社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2009-ARC-181／2009-EMB-11 (11)
2009/1/14

マルチメディア応用ヘテロジニアスマルチコアアーキテクチャの評価

奥村大輔† ハシタムトゥマラ ウィシディスーリヤ† 松田岳久†
張山昌論† 亀山充隆†

† 東北大学大学院情報科学研究科
E-mail: {okumura, hasitha, take1010}@kameyama.ecei.tohoku.ac.jp,
{hariyama, kameyama}@ecei.tohoku.ac.jp

あらまし ヘテロジニアスプロセッサは異種のプロセッサの利用により全体の性能を向上させることが可能であり，画像処理や画像認識の分野で注目されている．しかしながら異なるコア間の利用におけるデータ転送がボトルネックとなり，十分な性能が発揮できない問題がある．本稿では4個のCPUと2個の動的再構成プロセッサからなるヘテロジニアスプロセッサRP1に画像認識アルゴリズムであるオプティカルフローおよびHOGを実装し，その評価を行った．動的再構成プロセッサである"Flexibe Engine/Generic ALU arrays(FE-GA)"はマルチメディアアプリケーション向けのアクセラレータとして動作する．処理速度向上のために，異なるコア間におけるデータの受け渡し時のオーバヘッドを減少させるタスク割り当て，処理するプロセッサに適したアルゴリズムへの変形を行った．これによりCPU単体の処理と比較しヘテロジニアスプロセッサではオプティカルフローでは30.3倍，HOGでは2.1倍の消費電力あたりの処理速度を向上することができた．

キーワード 動的再構成，リコンフィギャラブルアーキテクチャ，オプティカルフロー，HOG

Daisuke OKUMURA†, Hasitha MUTHUMALA WAIDYASOORIYA†, Takehisa MATSUDA†,
Masanori HARIYAMA†, and Michitaka KAMEYAMA†

† Graduate School of Information Sciences, Tohoku University
E-mail: {okumura, hasitha, take1010}@kameyama.ecei.tohoku.ac.jp,
{hariyama, kameyama}@ecei.tohoku.ac.jp

Abstract Heterogeneous processors are attracted by the image processing and recognition applications due to their capability of drawing strengths of different cores to improve the overall performance. However, data transfer between different cores caues serious performance degradetion. In this paper, we evaluate optical flow and Histogram of Oriented Gradients(HOG) on a heterogeneous multicore processor RP1 that has four CPUs and two reconfigurable processors. The reconfiguralbe processor called Flexible Engine/Generic ALU arrays (FE-GA) works as an accelerator for multimedia applications. To improve the total performances, we present task allocation to reduce the data transfers between different cores and modification of an algorithm suitable for the FE-GA. The performances per watt for optical flow and HOG are improved by 30.3 times and 2.1 times in comparison with a single CPU.

Key words Dynamic Reconfiguration, Reconfigurable architecture, Optical flow, HOG

1. まえがき

情報家電や高安全自動車などの実現には低消費電力な画像処理プロセッサが重要となる．従来の汎用CPUを用いて画像認識を行う場合には消費電力が大きく，携帯機器での利用は困難であった．解決方法としては並列で処理を行う専用プロセッサを用いる方法が考えられるが，画像認識は一般的に並列処理が可能な大規模演算と条件分岐などの複雑な制御を行う必要のある演算の両方が必要とされる場合が多い．そのため画像認識では複雑な制御を行うCPUと並列演算を行う動的再構成プロセッサなどの異なるコアの組み合わせからなるヘテロジニアスアーキテクチャを用いることが有効であると考えられる．ヘテロジニアスアークテクチャを用いる場合，演算の種類により適したコアに演算を割り当てることにより消費電力の低減と処理

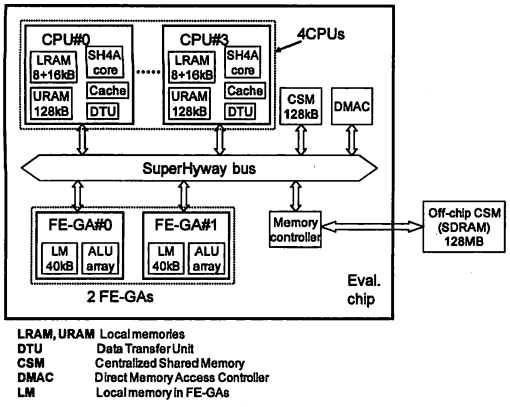


図 1　RP1 の全体構造

速度の向上が見込まれる．一方で問題点として，異種コア間に対する最適なタスク割り当て，異なるプロセッサ間でのデータの受け渡しにおいて発生するオーバヘッドなどがある．本稿ではヘテロジニアスマルチコアプロセッサ RP1 に対して画像認識のアルゴリズムであるオプティカルフローおよび Histgrams of Oriented Gradients(HOG) に実装を行い評価を行った．オプティカルフローの実装においては線形アドレッシング関数の制限によりデータの重複が発生し，データ転送量の増加が問題となる．本稿ではコンテクストの切り替えにより重複データの削減を行うとともに，複数コアへのタスク割り当てを行うことで消費電力あたりの処理速度を CPU 単体の場合と比較し消費電力あたりの処理速度を 30.3 倍に向上できることを示す．HOG の実装では，データ転送時間が全体の処理速度低下の原因となっている．異なるコア間のデータ転送の時間を削減するために，コア間でのタスク割り当てを行った．また，アルゴリズムの変更を行うことでヘテロジニアスプロセッサの仕様上の制限を解決し，全体で消費電力あたりの処理速度を 2.1 倍に向上できることを示す．

2. アーキテクチャ

本研究では文献 [1] で述べられているヘテロジニアスアーキテクチャを利用した．このアーキテクチャは 4 個の CPU(SH) と 2 個の"Flexible Engine/Generic ALU Arrays(FE-GA)"から構成されている．アーキテクチャの全体構成を図 1 に示す．CPU は 600MHz，FE-GA は 300MHz という比較的遅い周波数で動作する．CPU と FE-GA 間のデータ転送は"Super Highway"バスを介して行われる．チップ外部には 128MB の SDRAM が存在しチップ内部のメモリコントローラーを用いてデータの転送が行われる．この SDRAM はカメラからの画像データなどの大きなデータを格納するために利用される．

FE-GA のアーキテクチャを図 2 に示す．FE-GA は 32 個のアレイ状に配置された"Processer Element"(PE) から構成される．その内 24 個の PE(ALU) は加算器，減算器，マルチプレクサ，ビットシフト，論理回路，比較器として使用できるが乗算機能は持たない．残りの 8 個の PE(MLT) は乗算器，アキュ

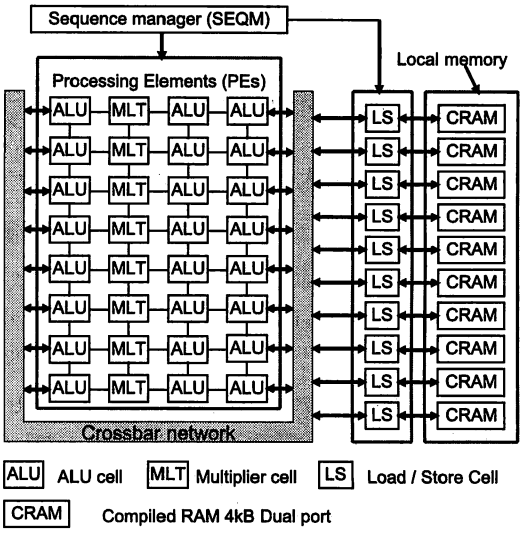
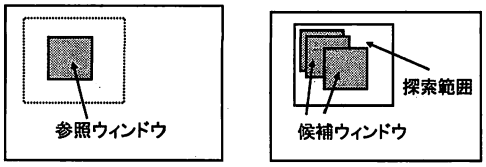


図 2　FA-GA の構成

ムレータ，カウンタなどとして用いることができる．それぞれの PE は隣接している 4 つの PE 間のみで接続が可能でありその接続も動的再構成が可能となっている．FE-GA の内部には CRAM と呼ばれるそれぞれ 4 キロバイトのデュアルポートメモリが 10 個用意されている．右端と左端の PE は cross bar network を通じて CRAM と接続されている．CRAM と PE との間には LS（Load/Store) セルと呼ばれる特別なセルが用意されており，ロード，ストアのいずれかの設定が可能である．また，LS セルはデータメモリのアドレス生成を行う．あらかじめ設定されたサイクル数を経過するとシーケンスマネージャによってシーケンスの切り替えが行われる．PE，PE 間配線，LS セル，PE-CRAM 間の配線は動的再構成が可能であることから，シーケンスの切り替えによりそれぞれ最大で 256 個の設定を持つことが可能となる．シーケンスの切り替えがあった場合にも CRAM の値は保たれる．また，FA-GA にはバンクと呼ばれるシーケンスの記憶部が 4 つ用意されている．ひとつのバンクがあるシーケンスにおいて利用されている間に残りの 3 つのバンクは書き換えが可能であり，最大 3 つのシーケンスをプリロードすることが可能である．

FE-GA にてアプリケーションの実装を行う上で大きな問題点としてデータメモリのアドレッシングがある．LS セルでは複雑なアドレス生成を行うことはできない線形アドレッシング関数が用いられている．線形アドレッシング関数のパラメータは"インクリメント"，"繰り返し回数"，"ベースアドレス"である．初めにアドレスはベースアドレスの値に設定されその後サイクルごとにインクリメントの値だけ加算される．サイクル数が繰り返し回数の値と一致した場合には，アドレスはベースアドレスの値にリセットされる．必要とされるアドレッシングが線形アドレッシング関数では実現できない場合には PE アレイを用いてアドレスの生成を行うことも可能であるが，FE-GA では 32 個の PE アレイしか用意されていないため，アドレス



(a) 時刻 t の画像　　(b) 時刻 $t+\delta t$ の画像

図 3　対応点探索

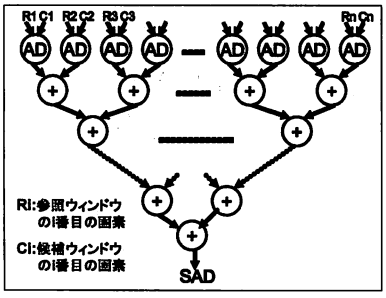
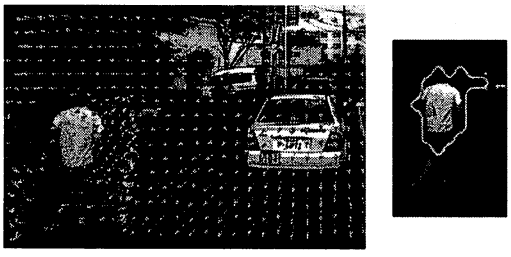


図 4　SAD 演算のデータフローグラフ



(a)　　(b)

図 5　連結されたオプティカルフローと人間抽出

生成部が多くの面積を占めることとなる．

3. オプティカルフロー

3.1 オプティカルフローのアルゴリズム

オプティカルフローは時刻 t と $t+\delta t$ における画像間における物体の動きをベクトルで表したものである．オプティカルフローでは図 3 に示すように t と $t+\delta t$ における画像内での対応点探索のために t における参照ウィンドウと $t+\delta t$ における探索範囲を設定し，参照ウィンドウと探索範囲内の候補ウィンドウにおいて SAD 演算を行う．図 4 は n 画素の参照ウィンドウと候補ウィンドウの SAD 演算の DFG を示している．特定の参照ウィンドウに対応する探索範囲内での SAD 演算の結果が最小となった点を対応点と判定する．対応点探索により得られる物体の δt の間に移動した方向，距離をベクトルとして表わす [2].

時間的に連続した画像に対するオプティカルフローベクトルを連結することにより，物体認識が行える．例として図 5(a) に示すように人間の特徴的な動きに着目することにより図 5(b) に示すように人間領域を抽出することができる．

表 1　使用した画像の仕様

画像サイズ	640 × 480
画像タイプ	RGB color image
候補ウィンドウ	16 × 16 pixels
参照ウィンドウ	16 × 16 pixels
探索範囲	24 × 24 pixels

表 2　CPU における処理時間

タスク	処理時間 (ms)	並列度
SAD 演算	584	非常に大
SAD 最小値の探索	0.02	小
動きベクトルの連結	0.5	非常に小

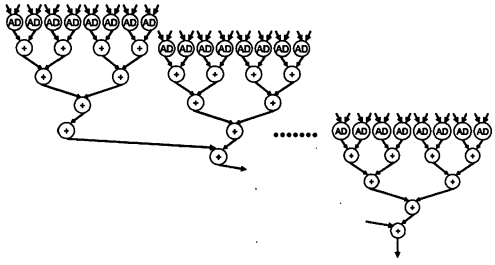


図 6　8 並列で行った SAD 演算のデータフローグラフ

3.2 アロケーション

本稿では表 1 で示す画像サイズおよびウィンドウサイズを使用してオプティカルフローを実行した．表 2 は CPU を用いてオプティカルフローの計算を行った場合におけるタスクごとの処理時間を示す．表 2 にあるように，オプティカルフローでは SAD 演算の処理時間の割合が非常に大きいとともに，並列度が非常に大きいことがわかる．一方，他のタスクについては条件分岐や並列度の低さから CPU による演算が適していると考えられる．

SAD 演算は図 4 に示すように完全な並列演算であり，16 × 16 のウィンドウサイズでは 256 の並列演算となる．しかしながら FA-GA は 32 個のセルしか存在しないため 256 個すべてを並列で処理することはできない．そこで図 6 のように 8 並列にて SAD 演算を行った．この DFG を FA-GA へ実装を行った PE アロケーションを図 7 に示す．

探索範囲には 81 個の候補ウィンドウが存在するがその際に利用される画素データは図 8 のように重複して利用される．そのため重複する画素データは他の候補ウィンドウにおける SAD 演算でも再利用することにより FA-GA 内部に存在する CRAM に転送するデータ量を減少させることになる．一方で，画素データを再利用するには複雑なアドレッシングが必要とされるが，2 章で述べたように LS セルでは線形アドレッシング関数のみを持ち，複雑なアドレッシングを行うことは困難である．そこで FA-GA のシーケンスを切り替えにより候補ウィンドウごとにベースアドレスの変更を行うことで線形アドレッシングのみで画素データの再利用を実現した．

3.3 オプティカルフローの評価

表 2 で示したようにオプティカルフローでは SAD 演算が処

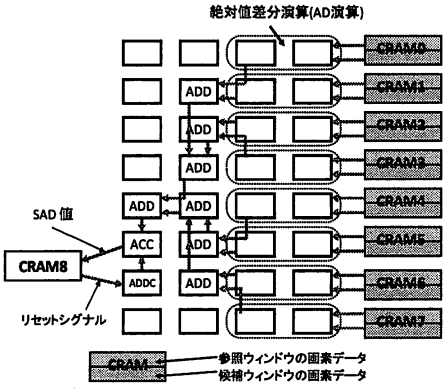


図 7 PE アロケーション

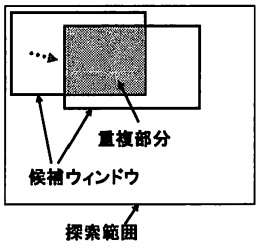


図 8 候補ウィンドウの重複

表 3 CPU × 1，FA-GA × 1 による処理時間

タスク	プロセッサー	実行時間 (ms)
SAD 演算	FE-GA	14.6
SAD の最小値探索 and CRAM からの読み出し	CPU	3.6
動きベクトルの連結	CPU	0.5
SDRAM から CRAM へのデータ転送	CPU	13.1

理時間の多くを占めている．そこで SAD 演算を FA-GA に実装を行った．表 3 は FA-GA と CPU を用いたオプティカルフローのタスクごとの処理時間を示している．CPU のみの場合での結果と比較し，SAD 演算の時間は大幅に減少している一方で SDRAM から FE-GA 内部の CRAM へデータ転送の時間と，FA-GA による SAD 演算結果を SDRAM へ書き出すためのデータ転送の時間が発生している．この結果は FA-GA は内部に存在する CRAM からのみデータのロードが可能であることに起因している．

3.2 節で述べたように SAD 演算は完全に独立した処理であり，異なる対応点に対しては並列演算が可能である．そのため 2 個の FA-GA に異なる対応点に対する処理を割り当てることが可能である．この場合にはある参照ウィンドウに対する SAD 演算が完了し，最小 SAD 値の探索，動きベクトルの連結，次の参照ウィンドウに必要となるデータの転送を行うと同時にもう一方の FA-GA によって異なる参照ウィンドウに対する SAD 演算が行われる．CPU および FA-GA をひとつずつ利用した場

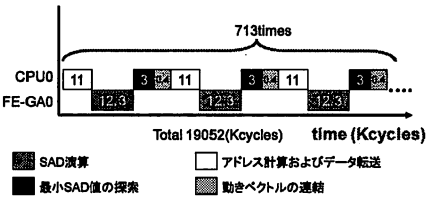


図 9 $CPU \times 1$，$FA-GA \times 1$ におけるタスクスケジューリング

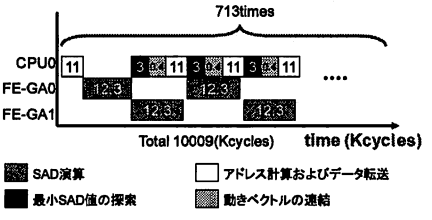


図 10 CPU ×1，FA-GA ×2 におけるタスクスケジューリング

表 4 実装ごとの処理時間

実装	処理時間 (ms)	消費電力
1 CPU (600MHz)	583	1.21
2 CPUs (600MHz)	292	1.39
1 CPU (600MHz) and 1 FE-GA (300MHz)	32	1.30
1 CPU (600MHz) and 2 FE-GAs (300MHz)	17	1.37

合と 1 個の CPU と 2 個の FA-GA を利用した場合のタスクのスケジューリングをそれぞれ図 9 および図 10 として示す．このように 2 つの FA-GA に並列演算を割り当て，CPU の稼働率を向上させることで全体の処理時間を短縮することが可能となる．

表 4 はそれぞれの実装に対する処理時間と消費電力を示している．1 個の CPU と 1 個の FA-GA の組み合わせにより，CPU 単体に比べ 18.2 倍，1 個の CPU と 2 個の FA-GA の組み合わせでは 34.3 倍の処理実現を実現できた．それに対し消費電力は CPU 単体と比較し 13%の増加であり消費電力あたりの処理速度は 30.3 倍に向上している．3.2 説で述べたように画素データの再利用を行うとデータ転送量を減少させることが可能である．データ再利用はシーケンスの切り替えによって実現しているため，利用するシーケンスの数を減少させると SAD 演算には重複した画素データが必要になるり，CRAM に転送するデータ量は増加する。逆にシーケンス数を増加させると転送データ量は低下する関係がある．また，データ転送の増加は全体の処理時間の増加の原因となる．図 11 はシーケンス数と処理時間の関係を表している．

4. Histgrams of Oriented Gradients

4.1 HOG 特徴量算出アルゴリズム

HOG は輝度勾配の強度と方向を用いた一般物体認識に利用される特徴量である [3], [4]．以下に HOG 特徴量算出のアルゴ

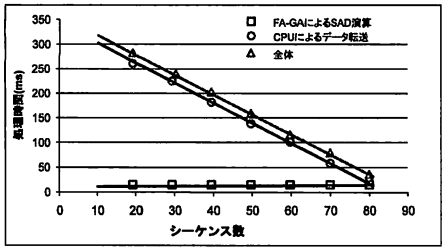


図 11　シーケンス数と処理時間

リズムを述べる．

4.1.1　輝度勾配算出

各画素の輝度より次式により注目画素 (x,y) における勾配強度 $m(x,y)$ および勾配方向 $\theta(x,y)$ を求める．ここで $I(x,y)$ は (x,y) における輝度値とする．

$$m(x,y) = \sqrt{f_x(x,y)^2 + f_y(x,y)^2} \tag{1}$$

$$\theta(x,y) = tan^{-1}\frac{f_y(x,y)}{f_x(x,y)} \tag{2}$$

$$\begin{cases} f_x(x,y) = I(x+1,y) - I(x-1,y) \\ f_y(x,y) = I(x,y+1) - I(x,y-1) \end{cases} \tag{3}$$

カラー画像の場合には RGB それぞれにおいて算出される勾配強度の中で最大となる勾配強度および対応する勾配方向をその画素の勾配強度，勾配方向とする．

4.1.2　セルにおけるヒストグラムの作成

4 × 4 画素の重複しない領域をセルと定義し，セルごとに勾配方向ヒストグラムを作成する．$\theta(x,y)$ を $[-90°, 90°]$ の範囲で 20° ずつ 9 方向に量子化し，その方向ごとに $m(x,y)$ を累積することによりヒストグラムを生成する．

4.1.3　ブロックにおける正規化

3×3 セルの領域をブロックとして定義する．ブロックはセルとは異なり，重複しながら 1 セルずつ移動する領域とする．各セルは 9 次元の特徴ベクトルを持つため，ブロックでは 81 次元の特徴ベクトルを持つことになる．あるブロックの特徴ベクトルを V，ブロック内における位置 (i,j)，$\{1 <= i <= 3, 1 <= j <= 3\}$ のセルの特徴ベクトルを f_{ij} としたときに，次式によりそのブロックの正規化を行う．

$$v = \frac{f_{ij}}{\sqrt{||V||_2^2 + \epsilon^2}} \quad (\epsilon = 1) \tag{4}$$

4.2　アロケーション

本稿では表 5 で示す仕様に基づき，HOG アルゴリズムを 2 章で述べたヘテロジニアスアーキテクチャにて実装および評価を行った．表 6 は 1 個の CPU を使用し HOG 特徴量の算出を行った場合におけるタスクごとの処理時間を示したものである．FA-GA を用いて全体の処理速度を向上させるには 3.3 節で述べたようにデータ転送を少なくすることが重要である．そのため CPU と FA-GA 間でのタスク分割の仕方が重要となる．図 12(a) で示すタスク分割にした場合を考える．この場合，輝度勾配算出は RGB ごとにすべての画素において行われるため 64 × 64 の画像では x 方向，y 方向にそれぞれ 12288 回 (=3 × 64 × 64) の差分演算が必要となり，それらの計算結果はすべて外部メモリに書き出される．一方図 12(b) では 16 × 16 のセルがそれぞれ 9 次元のベクトルを持つため，2394 回の書き出しが行われれ，データ転送量は図 12(a) のタスク分割と比較し小さくなる．

表 5　使用した画像の仕様

画像サイズ	64 × 64
画像タイプ	RGB カラー画像
セルサイズ	4 × 4 ピクセル
ブロックサイズ	3 × 3 セル

表 6　CPU における処理時間

タスク	処理時間 (μs)
輝度勾配算出 & 勾配強度の最大値算出	2461
セルにおけるヒストグラムの作成	2616
ブロックにおける正規化	1960

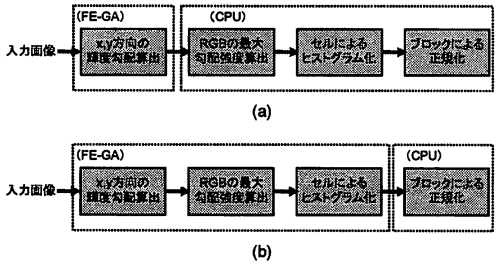


図 12　タスク割り当て

4.3　アルゴリズムの変形とタスク分割

FA-GA のセルは除算器と平方根演算器を備えていないないため，図 12(b) のタスク割り当てを FA-GA で実行するためには HOG のアルゴリズムを FA-GA の処理に適した形に変形する必要がある．そこで式 (1) は次式のように平方根を含まない形へ変形する．

$$m(x,y)^2 = f_x(x,y)^2 + f_y(x,y)^2 \tag{5}$$

RP1 では FA-GA の PE 間，PE と CRAM 間のデータ転送は 16 ビット単位で行われる．一方，画素情報は 8 ビットで表わされるため式 (5) の結果 17 ビットとなる．そのため中間出力となるセルの特徴ベクトルが 16 ビットで表せるように，式 (5) の出力は理論上の最悪値を考慮し下位 5 ビットをシフトし，12 ビットの値として扱う．

また，量子化された角度をそれぞれ $\alpha_0, \alpha_1, \ldots, \alpha_9$ とすると量子化は次式を用いて行われる．

$$tan\alpha_n \leqq tan\theta = \frac{f_y}{f_x} < tan\alpha_{n+1} \ (n = 0, 1, \ldots, 8) \tag{6}$$

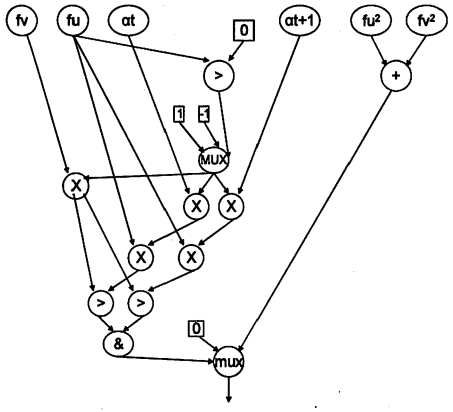


図 13　ヒストグラム生成の DFG

式 (6) の f_x が負の場合には不等号の向きが変わることを考慮し，次式のように除算を必要としないアルゴリズムへ変形を行う．

$$\begin{cases} f_x \times tan\alpha_n \leq f_y < f_x \times tan\alpha_{n+1} (f_x \geqq 0) \\ f_x \times tan\alpha_n > f_y \geqq f_x \times tan\alpha_{n+1} (f_x < 0) \end{cases} \quad (7)$$

以上のように FA-GA を用いて実行できるよう形式にアルゴリズムを変形することにより，図 12(b) で示すタスク分割の処理を実現した．

提案したアルゴリズムを用いたヒストグラム作成を行う DFG を図 13 に示す．この DFG を実装する上で問題となる点としてアドレス生成がある．各画素データである f_x, f_y, f_x^2, f_y^2 は 9 サイクルに 1 回インクリメントする必要がある．このようなアドレス生成は線形アドレッシング関数では実現できないため，4 個の PE を用いてアドレス生成を行った．

4.4 評　　価

前述のように本稿では中間結果のデータ転送時間を考慮し，図 12(b) で示すタスク割り当てで，CPU と FA-GA を用いて HOG 特徴量の算出を行った．表 7 は割り当てられたタスクごとの処理時間である．CPU 単体と比較し，消費電力の増加は 7.4%であるのに対し，処理速度はヒストグラムの生成で 5.5 倍，全体で 2.2 倍となる結果が得られた．消費電力あたりの処理速度ではそれぞれ 5.0 倍，2.1 倍の向上である．一方で FA-GA を用いた HOG 特徴量算出では CPU による演算に比べ，精度が低下するという問題がある．この原因としては FE-GA が持つビット幅制限のため，式 (5) においてビット幅の削減を行い，式 (6) では比較は 16 ビットで行っていることにある．そのため，データのビット幅制限が少ない CPU による演算に比べ，精度が低いという問題がある．精度に関しては物体認識の段階において十分評価する必要がある．

5. ま と め

本稿では 2 章で述べたヘテロジニアスアーキテクチャに対してアプリケーションを実装し，消費電力あたりの処理速度を向上できることを示した．結果より FA-GA では処理時間のうち大部分をデータ転送時間が占めていることが分かる．このため全体のパフォーマンスの向上にはデータ転送量を考慮に入れたタスク分割が有効であることを示した．大量の途中結果を外部メモリへ書き出さないようにするにはある程度多くの処理を FA-GA 内部で処理させ，出力結果を少なくする必要がある．それを実現するためには十分な内部メモリ容量などのハードウェアリソースが必要である．このため将来のアーキテクチャではメモリや PE の増加，高速で並列なデータ転送といったハードウェアの追加を低消費電力を維持しつつ行っていく必要があると考えられる．

表 7　CPU と FE-GA における処理時間

タスク	処理時間 (μs)
輝度勾配算出 勾配強度の最大値算出 セルにおけるヒストグラムの作成	936
ブロックにおける正規化	2616
全体	3185

謝辞　本研究を行うに当たってサポート頂いた株式会社日立製作所中央研究所関連各位に深く感謝致します．

文　　献

[1] Hiroaki Shikano, Masaki Ito, Masafumi Onouchi, Takashi Todaka, Takanobu Tsunoda,Takanobu Tsunoda, Tomiyuki Kodama, Kunio Uchiyama, Toshihiko Odaka, Tatsuya Kamei, Hironori Kasahara, "Heterogeneous multi-core architecture that enables 54x aac-lc stereo encoding", IEEE Journal of Solid-State Circuits, Vol.43, No.4, pp.902-910(2008)

[2] Seunghwan Lee, Masanori Hariyama, Michitaka Kameyama,"An FPGA-Oriented Motion-Stereo Processor with a Simple Interconnection Network for Parallel Memory Access", IEICE Trans. INF. Syst., Vol.E83-D, No.12,pp. 2122-2130(2000)

[3] Navneet Dalal and Bill Triggs .Histgrams of Oriented Gradients for Human Detection

[4] 藤吉　弘亘, "Gradient ベースの特徴抽出", 情報処理学会研究報告, 2007-CVIM-160, pp.211-224(2007)