情報処理学会論文誌　コンピューティングシステム　Vol. 1　No. 2　96–110 (Aug. 2008)

マルチプロセッサ用の実時間電圧周波数制御

船岡健司[†1]　加藤真平[†1]　山崎信行[†1]

実時間システムにとって，実時間性を保証可能なシステム使用率が高いほど，プロセッサの電力量の効率にも優れている．本論文では，システム使用率を 100%でも実時間性を保証可能なマルチプロセッサ用の実時間スケジューリング手法を拡張した実時間電圧周波数制御手法を提案する．静的な手法は，電圧と周波数をプロセッサ間で均一もしくは独立に制御可能であるとき，理論的に最適である．動的な手法は，静的手法が多項式時間で電圧と周波数を決定可能であることを利用している．シミュレーションによる評価の結果，静的な手法は電圧と周波数を細かく制御可能であれば最適な値に近づくことを示した．また，動的な手法はシステムの選択可能な周波数に依存する度合が低いことを示した．

Real-time Voltage and Frequency Scaling on Multiprocessors

Kenji Funaoka,[†1] Shinpei Kato[†1]
and Nobuyuki Yamasaki[†1]

Optimal real-time scheduling is effective to not only schedulability improvement but also energy efficiency for real-time systems as compared to non-optimal algorithms. In this paper, real-time voltage and frequency scaling techniques based on optimal real-time scheduling for multiprocessors are presented. The static techniques are theoretically optimal when the voltage and frequency can be controlled both uniformly and independently among processors. The dynamic techniques leverage the static techniques which can solve the problem in polynomial time. Simulation results show that the static technique closely approaches the lower bound on energy consumption if the voltage and frequency can be controlled minutely. Furthermore the dynamic techniques cover up the disadvantages of the static techniques in the sense that the dynamic techniques are not practically affected by selectable frequencies.

1. 序　　論

組み込みシステムには，限られた資源および消費電力量という制約のもと，タスクの実時間性やシステム全体の高いスループットが要求される．このような組み込みシステムに対して，マルチプロセッサアーキテクチャである Simultaneous Multithreading（SMT）や Chip Multiprocessing（CMP）が利用されるようになった．本論文では，オペレーティングシステムによるマルチプロセッサシステムの実時間性の保証と消費電力量の削減を両立する手法に焦点を当てる．

最も重要な問題は，システムの実時間性の保証である．ロボットなどの時間制約を持つシステムでは，時間制約を守ることができなければシステムとしての価値が著しく低下する．実時間性の保証は，実時間スケジューリングアルゴリズムによって行う．実時間スケジューリングアルゴリズムに対する重要な指標の 1 つは，実時間性を保証可能なシステム使用率である．実時間性を保証可能なシステム使用率が高いと，より性能が低いプロセッサでも実時間性を保証可能となる．そのようなプロセッサは消費電力が低い傾向にあることから，消費電力量の削減という観点からも優れている．

システム使用率が 100%でも実時間性を保証可能なマルチプロセッサ用の実時間スケジューリングアルゴリズムは，NP 困難な問題であると考えられていた．そのため，シングルプロセッサ用のアルゴリズムである EDF [1] を拡張した EDF-FF [2] や EDF-US [3] が提案されていた．これらのアルゴリズムが実時間性を保証可能なシステム使用率は約 50%である．すなわち，実時間性を保証するには，残りの約 50%のプロセッサ時間が無駄となる可能性がある．理論的にはシステム使用率を 100%まで利用可能なアルゴリズムと比較して，2 倍のスループットのプロセッサ，もしくは，2 倍の数のプロセッサが必要になる可能性がある．そのようなプロセッサは消費電力が高い傾向にある．その後の研究により，システム使用率を 100%まで利用可能なマルチプロセッサ用の実時間スケジューリングアルゴリズムである PD^2 と EKG，LLREF が示された[4)–6)]．PD^2 は，Pfair [7] という厳しい制約を満たすため，時間を固定長に分割してスケジュールを行う．これにより頻繁にスケジュールを行う必要があるため，大きなオーバヘッドが発生する可能性がある．EKG は，タスクを特定のプロセッサに割り振るパーティショニングに類似した手法を利用しているため，特定のプロセッ

†1 慶應義塾大学
Keio University

サに負荷が偏ることになる．消費電力は，プロセッサ間の負荷を平均化することにより最小化される[8]．LLREF は，PD^2 よりオーバヘッドを削減できる可能性があり，プロセッサを平均的に利用することが可能である．

実時間スケジューリングに，プロセッサの電圧と周波数の制御を行い消費電力量を抑えるという概念を加えたものが実時間電圧周波数制御（RT-VFS）である．多くのシステムにとって，最も消費電力の大きな要素はプロセッサであることから，プロセッサの消費電力はシステム全体の消費電力に大きな影響を与える．多くのプロセッサは CMOS 回路によって形成されており，動作周波数の最大値は動作電圧に依存する．また，CMOS 回路の消費電力 E は，動作周波数 f と動作電圧 V の 2 乗に比例[9]（$E \propto fV^2$）することから，プロセッサの動作周波数と動作電圧を制御することにより，消費電力量を大きく削減可能である．ある実時間電圧周波数制御手法が実時間性を保証したうえで消費電力量を最小化するとき，その手法は**最適**であるという．

実際のシステムでは，選択可能なプロセッサの周波数は非連続であるなどの制約が存在する．そのような制約が存在する場合，最適な実時間電圧周波数制御は NP 困難な箱詰問題の要素を含むことになる．システム構成の変化や非周期処理への対応など，タスクセットの変化するシステムでは，変化のたびに NP 困難な箱詰問題を解くことは難しいと考えられる．本論文では，連続した値の周波数を選択可能であると仮定したうえで，理論的に最適な静的実時間周波数制御手法を提案する．また，その多項式時間で電圧と周波数を決定可能な静的手法を基にして動的な電圧周波数制御手法を提案する．最後に，設定可能な周波数に制限のある実際のシステムを想定して提案手法の評価を行う．

2. 関 連 研 究

実時間電圧周波数制御は，主にシングルプロセッサを対象として研究が行われてきた．Pillai ら[10] は，システム使用率を 100%まで利用可能なシングルプロセッサ用の実時間スケジューリングアルゴリズムである EDF を拡張した静的実時間電圧周波数制御手法を示した．また，EDF を実行時間が変動するシステムを対象とした Cycle-Conserving アルゴリズムと Look-Ahead アルゴリズムを提案した．Stankovic ら[11] は，時間制約が緩いシステムを対象とした Feedback アルゴリズムを提案した．Lee ら[12] は，非周期タスクも含まれるシステムでの実時間電圧周波数制御手法を提案した．このように，シングルプロセッサには，システム使用率を 100%まで利用可能な EDF を基礎として，多くの手法が提案されている．

マルチプロセッサを対象とした従来の実時間電圧周波数制御手法[13)–15)] は，シングルプロセッサのアルゴリズムを拡張したアルゴリズムを土台として実時間電圧周波数制御を行っている．前述のとおり，これらの手法で実時間性を保証するためには，より高い性能のプロセッサが必要となることから，結果的に消費電力の大きなプロセッサを利用しなければならない．

本論文では，マルチプロセッサ上でシステム使用率を 100%まで利用可能であり，そのうえで消費電力量の削減を行う初めての実時間電圧周波数制御を提案する．

3. システムモデル

システムは，M 個のプロセッサ $P = \{P_1, \ldots, P_M\}$ からなる．それぞれのプロセッサには，連続した値で周波数を決定可能である．プロセッサ P_k の正規化した周波数を α_k（$0 \leq \alpha_k \leq 1$）とする．本論文では，α_k を単に周波数と表現する．各プロセッサで周波数 α_k を独立に設定可能なシステムもあれば，均一な周波数にしか設定できないシステムも存在する．それぞれのプロセッサ P_k に周波数 α_k を設定したとき，プロセッサ P_k が動作可能な最小の電圧は一意に決定可能であるため，このときの消費電力を $g(\alpha_k)$ と表す．現在の電圧や周波数が可変なプロセッサ技術では，$g(\alpha_k)$ を狭義凸関数かつ狭義単調増加と仮定可能である[16)–19)]．実際のシステムでは，制御可能な周波数に制限がある．制御可能な周波数の集合を $f = \{f_1, \ldots, f_m | f_1 < \cdots < f_m\}$ とする．電圧周波数制御手法がプロセッサ P_k の正規化された周波数を α_k にすると決定したとき，$\alpha_k \leq f_i/f_m$ を満たす最小の周波数 f_i をシステムに設定する．

システムには，システムが処理しなければならない N 個の周期タスクの集合であるタスクセット $T = \{T_1, \ldots, T_N\}$ が存在する．それぞれのタスクは独立しており，いつでも横取り（実行の一時中断）やマイグレーション（タスクのプロセッサ間の移動）が可能である．タスク T_i は，周期 p_i と最悪実行時間 c_i に特徴付けられる．最悪実行時間は，スケジューリングや電圧周波数制御のオーバヘッドを含めた時間である．最悪実行時間の値は，最悪実行時間解析[20)–22)] によって与えられているものとする．タスク T_i のプロセッサ使用率を $u_i = c_i/p_i$（$0 < u_i \leq 1$）と定義する．タスク T_i をプロセッサ P_k で実行したとき，タスク T_i は単位実行時間あたり u_i/α_k のプロセッサ時間を要求する．タスク T_i の相対デッドラインは周期 p_i と等しく，このデッドラインまでに実行を終えなければならない．タスクセット T のプロセッサ使用率を $U = \sum_{T_i \in T} u_i$，タスクセット T のシステム使用率を $U_s = U/M$ と定義する．タスクセット T に含まれるタスクのプロセッサ使用率の最大値を $X = \max\{u_i | T_i \in T\}$ と定義する．

4. LLREF スケジューリングアルゴリズム

LLREF[6] は，$U_s \leq 1$ を満たすいかなるタスクセットでも実時間性を保証可能なマルチプロセッサ用の実時間スケジューリングアルゴリズムである．理論的に，LLREF でスケジュール不可能なタスクセットは，他のいかなるアルゴリズムをもってしてもスケジュール不可能である．LLREF は，fluid スケジュール[23] の概念を利用してタスクのスケジュールを行う．

図 1 に fluid スケジュールの概念図を示す．下段には実際のシステムでタスク T_i が実行される様子，上段にはタスクの実行によりタスク T_i の残り実行時間が減少する様子が示されている．タスクは到着時刻から実行可能な状態となり，デッドラインまでに残り実行時間をゼロにしなくてはならない．実際のスケジュールでは，ある時刻に 1 つのタスクしか実行することができない．そのため，図の下段のようにタスクの実行がブロックされる可能性がある．fluid スケジュールは，つねに一定の割合でタスクの実行を行う概念的なスケジュール手法である．図中の fluid スケジュールによって残り実行時間が減少する様子を示す線分を fluid スケジュールパスと呼ぶ．これを実現するには，1 つのプロセッサで複数のタスクを同時に実行しなくてはならないことから，実現は不可能である．

LLREF は，fluid スケジュールの考え方を応用して実時間性を保証する．**図 2** 上段は，図 1 上段を複数タスクかつ複数周期に広げて示したものである．図 2 の垂直方向の破線で示したように，全タスクのデッドラインごとに時間を分割する．このデッドラインで区切られた各区間を**局所**という．重要な点は，局所の右端において，全タスクの残り実行時間を fluid スケジュールの値以下にすることにより，全デッドラインを守ることが可能であるという点である．LLREF では，この考え方を T-L Plane（Time and Local Execution Time Domain Plane）と呼ばれる直角二等辺三角形によって実現する．図 2 上段に示したように，全タスクの各局所の中で T-L Plane の右頂点が fluid スケジュールパスと重なるように T-L Plane を配置する．この T-L Plane の右頂点を目指してタスクを実行することにより，タスクの残り実行時間を fluid スケジュールの値に合わせる．同じ区間の T-L Plane は合同であることから，図 2 下段のように T-L Plane を重ね合わせることにより，その局所では単一の T-L Plane の中でのタスクのスケジュールを考えればよい．

図 3 上段に，図 2 下段で示した重ね合わせた 1 つの T-L Plane を示す．縦軸と横軸は同じ時間の尺度であるため，T-L Plane の斜辺（NNLD）の傾きは −1 である．T-L Plane の中で，タスク T_i を実行しなければならない残り時間をタスク T_i の**局所の残り実行時間（local remaining execution time）**という．T-L Plane は，横軸に時刻，縦軸にタス

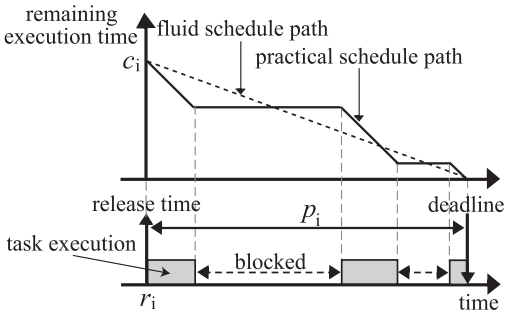

図 1　fluid スケジュールと実際のスケジュール
Fig. 1　Fluid schedule and a practical schedule.

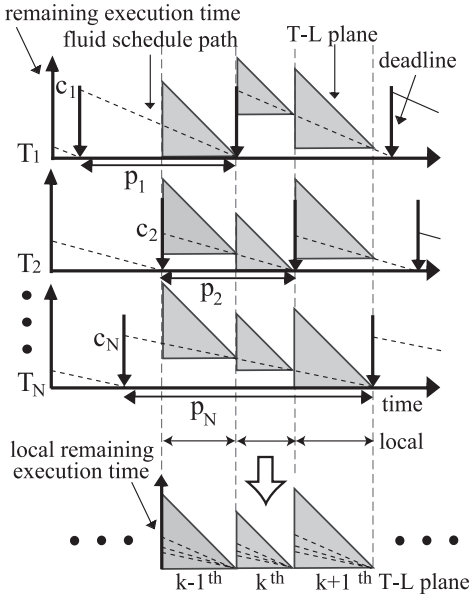

図 2　T-L Plane による実時間スケジューリングの抽象化
Fig. 2　T-L plane abstraction for real-time scheduling.

クの局所の残り実行時間を示している．それぞれのタスクは，T-L Plane の中でタスクの局所の残り実行時間を示すトークンに対応する．時刻 t_0 において，全トークンは対応するタスクの fluid スケジュールパス上に存在する．局所の残り実行時間は，タスクを実行するとトークンが傾き -1 の右下方向に移動して減少する．タスクを実行しないとトークンが傾き 0 の右方向に移動して変化しない．全トークンが T-L Plane の右頂点まで到達可能であることを**局所でスケジュール可能である**という．局所でスケジュール可能であることを実現するために，LLREF はイベントごとに局所の残り実行時間の大きい M 個のタスクを実行する（Largest Local Remaining Execution time First）．そのイベントとは，タスクの到着（T-L Plane の左端）と後述するイベント B，イベント C である．イベント B はトークンが T-L Plane の底辺に到達した際に，イベント C はトークンが T-L Plane の斜辺（NLLD）に到達した際に発生する．これらのイベントごとに実行するタスクの選択を繰り返すことにより，$U_s \leq 1$ であれば LLREF はすべてのデッドラインを守ることが可能である．

図 3 を利用して LLREF のスケジュール例を示す．図 3 上段には，T-L Plane の中でトークンが動く様子，下段にはその結果によりタスクを実行する様子を示した．図のように 4 つのタスク（$T_1, \ldots, T_4$）を 2 つのプロセッサ（P_1，P_2）上で実行するシステムを想定する．プロセッサ数が 2 であるため，同時に 2 つのタスクを実行可能である．時刻 t_0 では，すべてのタスクが fluid スケジュールパス上に存在する．ここで，局所の残り実行時間の大きい 2 個のタスク（T_1，T_2）を実行する．時刻 t_1 まで実行すると，トークン T_2 が三角形の底辺まで到達することにより，イベント B が発生する．同様に，この段階で局所の残り実行時間の大きな T_1 と T_3 を実行する．さらに実行を進めると，時刻 t_2 にタスク T_4 が三角形の斜辺まで到達して，イベント C が発生して T_1 と T_4 を選択する．これを繰り返すことにより，トークンを T-L Plane の右頂点まで導く．

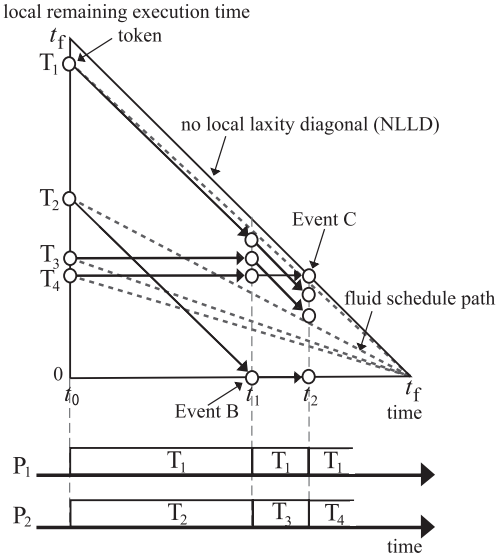

図 3　LLREF によるスケジューリング
Fig. 3　LLREF scheduling.

T-L Plane において，j 番目のイベントが発生する時刻を t_j とする．ただし，T-L Plane の左端のタスクの到着時刻を $t_0 = 0$，T-L Plane の右端の次のタスクの到着時刻を t_f とする．時刻 t_j におけるタスク T_i の局所の残り実行時間を $l_{i,j}$，タスク T_i の局所のプロセッサ使用率を $r_{i,j} = l_{i,j}/(t_f - t_j)$，局所のプロセッサ使用率の和を $S_j = \sum_{T_i \in T} r_{i,j}$ と定義する．

5. 静的電圧周波数制御

本章では，T-L Plane の斜辺の傾きを周波数に対応させる T-L Plane Transformation を提案する．また，T-L Plane Transformation を利用した静的実時間電圧周波数制御手法を示す．**図 4** にプロセッサの周波数が 0.5 である T-L Plane を示す．元の T-L Plane と異なる点は，タスクを実行すると T-L Plane の斜辺に沿って緩やかに残り実行時間が減少する点である．

システムによって，それぞれのプロセッサで独立して周波数を決定することが可能なシステムもあれば，すべてのプロセッサで均一な周波数のみしか決定できないシステムも存在す

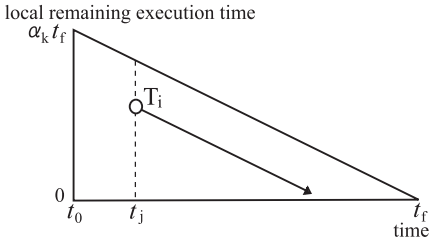

図 4　T-L Plane Transformation（$\alpha_k = 0.5$）
Fig. 4　T-L Plane Transformation ($\alpha_k = 0.5$).

Algorithm: DecideUniformFrequency
1: **foreach** 1...M **as** k
2: 　$\alpha_k = \max\{X, U/M\}$
3: **end foreach**

図 5　静的な均一周波数決定アルゴリズム
Fig. 5　Uniform RT-SVFS.

る．SMT では，プロセッサ資源を共有して実行することから，すべてのスレッドは同じ周波数で動作することになる．CMP では，設計によって独立して周波数を決定することが可能かどうか決まる．SMP（Symmetric Multiprocessing）では，多くの場合，それぞれのプロセッサで独立して周波数を決定可能である．そのため，これらの異なるシステムに適用可能な実時間電圧周波数制御手法をそれぞれ示す．

以下の節では，実時間性を保証可能な周波数 α_k の決定方法について述べる．また，最悪実行時間解析には，オーバヘッドの上限を求める必要がある．オーバヘッドの上限は，スケジューラの起動回数の上限を示すことにより求めることが可能である[6]．静的手法では，動的に電圧と周波数を制御しないため，電圧と周波数の制御によるオーバヘッドはゼロである．

5.1　均一電圧周波数制御

すべてのプロセッサの周波数を均一な値 α（$= \alpha_1 = \alpha_2 = \ldots = \alpha_M$）にのみ制御可能なシステムに対する静的な実時間電圧周波数制御手法 DecideUniformFrequency を**図 5** に示す．本手法は，周波数を独立に設定できないシステムにおいて，実時間性を保証することが可能であり，最適な実時間電圧周波数制御を行うことを付録 A.1 に示した．本手法は，次節の独立周波数制御手法と異なり，設定可能な周波数に制限のある実際のシステムでも最適であることは明らかである．

DecideUniformFrequency の 2 つの例を**図 6** に示す．図中に示されたプロセッサ使用率を持つタスクが 4 プロセッサ上で実行されるときの周波数決定の結果を示している．上下どちらのタスクセットも $U = 3$ であるため，理想的には各プロセッサの周波数を $U/M = 0.75$ として実行しても実時間性を保証可能である．しかしながら，下の例では，大きなプロセッサ使用率 0.8 を持つタスクにより，実行に必要な周波数が大きくなる．このように，均一周波数制御では，プロセッサ使用率の大きなタスクが消費電力削減のボトルネックとなる．

本提案手法のの計算量は，U や X を算出するために $O(N)$ である．また，均一周波数制御時の LLREF によるスケジューリングのオーバヘッドの上限は，元の LLREF と同じ概念でスケジュールを行うことから，即座に求めることが可能である．

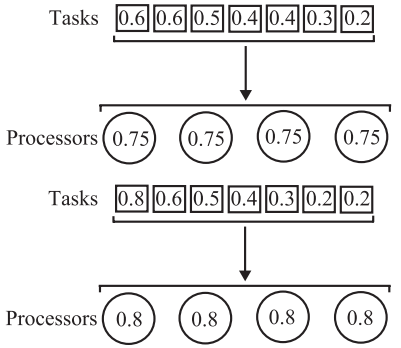

図 6　静的な均一周波数制御の例
Fig. 6　Examples of uniform RT-SVFS.

定理 1（スケジューラ起動回数の上限） *LLREF* によってスケジュールを行い，*DecideUniformFrequency* によって電圧周波数制御を行ったとき，時刻 t_s から時刻 t_e の間にスケジューラが起動される回数の上限は，

$$(N+1)\left(1 + \sum_{T_i \in T} \left\lceil \frac{t_e - t_s}{p_i} \right\rceil\right)$$

である．

証明　LLREF の証明[6] と同様である．　□

5.2　独立電圧周波数制御

独立電圧周波数制御の方針を示す．均一制御時のプロセッサ使用率の大きいタスクによるボトルネックを解消するため，タスクをヘビータスクとライトタスクに分類する．ヘビータスクに分類されたタスクは，それぞれ 1 つのプロセッサを占有する．ヘビータスク T_i をプロセッサ P_k で実行するときの周波数は，$\alpha_k = u_i$ である．ライトタスクに分類されたタスクは，ヘビータスクを実行していないプロセッサで LLREF によって実行され，プロセッサ周波数を DecideUniformFrequency で決定する．本手法は，実時間スケジューリングアルゴリズムである EDF-US[3] や EKG[5] の手法と似ているが，目的や分類手法はまったく異なる．

プロセッサ周波数を独立に制御可能なシステムに対する実時間電圧周波数制御手法 DecideFrequency を**図 7** に示す．ヘビータスクの集合を T^{heavy}，ライトタスクの集合を T^{light}，

Algorithm: DecideFrequency
Require: $u_1 \geq u_2 \geq \ldots \geq u_N$
1: $T^{\text{heavy}} = \phi$
2: $T^{\text{light}} = T$
3: **foreach** 1...M **as** i
4:　**if** $X^{\text{light}} > U^{\text{light}}/(M-H)$ **then**
5:　　$T^{\text{heavy}} = T^{\text{heavy}} \cup \{T_i\}$
6:　　$T^{\text{light}} = T^{\text{light}} \backslash \{T_i\}$
7:　**else**
8:　　**break**
9:　**end if**
10: **end foreach**
11: **foreach** 1...M **as** k
12:　**if** P_k executes a heavy task T_k **then**
13:　　$\alpha_k = u_k$
14:　**else**
15:　　$\alpha_k = U^{\text{light}}/(M-H)$
16:　**end if**
17: **end foreach**

図 7　静的な独立周波数決定アルゴリズム
Fig. 7　Independent RT-SVFS.

ライトタスクのプロセッサ使用率の和を U^{light}，T^{light} に含まれるタスクのプロセッサ使用率の最大値を $X^{\text{light}} = \max\{u_i | T_i \in T^{\text{light}}\}$ と定義する．ヘビータスクの数を H と表す．ヘビータスクは，P_1 から P_H の順に割り振る．ライトタスクは，ヘビータスク割り振られていない $M-H$ 個のプロセッサで実行される．本手法は，実時間性を保証可能であり，マルチプロセッサ用の静的な実時間電圧周波数制御として最適であることを付録 A.2 に示した．

DecideFrequency は理論的に最適であるが，制御可能な周波数に制限のある実際のシステムでは最適でない．プロセッサ使用率（1.0，0.9，0.6，0.5，0.1）のタスクが 4 プロセッサ上で実時間電圧周波数制御された結果を**図 8** に示す．それぞれのプロセッサには，**表 1** の System 1 に示された 3 段階の周波数と電圧を独立に設定可能である．上段に DecideFrequency の結果，下段に全探索を行い最適な組合せを決定する Exhaustive の結果を示している．DecideFrequency では，タスクのプロセッサ使用率の降順にソートを行い，$u_i = X^{\text{light}}$ であるライトタスク T_i をヘビータスクへ分類していく．分類していくとヘビータスク数が 2 のとき，初めて $X^{\text{light}} \leq U^{\text{light}}/(M-H)$ を満たす．したがって，プロセッサ使用率が 1.0 と 0.9 のタスクがヘビータスク，他のタスクがライトタスクとなる．Exhaustive では，

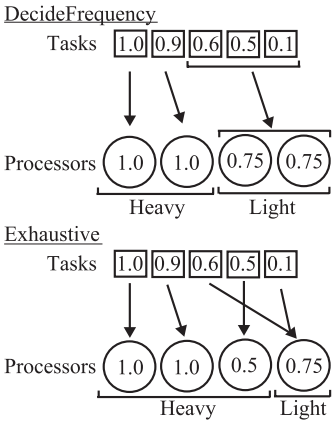

図 8　静的な独立周波数制御の例
Fig. 8　Examples of independent RT-SVFS.

表 1　評価用のシステム
Table 1　Systems for evaluation.

System 1		System 2		System 3	
α	V	α	V	α	V
0.5	3	0.5	3	0.36	1.4
0.75	4	0.75	4	0.55	1.5
1.0	5	0.83	4.5	0.64	1.6
		1.0	5	0.73	1.7
				0.82	1.8
				0.91	1.9
				1.0	2.0

プロセッサ使用率が 0.5 のタスクもヘビータスクとすることにより DecideFrequency よりも消費電力を削減することに成功している．しかしながら，NP 困難な Exhaustive が動的なシステムの変化に対応することは困難である．

DecideFrequency の計算量は，タスクをソートするために最悪 $O(N^2)$ である．また，独立周波数制御時は，ヘビータスクのスケジュールを行う必要がないことから，DecideUniformFrequency よりもスケジュールのオーバヘッドは小さい．独立周波数制御を利用した LLREF によるスケジューリングのオーバヘッドの上限は，ライトタスクをスケジュールするスケジューラの起動回数の上限から求めることが可能である．

定理 2（スケジューラ起動回数の上限） *DecideFrequency* によって実時間電圧周波数制御を行ったとき，時刻 t_s から時刻 t_e の間にスケジューラが起動される回数の上限は，

$$(N-H+1)\left(1+\sum_{T_i\in T^{light}}\left\lceil\frac{t_e-t_s}{p_i}\right\rceil\right)$$

である．

証明　独立周波数制御では，ヘビータスクのスケジュールを行う必要がない．したがって，定理 1 をライトタスクにのみ適用することにより得られる．□

6. 動的電圧周波数制御

動的な実時間電圧周波数制御は，タスクの実行時間の変動に対応することが可能であるため，静的な実時間電圧周波数制御よりも消費電力量を削減することが可能となる．静的な手法は，タスクのプロセッサ使用率に基づいてプロセッサの周波数を決定するのに対して，動的な手法は，タスクの局所のプロセッサ使用率に基づいてプロセッサの周波数を決定する．タスクセット T は，実行可能なタスクの集合に置き換えられる．すなわち，実行の終了しているタスクは，次の到着時刻までタスクセットに含まれない．実行の終了したタスクのプロセッサ使用率を考慮する必要がないため，動的な手法は静的な手法よりも消費電力量を削減できる可能性がある．時刻 t_j における T と T^{light} のタスクの局所のプロセッサ使用率の最大値をそれぞれ $Y_j=\max\{r_{i,j}|T_i\in T\}$ と $Y_j^{\mathrm{light}}=\max\{r_{i,j}|T_i\in T^{\mathrm{light}}\}$ と定義する．動的な均一電圧周波数制御手法と独立電圧周波数制御をそれぞれ**図 9** と**図 10** に示す．これらのアルゴリズムを時刻 t_0 と任意の時刻 $t_{j'}$ に行う．電圧と周波数の変更を行ったとき，LLREF による再スケジュールが必要となる．提案手法を利用しても全タスクがスケジュール可能であることを付録 A.3 に示す．

6.1 実環境での動的電圧周波数制御

前節では，任意の周波数でプロセッサが動作可能であると仮定していたが，実際のプロセッサでは動作可能な周波数に制限がある．周波数制御の例を示すために，**図 11** のような横軸に t_f で正規化された時刻と縦軸に f_m で正規化された周波数を表す T-F Plane（Time and Frequency Domain Plane）を考える．T-F Plane は，1 つの T-L Plane 内で実行しているときの周波数制御の結果を示しており，T-F Plane と T-L Plane は相似である．T-F Plane は，電圧と周波数制御を行うプロセッサの単位ごとに 1 つ存在する．すなわち，均一電圧周波数制御時には 1 つ，独立電圧周波数制御時には計 $H+1$ だけ存在する．それぞれの T-F Plane には，トークンが 1 つだけ存在する．トークンの初期位置は，電圧周波数制御手法によって時刻 t_0 に与えられた周波数 α_k である．また，それぞれの周波数 f_i に対して，T-F Plane の左辺の f_i/f_m から右頂点まで破線を引く．トークンは，選択されている周波数 f_i の破線に並行に移動する．T-F Plane のトークンの局所のプロセッサ使用率は，実時間性を保証可能なプロセッサ周波数と等しい．独立電圧周波数制御時にヘビータスクとライトタスクが入れ替わると，トークンは連続でない位置に変化する可能性がある．

図 11 に 3 段階の周波数 $f=\{f_1,f_2,f_m|f_1<f_2<f_m\}$ をプロセッサごとに独立して制

Algorithm: DecideUniformFrequencyDynamic

```
1: foreach 1...M as k
2:     α_k = max{Y_j', S_j'/M}
3: end foreach
```

図 9　動的な均一周波数決定アルゴリズム
Fig. 9　Uniform RT-DVFS.

Algorithm: DecideFrequencyDynamic

```
Require: r_{1,j'} ≥ r_{2,j'} ≥ ... ≥ r_{N,j'}
 1: T^heavy = φ
 2: T^light = T
 3: foreach 1...M as i
 4:     if Y_j'^light > S_j'^light/(M − H) then
 5:         T^heavy = T^heavy ∪ {T_i}
 6:         T^light = T^light \ {T_i}
 7:     else
 8:         break
 9:     end if
10: end foreach
11: foreach 1...M as k
12:     if P_k executes a heavy task T_k then
13:         α_k = r_{k,j}
14:     else
15:         α_k = S_j'^light/(M − H)
16:     end if
17: end foreach
```

図 10　動的な独立周波数決定アルゴリズム
Fig. 10　Independent RT-DVFS.

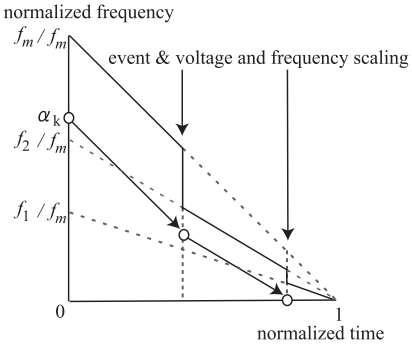

図 11　T-F Plane
Fig. 11　T-F Plane.

御可能であるシステムにおいて，ライトタスクを実行するプロセッサの周波数制御の例を示す．図のように時刻 t_0 にトークンが $f_2/f_m < r_{i,0} \leq f_m/f_m$ の位置にあると仮定すると，実時間性を保証するために $\alpha_k \leq f_i/f_m$ を満たす最小の周波数 f_m を選択する．すなわち，つねにトークンが T-F Plane 内部に存在可能である最小の周波数 f_i を選択する．この状態で実行していくと，トークンが周波数 f_2 を表す破線の下側に移動する．定理 12 より任意の時刻に電圧と周波数を制御可能である．ヘビータスクであれば，周波数 f_2 を表す破線との交点で即座に周波数を下げることにより，早い段階で周波数を下げることが可能である．しかしながら，ライトタスクでは，周波数を下げたときに再スケジュールを行う必要がある．そのため，電圧と周波数を制御するオーバヘッドだけでなく，再スケジュールを行うオーバヘッドも発生することになる．動的電圧周波数制御時に必要な再スケジュールを LLREF のスケジュールと同時に行うことにより，新たな再スケジュールが発生しないため，最悪のオーバヘッドは静的な手法と同様である．また，LLREF でスケジュールを行うときにタスクのソートが必要となるため，独立電圧周波数制御手法のタスクのソートを省略することが可能である．そのため，イベント時に電圧と周波数を下げられるか判断を行うことにより，消費電力とオーバヘッドを効率的に抑えることが可能であると考えられる．

動的な手法のスケジューリングのオーバヘッドの上限は，イベントの数の上限が静的な手法と同じであるため，スケジューリング自体のオーバヘッドも同様となる．また，静的手法と異なり実行時に電圧と周波数を制御するため，電圧と周波数の制御の回数もスケジューラの起動回数から算出する必要がある．

定理 3（電圧と周波数の変更回数の上限）　イベント発生時に動的電圧周波数制御を行ったとき，時刻 t_s から時刻 t_e の間に電圧と周波数を変更する回数の上限は，

$$(N - H + 1)\left(1 + \sum_{T_i \in T^{light}} \left\lceil \frac{t_e - t_s}{p_i} \right\rceil\right)$$

である．均一周波数制御時には $T^{light} = T$ である．

証明　電圧と周波数の制御はイベントが起こったときに限定したため，その回数の上限はスケジューラの起動回数の上限と等しい．□

7. 評　　価

実環境を想定した提案手法の有効性を示す．本提案手法は，特定のシステムのみを対象とする手法ではないため，あらゆるシステムを想定して評価をすることが望ましいが，限られた誌面上ですべての可能性を列挙することは不可能である．そのため，周波数と電圧の制御可能な値に制限がある表 1 のシステムを利用する．正規化した消費電力量 EnergyRatio の定義を示す．

$$\text{EnergyRatio} = \frac{1}{L}\int_0^L \frac{\sum_{P_k \in P} \alpha_k V_k^2}{M V_{\max}^2}\, dt$$

つねに最大の電圧と周波数で動作したとき EnergyRatio は 1 となる．ここで，システムの最大電圧を $V_{\max}$，システムを動作させ評価する時間を L とした．

7.1 評 価 方 法

提案手法の評価を以下のように行う．システムのプロセッサ数を $M = 4$ とする．それぞれのプロセッサは，表 1 に示された周波数 α とそれに対応する動作に必要な最小の電圧 V の組をプロセッサごとに独立して設定可能である．システム 1 からシステム 3 に行くほど，周波数と電圧を細かく制御可能である．

タスクセットは，タスクのプロセッサ使用率 u_i が $[0.1, 1.0]$ の範囲で一様分布となるように生成した．タスクの周期 p_i は $[100, 3000]$ の整数値から一様乱数により生成する．最悪実行時間 c_i は $u_i p_i$ の整数部分とする．システムモデルにも示したとおり，最悪実行時間にはすべての実行時のオーバヘッドを含んでいるものとする．初めにタスクセットを空集合としてタスクを追加していき，タスクセットのプロセッサ使用率が目標値を超えてしまったらタスクを破棄する．最後に，タスクセットのプロセッサ使用率が目標値となるようなタスクを生成して，これをタスクセットとする．

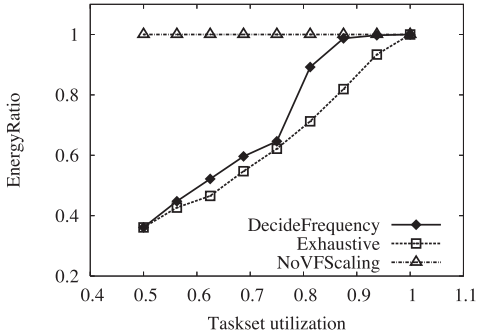

図 12　システム 1 における静的電圧周波数制御の消費電力量
Fig. 12　EnergyRatio of static techniques on System 1.

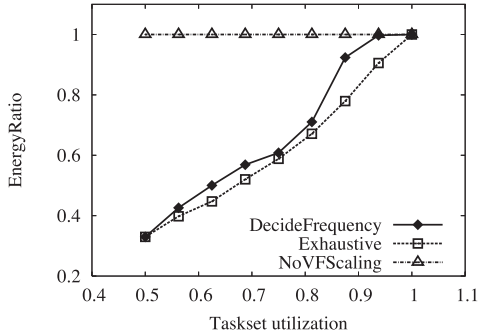

図 13　システム 2 における静的電圧周波数制御の消費電力量
Fig. 13　EnergyRatio of static techniques on System 2.

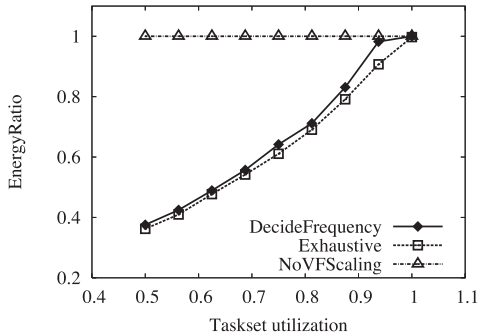

図 14　システム 3 における静的電圧周波数制御の消費電力量
Fig. 14　EnergyRatio of static techniques on System 3.

タスクセットのプロセッサ使用率 U が従来の非最適な手法では実時間性を保証不可能な [2.0,4.0] の 0.25 ごとに，それぞれ 100 のタスクセットを生成して EnergyRatio の平均値を評価結果として示す．各タスクセットの EnergyRatio は，以下のようにして求めた．静的な手法では実行時に周波数と電圧を変化させないため，タスクセットを生成した段階で EnergyRatio を求める．動的な手法では時間 $L = [0, \min\{\mathrm{lcm}\{p_i | T_i \in T\}, 2^{32}\}]$ だけシステムを動作させて EnergyRatio を算出する．周期タスクモデルでは，周期の最小公倍数 $\mathrm{lcm}\{p_i | T_i \in T\}$ ごとに同じスケジュールが繰り返されるため，周期の最小公倍数までスケジュールを行い評価を行うことが望ましい．しかしながら，周期を一様乱数で生成しているため，評価時間が非常に長くなってしまう可能性がある．そのため，十分に長い時間として 2^{32} を評価時間の最大値とした．システムの実行中は，変形された T-L Plane の中で LLREF によってスケジュールを行い，電圧と周波数が下げられるか判断を行うのはイベントが発生したときである．

7.2　静的電圧周波数制御

比較対象は，全探索を行い最適な組合せを決定する Exhaustive アルゴリズム，消費電力量削減を行わない NoVFScaling，本論文で提案した DecideFrequency である．実時間性を保証したうえで Exhaustive より消費電力量を削減することは不可能である．したがって，EnergyRatio の結果が Exhaustive に近くなるほど優れたアルゴリズムであることを意味する．

それぞれのシステムの評価結果を**図 12**，**図 13**，**図 14** に示す．図の横軸にタスクセットのシステム使用率，縦軸に EnergyRatio の平均値を示している．静的な手法では最悪の状態を想定しなければならないため，高いシステム使用率かつ制御可能な周波数が粗いほど消費電力量を削減することが難しいことが読み取れる．そのような状態よりも低いシステム使用率では，最適なスケジュールを実現可能な本提案手法が有効となると思われるタスクセットのシステム使用率が高い範囲であるほど，評価結果の曲線の傾きが大きい．提案手法は，システムの負荷が高いとき，システムの負荷に応じて消費電力量を特に大きく削減可能であることが読み取れる．システムに設定可能な周波数と電圧が細かいほど，DecideFrequency

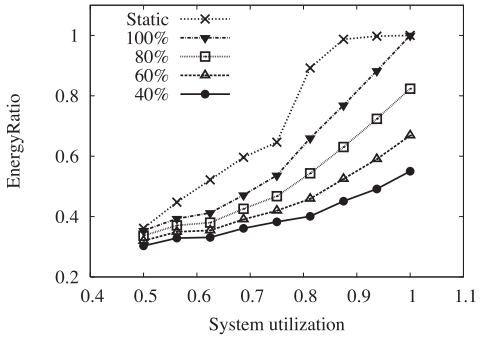

図 15　システム 1 における動的電圧周波数制御の消費電力量
Fig. 15　EnergyRatio of dynamic techniques on System 1.

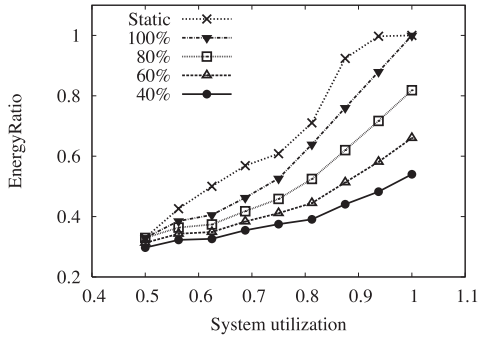

図 16　システム 2 における動的電圧周波数制御の消費電力量
Fig. 16　EnergyRatio of dynamic techniques on System 2.

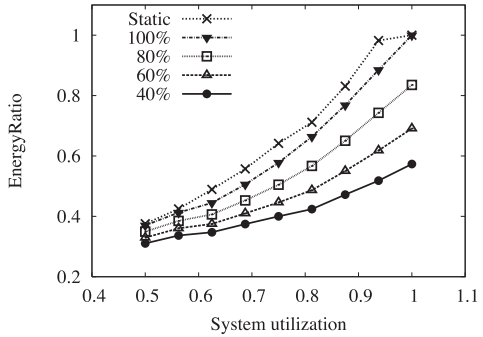

図 17　システム 3 における動的電圧周波数制御の消費電力量
Fig. 17　EnergyRatio of dynamic techniques on System 3.

の結果が Exhaustive の結果が近づいている．DecideFrequency と Exhaustive の最大の差は，システム 1 のシステム使用率 0.75 で 0.18，システム 2 のシステム使用率 0.875 で 0.14，システム 3 のシステム使用率 0.9375 で 0.075 となった．これは，DecideFrequency の理論的な最適性により，周波数と電圧を細かく制御可能であると，最適な値に近づいていくためであると考えられる．

システム 1 のシステム使用率 0.75 やシステム 2 のシステム使用率 0.75 と 0.8125 のように，周囲よりも DecideFrequency と Exhaustive の差が小さい点が存在する．これは，システム 1 が $\alpha = 0.75$ やシステム 2 が $\alpha = 0.75$，0.83 を設定可能であるため，そのあたりで効率的にタスクを分配できる可能性が高いためであると考えられる．いい換えれば，それ以外のシステム使用率ではより高い周波数を選択しなければならないため，理想的な場合よりも消費電力量が高くなっている．よって，プロセッサの電圧と周波数を細かく制御可能でない場合でも，タスクセットのシステム使用率を考慮してシステムの全体設計を行うことにより，消費電力量を抑えることができる可能性がある．

7.3　動的電圧周波数制御

動的電圧周波数制御は，周期ごとに実際の実行時間が一様に変動する状態を比較した．比較対象は，実際の実行時間が最悪の実行時間 c_i の 40% （$[0.4c_i, c_i]$）まで変動する場合，60%（$[0.6c_i, c_i]$），80% （$[0.8c_i, c_i]$），つねに 100%である．参考のために静的手法である DecideFrequency の結果も Static として示している．

それぞれのシステムの評価結果を**図 15**，**図 16**，**図 17** に示す．静的な手法では，システムの制御可能な周波数によって，特に高いシステム使用率において消費電力量が大きく異なっている．それに対して，動的な手法では，システムごとに制御可能な周波数が違うにもかかわらず大きな違いは見られない．これは，制御可能な周波数が荒く高い電圧と周波数を設定しなければならない場合でも，タスクの実行が早く終わることになり，その後は低い電圧と周波数で実行することが可能となるためであると考えられる．すなわち，最終的な消費電力量の差が動的な制御によって隠蔽されている．システム使用率が低いときに結果が同じ値に近づいていく原因は，システムの周波数と電圧が表 1 の最上段に示した最小の値に制

限されるためであると考えられる．

タスクの実行時間が変動する影響は，どのシステムでも大きな違いは見られず，システム使用率 1 において，80%で約 0.82，60%で約 0.67，40%で約 0.55 であった．この結果は，静的手法の結果と比較すると，実行時間の変動に応じて消費電力の削減効果を期待できることが読み取れる．また，ここで注目すべきは，タスクの実行時間がつねに最悪実行時間と等しい 100%でも，システム使用率 1 を除いて，静的な場合と比較して消費電力量を削減している点である．これは，先ほどの考察と同様に，一時的に高い電圧と周波数で実行していても，タスクの実行が早く終わったときに電圧と周波数を下げることができるためであると考えられる．システム使用率が 1 のときに静的な手法と同様になる原因は，システム使用率が 1 であるとシステムがアイドルとなる時間が存在しないためである．

8. 結　　論

本論文では，システム使用率を 100%まで利用可能なマルチプロセッサ用の実時間スケジューリングアルゴリズム LLREF を拡張することにより，実時間電圧周波数制御手法 T-L Plane Transformation を提案した．静的な手法は，電圧と周波数を均一にのみ制御可能なシステムと独立に制御可能なシステムの両方に対して，理論的に最適である．動的な手法は，静的な手法が多項式時間でプロセッサの電圧と周波数を決定可能である特徴を利用して，実行時間の変動などのシステムの動的な変化に対応することが可能となった．評価の結果，静的な手法は制御可能な電圧と周波数に制限があるような実際のシステムにおいても，細かく電圧と周波数を制御可能であれば，最適なアルゴリズムに近づくことを示した．全探索を行う手法は NP 困難であることから，提案手法は，動的なシステム構成の変化に対応する手法として有効であると考えられる．動的な手法は，静的な手法がシステムの制御可能な周波数に大きく依存するという欠点を克服することができていることを示した．また，実行時間の変動に合わせて電圧と周波数を制御できていることを示した．

今後は，実機上で LLREF の実装を行い，電圧周波数制御の頻度とスケジュール可能性のトレードオフを検討したい．電圧周波数制御を行うと，物理的な制約により一時的にプロセッサの動作が停止する．したがって，システムによって消費電力量を抑えることとスケジュール可能性を十分に両立できる電圧と周波数の変更頻度は異なると考えられる．これらを総合的に評価してシステムに組み込む必要がある．

謝辞　本研究は，科学技術振興機構 CREST の支援による．

参 考 文 献

1) Liu, C.L. and Layland, J.W.: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, *J. ACM*, pp.46–61 (1973).
2) Lopez, J.M., Garcia, M., Diaz, J.L. and Garcia, D.F.: Worst-Case Utilization Bound for EDF Scheduling on Real-Time Multiprocessor Systems, *Proc. 12th Euromicro Conference on Real-Time Systems*, pp.25–33 (2000).
3) Baker, T.P.: An Analysis of EDF Schedulability on a Multiprocessor, *IEEE Trans. on Parallel and Distributed Systems*, Vol.16, No.8, pp.760–768 (2005).
4) Anderson, J.H. and Srinivasan, A.: Early-Release Fair Scheduling, *Proc. 12th Euromicro Conference on Real-Time Systems*, pp.35–43 (2000).
5) Andersson, B. and Tovar, E.: Multiprocessor Scheduling with Few Preemptions, *Proc. 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp.322–334 (2006).
6) Cho, H., Ravindran, B. and Jensen, E.D.: An Optimal Real-Time Scheduling Algorithm for Multiprocessors, *Proc. 27th IEEE Real-Time Systems Symposium*, pp.101–110 (2006).
7) Baruah, S.K., Cohen, N.K., Plaxton, C.G. and Varvel, D.A.: Proportionate Progress: A Notion of Fairess in Resource Allocation, *Algorithmica*, Vol.15, No.6, pp.600–625 (1996).
8) Aydin, H. and Yang, Q.: Energy-Aware Partitioning for Multiprocessor Real-Time Systems, *Proc. 17th IEEE International Parallel and Distributed Processing Symposium*, pp.22–26 (2003).
9) Burd, T.D. and Brodersen, R.W.: Energy Efficient CMOS Microprocessor Design, *Proc. 28th Annual Hawaii International Conference on System Sciences*, pp.288–297 (1995).
10) Pillai, P. and Shin, K.G.: Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems, *Proc. ACM Symposium on Operating Systems Principles*, pp.89–102 (2001).
11) Stankovic, J.A., Lu, C. and Son, S.H.: The Case for Feedback Control Real-Time Scheduling, *Proc. 11th Euromicro Conference on Real-Time Systems*, pp.11–20 (1999).
12) Lee, C.H. and Shin, K.G.: On-Line Dynamic Voltage Scaling for Hard Real-Time Systems Using the EDF Algorithm, *Proc. 25th IEEE Real-Time Systems Symposium*, pp.319–335 (2004).
13) Xian, C., Lu, Y.-H. and Li, Z.: Energy-Aware Scheduling for Real-Time Multiprocessor Systems with Uncertain Task Execution Time, *Proc. 44th ACM/IEEE Design Automation Conrefence*, pp.664–669 (2007).

14) Chen, J.-J. and Kuo, T.-W.: Allocation Cost Minimization for Periodic Hard Real-Time Tasks in Energy-Constrained DVS Systems, *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pp.255–260 (2006).
15) Shu, D., Melhem, R. and Childers, B.R.: Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multiprocessor Real-Time Systems, *IEEE Trans. Parallel and Distributed Systems*, Vol.14, No.7, pp.686–700 (2003).
16) Hong, I., Qu, G., Potkonjak, M. and Srivastava, M.: Synthesis Techniques for Low-Power Hard Real-Time Systems on Variable Voltage Processors, *Proc. 19th IEEE Real-Time Systems Symposium*, pp.178–187 (1998).
17) Aydin, H., Melhem, R., Mosse, D. and Alvarez, P.M.: Determining Optimal Processor Speeds for Periodic Real-Time Tasks with Different Power Characteristics, *Proc. 13th Euromicro Conference on Real-Time Systems*, pp.225–232 (2001).
18) Aydin, H., Melhem, R., Mosse, D. and Alvarez, P.M.: Dynamic and Aggressive Power-Aware Scheduling Techniques for Real-Time Systems, *Proc. 22nd IEEE Real-Time Systems Symposium*, pp.95–105 (2001).
19) Shin, Y. and Choi, K.: Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems, *Proc. 36th Design Automation Conference*, pp.134–139 (1999).
20) Lindgren, M., Hansson, H. and Thane, H.: Using Measurements to Derive the Worst-Case Execution Time, *Proc. 7th International Conference on Real-Time Computing Systems and Applications*, pp.15–22 (2000).
21) Puschner, P.: A Tool for High-Level Lauguage Analysis of Worst-Case Execution Times, *Proc. 10th Euromicro Workshop on Real-Time Systems*, pp.130–137 (1998).
22) Bernat, G., Colin, A. and Petters, S.M.: WCET Analysis of Probabilistic Hard Real-Time Systems, *Proc. 23rd IEEE Real-Time Systems Symposium*, pp.279–288 (2002).
23) Holman, P. and Anderson, J.H.: Adapting Pfair Scheduling for Symmetric Multiprocessors, *Journal of Embedded Computing*, Vol.1, No.4, pp.543–564 (2005).

付　　録

A.1　均一電圧周波数制御の最適性

LLREF において，局所でスケジュール可能であるための必要十分条件は Critical Moment が発生しないことである[6]．Critical Moment とは，**図 18** のように，M 個を超えるトークンが同時に T-L Plane の斜辺に到達した時刻である．しかしながら，変形された T-L Plane のモデルでは必要十分条件とはならない．これは，設定された周波数によってはトークンが時刻 t_0 の時点ですでに T-L Plane の外部に存在している可能性があるためである．時刻 t_0 において，全トークンが T-L Plane の内部に存在するための条件を示す．

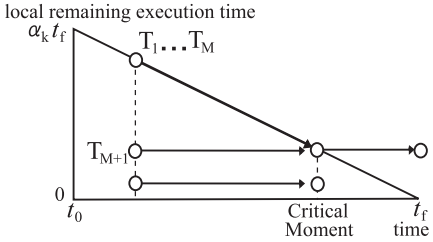

図 18　Critical moment
Fig. 18　Critical moment.

補題 4（トークンの初期位置）　時刻 t_0 に全トークンが変形された *T-L Plane* の内部に存在するための必要十分条件は，$X \leq \alpha$ である．

証明　時刻 t_0 におけるタスク T_i の局所の残り実行時間は $l_{i,0}$ である．これが T-L Plane の高さ αt_f 以下となれば T-L Plane の内部に存在する．したがって，$l_{i,0} \leq \alpha t_f$ となればよい．定義 $l_{i,0} = r_{i,0} t_f$ を代入することにより，$r_{i,0} \leq \alpha$ となる．$r_{i,0} = u_i$ であるため[6]，$X \leq \alpha$ であれば，時刻 t_0 にすべてのトークンが T-L Plane の内部に存在する．□

これ以降は $X \leq \alpha$ と仮定する．全トークンが T-L Plane の内部に存在するため，Critical Moment が発生しないことは局所でスケジュール可能であるための必要十分条件となる．これを以下に示す．

補題 5（Critical Moment）　変形された *T-L Plane* 内で局所でスケジュール可能であることと *Critical Moment* が発生しないことは等価である．

証明　全トークンが T-L Plane の内部に存在していると仮定しているため，元の T-L Plane と変形された T-L Plane は，同じ概念に基づいている．そのため，LLREF の証明[6] と同様である．□

Critical Moment が発生しない条件は，局所のプロセッサ使用率の和から導くことが可能である．

補題 6（Critical Moment が発生しない条件）　すべての整数 j について $S_j \leq \alpha M$ のとき，変形された *T-L Plane* において *Critical Moment* は発生しない．

証明　時刻 t_j に Critical Moment が発生したとき，$S_j > \alpha M$ であること示す．この対偶は，$S_j \leq \alpha M$ であるならば時刻 t_j に Critical Moment は発生しない，となる．したがって，すべての j について $S_j \leq \alpha M$ であるならば Critical Moment は発生しない．

Critical Moment が時刻 t_j に発生したとき，斜辺上に存在するタスクの局所の残り実行

時間は $\alpha(t_f - t_j)$ である．局所のプロセッサ使用率の和は，

$$S_j = \sum_{i=1}^{M} \frac{\alpha(t_f - t_j)}{t_f - t_j} + \sum_{i=M+1}^{N} \frac{l_{i,j}}{t_f - t_j} > \alpha M$$

となる．したがって，時刻 t_j に Critical Moment が発生したとき，$S_j > \alpha M$ である． □

以下の補題で S_j は単調減少であり，$S_0 \leq \alpha M$ であれば局所でスケジュール可能であることを示す．

補題 7（局所のスケジュール可能性） $S_0 \leq \alpha M$ であるならば *Critical Moment* は発生しない．

証明 帰納法により証明する．$S_0 \leq \alpha M$ であることから $S_{j-1} = S \leq \alpha M$ と仮定する．そして，$S_j \leq S$ であることを示す．帰納法の前提条件は，

$$\begin{aligned} & S_{j-1} = S \\ \Leftrightarrow & \sum_{T_i \in T} \frac{l_{i,j-1}}{t_f - t_{j-1}} = S \\ \Leftrightarrow & \sum_{T_i \in T} l_{i,j-1} = S(t_f - t_{j-1}) \end{aligned} \tag{1}$$

である．時刻 t_{j-1} から時刻 t_j までの間に N'（$\leq M$）個のタスクを実行可能であると仮定する．すべてのトークンは T-L Plane の内部に存在するため，補題 4 の証明と同様に $r_{i,j-1} \leq \alpha$ となる．$S \leq M$ と $r_{i,j-1} \leq \alpha$ より $S \leq \alpha N'$ が成立する．時刻 t_{j-1} から時刻 t_j までの間に残り実行時間は $\alpha N'(t_j - t_{j-1})$ だけ減少する．S_j は以下のよう算出される．

$$\begin{aligned} S_j &= \frac{1}{t_f - t_j} \sum_{T_i \in T} l_{i,j} \\ &= \frac{1}{t_f - t_j} \left(\left(\sum_{T_i \in T} l_{i,j-1} \right) - \alpha N'(t_j - t_{j-1}) \right) \end{aligned}$$

上式は，式 (1) より以下のように変形される．

$$S_j = \frac{S(t_f - t_{j-1}) - \alpha N'(t_j - t_{j-1})}{t_f - t_j}.$$

以上の結果より S_j は単調減少であることが導かれる．

$$\begin{aligned} & S_{j-1} - S_j \\ &= S - \frac{S(t_f - t_{j-1}) - \alpha N'(t_j - t_{j-1})}{t_f - t_j} \\ &= \frac{t_j - t_{j-1}}{t_f - t_j}(\alpha N' - S) \geq 0 \\ &\Rightarrow S_{j-1} \geq S_j \end{aligned}$$

前提条件 $S_0 \leq \alpha M$ と S_j が単調減少であることより，すべての j について $S_j \leq \alpha M$ である．したがって，補題 6 より Critical Moment は発生しない．すべてのトークンは，時刻 t_0 に T-L Plane の内部に存在するため局所でスケジュール可能である． □

LLREF のスケジュール可能性を示す．

定理 8（LLREF のスケジュール可能性） *LLREF* は，$\max\{U_s, X\} \leq \alpha$ を満たすいかなるタスクセットでもスケジュール可能である．

証明 $X \leq \alpha$ であるため補題 4 より，全トークンは変形された T-L Plane の内部に存在する．$U_s \leq \alpha$ より $U \leq \alpha M$ である．$U = S_0$ であるため[6]，$S_0 \leq \alpha M$ となる．補題 7 より Critical Moment は発生しないことから，補題 5 より局所でスケジュール可能である．局所でスケジュール可能であればすべてのデッドラインを守れることは明らかである． □

本手法は，周波数を独立に設定できないシステムに対して，最適であることを示す．

定理 9（DecideUniformFrequency の最適性） *DecideUniformFrequency* は，周波数を独立に設定できないシステムに対して，静的な実時間電圧周波数制御として最適である．

証明 DecideUniformFrequency よりも消費電力を削減するには，$U_s > \alpha$ か $X > \alpha$ を満たしても実時間性を保証可能なアルゴリズムでなくてはならない．$U_s > \alpha$ のとき，タスクセットの要求する時間が全プロセッサ時間を超えることから，実時間性を保証不可能である．$X > \alpha$ のとき，$u_i = X$ を満たすタスク T_i の要求する時間が 1 つのプロセッサ時間を超えることから，実時間性を保証不可能である．したがって，DecideUniformFrequency よりも消費電力を削減可能な静的アルゴリズムは存在しない． □

A.2　独立電圧周波数制御の最適性

DecideFrequency で電圧周波数制御を行ったとき，全タスクがスケジュール可能であることを示す．

補題 10（独立電圧周波数制御時のスケジュール可能性） *DecideFrequency* で電圧周波数制御を行ったとき，全タスクはスケジュール可能である．

証明 ヘビータスクは 1 プロセッサを独占して必ずスケジュール可能であるため，ライトタスクのスケジュール可能性を示す．記号は，DecideFrequency のヘビータスクとなる現在

の候補 T_i の分類操作前の状態を表すとき T のようにそのまま表し，分類操作後の状態を表すとき T' のようにダッシュを付加する．ライトタスクを実行するプロセッサの周波数を α とする．分類操作後に $\alpha' \leq 1$ であれば，定理 8 より分類後もライトタスクをスケジュール可能である．DecideUniformFrequency より，分類操作前後の周波数は，

$$\alpha = \max\left\{X^{\text{light}}, \frac{U^{\text{light}}}{M-H}\right\}$$

$$\alpha' = \max\left\{X^{\text{light}'}, \frac{U^{\text{light}'}}{M-H'}\right\}$$

$$= \max\left\{X^{\text{light}'}, \frac{U^{\text{light}} - X^{\text{light}}}{M-H-1}\right\}$$

となる．プロセッサ使用率が最大のライトタスク T_i をライトタスクセットから取り除くため，

$$X^{\text{light}} \geq X^{\text{light}'} \tag{2}$$

は明らか．DecideFrequency によりプロセッサ使用率が最大のライトタスク T_i をヘビータスクへ分類する条件は $X^{\text{light}} > U^{\text{light}}/(M-H)$ より，

$$\frac{U^{\text{light}}}{M-H} - \frac{U^{\text{light}} - X^{\text{light}}}{M-H-1}$$
$$= \frac{X^{\text{light}}(M-H) - U^{\text{light}}}{(M-H)(M-H-1)} > 0$$
$$\Rightarrow \frac{U^{\text{light}}}{M-H} > \frac{U^{\text{light}} - X^{\text{light}}}{M-H-1} \tag{3}$$

である．式 (2) と (3) より $\alpha' \leq \alpha$ である．分類操作前は $U_s \leq 1$ より $\alpha \leq 1$ であることから，$\alpha' \leq 1$ となる．よって，DecideFrequency がこの操作を繰り返してライトタスクをヘビータスクに分類していっても全タスクはスケジュール可能である．□

本手法が，マルチプロセッサ用の静的な実時間電圧周波数制御として理論的に最適であることを示す．

定理 11（DecideFrequency の最適性） *DecideFrequency* は，マルチプロセッサ用の静的な実時間電圧周波数制御として最適である．

証明　消費電力を周波数で表した関数 $g(\alpha_k)$ は，狭義凸関数かつ狭義単調増加である．そのため，$X \leq U^{\text{light}}/(M-H)$ のとき，DecideUniformFrequency が最適であることは明らか．DecideFrequency はこれと同じ制御を行うため最適である．

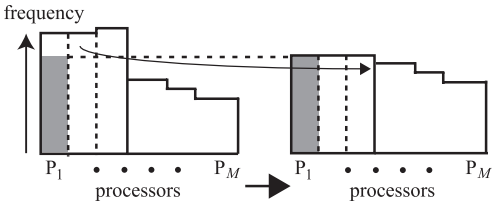

図 19　消費電力の小さな周波数
Fig. 19　Frequencies for lower energy consumption.

$X > U^{\text{light}}/(M-H)$ のときも最適であることを証明する．理想的には DecideUniformFrequency が最適であるが実時間性を保証することができない．DecideUniformFrequency で確実に実時間性を保証できない $u_m = X$ のタスク T_m をどのように実行するかに着目することにより，DecideFrequency は他のいかなる最適な実時間電圧周波数制御手法とも同じ消費電力を達成可能であることを示す．

T_m は最大のプロセッサ使用率のタスクであり，$g(\alpha_k)$ は狭義凸関数かつ狭義単調増加であるため，$\alpha_k > u_m$ のプロセッサが存在するとそのプロセッサの周波数を $\alpha_k = u_m$ とすることにより，必ずさらに消費電力の少ない周波数の組合せが存在する．**図 19** では，T_m のプロセッサ使用率を濃い長方形で示しており，必ず消費電力の少ない周波数の組合せが存在することを視覚的に表現している．このとき，$\alpha_k > u_m$ のプロセッサは存在しなくなるため，T_i を $\alpha_k < u_m$ のプロセッサで実行するとをデッドライン守れない．そのため，T_m が実行される 1 つ以上のプロセッサの周波数はすべて u_m と等しい．T_m が 1 プロセッサ上で実行されるときは DecideFrequency と同様の手法であるため，T_m が 2 以上のプロセッサ上で実行されると仮定する．このとき，T_m を実行するプロセッサの周波数は等しいことから，P_1 と T_m を切り離しても全タスクはスケジュール可能であり消費電力が変わらないことは明らか．この T_m と P_1 をシステムから切り離して始めから繰り返すことにより，DecideFrequency より消費電力を削減可能な静的アルゴリズムは存在しないことが示される．□

A.3　動的電圧周波数制御のスケジュール可能性

図 9 と図 10 に示した電圧周波数制御を時刻 t_0 と任意の時刻 $t_{j'}$ に行っても実時間性を保証可能であることを示す．**図 20** に電圧と周波数を制御する時刻 $t_{j'}$ を示す．重要な点は，時刻 $t_{j'}$ の厳密な時刻は示されておらず，時刻 $t_{j'}$ は $t_0 < t_{j'} < t_f$ を満たす任意の時刻を示

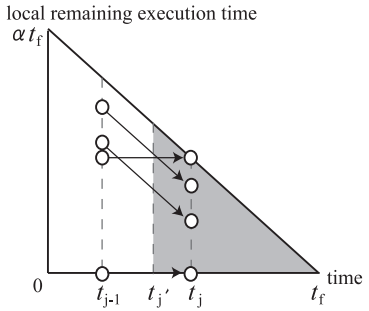

図 20　動的周波数制御を行う時刻 $t_{j'}$
Fig. 20　RT-DVFS performed at time $t_{j'}$.

すということである．定理 12 は，時刻 $t_{j'}$ に電圧と周波数を変更してもスケジュール可能であることを示すことにより，任意の時刻に電圧と周波数をアルゴリズムに基づいて変更可能であることを示している．

定理 12（動的電圧周波数制御時のスケジュール可能性） 全タスクは，任意の時刻に *DecideUniformFrequencyDynamic* や *DecideFrequencyDynamic* を行ってもスケジュール可能である．

証明　独立電圧周波数制御は，均一電圧周波数制御のボトルネックを解消する手法であるため，均一電圧周波数制御時にすべてのタスクがスケジュール可能であることを示すことにより，独立電圧周波数制御時もスケジュール可能であることが示される．時刻 t_0 に電圧と周波数の決定を行うことから，$Y_0 = X$ [6] と $Y_0 \leq \alpha$ より $X \leq \alpha$ である．補題 5 より全トークンは T-L Plane の内部に存在する．補題 7 より S_j は単調減少であることから，時刻 t_0 以降に電圧と周波数を変更しないとき $\forall j : S_j \leq \alpha M$ である．ここで時刻 t_{j-1} と時刻 t_j の間の時刻 $t_{j'}$（$t_{j-1} \leq t_{j'} < t_j$）に電圧と周波数の変更を行ったと仮定する．時刻 t_{j-1} の段階では電圧と周波数を変更していないため $S_{j-1} \leq \alpha M$ である．補題 7 の証明において t_j を $t_{j'}$ に置き換えることにより $S_{j-1} \geq S_{j'}$ となり，局所のプロセッサ使用率の和は時刻に対してつねに単調減少であることが示される．したがって，電圧と周波数を変更前は $S_{j'} \leq \alpha M$ である．このとき，補題 5 から時刻 $t_{j'}$ に Critical Moment は発生しない．そのため，全トークンは T-L Plane の内部に存在することから，補題 4 と同様に $Y_{j'} \leq \alpha$ である．ここで電圧と周波数の変更を行うと図 20 の影付きの三角形がアルゴリズムに従って変形されることになる．この状態は，静的な電圧周波数制御手法の証明において時刻 t_0 を時刻 $t_{j'}$ に読みかえたものと等しい．よって，全トークンは局所でスケジュール可能である．　□

(平成 20 年 1 月 29 日受付)
(平成 20 年 5 月 26 日採録)

船岡　健司（正会員）
1982 年生．2006 年慶應義塾大学理工学部情報工学科卒業．2008 年同大学大学院理工学研究科開放環境科学専攻修士課程修了．現在株式会社東芝勤務のかたわら，同大学大学院博士課程に在籍．リアルタイムシステム，オペレーティングシステム等の研究に従事．

加藤　真平（正会員）
1982 年生．2004 年慶應義塾大学理工学部情報工学科卒業．2008 年同大学大学院理工学研究科開放環境科学専攻博士課程修了．博士（工学）．現在同大学訪問研究員．リアルタイムシステム，オペレーティングシステム等の研究に従事．

山崎　信行（正会員）
1966 年生．1991 年慶應義塾大学理工学部物理学科卒業．1996 年同大学大学院理工学研究科計算機科学専攻博士課程修了．工学博士．同年電子技術総合研究所入所．1998 年 10 月慶應義塾大学理工学部情報工学科助手．同専任講師を経て 2004 年 4 月より同助教授．現在産業技術総合研究所特別研究員を兼務．並列分散処理，リアルタイムシステム，システム LSI，ロボティクス等の研究に従事．