LC-005

自動メモ化プロセッサの消費エネルギー評価

島崎裕介[†1] 池内康樹[†2] 津邑公暁[†1]
中島浩[†3] 松尾啓志[†1] 中島康彦[†4]

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，シミュレータレベルでの消費電力評価の枠組で SimpleScalar への拡張である Wattch を参考に，自動メモ化プロセッサシミュレータに消費電力計算モジュールを実装し，評価を行った．メモ化対象の入出力を記憶するために CAM の使用を前提としており，これが消費エネルギーに大きな影響を与えると予想されたが，SPEC CPU95 では最大 26%，平均 14%の増加に抑えられ，GA プログラムでは最大で 23%増加したが，平均では 0.1%の減少となった．このように，高速化が実現できるプログラムでは，サイクル数の減少により総消費エネルギーが減少することもあることが分かった．

Energy Consumption of Auto-Memoization Processor

Yusuke Shimazaki,[†1] Yasuki Ikeuchi,[†2] Tomoaki Tsumura,[†1]
Hiroshi Nakashima,[†3] Hiroshi Matsuo[†1]
and Yasuhiko Nakashima[†4]

We have proposed an auto-memoization processor. This processor memoizes functions automatically and reuses their results. This paper describes the power consumption of auto-memoization processor. We implemented power-analysis method for our auto-memoization processor simulator modeled on Wattch: a patch for SimpleScalar. The result of the experiment with SPEC CPU95 suite benchmarks shows that memoization units increase whole energy up to 35% and 20% on average. Meanwhile, the result with genetic algorithm programs shows that the energy increases up to 23%, and *decreases* 0.1% on average. Admittedly, the memoization units increase total energy, but the elimination of total cycles by auto-memoization sometimes decreases it.

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ラムにおいて，関数を命令区間として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．

さて，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

†1 名古屋工業大学
Nagoya Institute of Technology
†2 （株）ACCESS
ACCESS Co.,Ltd.
†3 京都大学
Kyoto University
†4 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

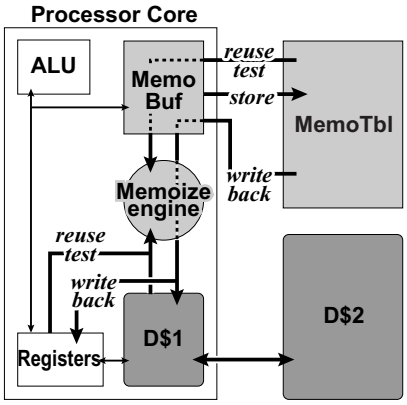

図 1 Structure of Auto-Memoization Processor.

れまで厳密な調査は行われていない．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．

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，一方でプロセッサ内の様々なユニットの使用頻度 a を計測することで，P_d 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，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. 評　　価

4.1 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，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 を参考にした．

4.2 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** に結果を示す．

各プログラムは 2 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，右がメモ化を行った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．

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

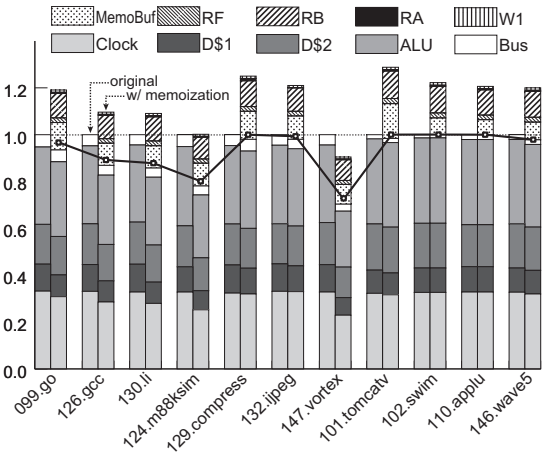

図 2　総消費エネルギー比 (SPEC CPU95)

評価の結果，メモ化のための機構により消費電力は約 30%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．

SPEC CPU95 は平均削減サイクル数 7%と，メモ化による効果が得にくいプログラムが多いが，124.m88ksim および 147.vortex のように約 30%のサイクル数が削減できれば消費エネルギーをオリジナルより抑えることができることが分かる．また，平均消費エネルギーはメモ化なしの場合の 14%増となり，参照頻度の高い CAM を用いているが，増加幅は比較的現実的な値に抑えられることが分かった．

また実行サイクル数と，メモ化機構以外の部分の消

表 2 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

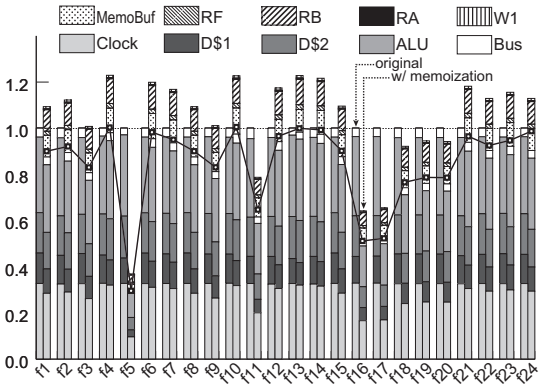

図 3 総消費エネルギー比 (GENEsYs)

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．

4.3 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．

GA は特に個体の適合度計算においてメモ化の効果が得られやすいことが分かっており[5]，今回はこの適合度関数をメモ化対象として評価を行った．また，交叉アルゴリズムは 2 点交叉とした．**表 2** に GENEsYs の実行パラメータ，**図 3** 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 2 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，全 24 適合度関数のうち 9 関数で，総消費エネルギーがメモ化なしの場合と同等あるいはそれ以下に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 18%，消費エネルギーの相乗平均は 0.1%の減少となった．また，f5 では削減サイクル数は 70%となり，消費エネルギーも 63%減少した．このように，メモ化の効果が大きいプログラムにおいては消費エネルギーはむしろ減少することが分かった．

5. 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 30% であった．また消費エネルギーでは，SPEC CPU95 で最大 26%，平均 14%の増加，GENEsYs で最大 23%の増加，平均 0.1%の減少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サイクル数の削減により総消費エネルギーが減少する場合もあることが分かった．

メモ化の効果が得にくいプログラムに対しては，当然ながら追加した機構による消費電力が総消費エネルギーを増加させてしまうが，メモ化による計算再利用率を動的に観測することは容易に実現できる．よって今後の課題としては，計算再利用率に応じてメモ化機構への電力供給を停止する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きたい．

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) 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).