コンピュータビジョン　29－1
（１９８４．　３．１２）

画像処理用ＬＳＩ－ＩＳＰのアーキテクチャ（その２）

福島　忠†　小林芳樹†　平沢宏太郎†　坂東忠秋†
柏岡誠治‡　加藤　猛‡

†（株）日立製作所日立研究所　‡同社中央研究所　‡同社大みか工場

Architecture of Image Signal Processor (2)

Tadashi Fukushima†, Yoshiki Kobayashi†, Kohtaroh Hirasawa†,
Tadaaki Bandoh†, Seiji Kashioka‡, and Takeshi Katoh‡

† Hitachi Research Laboratory, Hitachi Ltd.
‡ Central Research Laboratory, Hitachi Ltd.
‡ Omika Works, Hitachi Ltd.

Abstract　The architecture of a dedicated LSI, an Image Signal Processor (ISP), is presented which performs most local operations for eight-bit gray images. The ISP can effectively carry out multi-mask processing such as the template-type Prewitt operator which uses eight template masks. The device is designed fit for two kernel expansion methods: PE(Processor Element)-increase and PE-save methods. In the case of performing the template-type Prewitt operator onto a 256 x 256 pixel-size image, three ISPs require 87.4 ms with the PE-increase method and one ISP 262.1 ms with the PE-save method.

１．はじめに

濃淡画像処理におけるアルゴリズムの研究は、古くからなされているが、その実用化はそれ程進展してはいない。そこで、一般産業への適用を目的として、各種の濃淡画像演算を高速に実行する多機能画像処理ＬＳＩ-ISP（Image Signal Processor）を開発した[1]。

ＩＳＰは、出力画像の一つの画素（pixel）を算出するのに用いる入力画像の画素と、同数のPE（Processor Element）を用意して、局所的に並列演算する局所並列型に属し、次の特長を備えている。まず、空間積和演算に代表される局所近傍演算を、ビデオレートで高速処理できる。また、プログラマブル制御レジスタを内蔵し、２値・濃淡・色彩画像の基本演算をほとんど実行できる。さらに、局所演算領域（kernel－以下カーネルと呼ぶ）を容易に拡張できる。

これらの特長を実現するＩＳＰのアーキテクチャについては、すでに報告したが[2][3]、そこでは、空間積和演算やパターンマッチングなどのマスク演算に対しては、１個のマスクデータしか用いられないことが前提となっている。しかし、２個以上のマスクデータを用いるアルゴリズムも、これまでに数多く提案され、その有用性も実証されている。たとえば、８個の差分型マスクを用いて、その最大出力値からエッジの強度や方向を求めるものとして、Prewitt[4]、Kirsch[5]、Robinson[6]、などのテンプレート型オペレータがある[7]。また、半導体チップの自動ワイヤボンディング装置では、複数のテンプレートを用いたパターンマッチングにより、ある程度の回転ずれを許容する位置検出を実現している[8]。そこで、ＩＳＰにおいては、これまでに論じたアーキテクチャを基礎として、複数のマスクデータを用いる演算（Multi-mask Processing－以下マルチマスクプロセッシングと呼ぶ）を、効率良く実行できるようアーキテクチャを決定した。

本論文では、ＩＳＰの基本アーキテクチャを概説した上で、カーネルの拡張性を損なわずに、マルチマスクプロセッシングを効率良く実行できる、ＩＳＰのアーキテクチャについて論ずる。

２．ＩＳＰの基本アーキテクチャ

ＩＳＰの基本構成図と基本仕様を、それぞれ図１と表１に示す。図１に示すように、ＩＳＰは６個のユニットから構成されている。データユニットは、４個のシフトレジ

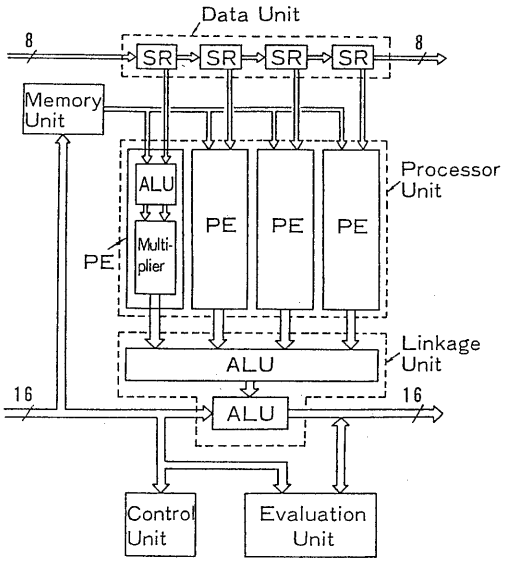

図１　ＩＳＰの基本構成図

スタなどから成り、画像データの転送に寄与する。入出力バスは８ビット幅で、256階調の濃淡データもしくは8個の2値データを扱える。メモリユニットは、8bit × 64 word から成るRAM(Random Access Memory)で構成され、荷重係数やテンプレートなどのマスクデータを記憶する。　画像演算時は、通常各PEに1バイトずつデータを供給する。プロセッサユニットは、4個のPEから構成され、各PEはSIMD(Single Instruction Multi-Data stream)の形態で画像データを並列処理し、それぞれ16ビットの演算結果を、リンケージユニットへ出力する。リンケージユニットは、2個のALU(Arithmetic Logic Unit)で構成され、PE間とLSI間の統合演算に寄与する。入出力データはすべて16ビットである。エバリュエーションユニットは、2個の比較器などから成り、リンケージユニットの出力データに対して、2値化やクラスタリングを行う。コントロールユニットは、プログラマブル制御レジスタなどから構成され、各ユニットを制御する。

ＩＳＰの最大の特長は、二つの方式によりカーネルを拡張できることである。一つは、カーネルの大きさに合わせて使用するPE数を増やす方式―PE増殖方式―であり、もう一つは、PEを時分割に使用することにより、カーネルを形成する画素数より少ないPEで処理する方式―PE節約方式―である。PE増殖方式において処理される画像データは、通常のテレビ画像の走査方式であるラスタ走査(Raster Scanning)により走査されればよい。一方、PE節約方式においては、画像データは、筆者らが提案したスティック走査(Stick Scanning)により走査されなければならない[(2)]。

表１　ＩＳＰの基本仕様

Technology	3 μm CMOS
Number of transistors	～61,000
Chip size	7.72×8.64 mm
Power suppy	5 V
Execution cycle time	167 ns
Power dissipation	～400 mW
Package	64 pin DIP

スティック走査とは、三つの走査方向から成る画像走査方式で、主走査方向は上から下、副走査方向は左から右、そして副々走査方向は上から下である。主走査方向において走査される画素の集合は、スティック(Stick)と呼ばれ、一つのスティックに含まれる画素数は、スティック長(Stick Length)と定義される。スティック長が１のスティック走査が、ラスタ走査となる。図2の小画像を例にとると、スティック長が４のスティック走査の場合、各画素は次の順に走査される。

①, ⑪, ㉑, ㉛, ②, ⑫, ㉒, ㉜, ③, ---→,
⑩, ⑳, ㉚, ㊵, ⑪, ㉑, ㉛, ㊶, ⑫, ㉒,
㉜, ㊷, ⑬, ---→, ㊿, (60), (70), (80), (90), (100).

PE節約方式によりカーネルを拡張して、カーネルが４×４の空間積和演算を実行する場合の、ＩＳＰのアーキテクチャを図3に示す。ここでは、各シフトレジスタの遅延段数は、スティック長に等しい4段となっている。

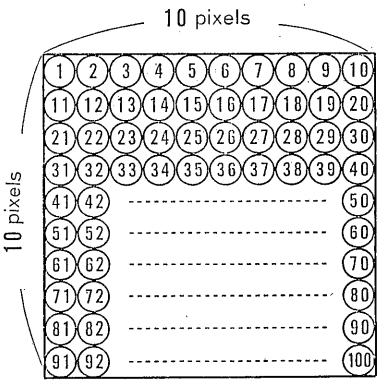

図２　10×10画像から成る画像

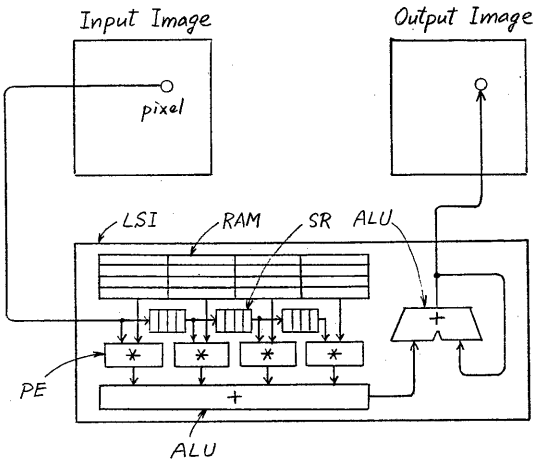

図３　ＰＥ節約方式における４×４空間積和演算実行のための構成

またRAMは、4x4個の荷重係数を記憶するため、16バイト必要になる。さらに、4マシンサイクルに分割されて算出される部分積和値を累算するために、リンケージユニットのALUは、アキュームレータとして動作する。

３．マルチマスクプロセッシング

ある原画像からカーネルを形成する画素を切り出して、局所近傍演算を施す場合、通常、マスクデータとのマスク演算となる。たとえば空間積和演算では、マスクデータとして荷重係数が用意され、画素データは、荷重係数と掛け合わされた後加算される。この時、一つの画像処理で用いられるマスクデータが、平滑化オペレータやラプラシアンオペレータなどの単一のものと、Prewittオペレータのテンプレート型のような複数のものとに分けられる。ここでは、後者を前者と区別して、マルチマスクプロセッシングと呼んでいる。代表的なマルチマスクプロセッシングを表２に示す。

表2の中で、Prewitt、Kirsch、Robinsonのテンプレート型オペレータはよく知られている。これらは、3×3のカーネルに8個の差分型マスクを掛け合わせ、その最大値をエッジの強度として、また、その時のマスクがMiならば、iをエッジの方向として出力するものである。

このようなマルチマスクプロセッシングの処理方式は、次の二つの観点から分類できる。一つは、全画面単位でマスク演算を繰り返すのか、それともカーネルを切り出

表２　代表的なマルチマスクプロセッシング

入力画像	名　称	内　容	マスクデータ
２値	複合パターンマッチング	複数個の標準パターンと照合して、最もよく一致した標準パターンと、その時の一致度を求める。	任　意
濃淡	Prewitt オペレータ（テンプレート型）	M0～M7のマスクを3×3のカーネルに掛け合わせ、その最大値をエッジ強度とし、その時のマスクがMiならば、エッジ方向をiとする。	M0 M1 M2 M3 M4 M5 M6 M7 1 1 1 / 1 -2 1 / -1 -1 -1 1 1 1 / 1 -2 -1 / 1 -1 -1 1 1 -1 / 1 -2 -1 / 1 1 -1 1 -1 -1 / 1 -2 -1 / 1 1 1 -1 -1 -1 / 1 -2 1 / 1 1 1 -1 -1 1 / -1 -2 1 / 1 1 1 -1 1 1 / -1 -2 1 / -1 1 1 1 1 1 / -1 -2 1 / -1 -1 1
	Kirsch オペレータ		5 5 5 / -3 0 -3 / -3 -3 -3 5 5 -3 / 5 0 -3 / -3 -3 -3 5 -3 -3 / 5 0 -3 / 5 -3 -3 -3 -3 -3 / 5 0 -3 / 5 5 -3 -3 -3 -3 / -3 0 -3 / 5 5 5 -3 -3 -3 / -3 0 5 / -3 5 5 -3 -3 5 / -3 0 5 / -3 -3 5 -3 5 5 / -3 0 5 / -3 -3 -3
	Robinson オペレータ		1 2 1 / 0 0 0 / -1 -2 -1 2 1 0 / 1 0 -1 / 0 -1 -2 1 0 -1 / 2 0 -2 / 1 0 -1 0 -1 -2 / 1 0 -1 / 2 1 0 -1 -2 -1 / 0 0 0 / 1 2 1 -2 -1 0 / -1 0 1 / 0 1 2 -1 0 1 / -2 0 2 / -1 0 1 0 1 2 / -1 0 1 / -2 -1 0
	Prewitt オペレータ（ディファレンシャル型）	エッジ強度およびエッジ方向を求める。	SX: -1 0 1 / -1 0 1 / -1 0 1 SY: -1 -1 -1 / 0 0 0 / 1 1 1 （エッジ強度）$= \sqrt{SX^2 + SY^2}$ or $\lvert SX \rvert + \lvert SY \rvert$
	Sobel オペレータ		-1 0 1 / -2 0 2 / -1 0 1 -1 -2 -1 / 0 0 0 / 1 2 1 （エッジ方向）$= \tan^{-1}(SY/SX)$
色彩	色彩距離による分類	複数個の標準色と照合して、色彩距離がある許容範囲の時、その標準色として分類する。	$Di = \sqrt{(R-\alpha i)^2 + (B-\beta i)^2 + (G-\gamma i)^2}$ or $\lvert R-\alpha i \rvert + \lvert B-\beta i \rvert + \lvert G-\gamma i \rvert$ Di　：クラスiの標準色からの色彩距離 R, B, G　：入力画素の光の3原色の各濃度値 αi, βi, γi：クラスiの標準色の光の3原色の各濃度値

す毎にマスク演算を繰り返すのかという観点である。もう一つは、マスク演算をすべて最終してからマスク演算値間の処理を行うか、もしくは個々のマスク演算と並行してマスク演算値間の処理を行うかという観点である。

ISPにおいては、それぞれの観点から、カーネルを切り出す毎に、すべてのマスク演算を行うと同時に、それらの演算値間の処理まで行って、一つのカーネルに対する最終値まで求めることにした。なぜなら、より効率的なシステムを構築するためには、画像メモリの参照を最小限にすることが望ましいからである。つまり上記の方法によると、入力画像の読み出しと出力結果の書き込みの、2回のメモリ参照でマルチマスクプロセッシングを処理でき、メモリ参照によるオーバーヘッドを最小限に抑えることができる。

なお、表2からも知れるように、マルチマスクプロセッシングにおけるマスク演算値間の処理のほとんどは、比較演算によりなされ得る。そこで、エバリュエーションユニットに比較器を設け、各種の2値化処理と合わせて、マスク演算値間の処理を行わせることにした。比較演算で算出できないマスク演算値間の処理については、チップ面積などの理由により、ISPで実現するのは断念した。

4. アーキテクチャの検討

4.1 アーキテクチャ上の課題

3.に述べた方針に基づいて、ISPのアーキテクチャを検討してゆく上で、次の四つの課題があると考えられる。

(1) 画像データのPEへの供給方式。
(2) マスクデータのPEへの供給方式。
(3) マスク演算値間の比較演算方式。
(4) 比較演算結果の出力方式。

(1)と(2)の課題は、PEにおける演算の順序と、リンケージにおける演算の制御に係わる問題である。一方、(3)と(4)の課題は、エバリュエーションユニットにおける演算に関する問題であり、(1)と(2)から切り離して考えることができる。なぜなら、3.に述べた方針に基づくと、カーネル毎の各マスクデータに対する演算値は、リンケージユニットからエバリュエーションユニットへ順次供給され、(1)と(2)の方式が異っても、転送のタイミングが多少違ってくるだけだからである。そこで、(1)、(2)の検討と、(3)、(4)の検討を、分けて考察する。

ここで考慮しなければならないのは、二つの方式によるカーネル拡張への対処である。なぜなら、PE増殖方式とPE節約方式では、画像走査方式が異るからである。PE増殖方式ではラスタ走査が用いられ、PE節約方式ではスティック走査が用いられる。

しかし、ラスタ走査はスティック走査の特殊なもの－スティック長が1のスティック走査－であるから、スティック走査により走査された画像データを処理できれば、ラスタ走査により走査された画像データも処理できることになる。つまり、PE節約方式に対処できれば、PE増殖方式にも対処できると言える。そこで、PE節約方式によるカーネル拡張に対して、上記の課題を検討することにする。

4.2 データ供給に関する検討

ここでは、画像データとマスクデータのPEへの供給方式について検討するが、討論を簡潔にするため、スティック長が2のスティック走査を用いて、2個のマスクデータを扱うマルチマスクプロセッシングを実行する場合について考察する。

上記の処理を実行する際、一つのPEとそれに関連する部分だけを抜粋した模式図を、図4に示す。同図において、P1・P2は画素データを、A1・A2は1個目のマスクデータを、B1・B2は2個目のマスクデータを表している。なお図4においては、空間積和演算を想定して演算機能を選択しているので、リンケージユニットのALUからは、演算結果として次の二つの値が出力される。

$$R1 = A1 * P1 + A2 * P2。$$
$$R2 = B1 * P1 + B2 * P2。$$

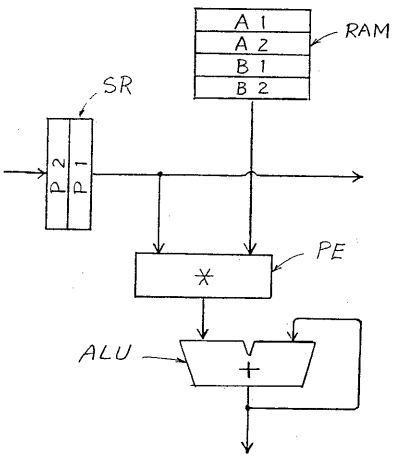

図4 スティック長2、マスクデータ数2のマルチマスクプロセッシングの模式図

PEへの画像データとマスクデータの供給方式は、PEにおける演算の実行順序に左右される。上記の演算では、一つのPEで4回の乗算が実行されなければならないが、次の二つの順序が考えられる。

(1) A1＊P1、A2＊P2、B1＊P1、B2＊P2。

(2) A1＊P1、B1＊P1、A2＊P2、B2＊P2。

つまり、(1)は、マスクデータ毎に演算を進める方式であり、(2)は、画像データ毎に演算してゆく方式である。

(1)の方式では、マスクデータ単位の演算を、マスク数だけ繰り返すことになり、演算の制御は容易であるが、同じ画素データを繰り返し入力する必要がある。この時、シフトレジスタの遅延段数は、(スティック長)×(マスク数)となる。図4の例では、シフトレジスタ当たり4段にすればよいが、データユニットには4個のシフトレジスタがあることや、スティック長やマスク数の増加を考慮すると、シフトレジスタのゲート数は厖大なものになる。たとえば、表2にあるPrewittオペレータのテンプレート型を処理する場合、各シフトレジスタの遅延段数は24段になる。

(2)の方式では、個々の画素データは、マスク数に等しい回数だけ続けて使用されるため、何度も入力する必要はない。マスク数が増加すれば、それに応じて、シフトレジスタの転送レートを低下すればよいので、シフトレジスタの遅延段数は、常にスティック長と同数でよい。しかし、それぞれのマスクデータに対する演算値を算出するためには、リンケージユニットに、マスク数と同数の一時記憶バッファを用意して、複雑なデータ制御を施さなければならない。

(1)、(2)の長所と短所を比較検討した結果、ISPにおいては、下記の理由により(1)の方式を採用することにした。まず、制御が簡単であること。次に基本アーキテクチャをほとんど変更する必要がないこと。さらに、シフトレジスタの段数増加は、図5に示す方式により解決され、若干のゲート数増で対応できるからである。

図5において、PEへ転送される画素データは、同時にシフトレジスタの初段にフィードバックされ、マスク数に応じて再度PEへ転送されることになる。一方、マスクデータは、RAMが外部端子から直接アドレスされるため、(1)の方式を採用することによる不都合は何もない。

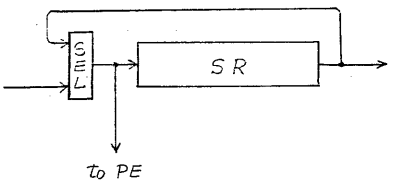

図5　シフトレジスタにおけるフィードバック機構

なお、それぞれの画素データには、2ビットのセマンティックコード(Semantic Code)を付加して、その画素データがスティック内のどこに位置するかという情報を持たせることにした。このセマンティックコードが、シフトレジスタのフィードバック機構や、リンケージユニットにおける統合演算を制御することになる。

これらの結果、それぞれのマスクデータに対する演算値は、スティック長に等しいマシンサイクル毎に、リンケージユニットから出力されることになる。

4．3　比較演算に関する検討

3.の方針に基づくと、エバリュエーションユニットは、比較器を中心とした構成になる。ここでは、比較器を用いる閾値処理(Thresholding)から、エバリュエーションユニットの基本的なアーキテクチャを明らかにした後、マルチマスクプロセッシングにおけるマスク演算値間の処理について検討する。

閾値処理には、閾値が固定されたままの固定閾値処理と、処理の進行に伴って閾値の変化する浮動閾値処理とがあるが、浮動閾値処理は、カーネル内の演算と固定閾値処理などに変換できるものが多いので、ISPにおいては、固定閾値処理を中心に、エバリュエーションユニットの基本的アーキテクチャを検討した。

固定閾値処理においては、参照データは固定の閾値により2値化されるが、より汎用性を高めるために、二つの閾値 T_H、T_L ($T_H > T_L$)を用いた4種類の2値化処理を実現することにした。参照データを f、出力結果を g とすると、4種類の2値化処理は次のように表される。

(1) $g=1$ if $f>T_H$, otherwise $g=0$。

(2) $g=1$ if $f \leq T_L$, otherwise $g=0$。

(3) $g=1$ if $f>T_H$ or $f \leq T_L$, otherwise $g=0$。

(4) $g=1$ if $T_L < f \leq T_H$, otherwise $g=0$。

これらを実行するためには、2個の比較器と2個の閾値レジスタが必要であるが、閾値レジスタの内容が、演算の実行に先立って任意に設定できるだけでよい。しかし、比較結果により(たとえば $g=1$ の時)、参照データで

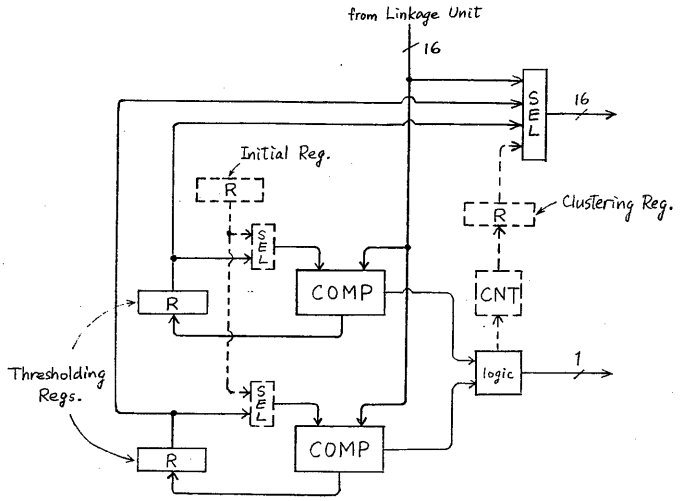

図6　エバリュエーションユニットの構成

この時、いずれか一方の閾値レジスタの内容（移動前のカーネルに対するマスク演算値の最大値もしくは最小値）が、リンケージユニットの出力バスを介して、LSI外部へ出力される。なお、初期値レジスタの内容は、演算実行前に任意の値に設定され、演算実行中書き替えられることはない。

(2)の機能を実現するためには、マスクデータを区別するための情報を、エバリュエーションユニット内で作り出さなければならない。ISPにおいては、バイナリィカウンタを内蔵して、このための数値を作り出すことにした。バイナリィカウンタは、カーネルが移動した時初期化され、一つのマスク演算が終わる度カウントアップし、比較判定の信号により、その内容は適宜クラスタリングレジスタに書き込まれる。その結果、一つのカーネルに対するマスク演算がすべて終了した時、マスク演算値の最大値（もしくは最小値）は閾値レジスタに、その値に対応するマスク番号はクラスタリングレジスタに残される。これらニつの値は、リンケージユニットの出力バスを介して、時分割でLSI外部に出力される。ここで、リンケージユニットの出力バスを用いたのは、LSIの端子数を削減するためである。

閾値レジスタを書き替えられるならば、参照データの最大値や最小値を求めることができる。また2値化情報と合わせることにより、最大値・最小値の座標値を求めることも可能である。

これらの検討から、エバリュエーションユニットの基本的アーキテクチャを、図6の実線で示す構成とした。同図に示すように、リンケージユニットからの参照データは、二つの比較器に同時に与えられ、閾値レジスタ内のT_H・T_Lと比較され、判定結果は、上記の(1)～(4)のいずれかの論理に従って、1ビットに統合される。閾値レジスタは、最大値・最小値探索の時のみ参照データで書き替えられる。

図6の実線で示した構成を基にして、マルチマスクプロセッシングにおけるマスク演算値間の処理を検討してゆく際、次の二つの機能が要求される。

(1)　カーネル毎に算出されるマスク演算値の内の、最大値もしくは最小値の抽出。

(2)　(1)で求めた最大値・最小値の元となるマスクデータ番号の抽出。

上記(1)の点については、閾値レジスタの内容が書き替えられることから、基本的には支障はないが、カーネルが移動する度に、閾値レジスタの内容を初期化する必要がある。しかし、カーネルに対する最大値・最小値を出力する点から、カーネルの移動時に閾値レジスタを初期化することは好ましくない。そこで、初期値レジスタを設け、カーネルの移動時は、閾値レジスタに代わって初期値レジスタの内容を、比較演算に用いることにした。

上に述べたマスク演算値間の処理に関する検討は、図6における点線で表される箇所として、アーキテクチャに反映される。この結果、マルチマスクプロセッシングにおける比較演算を、効率良く実行できるエバリュエーションユニットのアーキテクチャが確立する。なお、エバリュエーションユニットにおけるデータ制御も、データユニットやリンケージユニットと同様に、各画素データに付加される2ビットのセマンティックコードを用いることにした。

5．アーキテクチャの評価

ここでは、4．で検討したアーキテクチャにおいて、表2にある、8個の差分型マスクを用いるPrewittオペレータのテンプレート型を例に挙げて、処理速度について評価した。

上記のオペレータを、3個のISPを用いてPE増殖方式により処理する場合、256×256画素の画像ならば87.4msで、512×512の画像ならば376.3ms

で実行できる。ISP1個でPE節約方式により処理する場合でも、256×256画像を262.1msで、512×512画像を約1.1秒で実行できる。

6. むすび

画像処理用LSI－ISPは、Prewittオペレータのテンプレート型のように、複数個のマスクデータを用いるマルチマスクプロセッシングを、効率良く実行できる。

ISPにおいては、カーネルを切り出す毎にマスク演算値間の処理まで行うことにし、そのアーキテクチャを、画像データとマスクデータのPEへの供給方式と、マスク演算値間の処理方式に分けて検討した。前者については、マスクデータ毎に演算を進めてゆくことにし、後者については、比較器を中心としたエバリュエーションユニットの構成を明らかにした。

本論文では、マルチマスクプロセッシングを効率良く実行するISPのアーキテクチャについて述べたが、今後の課題は、ISPの特長を十二分に活かした効果的な画像処理システムの構築法の開発である。

謝辞　本研究をご支援下さった(株)日立製作所大みか工場桑原洋副工場長、有益なご討論を頂いた同社中央研究所江尻正員主管研究員、同社生産技術研究所秦清治主任研究員、およびLSI開発にご協力頂いた諸氏に深く感謝する。

参考文献

(1) Fukushima, T. et al: An Image Signal Processor, ISSCC Digest of Technical Papers, pp. 258-259 (1983).
(2) 福島, 小林他：画像処理用LSI－Image Signal Processorのアーキテクチャ, 信学論, Vol. J66-C, No. 12, pp. 959-966 (1983).
(3) 福島, 小林他：画像処理用LSI－ISPのアーキテクチャ, 情報処理, WGCV 26-6 (1983).
(4) Prewitt, J.M.S.: Object Enhancement and Extraction, in Picture Processing and Psychopictorics, Academic Press, pp. 75-149 (1970).
(5) Kirsch, R.: Computer Determination of the Constituent Structure of Biological Images, Comput. Biomed. Res., Vol. 4, pp. 315-328 (1971).
(6) Robinson, G. S.: Edge Detection by Compass Gradient Masks, CGIP, Vol. 6, pp. 492-501 (1977).
(7) 坂根, 田村：SPIDER開発を通して観たディジタル画像処理アルゴリズムの現状(3), 情報処理, WGCV 4-4 (1980).
(8) 柏岡, 江尻, 坂本：時分割パターン認識技術による群制御トランジスタ組立システム, 電学論C, Vol. 96, No. 1, pp. 9-16 (1976).