Vol.2014-SLDM-165 No.15
Vol.2014-EMB-32 No.15
2014/3/15

# 排他動作する非均質マルチコアプロセッサとそのリアルタイムOSの実装

高瀬 英希[1,a)]　李 景洙[1,b)]　石原 亨[1,c)]

**概要**：高性能 CPU コアと省電力 CPU コアを排他的かつ動的に切り替えて動作するマルチパフォーマンスプロセッサのテストチップを，65nm の商用プロセステクノロジを用いて試作した．各 CPU コアはそれぞれの電源電圧に最適化して設計しているため，低電圧動作時のエネルギー効率が従来型の DVFS プロセッサより高くなる．CPU コア間でオンチップメモリを共有することで，面積効率を向上させるとともに CPU コアの切替に掛かるオーバヘッドを抑止させる．さらに，組込みリアルタイムシステムへの応用を想定して，マルチパフォーマンスプロセッサの動作構成を切り替えて電力を管理する機能を提案する．電力管理機能を TOPPERS/ASP カーネル上に実装し，実チップ測定に基づいて評価した結果を報告する．

## 1. はじめに

携帯型情報端末の世界的普及を背景にマイクロプロセッサシステムの省エネルギー化が最重要課題の 1 つとなっている．2000 年には Dynamic Voltage and Frequency Scaling（以下 DVFS）技術がパーソナルコンピュータ向けのプロセッサに採用され，低消費電力プロセッサの 1 つのトレンドとなった．DVFS 技術は電源電圧と動作周波数を動的に変更することにより高速動作と低消費電力動作を両立する．しかし，DVFS 制御を行うプロセッサは，1 つのチップが時分割で異なる電源電圧を使用するため，広範囲に及ぶ電圧条件でのテストが必要になることやタイミング設計にかかる手間，オンチップ DC-DC コンバータのコスト，電圧・周波数切替のオーバーヘッドおよびこれに伴うリアルタイム性保証の困難さなどの理由から，これらのプロセッサがリアルタイム組込みシステムに適用された例は少ない．DVFS 技術に代わる方法として，筆者らはマルチパフォーマンスプロセッサ（以下 MPP）と呼ぶコンセプトを提案している [1]．MPP はプロセッサチップ内に高性能な CPU コアと省電力な CPU コアを搭載し，それらを排他的に動作させることによりプロセッサのピーク性能を保証しつつ低い平均消費電力を達成する．性能と消費電力の異なる CPU コアを複数搭載し，これらを並列動作させるヘテロジニアスマルチコアに関する提案は既に多数存在していたが，これらの CPU コアをあえて排他動作させるコンセプトの提案は，筆者らの知る限り MPP が世界初である．CPU コアが排他動作するマルチコアは，同時アクセス可能なマルチポートメモリなどが不要となるだけでなく，CPU コアを特定のタイミング条件に合わせてチューニングできるため設計コストを低く抑えることができ，設計品質を高めることができる．

NVIDIA 社が 2011 年に発表した Variable SMP と呼ばれる技術は，複数の高性能 CPU コアとコンパニオンコアと呼ばれる省電力 CPU コアを排他的に動作させることにより，高い性能が必要な場面と低い性能で十分な場面との両方で高いエネルギー効率を達成する [2]．また，ARM 社が提唱する big.LITTLE アーキテクチャは，省電力な Cortex-A7（または A53）と高性能な A15（または A57）の両方を同一チップ内に搭載し，これらを排他動作させることにより高性能と超低消費電力の両方を達成する [3]．2013 年には Samsung 社がスマートフォン向けのチップセットに big.LITTLE アーキテクチャを採用したことを発表した [4]．サーバ向けプロセッサにおいては KnightShift と呼ばれる同様の手法が提案されている [5]．計算負荷が軽い時には Knight と呼ばれる省電力・低性能 CPU が稼働し，それ以外の負荷の重い処理は高性能 CPU が担当する．

上述したように，性能と消費電力の異なる複数の CPU を排他的に動作させることによりピーク時の性能を保証しつつ平均消費電力を削減するアーキテクチャが高い注目を集めている．しかし，これらのアーキテクチャは過去数年内に提案された新しいアーキテクチャばかりであり，排他動作する非均質マルチコアを効率よく制御する手法に関す

[1] 京都大学大学院情報学研究科
Kyoto University, Sakyo-ku, Kyoto 606–8501, Japan
[a)] takase@i.kyoto-u.ac.jp
[b)] kslee@i.kyoto-u.ac.jp
[c)] ishihara@i.kyoto-u.ac.jp

る議論は十分に行われていない．本稿では，MPP アーキテクチャの詳細と設計方針を述べ，排他動作する非均質マルチコアの効率的な制御方法について議論する．

## 2. 諸準備

### 2.1 関連研究

#### 2.1.1 DVFS プロセッサ

DVFS プロセッサとは動作電圧と動作周波数を稼動時に変更することができるプロセッサである．**表 1** に商用の DVFS プロセッサとその電圧切替時間を示す [6]．表 1 から，電圧切替にかかる時間が非常に大きいことが確認できる．DC-DC コンバータが安定した電圧を出力するために，大容量（典型的には 100$\mu$F）のキャパシタを内蔵しており，電圧の昇降圧時にはこのキャパシタを充放電する必要があるためである [7]．電圧切替にかかるエネルギー消費のオーバヘッドも無視できない．電圧切替の際に 100$\mu$F のキャパシタを内蔵する DC-DC コンバータが消費する約 5$\mu$J のエネルギーは，組込みプロセッサのおよそ 2 万クロックサイクル分に相当する．

文献 [8] では，オンチップ電圧レギュレータを用いた細粒度 DVFS 技術が提案されている．コアごとの電圧を数十ナノ秒で切り替える．これ以外にも高速電圧・周波数切替を可能にする回路技術は多数提案されているが，オンチップ電圧レギュレータ（あるいは DC-DC コンバータ）のコストが非常に大きいため，オンチップ DC-DC コンバータを搭載した DVFS プロセッサが製品化された例は多くない．

DVFS プロセッサにおいてさらに注目すべき問題は，広い電圧範囲に対するタイミング設計の難しさとエネルギー効率の低さである．通常の設計は，対象とする 1 種類の電源電圧に対してタイミング設計とタイミング検証を行う．しかし DVFS プロセッサでは，動作対象とするすべての電圧条件でプロセッサが正しく動作するようにタイミング設計を行う必要がある．特にフリップフロップのホールド時間を考慮して広い電圧範囲でタイミング保証を行うことは容易ではない．また，文献 [9] でも指摘されている通り，1.2V を対象に設計したプロセッサを 0.7V で動作させると，0.7V の条件に最適設計したプロセッサと比べてエネルギー効率が劣化する．論理セルの種類によって電圧 - 遅延特性が大きく異なるためである．詳細は 3.4 節で述べる．

**表 1** 商用の DVFS プロセッサとその電圧切替時間

| Processor | Voltage (V) | Transiton Time |
|---|---|---|
| Transmeta Crusoe | 1.1-1.65 | 300$\mu$s |
| AMD Mobile K6 | 0.9-2.0 | 200$\mu$s |
| Intel PXA250 | 0.85-1.3 | 500$\mu$s |
| Compaq Itsy | 1.0-1.55 | 189$\mu$s |
| TI TMS320C55x | 1.1-1.6 | 3.2 ms (1.6 → 1.1V)<br>300$\mu$s (1.1 → 1.6V) |
| UCB [7] | 1.2-3.8 | 520$\mu$s |

#### 2.1.2 排他動作する非均質マルチコアプロセッサ

前節で述べた DVFS プロセッサ特有の問題を解決するために，高性能動作に最適化された CPU コアと低電力動作に最適化された CPU コアを 1 つのチップに搭載したアーキテクチャが注目を集めている [2], [3], [5]．高性能コアと低電力コアを排他動作させることにより高いピーク性能と低い平均消費電力を両立させる．文献 [4] では，ARM 社の big.LITTLE アーキテクチャを採用したチップが初めて公表された．高速動作用の CPU コア（Cortex-A15）を 4 個，低電力動作用の CPU コア（Cortex-A7）を 4 個搭載しており，前者の最大動作周波数は 1.8GHz，後者は 1.2GHz である．チップ上の占有面積は Cortex-A7 コア群が 3.8mm$^2$，Cortex-A15 コア群が 19mm$^2$ で，前者の 1MHz 当たりの消費電力は後者の約 1/6 である．

#### 2.1.3 非均質マルチコアプロセッサ向け OS

これまで提案されてきた非均質マルチコアプロセッサでは，基本方針として通常は低電力動作用の CPU コア群を動作させて処理の負荷が大きいときのみ高速動作用の CPU コア群を動作させる．これらのチップ上で動作するシステムソフトウェアとしては，筆者らの知る限りでは Linux または Android が提供されている．非均質マルチプロセッサは原理的には両コア群を並列動作させることが可能であるが，OS は両コア群を排他動作させる設定となっている．

これらの OS では，各プロセッサの負荷に応じてタスクを動的に割り付けるグローバルスケジューリング方式が採用されている．プロセッサの負荷変動によっては，タスクの実行中でも CPU コアの切替や割付プロセッサの変更が行われる．この方式は OS またはそれより下位のドライバのみで電力管理を担うため，アプリケーションの設計時には実行される CPU コアを意識する必要が無いという点では有用である．一方で，あるタスクが実行時にどちらの CPU コアで処理されるかを把握することはできなくなるため，実行時間の解析が困難となり，組込みシステムに求められるリアルタイム性が確保できなくなる．また，文献 [2], [3] のアーキテクチャでは L2 キャッシュのコヒーレントをとることを保証しているが，L1 キャッシュの内容は CPU コア間で共有しない．従って，タスクの割付コアが変更される時に L1 キャッシュのミスヒットが多発して著しく性能が悪化する可能性がある．

### 2.2 マルチパフォーマンスプロセッサ

マルチパフォーマンスプロセッサ（以下 MPP）は同じ命令セットアーキテクチャを持つ複数の CPU コアと動的に実効容量を変更できるキャッシュメモリとローカルメモリから構成される（**図 1** 参照）[1]．各 CPU コアはすべて同一の命令セットアーキテクチャを持つが，異なる消費エネルギー特性と動作性能を持つ．一般には低消費エネルギーと高速動作はトレードオフの関係にあるため，高速で消費

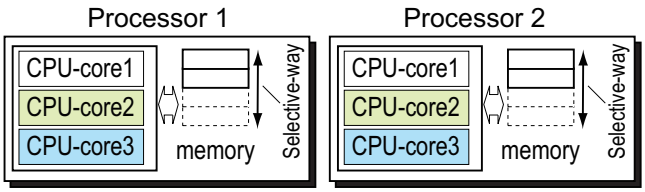


**図 1** マルチパフォーマンスプロセッサの概略図

エネルギーの大きい CPU コアと低速で消費エネルギーの小さい CPU コアをプロセッサコア内に複数搭載し，状況に応じて適切な CPU コアを稼動させる．同時には 1 つの CPU コアのみが稼動し，使用しない CPU コアへは信号供給とクロック供給を遮断する．長時間使用しないことがあらかじめ分かっている CPU コアにはパワーゲーティングによって電流供給を遮断し消費電力を削減する．キャッシュメモリやローカルメモリは CPU コア間で共有する．ただし，同時に 1 つの CPU コアのみが動作するため，メモリの入出力ポートはそれぞれ 1 ポートのみである．また，CPU コアはすべて同じ命令セットアーキテクチャを持つため，CPU コアを切り替えることによりソフトウェアからは 1 つの CPU コアが動的にその性能と消費エネルギーを変更しながら動作しているように見える．従って MPP は，マルチプロセッサでも命令レベル並列プロセッサでもない．タスクレベルの並列性が要求されるアプリケーションには MPP コアをバス接続することにより並列化する．

## 3. マルチパフォーマンスプロセッサの設計

### 3.1 回路構成

東芝社製の 32/16 ビット可変命令語長 RISC プロセッサ MeP をベースに MPP のプロトタイプを設計した．プロセステクノロジは商用の 65nm CMOS プロセスを使用した．CPU コア部分は同一の回路を複製し，性能と消費エネルギーの異なる CPU コアとして実現する．各 CPU コアは RT レベルでは同一の回路記述であるが別々の電源電圧を使用してキャラクタライズした標準セルライブラリを用いてそれぞれ設計する．本プロトタイプでは CPU コアの面積オーバーヘッドを考慮して，2 種類の CPU コアのみを搭載する構成とした．それぞれ 1.2V，0.7V を使って設計した．CPU コアの切替はスクラッチパッドメモリの特定のアドレスに値を書き込むことにより行う．メモリは 1.2V

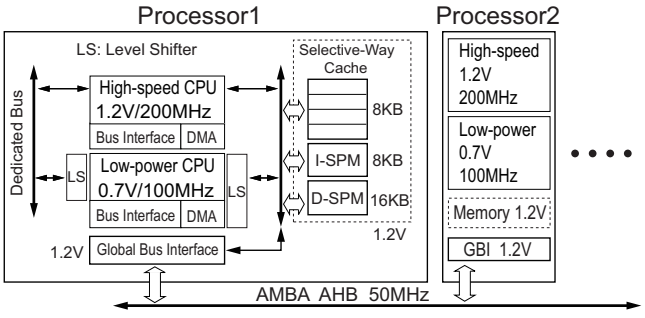


**図 2** マルチパフォーマンスプロセッサのプロトタイプ

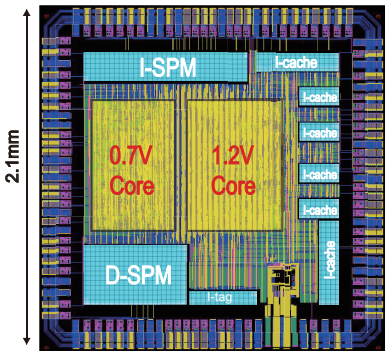


**図 3** プロトタイプチップのレイアウト図

を使用し，すべての CPU コアで共有する．ただし，CPU コアは同時に 1 つしか稼動しないためメモリはすべてシングルポートで実現する．今日のマイクロプロセッサチップは面積の大部分をメモリが占有するため，キャッシュメモリを含むプロセッサ自体を複数搭載するよりは面積効率が良い．本プロトタイプでは，オンチップメモリの面積がおよそ 0.674mm$^2$ であるのに対し，CPU コア 1 つ分の面積がおよそ 0.4mm$^2$ であるため，CPU コアの面積はメモリ部の面積のおよそ 60%である（**図 3** 参照）．

オンチップメモリは 8K バイト 4 ウェイセットアソシアティブの命令キャッシュと 8K バイトの命令コード用スクラッチパッドメモリおよび 16K バイトのデータ用スクラッチパッドメモリを搭載する．命令キャッシュは 4 ウェイのうち，稼動させるキャッシュウェイを選択する機能を持つ．つまり，キャッシュの連想度を動的に変更できるだけでなく，プログラムの動作状況に応じてどのキャッシュウェイを使うかをプログラマが指定できる．キャッシュウェイの選択もスクラッチパッドメモリの特定のアドレスに値を書き込むことにより行う．

### 3.2 設計フロー

多電源電圧設計を可能とするために，数種類の異なる電圧条件に対して標準セルライブラリのキャラクタライズを行った．キャラクタライズには SYNOPSYS 社の SiliconSmart を使用した．具体的には，0.65V，0.68V，0.7V，0.72V，0.75V でキャラクタライズを行った．次の手順で設計を行った．

(1) 1.2V の標準セルライブラリを使用して CPU コア部の論理合成を行い，CPU コアのターゲット周波数を決定する．ここで 1.2V は対象とするプロセステクノロジにおける定格電圧である．論理合成には SYNOPSYS 社の Design Compiler を使用した．プロトタイプ設計では，遅延制約が 5ns のときに PE コアの遅延制約がすべて満たされたため，200MHz を第一の CPU コアの最大動作周波数とした．

(2) 上記の最大動作周波数である 200MHz の 1/2(100MHz) を第二の CPU コアのクロック周波数とした．2 つの

CPU コアの動作周波数の比を整数とした理由はコア間通信の際の同期を取りやすくするためである．

(3) 第二の CPU コアを上記 5 種類の電圧条件でキャラクタライズしたセルライブラリを用いて設計した．遅延制約（100MHz）を満たした上で最もエネルギー消費が小さくなる電圧を選択したところ 0.7V となった．第二の CPU コアの電源電圧として 0.7V を採用した．

(4) オフチップバスのクロック周波数は上記最大動作周波数の 1/4（50MHz）とした．CPU コアの動作周波数との比を整数にした理由は，上記理由と同様である．

### 3.3 レベルシフタ回路

低電圧 CPU コアとオンチップメモリ，および CPU コア間はレベルシフタを使用して信号振幅を補正する．レベルシフタは文献 [10] でチャンらによって提案された回路を参考に自作した．自作のレベルシフタ回路を使用することにより 0.7V から 1.2V までの昇圧が最小インバータ回路約 3 段分の遅延で実現できた．スイッチング電力は 1.2V を使用した時の最小インバータセルの約 4.5 倍，リーク電流は約 4.3 倍であった．レベルシフタは標準セルと同様に SYNOPSYS 社の SiliconSmart を使用してキャラクタライズを行い，プロセッサの論理合成時にパス遅延解析と電力解析ができるようにした．

### 3.4 DVFS プロセッサとの比較

MPP 内の CPU コアは使用するそれぞれの電源電圧に最適化されるため，従来型の DVFS プロセッサよりも低電圧動作時のエネルギー効率が高い．**図 4** は，商用の 90nm プロセスを用いて 1.0V 条件で論理合成した CPU コアの最長パスを抽出し，そのパスに対する電圧・遅延特性を SYNOPSYS 社の HSPICE で計測した結果である．最長パスは SYNOPSYS 社の静的遅延解析ツール（STA）で抽出した．1.0V 動作時には 5ns であるパス遅延は，0.68V の電源電圧では 12.5ns まで増大する．この場合，CPU の動作周波数は 80MHz になる．0.68V でキャラクタライズしたセルライブラリを用いて最適設計した CPU コアは 133MHz で動作する．次に，1.0V 条件で設計した CPU コアの電圧を 0.52V まで下げるとそのパス遅延は 37ns になり，27MHz でしか動作しない．多入力論理セルと高い閾値電圧を使用する論理セルは低電圧動作時に遅延が急増するためだと考えられる．3.2 節で述べたとおりマルチパフォーマンスプロセッサはそれぞれの電圧に合わせたセルライブラリを使用して設計しているため，低電圧動作時でも回路のクリティカルパスの遅延はそれほど大きくならない．

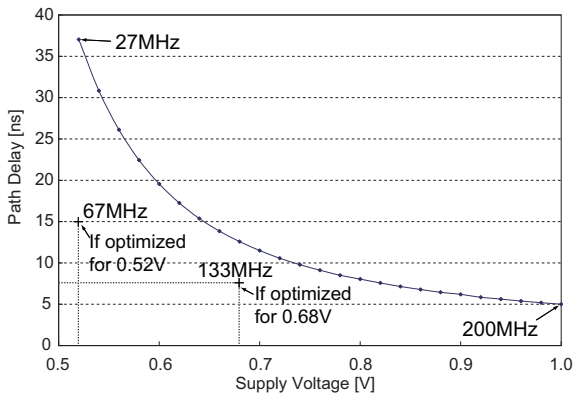


**図 4** 電圧・遅延特性

## 4. 電力管理機能を持つリアルタイム OS の実装

### 4.1 実装方針

リアルタイム性が求められる組込みシステムにおいては，タスクと稼働コアを静的に割り付ける機能分散型の処理方式が向いている．本方式では，設計者が各タスクの稼働コアを明示的に指定する必要があるものの，タスクのマイグレーションおよびそれに伴う性能の悪化を抑止できる．さらに，タスクの実行時間が解析しやすくなり，リアルタイム性の確保が容易になるという利点もある．

組込みシステムに適した排他動作する非均質マルチコアプロセッサの電力管理は，タスクにプロセッサの動作構成の情報を対応付け，その動作構成を制御することで実現する．組込みシステムのアプリケーションは一般に機能単位でタスクに分割され，リアルタイム OS はタスク単位で処理を実行する．実行されるタスクに応じてプロセッサの動作構成も切り替えることで，電力管理が効率的に行えるようにする．さらに，タスクに対応する動作構成の設定を実行時に変更するための API を提供する．MPP において選択しうるプロセッサの動作構成は，稼働させる CPU コア，および，稼働させる命令キャッシュのウェイがある．

提案する電力管理機能は，TOPPERS 新世代カーネル統合仕様書に準拠したリアルタイム OS の機能拡張により実装する．本仕様書は，国内の組込み分野のデファクトである $\mu$ITRON 4.0 仕様を出発点として規定されている．カーネルの実装も TOPPERS プロジェクトよりオープンソースで公開されており，実製品での採用実績もある [11].

### 4.2 タスク管理情報の拡張

タスクと実行時の動作構成を対応付けるため，TOPPERS カーネルにおけるタスク管理ブロック（TCB）の仕様を拡張する．具体的には，TCB に対して現在の動作構成の設定値を保持する `config`，タスク初期化ブロック（TINIB）に対して動作構成の設定の初期値を保持する `iconfig` を追加する．いずれのメンバも型は符号付き整数の `CFG` である．

初期値 `iconfig` を TINIB に格納するため，タスクを生成する静的 API である `CRE_TSK` の仕様も拡張する．具体的には，パラメータであるタスクの生成情報のパケット `C_CTSK *pk_ctsk` に `CFG iconfig` を追加する．`iconfig` で指定された値は，カーネルの初期化ルーチンにおいて各

set_cfg　タスクの動作構成の変更
【C 言語 API】
ER ercd = set_cfg(ID tskid, CFG config)
【パラメータ】

| | | |
|---|---|---|
| ID | tskid | 対象タスクの ID 番号 |
| CFG | config | プロセッサの設定値 |

【リターンパラメータ】

| | | |
|---|---|---|
| ER | ercd | 正常終了（E_OK）またはエラーコード |

【エラーコード】

| | |
|---|---|
| E_CTX | CPU ロック状態からの呼出し |
| E_ID | 不正 ID 番号（tskid が不正） |
| E_PAR | パラメータエラー（config が不正） |

**図 5**　set_cfg() の仕様

get_cfg　タスクの動作構成の参照
【C 言語 API】
ER ercd = get_cfg(ID tskid, CFG *p_tskconfig)
【パラメータ】

| | | |
|---|---|---|
| ID | tskid | 対象タスクの ID 番号 |
| CFG | *p_tskconfig | 動作構成の設定値を入れるメモリ領域へのポインタ |

【リターンパラメータ】

| | | |
|---|---|---|
| ER | ercd | 正常終了（E_OK）またはエラーコード |

【エラーコード】

| | |
|---|---|
| E_CTX | CPU ロック状態からの呼出し |
| E_ID | 不正 ID 番号（tskid が不正） |

**図 6**　get_cfg() の仕様

タスクの TCB のメンバ config に設定されるようにする．

タスク切替時には，プロセッサの動作構成を対応した設定値のものに切り替えるようにする．この処理は，ディスパッチャへの機能拡張によって実現される．ディスパッチャでは，実行状態となるタスクのディスパッチ直前に，その TCB に保持されている config を参照する．この参照値がスクラッチパッドメモリの特定のアドレスに保存されている現在の動作構成の設定値と異なる場合には，プロセッサの動作構成を切り替えるための処理を行う．なお，カーネルの内部では CPU コアの変更が必要な場合とウェイのみの変更が必要な場合とで処理を分けるようにしている．処理を分けた理由は，MPP における CPU コアの変更には命令キャッシュのウェイの変更と比較して実行時間のオーバヘッドが大きく掛かり，さらにソフトウェアでレジスタの保存と復帰を行う必要があるためである．

### 4.3　電力管理のための API

#### 4.3.1　set_cfg

set_cfg は，指定されたタスクの動作構成の設定値を変更する API である．本 API の仕様を**図 5** に示す．

set_cfg が呼び出されると，リアルタイム OS は対象タスクの TCB から現在の動作構成の設定値を参照し，指定された config と異なっていれば TCB の値を更新する．tskid に自タスクまたは実行状態のタスクが指定された場合は，プロセッサの動作構成を切り替える処理にただちに遷移する．つまり，タスクの実行内でも明示的に動作構成を切り替えることが可能である．動作構成の切替は割込みを禁止して行う必要があるため，本 API を CPU ロック状態から呼出すことは禁止する．なお，MPP のターゲット依存の API として，CPU コアおよび命令キャッシュのウェイを個別に変更する set_cor および set_ica も提供する．

#### 4.3.2　get_cfg

get_cfg は，指定されたタスクの動作構成の設定値を参照する API である．本 API の仕様を**図 6** に示す．CPU ロック状態からの呼出しでない場合は，対象タスクの TCB に保持されている動作構成の設定値が，*p_tskconfig で指定したメモリ領域に返される．

## 5.　テストチップによる実機評価

### 5.1　MPP のプロセッサ構成の切替に掛かるオーバヘッド

3 章で設計した MPP のテストチップについて，プロセッサの動作構成を切り替えるのに掛かる実行時間を測定した．

稼働する CPU コアの切替については，1.2V/200MHz の High-speed CPU から 0.7V/100MHz の Low-power CPU に切り替えた場合は 1.65$\mu$s，逆に切り替えた場合は 1.48$\mu$s となった．これらの値には，レジスタの保存と復帰に掛かる実行時間も含まれている．表 1 に示す商用の DVFS プロセッサと比較すると，MPP は大幅に高速に電圧と性能を切り替えることができることがわかる．big.LITTLE アーキテクチャの非均質マルチコアプロセッサではブラックアウト期間（CPU 切替によりプログラムが実行できない期間）が最小でも 20$\mu$s とされているが，我々の設計した MPP はこれよりも優れてる．さらに，MPP において CPU コアを切り替えた後には，CPU コア間でオンチップメモリを共有しているため 2.1.3 節で指摘した既存の非均質マルチコアプロセッサが抱えるキャッシュのミスヒットの問題も発生しない．なお，コア切替の実行時間が非対称であるのは，実行サイクル数自体は同じものの，切替の前処理よりも後処理が多くなり，High-speed CPU でのサイクル数と Low-power CPU でのサイクル数が異なることに起因する．

命令キャッシュのウェイのみを切り替える際には，High-speed CPU の稼働時には 25ns，Low-power CPU の稼働時には 50ns となった．いずれも 5 クロックサイクル分の実行時間であり，非常に小さなオーバヘッドで切り替えることができる．ウェイの切替に掛かる実行時間は，切替前後のウェイ数に依存せず同一である．なお，CPU コアと命令キャッシュのウェイの切替は並列に行えるため，これらを同時に切り替える場合は後者の実行時間は隠蔽される．

### 5.2　リアルタイム OS の電力管理に掛かるオーバヘッド

4 章で提案した電力管理機能を TOPPERS/ASP カーネル上に実装し，メモリサイズを測定した．TOPPERS カー

ネルでは，カーネルおよびアプリケーションのオブジェクトを 1 つのバイナリにまとめてリンクするモデルを採用している．タスク数 5 の sample1 でメモリサイズを評価したところ，電力管理機能が未実装のカーネルでは 43,308Byte，実装したカーネルでは 45,136Byte となった．sample1 の場合は電力管理機能のメモリサイズが全体の 4.2 %のみの増加に留まっている．このことから，提案手法はメモリ容量に制約のある組込みシステムに適していることがわかる．

### 5.3 アプリケーションの平均消費電力

テストチップ上でアプリケーションを実行した時の平均消費電力を測定した．評価アプリケーションには，それぞれに異なる実行周期のタスク（内容は離散コサイン変換）を 3 個用意し，レイトモノトニック方式に従って TOPPERS/ASP カーネルでスケジューリングされるものを用いた．平均消費電力は，テストチップに対してアジレント社 U2722A を用いてサンプリングレート 60Hz で消費電流および電圧を採取して算出した．電力管理の方針は，以下の 3 種類を適用した．

**Nonmanage** 動作構成の設定を行わない．全てのタスクを MPP の起動時の構成である High-speed CPU/4-way として実行する．

**Static** 動作構成の設定を静的に行う．タスクの優先度順に，High-speed CPU/4-way，Low-power CPU/4-way，Low-power CPU/1-way として実行する．

**Dynamic** 動作構成の設定を動的に行う．タスクの起動時に実行可能状態または実行状態となっているタスク数に応じて，3 つの時は High-speed CPU/4-way，2 つの時は Low-power CPU/4-way，1 つの時は Low-power CPU/1-way として実行する．

評価結果を**表 2** に示す．平均消費電力は，0.7V CPU 部，1.2V CPU 部および 3.3V I/O 部に分類して示した．Nonmanage と他 2 方針を比較すると，Static では 23.3%，Dynamic では 46.9%の平均消費電力の削減が達成できた．このことからタスクとプロセッサの動作構成を対応付けてリアルタイム OS で制御する提案手法が効率的であることがわかる．さらに，Dynamic における全体の平均消費電力は Static よりも 31.6%小さくなった．Dynamic は 4.3 節で提案した API を用いて実現されており，この API が有用であることがわかる．また，文献 [12] で提案されている Dynamic Energy and Performance Scaling のような，より積極的な方針を電力管理機能によって実現することで，さらなる消費エネルギー削減が可能であると考えられる．

**表 2** 評価アプリケーションの平均消費電力［mW］

| Policy | 0.7V CPU | 1.2V CPU | 3.3V I/O | Total |
|---|---|---|---|---|
| Nonmanage | 0.38 | 11.80 | 8.26 | 20.45 |
| Static | 0.58 | 9.50 | 5.80 | 15.88 |
| Dynamic | 0.73 | 7.47 | 2.66 | 10.87 |

以上のことから，MPP と電力管理機構の有効性が示せた．

## 6. おわりに

本稿では，非均質な CPU コアを排他的かつ動的に切り替えて動作する MPP のテストチップを，65nm の商用プロセステクノロジを用いて試作した．MPP はその動作構成を瞬時に切り替えることができる．さらに，MPP の動作構成の切替によって電力管理を実現する機能をリアルタイム OS に実装した．実チップ測定に基づいて MPP および電力管理機能を評価してそれらの有効性を示した．今後は，電力管理機能を活用した消費エネルギー削減手法の提案やマルチプロセッサへの対応を計画している．

**謝辞** 本研究の一部は最先端・次世代研究開発支援プログラム（課題番号：GR076）による．本研究は東京大学大規模集積システム設計教育研究センターを通し，株式会社東芝，シノプシス株式会社，日本ケイデンス株式会社の協力で行われたものである．

### 参考文献

[1] Oyama, Y. et al.: A Multi-Performance Processor for Low Power Embedded Applications, Proc. of COOL Chips X, Vol. 1, pp. 138 (2007).
[2] NVIDIA Corporation: Variable SMP – A Multi-Core CPU Architecture for Low Power and High Performance, White Paper (2011).
[3] Jeff, B.: Advances in big.LITTLE Technology for Power and Energy Savings, White Paper (2012).
[4] Shin, Y. et al.: 28nm High-Metal-Gate Heterogeneous Quad-Core CPUs for High-Performance and Energy-Efficient Mobile Application Processor, Proc. of Int'l Solid-State Circuit Conference, pp. 154–155 (2013).
[5] Wong, D. and Annavaram, M.: KnightShift: Scaling the Energy Proportionality Wall Through Server-Level Heterogeneity, Proc. of IEEE/ACM Int'l Sympo. on Microarchitecture, pp. 119–130 (2012).
[6] Shin, D and Kim, J.: Intra-Task Voltage Scheduling on DVS-Enabled Hard Real-Time Systems, IEEE Trans. on CAD of Integrated Circuits and Systems, Vol. 24, Issue 10, pp. 1530–1549 (2005).
[7] Burd, T. and Brodersen, R. W.: Design Issues for Dynamic Voltage Scaling, Proc. of Int'l Sympo. on Low Power Electronics and Design, pp. 9–14 (2000).
[8] Kim, W. et al.: System Level Analysis of Fast, Per-Core DVFS Using On-Chip Switching Regulators, Proc. of Int'l Sympo. on High Performance Computer Architecture, pp. 123–134 (2008).
[9] Ishihara, T. et al.: AMPLE: An Adaptive Multi-Performance Processor for Low-Energy Embedded Applications, Proc. of Int'l Sympo. on Application Specific Processors, pp. 83–88 (2008).
[10] チャン・クワン・カイン，桜井貴康：低電圧対応のレベルコンバータ，電子情報通信学会総合大会予稿集，pp. 8 (2004).
[11] TOPPERS プロジェクト (online), http://toppers.jp/ (2014.2.13).
[12] Takase, H. et al.: An Integrated Optimization Framework for Reducing the Energy Consumption of Embedded Real-Time Applications, Proc. of Int'l Sympo. on Low Power Electronics and Design, pp. 271–276 (2011).