Vol.2018-ARC-230 No.29
Vol.2018-SLDM-183 No.29
Vol.2018-EMB-47 No.29
2018/3/8

(m, k)-firm制約下の弱ハードリアルタイムシステムにおける動的電圧周波数制御

水野 有美[1,a)]　高瀬 英希[1]　高木 一義[1]　高木 直史[1]

概要：本研究では，組込みリアルタイムシステムにおける消費エネルギー削減のためのタスクスケジューリング手法を提案する．対象は，各タスクについて連続する k 個のジョブのうち m 個を必須ジョブとしてリアルタイム性を保証する必要のあるモデルである，(m, k)-firm 制約に基づく弱ハードリアルタイムシステムである．提案手法は，まず第 1 フェーズにて，与えられたサービス品質要件を満たすように，各タスクで静的なジョブ分配を行う．次に第 2 フェーズでは，分配されたジョブパターンに対して，選択ジョブのスラック時間を活用する動的電圧周波数制御を適用する．評価の結果，ハードリアルタイムシステムと比較して，提案手法は平均 35%，最大 55%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開発では，システムの高性能化・複雑化による，消費エネルギーの増加が課題となっている．そこで，システムのサービス品質（QoS）を維持し，かつ消費エネルギーを抑えるアプローチが注目されている．消費エネルギーを削減する代表的な手法として，動的電圧周波数制御（Dynamic Voltage and Frequency Scaling, DVFS）が挙げられる．DVFS は，プロセッサ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，消費エネルギーを削減する技術である．

DVFS を含め，組込みシステムの消費エネルギー削減を目的としたリアルタイムスケジューリングに関するこれまでの研究のほとんどは，全てのタスクの実行をデッドライン内に完了しなければならない，ハードリアルタイムシステムに焦点を当てている．しかしながら，映像処理に代表されるような実用的なシステムでは，ある一定の割合についてその処理のリアルタイム性を確保できれば，要求されるシステムの QoS を満たすことができるシステムも多く存在する．このようなシステムを弱ハードリアルタイムシステムという．

本研究では，(m, k)-firm モデル [5] に基づく弱ハードリアルシステムを対象とする．(m, k)-firm モデルで定義される (m, k)-firm 制約とは，周期タスクの連続する k 個のジョブのうち，少なくとも m 個はデッドライン制約を満たす必要があることを意味する．システム内の各タスクに対し，ユーザーがあらかじめ 2 つのパラメータ m, k を与えることにより，システムの QoS 要件を柔軟に指定することが可能となる．

本研究の目的は，(m, k)-firm 制約下の弱ハードリアルタイムシステムの消費エネルギーを削減することである．そのために，2 つのフェーズから成るタスクスケジューリング手法を提案する．第 1 フェーズでは，各タスクに与えられた (m, k)-firm 制約のパラメータから，システムの QoS 要件を満たすように，必須ジョブと選択ジョブを適切に静的分配する．第 2 フェーズでは，動的な電圧周波数制御を行う．DVFS アルゴリズムとしては，高い消費エネルギー削減効果が示されている look ahead EDF（laEDF）[9] を用いる．ハードリアルタイムシステム向けのアルゴリズムである laEDF を，弱ハードリアルタイムシステム向けに拡張し，第 1 フェーズのジョブ分配によって定めたジョブパターンに適用する．これにより，ユーザの所望する QoS 要件を保証しつつ，消費エネルギーの最小化を図る．

2. 準備

2.1 弱ハードリアルタイムシステム

リアルタイムシステムは，デッドライン制約の厳密さにより，ハードリアルタイムシステム，ソフトリアルタイムシステム，および，弱ハードリアルタイムシステムの 3 つに分類される．ハードリアルタイムシステムでは，デッドライン内に全てのタスクの終了が保証されねばならない．それに対し，ソフトリアルタイムシステムでは，デッドラインを超えるとシステムの価値が緩やかに減少する．多くの場合，ハードリアルタイムタスクとソフトリアルタイム

[1] 京都大学
[a)] emb@lab3.kuis.kyoto-u.ac.jp

Vol.2018-ARC-230 No.29
Vol.2018-SLDM-183 No.29
Vol.2018-EMB-47 No.29
2018/3/8

タスクはシステム内において共存する．このようなシステムを，弱ハードリアルタイムシステムという．文献 [2] では，弱ハードリアルタイムシステムを，ある一定時間内に，デッドライン以内に完了したタスクとデッドラインミスを起こしたタスクが正確に分散している (precisely distributed) システムであると定義している．

リアルタイムシステムのスケジューリングにおいては，デッドラインミスの発生頻度によってシステムへの影響が異なる場合がある．一般には，あるデッドラインミスと，次に発生するデッドラインミスとの間隔が大きくなればなるほど，システムへの影響は小さくなる．

2.2 (m, k)-firm モデル

弱ハードリアルタイムシステムのモデルのひとつである (m, k)-firm モデル [5] は，デッドラインミスの頻度，つまり，システムの QoS 要件を定義するのに優れている．システム内の各タスクは，連続する k 個のジョブのうち，少なくとも m 個のジョブがデッドライン以内に実行を完了させる必要がある．これらのパラメータは，タスク毎にあらかじめ定めることができ，システムの望ましい QoS 要件として与えられる．このとき，タスクは (m, k)-firm 制約を有すると言う．連続する k 個のジョブのうち，リアルタイム性の保証が求められる m 個を必須ジョブ，デッドラインミスが許容される $(k-m)$ 個を選択ジョブと呼ぶ．

(m, k)-firm モデルの概念は一般性を有する．例えば，ハードリアルタイムシステムは，各タスクが (1, 1)-firm 制約を持つと定義される．また，(m, k)-firm 制約は，適切な間隔で必須ジョブが分配されていることに加え，$\frac{k-m}{k}$ が許容最大損失率を意味することと成る．例えば，(4, 5)-firm 制約は，許容最大損失率が 20%であることを意味する．

2.3 関連研究

文献 [3]，[10] は，単一プロセッサで動作するマルチタスクシステムを対象に，動作時にハードウェアをアクティブ状態やスリープ状態に移行させる動的電力管理（Dynamic Power Management, DPM）を，弱ハードリアルタイムシステムに対して適用するスケジューリングアルゴリズムを提案している．文献 [8] では，(m, k)-firm モデルを用いてジョブ分配を行なった後，ジョブの実行時刻を動的に変化させ，コアのスラック時間を最大化させている．スラック時間は選択ジョブの実行に利用され，システムの QoS 向上を目指している．文献 [7] では，ジョブ分配を行なった後，静的な電圧周波数制御を行っている．システム内の各タスクは，全てのジョブがデッドラインミスせずに実行できる最低周波数で，時刻によらず一定に動作する．この手法では，システムの負荷が時刻により異なる場合，ある時刻において必要以上に高い周波数で動作してしまう場合がある．

3. 対象システムのモデル化

本研究の対象システムを説明する．タスクセットは，EDF スケジューリングに従って実行されるものとする．つまり，全タスクは周期タスクであり，各タスクの周期と相対デッドラインは等しいものとする．また，タスクの実行は独立しており，プリエンプションによるタスク切り替えがあるものとする．各タスクの最悪実行時間は既知であり，どのリリース時でも一定であるとする．また、各タスクの実行にかかる時間は，最悪実行時間と等しいとする．対象システムのプロセッサコアは，有限個である離散値の動作周波数，および，それに対応した供給電圧に設定できる．また，タスクを実行していない場合はスリープ状態になるものとする．あるプロセッサコアにおけるタスクの実行時間は，その動作周波数に反比例するとする．ある周波数設定値におけるプロセッサコアの消費電力は，時刻や実行するタスクによらず一定であるとする．プリエンプション，DVFS，スリープ状態への遷移およびスリープ状態からの復帰にかかるオーバヘッド時間は無視する．

以上の説明をふまえ，対象システムを数理的にモデル化する．システムアーキテクチャであるプロセッサは単一であり，設定可能な動作周波数を $F_0, F_1, ..., F_{n-1} (F_0 > F_1 > ... > F_{n-1})$ と定義する．タスクセット T は，I 個のタスク $\tau_0, \tau_1, \ldots, \tau_{I-1}$ から構成される．タスク τ_i $(i = 0, 1, \ldots, I-1)$ の周期は P_i，相対デッドラインは D_i，最悪実行時間は C_i とする．周期と相対デッドラインは等しいものとするため，$P_i = D_i$ である．また，タスク τ_i はデッドライン時刻 d_i をもつ．ここで，d_i は絶対的な時刻であり，タスクのリリース毎に P_i を加算した値に更新される．各タスク τ_i の最悪実行時間 C_i は，タスクをコアプロセッサの最高周波数 F_0 で実行する場合に，実行完了までにかかる時間とする．タスク τ_i の添字 i は，その時点で優先度が高いものほど小さい値となる．

対象システムでは，EDF スケジューリングによる優先度割り当てを採用しており，$d_i \leq d_{i+1}$ $(i = 0, 1, \ldots, I-2)$ を満たす．つまり，タスクのリリース毎に，d_i が更新され，その値によって，τ_i の順番が入れ替わる．アルゴリズムの説明の都合上，あるタスクの添字 i はタスクを識別する値ではなく，時間とともに変動することに注意されたい．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 $U = \sum_{i=0}^{I-1} u_i$ と定義される．

各タスクは，2.2 節で述べた (m, k)-firm 制約に関するパラメータ m_i, k_i を持つ．タスク τ_i の連続する k_i 個のジョブのうち，少 m_i 個が必須ジョブであり，デッドライン制約を満たす必要がある．タスク τ_i の j 番目のジョブを $\tau_{i,j}$ と表すことにする．ここで，添字 j は，動的優先度により変動する i と従属性があるものとする．

Vol.2018-ARC-230 No.29
Vol.2018-SLDM-183 No.29
Vol.2018-EMB-47 No.29
2018/3/8

4. タスクスケジューリング手法

4.1 全体像

提案するスケジューリング手法は，2 つのフェーズから構成される．

第 1 フェーズでは，静的なジョブ分配を行う．2.2 節で説明した (m, k)-firm 制約の 2 つのパラメータ m, k により，各タスクの QoS 要件が指定される．これを満たすように，タスクごとのジョブを必須ジョブと選択ジョブに静的に分配する．本研究では，既存研究 [3][10][7] で提案されている 3 種類の分配方式を利用するものとする．第 1 フェーズにより，各タスクの QoS 要件が保証される．

第 2 フェーズでは，動的な電圧周波数制御を行う．本研究が参考とする laEDF[9] は，ハードリアルタイムシステム向けのアルゴリズムである．laEDF は，スラック時間を利用して動作電圧を下げるため，スラック時間を拡大することにより更なる消費エネルギーの削減効果が期待できる．本研究では，laEDF を弱ハードリアルタイムシステム向けに拡張したアルゴリズムを提案する．実行するジョブを第 1 フェーズで決定した必須ジョブに限定し，残りの時間をスラック時間として利用する．

4.2 必須ジョブと選択ジョブの静的分配

提案手法では，まず，タスクキュー内で，各タスクの必須ジョブと選択ジョブを静的に分配する．タスク τ_i の必須ジョブのパターンを，バイナリ文字列 $\Pi_i = \pi_{i,0}\pi_{i,1}...\pi_{i,(k_i-1)}$ で表す．ここで，$\pi_{i,j}$ は τ_i の j 番目のジョブが必須ジョブであるか選択ジョブであるかを表すバイナリ変数である．$\pi_{i,j}$ が 1 のときは $\tau_{i,j}$ が必須ジョブであることを，0 のときは選択ジョブであることを意味する．Π_i によって，分配結果が (m, k)-firm 制約を満たすジョブパターンであるかの解析が容易となる [4].

本研究では，文献 [3][10][7] で消費エネルギー削減に有効だと示されているジョブ分配方式である，R-pattern，E-pattern，および，ER-pattern を用いる．**表 1** は，(m, k) が (1, 2), (2, 5) および (3, 7) における各分配パターンの結果である．

それぞれの分配方式について説明する．1 つ目は R-pattern あるいは deeply-red pattern と呼ばれる方式である [3].

表 1 提案手法による静的なジョブ分配の適用例

(m, k)	(1, 2)	(2, 5)	(3, 7)
R-pattern	101010...	1100011000...	111000011...
E-pattern	101010...	1010010100...	101010010...
ER-pattern	010101...	0010100101...	001010100...

$$\pi_{i,j} = \begin{cases} 1 & 0 \le j \bmod k_i < m_i \\ 0 & \text{otherwise} \end{cases} \quad j = 0, 1, ..., k_i - 1 \tag{1}$$

R-pattern では，各タスク τ_i の m_i 個の必須ジョブをスケジューリング開始時から順に実行していくため，スケジューリング可能性はハードリアルタイムシステムと同様となる．これは，必須ジョブが R-pattern に従ってスケジューリング可能である限り，任意の動的スケジューリングにおいて必須ジョブの実行を保証することができることを意味する．しかし，必然的に，各タスク τ_i の最初の m_i 個のジョブ実行時に負荷が集中するため，スケジューリング可能性は過度に悲観的になる．

2 つ目は E-pattern あるいは evenly-distributed pattern と呼ばれる方式である [10].

$$\pi_{i,j} = \begin{cases} 1 & if\ j = \lfloor \lceil \frac{j*m_i}{k_i} \rceil * \frac{k_i}{m_i} \rfloor \\ 0 & \text{otherwise} \end{cases} \quad j = 0, 1, ..., k_i - 1 \tag{2}$$

E-pattern でのスケジューリングが可能であるための必要十分条件は，$\lceil x \rceil^+$ を

$$\lceil x \rceil^+ = 1 + \lfloor x \rfloor \tag{3}$$

と定義すると，任意の時刻 t において，

$$\sum_i (\frac{m_i}{k_i} * \lceil \frac{t - D_i}{Pi} \rceil^+) * C_i \le t \tag{4}$$

と表せる．m_i および k_i がともに 1 の場合，式 (4) がハードリアルタイムシステムにおける EDF スケジューリング可能性の必要十分条件の式と等しいこと，そして，弱ハードリアルタイムシステムでは $\frac{m_i}{k_i} \le 1$ が成立することを考慮すると， E-pattern のスケジューリング可能性は R-pattern よりも優れている，または等しいことが言える（式 (4)）の証明を含む， E-pattern のスケジューリング可能性に関する詳細は，文献 [7] を参照されたい）.

E-pattern では，最初のジョブは必ず必須ジョブである．E-pattern を，水平方向に反転させた方式は以下の式 (5) で表される．ER-pattern あるいは　reverse-evenly-distriuted pattern と呼ばれている [7].

$$\pi_{i,j} = \begin{cases} 0 & if\ j = \lfloor \lceil \frac{j*(k_i - m_i)}{k_i} \rceil * \frac{k_i}{k_i - m_i} \rfloor \\ 1 & \text{otherwise} \end{cases} \quad j = 0, 1, ..., k_i - 1 \tag{5}$$

ER-pattern のスケジューリング可能性は，E-pattern と等しい．つまり， E-pattern でスケジューリング可能なタスクセットは常に ER-pattern でもスケジューリング可能である．更に， ER-pattern では，ある区間における必須ジョブの個数は常に E-pattern 以下であることが保証される．

Vol.2018-ARC-230 No.29
Vol.2018-SLDM-183 No.29
Vol.2018-EMB-47 No.29
2018/3/8

4.3 動的電圧周波数制御の適用

第 2 フェーズでは，第 1 フェーズで定めた必須ジョブのみに対して，スラック時間を活用した DVFS を適用する．DVFS アルゴリズムとしては，EDF を採用したシステムにおいて高い消費エネルギー削減効果がある，laEDF を用いる [9]．

laEDF は，後続の周波数を最大で実行すると仮定して，デッドライン制約を保証する範囲で現在の周波数を出来るだけ引き下げ，消費エネルギーの削減を狙うアルゴリズムである．アルゴリズムの説明に際し，ある区間の実行容量を，その区間の時間と最高周波数を乗算した値と定義し，タスクの仕事量を実行時間と実行される周波数を乗算した値と定義する．laEDF では，各タスクの最悪時の仕事量を，直近のデッドライン時刻 d_0 から最低優先度タスクのデッドライン時刻 d_{I-1} までの実行容量に出来るだけ予約する．そうすることで，タスクの仕事量のために予約される現在時刻から d_0 までの区間の実行容量が小さくなる．このため，laEDF によって，d_0 以前の周波数を低く抑えることができる．

laEDF 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，導出した動作周波数によりタスクを実行する．入力パラメータは，アルゴリズムが実行される時刻 t，そのとき実行されるタスクセット T およびその全タスクの情報である．また，出力は，laEDF で導出される周波数 F_{laEDF} である．アルゴリズムの詳細は，文献 [9] を参照されたい．

本研究では，laEDF を弱ハードリアルタイムシステムに適用できるように拡張したアルゴリズム（以下，laEDF* と記述する）を提案する．Algorithm 1 に，laEDF* の手続きを示す．

laEDF* では laEDF の入力パラメータに加え，変数 $\pi_{i,j}$ を導入する．$\pi_{i,j}$ はブール代数であり，定義は 4.2 と同様である．アルゴリズムの呼び出し時に，それが必須ジョブであるか選択ジョブであるかを判断する処理を追加する．$\pi_{i,j} = 1$ のときは，$\tau_{i,j}$ の仕事量を予約し，$\pi_{i,j} = 0$ のときは，$\tau_{i,j}$ の残り実行時間 c_rem_i を 0 とする．その場合，選択ジョブのために予約されていた仕事量が 0 となり，その区間の実行容量がスラック時間となることで，より小さな実行周波数を出力することが期待できる．

Algorithm 1 laEDF*

Input: 現在時刻 t，タスク $\tau_i \in T$ の残り最悪実行時間 c_rem_i，d_i，u_i，タスクセット T の U，及び $\pi_{i,j}$
Output: $F_{laEDF'}$

```
U' ⇐ U
s ⇐ 0
for i = I − 1 to 0 do
    U' ⇐ U' − u_i
    if π_{i,j} = 1 then
        x ⇐ max{0, c_rem_i − (1 − U')/(d_i − d_0)}
        U' ⇐ U' + (c_rem_i − x)/(d_i − d_0)
        s ⇐ s + x
    else
        c_rem_i ⇐ 0
    end if
end for
F_{laEDF'} ⇐ F_0 · s/(d_0 − t)
```

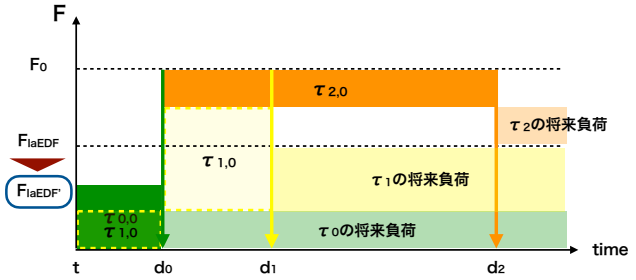

図 1　laEDF を弱ハードリアルタイムシステムに適用した例（$\tau_{1,0}$ が選択ジョブ，$\tau_{0,0}$, $\tau_{2,0}$ が必須ジョブの場合）

図 1 は提案手法の適用例を示している．図 1 において，$\tau_{1,0}$ のジョブが選択ジョブであるとする．このとき，時刻 d_0 以前では予約されている $\tau_{1,0}$ の実行容量が 0 となり，そこへ $\tau_{0,0}$, $\tau_{2,0}$ の実行容量を予約できるようになる．その結果，従来の laEDF から算出した動作周波数よりも小さな周波数が算出され，消費エネルギーの削減が可能となる．

ハードリアルタイムシステムを対象とする場合は，タスクの属性 $(m,\ k) = (1,\ 1)$ とすれば良い．各タスクの全てのジョブが必須ジョブであるため，拡張前の laEDF アルゴリズムとなる．

5. 評価

5.1 評価環境

設定できる動作周波数，および，それに対応する消費エネルギーのパラメータは， Cortex-A15 コアアーキテクチャ [1] を採用した Exynos 5422 を搭載する ODROID-XU3[6] から取得した．また，スリープ時の消費電力は無視できることとした．

タスクセットは，入力されたパラメータをもとに自動的に生成されるランダムタスクセットを利用する．すなわち，各タスク τ_i のパラメータ P_i，D_i，C_i，m_i および k_i は，決められた範囲内でランダムに設定される．

提案手法の評価のために，タスクシミュレータを Ruby で自作した．シミュレータは，時刻 0 より，タスクセットのハイパーピリオド（全タスクの周期の最小公倍数）に各タスク τ_i のパラメータ k_i の最小公倍数を乗じた時刻までのタスクスケジューリングをシミュレーションする．これは，以降の時刻におけるタスクスケジューリングは，それ以前のタスクスケジューリングの繰り返しとなるからである．なお，laEDF アルゴリズムの実行にかかる時間および消費エネルギーのオーバヘッドは無視するものとする．シ

Vol.2018-ARC-230 No.29
Vol.2018-SLDM-183 No.29
Vol.2018-EMB-47 No.29
2018/3/8

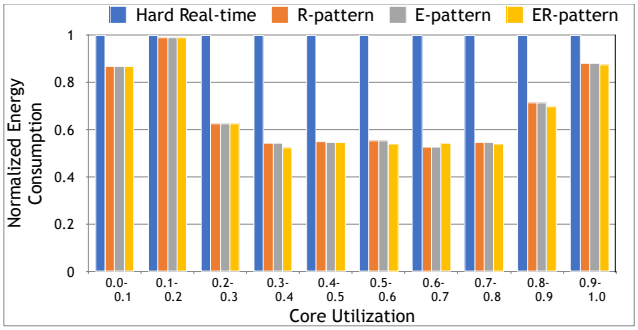

図 2 コアの使用率を変化させた場合の消費エネルギー評価結果

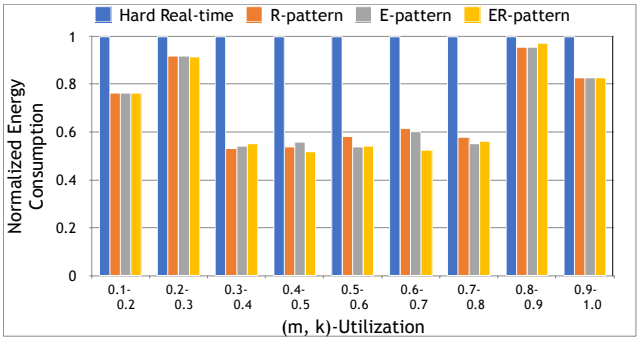

図 3 (m, k)-比を変化させた場合の消費エネルギー評価結果

ミュレーションの後，それまでにかかった消費エネルギーが算出される．

3 種類のジョブ分配に基づく DVFS 手法の有効性を評価するため，全タスクを $(m,k)=(1,1)$ としたハードリアルタイムシステムに対して laEDF[9] を適用する手法を導入する．提案手法の適用により算出される消費エネルギーは，本手法による値で正規化したものを求める．

5.2 結果

図 2 および**図 3** に，評価結果を示す．図 2 は，タスク数 I=5，$(m,\ k)=(2,\ 3)$ に固定し，コアの使用率を変化させた場合の評価結果である．$\tau_i(P_i, D_i, C_i, m_i, k_i)$ のパラメータについては，10ms $\leq P_i \leq$50ms，1ms $\leq C_i \leq P_i$ でランダムに決定した．3 は，3 つのタスク $\tau_0(8,\ 2,\ 2,\ m_0,\ k_0)$, $\tau_1(10,\ 3,\ 3,\ \ m_1,\ k_1)$, $\tau_2(12,\ 3,\ 3,\ m_2,\ k_2)$ を用いて，各タスクの $(m,\ k)$ を，$1 \leq k_i \leq 10,\ 1 \leq m_i \leq k_i$ の範囲でランダムに決定した場合の評価結果である．このとき，コアの使用率は $\frac{2}{8}+\frac{3}{10}+\frac{3}{12}=0.8$ で一定である．図中の縦軸は，いずれも各入力値の場合の消費エネルギーを，$(m,\ k)=(1,\ 1)$ の場合の消費エネルギーで正規化した値である．横軸は，図 2 ではコアの使用率を，図 3 では $\frac{m_i}{k_i}$（以下，(m, k)-比）を示している．

評価の結果，全ての入力パラメータに対して，各分配パターンにおける消費エネルギーは $(m,\ k)=(1,\ 1)$ の場合の消費エネルギー以下であることが確認できた．コアの使用率を変化させた場合は，最大で 55%，平均で 35%の消費エネルギーを削減することができた．(m, k)-比を変化させた場合は，最大で 48%，平均で 31%の消費エネルギーを削減することができた．ジョブの分配パターンによる消費エネルギーの違いは，(m, k)-比が 0.6-0.7 のとき最大で，8.9%であった．このとき，ER-pattern を用いた分配方式における消費エネルギーが最も小さかった．一方，他の (m, k)-比では，R-pattern や E-pattern における消費エネルギーが最も小さい場合も確認された．

5.3 考察

コアの使用率を変化させた場合，提案手法は，実験した全てのコア使用率において消費エネルギー削減効果があることが確認できた．コア利用率が 0.6-0.7 のとき，最も大きな効果が確認できた．コア使用率が低くなる，あるいは高くなるにつれ，提案手法の有効性は低減した．コア使用率が 0.2 以下である場合，$(m,\ k)=(1,\ 1)$ に対し laEDF が算出する周波数は，その他の $(m,\ k)$ に対する周波数と等しくなる可能性が高い．これは，コアの使用率が低い場合，全ての時刻においてコアのスラック時間が十分に残り，$(m,\ k)=(1,\ 1)$ に対し laEDF が決定する周波数が小さくなるためであると考えられる．また，コアの使用率が 0.8 を超えても，laEDF が算出する周波数と，静的に決定される周波数が等しくなる可能性が高い．これは，各時刻におけるスラック時間が小さくなり，laEDF が算出する周波数が，いずれの場合も最大周波数に近づくためであると考えられる．

(m, k)-比を変化させた場合も，全体としてはコアの使用率を変化させた場合と同じ傾向が見られた．(m, k)-比に比例しコアの使用率が大きくなることから，結果に至るまでのプロセスも同様であることが推測できる．

一方，同じ (m, k)-比を持ちながら，$(m,\ k)$ の値が異なるとき，各分配パターンにより消費エネルギーが異なる場合があった．**図 4** は，$(m,\ k)=(1,\ 3)$ と $(2,\ 6)$ における消費エネルギーを評価したものである．(m, k)-比は，いずれの場合も $\frac{1}{3}$ だが，$(m,\ k)=(2,\ 6)$ では，ER-pattern が R-pattern よりも 14%消費エネルギーが小さいことが確認された．理由としては，コア内のジョブの偏りが関係していることが予想される．

図 5 は，$(m,\ k)=(2,\ 6)$ における，R-pattern および ER-pattern の，コア内の実行状況の一部である．R-pattern では，必須ジョブを前から順に連続して実行するため，スケジューリングの開始時にコアの競合が生じやすく，その結果 laEDF が出力する動作周波数が必要以上に高くなることが考えられる．今回の結果では R-pattern よりも ER-pattern の消費エネルギーが削減されたが，laEDF を行う際，より有効なジョブ分配パターンを与えるには，ある時間区間における必須ジョブの偏りを考慮する必要がある．具体的には，必須ジョブの間隔を動的に出来るだけ均

Vol.2018-ARC-230 No.29
Vol.2018-SLDM-183 No.29
Vol.2018-EMB-47 No.29
2018/3/8

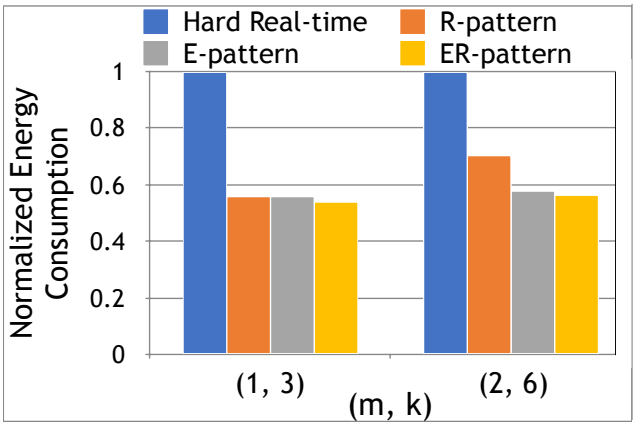

図 4 $(m,\ k) = (1,\ 3), (2,\ 6)$ における消費エネルギー評価結果

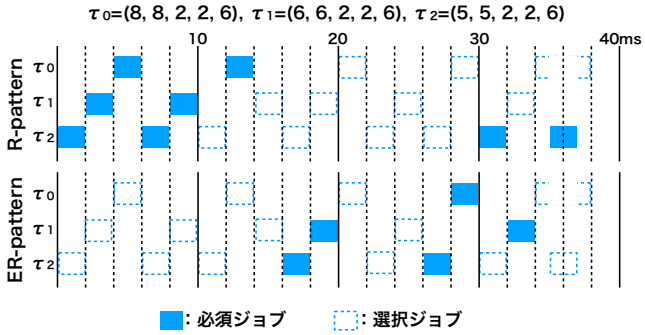

図 5 $(m,\ k) = (2,\ 6)$ における R-pattern および ER-pattern のタスクの実行状況

等に配置させることが，消費エネルギー削減に有効であると考えられる．

6. おわりに

本研究では，(m, k)-firm 制約を有する弱ハードリアルタイムシステム向けに，QoS 要件を保証し，かつ消費エネルギー削減を実現するスケジューリングアルゴリズムを提案した．提案手法は，静的なジョブ分配，および，動的な電圧周波数制御の 2 つのフェーズからなる．評価の結果，コアの使用率を変化させた場合は最大 55%，平均 35%の消費エネルギーを削減することができた．(m, k)-比を変化させた場合は，最大で 48%，平均で 31%の消費エネルギーを削減することができた．

今後の課題としては，DVFS により適したジョブ分配パターンを考案すること挙げられる．今回の研究では既存の静的ジョブ分配手法を用いたが，コアをより低周波数で動作させるには，ある時間区間に，どの程度ジョブが密集しているかを考慮することが求められる．

参考文献

[1] arm Developer: Cortex-A15 (online), https://developer.arm.com/products/processors/cortex-a/cortex-a15. (2018.02.04).
[2] Bernat, G., et al.: Weakly Hard Real-Time Systems, *IEEE Trans. on Computers*, Vol. 50, No. 4, pp. 308-321 (2001).
[3] Koren, G. and Shasha, D.: Skip-Over: algorityms and complexity for overloaded systems that allow skips, *Proc. of RTSS*, pp. 110–117 (1995).
[4] Quan, G. and Hu, X.: Enhanced fixed-priority scheduling with (m, k)-firm guarantee, *Proc. of RTSS*, pp. 79–88 (2000).
[5] Hamdaoiu, M. and Ramanathan, P.: A dynamic priority assignment technique for streams with (m, k)-firm deadlines, *IEEE Trans. on Computers*, Vol. 44, No. 12, pp. 1443-1451 (1995).
[6] Hardkernel co., Ltd,: ODROID-XU3 (online), http://www.hardkernel.com/main/products/prdt_info.php?g_code=g140448267127. (2018.02.04).
[7] Niu, L. and Quan, G.: Energy minimization for real-time systems with (m,k)-guarantee, *IEEE Trans. on VLSI Systems*, Vol. 14, No. 7, pp. 717-729 (2006).
[8] Niu, L. and Quan, G.: System-wide dynamic power management for portable multimedia devices, *Proc. of ISM*, pp. 97–104 (2006).
[9] Padmanabhan, P. and Shin, K. G.: Real-time dynamic voltage scaling for low-power embedded operating systems, *ACM SIGOPS Operating Systems Review*, Vol. 35, No. 5, pp. 89-102 (2001).
[10] Ramanathan, P.: Overload management in real-time control applications using (m, k)-firm guarantee, *IEEE Trans. on Parallel and Distributed Systems*, Vol. 10, No. 6, pp. 549-559 (1999).