平成28年度　修士論文

ヘテロジニアス・プロセッサの設計探索手法に関する研究

電気通信大学 大学院情報システム学研究科
情報システム基盤学専攻

1553007　澁谷 俊憲

指導教員
本多 弘樹 教授
三輪 忍 准教授
新谷 隆彦 准教授

平成29年1月26日

平成28年度　修士論文

ヘテロジニアス・プロセッサの設計探索手法に関する研究

電気通信大学 大学院情報システム学研究科
情報システム基盤学専攻

1553007　澁谷 俊憲

指導教員
本多 弘樹 教授
三輪 忍 准教授
新谷 隆彦 准教授

平成29年1月26日

目 次

第1章

序論

1.1 研究背景

低消費エネルギーなプロセッサとしてヘテロジニアス・プロセッサが注目を集めている [1–15]．ヘテロジニアス・プロセッサは，1つのチップ上に電力効率の異なる複数のコアやアクセラレータを搭載したプロセッサである．ヘテロジニアス・プロセッサは，実行するプログラム中のフェーズごとに，そのフェーズを最も高い電力効率で実行可能なコアをチップ内から選出し，そのコアで当該フェーズを実行することにより，消費電力を削減する．フェーズはプログラムの瞬間的な特性により区切られた，プログラム中の区間を示す．

ヘテロジニアス・プロセッサの消費電力削減効果は，チップに搭載したコアの組み合わせに依存する．そのため，ヘテロジニアス・プロセッサの設計においては，どのような特性のコアをチップ上に搭載するかを決定することが重要な課題である．例えば，コアを汎用的なものとするか，何らかの処理性能に特化したアクセラレータとするか，あるいは，実行方式をアウト・オブ・オーダ方式にするか，インオーダ方式にするか，さらには，命令発行幅，メモリ，キャッシュサイズなどのパラメータをいくつにするか，などを決定する必要がある．このような問題はプロセッサ・アーキテクチャの設計探索問題と呼ばれている．設計探索問題は，1つのチップ上に同一のコアを複数搭載したプロセッサであるホモジニアス・プロセッサにおいては多くの研究事例があるものの，ヘテロジニアス・プロセッサの設計探索問題についてはまだ研究が少ない．

ヘテロジニアス・プロセッサの設計探索は，ホモジニアス・プロセッサと比べて，探索空間が膨大となり，それにより，探索時間も長くなる．例えば1つのコア

が取りうるパラメータの数，つまり，候補となるコアの数を N，ヘテロジニアス・プロセッサに搭載するコアの数を k とする．ホモジニアス・プロセッサの場合，N 通りの候補コアの中から最適なコアを 1 つ選出し，それを複数搭載するため，計算時間は $O(N)$ となる．これに対して，ヘテロジニアス・プロセッサの場合，N 通りの候補コアの中から k 個のコアの最適な組み合わせを選出するため，探索点の数は ${}_NC_k$，すなわち $O(N^k)$ となる．設計探索に要する時間は，探索点の数と 1 つの探索点の評価に要する時間の積で与えられるため，ヘテロジニアス・プロセッサの設計探索に要する時間も $O(N^k)$ となってしまうことが問題である．

1.2　研究目的

前述の問題を踏まえ，本研究は，高速かつ高精度なヘテロジニアス・プロセッサの設計探索を行う手法を実現することを目的とする．

k 個のコアを搭載したヘテロジニアス・プロセッサを設計する場合，通常は，まず k 個のコアの組み合わせを決定し，その組み合わせに対してプログラムを実行した時の性能や消費エネルギーなどをシミュレーション等により評価する．

上述したように，コアの組み合わせの総数が $O(N^k)$ であることから，この組み合わせすべてに対してシミュレーションを行うとすると，シミュレーション回数は通常 $O(N^k)$ となる．それに対し，本研究の目指す手法は，最初に，プログラム内のすべてのフェーズをある 1 つのコアで実行した時の，フェーズごとの性能と消費エネルギーをシミュレーション等により求めた結果を用いるものである．コアの種類は N であるため，この計算に必要なシミュレーション回数は $O(N)$ である．そして，得られたコアごと，およびフェーズごとの性能と消費エネルギーのトレースから，最適な k 個のコアの組み合わせをヒューリスティックな探索手法によって求める．

1.3　本論文の構成

以下に本論文の構成を述べる．

第 2 章

本研究が対象とするヘテロジニアス・プロセッサについて，代表的な研究を紹介する．続いて，プロセッサの設計探索についての関連研究を紹介する．

第3章

本研究で手法を実現するにあたり行った予備実験について述べ，結果を示す．

第4章

本研究における提案手法について詳細を述べる．

第5章

提案手法を用いて実際に設計探索を行い，その性能を評価する．

第6章

本研究のまとめと及び今後の課題について述べる．

第2章

ヘテロジニアス・プロセッサとその設計探索

本章では，本研究が対象とするヘテロジニアス・プロセッサ，並びに，プロセッサの設計探索に関する研究について紹介する．

2.1 ヘテロジニアス・プロセッサ

代表的なヘテロジニアス・プロセッサとして ARM 社が開発した big.LITTLE が知られている [16,17]．big.LITTLE は高性能だが消費電力も大きい big コアと，低性能だが消費電力が小さい LITTLE コアを 1 つのチップ上に搭載する．これら 2 種類のコアをプログラムのフェーズに応じて使い分け，シングルスレッド実行をすることで，プロセッサの消費電力を削減する．big コアにはアウト・オブ・オーダ実行を行う Cortex-A15/A57，LITTLE コアにはインオーダー実行を行う Cortex-A7/A53 が用いられる．図 2–1 に big.LITTLE のアーキテクチャを示す．このように，これらのコアはそれぞれが個別の L2 キャッシュを有しており、共有キャッシュは存在しない．そのため，コアの切替えに伴う性能オーバヘッドが大きいという問題がある．

また，研究段階ではあるものの，5 種類のコアを搭載するヘテロジニアス・プロセッサが，Nowatzki らによって提案されている [18]．搭載するコアは，すべての命令を実行できる一般的用途の「General purpose core」の他に，ベクトルデータに対する SIMD 演算を行うコア，データに並列にアクセスし並列ループを得意とするコア，命令レベル並列性が高い命令列を得意とするコア，投機実行の分岐予測を得意とするコアといった，特有の処理を得意とするアクセラレータが含まれて

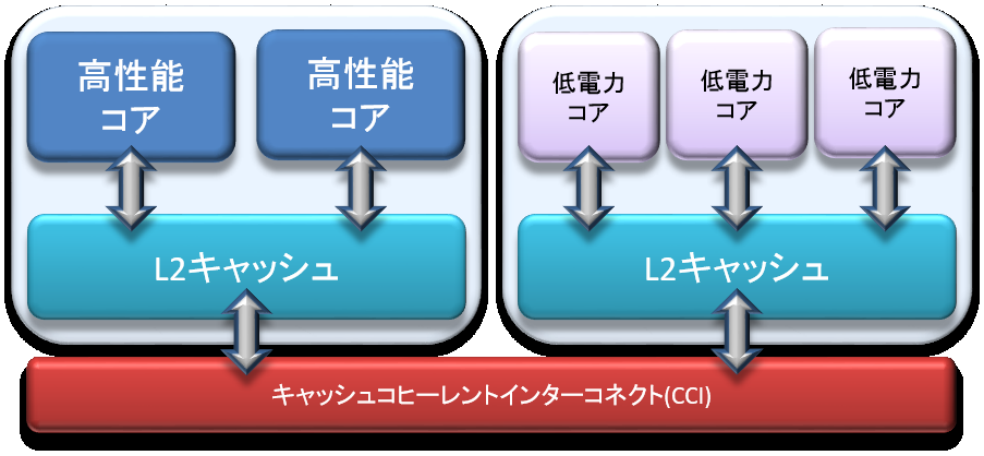


図 2–1: big.LITTLE

いる．

これらの他にも，様々なヘテロジニアス・プロセッサのアーキテクチャが提案されている [2–15]．

2.2　従来の設計探索手法

ここでは，従来のプロセッサの設計探索について紹介し，その問題点を述べる．

- モデル式を用いてシングル・コア・プロセッサの設計探索を行う手法

  Chen らはシングル・コア・プロセッサの設計探索手法として，ArchRanker を提案している [19]．この手法は設計パラメータを変更したときのコア性能を予測するモデルではなく，設計パラメータ間の優劣を判定するモデルを使用する．性能，消費電力それぞれについて少数の探索点 (全探索点の 0.0002%) のシミュレーション結果から設計パラメータ間のランク付けを行うモデルを学習し，これら 2 つのモデルを用いて電力制約を満たす中で最も高い性能を示す設計パラメータを探索する．

  しかし，ArchRanker は，シングル・コア・プロセッサの設計探索手法であり，ヘテロジニアス・プロセッサの設計探索に用いることができない．

- クラスタリングによる手法

  Guevara らは，各候補コアのシングルコアでの性能を示す $BIPS$(Billion

Instruction Per Second)，および $BIPS$ の 3 乗を平均消費電力で割った効率の指標 $BIPS^3/W$ を入力とするクラスタリング手法を提案している [20]．この手法は，主に電力効率を向上させることを目的とする．クラスタ内からコアを選び出す際の入力パラメータ F には CPI (Cycle Per Instruction)，および消費電力を用いる．それまでのコア選択のための各コアの指標は，プログラムの平均的な特性を得るため，プログラムごとに算出した指標の平均，中間値などを用いていた．しかし，一部のプログラムの影響により，安定した値が得られない場合に，性能が低いコアが選出されてしまう．そこで Guevara らは，単位区間ごとに安定した性能を示すコアを選びだすことを目的とするため，クラスタ内からコアを選び出す指標には，分散を新たに用い，分散の小さい値を示すコアの選出を提案した．また，単に分散が小さいコアを選んだ場合，低い性能で安定したコアを選び出す可能性があることから，高性能なコアに有利となるように，単位区間ごとの性能の分散 σ を平均 μ で割った変動係数 (Coefficient of Variation:CoV) の最小値 $argMinCoV(\sigma/\mu)$ を，クラスタ内からコアを選出する方法として採用した．

しかし，主に電力効率を向上させることを目的とするこの手法では，電力効率の高い類似した特性を持つコアを選択してしまう傾向があり，フェーズごとの特性に応じて実行コアを切り替えることによる，エネルギー削減効果が小さくなってしまうことが問題とされている [21, 22]．

- 逐次的な手法

  Tomusk らは，効率をもとにクラスタリングする手法の非妥当性を述べ，ヘテロジニアス・プロセッサの設計探索手法として LUCIE(Least Useful Configuration Iterative Elimination) 法を提案している [22]．この手法は候補コアの中から，特性の広がりが多くなるようにコアを選ぶことを目的とする．

  候補コア間で，電力-性能の 2 次元座標にマップした際の距離を計算し，最も距離が近いコアを，互いに類似した電力性能を持つコアとする．類似したコアの中で最も有用性の低い，候補から外しても良いコアを計算し，選択コア数と等しくなるまで削除していく手法である．入力として各設計パラメータをシングル実行させた場合の，単位区間あたりのパレート最適なコアを集計し評価する．論文 [22] では単位区間は 1 ベンチマークプログラムとしている．

しかし，このアルゴリズムではコアの有用性の算出に探索点間の距離を用いるため，計算量が$O(N^2)$となってしまい，コアの選出に時間がかかってしまうと考えられる．また，例えばBig.LITTLEのようなアウト・オブ・オーダ・コア，インオーダ・コアの組み合わせを考える場合，インオーダ・コアの性能を示す点の分布は一部でアウト・オブ・オーダ・コアの分布と近くなる，あるいはインオーダ・コアのほうがより広い範囲に分布するため，インオーダ・コアが優先して選ばれる傾向がある．

以上に述べたように，ヘテロジニアス・プロセッサの設計探索の研究例は未だ少なく，その精度や計算速度も低い．それらの問題を解決するために，本研究はヘテロジニアス・プロセッサの設計探索を高精度，かつ高速に計算できる手法を開発する．

第3章

問題設定と実験環境

本章は，本研究における範囲設定，及び提案を行うに至るまでの動機について述べる．

3.1 目的関数

本研究の目的関数を以下のように定める．

設計探索の目的関数

N 種類のコアのうち，あるプログラムの集合を実行した場合に，最速で実行可能なコアの実行時間を $T_{fastest}$ とする．$T_{fastest} \times (1 + \alpha/100)$ の実行時間内で実行できるコアのうち，消費エネルギーが最小となる k 個のコアの組み合わせを求める．

α はプロセッサの要求仕様によって定まる性能制約のパラメータであり，設計者によって与えられるものとする．

3.2 設計パラメータ数

先述の通り，設計パラメータのすべての組み合わせのコアを候補コアとし，その数を N とする場合，搭載するコアの数 k 個のヘテロジニアス・プロセッサの設計パラメータの探索点の総数は $O(N^k)$ となる．また，前章で述べた論文 [19] において，研究レベルのシングルコアの探索点数は 5000 万以上存在する．本研究はヘテロジニアス・プロセッサを対象とするため，探索点数は，その累乗となり組合せ爆発が起こる．しかし，各提案手法によって得られたヘテロジニアス・プロセッサ

を評価するにあたり，探索空間内の，性能制約を満たす組み合わせのうち、最も消費エネルギーが小さいコアの組み合わせとの比較を行いたい．この，言わば理想的なコアの組み合わせを求めるためには，全ての組み合わせを全探索する必要がある．そのため，本研究の実験に際しては，搭載するコア数を 2 個と固定，さらに設計パラメータも減らし，理想的な組み合わせを探索できる程度にまで探索空間を小さくする．

本研究において用いた設計パラメータと各パラメータの探索範囲を表 3–1 にまとめた．論文 [19] を参考に，コアの実行方式はアウト・オブ・オーダ，インオーダの 2 種類，他のパラメータは文献を参考に，フェッチ/コミット幅，ALU 数，キャッシュサイズ，分岐予測器のサイズを 3〜4 通りに変化させた．したがって本研究におけるコアの種類は 1944 通りとなり，2 つのコアの組み合わせの総数は ${}_{1944}C_2 = 1,888,596$ 通りとなる．

表 3–1: 設計パラメータ

パラメータ名	設定	パターン数
実行方式	アウト・オブ・オーダ，インオーダ	2
フェッチ/コミット幅	2, 4, 8	3
ALU の個数	2, 4, 6, 8	4
L1I$サイズ	2KB, 8KB, 32KB	3
L1D$サイズ	2KB, 8KB, 32KB	3
L2U$サイズ	256KB, 1024KB, 4096KB	3
GShare サイズ	2K, 8K, 32K	3

3.3　実験環境

プロセッサの性能評価には Onikiri-2 [23] を使用した．また，消費電力の評価には McPAT-1.3 [24] を用いた．

ヘテロジニアス・プロセッサのコア切り替えの判断は，ある一定の命令数がコミットされるごとに行われるものとし，今回の評価では，特記がある場合を除き，論文 [3] と同様 1,000 命令を 1 フェーズとし，フェーズごとに切り替え判断を行うものとした．また，ヘテロジニアス・プロセッサは，理想の切り替え判断を行った

と想定した．理想の切り替え判断を求めるアルゴリズムは，プログラムの処理を完了したときの全実行サイクルが，$T_{fastest} \times (1 + \alpha/100)$ 以内となる範囲で，全消費エネルギーが最小になるように，各々のコアが実行するフェーズを求め，当該フェーズをそのコアに実行させる．なお，アルゴリズムの詳細は付録に記した．

また，今回は簡単のため，コア切替えに伴う時間，エネルギーオーバヘッドを 0 と仮定した．これらのオーバヘッドには命令パイプラインの停止，再開に要する時間，エネルギーや，コアを切り替えることによって発生するキャッシュ・ミス，分岐予測ミスが含まれる．また各コアの静的消費電力は 0 と仮定し，動的消費エネルギーのみを評価した．また DVFS は考慮しない．

最後に，ベンチマークプログラムには，SPEC2006 [25] の中の SPECint2006 を用いた．評価にあたり，2G 命令をスキップした後の 100M 命令を用いた．1 フェーズを 1,000 命令としているため，1 ベンチマークにつきフェーズ数は 10 万となる．

第4章

予備実験

本章は，本研究における手法を発案するまでに行った予備実験について述べる．

4.1 探索空間の分析と研究の指針

本研究では，ヘテロジニアス・プロセッサがより良い性能を示すためには，より多くのフェーズを電力効率の良いコアで実行することが必要であると考えている．本研究では，文献 [22] と同様,「ある実行区間における電力効率が良いコア」を「候補コアのうち，その区間における実行サイクル数と平均電力がパレート最適性を満たすコア（パレート最適なコア)」と定義する．直観的には，あるコアが多くの区間においてパレート最適なコアであれば，そのようなコアをチップに搭載することで，電力効率の良いコアでプログラムを実行する機会が増えると期待できる．

予備実験として，パレート最適なコアの性能と電力の分布を確かめる次の実験を行った．

実験手法

はじめに 1944 種類の候補コアをそれぞれシングルコア実行させたフェーズごとの実行サイクル，及び消費電力のシミュレーション結果を得る．次に，全てのコア間でそれぞれ「フェーズごとに実行サイクル，平均消費電力についてパレート最適なコア」を求める．最後に，各コアについて「フェーズごとにパレート最適なコア」として選出されたフェーズの数を数える．この (コアごとの)「『フェーズご

とにパレート最適なコア』として選出されたフェーズ」を、(コアごとの)「パレート最適フェーズ」と定義する．

なお，評価にあたり，SPEC2006int すべてのベンチマークプログラムではなく，その中でもヘテロジニアス・プロセッサで実行することによるエネルギー削減効果が高いベンチマークプログラム 3 種 `401.bzip2`，`445.gobmk`，`483.xalancbmk` のみを用いた．

実験結果

図 4–1 は横軸に 1 フェーズの実行サイクル，縦軸に 1 フェーズあたりの平均電力をとり，パレート最適なコアの電力-性能分布をヒートマップで描いたものである．

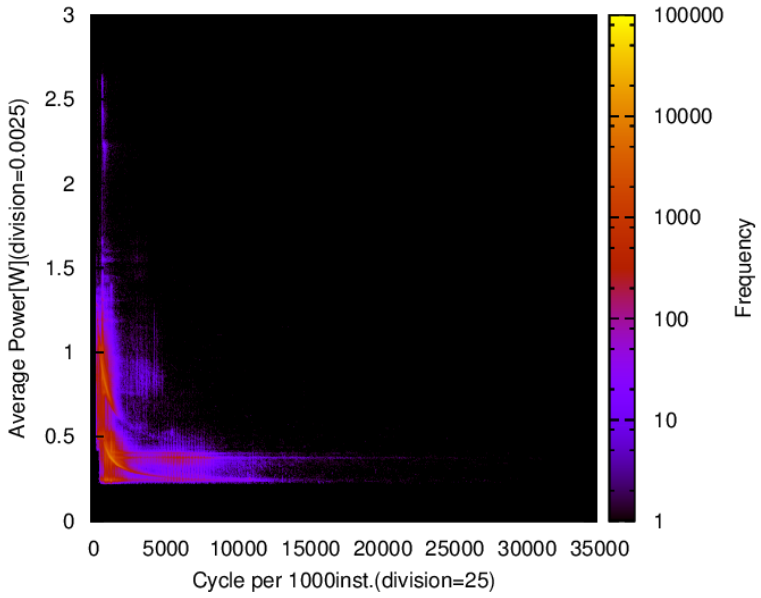


図 4–1: パレート最適なコアの電力-性能分布

図 4–1 からパレート最適なコアは大きく 2 つの領域（平均電力が約 0.6W 以上のコアのグループと，平均電力が約 0.6W 未満のコアのグループ）に分かれている．そのため，この 2 つのグループからそれぞれコアを選べば，十分に消費エネルギーの小さいコアが選出できると考えた．図から，仮の閾値として平均電力 0.6W を定め，その閾値より電力の高い領域にあるコアと低い領域にあるコアに分割し，それぞれの領域に含まれるアウト・オブ・オーダ・コアとインオーダ・コアの割合を

表 4–1 に示す．これより，グラフの上側がアウト・オブ・オーダ・コア，下側がインオーダ・コアが属するグループであることがわかる．

表 4–1: グループ内のコア割合

	アウト・オブ・オーダ・コア	インオーダ・コア
0.6W 以上	24%	1%
0.6W 未満	4%	71%

試しに，この 2 つのグループの分布から，アウト・オブ・オーダ・コアとインオーダ・コアの中でパレート最適コアとして選出された回数 (パレート最適フェーズ数) が最も多いコアをそれぞれ選び（このペアを ”頻度ペア ” と呼ぶことにする），ヘテロジニアス・プロセッサのシミュレーションを行った．

実験の結果を表 4–2 に示す．表中の “ 理想ペア ” は全探索により得られた，全実行サイクルが各性能制約の範囲内で動的エネルギーが最も小さいコアの組み合わせである．なお各性能制約ごとに理想的なコアの組み合わせは異なるため，表中の”理想ペア ” の行のプロセッサ構成は性能制約ごとに異なっている．一方，” 頻度ペア ” は性能制約に対して共通のプロセッサ構成である．

表 4–2: 頻度ペアの性能

性能制約	5%		10%		20%	
	サイクル比	エネルギー比	サイクル比	エネルギー比	サイクル比	エネルギー比
理想ペア	104.73%	77.38%	109.97%	71.90%	119.95%	66.07%
頻度ペア	104.73%	77.38%	109.95%	72.16%	119.76%	67.23%

理想ペアと頻度ペアの消費エネルギーの差は 2 ポイント以内と小さく，3 つのベンチマークを用いた場合について，十分高い精度でコアを選択することができていると言える．また，この際，性能制約 5%の場合に頻度ペアと理想ペアは完全に一致し，理想ペアを選び出すことができた．

以上の実験から，候補となるコアを何らかの手段でクラスタリングしたのち，それらのクラスタから何らかの指標を用いてコアを選び出す手法は，ヘテロジニアス・プロセッサの設計探索手法として有効であると考えられる．

また，頻度ペアの 2 つのコアのパレート最適フェーズにおける性能と電力を図 4–2 に示す．赤点が頻度ペアのアウト・オブ・オーダ・コア，緑点が頻度ペアのイ

ンオーダ・コアのパレート最適フェーズを示しており，それを図 4–2 はこれらの点を図 4–1 に重ねて表示してある．図 4–2 からわかるように，上記の点と図 4–1 と重なりのない広大な領域（右下の領域）が存在している．この事実は，頻度ペアの各コアは，性能制約が緩い場合に，フェーズを高い電力効率で実行できないことを意味している．よって，例えばこの 2 コアのヘテロジニアスプロセッサの不得意なフェーズを補う形で，3 つのコアを搭載したヘテロジニアス・プロセッサを設計する場合は，上で述べた 2 つのコアに加え，パレート最適フェーズが右下の領域にマップされるようなコアを搭載すれば良いと考えられる．

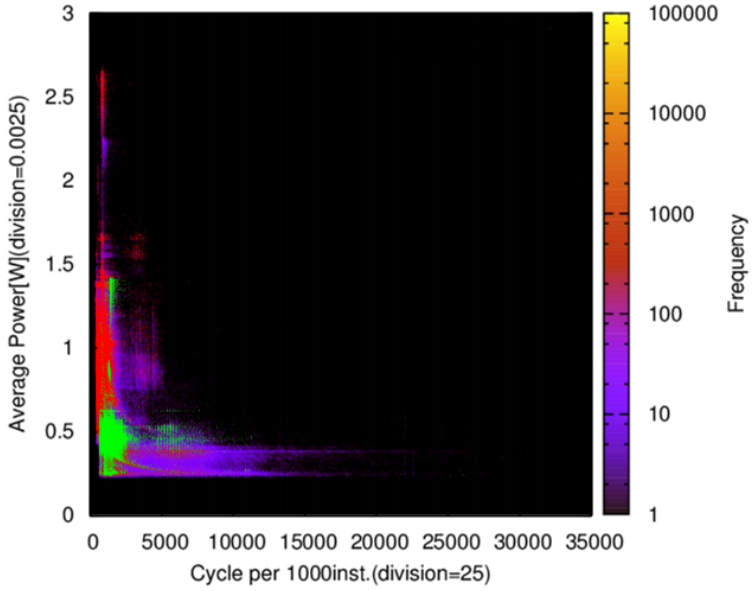


図 4–2: 頻度ペアのパレート最適フェーズの分布
（赤点：アウト・オブ・オーダ・コア，緑点：インオーダ・コア）

第5章

提案手法

本章では，前章の実験結果を基に，ヘテロジニアス・プロセッサの設計探索手法を提案する．

5.1 概要

前章で述べた通り，パレート最適なコアは，いくつかのグループに分けることができる．そのグループ内から適切にコアを選択することで，目的関数の出力に近い性能，エネルギー削減効果を持つヘテロジニアス・プロセッサを選出できると期待できる．

前章では予めアウト・オブ・オーダ・コアとインオーダ・コアという分類がわかっており，明らかなグループ分けが可能だったために，人為的にアウト・オブ・オーダ・コアグループ，インオーダ・コアグループの中から2つのコアを選び出すことができた．しかし，複数の領域が重なっているなど，明らかにグループ分けができない場合には困難となる．また，候補コアの実行方式の種類数と選択コア数が一致しない場合には，上記のグループ分けは難しい．そのため，人為的ではない分類手法を導入する必要がある．

そこで本論文は，クラスタリングを用いたヘテロジニアス・プロセッサの設計探索手法として以下の手法を提案する．ここで，候補コアの集合を S とする．

提案手法

多重集合 S を，実行サイクル，電力を用いた指標のもとに，搭載するコア数 k と等しい数のクラスタ (多重）集合 $S'_1, \ldots, S'_k$ へクラスタリングする．各クラスタ内のコアから，実行サイクル，平均電力などを用いてコアを 1 つずつ選択する．

本手法は，最適に近いと思われるコアの組み合わせをヒューリスティックに決定するものであり，そのコアの組み合わせでプログラムを実行したときの性能や，消費エネルギーを求めるものではない．性能や消費エネルギーを評価するためには，最終的に選択されたコアの組み合わせでシミュレーションを行う必要がある．しかし，このシミュレーションはコア決定後に 1 度だけ行えばよく，シミュレーションに要する時間は，全コアの組み合わせをシミュレートする時間と比べて問題にならないほど小さいと考えている．

5.2　アルゴリズム

本研究で検討したクラスタリング手法，および，クラスタ内からのコアの選択手法を示す．本手法の入力には，各候補コアをシングルコアとしてプログラムを実行させた場合のフェーズごとの実行サイクル，平均電力のシミュレーション結果を用いる．また，このシミュレーション結果を用いることで，各候補コアのプログラムをすべて通して実行した場合の実行サイクルについても既知のものとして使用することができる．

5.2.1　クラスタリング

本研究では，クラスタリング手法として 2 つの手法を検討する．

1 つ目は k-means を用いた手法である．k-means は，データの特徴をもとに意味のあるグループを見つける，単純なクラスタリング手法の一種として知られている [26]．今回は，$k = 2$ とし，前述したアウト・オブ・オーダ・コアグループ，インオーダ・コアグループを発見することを期待し，候補コアのパレート最適フェーズの性能を用いる場合と，候補コアをシングルコアとして全プログラムを実行した際の性能を用いる場合でクラスタリングを試みる。

2 つ目のクラスタリング手法には，搭載するコアのひとつが単体で性能制約を満

たすように選択する．ヘテロジニアス・プロセッサが，シングルコア実行した場合に性能制約を満たせないコアのみで構成されていた場合，それらのコアを切り替えながら実行しても性能制約を満たせる可能性は低い．そこで，各候補コアをシングルコアとしてプログラムをすべて通して実行した際の実行サイクルが，候補コア内の最速コアの実行サイクル $T_{fastest}$ から，性能制約 $\alpha\%$ 以内の実行時間の増加に納まるコアの集合，および，納まらないコアの集合となるようにクラスタリングする．

5.2.2　クラスタからのコアの選択

本研究では，各クラスタからコアを選択する手法として以下の 3 つを検討した．

1 つ目は，各クラスタのコアの中で電力効率が良かったフェーズが多いコアを選出する手法である．つまり，各クラスタ S'_n 内で最もパレート最適フェーズが多いコア C_i を選出する手法である．

2 つ目は，各クラスタ内から，フェーズごとの性能と電力のばらつきが大きなコアを選択する手法である．この手法では，各クラスタ S'_n のそれぞれのパレート最適フェーズの性能，電力の共分散を使用する．共分散は「実行時間が遅い」かつ「電力が大きい」であったり,「実行時間が速い」かつ「電力が小さい」場合，負の方向に大きな値を取る．そのため，様々な性能，電力特性を持つコアを選択することは，共分散の最小値を求めることに等しい．

3 つ目は，各クラスタ内で性能が良いコアを 2 つ選ぶ手法である．クラスタ分けされた S'_n 内のコアの中でシングルコアとして実行した際の性能が最も良いコア，つまり実行サイクルが最も小さいコアをそれぞれ選択する手法である．

第6章

評価実験

本章では提案した手法を実装し，得られたヘテロジニアス・プロセッサの性能評価を行う．

6.1 実験手法

6.1.1 クラスタリングアルゴリズムへの入力データ

コア，あるいはパレート最適フェーズをクラスタリングするにあたり，クラスタリングアルゴリズムへの入力データは2種類用意した．1種類目は，予備実験と同じもの，つまり，初めに候補コアをシングルコアとしてベンチマークを実行させたフェーズごとの実行サイクル，及び平均消費電力のトレースを用いた．それをもとに，各フェーズごとのパレート最適な特性を示す点のコア，実行サイクル，平均消費電力を選びだし，コアごとのパレート最適フェーズのみの性能を取り出したものを用いた．これを入力Aとする．2種類目は，上記のフェーズごとのシミュレーション結果をもとに，プログラム全体を通して実行した場合の実行サイクル，平均消費電力をコアごとに算出したものを用いた．これを入力Bとする．

なお，全区間通して実行した際に全く同じ性能を示すコアが存在した．それらのコアに関しては，計算量の削減のため，より演算器数が少なかったり，メモリサイズが小さいパラメータの構成のコアと等しいと仮定し，入力から除外した．

6.1.2　クラスタリング

前章の 2 つのクラスタリングについて，k-means については入力 A, B を使用した．また，性能制約によるものは入力 B のみを用いた．

なお，k-means はデータ列の間に著しくオーダーの差があると，クラスタリング結果が大きい数値のデータ列に強く影響されてしまう．平均電力の数値は $O(10^{-1})$ に対し，1 フェーズの実行サイクルは $O(10^3)$〜$O(10^4)$，プログラム全区間の場合には $O(10^9)$ 程度となるため，k-means を適用する前に，2 つのデータ列が同じ値の範囲になるように，実行サイクル，平均電力それぞれの最大値でそれぞれのデータ列を正規化している． なお，k-means クラスタリングには matlab を用いた．

6.1.3　クラスタからのコアの選択

分別したクラスタから，以下の手法でコアを選出した．最後にそのクラスタペアを全探索で得た理想ペアとエネルギー削減効果と比較した．

電力効率をもとにした選択

S'_1, S'_2 内のパレート最適フェーズが最も多いコアを探索し，ヘテロジニアス・プロセッサのコアの組み合わせ「パレート最適ペア」として出力する．

共分散をもとにした選択

S'_1, S'_2 のそれぞれに含まれる各コアについて，パレート最適なフェーズを選択し，その性能，電力の共分散を求める．前章の通り，負の方向に大きいほど目的とするコアの特徴に近づくため，それらのクラスタのコアに対して最小値を求めた．

クラスタ内で性能が良いコアを選択

S'_1, S'_2 のそれぞれに含まれるコアについて，実行サイクルが最小のものを選択した．

各手法により得られたコアのペアの性能は，全探索の過程で得られた，探索空間内全てのヘテロジニアス・プロセッサのコアの組み合わせの実行サイクル及び，消費エネルギーの表を参照する．

6.1.4　評価方法

各手法により選択されたコアの組み合わせについて，候補コア内の，全プログラムを通して実行した際の最速コアに対する性能比，消費エネルギー比を比較する．比較には前章の予備実験同様，全探索の過程で得られたコアの組み合わせに対する性能，電力の表を参照する．設計パラメータである許容する最速コアに対する実行サイクルの遅延は 5%，10%，20%，30%と定める．

6.2　実験結果

6.2.1　k-means によるクラスタリング

2 種類の入力に対し k-means を試みた．

パレート最適フェーズを用いた場合

SPECint2006 の 12 種のベンチマークの 120 万個のフェーズから得られたパレート最適フェーズは，すべてのコアを総計して約 3 億 5000 万になった．そのすべての点を入力として k-means を適用 (1000 回の反復，初期値を変えて 10 回) したが，重心の計算が収束せず，クラスタ分けすることができなかった．今回は用いているプログラムは SPECint2006 だが，SPEC2006 のプログラムは全 29 種になる．これらすべてを使用した場合を考慮すると，パレート最適フェーズを k-means でクラスタリングする手法は現実的ではない手法と判断できるため，以降，k-means によるクラスタリングですべてのプログラム区間のパレート最適フェーズを用いた手法は断念した．

補足の実験として，データ点を減らし，予備実験の際に用いた `401.bzip2`，`445.gobmk`，`483.xalancbmk` のみを用いる場合を検討した．この時の k-means によりクラスタリングされたパレート最適コア群 S'_1, S'_2 を図 6–1 に示す．おおよそ，予備実験において定めた閾値 0.6W 付近から上と下にクラスタリングされている．このクラスタからパレート最適フェーズが最も多いパレート最適ペアを選出した．その性能と理想ペアの性能，予備実験における頻度ペアの性能を表 6–1 に示す．

この手法によって選び出されたクラスタペアは頻度ペアと一致し，予備実験においてアウト・オブ・オーダ・コアで最もパレート最適フェーズが多かったコアと，インオーダ・コアで最もパレート最適フェーズが多かったコアを選び出すこ

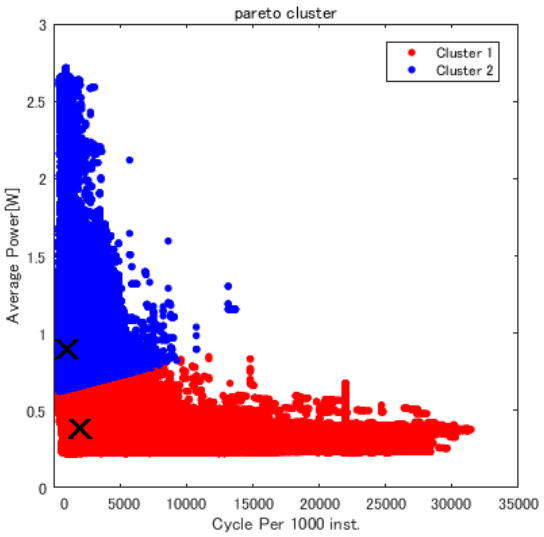


図 6–1: 少ないプログラム数における k-means でクラスタリングしたパレート最適コア群

表 6–1: クラスタペアの性能 (少ないプログラム数)

性能制約	5%		10%		20%		30%	
	サイクル比	エネルギー比	サイクル比	エネルギー比	サイクル比	エネルギー比	サイクル比	エネルギー比
理想ペア	104.73%	77.38%	109.97%	71.90%	119.95%	66.07%	127.95%	67.20%
頻度ペア	104.73%	77.38%	109.95%	72.16%	119.76%	67.23%	127.83%	68.45%
クラスタペア	104.73%	77.38%	109.95%	72.16%	119.76%	67.23%	127.83%	68.45%

とができた．

全プログラム通して実行した性能を用いた場合

全プログラムを通して実行した場合の性能を示す点は，候補コアと等しくなるため 1944 個である．そこから同じ性能を示すコアを除外した上で，k-means を適用した結果を図 6–2 に示す．

図 6–2 より，性能差の生じるアウト・オブ・オーダ・コアとインオーダ・コアを明らかに分別することができている．したがって，コアの集合をクラスタリングする際にはパレート最適フェーズを用いる必要はなく，プログラム全体を通して実行した際の性能を用いるだけで充分と推察できる．

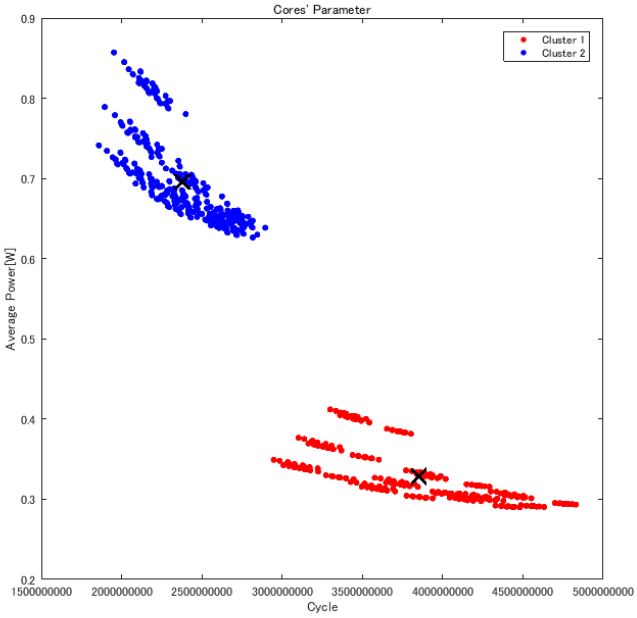


図 6–2: 全プログラム通して実行した性能を用いた場合の k-means クラスタリング

6.2.2　性能で分けた場合のクラスタリング

全プログラムを通して実行した際の性能で分けた場合のクラスタリングは図 6–3 の通りである．

本実験において，許容する最速コアに対する遅延，性能制約は 5%，10%，20%，30%である．遅延 5%以内のコアを赤，10%以内のコアを緑，20%以内のコアを青，30%以内のコアを紫，遅延が 30%よりも大きいコアを水色で示した．このクラスタリング結果は，例えば性能制約 20%のとき，性能制約を満たすクラスタ S'_1 に赤，緑，青のコア，満たさないクラスタ S'_2 には紫，水色のコアが含まれる．このクラスタリングの方法では，候補コア内のアウト・オブ・オーダ・コア，インオーダ・コアは完全には分離できない結果となった．

6.2.3　クラスタからのコアの選出

クラスタ内からそれぞれパレート最適フェーズが最多のコア，共分散が最小のコア，およびサイクル最速コアを選択した．それらの組み合わせによるコアが示した実行サイクル，消費エネルギーを表 6–2 に示す．ただし，パレート最適フェー

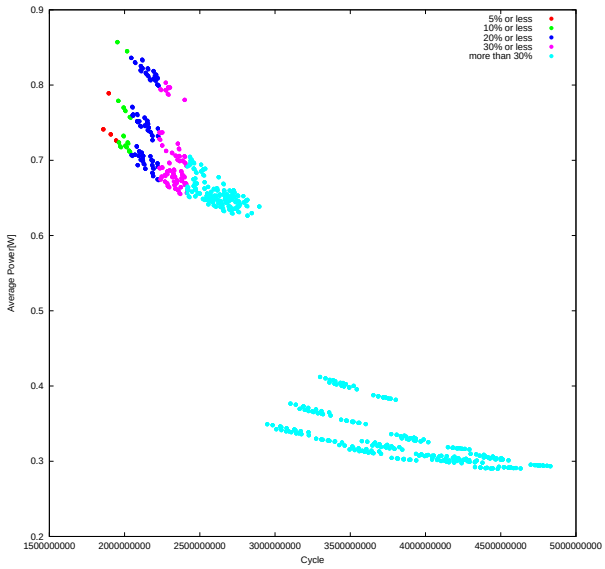


図 6–3: 全プログラム通して実行した性能を用いた場合の性能別のクラスタリング

ズを用いた k-means による手法は，現実的ではないと判断したため，全プログラムを通して実行した際の結果をクラスタリングした．

表 6–2: 選出したコアの性能

性能制約	5%		10%		20%		30%	
	サイクル比	エネルギー比	サイクル比	エネルギー比	サイクル比	エネルギー比	サイクル比	エネルギー比
理想ペア	104.93%	82.29%	109.83%	77.77%	119.89%	74.63%	129.90%	73.21%
k-means-パレート最適ペア	135.93%	118.84%	135.93%	118.84%	135.93%	118.84%	135.93%	118.84%
k-means-共分散ペア	119.33%	128.87%	119.33%	128.87%	119.93%	113.17%	129.91%	93.37%
k-means-最速ペア	**105.00%**	**82.38%**	**109.86%**	**77.87%**	**119.57%**	**75.13%**	**129.87%**	**74.16%**
性能制約-パレート最適ペア	104.80%	86.17%	109.87%	87.89%	119.56%	105.12%	129.56%	87.95%
性能制約-共分散ペア	100.96%	107.63%	109.30%	111.70%	119.33%	128.87%	123.36%	115.79%
性能制約-最速ペア	99.98%	99.89%	100.01%	99.86%	98.75%	98.75%	99.68%	96.07%

表 6–2 に示すように，いずれの性能制約においても理想ペアには約 1 ポイント程度及ばないものの,「全プログラムを通して実行した際の実行サイクル数，電力」を「k-means でクラスタリング」し,「各クラスタ内の実行サイクル最速のコアを選択」する手法で選んだプロセッサが 20% 程度のエネルギー削減効果を示した．またほかの手法は消費エネルギーがほとんど削減されない，または逆に消費エネルギーが増えてしまう結果となった．

性能制約をもとにクラスタリングした場合に適切なコアが選択できない原因は，

性能制約の閾値に近い実行時間のコアが高性能コア側に選択されてしまった結果，ヘテロジニアス・プロセッサとして実行するうえで低性能コア側に切り替える機会が少なくなってしまうためであると考えられる．その点，実行サイクル最速のコアを選ぶ手法は，性能制約の閾値に対して余裕があるため，切り替えることができる機会を多く持つ．したがって，ヘテロジニアス・プロセッサのエネルギー削減効果を大きく得られたと考えられる．

以上の結果から，本研究において設定した条件の下では，各提案手法の中で「全プログラムを通して実行した際の実行サイクル数，電力」を「k-means でクラスタリング」し,「各クラスタ内の実行サイクル最速のコアを選択」する手法が最適といえる．

また，3 つのベンチマークの際に選んだ頻度コアは,「アウト・オブ・オーダ・コア最速」のコアと「インオーダ・コア最速」のコアであったため，予備実験にて得られたコアの特徴とも一致する．

6.3　選出したコアのヘテロジニアス・プロセッサ実行時の切り替え回数

今回選択したコアで作成したヘテロジニアス・プロセッサが，全プログラムを通して実行した際にコアを切り替えた回数を数えた．本論文における性能評価ではコアの切り替えオーバヘッドを考慮していないが，将来的にはそれを含有した性能を示す必要があるためである．

本来，コアを切り替えることができる機会は，全プログラム 12 種の合計フェーズ数 120 万の間なので，1,199,999 回であるが，簡単のため，1 回の誤差を許容し，初期状態をアウト・オブ・オーダ・コアと仮定し，120 万回の切り替え機会があったとしている．表 6–3 に，性能制約ごとの理想ペアと本論文で最適な手法で選出したコアの切り替え回数を示す．表中の割合は切り替え機会の回数に対する切り替えた回数の割合を示す．

表 6–3: コアの切り替え回数

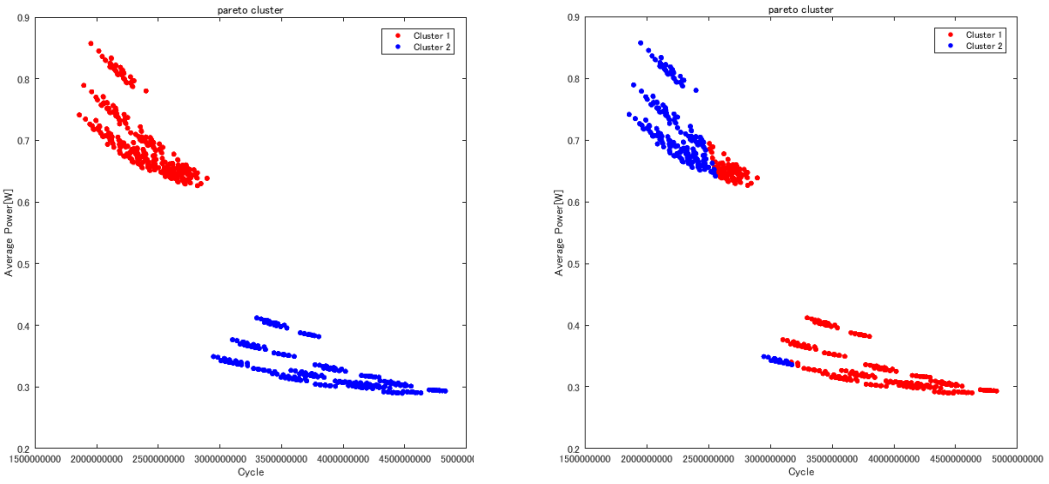
性能制約	5%		10%		20%		30%	
	切り替え回数	割合	切り替え回数	割合	切り替え回数	割合	切り替え回数	割合
理想ペア	70578	5.88%	49445	4.12%	29421	2.45%	21251	1.77%
k-means-最速ペア	63556	5.30%	40019	3.33%	16023	1.34%	19365	1.61%

性能制約が厳しい 5%では，アウト・オブ・オーダ・コアでの実行を基本として，

インオーダ・コアで実行可能なフェーズも少なく，さらに連続していないので，切り替え回数が多い．性能制約 10%，20%では，アウト・オブ・オーダ・コアで連続的に実行できるフェーズと，インオーダ・コアで連続的に実行できるフェーズがバランスよく出現し，切り替え回数が少なくなったと考えられる．一方，性能制約 30%で切り替え回数が増えたのは 5%の場合と反対に，それまで連続してアウト・オブ・オーダ・コアで実行していたフェーズをインオーダ・コアで実行するフェーズが増え始める．そのため，緩すぎる性能制約では，切り替えオーバヘッドが増加してしまうこともあるため，それも考慮しながら設計探索を行うべきである．

6.4　従来研究との比較

「全プログラムを通して実行した際の実行サイクル数，電力」を「k-means でクラスタリング」し,「各クラスタ内の実行サイクル最速のコアを選択」する手法で導き出した組み合わせと従来研究の BIPS をもとにクラスタリングし CoV を用いてコアを選出する手法 [20]，および LUCIE 法で選出する手法 [22] との比較をした．初めに $BIPS$ をもとにしたクラスタリングを図 6–4(a)，$BIPS^3/W$ をもとにしたクラスタリングを図 6–4(b) に示す．



(a) $BIPS$ を使用　　(b) $BIPS^3/W$ を使用

図 6–4: 従来手法によるクラスタリング

$BIPS$ をもとにクラスタリングした手法はアウト・オブ・オーダ・コアとインオーダ・コアを分別できている．しかし，$BIPS^3/W$ をもとにしたクラスタリングはアウト・オブ・オーダ・コアとインオーダ・コアを分別することができていない．これは，効率を指標にしているために，アウト・オブ・オーダ・コアの中の効率の値と，インオーダ・コアの中の効率の値が近くなってしまったためと考えられる．

BIPS を用いる手法では実行サイクル，または平均消費電力で CoV を計算する．本論文ではそれぞれの指標で各クラスタからコアを選出した．これら BIPS を用いたクラスタリングで得られたコアと，LUCIE 法を用いて得られたコアの性能を表 6–4 に示す．

表 6–4: 従来手法との性能の比較

性能制約	5%		10%		20%		30%	
	サイクル比	エネルギー比	サイクル比	エネルギー比	サイクル比	エネルギー比	サイクル比	エネルギー比
理想ペア	104.93%	82.29%	109.83%	77.77%	119.89%	74.63%	129.90%	73.21%
k-means-最速ペア	**105.00%**	**82.38%**	**109.86%**	**77.87%**	**119.57%**	**75.13%**	**129.87%**	**74.16%**
$BIPS$-CoV（サイクル）	145.15%	127.27%	145.15%	127.27%	145.15%	127.27%	145.15%	127.27%
$BIPS$-CoV（平均電力）	104.94%	90.81%	109.90%	86.54%	118.88%	82.47%	125.32%	81.46%
$BIPS^3/W$-CoV（サイクル）	118.76%	129.09%	118.76%	129.09%	119.99%	111.29%	120.55%	111.05%
$BIPS^3/W$-CoV（平均電力）	161.89%	74.80%	161.89%	74.80%	161.89%	74.80%	161.89%	74.80%
LUCIE	217.60%	88.70%	217.60%	88.70%	217.60%	88.70%	217.60%	88.70%

BIPS を用いてクラスタリングし，電力を用いた CoV でコアを選出した場合を除き，本探索空間において，性能制約を満たすコアを探し出すことはできなかった．また，BIPS を用いてクラスタリングし，電力を用いて CoV で選出する場合は，一定のエネルギー削減効果はあるものの，本研究よりも削減することができなかった．

まず，サイクルの CoV を求める場合，サイクル数が安定するのは，常にどのようなフェーズでも同じように動作する場合となる．つまり，どのようなフェーズでも回路資源が十分に存在するか，どのようなフェーズでも回路資源が不足している様なコアが選ばれやすくなると考えられる．本探索空間上では，常に回路資源が十分にあるコアの存在は想定しづらい．そのため，アウト・オブ・オーダ・コア側に性能が悪いコアを選んでしまい，性能が悪いヘテロジニアス・プロセッサとなってしまったと考えられる．

また，電力の CoV を求める場合，今回は動的電力の情報のみを用いているため，

フェーズごとの平均電力は IPC(Instruction Per Cycle) に相関する．本実験においては命令数が固定のため，実行サイクル数に依存する．BIPS を用いてクラスタリングした場合，アウト・オブ・オーダ・コアには最速コアが選出された．他のコアと比較し，フェーズごとに極端に実行が遅れることが少なく，実行サイクル数のばらつき，ひいてはフェーズごとの電力のばらつきが少なかったことがコアの選出された要因と考えられる．また，インオーダ・コアは，比較的回路資源が豊かなコアが選ばれた．そのため、他のインオーダ・コアより速く，また，平均電力のばらつきもなく実行でき，インオーダ・コアとして選出されたと考えられる．

LUCIE 法で選ばれたコアは，性能制約を満たせない結果となった．これはインオーダ・コア 2 つが選択されたことが要因である．本探索空間はインオーダ・コアの性能が広く分布しているためと考えられる．よって本研究による手法は，いずれの従来研究を用いた手法よりも優れた結果が得られた．

6.5　探索空間の大きさを変えた場合

異なる大きさの探索空間について評価する．表 3–1 の探索空間から実行方式の項目を減らし，アウト・オブ・オーダ・コアのみを組み合わせた探索空間を考える．このとき,「全プログラムを通して実行した際の実行サイクル数，電力」を「k-means でクラスタリング」し,「各クラスタ内の実行サイクル最速のコアを選択」する手法を用いて導き出したコアを，この探索空間における理想ペアと比較する．

この時の k-means によるクラスタリング結果を図 6–5 に示す．この 2 つのクラスタから，上記手法にて 2 つのコアを選択した．それらのコアの実行サイクル数，消費エネルギーを表 6–5 に示す．

表 6–5: 探索空間を変え選出したコアの性能

性能制約	5%		10%		20%		30%	
	サイクル比	エネルギー比	サイクル比	エネルギー比	サイクル比	エネルギー比	サイクル比	エネルギー比
理想ペア	98.47%	94.09%	98.47%	94.09%	98.47%	94.09%	98.47%	94.09%
k-means-サイクル最速	101.04%	96.82%	101.04%	96.82%	101.04%	96.82%	101.04%	96.82%

クラスタリングの結果は「速い」コアと「遅い」コアに分けることができている．理想ペアは，シングルコアの実行サイクルが最速コアよりも実行速度が速く，なおかつ消費エネルギーが低いコアとなっている．一方，選択したコアは最速コ

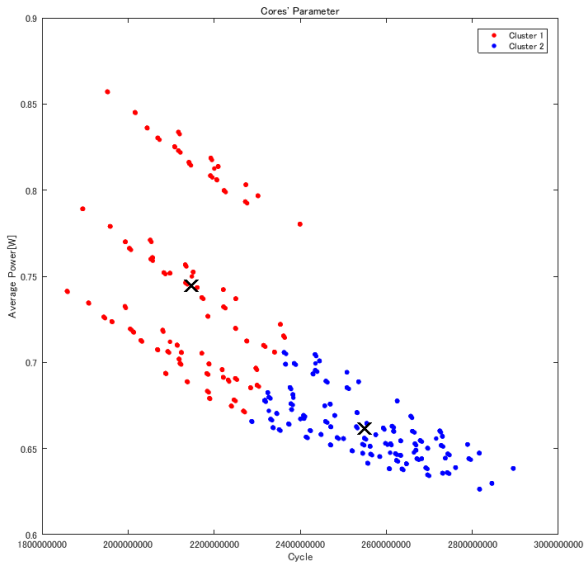


図 6–5: 探索空間を変えた場合の k-means クラスタリング

アよりは遅いものの，当初の目的関数の出力に近いエネルギー削減効果を持つコアを発見できている．いずれの組み合わせも，アウト・オブ・オーダ・コア同士の組み合わせということもあり，元来処理時間が短いコアなので，ほとんど遅延せず，今回設定した性能制約下では，各性能制約間で同じ結果になってしまった．以上の結果より，この手法は異なる探索空間でも目的関数の出力に近いコアを得ることを期待できる．

6.6　搭載するコア数を増やす場合

本研究においては搭載するコア数 $k = 2$ としているが，最新の研究では 3 つ以上のコアを搭載するヘテロジニアス・プロセッサも発表されている [18]．今回は探索空間の大きさの都合上，$k \leqq 3$ の場合のヘテロジニアス・プロセッサの性能評価には至らなかったものの，搭載するコアを増やした場合に本手法を適用する際のクラスタリングについて述べる．

2 章で述べた通り，インオーダのコアの性能は，アウト・オブ・オーダ・コアの性能より広く分布する傾向がある．したがって，例えば 3 コアのヘテロジニアス・プロセッサの設計探索を行う場合，インオーダ・コアが得意とする領域がさらに 2

種類にクラスタリングされると考えられる．

表 3–1 の探索空間から，搭載するコア数 $k = 3$ コアのヘテロジニアス・プロセッサを作成すると仮定する．プログラムを通して実行した際の性能，消費電力で示した点の k-means によるクラスタを図 6–6 に示す．

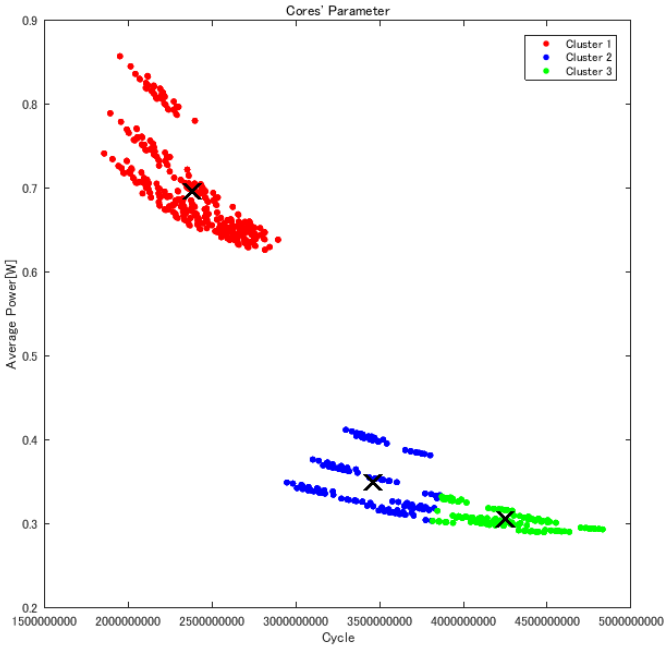
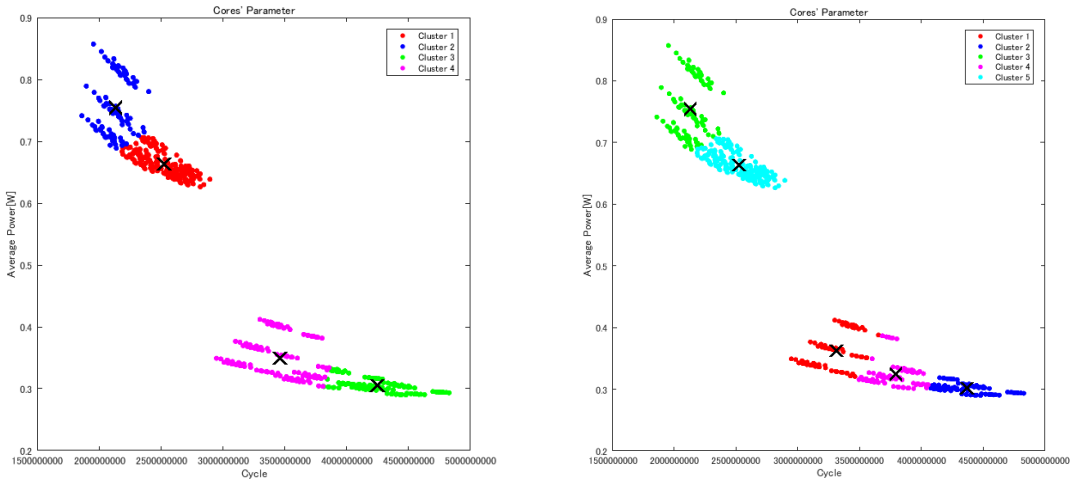


図 6–6: 搭載するコア数を 3 つにした場合の k-means クラスタリング

この手法において、各クラスタから選出されたコアは，そのクラスタに含まれるコアを代表するものとなる．よって搭載コア数を増やし，クラスタ数を増やした場合，より性能が広く分布し，より多くのコアを 1 つのコアが代表していたクラスタが分割されると考えられる．図 6–6 からインオーダ・コア群が分割され，予測したクラスタ分けとなった．よって，k-means でクラスタリングする手法は搭載するコア数を増やして適用しても性能の良いヘテロジニアス・プロセッサを作成できると期待できる．

さらに $k = 4$，$k = 5$ の場合を考える．$k = 3$ においてインオーダ・コア群が分割された結果，アウト・オブ・オーダ・コア群の領域が一番大きくなった．そのため，$k = 4$ では，アウト・オブ・オーダ側が分割されると考えられる．k-means を用いたクラスタリングを図 6–7 に示す．図 6–7(a) では，アウト・オブ・オーダ・コア側のクラスタが分割される結果となった．同様のことが $k = 5$ の場合にも考

えることができ、この場合にはインオーダ・コア群が更に分割される結果となることが図 6–7(b) からわかる.



(a) $k = 4$ の場合のクラスタリング　(b) $k = 5$ の場合のクラスタリング

図 6–7: さらに搭載コア数を増やす場合の k-means クラスタリング

第7章

結論

7.1 まとめ

本論文では，ヘテロジニアス・プロセッサに搭載するコアを決める設計探索手法を提案した．いくつか提案した詳細な手法のなかで，k-means でクラスタリングしたのち，各クラスタ内の最速コアを選ぶ手法が最も目的関数に近いコアを選択できることを示した．また，従来研究とも比較を行い，よりエネルギー削減効果の高いコアを選択することができることを示した．並びに，異なる大きさの探索空間での設計探索を行い，本手法が有効であることを示した．

本手法で用いた k-means の計算量はクラスタ分けする領域数 k に対し，$O(kN)$ であるため，ほぼ計算量を $O(N)$ まで減らせている．例えば LUCIE 法の場合，計算量は $O(N^2)$ であるから，従来研究と比較しても，高速に計算することができる．

また，本研究では詳細には検討できなかったものの，搭載するコア数を増やした場合にも本手法が適用できる可能性も示した．

7.2 今後の課題

本研究では，探索空間の変更は検討段階に用いていた探索空間を縮減する方向で行ったが，浮動小数点演算器の数などの設計パラメータを増やす方向でも有効であるかは調査を要する．

また，実験の中で 0 と仮定したコアの切り替えに伴う性能，エネルギーオーバヘッドを見積もったうえで，それを加味した評価 を行う必要がある．

また，搭載するコア数を増やす場合，例えば本論文の設計パラメータ上では $k = 3$ で探索空間の数は，$k = 2$ の 1,888,596 点から 1,222,551,144 点に増え，理想の組み

合わせを見つけることが困難となる．したがって，ほかの従来研究との比較の下に，数を増やした時の検討を行うことができると，この手法の有効性がさらに示せると考えられる．

謝辞

本研究を行うにあたり，研究の場を与えていただき，なおかつ適切な御指導，御助言をして頂いた本多弘樹教授，三輪忍准教授，八巻隼人助教，ならびに新谷隆彦准教授に心より深く感謝致します．また，名古屋大学の塩谷亮太先生，コロンビア大学の佐々木広先生，東京大学の有間英志先生にも，研究の方針について多くの助言を頂き，研究を進めることができました．

もともと同期だったのが 1 年ずれて，さらに進学に伴い建物も離れてしまった，先進理工学研究科上野研究室の卒業生の南出雄佑氏には，大学院生生活 1 年目の何から始めるべきかわからない状況で，様々なことを相談させていただき，心の支えとなりました．

最後に，本研究を御支援，御協力下さった高性能コンピューティング学講座，本多・三輪・八巻研究室の皆様，7 年間の学生生活をご支援いただきました，家族，親族の皆様にも御礼申し上げます．

平成 29 年 1 月 26 日

付 録A

コアを切り替えて実行するべきフェーズを見つけるアルゴリズムの詳細

序論に述べたとおり，ヘテロジニアス・プロセッサは，フェーズごとに実行するコアを切り替えながらプログラムを処理する。ここでは，本論文中でヘテロジニアス・プロセッサの理想ペアを見つける全探索時にエネルギ削減効果の評価をするために用いた，理想的に電力効率のよいコアで実行するべきフェーズを選択するアルゴリズムについて，理論的な段階での方針と実際に実装した手法を述べる．

なお，この理論もまだ理想的な手法と考えている段階で，実際に理想的な手法であることの証明に関してはなされていない．

理論的な段階としての手法

ヘテロジニアス・プロセッサはbig.LITTLE [17] のように，アウト・オブ・オーダ・コアとインオーダ・コアの2コアが搭載されているものとして，コア同士のプログラム全体の実行性能の速い，遅いの差，消費電力の大小の差が明確であると仮定する．方針は以下の通り．また，模式図を図 A–1 に示す．

切り替えるフェーズを選ぶ方針

全実行フェーズを速いコアで実行したとする．あるフェーズの実行コアを遅いコアに切り替えた場合に生じる遅延サイクル数が小さい順に，遅いコアに切り替えるフェーズを増やしていく．初期値を速いコアの実行サイクル数として，切り替えるフェーズの数について累積サイクル数が性能制約を超えない範囲内のフェーズを遅いコアで実行する．

その方針を，手法として表したものを箇条書きに書き下す。

1. フェーズごとの速いコアに対する遅いコアの実行サイクル数の差を求める．
   短い実行時間の中ではインオーダ・コアがアウト・オブ・オーダ・コアより

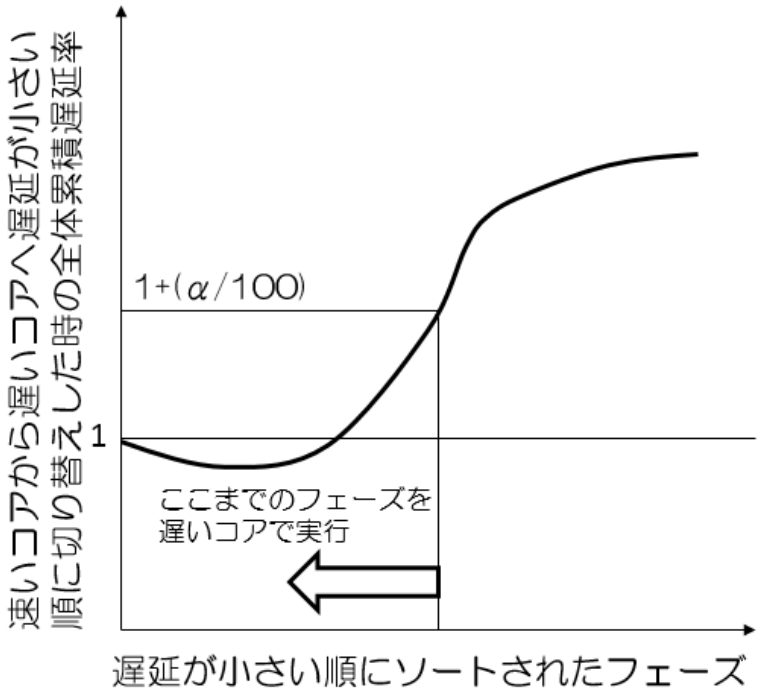


図 A–1: 理想のアルゴリズムの模式図

速く実行できる可能性があるため，これは負の値を取ることもある．

2. 速いコアで全プログラムを実行した場合の実行サイクル数に，遅延サイクル数が小さい順にサイクルを加えていく．

3. 累積サイクルが性能制約を超えるまで加えていき、超える直前のフェーズまでが遅いコアで実行すべきフェーズとなる．

本理論は，累積サイクルの変化が連続であると仮定していることが，現実と反している点である．また，全探索では同時に計算する数を増やす都合上，最大で 1943 通りの組み合わせを同時に処理する手法をとった．そのため，全フェーズ数 120 万 $\times$ 1943 コア分の計算を行う計算領域が実験環境上には必要となってしまった．そのため，このアルゴリズムをいくつかの条件を簡略化して実装した．

実際に実装したアルゴリズム

前述の事情により，以下のような手法を取った．変更した点は，遅いコアの速いコアに対する遅延率 $T_{phase}^{slower}/T_{phase}^{faster}$ ごとに遅延サイクルを合計している点，また，遅延率が小さい順にサイクルを加えていく点である．遅延サイクルごとに合計せ

ず，遅延率で行った理由は，遅延率で合計するほうが遅延サイクルごとにエネルギ削減効果が高い切り替えを行うことができた，つまりより良い切り替えるフェーズを見つける手法となったためである．

また、前述の理論では消費電力の大小関係が明確であると仮定しているが，アウト・オブ・オーダ・コア同士やインオーダ・コア同士の組み合わせの性能を計算する場合には，それも定かではない．フェーズ内で遅いコアで実行することにより，エネルギが削減されず，反対にエネルギが増えてしまうフェーズは，そのフェーズを遅いコアで実行する意味が無い．よってこのアルゴリズムでは無条件に速いコアで実行するフェーズとして処理を行った．

1. フェーズごとの速いコアと遅いコアの消費エネルギを比較し，遅いコアの消費エネルギが高い場合は，そのフェーズは無条件に速いコアで実行するとして，次のフェーズの計算に移る．

2. フェーズごとの速いコアに対する遅いコアの実行サイクル数の遅延サイクル数および遅延率を求める．

3. 分解能を設定し，分解能で区分した遅延率ごとに，フェーズごとの遅延サイクルを合計する．

4. 遅延率が小さい順に遅延率ごとの合計遅延サイクルを速いコアの実行サイクルに加えていく．

5. 累積サイクルが性能制約を超える直前のフェーズまでが遅いコアで実行すべきフェーズとなる．

また，模式図を図 A–2 に示す．この例は，全フェーズ数が 3，最速コアの実行サイクル数 $T_{fastest}$ が 50 サイクル，性能制約 40%，つまり，実行サイクル数が 70 サイクルまでの遅延を許容する場合の動作を示す．また，各フェーズの消費エネルギは常に遅いコアが小さいものとする．この例の場合，遅延率が低い順に遅延サイクルを加えていくと，遅延率 1.3 の時実行サイクルが 67 サイクルで性能制約を満たしていたものが，遅延率 1.4 の時に 74 サイクルとなり，性能制約を満たさなくなる．よって，各フェーズを遅いコアで実行すると判断する閾値を，フェーズごとの遅延率 1.3 以下として設定する．そして，シミュレートの際は,「フェーズごとの遅延率が 1.3 以下の時に遅いコアで実行する」としてヘテロジニアス・プロセッサの性能評価を行う．

なお，この例では,「フェーズごとの遅延率が 1.1，および 1.4 の時に遅いコアで実行する」方針を取るほうが，全実行サイクル 69 サイクルとなり，本来切り替えるべきフェーズとして，より有効である．その点については，本実験は実行サイクル数のオーダーが 4〜5 桁程度であること，全フェーズ数が 120 万と多いために影響が小さくなると考え，無視している．

(例)
フェーズ数3
最速コア=50サイクル
性能制約40%(全体で70サイクルまで許容)

区間	1	2	3
速いコア(サイクル)	20	20	20
遅いコア(サイクル)	26	27	23
遅延率 (分解能0.1，分解能の間は切り上げ)	1.3	1.4	1.2

遅延率	0.1	～	1.1	1.2	1.3	1.4	1.5	～
遅いコアに置き換えた時に増えるサイクル	0		2	0	5	7	0	

図 A–2: 実装したアルゴリズムの模式図

参考文献

[1] P. Greenhalgh: “big.LITTLE processing with ARM Cortex-A15 & Cortex-A7”, white paper (2011).

[2] S. Navada, N. K. Choudhary, S. V. Wadhavkar and E. Rotenberg: “A unified view of non-monotonic core selection and application steering in heterogeneous chip multiprocessors”, PACT ’13: Proceedings of the 22nd international conference on Parallel architectures and compilation techniques, pp. 133–143 (2013).

[3] A. Lukefahr, S. Padmanabha, R. Das, F. M. Sleiman, R. Dreslinski, T. F. Wenisch and S. Mahlke: “Composite Cores: pushing heterogeneity into a core”, MICRO 45: Proceedings of the 45th annual IEEE/ACM international symposium on Microarchitecture, pp. 317–328 (2012).

[4] M. Pricopi, T. S. Muthukaruppan, V. Venkataramani, T. Mitra and S. Vishin: “Power-performance modeling on asymmetric multi-cores”, CASES ’13: Proceedings of the 2013 international conference on Compilers, architectures and synthesis for embedded systems, pp. 15:1–15:10 (2013).

[5] J. Sampson, M. Arora, N. Goulding-Hotta, G. Venkatesh, J. Babb, V. Bhatt, S. Swanson and M. B. Taylor: “An evaluation of selective depipelining for FPGA-based energy-reducing irregular code coprocessors”, FPL ’11: Proceedings of the 2011 International conference on Field programmable logic and applications, pp. 24–29 (2011).

[6] K. Van Craeynest, A. Jaleel, L. Eeckhout, P. Narvaez and J. Emer: “Scheduling heterogeneous multi-cores through Performance Impact Estimation (PIE)”, ISCA ’12: Proceedings of the 39th annual international symposium on Computer architecture, pp. 213–224 (2012).

[7] S. Padmanabha, A. Lukefahr, R. Das and S. Mahlke: “Trace based phase prediction for tightly-coupled heterogeneous cores”, MICRO 46: Proceedings of the 46th annual IEEE/ACM international symposium on Microarchitecture, pp. 445–456 (2013).

[8] S. Padmanabha, A. Lukefahr, R. Das and S. Mahlke: “DynaMOS: dynamic schedule migration for heterogeneous cores”, MICRO ’15: Proceedings of the 48th International Symposium on Microarchitecture, pp. 322–333 (2015).

[9] T. S. Muthukaruppan, A. Pathania and T. Mitra: “Price theory based power management for heterogeneous multi-cores”, ASPLOS ’14: Proceedings of the 19th international conference on Architectural support for programming languages and operating systems, pp. 161–176 (2014).

[10] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan and D. M. Tullsen: “Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction”, MICRO 36: Proceedings of the 36th annual IEEE/ ACM international symposium on Microarchitecture, pp. 81–92 (2003).

[11] M. Becchi and P. Crowley: “Dynamic thread assignment on heterogeneous multiprocessor architectures”, CF ’06: Proceedings of the 3rd conference on Computing frontiers, pp. 29–39 (2006).

[12] J. Cong and B. Yuan: “Energy-efficient scheduling on heterogeneous multi-core architectures”, ISLPED ’12: Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design, pp. 345–350 (2012).

[13] J. Chen and L. K. John: “Efficient program scheduling for heterogeneous multi-core processors”, DAC ’09: Proceedings of the 46th annual Design automation conference, pp. 927–930 (2009).

[14] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi and K. I. Farkas: “Single-ISA heterogeneous multi-core architectures for multithreaded workload performance”, ISCA ’04: Proceedings of the 31st annual international symposium on Computer architecture, pp. 64–75 (2004).

[15] D. Shelepov, J. C. S. Alcaide, J. S. acey, A. Fedorova, N. Perez, Z. F. Huang, S. Blagodurov and V. Kumar: “HASS: a scheduler for heterogeneous multi-core systems”, ACM SIGOPS Operating Systems Review, **43**, 2, pp. 66–75 (2009).

[16] ARM Ltd.: “ARM unveils its most energy efficient application processor ever; redefines traditional power and performance relationship with big.LITTLE processing”, Internet:www.arm.com/about/newsroom/arm-unveils-its-most-energy-efficient-application-processor-ever-with-biglittle-processing.php (Oct. 19, 2011 [Nov. 16, 2014]).

[17] ARM Ltd.: “big.LITTLE technology”, Internet:www.arm.com/products/processors/technologies/
biglittleprocessing.php (2014 [Nov. 13, 2014]).

[18] T. Nowatzki and K. Sankaralingam: “Analyzing behavior specialized acceleration”, ASPLOS ’16: Proceedings of the 21st international conference on Architectural support for programming languages and operating systems, pp. 697–711 (2016).

[19] T. Chen, Q. Guo, K. Tang, O. Temam, Z. Xu, Z.-H. Zhou and Y. Chen: “ArchRanker: A ranking approach to design space exploration”, ISCA ’14: Proceeding of the 41st Annual International Symposium on Computer Architecuture, pp. 85–96 (2014).

[20] B. C. L. M. Guevara, B. Lubin: “Strategies for anticipating risk in heterogeneous system design”, HPCA ’14: Proceedings of the 20th International Symposium on High Performance Computer Architecture, pp. 154–164 (2014).

[21] E. Tomusk and C. Dubach: “Diversity: A design goal for heterogeneous processors”, IEEE Computer Architecture Letters, **15**, 2, pp. 81–84 (2016).

[22] E. Tomusk, C. Dubach and M. O’boyle: “Selecting heterogeneous cores for diversity”, ACM Trans. Archit. Code Optim., **13**, 4, pp. 49:1–49:25 (2016).

[23] Onikiri: “Onikiri”, Internet:www.mtl.t.u-tokyo.ac.jp/ onikiri2/wiki/index.php?Onikiri (Jun. 18, 2013 [May 11, 2015]).

[24] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen and N. P. Jouppi: “McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures”, MICRO 42: Proceedings of the

42nd annual IEEE/ACM international symposium on Microarchitecture, pp. 469–480 (2009).

[25] J. L. Henning: “SPEC CPU2006 benchmark descriptions”, ACM SIGARCH Computer Architecture News, **34**, 4, pp. 1–17 (2006).

[26] J. MacQueen: “Some methods for classification and analysis of multivariate observations”, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, Berkeley, Calif., University of California Press, pp. 281–297 (1967).

発表文献

[1] 澁谷 俊憲, 三輪 忍, 塩谷 亮太, 佐々木 広, 八巻 隼人, 本多 弘樹, ヘテロジニアス・プロセッサの設計探索手法の初期検討, 情報処理学会報告 2016-ARC-221, No.26, pp.1-7, (2016).

図一覧

表一覧