

コア状態を考慮したスラック収集型 リアルタイムタスクスケジューリング

松井 健太郎^{1,a)} 高瀬 英希^{1,b)} 高木 一義¹ 高木 直史¹

概要: 近年の組込みリアルタイムシステムでは、より高い性能が求められるとともに、その消費エネルギーを削減することが重要な課題となっている。本研究では、複数のコア状態をもつ単一コアを対象とした、消費エネルギー最小化のためのタスクスケジューリング手法を提案する。提案手法は、Slack Gathering laEDF (SGlaEDF) と Core State Aware Scheduling (CSAS) からなる。SGlaEDF は、周期タスクの性質を利用してタスクの実行完了時に次のデッドラインおよび残り最悪実行時間を更新することで、デッドライン制約を保証した上で、スラック時間をより正確に導出できるアルゴリズムである。CSAS は、コア状態を考慮した上で、総消費エネルギーを削減するスケジューリング手法である。提案手法では、SGlaEDF により導出される正確なスラック時間およびデッドライン制約を保証する動作周波数を用いて CSAS を実行することで、リアルタイムシステムのデッドライン制約を保証しつつ総消費エネルギーの最小化を目指す。評価の結果、提案手法は既存手法と比較して最大で 29%、平均で 12% の消費エネルギー削減効果があることが示された。

1. はじめに

近年の組込みリアルタイムシステムにおいては、高性能化を達成しつつ消費エネルギーを最小化することが重要な課題となっている。代表的な消費電力削減技術として、動的電圧・周波数制御 (Dynamic Voltage and Frequency Scaling, DVFS) および動的電力管理 (Dynamic Power Management, DPM) がある。DVFS は、コアの供給電圧および動作周波数を適切に制御してタスクを実行することで、システムの動作時消費エネルギーを削減する技術である。DPM は、CPU の負荷が無いアイドル時に CPU 各部のクロックまたは電源供給を停止することで、アイドル時消費エネルギーを削減する技術である。組込み向けプロセッサでは、アイドル時に供給停止するモジュール群に応じて複数のコア状態が用意されている。

DVFS および DPM では、各タスクの実行時間およびスラック時間を用いる。スラック時間とは、タスクの実行終了から次にタスクが実行開始されるまでの余裕時間のことである。スラック時間の見積もりの正確さは、DVFS および DPM の消費エネルギー削減効果に大きな影響を与える。また、全体の消費エネルギーには、動作時消費エネルギーだけではなくアイドル消費エネルギーも含まれる。このた

め、総消費エネルギーが最小となるように、DVFS および DPM を適用することが重要である。さらに、リアルタイムシステムにおいては、タスクの実行を所定の時刻までに必ず完了しなければならない制約であるデッドライン制約を保証する必要がある。

本研究では、複数のコア状態をもつ単一コアを対象とした、消費エネルギー最小化のためのタスクスケジューリング手法を提案する。提案手法は Slack Gathering laEDF (SGlaEDF) および Core State Aware Scheduling (CSAS) からなる。SGlaEDF によって正確なスラック時間およびデッドライン制約を保証する動作周波数を導出し、これらを用いて CSAS を実行することで、デッドライン制約を保証しつつ総消費エネルギーを最小化することを目指す。

SGlaEDF では、周期タスクの性質を利用し、タスクの実行完了時に次のデッドラインを算出して更新することで、タスクの実行可能時間が必ず直近のデッドラインまでに収まるようにする。これにより、タスクの実行可能時間が直近のデッドラインを超えてしまう場合に、スラック時間を過剰に見積もり、動作時消費エネルギーが増大してしまう問題を改善することができる。SGlaEDF は、消費エネルギー削減効果の高い既存手法である look ahead EDF (laEDF) [1] より正確なスラック時間およびデッドライン制約を保証する最低動作周波数を導出する。

CSAS では、コア状態遷移にかかる時間および消費エネ

¹ 京都大学 Kyoto University

^{a)} matsui@lab3.kuis.kyoto-u.ac.jp

^{b)} takase@i.kyoto-u.ac.jp

ルギーのオーバーヘッドを考慮し、スラック時間における消費エネルギーを最小にするコア状態遷移を決定する。さらに、動作周波数を上昇させることによる、動作時消費エネルギーの増大効果およびアイドル時消費のエネルギー削減効果を比較する。これにより、タスクの実行可能時間における最適な動作周波数設定およびコア状態遷移を導出し、総消費エネルギーの削減を実現する。

提案手法では、SGlaEDF および CSAS により、デッドライン制約を保証しつつ消費エネルギー削減効果の向上を実現する。具体的には、まず SGlaEDF により、正確なスラック時間およびデッドライン制約を保証する最低動作周波数を導出する。次に、正確に導出されたスラック時間を用いて、CSAS による適切なコア状態遷移を行うことで、CSAS のエネルギー削減効果の向上が期待できる。CSAS の設定可能な動作周波数を、SGlaEDF で導出した動作周波数以上に限定することで、デッドライン制約は保証される。

本研究の構成は以下の通りである。まず 2 章にて、関連研究について述べる。3 章にて研究対象とするシステムモデルおよび既存手法の問題点を説明し、4 章にて提案手法を解説する。5 章にて提案手法の有効性を評価したのち、6 章にて本稿のまとめと今後の方針を述べる。

2. 関連研究

DVFS および DPM の消費エネルギー削減効果の向上には、タスクの実行時間およびスラック時間のより正確な導出が重要となる。この課題を解決するために、これまでに様々な手法が提案されている。

文献 [2] [3] は、タスクの優先度関係からタスクのスラック時間をより厳密に算出する手法である。文献 [4] は、カルマンフィルタを使うことで、タスクの実際の実行時間およびスラック時間をより正確に推定する手法である。文献 [5] は、デッドライン制約を保証しつつ実際の実行時間およびスラック時間を使用することで、動作時消費エネルギーを削減する手法である。

アイドル時消費エネルギーについても、様々な研究が行われている。文献 [6] は、RTOS の実行駆動型シミュレーションにより、アイドル状態において適切な電力制御を行わない場合に、消費エネルギーが大きく増大することを示している。文献 [7] は、遅延分岐時間を用いて、最適なコア状態遷移を導出する手法である。文献 [8] は、機械学習によって最適なコア状態遷移動作をモデル化する手法である。文献 [9] は、コア状態遷移による時間および消費エネルギーのオーバーヘッドに着目し、頻繁なコア状態遷移を抑えつつ、アイドル時消費エネルギー削減を行う手法である。

3. 対象システムと問題の定義

3.1 対象とするシステムモデル

タスクセット T は、 I 個の互いに独立な周期タスク

$\tau_1, \tau_2, \dots, \tau_I$ から構成される。各タスク τ_i は、周期 P_i 、リリース時刻 r_i 、デッドライン時刻 d_i 、最高周波数で τ_i を実行した場合の最悪実行時間 C_i 、および現在時刻 t における残り最悪実行時間 c_{rem_i} をタスク情報としてもつ。 r_i および d_i は絶対時刻であり、 r_i はタスクの実行完了時に、 d_i はタスクのリリース毎にそれぞれ P_i を加算した値に更新される。対象システムでは EDF スケジューリング [10] による優先度割り当てを採用することとし、 $d_i \leq d_{i+1} (i = 1, 2, \dots, I-1)$ を満たす。全タスクの周期の最小公倍数であるハイパーピリオドを *HyperPeriod* とおく。タスク τ_i の負荷 u_i は $u_i = \frac{C_i}{P_i}$ 、タスクセットの負荷 U は $U = \sum_{i=1}^I u_i$ と定義される。

対象システムのプロセッサコアは J 個の設定可能な動作周波数および $K+1$ 個のコア状態をもつ。コア状態は、1 個の動作状態および K 個の動作状態以外のコア状態からなるとする。動作状態 ($k=0$) における設定可能な動作周波数をそれぞれ $f_1 < \dots < f_j < \dots < f_J$ とし、各周波数に対する動作時消費電力を $RPW_1 < \dots < RPW_j < \dots < RPW_J$ とする。なお、動作周波数および供給電圧の設定変更にかかるオーバーヘッドは無視できるものとする。また、各コア状態 ($k \neq 0$) におけるアイドル時消費電力を $IPW_1 < \dots < IPW_k < \dots < IPW_K$ とする。各コアへの状態遷移および動作状態への復帰には時間および消費エネルギーのオーバーヘッドが掛かるものとし、それぞれ TO_k および EO_k とする。なお、実行すべきタスクがなく、かつ動作状態である場合、動作状態は設定可能な最低動作周波数 f_1 で動作するものとする。以上の定義より、動作周波数 f_j およびコア状態 k におけるプロセッサコアの総消費電力 $PW_{j,k}$ は、以下の式で表される。

$$PW_{j,k} = \begin{cases} RPW_j & (k=0) \\ IPW_k & (k \neq 0) \end{cases} \quad (1)$$

動作状態から別のコア状態に遷移する際の損益分岐時間 [11] について定義する。損益分岐時間とは、コア状態の遷移によって総消費エネルギーの削減が見込めるスラック時間の最小値である。動作周波数 f_j で動作する動作状態からコア状態 $k (\neq 0)$ へ遷移して復帰する際の損益分岐時間 $BET_{j,k}$ は、以下の式で表される。

$$BET_{j,k} = \max\left[\frac{EO_k - IPW_k * TO_k}{RPW_j - IPW_k}, TO_k\right] \quad (2)$$

3.2 既存手法 look ahead EDF (laEDF) とその問題点

laEDF [1] は、タスクのリリース時およびディスパッチ時にアルゴリズムを実行し、動作周波数を導出する DVFS である。Algorithm 1 に laEDF のアルゴリズムを示す。laEDF では、ある区間の実行容量をその区間の時間と最高周波数の積、タスクの仕事量を実行時間と実行される周波数の積と定義する。そして各タスクの最悪時の仕事量を、

Algorithm 1 $\text{cal_laEDF_freq}(t, T)$

Input: 現在時刻 t , U , d_i , u_i , c_rem_i

Output: laEDF が導出する周波数 f_{laEDF}

```

1:  $U' \leftarrow U$ 
2:  $s \leftarrow 0$ 
3: for  $i = I$  to 1 do
4:    $U' \leftarrow U' - u_i$ 
5:    $x \leftarrow \max\{0, c\_rem_i - (1 - U')(d_i - d_1)\}$ 
6:    $U' \leftarrow U' + (c\_rem_i - x)/(d_i - d_1)$ 
7:    $s \leftarrow s + x$ 
8: end for
9:  $f_{laEDF} \leftarrow f_J * s/(d_1 - t)$ ;
```

直近のデッドライン時刻 d_1 から最低優先度タスクのデッドライン時刻 d_I までの実行容量にできるだけ予約する。これにより、タスクの仕事量のために予約される現在時刻から d_1 までの区間の実行容量が小さくなり、この区間における動作周波数を低く抑えることができる。

laEDF の問題点を示す。まず、コアの動作周波数について、laEDF では、スラック時間を過剰に見積もってしまう問題点がある。これにより、動作周波数が必要以上に引き上がることがあり、動作時消費エネルギーを増大させてしまう。さらに、laEDF ではコア状態の遷移に関する影響を考慮していない。

4. 提案手法

4.1 全体像

提案手法は、Slack Gathering laEDF (SGlaEDF) および Core State Aware Scheduling (CSAS) からなる。SGlaEDF では、デッドライン制約を保証した上でスラック時間を正確に導出する。CSAS では、動作時消費エネルギーおよびアイドル時消費エネルギーを同時に考慮することで、総消費エネルギーを削減する。提案手法では、SGlaEDF により導出される正確なスラック時間およびデッドライン制約を保証する動作周波数を用いて CSAS を実行することで、リアルタイムシステムのデッドライン制約を保証しつつ総消費エネルギーの最小化を目指す。

4.2 Slack Gathering laEDF (SGlaEDF)

本節では、正確なスラック時間およびデッドライン制約を満たす最低動作周波数を導出する DVFS 手法である SGlaEDF を提案する。本手法は、laEDF においてスラック時間を過剰に見積もる問題点が改善されており、各タスクをより適切な動作周波数で実行できるようになる。

スラック時間を過剰に見積もってしまう原因は、実行対象のタスクの実行可能時間が、現在時刻から直近のデッドラインまでの区間を超えてしまう場合があるためである。これは、直近のデッドラインがすでに実行完了したタスクの

デッドラインである場合に生じる。現在時刻 t において、実行対象のタスクとして $\tau_i (\neq \tau_1)$ が選ばれ、直近のデッドライン d_1 まで実行され、次に時刻 $t' (= d_1)$ において τ_1 のリリース処理が行われた結果、 τ_i が直近のデッドラインとなると、再び τ_i が実行されることになる。すなわち実行対象のタスクの実行可能時間が、現在時刻から直近のデッドラインまでの区間を超えてしまうことになる。なお、この問題は、直近のデッドラインが、まだ実行完了していないタスクのものである場合には発生しない。なぜなら、EDF アルゴリズムにおいて実行対象のタスクは、まだ実行完了していない直近のデッドラインのタスクとなるためである。この時、デッドライン制約により、タスクの実行可能時間は直近のデッドラインを超えることはない。

以上の点を踏まえ、SGlaEDF では、周期タスクの性質を利用し、タスクの実行完了時に次のデッドラインおよび残り最悪実行時間を更新する。これにより、現在時刻から直近のデッドラインの区間内、実行対象のタスクの実行可能時間を必ず含めることができるようになる。つまり、デッドライン d_i は、SGlaEDF では以下の式で導出される。

$$d_i = \begin{cases} d_i + P_i & (c_rem = 0) \\ d_i & (c_rem \neq 0) \end{cases} \quad (3)$$

この時、ある現在時刻 t における直近のデッドライン d_1 は、現時点でリリースされていないタスクも含めてまだ実行完了していない、最も優先度の高いタスクのデッドラインである。ここで、デッドライン制約から τ_1 は遅くとも時刻 $d_1 - c_rem_1$ から処理を行わなければならない。つまり、 τ_i の実行可能時間が理論上取りうる最大の区間は $[t, d_1 - c_rem_1]$ となり、これは現在時刻から直近のデッドラインまでの区間 $[t, d_1]$ に含まれる。よって τ_i の実行可能時間は必ず現在時刻から直近のデッドラインまでの区間に含まれる。

以上により、laEDF と同様に各隣接デッドライン間の実行容量に対してタスクの仕事量をできるだけ予約する。これにより、デッドライン制約を保証する範囲で、実行対象のタスクの実行可能時間における、必要最低動作周波数を導出することができる。

図 1 は、SGlaEDF によって動作時消費エネルギーを削減できる例である。時刻 t において τ_2 は実行対象のタスクとし、 τ_1 はすでに実行完了したタスクであるとする。図 1(a) および図 1(b) は laEDF で実行した場合であり、図 1(c) は SGlaEDF で実行した場合である。図 1(a) では、laEDF により区間 $[t, d_1]$ において動作周波数 $f_{laEDF} (= 0)$ が導出され、区間 $[t, d_1]$ はスラック時間となる。そして図 1(b) では、時刻 $t' (= d_1)$ において τ_1 が新たにリリースされ、タスク情報およびデッドラインが更新される。その結果、再び τ_2 が実行対象のタスクとなり、区間 $[t', d_2]$ において今度は大きく引き上がった動作周波数 f'_{laEDF} で実行される。つ

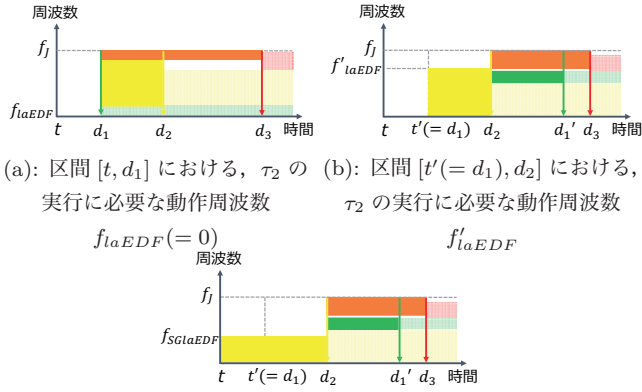


図 1 SGlaEDF によって動作時消費エネルギーを削減できる例

まり, laEDF が過剰にスラック時間を見積もったために大きな周波数変動を招き, 結果として動作時消費エネルギーが増大する. これに対して図 1(c) では, タスク τ_2 の実行可能時間を含んだ区間 $[t, d_2]$ について考えることができる. そこで, $[d_1, d_2]$ の実行容量を区間 $[t, d_1]$ に分配し, f'_{laEDF} より低い一定の動作周波数 $f_{SGlaEDF}$ で実行できる. これにより, 動作時消費エネルギーの削減が可能となる.

4.3 Core State Aware Scheduling (CSAS)

本節では, コア状態を考慮して総消費エネルギーの削減を図るスケジューリング手法 CSAS を提案する. 本手法は, 動作周波数を上昇させることによる, 動作時消費エネルギーの増大効果およびアイドル時消費エネルギーの削減効果を比較する. これによって, 総消費エネルギーを最小化するための適切な動作周波数およびコア状態を決定する. なお, 本手法は, laEDF においてアイドル時消費エネルギーを考慮していない問題点を解消している.

アイドル時消費エネルギーを考慮するため, 3.1 節の損益分岐時間 $BET_{j,k}$ およびタスクのスラック時間 $st_{i,j}$ を用いる. 動作周波数 f_1 で動作する動作状態からコア状態 $k(k \neq 0)$ へ遷移する際の損益分岐時間 $BET_{1,k}$ との間に, $st_{i,j} > BET_{1,k}$ が成立すれば, コア状態 k に遷移することでアイドル時消費エネルギーを削減できる. さらに CSAS では, 動作周波数を上昇させることによる, 動作時消費エネルギー増大効果およびアイドル時消費エネルギー削減効果を比較し, 適切な動作周波数およびコア状態を決定する.

図 2 は, CSAS によって総消費エネルギーの削減が可能となる例である. ここで, 説明を簡単化するため, プロセッサコアは動作状態と休止状態の 2 つのコア状態を持つとする. あるタスク τ_i を動作周波数 f_j で実行した時の, 実行時間およびスラック時間をそれぞれ $et_{i,j}$ および $st_{i,j} (< BET_{1,1})$ とする. τ_i を f_j より 1 つ上の設定可能な動作周波数 f_{j+1} で実行した時の, 実行時間およびスラック時間をそれぞれ $et_{i,j+1}$ および $st_{i,j+1} (\geq BET_{1,1})$ とする. 以上の条件において, それぞれの動作周波数における消

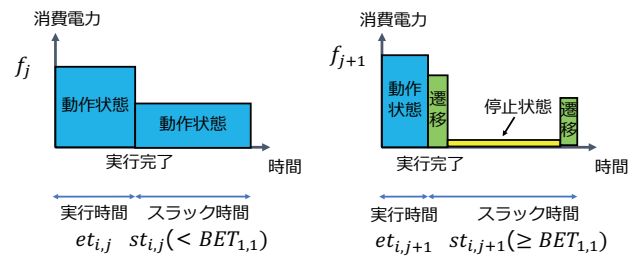


図 2 CSAS によって総消費エネルギーを削減できる例

費エネルギーを表したのが図 2(a) および図 2(b) である. 図 2(a) では, $st_{i,j} < BET_{1,1}$ より, スラック時間について動作状態を保ったままになる. それに対し図 2(b) では, $st_{i,j+1} \geq BET_{1,1}$ より, スラック時間について休止状態に遷移することができる. つまり, タスク τ_i の実行における動作時消費エネルギーは増大するが, アイドル時消費エネルギーは減少する. すなわち, アイドル時消費エネルギー減少量が動作時消費エネルギー増大量を上回れば, 結果として総消費エネルギーを削減できる.

Algorithm 2 は CSAS のアルゴリズムを示している. 1-2 行目にて, 時刻 t において, 実行するタスク τ_i よりも高優先度のタスク群の中で最も早いリリース時刻 $r_{highpri}$ および τ_i よりも低優先度のタスク群の中で最も早いリリース時刻 r_{lowpri} を求める. 5-6 行目にて, $et_{i,j}$ および $st_{i,j}$ を, 実行時間における動作周波数 f_j , $r_{highpri}$ および $r_{highpri}$ から求める. 次に 7-15 行目にて, f_j およびスラック時間における遷移先のコア状態 $st_{i,j,k}$ の組み合わせにおける, τ_i の実行可能時間の消費エネルギー $energy_{i,j,k}$ を求める. $st_{i,j,k}$ については, 損益分岐時間 $BET_{1,k}$ を用いて, コア状態 k に遷移するか否かを決定する. 以上の動作を, 全てのコア状態および f_a 以上の設定可能な動作周波数の組み合わせについて行い, $energy_{i,j,k}$ が最小となる動作周波数 f_{CSAS} およびコア状態遷移 k_{CSAS} を解として導出する.

4.4 スケジューラにおける手続き

Algorithm 3 は, 提案手法を実現するための, タスクスケジューラにおける手続きを示している. それぞれ 1-8 行目はシステム動作開始時の, 9-12 行目および 13-16 行目はタスクがリリースまたはディスパッチされた時の, 15-18 行目はタスクが実行完了した時の手続きである. タスクが実行完了した時以外の手続きについて, まず SGlaEDF により, デッドライン制約を保証する最低動作周波数 $f_{SGlaEDF}$ を求める. 次に, CSAS により, タスクの実行時間における動作周波数 f_{CSAS} およびスラック時間における状態遷移 k_{CSAS} を決定する. CSAS の設定可能な最低動作周波数 f_a を, $f_{SGlaEDF}$ 以上の動作周波数に限定することで, デッドライン制約は保証される. また, システム動作開始時には, 全タスクのデッドラインおよび残り最悪実行時間

Algorithm 2 cal_CSAS_freq(f, t, T)

Input: 現在時刻 t , r_i , d_i , c_{rem_i} , f_a

Output: 動作周波数 f_{CSAS} およびコア状態遷移 k_{CSAS}

```

1:  $r_{highpri} \leftarrow \{\min(r_1, r_2, \dots, r_{i-1}) | d_1 \leq d_2 \leq \dots \leq d_{i-1}\}$ 
2:  $r_{lowpri} \leftarrow \{\min(r_{i+1}, r_{i+2}, \dots, r_I) | d_{i+1} \leq d_{i+2} \leq \dots \leq d_I\}$ 
3: for  $k = 1$  to  $K$  do
4:   for  $j = a$  to  $J$  do
5:      $et_{i,j} \leftarrow \min(c_{rem_i}/f_j, r_{highpri} - t)$ 
6:      $st_{i,j} \leftarrow \max(0, \min(r_{highpri}, r_{lowpri}, d_i) - t - c_{rem_i}/f_j)$ 
7:     if  $st_{i,j} \geq BET_{1,k}$  then
8:        $energy_{i,j,k} \leftarrow et_{i,j} * RPW_j + (st_{i,j} - TO_k) * IPW_k + EO_k$ 
9:        $state_{i,j,k} \leftarrow k$ 
10:    else
11:       $energy_{i,j,k} \leftarrow et_{i,j} * RPW_j + st_{i,j} * RPW_1$ 
12:       $state_{i,j,k} \leftarrow 0$ 
13:    end if
14:  end for
15: end for
16:  $energy_{i',j',k'} \leftarrow \min(energy_{i,j,k})$ 
17:  $f_{CSAS} = f_{j'}$ 
18:  $k_{CSAS} = state_{i',j',k'}$ 

```

を最初に更新する。提案手法では、タスクの実行可能時間における最適な動作周波数および状態遷移を同時に導出するので、タスクが実行完了した時には呼び出す必要はない。ただし SGlaEDF については、実行完了したタスクのデッドラインおよび残り最悪実行時間を更新する。

5. 評価

5.1 評価環境

提案手法の有効性を、開発したタスクスケジューリングシミュレータ上で評価した。

タスクセットは、入力されたパラメータをもとに生成されるランダムタスクセットを利用した。各タスクの起動周期は、1,5,10,20,50ms の中からランダムに選ばれる。また、タスクの最悪実行時間は、タスクセットの負荷 U および正規分布を利用して決定した。

評価対象のプロセッサコアは、ODROID-XU3 [12] に搭載されている Cortex-A15 [13] コアとした。本プロセッサコアでは、200MHz から 2000MHz まで 100MHz 刻みで動作周波数を設定できる。Cortex-A15 コアの動作状態における動作周波数およびそれに対応する動作時消費電力の値は、文献 [5] の、ODROID-XU3 上の Cortex-A15 コアの 1 コアのみを、各動作周波数設定値で動作させた時の消費電力計測値を用いた。なお、評価実験の簡略化のため、本研究ではプロセッサコアが持つコア状態は、動作状態を含めて 2 つとして行った。アイドル時状態であるコア状態 $k(\neq 0)$

Algorithm 3 Scheduling SGlaEDF + CSAS

```

1: procedure THE SYSTEM IS INITIALIZED
2:   for  $i = I$  to 1 do
3:      $c_{rem_i} \leftarrow C_i$ 
4:      $d_i \leftarrow P_i$ 
5:   end for
6:    $f_{SGlaEDF} = \text{cal\_SGlaEDF\_freq}(0, T)$ 
7:    $f_{CSAS}, k_{CSAS} = \text{cal\_CSAS\_freq}(f_{SGlaEDF}, 0, T)$ 
8: end procedure
9: procedure ONE OR MORE TASKS ARE RELEASED AT  $t$ 
10:   $f_{SGlaEDF} = \text{cal\_SGlaEDF\_freq}(t, T)$ 
11:   $f_{CSAS}, k_{CSAS} = \text{cal\_CSAS\_freq}(f_{SGlaEDF}, t, T)$ 
12: end procedure
13: procedure A TASK IS DISPATCHED AT  $t$ 
14:   $f_{SGlaEDF} = \text{cal\_SGlaEDF\_freq}(t, T)$ 
15:   $f_{CSAS}, k_{CSAS} = \text{cal\_CSAS\_freq}(f_{SGlaEDF}, t, T)$ 
16: end procedure
17: procedure THE EXECUTION OF TASK  $\tau_i$  IS COMPLETED
18:   $d_i \leftarrow d_i + P_i$ 
19:   $c_{rem_i} \leftarrow C_i$ 
20: end procedure

```

におけるアイドル時消費電力 IPW_k については、以下の式により設定した。

$$IPW_k = \frac{RPW_1}{10} \quad (4)$$

状態遷移時にかかる時間および消費エネルギーのオーバーヘッドである、 TO_k および EO_k については、それぞれ $600\mu s$ および $5 * IPW_k$ とした。アルゴリズムの実行に掛かる時間および消費エネルギーのオーバーヘッドは無視した。

評価対象のスケジューリング手法は、次の 4 つである。

- (1) laEDF
- (2) laEDF+CSAS
- (3) SGlaEDF
- (4) SGlaEDF+CSAS (提案手法)

本実験では、タスクセットの自動生成のためのパラメータとして、タスクセットの負荷 U およびタスク数 I を利用した。 U は 0.1 から 0.9 まで 0.2 刻みとし、各 U に対して $I = 10, 30, 60, 90$ として合計 20 通りの組み合わせを実験した。実験では laEDF と他のタスクスケジューリングの有効性を比較するため、手法 (1) の消費エネルギーによって、他の手法の消費エネルギーを正規化した。

5.2 結果および考察

図 3 に、各タスクセットのハイパーピリオドにおける評価結果を示す。図中の縦軸は laEDF を基準として正規化した消費エネルギー、横軸はタスク数をそれぞれ示している。評価の結果、提案手法は、全ての入力パラメータに対して高い消費エネルギー削減効果が得られ、既存手法 laEDF

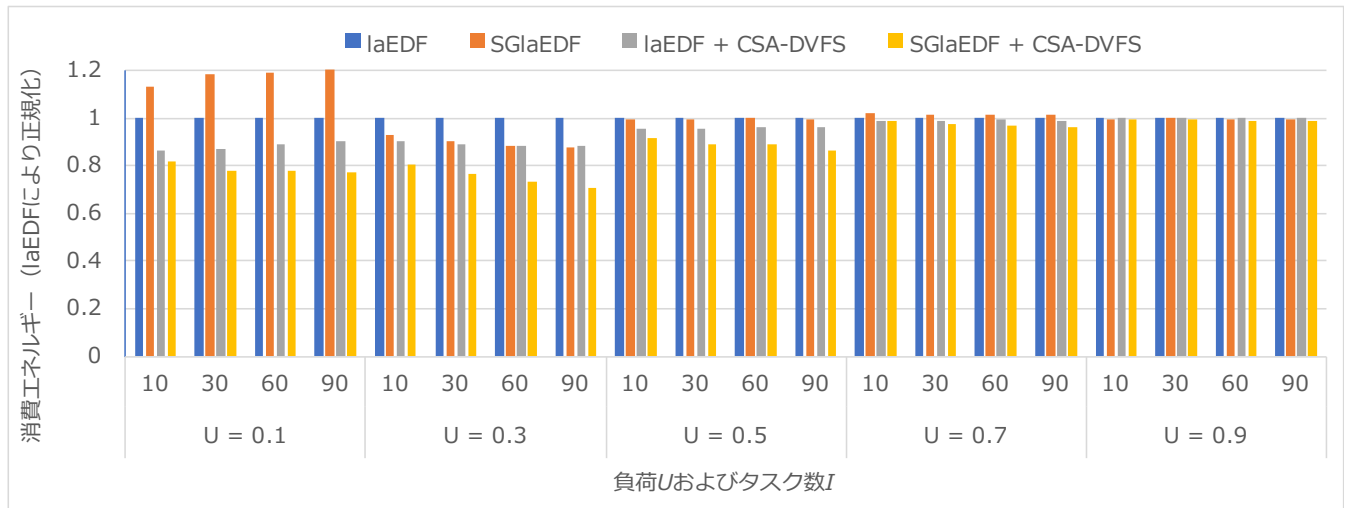


図 3 各 U における評価結果

と比較すると最大で 29%, 平均で 12% の消費エネルギーを削減することができた。

SGlaEDF を単独で適用した場合、消費エネルギー削減効果が認められたのは $U = 0.3$ の場合のみであった。 U が低い場合は、laEDF が過剰に見積もったスラック時間におけるアイドル時消費エネルギー削減効果が、SGlaEDF の動作時消費エネルギー削減効果を上回るためであると考えられる。 U が高い場合は、タスクスケジューリングにおいて生じるスラック時間が減ることで、動作時消費エネルギーを削減する余地が小さくなるためであると考えられる。

提案手法 SGlaEDF+CSAS は、全ての入力パラメータに対して、laEDF+CSAS よりも消費エネルギー削減効果が高かった。特に、タスク数が多い時に提案手法の優位性が高くなった。これは、laEDF+CSAS では、タスク数が多い場合には、実行可能時間が直近のデッドラインを超える回数が増加するためであると考えられる。

提案手法は、全ての入力パラメータに対して、最も消費エネルギー削減効果が高かった。したがって、提案手法の有効性が示された。

6. おわりに

本研究では、複数のコア状態を持つ単一コアプロセッサのための、消費エネルギー最小化のためのタスクスケジューリング手法を提案した。提案手法は Slack Gathering laEDF (SGlaEDF) および Core State Aware Scheduling (CSAS) からなり、SGlaEDF および CSAS によって、デッドライン制約を保証しつつ総消費エネルギーを削減する。提案手法は、 U が低い場合およびタスク数が多い場合に、特に高い削減効果が見られた。今後の課題として、実際のシステム上での検証およびアルゴリズムの実行にかかるオーバーヘッドを考慮することが挙げられる。

謝辞 本研究は JSPS 科研費 JP26870303 の助成を受け

たものである。

参考文献

- [1] Pillai, P. and Shin, K. G.: Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems, *ACM SIGOPS Oper. Syst. Rev.*, pp. 89–102 (2001).
- [2] 林和宏, 並木美太郎: EDF スケジューリングにおける DVFS を用いた CPU 省電力化手法, 情報処理学会研究報告, Vol. 2010-OS-113, No. 15, pp. 1–8 (2010).
- [3] Kim, W. et al.: Dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis, *Proc. of DATE*, p. 788 (2002).
- [4] Bang, S. Y. et al.: Run-Time Adaptive Workload Estimation for Dynamic Voltage Scaling, *IEEE Trans. on CAD*, Vol. 28, No. 9, pp. 1334–1347 (2009).
- [5] 岩田淳, 他: 同一命令セットヘテロジニアスマルチコアのための消費エネルギーを削減するタスクスケジューリング, 情報処理学会研究報告, Vol. 2015-EMB-36, No. 33, pp. 1–6 (2015).
- [6] Baynes, K. et al.: The performance and energy consumption of embedded real-time operating systems, *IEEE Trans. on Comp.*, Vol. 52, No. 11, pp. 1454–1469 (2003).
- [7] Awan, M. A. and Petters, S. M.: Energy-aware partitioning of tasks onto a heterogeneous multi-core platform, *Proc. of RTCSA*, pp. 205–214 (2013).
- [8] Diao, Q. and Song, J.: Prediction of CPU idle-busy activity pattern, *Proc. of HPCA*, pp. 27–36 (2008).
- [9] Zhan, X. et al.: CARB: A C-State Power Management Arbiter For Latency-Critical Workloads, *IEEE CAL*, No. 99 (2016).
- [10] Liu, C. L. and Layland, J. W.: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, *Journal of ACM*, Vol. 20, No. 1, pp. 46–61 (1973).
- [11] Lu, Y. H. and De Micheli, G.: Comparing System-Level Power Management Policies, *IEEE Design & Test of Computers*, Vol. 18, No. 2, pp. 10–19 (2001).
- [12] Hardkernel: ODROID-XU3, http://www.hardkernel.com/main/products/prdt_info.php?g_code=g140448267127.
- [13] ARM: Cortex-A15, <https://www.arm.com/ja/products/processors/cortex-a/cortex-a15.php>.