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

2006－ＳＬＤＭ－125（３）
2006／5／11

オペランドのビット幅を考慮したソフトウェアレベル消費エネルギー削減手法

山口誠一朗† 室山 真徳†† 石原 亨†† 安浦 寛人†††

† 九州大学 大学院システム情報科学府
〒816-8580 福岡県春日市春日公園 6-1
†† 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 大学院システム情報科学研究院
〒816-8580 福岡県春日市春日公園 6-1
E-mail: †seiichiro@c.csce.kyushu-u.ac.jp, ††{muroyama,ishihara}@slrc.kyushu-u.ac.jp,
†††yasuura@c.csce.kyushu-u.ac.jp

あらまし 本稿では，マイクロプロセッサベース組込みシステムのソフトウェアレベル消費エネルギー削減手法を提案する．提案手法はオペランドのデータをシフトし符号拡張ビット部の信号遷移を削減することにより消費エネルギーを削減する．提案手法は次の 3 つのステップから構成される．1) オペランドのデータを LSB 側から MSB 側へ最適なシフト量だけシフトする．2) シフトされたデータのまま命令を実行する．3) 計算結果を元の正しい位置までシフトする．実験により，マイクロプロセッサのデータパスにおける消費エネルギーを約 3.7%削減出来ることを示した．また，マイクロプロセッサ全体の消費エネルギーは約 1.5%削減出来た．

キーワード 低消費エネルギー，マイクロプロセッサ，組込みシステム，ビット幅，コンパイラ最適化技術

A Software-level Energy Reduction Technique for Embedded Microprocessor Exploiting Narrow Bitwidth Operations

Seiichiro YAMAGUCHI†, Masanori MUROYAMA††,
Tohru ISHIHARA††, and Hiroto YASUURA†††

† Graduate School of Information Science and Electrical Engineering, Kyushu University
6-1 Kasuga-koen, Kasuga, Fukuoka, 816-8580 Japan
†† System LSI Research Center, Kyushu University
3-8-33 Momochihama, Sawara, Fukuoka, Fukuoka, 814-0001 Japan
††† Graduate School of Information Science and Electrical Engineering, Kyushu University
6-1 Kasuga-koen, Kasuga, Fukuoka, 816-8580 Japan
E-mail: †seiichiro@c.csce.kyushu-u.ac.jp, ††{muroyama,ishihara}@slrc.kyushu-u.ac.jp,
†††yasuura@c.csce.kyushu-u.ac.jp

Abstract This paper proposes a software-level energy reduction technique for microprocessor-based embedded systems. A basic idea is to reduce switching activities in sign extension bits of instruction operands through shifting the operands. Our technique consists of following three steps: 1) shift operands from LSB side toward MSB side by optimal shift amount, 2) execute instructions with the shifted operands, and 3) shift back computational results to original positions. Experimental results show about 3.7% energy reduction of datapath, and about 1.5% energy reduction of overall microprocessor.

Key words Low energy, microprocessor, embedded system, bitwidth, compiler optimization technique

1. はじめに

半導体の微細化に伴いハードウェア設計は複雑化し，ハードウェアの設計コスト，検証コスト，テストコストおよびマスクコストが急激に増大している．アプリケーションごとに専用ハードウェアを新規設計することはコストの観点から見合わない．また，携帯情報機器やデジタル情報家電をはじめとする組込みシステムは製品のライフサイクルが短く，市場競争が激しい．このような状況下で収益を上げるためには，付加価値の高い新製品を低コストかつ短 TAT（Turn Around Time）で開発することが重要である．多品種少量生産である組込みシステムを低コストかつ短 TAT で実現する方法として現在多く利用されているのが，マイクロプロセッサベースの組込みシステムである．組込みシステムの設計環境において，ハードウェアの自由度はしばしば非常に限られている．一方で，ソフトウェアの自由度は高い．近年，柔軟に対応出来るソフトウェアベースのアプローチに対する期待が高まっている．

バッテリー駆動である携帯情報機器では，製品の付加価値を高めるためにバッテリー持続時間延長や発熱抑制といった要求がある．従って，LSI の消費エネルギーを削減することが重要となる．本稿では，マイクロプロセッサベース組込みシステムを対象とし，ソフトウェアレベルでアプリケーションプログラムを変更することにより消費エネルギーを削減する手法を提案する．マイクロプロセッサシステムでは数の表現形式として，算術演算を実装しやすい 2 の補数表現を用いている場合が多い．2 の補数表現の問題点の一つは符号拡張である．データパス幅が 32 ビットや 64 ビットといったマイクロプロセッサでは，実行される命令のオペランドのデータが小さい値で正負が交互に出現する場合，データパスにおいて符号拡張ビット部の信号遷移による無駄なエネルギーが消費される．我々はこの点に着目して消費エネルギー削減手法を提案している [1] [2]．本稿では，提案手法を *shift operation insertion* 手法と呼ぶ．

本稿の構成は以下の通りである．第 2 章では，まず準備として消費エネルギーモデルを示し，オペランドのビット幅を考慮する際に用いる有効ビット幅という用語について説明する．また，オペランドのビット幅を考慮した消費エネルギー削減手法の既存研究について紹介する．第 3 章では，まず我々が取り組む消費エネルギー最小化問題を定義し，簡単な例を示しながら *shift operation insertion* 手法のアプローチを説明する．次に，手法の最も重要な部分である最適シフト量決定問題を定式化する．第 4 章では，*shift operation insertion* 手法の有効性を示すために行った評価実験について述べる．最後に第 5 章で，本稿をまとめ今後の課題について述べる．

2. 準　　備

2.1 消費エネルギーモデル

ソフトウェア自体はエネルギーを消費しない．しかし，ソフトウェアがハードウェア上で実行される際に，ハードウェアがエネルギーを消費する原理を知る必要がある．現在多くのマイクロプロセッサは CMOS（Complementary Metal Oxide Semiconductor）ディジタル回路によって実現されている．従って，CMOS ディジタル回路における消費エネルギーモデルを示す．CMOS ディジタル回路の消費エネルギー E_{total} は式 (1) で表される [3]．

$$E_{total} = E_{leak} + E_{sw} + E_{sc} \tag{1}$$

ここで，E_{leak} はリーク電流によるリーク消費エネルギー，E_{sw} は負荷容量の充放電によるスイッチング消費エネルギー，および E_{sc} は貫通電流によるショートサーキット消費エネルギーである．*Shift operation insertion* 手法が削減するエネルギーは CMOS ディジタル回路中の論理ゲート値が遷移する際に消費されるエネルギーであり，ダイナミック消費エネルギーといい，E_{sw} と E_{sc} の和である．ショートサーキット消費エネルギーの時間微分であるショートサーキット消費電力は，全体の消費電力に対して 10%程度であると報告されている [4]．従って，ここでは E_{sw} を考える．E_{sw} は式 (2) で表される．

$$E_{sw} = \frac{1}{2} \cdot V_{dd}^2 \cdot \sum_{g=1}^{N_G} Swit_g \cdot C_{Lg} \tag{2}$$

ここで，V_{dd} は電源電圧，N_G は総論理ゲート数，$Swit_g$ は論理ゲート g の信号遷移回数，および C_{Lg} は論理ゲート g の出力負荷容量である．

式 (2) より，V_{dd}，N_G，$Swit_g$，および C_{Lg} のパラメータ値を削減することにより E_{sw} を削減出来ることがわかる．特殊なハードウェア機構を必要とせず，ソフトウェアレベルから容易に削減可能なパラメータは $Swit_g$ のみである．本稿では $Swit_g$ を削減することでダイナミック消費エネルギーを削減する．

2.2 有効ビット幅

本稿ではオペランドのビット幅を考慮する際に，アプリケーションプログラム中に存在する変数の有効ビット幅情報を用いる．本稿における有効ビット幅の定義を以下に示す．

[定義 1]　有効ビット幅

変数に代入され得る値の最大値，および最小値を表すことが出来る最小のビット幅のことを有効ビット幅という [5]．

unsigned 整数である変数 x に代入され得る値の最大値が d_{max} の時，x の有効ビット幅 $EB(x)$ は，式 (3) となる．

$$EB(x) = \lceil \log_2(d_{max} + 1) \rceil \tag{3}$$

また，signed 整数である変数 x' に代入され得る値の最大値が d'_{max}，最小値が d'_{min} の時，x' の有効ビット幅 $EB(x')$ は式 (4) となる．

$$EB(x') = \lceil \log_2 \mathrm{MAX}(|d'_{max}| + 1, |d'_{min}|) \rceil + 1 \tag{4}$$

ここで，$\mathrm{MAX}(a, b, \cdots)$ は引数の最大値を返す関数である．

図 1 のように，signed 整数である変数 x の値域が [-1000, 1000] であるとする．この時変数 x の有効ビット幅 $EB(x)$ は式 (4) より，

$$\begin{aligned} EB(x) &= \lceil \log_2 \mathrm{MAX}(|1000| + 1, |-1000|) \rceil + 1 \\ &= 11 \end{aligned} \tag{5}$$

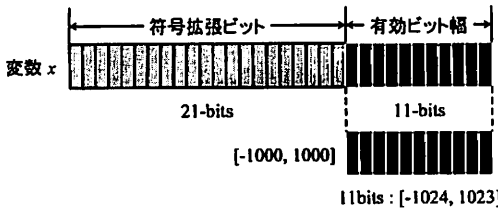

図 1 有効ビット幅の例
Fig. 1 An example of effective bitwidth.

である．

アプリケーションプログラム中の各変数の有効ビット幅を解析する手法はいくつか提案されている [6] [7] [8]．アプリケーションプログラム中の演算命令の種類や利用される変数などをスタティックに解析し有効ビット幅を求める手法と，サンプルデータを用いてシミュレーションし，その際に利用された変数の値域から有効ビット幅を求めるダイナミックな手法がある．*Shift operation insertion* 手法では，スタティックな手法を用いて変数の有効ビット幅解析を行う．

2.3 既存研究

本節では，オペランドのビット幅を考慮して符号拡張ビット部の信号遷移を削減する既存の消費エネルギー削減手法をいくつか紹介する．既存手法は全てハードウェアレベルでの手法である．ソフトウェアレベルでオペランドのビット幅を考慮して符号拡張ビット部の信号遷移を削減する消費エネルギー削減手法は，筆者の知る限りこれまでに提案されていない．

Sign-magnitude 手法は符号を表すビットを 1 ビット用意し，数を絶対値表現を用いて表現する手法である [9]．絶対値表現を用いることにより，小さい値の場合も値の正負に関係なく上位ビットは常に "0" である．*Significance compression* 手法は 2 または 3 ビットの付加ビットを用意し，データの有効な部分をバイト単位で表現する手法である [10]．*Reduced two's-complement* 手法は数の絶対値に応じて動的に数を表現する手法である [11]．オペランドのビット幅を動的に考慮した低電力加算手法も提案されている [12]．バスのコーディングに関する手法も存在する [13], [14]．SPECint95 ベンチマークセットにおいて，整数演算のうち半数以上が 16 ビットもしくはそれ以下の演算精度しか求められていない [15]．文献 [16] では，この事実を利用した *value-based clock gating* 手法を提案している．我々もこの事実を背景として *shift operation insertion* 手法を提案する．

3. *Shift operation insertion* 手法

3.1 消費エネルギー最小化問題

消費エネルギー最小化問題を定義する．*Shift operation insertion* 手法はここで定義する問題を解く手法である．

[定義 2] 記法

全ての自然数の集合を $\mathbb{N}$ と表記する．また，全ての非負実数の集合を $\mathbb{R}_+$ と表記する．

[定義 3] 命令

マイクロプロセッサの命令セット $IS = \{INST_1, INST_2, \cdots, INST_{N_S}\}$ $(N_S \in \mathbb{N})$ が与えられた時，アセンブリソースコード内の命令 $inst_i$ は，必ず $inst_i \in IS$ である．また $inst_i$ は，その格納されているメモリアドレスによって区別される．

[定義 4] 基本ブロック

分岐命令が存在しない連続した N_I $(N_I \in \mathbb{N})$ 個の命令で構成されているアセンブリソースコード内の基本ブロック P は命令を格納されているメモリアドレスの小さい順に並べて次のように表記する．$P = (inst_1, inst_2, \cdots, inst_{N_I})$.

[定義 5] 最小演算ビット幅

$inst_i$ の最小演算ビット幅 B_i $(B_i \in \mathbb{N})$ とは，$inst_i$ の全てのオペランドの有効ビット幅のうち，最大の有効ビット幅のことである．

[定義 6] 演算ビット幅

マイクロプロセッサのデータパス幅が W $(W \in \mathbb{N})$ ビットであるとき，$inst_i$ の演算ビット幅 w_i は，$w_i \in \{B_i, B_i+1, \cdots, W\text{-}1, W\}$ である．このとき，$inst_i$ の全てのオペランドは $(W\text{-}w_i)$ ビット上位にシフトされており，下位 $(W\text{-}w_i)$ ビットは必ず "0" である．

[定義 7] 回路モジュール

マイクロプロセッサのデータパス上には N_M $(N_M \in \mathbb{N})$ 個の回路モジュールが存在し，その集合を CM と表記する．$CM = \{cm_1, \cdots, cm_{N_M}\}$.

[定義 8] ダイナミック消費エネルギー関数

データパス上の回路モジュール cm_m $(cm_m \in CM)$ 上で $inst_i$ が実行される時のダイナミック消費エネルギーは，$inst_i$ の演算ビット幅 w_i，$inst_i$ が実行される直前に cm_m 上で実行された命令 $inst_{i'}$ の演算ビット幅 $w_{i'}$，および cm_m によって決定される．ダイナミック消費エネルギー関数 e: $CM \times \{1, \cdots, W\} \times \{1, \cdots, W\} \to \mathbb{R}_+$.

[問題 1] 消費エネルギー最小化問題

インスタンス：基本ブロック $P = (inst_1, inst_2, \cdots, inst_{N_I})$，サンプルデータ D，およびダイナミック消費エネルギー関数 e.

タスク：D に対して P と同じ結果を出力する $P' = (inst_{1'}, inst_{2'}, \cdots, inst_{N_{I'}})$ $(N_I = N_{I'}$ とは限らない) のうち，マイクロプロセッサのデータパスにおけるダイナミック消費エネルギーが最小となるものを求める．

3.2 アプローチ

Shift operation insertion 手法はシフト演算命令を挿入し，アプリケーションプログラムの基本ブロックの前でデータを上位へシフトし，基本ブロックの後で結果を下位へシフトすることで符号拡張ビット部の信号遷移を削減し，消費エネルギーを削減する．*Shift operation insertion* 手法は次の 3 つのステップから構成される．1) オペランドのデータを LSB 側から MSB 側へ最適なシフト量だけシフトする．2) シフトされたデータのまま命令を実行する．3) 計算結果を元の正しい位置までシフトする．*Shift operation insertion* 手法はソフトウェアレベルでアプリケーションプログラムの変更を行うため，既存のマイクロプロセッサシステムにも適用可能である．

図 2 は，2 つの入力オペランド，$x(t)$ および $y(t)$ が 2 の補数表現で表現されており，直前の入力オペランドが $x(t-1)$

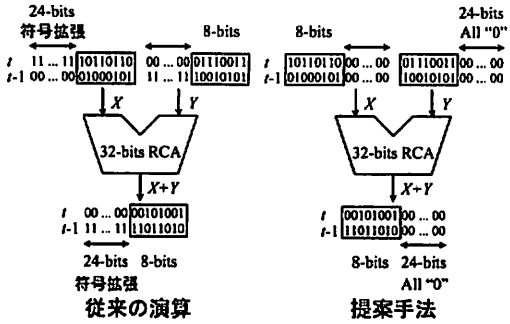

図 2　32 ビット RCA を用いた例

Fig. 2　Motivational examples by using 32-bits RCA.

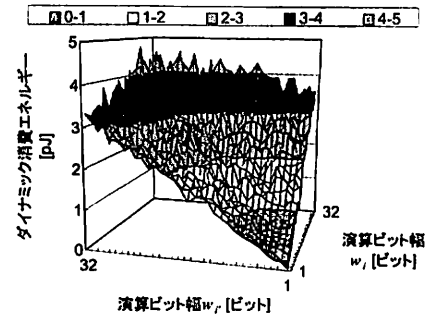

図 3　32 ビット RCA のダイナミック消費エネルギー

Fig. 3　Energy estimation results of 32-bits RCA.

および $y(t-1)$ である時に RCA（Ripple Carry Adder）上で加算が実行される様子を表している．ダイナミック消費エネルギーは 2 つの入力オペランドが $x(t-1)$ および $y(t-1)$ から $x(t)$ および $y(t)$ に変化する際の信号遷移に起因している．図 2 の例では，提案手法における演算ビット幅は時刻 t，および $t-1$ ともに 8 ビットであり，有効ビット部分を四角で囲んでいる．従来の演算では上位 24 ビットは符号拡張ビットである．一方，*shift operation insertion* 手法では下位 24 ビットに全て "0" が挿入されている．これはオペランドを LSB 側から MSB 側へ最適なシフト量だけシフトされているからである．図 2 に挙げた例におけるダイナミック消費エネルギーを見積もった．使用した EDA（Electrical Design Automation）ツールは Cadence 社の NC-Verilog，および Synopsys 社の Power Compiler である．使用したプロセステクノロジは HITACHI CMOS 0.18μm である．電源電圧は 2.5V，RCA へのクロック周波数は 10MHz とした．ダイナミック消費エネルギーは従来の演算では 6.13[pJ]，*shift operation insertion* 手法を適用した場合では 1.2[pJ] であった．ただしこの見積もり結果には，シフトに関するオーバーヘッドは含まれていない．

図 3 に 32 ビット RCA の時刻 t における演算ビット幅 w_i，および時刻 $t-1$ における演算ビット幅 $w_{i'}$ をそれぞれ 1 ビットから 32 ビットまで変化させた際のダイナミック消費エネルギーの見積もり結果を示す．有効ビット部分にはランダムデータを与えた．エネルギー見積もり環境は図 2 の際と同じである．図 3 より，RCA のダイナミック消費エネルギーは演算ビット幅の大きい方に比例して増加すると言える．本稿では，ダイナミック消費エネルギー関数 e は，データパス上の全ての回路モジュールにおいて演算ビット幅の大きい方に比例すると仮定する．従って，出来る限り演算ビット幅は小さい方がよいことになる．

シフト演算のオーバーヘッドに関する大きな問題点が二つある．一つは，演算ビット幅が小さい演算が常に実行されるとは限らないという点である．言い換えれば，全てのオペランドをシフト出来ない可能性がある．もう一つは，各命令の演算ビット幅は異なるという点である．オーバーヘッドを考えれば，全命令の前後にシフト演算を挿入することは出来ない．従って *shift operation insertion* 手法では，各命令の演算ビット幅を考慮して，基本ブロックの入出力変数に対してそれぞれ一度だけシフト演算を実行する．また，ダイナミック消費エネルギーは演算ビット幅に依存しているため，ダイナミック消費エネルギーが最小となるように変数の最適なシフト量を決定する必要がある．次節では *shift operation insertion* 手法の最も重要な部分である最適シフト量決定問題を定式化する．

3.3　最適シフト量決定問題

決定変数

$v_{cm_m,w_i,w_{i'}}$：回路モジュール cm_m 上で命令 $inst_{i'}$ が演算ビット幅 $w_{i'}$ で実行された直後に命令 $inst_i$ が演算ビット幅 w_i で実行される時 1，そうでない時 0 とする．

最小化目的関数

$$\sum_{m=1}^{N_M}\sum_{i'=0}^{N_I-1}\sum_{i=i'+1}^{N_I}\sum_{w_i=B_i}^{W}\sum_{w_{i'}=B_{i'}}^{W} e(cm_m,w_i,w_{i'})\cdot v_{cm_m,w_i,w_{i'}} \tag{6}$$

制約条件

For each i

$$\sum_{w_i=B_i}^{W}\sum_{w_{i'}=B_{i'}}^{W} v_{cm_m,w_i,w_{i'}} = 1 \tag{7}$$

For each i

$$\sum_{w_i=B_i}^{W}\sum_{w_{i'}=B_{i'}}^{W} w_{i'}\cdot v_{cm_m,w_i,w_{i'}} = \sum_{w_{i'}=B_{i'}}^{W}\sum_{w_{i''}=B_{i''}}^{W} w_{i'}\cdot v_{cm_m,w_{i'},w_{i''}} \tag{8}$$

For each i and j

$$\sum_{w_i=B_i}^{W}\sum_{w_{i'}=B_{i'}}^{W} w_i\cdot v_{cm_m,w_i,w_{i'}} = \sum_{z_j=B_j}^{W}\sum_{z_{j'}=B_{j'}}^{W} z_j\cdot v_{cm_m,z_j,z_{j'}} \tag{9}$$

ここで，$inst_j$ は $inst_i$ と少なくとも一つの変数を共有している命令であり，$inst_j$ の数は 1 つとは限らない．

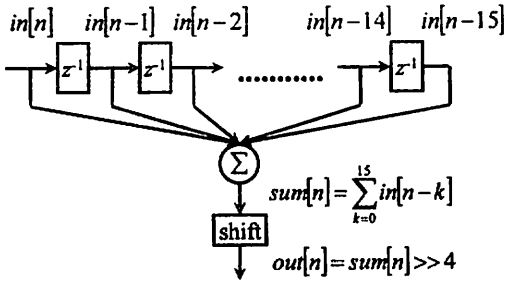

図 4 16 ポイント移動平均フィルタのアーキテクチャ
Fig. 4 An architecture of 16-points moving average filter.

最小化目的関数はマイクロプロセッサのデータパスにおけるダイナミック消費エネルギーである．制約条件 (7) は各命令に対して唯一ひとつの演算ビット幅が割り当てられなければならないということである．制約条件 (8) は $v_{cm_m,w_i,w_{i'}}$ によって決定される $inst_{i'}$ の演算ビット幅 w' と，$v_{cm_m,w_{i'},w_{i''}}$ によって決定される $inst_{i'}$ の演算ビット幅 w' は一致しなければならないということである．制約条件 (9) は同じ変数をオペランドとして持つ全ての命令の演算ビット幅は同じでなければならないことを意味する．定式化した最適シフト量決定問題は次のように定義出来る．

[問題 2] 最適シフト量決定問題

インスタンス：ダイナミック消費エネルギー関数 e，および各命令の最小演算ビット幅 B_i．

タスク：制約条件 (7)，(8)，および (9) を満たし，最小化目的関数 (6) を最小とするような変数 $v_{cm_m,w_i,w_{i'}}$ の集合を求める．

4. 評価実験

4.1 実験方法

Shift operation insertion 手法の有効性を示すために評価実験を行った．ターゲットアプリケーションには 16 ポイント移動平均フィルタを用いて，入力として正弦波を与えた．ターゲットマイクロプロセッサには組込み向け 32 ビット RISC マイクロプロセッサ（ルネサステクノロジ社 M32R-II プロセッサ）を用いた．使用したプロセステクノロジは HITACHI CMOS 0.18μm で，電源電圧は 2.5V，マイクロプロセッサへのクロック周波数は 10MHz とした．図 4 にターゲットアプリケーションである 16 ポイント移動平均フィルタのアーキテクチャを示す．入力変数は $in[n]$ であり，出力変数は $out[n]$ である．

評価実験において，C ソースコードをコンパイルする際は "-O3" オプションを用いた．一方，アセンブリソースコードをアセンブルする際はオプションは使用していない．従って，アセンブラレベルでシフト演算命令の追加を行った場合は必ずシフト演算命令の追加によるオーバーヘッドが存在する．実験はマイクロプロセッサのネットリスト上でオブジェクトコードをシミュレーションし，その際の実行命令数，実行サイクル数，信号遷移回数を求めた．シミュレーションおよび消費エネルギーを見積もる際に使用した EDA ツールは図 2，および図 3 の時と同じで NC-Verilog，および Power Compiler である．

表 1 16 ポイント移動平均フィルタの有効ビット幅解析結果
Table 1 Analysis results of effective bitwidth for 16-points moving average filter.

変数名	有効ビット幅	変数名	有効ビット幅
$in[n]$	16	$in[n-9]$	16
$in[n-1]$	16	$in[n-10]$	16
$in[n-2]$	16	$in[n-11]$	16
$in[n-3]$	16	$in[n-12]$	16
$in[n-4]$	16	$in[n-13]$	16
$in[n-5]$	16	$in[n-14]$	16
$in[n-6]$	16	$in[n-15]$	16
$in[n-7]$	16	$sum[n]$	20
$in[n-8]$	16	$out[n]$	16

表 2 実行命令数と実行サイクル数
Table 2 Number of instructions and cycles executed.

	実行命令数 (増減率 [%])	実行サイクル数 (増減率 [%])
original	204,844	289,246
C-level optimization	204,843 (-0.0)	279,520 (-3.4)
assembler-level optimization	208,940 (2.0)	292,833 (1.2)

4.2 実験結果

表 1 に 16 ポイント移動平均フィルタ中に存在する変数の有効ビット幅を示す．入力変数が short 型で宣言されているため $sum[n]$ を除いた全ての変数の有効ビット幅は 16 ビットである．変数 $sum[n]$ の有効ビット幅は，有効ビット幅が 16 ビットである変数を 16 個加算しているため，20 ビットとなる．16 ポイント移動平均フィルタでは全ての変数が $sum[n]$ に関与しているため，最適シフト量決定問題を解くと制約条件 (9) により，全ての命令の演算ビット幅は 12 ビットとなる．従って，全ての変数の最適シフト量は 12 ビットである．

入力変数 $in[n]$ および出力変数 $out[n]$ を 12 ビットシフトさせる演算を C ソースコードに追加しコンパイルした場合のオブジェクトコード，アセンブリソースコードに追加しアセンブルした場合のオブジェクトコード，およびオリジナルの C ソースコードをコンパイルした場合のオブジェクトコードをマイクロプロセッサのネットリスト上で実行させた際の実行命令数および実行サイクル数を表 2 に示す．C レベルでシフト演算を挿入した場合はオリジナルとはやや異なるオブジェクトコードが出力された．従って，シフト演算を挿入したにも関わらず実行命令数は増えておらず，実行サイクル数も異なる．一方，アセンブラレベルでシフト演算を挿入した場合は挿入したシフト演算命令以外の全ての部分でオリジナルと同じオブジェクトコードが出力された．シフト演算命令によるオーバーヘッドは実行命令数がおよそ 2.0%，実行サイクル数がおよそ 1.2%とわずかである．

16 ポイント移動平均フィルタのオブジェクトコードを実行した際のデータパス，およびマイクロプロセッサのダイナミッ

表 3 ダイナミック消費エネルギー見積もり結果
Table 3 Estimation results of dynamic energy consumptions.

	Datapath		Processor
	モジュール名	Energy [mJ] (増減率 [%])	Energy [J] (増減率 [%])
original	Registers	349.7	2.22
	Buses	136.8	
	ALU	136.2	
	Shifter	15.1	
	Total	**637.9**	
C-level optimization	Registers	319.4 (-8.7)	2.10 (-5.5)
	Buses	116.5 (-14.8)	
	ALU	111.9 (-17.8)	
	Shifter	21.7 (43.6)	
	Total	**569.6 (-10.7)**	
assembler-level optimization	Registers	336.5 (-3.8)	2.18 (-1.5)
	Buses	132.0 (-3.6)	
	ALU	123.3 (-9.5)	
	Shifter	22.3 (47.7)	
	Total	**614.1 (-3.7)**	

ク消費エネルギーを表 3 に示す．C レベルでシフト演算を挿入した場合，データパスのダイナミック消費エネルギーはおよそ 10.7%削減出来た．しかし，この結果は *shift operation insertion* 手法の効果であるとは断言出来ない．一方，アセンブラレベルでシフト演算を挿入した場合，実行サイクル数が増加しているにも関わらず，データパスの消費エネルギーはおよそ 3.7%削減出来た．

5. おわりに

本稿では，マイクロプロセッサベース組込みシステムを対象とし，オペランドのビット幅に着目してソフトウェアレベルでアプリケーションプログラムを変更する消費エネルギー削減手法 (*shift operation insertion* 手法) を提案した．評価実験を行った結果，*shift operation insertion* 手法を適用することで，マイクロプロセッサのデータパスにおける消費エネルギーはおよそ 3.7%削減出来た．

今後の課題は，*shift operation insertion* 手法を自動化するためのアルゴリズムを詳細に定義すること，および他の様々なベンチマークプログラムやマイクロプロセッサにおいても *shift operation insertion* 手法が有効であるか評価することである．

謝辞 本研究の一部は，科学研究費補助金（学術創成研究費 (2)）（課題番号：14GS0218），産学連携研究費福岡県知的クラスタ創成事業「システム LSI 設計開発拠点創成」（(財) 福岡県産業・科学技術振興財団），および，科学技術振興事業団 (JST) の戦略的創造研究推進事業 (CREST)「情報システムの超低消費電力化を目指した技術革新と総合化技術」の支援によるものである．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，株式会社日立製作所，株式会社ルネサステクノロジ，ケイデンス株式会社，および，シノプシス株式会社の協力で行われたものである．

文 献

[1] 山口 誠一朗, 室山 真徳, 樽見 幸祐, and 安浦 寛人, "低演算精度の計算における電力削減手法," *In Proceedings of IPSJ Symposium Series*, vol. 2005, no. 9, pp.267–272, August 2005.

[2] S. Yamaguchi, M. Muroyama, T. Ishihara, and H. Yasuura, "Exploiting narrow bitwidth operations for low power embedded software design," *In Proceedings of the Workshop on Synthesis And System Integration of Mixed Information Technologies*, pp.51–56, April 2006.

[3] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI design – a systems perspective – second edition*, Addison-Wesley, 1993.

[4] K. Nose and T. Sakurai, "Analysis and future trends of short-circuit power," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 9, pp.1023–1030, September 2000.

[5] H. Yamashita, H. Tomiyama, A. Inoue, E. F. Nurprasetyo, T. Okuma and H. Yasuura, "Variable size analysis for datapath width optimization," *In Proceedings of Asia Pacific Conference on Hardware Description Languages*, pp.69–74, July 1998.

[6] H. Yamashita, H. Yasuura, F. N. Eko and Y. Cao, "Variaable size analysis and validation of computation quality," *In Proceedings of IEEE International High Level Design Validation and Test Workshop*, pp.95–100, November 2000.

[7] S. Mahlke, R. Ravindran, M. Schlansker, R. Schreiber and T. Sherwook, "Bitwidth cognizant architecture synthesis of custom hardware accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 11, pp.1355–1371, November 2001.

[8] M. Stephenson, J. Babb and S. Amarasinghe, "Bitwidth analysis with application to silicon compilation," *In Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp.108–120, June 2000.

[9] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *In Proceedings of the IEEE*, vol. 83, no. 4, pp.498–523, April 1995.

[10] R. Canal, A. González and J. E. Smith, "Very low power pipelines using significance compression," *In Proceedings of International Symposium on Microarchitecture*, pp.181–190, December 2000.

[11] Z. Yu, M.-L. Yu, K. Azadet and A. N. Willson., Jr. "The use of reduced two's-complement representation in low-power DSP design," *In Proceedings of IEEE International Symposium on Circuit and Systems*, vol. 1, pp.I-77–I-80, May 2002.

[12] O. T.-C. Chen, R. R.-B. Sheen and S. Wang, "A low-power adder operating on effective dynamic data ranges," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 10, no. 4, pp.435–453, August 2002.

[13] M. Muroyama, A. Hyodo, T. Okuma and H. Yasuura, "A power reduction scheme for data buses by dynamic detection," *IEICE Transactions on Electronics*, vol. E87-C, no. 4, pp.598–605, April 2004.

[14] M. Saneei, A. A.-Kusha and Z. Navabi, "Sign bit reduction encoding for low power applications," *In Proceedings of Design Automation Conference*, pp.214–217, June 2005.

[15] D. Brooks and M. Martonosi, "Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance," *In Proceedings of the 5th International Symposium on High Performance Computer Architecture*, pp.13–22, Jan. 1999.

[16] D. Brooks and M. Martonosi, "Value-Based Clock Gating and Operation Packing: Dynamic Strategies for Improving Processor Power and Performance," ACM Transactions on Computer Systems, Vol. 18, No. 2, pp.89–126, May. 2000.