Vol.2014-ARC-208 No.11
2014/1/23

レジスタ・キャッシュ・システムにおけるレジスタ・ファイルへの書き込みの削減手法

山田 淳二[1,a)]　倉田 成己[1]　塩谷 亮太[2]　五島 正裕[1]　坂井 修一[1]

概要：巨大なレジスタ・ファイルの消費電力と熱の問題を解決するため，レジスタ・キャッシュが提案されている．レジスタ・キャッシュは，メイン・レジスタ・ファイル (MRF) からのリードを削減し，面積と消費電力を数分の 1 程度までに削減することができる．
これに加えて，筆者らが提案するライト・スカッシュ・バッファを用いると，上書きされたレジスタのメイン・レジスタ・ファイルへのライトも削減が可能である．しかし，従来のライト・スカッシュ・バッファの実装では，上書きされたレジスタの無効化及びライト・スカッシュ・バッファにヒットしたリードのフォワーディングのために CAM などを用いた連想検索が必要であった．
本稿では，(1) ライト・スカッシュ・バッファからデキューされるまでフリー・リストに上書きされたレジスタを返却しない (2) ライト・スカッシュ・バッファへのリードヒットをストールで処理するの 2 点によって，連想検索を不要とするライト・スカッシュ・バッファを提案する．
現在までの評価では，2-read 2-write の MRF と長さ 8 のライト・バッファ ×2 本を用いるレジスタ・キャッシュ・システムをベースとして，長さ 16×6 本のライト・スカッシュ・バッファを用いて MRF を 2-read 1-write とした場合に，IPC の低下は 94.5%，面積は 79.8%, 消費電力は 72.6%であった．

1. はじめに

物理レジスタ・ファイルは，最近のスーパスカラ・プロセッサの構成要素の中でも最も高コストなものの 1 つとなっている．

巨大なレジスタ・ファイル

この要因として，まず，レジスタ・ファイルの容量の増加がある．通常，物理レジスタ・ファイルには in-flight な命令数に応じた容量が必要となる．より多くの命令レベル並列性を抽出するため，近年ではスーパスカラ・プロセッサの in-flight 命令数を増やす傾向にあり，その一環として物理レジスタ・ファイルの容量も増大している．また，マルチスレッドをサポートするコアでは，スレッド数に比例した容量が必要となるため，更に数倍という規模での容量増を引き起こすことになる．

次に物理レジスタ・ファイルのポート数の増加がある．4 命令同時実行可能なスーパスカラ・プロセッサでは，レジスタ・ファイルに 8 つのリード・ポートと 4 つのライト・ポートが必要となる．レジスタ・ファイルは通常，多ポートの RAM によって構成されるが，RAM の回路面積はポート数の 2 乗に比例するため，その回路面積は容量の割に非常に大きなものとなる．

これら 2 つの理由により，レジスタ・ファイルは近年では L1 データ・キャッシュに匹敵するほど巨大な回路となっている．巨大なレジスタ・ファイルは，IPC の低下や回路の複雑さの増大に加えて，消費電力，熱の増大などの様々な問題を引きおこす．

消費電力と熱の増大

RAM の消費電力は，その回路面積に加えて，アクセス頻度にも比例する．ロード/ストア命令が L1 データ・キャッシュに対してそれぞれ 1 回しかアクセスを行わないのに対し，ほぼ全ての命令は通常，レジスタ・ファイルに対して 2〜3 回のアクセスを行う．このため，面積が同程度の L1 データ・キャッシュと比較して，レジスタ・ファイルはより大きな電力を消費する．

消費電力とそれによって発生する熱は，最近のプロセッサ・コアにおける問題の中でも，もっとも深刻なものの一つである．レジスタ・ファイルを含む領域は，プロセッサ・コア内のホット・スポットであり，その動作周波数を制限する主な要因の一つになっている．

[1] 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
[2] 名古屋大学大学院工学系研究科
Graduate School of Engineering, Nagoya University
[a)] yamadaju@mtl.t.u-tokyo.ac.jp

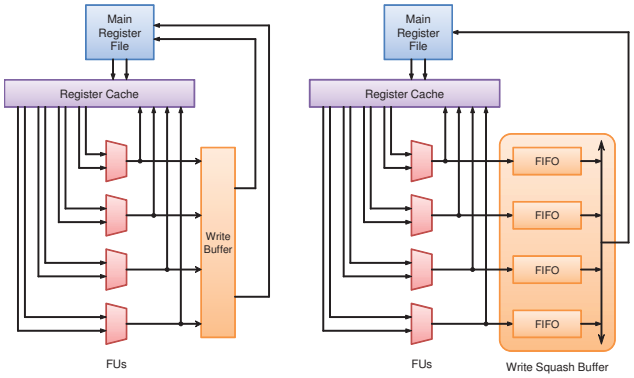

図 1 レジスタ・キャッシュ・システムのブロック図

レジスタ・キャッシュ

レジスタ・キャッシュを導入すれば，巨大なレジスタ・ファイルに起因するこれらの様々な問題を解決することが可能である．

図 1 に，レジスタ・キャッシュのシステムのブロック図を示す．レジスタ・キャッシュのシステムは，主に，多ポート少エントリの**レジスタ・キャッシュ**（Register Cache，**RC**）と少ポート多エントリの**メイン・レジスタ・ファイル**（Main Register File，**MRF**）の組合せによって構成される．RC は少エントリゆえ，MRF は少ポートゆえに，元のレジスタ・ファイルより格段に小面積・低電力となる．

特に，塩谷らが提案した **NORCS** (Non-latency-Oriented Register Cache System) では，IPC の低下を 2%程度に抑えながら，面積を 1/4 程度に，消費電力を 1/3 程度に削減することに成功している [1].

レジスタ・キャッシュ・システムへのライト

2 章で詳しく述べるが，レジスタ・キャッシュ・システムは通常，ライト・スルー方式である．そのため，MRF へのライトが消費電力の少なくない部分を占めることになる．

レジスタ・キャッシュ・システムでは，リードの大部分は RC でカバーされ，MRF に対してはほとんど行われない．その一方ライトは，ライト・スルーであるため，すべて RC と MRF の両方に対して行われることになる．そのため，MRF の電力の大部分はライトによるものとなる．NORCS の評価では，レジスタ・キャッシュ・システムの全電力のうち半分が RC，残りの半分が MRF によって消費されており，MRF の消費電力のほとんどがライトによって生じている．すなわち，レジスタ・キャッシュ・システムの全消費電力の半分程度を MRF へのライトが占めることとなっている [1].

このことは特に，マルチスレッド・プロセッサで重要な問題となる．マルチスレッド・プロセッサでは，スレッド数の増加に対して，RC の容量はほとんど増やす必要がないことが分かっている [2], [3]．その一方で，MRF はスレッド数分だけ増やす必要がある．MRF へのライトの占める割合は，2-way のマルチスレッド・プロセッサでは 2/3, 3-way では 3/4 程度にまで増加することになる．

更に近年では，プロセスの微細化に伴い，ライトが困難になりつつあり [4], [5]，ライト時の電圧制御 [6], [7] などの技術で，リードに比べてライトの電力が増加する傾向にある．

ライト・スカッシュ・バッファ

筆者らは，この対策として，この MRF へのライトを大幅に削減する**ライト・スカッシュ・バッファ**（Write Squash Buffer，**WSB**）を提案している [8].

WSB は，既存のレジスタ・キャッシュ・システムのライト・バッファと同様に，演算器と MRF の間に置かれる．図 1 においては，ライト・バッファを WSB に置き換えることになる．実行結果は，RC に書き込まれると同時に WSB にバッファリングされ，その後 MRF へと送られる．

既存のライト・バッファは，MRF へのライトの時間的なばらつきを平滑化するために置かれる．ライト・バッファにバッファリングされたライトは，MRF のライト・ポートが空いている限り速やかに MRF に送られる．

ライト・スカッシュ・バッファは，出来るだけスカッシュ率を高くするために，ライトを長く滞在させてスカッシュを出来るだけ待つ設計も可能であるが，本稿のライト・スカッシュ・バッファは，連想検索を必要とするフォワーディングを排除することを優先し，既存のライト・バッファと同様に，バッファリングされたライトを MRF のライト・ポートが空いている限り速やかに MRF に送る設計とした．

WSB は，既存のライト・バッファとは異なり，MRF へのライトだけでなくライトが「潰される (squash)」ことによってもライトが除去される．ライトは，物理レジスタの解放によって「潰される (squash)」ことから，WSB への滞在時間が長いライトほど潰される確率は高くなる．従って，MRF のライト・ポートが不足し，WSB サイズ (使用されているエントリ数) が大きくなると，「潰される (squash)」確率が高くなり，WSB のサイズはある程度のサイズで飽和することになる．

後述の評価からは，このサイズは，最も使用率が高い WSB でも 16 エントリで十分であることが分かった．WSB は，演算器ごとに分散された 1-read 1-write の FIFO で構成することができ，MRF は，2-read 2-write，128 エントリ程度であるから，16 エントリの FIFO はそれに比べて十分に小さい．

NORCS の評価では MRF は 2-read 2-write としているが，これを 2-read 1-write で済ませることができれば，MRF の回路面積は半分程度にまで削減することができる．

後述の評価によれば，MRF 2-read 2-write の NORCS から，1-write にライト・ポートを削減した場合，NORCS では相対 IPC が 79.2%まで低下するところ，WSB を使用することによって性能低下は 94.5%にまで抑えることができた．

本稿の WSB では，ライトを速やかに MRF へ送る設計としていることから，主に，ライト電力の削減は，MRF のポート削減に依存している．このため，2-read 1-write の MRF を使用する場合，ライト数の削減率は 20.1%に留まるが，MRF の 1 アクセスあたりのエネルギーは 75%程度であることから，MRF の消費電力は 58.9%となり，ライト・バッファ又はライト・スカッシュ・バッファレジスタ・キャッシュを加えたレジスタ・キャッシュ・システム全体の消費電力は，72.6%となった．

以下，2 章でレジスタ・キャッシュ・システムについて簡単にまとた後，3 章で WSB について詳しく説明する．4 章では，WSB の面積及び消費電力の簡易的な見積もりについて述べ，5 章で，性能の評価結果についてまとめる．

2. レジスタ・キャッシュ・システム

本章では，提案のベースとなる**レジスタ・キャッシュ・システム**（Register Cache System，**RCS**）について説明する．本稿で提案するライト・スカッシュ・バッファは，既存の RCS のライト・バッファを置き換えて，メイン・レジスタ・ファイルへのライトの消費電力を削減するものである．そこで本章では，RCS へのライトを中心に解説し，特にメイン・レジスタ・ファイルへのライトの消費電力が問題となることを述べ，既存の RCS のライト・バッファの役割について説明する．

2.1 レジスタ・キャッシュ・システムの概要

RCS のブロック図は，図 1 に示した．前章で述べたように，RCS は，多ポート少エントリの**レジスタ・キャッシュ**（Register Cache，**RC**）と少ポート多エントリの**メイン・レジスタ・ファイル**（Main Register File，**MRF**）の組合せによって構成される．RC は少エントリゆえ，MRF は少ポートゆえに，元のレジスタ・ファイルより格段に小面積・低電力となる．文献 [1] の評価では，面積を 1/4 程度に，消費電力を 1/3 程度に削減することに成功している．

特に, 塩谷らが提案した **NORCS** (Non-latency-Oriented Register Cache System) では，RC ヒット時にもレイテンシを削減しない「RC ミスを仮定したパイプライン」の採用により，IPC の低下を 2%程度に抑えることに成功している．

以下では NORCS を念頭に説明するが，提案のライト・スカッシュ・バッファは，それ以外の一般の方式にも応用可能である．

本提案を理解するためにはまず，物理レジスタの割り当てと解放について正確に理解する必要がある．そのため，次節でまず物理レジスタの割り当てと解放についてまとめた後，2.3 以降で RCS へのライトについて説明する．

2.2 物理レジスタの割り当てと解放

物理レジスタへのリネーミングをベースにする方式では，空き物理レジスタはフリー・リストによって管理されている [9]．

リネーム時に，フリー・リストから空き物理レジスタが取り出され，各命令のデスティネーションに対して割り当てられる．

プログラム・オーダ上で，ある論理レジスタ L をデスティネーションとする命令を順に I_{p1}，I_{p2}，$\cdots$ とする．また，I_{p1}，I_{p2} のデスティネーションには物理レジスタ P_1，P_2，$\cdots$ がそれぞれ割り当てられたとする．**図 2** では，$I_{p1} : L(P_1) = \cdots$ などと表現している．

P_1 に格納される命令 I_{p1} の実行結果を参照する可能性があるのは，プログラム・オーダ上で命令 I_{p1}〜I_{p2} の間にある命令だけである（図 2 では I_{c1}）．I_{p2} 以降に L をソースとする命令（I_{c2}）があったとしても，それは P_2 に格納される I_2 の実行結果を参照することになる．したがって，命令 I_{p2} がコミットされた時点で，I_{p1}〜I_{p2} の間にある命令（I_{c1}）も完了しており，命令 I_{p1} の結果を参照する命令は最早システム内に存在しないことが保証される．

そのため，I_{p1} のデスティネーションに割り当てられた P_1 は，その値を参照する命令が存在しないので，この時点で解放してよい．解放された物理レジスタは，フリー・リストに返却される．

2.3 レジスタ・キャッシュ・システムへのライト

RC に対するライトは通常，ライト・スルー方式で処理される．すなわち，レジスタへの書き込みは，RC と同時に MRF に対しても行われる．これは以下の理由による．

通常のメモリ階層における，ライト・スルーに対するライト・バックのメリットは，2 回目以降のライトをもキャッシュ上のエントリに対して実行することによって，次の階層へのライト（バック）の回数を 1 回で済ますことにある．ライト・スルーでは，1 回のライトごとに，次の階層への 1 回のライト（スルー）が発生することになる．

しかしレジスタ・キャッシュにおいては，ライト・バックのこのようなメリットは生じない．これは，通常のメモリのロケーションとは異なり，レジスタに対してはリネーミングが施されるためである．前節で述べたように，各命令のデスティネーションには別の物理レジスタが割り当てられる．したがって，RC 上のあるエントリに対するライトは，割り当てられた命令が実行された時の 1 回のみである．「2 回目のライト」はそもそも存在しない．

RC をライト・バックとすると，ライト・バック時に RC を読み出すためのポートが余計に必要になり，かえって不利である．

以上の理由により，RC はライト・スルーとするのである．

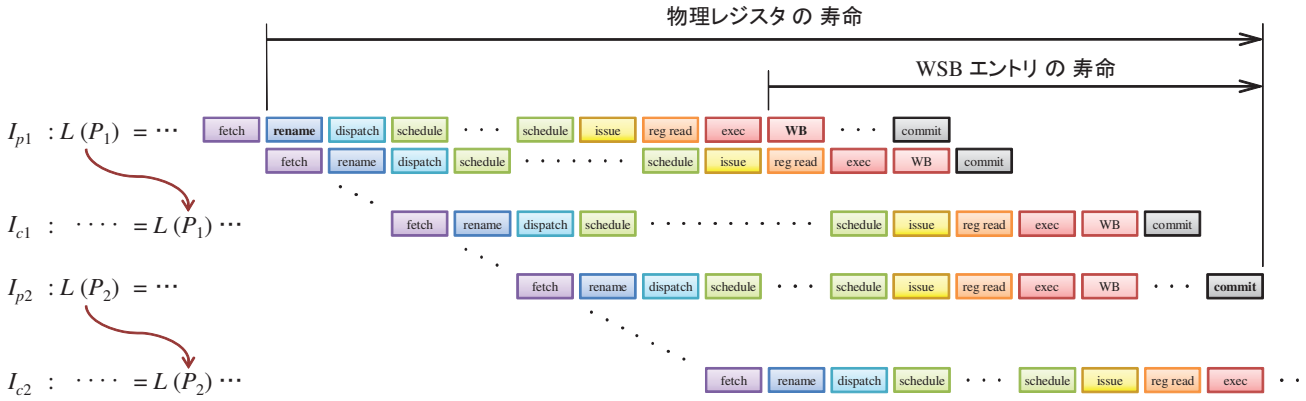

図 2 物理レジスタ・ファイルと WSB エントリの寿命

2.4 MRF へのライトの消費電力

RCS では，以下の理由により，MRF の電力の大部分はライトによるものとなる：

リード リードの大部分は RC でカバーされ，MRF に対してはほとんど行われない．RC ヒット率は，90% 程度以上になる．

ライト ライト・スルーのため，すべて MRF に対しても行われる．

NORCS の評価では，RCS の全電力の半分を RC，残りの半分を MRF が消費している．MRF のダイナミック電力のほとんどをライトが占めているため，結局 RCS の全電力の半分程度を MRF へのライトが占めることとなっている．

2.5 MRF へのライトの消費電力の増加

近年では，RCS の全消費電力における MRF へのライトの占める割合を更に増加させる要因がある．

2.5.1 マルチスレッド化

MRF へのライトの消費電力は，マルチスレッド・プロセッサでより重要な問題となる．

マルチスレッド・プロセッサでは，スレッド・スケジューリング・ポリシに強く依存するが，スレッド数の増加に対して，RC の容量はほとんど増やす必要がないことが分かっている [2], [3].

一方 MRF は，少なくとも各スレッドの論理レジスタを保持する必要があり，必要な容量はスレッド数に比例する．

したがってマルチスレッド・プロセッサでは，レジスタ・キャッシュ・システム全体の消費電力における MRF へのライトの占める割合は，2-way では 2/3 程度，3-way では 3/4 程度，と増加することになる．

2.5.2 プロセスの微細化

近年では，プロセスの微細化に伴い，低い動作電圧で RAM のセルを反転させるのが困難になりつつある [4], [5]. この問題に対処するためにライト時の電圧制御 [6], [7] などの技術を採用すると，リードに比べてライトの電力が増加することになる．

2.6 レジスタ・キャッシュ・システムのライト・バッファ

提案のライト・スカッシュ・バッファは，既存の RCS のライト・バッファを置き換えて MRF へのライトの消費電力を削減するものである．そこで本章の最後に，既存の RCS のライト・バッファについて説明する．

各サイクルに演算器から RCS へと送られて来るライトの最大数は，同時実行可能な命令数に等しい．しかし MRF のライト・ポートは，必ずしもこの最大数だけ用意する必要はなく，平均的には IPC 程度あればよい．既存の RCS のライト・バッファは，このライトの時間的なばらつきを平滑化するために設けられている．

そのため，このライト・バッファの容量はライトの「瞬間最大風速」を受け入れるのに十分であればよい．

文献 [1] の評価では，同時実行可能な命令数 4 に対して，MRF は 2 本の専用のライト・ポートを持てば十分であるとしている．この場合，バッファのエントリは毎サイクル 2 ずつ減らすことができ，その結果，バッファの容量は 4〜8 程度でよいとしている．

3. ライト・スカッシュ・バッファ

本章では，提案手法である**ライト・スカッシュ・バッファ**（Write Squash Buffer，**WSB**）について詳しく説明する．

3.1 ライト・スカッシュ・バッファの概要

WSB は，従来の RCS のライト・バッファと同様，演算器とメイン・レジスタ・ファイルの間に配置される．図 1 に示したブロック図においては，ライト・バッファを WSB に置き換えることになる．演算器から送られて来る実行結果は，RC に書き込まれると同時に，WSB にもバッファリングされる．WSB にバッファリングされた MRF へのライトは，その後 MRF へと送られ，MRF を更新すること

になる．

ライト・スカッシュ・バッファの役割

2.6 節で述べたように，既存の RCS のライト・バッファの役割は，MRF へのライトの時間的なばらつきを平滑化することにある．一方 WSB の役割は，WSB 上で MRF へのライトを「潰す (squash)」ことにある．WSB は，その容量の許す限り MRF へのライトを保持し続ける．そしてその間に，ライトが「潰される」のを待つ．

前述したように，ライトが「潰される」原理は，物理レジスタの解放を WSB 上でも行うことにある．2.2 節では，命令 I_{p2} がコミットされたら，I_{p1} のデスティネーションに割り当てられた P_1 を解放してよいと述べた．この時，解放される P_1 へのライトが WSB 上にあれば，それも削除してしまってよい．すなわち，論理レジスタへの上書きによって，WSB 上のライトも「潰される」のである．更に，I_{p3} がコミットされると，今度は P_2 に対するライトもまた「潰される」ことになる．このようにして，バッファに保持しきれなくなったライトだけが MRF に対して実行されることになる．

解放される物理レジスタは物理レジスタの管理機構に尋ねればよく，WSB の無効化のために別途管理する必要はない．

なお，このようなことが可能であるのは，WSB が RCS のライト・バッファであるからである．RCS へのリードのほとんどは RC によってカバーされ，MRF がリードされることは少ない．すなわち，「潰されて」MRF に書かれなかった実行結果を必要とする命令は，（ヒットすれば）RC からそれを得たのである．したがって，RCS の一部ではない，単独の WSB は，あまりうまく働かないことに注意されたい．

分散構成

WSB の構成自体は，既存の RCS のライト・バッファと大きな違いはない．ただし，既存の RCS のライト・バッファが 4〜8 エントリ程度であるのに比べると格段に大容量であるので，小面積化の工夫が必要となる．そこで WSB は，1 つの多ポートの RAM によって集中的に実装するのではなく，図 1 右側に示したように，各演算器ごとに分散配置する．演算器ごとの個々の RAM は 1-read/1-write でよく，フラグメンテーションを考慮しても，合計の回路面積を大幅に削減することができる．

3.2 WSB の動作

次に，WSB の動作について詳細に説明する．

エンキュー

エンキュー側の動作は単純である．対応するバッファに空きがあれば，演算器は実行結果のライト・リクエストをエンキューする．いずれかのバッファがフルであれば，バックエンドはストールする．

ライト・バッファ・ヒットとフォワーディング

RCS では一般に，リードの大半は RC にヒットする．しかし稀に，**ライト・バッファ・ヒット**，すなわち，RC にミスした実行結果がライト・バッファ（もしくは WSB）に存在することがある．その場合，その実行結果をライト・バッファ（WSB）から演算器へと提供する必要がある．

選択肢としては，以下の 2 つが考えられる：

フォワーディング バッファから読み出す．

ストール バックエンド・パイプラインをストールし，当該ライトが MRF へと送られるのを待ち，バイパスと同様の処理で演算器に送る．

本稿では，連想検索のための CAM を必要としない，後者の方法を検討する．

ライト・ヒットに対して対応する必要があるのは，レジスタ・リードに対して，(1) レジスタ・キャッシュミスしかつ (2) ライト・バッファにヒットする場合である．

ライト直後のデータはレジスタ・キャッシュにある確率が高いことから，レジスタ・キャッシュミスをしたレジスタの大半は MRF に存在すると考えられ， STALL が必要となるケースは少ないと考えられるが，ライト・バッファにより新しい値が存在すれば，必ずライト・バッファの値を使用する必要がある．

NORCS ではこれをストールで実現していたが，ライト・バッファに値が存在する場合のみストール処理とするためには，ライト・バッファ・ヒット判定が必要である．さもなければ，レジスタ・キャッシュミスが全てストールを招くことになり，現実的な性能は得られない．

しかし，ライト・バッファ・ヒット判定さえ行わえば，連想検索によるフォワーディングまでは行わなくても，ライト・バッファに滞在するライトの数を制御することで，ライト・バッファ・ヒットによるストールの回数は限定することができる．

このライト・バッファ・ヒット判定を連想検索を行わずに実現するには，例えば，各物理レジスタに対してライト・スカッシュ・バッファに滞在してるか否かを管理する 1bit のフラグがあればよい．このフラグは，WSB へのエンキュー及びデキューで更新する必要がありので，WSB 1 本あたり 2W のライト・ポートが必要であり，例えば，INT 系 WSB が 4 本であれば，8 ライト・ポートとなる．

しかし，エンキューで 1 書込み，デキューで 0 書込みに限定されていることを考慮すると，**図 3** に示すように，通常の SRAM セルよりも効率的な構造で，同じ機能を実装することが可能である．本稿では，この構造の”Set/Clear 型メモリ”を前提として，面積及び消費電力を見積もる．

無効化

対応する物理レジスタが解放された時には，以下の 2 通りの方法が考えられる．

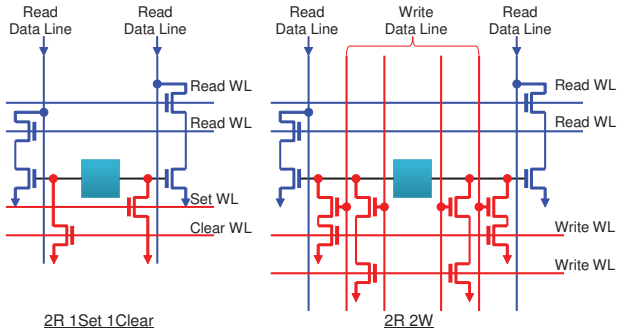

図 3 状態フラグ用 Set/Clear 型メモリ (左) と通常の SRAM(右)

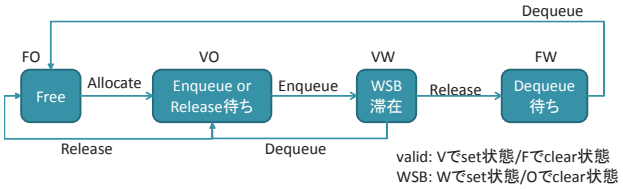

図 4 物理レジスタの状態遷移図

(1) 連想検索により，WSB の対応エントリを無効化する

(2) 解放された物理レジスタを記録し，デキュー時に照合する

本稿では，連想検索のための CAM を必要としない，後者の方法を検討した．

後者の実現には，解放された物理レジスタの管理が必要であるが，フリー・リスト内をサーチすることは現実的ではない．効率的な方法として，図 3 に示した，Set/Clear 型の効率的なメモリを使用することが考えられる．

後者の方法では，ライト・スカッシュ・バッファ滞在中の物理レジスタが無効化されても，デキューされるまでは，ライト・スカッシュ・バッファから取り除かれることはない．この場合，ライト・スカッシュ・バッファ滞在中の物理レジスタが再使用されると，ライト・スカッシュ・バッファからのデキュー時に不必要な MRF へのライトが発生することとなる．

フリー・リストへの物理レジスタ返却を，ライト・スカッシュ・バッファからデキューされるまで待つようにすれば，この問題は解決できる．この様子を状態遷移図にしたものを，**図 4** に示す．

ライト・バッファ・ヒットの判定にも使われた WSB 滞在フラグ 1bit と，物理レジスタが有効か否かの有効・フラグの物理レジスタ 1 つあたり合計 2bit のフラグで，全ての状態を管理することが可能である．

デキュー

デキューされたライトは，解放済みの物理レジスタに対応するライトは読み飛ばし，その他のライトはメイン・レジスタ・ファイルにライトする必要がある．ここで，ライト・スカッシュ・バッファを構成する FIFO を物理レジスタ番号用 (7bit) と Data 用 (64bit) とに分割し，サイズの大きな Data 用 FIFO からのデキューは，メイン・レジスタ・ファイルへのライトが発生する場合にのみ行うようにすれば，効率が上がると考えられる．これを実現するデキュー回路を**図 5** に示す．

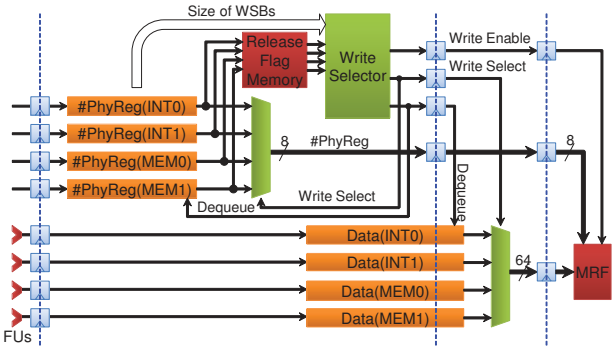

図 5 ライト・スカッシュ・バッファのデキュー回路

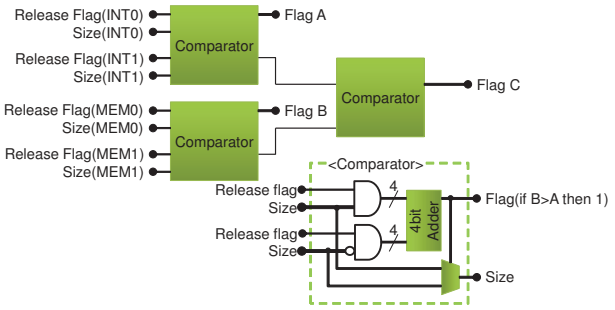

図 6 ライト・セレクタ

後述のように，ライト・スカッシュ・バッファの利用状態は演算器によってかなり異なるため，サイズが大きいライト・スカッシュ・バッファから優先的にデキューを行う．図 5 の内，ライト・セレクタは他にあまり例のない回路であるが，例えば，**図 6** に示すような回路が考えられる．図 6 では 4 つのライト・スカッシュ・バッファのサイズ (4 bit 幅) を勝ち残り方式で比較している．それぞれの比較結果を示す Flag A, B, C をデコードすれば，ライト・スカッシュ・バッファサイズの大きい順にライトを選ぶ ”Write Select” 信号を得ることができる．なお，先頭エントリが無効化済みのライト・スカッシュ・バッファについては，メイン・レジスタ・ファイルにライトする必要はないため，サイズを 0 として扱う．

これらは，以下のように 3 ステージのパイプラインとして動作する．

- ステージ 1
  - 物理レジスタ番号 FIFO の先頭エントリの解放状態をフラグ・メモリから読み出す
  - 物理レジスタ番号 FIFO のサイズ及び解放状態からメイン・レジスタ・ファイルへライトするライト・

スカッシュ・バッファを選択

- ステージ 2
  - 先頭エントリが解放済みであったライト・スカッシュ・バッファに対応する Data FIFO を読み飛ばし
  - Write Select 信号で選ばれたライト・スカッシュ・バッファに対応する Data FIFO からデキュー
- ステージ 3
  - 選ばれたライト・スカッシュ・バッファに対応する FIFO の出力をメイン・レジスタ・ファイルにライト

ステージ 1 のライト・セレクタの動作は，図 6 の回路でも速度に問題はないと考えられるが，

(1) 大小比較を上位ビットのみ行い回路を簡素化する

(2) ライト・セレクタをパイプライン化する

といったように，デキュー選択の精度を落とすことで，高速化する選択肢も考えられる．

3.3 WSB の容量とエントリの寿命

本章の最後に，WSB に必要な容量について考察する．空き物理レジスタが割り当てられて上書きする命令のコミットによって解放されるのは，物理レジスタ・ファイルも WSB も同様である．そのため，WSB の総容量は物理レジスタ・ファイルと同程度になるのではないかと懸念を持たれる向きもあろう．しかし実際には，WSB の総容量は物理レジスタ・ファイルの半分程度でよい．それは，WSB のエントリの寿命が物理レジスタのそれより短いからである．

図 2 に，命令 I_{p1} に割り当てられた物理レジスタ P_1 とそれに対応する WSB エントリの寿命を示す．同図に示されているように，エントリが解放されるタイミングは物理レジスタと WSB エントリで同一である．一方で割当てのタイミングは，物理レジスタがリネーム・ステージであるのに対して，WSB ではライトバック・ステージになる．したがって，WSB のエントリの寿命は物理レジスタのそれより，ディスパッチ〜スケジュール〜発行〜レジスタ読み出し〜実行の分だけ短くなる．特に，スケジューリング・ウィンドウ中で発行を待っている時間が長くなる可能性がある．

この寿命の差の分だけ，WSB の容量は物理レジスタ・ファイルのそれに比べて小さくてよくなるのである．

なお，同様の考えは，物理レジスタ 2 段階解放方式 [10] にも見ることができる．

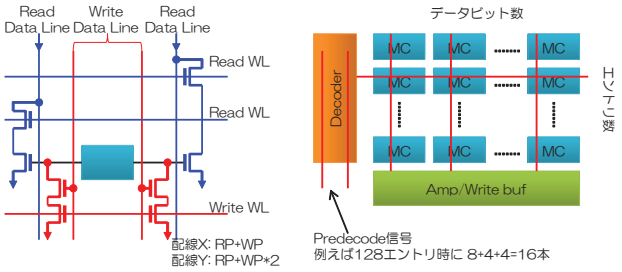

図 7 RAM のセル構造と配線本数

4. 面積及び消費電力

4.1 評価手法

4.1.1 概要

従来，RAM の面積及び消費電力の評価には CACTI [11] などが使用されていた．しかし，本稿で示すライト・スカッシュ・バッファは CACTI が計算の対象としていない 1*bit* × 128 エントリといった小規模で，ライトの代わりに Clear/Set を行う図 3 のような特殊メモリを含んでおり，算出が困難である．

従って，以下の簡易的な手法で相対的な面積及び消費電力を算出することとした．

面積 ポート数，エントリ数などから計算した配線本数で，面積が決まるとみなす

電力 アクセス時に同時に動作する配線の長さの合計に，電力が比例するとみなす

4.1.2 RAM のセル構造と配線本数

RAM の面積及び 消費電力の見積もりの根拠となるセル構造・配線本数を，**図 7** に示す．

デコーダ上の Y 方向配線は，プリ・デコード信号である．例えば，128 エントリの RAM であれば，8bit/4bit/4bit の 3 組のプリ・デコード信号合計 16bit を使い，3 入力 AND を利用してデコードを行う．8bit/4bit/4bit の 3 組の内，それぞれ 1 本が 1 状態で，特定のエントリが選択される．

アレイ上の Y 方向配線はデータ線であり，ビット幅数 × ポート (×2) 数の配線が存在し，1 ポートへのアクセス時には ビット幅本が動作する．ライトのデータ線が面積計算上と (×2) となるのは，相補信号であるためである．

アレイ上の X 方向配線はワード線であり，エントリ数 × ポート数の配線が存在し，1 ポートへのアクセス時にはエントリに対応した 1 本が動作する．

4.1.3 CAM のセル構造と配線本数

CAM の面積及び 消費電力の見積もりの根拠となるセル構造・配線本数を，図 7 に示す．

デコーダ上の Y 方向配線は，RAM と同様にプリ・デコード信号であり，比較値のライトに用いられる．

アレイ上の Y 方向配線はデータ線及び マッチ線である．この内データ線については RAM と同様である．サーチ線

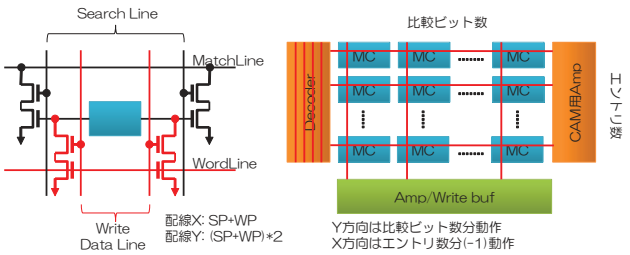

図 8 CAM のセル構造と配線本数

は 比較ビット数 × ポート × 2 の配線が存在する．ここで ×2 は，図 7 左側に示すようにサーチ線が相補信号であるためである．サーチ線は，1 ポートへのサーチ時には，相補信号の片側分，つまり，比較ビット分動作する．アレイ上の X 方向配線はワード線及び マッチ線であり，この内ワード線については RAM と同様である．

マッチ線については，エントリ数 × ポート数の配線が存在する．1 ポートへのサーチ時には全マッチ線がプリチャージされ，一致したエントリに対応した 1 本を除いて Low レベルに引き抜かれる．

4.1.4 見積もり手法のまとめ

以上をまとめると，**表 1**, **表 2**, **表 3** の値となる．本稿では，これらの値と，後述のシミュレーションによる各ポートの動作率から，面積及び 消費電力を算出した．

なお，後述の性能評価では，ライト・バッファは NORCS の実装を使用しているが，ポートの動作率が一部算出できないため，消費電力の算出の基礎となる動作率については，ほぼ等価となる，ライト・スカッシュ・バッファでスカッシュ動作を Disable とした状態のシミュレーション結果を用いている．

表 1 RAM の配線本数

項目	合計本数	動作本数
Y 配線 (dec.)	PreDec 信号 × ポート	各 PreDec 信号 1 本
Y 配線 (array)	ビット幅 × ポート (×2)	ビット幅数
X 配線 (array)	エントリ × ポート	1

表 2 CAM の配線本数

項目	合計本数	動作本数
Y 配線 (dec.)	PreDec 信号 × ポート数	各 PreDec 信号 1 本
Y 配線 (array)	比較ビット × ポート × 4	比較ビット
X 配線 (array)	エントリ × ポート	エントリ

表 3 Clear/Set RAM の配線本数

項目	合計本数	動作本数	
		set/clr	read
Y 配線 (dec.)	PreDec 信号 × ポート	各 PreDec 信号 1 本	
Y 配線 (array)	ビット幅 × リードポート	0	ビット幅
X 配線 (array)	エントリ × ポート	1	1

4.2 見積もり条件

面積及び消費電力の見積もり条件を**表 4** に示す．

この内 PRF は，NORCS が比較対象としていた，レジスタ・キャッシュを用いない，8R4W×128 エントリ ×2(INT/FP) の **PRF**(Pipelined Register File) である．見積もりは，PRF の面積・消費電力を基準 (100%) とした相対値で行う．

4.3 結果

面積の比較を**表 5** に，消費電力の比較を**表 6** に示す．本稿の提案する WSB の NORCS 比で見ると，面積は，合計では，MRF 2R1W で 20%減，2R/1W で 30%減，消費電力は，MRF 2R1W で 23%減，2R/1W で 30%減となっており，いずれも，MRF のポート数減少の寄与が大きい．

PRF と NORCS の比較

まず，基準となる PRF と，塩谷らが提案した NORCS に相当する”WB(2R2W)”を比較する．これらの値は，文献 [1] で示された，面積を 1/4 程度，消費電力を 1/3 程度と同等となることが期待されるが，いずれもやや多めである．

この差は，本見積りでは，Flag, WB 分の面積や消費電力を考慮していること，CACTI による見積もりでは，大面積のメモリについては高速化のための追加ドライバなどが挿入され，面積や消費電力が上振れする傾向があることなどが考えられる．

NORCS と WSB の比較

次に，NORCS に相当する”WB(2R2W)”と，本稿が提案するライト・スカッシュ・バッファを使った”WSB(2R2W, 2R1W, 2R/1W)”について比較する．

NORCS 比の面積は，合計では，MRF 2R1W で 20%減，2R/1W で 30%減となっており，主に，MRF の面積が削減されている．各成分を詳しく見ると，MRF の面積は，ポート数のみで決定されているため，面積削減効果は 2R1W, 2R/1W の場合のみに見られる．WB 又は WSB については，NORCS の 8 × 2 本に対して，本稿が提案するライト・スカッシュ・バッファは 16 × 6 本と大容量であるものの，全て 1W1R の小さいメモリで構成されているため，面積はほとんど変わらない．物理レジスタの状態 Flag は，NORCS が WB への滞在しているか否かの 1 ビットに対して，WSB は解放されているか否かのビットも必要なためほぼ 2 倍である．

表 4 見積もり条件

項目	PRF	WB	WSB
MRF のポート数	8R4W	2R2W	2R2W, 2R1W, 2R/1W
RC のポート数	無し	8R4W	8R4W
WB のポート数	無し	4R2W× 2	1R1W×6
WB 長	無し	8 × 2	16×6
演算器数	int:2, fp:2, mem:2		
物理レジスタ数	INT/FP=128/128 レジスタ		

NORCS 比の消費電力は, MRF 2R1W で 27%減, 2R/1W で 32%減となっており，主に，MRF の消費電力が削減されている．

5. 性能評価

5.1 評価手法

シミュレーションを行い，提案手法の性能について評価を行った．シミュレーションには本研究室で開発したシミュレータ「鬼斬弐」[12] を用いている．ベンチマークには，SPEC2006 の全プログラムについて，ref.0 パターンを使い，最初の 1G 命令をスキップし直後の 100M 命令を実行した．

5.2 評価条件

ライト・スカッシュ・バッファは，レジスタ・キャッシュ・システムを基礎として，性能への影響を最小限としたまま，メイン・レジスタ・ファイルへの書込みを減少させることを目標としている．例えば，従来のスカッシュされないライト・バッファを用いるレジスタ・キャッシュ・システムで 2R2W であったメイン・レジスタ・ファイルから，のライト・ポート数を減少させて，性能低下を最小限に抑えることができれば，ライト・スカッシュ・バッファの効果が示されたことになる．

このため，MRF 2R2W のスカッシュ無しのライト・バッファを基準として，MRF 2R2W, 2R1W, 2R/1W のライト・スカッシュ・バッファについて性能評価を行った．条件を**表 7** に示す．ここで，2R/1W とあるのは，リードとライトでデータ線を共有する，2R 又は 1W のいずれかが可能なメモリセルである．

表 5 面積比較

WB 形式	WB	WSB		
MRF ポート数	2R2W	2R2W	2R1W	2R/1W
MRF	13.0%	13.0%	6.6%	3.5%
RC	16.8%	16.8%	16.8%	16.8%
WB/WSB	1.6%	1.5%	1.5%	1.5%
Flag	0.09%	0.19%	0.19%	0.19%
合計	31.5%	31.5%	25.1%	22.0%
合計 (WB 比)	100.0%	100.1%	79.8%	69.9%

表 6 消費電力比較

WB 形式	WB	WSB		
MRF ポート数	2R2W	2R2W	2R1W	2R/1W
MRF	20.8%	20.5%	12.2%	11.0%
RC	14.9%	14.9%	14.2%	13.6%
WB/WSB	4.2%	2.6%	2.2%	2.1%
Flag	0.5%	0.6%	0.7%	0.7%
合計	40.4%	38.7%	29.3%	27.4%
合計 (WB 比)	100.0%	95.6%	72.6%	67.8%

5.3 結果

5.3.1 IPC

全プログラムの相対 IPC 平均値を**表 8** に，プログラム毎の相対 IPC を**図 10** に示す．相対 IPC の基準 (100%) は，スカッシュの無い WB を使用したレジスタ・キャッシュ・システムであり，塩谷らが提案した NORCS[1] で算出した IPC である．

全プログラムの平均

2R1W ではスカッシュのないライト・バッファ (NORCS) では，相対 IPC が 79.2%に落ち込むがライト・スカッシュ・バッファでは，94.5%と，性能低下は 5.5%にとどまる．更に，2R/1W までライト・ポートを減らしても，従来の NORCS の 2R1W よりも相対 IPC は 91.3%と高い．なお，2R/1W については，従来の NORCS の実装では評価が行われていないため，相対 IPC の値は省略する．

プログラム毎の相対 IPC

特定のプログラム (bzip2, leslie3d, namd, hmmer, h264ref, tonto, astar) で大きく性能が低下している．この点については,，さらに詳細な検討が必要であるが，**図 9** に示すように，Write/cycle と，相対 IPC にはある程度の相関がみられる．但し，右上の 3 点，libquantum, gamess, dealII の 3 プログラムは，比較的 Write/cycle が大きいにも関わらず，相対 IPC が低下していない．

5.3.2 スカッシュ率

次にスカッシュ率—ライトが WSB 滞在中にスカッシュされた比率—のデータを示す．

本稿の WSB は，フォワーディングを避けるために，WSB から積極的にライトをデキューする方針で設計されており，従って，MRF に十分なライト・ポートがある場合には，WSB の滞在時間は短く，ほとんどスカッシュされないことになる．

実際に，**表 9** を見ても，2R2W のスカッシュ率は 1.9%と低く，最もライト・ポートの少ない MRF 2R/1W の場合

表 7 シミュレーション・パラメーター

項目	WB	WSB
MRF のポート数	2R2W	2R2W, 2R1W, 2R/1W
WSB 長	8×2 本	16×6 本
演算器数	int:2, fp:2, mem:2	
論理レジスタ数	INT/FP=32/32 レジスタ	
物理レジスタ数	INT/FP=128/128 レジスタ	
MRF Latency	1	
RC Latency	1	

表 8 相対 IPC(基準: スカッシュ無し WB, MRF:2W2R)

MRF ポート数	相対 IPC	
	WB(NORCS)	WSB
2R2W	100.0%	100.5%
2R1W	79.2%	94.5%
2R/1W	-	91.3%

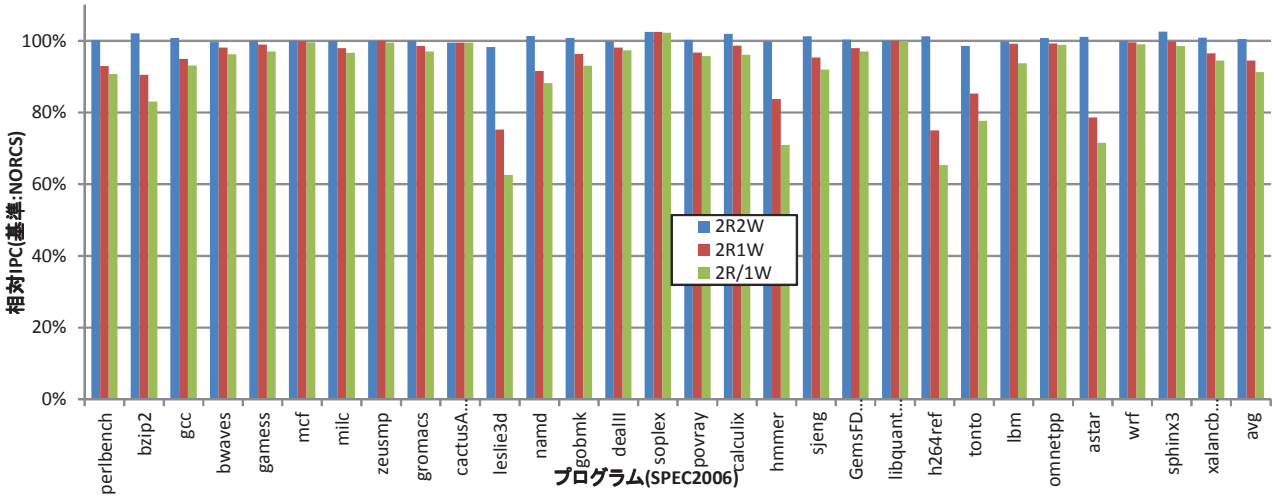

図 10 プログラム毎の相対 IPC(基準:スカッシュ無し WB)

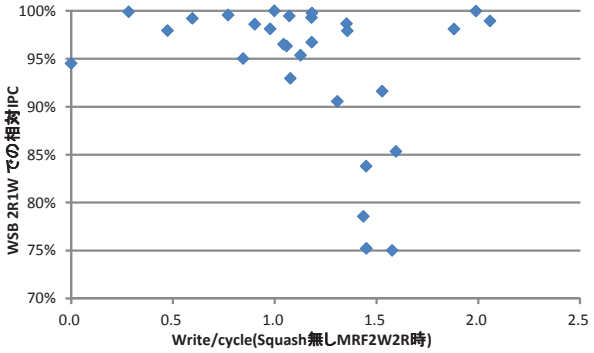

図 9 プログラム毎の Write/cycle 対 相対 IPC

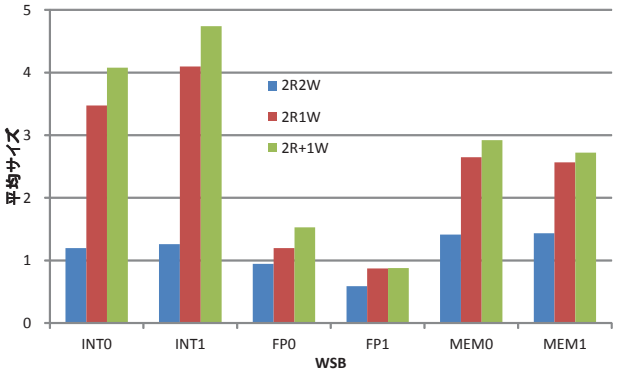

図 11 WSB 毎の WSB サイズ (平均)

には，スカッシュ率は 25.9% と高くなる．

5.3.3 WSB サイズ

次に，WSB サイズについて検討する．WSB のサイズは，エンキュー及びデキューのバランスで決まり，サイズが大きくなると，WSB の容量不足及び参照レジスタの WSB へのヒット確率が増加の 2 つの要因で，ストールが増加する可能性がある．

WSB サイズの性質

WSB では，デキューはライトへのライトだけでなく，物理レジスタの無効化によるスカッシュでも発生する．MRF のライト・ポートが少ない構成では，MRF へのライトを待つ間に WSB サイズが大きくなるが，WSB サイズが大きくなり，MRF へのライト滞在時間が長くなるほど，潰される確率が高くなる．従って，仮に容量無限大の WSB を用意しても，WSB サイズはある程度のサイズで飽和することが見込まれる．

表 9 スカッシュ率 (スカッシュ数/ライト数)

MRF ポート数	スカッシュ率
2R2W	1.9%
2R1W	20.1%
2R/1W	25.9%

シミュレーション結果

WSB 毎に見た WSB サイズについて，WSB 毎の平均サイズ及び，累積度数分布で 99%の点のサイズを，**図 11** 及び **図 12** に示す．また，最も平均サイズの大きい INT1 について，サイズの累積度数分布を**図 13** に示す．図 12 に示した”累積度数分布で 99%の点”とは，図 13 のような累積度数分布のグラフが 99%となる場合の WSB サイズである．

考察

図 12 に示したように，MRF のライトポート数が少ない構成では，INT0, INT1, FP0 の WSB サイズが増大し，累積度数 99%のサイズが 16 となる場合がある．

WSB の容量は各 16 エントリであるため，これを超えると，WSB Full によるストールが発生する．実際に，**表 10** に示すように，これらの条件はストールが発生している．但し，発生率はわずかであり，16 は，ほぼ十分な容量と考えられる．

他方，MEM0, MEM1, FP1 では，最もライト能力の低

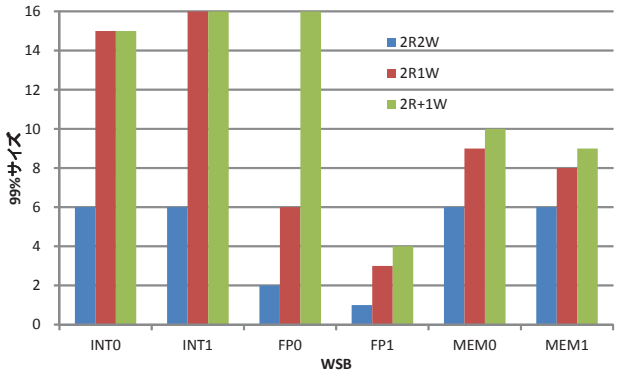

図 12 WSB 毎の WSB サイズ (累積度数 99%)

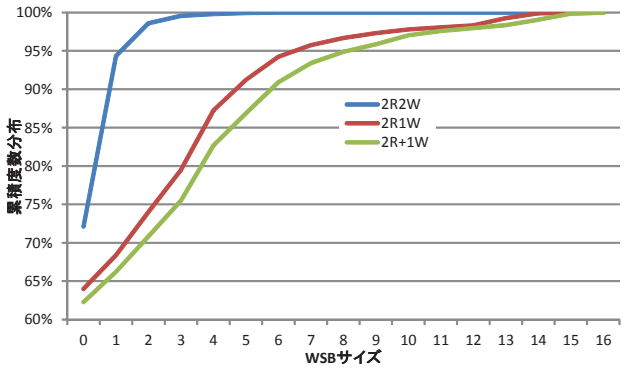

図 13 WSB サイズの累積度数分布 (INT1)

い 2R/1W の構成でも，16 に到達していない．これは，演算器によってライトの発生頻度が異なるためと考えられ，WSB 容量は一律とするより，演算器のライト頻度に応じて，WSB 毎に容量を変える設計が効率的であると考えられる．

6. おわりに

本稿では，レジスタ・キャッシュ・システムにおいて，ライトデータをライト・スカッシュ・バッファに一旦置き，ライト・スカッシュ・バッファ滞在中に発生する論理レジスタへのオーバーライトを利用して，メイン・レジスタ・ファイルへの書込みを削減する方法を提案した．フォワーディングを行わない構成としたことで，CAM などによる連想検索は不要となり，小面積のフラグ・メモリでもライト・スカッシュ・バッファを実現可能であることを示した．

今後の課題を 4 点述べる．

第一に，ライト・ポート数を少なくした構成で，特定のプログラムによる性能低下が目立つため，この点の対策が可能であるか検討する必要がある．

第二に，本稿では WSB の連想検索を完全に排除してフォワーディングを無くした設計であるが，完全に排除するのではなく，例えば，フォワーディングを行う WSB とストールに頼る WSB のハイブリッド構成なども検討する必要がある．

第三に，前項に関連するが，演算器毎の WSB 利用状況にかなりの差があり，最適な組み合わせを検討する必要がある．

最後に，物理レジスタ数の多いマルチスレッド・プロセッサなど，より大規模な物理レジスタ・ファイルを持つプロセッサへの対応が挙げられる．マルチスレッド・プロセッサでは，スレッド数分に対応してサイズの大きい物理レジスタ・ファイルを持つ．このため，メイン・レジスタ・ファイルへの書込み電力が大きく，ライト・スカッシュ・バッファでより大きな効果が得られることが期待される．

表 10 WSB Full ストール率
(ストール・サイクル/実行サイクル)

MRF ポート数	WSB Full ストール率
2R2W	0.00%
2R1W	0.03%
2R/1W	0.13%

謝辞 本論文の研究は一部，文部科学省科学研究費補助金 No. 23300013 による．

参考文献

[1] Shioya, R., Horio, K., Goshima, M. and Sakai, S.: Register Cache System Not for Latency Reduction Purpose, *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 301 –312 (online), DOI: 10.1109/MICRO.2010.43 (2010).

[2] 西川卓，倉田成己，塩谷亮太，五島正裕，坂井修一：レジスタ・キャッシュのマルチスレッド・プロセッサへの適用，情報処理学会第 75 回 全国大会 (2013).

[3] 西川卓，倉田成己，塩谷亮太，五島正裕，坂井修一：マルチスレッド・プロセッサにおけるレジスタ・キャッシュ・システムの評価，情報処理学会研究報告 (2013).

[4] Yuffe, M., Knoll, E., Mehalel, M., Shor, J. and Kurts, T.: A fully integrated multi-CPU, GPU and memory controller 32nm processor, *2011 IEEE International Solid-State Circuits Conference(ISSCC). Digest of Technical Papers*, pp. 264 –266 (online), DOI: 10.1109/ISSCC.2011.5746311 (2011).

[5] Agarwal, A., Hsu, S., Mathew, S., Anders, M., Kaul, H., Sheikh, F. and Krishnamurthy, R.: A 32nm 8.3GHz 64-entry x 32b variation tolerant near-threshold voltage register file, *2010 IEEE Symposium on VLSI Circuits (VLSIC)*, pp. 105 –106 (online), DOI: 10.1109/VLSIC.2010.5560334 (2010).

[6] Zhang, K., Bhattacharya, U., Chen, Z., Hamzaoglu, F., Murray, D., Vallepalli, N., Wang, Y., Zheng, B. and Bohr, M.: A 3-GHz 70MB SRAM in 65nm CMOS technology with integrated column-based dynamic power supply, *2005 IEEE International Solid-State Circuits Conference(ISSCC). Digest of Technical Papers.*, pp. 474 –611 Vol. 1 (online), DOI: 10.1109/ISSCC.2005.1494075 (2005).

[7] Yamaoka, M., Maeda, N., Shinozaki, Y., Shimazaki, Y., Nii, K., Shimada, S., Yanagisawa, K. and Kawahara, T.: Low-power embedded SRAM modules with expanded margins for writing, *2005 IEEE International Solid-State Circuits Conference(ISSCC). Digest*

of Technical Papers., pp. 480–611 Vol. 1 (online), DOI: 10.1109/ISSCC.2005.1494078 (2005).

[8] 山田淳二，倉田成己，塩谷亮太，五島正裕，坂井修一：レジスタ・キャッシュ・システムの省電力化手法，情報処理学会研究報告 (2013).

[9] Yeager, K.: The Mips R10000 superscalar microprocessor, *Micro, IEEE*, Vol. 16, No. 2, pp. 28–41 (online), DOI: 10.1109/40.491460 (1996).

[10] Hyodo, K. and Iwamoto, K.: Energy-efficient pre-execution techniques in two-step physical register deallocation, *IEICE transactions on information and systems*, Vol. 92, No. 11, pp. 2186–2195 (2009).

[11] Thoziyoor, S., Ahn, J., Monchiero, M., Brockman, J. and Jouppi, N.: A Comprehensive Memory Modeling Tool and Its Application to the Design and Analysis of Future Memory Hierarchies, *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, pp. 51 –62 (online), DOI: 10.1109/ISCA.2008.16 (2008).

[12] 塩谷亮太，五島正裕，坂井修一：プロセッサ・シミュレータ「鬼斬弐」の設計と実装，先進的計算基盤システムシンポジウム SACSIS，pp. 120–121 (2009).