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

2005－ARC－162 （35）
2005－HPC－101 （35）
2005／3／9

ソフトウェア制御オンチップメモリにおける演算処理を考慮した低消費電力化手法

藤田元信[†,††] 近藤正章[†] 中村宏[†]

今後，プロセスの微細化に伴いリーク電流が増大し，プロセッサ全体の消費電力おけるリーク電流に由来するスタティック消費電力の割合が大きくなると予測されている．我々はこれまでに，ソフトウェア制御可能メモリを対象としたスタティック消費電力削減手法を提案している．オンチップ記憶部分の消費エネルギーに関して，ソフトウェア制御オンチップメモリの利用率と，スタティック消費エネルギーの間にトレードオフがあることが分かっている．しかし，オンチップ記憶部分以外における消費エネルギーも考慮すると，最適なソフトウェア制御オンチップメモリの利用率が変ってくることが考えられる．本稿では，この点に着目し，プロセッサ全体の消費エネルギーを最小化することを目的として，消費エネルギーのモデル化を行った．

A Leakage-Power-Aware Compilation Method for Software Controlled On-chip Memory Architecture

Motonobu Fujita,[†,††] Masaaki Kondo[†] and Hiroshi Nakamura[†]

As semiconductor technology scales down, the static power due to leakage current becomes dominant in the total power dissipation of microprocessors. We have proposed a static power reduction method for Software-Controlled Memory, which cut off the electric power supply of the unused SRAM cells. As for Software-Controlled Memory, there is a trade-off between the execution time and the area which are activated. Good trade-off point may change when energy consumption of other part on microprocessors are considered together. In this paper, we modeled the energy consumption of a microprocessor which has Software-Controlled Memory in order to find minimize the energy consumption of a total chip.

1. はじめに

近年，消費電力はモバイル計算機だけでなく，高性能計算機にとっても設計上の制約となりつつある．マイクロプロセッサの消費電力は，トランジスタのスイッチングに伴うダイナミック消費電力と，リーク電流に由来するスタティック消費電力という二つの要素からなる．従来，ダイナミック消費電力が支配的であったが，今後スタティック消費電力の寄与が増大し無視できない存在になると予想されている．プロセッサ全体の消費エネルギーを最小化するためには，ダイナミック消費電力だけでなくスタティック消費電力も考慮する必要がある．

我々はこれまでに，ソフトウェア制御オンチップメモリ向けに低消費電力手法を提案している[7]．これまでに，オンチップ記憶部分のみの消費エネルギーに関して，ソフトウェア制御オンチップメモリの利用率と，実行時間，スタティック消費エネルギーの間にトレードオフがあることがわかっている．しかし，ロジック部分などオンチップ記憶部分以外における消費エネルギーも考慮すると，最適なソフトウェア制御オンチップメモリ利用率が変ってくると予想される．

本稿では，この点に着目し，プロセッサ全体の消費エネルギーを最小化することを目的として，消費エネルギーのモデル化を行った．

† 東京大学 先端科学技術研究センター
Research Center for Advanced Science and Technology, The University of Tokyo
†† 科学技術振興機構
Japan Science and Technology Agency

2. SCIMA における消費エネルギー削減

2.1 SCIMA の概要

SCIMA は，チップ上に従来のキャッシュに加えソフトウエア制御可能なメモリ *SCM (Software Controlled on-chip Memory)* を搭載する (**図 1**)．SCM は論理アドレス空間の一部の連続した領域を占めており，キャッシュと SCM の間にアドレス空間の包含関係はない．SCM 領域は *page* と呼ばれる単位☆に分割・

☆ アーキテクチャによって決まる固定値であり，数 KB 程度のサイズを想定している．

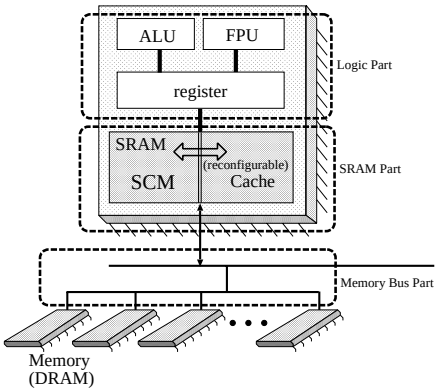

図 1 SCIMA の構成

管理され，メモリアクセス順序保証もこれを単位として行われる．キャッシュと SCM はハードウェアとしての SRAM 自体は共有し，アプリケーションの性質に応じてその容量比を動的に変更させることも可能である．

2.2 ダイナミック消費エネルギーの削減

従来のキャッシュはハードウエア制御により自動的にデータ配置，置き換えが行われるのに対し，SCM では，ソフトウエアから明示的にデータ配置，置き換えを行う．この特徴から，SCIMA では，プロセッサと主記憶間のデータ転送を最小限に抑えることが可能になる．プロセッサの消費電力について考えた場合，プロセッサと主記憶間のデータ転送に伴う消費電力は少なくない．したがって，プロセッサと主記憶間のデータ転送を削減することは，プロセッサのダイナミック消費電力の削減に有効である．

2.3 スタティック消費エネルギーの削減

これまでに，キャッシュを対象として，スタティック消費エネルギーを削減するための回路手法が数多く提案されている．SCM とキャッシュは，ハードウェアとしての SRAM 自体は共有するため，キャッシュ向けに提案されている回路手法をそのまま，あるいはわずかな改変により SCIMA にも適用できるものと考えられる．また，SCIMA では，SCM と主記憶間のデータ転送をソフトウェアで行うため，必要なデータを保持しているセルを特定し，不必要なデータを保持しているセルのみを選択的に Sleep モードに遷移することが可能である．我々はこの点に着目し，SCM のスタティック消費電力削減のためには，電源をオフにしたセルのデータを失う代わりに高いリーク電流削減率を実現する回路手法 Gated-Vdd が最適であると判断し，Gated-Vdd に基づくスタティック消費電力削減手法を提案している[1)]．

本稿においても，SCM 部分には Gated-Vdd を利用したスタティック消費電力削減手法を適用する．

2.4 ダイナミック消費エネルギーとスタティック消費エネルギーのトレードオフ

ダイナミック消費エネルギーは，スイッチングに由来する消費エネルギーであり，演算処理やデータ転送の量に依存する．したがって，ダイナミック消費エネルギーの削減だけを目的とする場合は SCM をできるだけ多く使い，SCM と主記憶間のデータ転送を削減することが有効であると考えられる．

スタティック消費エネルギーは，スイッチングとは無関係に流れるリーク電流に由来する消費エネルギーであり，実行時間とスタティック消費電力に依存する．ここで，先に述べた Gated-Vdd によるスタティック消費電力削減を考慮すると，SCM を多く使えば実行時間が短縮される分スタティック消費エネルギーは減少するが，一方で，SCM 上で電源をオフにできる領域が縮小するため，スタティック消費電力が増加してしまう．つまり，実行時間とスタティック消費電力の間にはトレードオフ関係があると言える．

今後，プロセッサの消費電力のうち，スタティック消費電力が占める割合が増加していくことを考えると，スタティック消費エネルギーにおけるトレードオフが，プロセッサ全体の消費エネルギーにも継承される可能性がある．したがって，プロセッサ全体の消費エネルギーを最小化するためには，両者のバランスを考慮したうえで，最適な SCM 利用率を決定する必要がある．次章では，両者を考慮に入れたプロセッサ全体の消費エネルギーのモデルについて述べる．

3. プロセッサの消費エネルギーのモデル化

3.1 消費エネルギーのモデル

我々は，プロセッサ全体の消費エネルギー E_{all} を，式 1 のようにモデル化した．まず．$E_{SRAM}^{dynamic}$ は SRAM アレイのダイナミック消費エネルギーを表す．この項は，SCM 上にあるデータの読み書き，主記憶から SCM にデータが転送されてきた際に消費されるダイナミック消費エネルギーを含む．E_{SRAM}^{static} は，SRAM アレイのスタティック消費エネルギーを表す．$E_{bus}^{dynamic}$ は，メモリバスのダイナミック消費エネルギーを表す．この項は主記憶から SCM 上にデータを転送する際に生じるバスのスイッチングによって生じる消費エネルギーからなる．$E_{logic}^{dynamic}$ は，ロジック部分☆のダイナミック消費エネルギーを表す．E_{logic}^{static} は，ロジック部分のスタティック消費エネルギーを表す．

$$\begin{aligned} E_{all} = & E_{SRAM}^{dynamic} + E_{SRAM}^{static} \\ & + E_{bus}^{dynamic} \\ & + E_{logic}^{dynamic} + E_{logic}^{static} \end{aligned} \quad (1)$$

式 1 の各項は，式 2〜 式 6 のように詳細化される．まず，$E_{sram}^{dynamic}$ について，$Num_{read/write}$ は

☆ ALU，分岐予測器などを含む

SRAM アレイに対する読み書きの回数を，$Tr(NB)$ は SCM と主記憶間のトラフィック（ワード単位）を，$avg(P_{SRAM}^{dynamic})$ は，SRAM アクセスの際のダイナミック消費電力の平均を表す．次に，E_{sram}^{static} について，$T(NB)$ は実行サイクル数を，P_{SRAM}^{static} は SRAM 部分のスタティック消費電力を表す．ここで，Gated-Vdd によるスタティック消費電力削減の効果を反映するため，SCM 領域の利用率として Activation ratio： Ac を導入する．Ac は，SCM 全体の容量に対する利用されている容量の比であり，$0 < Ac < 1$ の関係を満たす．また，Ac はブロックサイズ NB によって変化することから，以降 $Ac(NB)$ と表記する．$E_{bus}^{dynamic}$ について，$avg(P_{bus}^{dynamic})$ はメモリバス経由で 1 ワードのデータを転送する際の消費電力の平均を表す．$E_{logic}^{dynamic}$ は，ロジック部分におけるダイナミック消費エネルギーを表す．今回は，プログラム中の全命令数 Num_{inst}，ロジック部分のダイナミック消費電力の平均 $avg(P_{Logic}^{dynamic})$ を用いてモデル化することとした．E_{logic}^{static} についても同様に，実行サイクル数 T および P_{Logic}^{static} はロジック部分のスタティック消費電力の積としてモデル化を行った．

$$E_{sram}^{dynamic} = ((Num_{SRAM}^{read/write}) + Tr(NB)) \times avg(P_{SRAM}^{dynamic}) \quad (2)$$

$$E_{sram}^{static} = T(NB) \times P_{SRAM}^{static} \times Ac(NB) \quad (3)$$

$$E_{bus}^{dynamic} = Tr(NB) \times avg(P_{bus}^{dynamic}) \quad (4)$$

$$E_{logic}^{dynamic} = Num_{inst} \times avg(P_{Logic}^{dynamic}) \quad (5)$$

$$E_{logic}^{static} = T(NB) \times P_{Logic}^{static} \quad (6)$$

ここで，$Tr(NB)$, $T(NB)$, $Ac(NB)$ は，ブロックサイズ NB の関数として表される．したがって，プロセッサの消費エネルギー E_{all} は NB の関数となる．以降，この NB の値を通じて SCM の利用率を変化させることで，プロセッサの消費エネルギーの最小化を図る．実行時間 $T(NB)$ やオフチップトラフィック $Tr(NB)$ における，NB の係数はアプリケーションに依存する．したがって，$Tr(NB)$, $T(NB)$ は対象アプリケーションごとにモデル化を行う必要がある．

3.2 アプリケーションに依存した係数の決定

これまでに，SCIMA では SCM の利用方法は，扱うデータの再利用性の有無により，主に二種類に分類されることが分かっている．そこで，本節では再利用性のあるデータを扱うプログラムの単純な例として行列積を取り上げ，また，再利用性のないデータを扱うプログラムの例としてベクトル内積を取り上げ，アプリケーションごとに実行時間 T および主記憶トラフィック Tr，SCM の利用率 Ac を詳細化する例を示す．

行 列 積

行列積では，$Tr(NB)$, $Ac(NB)$, $T(NB)$ は以下のとおり詳細化される．

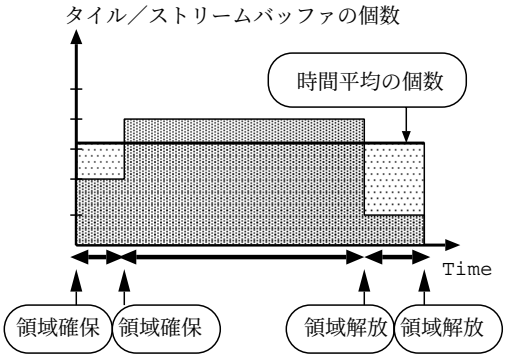

図 2 時間平均の個数.

```
!$scm begin(x, blk, blk, 0, 0)
!$scm begin(y, blk, blk, 0, 0)
!$scm begin(z, blk, blk, 0, 0)
    do ii = 1,NSIZE,NB
      do jj = 1,NSIZE,NB
!$scm load(z, ii, jj, MIN(blk,N-ii), MIN(blk,N-jj))
        do kk = 1,NSIZE,NB
!$scm load(y, kk, jj, MIN(blk,N-kk), MIN(blk,N-jj))
!$scm load(x, ii, kk, MIN(blk,N-ii), MIN(blk,N-kk))
          do i = ii,ii+blk-1,1
            do j = jj,jj+blk-1,1
              do k = kk,kk+blk-1,1
                z(i,j)=z(i,j)+x(I,J)*y(J,K)
              enddo
            enddo
          enddo
        enddo
!$scm store(z)
      enddo
    enddo
!$scm end(z)
!$scm end(y)
!$scm end(x)
```

SCM領域の確保
該当領域電源 オン

SCM領域の解放
該当領域電源オフ

図 3 行列積.

$$Tr(NB) = Num_{trans}(NB) \times TileSize(NB) \times avg(N_{tile}) \quad (7)$$

$$Ac(NB) = \frac{TileSize(NB) \times avg(N_{tile})}{SCM\_SIZE} \quad (8)$$

$$T(NB) = Num_{trans}(NB) \times (T_{latency}^{tile} + T_{trans}^{tile} + T_{comp}^{tile}) \quad (9)$$

ここで，$Num_{trans}(NB)$ は SCM と主記憶間のデータ転送回数，$TileSize(NB)$ はブロッキングを行った際の小行列のサイズ，SCM_SIZE は SCM の全容量，$T_{latency}^{tile}$ は小行列一個分の主記憶アクセスレイテンシ，T_{trans}^{tile} は小行列一個分の転送サイクル数，T_{comp}^{tile} は小行列一個分の演算サイクル数をそれぞれ表す．

ここで，SCM 上に載る小行列の個数は**図 2** の模式図で示すように，各小行列のための SCM 領域がいつ確保され，いつ解放されるかによって変化すると考えられるため，時間平均し $avg(N_{tile})$ で表す．

例として，二次元のブロッキングを行い，形は正方形しか許さないという制約を設けた場合 (**図 4**)，$Num_{trans}(NB)$, $TileSize(NB)$, $avg(N_{tile})$ は以下のように詳細化される．

```
!$scm begin(a,PAGE,0)
!$scm begin(b,PAGE,0)
!$scm begin(c,PAGE,0)
      do ii = 1,N,NB
!$scm load(a,ii,MIN(INT(PAGE),N-ii+1))
!$scm load(b,ii,MIN(INT(PAGE),N-ii+1))
!$scm load(c,ii,0)
         do i=ii, MIN(ii+NB-1, N),1
            c(i)=a(i)*b(i)
         enddo
!$scm store(c)
      enddo
!$scm end(c)
!$scm end(b)
!$scm end(a)
```

SCM領域の確保
該当領域電源 オン

SCM領域の解放
該当領域電源オフ

図 4 ベクトル内積.

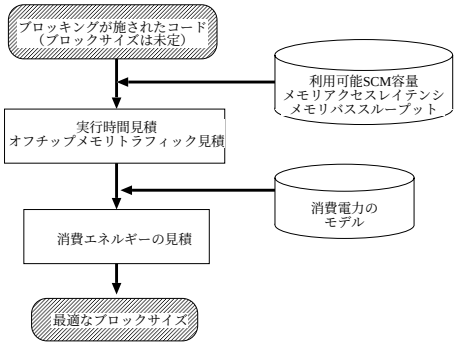

図 5 最適化処理の流れ

$$Num_{tran}(NB) = \left\lceil \frac{NSIZE}{NB} \right\rceil^3 \quad (10)$$

$$TileSize(NB) = NB \times NB \quad (11)$$

$$avg(N_{tile}) = 3 \quad (12)$$

$NSIZE$ はもとの行列の一辺の要素数を示す.

再利用性のないプログラム：ベクトル内積

ベクトル内積では，$Tr(NB)$, $Ac(NB)$, $T(NB)$ は以下のように詳細化される.

$$Tr(NB) = \frac{L_{stream}}{NB} \times NB = L_{stream} \quad (13)$$

$$Ac(NB) = 2 \times \frac{BufSize(NB)}{SCM\_SIZE} \quad (14)$$

$$T(NB) = \frac{L_{stream}}{NB} \times (T_{latency} + \frac{BufSize(NB)}{B_{membus}} + T_{comp}^{buf}) \quad (15)$$

L_{stream}, $T_{latency}$, B_{membus}, $BufSize(NB)$, T_{comp}^{buf} はぞれぞれ，ストリームの長さ (要素数)，主記憶レイテンシ，メモリバスバンド幅，バッファのサイズ，バッファ一つ分の演算時間を表す.

3.3 テクノロジに依存した係数の決定

プロセッサ全体の消費エネルギーを見積るためには，SRAM 部分，バス，ロジック部分の消費電力を求める必要がある．ダイナミック消費電力については，プロセッサ上の各部分が占める割合がこれまでにも紹介されている[2]．また，Wattch[3] を利用すれば，様々な構成のプロセッサについて，プロセッサの各部分の消費電力を計算することができる．一方，スタティック消費電力については，プロセッサ全体にわたって各部分の割合を示したものはない．そこで，テクノロジの違いに対応するため，ダイナミック消費電力に対するスタティック消費電力の割合として **Leakage Factor**：Lf を導入する．ダイナミック消費電力を $P_{dynamic}$，スタティック消費電力を P_{static} とすると，以下の関係が成り立つ.

$$P_{static} = Lf \times P_{dynamic}$$

ただし，$0 < Lf < 1$.

テクノロジに依存する各値の導出方法を**表 1** にまとめた.

表 2 Leakage Factor の値

Technology	180nm	130nm	70nm
Leakage Factor	0.05	0.20	0.50

4. コンパイラによる低電力化

提案手法では，まず、コンパイラは，SCM と主記憶間のデータ転送を行うようブロッキングが施されているがデータ転送粒度が決定されていないプログラムを入力として受け取る．コンパイラは，利用可能な SCM サイズの上限，消費エネルギーのモデルをもとに，プロセッサ全体の消費エネルギーを最小化するデータ転送粒度を探索する．求めたデータ転送粒度を，元のプログラムのデータ転送粒度に反映する.

提案手法の利点として，ソフトウェア制御オンチップメモリを対象としていることで，実際にプログラムを繰り返し実行することなく，最適なデータ転送粒度を求めることが可能，という点が挙げられる．また，消費エネルギーのモデルを変更することで，テクノロジの違いを吸収することが可能である.

5. 評　　価

5.1 評 価 方 法

今回は初期評価として，行列積とベクトル内積について，横軸にブロックサイズ NB をとり，本稿のモデルにより実行サイクル数，消費エネルギーを得る．また，Leakage Factor は，文献[?] を参考に，**表 2** のように決め，SRAM 部分，ロジック部分によらず値は同じとした.

5.2 評 価 結 果

行列積についての実行時間の見積を**図 6** に示す．ブロックサイズ NB を大きくするにつれて実行サイクル数が削減されていることが分かる．行列積について，消費エネルギーについての見積を**図 7**，**図 8** および**図 9** に示す．消費エネルギーの内訳について，$Edyn\_sram$ は SRAM のダイナミック消費エネルギー，Est_sram は

表 1 消費エネルギー見積に使用する定数値の導出方法

パラメータ	説明	導出方法
$avg(P_{SRAM}^{dynamic})$	SRAM 部分ダイナミック消費電力	ツール (Wattch) を使って求める
P_{SRAM}^{static}	SRAM 部分スタティック消費電力	$P_{SRAM}^{dynamic} \times Lf$
$avg(P_{MemBus}^{dynamic})$	バスダイナミック消費電力	ツール (Wattch) を使って求める
$avg(P_{Logic}^{dynamic})$	ロジック部分ダイナミック消費電力	ツール (Wattch) を使って求める
P_{Logic}^{static}	ロジック部分ダイナミック消費電力	$P_{Logic}^{static} \times Lf$

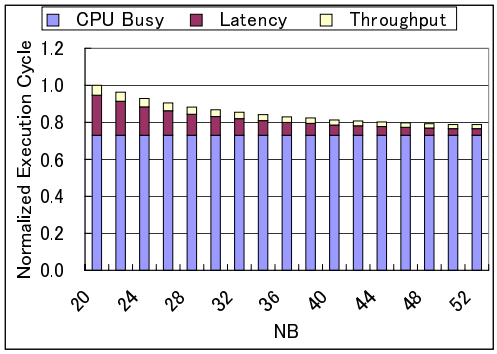

図 6 行列積 性能

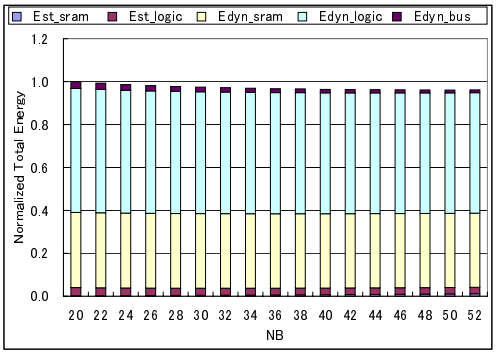

図 7 行列積 消費エネルギー (180nm)

SRAM のスタティック消費エネルギー，$Edyn\_bus$ はバスのダイナミック消費エネルギー，$Edyn\_logic$ はロジック部分のダイナミック消費エネルギー，Est_logic はロジック部分のスタティック消費エネルギーをそれぞれ表す．ここで，図 7〜図 9 におけるダイナミック消費エネルギーに着目すると，ブロックサイズが大きくなるにつれてトラフィックが減少するため，単調な減少傾向を示す．一方，スタティック消費エネルギー部分について，ロジック部分のスタティック消費エネルギーは NB 増加に伴い単純に減少する．一方，SRAM 部分のスタティック消費エネルギーについては，実行時間とアクティブになっている面積に依存する．ブロックサイズ NB が大きくなると，実行時間は短くなる分スタティック消費エネルギーは減少するが，一方で電源をオフにできる領域が縮小するため，スタティック消費電力が増加してしまう．

プロセッサ全体の消費電力において，スタティック消費電力が増加すると，SRAM 部分のスタティック消費エネルギーにおけるトレードオフ関係が，プロセッサ全体の消費エネルギーに継承され，消費エネルギーが最小となるブロックサイズ NB が小さくなる．このことは，今後スタティック消費電力の割合が増加するにつれて，オンチップメモリを節約することが消費エネルギーを抑えるうえで有効になることを示していると考えられる．

次に，ベクトル内積の性能を**図 10** に示す．バッファサイズに対して，性能はデータ転送のサイズが *page* サイズと一致するところで飽和する．これは，一回のデータ転送サイズの上限が *page* で抑えられ，これ以上のバッファサイズを設定してもメモリアクセスレイテンシが短縮されないためである．

消費エネルギーについての見積を**図 11**，**図 12** および**図 13** に示す．性能向上が飽和した NB=512 以上の点においても，ストリームバッファが占める SCM 領域のサイズは増加するため，SRAM 部分のスタティック消費電力は増加する．したがって，プロセッサ全体の消費エネルギーは，バッファサイズが *page* サイズと一致する点を最小として，以降増加する．SCM 領

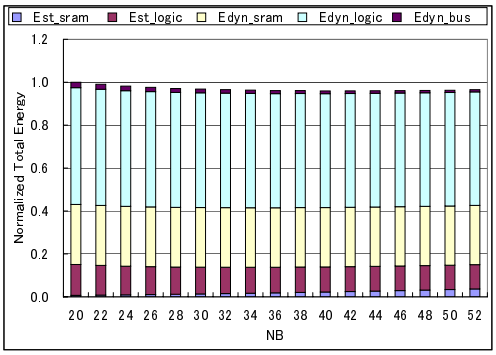

図 8 行列積 消費エネルギー (130nm)

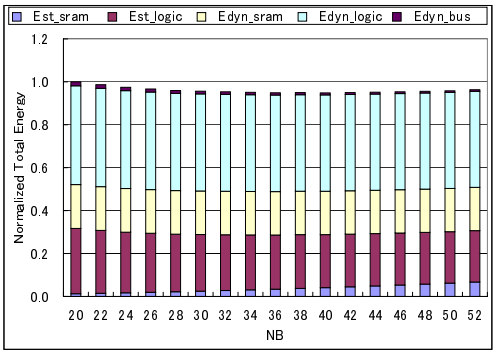

図 9 行列積 消費エネルギー (70nm)

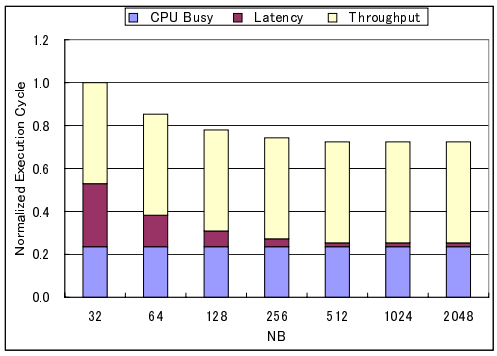

図 10 ベクトル内積 性能

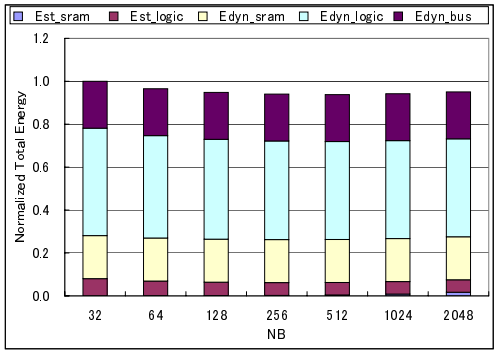

図 11 ベクトル内積 消費エネルギー (180nm)

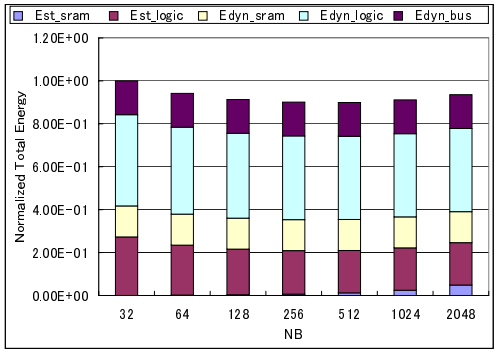

図 12 ベクトル内積 消費エネルギー (130nm)

域のうち，バッファとして利用される面積はわずかであるため，SRAM 部分のスタティック消費エネルギーのトレードオフがダイナミック消費エネルギーの減少により完全に相殺されたと考えられる．

6. ま と め

近年，テクノロジの微細化に伴い，プロセッサの消費エネルギーに占めるスタティック消費エネルギーの割合が増加している．スタティック消費エネルギーは，実行時間とスタティック消費電力に依存しており，両者のトレードオフが，プロセッサ全体の消費エネルギー

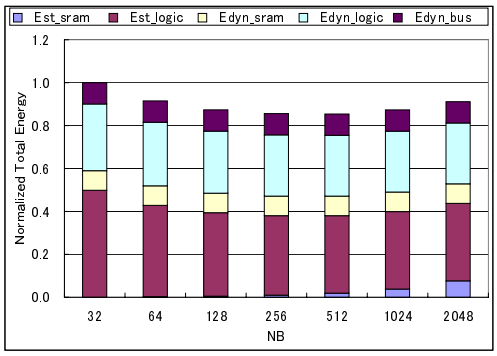

図 13 ベクトル内積 消費エネルギー (70nm)

にも影響を及ぼす場合がある．そこで，本稿では，プロセッサ全体の消費エネルギーの最小化を目的として，プロセッサ全体の消費エネルギーのモデル化を行った．これは，今後ソフトウェア制御オンチップメモリの最適な利用率を求めるコンパイルアルゴリズムを検討する上で有用であると考えられる．評価結果から，スタティック消費電力の割合が増加するにつれ，実行時間の増加を許容しても SCM の利用率を下げスタティック消費電力を削減することがプロセッサ全体の消費エネルギー削減のために有効であることがわかった．

今後の課題として，まず，本稿での見積結果をシミュレーションと比較，検証することが挙げられる．また，モデルの精緻化も課題である．今回は単純なプロセッサを想定したため，プログラムごとのダイナミックな挙動の差異を考慮していない．より複雑なプロセッサでは，ダイナミックな挙動がプログラムの実行時間や消費電力に大きく影響すると考えられる．本手法の適用範囲を広げるためには，消費エネルギーのモデル自体をプロファイルを取り作成するなど，消費エネルギーのモデル自体の作成方法を改良する必要があると考えられる．

謝辞 本研究の一部は，文部科学省科学研究費補助金 (基盤研究 (B) No. 14380136) によるものである．

参 考 文 献

1) 藤田元信, 近藤正章, 中村宏, ”ソフトウェア制御オンチップメモリにおけるスタティック消費電力削減手法 ”, Vol.45, No.SIG11(ACS7), pp. 219-228, 2004.
2) Michael K. et al., ”Power considerations in the design of the Alpha 21264 microprocessor”, Proc. DAC ’98, pp.726-731, 1998.
3) David Brooks et al.,”Wattch: a framework for architectural-level power analysis and optimizations”, Proc. ISCA’00, pp.83-94, 2000.
4) S.Borkar, ”Low power design challenges for the decade”, Proc. ASP-DAC ’01, pp.293-296, 2001. Aug. 1998.