Vol.2013-ARC-206 No.3
2013/7/31

レジスタ・キャッシュ・システムの省電力化手法

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は，スーパスカラ・プロセッサの性能向上の阻害要因となっている．
レジスタ・キャッシュを導入すると，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．しかし従来のレジスタ・キャッシュでは，リードの電力を減らすことはできるものの，ライトの電力を減らすことはできなかった．
本稿で提案するライト・スカッシュ・バッファは，上書きされたレジスタ・ファイルへのライトを省略することで，ライトの電力を削減する．
シミュレーションにより，16 エントリのバッファによって，ライトの数を 18% にまで，電力を 21% にまで削減できることが分かった．

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．

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．

[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

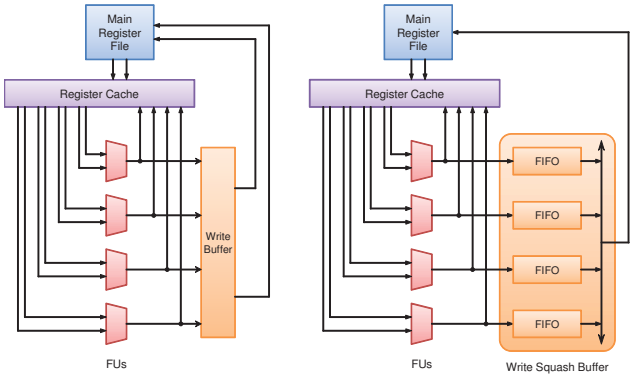

図 1 レジスタ・キャッシュ・システムのブロック図

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

本稿では，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**）を提案する．

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．

一方 WSB は，その容量の許す限り MRF へのライトを保持し，ライトが「潰される (squash)」のを待つ．

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．同一の論理レジスタに対する上書きを行う後続の命令がコミットされると，上書きされる論理レジスタにマッピングされた物理レジスタは解放される．この時，解放される物理レジスタへのライトが WSB 上にあれば，それも削除してよい．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，後続の命令がコミットされていくと，「潰した」ライトもまた「潰される」ことになる．こうして，バッファに保持しきれなくなったライトだけを MRF に転送すればよい．4 章で述べる評価では，MRF へのライトは 16 エントリの WSB により，18%程度にまで削減可能であることが示される．

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る．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．

レジスタ・ファイル系の電力は，NORCS の導入により 1/3 程度にまで削減されているが，WSB の導入により更にその半分程度にまで削減されることになる．

更に，MRF へのライトの削減により，MRF のポートを削減できる可能性もある．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，回路面積は更に半分程度にまで削減することができる．

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，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 [8].

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_2 がコミットされた時点で，I_1〜I_2 の間にある命令（I_{c1}）も完了しており，命令 I_1 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．

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

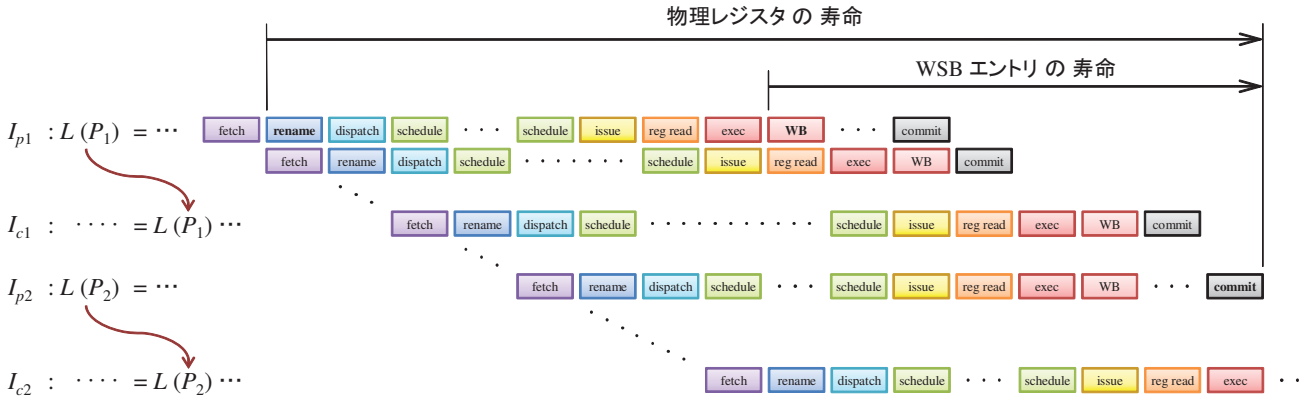

図 2 物理レジスタ・ファイルと WSB エントリの寿命

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．

無効化

対応する物理レジスタが解放された時には，連想検索により，エントリを無効化する．

デキュー

通常のライト・バッファは，バッファ内に有効なエントリが 1 つでもあれば，MRF に対してリクエストを送出する．WSB は，できる限りエントリをバッファ上に保持しようとするため，有効なエントリがあるだけでは，MRF に対してリクエストを送出しない．

バッファ・フルによるストールを避けるため，残りが n エントリを切ったところで，MRF に対してリクエストを送出する．n は 2〜4 程度である．

デキュー側の各サイクルの動作は以下のようにまとめられる：

(1) 先頭エントリが無効化されていれば，これを取り除き，このサイクルの動作は終了する．
(2) 先頭エントリが有効で，かつ，残りエントリ数が n を切ったところで，MRF に対してリクエストを送出する．各演算器のバッファ間で調停を行い，認められれば，先頭エントリのライトを MRF に対して送出する．

3.3 その他の考慮点

本節では，その他の実装上の考慮点について述べる．

ペイロードの実装方法

演算器ごとのバッファのペイロードは，以下の 2 つの実装方法が考えられる：

リング・バッファ型 1-read/1-write の RAM を用いてリング・バッファを構成する

シフト・レジスタ型 全体をシフト・レジスタによって構成し，エンキュー/デキューの対象を末尾と先頭のレジスタに限定する．

機能・性能に大きな違いはないと考えられる．

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．

WSB は，既存のライト・バッファより大容量で，できる限りライトを保持し続けようとするため，ライト・バッファ・ヒットの確率はより高い．

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，MRF から読み出す．当該ライトのあるバッファを優先的に処理すれば，10 サイクル程度以内に書きだされる．

なおリング・バッファ型では，フォワーディングのためのポートを別途追加する必要はなく，デキューを停止して，デキュー用のポートから読み出しを行えばよい．

コンパクション

無効化されたエントリは「歯抜け」になるので，コンパクションを施すことにより，エントリの有効利用を図ることが考えられる．

同様の考え方は，命令ウィンドウ [9] にも見られる．

ただしコンパクションは，リング・バッファ型の実装では実現がやや難しい．

3.4 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

表 1　シミュレーション・パラメーター

項目	値
演算器数	int:2, fp:2, mem:2
論理レジスタ数	INT 命令に対して 32 レジスタ
物理レジスタ数	INT 命令に対して 128 レジスタ
MRF のポート数	2R2W
WSB 長	8,16,24,32,48,64,96
WSB の実装	リング・バッファ方式
WSB のリード・ポート数	1,2,4
WSB の空き	1,2,4

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．

4. 性能評価

4.1 評価手法

シミュレーションを行い，提案手法の評価を行った．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」[11] を用いている．ベンチマークには，SPECINT2006 から，perlbench, bzip2, gcc, mcf, gobmk, hmmer, sjeng, libquantum, h264, omnet, astar, xalancbmk 各プログラムの ref プログラムを使用し，最初の 1G 命令をスキップし直後の 100M 命令を実行した．プログラム毎の差は 4.4 に示し，以下，特に明記しない限りは，全プログラムの平均値を結果として示す．他の主要なパラメーターを**表 1** に示す．

4.2 性能

この節では，ライト・スカッシュ・バッファの性能への影響についてまとめる．ライト・スカッシュ・バッファは，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おり，顕著な性能低下がみられないことが期待値である．

後述の結果からは，リード・ポート数は最小の 1，ライト・スカッシュ・バッファの空き数も最小の 1 でも，IPC は 2%しか低下せず，ほとんど影響を与えないことが分かった．

ライト・スカッシュ・バッファで性能に影響を与える要素として考慮されるのは，以下の 2 点である．

リード・ポート数　不足するとメイン・レジスタ・ファイルとフォワーディングが衝突してストールを発生

ライト・スカッシュ・バッファの空き数　不足すると一時的なライト数のピークでライト・スカッシュ・バッファフルとなってストールが発生

続く 4.2.1 及び 4.2.2 で，それぞれの影響を詳細に考察し，4.2.3 で IPC への影響をまとめる．

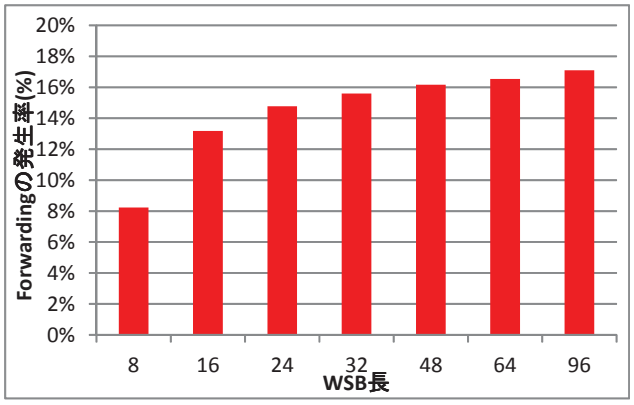

図 3　ライト・スカッシュ・バッファからのフォワーディング発生率

4.2.1 フォワーディング起因ストール

リード・ポート数が影響すると考えられる，フォワーディング起因ストールについて，**図 3** に結果を示す．図 3 の横軸はライト・スカッシュ・バッファの長さ，縦軸は書込み数を分母に取ったフォワーディング発生率であり，ライト・スカッシュ・バッファへ書込まれたデータが全てフォワーディングされれば，100%となる．

ライト・スカッシュ・バッファの長さが長くなるほどフォワーディングの発生率は上がっているが，長さ 96 のライト・スカッシュ・バッファでも 17%程度と，フォワーディングの発生率は飽和する傾向がある．これは，古いデータが参照される確率は低いためと考えられる．

次に，**図 4** に，フォワーディングとメイン・レジスタ・ファイルへの転送によって，ライト・スカッシュ・バッファのリード・ポートが競合してストールが発生した割合を示す．図 4 の横軸は同様にライト・スカッシュ・バッファの長さ，縦軸は，ストールが発生した回数を，実行サイクル数を分母に取った比率で示している．

ストールの発生率はリード・ポート数 1 でも十分低く，ライト・スカッシュ・バッファをリング・バッファ実装する場合のリード・ポート数は小さくて良いことが分かる．ここで，フォワーディングの発生率が低いライト・スカッシュ・バッファ長:8 で最もストール発生率が高いのは，メイン・レジスタ・ファイルへの書込み発生率が高く，メイン・レジスタ・ファイル書込みのための読出しが比較的多いためと考えられる．

4.2.2 ライト起因ストール

ライト・スカッシュ・バッファの空き数が影響すると考えられる，ライト起因ストールについて，**図 5** に結果を示す．

図 5 の横軸はライト・スカッシュ・バッファの長さ，縦軸は，ストールが発生した回数を，実行サイクル数を分母

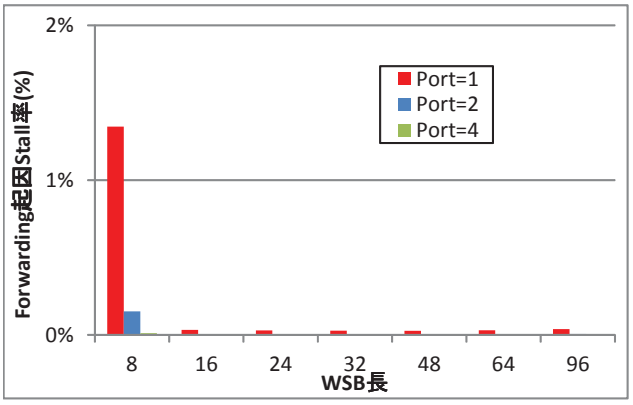

図 4　フォワーディングによるストール発生率

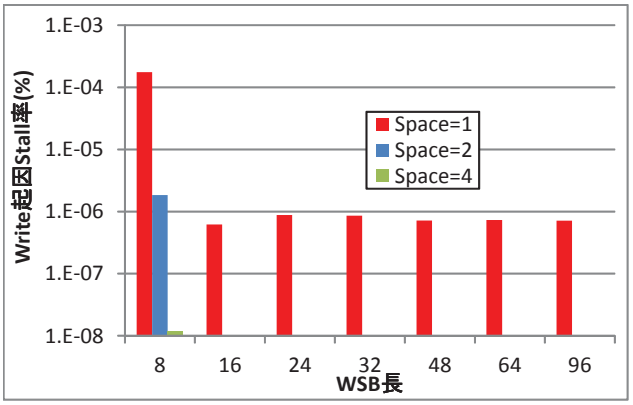

図 5　ライトによるストール発生率

に取った比率で示している．

ライト・スカッシュ・バッファの空きは 1 でも 1E-3 以下と十分にストール率は低く，ライト・スカッシュ・バッファの空きを 2 に増やすとさらにストールは減少する．

4.2.3　IPC

4.2.1 及び 4.2.2 で考察したこれらのストールは，最終的に IPC の低下という形でプロセッサの性能を低下させると考えられる．

図 6 に，ライト・スカッシュ・バッファのリード・ポート数とライト・スカッシュ・バッファの空き数を変えた場合の IPC への影響を示す．図 6 の横軸は，それぞれ 3 通りであるリード・ポート数とライト・スカッシュ・バッファ数の組合せ 9 通りであり，縦軸はリード・ポート数:4, ライト・スカッシュ・バッファの空き:4 の最もストールが少ない条件を 100%とした相対 IPC である．

図 6 からは，最大でも IPC は 2%しか低下せず，ライト・スカッシュ・バッファのリード・ポート数: 1,2,4，ライト・スカッシュ・バッファの空き: 1,2,4 の範囲では大きな影響は見られないと言える．

4.3　ライト電力

この節では，ライト電力についてまとめる．後述の結果から，比較的小容量の長さ 16 のライト・スカッシュ・バッファでも，ライト発生率は 18%まで下がり，ライト電力は 21%に削減されることが分かった．

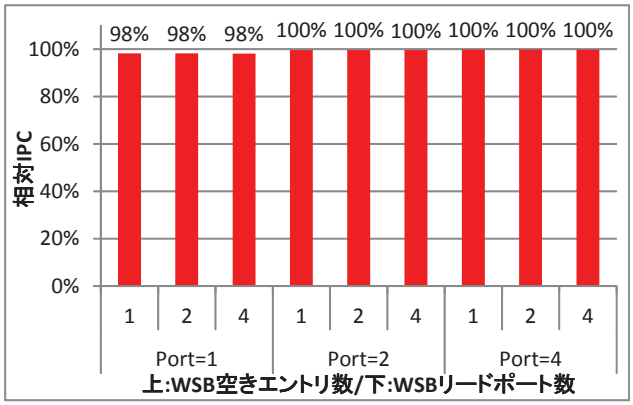

図 6　リード・ポート数，ライト・スカッシュ・バッファの空きを変えた場合の IPC の変化

続く 4.3.1 ではライト・スカッシュ・バッファによるライトの削減効果を示し，4.3.2 では，その結果からライト・スカッシュ・バッファのライト電力削減効果を見積もった過程を示す．

なお，前節の検討の結果，ライト・スカッシュ・バッファのリード・ポート数: 1,2,4，ライト・スカッシュ・バッファの空き: 1,2,4 の範囲で，性能に大きな影響は見られないと判断できたため，本節では，全て，面積が最小となるリード・ポート数: 1，ライト・スカッシュ・バッファの空き数: 1 の結果を示している．

4.3.1　メイン・レジスタ・ファイルへのライト発生率

図 7 に，ライト・スカッシュ・バッファから，メイン・レジスタ・ファイルへのライトの発生率を示す．図 7 の横軸は，ライト・スカッシュ・バッファの長さ，縦軸は，ライト・スカッシュ・バッファへの書込み数を分母とした，メイン・レジスタ・ファイルへの書込み数を示している．ライトが一切ライト・スカッシュ・バッファで潰されずに全てメイン・レジスタ・ファイルに書き込まれた場合に，ライトの発生率は 100%となり，小さいほど良い結果である．

図 7 から，ライト・スカッシュ・バッファ長が長くなるほど，ライト発生率は下がることがわかる．これは，ライト・スカッシュ・バッファ滞在中に上書きが発生する確率が高まるためであると考えられる．しかし，比較的小容量の長さ 16 のライト・スカッシュ・バッファでも，ライト発生率は 18%まで下がっている．

4.3.2　ライト電力

この結果を書込み電力に換算した結果を**図 8** に示す．

図 8 の横軸は，ライト・スカッシュ・バッファの長さ，縦軸はライト・スカッシュ・バッファ導入前のメイン・レジスタ・ファイルへの書込み電力を 100%とした，メイン・レジスタ・ファイルとライト・スカッシュ・バッファの書込み電力の合計電力を表している．図 8 から，長さ 16 の

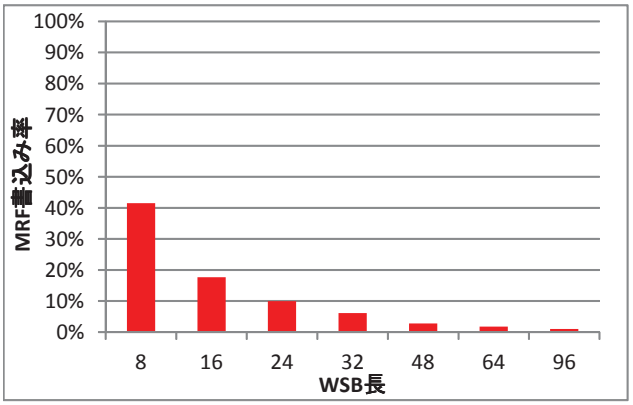

図 7 ライト・スカッシュ・バッファからメイン・レジスタ・ファイルへのライト発生率

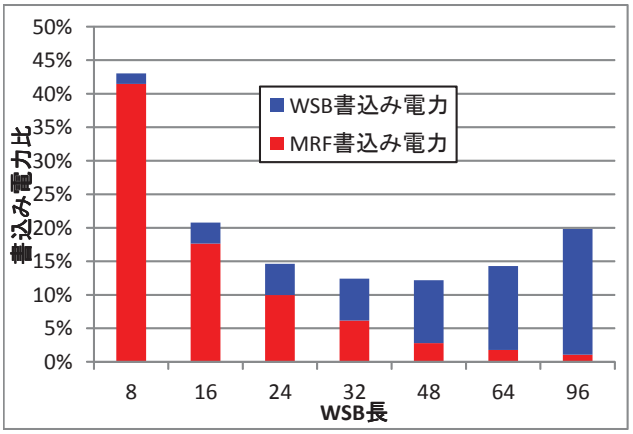

図 8 ライト・スカッシュ・バッファの導入によるライト電力の変化

ライト・スカッシュ・バッファで，ライト・スカッシュ・バッファ導入前の 21%に削減できることが読み取れる．

図 8 は，以下に示す計算式で算出した．

MRF 電力比 ＝ ライト発生率

WSB 電力比 ＝ エントリ数比 · ポート数比^2

この計算では，メイン・レジスタ・ファイルのライト・ポートはベース・システムの 2 から減少せず，1 ライトあたりエネルギーは減少しない前提としているが，仮に，ライトの削減に伴い，メイン・レジスタ・ファイルのライト・ポートを減少させることが出来れば，メイン・レジスタ・ファイル分の電力はさらに削減することができる．

4.4 ベンチマーク・プログラムによるライト発生率の違い

最後に，プログラムによるライト発生率の違いを，**図 9** に示す．いずれも，ライト・スカッシュ・バッファの長さ:16，面積が最小となるリード・ポート数: 1，ライト・スカッシュ・バッファの空き数: 1 の結果である．プログラム毎にメイン・レジスタ・ファイル書込み率は異なり，libquantum ではメイン・レジスタ・ファイルへの書込みはほとんど発生せず，hmmer では比較的書込み発生率が高い．プログラムによる差異は，今後の検討課題である．

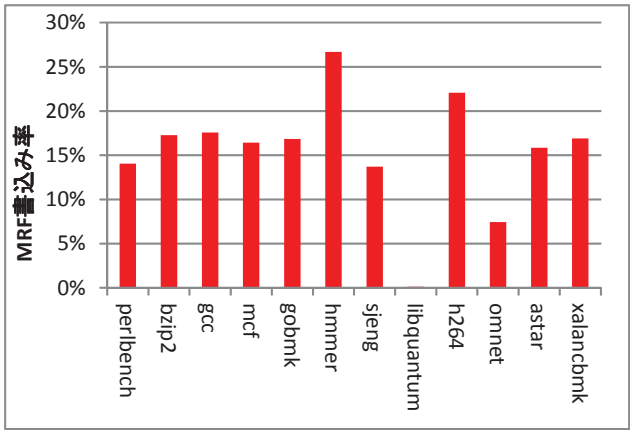

図 9 プログラムによるライト発生率の違い

5. 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．シミュレーションの結果によれば，物理レジスタ数 128 に対して，リング・バッファ構成の長さ 16 のライト・スカッシュ・バッファでも書込みを 18%と，効果的に削減することが可能であった．この場合のライト電力は，メイン・レジスタ・ファイルとライト・スカッシュ・バッファを合わせても，従来の 21%に削減できることが分かった．

今後の課題を 4 点述べる．

第一に，ライト・スカッシュ・バッファをシフト・レジスタ構成とすることの検討が挙げられる．ライト・スカッシュ・バッファが備えるべきアクセス自由度は，出入り口を 1 つずつ持ち空きを許さない純粋なシフト・レジスタは制約が強すぎ，ランダムアクセス可能な SRAM で構成されたリング・バッファの自由度は必要以上と考えられる．従って，適切な制約を課すことで，回路構成を簡素化することができれば，ライト電力のさらなる削減が可能が期待できる．

第二に，物理レジスタ数の多いマルチスレッド・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．

第三に，プログラムによるメイン・レジスタ・ファイル書込み率の差についても調査が必要である．

最後に，メイン・レジスタ・ファイルのライト・ポート数を更に削減できるかの検討が必要である．

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

[9] Farrell, J. and Fischer, T. C.: Issue logic for a 600-MHz out-of-order execution microprocessor, *Solid-State Circuits, IEEE Journal of*, Vol. 33, No. 5, pp. 707–712 (online), DOI: 10.1109/4.668985 (1998).

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

[11] 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).