九州大学学術情報リポジトリ
Kyushu University Institutional Repository

Non-uniform Selective Way Cache の動的制御による組込みプロセッサの省エネルギー化

石飛, 百合子
九州大学大学院システム情報科学府

石原, 亨
九州大学システムLSI研究センター

安浦, 寛人
九州大学

https://hdl.handle.net/2324/13843

出版情報：電子情報通信学会技術研究報告. 108 (464), pp.13-18, 2009-03-05. 電子情報通信学会
バージョン：
権利関係：

KYUSHU UNIVERSITY

社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Non-uniform Selective Way Cacheの動的制御による組込みプロセッサの省エネルギー化

石飛　百合子†　　石原　亨††　　安浦　寛人†††

† 九州大学大学院システム情報科学府　〒819–0395 福岡県福岡市西区元岡 744 番地
†† 九州大学システム LSI 研究センター　〒814–0001 福岡県福岡市早良区百道浜 3–8–33
††† 九州大学　〒812–8581　福岡市東区箱崎 6–10–1
E-mail: †ishitobi@c.csce.kyushu-u.ac.jp, ††ishihara@slrc.kyushu-u.ac.jp, †††yasuura@c.csce.kyushu-u.ac.jp

あらまし　本稿は Non-uniform Selective Way Cache(NSWC) の動的ウェイ切り替えによる組込みプロセッサの省エネルギー化手法の提案を行う．NSWC は，消費エネルギーの観点で異なる性質の 2 種類のウェイを保持する．提案手法は命令キャッシュとして NSWC を用い，アプリケーションプログラムへのウェイ切り替え命令を挿入することで，動的なウェイ切り替えを行う．動的ウェイ切り替えを行うことで，キャッシュメモリのアクセスエネルギーの削減と高いキャッシュヒット率を両立し，組込みプロセッサの省エネルギー化を実現する．提案手法を用いることで，セット・アソシアティブ方式のキャッシュメモリを利用した場合と比較して 7%〜20%の消費エネルギー削減効果を確認した．
キーワード　組込みプロセッサ，省エネルギー化，キャッシュメモリ

A Dynamic Management Technique of a Non-uniform Selective Way Cache for Reducing the Energy Consumption of Embedded Processors

Yuriko ISHITOBI†, Tohru ISHIHARA††, and Hiroto YASUURA†††

† Graduate School of Inf. Sci. & E.E.,Kyushu University
Motooka 744, Nishi-ku, Fukuoka-shi, 819–0395 Japan
†† System LSI Research Center,Kyushu University
Momochihama 3–8–33, Sawara-ku, Fukuoka-shi, 814–0001 Japan
††† Kyushu University
Hakozaki 6–10–1, Higashi-ku, Fukuoka-shi, 812–8581 Japan
E-mail: †ishitobi@c.csce.kyushu-u.ac.jp, ††ishihara@slrc.kyushu-u.ac.jp, †††yasuura@c.csce.kyushu-u.ac.jp

Abstract　This paper proposes a dynamic management technique of Non-uniform Selective Way Cache(NSWC) for reducing the total energy consumption of a CPU core, cache memories, and off-chip memories. NSWC has a way uses low supply(Vdd) and low threshould(Vth). In our approach, we decide insert points of instructions to change available ways in the Non-uniform Selective Way Cache. Experiments using parameters of a commercial embedded processor and an off-chip SDRAM demonstrate that our algorithm reduces the energy consumption of the processor system by 7%-20% compared to the result of a processor with a same size set associative cache memory.
Key words　embedded processor, energy reduction, cache memory

1. はじめに

携帯機器の長時間駆動の必要性から，組込みプロセッサの省エネルギー化が重要となっている．組込みプロセッサの消費エネルギーは，搭載するキャッシュメモリの構成に大きく影響を受ける．この要因として第一に，キャッシュメモリにおける消費エネルギーが総消費エネルギーの中で大きな割合を占めていることが挙げられる．ARM920T^{TM} マイクロプロセッサでは消費電力の 44%，低消費電力プロセッサである StrongARM においても 27%の消費電力がキャッシュメモリにより占められている [1]〜[3]．また，第二の要因は，キャッシュメモリのヒット率が下層のメモリへのアクセスエネルギーに影響を与えることが挙げられる．ヒット率低下は，下層のメモリへのアクセス

回数を増加させ消費エネルギーの増加につながる．組込みプロセッサの省エネルギー化を実現するには，キャッシュメモリの消費エネルギー削減およびヒット率向上を両立することが必要となる．

ヒット率の向上を目的として，セット・アソシアティブ方式を用いたキャッシュメモリが一般的に利用されている．セット・アソシアティブ方式は，キャッシュアクセス時にすべてのウェイからのデータ読出しを行う．複数ウェイからのデータ読み出しがアクセスエネルギー増大の要因の一つとなっている．このため，アクセスに用いるウェイを制限することでアクセスエネルギーを削減する手法が提案されている [7]．文献 [7] で提案するキャッシュメモリはメモリアドレス空間内に，ウェイを指定してアクセスを行う領域を持つ．この領域に対しアクセス回数が多いコードを配置することで，アクセスエネルギーの削減を行っている．しかし，利用するウェイがアクセスアドレスから一意に決定するため，セット・アソシアティブ方式と比較して競合性ミスが増加する可能性がある．

本稿は組込みプロセッサの省エネルギー化を目的とし，キャッシュメモリのアーキテクチャとして Selective Way Cache(SWC) [6] を利用し，動的ウェイ切り替え手法の提案を行う．提案する動的ウェイ切り替えは，キャッシュアクセス時に常に単一のウェイのみ利用する．キャッシュメモリにおける競合性ミスを削減するために，アクセスに用いるウェイをウェイ切り替え命令の実行により動的に切り替える．加えて本稿では，SWC のアーキテクチャとして Non-uniform Selective Way Cache(NSWC) の利用を検討する．NSWC は，エネルギーの観点で異なる 2 種類のウェイを用いて構成した SWC である．NSWC にはアクセスエネルギーが大きくリークエネルギーが小さい SP ウェイと，アクセスエネルギーが小さくリークエネルギーが大きい DP ウェイが存在する．提案手法では，動的ウェイ切り替えによりアクセスの大部分を DP ウェイを用いて実行することで，アクセスエネルギーの削減を実現する．

本稿の構成は，2 章において関連研究を提示し提案手法の概要を示す．3 章において動的ウェイ切り替えおよび NSWC の利用による省エネルギー化手法の提案を行う．4 章は提案手法の評価実験および評価結果を示す．5 章は本稿のまとめである．

2. 関連研究および提案手法概要

2.1 関 連 研 究

セット・アソシアティブ方式のキャッシュメモリでは，アクセス時の複数のタグアドレスおよびデータの読出しがアクセスエネルギー増大の要因となっている．このため，アクセスに用いるウェイ数を制限し，アクセスエネルギーを削減するキャッシュメモリが提案されている．

実行するプログラムに応じて，必要なウェイ数を選択可能なキャッシュメモリとして Selective Way Cache(SWC) が David H. Albonesi により提案されている [6]．David H. Albonesi により提案されている SWC の構成を図 1 に示す．SWC は，タグアレイおよびデータアレイをウェイごとに異なるメモリアレイで構成する．アクセスに利用するウェイは Cache Way Select Register(CWSR) に保持する値によって指定する．CWSR の値に応じたウェイ選択信号が各ウェイに入力され，選択されたウェイのみタグ比較およびデータの読み書きが行われる．文献 [6] では，アプリケーションプログラムの事前実行により，実行するアプリケーションプログラムごとに必要なウェイ数を決定している．このため，必要なウェイ数が多い場合は消費エネルギー削減効果は小さい．

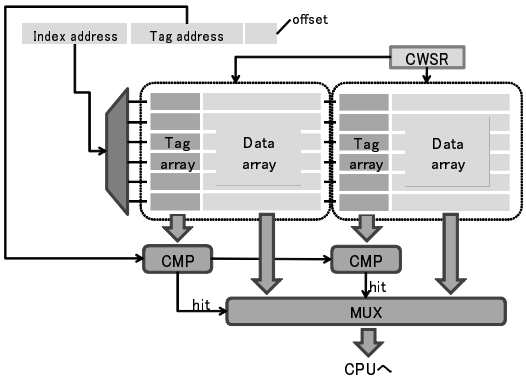

図 1　SWC の構成

アクセスに用いるウェイを限定し，キャッシュメモリの省エネルギー化を実現する手法として，Way Placement を用いる手法が Timothy M. Jones らにより提案されている [7]．Timothy M. Jones らの手法は，すべてのウェイを用いるアクセス (Normal access) を行う領域である Normal area と，特定の 1 つのウェイのみを利用したアクセス (Way placement access) を行う領域である Way placement area の 2 つの領域を利用できるキャッシュメモリを用いる．Way placement access は，タグアドレスの下位ビットをウェイ選択信号として使用し，指定したウェイのみに対しアクセスを行う．Way placement area 内のあるコードには，常に 1 つのウェイを利用しアクセスが行われるため，アクセスエネルギーが小さい．Way placement area に対し，アクセス回数の多いコードを配置することで，キャッシュメモリの省エネルギー化を実現している．

2.2 提案手法概要

Way Placement の利用と SWC の利用に違いは，単一のウェイをアクセスする際，SWC は任意のウェイを選択することが可能であるが，Way Placement はコードが配置されたアドレスにより，アクセスに利用するウェイが一意に決定することが挙げられる．アクセスに用いるウェイが限定されるため，Way Placement の利用により，セット・アソシアティブ方式を用いた場合と比較してキャッシュヒット率が低下する可能性がある．文献 [7] では，この理由から Way placement access と Normal access の併用を行っていると考えられる．Normal access は，通常のセット・アソシアティブ方式と同様に，アクセスにすべてのウェイを用いるため，アクセスあたりの消費エネルギーが大きい．

本稿では，キャッシュメモリのアクセスエネルギー削減および高いキャッシュヒット率の両立をするために，SWC の動的ウェイ切り替えを提案する．アクセスに用いるウェイが一意に

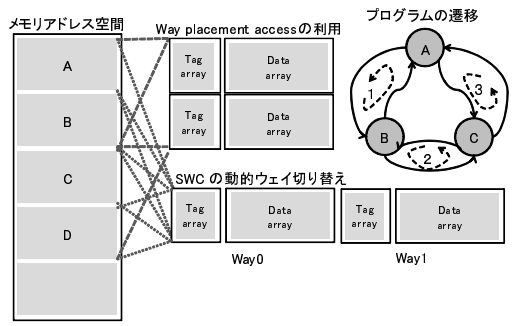

図 2　SWC の動的ウェイ切り替え活用例

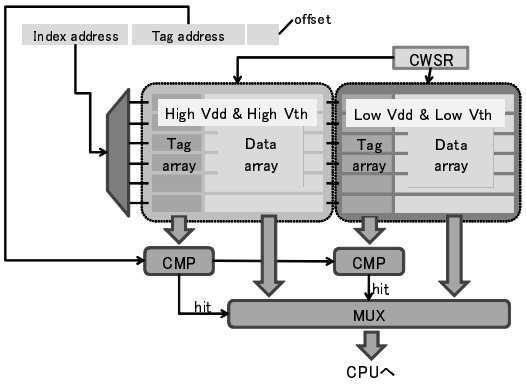

図 3　NSWC の構成

決定する Way Placement と SWC の動的ウェイ切り替えの違いを図 2 の例を用いて説明する．メモリアドレス空間内に配置される A〜D はそれぞれ関数とする．図 2 中の右上に示すプログラムの遷移において，A と B が繰り返されるループをループ 1，B と C が繰り返されるループをループ 2，A と C が繰り返されるループをループ 3 とする．ループ 1 およびループ 2 を実行する場合は，Way Placement の利用時に競合性ミスは発生しない．しかし，ループ 3 を実行する場合，A と C がキャッシュメモリにおいて競合するため，競合性ミスが多発する．C を Normal area に配置することで，競合性ミスの削減が可能であるが，Way placement access を用いた場合と比較してキャッシュメモリのアクセスエネルギーが増大する．一方で，SWC の動的ウェイ切り替えを利用する場合は各関数間の遷移時にウェイ切り替えを行う．動的ウェイ切り替えを行うことで，ウェイ切り替えのオーバーヘッドが発生するが，キャッシュメモリにおける競合性ミスを削減することが可能となる．加えて，すべてのアクセスは単一のウェイを用いるため，キャッシュメモリにおけるアクセスエネルギーの削減を可能とする．

提案手法では，SWC の一部のウェイを低い電源電圧および閾値電圧で動作する SRAM を用いて構成した NSWC(Non-uniform Selective Way Cache) の利用を検討している．動的ウェイ切り替えにより，アクセスの大部分を低い電源電圧を利用するウェイに集中させることで，更なるアクセスエネルギーの削減を実現する．

3. NSWC の利用と動的ウェイ切り替え

3.1 NSWC

3.1.1 NSWC の構成

提案する NSWC の構成を図 3 に示す．NSWC は SWC と同様に各ウェイが異なるメモリのアレイを用いて構成されている．オンチップメモリはアクセス速度の向上とリーク電力の低減のために，高い電源電圧および高い閾値電圧で動作するメモリを用いて構成することが一般的である．しかし，メモリへのアクセスエネルギーは電源電圧の二乗に比例するため，高い電源電圧を利用することでアクセスエネルギーが増大する．NSWC は高い電源電圧および高い閾値電圧で動作するメモリのアレイで構成したウェイ (以下 SP ウェイ) と低い電源電圧および低い閾値電圧で動作するメモリのアレイを用いて構成したウェイ (以下 DP ウェイ) を持つ．DP ウェイは低い電源電圧を用いることで，SP ウェイよりもアクセスエネルギーを小さくすることができる．DP ウェイでは，アクセス速度の低下を防ぐため電源電圧の低下に合わせ，閾値電圧を低下させたメモリを用いる必要がある．このため，DP ウェイは SP ウェイと比較してリークエネルギーが増大する．リークエネルギーの増大と比較して，DP ウェイの利用によるキャッシュアクセスエネルギーの削減が大きい場合，NSWC を用いたキャッシュメモリの省エネルギー化が可能となる．

アクセスに用いるウェイは，CWSR に保持する値によって指定される．CWSR の値から，各ウェイへのウェイ選択信号の値が決定される．CWSR の値の更新は，プログラム中のウェイ切り替え命令の実行により行われる．

3.1.2 エネルギーモデル

NSWC を用いた組込みプロセッサの消費エネルギーの見積もり式を以下に示す．

$$TE_{total} = TE_{NSWC} + TE_{DCACHE} + TE_{main} + TE_{logic} \tag{1}$$

式 (1) において，TE_{NSWC}，TE_{DCACHE}，TE_{main} および TE_{logic} はそれぞれは NSWC の総消費エネルギー，データキャッシュの総消費エネルギー，メインメモリの総消費エネルギーおよびロジック部の総消費エネルギーを示す．TE_{NSWC} は以下の式で表すことができる．

$$\begin{aligned} TE_{NSWC} = \quad & n_{DP} \cdot E_{DP} + n_{SP} \cdot E_{SP} \\ & + n_{wch} \cdot E_{wch} + n_{missi} \cdot E_{missi} \\ & + t_{all}(N_{DP} \cdot P_{DPleak} + N_{SP} \cdot P_{SPleak}) \end{aligned} \tag{2}$$

式 (2) における n_{DP}，n_{SP} はそれぞれ DP ウェイおよび SP ウェイへのアクセス回数を示す．E_{DP} および E_{SP} は DP ウェイおよび SP ウェイへのアクセスあたりの消費エネルギーを示す．n_{wch} は NSWC におけるウェイ切り替え回数を示し，E_{wch} はウェイ切り替えのエネルギーオーバーヘッドを示す．n_{missi} は NSWC におけるキャッシュミス回数を示し，E_{missi} は NSWC でのミス処理による消費エネルギーである．P_{DPleak} および

P_{SPleak} はそれぞれ DP ウェイおよび SP ウェイのリーク電力である．N_{DP} および N_{SP} はそれぞれ NSWC 内の DP ウェイ数および SP ウェイ数である．t_{all} はプログラムの総実行時間を示す．また，式 (1) における TE_{DCACHE} は以下の式で表わされる．

$$TE_{DCACHE} = n_{dcache} \cdot E_{dcache} + n_{missd} \cdot E_{missd} + t_{all} \cdot P_{leakd} \tag{3}$$

式 (3) において，n_{dcache} はデータキャッシュへのアクセス回数，E_{dcache} はデータキャッシュのアクセスエネルギーを示す．n_{missd} はデータキャッシュミス数を示し，E_{missd} はデータキャッシュにおけるミス処理の消費エネルギーを示す．P_{leakd} はデータキャッシュのリーク電力を示す．また，式 (1) における TE_{logic} および TE_{main} は，ロジック部平均電力 P_{logic}，オフチップメモリ定常電力 P_{main}，オフチップアクセスエネルギー E_{main} およびオフチップアクセス回数 n_{main} を用いて以下のように表わす．

$$TE_{logic} = t_{all} \cdot P_{logic} \tag{4}$$

$$TE_{main} = t_{all} \cdot P_{main} + n_{main} \cdot E_{main} \tag{5}$$

t_{all} は以下の式で見積りを行う．

$$t_{all} = n_{hit} \cdot T_{hit} + n_{missi} \cdot T_{miss} + \alpha \tag{6}$$

n_{hit} は命令キャッシュヒット数，T_{hit} および T_{miss} はヒット時およびミス時の処理時間を示す．以上の式を用いて，NSWC を用いた組込みプロセッサの消費エネルギー見積もりを行う．

3.2 ウェイ切り替え命令の挿入

SWC のウェイ切り替えはウェイ切り替え命令を実行することで実現する．ウェイ切り替え命令は専用命令またはサブルーチン呼び出しにより実装する．提案手法での動的ウェイ切り替えは，プログラムに対しウェイ切り替え命令を挿入することで実現する．ウェイ切り替え命令は任意の位置に挿入可能であるが，挿入位置の決定を簡単化するため，ウェイ切り替えは基本ブロック間の遷移時に行うものとする．ウェイ切り替え命令挿入位置の決定は以下のフローに従って実行する．

（1） ウェイ切り替え命令挿入候補の抽出
（2） 競合性ミスを避けるコード配置の導入
（3） ウェイ切り替え命令挿入の決定

フロー 1 では，すべての基本ブロック間遷移の中からウェイ切り替え命令を挿入する候補を選定する．候補を限定することで，後のウェイ切り替え命令挿入の決定を容易に実行することができる．フロー 2 の競合性ミスを避けるコード配置の導入では，アプリケーションプログラムの実行開始から終了までを 1 つのウェイで実行したと仮定した際の競合性ミスを削減するコード配置を行う．最後にフロー 3 で，各候補に対しウェイ切り替え命令の挿入を決定する．

3.2.1 ウェイ切り替え命令挿入候補の抽出

ウェイ切り替え命令挿入候補の抽出方法を図 4 を用いて説明する．プログラムのプロファイルより，図 4 の左部に示す CFG が得られると想定する．CFG の各ノードは基本ブロックであり，エッジは基本ブロック間の遷移を示す．ウェイ切り替え命令の挿入は，CFG 上のエッジに対する切り替えウェイの割り当てと同義と考える．

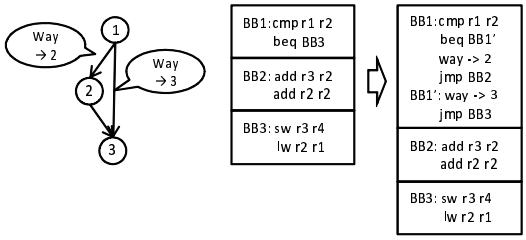

図 5　ウェイ切り替え命令の挿入

エッジに対し，特定のウェイへの切り替えを割り当てた場合，図 5 に示すようにウェイ切り替え命令の挿入が行われる．ウェイ切り替え命令を挿入した場合，ウェイ切り替え命令のコードサイズ分だけ後続のコードが配置されるアドレスがスライドする．コードが配置されるアドレスの変化はキャッシュメモリでのコード間の競合に影響を与えるため，競合性ミスを考慮したウェイ切り替え命令挿入が困難になる．このため，提案手法ではウェイ切り替え命令を挿入するエッジを限定し，予めウェイ切り替え命令のコードサイズと同サイズの nop 命令を挿入しておく．nop 命令をウェイ切り替え命令に変更することでウェイ切り替え命令挿入を実現し，アドレスの変更なしにウェイ切り替え命令の挿入が可能となる．しかしこの手法を用いることで，ウェイ切り替え命令に変更されなかった nop 命令の実行がオーバーヘッドとして発生する．

ウェイ切り替え命令挿入候補の抽出を行うために，基本ブロック群を複数のグループに分割する．異なるグループに属する基本ブロック間のエッジをウェイ切り替え命令挿入候補とする．図 4 の中央の図に示すように，はじめに各関数内においてグループ化を行う．関数のサイズが 1 ウェイのサイズよりも小さい場合，関数内の基本ブロック間での競合は発生しない．よって関数のサイズが 1 ウェイのサイズ以下である場合，関数に含まれる基本ブロックをグループ化する．このとき関数のサイズは，他の関数への遷移があるとき，ウェイ切り替え命令が挿入される可能性があるため，ウェイ切り替え命令挿入が行われたと仮定して見積もりを行う．関数のサイズが 1 ウェイよりも大きい場合，関数内の基本ブロックを複数のグループに分割する．このとき，競合する基本ブロックが異なるグループに属するように分割する．各関数におけるグループ化の後に，グループの統合を行う．グループ間の遷移回数が多いグループを選択し，グループのサイズの和が 1 ウェイのサイズ以下である場合は統合し 1 つのグループとする．ここで，統合する対象となるグループには 1 つの関数を分割したグループを含まない．グループの統合は，統合可能なグループがなくなるまで繰り返し行う．グループ化終了後に，異なるグループに属する基本ブロック間のエッジをウェイ切り替え命令挿入候補とする．

3.2.2 競合性ミスを避けるコード配置

キャッシュメモリにおける競合性ミスを削減する手法として，コード配置の変更を用いる [5]．本稿におけるコード配置は関数

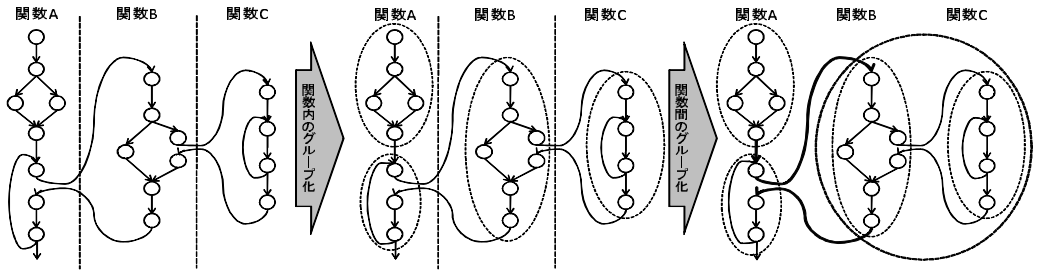

図 4　ウェイ切り替え命令挿入候補の抽出

単位で変更可能であると仮定している．

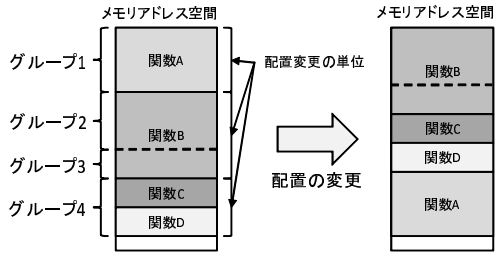

図 6　コード配置例

提案手法におけるコード配置例を図 6 に示す．同一グループに属する関数は連続してメモリアドレス空間にマッピングを行っている．グループのサイズは 1 ウェイのサイズよりも小さいため，連続してマッピングした場合同グループ内の競合は発生しない．このため，グループ単位で配置の入れ替えを行い，グループ間の競合を削減するコード配置に変更する．ここで 1 つの関数を分割した基本ブロックのグループである場合は，分割前の関数を配置の単位とする．配置の変更は，配置の単位となったグループおよび関数の中から 1 つを選択し，他のグループまたは関数と配置の入れ替えを行う．配置の変更後にキャッシュミス回数の見積もりを行い，キャッシュミス回数が減少した際の配置を記憶する．段階的にキャッシュミス数の少ない配置に変更していくことで，コード配置を改善していく．

3.2.3　ウェイ切り替え命令挿入の決定

ウェイ切り替え命令挿入候補およびコード配置が決定した後に，ウェイ切り替え命令挿入候補に対し，ウェイ切り替え命令挿入の有無および挿入する場合の切り替えウェイを決定する．初期状態はウェイ切り替え命令を挿入がない状態である．グループ間のキャッシュメモリにおける競合を考慮し，プログラム実行時のキャッシュメモリにおける競合が最も多いグループ間から順にウェイ切り替え命令の挿入を決定していく．競合が最も多いグループ間に存在するウェイ切り替え命令挿入候補となるエッジに対し，ウェイ切り替え命令を挿入しない状態および特定のウェイを指定したウェイ切り替え命令を挿入した状態のそれぞれにおける消費エネルギーを式 (1) を用いて見積もりを行う．ウェイ切り替え命令の挿入は，最も消費エネルギー値が小さい値となった状態に決定する．同様にすべてのグループ間に対し，ウェイ切り替え命令の挿入を決定する．

4. 評価実験

4.1 実験概要

提案手法を用いた際の消費エネルギー削減効果を評価する実験を行った．評価は，セット・アソシアティブ方式のキャッシュメモリを用いた場合 (SA)，文献 [7] で示される Way Placement を実行する手法 (WP) および SWC の動的ウェイ切り替えを行う場合 (SWC) と提案手法 (NSWC) の比較により行う．

評価実験は，ベンチマークプログラムのメモリマップおよび命令セットシミュレータを用いて得たアクセスレースを入力とし，各手法における消費エネルギーの見積もりを行う．消費エネルギー見積もりを行うために，セット・アソシアティブ方式，Way Placement，SWC および NSWC のキャッシュシミュレータを作成し，アクセストレースを用いてキャッシュアクセス回数，キャッシュミス数およびウェイ切り替え回数の計算を行う．キャッシュシミュレータの出力を用いて，式 (1) により見積もりを行った．評価ベンチマークプログラムには EEMBC ベンチマークを利用した．

4.2 実験環境

プロセッサのロジック部の消費電力は，東芝社製の MeP においてゲートレベルシミュレーションを実行して得た SAIF ファイルを入力とし，Synopsys 社の電力解析ツールである Power Compiler を用いて見積もりを行った．オフチップメモリのアクセスエネルギーおよび定常消費電力は，MICRON 社の Mobile DDR SDRAM のパラメータを用いた [10]．

命令キャッシュの構成は 4KB，2 ウェイ，ラインサイズ 32 バイトとした．キャッシュメモリおよびオンチップ RAM の消費エネルギーは，CACTI5.1 を用いて見積もりを行った [9]．CACTI は HP(High Performance)，LSTP(Low Standby Power) および LOP(Low Operating Power) の 3 種類のデバイスタイプによるパラメータの見積もりが可能である．SP ウェイおよび DP ウェイにおける消費エネルギーを見積もるために，それぞれ LSTP および LOP を利用した．

評価環境における CWSR はオンチップ RAM 上の固定アドレスに配置することを想定し，ウェイ切り替え命令の実行による CWSR の更新のオーバーヘッドは表 2 に示す値を用いた．

4.3 評価結果および考察

評価結果を図 7 に示す．セット・アソシアティブ方式を用いた場合 (SA) と比較して，提案手法 (NSWC) において 7%〜20%の消費エネルギー削減効果が確認された．また，実行時

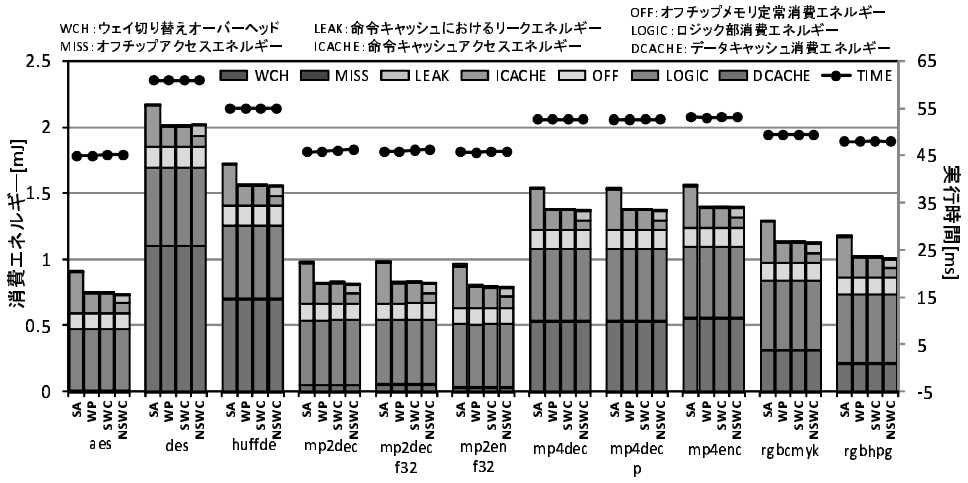

図 7 評価結果

表 1 キャッシュメモリのパラメータ

デバイスタイプ	LSTP	LOP
V_{dd}[V]	1.2	0.8
V_{th}[mV]	554	315
Read energy[pJ]	52.43	25.06
Leak power[mW]	0.0118	1.431
Access time [ns]	1.93704	1.0999

表 2 CWSR 更新のオーバーヘッド

切り替え時間	15[ns]
消費エネルギー	48.34896[pJ]

間の増加は 1%以下であった．Way Placement を用いた手法 (WP) と比較した場合，提案手法は最大 2%程度の消費エネルギー削減効果を確認した．また，SWC の動的ウェイ切り替えを行った場合 (SWC) と比較すると，最大 2.3%程度の消費エネルギー削減効果が確認された．WP と比較した際の消費エネルギー削減効果が小さい原因として，命令キャッシュにおけるミス率の低さが挙げられる．SWC および NSWC において動的ウェイ切り替えを用いることで，命令キャッシュにおける競合性ミスが削減されている．しかし，競合性ミスの発生が少ないことから，消費エネルギーの改善が小さかったものと考えられる．また，動的ウェイ切り替えを行うにあたり，ウェイ切り替え命令およびウェイ切り替え命令挿入候補の部分に組み込んだ nop 命令の実行によるオーバーヘッドにより SWC および NSWC の消費エネルギーおよび実行時間が若干増加した点も原因と考えられる．

SWC と NSWC の結果を比較すると，命令キャッシュにおけるアクセスエネルギーは NSWC の利用により削減されるが，リークエネルギーの増大により命令キャッシュ全体の消費エネルギーに大きな変化は見られなかった．しかし，表 1 における LOP デバイスを用いる DP ウェイのアクセス時間は，LSTP を用いる SP ウェイと比較して小さい．このため，表 1 に示した値よりも DP ウェイの閾値電圧は上昇させることが可能である．閾値電圧の値を適切に設定することで，NSWC の利用による消費エネルギー削減の余地はあると考えられる．

5. 終わりに

本稿では，組込みプロセッサの省エネルギー化を目的として，NSWC の提案および NSWC の動的ウェイ切り替え手法の提案を行った．

謝辞 本研究は東京大学大規模集積システム設計教育研究センターを通し，株式会社半導体理工学研究センター，(株) イー・シャトルおよび富士通株式会社の協力で行われたものである．本研究は，科学技術振興機構 (JST) の戦略的創造研究推進事業 (CREST) によるものである．

文 献

[1] Ching-Long Su and Alvin Despain, “ Cache Design Trade-offs for Power and Performance Optimization: A Case Study ”, In Proc. of ISLPED, pp.63-68, August 1995.

[2] Patrick Hicks, Matthew Walnock, and Robert Michael Owens, “ Analysis of Power Consumption in Memory Hierarchies ”, In Proc. of ISLPED, pp.239-242, August 1997.

[3] S.Seger, “ Low Power Design Techniques for Microprocessors ”, ISSCC Tutorial note, February 2001.

[4] Lars Wehmeyer, Peter Marwedel, “ Fast, Efficient and Predictable Memory Accesses ”, Springer

[5] Hiroyuki Tomiyama and Hiroto Yasuura, “ Optimal Code Placement of Embedded Software for Instruction Cachea ”, In Proc. of European Design and Test Conference, pp.96-101, March 1996.

[6] David H. Albonesi, ”Selective Cache Ways: On-Demand Cache Resource Allocation ”, In Proc. of the 32nd Annual IEEE/ACM International Symposium on Microarchitecture, pp. 248-259, 1999.

[7] Timothy M. Jones, Sandro Bartolini, Bruno De Bus, John Cavazos, Michael F.P. O’Boyle, ”Instruction Cache Energy Saving Through Compiler Way-Placement ”, In Proc. of Design Automation and Test in Europe, pp. 1196-1201, March 2008.

[8] 松村忠幸，石原亨，安浦寛人, “ コード配置とメモリ構成の同時最適化による省電力化手法 ”, DA シンポジウム 2008 論文集, pp. 13-18.

[9] Shyamkumar Thoziyoor, Naveen Muralimanohar, Jung Ho Ahn, Norman P. Jouppi, “ CACTI5.1 ”, Technical Reports, HP Labs, http://www.hpl.hp.com/techreports/2008

[10] “ 128Mb: x16, x32 Mobile DDR SDRAM Features ”, *http://www.micron.com/*