頻出値を利用した物理レジスタの共有化手法

山本哲弘[†] 大熊穣[†,☆] 片山清和[†,☆☆]
小林良太郎[†] 安藤秀樹[†] 島田俊夫[†]

近年のプロセッサではより多くの命令レベル並列性を利用するため，大きなレジスタ・ファイルを必要としている．しかし，大きなレジスタ・ファイルは，アクセス時間，面積，消費エネルギーの点で問題がある．本論文では，頻繁に生成される値について，物理レジスタを共有化することにより，要求される物理レジスタ数を削減する手法を提案する．本手法は，レジスタ・ファイル内に存在するすべての値を共有化するこれまでの手法とは異なり，単純なハードウェアで実現できる．SPECint95 ベンチマークを用いて評価した結果，物理レジスタを共有しない従来手法では，使用可能な物理レジスタ数をプログラムを実行可能な最少数にした場合，十分に物理レジスタが存在する場合に比べ IPC が 70%低下する．これに対して，本手法では強い耐性を示し，0 と 1 を共有する方式で 25%，0 のみ共有する方式で 28%まで抑制できることが分かった．また，十分にレジスタが存在する場合の IPC をほぼ達成するために必要な物理レジスタ数を，レジスタを共有しない従来手法に比べて，0 と 1 を共有する方式で 34%，0 のみを共有する方式で 30%削減できることが分かった．その結果，レジスタ・ファイルのアクセス時間を 27%，面積を 31%，消費エネルギーを 33%削減できることが分かった．

A Sharing Scheme of Physical Registers by Exploiting Frequent Values

Akihiro Yamamoto,[†] Minoru Okuma,[†,☆]
Kiyokazu Katayama,[†,☆☆] Ryotaro Kobayashi,[†] Hideki Ando[†]
and Tosio Shimada[†]

Processors in recent years require a larger register file to exploit a more amount of instruction-level parallelism. However, a large register file has problems of access time, area and energy consumption. This paper proposes a scheme that reduces the required number of physical registers by sharing physical registers only among frequently-generated values. Our scheme can be implemented with simple hardware unlike the previous sharing scheme, which can fully share physical registers among the value in the register file. Our evaluation results using SPECint95 benchmark show that the conventional scheme that does not share physical registers decreases IPC by 70% with the minimum number of registers which allows the processor to normally execute programs, compared to that with the enough number of registers. On the other hand, our scheme exhibits high tolerance; it can suppress IPC degradation to 25% with 0 and 1 values sharing, or to 28% with 0 value sharing. Our evaluation results also show that our scheme can reduces the number of required registers by 34% with 0 and 1 values sharing and by 30% with 0 value sharing to achieve IPC with the enough number of registers. As a result, our scheme reduces the access time by 27%, the area by 31%, and the energy consumption by 33%, compared to the conventional non-sharing scheme.

1. はじめに

近年のスーパスカラ・プロセッサでは，命令の並列性をより多く利用することによって性能を上げている．より多くの並列性を利用するためにはより大きなレジスタ・ファイルを必要とする．

また，近年注目されているアーキテクチャとして，SMT（Simultaneous Multi-Threading）アーキテクチャが存在する[22]．SMT は，スーパスカラ・プロセッサの資源を利用し，複数のスレッドからの命令を同時実行する技術である．SMT は，近年のプロセッサの豊富な資源を有効に利用することができるという利点がある．しかし一方で，複数のスレッドのコンテキストを同時に保持しなければならないため，単一スレッド実行のプロセッサに比べ，より大きなレジスタ・ファイルを必要とする．このため，今後もレジスタ・ファ

† 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University
☆ 現在，トヨタ自動車株式会社
Presently with Toyota Motor Corporation
☆☆ 現在，四日市大学経済学部
Presently with Faculty of Economics, Yokkaichi University

イルのサイズは大きくなると思われる．

レジスタ・ファイルは大きくなるほどアクセス時間は長くなり，その結果クロック・サイクル時間に悪影響を与える可能性がある．また，チップ上に占める面積，消費エネルギーも大きくなる．

この問題を解決する方法として，ある目標性能を達成するために必要な物理レジスタ数を削減するアプローチが存在する[10),14)]（以下，明に記述しない限り，レジスタとは物理レジスタを指すとする）．その中で，複数の命令間でレジスタを共有する手法は，有望なアプローチである．Jourdan らは，これをすべての命令間で実現する手法を提案した[10)]．この手法では，命令の実行結果が得られたときに，レジスタ・ファイルを連想検索し，同一の値を保持するレジスタを共有する．これにより，必要レジスタ数は削減される．しかし，レジスタ・ファイルを検索可能にするために回路は複雑化し，レジスタ数削減との間にトレードオフが存在する．したがって，真にアクセス時間，面積，消費エネルギーが減少するかどうかは明らかでない．

これに対して，我々は，すべての値ではなく，頻出するあらかじめ決められた値のみを共有化し，実行時のオーバヘッドなく必要レジスタ数を削減する手法を提案する[15),16)]．実行前に共有化する値を定め，実行時にはそこで定められた値を保持するレジスタを共有化する．この方法では，実行時に結果値の連想検索が不要なので，レジスタ・ファイルを複雑化させることがない．これによって必要レジスタ数削減の効果が，直接，アクセス時間，面積，消費エネルギーの削減に結び付けることができる．

効果的にこのような共有化を実現するには，種々のアプローチが考えられる．最も基本的なアプローチは，実行結果に最も頻繁に出現する少数の値についてのみ共有化する方法である．この方法では，定められた値を保持するレジスタを用意し，実行時にそのいずれかに一致する結果を出力する場合のみ共有化する．このような手法が有効となるためには，プログラムに依存することなく，頻繁に出現する値がなくてはならない．これについては，文献 20)，25) において調査がなされている．これらによれば，多くの動的命令が 0 または 1 を出力することが報告されている．この事実から，本手法の有効性が期待できる．

このほかに考えられるアプローチとして，共有化する値をコンパイラによりハードウェアに指示する方法が考えられる．この方法はプログラムに埋め込まれた指示により共有化する値を実行時に変化させる点が第 1 の手法と異なる．頻繁に出現する値が実行の過程において変化するなら，この方法は第 1 の手法に比べて有利となる．

本論文では，上記の共有化アプローチのうち，頻繁に出現する結果値について共有化する手法と機構について提案する．まず，2 章で，すべての値を共有化する手法について説明する．次に 3 章で，提案手法と機構について説明し，4 章で評価する．5 章では関連研究について述べ，最後に 6 章で本論文をまとめる．

2. 従来のレジスタ共有化手法

Jourdan らは，同じ値を生成する複数の命令間でレジスタを動的に共有化する手法を提案した[10)]．彼らは，ある命令が生成した値と同じ結果を生成する命令が，その命令より以前の最近の命令の中に見つかる確率が非常に高いことを発見した．この性質を利用すれば，多くの命令がレジスタを共有できる．

Jourdan らの方法では，命令の実行終了時に物理レジスタを割り当てる**仮想物理レジスタ方式**[7),8)] を前提としている．この方式では，通常のレジスタ・ファイルによるリネーミングと異なり，フロントエンドでは，デスティネーション・レジスタには，仮想物理レジスタと呼ぶタグを与えるのみで，物理レジスタの割当ては行わない．割当ては，実行終了時にまで遅らせて行う．こうすることにより，物理レジスタの生存期間が短くなり，必要レジスタ数が削減される．

機構としては，GMT（General Map Table）および PMT（Physical Map Table）と呼ばれる 2 つの表を用意する．GMT は，論理レジスタ番号でインデックスされる．各エントリは，その論理レジスタ番号に割り当てられている仮想物理レジスタ番号（VP），その仮想物理レジスタ番号に物理レジスタが割り当てられているかどうかを示すフラグ（P），その物理レジスタが有効な値を保持しているかどうかを示すフラグ（V）を保持する．PMT は，仮想物理レジスタ番号でインデックスされ，エントリにはその仮想物理レジスタ番号に割り当てられている物理レジスタ番号を保持する．また，物理レジスタ用のフリー・リストに加えて，仮想物理レジスタ用のフリー・リストも用意する．

命令のデコード時，ソース・レジスタをリネームするために，GMT を参照する．すでに P フラグがセットされていれば，物理レジスタ番号にリネームし，そうでなければ仮想物理レジスタ番号にリネームする．デスティネーション・レジスタは，仮想物理レジスタにリネームし，GMT を更新する．

発行は通常どおり行われるが，命令は命令ウィンドウから即座に削除されない．これは，レジスタへの書

き込みの際に，物理レジスタが得られなければ，再実行の必要があるからである．

実行結果の書き込みの際は，物理レジスタ番号をフリー・リストから得，結果値を書き込む．同時に，GMT，PMT を更新する．もしフリー・リストが空であれば，実行をキャンセルし，後に再実行される．

コミットの際には，デスティネーション・レジスタ番号と同じ論理レジスタ番号に以前割り当てられた物理レジスタ番号をフリー・リストに返す．同時に，仮想物理レジスタ番号もフリー・リストに返す．

Jourdan らの方法では，この方式を前提とし，命令の実行結果が得られた後に，レジスタ・ファイルを実行結果で検索する．同一の値を保持するレジスタがあればそれを共有する．そうでなければ新しく物理レジスタを割り当てる．

この方法により，必要なレジスタ数は大きく削減され，レジスタ・ファイルを小さくできる．この結果，アクセス時間，面積，消費エネルギーの削減が期待できる．しかし，実際には次の 2 つの理由で，その効果は減少することになる．第 1 に，レジスタ・ファイルの複雑化により，アクセス時間，面積，消費エネルギーが増加する．まず，値の連想検索のために，レジスタ・ファイルを CAM で構成しなければならない．また，検索する値を入力するために，ポートが新たに必要である．これらにより，セル・サイズが増大する．第 2 に，レジスタ・ファイルを検索するためのステージが実行ステージの後に必要となる．これにより，バイパス経路が増加し，回路が複雑化する．以上 2 つの短所により，このようなレジスタ共有化が真に有効かどうかは明らかでない．

3. 頻出値のみの共有化手法

本章では，頻繁に出現する少数の値についてのみ，レジスタを共有化する手法について提案する．本手法が有効となるためには，頻繁に出現する少数の値がプログラムに大きく依存することなく存在することが必要となる．本章では，まず最初に，このことを我々の使用したベンチマーク・プログラムで確認する．その後，提案手法を説明する．

3.1 頻 出 値

Zhang らによると，メモリ・アクセス命令により参照される値には出現頻度の高い値があり，SPECint95 ベンチマークにおいては 0 と 1 が上位を占める場合が多いことを見つけた[25]．Sato らは，さらにレジスタへの書き込み値について調べ，同様の傾向があることを確認した[20]．本節では，その結果を我々の使用したベンチマーク・プログラムで確認する．

表 1 ベンチマーク・プログラム
Table 1 Benchmark programs.

ベンチマーク	入力	実行命令数
compress95	bigtest.in	95M
gcc	genoutput.i	84M
go	2stone9.in	75M
ijpeg	specmun.ppm	450M
li	train.lsp	183M
m88ksim	ctl.in	100M
perl	scrabbl.in	80M
vortex	vortex.in	80M

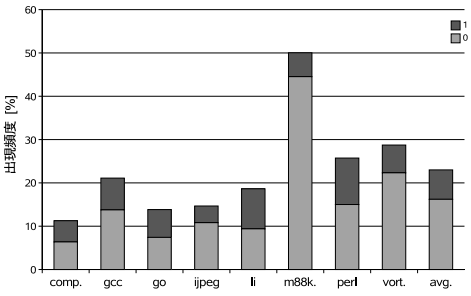

図 1 0 と 1 の出現頻度
Fig. 1 Occurrence frequency of 0 and 1.

ベンチマーク・プログラムとして，**表 1** に示す SPECin95 の 8 本を使用した．バイナリは，gcc ver.2.7.2.3 を用い，-O6 -funroll-loops のオプションでコンパイルし作成した．シミュレーション時間が過大にならないようにするために，命令ミックス，関数の出現頻度など，特徴をほぼ維持しつつ，それぞれのプログラムへの入力を調整した．さらに ijpeg は先頭から 50M をスキップし，450M 命令を実行した．vortex は先頭から 100M 命令スキップし，80M 命令を実行した．命令をスキップしたのは，初期化ルーチンを飛ばすためであり，実行命令数を限定したのは，シミュレーション時間が膨大になるのを避けるためである．

図 1 にベンチマークごとの 0 と 1 の出現頻度を示す．各棒グラフは 2 つの部分からなり，上部，下部はそれぞれ 1，0 の出現頻度である．0 と 1 の平均出現頻度は，それぞれ 16%，7%であった．0 と 1 以外で出現頻度の高い値はベンチマークごとに異なるが，最も高いものでも 7%で，多くは 3%以下であった．したがって，これまでに知られている調査結果と同様に，0 と 1 の出現頻度が，他の値に比べて圧倒的に高いことが分かった．

また，これらの値が静的に定まったものか，実行時にしか定まらないものかを調べた．その結果，前者に属する命令の出現頻度は約 3%と少なかった．したがっ

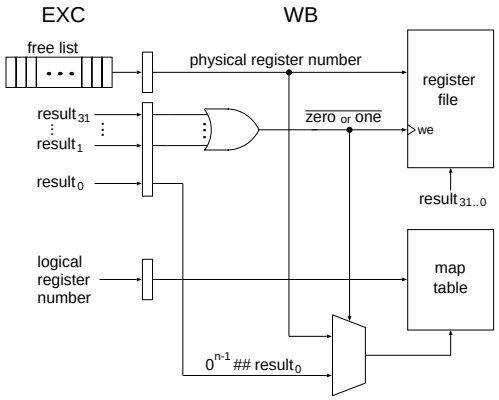

図 2 レジスタ共有化のための論理回路
Fig. 2 Logic circuit diagram for register sharing.

て，共有化においては，動的な機構が必要であることが分かる．

3.2 機　　構

前節の評価結果から，我々は，最も出現頻度の高い 0 と 1，または 0 のみについて，レジスタを共有化することを提案する．本手法は，レジスタ・リネーミングとして，論理レジスタをレジスタ・ファイルの中のレジスタへマップする方式を前提としている．まず，つねに 0 を保持するレジスタと，つねに 1 を保持するレジスタを用意する．整数レジスタの場合，実際には，0 を保持するレジスタは既存なので，新たに用意するレジスタは，1 を保持するレジスタのみである．仮想物理レジスタ方式を前提とし，命令の実行終了時に結果が 0 であったら 0 を保持するレジスタに割り当て，1 であったら 1 を保持するレジスタに割り当てる．それ以外の場合は，任意の値を保持できる通常のレジスタに割り当てる．0 のみについてレジスタを共有化する場合は，同様の動作を 0 のみに限定して行う．

提案手法では，従来の共有化手法とは異なり，結果値が 0 または 1 であると判定された時点で，共有するレジスタ番号が明らかである．したがって，共有するレジスタの検索は不要である．このため，レジスタ・ファイルを RAM で構成でき，ポート数の増加もなく，レジスタ・ファイルを複雑化することがない．

図 2 に，本機構の論理図を示す．図では，レジスタ・ファイルの第 0 エントリはつねに 0 を，第 1 エントリはつねに 1 を保持するレジスタと仮定している．

実行ステージ（EXC）での命令の実行と並行して，フリー・リストから空き物理レジスタ番号を得る．書き込みステージ（WB）の OR ゲートにより，結果が 0，1 のいずれでもないかどうかを判定する．0，1 のいずれでもない場合，レジスタ・ファイルへの書き込みを指示する信号（we: write enable）をアサートし，物理レジスタ番号で指定されたエントリに結果を書き込む．0 または 1 の場合，we をアサートせず，書き込みを行わない．一方，論理レジスタ番号で指定されたレジスタ・リネーミングのマップ表のエントリには，実行結果が 0 ならば 0 を，1 ならば 1 を，いずれでもなければ，フリー・リストから得た物理レジスタ番号を，MUX を介して書き込む☆．

本機構は，以下の理由で，クロック・サイクル時間に悪影響を与えないと考える．問題となるパスは，OR ゲートからレジスタ・ファイルの we をアサートするパスと，同じく OR ゲートが MUX を制御しマップ表に物理レジスタ番号を与えるパスの 2 つである．明らかに，後者の方が遅延時間が長い．一般に，レジスタ・ファイルのビット線書き込みドライバは，ワード線が立ち上がった後に駆動させる．したがって，問題のパスの遅延時間が，レジスタ・ファイルのワード線が立ち上がるまでの遅延時間より短ければ，クロック・サイクル時間に悪影響を与えないといえる．

そこで，まず CACTI[19),21),24)] を修正したシミュレータを用いて，16 個の読み出しポートと 8 個の書き込みポートを持つ 64 エントリのレジスタ・ファイルのアクセス時間を測定した．CACTI とは，キャッシュの回路を基本的なゲートと RC の等価回路にモデル化し，Horowitz の遅延モデル[9)] に基づいてゲート遅延を計算するプログラムである．プロセス技術には 0.07 μm CMOS を仮定した．次に，CACTI と同様に，Horowitz の遅延式を用いて，問題のパス（OR ゲートから MUX の出力を得るまで）の遅延時間を求めた．その結果，レジスタ・ファイルのワード線が立ち上がるまでの時間に対する，問題のパスの遅延時間の割合は，18%であり，十分短いことが分かった．したがって，本機構は，クロック・サイクル時間に悪影響を与えないといえる．

4. 評　　価

本章では，まず評価環境，評価モデル，測定条件について述べる．次に評価結果について述べる．

4.1 評 価 環 境

SimpleScalar Tool Set Version 3.0a[3)] をベースにシミュレータを作成し，評価した．命令セットは，MIPS R10000 を拡張した SimpleScalar/PISA である．

☆ MUX の入力 0^{n-1}##result$_0$ は，result$_0$ の上位に $n-1$ 個の 0 を連結した数を表す．ここで，n は $\lceil \log_2 (\text{物理レジスタ数}) \rceil$ である．

4.2 評価モデル

評価したすべてのモデルは，通常のスーパスカラ・プロセッサと同様のパイプラインを備えている．まず，命令のフェッチを行い，デコードし，レジスタをリネームし，スケジューリングを行う．そして，実行の準備ができた命令から発行し，実行し，実行結果をレジスタに書き込む．

以下のモデルについて評価した．

- **Base モデル**：レジスタの共有を行わない，基本のモデルである．
- **No-share モデル**：レジスタの共有化の前提となるモデルである．このモデルは Base モデルと異なり命令の実行終了時に物理レジスタを割り当てる．
- **Zero モデル**：0 についてのみレジスタを共有するモデルである．
- **Binary モデル**：0 と 1 についてのみレジスタを共有するモデルである．
- **Full モデル**：すべての結果値についてレジスタを共有するモデルである．

Base モデル以外のモデルでは，仮想物理レジスタ方式を使用している．2 章で述べたように，仮想物理レジスタ方式では，ソース・レジスタのリネームの際に GMT および PMT と呼ばれる表を参照する．同時に，デスティネーション・レジスタを仮想物理レジスタにリネームし，GMT を更新する．また，命令の実行結果の書き込み時に，GMT および PMT を更新する．

Zero モデルと Binary モデルでは，3.2 節で述べたように命令の実行結果の書き込みステージで，結果値が 0 もしくは 1 であるか判定する．結果値が 0 であればつねに 0 を保持するレジスタに割り当て，1 であればつねに 1 を保持するレジスタに割り当てる．それ以外の場合は，任意の値を保持できる通常のレジスタに割り当てる．

Full モデルでは，命令の実行結果の書き込みステージで，レジスタ・ファイルを実行結果で検索する．同一の値を保持するレジスタがあればそれを共有し，そうでなければ新しく物理レジスタを割り当てる．

4.3 測定条件

各評価モデルにおいて共通する測定条件を**表 2** に示す．4.4 節で示す各モデルの IPC は，当然ながらパイプラインの各動作に割り当てるステージ段数の影響を受ける．しかし，本手法の有効性を評価する本質的な違いとしては現れない．我々は表 2 に示した仮定は，妥当なものと考える．

表 2 測定条件
Table 2 Processor parameters.

命令デコード幅	最大 8 命令
命令発行幅	最大 8 命令
命令コミット幅	最大 8 命令
命令ウィンドウ	RUU（Register Update Unit）128 エントリ LSQ（Load/Store Queue）64 エントリ
機能ユニット	8 iALU, 4 iMULT/DIV, 4 Ld/St, 6 fpALU, 4 fpMULT/DIV/SQRT
命令キャッシュ	32 KB 2 ウェイ・セット・アソシアティブ, ライン長 32 バイト, ヒット・レイテンシ 1 サイクル
データキャッシュ	32 KB 2 ウェイ・セット・アソシアティブ, ライン長 64 バイト，4 ポート, ヒット・レイテンシ 1 サイクル
2 次キャッシュ	統合, 1 MB 2 ウェイ・セット・アソシアティブ, ライン長 64 バイト, ヒット・レイテンシ 12 サイクル
分岐予測機構	gshare 17 ビット履歴 256 K エントリ PHT, 分岐予測ミス・ペナルティ 10 サイクル

一般に仮想物理レジスタ方式では，リオーダ・バッファの先頭の命令が実行された際に割り当てる物理レジスタが存在しない場合には，それ以後の命令のコミットが不可能となる．物理レジスタが解放されるのは，リオーダ・バッファから命令が削除されるときである．したがって，これ以降，物理レジスタが解放されることはないため，実行はデッドロックする．本論文では，このデッドロック回避方式として文献 7)，8) の方式（以下，**区分け方式**と呼ぶ）をベースとし，文献 14) の **DSY 方式**を組み合わせて使用した．区分け方式は次のような方式である．まず，ある数のレジスタ数を定める．これを**予約レジスタ数**と呼ぶ．リオーダ・バッファの先頭から末尾に向かって，書き込みレジスタ数の総和が予約レジスタ数を超えない最も後方のエントリを見つける．このエントリより先頭のエントリを**予約領域**と呼ぶ．予約領域のエントリに割り当てられた命令には，空き物理レジスタがある限り，結果の書き込み時に無条件に物理レジスタを割り当てる．これに対して，予約領域外のエントリに割り当てられた命令に関しては，空きレジスタ数が，予約レジスタ数よりも多い場合のみ割り当てる．このようにして，予約領域の命令に対してできるだけレジスタが割り当てられるようにする．

区分け方式は，仮想物理レジスタ方式におけるデッドロック回避手法としては十分であるが，レジスタの共有化を行うと，共有レジスタの存在により，コミット時に必ずしも物理レジスタが解放されないため，これのみでは回避できない．この問題は，DSY を組み合わせることにより，解決できる．具体的には，次の

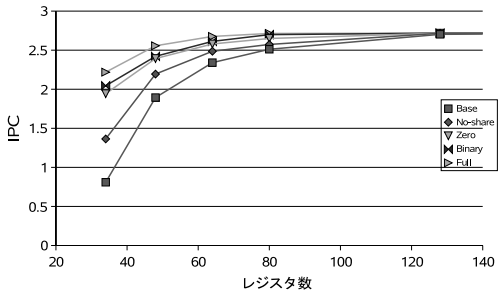

図 3 使用可能レジスタ数と IPC の関係
Fig. 3 Relation between the number of available registers and IPC.

ように DSY を使用する．リオーダ・バッファの先頭に割り当てられた命令が結果を書き込むときに，割り当てるレジスタが存在しない場合，パイプラインをストールさせる．そして，リオーダ・バッファの末尾のエントリに割り当てられた命令から先頭に向かって，命令の探索を開始する．すでに実行が終了しており，物理レジスタが割り当てられている命令のエントリが見つかったら，探索を終了する．そして，その命令からレジスタを奪い，自身に割り当てる．奪われた命令は後に再実行される．これによって，レジスタの共有化時であってもデッドロックを回避することができる．

4.4 IPC の評価

共有化によって必要レジスタ数を削減できれば，少ないレジスタで高い IPC を達成できると考えられる．本節では，使用可能なレジスタ数を実行が継続可能となる最小値から変化させ，各モデルにおける IPC を評価した．なお，実行が継続可能となるレジスタ数の最小値とは，論理レジスタ数に 1 つの命令が書き込みに使用するレジスタ数を加えた値である．

図 3 に評価結果を示す．横軸は，使用可能な整数レジスタ数であり，縦軸は，各測定における IPC の平均である．整数レジスタとして実行が継続可能となる最小値から k 個の書き込み可能レジスタを加えた場合の測定においては，浮動小数点レジスタの数は，その最小値に k を加えた数とした．

図より，基本のモデルである Base モデルでは，使用可能レジスタ数の減少にともない，最も IPC が大きく低下していることが分かる．ここで，レジスタ数が十分な数ある場合の IPC を最大 IPC とすると，Base モデルでは，レジスタ数がプログラムの実行が継続可能となる最小値の場合には，実行時に 70%もの低下となっている．

次にレジスタを共有しない No-share モデルでは，Base モデルよりは IPC の低下が緩やかであるものの，やはり大きく低下している．最少レジスタ時の最大 IPC からの低下率は 50%となっている．

一方，レジスタを共有化する Full モデルでは，低下率が 18%にとどまっている．すべての値を共有化することにより，少ないレジスタ数で高い性能を発揮できることが分かる．

Binary モデルでは，レジスタ数の減少にともない IPC は低下するものの，その度合いは Base および No-share モデルよりはるかに小さい．最少レジスタ時の最大 IPC からの低下率は 25%である．

Zero モデルは，Binary モデルとほぼ等しい結果を示していることが分かる．最少レジスタ数の最大 IPC からの低下率は 28%である．

以上より，我々の共有化手法は，レジスタ数削減に対し強い耐性を示し，完全な共有化が可能な共有化手法にほぼ匹敵する有効性を示すといえる．

4.5 レジスタ数削減効果の評価

前節の評価結果より，レジスタの共有化によって，IPC を大きく低下させることなくレジスタ数を削減できることが分かった．レジスタ数を削減できれば，アクセス時間と消費エネルギーを削減できると考えられる．そこで本節では，レジスタの共有化がアクセス時間，面積，消費エネルギーに与える影響について評価する．

4.5.1 必要レジスタ数の削減

レジスタの共有化によるレジスタ数削減効果について述べる．最大 IPC の約 90% を達成することができる Base モデルのレジスタ数 80 個の時の性能を目標性能とし，各モデルで目標性能を達成するために必要な最少レジスタ数を測定した．**図 4** に測定結果を示す．図から分かるように，No-share モデルの必要レジスタ数削減効果は小さく，Base モデルに対して 13%しか削減できない．これに対して我々の手法は，Binary モデルでは Base モデルに対して 34%，Zero モデルでも 30%もの大きな削減率を達成できることが分かった．この削減率は，Full モデルの 41%に匹敵するものである．

また，No-share モデルを基準にとると，レジスタ共有による必要レジスタ削減効果を評価することができる．Binary モデルは必要レジスタ数を約 24%削減している．これは，3.1 節で示した，0 と 1 の出現頻度の和に偶然ではあるが一致している．

4.5.2 アクセス時間の削減

レジスタ・ファイルのアクセス時間を測定した結果を示す．測定には，CACTI を修正したシミュレータを用いた．プロセス技術には 0.07 μm CMOS を仮定した．

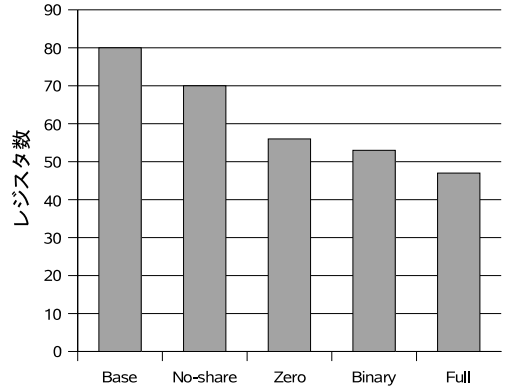

図 4 目標性能を達成するための最少レジスタ数
Fig. 4 Minimum number of registers to achieve the target performance.

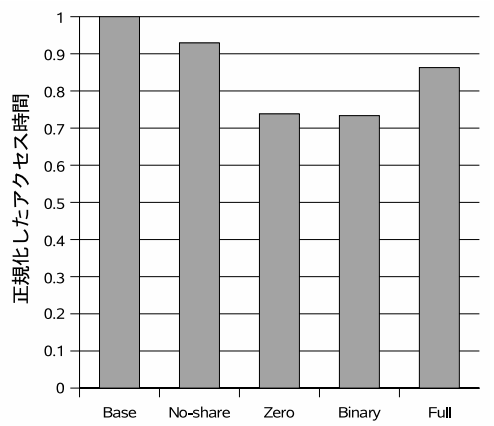

図 5 目標性能を達成するレジスタ・ファイルのアクセス時間
Fig. 5 Access time of the register file to achieve the target performance.

Full モデル以外では，レジスタ・ファイルは RAM で構成され，仮定するプロセッサでは，16 個の読み出しと 8 個の書き込みの合計 24 ポートを持つ．一方，Full モデルで必要とされるレジスタ・ファイルは，CAM で構成されなければならない．一般に，CAM セルは，RAM セルに比較器を付加したものである．RAM セルが 6 トランジスタで構成されるのに加え，通常，比較器用に 4 トランジスタを必要とする．本測定では，CACTI で仮定されているように，CAM セルの幅は RAM セルと変わらず，高さが倍になるとした．一方，ポート数については，検索値の入力用に 8 個追加する必要があるので，合計 32 ポートとした．これによっても，セル・サイズが増加する．

図 5 に，目標性能を達成するために必要なレジスタ数におけるアクセス時間を示す．縦軸は Base モデルのアクセス時間で正規化したアクセス時間である．

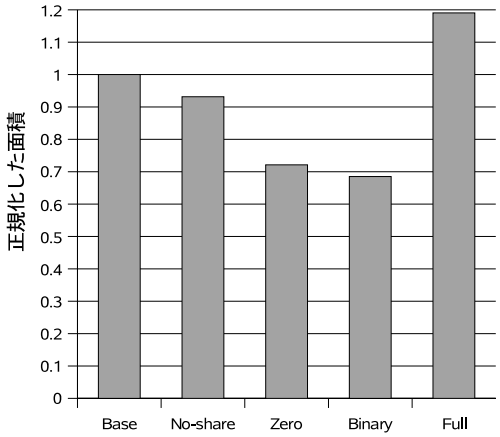

図 6 目標性能を達成するレジスタ・ファイルの面積
Fig. 6 Area of the register file to achieve the target performance.

Zero モデルおよび Binary は Base モデルに対してそれぞれ，26%，27%高速である．これに対して Full モデルは，必要レジスタ数が最も少ないにもかかわらず，Base モデルに対して 14%の高速化にとどまっている．これは，レジスタ・ファイルが他のモデルでは 24 ポートの RAM で構成されているのに対し，Full モデルでは 32 ポートの CAM で構成されているためである．

アクセス時間の短縮が，クロック・サイクル時間の短縮につながるかどうかは，プロセッサの設計に依存している．もし，レジスタ・ファイルがクリティカル・パスとなっている設計ならば影響を与え，そうでなければ一般に影響を与えない．ただし，後者の場合であっても，元の設計がパイプライン化されており，アクセス時間短縮により，パイプライン化の必要がなくなる場合がある．この場合，パイプライン短縮によるペナルティの削減が期待される．

4.5.3 面積の削減

必要レジスタ数削減によるレジスタ・ファイルの面積の削減効果について述べる．測定については，アクセス時間と同じく CACTI を使用した．**図 6** に，基準性能を達成するために必要なレジスタ数における面積を示す．縦軸は Base モデルの面積で正規化した面積である．Zero モデルは Base モデルに対して 28%，Binary モデルは 27%面積が小さい．これに対して CAM で構成されている Full モデルは必要レジスタ数が最も少ないにもかかわらず，逆に 19%増加する結果となった．

4.5.4 消費エネルギーの削減

必要レジスタ数削減によるレジスタ・ファイルの消費エネルギーの削減効果について述べる．測定については，アクセス時間と同じく CACTI を使用した．

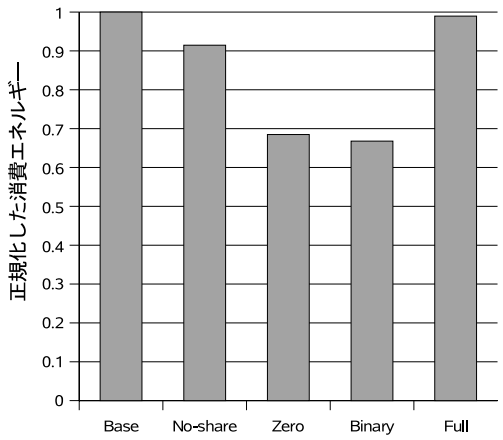

図 7 目標性能を達成するレジスタ・ファイルの消費エネルギー
Fig. 7 Energy consumption of the register file to achieve the target performance.

図 7 に，目標性能を達成するために必要なレジスタ数における消費エネルギーを示す．縦軸は Base モデルの消費エネルギーで正規化した消費エネルギーである．Zero モデルは Base モデルに対して 32%，Binary モデルは 33%消費エネルギーが低い．これに対して CAM で構成されている Full モデルは，わずか 1%の削減にとどまった．

5. 関 連 研 究

物理レジスタの生存期間を短縮することにより，必要レジスタ数を削減する手法が提案されている．物理レジスタの割当てを命令の実行終了時まで遅らせることにより実現する手法が，文献 7)，8)，14) で提案されている．また，論理レジスタを最後に参照する命令の情報をコンパイラがハードウェアに与えることにより生存期間を最短にする手法が，文献 13) で提案されている．

レジスタ・ファイル階層化によりアクセス時間を短縮する方式が文献 4) で，バンク化によるアクセス時間を短縮する方式が文献 2)，5) で提案されている．我々の方式とこれらの方式は，組み合わせてさらに大きな効果を得ることができる．

レジスタ・ファイルのポート数を削減する手法として，文献 2)，17) では，バイパスにより得られるレジスタ値についてはレジスタ・ファイル・アクセスが不要であることを利用する手法が提案されている．文献 17) ではレジスタ・ファイルのバンク化を利用して書き込みポートを削減する方法を提案している．また，Compaq Alpha21264[11] ではレジスタ・ファイルの複製によりポート数を削減している．これらの方式も，我々の方式と組み合わせることができる．

頻出する値を圧縮して保持する別キャッシュを用意することにより，キャッシュのヒット率を向上させる提案が，文献 25) で提案されている．また，頻出する値を圧縮して保持することにより，キャッシュの消費エネルギーを削減する提案が，文献 23) でなされている．これらの研究は，我々と同じく頻出値を利用しているが，我々の適用の対象はキャッシュではなく，レジスタ・ファイルである点が異なる．

頻出する値についてのみ予測することにより，値予測器のコスト対性能比を向上させる提案が，文献 20) で提案されている．これも我々と同じく，頻出値を利用している．

入力が等しく結果を再利用可能な命令間でレジスタを共有化する提案が，文献 18) でなされている．彼らの方法では，入力から共有を判断するのに対し，我々は出力結果から共有を判断している．

我々の提案に遅れて，頻出値のみを共有化する手法が文献 1) で提案されている．この手法では実行終了時ではなく，デコード時に物理レジスタ割当てを行う．そして，実行終了時に結果値が共有値であれば，割り当てられたレジスタを解放し，共有レジスタに割り当てている．機構としては，我々の手法とは異なり，仮想物理レジスタ方式が必要ないという利点がある．しかし，物理レジスタの生存期間が長くなり，仮想物理レジスタ方式より多くのレジスタを要する．また，共有レジスタへの再割当てのために，レジスタマップ表により多くのポートを必要とし，マップ表のアクセス時間，消費電力が増加するという欠点がある．

6. ま と め

本論文では，必要レジスタ数を削減するアプローチとして，複数の命令間でレジスタを共有する手法に着目し，これを実現する手法を提案した．本手法は，命令実行結果には頻出する値が存在することを利用し，それらを共有するものである．本手法は，レジスタ・ファイルが複雑化する従来の共有化手法と異なり，ハードウェアをほとんど複雑化させることなく実現できる．

SPECint95 ベンチマークを使い評価を行った．デコード時にレジスタを割り当てている通常の方式と，ライトバック時にまでレジスタ割当を遅らせる方式では，使用可能なレジスタ数をプログラムを実行可能な最少数にすると，レジスタが十分に存在する場合に対し，IPC はそれぞれ 70%，50%もの大きな低下を示した．これに対して本手法では，IPC 低下を 0 と 1 を共有する方式で 25%，0 のみを共有する方式で 28%と大

きく抑制できた．さらに，レジスタが十分にある場合の IPC の約 90%を達成できるレジスタ数を比較した場合，デコード時にレジスタを割り当てる方式に対し，ライトバック時にレジスタを割り当てる方式では 13%しかレジスタ数を削減できなかった．これに対して我々の方式では 34%もの大きなレジスタ数削減を達成できることが分かった．また，従来の共有化手法ではアクセス時間は 14%しか削減することができなかったが，我々の手法では 27%もの大きなアクセス時間削減を達成することができた．また，従来の共有化手法では面積および消費エネルギーを小さくすることはできないことが分かった．これに対して，我々の手法ではデコード時にレジスタを割り当てる方式に対してアクセス時間を 31%，消費エネルギーを 33%小さくすることができることを確認した．

近年，ハイエンドのプロセッサでは，同時に複数のスレッドを実行する SMT（Simultaneous Multi-Threading）[22] が注目され，一部のプロセッサでは実際に採用されている[6),12)]．SMT では，複数のスレッドのコンテキストを同時に保持するために，単一スレッド実行のプロセッサに比べ，非常に大きなレジスタ・ファイルを必要とする．今後も SMT を採用するプロセッサは増加し，また，現在よりさらに多くのスレッドを扱えることが要求されるようになると思われる．このため，レジスタ・ファイル・サイズ増加の問題は，今後さらに大きくなる．本論文では，単一スレッド実行のプロセッサを仮定したが，SMT でも同様の効果を得ることができる．レジスタ・ファイルが非常に大きい SMT では，我々の手法の重要性は，より大きくなると考えられる．

謝辞 本研究の一部は，文部科学省科学研究費補助金基盤研究（C）課題番号 15500036，文部科学省 21 世紀 COE プログラム，財団法人栢森情報科学振興財団研究助成の支援により行った．

参 考 文 献

1) Balakrishnan, S. and Sohi, G.S.: Exploiting Value Locality in Physical Register Files, *Proc. 36th Annual International Symposium on Microarchitecture* (2003).
2) Balasubramonian, R., Dwarkadas, S. and Albonesi, D.H.: Reducing the Complexity of the Register File in Dynamic Superscalar Processors, *Proc. 34th Annual International Symposium on Microarchitecture*, pp.237–248 (2001).
3) Burger, D. and Austin, T.M.: The SimpleScalar Tool Set Version 2.0, Technical Report CS-TR-97-1342, University of Wisconsin (1997).
4) Cruz, J.L., González, A., Valero, M. and Topham, N.P.: Multiple-Banked Register File Architectures, *Proc. 27th Annual International Symposium on Computer Architecture*, pp.316–325 (2000).
5) Farkas, K.I., Chow, P., Jouppi, N.P. and Vranesic, Z.: The Multicluster Architecture: Reducing Cycle Time Through Partitioning, *Proc. 30th Annual International Symposium on Microarchitecture*, pp.149–159 (1997).
6) Glaskowsky, P.N.: IBM Raises Curtain on Power5, 10/14/03-01 (2003).
7) González, A., González, J. and Valero, M.: Virtual-Physical Registers, *Proc. 4th International Symposium on High Performance Computing*, pp.175–184 (1998).
8) González, A., Valero, M., González, J. and Monreal, T.: Virtual Registers, *Proc. International Conference on High Performance Computing*, pp.364–369 (1997).
9) Horowitz, M.A.: Timing Model for MOS Circuits, Technical Report SEL 83-003, Integrated Circuits Laboratory (1983).
10) Jourdan, S., Ronen, R., Bekerman, M., Shomar, B. and Yoaz, A.: A Novel Renaming Scheme to Exploit Value Temporal Locality through Physical Register Reuse and Unification, *Proc. 31st Annual International Symposium on Microarchitecture*, pp.216–225 (1998).
11) Kessler, R.E.: The Alpha 21264 Microprocessor, *IEEE Micro*, Vol.19, No.2, pp.24–36 (1999).
12) Krewell, K.: Intel Embraces Multithreading, *Microprocessor Report*, Vol.16, No.9, pp. 1–5 (2001).
13) Lo, J.L., Parekh, S.S., Eggers, S.J., Levy, H.M. and Tullsen, D.M.: Software-Directed Register Deallocation for Simultaneous Multithreaded Processors, *IEEE Trans. Parallel and Distributed Systems*, Vol.10, No.9, pp.922–933 (1999).
14) Monreal, T., González, A., Valero, M., González, J. and Viñals, V.: Delaying Physical Register Allocation Through Virtual-Physical Registers, *Proc. 32nd Annual International Symposium on Microarchitecture*, pp. 186–192 (1999).
15) 大熊 穣，片山清和，小林良太郎，安藤秀樹，島田俊夫：最近の値の局所性に着目した共有化による物理レジスタ削減，情報処理学会研究報告，Vol.2002-ARC-149, pp.73–78 (2002).

16) 大熊 穣, 片山清和, 小林良太郎, 安藤秀樹, 島田俊夫：頻出値を利用した物理レジスタの静的共有化手法, 2003 年先進的計算基盤システムシンポジウム SACSIS 2003, pp.291–298 (2003).
17) Park, I., Powell, M.D. and Vijaykumar, T.N.: Reducing Register Ports for Higher Speed and Lower Energy, *Proc. 35th Annual International Symposium on Microarchitecture*, pp.171–182 (2002).
18) Petric, V., Bracy, A. and Roth, A.: Three Extensions To Register Integration, *Proc.35th Annual International Symposium on Microarchitecture*, pp.37–47 (2002).
19) Reinman, G. and Jouppi, N.P.: CACTI 2.0: An Integrated Cache Timing and Power Model, WRL Research Report 2000/7, COMPAQ Western Research Laboratory (2000).
20) Sato, T. and Arita, I.: Low-Cost Value Predictors Using Frequent Value Locality, *Proc. 4th International Symposium on High Performance Computing*, pp.106–119 (2002).
21) Shivakumar, P. and Jouppi, N.P.: CACTI 3.0: An Integrated Cache Timing, Power, and Area Model, WRL Research Report 2001/2, COMPAQ Western Reseach Laboratory (2001).
22) Tullsen, D.M., Eggers, S. and Levy, H.M.: Simultaneous Multithreading: Maximizing On-Chip Parallelism, *Proc. 22nd Annual International Symposium on Computer Architecture*, pp.392–403 (1995).
23) Villa, L., Zhang, M. and Asanović, K.: Dynamic Zero Compression for Cache Energy Reduction, *Proc. 33rd Annual International Symposium on Microarchitecture*, pp.214–220 (2000).
24) Wilton, S.J.E. and Jouppi, N.P.: An Enhanced Access and Cycle Time Model for On-Chip Caches, WRL Research Report 93/5, Digital WRL Research Laboratory (1994).
25) Zhang, Y., Yang, J. and Gupta, R.: Frequent Value Locality and Value-Centric Data Cache Design, *Proc. 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp.150–159 (2000).

(平成 16 年 2 月 1 日受付)
(平成 16 年 7 月 8 日採録)

山本 哲弘（学生会員）

1981 年生．2000 年名古屋市立名東高校卒業．2004 年名古屋大学工学部電気電子・情報工学科卒業．同年名古屋大学大学院工学研究科電子情報システム専攻博士課程前期課程に進学．計算機アーキテクチャの研究に従事．

大熊 穣

2001 年名古屋大学工学部電子情報学科卒業．2003 年名古屋大学大学院工学研究科電子情報学専攻博士課程前期課程修了．同年トヨタ自動車株式会社に入社．

片山 清和（正会員）

1994 年名古屋大学工学部情報工学科卒業．1996 年名古屋大学大学院工学研究科情報工学専攻博士課程前期課程修了．1999 年名古屋大学大学院工学研究科電子情報学専攻博士課程後期課程満了．同年より名古屋大学大学院工学研究科研究生．2004 年四日市大学経済学部経営学科講師．計算機アーキテクチャの研究に従事．

小林良太郎（正会員）

1995 年名古屋大学工学部電子情報学科卒業．1997 年名古屋大学大学院工学研究科電子情報学専攻博士課程前期課程修了．2000 年名古屋大学大学院工学研究科電子情報学専攻博士課程後期課程満了．工学博士．2000 年名古屋大学大学院工学研究科電子情報学専攻助手．1999 年情報処理学会山下記念研究賞受賞．2002 年情報処理学会論文賞受賞．計算機アーキテクチャの研究に従事．

安藤 秀樹（正会員）

1959 年生．1981 年大阪大学工学部電子工学科卒業．1983 年大阪大学大学院修士課程修了．京都大学工学博士．1983 年三菱電機（株）LSI 研究所．ISDN 用ディジタル信号処理 LSI，第 5 世代コンピュータ・プロジェクトの推論マシン用プロセッサの設計に従事．1991 年 Stanford 大学客員研究員．1997 年名古屋大学大学院工学研究科電子情報学専攻講師．1998 年名古屋大学助教授．1998 年〜2001 年東京大学大学院理学系研究科助教授併任．2004 年名古屋大学教授．1998 年，2002 年情報処理学会論文賞受賞，電子情報通信学会/情報処理学会 2003 年先進的計算基盤システムシンポジウム優秀学生論文賞．計算機アーキテクチャ，コンパイラの研究に従事．ACM，IEEE，電子情報通信学会会員．

島田 俊夫（正会員）

1968 年東京大学工学部計数工学科卒業．1970 年東京大学大学院修士課程修了．同年電子技術総合研究所入所．1993 年より名古屋大学大学院工学研究科電子情報学専攻教授．人工知能向き言語，LISP マシン，データフロー計算機の研究に従事．最近はマイクロプロセッサのアーキテクチャやチップ内並列処理の研究を行っている．1988 年度市村賞，1994 年度情報処理学会論文賞，1995 年注目発明，2002 年度情報処理学会論文賞受賞．工学博士．