キャッシュウェイ割当てとコード配置最適化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
九州大学大学院システム情報科学研究院

出版情報：情報処理学会シンポジウムシリーズ, pp.19-24, 2008-08-26. 情報処理学会
バージョン：

キャッシュウェイ割当てとコード配置最適化による組込みプロセッサの省エネルギー化

石飛百合子[1]　石原亨[2]　安浦寛人[3]

[1] 九州大学　大学院　システム情報科学府
[2] 九州大学　システム LSI 研究センター
[3] 九州大学　大学院　システム情報科学研究院

本稿は Selective way cache を搭載したプロセッサにおいて，CPU コア，キャッシュメモリおよびオフチップメモリの総消費エネルギーを削減するウェイ割当てとコード配置手法の提案を行う．提案手法は，命令キャッシュとして Selective way cache を用い，ウェイ切り替えを行う命令の挿入位置およびコード配置をコンパイル時に決定する．Selective way cache において選択ウェイを制限すると，キャッシュアクセスエネルギーの削減が可能であるが，競合性のミスが増加する可能性がある．本稿では，競合性のミスおよびウェイ切り替え時に発生するオーバーヘッドを考慮し，消費エネルギーを最小化するウェイ割当てとコード配置の探査を行った．商用プロセッサおよび SDRAM のパラメータを用いた評価実験を行い，同容量のセット・アソシアティブキャッシュと比較して 9%の消費エネルギー削減効果を確認した．

The Energy Reduction of Embedded Processors Using Cache Way Assignment and Code Placement

Yuriko Ishitobi[1]　Tohru Ishihara[2]　Hiroto Yasuura[3]

[1] Graduate School of Inf. Sci. & EE, Kyushu University
[2] System LSI Research Center, Kyushu University
[3] Faculty of Inf. Sci. & EE, Kyushu University

This paper proposes a way assignment and a code placement to a processor with a selective way cache memory for reducing the total energy consumption of a CPU core, cache memories and off-chip memories. In our approach, we decide insert points of instructions to change available ways in the selective way cache and a code placement at a compile time. First, we assign a way to each basic blocks and then a way change instruction is inserted to a point needs to change available ways. 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 9% compared to the result of a processor with a same size set associative cache memory.

1　はじめに

携帯電話，携帯ゲーム機および携帯型オーディオ機器などの小型携帯機器の高機能化および長時間駆動の実現が求められている．機器の高機能化と長時間駆動を両立させるために，搭載するマイクロプロセッサとメモリのさらなる高性能化と低消費エネルギー化が必要となる．マイクロプロセッサとメモリの低消費エネルギー化および高性能化を実現するための機構として，キャッシュメモリがある．キャッシュメモリは下層のメモリのコピーを保持しておくことで，低速かつアクセスエネルギーの大きい大容量メモリへのアクセス回数を削減し，高性能化および低消費エネルギー化を実現している．キャッシュメモリはヒット率を向上させるため，大容量化および複雑化が進んでいる．これに伴い，キャッシュメモリへのアクセスエネルギーが増大しており，ARM920T^{TM}

マイクロプロセッサでは消費電力の 25%，低消費電力プロセッサである StrongARM においても 27% の消費電力が命令キャッシュにより占められる例がある [1, 2, 3]．マイクロプロセッサおよびメモリの低消費エネルギー化には，アクセスエネルギーおよびヒット率を考慮したキャッシュメモリの構成およびその使用方法が重要となる．

キャッシュメモリのヒット率を向上させるため，ウェイ・セット・アソシアティブ方式が多く利用されている．ウェイ・セット・アソシアティブ方式を用いるキャッシュメモリは，アクセス時にウェイ数分のタグメモリアレイ読み出しおよびデータメモリアレイ読み出しが必要とする．キャッシュメモリのアクセスエネルギーはアクセス時に充放電するメモリアレイのワードラインおよびビットラインの負荷容量に依存するため，ウェイ・セット・アソシアティブキャッシュは高いヒット率を持つ一方で消費エネルギーが大きい．キャッシュメモリの消費エネルギー削減手法として，アクセスに利用するウェイを選択可能にしたキャッシュメモリ (以下 Selective way cache) を用いる手法が存在する [4]．Selective way cache は，アクセスに用いるウェイを限定することでアクセスエネルギーを削減することが可能である [6, 7]．しかし，ウェイを限定することで，限定前には発生しない競合性のキャッシュミスが発生する可能性がある．Selective way cache を用いた低消費エネルギー化には，使用ウェイの選択と競合ミスの影響を同時に考慮する必要がある

Selective way cache におけるウェイの選択は，キャッシュミスによる性能低下を観測し，許容する性能低下の範囲内で最少のウェイ数に決定する手法が用いられている．しかしこの手法では，限定するウェイ数のみを決定するため，未選択となるウェイが将来アクセスされる有用なデータを保持している可能性がある．キャッシュメモリにデータを保持している場合であっても，データの配置されたウェイが選択されていない状態ではキャッシュミスが発生する．アプリケーションプログラムがあらかじめ決定している場合，プログラムの動作を解析し適切に使用ウェイを選択することで，未選択ウェイに配置されたデータに対するキャッシュミスを削減することができると考えられる．

提案手法では，コンパイル時に基本ブロック単位に選択ウェイを切り替える命令を挿入する位置を決定する．プログラムの実行トレースを解析し，各基本ブロックに対しアクセス時に用いるウェイを割当てる．割当てを決定した後，ウェイ切り替え命令を必要な位置に挿入する．各基本ブロックを実行する際は，常に割当てられたウェイを使用するため，キャッシュラインの再利用を促進できると考えられる．また，提案手法では基本ブロック単位のウェイ切り替え命令の挿入に加え，メモリアドレス空間内のコード配置を行うことを検討している．コード配置を同時に決定することで，キャッシュメモリにおける競合を抑制し，より低消費エネルギー化に効果的なウェイ切り替えを行うことができる．

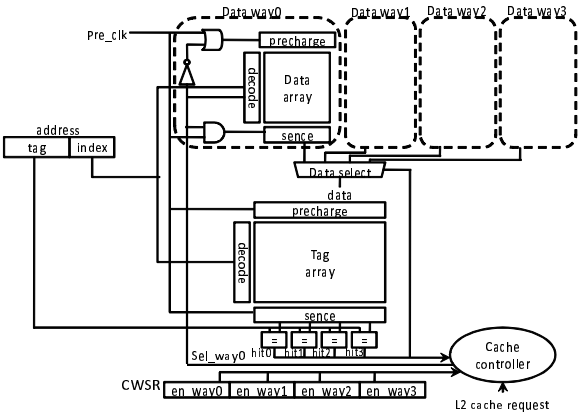

図 1: Selective cache ways

本稿の構成は次の通りである．2 章では関連研究として，Selective way cache のアーキテクチャと動作の説明および命令キャッシュで競合性ミスを削減するコード配置の説明を行う．3 章は Selective way cache 搭載のプロセッサにおける低消費エネルギー化手法の提案を行う．4 章では提案手法の評価実験の説明および評価結果の考察を示す．5 章で本稿をまとめる．

2 関連研究

2.1 Selective way cache

D.H.Albonesi は [4] でウェイの使用および不使用を切り替えることが可能なキャッシュメモリのアーキテクチャである Selective cache ways を提案している．図 1 にウェイ切り替え可能なキャッシュメモリの構成図を示す．提案している Selective cache ways では，ウェイ・セット・アソシアティブキャッシュにおける各ウェイのデータのメモリアレイが異なるメモリのサブアレイに分割されている．キャッシュメモリは CWSR(Cache Way Select Register) を保持し，CWSR の各 en_way ビットは各ウェイの使用，不使用を示す．CWSR の値はウェイ切り替えを行うための専用命令により書き換えられる．キャッシュコントローラは CWSR を参照してウェイの選択信号の制御を行う．en_way0 が 0 である場合，Sel_way0 も

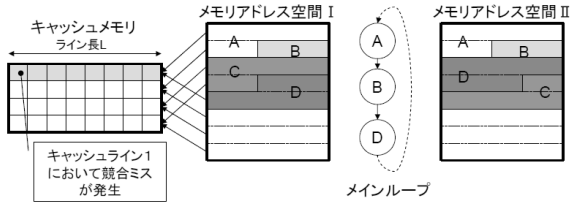

図 2: コード配置による競合ミスの削減

また 0 となる．Sel_way0 が 0 となる場合，way0 のデコーダではインデックスアドレスのデコードは行われず，データアレイのワードラインはアサートされない．また，プリチャージ回路へのクロック入力をマスクするため，ビットラインのプリチャージも行われず，センスアンプによる信号の増幅も行われない．このため，en_way を 0 としたウェイにおいてはデータアレイのワードラインおよびビットラインの充放電による消費エネルギーは発生しない．

[4] では，ウェイの切り替えを行う基準として性能低下の閾値 (PDT:Performance Degration Threshold) を用いている．性能低下が PDT を超えない最小のウェイ数を選択することで，性能低下 2%未満で 4 ウェイ・セット・アソシアティブキャッシュと比較して 40%の消費エネルギー削減を実現している．

2.2 競合ミスを削減するコード配置

マイクロプロセッサは固有のメモリアドレス空間を保持している．メモリアドレス空間内にはプログラムおよびデータなどのコードがマッピングされる．各コードがマッピングされるメモリアドレスを決定することをコード配置と呼ぶ．メモリアドレス空間内には命令や変数等の種類ごとに固有の領域が存在し，各コードは種類に応じた領域内にマッピングされる．

メモリアドレス空間からキャッシュメモリへのマッピングは一意に決定しているため，キャッシュメモリで競合するアドレスを考慮してコード配置を決定することで，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である．H.Tomiyama らは命令キャッシュにおける競合性ミスを最小化するコード配置を提案している [5]．図 2 にコード配置によるキャッシュメモリの競合ミス削減を行う例を示す．メモリアドレス空間内の各メモリブロックはキャッシュメモリのあるラインに対応している．ここでのメモリブロックとは，メモリアドレス空間をキャッシュラインサイズごとに区切った各領域を意味する．図 2 の左のメモリアドレス空間でメインループを実行した場合，A と D の配置されたメモリブロックが対応するキャッシュラインが競合するため，競合性のミスが発生する．一方で右のメモリアドレス空間に変更した場合，キャッシュラインの競合が発生しないため，メインループ実行時に競合ミスが発生しない．[5] では，命令キャッシュにおける競合ミスを最小化する問題の定義および定式化を示している．

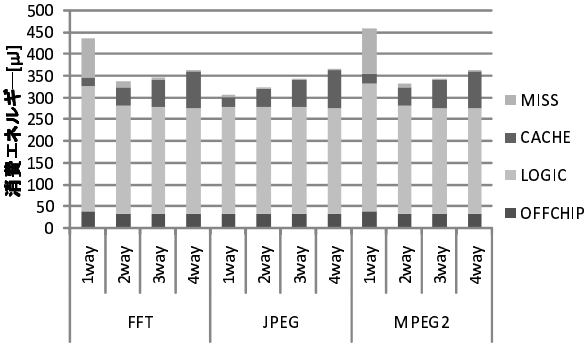

図 3: 選択ウェイ数と総消費エネルギー

3 提案手法

3.1 選択ウェイと総消費エネルギー

Selective way cache におけるウェイの選択数と総消費エネルギー値の関係の例を図 3 に示す．

棒グラフの値は下から各々ロジック部およびオフチップメモリの定常電力による消費エネルギー，キャッシュアクセスによる消費エネルギーおよびミス処理の消費エネルギーである．総消費エネルギーが最小となるウェイ数はアプリケーションプログラムにより異なることがわかる．ミス処理の消費エネルギー，ロジック部およびオフチップメモリの消費エネルギーはキャッシュミス数に依存して増減するため，ウェイ数が少ないほど大きな値をとる傾向がある．キャッシュアクセスによる消費エネルギーは選択ウェイ数の増加とともに大きな値をとる．選択ウェイ数を減らすことで，キャッシュアクセスの消費エネルギーを削減可能である．しかしアプリケーションプログラムによっては，キャッシュミスが増加し総消費エネルギーが増加する可能性がある．

3.2 ウェイ割当てとウェイ切り替え

Selective way cache を用いたプロセッサにおいて，プログラム実行時に選択ウェイを適切に切り替えることで，小さなアクセスエネルギーを維持しながら高いキャッシュヒット率を得ることができると考えられる．本稿では，命令キャッシュに Selective way

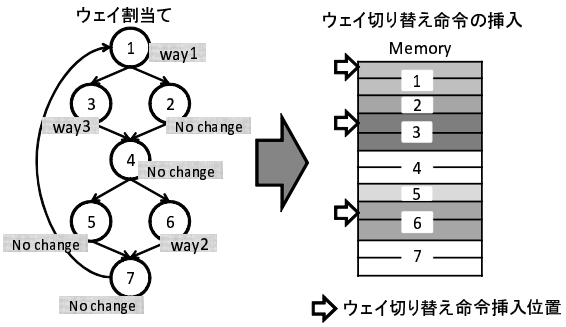

図 4: ウェイ割当て

cache を用いたプロセッサを対象とした消費エネルギー削減手法の提案を行う．提案手法では，アプリケーションプログラムのソースコードにウェイ切り替えサブルーチンへの分岐命令を挿入し，静的にウェイの切り替え位置を決定する．ウェイ切り替え命令の挿入位置を決定するために，アプリケーションプログラム中の各基本ブロックに対し，アクセスに用いるウェイの割当てを行う．本稿におけるウェイ割当てでは，各基本ブロックに対し Selective way cache の中のウェイを１つ割当てる．Selective way cache は常に１ウェイのみアクセスされるため，アクセスエネルギーを大きく削減することができる．図 4 にウェイ割当ての概念図を示す．ウェイ割当ては特定のウェイの指定またはウェイ切り替えなしを選択する．特定のウェイの指定を行う場合は，基本ブロックの入り口でウェイ切り替えを行う必要があるため，ソースコード中にウェイ切り替え命令の挿入を行う．ウェイ切り替えなしを選択した基本ブロックは，ウェイ切り替えを行わず，実行時に選択中のウェイを用いてアクセスを行う．

3.3 ウェイ割当ておよびコード配置の決定

ウェイ切り替え命令の実行からウェイの切り替えが完了し，通常プログラムの実行が再開するまでに必要な時間が，ウェイ切り替えのペナルティになる．頻繁に連続して交互に実行される基本ブロック同士に対し，異なるウェイを割当てウェイ切り替え命令を挿入した場合，プログラム実行時にウェイ切り替えが大量に発生する．ウェイ切り替えのペナルティによる実行時間の増加は，プロセッサおよびメモリにおいて定常的に消費される電力による消費エネルギーの増加につながる．実行時間を増加させないためには，頻繁に連続して実行する基本ブロック間はウェイの切り替えを行わず，同じウェイを用いて実行する必要がある．しかしながら，連続して頻繁に実行される基本ブロックのメモリアドレスがキャッシュメモリで競合する場合，ウェイの切り替えを行わなければ競合性のミスが増加する可能性がある．ウェイ切り替えおよびキャッシュミス双方の増加を防ぐため，提案手法ではメモリアドレス空間内のコード配置変更を行う．コード配置変更により，頻繁に連続して実行する基本ブロック同士のキャッシュメモリでの競合をなくし，ウェイ切り替えを行わない場合もキャッシュミスを増加を防ぐことが可能である．

3.4 問題定義

ウェイ割当てとコード配置は，CPU コア，オンチップメモリおよびオフチップメモリを含めた総消費エネルギーを最小化する目的で決定する．総消費エネルギー (E_{total}) を見積もるエネルギー見積もり関数を以下に示す．

$$E_{total} = \begin{array}{l} n_{ic} \cdot E_{ic} + n_{miss} \cdot E_{miss} \\ +t_{total} \cdot (P_{logic} + P_{off}) + TE_{dc} \end{array} \quad (1)$$

$$t_{total} = \begin{array}{l} CC(n_{ic} \cdot CN_{inst} + n_{miss} \cdot CN_{miss} \\ +n_{way} \cdot CN_{way}) + T_{dc} \end{array} \quad (2)$$

n_{ic} および E_{ic} は命令キャッシュへのアクセス回数およびアクセスあたりの消費エネルギーを表す．本稿におけるウェイ割当ては，単一ウェイの割当てであることから，E_{ic} は 1 ウェイ選択時のアクセスエネルギーとなる．n_{miss} および E_{miss} はキャッシュミス回数およびキャッシュミスあたりの消費エネルギーを表す．t_{total} は総実行時間を示し，P_{logic} はロジック部の平均消費電力，P_{off} はオフチップメモリの定常電力を示す．TE_{dc} はデータアクセスによる総消費エネルギーを表す．CC はクロックサイクル時間を表し，CN_{inst}，CN_{miss} および CN_{way} は各々１命令実行，キャッシュミスの処理およびウェイの切り替えに必要なクロックサイクル数である．n_{way} はウェイ切り替えがプログラム実行中に発生する回数である．T_{dc} はデータアクセスに必要な総実行時間を表す．消費エネルギー見積もり関数を最小にする各基本ブロックへのウェイ割当ておよび関数のコード配置を求める問題として定義される．

4 評価実験

4.1 実験概要

Selective way cache を搭載したプロセッサにおいて，ウェイ切り替え命令の挿入とコード配置の変更を行い，消費エネルギー削減効果の評価を行う．評価には，C 言語で記述した Selective way cache のシミュレータを用いる．Selective way cache のシミュ

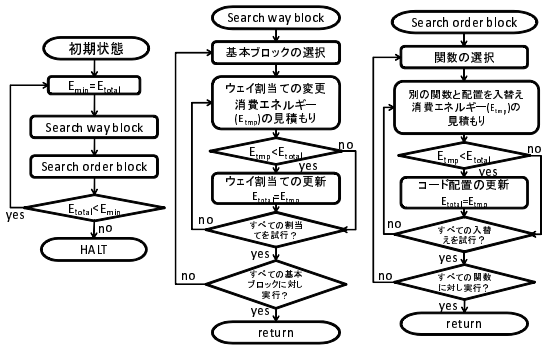

図 5: 探索アルゴリズム

表 1: 評価に用いたパラメータ

E_{ic}(1way)	0.0212 [nJ]	P_{off}	3.0 [mW]
E_{ic}(2way)	0.0424 [nJ]	P_{logic}	9.03 [mW]
E_{ic}(4way)	0.0848 [nJ]	CN_{inst}	2.99 [cycle]
E_{miss}	5.254 [nJ]	CN_{way}	20 [cycle]
CC	7.5 [ns]	CN_{miss}	31.56 [cycle]

表 2: FFT のパラメータ

パラメータ名	値
BB 数	406(113)
関数数	26

レータを用いて，ウェイ切り替え命令の挿入およびコード配置の変更を行ったアプリケーションプログラムのコードを実行した際のキャッシュミス回数およびウェイ切り替え回数の見積もりを行う．消費エネルギー値は，キャッシュミス回数およびウェイ切り替え回数から，式 (1),(2) を用いて算出する．

ウェイ切り替え命令の挿入およびコード配置を決定する際に用いたアルゴリズムを図 4.1 に示す．初期状態はウェイ切り替え命令およびコード配置を行わず，すべて同一のウェイを用いてアクセスする状態である．アルゴリズムでは，Search way block と Search order block の 2 つのブロックを繰り返し実行する．Search way block では，すべての基本ブロックに対しウェイ割当ての変更と消費エネルギーの見積もり，ウェイ割当ての更新を行う．基本ブロックを 1 つ選択し，特定のウェイを割当てた場合およびウェイの切り替えを行わない場合での消費エネルギーを見積もり，最も消費エネルギーが低下するウェイ割当てに更新する．Search order block では，関数単位でのコード配置の入れ替え，消費エネルギーの見積もり，コード配置の更新を繰り返す．関数を 1 つ選択し，他のすべての関数と配置を入れ替えた状態での消費エネルギーを見積もり，最も消費エネルギーが低下するコード配置に更新する．Search way block と Search order block を消費エネルギー値の低下が停止するまで繰り返し実行する．

4.2 実験環境

4.2.1 評価対象

評価に用いるプロセッサのパラメータは，東芝社製の MeP をベースとし命令キャッシュとして 4 ウェイの Selective way cache を搭載したプロセッサの値を用いた．Selective way cache は 8KB 容量，64 セットであり，ラインサイズは 32B である．プロセッサは Selective way cache の命令キャッシュに加えて，16KB 容量のデータキャッシュを持つ．比較対象として，命令キャッシュとしてダイレクトマップ方式，64 セットのキャッシュメモリ，2 ウェイ，64 セットのキャッシュメモリおよび 4 ウェイ，64 セットのキャッシュメモリを用いたプロセッサの消費エネルギー見積もりを行った．評価に用いたプロセッサ，Selective way cache および SDRAM のパラメータを表 1 に示す．

キャッシュメモリのアクセスエネルギー (E_{ic}) およびキャッシュメモリにおけるミスエネルギー (E_{miss}) は，回路シミュレータ SPICE を用いてキャッシュメモリを構成するメモリモジュールのアクセスエネルギーを測定し算出した．ウェイ切り替えに必要な実行サイクル数 (CN_{way}) は，Cadence 社製の Verilog-XL を用いたゲートレベルシミュレーションを行い求めた値である．ロジック部の平均消費電力 (P_{logic}) は SYNOPSYS 社製の電力解析ツールである Power Compiler を用いて求めた．命令あたりの実行サイクル数 (CN_{inst}) およびキャッシュミス処理に必要な実行サイクル数 (CN_{miss}) は RTL シミュレーションを行って得た値の平均値である．オフチップメモリの定常電力 (P_{off}) およびアクセスエネルギー (E_{miss}) は，Qimonda 社の SDR Mobile-RAM のモデルを用いた．評価に用いたベンチマークプログラムは FFT である．表 2 に FFT の基本ブロック数および関数数を示す．表内の基本ブロック数の括弧内の値は，すべての基本ブロックの中で機種依存でない関数内の基本ブロック数であり，ウェイ切り替え命令の挿入は，これらの基本ブロックのみを対象とした．実行開始から 100 万命令の実行アドレストレースを取得し，キャッシュシミュレータの入力とした．

4.3 評価結果

評価結果を図 6 に示す．グラフの棒グラフは消費エネルギーを示し，下からオフチップメモリの定常

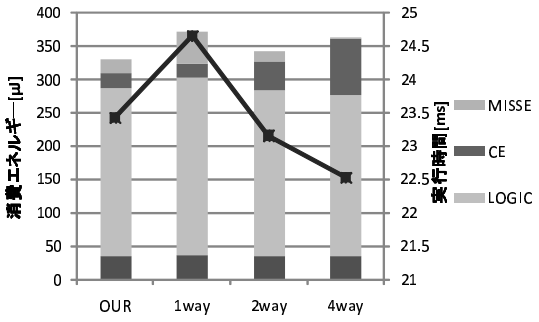

図 6: 評価結果

電力による消費エネルギー (OFFCHIP)，ロジック部の消費エネルギー (LOGIC)，キャッシュアクセスの消費エネルギー (CE) およびキャッシュミス処理の消費エネルギー (MISSE) である．折れ線グラフは実行時間 (TIME) を示している．実験結果は左から提案手法を用いた際の消費エネルギーおよび 64 セットの 1 ウェイ，2 ウェイおよび 4 ウェイのキャッシュメモリを用いた際の消費エネルギーである．1 ウェイのキャッシュメモリと比較して 11%，2 ウェイのキャッシュメモリと比較して 3.6%，4 ウェイのキャッシュと比較して 9%の消費エネルギー削減効果を確認した．一方実行時間は，1 ウェイのキャッシュメモリと比較して 5%の向上，2 ウェイのキャッシュメモリと比較して 1.1%の低下，4 ウェイのキャッシュと比較して 4.0%の低下を確認した．提案手法におけるウェイ切り替え命令の挿入数は 12 であった．

4.4 考察

常に 1 つのウェイのみを選択しているため，提案手法におけるキャッシュアクセスの消費エネルギーは 1 ウェイのキャッシュメモリと同等の値となっている．1 ウェイのキャッシュメモリと比較した際の消費エネルギーの削減要因として，キャッシュミス回数の低下が挙げられる．1 ウェイのみでは多く発生していた競合性のキャッシュミスを，ウェイの切り替えおよびコード配置の変更により削減できたためである．2 ウェイおよび 4 ウェイのキャッシュメモリと比較すると，消費エネルギー削減の要因として，キャッシュアクセスエネルギーの違いが挙げられる．2 ウェイでは提案手法の 2 倍，4 ウェイでは 4 倍の消費エネルギーが必要なことが挙げられる．提案手法は，キャッシュミス数が 2 ウェイおよび 4 ウェイと比較して大きいことに加え，ウェイ切り替えのオーバーヘッドが発生するため，実行時間は 2 ウェイおよび 4 ウェイのキャッシュメモリと比較して増加している．実行時間の著しい増加を防ぐために，ウェイ割当ておよびコード配置の探査に実行時間の制約が必要になると考えられる．

5 まとめ

本稿では，Selective way cache を搭載したプロセッサにおけるウェイ割当てとコード配置による低消費エネルギー化手法の提案を行った．プロセッサとメモリの総消費エネルギーの削減を目的とし，各基本ブロックに対してのウェイ割当ておよび関数のコード配置変更を行い，同容量のセット・アソシアティブキャッシュと比較して 9%の消費エネルギー削減を確認した．今後の課題として，基本ブロックのサイズが大きな場合の単一ウェイの割当てはキャッシュミスの増加の原因となりうるため，ウェイ割当てを単一ウェイに限定せず，複数ウェイを考慮していく．

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，株式会社東芝，シノプシス株式会社，日本ケイデンス株式会社の協力で行われたものである．本研究の一部は，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．

参考文献

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

[2] ARM Ltd., “ ARM Processor Core Overview ” , *http://www.arm.com/products/CPUs/*

[3] J. Montanaro et al., “ A 160 MHz, 32b 0.5W CMOS RISC Microprocessor ”, In Proc. of ISSCC, February 1996.

[4] David H. Albonesi, “ Selective cache ways: On-demand cache resource Allocation ”, In Proc. of the 32nd Annual IEEE/ACM International Symposium on Microarchitecture, pages 248-259.

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

[6] 山口誠一朗, 大山裕一郎, 国武勇次, 松村忠幸, 石飛百合子, 山口聖貴, 李東勲, 金田裕介, 舟木敏正, 室山真徳, 石原亨, 佐藤寿倫, “ 負荷変動に瞬時適応可能なマルチパフォーマンスプロセッサの設計と評価 ”, 電子情報通信学会技術研究報告, Vol.107, No.558, pages 1-6, March 2008.

[7] T. Ishihara, S. Yamaguchi, Y. Ishitobi, T. Matsumura, Y. kunitake, Y. Oyama, Y. Kaneda, M. Muroyama and T. Sato, “ AMPLE: An Adaptive Multi-Performance Processor for Low -Energy Embedded Applications ”, In Proc of IEEE Symposium on Application Specific Processors 2008, June 2008.