同一命令セットヘテロジニアスマルチコア向け消費電力管理手法の評価環境

青野 和巳[1,a)]　高瀬 英希[1]　松原 豊[2]　高木 一義[1]　高木 直史[1]

概要：これまでに様々な組込みシステムに対する消費電力管理手法が提案されているが，対象とするプロセッサアーキテクチャによって消費電力の削減効果は異なるため，最も高い削減効果が得られる手法をシステム設計者が探索する必要がある．本研究では，同一命令セットヘテロジニアスマルチコアを採用した組込みリアルタイムシステムにおいて，消費電力管理手法を実機上で評価する手法を提案する．同一命令セットヘテロジニアスマルチコアとは，高性能コアと高電力効率コアを排他的に動作させるプロセッサアーキテクチャであり，性能向上と消費電力削減の両立を実現するアーキテクチャとして注目されている．本アーキテクチャを搭載した ODROID-XU3 ボードにおいて性質の異なるタスクセットを対象に複数の消費電力管理手法を適用し，リアルタイム性制約下で消費電力を評価することで，提案手法の有用性を示す．

キーワード：組込みシステム，消費電力管理，ヘテロジニアスマルチコア，タスクスケジューリング

An Evaluation Environment of Power Management for Single-ISA Heterogeneous Multi-core Systems

Aono Kazumi[1,a)]　Takase Hideki[1]　Matsubara Yutaka[2]　Takagi Kazuyoshi[1]　Takagi Naofumi[1]

Abstract: Because energy reduction effects of power management methods proposed until now depend on target architectures, system designers have to search the best method which derive the maximum energy reduction effect. In this work, we propose an evaluation environment of power management for single-ISA heterogeneous multi-core embedded real systems. Single-ISA heterogeneous multi-core is a processor architecture which operates high performance cores and power efficient cores exclusively. This architecture has been attracted attention as an architecture which derive an improvement performance and an energy reduction. ODROID-XU3 is a computing device which has this architecture. We show usability of the proposed method by applying some power management methods and some tasksets to ODROID-XU3, and by evaluating these power consumption under the real-time constraint.

Keywords: Embedded System, Power Management, Heterogeneous Multi-core, Task Scheduling

1. はじめに

近年の組込みシステムでは，より高い性能が求められるとともに，消費電力を最小化することが重要な課題となっている．組込みシステムの高性能化と低消費電力化の両立を実現するための解決策のひとつとして，同一命令セットヘテロジニアスマルチコアアーキテクチャが注目されている．本アーキテクチャは，同一の命令セットで，異なる性能を持つ複数のコアで構成される．ゆえに，命令セットの違いを意識せずにタスクの動作コアを切り替えることができる．負荷の大きい時は高性能コアを，そうでない時は高電力効率コアを動作させることで，性能を落とすこと無く消費電力の最小化を実現する [1]．ただし，組込みシステムにおいては，求められるリアルタイム性を保証することも考慮しなければならない．ここで，リアルタイム性とは，タスクの実行をそれぞれの所定の時刻であるデッドラインまでに完了しなければならないことを指す．つまり，同一命令セットヘテロジニアスマルチコアを採用した組込みリアルタイムシステムでは，リアルタイム性制約を保証した上での適切なタスクスケジューリングによって，消費電力を最小化することが望まれる．

[1] 京都大学 大学院情報学研究科
Graduate Shcool of Informatics, Kyoto University
[2] 名古屋大学 大学院情報科学研究科
Graduate Shcool of Information Science, Nagoya University
[a)] aono@lab3.kuis.kyoto-u.ac.jp

組込みシステムを対象としたこれまでの研究によって，動的電力管理（Dynamic Power Management，DPM）および動的電圧・周波数制御（Dynamic Voltage and Frequency Scaling，DVFS）といった消費電力管理手法が提案されている．しかし，既存研究における提案手法の多くは，シミュレータ上での有効性の評価に留まっており，実機によるものであっても，それぞれ異なる環境下で評価が示されている．また，ここ数年で注目されてきたアーキテクチャである同一命令セットヘテロジニアスマルチコアの環境への対応は議論されていない．

実システムにおいてアプリケーションがタスクセットとして与えられた時，最も高い電力削減効果が得られるものを，システム設計者が探索する必要がある．しかし，実システムで複数の消費電力管理手法を評価できる環境は，著者らの知る限りこれまで提案されていない．このため，アーキテクチャとタスクセットに対応した最適な消費電力管理手法を探索するための負担が大きくなる．そこで本研究は，この探索を容易にすることができるため，システム設計者にとって有用となる．

本稿では，1 つずつの高性能コアと高電力効率コアで構成される同一命令セットヘテロジニアスマルチコアを採用した組込みリアルタイムシステムにおいて，消費電力管理手法を実システム上で評価する手法を提案する．提案手法は Linux 上で動作し，与えられたタスクセットに対して複数の管理手法を適用した際の消費電力削減効果を評価する．システム設計者は，最も高い削減効果が得られる消費電力管理手法を容易に探索することが可能となる．さらに，本研究では，ODROID-XU3 ボードにおいてリアルタイム性制約下で消費電力が評価できることを示した．

本研究による貢献は以下の通りである．

(1) 実システム上で，消費電力管理手法を評価するための評価環境を開発した．これにより，対象とするアーキテクチャによって消費電力削減効果の異なる消費電力管理手法を，タスクセットを指定した上で比較評価することが可能となった．
(2) 既存の電力管理手法を同一命令セットヘテロジニアスマルチコアの実システムに対応できるよう検討を行った．これにより，実システムの環境に即して消費電力管理手法を評価することが可能となった．
(3) Linux の動作する ODROID-XU3 に対して，提案する評価環境を適用した．それぞれのタスクセットの特性に対応して，消費電力管理手法の削減効果に違いと傾向があることを議論できた．これにより，提案する評価環境の有用性が示された．

本稿の構成は，以下の通りである．2 節にて，本研究で対象とする同一命令セットヘテロジニアスマルチコアについて説明する．3 節にて，これまでに提案されてきた組込みシステムのための消費電力管理手法について紹介する．4 節にて，提案手法である同一命令セットヘテロジニアスマルチコアのための消費電力管理手法の評価環境について詳説する．5 節にて，ODROID-XU3 への適用事例による提案手法の有用性を示す．最後に，6 節にて本稿のまとめと今後の展望を述べる．

2. 同一命令セットヘテロジニアスマルチコア

同一命令セットヘテロジニアスマルチコアとは，同じ命令セットを持ち，異なる性能と電力効率で設計された複数のコアを持つプロセッサアーキテクチャである．このため，命令セットの差異を意識することなく，タスクの動作するコアを変更できる．同一命令セットヘテロジニアスマルチコアでは，高性能なコアと高電力効率なコアを対応づけたコアペアとして，動作するコアを排他的に切り替えてタスクが実行される．負荷の大きい時は高性能コアを，そうでない時は高電力効率コアを動作させることで，ピーク時の性能を落とすこと無く消費電力の最小化を実現する [1]．今後，同一命令セットヘテロジニアスマルチコアを採用した組込みシステムの普及が見込まれ，高性能化と低消費電力化の両立に貢献することが期待される．

NVIDIA 社は，2011 年に Variable SMP [2] と呼ぶ技術を発表した．複数の高性能コアとコンパニオンコアと呼ばれる高電力効率コアを排他的に動作させることで，高い性能が必要な場面と低い性能で十分な場面との両方で，高い電力効率を達成する．同一命令セットヘテロジニアスマルチコアの代表例としては，ARM 社の提唱する big.LITTLE アーキテクチャ [3] が挙げられる．省電力な Cortex-A7（または A53）と高性能な A15（または A57）の両方を同一チップ上に搭載し，これらを排他動作させることで高性能と低消費電力の両方を達成する [4]．Samsung 社は，2013 年にスマートフォン向けのチップセットに big.LITTLE アーキテクチャを採用したことを発表し，Exynos 5 Octa という製品群で商用化している [5]．サーバ向けプロセッサにおいては，KnightShift と呼ばれる同様の手法が提案されている [6]．計算負荷が軽い時には Knight と呼ばれる省電力かつ低性能なコアが稼働し，それ以外の負荷の重い処理は高性能なコアが担当する．文献 [7] では，オンチップメモリを共有した高性能コアと高電力効率コアで構成される同一命令セットヘテロジニアスマルチコアの回路設計および実測による評価を示している．

3. 消費電力管理手法

3.1 動的電力管理

DPM とは，システムの運用中に未使用となるコアの動作を停止させる技術である [8]．多くの組込み向けプロセッサでは，通常の実行時であるアクティブモード，クロックの供給を停止して動的電力を削減するアイドルモード，および，電源の供給を停止して動的および静的電力を削減す

るスリープモードなどの，複数の電力モードを備えている．DPM を実現するハードウェア技術の代表的なものとしては，クロックゲーティング [9] がある．DPM では，タスクスケジューリング中に実行すべきタスクが存在せずコアが未稼働となった際に，コアの電力モードをこまめに遷移させることで消費電力の削減を狙う．

3.2 動的電圧・周波数制御

DVFS とは，コアの供給電圧および動作周波数を適切に制御する技術である [8]．タスクの実行時間は入力データや通過パスに依存して最悪実行時間より早く終わることが多いため，スラック時間と呼ばれるタスクの実行終了からデッドラインまでの余裕時間が生じる．CMOS 回路の消費電力は，周波数の 2 乗に比例すると近似できる．つまり，タスクスケジューリングにおいてスラック時間が生じた際には，DVFS によってコアの供給電圧と周波数を適切に下げてタスクを実行することで，消費電力の削減が図られる．

DVFS 手法は，静的なものと動的なものに分類される．前者は，システム設計時にタスクを実行する際の電圧および周波数を決定する手法である [10]．各タスクのリリース時刻と実行時間が既知の場合に有効であり，固定優先度ベースのリアルタイムスケジューリングで用いられることが多い．文献 [11] によれば，各タスクの実行時間が既知かつ相対デッドラインに一致するとき，全タスクの周波数が平準化されるように DVFS を適用できる場合に消費電力が最小となることが示されている．ただし，静的な DVFS では，タスクの実行時間の変動により生じるスラック時間を有効に活用できないという問題がある．一方，動的な DVFS 手法では，システムの稼働中にスラック時間が生じた際に，各タスクの電圧および周波数を制御する．文献 [12] では，周期タスクのみからなるタスクセットに対する動的な DVFS 手法の比較評価を示している．固定優先度ベースおよび動的優先度ベースのスケジューリングの双方で幾つかの DVFS 手法を比較しているが，シミュレータ上での議論に留まっている．

DVFS と DPM は同時に適用することもできる．文献 [13] では，DVFS に加えて DPM を組み合わせた手法を提案している．タスクスケジューリング中にスラック時間が生じた際には DVFS を適用し，さらに実行すべきタスクが存在せずコアが未稼働となった際には DPM を適用する．

3.3 CPU マイグレーション

同一命令セットヘテロジニアスマルチコアでは，高性能コアおよび高電力効率コアを対応付けてコアペアとして動作コアを切り替えることによって，CPU マイグレーションが適用できる．コアペアに割り付けられたタスクセットは，その状況に応じて，コアペア内の高性能コアと高電力効率コアのいずれかで実行される．**図 1** は，高性能コアから高電力効率コアへの CPU マイグレーションの例を示している．

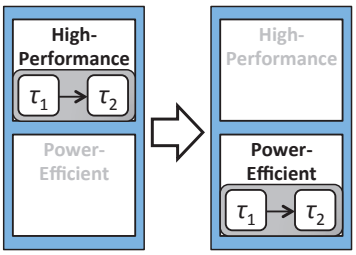

図 1 CPU マイグレーション
Fig. 1 CPU migration.

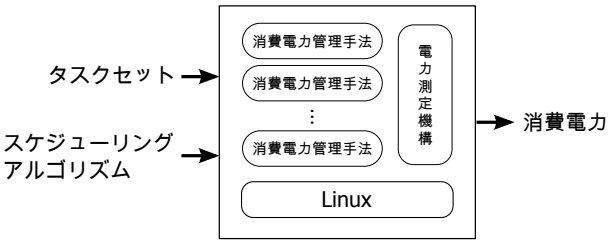

図 2 評価環境
Fig. 2 Evaluation Environment.

文献 [14] では，コアペアテーブルと呼ぶ動作コア，動作周波数，および，タスクの実行時間の情報を用いることで，DVFS 手法の枠組みでコアペア内の動作コアの切替えを実現する手法を提案している．CPU マイグレーションによって動作コアを排他的に切り替えることで，同一命令セットヘテロジニアスマルチコアにおいてピーク時の性能を保証しつつ消費電力の削減を実現することができる．ただし，組込みリアルタイムシステムにおいては，CPU マイグレーションはリアルタイム性が保証されるように行う必要がある．

4. 評価環境

4.1 評価環境の適用対象

本稿で提案する評価環境を**図 2** に示す．評価環境は，追加可能な複数の消費電力管理手法を持つ．評価環境にタスクセットとスケジューリングアルゴリズムを入力し，消費電力管理手法を選択することで，タスクセットの実行に掛かる消費電力を得る．スケジューリングアルゴリズムとしては，周期タスクのリアルタイムスケジューリングアルゴリズムとして代表的なものである Rate-Monotonic Scheduling (RMS) および Earliest Deadline First Scheduling (EDFS) に対応する．ただし，スケジューリングアルゴリズムとの組合せによって適用できない消費電力管理手法が存在する．

評価環境は，以下の条件を満たすシステムに対して適用可能である．

- Linux が動作する
- 同一命令セットヘテロジニアスマルチコア

- CPU の電力測定の機構をもつ

評価環境は，Linux のアプリケーションとして動作する．評価環境の対象とするシステムでは，アーキテクチャは 1 つずつの高性能コア $Core_{HP}$ と高電力効率コア $Core_{PE}$ で構成されるものとする．さらに各コアは M_{HP} および M_{PE} 種類の動作周波数 $f_{m_{HP}}$ および $f_{m_{PE}}$ に設定可能なものとし，特に，最高周波数を $f_{max_{HP}}$ および $f_{max_{PE}}$，最低周波数を $f_{min_{HP}}$ および $f_{min_{PE}}$ とする．

CPU の電力測定の機構としては，コアやメモリ毎に一定周期で電力値を取得できることとする．取得方法としては，オンチップに搭載された電流・電圧センサ値の取得やオシロスコープによる測定が考えられる．電力測定の機構は，スケジューラおよびタスクセットの実行に影響を及ぼさないように実行できることが望ましい．

以上の条件を備えたシステムであれば，異なるシステムに移植して評価環境を動作させることが可能である．ただし，電力測定手法についてはシステムによって仕様が異なるため，数行の変更を必要とする．

4.2 スケジューラ

評価環境におけるスケジューラのアルゴリズムを **Algorithm 1** に示す．タスクセット，スケジューリングアルゴリズムおよび消費電力管理手法を選択することで，タスクスケジューリングを行う．本スケジューラは，Linux のアプリケーションとして実現されるため，移植性が高い．

2 から 10 行目では，タスクのリリース判定を行う．5 行目に示すとおり，タスクをリリースするときにそのタスクが実行完了していなかった場合にはデッドラインミスと判定し，スケジューラを終了する．11 行目では，スケジューリングアルゴリズムによって次に実行すべき最高優先度のタスク $task_{max-Priority}$ を決定する．EDFS を採用する場合は，現在時刻から最もデッドラインが近いタスクを $task_{max-Priority}$ とする．12 から 25 行目では，$task_{max-Priority}$ のディスパッチを行う．13 および 14 行目での動作コアおよび動作周波数の選択は，評価対象とする消費電力管理手法に基づいて決定する．17 行目は，必要に応じてマイグレーションや DVFS を行う．20 および 21 行目は，$task_{max-Priority}$ 以外のタスクが実行中の場合にプリエンプションを実行して $task_{max-Priority}$ をディスパッチする．24 行目は，消費電力管理手法において DPM が採用されている場合に，実行すべきタスクが存在しない時にコアのオフライン化を行う．26 から 28 行目ではスケジューラの終了判定を行う．5 行目でデッドラインミスすることなくスケジューリング終了時刻に到達した場合，与えられたタスクセットは選択したスケジューリングアルゴリズムおよび消費電力管理手法によってスケジューリング可能であると判定される．

Algorithm 1 スケジューラ

```
1: loop
2:     for i ← 0, I − 1 do                ▷ リリースすべきタスクの確認
3:         if deadline_i ≤ time then
4:             if task_i is released then
5:                 break                  ▷ 終了．デッドラインミス発生
6:             end if
7:         else
8:             Release task_i
9:         end if
10:    end for
11:    Find task_{max−Priority}           ▷ 最高優先度タスクを探す
12:    if task_{max−Priority} exist then
13:        Select Core                    ▷ 消費電力管理手法に基いて決定
14:        Select f                       ▷ 消費電力管理手法に基いて決定
15:        if task_running = task_{max−Priority} then
16:            if task_running is not finished then
17:                Execute task_{max−Priority} on Core by f
18:            end if
19:        else
20:            Stop task_running          ▷ プリエンプション発生
21:            Execute task_{max−Priority} on Core by f
22:        end if
23:    else
24:        Offlining unused core          ▷ 消費電力管理手法に基いて決定
25:    end if
26:    if time ≥ hyperperiod exist then
27:        break                          ▷ スケジューリング終了
28:    end if
29: end loop
```

4.3 タスクセット

提案する評価環境では，組込みシステム向けのベンチマークスイートである MiBench [15] を用いて作成した複数のタスクセットを用意している．**表 1** に，標準で用意したタスクセットを示す．これらの標準タスクセットを用いることで，対象システムの消費電力削減効果の傾向を議論することが容易となる．つまり，設計者が採用するアーキテクチャから決定したい場合には，複数の対象システムに対して標準タスクセットを用いて管理手法の評価を行うことで，適切なシステムを選定することができる．対象システムと特定のアプリケーションが決定している場合には，それを用いて評価環境を実行することで，対象システムに最も適した消費電力管理手法を選定することができる．

タスクセットの作成手順について述べる．まず，タスクの選択をする，そして，各タスクの最悪実行時間 $wcet(Core, f, i)$ を計測して，各タスクの負荷 u_i を決定する．$wcet(Core, f, i)$ は，動作コア，動作周波数および タスクによって異なるため，あらかじめ各コア，動作周波数およびタスクの全ての組み合わせに対して実行時間を計測し，テーブルとして参照できるようタスクセットに保持する．実行周期 τ_i は，各タスクの負荷 u_i を任意に決定することによって，次式で求められる．

$$\tau_i = \frac{wcet(Core_{HP}, f_{max_{HP}}, i)}{u_i} \quad (1)$$

表 1 タスクセット
Table 1 Tasksets.

	特徴	タスク				
taskset1	実行時間大	patricia	gsm	basicmath	adpcm	blowfish
taskset2	実行時間小	stringsearch	sha	jpeg	dijkstra	susan
taskset3	ばらつき大	blowfish	patricia	typeset	basicmath	adpcm
taskset4	ばらつき小	sha	susan	bitcount	stringsearch	jpeg
taskset5	実行時間大中小	stringsearch	sha	bitcount	adpcm	blowfish

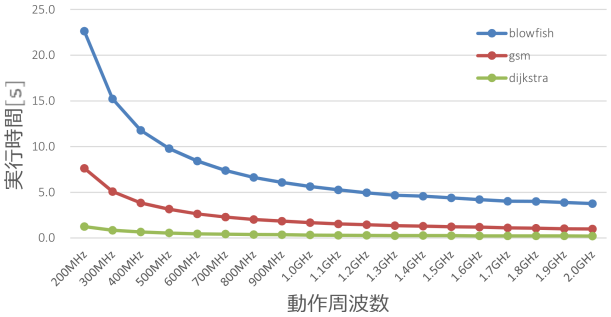

図 3 動作周波数と実行時間の関係
Fig. 3 Operating Frequency and Execution Time.

タスクセットの負荷 U は，次式で求められる．

$$U = \sum_{i=0}^{I-1} u_i \tag{2}$$

EDFS の場合は，タスクセットの負荷 U が 1 以下であれば，タスク数 I および 負荷 u_i に関係なく，理論上実行可能なタスクセットである．

4.4 消費電力管理手法の実システムへの対応方法

これまでに提案されている消費電力管理手法 [8], [10], [12], [13], [14] では，動作周波数と実行時間が線形であると仮定している．ここで，ODROID-XU3 にて MiBench の blowfish，gsm および dijkstra を実行した際の，動作周波数と実行時間の関係を，**図 3** に示す．図 3 より，実システム上では動作周波数と実行時間が非線形であることが分かる．リアルタイム性を保証する必要がある消費電力管理手法においては，スケジューリング中においてタスクの残り実行時間を正確に見積もることが重要となる．残り最悪実行時間とは，最高周波数でタスクを実行した際に要する残り時間のことであり，特に DVFS で重要となるスラック時間はデッドラインから残り最悪実行時間を引くことで求められる．さらに，タスクによって動作周波数と実行時間の関係が異なるということが分かる．本項では，動作周波数と実行時間の非線形性に適用可能にする方法を説明する．

まず，あらかじめ各コアと動作周波数の全組み合わせにおける実行時間を全タスクについて計測し，コアペアテーブルを作成する．残り最悪実行時間の初期値には，最高性能コア $Core_{HP}$ にて最高周波数 $f_{max_{HP}}$ でタスク i を実行した際の最悪実行時間 $wcet(Core_{HP}, f_{max_{HP}}, i)$ を用いる．コアペアテーブルに記録されている $wcet(Core_{HP}, f_{max_{HP}}, i)$ を元に，全タスク中で最も早いデッドライン時刻 d_0 までに実行しなければならない残り最悪実行時間 c_{rem} を計算する．ここで，x_i を，タスク i を時刻 d_0 までに実行しなければならない最悪実行時間とすると．x_i は，d_0 以降に $Core_{HP}$ にて $f_{max_{HP}}$ でタスク i を実行すると仮定しても，実行を完了することができない残り最悪実行時間として計算することができる．x_i を用いて c_{rem} は，以下の式で求められる．

$$c_{rem} = \sum_{i=0}^{I-1} \left(x_i \times \frac{wcet(Core, f, i)}{wcet(Core_{HP}, f_{max_{HP}}, i)} \right) \tag{3}$$

ここで，現在時刻を t とすると，次式

$$c_{rem} < d_0 - t \tag{4}$$

を満たす最小の $Core$ および f の組合せを DVFS 手法によって決定する．

実システムにおいて消費電力管理手法を適用する際には，実行時のオーバヘッドも考慮しなければならない．ここで，オーバヘッドとしては，消費電力管理手法を含むスケジューラの処理，および，消費電力の測定がある．これらのオーバヘッドは，実システム上で実測することによって加味する．具体的には，スケジューラの処理として，消費電力管理手法の適用，タスクのディスパッチやプリエンプションによる中断については，実システム上でそれぞれの実行時間を測定する．消費電力の測定については，スケジューラおよびタスクセットの実行を妨げないように実現できる場合には無視できる．

5. 適用事例

5.1 適用対象

本研究の提案手法である評価環境を，big.LITTLE アーキテクチャを採用した Exynos 5422 [16] を搭載する ODROID-XU3 [17] に適用し，複数の消費電力管理手法を評価した．ODROID-XU3 では，Ubuntu 14.04 カーネルが動作する．ODROID-XU3 は，高電力効率コアとして 4 個の Cortex-A7 コアからなる LITTLE クラスタ，および，高性能コアとして 4 個の Cortex-A15 コアからなる big クラスタを持つ．LITTLE クラスタの各コアを cpu0-3，big クラスタの各コアを cpu4-7 とそれぞれ呼ぶこととする．

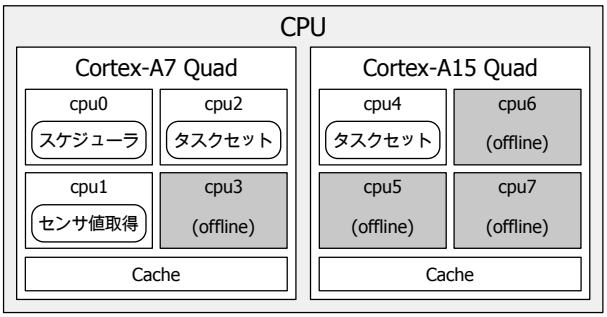

図 4 スケジューラ，電力センサおよびタスクセットの配置図
Fig. 4 Allocations of a scheduler, a power sensor and Tasksets.

DVFS はクラスタ単位で可能であるが，コア単位で設定することはできなかった．LITTLE クラスタの動作周波数は，200 MHz から 100 MHz 毎に 1.4 GHz まで設定可能である．big クラスタの動作周波数は，200 MHz から 100 MHz 毎に 2.0 GHz まで設定可能である．各クラスタの動作周波数と電圧は，実際に測定したところ比例関係にはならなかった．DPM は，cpu0 以外のコアについてコア単位で可能である．

電力センサは LITTLE クラスタ，big クラスタ，メモリおよび GPU に INA231 [18] が搭載されており，これらの電力センサを使用した．サンプリング周期は Linux カーネルの設定によって 264 ミリ秒に固定されており，今回はこの値を用いた．センサ取得スクリプトは，C によって作成した．センサの値を取得する周期は，サンプリング周期の約半分の 100 ミリ秒とした．電力センサの実行には，スケジューラの実行およびタスクセットの実行へ影響を及ぼさないよう，cpu1 を使用した．

5.2 評価環境の適用

スケジューラ，電力センサ取得スクリプトおよびタスクセットの配置図を**図 4** に示す．スケージューラおよび電力センサ取得スクリプトは，それぞれ cpu0 および cpu1 で実行する．タスクセットは，消費電力管理手法の方針によって cpu2 および cpu4 のいずれかのコアを選択する．cpu2 を使用する際には cpu4 をオフライン化し，cpu4 を使用する際には cpu2 をオフライン化する．cpu3,5,6,7 は今回は使用しないため常にオフライン化する．

5.2.1 評価対象の消費電力管理手法

評価対象とする消費電力管理手法を **表 2** に示す．スケジューリングアルゴリズムは EDFS を採用した．

DPM.big の消費電力管理手法の方針では，使用するコアは常に big クラスタに属する cpu4 で，動作周波数は常に最高周波数の 2.0 GHz でタスクセットを実行する．実行するタスクが存在しない場合には，cpu4 をオフライン化してスリープモードに遷移させる．

DPM.LITTLE の消費電力管理手法の方針では，使用するコアは常に LITTLE クラスタに属する cpu2 で，動作周波数は常に最高周波数の 1.4 GHz でタスクセットを実行する．実行するタスクが存在しない場合には，cpu2 をオフライン化する．

S-DVFS.big の消費電力管理手法の方針では，使用するコアは常に big クラスタに属する cpu4 で，動作周波数はデッドラインミスしない最低周波数でタスクセットを実行する．DVFS 手法は静的なものを採用する．動作周波数の決定には，まずは最低周波数で実行し，デッドラインミスが発生した場合は動作周波数を増加させてタスクセットを再実行していき，デッドラインミスしない最低周波数を決定する．

S-DVFS.LITTLE の消費電力管理手法の方針では，使用するコアは常に LITTLE クラスタに属する cpu2 で，動作周波数はデッドラインミスしない最低周波数でタスクセットを実行する．S-DVFS.big の場合と同様にしてデッドラインミスしない最低周波数を決定する．

CPUmig の消費電力管理手法の方針では，使用するコアは最初は LITTLE クラスタに属する cpu2 で実行し，cpu 使用率を監視することによって 動的に使用コアを変更する．今回は，LITTLE で実行中に cpu 使用率が 80%を超えた場合に big に切り替え，big で実行中に cpu 使用率が 50%を下回った場合に LITTLE に切り替えるものとした．動作周波数は常に各クラスタにおける最高周波数でタスクセットを実行する．

D-DVFS の消費電力管理手法の方針では，使用するコアおよび動作周波数を，laEDF アルゴリズム [19] に基いて動的に変更する．laEDF は動的な DVFS 手法であり，ある時刻において最高優先度タスクに後続するタスクが最高の周波数で実行されると仮定して，デッドライン制約を保証する範囲で最低となる周波数を算出する．スケジューリングループ，タスクの停止およびマイグレーションに要するオーバヘッドは，それぞれ 0.25 秒，0.5 秒，0.5 秒として計測した．これらの値は実際にスケジューラを実行する中で決定した．

S-DVFS.DPM.big の消費電力管理手法の方針では，S-DVFS.big の方針に加えて，実行するタスクが存在しない場合に cpu4 をオフライン化する．DPM に要するオーバヘッドを計測したところ，タスクの実行時間と比べて非常に小さかったため，今回は無視した．

S-DVFS.DPM.LITTLE の消費電力管理手法の方針では，S-DVFS.LITTLE の方針に加えて，実行するタスクが存在しない場合に cpu2 をオフライン化する．ここでも，DPM に要するオーバヘッドは無視した．

5.2.2 タスクセット

タスクセットは，表 1 に示した評価環境の標準タスクセットである MiBench を用いた 5 種類のものを使用し，負荷についてそれぞれ 5%，10%，20%，40%，60%および 80%に設定したもの，合計 30 種類のタスクセットを用意

表 2 評価対象の消費電力管理手法
Table 2 Evaluation target of power management methods.

スケジューラ名	使用コア	使用コア数	動作周波数	DVFS	マイグレーション	DPM
DPM.big	big	1	2.0GHz	×	×	○
DPM.LITTLE	LITTLE	1	1.4GHz	×	×	○
S-DVFS.big	big	1	デッドラインミスしない最低周波数	×	×	×
S-DVFS.LITTLE	LITTLE	1	デッドラインミスしない最低周波数	×	×	×
CPUmig	両方	1	2.0GHz/1.4GHz	×	○	○
D-DVFS	両方	1	-	○ (laEDF)	○	×
S-DVFS.DPM.big	big	1	デッドラインミスしない最低周波数	×	×	○
S-DVFS.DPM.LITTLE	LITTLE	1	デッドラインミスしない最低周波数	×	×	○

した．ハイパーピリオドは，全タスクセット 120 秒に統一した．

5.3 結果

スケジューラとタスクセットの全組み合わせについて提案手法を適用し，電力測定を行った．**表 3** に LITTLE クラスタの電力センサの平均値，big クラスタの電力センサの平均値およびメモリの電力センサの平均値の和を示す．各タスクセットについて，最も消費電力の小さいものを下線で表している．消費電力管理手法および負荷によっては，デッドラインミスが発生してタスクスケジューリングを完了できないものもあった．S-DVFS における動作周波数を，**表 4** に示す．

5.4 考察

ODROID-XU3 における消費電力管理手法と適用したタスクセットについて，測定結果より以下のことがいえる．負荷の低い場合（20%以下）は D-DVFS が，それ以外の場合（40-80%）では S-DVFS.DPM.LITTLE が効果があり，S-DVFS.LITTLE でデッドラインを満たせない場合（taskset1,3 の負荷 80%）には，S-DVFS.big が効果がある．また，DPM.big, CPUmig が効果があるものは存在しなかった．そして，DVFS を適用した上でさらに DPM をやるのはあまり意味が無いということが分かった．

このように，対象とする同一命令セットヘテロジニアスマルチコアシステムに対して，複数の消費電力管理手法を評価環境に適用することで，タスクセットに応じてどの消費電力管理手法が適しているのかを議論することができる．これにより，提案する消費電力管理手法の評価環境にある有用性が示された．

6. おわりに

本研究では，同一命令セットヘテロジニアスマルチコアを採用した組込みリアルタイムシステムにおいて，消費電力管理手法を実機上で評価する手法を提案した．また，既存の消費電力管理手法を対象システムに適用できるよう変更した．同一命令セットヘテロジニアスマルチコアアーキテクチャを搭載した ODROID-XU3 ボードにおいて性質の異なるタスクセットを対象に複数の消費電力管理手法を適用し，リアルタイム性制約下で消費電力を評価することで，提案手法の有用性を示した．

今回の実装ではタスクの実行にシステム関数を使用したため，オーバヘッドが大きくなってしまっている．そこで今後の方針としては，スケジューラのオーバヘッドを小さくできるよう検討する．さらに，他のアーキテクチャに適応することによって，提案手法の有用性と移植性の実証を行う．今回の評価では，big.LITTLE のマルチコアを搭載した開発ボードを用いたが，タスクを実行しているのは cpu2 もしくは cpu4 の 1 コアのみである．しかし，近年ではマルチコアでタスクを処理することが主流となっている．そこで，今後はタスクの実行に複数のコアを用いた評価を行なう．

RSM

謝辞 本研究の一部は，JSPS 科研費 26870303 の助成による．また，本研究の一部は，文部科学省情報技術人材育成のための実践教育ネットワーク形成事業「分野・地域を越えた実践的情報教育協働ネットワーク enPiT」の支援を受けて実施したものである．

参考文献

[1] Kumar, R., et al.: Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction, *Proc. of MICRO*, pp. 81-92 (2003)
[2] NVIDIA Corporation: Variable SMP - A Multi-Core CPU Architecture for Low Power and High Performance, *White Paper* (2011).
[3] Greenhalgh, P.: Big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7, *White Paper* (2011).
[4] Jeff, B.: Advances in big.LITTLE Technology for Power and Energy Savings, *White Paper* (2012).
[5] Shin, Y. et al.: 28nm High-Metal-Gate Heterogeneous Quad-Core CPUs for High-Performance and Energy-Efficient Mobile Application Processor, *Proc. of Int'l Solid-State Circuit Conference*, pp. 154–155 (2013).
[6] Wong, D. and Annavaram, M.: KnightShift: Scaling the Energy Proportionality Wall Through Server-Level Heterogeneity, *Proc. of IEEE/ACM Int'l Sympo. on Mi-*

表 3 消費電力

Table 3 Power Consumption.

消費電力 [W]	負荷	DPM.big	DPM.LITTLE	S-DVFS.big	S-DVFS.LITTLE	CPUmig	D-DVFS	S-DVFS.DPM.big	S-DVFS.DPM.LITTLE
	5%	0.8550	0.3590	0.2348	0.1516	0.6063	0.1506	0.2284	0.1524
	10%	0.8080	0.3557	0.2338	0.1513	0.6005	0.1510	0.2301	0.1504
taskset1	20%	0.8337	0.3694	0.2794	0.1922	0.5775	0.1720	0.2759	0.1917
実行時間大	40%	1.1939	0.5190	0.4068	0.2994	0.7840	0.3220	0.4035	0.2989
	60%	1.3730	0.6053	0.5379	0.4819	0.9986	0.4880	0.5365	0.4798
	80%	1.7854	-	0.8393	-	1.5029	0.9998	0.8403	-
	5%	0.2544	0.2257	0.1803	0.1382	0.1663	0.1349	0.1637	0.1375
	10%	0.3369	0.1940	0.1956	0.1457	0.1862	0.1525	0.1908	0.1461
taskset2	20%	0.4697	0.3717	0.2251	0.1706	0.2769	0.2003	0.2259	0.1704
実行時間小	40%	0.7423	0.4464	0.3237	0.2294	0.4248	0.3248	0.3243	0.2280
	60%	1.0157	0.4573	0.4224	0.3580	0.6325	0.4654	0.4248	0.3487
	80%	1.2869	-	0.6374	0.5891	-	0.7641	0.6347	-
	5%	0.7498	0.3316	0.2340	0.1543	0.5486	0.1492	0.2352	0.1497
	10%	0.7883	0.3488	0.2352	0.1547	0.5726	0.1520	0.2357	0.1516
taskset3	20%	0.8293	0.3820	0.2814	0.1948	0.6081	0.1556	0.2811	0.1942
ばらつき大	40%	1.1689	0.5211	0.4082	0.3024	0.8584	0.3250	0.4088	0.2999
	60%	1.3836	0.6161	0.5396	0.4833	1.0604	0.5121	0.5411	0.4874
	80%	1.7273	-	0.8444	-	1.2860	1.0263	0.8414	-
	5%	0.2521	0.2326	0.1815	0.1396	0.1598	0.1353	0.1663	0.1380
	10%	0.3232	0.1960	0.2023	0.1475	0.2432	0.1532	0.2031	0.1472
taskset4	20%	0.4726	0.3680	0.2283	0.1724	0.2428	0.2056	0.2287	0.1709
ばらつき小	40%	0.7594	0.5004	0.3357	0.2305	0.4258	0.3427	0.3373	0.2300
	60%	1.0417	0.4703	0.4385	0.3613	0.7822	0.4795	0.4412	0.3566
	80%	1.3170	-	0.6546	0.5908	0.6830	0.8835	0.7050	-
	5%	0.5992	0.2715	0.2237	0.1526	0.3177	0.1484	0.2238	0.1521
	10%	0.6641	0.2837	0.2259	0.1531	0.3616	0.1495	0.2246	0.1524
taskset5	20%	0.7450	0.4136	0.2628	0.1730	0.3938	0.2045	0.2620	0.1715
実行時間大中小	40%	0.9954	0.4249	0.3345	0.2064	0.5397	0.3176	0.3347	0.2069
	60%	1.2030	0.4584	0.4353	0.2938	0.6393	0.6301	0.4373	0.2903
	80%	1.4264	0.5278	0.7317	0.4080	-	1.2168	0.7295	0.4069

croarchitecture, pp. 119–130 (2012).

[7] 高瀬, 他: 排他動作する非均質マルチコアプロセッサとそのリアルタイム OS の実装, 情報処理学会研究報告, Vol. 2014-SLDM-165, No. 15 (2014).

[8] Hu, S. X., et al.: *Fundamental of Power-aware Scheduling, Designing Embedded Processors: A Low Power Perspective*, Springer (2007).

[9] Pouiklis, G. and Sirakoulis, G. C.: Clock gating methodologies and tools: a survey, Int'l Journal on Circuit Theory and Applications, (2015).

[10] Aydin, H., et al.: Determining optimal processor speeds for periodic real-time tasks with different power characteristics, *Proc. of ECRTS*, pp. 225–232 (2001).

[11] Ishihara, T. and Yasuura, H.: Voltage Scheduling Problem for Dynamically Variable Voltage Processors, *Proc. of ISLPED*, pp. 197-202 (1998).

[12] Kim, W., et al.: Performance comparison of dynamic voltage scaling algorithms for hard real-time systems, *Proc. of RTAS*, pp. 219–228 (2002).

[13] Bhatti, K. M., et al.: Hybrid power management in real time embedded systems: an interplay of DVFS and DPM techniques, *Real-Time Systems*, Vol. 47 No. 2, pp.143–162 (2011).

[14] 岩田, 他: 同一命令セットヘテロジニアスマルチコアのための消費エネルギーを削減するタスクスケジューリング, 情報処理学会研究報告, Vol. 2015-EMB-36, No. 33 (2015).

[15] Matthew R. Guthaus, et al.: MiBench: A free, commercially representative embedded benchmark suite, *Proc of IEEE 4th Annual Workshop on Workload Characterization*, (2001).

[16] Samsung: Exynos 5 Octa (5422) (online), `http://www.samsung.com/global/business/semiconductor/product/application/detail?productId=7978&iaId=2341` (2014).

[17] Hardkernel: Odroid XU3 - ODROID (online), `http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127` (2014).

[18] Texas Instruments: INA231 (online), `http://www.ti.com/product/ina231`.

[19] Pillai, P. and Shin, K. G.: Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems, *Proc. of SOSP*, pp. 89–102 (2001).

表 4 S-DVFS における動作周波数
Table 4 Voltage of S-DVFS methods.

動作周波数	負荷	S-DVFS.big	S-DVFS.LITTLE	S-DVFS.DPM.big	S-DVFS.DPM.LITTLE
	5%	200MHz	200MHz	200MHz	200MHz
	10%	200MHz	200MHz	200MHz	200MHz
taskset1	20%	300MHz	400MHz	300MHz	400MHz
実行時間大	40%	600MHz	800MHz	600MHz	800MHz
	60%	900MHz	1.2GHz	900MHz	1.2GHz
	80%	1.3GHz	-	1.3GHz	-
	5%	200MHz	200MHz	200MHz	200MHz
	10%	200MHz	200MHz	200MHz	200MHz
taskset2	20%	200MHz	300MHz	200MHz	300MHz
実行時間小	40%	500MHz	600MHz	500MHz	600MHz
	60%	800MHz	1.0GHz	800MHz	1.0GHz
	80%	1.2GHz	1.4GHz	1.2GHz	-
	5%	200MHz	200MHz	200MHz	200MHz
	10%	200MHz	200MHz	200MHz	200MHz
taskset3	20%	300MHz	400MHz	300MHz	400MHz
ばらつき大	40%	600MHz	800MHz	600MHz	800MHz
	60%	900MHz	1.2GHz	900MHz	1.2GHz
	80%	1.3GHz	-	1.3GHz	-
	5%	200MHz	200MHz	200MHz	200MHz
	10%	200MHz	200MHz	200MHz	200MHz
taskset4	20%	200MHz	300MHz	200MHz	300MHz
ばらつき小	40%	500MHz	600MHz	500MHz	600MHz
	60%	800MHz	1.0GHz	800MHz	1.0GHz
	80%	1.2GHz	1.4GHz	1.3GHz	-
	5%	200MHz	200MHz	200MHz	200MHz
	10%	200MHz	200MHz	200MHz	200MHz
taskset5	20%	300MHz	300MHz	300MHz	300MHz
実行時間大中小	40%	500MHz	500MHz	500MHz	500MHz
	60%	800MHz	800MHz	800MHz	800MHz
	80%	1.3GHz	1.1GHz	1.3GHz	1.1GHz