情報処理学会論文誌　コンピューティングシステム　Vol.8　No.1　34–50 (Mar. 2015)

粗粒度な電圧ドメインを持つメニーコアプロセッサ向け低消費電力化タスクスケジューリング

和田 康孝[1,†1,a)]　近藤 正章[2]　本多 弘樹[1]

受付日 2014年7月23日, 採録日 2014年12月4日

概要：消費電力・消費エネルギーの低減は今日のあらゆるコンピュータシステムにおいて最も重要な課題の 1 つである．電力効率の向上のため，DVFS（Dynamic Voltage/Frequency Scaling）や PG（Power Gating）が広く用いられているが，並列アプリケーションに対してこれらの技術を適切に適用するのは難しい．加えて，メニーコア化により 1 チップ上に集積されるコア数の増加は著しく，将来的にはコア単位での電圧制御は難しくなると考えられる．そのため，チップ単位あるいは複数コア単位で電圧制御が可能なメニーコアにおいて，DVFS を効率良く適用する手法が求められる．本論文では，電圧の制御単位が粗く，その中に複数のコアが含まれるようなメニーコアに対して，並列アプリケーション内の各タスクを適切に割り当て，DVFS の効果を向上させるタスクスケジューリング手法を提案する．提案手法では，ドメイン内の各コアの状況とアプリケーション実行のデッドラインを考慮してタスク割当てを行うことにより，ホモジニアスメニーコアおよびヘテロジニアスメニーコアの両方において，アプリケーション実行時の消費エネルギー削減を可能とする．提案手法を用いることにより，特にヘテロジニアスメニーコアにおいて，コア単位で電圧制御が可能な環境を想定した従来手法と比較して，より高いエネルギー削減効果を得ることができた．

キーワード：タスクスケジューリング，メニーコア，低消費電力化，グリーンコンピューティング，DVFS

Energy-aware Task Scheduling for a Manycore Processor with Coarse Grain Voltage Domains

Yasutaka Wada[1,†1,a)]　Masaaki Kondo[2]　Hiroki Honda[1]

Received: July 23, 2014, Accepted: December 4, 2014

Abstract: Power/Energy efficiency is one of the most important issues for today's computer systems. To improve the efficiency, DVFS (Dynamic Voltage/Frequency Scaling) and PG (Power Gating) can be utilized on many current multicore/manycore processors. However, applying them to parallel applications imposes large burdens upon the programmers. In addition, it would become difficult to utilize per-core DVFS on a manycore processor because of the increasing number of cores, and DVFS can ruin its effectiveness. This paper proposes an energy-aware task scheduling algorithm for a manycore processor with coarse grain voltage domains. The proposed scheme assigns tasks in an input parallel application considering performance and power efficiency of each core and DVFS efficiency of each voltage domain. This makes it possible to improve DVFS efficiency on a manycore processor with coarse grain voltage domains. The proposed scheme gives us better energy reduction with a heterogeneous manycore in comparison with the case that a conventional scheduling method is applied on a per-core DVFS enabled manycore.

Keywords: task scheduling, manycore processor, low power, green computing, DVFS

1. はじめに

HPC システムから組込みシステムまであらゆる計算機システムにおいて，電力効率・エネルギー効率の向上は共通の重要な課題となっている．多くの計算機システムではプロセッサの消費電力が大きな割合を占めるため，半導体集積技術の向上とともに，マルチコア化・メニーコア化および並列処理による電力効率・エネルギー効率の向上が進

[1] 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications, Chofu, Tokyo 182–8585, Japan
[2] 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo, Bunkyo, Tokyo 113–8656, Japan
[†1] 現在，早稲田大学理工学術院基幹理工学研究科
Presently with Graduate School of Fundamental Science and Engineering, Waseda University
[a)] yasutaka@kasahara.cs.waseda.ac.jp

んでいる．

また近年では，並列処理による性能向上と逐次処理性能の維持を両立させるため，様々な種類のコアをチップ上に集積するヘテロジニアスマルチコア・メニーコアも広く用いられるようになった [1], [2]．ヘテロジニアスメニーコアでは搭載するコアの種別ごとに性能や電力効率が異なるため，並列アプリケーションを効率良く実行するためには，各コアの特性とアプリケーション内部の各タスクの依存関係や特性を考慮して並列処理を行う必要がある．

さらに，アプリケーション実行時の消費電力あるいは消費エネルギーをより削減するために，使用しないコアや回路に対する電力供給を遮断するパワーゲーティング（Power Gating，PG）や，動的に周波数と電圧を制御する DVFS（Dynamic Voltage/Frequency Scaling）といった低消費電力制御機能が広く用いられている．これらの機能を適切に用いることにより，アプリケーションの性能低下を抑えつつ，さらなる電力効率向上を達成することができる．

しかし，プロセッサコア数が任意のシステムに対して並列アプリケーション内の各タスクを割り当てるマルチプロセッサスケジューリング問題は，従来より強 NP 困難な問題であることが広く知られており [3]，これまで多くのヒューリスティック解法が提案されてきた．並列アプリケーションに対して DVFS や PG を効果的に適用するには，この複雑なタスクスケジューリング問題において，DVFS による各タスクの実行時間の変化や，またそれによる後続タスクへの影響，アプリケーション実行完了のデッドライン等を同時に考慮する必要がある．そのため，並列アプリケーションに対してこれらを適用し，適切な制御を行うことは難しいといえる．また，プログラマが PG や DVFS を適用するタイミングを決定しアプリケーションを最適化する必要があり，開発コストが問題となるため，効率の良い適用手法およびその自動化が求められる．

メニーコアプロセッサにおいて DVFS を適用する際，現在の多くのプロセッサでは電圧の制御単位がチップ一括となっている．この場合，電圧の制御がコア単位で可能である場合と比較して，20%以上電力効率が悪化してしまうという報告もある [4]．そのため，現在ではより細粒度な電圧制御を行うための研究開発が行われており，チップ上に電圧レギュレータを統合して実装する技術 [5], [6] が用いられるようになった．その結果，少ないコア数であれば，Intel の Haswell プロセッサ [7] や Qualcomm の Snapdragon プロセッサ [8] 等のように，コア単位での周波数・電圧制御が可能となっている．しかしながら，これらのプロセッサはコア数が少なく，シンプルなコアを数十から数百基搭載するメニーコアプロセッサ [9], [10] や GPGPU [11] 等への適用はいまだ難しい．また，今後チップ上に集積されるコア数がさらに増加していくことと比較して，電圧レギュレータ自身の電力ロスが大きい（10～20%以上）ことや実装コストの問題から [5], [6], [12], [13]，電力制御ドメインの細分化は難しいものと考えられる．

以上から，チップ単位あるいは複数コア単位での電圧制御，つまり粗粒度な電圧制御のみが可能な環境に対して適切に DVFS を適用していくことが求められる．本論文では，粗粒度な電圧制御のみが可能なホモジニアスメニーコアおよびヘテロジニアスメニーコアを対象として，DVFS を効率良く適用し，消費エネルギーを削減するためのタスクスケジューリング手法を提案する．

本論文の構成は以下のとおりである．2 章では関連研究について紹介する．3 章では本論文で対象とする並列アプリケーションおよびメニーコア，低消費電力制御のモデルについて述べる．4 章では粗粒度な電圧制御ドメインを持つホモジニアスメニーコアを考慮したタスクスケジューリング手法について述べる．5 章では粗粒度な電圧制御ドメインを持つヘテロジニアスメニーコアを考慮したタスクスケジューリング手法について述べる．6 章で標準タスクグラフセット [14] および実アプリケーションを想定したタスクグラフを用いた性能評価について述べる．最後に，7 章でまとめる．

2. 関連研究

メニーコアプロセッサ上で並列アプリケーションを効率良く実行し，かつ消費電力・消費エネルギーを削減するためには，当該アプリケーション内部のタスクをチップ上の各コアに適切に割り当て，適切に DVFS や PG を適用する必要がある．

従来，消費電力および消費エネルギーが特に問題となるのは組込みシステムであったため，組込みシステム向けのマルチコア上でタスクのデッドラインを考慮して低消費電力化を行う手法が考案されてきた [15]．たとえば，周期タスクとそのデッドラインを考慮して消費電力を削減するタスクスケジューリング手法が複数提案されている [16]．これらの手法の多くはタスク間に依存関係のない場合を想定しているが，それでもタスク割当てと動作周波数の組合せの多さから，NP 困難な問題であることが知られている [17]．

マルチコアプロセッサやクラスタシステムの多くは各コアやノードの性能が均一なホモジニアスシステムであり，このようなシステムにおける低消費電力化手法は多く提案されている．たとえば，内部のタスク間に依存関係のある並列アプリケーションを想定し，DVFS の適用によって消費エネルギーの削減を行う低消費電力化手法として，タスク間の依存関係によって生じる同期待ちの時間を利用して DVFS を適用する手法 [18] や，タスク間の依存関係を考慮して通信が少なくなるようにタスクを割り当てたうえで DVFS を適用する手法 [19], [20]，消費エネルギー削減に貢献する場合のみタスクの複製を行い，クラスタ内ノード間

の通信を削減する手法 [21] 等が提案されている．

近年では，クラウドシステム等のように，各計算ノードやコアごとに性能の異なるヘテロジニアス環境向けの低消費電力化手法も提案されている．たとえば，タスク複製により通信オーバヘッド削減を行うスケジューリング手法に対し，不要な複製タスクを削除することで消費エネルギーを削減する手法 [22] や，アプリケーション内の各タスクを実行するために消費されるエネルギーを目的関数に取り入れ，それを用いてタスク割当先の優劣を比較する手法 [23]，デッドラインを持つ独立した多数のタスクを，デッドラインと消費電力を考慮して割り当てる手法 [24], [25] 等がこれまでに提案されている．

その他，実アプリケーションの階層的な並列性と低消費電力化制御のオーバヘッドを考慮して並列化コンパイラにより自動的に DVFS と PG を適用する手法 [26], [27] や，DVFS および CPU と GPGPU 間での処理分担の最適化により消費エネルギーを削減する手法 [28] 等も提案されている．

しかしながら，これらのスケジューリング手法はいずれも各コア・ノード単位で DVFS を適用可能な環境を前提としており，電圧の制御が複数のコア単位でのみ可能なシステムは考慮されておらず，本論文の提案手法とは大きく異なる．

3. 対象モデル

本章では，提案するタスクスケジューリング手法が対象とする並列アプリケーションおよびメニーコアのモデルについて述べる．提案スケジューリング手法では，本章で述べるモデルを対象として，アプリケーションの実行前に静的にタスクの実行順・割当て先コア・動作周波数を決定する．本論文では，本章で述べる並列アプリケーションおよびメニーコアのモデルに基づき，シミュレーションにより提案スケジューリング手法の評価を行う．

なお，コア単位での細粒度な DVFS が適用できるメニーコアを対象としたものではあるが，本論文と同様の動作周波数および消費電力に対するモデルを用いた低消費電力化手法が提案されており [26]，実機上で高い低消費電力化効果を得られている [29]．また，理論的な低消費電力化タスクスケジューリング問題では，同様のモデルが広く用いられている [15]．そのため，本章で述べる消費電力モデルに基づく評価と比較を通じて，十分に手法の有効性を確かめることができる．

3.1 並列アプリケーションモデル

ここでは，本論文が対象とする並列アプリケーションおよびその実行形式のモデルについて述べる．本論文では粗粒度タスク並列性を持つアプリケーションを対象としており，アプリケーション内の粗粒度タスクおよびタスク間の

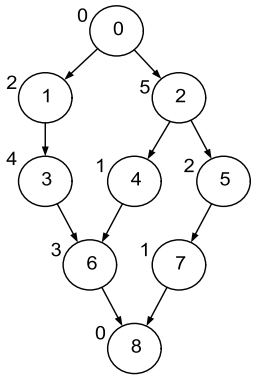

図 1　タスクグラフの例
Fig. 1　An example of task graph.

依存関係は有向非循環グラフ（DAG）によって表現される．つまり，n 個の粗粒度タスクを表すノード集合 N およびタスク間の依存関係を表すノード間の有向エッジ集合 E からなるタスクグラフ $G = (N, E)$ として記述される．なお，このタスクグラフは 1 つの入口ノードと 1 つの出口ノードを持ち，すべてのノードは入口ノードから到達可能であり，他のすべてのノードから出口ノードに到達可能である．また，この入口ノードおよび出口ノードは実行時間 0 のダミーノードである．**図 1** に対象とするタスクグラフの例を示す．

図 1 において，各ノードはアプリケーション内の 1 つの粗粒度タスクを表し，ノード内の数字はタスク番号 i，ノード左上の数字は当該タスクの実行時間 t_i を表す．また，ノード i からノード j へのエッジは，タスク j はタスク i が完了しなければ実行を開始できないという制約を表している．本論文で扱うスケジューリングアルゴリズムにおいては，これらの制約を満たすように各コアにスケジューリングする必要がある．このとき，各タスクはノンプリエンプティブに実行されるものとする．

また，各タスクグラフに対しては，スケジューリング長の上限を規定するデッドラインが与えられる．提案手法においては，入力タスクグラフのスケジューリング長が与えられたデッドラインを超えない範囲でタスクの割当て先コア，実行順，動作周波数を決定する．

3.2 粗粒度な電圧ドメインを持つメニーコアプロセッサ

1 章で述べたとおり，将来的には 1 チップ上に集積されるプロセッサコア数が増加しメニーコア化が進んでいくことが考えられる．そのため，動作周波数はコア単位あるいはそれより細粒度な単位で制御することが可能であっても，コア単位で電源電圧を制御することは難しくなる．その結果，複数のコアをグループとして電圧制御ドメインを形成することになるものと予想される．それに対して，PG は従来からコア内の演算器単位等粒度の小さい単位でも適用されており [30]，将来的にもコア単位での適用は可能であ

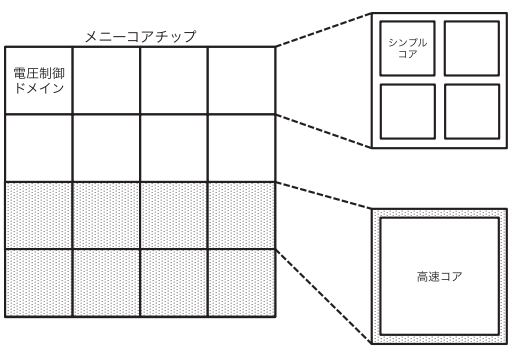

図 2　対象とするメニーコアの例
Fig. 2　An example of a manycore processor.

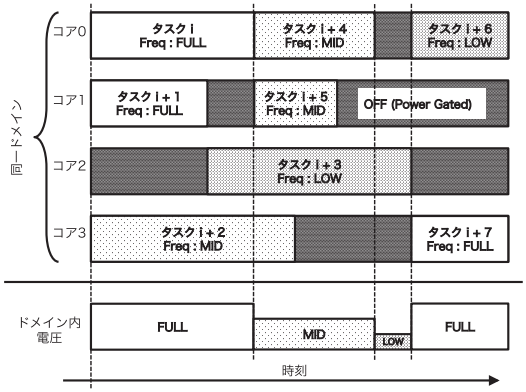

図 3　DVFS・PG の適用例
Fig. 3　DVFS and PG in a coarse grain voltage domain.

ると考えられる．

以上の点を考慮して，本論文において対象とするメニーコアプロセッサのモデルを下記のように定める．

- パワーゲーティング（PG）は各コア個別に適用可能．
- 動作周波数はコアごとに動的制御可能．
- 設定可能な動作周波数は離散的である．
- 各コアに割り当てられたタスクの実行時間は，コアの動作周波数に反比例する．
- 各動作周波数で動作するために必要な電圧も一意に定められている．
- 電圧は電圧制御ドメイン単位で制御可能であり，1 つのドメインには 1 つ以上のコアが含まれる．
- 1 つの電圧制御ドメインは同じ種別のコアのみから構成される．
- 各電圧制御ドメインの電圧は，ドメイン内で最も高い周波数で動作しているコアにあわせて，動的に設定される．

図 2 に対象メニーコアプロセッサの例を示す．図 2 は性能・面積の異なるコアを集積したヘテロジニアスメニーコアの例であり，16 の電圧制御ドメインからなる．そのうち半分はシンプルなコアを 4 基含んでおり，残りのドメインは高速なコアを 1 基含む．シンプルなコアからなるドメインには，この 4 基のコアのうち，最も高い動作周波数で動作しているコアにあわせた電圧が供給される．これ対して，高速なコア 1 基のみを含むドメインに供給される電圧は，このコアの動作周波数のみに依存する．

また，実際のマルチコアシステム上での評価においては，DVFS や PG を適用する際の遷移オーバヘッドは 10 マイクロ秒程度あるいはそれ以下であるため [31]，タスクの実行時間（10 ミリ秒程度以上を想定）に対して十分小さく，無視できるものとする．

3.3　電力モデル

ここでは，上記で述べたメニーコアモデルにおいて，どのように PG および DVFS が適用され，消費電力・消費エネルギーがどのようにモデル化されるかについて述べる．

なお，対象とするモデルにおいては，3.2 節で述べたとおり，周波数制御および PG はコア単位で適用可能であるものとする．ただし，これらの制御はタスクの実行開始時および実行完了時にのみ可能であり，ある 1 つのタスクを実行している間，そのタスクを実行しているコアは一定の動作周波数で駆動されるものとする．

他方，電圧制御は電圧制御ドメイン単位で可能であり，かつ動的に行われるものとする．そのため，同一ドメイン内の各コアの動作周波数の設定状況に応じて，タスクの実行途中においても電圧が制御・変更される．

3.3.1　複数コアを含む電圧ドメインにおける DVFS

ここでは，上記のマルチコアモデルおよびアプリケーションモデルにおいて，DVFS および PG がどのように適用されるかについて述べる．**図 3** に DVFS および PG の適用例を示す．図 3 は，4 基のコアが 1 つの電力制御ドメインを形成しており，タスク i からタスク $i+7$ の 8 つのタスクが割り当てられている例である．また，各コアは高い方から FULL，MID，LOW の 3 通りの動作周波数に設定可能である．

図 3 において，コア 0 がタスク i を実行している間，他のコアはそれぞれ

コア 1：動作周波数 FULL でタスク $i+1$ を実行した後電源 OFF（パワーゲーティング）．
コア 2：電源 OFF の後，周波数 LOW でタスク $i+3$ を実行．
コア 3：動作周波数 MID でタスク $i+2$ を実行．

と各コアが様々な動作周波数や状態をとる．しかし，この間コア 0 が通常の動作周波数（FULL）で高速にタスクを実行しているため，ドメイン全体には動作周波数 FULL で駆動可能な電圧が供給される．

次に，コア 0 がタスク $i+4$ を中程度の動作周波数（MID）で実行している間は，他のコアはコア 0 と同じあるいはそれより低い動作周波数で動作しており，MID がこれら 4 基のコアで最も高い動作周波数となる．その結果，ドメイン

に供給される電圧は動作周波数 MID に合わせた値に制御される．以下同様に，コア 0 がタスク $i+4$ の実行を終了する時刻からタスク $i+6$ を開始する時刻までは動作周波数 LOW にあわせた電圧が，その後は動作周波数 FULL にあわせた電圧が供給される．

以上のように，ドメイン内のコアで最も高い動作周波数で動作しているコアにあわせて，供給される電圧が自動的に制御される．

3.3.2　タスク実行時の消費電力および消費エネルギー

プロセッサの消費電力は主として動的電力と静的電力からなり，動的電力は電圧の 2 乗および動作周波数に比例することが知られている．また，本論文では，動作状態にあるコアはつねに静的電力を消費するものとする．つまり，各コアの消費電力 P_{core} は，動的電力を $P_{dynamic}$，静的電力を P_{static}，C を定数，f_{core} を当該コアの動作周波数，I_{leak} を当該コアのリーク電流，V_{domain} を当該コアが属する電圧制御ドメインの電圧とすると，式 (1) のように表される．

$$\begin{aligned} P_{core} &= P_{dynamic} + P_{static} \\ &= C \cdot f_{core} \cdot (V_{domain})^2 + I_{leak} \cdot V_{domain} \end{aligned} \quad (1)$$

よって，あるタスクの実行開始から終了までに消費されるエネルギー E_{task} は，T_{start} を当該タスクの開始時刻，T_{end} を当該タスクの終了時刻とすると，式 (2) のように計算される．なお，上述のとおり，T_{start} から T_{end} までの間，当該タスクを実行するコアの動作周波数は f_{core} で一定であるものとする．また，I_{leak} は V_{domain} に依存しており，その程度は半導体プロセス等によって異なる [32]．

$$\begin{aligned} E_{task} &= \int_{T_{start}}^{T_{end}} P_{core}(t) \cdot dt \\ &= \int_{T_{start}}^{T_{end}} (P_{dynamic}(t) + P_{static}) \cdot dt \\ &= C \cdot f_{core} \cdot \int_{T_{start}}^{T_{end}} (V_{domain}(t))^2 \cdot dt \\ &\quad + \int_{T_{start}}^{T_{end}} I_{leak}(t) \cdot V_{domain}(t) \cdot dt \end{aligned} \quad (2)$$

なお，本論文では，コアの動作周波数と電圧が一定である間はコアの消費電力も一定であり，タスクを実行していない期間，当該コアは PG が適用されるため動的電力・静的電力どちらも消費しないものとする．

4.　ホモジニアスメニーコア向け低消費電力化タスクスケジューリング

本章では，3 章で述べた並列アプリケーションおよび電圧制御の単位となる電圧ドメイン内に複数のコアを含むメニーコアプロセッサにおける，DVFS を用いた低消費電力化タスクスケジューリング手法について述べる．ここでは特に，チップ上のすべてのコアが同一の性能を持つホモジニアスメニーコア向けの低消費電力化タスクスケジューリング手法について述べる．ホモジニアスメニーコアの場合，各コアにおいてタスク実行時の電力効率は同一であるため，DVFS 適用後の各タスクの最低動作周波数を予測し，これが同じタスクを同時期に同じドメイン内のコアに割り当てることで DVFS の効果を高めることができると考えられる．

複数のコアが電圧ドメイン内に含まれる場合，ドメイン内の最も動作周波数の大きいコアにあわせて電圧が決定される．このような条件において DVFS を効果的に適用するためには，各タスクを実行する際の動作周波数を小さくするとともに，同一ドメイン内のコアの動作周波数を均一にする必要がある．

そこで本提案手法では，タスク間の依存関係と CP/MISF 法 [3] による事前の仮スケジューリング結果をもとに，各タスクの最早開始時刻と最遅終了時刻を算出する．その差分から適用可能な最低動作周波数を予測することができるため，この情報を利用することで，後述の DVFS 適用時に，同じ動作周波数が設定される可能性が高いタスクを同じドメインのコアに割り当てるようにする．なお，CP/MISF 法は，入力タスクグラフ中の各タスクに対し，出口ノードからの最長パス長をスケジューリングのための優先度として用いるリストスケジューリング手法である．同一の CP 長を持つタスクが複数存在する場合，直接後続タスクの数が大きいものを優先する．ここで，CP/MISF 法でスケジューリングを行う際は電圧ドメインを考慮せず，すべてのコアを平等に対象として扱う．つまり，割当て先コアの数は「ドメイン数 × ドメイン内コア数」となる．

また，ヒューリスティックアルゴリズムである CP/MISF 法をベースとすることで短時間でスケジューリング結果を得ることができる．これにより，並列化支援ツールや並列化コンパイラ等の環境上で利用する場合に対しても有用性を高められる．

4.1　最遅終了時刻と最早開始時刻の差分による最低動作周波数の算出

ホモジニアスメニーコア向けの提案手法では，まず，タスク間の依存関係と与えられたデッドラインから，各タスクの最早開始時刻と最遅終了時刻を求め，これら 2 つの時刻の差から当該タスクがとりうる最も低い動作周波数を算出する．同時期に実行開始されるタスクのうち，この最低動作周波数が同じタスクを同じ DVFS ドメインのコアに割り当てられるように割当て先コアを選択することで，DVFS ドメインに含まれるコアの動作周波数が一致する可能性が高まり，効果的に DVFS が適用されることが期待できる．

図 4 に最遅終了時刻と最早開始時刻をもとにした最低動作周波数算出の例を示す．図 4 a) は，CP/MISF 法により 2 基のコアに対して時刻 0 からスケジューリングを行った結果である．また，図 4 b) は，図 4 a) のタスク実行順とタ

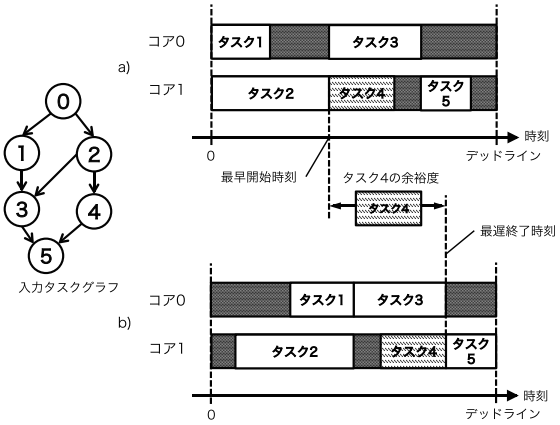

図 4　最遅終了時刻と最早開始時刻
Fig. 4　Latest finish time and earlist start time.

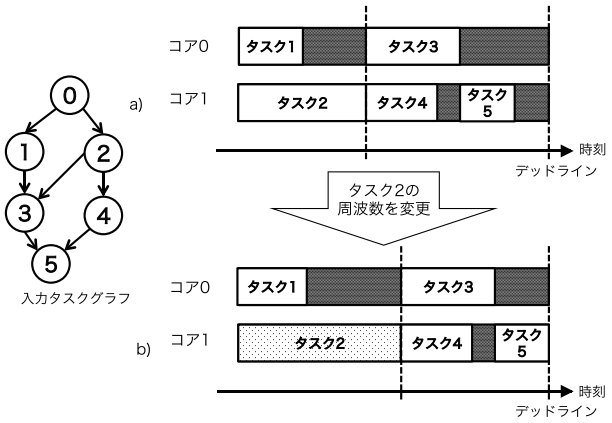

図 5　DVFS の適用
Fig. 5　An example of DVFS application on a homogeneous manycore.

スク間の依存関係を保ったままタスクの開始時刻を可能な限り遅くしたものである．つまり，図 4 a) から各タスクの最も早く実行開始される時刻（最早開始時刻）が，図 4 b) から各タスクが最も遅く実行終了する時刻（最遅終了時刻）が分かる．たとえば，図 4 a) のタスク 4 の最早開始時刻は CP/MISF 法によるスケジューリング結果から，タスク 2 の終了時刻となり，最遅終了時刻は，デッドラインからタスク 5 の実行時間分さかのぼった時刻となる．タスク 4 の実行時間が最遅終了時刻と最早開始時刻の差分（図中の余裕度）を超えない最も小さい動作周波数がタスク 4 の最低動作周波数となる．このようにして，各タスクがとりうる最も低い動作周波数を算出する．ただし，ここで選択可能な動作周波数は，対象システムにおいて設定可能な動作周波数のみである．

なお，この時点で利用する CP/MISF 法によるスケジューリング結果は各タスクの最遅終了時刻と最早開始時刻を求めるためのものであり，最終的なタスクの実行順は次節の手法を用いて改めて決定する．

4.2　タスクスケジューリング

各タスクのとりうる最低動作周波数の算出を行った後，CP/MISF 法をベースとした手法によりタスクの割当て先および割当て順を決定する．このスケジューリング手法では，CP/MISF 法の各スケジューリング時刻において各タスクの割当て先を選択する際，下記の優先度に基づいて割当て先のコアを決定する．

1. 当該スケジューリング時刻における予想動作周波数が，現在割当てを行っているタスクの最低動作周波数と同じドメインから空いているコアを選択．
2. 当該スケジューリング時刻において，すべてのコアがアイドルであるドメインからコアを選択．
3. 当該スケジューリング時刻における予想動作周波数が，現在割当てを行っているタスクの最低動作周波数より大きいドメインから空いているコアを選択．

ここで，DVFS ドメインの予想動作周波数とは，あるスケジューリング時刻において当該ドメインに含まれるコアに割り当てられているタスクの最低動作周波数のうち最も高いものをいう．なお，このタスクスケジューリングを行った時点，つまり各タスクの割当て先コアと実行順序を決定した時点では，まだ各タスクの動作周波数は決定されない．各タスクのとりうる最低動作周波数を割当て先コアの選択の指針として利用することで，4.3 節で適用する DVFS の効果を高める．なお，このタスクスケジューリングの時点では，各タスクの実行時間として DVFS を適用しない場合のものを用いる．これは，この時点で各タスクのコストとして最低動作周波数によるものを用いると，スケジューリング長をデッドライン以下に保つ制約が守れない可能性があるからである．

4.3　DVFS の適用

4.2 節のようにしてタスクの割当て先・実行順の決定を行った後，各タスクの動作周波数を実際に Algorithm 1 の手順で決定する．なお，この手順は Wang らが提案した手法 [19] を参考に，対象モデルへの適用が可能となるように変更を施したものである．

図 5 に本手法の適用例を示す．図 5 a) は本手法を適用する前のスケジューリング結果，図 5 b) は図中のタスク 2 を対象として動作周波数を設定・変更した段階の図である．また，CP タスクはタスク 2，3，5 である．図中，タスク 3 およびタスク 4 はタスク 2 に依存しているため，タスク 2 に DVFS を適用して実行時間が延びた分，開始時刻・終了時刻を遅らせる．それにともない，これらのタスクに依存するタスク 5 の開始時刻・終了時刻も遅らせる．この例では，この時点でタスク 5 の終了時刻がデッドラインに到達するため，CP タスクに対する DVFS の適用はここで終了する．この後，非 CP タスク（タスク 1 および 4）は CP タスクの開始・終了時刻に影響を及ぼさない範囲で，同様

Algorithm 1 ホモジニアスメニーコアにおける DVFS

Input: $G(N, E)$ :: 入力タスクグラフ
$Sched$ :: DVFS 適用前のスケジューリング結果
$SchedLength$ :: スケジューリング長
$CPTask[]$ :: CP タスクのリスト（実行時間の降順にソート）
N_{CPTask} :: CP タスクの数
$nonCPTask[]$ :: 非 CP タスクのリスト（実行時間の降順にソート）
$N_{nonCPTask}$ :: 非 CP タスクの数
$Dline$ :: スケジューリング長の上限（デッドライン）
Output: $Sched$ :: DVFS 適用後のスケジューリング結果
$Freq[]$ :: 各タスクを実行する際の動作周波数

```
1: Margin ← (Dline − SchedLength)
2: Freq[] を最高動作周波数にすべて初期化
3: while Margin ≥ 0 do
4:   for i = 0 to N_CPTask do
5:     if Freq[CPTask[i]] が最低動作周波数でない then
6:       Diff ← Freq[CPTask[i]] を 1 段階下げることによる実
         行時間の増分
7:       if Margin ≥ Diff then
8:         Freq[CPTask[i]] を 1 段階下げる
9:         Sched 中の CPTask[i] の終了時刻を Diff 増やす
10:        Sched 中の CPTask[i] に依存するすべてのタスクの開
           始・終了時刻を Diff 増やす
11:        Margin ← Margin − Diff
12:      end if
13:    end if
14:  end for
15: end while
16: 現在の Freq[] および Sched に基づいて各非 CP タスクの最早開始
    時刻・最遅終了時刻を求める
17: for f = 最高動作周波数 to 最低動作周波数 do
18:   for i = 0 to N_nonCPTask do
19:     if Freq[nonCPTask[i]] > f then
20:       Margin ← nonCPTask[i] の最遅終了時刻 − 最早開始
          時刻
21:       Diff ← Freq[nonCPTask[i]] を f に変更した際の実行
          時間の増分
22:       if Margin ≥ Diff then
23:         Freq[nonCPTask[i]] ← f
24:         Sched 中の nonCPTask[i] の終了時刻を Diff 増やす
25:         Sched 中の nonCPTask[i] に依存するすべてのタスク
            の開始・終了時刻を Diff 増やす
26:         各非 CP タスクの最早開始時刻・最遅終了時刻を更新する
27:       end if
28:     end if
29:   end for
30: end for
```

に動作周波数を決定していく．

以上の手順により，各コアでのタスクの実行順序および各タスク実行時の動作周波数が決定される．CP タスクに対して先に DVFS を適用して実行時間を決定し，非 CP タスクに対しては CP タスクに影響しない範囲で DVFS を適用することで，アプリケーションの終了時刻（スケジューリング長）をデッドライン以下に保つことを容易にしている．

4.4　スレッドのグルーピング

4.2 節および 4.3 節の手順により，タスクの実行順序および動作周波数を決定するが，各タスクがとりうる最低動作周波数と実際に DVFS を適用した後の動作周波数は異なる場合が考えられる．この場合，電圧を効率良く下げることができない．そのため，4.2 節の時点で同じコアに割り当てられたタスクをスレッドとしてまとめて取り扱い，動作周波数の変化の傾向が同じものを同じ電圧ドメイン内のコアに割り当て，異なる傾向を持つものを別の電圧ドメイン内のコアに割り当てることを考える．これにより，同じ DVFS ドメイン内のコアが同じタイミングで同じ動作周波数に制御されている機会をより多く確保することができ，ドメインの電圧を下げる機会を増やすことができる．ここで，スレッドはある 1 つのコアに割り当てられたタスク群のことをいい，各タスクの開始・終了時刻と実行の順番，4.3 節で決定された動作周波数の情報を含む．

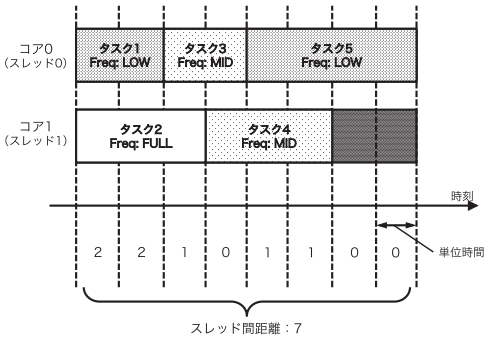

図 6　スレッドの距離
Fig. 6　Distance among threads.

この手法では，各スレッド間の距離を算出し，DVFS ドメイン内のスレッド間距離の合計を最適化する．ここでいうスレッド間の距離とは，各時点でのスレッドの動作周波数状態の差分を積算したものである．たとえば，**図 6** において，最初の 2 単位時間の間はコア 0 が実行するスレッドが周波数 LOW，コア 1 が周波数 FULL で 2 段階異なるため，単位時間あたりの距離を 2 として積算する．次の 1 単位時間はコア 0 が動作周波数 MID，コア 1 が FULL で 1 段階異なるため 1 を積算する．なお，動作周波数が同じ場合，あるいはどちらかがパワーゲーティングされている場合は距離 0 である．このようにして，アプリケーションの開始から終了までスレッド間の距離を積算し，動作周波数の推移の相違度を求める．その後，スレッドの入れ替えによってドメイン内スレッド間の距離，つまり相違度の合計が小さくなるようにスレッドとコアの対応付けの入れ替えを順次行う．スレッドとコアの対応付けを行う手順を Algorithm 2 に示す．

Algorithm 2 では，まず図 6 と同様にして各スレッド間の距離を求め，各ドメイン内のコアに対応付けられたスレッド間距離の合計が少なくなるよう，各ドメインに均等にスレッドを分配する．ここで，スレッド間距離の総和とは，「ドメイン内のコアに割り当てられた異なる 2 つのス

Algorithm 2 スレッドのグルーピング

Input: $G(N, E)$ :: 入力タスクグラフ
$Sched$ :: 動作周波数変更後のスケジューリング結果
$WorkThreads[]$ :: タスク割当てのあるスレッド集合
Output: $Thread[]$ :: 各スレッドの割当て先コア

```
1: N_domain ← 電圧ドメインの数
2: N_core ← 電圧ドメイン内のコア数
3: if N_core == 1 then
4:     return
5: end if
6: for i = 0 to N_domain do
7:     if 未処理のスレッド数 == 0 then
8:         break
9:     else if 未処理のスレッド数 == 1 then
10:        WorkThreads[] から未処理のスレッドを 1 つ選択
11:        選択したスレッドをドメイン i 中のコアに対応付ける
12:    else if 未処理のスレッド数 > 1 then
13:        WorkThreads[] から，スレッド間距離が最も小さい 2 スレッドを選択
14:        選択したスレッドをドメイン i 中のコアに対応付ける
15:    end if
16: end for
17: repeat
18:    for i = 0 to N_domain do
19:        if 未処理のスレッド数 == 0 then
20:            break
21:        end if
22:        WorkThreads[] 中の未処理のスレッドから，ドメイン i 内のコアに対応付けられた各スレッドとの距離の総和が最も小さいものを選択
23:        選択したスレッドをドメイン i 中のコアに対応付ける
24:    end for
25: until すべてのスレッドが処理済み
26: repeat
27:    for i = 0 to N_domain do
28:        for j = i + 1 to N_domain do
29:            if ドメイン i 内のスレッドとドメイン j 内のスレッドに，入れ替えることによって両ドメイン内のスレッド間距離の総和が減少する組合せが存在する then
30:                スレッドとコアの対応付けを入れ替える
31:            end if
32:        end for
33:    end for
34: until スレッドの入れ替えが行われない
35: スレッドの対応付けがないコアは常時 PG を適用
```

レッド間距離の総和」として計算される．

その後，異なるドメインのコアに対応付けられた 2 つのスレッドの組合せを調べ，入れ替えることで当該両ドメインのスレッド間距離の合計がともに減少する場合には，この 2 つのスレッドを入れ替える．これを繰り返すことにより，各ドメイン内のスレッド間距離の合計を削減する．Algorithm 2 を適用するにあたり，チップ上の電圧ドメインの数は比較的小さく，すべての組合せを考慮しても計算量は問題とならない．さらに，各ドメインに割り当てられるスレッドの数をあらかじめ固定しておくことで，計算量を削減している．

本論文で対象とするホモジニアスメニーコアではチップ上のすべてのコアが同じ性質を持つため，スレッドの入れ替えが適用可能である．ここでは，タスクの割当て・実行順とともに各タスクの動作周波数が決定されている必要があるため，4.2 節および 4.3 節の手順によりタスクスケジューリングおよび DVFS を行った後に本手法を適用する．

5. ヘテロジニアスメニーコア向け低消費電力化タスクスケジューリング

本章では，チップ上に性能の異なるコアが搭載されるヘテロジニアスメニーコアを対象とした，低消費電力化スケジューリング手法について述べる．なお，3.2 節で述べたとおり，本手法では，ある 1 つの電圧ドメインに属するコアは同一のものであるとする．

ホモジニアスメニーコアと異なり，ヘテロジニアスメニーコアにおいては，コアの種類ごとに電力効率や DVFS の有効性が異なる．たとえば，高速なコアの場合，シンプルなコアと比較してタスクの実行時間は短いが，一般にタスク実行時の電力効率は悪くなる [33]．その反面，コア単体の面積が大きくなるため電圧ドメイン内のコア数が小さく，ドメイン内に高い周波数で動作しているコアが存在する可能性が下がるため，電圧を下げられる可能性が高くなる．シンプルなコアからなる電圧ドメインにおいては，各コアの電力効率は高いが，ドメイン内のコア数が大きく，ドメイン内に高い周波数で動作しているコアが存在する可能性が高くなり，ドメインの電圧を下げにくくなる．さらに，各コアの性能が低いため，高速なコア上でタスクを実行する場合と比較してタスクの実行時間が長くなるという問題がある．

さらに，ヘテロジニアスマルチコアにおいては，各タスクの割当て先コアによってタスクの実行時間が異なるため，タスクスケジューリング結果によってタスクグラフのクリティカルパスが変化する．そのため，事前に与えられたデッドラインを守りつつ，各タスクに適する割当て先コア・動作周波数を決定するには組合せの数が大きくなりすぎてしまう．

これらの問題点に対応するため，提案手法においては，4 章で述べたホモジニアスメニーコア向けの手法を拡張し，大きく分けて下記の 3 つのフェーズを順番に適用する．

フェーズ 1. 事前に仮スケジューリングを行い，各タスクにとって望ましいコア種別と動作周波数を設定する．

フェーズ 2. 実際にタスクスケジューリングを行う．この際，フェーズ 1 の結果を考慮してタスクの割当て先を決定する．

フェーズ 3. フェーズ 1 の結果を考慮して，フェーズ 2 のスケジューリング結果に対して DVFS の適用を行う．

これらの各ステップは短時間で解を求めることができるヒューリスティックアルゴリズムをベースとしており，これを複数組み合わせることで，消費電力削減効果の高いス

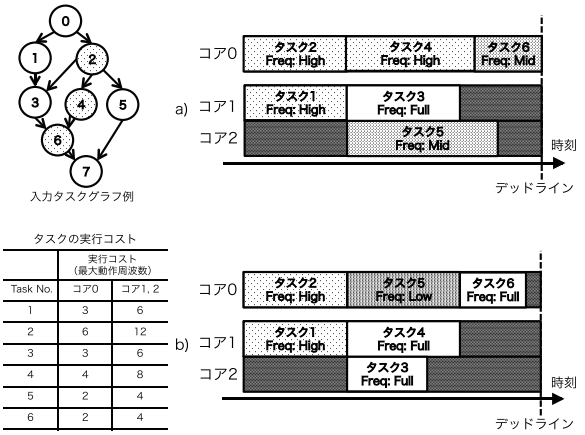

図 7　ヘテロジニアス環境における提案手法による改善例
Fig. 7　A motivative example with a heterogeneous system.

ケジューリングを短時間で求めることができる．

図 7 に，提案手法によって消費エネルギーが低減される例を示す．図 7 では，対象プロセッサは 3 コアからなり，コア 0 は単体で 1 つの電圧ドメインを，コア 1 とコア 2 はあわせて 1 つの電圧ドメインを形成する．また，動作周波数は高い方から FULL，HIGH，MID，LOW の 4 種類のいずれかに設定可能である．また，図 7 中，左上は入力タスクグラフを表し，左下の表は各タスクの各コア上での実行コスト（最大動作周波数時）を示す．

図 7 a) のスケジューリング結果は後述の HEFT アルゴリズム [34] に基づいてスケジューリングを行った場合であり，図 7 b) は提案手法によるものである．HEFT アルゴリズムを用いた場合，クリティカルパス上にあるタスク 2，4，6 が優先的に高速なコア 0 に割り当てられ，タスク 3 および 5 はコア 1 およびコア 2 に割り当てられる．これに DVFS を適用した場合，タスク 5 は動作周波数を中程度（MID）まで下げることができるが，タスク 3 はタスク間の依存関係から動作周波数を下げることができない．その結果，タスク 3 の実行中，コア 1 とコア 2 からなる電圧ドメインは電圧を下げることができず，十分な消費エネルギー削減効果を得ることができない．

これに対し，提案手法を用いた場合では，タスク 5 がコア 0 に割り当てられ，タスク 3 および 4 はコア 2 および 1 にそれぞれ割り当てられている．これは，タスク 4 の実行に必要な消費エネルギーの削減率を考慮した場合，高速なコア 0 で動作周波数を HIGH に設定して実行するよりも，エネルギー効率の良いコア 1 あるいはコア 2 で実行した方が効果が高いということが考慮されているためである．その結果，タスク 5 実行中のコア 0 も十分に動作周波数と電圧を下げることができ，全体として消費エネルギー削減効果が高めることができている．

5.1　フェーズ 1：望ましいコア種別と動作周波数の導出

DVFS による消費エネルギー削減効果を向上させるため，提案手法では，HEFT（Heterogeneous Earlist Finish Time）アルゴリズム [34] を用いて仮にタスクスケジューリングを行い，その結果に基づいてクリティカルパス上のタスク（CP タスク）とそれ以外のタスク（非 CP タスク）を分ける．以降，CP タスクと非 CP タスクを分けて取り扱うことにより，クリティカルパスが変化することを防ぐ．

なお，HEFT アルゴリズムはヘテロジニアスなシステム向けのタスクスケジューリング手法である．HEFT アルゴリズムでは，各タスクを各コアで実行する際の実行時間の平均を用いて CP 長を計算し，これをリストスケジューリングの優先度として用いる．各タスクを割り当てる際は，当該タスクを最も早く実行完了するコアを選択する．

本フェーズでは，クリティカルパス上のタスクの実行時間の総和がデッドラインを超えないように，CP タスクに対して動作周波数と割当て先コアの種別を決定する．その後，各非 CP タスクに対して望ましいコア種別と動作周波数を求める．本フェーズにおける具体的な手順を Algorithm 3 および Algorithm 4 に示す．

以上のように，CP タスクに対してコア種別と動作周波数を先に求めることで，以降の処理においてクリティカルパスが変化することを防ぐ．クリティカルパス上のタスクが以降の各フェーズにおいて変化しないことで，CP タスクのスケジューリングおよび動作周波数に大きく依存するスケジューリング長を調整することが容易となる．

なお，非 CP タスクに対しては，Algorithm 4 で求めた望ましいコア種別・望ましい動作周波数になるべく近づけるように，フェーズ 2 で割当てを改めて決定する．しかし，後述のように，フェーズ 2 では CP タスクが優先されるため，非 CP タスクは必ずしもここで求めたコア種別あるいは動作周波数になるとは限らない．そのため，Algorithm 4 では，タスク間の依存関係・実行順とデッドラインのみを考慮しており，ドメイン数やコア数を制約として考慮しない．

5.2　フェーズ 2：タスクスケジューリング

2 番目のフェーズでは，5.1 節の結果に基づき，実際にタスクの割当て先と割当て順を決定する．CP タスクの割当て先コア種別が 1 番目のフェーズで仮設定したものと同じになるように優先することで，クリティカルパスが変化しないようにタスク割当てを行うことができる．

このフェーズにおけるタスク割当ては CP/MISF 法や HEFT アルゴリズム同様，リストスケジューリングによって行われる．各タスクの割当て先コアを決定する際，CP タスクについてはフェーズ 1 で求めたコア種別と同じ種別のコアから最も早い時刻に当該タスクを実行完了するコアに割り当てる．非 CP タスクについては，下記の優先度に従って割当て先コアを決定する．

(1) 当該タスクに仮設定されたコア種別と同じ種別のコア

Algorithm 3 CP タスクのコア種別と動作周波数の導出

Input: $G(N, E)$ :: 入力タスクグラフ
$Dline$ :: スケジューリング長の上限（デッドライン）
Output: $Core[]$ :: 各 CP タスクの割当て先コア種別
$Freq[]$ :: 各 CP タスクを実行する際の動作周波数
$Sched$ :: CP タスクのコア種別・動作周波数変更後のスケジューリング結果

$Sched \leftarrow$ HEFT アルゴリズムによるスケジューリング結果
$SchedLength \leftarrow$ HEFT アルゴリズムによるスケジューリング長
$CPTask[] \leftarrow$ CP タスクのリスト（実行時間の降順にソート）
$N_{CPTask} \leftarrow$ CP タスクの数
for $i = 0$ to N_{CPTask} **do**
  $Core[i] \leftarrow Shced$ において $CPTask[i]$ が割り当てられたコアの種別
  $Freq[i] \leftarrow$ 設定可能な最高動作周波数
end for
$Margin \leftarrow (Dline - SchedLength)$
for $i = 0$ to N_{CPTask} **do**
  while $Core[i]$ が最も低速なコアでない **and** $Margin \geq 0$ **do**
    $Cost_{curr} \leftarrow Core[i]$ における $CPTask[i]$ の実行時間
    $Cost_{slow} \leftarrow Core[i]$ よりも 1 段階低速なコアにおける $CPTask[i]$ の実行時間
    $Diff \leftarrow (Cost_{slow} - Cost_{curr})$
    if $Margin > Diff$ **then**
      $Core[i]$ を 1 段階低速なコアに変更
      $Sched$ における $CPTask[i]$ の終了時刻を $Diff$ 増やす
      $Sched$ において $CPTask[i]$ に依存するすべてのタスクの開始・終了時刻を $Diff$ 増やす
      $Margin \leftarrow (Margin - Diff)$
    end if
  end while
end for
for $i = 0$ to N_{CPTask} **do**
  while $Freq[i]$ が最低の動作周波数でない **and** $Margin \geq 0$ **do**
    $Cost_{curr} \leftarrow Core[i]$ における，動作周波数 $Freq[i]$ での $CPTask[i]$ の実行時間
    $Cost_{slow} \leftarrow Core[i]$ における，$Freq[i]$ よりも 1 段階低い動作周波数での $CPTask[i]$ の実行時間
    $Diff \leftarrow (Cost_{slow} - Cost_{curr})$
    if $Margin > Diff$ **then**
      $Freq[i]$ を 1 段階低い動作周波数に変更
      $Sched$ における $CPTask[i]$ の終了時刻を $Diff$ 増やす
      $Sched$ において $CPTask[i]$ に依存するすべてのタスクの開始・終了時刻を $Diff$ 増やす
      $Margin \leftarrow (Margin - Diff)$
    end if
  end while
end for

から，さらに下記の優先度に従って選択．

(a) 当該タスクと同じ動作周波数が仮設定されたタスクが，同一ドメイン内の他のコアで実行されているもの．
(b) 同じドメイン内の他のコアでタスクが実行されていないもの．
(c) コア種別は一致するが，上記にあてはまらないもの．

Algorithm 4 非 CP タスクに対する望ましいコア種別と動作周波数の導出

Input: $G(N, E)$ :: 入力タスクグラフ
$Sched$ :: Algorithm 3 が出力するスケジューリング結果
$Dline$ :: スケジューリング長の上限（デッドライン）
$nonCPTask[]$ :: 非 CP タスクのリスト（実行時間の降順にソート）
$N_{nonCPTask} \leftarrow$ CP タスクの数
Output: $Core[]$ :: 各非 CP タスクに対する望ましいコア種別
$Freq[]$ :: 各非 CP タスクに対する望ましい動作周波数

repeat
  for $i = 0$ to $N_{nonCPTask}$ **do**
    $Core[i] \leftarrow Sched$ における $nonCPTask[i]$ の割当て先コアの種別
    $Freq[i] \leftarrow Sched$ における $nonCPTask[i]$ の動作周波数
  end for
  for $i = 0$ to $N_{nonCPTask}$ **do**
    現在の $Sched$ に基づいて各非 CP タスクの最早開始時刻・最遅終了時刻を求める
    $Margin[] \leftarrow$ 各非 CP タスクの最遅終了時刻 − 最早開始時刻
    $Energy_{core} \leftarrow Core[i]$ において動作周波数 $Freq[i]$ で $nonCPTask[i]$ を実行した際の消費エネルギー
    if $Core[i]$ が最も低速なコアでない **then**
      $Cost_{core} \leftarrow Core[i]$ よりも 1 段階低速なコアにおいて，動作周波数 $Freq[i]$ で $nonCPTask[i]$ を実行した際の実行時間
      if $Margin[i] \geq Cost_{core}$ **then**
        $Energy_{core} \leftarrow Core[i]$ よりも 1 段階低速なコアにおいて，動作周波数 $Freq[i]$ で $nonCPTask[i]$ を実行した際の消費エネルギー
      end if
    end if
    $Energy_{freq} \leftarrow Core[i]$ において動作周波数 $Freq[i]$ で $nonCPTask[i]$ を実行した際の消費エネルギー
    if $Freq[i]$ が最低の動作周波数でない **then**
      $Cost_{freq} \leftarrow Core[i]$ において，$Freq[i]$ よりも 1 段階低い動作周波数で $nonCPTask[i]$ を実行した際の実行時間
      if $Margin[i] \geq Cost_{freq}$ **then**
        $Energy_{core} \leftarrow Core[i]$ において，$Freq[i]$ よりも 1 段階低い動作周波数で $nonCPTask[i]$ を実行した際の消費エネルギー
      end if
    end if
    if $Energy_{core} < Energy_{freq}$ **then**
      $Core[i] \leftarrow Core[i]$ よりも 1 段階低速なコア種別
    else if $Energy_{core} > Energy_{freq}$ **then**
      $Freq[i] \leftarrow Freq[i]$ よりも 1 段階低い動作周波数
    end if
  end for
until $Core[]$ あるいは $Freq[]$ が更新されない

(2) コア種別にかかわらず，当該タスクを最も早く実行完了するコア．

このようにすることで，CP タスクはフェーズ 1 で求めた種別のコアに割り当てられ，非 CP タスクは CP タスクの割当て先コア選択を阻害しない形で割当て先が決定できる．

5.3 フェーズ 3：DVFS の適用

最後に，割当て先・割当て順が決定された各タスクの動

作周波数を決定する．5.2 節で述べた 2 番目のフェーズ同様，与えられたデッドラインを考慮しつつ CP タスクの動作周波数を先に決定し，その後，非 CP タスクの動作周波数を決定する．動作周波数の決定は Wang らの手法 [19] を参考に，対象モデルに適用可能となるよう変更を加えたものであるが，上記の 2 つのフェーズを経ることで，粗粒度な電圧ドメインを持つヘテロジニアスメニーコアにおいても有効なものとなっている．本フェーズの具体的手順を Algorithm 5 に示す．Algorithm 5 により，各タスクの割当て先および動作周波数が最終的に決定される．

Algorithm 5 では，CP タスクの動作周波数をフェーズ 1 で求めた動作周波数とすることでスケジューリング長がデッドラインを超えることを防ぐ．また，非 CP タスクについては，CP タスクに影響しない範囲で，フェーズ 1 で求めた望ましい動作周波数に近づける．

Algorithm 5 DVFS の適用

Input: $G(N, E)$ :: 入力タスクグラフ
$Sched$ :: スケジューリング結果
$CPTask[]$ :: CP タスクのリスト（実行時間の降順にソート）
N_{CPTask} :: CP タスクの数
$nonCPTask[]$ :: 非 CP タスクのリスト（実行時間の降順にソート）
$N_{nonCPTask}$ :: 非 CP タスクの数
$Core[]$ :: 各タスクの望ましいコア種別
$Freq[]$ :: 各タスクの望ましい動作周波数
Output: $Sched$ :: スケジューリング結果

for $i = 0$ to N_{CPTask} **do**
  $CPTask[i]$ の動作周波数を $Freq[i]$ に設定する
  $CPTask[i]$ の終了時刻を動作周波数 $Freq[i]$ にあわせて更新する
  実行順序・依存関係を満たすように，$CPTask[i]$ に依存するすべてのタスクの開始・終了時刻を更新する
  if スケジューリング長がデッドラインに到達 **then**
    break
  end if
end for
for $f =$ 最高動作周波数 to 最低動作周波数 **do**
  for $i = 0$ to $N_{nonCPTask}$ **do**
    if $Freq[i] > f$ **then**
      現在のスケジューリング結果に基づいて $nonCPTask[i]$ の最早開始時刻・最遅終了時刻を求める
      $Margin \leftarrow nonCPTask[i]$ の最遅終了時刻 − 最早開始時刻
      $Diff \leftarrow Freq[nonCPTask[i]]$ を f に変更した際の実行時間の増分
      if $Margin \geq Diff$ **then**
        $Freq[nonCPTask[i]] \leftarrow f$
        $Sched$ における $nonCPTask[i]$ の終了時刻を $Diff$ 増やす
        実行順序・先行制約を満たすように各非 CP タスクの最早開始時刻・最遅終了時刻を更新する
      end if
    end if
  end for
end for

6. 性能評価

本章では，4 章および 5 章で述べた提案タスクスケジューリング手法による消費エネルギー削減効果に関する評価およびその結果について述べる．本論文では，並列アプリケーションに相当する入力として標準タスクグラフセット [14] および実アプリケーションを想定したタスクグラフを用い，提案手法の消費エネルギー削減効果について評価を行った．

6.1 評価環境および条件

本評価では，入力タスクグラフとして，標準タスクグラフセット [14] からタスク数 2,000（入口ノード・出口ノードを含めた場合，タスク数 2,002）の 180 例（各グラフの並列度の平均：30.75）を用いた．標準タスクグラフセットでは，4 種類のグラフ生成法を用いてランダムに生成したタスクグラフが用いられている．

また，実アプリケーションを想定した評価では，ラプラス方程式求解，LU 分解，ステンシル演算の並列アルゴリズムを想定したタスクグラフ [35], [36] を作成し，これらを用いた．これらのアプリケーションを構成するタスクのコストは入力データや配列サイズに依存するが，通常はタスクの粒度と並列性を考慮し，同期や DVFS，PG による遷移オーバヘッドが無視できる程度に大きいサイズに保たれる．この実アプリケーションを想定した評価では，入力となるタスクグラフの最大の並列度が対象メニーコアのコア数と同数となるように設定した．

各タスクグラフに対するデッドラインとしては，DVFS 適用前のスケジューリング長に対して 100[%]，120[%]，140[%]，160[%]，180[%]，200[%] のスケジューリング長を想定した．

各コアに対して設定可能な動作周波数および動作周波数と対応する電圧の状態は FULL，HIGH，MID，LOW および PG 適用時を示す OFF の 5 種類とした．それぞれの状態における相対的な動作周波数および電圧を**表 1** に示す．各コアが動作している際のリーク電力は，通常動作時（FULL）においてダイナミック電力の 20[%] であり，電圧の 1.5 乗にほぼ比例すると仮定した [32]．

表 1　各動作状態における周波数・電圧（相対値）
Table 1　Voltage and operating frequency of each DVFS state (normalized to FULL state).

状態名	周波数	電圧	動的電力	静的電力
FULL	1.00	1.00	1.00	0.200
HIGH	0.67	0.92	0.57	0.169
MID	0.50	0.85	0.36	0.143
LOW	0.25	0.70	0.12	0.121
OFF	0.00	0.00	0.00	0.000

表 2　ヘテロジニアスメニーコアにおけるコア種別と動作周波数 FULL 時のスペック
Table 2　Core spec on the supposed heterogeneous manycore (with highest operating frequency).

コア種別	シンプル	高速
電圧制御ドメイン内のコア数	4	1
タスク実行時の性能（高速なコアに対する相対値）	0.50	1.00
動的消費電力（高速なコアに対する相対値）	0.25	1.00
静的消費電力（高速なコアの動的電力に対する相対値）	0.05	0.20

本論文ではメニーコア内の電圧制御ドメイン数が 4 および 8 の場合について，3 章で述べたモデルに基づいたシミュレーション評価を行った．この際，ホモジニアスメニーコアにおいては各電圧制御ドメインに 4 コアが含まれるものとした．ヘテロジニアスメニーコアの場合は，半数のドメインがシンプルなコアを 4 コア，もう半数のドメインが高速なコアを 1 コア含むものとした．シンプルなコアと高速なコアの通常動作時（FULL）における電力効率およびタスク実行時の性能については，**表 2** に示すとおりである．入力タスクグラフによって与えられる各タスクの実行時間は，高速なコアが最も高い動作周波数で動作している場合のものとして取り扱う．コアの種別あるいは動作周波数が異なる場合，表 2 に示す性能比および動作周波数に応じてタスクの実行時間が変化するものとする．

本性能評価における比較対象としては，ホモジニアスメニーコアにおいては

- CP/MISF アルゴリズム（PG・DVFS を適用せず，IDLE のコアはリーク電力のみ消費すると仮定）
- CP/MISF アルゴリズムに PG のみを適用
- CP/MISF アルゴリズムに PG および DVFS を適用（ドメイン単位での DVFS を仮定）
- CP/MISF アルゴリズムに PG および DVFS を適用（コア単位の DVFS を仮定）
- 提案手法（ドメイン単位での DVFS を仮定）

の 5 種類の手法・環境を比較し，ヘテロジニアスメニーコアにおいては

- HEFT アルゴリズム（PG・DVFS を適用せず，IDLE のコアはリーク電力のみ消費すると仮定）
- HEFT アルゴリズムに PG のみを適用
- HEFT アルゴリズムに PG および DVFS を適用（ドメイン単位での DVFS を仮定）
- HEFT アルゴリズムに PG および DVFS を適用（コア単位の DVFS を仮定）
- 提案手法（ドメイン単位での DVFS を仮定）

の 5 種類の手法・環境を比較した．「CP/MISF アルゴリズムに PG および DVFS を適用（コア単位の DVFS を仮定）」および「HEFT アルゴリズムに PG および DVFS を適用（コア単位の DVFS を仮定）」はコアごとに周波数制御・電圧制御が可能な環境を想定したものである．提案手法が対象とする粗粒度な電圧ドメインを持つ環境とは異なるが，本評価では，従来手法に対する理想的な条件，つまり最大限消費エネルギー削減可能となる場合として比較対象に加えた．

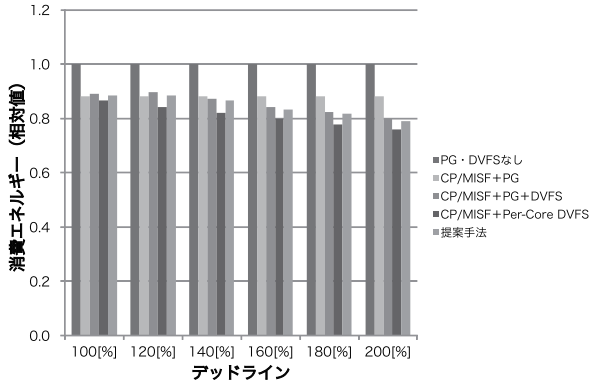

図 8　ホモジニアスメニーコア（16 コア，4 ドメイン）に対する評価結果
Fig. 8　Evaluation results for 16 core homogeneous manycore.

なお，提案手法はチップ上のコア部分による消費エネルギーの削減を目的としているため，バスやメモリ等の消費電力・消費エネルギーについては評価に含めていない．

6.2　評価結果

ここでは，ホモジニアスメニーコアおよびヘテロジニアスメニーコア上での提案手法の評価結果を示し，これらに関して議論する．

6.2.1　ホモジニアスメニーコアに対する評価

図 8 に 16 コアのホモジニアスメニーコアにおける評価結果を，**図 9** に 32 コアのホモジニアスメニーコアにおける評価結果を示す．これらの図において，横軸は与えられたデッドライン，縦軸は各手法を適用した場合の各タスクグラフにおける消費エネルギーの相乗平均であり，PG・DVFS を適用しない場合のもの（PG・DVFS なし）に正規化されている．

ホモジニアス環境においてはすべてのコアの特性が同等であるため，コア単位での DVFS が可能な環境を仮定した場合（CP/MISF+Per-Core DVFS）に最も DVFS の効果が高く，評価結果からもそのことが分かる．提案手法が対象とする粗粒度な電圧制御ドメインを持つメニーコアの環境においては，“CP/MISF+Per-Core DVFS” には及ばないものの，CP/MISF 法によるスケジューリング後に DVFS と PG を適用した場合（CP/MISF+PG+DVFS）と比較して，提案手法は平均して 2[%] 程度消費エネルギー削減量を改善できている．

また，16 コアの場合と 32 コアの場合を比較すると，全

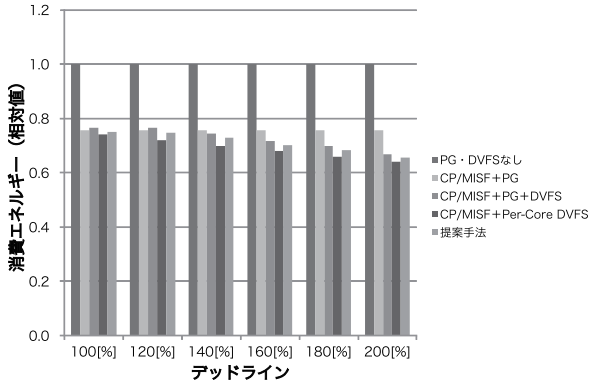

図 9　ホモジニアスメニーコア（32 コア，8 ドメイン）に対する評価結果

Fig. 9　Evaluation results for 32 core homogeneous manycore.

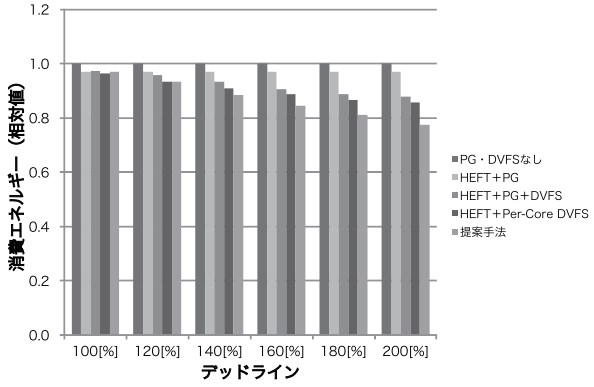

図 10　ヘテロジニアスメニーコア（10 コア，4 ドメイン）に対する評価結果

Fig. 10　Evaluation results for 10 core heterogeneous manycore.

体として 32 コアの方が各手法によるエネルギー削減率が大きくなっていることが分かる．これは，16 コアの場合は入力タスクグラフの並列度と比較してコア数が小さいため，コアが IDLE となる機会，つまり PG が適用される機会が少ないためである．

なお，DVFS を用いずに PG のみを適用した場合（CP/MISF+PG）では，16 コアと 32 コアで消費エネルギー量は同じであった．これは，DVFS を適用しないためタスク実行に必要な消費エネルギーはコア数を変更しても変わらず，使用しないコアは PG を適用することで電力を消費しないためである．このことから，PG が適切に利用可能であれば，よりコア数の大きい環境を提供することにより，消費電力を無駄にすることなく，様々な並列度のアプリケーションに対応可能であるといえる．

6.2.2　ヘテロジニアスメニーコアに対する評価

図 10 に 10 コアのヘテロジニアスメニーコアにおける評価結果を，**図 11** に 20 コアのヘテロジニアスメニーコアにおける評価結果を示す．前節同様，横軸は与えられたデッドライン，縦軸は各手法を適用した場合の各タスクグラフにおける消費エネルギーの相乗平均であり，PG・DVFS ともに適用しない場合（PG・DVFS なし）に正規化されている．

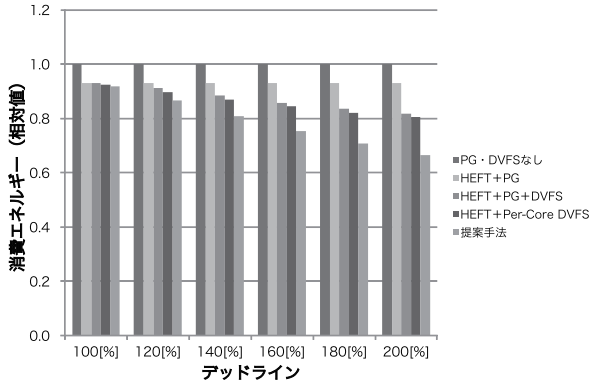

図 11　ヘテロジニアスメニーコア（20 コア，8 ドメイン）に対する評価結果

Fig. 11　Evaluation results for 20 core heterogeneous manycore.

図 10 を見ると，HEFT アルゴリズムの結果に対して PG のみを適用した場合（HEFT+PG）による消費エネルギー削減率が小さいことが分かる．これは，ホモジニアスメニーコアの場合と同様，入力タスクグラフの並列度と比較してコア数が少ないため，PG を適用可能な機会，つまりコアが IDLE となる期間が少ないためである．また，デッドラインが短い場合には，スケジューリング長を大きく延ばせないため，DVFS をあわせて適用した場合（HEFT+PG+DVFS）の効果も小さい．この問題は，図 11 に示すとおり，コア数を増加させ，タスクグラフの並列度に対して十分な数のコアを用いることで，DVFS・PG ともに適用機会が増え改善される．

それに対して提案手法では，デッドラインが短い場合においても，コア単位での DVFS が可能な環境を想定した場合（HEFT+Per-Core DVFS）と比較してより高いエネルギー削減効果を得ることができている．これは，ヘテロジニアスメニーコアにおいては，PG および DVFS のほか，タスクを実行するコアを適切に選択することによる消費エネルギーの削減が可能であるためである．たとえば，20 コアを用いた図 11 を見ると，実行時間を延長しない場合（デッドライン 100[%]）であっても，提案手法が最もエネルギー削減効果が高いことが分かる．HEFT アルゴリズムでは，タスクの割当て先を決定する際に当該タスクが最も早く実行を終了できるコアを選択するため，高速なコアを割当て先に選択する可能性が高い．これに対して，提案手法は，スケジューリング長がデッドラインを上回らない範囲で，低速であっても電力効率の高いシンプルなコアを割当て先として選択する．PG・DVFS に加え，適切にコア種別を選択することによっても，消費エネルギー削減を実現している．HEFT アルゴリズムによるスケジューリング結果をコア単位での DVFS が可能な環境に適用した場合（HEFT+Per-Core DVFS）と比較して，デッドライン 120[%] の場合で約 4[%]，140[%] の場合で約 7[%]，160[%] の場合で約 11[%]，180[%] の場合で約 14[%]，200[%] の場合で

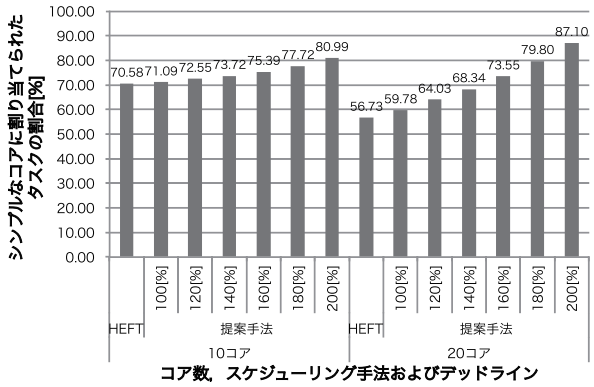

図 12　シンプルなコアに割り当てられたタスクの割合

Fig. 12　The propotion of tasks assigned to simple cores.

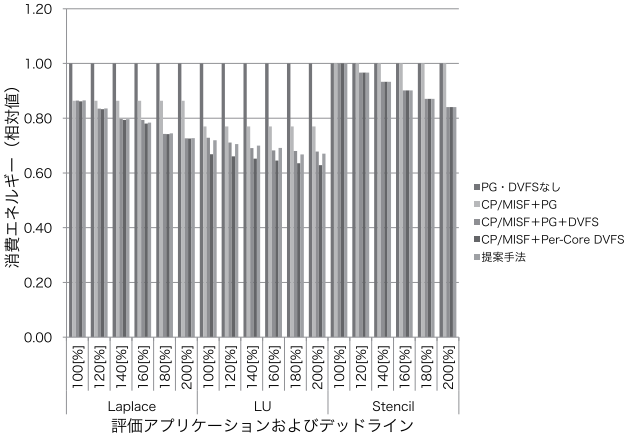

図 13　実アプリケーションを想定したホモジニアスメニーコア（16 コア，4 ドメイン）に対する評価結果

Fig. 13　Evaluation results for homogeneous manycores (16 cores, 4 domains) with real applications.

約 17[%] 程度，提案手法の方がエネルギー削減効果が高い．

また，**図 12** に，HEFT アルゴリズムおよびヘテロジニアスメニーコア向けの提案手法において，シンプルなコアに割り当てられたタスクの割合を示す．HEFT アルゴリズムと比較して，提案手法では，より多数のコアがシンプルなコア，つまり電力効率の高いコアに割り当てられていることが分かる．PG，DVFS とともに割当て先コアの変更によるエネルギー削減が効果的であるといえる．

提案手法により，スケジューリング長が与えられたデッドラインを超えないという制約のもと，適切なコアを利用しつつ DVFS を効果的に用いることで，粗粒度な電圧制御ドメインを持つメニーコアにおいても高いエネルギー削減効果を得ることができた．

6.2.3　実アプリケーションを想定した評価

実アプリケーションを想定したタスクグラフ（ラプラス方程式求解，LU 分解，ステンシル演算）を用いた評価結果について，ホモジニアスメニーコアを対象とした結果を**図 13** および**図 14** に，ヘテロジニアスメニーコアを対象とした結果を**図 15** および**図 16** に示す．これらの各図に

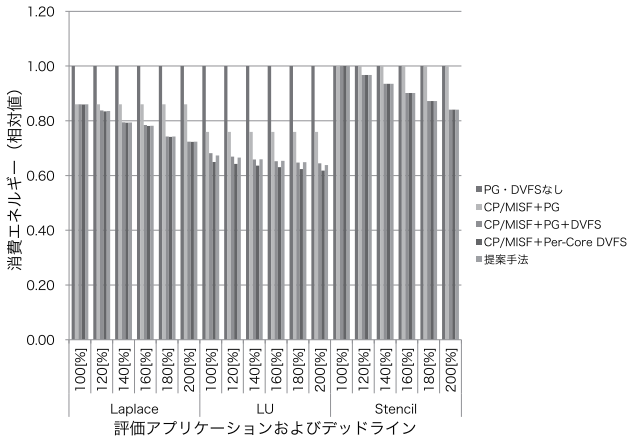

図 14　実アプリケーションを想定したホモジニアスメニーコア（32 コア，8 ドメイン）に対する評価結果

Fig. 14　Evaluation results for homogeneous manycores (32 cores, 8 domains) with real applications.

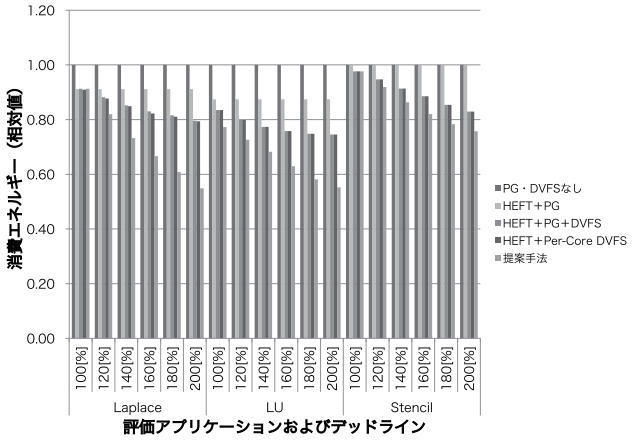

図 15　実アプリケーションを想定したヘテロジニアスメニーコア（10 コア，4 ドメイン）に対する評価結果

Fig. 15　Evaluation results for heterogeneous manycores (10 cores, 4 domains) with real applications.

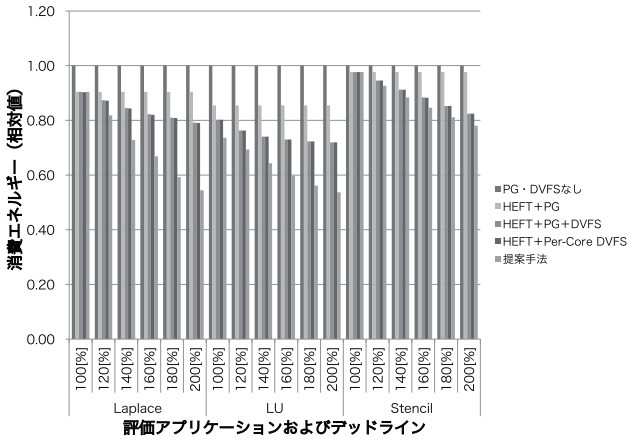

図 16　実アプリケーションを想定したヘテロジニアスメニーコア（20 コア，8 ドメイン）に対する評価結果

Fig. 16　Evaluation results for heterogeneous manycores (20 cores, 8 domains) with real applications.

おいて，横軸は評価アプリケーションおよびデッドライン，縦軸は各手法を適用した場合の各タスクグラフにおける消費エネルギーであり，それぞれのアプリケーションごとに，PG・DVFS ともに適用しない場合（PG・DVFS なし）に正規化されている．

実際の並列アプリケーションでは，並列に実行可能なタスクの粒度が均一になるようにアルゴリズムが工夫されている．そのため，ホモジニアスメニーコア上では，従来手法と提案手法の間で消費エネルギー削減率に大きな差は見られない．特に，ラプラス方程式求解およびステンシル演算では，各タスクのコストがほぼ一定であり，並列なタスクはほぼ同時に実行可能となる．そのため，DVFS を適用可能な空き時間が少なく，いずれの手法においても，消費エネルギーの削減効果はほぼ同一であった．LU 分解では，他の 2 つのアプリケーションとは異なり，クリティカルパス上にないタスクに対しては DVFS を適用する余裕度がある．そのため，コア単位での DVFS が可能な環境におけるエネルギー削減効果が若干高くなっている．

ヘテロジニアスメニーコア上では，高速なコアとシンプルなコアを併用することにより，並列性能を保ちつつ DVFS を適用する機会を増やすことができる．しかしながら，実アプリケーションを想定したタスクグラフを用いた場合，HEFT アルゴリズムでは各ドメイン内のコアの挙動がほぼ同じになるため，コア単位での DVFS が可能な環境を想定した場合と，粗粒度な電圧ドメインを想定した場合を比較しても，ほぼ同様の消費エネルギー削減量となる．一方，ヘテロジニアスメニーコア向けの提案手法では，使用するコアの変更と DVFS の適用による効果を比較し，より消費エネルギー削減効果の高い方法をとっている．そのため，高速なコアを利用することで生じる DVFS 適用機会の増加と，シンプルなコアを利用することによる消費電力削減効果の恩恵を同時に受けることができる．その結果，20 コアのヘテロジニアスメニーコアにおいて，“HEFT+Per-Core DVFS” と比較して，デッドライン 120[%] の場合で約 6[%]，140[%] の場合で約 14[%]，160[%] の場合で約 19[%]，180[%] の場合で約 27[%]，200[%] の場合で約 31[%] 程度，ラプラス方程式求解において提案手法の方が高いエネルギー削減効果を示した．

7. まとめ

本論文では，チップ上のコアの大きさと比較して DVFS における電圧制御ドメインの粒度が大きいメニーコアプロセッサを対象として，DVFS による消費エネルギー削減効果を向上させるタスクスケジューリング手法を提案した．提案手法では，あらかじめ各タスク実行時の動作周波数を予測し，電圧制御ドメイン内コアの動作周波数を均一に保つようタスクを割り当てることで，DVFS の効果を向上させる．また，ヘテロジニアスメニーコアでは，スケジューリング長がデッドラインを超えない範囲で割当て先のコアと動作周波数を選択することで，エネルギー削減効果を向上させる．提案手法を用いることで，チップ上のすべてのコアが均一であるホモジニアスメニーコア，異なる性能のコアを集積するヘテロジニアスメニーコアの両方において，従来手法を粗粒度な電圧ドメインを持つメニーコアに適用した場合と比較して，消費エネルギー削減量を向上させることができた．特に，ヘテロジニアスメニーコアにおいては，電力効率の高いコアを適切に選択することにより，コア単位での電圧制御が可能な条件において従来手法を適用した場合と比較しても，高い消費エネルギー削減量を達成した．

今後の課題として，コア間のデータ転送を考慮した手法や，チップ内外のネットワーク・メモリ等の要素を考慮した低消費電力化手法の開発，および自動並列化コンパイラや並列化支援ツールへの組込み等があげられる．

謝辞　本研究の一部は JSPS 科研費 24700055 の助成を受け行われた．

参考文献

[1] Kumar, R., Farkas, K.I., Jouppi, N.P., Ranganathan, P. and Tullsen, D.M.: Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction, *Proc. 36th International Symposium on Microarchitecture*, pp.81–92 (2003).

[2] Jeff, B.: *Advances in big.LITTLE Technology for Power and Energy Savings*, ARM Limited (2012).

[3] Kasahara, H. and Narita, S.: Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing, *IEEE Trans. Comput.*, Vol.C-33, No.11, pp.1023–1029 (1984).

[4] Kim, W., Gupta, M.S., Wei, G.-Y. and Brooks, D.: System Level Analysis of Fast, Per-Core DVFS Using On-Chip Switching Regulators, *IEEE 14th International Symposium on High Performance Computer Architecture*, pp.123–134 (2008).

[5] Jain, R., Geuskens, B., Khellah, M., Kim, S., Kulkarni, J., Tschanz, J. and De, V.: A 0.45-1V Fully Integrated Reconfigurable Switched Capacitor Step-Down DC-DC Converter with High Density MIM Capacitor in 22nm Tri-Gate CMOS, *2013 Symposium on VLSI Circuits Digest of Technical Papers*, pp.C174–C175 (2013).

[6] Sturcken, N., O'Sullivan, E.J., Wang, N., Herget, P., Webb, B.C., Romankiw, L.T., Petracca, M., Davies, R., Fontana, R.E., Decad, G.M., Kymissis, I., Peterchev, A.V., Carloni, L.P., Gallagher, W.J. and Shepard, K.L.: A 2.5D Integrated Voltage Regulator Using Coupled-Magnetic-Core Inductors on Silicon Interposer, *IEEE Journal of Solid-State Circuits*, Vol.48, No.1, pp.244–254 (2013).

[7] Kurd, N., Chowdhury, M., Burton, E., Thomas, T.P., Mozak, C., Boswell, B., Lal, M., Deval, A., Douglas, J., Elassal, M., Nalamalpu, A., Wilson, T.M., Merten, M., Chennupaty, S., Gomes, W. and Kumar, R.: Haswell: A Family of IA 22nm Processors, *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp.112–113 (2014).

[8] Qualcomm, Inc.: *Snapdragon S4 Processors: System on*

Chip Solutions for a New Mobile Age (2011).

[9] Bell, S., Edwards, B., Amann, J., Conlin, R., Joyce, K., Leung, V., MacKay, J., Reif, M., Bao, L., Brown, J., Mattina, M., Miao, C.-C., Ramey, C., Wentzlaff, D., Anderson, W., Berger, E., Fairbanks, N., Khan, D., Montenegro, F., Stickney, J. and Zook, J.: TILE64™ Processor: A 64-Core SoC with Mesh Interconnect, *2008 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp.88–598 (2008).

[10] Howard, J., Dighe, S., Hoskote, Y., Vangal, S., Finan, D., Ruhl, G., Jenkins, D., Wilson, H., Borkar, N., Schrom, G., Pailet, F., Jain, S., Jacob, T., Yada, S., Marella, S., Salihundam, P., Erraguntla, V., Konow, M., Riepen, M., Droege, G., Lindemann, J., Gries, M., Apel, T., Henriss, K., Lund-Larsen, T., Steibl, S., Borkar, S., De, V., Wijngaart, R.V.D. and Mattson, T.: A 48-Core IA-32 Message-Passing Processor with DVFS in 45nm CMOS, *2010 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp.108–109 (2010).

[11] Wang, P.-H., Yang, C.-L., Chen, Y.-M. and Cheng, Y.-J.: Power Gating Strategies on GPUs, *ACM Trans. Architecture and Code Optimization*, Vol.8, No.3 (2011).

[12] Kim, W., Brooks, D.M. and Wei, G.-Y.: A Fully-Integrated 3-Level DC/DC Converter for Nanosecond-Scale DVS with Fast Shunt Regulation, *2011 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp.268–270 (2011).

[13] Ghasemi, H.R., Sinkar, A.A., Schulte, M.J. and Kim, N.S.: Cost-Effective Power Delivery to Support Per-Core Voltage Domains for Power-Constrained Processors, *Proc. 49th Annual Design Automation Conference*, pp.56–61 (2012).

[14] Standard Task Graph Set Version 2, available from ⟨http://www.kasahara.cs.waseda.ac.jp/schedule/⟩.

[15] Zhuravlev, S., Saez, J.C., Blagodurov, S., Fedorova, A. and Prieto, M.: Survey of Energy-Cognizant Scheduling Techniques, *IEEE Trans. Parallel and Distributed Systems*, Vol.24, No.7, pp.1447–1464 (2013).

[16] Seo, E., Jeong, J., Park, S. and Lee, J.: Energy Efficient Scheduling of Real-Time Tasks on Multicore Processors, *IEEE Trans. Parallel and Distributed Systems*, Vol.19, No.11, pp.1540–1552 (2008).

[17] Yang, C.-Y., Chen, J.-J. and Kuo, T.-W.: An Approximation Algorithm for Energy-Efficient Scheduling on A Chip Multiprocessor, *Proc. Design, Automation and Test in Europe Conference and Exhibition*, pp.468–473 (2005).

[18] Kimura, H., Sato, M., Hotta, Y., Boku, T. and Takahashi, D.: Emprical Study on Reducing Energy of Parallel Programs Using Slack Reclamation by DVFS in a Power-Scalable High Performance Cluster, *Proc. 2006 IEEE International Conference on Cluster Computing*, pp.1–10 (2006).

[19] Wang, L., von Laszewski, G., Dayal, J. and Wang, F.: Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS, *Proc. 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp.368–377 (2010).

[20] Wang, L., Tao, J., von Laszewski, G. and Chen, D.: Power Aware Scheduling for Parallel Tasks via Task Clustering, *Proc. 16th International Conference on Parallel and Distributed Systems*, pp.629–634 (2010).

[21] Zong, Z., Manzanares, A., Stinar, B. and Qin, X.: Energy-Aware Duplication Strategies for Scheduling Precedence-Constrained Parallel Tasks on Clusters, *Proc. 2006 IEEE International Conference on Cluster Computing*, pp.1–8 (2006).

[22] Mei, J. and Li, K.: Energy-Aware Scheduling Algorithm with Duplication on Heterogenous Computing Systems, *Proc. ACM/IEEE 13th International Conference on Grid Computing*, pp.122–129 (2012).

[23] Lee, Y.C. and Zomaya, A.Y.: Energy Conscious Scheduling for Distributed Computing Systems under Different Operating Conditions, *IEEE Trans. Parallel and Distributed Systems*, Vol.22, No.8, pp.1374–1381 (2011).

[24] Sun, W. and Sugawara, T.: Heuristics and Evaluations of Energy-Aware Task Mapping on Heterogeneous Multiprocessors, *Proc. 2011 IEEE International Parallel & Distributed Processing Symposium*, pp.599–607 (2011).

[25] Li, D. and Wu, J.: Energy-Aware Scheduling for Frame-Based Tasks on Heterogeneous Multiprocessor Platforms, *Proc. 2012 41st International Conference on Parallel Processing*, pp.430–439 (2012).

[26] 白子　準，吉田宗弘，押山直人，和田康孝，中野啓史，鹿野裕明，木村啓二，笠原博徳：マルチコアプロセッサにおけるコンパイラ制御低消費電力化手法，情報処理学会論文誌コンピューティングシステム，Vol.47, No.SIG12(ACS15), pp.147–158 (2006).

[27] 林　明宏，和田康孝，渡辺岳志，関口　威，間瀬正啓，白子準，木村啓二，笠原博徳：ヘテロジニアスマルチコア向けソフトウェア開発フレームワークおよび API，情報処理学会論文誌コンピューティングシステム，Vol.5, No.1, pp.68–79 (2012).

[28] Li, D., de Supinski, B.R., Schulz, M., Cameron, K. and Nikolopoulos, D.S.: Hybrid MPI/OpenMP Power-Aware Computing, *Proc. 2010 IEEE International Symposium on Parallel & Distributed Processing*, pp.1–12 (2010).

[29] 間瀬正啓，中川　亮，大國直人，白子　準，木村啓二，笠原博徳：マルチコア上での OSCAR API を用いた並列化コンパイラによる低消費電力化手法，情報処理学会論文誌コンピューティングシステム，Vol.2, No.3, pp.96–106 (2009).

[30] Zhao, L., Ikebuchi, D., Saito, Y., Kamata, M., Seki, N., Kojima, Y., Amano, H., Koyama, S., Hashida, T., Umahashi, Y., Masuda, D., Usami, K., Kimura, K., Namiki, M., Takeda, S., Nakamura, H. and Kondo, M.: Geyser-2: The Second Prototype CPU with fine-Grained run-Time Power Gating, *Proc. 16th Asia and South Pacific Design Automation Conference*, pp.87–88 (2011).

[31] Hillenbrand, D., Furuyama, Y., Hayashi, A., Mikami, H., Kimura, K. and Kasahara, H.: Reconciling Application Power Control and Operating Systems for Optimal Power and Performance, *Proc. 8th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip*, pp.1–8 (2013).

[32] 水野弘之：DVS/DVFS 技術による低電力化の今後，電子情報通信学会技術研究報告，SDM，シリコン材料・デバイス，Vol.107, No.194, pp.41–46 (2007).

[33] Greenhalgh, P.: *Big.LITTLE Processing with ARM Cortex™-A15 & Cortex-A7*, ARM Limited (2011).

[34] Topcuoglu, H., Hariri, S. and Wu, M.-Y.: Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing, *IEEE Trans. Parallel and Distributed Systems*, Vol.13, No.3, pp.260–274 (2002).

[35] Boudet, V.: Heterogeneous Task Scheduling: A Survey, Technical Report LIP-RR-2001-07, ENS-Lyon (2001).

[36] Mohamed, M.R. and Awadalla, M.H.A.: Hybrid Algorithm for Multiprocessor Task Scheduling, *IJCSI International Journal of Computer Science Issues*, Vol.8, No.2, pp.79–89 (2011).

和田 康孝　（正会員）

1979 年生．2002 年早稲田大学理工学部電気電子情報工学科卒業．2004 年同大学大学院理工学研究科電気工学専攻修士課程修了．2007 年同大学院理工学研究科情報・ネットワーク専攻博士課程満期退学．2009 年博士（工学）早稲田大学．2006 年早稲田大学理工学部助手．2007 年同大学理工学術院基幹理工学部助手．2009 年同大学 IT 研究機構次席研究員．2010 年同大学理工学術院基幹理工学研究科助教，エジプト日本科学技術大学特任准教授．2012 年電気通信大学大学院情報システム学研究科助教．2014 年より早稲田大学理工学術院基幹理工学研究科助教，現在に至る．自動並列化技術，メニーコアプロセッサアーキテクチャ，ヘテロジニアスコンピューティング等に関する研究に従事．電子情報通信学会，IEEE-CS，ACM 各会員．

近藤 正章　（正会員）

1998 年筑波大学第三学群情報学類卒業．2000 年同大学大学院工学研究科博士前期課程修了．2003 年東京大学大学院工学系研究科先端学際工学専攻修了．博士（工学）．独立行政法人科学技術振興機構戦略的創造研究推進事業 CREST 研究員，2004 年東京大学先端科学技術研究センター特任助手，2007 年同特任准教授，2008 年電気通信大学大学院情報システム学研究科准教授を経て，現在，東京大学大学院情報理工学系研究科准教授．計算機アーキテクチャ，ハイパフォーマンスコンピューティング，ディペンダブルコンピューティングの研究に従事．電子情報通信学会，IEEE，ACM 各会員．

本多 弘樹　（正会員）

1984 年早稲田大学理工学部電気工学科卒業．1991 年同大学大学院理工学研究科博士課程修了．1987 年より同大学情報科学研究教育センター助手．1991 年より山梨大学工学部電子情報工学科専任講師．1992 年より同助教授．1997 年より電気通信大学大学院情報システム学研究科助教授．2007 年より同教授．並列処理方式，並列化コンパイラ，並列コンピュータアーキテクチャ，GPU コンピューティング等の研究に従事．工学博士．電子情報通信学会，IEEE-CS，ACM 各会員．平成 15 年度情報処理学会山下記念研究賞受賞．