時系列２値画像データに対する
即時ラベリング処理のための
局所的メモリを用いた回路構成の検討

指導教官

鈴木正國教授
北川章夫助教授
秋田純一助手
深山正幸助手

提出者

金沢大学工学部電気情報工学科
集積回路工学研究室

高松　直樹

提出日　平成１２年２月

目次

第1章　はじめに

画像の領域認識を行うためには逐次走査を行うのが一般的である。この画像処理をソフトウェアで行おうという研究は盛んに行われてきた。ソフトウェアでの画像処理においては膨大な時間がかかるのであるが近年の計算機の処理速度の発展に伴いかなりの時間のネックも解消されてきている。しかし、ロボットビジョンのように実時間処理を試みようとする場合、まだまだ処理しきれないというのが現在の段階である。

２値画像の領域を認識する方法としてラベリング処理という方法がある。このラベリング処理をハードウェアで行おうと研究は多数報告されているが各領域の重心や面積の計算というようなことを行う研究はあまり行われていない。また、ラベリング処理を行うためには一般に全画像の情報を保持しておくフレームメモリが必要となる。このようなメモリは処理回路に対して外付けとなってしまうため処理時間が増加してしまう。そこで本研究では各領域の認識と重心、面積計算を行うことを対象とし、ビデオ信号のような時系列のデータに対して効率の良いラベリングのアルゴリズムを用い、フレームメモリを必要としない処理回路の構成について検討をする。

第２章　アルゴリズムの検討

2-1　システム構成

本研究におけるラベリング処理システムの全体の構成を図1に示す。各ブロックについて説明をする。

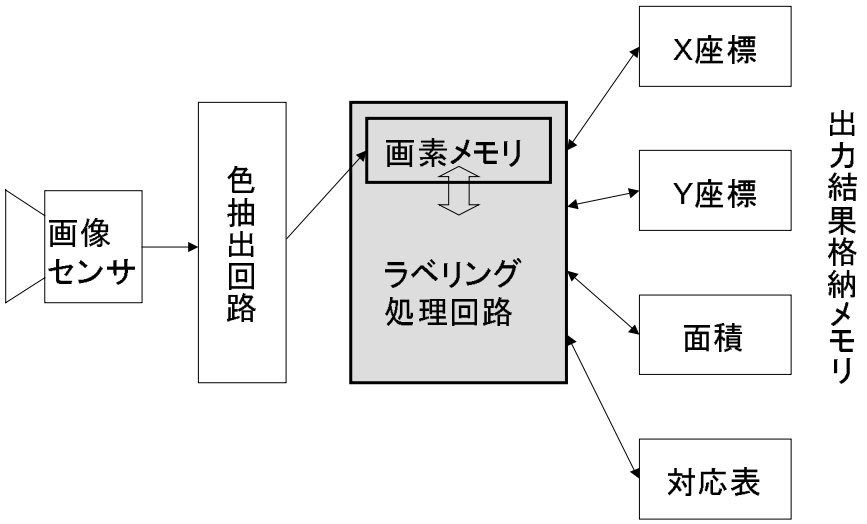


図１:本研究におけるラベリング処理システムの構成

・画像センサ・・・実画像を取り込んで色抽出回路へ送る。その際にデータは　　ディジタル化される。

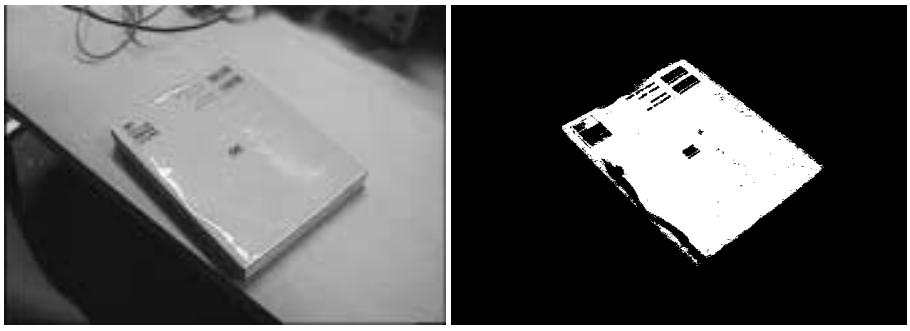
・色抽出回路・・・送られてきたデータから特殊な色のみを抽出して色分解し、各色ごとに２値化を行なう。複数色抽出できる。

・ラベリング・・・２値化されたデータを受け取り、実際にラベリング処理
　処理回路　　を行う。ここで各領域の重心のｘ座標、ｙ座標、面積の総和を計算してそれぞれの格納メモリへ送る。本研究においては、２値化されたデータ（画素）を保存する画素メモリは内蔵である。

・格納メモリ・・・ラベリング処理回路で得られたｘ座標、ｙ座標、面積の各総和をそれぞれのメモリへ格納する。対応表については後に説明する。

2-2　色抽出部

画像センサから取り込まれた画像は次の色抽出回路に入る。ここで、特定の色を抽出して各色について２値化を行う。そして、次のラベリング処理回路で処理をできる形を作る。昨年度の研究においては4色を想定していたが、本研究においては２値画像であるため1色と背景のみである。実画像の例を図2に示す。



実画像　　黄色抽出後の2値画像

図2：サンプルの実画像

2-3　ラベリングのアルゴリズム

2値画像中の異なる連結領域にそれぞれ固有の番号（ラベル）を割り当てる処理をラベリングと呼ぶ。ラベリングについて図3に示す。ラベリングを行うことによって固有の連結領域を認識し、それぞれの領域についての重心座標や面積を求めることができるのである。ラベリングのアルゴリズムについてはさまざまなものが考案されているが、ここでは昨年度に研究された1pass方式（境界探索法）と本研究で用いる2-pass方式について検討する。また、連結成分の定義として8近傍を用いることにする。

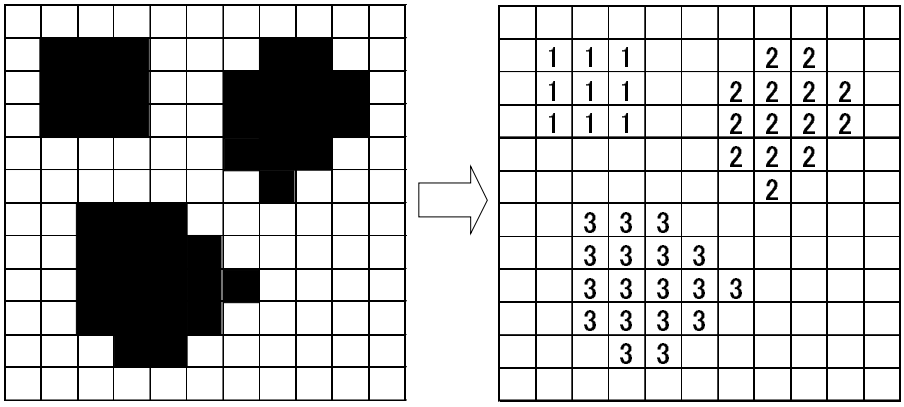


図3：ラベリング

2-3-1　1-pass方式

1-passのアルゴリズムは昨年度研究されたアルゴリズムである。これは画素を左上から右下へ向かって走査を行い、対象画素を見つけると隣にある連結領域を検出しながら境界探索を行いラベル付けしていくものである。下の図4に示してあるがこれは4近傍を例に挙げて行っている。そして元の場所へもどってくると、また左上から右下へ向かっていく逐次走査が始まる。これを繰り返しながら右下まで行くとラベリングか完了する。このアルゴリズムでは1度の境界探索でどのくらいの領域まで行われるのか分からないため、全画素を保持しておく外部メモリが必要となる。また、有値画素を見つけるたびに境界探索を行うため複雑な領域となる画像の場合にメモリへのアクセス回数が多くなるという欠点があるが、画像の走査を1回行うだけでラベリング処理を行うことができる。

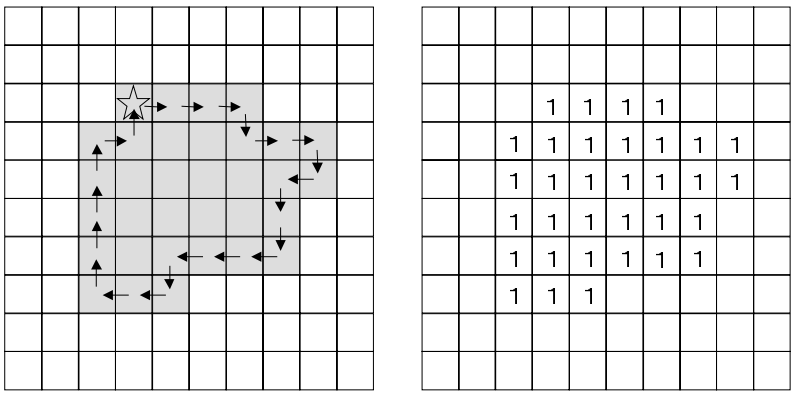


図4：1-pass方式（境界探索法）によるラベリング

2-3-2　2-pass方式

本研究で採用する2-pass方式はラベリングのアルゴリズムとしてよく知られている方式である。これは2回の走査でラベル付けを行うアルゴリズムである。ラベル付けの方法を図5に示す。例に示された２値画像を左上から右下へ向かって逐次走査を行っていくとする。まず図5(a)、(b)のようにラベル番号1、2と与えていく。ラベル付けをする際には図5にあるマスクを使う。☆がこれからラベルをつける場所であり灰色がすでに走査が終わっている領域である。この灰色の領域にラベル番号があればそれにしたがって☆にラベル番号を割り当て、もしもラベル番号がなければ新しいラベル番号を割り当てる。2行目以降このマスクに従い図5(c)、(d)のようにラベル付けされる。そして図5(e)ではマスクの灰色の領域に2種類のラベル番号がある。マスクの5マス中の有値画素については同一領域であるため1と2が同一領域であることがわかる。それを記録しておくために対応表というものを作り記録をしておく。そして2種以上のラベル番号があれば小さい方のラベル番号を割り当てるのである。このようにして1回目の走査が終了すると図5(f)となる。1回目の走査は仮のラベル付けであるため2回目の走査を行う。1回目の走査で記録した対応表に従ってラベル付けを行い直すことにより図5(g)のようにラベル付けが完了する。この方法では、走査回数は入力画像の内容に影響を受けにくいが、どのような画像に対しても2回の走査を行うため、走査する画素数は総画素数に比例して増加していくという傾向がある。

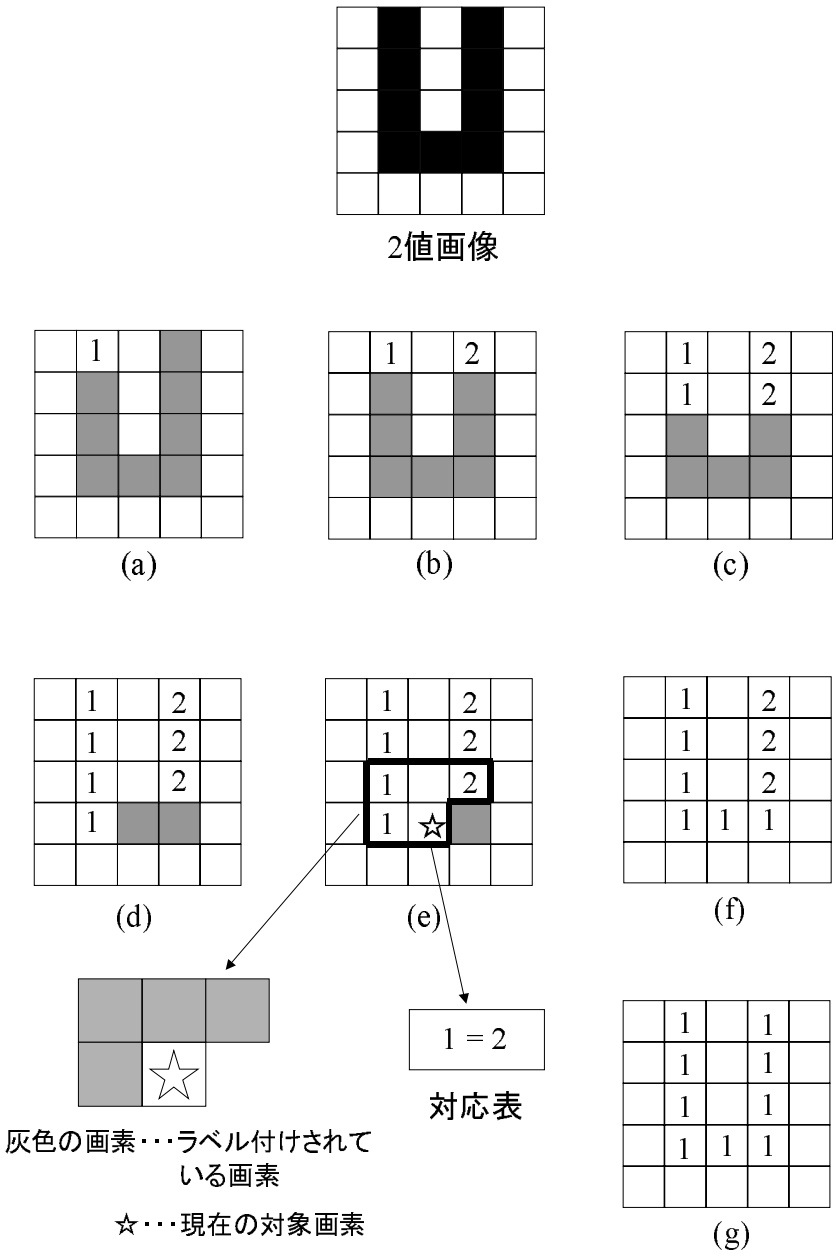


図5：2pass方式におけるラベル付け

しかし、本研究のように重心と面積の計算を行うだけであれば、2回目の画像の走査は必要ない。つまり1回目の走査で各領域の重心座標と面積の累積和を求めておき、対応表に従って走査の途中あるいは終了後に座標と面積の累積和を求め直すことで各連結領域の座標と面積が求められる。従ってこの方式では、1回の走査でラベリング処理が完了するためにビデオ信号のような時系列連続データを扱いやすく、また捜査の過程で参照するのが対象画素のある行と1つ前の行のみであるため、全画素を保持する大規模なフレームメモリが必要ないという利点がある。

2-4　シミュレーション

1回目の走査終了後に生じる最大ラベル数が、対応表や座標、面積のそれぞれのレジスタの必要数、および画素メモリに入るラベル番号（つまりbit数）になる。また、画素メモリへのアクセス数が多くなる可能性がある。これらのために最大ラベル数とアクセス回数を見積もるためにサンプル画像を用いてシミュレーションを行った。

2-4-1　サンプル画像

想定している画像サイズが640×480であるためこのサイズのサンプル画像を用意した。また、画像の内容は領域と背景しか用いない2値画像である。パラメータとして用いたのは、対象画素の割合を示すp、空間周波数fに対してパワースペクトルが$1/f^n$に比例することを表すnの２つである。わかりやすく述べるとnは画像の滑らかさを表すものである。サンプル画像の例を図6に示す。

p=0.1　　p=0.5　　p=0.9

n=0.2

p=0.1　　p=0.5　　p=0.9

n=1.0

p=0.1　　p=0.5　　p=0.9

n=1.8

図6：サンプル画像

2-4-2　画素メモリ

本研究においては全画素を保持する必要がない。2-3の2-pass方式のマスクを見るとわかるように、ラベル付けをする際に参照している場所は前の行の3画素と現在の対象画素とその前の画素の5画素である。つまり全画像に対して2行分の情報だけあれば良いのである。そこで640画素×2行分の画素格納メモリを用意することにする。アクセスポイントの移動、メモリの中身の移動方法について説明する。図7に示すようなメモリがある。まず2行目のメモリの内容を1行目のメモリへ移動する。それから画像から次の行の640画素を2行目のメモリへ代入する。そして2行目の対象画素を左から右へ逐次走査することでラベル付けを行っていく。一番右までの走査が終わるとまた2行目の画素値を1行目に代入する。これを繰り返すのである。

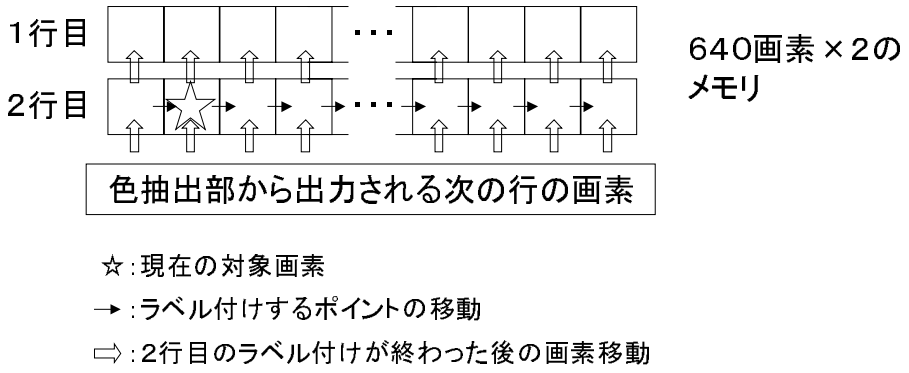


図7：画素メモリ

2-4-3　最大ラベル数についてのシミュレーション結果

シミュレーション結果を図8に示す。z 軸が最大ラベル数である。最大ラベル数がかなり大きくなっているところがあるが、それについては有値画素が1つもしくは少ししかない領域がとびとびで大量に存在しそれぞれの領域についてラベル番号が割り振られたために生じたと思われる。実画像においてはnは1.5～2.0であることが知られている。従ってかなり滑らかな画像であるため有値画素が1つしかない画素というのはnoiseと考えられる。サンプル画像についてはnの値が0.0から2.0まで0.2おきであるため1.4のところと1.6のところで判断した。nが1.6の時、最大ラベル数は512つまり9bitであり、nが1.4の時、最大ラベル数は1024つまり10bitであった。従って1024であればよいということになる。しかし、行ったのが図7で示したサンプル画像であるため実際の画像についてもシミュレーションを行った。使用したのは図2の画像であるが最大ラベル数は50となり1024以内でおさまった。したがって1024とする。

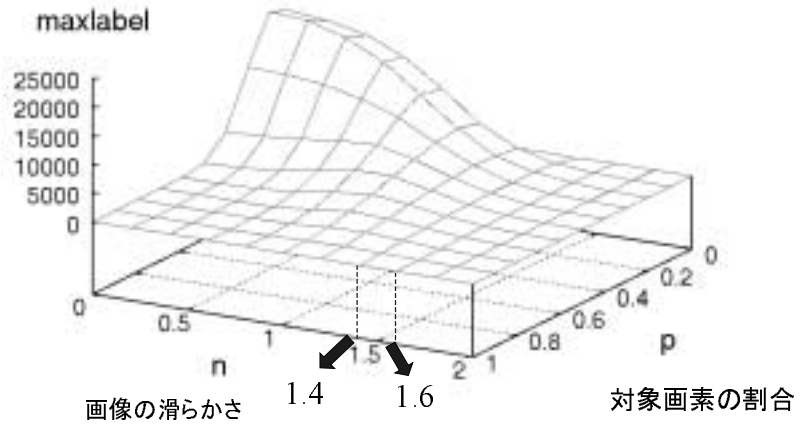


図8：最大ラベル数のシミュレーション結果

2-4-4　アクセス回数についてのシミュレーション結果と考察

C言語によるシミュレーションを行っていたが、周りの4画素へアクセスするときに逐次数値比較を行っていた。図9に示すようにまず3を呼び出してからその後に2と比較して最終的なラベル番号を決定していた。そしてシミュレーション結果は全画素数の約1.2倍となった。しかし、アクセス回数についてはどの画素においても1回ですむということがわかった。その方法は、図10にしめすように周りの4画素に同時にアクセスをして数値比較をし、最も小さいラベル番号がなにかを決定して対象画素にラベル番号を割り当てるというものである。これにより、入力画像が時系列データであっても処理が行えるため全画素を保持する必要がない理由のひとつともなる。

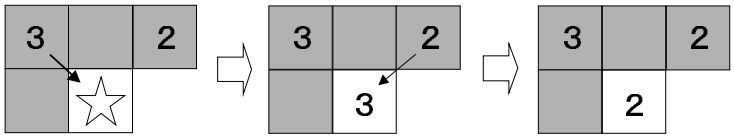


図9：逐次比較によるラベル付け

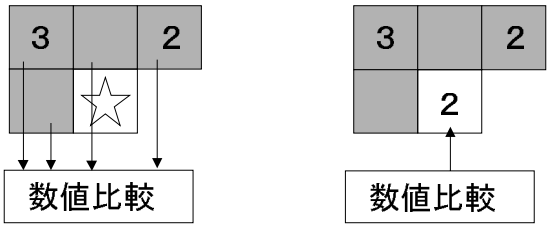


図10：数値比較によるラベル付け

第3章　2pass方式による
ラベリング処理回路の設計

3-1　設計仕様

3-1-1　仕様詳細

前章において入力画像を640×480画素と想定し、画素を保持するメモリの大きさを640画素×2とした。この条件において最大ラベル数を検討した結果1024となった。しかし、改良する点が見つかったため以下のような特徴となった。

・ラベル数・・・シミュレーション結果では1024と考えていたが回路規模を小さくするため、またいくつかのサンプルの実画像においても512以内でおさまることがわかったため512（9bit）とする。

・画素メモリ・・・これは外部メモリとしてではなく内部メモリ、つまりチップ上に置くことにした。全章で9bit格納できるレジスタが640画素分×2行分必要であると想定したが、あまりに大きすぎるために改良して640+2画素分とした。（後に説明）

・色情報・・・2値画像を検討しているため、1色と背景のみである。理由として、色情報を増やすと最大ラベル数が増えてしまうため画素メモリ、格納するレジスタ数などが大きくなってしまうからである。

・重心座標・・・重心座標については、各領域のｘ座標の累積和、ｙ座標の累積和をその領域の面積の累積和で割ることによって得られる。しかし、除算回路を含めると回路規模が大きくなりすぎてしまうため除算回路は作成しない。累積和のみを出力する。ｘ座標の累積和は27bit、ｙ座標の累積和は26bit、面積の累積和は19bitを用意した。

・対応表・・・周りの4画素中に2種以上のラベル番号があれば、それらが同じ領域であるということを記録しておくメモリである。9bit格納できるレジスタを512用意する。その中身の初期値はレジスタのアドレス番号と同じ値を与える。

・オーバーフロー・・・最大ラベル数が512を越えた場合、またｘ座標、ｙ座標、面積の各総和のいずれかがあふれた場合に出力するものとする。

3-1-2　画素メモリ

画素格納メモリであるが、全章で提案したように640画素(9bit)×2行分と考えていたが規模がかなり大きくなるためメモリの構造を変えてみた。ここでは画素メモリを640画素分と直前の画素分と現在の対象画素分のみ、つまり640+2画素とした。このようなメモリの構成にし以下の図11の動作を行うようにすることで全章で考えた画素メモリと同様の動作をさせることができる。

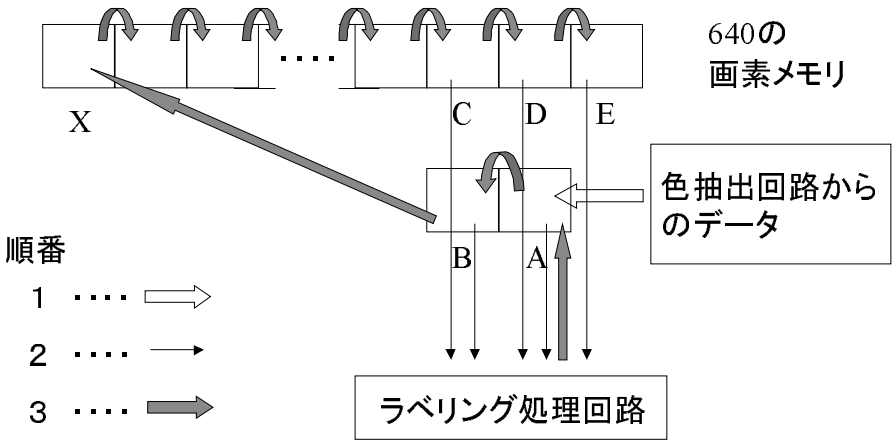


図11：画素格納メモリの構成

図11の画素値の移動を説明しておく。まず色抽出回路から出力されるデータをAに代入する。そしてラベル付けするためにA、B、C、D、Eのメモリの内

容が参照される。ラベリング処理されラベル値が返されるとそれをAに代入する。その後、640のメモリの中身を右へシフトしBの値をXへ、Aの値をBへ移動する。以上のような流れが1画素のラベリング処理中に起こる動作である。従って、640+2画素分の内部メモリを使い以上のような動作をさせるような回路を設計する。

3-1-3　ユニット

ラベリング処理回路において、機能別のユニットを以下の図12に示す。ユニットについては5つを考えている。

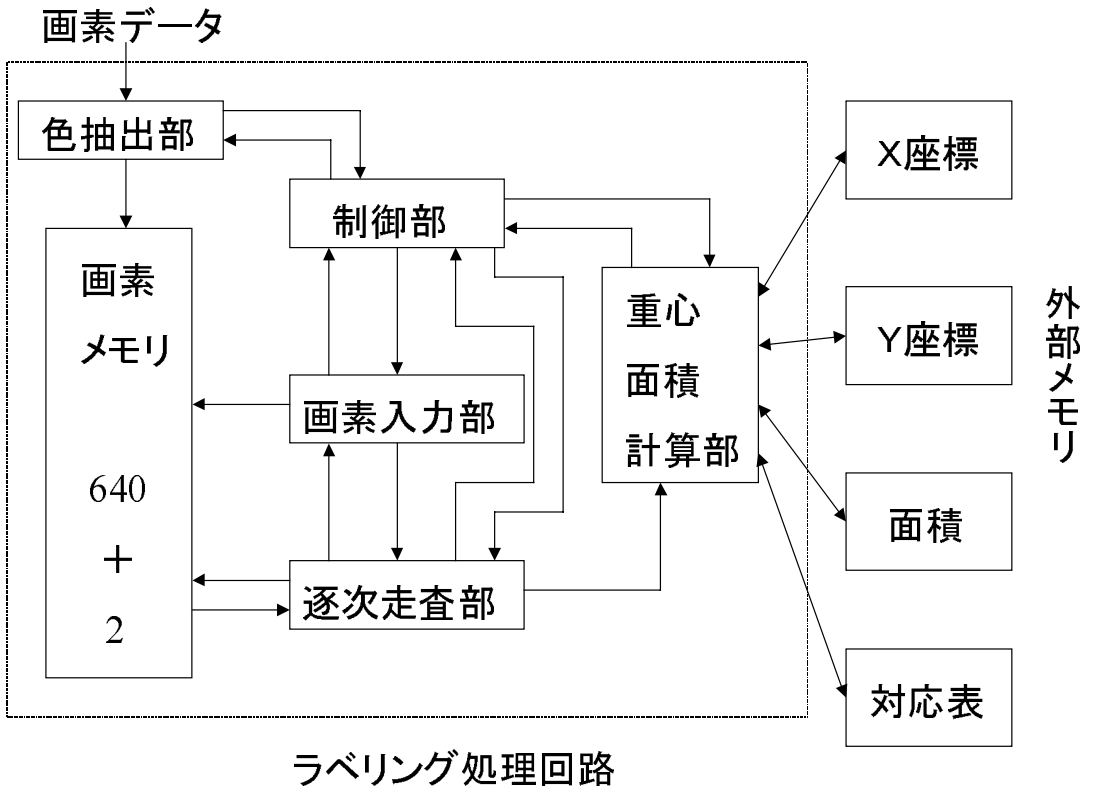


図12 ： ラベリング処理回路のブロック図

・制御部・・・クロック信号やリセット信号を受け取り、各ユニットへ仕事させるか停止させるかを制御する。また、処理実行中の場合にBUSY信号を出したりオーバーフローしたらOVER信号を出力する。

・色抽出部・・・画像センサから画素データを受け取り、色別に分けて2値化する。1色を抽出する回路をつくりそれを並列にならべることで複数色の2値データを得ることができる。本研究では最大ラベル数の増加に伴うピン数の増加、画素メモリの増大を防ぐため1色と背景のみで行った。

・画素入力部・・・色抽出回路からの２値データを受けとったり、全章で説明したマスクに該当する部分の出力、決定したラベル番号の書き込み、画素値のシフト移動などを行う。

・逐次走査部・・・画素入力部より出力される画素に対してラベリング処理を行う。そして決定されるラベル番号を画素メモリへ書き込む。また、２種以上のラベル番号があればその値をすべて記憶する。

・重心面積計算部・・・逐次走査部でラベリング処理が行われている画素についてのｘ座標、ｙ座標、面積値として画素値1を各累積メモリへ出力する。また、逐次走査部で同値関係が見つかったら対応表へ出力する。

3-1-4　各ユニットの内部ステート

今回設計して各ユニットについて、そのほとんどがステートマシンとなっている。各ユニットについての内部ステートを以下に示す。

1. 制御部
・RESET信号・・・初期化を行う
・DA信号・・・色抽出部および画素入力部での作業命令
・LA信号・・・逐次操作部つまりラベリング処理の作業命令
・CA信号・・・重心面積計算部の作業命令

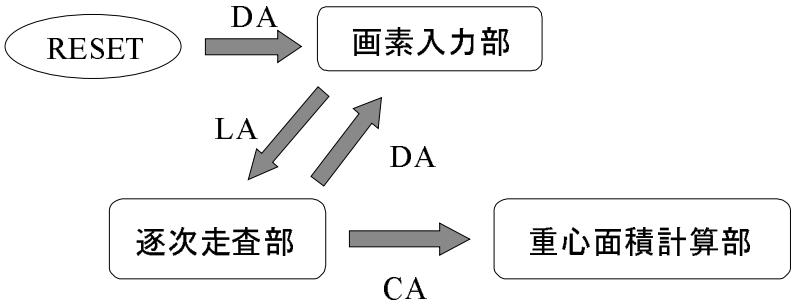


図13：制御部の状態図

流れとしてはDA信号出力→LA信号出力→CA信号出力だが、重心面積計算を行う間に次の画素の入力を行うため、DA信号を出力しその後LA信号を出力する。そのため、DAとCA、またLAとDAが同時に出力されるケースがある。また、画素入力部→逐次走査部という過程でかかる時間よりも重心面積計算部でかかる時間の方が若干多いため、逐次走査部で少し待機状態となる。

2. 色抽出部、画素入力部

この二つのブロックを含めて画素入力部と考えることにする。ここでは前節で説明した画素メモリの中での画素値の移動を行う。

・RESET信号・・・初期化を行う
・IN・・・色抽出回路からの画素値を入力
・OUT・・・全章で説明したマスクに相当する画素値を出力
・SHIFT・・・640のメモリの画素値をシフトする
・WAIT・・・逐次処理においてラベル値が決定するまでの待機状態

状態遷移を図14に示す。

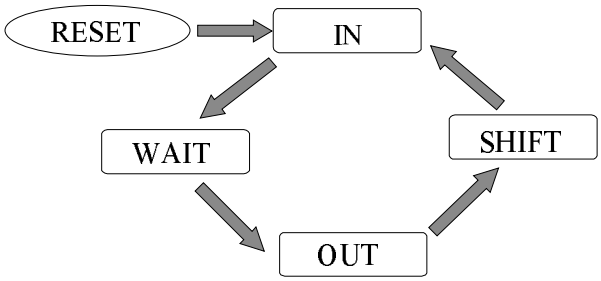


図14： 画素入力部

3．逐次走査部

・RESET信号・・・初期化を行う

・LABEL1・・・ラベルの付いている4画素についてどれが最も小さいラベル番号か、2種以上のラベル番号があるかを調べる。

・LABEL2・・・対象画素が有値画素であればLABEL1で決定したラベル番号を、模しなければ新しいラベル番号を決定する。

・WAIT・・・次の画素が入力されるまでの間の待機状態

状態遷移を図15に示す。

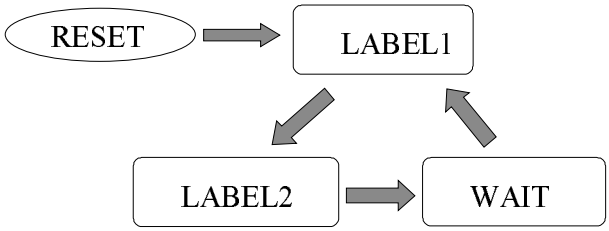


図15：逐次走査部（ラベリング処理部）

4. 重心面積計算部

PART1　逐次走査しながらの段階

・RESET信号・・・初期化を行う
・READ1、READ2、READ3・・・重心ｘ座標、ｙ座標、面積の各メモリをreadする。また、同値関係が存在すれば対応表メモリのデータをreadする。
・WRITE1、WRITE2、WRITE3・・・ラベル付けされた画素のｘ座標、ｙ座標、面積値を画素1として加算してから重心ｘ座標、ｙ座標、面積の各メモリをwriteする。またread時に同値関係があることが分かれば対応表をwriteする。
このPART1では画素に対するラベル付けを行いながらx座標、y座標、面積値をそれぞれ加算していく段階である。状態遷移を図16に示す。

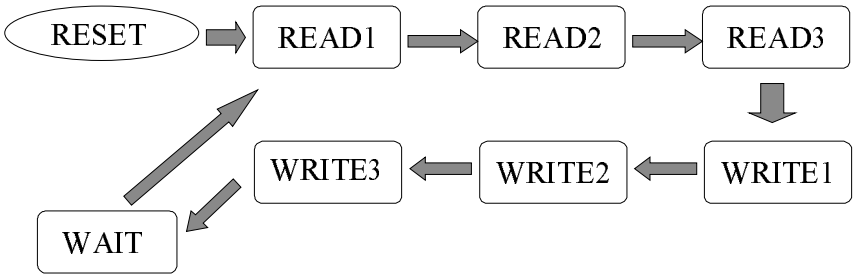


図16：重心面積計算部PART1

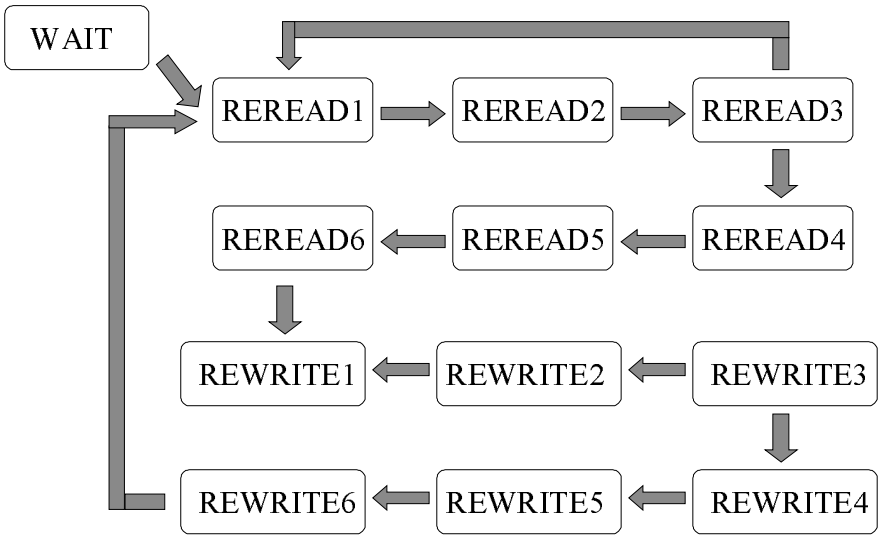
PART2　逐次走査が終わり、対応表に従ってそれぞれの累積和を求め直す段階

・REREAD1、REREAD2、REREAD3・・・対応表メモリをreadする。これはラベリング処理により決定される最大ラベル数のアドレスから呼び出し、その後１デクリメントする。また、x座標、y座標、面積値をreadする。
・REREAD4、REREAD5、REREAD6・・・対応表メモリのアドレスとそのデータ値が一致していればREREAD1へもどる。一致していなければ呼び出したデータのｘ座標、ｙ座標、面積それぞれの累積メモリをreadする。

・REWRITE1、REWRITE2、REWRITE3・・・REREADフェーズで呼び出した値を足し算してそのデータをREREAD4～REREAD6で使用したアドレスへwriteする。

・REWRITE4、REWRITE5、REWRITE6・・・REREAD1～REREAD3で使用したアドレスのｘ座標、ｙ座標、面積それぞれの累積メモリに対して0をwriteする。

つまりやっていることは例えば、3と6が同値であるとすればアドレス6のそれぞれの累積メモリの値をアドレス3の累積メモリの値へ加算しその後にアドレス6のそれぞれの累積メモリへ0を与え、書き換え終了を示すのである。状態遷移を図17に示す。



図？： 重心面積計算部その2

このように重心面積計算部では、read、writeそれぞれにおいて3ステート分用意をした。外部メモリとアクセスしているために時間が必要だからである。

3-1-5　ポート

ラベリング処理回路における入出力ポートの一覧を図18に示す。

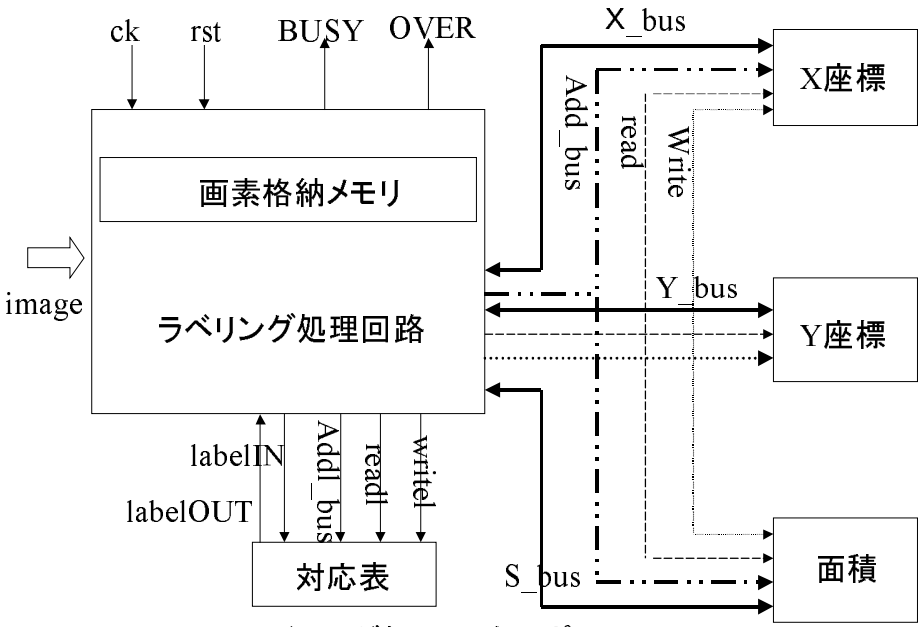


図18：ラベリング処理回路のポート

入力信号

・CK・・・メインクロック　1bit
・RST・・・リセット信号。1のとき初期化。　1bit
・image・・・取り込まれる画素データ。RGB構成、24bit

出力信号

・read・・・ｘ座標、ｙ座標、面積メモリをreadするとき1　1bit
・readl・・・対応表をreadするとき1　1bit
・write・・・ｘ座標、ｙ座標、面積メモリをwriteするとき1　1bit
・writel・・・対応表をwriteするとき1　1bit
・BUSY・・・処理回路が実行中の時1　1bit
・OVER・・・ｘ座標、ｙ座標、面積のいずれかがデータバスに入りきらなくなった時、また最大ラベル数が512を越えたとき1　1bit
・labelIN・・・対応表からreadした値を呼び出すためのデータバス　9bit
・labelOUT・・・対応表へwriteする時の値を送るデータバス　9bit

・Add_bus・・・x座標、y座標、面積メモリのアドレスを指定するバス　9bit
・Addl_bus・・・対応表のアドレスを指定するバス　9bit

以下の３つは双方向バスである
・X_bus・・・重心x座標累積メモリとをつなぐデータバス　27bit
・Y_bus・・・重心y座標累積メモリとをつなぐデータバス　26bit
・S_bus・・・面積累積メモリとをつなぐデータバス　19bit

合計のピン数は140である。

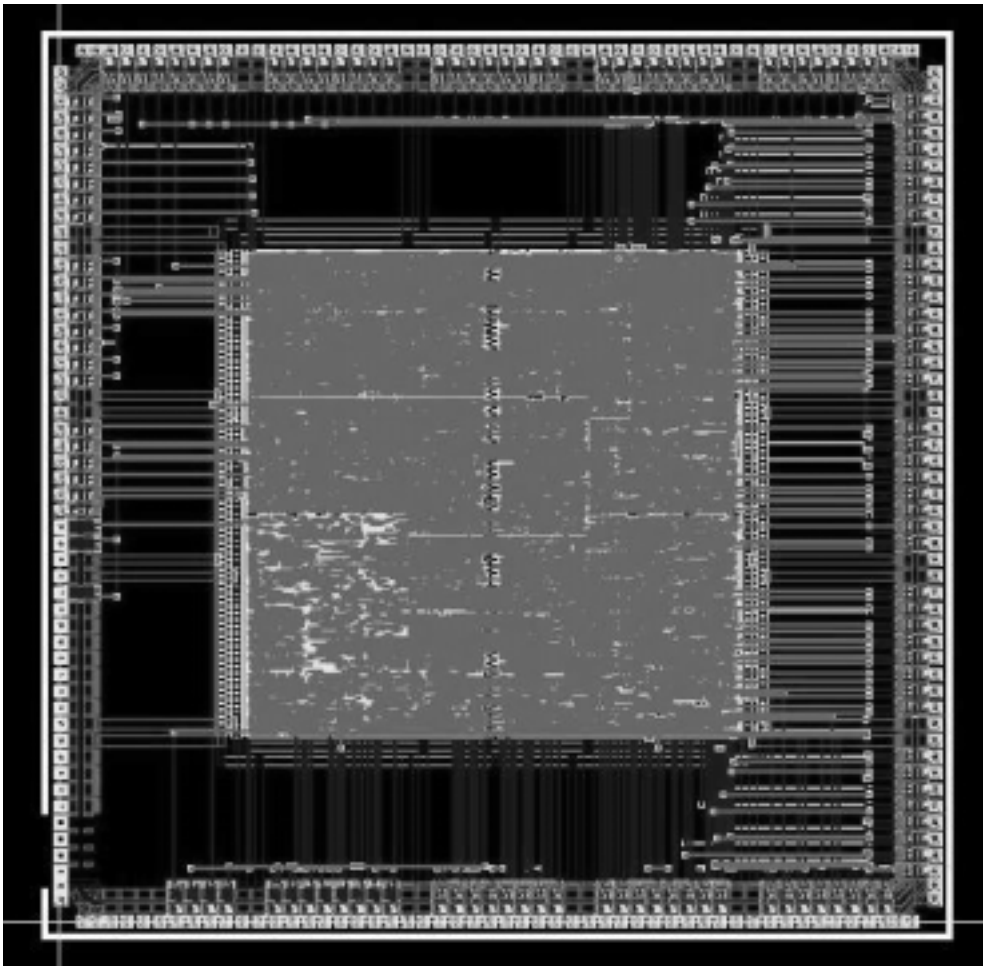
3-2　設計結果

3-2-1　論理合成

3-1で述べたような設計仕様として回路構成を行った。設計言語としてVerilog-HDLを用い回路の記述を行いシミュレーションにより動作確認がとれたため、その後にSynopsys社の論理合成ツールであるDesign Analyzerを使用して論理合成を行った。

3-2-2　配置配線

論理合成が終わり、動作確認がとれたのでチップのレイアウトを行った。そこでAvant!社の配置配線ツールであるApolloを使用して配置配線を行い、終了後にDesign Rule Checkを行うことで設計ルールの違反がないことを確認した。チップのレイアウトを図19に示す。



ROHM CMOS 0.6μm プロセス　3層金属メタル　9mm角チップ
セル数　20,829　　トランジスタ数　217,022

図19：チップのレイアウト

3-3　シミュレーションおよび考察

Verilog-HDLで記述した回路を論理合成ツールによりゲートレベル記述にした回路においてシミュレーションを行った。ゲートレベル記述にすることによりゲート遅延を考慮に入れることができる。本研究では時系列のデータを扱うのものとし、1画素ごとの処理時間もすべて同じであると考えているためにどのような画像に対しても同じ時間で処理が終了する。そこで、図2で示したサンプルの実画像においてシミュレーションを行った。処理時間は最速の場合約15msであった。先ほどどのような画像に対しても同じ処理時間であると言ったが、対応表に従ってx座標、y座標、面積の累積和を求め直す時間については画像ごとにかわる。この処理時間は3-1-4の重心面積計算部のPART2の部分に相当し処理時間をαとする。PART2ではステートを12用意しているためクロックサイクルにおいても12サイクル必要とする。そこで例えばクロックサイクルを10ns(100MHz)とした場合、10ns×12ステート×対応表のレジスタ数512により約60μsとなる。この値は、画像の処理時間である数十msに比べると十分に小さい値と考えられる。従って、どのような画像においても最速で約15msで処理を行うことができる。ビデオレートは1画像につき約33msであるため、本研究で想定しているビデオ信号のような時系列データに対して十分有効である。

第４章　まとめ

画像の領域認識に対してビデオ信号のような時系列の2値画像を想定し、このようなデータを保持するのに全画像を保持するフレームメモリを用いるのではなく、一部のみを保持する小規模なメモリを使うことで高速な画像処理をすることができるようなラベリング処理回路の構成を検討した。

本研究で用いたラベリングのアルゴリズムは2pass方式であった。そしてC言語によるシミュレーションを行うことで動作確認を行い、また設計仕様も決定した。

そして設計においては、VDECのチップ試作サービスを利用してラベリング処理回路の設計を行った。そしてゲートレベル記述での回路動作を確認し処理時間を見積もることで、画像処理を高速にできるということが確認できた。従って本研究で設計した回路は画像処理において有効であるといえる。

本研究は1色と背景の2値画像ということで行っていたが、色情報を加え数色にも対応できる処理回路もできると思われる。ただし回路規模が大きくなる恐れがあるため、少し工夫が必要になると思われる。

謝辞

本研究を行うにあたり、多くの方からの御指導、御協力をいただきました。感謝の意を表したいと思います。

指導教官として御指導を頂きました金沢大学集積回路工学研究室教授　故鈴木正國教授に深く感謝するとともに、心からご冥福をお祈りします。忙しい中、北川章夫助教授にはさまざまな分野において御指導をいただいたことを深く感謝しております。さらに研究を進めるにあたって多くの御指導、御助言を頂いた秋田純一助手に深く感謝しております。1月からということで短い期間でしたが、設計に関して御指導、御助言を頂いた深山正幸助手に深く感謝しております。研究生活を行うにあたりさまざまな面でご協力いただいた集積回路工学研究室卒の中山和也助手に深く感謝しております。半年間でしたが御指導いただいたDr.Nandana Fernandoさんに深く感謝しております。また、VLSI設計室の設備や管理作業において多くの御協力を頂いた柿本芳雄文部技官に深く感謝しております。

研究、および学生生活において有意義で楽しいものにしてくださった大学院修士課程の高瀬信二氏、小川明宏氏、中橋憲彦氏、早川史人氏、藤井直樹氏、渡辺晃氏、数馬晋吾氏、藤田隼人氏、村上崇氏、今井豊氏、水野浩樹氏に心から感謝の意を表します。そして、この一年間苦楽を共にし、充実した日々を共に過ごした卒研生の笠井稔彦君、佐々木勝光君、大門慎治君、辻川隆俊君、遠山治君、中村公亮君、蓮達弘君、水木誠君に感謝します。

どうもありがとうございました。

参考文献

鳥脇純一郎： "画像理解のためのディジタル画像処理(II)", 昭晃堂 (1998)

CQ出版社：Verilog-HDLによるトップダウン設計

服部哲郎：パイプライン処理方式における領域ラベリングに関する研究

翔泳社：独習C