Vol.2010-SLDM-144 No.60
Vol.2010-EMB-16 No.60
Vol.2010-MBL-53 No.60
Vol.2010-UBI-25 No.60
2010/3/28

優先度付きSMTプロセッサ向け実時間動的電圧周波数制御

藤井　啓[†1]　千代浩之[†2]　山崎信行[†2]

現代の組込みリアルタイムシステムにはリアルタイム性だけでなく，高スループットや低消費電力が要求される．高スループットは SMT や CMP などの高並列なアーキテクチャで実現し，低消費電力は Dynamic Voltage Frequency Scaling（DVFS）で実現することが主流となっている．本論文では優先度付き SMT プロセッサを対象に，シンプルかつ効果的にプロセッサの消費電力を削減する Hetero Efficiency to Logical Processors（HeLP）を提案する．また，HeLP にテンポラルマイグレーション手法を応用することでさらなる消費電力の削減を目指した HeLP-Temporal Migration（TM）も提案する．シミュレーションによる評価では，HeLP を用いて消費電力量を効果的に削減できることを示す．また，HeLP-TM は HeLP よりも消費電力量を削減できることを示す．

Real-Time Dynamic Voltage and Frequency Scaling for Prioritized SMT Processors

KEI FUJII,[†1] HIROYUKI CHISHIRO[†2] and NOBUYUKI YAMASAKI[†2]

Current embedded and real-time systems require not only real-time capabilities but also high throughput and low power consumption. High throughput is mainly achieved by parallel architectures such as SMT and CMP, and low power consumption is mainly achieved by Dynamic Voltage and Frequency Scaling (DVFS). In this paper, we present a DVFS algorithm called Hetero Efficiency to Logical Processor (HeLP) which can reduce power consumption simply and effectively in prioritized SMT processors. We also present Hetero Efficiency to Logical Processor with Temporal Migration (HeLP-TM) which applies the temporal migration technique to HeLP. Simulation results show that HeLP can reduce power consumption effectively and HeLP-TM is more effective than HeLP.

1. はじめに

現代の組込みリアルタイムシステムは，リアルタイム性だけでなく，高スループットと低消費電力を実現する必要がある．これは，携帯電話やノートパソコン，デジタルカメラ，バッテリー駆動ロボットなどに次々と高度な機能が搭載され，システムに対する処理要求量が増加しているためである．これらの機器はバッテリー駆動するものが多く，利用可能な消費電力量はバッテリー容量分に限られる．そのため，システムを長時間利用するためには，システムの単位時間当たりの消費電力量を低く抑えることが要求される．これらのシステムにおいてエネルギーを最も消費する要素はプロセッサであり，プロセッサの消費電力量はシステム全体の消費電力量のうち，大きな割合を占める．現在のプロセッサの多くは CMOS 回路により形成されており，プロセッサの消費電力量は動作周波数と動作電圧の 2 乗に比例する[1] ことから，動作周波数と動作電圧を抑えることで消費電力量を大幅に削減することができる．スループットを向上する手段としては動作周波数を上げることが考えられるが，動作周波数を上げると消費電力量も増加するため単純に動作周波数を上げることはできない．そのため，近年では Simultaneous Multithreading（SMT）[2] や Chip Multiprocessor（CMP）[3] 等の高並列アーキテクチャが主流となりつつある．特に SMT は 1 つのコアにおいて複数スレッドでハードウェア資源を共有することで CMP より資源の利用効率を上げることができるため，低い動作周波数で高いスループットを実現できる．したがって消費電力量の観点からも SMT は有効である．また，プロセッサは常に最大利用率で稼働しているわけではない．処理要求が集中して利用率が最大となっている時もあれば，要求が少なくアイドル状態となっている時もある．この時，オペレーティングシステムによってアプリケーションが実行可能な必要最低限の動作周波数及び動作電圧でプロセッサを稼働させることで，無駄な消費電力量を削減することが可能である．このように，動的に動作電圧及び動作周波数を変化させる手法を Dynamic Voltage Frequency Scaling（DVFS）と呼ぶ．しかしながら，動作周波数を低くするとアプリケーションの実行速度が低下するため，リアルタイムシステムにおいてはリアルタイム性を保証可能な範囲で DVFS を行わなければならない．これを Real-Time DVFS（RT-DVFS）という．このような作業はすべてオペレーティングシステム

†1 慶應義塾大学理工学部情報工学科
Department of Information and Computer Science, Faculty of Science and Technology, Keio University
†2 慶應義塾大学大学院理工学研究科開放環境科学専攻
Department of Computer Science, Graduate School of Science and Technology, Keio University

Vol.2010-SLDM-144 No.60
Vol.2010-EMB-16 No.60
Vol.2010-MBL-53 No.60
Vol.2010-UBI-25 No.60
2010/3/28

の行う仕事であり，組込みリアルタイムシステムにおいて重要な役割を果たす．

SMT では同時に実行するタスクの組み合わせにより実行時間が変動するため，リアルタイム性の保証が困難である．そこで本論文では，リアルタイム処理で用いる優先度を導入した優先度付き SMT の特徴を生かした RT-DVFS 手法を提案する．提案手法は，優先度と実行効率をスレッド毎に固定し，予めスレッド毎に割り当てられたタスクを実行しているスレッドの中で，一番優先度が高いスレッドの実行効率に合わせて動作周波数を決定する．また，マイグレーション機能を追加することで，さらなる消費電力量の削減を目指す手法も提案する．

本論文の構成は次のとおりである．第 2 章では，関連研究とその問題点について述べる．第 3 章では，本論文が対象とするシステムモデルについて述べる．第 4 章では，優先度付き SMT 向けの RT-DVFS 手法を提案する．第 5 章では，提案手法，およびアルゴリズムの性能をシミュレーションにより評価する．最後に，第 6 章で本論文の結論と今後の課題を述べる．

2. 背景及び関連研究

RT-DVFS 手法として，Pillai と Shin は，動的優先度スケジューリングである Earliest Deadline First（EDF）[4] 向けの Cycle-Conserving（CC）と Look-Ahead（LA）アルゴリズムを提案した[5]．CC アルゴリズムは，タスクの実際の実行時間が最悪実行時間よりも短くなることを利用し，次のタスクの動作周波数を下げる手法である．LA アルゴリズムは，すべてのタスクのリアルタイム性を保証することが可能な最低動作周波数を算出する手法である．Zhu らは，EDF 向けのフィードバック制御による RT-DVFS 手法を提案した[6]．このアルゴリズムは，フィードバック制御を用いて実行時間を予測し，それにより生じる余裕時間を用いて，動作周波数を算出する手法である．

SMT でのタスク実行例を**図 1** に示す．SMT では 1 つのコアで計算資源を有効活用することで同時にタスクを実行するが，同時に実行するタスクの組合わせによって各タスクの実行時間が変動する．そのため，各タスクの実行時間の見積もりが困難である．また，スレッド毎に動作周波数を設定することができない．RT-DVFS を行うためには，全スレッドのリアルタイム性を保証した上で動作周波数を下げる必要があるが，実行時間の見積もりが困難であるため，SMT において RT-DVFS を行うことは困難である．

我々はリアルタイムシステムにおける優先度の概念を SMT に応用し，優先度付き SMT 機構を持つ RMT Processor[7] を開発している．RMT プロセッサはスレッド毎に優先度を設

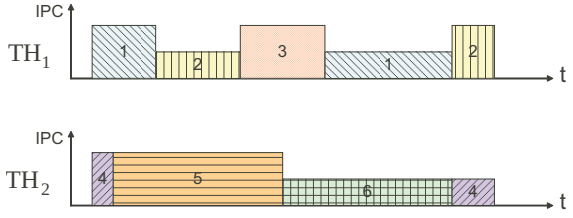

図 1 SMT での実行例
Fig. 1 Example of SMT execution.

定でき，スレッド間で資源の競合が発生した際には，優先度にしたがってスレッドに資源が割り当てられる．さらに，我々は RMT Processor において，スレッドの実行速度を制御可能にする Instruction Per Clock（IPC）制御機構を提案した[8]．これにより，各スレッドの IPC を制御することが可能となり，優先度付き SMT プロセッサにおいてタスクの実行時間の見積もりを行うことができる．

本論文では，シンプルに動作周波数を決定することが可能な優先度付き SMT プロセッサ向け動的電圧周波数制御手法を提案する．また，Kato らが非周期タスクの応答性を向上させるために提案したテンポラルマイグレーション手法[9] を応用することで，さらなる消費電力量の削減を目指す手法を提案する．

3. システムモデル

本論文では，優先度付き SMT プロセッサである RMT Procressor におけるパーティショニング方式[10] の EDF スケジューリングを想定する．

システム開始時に，n 個の周期タスクから構成されるタスクセット $\tau = \{\tau_1, \tau_2, \cdots, \tau_n\}$ が与えられ，システム実行中に新しいタスクの到着や消滅はないものとする．すべてのタスクは互いに独立に処理を行い，どの時点でもプリエンプション可能とする．ただし，如何なるタスクも複数のスレッドにおいて同時に実行できないものとする．各タスク τ_i は (C_i, T_i) のタプルで定義され，C_i はシングルスレッド実行時における最悪実行時間，T_i は周期を指す．τ_i の CPU 利用率を $U_i = C_i/T_i$，タスクセット τ に含まれるタスクの合計 CPU 利用率を $U(\tau) = \Sigma_i U_i$ と定義する．すなわち，$U(\tau)$ はシステム全体の負荷を意味する．

本論文では，周期毎の実行単位をジョブと呼ぶ．タスクは一連のジョブを周期的に生成する．タスク τ_i の k 番目のジョブを $\tau_{i,k}$ と表し，$\tau_{i,k}$ は時刻 $r_{i,k}$ でリリースされ，デッドライン $d_{i,k}$ は次のジョブのリリース時刻とする．すなわち，$d_{i,k} = r_{i,k+1} = r_{i,k} + T_k$ となる．時刻

Vol.2010-SLDM-144 No.60
Vol.2010-EMB-16 No.60
Vol.2010-MBL-53 No.60
Vol.2010-UBI-25 No.60
2010/3/28

t において，ジョブ $\tau_{i,k}$ の最悪実行時間を基準とした残り実行時間を $R_{i,k}(t)$ とする．

本論文では，ハードウェアで同時実行可能なスレッドを Logical Processor（LP）と呼ぶことにする．システムには，M 個の $LP(LP_1,\cdots,LP_m\cdots,LP_M)$ が存在する．優先度付き SMT では，各 LP にハードウェアで優先度を付けることが可能であり，LP_m の優先度を p_m とする．本論文では LP の優先度は固定とし，システム実行中に付け替えることはない．優先度は $p_m > p_{m+1}$ を満たすように設定する．また，LP_m の実行効率を e_m とし，IPC を用いて以下のように定義する．

$$e_m = \frac{actual\ IPC\ of\ jobs\ on\ LP_m}{non_blocked\ IPC\ of\ jobs\ on\ LP_m} \tag{1}$$

LP の優先度は固定されているので，動作周波数が一定の場合は，その実行効率も一定であり，$e_m > e_{m+1}$ を満たすとする．本論文ではすべての LP の実行効率は IPC 制御機構により保証できるものと仮定する．また，今回は具体例として，LP 数が 3，実行効率が優先度の高い順に常に 100%，50%，25%となるようにハードウェア的に保証されているものを基本モデルとして扱う．

動作周波数 f_c は，LP の実行効率を用いて決定できるものとする．すなわち f_c は，最大動作周波数 f_{max} を用いて $f_c = f_{max} \times e_c$ で決定する．また，どの LP でもタスク実行を行っていないときには，動作周波数は 0 に設定できるとする．

動作周波数を下げるとシステム全体の IPC が小さくなるが，各 LP の実行効率を一定に維持するように，各 LP への IPC の割り当てをハードウェアで調整するものと仮定する．ただし，如何なる場合においても，LP に割り当てた IPC の総和がプロセッサのトータル IPC よりも小さくなっているものとする．実行効率は LP の実行速度比と考えることができ，タスク τ_i を LP_m で実行しているとき，τ_i を完了するために必要な最悪実行時間は C_i/e_m となる．

パーティショニング方式なので，与えられたタスクセットの各タスクの CPU 利用率を基に，各 LP の実行効率に合わせてタスクを割り当てる．m 番目の LP に割り当てられたタスクセットを Π_m とし，その LP の利用率は $U(\Pi_m) = \Sigma_{\tau_i \in \Pi_m} U_i$ となる．LP_m に割り当てられた i 番目のタスク τ_i を τ_i^m と表し，(C_i^m, T_i^m) のタプルで定義する．τ_i^m のシングルスレッド実行時の CPU 利用率を $U_i^m = C_i^m/T_i^m$ と表す．同様に，τ_i^m の k 番目のジョブを $\tau_{i,k}^m$ と表し，そのリリース時刻を $r_{i,k}^m$，デッドラインを $d_{i,k}^m$ と表す．時刻 t において，ジョブ $\tau_{i,k}^m$ の最悪実行時間を基準とした残り実行時間を $R_{i,k}^m(t)$ とする．また，タスクは実行可能状態になると，そのジョブがそれぞれの LP の Ready Queue（RQ）に格納される．各 LP で実行中のタスクは RQ の先頭にあるものとして扱う．

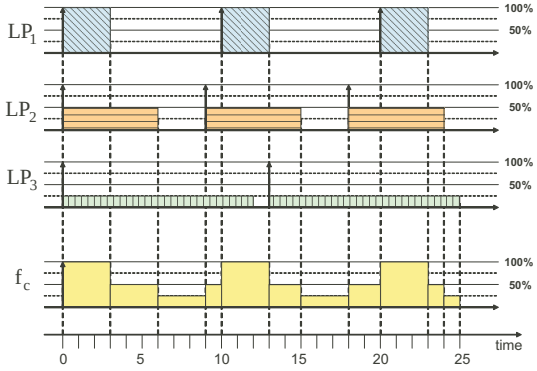

図 2 HeLP の実行例
Fig. 2 Example of HeLP execution.

表 1 タスクセット 1
Table 1 task set1.

LP	実行時間	周期
LP1	3	10
LP2	6	9
LP3	12	13

4. 動的電圧周波数制御アルゴリズム

本章では，LP 毎に優先度と実行効率を設定し，パーティショニング方式でスケジューリングを行い，タスクを実行しているスレッドの中で，一番優先度の高いスレッドの実行効率に合わせて動作周波数および動作電圧を決定する RT-DVFS 手法を提案する．また，テンポラルマイグレーション手法を応用することにより，さらなる消費電力の削減を目指す手法を提案する．

4.1 Hetero Efficiency to Logical Processor

Hetero Efficiency to Logical Processor（HeLP）は LP 毎に設定された実行効率を超えないようにタスク割り当てを行い，パーティショニング方式の EDF スケジューリングを行う．この時，HeLP のアルゴリズムは次のようになる．

(1) ジョブのリリースと完了時に各 LP のタスク実行を解析
(2) タスク実行を行っている LP の中で最も優先度の高い LP の実行効率に合わせて周波数と電圧をスケーリング

HeLP アルゴリズムを用いて**表 1** のタスクセットをスケジューリングした場合の実行例を**図 2** に示す．時刻 [0,3) ではすべての LP でタスクを実行しており，最も優先度の高い LP は LP_1 となっている．従って，この時刻での動作周波数は LP_1 の実行効率に合わせて f_{max} の 100%に設定される．時刻 [3,6) では，LP_2 と LP_3 でタスクを実行しており，タスクを実行し

Vol.2010-SLDM-144 No.60
Vol.2010-EMB-16 No.60
Vol.2010-MBL-53 No.60
Vol.2010-UBI-25 No.60
2010/3/28

ている LP の中で最も優先度の高い LP は LP_2 である．従って，この時刻での動作周波数は LP_2 の実行効率に合わせて f_{max} の 50%に設定される．時刻 [6,9) では，LP_3 だけでタスクを実行しており，タスクを実行している LP の中で最も優先度の高い LP は LP_3 である．従って，この時刻での動作周波数は LP_3 の実行効率に合わせて f_{max} の 25%に設定される．以下，同様にタスクを実行している LP の中で最も優先度の高い LP の実行効率に合わせて動作周波数の設定を行う．

4.2 Hetero Efficiency to Logical Processor with Temporal Migration

HeLP では最高優先度の LP でタスク実行を行っていないときに動作周波数を下げる．そのため，最高優先度の LP だけでタスク実行を行う場合には動作周波数制御を行わない場合と消費電力量は変わらない．リアルタイム性を保証した上で，優先度の高い LP のタスクをできるだけ優先度の低い LP に移動させて実行させれば，低い動作周波数で実行可能である．すなわち，消費電力をより削減することが可能となる．そこで，テンポラルマイグレーション手法を HeLP に応用した Hetero Efficiency to Logical Processor with Temporal Migration（HeLP-TM）を提案する．

4.2.1 HeLP-TM

HeLP-TM におけるテンポラルマイグレーションは Total Bandwidth Server（TBS）[11] アルゴリズムに基づいて行う．マイグレーションは，ジョブ単位で行われ，マイグレーションされるジョブは，マイグレーション先の LP で非周期タスクとして扱う．TBS アルゴリズムでは，非周期タスクが到着した際に，到着した非周期タスクに仮想デッドラインを割り当て，周期タスクと共に EDF でスケジューリングを行う．仮想デッドラインは，非周期タスク用の擬似的なデッドラインである．LP_x でサーバー τ_x^{srv} が実行されていたとする．サーバーの帯域幅は $U_x^{srv} = e_x - U(\Pi_x)$ である．LP_x に k 番目に到着した非周期タスクを $\alpha_{x,k}$，その到着時間を $a_{x,k}$，その実行時間を $E_{x,k}$ とすると，その仮想デッドライン $v_{x,k}$ は式（2）で与えられる．ここで，$v_{x,0} = 0$ とする．

$$v_{x,k} = max(a_{x,k}, v_{x,k-1}) + E_{x,k}/U_x^{srv} \tag{2}$$

テンポラルマイグレーションが発生するのは次の時である．

(1)　ジョブのリリース時

(2)　ジョブの完了時

時刻 t に LP_y においてジョブ $\tau_{i,l}^y$ がリリースされた，もしくは実行を完了したとする．LP_y では帯域幅 U_y^{srv} のサーバーが実行されているものとする．この時，以下の手続きに従って周期タスクの一時的なマイグレーションが発生する．

(1)　$\tau_{i,l}^y$ がリリースされた場合

(a)　LP_y のレディーキューの先頭から順番に，現在の周期でまだマイグレーションしていないジョブを調べ，そのようなジョブを $\tau_{j,m}$ とする．もし，このようなジョブが存在しなければテンポラルマイグレーションは発生せずに手続きは終了する．

(b)　式（3）を満たす LP_x を検索する．

$$max(t, v_{x,k}) + (R_{j,m}(t) \times e_y/e_x) \leq d_{j,m} \tag{3}$$

ただし，LP_x は LP_y よりも優先度の低い LP であるものとし，複数該当する場合はそのうちの最も優先度の低い LP とする．ここで，$v_{x,k}$ は時刻 t 以前に P_x に到着した最後の非周期タスク $\alpha_{x,k}$ の仮想デッドラインとする．このとき，$R_{j,m}(t)$ が e_y/e_x 倍されることに注意されたい．もし，このような LP が存在しないときは手順（a）に戻る．

(c)　$\tau_{j,m}$ を一時的に LP_x にマイグレーションする．ここで，$\tau_{j,m}$ は LP_x 上での $k+1$ 番目の非周期タスクとして扱われ，式（4）で仮想デッドライン $v_{x,k+1}$ が割り当てられる．

$$v_{x,k+1} = max(t, v_{x,k}) + (R_{j,m}(t) \times e_y/e_x)/U_x^{srv} \tag{4}$$

(2)　$\tau_{i,l}^y$ が実行を完了した場合

(a)　LP_y よりも優先度の高い LP のうち高い方から順に調べていく．LP の ready queue の先頭から順番に，現在の周期でまだマイグレーションしていないジョブを調べ，そのようなジョブを $\tau_{j,m}$，このジョブがあった LP を LP_x とする．もし，このようなジョブが存在しなければテンポラルマイグレーションは発生せずに手続きは終了する．

(b)　式（5）を満たすかどうか判定する．ここで，$v_{y,k}$ は時刻 t 以前に P_y に到着した最後の非周期タスク $\alpha_{y,k}$ の仮想デッドラインとする．

$$max(t, v_{y,k}) + (R_{j,m}(t) \times e_x/e_y) \leq d_{j,m} \tag{5}$$

このとき，$R_{j,m}(t)$ が e_x/e_y 倍されることに注意されたい．もし，満たさないときは手順（a）に戻る．

(c)　$\tau_{j,m}$ を一時的に LP_y にマイグレーションする．ここで，$\tau_{j,m}$ は LP_y 上での $k+1$ 番目の非周期タスクとして扱われ，式（6）で仮想デッドライン $v_{y,k+1}$ が割り当てられる．

$$v_{y,k+1} = max(t, v_{y,k}) + (R_{j,m}(t) \times e_x/e_y)/U_y^{srv} \tag{6}$$

Vol.2010-SLDM-144 No.60
Vol.2010-EMB-16 No.60
Vol.2010-MBL-53 No.60
Vol.2010-UBI-25 No.60
2010/3/28

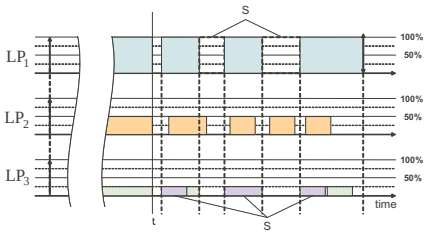

図 3 テンポラルマイグレーション前
Fig. 3 Before Temporal Migration.

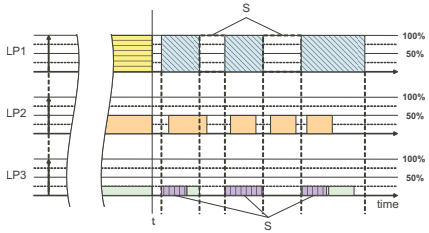

図 4 テンポラルマイグレーション後
Fig. 4 After Temporal Migration.

4.2.2 HeLP-TM-guarantee

HeLP-TM はヒューリスティックな手法だが，最高優先度の LP から最低優先度の LP へのマイグレーションだけに限定すれば，HeLP より必ず消費電力量を削減することが可能となる．**図 3**，**図 4** を用いて例示する．

消費電力量は，電圧の二乗とその電圧での演算数の乗算したものを全ての動作電圧に関して計算し，その総和により求める．タスク τ_i^1 の k 番目のジョブ $\tau_{i,k}^1$ の残り演算数を S とする．この時，時刻 t から $d_{i,k}^1$ までの間に，LP_1 以外の LP で周波数 f_{max} で実行した演算数の合計を D とする．ここで，$\tau_{i,k}^1$ を LP_1 から LP_3 へマイグレーションする場合を考える．$\tau_{i,k}^1$ をマイグレーションさせると実行時間は長くなるが，演算数は S のまま変わらない．そのため，$\tau_{i,k}^1$ をマイグレーションさせても，$\tau_{i,k}^1$ の実行を LP_3 で周波数 f_{max} で動作させた場合，$\tau_{i,k}^1$ の演算による消費電力量は変わらない．しかし，$\tau_{i,k}^1$ の演算による消費電力量は，$\tau_{i,k}^1$ をマイグレーションさせた場合（$P_{migration}^S$）にマイグレーションさせなかった場合（$P_{no\_migration}^S$）よりも大きくなることはない．すなわち次式のように表される．

$$P_{migration}^S \leq P_{no\_migration}^S \tag{7}$$

また，$\tau_{i,k}^1$ を LP_1 から LP_3 へマイグレーションしなかった場合に，時刻 t から $d_{i,k}^1$ までの時間に，LP_1 以外の LP で周波数 f_{max} で実行される演算は，$\tau_{i,k}^1$ をマイグレーションさせたことによって f_{max} よりも動作周波数を下げることができる．図 4 の D の部分の実行による消費電力量はマイグレーションさせた場合（$P_{migration}^D$）にマイグレーションさせなかった場合（$P_{no\_migration}^D$）を用いて次式で表される．

$$P_{migration}^D \leq P_{no\_migration}^D \tag{8}$$

したがって，式（7），式（8）より，タスク実行を行っている LP のうち，最高優先度の LP から最低優先度の LP へのタスクマイグレーションに限定した場合，HeLP よりも消費電力量を必ず削減できるといえる．

5. 評　　価

本章では提案した HeLP と HeLP-TM，HeLP-TM-guarantee についてシミュレーションにより評価を行った．

5.1 評価環境および方法

5.1.1 評 価 環 境

3 章で述べたシステムモデルに基づいて評価を行う．LP の数は 3 とし，優先度が高い方から順に LP_1, LP_2, LP_3 とする．IPC は優先度が高い LP から順に 1.0，0.5，0.25 となるようにハードウェアで保証されているものとする．そのため，実行効率も高い方から順に 100%，50%，25%となる．

U_{max} と U_{min}，U_{total} の 3 つのパラメータを用いてシミュレーションを行う．U_{max} はシステムに与えられるタスクセットに含まれるタスクの利用率の最大値とし，$[0.1, 1.0]$ の範囲から 0.1 刻みで設定する．U_{min} はシステムに与えられるタスクセットに含まれるタスクの利用率の最小値とし，0.01 および 0.26 のいずれかに設定する．U_{total} はタスクセットに含まれるタスクの利用率の合計とし，$[0.055, 1.75]$ の範囲から 0.05 刻みで設定する．

タスクの生成方法は以下の通りである．タスクの利用率は，$[U_{min}, U_{max}]$ の範囲で一様分布となるように 0.01 刻みで生成する．タスクの周期は，$[100, 3000]$ の範囲で一様分布となるように 1 刻みで生成する．シングルスレッド実行時の最悪実行時間は利用率と周期を決定すれば自動的に決まる．

また，スケジューラによるオーバーヘッド及びコンテキストスイッチにかかるオーバーヘッドはないものとして評価を行う．ここでは，動作周波数と動作電圧の変更は即座に行われるものと仮定する．また，スイッチング容量を c，タスクの単位時間当たりの演算数を k とし，それぞれ一定であると仮定する．**表 5.1.1** に，選択可能な動作周波数 f と，それを満たすのに必要な最低電圧 V を，それぞれ最大値との比で表す．

5.1.2 評 価 方 法

消費電力量に関する評価指標である PowerRatio を次のように求める．まず，LP_m において，動作周波数 f_j でタスクを実行した時間を $A_m(f_j)$，動作周波数 f_j に必要な最低電圧を

Vol.2010-SLDM-144 No.60
Vol.2010-EMB-16 No.60
Vol.2010-MBL-53 No.60
Vol.2010-UBI-25 No.60
2010/3/28

表 2 動作周波数と動作電圧
Table 2 frequency and Voltage.

f	V
0.25	0.83
0.50	0.90
1.00	1.00

$V(f_j)$ と定義する．これを用いると動作周波数 f_j における演算毎の消費電力量 E_j^{op} は次式となる．

$$E_j^{op} = cV(f_j)^2 \tag{9}$$

また，LP_m において動作周波数 f_j で実行された演算数 $N_{m,j}$ は，LP_m の実行効率 e_m を用いて次式となる．

$$N_{m,j} = e_m \times A_m(f_j) \tag{10}$$

式（9）と式（10）を用いて，システムの消費電力量 E_{sys} は次式となる．

$$E_{sys} = \sum_m \sum_j N_{m,j} \times E_j^{op} \tag{11}$$

同様の方法で比較を行うアルゴリズムを適用したときの消費電力量を計算したものを E_{cmp} とすると，PowerRatio は次式となる．

$$PowerRatio = E_{sys}/E_{cmp} \tag{12}$$

以下の評価ではこの PowerRatio を用いる．

5.2 消費電力量に関する評価

5.2.1 システム利用率を変化させた場合

$U_{max} = 0.5$，$U_{min} = 0.01$ に固定し，U_{total} を $[0.05, 1.70]$ の範囲から 0.05 刻みで変動させ，DVFS を行わなかった場合との比較である PowerRatio を**図 5** に示す．各プロットは 100 タスクセットの平均値である．システム利用率が $[0.05, 0.25]$ では 3 つの手法の PowerRatio が等しく一定となっている．これは，システム利用率が 25% までは 1 タスクの利用率が 25% よりも小さくなるため，必ず LP_3 にタスクが割り当てられるためである．システム利用率を増やしていくにつれて PowerRatio は no-scaling に近づいていく．これは，システム利用率が増えるに従って，高い動作電圧で動作する時間が増えるためである．しかし，どのような場合においても no-scaling よりも消費電力量を削減できていることがわかる．

次に図 5 と同じ条件で，HeLP アルゴリズムによる消費電力と比べた場合の PowerRatio を

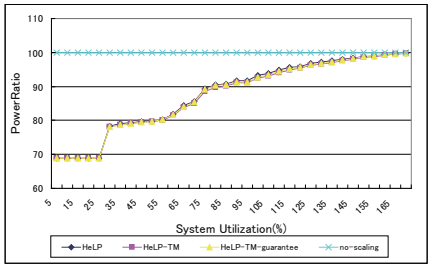

図 5 $U_{max} = 0.5$，$U_{min} = 0.01$
Fig. 5 $U_{max} = 0.5$，$U_{min} = 0.01$.

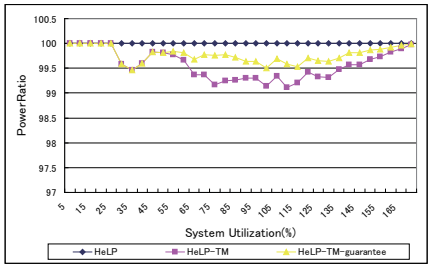

図 6 $U_{max} = 0.5$，$U_{min} = 0.01$
Fig. 6 $U_{max} = 0.5$，$U_{min} = 0.01$.

図 6 に示す．図 6 から分かるように，システム利用率がどんな場合でも HeLP-TM-guarantee は HeLP よりさらに消費電力量を削減できていることがわかる．また，HeLP-TM-guarantee より HeLP-TM の方が消費電力量を削減できていることから，マイグレーションにより消費電力量が増加するケース（4.2.2 節参照）の発生頻度は低いと考えられる．

5.2.2 1 タスクの最大利用率を変化させた場合

$U_{total} = 1.15$，$U_{min} = 0.01$ に固定し，U_{max} を $[0.1, 1.0]$ の範囲から 0.1 刻みで変動させ，DVFS を行わなかった場合との比較である PowerRatio を**図 7** に示す．1 タスクの最大 utilization を小さくすると，優先度の低い LP にタスクが割り当てられる可能性が高くなるため，消費電力量の削減率は大きくなる．

次に図 7 と同じ条件で，HeLP アルゴリズムによる消費電力と比べた場合の PowerRatio を**図 8** に示す．1 タスクの最大利用率が大きいほどタスクセットに含まれるタスク数は少

Vol.2010-SLDM-144 No.60
Vol.2010-EMB-16 No.60
Vol.2010-MBL-53 No.60
Vol.2010-UBI-25 No.60
2010/3/28

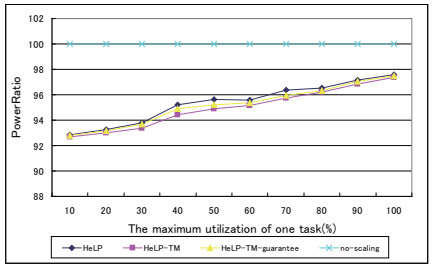

図 7 $U_{total} = 1.15$, $U_{min} = 0.01$
Fig. 7 $U_{total} = 1.15$, $U_{min} = 0.01$.

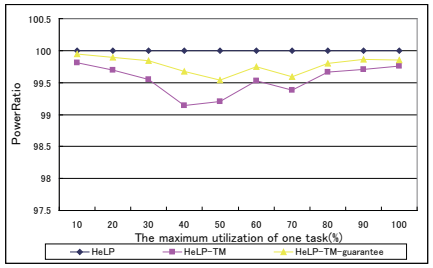

図 8 $U_{total} = 1.15$, $U_{min} = 0.01$
Fig. 8 $U_{total} = 1.15$, $U_{min} = 0.01$.

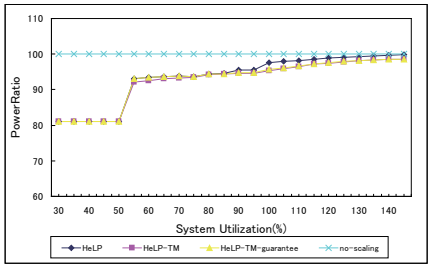

図 9 $U_{max} = 0.5$, $U_{min} = 0.26$
Fig. 9 $U_{max} = 0.5$, $U_{min} = 0.26$.

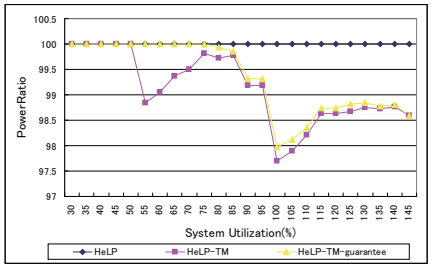

図 10 $U_{max} = 0.5$, $U_{min} = 0.26$
Fig. 10 $U_{max} = 0.5$, $U_{min} = 0.26$.

なくなり，小さいほどタスク数は多くなる．タスク数が少ない場合にはマイグレーションが起こりにくくなるため，HeLP-TM と HeLP-TM-guarantee の HeLP の消費電力量からのさらなる削減率は小さくなる．また，タスク数が多い場合には，優先度の低い LP にタスクが割り当てられる可能性が高くなり，低優先度 LP のサーバの帯域幅が少なくなるので，マイグレーションは起こりにくくなり，HeLP-TM と HeLP-TM-guarantee の HeLP の消費電力量からの削減率は小さくなる．1 タスクの最大利用率がどんな場合でも HeLP-TM-guarantee は HeLP よりさらに消費電力量を削減できていることがわかる．また，HeLP-TM-guarantee より HeLP-TM の方が消費電力量を削減できていることから，マイグレーションにより消費電力量が増加するケース（4.2.2 節参照）の発生頻度は 1 タスクの最大利用率によらず低いと考えられる．

5.2.3 1 タスクの最低利用率を高くした場合

$U_{max} = 0.5$, $U_{min} = 0.26$ に固定し，U_{total} を $[0.30, 1.45]$ の範囲から 0.05 刻みで変動させ，DVFS を行わなかった場合との比較である PowerRatio を**図 9** に示す． 図 9 と同じ条件で，HeLP アルゴリズムによる消費電力と比べた場合の PowerRatio を**図 10** に示す．$U_{total} = 1.15$, $U_{min} = 0.26$ に固定し，U_{max} を $[0.1, 1.0]$ の範囲から 0.1 刻みで変動させ，DVFS を行わなかった場合との比較である PowerRatio を**図 11** に示す． 図 11 と同じ条件で，HeLP アルゴリズムによる消費電力と比べた場合の PowerRatio を**図 12** に示す． この場合も no-scaling と比べて HeLP，HeLP-TM，HeLP-TM-guarantee のいずれも消費電力を削減できていることがわかる．$U_{min} = 0.01$ の時と比較すると，HeLP-TM と HeLP-TM-guarantee の消費電力量の削減率が向上している．これは，LP_3 にタスクが 1 つも割り当てられていないため，マイグレーションが起こる可能性が高いためである．すなわち，低優先度 LP にタスク割り当てが

Vol.2010-SLDM-144 No.60
Vol.2010-EMB-16 No.60
Vol.2010-MBL-53 No.60
Vol.2010-UBI-25 No.60
2010/3/28

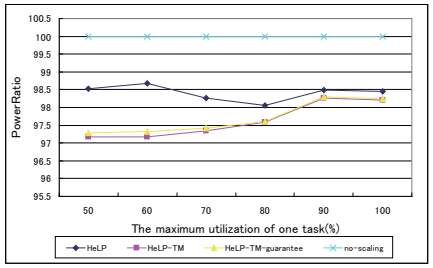

図 11 $U_{total} = 1.15$, $U_{min} = 0.26$
Fig. 11 $U_{total} = 1.15$, $U_{min} = 0.26$.

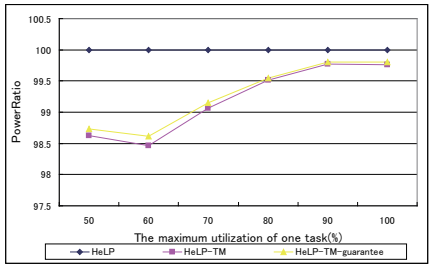

図 12 $U_{total} = 1.15$, $U_{min} = 0.26$
Fig. 12 $U_{total} = 1.15$, $U_{min} = 0.26$.

されていない時ほど HeLP よりもさらに消費電力量を削減できる．

6. 結　　論

本論文では，優先度付き SMT においてリアルタイム性を保証しながら省電力を実現する DVFS 手法として，Hetero Efficiency to Logical Processor（HeLP），HeLP-Temporal Migration（TM），HeLP-TM-guarantee を提案し，その有効性を評価した．シミュレーションによる評価では，HeLP は低優先度 LP にタスクが割り当てられている時ほど no-scaling と比較して消費電力量を削減でき，最大で消費電力量を約 30%削減できた．また，HeLP-TM は低優先度 LP にタスク割り当てがされていない時ほど HeLP と比較して消費電力量を削減でき，HeLP と比較して最大約 2.5%の消費電力量を削減できた．今回は最悪実行時間を用いてアルゴリズムを考えたが，実際のタスク実行は最悪実行時間よりも早く完了する．今後は実際の実行時間も考慮した消費電力量を削減できる手法を考えていく予定である．

謝辞　本研究は科学技術振興機構 CREST の支援によるものであることを記し，謝意を表す．また，本研究の一部は文部科学省グローバル COE プログラム「環境共生・安全システムデザインの先導拠点」に依るものであることを記し，謝意を表す．

参 考 文 献

1) Burd, T. and Brodersen, R.: Energy Efficient CMOS Microprocessor Design, *In Proc. of the 28th Annual Hawaii International Conference on System Sciences*, pp.288–297 (1995).
2) Tullsen, D.M., Eggers, S.J. and Levy, H.M.: Simultaneous Multithreading: Maximizing On-Chip Parallelism, *the 22nd Annual International Symposium on Computer Architecture*, pp. 392–403 (1995).
3) Olukotun, K., Nayfe, B., Hammond, L., Wilson, K. and Chang, K.: The Case for a Single-Chip Multiprocessor, *Architectual Support for Programming Language and Operating Systems* (1996).
4) Liu, C. and Layland, J.: Scheduling algorithms for multiprogramming in a hard real-time environment, *Journal of the ACM*, Vol.20, pp.46–61 (1973).
5) Pillai, P. and Shin, K.: Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems, *In Proc. of the ACM Symposium on Operating Systems Principles*, pp.89–102 (2001).
6) Zhu, Y. and Muller, F.: Feedback EDF scheduling exploiting dynamic voltage scaling, *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*, pp.84–93 (2004).
7) Yamasaki, N.: Responsive Link for Distributed Real-Time Processing, *Proceedings of the Innovative Architecture for Future Generation High-Performance Processors and Systems*, pp.20–29 (2007).
8) Yamasaki, N., Magaki, I. and Itou, T.: Prioritized SMT Architecture with IPC Control Method for Real-Time Processing, *Proceedings of the 13th IEEE Real Time and Embedded Technology and Applications Symposium*, pp.12–21 (2007).
9) Kato, S. and Yamasaki, N.: Scheduling Aperiodic Tasks using Total Bandwidth Server on Multiprocessors, *In Proc. The 6th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, pp.82–89 (2008).
10) Andersson, B. and Jonsson, J.: Fixed-Priority Preemptive Multiprocessor Scheduling: To Partition or not to Partition, *Proceedings of International Conference on Real-Time Systems and Applications*, pp.337–346 (2000).
11) Spuri, M. and Buttazzo, G.: Efficient Aperiodic Service under Earliest Deadline Scheduling, *Proceedings of the IEEE Real-Time Systems Symposium*, pp.2–11 (1994).