2H-05

FPGA 上でのクアッドパイプラインを用いた BLOB 検出

真崎　世文†　孟　林‡　山崎　勝弘‡

立命館大学大学院　理工学研究科†　立命館大学　理工学部‡

1. はじめに

Binary Large OBject(BLOB)検出は、車載カメラ、監視システムなどの交通システムや、動画像内物体の追跡など、様々な分野で用いられており、画像中の対象の認識速度向上が重要である。本研究では、Field Gate Programmable Gate Array(FPGA)上でクアッドパイプラインシステムを用いて、BLOB 検出の高速化を図る。ラベリングは画素間のデータの依存性があるので、ラベル補正を行わない簡略化ラベリングを用いて並列化する。本稿では、BLOB 検出の流れ、クアッドパイプラインの構成、BLOB 検出の処理例、及び実験結果について述べる。

2. クアッドパイプラインシステムを用いた BLOB 検出

2.1 BLOB 検出の流れ

原画像を上下に分けて格納した Block RAM(BRAM)から、2 ポートで上下に分けて出力し、合計で 4 分割した画像を並列に処理を進めていく。

処理内容は、まず処理方向に沿ってガウシアンフィルタ処理を行った後、画像を 2 値化する。次に、ラベル補正は行わず、仮ラベルと LUT を生成して画素の連結成分と仮ラベル数と座標を格納するまでの簡略化ラベリングを行う[1]。最後に、分割画像ごとの LUT を合成と更新し、ラベルの個数と座標を元に、BLOB 検出を行う。

2.2 クアッドパイプラインシステムの構成

図 1 に処理対象となる 4 分割画像を示す。FPGA 内の BRAM には 100×100 ピクセルの画像を格納しており、1 パイプラインにつき、0 から 24 まで 25 ラインの処理を行う。

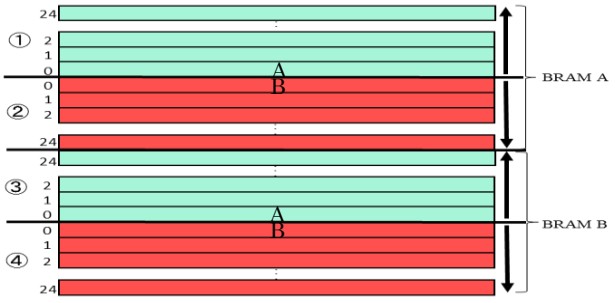


図 1　4 分割画像

図 2 にクアッドパイプラインを用いた BLOB 検出システムを示す。本システムでは、5 つの画像処理モジュール(FE,GAU,LAB,SYN,BLOB)、4 つの制御モジュール(FE_Con, GAU_Con,LAB_Con,BLOB_Con)、アドレス生成モジュール、3 つのパイプラインレジスタ、及び原画像と処理結果を格納する 3 つの BRAM から構成される。

FE は対象の BRAM から 1 ピクセルずつの画素値データを読み込む。GAU はガウシアンフィルタによるノイズ処理と 2 値化を行う。LAB は仮ラベルと LUT を生成する。SYN は分割画像ごとの LUT を合成、更新し、1 つの LUT にまとめる。BLOB は LUT 内の仮ラベル連結成分を検出し、BLOB の個数と画像中の BLOB の面積、重心を算出する。

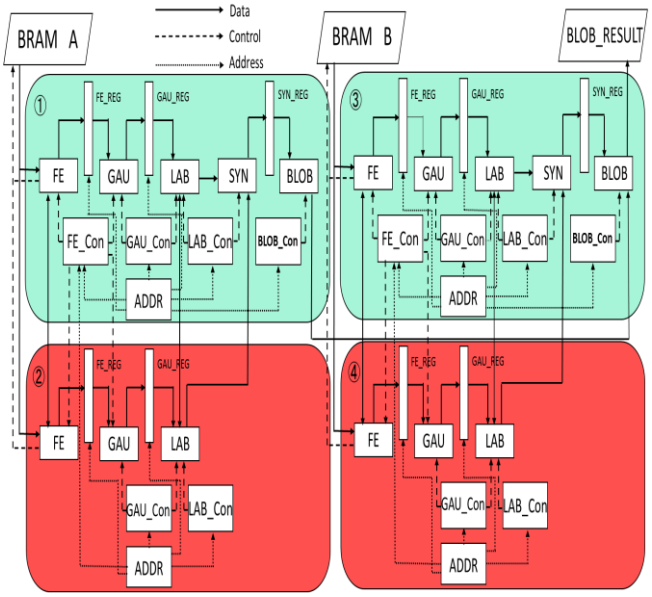


図 2　クアッドパイプラインを用いた BLOB 検出システム

2.3 分割画像境界における処理

4 つに分割した画像ごとの連結性を保つために、ラベリング時に分割画像ごとの処理を遅延させ、遅延させた動作がその前に処理を行った画像処理箇所の読み込みを行う。

図 3 にクアッドパイプラインシステムにおけるタイミングチャートを示す。図 3 の上部が、図 1 の①と③、下部が②と④に対応している。GAU までの処理を同タイミングで並列に行い、次の LAB では、まず図 1 中の①と③最下段のライン A の処理を開始し、ライン A の処理が終わり次第、その処理結果を参照して、②と④最上段のライン B のラベリングを開始する。これにより①と②、③と④の結合性が保たれる。また、③のラベリングが終わった後、1 ライン遅延した②のラベリングが③最上段の結果を参照するので、これで②と③の結合性が保持され、全ての分割画像の結合性が保持される。

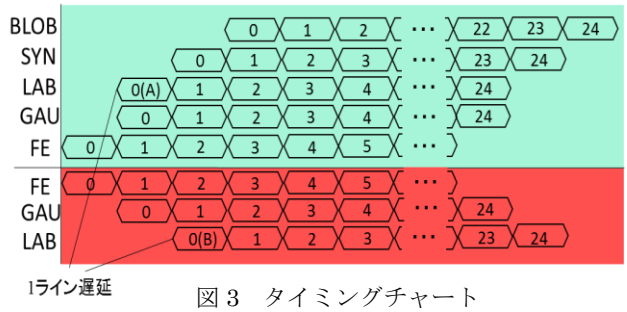


図 3　タイミングチャート

FPGA-based BLOB Detection Using Quad-pipelining
Sebun Masaki†, Lin Meng‡, Katsuhiro Yamazaki‡
†Graduate School of Science and Engineering, Ritsumeikan University.
‡College of Science and Engineering, Ritsuemikan University.

3. BLOB 検出

3.1 LUT 合成と更新

クアッドパイプラインシステムによる LAB では、それぞれのパイプラインに分割した画像ごとの LUT を作成する。これを LUT 合成と更新により 1 つにまとめ、原画像としての LUT を作成し、BLOB 検出に用いる。

合成は LUT 内の連結成分を参照し、他の LUT 内に同値の仮ラベルが存在した場合は、その仮ラベル値の連結成分に元連結成分を代入することを繰り返す。また、同値の仮ラベルが存在しない場合は、他の LUT にその仮ラベルと連結成分を追加する。更新は LUT を合成し終わった際、不要な仮ラベルと連結成分を削除し、内容を更新する。

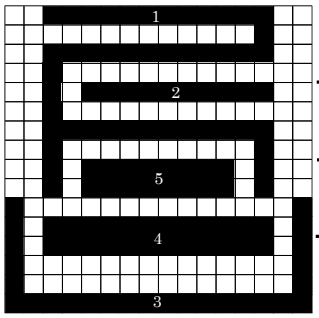
3.2 BLOB 検出処理内容

LUT 合成と更新を行った後に行う BLOB 検出では、各画素のグループである BLOB の数とそれらの面積、重心を検出し、その結果を図 2 の BLOB_RESULT に保存する。面積は画素数、重心は式(1)を用いて算出する。

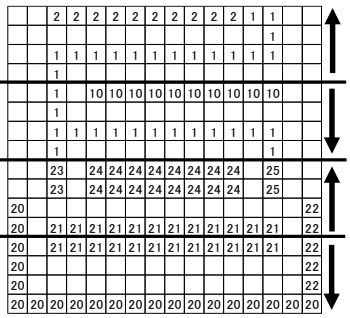
$$\begin{cases} Gx = X\text{ 座標の合計/面積} \\ Gy = Y\text{ 座標の合計/面積} \end{cases} \quad \cdots\cdots\cdots\cdots \quad (1)$$

3.3 BLOB 検出処理例

図 4 は、16×16 ピクセルの 2 値画像をクアッドパイプラインシステムにより、BLOB 検出を行った結果例である。原画像(a)に簡略化ラベリングを行い、仮ラベル結果(b)を作成し、その連結成分をそれぞれの LUT に保存する。保存した LUT を合成と更新した更新 LUT(c)を元に BLOB 結果(d)を出力する。



(a)16×16 原画像



(b)ラベリング結果

仮ラベル値	連結成分
1	2,23,25
10	
20	22
21	
24	

(c)更新 LUT

BLOB	連結成分	面積	重心
1	1,2,23,25	46	(8,3)
2	10	10	(8,4)
3	20,22	26	(7,13)
4	21	24	(7,11)
5	24	16	(7,9)

(d)BLOB 検出結果

図 4 BLOB 検出処理例

4. FPGA 上での実験

4.1 実験条件

対象画像サイズは 100×100 であり、FPGA は Virtex5 (XC5VFX-70T)評価ボード、デザインツールは ISE14.6、シミュレーションツールは Isim、ハードウェア言語は VerIiog2001 を使用している。

4.2 ハードウェアサイズ

表 1 にハードウェア使用量を示す。デュアルパイプラインと比較して、3 倍から 4 倍のハードウェア量を要するが、クアッドパイプラインシステムは使用した Viretx5 に実装できる範囲で設計を行うことができた。ハードウェア使用量の増加については、デュアルパイプラインシステムの使用量を倍にした上で、保存するデータ量と計算量の多い LUT 合成と更新部を新たに作成したためであると考えられる。BRAM の使用率については変わらず、原画像と BLOB 検出結果の格納以外には新たに使用していないからである。

表 1 ハードウェア使用量

	クアッドパイプライン	デュアルパイプライン	シングルパイプライン	Virtex5
Registers	2846(6.4%)	660(1.5%)	453(1.0%)	44800
LUTs	10053(22.4%)	3758(8.4%)	1945(4.3%)	44800
LUT-FF pairs	2113(99.3%)	518(24.4%)	271(12.7%)	2127
BRAM	138(93.2%)	138(93.2%)	132(89.2%)	148

4.3 動作周波数と実行時間

表 2 に動作周波数と実行時間を示す。シングルパイプラインとデュアルパイプラインと比較して、それぞれ 3.55 倍、1.94 倍の速度向上率を達成している。この結果からクアッドパイプラインシステムが高速化に有効であることを確認することができた。また、シングル・デュアルパイプラインと比較して、動作周波数が約 7%低下している。理由としては、LUT 合成と更新部で、一部の動作を 1 クロック内で逐次処理で行うため、その分動作に時間がかかるためであると考えられる。

表 2 動作周波数と実行時間

	クアッドパイプライン	デュアルパイプライン	シングルパイプライン	Kiran[2]
動作周波数(MHz)	92.05	99.18		100
実行時間/ピクセル(ns)	10.9	10.08		
全実行時間(μs)	31.6	57.7	112.13	220
速度向上率(倍)	3.55	1.94	1	

5. おわりに

本研究では、FPGA 上でクアッドパイプラインを用いて BLOB 検出を高速に行う手法を提案した。100×100 ピクセルの画像において、シングルパイプラインによる BLOB 検出と比較して 3.55 倍の速度向上を達成できた。今後の課題として、パイプライン各ステージの負荷バランスの解析、本手法の前方車両検出への応用などが挙げられる。

参考文献

[1] 野尻 直人, 孟 林, 山崎 勝弘, “FPGA 上でのデュアルパイプラインを用いた BLOB 検出と前方車両検出への応用”, FIT2015 RC-002, 2015.

[2] D.Kiran, A.I.Rasheed and H.Ramasangu, “FPGA Implemetation of BLOB Detection Algorithm for Object Detection in Visual Navigation”, the 2013 Int. Conf. on Circuits, Control and Communication (CCUBE), pp. 1-5, 2013.