社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－MPS－50　(10)
2004／6／22

SMT プロセッサにおける物理レジスタバンクの動的割当て

加藤義人[†]　大和仁典[†]　辻元治[†]
佐藤未来子[†]　笹田耕一[†]　内倉要[†]
並木美太郎[†]　中條拓伯[†]

SMT プロセッサでは，同時に複数のスレッドのレジスタ・コンテキストを格納する必要があるため，より多くの物理レジスタが必要となる．また，実行ユニットの増強に際し，レジスタファイルに読み出しポート，書き込みポートを追加する必要がある．レジスタファイルのポート数の増加はレジスタファイルの面積の増大，レジスタアクセスの遅延増加などを引き起こす．これらの影響は，多くの実行ユニットを持ち，多数のスレッドを実行可能な SMT プロセッサの実装の障害になる．特にアクセス時間の増加は性能に与える影響が大きい．本研究では，レジスタファイルのアクセス時間を削減する手法として，物理レジスタファイルのバンク化と，その動的な割当てを提案する．

本方式により，レジスタファイルの面積の増加を起こさずにアクセス時間を最大で約 60% 削減させることができた．また，提案方式の導入による IPC の低下は最大でも 6%程度に抑えることができた．

Dynamic Allocation of Physical Register Banks for an SMT Processor

Norito Kato,[†] Masanori Yamato,[†] Osamu Tujimoto,[†]
Mikiko Sato,[†] Koichi Sasada,[†] Kaname Uchikura,[†]
Mitaro Namiki[†] and Hironori Nakajo[†]

In an SMT processor, an increasing number of register contexts of a thread requires a large number of physical registers. Moreover, a physical register file in an SMT processor requires more ports for the increasing number of execution units, which causes significant growth of area, access time and power consumption of a register file. These problems are significant hurdles to implement an SMT processor which executes the more number of threads with the more execution units. Especially, growth of access time of a register file has a large impact on performance. In this paper, we propose a strategy of dividing a physical register file into some banks and dynamic allocation of them to threads in order to reduce the access time of a register file. We have accomplished reduction in access time of a register file up to 60% without growth of area by using the proposed strategy. On the contrary, IPC degradation can be limited up to 6% by this strategy.

1. はじめに

近年，プログラム中の ILP(Instruction Level Parallelism) のみによる並列度の向上の限界が指摘され，TLP(Thread Level Parallelism) への注目が増してきている．TLP を利用するアーキテクチャの一つに SMT[1](Simultaneous Multithreading) がある．SMT とは，複数のスレッドが実行ユニットなどの命令実行に必要なリソースを共有し，効率よく並列度を上げることのできるアーキテクチャであり，いくつかの商用プロセッサ[2][3] にもこの技術は取り入れられている．SMT プロセッサでは，TLP の増加により処理性能の向上が期待されるが，複数のスレッドのレジスタ・コンテキストを同時に保持する必要があるため，必要な物理レジスタ数は増加する[4] また，高い並列効果を上げるためには多くの命令を同時にフェッチし，かつ多数の実行ユニットを同時に稼動状態にする必要があるため，レジスタファイルのポート数はシングルスレッドプロセッサよりも多くなる．このようなレジスタ数，ポート数の増加はレジスタファイルの面積，アクセス時間，消費電力を増大させる．これらの影響は，大規模な SMT プロセッサを実装する際の障害となる．本稿ではレジスタファイルのアクセス時間を削減するため，SMT プロセッサにおける物理レジスタファイルのバンク化と，スレッドへのバンクの動的割当てを提案する．

第 2 章では本稿で提案する物理レジスタバンクの動的割当てについて説明する．第 3 章で本方式の評価を行い，第 4 章で関連研究を挙げ，第 5 章でまとめる．

† 東京農工大学
Tokyo University of Agriculture and Technology

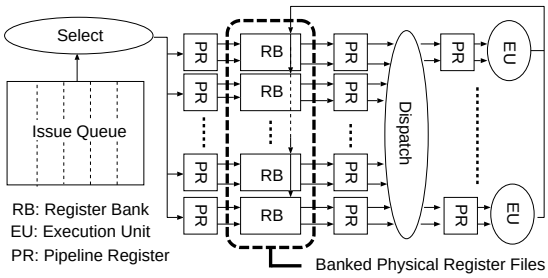

図 1 物理レジスタバンク動的割当て方式
Fig. 1 Banked physical register file scheme

2. 物理レジスタバンクの動的割当て方式

まず，用語の定義を行う．本稿では，プロセッサがスレッド実行の際に静的に割り当てる PC などの CPU のリソースのセットを，ソフトウェアのスレッドと区別するため，**AT(Architecture Thread)** と呼ぶことにする．

本稿ではレジスタファイルの巨大化を回避しつつレジスタファイルへのアクセス速度を改善するため，SMT プロセッサにおける物理レジスタバンクの，AT への動的な割当てを提案する．本稿ではこの方式を**物理レジスタバンクの動的割当て方式**と呼ぶ．この方式の概略図を**図 1**，パイプラインの概略図を**図 2(b)** に示す．本方式では物理レジスタファイルをリードポートの少ない複数個のバンクに分割し，それらを動作中の AT に割り当てる．

2.1 物理レジスタのバンク化

物理レジスタのバンク化はレジスタファイルのアクセス遅延を改善するため，多く提案されてきたが，本方式が採用するバンク化は Alpha21264 など[5)6)] で採用されているバンク化手法に基づくものである．物理レジスタファイルはリードポートの少ないバンクへ分割され，それらは物理レジスタファイルのコピーとして使用される．本方式では，一つの物理レジスタバンクの使用権を持つ AT(**オーナ AT**) は 1AT に限るとする．そのため，バンクは最低でも AT 数と同数，また，1 バンクあたりの物理レジスタ数は，シングル AT の物理レジスタファイルとして使用可能な数必要である．

2.2 バンクの動的割当て

本方式では，リードポート数の削減による発行幅の低下を防ぐため，各物理レジスタバンクは，プロセッサにより，AT に対し動的に割り当てられるとする．割り当てられた物理レジスタバンクが多いほど，1 サイクルで多くのオペランドをフェッチ可能なため，多くの命令を同時に発行できる．

物理レジスタバンクが動的に AT に割り当てられている様子を**図 3** に示す．この図では物理レジスタのバンク数は 4 で，バンク一つあたりに 2 のリードポートを持っている．図 3(a) では AT#1 のみが動作中で，AT#1 が全ての物理レジスタバンクを使用可能であり，1 サイクルで最大 4 命令発行可能である様子を示している．図 3(b) では AT#1 と AT#2 が動作中で，AT#1，#2 がそれぞれレジスタバンクを二つずつ使用可能であり，1 サイクル中に最大で 2 命令ずつ発行可能である様子を示している．

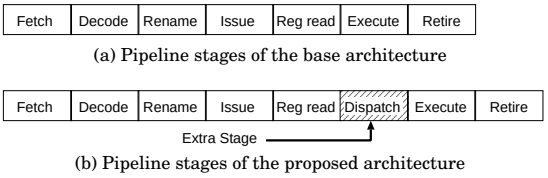

図 2 物理レジスタバンクの動的割当て方式におけるパイプラインステージ
Fig. 2 Pipeline stages of dynamic allocation of physical register banks scheme

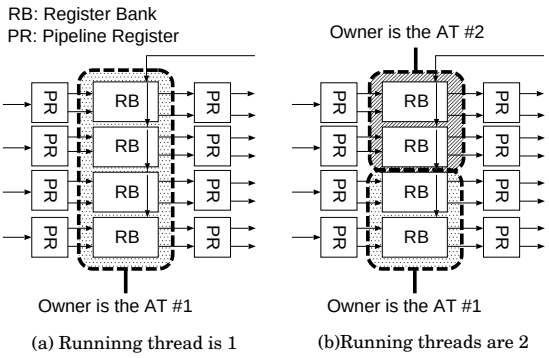

図 3 物理レジスタバンクの動的な割当て
Fig. 3 Dynamic allocation of physical register banks

実行ユニットからの結果は全物理レジスタバンクへブロードキャストされ，レジスタファイルのオーナの AT 番号と演算結果を生成した命令の AT 番号が等しい場合書き込まれる．本方式では，オペランドのフェッチ後に命令を演算器へ振り分ける．そのため，パイプラインステージを一段追加する必要がある（図 2(b)）．

2.3 バンク割当ての決定

物理レジスタバンクの割当ての変更はスレッドの生成，消去時に行われる．割当ての決定は，複雑なアルゴリズムを採用したとしても高速に行うことができるよう，スレッドの生成，消去後の各 AT の状態をインデックスとし，次の物理レジスタバンクの割当て状態が予め保存されているテーブルへアクセスすることで決定するとした．

割当て決定アルゴリズムには，様々なものが考えられるが，以下の条件を満たす必要がある．

条件 1: スレッドを割り当てられている AT は一つ以上の物理レジスタバンクを割り当てられる必要がある．

条件 2: スレッドが生成されてから消去されるまでの間，一貫して所有し続けるレジスタファイルを一つ以上もつ．

条件 1 は AT が物理レジスタバンクを所有しなければレジスタの値を保持することができないためである．条件 2 については，本方式では各物理レジスタバンクはオーナが変更された後の演算結果のみを持つためである．そのため，条件 2 を満たさない割当てを行った場合，どの物理レジスタファイルバンクも持たないレジスタの値が存在することになる．

表 1 レジスタファイルのハードウェア見積もりにおける基本パラメータ
Table 1 Base parameters in the evaluations of register files

Parameter	Value
命令発行幅	8
論理レジスタ数	32
データ幅	64bit
テクノロジ	$0.13\mu m$

表 2 提案方式でのレジスタファイルのパラメータ
Table 2 Parameters of the register files of the proposed architecture

Parameter	Value
Issue_width	8
Number_of_banks	Number_of_ATs
Number_of_read_ports_per_banks	Issue_width * 2 / Number_of_Banks
Number_of_write_ports_per_banks	Issue_width

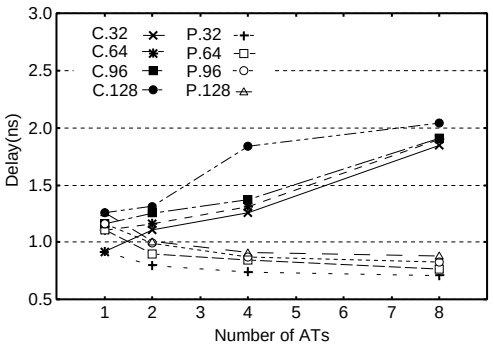

図 4 各構成でのレジスタファイルのアクセス時間
Fig. 4 Access time of a register file of each configuration

3. 評　　価

提案方式と従来型の SMT プロセッサのレジスタファイル，IPC の比較評価を行う．今回は簡単化のため，プロセッサは整数演算器のみを持つと仮定した．なお，浮動小数点演算用のレジスタファイルも同様に本方式を適用することが可能である．

3.1 レジスタファイルの評価

ここでは本方式を適用した場合の物理レジスタファイルの総面積，遅延の評価を行う．評価には CACTI を使用した．評価中で固定なパラメータは**表 1** に示したものとする．また，提案方式のいくつかのパラメータは**表 2** のように決定される．

図 4 は各構成でのアクセス時間である．レジスタファイルが同時にリネーム可能な命令の数を変化させて測定した．図中では各構成を **“アーキテクチャ. リネーム可能命令数”** と表記している．アーキテクチャは従来型の SMT プロセッサを表す $C(Conventional)$，提案方式を表す $P(Proposed)$ のいずれかである．

図 4 より，従来型の SMT プロセッサと提案方式では全く傾向が異なることが分かる．従来型の SMT プロセッサでは，AT 数が増えるに従い，遅延も増加している．これは AT 数が増加することにより，物理レジスタファイルのレジスタ数が増加するためである．提案方式では逆に AT を増やすにしたがってアクセス時間が短くなっている．これは，AT 数が増えるに従い，物理レジスタバンク一つあたりのリードポート数が減るためである．提案方式では従来型の SMT プロセッサに比べ，最大で約 60%レジスタファイルのアクセス時間が削減されている．

表 3 IPC 評価でのプロセッサのパラメータ
Table 3 Parameters of a processor in the IPC evaluation

AT 数	8
実行ユニット	ALU 6, Load/Store Unit 2, Complex ALU 3,
発行幅	8
分岐予測器	gshare, PHT 4K entries, GHR 12bit, BTB 512 entries 4 set associative
リネーム可能レジスタ数	96
Issue Queue	96 entries
Reorder Buffer	128 entries
Retire	8
フェッチポリシー	ICOUNT.2.8[4]

3.2 IPC の評価

命令レベルシミュレータにより IPC の測定を行った．プロセッサの各パラメータを**表 3** に示す．命令セットは MIPS-II をベースとした OChiMuS/PE[7] ISA を使用し，スレッドライブラリ MULiTh[8] を利用して評価した．今回の実験で使用したプログラムは以下の四つである．

- Radix ソート（SPLASH-2[9]）
- LU 分解（SPLASH-2）
- 行列乗算
- Dhrystone

浮動小数点演算は gcc の soft-float を使用しエミュレートした．また，プロセッサコアの性能の変化を調査するため，キャッシュはデータキャッシュ，命令キャッシュともに 100%ヒットすると仮定した．gcc のバージョンは 3.2，最適化オプションには “-O2 -mips2” を使用した．なお，今回使用したベンチマークのうち，Dhrystone はシングルスレッドプログラムであるため，同じプログラムを複数のスレッドで同時に実行した．

今回の実験ではできるだけ，各 AT の所有物理レジスタバンク数が平等になるようなバンク割当てアルゴリズムを用いた．このアルゴリズムの概略を以下に示す．

(1) ある AT がスレッドに割り当てられている場合，その AT と同じ番号の物理レジスタバンクはその AT に割り当てられる．
(2) その後あまったレジスタファイルは順に一つずつ，所有している物理レジスタバンクが最も少ない動作中の AT に割り当てられる．

このアルゴリズムでは AT 番号と番号が等しい物理レジスタバンクが，AT がスレッドに割り当てられてから実行を終了するまで所有し続ける物理レジスタバンクになる．

測定結果を**図 5** に示す．この図は提案方式の IPC の，従来型の SMT プロセッサの IPC との比を示している．提案方式の IPC の低下は最大でも 6%程度で

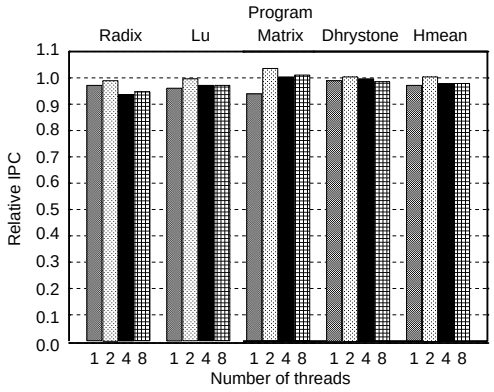

図 5 従来型の SMT プロセッサと提案方式の IPC 比
Fig. 5 Relative IPC of proposed architecture to conventional one

あることが分かる．

4. 関 連 研 究

文献 10) では，アクセスに数サイクルかかるレジスタファイルの遅延を隠蔽するため，レジスタファイルに規模の小さなキャッシュを持たせるという方式が提案されている．この方式は Alpha21464[11] にも採用されている．この方式ではさらにレジスタファイルのアクセス遅延が増加した場合，性能の低下もより大きくなる．そのため，この方式を SMT プロセッサに適用した場合，プロセッサの AT 数の増加に対する耐性が低い．本稿の提案方式では AT 数が増えてもレジスタファイルのアクセス時間は変わらないため，本方式の方がスケーラブルであるといえる．

文献 5)11)6) などでは，物理レジスタファイルを 2 バンクに複製することにより，レジスタバンク一つあたりのリードポートの削減を図っている．これらの文献で説明されているアーキテクチャでは，物理レジスタバンク同様，命令 Window，Bypass Logic，演算器なども分割され，クラスタを構成している．本方式はこれらのバンク化手法を SMT 向けに拡張したものである．ただし，本稿の実験ではクラスタ化は行っていない．文献 12) などで提案されている方式では，読み出しポートに加え，書き込みポートの削減も行う．レジスタの値が格納されるべきバンクは，主にレジスタ番号により決定される．一般的にこのバンク化方式はライトポートも削減するため，前述の，本方式が採用したレジスタファイルのコピーを作成する方式に比べ，レジスタファイルの面積，遅延，消費電力の削減効果が大きいが，競合による IPC の低下も大きい．

5. お わ り に

5.1 提案方式の効果

本稿では，SMT プロセッサにおけるレジスタファイルのアクセス時間削減手法として，物理レジスタバンクの動的な割当てを提案した．この方式により，レジスタファイルのアクセス時間を大幅（本稿の実験では最大 60%程度）に改善できることが分かった．また，本方式の適用による IPC の低下は最大でも 6%程度と小さく，レジスタファイルのアクセスがクリティカルパスになると仮定した場合，動作周波数の向上により，大幅に性能を向上することができる．

5.2 今後の課題

本稿では，レジスタファイルの低速化についての対策を提案したが，発行幅の大きなプロセッサでは Wakeup，Selection，Bypass などの論理もクリティカルパスになることが予想される．本稿ではレジスタファイルの影響のみを調べるため，それらは理想化されているが，今後は，演算器などのクラスタ化[5] や FIFO ベースの命令 Window[6] などの手法と組み合わせた場合の影響を調べていく．

参 考 文 献

1) Tullsen, D. et al.: Simultaneous Multithreading: Maximizing On-Chip Parallelism, *ISCA-22*, pp. 392–403 (1995).
2) Marr, D. et al.: Hyper-Threading Technology Architecture and Microarchitecture: A Hypertext History, *Intel Technology Journal*, Vol. 6, No. 1 (2002).
3) Kalla, R. et al.: POWER5: IBM's Next Generation POWER Microprocessor.
4) Tullsen, D. et al.: Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor, *ISCA-23*, pp. 191–202 (1996).
5) Kessler, R. E.: The Alpha 21264 Microprocessor, *IEEE Micro*, Vol. 19, No. 2, pp. 24–36 (1999).
6) Palacharla, S. et al.: Complexity-Effective Superscalar Processors, *ISCA-24*, pp. 206–218 (1997).
7) 河原ほか: システムソフトウェアとの協調を目指すシングルチップマルチスレッドアーキテクチャの構想, *CSS2002*, Vol. 2002, No. 18, pp. 1–8 (2002).
8) 笹田ほか: マルチスレッドアーキテクチャにおけるスレッドライブラリ の実装と評価, *SACSIS2003*, pp. 13–20 (2003).
9) Woo, S. C. et al.: The SPLASH-2 Programs: Characterization and Methodological Considerations, *ISCA-22*, pp. 24–36 (1995).
10) Cruz, J.-L. et al.: Multiple-Banked Register File Architectures, *ISCA-27*, pp. 316–325 (2000).
11) Preston, R. P. et al.: Design of an 8-wide Superscalar RISC Microprocessor with Simultaneous Multithreading, *ISSCC Digest of Technical Papers* (2002).
12) Tseng, J. H. et al.: Banked Multiported Register Files for High-Frequency Superscalar Microprocessors, *ISCA-30*, pp. 62–71 (2003).