ヘテロジニアス・プロセッサの設計探索手法の初期検討

澁谷 俊憲[1,a)]　三輪 忍[1]　塩谷 亮太[2]　佐々木 広[3]　八巻 隼人[1]　本多 弘樹[1]

概要：本稿は，ヘテロジニアス・プロセッサの設計探索手法，すなわち，ヘテロジニアス・プロセッサを構成する各コアのアーキテクチャを決定する手法についての初期検討を行うものである．ホモジニアス・プロセッサの設計探索とは異なり，ヘテロジニアス・プロセッサの設計探索では探索空間が搭載するコア数に対して指数関数的に増大するため，探索時間もコア数に対して指数関数的に増大する．そこで我々は，搭載するコア数によらず計算時間がほぼ一定な，ヘテロジニアス・プロセッサの設計探索手法を現在検討している．本稿では，上記の設計探索手法について述べるとともに，シミュレーション実験を通して明らかになった現在の手法の問題点について述べる．

1. はじめに

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 [2], [3], [5], [8], [14], [15], [19], [20], [21], [25], [26], [28], [29], [30], [31]．ヘテロジニアス・プロセッサは，1 つのチップ上に，電力効率が異なる複数のコア（あるいはアクセラレータ）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，実行されるプログラムのフェーズごとに，そのフェーズを最も高い電力効率で実行するコアをチップ内から選び出し，選び出されたコアで当該フェーズを実行することによってプロセッサの消費エネルギーを削減する．ヘテロジニアス・プロセッサの実例としては，ARM 社が開発した big.LITTLE アーキテクチャが知られている [1], [8]．big.LITTLE は，アウト・オブ・オーダ実行を行う big コアとインオーダ実行を行う LITTLE コアの 2 種類のコアからなるヘテロジニアス・プロセッサである．

ヘテロジニアス・プロセッサによる消費エネルギーの削減効果は，チップ上に搭載するコアの組み合わせに強く依存する．そのため，ヘテロジニアス・プロセッサの設計においては，どのような構成のコアをチップに搭載するかを決定することが重要な課題である．コアを汎用コアにするかアクセラレータにするか，実行方式をアウト・オブ・オーダにするかインオーダにするか，命令発行幅やキャッシュ・サイズ等のパラメータをいくつにするか，などを決定しなければならない．このような問題は一般に設計探索問題と呼ばれており，ホモジニアス・プロセッサの設計探索問題に対しては多くの研究事例が存在する [4], [6], [7], [9], [10], [13], [17], [27]．しかしながら，ヘテロジニアス・プロセッサの設計探索問題はほとんど研究が行われていない．

ヘテロジニアス・プロセッサの設計探索は，ホモジニアス・プロセッサの設計探索と比べて，探索空間が広大であり，その結果，探索に必要な時間も膨大なものとなる．以下の説明では，1 つのコアが取り得る構成の数を N，プロセッサに搭載するコア数を k とする．ホモジニアス・プロセッサの設計探索では，N 種類のコアの中から最適な構成のコアをまず 1 つ発見し，あとはそのコアを k 個複製すればよいため，探索点の数は $O(N)$ となる．一方，ヘテロジニアス・プロセッサの設計探索では，N 種類のコアの中から k 個のコアの最適な組み合わせを選ぶため，探索点の数は ${}_NC_k$，すなわち，$O(N^k)$ となる．例えば，5 コアのヘテロジニアス・プロセッサ [23] の設計探索を行う場合は，$O(N^5)$ もの探索空間の中から最適な設計パラメータを発見しなければならない．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と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 $O(N^k)$ となる．

そこで本研究では，$O(N)$ の計算時間でヘテロジニアス・プロセッサの設計探索を行う手法を検討する．k コアのヘテロジニアス・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．上述のように，この組み合わせの総数が $O(N^k)$ であることから，シミュレー

[1] 電気通信大学
[2] 名古屋大学
[3] コロンビア大学
[a)] shibuya@hpc.is.uec.ac.jp

ション回数は通常 $O(N^k)$ となる．それに対し，現在検討中の手法では，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ておく．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，上述のようにして得られた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．

プロセッサの設計探索において最も時間が必要とされるのは，シミュレーションによってプロセッサの性能や消費エネルギーを評価する部分である．我々の手法は，ヘテロジニアス・プロセッサの設計探索におけるシミュレーション回数を $O(N^k)$ から $O(N)$ に削減することによって，設計探索に必要な時間を $O(N)$ に削減することが期待される．

本論文の構成は以下のようになっている．まず 2 章では関連研究を紹介する．続く 3 章では現在検討中のヘテロジニアス・プロセッサの設計探索手法を詳しく説明する．4 章で手法の評価を行い，5 章で本論文をまとめる．

2. 関連研究

前章で述べたように，プロセッサの設計探索において多くの時間を必要とするのは，各探索点（コア構成）の性能や消費エネルギーを評価するためのシミュレーションである．このシミュレーションの総回数をいかにして減らすかが設計探索では重要であり，そのための手法がいくつか提案されている．本章ではこれらの手法についてまとめる．

2.1 モデル式を利用する手法

Joseph らは，設計パラメータを変更した時のシングル・コアの性能を，線形回帰モデルを用いて予測する手法を提案している [12]．この手法では，パイプライン段数，キャッシュ・サイズ，ROB サイズなどの各種設計パラメータを説明変数とし，その構成のコアでプログラムを実行した時の性能を目的変数とする線形回帰モデルを作成する．線形回帰モデルの学習は，探索空間内の一部の探索点についてシミュレーションを行うことで学習データ（設計パラメータとコア性能の組）を生成し，これらの学習データを使用して行う．この手法はシミュレーション回数を学習データ数にまで削減できるが，設計パラメータに対して非線形な振る舞いをするコアの性能を正しく予測できないという問題がある．

上記の問題点に対し，ニューラル・ネットワークやスプライン関数を用いてシングル・コアの性能を予測する手法が提案されている [10], [11], [16]．これらの手法では，線形回帰モデルの代わりに，設計パラメータを入力変数とし，コア性能を出力変数とするニューラル・ネットワークやスプライン関数を作成する．線形回帰モデルの学習と同様，ニューラル・ネットワークやスプライン関数の学習は，少数の探索点をシミュレーションすることによって得た学習データを用いて行う．これらの手法は線形回帰モデルを用いる手法よりも高い精度でコア性能を予測できるが，予測精度はまだ十分とは言えない．

Chen らは，シングル・コアの設計探索手法として ArchRanker を提案している [4]．この手法では，設計パラメータを変更した時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（ランク）を判定するモデルを作成する．性能と消費電力それぞれについて，少数の探索点（探索空間の 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づけを行うモデルを学習し，これら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し出す．ArchRanker は，上述のコア性能の予測モデルを用いた手法よりも優れた設計パラメータを発見できるが，ヘテロジニアス・プロセッサの設計探索を行うための手法ではない．

2.2 依存グラフを利用する手法

Lee らは，マイクロアーキテクチャの依存グラフを用いてシングル・コアの設計探索を行う手法を提案している [17]．この手法では，キャッシュ・ミス，分岐予測ミスなどの命令パイプラインを乱すイベントに着目し，設計パラメータを変更した時のコア性能を，依存グラフを解析することにより求める．依存グラフはプログラムをある 1 つの構成のコアで実行した時の実行トレースから生成できるため，この手法の総シミュレーション回数は総プログラム数に等しくなる．しかし，この手法はプログラムの実行中にコア・アーキテクチャを変更することを想定しておらず，ヘテロジニアス・プロセッサの性能予測には利用できない．

依存グラフをヘテロジニアス・プロセッサの性能と電力の見積もりに応用した手法も提案されている [22], [23]．この手法では，プログラム内のループをアクセラレータで実行した場合の性能と電力を，依存グラフの該当する部分を変形することによって計算する．文献 [23] では，この手法を用いて，1 つの汎用コアと最大 4 つのアクセラレータからなるヘテロジニアス・プロセッサにおいて，汎用コアの構成とアクセラレータ数を変更した時の性能と電力の見積もりを行っている．この手法に必要なシミュレーション回数は，Lee らの手法と同様，総プログラム数である．この手法はヘテロジニアス・プロセッサの設計探索に要する時間を大幅に短縮できるが，性能と電力の見積もり誤差はそれぞれ最悪 15% と大きい．

3. 提案手法

我々の目的は，1 つのコアが取り得るパラメータの組み合わせを N とすると，ヘテロジニアス・プロセッサの設計

Algorithm 1 区間ごとのコア候補の取り出し

Input: N //1 つのコアの取り得るパラメータ数
Input: β //区間ごとに取り出すコアの数
Input: $\boldsymbol{C}_i$ //N 種のコアの i 命令区間目の実行サイクルのトレース
Input: $\boldsymbol{E}_i$ //N 種のコアの i 命令区間目の消費エネルギのトレース
  for all i **do**
    Sort ($\boldsymbol{C}_i$) //昇順
    $\boldsymbol{L}^i_{cycle} \leftarrow$ each core on $\boldsymbol{C}_i < \boldsymbol{C}_i[0] \times (1+\alpha)\%$ //α は性能制約
    Sort ($\boldsymbol{E}_i$ on $\boldsymbol{L}^i_{cycle}$) //昇順
    $\boldsymbol{E}_{interval} \leftarrow \boldsymbol{L}^i_{cycle}[0]$ to $\boldsymbol{L}^i_{cycle}[\beta]$
  end for
Output: 命令区間ごとの $\boldsymbol{E}_{interval}$

Algorithm 2 搭載するコアの選出

Input: 命令区間ごとの $\boldsymbol{E}_{interval}$ //Algo.1 の出力
Input: k //マルチコアに搭載するコア数
  for $i = 1$ to k **do**
    while each interval **do**
      if $\boldsymbol{L}_{output}$has cores in one of$\boldsymbol{E}_{interval}$ **then**
        continue
      else
        Count in $\boldsymbol{E}_{interval}$'s cores
      end if
    end while
    $\boldsymbol{L}_{output} \leftarrow$ Most counted core
    Reset count
  end for
Output: $\boldsymbol{L}_{output} = \mathrm{core}_1, \ldots, \mathrm{core}_k$

図 1: 探索方法の模式図

探索に要する時間を $O(N)$ にまで削減することである．そのために，以下の手法を初期検討として提案する．

提案手法では，まず候補となる N 種類のコアそれぞれについて，各コアをシングル・コア・プロセッサとみなしてベンチマーク・プログラムを実行した時の，一定命令区間ごとの実行サイクル数と消費エネルギーを評価する．命令区間の長さは，設計後のヘテロジニアス・プロセッサにおけるコアの切替え間隔と等しくなるように定める．コアの区間ごとの性能と消費エネルギーを評価するためには，通常，プロセッサ・シミュレーションが必要であり，N 種類のコアの評価に必要なシミュレーションの総回数は $O(N)$ となる．この結果，各コアに対応したサイクル数と消費エネルギーのトレース（計 N 本）が得られる．

以降は，上記のトレースを用いて，設計探索の目的関数に応じた k 個（k は搭載コア数）のコアを選ぶ．コア選択の過程では，シミュレーションは不要である．以下の説明では，目的関数を「最速コアから最大 α%の性能低下を許す範囲内で消費エネルギーを最小化する」とする．

提案手法の前段部分をアルゴリズム 1 に示す．まずは各コアのサイクル数のトレースを使用して，命令区間ごとにコアを速い順にソートする．**図 1** 上の例では，区間 1 は速い順にコア $A, B, F, D, E \cdots$，区間 2 はコア $B, C, A, G, H \cdots$ となっている．このような，区間ごとに速い順にソートしたコアのリストをまずは作成する．

次いで，命令区間ごとに，最速のコアから α%以上の性能低下を示すコアはその区間を実行する機会がないものと見なし，上記のリストから取り除く．図 1 の例では，区間 1 の最速コアは A であり，コア A から α% の性能低下の範囲内にあるコアは 4 番目のコア D までであるとして，E 以下のコアをリストから削除する．同様に，区間 2 の最速コア B に対し，α% を超える性能低下を示すコア（H 以下のコア）をリストから削除する．

上記のようにして得たコアのリストを，今度は，区間ごとに消費エネルギーの小さい順にソートし直す．図 1 下の例では，区間 1 のコア A, B, F, D を消費エネルギーの小さい順に並び変えた結果，F, B, A, D の順になったものとしている．同様に，区間 2 は消費エネルギーの小さい順に B, A, G, C となっている．この結果，区間ごとに，性能制約を満たすコアの中で，消費エネルギーの小さい順に並んだコアのリストを得ることができる．

最後に，上記のコアのリストを用いて，k 種類のコアを選択する (アルゴリズム 2)．コアの選択は，なるべく多くの命令区間の実行をカバーするように行う．

まず最初に，コアごとに各命令区間を眺めていき，消費エネルギーの小さい順で β 番目以内に入る区間数をカウ

ントする．ここで β はチューニング・パラメータであり，図 1 下は $\beta = 3$ の例である．この例では，コア A は 5 回（区間 1, 2, 3, 5, 7），コア B は 3 回（区間 1, 2, 7），β 以内に入っている．このようにして，コアごとに消費エネルギーが相対的に少なかった区間数をカウントし，この数が最も多いコアを 1 番目のコア（図 1 ではコア A）として選択する．そして，以降のコア選択のために，上記のリストからコア A がカバーする区間（図 1 では区間 1, 2, 3, 5, 7）の行を削除する．

2 番目以降のコアの選択は，上述の区間が削除されたリストを用いて，1 番目のコアを選択した時と同様の処理を行う．すなわち，区間 1, ,2 ,3, 5, 7 の行が削除されたリストを用いて，消費エネルギーの小さい順で β 以内に入る区間数をコアごとにカウントし，この数が最も大きなコアを 2 番目のコアとして選択する．そして，選択されたコアがカバーする区間の行をリストから削除する．

上記の処理は，k 番目のコアが選択されたら終了する．このアルゴリズムにより，1) 区間ごとの性能制約を満たし，2) 区間ごとの消費エネルギーが相対的に小さく，3) プログラム中の命令区間をより多くカバーするコアの組み合わせが選択できる．このようにして選出されたコアの組み合わせが，プログラム全体の性能制約を満たし，かつ，消費エネルギーが小さなコアの組み合わせになっていることを期待する．

上記の説明では，N コア分の区間ごとの実行サイクル数と消費エネルギーを求めるために $O(N)$ 回のシミュレーションを行うことを想定していたが，このシミュレーション回数はさらに削減できる．設計パラメータを変更した時の区間ごとの実行サイクル数と消費エネルギーをシミュレーションで求める代わりに，2.1 節で述べた手法と同様，実行サイクル数と消費エネルギーそれぞれを予測するモデル式を構築して予測すればよい．モデル式を利用する場合は，総シミュレーション回数はモデル式の学習に用いる学習データ数に抑制できる．

上記のアルゴリズムは，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時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が選ばれた後で 1 回だけ行えばよく，このシミュレーションに要する時間は，探索時間全体と比べて問題にならない程小さいと考えている．

4. 実験および結果

4.1 目的関数

本実験では，設計探索の目的関数を「N 種類のコアの中から 1 つを選び，あるプログラムの集合を実行した場合の最速コアの実行サイクルを $T_{fastest}$ とする．$T_{fastest}$ から α% の実行サイクルの増加を許す範囲内で，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」ことと定義する．α はプロセッサの要求仕様によって定まるパラメータであり，設計者によって与えられるものとする．

今回は，ある性能制約が与えられた時の消費エネルギーの最小化を目的関数としたが，提案手法は別の目的関数の設計探索にも応用できると考えている．例えば，提案手法は，ある消費エネルギー制約が与えられた時に，その制約を満たす範囲で性能を最大化するコアの組み合わせを求める場合にも利用できる．

4.2 実験環境

今回の実験で使用した設計パラメータと各パラメータの探索範囲を**表 1** にまとめる．実行方式はアウト・オブ・オーダとインオーダの 2 種類とする．それ以外の設計パラメータとして，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，それぞれ 3 ∼ 4 通り変化させる．したがって，今回の実験におけるコアの種類数 N は $1,944 (= 2 \times 3 \times 4 \times 3 \times 3 \times 3 \times 3)$ となる．これら 1,944 種類のコアを組み合わせて，2 コアからなるヘテロジニアス・プロセッサを設計するものとする．したがって，今回の実験の探索空間は $1,888,596 (= {}_{1944}C_2)$ 点からなる．この探索空間において，提案手法により選出された組み合わせを全探索による理想の組み合わせと比較する．

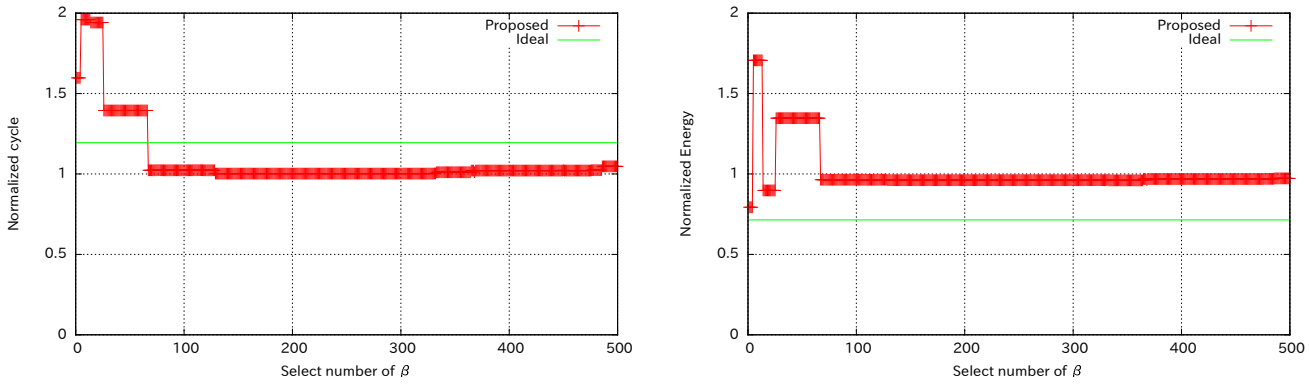
プロセッサの性能評価には Onikiri-2 [24] を使用した．また，消費電力の評価には McPAT-1.3 [18] を使用した．ベンチマークには，SPECint 2006 の中から，big.LITTLE による消費エネルギーの削減効果が高い 3 つのプログラム（`401.bzip2`, `445.gobmk`, `483.xalancbmk`）を使用した．性能制約 α は 20 とした．

コア切替えの判断は，ある一定の命令数がコミットされる度に行われるものとする．今回は，文献 [19] と同様，1,000 命令がコミットされる度に切替えの判断を行うものとした．

今回の実験では，簡単のため，コアの切替えにともなう時間オーバヘッドとエネルギーオーバヘッドは 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/

表 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	OoO, Inorder	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
L2$サイズ	256KB, 1024KB, 4096KB	3
GShare サイズ	2KB, 8KB, 32KB	3



(a) 実行サイクル数　　(b) 消費エネルギー

図 2: 提案手法により発見されたヘテロジニアス・プロセッサの性能と消費エネルギー（各グラフの縦軸は最速コアの性能と消費エネルギーで正規化）

再開に要する時間/エネルギーだけでなく，コアの切替えによって追加発生するキャッシュ・ミスや分岐予測ミスも含まれる．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．

4.3 結果

区間ごとに取り出すコアの数 β を 1 から 500 の間で変更した場合に，選出されたヘテロジニアス・プロセッサが示す実行サイクル，消費エネルギーのシミュレート結果を**図 2a**，**図 2b** に示す．このグラフでは 1,944 種類のコアがシングル・コアとしてベンチマークを実行した際に最も高速に処理できたコアの実行サイクル，消費エネルギーに正規化されている．図内の線は，${}_{1944}C_2 = 1,888,596$ 種類全ての組み合わせの中で，理想的な出力結果を持つ組み合わせ，つまり，シングル・コア実行の最速コアより 20% の性能低下の範囲内で，消費エネルギーが最小だったヘテロジニアス・プロセッサが示す実行サイクル，消費エネルギーである．この理想的な組み合わせの結果は最も高速に処理できたコアに対して 19%性能低下を示しており，約 29%のエネルギー削減効果があった．

グラフより，提案手法では，β が小さい場合は $\alpha = 20\%$ の性能制約をオーバーするものの，β が 70 を超えたあたりから性能制約を満たすようになり，その後 β を増やしても選出されるコアの組み合わせに大きな変化は見られなかった．β が 70 以上の探索結果の中で，消費エネルギーが最小となったヘテロジニアス・プロセッサのパラメータを**表 2** に示す．このヘテロジニアス・プロセッサは，最も高速に処理できたコアに対し，約 4%のエネルギー削減となり，1%の性能低下を示した．また，2 つともアウト・オブ・オーダのコアが選出された．

提案手法では，β を大きくするほど 1 つのコアがカバーする区間数が増加し，その結果，選出されたコアが全命令区間数をカバーする割合が上昇する．$\beta = 100$ において選出された 2 つのコアによってカバーされる区間の割合は，全命令区間数の 47% であった．

表 2: 選出されたコアのパラメータ

	提案手法		理想	
要素	コア 1	コア 2	コア 1	コア 2
実行方式	OoO	OoO	OoO	Inorder
フェッチ/コミット幅	4	8	8	2
ALU の個数	4	4	2	2
L1I$サイズ	32KB	32KB	32KB	32KB
L1D$サイズ	32KB	32KB	32KB	2KB
L2$サイズ	4096KB	1024KB	4096KB	4096KB
GShare サイズ	32K	8K	32	32

4.4 考察

提案手法によって選出したコアがエネルギーをほとんど削減できていないのは，選出されたコアが 2 つともアウト・オブ・オーダのコアだったことが大きな要因である．

提案手法では 1 つの命令区間内で上位 β 以内に含まれるとエネルギーの順位にかかわらず 1 回とカウントされる．各区間において，インオーダが得意とする区間では上位 β の低い順位にアウト・オブ・オーダが含まれることはしばしばあるが，アウト・オブ・オーダが得意とする区間では，速度の制約によりインオーダが β 以内に含まれることは少ない．そのため，インオーダが得意とする区間でもアウト・オブ・オーダが順位にかかわらず等しくカウントされた結果，カウントされるインオーダが少なくなったと考えられる．

5. おわりに

本稿では，N 種類のコアを探索対象とするヘテロジニアス・プロセッサの設計探索問題において，$O(N)$ 回分のシミュレーション結果から最適に近いコアの組み合わせを導出する手法を提案した．本実験の条件下では，提案手法によって発見されたコアの組み合わせは，全探索によって発

見できる理想的なコアの組み合わせと比べて，消費エネルギーが 25 ポイント悪化する結果が得られた．

今回は一定命令区間ごとに測定されたトレースを用いて計算しているが，1 つの命令区間の全実行サイクル数に対する寄与率は区間によって異なる．今回提案したアルゴリズムでは，消費エネルギーの小さい順で β 以内に入る区間数の大小に基づいてコアを選択しており，上記の寄与率は勘案されていない．提案アルゴリズムによって発見できる解を最適解に近づけるためには，各区間におけるサイクル数の差異で何らかの重み付けをする必要があると考えられる．

また，今回提案したアルゴリズムでは，消費エネルギーの小さい順で β 以下のコアの情報を切り捨てている一方，β 以内のコアの価値を等しいとみなしている．この点についても何らかの重み付けをする必要があると考えられる．

謝辞 本研究の一部は JST CREST による．

参考文献

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

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

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

[4] Chen, T., Guo, Q., Tang, K., Temam, O., Xu, Z., Zhou, Z.-H. and Chen, Y.: ArchRanker: A Ranking Approach to Design Space Exploration, *ISCA '14: Proceeding of the 41st Annual International Symposium on Computer Architecuture*, pp. 85–96 (2014).

[5] Cong, J. and Yuan, B.: 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).

[6] Dubach, C., Jones, T. and O'Boyle, M.: Microarchitectural Design Space Exploration Using an Architecture-Centric Approach, *MICRO '07: Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 262–271 (2007).

[7] Genbrugge, D. and Eeckhout, L.: Chip Multiprocessor Design Space Exploration through Statistical Simulation, *IEEE Transactions on Computers*, Vol. 58, No. 12, pp. 1668–1681 (2009).

[8] Greenhalgh, P.: big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7, white paper (2011).

[9] Guo, Q., Chen, T., Chen, Y., Zhou, Z.-H., Hu, W. and Xu, Z.: Effective and Efficient Microprocessor Design Space Exploration Using Unlabeled Design Configurations, *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, pp. 1671–1677 (2011).

[10] İpek, E., McKee, S. A., Caruana, R., de Supinski, B. R. and Schulz, M.: Efficiently Exploring Architectural Design Spaces via Predictive Modeling, *ASPLOS '06: Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 195–206 (2006).

[11] Joseph, P. J., Vaswani, K. and Thazhuthaveetil, M. J.: A Predictive Performance Model for Superscalar Processors, *MICRO '06: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 161–170 (2006).

[12] Joseph, P., null Kapil Vaswani and Thazhuthaveetil, M.: Construction and use of linear regression models for processor performance analysis, *HPCA '06: Proceedings of the 12th International Symposium on High Performance Computer Architecture*, pp. 99–108 (2006).

[13] Khan, S., Xekalakis, P., Cavazos, J. and Cintra, M.: Using PredictiveModeling for Cross-Program Design Space Exploration in Multicore Systems, *PACT '07: Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques*, pp. 327–338 (2007).

[14] 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, *MICRO 36: Proceedings of the 36th annual IEEE/ACM international symposium on Microarchitecture*, pp. 81–92 (2003).

[15] Kumar, R., Tullsen, D. M., Ranganathan, P., Jouppi, N. P. and Farkas, K. I.: 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).

[16] Lee, B. C. and Brooks, D. M.: Accurate and Efficient Regression Modeling for Microarchitectural Performance and Power Prediction, *ASPLOS '12: Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 185–194 (2006).

[17] Lee, J., Jang, H. and Kim, J.: RpStacks: Fast and Accurate Processor Design Space Exploration Using Representative Stall-Event Stacks, *MICRO '14: Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 255–267 (2014).

[18] Li, S., Ahn, J. H., Strong, R. D., Brockman, J. B., Tullsen, D. M. and Jouppi, N. P.: 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).

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

[20] Muthukaruppan, T. S., Pathania, A. and Mitra, T.: 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).

[21] Navada, S., Choudhary, N. K., Wadhavkar, S. V. and Rotenberg, E.: 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).

[22] Nowatzki, T., Govindaraju, V. and Sankaralingam, K.: A Graph-Based Program Representation for Analyzing Hardware Specialization Approaches, *IEEE Computer Architecture Letters*, Vol. 14, No. 2, pp. 94–98 (2016).

[23] Nowatzki, T. and Sankaralingam, K.: 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).

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

[25] Padmanabha, S., Lukefahr, A., Das, R. and Mahlke, S.: 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).

[26] Padmanabha, S., Lukefahr, A., Das, R. and Mahlke, S.: DynaMOS: Dynamic Schedule Migration for Heterogeneous Cores, *MICRO '15: Proceedings of the 48th International Symposium on Microarchitecture*, pp. 322–333 (2015).

[27] Palermo, G., Silvano, C. and Zaccaria, V.: ReSPIR: A Response Surface-based Pareto Iterative Refinement for Application-specific Design Space Exploration, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 28, No. 12, pp. 1816–1829 (2009).

[28] Pricopi, M., Muthukaruppan, T. S., Venkataramani, V., Mitra, T. and Vishin, S.: 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).

[29] Sampson, J., Arora, M., Goulding-Hotta, N., Venkatesh, G., Babb, J., Bhatt, V., Swanson, S. and Taylor, M. B.: 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).

[30] Shelepov, D., Alcaide, J. C. S., acey, J. S., Fedorova, A., Perez, N., Huang, Z. F., Blagodurov, S. and Kumar, V.: HASS: a scheduler for heterogeneous multicore systems, *ACM SIGOPS Operating Systems Review*, Vol. 43, No. 2, pp. 66–75 (2009).

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