C-012

FPGA を用いた液晶用ガラス欠損検出システムの高速化

A High-Speed FPGA Based Defect Detection System for Flat-Panel Displays

松山　圭輔 †　孟　林 ‡　野尻　直人 †　泉　知論 ‡　山崎　勝弘 ‡
Keisuke Matsuyama　Lin Meng　Naoto Nojiri　Tomonori Izumi　Katsuhiro Yamazaki

1．まえがき

液晶はディジタル情報機器のディスプレイを中心に様々な分野で使用されている．液晶用ガラス(Flat-Panel Displays:FPD)の需要が高まっており，それらの欠損の検査をリアルタイムに行うことが求められている．本研究では，液晶用ガラスの欠損の一つであるアワを検出するために，ラプラシアンフィルタ，2 値化，及びラベリングの画像処理を FPGA 上で実装し，処理を高速化することを目標とする．

現在までに，いくつかの液晶用ガラスの欠損検出方法が提案されている．[1]はマルチレベルの B-スプライン近似を用いて適応的な曲線割り当て法を提案した．[2]はマシンビジョンを用いて，自動的に曲線型の欠損を検出する手法を提案した．これらの手法は，複雑な欠損を検出することができるが，ソフトウェアによる実現だけを検討している．

また，ラベリングは画像処理の基礎的で重要な処理であるので，FPGA 上での高速化が研究されている．[3]は k-Concave な二値画像に対する低レイテンシラベリング方法を実現した．[4]は Handel C を用いて，RC1000 ボード上で，ラベリングの実装を実現した．

我々は FPD の欠損をリアルタイムに検出するために，ラベリングだけではなく，ラプラシアンと 2 値化を FPGA 上で実現している．この際，画像データを一時保存するメモリ領域が必要である．我々はすでに画像 1 枚分のフレームバッファを用いた高速化の実験を行った[5]が，本研究では，3 ライン分のみのデータを保持するラインバッファを用いる方法で新たに実装し，両者を比較する．これらのバッファは共にレジスタであり，FPGA 内の LUT を用いて実現する．

本論文では，液晶用ガラスの欠損検出方法を説明した上で，FPGA 上での画像処理モジュールの機能と一時データを保存するバッファについて述べる．次に，Virtex5 を用いた実験結果を示し，二つのバッファの構成を回路規模と実行速度の観点から比較する．さらに，パイプライン処理による高速化について検討する．

2．液晶用ガラスの欠損検出方法

液晶用ガラスの欠損検出は，あらかじめ撮影した液晶用ガラスの画像を用いて，画像処理を行う．本論文で設計した液晶用ガラスの欠損検出画像処理の流れを図 1 に示す．

実際に使用した液晶用ガラスの欠損画像の 1 つを図 2 に示す．液晶用ガラスの撮影画像のサイズは 256×256，1 画素 8bit の 256 階調データである．画像の画面中央左にある点が欠損で，欠損検出エリアはその時点で撮影した領域である．

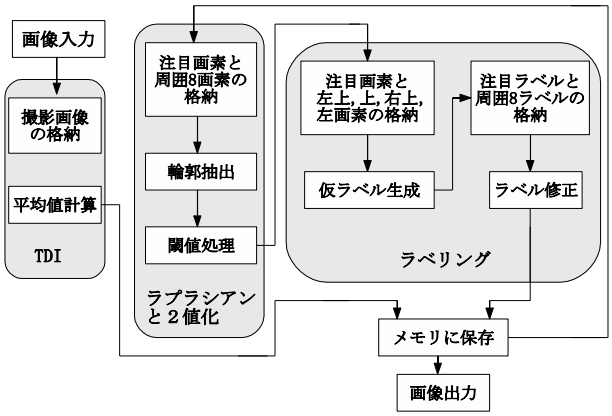


図 1　ガラス欠損検出処理の流れ

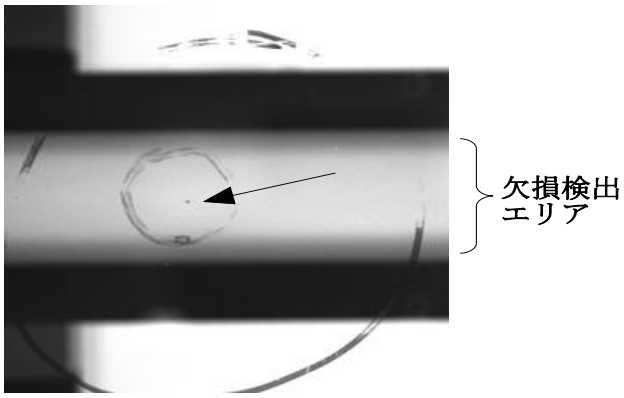


図 2　液晶用ガラスの欠損画像

画像処理には TDI，ラプラシアンフィルタ，2 値化，ラベリングの 4 つのアルゴリズムを使用する．

TDI では，対象物を縦方向に 1 ラインずつずらして 128 回撮影し，撮影した画像の共通部分を重ね合わせて平均値を取ることで，一枚の画像となる．それによりノイズの影響を減少させる．

ラプラシアンフィルタでは，特定の微分オペレータを使用して画素値を二次微分処理し，画像中に含まれる対象物のエッジ検出を行う．

2 値化では，濃淡のある画像をあらかじめ設定した閾値を用いて白と黒に分ける．

ラベリングでは，画像中の連結成分に同じラベルをつけ，異なった連結成分には異なったラベルをつける．

†立命館大学 大学院 理工学研究科, Graduate School of Science and Engineering, Ritsumeikan University
‡立命館大学 理工学部, College of Science and Engineering, Ritsuemikan University

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

3. FPGA 上での欠損検出システムの設計と実現

3.1 欠損検出システムの構成

欠損検出システムは，まず，対象とした液晶用ガラスを縦方向に 1 ラインずつずらして 128 回撮影し，撮影した 128 枚の画像の共通部分の平均値を取ることで，ソフトウェアにより TDI 処理を行い，一枚の画像となる．次に TDI 処理後のデータを FPGA チップ内の Block RAM(BRAM)に格納し，ハードウェア化したラプラシアンフィルタ，2 値化，ラベリング処理を FPGA 上で行う．

本論文では，画像バッファとして，FPGA 内に 256 ライン分を保存するフレームバッファ(以下，構成 1)と，3 ライン分だけのデータを保存するラインバッファ(以下，構成 2)の 2 種類を使用して，画像処理のハードウェア化を行う．

(1) フレームバッファを用いた欠損検出システム[5]

フレームバッファを用いた欠損検出システムを図 3 に示す．スタート信号や座標を生成する制御モジュール(Generate_ADDR_1)，フィルタ処理などを行う画像処理モジュール(BRAM_RW_1, Laplacian, Labeling, True Labeling)と，Frame_Buffer から構成される．Frame_Buffer は BRAM 内の画像 1 枚分を保持するバッファであり，256 ラインのレジスタから構成される．BRAM から 256 ライン分のデータの読み出しと，BRAM へ 256 ライン分のデータの書き込みを BRAM_RW_1 により行う．

以下に，構成 1 の動作を説明する．

①②CPUで画像データをメモリから読み出し,TDI処理を行って,FPGA内のBRAMに書き込む.

③BRAM から全データ(256 ライン)を読み出して Frame_Buffer に書き込む.

④⑤Laplacian により全データに対してラプラシアンフィルタと 2 値化処理を行い，BRAM に書き込む.

⑥BRAM から全データを読み出して Frame_Buffer に書き込む.

⑦⑧Labeling により全データに対して仮ラベルの生成を行い，BRAM に書き込む.

⑨BRAM から全データを読み出して Frame_Buffer に書き込む.

⑩⑪True Labeling(TLabeling)により全データに対してラベルの補正を行い，BRAM に書き込む.

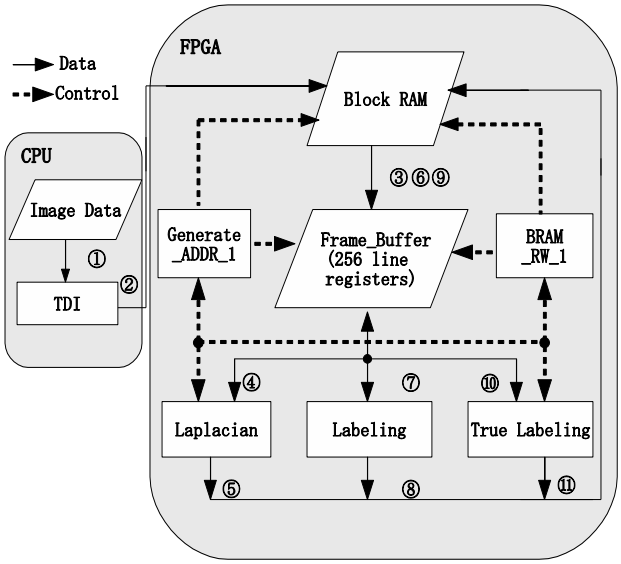


図 3　フレームバッファを用いた欠損検出システム(構成 1)

(2) ラインバッファを用いた欠損検出システム

ラインバッファを用いた欠損検出システムを図 4 に示す．本システムは，スタート信号や座標を生成する制御モジュール(Generate_ADDR)，フィルタ処理などを行う画像処理モジュール(BRAM_RW, Laplacian, Labeling, True Labeling)と，各画像処理モジュールにより生成されたデータを 3 ライン分一時保存する 3 つのラインバッファ(PIX_Register, LAP_Register, LAB_Register)とその制御から構成される．BRAM から 1 ライン分のデータの読み出しと，BRAM への 1 ライン分のデータの書き込みを BRAM_RW により行う．

以下のように動作する．

①②CPU で画像データをメモリから読み出し，TDI 処理を行って，FPGA 内の BRAM に書き込む.

③BRAM_RW により BRAM から 1 ライン分のデータを読み出して PIX_Register に書き込む.

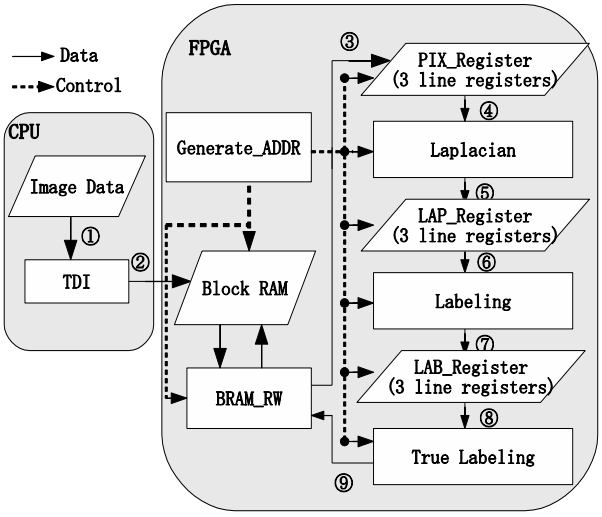


図 4　ラインバッファを用いた欠損検出システム(構成 2)

④⑤Laplacian によりラプラシアンフィルタと 2 値化処理を行い，LAP_Register に書き込む.

⑥⑦Labeling により仮ラベルの生成を行い，LAB_Register に書き込む.

⑧⑨True Labeling(TLabeling)によりラベルの補正を行い，BRAM に書き込む.

256 ラインの画像に対して，③から⑨まで 256 回繰り返し行う．ただし，Laplacian と TLabeling では最初の処理で 2 ライン分のデータが必要であり，例外的な処理が必要である（後述）．

以下，主に構成 2 の欠損検出システムについて述べ，構成 1 と回路規模，実行速度の観点から比較する．

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

表 1　ステージフラグとその動作モジュール

SF	処理モジュール	動作
0	BRAM_RW	BRAM から画像データを 1 ライン分読み出す．BRAM へ補正されたラベルを 1 ライン分書き込む．
1	Laplacian	8 近傍を用いてラプラシアンを行う．
2	Labeling	4 近傍を用いて仮ラベルを生成する．
3	True labeling	8 近傍を用いて仮ラベルを補正する．

3.2　制御モジュールと画像処理モジュール

(1) 制御モジュール

図 4 の Generate_ADDR(G_ADDR)は，制御信号と処理で使用する X，Y 座標，各モジュールのスタート信号を生成している．

座標生成について，1 ラインのデータにおいて，4 つの画像処理モジュールが順番に動作し，それぞれ X 座標が 0 から 255 までのすべての画素に対して処理を行う必要がある．従って，X 座標を 0 から 255 までカウントアップすることを 4 回繰り返してから，Y 座標に 1 を加え，次のラインの処理を行う．X，Y 座標により，ステージフラグ(SF)を生成し，画像処理モジュールを制御する．各ステージでの動作を表 1 に示す．

- SF が 0 の時には，BRAM_RW が 1 ライン分のデータを BRAM から読み出す．
- SF が 1 の時には，Laplacian が 1 ライン分のラプラシアンフィルタ処理と 2 値化処理を行う．
- SF が 2 の時には，Labeling が 1 ライン分の仮ラベルを生成する．
- SF が 3 の時には，TLabeling が 1 ライン分の仮ラベルの補正を行う．

最初の処理において，各画像処理によって対応する領域のデータが揃うタイミングが異なっているため，スタート信号が必要である．スタート信号は Lap_start，Lab_start，TLab_start の 3 つで，図 5 にスタート信号が立ち上がるタイミングを示す．

Lap_start は PIX_Register の出力とラプラシアンフィルタと 2 値化モジュールのスタート信号，Lab_start は LAP_Register の出力と仮ラベル生成モジュールのスタート信号，Tlab_start は LAB_Register の出力とラベル補正モジュールのスタート信号である．

Lap_start は，処理に注目画素以降のデータも必要なので，PIX_Register に 2 ライン分画像データが保存されてから立ち上げる．Lab_start は，処理に注目画素以降のデータは必要ないので，LAP_Register に 1 ライン分ラプラシアンデータが保存されてから立ち上げる．Tlab_start は，処理に注目画素以降のデータも必要なので，LAB_Register に 2 ライン分仮ラベルが保存されてから立ち上げる．

(2) 画像処理モジュール

Laplacian では，PIX_Register から注目画素とその 8 近傍の画素の値を読み出して処理を行う．すなわち，8 近傍の画素値の総和から注目画素値の 8 倍を引き算して，その結果を 9 で割る．得られた商がラプラシアンの結果である．

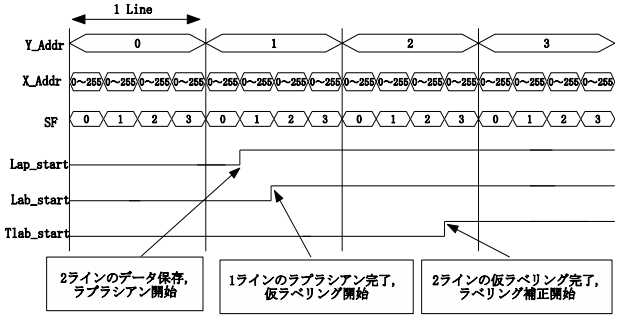


図 5　スタートのタイミング

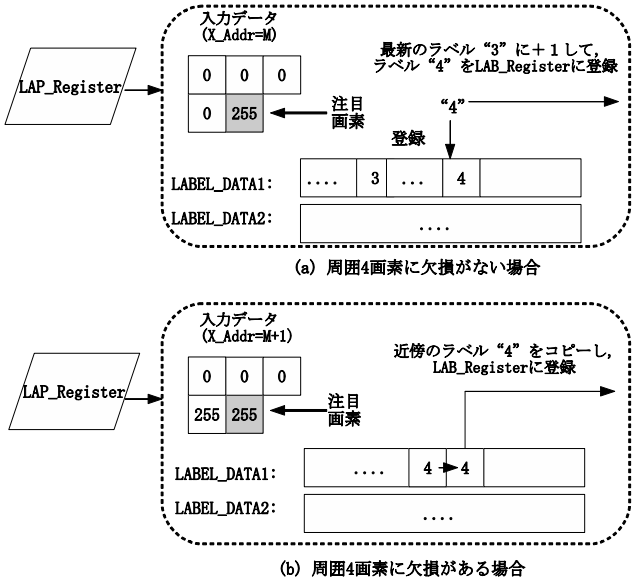


図 6　内部レジスタを用いた仮ラベルの生成

ラプラシアン処理結果を用いて 2 値化を行い，閾値を越えた時は値を 255，そうでない時は 0 として，LAP_Register に保存する．

Labeling では，注目画素の左上，上，右上，左のラベルを参照しながら仮ラベルをつける．周囲に欠損がある場合は該当するラベルをコピーし，周囲に欠損がない場合は新しいラベルを出力する．Labeling は注目画素以降のデータを参照しないので，必要なデータ数は注目画素を含め，5 つである．

仮ラベル生成処理には注目画素の上のラインと左の仮ラベルを参照する必要がある．しかし，LAP_Register に保存されているデータは 2 値化の結果なので，Labeling モジュールでは 2 ライン分の内部レジスタにより生成された仮ラベルを用いて処理を行う．

図 6 に，注目画素が欠損で，その周囲に欠損がない場合とある場合の二つを示す．LABEL_DATA1・2 は仮ラベルを保存する内部レジスタである．

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

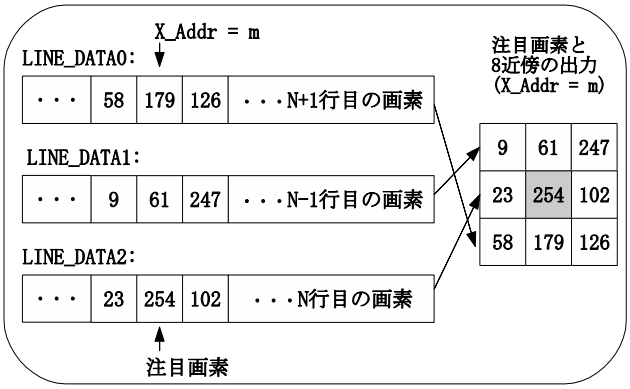


図 7　3 ライン分のラインバッファのデータ転送

図 6(a)の場合は，周囲 4 画素に欠損がないので，最新ラベル 3 に 1 を加えて，4 を仮ラベルとして，LAB_Register と LABEL_DATA1 に保存する．図 6(b)の場合は，左画素が欠損であるため，近傍のラベル 4 を仮ラベルとして，LAB_Register と LABEL_DATA1 に保存する．

Ture Labeling では LAB_Register に保存された仮ラベルをもとに，注目画素の周囲 8 近傍のラベルを比較し，最も小さいラベルを注目画素のラベルとして出力する．

3.3　ラインバッファ

(1) ラインバッファの順番調整

構成 2 において，3 ライン分のデータだけを保存して画像処理を実現するために，ラインバッファへの書き込みと読み出しに関する順番の制御が必要である．図 7 に 3 ライン分のラインバッファと，N ライン目のデータが注目画素として取りだされている様子を示す．出力の順番は G_ADDR により制御されている．

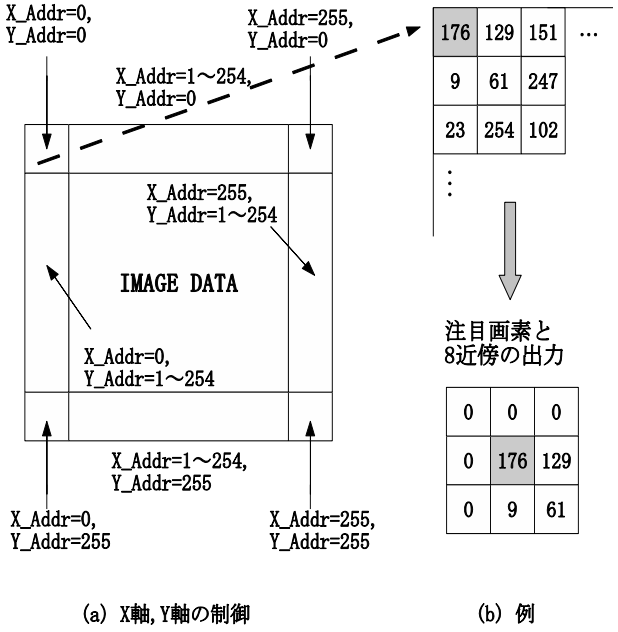
PIX_Register,LAP_Register,LAB_Registe が LINE_DATA を持ち，前の動作モジュールにより生成されたデータを保存する．LINE_DATA の保存順番は 0 →1→ 2 →0 の繰り返しを行っている。

図 7 では，注目画素が LINE_DATA2 と仮定しているため，データの保存順が LINE_DATA1, LINE_DATA2, LINE_DATA0 となり，それにより，図 7 のような 8 近傍のデータとなる．

(2) 端処理

対応する領域に隣接するデータがない場合が存在するため，端の処理が必要になる．これらの処理は，G_ADDR から入力される X,Y 座標を使用して制御を行う．

図 8(a)は 8 通りの端を示す．これらは対応する領域に隣接するデータがないので，処理モジュールに 0 を出力する．図 8(b)は端処理の例である．注目画素が画像の 1 画素目で，X，Y 座標が両方 0 の場合である．端処理により，左と上のラインを 0 にして，8 近傍を生成する．



(a) X軸,Y軸の制御　　(b) 例

図 8　端処理の制御

4. FPGA 上での実験

4.1　実験条件

実験には TDI を 128 回行って，雑音除去をあらかじめ施したデータを使用し，それを FPGA 内の BRAM に保存して画像処理を行う．FPGA 上で構成 1 と構成 2 の回路規模や処理時間を計測する．処理時間は，BRAM からデータを読み出し，全データに対して画像処理を行い，BRAM に保存するまでの時間とする．

FPGA は Xilinx 社の Virtex5(XC5VFX70T) ML507 評価ボードを用いる．デザイン設計と論理合成には同社の設計ツール ISE を用い，ISim でシミュレーションを行った．

また，処理時間の比較を行うために，同じ画像処理を CPU 上のソフトウェアで実行して処理時間を計測する．その動作環境は Intel Core 2 Quad CPU Q9400 2.67GHz，実装メモリ 4.00GB，OS が Windows7 Ultimate である．

4.2　実験結果

構成 1 と構成 2 の性能評価を行うため，2 つの構成を論理合成し，回路規模や動作周波数，処理時間を計測し，速度向上率とハードウェア規模の比較を行った．

(1) 回路規模

表 2 に 2 つの構成の全体の回路規模を示す．()内は Virtex5 のハードウェア使用率である．使用 LUT 数について，構成 1 はデバイスのハードウェア容量を超え，272%となった．一方，構成 2 は構成 1 の 1.9%で，デバイスのハードウェア容量の 5%しか使用していない．その理由として，構成 1 では 256 ライン分のフレームバッファを使用しているが，構成 2 では 3 ライン分のラインバッファしか使用していないからである．

レジスタ数・LUT-FF ペア数について，構成 2 は構成 1 と比べると少々多くなる．これは，構成 2 を実現するために，3 ライン分のデータを保存するラインバッファなどの制御信号の数が多くなったことが考えられる．しかし，両構成ともデバイスのハードウェア容量の 1%未満に収まっている．

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

表 2　全体の回路規模

	構成 1 (256 Line)	構成 2 (3 Line)	Virtex5
レジスタ数	20 (0.04%)	52 (0.12%)	44800
LUT 数	121772 (272%)	2262 (5.05%)	44800
LUT-FF ペア数	8 (0.35%)	19 (0.83%)	2295
入出力数	51 (7.97%)	51 (7.97%)	640
Block RAM と FIFO 数	16 (12.5%)	16 (12.5%)	128

表 3　各モジュールの回路規模

	レジスタ数		LUT 数		LUT-FF ペア数	
構成	1	2	1	2	1	2
RF	0	0	119116	1765	0	0
G_ADDR	19	27	47	60	19	27
Laplacian	0	0	77	105	0	0
Labeling	9	9	389	397	9	9
TLabeling	0	0	148	148	0	0

表 3 に 2 つの構成の各モジュールの回路規模を示す．RF(Register File)はバッファ(フレームバッファ，またはラインバッファ)，及びバッファと BRAM 間のデータの入出力のコントロールを含む．RF は他のモジュールとべて，使用する LUT 数が遥かに多くなっている．また，RF での LUT 使用数について，構成 2 は構成 1 の 1.5%となっている．LUT 数が多い原因は，RF は構成 1 では 256 ライン分のデータを保存するモジュールであり，構成 2 では PIX_Register，LAP_Register，LAB_Register がそれぞれ 3 ライン分のデータだけを保存するモジュールとなっているからである．

構成 2 の制御モジュール G_ADDR において，レジスタ数・LUT 数・LUT-FF ペア数が構成 1 と比べて増えている．これは，構成 2 では PIX_Register，LAP_Register，LAB_Register に対する制御信号の数が多いからである．

構成 2 の Laplacian と Labeling において，ハードウェア規模が構成 1 と比べて少々増えている．これは，3 ライン分のデータの順番調整のためである．

以上により，構成 2 では構成 1 と比べて，ハードウェア量を大幅に削減できることを確認した．また，両構成とも LUT を大量に使用し，レジスタの使用量が少ないことが分かる．画像バッファを LUT により構成したからである．

(2) BRAM へのアクセス回数

フレームバッファとラインバッファはレジスタであり，LUT から構成されている．もし，これらのバッファが BRAM から構成されているとすると，画像処理を行うために周囲の画素が必要になり，ラプラシアンフィルタの実行では注目画素と 8 近傍を揃えるために BRAM へのアクセス回数は 9 回となる．仮ラベル生成では周囲の 4 近傍と注目画素を揃えるために BRAM へのアクセス回数は 5 回である．ラベル補正では注目画素と 8 近傍を揃えるために BRAM へのアクセス回数は 9 回である．

本システムでは，バッファが LUT から構成されているので，BRAM へのアクセス回数は以下のようになる．

構成 1 では，各処理を開始する前に 256 ライン分のデータを BRAM から読み出し，画像処理終了後に BRAM に保存する．そのため，構成 1 の BRAM アクセス回数は，256×256×3×2 = 393216　となる．

構成 2 では 1 ラインずつ処理を行うため，Laplacian を処理する前に 1 ライン分のデータを BRAM から読み出し，TLableing の処理後にデータを BRAM に保存する．そのため，構成 2 の BRAM アクセス回数は 256×256×2 = 131072 となり，構成 1 と比べて 1/3 となる．

(3) 実行速度

表 4 に，構成 1 と構成 2 の動作周波数と実行時間，及び CPU ソフトウェアでの実行時間を示す．また，ソフトウェア処理時間に対する速度向上率を示す．

構成 1 の最大動作周波数は 69.8MHz で，構成 2 より 1.79 倍速い．

BRAM ではアドレスを与えてからデータを得るまでのレイテンシが 2 クロック必要なので，アドレスを先行発行している．

構成 1 では，1 ピクセルに対して，BRAM から画像データの読み出し，ラプラシアン処理と結果の BRAM への保存，BRAM からラプラシアンデータの読み出し，仮ラベリング処理と結果の BRAM への保存，BRAM から仮ラベリングデータの読み出し，ラベリングの補正と結果の BRAM への保存の 6 クロックが必要である．

構成 2 は BRAM からのデータの読み出し 1 回，及びラプラシアン，仮ラベリング，ラベリング補正の 3 つの動作(ラベリング補正では BRAM への書き込みも含まれる)で，合わせて 4 クロックが必要である．

構成 1 の最大動作周波数は構成 2 より 1.79 倍速い．一方，全体の処理速度について，構成 1 は構成 2 より 1.19 倍速い．その原因は，構成 2 では BRAM へのアクセス回数が構成 1 の 1/3 となっているからである．

表 4　動作周波数と実行時間

	CPU*	構成 1	構成 2
動作周波数(MHz)	—	69.8	39.1
実行時間/ピクセル(ns)	2274.2	85.9	102.4
全体の実行時間(ms)	149.2	5.63	6.71
速度向上率（倍）	1	26.5	22.2

*: Intel Core2 Quad CPU Q9400 2.67GHz，実装メモリが 4.00GB，OS が Windows 7 Ultimate

5. パイプライン処理による高速化の検討

5.1　方針

本システムの 4 つの画像処理モジュールは独立に動作するので，パイプライン処理による高速化が可能である．

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

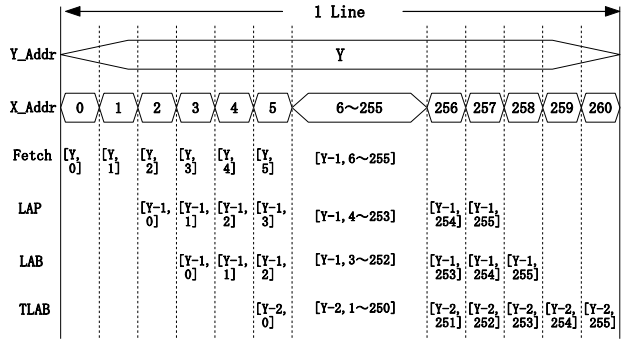


図 9 1 ライン生成のタイミングチャート

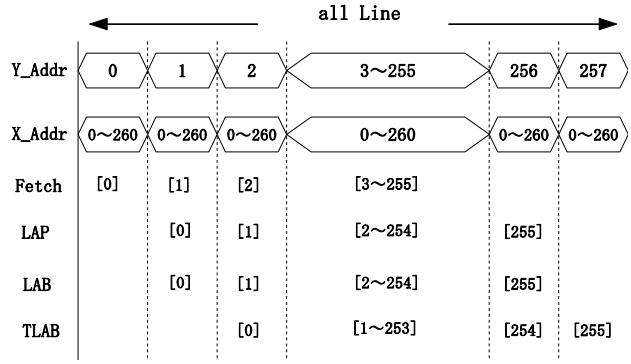


図 10　全ライン生成のタイミングチャート

本システムでは，注目画素のラインと前後のラインを含めて，3 ライン分のレジスタを用意している．従って，図 4 の 3 つのレジスタをパイプラインレジスタとして使用する．また，各モジュール間でデータ依存があるため，タイミングの調整を行う必要がある．そこで，主に X 軸,Y 軸のアドレスを拡張し，調整を行う．

5.2　1 ライン生成のパイプライン化処理

図 9 に 1 ライン生成のための Y 軸と X 軸の調整を示す．Fetch(BRAM からデータの読み出し)，LAP(ラプラシアンと 2 値化)，LAB(仮ラベルの生成)，TLAB(仮ラベルの補正)の各行に，処理を行う注目画素の[Y 座標,X 座標]を示す

Fetch では，X_Addr が 0 から 255 まで行う．

LAP では，注目画素の次の画素を参照する必要があるので，必要なデータを揃えた上で，X_Addr が 2 から 257 まで LAP を行う． X 軸の座標は X_Addr-2 である．

LAB では，注目画素の左の LAP データに依存しているので，必要なデータを揃えた上で，X_Addr が 3 から 258 まで LAB を行う． X 軸の座標は X_Addr-3 となる．

TLAB では，注目画素の右の仮ラベルを参照する必要があるので，必要なデータを揃えた上で，X_Addr が 5 から 260 まで TLAB を行う． X 軸の座標は X_Addr-5 となる．

これにより，パイプライン処理では 1 ラインを実行するのに 261 クロックが必要で，構成 2(4×258 クロック)と比べると約 1/4 のクロック数に削減できる．

5.3　全ライン生成のパイプライン化処理

図 10 に全ライン生成のための Y 軸の調整を示す．Fetch，LAP，LAB，TLAB の各行に，処理を行う注目画素の Y 軸の座標を示す．

Fetch では，Y_Addr が 0 から 255 まで BRAM から画像データを読み出し，PIX_Register に保存する．

LAP では，注目画素の次のラインの画素を参照するので，必要なデータを揃えた上で，Y_Addr が 1 から 256 まで LAP を行う．生成されたデータを LAP_Register に保存する．

LAB では，注目画素の次のラインの画素を参照しないので，Y_Addr が 1 から 256 まで LAB を行う．生成されたデータを LAB_Register に保存する．

TLAB では，注目画素の次のラインの画素を参照するので，必要なデータを揃えた上で，Y_Addr が 2 から 257 まで TLAB を行う．生成されたデータを BRAM に保存する．

図 10 で，4 つの画像処理モジュールが同時に動作するのは Y_Addr が 2 から 255 までであり，最初の 2 ラインと最後の 2 ラインは一部が同時に動作する．

6. おわりに

本論文では FPGA を用いて，液晶用ガラスの欠損検出システムの高速化を行った．本システムでは,256 ラインのフレームバッファ(構成 1)と 3 ラインのラインバッファ(構成 2)の 2 種類を用いて，評価を行った．ソフトウェア処理と比べて，構成 1 では 26.5 倍，構成 2 では 22.2 倍の速度向上が得られた．一方，LUT 数に関して，構成 2 が構成 1 の 1.9%まで削減できた．これにより，3 ラインのラインバッファを使用して，より小規模なハードウェアで，液晶用ガラスの欠損検出システムの高速化が実現できた．

また，構成 2 の 4 ステージでのパイプライン処理について検討し，処理時間が 1/4 に短縮できることを示した．現在，パイプライン処理の実装を進めている．

謝辞

本研究は，株式会社ケー・デー・イーとの共同研究であり，画像データを提供して頂いた．多大なご協力と有益なコメントを頂いた三宅淳司氏と西田洋隆氏に厚く感謝します．

参考文献

[1] Gyu-Bong. Lee, Woo-Seob. Kim, Yun-Su. Chung and Joon-Jae Lee, “Adaptive Surface Fitting for Inspection of FPD Devices Using Multilevel B-spline Approximation,” TENCON 2005, pp.1-4, 2005.

[2] Woo-Seob Kim, Jong-Hwan Oh, Yun-Su Chung, Il Choi and Kil-Houm Park, “The Detection of Curve-type Defects in the TFT-LCD panels with Machine Vision,” TENCON 2005, pp.1-5, 2005.

[3] Yasuaki Ito and Koji Nakano, “Low-Latency Connected Component Labeling Using an FPGA ,” International Journal of Foundations of Computer Science, pp.405-425, 2010.

[4] K. Benkrid, S. Sukhsawas, D. Crookes and A. Benkrid, “An FPGA-Based Image Connected Component Labeller,”Field Programmable Logic and Applications, Lecture Notes in Computer Science, Vol.2778, pp.1012-1015, 2003.

[5] 松山圭輔，孟林，天井康雄，山崎勝弘，“FPGA を用いた液晶用ガラス欠損検出システムの高速化，” IEICE Technical Report RECONF2012-37，pp.79-84，2012.

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.