卒業論文

FPGA を用いた液晶用ガラス欠損検出システムの並列化

氏　　名　：　野尻　直人

学籍番号　：　2260090048-3

担当教員　：　山崎 勝弘 教授

提 出 日　：　2013 年 2 月 20日

立命館大学　理工学部　電子情報デザイン学科

内容梗概

近年、液晶用ガラスはテレビやパソコンディスプレイ、スマートフォンなど、様々な分野で使用されている。液晶用ガラスの欠損を検出するためには、大量の画像データの処理が必要であり、その高速化が極めて重要である。本研究では、FPGA ボードを用いて液晶用ガラス欠損検出システムを並列化することで高速化を図る。本システムでは、ラプラシアンフィルタ/2 値化、ラベリングの各処理を並列に行うことで速度を向上させることを目的とする。本研究では、画像処理を高速化に行うために、ラプラシアンフィルタ/2 値化、ラベリングの各 module を 2 並列処理することで、実験を行った。

実験では、逐次処理と 2 並列処理の構成を用いてラプラシアンフィルタ/2 値化，ラベリングのハードウェア化を行い，回路規模や処理時間を計測し，異なるハードウェア量で同じ画像処理を行う回路の比較を行った。実験の結果として、2 並列処理の構成では逐次処理よりも約 1.64 倍の速度向上を得られた。ハードウェア量に関しては、約 50%削減することが出来た。

目次

図目次

表目次

1. はじめに

近年、PC、液晶テレビ、スマートフォンなどの需要が高まってきており、液晶用ガラスの需要も高まってきている。これらの需要に見合う生産スピードの確保と，更なる高速化を実現するために，生産した液晶用ガラスに傷がないかを調べる検品に要する時間も短縮されることが望まれる。それにより欠損検出も重要となる。本研究では、液晶用ガラス欠損検出の高速化を目的としている。これを解消するために本研究では，FPGA ボードを用いて高速化を図る。FPGA とは、自分で論理回路を作成し、書き換え可能な集積回路である。近年では、テクノロジーの進化により高集積化、高性能化、低消費電力化、小型化、再設計可能な特徴を兼ね備えている。

まず TDI （Time Delay Integration）により原画像のノイズを軽減させる。次にラプラシアンフィルタをかけて画像内のエッジを検出し 2 値化を行う。そして、ラベリングを行って画像内の各オブジェクトを識別することにより、欠損を検査する。TDI とは、同じ画像の画素をずらして撮影を繰り返し、その共通部分を重ね合わせ平均値を取ることで、ノイズの影響が小さい画像を得る手法である。ラプラシアンフィルタは画像に輪郭強調処理を行い、欠損の存在を鮮明にすることで、小さな欠損であっても検出できるようにするアルゴリズムである。それによって液晶用ガラスの小さくて見逃しやすい欠損を発見しやすくする。3 つ目がラベリングフィルタである。ラベリングフィルタは画像にラベリング処理を行って液晶用ガラスに存在する各欠損にラベルとして番号を割り振り、それによって確認すべき箇所の数の目安を作る。それによって液晶用ガラスに存在する欠損の数の大まかな目安を作り、確認作業を高速化する。

本研究では、液晶用ガラス欠損検出画像処理の FPGA を用いた高速化を目指している。これらを用いることで液晶用ガラス欠損検出システムを高速に行う。メモリの使用については、BRAM から画素データを取出し、処理後のデータをそのまま保存すると値が更新されてしまい、次の処理での正確な結果が得られない。そこで、レジスタファイルをあらかじめ用意し、処理に必要なデータを格納することにする。さらに、256*256 のレジスタファイルを 2 並列化すること処理速度を向上することを目標にする。

本論文では、液晶用ガラス欠損検出のための画像処理の方法，逐次処理と 2 並列化の比較、シミュレーション結果、及び考察について述べる。

2. 液晶用ガラス欠損検出システム

2.1 欠損検出アルゴリズム

本システムでは，画像データを FPGA ボード内の BRAM (Block RAM)に保存している。あらかじめ撮影しておいた液晶用ガラスの欠損画像を使用して，画像処理によって欠損を検出していく。画像処理の流れを図 1 に示す。また、処理の手順を以下に示す。

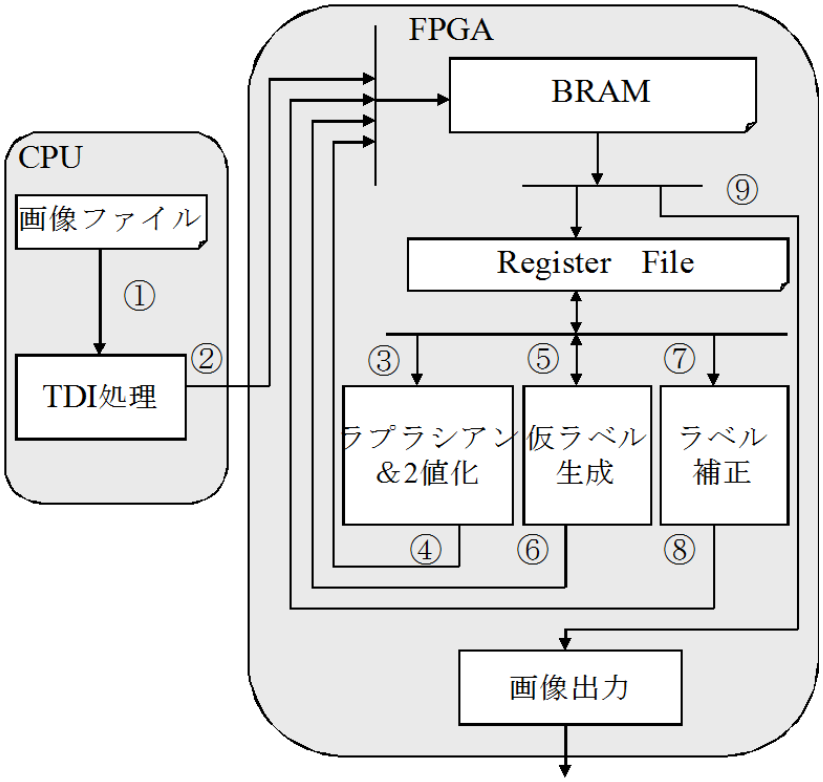


図 1: 液晶用ガラス欠損検出の流れ

①画像ファイルから画像データを取り出し，TDI 処理を行う。
②TDI 処理後のデータを BRAM に保存する。
③レジスタファイルに画像データを転送し，ラプラシアン&2 値化処理を行う。
④2 値化処理後のデータを BRAM に保存する。
⑤レジスタファイルに画像データを転送し，仮ラベルを生成する。
⑥生成した仮ラベルを BRAM に保存する。
⑦レジスタファイルに仮ラベルを転送し，ラベルの補正処理を行う。
⑧補正後のラベルを BRAM に保存する。
⑨BRAM から画像データを転送し，出力する。

2.2　TDI

TDI（Time Delay Integration）とは、同じ画像の画素をずらして撮影を繰り返し、その共通部分を重ね合わせ平均値を取ることで、ノイズの影響が小さい画像を得る手法である。

3	**3**	**3**	**3**	**3**	**3**	**3**
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6

(a)理想の画像

3	**4**	**3**	**3**	**2**	**4**	**3**
4	3	4	3	4	4	4
4	5	5	4	5	4	5
6	6	5	6	6	5	6

(b) 実際の画像

図 2: 撮影時のノイズ混入

図 2 の(a)が理想の画像なのだが、実際は図 2(b)のようにノイズ（赤い数字の部分）が入ってしまう。このノイズの部分をなるべく左の画像の状態に近づけるために画素の平均化を行う。画素の平均化のために、撮影した画像の画素を 1 画素ずつずらした画像を複数枚用意し、共通部分を重ねる。その導出過程を図 3 に示す。

0	**0**	**0**	**0**	**0**	**0**	**0**
1	1	1	0	1	1	1
2	2	2	2	2	2	1
3	4	3	3	2	3	2
4	4	4	3	4	4	4
5	5	5	5	5	4	5
6	6	6	6	6	6	6

1	**0**	**1**	**1**	**1**	**1**	**1**
2	2	1	2	1	2	2
3	4	3	3	2	3	2
4	4	4	4	4	4	4
5	5	5	5	5	3	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7

2	**2**	**2**	**2**	**2**	**2**	**2**
3	3	3	3	3	4	3
4	4	4	3	4	4	4
4	5	5	5	5	4	5
6	6	5	6	6	6	6
7	7	7	7	7	7	7
8	7	8	8	8	8	8

3	**3**	**3**	**3**	**3**	**3**	**3**
4	3	4	4	4	4	4
5	5	5	3	5	5	5
6	6	6	6	6	5	6
7	7	7	7	7	7	7
8	8	8	8	8	8	8
9	9	9	9	9	9	9

図 3: 画素の平均化

図 3 より、共通している部分は 4 行あるので、この部分をそれぞれ平均して出力する。

3	**3**	**3**	**3**	**3**	**3**	**3**
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6

(a)理想の画像

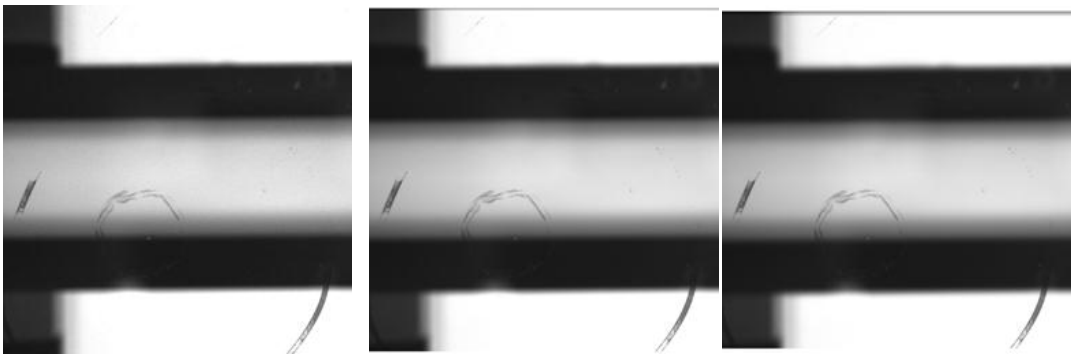
3	**3.5**	**3**	**3**	**2.5**	**3.25**	**2.5**
4	3.75	4	3.5	4	4	4
4.75	5	5	4.5	5	4	5
6	6	5.75	6	6	5.75	6

(b)平均化後の画像

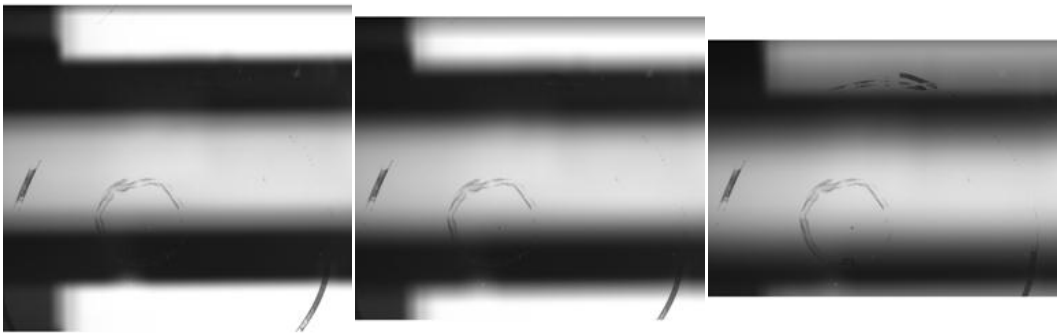
図 4: TDI 処理

図 2(b)と、図 4 の(b)と比較して理想の画像である図 4(a)により近付いたことが分かる。このように、画像を重ね合わせ、平均値をとることによりノイズを減らすことが可能である。画像を読み込む前処理として TDI を行うことにより、よりスムーズに画像処理ができる。

TDI 処理は画像の共通部分を重ね合わせるので，処理に使用する画像の枚数が多ければ多いほど画像が縦方向に小さくなっていく。図 3 では画像を 3 枚使用しての結果となるので雑音除去の影響は小さいが，使用する画像の枚数を 10 枚，20 枚と増やすほど平均値は真の値に近づいていき，よりノイズの影響が小さい画像が生成される。実験では，液晶用ガラスの欠損画像を 128 枚使用して TDI 処理を行っている。TDI 処理に使用する画像の枚数による画像の変化を図 5 に示す。



（a）1 枚　　（b）8 枚　　（c）16 枚



（d）32 枚　　（e）64 枚　　（f）128 枚

図 5: TDI 処理後の画像の変化

2.3　ラプラシアフィルタ/2 値化

ラプラシアンフィルタとは、画像中に含まれる物体の輪郭部分（エッジ）を抽出するフィルタである。画像に対して 2 次微分をするフィルタで、2 変数関数 f(x,y)のラプラシアンは、偏微分を使って以下の式で表せる。

$$L(x,y) = \frac{\partial^2}{\partial x^2} f(x,y) + \frac{\partial^2}{\partial y^2} f(x,y)$$

また、これを差分形式で表すと以下のようになる。

$$\begin{aligned} \nabla^2 f(x,y) &= f_{xx}(x,y) + f_{yy}(x,y) \\ &= f(x-1,y) - 2f(x,y) + f(x+1,y) + f(x,y-1) - 2f(x,y) + f(x,y+1) \\ &= f(x-1,y) + f(x+1,y) + f(x,y-1) + f(x,y+1) - 4f(x,y) \end{aligned}$$

これを係数で表現すると

0	**1**	**0**
1	-4	1
0	1	0

となる。これを 4 近傍のラプラシアンフィルタという。

さらに、45°方向も含めた係数は

1	**1**	**1**
1	-8	1
1	1	1

となる。これを 8 近傍のラプラシアンフィルタという。本研究では、8 近傍を用いている。
処理の流れを図 6 に示す。

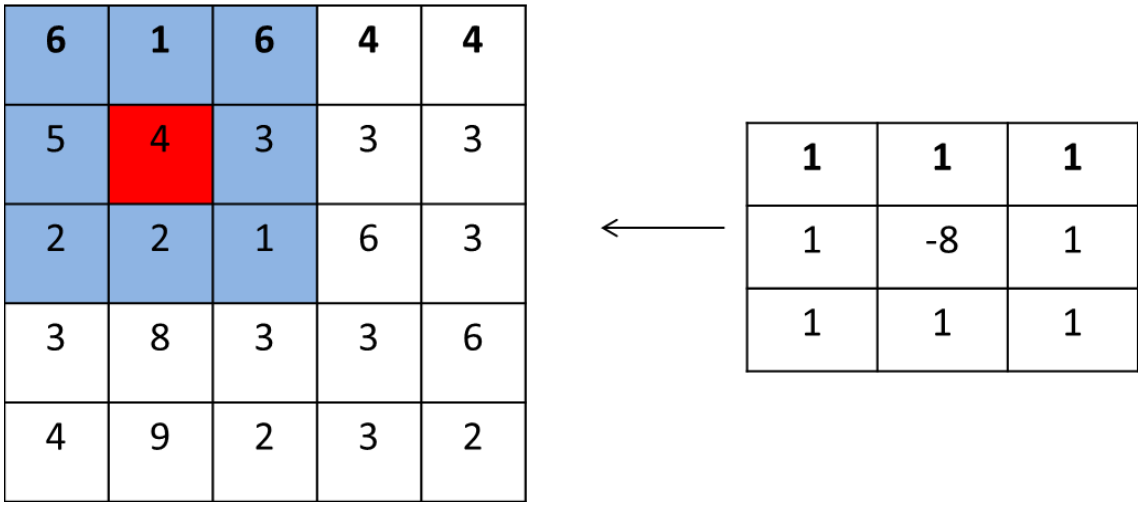
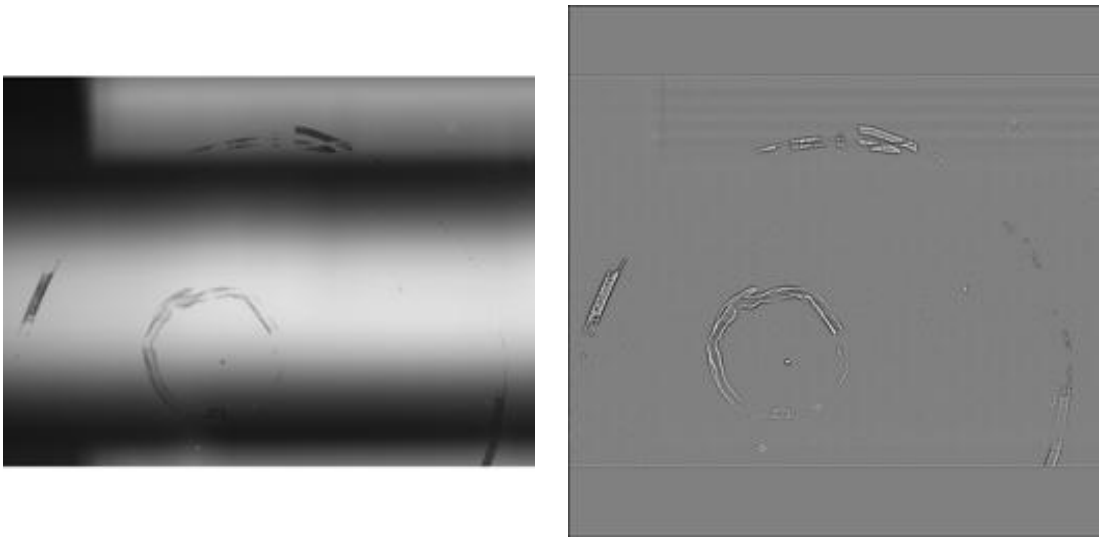


図 6: ラプラシアンフィルタ

図 6 において、(1.1)座標の 4 でのフィルタ処理後の値を f'(1.1)とする。

$$f'(1.1)=6\times 1+1\times 1+6\times 1+5\times 1+4\times(-8)+$$
$$3\times 1+2\times 1+2\times 1+1\times 1$$
$$=-6$$

ラプラシアンフィルタによってエッジ検出された画像は，輪郭の強さに応じた濃淡画像となり，欠損部分が浮いているような画像に変化する。ラプラシアンフィルタによる画像の変化を図 7 に示す。



(a)TDI 処理後の画像　　(b)ラプラシアンフィルタ処理後の画像

図 7: ラプラシアンフィルタ処理後の変化

2 値化とは濃淡のある画像を白と黒の 2 階調に変換する処理です。ある閾値を定めて、各画素ごとの値が閾値を上回っていれば白、下回っていれば黒に置き換えます。画像を 2 値化することで、画像からの検出対象の抽出が容易になります。また、判定処理なども高速に実行できる。ある入力画像に対して，閾値を“120”と設定した場合の 2 値化処理の様子を図 8 に示す。

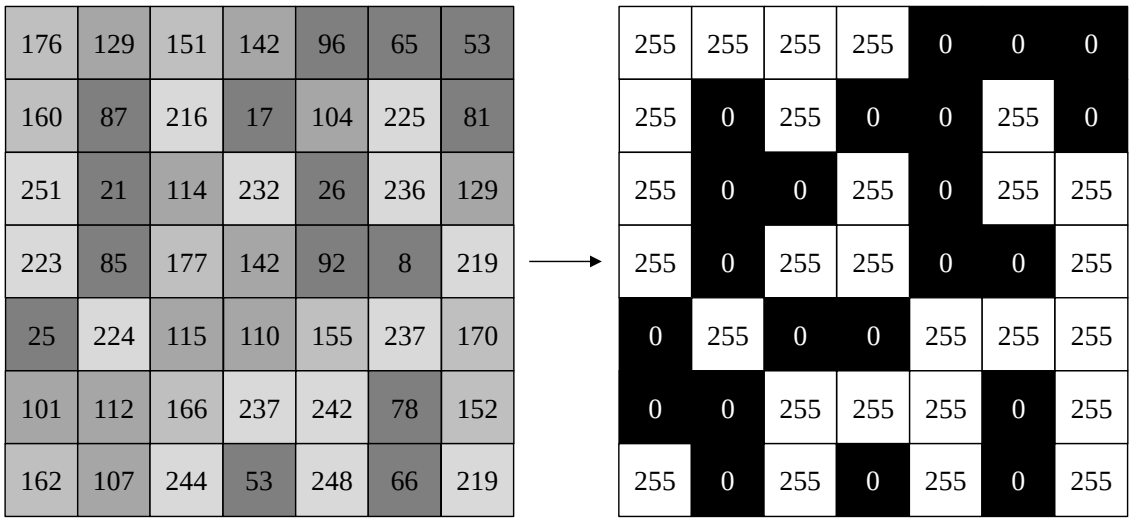
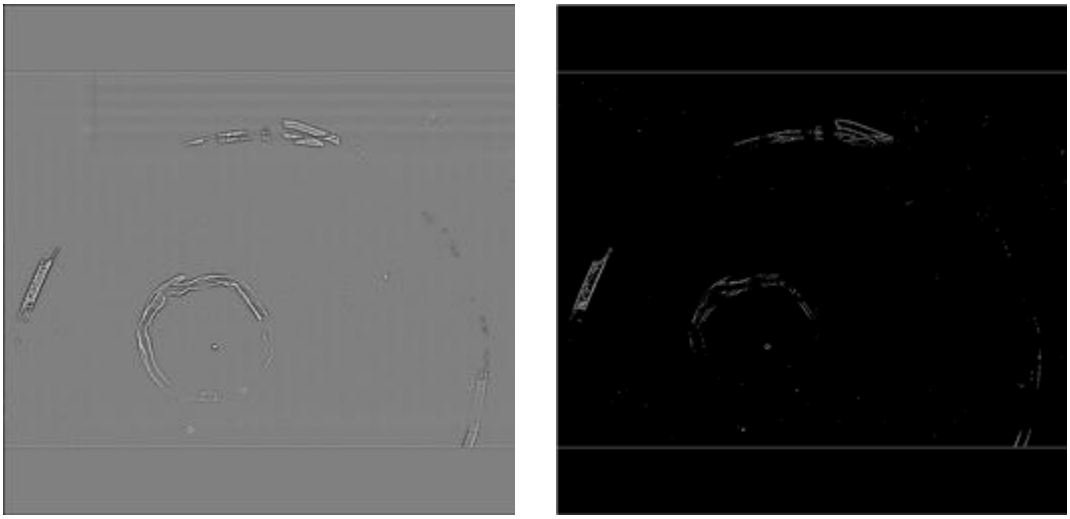


図 8: 2 値化を用いた閾値処理

2 値化によって閾値処理された画像は，欠損部分は白く，欠損以外の部分は黒くなった画像に変化する。2 値化による画像の変化を図 9 に示す



(a)フィルタ処理後の画像　　(b)2 値化処理後の画像

図 9: 2 値化処理による画像の変化

2.4　ラベリング

ラベリングとは、つながっている画素（連結成分）に同じラベル（番号）をつけ異なった連結部分には異なった番号を付ける処理のことであり、2 値画像処理では非常に重要な処理である。この処理により個々の連結成分に分離することができ、各連結成分の特徴量を抽出することができる。図 6(a)のように、1 回目のスキャンでは、2 値化後の画像データを元に仮ラベルをつけていく。2 回目のスキャンでは、仮ラベルがつけられた画像データを元にラベルの補正を行い、図 6(b)の画像を生成する。図 10（a）の 1 の領域と 2 の領域は連結しているので、ラベルの値を 2 から 1 に補正して図 10(b)のようになる。

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	2	1	1	1	0	0	0	0
0	0	2	2	1	1	0	0	0	0
0	0	0	2	2	0	0	0	0	0
0	3	0	0	0	0	0	0	0	0
0	3	3	0	0	0	4	4	0	0
0	0	0	0	0	0	4	4	0	0
0	0	0	0	0	0	0	0	0	0

(a)仮ラベリングの画像

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0
0	0	0	1	1	0	0	0	0	0
0	3	0	0	0	0	0	0	0	0
0	3	3	0	0	0	4	4	0	0
0	0	0	0	0	0	4	4	0	0
0	0	0	0	0	0	0	0	0	0

(b)補正後の画像

図 10: ラベリング

ラベリングのアルゴリズムとしては、図 11 に示すように、画素上を走査して、ラベルが付けられていない画素を見つけ、新しいラベルを付ける。そのラベリングされた画素に対して、連結している画素に同じラベルを付ける。さらに今ラベル付けした画素と連結しているすべての画素に同じラベルを付ける。これで、ひとつの連結成 9 分全体に同じラベルが付けられたことになる。その後、また初めのステップに戻り、まだラベルの付けられていない画素が見つかったら、新しいラベルを付けて同じ操作を繰り返す。画像全体の走査が終わったとき、すべての処理が終了する。

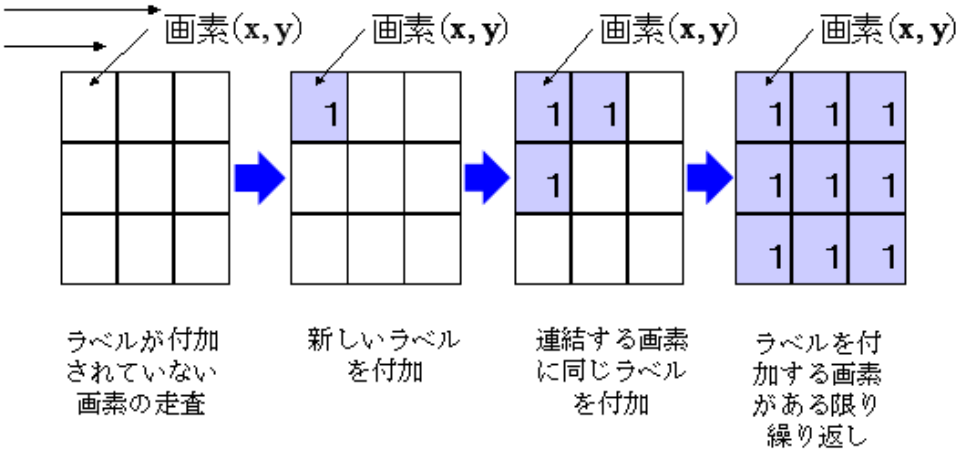


図 11: ラベリングのアルゴリズム

ラベルの補正を行う 2 回目の走査では，図 12 のように注目画素の左上，上，右上，左，右，左下，下，右下の全方位のラベルを参照し，注目画素が持つラベルよりも小さいラベルがあればそれをコピーしてラベルをつけ直し，周囲 8 画素に注目画素よりも小さいラベルがない場合は補正を行わず，注目画素が持っているラベルをそのまま出力する。

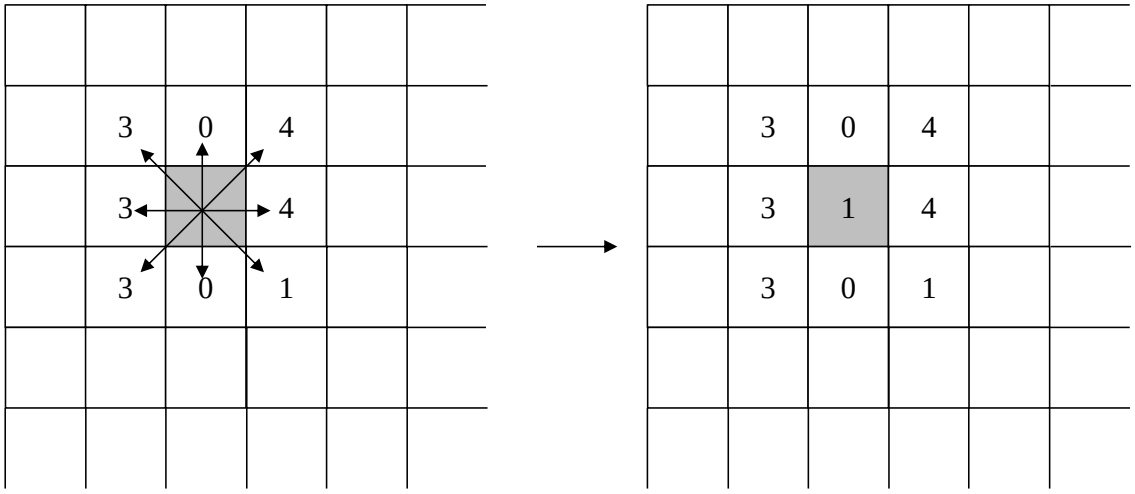
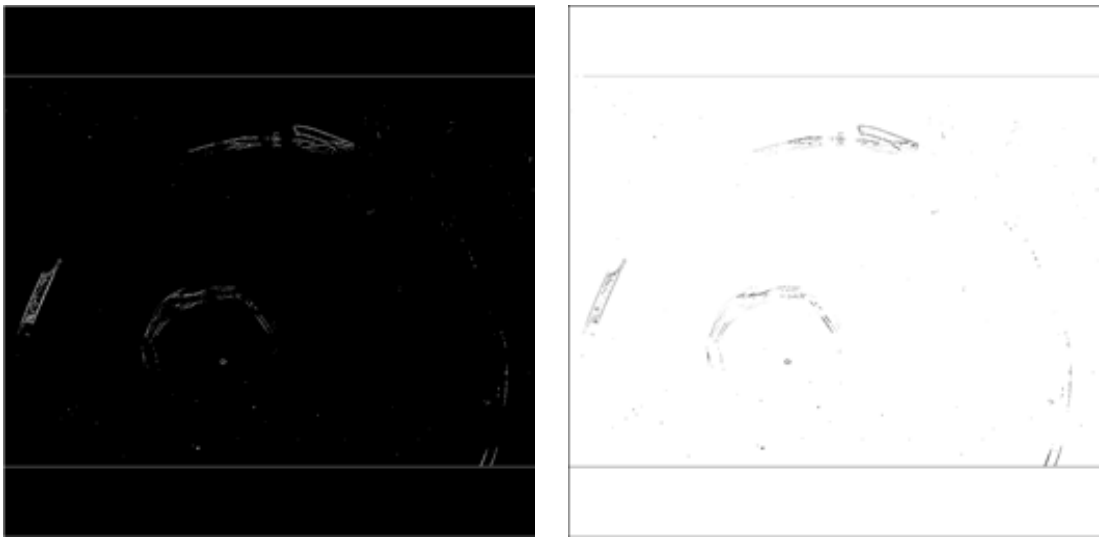


図 12: ラベリング補正時の参照データ

1 回目と 2 回目の走査を合わせたラベリングによる画像の変化を図 13 に示す。



(a)2 値化処理後の画像　　(b)ラベリング後の画像

図 13: ラベリング後の画像の変化

3. 画像メモリの構成

3.1　全画面用レジスタファイル

各モジュールの処理において、ラプラシアンフィルタでは，画像自体が持っている画素データを参考にマスクパターンとかけていく必要がある。そのため BRAM から画素データを取り出して処理後のデータをそのまま保存してしまうと，値が更新されてしまうので，次の処理で正しい結果が得られなくなる。そこで，レジスタファイルを用意して，はじめに必要な画素データをレジスタファイルに入れてから処理することにした。なお、本研究では、このレジスタファイルを用いて、並列処理を行う。

ラベリングにおいても、レジスタファイルを用いてラベルづけを行う。今回使用する画素は同一なので、ラプラシアンフィルタで使用したレジスタファイルを用いる。

レジスタファイルのサイズは、256*256 の符号付き 12bit である。256*256 の全画面用レジスタファイルを図 14 に示す。

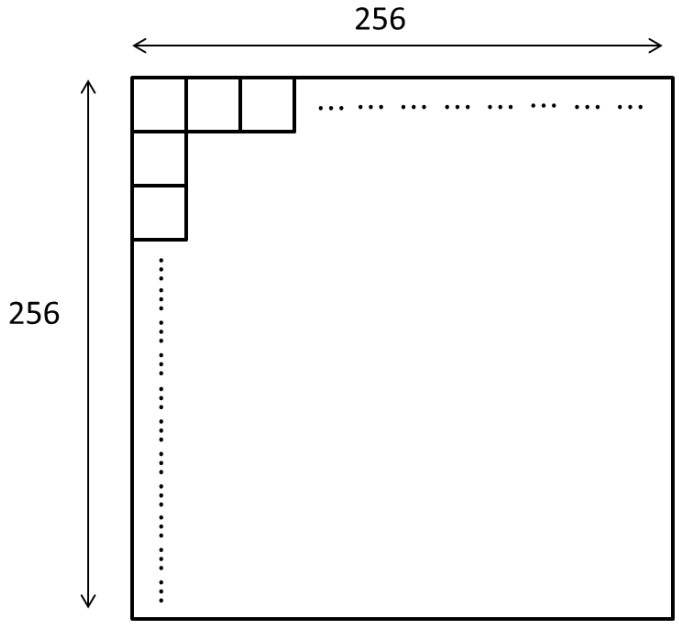


図 14:　全画面用レジスタファイル

3.2　3 行分のレジスタファイル

3 行分のレジスタファイルとは、3.3 の全画面用レジスタファイルと同様の処理を行うが、サイズが 256*3 の符号付き 12bit のレジスタファイルである。3 行分のレジスタファイルを図 15 に示す。この方法を用いることで処理速度を向上させることができると思われる。

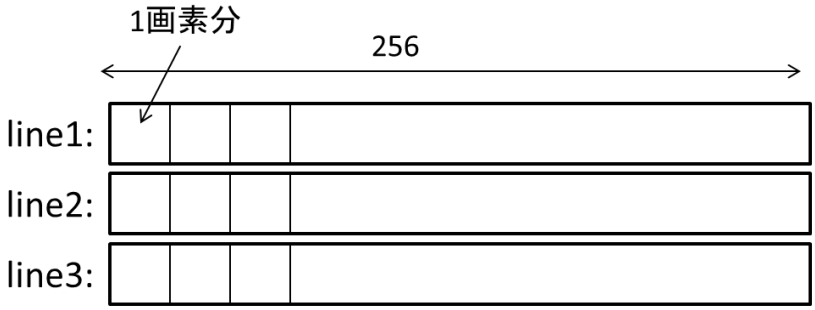


図 15: 3 行分のレジスタファイル

3.3　Block RAM の構成

本研究では、Block RAM は Xilinx ISE Project Navigator の CORE　GENERATOR で生成した IP を使用した。BRAM の容量は，画像データの各画素が 8bit なので 256*256*8 とし, 処理が終わったあとにデータを書き込むためイネーブル信号がつけてある。図 16 に，BRAM のモジュール構成を示す。入力信号は，16bit のアドレス，8bit の入力データ，ライトイネーブルとクロック信号である。出力信号は，8bit の出力データである。

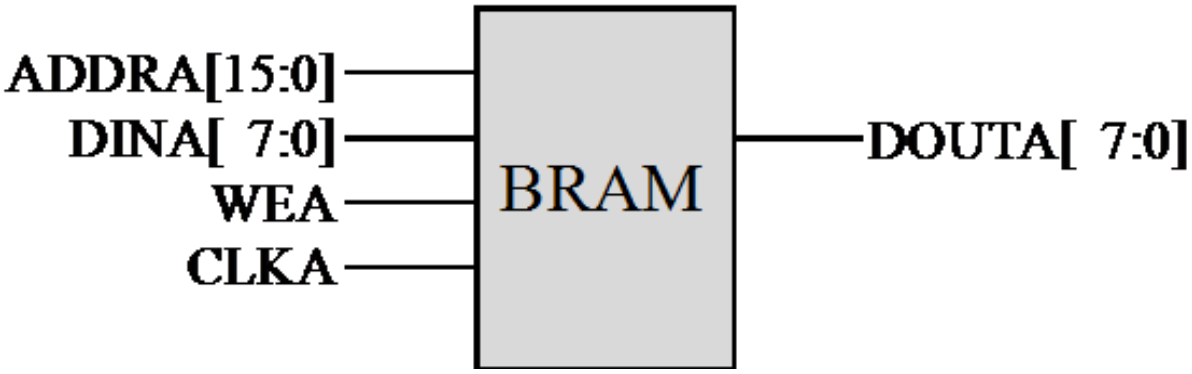


図 16:　BRAM へのアクセス方法

3.4 並列化による欠損検出システムの構成

本システムでは，画像データを FPGA ボード内の BRAM (Block RAM)に保存している。TDI ではオリジナル画像を 128 枚(256*256*8bit/1 枚)用いて処理する必要があるため，BRAM では容量が不足する。従って本システムでは，TDI は CPU 上のソフトウェアで処理し，その他の処理を FPGA ボードで処理することにする。

FPGA ボード内のラプラシアンフィルタ&2 値化モジュールとラベリングモジュールは，コントロールモジュールにより制御し，入力された画像データに対して画像処理を行う。

現在レジスタファイルの使用については 256*256 の全画面用レジスタファイルと 3 行分のレジスタファイルを用いて欠損検出を行っている。なお、本研究では、256*256 の全画面用レジスタファイルを 2 分割し欠損検出を行う。欠損検出システム全体の構成を図 17 に示す。

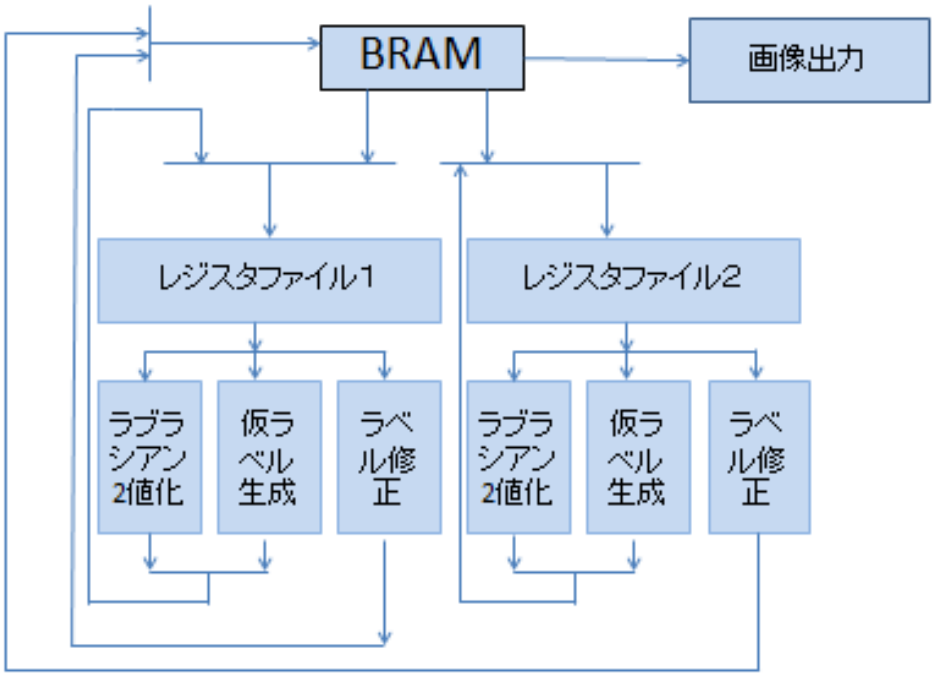


図 17: 並列化による欠損検出システム

4. 液晶用ガラス欠損検出システムの並列化

4.1 液晶用ガラス欠損検出システム全体の並列化

欠損検出システムの並列化とは、256*256 サイズのレジスタファイルを 2 分割して、ラプラシアンフィルタ/2 値化、ラベリングの処理を同時に行うことである。図 18 に 2 並列化による処理の流れを示す。

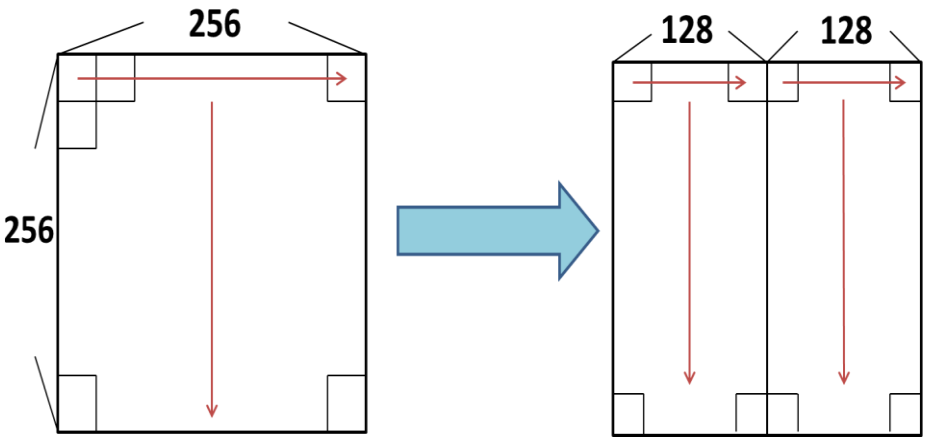


図 18: 2 並列化による処理

4.2 ラプラシアンフィルタ/2 値化の並列化

TDI 処理で生成されたノイズ除去後の画像データにラプラシアンフィルタをかけ，それを 2 値化して BRAM に保存する。まず，ノイズ除去後の画像データを，BRAM からレジスタファイルに転送する。次に，画像データから演算に必要な 3*3 画素分の画素データを取り出し，8 近傍マスクパターンとかけて 2 次微分処理を行う。2 次微分処理後の画素データを，閾値によって 2 値化処理し，BRAM へ保存する。この処理を 2 分割したレジスタファイルで同時に左端から右端に向かって、処理を行う。右端の処理を終えると、次の行の左端から処理を始める。この処理を繰り返して行う。図 19 にラプラシアンフィルタと 2 値化の動作を示す。

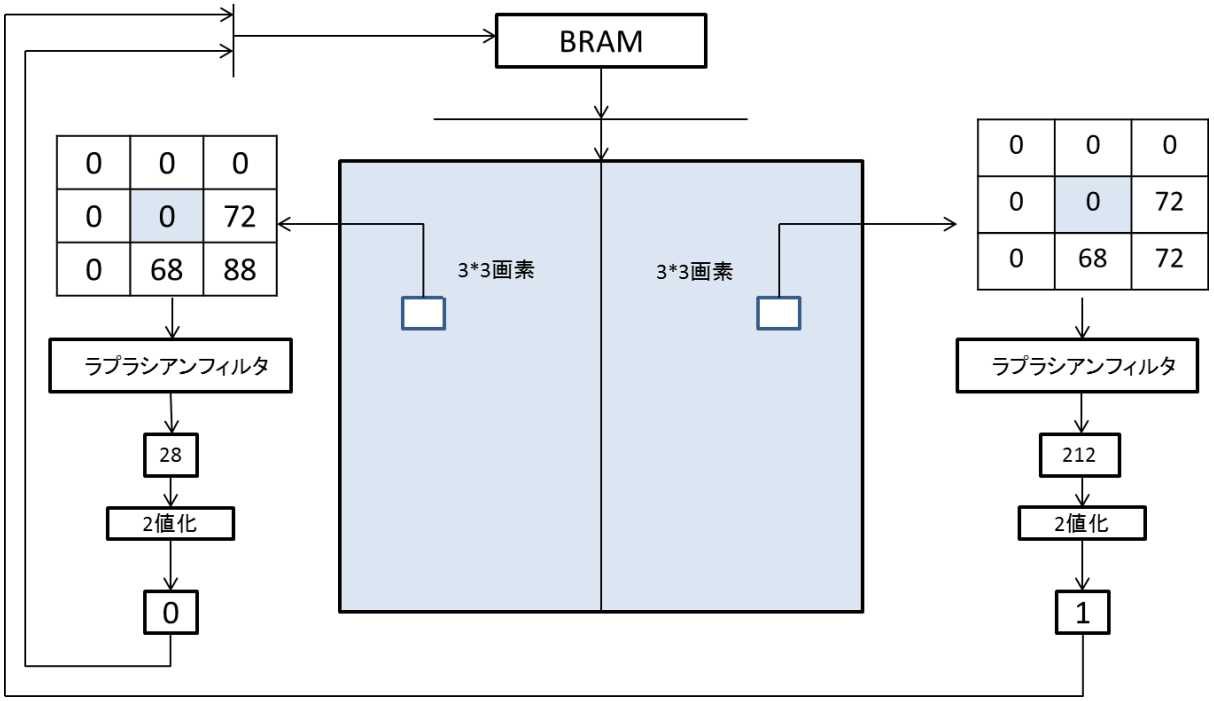


図 19: ラプラシアンフィルタ/2 値化の動作の流れ

4.3 ラベリングの並列化

2 値化処理後の画像データを BRAM からレジスタファイルに転送する際，欠損(画素値が 255)があるならば，レジスタの flag に 1 を入れる。この処理と並行して，レジスタファイルから注目画素と，参考にする 4 方向(左上，上，右上，左)の画素の 5 つを取り出す。

注目画素の flag が 1 の場合，周囲 4 画素の flag を調べ、周囲 4 画素にラベルがある場合は，左上，上，右上，左の優先順位順にラベルをコピーしてくる。周囲 4 画素にラベルがなければ新しいラベルをつけていく。注目画素の flag が 0 の場合は，そのままにする。BRAM への保存は，12bit の信号のうち下位 8bit を保存する。注目画素の flag が 1 で，周囲 4 画素中に 1 と 2 のラベルがあるが，優先順位により，注目画素には右上の 1 というラベルがつけられる。この時，周囲 4 画素しか見ないのは，注目画素より後ろの値にはまだラベルがつけられていないからである。

1 回のスキャンでは、連結しているのにもかかわらず異なったラベルがつく場合があるので，もう一度スキャンをしてラベルの補正を行っていく。

2 回目のスキャンは，仮ラベルがつけてある画像データを BRAM からレジスタファイルに転送し，注目画素と周囲 8 近傍のラベルの 9 つを取り出し，周囲に異なるラベルがある場合は，最も小さいラベルをつけて補正し，BRAM へ保存する。注目画素につけられているラベルは 2 であるが，周囲に 1 という最も小さいラベルがあるので，1 というラベルをコピーしてくる。2 回目のスキャンでは，1 回目のスキャンにより，全ての画素にラベルがつけられているので， 8 連結でラベリング処理を行っていく。ラベルの総数は，画素値と同じ 8bit(0 から 255)までとしてある。ラベリングの動作を図 20 に示す。

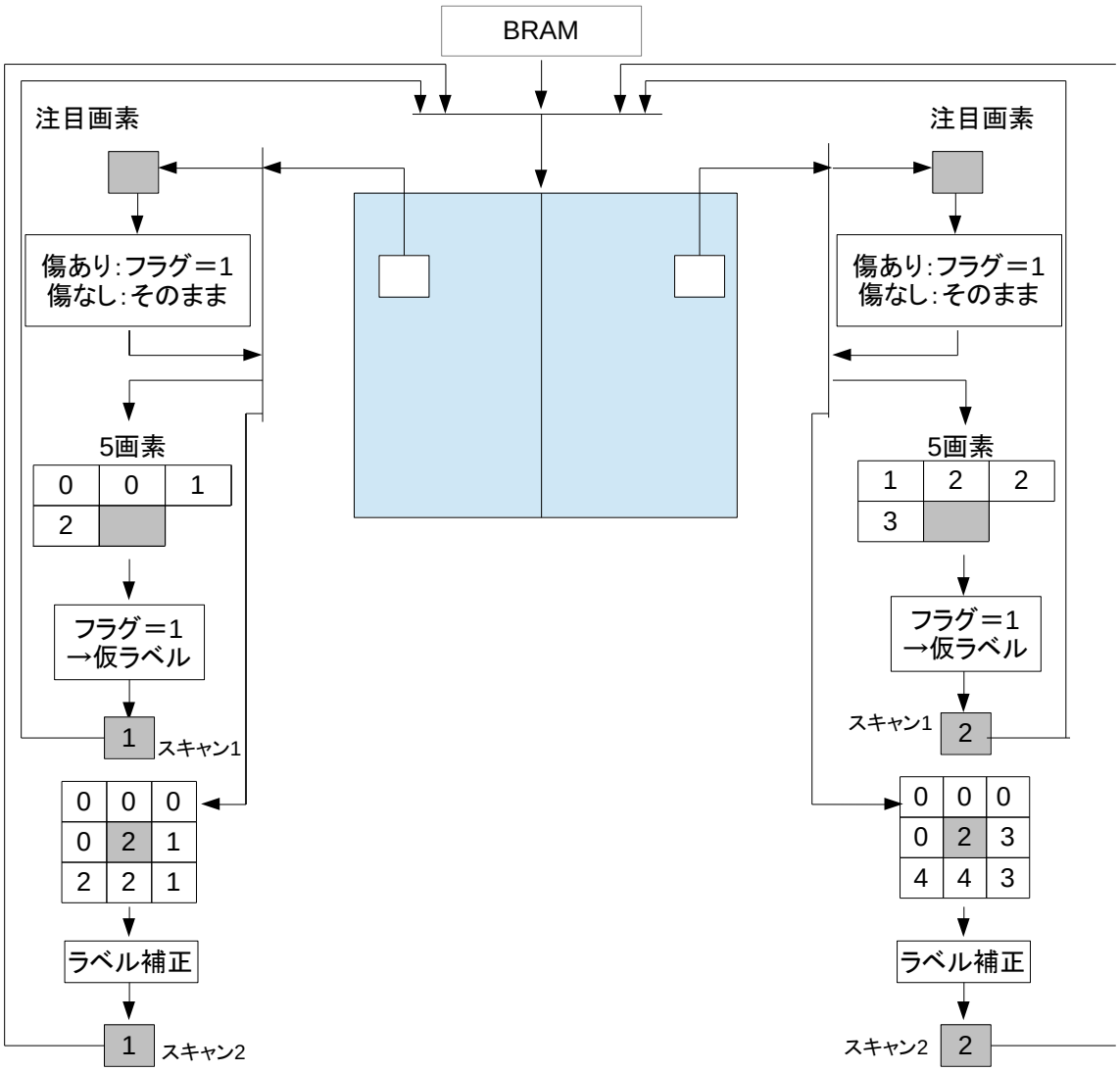


図 20:　ラベリングの動作の流れ

5. FPGA を用いた実験

5.1 実験条件

実験には TDI を用いて雑音除去をあらかじめ施したデータを使用し，データを FPGA 内の Block RAM に格納して画像処理を行う。

使用した液晶用ガラスの撮影画像のサイズは 256*256，1 画素 8bit のものを使用し，回路規模や処理時間を計測する。 処理時間は，画像ファイルからデータを取り出し，全データに対して画像処理を行い，メモリに格納するまでの時間を処理時間とした。

FPGA は Xilinx 社の Virtex5 ML507 評価ボードを用い，スライス数 11200，FF（フリップフロップ）回路数 44800，LUT 数 44800，Block RAM 容量 5328Kbit となっている。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。比較用に実験したソフトウェア処理の動作環境は Intel Core 2 Quad CPU Q9400 2.67GHz, 実装メモリ 4.00GB, OS は Windows7 Ultimate のものを使用した。

5.2 実験結果

設計した逐次処理の構成と 2 並列処理の構成のデザインの論理合成を行い，回路規模や動作周波数，処理時間を計測した。表 1 に逐次処理の全体の回路規模、表 2 に 2 並列処理の全体の回路規模を示す。

表 1：逐次処理の全体の回路規模

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	20	44800	0%
Number of Slice LUTs	121772	44800	271%
Number of fully used LUT-FF pairs	8	121784	0%
Number of bonded IOBs	44	640	6%
Number of Block RAM/FIFO	16	148	10%
Number of BUFG/BUFGCTRLs	1	32	3%

表 2：2 並列処理の全体の回路規模

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	31	44800	0%
Number of Slice LUTs	63458	44800	141%
Number of fully used LUT-FF pairs	8	63481	0%
Number of bonded IOBs	44	640	6%
Number of Block RAM/FIFO	16	148	10%
Number of BUFG/BUFGCTRLs	1	32	3%

逐次処理の LUT 数は、121772/44800 となった。これに対して、2 並列処理の LUT 数は約 50%減の 63458/44800 となった。

全画面用と 2 並列処理の各モジュールの回路規模を表 3、表 4 に示す。

表 3：逐次処理の各回路規模

	Registers	LUTs	LUT-FF pairs
RF_Read_Write	0	61847	0
Generate_ADDR	35	74	35
Generate_LAP	0	77	0
Generate_LAB	9	389	9
Generate_TLAB	0	148	84

表 4：2 並列処理の各回路規模

	Registers	LUTs	LUT-FF pairs
RF_Read_Write	0	60264	0
Generate_ADDR	19	47	19
Generate_LAP	0	77	0
Generate_LAB	9	389	9
Generate_TLAB	0	148	0

逐次処理と 2 並列処理の動作周波数とソフトウェア処理も含めた処理時間を表 5 に示す。

表 5：動作周波数と処理時間

	CPU	逐次処理	2 並列処理
Maximum Frequency	――	69.85MHz	57.42MHz
Processing Time／pixel	2276.2ns	14.31ns	17.417ns
Processing Time	149.17ms	4.69ms	2.85ms
Speed up ratio	1	31.80	52.34

動作周波数は、全画面用が 69.85MHz,2 並列処理が 57.42MHz という結果になり、処理時間に関しては、全画面用が 4.69ms,2 並列処理が 2.85ms という結果になった。処理時間はソフトウェア処理と比較して、全画面用が 31.8 倍で 2 並列処理が 52.34 倍という速度向上が得られた。また、2 並列処理は逐次処理と比較して、約 1.64 倍の速度向上が得られた。

5.3 考察

実験の結果より、逐次処理を 2 並列処理することで約 1.64 倍の速度向上を得られた。また、ハードウェア量に関しても約 50%削減することが出来た。さらに、速度向上に関しては、CPU との比較においては、逐次処理では 31.8 倍の速度向上があり、2 並列処理では 52.34 倍の速度向上が得られた。

本研究では、処理速度が期待されるラプラシアンフィルタ/2 値化、ラベリングを 2 並列処理することで速度向上を目指すことにした。今後の課題としては、本研究では 2 並列処理を行っていない TDI を並列化することで、処理速度の向上を検討したいと考えている。さらに、ラベリングの手法において、輪郭追跡を用いることで高速化も見込まれる。

また、本研究において 2 並列化処理を行う際に端の処理のタイミングの調整や、処理の仕方などに手間取る部分が多々あったので、改良する点が多いと思われる。以上のような内容を踏まえて、今後は並列処理を 4 並列処理、8 並列処理を行うことでさらなる画像処理の高速処理が見込まれる。

6. おわりに

本論文では、CPU を用いたソフトウェア処理による画像処理や，FPGA 上に実装させるためのモジュールの設計を行い、ハードウェアを用いた液晶用ガラス欠損を画像処理によって欠損を高速に検出する手法を提案した。

欠損検出システムの設計では、並列処理を用いてラプラシアンフィルタ、2 値化、ラベリングのハードウェア化を行い、ソフトウェア処理と比べて逐次処理の回路で 31.8 倍、2 並列処理の回路で 52.34 倍の速度向上を得ることができた。また、2 並列処理の回路設計では、逐次処理の回路と比べて約 50%のハードウェア量の削減を行った。

今後の課題として、液晶用ガラスのノイズを軽減させる TDI のハードウェア化と、考察で述べた処理モジュールのパイプライン化があげられる。FPGA の仕組みやデザインツールの使用方法などをさらに学び、目標とする欠損検出画像処理システム全体をハードウェア化することが課題である。

謝辞

本研究の機会を与えてくださり、ご指導をいただきました山崎勝弘教授に深く感謝いたします。本研究は、株式会社ケー･デーイーとの共同研究であり、研究に必要なデータを提供していただいた三宅淳司氏と西田洋隆氏をはじめ、様々な面で貴重な助言や励ましを下さった孟林助手や松山圭輔さん、研究室の皆様に深く感謝します。

参考文献

[1]松山圭輔: ”FPGA を用いた液晶用ガラス欠損検出システムの高速化, 立命館大学理工学研究科修士論文” 2013.

[2 松山圭輔・孟林・天井康夫・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, 2012.