Chapter 1

はじめに

1.1 本研究の背景

近年の画像処理システムにおいては、画像の高解像度化や扱うフレーム数の増加に伴い扱う情報量が著しく増加している。従って画像の処理に莫大な時間がかかるためリアルタイムでの画像処理が困難なものとなってきている。

また、半導体技術の発展により、処理系のデータ処理能力は飛躍的に向上してきた。しかし近年ムーアの法則に沿った発展にはかげりが見えはじめ、いずれは集積度やクロック速度の発展は飽和状態となることが予想されている。そのため処理時間のさらなる短縮にはこれまでの方法とは異なる革新的な方法をとらなければならない。

この問題を解決する方法として、集積回路を用いてノイズ除去やエッジ検出といった通常の画像処理のうちでの前処理部分の処理をハードウエアで行うことにより、処理形式を従来のようにソフトウェアを用いたシーケンシャルな処理ではなく並列的な処理を行うことによによって高速化することができる。

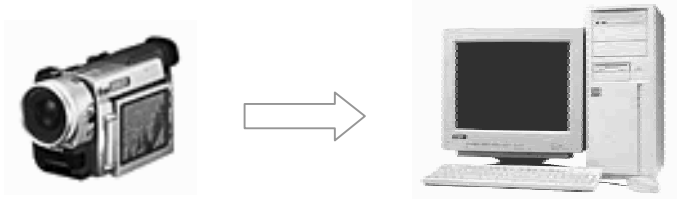


図1.1　従来の画像処理システム

1.1 本研究の目的

本研究では、画像処理の一種である特定形状の物体の位置や大まかな大きさの検出を、従来のようにマトリクス状の数値列に対して逐次処理をすることによって行うのではなく、比較的単純な機能を有するオートマトンを平面上に規則的に配置した構造により、画素の空間的二次元相関を考慮しつつ、並列的に処理をすることによって特定形状を持つ物体を検出し、またその物体の大まかな位置や大きさなどを知ることができるアルゴリズムを提案する。また、その回路構成についても検討する。

本研究においては特定の形状を持つ物体の例として長方形を用いて検出アルゴリズムを検討していくが、最終的には長方形の定義を拡張し、ある幅以下で少しの幅のばらつきを含む線分を検出することができるアルゴリズムを考え、また線分を検出する回路の回路構成についても検討し、実際の回路のレイアウトを設計することを目的とする。

Chapter 2

特定形状を持つ物体の検出アルゴリズムの検討

この章においては、セルオートマトンを用いて長方形及び線分を検出するための基本的なアルゴリズムを検討しその正当性を示すが、まず本研究におけるもっとも根本的な原理であるセルオートマトンについて説明する。

2.1 セルオートマトン

図2.1のように平面上にマトリクス状に配置された画素の間にセルオートマトンを配置した構造を考え、各セルオートマトンは隣接する４つの画素または近傍のセルオートマトンからの情報を受け取り、クロックに同期して遷移するものとする。また本研究においての画像は二値画像であるとし、各セルオートマトンには図2.2のように隣接する４つの画素の他に、それぞれ上下左右２つ隣りまでのセルオートマトンの情報を入力するものとする。以下、簡単のため単位セルオートマトンをセルと略す。

このセルオートマトンを用いる利点として、各セルの遷移方法は近傍のセルの状態のみから決まり、局所的であるため画素数が増加したとしても各セルの持つ機能は変化しないことが挙げられる。つまり各セルは単独での存在が可能であり、画素数が増えるにつれてセルの数を増やすだけで良く、機能の拡張が比較的容易である。

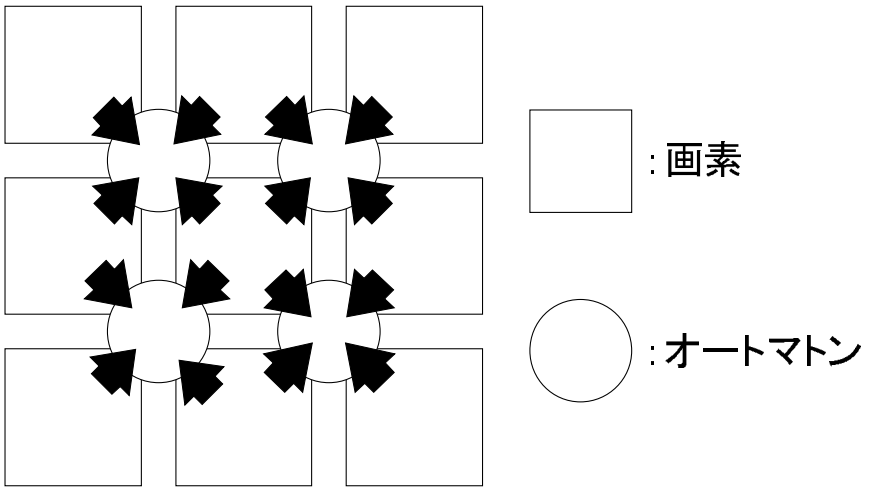


図2.1　画素とセルオートマトンの配置

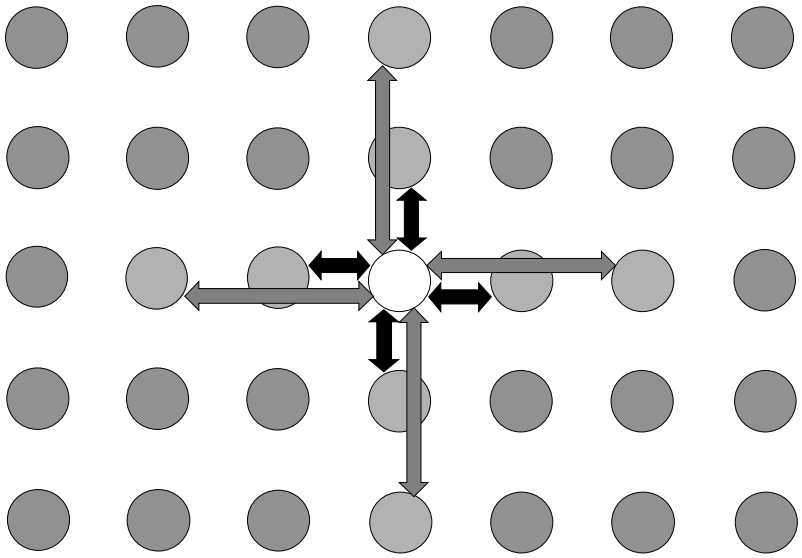


図2.2　各セルへ入力する近傍のセルの情報

2.2　長方形の検出法

はじめにセルオートマトンを用いて検出する物体の例として長方形を考え、長方形を検出することができるアルゴリズムについて考える。

長方形を検出することができるアルゴリズムとしては、はじめに縦方向へ細線化していき、縦方向における最小の状態が出現した時点で全体の遷移関数を変更し、つぎに横方向へ収縮し、横方向においても最小の状態が出現したら残りのセルの状態より、入力された画像に含まれる物体が長方形であるか否かを判定し、長方形であるならばフラグをたてることによって検出する方法が考えられる。

この方法におけるアルゴリズムを以下に示す。ただし、$d_{v,h}$は垂直方向v番目、水平方向h番目の画素を表し、$s_{i,v,h}$はi段目の垂直方向v番目、水平方向h番目のセルの状態を表す。また、アルゴリズムを図式化したものを図2.3に示す。

1.まず、$i = 1$で各セルは隣接する４つの画素の論理積をとる。

$$S_{1,v,h} = d_{v,h} \cdot d_{v,h+1} \cdot d_{v+1,h} \cdot d_{v+1,h+1} \tag{2,1}$$

２．次に$i = 2$より縦方向に長方形を細線化していき、縦方向における最小の状態が出現するまで収縮させる。

$$\begin{aligned} S_{i+1,v.h} = {} & \overline{S_{i,v-1,h}} \cdot S_{i,v,h} \cdot \overline{S_{i,v+1,h}} \\ & + \overline{S_{i,v-2,h}} \cdot S_{i,v-1,h} \cdot S_{i,v,h} \cdot \overline{S_{i,v+1,h}} \\ & + \overline{S_{i,v-1,h}} \cdot S_{i,v,h} \cdot S_{i,v+1,h} \cdot \overline{S_{i,v+2,h}} \end{aligned} \tag{2,2}$$

３．縦方向における最小の状態が出現した時点で全体の遷移関数を変更し、次に横方向に収縮させていく。

$$\begin{aligned} S_{i+1,v,h} = {} & \overline{S_{i,v,h-1}} \cdot S_{i,v,h} \cdot \overline{S_{i,v,h+1}} \\ & + \overline{S_{i,v,h-2}} \cdot S_{i,v,h-1} \cdot S_{i,v,h} \cdot \overline{S_{i,v,h+1}} \\ & + \overline{S_{i,v,h-1}} \cdot S_{i,v,h} \cdot S_{i,v,h+1} \cdot \overline{S_{i,v,h+2}} \end{aligned} \tag{2,3}$$

４．縦と横双方向に対して最小の状態が出現した時点で残りのセルの状態より入力された物体が長方形であるか否かを判定し、長方形であるならば全てのセルを”0”とし、フラグをたてることにより検出する。

またここで、長方形の検出に必要なステップ数について考えてみると、画素数が増加したとしても、検出に必要なステップ数は長方形の面積、さらに言うならば長方形の辺の長さに依存する。つまり、画素が“１”である領域が $m \times n$ である長方形が取り込まれた場合にも検出にかかるステップ数は約 $1/2(m+n)$ であり、n の１乗に比例するため時間計算量は $O(n)$ となる。また、各セルが持つ機能も比較的単純であるために、回路で実現した場合にも高速動作が期待できる。

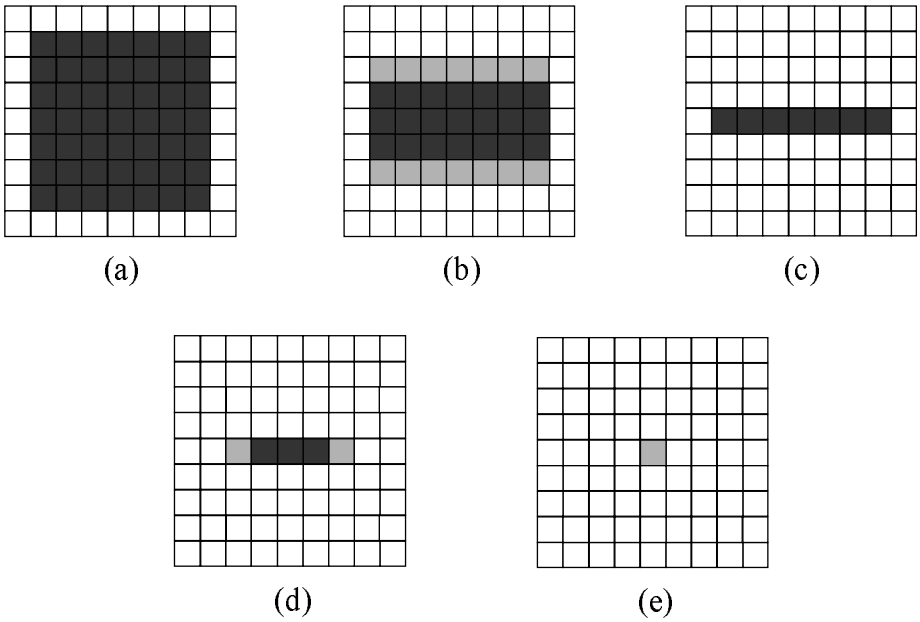


図 2.3　長方形の検出方法

2.3　線分の検出法

2.2 では長方形の検出方法について述べてきたが、この方法では正確でかつ凹凸がない長方形の場合にのみしか検出することができず、応用範囲は狭いといえる。従ってある一定の幅（線分の太さ）以下で、ある程度の幅のばらつきを含んでいる線分も検出できるようにすると応用範囲は広がると考えられる。よって以下にそのような線分を検出することができるアルゴリズムについて示す。

１．まず、線分表面の幅のばらつきを容認することと、単発的な画像のノイズを無視することを兼ねて、線分の凹凸等を無視して表面をなめらかにする。このステップを図式化したものを図 2.4 に示す。

$$\begin{aligned} S_{i+1,v.h} = {} & \overline{S_{i,v,h-1}} \cdot S_{i,v,h} \cdot \overline{S_{i,v,h+1}} \\ & + \overline{S_{i,v-1,h}} \cdot S_{i,v,h} \cdot \overline{S_{i,v+1,h}} \\ & + S_{i,v,h-1} \cdot \overline{S_{i.v.h}} \cdot S_{i,v,h+1} \\ & + S_{i.v-1.h} \cdot \overline{S_{i,v,h}} \cdot S_{i,v+1,h} \end{aligned} \qquad (2.4)$$

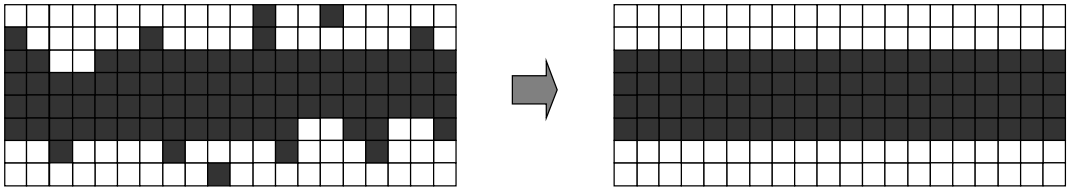


図 2.4　単発的画像ノイズや単発的凹凸の除去

２．次に、線分の幅方向に一定時間だけ線分を細線化する。このときの細線化の時間、つまりステップ数によって検出することができる線分の幅が決まってくる。だが、ただ線分をただ収縮していったのでは、線分の芯になる部分が列によって異なってくるために、一本の細線化された線分になるためには、線分が一本のまっすぐな芯を中心として持ち、かつ芯に対して対称である線分である必要がある。よってこの問題を解決するために線分の芯となることのできる横の列を交互に配置した（図 2.5、図 2.6）。だがこのままでは、線分の芯となることのできないとした横の列に線分の中心がある場合に問題が発生する。よって収束することができる横の列を変更することが必要である（図 2.7）。

収束できる列でのアルゴリズムとして。

$$
\begin{aligned}
S_{i+1,v,h} = {} & \overline{\overline{S_{i,v-1.h}} \cdot S_{i,v,h} \cdot S_{i,v+1,h} \cdot S_{i,v+2,h}} \\
& + \overline{S_{i,v-2,h} \cdot S_{i,v-1,h} \cdot S_{i,v,h} \cdot \overline{S_{i,v+1,h}}} \\
& + \overline{S_{i,v-1}} \cdot S_{i,v,h} \cdot \overline{S_{i,v+1,h}} \\
& + \overline{S_{i,v-2,h}} \cdot S_{i,v-1.h} \cdot S_{i,v,h} \cdot \overline{S_{i,v,h+1}} \\
& + \overline{S_{i,v-1,h}} \cdot S_{i,v,h} \cdot S_{i,v+1,h} \cdot \overline{S_{i,v+2,h}}
\end{aligned}
\tag{2.5}
$$

一方、収束できない列でのアルゴリズムは、

$$
\begin{aligned}
S_{i+1,v,h} = {} & \overline{\overline{S_{i,v-1.h}} \cdot S_{i,v,h} \cdot S_{i,v+1,h} \cdot S_{i,v+2,h}} \\
& + \overline{S_{i,v-2,h} \cdot S_{i,v-1,h} \cdot S_{i,v,h} \cdot \overline{S_{i,v+1,h}}} \\
& + \overline{S_{i,v-1}} \cdot S_{i,v,h} \cdot \overline{S_{i,v+1,h}} \\
& + \overline{\overline{S_{i,v-2,h}} \cdot S_{i,v-1.h} \cdot S_{i,v,h} \cdot \overline{S_{i,v,h+1}}} \\
& + + \overline{\overline{S_{i,v-1,h}} \cdot S_{i,v,h} \cdot S_{i,v+1,h} \cdot \overline{S_{i,v+2,h}}}
\end{aligned}
\tag{2.6}
$$

となる。

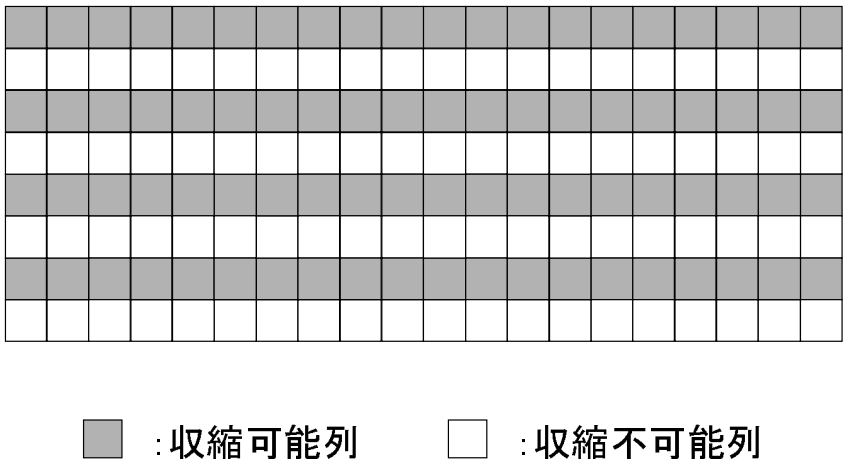


図 2.5　対称ではない線分の検出(1)

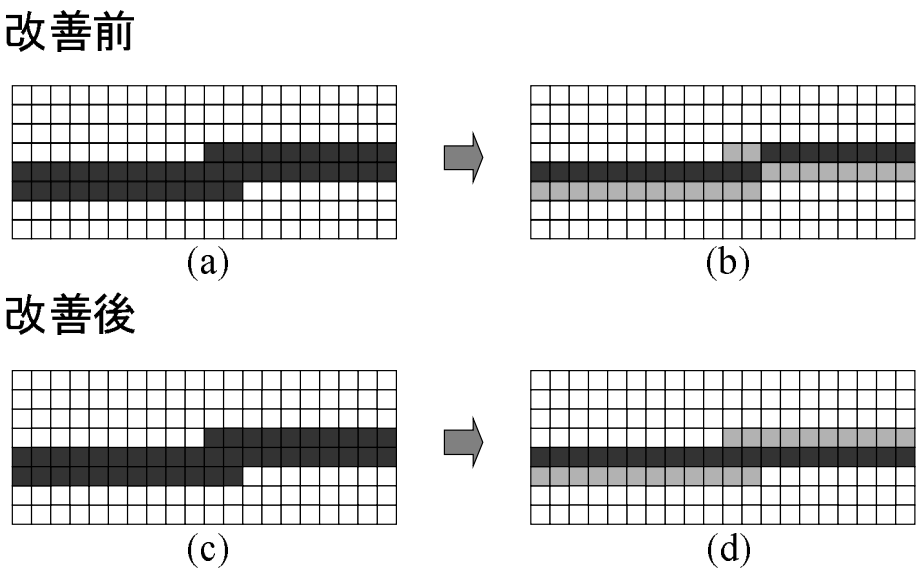


図 2.6　対称ではない線分の検出(2)

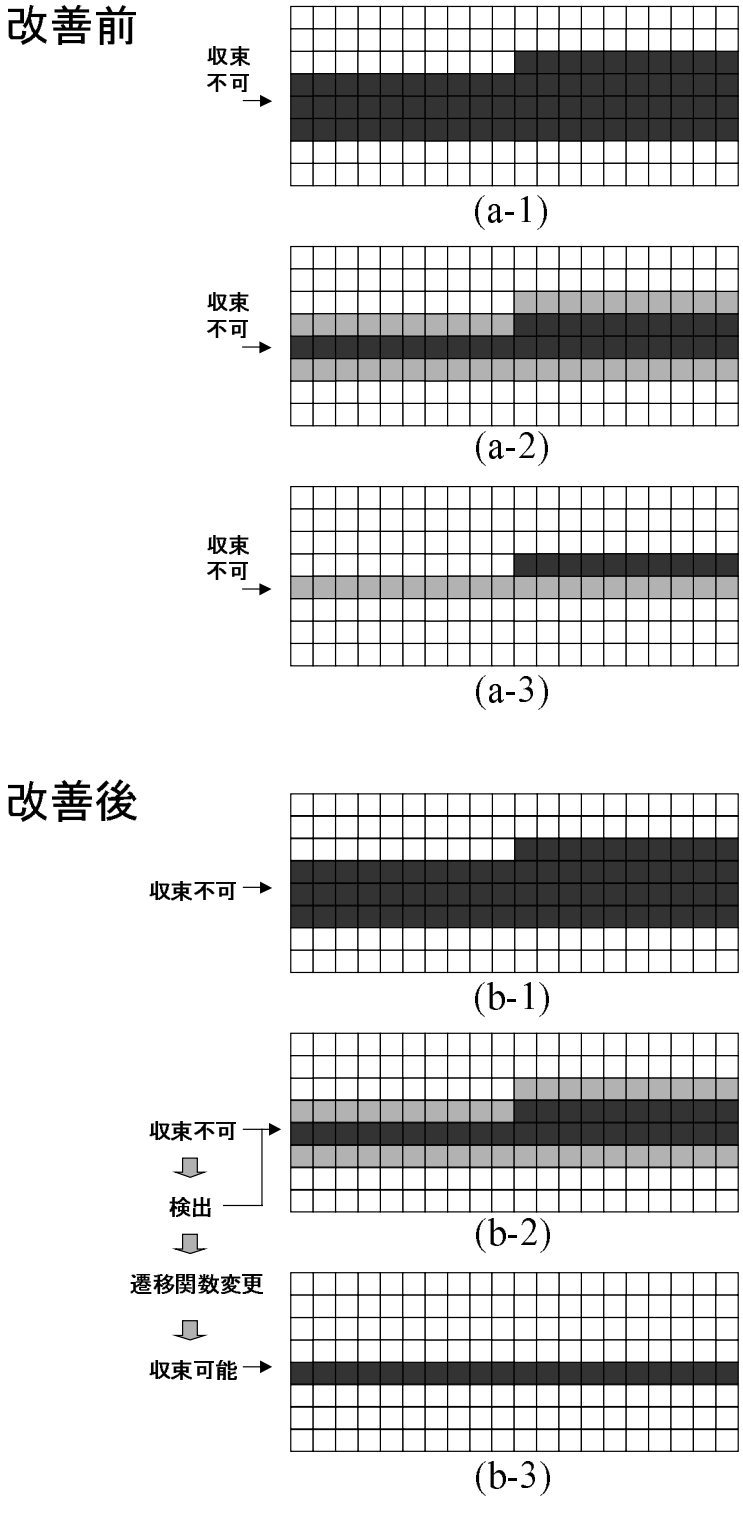


図 2.7 収束可能列の変更

３．線分の幅方向へ細線化された後、次に線分の長さ方向へ単位セルの大きさになるまで収縮していく。このステップを図式化したものを図 2.8 に示す。

$$\begin{aligned} S_{i+1,v,h} = {} & \overline{S_{i,v,h-1}} \cdot S_{i,v,h} \cdot S_{i,v,h+1} \cdot S_{i,v,h+2} \\ & + S_{i,v,h-2} \cdot S_{i,v,h-1} \cdot S_{i,v,h} \cdot \overline{S_{i,v,h-1}} \\ & + \overline{S_{i,v,h-2}} \cdot S_{i,v,h-1} \cdot S_{i,v,h} \cdot \overline{S_{i,v,h+1}} \\ & + \overline{\overline{S_{i,v,h-1}} \cdot S_{i,v,h} \cdot S_{i,v,h+1} \cdot \overline{S_{i,v,h+2}}} \end{aligned} \tag{2.7}$$

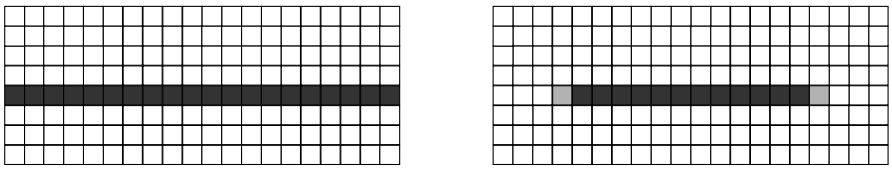


図 2.8　線分の長さ方向への収縮

４．最後に、残りのセルの状態より入力された物体が一定の幅以下の線分であったか否かを判定し、線分であった場合にはフラグをたてて検出する。

またここで、線分の検出に必要なステップ数について考えてみると、画素数が増加したとしても、検出に必要なステップ数は線分が十分に細いとした場合にはほぼ線分の長さに比例する。つまり、長方形の場合と同様に、n の１乗に比例するため時間計算量は $O(n)$ となる。

2.4　他の角度への応用

これまでに述べてきた長方形検出のアルゴリズムにおいては長方形が画素に対して水平垂直に取り込まれた場合にのみしか検出できない、また、線分検出回路においても線分の中心が１画素分だけずれる程度には検出できるが、それ以上の場合には検出できない。よって、今後を考えた場合には他の角度への対応を考えなければならない。

そこで、画像が画素に対して45度の傾きを持って取り込まれた場合について考えてみると、長方形検出アルゴリズムおよび線分検出アルゴリズムの両方に対して、遷移方向が 2.2 及び 2.3 で述べてきた縦と横の方向より、それぞれ 45 度ずれることが予想される。そこで、2.1 において各セルへ上下左右2個隣のセルまでの情報となっている入力も、45 度傾けた角度からとする必要がある。つまり図 2.9 に示すような入力とすればよい。

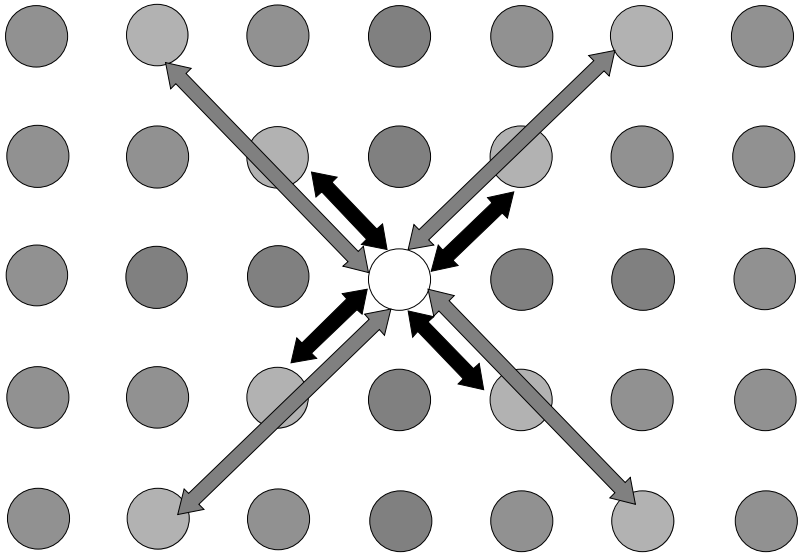


図 2.9　各セルへ入力する近傍のセルの情報（４５度）

次に、45 度の角度における遷移関数について、長方形検出アルゴリズムを例にとって考えてみると、画素とセルオートマトンの配置そのものは変化していないため、$d_{v,h}$ は垂直方向 v 番目、水平方向 h 番目の画素を表し、$S_{i,v,h}$ は i 段目の垂直方向 v 番目、水平方向 h 番目のセルの状態を表すとする。

以下において $v=0, h=0$ から v, h の値が共に増加していく斜めの方向を“正”の方向であるとし、それと垂直に交わる方向を“負”の方向であると定義する。

1. まず、$i=1$ で各セルは隣接する４つの画素の論理積をとる。

$$S_{1,v,h} = d_{v,h} \cdot d_{v,h+1} \cdot d_{v+1,h} \cdot d_{v+1,h+1} \tag{2,8}$$

２．次に $i=2$ より正方向に長方形を細線化していき、正方向における最小の状態が出現するまで収縮させる。

$$\begin{aligned} S_{i+1,v.h} = {} & \overline{S_{i,v-1,h-1}} \cdot S_{i,v,h} \cdot \overline{S_{i,v+1,h+1}} \\ & + \overline{S_{i,v-2,h-2}} \cdot S_{i,v-1,h-1} \cdot S_{i,v,h} \cdot \overline{S_{i,v+1,h+1}} \\ & + \overline{S_{i,v-1,h-1}} \cdot S_{i,v,h} \cdot S_{i,v+1,h+1} \cdot \overline{S_{i,v+2,h+2}} \end{aligned} \tag{2,9}$$

３．正方向における最小の状態が出現した時点で全体の遷移関数を変更し、次に負方向に収縮させていく。

$$\begin{aligned} S_{i+1,v,h} = {} & \overline{S_{i,v+1,h-1}} \cdot S_{i,v,h} \cdot \overline{S_{i,v-1,h+1}} \\ & + \overline{S_{i,v+2,h-2}} \cdot S_{i,v+1,h-1} \cdot S_{i,v,h} \cdot \overline{S_{i,v-1,h+1}} \\ & + \overline{S_{i,v+1,h-1}} \cdot S_{i,v,h} \cdot S_{i,v-1,h+1} \cdot \overline{S_{i,v-2,h+2}} \end{aligned} \tag{2,10}$$

４．双方向に対して最小の状態が出現した時点で残りのセルの状態より入力された物体が長方形であるか否かを判定し、長方形であるならば全てのセルを”0”とし、フラグをたてることにより検出する。

これらアルゴリズムの式よりわかることは、水平垂直方向での検出アルゴリズムにおける遷移関数と遷移関数そのものは変化せずに、入力のみが変化したものとなっている。つまり、2.2 及び 2.3 において述べたアルゴリズムにおいての入力を

$$
\begin{aligned}
S_{i,v-n,h} &\rightarrow S_{i,v-n,h-n} \\
S_{i,v,h-n} &\rightarrow S_{i,v+n,h-n} \\
S_{i,v+n,h} &\rightarrow S_{i,v+n,h+n} \\
S_{i,v,h+n} &\rightarrow S_{i,v-n,h+n}
\end{aligned}
\tag{2.11}
$$

と置き換えることによって 45 度の角度で物体が取り込まれた場合のアルゴリズムを求めることができる。

以上のことより、画素に対して水平垂直な角度以外で取り込まれた場合にも検出可能とするには、遷移関数そのものの変化は必要なく、入力情報のみを変化することで対応できることがわかる。

Chapter 3

検出回路の設計

本研究のアルゴリズムはディジタル回路にて実現可能なためVerilog-HDLによって動作を記述し、Synopsys社のDesign Analyzerを用いて論理合成し、その後Avant!社のApolloを用いることにより単位セルを配置配線し、その単位セルを規則的に並べて配線することによって実際の回路のレイアウトを設計していく。

3.1　検出回路の動作記述と検証

２章において考えたアルゴリズムに基づき、実際の回路の設計をVerilog-HDLによって動作を記述することによって行った。また、回路を設計する際にそれぞれのアルゴリズムの正当性を確かめるためにシミュレーションを行った。

図3.1、3.2にはそれぞれ長方形の検出回路において長方形と長方形ではない物体を入力した場合のシミュレーション結果を図式化したものを、また図3.3、3.4には線分の検出回路において検出できる線分と検出できない線分の場合に分けて示す。

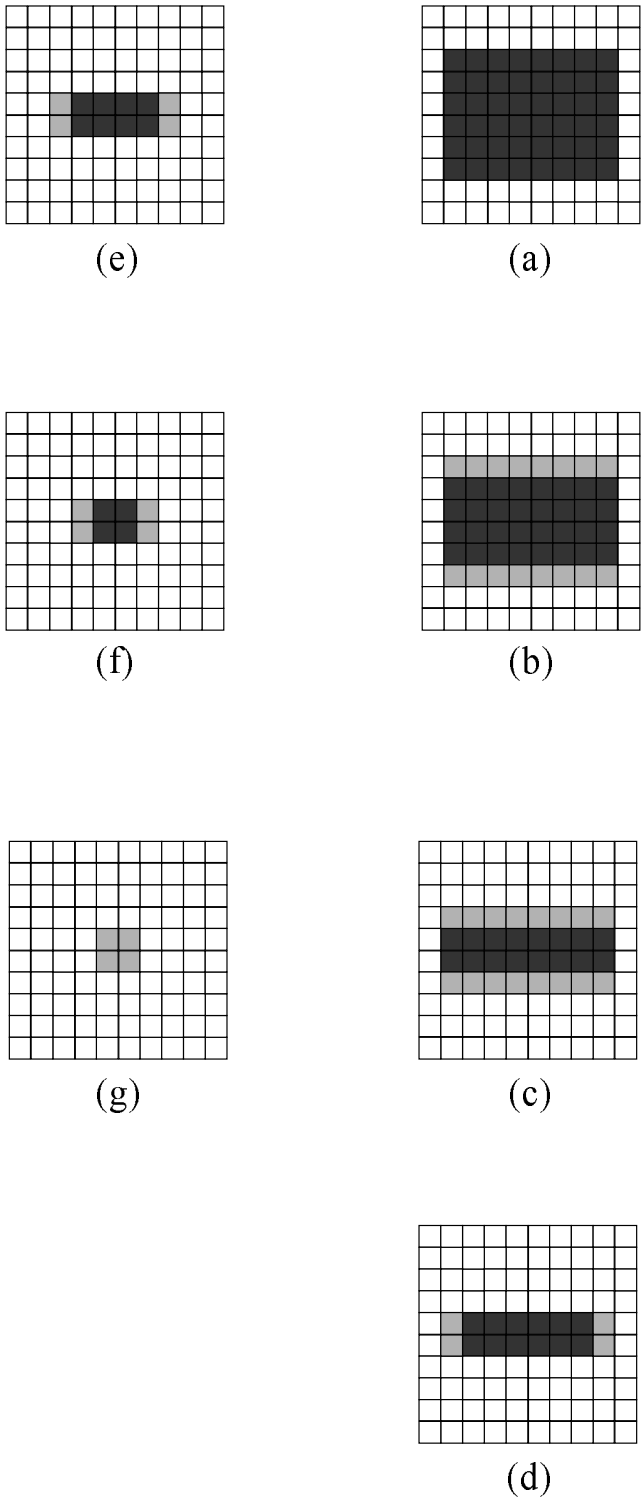


図 3.1　長方形検出アルゴリズムシミュレーション結果（長方形）

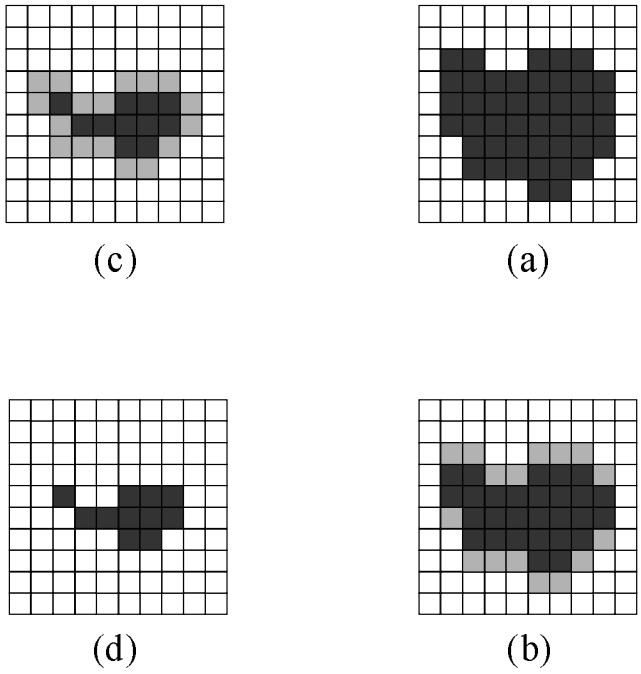


図 3.2　長方形検出アルゴリズムシミュレーション結果（非長方形）

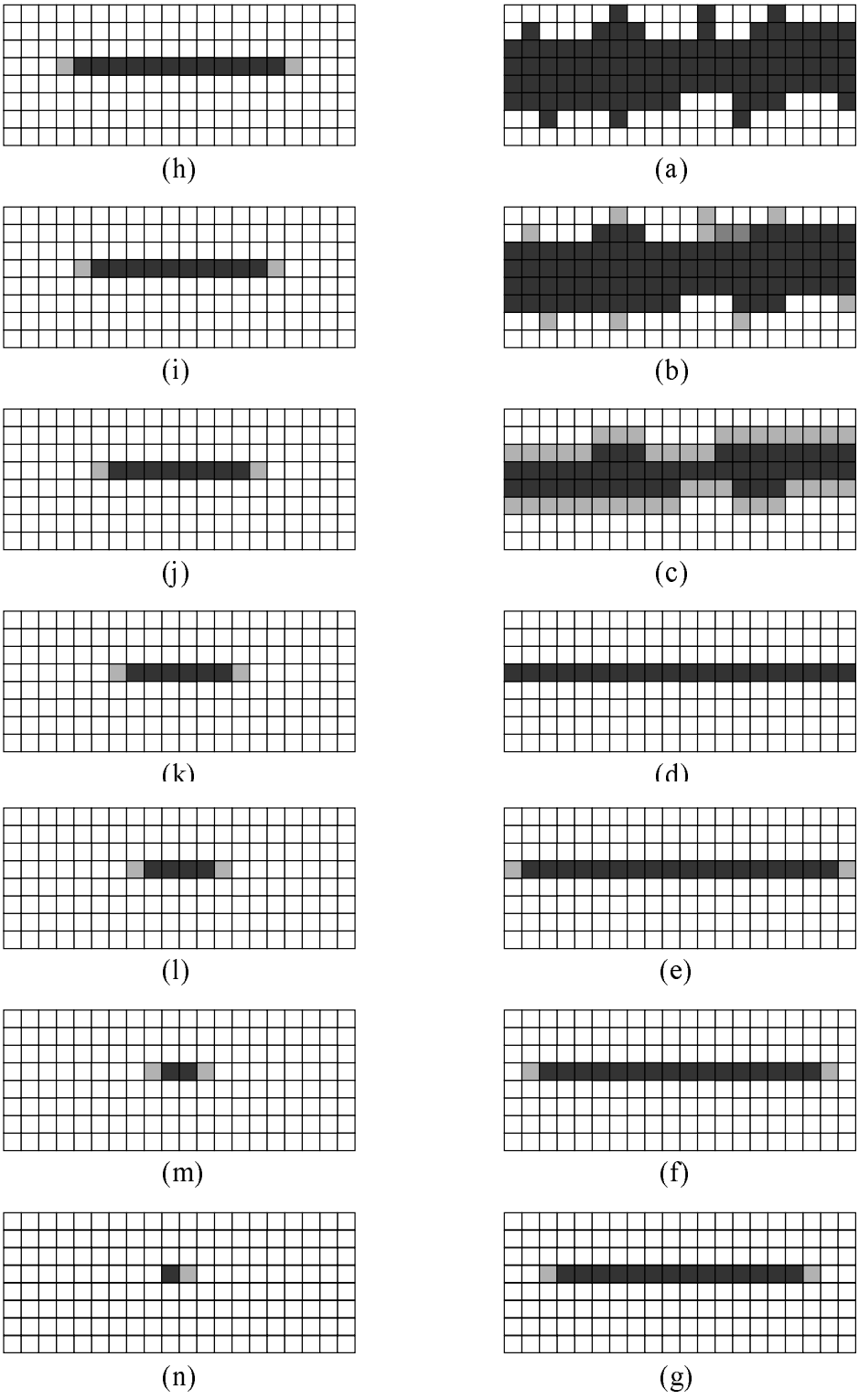
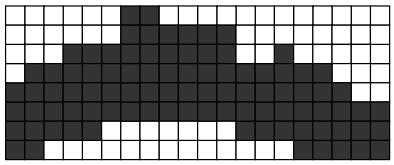
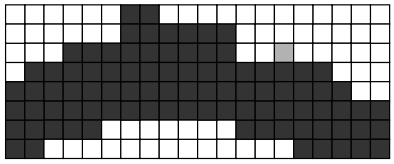


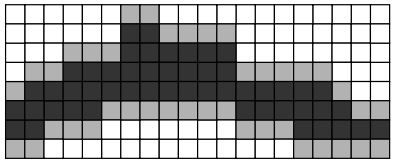
図 3.3　線分検出アルゴリズムシミュレーション結果（検出可能）



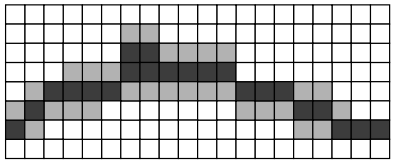
(a)



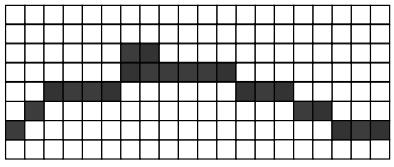
(b)



(c)



(d)



(e)

図 3.4　線分検出アルゴリズムシミュレーション結果（検出不可）

3.2 検出回路の論理合成と動作検証

3.1においてVerilog-HDLで動作を記述し、アルゴリズムの正当性を確かめた後、Synopsys社の論理合成ツールであるDesign-Analyzerを用いることによって論理合成を行った。論理合成を行うことによりゲートレベルの記述に変換することができ、ゲートの遅延時間や、仮想的な配線を行うことにより仮想配線遅延時間を求めることができる。その結果、遅延時間を含めてのより現実に近い状態のシミュレーションを行うことができる。

今回も長方形と線分を検出する回路それぞれにおいて、遅延時間を含めた状態でのシミュレーションを行い回路の正当性を確かめた。

3.3 線分検出回路のレイアウト設計

これまでに長方形検出することができる回路と、線分を検出することができる回路について回路の動作を記述し、そのシミュレーションを行うことにより、その正当性を確かめてきた。だが長方形を検出することができる回路は、ノイズなどがない正確な長方形が画素に対して水平垂直に取り込まれた場合にしか検出することができない。よってある程度の幅のばらつきや単発的な画像のノイズを無視することができる線分検出回路の方が、より現実的で応用範囲が広いといえる。よって以下において線分を検出することができる回路の実際のレイアウトを求めていく。

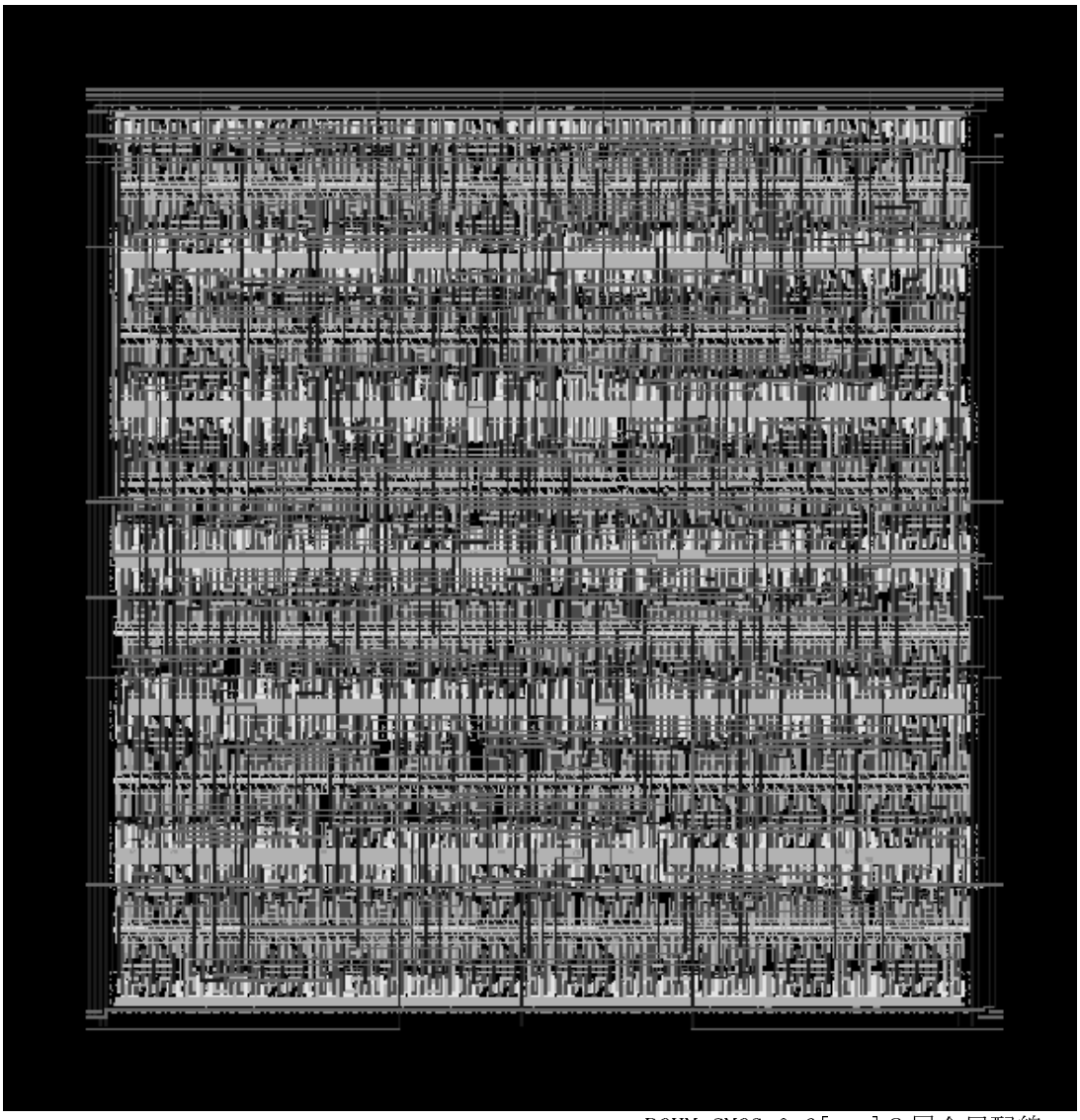
線分検出回路のレイアウトを設計するにあたり、まず単位セルのレイアウトを設計し、その後、設計した単位セルをマトリクス状に配置し、そのほかの全体の遷移関数を変更し線分が検出された場合にフラグをたてるためのセルや、各セルに画素の情報を与えるためのセルを配置し、それらを配線することによって線分検出回路のレイアウトを設計することとする。

3.3.1 単位セルのレイアウト設計

単位セルのレイアウトを設計する方法として、論理合成後のネットリスト（ゲートとそれらをつなぐ配線の情報）から、実際のセルの配置と配線をAvant!社のMilkywayとApolloを用いることによって行った。

ここで使用したプロセスはROHM CMOS 0.6[μm] 3層金属配線のプロセスを用い、ライブラリは東京大学のEXDライブラリを用いて行った。

単位セルのレイアウトを求めた結果、単位セルのレイアウトの大きさは約、360[μm]×360[μm]であり、そのトランジスタ数は2,162個であった。設計した単位セルのレイアウトを図3.5に示す。



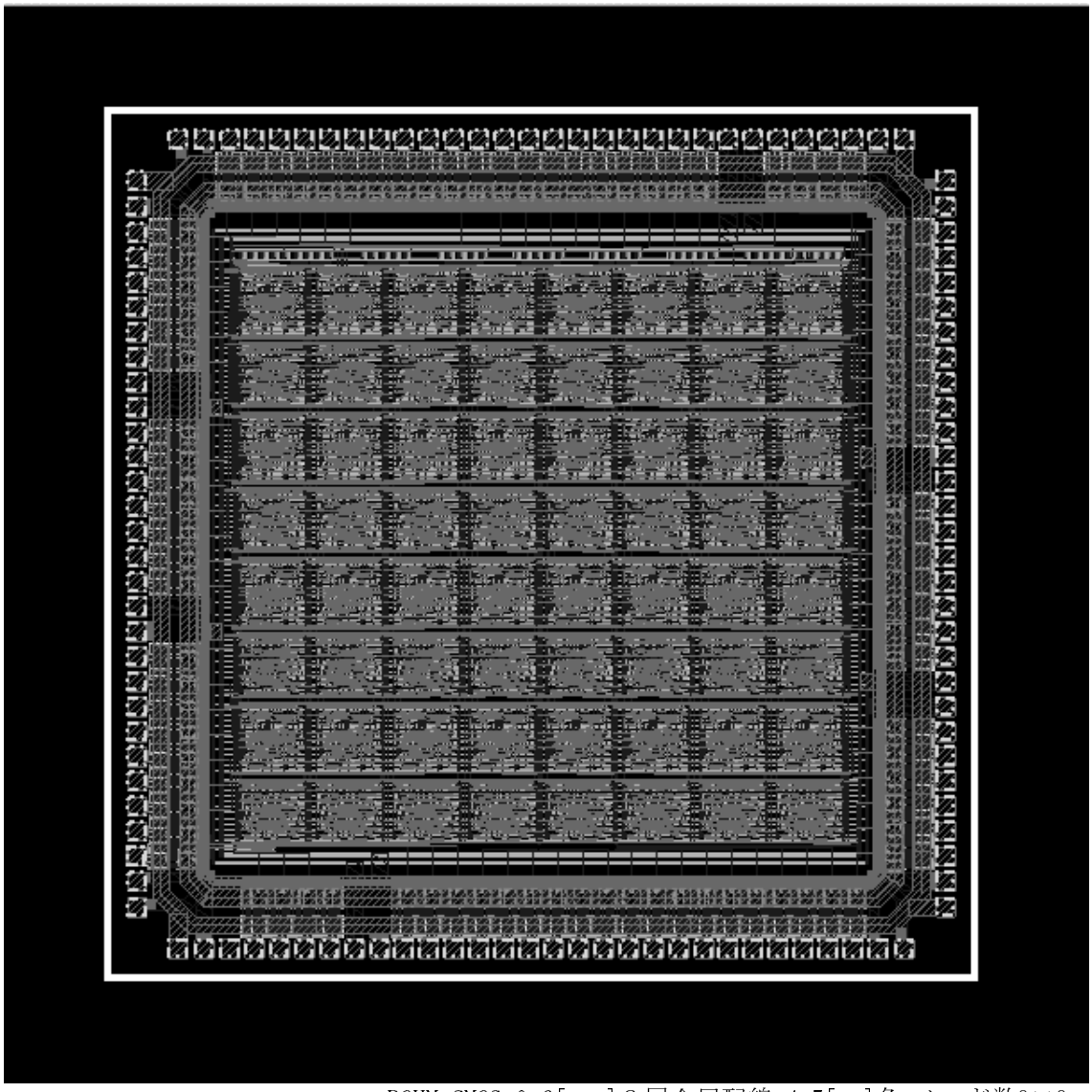
ROHM CMOS 0.6[μm]3層金属配線

図3.5　単位セルのレイアウト

3.3.2 線分検出回路のレイアウト設計

3.3.1で設計した単位セルのレイアウトの他に、全体の遷移関数を変更したり、線分を検出したときにフラグをたてたりするための制御セル、画素の情報を各セルに分配するための画像情報分配セルを単位セル設計のときと同様に設計し、それらをマトリクス状に配置された単位セルの脇に配置し、全体を配線することによって全体の回路のレイアウトを設計した。

また、線分検出回路のチップサイズは4.5[mm]×4.5[mm]角でノードの数は8×8個、トランジスタ数は140,962個であった。全体のレイアウトを図3.7に示す。



ROHM CMOS 0.6[μm]3層金属配線 4.5[mm]角 ノード数8×8

図3.6　線分検出回路のレイアウト

Chapter 4

考察とまとめ

これまでに長方形検出回路のアルゴリズムや、線分検出回路のアルゴリズムとその回路設計について述べてきたが、ここでは設計した線分検出回路についての考察及び、まとめを行っていく。

4.1 考察

はじめに線分検出回路の動作速度について考察すると、回路の動作をVerilog-HDLで動作を記述しその後論理合成を行ったが、その論理合成後の回路においてシミュレーションを行い、ゲートの遅延時間を含めた状態での動作速度を求めたところ 333[MHz]であった。だがこの動作速度はゲートの遅延時間のみしか考慮していないため、実際の動作速度とは大きく離れたものであると考えられる。そこで論理合成を行う際には回路をゲートレベルの記述にし、それらを仮想配線で配線しているが、その仮想配線による遅延時間は計算によって求めることができるため、それより求めた仮想配線による遅延時間とゲートの遅延時間の両方を含めた状態で線分検出回路の動作速度を求めてみたところ90[MHz]であった。この動作速度は実際設計した線分検出回路のレイアウトより求めた動作速度ではないため、実際にチップを作った後の回路での動作速度はもう少し遅くなることが予想されるが、ゲートの遅延時間のみを考慮した場合の動作速度よりは現実に近く、より信頼できる動作速度であるといえる。

また、今回の研究においては検出する物体の例として長方形及び線分を用いたが、検出することのできる物体の形状は回路の遷移関数を変更することによ

って安易に変更することが可能である。他に、単位セルそのものの働きが局所的であり、またChapter2でのアルゴリズムの考察より時間計算量は$O(n)$となることがわかっているため、画素数の増加に対して処理時間が莫大に大きくなることがなく、面積が許す限りセル数を比較的安易に増加させることができるという利点がある。

以上のことより、本研究において考えたセルオートマトンを用いて物体を検出するという手法は、超高速なロボットビジョンなど様々な分野への分野への応用が期待できるものであるといえる。

4.2 まとめ

本研究では、画素とセルオートマトンを平面上にマトリクス状に配置した構造を用い、隣接する画素や近傍のオートマトンの状態を受け取り遷移させる方法を用いた。また長方形及び線分を検出することができる手法についてそのアルゴリズムを考え、またシミュレーションによってその正当性を確かめた。この手法は各セルオートマトンが並列に動作することにより時間計算量が$O(n)$となり、従来の画像処理において行われてきたシーケンシャルな処理と比べて、少ないクロック数で長方形及び線分を検出することができ、構造が比較的シンプルなため高速動作が可能であることを示した。

また、その回路を設計するにあたり、Verilog-HDLで動作を記述しそのシミュレーションを行うことによってその正当性を確かめ、その後Synopsys社の論理合成ツールであるDesign_Analyzerを用いることによって論理合成を行いVerilog-HDLで記述した回路をゲートレベルの記述に直すことによって、ゲート遅延及び仮想配線遅延を含めた状態でのシミュレーションを可能とし、遅延情報を含めたシミュレーションにおいての回路の正常動作を確認した。その後、線分検出回路のレイアウトの設計を行ったが、まず単位セルのレイアウトをAvant!社の配置配線ツールであるApolloを用いて設計し、それら単位セルのレイアウトと、全体の遷移関数を変更するための制御部のセルや、各セルオートマトンへ画像の情報を与えるためのセルを単位セルを設計した方法と同じ方法で設計し、それらをCADツールを使って配置、配線する事によって線分検出回路全体のレイアウトを設計した。

また、考察における動作速度の見積もりより、実際のチップを作った後の回路においても十分な高速動作が期待できることを示し、今回与えた遷移関数とは異なる遷移関数を与えることにより他の形状を持つ物体を検出できることと、単位セルの動作の局所性より機能の拡張が容易であることより様々な物体の検出を行うことが可能なことより、画像処理をより高速に行うことができる可能性を示すことができた。

謝辞

本研究を行うにあたり多くの方々より御助言、御協力を頂きました。この場を借りて深く感謝の意を表したいと思います。

まず、指導教官としてご指導頂いた金沢大学工学部電気・情報工学科教授　故鈴木正國先生に深く敬意を示し、感謝すると共に心からのご冥福をお祈りします。また忙しい中、何度も遠路より足を運び研究面でのご指導、ご協力くださった北川章夫助教授に深く感謝します。研究するにあたり指導していただくだけでなく親身になり共に考えて頂き、また私生活においてもいろいろとお世話になった秋田純一助手、研究内容のより一層の向上のためにご教授を頂いた深山正幸助手に深く感謝します。機器の発注、工作や生活面でいろいろなご指導を頂きました柿本芳雄技官に深く感謝します。研究生活を行って行くにあたり様々な面で御協力いただいた集積回路工学研究室卒の中山和也助手に感謝します。半年間でありましたが、異境の地より私たちを指導するために来て頂いた Nandana Fernando さんに感謝します。また大学院生の小川明宏さん、高瀬信二さん、中橋憲彦さん、早川史人さん、藤井直樹さん、今井豊さん、数馬晋吾さん、藤田隼人さん、水野浩樹さん、渡辺晃さんには本研究を行う上での多大なるご指導、御協力を頂き、また温かい目で見守って頂き、心から感謝します。また同じ研究生として苦労を共にした笠井稔彦君、佐々木勝光君、高松直樹君、大門慎治君、辻川隆俊君、遠山治君、中村公亮君、水木誠君に感謝します。

大学での LSI 設計を可能にしてくれた大規模集積システム設計教育研究センターに感謝します。最後に、大学生活をとても有意義なものとし、研究面以外で私を支え、成長させてくれた金沢大学テニス同好会 LA-POMME のみなさんに感謝します。

参考文献

1. セルオートマトン法：複雑系の自己組織化と超並列処理
加藤恭義, 光成友孝, 築山洋共著-森北出版-1998

2. HDL によるデジタル設計の基礎
桜井至著-テクノプレス-1997

3. 改訂　新 C 言語入門　ビギナー編
林晴比古著　ソフトバンク

4. Verilog-HDL によるトップダウン設計
井上博史、鈴木隆訳　ＣＱ出版社