Vol.2012-ARC-201 No.15
2012/8/1

アイドル時キャッシュ電源遮断における性能ペナルティ削減手法の実装

有間 英志[1]　薦田 登志矢[1]　三輪 忍[1]　中村 宏[1]

概要：プロセッサがアイドル時に消費するリーク電力が全消費電力に占める割合は，トランジスタの微細化が進むにつれて年々上昇を続け問題となっている．このようなリーク電力を削減する目的で，アイドル時に OS の判断によりコアへの電源供給を遮断するパワーゲーティング技術が広く用いられている．しかし，キャッシュの電源を遮断した場合には格納されていたデータが揮発するため，電源復帰後に失われたデータを参照した場合，それによるキャッシュミスが性能低下を引き起こす．そのため，本研究ではキャッシュの電源を遮断する場合においても，タグアレイには通電させておき，電源復帰後にタグを用いてデータを復帰させる技術を提案する．また，無駄なデータ復帰を防ぐため，再利用されるデータの識別手法についての検討を行い，予備的な評価を行う．

Eishi Arima[1]　Toshiya Komoda[1]　Shinobu Miwa[1]　Hiroshi Nakamura[1]

1. はじめに

1.1 背景と目的

プロセッサがアイドル時に消費するスタティックな電力は，トランジスタの微細化が進むにつれて指数関数的に増加し続けており，その傾向は今後も続くものと予想されている [1]．また，一般的な PC ユーザの使用状況では，アイドル時間がほとんどを占めているため，プロセッサが無駄なスタティック電力を消費する場面は多い [2]．

リーク電力を削減するための回路技術として，パワーゲーティング (Power Gating :以下，PG) と呼ばれる回路技術が知られている．PG 技術では，処理を行っていない回路への電源電圧供給を遮断することで当該回路に流れるリーク電流を削減し，消費リーク電力を大きく削減することができる．このような PG 技術は，バッテリーによるエネルギー制約のあるモバイル用途のプロセッサをはじめ，デスクトップ向けのマルチコアプロセッサにも搭載されている [3]．

現行のプロセッサでは ACPI [4] 等の電力制御規格に従って，プロセッサコアの省電力モードを動的に制御している．ACPI に準拠するするプロセッサでは，複数のスリープモードが用意されており，アイドル時間が長くなるにつれて，より深いスリープモードへと移行していく．このとき，より深いスリープモードに入ることにより多くのリーク電力を削減することが可能になる一方，スリープからの復帰にかかる時間が長くなるというトレードオフ関係が存在する．具体的にはコア内部の演算部，クロック生成器，キャッシュといったプロセッサの構成要素を復帰する際のペナルティが小さい物から順に電源を遮断していくことになる．

システム稼働時におけるプロセッサコアのスリープにおいて問題になるのが，スリープからの復帰に要するウェイクアップ時間である．スリープモードのプロセッサは，入力デバイス等からの割り込みによって電源を復帰させ，アクティブなモードに移行する．そうして，要求された処理を実行することになる．このように電源を復帰してから処理を実行可能になるまでの時間が存在するため，電源遮断によってアプリケーションの性能は低下してしまう．そのため，現行のシステムでは十分に長いアイドルがプロセッサに生じたときのみ深いスリープモードに入るようなモード制御が行われている．逆にスリープからの復帰に要するウェイクアップ時間を削減することができれば，性能低下を引き起こすことなく従来よりも短いアイドル期間において深いスリープモードへと移行することが可能となる．このような目的のもと，パワーゲーティングによるスリープモードからの復帰時間を削減する研究が多くなされてき

[1] 東京大学
The University of Tokyo

た [5].

ウェイクアップ時間削減に関するこれまでの研究では，主に回路的な工夫によってスリープモードからの復帰時間を削減するものであった．一方，プロセッサに含まれるキャッシュの電源を遮断した場合，キャッシュに格納されていたデータがすべて揮発してしまうため，スリープをしなかった場合に存在しないキャッシュミスが発生するというアーキテクチャ的な性能オーバーヘッドが存在する．このペナルティを削減できれば，電源遮断にともなう性能低下を抑えることができる．結果として，キャッシュの電源を遮断する機会が広がり，さらなる電力削減へと繋がることが期待できる．

この様な性能ペナルティに対処するため，STT-MRAM(Spin Transfer Torque Magnetic RAM) といった，データの保持に電源供給がいらないような不揮発性メモリを，キャッシュに用いる研究が盛んに行われている [6][7]．しかし，このような不揮発性メモリは，通常キャッシュに用いられる SRAM に比べてアクセス性能が低いため，プロセッサの性能が低下するという問題がある．特に高負荷時の性能が低下してしまうことが問題となる．

そこで本研究では，スリープ復帰時に復帰後に使用される可能性が高いデータをバースト的にプリフェッチすることでキャッシュの電源遮断に伴うキャッシュミス増加に伴うウェイクアップ時間の削減手法を検討する．このようなプリフェッチを可能にするため，文献 **??**においてキャッシュのデータアレイのみスリープさせ，タグデータはスリープさせず復帰時のプリフェッチに利用するというスリープ方式が提案されている．しかし，文献 **??**では復帰後に再利用されるデータが完全に分かるという仮定のもと，ウェイクアップ時間削減の効果を見積もっており，具体的なデータのプリフェッチ手順については考慮されていない．そこで，本稿ではタグを用いたスリープ復帰時におけるプリフェッチ手法をより詳細に検討するため，具体的なプリフェッチ手順とそれによる効果を検討する．具体的には，スリープ前のアクセス履歴をもとにして，ページ単位・プロセス ID 単位でデータを復帰するプリフェッチアルゴリズムを提案し，その効果を見積もる．

以下，まず次章ではキャッシュの電源遮断による性能ペナルティを削減する具体的な手法について述べ，3 章ではその評価方法の説明と評価結果について述べ，4 章では関連研究について述べ，5 章ではまとめと今後の課題について述べる．

2. 性能ペナルティ削減手法の検討

2.1 タグを用いたデータアレイの復帰

本稿では，STT-RAM のようにスリープ中もキャッシュデータを保持するのではなくスリープ復帰時に再利用されるデータだけを選択的にプリフェッチすることで，スリープに伴うキャッシュミス増加を抑制する手法を考える．この手法は，STT-RAM のような不揮発性メモリを使用したキャッシュミス発生の抑制手法に対して， SRAM 等，高速な揮発性メモリデバイスを用いて実装することができる点が利点となる．

スリープ復帰時におけるプリフェッチを可能にするためには，スリープ後に利用されるデータを予測するためスリープする前に保持していたキャッシュデータのメモリアドレスを，スリープをまたいで保存しておく必要がある．提案手法では，既存キャッシュのタグアレイとデータアレイのパワードメインを分離し，面積的に大部分を占めるデータアレイのみスリープさせ，タグアレイはスリープさせずタグ内の情報を保持しておくことでこの問題に対処する．

図 1 にタグアレイに残された情報を用いたスリープ復帰時のプリフェッチの概要を示す．ここでは，キャッシュの電源復帰直後でデータアレイは揮発し，タグアレイのみデータが残っている状態を考える．新しいハードウェアとしてプリフェッチャを用意する．プリフェッチャは，復帰させたいラインの位置を出力するものであり，セットへのインデックスとウェイ番号で指定する．これを使って，復帰させたいラインのタグにアクセスし，これとプリフェッチャの出力であるインデックスとを組み合わせることで，ブロックアドレスを割り出すことができる．これを用いて下位のメモリにアクセスし，データを復帰させる．

キャッシュの電源遮断によるデータ損失に伴う性能ペナルティを削減するため，タグアレイの情報をキャッシュの電源遮断時にも残しておき，電源復帰時に，今後再利用されるラインについてのみ，タグアレイを参照してデータを復帰させる手法を提案する．タグのデータとラインから電源遮断前に存在していたデータのアドレスが分かるため，タグのデータがあれば電源遮断直前のキャッシュを復元することができる．また，タグの bit 数は 32bit メモリ空間では 20bit 程度であり，ラインサイズを 64B と仮定すると，タグの容量はキャッシュ全体の 4 パーセント程度であり，タグアレイのみパワードメインを分離しておき，電源を供給し続けた場合でも，リーク電力の大部分は削減できているものと考えられる．また，タグアレイのみ STT-MRAM のような不揮発性メモリを用いるということも考えられる．タグへのアクセス性能は低下するが，L2 以下のキャッシュ階層では，タグへのアクセス自体がそれほど頻繁には起きないことを考えると，それ程性能に影響を与えないと考えられる．

2.2 数理モデル

プリフェッチャは，理想的には，再利用されるラインとそうでないラインを識別し，再利用されるラインについてのみ，その位置を出力すべきである．これは，キャッシュ

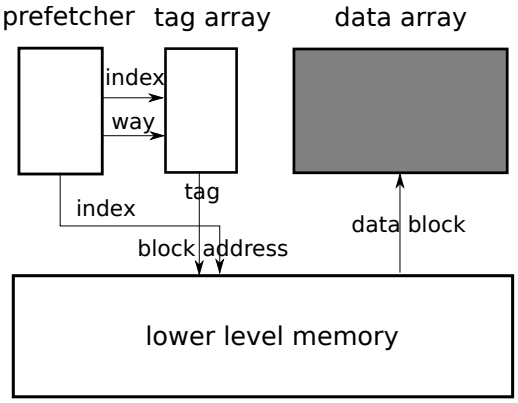

図 1 データ復帰手順の概要

の中には再利用されないラインも存在し，その様なラインについてデータを復帰させるのはエネルギー的にも性能的にも無駄であるからである．ここでは，プリフェッチすべきデータを決定する問題を数理的なモデルを用いて定式化し本稿で取り組む問題を整理する。

まずキャッシュを N 個の断片 $f_i(1 \leq i \leq N)$ に分割する．電源復帰後の f_i の容量を $C(f_i)$ で表し，f_i の中で再利用されるデータの容量を $R(f_i)$ で表す．プリフェッチャは電源復帰後に以下の問題を解決すれば良い．

$$\begin{aligned} &\max \sum_{i=1}^{N} \alpha_i R(f_i) \\ &s.t. \sum_{i=1}^{N} \alpha_i C(f_i) \leq C \\ &\alpha_i \in \{0,1\} \end{aligned}$$

ただし，C は前述のエネルギー制約から来る定数である．これは，不必要なデータを復帰させてもよいが，復帰させられるデータの容量には制限が有り，その制約下で再利用されるデータの復帰を最大化する問題である．一見するとナップサック問題と同等の式で表されているが，異なる点は，$R(f_i)$ が定数として与えられておらず，予測する必要があることである．また，断片数 N も可変であり自由に決めることができる点もナップサック問題と異なる点である．さらに，電源復帰直後にデータがすぐに必要になるような場面では，再利用されるデータから順に復帰させなければならないという状況も考えられる．この様な場面ではさらに，f_i の順序付けが必要となる．この様に，ナップサック問題と異なる部分も存在しているが，ナップサック問題で有効な手法は本問題でも有効であると考えられる．例えば，ナップサック問題では，ヒューリスティックな解法として，貪欲法が有効であるが本問題でも同様に，データ復帰手順が問題にならない場面では $R(f_i)/C(f_i)$ が高いもの，すなわち再利用率が高いものを順に復帰させると良いと考えられる．

本稿では，特にどの様な空間粒度でキャッシュを断片に分割すれば良いかについて議論する．また，分割が与えられたとしていくつかの手法を上記の問題に適用し，どの程度再利用されるデータを選択できるかについて検証する．

2.3 プリフェッチ時におけるデータ転送の粒度

スリープ復帰時のプリフェッチを考えた場合，再利用されるデータはできる限り早くキャッシュに戻すべきである．このため，再利用されるであろうデータを予測するまでの時間は短ければ短いほど良く，従ってウェイクアップ時間の短縮という意味では，上記のキャッシュ分割の粒度は粗い方が適していると考えられる．一方で，空間粒度のプリフェッチを行った場合実際には再利用されないデータをプリフェッチしてしまう可能性が高くなり，従って消費エネルギー的に損をする可能性が高くなる，というトレードオフの関係が存在する．

従って，プリフェッチを行うデータ転送単位の最適な空間粒度について検討する必要がある．まず，キャッシュ内のデータは，どのプロセスのメモリ空間に属するかによって分類できる．キャッシュの電源復帰後に実行されるプロセスに属さないデータは，参照されることはないので復帰させる必要はない．また，あるプロセスのメモリ空間は複数のページによって構成される．電源復帰後にあるプロセスを実行する際に，利用されるページとそうでないページに分類できれば，利用されないページのデータを復帰させずにすむ．さらに，ページをラインごとに見ていくと，再利用されるラインとそうでないラインに分類できる．ラインごとに再利用されるラインを識別できれば，前述の理想的なプリフェッチャとなる．このデータ分類の様子を図 2 に示す．ここでは利用されるプロセスに属するデータを赤枠で囲っており，さらにその内のデータをページごとに色分けしており，さらにその中で再利用されるデータについて明るい色で示している．

この様に，キャッシュ分割の空間粒度は単純にプロセスごと，ページごと，ラインごとに分類できる．図 2 で見たように，プリフェッチの空間粒度はプロセスごと，ページごと，ラインごとの順に粗くなる．空間粒度が粗くなればなるほど，プリフェッチすべきデータの識別は単純になるが，再利用されないデータをより多く復帰させてしまうという問題がある．すなわち，前述の式では N を小さくすると $R(f_i)/C(f_i)$ が小さくなる傾向にあり，最適解は悪くなってしまう．しかし，N が小さくなる分 $R(f_i)$ の予測や管理が容易になるため，最適解を得るのは細かい空間粒度の場合に比べて容易となる．

そのため本研究では，どの空間粒度でキャッシュを分割してプリフェッチをするのが最適であるかについて検証を行う．キャッシュ全体では扱うデータが大きいため，特にページごとからプロセスごとの分割について検討する．

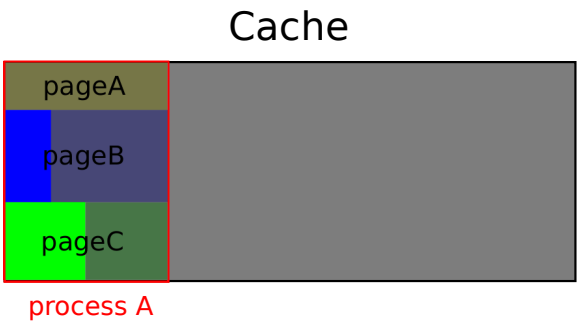

図 2 キャッシュ内のデータの分類

2.4 プリフェッチャの実装

以上を踏まえた上で，様々なデータ転送単位を用いた場合のプリフェッチの実装について検討する．

2.4.1 プロセスごとのプリフェッチ

プロセス ID を保存できる様にタグに数 bit を追加しておき，現在実行されているプロセスの ID を同様に数 bit で表し，プロセス切り替えの直後に，全てのタグを走査してプロセス ID を比較し，プロセス ID が等しい場合に，そのラインの有効ビットが立っていなければ，そのデータは今後利用されるが損失している状態であると判断し，復帰させるという方式が考えられる．この場合，実行されるプロセスに属するデータの内，再利用されるのが早いものから順に復帰させるという様な，プリフェッチの順序付けは行ってはいない．また，プロセッサがアクティブな時間が長く，キャッシュ内のデータがほとんど有効になっている状況では無駄な動作となるので，有効なラインの数を数えておき，それに応じてプリフェッチの動作をするかどうかを判断させるようにしておくべきである．

2.4.2 ページごとのプリフェッチ

あるページが利用された際に，そのページに属するラインは全て復帰させる方式で，そのページに対応するタグの値を用いて，全てのセットについてタグがヒットするかどうか走査し，ヒットした場合には，そのセットの中でヒットしたラインの有効ビットが立っていなければ，復帰させるという方式が考えられる．この場合も，キャッシュ内で有効なラインが多くなってくるとプリフェッチをしないという様にすれば，タグへの無駄なアクセスを減らすことができる．また，この方式では，利用されるのが早いデータから復帰させるという様な制御は行っていない．

2.4.3 ラインごとのプリフェッチ

今回は評価を行わなかったが，ラインごとのプリフェッチも考えられる．キャッシュの電源復帰後にデータが再利用されるかどうかをラインごとに識別する際，電源遮断前によく利用されていたデータほど再利用されやすいという，時間的局所性に従って識別する方式が考えられる．また，復帰順序についても，利用されやすいものから復帰させる方式と，ランダムに復帰させる方式が考えられる．

前者については，電源遮断前に利用されていたデータについて MRU に基づいて復帰させることで実現できる．ラインごとの MRU の情報を保持することは困難であるが，例えばキャッシュアクセスのトレースを保存しておき，電源復帰後にそれに従ってデータを復帰させれば，擬似的に MRU を実現することは可能である．この場合，利用されやすいデータほど先に復帰させることができるが，アクセストレースを保存しておく領域が必要となるためハードウェアコストがかかる．

後者については，最後にアクセスされてからどれくらい時間が経過したかという情報をラインごとにカウンタを用いて保持しておき，カウンタが閾値を越えた場合にはそのラインはもう再利用されないと判断し，データの復帰をしないという方式が考えられる．ただし，プロセッサがアイドルの状況では，カウンタの追加は行わないものとする．この様な再利用データの識別は cache decay[8] 等で行われている．cache decay とは，キャッシュ内で再利用されないと予測されるラインについて電源を遮断することで，性能を維持しつつ実行時のリーク電力を削減する手法であるが，これを本手法に利用するということも考えられる．その場合，キャッシュ全体の電源遮断直前では，再利用される可能性が高いラインのみがキャッシュに残っている状況であり，これらの有効ビットを立てたままキャッシュ全体の電源遮断を行い，電源復帰時には，全てのタグを走査して有効ビットが立っているを検出し，データを復帰させれば良い．

上記の 2 つの方式が上手くいくと考えられる状況としては，シングルプロセスのみが動いていて，さらにそのプロセスのデータの再利用性が高い場合が考えられる．また，マルチプロセスであっても，カーネルプロセスがスリープの前後で動くと考えられるため，カーネルプロセスについてデータの再利用性が高ければ，それについてはキャッシュミスを防ぐことができる．

3. 予備実験

3.1 実験内容

今回の予備実験ではプリフェッチの空間粒度はページごととして評価を行った．これは，どの空間粒度でプリフェッチを行うのが最適かを判断するには，まず，ある空間粒度では理想的にどの程度の性能が出て，さらに現実的な手法ではどの程度の性能がでるのかをまず判断する必要があり，そのための第一歩としてまず，ページごとにプリフェッチを行った場合について，理想的にはどの程度の性能が出るのかを抑える必要があるからである．アプリケーションにはネットワークやディスクアクセス等を行うものを用い，ページごとにプリフェッチを行う際のページの優先度付けの方法について，MRU(Most Recently Used)，MFU(Most Frequently Used)，Ideal(再利用されるライン数の多い物から順に復帰) を考え評価を行った．

3.2 実験環境

フルシステムシステムシミュレータ gem5[9] 上でワークロードを与え前述の評価を行った．シミュレーションに用いたパラメータを表 1 に示す．キャッシュ階層については L1，L2 の 2 階層とし，特に電源遮断によるペナルティが大きいと予想される L2 キャッシュについてのみ評価を行った．アプリケーションが I/O 待ち状態に遷移することで，長いアイドルが生じるため，ワークロードにはディスクやネットワーク等の I/O を使用するものを選定した．具体的には ping コマンドの実行と，netcat コマンドよる巨大なファイルの転送を行った．

実験を行うために，gem5 に変更を施した．具体的には，各キャッシュラインに, アイドル後の最初のアクセスかどうかを判断できるようにフラグを追加しておき, システムがアイドルモードから復帰する際に, 各キャッシュラインについてフラグを立てにいき, その後キャッシュラインの操作時にフラグが立っていると最初の操作と見なしてフラグを下ろし，そのラインはアイドル後に再利用されたとして記録した．その際，タグの値からどのページに属しているかが分かり，これらを記録しておき，MRU，MFU，Ideal のいずれの手順で復帰するのがよいかについて静的に評価を行った．また，復帰させるページの数を変化させて評価を行った．

3.3 実験結果

あるアイドルから次のアイドルまでの間に，再利用されるラインの内，プリフェッチによって復帰させることができたラインの割合を，ping の場合と，netcat の場合のそれぞれについて，図 3 と図 4 に示す．いずれの場合も，MRU による復帰よりも MFU による復帰の方が良い結果が出ている．これは下位のキャッシュでは時間的局所性の強いデータはあまりなく，長い周期でアクセスされるデータが多いことが影響していると考えられる．特に ping においては，MFU がほとんど Ideal と変わらない状況である．この

表 1 シミュレーションに用いたパラメータ

Parameters		Remarks
#Core		1
Clock Frequency		1GHz
Main Memory Latency		100ns
Page Size		8KB
L1C:	Capacity	32KB(inst.)
		64KB(data)
	Latency	1ns
	#Way	4
	Line Size	64B
LLC:	Capacity	2MB
	Latency	10ns
	#Way	8
	Line Size	64B

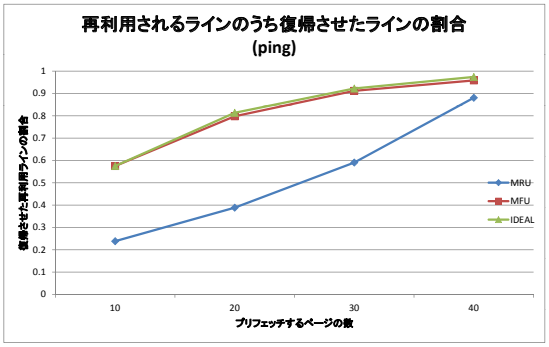

図 3 再利用されるラインの内プリフェッチによって復帰させたラインの割合 (ping)

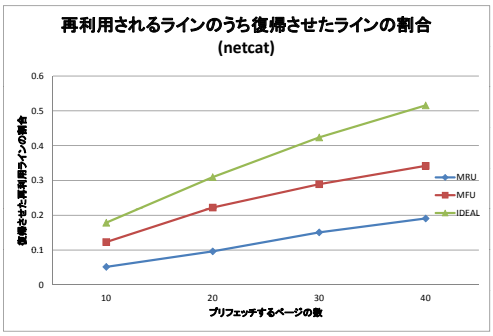

図 4 再利用されるラインの内プリフェッチによって復帰させたラインの割合 (netcat)

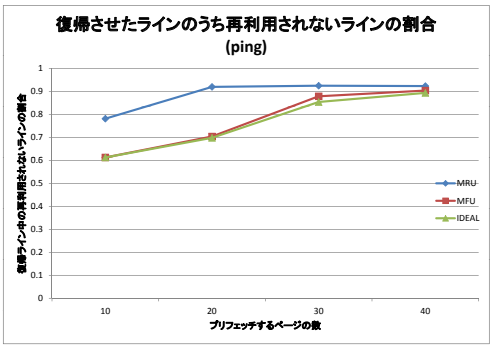

図 5 復帰させたラインのうち再利用されないラインの割合 (ping)

様に，良くアクセスされるデータについては，再利用性が高い傾向が見られる．また，図 5 と図 6 にプリフェッチによってデータを復帰させた場合に，無駄なデータの復帰がどれくらいあるのかを示した．プリフェッチするページを増やせば増やすほど，無駄なデータ復帰が増える傾向に有り，その割には，プリフェッチによって救うことのできた再利用されるデータの割合は飽和していることが分かる．

netcat では MFU の方が Ideal よりも無駄なデータの復

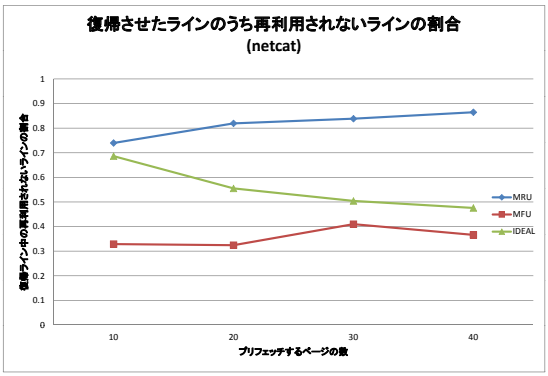

図 6 復帰させたラインのうち再利用されないラインの割合 (netcat)

帰が少なくなっている．これは，再利用されるラインが多いページについては，その他の再利用しないデータもキャッシュに多く残っていたと思われる．そのため再利用率が低いページを Ideal がプリフェッチしたものと考えられる．

これらの結果より，タグを用いたプリフェッチでは，アクセスされたページのラインをすぐに復帰させるのではなく，何度かアクセスがあり，アクセス数が閾値を越えた場合において復帰させれば良いものと考えられる．

4. 関連研究

ウェイクアップ時間の短縮は，スリープ制御による性能低下の防止，およびさらなる時間的細粒度なアイドル時間でのリーク削減を可能にする．このため様々な手法が提案されてきた．Kawasaki[10] らは，ウェイクアップ時に発生するラッシュカレントをバイパス線に誘導することで高速なウェイクアップを可能にする方式を提案した．Tong[11] らはウェイクアップ時に充放電を発生しない安定化容量を導入することで，ウェイクアップ時のラッシュカレントを低減する手法を提案した．

一方，ウェイクアップ時間だけでなく，スリープイン，ウェイクアップにともなうオーバヘッドエネルギーを考慮することで正味のリーク消費エネルギーの最小化を目的とした回路方式も提案されている．Agarwal[12] らは深さのことなる複数のスリープモードをもつ PG 回路を提案しており，アイドル期間の長さに合わせて適切なスリープモードを選択することで電力削減効率が高められる可能性を示した．

キャッシュの PG については，Stefanos Kaxiras[8] らが，cache decay と呼ばれる，コアがアクティブな状況において，長くアクセスされていないラインを PG する技術によって，性能低下を抑えつつリーク電力を削減できることを示している．

データの損失なしにキャッシュのリーク電力を削減するため，不揮発性メモリをキャッシュに用いる研究もさかんに行われている．Xiaoxia Wu[6] らは，キャッシュの各階層に不揮発性メモリを用いる方式を提案しており，それによってキャッシュの大部分のリーク電力を削減できることを示している．Clinton W. Smullen, IV[7] らは，データの保持期間を犠牲にすれば STT-MRAM の性能を向上させることができ，キャッシュに用いた場合のプロセッサの性能低下を抑えることができることを示している．

本研究で検討した荒い空間粒度のプリフェッチの関連研究としては，Hanyu Cui ら [13] の，コンテクストスイッチ後に必要なデータをプリフェッチすることで，キャッシュヒット率を向上させる研究が挙げられる．

5. まとめと今後の課題

本稿では，アイドル時にキャッシュの電源を遮断した場合に生じるキャッシュミスに起因する性能ペナルティを削減するため，キャッシュの電源を遮断する際に，タグアレイのみデータを残しておけるようにしておき，電源が復帰すると，利用される可能性の高いキャッシュブロックから順に，タグ部を参照してプロセッサの命令実行と並行してデータを復帰させるという手法を提案した．

データ復帰にはラインごと，ページごと，プロセスごとの様にいくつかの空間粒度が考えられるが，空間粒度が粗くなるほど復帰させるべきデータの識別が容易になるが，必要ないデータを復帰させてしまうというトレードオフ関係があると考えられるため，どの空間粒度でデータ復帰を行うのが適切かを判断する必要がある．そのためには，まず，ある空間粒度について，最適な復帰ポリシーでは，どの程度無駄なデータの復帰なしに，再利用されるデータを復帰できるのかについて知る必要が有り，本稿では，L2 キャッシュにおける，ページ単位のデータ復帰について検証した．その結果，MFU の方が MRU よりも，無駄なデータの復帰を少なくすることができ，かつ，再利用されるデータを多く復帰させることが出きることが分かった．今後の方針としては，ページ単位のプリフェッチだけではなく，複数ページをひとまとめにしたプリフェッチや，プロセスごとのプリフェッチについて検証していく．また，他のキャッシュ階層においても同様に検証を行っていく．それぞれのキャッシュ階層で，最適なプリフェッチの空間粒度が特定できれば，実際にプリフェッチャを実装して効果を確認する．

6. 謝辞

本研究の一部は，NEDO「ノーマリーオフコンピューティング基盤技術開発」事業による．

参考文献

[1] Nam Sung Kim, Todd M. Austin, David Blaauw, Trevor N. Mudge, Krisztián Flautner, Jie S. Hu,

Mary Jane Irwin, Mahmut T. Kandemir, and Narayanan Vijaykrishnan. Leakage current: Moore's law meets static power. *IEEE Computer*, Vol. 36, No. 12, pp. 68–75, 2003.

[2] Paul Zagacki and Vidoot Ponnala. Power improvements on 2008 desktop platforms. *intel technical journal*, 2008.

[3] Rajesh Kumar and Glenn Hinton. A Family of 45nm IA Processors. In *Proc. IEEE International Solid-State Circuits Conference* , California,USA, Feb 2009.

[4] Advanced configuration and power interface. http://www.acpi.info/.

[5] Yusuke Kanno and et.al. Hierarchical power distribution with 20 power domains in 90-nm low-power multi-cpu processor. In *Proceedings of the 2006 IEEE International Solid-State Circuits Conference*. IEEE, 2006.

[6] Xiaoxia Wu, Jian Li, Lixin Zhang, Evan Speight, Ram Rajamony, and Yuan Xie. Hybrid cache architecture with disparate memory technologies. In *Proceedings of the 36th annual international symposium on Computer architecture*, ISCA '09, pp. 34–45, New York, NY, USA, 2009. ACM.

[7] Clinton Wills Smullen IV, Vidyabhushan Mohan, Anurag Nigam, Sudhanva Gurumurthi, and Mircea R. Stan. Relaxing non-volatility for fast and energy-efficient stt-ram caches. In *HPCA*, pp. 50–61, 2011.

[8] Stefanos Kaxiras, Zhigang Hu, and Margaret Martonosi. Cache decay: exploiting generational behavior to reduce cache leakage power. In *Proceedings of the 28th annual international symposium on Computer architecture*, ISCA '01, pp. 240–251, New York, NY, USA, 2001. ACM.

[9] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, Vol. 39, pp. 1–7, August 2011.

[10] K.-i. Kawasaki, T. Shiota, K. Nakayama, and A. Inoue. A sub-us wake-up time power gating technique with bypass power line for rush current support. In *VLSI Circuits, 2008 IEEE Symposium on*, pp. 146 –147, june 2008.

[11] Tong Xu, Peng Li, and Boyuan Yan. Decoupling for power gating: sources of power noise and design strategies. In *Proceedings of the 48th Design Automation Conference*, DAC '11, pp. 1002–1007, New York, NY, USA, 2011. ACM.

[12] Kanak Agarwal, Kevin Nowka, Harmander Deogun, and Dennis Sylvester. Power gating with multiple sleep modes. In *Proceedings of the 7th International Symposium on Quality Electronic Design*, ISQED '06, pp. 633–637, Washington, DC, USA, 2006. IEEE Computer Society.

[13] Hanyu Cui and S. Sair. Extending data prefetching to cope with context switch misses. In *Computer Design, 2009. ICCD 2009. IEEE International Conference on*, pp. 260 –267, oct. 2009.