Vol.2015-SLDM-170 No.20
Vol.2015-EMB-36 No.20
2015/3/6

OSCAR自動並列化コンパイラを用いたリアルタイム動画像アプリケーションのHaswellマルチコア上での低消費電力化

飯塚 修平[1,a)]　山本 英雄[1]　平野 智大[1]　岸本 耀平[1]　後藤 隆志[1]　見神 広紀[1]　木村 啓二[1]
笠原 博徳[1]

概要：スマートフォンやノートパソコンといったモバイル端末からデータセンタで利用されるサーバーマシンまで, あらゆる計算機において消費電力の削減が最重要課題となっている. これは、消費電力の削減によりモバイル機器においてはバッテリー持続時間の延長により利便性が大幅に向上し, また サーバーマシンにおいては膨大な電力コストや空調コストの削減が実現できるからである. これらの計算機は高性能かつ低消費電力を実現するためにマルチコアプロセッサを搭載したものが主流となっている. しかしながらマルチコアの資源を有効活用してこれらを実現するためには, プログラムの並列化が不可欠であり手動で行うには膨大な工数を必要とする. 本稿では, 医用・防犯・個人認証・車載などで広く利用されているリアルタイム物体認識処理に対して, OSCAR 自動並列化コンパイラによる DVFS 及び clock gating による電力制御を適用し, 現在幅広く利用されている Intel Haswell Core i7-4770K マルチコア上で評価した. Intel Haswell マルチコア上で, Web カメラからの画像の入力・人の顔の認識処理・画面描画というリアルタイムなシステム全域における消費電力の削減を行ったところ, 1PE 逐次実行では電力制御なしの場合の 31.06[W] から電力制御ありの場合では 28.74[W] に、3PE で並列化実行した場合では電力制御なし場合のの 41.73[W] から電力制御の場合では 17.78[W] に消費電力を削減したことが確認され，物体認識処理におけるマルチコア用のコンパイラ自動電力制御の有用性が確認できた.

1. はじめに

スマートフォンやノートパソコンといったモバイル端末ではバッテリー持続時間が利便性に直結すること，及びデータセンタで利用されるサーバーにおいては膨大な電力コスト・空調コストが問題となっていることから，消費電力の削減の需要が著しく高まっている．これらの計算機は高性能を実現するためにマルチコアプロセッサを搭載したものが主流となっているが，上記の背景からマルチコアでの処理性能の向上に加えて、いかに消費電力の削減を行うかが求められている．

Intel 社の第 4 世代 Intel Core プロセッサとして現在広く普及している Haswell マルチコアでは FIVR(Fully Integrated Voltage Regulator) の実装により電源レギュレータをオンチップで搭載することで高速な電圧の変更を可能とし，低消費電力化を実現している [1]．Linux 環境下での動作周波数や電圧の変更は CPUFreq 機構 [2] の ondemand governor が管理しているが，より高い電力効率を実現するためには，ソフトウェアの並列化やハードウェアとソフトウェアの協調によりマルチコアの性能を最大限に引き出す必要がある．

一般に，プログラムの並列化を OpenMP[3] や MPI を用いて手動で行う場合は膨大な工数が必要となる．このような問題に対して，コンパイラを用いてプログラムから多くの並列性を抽出し，自動で並列化することができれば作業が効率的になり，高い生産性に寄与できる．

自動並列化機能を持つコンパイラとして，OSCAR 自動並列化コンパイラ [4] が開発されている．OSCAR 自動並列化コンパイラでは自動並列化機能に加え，DVFS やクロックゲーティング，パワーゲーティング指示文による電力の最適化機能も備えており，ソフトウェア側から電力値を最適化することが可能である．

一方，物体認識は医用・防犯・個人認証・車載など多くの応用先が存在しており，これら認識処理の低消費電力なリアルタイム処理が求められている．認識処理を含めた画像処理ライブラリの 1 つとして OpenCV が近年広く利用されている [5]．OpenCV の顔画像認識処理に対して OSCAR コンパイラにより並列化及び低消費電力化を適用し，さらに Web カメラから取り込んだ画像を入力として人の顔と

[1] 早稲田大学
Waseda University.
[a)] shuhei@kasahara.cs.waseda.ac.jp

座標の認識処理を行い，ディスプレイへの描画も含めた環境で評価を行った．

本稿では，第 2 章で OSCAR 自動並列化コンパイラについて，第 3 章で Intel Haswell プラットフォームでの電力制御について，第 4 章で OpenCV 顔認識処理の低消費電力化について，第 5 章で電力値評価結果について述べる．

2. OSCAR 自動並列化コンパイラ

OSCAR 自動並列化コンパイラは，マルチグレイン並列化，キャッシュ及びローカルメモリ最適化，自動電力制御を可能とする自動並列化コンパイラである [6], [7], [8]．本章では OSCAR コンパイラのマルチグレイン並列化，自動電力制御機能，及び OSCAR API について述べる．

2.1 マルチグレイン並列化

マルチグレイン自動並列化では，複数の関数呼び出しやループ間に存在する粗粒度並列性，ループイテレーション間に存在する中粒度並列性，ステートメント間に存在する近細粒度並列性を組み合わせて自動で並列化を行う [4], [9]．OSCAR コンパイラは Parallelizable C や Fortran で記述された逐次ソースコードを入力とし，並列化済みソースコードを出力する．Parallelizable C とはポインタ利用の制約等を定めることで自動並列化を可能にする C 言語の記述手法である．ソースプログラムから 3 種類のマクロタスク (MT)，すなわち基本ブロック (BB)，ループなどの繰り返しブロック (RB)，関数呼び出しの (SB) に分割し，マクロタスク内部でも分割を行うことで階層的なマクロタスクを生成する．粗粒度並列化では，マクロタスク間の依存関係とコントロールフローを解析し，同時実行可能なマクロタスクをプロセッサに割り当てるスケジューリングを行う．コンパイラは条件分岐や実行順不確定性の高い並列化階層にはダイナミックスケジューリングコードを合わせて並列化コードを生成し，そうでない場合はスタティックスケジューリングを適用する．

2.2 OSCAR 自動並列化コンパイラによる自動電力制御

OSCAR コンパイラは，マクロタスクのスケジューリング結果に基づいて電力制御を行う [8]．本稿で対象とする OpenCV の物体認識のようなリアルタイム動画像アプリケーションでは，周期的にデッドラインが到達する．OSCAR コンパイラは与えられたデッドライン情報とマクロタスクのスケジューリング結果を元に，動作周波数やクロックゲーティング，パワーゲーティングのスケジューリングを適用し，その結果を第 2.3 節で述べる OSCAR API の指示文として，出力する並列化プログラム中に記述する．本電力制御の適用例として，マクロタスクのスケジューリング結果のうち動作周波数の低下に対して最も電力削減が大きくなるマクロタスクについてデッドラインを保証する範囲で DVFS を適用し，またデッドラインまでの待機時にはクロックゲーティングを行う．

表 1 Intel Haswell プラットフォーム 評価環境の構成

	Intel Haswell プラットフォーム
Motherboard	H81M-A
CPU	Intel Haswell Corei7-4770K
The number of cores	4
Frequency	3500 [MHz]
L1 cache size	4×32 [KB] instruction chaches 4×32[KB] data chaches
L2 cache size	4×256 [KB]
L3 cache size	8 [MB]
DDR	8×2 [GB]
OS	Ubuntu 13.10

2.3 OSCAR API

OSCAR API[10] は様々なマルチコアプラットフォーム上で並列化や電力削減を行うために定義された API である．並列実行，電力制御，リアルタイム制御，キャッシュ制御，データのメモリ配置，DMA によるデータ転送，グループバリア同期，及びアクセラレータ制御指示文から構成されている．OSCAR API は OpenMP をベースにして仕様が定義されているため，OSCAR コンパイラにより並列化され，OSCAR API 指示文が挿入されたプログラムは，OpenMP に対応したコンパイラで並列実行可能である．

OSCAR API の提供する fvcontrol や get_fv_status といった電力制御指示文等を含むプログラムをバイナリに変換する場合は，OSCAR API 解釈系を利用する．OSCAR API 解釈系が OSCAR API の指示文から fvcontrol() や get_fv_status() といったランタイム関数への変換を行うことで，逐次コンパイラさえ用意されていれば様々なマルチコアプラットフォーム上で並列化や低消費電力化が可能となる．

3. Intel Haswell プラットフォームでの電力制御

本章では，まず本稿の低消費電力化のターゲットである Intel Haswell プラットフォームの評価環境の構成について述べる．次に DVFS やクロックゲーティングの方式など，電力制御手法について述べ，最後に電力値の測定方法について述べる．

3.1 Intel Haswell プラットフォーム 概要

Intel Haswell プラットフォームとして，本稿ではマザーボードは ASUS 社の H81M-A を利用し，プロセッサは Intel Haswell Corei7-4770K[11] を利用した．このプロセッサは 4 つのプロセッサコアから構成されており，DVFS は各プロセッサコア毎に行うことが可能であり，電圧値はチップ

一括で同一の値を持つ．最大動作周波数は 3500[MHz] で，200[MHz] 単位での動作周波数の変更が可能となっている．評価環境の構成を表 1 に示す．

3.2 評価対象プラットフォームの電力制御手法

現状で Linux 環境では，CPUFreq 機構による動作周波数及び動作電圧の動的制御が一般的であり，動作周波数の決定は CPUFreq 機構の Governor により行われている．Governor には動作周波数を最大に設定する performance や，動作周波数を最低に設定する powersave，負荷に応じて段階的に値を変更する conservative, 及びユーザが /sys/devices/system/cpu/cpuN/cpufreq/scaling_set_speed (N はコア番号を表す 0 以上の整数) を介して自由に動作周波数の値を設定できる userspace などが存在する．標準状態では ondemand governor が利用されている．ondemand governor では一定間隔毎に負荷状況を監視し，あるしきい値を超えた時点で動作周波数の変更を行う．

本稿が対象とするリアルタイム動画像アプリケーションのように数十ミリ秒ごとにデッドラインを持つプログラムの場合は，ondmend governor の一定間隔毎の負荷状況に基づく電力制御ではきめ細かい電力制御の適用が困難である．すなわち，アプリケーション側からの積極的な電力制御を行う必要がある．

本稿では動作周波数の段階として HIGH[3500MHz], MID[1800MHz], LOW[800MHz] を設定し，OSCAR コンパイラのマクロタスクのスケジューリング結果のうち動作周波数の低下に対して最も電力削減が大きくなるマクロタスクについてデッドラインを保証する範囲で MID や LOW を適用することで DVFS を実現している．電力制御時は Governor を userspace として，OSCAR API の指示文経由でアプリケーションから CPUFreq の制御を行うことにより動作周波数の変更を行っている．

また，Linux 環境における電源遮断に関しては，sysfs によるコアの online に電源状態を記述する方式が一般的であるが，sysfs による方式は遅延時間が大きいためアプリケーションによる電力制御には不向きである．この問題を踏まえ，デッドラインまでの待機処理時には MWAIT(Monitor Wait) 命令を用いたクロックゲーティング手法を適用した．MWAIT 命令を用いることで特定のクラスのイベントが発生するまで，コアを C state という低消費電力状態に 5[μs] 程度で遷移することが可能となる．MWAIT は特権命令であるため，新たにカーネルモジュールを作成し，ioctl() 関数を通じてユーザアプリケーションからアクセス可能としている [12].

3.3 電力値の測定手法

本稿で評価を行う Intel Haswell プラットフォームを構成するマザーボード H81M-A 及び Corei7-4770K プロセッサは電力値の測定機能をサポートしていない．電力値測定のために PMIC(Power Management IC) と CPU コアの間に 5[mΩ] のシャント抵抗を挿入し，シャント抵抗間の電位差とコアに供給される電圧を計測することで，電流値及び CPU の消費電力の測定を可能としている [13].

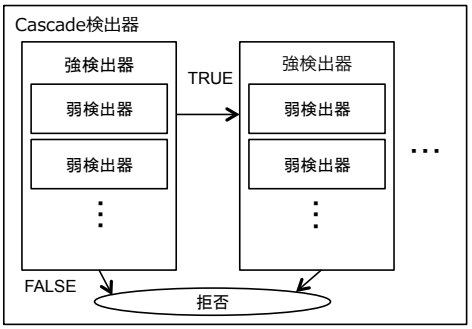

図 1 Cascade 検出器の構成

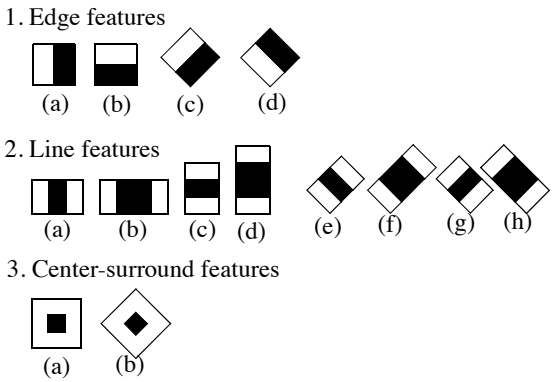

図 2 Haar-Like 特徴

4. OpenCV 顔認識処理の低消費電力化

本稿では, OpenCV の物体認識アプリケーションに第 3 章で述べた低消費電力化手法を適用した．本章では OpenCV 顔認識アプリケーションの概要と，低消費電力化の適用手法について述べる．

4.1 OpenCV 顔認識処理 アプリケーション概要

OpenCV は Intel 社が開発した画像処理ライブラリであり，オープンソースとして公開されている．物体認識の例としては人の顔認識や白線の認識など多岐に渡り，医用・防犯・個人認証・車載といった分野に応用されている．

OpenCV では Paul Viola によって提案され, Rainer Lienhart により改良された物体認識アルゴリズムが利用されており，本稿で取り上げる顔認識処理では事前に作成された検出器を元に顔認識を行う．検出器は図 1 のように，複数の強検出器を数珠つなぎに連結したカスケード検出器から構成される．強検出器内の弱検出器で Haar-Like 特徴と入力画像の矩形領域の積分値を比較することで物体認識を行い，人の顔であると判断された物体のみが次の検出器に

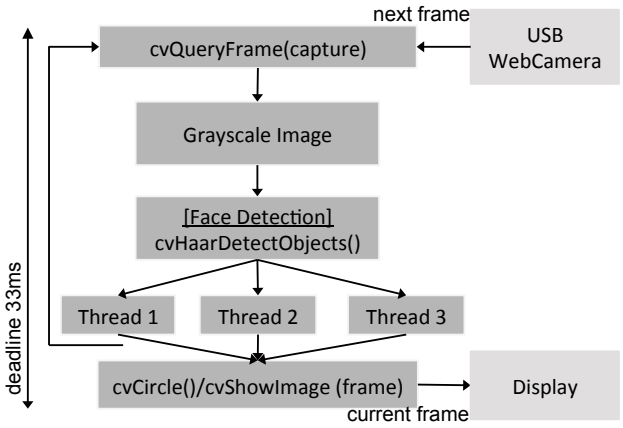

図 3 顔認識処理 システム全体構成

進む．最後の強検出器まで通過した物体のみが最終的に人の顔であると認識される [14]．強検出器は認識対象の画像を複数枚用いて機械学習を行い，弱検出器に重みを付けてブースティング技法により組み合わせることで作成される．図 2 は Haar-Like 特徴を示しており，白黒の矩形で表現される．2 つの領域の輝度差に注目し，人の目や鼻そして口のように，積分画像に明暗の特徴が見られる個所について，eage 特徴や Line 特徴，及び Center-surround 特徴を検出することが可能となる [15]．Haar-Like 特徴の位置，サイズ，種類の情報が弱検出器の役割を果たす．

次に，顔認識処理のシステム全体の構成を図 3 に示す．まず cvQueryFrame 関数を用いて USB Web カメラのキャプチャを入力画像として取得する．ここで，本環境の Web カメラの最大フレームレートは 30fps である．次に入力画像の積分画像を作成し，画像中の探索窓のサイズを変更しつつ，画像の x 軸と y 軸を移動させてそれぞれの領域に対しカスケード検出器を用いて顔認識を行う．人の顔であると判断された物体の数と座標を保持し，cvShowImage 関数により入力フレームをディスプレイへ描画する際に，人の顔と判断された物体が存在した場合は cvCircle 関数で全ての認識済みの顔に対して円の描画を行う．この処理を繰り返し行うことでリアルタイムに Web カメラからキャプチャを行い，顔認識結果をディスプレイに出力することが可能となる．

また，システム全体の実行時間に対して 89.1%の時間を占め，顔認識処理を行う cvHaarDetectObjects 関数に対しては OSCAR コンパイラを用いて並列性を抽出している．画像の xy 方向の探索はイタレーションごとに依存のない DOALL ループであり，Parallelizable C 規約を満たす記述にチューニングを行うことで，コンパイラによる自動並列化が可能となる．

表 2 は図 3 のシステムで画面解像度を 352×288 として顔認識処理を行った時の顔認識時間及びシステム全体でのフレームレートをまとめたものである．カメラのキャプチャ速度が制約となり，システム全体としてのフレームレートは 30fps 付近から向上しないが，並列化対象である顔認識部分の処理速度が向上し，デッドライン制約下で周波数制御を行う余地が生まれていることが確認できる．

表 2 OpenCV 顔認識処理の並列化後の実行時間比較

	顔認識時間 [ms]	フレームレート [fps]
1PE	22.82	29.85
3PE	11.23	29.88

4.2 低消費電力化手法

顔認識処理の並列化により処理時間が短縮され，デッドライン制約下での周波数制御を行う余地が生じる．本節では顔認識アプリケーションの低消費電力化手法について述べる．

第 3 章で述べた 4 種類の周波数段階，HIGH[3500MHz]，MID[1800MHz]，LOW[800MHz]，そして MWAIT 命令によるクロックゲーティングを図 3 の顔認識アプリケーションに適用した．システム全体として 1 フレームを 33ms で描画まで行うために，実行時間のプロファイル情報を基に顔認識処理部分にデッドラインを設けている．OSCAR コンパイラはデッドラインとコスト情報を基にマクロタスクの周波数制御，デッドラインまでの待機時間の計算及びコアのアイドル状態の管理，逐次処理部分の不要なコアのクロックゲーティングを行う．

本稿の電力制御対象アプリケーションは，画面の描画も含んだシステムから構成されており，顔認識を始めとする計算処理と描画処理に大別できる．計算部分に DVFS を適用し低周波数で処理を行い，更に描画処理の時間まで加わるためにシステム全体としてデッドラインに間に合わなくなり，低周波数での動作が困難となる可能性がある．この問題を解決するために CPU Isolation[16] を利用している．CPU Isolation により core1 から core3 をユーザプログラムが自由に利用することを可能となる．顔認識処理のスレッドを core1 から core3 にコアバインドし，描画処理を core0 で行うことで，顔認識処理のスレッドは画面描画とは非同期に，つまり現在のフレームの描画処理と次のフレームの顔認識処理とを並行して動作させることで画面描画に必要とされるコストを実質無いものと見積もり，計算部分だけで 33ms のデッドラインを利用可能とした．

5. 評価結果

本章では，第 3 章で述べた Intel Haswell プラットフォーム上での OpenCV 顔認識処理の電力値の評価結果を述べる．

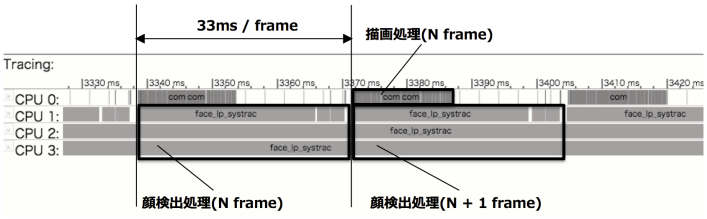

図 4 OpenCV 顔認識処理の 3PE 実行における顔認識処理部分と描画部分

5.1 消費電力値評価

まず，第 4 章で述べた顔認識処理と描画処理との同時実行の動作状況を確認するために，Systrace[17] を用いて各コアの負荷状況の観測を行った．図 4 は OSCAR コンパイラによる電力制御と，スレッドのコアバインディングを適用して core1〜3 で顔認識処理を行い，core0 で描画処理を行った時の Systrace 結果を示したものである．

Systrace 結果から，まず顔認識処理が core1〜3，描画処理が core0 で動作している様子が確認できる．さらに顔認識処理の後続の描画処理と，次のフレームの顔認識処理とを並行して処理を行っている様子も確認できる．結果として，1 フレームにつき顔認識処理部分のみで 33ms のデッドラインを利用しており，システム全体を通しても 1 フレームあたり 33ms のデッドラインを満たしながら画面描画まで行っていることが確認できる．

次に，OSCAR コンパイラを用いて DVFS やクロックゲーティングなどの消費電力削減手法を適用した OpenCV 顔認識処理の Intel Haswell プラットフォーム上での消費電力と，プログラム側からの消費電力の制御を行わない場合での消費電力を計測した．いずれの環境ともシステム全体でのフレームレートが 30fps となるように 33[ms] のデッドライン制約を設けている．また，入力及び出力画面解像度は 352×288 である．OSCAR コンパイラによる電力制御を行う場合は，CPUFreq の governor を userspace とし，ユーザプログラムからの DVFS 及びクロックゲーティングを受け付けている．一方で，電力制御を行わない場合は，ondemand governor による負荷状況に応じた動作周波数の決定を用いた．動作環境は表 1 のプラットフォームを用いた．また，core0 で行う描画処理に関しては DVFS の適用範囲外であるため，governor を userspace として動作周波数を 800[MHz] に設定している．OSCAR コンパイラによる電力制御ありの場合となしの場合における，OpenCV 顔認識処理のシステム全域における消費電力の計測結果を図 5 に示す．

1PE の電力制御なしと電力制御ありの消費電力を比較すると，31.06[W] から 28.97[W] となり 7.05%の消費電力を削減したことが確認できる．3PE の電力制御なしと電力制御ありの消費電力とを比較すると，41.73[W] から 17.78[W] に削減され，57.40%の消費電力を削減できたことが確認できた．OSCAR コンパイラによる並列化や電力の最適化を行っていない 1PE 電力制御なしと 3PE 電力制御ありの消費電力を比較しても，31.06[W] から 17.78[W] に削減され，42.76%の消費電力を削減したことが確認できた．

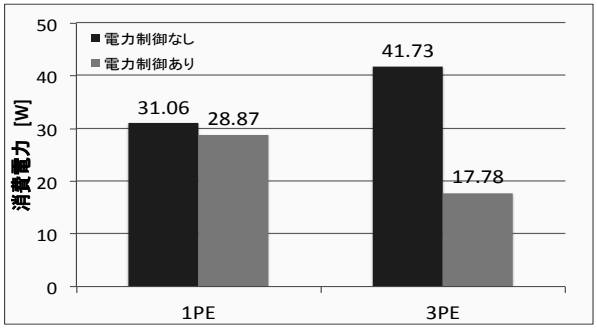

図 5 OpenCV 顔認識処理の消費電力評価結果

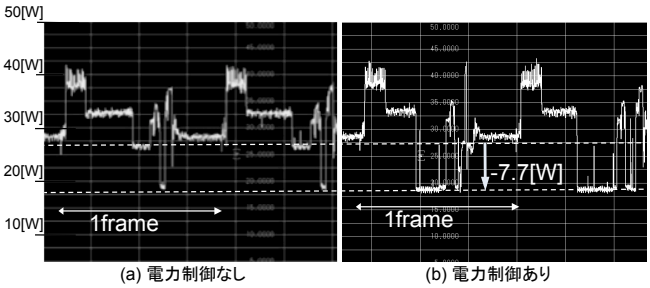

図 6 OpenCV 顔認識処理の消費電力波形図 1PE

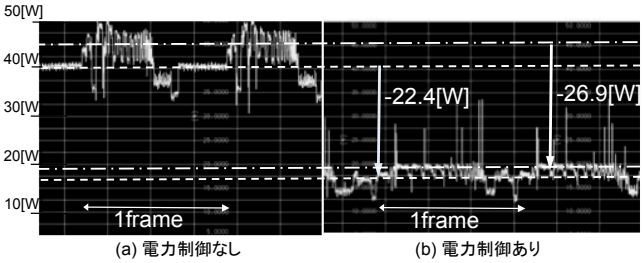

図 7 OpenCV 顔認識処理の消費電力波形図 3PE

図 6 は OpenCV 顔認識処理を 1PE で動作させた時の消費電力波形図を示しており，(a) は電力制御なし,(b) は電力制御ありの波形図を示している．両波形図において周期的に波形にピークが見られるが，これは顔認識計算処理と描画処理とを同時に実行している個所で現れている．また，両波形を比較すると，電力制御ありの時にデッドラインまでの待機の部分で，ondemand governor による待機時間と比較して 7.7[W] ほど低下しており，これが図 5 で 1PE の間の電力差となっていることが確認できる．

図 7 は 3PE で動作させた時の消費電力波形図を示している．電力制御なし，すなわち ondemand governor による周波数制御では 1 フレーム中の消費電力が 40〜45[W] 付近となる．一方 OSCAR コンパイラによる電力制御を行った場合，周波数段階として MID が適用され，デッドライン

を満たしつつも 20[W] 未満の消費電力での顔認識処理の動作が可能であることが確認できる．このことから、認識処理において、OSCAR コンパイラを用いてアプリケーション側からの DVFS による低周波数での処理の実行，及びデッドラインまでのクロックゲーティングを適用することが消費電力の削減に有用であることが確認できた．

6. おわりに

本稿では，Web カメラによる画像入力，顔認識処理，画面描画という一連のシステムに対して OSCAR 自動並列化コンパイラを用いて電力制御を行い，Intel Haswell マルチコア上での消費電力の評価を行った．画面の描画も含めたシステム全体でのデッドラインを保証するために，顔認識処理のスレッドは画面描画とは非同期に実行可能とし，現在のフレームの描画処理と次のフレームの顔認識処理とを並行動作可能にすることで，計算部分だけで 33ms のデッドラインを利用可能とした．Haswell マルチプラットフォーム上で電力制御を適用することで，消費電力が 1PE において 31.06[W] から 7.05%削減されて 28.87[W] となり，3PE において 41.73[W] から 57.40%削減されて 17.78[W] となり，1PE 電力制御なしと比較しても 3PE 電力制御ありでは 31.06[W] から 17.78[W] に削減され，42.76%の消費電力を削減したことが確認できる．この結果から，Haswell マルチコアプラットフォームでの OSCAR コンパイラの電力制御の有用性及び，カメラ画像による入力や画面描画を含む顔認識処理のシステム全域に対して OSCAR コンパイラを用いた電力制御が有用であることが確認できた．

参考文献

[1] Intel: Intel Website.
[2] Brodowski, D.: CPU frequency and voltage scaling code in the Linux(TM) kernel.
[3] OpenMP: OpenMP Website.
[4] 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).
[5] OpenCV: OpenCV Website.
[6] Ishizaka, K., Obata, M. and Kasahara, H.: Coarse Grain Task Parallel Processing with Cache Optimization on Shared Memory Multiprocessor, *Proc. of 14th International Workshop on Languages and Compilers for Parallel Computing (LCPC2001)* (2001).
[7] 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).
[8] 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*, Vol. 4339, pp. 362–376 (2007).
[9] Kimura, K., Wada, Y., Nakano, H., Kodaka, T., Shirako, J., Ishizaka, K. and Kasahara, H.: Multigrain Parallel Processing on Compiler Cooperative Chip Multiprocessor, *Proc. of 9th Workshop on Interaction between Compilers and Computer Architectures (INTERACT-9)* (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*, Vol. 5898, pp. 188–202 (2010).
[11] Intel: Intel Haswell Processor Corei7-4770K.
[12] : MWAIT:Monitor Wait(x86 Instruction Set Reference).
[13] Hirano, T., Yamamoto, H., Iizuka, S., Muto, K., Goto, T., Wake, T., Mikami, H., Takamura, M., Kimura, K. and Kasahara, H.: Hierarchical Parallelism Control for Multigrain Parallel Processing (2014).
[14] Viola, P. and Jones, M.: Rapid object detection using a boosted cascade of simple features, *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, Vol. 1, IEEE, pp. I–511 (2001).
[15] Lienhart, R. and Maydt, J.: An extended set of haar-like features for rapid object detection, *Image Processing. 2002. Proceedings. 2002 International Conference on*, Vol. 1, IEEE, pp. I–900 (2002).
[16] Intel: Intel Data Plane Development Kit.
[17] Fukui, D., Shimaoka, M., Mikami, H., Kimura, K., Kasahara, H. et al.: Tracing method of a parallelized program using Linux ftrace on a multicore processor, *Technical Report of IPSJ*, Vol. 2014, No. 6, pp. 1–6 (2014).