Vol.2013-ARC-206 No.23
2013/8/1

Androidベースマルチコア上での自動電力制御

平野　智大[1]　山本　英雄[1]　武藤　康平[1]　三神　広紀[1]　後藤　隆志[1]　Dominic Hillenbrand[1]
木村　啓二[1]　笠原　浩徳[1]

概要：近年，スマートフォン，タブレットといったスマートデバイスはより高いパフォーマンスの要求を満たすためにシングルコアプロセッサからマルチコアプロセッサに移行している．しかしながら，スマートデバイスはより高頻度で利用されるに伴って，電力消費量は増加している．スマートデバイスの消費電力の問題は，スマートデバイスの利用者と生産にかかわる産業界が直面する最も重要な難題の１つである．本稿では，ODROID-X2 と呼ばれる開発ボードを用いて，疑似クロックゲーティング手法と GPIO を利用した正確な電力測定環境を用意し，Android プラットフォーム上での OSCAR 自動並列化並列化コンパイラによる電力制御の有用性の調査を行った．また，新しい疑似クロックゲーティング手法として WFI を用いて演算命令の中断とクロックの停止を 500[us] 間隔で実現可能にした．さらに，プロセッサ内のの GPIO の動作の制御を可能にし，プログラム中から GPIO の制御 API を呼び出すことで，電力波形図に GPIO 制御 API の呼び出し箇所が明示できるようにした．これにより GPIO 制御の状態からプログラムと電力波形の対応関係が明確化された正確な電力測定環境を実現した．リアルタイム制約のもと，MPEG2 デコーダと Optical Flow を用いて評価を行ったところ，MPEG2 デコーダでは 1PE で 0.97[W] から 0.63[W] に，2PE で 1.88[w] から 0.46[W] に，3PE で 2.79[W] から 0.37[W] へ電力削減が確認できた．また，Optical Flow を用いて評価を行ったところ，1PE で 0.95[W] から 0.72[W] に，2PE で 1.50[w] から 0.36[W] に，3PE で 2.23[W] から 0.30[W] へ電力削減が確認できた．

キーワード：スマートデバイス， 自動並列化，API，電力制御，電力削減，マルチコア，Android，GPIO，WFI

1. はじめに

デスクトップパソコン，ワークステーションや組み込みシステムに至るまでマルチコアプロセッサは適用されている [1], [2], [3]．近年においてはスマートフォンやタブレット端末においても膨大な演算処理能力と低消費電力野要求を満たすために，マルチコア化が進んでいる．しかし，スマートデバイスはより頻繁に利用されるようになり，スマートデバイスの消費電力は増大している．消費電力の増大は利用可能時間の低下を招く．利用可能時間の拡大のためにも，低消費電力化はスマートデバイス産業が最優先で対処すべき重大な問題である．消費電力の増大を防ぐためにモバイルデバイス産業では ARM 社 [4] より提供されている big.LITTLE[5] や NVIDIA 社より提供されている Tegra[6] Samsun Exynos 5 OCta[7] といったような低消費電力アーキテクチャを利用することで解決を試みようとしている．

最近のスマートデバイスはマルチコアプロセッサを適用しているが，より高い性能を得るためには，ソフトウェアの並列化と，ソフトウェアとハードウェアの更なる協調により，マルチコアプロセッサの能力を最大限引き出すことが重要である．

また，現在の Linux には電力制御の仕組みが備わっているが，その対象はシステムの負荷状況に応じた電力制御であるため，コンパイラが能動的に電力制御を行うにはオーバーヘッドが大きい．電力制御を最大限行うためにも，ソフトウェア側からの電力最適化が重要である．そのためには，コンパイラと OS の協調が重要である．

OpenMP や MPI を含む現在の並列化の方法としては手動で行われているが，手動の最適化は生産性を低下させ，ソフトフェアの複雑化が高まるに応じて並列化が困難となる．マルチコアプロセッサのためのソフトフェアの最適化を容易にするためには自動並列化が必要である．このような自動並列化コンパイラは OSCAR 自動並列化コンパイラ [8], [9] といったものが存在する．

さらに，OSCAR コンパイラは DVFS やクロックゲー

[1] 早稲田大学
Waseda University

ティング，パワーゲーティング [10] といった電力制御を利用できるようにする最適電力プログラムの自動生成が実現できる．

本稿では OSCAR 自動並列化コンパイラを利用して電力削減を適用したリアルタイムアプリケーションを Android[11] プラットフォームの ODROID-X2[12] で評価を行った．

さらに，WFI(Wait For Interrupt) の命令を使い，500[us] 間隔で利用できる疑似クロックゲーティングを開発した．これにより現状の Android プラットフォームの電力制御環境と比較した結果，より効率のよい電力制御が実現できた．また，これまではプログラムと消費電力の対応関係を正確に示す方法がなかった．これを改善するために，GPIO を利用することによって，プログラムと電力波形の対応関係を正確にした．

本稿では Android プラットフォームでの現状の電力制御について 2 章で，方法について 3 章で，測定環境について 4 章で，評価結果について 5 章で，本稿の結論を 6 章で説明する．

2. 現状の Android プラットフォームにおける電力制御

この章では Android の電力制御について述べる．Android は Linux ベースで作られており，電力制御においても，Linux の電力制御がベースとなっている．

Linux の電力制御は動作時のクロックと電圧を変更する CPUFreq， 休眠可能時に休眠の状態を管理する CPUIdle，

マルチコアのコア毎の電源を管理する HotPlug[13] の 3 つを通して実現する．

2.1 CPU Freq．

`cpufreq` は CPU の周波数変化を可能とする，Linux に標準搭載されているドライバである．

Android デバイスでは周波数の値の決定は `ondemand` ガバナーによって実現される．このガバナーモニターは現在の各コアの使用状況を一定周期で確認し，負荷状況がしきい値よりも高い，または低い値になった場合，周波数は動的に変化する．

2.2 CPUIdle．

多くの Android デバイスの CPU は複数の待機レベルを持っていて，消費電力量と復帰時間が異なる．`cpuidle` は CPU のコア毎に待機レベルを管理し，デバイス上で低消費電力を実現する．

Linux は演算処理が無い場合，待機状態に移行する決定をする．待機状態は CPU 上のスリープに移行する機能ユニットの数に応じて決定される．多くの機能ユニットがスリープ状態に移行する場合は消費電力は非常に低くなるが，スリープ状態から復帰する場合には復帰時間が増大する．逆に，少ない数の機能ユニットがスリープ状態になる場合，消費電力の減少幅は少なくなるものの，復帰時間が速くなる．通常，一定期間スリープ状態が続いた場合，待機状態レベルは深くなっていく．

2.3 HotPlug．

`hotplug` は `cpufreq` の拡張機能であり，マルチコアプロセッサの消費電力の削減に向けて使用される技術である．

あるコアの `cpuidle` に最大周波数がセットされており，一定期間同様の状態が続く場合，`hotplug` は他のコアを立ち上げ，負荷分散を行う．同様に，あるコアの `cpuidle` に最小周波数がセットされており，一定期間同様の状態が続いた場合，`hotplug` はコアの電源を遮断し，消費電力の低下を行う．

3. 電力制御適用手法

この章では OSCAR 自動並列化コンパイラと OSCAR API による電力制御の実現の方法の詳細について述べる．さらに，疑似クロックゲーティングと，電力値とプログラムの対応関係の観測方法についても同様に述べる．

3.1 OSCAR 自動並列化コンパイラ

OSCAR (Optimally SCheduled Adavanced multiprocessoR) 自動並列化コンパイラは一般的なループ間のみの並列化を行うのではなく，粗粒度並列化，ループレベルの中粒度並列化，ステートメントレベルの近細粒度並列化を組み合わせた，マルチグレイン並列化を行う．マルチグレイン並列化を行う為に OSCAR 自動並列化コンパイラでは，逐次プログラムである C コードや fortran のコードを基本ブロック (BB)，ループブロック (RB)，サブルーチンブロック (SB) で構成される，マクロタスクと呼ばれる粗粒度タスクに分解する．これらのマクロタスクを用いて，OSCAR コンパイラはコントロールフローとデータ依存関係を表現したマクロフローグラフ (MFG) を生成する．その後，コンパイラはソース中のタスクの入力変数の定義，出力変数の使用を解析することでデータ依存解析を行い，MFG から MT 間の並列性を最早実行可能条件解析により引き出した結果をマクロタスクグラフ (MTG) という．

MT がサブルーチンコールや粗粒度並列化が可能なループである場合，OSCAR コンパイラは階層的にその MT の中に MT を生成する．さらに，ループ間レベルの並列性はループ分割により粗粒度並列タスクとして分割される．

これらのマクロタスクは各レイヤの階層型マクロタスクグラフの並列性の論理的かつ階層的な考慮によりプロセッサグループ (PG) にグループ化され，プロセッサコアに割り当てられる．

MTG が実行時に変動するか，条件分岐がある場合，動

的スケジューリングが適用される．そうでなければ，静的スケジューリングが MTG に適用される．[14]

静的スケジューリングされた MTG 間にビジーウェイト処理がある場合，コンパイラーは DVFS により MT の実行時間を延長するか待機処理にクロックゲーティングおよびパワーゲーティングを適用することにより，消費電力の合計を最小化しようとする．この演算モードは最速実行モード [15] と呼ばれる．最速実行モードの場合，OSCAR コンパイラはプログラム実行時間が延長しないように，DVFS，クロックゲーティングおよびパワーゲーティングの適用を管理する．

同様に，MTG のデッドラインが与えられ，デッドラインまで十分な待機時間がある場合，OSCAR コンパイラは消費エネルギーの合計を最小になるよう考え，MT に DVFS を適用するか，あるいはデッドラインになるまでの待機時間，クロックゲーティングおよびパワーゲーティングを適用する．この演算モードをデッドラインモードと呼ぶ．

動画再生のように，電力最適化された MTG がデッドラインを繰り返し守るような場合，リアルタイム制御モードと呼ぶ．今回の評価はリアルタイム制御モードによる評価を行った．

3.2 OSCAR API

OSCAR API (Application Programming Interface) はサーバーやデスクトップコンピュータ，組み込みシステム [10] など，様々な共有メモリマルチコアプロセッサやマルチコアシステムに OSCAR コンパイラの最適化を適用するための標準解釈系である．

OSCAR API は OpenMP のサブセットに基づいたコンパイラディレクティブセットで構成される．OSCAR API は thread の生成，ローカルメモリや共有メモリへの分配を考慮したメモリアロケーションに加え，ユーザーレベル電力制御を使用する．OSCAR コンパイラはこれらのディレクティブを挿入し，OpenMP が並列化プログラムを生成できるようにする．その後，サーバーなどで利用する場合には OpenMP コンパイラが並列化されたプログラムを演算できるバイナリを生成する．

API 標準解釈系は，組み込みシステムなどで用いるために OSCAR API ディレクティブをランタイムライブラリコールに変換する目的で開発された．

今回の場合，通常の gcc のようなコンパイラが並列演算可能なバイナリを生成する．電力制御のために，OSCAR API は `fvcontrol` や `get_fv_status` ディレクティブが存在する．

`fvcontrol` ディレクティブはターゲットシステムの中でハードウェアモジュールの電力状態を指定された値にセットする．`get_fv_status` ディレクティブはハードウェアモジュールの電力状態の値を取得する．これらのディレクティブの中で使用される電力状態の記述は-1 から 100 に及ぶ整数値を用いる．-1 は電源遮断命令を実行し，0 はクロックゲーティングの実行を表す．1 から 100 までの値は，100 を最大周波数値とし，他の値は 100 を基準とした周波数値の割合で決定される．

API 標準解釈系は `fvcontrol` と `get_fv_status` ディレクティブをそれぞれ `oscar_fvcontrol()` と `oscar_get_fv_status()` に変換する．これらの関数の中でターゲットシステムのランタイムライブラリを利用し，電力制御を実現する．

3.3 WFI による疑似クロックゲーティング

ここでは OSCAR API を用いて OSCAR ランタイムライブラリに実装された疑似クロックゲーティングについて説明する．

現状の Android デバイスで使用可能な電力制御手法は `cpufreq` を利用する方法である．しかしこの手法は遅延時間が非常に大きく，この遅延時間は OSCAR コンパイラのランタイムの使用による電力削減の妨げとなる．そのため Android プラットフォームで電力削減を減らすために，新しいクロックゲーティング手法の開発をおこなった．

新しい手法では ARM 社のアーキテクチャに提供されている WFI(Wait For Interrupt) 命令 [4] を用いた．WFI 命令は演算するプロセスが無い状態であるとヒント情報としてプロセッサに通知する．この命令はプロセッサコアの演算処理を中断し，クロックを止める．

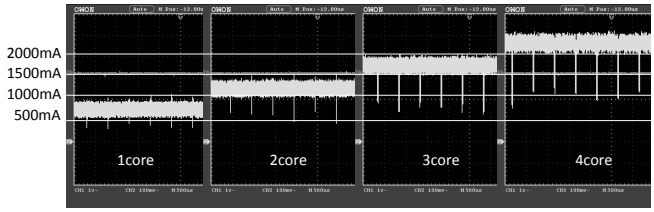
具体的には，割り込みあるいはデバックイベントが生じるまで，WFI 指示は，新しいプロセスのどんな指示も受け付けず，クロックを停止させる．

ランタイムライブラリ中の待機処理電力最小化のため，WFI の特性を利用する．そのために，WFI 命令を 500[us] の間隔で使用できるように，アンドロイドの linux カーネルに修正を施した．

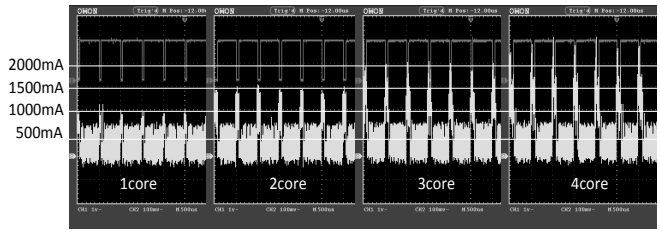
図 1 は疑似クロックゲーティング適用前後の各コアの電流値の測定結果を示している．

図 1-(a) は疑似クロックゲーティングなしの場合の各コアの電流値の測定結果を示している．この図からコアの使用数が増加するに応じて，電流量が増加していることが伺える．しかし疑似クロックゲーティングを適用した場合の測定結果の図 1-(b) を見るとコア数の増加をした場合であっても， 電流値は 500[mA] 以下を保つことが確認できる．また，図 1-(b) は今回の手法が 500[us] 間隔で停止させることを示している．

`cpufreq` を用いた数 [ms] のオーバーヘッドが観測できるパワーゲーティングと比較すると，今回の疑似クロックゲーティングは高速で正確であるといえる．この新手法をランンタイムライブラリの中に実装することにより，Android プラットフォーム上でより高い電力削減を実現



(a) 疑似クロックゲーティングなし



(b) 疑似クロックゲーティングあり

図 1 疑似クロックゲーティング適用前後の動作比較

する．

3.4 GPIO による電力測定手法

ここでは，ODROID-X2 上の GPIO(General Purpos Input Output)[16] ピンを利用した，正確な電力測定方法について述べる．

GPIO ピンはチップ上の一般的な入出力ピンで，通常組み込みシステムなどで利用される．通常は、LED, スイッチ, 周辺装置との通信に利用され、場合によりシステム上でデバックか割り込みにも利用される。GPIO ピンはソフトウェア上からランタイムによってコントロールすることができる．また，挙動の変化は GPIO ピンの電圧値が 0v から I/O 電圧値に変化に変化することで観測ができる．

この GPIO の特徴を使用する GPIO API 制御命令をランタイムに実装した．GPIO API はカーネルを経由して GPIO の制御レジスタに 0 または 1 を設定することで GPIO ピンの電圧を I/O 電圧値に変化させる．

図 2 は GPIO の使用例である．

図 2-(a) はアプリケーションに GPIO 制御命令を挿入したプログラム例を示している．関数 `gpio_value` は GPIO の状態を変化させる．変数 `gpio` は GPIO のピン番号を指定し，第二引数に GPIO レジスタに書き込む値を指定する．

図 2-(b) は図 2-(a) の電力波形図と GPIO の状態値を表す．上が GPIO の波形図で，下が電力波形図である．

図 2-(a) の (i) と (ii) の状態変化と，図 2-(b) の (A) と (B) の状態変化が一致していることがわかる．これは GPIO 制御命令の挿入箇所が正確な位置で一致していることを示している．よって，GPIO を使用することにより，プログラムと電力測定の間の正確な同期が確認できると言える．

4. 評価環境

この章では電力測定の際に用いる環境についての概要を

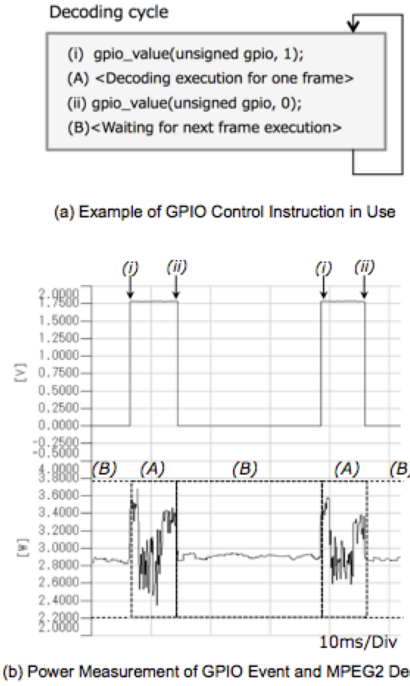


図 2 GPIO 制御命令例と GPIO 制御命令の電力波形図

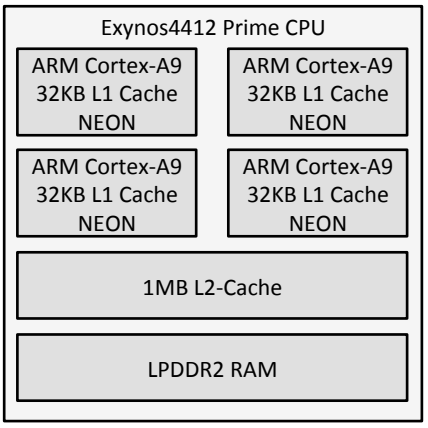


図 3 Exynos4412 のブロック図

説明する．様々な種類のチップを搭載した各種スマートデバイスが存在するが，チップの電力を測定できるプラットフォームは稀であり，民生品市場には存在しない．そのため，Android プラットフォームにおいて電力測定環境を構築しなければならない．

本稿では，電力測定環境の構築に ODROID-X2 を使用した．ODROID-X2[11] は Samsung の Exynos4412[17] のチップを積んだ開発ボードである．図 3 にこのチップのブロック図を示す．

ODROID-X2 に実装されているチップについて説明する．Exynos4412 は Samsung 社が開発したチップで ARM 社の最大周波数 1.7GHz の Cortex-A9 が 4 つと，1MB の共有 L2 メモリ，2GB のデュアルチャネルの LPDDR2 RAM が搭載されている．周波数と電圧は個別のコアで制御はできず，チップ一括で同一の値を持つ．

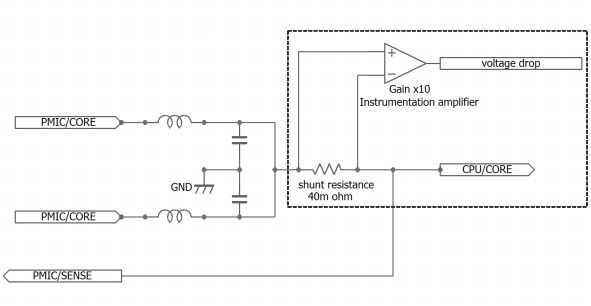


図 4 ODROID-X2 の修正回路図

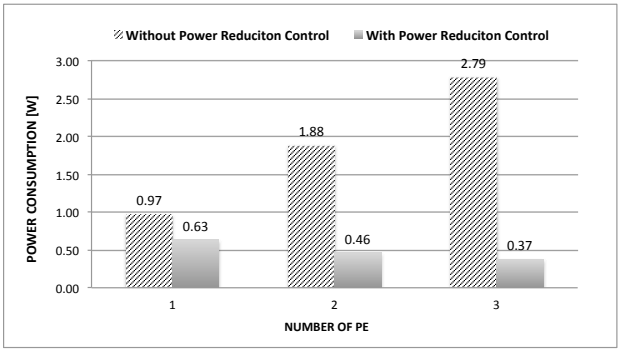


図 5 MPEG2 Decoder の消費電力評価

また，今回測定に使用した Android のバージョンは 4.2.0 を利用している。

今回測定に利用した ODROID-X2 開発ボードは電力測定用に設計されていない．そのため, CPU と CPU の動力源コントローラーとして動作をする PMIC(Power Management IC)[18] の間の電力を測定できるようにした．

ODROID-X2 の PMIC はバッテリー，コア，メモリ，割り込みコントローラー，アクセラレータといった CPU 上のユニットへの電源供給の管理を行う．

PMIC に接続された修正回路図を図 4 の点線の中に記載する．修正適用内容は PMIC と CPU の接続の間に 40[mΩ] のシャント抵抗を加え，10 倍のアンプを加えた．この 10 倍のアンプによりシャント抵抗間の電圧差が明確化され，正確な測定ができるようにした．これにより Android 環境でコアに供給される電圧，電流の測定を可能にした．

4.1 評価アプリケーション

この章では評価に用いるリアルタイムアプリケーションの概要について説明する．

4.1.1 MPEG2 Decoder．

MPEG2 デコーダは標準ビデオコーディングアプリケーションである．

このアプリケーションの並列化にはスライスの利用とマクロブロック並列化を適用している．

MPEG2 デコーダのデッドラインとしてはさらに，Android の描画環境として最高処理値である 60[fps](1 フレームあたり 16.6[ms]) を設定している．

4.1.2 Optical Flow．

Optical Flow は OpenCV[19] のベンチマークアプリケーションである．

このアプリケーションは時間的に連続な動画像を入力とし，画像中の物体の速度場を求める計算である．入力画像を 16x16 のブロックに分割し，各ブロックに対して連続する 2 フレーム間で対応する点を探索して速度ベクトルを生成する．

OSCAR コンパイラでは画像の横方向の処理を 1 つの粗粒度タスクとし，画像の縦方向に相当するループにおいてループレベルの並列性を抽出する．デッドラインとしては 30[fps](1 フレームあたり 33.3[ms]) を設定している．

5. 評価結果

この章では Optical Flow の電力制御結果について述べる．

この評価では，各ベンチマークアプリケーションは OSCAR コンパイラで自動並列化がされており, また, OSCAR API によって電力制御を実現している．さらに，章 3.3 で説明した疑似クロックゲーティングを OSCAR ランタイムライブラリに適用し，利用している．すべての測定は Section 3.4 で説明した GPIO を用いた電力測定手法が適用されている．

電力制御のため OSCAR コンパイラには FULL(1704[MHz])，MID(900[MHz])，LOW(400[MHz])，そして VLOW(200[MHz]) といった周波数パラメータが設定されている．さらに，Android 上の `cpufreq` ガバナーは OSCAR による電力制御を適用しない場合 `ondemand` として設定し，OSCAR コンパイラによる電力制御を適用する場合 `userspace` を利用した．

5.1 MPEG2 デコーダ

MPEG2 デコーダの評価結果について述べる．図 5 は各コア数に対応する MPEG2 デコーダの消費電力について示している．今回の測定では 1PE(Processor Element) において 0.97[W] から 0.63[W]，2PE において 1.88[W] から 0.46[W]，3PE において 2.79[W] から 0.37[W] に減少する結果が得られた．

図 6 は MPEG2 デコーダを ODROID-X2 で逐次実行させた場合の電力波形図を示している．図 6-(a) は OSCAR による電力制御なしの電力波形図を示している．この図からわかるように，最大周波数 (1704[MHz]) で動作をして，`ondemand` ガバナーにより，デッドラインまでの待機時間中は最大周波数 (1704[MHz]) よりも低い電力で動作していることがわかる．

図 6-(b) は OSCAR による電力制御ありの電力波形図を示している．OSCAR コンパイラではデッドラインまで最

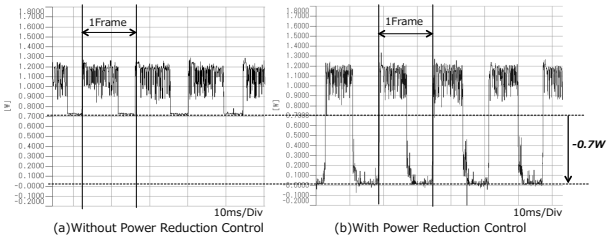


図 6 1PE MPEG2 Decoder の電力波形図

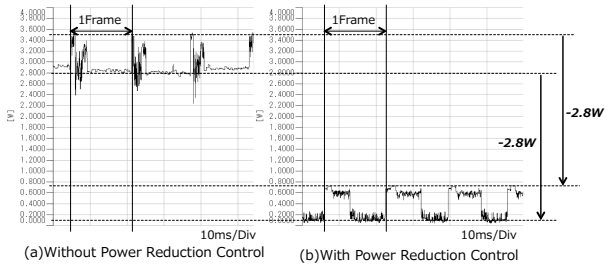


図 7 3PE MPEG2 Decoder の電力波形図

大周波数 (1704[MHz]) で動作をするという解析をし，デッドラインまで待機をするという処理を行う．デッドラインまでの待機処理には，疑似クロック制御を用いて消費電力値はほぼ 0[W] になるまで下がっている．

図 7 は ODROID-X2 で MPEG2 デコーダを 3PE で動作できるように並列化し，3PE で実行させた場合の電力波形図を示している．図 7-(a) は OSCAR コンパイラによる電力制御コードが含まれない場合の電力波形図を示している．図 7-(b) は OSCAR コンパイラによる電力制御ありの電力波形図を示している．

この図 7 からわかるように，最大周波数で動作し，デッドラインまでの待機処理では低い電力値で待機していることが伺える．OSCAR コンパイラで 3 並列化した MPEG2 デコーダは十分な並列性を抽出できているため，MID で動作し，デッドラインまで待機処理を行う．

この結果からアプリケーションレベルにおいて並列化による電力削減の重要性が述べられる．さらに， 図 6-(b) より，デッドラインまでの待機処理部分がほぼ 0[W] で実現ができていることが確認できる．

5.2 Optical Flow

Optical Flow の評価結果について述べる．図 8 は各コア数に対応する Optical Flow の消費電力について示している．今回の測定では 1PE において 0.95[W] から 0.72[W], 2PE において 1.50[W] から 0.36[W], 3PE において 2.23[W] から 0.30[W] に減少する結果が得られた．

図 9 は Optical Flow を ODROID-X2 で逐次実行させた場合の電力波形図を示している．

図 9-(a) は OSCAR による電力制御なしの電力波形図を示している．この図からわかるように，最大周波数

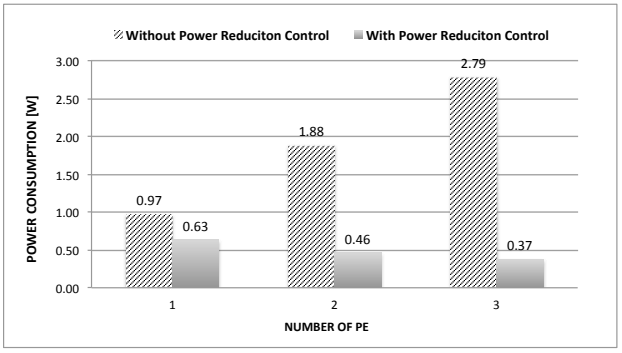


図 8 Optical Flow の消費電力評価

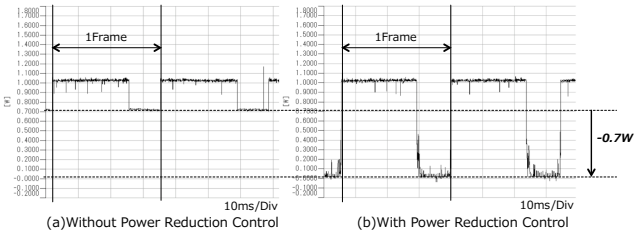


図 9 1PE Optical Flow の電力波形図

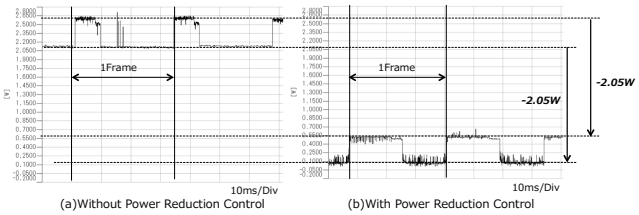


図 10 3PE Optical Flow の電力波形図

(1704[MHz]) で動作をして，ondemand ガバナーにより，デッドラインまでの待機時間中は最大周波数 (1704[MHz]) よりも低い電力で動作していることがわかる．

図 9-(b) は OSCAR による電力制御ありの電力波形図を示している．OSCAR コンパイラではデッドラインまで最大周波数 (1704[MHz]) で動作をするという解析をし，デッドラインまで待機をするという処理を行う．デッドラインまでの待機処理には，疑似クロック制御を用いて消費電力値はほぼ 0[W] になるまで下がっていることが伺える．

図 10 は ODROID-X2 で Optical Flow を 3PE で動作できるように並列化し，3PE で実行させた場合の電力波形図を示している．図 10-(a) は OSCAR コンパイラによる電力制御コードが含まれない場合の電力波形図を示している．図 10-(b) は OSCAR コンパイラによる電力制御ありの電力波形図を示している．

この図 10 からわかるように，最大周波数で動作し，デッドラインまでの待機処理では低い電力値で待機していることが伺える．OSCAR コンパイラで 3 並列化した Optical Flow は十分な並列性を抽出できているため，LOW で動作し，デッドラインまで待機処理を行う．この結果からアプリケーションレベルにおいて並列化による電力削減の重要

性が述べられる．さらに，図 9-(b) と同様に，デッドラインまでの待機処理部分がほぼ 0[W] で実現ができていることが確認できる．

6. まとめ

本稿では OSCAR コンパイラによる Odroid-X2 を用いた Android プラットフォーム上での電力制御について述べた．さらに，WFI による疑似クロック制御手法を実装した．また，電力測定環境はプログラムと電力の正確な対応付けが可能になるよう，GPIO を用いて測定を行った．MPEG2 デコーダでは，1pe において 0.97[W] から 0.63[W] に減少し 24.3%の電力削減が，2pe において 1.88[W] から 0.46[W] に減少し 75.5%の電力削減が，3pe において 2.79[W] から 0.37[W] に減少し 86.7%の電力削減が，実現できた．

Optical Flow では，1pe において 0.95[W] から 0.72[W] に減少し 24.2%の電力削減が，2pe において 0.36[W] から 1.50[W] に減少し 75.9%の電力削減が，3pe において 0.30[W] から 2.23[W] に減少し 13.5%の電力削減が，減少することが確認できた．この結果は疑似クロック制御手法の有用性を示すとともに，Android プラットフォームにおいて OSCAR コンパイラの低消費電力化の有用性があることを示す結果が得られた．また，Android プラットフォーム上でマルチコアによる低消費電力化の有用性が確認できた．

参考文献

[1] Taylor, M., Kim, J., Miller, J. and Wentzlaff, D.: THE RAW MICROPROCESSOR: A COMPUTATIONAL FABRIC FOR SOFTWARE CIRCUITS AND GENERAL-PURPOSE PROGRAMS, *Micro, ...*, pp. 25–35 (2002).
[2] Hammond, L., Hubbert, B. and Siu, M.: THE STANFORD HYDRA CMP, *Micro, ...*, pp. 71–84 (2000).
[3] Friedrich, J. and McCredie, B.: Design of the Power6 microprocessor, pp. 96–97 (2007).
[4] ARM Corporation: Cortex-A9 Technical Reference Manual.
[5] Jeff, B.: Advances in big . LITTLE Technology for Power and Energy Savings, No. September, pp. 1–11 (2012).
[6] NVIDIA Corporation: Whitepaper NVIDIA ®Tegra ™Multi-processor Architecture, pp. 1–12.
[7] Samsung Electronics Co., L.: White Paper of Exynos 5, Vol. 1, No. 1, pp. 1–8 (online), DOI: 10.5663/aps.v1i1.10138 (2011).
[8] Kasahara, H., Obata, M. and Ishizaka, K.: Automatic coarse grain task parallel processing on smp using openmp, *Workship on Lan- guages and Compilers for Parallel Computing*, pp. 1–15 (2001).
[9] Obata, M., Shirako, J., Kaminaga, H., Ishizaka, K. and Kasahara, H.: Hierarchical Parallelism Control for Multigrain Parallel Processing, *Lecture Notes in Computer Science*, Vol. 2481, pp. 31–44 (2005).
[10] Kimura, K., Mase, M., Mikami, H., Miyamoto, T., Shirako, J. and Kasahara, H.: OSCAR API for Real-time Low-Power Multicores and Its Performance on Multicores and SMP Servers, *Lecture Notes in Computer Science*, pp. 188–202 (2010).
[11] : ODROID-X2.
[12] Google: Android Developers.
[13] : CPU hotplug Support in Linux(tm) Kernel.
[14] Obata, M., Shirako, J. and Kaminaga, H.: Hierarchical parallelism control for multigrain parallel processing, pp. 31–44 (online), available from ⟨http://link.springer.com/chapter/10.1007/11596110_3⟩ (2005).
[15] Shirako, J., Oshiyama, N., Wada, Y., Shikano, H., Kimura, K. and Kasahara, H.: Compiler Control Power Saving Scheme for Multi Core Processors, *Lecture Notes in Computer Science*, pp. 362–376 (2007).
[16] : GPIO Interfaces.
[17] SUMSUNG ELECTRONICS: Samsung Exynos 4 Quad (Exynos 4412) RISC Microprocessor User's Manual, No. October (2012).
[18] : Samsung Semiconductors Global Site.
[19] : Opencv.