静岡大学 博士論文

1 次元化 Morphological 画像処理アーキテクチャ

1996 年 2 月

静岡大学大学院電子科学研究科
電子応用工学専攻

小島 昭二

論文要旨

数学的 Morphology は画像処理における論理的演算の基礎を提供するという意味において重要であり，ノイズ除去，輪郭の平滑化，形状記述，テクスチャ解析，フラクタル次元の解析等，従来より様々な応用が考え出されている．Morphology 処理は集合理論にその基礎を置くため，フーリエやコンボリューションなどによるフィルタリングとは，また異った種類の論理の体系を持っている．Morphology は，画像情報に対するプローブとも言うべき「構造要素」と画像との間の演算という形で定義されている．Morphology を用いる事により，様々な画像処理を式の形で記述できることに加え，演算の結果得られた画像の持つ性質があらかじめわかると言った論理的な見通しの良さが保証される．

この Morphology 演算を高速に実行するため，従来より専用ハードウエアが研究されてきた．一例として Cytocomputer では 3×3 マスクによる画像処理を高速に実行する事ができる．しかしこの方法で大きな構造要素（例えば 16×16 画素）を実現しようとするとハードウエアの複雑化がさけられず，また Gray-scale Morphology にこのアーキテクチャを適用した場合，n bit の Gray-scale の深さに対し 2^n に比例する量のデバイスが必要になってしまい，実現が困難となる．また，Cytocomputer 型アーキテクチャのために構造要素の分解手法が提案されているが，これらの手法を用いる場合，目的の構造要素の形状を記述するためにしばしば多数の縦列処理が必要になったり，構造要素が必ず凸型でなければいけないといった制限が生ずる．そこで本研究では高速な Morphology 処理を実現するための新たなアーキテクチャを提案する．このアーキテクチャの考え方において重要な点は二つあり，一つは Morphology 処理を 1 次元で行う事であり，もう一つは画像データの縦・横を変換して 2 回走査を行うことである．この方法により高速な計算と，ハードウエアの規模の大幅な削減が可能となった．そして本研究では目的ごとに三つのアーキテクチャを提案し，それぞれにおいて試作機を製作した．

第 1 番目のアーキテクチャは Binary Morphology の実時間処理装置である．そこではまず Morphology のアルゴリズムを 1 次元化，並列化する方法を論じ，次にそのアルゴリズムを用いて構造要素の大きさに制限がなく，ビデオ信号を実時

間で処理することを可能とするハードウェアを示す．また応用としてBright eye（赤外線の網膜反射像）の画像補正にこの装置を組み込んだ例を示す．この例ではビデオ信号として取り込んだBright eyeの輪郭の明確化とノイズ除去を実時間で実行することを目的としている．

第２番目はGray-scale Morphology のための計算機であり，これは先に述べたBinary アーキテクチャのGray-scale 版と位置付けることができるが，その実現方法はGray-scale Morphology にあわせて大幅に変更されたものとなっている．このアーキテクチャの心臓部ともいえる「１次元Dilation プロセッサ」は縦・横に２回走査された画像に対して１次元方向にDilation 演算を施して出力するパイプライン式のプロセッサである．この１次元Dilation プロセッサの有利性に関して，Cytocomputer 型アーキテクチャとの比較において，回路規模や回路のスピードといった点で詳しく比較する．また１次元化アーキテクチャによって構造要素の形状が受ける制限や，その回避方法に関しても議論を行う．

第３番目のアーキテクチャは計算幾何学に適用するための機能を持ったGray-scale Morphology 計算機である．これは第２番目のアーキテクチャを含む形で発展させたものであり，Dilation プロセッサの部分にテーブル変換器（RAMを含むロジック）を組み込むことにより，図形の距離変換に適用範囲を広げたものである．またここでは，画素の輝度を表すGray-scale 方向のbit 数と誤差の関係，さらに誤差の影響を少なくするための nose 型構造要素を提案する．さらにこのアーキテクチャのもう一つの特長として，ボロノイ図を求める際に必要となる母点の座標を保持するために付加したロジックがある．この部分の論理はMorphologyの演算とは建前のうえでは独立したものであるが，Morphology 演算を計算幾何学的なアプリケーションに適用しようとした場合，ぜひ必要な機能となる．またこのようなアプリケーションに使用する場合Gray-scale のbit 数をどれだけ取る必要があるかについても議論を行う．

本論文を総括する部分では，本論文に述べた三つのアーキテクチャがどのような場面で用いられるべきかといった役割に関して述べる．また固定構造要素の１次元Gray-scale アーキテクチャとテーブル変換器を用いたアーキテクチャとで，適用可能な構造要素の形状の種類がどのように決まるかを詳しく述べる．

目 次

Chapter 1　導入

1.1　はじめに

本論文では画像の Morphology 演算を高速に実行するためのアルゴリズムと，そのハードウェアへのインプリメンテーションについて扱う．本論文は主に３つの主題からなっており，第一番目が Binary Morphology のための手法，第二番目が Gray-scale Morphology のための手法，そして第三番めは第二番目の手法をさらに発展させたものとなっている．本論文に述べる３つの主題に関して共通しているのは，画像処理を１次元化することによる高速化手法に関して論じていることである．本研究の特徴は，アルゴリズムやアーキテクチャを示すことに加え，実際に試作機を製作し，動作の検証を行っている点にあるといえよう．またこれらの試作機はある程度実用になるものをめざして作ったものであり，マイクロプロセッサとメモリからなる直列計算機では得られない計算速度を引き出すことに成功している．本論文の総括の章では，ここで述べる各アーキテクチャの位置付けや適用範囲について詳しく述べる．

1.2　Morphology に関する経緯

信号処理の新しい手法としての数学的 Morphology 演算は特に画像処理の分野において従来より様々な応用が考え出されており，ノイズ除去，輪郭の平滑化，形状記述，テクスチャ解析，フラクタル次元の解析等その有用性は非常に高い．Morphology 処理は集合理論にその基礎を置くため，フーリエやコンボリューションによるフィルタリングとは，また異った種類の論理の体系を持っているが，画像情報に対するプローブとも言うべき構造要素（Structuring element）という概念を用いることにより，論理体系として整理されている．

Mathematical Morphology は，集合理論に基づく演算による図形や信号の操作である，と言うことができる．この理論で用いる基本的な演算手法は２０世紀初頭の数学者，Minkowski の考えた集合どうしの演算である Minkowski 和・差にその起源を求めることができ，Morphology の最も基本的な部分はこの Minkowski の論理を取り込む形で成り立っている．1960 年代に入り計算機による画像処理が行われるようになると，画像に含まれる画素どうしの近傍演算である，膨張・収縮演算を利用した，画像処理の手法が考え出された．特に画像から線素を抽出して図形の持つトポロジー的な性質を解析し，文字や図形の認識に応用する手法がこのころの発展し，大きな成果となっている．

Morphology 画像処理はこれまでに考えられてきた，集合理論を用いた画像処理手法を総括的に取り扱う理論であり，J.Serra による 1982 年の著作”Image Analysis & Mathematical Morphology” [1] によってそのパラダイムがほぼ完成された形で示されたとされる．この Morphology 画像処理は，類似の手法を単に寄せ集めて名付けた総称というものではなく，その概念は数学的な基盤を持つ体系化された理論となっている．

一方，Morphology を含めた画像処理ハードウェアに関する研究も盛んであり，とくに 1985 年 IEEE Computer Society 主催の Workshop on Computer Architecture for Pattern Analysis and Image Database Management において Morphology 演算のための計算機アーキテクチャが集大成された．その後 Massively Parallel Processor を基本とするアーキテクチャ [2] や Scan line array processor と呼ばれるアーキテクチャ，光学系を使った Morphology 演算の方法 [3]，そして F.Y.Shih らによる Cytocomputer 型アーキテクチャの Gray-scale 版ともいえる具体的なアーキテクチャが発表されるなど [4]，Morphology 演算のためのアーキテクチャ研究は盛んである．

1.3 Morphology 処理のハードウェア化に関する経緯

Morphology 処理の基本となる演算は集合理論の演算として記述される Dilation，Erosion であり，これらは and，or を用いた論理式で表現することが可能である．

ところが Morphology 処理を含め２次元の重畳積分の形をした演算を逐次処理方式の計算機上に実装した場合，メモリアクセスのボトルネックにより一画面を処理するための時間が長くかかってしまう．つまりこのような計算は逐次処理型の計算機にとっては不得手なタイプの問題であり処理速度が上げられず，動画像のようなスループットの高い情報を扱うことは困難であった．これは基本的には１アドレスのメモリアクセスに対し数マシンサイクルを要するプロセッサのアーキテクチャによるもので，１画面を処理するのに必要となるメモリアクセスの回数は｛原画像のピクセル数｝×｛構造要素 (マスク) のピクセル数｝回となるため，これが高速化を阻むボトルネックとなっている．このため高速な Morphology 処理を目的とするハードウェアとして従来からいくつかの方式が提案されている．

1.3.1 MPP

まず第一番目に紹介するべきは，逐次処理方式の計算機の対局に位置づけられる稠密アーキテクチャ（MPP : Massively Parallel Processor）と呼ばれる並列計算機であろう．

図 1.1 にその模式図を示す．このアーキテクチャを実際の画像処理に適用した例として人工衛星から送られた画像を高速処理するために開発された計算機 [5] があり，128 × 128 台のＰＥ (1 bit の演算を実行する Processing Element) により 6,000MOPS(Mega Operation Per Second) の計算能力を持つとされている．さらに現在では 64 × 64 の整数演算ＰＥと周辺コントローラを 1 chip に集積した汎用画像処理プロセッサが作られている．この種の計算機が使える場合，理想的なモデルとしては一つの画素が一つのプロセッサに対応するため，非常に高速な画像処理が期待できる．しかし MPP は回路全体の規模が大きいことに加え，Morphology 処理では大きな構造要素による処理を実行する際，プロセッサ間のデータ・トラフィックが多くなる問題は避けられない．これは MPP の場合，基本的には各ＰＥ同士が４近傍のみにしかコネクションを持たないことによる．現在最もコネクション密度の高いとされている並列計算機もおいても１６コネクションが最高であるので大きなマスクサイズを必要とする画像処理を行おうとした場合，データトラフィックの増大はさけられない．

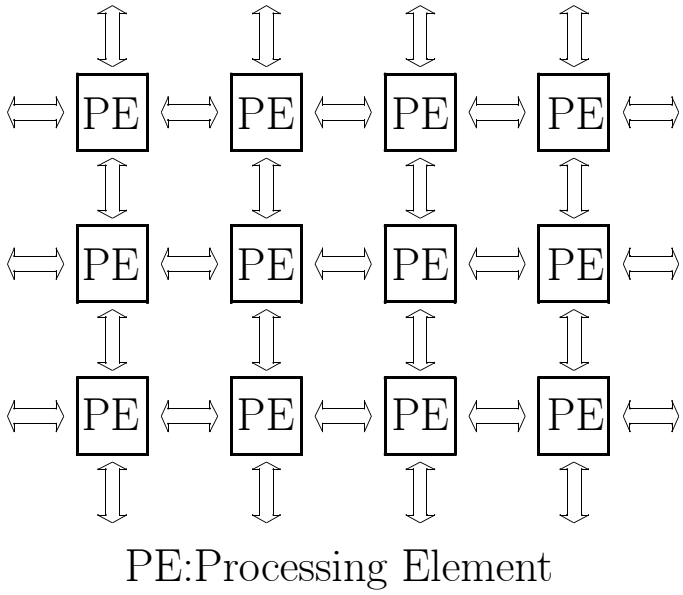


図 1.1: MPP

1.3.2 Cytocomputer

MPP のような大きな計算機によらないで高速な Morphology 処理を効果的に実現する方法として S.R.Sternberg は，図 1.2 に示すような構成のハードウェアによって 3 × 3 ピクセルの構造要素で画像処理を高速に行う Cytocomputer [6] を考案している．また Cytocomputer のサブセットと言える Imageflow Computer [7] と呼ばれる画像処理装置も考えられており，これも高速な画像処理を目的としている．しかしその手法では素子の実装上の問題から大きな構造要素 (例えば 16 × 16 ピクセル以上のような) を作ることが困難となる．また同じ Cytocomputer の構成で Gray-scale の論理に適用するアーキテクチャ[4] も考えられてはいるものの 8 bit（256 階調）の Gray-scale を実現するためには数百K〜数千Kゲート規模の回路を必要とし，さらにこれより大きな構造要素を実現しようとした場合，その実現は容易とは言えない．またこのアーキテクチャの場合 Gray-scale の深さの段数（2 の bit 数乗）の fan in 数を持つ OR ゲートを必要とするため，8bit 程度までであれば実現できるであろうが，16bit 以上の Gray-scale に対してはこの

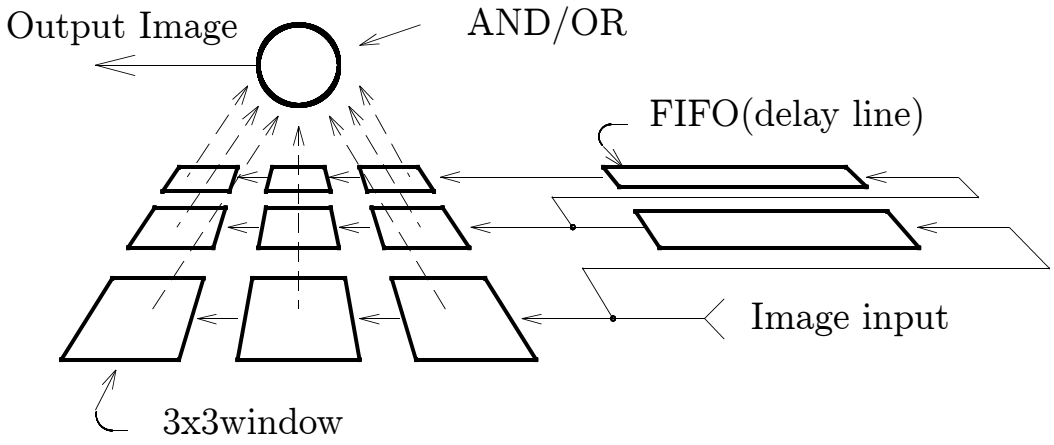


図 1.2: Cytocomputer 型アーキテクチャ

アーキテクチャの実現は難しそうに思われる．

一方，3 × 3 画素より大きな構造要素による Morphology 処理を Cytocomputer で実現する方法として，大きな構造要素を 3 × 3 画素に分割して，多段にパイプライン接続された Cytocomputer でそれらを順番に処理する方法が考えられている．その手法は，”Structuring Element Decomposition” と呼ばれ，Cytocomputer のための理論として位置付けられており，多くの研究がなされているが，たとえば Binary Morphology に対しては J.Xu [8] が，Gray-scale Morphology に対しては F.Y.Shih [9] がそれぞれ効果的な手法を示している．ただしそれらの手法を用いる場合，Binary では，ななめ線が４５° に固定された８角形 (正８角形でなくともよい) に構造要素の形状が制限される．これに対しては複数の構造要素を重ねあわせることでこの制限は回避できるとされているが，その組み合わせを見つけ出す方法は単純とはいえない．また Gray-scale では構造要素の表面形状の関数が単調増加でなければならないという制限があり，しかも任意の形状の構造要素を実現するためには多くのパイプラインの段数が要求されることがしばしばあり得る．

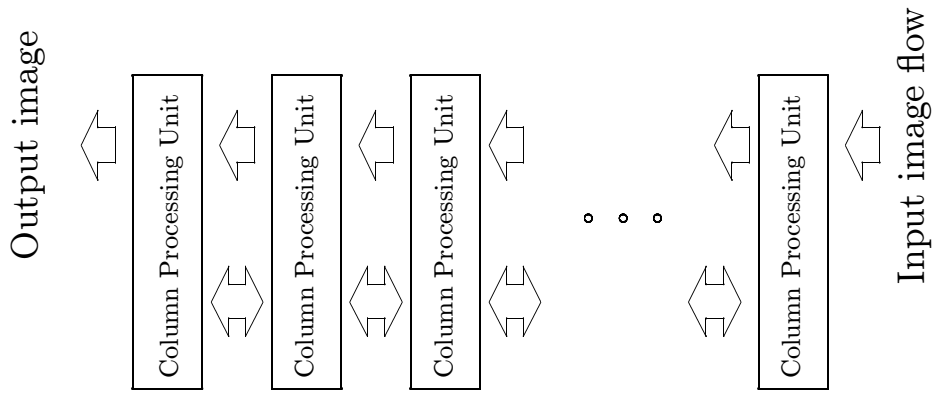


図 1.3: Scan Line Array Processor

1.3.3 SLAP

SLAP [10](Scan line array processor) はどちらかというと汎用画像処理アーキテクチャと言った位置付けがされており，Morphology のための専用計算機ということではないが，その運用しだいで Morphology 画像処理にも使うことができる．このアーキテクチャは，形態としては MPP の 1 次元版とも呼べるもので，図 1.3 に示したような構造を持っている．SLAP は取り扱う画像の一辺の長さの画素数のプロセッサをじゅず繋ぎに配置したアーキテクチャで，入力画像はラスタ走査され SLAP に送り込まれる．各 ”Column Processing Unit” は送り込まれた画像を 1 ラインごとに取り込み，処理を行う．SLAP では Column Processing Unit の一個一個がメモリとマイクロプロセッサを内蔵するため，MPP ほどではないが，ある程度大規模である．またこのアーキテクチャでは，画像の縦方向に渡る演算は一個の Column Processing Unit で行い，横方向の演算には隣どうしの Column Processing Unit 間での通信が必要となるため，その縦・横の非対称性のために，SLAP にインプリメントされるソフトウエアは高度に技巧的である．また取り扱う画像のサイズが Column Processing Unit の個数に押さえられるという問題もあり，これはエミュレーション技術で解決できるとされているものの，簡単な問題ではないと言える．

1.4 Morphology アルゴリズムの１次元化に関する従来の研究

Morphology アルゴリズムの改良方法のほとんどは，構造要素の分解手順の最適化に帰着されている．そしてその分解手法のほとんどは大きな構造要素を 3×3 に分解する手法であると言える．これらは Cytocomputer 型アーキテクチャや MPP に実装するためのアルゴリズムであるといえ，Binary，Gray-scale ともにさまざまな最適化手法が提案されている．

その一方で，画像処理の１次元化に関する研究も行われてきた．ここで言う１次元化とは，画像を横に走査して処理し，その結果を次に縦に走査して処理することである（縦・横の順番は逆でもかまわない）．このような考え方はフィルタや 画像のフーリエ変換といった場面ですでに使われているが [11]，同じことが Morphology 画像処理にも当てはまる場合がある．Morphology 画像処理において,「処理が１次元化できること」と,「構造要素が１次元化できること」とは等価であるため，本論文においては，上の２つの表現を同義として取り扱っている．これらのことがなぜ等価となるかは，Section 2.4 に説明する．

1.4.1 Binary Morphology の１次元化

Binary Morphology に関しては 3×3 構造要素よりもさらに小さい構造要素を用いる方法が考えられており，これは最終的な段階として，Morphology 処理を１次元の処理に帰着することができることを示した例と言える．図 1.4(a)，(b) に示した構造要素のベクトル化は，Zhuang と Haralick が提唱している Image flow computer による計算の手順を示したものである [7]．Zhuang と Haralick はさらにこのベクトル化された構造要素の構成手順を最適化するため，検索木の考え方を導入することにより，なるべく少ないベクトルの組み合わせで済ませる方法に関しても提案している．

このような Image flow computer の考え方を用いることにより，もとの構造要素に含まれる画素数を m 画素としたとき，最良の場合では計算量を $log_2 m$ に押さえることができる．ただし，その最良の場合というのは構造要素の形状が長方

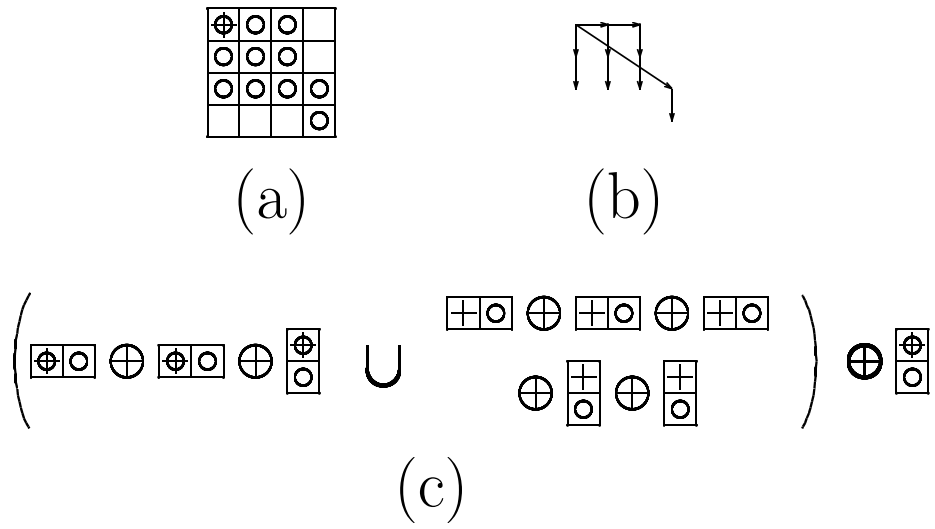


図 1.4: 2 画素構造要素への分解

形で，かつ 2 辺のそれぞれの長さが 2 のべき乗画素という非常に特殊な場合のみである．このため，一般に $log_2 m$ 以上の計算量が必要であるし，最悪の場合では改良を加えない定義式通りの計算方法の場合と計算量が変らないことになる．

図 1.4(c) は，もとの構造要素である図 1.4(a) を 2 画素の構造要素まで分解したものであり，この段階において，構造要素は 1 次元化されたと言うことができる．図 1.4(c) では 2 画素大の構造要素が 9 個含まれるため，この例では構造要素を分解したことによる計算量低減のメリットは現れていない．その理由は，図 1.4(b) に含まれる斜めのベクトルによる移動を実行するために，5 個の 2 画素構造要素が必要となることに起因する．

1.4.2 Gray-scale Morphology の 1 次元化

画像処理の 1 次元化に関しては Paul D.Gader の研究が有名であり，その中で Gray-scale Morphology 処理の 1 次元化に関しても取り扱われている [12]．この

中で Paul D.Gader は，Gray-scale Morphology 処理が 1 次元化できる条件として，構造要素が Additively separable という特殊な形状のクラスに属する必要があることを示している．そして Additively separable ではない構造要素に対しては，輝度値に対して最小 2 乗誤差で Additively separable に近似する手法を示している．Gray-scale 構造要素の 1 次元化に関する諸事情や Additively separable の定義に関しては Section 3.3.1 ～ Section 3.3.3 に述べる．

また，1 次元化された Gray-scale 構造要素をさらに小さな構造要素に分解する方法 [13] がある．その手法では，Gray-scale 構造要素を，その表面形状の持つ曲線を直線で近似し，さらに変曲点で複数の構造要素に切り分けることにより，Cytocomputer 型アーキテクチャに実装可能な計算方法としている．

1.5 Morphology 処理に対する問題提起

1.5.1 必要とする処理速度

どのような場面でどのような画像処理を行うかによって，高速性が要求される場合とそうでない場合がある．また小さな構造要素を用いるのであれば，処理速度はさほど問題にならないが，Morphology 処理において大きな構造要素による処理と高速性とは同時には実現しがたい．このような理由から高速性が要求されなければ，また用いる構造要素が小さければ逐次処理方式の計算機による処理で充分である．

ところがある程度高速な処理と大きな構造要素が要求される場合には専用計算機がぜひ必要となる．Cytocomputer はこの目的に対する現実的な解答であるといえるが，3×3 画素の構造要素による縦列処理を前提とするこのアーキテクチャの場合，最終的に構成したい構造要素の大きさや形状によって，装置の規模を増やす必要もあるため設計時に想定した目的以上の使い方が難しいことになる．

また画像の実時間処理のように高速性が要求されるのなら，MPP や SLAP といった選択肢も当然出て来るであろう．しかし，高速処理と大きな構造要素を両方とも実現するにはそれらに比例したコストが必要になるといっておそらく間違いない．

1.5.2 要求される構造要素の大きさ

Morphology 画像処理にもさまざまなものがあり，低レベルプロセシングと呼ばれる 3 × 3 画素構造要素を使った Morphological フィルタリングから，Convex hull と呼ばれる構造要素の大きさが無限大となる極限における画像を議論する問題まで，アプリケーションごとに構造要素の大きさは全く異なる．すなわち「必要とする構造要素の大きさの見積は画像処理の目的によって，いかようにもなり得る」と言わざるを得ない．

また大きな構造要素を小さな構造要素に分割して処理する手法が数多く考えられており，Morphology の研究の多くの部分がこの部分に注がれているといっても過言ではないが，小さな構造要素によるくり返しの Dilation で構成可能な部分とそうでない部分にわけて考える必要があるなどそれらの手法は単純ではない．

1.5.3 構造要素の形状に関する議論

Morphology 画像処理で用いる構造要素は，一般にその大きさが 3 × 3 画素といった小さなものに限定されることは少なく，むしろ構造要素自体が何らかの幾何学的な (ジオメトリックな) 形状を持つものとして取り扱われることが多い．では,「画素の集合がある形状を反映するためには，どれだけの画素数が必要か」,という議論が必要になるが，これに対して一般的な回答を示すことは困難であろう．ただ，モザイク画のようなものを思い浮かべて見ると，人間の顔のような形状を表現するのに数画素四方の解像度では到底不可能であることは明白である．Morphology 画像処理に用いる構造要素においても，3 × 3 画素のような小さな領域では，たかだか正方形か菱形しか表現できず，5 × 5 をもってしてもせいぜい八角形が表わせるのみである．そしてこの「八角形 イコール 円形」といった従来から使い古された近似表現があるが，この表現は妥当であろうか．

このような近似が従来行われて来た背景には，Morphology の定義がデジタル画像ではなくベクトルの集合を使っていわばアナログ的に表現されていることによるものと思われる．そしてこの定義から導かれる帰結である「小さな円形構造要素どうしの足しあわせは大きな円形構造要素となる」ことをそのまま利用して

デジタル画像にあてはめてしまうことが当然のように行われている．ここで不幸にも，数画素大の円形は「たまたま」八角形と同じ形である．そして八角形をいくらたしあわせても八角形は八角形であって円形とはならない．ところがなぜかこの八角形をそのまま拡大した差渡しが数十画素におよぶ八角形までも円形構造要素として定義してしまう画像処理手法が見受けられる．このような見た目にも円形とは言いがたい，大きな八角形構造要素までも円形と言ってしまうにはいささか強引にすぎるのではないだろうか．そのような無理な近似を用いず，円形であるべき構造要素はもっと円形らしい構造要素を使えばさらに好ましい画像処理結果が得られるのではないか．

1.5.4 ラスタ走査を基本とした画像処理

ラスタ走査方式は２次元の画像を１次元のデータ列に変換するもっともオーソドックスな方法であると言えるが，画像処理のハードウェアや，そのアルゴリズムまでラスタ走査を基本としてつくられているものが多く見受けられる．これは画像を取り込む装置であるカメラから，ビデオ装置，放送局設備から受像装置に至るまで，画像を扱う装置がほとんどすべてラスタ走査された画像信号を取り扱うように設計されていたために，画像処理装置やそのアルゴリズムもたまたまそのように作ることが都合が良かったという理由なのであろう．Cytocomputer も SLAP もこの方式と言えるが，画像処理アルゴリズム全般にとってこの流儀は好ましいものであろうか．

1.5.5 処理の semi-group 化

フィルタリングの演算の計算量を低減する手法として”semi-group operation”という方法が紹介されている [14]．これは，大きなマスクによる画像処理をそのまま実行するのではなく，分解された小マスクによる処理の組み合わせで同じ処理結果を得ようとするものである．この手法がうまく適用出来れば，画像処理に必要な計算量をマスクの面積のオーダからマスクの一辺の長さのオーダに削減することができる．この考え方が有効に機能する条件は「マスクが四角形であること

(画像に対して斜めに置かれてはいけない)」と,「マスク一様」であることである.ここで「マスク一様」とは,ある出力画素の値を決定するとき,マスクの覆っている範囲の原画像のすべての画素に対し平等な演算を施すことである.この方法を導入することにより,マスク内の最大,最小および平均値を求めるフィルタリングなどの計算量のオーダを低減することが可能である.

では Morphology 画像処理にこの考え方が適用可能であるかといえば,残念ながら一般に不可能である.Morphology 画像処理における構造要素という概念自体がすでに「マスク内非一様」を象徴しているともいえるからである.このような場合,計算の過程で semi-group 化を行ったとしても計算量の低減にはつながらない.なぜなら,マスクの位置を移動たときに,以前の位置で計算した小マスクの計算結果を利用することができないからである.また Gray-scale 構造要素の各画素がすべて同じ輝度である場合に限り「マスク一様」とも解釈できるが,それは結果的に最大値・最小値フィルタと同じことになり,これも Morphology 処理のうちの非常に特殊な場合にしかすぎない.すなわち,一般に Morphology 画像処理に semi-group 化の考え方は適用できないとしてよいだろう.

1.6 本研究のアプローチ

本論文では Section 1.5 に述べた疑問点に対して,ある一定の解決になるであろう方法を述べる.この方法は Morphology 画像処理のハードウェアを,Morphology のアルゴリズムとハードウェアの両面から再検討する形で構築するものである.本論文では Binary Morphology のためのアーキテクチャと Gray-scale Morphology のためのアーキテクチャについて述べるが,それらはすべて 1 次元化された Morphology 演算を実行する.それらのアーキテクチャとアルゴリズムは不可分な関係にあり,「処理を 1 次元化したからこそ,そのようなアーキテクチャが生まれた」という言い方と,「そのようなアーキテクチャがあるからこそ, 1 次元化することに意義がある」という両方の言い方ができる.

1.6.1 本論文における主張点

まず Binary Morphology に関しては，長方形構造要素の組み合わせを基本とする Morphology 処理の 1 次元化法について述べる．そして，大きな構造要素による Morphology 処理を高速に実行でき，かつ小規模な回路で構成できるハードウェアを提案する．この方法は Section 1.4.1 で紹介した 1 次元化法と比較して，ハードウェアに対する要求が構造要素の大きさによらないことが特徴である．

Gray-scale Morphology に関しては Additively separable 構造要素を用いる 1 次元化を基本とし，あらたにパイプライン式の 1 次元 Gray-scale Dilation プロセッサを提案する．Dilation 演算の定義式には，多数の入力値の中から最大値を得る演算である ”max” 演算が含まれるが，この演算を実行するためのハードウェアは複雑にならざるをえない（このことに関しては，Section refsec:ComplexHardGray で詳しく議論する）．これを解決するため，”max” 演算を比較と代入に置き換えた，パイプライン式のプロセッサで Dilation 演算を実行する方法を提案する．この方法は 1 次元上で扱う Gray-scale Morphology 処理ならではのアーキテクチャであるといえる．この方式の利点は，Cytocomputer を基本とする方式と比較してハードウェアの規模が小さくて済むこと，画像の Gray-scale の bit 数を多とることにハードウェアが対応出来ること，大きな構造要素を用いることに無理がない，といった利点がある．

さらに，このアーキテクチャを発展させた「テーブル変換器」を用いたアーキテクチャを提案する．これは Gray-scale Morphology の論理を一部拡張することによって，適用可能な範囲をひろげたものであり，主に図形画像の距離変換をターゲットとしている．このなかで，1 次元のパイプラインプロセッサにわずかなロジック回路を付加することで，距離を求める際の「種」画素の座標を記録する「indexing 機能」が実現出来ることに関しても提案する．そしてテーブル変換器アーキテクチャに適用することによりその性質が重要な意味を持ってくる「nose 型構造要素」を紹介する．以上に述べた要素を組み合わせると，ロボットビジョンや図形処理の各分野における基本的計算手法として良く用いられるボロノイ図を高速に，かつ安定に求めることが可能になる．

最後に本論文に述べるいくつかのアーキテクチャやアルゴリズムに関して，そ

の適用範囲がどこまででであるかを明らかにする．Morphology のアルゴリズムに関して言えば，どういったアーキテクチャに実装するかによって，同じアルゴリズムが効率的な物にもそうでないものにもなりうる．さらに，構造要素の形状によっては，あるアルゴリズムでは適用可能だが，別のアルゴリズムでは適用不可能という場合もおこる．このため，構造要素の形状によって，どのアーキテクチャではどのような構造要素が適用可能なのか，といった各手法における適用範囲について，それぞれを比較しながら述べる．

1.6.2 アプリケーションから見た本研究の意義

本論文は，Morphology 処理を高速に実行するためのハードウェアに関して取り扱っているが，この研究の成果の上にどのようなメリットがあるかについてここで述べる．

まず，ロボットビジョンの分野のように画像処理を実時間で実行するような必要のある分野においては，このような高速な処理が要求されるであろう．本論文の Chapter 2 でも，ビデオレートでの実時間処理を目的としたハードウェアとアプリケーションを取り扱っている．

つぎに，現状で研究されている Morphology 画像処理の中には，非常にたくさんの大きな構造要素を何回もくり返し用いる必要のあるものがあり，それらのアプリケーションに対しては，専用ハードウエアによる高速な画像処理がぜひ必要となる．たとえば I. Pitas が提案している図形分解は，Gray-scale Morphology を用いるもの [15]，Binary Morphology を用いるもの [16] 双方に，大きな円形構造要素による Morphology 演算をくり返し実行する必要がある．また P.Maragos が提案している画像の Pattern spectrum 解析 [17] や Morphological skeleton [18] なども非常にたくさんの構造要素使うという意味でを好例である．また本論文で述べる Dilation のボロノイ図への応用においても，原画像の一辺の長さの $2\sqrt{2}$倍の大きさの構造要素を必要とするなど，大きな構造要素の需要は少なくないはずである．

1.6.3 実効性という点での本研究の意義

またハードウェア面での基本的な用件として，提案するアーキテクチャが入手可能なデバイス技術で実際に作れることが必要である．これは当然満たしているべき用件ではあるものの，アーキテクチャを提案している論文の中には，実際のデバイスでによる実装に懐疑的にならざるをえないものがあることも確かである．この意味において，本論文では提案するすべてのアーキテクチャに関して実際の試作機を設計・製作することにより，動作の検証ができているため，アーキテクチャの妥当性に対する強力な裏付けとなっていると言えよう．また，実装の段階でどういった問題が生ずるかといった，回路の技術的な検討を行うことができることや，どれだけの実行速度が得られるか，といった実用性の検証という意味においても，この試作機が裏付けとなっている．

1.6.4 専用アーキテクチャの意義

さらに付け加えると，汎用コンピュータの性能が１年で数倍向上する現状では，ある処理に特化した専用アーキテクチャを考えたとしても，すぐに意味は薄れるという意見もあるが，この意見に対しては２通りの答えができると考える．

第一はその意見を肯定する答えであり，使う目的が限定されていて，ある程度の処理速度が得られればそれ以上の高速化はまったく不要な場合である．その場合にはたしかに汎用プロセッサの処理速度向上を待った方が，労せずに同じ結果を得ると言う意味では得策といえよう．ただし，画像処理の速度に関して言えば，プロセッサの処理速度よりもメモリのアクセス速度に依存する所が多いため，プロセッサの劇的な処理速度の向上に比べてメモリアクセス速度の向上がほとんどみられない現状では，汎用計算機による画像処理の速度の向上はしばらく待つ必要があると言えよう．

第二は上記の意見を否定する答えであり，処理が速ければ速いほど良い場合である．すなわち，要求される処理速度に対して現状がまったく追い付かず，汎用計算機の性能向上を待っていることができないという場合である．こういった場面では専用アーキテクチャは必要であり続けるだろう．また「高速な汎用プロセッ

サを作るのと同じデバイス技術を用いるのであれば，専用アーキテクチャの方が必ず高速である.」という命題は当分の間は真でありつづけるであろう．

1.7 本論文の構成

本論文は Chapter 2 で Binary Morphology のための Morphological アーキテクチャを示す [19]．そこではまず Morphology のアルゴリズムを 1 次元化，並列化する方法を論じ，次にそのアルゴリズムを用いて構造要素の大きさに制限がなく，ビデオ信号を実時間で処理することを可能とするハードウェアを示す．また応用として Bright eye（赤外線の網膜反射像）の画像補正にこの装置を組み込んだ例を示す．この例ではビデオ信号として取り込んだ Bright eye の輪郭の明確化とノイズ除去を実時間で実行することを目的としている．

Chapter 3 では Gray-scale Morphology のためのアーキテクチャを示す [20]．これは Chapter 2 に述べたアーキテクチャの Gray-scale 版と位置付けることができるが，その実現方法は Gray-scale Morphology にあわせて大幅に変更されたものとなっている．このアーキテクチャの心臓部は「1 次元 Dilation 回路」であり，もっとも重要な部分である．この 1 次元 Dilation 回路がなぜ有効なのかといった有利性に関して，Cytocomputer 型アーキテクチャとの比較において，回路規模や回路のスピードといった点で詳しく比較する．

Chapter 4 で述べるアーキテクチャ[21] は Chapter 3 に述べたアーキテクチャを含む形で発展させたものである．これは，Dilation プロセッサの部分にテーブル変換器（RAM を含むロジック）を組み込むことにより，図形（2 値画像）の距離変換に適用範囲を広げたものである．またここでは，画素の輝度を表す Gray-scale 方向の bit 数と誤差の関係，さらに誤差の影響を少なくするための nose 型構造要素の形状について提案する．さらにもう一つの特長として，ボロノイ図を求める際に必要となる母点の座標を保持するための工夫もされている．この部分の論理は Morphology の演算の論理とは建前のうえでは無関係であるが，Morphology 演算を計算幾何学的なアプリケーションに適用しようとしたばあい，ぜひ必要な機能となる．

Chapter 5 は論文全体を通しての各方式の位置付けやその役割について述べる．

本論文では本論の構成をシンプルにするため，ストーリーの構成から見て冗長と思われる部分は補遺の章として Appendix に一括してまとめるスタイルをとった．また Appendix C にはそれぞれの Chapter において示したアーキテクチャの試作機の全回路図を掲載する．

1.8 Morphology の表記法に関して

Morphology の演算の定義の方法にはいくつかの表記法が並立する形で存在し，J.Serra[22] や P.Maragos が用いる表記法と Haralick[23] や F.Y.SHIH[4] が用いる表記法とそれ以外に大別されるが，本論文では Haralick の表記法に従うことにした．これは Haralick による表記法が Dilation と Erosion における座標の扱い方が統一されているなど，論理全体の見通しの良いことと Chapter 3 以降に述べる，Morphology 演算のコンボリュート演算による表記の際，導入が無理なく行えるという理由による．

また本論文では議論の簡単化のため，画像は $N \times N$，構造要素は $M \times M$ 画素の正方形としている．また Binary Morphology の議論では原画像を A，構造要素を B, C などと表記しているが，Gray-scale Morphology の議論では function processing で用いられる表記にならって，原画像を f，構造要素を k と表記している．また本来は，Morphology 演算は N 次元ユークリッド空間上の集合どうしの演算として定義されるものであるが，本論文では計算機の論理としての見通しをよくするために，その部分集合である離散空間上での演算に限定して用いている．

Chapter 2　Binary Morphology のためのアーキテクチャ

2.1　Chapter2のあらまし

Binary Morphology は２値画像の画素間の演算であり，これを画像処理に応用することにより，ノイズ除去，境界線の平滑化，形状記述など様々な応用が可能とりる．また Morphology 処理は，他の多くの画像処理においてその前処理に用いられたり基本演算として組み入れられる形で使われるなど，その有用性は非常に高い．処理の基本となる演算は集合理論の演算として記述される Dilation，Erosion であり，これらは AND，OR を用いた論理式で表現することが可能である．ところが Morphology 処理を含む画像の局所並列演算は直列計算機上に実装した場合，メモリアクセスのボトルネックにより一画面を処理するのにかなりの時間がかかってしまい，ビデオ画像のようなスループットの高い情報を扱うことは困難であった．また Cytocomputer のような高速な画像処理を目的としたハードウエアにおいても，パイプライン１段において扱える構造要素が狭いという欠点があった．Chapter 2 ではまず，画像の Morphology 処理を高速化するためのアルゴリズムを示し，次に２値化されたビデオ信号を実時間で処理するための専用のハードウェアにこれを適用し，最後にこの装置を実際に組み込んだ医用への応用例を示す.

そこで Chapter2 ではまず Morphology のアルゴリズムを１次元化，並列化する方法を論じ，次にそのアルゴリズムを用いて構造要素の大きさ，形状に制限がなく，ビデオ信号を実時間で処理することを可能とするハードウェアを示す．また応用として Bright eye（赤外線の網膜反射像）の画像補正にこの装置を組み込んだ例を示す．この例ではビデオ信号として取り込んだ図形の輪郭の明確化とノ

イズ除去を実時間で実行することを目的としている．そして最後に Chapter 2 に述べる手法と他のアルゴリズムについて，計算量，扱える構造要素の形状，及びパソコンにプログラムした場合の計算時間について比較を行う．

2.2 Binary Morphology の基本演算

図 2.1 に Morphology の基本演算である Dilation , Erosion , Closing , Opening を示す．図 2.1 （c）に示す Dilation は図 2.1 （a）の原画像（2 値）の on 画素すべてに（b）の構造要素の座標原点を重ね合わせ，構造要素の形で塗りつぶしたものと解釈することができる．それと対の演算としての Erosion は図 2.1（d）に示すように，構造要素が原画像に完全に含まれるときに，そのときの構造要素の座標原点をプロットすることによって求めることができる．図 2.1（e) , (f）に示した Closing , Opening は Dilation , Erosion の組み合わせによって求められる．すなわち Closing は Dilation によって得られた画像に Erosion を施したものであり，Opening はその逆に Erosion によって得られた画像に Dilation を施したものである．Dilation と Erosion はいわば足し算と引き算のように基本的な演算であるが，画像の持つ細かな特徴の欠落を伴うので，Opening と Closing は一般に一致しない．細かい特徴を含む画像に対して Closing を用いると構造要素よりも小さなすき間や穴が埋め尽くされ，逆に Opening を用いると細い突端や孤立点が削り取られる，といった一種のフィルタリングとなる．P.Maragos はこの Closing と Opening を Morphology における基本フィルタとして位置付けており [24]，この意味において Closing と Opening は,「単に Dilation と Erosion から派生したもの」と見なせないほどに重要視されているといってよい．

Chapter 2 では Morphology の演算を表現するのに以下のような記号を用いる（ここでの定義と表記法は Haralick[23] に従っている）．なお以下で用いる記号の A，B，C はそれぞれ 2 値画像の on 画素の集合を表すが，Chapter2 ではとくにことわらない限り A は原画像，B，C は構造要素を意味するものとする．

`Dilation`

$$A \oplus B = \{a + b : a \in A, b \in B\} \tag{2.1}$$

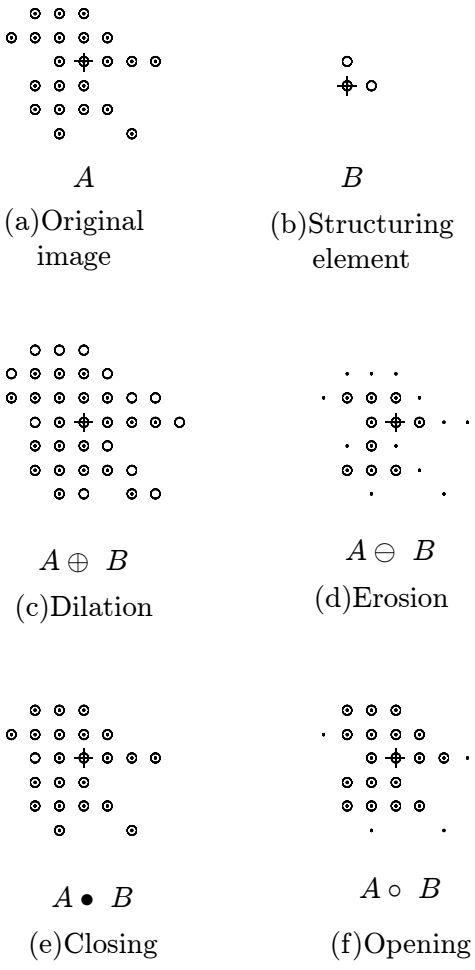


図 2.1: Binary Morphology の基本演算

`Erosion`

$$A \ominus B = \{x : x + b \in A, b \in B\} \tag{2.2}$$

ここに a,b はそれぞれ A,B の各画素の要素（原点から on 画素に向かう位置ベクトル）を表す.

`Closing`

$$A \bullet B = (A \oplus B) \ominus B \tag{2.3}$$

`Opening`

$$A \circ B = (A \ominus B) \oplus B \tag{2.4}$$

さらに式（2.5）,（2.6）に示す結合則と式（2.7）,（2.8）に示す分配則が成り立ち，この性質は複数の構造要素を重ね合わせるときに用いられる.

結合則

$$(A \oplus B) \oplus C = A \oplus (B \oplus C) \tag{2.5}$$

$$(A \ominus B) \ominus C = A \ominus (B \oplus C) \tag{2.6}$$

分配則

$$A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C) \tag{2.7}$$

$$A \ominus (B \cup C) = (A \ominus B) \cap (A \ominus C) \tag{2.8}$$

ここに”∪”，”∩”はそれぞれ集合の論理和，論理積を表す.

さらに，以下の双対性の定理が成り立つ，この定理は Erosion の計算を Dilation の計算で代用することができることを示しており，計算機を構成する上で重要な性質と言える.

双対性

$$A \ominus B = (A^c \oplus \check{B})^c \tag{2.9}$$

ここに，B^c は B の補集合（on，off の反転に相当する）であり，$\check{B}$ は B の反射と呼び B の座標原点を中心に B を 180°回転させた構造要素を表す.

2.3 Binary Morphology 処理に関する問題点

Morphology 処理を含め小マスクによる走査を要する画像処理の問題点としてまず挙げられることは，そのアルゴリズムが直列型の計算機にとって不得手であり処理速度が上げられないことである．これは基本的には1アドレスのメモリアクセスに対し1マシンサイクル以上を要するプロセッサのアーキテクチャによるもので，1画面を処理するのに必要なメモリアクセスの回数は｛原画像の画素数｝×｛構造要素（マスク）の画素数｝回となるため高速化を阻むボトルネックとなっている．

その解決策として S.R.Sternberg は図 1.2 に示すような構成のハードウェアによって 3×3 画素の画像処理を高速に行う Cytocomputer [6] を考案しているが，その手法では素子の実装上の問題から大きな構造要素は作りにくい．これは構造要素中の各素子からの出力をすべてまとめて AND もしくは OR を取る必要があるため配線数が多くなりすぎたり，多入力の論理回路における fan in 数が足りなくなるという実装上の問題による．仮に fan in 数の問題を解決するために AND OR を多層に分けてとろうとすると，今度はゲート遅延が問題となる．また構造要素の大きさを目的に応じて変化させる必要がある場合には，必要となる最大の大きさの構造要素に足るだけのハードウェアをあらかじめ用意しておかなければならず，この場合，内部の素子の稼働率がほとんどの場合において低い，ということになりかねない．

また Cytocomputer に対して有効な，大きな構造要素を複数の小さな構造要素に分割して Morphology 処理を行う手法 [8] も研究されているが，小さな構造要素による縦列処理を多段のパイプライン処理により繰り返すその手法では，扱える構造要素の形状に制約が生じ，この手法においては基本的には八角形の構造要素のみが適用範囲となる．この Cytocomputer 型アーキテクチャとの比較は Section 2.7 において詳しく行う．

これとは別に R.M.Haralick が提唱している Image flow computing という手法がある [7]．これは構造要素に含まれる on 画素をベクトルの組み合わせで表現することにより，もっとも良い場合で計算量を $log_2 m$ におさえる方法である（ここに m は構造要素に含まれる on 画素の個数）．ここで，$log_2 m$ という計算量は画像

の任意シフトと，二つの画像の同じ座標の画素どおしの AND OR 演算を 1 単位とする計算量である．この手法は，原画像中の on 画素の密度が濃い場合にはソフトウエア的に実行する場合でも計算量削減の効果はあるものの，専用ハードウエア向きのアルゴリズムといえる．この際用いるハードウエアは Cytocomputer の構造要素を 2×2 に限定した形のもので可能とされているため，ハードウエアの形態としては Cytocomputer に含まれるとしてよいだろう．また画像の任意シフトと，二つの画像の AND OR 演算を一度に実行出来るようなハードウエアを用いることも可能であり，図 1.1 に示した 4 近傍接続の MPP にも実装可能なアルゴリズムである．

2.4 Binary Morphology 演算の１次元化

Dilation ，Erosion の定義式（2.1），(2.2）をそのまま用いて Morphology 処理の計算を行うと，必要となる論理演算の回数は，原画像の面積と構造要素の面積を乗じた回数となり大きな構造要素の使用は計算コストの増加を招く．そこで Section 2.4 ではこの計算を効率化するため，2 次元の構造要素を 1 次元化して Morphology 処理を行うアルゴリズムについて述べる．そして Section 2.5 においてこの効率化されたアルゴリズムを実装するための専用ハードウエアについて述べる．この，Section 2.4 に述べるアルゴリズムの目的は，

１． 扱える構造要素の大きさに制限がないこと，

２． 構造要素の形状の自由度がある程度確保できること，

３． ハードウェアとして具現化でき，しかも高速であること，

の 3 点にまとめられる．以下に例として円形の構造要素を 1 次元化するアルゴリズムを 3step に分けて説明る．

step1 図 2.2（a）のように目的とする円形の構造要素を複数の長方形の重ね合わせによって近似し，その外形（太枠）の内側の領域を B とする．

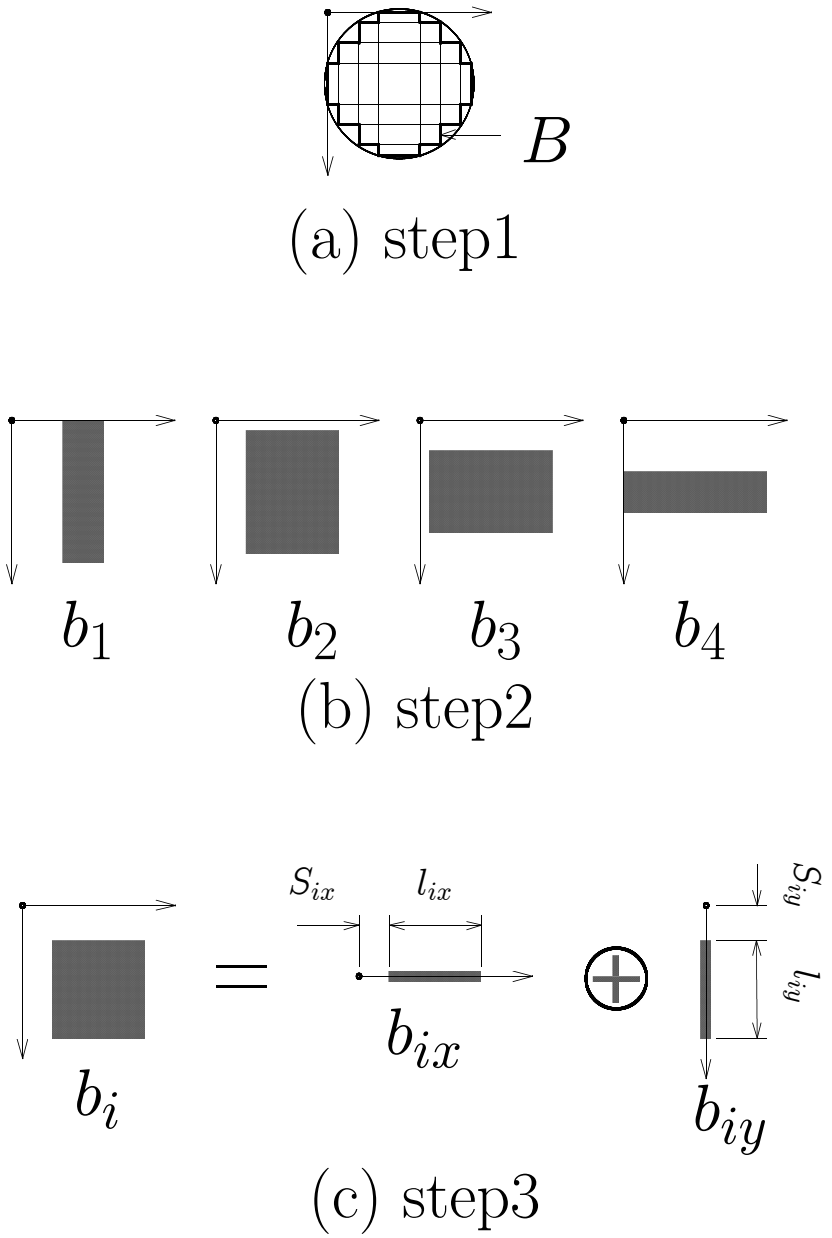


図 2.2: 構造要素の分割

step2 同図（b）のように，

$$B = \bigcup_{i=1}^{n} b_i = b_1 \cup b_2 \cup \cdots b_n \tag{2.10}$$

となる分割された長方形構造要素 b_iを決める（図 2.2 では $n = 4$）．

step3 同図（c）のように長方形 b_iに対し，

$$b_i = b_{ix} \oplus b_{iy} \tag{2.11}$$

となるような 1 次元構造要素 b_{ix}，b_{iy}を決める．

step2，3 をまとめると，

$$B = \bigcup_{i=1}^{n} (b_{ix} \oplus b_{iy}) \tag{2.12}$$

のように表すことができ，このことは出発点の構造要素 B が $2n$ 本の 1 次元構造要素に分割されたことを意味する．この 1 次元構造要素を用いると画像 A の Dilation，Erosion は式（2.5）～（2.8）よりそれぞれ，

$$A \oplus B = \bigcup_{i=1}^{n} (A \oplus b_{ix} \oplus b_{iy}) \tag{2.13}$$

$$A \ominus B = \bigcup_{i=1}^{n} (A \ominus b_{ix} \ominus b_{iy}) \tag{2.14}$$

となる．同図（c）中の l_{ix}，l_{iy}は 1 次元化された構造要素の長さ（以下：要素長）を，s_{ix}，s_{iy}は座標原点から 1 次元構造要素の先端までの距離（以下：シフト量）をそれぞれ表す．l_{ix}，s_{ix}，l_{iy}，s_{iy}を用いれば出発点の構造要素 B は $4n$ 個のパラメタに置き換えられたことになり，これらは Section 2.5 のハードウエアに与えるパラメタとして反映される．

以上の手順において b_iの選び方を工夫することにより，上記の例のような円形構造要素に限らず楕円形や凹形，さらに内部に穴を有する構造要素なども作ることができる．また曲線的な輪郭を持つ構造要素をより精度良く近似するためには，より多くの長方形を組み合わせればよいことになるが，この組み合わせに用いることができる長方形の個数（n）は Section 2.5 に述べるハードウエアによって上限が決まる．

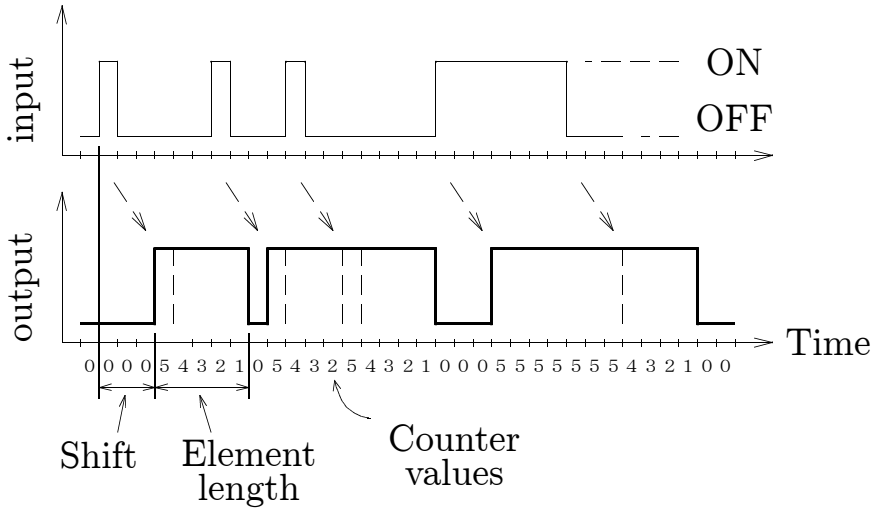


図 2.3: 1 次元 Dilation 回路のタイミングチャート

2.5 Binary Morphology 処理のためのハードウェア

以下に示すハードウエアによる Dilation は Section 2.4 の構造要素の分割とちょうど逆の手順で実行され，もとの構造要素 B が再構成される．このハードウエアの重要な部分である 1 次元 Dilation 回路と H-V 走査方向変換回路をそれぞれ図 2.4，図 2.5 に示す．1 次元 Dilation 回路の入出力関係のタイミングは図 2.3 に示すもので，この出力信号は入力信号（input を Shift だけ遅延させたもの）が on のときは無条件に on となり，その後入力信号が on から off 状態になった後更に Element length に渡って on 状態を保持する（引き延ばす）ことによって得られる．

この動作を実現するのが 1 次元 Dilation 回路（図 2.4）であり，横方向の Dilation について，シフト量（Section 2.4 中の s_{ix}）は可変長シフトレジスタ（Variable Delay）の遅延量（delay time）として，要素長（l_{ix}）は減算カウンタ（Decrement Counter）の初期値（init）としてそれぞれ入力され，縦方向の Dilation についても同様に（s_{iy}，l_{iy}）が入力される．図 2.4 に示す回路の動作おいて，1 次元 Dilation 回路の入力時系列である Input signal はまず Variable Delay でシフト量だけ遅延され，減算カウンタの load に入力される．この減算カウンタは，load 入力が on である間は Clock 信号に伴って要素長（Element Length）がロードされ続け，load

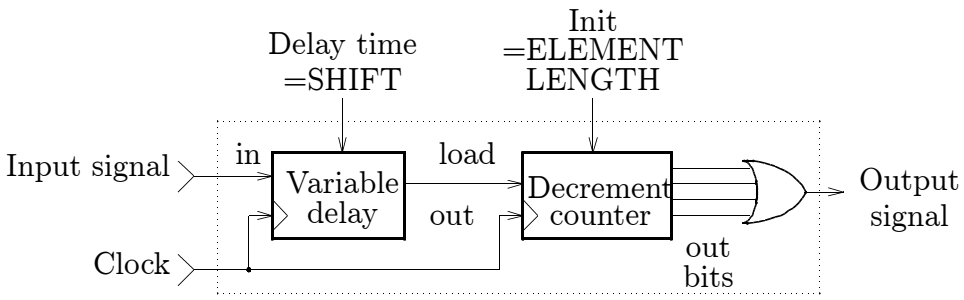


図 2.4: 1 次元 Dilation 回路

入力が off となった以降の Clock パルスから減算が開始され，カウント値がゼロになると動作を止めて，以後 load 信号が on になるまでゼロを出力しつづける．1 次元 Dilation 回路の出力は，減算カウンタのカウント値がゼロでない値を持つとき on となるような信号を作ることによって得られる．図 2.3 中の Counter values は減算カウンタが動作する様子を，要素長＝5 として示したものである．

図 2.4 に示す H-V 走査方向変換回路はフレームメモリの取り込み時と吐き出し時に走査方向の縦横を変えることにより，画像を 90°回転させる目的で使われ，16bit のアドレスを持った RAM を用いることにより 256 × 256 画素の画像を扱うことができる．ビデオ信号の H-V 座標を交換するためのアドレス発生方法はいくつか考えられるが図 2.4 に例示した回路では 16BIT カウンタの出力の上位と下位をマルチプレクサによって入れ替えることにより RAM の 16bit アドレスの上位と下位を交換して，画像の走査方向の縦横を変えている．

以上に説明した回路を図 2.6 のように n 系統並列に組み合わせることにより，２次元の Dilation が実現される．同図において，入力画像はまず横方向に走査され，1 次元構造要素 b_{1x}〜b_{nx}による横方向の Dilation が並列に実行され，RAM に書き込まれる．次に RAM から読み出す段階において走査方向の縦横を交換し，1 次元構造要素 b_{1y}〜b_{ny}によって縦方向の Dilation を並列に実行する．最後に，得られた n 系統の長方形構造要素による Dilation 画像の論理和を取ることにより，目的の構造要素による Dilation 画像（出力画像）を得る．以上の過程で入力画像

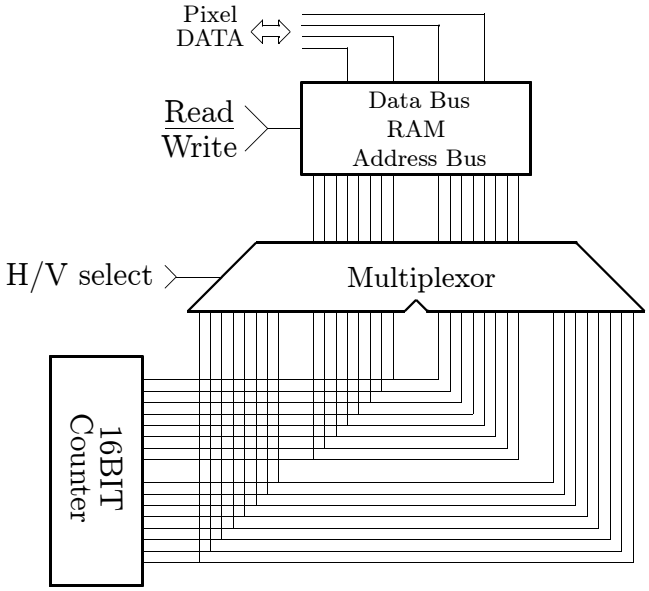


図 2.5: H-V 走査方向変換回路

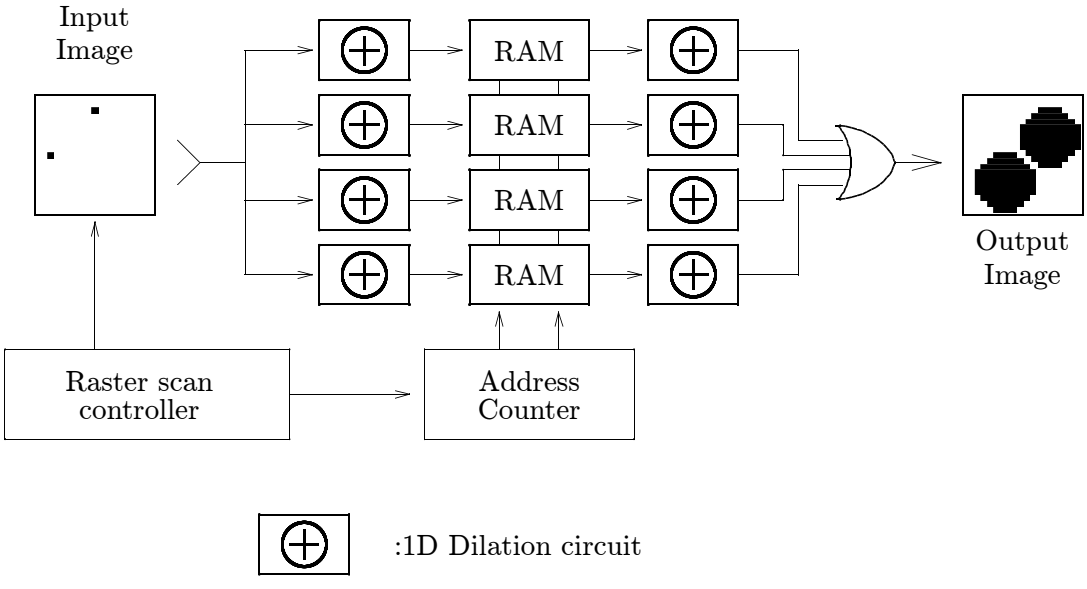


図 2.6: 1 次元 Dilation 回路と H-V 走査方向変換回路の結線

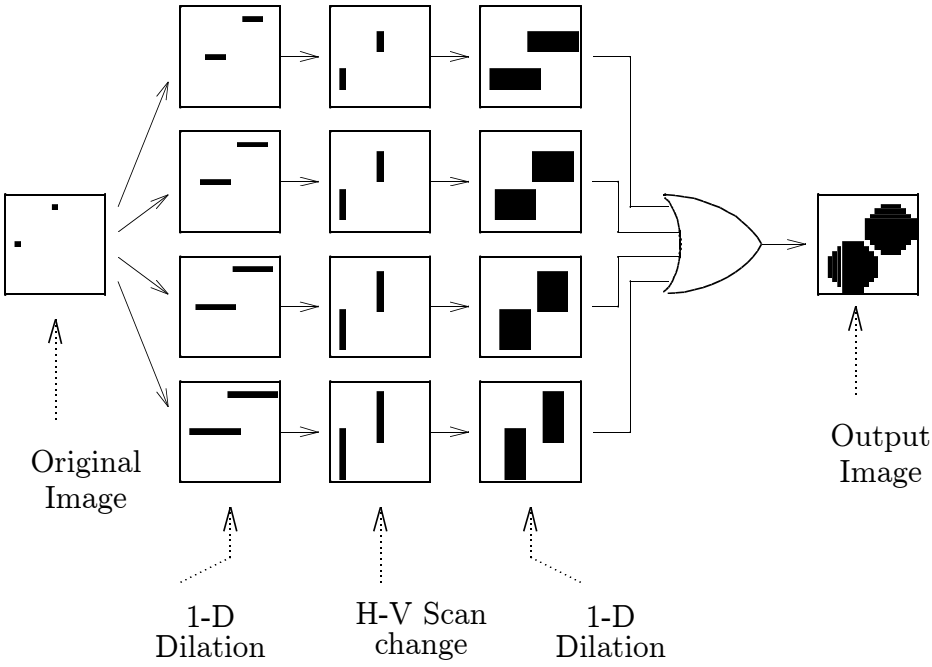


図 2.7: 信号の流れと構造要素の再構成

と出力画像の間には 1 フレーム分の遅延が生ずる.

図 2.7 は構造要素の再構成と信号の流れる様子を簡単な入力画像を例として示したもので，ここでは Section 2.4 の構造要素の分割で用いた，長方形が 4 系統の場合を例にとって描いている．なおこの構成の装置の場合，一度交換した H-V 座標をもとに戻す操作を行っていないので，出力画像は縦横が入れ替わったままになっている．もしそのことが使用上問題となる場合には，最終の出力段でもう一度 H-V 走査方向変換を行えばよいことになるが，その場合 RAM への書き込み，読み出しの操作によって最終的な出力画像は更に 1 フレーム分遅れることになる．画像の Erosion は双対性（式 (2.9)）により，Dilation 回路の入力信号と出力信号を反転させることで得られる（図 2.8）．なお図 2.8 中の「Dilation circuit」は 1 次元 Dilation 回路ではなく，画像の Dilation すなわち図 2.6 全体に相当するも

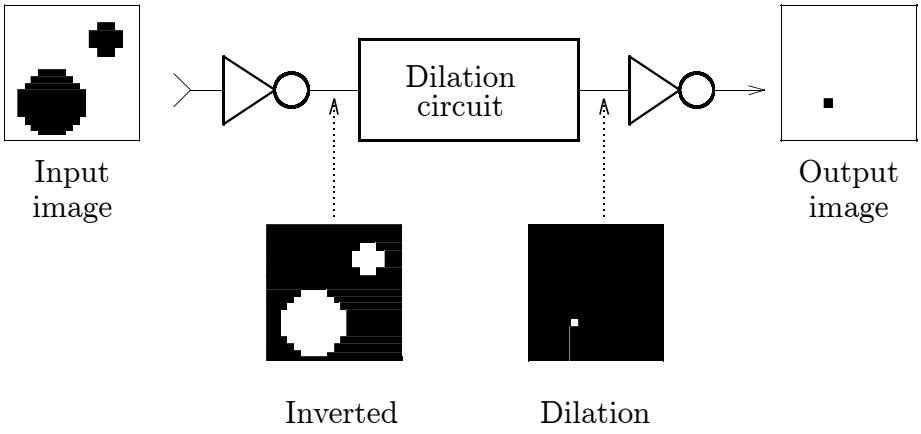
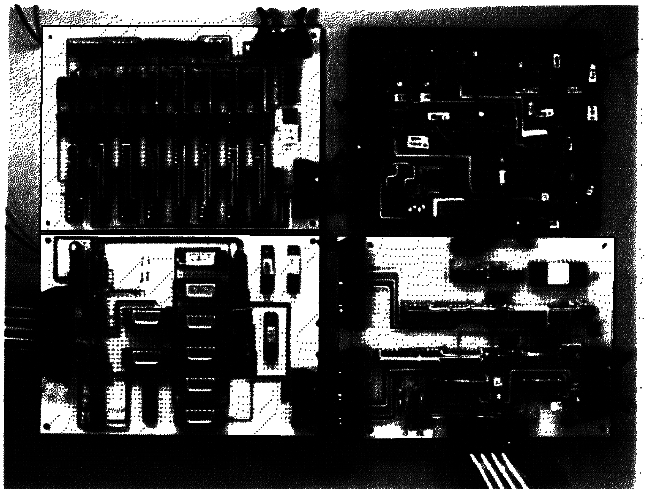


図 2.8: erosion 回路

のを意味する.

2.6 試作機と適用例

Section 2.5 で示したハードウエアを含むMorphology 画像処理装置を試作し実際のビデオ画像に用いた例をここで示す. この試作機はNTSC ビデオ信号（even/odd 両フィールド合わせて 60 フレーム毎秒）を 256 × 256 画素でデジタイズ，２値化したものを実時間で処理するものである. このため画像処理の速度としては 1 画素当たり約 200ns で処理することが要求されるため，このマシンサイクルにちょうど見合うロジック回路の設計が必要となる. 試作機全体の構成は大まかに分けて，ビデオ画像入出力部，Morphology 処理部，及びコンピュータとのインタフェース部からなる. 図 2.9 がその主要回路であり，右上がビデオ画像入力部，左上が 1 次元 Morphology 処理部，右下が H/V 変換回路を含むアドレス発生回路，左下が画像メモリである. この試作機では even（odd）フィールドのX方向 Dilation を実行しながら odd（even）フィールドのY方向 Dilation を同時に実行するインタリーブ構成としたため，even/odd それぞれの専用メモリを持っている. そして 8 系統の 1 次元 Dilation 回路を持たせることにより，$n = 4$ 系統の長方形 Dilation を同時に実行することが可能となっている. また Morphology 処理



1 次元 Morphology 処理部	ビデオ画像入力部
画像メモリ	アドレス発生回路

図 2.9: Binary Morphology 処理回路

部の回路規模は，８系統の１次元 Dilation 回路全体がＴＴＬ相当のロジックＩＣで 32 チップ，及び H-V 変換回路に要する画像メモリが 64 K × 1bit の高速 RAM 8 個となっている．

図 2.10 に示した画像は，試作機を実際のビデオ画像に適用した際の入出力画像である．これは医用応用への研究の一環として，Bright eye 画像（赤外線の網膜反射像）から視線方向を検出するための装置 [25] に Morphology 回路の試作機を組み込んで，カメラから取り込んだ画像の前処理に用いた例である [26]．CCD カメラで Bright eye を捕える際に眼とカメラが距離的に離れていると，Bright eye の光強度が低くなり２値化が完全には行えず，２値化した Bright eye 内部に off 画素が相当数残るという問題が生ずる．これは Bright eye 画像自体が充分な光量を持たないため CCD カメラでとらえた画像が図 2.11 に示したような条件の厳しい輝度分布になってしまうためである．また場合によって，目頭などからの乱反射によるノイズの影響を受けたり，まつげによって Bright eye の輪郭が変形を受けるという問題点も生ずる．

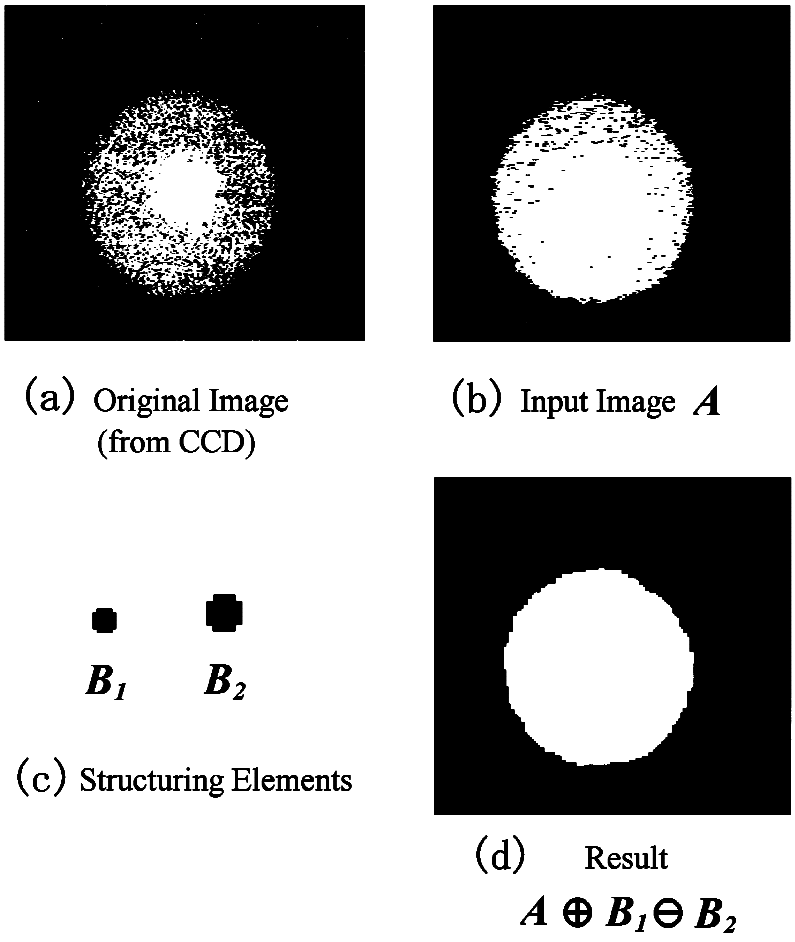


図 2.10: ビデオ信号の Morphology 処理の例

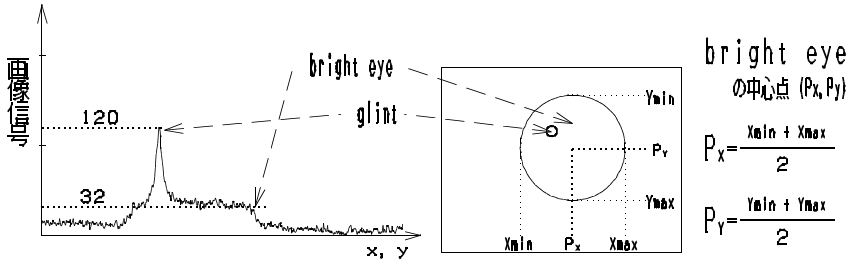


図 2.11: Bright eye 画像の輝度分布

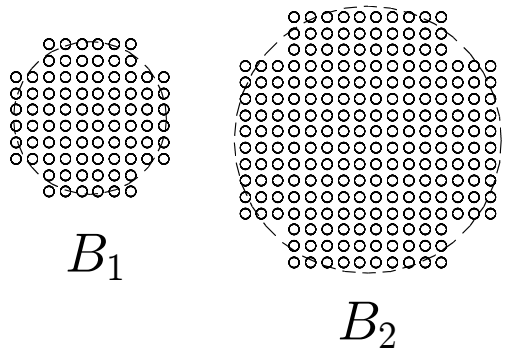


図 2.12: B_1,B_2の詳細

表 2.1: B_1，B_2に与えるパラメタ

	s_{1x}	l_{1x}	s_{1y}	l_{1y}	s_{2x}	l_{2x}	s_{2y}	l_{2y}
B1	2	6	0	10	0	10	2	6
B2	3	10	0	16	0	16	3	10

図 2.10（a）はCCD カメラで捕えたBright eye の原画像，同図（b）は原画像の 2 値化画像（これを A とする），同図（c）はここで用いる 2 つの構造要素，同図（d）は入力画像 A に対し構造要素 B_1 でDilation を行ったのち構造要素 B_2 でErosion を行ったもの，すなわち $A \oplus B_1 \ominus B_2$ である．同図（c）中に示した構造要素は画像に対して同じ縮尺で，ここではDilation とErosion の縦列処理を同時に実行するため，試作機が構造要素の組み合わせに使うことのできる 4 系統の長方形を 2 系統（$n = 2$）ずつに分けて B_1，B_2に割り当てた．図 2.12 は B_1，B_2の詳細であり，目的としている構造要素（同図中の破線で描いた円）に対し 1 画素程度の誤差を含んでいる．なお，Section 2.4 に説明したハードウエアに与えるパラメタはこの場合，表 2.1 に示した通りである．

また構造要素 B_1による Dilation と，B_2による Erosion の縦列処理は，10 画素大の構造要素による Closing 画像に 6 画素大の構造要素による Erosion を行った縦列処理とほぼ等価になる．同図（b）と（d）を比較すると，Morphology 処理

$(A \oplus B_1 \ominus B_2)$ により，

· Bright eye 内部の off 画素が埋め尽くされる．
· Bright eye 外部の細かなノイズが除去される．
· 輪郭がはっきりする，

といった効果が得られていることがわかる．

2.7 アルゴリズムの比較

Section 2.7 では以下の 3 種類の Morphology ・アルゴリズムについて，計算量，扱える構造要素の形状及びパソコン上にプログラムした場合の計算時間の比較を行う．

アルゴリズム１ ：定義式（2.1），(2.2）をそのまま用いる方法．

アルゴリズム２ ：Cytocomputer が用いる方法で複数の 3×3 構造要素による縦列処理．

アルゴリズム３ ：Section 2.4 に述べた，1 次元化された構造要素を用いる方法．

まず計算量（データの参照回数）のオーダについて考える．原画像の面積（画素数）を S，扱う構造要素を $M \times M$ 画素とした場合，Dilation ，Erosion で必要とする論理演算の回数のオーダは，アルゴリズム 1 の場合 $O(S \cdot M^2)$ となる．これはこの方法が出力画像の 1 画素毎に $M \times M$ 画素の原画像を参照する必要があるためである．アルゴリズム 2 の場合には $M \times M$ 画素の構造要素を最小の場合で $M/2$ 個の 3×3 画素の構造要素に分割して，それぞれの 3×3 構造要素についての縦列処理を行うため，計算量はおよそ $O(S \cdot M)$ となり構造要素の一辺の長さに比例するといえる．ただし，これは最小の場合であり，構造要素の形状によってこれより大きくなるのが一般的である．

これらの手法に対しアルゴリズム 3 では，構造要素の近似に用いる長方形の個数を n としたとき，計算量は $O(S \cdot n)$ となり，構造要素の大きさには独立である．

これはアルゴリズム 1，2 が原画像と構造要素の画素ごとの論理演算を行うのに対し，アルゴリズム 3 が原画像をラスタ走査しながらそれに伴って変化する内部変数（Section 2.4 中の「カウント値」）を参照することによって出力画像を得ることによる．ただしアルゴリズム 2 と 3 に関しては，それぞれの計算手順をパイプライン化，並列化することができるため，それらの専用ハードウエアを用いてビデオ画像を実時間処理する目的に関しては両者の処理速度（スループット）は等しいと言える．

次に扱える構造要素の形状について考える．アルゴリズム 1 では定義式通りの計算を行うため構造要素の形状は完全に任意である．アルゴリズム 2 では構造要素の形状はその外形が縦，横，斜め 45 度の輪郭線のみから成る 8 角形以下の凸型多角形とその組み合わせに限定される [8]．これに対しアルゴリズム 3 では構造要素の形状は基本的には任意であり輪郭に凹部を含む形状の構造要素にも適用可能である．またアルゴリズム 2 では構造要素を拡大したい場合には多段の縦列処理を必要とするが，アルゴリズム 3 の場合はパラメタの変更のみで足るという特徴を持つ．

構造要素の形状に対するアルゴリズム 3 の弱点は，少数の長方形による当てはめが困難な形状の構造要素を必要とする場合であり，必要とする長方形の個数（n）が多くなりすぎる場合にはアルゴリズム 3 の利点は生かしにくいことである，このような構造要素を Section 2.5 に述べたハードウエアに適用する場合には扱える長方形構造要素の個数（n）をあらかじめ多く用意しておかなければならない．

最後にそれぞれのアルゴリズムをパソコン上にプログラムした場合の計算時間を比較するため Section 2.6 で用いた Bright eye 画像の Morphology 処理と同様の計算をそれぞれのアルゴリズムを用いて実行した．結果は，アルゴリズム 1 では 6.34（秒／1 画面），アルゴリズム 2（3 × 3 画素構造要素による縦列処理 13 段）では 2.71（秒／1 画面），そしてアルゴリズム 3（$n = 2$ による Dilation と Erosion の縦列処理）では 0.65（秒／1 画面）となった．ここに使用した計算機はＩＢＭ－ＰＣ系（i486-33MHz），及びＧＣＣコンパイラを用いている．これらの結果からパソコン上にプログラムした場合でもアルゴリズム 3 を用いて高速化し得ることが確認できたが，本来アルゴリズム 2，3 は専用ハードウエアに実装

することを目的としており，これらの専用ハードウエアがビデオレート（0.0166 秒／1 画面）での処理を実現することを考えると，ハードウエアによる高速化がいかに大であるかがわかる．

2.8 Chapter2 のむすび

Chapter 2 では 2 値化画像の Morphology 処理を高速化するアルゴリズムと，それを実行するのに適したハードウエアを示した．更にビデオ信号に追従できる試作機を製作し，本手法がビデオ画像のノイズ除去等に対し効果があることが示された．また，Chapter 2 で示したハードウエアは全体として小規模であり，組み合わされた Morphology の演算を縦列化，並列化して行うためには小さなモジュールを共通のバス上に拡張するだけで足るという柔軟性があるため，様々な目的に応用可能であると考えられる．

また Chapter 2 に示したアルゴリズムは構造要素の大きさに対して計算量が独立である特徴があり，構造要素の形状によってはパソコン等の既存の計算機上で実行した場合にも高速化され得ることが示された．しかし現状ではパソコン等でビデオ信号の実時間 Morphology 処理を実現することは処理速度の点からまだ困難であるため高速化のためにはハードウエア化が不可欠となる．また Chapter 2 に述べた方法は扱える構造要素の形状の自由度が高いため画像のフィルタリング以外にも，例えば中空形状の構造要素を用いた形状抽出等のアプリケーションにも応用可能となる．

Chapter 3　Gray-scale Morphology のためのアーキテクチャ

3.1　Chapter3のあらまし

Chapter 3 では画像処理の一手法である Gray-scale Morphology の計算を高速に実行するためのハードウェアと，そのその中で行われる処理の手順について述べる．Chapter 2 で述べた 2 値画像を扱う Binary Morphology に加え，濃淡画像を扱う Gray-scale Morphology にもその有用性は高く，様々な応用が考えられている．Gray-scale Morphology 処理は，２値画像を扱う Binary Morphology 処理の自然な拡張と見ることができ，文字どおり一つ高い次元から Binary Morphology 処理を包含していると言える．これは，Morphology 処理の論理が 2 次元の画像に対するものに限定されるものでなく，３次元以上の情報に対しても自然な定義づけがなされているためとも言える．Chapter 2 に述べた Binary Morphology の場合，その基本演算は AND,OR で記述される論理演算で表現さたが，Chapter 3 に述べる Gray-scale Morphology の場合には，その演算は数値の和と大小比較にその演算を帰着することができる．

ところで，Gray-scale Morphology の場合にも，その演算は 2 次元関数どうしの重畳演算の形となっていることから，この演算を逐次処理型の計算機上で実行する場合，計算コストが大きくなりすぎることは Binary Morphology の場合と共通な問題点として挙げられる．すなわち，原画像の面積を $N \times N$ ，構造要素の大きさを $M \times M$ としたとき，計算量 (論理演算とメモリアクセスの総数) は，$O(N^2M^2)$ となる．このため大きな構造要素と高速処理を同時に実現することが困難となる．

そこで Gray-scale Morphology 処理においても，その高速化のため特に図 3.1 のような 3×3 のマスク処理を高速に実行するための専用プロセッサが考えられている．またそのようなプロセッサのために，大きな構造要素を小さな構造要素に効率的に分割する手法も考えられている．一例として Cytocomputer [6] では，このようなプロセッサを縦列に複数接続することにより，3×3 よりも大きな構造要素による計算を実現している．また，大きな構造要素を小さな構造要素に効率良く分割する手法は，Gray-scale Morphology においても研究されている [9].

このような 3×3 のマスク処理をベースとする計算機（「Cytocomputer 型アーキテクチャ」と呼ぶ）によって Gray-scale Morphology 処理を実現する場合の問題点として挙げられることは，計算の論理が Binary Morphology ほどには単純ではないことである．すなわち，Gray-scale Morphology では，構造要素に $M \times M$ 個の画素が含まれるとき， $M \times M$ 個の Gray-scale 値（輝度）の中から最大あるいは最少の値を選び出す計算が必要となり，これを組合せ回路として直接的に実現するには，図 3.2 のようにマグニチュード・コンパレータを含む比較器を $log_2(M^2)$ 段にわたって重ね合わせる必要があるためである．このため Threshold decomposition 法 [4] や Hybrid アーキテクチャといった [27]，ゲート段数削減に効果的な手法も考えられているが，それらは回路規模が大きくならざるをえない（これらの計算機の比較を Section 3.8 で詳しく行う）.

そこで本研究では，Gray-scale Morphology に用いる構造要素を分割する段階に立ち返り，大きな構造要素を 3×3 構造要素ではなく，1 次元の構造要素に分割することにより計算の高速化を図ることを目的とした．Chapter 3 では Cytocomputer 型アーキテクチャをベースとした計算機とは異なったアーキテクチャを提案し，小さい回路規模のハードウェアで高速な Gray-scale Morphology 処理が実現できることを示す．またこのアーキテクチャの実効性を検証するための試作機も製作した.

ここで，Chapter 3 で提案する手法は以下のようにまとめられる.

1． 一般に多くの画像処理では，数画素四方のマスクを画像上にラスタ走査することで，画像全体の処理を実行するが，Chapter 3 における考え方では 1 次元の構造要素を用いて水平・垂直のそれぞれ方向に対して走査すること

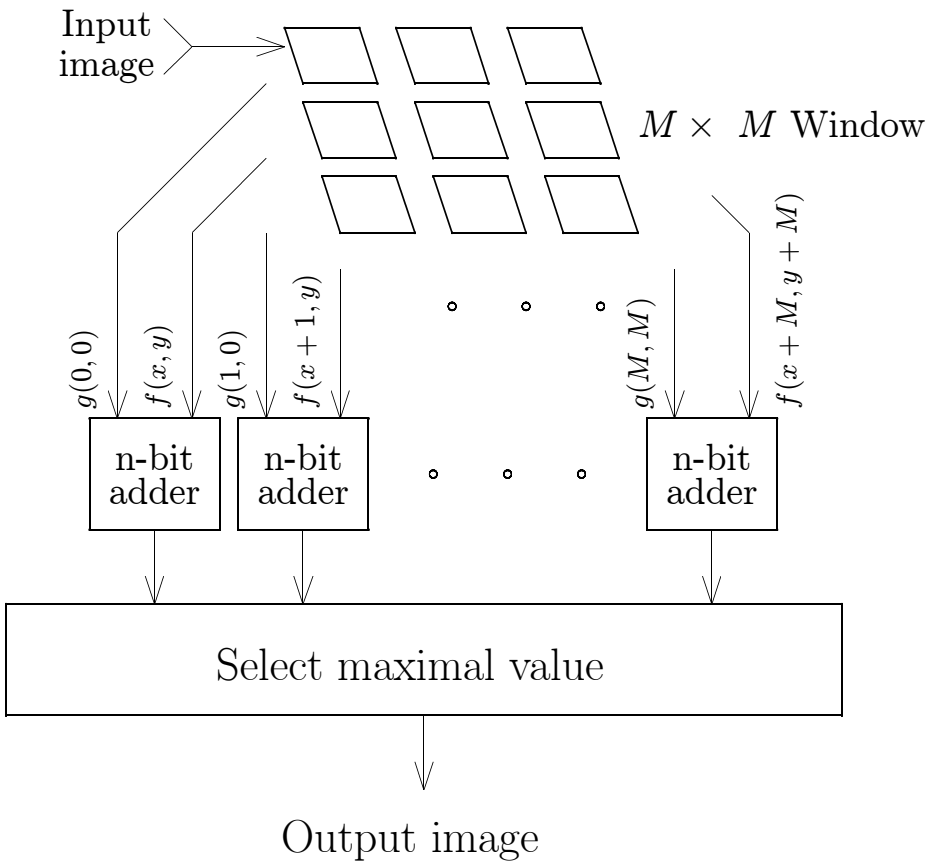


図 3.1: Cytocomputer 型アーキテクチャ

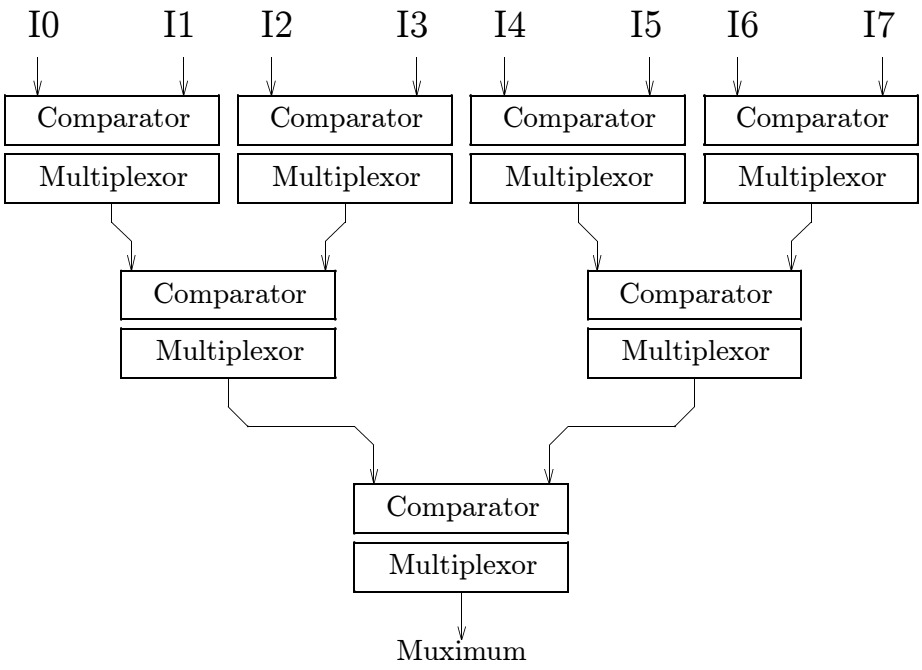


図 3.2: 最大値を得るためのロジック

によって処理を行う．

２． ２次元の Gray-scale の構造要素を，縦･横に配置された２つの１次元の構造要素の Dilation の形となるように分解する手法を用い，２次元の Morphology 演算を１次元化し，上述１の手法に適用する．

３． Gray-scale Morphology のアルゴリズムを並列化し，それを実行するのに適したパイプライン式のプロセッサを考案する．

このようなアプローチを取ることにより，パソコン等の汎用的な計算機に比べ，Gray-scale Morphology 演算の処理速度の格段の向上が可能となった．またこの計算機はハードウェアの規模が小さくて済むことが大きな特徴であり，Section 3.8 で，Cytocomputer をベースとする計算機との比較を，ゲート段数，ゲート個数において行う．

3.2 Gray-scale Morphology

Chapter 2 同様，Chapter 3 においても Haralick が用いている表記方法をにならって Gray-scale Morphology 演算を定義する．本来は Morphology 演算を N 次元ユークリッド空間上の集合どうしの演算として定義されるものであるが，Chapter 3 では計算機の論理としての見通しをよくするために，その部分集合である１次元もしくは２次元の離散空間上での演算に限定して用いることにする．

3.2.1 Gray-scale Morphology の基本演算

Gray-scale Morphology の演算を表現するのに用いる記号は Binary Morphology のそれとほとんど同じと言えるが，その論理は AND，OR で表現するよりも，数値どうしの和，差，および複数の数値の最大，最小と言った演算を用いて Gray-scale Morphology を表現すると扱いやすいものとなる．このため Chapter 3 においても再度，ここで用いる記号や式について Gray-scale Morphology にマッチする形で

定義しなおすことにする．なお，Binary Morphology と Gray-scale Morphology の定義はなんら矛盾するものではなく，Binary 画像の Top,Umbra という概念を導入することにより Binary Morphology の自然な拡張として定義付けられている [23].

図 3.3 は画像の Gray-scale Morphology の例である．本論文においては図 3.3 のように，Gray-scale 画像を，輝度を上向きのベクトルとする 3 次元ワイヤフレームとして表現している．Gray-scale Morphology の場合にも Binary Morphology の場合と同様，Dilation , Erosion , Closing , Opening を定義しているが，そのとき用いる構造要素は一般に Gray-scale 画像である．

図 3.4 は 1 次元信号に対する Gray-scale Morphology の基本的な演算の説明である．ここに，図 3.4(a)(b) に示す $f(x)$ および $k(z)$ はそれぞれ Gray-scale の原画像と構造要素を表しており，変数 x, z は原画像および構造要素の座標（整数値）であり，F および K はそれぞれ，原画像と構造要素の定義域を表している．図 3.4(c)，(d) は Morphology 演算の基本となる Gray-scale Dilation , Erosion であり，それぞれ，式（3.1）,（3.2）で定義される．これらの式は 1 次元であっても 2 次元であっても同様に定義される．

`Dilation`

$$(f \oplus k)(x) = \underset{\substack{x-z \in F \\ z \in K}}{max}\{f(x-z) + k(z)\} \tag{3.1}$$

`Erosion`

$$(f \ominus k)(x) = \underset{\substack{x+z \in F \\ z \in K}}{min}\{f(x+z) - k(z)\} \tag{3.2}$$

Binary Morphology の場合と同じく Gray-scale Morphology においても，Erosion と Dilation は双対な演算のため，式 (3.3) に示すように Dilation を用いて Erosion の計算をすることが可能である．

双対性

$$(f \ominus k)(x) = ((f^c \oplus \check{k})(x))^c \tag{3.3}$$

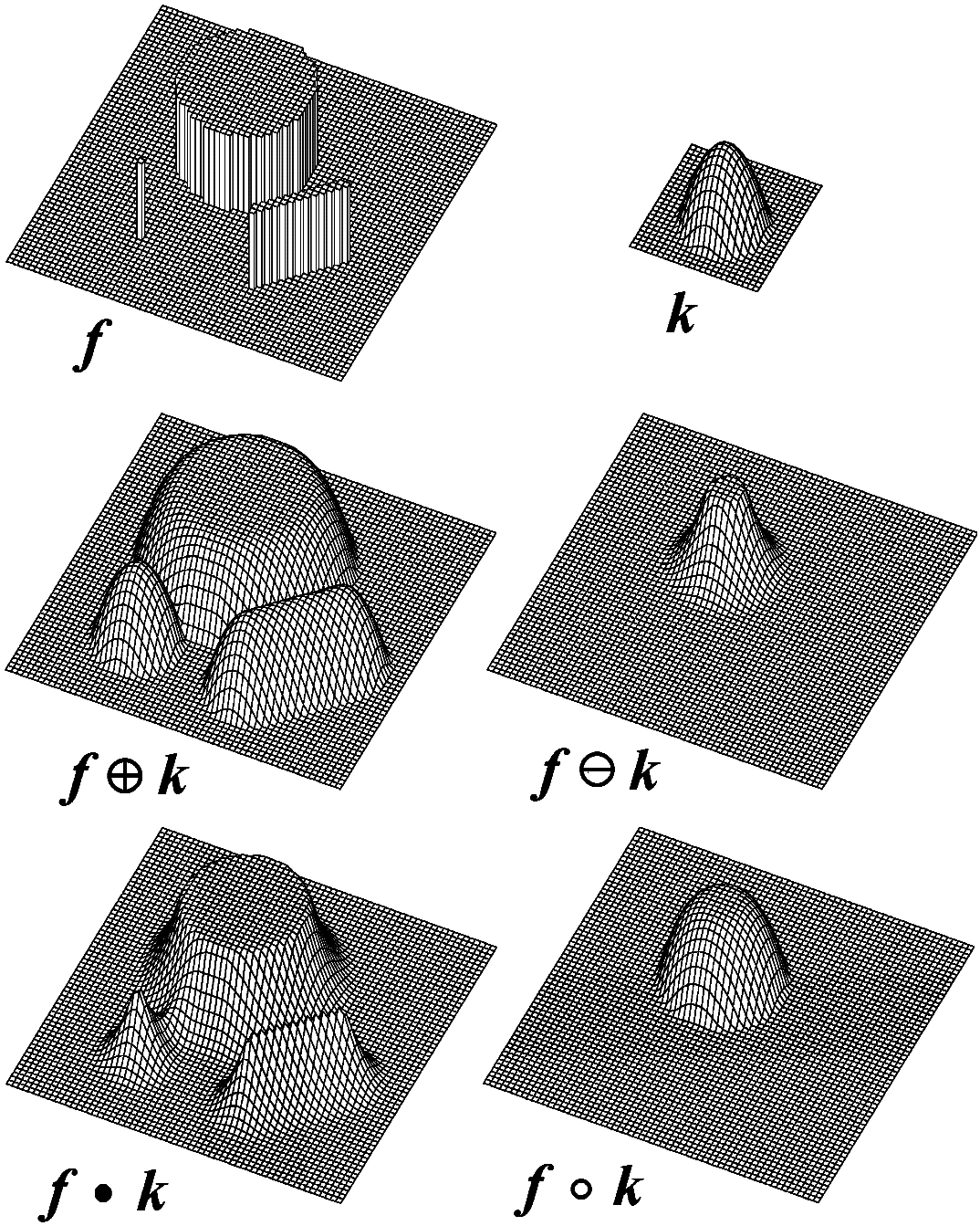


図 3.3: 画像の Gray-scale Morphology 処理

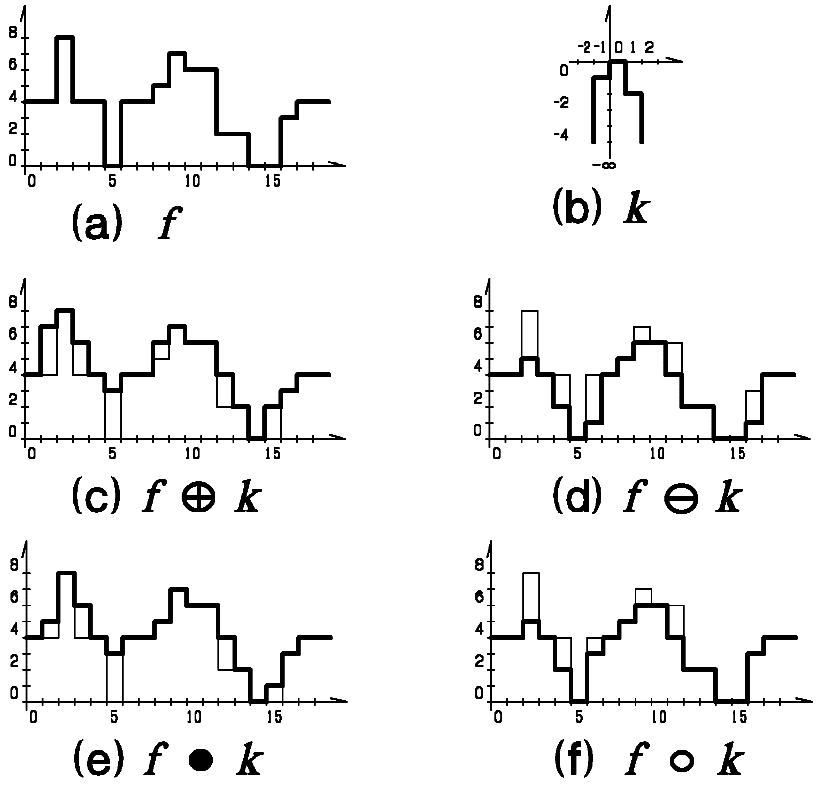


図 3.4: 1 次元 Gray-scale 信号の Dilation と Erosion

ここに，f^c は f の補数（Gray-scale のネガに相当する）であり，$\check{k}$ は k の反射（座標の反転）を表す．$f^c(x)$ と $\check{k}(z)$ は それぞれ $-f(x)$ と $k(-z)$ のようにかかれることがあるが，この表記はむしろ本質をよく反映している．

さらに式 (3.4) に示す結合則と式 (3.5) に示す分配則が成り立ち，この性質は複数の構造要素を重ねあわせるときに用いられる．

結合則

$$f \oplus (k_a \oplus k_b) = (f \oplus k_a) \oplus k_b \tag{3.4}$$

分配則

$$f \oplus (k_1 \cup k_2) = (f \oplus k_1) \cup (f \oplus k_2) \tag{3.5}$$

ここに ”∪” は Gray-scale 画像どうしの論理和であり，式 (3.6) のように 2 つの画像を入力としたとき，その座標においてどちらか輝度値の大きい側を出力する演算として定義される．

$$(f_1 \cup f_2)(x) = larger(f_1(x), f_2(x)) \tag{3.6}$$

3.2.2 本論文で扱う構造要素

ハードウェア構築のため，本論文では取り扱う画像の輝度は有限の正の整数に限定している．このため本論文で用いる Gray-scale 構造要素は他の論文で用いられるそれと比べてその作り方に（癖ともいえる）特徴がある．これは，Gray-scale Dilation ，Erosion の定義式 (3.1) , (3.2) において演算結果の範囲 (最大値，最小値) が $f(x)$ 及び $k(z)$ によってどのように決まるか，ということに注意しなければならないことによる．

どのような場合に問題が起こるのかを例えて言えば次のようになる．いま，足し算と引き算の際，答えの取りうる範囲を 0 ～ 10 でクリッピングする場合を考えてみる．この場合 $7+6=10$ となるが，これは計算結果が 10 を越えているのでこの限りにおいては正しい．しかし，$7+6-5$ の計算をする際，$7+6$ を先に

計算してしまうと $10-5=5$ がその答えとなるが，本来ならその答えは”8”となるはずである．しかも”8”は答えとして許される範囲に収まっているわけだから，計算過程におけるクリッピングの影響だけを受けるのは「おかしい」という考え方もできるわけである．

通常の Gray-scale Morphology の定義では Dilation ， Erosion の計算結果は入力信号及び構造要素と同じ次元のユークリッド空間内に存在することのみを保証しているが，計算機による計算を行う段階では，計算結果のオーバーフローの可能性について事前に検討する必要がある．ここでいうオーバーフローとは，計算機が取り扱い可能な Gray-scale 値の範囲（最大輝度，最小輝度）を越えた計算結果が生ずることを意味している．もし，Dilation ， Erosion においてオーバーフローした場合，計算結果を取り扱い可能な数値の範囲で単にクリッピングしたとすれば，Closing ，Opening 演算の結果は，上の足し算・引き算の例に示したのと同様な不具合を生ずる．つまり計算結果の輝度値自体は本来オーバーフローしないにもかかわらず計算過程におけるクリッピングの影響のみを受けることになる．

この問題に対するもっともシンプルな回避策は，構造要素のとる値の範囲をあらかじめ限定することによってオーバーフローを未然に防ぐことであろう．実際上の手続きは図 3.4（b）に示したように，Gray-scale 構造要素の最大値をゼロに固定し，その他の輝度値を負にすることによって Dilation ， Erosion 双方の演算に対してオーバーフローを防ぐ方法をとる．そのような構造要素の限定によってオーバーフローが防げる理由は以下に説明するとおりである．

入力信号 $f(x)$ が $x \in F$ 内の x_{max} で最大値 f_{max} をとり，$k(z)$ が $z \in K$ で最大値 k_{max} をとったとすれば式（3.1）より $(f \oplus k)(x)$ は x_{max} において最大値 $f_{max}+k_{max}$ をとる．同様に $f(x)$ が $x \in F$ 内の x_{min} で最小値 f_{min} をとり，$k(z)$ が $z \in K$ 内で最大値 k_{max} をとったとすれば式（3.2）より $(f \ominus k)(x)$ は x_{min} において最小値 $f_{min}-k_{max}$ をとる．これら 2 つの事実から，もし $k_{max}=0$ であれば $f \oplus k$，$f \ominus k$ の最大値，最小値は $f(x)$ と同じ f_{max}，f_{min} で押さえられることになる．つまり $k_{max}=0$ としておくだけで Dilation も Erosion も入力信号 $f(x)$ の取りうる範囲を越えないことになるので，計算結果にオーバーフローは生じないことになる．

このようにして構造要素の輝度値のとる範囲を限定してオーバーフローを避け

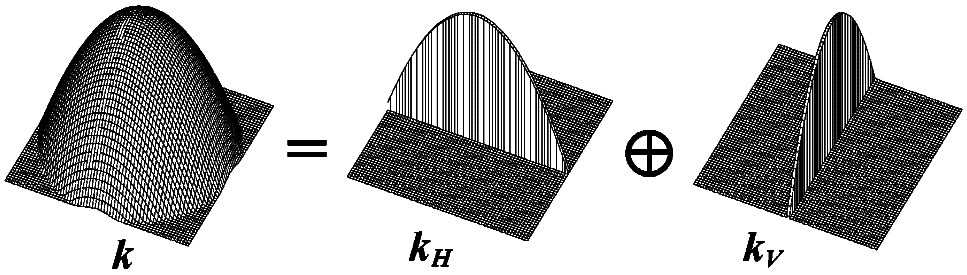


図 3.5: 構造要素の１次元分割

ることができる．逆に入力信号の最大値，最小値を限定することによるオーバーフローの回避法も考えられるが，１つの入力信号に対し同じ Morphology の演算を何回も繰り返すことを想定した場合，入力信号が限定される範囲を決定することが難しくなるため，構造要素の限定による回避策の方がより現実的な方法であると言えよう．

3.3 構造要素の１次元化とそれに伴う制限

3.3.1 １次元化可能な構造要素

Chapter 3 で述べる Gray-scale Morphology 演算に用いる構造要素はそれ自体が２次元の Gray-scale 画像であり，例えば図 3.5 中の k のように，ある３次元的な形状を持っているものとする．ここで仮にこの構造要素が，図 3.5 に示すような２つの直交する１次元構造要素 k_H，k_V の Dilation で表すことが可能であれば，k による Morphology 処理は１次元上のデータ列上の計算として実行することができる [12]．このことは式（3.4）によって裏付けられ，画像 f の k による Dilation 演算は式（3.7）に示すような直交する２つの１次元構造要素による Dilation の縦列接続で実行することが可能となる．

$$f \oplus k = f \oplus k_H \oplus k_V \tag{3.7}$$

この性質を画像処理で用いるには図 3.6 に示すように，画像データのラスタ走査を２回に分ける方法をとる．このときそれぞれの走査の方向は直交したものと

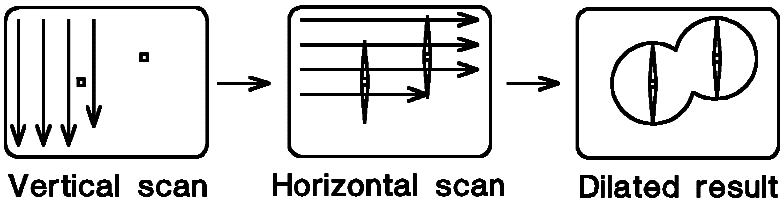


図 3.6: 直交する走査による画像処理

し，それぞれの 1 次元構造要素で Dilation を実行すれば，2 次元構造要素 k による Dilation と等価な演算を，$O(N^2M)$ の計算量で実行することが可能となる．但し，式（3.7）を成り立たせるような 1 次元分解可能な構造要素は，Additively separable と呼ばれる特殊なクラスに属する関数である [12]．Additively separable とは，例えば図 3.7 に示すように，2 次元座標上に配置された数列を式（3.8）に示すような 1 次元関数同士の和によって生成可能であることを意味する．このためすべての形状の構造要素に 1 次元化手法が適用可能ではない．ただし他の手法で適用不可能な形状の構造要素も存在するため，本手法で適用可能な構造要素の形状のカテゴリーを Section 5.5 において他方式と比較しながら述べる．

`Additively separable function`

$$k(x, y) = h(x) + v(y) \tag{3.8}$$

3.3.2 Morphology 画像処理の回転不変性

ところで，フィルタリング等の一般の画像処理に Morphology 処理を用いる場合，構造要素がなめらかな凸形状であることや，画像処理の演算が移動・回転に対して不変であること (Shift invariant, Rotationally invariant) であることが，フィルタの特性として好ましい場合が普通である思われる．ここで言う「回転不変性」と「移動不変性」という 2 つの性質はフィルタ理論の表記式を用いると式（3.9），(3.10）のように定式化できる．

移動不変性

$$\Psi(X_t) = (\Psi(X))_t \tag{3.9}$$

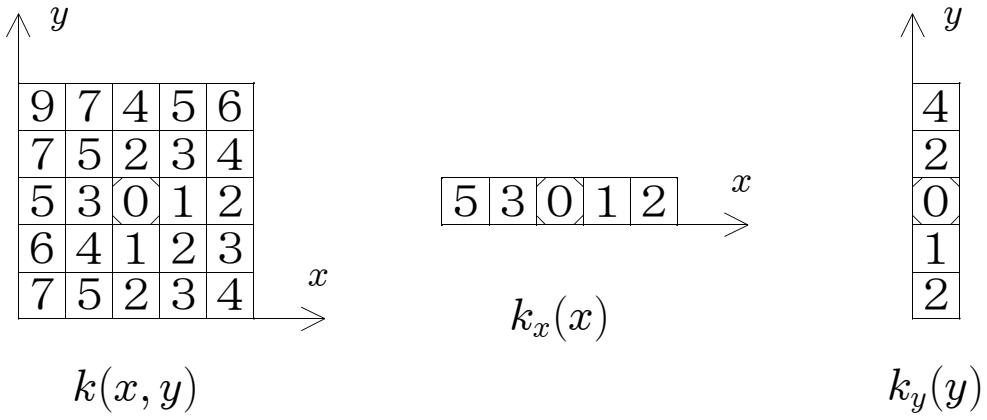


図 3.7: Additively separable の例

回転不変性

$$\Psi(X_r) = (\Psi(X))_r \tag{3.10}$$

ここに X は処理される原画像，下付きの t,r はそれぞれ画像の平行移動と回転であり，$\Psi(X)$ は X のフィルタリング後の画像を表す．すなわち式 (3.9) , (3.10) は原画像に対し平行移動 (回転) を施したものにフィルタΨを作用させた画像と，原画像にあらかじめフィルタΨを作用させてから平行移動 (回転) を施した画像とが等価となる性質を表している．

Morphology 演算は画像の場所によって構造要素を変えるといったことをしないかぎり，もとより移動不変な演算である．そして Morphology 演算が「回転対称な演算であること」は「構造要素が回転対称な形状を持つこと」と等価である [28]．そのような「回転対称な形状」かつ本手法で必要な「１次元化可能」という２つの条件を同時に満たす構造要素は２次曲面 (paraboloid) の構造要素のみとなり [13]（この理由は Appendix A に述べる），しかもなめらかな凸曲面を有しているという意味でも使い易い構造要素と言え，本手法においても特に重要である．

3.3.3 １次元化不可能な構造要素に対する方策

１次元化不可能な構造要素の形状に対する方策をここでは２つ述べる．

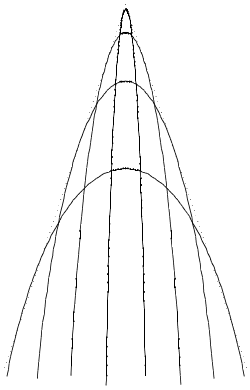


図 3.8: Additively separable な構造要素の重ねあわせ

一つはP.D.Gader[12] が提案したGray-scale 構造要素の近似手法である．式（3.7）の形で表すことが不可能な構造要素は厳密には 1 次元分割不可能であるが，その手法により任意の構造要素を最小自乗誤差で 1 次元化可能な構造要素に近似することができる．主として凸型形状の構造要素に対して適用され，楕円球のように良く近似できる場合もあるが，構造要素の曲面の次数によっては近似が大幅ずれることがある．

もう 1 つの方策として考えられる手だては，複数の 2 次曲線構造要素の重ねあわせによって目的とする構造要素の近似を行う方法である．すなわち構造要素の形状を少しずつ変化させながら分配則（式（3.5））を適用して，複数の Additively separable な構造要素のずらし重ねによって目的とする構造要素に近づける方法である．そのようにして得られた構造要素の断面形状を図 3.8 に示す（点線がもとの構造要素，実線が近似した構造要素である）．この方法では構造要素の回転対称性を保つことはできるが，一方欠点として，重ね合わせる構造要素の個数に計算量が比例すること，また滑らかな構造要素の表面を得ることが難しくなることがある．

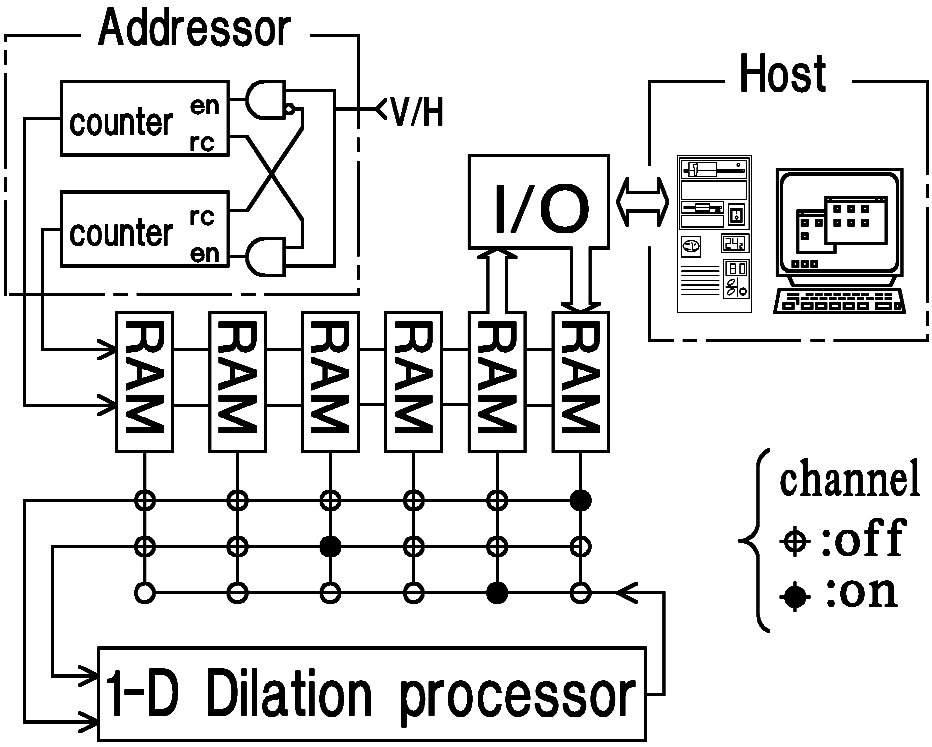


図 3.9: 試作機のアーキテクチャ

3.4 Gray-scale Morphology 処理のためのアーキテクチャ

以上に述べた 1 次元化された Morphology 演算を高速に実行するためのハードウェア (試作機) の概略図を図 3.9 に示す．図 3.9 中において，RAM(16bit Address Bus, 8bit Data Bus) はそれぞれ 256×256 画素，256 階調の画像を取り込むことができ，クロスバースイッチを介してすべての RAM が後述のパイプラインプロセッサとのデータの受け渡しを可能としている．RAM のうちの 2 つはホストとのやり取りをするためのインタフェースとして用いるようバスが拡張されている．

図 3.9 中の”Addressor”は 2 個の 8bit カウンタの rc(ripple carry) 端子と en(enable) 端子を交換することにより，RAM に与える 16bit アドレスの上位と下位を入れ替え，画像の走査方向の縦横変換を可能としたものであり，その詳細は図 3.10 の

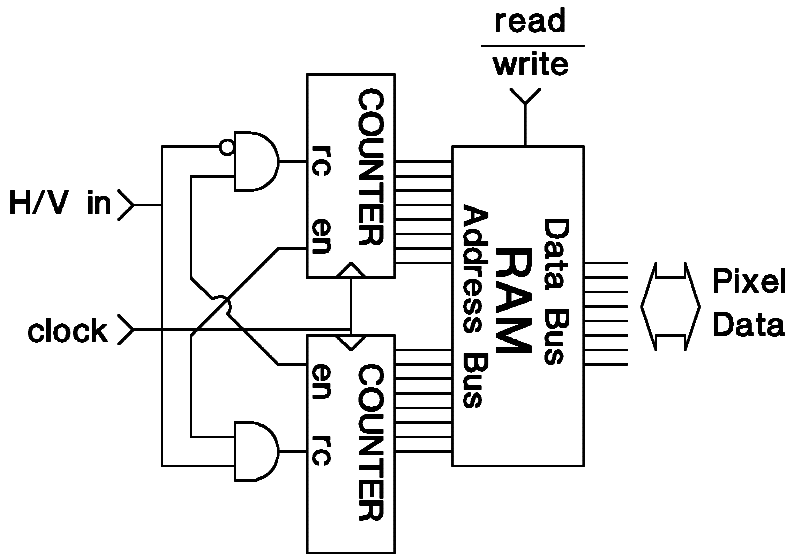


図 3.10: Addressor の詳細

ようになっている．これは，Chapter 2 においてに示したマルチプレクサを用いた「H-V 走査方向変換回路」（図 2.5）にくらべて，アドレスバスの信号の切り変わるタイミングが Clock に同期するため，RAM にとってタイミング的に有利なアドレス信号を少ない素子で作ることができるという利点がある．この回路では Clock が共通な 2 つの 8bit カウンタの ripple carry（rc）端子と enable（en）端子を外部入力（HV in）によって入れ替えることにより RAM の 16bit アドレスの上位と下位を交換して，画像の走査方向の縦横を換えている．図 3.9 中のパイプラインプロセッサは 1 次元 Dilation の演算を行うための計算機であり，Section 3.5 にその詳細を説明する．

3.5 1 次元 Dilation プロセッサ

式（3.1）の計算方法は，ある画素に着目してその周辺の既知である画素（原画像）から Gray-scale 値を拾い集めて計算する形になっており，これはいわば Dilation の畳み込み演算としての表記と言うことができる．この演算はソフトウエア的に実行するのであれば定義式そのままでも問題ないが，ハードウェアに置

for($x := 0$ to x_{max}) — loop 1
{ for($z := 0$ to $M-1$) — loop 2
{ $m(z) := larger(f(x)+k(z), m(z+1))$
}
$(f \oplus k)(x) := m(0)$
}

図 3.11: 一次元 Dilation のアルゴリズム

き換える際には，多入力の最大値を得るためのロジックを実装しなければならず，これには図 3.2 に示した方法やそれを改良した方法（Section 3.8 に述べる）が考えられているが，いずれも簡単なものではない．

このため本研究においては，Dilation 演算を 1 次元化することによる計算量のオーダの減少に加え，1 次元空間上の Dilation 演算ならではの効率的な計算方法とそれを実行するためのハードウェアを提案する．

Gray-scale Dilation の定義式である式 (3.1) は図 3.11 に示すアルゴリズムに置き換えることができる（この置き換えができる理由は Appendix B に述べる）．図 3.11 中において $m(0)$〜$m(M)$ は計算のために用いる中間変数の配列で，z の範囲である $0 \sim M-1$ は，式 (3.1) 中の構造要素の領域 K を具体的に表したものと言える．図 3.11 中の $larger(A,B)$ は式 (3.11) で定義され，入力された値 A, B のうちのどちらか大きな値を返す関数である．

$$larger(A,B) = \begin{cases} A & (A > B) \\ B & (A \leq B) \end{cases} \tag{3.11}$$

ここで図 3.11 に示したアルゴリズム中の loop2 を展開すると，$f(x)$ を逐次入力することにより，それに伴って $(f \oplus k)(x)$ が出力されるような構成のパイプライン型の信号フロー (図 3.12) を得る．このことは，図 3.11 の loop2 において中間変数配列 $m()$ の内容が順次渡されていくことからこのようなパイプライン化が可能であることがわかる．図 3.12 中で $k()$ は，構造要素の Gray-scale 値を一時的に格納するレジスタ，[D] は時系列信号のラッチ，[L] は $larger$ 関数の働きをする計算素子，そして［+］は加算器であり，どの素子も実際のロジック回路として

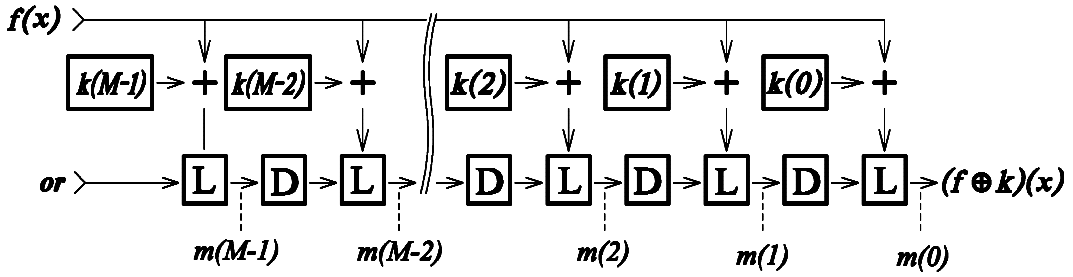


図 3.12: 1 次元 Dilation 演算の信号フロー

容易に実現可能である．各素子はこの信号フローのように接続されることにより，全体として 1 次元 Dilation が計算される．このように構成された 1 次元 Dilation 回路は図 3.9 において”1D Dilation Processor”として組み込まれるものである．

3.6 パイプライン段数に収まらない構造要素に対する方法

図 3.13 はハードウェアとして実装された Dilation パイプラインの段数よりも大きな 1 次元構造要素による処理を実現する際の具体的な手順を示している．

例えばパイプラインの段数が 16 段のところ，64 画素大の構造要素 k による Dilation を実現したい場合，k をあらかじめパイプラインの段数に収るような k_1 ～k_4に分割し，式 (3.5) を適用することにより，式 (3.12) で示される計算を逐次実行して，分割前の構造要素による Dilation と等価な結果を得る．

$$
\begin{aligned}
& f \oplus k \qquad (3.12)\\
&= f \oplus (k_1 \cup k_2 \cup k_3 \cup k_4)\\
&= (f \oplus k_1) \cup (f \oplus k_2) \cup (f \oplus k_3) \cup (f \oplus k_4)\\
&= ((((f \oplus k_1) \cup (f \oplus k_2)) \cup (f \oplus k_3)) \cup (f \oplus k_4))
\end{aligned}
$$

式 (3.12) における”∪”演算は，式（3.6）で定義した 2 つの 1 次元信号の Gray-scale 値の内のどちらか大きい方を結果として返す演算であるが，この演算は Dilation パイプラインを利用して Dilation 演算と同時に実行することができる．例

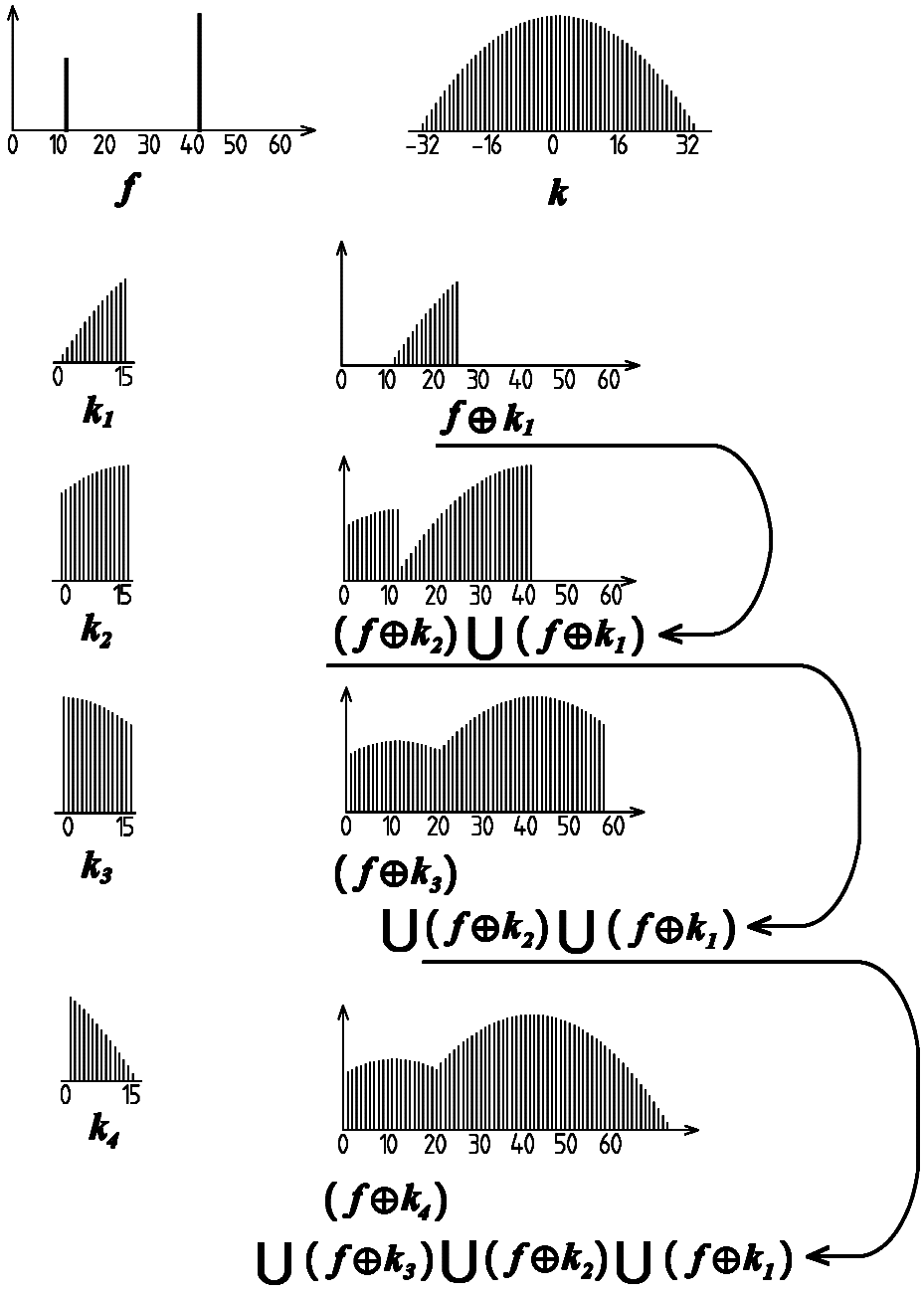


図 3.13: 分割された構造要素による Dilation

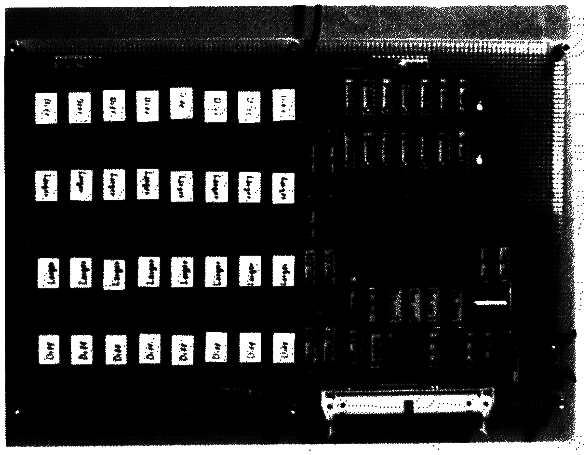


図 3.14: 試作機の概観

えば，図 3.13 の第一段階を使って説明すると，まず図 3.12 中の構造要素レジスタに，k_1を格納し，$f(x)$ との Dilation を求め，結果の $f \oplus k_1$を RAM に格納する．次に構造要素レジスタに k_2を格納し，$f(x)$ との Dilation を求めるが，このとき同時に図 3.12 中の”or”入力に前回求めた $f \oplus k_1$を $f(x)$ と同時に入力することにより，$(f \oplus k_1) \cup (f \oplus k_2)$ の結果を得る．以下同じ手順を繰り返すことにより，式 (3.12) に示したように再帰的に計算結果が得られる．この *or* 入力を用いる方法は Dilation の演算が含む大小比較の部分だけを利用したにすぎないが，”∪”に相当する演算を別途必要としないという利点がある．

なおこのような構造要素のずらし重ねをハードウェアで実現するには，図 3.10 に示したカウンタ回路にオフセット機能を付加する必要がある．そのとき，画像の read 側 RAM のカウンタと，write 側 RAM のカウンタとを独立させることにより，それぞれのカウンタに対し別のオフセット値を与えて，任意の画像シフトを実現することができる．

3.7 試作機による評価

ここで示した手法の有効性を検証するため，Chapter 3 に提案したアーキテクチャに基づく試作機 (図 3.14) を製作した．以下に実際に行った画像処理における諸元を示す．用いる画像は，大きさ 256 × 256 画素，輝度の深さ 256(8bit)Gray-scale ，構造要素の大きさは 64 × 64 画素である．この試作機ではパイプラインの段数が 16 段であるため図 3.13 に示した構造要素分割（4 分割）を行っている．この処理において試作機による実際の Dilation の計算時間は 0.18 秒であった．ここで，この計算時間は式 (3.13) のように表すことができる．

$$T_p = N^2 \times m_p \times C \times 2 \tag{3.13}$$

ここに N^2は画像の大きさの 画素数，m_p は 1 次元構造要素の分割数，C はマシンサイクルであり，本手法では縦と横に一回ずつ走査する必要があるため全体を 2 倍している．この試作機ではそれぞれ，$N = 256$，$m_p = 4$，C=250ns であるので，式 (3.13) からは T_p =0.13 秒となるが，実際には各種のコマンドを発行するための I/O 命令や，画像を 1 ライン走査するたびにパイプラインを初期化するといったオーバ・ヘッドが生ずるだめ，実際の処理時間は T_pよりも大きくなる．

この計算時間を短縮するためには，m_p を減少させること，すなわちできるだけ長いパイプラインを作ることである．C 自体は Section 3.8 で述べるゲート段数の議論から数 10ns を目標に設計することは可能であるので，このアーキテクチャでのビデオ信号の実時間処理も充分に可能と思われる．

参考のため同じ画像に対する同じ演算（1 次元構造要素を用いた Dilation 演算）をパソコン (IBM-PC 系 P5-90) による処理時間と比較を行ったところ，パソコンによる計算時間が 6.9 秒であるので，試作機が 40 倍程度高速であった．ソフトウエア的に実行する場合には，1 次元化された構造要素を使うことによるメリットは享受できるが，計算過程をパイプライン化したことによるメリットは生じないといえる．これはパイプライン上のすべての計算を一個の CPU で同時に実行することが現状では不可能であることによる．よって計算時間は原理的に $O(N^2 \times M)$ となることは避けられない．

3.8 アーキテクチャの比較

ここでは，Chapter 3 に述べた 1 次元化する方法と Cytocomputer 型アーキテクチャ，すなわち大きな構造要素を複数の $M \times M$ 構造要素（代表的には 3×3）に分解する手法について計算時間とロジック回路の複雑さについて，アーキテクチャの面から比較する．

3.8.1 計算量の比較

計算時間については，入出力部分を含む全体的なアーキテクチャは，両者とも入力画像の取り込みに同期して動作する計算機であるとみなすことができるため，スループット（単位時間当たりに処理できる情報量）は，基本的には同じとすることができる．ただし，1 次元化する方法では，一旦 1 次元の処理を行った後それをメモリにたくわえ，次に走査方向を変えて読み出すという作業が必要となるため，入力画像と出力画像の間に 1 画面分の遅延（$N \times t_l$秒）が生じることになる．ここに N は画像の 1 辺の長さ（画素数），t_lは画像を 1 ライン走査するのに必要な時間である．

この点において，Cytocomputer 型アーキテクチャでの遅延は，$M \times t_l$ 秒となる（M は構造要素の 1 辺の長さ）．両者について t_lを一定とした場合，その遅延時間は，1 次元化する手法では扱う画像の 1 辺の長さに比例し，Cytocomputer 型アーキテクチャでは構造要素の 1 辺の長さに比例するといえる．構造要素の大きさが原画像の大きさと等しければ，両者の遅延時間は一致するが，一般的には原画像より構造要素の方が小さい場合が多いと考えられ，その場合には，Cytocomputer 型アーキテクチャの方が遅延時間は少ないと言える．ただし Cytocomputer 型アーキテクチャを縦列接続して，それぞれのステージに対し，分割された構造要素を適用した場合 [6] には，分割されたそれぞれの構造要素の 1 辺の長さをすべて足しあわせた値と t_lとの積がその遅延時間となる．この場合，上述の $M \times t_l$が最短の場合の計算時間となるが，一般には任意の形状の構造要素を実現するためにはより多くの構造要素を組み合わせる必要がある [9]．各アーキテクチャにおける取り扱い可能な構造要素のクラスに関しては Section 5.5 で詳しく扱う．

3.8.2 ハードウェアの複雑さの比較

Gray-scale Morphology を実行する回路について，ハードウェアの複雑さを R.Lin らが示している方法で比較する [27]．よって以下では，Cytocomputer 型プロセッサの構造要素部分と Chapter 3 に述べた 1 次元 Dilation プロセッサ部分のみの比較とし，周辺のコントロール回路等は含めていない．用いる尺度は，回路の大きさを表す指標となるゲート個数と，システムの動作速度を決定付けるゲート段数である．ここでゲート個数とは，ある機能を持った論理回路を構成するのに必要なゲート回路（OR,NOT など）の総数である．ゲート段数とは，ゲートがラッチとラッチの間に直列に（最大で）何段つながれているかを示すもので，1 マシンサイクルを実行するのに必要な時間がそれに比例すると言える．

ここでは以下の 4 つの方式について比較を行う．このうち，1，2，3 はいずれも，Cytocomputer 型アーキテクチャであり，1 がその原型，2，3 が $M \times M$ マスクを切り出した後の計算の方式に改良を加えたものと位置付けることができる．

1. 直接法 (D) : 図 3.1 に示したように，式（(3.1)）の中の $M \times M$ 個の $f \oplus k$ の値の中から最大値を得るために，逆 2 分木型につないだコンパレータ（図 3.2）を用いる方法．最大値を得るための組合せ回路としては，もっとも基本的な方法と言える [27].

2. Threshold decomposition 法 (T)：原画像と構造要素をすべての Gray-scale 値で 2 値化し，それぞれに Binary Morphology 処理を実行し，得られた結果を Gray-scale 画像に戻す方法 [4]（図 3.15）.

3. Hybrid 法 (H) : 式 (3.1) において，足し算の部分には加算器を用い，最大値を求める計算には Threshold decomposition 法と同じ組合せ回路を用いる方法 [27]（図 3.16）.

4. 1 次元化法 (1D):Chapter 3 に述べた方法．加算器 (adder) と比較器 (comparator & multiplexor) を図 3.12 のように組み合わせたもの.

以上のそれぞれの方式において，必要となる論理回路の個数と，その論理回路を直列に何段つなぐ必要があるかを列挙したものが表 3.1 である．そして各々の論

理回路のゲート個数及びゲート段数を表 3.2 に示す．そして各方式に関してゲート個数とゲート段数が，構造要素の大きさと Gray-scale の bit 数に応じて，どのように決まるかを示した例が表 3.3 である．このように，１次元化する手法ではいずれの場合においても回路規模を大幅に削減することができ，さらにその回路規模が構造要素の１辺の長さのみに比例することから特に大きな構造要素に対して有効であると言える．また，ゲート段数はパイプラインの段数及び Gray-scale の bit 数に無関係であり，かつ最も少なくできることから，同じデバイスを用いる場合，より短いマシンサイクルに耐えることを示している．さらに付け加えるならば，Binary OR のロジックを含む Cytocomputer 型アーキテクチャでは，Gray-scale の bit 数を多くできないという点が指摘出来る．なぜなら Binary OR には Gray-scale の 深さの個数に相当する入力ラインを持つＯＲゲートが必要となるためである [4]． このような論理回路を実際に構成する場合，8bit（256 入力 OR） までが限界と考えられ，16bit 以上の Gray-scale を扱うことは困難であると思われる．

表 3.1: 各方式で用いる論理回路の個数（上段）と，それぞれの論理回路が直列接続される段数（下段）

	AD	CM	TH	BD	BS	OR
D	$\frac{M^2}{1}$	$\frac{M^2-1}{log_2 M^2}$				
T			$\frac{M^2}{1}$	$\frac{M^2+1}{1}$	$\frac{1}{1}$	$\frac{2^n}{1}$
H	$\frac{M^2}{1}$		$\frac{M^2}{1}$		$\frac{1}{1}$	$\frac{2^n}{1}$
1D	$\frac{M}{1}$	$\frac{M}{1}$				

n : Gray-scale の bit 数
M : 構造要素の 1 辺の画素数（長さ）

AD : Adder(Carry look ahead)
CM : Comparator & Multiplexor
TH : Thresholding circuit
BD : Binary Dilation
BS : Binary Search
OR : Binary OR at each threshold level

表 3.2: 個々の論理回路に必要なゲート個数とゲート段数

論理回路	ゲート個数	ゲート段数
AD	$(n+1)n/2+4n$	4
CM	$2(3n+1)$	6
TH	$2^{n+1}+n-1$	3
BD	$2^{n-1}(2^n-1)$	2
BS	$3\cdot 2^n+n-7$	3(n-1)
OR	1	1

表 3.3: ゲート個数とゲート段数の比較

Bits of gray level : 4

size	D	T	H	1D
3x3	442(28)	1.6K(15)	617(17)	156(10)
4x4	806(28)	2.6K(15)	1.0K(17)	208(10)
5x5	1.3K(34)	4.0K(15)	1.6K(17)	260(10)
20x20	20.8K(58)	62.2K(15)	24.5K(17)	1.04K(10)

Bits of gray level : 6

size	D	T	H	1D
3x3	709(28)	21.5K(21)	1.9K(23)	249(10)
4x4	1.3K(28)	36.6K(21)	3.1K(23)	332(10)
5x5	2.1K(34)	55.9K(21)	4.7K(23)	415(10)
20x20	33.2K(58)	861.8K(21)	71.5K(23)	1.66K(10)

Bits of gray level : 8

size	D	T	H	1D
3x3	1.0K(28)	332.2K(27)	6.3K(29)	354(10)
4x4	1.8K(28)	563.9K(27)	10.4(29)	472(10)
5x5	2.9K(34)	862.4K(27)	15.7K(29)	590(10)
20x20	47.2K(58)	13.3M(27)	235.K(29)	2.36K(10)

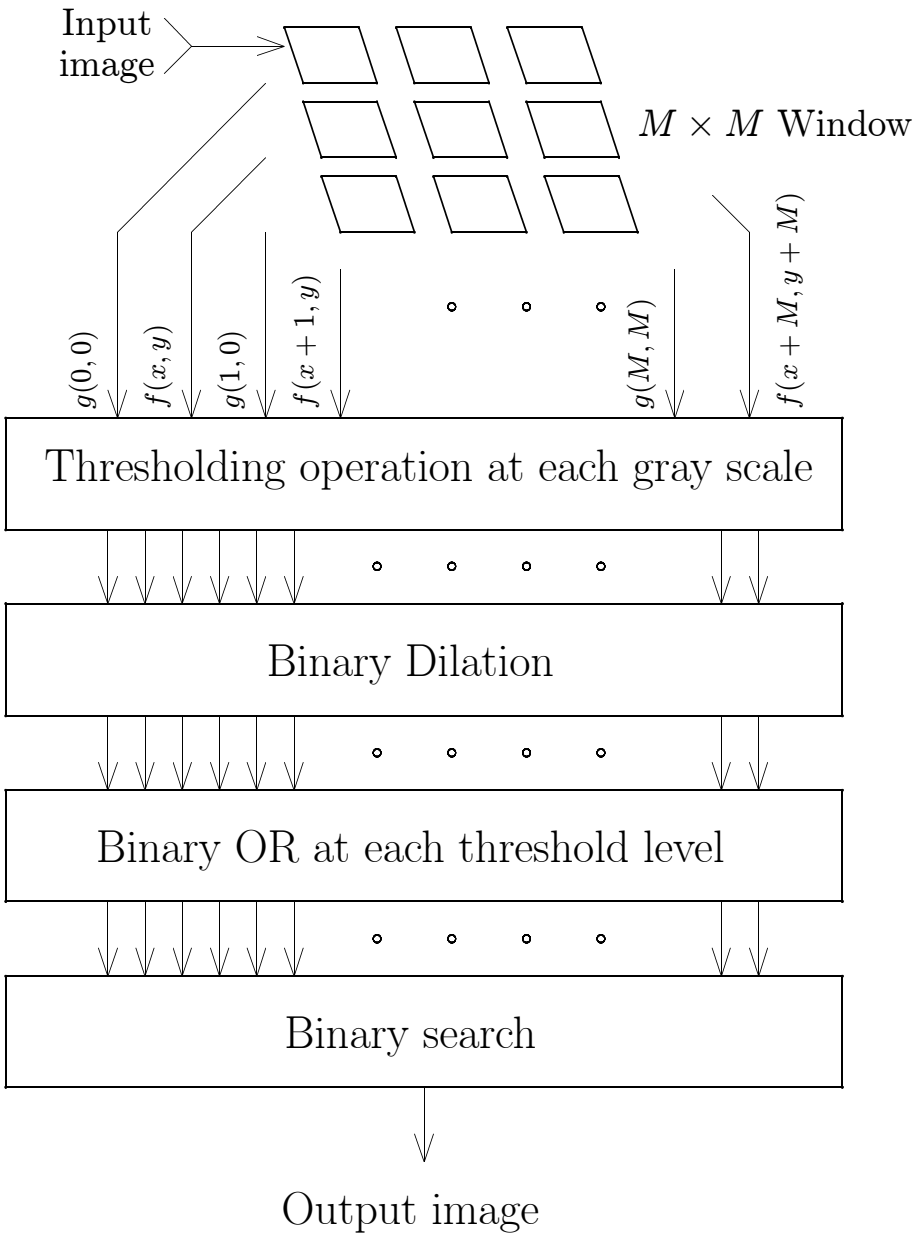


図 3.15: Threshold Decomposition 法

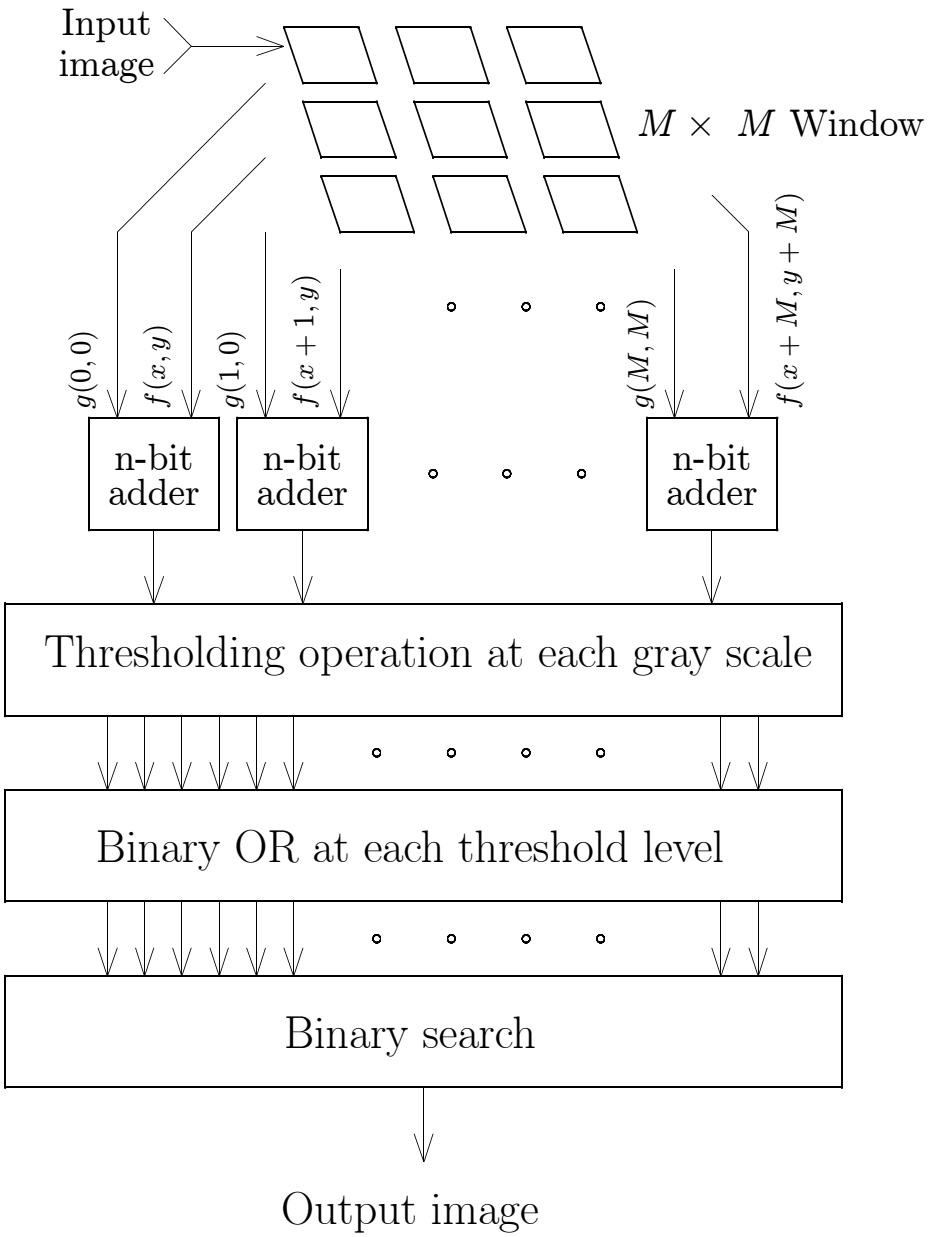


図 3.16: Hybrid 法

3.9 Gray-scale Morphology のフィルタへの応用

ここでは，大きな構造要素を使った Gray-scale Morphology の応用例を示す. 図 3.17 に示した原画像（256×256）は，段差をもった滑らかな画像に十数画素の大きさの白，黒のノイズおよび salt & pepper ノイズを挿入したものである．この原画像に対し，64×64 画素の大きさの paraboloid 型構造要素により式（3.14）〜（3.17）のような処理を行う．大きな構造要素を用いた Gray-scale Morphology により，このような十数画素もの大きさを持つノイズ部分をほぼ完全に消去でき，しかも原画像中の段差をそのまま保持するという特異な特性をもつフィルタを構成することができる．図 3.17 に示したフィルタは以下のように設計したものである．まず，残すべき滑らかな背景の持つ曲率を決め，その曲率にフィットする構造要素を生成する．次に原画像を f，構造要素を k としたとき，f の中に含まれるノイズ部分を式（3.14）に示した方法で検出する．ここでは Openclosing と Closeopening を原画像の輝度と比較する事でノイズ箇所を検出している．次に原画像からノイズ部分を取り除くため，原画像のノイズ箇所を黒レベルに置き換えた $fill$ 画像と，逆に白レベルに置き換えた $cutoff$ 画像の二種類を用意し（式 (3.15) , (3.16)），それぞれに Dilation , Erosion を施し，原画像とノイズ部分の統合を行う（式（3.17)）.

$$noise = \begin{cases} 0 & where\ (f \circ k) \bullet k \leq f \leq (f \bullet k) \circ k \\ 1 & elsewhere \end{cases} \tag{3.14}$$

$$fill = \begin{cases} black & where\ noise = 1 \\ f & elsewhere \end{cases} \tag{3.15}$$

$$cutoff = \begin{cases} white & where\ noise = 1 \\ f & elsewhere \end{cases} \tag{3.16}$$

$$Result = \begin{cases} (fill \oplus k + cutoff \ominus k)/2 & where\ noise = 1 \\ f & elsewhere \end{cases} \tag{3.17}$$

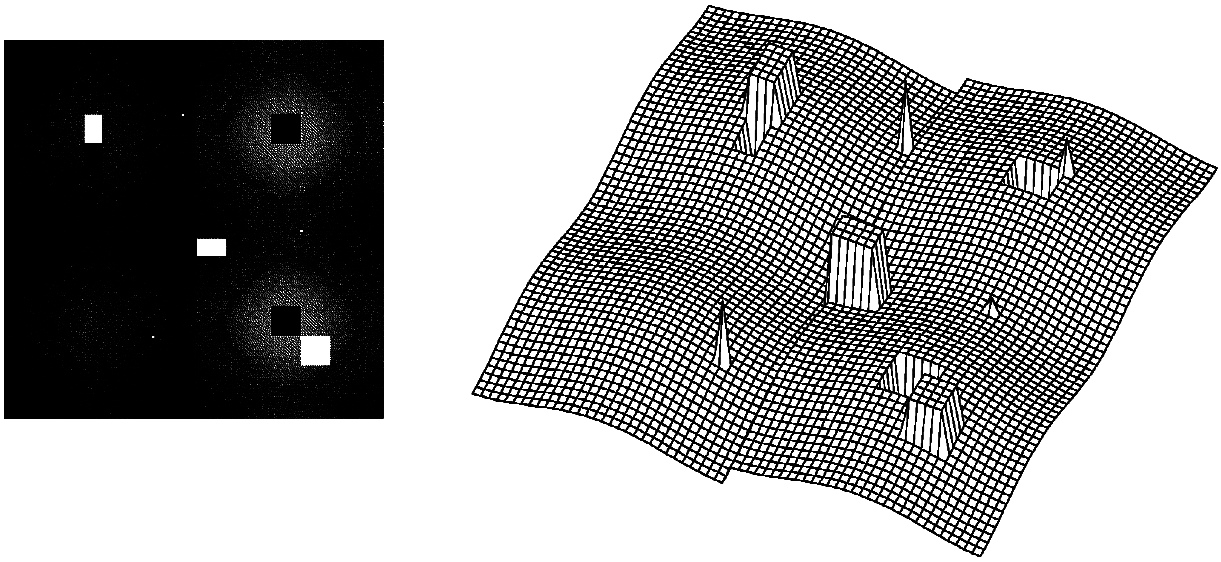


図 3.17: 原画像

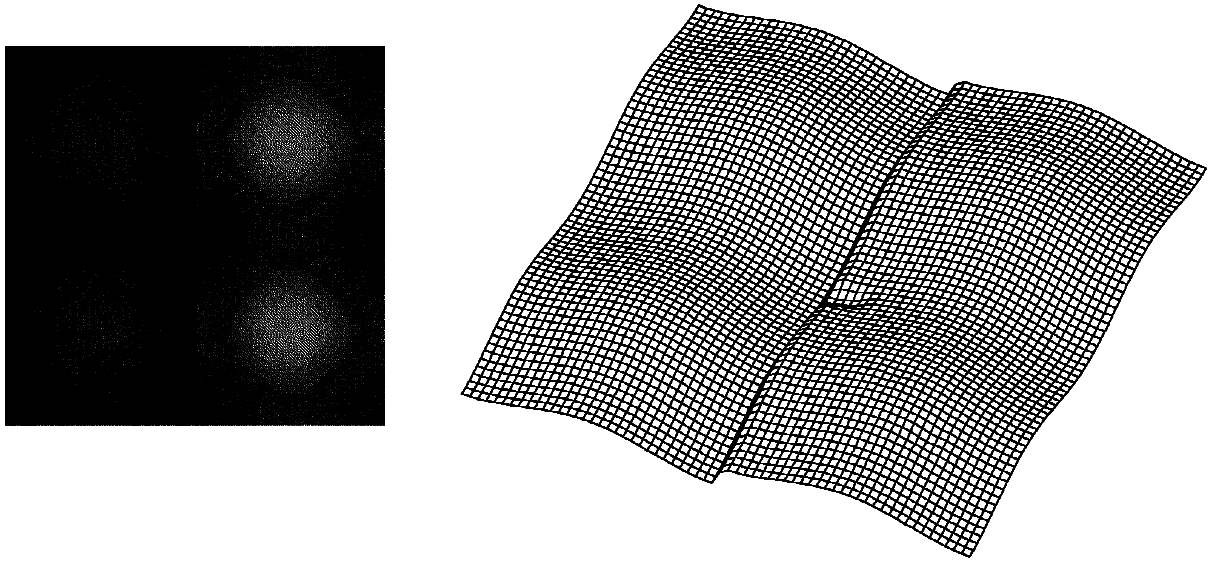


図 3.18: フィルタリング後の画像

3.10 Chapter3 のむすび

Chapter 3 では Gray-scale 画像の Morphology 処理のためのアーキテクチャを示した．Chapter 3 で述べたアーキテクチャの中で重要なのは，画像の走査方向を任意に変更することが可能な画像メモリと，１次元の Dilation 演算を実行するプロセッサである．その１次元 Dilation プロセッサにおけるアルゴリズムを工夫することにより，パイプライン式の信号フローを用いて１次元信号の Dilation 演算を小規模なハードウェアで実現できることを示した．そして１次元化された構造要素をさらに分割する手法により，実装されたパイプラインの段数よりも大きな構造要素による Dilation が可能であることを示した．このアーキテクチャでは，２次元の構造要素を１次元化しておく必要があるため，構造要素の形状が Additively separable に限られるものの，ハードウェア化する際の回路規模及びゲート段数の削減の効果は非常に大きい．さらにこのアーキテクチャに基づくハードウェアを試作し，その高速性が確認できた．このアーキテクチャは Chapter 2 に示した Binary Morphology の１次元処理アーキテクチャの次の段階として，これを Gray-scale Morphology において実現したものと位置付けることができる．

このアーキテクチャにおいては，適用可能な構造要素が Additively separable な形状に限られていたが，この制限を一部解消した，２値画像の距離変換に適するアーキテクチャを Chapter4 に述べる．

Chapter 4　テーブル変換器を用いたアーキテクチャ

4.1　Chapter4のあらまし

Chapter 4 で述べるアーキテクチャは Dilation プロセッサの部分にテーブル変換器を含むロジックを用いることにより，Chapter 3 に述べたアーキテクチャを拡張したものである．さらに，Gray-scale 方向の bit 数を多く取ることにより，１次元化された Gray-scale Morphology 画像処理を高速に実行することに加え，誤差の少ない距離変換に応用できるように考えたものである．

さらにもう一つの特長として，ボロノイ図を求める際に必要となる母点の座標を保持するための工夫もされている．この部分の論理は Morphology の演算の論理とは建前のうえでは無関係であるが，Morphology 演算を計算幾何学的なアプリケーションに適用しようとした場合，ぜひ必要な機能となる．このように Chapter 4 に述べるアーキテクチャは Chapter 3 に述べたアーキテクチャのスーパーセット的なものであり，機能が多くなっている．

ここで Chapter 4 に述べるアーキテクチャの特徴は，以下のようにまとめられる．

１．　２次元の Gray-scale 構造要素を，それぞれ縦・横に配置された２つの１次元の構造要素の Dilation の形となるように分解し，これに Morphology 演算を１次元化する手法を用いて画像処理を行う．

２．　１次元化された Gray-scale Morphology のアルゴリズムを並列化し，パイプライン式のプロセッサとして具現化する．

３．　構造要素を格納するためのレジスタは定数でなく，ここで新たに提案する

「テーブル変換器」を用いることにより，より柔軟な処理を可能とする．さらに扱う画像の Gray-scale の bit 数 を大きく取ることにより，Gray-scale 方向に精度の高い演算を可能とした．例えばこれにより点群の距離変換が可能となる．

4． Dilation 画像の各画素を作り出した構造要素の原点が原画像のどの座標に存在したかを検索するためのインデクス情報を出力画像に付加する機能を設けた．これにより，例えば求めた距離変換画像から，ボロノイ領域のラベリングが可能となる．

Chapter 4 に述べるアーキテクチャは以上のような特徴をもっているが，上記１．２．の部分はすでに Chapter 3 に示したこのアーキテクチャの基本部分であり，３，４が，Chapter 4 において新たに提案する拡張部分と位置付けることができ，section4.2 以降においてその詳細を述べる．

なお Chapter 4 において用いる Morphology の論理は Gray-scale のものであり，これらの定義や定理は Chapter 3 の式（3.1）～（3.6）の通りである．

4.2　テーブル変換器を用いた Dilation 演算

Chapter 3 に述べたように Additively separable な構造要素を用いることにより，Gray-scale Morphology の演算の 1 次元化が可能となるが，この考え方を発展させた手法を以下で提案する．この手法は入力画像が 2 値画像の場合には，非 Additively separable な構造要素を用いることが可能であり，また Additively separable な構造要素を用いる場合には，入力画像が Gray-scale である一般の場合にも適用可能である．このため，本手法は Chapter 3 に述べた 1 次元化手法でできることを含んだうえで更に適用範囲を広げた方法であるといえ，後に述べるようにその応用範囲も広がる．

まず，式（3.1）で示された Gray-scale Dilation の定義式において，式（4.1）のような置き換えを考える．

$$CT_z(f(x-z)) = f(x-z) + k(z) \tag{4.1}$$

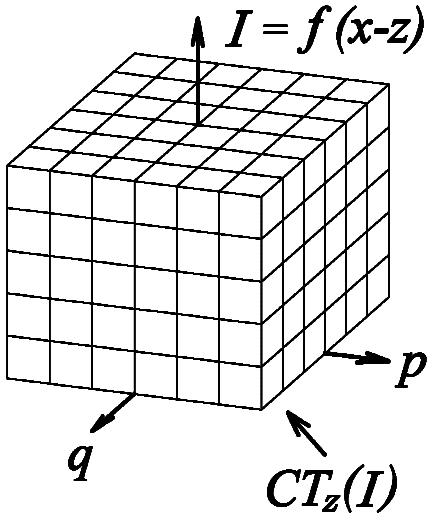


図 4.1: テーブル変換の原型

この置き換えにより，テーブル変換器を用いた Dilation として新たに式（4.2）を定義する．

Dilation

$$(f \oplus k)(x) = \underset{\substack{x-z \in F \\ z \in K}}{max}\{CT_z(f(x-z))\} \tag{4.2}$$

$CT_z()$ は入力された原画像と出力される Dilation 画像とを関係づける変換であり，構造要素の形状と原画像との演算を同時に定義するものであり，式（3.1）に示した演算を含む形で拡張されたものとみなすことができる．式（4.2）の CT_z は構造要素の２次元上の座標を表す変数 z と，現在着目している原画像の輝度値 $f(x-z)$ を入力としてある値を出力する働きをしている．すなわち CT_z は３つの入力値から一つの出力値を得る関数であると解釈できる．このことを模式的に表すと図 4.1 のようになる．図 4.1 において，p, q は構造要素の着目点を表す２次元座標 z の水平・垂直方向成分である．

定義式 （4.2）では，x,z はともに２次元の変数であるため，定義式通りの計算を行うためには図 4.1 に示したような３次元のテーブル変換が必要となる．このようなテーブル変換を用いる場合，Gray-scale の bit 数が多くなった場合，メモ

$CTV_q(I)$						
I		q=1	q=2	q=3	$\cdots$	q=Q
1	$\rightarrow$	v(1,1)	v(1,2)	v(1,3)	$\cdots$	v(1,Q)
2	$\rightarrow$	v(2,1)	v(2,2)	v(2,3)	$\cdots$	v(2,Q)
3	$\rightarrow$	v(3,1)	v(3,2)	v(3,3)	$\cdots$	v(3,Q)
			$\cdots$			
I	$\rightarrow$	v(I,1)	v(I,2)	v(I,3)	$\cdots$	v(I,Q)

$CTH_p(CTV_q(I))$						
$CTV_q(I)$		p=1	p=2	p=3	$\cdots$	p=P
v(1,1)	$\rightarrow$	h(v(1,1),1)	h(v(1,1),2)	h(v(1,1),3)	$\cdots$	h(v(1,1),P)
v(1,2)	$\rightarrow$	h(v(1,2),1)	h(v(1,2),2)	h(v(1,2),3)	$\cdots$	h(v(1,2),P)
			$\cdots$			
v(2,1)	$\rightarrow$	h(v(2,1),1)	h(v(2,1),2)	h(v(2,1),3)	$\cdots$	h(v(2,1),P)
			$\cdots$			
v(I,Q)	$\rightarrow$	h(v(I,Q),1)	h(v(I,Q),2)	h(v(I,Q),3)	$\cdots$	h(v(I,Q),P)

図 4.2: テーブル変換

リを大量に必要とする点や，その計算が原画像に対する２次元の重畳演算となってしまい計算コストが大きくなるという問題点が生ずる．そこで，Chapter 3 に述べた手法を発展させ，テーブル変換器を用いた手法においても直交する２つの変換による画像処理とすることを考える．

まず式 4.1 のテーブル変換を，直交する２つの１次元のテーブル変換に分けるため，式（4.3）に示すように，二段階の変換に置き換える．

$$CT_z(I) = CTH_p(CTV_q(I)) \tag{4.3}$$

$CT_z()$ は原画像の輝度値列 I を入力として２つの変換 CTV_q，CTH_pによって作り出された出力画像と解釈することができる．図 4.2 にこの変換が行われる様子を説明する．このように原画像の輝度値列 I は，いわば種のようなものと見なせ，２段階の変換テーブルである $v(*,*)$，$h(*,*)$ によって２次元に展開される．このときの $v(*,*)$，$h(*,*)$ の決め方しだいで様々な構造要素による Dilation 演算を行うことができる．

4.3 テーブル変換器の利点

Dilation 演算を１次元化する際にこのような変換を導入することによる利点は，２値の原画像に対して，Additively separable でない構造要素による距離変換を実行することが可能になることである．これを実行するには，先に垂直方向 (V) に Dilation を行い，その実行結果に対して水平方向 (H) に Dilation を行う（垂直と水平の順序は逆でもかまわないが以下の説明では先に垂直，後で水平の順としている）．その際 縦方向と 横方向でテーブル変換器の内容を変更することで，Additively separable でない構造要素にも対応可能となる．具体的には，２値画像の距離変換をこの方法で求める場合には（4.4）のようにテーブル変換を定義する．

$$CTV_q(I_V) = \begin{cases} 0 & : \ I_V = 0 \\ d(q,0) & : \ I_V = 1 \end{cases}$$

$$CTH_p(I_H) = \begin{cases} 0 & : \ I_H = 0 \\ d(p,q) & : \ I_H = CTV_q(I_V) \end{cases} \tag{4.4}$$

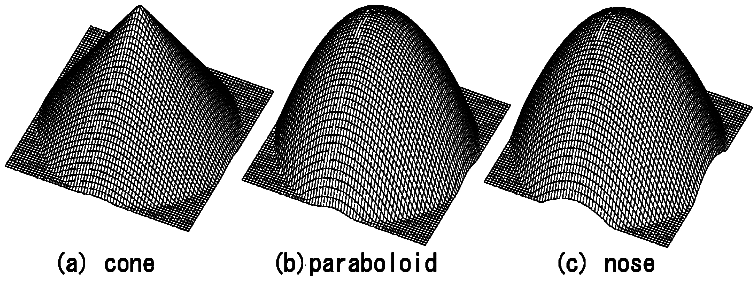


図 4.3: 距離変換に用いる構造要素

ここに I はテーブル変換器に入力される値で，I_V は原画像を縦方向に走査して得られた個々の画素の輝度値，I_H は CTV によってテーブル変換された画像を横方向に走査して得られた個々の画素の輝度値である．また d(p,q) は，構造要素の座標 (0,0) と (p,q) の間の距離であり，たとえば図 4.3 のような形状を持った関数である．このような変換で扱うことが可能な構造要素の形状は，$CTV_q()$ を入力値とする，変換 $CTH_p()$ が一意に定まるという条件のみが必要となる．

このように，2 値画像を原画像とする距離変換を求める場合には，垂直方向と水平方向にそれぞれ別のテーブル変換を組み合わせて用いることにより，Additively separable でない構造要素にも対応可能となる．これにより，ユークリッド距離変換を含め，他のさまざまな距離変換が可能となる．

4.4 2 値画像の距離変換に適した構造要素

以下の説明では，図 4.5(a) に示したような 2 値画像において，白の部分を対象物（on 画素），黒の部分を背景（off 画素）として表現する．2 値画像の距離変換の定義は，着目する off 画素が，もっとも近い on 画素からどれだけ離れているかをすべての off 画素に対して調べる作業である．たとえば図 4.4 のように，On-pixel を距離ゼロとし，そこからユークリッド距離で何画素分離れているかをマッピングする作業であるといえる．

一般に Gray-scale Dilation は，Gray-scale 画像に対して Gray-scale の構造要

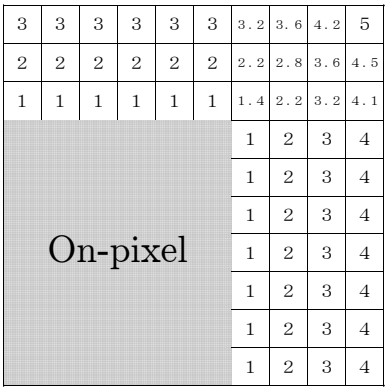


図 4.4: ユークリッド距離変換

素で Dilation 演算を施すものであるが，この論理をそのまま用いて，２値画像に対して何らかの距離を表す構造要素で Dilation を施せば，それによって得られた Dilation 画像の輝度値は距離そのものとなる．例えば図 4.5(a) のような２値の原画像に対し on 画素を Gray-scale の最高輝度，off 画素を Gray-scale の最低輝度に対応させ，図 4.5(b) のような Gray-scale 画像を作り，これに図 4.5(c) のような構造要素で Dilation を施すことにより図 4.5(d) のような Dilation 画像を得る．この Dilation 画像は２値の原画像の on 画素だった部分が最高輝度で，そこから離れるごとに輝度値が低くなって行くような画像であり，この輝度値そのものが最も近い on 画素への距離を直接あらわすものと解釈できる．すなわち，Gray-scale Morphology の Dilation は 距離変換の演算を含んでいるといえる．

このとき，２点間の距離を評価する尺度としては，実に様々な種類の距離が考えられており，それぞれに固有の目的や適用例が紹介されている [29]．画像処理においては，図 4.6 に示したような ユークリッド距離，街区画距離，チェスボード距離などが代表的に用いられるが，ごく一般的な目的にはユークリッド距離やその２乗距離が用いられるようである．なぜならこれらの距離は原点から点対称な距離となっており，これらを用いることで距離変換の回転不変性を保てるからである．ところが，街区画距離やチェスボード距離に比べ計算が簡単ではないことや，計算の論理が Binary の Cytocomputer 型アーキテクチャにのりにくいといった理由から Morphology の分野ではあまり使われて来なかったと言えよう．

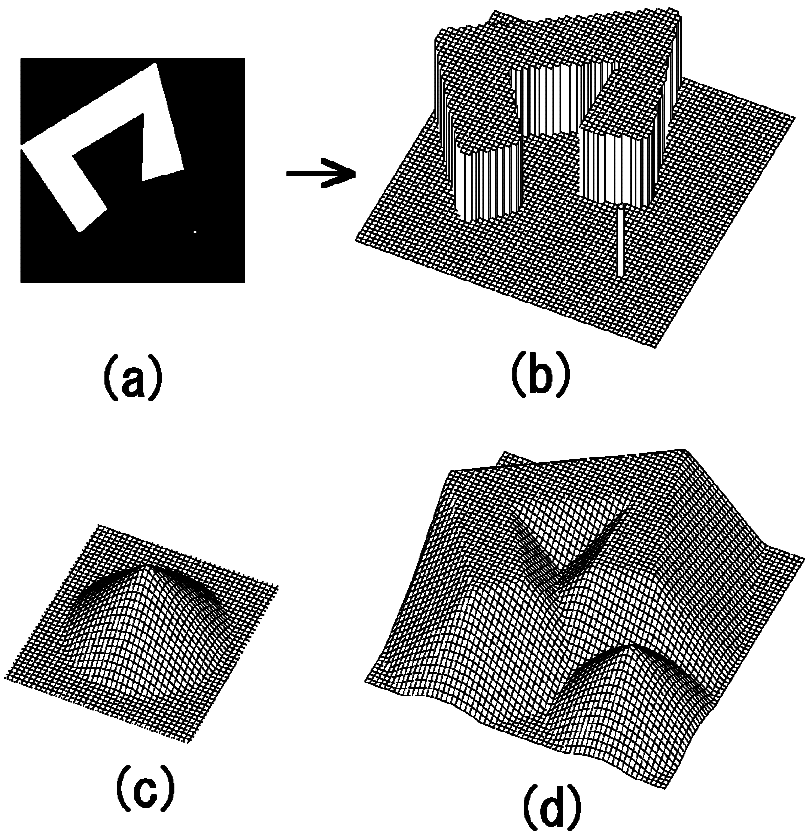


図 4.5: Gray-scale Morphology を用いた 2 値画像の距離変換

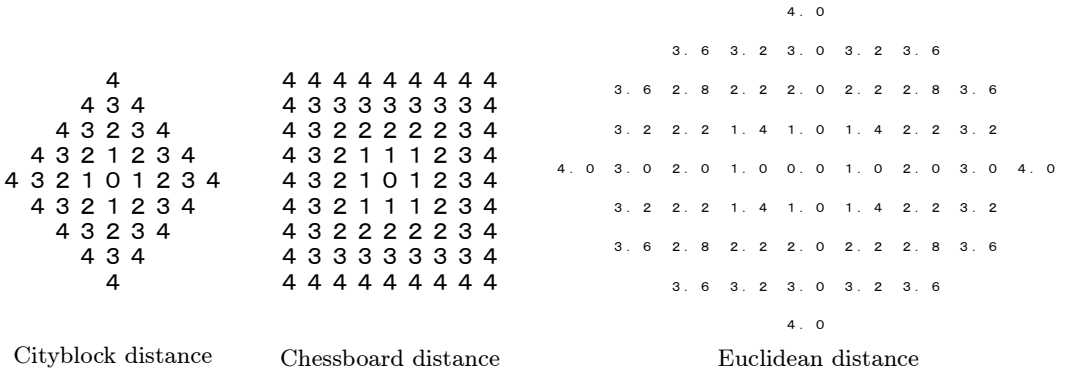


図 4.6: 代表的な距離変換

以下の議論においては，回転対称な距離としてユークリッド距離とその２乗距離を例にとりそれらの性質を述べ，さらに特にデジタル画像で扱うのに便利な新たな距離を提案する．

まずどのような距離が好ましいかについて検討するため，ユークリッド距離を用いる場合の問題点を指摘してみる．Gray-scale Morphology では cone 形の構造要素 (図 4.3(a)) を用いて２値画像に Dilation を施すことにより，直接的にユークリッド距離変換を得ることができるが，この方法の問題点には以下のような点が挙げられる．

単位距離の正方格子上の各点をユークリッド距離でマッピングすると，当然その距離は一般に無理数となるため，その距離を量子化する際に近似値にならざるをえず，小数点以下何桁までを有効とするかで，距離変換の精度が左右される．また，cone 型構造要素の表面の輝度変化を考えると，cone 中心部における隣り合う画素間の輝度変化に比べ，cone 周辺部における座標軸に垂直な方向の輝度変化が非常に小さくなってしまうことがわかる．このため Gray-scale を表す bit 数が固定の場合には，大きな cone 型構造要素の周辺部での輝度の近似が荒くなり bit 落ちが発生してしまう．逆にいえば，周辺部で bit 落ちを生じさせないだけの Gray-scale 方向の bit 数を確保すると中心部での bit の使用効率が悪くなる．

このような原因により，Gray-scale 方向への精度が不足したために起こる計算誤差を含む例を図 4.7 に示す．図 4.7(a) は２つの母点から得られるボロノイ領域

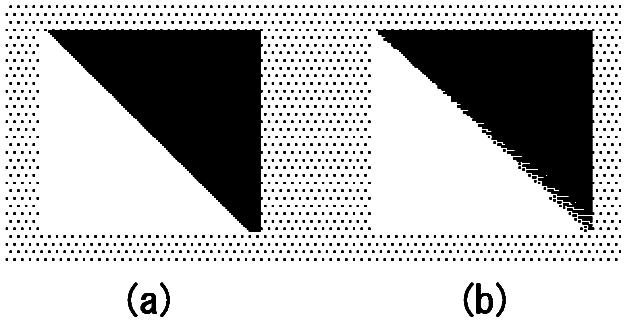


図 4.7: 計算誤差の無い場合 (a) と計算誤差がおきる例 (b)

を示しており，直線的な境界線によって２領域が隔てられている良好な例である．図 4.7(b) は cone 形構造要素の Gray-scale 方向への精度の不足により，正常な領域分割ができなかった例である．

上記の問題に関して，近似を生じない関数としてまず挙げられるのは ユークリッド２乗距離である．この距離は構造要素の形としては図 4.3(b) に表すような paraboloid 形である．またこの距離関数は周辺部における隣り合う画素間の輝度変化がユークリッド距離の場合よりも大きいため比較的大きな構造要素に対しても計算誤差が起きにくいという特徴がある．さらにこの形状そのものが，Additively separable な関数であるという Gray-scale Morphology にとって重要な性質がある [12]．このことからユークリッド距離より，２乗距離の方が距離変換に適するといえるが，さらに計算誤差が無くもっとも大きな距離に対応できる構造要素を実現するための距離について次に提案する．

図 4.8 に示すマッピングは Gray-scale 方向に対して bit の使用効率のもっとも良い輝度変化をもつ構造要素である．このマッピングのことをここではその形状（図 4.3(c)）から仮に nose 形と呼ぶことにする（飛行機の機首の形から）．これは正方格子上の各点において，構造要素の中心をゼロとし，中心からの距離の近い順に整数値を割り当てるというマッピングである．この構造要素は cone 形同様 Additively separable な関数ではないため，その使用に当たっては，構造要素を２次元情報としてもっている Morphological アルゴリズムかもしくは Chapter 4 で述べるハードウェアが必要となる．

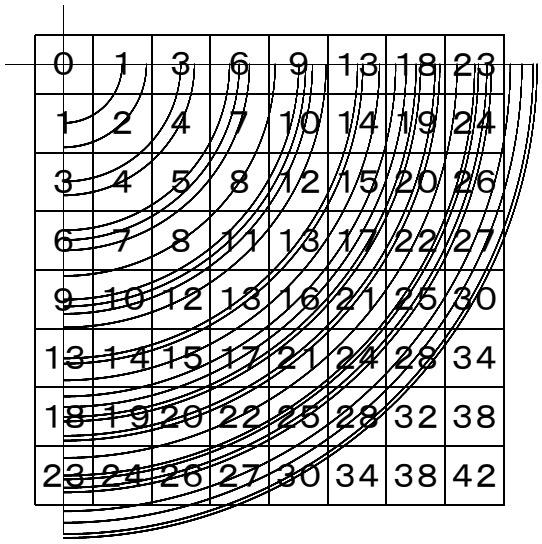


図 4.8: nose 形構造要素のマッピング

表 4.1: 誤差がない最大の構造要素の半径画素数

構造要素の種類	4bit	8bit	16bit	n bit
cone	2	11	181	$[\sqrt{2^{n-1}}]$
paraboloid	4	16	256	$[\sqrt{2^n}]$
nose	5	27	539	

以上に述べた 3 つの構造要素に関して，計算に用いることのできる Gray-scale の bit 数に応じて，計算誤差の起きない構造要素の半径がどのようにきまるかを表 4.1 に示す．表 4.1 で示した半径とは，構造要素を量子化する際，異なっているべき距離が同一の値に量子化されることのないよう構造要素を輝度方向にスケーリングしたときの，その bit 数で表現可能な最大の距離（画素数）を表している．

4.5 テーブル変換器を用いたアーキテクチャ

Chapter 3 に述べたように Gray-scale Morphology の演算を 1 次元処理することの最大の利点は，その計算量を大幅に減らせることであるが，このことは，こ

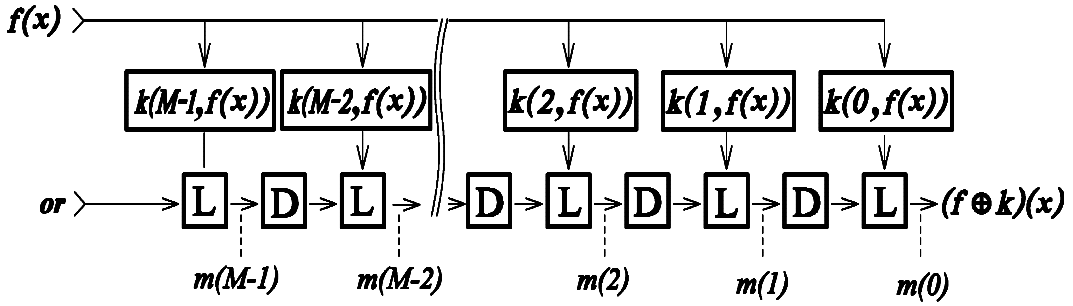


図 4.9: テーブル変換器を組み込んだ信号フロー

れをハードウェア化した場合，回路規模を大幅に削減することができることにもつながり，とくに大きな構造要素による処理を実現する際にこの手法が有効である．図 4.9 はテーブル変換器を用いた 1 次元 Dilation の信号フローを表している．この中で $k(*,*)$ は，入力信号 $f(x)$ とその座標における構造要素の値との演算をテーブル化したものである．すなわち，図 3.12 に示した固定構造要素の 1 次元 Dilation の信号フローの構造要素レジスタの部分にテーブル変換器を組み込んだものがこのアーキテクチャの基本となっている．図 4.10 は 1 次元信号の Dilation をテーブル変換器を用いて実行するためのプロセッサの概略図である．大まかな構成としては，ホストとなる計算機（パソコン）と Dilation プロセッサ，indexing プロセッサと，それらを結ぶ I/O からなる．

この試作機の構成では，画像データはホストとなるパソコンのメインメモリ上に持っており，これをソフト的に縦，横に走査して，1 画素ずつパソコンのバスに取り付けられた I/O を介して，Dilation 計算機に送る．Dilation 計算機は，この I/O に同期して動作し，得られた計算結果は 1 画素ずつホストに取り込まれる．

ここで式（4.2）に示した Dilation 演算を並列に実行するためのアルゴリズムに書き換えると，図 4.11 のようになる（ここでは x, z は 1 次元）．図 4.11 中において $m(0)$〜$m(M)$ は計算のために用いる中間変数，$larger(A, B)$ は入力された A, B のうちのどちらか大きな値を返す関数である．$CT_1()$〜$CT_M()$ は，入力されたある一つの値に対して，あらかじめ設定されたひとつの値を返すテーブル変換器であり，入力と出力の精度（bit 数）が等しく，その関数関係は Dilation

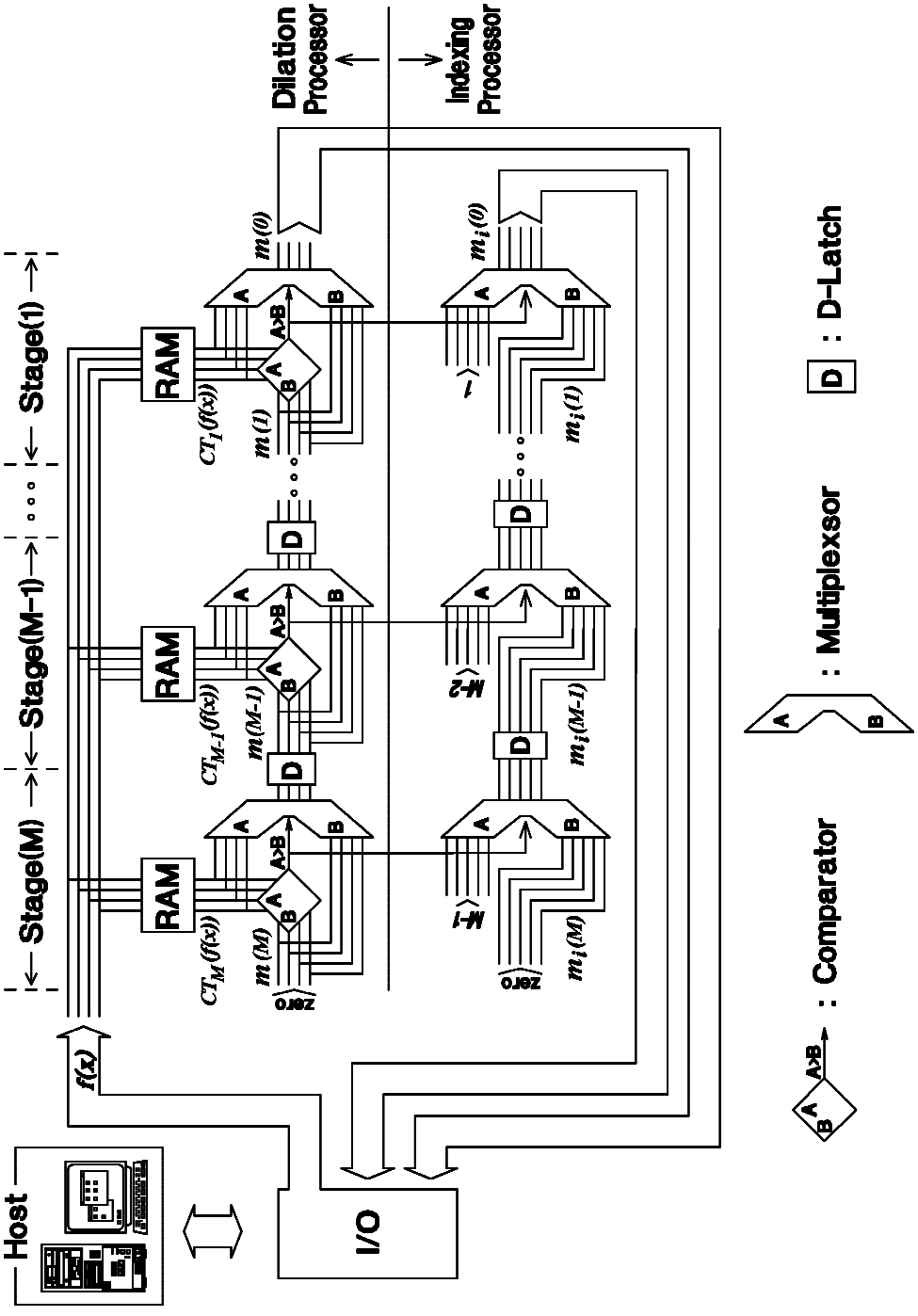


図 4.10: 試作機のアーキテクチャ

for($x := 0$ to x_{max}) — loop 1
{ for($z := 0$ to $M-1$) — loop 2
{ $m(z) := larger(CT_{z+1}(f(x)), m(z+1))$
}
$(f \oplus k)(x) := m(0)$
}

図 4.11: テーブル変換器を用いた一次元 Dilation のアルゴリズム

処理前に設定し，1 画面の走査が終了するまでは変化しないものとする．

ここで図 3.11 のアルゴリズムから図 3.12 の信号フローが求められた時と同様，図 4.11 中の loop2 が並列化可能であることに着目すると，これをもとにしたパイプライン型のプロセッサ（図 4.10 の Dilation processor 部分）が設計できる．このプロセッサは，メモリ，マグニチュードコンパレータ，マルチプレクサ，D ラッチから構成することができる．この構成のプロセッサを用いる場合，一回の画像の走査で処理が可能な構造要素の大きさはプロセッサのパイプラインの段数に押さえられるが，これより大きな 1 次元構造要素による処理を実現するには，式（4.5）で示したように，もとの構造要素 k を パイプライン段数に収まるように区切って，それぞれについて Dilation を実行することで可能となる．Chapter 3 の式 3.12 と式（4.5）は本質的には同じ物であるが，Chapter 4 に示したハードウェアの実情（OR 入力が存在しないこと，1 次元構造要素の分割数 (n) が無制限であること）を反映してあらたに書き直すことにした．

$$f \oplus k = (f \oplus k_1) \cup (f \oplus k_2) \cdots (f \oplus k_n) \tag{4.5}$$

4.6 テーブル変換に必要なメモリ量

図 4.2 に示した変換テーブルでは，$I \times Q \times P$ 個の要素を持っているが，このテーブルの要素のうち，独立した値として実際に保持すべき要素の数はこれより少なくて済む．実際にどれだけのメモリ量が必要になるかはアプリケーションによってきまるが，このことを距離変換に用いる場合と，Additively separable 構

造要素による Gray-scale Dilation の場合に分けて説明する．

まず，距離変換にテーブル変換器を用いる場合，一個のテーブル変換器に必要なテーブルの要素は，式 (4.4) から，たかだか $1 + \times Q \times P$ ワードである．ここで 1 ワードは 2^n bit であり，n は Gray-scale の bit 数，すなわち距離を表す数値の最大値に相当する．また 1 次元化アーキテクチャにおいては，p,q の各 1 次元座標にテーブル変換器が置かれる構成となるため，一個のテーブル変換器が持つべき要素は CTV_q に 2 ワード，CTH_p に q ワードのみである．

次に，Chapter 3 に述べた Additively separable 構造要素による Gray-scale Dilation の場合には，構造要素は固定でよいため，必要な変換テーブルの要素は $2 \times I$ ワードである．また，図 3.12 からわかるようにそれぞれのテーブル変換器が持つべき要素は CTV_q，CTH_p ともに I ワードである．

上記の 2 つのアプリケーションにこのアーキテクチャを適用する場合には，実際のハードウェアとして一個のテーブル変換器に用意すべきメモリ量はたかだか Q ワードにすぎない．すなわち個々のテーブル変換器に用いるメモリに必要なアドレスバスの本数は $log_2 Q$ 本であり，データバスの本数は必要とする Gray-scale の bit 数だけ持たせればよいことになる．ただし，式 (4.5) に示した構造要素分割を行う場合には，分割数を乗じただけのアドレス数が必要となることに注意が必要である．この場合，アドレスバスの本数が，具体的にどれだけ必要になるかは Section 4.7 に述べる．

4.7 テーブル変換器アーキテクチャの試作機

図 4.12 は本研究において試作した Dilation 計算機で，パイプライン段数は 32 段，Dilation 計算機部分のマシンサイクルはホスト計算機とのやり取り（I/O 命令）を含めて 1 画素当たり約 900ns 程度である（これは I/O のバス速度のネックを解消することによりさらに 3 倍程度の高速化が見込まれる）．またホスト計算機側では，主記憶から原画像を読み込む時間と，Dilation 計算機から取り込んだ画素の輝度情報を処理して主記憶に格納する時間として合計 200ns ほどの時間が 1 画素当たりに必要となるため，試作機のマシンサイクルは実行性能としては

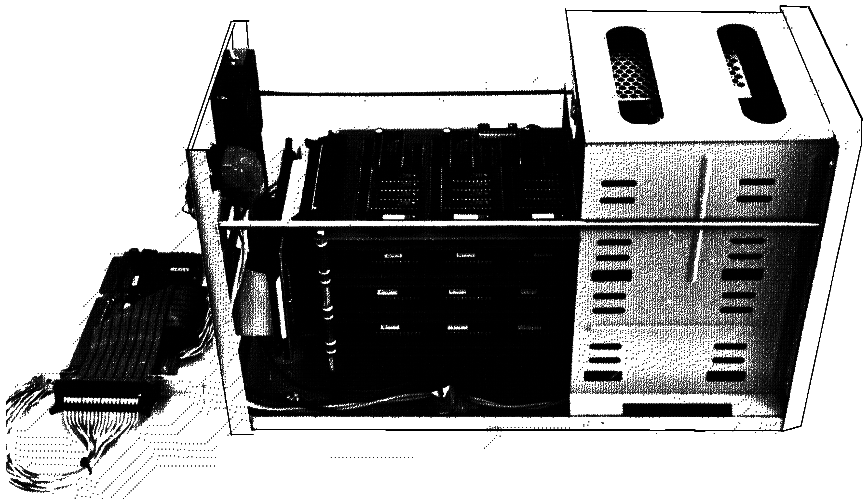


図 4.12: 試作機（Dilation プロセッサと I/O 基板）

900ns に 200ns を加えて 1 画素当たり 1.1 μ s 程度となる．

実際の画像処理においては，縦・横両方に画像を走査してプロセッサに送り込むため，1 画面の処理には原画像の画素数の 2 倍のマシンサイクルが必要となる．またパイプライン段数より大きな構造要素を使う場合には，分割した構造要素の個数倍のマシンサイクルがさらに必要となる．このため計算時間は Chapter 3 Section 3.7 で式（3.13）に示したのと同様になるがマシンサイクルを 1.1 μ s としたときの計算値 T_pが実行性能そのものとなる点が Chapter 3 の場合と異なる．

この試作機における具体的な計算時間の例をあげると，原画像 512×512 画素，16bitGray-scale の画像に対する 128× 128 画素の構造要素における Dilation 演算が約 2.5 秒である．この計算時間は，Pentium 90MHz の計算機でソフト的に同じアルゴリズムによる計算を実行した場合に比べ約 20 倍程度高速である．

この試作機ではテーブル変換器として，1M Bit SRAM 2 個による 17bit アドレスバス, 16bit データバスのメモリ構成を用いている．メモリの持つアドレスバスの本数としては，これが必要かつ充分な本数といえる．この設計指針は，実際のアプリケーションとして最もメモリをたくさん要求する場合を想定した，使用段階における要求から来ている．

たとえば nose 型構造要素を使って Section 4.8.3 に示す Farthest ボロノイ図を求める場合について考えて見る．まず，構造要素の半径を表す bit 数，すなわち log_2Q は，表 4.1 の 16bit の場合から log_2539 を切り上げた 10 bit が必要である．そして，パイプラインの段数を 32 段としたとき，その分割数は $2^{10}/32 = 2^5$ となるためさらに 5bit が必要となり，合計で 15bit が必要とするアドレスバスの本数となる．ただしアドレスバスに 15bit しか用意しない場合には，実際に使用する段階において画像処理を行うたびに CTV の内容と CTH の内容を入れ換える必要が生ずる．このためアドレスバスの bit 数に少しだけ余裕をもたせることによって，テーブル変換器のメモリ内容を入れ替えるための作業を省くことができるように配慮した．このため 実際に入手できるデバイスとして，17bit アドレスバスを採用することに決定した．また 17 本のアドレスバスのうちの 1 本は，バスの構成上どうしても扱いにくい半端なアドレス線となるため，これはソフト側からのコマンドにより，トグル動作をするように設計した．この特殊なアドレス線は例えば，CTV と CTH とでテーブルの内容を入れ替えることや，２種類のまったく違った構造要素を取り扱うといったことに利用できる．

また，Gray-scale の深さに 16bit を要さない処理のために，データの上位 bit を固定するためのマスク機能を I/O 部に持たせている．これにより，一回のメモリロードで，非常にたくさんの種類の構造要素を切り替えて処理を行うという機能も実現している．この機能により，例えば Section 3.6 に示したような構造要素の分割を行った際に，オーバ・ヘッド無しにテーブル変換器の内容を切り替えることができるといった利点が生ずる．

4.8 ボロノイ図のためのインデクス機能

4.8.1 距離変換を用いたボロノイ図生成法

計算幾何学の分野における重要な道具立ての一つであるボロノイ図を，安定かつ高速に求めるための研究がされているが [30]，浮動小数点演算をベースとする幾何的アルゴリズムを用いたボロノイ図構成法が考えられている一方で，画像情報のような画素単位の精度のボロノイ図を求める場合には，Chapter 4 に述べた

計算機を有効に活用することができる．画像処理におけるボロノイ図とは，着目する２値画像に含まれる on 画素の勢力範囲図と表現することができる．例えば図 4.13 (a) を母点とするボロノイ図は図 4.13 (e) のようになるが，このように着目する off 画素に対してそれに最も近い on 画素の属性を割り付ける作業を，すべての off 画素に対して実行することによりボロノイ図を求めることができる．

Dilation を用いてボロノイ図を求めるにはまず，図 4.13(a) のような２値の原画像を図 4.13(b) のような，on 画素が最高輝度，off 画素が最低輝度となるような Gray-scale 画像に変換し，これに図 4.13(c) のような凸型の構造要素を用いて Dilation を施し，その結果 (図 4.13(d)) を塗り分けることにより最終的なボロノイ図を得る (図 4.13(e))．

Dilation を用いるボロノイ図作成方法の利点としてまず，浮動小数点演算による誤差の問題が無いため，解が常に安定であることが挙げられる．もちろん section4.4 で述べた「計算誤差が起きない場合」が前提であるため無制限に大きな画像は取り扱えないという制約は存在するが，ターゲットとするアプリケーションが，遠く離れた母点間のボロノイ辺を求める必要が無いものであればこの制約条件は問題にならないであろう．さらに大きな特徴として，計算幾何学的手法では母点の個数を n 個としたとき，その計算量のオーダが $O(n^2)$ や $O(n\log(n))$ となるのに対し Chapter 4 に述べたハードウェアを用いた場合には，計算時間がボロノイ母点の個数に依存しないことがあり，その計算時間は原画像の面積のみに比例することが言える．また幾何学的アルゴリズムではボロノイ母点が疎に撒かれた点群であることが前提となっているが，距離変換を用いた方法では，塗り潰された領域のようにボロノイ母点が局在化していてもまったくさしつかえない．例えば図 4.14 は，ボロノイ図を利用してプリント配線の異常接近箇所を調べる応用例であるが，図 4.14(a) のような，点群ではなく塗り潰された領域を母点群とするボロノイ領域を求めるのにこの計算方法が適しているといえる．ここで図 4.14(b) は距離画像であり，配線同士の接近箇所は浅い溝状の部分を探す事で図 4.14(d) のように求められる．また図 4.14(c) がボロノイ領域を表しており，接近箇所がどの配線とどの配線の間に存在するかを調べるためのネットリストにそのラベリング情報を反映させることができる．

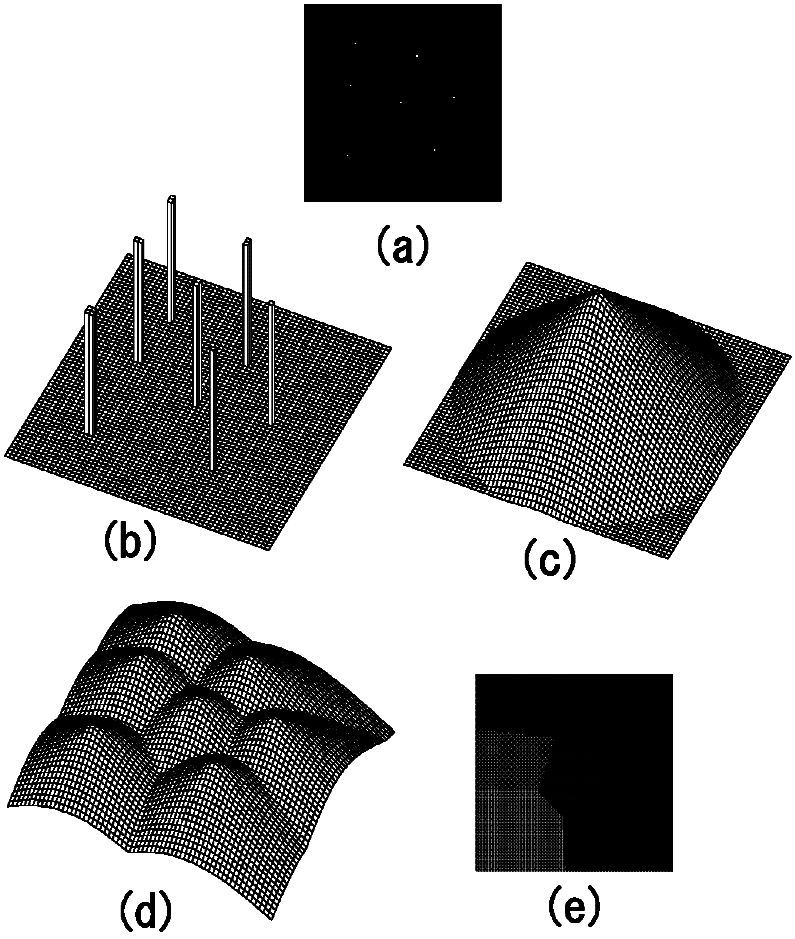


図 4.13: Dilation を用いたボロノイ図の作成方法

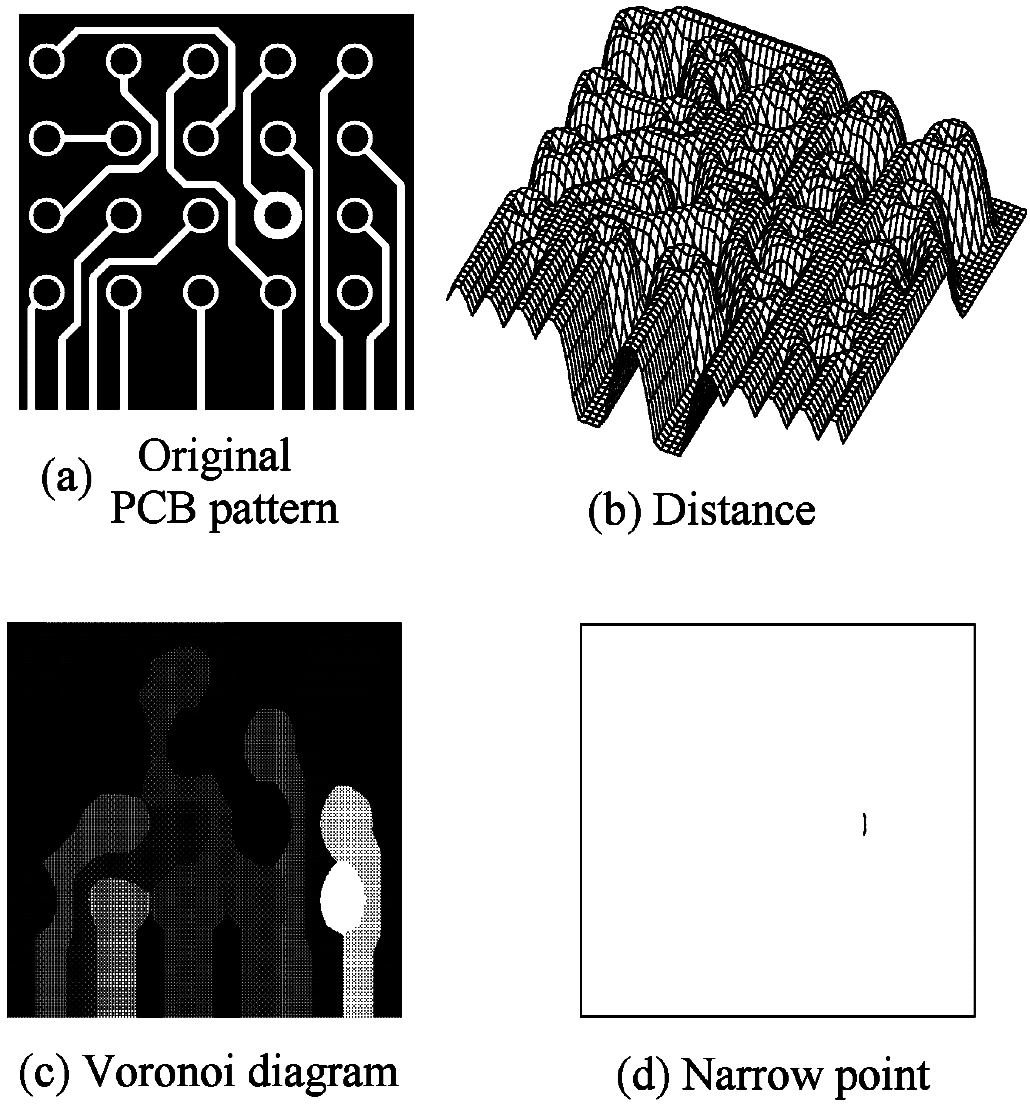


図 4.14: PCB の Design rule check への応用

4.8.2 ボロノイ図のためのハードウェア

cone 形や nose 形の構造要素で Dilation を施すことにより，ボロノイ図を求めるための前段階となる距離画像を求めることが可能であるが [6] ，その際 Dilation によって産み出されたある画素が，原画像のどの座標の画素がもとになって作り出されたかを求めることが同時にできれば，ボロノイ領域のラベリングにその情報を直接用いることができるようになり都合が良い．

このため本研究における試作機では，図 4.10 の indexing processor 部分に示すように２入力マルチプレクサからなるロジックを付加している．このロジックから得られる数列を検索することにより，距離変換に用いる構造要素の頂点の座標を得ることができ，距離変換と同時にボロノイ領域のラベリングが可能となる．このように Dilation プロセッサで母点の座標を求める具体的な手順は，図 4.15 に示すアルゴリズムによって示される．図 4.15 のなかで $m_i(z)$ は，$m(z)$ が $CT_{z+1}(f(x))$ によって上書きされたとき，パイプライン上のどの stage においてその上書きが起ったかを記録するための配列である．そして loop 1 終了後ホスト計算機上で loop 3 を実行することで，配列 Index() にボロノイ母点の１次元座標を得る．実際にはこのアルゴリズムは原画像に対し縦方向と横方向に２回施されるため，それぞれの過程において得られた配列からボロノイ母点となる２次元上の座標を得ることになる．

4.8.3 凹形構造要素の使用

Cytocomputer 型の Gray-scale Morphology 計算機と比較した場合，本研究によるアーキテクチャの特長として，構造要素の中心画素の輝度が低く，周辺の画素の輝度値の方が高いような凹形構造要素が使用可能であるということが言える．これは式 (4.5) に示したように，分割した構造要素が，構造要素どうしの入れ子の Dilation でなく，それぞれ独立した Dilation 結果の論理和の形のまま計算することが可能だからである．この性質を利用することにより，Farthest ボロノイ図（ある背景画素に対し最も遠い母点の属性を割り当てるマッピングで，図形の外接円を得る際などに用いられる）を求めるためにこの計算機が有効に利用でき

```
for( x := 0  to  x_max )                                  — loop 1
{ for( z := 0  to  M − 1 )                                — loop 2
  { if ( CT_{z+1}(f(x)) > m(z + 1) )
    { m_i(z) := z
      m(z) := CT_{z+1}(f(x))
    }
    else
    { m_i(z) := m_i(z + 1)
      m(z) := m(z + 1)
    }
  }
  (f ⊕ k)(x) := m(0)
  Index(x) := m_i(0)
}

for( x := 0  to  x_max )                                  — loop 3
{
  Index(x) := Index(x) + x
}
```

図 4.15: インデクス機能を含むアルゴリズム

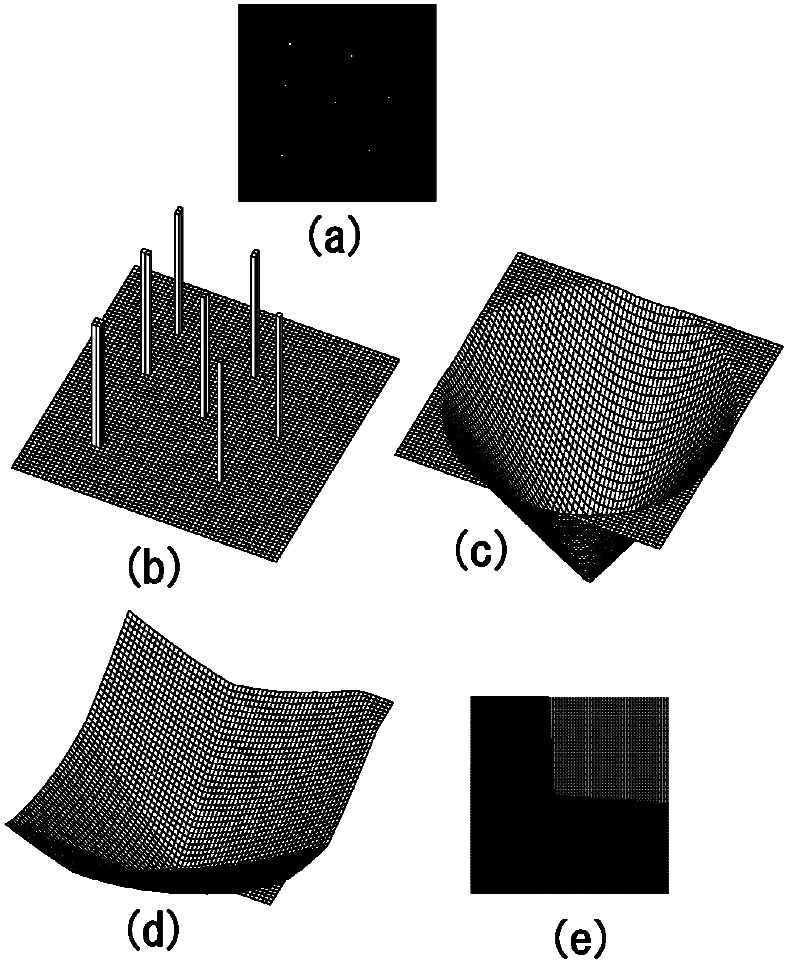


図 4.16: Farthest ボロノイ図の作成法

るといえる．Farthest ボロノイ図を距離変換を使って求めるには原画像の一辺の長さの$\sqrt{2}$倍の半径を持つ「すり鉢型」構造要素を使っての Dilation が必要となるが，そのような大きな凹形構造要素を実現するために，Chapter 4 に述べたアーキテクチャと，nose 型構造要素が有効に機能すると言える．図 4.16 に Farthest ボロノイ図を作る過程を示したが，これは図 4.13 に対比して構造要素が替っているだけであり手順そのものは同じである．

4.9　Chapter4 のむすび

Chapter 4 ではテーブル変換器を含むロジック回路をパイプラインに構成することにより Gray-scale Dilation の計算を高速に実行するプロセッサを示した．またこのプロセッサでは，Gray-scale 画像の Additively separable な構造要素による Dilation に加え，非 Additively separable な構造要素による 2 値画像の距離変換を効率的に実行できることを示した．また誤差の無い距離変換を，より少ない Gray-scale の bit 数で計算するのに適する nose 型構造要素を提案した．そしてボロノイ図を求める際に必要となる母点の座標を保持するためのレジスタを Dilation プロセッサに付加する方法を示した．

Chapter 4 で述べたアーキテクチャの試作機は Morphology プロセッサの部分に大きな比重を置き，画像を貯えておくメモリはホストとなるパソコンの主記憶に置いている．これはパソコンのバス性能の向上に伴い，外部メモリに画像をすべて転送してから処理をするメリットがさほどなくなって来たことが主な理由であり，かつハードウェア構成をシンプルにしたり，使用段階での機能の柔軟性を向上することにも寄与している．すなわち Chapter 2，Chapter 3 に紹介したアーキテクチャに共通していた，ハードウェア的に画像の座標を入れ換える回路はここでは用いていない．このように見ると，Chapter 2 に述べた試作機はスタンドアロン動作に近いもの，Chapter 3 に述べた試作機は，ホストに取り付けられた外部画像処理装置的なもの，そして Chapter 4 の後半で述べる試作機はホストのバスに組み込まれた画像処理用アクセラレータ的なものと位置付けることができよう．

Chapter 5　総括

5.1　本研究の成果

本研究ではMorphology 画像処理を高速化する手法を，計算のアルゴリズムとそれを実行するためのハードウエアの両面から検討し，望ましい結果を得た．提案した手法に関する重要な事項は以下のようにまとめることができる．

１．Morphology 処理を１次元化する手法を採用することにより原理的に計算量のオーダが削減されるが，１次元化のメリットはそれだけにとどまらず，ハードウェアを簡単化することにも役立つことを示した．

２．試作機によるアーキテクチャの検証を行った．

３．大きな構造要素による Morphology 処理が使えることによるメリットを例示した．

これらは各 Chapter に共通した結果であるといえるが，本論文に示した３つのアーキテクチャには得意な点やそうでない点がそれぞれに存在するといえる．どのような目的に Morphology を用いるかで，Binary か Gray-scale のどちらかの Morphology 処理を選択することになるが，Binary Morphology を行うために Gray-scale Morphology を使って実行する方が良い場合も考えられ，一概にその役割を最初から分けてしまうことはできない．このため以下の Section では論文全体を通しての各方式の位置付けやその役割について述べる．

5.2 Binary Morphology (Chapter2)

本論文の Chapter 2 と，Chapter 3 に示したアーキテクチャを比較すればわかるとおり，Binary Morphology のハードウェアは Gray-scale のそれに比べて小規模かつシンプルである．これは画像の取り込みから出力まですべて 2 値の論理で計算を実行することができるからである．また 1 次元 Morphology の論理は，シフトとカウントという 2 つの単純な論理的動作に帰着することができるため，高速な画像処理が実現可能である．このため特にビデオレートでの実時間処理が要求されるようなアプリケーションに向くと言える．また構造要素の大きさが自由に変えられるという点もこのアーキテクチャの利点である．構造要素の拡大・縮小は，カウンタおよびシフタに設定するパラメタの変更のみで可能である．

構造要素の形状について言えば，Chapter 2 に示したアーキテクチャの場合，その形状の自由度は使用可能な長方形構造要素の個数によって決まるとしてよい．つまり，必要な長方形の個数に見合うハードウェアをあらかじめたくさん作り込んでおけば，構造要素の形状の自由度は確保できるが，反面ハードウェアの規模が大きくなってしまう．このためどのようなアプリケーションに適用するのかをあらかじめ決めた上でハードウェア構成を決める必要があるため，ハードウェア構成を柔軟に変えられるような構造としておくことが汎用性を持たせるための用件となる．

5.3 Gray-scale Morphology (Chapter3)

Threshold decomposition 法を使えば，Binary Morphology のハードウェアで Gray-scale Morphology の計算を実行することはできるが，Gray-scale の画像処理は Gray-scale を指向したハードウェアで実行するほうが無理がないということはこの場合確かなことであり，Section 3.8 でも述べたとおり，数値どおしの加算や比較を Gray-scale の論理のまま計算する方が，ハードウェアの規模の削減において有利であると言える．また使用する構造要素が Additively separable であれば，1 次元化法による大幅な計算量の削減が可能となる．また本来 Dilation 演算は，それを実行する計算の中に多入力の最大値演算を含んでいるが，1 次元の時

系列信号の Dilation 演算においてはこの「多入力 MAX 演算」を複数の「比較代入演算」に置き換えることが可能であるため，ここでも大幅な回路規模の削減と，ハードウェア化した場合の動作の高速化が実現でき，２重の意味で１次元化は重要である．

ところで Gray-scale Morphology は Binary Morphology の論理を含んでいるため，汎用の画像処理装置を構築するのが目的であれば Gray-scale を選択するべきである．さらに Binary Morphology において画像処理の回転不変性を保証したい場合には，２値画像をいったん Gray-scale 画像に変換しておき，これに対し Paraboloid 構造要素による Morphology 演算を実行した後，再び２値画像に戻す，という方法をとれば良い．このように Binary Morphology における円形構造要素が必要な場合，構造要素が小さいもの（例えば直径が十数画素以内）であれば Binary Morphology のアーキテクチャが適すると言えるが，数十画素大の構造要素を実現するためには Gray-scale Morphology のハードウェアを使う方が小さい回路で実現可能である．

5.4 テーブル変換器アーキテクチャ (Chapter4)

このアーキテクチャでは Gray-scale Morphology の基本式に含まれる，構造要素と原画像との和の部分に替えて，ある入力に対しある出力パターンを出すテーブル変換器を用いるという試みを行った．このため Chapter 3 に述べた Gray-scale Morphology の１次元処理の論理に対しさらに柔軟性を持たせることができ，適用可能なアプリケーションの範囲を広げることができた．

また Morphology の論理に付加する形の「indexing 機能」を新たに加えることにより，Dilation の際，出力画像が入力画像のどの部分がもとになって作られたかを，出力をもとに検索できるようになったため，ボロノイ図を直接的に求めることが可能となっている．この indexing 機能はボロノイ図を求めるために付加したものであるが，これ以外になにか応用可能な画像処理の分野があるかもしれない．

また Chapter 4 に示したアーキテクチャで実行するのに適した形状の構造要素

として nose 型構造要素を提案した．この構造要素を用いることにより誤差の無い距離変換を広い範囲で行うことが可能となった．

このアーキテクチャの試作機では，画像処理を 16bit の Gray-scale の深さで扱っている．通常の画像処理では，Gray-scale の深さが 8bit あれば白黒写真に近いクオリティーが得られるため，画像処理の多 bit 化はあまり行われなかったと思われるが，これを 16 bit とすることにより，２値画像の距離変換が高い精度で実行できるようになった．またこのような多 bit 化が可能であることは１次元アーキテクチャに得意な点であると言える．

5.5 適用可能な Gray-scale 構造要素に関する各方式の比較

以下では４種類の Gray-scale Morphology のアーキテクチャに関して，構造要素の自由度という側面において比較を行う．比較の対象とするアーキテクチャは以下の４種の通りである．

１． Chapter 3 に述べた１次元 Gray-scale Morphology アーキテクチャ

２． Chapter 4 に述べたテーブル変換器を用いた Gray-scale Morphology アーキテクチャ

３． Gray-scale の Cytocomputer 型アーキテクチャ（3×3 構造要素の縦列処理）

４． 直列計算機や MPP を用いて定義式の通りに実行する方法（直接法 = Brute force method）

そしてもう一方の比較の対象である構造要素は，図 5.1 に示すように，クラス１からクラス６までの６種類に分類して考える．分類の基準は以下の４通りであるが，分類に包含関係があるのでそのバリエーションは６種類となる．

任意 すべての構造要素が属するクラス．極小点をもっていたり，中央が窪んだ形の構造要素も含んでいる．

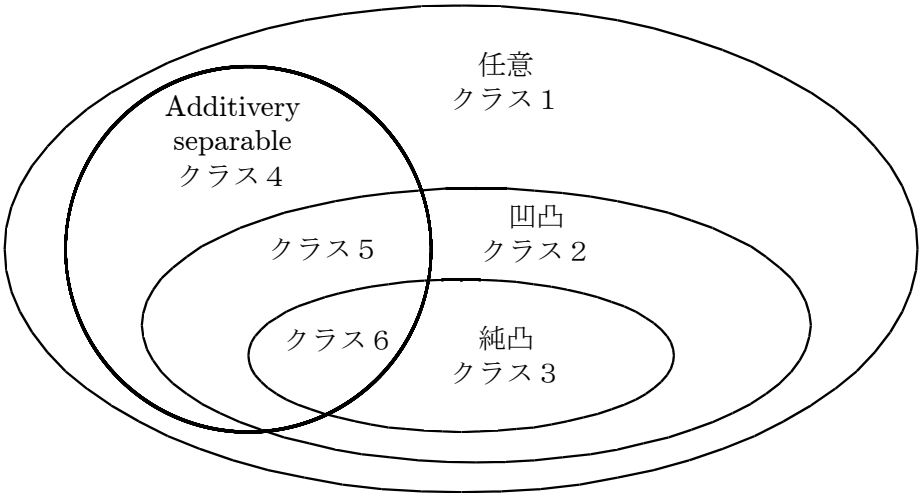


図 5.1: 構造要素のクラス

Additively separable Section 3.3.1 に示したような構造要素に属するクラス.

凹凸 このクラスは図 5.2(a)(b)(c) に示したように中心の高い山の形をしている. 山の斜面に浅いへこみが存在することは許すが「水が溜まるようなくぼみとなる」極小点は持たない. またその斜面は図 5.2(b) の”Mixed type”のように変極点を有する場合がある. ここではクラス凹凸は F.Y.Shih[9] の分類による ”Mixed type” および ”Concave type” をあわせた分類として扱っている. また,「クラス凹凸」は以下の「クラス純凸」を含んでいる.

純凸 このクラスは図 5.2(c) に示したように中心が高い山でその斜面は, 常に「上に凸」となる極率を有する, いわゆる ”Convex Class” である.

以上のような分類において, 原画像が Gray-scale であるような一般の Gray-scale Morphology の場合に, それぞれのアーキテクチャが得意とする構造要素の形状を書き出したのが表 5.1 である. 表 5.1 における「◎○△×」はそれぞれのアーキテクチャにおいて, 以下のような評価基準で場合分けしたものである.

◎ そのアーキテクチャにおける理論的な最小の計算量で実行可能な場合.

図 5.2: 構造要素の凹凸の分類

○ そのアーキテクチャの範疇であるが，多数の構造要素による縦列処理が必要となる場合で「◎」より計算量が多くなる場合..

△ 多数の構造要素を Morphology の分配則（式（3.5））によって組み合わせる必要がある場合．しばしば技巧的かつ多数の組み合わせが必要となる．

× 実現が不可能な場合．もしくは実現可能であったとしても，計算量のオーダが直接法の場合と変らない場合．

表 5.1 のような分類において 1 次元化法（1D および table）と Cytocomputer 型を比較すると，一長一短という見方もできるが，Additively separable なクラスの構造要素においては，くぼみを有する構造要素に対しても有効であり，この点が 1 次元化法の得意な点と言える．また Additively separable な構造要素の場合には，計算量がその形状に影響されないということも言える．逆に非 Additively separable な構造要素の場合には 1 次元化法は適用しにくく，テーブル変換器アーキテクチャもここでは有効ではない．なぜなら，輝度値が任意の原画像に対し Dilation を実行した場合の結果は，複数の構造要素の相貫図形と見なすことができるが，図 5.3 に示したような場合において，1 次元アーキテクチャでは谷間の部分（ハッチ部）を正しく計算できないことによる．これは原画像に含まれる複数の「種画素」が垂直（水平）の位置にあった場合，先に 1 次元処理する方向が垂直（水平）であるなら，相貫図形の接合面の部分に「食い込み」が発生することによる．

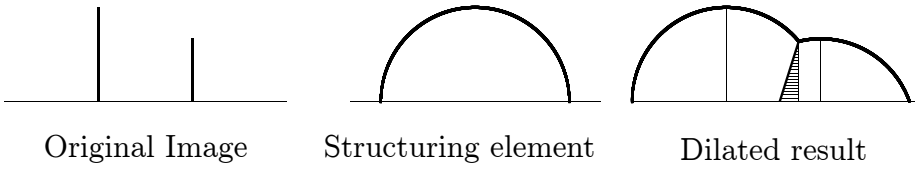


図 5.3: Gray-scale 原画像と非 Additively separable 構造要素との Dilation

表 5.1: 各アーキテクチャの得意とする構造要素

原画像が Gray-scale の場合

構造要素の種類	1D	table	Cyto	Brute
クラス 1	×	×	×	◎
クラス 2	△	△	△	◎
クラス 3	△	△	○	◎
クラス 4	◎	◎	×	◎
クラス 5	◎	◎	△	◎
クラス 6	◎	◎	◎	◎

1D : １次元アーキテクチャ
table : テーブル変換器アーキテクチャ
Cyto : Cytocomputer
Brute : 直接法

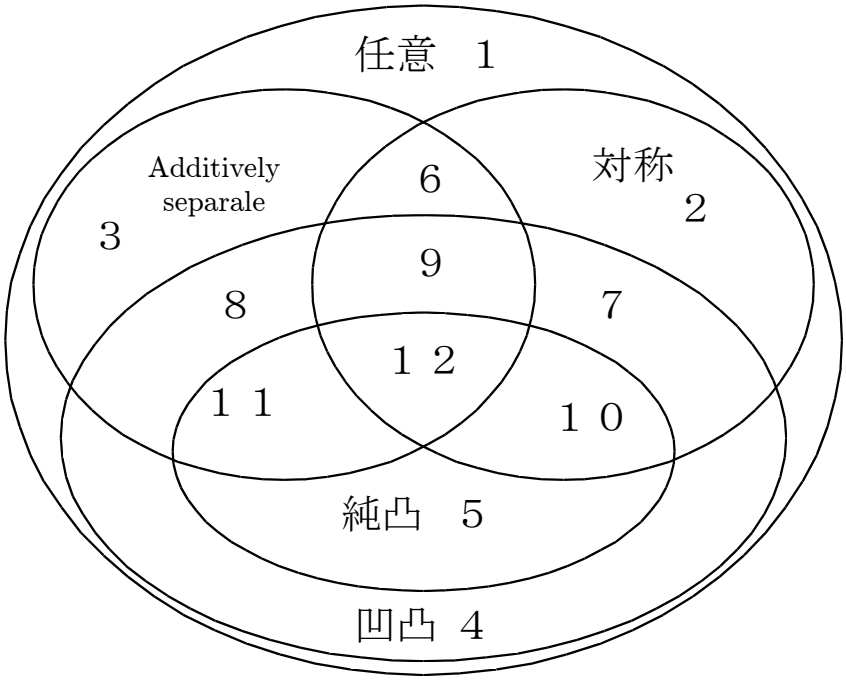


図 5.4: 対称，非対称の分類を含む構造要素のクラス

このような理由で，原画像が Gray-scale の場合においては，２つの１次元化法（1D と table）の間に差はなく，構造要素が Additively separable であるかどうかのみが可否を決定する用件となる．

一方，図形の距離変換画像を求める場合には原画像を Binary と限定することができ，このようなアプリケーションに対しては,「1D」と「table」に大きな差が出る．以下ではあらたな構造要素の分類として,「対称であるかどうか」という用件を導入する.「対称」とは，１次元分解された構造要素のうち先に処理される方向に対して対称な形状を持っているかどうかということであり，対称であるなら，図 5.3 のハッチ部分のような計算不可能領域は生じない．また回転対称な構造要素はすべてここで言う「対称な」構造要素である．この「対称」か「非対称」かという新たな用件を加えた構造要素の分類を図 5.4 に示す.

表 5.2 は，原画像が Binary の場合，どのアーキテクチャがどのクラスの構造要素に対して有効かを書き出したものである．ここからわかるとおり，テーブル

表 5.2: 各アーキテクチャの得意とする構造要素

原画像が Binary の場合

構造要素の種類	1D	table	Cyto	Brute
クラス１	×	×	×	◎
クラス２	×	◎	×	◎
クラス３	◎	◎	×	◎
クラス４	×	×	△	◎
クラス５	×	×	○	◎
クラス６	◎	◎	×	◎
クラス７	△	◎	△	◎
クラス８	◎	◎	△	◎
クラス９	◎	◎	△	◎
クラス１０	△	◎	○	◎
クラス１１	◎	◎	◎	◎
クラス１２	◎	◎	◎	◎

変換器アーキテクチャが有効に機能する構造要素の用件は，対称であることと，Additively separable であるという２つの条件のうちどちらか一方が成り立っていれば良いため，原画像が Binary の場合には適用範囲が格段に広くなることになる．しかし「対称」でも Additively separable でもない，クラス４，５ に関しては，２つの１次元化法は適用できず，このクラスでは Cytocomputer が有利である．

5.6 より良いアーキテクチャのために

次の Morphological アーキテクチャはこういう物にすればより良いものができるであろう，という展望を述べることで，本論文の締めくくりとする．この Section の表題の「より良いアーキテクチャ」とは，使う目的によって決めるべきものである，としか言い様の無いものではあるが，より汎用的で，かつよりシンプルなアーキテクチャがあれば，それがより良いものであろうということは確かである．

5.6.1 Dilation パイプラインの 32bit 化

Chapter 4 に述べたアーキテクチャは，適用可能なアプリケーションの範囲を広げるという意味でたしかに強力ではあるが，大量のメモリーを使うという点で，ハードウェアの小規模化の目的に逆行する．これは，メモリは他の半導体素子に比べコストが高いことに加え，ウエハ上に大きな面積を占めるため Morphology 回路全体として見た場合，集積化の妨げとなる．また入力画像が Gray-scale であるアプリケーションに対しては，Additively separable な 構造要素しか使えないのであるから，Gray-scale 画像を扱うことが主な目的であれば，テーブル変換器アーキテクチャは固定構造要素の 1 次元アーキテクチャとあまり違わないものであるという見方もできよう．

ところで，テーブル変換器アーキテクチャの試作において，画像処理の 16bit 化という試みをしたわけだが，このことで画像処理の計算幾何学への応用という意味では，一つの足がかりとなったと考える．この方法でボロノイ図をある程度正確に作成しようとした場合，8bit の Gray-scale ではあきらかに精度が不足してしまう．このため，それまで 8bit で行っていた画像処理を 16bit でおこなうということに決めたわけだが，ではこの延長で画像処理を 32bit の精度で行って見てはどうだろうか．

そこで「Gray-scale 32Bit の画像処理は Morphology にとって有効か？」という疑問が起こる．もしこれがＣＧの分野であれば，R,G,B に 8bit ずつ割り当てた 24bit color と，8bit の透過情報が加えられた，32bit per pixel の情報量をもつ画像を扱うというのは現時点でも妥当な選択である．しかし，モノクロの画像の分野では 16bit の深さの Gray-scale さえもオーバークオリティーであろう．まして 32bit Gray-scale の画像にさほどの意味があるとは思えない．しかし．いわゆる画像処理としてではなく，計算幾何学的アプリケーションをターゲットとした距離変換を行うのであれば，32bit の Gray-scale は非常に強力な道具となり得る．

そこで，画像の入力が Binary であるようなアプリケーションに用いるため，Chapter 3 に述べた固定構造要素の 1 次元アーキテクチャを 32bit 化してみるというアプローチはどうであろうか．32bit 化した場合の 表 4.1 の続きを考えてみればわかるとおり，65535 画素の大きさの構造要素による距離変換が可能となる

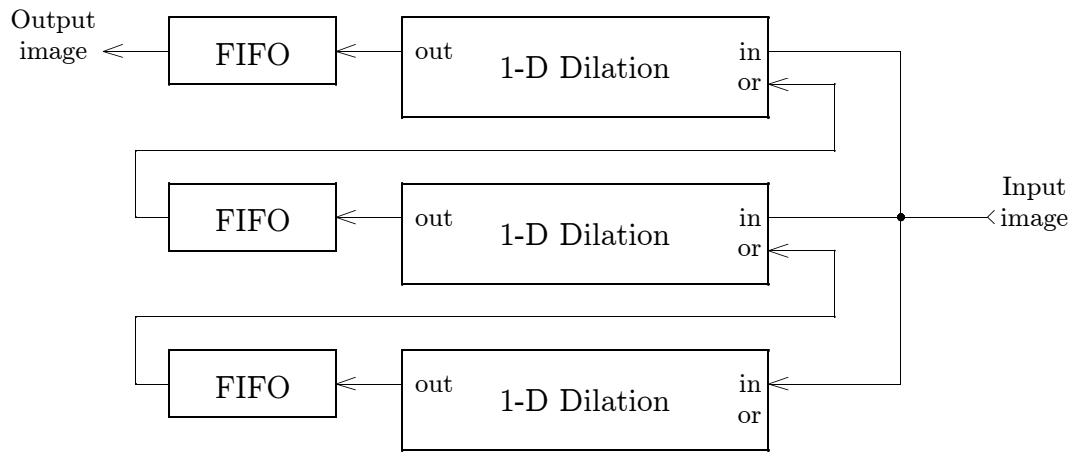


図 5.5: 1 次元 Dilation 回路の 2 次元化

のである．この大きさであれば通常の画像処理の目的においてキャパシティーを越えることは考えにくい．

5.6.2 1 次元 Dilation 回路の 2 次元化

原画像が Gray-scale の Morphology に対して，もっとも高速で制限のない手段を提供する方法が図 5.5 に示したような，1 次元 Dilation 回路を並列につないで 2 次元化する方法である．図 5.5 中の「1D Dilation 」は図 3.12 に示した 1 次元 Dilation プロセッサであり，通常の Morphology 処理に使う目的では，ここにテーブル変換器を用いる必要はおそらくないであろう．本論文に述べたアーキテクチャはすべて画像を縦・横に走査しながら 1 次元 Morphology 処理を行うものであったが，このアーキテクチャでは画像の走査は 1 方向だけであるため，むしろ Cytocomputer のマスク処理部分に 1 次元 Dilation プロセッサを取り込んだものと解釈できる．このなかで，1 次元 Dilation 回路の出力に 1 ラインの遅延をかけ，さらに次のラインの Dilation プロセッサの入力に戻すという点が 1 次元 Dilation プロセッサを重ねる方法に特有の構成方法と言える．

5.6.3 ホストとなる計算機との関係

本論文に述べたような画像処理計算機は単体で動作するものではなく，それを制御するホスト計算機が必要である．ところが外部計算機とホストとのコネクションをどのように行うかで実行性能に大きな影響が出るため，設計に当たってはホストとのインタフェースも重要な部位なる．

この点に関して画像メモリはどこに置くべきかという検討課題に対して，Chapter 4 で一つの極であるホスト計算機の主記憶を利用するという形態による実験ができたことはひとつの成果である．この方式の利点としては，画像の走査はソフト的に行われるため，使用可能な画像の大きさに制限がないことや，Morphology 以外にホストの CPU で行われるような画像処理との連携がスムーズに行える，と言ったことがあり，その点において運用形態の柔軟性が非常に高いといえる．反面欠点として，外部機器の性能よりも，ホスト計算機の I/O 性能の方がネックになりやすいということが言える．また計算機のメモリアーキテクチャは，メモリデバイス自体のページング機能により連続アドレスのデータ転送では非常に高い転送レートが実現できるものの，本来の意味でのランダムアクセスメモリーとは実態が離れてきており，今後もこの傾向は続くであろう．このことは，本論文の主題である 1 次元処理を行う際，少なからぬ障害となる．すなわち，画像をラスタ走査して計算機の主記憶に取り込んだ場合，横方向のメモリ走査は連続アドレスのため速いが，縦方向の走査にはアドレスが飛び飛びとなるため，メモリアクセス速度が数分の一になってしまうことになる．この問題に対処するには，やはり Chapter 3 で述べたようなホストとは独立した専用の画像メモリを持つことが解決の近道であろう．ただしこのメモリは，ホストとの転送の高速化のためバススロット上に配置し，ホスト側からもメモリデバイスとしてアクセスできるようにしておくことが望まれる．またこのメモリには回転機能を含めたハードウェア的なアドレス発生機能をもたせるのが良いであろう．このアドレス発生機能をプログラマブルなものとすることにより，柔軟性では本体メモリと共有する場合に劣らないものとすることができよう．

5.6.4 小型化と大規模化

表 3.3 からわかるように，1 次元アーキテクチャは現状のデバイス技術だけでも充分小さく作ることが可能であり，また集積化によって，一回の走査で処理可能な 1 次元構造要素の長さを大きくとることもできよう．また 32bit の Carry look ahead による足し算も現状において技術的な問題はないと言えるので，上述の 32bit 化案は現実的なものと言える．すなわち「計算機の小型化と処理の大規模化」は同時に実現可能である．

謝辞

本研究を進めるにあたって長い間ご指導いただいた宮川達夫先生に感謝いたします．また，本論文の審査をしてくださいました浅井秀樹先生，阿部圭一先生，大坪順次先生，中谷広正先生 に感謝いたします.

本研究の初期において熱心に指導してくださった海老澤嘉伸先生，学期の最後に指導教官となっていただいた浅井秀樹先生，に深く感謝いたします．また論文をまとめる際に高い見地からの御助言をしてくださった阿倍圭一先生，博士論文の審査委員長を務めてくださった大坪順次先生に重ねて感謝いたします.

英語論文執筆の際，多忙にもかかわらず校正を引き受けてくださった Bandi Vijyaya Gopal 博士に感謝いたします.

本研究をまとめるにあたって有意義な情報を提供してくださった 山本眞司先生（豊橋技科大）に感謝いたします.

修士課程，博士課程の５年間，奨学金の貸与をしていただきました株式会社アドバンテストに対し深謝いたします.

研究に必要な器材を無償で貸与してくださいました ソフトワークス株式会社 塩見俊夫社長に感謝いたします.

ソフトウエアやハードウエアの技術の向上を助けていただいた Nifty-serve Extender Forum のメンバーの皆様に感謝いたします.

研究用の基板の製作にあたって，要求にかなう基板 CAD をわざわざ製作してくださった 平田和貴氏に感謝いたします.

博士課程進学の動機づけをしてくださった ノイマン株式会社 岩崎倫明社長に感謝いたします．また同社においては，研究資材の調達や基板製作の際，いろいろ無理を聞いていただきましたことを感謝いたします.

References

[1] J.Serra. *Image analysis and mathematical morphology.* Academic Press, New York, 1982.

[2] S.S.Wilson L.A.Schmitt. The ais-5000 parallel processor. *IEEE Trans.Pattern Anal.& Mach.Intell.*, 10(3):320–330, May 1988.

[3] W.T.Rhodes J.M.Hereford. Nonlinear optical image filterring by time-sequential threshold decomposition. *Optical Engineering*, 27(4):274–279, 1988.

[4] O.R.Mitchell F.Y.Shih. Threshold decomposition of gray-scale morphology into binary morphology. *IEEE Trans.Pattern Anal.& Mach.Intell.*, 11(1):31–42, January 1989.

[5] J.L.Potter. Image processing on the massively parallel processor. *IEEE Computer*, pages 62–67, January 1983.

[6] S.R.Sternberg. Biomedical image processing. *IEEE Computer*, 16(1):22–34, January 1983.

[7] R.M.Haralick X.Zhuang. Morphological structuring element decomposition. *Comput.Vision,Graphics & Image Process.*, 35:370–382, 1986.

[8] J.Xu. Decomposition of convex polygonal morphological structuring elements into neighborhood subsets. *IEEE Trans.Pattern Anal.&Mach.Intell.*, 13(2):153–162, 1991.

[9] O.R.Mitchell F.Y.Shih. Decomposition of gray-scale morphological structuring elements. *Pattern recognition*, 24(3):195–203, 1991.

[10] P.T. Highnam A.L. Fisher. Real-time image processing on scan line array processors. In *Proc. IEEE Computer Soc. Workshop, Computer Architecture for Pattern Analysis and Image Database Management*, pages 484–489, 1985.

[11] B.F.Buxton J.S.Wiejak, H.Buxton. Convolution with separable masks for early image processing. *Comput.Vision,Graphics & Image Process.*, 32:279–290, 1985.

[12] P.D.Gader. Separable decompositions and approximations of grayscale morphological templates. *Image Understanding*, 53(3):288–296, May 1991.

[13] C.C.Chen J.Y.Yang. Decomposition of additively separable structuring elements with applications. *Pattern recognition*, 26(6):867–875, 1993.

[14] M.Werman J.Gil. Computing 2-d min, median, and max filters. *IEEE Trans.Pattern Anal.& Mach.Intell.*, 15(5):504–507, May 1993.

[15] A.Maglara I.Pitas. Range image analysis by using morphological signal decomposition. *Pattern recognition*, 24(2):165–181, 1991.

[16] A.N.Venetsanopoulos I.Pitas. Morphological shape decomposition. *IEEE Trans.Pattern Anal.& Mach.Intell.*, 12(1):38–45, January 1990.

[17] P.Maragos. Pattern spectrum and multiscale shape representation. *IEEE Trans.Pattern Anal.& Mach.Intell.*, 11(7):701–716, July 1989.

[18] R.W.Schafer P.Maragos. Morphological skeleton representation and coding of binary images. *IEEE Trans. Acoust. Speech & Signal Proc.*, 34(5):1228–1244, October 1986.

[19] 宮川達夫 小島昭二, 海老沢嘉伸. 大きな構造要素が使える画像の高速モルフォロジー・ハードウエア. **信学論**, J76-D-II(6):1106–1113, June 1993.

[20] 宮川達夫 小島昭二. グレイスケールモルフォロジーのための 1 次元処理アーキテクチャ. **信学論**, J79-D-II(1):53–60, January 1996.

[21] 宮川達夫 小島昭二. コード変換機を用いた gray-scale morphology ハードウエア. In **第** *26* **回画像工学コンファレンス**, volume 5-2, pages 67–70, December 1995.

[22] J.Serra. Introduction to mathematical morphology. *Comput.Vision,Graphics & Image Process.*, 35:283–305, September 1986.

[23] X.Zhuang R.M.Haralick, S.R.Sternberg. Image analysys using mathematical morphology. *IEEE Trans. Pattern Anal. and Mach.Intell*, 9(4):532–550, July 1987.

[24] R.W.Schafer P.Maragos. Morphological skeleton representation and coding of binary images. *IEEE Acoustics, Speech, and Signal Processing*, 34(5):1228–1244, October 1986.

[25] W.Martin T.E.Hutchinson, K.P.White. Human-computer interaction using eye-gaze input. *IEEE Trans.Syst.,Man.&Cybern.*, 19(6):1527–1534, 1989.

[26] S.Kojima T.Ushikubo T.Miyakawa Y.Ebisawa, K.Kaneko. Non-invasive eye-gaze position detecting method used on man/machine interface for the disabled. In *Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications*, pages 374–378, February 1991.

[27] E.K.Wong R.Lin. Logic gate implementation for gray-scale morphology. *Pattern Recognition Letters*, 13:418–487, 1992.

[28] S.R.Sternberg. Grayscale morphology. *Comput.Vision,Graphics & Image Process.*, 35:333–355, 1986.

[29] G.Borgefors. Distance transformations in digital images. *Comput.Vision,Graphics & Image Process.*, 34:334–371, 1986.

[30] 杉原 厚吉. 計算幾何学的手法と画像解析 - ボロノイ線図を中心として. **情報処理**, 30(9):1067–1075, September 1989.

著者による論文

[1] 小島昭二, 海老澤嘉伸, 宮川達夫. 大きな構造要素が使える画像の高速モルフォロジー・ハードウェア. **信学論**, J76-D-II(6): 1106–1113, June 1993.

[2] Shoji Kojima,Yoshinobu Ebisawa,Tatsuo Miyakawa. Fast Morphology Hardware Using Large-sized Structuring Element. *Systems and Computer in Japan*, 25(6): 41–49, June 1994.

[3] 小島昭二, 宮川達夫. Gray-scale 画像の Morphology ハードウエア. **第 24 回画像工学コンファレンス**, 12-1: 383–386, December 1993.

[4] Shoji Kojima, Tatsuo Miyakawa. Morphological Processor for Gray-scale Image. *The 3rd International Conference on Automation, Robotics and Computer Vision*, WP1.1: 148–152, November 1994.

[5] 小島昭二, 宮川達夫. グレイスケール・モルフォロジーのための 1 次元処理アーキテクチャ. **信学論** J-79-D-II(1): 53–60, January 1996.

[6] 小島昭二, 宮川達夫. コード変換機を用いた Gray-scale Morphology ハードウエア. **第 26 回画像工学コンファレンス** 5-2: 67–70, December 1995.

[7] 小島昭二, 宮川達夫. テーブル変換器を用いた Gray-scale Morphology ハードウェアとボロノイ図への応用. **静岡大学大学院電子科学研究科研究報告 第 17 号** 1996（掲載予定）

共著者による論文

[1] Yoshinobu Ebisawa,Koichi Kaneko,Shoji Kojima,Takashi Ushikubo,Tatsuo Miyakawa. Non-invasive Eye-gaze Position Detecting Method Used on Man/Machine Interface for The Disabled. *Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications* 374–378, February 1991.

口頭発表など

[1] Yoshinobu Ebisawa,Koichi Kaneko,Shoji Kojima,Takashi Ushikubo,Tatsuo Miyakawa. Video-based Noninvasive Eye-gaze Point Detection Method Used on Nonverbal Communicator for the Disabled. *Medical and Biological Engineering and Computing*, 29: p.1055, July 1991.

[2] 兼子浩一，海老澤嘉伸，小島昭二，牛窪隆志，宮川達夫. 身体障害者介助システム用 非接触注視点検出法の提案. **1991 年電子情報通信学会春季全国大会**, A-253: p.1-255, March 1991.

[3] 海老澤嘉伸，兼子浩一，小島昭二，牛窪隆志，宮川達夫. 身体障害者介助システムに用いる非接触凝視点検出法. **第 30 回日本エム・イー学会大会**, C2-32: p.202, July 1991.

Appendix A　回転不変，かつ１次元化可能な構造要素

１次元化可能で，かつ回転不変な形状の構造要素は２次曲面のみであることは次のように示される．まず Additively separable な関数は式（A.1）で表される．
`Additively separable function`

$$z = k(x, y) = h(x) + v(y) \tag{A.1}$$

式（A.1）は２次元の Gray-scale 構造要素の Additively separable 性を表現した Morphology の式である式（3.7）を関数の形に置き換えたものと解釈することができる．ここで $h(x)$ は y に対して独立であり，$v(y)$ は x に対して独立な関数である．

次に図 A.1 に示すように，構造要素の形状である式（A.1）の $k(x, y)$ を極座標上の関数 $z = f(r, \theta)$ で表すことを考える．もし f が z 軸に対して回転対称な形状であれば，f はθに独立であるから，構造要素の回転対称性は f を用いて式（A.2）のように表すことができる．

$$\begin{aligned} z &= f(r) \\ r &= \sqrt{x^2 + y^2} \end{aligned} \tag{A.2}$$

ここで式（A.1）と式（A.2）は同じ形状の構造要素をそれぞれデカルト座標系と極座標系で表現したものであるから，x, y がどのように変化しても z の値はいつでも等しくなければならない．よって式（A.3）のような恒等式が成り立つ．

$$f(\sqrt{x^2 + y^2}) \equiv h(x) + v(y) \tag{A.3}$$

恒等式（A.3）を解くためには h, v, f をそれぞれ式（A.4）のようなべき級数で表して式（A.3）に代入し，両辺の係数を比較する．

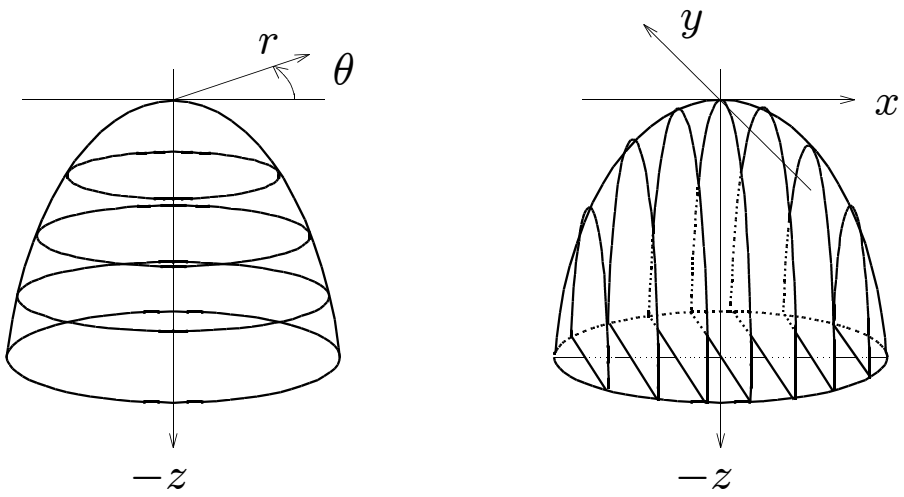


図 A.1: x, y 形式から r, θに変換

$$\begin{aligned} h(x) &= h_0 + h_1 x + h_2 x^2 + h_3 x^3 \cdots \\ v(y) &= v_0 + v_1 y + v_2 y^2 + v_3 y^3 \cdots \\ f(r) &= f_0 + f_1 r + f_2 r^2 + f_3 r^3 \cdots \end{aligned} \tag{A.4}$$

式（A.4）の $f(r)$ を式（A.3）に代入して整頓すると式（A.5）左辺のように，root を含む項と因数に $x \cdot y$ を含む項とそれ以外の項に分けることができる．

$$\begin{aligned} & f_0 + f_2(x^2+y^2) \\ & + \sqrt{x^2+y^2} \cdot \sum_{n=0}^{\infty} f_{2n+1}(x+y)^n \\ & + \sum_{n=2}^{\infty} f_{2n}(x^2+y^2)^n \end{aligned} \equiv \begin{aligned} & h_0 + h_1 x + h_2 x^2 + h_3 x^3 \cdots \\ & + v_0 + v_1 y + v_2 y^2 + v_3 y^3 \cdots \end{aligned} \tag{A.5}$$

式（A.5）は独立変数 x, y についての恒等式であり右辺が x, y のべき級数であることから x, y の 1 次と 3 次以上の係数はすべてゼロでなければならず，その結果，式（A.6）を得る．

$$\begin{aligned} f_0 + f_2(x^2+y^2) &\equiv h_0 + v_0 + h_2 x^2 + v_2 y^2 \\ \therefore f_0 &= h_0 + v_0, f_2 = h_2 = v_2 \end{aligned} \tag{A.6}$$

よって 1 次元化可能かつ回転対称な構造要素は，

$$z = a(x^2+y^2) + b \qquad \text{(a,b は定数)} \tag{A.7}$$

のような２次曲面のみに限定されることが示された．さらに Section 3.2.2 で議論したオーバーフローを防ぐための構造要素の条件，すなわち $z_{max} = 0$ を考慮すると式（A.7）の定数は $b = 0, a < 0$ となる．よって以上に得られた結果を再び式（A.1）に戻せば，式（A.8）を得る．

$$k(x, y) = a(x^2 + y^2) \qquad (a < 0) \tag{A.8}$$

Appendix B　１次元Dilation アルゴリズムの導出

ここではDilation の定義式から１次元 Dilation 回路の信号フローを導出する．まず，１次元 Dilation 回路の設計の出発点になるのは Gray-scale Dilation の定義式（B.1）であるが，ハードウェアとして具現化する必要上，原画像 f と構造要素の定義域を明確にしておく．すなわち，原画像 f，構造要素 k 共に独立変数が整数の離散信号とし，

$$(f \oplus k)(x) = \underset{\substack{0 \le x \le x_{max} \\ 0 \le z < M}}{max} \{f(x-z) + k(z)\} \tag{B.1}$$

において，構造要素は M 画素の大きさを持つものとし，変数 z の範囲は $z = 0 \sim M-1$ と決める．画像 $f(x)$ は $0 \le x \le x_{max}$の範囲で定義され，$x < 0, x_{max} < x$ の範囲では $f(x) = 0$ とする（図 B.2）．式（ B.1）は原画像 f と構造要素 k の間の演算であり，より直接的には，図 B.1 のように表すことができ，信号の流れを考えると，図 B.3 に示した信号流れ図を得る．

$$(f \oplus k)(x) = max \begin{cases} f(x) + k(0) \\ f(x-1) + k(1) \\ f(x-z) + k(z) \\ \qquad \vdots \\ f(x-M+1) + k(M-1) \end{cases}$$

図 B.1: 一次元 Dilation のアルゴリズム

図 B.3 において［D］は遅延素子であり，入力された離散信号 (Gray-scale 値) を x 方向 (時間方向) に対し１段階遅らせる働きをする．そして［Max］は入力さ

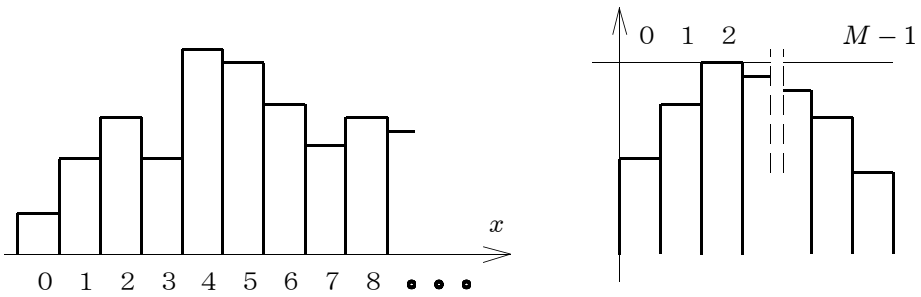


図 B.2: 原画像 f と構造要素 k

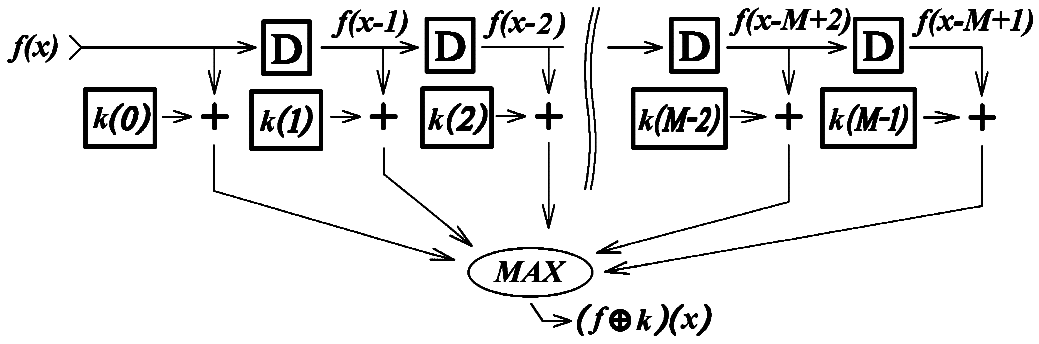


図 B.3: 信号流れ図（直接法）

れた Gray-scale 値すべてを比較して最大値を出力する働きをする．ところがこの「多入力 MAX」を実際の回路として実現することは，比較する値の個数が多い場合困難となるためこのような論理を用いないで 1 次元 Dilation 演算を実行する方法について次に考える．すなわち「多入力 Max」を用いない計算方法を得るため，再び式（ B.1）から出発して順を追いながらアルゴリズムの変形を行い，最終的にハードウェア化可能なアルゴリズムを求める．

まず，図 B.1 を for loop を使ったアルゴリズムに書き換えると図 B.4 のようになる．ここで最大値を求める MAX 演算は式 （B.2）に示した，比較・代入演算に置き換えている．

```
1  for( x := 0  to  x_max )                          — loop 1
2  {    S_max := 0
3  {    for( z := 0  to  M − 1 )                     — loop 2
4       { S_max := larger(f(x − z) + k(z), S_max)
5       }
6  (f ⊕ k)(x) := S_max
7  }
```

図 B.4: 一次元 Dilation のアルゴリズム

$$larger(A, B) = \begin{cases} A & (A > B) \\ B & (A \leq B) \end{cases} \tag{B.2}$$

図 B.4 において，入力信号 f，出力信号 $f \oplus k$ をともに時系列信号と考えることができる．これは　loop1 において座標 x は 0 から整数刻みで増加する時刻として考えることができ，途中の値を飛ばしたり，あるいは逆戻りしたりすることはないと仮定できるからである．すなわち図 B.1 に示した計算手順においては x がどのような値を取ろうとも $f \oplus k$ を求めることが可能であるが，図 B.4 のように変数 x の振る舞いを限定することにより，逆に計算のアルゴリズムに自由度を持たせることができる．この枠組みで以下にアルゴリズムの変形を行う．

まず，図 B.4 の 4 行目の変数 S_{max} を $n(x)$ に変更することで図 B.5 のように書き換える．配列 $n(x)$ の初期化は loop1 の外で行うことができる．

```
1  n(0 ~ x_max + M − 1) := 0                          — 配列の初期化
2  for( x := 0  to  x_max )                                — loop 1
3  {   for( z := 0  to  M − 1 )                            — loop 2
4      { n(x) := larger(f(x − z) + k(z), n(x))
5      }
6      (f ⊕ k)(x) := n(x)
7  }
```

図 B.5: 一次元 Dilation のアルゴリズム

図 B.5 のアルゴリズムの中では中間変数 $n()$ が最大値を求めるための変数となっている．この中間変数 $n()$ は $n(0)$ から $n(x_{max} + M - 1)$ までの，入力信号の長さに等しい $x_{max} + M$ 個の要素を持つ．つぎに図 B.5 中の 4 行目の $f(x - z)$ を $f(x)$ に置き換えることを考える．これは 4 行目の x を $x + z$ に置き換えることで可能となる．

```
1  n(0 ~ x_max + M − 1) := 0                          — 中間変数の初期化
2  for( x := 0  to  x_max )                                — loop 1
3  {   for( z := 0  to  M − 1 )                            — loop 2
4      { n(x + z) := larger(f(x) + k(z), n(x + z))
5      }
6  }
7  (f ⊕ k)(x ~ x_max) := n(x ~ x_max)
```

図 B.6: 一次元 Dilation のアルゴリズム

図 B.6 は 2 重の loop によって中間変数 $n()$ を部分的に繰り返し更新していき，最終的な $n(0 \sim x_{max})$ の値が Dilation となるアルゴリズムである．図 B.1 と図 B.6 は Dilation の計算方法としては対をなすアルゴリズムと解釈することができ，図 B.1 は explicit 形式，図 B.6 は implicit 形式と呼ぶことができよう．すなわち，図 B.1 は独立変数 x がどのような値を取ろうとも $f \oplus k(x)$ が即座に求まる陽関数である

のに対し，図 B.6 は中間変数 $n(x)$ を求めるためにそれ自体の値を必要とする陰関数となっているということである．

アルゴリズム図 B.1，図 B.6 は双方ともソフトウエア的に (直列演算で) 実行する限りにおいては，どちらも同じ計算量が必要であり，冗長性は残されていない．しかし図 B.6 に示したアルゴリズムでは多入力 MAX を必要とせず，２つの値を比較してどちらか大きい方を出力する関数「$larger$」を用いた比較計算の繰り返しのみで Dilation を求めることができるため，図 B.6 の方がハードウェア向きということができる．以下に，ハードウェアとしての実装を容易にするためのアルゴリズムの最適化をさらに試みる．

図 B.6 では中間変数 $n()$ の値がすべて確定した後（行 7）で変数の移し替えを行っているが，loop2 が完了した時点ですでにそのときの x における $n(x)$ の値は確定しており，loop1 によるそれ以降の繰り返しでは変化を受けない．このことに注意すると図 B.6 に代えて図 B.7 を得る．

```
1   n(0 ~ x_max + M − 1) := 0
2   for( x := 0  to  x_max )                          — loop 1
3   {   for( z := 0  to  M − 1 )                      — loop 2
4       { n(x + z) := larger(f(x) + k(z), n(x + z))
5       }
6       f ⊕ k := n(x)
7   }
```

図 B.7: 一次元 Dilation のアルゴリズム

図 B.6 と図 B.7 の違いは最後の部分のみにある．すなわち図 B.6 では，loop1 の計算終了後まで Dilation 結果が得られない形式となっているが，図 B.7 では $n(x)$ が確定した時点で Dilation 結果が逐次求まる点が異なる．

次に中間変数 $n()$ の削減を考える．中間変数 n は原画像の１辺の長さに等しいが，一般に画像の Morphology 処理では構造要素の大きさは，たかだか原画像の数分の１から数十分の１である場合が普通であり，もし中間変数の個数を１次元構造要素の長さにできれば数値を格納する容量を大幅に削減できる．そのような中

間変数の削減は次のようなアルゴリズムの変形によって可能となる．まず図 B.7 の loop2 に注目すると，$n()$ は $n(x)$ から $n(x+M-1)$ までの M 個の要素のみしか変化を受けず，さらに loop2 が終了した直後の (行 5) ですでに座標 x における Dilation 結果が得られることがわかる．この事実を利用すれば，中間変数の個数を M 個に減らすことが可能となる．すなわち新たに $m(0)$ から $m(M-1)$ までの M 個の要素を持つ中間変数 $m()$ を定義し，$n(x+z)$ の代わりに $m(z)$ を用いることによって図 B.7 を図 B.8 に書き改める．

```
1   m(0 ~ M − 1) := 0
2   for( x := 0  to  x_max )                          — loop 1
3   {   for( z := 0  to  M − 1 )                      — loop 2
4       { m(z) := larger(f(x) + k(z), m(z))
5       }
6       (f ⊕ k)(x) := m(0)
7       for( z := 0  to  M − 2 )                      — loop 3
8       { m(z) := m(z + 1)
        }
    }
```

図 B.8: 一次元 Dilation のアルゴリズム

図 B.8 のアルゴリズムでは，loop 3 で実行しているように中間変数 $m()$ をシフトさせる必要がある．なぜなら図 B.7 の loop 2 内のみに注目した場合，loop1 による x の増加にしたがってあたかも $n(x)$ が $n(x+1)$ にシフトしたように「見える」ため，これに代えて実際の変数シフトが必要となるためである．しかしこの loop3 による変数シフトは，(行 4) の代入文を工夫することにより，loop2 内で同時に実行することが可能である．すなわち中間変数 $m()$ の要素を 1 つ増やして $m(0)$ から $m(M)$ までの，$M+1$ 個の要素を持つものとして図 B.8 を図 B.9 のように書き改める．

図 B.8 と図 B.9 の主だった違いは図 B.9 の (行 4) において変数 $m()$ のシフトを loop2 内で順次実行する点であり，図 B.9 の loop2 の終了時においてすでに

```
1   m(0 ~ M) := 0
2   for( x := 0  to  x_max )                          — loop 1
3   {    for( z := 0  to  M − 1 )                     — loop 2
4        { m(z) := larger(f(x) + k(z), m(z + 1))
5        }
6        (f ⊕ k)(x) := m(0)
    }
```

図 B.9: 一次元 Dilation のアルゴリズム

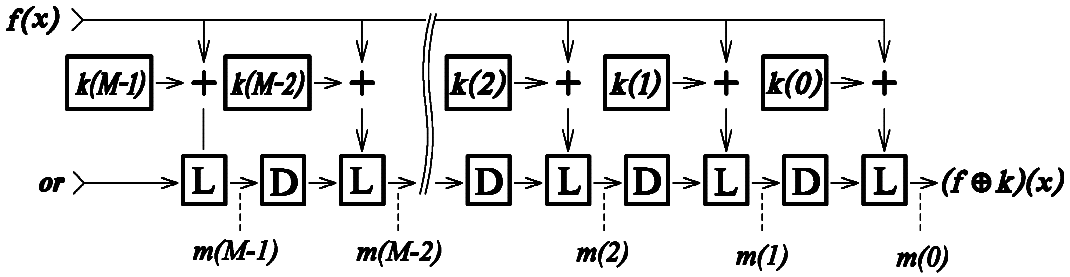


図 B.10: 1 次元 Dilation 回路の信号流れ図

図 B.8 の loop3 と同様の変数シフトが完了している．中間変数 $m(M)$ が一か所だけ増えた部分は，シフトのための作業変数であると解釈できる．

図 B.6 と図 B.9 のアルゴリズムを比較すると，実行される演算の回数は双方とも（$x_{max} \times M$）回となるため逐次処理による計算ではどちらも同じだけの計算時間がかかることになる．しかし図 B.9 のアルゴリズムからは図 B.10 のようなパイプライン式の信号流れ図が得られるため，演算の並列化が可能となる．ここに［D］は遅延素子であり［L］は入力された 2 つの数値のうち大きい方を出力する素子，すなわち $larger(A,B)$ を計算する素子である．この信号流れ図は図 B.9 のアルゴリズムの loop2 を並列化したものであると言え，その外側の loop1 は入出力信号の変化の時間刻みを意味すると解釈できる．

Appendix C　試作機の回路図

C.1　視線方向検出装置（Binary Morphology 回路）

ここに示す回路の中に Chapter 2 に示したアーキテクチャの Morphology 処理回路を含んでいる．これは画像処理の医用応用の研究として，Bright eye 画像（赤外線の網膜反射像）から視線方向を検出するための装置に Morphology 回路の試作機を組み込んで，カメラから取り込んだ画像の前処理に用いたものである [26]．カメラのサーボ系を含む装置全体は図 C.1 のようになっている．

この装置は CCD カメラから画像を取り込み，ノイズを消すために Binary Morphology 処理を行い，その結果から視線方向の決定要素となる網膜反射像と角膜反射像の座標を検出し，パソコンにその座標を伝えるという働きをする．このため，Morphology 処理により得られた画像そのものをパソコンに取り込む機能は持っていない（図 2.10 は，この回路からイメージグラバでプロービングすることによって得た）．この回路は CCD カメラの同期信号にシンクロする形で処理を行い結果をパソコンの I/O に見せるという形態をとっているため，ソフトウエア側から見た場合このハードウェアは計測器のような位置付けとなる．またこのシステムにはカメラの位置とフォーカスおよび絞りを制御するための機構も含まれている．

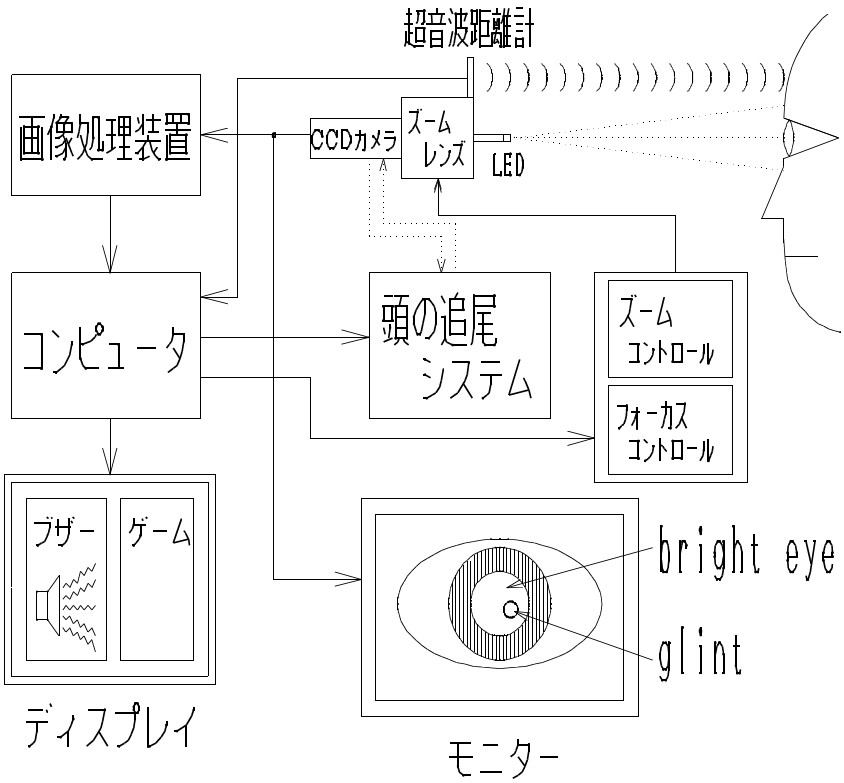


図 C.1: 装置全体のブロック図

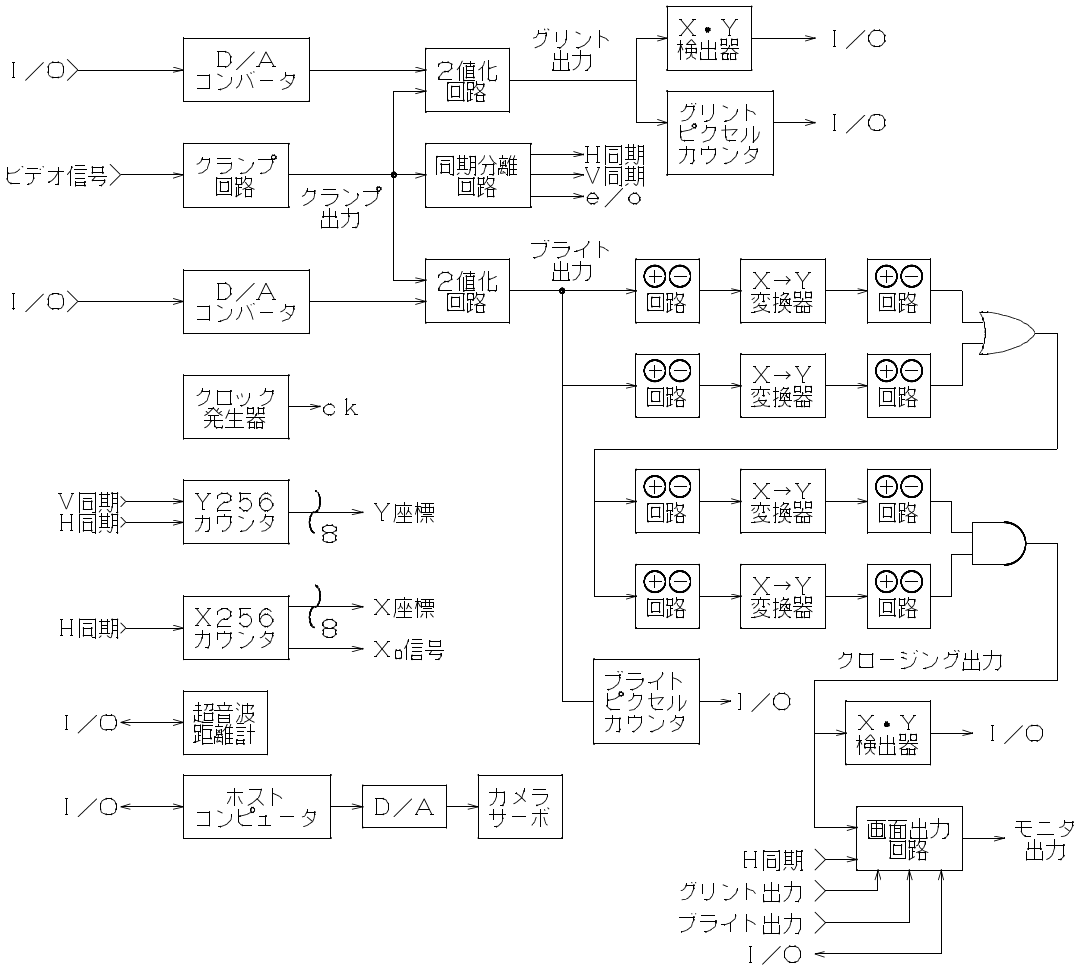

I／O
D／A
コンバータ
2値化
回路
グリント
出力
X・Y
検出器
I／O
グリント
ピクセル
カウンタ
I／O
ビデオ信号
クランプ
回路
クランプ
出力
同期分離
回路
H同期
V同期
e／o
ブライト
出力
I／O
D／A
コンバータ
2値化
回路
回路
X→Y
変換器
回路
回路
X→Y
変換器
回路
クロック
発生器
c k
回路
X→Y
変換器
回路
回路
X→Y
変換器
回路
V同期
H同期
Y256
カウンタ
8
Y座標
H同期
X256
カウンタ
8
X座標
X0信号
ブライト
ピクセル
カウンタ
I／O
クロージング出力
I／O
超音波
距離計
X・Y
検出器
I／O
I／O
ホスト
コンピュータ
D／A
カメラ
サーボ
画面出力
回路
モニタ
出力
H同期
グリント出力
ブライト出力
I／O

図 C.2: 系統図

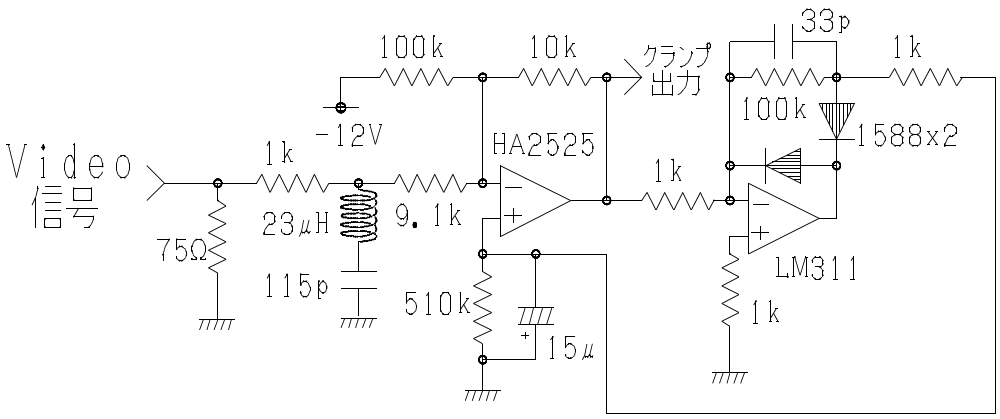


図 C.3: Video cramp 回路

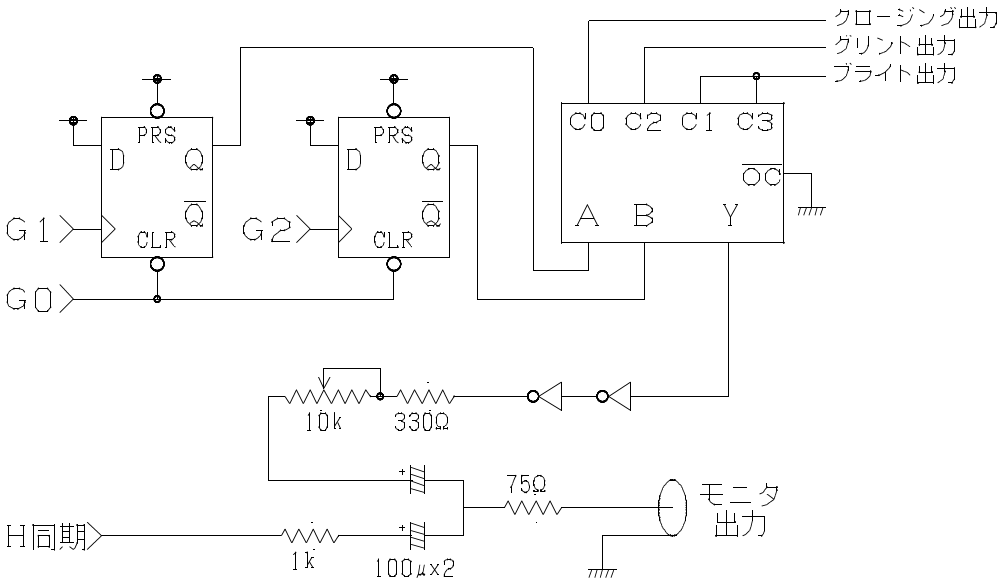


図 C.4: 画像出力回路

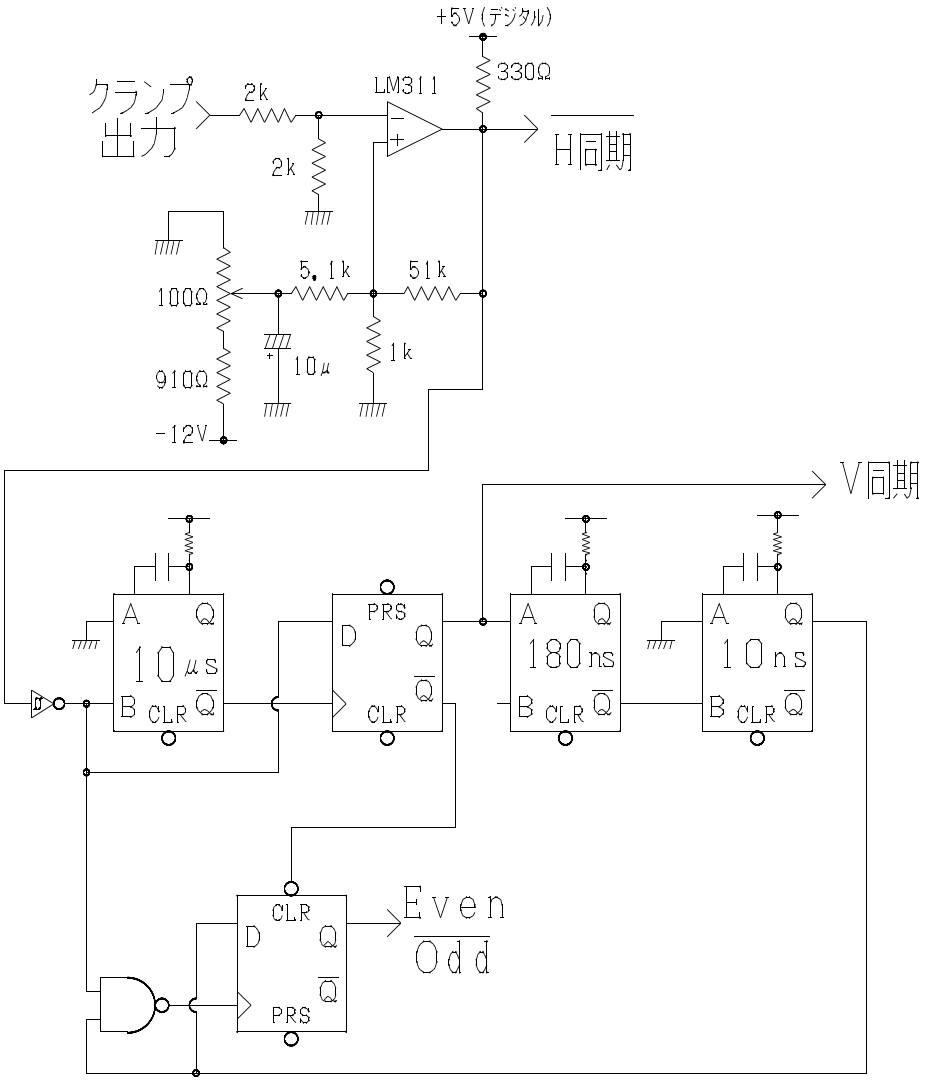

+5V(デジタル)
330Ω
クランプ
出力
2k
LM311
H同期
2k
100Ω
5.1k
51k
10μ
1k
910Ω
-12V
V同期
A Q
10μs
B CLR Q
PRS
D Q
CLR Q
A Q
180ns
B CLR Q
A Q
10ns
B CLR Q
CLR
D Q
PRS Q
Even
Odd

図 C.5: H-V 同期回路

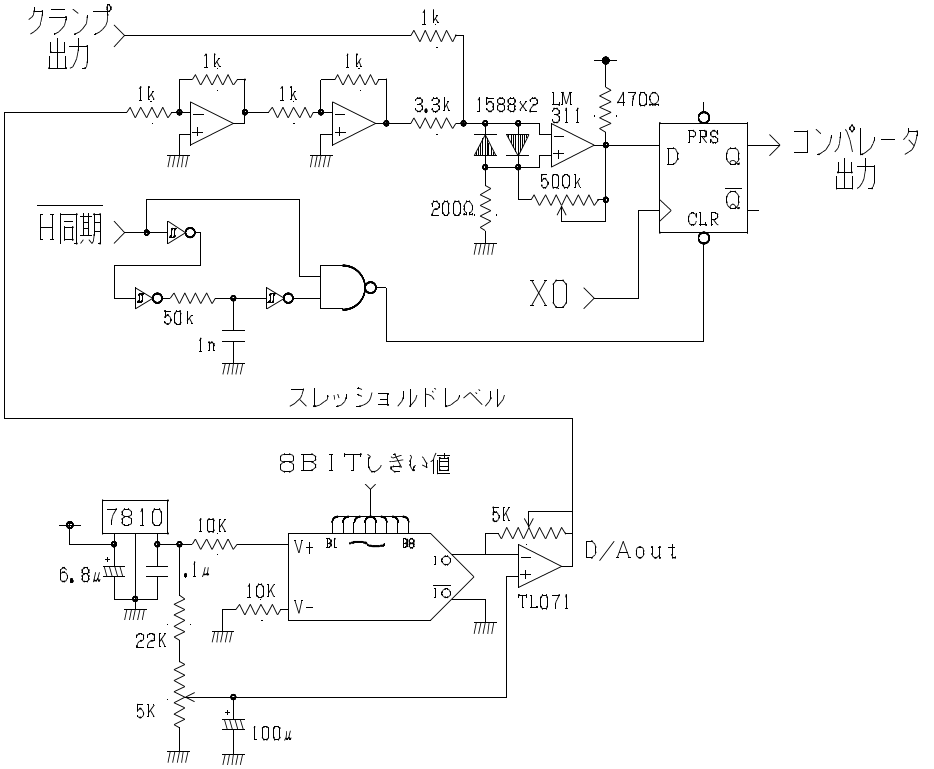


図 C.6: ２値化回路

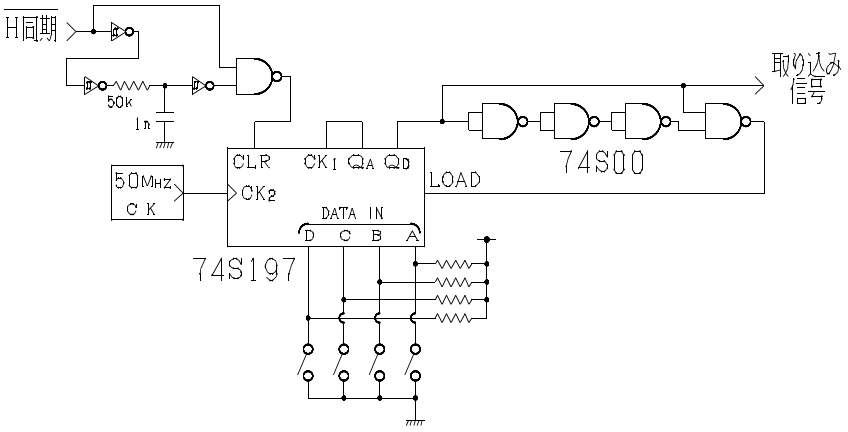


図 C.7: clock 発生回路

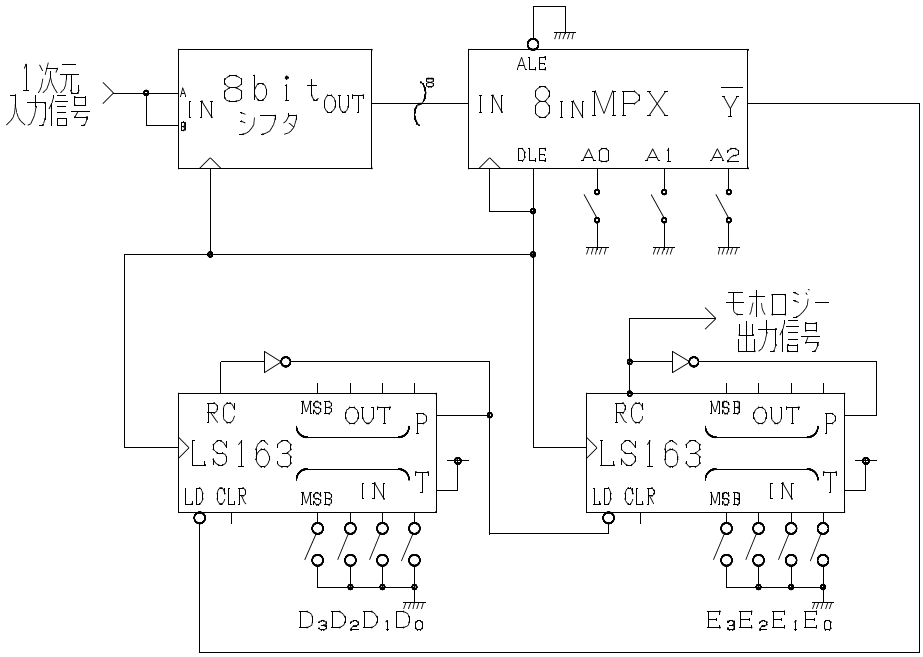


図 C.8: Dilation Erosion 回路

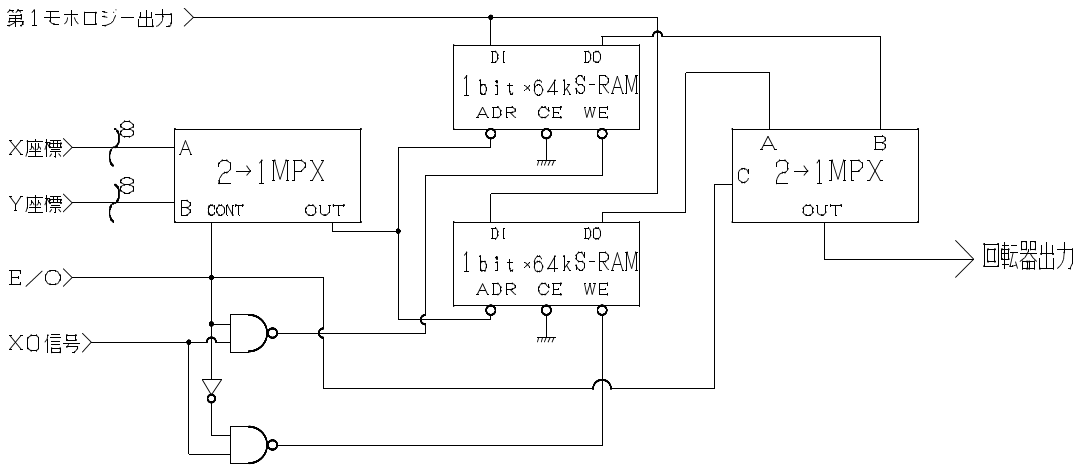


図 C.9: x-y 変換回路

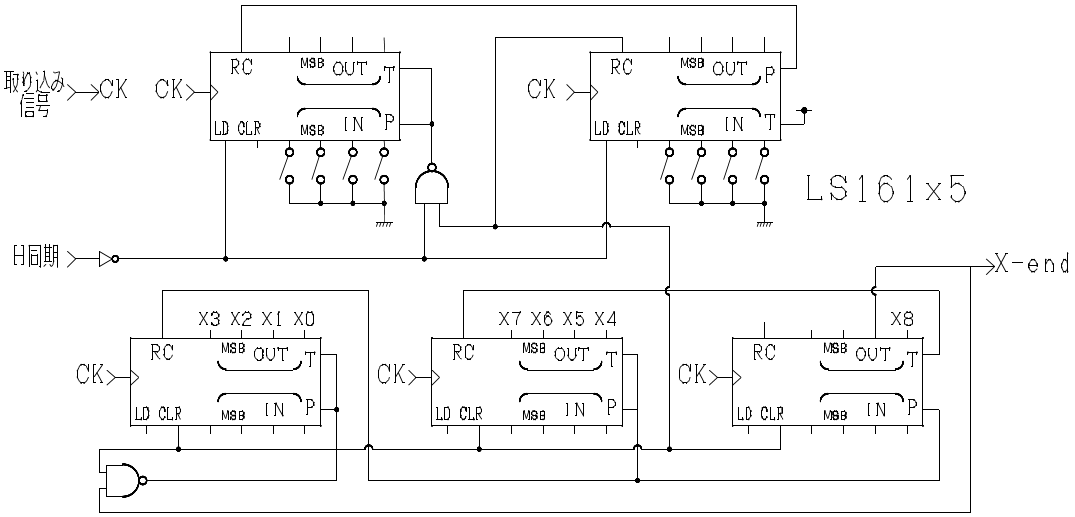


図 C.10: x 方向 512 カウンタ

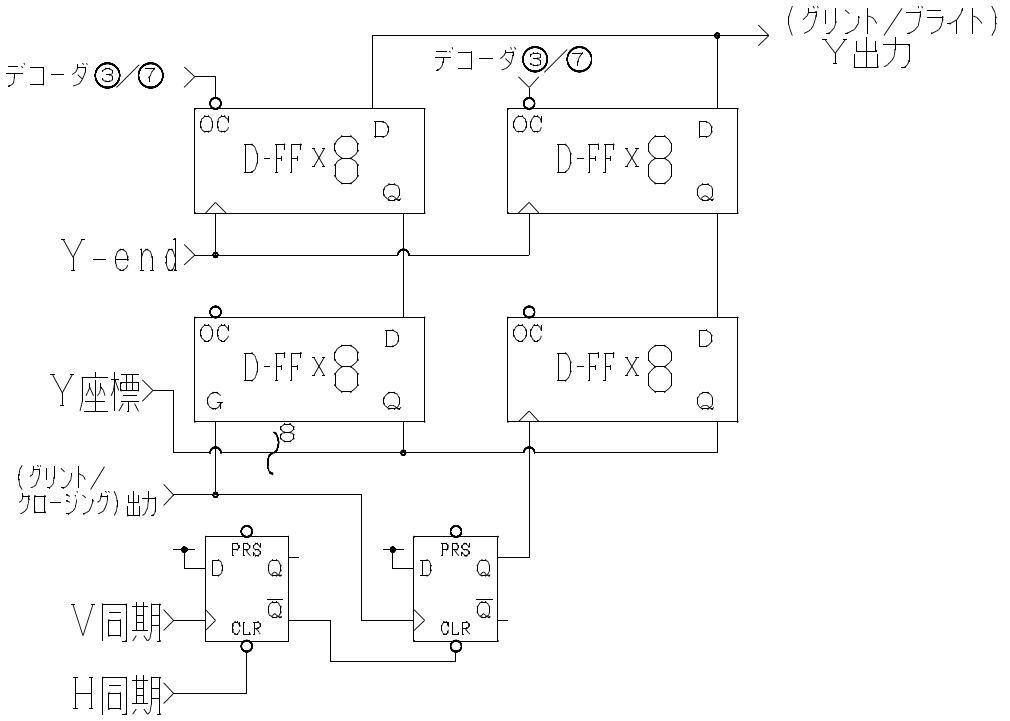


図 C.11: グリント y 座標検出回路

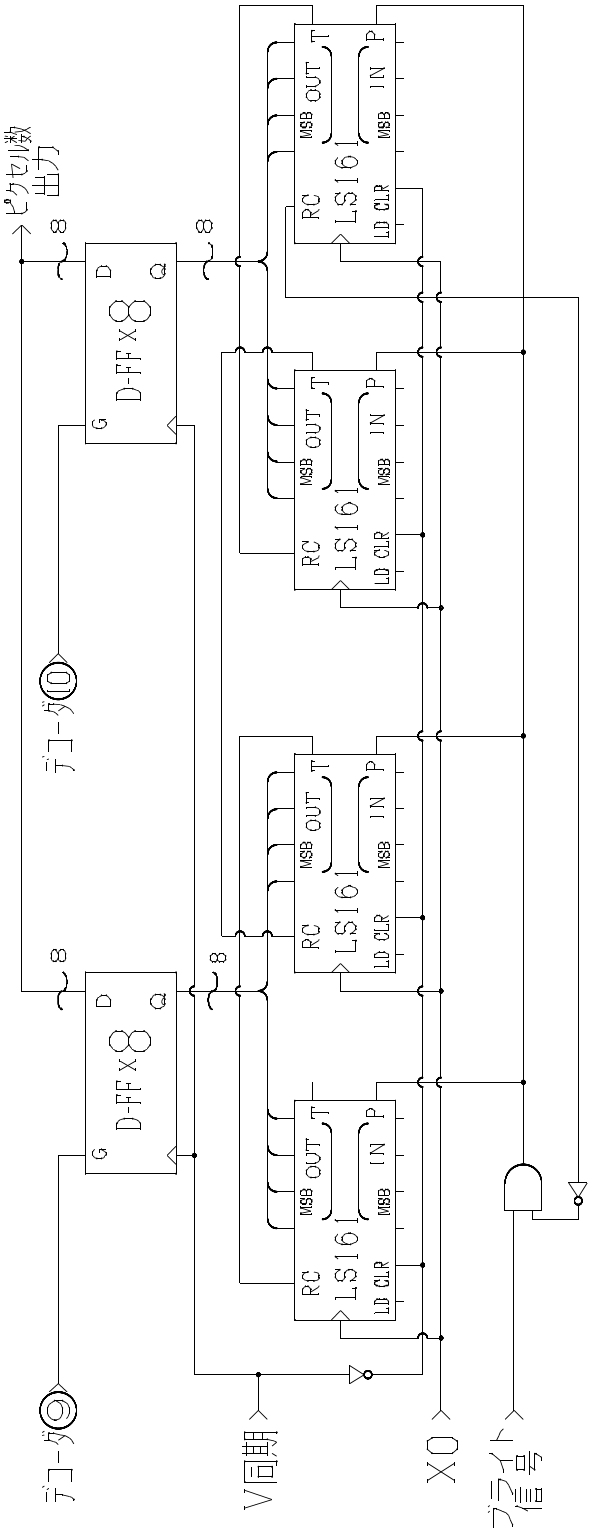


図 C.12: ブライトアイ画素カウンタ

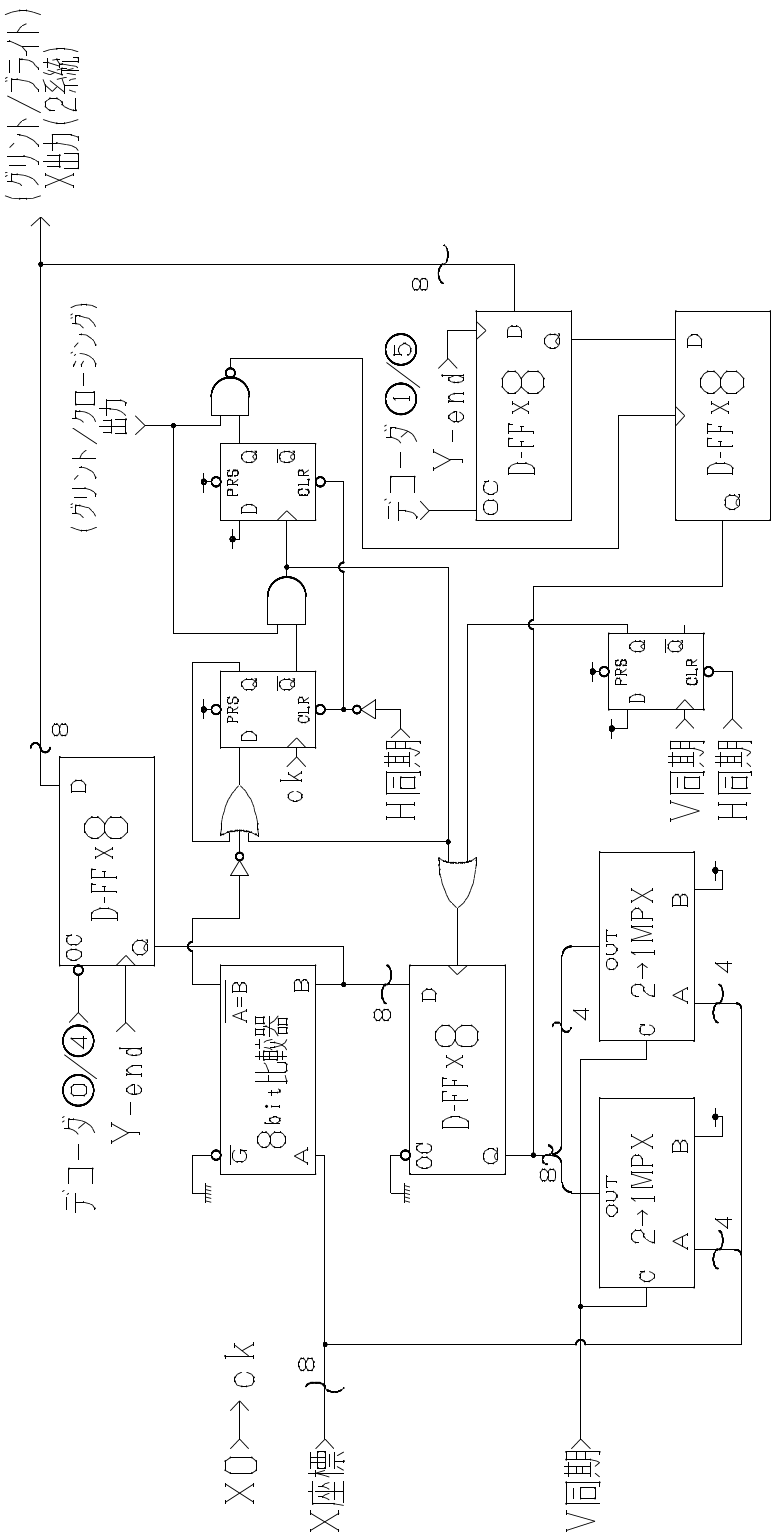


図 C.13: グリント x 座標検出回路

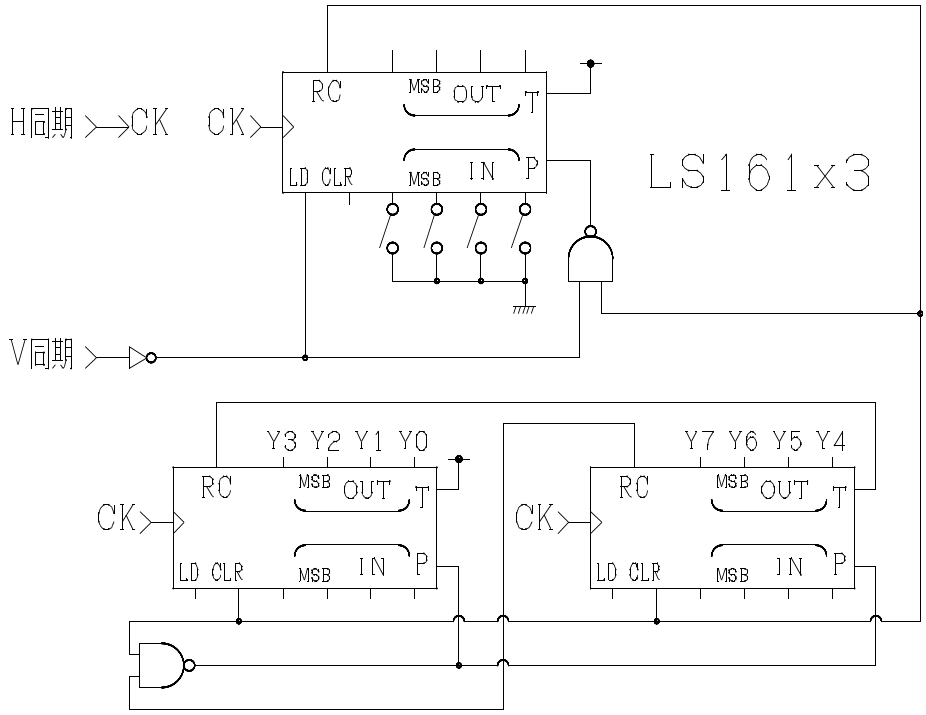


図 C.14: y 方向 256 カウンタ

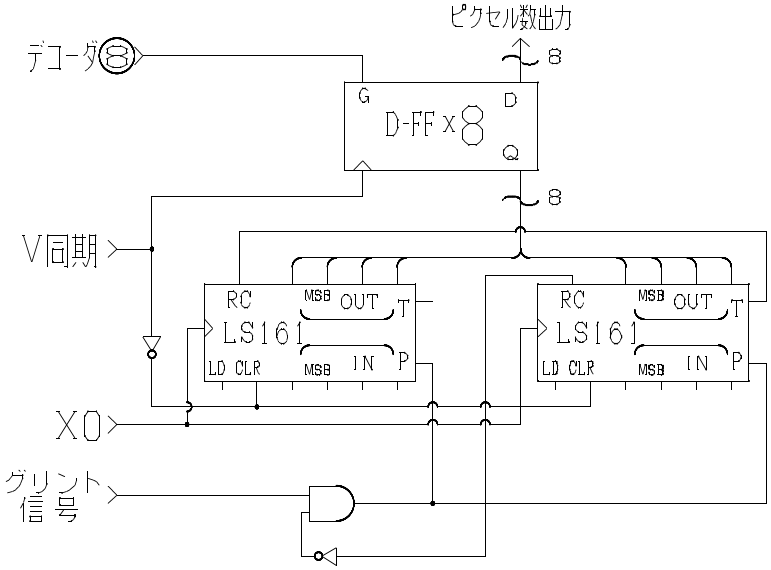


図 C.15: グリント画素カウンタ

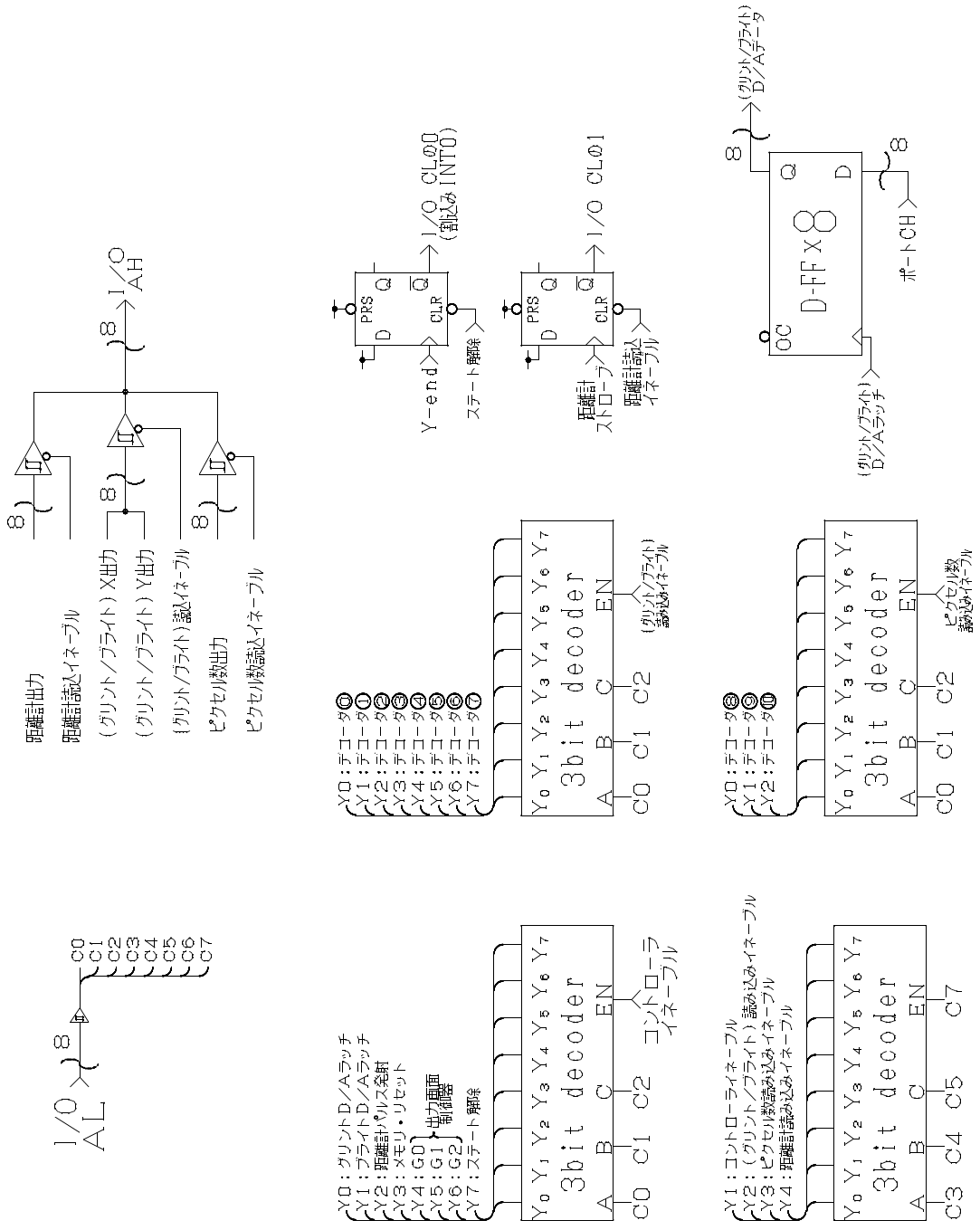

I/O AL
8
C0
C1
C2
C3
C4
C5
C6
C7
コントローラ イネーブル
3bit decoder
A B C EN
C0 C1 C2
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
Y0：グリントD／Aラッチ
Y1：プライトD／Aラッチ
Y2：距離計パルス発射
Y3：メモリ・リセット
Y4：G0
Y5：G1
Y6：G2
出力画面制御器
Y7：ステート解除
Y1：コントローライネーブル
Y2：（グリント／プライト）読み込みイネーブル
Y3：ピクセル数読み込みイネーブル
Y4：距離計読み込みイネーブル
C3 C4 C5 C7
距離計出力
距離計読込イネーブル
（グリント／プライト）X出力
（グリント／プライト）Y出力
（プリント／プライト）読込イネーブル
ピクセル数出力
ピクセル数読込イネーブル
I/O AH
Y0：テコータ0
Y1：テコータ1
Y2：テコータ2
Y3：テコータ3
Y4：テコータ4
Y5：テコータ5
Y6：テコータ6
Y7：テコータ7
（グリント／プライト）読み込みイネーブル
ピクセル数読み込みイネーブル
PRS
D
Q
CLR
Y-end
ステート解除
I/O CL0.0 （読込み INT0）
距離計ストローブ
距離計読込イネーブル
I/O CL0.1
D-FF X 8
OC
（グリント／プライト）D／Aラッチ
（グリント／プライト）D／Aデータ
ポートCH
Y0：テコータ0
Y1：テコータ1
Y2：テコータ2

図 C.16: I/O

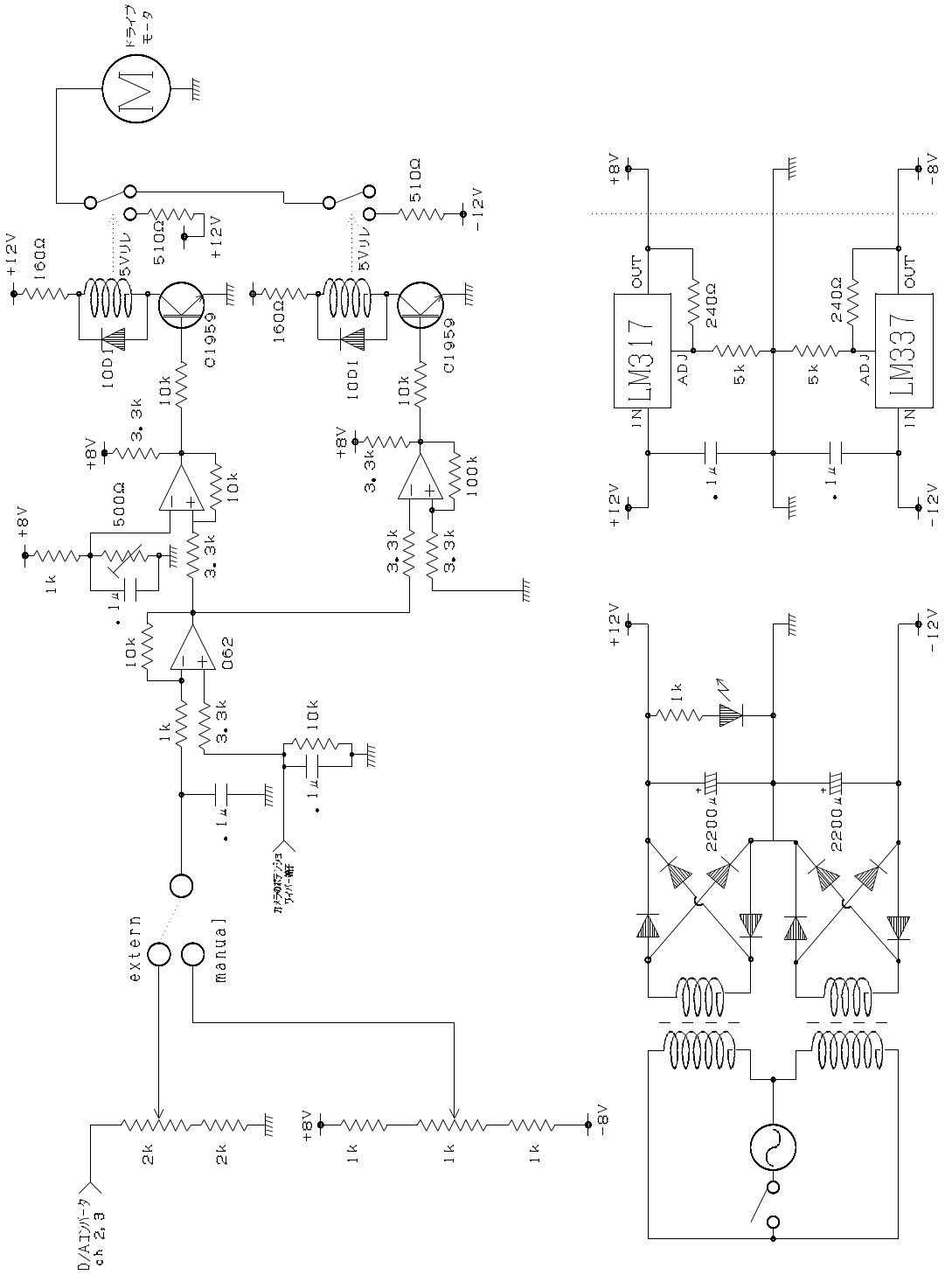


図 C.17: CCD カメラ コントローラ

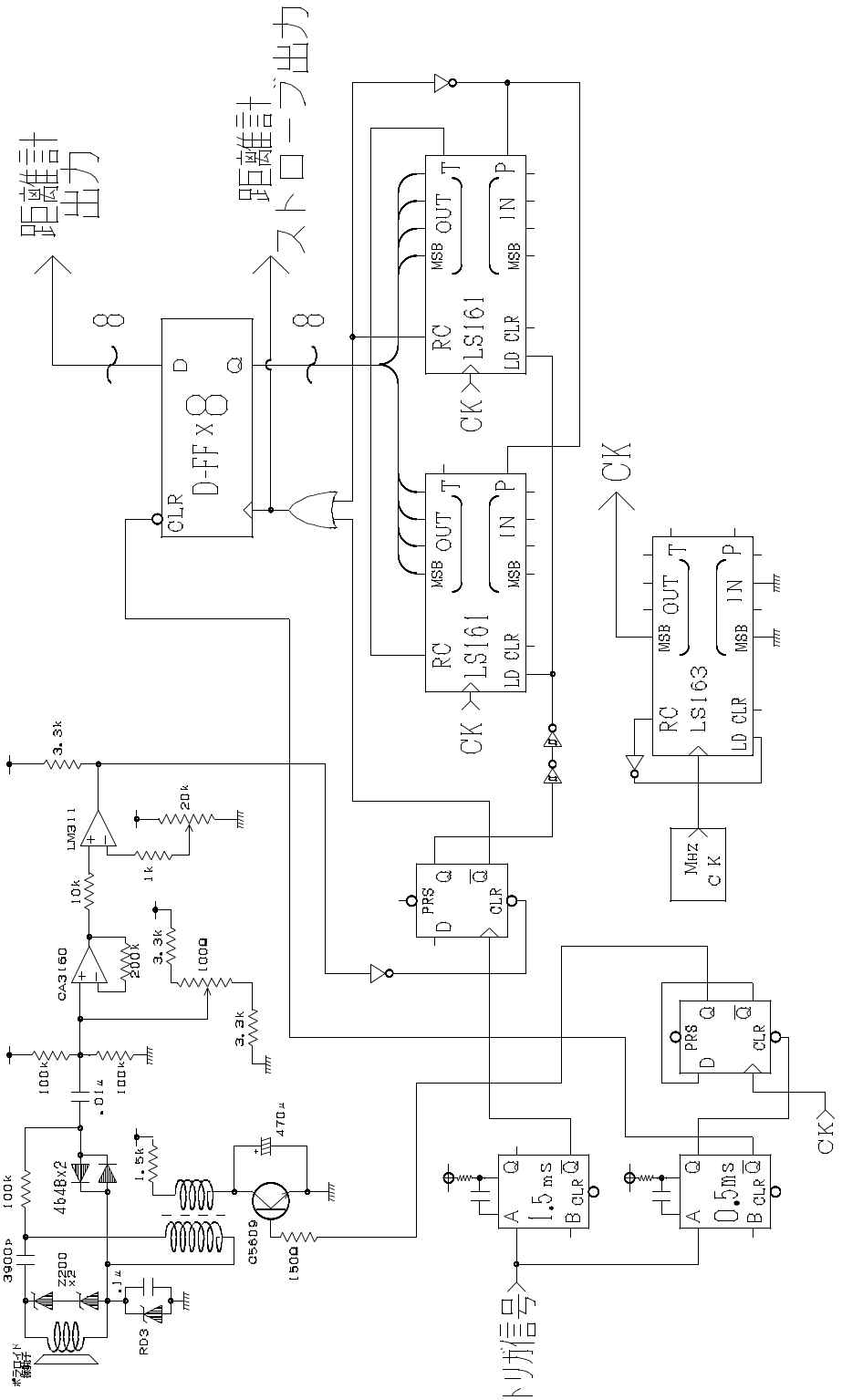

距離計出力
距離計データ出力
8
8
D
Q
D-FF X8
CLR
RC
LS161
LD CLR
MSB OUT T
MSB IN P
CK
RC
LS163
CK
MHz
CK
PRS
CLR
LM311
CA3160
3.3k
20k
1k
10k
200k
100Ω
100k
.01μ
470μ
1.5k
150Ω
Q2SC5609
4648x2
Z200
3900p
.1μ
RD3
A
B
1.5ms
0.5ms
トリガ信号
CK

図 C.18: 超音波距離系

C.2 Gray-scale Morphology 回路

ここに示す回路図は Chapter 3 に述べた Gray-scale Morphology 処理装置の試作機である．この回路は内部に持つクロックに同期して画像処理を行うものであり，ホスト側のソフトウエアと制御回路のロジックによる数十種類のコマンドを持っている．これにより多様な Gray-scale Morphology 処理を実現することができる．

この回路はパソコンから I/O を介して画像を転送し，一通りの画像処理を終えた後，再び元に戻すという方式をとっている．このため画像の転送を最小限に押さえることが可能で，I/O 性能の低い計算機でもスループットを落とさないで済む．

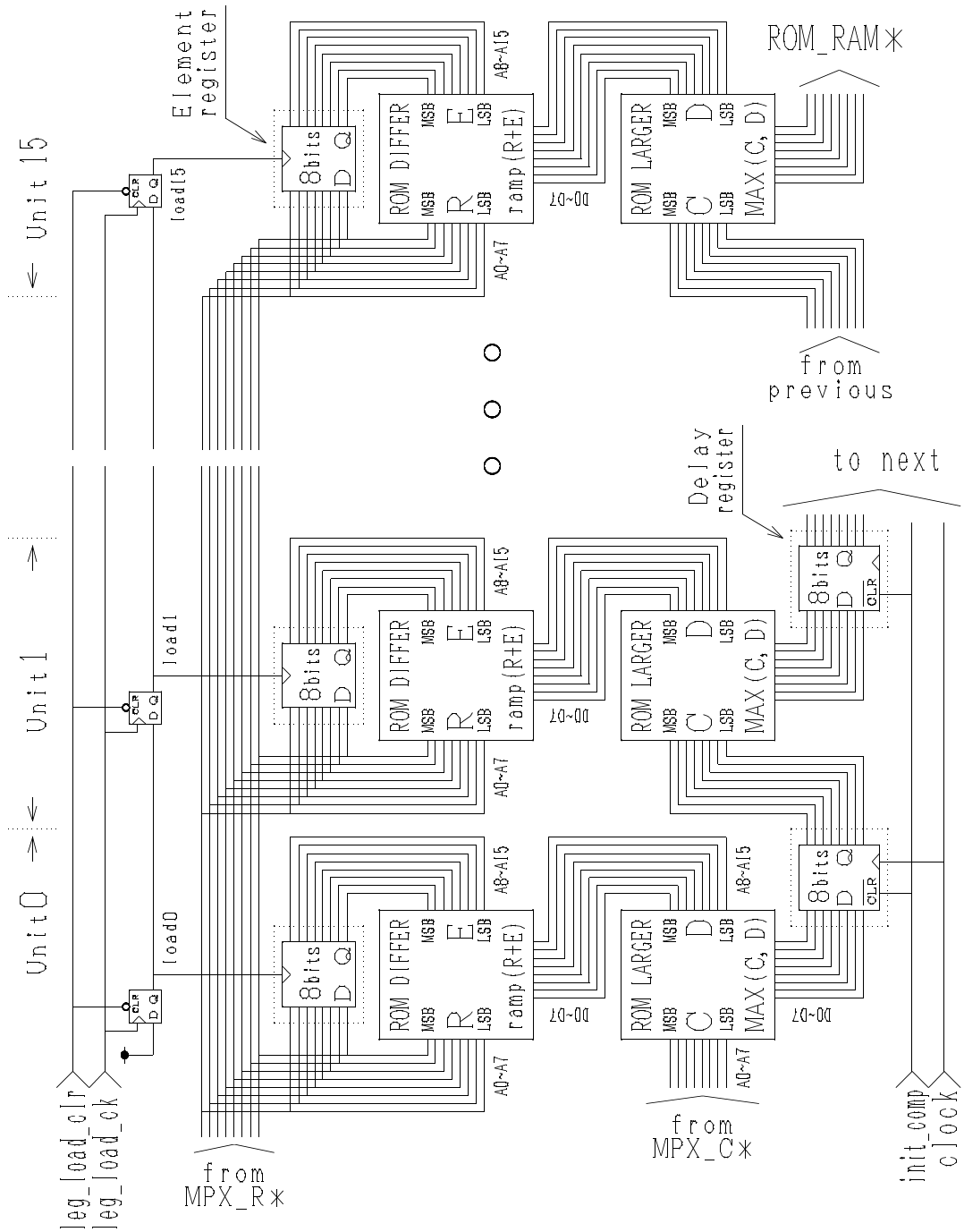


図 C.19: Gray-scale Morphology 回路

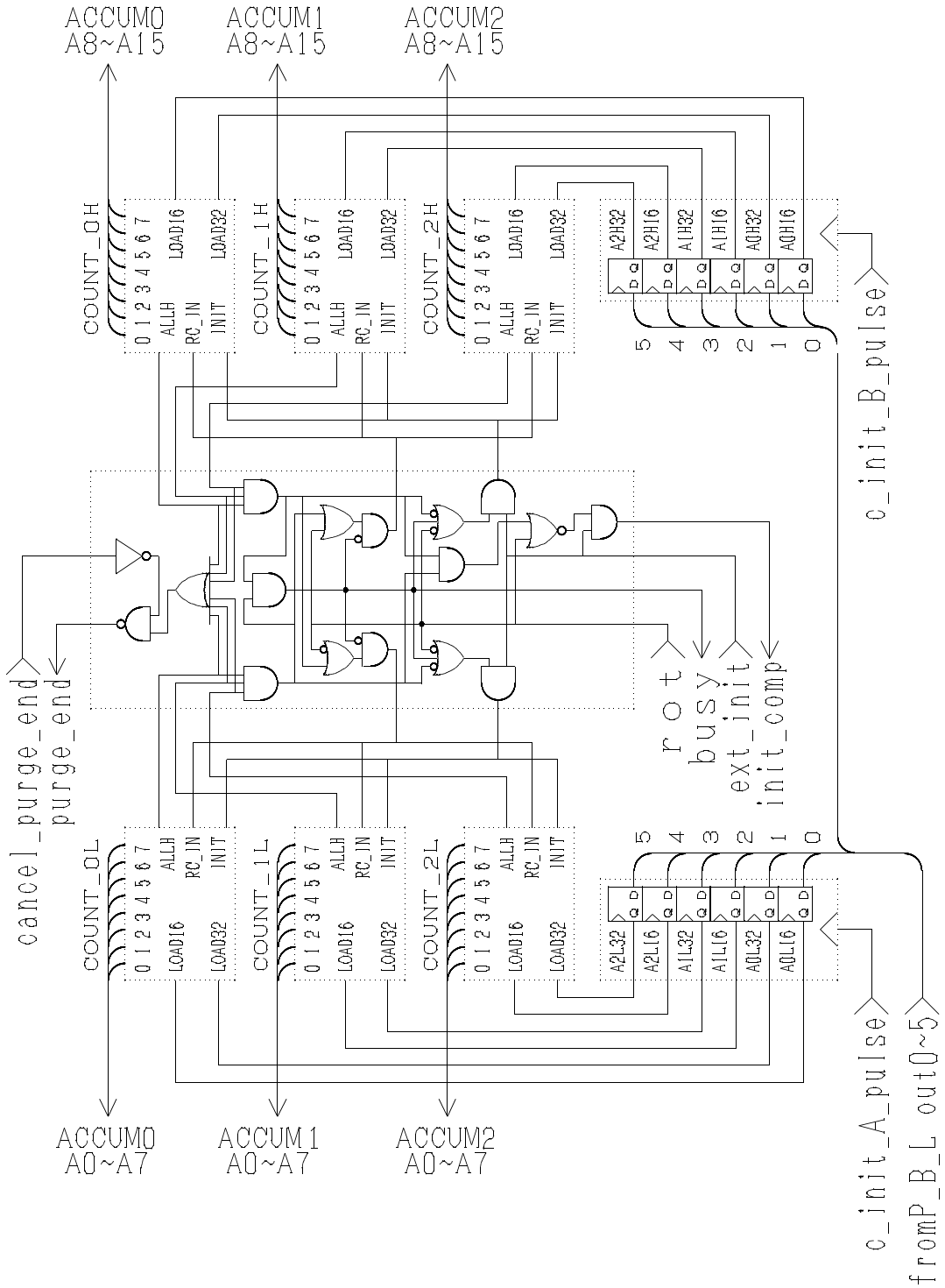
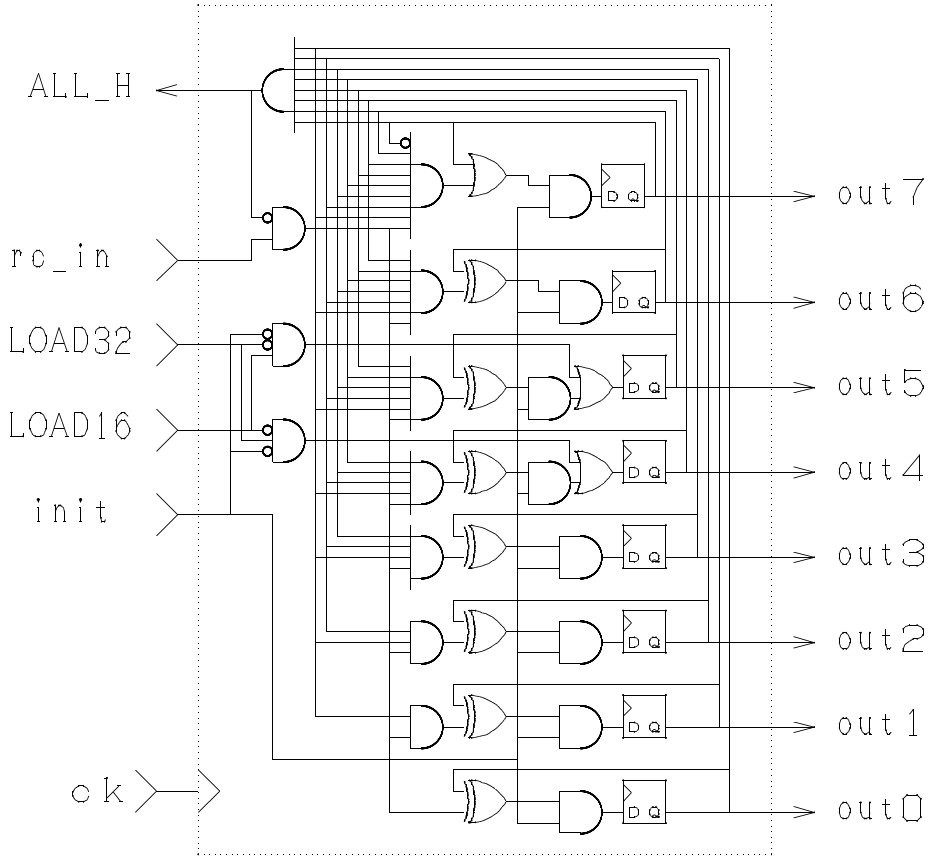


図 C.20: カウンタ制御ブロック



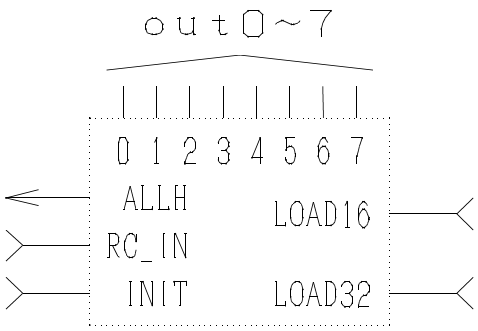


図 C.21: 8 bit カウンタ

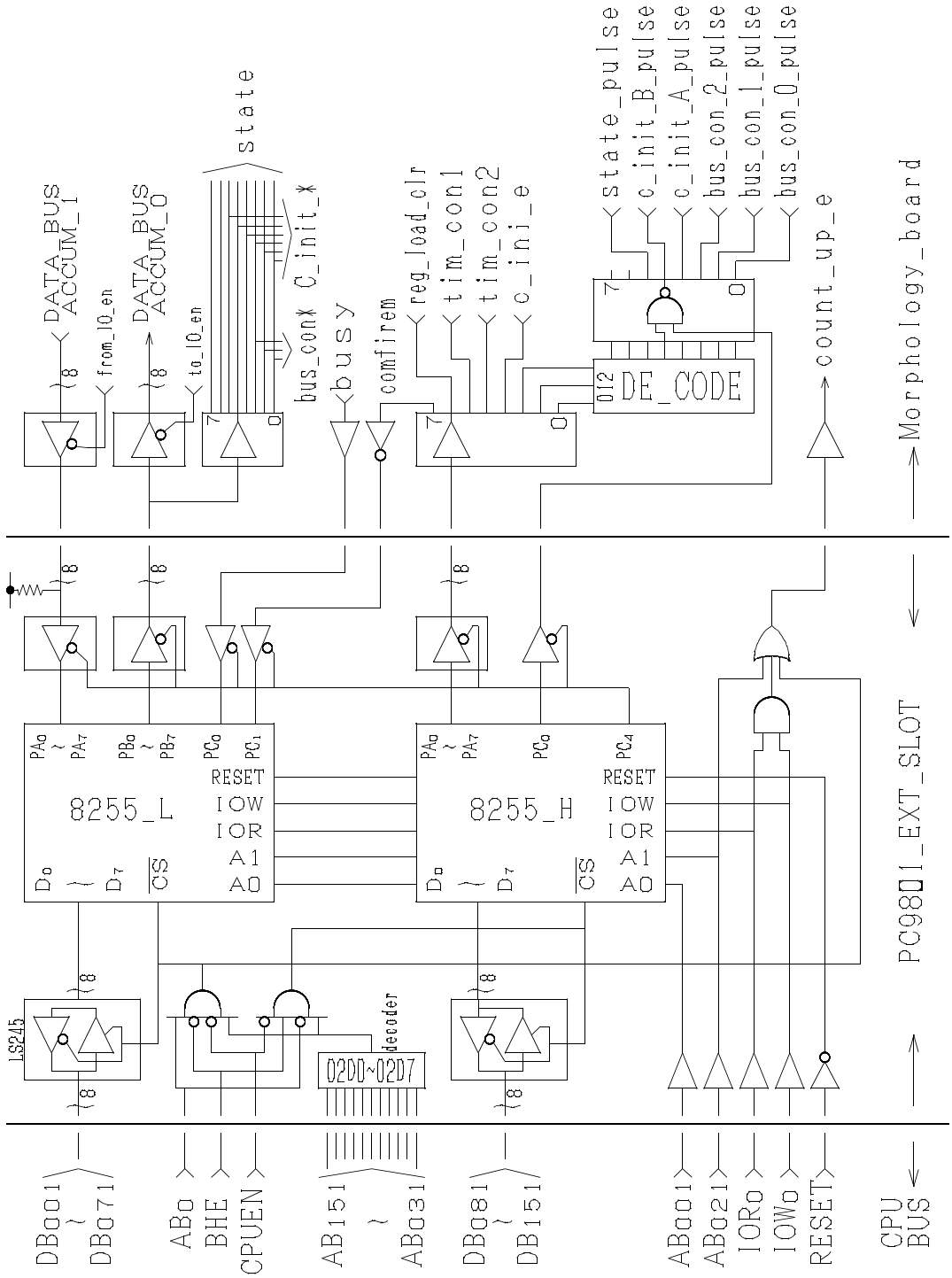


図 C.22: I/O(PC-9801 C bus)

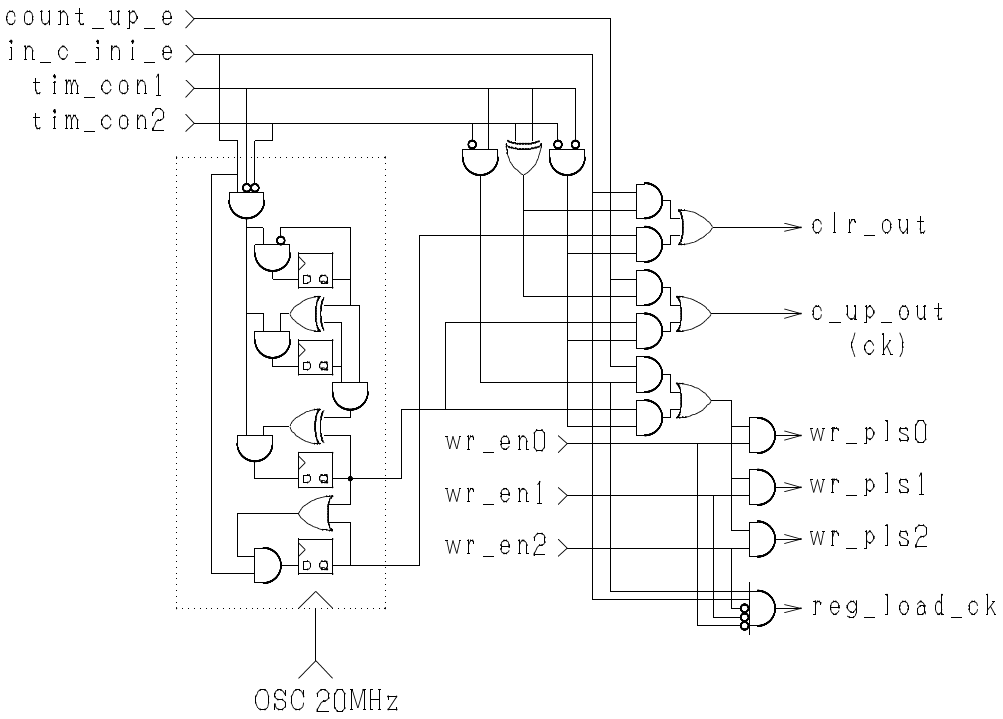


図 C.23: Timing 発生回路

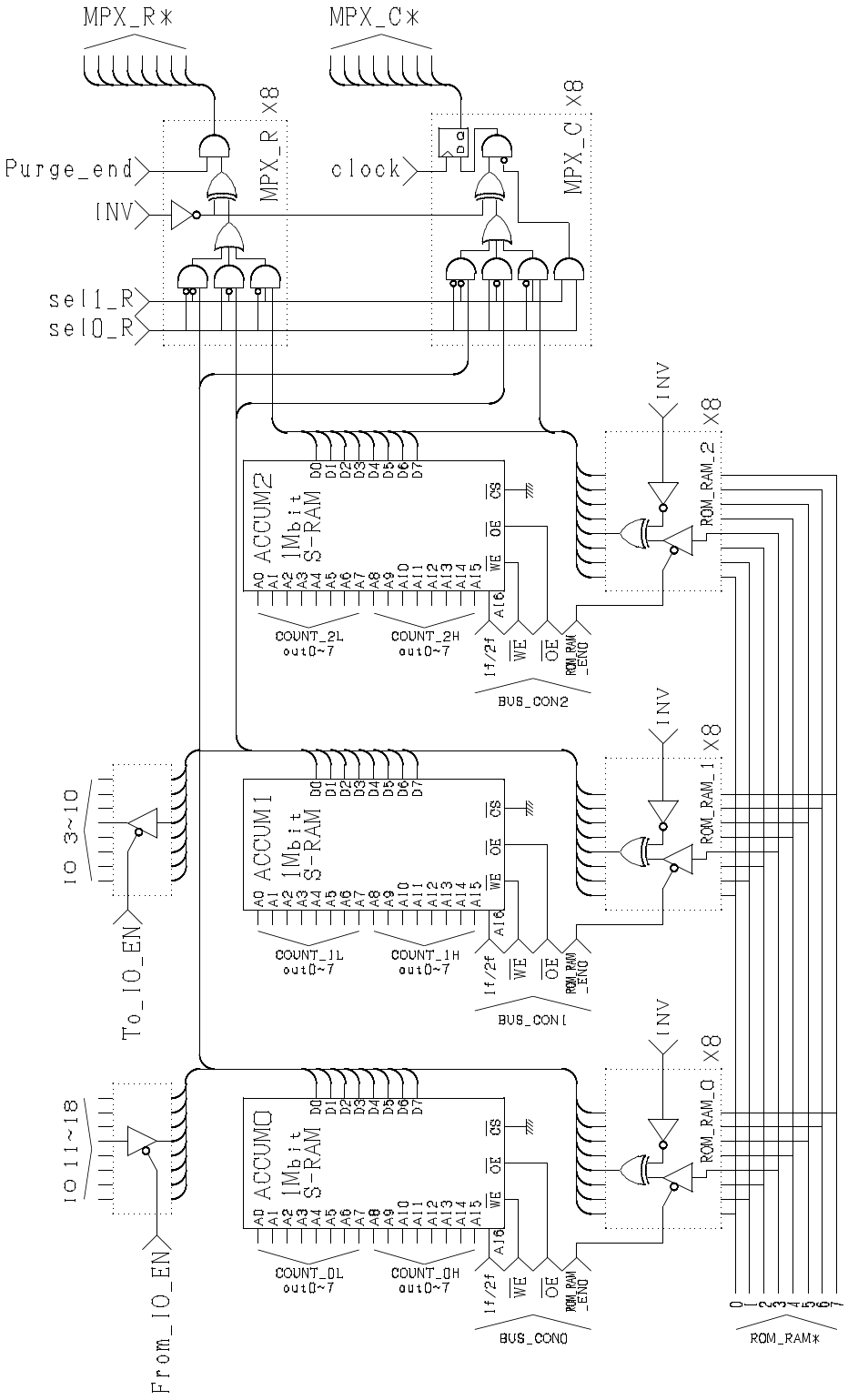


図 C.24: Data bus 結線図 (クロスバースイッチを含む)

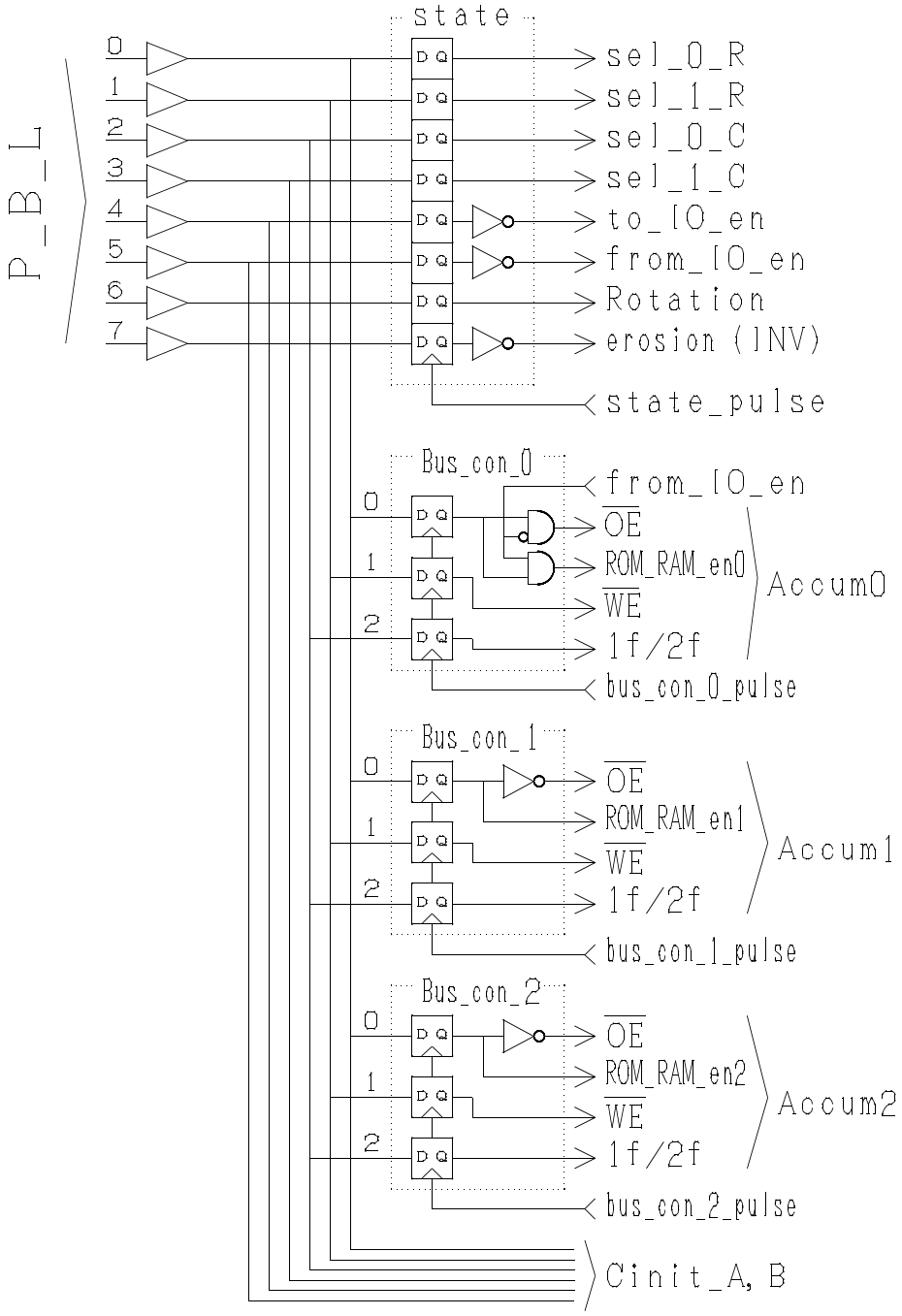


図 C.25: コントロール・バス

C.3 テーブル変換器 Morphology 回路

ここに示す回路図は Chapter 4 に述べた，テーブル変換機を用いた Gray-scale Morphology 処理装置の試作機である．テーブル変換機は 1Mbit の SRAM 2 個による．また画像情報を 16bit の Gray-scale の深さで扱うことにより，誤差のない距離変換への応用が可能となった．

この回路では画像メモリをホストの主記憶に置いている．そしてこの画像処理装置とホストとは 4MByte/sec 程度の転送容量をもつ BUS で接続している（CPU の I/O 転送命令ではなく memory to memory 転送命令を使うことにより倍以上の転送の高速化を実現した）

この回路はソフトウエア側から見れば，あたかも Morphology の数値演算プロセッサのように振る舞う．また試作機をつなげない状態ではハードウェアをエミュレートするライブラリを呼び出すことによって速度的には遅くなるもののソフトウエア的な互換性が保てるような作りになっている．

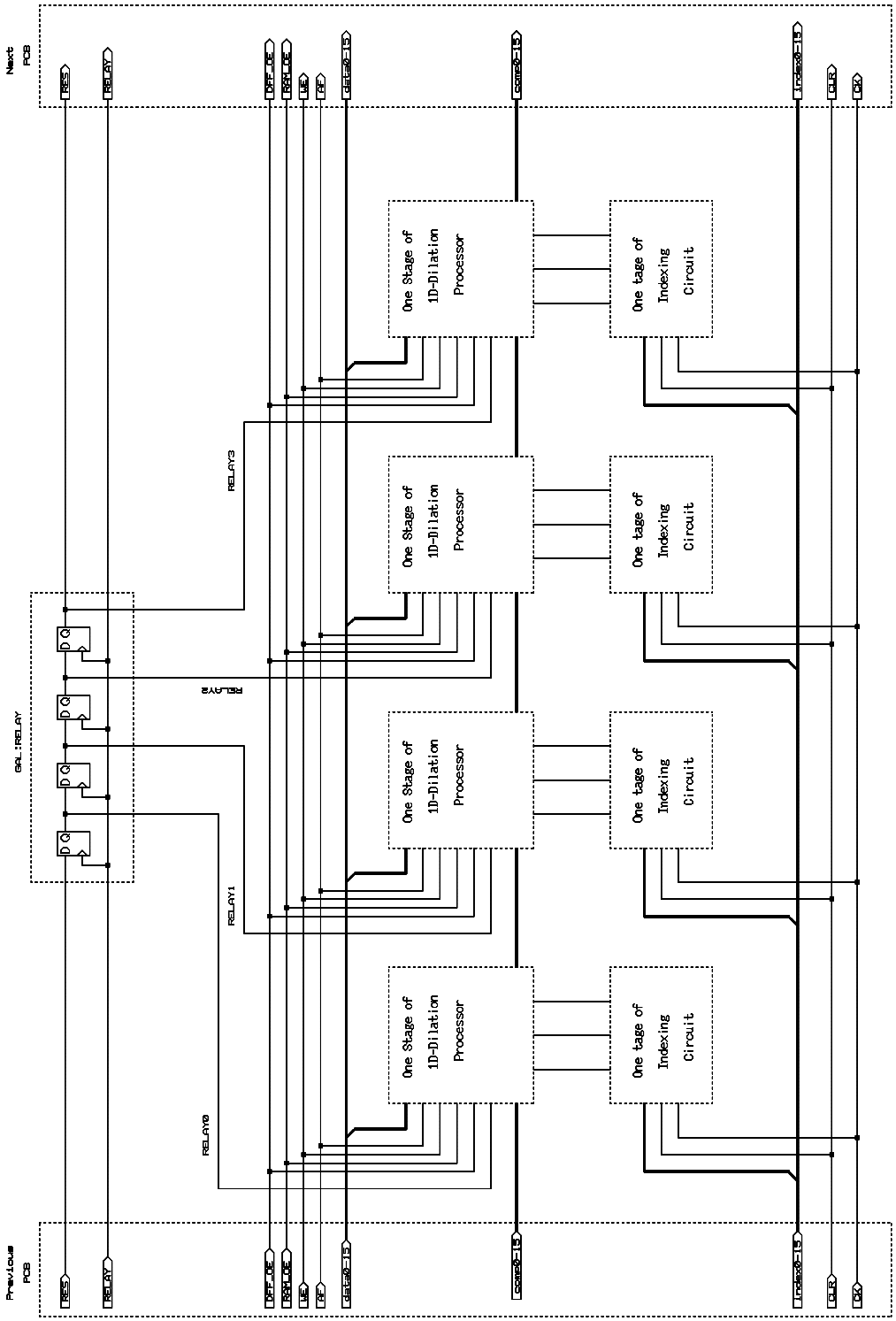


図 C.26: Dilation 基板の回路配置図

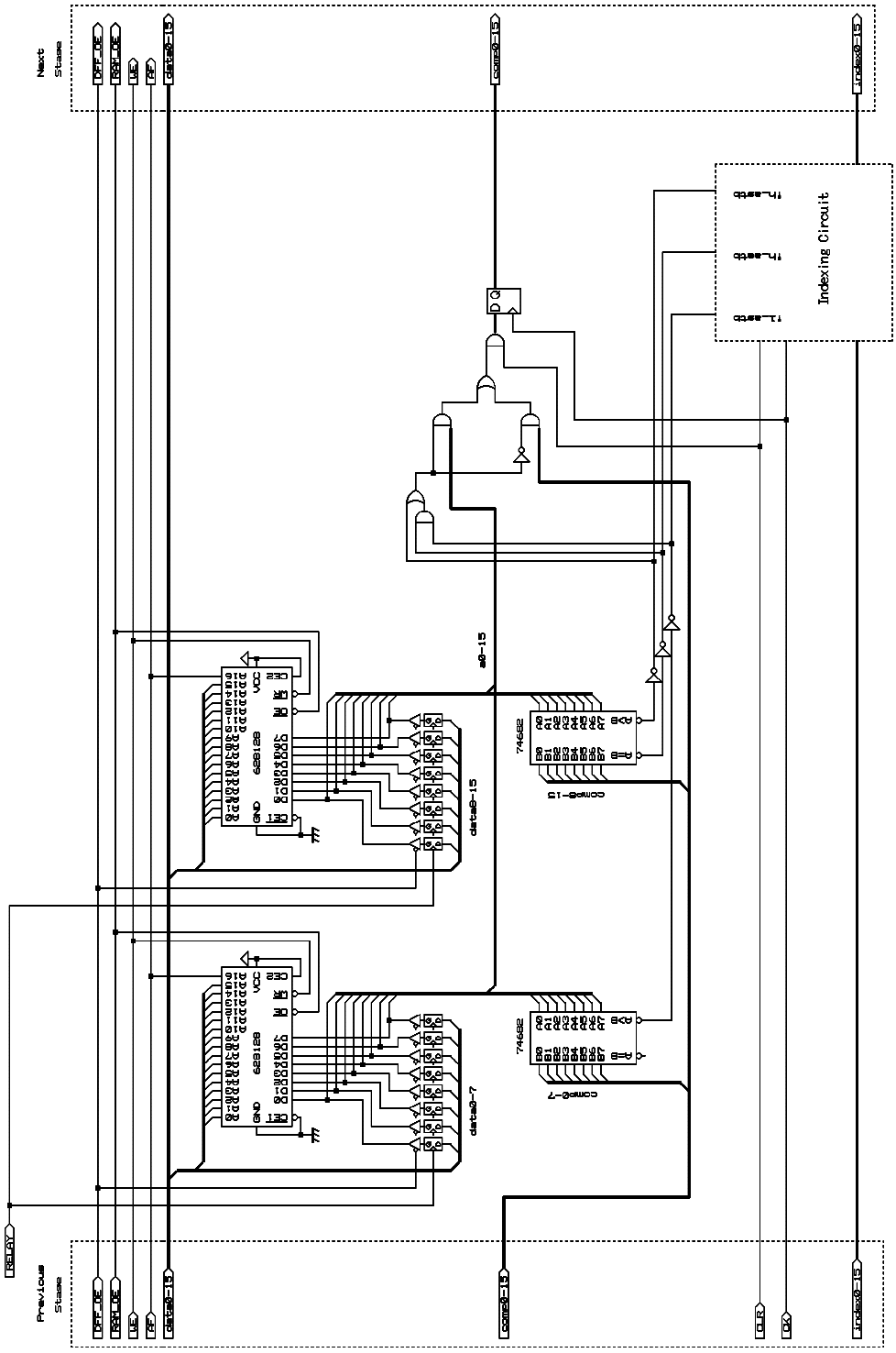


図 C.27: 1 次元 Dilation 回路の 1 stage

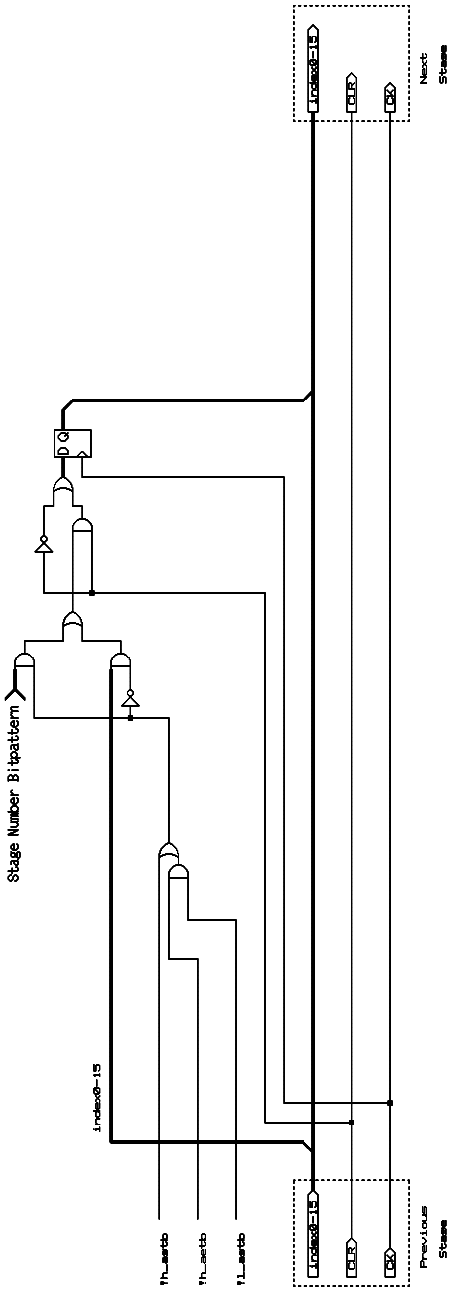


図 C.28: Indexing 回路の 1stage

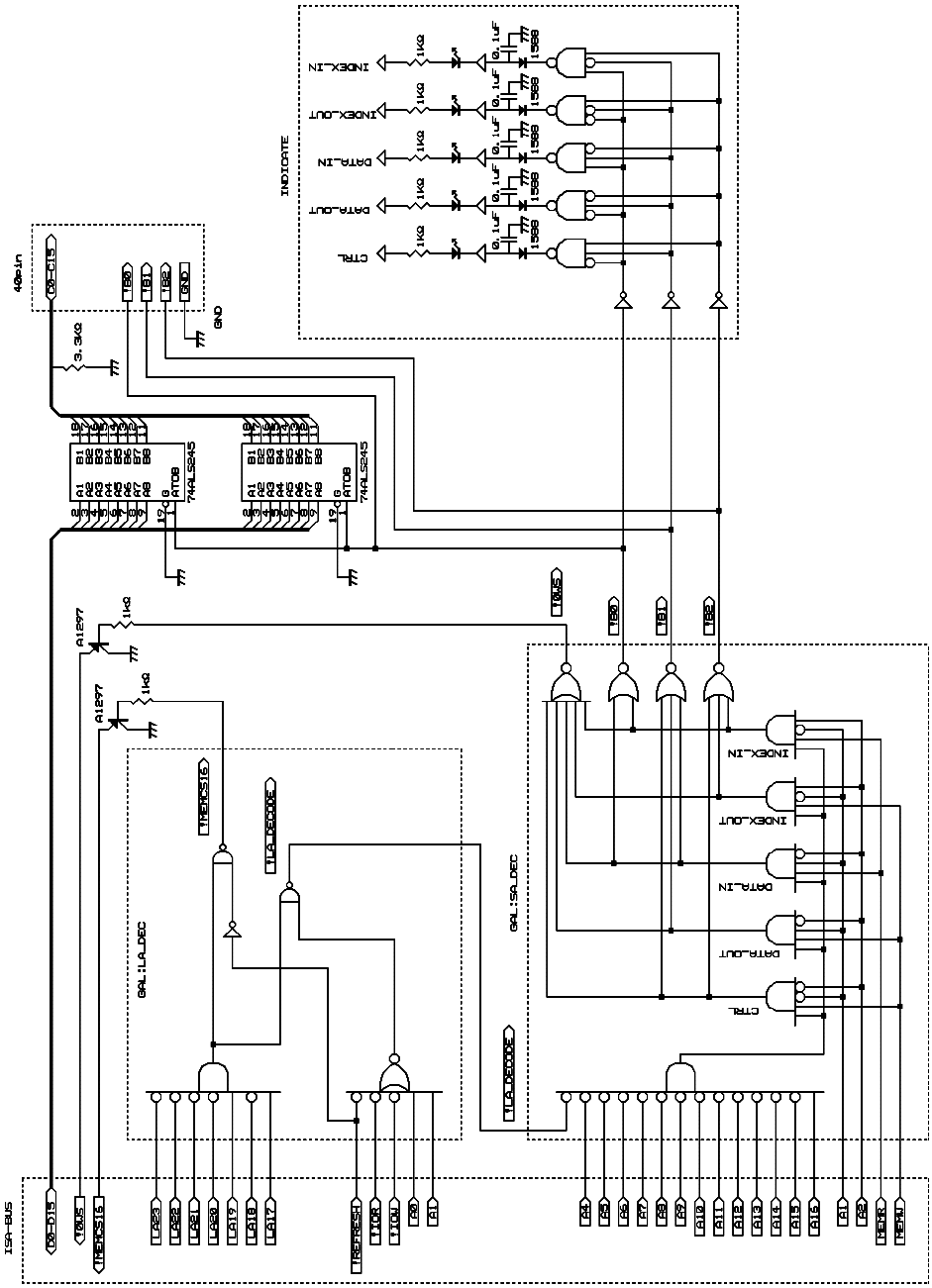


図 C.29: I/O(IBM-PC の AT bus)

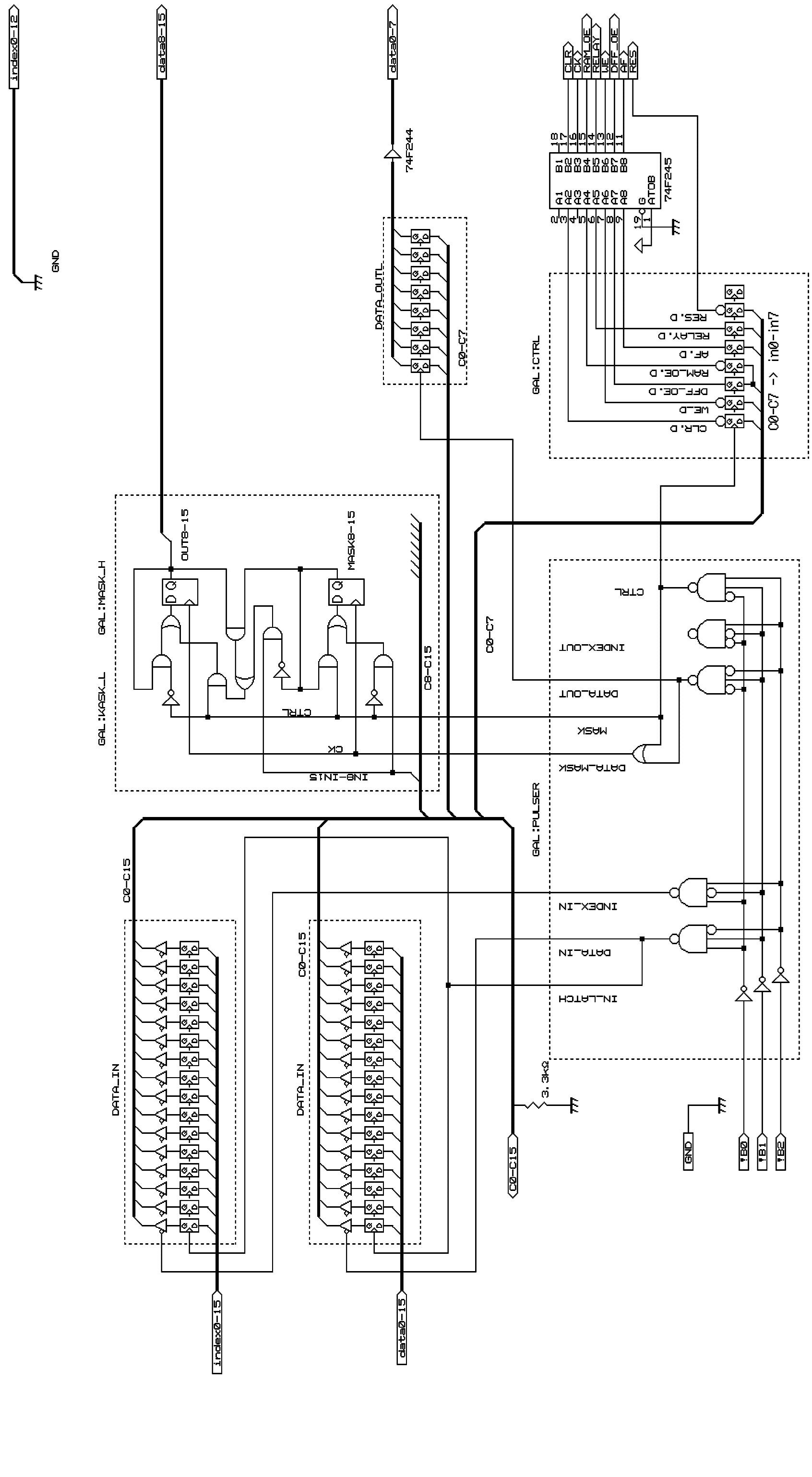

index0-12
data8-15
data0-7
GND
74F244
74F245
DATA_OUTL
C0-C7
GAL:CTRL
CLR.D
WE.D
OFF_OE.D
RAM_OE.D
AF.D
RELAY.D
RES.D
C0-C7 -> in0-in7
GAL:MASK_L
GAL:MASK_H
OUT8-15
MASK8-15
C8-C15
CTRL
CK
IN8-IN15
GAL:PULSER
INDEX_OUT
DATA_OUT
MASK
DATA_MASK
INDEX_IN
DATA_IN
IN_LATCH
C0-C15
index0-15
data0-15
TB0
TB1
TB2
3.3kΩ

図 C.30: コントローラ