　(b) PRF 方式　(c) 提案方式

R　PRF 方式とは異なり，PRF に対する読み出しは，フロントエンドで，N の直後に行われる．

D　命令がディスパッチされる時に，PRF から読み出されたソース・オペランドが RS に書き込まれる．

S　命令の実行が決まると，

I　RS+ROB 方式と同様に，命令がペイロードから発行されると同時に，ソース・オペランドは RS から読み出される．

E　命令が実行され，

W　実行結果は，PRF 方式と同様 PRF に書き戻されると同時に，RS+ROB 方式と同様この結果を利用する命令の RS エントリにフォワードされる．

このように，RS を併用することによって，PRF 方式ではバックエンドで行われる PRF 読み出しを，RS+ROB 方式と同様フロントエンドで行うことができる．前述したように，PRF の読み出しレイテンシは発行レイテンシのかなりの部分を占めるから，これによって発行レイテンシを大幅に削減することができる．

なお，PRF 読み出しをフロントエンドで行うためフロントエンドのレイテンシは増大することになるが，それによる性能低下は分岐予測のヒット率の高さによってある程度補償することができる．

3.3　提案方式のロジック

提案方式では，従来の PRF 方式に対して RS を追加する．本節では，RS の追加が，各フェーズのレイテンシに与える影響を明らかにする．

RS の処理は，以下の 2 つである：

(1)　**ディスパッチと発行**　ディスパッチ D と発行 I において，ソース・オペランドの書き込みと読み出しが行われる．
これらの遅延がペイロード RAM のそれより長いと，D，I のレイテンシの増加要因になる．

(2)　**フォワーディング**　W フェーズにおいて，実行結果は，PRF に書き戻されると同時に，RS の必要なエントリに対してフォワーディングする必要がある．フォワーディングは，書き込みエントリの特定と，実際の書き込みの 2 つに分けられる．W フェーズでの書き込みエントリは，PRF では物理レジスタ番号によって直接指示されるが，RS では実行結果に対応するソース・オペランドを見つける必要がある．また RS 本体 RAM に対する書き込み遅延が PRF のそれより長いと，W のレイテンシの増加要因となる．

3.3.1　書き込みロジック

書き込みエントリは，E フェーズで並列に求めることが可能である．したがってフォワーディングが W のレイテンシに与える影響は，RS 本体 RAM の書き込みの遅延のみ考えればよい．

また，この処理の遅延は，**依存行列**[9] を応用することで短縮できる．従来，書き込みエントリの特定はタグの連想検索によって行われており[5),6)]，その遅延が問題となると考えられていた．しかし，依存行列を用いることで，連想検索を省略することができる．依存行列は，文献 9) では，S フェーズの一部 (*wakeup*) を高速

表 1　R10000 の整数キューのペイロード，RS，PRF のパラメタ

	bit	word	read port	write port
ペイロード	60	16 (*WS*)	2 (*IW*)	4 (*DW*)
RS	64	16 (*WS*)	2 (*IW*)	7 (*DW* + *WP*)
PRF	64	64	7 (*RP*)	3 (*WP*)

WS : ウィンドウ・サイズ　(16)
IW : 発行幅　(2)　*DW* : ディスパッチ幅　(4)
RP : PRFリード・ポート数 (7)　*WP* : PRFライト・ポート数 (3)

化する技術として提案されているが，書き込みエントリの特定にも応用できる．行列の各行は生産側の命令に，各列は消費側の命令のソース・オペランドに対応している．この行列を格納する RAM を読み出すことで，書き込みが必要となるエントリを求めることができる．

3.3.2　ロジックの規模

表 1 に，R10000 の整数系の構成の場合のペイロード，RS，PRF の RAM のパラメタをまとめる[8]．以下の点に注意されたい：

- RS はソース・オペランドごとに分割することができる．表 1 には，そのうちの 1 つを挙げてある．
- RS に対する書き込みは D とフォワーディングで行われ，ライト・ポート数は *DW* と *WP* の和になる．

RS は，ペイロードに比べて，ライト・ポート数が *WP* = 3 本多い分だけ大きい．RS と PRF では，ポート数の合計は同程度で，PRF は RS に比べて 4 倍も大きい．したがって，ペイロード RAM，RS，PRF の遅延をそれぞれ D_{PL}，D_{RS}，D_{PRF} とすると，$D_{PL} < D_{RS} < D_{PRF}$ のように表せる．

3.3.3　各フェーズのレイテンシ

従来の PRF 方式に対して，提案方式における各フェーズのレイテンシは以下のように変化する：

W　3.3.1 項の議論により，W のレイテンシは，RS と PRF の書き込み遅延の大きい方で与えられる．$D_{RS} < D_{PRF}$ より，RS への書き込みが W のレイテンシに影響を与えることはない．

D　D のレイテンシは，ペイロードと RS の書き込み遅延の大きい方で与えられる．$D_{PL} < D_{RS}$ であるから，PRF 方式より増加することになる．

発行レイテンシ　発行レイテンシは，$D_{PL} + D_{PRF}$ から D_{RS} に変化することになる．$D_{PL} < D_{RS} < D_{PRF}$ であるから，例えば，RS の遅延が，ペイロードと PRF の遅延の中間程度の値である，すなわち，$D_{RS} \simeq (D_{PL} + D_{PRF})/2$ と仮定すると，発行レイテンシは半減されることになる．

4. 性能評価

SimpleScalar ツールセットにライン・バッファ（以下，LB）を実装し，SPEC95 ベンチマークを用いて IPC を評価した．提案方式に加え，比較対象として，スケジューリングの工夫によってミス・コストの低減を図る，福田らの gsahre 予測器を用いる方法[3] と，キャンセルされた命令を即時再発行する方法の 2 つを評価した．

以下，4.1 節で即時再発行を行う方式について述べた後，4.2 節以降で，評価の方法と結果について述べる．

4.1　即時再発行方式

福田らの方法では，予測ミスによりキャンセルされた命令は，LBリフィルを待って再発行している[3]．これに対して，LB ミスを起こしたロード命令であっても L1C にはヒットすると予測し，キャンセルの直後に即時再発行する方法が考えられる．前者を **wait** モデル，後者を **imm** モデルと呼ぶことにする．

図 3 に，それぞれのモデルでのスケジューリングの様子を示す．同図は LB ミス/L1C ヒットの場合のものである．wait モデルではペナルティは 3 サイクルとなっているが，imm モデルでは，これより 2 サイクル短縮され，1 サイクルとなっている．

4.2　評価モデル

評価モデルに共通のパラメータを**表 2** に示す．各モデルに固有のパラメータを以下に示す：

基本方式　従来の PRF 方式と提案方式では，PRF 読み出しをそれぞれバックエンド，フロントエンドで行うので，それぞれを **B**，**F** で表す．

各フェーズのサイクル数　今回の評価では，提案方式により発行レイテンシが半減すると仮定した．$l = 1$, 2, 4 に対して，発行レイテンシは，F の場合 l サイクル，B の場合 $2l$ サイクルとした．また，各フェーズのサイクル数は次のように与えた：F – 4, N – l, D – l, S – 1.

再発行のタイミング　前節で述べた wait と imm の 2 つの方法を評価した．

ライン・バッファ・ヒット/ミス予測　予測器を用いず全て LBヒットと予測する方法 **hit** と，福田らが提案した gshare 予測器を用いる方法[3]**pred** を評価した．

1 つのモデルは，(B/F)-(*l*)-(wait/imm)-(hit/pred)の 4 B-1-wait- hit のように表される．

4.3　評価結果

結果を**図 4** に示す．グラフは，B-1-hit で正規化した IPC を示している．各プログラムの 4 本のバーのうち，左 2 本と右 2 本は，それぞれ B，F を示している．B，F の各 2 本はそれぞれ，hit と pred を示している．1 本のバーはそれぞれ 3 つの部分に分かれており，上から $l = 1$, 2, 4 の場合を示している．

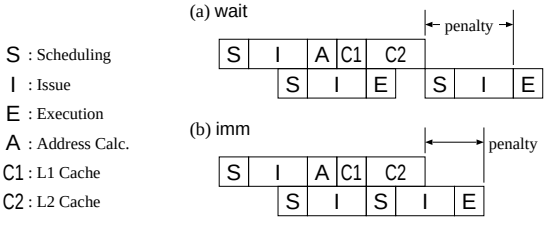

図 3　即時再発行方式

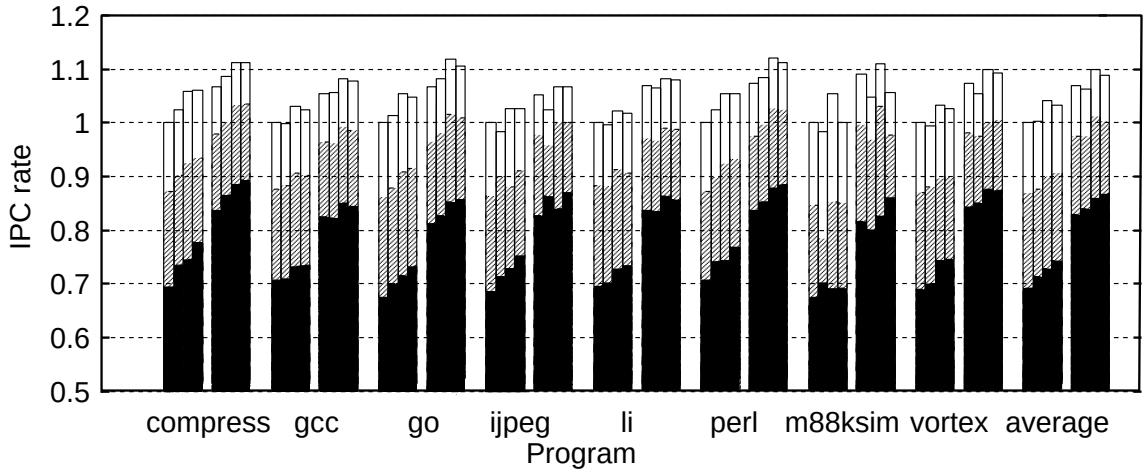

図 4　IPC の比

表 2　モデルの主要なパラメータ

ウェイ数	8
機能ユニット	iALU 4, LD/ST 4, iMUL/DIV 2, fADD 4, fMUL 4, fDIV 2
分岐予測機	gshare，PHT エントリ数 8K，履歴長 7b
PHT エントリ	2b カウンタ，ヒット閾値 2，増減値 1
ライン・バッファ	64B × 16 エントリ
1 次キャッシュ	64KB, 2 way, レイテンシ 4
2 次キャッシュ	1MB, 8 way, レイテンシ 10

wait/imm

wait モデルと比較して imm モデルは，平均で 5% 程度の性能向上が得られている．

hit/pred

pred モデルは hit モデルに対して，$l = 4$ の場合で最大 5% 程度の性能向上が得られるが，l が短い場合には予測ミスの悪影響によってかえって性能低下するプログラムもある．

文献 3) では，カウンタの増減値や履歴長をチューニングすることで最大で 16.0% の性能向上が得られると報告しているが，今回そのようなチューニングを行っていないため，それほどの結果が出ていない．

提案方式の効果

l が同一の場合，提案手法のみ適用した F-l-wait- hit モデルは B-l-imm- pred よりも高い IPC を示しており，ヒット/ミス予測よりも，発行レイテンシを短縮する効果の方が大きいことが分かる．F-4-hit は，B-4-hit に対して平均で 20.0% の IPC が向上している．しかし，更に imm と pred を組み合わせても，平均で 3% しか IPC は向上しない．

5. おわりに

本稿では，物理レジスタ・ファイルに対してリザベーション・ステーションを併用するスーパースカラ・プロセッサの構成方式について述べた．この方式では，物理レジスタ・ファイルの読み出しをバックエンドからフロントエンドに移すことによって，発行レイテンシを大幅に削減することができる．発行レイテンシを 8 から 4 サイクルに削減できるとすると，平均 2 割程度の性能向上を達成することができる．今後，各機構のレイテンシを評価によって明らかにし，より詳細な評価を行う予定である．

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) #13480083 による．

参考文献

1) Palacharla, S. et al.: Quantifying the Complexity of Superscalar Processors, Technical report, Univ. of Wisconsin-Madison (1996).
2) Keller, J.: The 21264: A Superscalar Alpha Processor with Out-of-Order Execution, *9th Annual Microprocessor Forum* (1996).
3) 福田祥貴, 片山清和, 島田俊夫: ライン・バッファ・ヒット/ミス予測を利用した動的命令スケジューリングの高精度化手法, *SACSIS* (2003).
4) Wilson, K. et al.: High Bandwidth On-Chip Cache Design, *IEEE Trans. Comp.*, Vol. 50, No. 4 (2001).
5) Tomasulo, R.: An effective algorithm for exploiting multiple arithmetic units, *IBM J.*, Vol. 11 (1967).
6) Johnson, M.: *Superscalar Micorprocessor Design*, Prentice-Hall (1991).
7) Kumar, A.: The HP PA-8000 RISC CPU, *Hot Chips VIII*, pp. 9–20 (1996).
8) Yeager, K.: The MIPS R10000 Superscalar Microprocessor, *IEEE Micro*, No. 4 (1996).
9) 五島正裕ほか: 行列に基づく Out-of-Order スケジューリング方式の評価, 情報処理学会 HPS 論文誌, Vol. 43, No. SIG 6(HPS5) (2002).