愛知工業大学研究報告
第 44 号　平成 21 年

ＦＰＧＡによる画像処理演算の研究
（カメラ画像の入力及びパイプライン演算による高速化）

The Research of Image Processing System with FPGA
(Input of Camera Image and Speed-up by Pipeline Operation)

天野　国廣†，堀田　厚生††
Kunihiro AMANO, Atsuo HOTTA

Abstract　An image processing system with FPGA has been designed and implemented. It adopted the following design techniques. 1) Optimization of the module partition, 2) Improvement of memory access sequences, 3) Pipelining is introduced to the computing unit. The system was applied to an image processing implementing background subtraction. It showed the processing speed improvement by 6.5 over the system of the last design that was developed in 2006.

1. はじめに

1･1　研究の背景

近年、LSI に関する技術が急成長を遂げる中、その応用範囲は拡がり続けている。例えば、カメラ画像などを取り扱う事例がある。人の命の保護や安全の確保のための装置や、人の代わりとなって工場などのラインを自動化するシステムなどの目的に導入されている。とりわけ自動車業界では人の目や耳の代わりとなって危険を察知するために車載用カメラを利用しドライバのアシストを行うものや[1]、交通の状態を把握しその情報を活用しようというものもある[2]。これらの処理は一般的にCPU(Central Processing Unit)を用いる場合が多い。

コンピュータ画像処理の研究が盛んになった背景にはコンピュータの高性能化が挙げられる。主要な CPU メーカである Intel 社や AMD 社による高クロック化やマルチコアによる並列化を行い、性能を向上している。開発環境にコンピューター一つあれば十分であることから、C、C++などの高級言語で組まれたライブラリが多数出現した。しかし、実用的な利用、すなわち組み込み用途などではライブラリを用いるだけでは動作速度や消費電力、実装スペース的な点で不十分である。開発環境であるコンピュータは汎用的な装置であり、直接機器に組み込むことはサイズ及びコストの点で難しい。処理能力の割に消費電力が大きくまた、多くのスペースを要してしまう。このような背景から、汎用的な CPU に捉われず、ハードウェアレベルで問題を解決する必要がある。

1･2　研究の目的

本研究の目的は、現状の画像処理システムを見直し、処理の高速化及びコントローラのライブラリ化を行うことである。本研究における高速化とは、変化の激しいデバイスの条件を従来の研究と同一としたときの、処理速度の向上である。処理の高速化、様々な構成に柔軟に対応するためには次の項目が挙げられる。

1)　モジュール分割の最適化
2)　メモリコントローラの転送効率の改善
3)　演算器へのパイプライン処理の導入

また、カメラ画像の入力は、規格の画像サイズに切り出しをあらかじめ行えるように設計する。こうすることで、余分なメモリアクセスを削減できる。

2. 画像処理システムの構成

本研究における、ベースシステムはカメラから入力し、FPGA 上で処理を行い、汎用コンピュータに出力するものである。システム全体の構成を図 1 に示す。

†　愛知工業大学大学院 工学研究科（豊田市）
††　愛知工業大学　工学部　電気学科（豊田市）

愛知工業大学研究報告,第 44 号,　平成 21 年,Vol.44,　Mar,2009

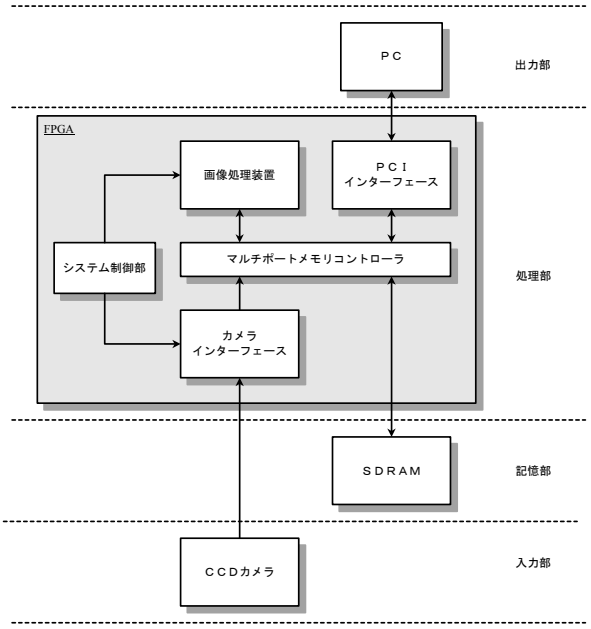

図 1　システム全体の構成

2･1 システムの構成

画像処理システムの構成は、大きく分けて入力部、記憶部、処理部、出力部に分けられる。入力部はカメラから送られてくる画像のデータを取得する。記憶部はカメラから取得した画像データを記憶、画像の処理後の結果や一時データの保存を行う役割を果たす。処理部は主な画像処理を行う部分である。出力部は処理結果を PC などに出力する役割を果たす。どの部分に対しても言えることはバッファの扱い方次第で処理能力に大きな影響を与える。カメラ画像の入力部を考えてみる。画像の処理とカメラ画像の取得を逐次実行すると、処理中の部分の入力画像がコマ落ちしてしまう。すなわち処理能力が低下しており、このため、バッファを用いる必要がある。

2･2 機能分割及び階層化

近年の傾向として LSI 上に様々なモジュールを載せて、ワンチップで多くの機能を実現する SOC(System On Chip)がよく見られるようになった。複雑な多くの機能を実現しようとすると、開発に長い期間を要してしまう。そこで、LSI の回路情報を記述したハードウェアを IP(Intellectual Property)として蓄積しておき、必要な時に取り出して進行中のプロジェクトに適用していく。こうすることで新規に追加するモジュールを削減することができ、開発期間を短縮できる。

IP を利用しやすいものにするためには、モジュール化する際に適切な機能分割が必要となる。本研究では次のようにモジュール分割した。

1)　カメラインターフェース
2)　メモリコントローラ
3)　PCI バスインターフェース
4)　ブリッジモジュール
5)　画像処理装置

また、本方式における下位モジュールの、システム構成を図 2 に、従来方式によるものを図 3 に示す

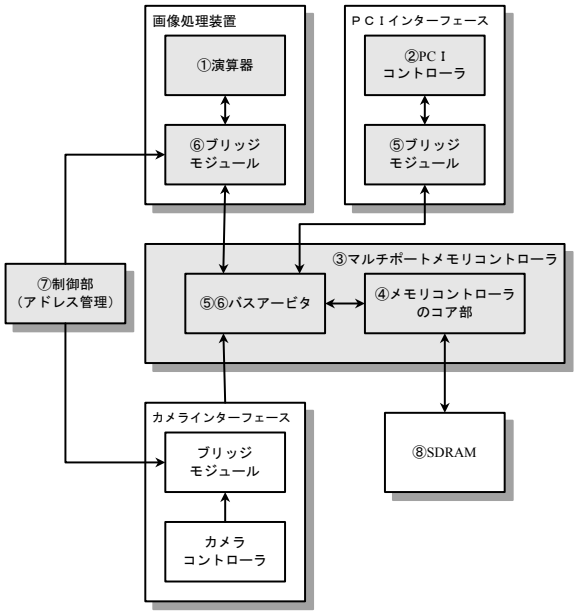

図 2　本方式のシステム構成

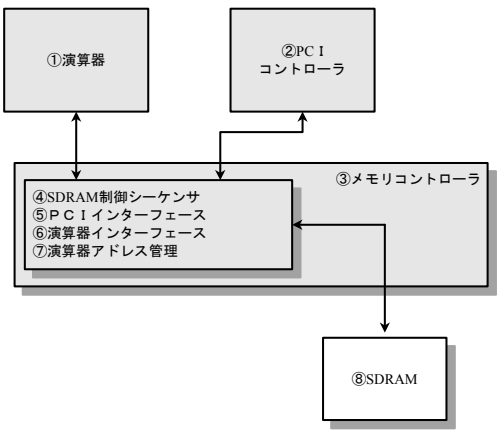

図 3　従来方式のシステム構成

従来型のシステムはメモリコントローラの中に複数のモジュールとのインターフェースを持つ。例えば、半導体技術の進歩により SDRAM に新しい規格が登場し、メモリコントローラのコア部、すなわち物理層に近い部分に設計変更が必要になったとする。この場合、メモリコントローラ全体を見直し、処理の変更を行わなければな

らず、設計が困難である。一方、本研究で採用したシステム構成であればメモリコントローラのコア部は独立しており、新デバイスを導入する際、余計な時間的コストを要しない。

2･3 メモリコントローラ

画像情報を取り扱う場合はFPGA内部のSRAMブロックだけでは容量の面で不十分である。本研究では画像格納用メモリとしてSDRAMを用いる。SDRAMとはキャパシタ内の電荷の有無をビットデータとした、4～8バンク構成のクロックに同期して動作するメモリである。メモリにアクセスするための決められた手順が存在し、その手順でメモリにアクセスできるモジュールをメモリコントローラという。SDRAMのバンク構成を図4に示す。

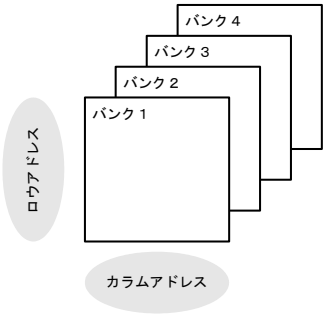

図4　一般的なSDRAMのバンク構成

SDRAMは一般的に複数のバンクから構成され、ロウアドレスとカラムアドレスという縦横のアドレス線を持ちI/Oピンはマルチプレクスされている。SDRAMにアクセスするためにはまずバンクをアクティブ状態にする必要がある。次にバンクに対して読み書き動作を行う。バンクへのアクセスは終了したら、バンクを閉じる動作（プリチャージ）を行う。これは読み書きしたアドレスのメモリビットに対して電荷を補充することで揮発を防ぐ意味がある。SDRAMへのアクセスの流れを図5に示す。

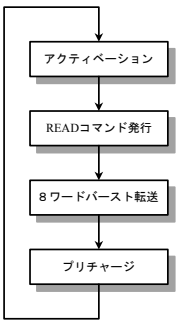

図5　アクセス手順（従来方式）

この場合の転送速度T_R (Translate Rate)は次の式で求められる。

$$T_R = \left[\frac{W_B \times L_B}{tAC \times 8} \right] \Big/ 1024^2 \ [MByte / \sec] \quad (1)$$

W_B (Bus Width) : バス幅
L_B (Burst Length) : バースト長
tAC (Access Time) : 1アクセス当たりの所要時間

しかし、アクセスするたびに、バンクを閉じていたのでは効率が悪い。そこでアクセスするたびに、バンクアドレスとロウアドレスを保持しておき、次にアクセスするときにそれを参照する。同じバンクアドレスとロウアドレスであれば再びアクティベーションを行う必要はない。言い換えればカラムアドレスをいくら変化させようともプリチャージを行わなくてもよい。但し、異なるアドレスの場合はプリチャージとアクティベーションを行う必要がある。その場合のアクセス手順を図6に示す。

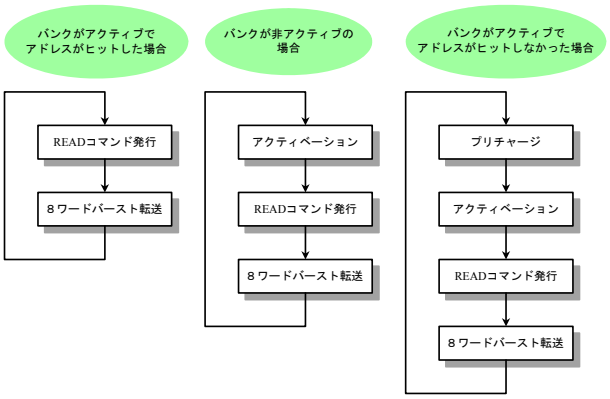

図6　状態に応じて制御を分岐（本方式）

この場合の転送速度$T_R{}'$は次の式で求められる。

$$T_R{}' = \left[\frac{W_B \times L_B}{tAC_{hit} \times 7 + tAC_{nohit}} \right] \Big/ 1024^2 \ [Mbyte / \sec] \quad (2)$$

ここでtAC_{hit}はアドレスヒットしたときのアクセス時間。tAC_{nohit}はアドレスヒットしていないときのアクセス時間である。

画像処理などの場合は連続したアドレスにアクセスする可能性が高い。連続した領域にアクセスする場合、異なるアドレスにアクセスする頻度はカラムアドレスのアドレス長で決まる。カラムアドレス長が８ビットであれば８ワードバースト転送で７回のアクセスに対して１度のみプリチャージ、アクティベーションを行う。電源の投入は１回、リフレッシュ間隔は十分長い期間であるため無視するとアクセス速度は式(2)となる。

従来の研究ではメモリコントローラがバースト転送を行っておらず、メモリバスの帯域を十分に利用できてい

愛知工業大学研究報告,第 44 号,　平成 21 年,Vol.44,　Mar,2009

なかった。本研究では、バースト転送への対応と、アクセス手順の見直しをおこなった。シミュレーションによる転送速度の理論値を図 7 に示す。

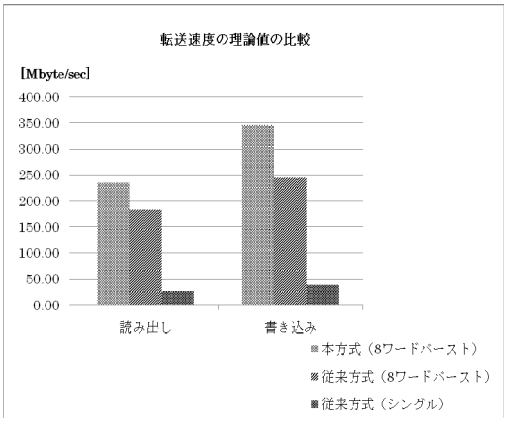

図 7　シミュレーションによる転送速度の理論値

本方式のメモリコントローラは従来方式のものと比べて、読み出しで約 9.2 倍、書き込みで約 9.0 倍の転送速度を得ることができる。アクセス手順は従来方式のまま、バースト転送に対応させた場合と比べても、読み出しで 1.3 倍、書き込みで 1.4 倍の転送速度が得られる。よって、本方式のアクセス手順を導入することで、バースト転送に対応しただけの方式と比べてもパフォーマンスが向上すると言える。

3. 画像処理モジュールの実装

構築したプラットフォームに画像処理モジュールを実装する。システムの中で最も稼働率が高い部分がメモリコントローラであり、この影響により処理能力は制限を受ける。そのため各リソースに遊びを持たせないように工夫する必要がある。そこでメモリと演算器の間に FIFO を設け、データが FIFO に残っている間は他のリソースからのメモリアクセスを受け付けることが可能となる。

3･1 背景差分法

背景差分法はあらかじめ背景画像を取得して、その後に得られる画像との差の絶対値を取り設定した閾値を基に 2 値データを得る方法であり、カメラの視野に侵入した物体を検出することができる。この方法で用いる背景画像は定点カメラから得られる画像である必要があり、カメラ自体にぶれが生じる場合は差分値が大きくなって検出の原因となる。

今、対象とする入力画像の輝度値を $I_a(i,j)$、背景画像の輝度値を $I_b(i,j)$ として差分画像の輝度値 $I(i,j)$ を求める。

$$I(i,j)=\left|I_a(i,j)-I_b(i,j)\right| \qquad (3)$$

次に差分値を 2 値情報に変換する

$$B(i,j)\begin{cases}1: I(i,j)\geq I_{th}\\ 0: I(i,j)<I_{th}\end{cases} \qquad (4)$$

ここで I_{th} は画素値の閾値である。

3･2 パイプライン演算

演算の高速化を図るために演算リソースを無駄なく活用する必要がある。そこで各処理を時間的にオーバーラップさせメモリアクセス時間を削減できるパイプライン演算を採用する。画像を対象にしたパイプライン構成でのタイムチャートを図 8 に、パイプライン処理を行わない場合のタイムチャートを図 9 に示す。

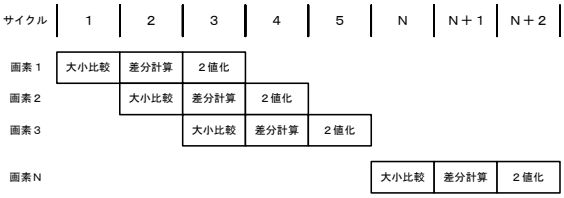

図 8　パイプライン化された処理のタイムチャート

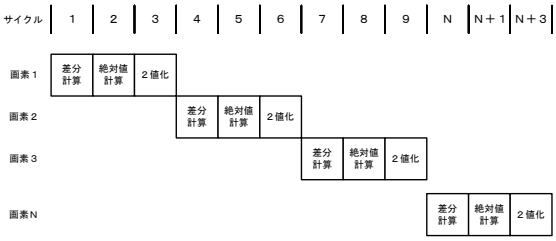

図 9　一般的なソフトウェア処理のタイムチャート

また、パイプライン化を行わない場合は図 9 のタイムチャートであり、時間的な無駄が大きいことがわかる。

$$T_p=\sum_i\sum_j(T_1+T_2+T_3) \qquad (5)$$

$$T_p'=\sum\sum T_{\max}+T_1+T_2+T_3 \qquad (6)$$

ここで、T_1,T_2,T_3 は各処理の処理時間であり、$T_{\max}$ は T_1,T_2,T_3 の最大値である。

パイプライン動作を可能とするため、演算器の入力と出力部に FIFO(First In First Out)型メモリ付加したこ

れにより、連続的にデータを転送できる。

4. 実機動作

4･1 動作テスト環境

設計したシステムにおける処理速度を測定するためのテスト環境を作成した。テスト環境を図 10 に示す。

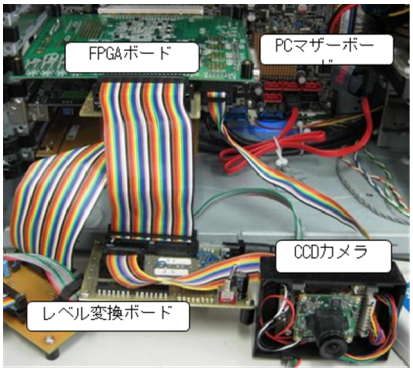

図 10　テスト環境

カメラから得られた画像をFPGA内部のSDRAMに複数枚格納する。次に、PCのプログラムにより画像の背景差分を行うようシステムに命令を送る。命令を受けると同時に内部に設置した計測用カウンタがスタートされる。処理が終了したらSDRAMに結果を書きこむと同時に、計測用カウンタをストップさせ計測を終了する。PCのプログラムからFPGA内部に命令を出し、SDRAMに記憶されている処理後の画像を得る。尚、PCI コントローラがバースト転送に非対応であるため転送速度は非常に低い。そのためSDRAMに画像を複数枚ストックさせるといった手段をとる。

4･2 結果

背景画像として図 11 を与えた時、入力物体画像図 12 との差分をとると図 13 の画像が得られる。更に差分画像を２値化した画像を図 14 に示す。

図 11　背景画像

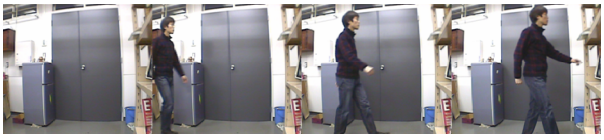

図 12　入力された物体画像

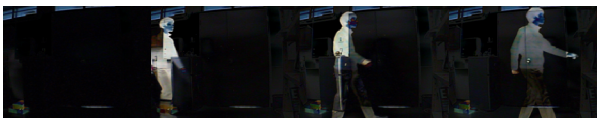

図 13　背景差分された画像

図 14　２値化後の画像

図 11 の背景画像と図 12 の入力物体画像を比べると背景に変化はないように見える。しかし、図 14 に示す２値画像は画面上方の窓が物体でないにも関わらず浮き出て見えている。これではラベリング処理を施した際に上方の窓まで物体と認識されてしまう。原因としては、照明に蛍光灯を使用しているためカメラのスキャンレートと干渉してしまうためであると考えられる。尚、現状システムの構成からランダムアクセスに時間がかかるため、ラベリング処理は実装していない。

次に、本研究、過去の研究の処理時間の比較を行う。これは、画像の背景差分、２値化を画像１枚分行った場合である。参考として、CPU による処理の例として Intel Core2 Duo E8400 3.00GHz により処理時間を調べた。画像処理ライブラリとして OpenCV1.0 を使用した。

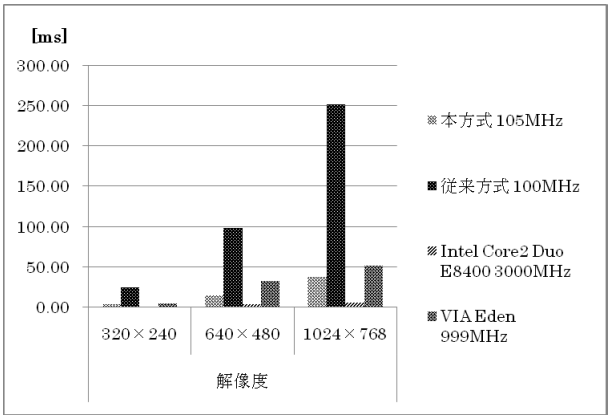

図 15　実機動作時の処理時間の比較

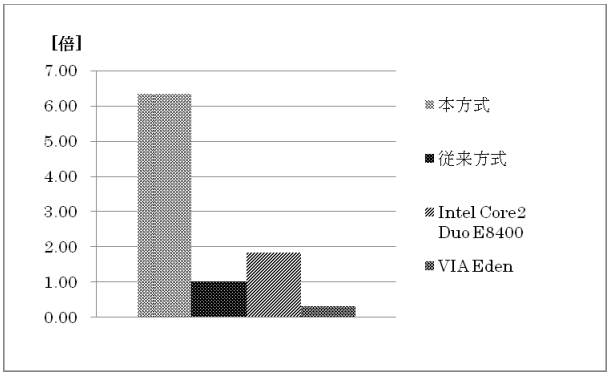

図 16　周波数で正規化した処理能力

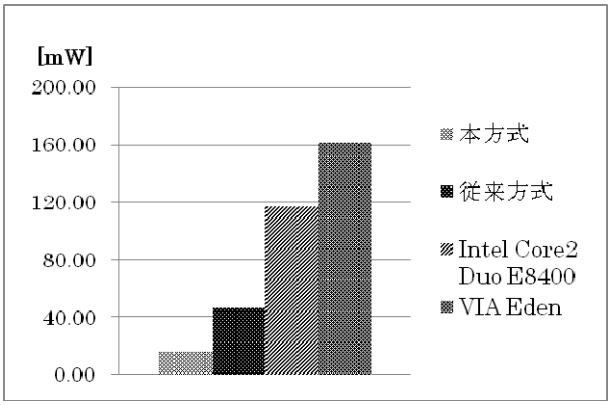

図 17 画像 1 枚を処理するために必要な電力

本研究で設計した画像処理システムの処理時間は従来のシステムの処理時間の約 1/6.5 程度に短縮できている。この要因としては、メモリコントローラの転送速度を従来方式の約 9 倍に、改善したことが挙げられる。尚、本研究ではメモリコントローラの転送をバーストとしたため、単純計算では 1/8 程度の処理時間となるはずであるが、これは各種インターフェースの抽象化、モジュール分割によるレイテンシの増加の影響が考えられる。

汎用 CPU で実験を行った結果の処理時間は極端に短くなった。これは動作周波数が 3GHz であり FPGA ボードの動作周波数 105MHz の約 28.6 倍であるため単純比較はできない。またデバイスのプロセスも異なる。

評価方法として VGA サイズの画像を処理するときの処理能力、画像 1 枚当たりの消費電力を追加した。処理能力(Performance)を式(7)に、画像 1 枚当たりの消費電力(Pd)の求め方を式(8)に示す。

$$Performance = \frac{T_{legacy} \times F_{legacy}}{T \times F \times L} \quad (7)$$

$$P_d = \int_0^T Pdt = P_{\max} \times L \times T \quad (8)$$

T_{legacy} ：従来法の処理時間
F_{legacy} ：従来法の動作周波数
T ：対象の処理時間
F ：対象の動作周波数
L ：システムの負荷率
P ：実行消費電力
$P_{\max}$ ：最大付加時の消費電力(TDP)

ここで、ハードウェアは常に動作しているものと仮定するため、負荷率は 100%であるが、CPU はシステムから負荷率を取得するために、専用の API を用いる。

本研究のシステムにおける処理速度に関しては、E8400 の CPU に負けるが、消費電力に関しては本研究のシステムが有利である。

この結果は、本方式における画像処理を 1 種類に限定していることによるものである。更に多彩な画像処理を行わせることができるよう本システムを改良したならば、この結果は変わる可能性がある。すなわち、低周波数動作でメモリアクセスに無駄のない本システムは、背景差分処理を行う上で最適化されていると言えるためである。

5. 結言

本研究ではソフトウェアで画像処理を行う場合の問題点を挙げ、以下の点について改善を行った。

1)モジュール分割の最適化。
モジュールの交換や機能拡張を簡単化。

2)メモリコントローラの転送効率の改善。
従来方式に比べて、約 9 倍の高速化を実現。

3)演算器やパイプライン処理の導入。
画像処理を高速化。各処理が時間的にオーバーラップできるからである。

これらの改善点を含めて、処理時間を計測した結果、本研究におけるシステムの処理時間は、従来のものと比べて約 1/6.5 に短縮できた。また、市販されている CPU との比較も行った。画像 1 枚当たりの消費電力を比べると、本方式が有利であることが示された。

参考文献

[1] 田岡 武司, 真鍋 真, 上林 学, 大西 陽介, 福井 正博 “自動車用白線認識アルゴリズムの一実現” 情報処理学会研究報告 No.2006-SLDM-126, 12, 2006 年

[2] 岡田大輝, 和田俊和 “動的色境界の提案と道路標識追跡・認識への応用” 画像の認識・理解シンポジウム論文集 2008, No.11 pp.283-288, 2008 年

[3] 佐久間 湖, 堀田 厚生 “動画像からの移動物体抽出と速度の推定” 愛知工業大学研究報告B, Vol.40, pp. 53-62, 2005 年

[4] 山部 選, 堀田 厚生 “FPGA による画像処理演算器の設計” 愛知工業大学研究報告B, No.42, pp. 27～32, 2007 年

[5] 谷 誠一, 伊谷 裕介, 渡辺 裕, 富永 英義 “照明変化のある環境下での移動物体検出の検討” 電子情報通信学会総合大会講演論文集 Vol.2007, No.2, pp. 102, 2007 年

（受理 平成 21 年 3 月 19 日）