大規模シストリックアレイプロセッサにおける 局所並列画像処理

正員 岡田 至 弘 (龍谷大)

正員 坂井 利 之 (龍谷大)

Locally Parallel Image-Processing in Large-Scale Systolic Array Processor
Yoshihiro Okada, Member, Toshiyuki Sakai, Member (Ryukoku University)

Recent years there has been a remarkable development in VLSI technology. A proposal has made for the realization of VLSI algorithm performance in Image Processing fields.

The Systolic Array is an example for VLSI architecture. Systolic Array has a control system of parallel processing abilities. Pre-processing and transformation characteristic are very useful for industrial applications. In this paper we present a discussion about the method of realization of parallel image processing using Systolic Array Processor (Geometric Arithmetic Parallel Processor chips; GAPP) and LSAPS (Large Scale Systolic Array Processor System). Particularly, here, we discuss in detail about pre-processing and transformation characteristics of images, in which high speed processing capability are essential to deal with large volume data.

キーワード：シストリックアレイ，シストリックアルゴリズム，並列処理機構，並列画像処理

1. ま え が き

近年のLSI技術の著しい発展により，画像処理を高速に実行するアルゴリズムをVLSI上で実現しようという提案⁽¹⁾⁽⁴⁾⁽⁵⁾がなされている。このVLSI指向アーキテクチャの代表的なものとしてシストリックアレイ⁽²⁾がある。高度の並列性をもち，かつ規則性のある制御方式をもつシストリックアレイを，産業応用での画像処理，特に画像の前処理・特徴抽出に用いることは有効であると思われる。

本論文では，シストリックアレイアーキテクチャ⁽³⁾⁽⁶⁾⁽⁷⁾のVLSIであるGAPP (Geometric Arithmetic Parallel Processor chips)を用いた局所並列画像処理の実現方法について考察する。ここでは，膨大なデータ量を扱うために高速化が要求される前処理・特徴抽出段階の画像処理を対象に詳細に検討する。

GAPPはSAL (Systolic Algorithm Language)と呼ぶ専用記述言語により，SIMD (Single Instruction Multiple Data Stream)型の専用アレイに比べ，プログラマブルで柔軟な処理が実現できる。

以下，シストリックアーキテクチャについて述べ，

実際に構築したGAPPによる大規模シストリックアレイプロセッサシステム (Large Scale Systolic Array Processor System; LSAPS)の構成を示す。そして，画像演算の実行形式をシストリックアレイとのかかわりから考察した後，シストリックアレイ上での局所並列画像処理の実現方法について述べる。また，実際にLSAPSによる画像の前処理・特徴抽出を行い，シストリックアレイの有効性を示す。

2. GAPPによる大規模シストリックアレイプロセッサシステム (LSAPS)

〈2・1〉 GAPP シストリックアレイの特徴は，まずデータと制御の流れが単純で規則的であることである。そのため，高い並列動作が可能になる。また，少ない種類の簡単なセルで実現できるため拡張性があり，設計・デバッグが簡単でありVLSI化に向いている。多数のセル (processor)を配置するという実装面での難しさは，これによりある程度解消することができる。更に，データの授受は隣接セル間で行われるのでホストの負担が少ないことや，セル数の増加につれ，処理能力が比例的に増加するという長所があるた

め、画素数の膨大な画像処理には向いているといえる。

図1に本研究で用いた、NCR製シストリックアレイプロセッサチップ (GAPP)⁽⁶⁾ の構成を示す。1チップ上に 12×6 のセルが二次元状に配置され、各々のセルが隣接する (東西南北) 四つのセルと接続されている。セルへのデータの入出力は南端のセルより伝搬させて行う。また大局的な情報として全セルの1ビットデータの NOR をとる GO (Global Output) 出力がある。各セルは図2のように、1ビット ALU、四つの1ビットレジスタ (NS, EW, C, CM)、ローカル

図1 GAPPの構成 (セルの配列)
Fig. 1. GAPP Processor element.

図2 セル内の構成
Fig. 2. Schematic diagram of one processor.

図3 RAMの構成
Fig. 3. Schematic diagram of the RAM element.

メモリーにより構成されている。ローカルメモリーは128ビットのRAMであり、ビット単位のアクセスが可能である。図3にRAMの構成を示す。

全セルのRAMは128枚のビットプレーンとしてビットごとに統一的に扱う。

GAPPの特徴は二次元格子状に配列されたセルの並列動作であり、これは多分野の画像処理の高速化に適している。完全画素並列プロセッサでは、一般的にプロセッサ配列に対応して、処理可能な画像データ配列は固定化される。これに比して、GAPP複数個からなるシステム構成では、言語SALを用いたプログラマブルな処理により、完全画素並列プロセッサとは異なった柔軟な処理を実行できる。また、ビットシリアルな処理であるので、対象画像のビット構成に応じて任意精度の演算ができる。

GAPPの基本動作は、セル間でのデータの移動、1ビット加減算、RAMへの書込み、セル内のデータの移動の4種類からなる。この四つの動作を各レジスタとRAMに対し、繰り返し施すことによって処理を実行していく。

一方短所として、離れたセル間での通信に時間がかかることや、各画素独立に動作させることが困難であることなどがある。しかし画像処理、特に前処理・特徴抽出段階においては、単純な基本動作の組合せにより処理が構成できるので、この欠点はあまり障害にはならないと思われる。

〈2・2〉 LSAPS 構成 図4にGAPPチップを複数個組み合わせたLSAPSの構成を示す。LSAPSは、 96×96 個のセルから構成される大規模シストリックアレイプロセッサシステムである。また、局所並列画像処理での柔軟な処理アルゴリズムの開発を目的として、全体のセルは、大きく四つのブロックに分けられている。一つのブロックは、32個のGAPPチップ

図4 LSAPSの構成
Fig. 4. Organization of the LSAPS.

ブからなり、 48×48 のセル構成となる。四つのブロックは、局所並列画像処理における処理単位に応じて組合せが可変となっている。すなわち、カラー画像を対象とするRGB 3入力対応の $(48 \text{ セル} \times 48 \text{ セル}) \times 3$ ブロック構成(残り1ブロックは作業用)や、方形領域を処理単位としたブロック構成をとることができる。これらのブロック構成は、図中のarray制御部で組合せ制御が行われる。また、入力画像は図中のフレームバッファで取り込まれ、ホスト計算機からのDMA制御により各ブロックに転送される。

ホスト計算機では、〈2・3〉節で述べるシストリックアルゴリズム言語(SAL)で生成した実行コード(LSAPS制御信号)を各ブロック内のGAPPヘダウロードしていく。

〈2・3〉シストリックアルゴリズム言語(SAL)

〈2・1〉節で述べたように、GAPPは非常に単純な4種類の動作だけを組み合わせ、複雑な処理を行う。そのため、一般にそのアルゴリズム設計は複雑になってしまう。これをマイクロコードレベルでプログラム設計した場合、一つのGAPP命令ごとにRAMアドレスを決定しなければならないことや、単純な繰返しもすべて書き並べなければならないなど、多くのわずらわしさをもつ。これは、GAPP命令がプログラムフロー制御をもたないために起こる。

このようなGAPPプログラム開発を簡単化するために、シストリックアルゴリズム記述言語としてSAL(Systolic Algorithm Language)を開発した。SALはC言語を基本に、ハードウェア独立なプログラムのフロー制御を行い、LSAPSに対する命令を制御する。SALで書かれたプログラムはSALコンパイラによって、GAPP命令とそれを実行するRAMアドレスの組からなるオブジェクトファイルに展開される。これを更にホストコンピュータ上でLSAPS制御信号に変換して各ブロック内のGAPPに送る。

SALの構成は大きく分けて、Cライクな部分、GAPP命令部、RAMアドレス指定部の三つになる。Cライクな部分は、プログラムフロー制御をもたないGAPP命令に代わって流れを制御し、プログラム開発を簡易化している。GAPP命令部は実際にアレイ上のセルの動作を指示する命令である。前述の、GAPPのもつ4種類の基本動作を各セルのレジスタに対して命令する。RAMアドレス指定部は、128枚のビットプレーンであるRAMのアドレスを、IMAGE変数と呼ぶ変数を用いて指定する。IMAGE変数を用いることにより、各GAPP命令に対するアドレス指定を相対的に行うことができる。また、一つ

のGAPP命令および一つのRAMアドレス指定は、GAPPアレイ上の全セルに対して同時に実行される。

3. シストリックアルゴリズムと画像処理

ここでは、シストリックアーキテクチャ上での並列画像処理の方法について述べる。以下において、シストリックアレイは構造的な意味で用い、シストリックアレイプロセッサはシストリックアレイ構造をもった処理装置を指すものとして使い分ける。

〈3・1〉局所並列画像処理とシストリックアレイ

局所並列画像処理⁹⁾は、実際の画像解析に何らかのかたちで必ず用いられる重要な処理手法である。雑音除去、画像強調、輪郭抽出などの、前処理で不可欠な処理の大部分がこれに属する。この処理は、「各画素の濃淡値を近傍画素の情報に基づいて書き換える処理」と言い換えることができる。これは、シストリックアレイの定義である「東西南北の四つのセルとデータをやりとりしながら演算を進める」と、双対の関係にある。従って、並列型局所処理はもっとも有効なかたちでシストリックアレイ上に実現しうる処理である。よって、局所並列画像処理に属するか、あるいはこれに書き換えることができる処理は、一般にシストリックアレイ構成上で実現可能であるといえる。

局所並列画像処理は主に非反復型と反復型に分けることができる。

非反復型局所並列画像処理(単純近傍処理)は、反復を必要とせず、1回の近傍処理だけ、あるいは1回の近傍処理の組合せにより行う処理である。線形フィルタや濃淡値差分など画像処理において重要な処理が多く含まれるが、アルゴリズムは比較的簡単であるので、シストリックアレイ上でも容易に構成することができる。

反復型局所並列画像処理(再帰的近傍処理)は、特定の非反復型処理を再帰的に繰り返すことにより実行される処理であり、距離変換、細線化、図形収縮、伝搬などの処理が含まれる。処理の再帰的な繰返しにより対象画像のもつ大局的な特徴を抽出することが可能となる。

〈3・2〉シストリックアレイ上での並列画像処理

画像処理、特に前処理・特徴抽出の段階においてはフィルタリングや微分処理など比較的似通った処理が多く、それらの処理は数種類の単純な動作の組合せにより構成することができる。GAPPは、セル間のデータの移動・1ビット加減算・RAMへの書込み・セル内のデータの移動の4種類の動作しかもたない。しかし、この四つの動作を組み合わせることにより、多く

の画像処理を構成することができる。

ここでは、シストリックアレイ上での画像処理の構成方法について、メディアンフィルタの構成法を例示することにより考察する。

メディアンフィルタとは、濃淡値で表された各画素の値を、3×3の近傍9画素の濃淡値の大小の値のうち、ちょうど中間（5番目）の値にすべての値を書き換える処理であり、雑音除去に有効な処理である。

シストリックアレイ上でメディアンフィルタを構成するポイントは、全セルが同一の動作しかできないアレイ上で、いかにして近傍画素に対応する九つのセルのデータを大小順に並べ換えるかということである。9画素の濃淡値の大小は画素ごとにランダムである。ここでは、1ビット加減算による“借り”（Borrow）を使って並べ換えを行う方法を示す。

以下において、GAPP上で1ビット加減算により得られる値をSum（サム）、桁上げをCarry（キャリー）、借りをBorrow（ボロウ）と表すことにする。

（1）データの移動 比較すべき近傍9画素のセルのデータを中心セルに集める。各セルのRAMアドレス0～7（図3参照）には入力画像データ（8ビット）が書き込まれているものとする。図5（a）の入

図5 入力データと移動

Fig. 5. Input data (8 bits) and data movement.

Fig. 6. The order of C → E → NE → N → NW → W → SW → S → SE moving data to an adjacent cell.

力データを例に考えていく。

（i）EWレジスタ（図2参照）により、入力データを東隣のセルへ移動させる。移動は1ビットずつ順番に行う（図5（b）参照）。

（ii）移動したデータを、次のRAMアドレス8～15に書き込む。

（iii）同様の方法で、図6のように各セルの入力データをC → E → NE → N → NW → W → SW → S → SEの順に隣接セルへ一つずつ移動させ、セルのRAMアドレスの続く8ビット（16～23, 24～31, …）に順番に書き込んでゆく。ただし、Cは中心セル、Eは東隣のセル、NEは東北にあるセル、というように方向でセルを表している。移動は1ビットずつ行う。これにより、ソートすべき九つのデータが各セルのRAM上にそろそろ。各セルのRAM上に積まれたデータは、図7のようにデータの移動方向と対称なセルの順番になっている。

（2）データの並べ換え 九つのデータを値の大きい順番に並べ換える。ソートは、RAMに積んだ上のデータから順に比較・交換していくことで行う。ここで、RAMの上とは、アドレスの大きいほうを指すものとする。

（iv）RAMの一番上のデータと2番目のデータを比較する。データの最下位ビットから順に、Borrowをおくりながらビット減算を繰り返す。最後に残るBorrowは、上のデータのほうが下のデータよりも値が大きいセルでは1になり、そうでないセルにおいては0になる。よって、Borrowが1のセルは上のデータと下のデータを入れ換える。Borrowが0のセルはそのままにしておく。

その入れ換えの方法を考える。

（v）シストリックアレイ上ではデータをビット単位で扱うので、データの入換えのためには、ビットごとの論理的な交換条件を考えなければいけない。ビット交換論理表を表1に示す。

NW	5	5	20	25	50	25	20
N	5	20	25	50	25	20	5
NE	20	25	50	25	20	5	5
E	20	50	75	50	20	5	5
SE	20	25	50	25	20	5	5
S	5	20	25	50	25	20	5
SW	5	5	20	25	50	25	20
W	5	5	20	50	75	50	20
C	5	20	50	75	50	20	5

図7 データ移動後のRAMの様子

Fig. 7. The state of RAM at moved data.

表 1 ビット交換論理表
Table 1. Logic table of the bit exchange.

データ ボロウ	入換え前		入換え後	
	上のビット	下のビット	上のビット	下のビット
0	0	0	0	0
	0	1	0	1
	1	0	1	0
	1	1	1	1
1	0	0	0	0
	0	1	1	0
	1	0	0	1
	1	1	1	1

表から、二つのデータの各ビットの交換が必要となるのは、Borrow が 1（つまりデータの大小順が逆）であり、かつ上のデータと下のデータのビットが異なる時のみであることがわかる。Borrow が 1 であっても、ビットが上のデータ・下のデータとも同じときは、ビットを交換する必要はない。Borrow が 0 のときは、ビットの交換の必要はない。

これを基に、シストリックアレイ上でのデータの入換えを行う。

(vi) データの比較・交換操作を、九つのデータすべてに対しバブルソートの順に繰り返す。これにより、各セルの RAM 上には 9 画素のデータが値の大小順に並べ換えられて積まれていく。

(3) 結果の取出し

(vii) RAM 上のソートされたデータより、中間のデータ（下から 5 番目のデータ）がメディアンフィルタ出力となる。

以上により、シストリックアレイ上でメディアンフィルタが構成される。

4. LSAPS を用いたエッジ抽出処理

ここでは、LSAPS を用いた画像のエッジ抽出処理について述べる。

〈4・1〉 処理の手順 入力データは 480×480 画素の 8 ビット濃淡画像である。LSAPS の全セル数は 96×96 であり、この大きさだけでは入力画像全画素に対応できない。そこで、ホスト計算機上で入力画像をアレイの大きさ（ 96×96 画素）にブロック化（25 ブロック）して、ブロックレベルでラスタ走査しながら順番に GAPP にデータを流す方法をとる。このため、入力画像は分割処理される。画像のブロック化は GAPP 上での処理回数を増やすが、反面、ブロック単位で可変な処理ができるようになる。

次に、GAPP 上の RAM に入力されたデータは、

(1) 式に示す Sobel 変換によりエッジ抽出を行う。

入力データは、 3×3 マトリックス $\begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}$ に配置さ

れているとする。

$$\left. \begin{aligned} \text{Sobel}(E) &= \{(\Delta X)^2 + (\Delta Y)^2\}^{1/2} \\ &= \max(|\Delta X|, |\Delta Y|) \\ &\quad + 1/2 \min(|\Delta X|, |\Delta Y|) \end{aligned} \right\} \cdots (1)$$

$$\Delta X = (C + 2F + I) - (A + 2D + G)$$

$$\Delta Y = (G + 2H + I) - (A + 2B + C)$$

以下に Sobel 変換の構成法を示す。ただし、入力データを $F = \{f_{ij}\}$ 、出力データを $G = \{g_{ij}\}$ とする。

(1) 入力データ F を一つ北のセルに移動して、ビット加算する。

$$G_1 = \{g_{1ij} | g_{1ij} = f_{ij} + f_{i-1j}\} \cdots \cdots (2)$$

(2) G_1 を一つ南のセルに移動して、ビット加算する。

$$G_2 = \{g_{2ij} | g_{2ij} = g_{1ij} + g_{i+1j}\} \cdots \cdots (3)$$

(3) G_2 を東西のセルに一つずつ移動して、ビット減算する。

$$\begin{aligned} G_3 &= \{g_{3ij} | g_{3ij} = g_{2ij+1} - g_{2i-1j}\} \\ &= (f_{i-1j+1} + 2f_{ij+1} + f_{i+1j+1}) \\ &\quad - (f_{i-1j-1} + 2f_{ij-1} + f_{i+1j-1}) \\ &= \Delta X \end{aligned} \cdots \cdots (4)$$

(4) 符号のビットプレーンを取り出し、データの各ビットとの排他的論理和をとる。更に、符号ビットの値を加え、 ΔX の絶対値を求める。

$$\left. \begin{aligned} S &= \{s_{ij} | s_{ij} \text{ は画素 } (i, j) \text{ の符号ビット}\} \\ G_4 &= \{g_{4ij} | g_{4ij} = (g_{3ij(k)}, s_{ij}) + s_{ij}\} \end{aligned} \right\} \cdots \cdots (5)$$

k : bit number

(5) (1)～(4)までを Y 方向にも繰り返し、 $G_4 = |\Delta Y|$ を求める。

(6) G_4 から G_4' をビット減算した Borrow により、 $\max(|\Delta X|, |\Delta Y|)$ と、 $\min(|\Delta X|, |\Delta Y|)$ を求める。

$$\begin{aligned} B &= \{b_{ij} | b_{ij} \text{ は } (g_{4ij} - g'_{4ij}) \text{ の Borrow}\} \\ G_{\max} &= \max(|\Delta X|, |\Delta Y|) \\ &= \{g_{\max ij} = g_{4ij} \cdot b_{ij} + g'_{4ij} \cdot b_{ij}\} \end{aligned} \cdots \cdots (6)$$

$$\begin{aligned} G_{\min} &= \min(|\Delta X|, |\Delta Y|) \\ &= \{g_{\min ij} = g_{4ij} \cdot b_{ij} + g'_{4ij} \cdot b_{ij}\} \end{aligned} \cdots \cdots (7)$$

(7) $G_{\max} + 1/2 G_{\min}$ を求めて終了。

$$G_{\text{out}} = \{g_{ijk} | g_{ijk} = g_{\max ijk} + g_{\min ijk+1}\} \cdots \cdots (8)$$

k : bit number

〈4・2〉 実行結果の考察 前節の処理では、入力画像を図4のLSAPS上のFrame Bufferで25ブロックに分割して行った。ここでは分割処理後のGAPP上の処理時間について述べる。GAPPのクロックは10 MHzであり、1クロックでGAPP命令一つを実行する。また、前節の処理においてGAPP上での命令数は、1ブロック当たり210であり、1ブロックのデータ入出力の命令数は180となった。ここから全GAPP命令数は、 $(210+180) \times (25) = 9,750$ となる。よって、処理時間は約1 msとなる。

次に、ブロック分割処理について考察する。前節の処理では、ブロックの分割および入出力処理には、1ブロック当たり約2 ms要した。よって、LSAPS上での全処理時間は、 $(2 \times 25 + 1) \text{ms}$ となる。このことは、GAPPなどシストリックアレイプロセッサの優位性を生かすためには、アレイに対して画像データを間断なく供給する強力な専用のアーキテクチャが不可欠であることを示している。また、前節のエッジ抽出処理結果では、ブロック境界近辺での抽出誤りが生じた。このため、ブロック分割を行う際、隣接ブロック間で画素オーバーラップして分割する必要があると思われる。

5. おわりに

ここでは、ブロックサイズを 96×96 画素で、画像を25にブロック化した方が、更に多くのセルを用いてブロックサイズを大きくした場合は、より高速な処理が可能であるといえる。

同様に、シストリックアレイでは、画素数に合わせてセル数を増減させることにより、入力画像のサイズにかかわらず高速な処理が実現できる。逐次型処理では、入力画像の画素数が増加するとそれに比例して実行時間も増加する。よって、サイズの大きい入力画像であるほどシストリックアレイは有力であるといえる。また、完全画素並列プロセッサによる処理と異なり、入力画像のサイズに左右されない処理が可能となり、近傍処理などの非反復型局所並列画像処理に対して有用であると考えられる。

今後の課題として、LSAPS上のシストリックアレイ構造に対応した処理アルゴリズムの開発と、ブロック分割およびデータの入出力機構のロジック化を進めていく必要がある。

最後に本研究を進めるにあたって、LSAPS開発に御尽力いただいた日本NCR(株)高橋秀明氏、SAL開発に従事した谷口美香氏らに謝意を表す。

(平成5年3月22日受付、同5年7月12日再受付)

文 献

- (1) R. Davis & D. Thomas: Systolic array chip matches the pace of high-speed processing", *Electronic Design*, October 31 (1984)
- (2) H. T. Kung: "Why Systolic Architectures?" *IEEE Trans. Comput.*, **Computer15**, No. 1, 37 (1982)
- (3) N. Petkov: Systolic Parallel Processing, p. 1 (1993) North Holland
- (4) T. Gross & H. T. Kung: "Warp as a Machine for Low-level Vision", IEEE Int. Conference on Robotics & Automation (1985)
- (5) L. Uhr (Editor): Parallel Computer Vision, p. 1 (1987) Academic Press
- (6) V. K. Prasanna Kumar: Parallel Architectures & Algorithms for Image Understanding, p. 278 (1991) Academic Press
- (7) E. L. Cloud: "The Geometric Arithmetic Parallel Processor"; Proc. 2nd Symposium on the Frontiers of Massively Parallel Computation, CMU, p. 43 (1988)
- (8) "Geometric Arithmetic Parallel Processor" (NCR45 CG72), NCR-Data sheet (1987)
- (9) 前田 明:「画像処理マシン」, 情報処理, **28**, 19 (昭62-1)
- (10) 日本色彩学会編: 色彩科学ハンドブック (昭55) 東京大学出版会
- (11) 鳥脇純一郎・横井茂樹:「画像処理のアルゴリズム」, 情報処理, **21**, 613 (昭55-6)

岡 田 至 弘 (正員)

昭和52年立命館大学理工学部卒業。現在、龍谷大学理工学部電子情報学科助教授。工学博士。学内キャンパスLANの構築、および分散型計算機システム、パターン認識・理解の研究開発に従事。情報処理学会、電子情報通信学会、人工知能学会会員。

坂 井 利 之 (正員)

昭和22年9月京都大学工学部電気工学科卒業。同大学大学院特別研究生、講師、助教授を経て、35年同教授となり、有線通信工学、情報基礎論担当。63年3月定年退官。現在、龍谷大学教授。京都大学名誉教授。