2004－ARC－159　(3)
2004／7／30

ビット分割によるレジスタファイルサイズ削減手法

近藤正章[†]　中村宏[†]

動的命令スケジューリング，多命令同時発行を行うプロセッサにおいては，大容量・多ポートのレジスタファイルが必要不可欠である．近年では，さらなる高速化のためにレジスタサイズやポート数が増加傾向にある．しかし，これはレジスタアクセス時間や消費電力の増大といった問題を引き起こす．そこで，本稿ではレジスタファイルサイズの削減を目的として，ビット分割レジスタファイルを提案する．提案するレジスタ構成を用いることで，小容量のもとでも高性能が達成できる．本稿は，ビット分割レジスタファイルのためのマイクロアーキテクチャの拡張と，提案手法の評価について述べる．

Reducing Register Space by Bit-Partitioning

Masaaki Kondo[†] and Hiroshi Nakamura[†]

A large multi-ported register file is indispensable for exploiting instruction level parallelism in the recent dynamically scheduled superscalar processors. However, such a large register file causes problems of long access delay and huge power consumption.

To tackle these problems, in this paper, we propose *Bit-Partitioned Register File* to reduce required register file size. We show the basic idea and mechanism of proposed register file and evaluation results on performance and power consumption. Evaluation results reveal that the proposed register file achieves higher IPC even with small register file.

1. はじめに

動的命令スケジューリング・多命令同時発行を行うマイクロプロセッサでは，命令レベル並列性 (ILP) を活用するために，大容量かつ多ポートのレジスタファイルが必須である．近年では，さらなる ILP 活用のために命令ウィンドウサイズや同時発行命令数を増加させる傾向にあるが，それにともないレジスタファイルのサイズ・ポート数も増加している．しかし，このレジスタファイルの大容量・多ポート化は，アクセス時間や消費電力の増大という問題を引き起こす．

これらの問題への対処を目的として，アクセスすべきレジスタのサイズ・ポート数を削減するための手法がこれまでにも多く提案されている．たとえば，要求されるサイズ削減のために，物理レジスタファイルの割り当てを遅らせる手法[1),2)] や，同じ値を持つオペランドを 1 つのレジスタエントリで共有化する手法[3),4)]，階層化レジスタファイル[5),6)] などが提案されている．また，ポート数を削減するものとしては，マルチバンク化[5),7)] やクラスタ型マイクロアーキテクチャ[8),9)] などが良く知られる手法である．

本稿では，必要とするレジスタファイルサイズの削減を目的として，ビット分割レジスタファイル (*Bit-Partitioned Register file: BPRF*) と呼ぶレジスタファイル構成を提案する．BPRF は，多くのレジスタオペランドがデータパスの全ビット幅分を必要としていないという事実に基づくものである．

一般的にワークステーションやパーソナルコンピュータに搭載されるマイクロプロセッサは 32 ビット，あるいは 64 ビット幅のデータパスを持ち，32 または 64 ビット演算を行うことが可能である．しかし，実際のプログラム実行の際には，多くのオペランドはそれらのビット幅よりも有効ビット幅が小く，上位ビットが全て 0 (負の値の場合は 1) であることが多い．64 ビットプロセッサにおいて SPECint95 ベンチマークを実行した場合に，整数命令のうちのおよそ 50%が 16 ビット以下の演算であるという報告もある[10)]．

従来のプロセッサでは，有効ビット幅が小さなデータに対しても，フルビット幅のレジスタエントリを割り当てるため，それらのエントリの上位ビット部分は無駄に使われていた．そこで，提案する BPRF ではレジスタファイルをビット方向に分割し，ビット幅は小さくなるがエントリ数を増やすことで，レジスタの記憶領域の有効活用を狙うものである．有効ビット幅が分割後のレジスタのビット幅よりも小さなオペランドに対しては，1 エントリのみを割り当てる．一方，有効ビット幅が大きなオペランドに対しては，複数エントリを用いてデータの記憶を行なう．

BPRF を用いることで，有効ビット幅の小さいデータが多い場合，従来構成のレジスタファイルに比べて，同容量でもより多くのオペランドを保存することが可能となり，高い IPC 性能を達成できる．また逆に，同じ IPC 性能を得るためのレジスタ容量を削減することができるとも考えられるため，結果としてレジスタ

† 東京大学 先端科学技術研究センター
Research Center for Advanced Science and Technology, The University of Tokyo

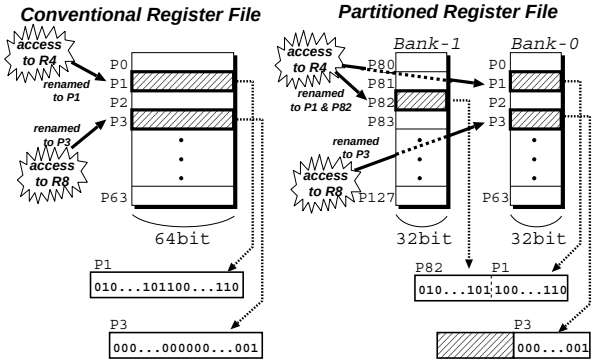

図 1 BPRF の概要

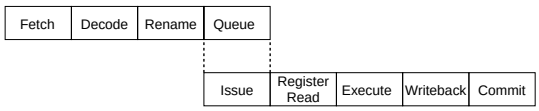

図 2 従来型プロセッサのパイプライン

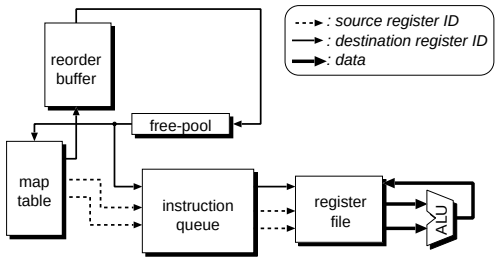

図 3 従来型プロセッサのブロック図

アクセス時間や消費電力の削減効果も期待できる．

本稿では，提案する BPRF の概要とマイクロアーキテクチャの実装について述べ，性能，およびレジスタアクセス時間や消費エネルギーの評価を行なう．

2. ビット分割レジスタファイル

図 1 に BPRF の概要を示す．図は，64 ビットデータ幅，64 エントリの従来のレジスタファイルを，ビット方向に 2 分割した場合を示している．本稿では，分割されたレジスタファイルの各ブロックをバンクと呼び，オペランドの 64 ビットワードを 2 つに分割した場合の各部分をサブワードと呼ぶ．ここで，ビット方向に分割しても，分割後の各バンクのエントリ数は 64 で変らず，両バンク合計で 128 エントリとなる (図では説明のために，エントリの番号を P0 から P127 までの通し番号で表している)．従って，従来のレジスタに比べ，同じ記憶容量のもとでエントリ数が増えることになる．

この BPRF を用い，オペランドの有効ビット幅に合わせて必要なエントリだけを割り当てることで，レジスタファイルの記憶領域を有効に利用できる．一方で，64 ビットデータも複数エントリを用いて記憶できるため，プロセッサの論理的な動作には影響を与えない．

BPRF の基本的なアイデアを，図 1 のレジスタアクセスの様子を用いて述べる．図において，アーキテクチャレジスタ R4 のオペランドの値は有効ビット幅が大きく，上位および下位サブワードともに有効な値を持ち，R8 のオペランドの値は有効ビット幅が小さく，下位サブワードのみに有効な値を持つものとする．従来のレジスタファイルでは，アーキテクチャレジスタ R4 が物理レジスタ P1 に割り当てられ，R8 が P3 に割り当てられている．ここで，R8 のオペランドは有効ビット幅が小さく，上位サブワードが全て 0 であるにも関わらず，64 ビット幅のエントリが割り当てられる．そのため，R4 と R8 の両オペランドを保存するためには，128 ビット分の記憶領域が必要となる．一方 BPRF では，アーキテクチャレジスタ R4 は物理レジスタバンク 0 の P1，およびバンク 1 の P82 に割り当てられ，R8 はバンク 0 の P3 のみに割り当てられている．したがって，両オペランドを記憶するためのレジスタ記憶領域は 96 ビットとなる．このように，BPRF を用いることで，同じオペランドを保存する場合の必要とされるレジスタ記憶領域を小さくすることができる．なお，上位ビットが全て”1” である場合も有効ビット幅が小さいと考えることができるが，本稿では以降，”0” の場合のみを対象とする．

ここで，BPRF においてバンク 0 とバンク 1 にオペランドをバランス良く割り当てるために，各演算命令，あるいはロード命令のデスティネーションレジスタ毎に，上位サブワードと下位サブワードに割り当てるバンクを変更する．この目的で，*Least Significant Bnak Pointer: LSBP* と呼ぶフラグを新たに導入する．LSBP は，各オペランド毎に，どのバンクがオペランドの下位サブワードに割り当てられているかを示すものである．

3. マイクロアーキテクチャ

本節では，BPRF のアイデアを実装するためのマイクロアーキテクチャの拡張例について述べる．なお，前提とするマイクロアーキテクチャとしては Alpha21264[9] や MIPS R10000[11] の構成をベースとし，物理レジスタ構成方式は *merged architectural and rename register file* 方式[12] である．また，レジスタリネーミングの機構としては，RAM ベースのリネーミング機構を仮定する．

3.1 従来型プロセッサ

図 2，図 3 に，従来型プロセッサのパイプライン，およびブロック図を示す．以下，このプロセッサの動作について簡単に説明する．

まず，*Fetch*，*Decode* された命令のソースレジスタ ID は，*Rename* ステージにおいて，*map table* を参照することで物理レジスタ ID に変換される．また，それと同時にその命令のデスティネーションレジスタとして新しい物理レジスタエントリが *free-pool* から割り当てられる．*free-pool* から取得された物理レジス

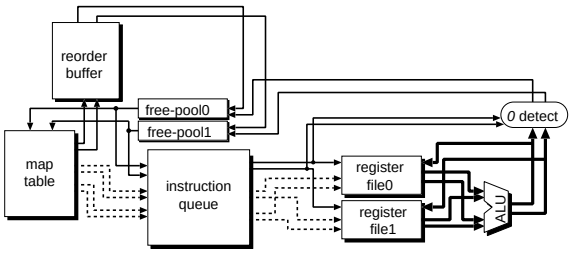

図 4 BPRF 用の拡張を行なったブロック図

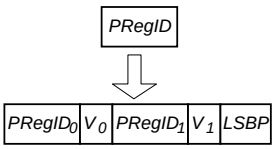

図 5 物理レジスタ番号フィールドの拡張

タ ID は，デスティネーションレジスタ ID に対応する *map table* エントリに登録される．

また，各命令の *Commit* を命令順序通りに行なうため，*Rename* ステージでは全ての命令の情報が *reorder buffer* に命令順序通りに記憶される．この際，命令のデスティネーションレジスタ ID と，それ以前にその ID に割り当てられていた物理レジスタ ID が *reorder buffer* に登録される．命令が *Commit* される際には，*reorder buffer* の該当エントリに登録されていた物理レジスタエントリが解放され *free-pool* に戻される．

各命令は，*Rename* ステージで得られたソース・デスティネーション物理レジスタ番号と共に，*Queue* ステージで命令キュー (*instruction queue*) に登録され，すべてのオペラドが揃い次第，発行 (*Issue*) される．発行された命令は，レジスタの読み込み (*Register Read*) が行なわれ，ALU において演算が実行される (*Execute*)．演算結果は *Writeback* ステージでレジスタに書き込まれる．

3.2 BPRF の拡張

3.2.1 概　　要

図 4 に，BPRF の拡張を行なった場合のプロセッサのブロック図を示す．図は，従来のレジスタファイルを 2 バンクに分割した場合である．なお，パイプライン構成は図 2 と同様である．

本拡張では，*Rename* ステージにおいて，各命令のデスティネーションレジスタに対し，2 バンク分の物理レジスタエントリを割り当てる．これは，*Rename* ステージでは，その命令のオペランドの有効ビット幅が分からないためである．*map table* や命令キューなど，物理レジスタ番号を保存するためのフィールドは，図 5 のように 2 バンク分の ID が保存できるように拡張される．$PRegID_X$ は，それぞれバンク X 用の物理レジスタ ID であり，V_X は，このレジスタエントリが有効かどうかを示す valid ビットを表している．valid ビットがセットされている場合，このエントリのオペランドは有効な値を保持しており，セットされていない場合は有効な値を保持していない，つまり全ビットが 0 であることを意味する．さらに，どちらのバンクが下位ビットであるかを示す LSBP も追加される．また，*free-pool* もバンク毎に設けられる．

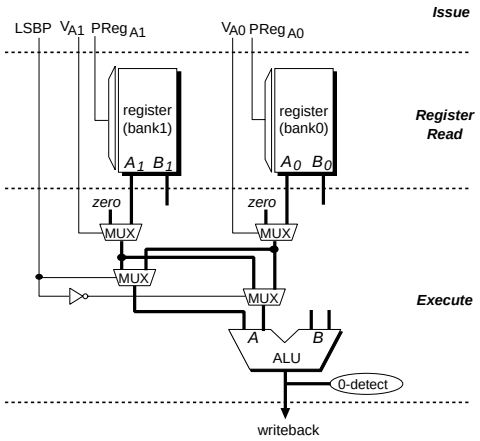

図 6 レジスタから ALU へのデータパス

発行された命令は，ソースオペランドを 2 つのバンクのレジスタから読み込み ALU で実行される．実行結果は *0-detect* 機構により，有効ビット幅が判定される．実際には，各サブワード毎にそのビットがすべて 0 かどうかが判定され，すべて 0 であった場合はそのサブワードに割り当てられていた物理レジスタエントリが Writeback ステージで解放され，その ID が該当バンクの *free-pool* に書き込まれる．

一般的に物理レジスタエントリの解放は，その物理レジスタと結びつけられたアーキテクチャレジスタ ID をデスティネーションレジスタに持つ次の命令が *Commit* した時点で行なわれる．ここで，本拡張のように物理レジスタのエントリを早い段階で解放することを *Early Register Deallocation (ERD)* と呼ぶ．ERD によりそのレジスタエントリを別の命令のデスティネーションレジスタとして用いることができるため，レジスタの使用効率が向上し，結果として，小容量のレジスタサイズでも高性能を達成することができる．この ERD を行なうためのマイクロアーキテクチャ的な拡張については 3.2.3 節で述べる．

3.2.2 レジスタから ALU へのデータパスの拡張

図 6 にレジスタから ALU へのデータパスを示す☆．まず，バンク 0，バンク 1 のレジスタファイルが $PRegID_{A0}$，$PRegID_{A1}$ によって並列にアクセスされる．次段にあるマルチプレクサは，それぞれのバンク用の valid ビット V_{A0}，V_{A1} がセットされていればリードされた値を，そうでなければ全ビット 0 の値を選択する．さらに，次のマルチプレクサでは，LSBP に依存して，上位サブワードと下位サブワードとの入れ替えを行なうためのものである．

上記の動作は，従来のプロセッサにはなかったもの

☆ フォワーディングのパスなどは省略している．

であり，この追加のロジックはクリティカルパス遅延を増加させてしまう可能性がある．しかし，この動作は，図のように *Execute* ステージにて行なうことができる．現在のプロセッサにおけるクリティカルパスは，*Issue* ステージや *Regiter Read* ステージであることが多く，*Execute* ステージへの多少のロジック追加はプロセッサの周波数に悪影響は及ぼさないと考えられる．さらには，もともと ALU に備わっている符合拡張やオペラドシフトロジックを拡張することで上述の動作を行なえるため，追加ロジックによるハードウェア複雑さの増大は大きな問題にならないと考えられる．

3.2.3 Early Register Deallocation

これまでにも，ERD を用いたレジスタ使用効率の改善手法に関する提案が行なわれており[4),13),14)]，本拡張においても同じように ERD を行なうことができると考えられる．ここでは詳細については省略するが，具体的には以下の拡張が必要である．

- *map-table* の該当レジスタエントリの無効化
- 命令キューの該当レジスタエントリの無効化
- *Commit* の際にすでに解放されているレジスタエントリの二重解放を防ぐための制御
- *free-pool* への物理レジスタ ID の書き込み

4. 評　　価

4.1 評価環境

提案する BPRF による性能を調べるため，SimpleScalar Tool Set[15)] を用いたサイクルレベルシミュレーションにより評価を行なう．なお，3.2 節で述べたプロセッサ構成を評価できるよう，SimpleScalar のマイクロアーキテクチャを大幅に変更している．

評価プログラムは，SPEC CPU2000 の整数ベンチマークから 253.perlbmk 以外の全てのプログラム，および MediaBench[16)] から adpcm，epic，g721，mpeg2 (それぞれエンコードとデコード) を用いる．コンパイラは，Alpha 用の命令セットを生成する DEC C コンパイラを用い，オプションは “-arch ev6 -fast -O4 -non_shared” である．なお，SPEC CPU2000 ベンチマークには *ref* インプットセットを用い，最初の 10 億命令実行後の 200 万命令を評価した．また，MediaBench の各プログラムはプログラム実行の最初から最後までを評価した．

4.2 評価の仮定

表 1 に評価におけるプロセッサの仮定を示す．

なお，*0-detect* は ALU の演算結果だけでなく，ロード命令のオペランドにも適用するものとして評価を行なう．

5. 評 価 結 果

5.1 性　　能

まず，従来型のレジスタファイルおよび BPRF において，レジスタファイルサイズが性能に与える影響

表 1　評価における仮定

Data path width	64 bit
Fetch, Decode,Commit width	8
Branch prediction	bimodal 2Ktable
BTB	512sets, 4way
Mis-Prediction penalty	3cycles
Instruction queue size	
- integer	64
- load/store	64
- floating-point	64
Issue width	
- integer	8
- load/Store	4
- floating-point	4
L1 I-Cache	32KB, 32B line, 2way 1 cycle latency
L1 D-Cache	32KB, 32B line, 2way 2 cycle latency
L2 unified Cache	512KB, 64B line, 8way 10 cycle latency
Memory latency	64 cycle
Bus width	8B
Bus clock	1/5 of processor core

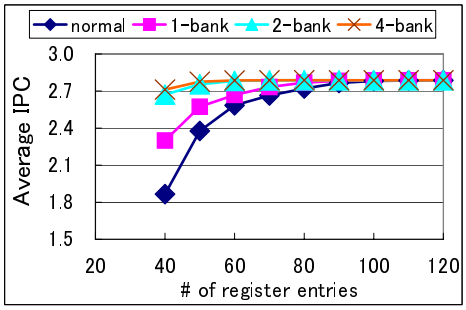

図 7　IPC (全プログラム平均)

を調べるために，図 7 にレジスタファイルサイズを変化させた場合の，評価に用いた全プログラムの平均 IPC を示す．図中の *normal* は従来型のプロセッサを表し，*1-bank*，*2-bank*，*4-bank* は，BPRF においてそれぞれ何バンクに分割したかを表している．ここで，1-bank の場合は実際には分割をしないが，演算結果のオペランドの 64 ビット全てが 0 であった場合に，ERD を行なった場合を示している．また，横軸のレジスタファイルサイズは，各バンクのエントリ数である．

図 7 の結果より，全ての場合で，レジスタサイズが増加するにつれて IPC が向上しているのがわかる．これは，レジスタエントリが多い場合，命令キューにおいて発行するべき候補の命令数が増え，より ILP が活用できるためである．レジスタサイズが十分でないと，*Rename* ステージにおいて *free-pool* からデスティネーションレジスタに割り当るべきエントリが無くなり，割り当て可能なエントリが確保できるまで *Queue* ステージ以前のパイプラインをストールさせなければならない．その結果，少ないレジスタエントリでは

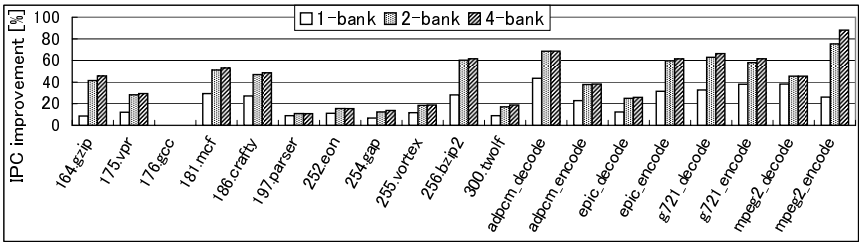

図 8 IPC 向上率 (# of register entries for each bank = 50)

表 2 レジスタのアクセス時間と消費エネルギー

#entries	Time [ns]	Energy(1bank/2bank/4bank) [nJ]
40	1.389	5.888 / 6.101 / 6.526
50	1.469	6.616 / 6.876 / 7.396
60	1.548	7.345 / 7.651 / 8.265
70	1.732	8.123 / 8.528 / 9.336
80	1.888	8.858 / 9.316 / 10.233
90	1.980	9.593 / 10.106 / 11.130
100	2.075	10.330 / 10.895 / 12.026
110	2.174	11.065 / 11.683 / 12.923
120	2.357	11.800 / 12.473 / 13.820

ILP が十分に活用できなくなる．

しかし，ある程度までレジスタサイズが増えると，IPC は飽和し，すべての場合でほぼ同じ IPC に収束する．これは，レジスタサイズが性能上のボトルネックでなくなった時には，他の機構 (命令キューサイズなど) の制限から，それ以上 IPC が向上しなくなるためである．

ここで，注目すべき点は，BPRF では normal に比べて少ないレジスタサイズでも高性能を達成できる点である．normal や 1-bank では 90 あるいは 80 エントリ付近で最高 IPC に到達するのに対し，2-bank あるいは 4-bank では 50 エントリ程度で同じ IPC を達成できる．BPRF ではレジスタを分割し，有効ビット幅の小さいオペランドが検出された時点で必要のないレジスタエントリを解放するため，より多くのエントリを新たなオペランドのために用いることができ，効率的にレジスタを使うことができた結果である．

次に，プログラム毎の結果を見るため，ベンチマークプログラム毎の normal 構成に対する性能向上率を図 8 に示す．なお，2-bank 構成でほぼ性能が飽和するレジスタサイズ 50 エントリの場合を示している．

図より，`176.gcc` を除く全てのベンチマークで，BPRF を用いることで IPC が向上していることがわかる．特に，MediaBench のアプリケーションでは性能向上率が高い．メディア系のアプリケーションでは，処理すべきデータの有効ビット幅が小さいため，BPRF が有効に機能した結果である．また，1-bank に比べ 2-bank において性能向上率が大きくなるアプリケーションが多い．この点からも，ビット分割による利点の大きさが伺える．

5.2 アクセス時間および消費電力

表 2 に，各レジスタファイルサイズにおけるアクセスタイム，消費エネルギーを示す．アクセス時間は CACTI-3.2[17] を，消費エネルギーは Wattch[18] を拡張して評価を行なった．本結果は，0.18μm プロセスを用い，16 リード/8 ライトポートの場合のデータを示している．アクセスタイムに関しては，ビット方向に分割してもそれほど変化はないため，1 バンク構成 (64 ビット幅) の結果のみを示す．なお，normal と 1-bank ではレジスタファイルの構成は同じである．

結果より，レジスタサイズ (#entries) が増えるごとにアクセスタイム，および消費エネルギーが増加していることがわかる．また，多くのバンクに分割することで消費エネルギーが増加する傾向にあるが，これは各バンク毎にアドレスデコードの回路が必要であり，それによる消費エネルギーが増えるためである．

前節の評価で，同じ IPC を達成するためには normal の場合で 90 エントリ，1-bank では 80 エントリ，また 2-bank (あるいは 4-bank) では 50 エントリが必要であった．90 エントリの 1-bank 構成に比べ，50 エントリの 2-bank 構成では，アクセス時間が 26%，消費エネルギーは 28%ほど削減されている．また，80 エントリの 1-bank 構成と比べても，50 エントリの 2-bank の場合ではアクセスタイム，および消費エネルギーがそれぞれ 22%，26%程度削減されている．このことより，提案する BPRF を用いることで，IPC 性能を低下させずに，レジスタアクセス時間や消費エネルギーを大きく削減できることがわかる．

6. 関 連 研 究

従来より，レジスタファイルの大容量・多ポート化によるアクセス時間や消費電力の増大の問題への対処を目的として，アクセスすべきレジスタの容量・ポート数を削減するための手法が多く提案されている．また，本研究と同じ視点から，オペランドのビット幅を考慮したレジスタファイルの効率的使用に関する研究も行なわれている[14),19),20)]．

文献[14)] では，有効ビット幅が小さいオペランドを検出し，有効ビット幅がある閾値よりも小さいオペランドは，レジスタリネーミング用の *map-table* の物理レジスタ ID フィールドに保存し，そのオペランド用にエントリを割り当てないことで，レジスタファイルを効率的に使用する手法を提案している．文献[19)] では，従来のレジスタファイル中にはオペランドの上位

ビットの値が同じで，下位ビットのみが違うオペランドが多数存在することに着目し，上位ビットをそれらのオペランドで共有して保存することで，レジスタを有効利用する手法を提案している．文献[20]では，一部のポートのみが全ビットにアクセスできる一方，他のポートは下位ビット部分のみアクセス可能なレジスタファイル構成を提案している．上位ビット部分で消費されるエネルギーを削減することが目的である．

また，組み込み分野の LSI 設計時に，有効ビット幅を意識して，レジスタだけではなくデータパス全体を最適化する手法も提案されている[21]．

さらには，コンパイラがビット幅を意識することで，レジスタを有効利用する手法もある．たとえば，MIPS の命令セットアーキテクチャでは，浮動小数点データに関し，単精度 (32 ビット幅) の場合は 32 個のレジスタエントリが使用可能であるが，倍精度 (64 ビット幅) の場合は連続する 2 エントリを統合してオペランドを保存するように (偶数番のレジスタ ID のみを用いる) 命令の生成が行なわれる．

本稿で提案する BPRF は，動的に有効ビット幅を判定することで，たとえば汎用プロセッサのように，実行するデータに依存して有効ビット幅が変るような場合にも対応できることが利点である．

7. まとめと今後の課題

本稿ではレジスタファイルサイズ削減を目的とした，ビット分割レジスタファイル (*Bit-Partitioned Register file: BPRF*) を提案した．BPRF ではレジスタファイルをビット方向に分割し，有効ビット幅の小さいオペランドに対しては必要な分だけのレジスタバンクを割り当てることで，レジスタの記憶領域の有効利用を狙うものである．

BPRF を評価した結果，従来のレジスタファイルに比べて，およそ半分のサイズで同程度の IPC を達成できることがわかった．また，その場合レジスタアクセス時間やレジスタアクセスの消費エネルギーを大幅に削減できることもわかった．しかし，本稿で述べた実装方式では，レジスタリネーミング用の *map table* や命令キューなどに追加の物理レジスタ ID 情報などを保持しなければならず，プロセッサ全体としては，かえって消費エネルギーが増えてしまうことも予想される．今後，*Virtual-Physical Registers*[1),2)] と組み合わせるなど，マイクロアーキテクチャ方式を工夫することで，ハードウェアコストを抑えつつ BPRF を実装できる方法を検討していく予定である．

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 (B) No. 14380136) によるものである．

参 考 文 献

1) A. Gonzalez, et al., “Virtual-Physical Registers”, *In Proc. the 4th HPCA*, pp.175–184, Feb. 1998.
2) T. Monreal, et al., “Delaying Physical Register Allocation Through virtual-Physical Registers”, *In Proc. the 32nd MICRO*, pp.186–198, Nov. 1999.
3) S. Jourdan, et al.,“A Novel Renaming Scheme to Exploit Value Temporal Locality through Physical Register Reuse and Unification”, *In Proc. the 31st MICRO*, pp.216–225, Nov. 1998.
4) S. Balakrishnan and G.S. Sohi, “Exploiting Value Locality in Physical Register Files”, *In Proc. 36th MICRO*, pp.265-276, Dec. 2003.
5) J.-L. Cruz, et al., “Multiple-Banked Register File Architectures”, *In Proc. 27th ISCA*, pp.316–325, June 2000.
6) R. Balasubramonian, et al., “Reducing the Complexity of the Register File in Dynamic Superscalar Processors”, *In Proc. 34th MICRO*, pp.237-248, Dec. 2001.
7) J.H. Tseng and K. Asanovic, “Banked Multiported Register files for High-Frequency Superscalar Microprocessors”, *In Proc. the 30th ISCA*, pp.62–71, June 2003.
8) G.S. Sohi, et al.,“Multi-scalar processors”, *In Proc. the 22th ISCA*, 1995.
9) R.E. Kessler, “The Alpha 21264 Microprocessor”, *IEEE Micro*, Vol.19, No.2, pp.24–36, Apr. 1999.
10) D. Brooks and M. Martonosi, “Value-Based Clock Gating and Operation Packing: Dynamic Strategies for Improving Processor Power and Performance”, *ACM Tr. on Computer Systems*, Vol.18, No.2, pp.89–126, May 2000.
11) K.C. Yeager, “The MIPS R10000 Superscalar Microprocessor”, *IEEE Micro*, Vol. 16, No. 2, pp.28-40, Aug. 1996.
12) D. Sima, “The Design Space of Register Renaming Techniques”, *IEEE MICRO*, Vol.20, No. 5, pp.70-83, Sep. 2000.
13) M.M. Martin, et al., “Exploiting Dead Value Information”, *In Proc the 30th MICRO*, pp.125-135, Dec 1997.
14) M.H. Lipasti, et al., “Physical Register Inlining”, *In Proc the 31st ISCA*, pp.325-335, June 2004.
15) T. Austin, et al., “SimpleScalar: An Infrastructure for Computer System Modeling”, *IEEE Computer*, Vol. 35, No. 2, pp.59–67, Feb. 2002.
16) C. Lee, M. Potkonjak, and W.H. Mangione-Smith, “MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems”, *In Proc. the 30th MICRO*, pp.330–335, Dec. 1997.
17) P.Shivakumar and N.P.Jouppi, “CACTI 3.0: An Integrated Cache Timing, Power, and Area Model”, *WRL Research Report 2001/2*, Compaq Computer Corporation, Western Research Laboratory, Aug. 2001.
18) D. Brooks, et al., “Wattch: A Framework for Architectural-Level Power Analysis and Optimizations”, *In Proc. 27th ISCA*, pp.83–94, June 2000.
19) R. Gonzalez, et al., “A Content Aware Integer Register File Organization” *In Proc the 31st ISCA*, pp.314-324, June 2004.
20) A. Aggarwal and M. Franklin, “Energy Efficient Asymmetrically Ported Register Files”, *In Proc. 21st ICCD*, pp.2–7, Oct. 2003.
21) B. Shackleford, et al., “Embedded System Cost Optimization via Data Path Width Adjustment”, *IEICE Trans. on Inf. & Syst*, Vol.E80-D, No.10, pp.974–981, Oct. 1997.