ソフトウェアの消費エネルギー解析と最適化技術

石原, 亨
九州大学システムLSI研究センター

出版情報：回路とシステム軽井沢ワークショップ. 21, pp.343-348, 2008-04-21
バージョン：
権利関係：

ソフトウェアの消費エネルギー解析と最適化技術

Techniques for Estimating and Reducing the Energy Consumption of Embedded Software

石原　亨
九州大学システム LSI 研究センター
Tohru ISHIHARA
System LSI Research Center, Kyushu University

1 はじめに

携帯機器の普及と共に組込みシステムの省エネルギー化はますます重要になっている。しかし、組込みソフトウェア設計の分野では省エネルギー化に対する意識はそれほど高くない。本稿では、筆者らの提案するソフトウェアデバッガベースの消費エネルギー解析手法を紹介する[1]。組込みソフトウェア設計者が設計時に消費エネルギーを解析できる環境の構築を目標としている。本手法は、1）ソフトウェアデバッガから容易に抽出可能なパラメータを用いてプロセッサシステムの消費エネルギーをキャラクタライズする手法と、2）キャラクタライズにより得られたモデルからソフトウェアの消費エネルギーを解析する技術から構成される。モデルには線形近似モデルを使用する。ゲートレベルの電力見積もり値を基準値としてキャラクタライズを行う。プロセッサコア、キャッシュメモリおよび外部主記憶のエネルギー消費をキャラクタライズの対象とする。

デジタルシステムのエネルギー見積もり手法は今日までに数多く提案されている。ソフトウェアの振る舞いが消費エネルギーに与える影響を見積もる最も正確かつ高速な方法は実チップの電力を測定することである。しかし、チップの電力測定では、プロセッサ内部のホットスポット解析や、数μ秒オーダーの間に消費されるエネルギーを見積もることは難しい。一方、設計初期段階の見積もり手法の多くはゲートレベルやRTLのシミュレーション[2]をベースとしているため、大規模ソフトウェアの解析には膨大な時間を要する。命令セットシミュレータ（以下 ISS）を用いた電力見積もり手法も数多く提案されているが[12][15-18]、電力見積もり機能を持つ ISS は一部のプロセッサに限られているため、この機能が無い場合は、専用のシミュレータと電力モデルを開発する必要がある。本稿で紹介する手法は対象とするプロセッサシステムに対して半自動でキャラクタライズを行うことを特徴とする。生成された線形モデルは、既存のソフトウェアデバッガと組み合わせて使用することができる。

本稿の構成は次の通りである。2 章で関連研究を紹介し、3 章で提案手法の詳細を説明する。4 章では実験結果を示し、5 章で提案手法の応用例を紹介する。6 章で本稿をまとめる。

2 関連研究

文献[3,4]では実チップの電力測定手法が提案されている。これらの手法はホストマシンから制御可能なマルチメータを使用する。文献[3]の PowerScope はメモリや CPU などの部分システムの電流を個別に測定できないという問題がある。Itsy［4］では上記の問題が解決されているが、短時間の電流変化を測定できない。従って、上記の手法を用いる場合、プロセッサ上でプログラムを連続実行して、平均電流を測定することになる。

ISS を用いた見積もり手法も数多く提案されている。文献[5,6]では、同じ種類の命令を連続実行させた時の実チップの電流測定値を用いて各命令の消費エネルギーをモデル化する手法が提案されている。文献[7]では、上記の手法の拡張が行われている。命令アドレスやデータアドレス、オペランドの値およびレジスタファイルのアドレスを考慮に入れることにより見積もり精度が改善されることが報告されている。主な問題は、すべての命令および異なる命令間の影響を見積もるために大量のテストベンチを使用して繰り

返し電流測定を行う必要がある点である。文献[8,9]では一部の命令のみを使ってキャラクタライズを行うことにより測定の効率化を行っている。しかし、何れの手法もチップ内部の消費エネルギーの内訳を解析することは難しい。文献[10,11]では命令の実行回数をパラメータとした単純な線形近似により 10%未満の誤差でプロセッサコアの電力見積もりを行えることが示されている。 文献 [12]では、サイクル精度のシミュレータ（Trimaran [13]）を使用して VLIW プロセッサのパイプラインステージ毎の消費エネルギーを見積もる手法が提案されている。この手法は特定のプロセッサを対象とした手法であるため、ターゲットが変わればシミュレータおよび電力モデルを作り直す必要がある。文献[14] は RTL 記述からサイクル毎に各命令の平均エネルギーをキャラクタライズする手法を提案している。しかし、命令毎の平均エネルギーからプログラムの実行時に消費されるエネルギーを見積もる方法は示されていない。上記以外にも線形式を用いてプロセッサシステムの消費エネルギーを見積もる手法は多数提案されているが[15-20]、キャラクタライズの手法に関して議論した研究は少ない。 文献[21]では、より高精度な消費エネルギーの見積もり値を基準値とした重回帰分析により、精度の良いモデリングが行えるという一般的な議論が行われている。しかし、テストベンチの決定方法に関する議論はない。本稿では、キャラクタライズのためのテストベンチ生成のガイドラインを示し、精度の良い線形近似式を得る方法を提案する。係数の決定には線形計画法を用いる。

3 エネルギー消費モデル

式(1)に示す線形近似式を用いてプロセッサシステムの消費エネルギーを見積もる手法が数多く提案されている [1-2][5-12][14-21]。

$$E_{estimate} = \sum_{i=0}^{N} c_i \cdot P_i \qquad (1)$$

ここで、$E_{estimate}$、P_i、c_i、および N は、それぞれ消費エネルギーの見積もり値、線形モデルのパラメータ、各パラメータの係数、およびパラメータの数を表す。パラメータ P_i の種類とその係数 c_i の値が決定されれば後はパラメータ P_i の値を計測することにより、消費エネルギー$E_{estimate}$ を高速に求めることが出来る。

本稿では、ソフトウェアデバッガから容易に抽出可能なパラメータを用いてプロセッサおよび外部主記憶のエネルギー消費をキャラクタライズする方法を提案する。ソフトウェアデバッガを利用する理由は以下の通りである。

1. ソフトウェア開発者が利用しやすいこと
2. 容易にリターゲット可能であること
3. ハードウェアシミュレータと比較して高速であること

3.2 キャラクタライズの概要

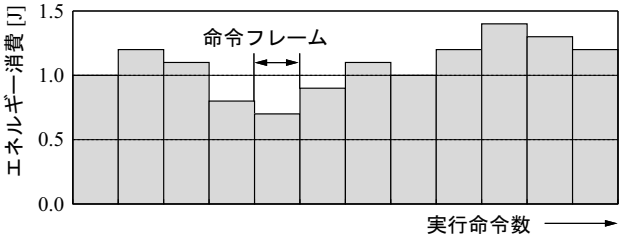

図 1　トレーニングベンチの分割

キャラクタライズ用のテストベンチを**トレーニングベンチ**と呼ぶことにする。まず、図１に示す通り、トレーニングベンチをサブベンチ（以下、命令フレーム）に分割する。命令フレーム毎にゲートレベルの電力見積もりを行い基準値とする。以下のステップで消費エネルギーのキャラクタライズを行う。

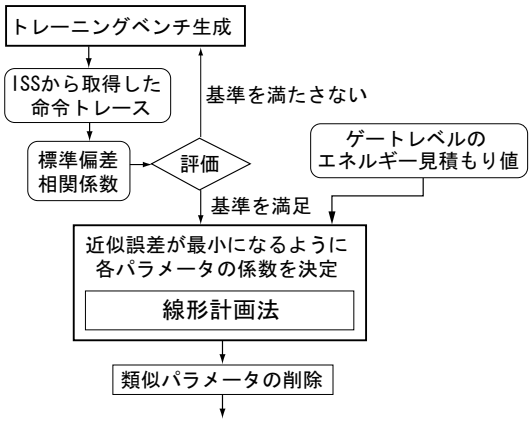

図 2　キャラクタライズの概要

1. パラメータの決定
2. トレーニングベンチを命令フレームへ分割

3. 命令フレーム毎に命令トレースを生成
4. 命令フレーム毎にパラメータ値を抽出
5. パラメータ毎に標準偏差と相関係数を評価
6. 標準偏差と相関係数の値が基準を満たすまでトレーニングベンチを変更
7. 命令フレーム毎に消費エネルギーを計測する（ゲートレベルの見積もり値を使う）。
8. 線形計画法を用いて係数を決定する。

3.3 パラメータとトレーニングベンチの決定

パラメータにはソフトウェアデバッガから容易に抽出でき、かつ消費エネルギーに影響を与えるものを選択することが重要である。多くのプロセッサシステムでは、メモリアクセスに要するエネルギーが非常に大きな割合を占めるため、キャッシュメモリやスクラッチパッドメモリ、外部メモリへのアクセス回数をパラメータとして使用することは必要不可欠である。これ以外にCPUがストールを起こす原因をパラメータとして使用する。本フレームワークでは以下のパラメータを使用してキャラクタライズを行う。

- 各命令の実行回数
- 命令/データキャッシュミス回数
- 命令キャッシュとデータキャッシュが同時にミスヒットする回数
- ロードまたはストア命令が連続して実行される回数（バーストアクセス回数）
- 分岐命令のうち分岐が不成立であった回数
- 分岐不成立と命令キャッシュミスが同時に発生する回数
- Read After Write (RAW)ハザードの発生回数
- データキャッシュミスとRAWハザードが同時に発生する回数
- 乗算命令や除算命令などのマルチサイクル命令がキャッシュミスを引き起こす回数

キャラクタライズで最も重要なのはテストベンチの決定である。無作為に選択したテストベンチを使用してキャラクタライズを行い、生成された線形式でエネルギー見積もりを行うと、100%以上の誤差を生じることを確認している。テストベンチは以下の二つの基準に基づいて生成する。

1. **各命令フレームから抽出した同じパラメータ値の標準偏差が基準値以上であること**
   命令フレーム（図１参照）毎のパラメータ値の変化が小さい場合には、そのパラメータが消費エネルギーに与える影響を見積もることが出来ない。従って、キャラクタライズを行う際にはパラメータの値が命令フレーム毎にばらついていることが望ましい。
2. **あるパラメータ値と別のパラメータ値との相関係数が基準値未満であること**
   パラメータ同士の相関が強い場合は、どちらかの影響が他方のパラメータによって隠蔽される。従って、各パラメータの値がなるべく独立に変化するようなテストベンチおよび命令フレームを作成する必要がある。

現在は上記の基準に基づいてテストベンチを手作成しているが、将来は命令セットやメモリの構成情報などからキャラクタライズ用のテストベンチを自動生成することを目標としている。

3.4 係数の決定方法

係数の決定には線形計画法を用いる。以下に変数を定義する。

- i: 命令フレームの番号
- j: パラメータの番号
- N: パラメータの数
- M: 命令フレームの数
- E_i: ゲートレベルシミュレーションで求めた i 番目の命令フレームの消費エネルギー値
- E'_i: 式(1)に示す線形近似式を用いて見積もった消費エネルギーの見積もり値
- Y_i: 見積もり誤差の絶対値 $|E'_i - E_i|$
- P_{ij}: 命令セットシミュレータを用いて抽出したパラメータの値
- c_j: パラメータの係数（決定変数）

Minimize

$$Y_{obj} = \sum_{i=0}^{M} Y_i$$

Subject to

$$-Y_i \le \left(\sum_{j=0}^{m} c_j \cdot P_{ij} \right) - E_i \le Y_i \quad (i = 0..M-1)$$

$$c_j,\ P_{ij},\ E_i,\ Y_i \ge 0 \quad (i = 0..M-1,\ j = 0..N-1$$

4 実験と結果

4.1 実験環境

実験にはM32R-IIプロセッサとSH3-DSPを使用した。何れのプロセッサとも Renesas Technology 社から東京大学VDECを通じて提供された、32 ビット RISC マイコンである。CPUは5段パイプライン構造を持っている。キャッシュメモリとスクラッチパッドメモリを内蔵する。生成した線形式を評価するために5種類のベンチマーク(JPEG エンコーダ、MPEG2 エンコーダ、compress、FFT、DCT)を用いた。

4.2 キャラクタライズの詳細フロー

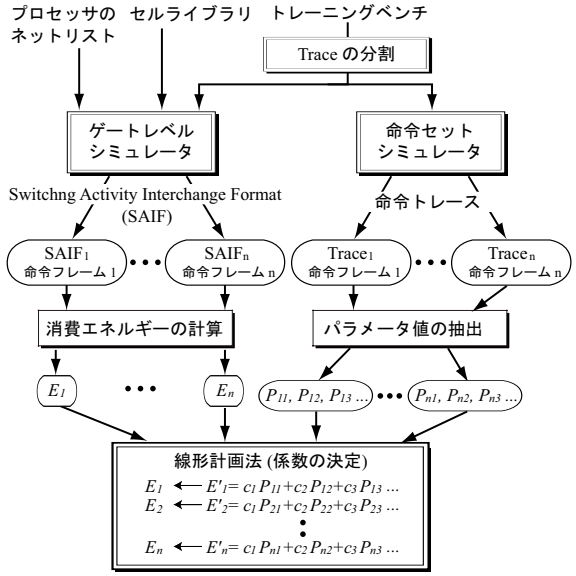

図 3 キャラクタライズの詳細フロー

キャラクタライズの具体的な流れを図3に示す。まず C コンパイラを用いてトレーニングベンチから目的コードを生成する。次にプロセッサのゲートレベルシミュレーションモデル上で目的コードを実行する。ゲートレベルのシミュレーションには Cadence 社の VerilogXL を用いた。ゲートレベルシミュレーションにより得られたゲートのスイッチング情報(SAIF:Switching Activity Interchange Format)とセルライブラリの消費エネルギー情報から命令フレーム毎の消費エネルギーを計算する。セルライブラリおよびメモリマクロライブラリには東京大学 VDEC を通じ日立製作所より提供を受けた、0.18μm テクノロジのライブラリを使用した。外部主記憶にはSDRAMを仮定した。SDRAMの消費エネルギーモデルは、Micron テクノロジ社が公開している電力計算スクリプト[21]を使用した。プロセッサの消費エネルギーの計算には SYNOPSYS 社の PowerCompiler を用いた。各命令フレームに対応する各種パラメータ値は命令セットシミュレータにより抽出する。命令セットシミュレータには各プロセッサ向けの GNU デバッガを用いた。係数を決定する際の LP ソルバには ILOG 社の CPLEX を使用した。

4.3 実験結果

キャラクタライズには各命令の実行回数やキャッシュミス回数など合計 82 個のパラメータを使用し、3.2 節で述べた方法に基づいて生成したトレーニングベンチを用いた。このトレーニングベンチは、各命令を一定回数実行するサブルーチン、キャッシュミスを引き起こすサブルーチンなどから構成される。トレーニングベンチの実行命令数は M32R-II で 375,000 命令、SH3-DSP で 140,000 命令である。キャラクタライズに要した時間は M32R-II で約 160 分、SH3-DSP で約 370 分であった(表2参照)。キャラクタライズにはIntel 社製 Xeon™ CPU 3.80GHz の計算機を使用した。

表 1. キャラクタライズに要した時間（分）

Target Processor	M32R-II	SH3-DSP
Gate-Level Sim.	127	328
Power Calculation	32	41
Instraction-Set Sim.	< 1	< 1
LP Solver	< 1	< 1
Total CPU Time	160	370

キャラクタライズは長時間を要するが、一度このステップが完了すると後はソフトウェアデバッガの実行速度（約 300,000 命令/秒）でソフトウェアの消費エネルギー解析を行うことが出来る。

実験結果を表1に示した。名前の最後に_opt を付けたベンチマークプログラムは-O3 オプションでコンパイルした目的コードである。5,000 命令実行した際の消費エネルギー200 ポイントに対して比較した結果である。実験の結果から 7%未満の平均誤差で見積もりが行えることを確認した。図 4 と図 5 には最も誤差の大きかった JPEG エンコーダ(JPEG_O)を、それぞれ M32R-II と SH3-DSP 上で実行させた際の消費エネルギーを示した。消

費エネルギーの傾向が正確に見積もられていることが確認できる。

表 2. 見積もり誤差

	M32R-II		SH3-DSP	
	平均誤差%	最大誤差%	平均誤差%	最大誤差%
JPEG	2.70	10.32	3.17	11.89
JPEG_O	6.09	16.46	6.33	10.02
MPEG2	1.54	3.97	1.32	3.41
MPEG2_O	1.78	5.15	1.31	5.63
compress	5.00	6.41	5.73	10.84
compress_O	4.35	7.18	1.73	15.15
FFT	1.55	6.87	1.27	3.26
FFT_O	1.45	5.59	1.15	4.75
DCT	1.42	8.58	1.12	2.20
DCT_O	1.47	8.07	1.51	3.04
Total	2.74	16.46	2.47	15.15

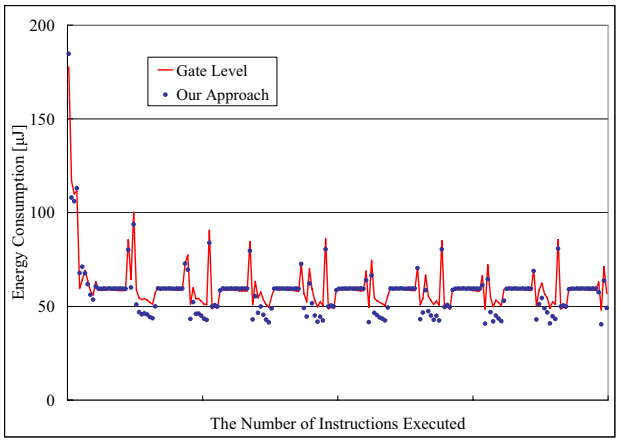

図 4 JPEG encoder を M32R-II で実行した結果

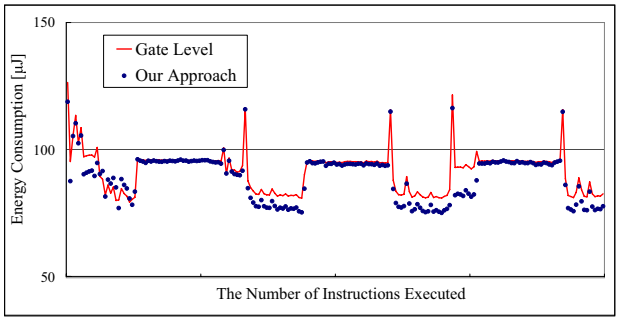

図 5 JPEG encoder を SH3-DSP で実行した結果

5 コード配置最適化への応用

5.1 コード配置の最適化

組込みプロセッサの消費エネルギーを改善する手法の 1 つとして，主記憶内のアプリケーションプログラムのコード配置を変更することによりメモリアクセスに要するエネルギーを削減する手法が数多く提案されている。文献[23]では、頻繁に実行されるプログラムコードをスクラッチパッドメモリ（以下 SPM）に配置することによりメモリアクセスに伴うエネルギーを削減する手法が提案されている。SPM はキャッシュメモリに比べてアクセス当りの消費エネルギーが小さいため、頻繁に実行される命令コードやデータをキャッシュの代わりに SPM に配置することによりメモリアクセスの消費エネルギーを削減することができる。文献[24]や[25]では、プログラム中の関数やデータオブジェクトを主記憶上に適切に配置することによりキャッシュの競合ミスを削減する手法が提案されている。文献[26]では、参照の局所性が小さいデータをキャッシュせずにバイパスすることによってキャッシュデータの無用な追い出しを減らしキャッシュミス率を削減する手法が提案されている。これらの手法は主にメモリアクセスの回数を目的関数としているため、必ずしもプロセッサシステム全体の消費エネルギーを最小にするとは限らない。

5.2 省エネルギー化のためのコード配置最適化

文献[27]で、筆者らは本稿の 3 章で述べた消費エネルギーの近似モデルと同様のモデルを用いてプロセッサシステム全体の消費エネルギーを削減する手法を提案している。具体的には、プロセッサシステムの詳細な消費エネルギーモデルに基づいて、目的コードを主記憶上のキャッシュ領域、SPM 領域、あるいはアンキャッシュ領域のいずれかに配置することによりプロセッサシステムの消費エネルギーを最小化する。

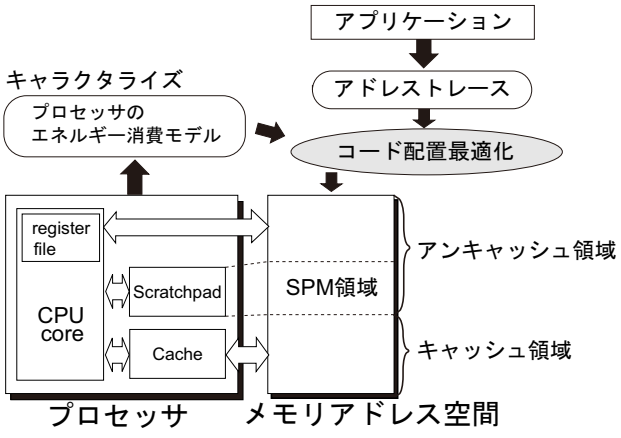

図 6 コード配置最適化のフロー

SH3-DSP と 3 種類のベンチマークプログラム(JPEG エンコーダ、MPEG2 エンコーダ、compress)を使用した実験により、筆者らの提案手法がプロセッサシステムの性能を劣化させることなく消費エネルギーを約 25%削減できることを確認した[27]。

6 おわりに

本稿ではマイクロプロセッサシステムの消費エネルギーをキャラクタライズする一手法を示した。また本手法を目的コードの配置最適化に応用することによりプロセッサシステムの大幅な省エネルギー化を達成できることを確認した。今後はキャラクタライズフローの自動化とマルチプロセッサシステムへの拡張を行う予定である。

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、ケイデンス(株)、シノプシス(株)、(株)ルネサステクノロジ、(株)日立製作所、及び、大日本印刷(株)の協力で行われたものである。本研究の一部は JST の CREST-ULP によるものである。

参考文献

[1] D. Lee, T. Ishihara, and H. Yasuura “An Energy Characterization Framework for Software-Based Embedded Systems,” in Proc. of Workshop on Embedded Systems for Real Time Multimedia, pp.53-58, Oct. 2006.

[2] W. Ye, N. Vijaykrishnan, M. Kandemir and M.J. Irwin, “The Design and Use of SimplePower: A Cycle-Accurate Energy Estimation Tool”, Proc. of 37th DAC, pp.340-345, June 2000.

[3] J. Flinn and M. Satyanarayanan, “Powerscope: a Tool for Profiling the Energy Usage of Mobile Applications”, in Proc. of the 2nd IEEE workshop on Mobile Computing Systems and Applications, pp.2-10, February 1999.

[4] W. R. Hamburgen, D. A. Wallach, M. A. Viredaz, L. S. Brakmo, C. A. Waldspurger, J. F. Bartlett, T. Mann, and K. I. Farkas, “Itsy: Stretching the Bounds of Mobile Computing”, IEEE Computer, vol. 34, pp.28-37, April 2001.

[5] V. Tiwari, S. Malik, and A. Wolfe, “Power Analysis of Embedded Software: A First Step towards Software Power Minimization”, in Proc. of ICCAD, pp.384-390, Nov. 1994.

[6] M. T. C. Lee, V. Tiwari, S. Malik and M. Fujita, “Power Analysis and Low-Power Scheduling Techniques for Embedded DSP Software”, in Proc. of the ISSS, pp.110-115, Sept. 1995.

[7] N. Chang, K. Kim, and H. G. Lee, “Cycle-Accurate Energy Consumption Measurement and Analysis: Case Study of ARM7TDMI”, In Proc. of ISLPED, pp.185-190, Aug. 2000.

[8] C. Brandolese, W. Fornaciari, F. Salice, and D. Sciuto, “An Instruction-level Functionality-based Energy Eestimation Model for 32-bit Microprocessors,” in Proc. of DAC, pp.346–351, June 2000.

[9] A. Sama, M. Balakrishnan, and J. F. M. Theeuwen, “Speeding Up Power Estimation of Embedded Software”, in Proc. of ISLPED, pp.191-196, Aug. 2000.

[10] T. Sinha, and A. P. Chandrakasan, “JouleTrack – A Web Based Tool for Software Energy Profiling”, in Proc. of DAC, pp.220-205, June 2001.

[11] A. Sinha, N. Ickes, A. P. Chandrakasan, “Instruction level and operating system profiling for energy exposed software”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.11, no.6, pp.1044–1057, Dec. 2003.

[12] M. Sami, D. Sciuto, C. Silvano and V. Zaccaria, “Instruction-Level Power Estimation for Embedded VLIW Cores” in Proc. of 8th Int’l Workshop on Hardware/Software Co-design, pp.34-38, May 2000.

[13] Trimaran: http://www.trimaran.org.

[14] C. T. Hsieh, L. S. Chen, M. Pedram, “Microprocessor Power Analysis by Labeled Simulation,” in Proc. of the Conference on DATE, pp.182–189, March 2001.

[15] D. Brooks, V. Tiwari, and M. Matonosi, “Wattch: A Framework for Architectural-Level Power Analysis and Optimization”, in Proc. of ISCA, pp.83-94, June, 2000.

[16] J. T. Russell and M. F. Jacome, “Software power estimation and optimization for high performance, 32-bit embedded processors,” in Proc. of ICCD, pp.328–333, Oct. 1998.

[17] S. Steinke, M. Knauer, L. Wehmeyer, and P. Marwedel, “An Accurate and Fine Grain Instruction-Level Energy Model Supporting Software Optimizations”, in Proc. of Int’l Workshop on Power And Timing Modeling, Optimization and Simulation, pp.3.2.1-3.2.10, September 2001.

[18] T. Li and L. K. John, “Run-time Modeling and Estimation of Operating System Power Consumption”, in Proc. of Int’l Conference on Measurements and Modeling of Computer Systems, pp.160-171, June 2003.

[19] G. Contreras and M. Martonosi, “Power Reduction for Intel XScale® Processors Using Performance Monitoring Unit Events”, in Proc. of ISLPED, pp.221-226, Aug. 2005.

[20] W. L. Bircher, M. Valluri, J. Law, and L. K. John, “Runtime Identification of Microprocessor Energy Saving Opportunities”, in Proc. of ISLPED, pp.275-280, Aug. 2005.

[21] T. K. Tan, A. Raghunathan, G. Lakshminarayana, N. K. Jha, “High-level Software Energy Macro-modeling”, in Proc. of DAC, pp.605-610, June 2001.

[22] “The Micron System Power Calculator”, *http://www.micron.com/support/part_info/powercalc*

[23] S. Stenke, L. Wehmeyer, B.-S. Lee, P. Marwedel, “Assigning Program and Data Objects to Scratchpad for Energy Reduction”, in Proc. of DATE, pp.409-415, 2002.

[24] S. McFarling, “Program Optimization for Instruction Caches”, in Proc. of ASPLOS, pp.183-191, April 1989.

[25] P.Panda, N.Dutt, and A.Nicolau, “Memory Organization for Improved Data Cache Performance in Embeded Processors”, in Proc. of ISSS, pp.90-95, November 1996.

[26] J. A. Rivers and E. S. Davidson, “Reducing Conflicts in Direct-Mapped Caches with a Temporality-Based Design”, in Proc. of the 25th Int’l Conference on Parallel Processing, pp.154-163, Aug., 1996.

[27] Y. Ishitobi, T. Ishihara, and H. Yasuura “Code Placement for Reducing the Energy Consumption of Embedded Processors with Scratchpad and Cache Memories,” in Proc. of Workshop on Embedded Systems for Real Time Multimedia, pp.13-18, Oct. 2007.