シストリックアレイによるユークリッド距離変換アルゴリズムの実現

宮澤 雅史† 曽 培峰† 磯 直行‡ 平田 富夫†

† 名古屋大学工学研究科電子情報工学専攻
〒 464-8603 名古屋市千種区不老町 1
Tel: 052-789-3440
Fax: 052-789-3089

‡ 中京大学情報科学部
〒 470-0393 愛知県豊田市貝津町床立 101
Tel: 0565-45-0971
Fax: 0565-46-1299

†miyazawa@hirata.nuee.nagoya-u.ac.jp, †zengpf@ieee.org,
‡fmiso@sccs.chukyo-u.ac.jp, †hirata@nuee.nagoya-u.ac.jp

あらまし ユークリッド距離変換はパターン認識やモルフォロジーフィルターなど画像処理の様々な分野で応用されている。本論文ではユークリッド距離変換のための、シストリックアレイを用いたハードウェアアルゴリズムを提案する。計算時間の効率化と乗算器などのハードウェア資源を削減するための工夫をし、サイズ $N \times N$ の 2 値画像に対しクロック数 $3N-1$ でユークリッド距離変換処理を実行する。そのために必要なハードウェア量は $O(N^2)$ である。

キーワード ユークリッド距離変換、シストリックアレイ、ハードウェアアルゴリズム、画像処理

Implementation of Euclidean Distance Transform on A Systolic Array

Masafumi Miyazawa† PeiFeng Zeng† Naoyuki Iso‡ Tomio Hirata†

†Graduate School of Engineering,
Nagoya University
Furo-cho Chikusa-ku Nagoya, 464-8603
Tel: +81-52-789-3440
Fax: +81-52-789-3089

‡Chukyo University
101 Tokodachi Kaizu-cho Toyota, 470-0393
Tel: +81-565-45-0971
Fax: +81-565-46-1299

†miyazawa@hirata.nuee.nagoya-u.ac.jp, †zengpf@ieee.org,
‡fmiso@sccs.chukyo-u.ac.jp, †hirata@nuee.nagoya-u.ac.jp

Abstract The Euclidean distance transform is applied in various fields of image processing, such as pattern recognition and morphological filter. In this paper, we propose a hardware algorithm using a systolic array that performs Euclidean distance transform. It is designed so that hardware resources, such as multipliers and comparators, are reduced and processing speed is efficient. It performs Euclidean distance transform for an input of $N \times N$ binary image in $3N-1$ clocks, and the size of the required hardware resources is $O(N^2)$.

Keywords Euclidean Distance Transform, Systolic Array, Hardware Algorithm, Image Processing

1 はじめに

距離変換とは、2 値画像の各画素について、それに最も近い 0 画素との距離を求めることである。距離変換はコンピュータビジョン、パターン認識、モルフォロジーフィルターおよびロボット工学などの分野で応用されている。これまでに、ユークリッド距離 (L_2) をはじめ、シティブロック距離 (L_1)、チェスボード距離 (L_∞)、八角形距離などに対する距離変換アルゴリズムが提案されてきた。ユークリッド距離変換に関して、Koloutnzakis と Kutulakos[3] は $N \times N$ の 2 値画像に対する $O(N^2 \log N)$ 時間のアルゴリズムを提案した。その後、[1][2] で計算時間は $O(N^2)$ に減じられた。特に [2] で提案されたアルゴリズムはユークリッド距離をはじめとする様々な距離に対して有効である。

一方で、距離変換アルゴリズムのハードウェア化に関しても研究がなされている。ロボットビジョンなどの画像処理の応用分野ではリアルタイム処理が要求されており距離変換アルゴリズムのハードウェア化は処理の高速化のため必要不可欠なものである。例えば、Paglieroni[5] は並列アルゴリズムを用いたアーキテクチャを提案している。しかし、このアーキテクチャでは多くのハードウェア資源を消費する。[6] で提案されたハードウェアアルゴリズムではハードウェア資源節約の工夫をしているが、ワークメモリに対してランダムなアクセスをするのでやはりハードウェア量が膨大となる。

シストリックアレイは H. T. Kung[4] が提案した VLSI アルゴリズムアーキテクチャである。このアーキテクチャでは、単純な計算能力を持つセルが規則的に配置され、クロックごとに計算および隣接セルとのデータ交換を行う。データの入出力は逐次的に行われ、処理は並列かつパイプライン的に動作する。シストリックアレイの利点は、データ交換が局所的に行われるので配線上の伝播遅延の影響が少ないこと、単純なセルの規則的な配列で構成されるため設計およびデバッグが容易であること、パイプライン処理、並列処理が効果的に利用できることがあげられる。

本論文では、[2] のアルゴリズムにもとづいて、シストリックアレイを用いたハードウェアアルゴリズムを提案する。このアルゴリズムは $N \times N$ の 2 値画像の距離変換をクロック数 $3N-1$ で行い、消費するハードウェア資源は $O(N^2)$ である。[2] で提案されたアルゴリズムは列ごとのスキャンと行ごとのスキャンの 2 つのステップから成り、この構成はハードウェア化をする際には非常に扱いやすい。このアルゴリズムでは N 個の 2 次関数の下側包絡線を求めるために 2 次関数どうしの交点を求めている。しかし、この処理を効率良く実行するハードウェア化は難しい。本論文では、シストリックアレイを用いることによって、交点を計算することなく下側包絡線を求める。また列ごとのスキャンでは通常 2 回のスキャンが必要であるが、本論分では一回のスキャンで処理が終わるように工夫している。

本論文の構成について述べる。第 2 章では距離変換の定義とユークリッド距離変換について述べる。第 3 章では我々の提案するハードウェアアルゴリズムについて説明する。第 4 章で動作速度と回路サイズを述べる。第 5 章はまとめである。

2 準備

$B = \{b_{i,j}\}(b_{i,j} \in \{0,1\})$ をサイズ $N \times N$ の 2 値画像とし、(i,j) を 2 値画像の i 行 j 列の要素とする。$B = \{b_{i,j}\}$ に対するユークリッド距離変換 $D = \{d_{i,j}\}$ は次の式で与えられる。

$$d_{i,j} = \min_{0 \le p,q \le N-1} \left\{ \sqrt{(i-p)^2 + (j-q)^2} |_{b_{p,q}=0} \right\}$$

[2] のアルゴリズムは、列のスキャン (**T1**) と行のスキャン (**T2**) の 2 つのステップで構成される。以下、**T1** と **T2** について説明する。

2.1 列のスキャン (T1)

T1 では 2 値画像 $B = \{b_{i,j}\}$ の各列における距離変換 $G = \{g_{i,j}\}$ を計算する。すなわち、各列 j に対して、

$$g_{i,j} = \min_{0 \le p \le N-1} \left\{ |i-p| \ |_{b_{p,j}=0} \right\}$$

を計算する。ただし、列 j の要素が全て 1 であるときは $g_{i,j} = \infty (0 \le i \le N-1)$ とする。

2.2 行のスキャン (T2)

T2 では **T1** で計算された $G = \{g_{i,j}\}$ からユークリッド距離変換 $D = \{d_{i,j}\}$ を計算する。画素 (i,j) か

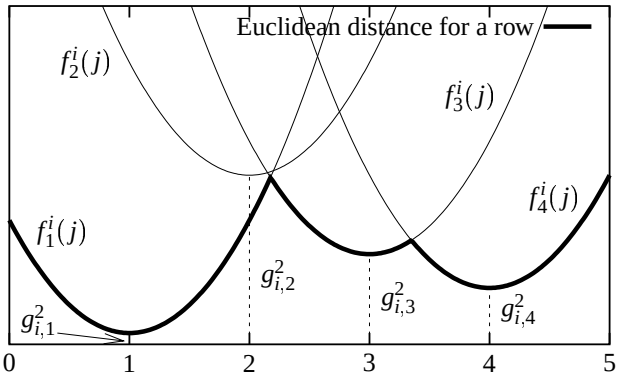

図 1: ユークリッド距離と下側包絡線

ら k 列内の最も近い 0 値の画素との距離は

$$f_k^i(j) = \sqrt{(j-k)^2 + g_{i,k}^2}$$

で与えられる。それゆえ、画素 (i,j) の距離値 $d_{i,j}$ を求めるには i 行内の $g_{i,k}(0 \le k \le N-1)$ をスキャンし、

$$d_{i,j} = \min_{0 \le k \le N-1} f_k^i(j)$$

を計算すればよい。

計算を簡単にするため、我々はユークリッド距離の 2 乗を求めることにする。つまり、$f_k^i(j) = (j-k)^2 + g_{i,k}^2$ とする。すると、$f_k^i(j)$ は j を変数とする 2 次関数とみなせる。そこで、行 i について、$0 \le k \le N-1$ のすべての 2 次関数 $f_k^i(j)$ をひとつのグラフに描くことを考える (図 1)。距離変換の定義から全ての曲線の下側包絡線が行 i の各画素の距離値 $d_{i,j}(0 \le j \le N-1)$ に対応する。

3 ハードウェアアルゴリズム

この章では、シストリックアレイを用いたハードウェアアルゴリズムを提案する。このアルゴリズムも [2] のそれと同様に T1、T2 の 2 つのステップから構成される。

3.1 T1 のアルゴリズム

T1 では列毎の処理を行うため列 j を固定して考える。**T1** をシストリックアレイで実現する方法ですぐ思いつくものは次のようなものである。図 2 のように N 個のセルが横方向に並んだ 1 次元のシストリックアレイを考え、2 値画像の列入力 $b_{0,j}b_{1,j}b_{2,j}\cdots b_{N-1,j}$ をこの順番にセル 0 に入力する。セル 0 はカウンタを一

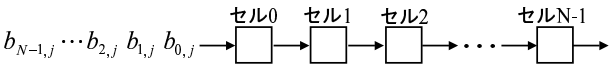

図 2: 簡単な **T1** のハードウェア構成

つ持ち、他のセルはレジスタを一つ持つ。セル 0 のカウンタは、0 が入力されると値を 0 にセットし、1 が入力されるとカウンタを $+1$ する。各レジスタおよびカウンタ値はクロックごとに右にシフトする。このようにすると、最後の入力 $b_{N-1,j}$ がセル 0 に入力されたとき、レジスタ値 0 のセル c より左側のセルにはセル c からの距離がレジスタにセットされた状態となる。セル c とそれより右側のセルとの距離を求めるため、列入力を逆順にセル $N-1$ に入力し同じことを行う。各セルにおいて順方向入力に対する距離と逆方向のそれとを比較し、小さい方を採用し **T1** の出力とする。この方法では **T1** の実現に列を 2 回スキャンしていることになる。また、各セルでは順方向と逆方向の距離値を保持するための二つのレジスタと一つの比較器が必要である。[6] では列のスキャンを一回ですます回路を提案しているが、セル間の通信が隣接したセルだけでなく遠く離れているセルどうしでも必要となる。このためハードウェア量は大きくなってしまう。

以下では、スキャンを 1 回で行うために各セルのレジスタに確定フラグ (Valid Flag、以下 VF と表記) を持たせる。その意味は、VF=1 のときそのセルでは、それから 1 番近いレジスタ値 0 を持つセル c との正しい距離がレジスタにセットされており、その値が確定していることを示す。VF の値はレジスタ値と共に、クロックごとに右へシフトする。ただし、セル 0 の VF にはクロックごとに 0 をセットする。また、左のセルから順に $0, 1, 2, \cdots, N-1$ の値を用意する (図 3)。これらを固定入力と呼ぶ。

まず、レジスタ値 0 のセル c より左側のセルでの距離値の確定を考える。0 がセル 0 に入力された後、連続で 1 が入力される状況で、どのような条件で距離値が確定するかを見ている (図 3)。図 3 ではフラグを丸印で表し、VF=0 のとき白丸で、VF=1 のとき黒丸で表す。セルのレジスタ値が確定するのは、そのセルの固定入力とそのセルより右側のレジスタ値 0 のセル c との距離が一致したときである。つまり、クロックご

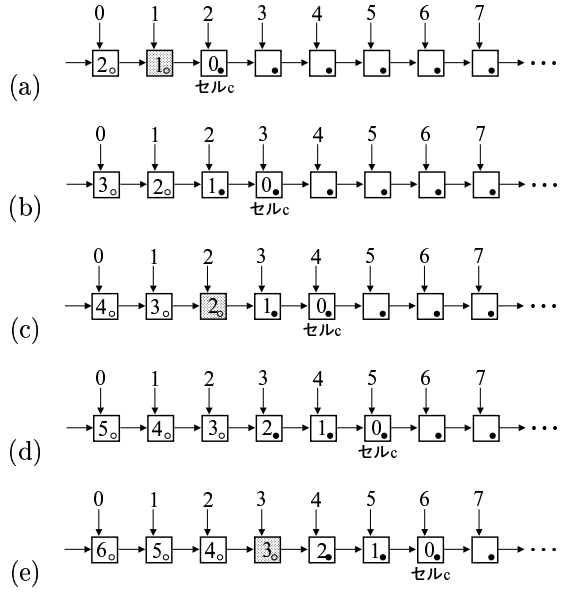

図 3: フラグ VF のセット

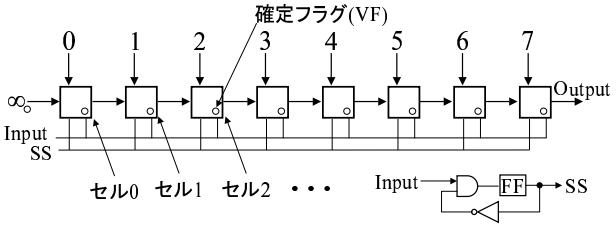

図 5: **TI** のハードウェア構成

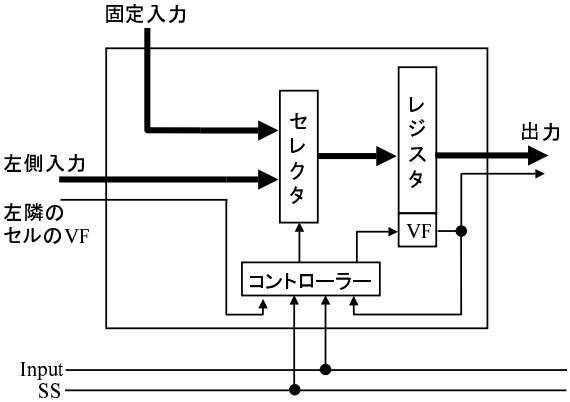

図 6: **TI** のセル i の内部構成

とにレジスタ値、カウンタ値および VF が右にシフトし、そのあと、レジスタ値が確定したかを調べ、確定したならば VF=1 とする。図 3 は、レジスタ値、カウンタ値および VF が右にシフトしたあとの状態であり、その後、網掛けのセルでのレジスタ値が確定する (このとき、VF=1 となる)。例えば図 3(c) では、セル 2 の固定入力が 2 で、セル 4 との距離が 2 であるためレジスタの値を確定する (VF=1 にする)。図 3(e) でも同様にセル 3 でレジスタ値が確定する。

次にレジスタ値 0 のセル c より右側のセルでの距離値のセットおよびレジスタ値の確定について考える。図 4 のようにすべてのセルに接続する **Input** 信号を導入する。列入力はセル 0 のほかに **Input** にも入力される。各セルはレジスタ値および VF が右にシフトし

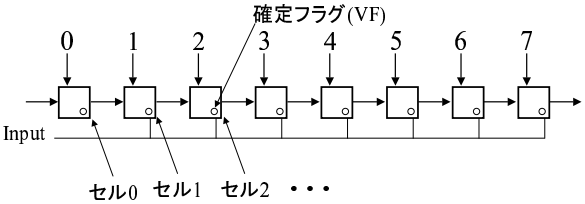

図 4: スキャンを 1 回で行う **TI** のハードウェア構成

たあと、**Input**=0 のとき VF=0 のセルのレジスタに固定入力をセットし VF=1 ととする。その結果、セル 0 より右側にあるセルで距離が確定していないセルのレジスタに、セル 0 からの距離がセットされる。これによりセル 0 より右側のセルのレジスタに 1 クロックで距離値がセットされる。

更に工夫することで、カウンタや比較器を用いずに距離値を確定することが出来る。まず、距離値は固定入力を直接レジスタにセットすればセル 0 のカウンタは必要としない。つまり、列入力は Input にのみ与えられる。また、セル 0 の左からの入力は ∞ が入るとする。これは列のすべての値が 1 の場合を考慮しているためである。次に、距離値をセットするタイミングについて述べる。レジスタ値 0 のセル c に近いセル順に距離をセットするので、あるセルにおいて、そのレジスタに入っている距離値が確定するとき、右隣のセルは VF=1 で自身は VF=0 である。さらに図 3 より、レジスタ値 0 のセルが右に 2 つシフトするたび距離が確定することがわかる。ここで、新しい制御信号 SS(Set Signal、以下 SS と表記) を導入する。SS はクロックごとに **Input** が 0 のとき 0 をとり、1 のとき以前の SS 値を反転した値をとる。SS は、列入力に 1 が連続しているとき、偶数番目と奇数番目を区別するための信号である。

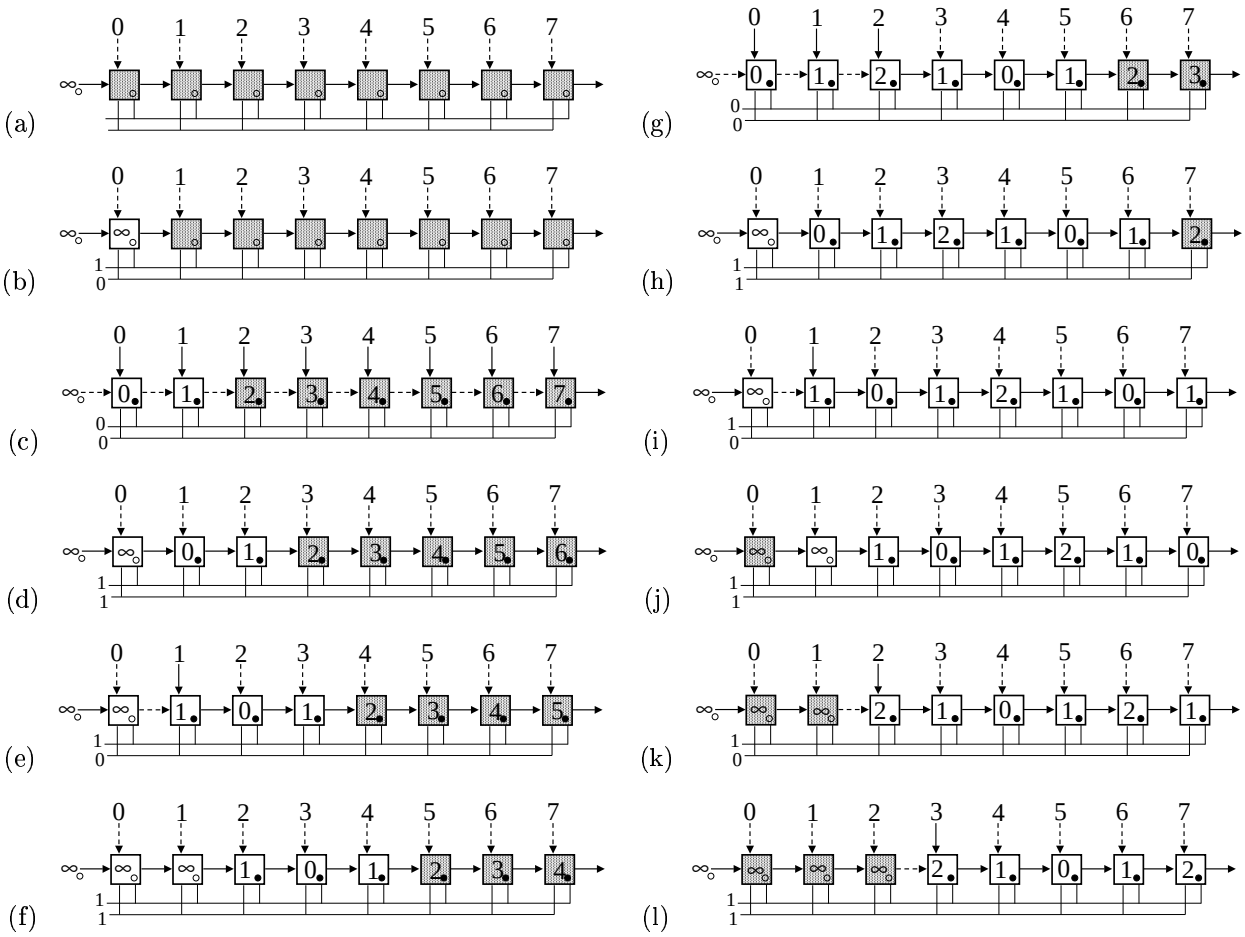

図 7: **T1** のアルゴリズムの例

提案するアルゴリズムは図 4 に SS 信号を加えた図 5 の構成になる。各セルの内部構成を図 6 に示す。図 5 の FF はフリップフロップであり、クロックごとに値が取り込まれ変化する。また、各セルはレジスタ値および VF が右にシフトしたあと、右隣のセルの VF、自身の VF、**Input**、SS の値から、固定入力をレジスタにセットし VF=1 にするかを決める。それは次の条件で決まる。

(a) **Input**=0 かつ自身のセルの VF が 0 のとき、固定入力をレジスタにセットし、自身の VF を 1 にする。

(b) SS=0 かつ自身のセルの VF が 0 かつ右隣のセルの VF が 1 のとき、固定入力をレジスタにセットし、自身の VF を 1 にする。

列データが入力され始めて N クロック後、**Output** には $g_{1,j}$ が出力される。それ以降、$g_{2,j}, g_{3,j}, \ldots$ が順次出力される。N クロック以降、つまり $b_{N-1,j}$ が入力された後の **Input** には N 個の 1 を入力することとする。**T1** を実行するにはクロック数 $2N$ が必要である。

例として 2 値画像の列データが $b_7 b_6 b_5 \ldots b_0$ = 11011101 のときのセルの振る舞いを図 7 に示す。図 7 ではクロックごとに (a)→(b)→(c)→ $\cdots$ の状態をとる。各状態は、レジスタ値および VF がシフトし、固定入力がセットされた後の状態である。(a) は初期状態である。実線の矢印の入力がセル内のレジスタにセットされる。図 7 の 8 クロック目の状態 (h) ではまだセル 0 のレジスタ値が確定していない。しかし 2 クロック後、正しい値がセットされる。ある時点でレジスタになにもセットされてなくても **Output** に出るまでに

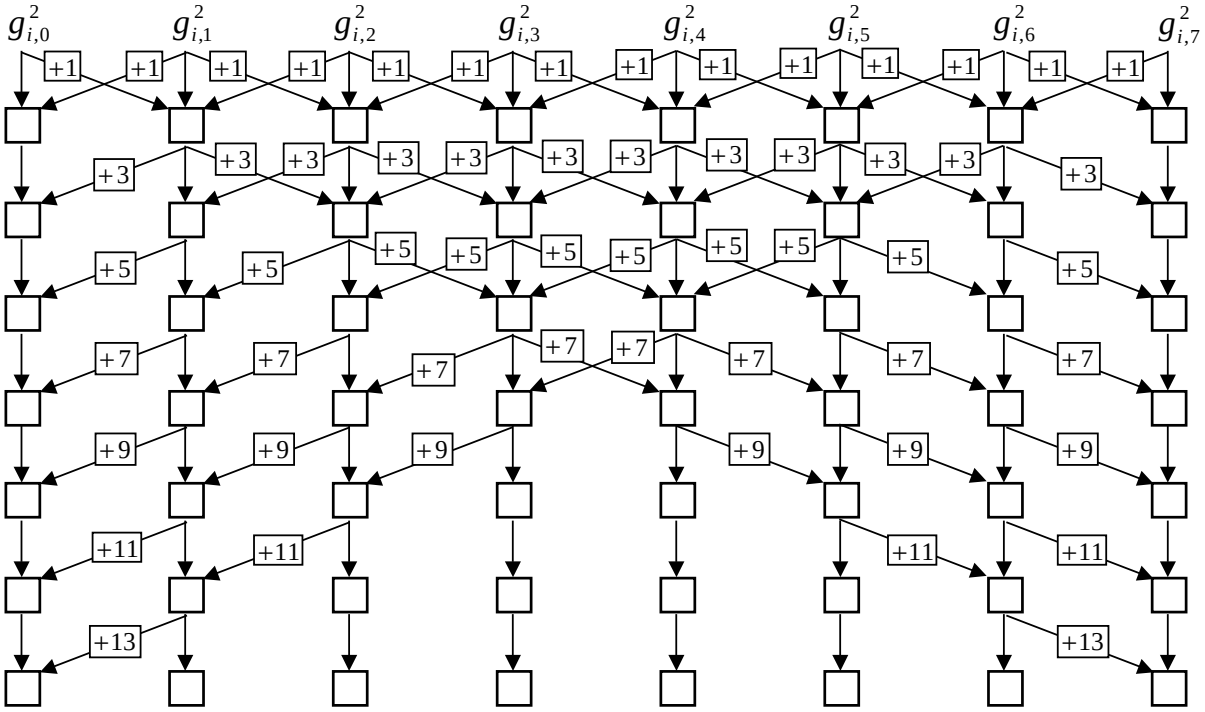

図 8: **T2** のハードウェア構成

正しい値がセットされる。

最後に乗算器を削減する方法を述べる。**T2** では **T1** の出力を 2 乗したもの、すなわち $g^2_{i,j}$ を用いるので、このために乗算器が必要となる。**T1** ではセル内のレジスタにセットされる値は固定入力値であり、一度セットされると **T1** の処理中に変化することはない。よって **T1** の固定入力に初めから 2 乗した値を用意すればこの乗算器は必要ない。

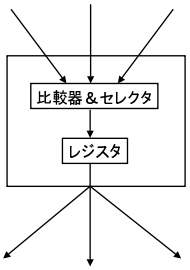

図 9: **T2** のセル i, j の内部構成

3.2 T2 のアルゴリズム

T1 のときと同様に、**T2** は行毎の処理であるため行 i を固定して扱う。[2] では下側包絡線を得るため 2 次関数どうしの交点を求めているが、この処理を実行する効率的なハードウェア化は困難である。そこで、シストリックアレイ構造を利用した次のようなハードウェアアルゴリズムを考える。

2 次関数 $f^i_k(x)$ の値は $x = k$ から 1 画素離れるごとに $1, 3, 5, 7, \ldots$ ずつ増えてゆく。よって、クロックごとに $g^2_{i,j}$ に $1, 3, 5, \ldots$ を順次加えていき $f^i_k(x)$ の値を計算する。

T2 を計算するハードウェア構成を図 8 に示し、セルの内部構成を図 9 に示す。入力は **T1** の出力の 2 乗、すなわち $g^2_{i,j}$ である。各セルはクロックごとに最小な入力値をレジスタにセットする。

1 クロック後の図 8 の上から 1 行目のセルには、すべての 2 次関数 $f^i_k(x)(0 \le k \le N-1)$ の $k-1 \le x \le k+1$ の範囲で構成される下側包絡線の値が保持される。次のクロックでは 2 行目のセルにすべての関数の $k-2 \le x \le k+2$ の部分で構成される下側包絡線の値が保持される。

$0 \le x \le N-1$ のすべての範囲で関数 $f^i_0(x)$ および $f^i_{N-1}(x)$ がその関数値を計算するにはクロック数 $N-1$ が必要である。また、すべての行についてパイプライ

ン的に計算できるので、このアルゴリズムでは、すべての行に対する下側包絡線を求めるためにクロック数 $2N-1$ を必要とし、$N \times N-1$ 個のセルが必要である。

4 動作速度と回路サイズ

2 値画像の入力を $N \times N$ とする。セル間でのデータ交換およびセル内の計算を 1 クロックで行うとする。**T1** の処理と **T2** の処理は、それぞれ $2N$ クロック、$2N-1$ クロックかかる。しかし、**T1**、**T2** は共に行毎にデータが入出力かつシフトするので、**T1** の出力と **T2** の入力を接続したとき、全体でパイプライン的に動作でき、全体の処理で必要なクロック数は $3N-1$ となる。

次に回路サイズについて考える。**T1** では各セルについてレジスタおよびセレクタが必要であるので、それぞれ N^2 個要する。**T2** ではレジスタが $N(N-1)$ 必要で、加算器、比較器については $O(N^2)$ 必要である。また、各セルの状態を決定するコントローラーとセルのフラグ、そして配線資源が必要である。

5 まとめ

2 値画像に対するユークリッド距離変換を行うハードウェアアルゴリズムを、シストリックアレイを用いて実現した。アルゴリズムは [2] のアルゴリズムに基ずいて構成された。**T1** の変換を 1 回のスキャンで実現し、乗算器を使わずに下側包絡線を得ることが出来る。クロック数 $3N-1$ で処理を行い、ハードウェア量は $O(N^2)$ である。

参考文献

[1] L. Chen and H. Y. H. Chuang, "A fast algorithm for Euclidean distance of 2-D binary image" *Information processing letters*, 51, pp.25-29, 1994.

[2] T. Hirata, "A unified linear-time algorithm for computing distance maps" *Information processing letters*, 58, pp.129-133, 1996.

[3] M. N. Kolountzakis and K. N. Kutulakos, "Fast computation of Euclidean distance maps for binary images" *Information processing letters*, 43, pp.181-184, 1992.

[4] H. T. Kung, "Why systolic architecture?" *IEEE Computer*, 15, 1, pp.37-46, 1982.

[5] D. W. Paglieroni, "A unified distance transformation algorithm and architecture" *Machine Vision Appl.* 5, pp.47-55, 1992.

[6] P.F.Zeng and T. Hirata, " ユークリッド距離変換アルゴリズムのハードウェア化に関する研究" 情処研報, 2002-AL-84, pp.17-24, 2002.