Vol.2013-EMB-30 No.5
2013/9/17

周期実行システムにおける中間データに着目した電力制御手法

重松 拓也[1,a)]　薦田 登志矢[1]　中田 尚[1]　三輪 忍[1]
佐藤 洋平[2]　植木 浩[2]　林越 正紀[2]　清水 徹[2]　中村 宏[1]

概要：現在，様々な分野でスマートセンサシステムが使われている．しかし各ノードに十分な電源を確保することは難しく，電池で動作しなければならない状況で利用されることも多い．このような状況下では電池交換の回数を最小化したいといった要求があり，ノード自体の省電力化が強く求められている．特に，ノード内で高度な処理を行うセンサシステムではマイコンが消費する電力が問題となっている．従来より，マイコンの消費電力を抑えるためにパワーゲーティングと呼ばれる手法が用いられるが，近年ではワーキングメモリをもパワーゲーティングするより深いスリープモードを持つマイコンも登場している．深いスリープを行えば大幅な電力削減効果が得られるが，ワーキングメモリの内容が失われるため，復帰後も必要な中間データは不揮発メモリに退避させる必要がある．また，中間データの退避にも追加エネルギが必要であるため，深いスリープによって得られる消費エネルギの削減分より，この退避エネルギが大きくなる場合には深いスリープを行うべきではない．そこで本稿では，中間データのサイズと保持期間に着目し，マイコンが消費するエネルギを最小化する最適な電源制御とデータ退避方法を導出するアルゴリズムを提案する．

1. はじめに

近年，マイコンとセンサが一体となったスマートセンサシステムが普及しており，センサからの情報をノード内で解析し，センサ情報に基づくリアルタイムな動作が可能となっている．例としては，イメージセンサによる侵入者検知システム，農園の環境モニタリングシステム，などが挙げられる．一方で，これらのシステムは電池で動作する場合が多く，電池交換の手間を考えるとノード自体の省電力化が重要である．

このような高度なスマートセンサシステムでは，ノードが消費するエネルギの中でも特にマイコンが消費するエネルギが多くを占めている．そこで本稿ではノード内のマイコンの省電力化を目指す．

消費電力を抑えるため，最近のマイコンは様々なスリープモードを搭載している．特にメモリをパワーゲーティングするスリープモードを持つマイコンも存在し，大幅な電力削減効果が得られる代わりに，メモリの内容を失う．復帰後に必要となる中間データが存在する場合は，退避用メモリに退避させる必要があり，退避にかかるエネルギも無視することはできない．

本稿では，中間データのサイズと保持期間に着目した電力制御とデータ退避方法を提案し，多項式時間で解を導出するアルゴリズムを提案する．また提案アルゴリズムがある条件下で最適な解を導出することを示す．

2. 背景

スマートセンサシステムのノードの中で，特にマイコンの消費電力が大きな割合を占めている．今後はより高度な処理が求められるようになることが予想されるので，マイコンの低消費電力化は重要な課題である．ここでは本稿で対象とする周期実行システム，マイコンにおけるスリープモード，そして周期実行システムでマイコンのスリープ制御を行う場合に問題となる中間データのそれぞれについて概要を述べる．

2.1 周期実行システム

スマートセンサシステムは**図 1** で示すように，センサ，マイコン，出力機器で構成される．センサは複数の場合もあり，出力機器とは無線機器や警報器などを指し，センサからの周期的な入力をマイコンで処理し，出力機器で出力

[1] 東京大学
The University of Tokyo
[2] ルネサス エレクトロニクス株式会社
Renesas Electronics Corporation
[a)] takuya_shigematsu@ipc.i.u-tokyo.ac.jp

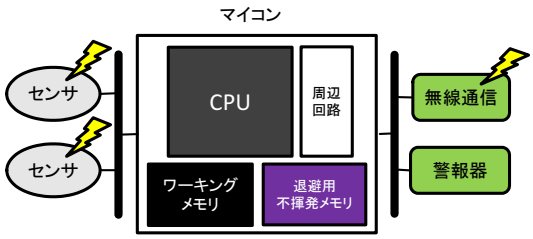

図 1 スマートセンサシステムのノードの構成

する，という動作をする．

スマートセンサシステムは周期実行システムの一つである．周期実行システムでは，マイコンが処理を終え次のセンサ入力を待つ間にアイドル区間が生じる．アイドル区間では，クロックの供給を止めるクロックゲーティングと呼ばれる手法を用いて消費電力を抑える．しかし，半導体プロセスの微細化によりアイドル区間で消費されるスタティック電力が大きな問題となってきた．そこで近年では，電圧を落としスタティック電力を削減するパワーゲーティングと呼ばれる手法が用いられることが増えてきた．

周期実行システムでは中間データと呼ばれるデータが生じることがある．中間データとは，アイドル区間から復帰した後に使用されるため，アイドル区間でも保持しなければならないデータのことである．マイコンのメモリの内容が失われるスリープモードを使用するとき，この中間データの存在が問題になる．

2.2 マイコンの消費電力とスリープモード

マイコンは，図 1 に示すように大きく分けて CPU，ワーキングメモリ，退避用メモリ，周辺回路から構成される．ワーキングメモリは揮発性，退避用メモリは不揮発性である．一般に不揮発性メモリはアクセスエネルギが大きいという特徴がある．

マイコンの各コンポーネントが消費する電力にはダイナミック電力とスタティック電力の 2 種類がある．ダイナミック電力とはトランジスタのスイッチングによって消費される電力のことで，コンポーネントが動作するときには必ず消費される．一方動作しないときは，コンポーネントへのクロック供給を止めるクロックゲーティングと呼ばれる手法を用いることでダイナミック電力の消費を抑えることができる．スタティック電力とは漏れ電流によって消費されるエネルギのことで，コンポーネントが動作しているかどうかにかかわらず消費され続ける．スタティック電力を削減するには，コンポーネントにかかる電圧を落とすパワーゲーティングと呼ばれる手法が用いられる．

従来からマイコンが処理を行っていない間はクロックゲーティングによりダイナミック電力を削減するという制御が用いられてきた．しかし半導体プロセスの微細化により，スタティック電力の方が問題となってきた．そこでスタティック電力の削減を狙い，クロックゲーティングだけでなくパワーゲーティングも用いられるようになった

現在普及している多くのマイコンが，どのコンポーネントを対象としてパワーゲーティングを行うかにより複数段階のスリープモードを持つ．より多くのコンポーネントを対象とした方が大きなスタティック電力削減効果が得られるが，遷移するために生じる遷移オーバヘッドも大きくなる．一般にスタティック電力の小さいスリープモードの方が遷移オーバヘッドは大きく，トレードオフの関係が存在する．そのためアイドル区間の長さ，スタティック電力，遷移オーバヘッドから適切なスリープモードを選択する必要がある．

近年ではより大きな電力削減効果を狙い，ワーキングメモリまでパワーゲーティングするスリープモード（本稿では**深いスリープ**と呼ぶ．一方，ワーキングメモリの内容が保持されるスリープモードを**浅いスリープ**と呼ぶ）を持つマイコンが登場している．ワーキングメモリをパワーゲーティングするとメモリの内容が失われるため，中間データを退避用メモリに退避させる必要がある．

2.3 中間データの退避によるオーバヘッド

中間データを退避させるためにも，退避用メモリの書き込み読み込みエネルギなどが必要である．深いスリープに入ることによって得られるエネルギの削減幅よりも，中間データを退避させるためにかかる追加エネルギの方が大きい場合，中間データを退避させずに浅いスリープに入るべきである．また，深いスリープに入るべきかどうかの判断は，アイドル区間の長さ，スタティック電力，遷移オーバヘッドに加え，中間データのデータサイズと保持期間にも影響される．特に保持期間が複数のアイドル区間にまたがる場合，単純に一つのアイドル区間から深いスリープに入るべきかどうかを判断することはできない．しかし全ての中間データに対して全探索を行い適切なスリープモードを求めるには中間データの数に対して指数オーダーの計算時間がかかり，中間データの数が増えた場合現実時間で最適解を求めることができなくなる．

そこで本稿では，深いスリープに入るべきかどうか，中間データを退避させるべきかどうか，を多項式時間で判断するアルゴリズムを提案する．

3. 中間データに着目した電力制御手法

中間データはデータサイズと保持期間で表される．深いスリープに入ることで削減できるエネルギより，中間データを退避させるエネルギの方が大きい場合，中間データは退避させず浅いスリープに入るべきである．削減されるエネルギはアイドル区間の長さに比例し，中間データの退避に必要なエネルギはデータサイズに比例する．これらの関係を考慮すると，保持期間の長いものほど退避による効果

が大きく，データサイズの大きいものほど退避によるオーバヘッドが大きいといえる．本節ではデータサイズと保持期間を考慮して，アイドル区間でのスリープモードと中間データの退避方法を導出する手法を提案する．

3.1 複数のスリープモードを持つマイコンの電力モデル

マイコンとタスクのパラメータを**表 1** にまとめる．ここで，クロックゲーティングのみ行った場合のスタティック電力を $SP_{cg}[W]$ とする．また，浅いスリープモードでのスタティック電力を $SP_{shallow}[W]$，アクティブ状態からの遷移オーバヘッドを $EOH_{shallow}[J]$ とする．深いスリープモードでのスタティック電力を $SP_{deep}[W]$，アクティブ状態からの遷移オーバヘッドを $EOH_{deep}[J]$ とする．

退避用メモリの書き込みエネルギを $WE[J/byte]$，読み込みエネルギを $RE[J/byte]$ とする．この書き込みエネルギ，読み込みエネルギには，書き込み読み込み命令を実行している間のマイコンのダイナミック電力とスタティック電力も含まれている．また退避用メモリは不揮発性であるため，書き込み読み込み時以外はパワーゲーティングしており消費電力は 0 とする．

次にタスクのパラメータを説明する．本稿では周期実行タスクを対象とする．**図 2** の上段に示すようなタスクグラフが与えられるとする．各ノードはサブタスクを，各エッジはデータの依存関係を表している．タスクグラフによってサブタスクの処理時間，データの依存関係，データサイズが表現される．

図 2 の中段に示すように，タスクグラフのスケジューリングが決定すると，アイドル区間と中間データの存在が決定する．ここで，アイドル区間の ID をタイミングの早いものから順に i $(i = 0, 1, ..., N-1)$ で表し，各アイドル区間の長さを $idleTime[i]$ で与える．

中間データはデータサイズと保持期間で表現される．中間データの ID を j $(j = 0, 1, ..., M-1)$ とし，データサイズを $dataSize[j](j = 0, 1, ..., M-1)$ と表す．中間データ j がまたがるアイドル区間の ID の集合を $set\_id[j]$ と表す．例えば，中間データ 1 であればアイドル区間 0 と 1 にまたがるので，$set\_id[1] = \{0, 1\}$ である．なお，退避用メモリは不揮発性なのでスタティック電力は 0 である．よって保持期間はマイコンの処理時間には影響されず，アイドル区間にのみ依存する．また中間データを退避させる場合，必ず生成タイミングで退避用メモリに書き込み，消費タイミングで読み出すものとする．

以降の議論ではスケジューリング済みのタスクグラフを入力とし，アイドル区間と中間データに論点を絞るため図 2 の下段に示すような表記を用いる．赤で示した双方向矢印はアイドル区間を表しており，青で示した実線は中間データを表している．

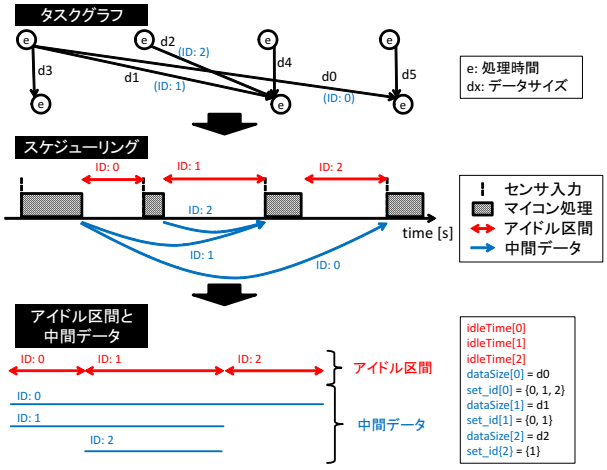

図 2 アイドル区間と中間データの抽出

表 1 パラメータ一覧

パラメータ	説明
SP_{cg}	クロックゲーティング時のマイコンのスタティック電力 [W]
$SP_{shallow}$	浅いスリープ時のマイコンのスタティック電力 [W]
$EOH_{shallow}$	アクティブから浅いスリープに入るための マイコンの遷移オーバヘッド [J]
SP_{deep}	深いスリープ時のマイコンのスタティック電力 [W]
EOH_{deep}	アクティブから深いスリープに入るための マイコンの遷移オーバヘッド [J]
WE	退避用メモリの書き込みエネルギ [J/byte]
RE	退避用メモリの読み込みエネルギ [J/byte]
i	アイドル区間の ID
$idleTime[i]$	アイドル区間 i の長さ [s]
j	中間データの ID
$dataSize[j]$	中間データ j のデータサイズ [byte]
$set\_id[j]$	中間データ j がまたがるアイドル区間の ID の集合

3.2 定式化

解くべき問題を定式化する．アイドル区間の長さ $idleTime[i]$ と，中間データのサイズ $dataSize[j]$，保持期間 $set\_id[j]$ が与えられる．このとき表 1 のハードウェアパラメータを用いて，$costStatic[i]$ と $costData[j]$ を式 (1) と式 (2) によって定義する．

$$costStatic[i] = idleTime[i] \times (SP_{shallow} - SP_{deep}) + (EOH_{shallow} - EOH_{deep}) \quad (1)$$

$$costData[j] = dataSize[j] \times (WE + RE) \quad (2)$$

$costStatic[i]$ は浅いスリープから深いスリープにしたときに，各アイドル区間で削減できるスタティックエネルギと遷移オーバヘッドを表す．また $costData[j]$ は各中間データを退避用メモリに退避させるときに生じるエネルギを表す．

各アイドル区間でのスリープモードを表す $sleepMode[i]$（深いスリープのとき 0，浅いスリープのとき 1）と，中間データの保存先を表す $dataMode[j]$（退避させないとき 0，退避させるとき 1）を変数として最適化問題の形に記述できる．

$$min \sum_i sleepMode[i] \times costStatic[i]$$

$$+\sum_j dataMode[j] \times costData[j] \quad (3)$$

$$s.t. \quad sleepMode[i] = 0 \Rightarrow \forall j \ (i \notin set\_id[j] \ or \ dataMode[j] = 1) \quad (i = 0, 1, ..., N-1) \quad (4)$$

目的関数 (3) の 1 行目は各アイドル区間にかかるコストの総和である．深いスリープ（$sleepMode[i] = 0$）ではコスト（$costStatic[i]$）がかからず，浅いスリープ（$sleepMode[i] = 1$）ではコスト（$costStatic[i]$）がかかる．2 行目は各中間データの退避にかかるコストの総和である．退避させないとき（$dataMode[j] = 0$）はコスト（$costData[j]$）がかからず，退避させるとき（$dataMode[j] = 1$）はコスト（$costData[j]$）がかかる．

制約条件 (4) は，深いスリープにするためにはそのアイドル区間をまたがる中間データを全て退避させなければならないことを意味する．この制約条件を守りながら，全体のコストを最小化する最適化問題である．

以降の議論では，解くべき最適化問題を $\pi(J)$ と表す．ここで J は最適化問題 $\pi(J)$ で対象となる中間データの ID の集合を表す．なお，対象となる中間データの集合がわかれば対象となるアイドル区間の集合は一意に決まるので，最適化問題は中間データの集合を用いて表すことにする．

3.3 提案アルゴリズム

各中間データに対して退避させない，退避するの可能性があるため，中間データの個数を M 個とすると，全探索では $O(2^M)$ の計算量が必要となる．例えば，過去 100 個のセンサ入力をまとめてデータセンタに送信するといったタスクを実行した場合，2^{100} 通りの計算をしなければならない．また今後この電力制御手法を他の低電力化技術と組み合わせて使う場合を考えると，計算時間を減らすことは必須である．そこで，高速に最適解を導くアルゴリズムが必要となる．

提案アルゴリズムでは，まず中間データを以下の法則に従うツリー構造で表現する．

M 個の中間データに対して，以下の 2 条件を満たす中間データ j_c と中間データ j_p が存在したとき，中間データ j_c を中間データ j_p の子とする．

- 中間データ j_c の保持期間を中間データ j_p の保持期間が包含している．
- 中間データ j_c の保持期間を包含し，かつ中間データ j_p の保持期間に包含される中間データが，j_c，j_p 以外に存在しない

ここで，中間データ j の子中間データの ID の集合を $children[j]$ と表す．また全ての子孫中間データの ID の集合を $descendants[j]$ と表す．

次に生成した中間データのツリーに対し，親を持たない中間データ（1 つとは限らない）の ID の集合を $root$ とする．アルゴリズムを適用する前に，全ての i に対し $sleepMode[i] = 0$，全ての j に対し $dataMode[j] = 1$ をセットする．$root$ に属する中間データ全てに対し，それぞれ Algorithm 1 を適用すると，$sleepMode[i]$ と $dataMode[j]$ に解が得られる．

提案するアルゴリズムは，ツリーの葉から探索していく再帰的探索を用いている．1-6 行目は中間データ $dataID$ を退避させた場合にかかるエネルギ（$addEnergy$）を計算している．再帰的に energy 関数を呼び出しており，8-14 行目は中間データ $dataID$ を退避させなかった場合に中間データ $dataID$ がまたがるアイドル区間 $set\_id[dataID]$ で生じる浅いスリープのスタティックエネルギ（$reduceEnergy$）を計算している．16-25 行目では $addEnergy$ と $reduceEnergy$ の値を比較し，部分問題 $\pi(\{dataID, descendants[dataID]\})$ の最適解を導出している．

Algorithm 1 energy($dataID$)

Require: costStatic[i], costData[j], $sleepMode$[i], $dataMode$[j]
1: // 中間データ退避にかかるオーバヘッドエネルギ
2: $addEnergy \leftarrow 0$
3: **for** $j \in children(dataID)$ **do**
4: $addEnergy \leftarrow addEnergy + energy(j)$
5: **end for**
6: $addEnergy \leftarrow addEnergy + costData[dataID]$
7:
8: // この node がまたがるアイドル区間を深いスリープにすると削減できるエネルギ
9: $reduceEnergy \leftarrow 0$
10: **for** $i \in set\_id[dataID]$ **do**
11: **if** $sleepMode[i] = 0$ **then**
12: $reduceEnergy \leftarrow reduceEnergy + costStatic[i]$
13: **end if**
14: **end for**
15:
16: // addEnergy と reduceEnergy を比較
17: **if** $reduceEnergy > addEnergy$ **then**
18: **return** $addEnergy$
19: **else**
20: **for** $i \in set\_id[dataID]$ **do**
21: $sleepMode[i] \leftarrow 1$
22: **end for**
23: $dataMode[dataID] \leftarrow 0$
24: **return** 0
25: **end if**

図 3(a) を例として用いて，アルゴリズムの説明をする．赤で示された双方向矢印はアイドル区間を表しており，ID と cost を持つ．ここで cost は $costStatic[i]$ を表す．また，黒で示された実線は中間データを表しており，ID と cost を持つ．ここで cost は $costData[j]$ を表す．

始めに，図 3(b) で示すように全てのアイドル区間に対して $sleepMode[i] = 0$，全ての中間データに対して $dataMode[j] = 1$ をセットする．ここでは $sleepMode[i]$

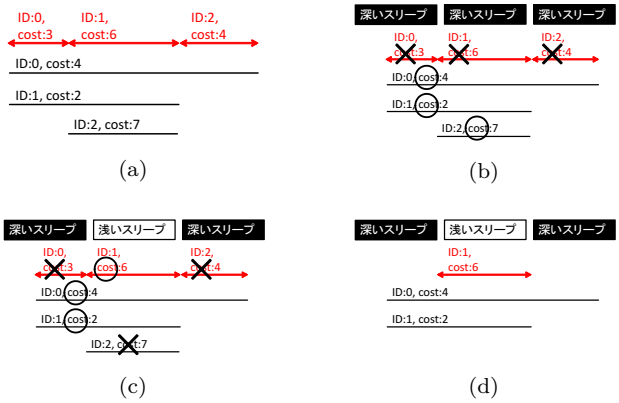

図 3 アルゴリズム適用例

あるいは $dataMode[j]$ が 0 のときを × で，1 のときを ◯ で表す．

次に最も階層の低い中間データ 2 から評価する．中間データ 2 はアイドル区間 1 にまたがる．Algorithm 1 中の $addEnergy$ は 7，$reduceEnergy$ は 6 となる．$addEnergy > reduceEnergy$ となるので，図 3(c) で示すように $sleepMode[1] = 1$，$dataMode[2] = 0$ として，$energy(2)$ は 0 を返す．

次に一つ上の階層の中間データ 1 を評価する．中間データ 1 はアイドル区間 0 と 1 にまたがる．$addEnergy$ は $2 + energy(2) = 2$，$reduceEnergy$ は 3 となる．アイドル区間 1 は浅いスリープになっているので，$reduceEnergy$ には含めない．$addEnergy < reduceEnergy$ となるので，$energy(1)$ は 2 を返す．$sleepMode[0]$，$sleepMode[1]$，$dataMode[1]$ の値は変わらず，図 3(c) のままである．

最後に最上層の中間データ 0 を評価する．中間データ 0 はアイドル区間 0 と 1 と 2 にまたがる．$addEnergy$ は $4 + energy(1) = 6$, $reduceEnergy$ は $3+4 = 7$ となる．アイドル区間 1 は浅いスリープになっているので，$reduceEnergy$ には含めない．$addEnergy < reduceEnergy$ となるので，$energy(0)$ は 2 を返す．$sleepMode[0]$，$sleepMode[1]$，$sleepMode[2]$，$dataMode[0]$ の値は変わらず，図 3(c) のままである．

全ての中間データに対して評価が終わると，アルゴリズムは終了である．最適解は図 3(d) となる．

Algorithm 1 では各中間データに対して一度ずつ計算を行うため，計算時間は中間データの個数 M を用いて $O(M)$ である．

3.4 提案アルゴリズムの最適性

以下の条件 (5) を満たすとき，中間データ集合を層族であるという．

$$\begin{aligned}\forall j, j' \quad & (set\_id[j] \subset set\_id[j'], \\ & \ or\ set\_id[j] \supset set\_id[j'], \\ & \ or\ set\_id[j] \cap set\_id[j'] = \phi) \end{aligned} \tag{5}$$

(a) 層族　(b) 非層族

図 4 層族と非層族の例

これは，任意の二つの中間データを選んだとき，一方の保持期間が他方の保持期間に包含される，あるいは保持期間が重ならない，のどちらかが成り立つことを意味する．また，条件 (5) を満たさない中間データ集合を非層族と言う．**図 4** (a) に層族の例，図 4 (b) に非層族の例を示す．

中間データの集合が層族である場合に，提案アルゴリズムで最適解を導出できることを示す．

補題 3.1 ある中間データ j_0 と j_1 ($set\_id[j_0] \cap set\_id[j_1] = \phi$) が与えられたとき，式 (6)-(8) のように最適化問題を定義する．

$$\pi = \pi(\{j_0, descendants[j_0], j_1, descendants[j_1]\}) \tag{6}$$

$$\pi_0 = \pi(\{j_0, descendants[j_0]\}) \tag{7}$$

$$\pi_1 = \pi(\{j_1, descendants[j_1]\}) \tag{8}$$

このとき，最適化問題 π を解くことは，部分最適化問題 π_0 と π_1 を解くことと等価である．

証明 1 最適化問題の制約条件 (4) を変形する．

$$\begin{aligned} s.t.\quad & sleepMode[i] = 0 \Rightarrow \\ & \forall j\ (\in \{j_0, descendants[j_0]\}) \\ & (i \notin set\_id[j]\ or\ dataMode[j] = 1) \\ & \qquad (i \in set\_id[j_0]) \end{aligned} \tag{9}$$

$$\begin{aligned} & sleepMode[i] = 0 \Rightarrow \\ & \forall j\ (\notin \{j_1, descendants[j_1]\}) \\ & (i \notin set\_id[j]\ or\ dataMode[j] = 1) \\ & \qquad (i \in set\_id[j_0]) \end{aligned} \tag{10}$$

$$\begin{aligned} & sleepMode[i] = 0 \Rightarrow \\ & \forall j\ (\in \{j_0, descendants[j_0]\}) \\ & (i \notin set\_id[j]\ or\ dataMode[j] = 1) \\ & \qquad (i \in set\_id[j_1]) \end{aligned} \tag{11}$$

$$\begin{aligned} & sleepMode[i] = 0 \Rightarrow \\ & \forall j\ (\notin \{j_1, descendants[j_1]\}) \\ & (i \notin set\_id[j]\ or\ dataMode[j] = 1) \\ & \qquad (i \in set\_id[j_1]) \end{aligned} \tag{12}$$

中間データの集合は層族を仮定しているので，常に $\forall i\ (\in set\_id[j_0])\ i \notin \{j_1, set\_id[j_1]\}$ あるいは $\forall i\ (\in set\_id[j_1])\ i \notin \{j_0, set\_id[j_0]\}$ が成り立つ．よって制約条件 (10) と (11) を常に満たしている．考慮すべき制約条件は (9) と (12) のみである．これは $sleepMode[i]$ $(i \in set\_id[j_0])$ あるいは $dataMode[j]$ $(j \in j_0, descendants[j_0])$ の値が，$sleepMode[i]$ $(i \in set\_id[j_1])$ あるいは $dataMode[j]$ $(j \in$

$\{j_1, descendants[j_1]\}$) の値に影響しないことを意味する．以上より，最適化問題 π を部分最適化問題 π_0 と π_1 に分割することができる．

補題 3.2 ある中間データ j_0 が与えられたとき，最適化問題 $\pi(\{j_0, descendants[j_0]\})$ において，中間データ j_0 を退避させないのならば全ての子孫中間データ $j \in descendants[j_0])$ は退避させない方がよい．

$$dataMode[j_0] = 0 \Rightarrow \forall j \in descendants[j_0] \quad dataMode[j] = 0 \quad (13)$$

証明 2 制約式 (4) の対偶を取る．

$$s.t. \quad \exists j \ (i \in set\_id[j] \ and \ dataMode[j] = 0) \Rightarrow sleepMode[i] = 1 \ (i = 0, 1, ..., N-1) \quad (14)$$

制約式 (14) より，中間データ j_0 がまたがるアイドル区間 $i(\in set\_id[j_0])$ は全て $sleepMode[i] = 1$ となる．このとき目的関数は，

$$min \quad \sum_i costStatic[i] + \sum_{j \ (\neq j_0)} dataMode[j] \times costData[j] \quad (15)$$

となる．目的関数 (15) を最小化するためには，$dataMode[j] = 0 \ (\forall j \neq j_0)$ である．

補題 3.3 ある中間データ j_0 が与えられたとき，最適化問題 $\pi(\{j_0, descendants[j_0]\})$ において，中間データ j_0 を退避させるのであれば，部分最適化問題 $\pi(descendants[j_0])$ の最適解と一致する．

証明 3 最適化問題を変形する．

$$\begin{aligned} min \quad & \sum_i sleepMode[i] \times costStatic[i] \\ & + \sum_{j \in descendants[j_0]} dataMode[j] \times costData[j] \\ & + dataMode[j_0] \times costData[j_0] \quad (16) \\ s.t. \quad & sleepMode[i] = 0 \Rightarrow \\ & \forall j \ (\in descendants[j_0]) \\ & (i \notin set\_id[j] \ or \ dataMode[j] = 1) \\ & (i = 0, 1, ..., N-1) \quad (17) \\ & sleepMode[i] = 0 \Rightarrow \\ & (i \notin set\_id[j_0] \ or \ dataMode[j_0] = 1) \\ & (i = 0, 1, ..., N-1) \quad (18) \end{aligned}$$

$dataMode[j_0] = 1$ より，制約式 (18) は常に満たされる．また式 (16) の 4 行目は定数項となり，目的関数から外すことができる．よって最適化問題は，

$$\begin{aligned} min \quad & \sum_i sleepMode[i] \times costStatic[i] \\ & + \sum_{j \in descendants[j_0]} dataMode[j] \times costData[j] \quad (19) \\ s.t. \quad & sleepMode[i] = 0 \Rightarrow \\ & \forall j \ (\in descendants[j_0]) \\ & (i \notin set\_id[j] \ or \ dataMode[j] = 1) \\ & (i = 0, 1, ..., N-1) \quad (20) \end{aligned}$$

となる．これは部分最適化問題 $\pi(descendants[j_0])$ である．

証明 4 数学的帰納法によって，中間データの集合が層族のときに提案アルゴリズムが最適解を導出することを示す．

最下層の中間データ j^0（子を持たない中間データ）について，部分最適化問題 $\pi(j^0)$ の最適解は明らかに，

$$\begin{aligned} if \quad & (costData[j^0] > \sum_{i \in set\_id[j^0]} costStatic[i]) \\ & sleepMode[i] = 1 \ (i \in set\_id[j^0]), \\ & dataMode[j^0] = 0 \quad (21) \\ else \quad & \\ & sleepMode[i] = 0 \ (i \in set\_id[j^0]), \\ & dataMode[j^0] = 1 \quad (22) \end{aligned}$$

である．なお，上付き添え字は階層を表している．

次に中間データ j^k とその子である中間データ $j_0^{k-1}, j_2^{k-1}, ..., j_{p-1}^{k-1}$ が与えられるとする．最適化問題を式 (23)–(25) のように定義する．

$$\pi^k = \pi(\{j^k, descendants[j^k]\}) \quad (23)$$

$$\pi^{k-1} = \pi(descendants[j^k]) \quad (24)$$

$$\pi_l^{k-1} = \pi(\{j_l^{k-1}, descendants[j_l^{k-1}\}) \ (l = 0, 1, ..., p-1) \quad (25)$$

中間データ j^k を退避させないとき，補題 3.2 より中間データ $descendants[j^k]$ は全て退避させない．

中間データ j^k を退避させるとき，補題 3.3 より部分最適化問題 π^{k-1} を解くことに等しい．また補題 3.1 より，部分最適化問題 π^{k-1} の最適解は部分最適化問題 $\pi_l^{k-1} \ (l = 0, 1, ..., p-1)$ の最適解の和となる．中間データ j^k を退避させた場合と退避させない場合との総コストを比べ，小さいほうが部分最適化問題 π^k の最適解である．

以上から，部分最適化問題 $\pi_l^{k-1} \ (l = 0, 1, ..., p-1)$ の最適解が求まれば，部分最適化問題 π^k の最適解が求まる．式 (21) と (22) より最下層の最適解は求まるので，帰納的に元の最適化問題の最適解を求めることができる．

4. 評価

評価に使用するマイコン（ルネサスエレクトロニクス製 RX63N [1]）のパラメータを**表 2** に示す．ここで退避用メモリは内蔵のフラッシュメモリを用いている．

表 2 マイコンパラメータ

パラメータ	値
SP_{cg}	0.0825[W]
$SP_{shallow}$	0.66[mW]
$EOH_{shallow}$	0.402[mJ]
SP_{deep}	0[W]
EOH_{deep}	0.474[mJ]
WE	1.08×10^{-6}[J/byte]
RE	1.33×10^{-10}[J/byte]

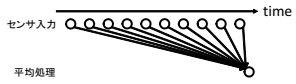

図 5 平均処理

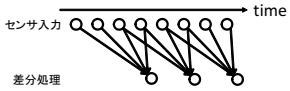

図 6 移動平均処理

表 3 平均処理パラメータ

パラメータ	値
$idleTime[i]$	[II, II, II, II, II, II, II, II][s]
$dataSize[j]$	[DS, DS, 2DS, DS, DS, 2DS, DS, DS, 2DS][byte]
$set\_id[j]$	[{0,...,8}, {1,...,8}, {2,...,8}, {3,...,8}, {4,...,8}, {5,...,8}, {6,...,8}, {7,8}, {8}]

表 4 移動平均処理パラメータ

パラメータ	値
$idleTime[i]$	[II, II, II, II, II, II, II][s]
$dataSize[j]$	[DS, DS, 2DS, DS, DS, 2DS, DS, DS, 2DS][byte]
$set\_id[j]$	[{0,1,2}, {1,2}, {2}, {2,3,4}, {3,4}, {4}, {4,5,6}, {5,6}, {6}]

実行するタスクとして，平均処理と移動平均処理を用いる [2]．平均処理とは一つの出力が複数の入力に依存し，どの出力も他の出力と同一の入力を共有しない処理のことであり，具体的には 3 つの入力の平均値を算出する，などの処理である．平均処理で生成される中間データの集合は層族である．**図 5** に 10 入力を平均処理するタスクグラフの 1 周期分の例を示す．ここで，丸で示されるノードは処理に対応し，線で表されるエッジは依存関係にともなう中間データを保持しなければならない期間に対応している．

次に移動平均処理とは一つの出力が複数の入力に依存し，同一の入力を共有する出力が存在する処理のことであり，**図 6** は移動平均処理から一部を切り出して作成したタスクグラフの例である．具体的には 3 つの入力の平均値を算出するという処理を時系列に対して入力を一つずつずらしながら適用する，などの処理である．移動平均処理で生成される中間データの集合は非層族である．

平均処理と移動平均処理について，入力周期と中間データサイズを変化させて評価を行う．平均処理をパラメータを用いて表現したものを**表 3** に示す．ここで II は入力周期，DS は中間データサイズを表している．同様に，移動平均処理をパラメータを用いて表現したものを**表 4** に示す．

4.1 比較対象

単純な制御手法として，中間データのサイズのみを考慮したスリープ制御を考える．その制御方法を以下に述べる．

中間データのサイズのみを考慮したスリープ制御では，ID の小さいアイドル区間から順にスリープモードを決定していく．

あるアイドル区間 i のスリープモードを決定する．アイドル区間 i で深いスリープにしたときに消費するエネルギは式 (26) で計算される．アイドル区間 i を深いスリープにするには，アイドル区間 i をまたがり，かつ，まだ退避されていない中間データを全て退避させなければならない．これらの中間データを全て退避させるために必要な書き込みエネルギを $writeEnergy[i]$，読み込みエネルギを $readEnergy[i]$ で表す．

$$SP_{deep} \times idleTime[i] + EOH_{deep} + writeEnergy[i] + readEnergy[i] \quad (26)$$

またアイドル区間 i で浅いスリープにしたときに消費するエネルギは式 (27) で計算される．

$$SP_{shallow} \times idleTime[i] + EOH_{shallow} \quad (27)$$

式 (26) と (27) の値を比較し，式 (26) の方が小さければ深いスリープに，式 (27) の方が小さければ浅いスリープに決定し，次のアイドル区間 $i+1$ のスリープモードの判別に移る．深いスリープに決定した場合，退避させた中間データに対しては以降のアイドル区間のスリープモードの判別に影響を与えない．以上を全アイドル区間に対して繰り返すことで，全てのスリープモードが決定する．

4.2 評価結果

4.1 節で説明したデータサイズのみを考慮したスリープ制御手法と，データサイズと保持期間を考慮した提案手法を比較する．クロックゲーティングのみを適用した場合の消費エネルギを 1 とし，それぞれの手法での消費エネルギを正規化した．

平均処理と移動平均処理について，それぞれ入力周期 II を 0.2-1.2[s] の間で振り，データサイズを 1000[byte] で固定した場合，入力周期 II を 1.0[s] で固定し，データサイズ DS を 700-1700[byte] の間で振った場合の結果を示す．

また全区間で浅いスリープを適用した場合（shallow sleep）と，全中間データを退避させ全区間で深いスリープを適用した場合（deep sleep）の正規化消費エネルギ結果を折れ線グラフで示した．

平均処理タスクに対して，横軸に入力周期 II をとったものを**図 7**，横軸にデータサイズ DS をとったものを**図 8** に示す．ここで，比較対象であるデータサイズのみを考慮したスリープ制御を data size only，提案アルゴリズムを proposal，全探索による最適解を optimal で表している．

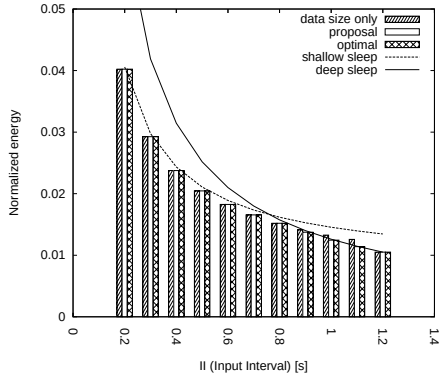

図 7 入力周期 II と正規化エネルギ，DS=1000[byte]（平均処理）

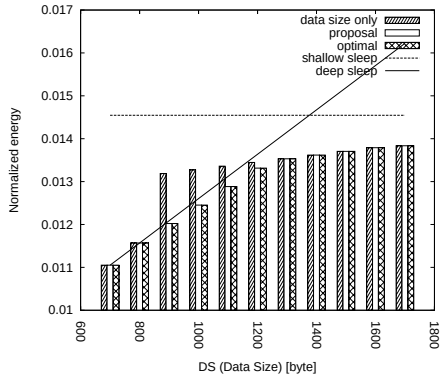

図 8 データサイズ DS と正規化エネルギ，II=1.0[s]（平均処理）

多くの結果で shallow sleep と deep sleep の結果を下回っており，スリープモードを適切に選択することによりオーバヘッドエネルギを削減できることがわかる．また，data size only ではデータサイズが 900 から 1100[byte] の範囲で deep sleep の結果を上回っており，単一の区間ごとにスリープモードを選択する方法では，適切な制御ができない場合があることがわかる．平均処理タスクの中間データは層族となるので，3.4 節で示したように proposal と optimal の結果は常に等しくなっている．

移動平均処理タスクに対して，横軸に入力周期 II をとったものを**図 9**，横軸にデータサイズ DS をとったものを**図 10** に示す．平均処理の結果と同様に，data size only ではスリープモードを適切に選択できない場合があることがわかる．また，データサイズが 1200 から 1600[byte] においても最適な制御ができていない．これは平均処理と異なり，移動平均処理の中間データの集合が層族ではないためである．

5. おわりに

本稿では，スマートセンサシステムにおいてマイコンの深いスリープを使用した省電力手法について述べた．深いスリープでは中間データに着目し，スリープ制御と退避オーバヘッドをモデル化することで，最適なスリープモードと中間データの退避方法を導出するアルゴリズムを提案した．また中間データの集合が層族となる場合に提案アルゴリズムが最適解を導出することを証明した．実際のマイコンのパラメータを用いて平均処理と移動平均処理に対して評価を行い，提案アルゴリズムはほとんどの場合で最適解を導出することを確認した．ただし，非層族の一部のタスクでは最適でない解を導出することもあり，これは今後の課題である．

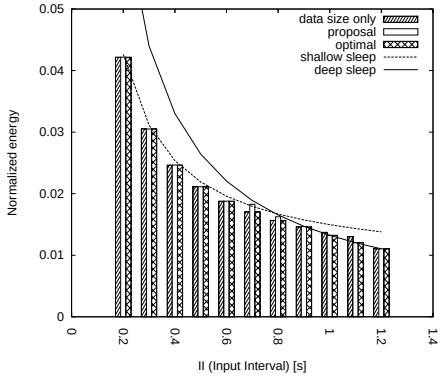

図 9 入力周期 II と正規化エネルギ，DS=1000[byte]（移動平均処理）

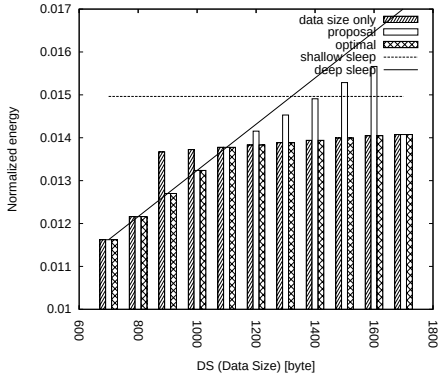

図 10 データサイズ DS と正規化エネルギ，II=1.0[s]（移動平均処理）

6. 謝辞

本研究の一部は，NEDO「ノーマリーオフコンピューティング基盤技術開発」事業による．

参考文献

[1] RX63N グループ，ユーザーズマニュアル ハードウェア編．(http://documentation.renesas.com/doc/products/mpumcu/doc/rx_family/r01uh0041jj0160_rx63n631.pdf)
[2] 岡本和也：エネルギモデルを用いたマルチコア組み込みシステムの設計支援 (2012)．東京大学情報理工学系研究科システム情報学専攻修士論文．