Vol.2014-DBS-160 No.1
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.1
2014/11/18

マルチコア周期実行システムにおける省電力タスクスケジューリングの検討

柳橋 宏行[1,a)]　中田 尚[1]　中村 宏[1]

概要：組込みシステムにおいてプロセッサの消費エネルギを削減することは重要な課題となっている．本稿では，DVFS 制御が可能な同種のコアが複数搭載されたホモジニアスマルチコア環境において，確率的に実行時間が変動するような周期的なタスクへの省電力タスクスケジューリングについて検討する．これは，デッドラインまでの余裕時間に応じて，余裕がなくなれば稼働するコアの性能を上げ，余裕が生まれたら性能を下げるという，動的な性能の変更をすることで，デッドライン制約下での省電力化を実現する，というものである．これにより，組込みシステムにおいて省電力化を実現するための最適なコア構成を明らかにする．

1. はじめに

近年，マイクロプロセッサやメモリの製造技術の進歩によって家電製品や産業機器，医療製品に至るまで，さまざまなシステムに組込みシステムが用いられるようになった．また，組込みシステムの高性能化および小型化にともない，バッテリ駆動の組込みシステムが増加している．バッテリ駆動のシステムにとっては，システムの利便性の観点から要求される性能を満たした上で，消費エネルギを削減することは重大な課題である．

組込みシステムにおける消費エネルギは，システムによって多少の差異は見られるものの，概してプロセッサによって消費されるエネルギが大部分である．したがって，省エネルギ化を図るためには，プロセッサにおける消費エネルギの削減が効果的である．しかし，プロセッサの性能と消費エネルギ効率の間にはトレードオフの関係があり，高性能なプロセッサを用いるほど，同一タスクに対する消費エネルギが増加する．よって，要求性能を満たすような性能をもつプロセッサを搭載した上で，タスクの実行時間の変動に応じて，適切なプロセッサを選択することで全体としての消費エネルギを削減する必要がある．

また，近年の性能要求の増加と製造単価の減少に伴って，組込みシステムにおいても複数のプロセッサを搭載したマルチコアプロセッサが広く使われるようになっており，マルチコア環境における省電力スケジューリングの実現が強く求められている．

本稿で扱うタスクは，入力データに依存して，実行時間が変化するようなタスク（**動的変動タスク**）とする．また，入力データは周期的に到着する周期的なリアルタイムシステムとし（**周期実行システム**），入力周期よりもデッドラインが長いシステムを対象とする．

一般のリアルタイムシステムにおいては，エネルギ削減の手法として **DVFS**（Dynamic Voltage and Frequency Scheduling）や **DPM**（Dynamic Power Management）が用いられてきた．DVFS は，プロセッサの負荷が低いときに電圧および周波数を低下させることで，プロセッサの消費電力を削減する手法である．リアルタイムシステムでは，タスクがデッドラインまでに完了することができる範囲で，できるだけ低い電圧と周波数で動作させることで低消費電力化が図れる．一方，DPM は，プロセッサがアイドル状態のときにプロセッサ全体の電源を遮断することで，エネルギを削減する手法である．リアルタイムシステムでは，タスクが完了してから次のタスクを処理するまでの間にプロセッサの電源を遮断することで，無駄な待機電力を削減することができる．ただし，前者の手法はリーク電力を削減する効果はなく，後者の手法は電源の ON/OFF を切り替える際に，エネルギオーバーヘッドが生じてしまう．

また，消費エネルギの削減を目的とした動的変動タスクのスケジューリングにあたり，デッドラインまでの余裕時間に応じて，実行中に稼動するコアを変更する，といった手法が提案されている [1]．これは，性能の異なる異種のコアが混在して搭載されているシステムに対し，デッドラインまでの余裕時間に応じて，実行するコアを変更していく

[1] 東京大学大学院 情報理工学系研究科 システム情報学専攻
7-3-1 Hongo, Bunkyo, Tokyo 113–8656, Japan
[a)] yanagi@hal.ipc.i.u-tokyo.ac.jp

Vol.2014-DBS-160 No.1
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.1
2014/11/18

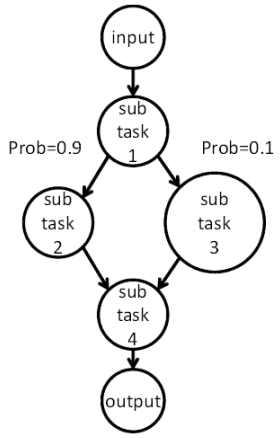

図 1 タスクグラフの例

手法である．この手法は同時に稼動するコアを最大 1 つに限定しており，離散的な DVFS 制御が可能なプロセッサ 1 つが搭載されたシステムに対して，デッドラインまでの余裕時間に応じて，性能を変更しながら消費エネルギの削減を図る，という手法と同等と見ることができる．

本稿では，DVFS 制御が可能なプロセッサが複数搭載されたマルチコアシステムにおける省電力スケジューリングを提案する．これは，離散的な DVFS 制御が可能なプロセッサが複数搭載されたホモジニアスマルチコアプロセッサにおいて，各コアで実行中のタスクのデッドラインまでの余裕時間を足し合わせた総余裕時間を見ることで，性能および稼動コア数を変更していくというものである．さらに，動的変動タスクの変動確率から求まる性能・稼動コア数変更確率を用いて，消費エネルギ期待値を評価し，消費エネルギ期待値を最小化するようなスケジューリングを導く．

2. 対象システムと既存省電力化手法

2.1 対象システム

2.1.1 タスクモデル

前述のように，本稿では周期実行システムのうち，デッドラインが入力周期よりも十分に長いシステムを扱う．一つの出力には一つの入力データしか用いない，単入力単出力系を対象とし，処理内容，処理時間は直前までの処理によって影響は受けないものとする．また，入力データの周期は一定であるとする．

一般に，タスクはそれ以上分割できない処理（**サブタスク**）をノードとしたタスクグラフを用いて表現できる．タスクグラフでは，ノードの大きさはタスクサイズを，有向エッジはサブタスクの順序関係を表している．サブタスクには，入力データによって処理内容や処理時間が変化するものが存在する．このようなサブタスクを含むタスクを**動的であるといい，動的なサブタスクを含んだタスクを**動的変動タスク**という．これに対し，処理時間が入力データによらずに一定であるサブタスクを**静的であるという．本稿では，動的なサブタスクは実行が完了するまでに実行時間を知ることはできないとし，サブタスクの実行時間の変動確率は既知であるとする．動的なサブタスクは，**図 1** のようにタスクグラフの分岐を用いて表現できる．変動しうる実行時間の数だけ分岐を設けて，いずれかのパスのみが選択される，とすることで，静的なサブタスクのみでタスクグラフを表すことができ，各分岐の分岐確率が，実行時間の変動確率となる．本稿では，並列実行可能なタスクは扱わず，タスクグラフにおける枝分かれは，いずれかのパスのみが選択され実行されるということに注意する．

2.1.2 ハードウェア構成とプロセッサモデル

本稿で対象とするハードウェア構成は，離散的な DVFS 制御が可能なプロセッサが複数搭載された，ホモジニアスマルチコアプロセッサを想定する．また，入出力データが読み書きされるメモリは各コアからアクセスできるものとする．

DVFS 可能なプロセッサは，周波数が f_1 から f_m までの m 段階の離散的な周波数を用いることができるとする（$f_1 < f_2 < ... < f_m$）．最小周波数 f_1 および最大周波数 f_m の値が満たすべき条件は，後のスケジューリング手法の説明で述べる．以降では，この m 段階の DVFS が可能な同種のプロセッサが N 台搭載されているホモジニアスマルチコアプロセッサを仮定する．

2.2 消費エネルギモデル

一般に，プロセッサの消費エネルギは以下の式で表される [2]．

$$E_{proc} = \alpha_1 T_1 C V^2 f + T_2 V I_{leak} \qquad (1)$$

第 1 項はタスクの実行する際に消費されるダイナミックエネルギを表している．ただし，式中の T_1 はタスクの実行時間，C は回路の負荷容量，V は電源電圧，f は動作周波数，α_1 は比例定数である．動作周波数を大きくして性能を上げると，プロセッサの消費ダイナミックエネルギも増大する．第 2 項はスタティックエネルギを表している．スタティックエネルギは，回路内にリーク電流が流れることから消費されるエネルギであり，動作/非動作に関わらず電源が入っている限り常に消費される．ただし，式中の T_2 はプロセッサの稼動時間，V は電源電圧，I_{leak} はリーク電流である．リーク電流は回路の微細化につれ増加するため，プロセッサが高性能であるほどスタティックエネルギも増加する．

同じ動作周波数で動作させるとき，プロセッサの性能はその面積の平方根に比例することが経験則（ポラックの法則）として知られており [3]，同面積で比較すると，シングルコアで動作させるときよりも性能の小さいコアを複数搭載したマルチコアで動作させたときのほうがエネルギ効率

Vol.2014-DBS-160 No.1
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.1
2014/11/18

が良い．よって，マルチコアにおいて適切に並列化ができれば，同性能のシングルコアと比較して，1タスクあたりのエネルギ消費を抑えることができる．

2.3 DVFS

プロセッサの消費ダイナミックエネルギを削減する方法としてDVFS（Dynamic Voltage and Frequency Scheduling）が存在する．DVFSでは，動作中のプロセッサに対し，負荷に応じて動作周波数（およびそれに付随して電圧）を変更することができる．よって，負荷が小さいときには動作周波数を小さくして供給電圧を下げることで，消費されるダイナミックエネルギを抑えることができる．ただし，制約（デッドライン）を守らなければならないことからくる制御の複雑さや，動作周波数の切り替えに伴うオーバーヘッドの影響も考慮しなければならない．

2.4 DPM

プロセッサの消費スタティックエネルギを削減する方法としてDPM（Dynamic Power Management）が存在する．DPMは，プロセッサの非動作時に流れる電流を遮断することでスタティックエネルギを削減する手法である．電流が遮断されている間は，プロセッサは処理を行うことができない．以降，本稿ではプロセッサについて通常の動作が可能な状態を**アクティブ状態**，電流が遮断されて処理できない状態を**省電力状態**と呼ぶ．こちらも状態の遷移に伴うオーバーヘッドエネルギが伴うため，期待できるスタティックエネルギの削減量と比較して遷移を行うべきか判断する必要がある．

2.5 既存スケジューリング

本稿の提案スケジューリングの検討にあたって，非常に関連の深い手法[1]を紹介する．

[1]では動的省電力スケジューリングが提案されており，動的変動タスクを扱う周期実行システムにおいて，デッドラインまでの余裕時間に応じてプロセッサの性能を変更することで，消費エネルギの削減を図る，というものである．ここで扱っているタスクモデルは本稿で扱うタスクモデルと全く同様で，デッドラインが入力周期よりも十分に長い動的変動タスクを扱う．プロセッサの構成に関しては，異種のコアが混在するヘテロジニアスマルチコアプロセッサを用いるが，同時に稼動するコアは最大で一つのみとしている．

デッドラインが長い場合のスケジューリングとして，タスクの実行開始を入力到着直後よりも遅らせて，後続の複数のタスクと連続して実行する，というタスクのまとめ実行が提案されている[4]．**図2**のように，まとめ実行をすることでDPMを行う際のアクティブ状態と省電力状態の遷移回数を減らすことができ，赤色で示されたオーバーヘッドエネルギが3分の1に削減されている．

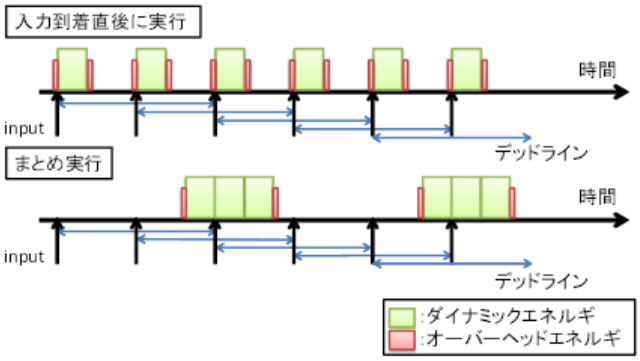

図2 まとめ実行の様子

表1 入力変数

変数	意味
M	タスクの変動パターン数
$S(i)$	パターン i のタスクサイズ
$p(i)$	パターン i が実行される確率
I	入力周期
d	デッドライン
$et(i,j)$	DVFSコアが周波数 f_j でパターン i を実行するのに要する時間
$E_d(i,j)$	DVFSコアが周波数 f_j でパターン i を実行するのに要するダイナミックエネルギ
$P_{S_a}(j)$	DVFSコアが周波数 f_j でのアクティブ状態のスタティック電力
P_{S_s}	DVFSコアが省電力状態のスタティック電力
$E_{ov}(j,k)$	DVFSコアが周波数 f_j から周波数 f_k へ遷移するのに要するオーバーヘッドエネルギ（ただし，f_0 は省電力状態を指すとする）

[1]の動的省電力スケジューリングでは，このまとめ実行を採用しており，タスクはまとめられるだけまとめて実行し，タスク実行終了時に他の実行可能なタスクがある場合は続けて実行する．そして，実行中におけるデッドラインまでの余裕時間に関して，余裕があるときは低性能のコアで実行し，余裕がなくなると高性能のコアで実行するというスケジューリングを行うことで消費エネルギを削減する．

そうしたスケジューリングを行うためには，デッドラインまでの余裕時間がどの値を超えたら稼動コアの性能を下げ，どの値を下回ったら稼動コアの性能を上げるか，という稼動コアの変更点を決定する必要がある．このとき，消費エネルギの期待値を最小化するような稼動コアの変更点を決定することで，省電力化を図る．

3. 提案スケジューリング

3.1 問題設定

本節では，設計時に既知のパラメタを列挙し，本稿での問題の目的関数と制約条件を述べる．入力変数は**表1**にまとめた．

Vol.2014-DBS-160 No.1
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.1
2014/11/18

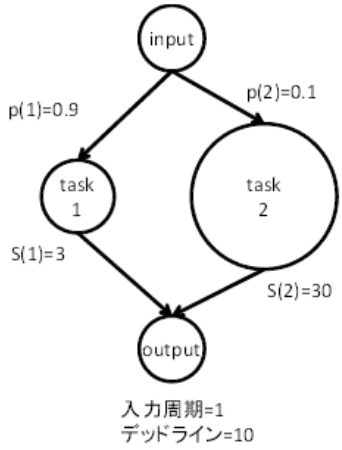

図 3 変動タスクの例

表 2 各周波数での実行時間

周波数	サブタスク 1 の実行時間	サブタスク 2 の実行時間
f_1	0.6	5.4
f_2	0.3	2.7
f_3	0.2	1.8

タスクに関するパラメタは，データの入力周期 I，デッドライン d，変動パターン数 M，各タスクパターンの ID を $i = 1, 2, ..., M$ として，変動確率 $p(i)$，タスクサイズ $S(i)$ が与えられる．変動パターン数は，タスクグラフにおけるパスの通り方の総数を表しており，変動確率は各パスを通る確率を表す．タスクサイズは各パターンを実行したときの命令数に相当するパラメタである．**図 3** は図 1 を書き直したもので，この例では各パラメタは，$I = 1, d = 10, M = 2, p(1) = 0.9, p(2) = 0.1, S(1) = 3, S(2) = 30$ となる．

離散的な DVFS 制御可能なコアについて，変更可能な周波数を低いものから順に，$f_1, f_2, ..., f_m$ とする．このとき周波数 f_j でタスクのパターン i を実行したときの実行時間を $et(i, j)$，実行に要するダイナミックエネルギを $E_d(i, j)$，周波数 f_j 時のアクティブ状態時スタティック電力を $P_{S_s}(j)$，周波数 f_j から周波数 f_k へ遷移するのに要するオーバーヘッドエネルギを $E_{ov}(j, k)$ とする．ただし，最後の $E_{ov}(j, k)$ に関しては，f_0 は省電力状態を表すとする．これらのパラメタは事前に得られるものとし，状態の移行に要する時間のオーバーヘッドは無視できるものとする．この DVFS コアと同種のコアが N 個搭載されているホモジニアスマルチコアシステムを考える．

確率的に実行時間が変動するタスクを扱うため，本稿では目的関数は「プロセッサ全体の消費エネルギの期待値」，制約条件は「全てのタスクがデッドラインを満たす」とする．

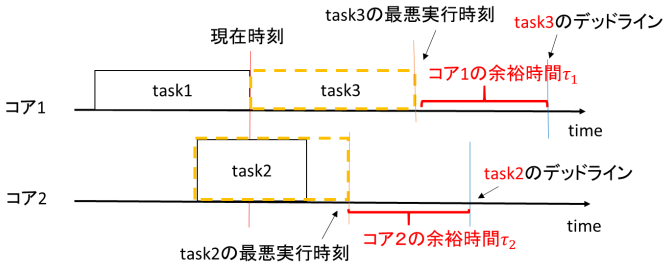

図 4 余裕時間の概念図

3.2 省電力タスクスケジューリング

3.2.1 概要

本稿で提案するスケジューリングでは，タスクはまとめられるだけまとめて実行するものとし，あるタスクの実行終了時に他の実行可能なタスクがある場合は，続けて実行する．

m 段階の DVFS 制御可能なコアが N 個搭載されているとするとき，最低周波数 f_1 および最高周波数 f_m での性能が満たすべき条件を考える．この構成においては N 個のコアが全て f_m で稼動しているときが最も性能が高い．よって，このときに最悪のパターンを連続して実行することになってもデッドラインが守られるためには，一つのコアが周波数 f_m で稼動するときの最悪実行時間を N で割った値が入力周期よりも下回っている必要がある．これは，コアが N 個あることによって，1 つだけで処理するときと比べてスループットが N 倍になっていると考えると分かりやすい．また，最小周波数 f_1 に関しては，いずれのタスクのパターンを実行しても実行時間がデッドラインを超えない周波数のうち，最も低いものである．本稿では，入力から出力までを 1 つのコアで実行するため，これよりも低い性能の周波数で実行してしまうと，必ずデッドラインを超えてしまうことになる．

図 3 のような，高確率で小タスク，低確率で大タスクとなるような動的変動タスクに対し，**表 2** のような周波数で DVFS 制御が可能なコアが 2 個ある ($N = 2$) とする．このとき，2 個のコアがともに周波数 f_3 で稼動している最高の性能状態を考えると，2 個のコアを合わせたときのスループットを考えて，入力周期 1 に対し，最悪実行時間が 1.8 であることから，$1.8/2 = 0.9$ と，入力周期を下回っていることから，最悪パターンが連続で到着したとしても必ずデッドライン以内に終わらせることができる．しかし，この例のように，小タスクが実行される確率が高い場合には，常に最高の性能で稼動し続けるのではなく，余裕があるときは性能を下げて実行することで，消費エネルギを削減することができると考えられる．

3.2.2 総余裕時間

タスクの実行中，余裕に応じて性能の変更を行うとするとき，以下で定義する**総余裕時間**を余裕の定義とする．

Vol.2014-DBS-160 No.1
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.1
2014/11/18

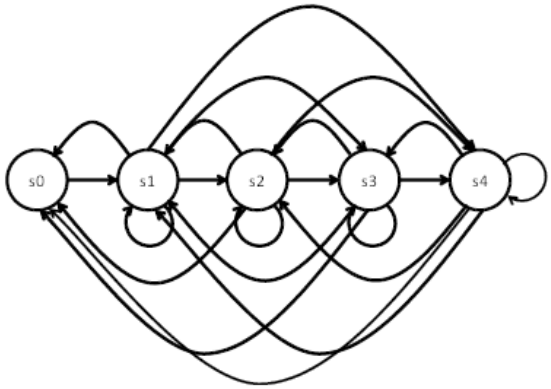

図 5 状態遷移図（状態数 N'=5 の場合）

$$t_{total} = \sum_{1 \leq l \leq N} \tau_l \tag{2}$$

ただし，τ_l はコア l における，実行するタスクのデッドライン時刻から，最悪実行時間で実行完了時刻を引いた余裕時間であり，**図 4** はその例である．

3.2.3 状態と状態遷移

プロセッサの性能に対応した状態を定義する．本稿では，N 個のコアのうち，何個のコアがどの周波数で稼動しているか，についての場合を表したものを性能の状態と定義する．これは，DVFS 制御可能な周波数が m 段階で，省電力状態 f_0 も加えた $m+1$ 種類の周波数から，重複を許して N 個取り出す重複組み合わせに等しいので，状態の組み合わせは以下のようになる．

$$_{m+1}\mathrm{H}_N = {}_{m+N}\mathrm{C}_N \tag{3}$$

例えば，$m = 3$, $N = 2$ の場合を全て書き下すと，(f_0, f_0)，(f_0, f_1)，(f_0, f_2)，(f_0, f_3)，(f_1, f_1)，(f_1, f_2)，(f_1, f_3)，(f_2, f_2)，(f_2, f_3)，(f_3, f_3) の ${}_{3+2}\mathrm{C}_2 = 10$ 通りとなる．

この状態に対し，各周波数での実行時間の逆数の和を取ったものをその状態の性能とする．これはプロセッサ全体のスループットを表している．状態を性能の順に並べて順に $s_0, s_1, ...$ と名付ける．これらの状態間の遷移を状態遷移図で表すと**図 5** のようになる．

状態遷移図を描く際，エネルギー効率を考えることで，状態数を削減することが考えられる．たとえば，全ての状態の中で性能が等しいものが存在したとしたら，その状態での消費電力の低いものを残し，消費電力の高いものを状態から削除することで，予め状態数を削減することができる．具体的に表 2 のような周波数とタスクの実行時間の関係があった場合，状態 (f_2, f_2) と状態 (f_1, f_3) の性能は等しくなるが，ダイナミックエネルギは前者のほうが小さくなることを以下に説明する．

同種の DVFS コアで比較した場合，プロセッサ全体のスループットは各コアの周波数の和に比例すると考えられるので，$2f_2 = f_1 + f_3$ の等式が成り立つ．また，DVFS コアの場合，一般的に供給電圧は周波数に比例するので，$V \propto f$ となる．式 (1) のダイナミックエネルギの項より，ダイナミック電力は $\alpha_1 CV^2 f$ で与えられるので，これを周波数 f の関数 $g(f)$ であると見ると，$g(f)$ は f に関して下に凸な関数であることから，以下の式が成り立つ．

$$\frac{g(f_1) + g(f_3)}{2} \geq g\left(\frac{f_1 + f_3}{2}\right) = g(f_2)$$

$$\Leftrightarrow g(f_1) + g(f_3) \geq 2g(f_2) \tag{4}$$

これは状態 (f_2, f_2) のダイナミックエネルギが状態 (f_1, f_3) のダイナミックエネルギよりも小さいことを示しており，状態 (f_2, f_2) を選択するほうが有利であることを示す．よって状態 (f_1, f_3) を削除して，状態数を減らすことができる．

これと同様の議論をすることで，性能の等しいものに関しては，2 段階以上離れた周波数を組み合わせるよりも，段階の近い周波数を組み合わせたほうが，よりエネルギー効率がよい．よって状態数を予め減らして考えることが可能となる．

3.2.4 定式化と求解

本稿で提案する省電力タスクスケジューリングでは，[1] で提案されている定式化と求解の手法に基本的に従う．よって本節では，[1] における定式化と求解の手法を基にした手法を述べる．[1] と異なる点は，複数のコアが同時に稼動しているために，あるタスクが終了したとしても，他のタスクが他のコアで実行されていることがあるという点である．

先ほど定義した総余裕時間に応じて，余裕があるときは性能の低い状態へ遷移し，余裕がないときは性能の高い状態へと遷移する．したがって，解決すべき問題は，総余裕時間がどの値を超えたらより性能の低い状態へと遷移し，どの値を下回ったらより性能の高い状態へと遷移するのか，という状態の変更点を決定する問題へと帰着する．最適な状態の変更点を決定することで，消費エネルギの期待値を最小化することを目的とする．

目的関数の定式化にあたり，**表 3** に示すパラメタを導入する．前節で述べた状態の総数を N' とすると，状態は $s_0, ..., s_{N'-1}$ と表せる．ただし，状態 s_0 は全てのコアが省電力状態であるものとし，s_0 以外の状態では少なくとも 1 つのコアはアクティブ状態であることに注意する．総余裕時間に応じてプロセッサの状態を変更するが，実行中の総余裕時間の計算は，1 タスクが終了するたびに行うこととする．計算された余裕時間が変更点を超えていたとき，状態の変更を行う．状態の変更は段階的な変更を想定し，主に性能が隣り合う状態への変更を考えるが，性能を一気に変更することも許す．また，状態を変更するときの周波数の変更は，タスクの実行が終了したコアでも，他のタスク

Vol.2014-DBS-160 No.1
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.1
2014/11/18

表 3　定式化にあたり導入したパラメタ

変数	意味
N'	状態数
P	遷移確率行列
π	P に従う定常分布
t_{active}	平均アクティブ時間
t_{sleep}	平均スリープ時間
E_D	1 入力周期あたりの各状態の平均アクティブ時スタティックエネルギ
E_{S_a}	1 入力周期あたりの各状態の平均アクティブ時スタティックエネルギ
E_{S_s}	1 入力周期あたりの状態 s_0 での平均スタティックエネルギ
E_{OV}	各状態から各状態へ遷移するのに要するオーバーヘッドエネルギ
x^i_{up}	状態 s_i から状態 s_{i+1} への変更点
x^i_{down}	状態 s_{i+1} から状態 s_i への変更点
$s_\star$	まとめ実行を開始する状態

を実行しているコアでもよい．変更すべき周波数のコアが複数存在するとき，性能を下げるときはその中で一番新しいタスクを実行している（する）コアを，性能をあげるときはその中で一番古いタスクを実行しているコアを変更するものとする．

状態の変更の様子を状態遷移図で表すと，図 5 のように N' 状態の状態遷移図で表される．図 5 中の状態 s_0 からの遷移が状態 s_1 に限られているのは，まとめ実行を開始する状態を状態 s_1 に設定しているためである．ただし，まとめ実行を開始する状態は任意に選択することができる．

性能の状態の変更を，このような状態遷移図と捉えると，各状態間の遷移確率行列 P を定義できる．P の (i,j) 成分は，状態 s_i から状態 s_j に遷移する確率，すなわち状態 s_i であるタスクが完了した際に，状態 s_j へと遷移する確率を表す．P は $N' \times N'$ の行列であり，行和は 1 となる．

また，遷移確率行列 P に対し，定常分布 π を定義することができる．定常分布 π は $1 \times N'$ のベクトルで，$\pi P = \pi$ を満たすような分布であり，各状態に平均してどの程度分布しているかを総和が 1 になる割合として表したものと考えることができる．

これらに加えて，まとめ実行開始から終了までに要する平均時間を，平均アクティブ時間 t_{active}，まとめ実行終了から次のまとめ実行を開始するまでの平均時間を，平均スリープ時間 t_{sleep} とする．

目的関数として，消費エネルギの期待値を定式化する．しかし，タスクの実行を限りなく繰り返したとき，消費エネルギの期待値は無限に発散してしまう．そこで，入力周期を時間幅として区切り，時間幅内で消費されるエネルギの期待値を目的関数として定義する．遷移確率行列 P，定常分布 π，平均アクティブ時間 t_{active}，平均スリープ時間 t_{sleep} を用いて，目的関数 J をパラメタ表示すると，以下

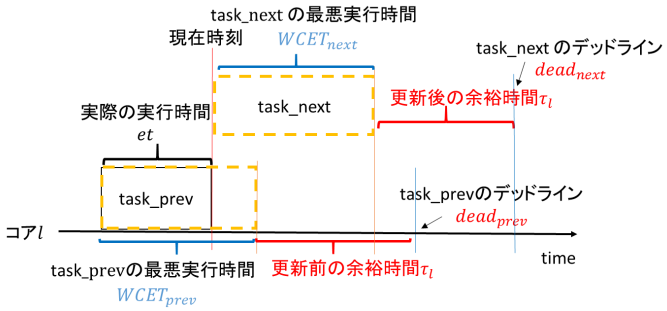

図 6　総余裕時間の更新の様子

のように表される．

$$J = \pi E_D + \pi E_{S_a} + ||\pi(P * E_{OV})|| + \frac{t_{sleep}}{t_{active} + t_{sleep}} E_{S_s} \tag{5}$$

ただし，$*$ は行列の要素ごとの積を表し，$||\cdot||$ は要素の総和を表す．E_D，E_{S_a} は，ともに $N' \times 1$ のベクトルであり，第 i 成分は，それぞれ，状態 s_i で 1 入力周期あたりの平均ダイナミックエネルギ，平均スタティックエネルギを表す．E_{OV} は，$N' \times N'$ の行列であり，(i,j) 成分は，状態 s_i から状態 s_j に遷移するのに要するオーバーヘッドエネルギを表す．また，E_{S_s} は定数であり，状態 s_0 での 1 入力周期あたりに消費される省電力時平均スタティックエネルギを表す．したがって，各項は順にプロセッサ全体の，1 入力周期あたりの平均ダイナミックエネルギ，1 入力周期あたりの平均アクティブ時スタティックエネルギ，1 遷移あたりの平均オーバーヘッドエネルギ，1 入力周期あたりの省電力時スタティックエネルギを表している．

定常分布及び平均アクティブ/スリープ時間は遷移確率行列を用いて計算されるため，式 (5) を最小にする遷移確率行列を与え，その遷移確率に従うように稼動コアを変更させれば，消費エネルギの期待値を最小化することができる．

以上をまとめると，評価関数 J を最小にするような遷移確率行列 P を与えるような，状態の変更点を求めればよい．状態 s_i から状態 s_{i+1} への変更点を x^i_{up} とし，状態 s_{i+1} から状態 s_i への変更点を x^i_{down} とする．また，まとめ実行を開始する際に使用する状態，すなわち，状態 s_0 から遷移する状態を $s_\star$ とする．このとき，状態 s_i から状態 s_{i+1} への変更点 x^i_{up} と，状態 s_{i+1} から状態 s_i への変更点は異なる値をとってもよい．一度状態 s_i に遷移すると，総余裕時間が x^i_{up} から x^{i-1}_{down} の間は，状態 s_i で実行し続ける．この区間を，ステージ i と呼ぶ．x^0_{up} は，まとめ実行開始時の総余裕時間を表す．また，x^0_{down} は，状態 s_0 への遷移であることから，まとめ実行の終了タイミングを表している．タスクはまとめられるだけまとめて実行するため，まとめ実行の終了タイミングは実行可能タスクがなくなったときである．

まとめ実行が開始されてから n 回目のタスク終了がコア

Vol.2014-DBS-160 No.1
Vol.2014-OS-131 No.1
Vol.2014-EMB-35 No.1
2014/11/18

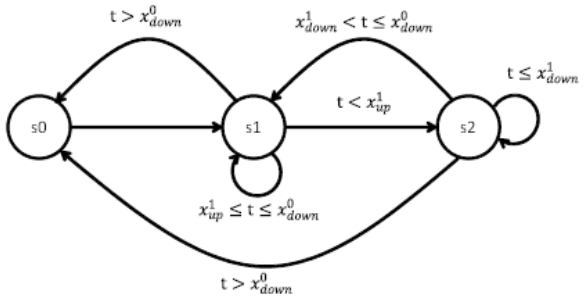

図 7 状態遷移と遷移条件

l でなされたときの総余裕時間を t_n とすると，その計算式は以下のように与えられる．

$$t_n = t_{n-1} + (WCET_{prev} - et) - WCET_{next} + (dead_{next} - dead_{prev}) \quad (6)$$

ただし，コア l において実行終了したタスクを $task_{prev}$，コア l で次に実行するタスクを $task_{next}$ とし，それらのデッドラインおよび最悪実行時間を $dead_{prev}$，$WCET_{prev}$ のように表す．また，$task_{prev}$ の実際の実行時間を et と表す．ここで，コアが周波数 f_j で稼動しているときのタスクの最悪実行時間は次で与えられる．

$$WCET = \max_{1 \leq i \leq M} et(i, j) \quad (7)$$

式（6）の更新の例を図で表すと**図 6** のようになる．t_n が x^i_{up} または x^{i-1}_{down} を超えたとき，次の状態へと遷移する．状態数 $N' = 3$ のときの遷移条件を状態遷移図で表すと**図 7** のようになる．

状態変更点 x^i_{up}，x^i_{down} および実行開始状態 $s_\star$ から，タスクの変動確率を用いて，遷移確率行列 P を求めることができる．また，P から定常分布 π，および平均アクティブ時間 t_{active}，平均スリープ時間 t_{sleep} を求めることができる．したがって，評価関数 J を計算することができ，状態変更点および実行開始状態の評価が可能となる．

具体的な状態変更点および実行開始状態の最適値は，遺伝的アルゴリズムを用いて求める．

4. まとめ

本稿では，DVFS 制御が可能な同種のコアが複数搭載されたホモジニアスマルチコア環境において，確率的に実行時間が変動するような周期的なタスクを実行する際の省電力タスクスケジューリングについて検討した．そして，デッドラインを考慮した余裕時間に応じて，余裕がなくなれば稼働するコアの性能を上げ，余裕が生まれたら性能を下げるという，動的な性能の変更をすることで，デッドライン制約下での省電力化を実現する，という手法を提案した．

今後の課題として，提案した手法を元に消費エネルギ期待値を計算し，消費エネルギ期待値の削減効果があるか確認することや，実機にこの手法を適用して，実際に消費エネルギを計測して削減効果を確かめる，といったことが挙げられる．

謝辞 本研究の一部は，NEDO「ノーマリーオフコンピューティング基盤技術開発」事業による．

参考文献

[1] 畑中 智貴，中田 尚，中村 宏. 周期実行システムにおける動的省電力タスクスケジューリング，情報処理学会研究報告，Vol.2014-EMB-32, No.4, pp.1–6, 2014.
[2] Thomas.D. Burd and Robert.W. Brodersen. “Energy efficient cmos microprocessor design”. Hawaii International Conference on System Sciences, p. 288, 1995.
[3] Fred Pollack, Intel Corp. “Micro32 conference keynote” 1999.
[4] Takashi Nakada, Kazuya Okamoto, Toshiya Komoda, Shinobu Miwa, Yohei Sato, Hiroshi Ueki, Masanori Hayashikoshi, Toru Shimizu, Hiroshi Nakamura : Design Aid of Multi-core Embedded Systems with Energy Model, IPSJ Transactions on Advanced Computing Systems, Vol.7, No.3 (ACS46), pp. 37–46, (2014)