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

2004－OS－95　(5)
2004／2／26

スタックのオンチップメモリへの割り当てによるキャッシュ消費電力の削減

森若和雄 [1] 中西恒夫 [2] 福田晃 [2]

[1] 九州大学 システム情報科学府　[2] 九州大学 システム情報科学研究院

あらまし

スクラッチパッドは、主記憶と同一のアドレス空間の一部を占める、プロセッサ上のオンチップメモリである。記憶階層上はキャッシュと同一レベルにあり、その消費電力は小容量ではキャッシュより小さい。

本論文では、1 ページあたりのアクセス頻度の高いスタック領域を、OS のページ割り当て機構においてスクラッチパッドに配すことにより、消費電力の削減を図る。定期的に各プロセスのディスパッチ回数を参照し、ページを移動することで複数プロセス間でのスクラッチパッドの有効利用をはかる。

シミュレーションによって、データキャッシュへのアクセスの約 42%を スクラッチパッドへのアクセスに置き替え、データキャッシュの消費電力のうち約 23.4%を削減しうることを確認した。

Stack Area Allocation Technique for Power Reduction

Kazuo Moriwaka [1] Tsuneo Nakanishi [1] Akira Fukuda [1]

[1]Graduate School of Information Science and Electrical Engineering, Kyushu University

Abstract

The scratchpad memory is located at the same level as the cache memory in the storage hierarchy. Power consumption to access the scratchpad memory of smaller capacity is less than that to access the cache memory of the same capacity.

In this paper we reduce power consumption by allocating the stack pages, which are accessed frequently than other pages, to the scratchpad memory with memory management system of the operating system.

The OS probes dispatch frequency of processes periodically and swaps pages to use scratchpad memory effectively among multiple processes.

In our simulation, about 42% of accesses to data cache memory are redirected to the scratchpad memory and power consumption of data cache is decreased about 23.4%.

1　はじめに

携帯電話、PDA 等のモバイル情報機器において消費電力の削減は重要なテーマである。本論文はオペーレーティングシステム (OS) のメモリ管理を工夫することによる消費電力削減手法について述べる。

本論文では、スクラッチパッドを有するアーキテクチャにおいて、スクラッチパッドを意識したページ割り当てを行うことで消費電力の削減を図る。スクラッチパッドはプロセッサと同じチップ上に SRAM によって実装される小容量のメモリである。スクラッチパッドは主記憶と同じアドレス空間に位置するが、記憶階層上はキャッシュと同レベルである。

対象プログラムの静的な解析に基いてスクラッチパッドを活用する研究は既にあるが、提案手法は、対象プログラムのメモリアクセスに関する事前の知識を必要としないため、携帯電話や PDA のようにユーザーが様々なプログラムを動作させる機器に向いている。

本論文第 2 章では関連研究について述べ、第 3 章で提案手法の前提条件と詳細を解説する。第 4 章で提案手法の効果を推定するために行ったシミュレーションについて述べ、第 5 章でシミュレーションの結果とその考察を行う。

2　関連研究

スクラッチパッドを利用して消費電力を削減する研究として、Rajeshwari ら [1] はスクラッチパッドの消費電力のモデルを作り、ある領域をキャッシュ+通常のメモリに割り当てた場合とスクラッチパッドに割り当てた場合を比較するフレームワークを作った。

Stefan らはそのフレームワークのもとで、プログラムをコンパイル時にシミュレータ上で実行し、参照回数の多い命令やデータをロード時にスクラッチパッドに割り当てることで消費電力を削減した [2]。さらに、コピー命令をプログラム中に埋め込み、プログラム動作中にスクラッチパッドの内容を書き替えることで、スクラッチパッドへのアクセス回数を増やし、オフチップメモリへのアクセスを削減してより大きな消費電力削減効果を確認した [3]。

Mahesh ら [4] は、スクラッチパッド上にスタック専用領域を確保し、スタック領域のうち LIFO アクセスを行う部分をその領域に配置してキャッシュを回避することで消費電力を削減した。またキャッシュミスによるストールの減少により実行速度が上がることを示した。

ただしこれらの研究では実行されるプログラムはあらかじめ静的に解析された単一のプログラムであり、複数プロセスの並列動作、スケジューリング、および入出力による振舞いの変化は考慮していない。本研究では複数プロセスが並列に動作する環境でのスクラッチパッドを利用した消費電力削減手法を提案する。

提案手法では、OS のメモリ管理で物理アドレスによる消費電力の差を考慮してページ割り当てを工夫することで消費電力の削減を狙う。

OS のページ割り当て部分で工夫することによって消費電力を削減する研究として、Alvin らは大量に DRAM が存在する場合に、ページ割り当てに first-fit アルゴリズムを適用して利用されない DRAM モジュールが発生する確率を高くすることで、DRAM の消費電力が少ない動作モードを利用して消費電力の削減を行った [5]。

3　提案手法

3.1　対象アーキテクチャ

本手法が対象とするアーキテクチャのブロック図を図 1 に示す。

MPU が参照する仮想アドレスは MMU で物理アドレスに変換される。物理アドレスに従って、スクラッチパッドまたはキャッシュを参照する。キャッシュは命令キャッシュとデータキャッシュを別々に持つ。キャッシュのミスヒットが発生した場合はオフチップメモリを参照する。

スクラッチパッドおよびキャッシュは SRAM で実装される。オフチップメモリは SDRAM で実装されている。このアーキテクチャは ARMv6 アーキテクチャ [6] を参考にした。

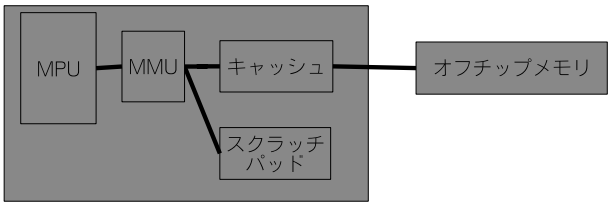

図 1: 対象とするアーキテクチャのブロック図

表 1: スクラッチパッドとキャッシュの 1 アクセスあたりの電力量例 (文献 [2]) より引用

容量 (bytes)	キャッシュ	スクラッチパッド
1024	3.75 nJ	0.82 nJ
2048	4.04 nJ	1.07 nJ
4096	4.71 nJ	1.21 nJ
8192	5.39 nJ	2.07 nJ
16384	(5.99 nJ)	(2.42 nJ)
32768	(6.76 nJ)	(3.02 nJ)

3.2 スクラッチパッドとキャッシュの消費電力

表 1 に文献 [2] から部分的に引用したデータ容量ごとのキャッシュとスクラッチパッドの 1 アクセスあたりの電力量例を示す。ただし 16,384bytes 以降の欄は筆者が二次近似により推定した値である。消費電力量はプロセスルール、電圧、実装方式などによって変わる。

多くのプロセッサではデータキャッシュのサイズは 8kbytes から 32kbytes である。32kbytes 程度のスクラッチパッドでは、キャッシュへのアクセスに必要なエネルギーよりも、スクラッチパッドへのアクセスに必要な電力量のほうが少ない。

この消費電力量の差を利用して、アクセスの多い部分をオフチップメモリからスクラッチパッドへ移動することでメモリアクセス全体の消費電力を減らすことができる。

3.3 メモリ領域ごとの局所性の比較

表 2 は、筆者らが SimpleScalar[7] を用いて調べた、spec CPU INT 2000[8] と SNU Real-Time Benchmark Kernel [9] の 2 つのベンチマークプログラム群における、セグメント別のページ数と各セグメントに属するページへのアクセスが全体に占める割合を示す。adpcm-test プログラムの例では、静的に確保されるセグメント (text, data, bss) が全 26 ページで 77.9%のアクセスを占めるのに対して、スタックは 2 ページで 22.1%を占めている。他のプログラムにおいてもスタック領域のページの 1 ページあたりアクセス率がこの 3 つの分類の中では一番大きい。

表 2: セグメント別のページ数と総メモリアクセスに占める割合

プログラム	静的	スタック	ヒープ
bzip2	39(75.1%)	4(21.4%)	49153(3.5%)
cc1	414(83.8%)	5(16.1%)	2(0.2%)
gzip	44(75.1%)	4(21.4%)	49153(3.5%)
mcf	37(83.9%)	2(14.4%)	686(1.7%)
parser	39(72.5%)	2(20.7%)	99(6.9%)
twolf	64(85.3%)	3(13.9%)	5(0.7%)
vortex	81(89.3%)	3(10.1%)	6(0.6%)
adpcm-test	26(77.9%)	2(22.1%)	0(0.0%)
bs	19(83.1%)	2(16.9%)	0(0.0%)
crc	18(78.3%)	2(21.7%)	0(0.0%)
fft1	18(78.2%)	2(21.8%)	0(0.0%)
fft1k	22(72.7%)	2(27.3%)	0(0.0%)
fibcall	20(81.2%)	2(18.8%)	0(0.0%)
fir	19(72.2%)	2(27.8%)	0(0.0%)
insertsort	20(80.4%)	2(19.6%)	0(0.0%)
jfdctint	18(84.5%)	2(15.5%)	0(0.0%)
lms	19(74.6%)	2(25.4%)	0(0.0%)
ludcmp	21(81.1%)	2(18.9%)	0(0.0%)
matmul	20(81.3%)	2(18.7%)	0(0.0%)
minver	19(80.9%)	2(19.1%)	0(0.0%)
qsort-exam	18(82.2%)	2(17.8%)	0(0.0%)
qurt	20(81.0%)	2(19.0%)	0(0.0%)
select	18(82.9%)	2(17.1%)	0(0.0%)

3.4 スタック領域のページ割り当て

汎用 OS ではどのようなプログラムが実行されるかは一定でなく、メモリアクセスのパターンは入力データにも依存する。アクセスの多いオブジェクトをコンパイル時解析で予測して、スクラッチパッドに最適配置することは困難である。

そこでセグメントを考慮することでアクセス頻

度の高いページを見つける手法が考えられる。OS側であらかじめプロセスのメモリマップがわかっていれば、プログラムのロード時やページ割り当て要求時に、ページの論理アドレスを調べることで要求されたページがどのセグメントに属するものか判定できる。

プロセス中のある 1 ページをスクラッチパッドに移動することによって得られる省電力効果は、そのページへのアクセス回数に大きく依存するので、提案手法では 1 ページあたりのアクセス回数が一番多いスタック領域をスクラッチパッドへ割り当てる領域とすることで消費電力の削減を行う。

物理アドレス空間において、スクラッチパッドは小さなアドレスに、オフチップメモリは大きなアドレスに割り当てられていると仮定する。

スタック領域に属するページは物理アドレスの小さいページから割り当て、その他のページは物理アドレスの大きいページから割り当てることで、スタック領域に属するページをスクラッチパッドへ割り当てる。

3.5 ページ入れ換え

今回の想定ではスクラッチパッドの容量は 32kbytes で、ページ数は 8 ページとしている。複数のプロセスが並列に動作する環境では全てのスタックをスクラッチパッドに配置することは難しい。その場合、実行時間のほとんどをスクラッチパッドにページを持たないプロセスが占め、消費電力削減の効果が非常に低くなる可能性がある。ユーザの入力などで実行されるプロセス群が入れ変わる場合にも消費電力削減を見込むためには、ページ入れ替えを行う必要がある。

このようにページの物理アドレスによって振舞いをかえるためには、汎用 OS の古典的な実装ではアドレス解決のために大量のメモリアクセスが必要になる。OS によるページの扱いについて実メモリアドレスを考慮した拡張には、NUMA 型マルチプロセッサへの対応や、メモリホットプラグへの対応があり、これらの技術が適用可能であると考えられる。物理ページアドレスによるページの参照については本論文では特に論じない。

ページ入れ替え間隔 t の最低値は、以下のように計算する。入れ替えたページを含むプロセスが必ず実行されると仮定して、

- 1 ページ入れ替えに必要な電力量: E_1
- スクラッチパッドへ移動するページの単位時間あたりアクセス回数: N
- スクラッチパッドとキャッシュメモリの 1 アクセスあたりの消費電力量の差: D

とするとき、ページ入れ換えに必要な電力量は、ページ入れ換えの間に削減が期待できる電力量より十分小さい必要があるので、$t * N * D >= E_1$ となり、t について解くと $t >= E_1/(N * D)$ となる。

スタックの 1 ページのアクセス頻度を表 2 より 10%、動作周波数はシミュレータで利用する 40MHz とする。

1 アクセスあたりの消費電力量の差は表 1 の 32kbytes キャッシュと 32kbytes スクラッチパッドの消費電力量の差である 3.74nJ とする。

メモリと CPU、およびバスの合計消費電力を 900mW とする。これは文献 [10] の従来型 LSI システムの消費電力 891mW を参考にした。

ページの入れ替えは 4,096bytes のページを 2 回コピーするので、8,192bytes のデータを 32bit ごとに読み込み、転送、書き込みのループを 2,048 回繰り返して実行すると仮定する。1 イテレーションに 12 サイクルかかるとして、ページ入れ換えに必要な時間は約 614 μ秒となる。このとき必要な電力量は、実行時間にシステムの消費電力を掛けた約 552 μ J である。

上のパラメータを式に代入すると $t >= 37.0(\mathrm{ms})$ となるので、ページ入れ換えの平均間隔は少なくとも 37ms 以上でなければ、ページ入れ換えの電力量が入れ換えによって削減される電力量を上まわる。今回のシミュレーションでは 100ms ごとにページ入れ換えの判断を行い、必要と判定された時にだけページ入れ換えを実行する。

3.6 ページ入れ換え対象プロセスの決定

入れ替え対象となるページは、この先に頻繁なアクセスが予想されるプロセスのスタックである。入れ換え対象ページを決めるために利用する情報としては以下のものが考えられる。

- プロセスが run キュー、wait キューのどちらに存在するかという情報
- プロセスの優先度
- プロセスの実行時間
- プロセスのディスパッチ回数
- プロセスから与えられるヒント

今回のシミュレーションでは 10ms ごとに再ディスパッチを行うプロセススケジューラを用いて、プロセスのディスパッチ回数による選択を採用する。100ms ごとに以下の動作を行い、ページ入れ替えの判断をする。

1. 全てのプロセスをディスパッチ回数でソートする。
2. 1 ページもスクラッチパッドにページを持っていないプロセスの内一番ディスパッチ回数が多いプロセス P に注目する。
3. P よりディスパッチ回数が少なく、スクラッチパッドにページを持っているプロセスの内、一番実行回数が少ないものを、ページ入れ替えの対象とする。
4. ページ入れ換え対象のプロセスが存在した場合、ページ入れ替えを行う。
5. 全てのプロセスのディスパッチ回数をリセットする。

3.7 ページ入れ替え対象プロセス決定手法の考察

今回はディスパッチ回数をもとにページ入れ換え対象となるプロセスの決定を行う。しかし携帯電話など、プロセスの一部や優先度の扱いについて事前にプロセスの振舞いを知ることができるシステム、およびプロセス実行時間の比較を簡単に行えるシステムでは他の手法を利用することが考えられる。

run キューおよび wait キューを見て、一般に run キューに存在するプロセスをこれから実行されるプロセスと予測することはできないが、以下のような条件を満たす場合にはプロセスが存在するキューを元にページ入れ換え対象プロセスを決定できる。

- プロセス A が実行キューに入っている場合はプロセス A,B,C が利用されることが多いなど、事前にプロセス間の状態について依存関係がわかっていればプロセス A がどのキューに存在するかを元に B,C のページを優先的にスクラッチパッドに配置する。
- イベントの生起頻度がイベントの種類によって大きく異なり、それが事前に知られている場合、待っているイベントの種類によって wait キューを分け、発生頻度が高いイベントを待つプロセスをページ入れ換え対象にする。

プロセスの優先度に関しては実行時間との関連を一般的に言うことはできないが、以下のような場合には利用できる。

- 優先度が高いプロセスはリアルタイム性が求められ実行時間が短く、優先度が低いプロセスは実行時間が長いという関連がある場合、優先度が低いプロセスのページをスクラッチパッドに優先的に配置する。
- テレビ電話システムの CODEC のように優先度は高いが大量の処理を必要とするタスクがある場合、そのタスクだけスクラッチパッドに常にページを割り当てる。

プロセスの実行時間はアクセス回数との関連が大きい。ユーザーによる状態遷移などに備えて 100ms から数秒程度のウインドウを設けて実行時間を積算することで、現在よく実行されているプロセスを判定することができる。

プロセスのディスパッチ回数による選択は単純な実装が期待できるが、ディスパッチ毎の実行時間が短くかつ優先度の高いプロセスが優先され、消費電力削減に貢献しない場合がある。

プロセスから与えられるヒントは、最も適切な予想を提供できる点と、実行時の予想にかかる計算量やメモリの削減という点で有望である。これは事前に実行されるプログラムがわかっている組み込みシステムおよび、開発環境でヒント生成を行うことで利用できる。

4　シミュレーション

提案手法による消費電力削減の効果を推測するためにシミュレーションを行う。

4.1　シミュレーション環境

シミュレーション環境の概要を図 2 に示す。本研究では OS の挙動を変更する必要があるため、MMU、キャッシュと OS のページ管理機構の振舞いを模擬するシミュレータを作成する。

シミュレータへの入力としてシナリオファイルと各プログラムのトレース情報を与える。シナリオファイルにはプロセスのトレース名 (プログラム名)、生成時刻、優先度のリストが入っている。トレース情報は各プログラムの命令インスタンスおよびその命令インスタンスが参照するデータの論理アドレスの組のリストである。

シミュレータの出力はキャッシュ、スクラッチパッド、オフチップメモリそれぞれへのアクセス回数と消費電力量を積算した結果である。

プロセススケジューラはプロセスの終了、新しいプロセスの生成、タイマなどのイベントによって起動され、固定優先度でプロセスをスケジューリングする。同一優先度のプロセスはラウンドロビン方式でディスパッチする。

MMU とページテーブルを模擬したアドレス変換機構によって物理アドレスが生成される。物理アドレスに従ってキャッシュまたはスクラッチパッドへのアクセスが発生し、キャッシュのミスヒット時にはオフチップメモリへのアクセスを発生させる。

このシミュレータでは全ての命令が 1 クロックで実行されるものとしているが、発生したメモリ参照や TLB ミスヒットに応じて遅延が挿入される。キャッシュ、スクラッチパッド、オフチップメモリ、オフチップメモリを接続するバスについてアクセス回数をカウントし、消費電力量を積算する。

スタックおよび BSS 領域のページ割り当ては実際に利用された時に実行する。これは Linux[11] のページ割り当て方式に準拠する。

シミュレーションに利用したパラメータは表 3 の通りである。

キャッシュの構成、キャッシュブロック吐き出し

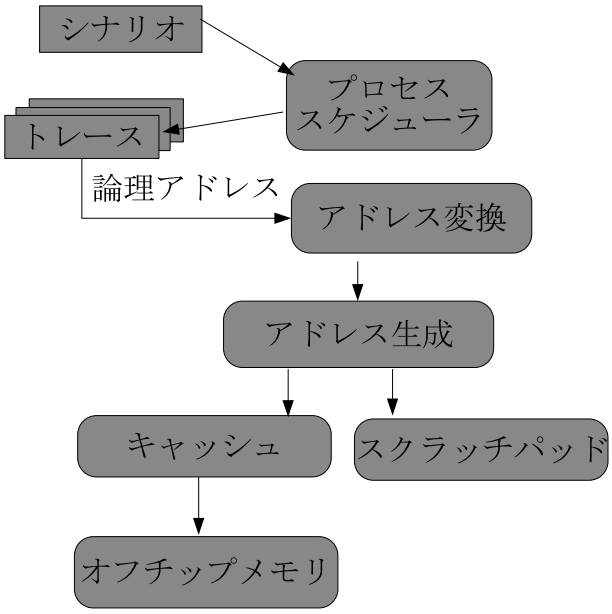

図 2: シミュレータの概要

手法とページサイズに関しては ARM9TDMI を参考にした。スクラッチパッドとキャッシュの消費電力量は文献 [2] からの二次近似による。キャッシュとスクラッチパッドのサイズは一律 32kbytes とする。シミュレータの実行にかかる時間とトレースのサイズの問題から、シミュレーション時間は 1 秒、CPU の動作周波数は 40MHz とする。メモリマップは 0 番地からスクラッチパッド、0x100000 番地からオフチップメモリとする。

4.2　アプリケーション

実際に動作するアプリケーションプログラムのメモリアクセスによってスクラッチパッドへのアクセスの頻度やキャッシュのヒット率などは変化する。

このシミュレーションでは表 2 で示したベンチマークプログラム群を、ARM 版 SimpleScalar に命令およびデータの論理アドレスを出力する変更を行い生成した実行トレースと、PDA 等を想定したアプリケーションを実機で動作させた関数実行タイミングから生成した擬似トレースを対象にする。

spec CPU INT 2000 ベンチマークに含まれるプログラムではヒープ領域に確保したメモリを大量

表 3: シミュレーションに利用したパラメータ

スクラッチパッドのサイズ	32kbytes
1 アクセスの消費電力量	3.02nJ
オフチップメモリのサイズ	256Mbytes
キャッシュ構成	4way set assoc.
命令キャッシュ容量	32kbytes
データキャッシュ容量	32kbytes
1 アクセスの消費電力量	6.76nJ
キャッシュ吐き出し手法	ラウンドロビン
ページサイズ	4,096bytes
CPU、メモリ動作周波数	40MHz
シミュレーション時間	1 秒

に利用する傾向がある。特に gzip ではスタック 4 ページに対してヒープ 49,153 ページとヒープ領域が非常に大きくなる。実行時間が長いものが多いが、1 秒以上かかるものはプログラム開始から 1 秒程度実行した途中までのトレースを対象にする。

SNU Real-Time Benchmark Kernel に含まれるベンチマークプログラムではヒープ領域を全く利用していない。こちらに含まれるプログラムは実行時間が短いものが多い。

シナリオファイルは対象プログラムからランダムにプロセス・優先度・実行開始時刻を選び、1 秒間に 50 プロセスの確率でプロセスが実行開始されるシナリオファイルを生成するプログラムで生成する。

5 結果

ランダムに生成した 3 つのシナリオを実行して消費電力量の削減について調べた。合計 3 秒間に、スクラッチパッドへのアクセスが 1,095,434 回、データキャッシュへのアクセスが 1,498,893 回、命令キャッシュへのアクセスが 8,470,118 回行われた。各シナリオのアクセス回数は表 4 にまとめた。データアクセスのうち約 42%(各シナリオでは 66.6%、35.1%、54.7%) をスクラッチパッドが占めた。アクセス回数の割合を示したものが図 3 である。

キャッシュのミスヒットの増加によるオフチップメモリへのアクセスを無視しても、データキャッシュだけの場合に比べてスクラッチパッドを併用した場合、アクセスによる消費電力量は約 23.4%減少した。

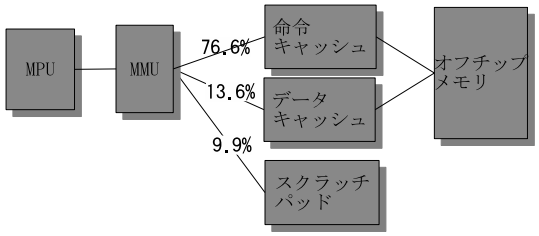

図 3: キャッシュとスクラッチパッドのアクセス回数の割合

表 4: 各シナリオのキャッシュとスクラッチパッドへのアクセス回数

シナリオ	1	2	3
スクラッチパッド	7221305	3329081	5421157
データキャッシュ	3565638	6154436	4499400
命令キャッシュ	37518720	24733738	29901875

100ms ごとのアクセス回数の平均をグラフにしたものが図 4 である。最初の 100ms での命令キャッシュのアクセスが少ない理由は、ランダムにプロセスの実行開始時刻を決めているためにシミュレーション開始時には何も実行されていないためである。

スクラッチパッドのアクセス率が時間が経つにつれて落ちる原因としては、優先順位の低いプロセスが先に実行開始し、優先順位が高いプロセスが後で実行開始された場合に、ページ入れ替えが行われるまでスクラッチパッドのアクセスがほとんど行われないことが考えられる。

6 おわりに

OS によってプロセスのスタック領域をスクラッチパッドに配置することによって消費電力の削減を図った。ページ割り当て手法とページ入れ替え手法について提案した。効果の推定のためにシミュレータを作成し、複数のベンチマークソフトウェアを組み合わせたランダムなシナリオで効果を測

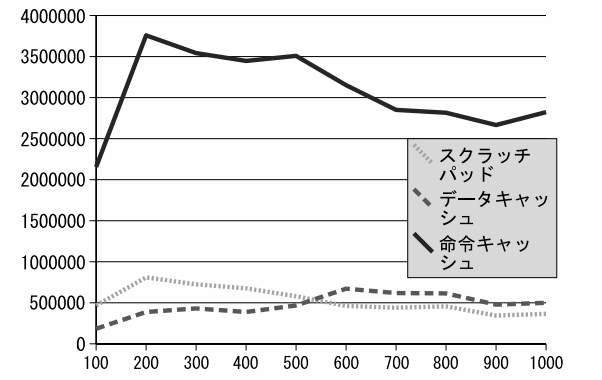

図 4: キャッシュとスクラッチパッドのアクセス回数の変化

定した。その結果、データキャッシュへのアクセス中約 42%をスクラッチパッドに置き替え、アクセスに必要な消費電力量を約 23%削減することを確認した。

今後の課題としては、プログラムにアクセス頻度に関するヒント情報を付加して消費電力の削減を大きくする手法が考えられる。具体的には、プログラムのインストール時や出荷時にアドレスごとのアクセス頻度に関するヒントを生成し、それを利用することによりスタックに限らずアクセス頻度の高い関数やデータをスクラッチパッドに配置する候補とする。

また、今回シミュレータを作成するにあたりプロセスのモデルを命令およびデータのアドレス列という形に単純化したため、スタックポインタの情報を利用していない。ページ入れ換え時のスクラッチパッドへの移動対象ページを選択する際に、プロセスのスタックポインタを利用することでより適切な判断が期待できる。

謝辞

本研究は，松下電器 CE アーキテクチャ開発センターとの共同研究として行われたものである。特に南方郁雄氏、水山正重氏、山本哲士氏、堀江佳恵氏には数多くの助言や資料およびトレースデータを頂いた。

参考文献

[1] Rajeshwari Banakar, Stefan Steinke, Bo-Sik Lee, M. Balakrishnan, and Peter Marwedel (Indian Inst. Tech.,Univ. Dortmund): Scratchpad memory: A design alternative for cache on-chip memory in embedded systems In *Proc. of the 10th International Workshop on Hardware/Software Codesign, CODES, Estes Park (Colorado)*, pp. 73–78 ,(2002).

[2] Stefan Steinke, Lars Wehmeyer Bo-Sik Lee, Peter Marwedel(Univ. Dortmund): Assigning Program and Data Objects to Scratchpad for Energy Reduction, *DATE 2002*, pp. 409–415 (2002).

[3] Stefan Steinke, Nils Grunwald, Lars Wehmeyer, Rajeshwari Banakar, M. Balakrishnan and Peter Marwedel(Indian Inst. Tech.,Univ. Dortmund): Reducing Energy Consumption by Dynamic Copying of Instructions onto On-chip Memory, *ISSS 2002*, pp.213–218 (2002).

[4] Mahesh Mamidipaka and Nikil Dutt: On-chip stack based memory organization for low power embedded architectures. In *Proc. of the DATE 2003*, pp. 11082 – 11089, (2003).

[5] Alvin R. Lebeck, Xiaobo Fan, Heng Zeng and Carla Schlatter Ellis(Duke Univ.): Power Aware Page Allocation, *ASPLOS-IX*, pp. 105–116 (2000).

[6] David Brash(ARM Ltd.): The ARM Architecture Version 6 (ARMv6), `http://www.arm.com/pdfs/V6_whitepaper_A01.pdf`

[7] SimpleScalar (SimpleScalar LLC): `http://www.simplescalar.com/`

[8] spec CPU INT2000 (Standard Performance Evaluation Corp.): `http://www.specbench.org/osg/cpu2000/`

[9] SNU Real-Time Benchmark Kernel (Seoul National Univ.): `http://archi.snu.ac.kr/realtime/benchmark/index.html`

[10] T.Nishikawa *et al.*(Toshiba Corp.): A 60MHz 240mW MPEG-4 video-phone LSI with 16Mbit embedded DRAM, *ISSCC 2000*, pp. 230–231 (2000).

[11] Linux: `http://kernel.org/`