〈一般研究課題〉　生活用携帯情報機器のための
高速距離変換VLSIの実現に関する研究

助 成 研 究 者　中京大学　　磯　直行

生活用携帯情報機器のための 高速距離変換VLSIの実現に関する研究

磯　直行
（中京大学）

Implementation of Fast Distance Transform VLSI for Mobile Terminals

Naoyuki Iso
(Chukyo University)

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 for mobile terminals 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 $O(N^2)$. And, this algorithm was tried to implement in a field programmable gate array (FPGA).

1. はじめに

距離変換は，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処理である．距離変換はコンピュータビジョン，パターン認識，モルフォロジフィルタ及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いる．[1]では，ユークリッド距離変換に関して $N\times N$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

ている．その後，[2]～[4]で計算時間が$O(N^2)$に改良された．特に[3]では，ユークリッド距離をはじめとする様々な距離定義に対しても有効な距離変換アルゴリズムを提案している．

一方，距離変換アルゴリズムのハードウェア化に関する研究も行なわれている．特に，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求められており，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には必要不可欠なものである．例えば，[5]では並列アルゴリズムを用いたアーキテクチャを使用しているが，多くのハードウェア資源を消費してしまう．ハードウェア資源を消費することは回路規模が大きくなることであり，従って消費電力が大きくなることを意味する．そのため，携帯端末やロボットのような電池で動作する機器で行う処理では，ハードウェア資源をできるだけ削減する工夫が必要である．[6]で提案されたハードウェアアルゴリズムでは，ハードウェア資源を節約する工夫をしているが，ワークメモリに対してランダムなアクセスを行なうために実際に必要とするハードウェア量は膨大である．

本論文は，シストリックアレイを用いた距離変換アルゴリズムに注目する．シストリックアレイは，H.T. Kungが[7]で提案したVLSIアルゴリズムアーキテクチャである．シストリックアレイは，単純な計算能力を持つセルが規則的に配置され，各セルで与えられたクロックごとに計算および隣接セルとの間でデータ交換を行なう．データの入出力は逐次的に行なわれ，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のみ行なわれるので遠距離配線による信号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及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適用できることである．

本論文では，[3]のアルゴリズムに基づいたシストリックアレイによるハードウェアアルゴリズムをVLSIとして実現することを目標とする． [3]のアルゴリズムの構成は，列ごとのスキャン(T1)と，行ごとのスキャン(T2)の2つの単純なステップからなり，ハードウェア化するには非常に扱いやすい．また，[3]のアルゴリズムでは $N\times N$の入力画像に対して， N個の2次関数の下側包絡線を求めるために2次関数同士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を実現するのは難しい．そこで本論文は，シストリックアレイを用いることにより，2次関数同士の交点を計算することなく下側包絡線を求めることにより高速化する．また，列ごとのスキャン(T1)では通常2回のスキャンが必要であるが，本論文では1回のスキャンで処理が終わるようにさらに高速化する．この結果，本論文で提案するアルゴリズムは $N\times N$の2値画像の距離変換をクロック数$3N-1$で行い，$O(N^2)$のハードウェア資源を消費する．そして，本論文で提案したアルゴリズムの実用性を確かめるため，ソフトウェア的に内部回路を書き換え可能なハードウェアとしてFPGA(Field Programmable Gate Array)上に実現することを試みる．

本論文の構成について述べる．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．3章で提案するハードウェアアルゴリズムについて述べる．4章で動作速度と回路サイズ，およびFPGAによる試作について述べる．5章でまとめる．

2. 定義

$B=\{b_{i,j}\}$ $(b_{i,j}\in\{0,1\})$ 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．

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

ユークリッド距離変換を行なう[3]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の処理についてそれぞれ説明する．

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，

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

を計算する．ただし，列jの要素がすべて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．よって，画素(i,j)の距離値$d_{i,j}$を求めるためには，i行内の$g_{i,j}\ (0 \le k \le N-1)$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，本論文で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．このようにすると$f_k^i(j)$は$j$を変数とする2次関数とみなすことができる．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1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，すべ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．

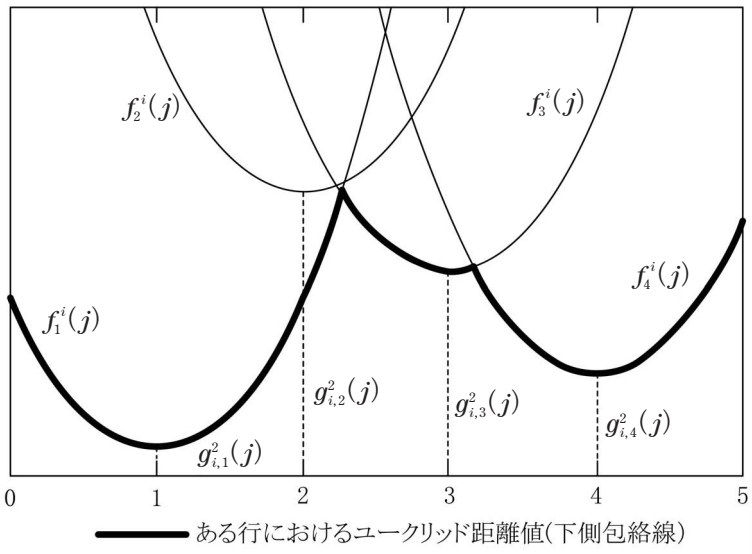

図1　ユークリッド距離と下側包絡線

3. ハードウェアアルゴリズム

本章では，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[3]のアルゴリズムと同様に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では列ごとの処理を行なうため，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ときの単純手法は次の通りである．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$b_{0,j}\,b_{1,j}\,b_{2,j}\cdots b_{N-1,j}$をこの順にセル0に入力する．セル0はカウンタを1つもち，他のセルはレジスタを1つもつ．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はクロックごとに右へシフトする．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になる．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なう．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，小さいほうを採用しT1の出力とする．この方法ではT1の実現に列を2回スキャンする．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2つのレジスタと1つの比較器が必要である．[6]では，列のスキャンを1回で済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同士でも必要となり，セル間を接続するための配線のハードウェア量が大きくなってしまう．

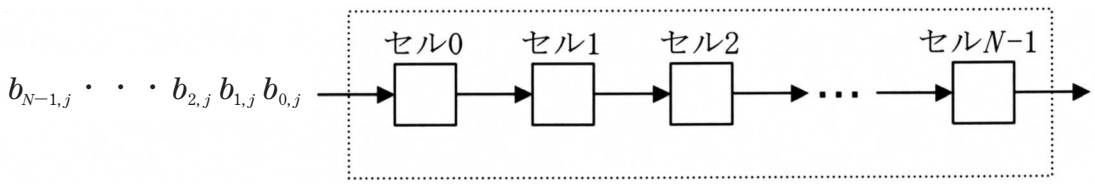

図2　簡単なT1のハードウェア構成

本論文では，スキャンを1回で行な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．VF=1のとき，そのセルではそれから最も近いレジスタ値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ともにクロックごとに右へシフトする．ただし，セル0のVFの値はクロックごとに0をセットする．また，左のセルから順に0,1,2・・・,N−1の値を用意する（図3）．これらを固定入力と呼ぶ．

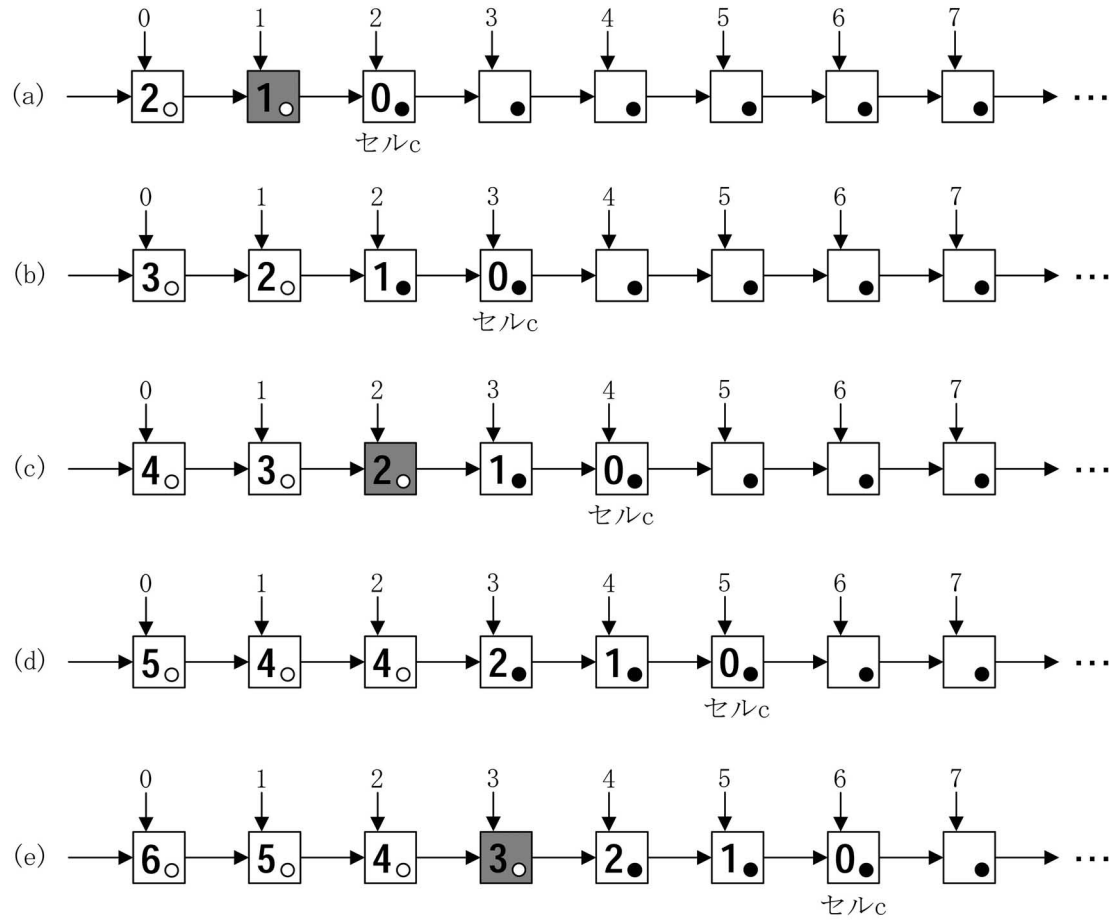

図3　フラグVFのセット

まず，レジスタ値0のセルcより左側のセルでの距離値を確定することを考える．図3は，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表している．図3では，フラグを丸印で表し，VF=0のとき白丸で，VF=1のとき黒丸で表している．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も右側のレジスタ値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，その後，レジスタ値が確定したかどうかを調べ，確定したならVF=1とする．図3はレジスタ値，カウンタ値およびVFが右にシフトした後の状態であり，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を確定するときを考える．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．

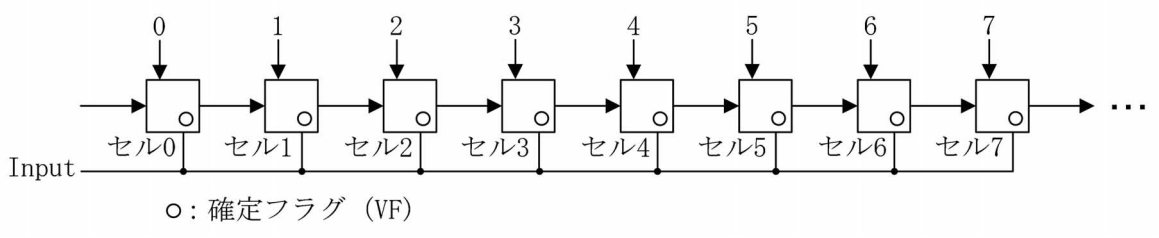

図4　スキャンを1回で行なうT1のハードウェア構成

さらに工夫することで，各セルにカウンタや比較器を用意することなく距離値を確定することができる．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∞が入るものとする．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 Singal，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，偶数番めと奇数番めを区別する信号である．

提案するアルゴリズムは，図4にSS信号を加えた構成である(図5)．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にするかどうかを決定する．それは次の条件によって決定される．

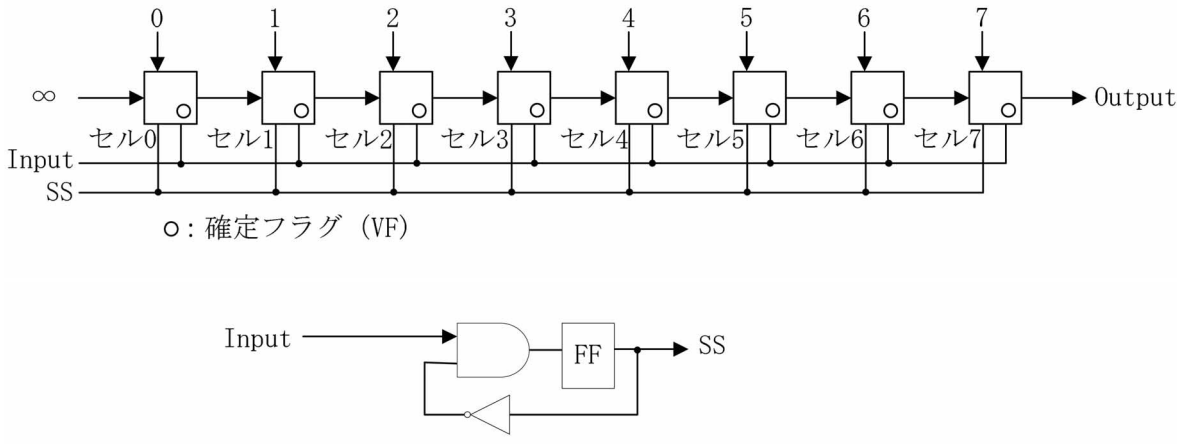

図5　T1のハードウェア構成と制御信号SSの生成

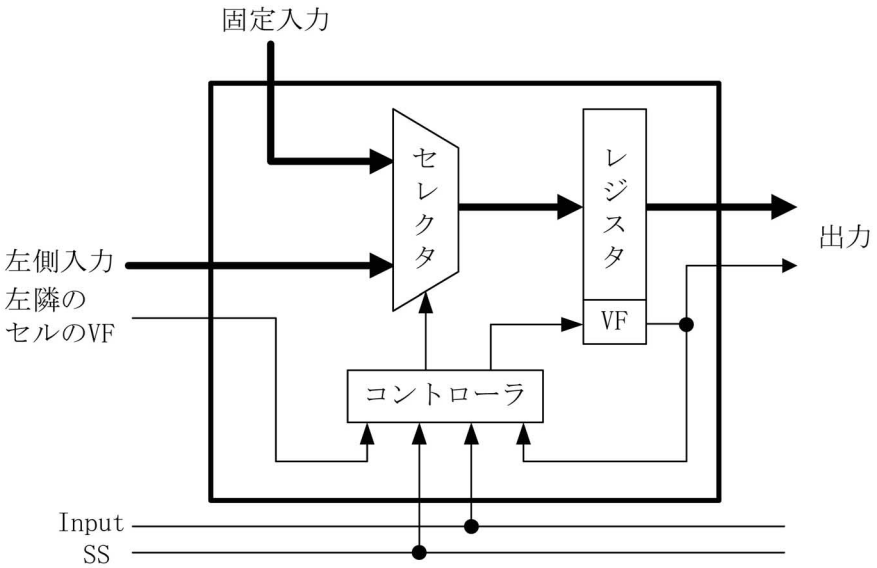

図6　T1のセルの内部構成

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

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

列データが入力され始めてNクロック後，Outputに$g_{1,j}$が出力される．それ以降，$g_{2,j}$，$g_{3,j}$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$b_7b_6b_5\cdots b_0$=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)→…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．実線の太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(i)では，まだセル0のレジスタ値は確定していない．しかし，2クロック後の状態(k)で正しい値がセットされる．ある時点でレジスタに値がセットされていなくてもOutputに出るまでの間に正しい値がセットされる．

最後に乗算器を削減する方法について述べる．T2ではT1の出力を2乗したもの，すなわち，$g_{i,j}^2$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の固定入力にはじめから2乗した値を用意しておけばこの乗算器は必要ない．

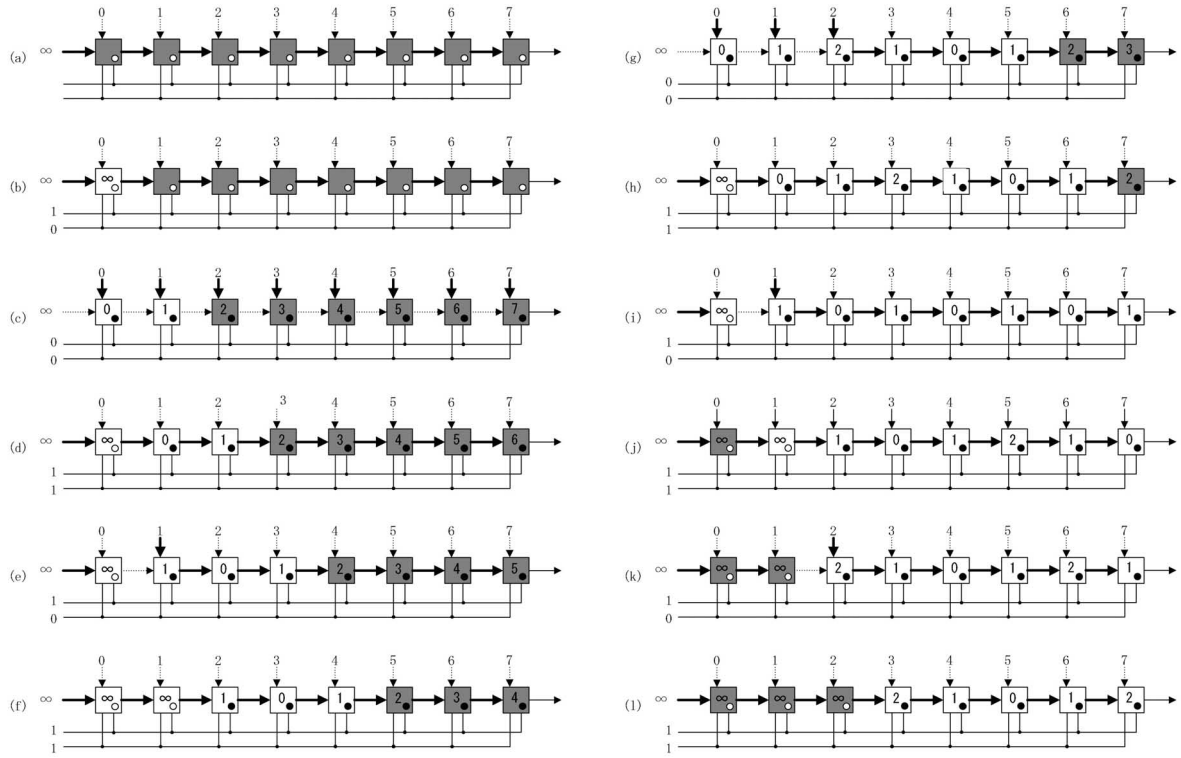

図7　T1のアルゴリズムの動作例

3.2 T2のアルゴリズム

T2は行ごとの処理であるため，T1のときと同様，行iを固定して扱う．[3]では下側包絡線を得るため2次関数同士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の実現は困難である．そこで，シストリックアレイ構造を利用した次のハードウェアアルゴリズムを考える．

2次関数$f_k^i(x)$の値はx=kから1画素離れるごとに1,3,5,7,…ずつ増えていく．よって，クロックが入力されるごとに$g_{i,j}^2$に1,3,5,…を順次加えていき$f_k^i(x)$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，すなわちで$g_{i,j}^2$ある．各セルはクロックが入力される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2次関数の$f_k^i(x)$ $(0 \leq k \leq N-1)$の$k-1 \leq x \leq k+1$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$k-2 \leq x \leq k+2$の部分で構成される下側包絡線の値が保持される．

$0 \leq x \leq N-1$のすべての範囲で関数$f_0^i(x)$がその間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．

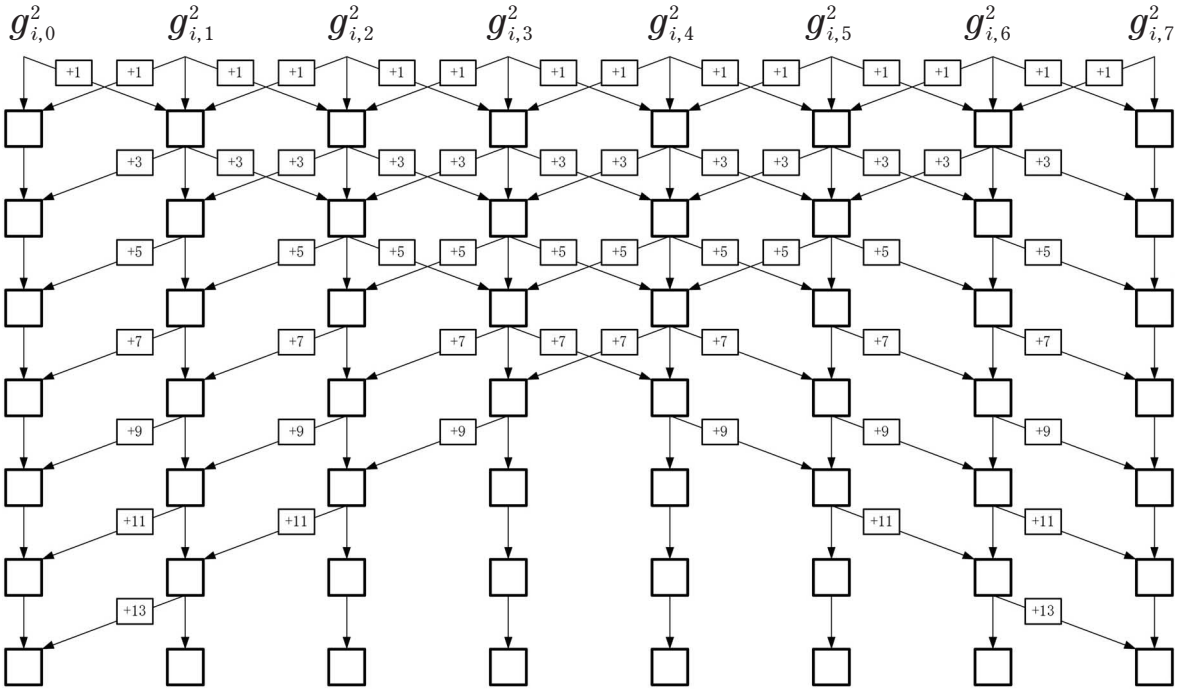

図8　T2のハードウェア構成

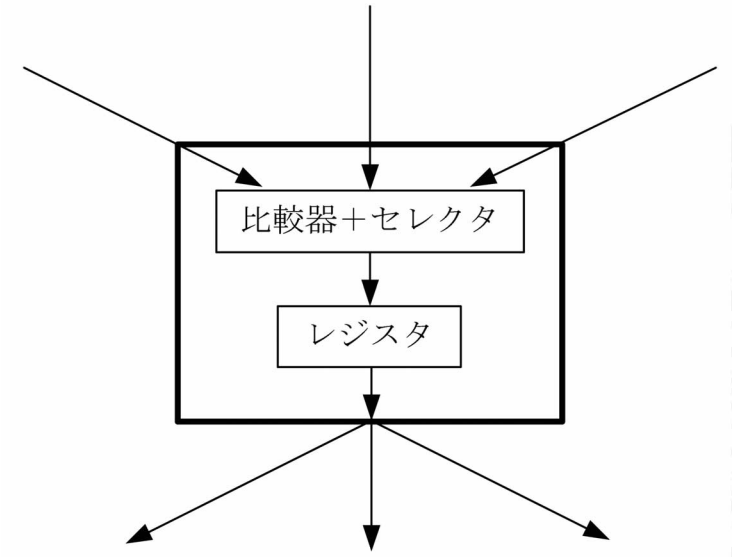

図9　T2のセル*i*, *j*の内部構成

4. 動作速度と回路規模，FPGAによる試作

$N\times N$の2値画像を入力とする場合を考える．セル間のデータ交換及びセル内の計算を1クロックで行なうものとする．T1の処理とT2の処理は，それぞれ$2N$クロック，$2N-1$クロック必要である．しかし，T1，T2はともに行ごとにデータ入出力とシフトが行なわれるので，T1の出力とT2の入力を直接接続すればパイプライン的に動作でき，処理全体で必要なクロック数は$3N-1$である．

次に回路規模について考える．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個必要である．T2ではレジスタが$N(N-1)$個必要で，加算器，比較器については

$O(N^2)$個必要である．また，各セルの状態を決定するコントローラとセルのフラグ，そして配線資源が必要である．

そして，このユークリッド距離変換ハードウェアアルゴリズムの実用性を確認するため，ソフトウェア的に内部回路を書き換え可能なハードウェアとしてField Programmable Gate Array (FPGA)を使用した試作を行なう． FPGAはVLSIよりも動作速度は遅いながらも回路の動作は十分確認できる．試作は現在進行中であるが， VLSIに近い大規模FPGAが搭載された評価ボードを用いることで，実現可能性に対する信憑性は高いと考えられる．FPGAの設計には、東京大学大規模集積システム設計教育研究センター（VDEC）から提供された各種EDAツールを利用した．

5. おわりに

2値画像に対するユークリッド距離変換を行な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[3]のアルゴリズムに基づいて構成された．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できる．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．また，提案したハードウェアアルゴリズムの実用性を確認するためFPGAによる試作を行なった．

参考文献

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

[2] 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.

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

[4] M. Miyazawa, P.F. Zeng, N. Iso and T. Hirata, “A Systolic Algorithm for Euclidean Distance Transform,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 28, 7, pp.1127-1134, 2006.

[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.

[7] H.T. Kung, “Why systolic architecture?,” IEEE Computer, 15, 1, pp.37-46, 1982.