ＦＰＧＡ向け超高速画像処理システム開発環境の構築

代表研究者　　伊藤　靖朗　　広島大学　大学院工学研究院　准教授

1 はじめに

HDTV や Blu-ray 等の新しい技術が登場し，高画質・高解像度のリアルタイムな画像処理に対するニーズが今後ますます高くなると予想される．現在このような処理を行うには，処理内容をあらかじめ決めて ASIC 上にハードウェア化した専用 LSI が広く用いられている．しかし半導体の高集積・高密度化が進んできたとはいえ回路規模にも制限があり，ユーザが要求する全ての機能を搭載するのは困難である．そこで，書き換え可能な LSI である FPGA (Field Programmable Gate Array)を用い，動的に回路を書き換えることでソフトウェアのようにハードウェアを扱う手法が広く用いられている．

そこで本研究では，デジタル信号処理のために FPGA 内に多数配置されている DSP Block や Block RAM などの組み込み回路を利用し，現在広く用いられる専用 LSI よりさらに一歩進んだ超高速画像処理回路の作成を行う．

本研究では画像処理の例としてハフ変換を用いた直線検出のハードウェアの設計を行った．次にこれらについて説明する．

2　FPGA

最新の FPGA は，任意の順序回路を埋め込むことができる CLB (Configurable Logic Block) 以外に，デジタル信号処理(DSP) 用途に DSPB (DSP Block) と BRAM (Block RAM) が多数配置されている(図 1)．アルゴリズムを回路化して FPGA に埋め込むことにより，処理の高速化を図る研究が盛んに行なわれているが，いかに DSPB と BRAM を効果的に利用するかという点が鍵となっている．例えば，申請者は，暗号化のアルゴリズムを DSPB と BRAM を効率よく利用するように回路化して FPGA に埋め込み，従来ソフトウェアによる処理では達成できない高速処理が可能であることを実証した．しかし，汎用計算に FPGA を用いられることは圧倒的に少ない．これは，アルゴリズムの回路化自体かなり困難であり，さらに FPGA の DSPB と BRAM を効果的に利用するには，職人的なノウハウと複雑な作業が必要なためである．上記のような背景から，本研究では，画像処理の例として直線のハフ変換のハードウェアの設計を行った．次にこれらについて説明する．

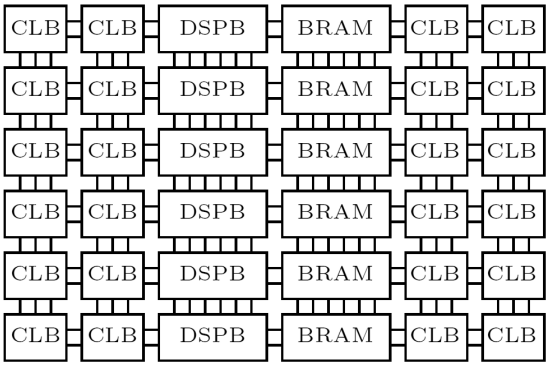


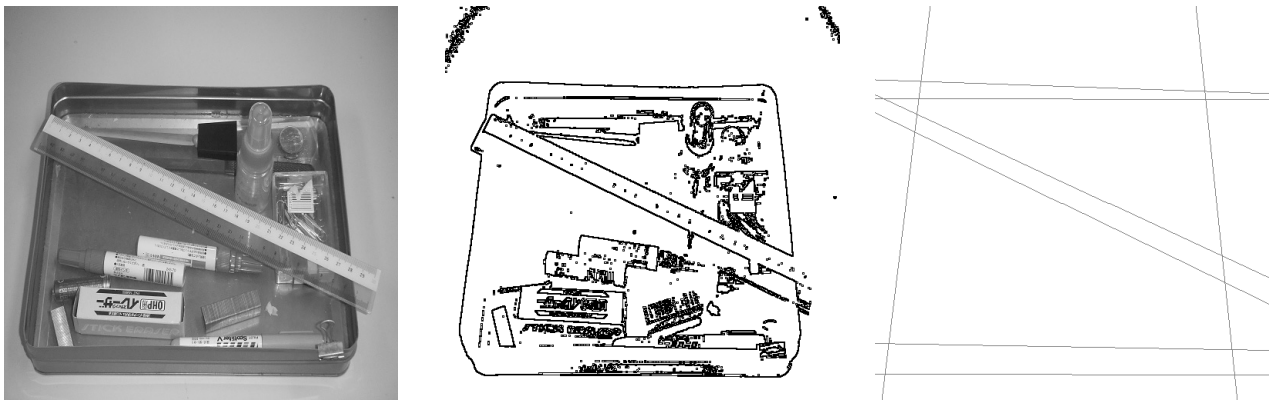
図 1: FPGA の内部構成

3 ハフ変換

ハフ変換とは，画像中の様々な形状を見つける技術で，特に直線，円，楕円，任意形状の図形を抽出するために利用されてきた[1]．ハフ変換は蓄積配列によって表されたパラメータ空間への写像を定義する．そのパラメータ空間は検出する形状をパラメータ化することによって定義され，画像中のエッジ点に対して，その写像は，蓄積配列のある要素に投票することに対応する．投票される要素は検出する形状に基づいたパラメータを表しているので，高頻度で投票される要素は，画像中の形状が表すパラメータに対応している．

ハフ変換は２値画像中の直線検出に用いられている [2]．この手法のアイデアは，直線上の点とその直線

のパラメータの対応を利用することである．画像中のある点は，パラメータ空間の曲線によって表され，直線上の点はパラメータ空間では一点で交差する．これらの交点は，パラメータ空間を適切に量子化した蓄積配列において数え上げられる．以下では，この数え上げを**投票**と呼ぶ．つまり，二次元画像中の各エッジ点 (x, y) に対して，曲線 $\rho = x\cos\theta + y\sin\theta\ (0 \le \theta < 180)$ に沿って投票を実行する．検出される可能性のある直線は何度も投票される点を探すことで検出することができる．図 2にハフ変換を用いた直線検出の例を示す．



(a) 入力画像　　(b) エッジ画像　　(c) 直線検出結果

図 2: 直線のハフ変換の例

入力画像である図 2(a)に対して，ソーベルフィルタなどのエッジ検出器によって得られた二値画像が図 2(b)である．そのエッジ画像に対して投票操作を行い，その結果得られた直線が図 2(c)である．

3-2 直線ハフ変換アルゴリズム

以下では，$n \times n$ の入力画像に対して，直線のハフ変換アルゴリズムを説明する．図 3に示すように，入力画像の中心が原点になるように２次元配列に格納されているとする．よって，座標 x と y はともに区間$[-n/2+1, n/2]$の整数値をとる．xy 空間の点 (x, y) $(-n/2+1 \le x, y \le n/2)$ は，次の式によって $\rho\theta$ 空間における曲線に変換される．

$$\rho = x\cos\theta + y\sin\theta\left(0 \le \theta < 180\right) \tag{1}$$

式より，$-2/\sqrt{n} < \rho \le 2/\sqrt{n}$ を満たし，ρとθ は幾何的に求めることができる．

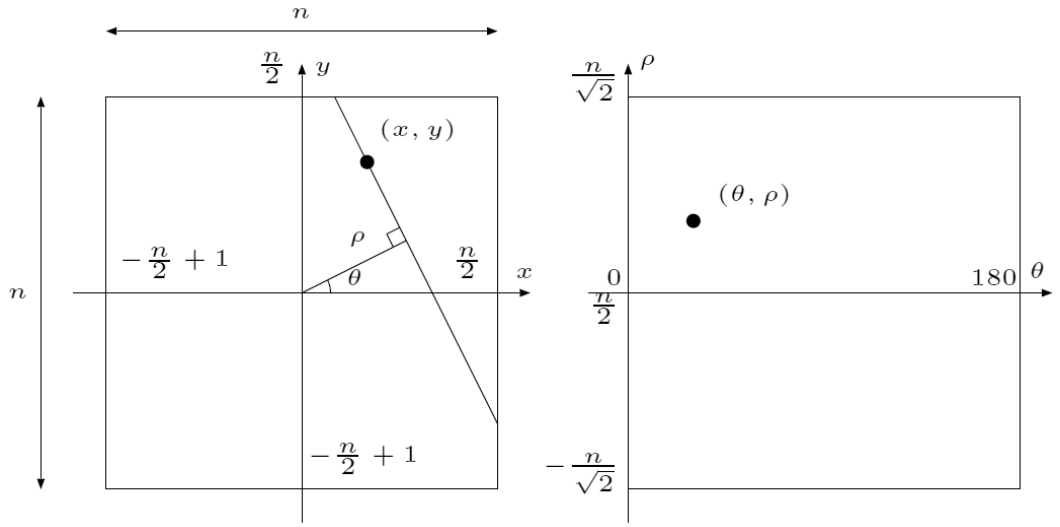


図 3: 2 次元空間 *xy* と *θρ*

図 3のように，原点を通り，x 軸と成す角がθ である直線を考える．そのような直線に対して，点(x, y)を通る垂線をひくことができ，ρ は原点からその直線までの距離に対応する．つまり，$\theta\rho$ 空間の点(θ, ρ)は，xy 空間の直線に対応する．ハフ変換の主要なアイデアは，　xy空間のすべてのピクセルに対して，$\theta\rho$ 空間で投票することである．xy空間の k 個の点$(x_0, y_0), (x_1, y_1), \ldots, (x_{k-1}, y_{k-1})$に対してハフ変換は次のように表される．

[Straight Forward Hough Transform]

for $i \leftarrow 0$ to $k - 1$
　for $\theta \leftarrow 0$ to 179
　　begin
　　$\rho \leftarrow x_k \cos\theta + y_k \sin\theta$
　　$v[\theta][\rho] \leftarrow v[\theta][\rho] + 1$
　　end
for $\theta \leftarrow 0$ to 179 do in parallel
　for $\rho \leftarrow -n/\sqrt{2}$ to $n/\sqrt{2}$ do in parallel
　　output (θ, ρ) if $v[\theta][\rho] \geq threshold$

簡単のために，ρ は自動的に整数に丸めるとする．上記 Straight Forward Hough Transformでは，各点(x_k, y_k)に対して，$x_k \cos\theta$ と$y_k \sin\theta$ の値を$\theta = 0, 1, \ldots, 179$ に対して求める．$v[\theta][\rho]$が大きい値をとる場合，多くの点が$\theta\rho$ 空間の点(θ, ρ)に対応するxy 空間の直線上に存在する．三角関数の加法定理より，式(1)は次のように変形することができる．

$$\begin{aligned}\rho &= x_k \cos(180-\theta) + y_k \sin(180-\theta) \\ &= -x_k \cos(\theta) + y_k \sin(\theta)\end{aligned} \tag{2}$$

式(2)を用いることで，ハフ変換はθ の範囲[0, 179]を二つの範囲[0, 89]と [90, 179]に分割することができる．また，閾値$threshold$より大きな要素を探索するために配列v の走査を回避することができる．以上より，新しいハフ変換アルゴリズムCircuit-oriented Hough Transformを次に示す．

[Circuit-oriented Hough Transform]

for $i \leftarrow 0$ to $k - 1$ do
　begin
　　for $\theta \leftarrow 0$ to 89 do
　　begin
　　$\rho \leftarrow xk \cos\theta + yk \sin\theta$
　　$v[\theta][\rho] \leftarrow v[\theta][\rho] + 1$
　　output (θ, ρ) if $v[\theta][\rho] = threshold$
　　end
　　for $\theta \leftarrow 1$ to 90 do
　　begin
　　$\rho \leftarrow -x \cos(\theta) + y \sin(\theta)$
　　$v[180 - \theta][\rho] \leftarrow v[180 - \theta][\rho] + 1$
　　output (θ, ρ) if $v[\theta][\rho] = threshold$
　　end
　end

3 二値画像に対するハフ変換を用いた直線検出回路

本節では，Xilinx社のVirtex-6 FPGA (XC6VLX240T-1[3])に対するFPGAアーキテクチャを説明する．

3-1 ハフ変換回路のアーキテクチャ

図 4にハフ変換アーキテクチャを示す．アーキテクチャでは，178個のDSPB $X_1, X_2, ..., X_{89}$と$Y_1, Y_2, ..., Y_{89}$を用いる．各$\theta(0 \le \theta \le 90)$に対して，$X_\theta$ とY_θ は与えられたx_kとy_kに対して，$x_k \cos\theta$と$y_k \sin\theta$ を計算する．

$x_k \cos 0 = x_k, x_k \cos 90 = 0, y_k \sin 0 = 0, y_k \sin 90 = y_k$ より，DSPB X_0, X_{90}, Y_0, Y_{90}は必要ないことがわかる．

加算器と減算器をX_θとY_θに対して用いることで，$\rho_\theta = x_k \cos\theta + y_k \sin\theta$ と $\rho_{180-\theta} = -x_k \cos\theta + y_k \cos\theta$ が求められる．また，投票値を格納するために，180個のBRAM $V_0, V_1, \ldots V_{179}$を用いる．各V_θ のアドレスρは，$v[\theta][\rho]$の値を格納するのに使用する．レジスタ間の遅延を最小化するために，DSPB $X_1, ..., X_{90}$は図 4に示したように，パイプライン化して接続される．各X_θ はx_k の値を格納するためにレジスタを持つ．クロックサイクル毎に，X_θ から$X_{\theta+1}$ に送られる．同様に，DSPB $Y_0, Y_1, ..., Y_{90}$ はパイプライン接続される．図 5にρを計算するための加算器と減算器を用いたDSPB X_θとY_θ を図示する．

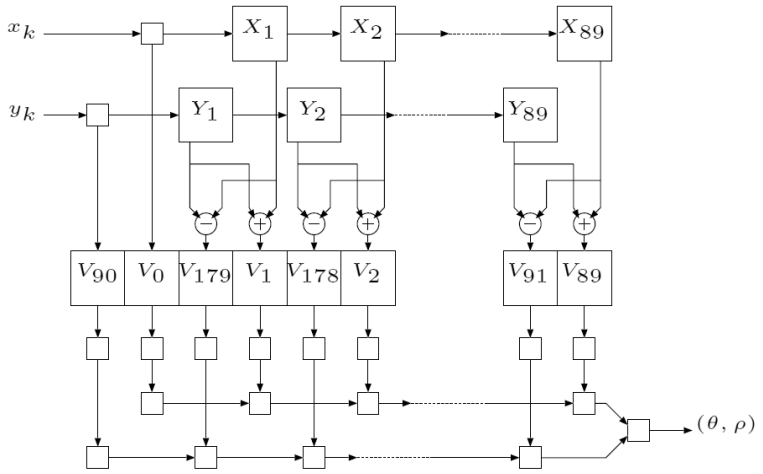


図 4：ハフ変換回路の構成

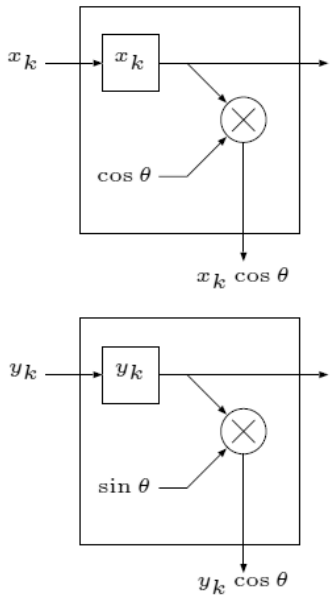


図 5: DSPB Xθ と Yθ の構成

DSPB X_θ では，x_k の値を内部レジスタにロードされ， cosθ の値を前計算した値が入力される．このとき，cosθ の値は固定値であり，x_k とcosθ の積をDSPB X_θ の乗算器で計算する．同様に，DSPB Y_θ では，y_k の値を内部レジスタにロードされ， sin θ の値を前計算した値が入力される．このとき，sin θ の値は固定値であり， y_k とsinθ の積をDSPB Y_θ の乗算器で計算する．今回のターゲットデバイスであるVirtex-6 FPGA XC6VLX240Tには，96の隣接するDSP48E1 blockが8列配置されている．隣接するDSP48E1 blockはパイプラインレジスタを介して直接接続されている．提案ハフ変換アーキテクチャは，$x_k \cos\theta$と$yk \sin\theta$を計算するために2列のDSP48E1 blockを利用し，動作周波数を最大化するために，パイプライン手法を用いる(図 6)．

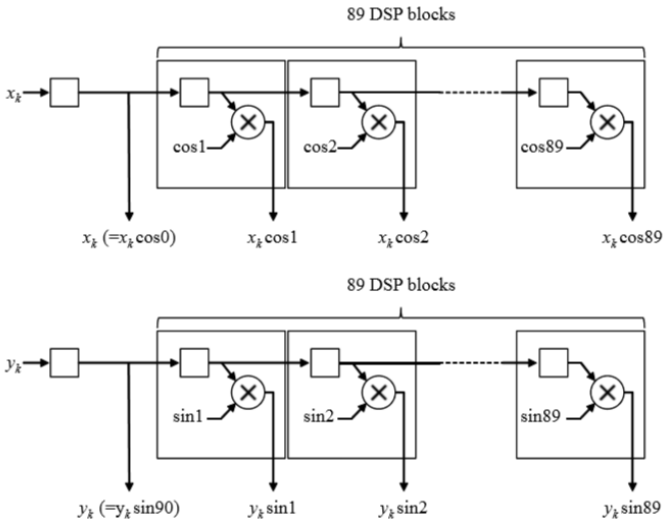


図 6: DSPB を用いたパイプラインアーキテクチャ

図 7はBRAMを用いたV_θのアーキテクチャを図示したものである．FPGAに搭載されたBRAMはデュアルポートの構造を持ち，Xilinx Virtex-6シリーズのFPGAには，独立した操作が可能なポートセットを2つ持つ18kビットのRAMがある． ２つのポートセットは，

ポートセット A *ADDRA* (ADDRess A)，*DOA* (Data Output A)，*DIA* (Data Input A)，

ポートセット B *ADDRB* (ADDRess B), *DOB* (Data Output B), *DIB* (Data Input B)
で構成される.
BRAMのアドレスi のデータを $M[i]$ とすると，Port Set Aの読み出し操作では，$M[ADDR]$ がクロックの立ち上がりの後，*DOA*から出力される．書き込み操作では，*DIA*に与えられたデータがクロックの立ち上がりで$M[ADDRA]$に書き込まれる．読み出しと書き込み操作はRF (Read First)モードかWF(Write First)モードのどちらかに設定できる.RFモードでは,読み出しと書き込み操作が同一のアドレスに対して実行されたとき，書き込み操作が実行される前に読み出し操作が実行される．つまり，読みだされるデータは書き込まれるデータの前のデータになる．逆に，WFモードでは，読み出しと書き込み操作が同一のアドレスに対して実行されたとき，読み出し操作より前に書き込み操作が実行されるため，読みだされるデータは更新されたデータになる．しかし，デュアルポートを利用する際に，読み出しと書き込み操作を同じアドレスに対して実行するなら，BRAMの設定はRFモードにすべきである[4].

提案回路では，$v[\theta][\rho]\left(-n/\sqrt{2} < \rho \le n/\sqrt{2}\right)$の値を格納するためにBRAMを使用する．BRAM v_θ においてアドレスiのデータを$v_\theta[i]$と表す．*ADDRA*にρが入力されるため，図 7に示したように，クロックの立ち上がりの後，*DOA*から$v_\theta[\rho]$が出力される．その後，$v_\theta[\rho]$+1を求め，その値を*DOB*に入力する．ρは*ADDB* に入力され，$v_\theta[\rho]$に$v_\theta[\rho]+1$ が書き込まれる．つまり，$v_\theta[\rho] \leftarrow v_\theta[\rho]$+1を実行したことになる．その時，上記で述べた制約によって，ρが同じ値で連続するかもしれないから，BRAMはRFモードに設定しなければならない．つまり，連続して同じρの値が入力されるとき，直前の投票値はBRAMから読み出すことができない．このような状況を避けるため，最後の投票値を格納するために追加のレジスタを利用し，ρの値が同じときは，BRAMから読み出した値の代わりに，そのレジスタに格納された値を用いる．それと同時に，比較器を使って，$v_\theta[\rho]+1 = threshold$かどうかを求め，もしそうなら，$\rho$の値を別のレジスタに格納しておく．その後，$(\theta, \rho)$の組を検出する可能性がある直線として，続くレジスタに書き込む． その値は，図 4に示したシフトレジスタ列を用いて回路の出力へクロック毎に送られていく．データが出力されるまで移動するクロックサイクル数を減らすために，２列のシフトレジスタを用いる．一つは，$V_0, \ldots, V_{89}$からのデータを出力するために利用し，もう一つは$V_{90}, \ldots, V_{179}$からのデータを出力するために使用する．よって，データの移動に必要なクロックサイクル数は高々90クロックサイクルに減少する．

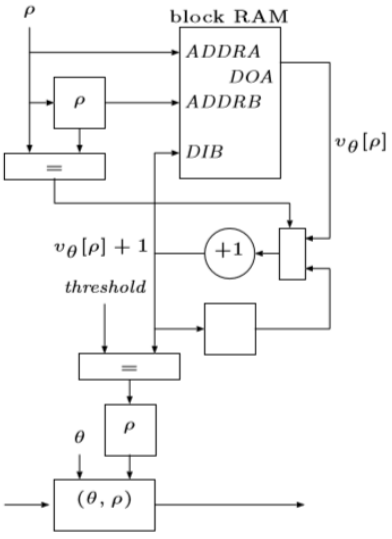


図 7: BRAM を用いた V θ のブロック図

3-2 データ表現

データの精度は，回路面積，設計の単純化，スピード，電力消費といった実装コストによって決まる．高精度にすることで最終的な実装の誤差は小さくなるが，逆に精度を下げることで，低消費電力でかつコンパクトで高速な設計がもたらされる．このトレードオフはアプリケーションや利用可能なFPGAのリソースに依

存する．そのため，本研究では，回路面積と計算時間を最小化するために，短い固定小数点表現を使用する．

DSPBやBRAMの構造を考慮することによって，我々の提案回路は次のよう実装にした．座標x_kとy_kの入力のデータフォーマットはそれぞれ10ビットの２の補数表現の整数．また，$\cos\theta$ と$\sin\theta$は16ビットの固定小数点表現の数で，符号ビット，1ビットの整数部，14ビットの小数部で構成される．一方，ρのデータフォーマットは，10ビットの２の補数表現の整数で表す．投票値のデータフォーマットは，18ビットの整数，つまり，最大$2^{18}-1$まで投票値を格納できる．θの範囲は0から180なので，θのデータフォーマットは8ビットの整数で表す．

3-4 性能評価

提案ハフ変換アーキテクチャを Xilinx 社の Virtex-6 FPGA XC6VLX240T-1 に実装した．表 1 に Xilinx 社の ISE 13.1 を用いた実装結果を示す．実装では，回路遅延を減らすため，回路の要素間にパイプラインレジスタを挿入している．実行時間は，入力された x_k と y_k に対して ρ を計算するのに 3 クロックサイクルかかる．また直線の候補として表される組(θ, ρ)を出力するのに 4 クロックサイクル必要である．さらに，最大 90 クロックサイクルかけてデータを出力に移動する．よって，この回路は，エッジ点の数が m のとき，m +97 クロックサイクル，つまり，$(m+97)/245.519\,\mu s$ かけて(θ, ρ)で表現された直線を検出し，それらを出力する．

例えば，図 2(b)の 33232 のエッジ点を含む画像に対して，この回路は 135.75μs でハフ変換を実行できる．計算時間について最悪になる入力画像は画像中のすべての点がエッジ点の場合である．例えば 512×512(=262144)の画像のすべてがエッジ点のとき，1068.11μs で結果の出力を完了する．すべての点がエッジ点の画像は当然存在しないが，これは 512×512 の画像に対してハフ変換を 1068.11μs より短い時間で実行できることを示す．

表 1: ハフ変換回路の性能評価

DSP48E1 blocks (out of 768)	178 (23.1%)
18kbit block RAMs (out of 832)	180 (21.6%)
Slices (out of 301440)	14493 (4.81%)
Clock frequency [MHz]	245.519

提案回路の FPGA 実装の高速化を見積もるために，ハフ変換の逐次処理ソフトウェアを GNU C を用いて実装した．実行時間の評価には，Intel 社の Xeon X7460（2.66GHz）の CPU と 128GB の主記憶をもつ PC を使用した．エッジ点を 33232 点もつ図 2(b)の画像に対してソフトウェア実装は 413.98ms でハフ変換を実行した．512×512(=262144)のすべてがエッジ点の画像に対しては，3266.75ms で結果の出力を完了した．よって，提案するハフ変換回路の FPGA 実装は，CPU 上の逐次処理に対して約 3000 倍の高速化を実現した．

4 画像の勾配を考慮したハフ変換を用いた直線検出回路

4-1 勾配を考慮した直線のハフ変換アルゴリズム

本節では，画像の勾配を考慮したハフ変換[5]の FPGA 実装を提案する．勾配を考慮したハフ変換では，直線検出を二値画像ではなく，グレイスケール画像に対して実行する．グレイスケール画像から勾配情報を求めるために，本研究ではソーベルフィルタを用いる．ソーベルフィルタは２つの 3×3 畳み込み G_x と G_y を使って水平及び垂直方向の偏差の近似を画像に対して求める．

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \otimes I, G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \otimes I \qquad (3)$$

ことのき，I は入力画像，⊗ は 2 次元畳み込み操作を示す．G_x と G_y はそれぞれ水平方向と垂直方向の近似で，各ピクセルで次の式により勾配強度を求めることができる．

$$G = \sqrt{G_x{}^2 + G_y{}^2} \tag{4}$$

同様に，次に式を用いて勾配方向 θ' を求めることができる．

$$\theta' = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{5}$$

上記で得られた勾配情報に基づき，投票空間に投票する．しかし，局所的な勾配情報と実際の直線の方向には量子化誤差が存在する．そのため，勾配方向によって得られる角度だけでなく，その角度の近傍に対しても投票操作を行う．提案手法では，以下に示す角度に依存する重み w を導入する．

$$w(\theta - \nu') = \begin{cases} 2^{\lambda - |\theta - \theta'|} & |\theta - \theta'| \le \lambda \\ 0 & otherwise \end{cases} \tag{6}$$

コンパクトな FPGA 実装にするため，重みを 2 のべき乗数として利用する．また，投票範囲を[0, 179]の代わりに[−λ,+λ]に制限する．勾配を考慮したハフ変換のアルゴリズムを次に示す．

[Gradient-based Hough Transform]

for $y \leftarrow -n/2+1$ to $n/2$

  for $x \leftarrow -n/2+1$ to $n/2$

    Compute G and θ' for $p[x][y]$

    for $\theta \leftarrow \theta' - \lambda$ to $\theta' + \lambda$ do

      begin

        if $\theta < 0$ then $\theta \leftarrow \theta + 180$

        $\rho \leftarrow x\cos\theta + y\sin\theta$

        $v[\theta][\rho] \leftarrow v[\theta][\rho] + G \cdot w(\theta - \theta')$

      end

上記アルゴリズムでは，各ピクセルに対して，勾配強度に比例した重み $G \cdot w(\theta - \theta')$を投票する．パラメータ空間は通常の直線のハフ変換に対して投票が集中しやすくなり，精度も向上する．

4-2　勾配を考慮した直線のハフ変回路の FPGA 実装

ここでは，DSPB と BRAM を用いた勾配を考慮した直線のハフ変回路の FPGA 実装を提案する．図 8はアーキテクチャ全体の構成図である．以下では，各回路の詳細を説明する．

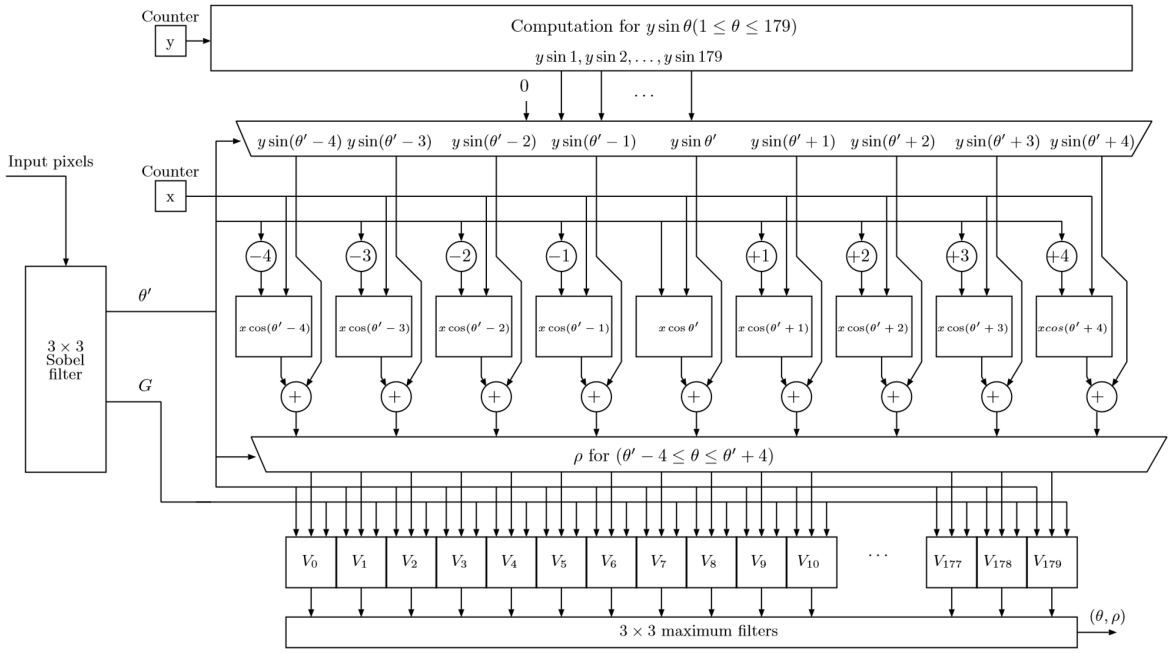
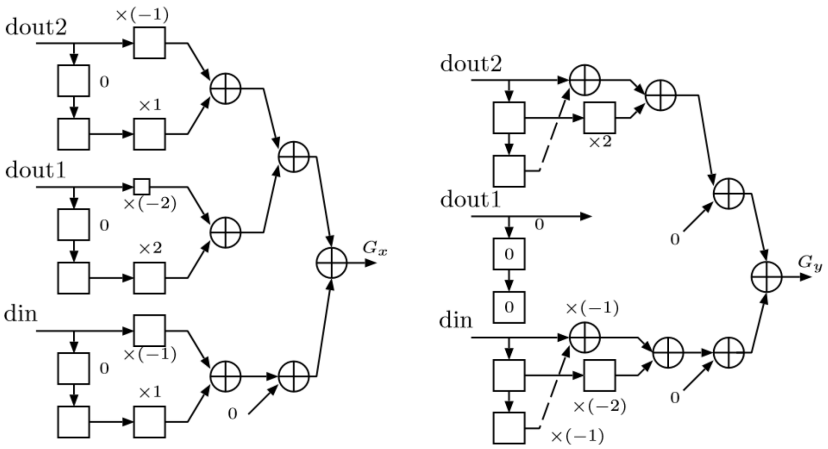


図 8：勾配を考慮した直線のハフ変回路の全体図

（１）勾配情報計算回路

提案アーキテクチャでは，勾配情報を求めるために，3×3 のソーベルフィルタを使用する．本回路では，入力画像の画素値がラスタスキャン順 1 ピクセルずつ与えられるとする．そのため，3×3 の部分画像をフィルタに入力するために，２行分のラインバッファを利用する．回路では，図 9に示す組合せ回路を用いて水平方向と垂直方向の偏差の近似を計算する．



(a) Gx　　(b) Gy

図 9：ソーベルフィルタ計算回路

上記で説明したように，アルゴリズムでは，勾配強度と勾配方向が必要である．そのため，平方根と逆正接の計算が必要になる．回路で直接それらの計算をするのは容易でないため，本研究では，Xilinx 社が提供する CORDIC IP[6]を使用する．CORDIC IP は完全パイプラインする回路として FPGA 上に簡単に利用できるハードウェアモジュールである．

（２）投票操作回路

次に，上記パイプライン回路を用いてピクセル毎に求めた勾配方向 θ' と勾配強度 G から ρ を計算し，

投票操作を行う回路を説明する．回路では，勾配方向 θ'と勾配強度 G が入力される度にその座標 x, y を 2 つのカウンタを用いて計算する．DSPB と BRAM を用いて $x\cos(\theta'-\lambda),\ldots,x\cos(\theta'+\lambda)$を計算する回路を図 10に示す．この回路は DSPB 一つと BRAM 一つで構成しており，BRAM をルックアップテーブルとして使い $\cos\theta$ を計算し，DSPB を用いて x と $\cos\theta$ の積を計算する．このとき，BRAM のデュアルポートを利用することで，2 つの回路でルックアップテーブルを共有することができる．よって，$x\cos(\theta'-\lambda),\ldots,x\cos(\theta'+\lambda)$の計算に $2\lambda+1$ 個の DSPB と $\left\lceil \frac{2\lambda+1}{2} \right\rceil$ 個の BRAM を使用する．

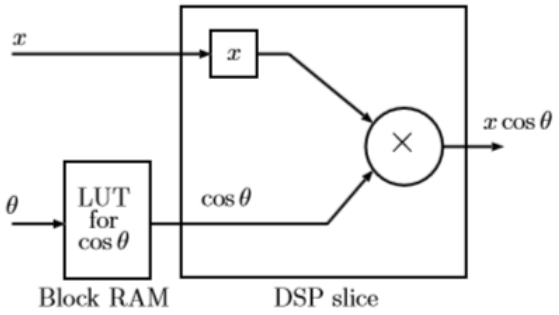


図 10：DSPB と BRAM を用いた $x\cos\theta$ 計算回路

また，$y\sin\theta$ $(1 \le \theta \le 179)$を計算するために，ラスタスキャン順にピクセル毎の値が入力されるからある行のピクセルを処理している間は y の値が変化しないことを利用する．従って，y 行目の処理をしているときに, 次の行, つまり y+1 行目の計算$(y+1)\sin\theta$ $(1 \le \theta \le 179)$を前もって行い, その値をレジスタに格納しておく．このとき，$\sin\theta$ を計算するために BRAM をルックアップテーブルとして利用し，DSPB を使って y と $\sin\theta$ の積を計算する．さらに, 計算結果を格納するためのレジスタを 2 組用意して, 行ごとに切り替えて使用する．図 11は $y_{k+1}\ sin\theta$ を計算する回路の構成図である．

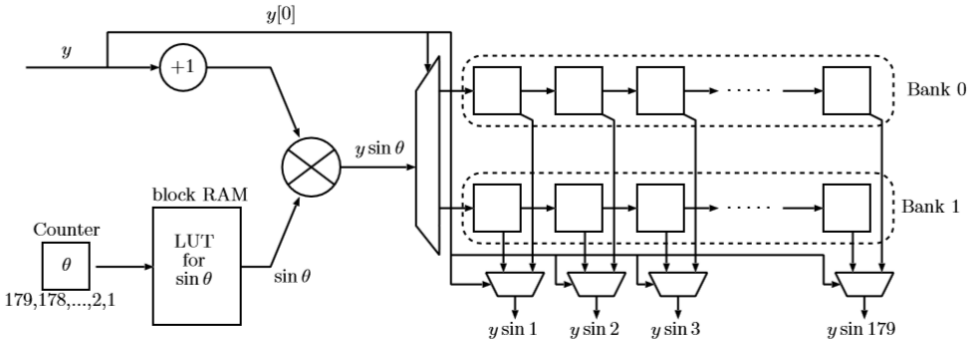


図 11：DSPB と BRAM を用いた yk+1 sin θ を計算回路

BRAM を用いた投票回路を図 12に示す．基本的な構成は第 3 節の投票回路と同じで，式 6 にある$|\theta-\theta'|$を計算する減算器とべき乗倍を計算するビットシフト回路が追加されている．

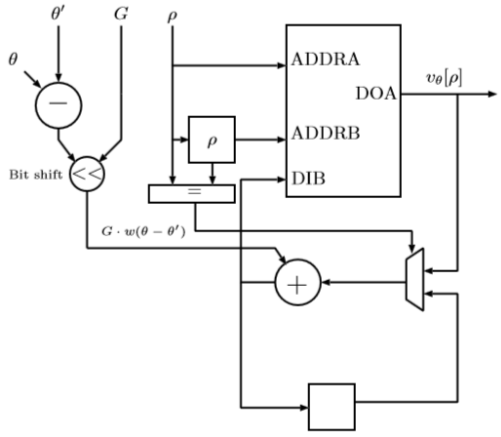


図 12：BRAM を用いた投票回路

次に，投票操作が完了した際の最終的な直線の検出結果を出力するための極大値フィルタについて説明する．極大値フィルタは，画像の局所領域の中での最大値として定義される．ここでは，投票空間の各投票値について，このフィルタは，注目している値が 3×3 の領域で最も大きい値のとき，その値をコピーする．図13に投票結果に対する 3×3 の極大値フィルタを計算する回路を示す．基本的な動作は，各 θ の値で並列にパイプライン処理をして極大値を求めていく．得られた極大値がある閾値よりも大きいときに，(θ,ρ) で表現された直線としてシフトレジスタに入力され，順に出力に送られていく．

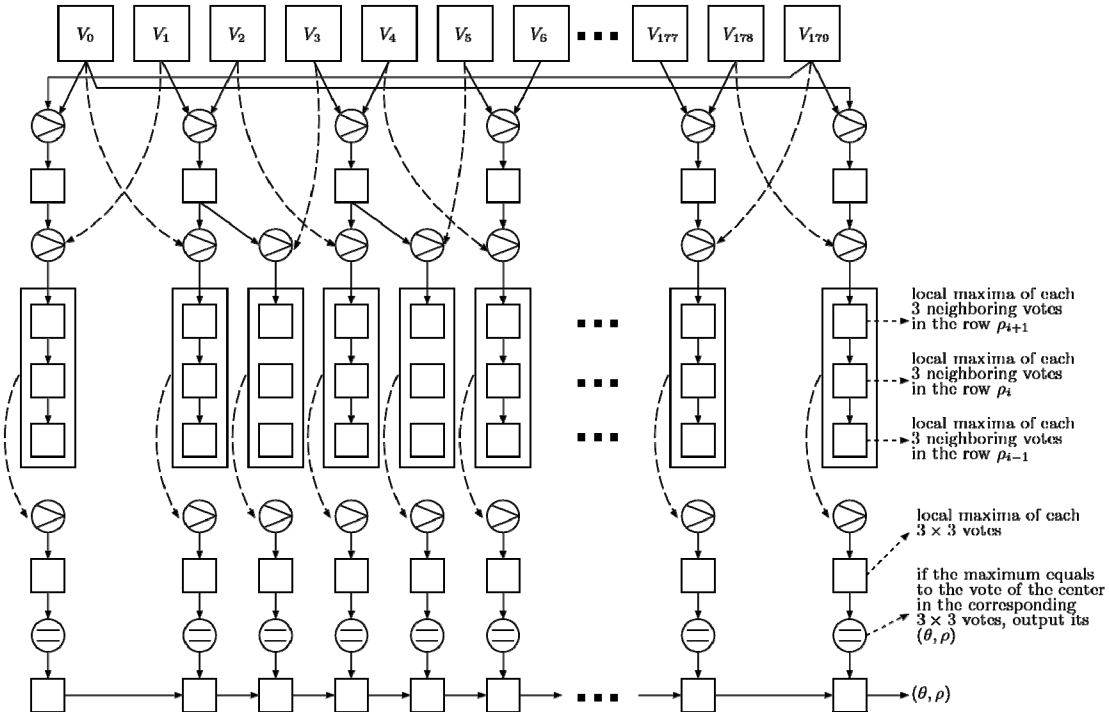


図 13：パイプラインを用いた極大値フィルタ回路

（３）データ表現

回路中のデータの精度は第 3 節のそれとほぼ同じで，**座標 x と y の入力のデータフォーマットはそれぞれ 10 ビットの２の補数表現の整数．また**，$\cos\theta$ と $\sin\theta$ は 16 ビットの固定小数点表現の数で，符号ビット，1 ビットの整数部，14 ビットの小数部で構成される．一方，ρ のデータフォーマットは，10 ビットの２の補数表現の整数で表す．投票値のデータフォーマットは，18 ビットの整数，つまり，最大 2^{18}-1 まで投票値を格納できる．θ の範囲は 0 から 180 なので，θ のデータフォーマットは 8 ビットの整数で表す．

4-2 性能評価

提案回路を Xilinx 社の Virtex-7 FPGA XC7VX485T-2 に実装した．表 2 に Xilinx 社の ISE 14.1 を用いた実装結果を示す．実装では，式 6 の λ=4，つまり，$\theta'-4 \le \theta \le \theta'+4$ に対して投票する設定とした．この範囲は実験から得た値である．本実装は，$n \times n$ のグレイスケール画像に対して，パイプライン動作で処理することができる．つまり，入力ピクセルは 1 クロックサイクルごとにラスタスキャン順に回路へ入力することができる．実装では，回路遅延を減らすため，回路の要素間にパイプラインレジスタを挿入している．実行時間は，ピクセル値が入力されてから $2n$+44 クロックサイクル後にそのピクセルの投票操作が完了する．入力ピクセルの数は n^2 だから，$n^2+2n+44$ クロックサイクル，つまり，$\frac{n^2+2n+44}{260.061}\mu s$ 必要である．投票操作の後，3×3 の極大値フィルタを求めるのに $\sqrt{2}n+188$ クロックサイクル，つまり，$\frac{\sqrt{2}n+188}{260.061}\mu s$ 必要である．以上より，全体で $n^2+\left(\sqrt{2}+2\right)n+232$ クロックサイクル，つまり，$\frac{n^2+\left(\sqrt{2}+2\right)n+232}{260.061}\mu s$ で直線検出を実行することができる．入力画像が 1000×1000 のとき，提案回路は 3.859ms で直線を検出することができる．

提案回路の FPGA 実装の高速化を見積もるために，勾配を考慮したハフ変換の逐次処理ソフトウェアを GNU C を用いて実装した．実行時間の評価には，Intel 社の Xeon X7460（2.66GHz）の CPU と 128GB の主記憶をもつ PC を使用した．サイズが 333×333 のグレイスケール画像に対してソフトウェア実装は 133.519ms で勾配を考慮したハフ変換を実行した．一方，提案回路では，431.660μs で処理可能である．よって，提案回路の FPGA 実装は，CPU 上の逐次処理に対して約 309 倍の高速化を実現した．

5 結び

本研究では，FPGA に搭載されている DSPB と BRAM を効果的に利用することにより高速な画像処理システムが構築できることを確認した．特に，列状に配置されている DSPB を考慮したアーキテクチャにすることで大幅な性能向上ができることを確認した．画像処理の例として，ハフ変換による直線検出を FPGA に実装することで CPU を用いた逐次処理と比べて最大で約 3000 倍の高速化を実現した．

【参考文献】

[1] P.V.C. Hough, “Method and means for recognizing complex patterns,” U.S. Patent 3,069,654, 1962.

[2] R.O. Duda and P.E. Hart, “Use of the Hough transformation to detect lines and curves in pictures,” Communications of the ACM, vol.15, no.1, pp.11–15, 1972.

[3] Xilinx Inc., “Virtex-6 family overview v2.4,” 2012.

[4] Xilinx Inc., “Virtex-6 FPGA memory resources user guide v1.6,” 2011.

[5] F. O’Gorman and M. Clowes, “Finding picture edges through collinearity of feature points,” Computers, IEEE Transactions on, vol. C-25, no. 4, pp. 449–456, 1976.

[6] Xilinx Inc., “LogiCORE IP CORDIC v6.0,” 2013.

〈発　表　資　料〉

題　　名	掲載誌・学会名等	発表年月