Vol.2015-SLDM-170 No.5
Vol.2015-EMB-36 No.5
2015/3/6

マルチレート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

藤原 賢[1,a)]　柳橋 宏行[1]　中田 尚[1]　中村 宏[1]

概要：動的変動タスクを扱う周期実行システムにおいては，ヘテロジニアスマルチコアプロセッサを用いて，タスクの負荷要求に応じて処理するコアを使い分けることが，省電力化のために有効である．本研究では，周期の異なる（マルチレート）複数の周期タスクが実行されるシステムにおいて，デッドラインまでの余裕がなくなれば，性能を上げ，余裕が生まれると，性能を下げる，というように動的に稼動コアを変更する手法を提案する．既存の単一タスク向けスケジューリングをタスクを拡張する手法と状態遷移を拡張する手法の 2 種類の方法で拡張することによりマルチレートシステムのスケジューリングを実現し，その省電力効果を比較する．

1. はじめに

組込みシステムの高性能化，小型化により，バッテリ駆動の組込みシステムが増加しているとともに，システムの高度化が進んでいる．バッテリ駆動のシステムにとって，バッテリ寿命はシステムの利便性に直結するため，要求される性能を満たしつつ，消費エネルギを削減することは，重要な課題である．システムによって多少の差異はあるが，組込みシステムにおける消費エネルギは，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を削減するのが効果的である．

プロセッサの性能と消費エネルギ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使用するほど，消費エネルギは増加する．近年，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，マルチコアプロセッサが広く使われるようになった．特に，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**を用いると，タスクの負荷要求に応じてコアへの割り当てを選択することができ，エネルギ効率を改善させることができる．

また，システムの高度化によりその上で実行されるタスクも複雑化しており，お互いに周期の異なる（**マルチレート**）複数のタスクが同時に実行されることもめずらしくなくなっている．

本研究では，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**）を扱う．また，入力データは周期的に訪れるものとし（**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．

実行を行うコアの候補が複数かつ実行すべきタスクも複数あり，また，デッドラインが入力周期よりも長い場合は，どのコアでいつ開始させるか，の選択肢が増えるため，消費エネルギが最小となるようなスケジューリングを求めるのは難しく，搭載するプロセッサの選択も合わせると，最適なハードウェア構成とスケジューリングの組み合わせを導くのは困難を極める．

以上の問題に対し，本研究ではこれまでに提案されている単一周期タスク向け省電力タスクスケジューリング [2] を拡張し，デッドラインまでの余裕時間に応じて，実行中に稼動コア変更を行う，という手法をマルチレート動的変動タスクに拡張する．また，タスクの変動確率から求まる稼動コアの変更確率を用いた，消費エネルギ期待値の評価方法を提案し，最適なスケジューリングを導く．本手法を用いることで，プロセッサの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実現し，また同時に，ヘテロジニアスマルチコアプロセッサを含むプロセッサ候補の中から，最適なハードウェア構成を選択することが可能となる．

2. 組込みシステムの省電力化手法

2.1 対象システム

本研究で対象とするハードウェア構成は，マルチコア，特にヘテロジニアスマルチコアプロセッサを想定する．組込みシステムでは，あらかじめ扱うタスクがわかっており，

[1] 東京大学
7-3-1 Hongo, Bunkyo, Tokyo 113–8656, Japan
[a)] fujiwara@hal.ipc.i.u-tokyo.ac.jp

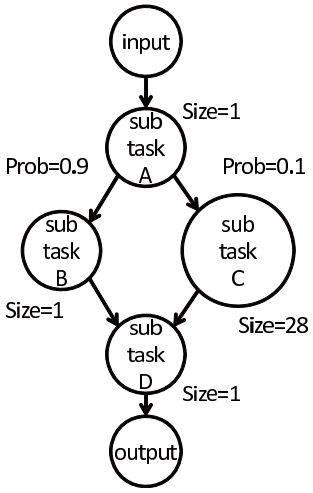

図 1　タスクグラフ例

タスクの特性によって，実行するコアをうまく使い分けることで，性能効率の向上が期待できるからである．本研究では，同時に稼動するコアは 1 つのみとする．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．

前述の通り，本研究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との依存関係はないものとする．また，データの入力周期は各タスクで一定であるがタスク間では異なることもあるとするが，デッドラインは全てのタスクで同一であると仮定する．

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．ノードの大きさはタスクサイズ，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，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**動的である**，といい，動的なサブタスクを含むタスクを**動的変動タスク**という．これに対して，処理時間が入力データによらず一定であるサブタスクを**静的である**，という．動的なサブタスクは実行完了までに実行時間を知ることはできないとする．本研究では，動的なサブタスクの実行時間の変動確率は既知のパラメタとして扱う．動的なサブタスクは，**図 1** のように，タスクグラフでは分岐を用いて表現される．変動しうる実行時間の数だけ分岐数を設け，いずれかのパスが選択される，とすれば，静的なサブタスクのみで表すことができ，各分岐点での分岐確率は，実行時間の変動確率に一致する．本研究では，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ことに注意する．

2.2　消費エネルギの削減技術

2.2.1　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ことが知られている [6]．

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

第 1 項は，タスクを実行する際に消費されるダイナミックエネルギを表す．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 は比例定数を表す．

また，第 2 項は，スタティックエネルギを表す．スタティックエネルギとは，回路内にリーク電流が流れることにより消費されるエネルギであり，動作/非動作に関わらず，電源が入っているときには常に消費される．ただし，式中の T_2 はプロセッサが稼動する時間，V は電源電圧，I_{leak} はリーク電流を表す．

おおまかには，プロセッサの面積は性能の 2 乗に比例し [4]，スタティックエネルギは面積に比例することから，高性能なコアほどエネルギ効率は悪くなる．一方，マルチコアを用いると，面積は性能に比例するので，タスクを適切にスケジューリングできれば，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られる．

2.2.2　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の 1 つに **DPM**（Dynamic Power Management）が存在する [5]．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によって，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，本研究では通常の動作が可能な状態を**アクティブ状態**と呼び，電流が遮断されて，処理のできない状態を**省電力状態**と呼ぶ．状態の遷移には，オーバーヘッドエネルギが発生するため，期待できるスタティックエネルギの削減量と比較し，遷移を行うべきか判断する必要がある．

2.3　既存スケジューリング

タスクスケジューリングに関する先行研究のうち，本研究と関連の深い，デッドラインが十分長い場合のスケジューリング及び動的変動タスクを含むスケジューリングを紹介する．

2.3.1　デッドラインが十分長い場合のスケジューリング

デッドラインが長い場合のスケジューリングとして，タスクのまとめ実行が提案されている [1]．まとめ実行とは，デッドラインが長いことを利用して，タスクの実行開始を入力到着直後よりも遅らせ，後続の複数のタスクと連続して実行する，という手法である．まとめ実行を行うことで，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，オーバーヘッドエネルギの削減につながる．

表 1　入力変数

変数	意味
K	タスク数
M^k	タスク k の変動パターン数
$S^k(i)$	タスク k のパターン i のタスクサイズ
$p^k(i)$	タスク k のパターン i が実行される確率
I^k	タスク k の入力周期
d	デッドライン
$et^k(i,j)$	コア j でタスク k のパターン i を実行するのに要する時間
$E_d^k(i,j)$	コア j でタスク k のパターン i を実行するのに要するダイナミックエネルギ
$P_{Sa}(j)$	コア j のアクティブ状態のスタティック電力
$P_{Ss}(j)$	コア j の省電力状態のスタティック電力
$E_{ov}(j)$	コア j のアクティブ状態から省電力状態に移行するのに要するオーバーヘッドエネルギ

2.3.2　動的変動タスクのスケジューリング

動的変動タスクを扱うスケジューリングとして，DVFS（Dynamic Voltage and Frequency Scaling）を用いた手法が提案されている [3]．具体的には，後続のサブタスクの実行時間変動確率を考慮した上で，最悪パターンを実行することになってもデッドライン制約を満たし，かつ，消費エネルギの期待値が最小となるように動作周波数を設定し，サブタスクが完了する度に，デッドラインまでの余裕時間に応じて，動作周波数を変化させる，という手法である．この手法において，デッドラインは入力周期に一致しており，制御は 1 周期単位で行われるため，効果は限定的である．また，ダイナミックエネルギのみを削減する手法である．

3.　動的省電力スケジューリング

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，本研究が掲げる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．ここで，デッドラインは全てのタスクで同一である点に注意する．以降では混乱のない場合に限りタスク番号 k を省略する．

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，また各タスクパターンの ID を $i = 1, 2, \ldots, M$ 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，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 1 の例は，**図 2** のように変換でき，$I = 1$，$d = 10$，$M = 2$，$p(1) = 0.9$，$p(2) = 0.1$，$S(1) = 3$，$S(2) = 30$ となる．

タスクのパターン同様，コアにも性能が低いものから順に，ID を $j = 1, 2, \ldots$ とする．コア j でパターン i を実行したときの実行時間 $et(i,j)$，実行に要するダイナミックエネルギ $E_d(i,j)$，コア j のアクティブ状態時スタティック電力 $P_{Sa}(j)$，省電力状態時スタティック電力 $P_{Ss}(j)$，アクティブ状態から省電力状態に移行するのに要するオーバーヘッドエネルギ $E_{ov}(j)$ はすべて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．

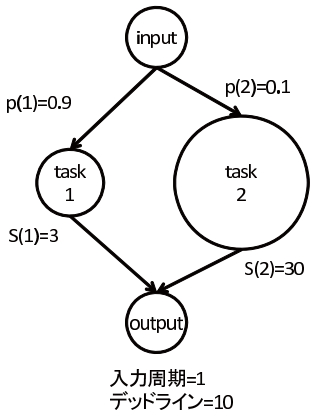

図 2　変動タスク例

表 2　各コアの実行時間

コア	サブタスク 1 の実行時間	サブタスク 2 の実行時間
コア 1	0.3	3.6
コア 2	0.15	1.8
コア 3	0.1	1.2
コア 4	0.075	0.9

確率的に実行時間が変動するタスクを扱うため，目的関数は，「1 タスクあたりのプロセッサの消費エネルギの期待値の最小化」，制約条件は，「すべてのタスクがデッドラインを満たす」となる．

3.2　単一タスクの動的省電力スケジューリング

まず単一タスクの動的スケジューリング [2] の概要を説明する．

3.2.1　概要

本研究が提案するスケジューリングでは，タスクのまとめ実行を採用する．タスクはまとめられるだけまとめて実行するものとし，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．

実行時に用いられるコアの候補は以下のように選択する．最悪のパターンを続けて実行することになっても，デッドラインは守らなければならないため，最悪実行時間が入力周期を下回るような性能を持つコアが 1 つ必要であり，これが最大性能のコアとなる．

図 2 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を**表 2** 中のコア候補を用いて処理するとき，コア 4 のみを用いれば，入力周期 1 に対し，最悪実行時間は 0.9 であるから，必ずタスクをデッドライン以内に終わらせることができるが，この例のように，小

タスクを実行する確率が高い場合には，オーバースペックとなることが考えられる．しかし，コア 4 以外のコアでは，最悪実行時間が入力周期よりも長いため，最悪パターンが続くとデッドラインを超えるおそれが発生する．したがって，必要に応じて実行中に稼動コアを変更させた方が消費エネルギを削減できると考えられる．

稼動コアの変更は，実行中におけるデッドラインまでの余裕時間に応じて行う．つまり，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．したがって，問題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問題に帰着する．このスケジューリングは，最適な稼動コア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．

3.2.2 定式化と求解

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．前節で述べた手法に従い，選択されたコア候補の総数を N とする．デッドラインまでの余裕時間に応じて実行中に稼動コアを変化させるが，実行中の余裕時間の計算は，1 タスクが終了する度に行うこととする．計算された余裕時間が変更点を超えていたとき，稼動コアの変更を行う．

ここで，稼動しているコアを一つの状態，1 タスクの実行を状態遷移として考えると，稼動コアの変更は，$N+1$ 状態の状態遷移図を用いて表すことができる．ここで，どのコアも稼動していない状態を s_0 とし，コア j が稼動している状態を，状態 s_j とする．

このように，稼動コアの変更を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，コア i 稼動時に 1 タスク実行後，コア j に変更する確率を表す．

また，遷移確率行列 P に対して，定常分布 π を定義することができる．

加えて，まとめ実行開始から終了する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．

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，以下のように表せる．

$$J = \pi E_D + \pi E_{Sa} + \| \ \pi \ (P * E_{OV}) \| + \frac{t_{sleep}}{t_{active} + t_{sleep}} E_{Ss} \qquad (2)$$

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_{Sa} は，ともに $(N+1)\times 1$ のベクトルであり，第 i 成分は，それぞれ，状態 s_i で 1 タスクを行ったとき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 $(N+1)\times(N+1)$ 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_{Ss} は $(N+1)\times 1$ のベクトルであり，第 i 成分は，状態 s_i において 1 入力周期で消費される省電力状態時スタティックエネルギを表す．

表 3 定式化にあたり導入したパラメタ

変数	意味
N	稼動コア候補数
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 タスクあたりの各状態の 平均ダイナミックエネルギ
E_{Sa}	1 タスクあたりの各状態の 平均アクティブ時スタティックエネルギ
E_{Ss}	1 入力周期あたりの各状態の 平均省電力状態時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

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，稼動コア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稼動させるコア，すなわち，状態 s_0 から遷移する状態を $s_\star$ とする．タスクはまとめられるだけまとめて実行するため，$x^0_{down} = d$，すなわち，まとめ実行の終了タイミングは余裕時間がデッドラインを超え，実行可能タスクがなくなったときである．

稼動コア変更点 $\{x^i_{up}\}$，$\{x^i_{down}\}$ 及び実行開始コア $s_\star$ 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，及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，稼動コア変更点及び実行開始コアの評価が可能となる．

具体的な稼動コア変更点及び実行開始コアの最適値は，遺伝的アルゴリズムを用いて求める．

4. マルチレートシステムの動的省電力スケジューリング

次にマルチレートタスクに対応するために前節のスケジューリングを拡張する．ここでは，「タスク展開方式」と「状態展開方式」の 2 つの方式について説明する．

4.1 タスク展開方式

ここでは，複数のタスクを単一タスクに変換することに

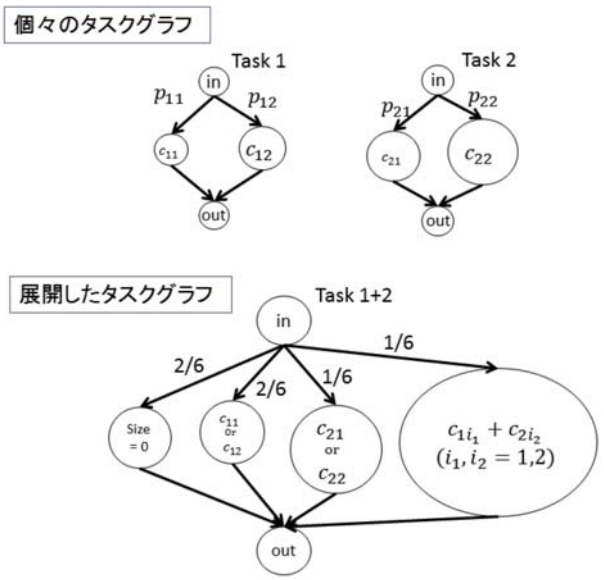

図 3　タスク展開の例

より，既存のスケジューリングアルゴリズムを変更することなく，マルチレートシステムに対応する方法について述べる．以降では，タスク 1 とタスク 2 の 2 つの入力周期の異なるタスクが存在するシステム ($K = 2$) を仮定し，**図 3** を用いて説明する．ここで，タスク 1 の入力周期 I^1 は 2，タスク 2 の入力周期 I^2 は 3 とした．

まず，各タスクの周期の最小公倍数を**ハイパーピリオド** (Hyper Period, 以降 HP と略す)，周期の最大公約数をマルチレートシステムの周期とする．HP 内は（HP/周期）個の区間に分けられる．

各タスクの入力周期の同期が取れている，すなわちハイパーピリオド内で最低 1 回は入力が同一のタイミングで到着すると仮定すると，各区間で 1 と 2，1 のみ，2 のみ，どちらも入力されない，の 4 パターンのどれかが入力となる．この時のそれぞれのタスクサイズは入力があるタスクの和となる．HP 内でそれぞれのパターンの発生回数と区間数の逆数を掛け合わせることにより，それぞれのパターンの発生確率を求めることができ，これにタスク内の発生確率を掛けることにより単一タスクとみなすことができる．ここでは 2 つのタスクを例に説明を行ったが，3 つ以上のタスクにも同様に適用可能である．

以上によりマルチレートシステムを単一タスクに変換することができ，3.2 節のスケジューリングが適用可能となる．

本方式ではタスクの大きさと出現確率のみで表現し，出現順序を考慮出来ていないため，最適なスケジューリングが得られない可能性がある．ただし，その場合であってもデッドライン制約が守られることは保証される．

4.2　状態展開方式

まず，本方式においても前節のタスク展開方式と同様に

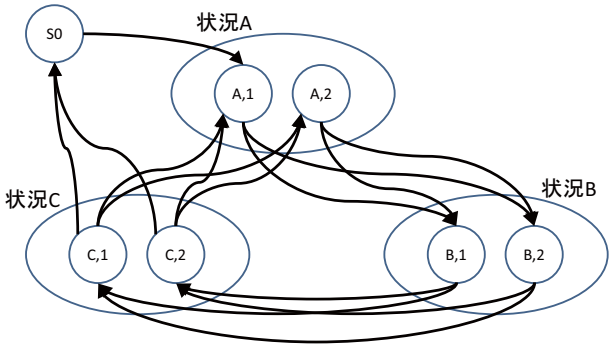

図 4　状態展開の例

HP と区間を考える．

状態展開方式は，**図 4** に示すように，これから実行する区間に応じて状態を区別し，それぞれに対して性能を変更する閾値を個別に設定する．各区間における状態の集合を**状況**と呼ぶ．この図は区間数が 3 の例であり，状況数も 3 となりそれぞれを状況 A, B, C としている．

ただし，現在は 3.2 節のスケジューリングからの変更を最小限とするため HP 途中における実行終了に対応していない．これは s_0 からは状況 A に属する状態のみに遷移し，s_0 への遷移は状況 C に属する状態からのみに制限していることに対応する．この制約により，複数のタスクの連続実行を仮想的な 1 つのタスクとみなし，既存のスケジューリング手法を利用する．この制約の下で正しいスケジューリングを保証するために，HP 中の全ての入力が揃うまで待機することとした．これは，待機する時間だけデッドライン制約が短くなることに相当するため，最適スケジューリングとは異なる結果になることが考えられる．

5.　評価

5.1　実験目的

前章で提案した手法によって得られるスケジューリングと，既存手法及び経験的に得られるスケジューリングの評価値を比較することで，本手法の消費エネルギ削減効果を評価する．

5.2　実験条件

評価に用いるタスクのパラメタは**表 4**，コアのパラメタは**表 5** にまとめた．タスクは人物検知センサの画像処理を想定し，あらかじめ与えられた背景との差分をとることにより，変化がなければその旨を送信し，変化があれば画像処理を行うことで，人物かどうか認識し，結果を送信する，というタスクとした．したがって，タスクパターンは軽い処理と重い処理の 2 パターンとし，タスクサイズの比は 1:10 とした．また，デッドラインは 10ms，入力周期は 0.5ms に設定し，2 種類のタスクの大きさの比は 3:7 とした．実行時間変動確率は，人物検知センサが設置される場所によって異なるため，人通りが少ない ($p(1) = 0.1$,

Vol.2015-SLDM-170 No.5
Vol.2015-EMB-36 No.5
2015/3/6

表 4　タスクに関するパラメタ

タスクに関するパラメタ	値
タスク数 K	2
タスクパターン数 M^1, M^2	2,2
タスクサイズ $S^k(i)$	$S^1(1)=30, S^2(1)=70,$ $S^1(2)=3, S^2(2)=7$
入力周期 I^1, I^2	0.5 ms, 0.5 ms
デッドライン d	10 ms

表 5　コアに関するパラメタ

	MCU1	MCU2	MCU3	MCU4
コア ID j	1	2	3	4
ダイナミック電力 $P_d(j)$[W]	0.0138	0.031	0.070	0.156
アクティブ時スタティック電力 $P_{Sa}(j)$[W]	1.62e-3	5.84e-3	2.09e-2	7.50e-2
省電力状態時スタティック電力 $P_{Ss}(j)$[W]	6.90e-7	2.07e-6	6.20e-6	1.86e-5
オーバーヘッドエネルギ $E_{ov}(j)$[J]	2.24e-7	1.75e-6	1.37e-5	1.07e-4

$p(2)=0.9$) から人通りが多い ($p(1)=0.9$，$p(2)=0.1$) までの 5 つの場合を想定し，それぞれ結果を求めた．プロセッサ候補数は 4 とし，本論文では一例として，文献 [1] で利用されているコアのうち最小性能のコアを MCU1，最大性能のコアを MCU4 とした．MCU4 の MCU1 に対する性能比がほぼ 4 であることから，MCU1 に対する性能比が 2 及び 3 となるような仮想的なコア (MCU2, MCU3) を設定し，MCU2，MCU3 の各パラメタは MCU1 と MCU4 の値を用いて補間した．

遺伝的アルゴリズムにおけるパラメタは，集団数を 10000 とし，世代数を 1000 とした．また，評価値の上位 1 割をエリートとして保持し，集団の 1%に突然変異が発生する，と設定した．

5.3　実験結果

図 5 は，$p(1)$ の値を変化させたときの，各方式における消費エネルギの内訳の変化を表したグラフである．内訳については，各コアで消費されるダイナミックエネルギとアクティブ時スタティックエネルギの和を各コア名で表し，オーバーヘッドは総オーバーヘッドエネルギ，スタティックは省電力状態時スタティックエネルギの総和を表している．評価関数の定義が，1 入力周期あたりの消費エネルギの期待値であることから，単位が [J/周期] となることに注意する．

この結果から，タスクのパターンによって各方式の優劣が変化していることがわかる．これはタスク拡張ではタスクの出現順序が無視されること，状態拡張ではデッドライン制約が本来のものよりも短くなっていること，によりどちらも最適解が実現出来ていないためと考えられる．

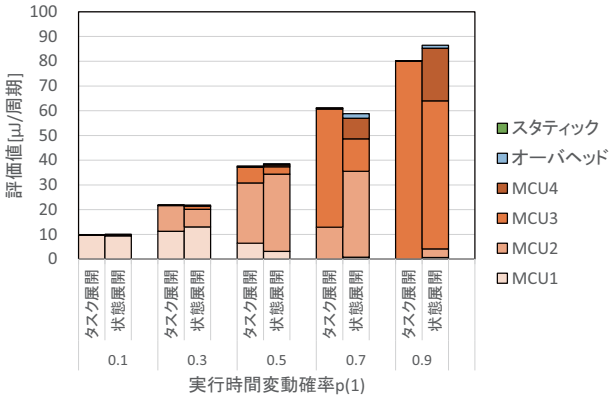

図 5　各手法の評価値とエネルギ内訳

6.　結論

本研究では，確率的に実行時間が動的変動する複数のタスクを周期的に実行するマルチレート組込みシステムにおけるプロセッサの消費エネルギを削減するという問題に対し，既存スケジューリングをタスク展開と状態展開の 2 つの方式で拡張することによる動的省電力スケジューリングを提案した．

評価の結果どちらの手法においても動的スケジューリングは実現可能であることがわかり，どちらの手法においても近似解が得られていることがわかった．

今後はさらなる改良により，マルチレート組込みシステムの最適なスケジューリングを目指す予定である．

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] 岡本和也. “エネルギモデルを用いたマルチコア組み込みシステムの設計支援”, 東京大学大学院情報理工学系研究科修士論文, 2013.

[2] 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.

[3] Jason Cong and Karthik Gururaj. “Energy Efficient Multiprocessor Task Scheduling under Input-dependent Variation” Proceedings of the Conference on Design, Automation and Test in Europe, pp. 411–416 2009.

[4] Fred Pollack, Intel Corp. “Micro32 conference keynote” 1999.

[5] Luca Benini, Alessandro Bogliolo, and Giovanni De Micheli. “A survey of design techniques for system level dynamic power management”. IEEE Trans. on Very Large Scale Integration Systems, Vol. 8, No. 3, 2000.

[6] Thomas.D. Burd and Robert.W. Brodersen. “Energy efficient cmos microprocessor design”. Hawaii International Conference on System Sciences, p. 288, 1995.