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

2007－ARC－174 (21)
2007／8／2

自動メモ化プロセッサの低消費エネルギー化

島崎裕介[†] 池内康樹[††,☆] 鈴木郁真[††,☆☆]
津邑公暁[†] 松尾啓志[†] 中島康彦[†††]

我々は，計算再利用技術に基づく自動メモ化プロセッサを提案している．本稿では，自動メモ化プロセッサに電力評価モジュールを実装しシミュレータレベルでの消費エネルギー評価を行った．メモ化により高速化を図ることが可能だが，プログラムによってはメモ化の効果が現れず，メモ化機構追加分のエネルギーを余分に消費する場合がある．そこで計算再利用率に応じてメモ化機構への電力供給を遮断する機能を実装した．メモ化を中断しない場合，SPEC CPU95 では平均 14%，GA プログラムでは平均で 1.8%のエネルギー増加となったが，メモ化を中断したことで，SPEC CPU95 で平均 1.5%の増加，GA プログラムでは平均 4.7%の減少となり消費エネルギーを削減することができた．メモ化中断の動作を取り入れても削減サイクル数の低下は微少に抑えることができた．この結果から，メモ化の中断によりサイクル数削減率を維持しつつ消費エネルギーの増加を抑えることができることが分かった．

An Energy Reduction Technique for Auto-Memoization Processor

Yusuke Shimazaki,[†] Yasuki Ikeuchi,[††,☆] Ikuma Suzuki,[††,☆☆]
Tomoaki Tsumura,[†] Hiroshi Matsuo[†]
and Yasuhiko Nakashima[†††]

We have proposed an auto-memoization processor. We implemented power-analysis method for our auto-memoization processor. This paper describes the energy consumpiton of auto-memoization processor. With memoization, some programs turn out to finish faster, but others do not. In the latter case, the processor increases their energy comsumption, without speed-up by the memoization. To decrease the energy comsumption, we added a function to the auto-memoization processor to discontinue memoization. The result of the experiment with SPEC CPU95 suite benchmarks and GENEsYs benchmarks shows that original memoization units increase whole energy by 14% and 1.8% on geometric mean respectively. After adding the function to discontinue memoization, in each benchmarks, the processor turned out to increase whole energy comsumption by 1.5% with SPEC CPU95, and decrease whole energy consumption by 4.7% with GENEsYs, on geometric mean. Consequently, by discontinuing memoization, the memoization units can decrease total energy consumption with small speed-down against traditional auto-memoization processor.

1. はじめに

ゲート遅延が支配的であったこれまでは，微細化による高クロック化で高速化を実現できた．しかし配線遅延の相対的な増大に伴い，高いクロックだけでは高速化を実現しにくくなっている．このような状況の下 SIMD やスーパスカラなどの命令レベル並列性（ILP: Instruction Level Parallelism）に基づく高速化手法が注目されてきた．しかし，多くのプログラムは明示的な ILP を持たないことや，ILP を抽出できる場合でもメモリスループットなどの資源的制約があることから，これらの手法にも限界がある．

そこで我々は，従来の高速化手法とは着眼点の異なる計算再利用技術に基づく，自動メモ化プロセッサを提案している[1]．**メモ化（Memoization）**[2] は本来，主に lisp などで使用されるプログラミングテクニックであり，計算量の大きい関数に対してその入出力を保存しておくことで，同一入力による当該関数の再計算を省略し実行を高速化する手法である．我々の提案する自動メモ化プロセッサは，既存のバイナリプログ

† 名古屋工業大学
Nagoya Institute of Technology
†† 豊橋技術科学大学
Toyohashi University of Technology
††† 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
☆ 現在，(株) ACCESS
Presently with ACCESS Co.,Ltd.
☆☆ 現在，トヨタ自動車 (株)
Presently with Toyota Motor Corp.

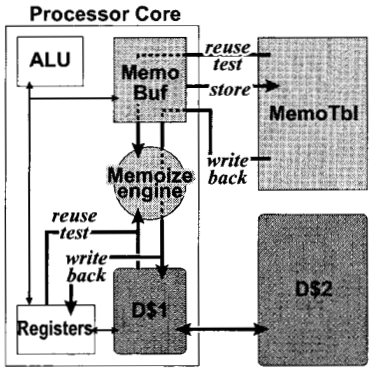

図 1 Structure of Auto-Memoization Processor.

ラムにおいて関数を実行時に動的に検出し，その入出力をハードウェアで記憶することで再計算の省略を自動的に行うアーキテクチャである．

この自動メモ化プロセッサは，動的に切り出した命令区間の入出力を表の形で保存する．また入力一致比較のため，当該区間実行のたびにこれを検索する必要がある．我々はこの表を CAM (Content Addressable Memory) を用いて構成することを仮定しているが，この表は参照頻度が大きく，また CAM 自体も消費電力が大きいため，プロセッサ全体のエネルギー消費量を大きく増加させてしまう可能性がある．

計算再利用技術の消費電力に対する影響に関しては，load/store 命令のみを再利用する場合に消費電力が抑えられる場合もあるという報告があるが[3]，これまで厳密な調査は行われていなかった．そこで本稿では，SimpleScalar 向け電力評価フレームワークである Wattch を参考に自動メモ化プロセッサシミュレータに電力評価機構を実装し，その消費エネルギーの評価を行った．また，メモ化の技術を用いても，プログラムによってはメモ化の効果がなく高速化が図れない場合があることが分かっている．このようなプログラムに対してメモ化の効果を動的に観測し，効果が得られにくいと判断した場合にはメモ化機構への電力を遮断してメモ化を中断する動作を実装し，評価を行った．

2. 自動メモ化プロセッサ

2.1 概要と動作モデル

メモ化とは，関数の入出力ペアを配列等に記憶させることで，当該関数の同一入力による実行を省略する高速化手法である．我々の提案している自動メモ化プロセッサは，プログラム実行中に関数を命令区間として動的に検出し，それらを自動的にメモ化する．具体的には，`call` 命令のターゲットと `return` 命令の間の区間を関数として検出する．

図 1 にプロセッサ構成の概略を示す．自動メモ化プロセッサは，入出力を記憶するためのテーブル（以下，**MemoTbl**）と，MemoTbl への書込バッファ（以下，**MemoBuf**）を持つ．メモ化機構は上記命令区間の開始を検出すると，MemoTbl に記憶されている当該区間の入力アドレス全てに対応する値をキャッシュから読みだし，MemoTbl 内の入力値とその値とを比較する．MemoTbl 内に完全に一致するエントリが存在した場合，そのエントリに対応する出力を MemoTbl から読みだし，レジスタおよびキャッシュに書き戻すことで命令区間の実行を省略する．

一致するエントリが存在しなかった場合，通常どおり命令区間を実行するが，その際レジスタおよびキャッシュへの参照を入力，書込みを出力として MemoBuf に記録する．そして命令区間を終端まで実行すると，MemoBuf 内に蓄積された入出力セットを MemoTbl に保存する．関数の入出力として扱われるのは引数および関数内で参照・書込みが行われた変数であるが，局所変数は除外できる．我々の手法では，一般に OS がデータサイズおよびスタックサイズの上限を決定することを利用し，この上限および関数呼出が行われる直前のスタックポインタの値と変数アドレスとの関係から，局所変数を判別している．

2.2 メモテーブルの構成

一般に命令区間内においては，ある入力の値によって次に参照すべき入力アドレスが変化する．変数に主記憶アドレスが格納されている場合や，条件分岐の存在がこの原因である．つまりある命令区間の全入力パターンはツリー構造で表すことができ，ある入力セットはそのツリー上の 1 本のパスで表される．このようなデータ構造の格納・検索を可能とするため，我々は MemoTbl を，以下の複数の表を用いて構成している．

RF: 命令区間の開始アドレスを記録しておくための表であり，RAM で構成する．

RA: 命令区間の入力アドレスを記録しておくための表であり，RAM で構成する．

RB: 命令区間の入力値を記録しておくための表であり，CAM で構成する．

W1: 命令区間の出力アドレスおよび出力値を記録しておくための表であり，RAM で構成する．

紙面の都合上，詳細は文献 1) にゆずるが，MemoTbl 検索手順の概要は以下の通りである．まず現在のレジスタ上の入力値を RB から検索する．RB の各エントリは，次入力アドレスを格納する RA エントリへのインデクスを保持している．RB 内で入力が一致するエントリが存在した場合，RA から得た次入力アドレスを用いてキャッシュを参照し，次入力値を得る．この入力値を用いて再び RB を検索する．これを繰り返し，全ての入力の一致が確認できると，入力セットの終端を保持する RB エントリは W1 へのインデクスを保持しており，これを用いて W1 を参照して出力値を得，これをレジスタおよびキャッシュに書き戻すことで命令区間の処理を省略する．

3. 消費電力の見積り

3.1 Wattch

自動メモ化プロセッサのエネルギー消費量を見積もるにあたり，**Wattch**[4] を参考に，シミュレータに消費電力推定の機構を実装した．

Wattch は，アーキテクチャレベルの消費電力シミュレータであり，SimpleScalar へのパッチという形で提供されている．Wattch はプロセッサ内ユニットを RAM，CAM，組合せ回路，クロック回路の 4 つに大別し，ユニットそれぞれに対する静電容量 C，ゲート稼働率 a と，電源電圧 V_{dd}，クロック周波数 f を用いて，消費電力を $P_d = CV_{dd}^2af$ として算出する．C および V_{dd} は仮定されたプロセスルールから算出し，プロセッサ内のユニットの使用頻度 a を動的に計測することで，P_d を算出する．

ここで C は，その実装モデルより決定される．Wattch では cacti[5] を参考に，実装モデルを決定している．cacti はキャッシュにおいて遅延時間が最適となるハードウェア構成を決定するツールであり，その遅延時間とハードウェア構成をもとにキャッシュ全体やキャッシュの TLB (CAM) の消費電力を算出する．

RAM : エントリ数，幅，読み書きポート数をパラメータとして消費電力が決定される．デコーダ，語ライン，ビットライン，出力のそれぞれでモデル化され，総消費電力が算出される．

CAM : RAM と同様，タグライン，マッチライン等をベースに算出される．

組合せ回路 : リザルトバスや ALU など，それぞれの構成に応じて消費電力が計算される．

クロック回路 : クロック線，クロックバッファ，クロックロードのそれぞれにおいてモデル化を行い，クロック回路における総消費電力が算出される．

3.2 実　　装

まず，メモ化機構に含まれないプロセッサ要素であるクロック回路，キャッシュ，ALU，レジスタ書込バスに関しては，Wattch の評価式をそのまま移植する形で実装した．

また 2.2 節で述べたように，我々の想定するメモ化機構は，MemoBuf および MemoTbl（RF, RA, RB, W1）から成る．これらについては，それぞれ以下のように RAM および CAM として消費電力評価関数の実装を行った．

MemoBuf : MemoTbl への書込バッファとして頻繁にアクセスされる．Wattch における 2 ポートの 1 次キャッシュと同じ電力評価式で実装した．ブロックサイズを 32B，総容量を 19kB とした．

RF : 命令区間表であり複数ポートは必要ないため，1 ポートの 1 次キャッシュとして実装した．サイズは幅 46B，エントリ数 256 の，総容量 12kB とした．

RB : 入力値セットのための表であり，連想検索が必要となる．Wattch では TLB が CAM を使って構成されているため，これと同じ電力評価式を用いて実装した．幅 36B，深さ 1k 行の，総容量 36kB とした．

RA，**W1** : 入力値のアドレス表および出力値表である．1 ポートの 2 次キャッシュと同じ評価式で実装した．RA，W1 の幅はそれぞれ 25B，75B とした．エントリ数は RB と同じ 1k であるため，総容量はそれぞれ 25kB，75kB となる．

4. メモ化プロセッサの低消費エネルギー化

4.1 メモ化機構と消費エネルギー

前述したようにメモ化は，MemoBuf 及び MemoTbl (RF,RB,RA,W1) のメモ化機構を必要とし，これらユニットの消費電力を考慮する必要がある．

一方で，メモ化はプログラムの値の局所性を利用した高速化手法であるため，値の局所性が無いプログラムの場合には，メモ化による高速化が期待できない．メモ化によりプログラムの実行の高速化が実現できるかどうかはメモ化を行ってみなければ分からず，プログラムによってはメモ化機構を用いる必要がない場合がある．そのようなプログラムにおいて問題となるのは，メモ化機構を稼働させることによる消費電力及びエネルギーの増大である．

4.2 メモ化中断アルゴリズム

メモ化によるサイクル数削減効果の見込めないプログラムは，メモ化を行う必要が無いといえる．そこで，メモ化の効果が顕著でないプログラムにおいてはメモ化を中断し，メモ化機構への電力供給を遮断することで無駄なエネルギー増加を抑えることが考えられる．

この考えに基づく自動メモ化プロセッサの実現にあたり，新たに 2 バイト程度の小さなカウンタを 2 つ用意する．なお，この追加したカウンタはプログラムカウンタに比べてインクリメントされる頻度が極めて低く，プロセッサ全体に及ぼす電力増加は誤差の範囲内である．

メモ化中断の手法として，メモ化の効果を示す値 r を次式にて定義する．

$$r = n_r/n_f$$

ここで n_r はメモ化により計算再利用が成功した回数，すなわち図 1 における `writeback` の起きた回数を表し，n_f は実行対象となるプログラム中で関数の命令区間が呼ばれた回数を表す．一定回数（N_f 回）関数が呼ばれる毎に r を求めることで，メモ化機構が有効に働いているか検知することができる．中断を決定する閾値を R_{th} とし，$r < R_{th}$ の場合にメモ化の中断を試行し，$r \geq R_{th}$ の場合には r をリセットしてメモ化を継続する．また，N_f が小さい場合，r を算出するための十分な n_r が得られない可能性が高いため，$n_f = 2^k N_f$ となる毎に，r が閾値を 1 度のみ下回ることを容認した．閾値 R_{th} を求める時の n_r の値を N_r と表すとし，$n_f = N_f$ となる時の n_r の値を N_r と表

表 1 シミュレータ諸元

D1 Cache 容量	32 KBytes
ラインサイズ	32 Bytes
ウェイ数	4
レイテンシ	2 cycles
ミス ペナルティ	10 cycles
D2 Cache 容量	2 MBytes
ラインサイズ	32 Bytes
ウェイ数	4
レイテンシ	10 cycles
ミス ペナルティ	100 cycles
Register Window 数	4 sets
Window ミス ペナルティ	20 cycles/set
MemoBuf. サイズ	19 kB
MemoTbl. サイズ (CAM 部)	36 kB
MemoTbl. サイズ (RAM 部)	112 kB
MemoTbl. ⇔ レジスタ 比較	32 Byte/cycle
MemoTbl. ⇔ キャッシュ 比較	32 Byte/2cycle

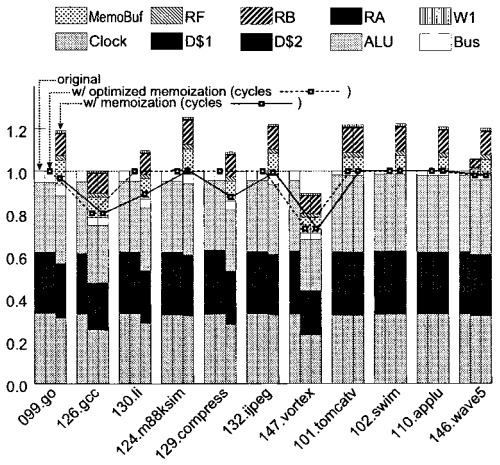

図 2 総消費エネルギー比 (SPEC CPU95)

すとし，本稿では各値を，$N_r = 16, N_f = 1024, k = 2$ としたが，各値は上下させることでより従来の自動メモ化プロセッサに近づけたり，反対にメモ化の中断を起こりやすくすることが可能となる．

5. 評　　価

5.1 評価環境

以上で述べたように，自動メモ化プロセッサシミュレータに消費電力評価の機能を追加し，更に再利用率の観測に基づくメモ化中断機構を実装し，評価を行った．シミュレータは単命令発行の SPARC-V8 をベースとしている．評価に用いたパラメータを**表 1** に示す．なお命令レイテンシは SPARC64-III[6] を参考にした．

5.2 SPEC CPU95

まず SPEC CPU95 (train) を gcc 3.0.2 (-O2 -msupersparc) によりコンパイルし，スタティックリンクにより生成したロードモジュールを用いて評価を行った．**図 2**，**表 2** に結果を示す．

表 2 SPEC CPU95 の結果

	中断動作あり	中断なし
中断プログラム数 (11 個中)	8	0
平均削減サイクル数	5.0%	7.4%
平均消費エネルギー比	+1.5%	+14%
最大サイクル削減率	23%	23%
最大消費エネルギー削減率	11%	11%

各プログラムは 3 本の棒グラフで表されており，左がメモ化を行わない場合，中央が本稿で提案するメモ化を中断した場合，右が従来の自動メモ化プロセッサの場合の消費エネルギーを表している．それぞれメモ化なしの場合で正規化した．凡例はその内訳を示しており，順にクロック (Clock)，一次キャッシュ (D$1)，二次キャッシュ (D$2)，演算器 (ALU)，レジスタ書込バス (Bus)，およびメモテーブルの各構成要素 (MemoBuf, RF, RB, RA, W1) における消費エネルギーである．また折線で結んだ点は，メモ化を行った場合の各プログラムの実行サイクル数を表しており，やはりメモ化なしの場合のサイクル数で正規化してある．従来手法を実線，提案手法を破線で示した．

評価の結果，メモ化のための機構により消費電力は約 25%増加した．また，メモ化機構の電力消費の多くは MemoBuf および RB (CAM) によって占められていることが分かる．MemoBuf は一次キャッシュと同様の構成を想定しており容量も小さいが，参照頻度が高いため一次キャッシュ以上の電力を消費している．また RB はサイズおよび参照頻度がそれほど大きくはないが，CAM で構成されているためやはり消費電力が大きい．

また実行サイクル数と，メモ化機構以外の部分の消費エネルギーを比較すると，多くのプログラムでサイクル数の減少以上に消費エネルギーが減少していることが分かる．これは，計算結果の再利用によりキャッシュミスやキャッシュアクセス自体が減少したことと，入出力登録時，命令区間内で書込みが発生した変数はその後キャッシュではなく MemoBuf から参照されることに起因していると考えられる．

SPEC CPU95 では，124.m88ksim および 147.vortex のようにおよそ 20%以上のサイクル数が削減可能な場合に，消費エネルギーをオリジナルと同程度もしくはそれ以下に抑えることができると分かる．しかし，中断せずメモ化を行った場合で平均削減サイクル数 7.4%と，メモ化による効果が得にくいプログラムが多い．その結果平均消費エネルギーは 14%増となり，増加幅は比較的現実な値に抑えられているとはいえ，129.compress や 132.ijpeg 等のようにメモ化の効果が現れないプログラムにおいて，余分なエネルギー消費が目立つ．

一方，提案手法では 099.go や 126.gcc 等，計 8 つのプログラムにおいてメモ化の中断が起きているが，

124.m88ksim, 147.vortex 及び 101.tomcatv では中断が起こらず従来の自動メモ化プロセッサと同じエネルギー消費となっている．全体の平均削減サイクル数は 5.0%となり，自動メモ化プロセッサの 7.4%と比べても比較的小さな削減サイクル数の縮小に抑えることができた．平均消費エネルギーは 1.5%増となり，改良前の 7.4%と比べかなりのエネルギー消費の抑制が実現できている．また，中断の起きた 8 つのプログラムの消費エネルギーに限っては，平均で 18%増だったが，中断したことにより僅か 0.9%増にまで減少した．以上より，中断により消費エネルギーの抑制が効果的に行われたことが分かる．

メモ化の中断が起こらなかった場合として次の 3 つの理由が考えられる．

(1) プログラム全体に値の局所性があり，メモ化の恩恵を受け続けた
(2) プログラムが小さく，閾値判定の前に終了した
(3) プログラムが実行サイクル数の大きな関数を呼んだ

(1) には 124.m88ksim, 147.vortex が該当する．逆に 126.gcc, 130.li ではプログラムの前半に値の局所性が無いため，本来メモ化により実行サイクル数を 10%以上削減できるところが中断によりメモ化が起きず，結果高速化されなかった．このようなプログラムに対しては，中断後もメモ化を定期的に再開し値の局所性の有無を確認することで，メモ化による高速化を図ることが可能だと考えられる．

(2) は，閾値判定の間隔を狭めることで幾分か対処はできるが，メモ化による MemoTbl への登録があまり起きていない可能性が高いため，そもそもメモ化には適していないことが考えられる．

(3) は 101.tomcatv が該当する．101.tomcatv は，比較的大きなプログラムであり，関数の呼び出し自体少なくはない．しかし中断が起こらずエネルギー消費の増加分が目立っているのは，無数の小さな関数と，ごく少数だが非常に大きな関数の呼び出しが存在し，その大きな関数にメモ化効果が現れなかったためである．そういったプログラムに対しては，例えばクロックやレジスタへの書き込みを測定し，ある一定限度の閾値を準備することで関数の呼び出し中でもメモ化を中断することが可能だが，その大きな関数が再度呼ばれてメモ化の恩恵をうける可能性が無いとは判断できない．以上の理由で 101.tomcatv は，従来の自動メモ化及び提案する中断を取り入れるプロセッサの両方共に消費エネルギー抑制が期待できないプログラムであったといえる．

5.3 GENEsYs

次に，汎用 GA ソフトウェア GENEsYs 1.0 を用いて評価を行った．GENEsYs には，GA のベンチマークや定量的評価に良く用いられる De Jong のテスト関数，巡回セールスマン問題，フラクタル関数などの標準的な関数をはじめとする，24 種の適合度関数が実装されている．

表 3 GENEsYs パラメータ

交叉率	60.0 %
突然変異率	0.1 %
個体数	50
世代数	25 世代
他のパラメータ	default 値

表 4 GENEsYs の結果

	中断動作あり	中断なし
中断プログラム数 (24 関数中)	15	0
平均削減サイクル数	14%	17%
平均消費エネルギー比	-4.7%	+1.8%
最大サイクル削減率	72%	72%
最大消費エネルギー削減率	65%	65%

メモ化効果はプログラムの記述方式に依存して変化するため，今回は，メモ化効果の高い適合度関数をメモ化効果が得られやすいように書き換えたもの[7] を用いて評価した．また，交叉アルゴリズムは 2 点交叉とした．**表 3** に GENEsYs の実行パラメータ，**図 3** 及び**表 4** にその結果を示す．図 2 同様，各適合度関数の消費エネルギーをそれぞれ 3 本の棒グラフで表した．使用した gcc のバージョン，コンパイルオプションなども SPEC CPU95 と同様である．

GENEsYs では大きなメモ化効果が得られており，中断せずメモ化を行った場合で全 24 適合度関数のうち 6 関数で，総消費エネルギーがメモ化なしの場合より抑えられていることが分かる．また GENEsYs では，特に全体に対して適合度計算量の占める割合の大きい関数，すなわち処理時間が長くかかる関数ほどメモ化の効果が得られることが分かっており GA は処理時間面においても消費エネルギー面においてもメモ化の恩恵をうけやすいプログラムであると言える．図 3 の結果より，GENEsYs における平均削減サイクル数は 17%，平均消費エネルギーは 1.8%増となった．しかし GENEsYs も SPEC CPU95 同様，メモ化の恩恵が現れない適合度関数がやはり存在し，結果的に無駄となったエネルギー消費の増加分がいくつかの適合度関数で目立っているのがわかる．

一方，提案手法による結果では，f1 をはじめ計 15 種の適合度関数でメモ化の中断が起きたが，f2, f3, f4, f5, f11, f12, f14, f16, f17 ではメモ化の中断が起こらず従来の自動メモ化プロセッサと同じエネルギー消費となった．平均消費エネルギーは 4.7%減となり，メモ化機構の追加により増加すると思われた消費エネルギーはメモ化中断の導入によって削減が可能であることが判明した．また，実行サイクル数の削減率は 14%となり，従来の自動メモ化プロセッサの 17%に近い削減率を維持できている．また，中断の起きた計 15 種の評価関数の消費エネルギーに限れば，中断なしメモ化で平均 12%増であったが，やはり中断したことにより僅か 0.9%増にまで減少したことからも，メモ化中断に

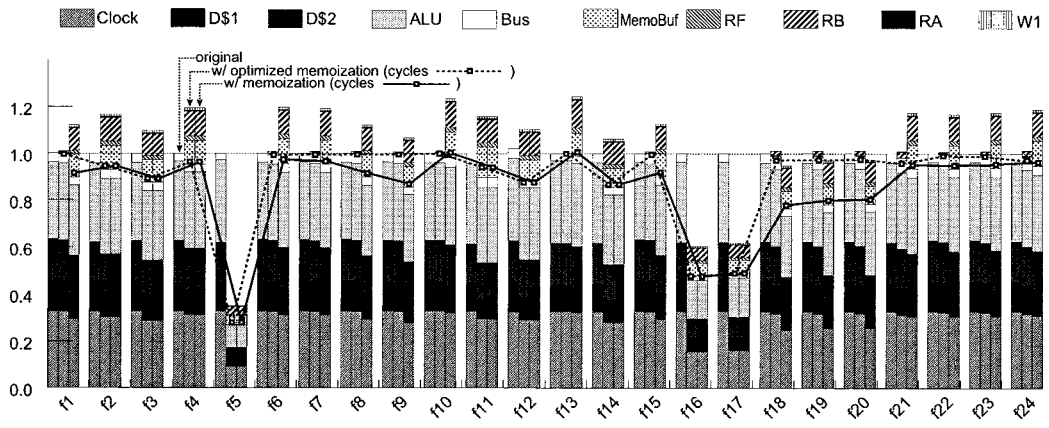

図 3 総消費エネルギー比 (GENEsYs)

より効果的な消費エネルギーの抑制が実現できたことが分かる．

但し，f2 や f4 はメモ化によるサイクル数削減効果が小さいが，メモ化の中断が起こらずエネルギーを消費し続けており，また，f18, f19, f20 の 3 つに関しては，本来消費エネルギーの削減が見込めるほど高速化されるが，プログラムの序盤でメモ化の中断が起きている．今後，プログラムの後半にメモ化の効果が現れる場合にも対処するため，中断閾値を動的に変化させたり，中断後も定期的にメモ化を再開する動作を組み込むといったことが必要であると考えられる．

6. おわりに

本稿では，計算再利用技術に基づく自動メモ化プロセッサに対し，その消費電力をアーキテクチャレベルで評価した．命令区間の入出力を記憶するための CAM 等による消費電力の大きな増加が懸念されたが，36kB の CAM を含むメモ化機構による消費電力増加は約 25% であった．また消費エネルギーでは，SPEC CPU95 で平均 14%の増加，GENEsYs で平均 1.8%程度の増加となることが確認できた．このことから，メモ化の効果が得られるプログラムに対しては僅かなエネルギー増加でサイクル数を削減できることが分かった．

次に，計算再利用率に応じてメモ化機構への電力供給を遮断する動作を実装したところ，消費エネルギーは，SPEC CPU95 で平均 1.5%の増加，GENEsYs で平均 4.7%の減少となり，更なる消費エネルギーの抑制及び削減が可能となった．

本稿で提案するメモ化中断は，実行しているプログラムの一部分を観測して中断の判断と決定を行うため，本来はメモ化の恩恵を受けられたプログラムでもメモ化を行えなかった場合がある．そのようなメモ化中断によるリスクを減らすため，今後の課題としては，メモ化機構への定期的な電力供給の再開などの対応が考えられる．また，他の高速化手法を採用した場合の消費エネルギー増加量との比較も行っていく予定である．

謝辞 本研究の一部は，文部科学省科学研究費補助金（萌芽研究 18650005，若手研究 (B) 19700041），および（財）大川情報通信基金研究助成金による．

参考文献

1) Tsumura, T., Suzuki, I., Ikeuchi, Y., Matsuo, H., Nakashima, H. and Nakashima, Y.: Design and Evaluation of an Auto-Memoization Processor, *Proc. of Parallel and Distributed Computing and Networks*, pp.245–250 (2007).
2) Norvig, P.: *Paradigms of Artificial Intelligence Programming*, Morgan Kaufmann (1992).
3) Yang, J. and Gupta, R.: Energy-Efficient Load and Store Reuse, *Intl. Symp. on Low Power Electronics and Design*, pp.72–75 (2001).
4) Brooks, D., Tiwari, V. and Martonosi, M.: Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, *Proc. of the 27th Annual Intl. Symp. on Computer Architecture*, pp.83–94 (2000).
5) Wilton, S.J. and Jouppi, N.P.: An Enhanced Access and Cycle Time Model for On-Chip Caches, Technical report, DEC Western Research Laboratory (1993).
6) HAL Computer Systems/Fujitsu: *SPARC64-III User's Guide* (1998).
7) 鈴木郁真，池内康樹，津邑公暁，中島康彦，中島浩：再利用による GA の高速化手法，情報処理学会論文誌：コンピューティングシステム，Vol.46, No.SIG 16(ACS 12), pp.129–143 (2005).