Vol.2014-ARC-212 No.12
2014/10/6

回路資源の投入により電力効率を改善するプロセッサ・アーキテクチャ

三輪 忍[1,a)]　塩谷 亮太[2]　佐々木 広[3]

概要：
LSI の微細化により，チップ上の回路資源は今後も増加すると予想されている．我々は，この豊富に存在する回路資源を利用した，シングル・スレッド実行時の電力効率を改善する手法を提案している．提案手法では，ループ実行に特化したコアなどのさまざまなタイプのコアを多数用意しておき，出現する命令列に応じて電力効率に優れたコアに切り替えながらプログラムを実行することで，シングル・スレッド実行時のプロセッサの電力効率を改善する．今回，提案アーキテクチャの詳細について検討を進めるとともに，予備評価として理想的な状況における提案手法の効果を測定した．本稿ではこれらについて報告する．

1. はじめに

プロセッサ・チップ上の回路資源量は依然として増加の一途を続けている．2000 年代半ばには既にムーアの法則の終焉が予想されており，早ければ 2015 年頃にはそれが訪れると考えられていた [19]．しかし，FinFET や High-K メタル・ゲート・トランジスタなどの新しい半導体技術の登場により，LSI の微細化は今なお続いている．ITRS ロードマップによれば，LSI の微細化技術は，2025 年頃にサブ 10nm プロセスの実現を視野に開発が進められている [8]．したがって，ムーアの法則が終わるまでにはまだあと 10 年程の余裕があり，チップ上の回路資源は今後も増え続けると予想される．

一方，汎用プロセッサにおけるシングル・スレッド実行時の電力効率の改善は，プロセッサ・アーキテクトにとって重要な課題の 1 つである．携帯情報端末においては，電力効率に優れたコアを使用することで性能を維持したままチップ全体の消費電力を抑制でき，ひいてはバッテリの駆動時間を延長できる．サーバにおいては，電力効率に優れたコアを使用することで TDP を維持したままコア数を増やすことができ，その結果，より多くのスレッドを並列に処理できるようになる．このように，あらゆるクラスのプロセッサにおいてシングル・スレッド実行の電力効率の改善が求められている．

このような背景のもと，我々は，回路資源の投入によりシングル・スレッド実行の電力効率を改善するプロセッサを提案している [23]．提案手法では，整数系パイプのみからなるコアやループ実行に特化したコアなど，汎用性を持たないコアも含めてさまざまなタイプのコアをチップ上に搭載し，出現する命令列に応じて使用コアを切り替えながらプログラムを実行する．整数系命令のみから構成された命令列など，プログラム中には汎用性を持たないコアでも実行可能な命令列が存在する．そうした命令列を汎用コアよりも電力効率の面で優れたコア上で実行することによって，シングル・スレッド実行時の電力効率を改善する．

提案手法では，チップ上に存在するさまざまなコアが常に排他的に利用される．実行に使用するコア以外のコアは，クロック/パワー・ゲーティングなどの方法により，その消費電力を抑制する．このように，提案手法は稼働するトランジスタが 1 つのコアに制限されていることから，ダーク・シリコン [3], [4], [9], [17], [18], [21], [22] を有効利用した手法，とみなすこともできる．

我々の前回の報告では，提案手法の構想を述べるにとどまっており，アーキテクチャの詳細な検討や予備評価は行っていなかった [23]．そこで今回，アーキテクチャの検討を進めるとともに，予備評価として理想的にコアを切り替えることができたと仮定した場合の提案手法の効果を測定した．本稿ではこれらについて報告する．

本稿の構成は以下の通りである．まず次章にて，回路規

[1] 東京大学
The University of Tokyo
[2] 名古屋大学
Nagoya University
[3] コロンビア大学
Columbia University
[a)] miwa@hal.ipc.i.u-tokyo.ac.jp

模の増加をともなう，シングル・スレッド実行の電力効率を改善する既存手法についてまとめる．続く 3 章では提案手法について詳しく述べ，4 章で予備評価の方法と結果について述べる．最後 5 章でまとめと今後の課題を述べる．

2. 関連研究

我々の知る限り，回路資源の投入によってシングル・スレッド実行の電力効率を改善することを明確に主張した論文は存在しない．しかし，回路規模の増加と引き換えにシングル・スレッド実行の電力効率を改善した手法はいくつか存在する．本章ではそれら既存研究との違いを述べる．

ARM 社の big.LITTLE は，電力/性能の異なる 2 種類のコア（big コア/LITTLE コア）を同一チップ上に搭載し，プログラムのフェーズに応じてそれらを切り替えながら実行することで消費電力を削減する手法である [5], [14], [20]. big コアには out-of-order 実行を行う Cortex-A15/A57, LITTLE コアには inorder 実行の Cortex-A7/A53 が用いられる．big.LITTLE はもともと単体で製品化されているコアを 2 つ搭載しているため，ラスト・レベル・キャッシュは 2 種類のコアそれぞれが独立に有している．そのため，コアの切り替えにともなう性能/エネルギ・オーバヘッドが大きいという問題を抱えており，プログラム中の細かなフェーズに応じてコアの切り替えを行うのが難しい．

Lukefahr らは， big/little マイクロ・エンジンと呼ばれる 2 種類の実行系を有した，Composite Core というアーキテクチャを提案している [12]．命令/データ・キャッシュや命令フェッチ・ロジックは 2 種類のマイクロ・エンジンで共通であり，各マイクロ・エンジンは命令パイプライン上のデコード以降の処理を担当する．big マイクロ・エンジンは out-of-order 実行を行う高性能かつ高消費電力な実行系であるのに対し，little マイクロ・エンジンは inorder 実行を行う低性能かつ低消費電力な実行系となっている．レジスタ・ファイルは big/little それぞれが別々に有しており，レジスタ値は実行系を切り替える際にレジスタ間転送を行うことでコヒーレンスをとる．このように 2 つの実行系を密に結合することで，Composite Core では 1K 命令ごとと細粒度に実行系を切り替えることを可能にした．しかし，Composite Core が搭載する実行系は big/little の汎用エンジン 2 つだけであり，整数系命令のみから構成される命令列のような，汎用の実行系を必要としない命令列に対しては電力効率の改善効果が限定的である．

big.LITTLE ではコアが 2 種類であったが，より多くの種類のコアを用いてシングル・スレッド実行のエネルギ削減を図った研究もある．Navada らは，発行幅，レジスタ・ファイル・サイズ，命令キュー・サイズ，命令/データ・キャッシュ・サイズなどさまざまな資源の量が異なる 4 つの out-of-order コアからなる CMP を用いて，実行時にそれらを切り替えることで，シングル・スレッド実行時のプロセッサの消費エネルギを削減する手法を提案している [13]．ただし，Navada らの手法では，コアを切り替える粒度は 1M 命令ごととかなり粗い．また，彼らの手法では，チップ上に搭載するコアはすべて汎用の out-of-order コアであり，汎用コアを必要としない命令列に対する電力効率の改善効果はあまり高くない．

Venkatesh らは，Conservation Core と呼ばれる，プログラム中の特定の関数やループを実行する専用ハードウェア（コプロセッサ）をチップ上に多数搭載する手法を提案している [16], [21]．Conservation Core は，基本的には，関数やループと 1 対 1 に対応する形で設計されている．専用コンパイラがプログラムをコンパイルする際に Conservation Core で実行可能な関数やループが含まれているかを検出し，そのような部分コードが含まれていた場合はオフロード用のコードを挿入する．通常コアでの実行中にオフロード用のコードに到達すると，実行に必要なデータが転送され，通常コアが停止し Conservation Core による実行が開始される．

Conservation Core は，それが実行可能な命令列に対しては大幅に電力効率を改善するものの，実行可能な命令列が限られているという問題がある．よく似たコードを実行できるように Conservation Core を改良する試みも行われている [22] が，1 つのコアが実行可能なコードは命令の種類や出現順序，データ依存関係が相当酷似している必要がある．そのため，このようなアプローチによって電力効率を改善できる命令列は，依然として限定的である．

Guha らは，我々と同様，さまざまなタイプのコアを多数チップ上に搭載し，それらを切り替えながら実行することでシングル・スレッド実行時の電力効率を改善することを提案している [6]．しかし，彼らの研究では，チップに搭載するコアを決めるためのベンチマーク・プログラムの解析に比重が置かれており，具体的にどのようなコアをチップに搭載するかといったアーキテクチャに関する話はまだない．加えて，彼らのアプローチによって電力効率がどの程度改善するか，といった評価もまだ行われていない．

3. 提案手法

我々は，回路資源の投入によりシングル・スレッド実行の電力効率を改善するプロセッサを提案している [23]．以下，その詳細を述べる．

3.1 提案の概要

提案アーキテクチャの概要を**図 1** に示す．提案手法では，big.LITTLE などと同様，同じ ISA からなるヘテロジニアスなコアを 1 チップ上に搭載する．ただし，コア数は 2 つに限定しておらず，回路資源の許す限り，できるだけ多数のコアを搭載する．また，コアの種類も千差万別である．例えば，コア A は整数系 out-of-order コア，コア B

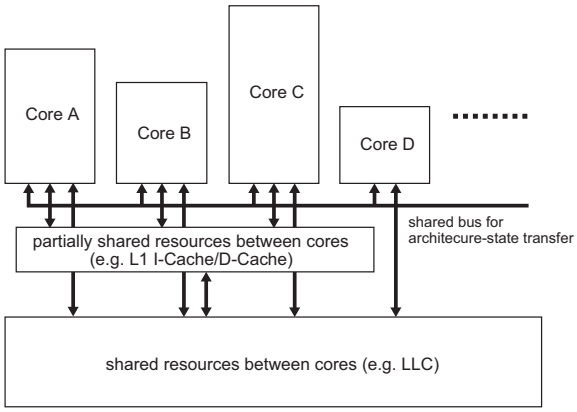

図 1 提案アーキテクチャの概要

は整数系 inoder コア，コア C は浮動小数点系コア，コア D はループ実行向けコア，のようにまったく異なる電力性能特性を有する．そうして，出現する命令列に応じてコアを切り替えながらプログラムを実行することにより，プロセッサの消費電力を削減する．

各コアは必ずしも任意の命令列を実行できる必要はない．例えば，「分岐を含まない数十の静的命令からなるループを実行するためのコア」のようなコアが存在してもよいし，「整数系パイプのみからなる out-of-order コア」のようなコアが存在してもよい．このようなコアでは実行可能な命令列はある程度制限されるが，実行可能な命令列が現れた場合は高い電力効率でそれを実行できる．ある程度機能を省いたコアも用意することで，大幅な消費電力の削減が見込める命令列に対して高い電力効率を実現する．

各コアは，基本的には，データ・キャッシュと命令フェッチ・ユニットを除くすべてのハードウェア資源を個別に有する．コアごとにこれらのハードウェア資源をカスタマイズすることにより，特定の命令列に対する電力効率を改善する．

提案アーキテクチャでは，プログラム・カウンタはコア間で共有するが，レジスタ・ファイルは各コアが有する．そのため，使用コアを切り替える際は，それまで使用していたコアから新しく使用するコアへと，生存中のレジスタ値を転送する必要がある．このレジスタ転送は，コアの切り替え時に全命令パイプラインを停止して行うことを考えている．最大で数十個のレジスタ値を転送する必要があるが，このレジスタ転送は時間的にも空間的にも並列に処理できることから，レジスタ転送によるパイプライン・ストールの影響はそれ程大きくないと考えている．

命令/データ・キャッシュ以下のすべてのキャッシュは，基本的には，全コアで共有する．これは，キャッシュをコアごとに所有すると，コアを切り替えた際に発生する追加のキャッシュ・ミスによる性能低下が無視できないと考えたためである．提案手法では，プログラム中の細かなフェーズに対応するために，100 命令程度の非常に短い時間でコアを切り替えることを想定している．このような短い時間間隔で L1 キャッシュを切り替えると，切り替えを行わなければ利用できたはずの時間的局所性が利用できなくなってしまう恐れがある．

ただし，後述するデータ・パス幅を狭くしたコアにおいては，通常のデータ・パス幅のデータ・キャッシュを利用するのは消費電力の面で無駄が多い．また，メモリ命令の発行数が他とは異なるコアについても，必要ポート数に応じたキャッシュを用意した方が電力効率はよいと予想している．そのため，これらのコアについては専用のデータ・キャッシュを設けることを考えている．これらのコアを利用する際には上述した追加のキャッシュ・ミスが発生するため，それが性能に与える影響を何らかの方法によって緩和する必要がある．その方法については今後詳しく検討する予定である．

各コアが有するハードウェア資源の中には ALU のような複数コアに共通の資源も存在するが，これらについては共有しない．これは，各コア内の回路遅延を，コアを個別に設計した場合と同程度に保つためである．また，このような構成をとることによって電源ドメインをコア単位とすることができ，それぞれのコアの電力管理が容易となる．

このように，提案アーキテクチャは冗長な機能を有するユニットがコアごとに存在するため，多くの回路資源を必要とする．ただし，そのコストは，本稿の冒頭で述べたように，LSI の微細化により償却できると考えている．

提案アーキテクチャがこれまでのヘテロジニアス・アーキテクチャと異なる点は，実行系の種類と実行系を切り替える時間粒度である．前章で述べたように，これまでのヘテロジニアス・アーキテクチャは各実行系が任意の命令列を実行することができ，また，その種類も 2~4 個と少なかった．さらに，実行系を切り替える時間間隔は 1K~1M 命令ごとと粒度が粗かった．それに対し，我々のアーキテクチャでは，1) 汎用コアだけでなくある程度機能が制限されたコアもチップ上に搭載し，2) それらを 100 命令程度の細かい時間粒度で切り替えながら実行することを考えている．多種多様なコアを細粒度に切り替えながらプログラムを実行することで，電力効率の更なる改善を狙う．

3.2 着眼点とコアの種類

SPEC CPU 2006 に含まれる 456.hmmer の部分コードを**図 2** に示す．図は，プログラムの先頭から数えて 1G と 17,000 命令ほど実行された後に実行されるコードである．コードは 10 個の静的命令から構成されており，末尾の分岐命令によって先頭の命令に戻るという小さなループ構造をしている．なお，図の命令列は全部で 2,000 命令ほど実行される．

図より，この命令列を実行するためには，汎用コアのハードウェア資源すべては必要ないことがわかる．出現す

```
 1: LOOP:
 2: (12000e4b0) r24 ← ldl r6
 3: (12000e4b4) r7 ← ldl r2
 4: (12000e4b8) r5 ← addl r5
 5: (12000e4bc) r2 ← lda r2
 6: (12000e4c0) r8 ← cmple r5, r23
 7: (12000e4c4) r6 ← lda r6
 8: (12000e4c8) r7 ← addl r24, r7
 9: (12000e4cc) r24 ← cmple r4, r7
10: (12000e4d0) r4 ← cmovne r24, r4, r7
11: (12000e4d4) bne r8 LOOP
```

図 2 456.hmmer の部分コード

る命令はすべて整数系であり，浮動小数点系パイプラインはまったく必要ない．また，出現する命令の種類はたった 6 種類であり，命令セットに含まれるすべての命令をデコードするようなデコーダや，乗算器などの複雑な演算を行う演算器も必要ない．さらには，この命令列はループを形成していることから，各静的命令のフェッチやデコードなどの処理は原理的には 1 回だけ行えば十分である．

このような命令列を高い電力効率で実行するため，提案アーキテクチャでは汎用コアだけでなく非汎用コアもチップに搭載し，それを用いてプログラムを実行する．具体的には，以下のようなコアを搭載することを考えている．

汎用コア

- 通常の out-of-order コア
- 通常の inorder コア
- フロントエンドを細くする代わりに LSU のエントリ数やメモリ命令の発行数などを増やした out-of-order コア．MLP を利用したい命令列に対して高い電力効率を示すと考えられる．

非汎用コア

- 整数系パイプラインのみによって構成されたコア．整数系命令のみを含む命令列を高い電力効率で実行する．
- 浮動小数点系パイプラインのみによって構成されたコア．浮動小数点系命令のみを含む命令列を高い電力効率で実行する．
- データ・パス幅が通常よりも狭いコア．例えば，64 ビット・アーキテクチャであるにも関わらず，下位 32 ビットの演算しかできないコア．下位ビットの演算だけで十分な命令列に対して，高い電力効率を提供する．
- ループ実行専用コア．Revolver [7] のように，1 イタレーション分の処理を行うとフロント・エンドを停止し，バック・エンドのみを利用して命令を実行する．アーキテクチャの詳細は今後詳しく検討する．

その他にどのようなコアを用意すれば電力効率を改善できるか，今後さらに検討する予定である．

3.3 コアの切り替え方法

提案手法では，プログラム中の細かなフェーズに応じて最適なコアを選択できるように，コアの切り替えはハードウェアが自動的に行う．提案手法による電力効率の改善効果はどのような方法でコアの切り替えを行うかに強く依存するため，上述のハードウェアの制御方法は非常に重要である．

詳細は現在検討中であるが，コアの切り替え方法として以下のようなものを考えている．

- 一定区間（サイクル数 or コミット命令数）ごとに実行された命令列に関するさまざまな情報（IPC，キャッシュ・ミス数，命令種別，使用データ・パス幅，静的命令数など）を取得する．
- 取得した情報から次にその区間を実行する際のコアを決定する．例えば，整数系命令しか存在しないのであれば，その区間は次は整数系コアで実行した方がよい．決定したコアの番号は，区間の先頭の命令アドレスとともにテーブルに格納しておく．
- 次にその区間が実行される際は上記のテーブルを参照し，実行するコア番号を取得し，コアを切り替える．

上記の方法は一例であり，他の制御方法も含め，今後さらに検討を重ねる予定である．

4. 評価

予備評価として，コアの切り替えが理想的に行えると仮定した場合の提案手法の効果を測定した．本章ではその実験方法と結果を述べる．

4.1 実験方法

理想的な状況におけるヘテロジニアスなコアを有するアーキテクチャの消費電力と実行時間を評価した．ここで理想的な状況とは，1) コアの切り替えにともなう時間/エネルギ・オーバヘッドがゼロ，かつ，2) 複数のコアの中から後述する最適なコアを瞬時に選択できる，という状況を意味する．

評価したのは次の 5 つのアーキテクチャである．

BASE big（out-of-order）コア 1 つのみからなるプロセッサ．

BIG-LITTLE big（out-of-order）コアと LITTLE（in-order）コアの 2 つからなり，コアを切り替えながらプログラムを実行するプロセッサ．Composite Core 同様，2 つのコアはすべてのキャッシュを共有しており，1,000 コミット命令ごとにコアの切り替えを行うか否かの判断を行うものとする．

PROPOSAL-10K 後述する 19 種類のコアの中から最適なコアを選択しながらプログラムを実行するプロセッサ．コアを切り替える判断は 10,000 コミット命令ごとに行う．

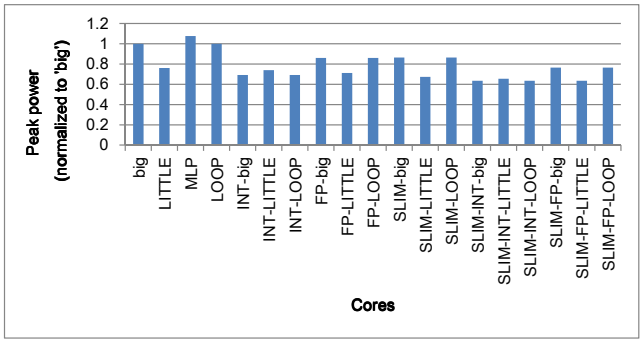

図 3 各コアのピーク電力

PROPOSAL-1K コアの種類は上記と同じだが，コアを切り替えるか否かの判断を 1,000 コミット命令ごとに行うプロセッサ．

PROPOSAL-100 コアの種類は上記と同じだが，コアを切り替えるか否かの判断を 100 コミット命令ごとに行うプロセッサ．

評価に用いたコアを**表 1** にまとめる．3 種類の汎用コア (big, LITTLE, MLP) と 16 種類の非汎用コア（LOOP, INT-*, FP-*, SLIM-* とその組み合わせ）を使用する．また，各コアのピーク電力を**図 3** に示す．図より，非汎用コアのピーク電力は big のそれの 6∼ 8 割程度である．図の値はピーク電力であり，LOOP コアのような一定時間経過後に一部のハードウェアが停止するコアの消費電力は，実際にはもっと小さな値となることに注意されたい．

代表的なコアのパラメタを**表 2** に示す．big コアのパラメタは Cortex-A57[1] のそれに，LITTLE コアのパラメタは Cortex-A53[10] のそれになるべく近づくように調整した．MLP コアは，フロントエンドの幅を 1 とし，big コアと比べて整数命令や浮動小数点命令の発行幅を半分にする代わりにメモリ命令の発行幅を倍にした．LOOP コアは，ハードウェア資源量は big コアに等しいが，1 イタレーション分の命令を処理するとフロントエンド（命令キューより前）に位置するすべてのハードウェアが停止する．

表 1 コア一覧

Name	Remarks
big	out-of-order 実行を行うコア
LITTLE	inorder 実行を行うコア
MLP	フロントエンドを細くし，メモリ・アクセス周りを強化した out-of-order コア
LOOP	10 命令以下の静的命令からなるループを out-of-order 実行するコア
INT-big	INT 系パイプのみを持つ big
INT-LITTLE	INT 系パイプのみを持つ LITTLE
INT-LOOP	INT 系パイプのみを持つ LOOP
FP-big	FP 系パイプのみを持つ big
FP-LITTLE	FP 系パイプのみを持つ LITTLE
FP-LOOP	FP 系パイプのみを持つ LOOP
SLIM-big	データ・パス幅が半分の big
SLIM-LITTLE	データ・パス幅が半分の LITTLE
SLIM-LOOP	データ・パス幅が半分の LOOP
SLIM-INT-big	INT 系パイプのみの SLIM-big
SLIM-INT-LITTLE	INT 系パイプのみの SLIM-LITTLE
SLIM-INT-LOOP	INT 系パイプのみの SLIM-LOOP
SLIM-FP-big	FP 系パイプのみの SLIM-big
SLIM-FP-LITTLE	FP 系パイプのみの SLIM-LITTLE
SLIM-FP-LOOP	FP 系パイプのみの SLIM-LOOP

今回の評価では，プロセス・ルールを 22nm とし，すべてのコアは 2 GHz で動作するものと仮定した．未使用のハードウェアに対しては理想的なパワー・ゲーティングが行われることを仮定し，それらのハードウェアは電力を消費しないものとして評価を行った．

BASE 以外の 4 つのアーキテクチャは，実行する命令列に応じて使用するコアを最適なものへと切り替える．ここで最適なコアとは以下のすべての条件を満たすコアを指す．

(1) 当該命令列が実行可能．
(2) 当該命令列の当該コアでの実行時間が性能制約以下．ただし，ここでの性能制約とは big コアに対する性能低下率を意味する．
(3) 上記の 2 条件を満たすコアの中で，当該命令列の実行における平均消費電力が最も少ないコア．

性能制約は，5%, 10%, 20% の 3 通りについて評価した．

性能評価には Onikiri2[15] を，消費電力の評価には改造した McPAT 1.0[11] を使用した．ベンチマーク・プログラムには SPEC CPU 2006 に含まれる全 29 本のプログラムを使用した．最初の 2G 命令をスキップし，続く 1G 命令を実行して評価を行った．

4.2 評価結果

性能制約を 5%とした場合の各アーキテクチャの平均消費電力を**図 4** に示す．グラフの横軸はプログラム名を，縦軸は平均消費電力（BASE で正規化）を表す．プログラムごとの 5 本の棒グラフは，左から順に BASE, BIG-LITTLE, PROPOSAL-10K, PROPOSAL-1K, PROPOSAL-100 である．

グラフより，まず，BIG-LITTLE は BASE よりも低消費電力であることがわかる．BIG-LITTLE は BASE に対して最大で 9.4%（zeusmp），平均で 1.5% の消費電力を削減している．

先行研究 [12] と比べて消費電力の削減効果が小さいのは，今回の実験で使用した big, LITTLE の回路規模の差が先行研究のそれよりも小さいことが原因と考えられる．実際，今回の実験で使用した LITTLE コアの IFU, L1D, LLC を除いた部分の面積は，big コアのそれの 53.6% であった．一方，文献 [12] によれば，little マイクロ・エンジンの面積は big マイクロ・エンジンの 28.6% に過ぎない．このような差が生じた原因については今後詳しく分析する．

表 2　代表的なコアのパラメタ

Parameters	big	LITTLE	MLP	LOOP
Type	out-of-order	inorder	out-of-order	out-of-order
Fetch width	3	3	1	3
Issue width	3	2	3	3
Issue queue	40	-	40	40
Function units (INT, FP, mem)	(2, 2, 1)	(2, 1, 1)	(1, 1, 2)	(2, 2, 1)
ROB	128	-	192	128
INT/FP PRF	160/160	-	240/240	160/160
Load/Store queue	16/16	-	24/24	16/16
Branch prediction	4KB-gshare, 512 entries BTB	←	←	←
Branch miss penalty	11 cycles	8 cycles	11 cycles	11 cycles
L1 I/D cache	32KB, 8 way, 64B/line, 1 cycle	←	←	←
L2 cache	1MB, 8 way, 64B/line, 15 cycles	←	←	←
Main memory	200 cycles	←	←	←
Remarks	-	-	-	IFU and RNU stop after processing the instructions correspondig to an interation

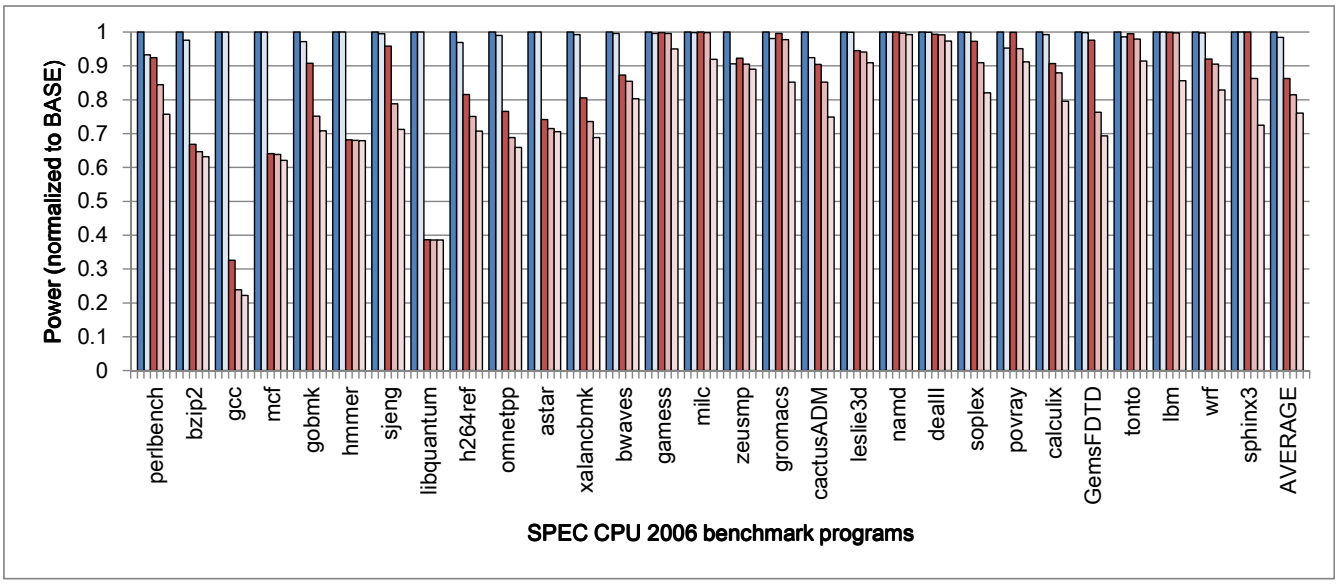

図 4　性能制約を 5%とした場合の各アーキテクチャの平均消費電力（左から順に BASE, BIG-LITTLE, PROPOSAL-10K, PROPOSAL-1K, PROPOSAL-100）

さらに，上記 2 つのアーキテクチャと提案アーキテクチャを比較すると，提案アーキテクチャはシングル・スレッド実行時の消費電力を大幅に削減できる可能性を秘めていることがわかる．提案アーキテクチャの消費電力削減効果は，今回の評価では切り替えオーバヘッドを無視しているため，コアの切り替え間隔が短くなる程改善する．BASE に対する提案アーキテクチャの消費電力削減効果は，コアの切り替え間隔が 100 命令ごとの時が最も大きく，最大 77.8%（gcc），平均 24.0% であった．また，BIG-LITTLE に対しては，最大 77.8%（gcc），平均 22.7% の消費電力を削減した．

各アーキクチャにおけるコアごとの実行サイクル数の内訳を**図 5** に示す．グラフの横軸はプログラム名を，縦軸はサイクル数ベースの各コアの使用率を表す．プログラムごとの 5 本の棒グラフは 5 つのアーキテクチャそれぞれに対応しており，その順序は図 4 と同様である．全実行サイクル数に占める各コアの稼働サイクル数を積み上げ棒グラフにより表示する．

グラフより，提案アーキテクチャでは，多くのプログラムにおいて非汎用コアの稼働時間が全実行時間の大部分を占めていることがわかる．特に libquantum においては，コアを 100 命令単位で切り替えた場合，INT-big と INT-LOOP の 2 つが稼働していた時間を合計すると全実行時間の 99.2 % に相当する．そのため，提案アーキテクチャは図 4 に示したような高い電力効率を達成する．

また，グラフより，コアの切り替え間隔が長くなるにつれて，非汎用コアの稼働時間が減少しているのがわかる．PROPOSAL-100 では非汎用コア（凡例における SLIM-

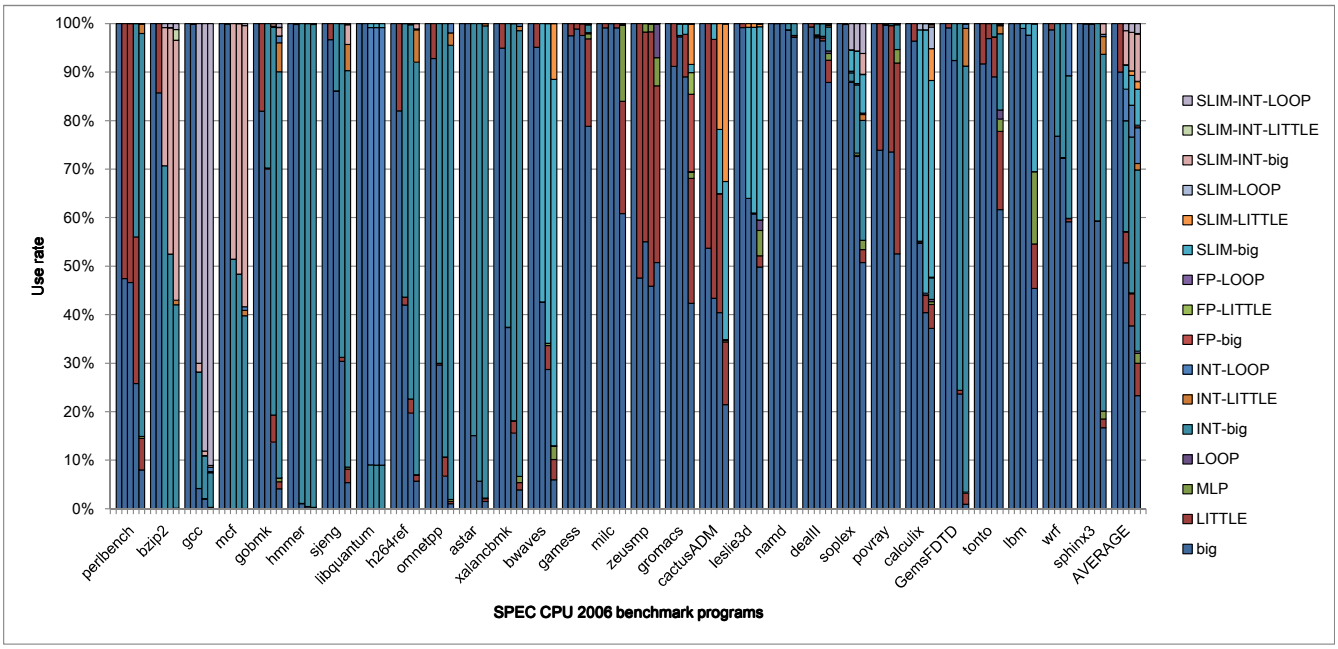

図 5 性能制約を 5%とした場合の各アーキテクチャにおけるコアごとの実行サイクル数（左から順に BASE, BIG-LITTLE, PROPOSAL-10K, PROPOSAL-1K, PROPOSAL-100）

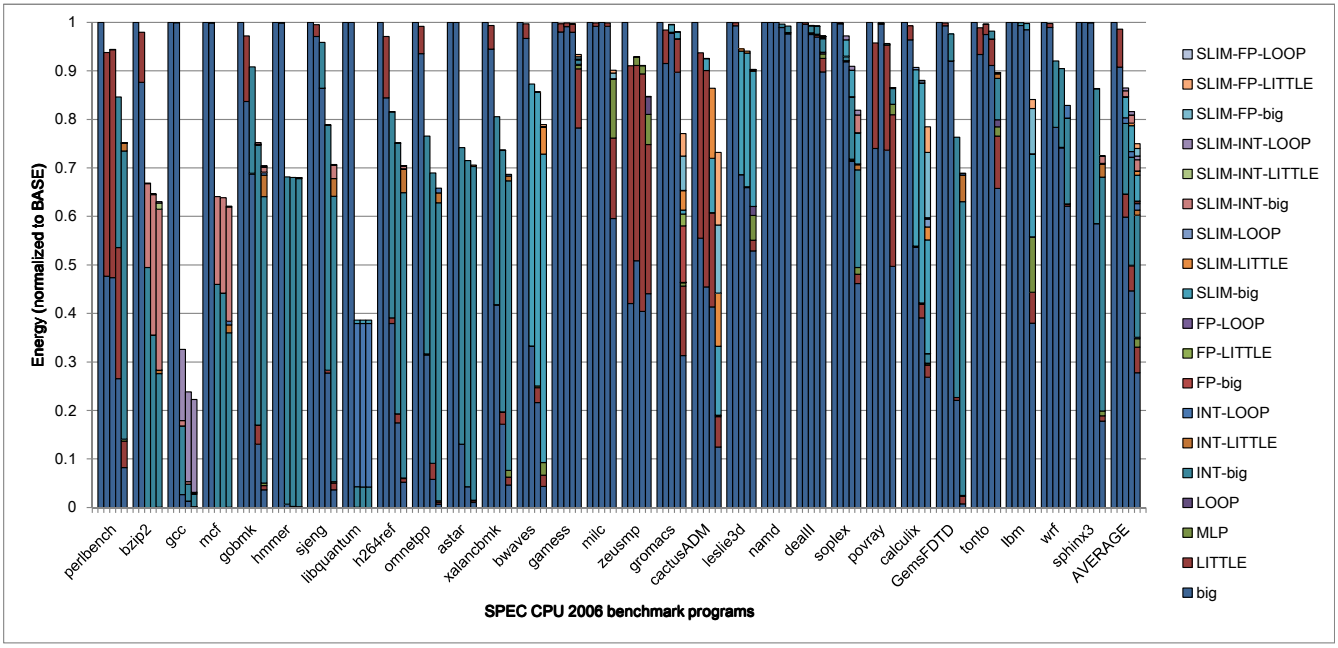

図 6 性能制約を 5%とした場合の各アーキテクチャにおけるコアごとの消費エネルギ（左から順に BASE, BIG-LITTLE, PROPOSAL-10K, PROPOSAL-1K, PROPOSAL-100）

INT-LOOP から LOOP までのコア）の稼働時間が平均で 69.0 % だったのに対し，PROPOSAL-1K では 55.6 %，PROPOSAL-10K では 42.9 % にまで減少する．これは，切り替え間隔が長くなる程，当該命令列が非汎用コアで実行できる（整数系命令が連続する，実行される命令がループ内に閉じている等の）確率が減少するためである．

各アーキテクチャの消費エネルギとその内訳を**図 6** に示す．グラフの見方は縦軸が消費エネルギに変更された点を除き，図 5 と同様である．ただし，各アーキテクチャの消費エネルギは BASE のそれで正規化してある．

グラフより，提案アーキテクチャでは，電力効率の悪い big コアの稼働が減り，電力効率に優れた非汎用コアの稼働が増えたことによって，消費エネルギを大幅に削減できていることがわかる．特に gcc においては，100 命令単位でコアの切り替えを行った場合，SLIM-INT-LOOP コアの稼働時間が 91.1%（図 5）に達したことで，BASE に対して 77.8% と大幅な消費エネルギの削減を達成した．これは，SLIM-INT-LOOP コアは big コアよりも電力効率

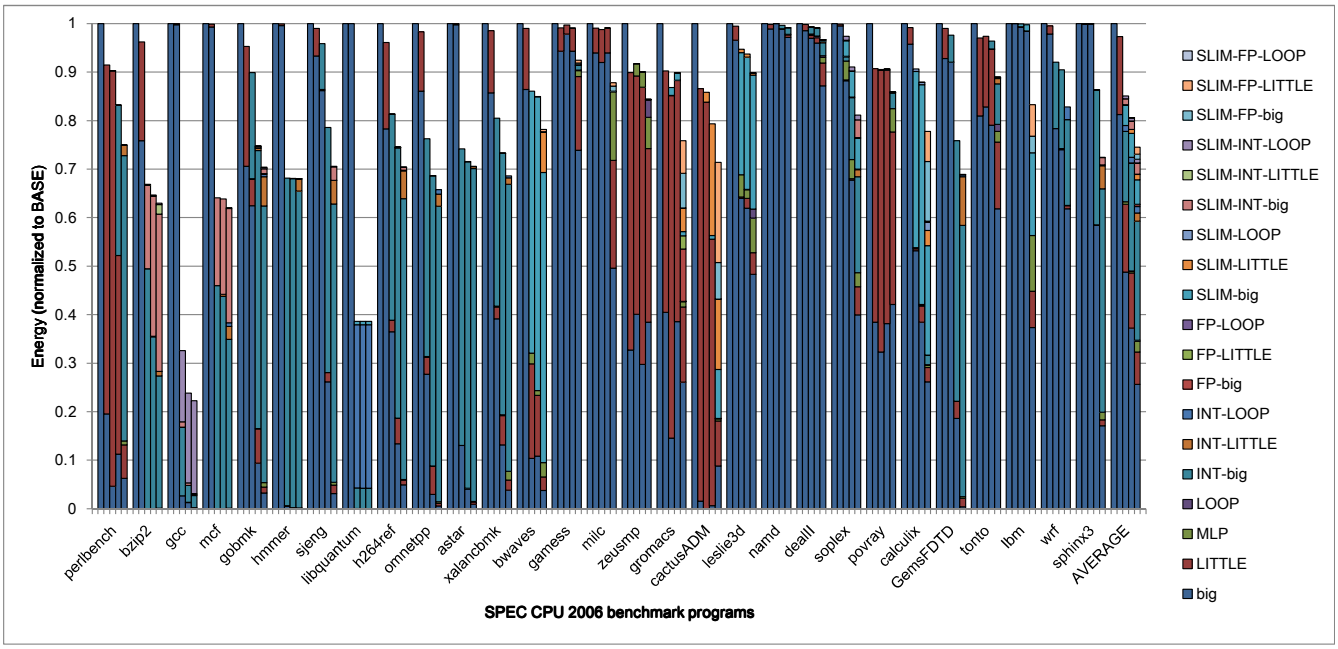

図 7 性能制約を 10%とした場合の各アーキテクチャにおけるコアごとの消費エネルギ（左から順に BASE, BIG-LITTLE, PROPOSAL-10K, PROPOSAL-1K, PROPOSAL-100）

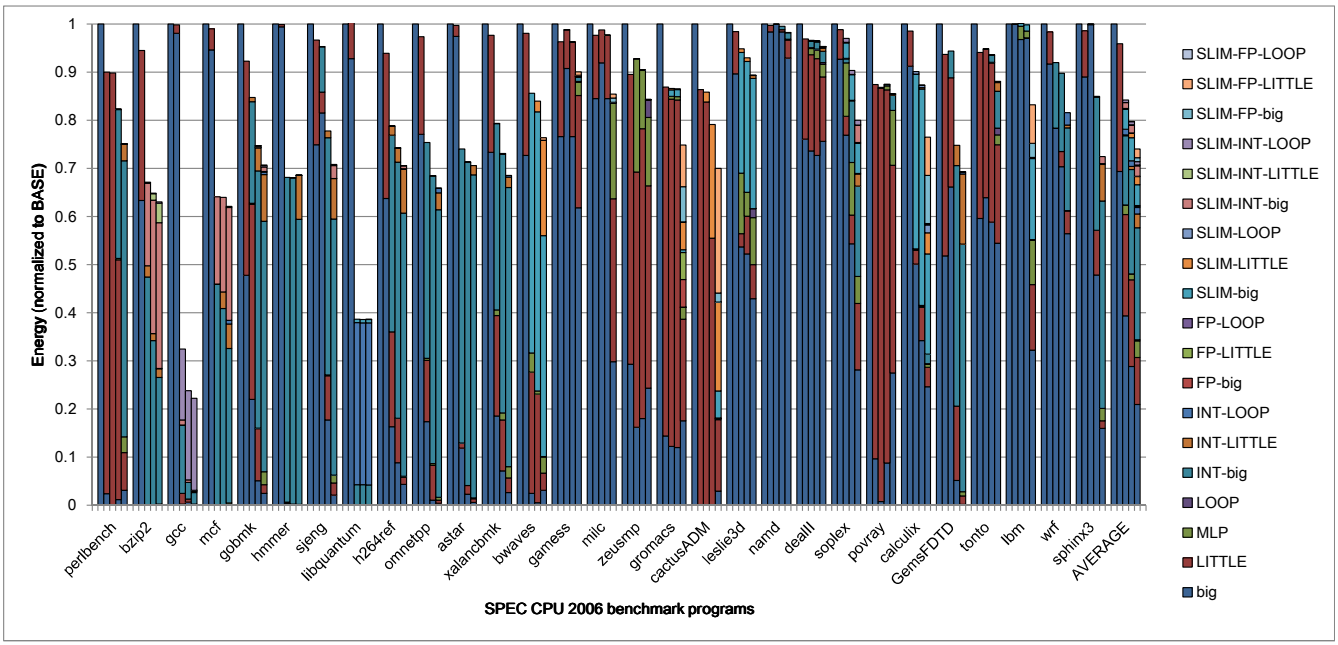

図 8 性能制約を 20%とした場合の各アーキテクチャにおけるコアごとの消費エネルギ（左から順に BASE, BIG-LITTLE, PROPOSAL-10K, PROPOSAL-1K, PROPOSAL-100）

が 86.7% 優れていることを意味している．なお，平均すると，コアの切り替え間隔を 10K, 1K, 100 命令単位とした場合，それぞれ，13.5%, 18.4%, 25.0% の消費エネルギの削減効果がみられた．

最後に，性能制約を 10%と 20%に変更した場合の各アーキテクチャの消費エネルギを，それぞれ**図 7** と**図 8** に示す．

グラフより，BIG-LITTLE/提案アーキテクチャともに，性能制約が緩くなるほど big コアの稼働割合が減り，電力効率が改善していることがわかる．ただし，その効果は，性能制約を 20% にまで緩めても，性能制約が 5% の時とほとんど変わらない．性能制約を 20% として 100 命令単位でコアの切り替えを行った場合，消費エネルギの削減効果は 26.0% であった．これは性能制約が 5% の時のそれと 1% ポイント異なるだけである．このように，性能制約を 5% よりも緩くする意味はあまりない．

以上の結果から，100 命令単位でコアの切り替えを行うことができれば，提案手法によってシングル・スレッド実行時の消費エネルギを大幅に削減できることがわかった．

5. まとめと今後の課題

Post Dennard 時代 [2] のプロセッサ設計においては，豊富に存在するトランジスタを如何にプロセッサの低消費電力化に繋げるかが重要である．このような考えにもとづき，我々は新たなプロセッサ・アーキテクチャとその設計手法を考えている．本稿ではそのアーキテクチャの概要と予備評価の結果について紹介した．

今後はアーキテクチャの細部，特にコアの切り替え方法に関して検討を進める予定である．また，提案アーキテクチャでは，プロセッサ設計時にどのようなコアをチップに搭載するかによって，シングル・スレッド実行時の電力効率の改善効果が異なってくる．今後は，そうしたコアの選択手法についても検討していきたいと考えている．

謝辞 本研究の一部は JST CREST，および，日本学術振興会 科研費若手 (A)（課題番号 24680005）による．

参考文献

[1] Bolaria, J.: Cortex-A57 Extends ARM's Reach, *Microprocessor Report 11/5/12-1*, pp. 1–5 (2012).
[2] Computer Community Consortium: 21st Century Computer Architecture, white paper (2012).
[3] Esmaeilzadeh, H., Blem, E., St. Amant, R., Sankaralingam, K. and Burger, D.: Dark Silicon and the End of Multicore Scaling, *Proceedings of the 38th Annual International Symposium on Computer Architecture*, pp. 365–376 (2011).
[4] Goulding-Hotta, N., Sampson, J., Zheng, Q., Bhatt, V., Auricchio, J., Swanson, S. and Taylor, M. B.: GreenDroid: An Architecture for the Dark Silicon Age, *Proceedings of the 17th Asia and South Pacific Design Automation Conference*, pp. 100–105 (2012).
[5] Greenhalgh, P.: big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7, white paper (2011).
[6] Guha, A., Zhang, Y., ur Rasool, R. and Chien, A. A.: Systematic Evaluation of Workload Clustering for Extremely Energy-efficient Architectures, *SIGARCH Computer Architecture News*, Vol. 41, No. 2, pp. 22–29 (2013).
[7] Hayenga, M., Reddy, V. and Lipasti, M. H.: Revolver: Processor Architecture for Power Efficient Loop Execution, *Proceedings of the 20th IEEE International Symposium on High Performance Computer Architecture*, pp. 591–602 (2014).
[8] ITRS: International Technology Roadmap for Semiconductors 2013 Edition (2013).
[9] Karpuzcu, U. R., Greskamp, B. and Torrellas, J.: The BubbleWrap Many-core: Popping Cores for Sequential Acceleration, *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 447–458 (2009).
[10] Krewell, K.: Cortex-A53 Is ARM's Next Little Thing, *Microprocessor Report 11/5/12-2*, pp. 1–4 (2012).
[11] 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, *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 469–480 (2009).
[12] Lukefahr, A., Padmanabha, S., Das, R., Sleiman, F. M., Dreslinski, R., Wenisch, T. F. and Mahlke, S.: Composite Cores: Pushing Heterogeneity Into a Core, *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 317–328 (2012).
[13] 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, *Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques*, pp. 133–144 (2013).
[14] Pricopi, M., Muthukaruppan, T. S., Venkataramani, V., Mitra, T. and Vishin, S.: Power-performance Modeling on Asymmetric Multi-cores, *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pp. 15:1–15:10 (2013).
[15] Processor Simulator Onikiri 2: http://www.mtl.t.u-tokyo.ac.jp/~onikiri2/.
[16] Sampson, J., Arora, M., Goulding-Hotta, N., Venkatesh, G., Babb, J., Bhatt, V., Taylor, M. B. and Swanson, S.: An Evaluation of Selective Depipelining for FPGA-based Energy-Reducing Irregular Code Coprocessors, *Proceedings of the Conference on Field Programmable Logic and Applications*, pp. 24–29 (2011).
[17] Taylor, M.: A Landscape of the New Dark Silicon Design Regime, *IEEE Micro*, Vol. 33, No. 5, pp. 8–19 (2013).
[18] Taylor, M. B.: Is Dark Silicon Useful? Harnessing the Four Horesemen of the Coming Dark Silicon Apocalypse, *Proceedings of the 49th Design Automation Conference*, pp. 1131–1136 (2012).
[19] Techworld: http://news.techworld.com/operating-systems/3477/moores-law-is-dead-says-gordon-moore/.
[20] Van Craeynest, K., Jaleel, A., Eeckhout, L., Narvaez, P. and Emer, J.: Scheduling Heterogeneous Multi-cores Through Performance Impact Estimation (PIE), *Proceedings of the 39th Annual International Symposium on Computer Architecture*, pp. 213–224 (2012).
[21] Venkatesh, G., Sampson, J., Goulding, N., Garcia, S., Bryksin, V., Lugo-Martinez, J., Swanson, S. and Taylor, M. B.: Conservation cores: reducing the energy of mature computations, *Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 205–218 (2010).
[22] Venkatesh, G., Sampson, J., Goulding, N., Venkata, S. K., Taylor, M. B. and Swanson, S.: QsCores: Configurable Co-processors to Trade Dark Silicon for Energy Efficiency in a Scalable Manner, *Proceedings of the 44th International Symposium on Microarchitecture*, pp. 163–174 (2011).
[23] 三輪忍，塩谷亮太，佐々木広：ダーク・シリコン時代のプロセッサ・アーキテクチャに関する初期検討，情報処理学会研究報告 2014-ARC-211，No. 5, pp. 1–7 (2014).