システムソフトウェアと
オペレーティング・システム 92－10
（２００３．２．２８）

# 未知処理量タスクに対する省電力化を目指した周波数制御リアルタイムスケジューリングアルゴリズム

新保稔康*，倉本到*，渋谷雄*，辻野嘉宏* 中本幸一**

携帯機器の普及に伴い，低消費電力技術が重要となってきている．これまで，我々の研究グループでは，プロセッサの周波数を制御することにより，タスクのデッドラインを守りつつ消費電力を削減する様々なリアルタイムスケジューリングアルゴリズムを提案してきた．本研究では，タスク到着時にタスクの処理量が分かるという条件を除いたリアルタイムスケジューリングアルゴリズムを提案し，シミュレーションによって消費電力削減効果を評価した．

# A Real Time Scheduling Algorithm for Tasks with Unknown Work Load to Minimize Power Consumption of Secondary Battery

Toshiyasu Shimbo*, Itaru Kuramoto*, Yu Shibuya*, Yoshihiro Tsujino* and Yukikazu Nakamoto**

The spread of portable devices leads an importance of low power techniques. We have already proposed real-time scheduling algorithms, which maximize utilized time of the secondary battery under preserving deadlines of tasks with arbitrary or discrete frequency control. In this paper we introduces a scheduling algorithm for tasks with unknown work load to minimize battery consumption, and illustrates the effectiveness of the algorithm by simulation.

## 1. はじめに

携帯機器の普及に伴い，低消費電力技術が重要となってきている．携帯機器のディスプレイやハードディスクに対しては，様々な技法を使って消費電力を小さくする試みが行われてきている[1,2,3]．また，回路技術や実装技術の進歩によって CPU 自身の消費電力対策も行われている[3]．しかし, 依然として CPU による消費電力は顕著であり，更なる低消費電力化にはソフトウェアによる粒度の細かい制御が必要である．

例えば，Lee ら[4]は高電圧高周波数と低電圧低周波数の 2 つの走行モードをもった CPU において，タスクのデッドラインを守りつつ，周期タスクに優先度を割り当てるアルゴリズムに基づき CPU の消費電力を最小にする静的スケジューリング方式，及び，タスクの実行の遅れを調整しながら可能な限り低電圧低周波数でタスクを実行させる動的スケジューリング方式を提案し，シミュレーションによって評価した．

しかし，これらの研究では，デッドラインを守りつつ消費電力を最小にするというスケジューリング問題の構造を明らかにしておらず，消費電力を最小にする最適アルゴリズムは示されていない．さらに，2 次電池の利用時間まで考慮した考察はなされていない．

そこで，筆者らの研究グループでは，リアルタイムタスクのデッドラインを守るという制約の下に，一定量の処理を行う場合に，1 回の充電当りに 2 次電池の利用可能な時間を最長にするリアルタイムスケジューリングアルゴリズムの提案を行った[5-8]．

本研究では，これまで提案してきたアルゴリズムにおける，タスク到着時にタスクの処理量が分かるという条件を緩めた場合のスケジューリングアルゴリズムを提案し，シミュレーションにより

* 京都工芸繊維大学
Kyoto Institute of Technology
** 日本電気
NEC

評価する．

## 2. 2 次電池の特性

通常，携帯機器は，CPU，メモリ，ディスプレイ，2 次電池，通信ポートなどから構成される．本研究では CPU の周波数が制御できる場合に，消費電力が周波数によって変化することを利用して 2 次電池の利用可能時間を最長にする手法を述べる．

以下では 2 次電池と周波数の関係について述べる．

**[定理 1]**[5,6]長さ $T$ の時間を区間 $D_0, D_1, \ldots, D_{m-1}$ に分割する．時間幅 $t_i$ である区間 $D_i$ における CPU の周波数を $f_i$ とする時，時間 $T$ における 2 次電池の残存電流量の減少量(以下，2 次電池の消費電力量と呼ぶ)は以下のように表される．

ただし，$C$，$\alpha$ は定数で，$C>0$，$\alpha>1$ である．

$$E(t_0, f_0; \ldots; t_{m-1}, f_{m-1}) = C\sum_{i=0}^{m-1} t_i \cdot f_i^{\alpha} \quad (1)$$

ここで，本研究のように CPU の周波数が可変の環境下では，CPU の周波数に依存しないプログラムの処理量の尺度が必要である．そこで，本研究では仕事量というプログラムの処理量を以下のように定義する．

**[定義 1]**実行周波数 $f$ で時間 $T$ の間に処理できる CPU の仕事量 $W$ を $W=fT$ と定義する．

このことは，CPU が 1 クロックで実行可能な処理は一定であるとの仮定に基づき，仕事量がその処理に必要なクロック数で表せることを示す．

また，定理 1 から，次の定理が成り立つ．

**[定理 2]**[5,6]仕事量を，時間幅 $t_0$ , $t_1$ である区間 $D_0, D_1$ に分割して実行した．区間 $D_0, D_1$ における CPU の周波数をそれぞれ $f_0, f_1$ とする時，2 次電池の消費電力量 $E(t_0, f_0; t_1, f_1)$ は $|f_0 - f_1|$ について単調増加である．

また

**[系 1]**[5,6] 仕事量 $W=fT$ を，時間幅 $t_0, t_1, \ldots, t_{m-1} (T = \sum_{i=0}^{m-1} t_i)$ である区間 $D_0, D_1, \ldots, D_{m-1}$ に分割して実行する．このとき，各区間を同一周波数 $f$ で実行したときに 2 次電池の消費電力量の式(1)は最小になる．

したがって，ソフトウェアによって CPU の周波数を制御できるシステムにおいて，周波数の変化の差が小さいほど，2 次電池の消費電力量は小さくなり，同一で可能な限り低い周波数でプログラムを実行した場合に 2 次電池の消費電力量は最小になる．また，周波数にあわせて電圧も制御できる場合も式(1)が成立し，電圧を下げる事でさらに消費電力を削減できる．

## 3. スケジューリングアルゴリズム

筆者らの研究グループでは文献[5]において，タスク到着時にそのタスクの仕事量が分かると仮定した 2 次電池の最長スケジューリングアルゴリズムの提案を行った．本研究ではこのアルゴリズムを基に，未知仕事量タスクに対するスケジューリングアルゴリズムを提案する．

### 3.1 仕事量が既知のタスクに対するスケジューリングアルゴリズム

本節で扱う問題は，リアルタイムタスクのデッドラインを守るという制約の下に，2 次電池の 1 回の充電当たりに利用可能な時間を最長にするスケジューリング更新問題[5]である．また，2 次電池の消費電力量が CPU の周波数によって制御できる点を利用している．

**[定義 2]**タスクにおいてデッドラインを守る性質を *dl-safe ( deadline-safe )* と呼ぶ

簡単化のために以下の仮定をおく．

1. 各タスクは独立である(他のタスクに依存せず実行することができる)．
2. タスクの切替えやスケジューリングに伴うオーバーヘッドは考慮しない．

本問題では，入力となるタスクは任意の時刻に到着する．ただし，到着タスクの仕事量 $w$，タスクの実行が完了しなければならないデッドライン時刻 $d$ がタスク到着時に分かるものとする．

本問題は，以下の入力から出力を得るスケジューリング更新問題である．

**入力**：以下の条件を満足する最適タスクスケジュールと到着したタスク

**条件 1**：各タスクは *dl-safe* であること，つまり，イベントの発生からデッドライン以内にタスクの実行を完了させること

**条件 2**：それ以降タスクが到着しない場合に，単位仕事量当たりの 2 次電池の消費電力量を最小にすること．

**出力**：到着したタスクを含め，上記 2 つの条件を満たすタスクスケジュール．

この問題を解くアルゴリズム MINBAT [5]の概要を述べる．動的スケジューリングアルゴリズムにおいて様々な実行環境下で最適であることが知

られている *EDF(Earliest Deadline First)*を応用して条件 1 を満たす. *EDF* はデッドライン順にタスクを並べた実行待ちタスクリストで実現できる．さらに条件 2 を満たすために，同一で可能な限り低い周波数でタスクを実行させるようにする．そこで，同一周波数で実行するタスクを管理するタスクブロックを導入する．タスクブロックは同一周波数で連続して実行されるタスクのリストであり，以下の性質を持つ.

**性質 1**：タスクブロック内のタスクはデッドライン順に並んでいる.

**性質 2**：タスクブロックの最終タスクの実行終了時刻はそのタスクのデッドラインと等しい.

**性質 3**：タスクブロック内のタスクは *dl-safe* である.

さらに，タスクブロックを実行開始時刻の昇順で並べたリストを実行待ちタスクリスト $L$ と呼ぶ.

**[定義 3]**実行待ちタスクリストにおいて, 隣接するタスクブロックの周波数が降順でない場合に，周波数逆転が生じているという.

タスクが到着した時の MINBAT の動作概要を示す.

1. 到着したタスク $\tau$ をデッドライン順に並ぶように実行待ちタスクリスト $L$ に挿入し，実行周波数 $f$ を更新する.
2. タスク $\tau$ が挿入されたタスクブロック *TB* は仕事量が増加するので，*TB* 内のタスクが *dl-safe* でなくなる可能性がある. この場合は *TB* の先頭のタスクから *dl-safe* でないタスクまでと，それ以降を $TB_x$ と $TB_y$ として 2 つのタスクブロックに分割し，実行周波数を更新する. これを *dl-safe* でないタスクがなくなるまで繰り返す.
3. 2 のようにタスクブロックを分割した場合，分割されたタスクブロックの前後で周波数逆転が生じる場合がある. 周波数逆転が生じているとき，その 2 つのタスクブロックを連結して同一周波数で実行することにより，*dl-safe* を保持したまま 2 次電池の消費電力量を減らすことができる.

実行待ちタスクリストからタスクを取り出す時には先頭タスクブロックの先頭タスクから順に取り出す.

MINBAT を応用して任意時刻に周波数の切替えを行うアルゴリズムを STEFC と呼ぶ．また，筆者らの研究グループではタスク切替え時のみに周波数を切替え，定数個の値(離散周波数)しか取れない場合のスケジュール更新問題は NP 完全であることを示し，MINBAT を基に周波数の切替えを任意時刻に行い，離散周波数に設定できる場合のアルゴリズム STEDFC の提案を行っている[6].

表 1 提案済み最適アルゴリズム

| | 任意時刻 | タスク切替え |
|---|---|---|
| 連続周波数 | STEFC | MINBAT |
| 離散周波数 | STEDFC | (NP 完全) |

表 1 に各アルゴリズムとその条件を示す.

### 3. 2 未知仕事量タスクに対するスケジューリングアルゴリズム

本問題ではタスク到着時にタスクの仕事量が分からないとする．そこで，仕事量を定数($Cw$)であると仮定することで，各アルゴリズムを用いてスケジュールの更新を行う手法をとる.

本問題は，以下の入力から出力を得るスケジューリング更新問題である.

**入力**：以下の条件を満足する最適タスクスケジュールと到着した未知仕事量タスク

**条件 1**：各タスクは *dl-safe* であること，すなわち，イベントの発生からデッドライン以内にタスクの実行を完了させる.

**条件 2'**：それ以降タスクが到着しない場合に，かつタスクの仕事量が仮定した仕事量と等しい場合に単位仕事量当たりの 2 次電池の消費電力量を最小にすること.

**出力**：到着したタスクを含め，上記 2 つの条件を満たすタスクスケジュール.

この問題を解くアルゴリズムを SWMFC と呼ぶ. このアルゴリズムの入力と出力を以下に示す.

**入力**：条件 1，2'を満足するスケジュールを実現するタスクリスト $L$ と到着した未知仕事量タスク $\tau$

**出力**：到着した未知仕事量タスク $\tau$ を含め，条件 1，2'を満たすタスクリスト $L$

#### 3. 2. 1 仮定した仕事量と実際の仕事量

仮定した仕事量($Cw$)と実際の仕事量($w$)の間には以下のような場合がある.

(1) 仮定した仕事量が実際の仕事量より大きい場合($Cw > w$)

(2) 仮定した仕事量が実際の仕事量と等しい場合($Cw = w$)

(3) 仮定した仕事量が実際の仕事量より小さ

い場合($Cw < w$)

(1)の場合はスケジュール上のタスク終了時刻よりも早くタスク $\tau_i$ の処理が終了する．その時点でスケジュールを更新する．

(2)の場合はスケジュール通りなので，スケジュールに従い次のタスクの実行を行う．また実行可能タスクがない場合には停止する．

(3)の場合にはスケジュール上の終了時刻には実際にはタスクを完了しないので，新たに仮定した仕事量($Cw$)を持ったタスクが到着したものとみなし，スケジュールを更新し実行する．しかし，タスクのスケジュール上の終了時刻とデッドラインが等しい場合には，その時点で新たなタスクが到着したとみなして処理を再開してもデッドラインを超えてしまう．そこで，本研究ではダミータスクを用いてデッドラインを守る方法を提案する．

### 3. 2. 2 ダミータスク

本アルゴリズムでは *dl-safe* を保持するためにダミータスクを導入する．ダミータスクとは以下のような特徴を持っている．

- 実際の仕事量は 0 である．スケジューリングする際には通常のタスクと同様に仕事量($Cw$)をもったタスクとして扱う．
- ダミータスクは各タスクブロックの最終タスクの次に挿入する．ダミータスクのデッドラインはタスクブロックの実行終了時刻と等しい．つまり，各タスクブロックの最終タスクのデッドラインと等しい．
- ダミータスクがタスクリストの先頭タスクブロックの先頭となった場合には時間 0 で完了したものとみなし，次の処理へ進む．

ダミータスクをスケジューリング済みの各タスクブロックに挿入すると，タスクブロック内では仕事量が増え，周波数が上昇する．しかし，タスクブロック内の各タスクの終了時刻はダミータスクを挿入する前の終了時刻よりも挿入後早くなり，*dl-safe* は保持される．

しかし，ダミータスク含むタスクブロックの間に周波数逆転が生じる可能性がある．従来のアルゴリズムでは，周波数逆転が生じたタスクブロックを連結していた．本アルゴリズムでは，ダミータスクは各タスクブロックの最後にだけ存在するので，タスクブロックを連結した場合一方のタスクブロックのダミータスクは削除しなくてはならない．しかし，ダミータスクを削除するとタスクブロックの仕事量が減少し，周波数が下降する．結果として，*dl-safe* を破るタスクが存在する可能性が生じる．また，*dl-safe* を破るタスクが存在する場合にはタスクブロック *TB* の分割を行う必要がある．しかし，タスクブロック *TB* を $TB_x$，$TB_y$ に分割を行うと $TB_x$ にはダミータスクが存在しないので挿入する必要がある．ダミータスクの挿入により仕事量が増加し，周波数は上昇する．結果として，周波数逆転が生じる可能性がある．これらのことを繰り返すと，スケジュール更新に伴う計算量が爆発的に増加してしまう．

本アルゴリズムでは，ダミータスクを挿入したタスクブロックの周波数が上昇する際に，周波数逆転の可能性があるのはそのタスクブロックよりも終了時刻が早いタスクブロックであるという点に注目した．そこで，最終タスクブロックから順にダミータスクを挿入する(図 1(a))．

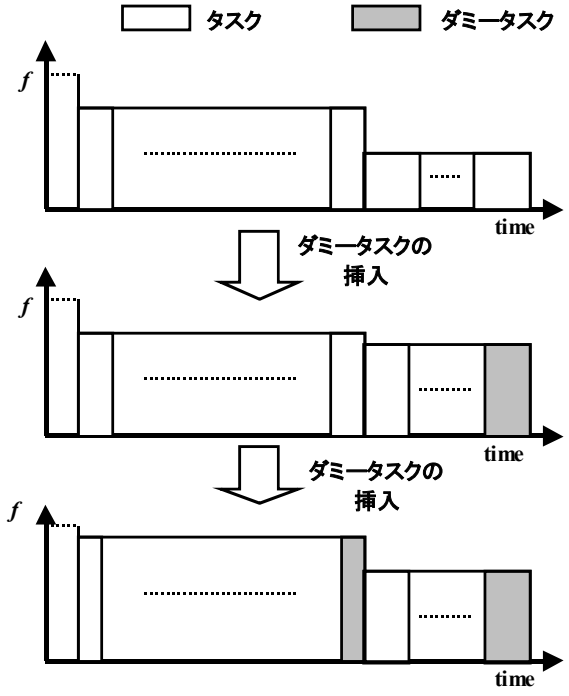
ダミータスクを挿入したタスクブロックとまだ挿入されていないタスクブロックの間で周波数逆転が生じた場合には，タスクブロックの連結を行う(図 1(b))．この連結では先行するタスクブロックにダミータスクがまだ挿入されていないので，ダミータスクを削除することによる周波数の低下が起こらず，*dl-safe* を破ることはない．これにより，一度タスクリストの最終タスクブロックから先頭のタスクブロックまで処理するだけでダミータスクを含む実行待ちタスクリスト $L'$ を生成することができる．

以下の手順でスケジュールの更新を行う．

1. 提案済みのアルゴリズムを用いて最適なタスクリスト $L$ を生成する．
2. タスクリスト $L$ の最終タスクブロックに対してダミータスクを挿入する．
3. 挿入されたタスクブロックとその前方のタスクブロックの間で周波数逆転が起こった場合には，2 つのタスクブロックを連結する．この処理を周波数逆転が起こらなくなるまで繰り返す．
4. ダミータスクの挿入を行っていない最後のタスクブロックに対してダミータスクを挿入する．
5. 3，4 をタスクリストの先頭のタスクブロックまで繰り返す．

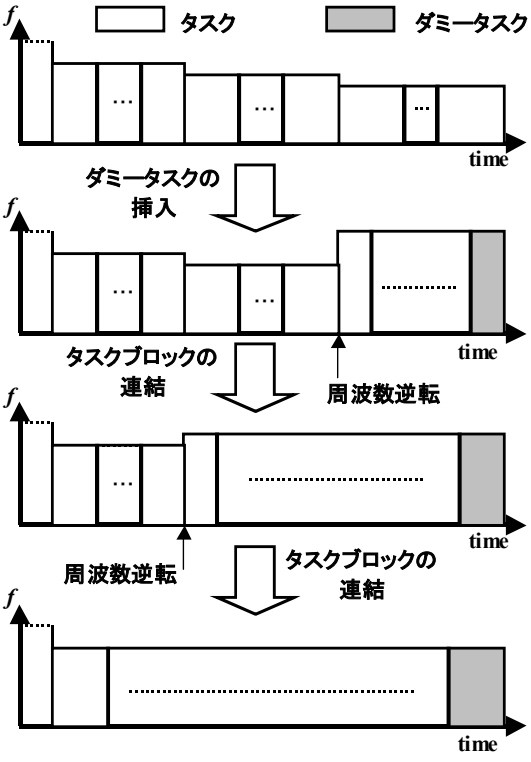
本アルゴリズムでは，上記のようにダミータスクを用いるため，ダミータスクを挿入する前のタスクリスト $L$ とダミータスクを挿入したタスクリスト $L'$ を保持し，$L$ にスケジューリングした後に

ダミータスクを挿入し$L'$を作成する．タスク実行の際にも$L$と$L'$内のタスクの関係を保持する．




(a) 周波数逆転が生じない場合

図1　スケジュールの更新




(b)周波数逆転が生じる場合

図1　スケジュールの更新

本アルゴリズムはダミータスクを挿入した事により，条件1を満たすが条件2'を満たしていない．すべてのタスクが仮定した仕事量と等しい場合に，ダミータスクの挿入に伴う実行周波数の増加はダミータスクを挿入しない場合に比べ最悪で2倍となり，消費電力が増加する．そこで，シミュレーションにより，本アルゴリズムの消費電力削減効果を評価する．

## 4. シミュレーション

仮定した仕事量の値が消費電力に与える影響をシミュレーションにより評価する．そこで，タスク到着時に分かる仕事量の情報を以下のように考え，それぞれのシミュレーションにおける条件を述べる．

(1) 未知の場合

(2) 最悪仕事量が既知の場合

(1)の場合は仕事量が未知なので，実際の仕事量の分布([2，20]MHz･msecの一様分布)に対して，非常に小さい場合(2MHz･msec)から非常に大き[illegible](40MHz･msec)までそれぞれの値でシミュレーションを行い，仮定する仕事量が消費電力に与える影響を評価した．

(2)の場合は最悪仕事量*WCWL(Worst Case Work Load)*が分かっているので，仮定する仕事量を最悪仕事量とし，実際の仕事量の分布を変えて，分布による消費電力への影響をシミュレーションによって評価する．実際の仕事量の分布を[$aWCWL$，$WCWL$] MHz･msecの一様分布とする．ただし，$0<a<1$である．

以下にシミュレーションの条件を示す．

タスク到着：1.0[msec]間隔

デッドライン：到着時刻+1.5~8.5[msec]

タスクの数：100

離散周波数の刻み幅：1，2，…，15[MHz]

ここで消費電力の計算には式(1)を用い，$C=1$，$\alpha=1.6$を用いた．

また，周波数切替えのタイミングと設定できる周波数により，表1のように各アルゴリズムを用いる．

各図の消費電力比とは1MHzで1msec　の間CPU を実行させた時の消費電力との比(100 回の平均値)である．

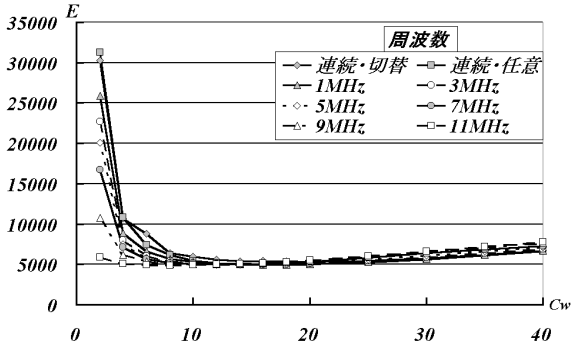



図2　消費電力比-仮定仕事量

(仕事量が不明の場合)

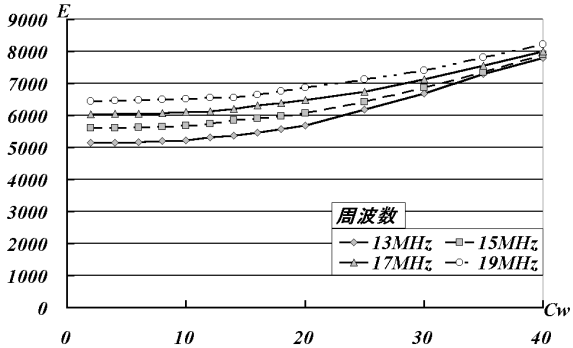



図3　消費電力比-仮定仕事量

(仕事量が不明の場合)

## 4. 1 タスクの仕事量が未知の場合

図2は仮定した各仕事量と消費電力比の関係を示しており，周波数は連続周波数か離散周波数の刻み幅が細かい場合である．周波数切替えタイミングの違いに関わらず仮定した仕事量が小さすぎると消費電力が非常に多くなり，大きくなるにつれて再び消費電力が増加する傾向にあることが分かる．

これは，仮定した仕事量が実際の仕事量の *1/n* であったとすると，*n* 回タスクの再スケジュールを行う必要があり，その度にデッドラインまでの時間は短くなるので周波数は上昇する．したがって *n* が大きい場合，つまり仮定した仕事量が小さすぎる場合には周波数が非常に大きくなり，消費電力が増加する．

そして，仮定した仕事量が実際の仕事量の *m* 倍であるとすると，タスクはスケジュール上の割り当てられた時間の *1/m* の時間で完了する．また周波数は実際の仕事量に対して *m* 倍になる．結果として，スケジュール上の終了時刻よりも早く完了する．そして，実行可能タスクがなくなった場合には停止する．この高い周波数での実行と停止を繰り返した結果，消費電力が増加する．

また，仮定した仕事量が実際の仕事量に近いほど，実行周波数はスケジュールの周波数に近くなるので消費電力は減少する．

図3は離散周波数の刻み幅が粗い場合の消費電力比と仮定した仕事量の関係を表している．周波数の刻み幅が粗いものは仮定した仕事量が非常に小さい場合でも消費電力比が他のものほど増加していない．

これは，仮定した仕事量が小さく必要な周波数が低くても周波数の刻み幅が粗いので，高い周波数が割り当てられる．結果として，わずかな時間で仮定した仕事量を完了し，実際にはタスクを完了していないので再びタスクが到着したとみなし，スケジュールを更新する．ここで，低い周波数で実行した場合よりもデッドラインまでの時間が長いので，再び低い周波数が必要となる．しかし，刻み幅が粗いのでほとんどの場合前に実行した時と同じ周波数が割り当てられる．このことを繰り返すので，周波数の異常な上昇が起こらず，消費電力の増加がほとんど起こらない．

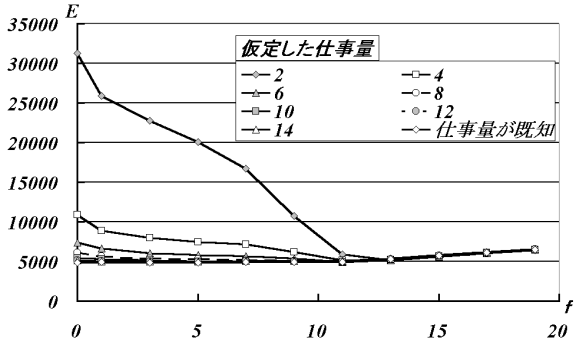



図4　消費電力比-周波数の刻み幅

(仮定仕事量：2~12MHz･msec)

また，仮定した仕事量が大きすぎる場合は図 2 と同様に実行と停止を繰り返す事が多くなり消費電力が増加する．

図4は周波数の刻み幅と消費電力の関係を示しており，仮定している仕事量は 2~14MHz･msec である．この図では周波数の刻み幅が粗くなるにつれ消費電力が減少し，11MHz 刻みを越えてからは消費電力が増加傾向にある．

これは，周波数の刻み幅が非常に細かい場合には，仮定した仕事量に対して必要な周波数に近い値が設定できる．しかし，仮定した仕事量と実際の仕事量に差があるとスケジュール通りにタスクは完了しない．その結果再スケジュールを行う機会が増え，周波数の上昇が生じるので消費電力が多くなる．

しかし，周波数の刻み幅が非常に粗い場合には

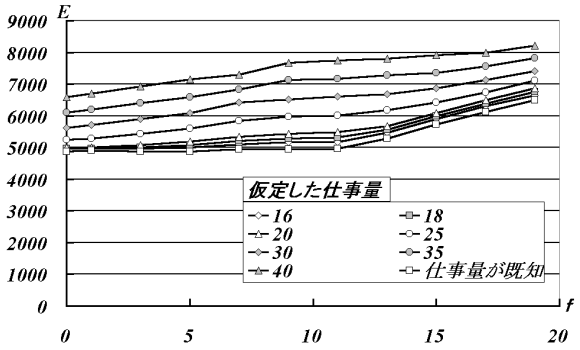



図5　消費電力比-周波数の刻み幅

(仮定仕事量：14~40MHz･msec)

仮定した仕事量に対し，必要な周波数よりも高い周波数が割り当てられる．この場合，仮定した仕事量はスケジュールよりも早く終了する．そして，タスクの実行と再到着を繰り返し，タスクは早く完了するが実行可能なタスクも全て完了させてしまい，停止する時間が長くなる．結果として，実行と停止を繰り返し消費電力が増加してしまう．そして，11MHz 付近では仮定した仕事量に対して必要な周波数よりもわずかに高い周波数が割り当てられる．その結果，仮定したタスクはデッドラインよりも早く終わり，再スケジュールの際にも急激には周波数が増加しないので，周波数の刻み幅が細かい場合や粗い場合に比べ消費電力が少ない．

図5は図4と同様のグラフであり，仮定している仕事量は 16~40MHz･msec である．このグラフでは周波数の刻み幅が粗くなるにつれ消費電力が増加している．

この場合は，仮定した仕事量が実際の仕事量よりも大きいということが頻繁に起こる．周波数の刻み幅が細かい場合は，スケジュールよりもタスクが早く完了しても余る時間は少なく，実行可能なタスクが存在する場合が多いのでスケジュールから大きくはずれる事は無い．しかし，周波数の刻み幅が粗くなるにつれ，実行周波数は高くなり，タスクの早期完了に伴う余剰の時間は長くなる．結果として，実行可能なタスクが無くなり停止する時間も長くなる．このことが消費電力の増加につながる．

## 4. 2 最悪仕事量のみ分かる場合

図6は最悪仕事量が既知で，仮定する仕事量を最悪仕事量にした場合である．横軸は実際の仕事量の分布を示しており，値が大きいほど最悪仕事量に近い値に分布している事になる．この図から周波数切替えタイミングの違いに関わらず分布

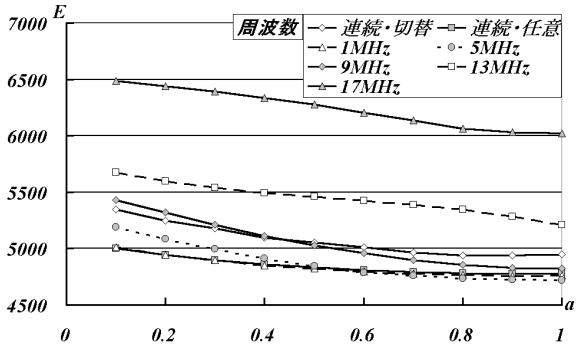



図6 消費電力比-仮定仕事量

(最悪仕事量が既知の場合)

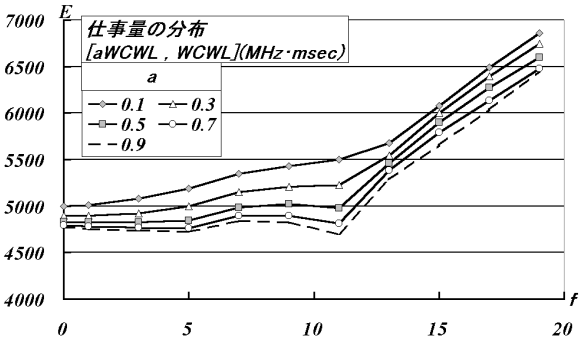



図7 消費電力比-周波数の刻み幅

(最悪仕事量が既知の場合)

が狭いほど消費電力が少ない事が分かる．これは，仮定した仕事量と実際の仕事量が近づくほど，タスクリストのスケジュールにタスクの実行と完了が近づくためである．

図7は最悪仕事量が既知で，仮定する仕事量を最悪仕事量にした場合の周波数と消費電力の関係を示している．周波数の刻み幅が細かい場合は必要な周波数に近く設定できるために消費電力は少ない．特に仮定した仕事量が実際の仕事量と近い場合には，必要な周波数よりも少し上の周波数が与えられる方が新たなタスクの到着に伴う仕事量の増加と周波数の上昇が抑えられる．しかし，周波数の刻み幅が粗いと必要以上の周波数で実行してしまうので，実行と停止を繰り返し消費電力が増加する．また周波数の刻み幅によって，仮定した仕事量が必要とする周波数に近い値が設定できない場合には必要以上の周波数で実行してしまい，消費電力が増加している．

またシミュレーション全体を通して，仕事量が既知の場合のアルゴリズムに比べ，最悪の場合で約 6.4 倍であった．これは仮定した仕事量があまりにも小さかったために再スケジュールを繰り返した場合に起こった．また，消費電力が最も少ない場合，仕事量が既知のアルゴリズムと同等の消

費電力であった．これは，周波数の刻み幅がやや粗く，仮定した仕事量は平均仕事量よりも少ない場合に起こった．この組み合わせの場合，再スケジュールを数回行う事で，実際の仕事量に対して最適な周波数よりもやや高い周波数で実行し，デッドラインよりもわずかに早く処理を完了する事で新たなタスクの到着時に周波数の急激な増加を抑えられたためだと考えられる．

最後に，固定周波数(20MHz)で実行した場合の消費電力比に対して，本アルゴリズムでは最大62%減少した．ここで，20MHz はシミュレーションの条件下で新たなタスクが到着する前に現在のタスクを確実に完了できる周波数である．

## 5. まとめ

本研究では，未知仕事量タスクに対するスケジューリングアルゴリズムの提案を行い，シミュレーションにより，仮定する仕事量の値，周波数のタイプ，周波数変更のタイミングの違いが消費電力に与える影響を評価した．

その結果，周波数切替えタイミングの違いに関わらず仮定する仕事量が小さすぎても大きすぎても消費電力は増加した．そして，周波数の刻み幅が粗い場合には，仮定する仕事量($C_W$)の値による消費電力への影響を受けにくいことが分かった．

また，仮定した仕事量が実際の仕事量に近づくにつれ，周波数の刻み幅による特徴が現れた．その特徴として，周波数の刻み幅は最適な周波数よりもわずかに大きい値の時に消費電力が減少することと，連続周波数よりもわずかに周波数の刻み幅が粗い時に消費電力が減少するというものであった．

仮定する仕事量と周波数の刻み幅の消費電力を減少させる組み合わせは，平均仕事量よりも小さい仕事量を仮定し，刻み幅を最適な周波数よりもわずかに粗くした場合である事が分かった．その場合，本アルゴリズムの消費電力削減効果は仕事量が既知の場合のスケジューリングアルゴリズムと同等であることを確認した．

## 参考文献


[1]相原, 関谷, 黒田, ”ノートブック PC におけるローパワーシステム技術”,電子情報通信学会誌, Vol.80, No.4, pp.370-376(1997).

[2]特集ノート PC のジレンマ, 日経バイト, No185(1998).

[3]携帯情報端末機器に対する低消費電力技法論文特集, 信学論, Vol.J80-A,No.5,pp.727-771(1997).

[4]Y.H.Lee and C.M.Krishna ,”Voltage-Clock scaling for Low Energy Consumption in Real-time embedded Systems”, Proc of 6th International Conference on Real-Time Embedded Systems and Application, pp.272-279(1999).

[5] 中本, 辻野, 都倉:”携帯機器の 2 次電池の最長利用を目的とするリアルタイムスケジューリングアルゴリズム”,信学論, Vol.J83-D-I No.1,pp.125-133 (2000).

[6] 中本, 辻野, 都倉:”離散周波数制御を利用した 2 次電池の最長利用のためのタスクスケジューリングアルゴリズム”, 信学論, Vol.J83-D-I, No.12, pp.1249-1259 (2000).

[7] 岡本, 渋谷, 辻野, 中本:”2 次電池の最長利用のための実行順序制約を用いたスケジューリングアルゴリズムについて”, 情報処理学会アルゴリズム研究会資料, 76-6, pp.35-42 (2001).

[8]新保,渋谷,辻野:”消費電力の最小化を目指したリアルタイムスケジューリングアルゴリズムの効果”,情報処理学会論文誌,Vol.43, No.8 , pp.2836-2839 (2002).