画像処理用超並列プロセッサ AMP のプログラミングと性能評価について†

山元規靖†† 鶴田直之††
谷口倫一郎†† 雨宮真人††

画像処理においては，膨大な計算量による処理時間の増大が大きな問題となり，これを解決するための様々な画像処理用並列プロセッサシステムが開発されている．筆者らは，各 PE を非同期に動作させることにより，単純な画像処理だけでなく画像理解のための処理も可能な柔軟で高能率な自律型非同期超並列プロセッサ AMP の開発を行っている．また，この AMP システム上のプログラミング言語として，関数型言語 Valid-A の開発も進めている．本論文では，主に，Valid-A による画像処理プログラミングとシステムの性能評価について述べる．まず AMP システムの概略について述べた後，Valid-A による具体的な画像処理プログラミングを細線化処理を例として述べる．さらに，Valid-A を用いて画像処理を記述した場合，AMP で高い処理能力を発揮することを，いくつかの画像処理アルゴリズムに対するソフトウェアシミュレーションによって示す．

1. はじめに

画像処理においては，ひとつひとつはさほど複雑ではないが，数万，数十万におよぶ画素に対して同様の計算を行うことが多く，膨大な計算量による処理時間の増大が大きな問題となる．画像処理では高い並列性が期待されるため，この問題を解決するために様々な画像処理用並列プロセッサシステムが開発されている[1)~9)]．しかし，すべての PE を同期的に動作させる必要があるシステムでは，**"プログラミングが困難である"**，**"プロセッサの稼働率が低くなりがちである"**，**"処理の柔軟性にかける"** といった問題が生じる．

そこで，筆者らは各 PE をデータ駆動制御に基づき非同期に動作させ，高速，高能率な処理が可能で，単純な画像処理のみならず画像理解のための処理にも対応できるような柔軟さを持つ自律型非同期超並列プロセッサ AMP の開発を行っている[10),11)]．また，この AMP システム上のプログラミング言語として，関数型言語 Valid-A[12)] の開発も同時に進めている．Valid-A は，AMP の特性を損なわず，分かりやすく単純な画像処理から画像理解のための処理まで柔軟な記述ができることを設計の基本方針としている．

本論文では，AMP システムの概略について述べた後，AMP システムにおける画像処理プログラムの記述方式，特に，細線化処理のような繰り返し処理に同期を必要とする処理のプログラミングについて述べる．また，ソフトウェアシミュレータによる AMP の性能評価について報告する．

2. AMP システムの概要

2.1 AMP システムの基本構成

AMP の基本構成は，**図1**のように多数の PE を通信ネットワークで結合したものである．各 PE は外部メモリ（共有メモリ）を持たず，プログラムコードと画像などのデータはそれぞれ PE の命令メモリ **IM** (Instruction Memory) とオペランドメモリ **OM** (Operand Memory) に分散して格納する．また，各 PE はデータフロー制御を用いて非同期に動作し，これにより柔軟で高能率な並列処理を実現する．これらの PE は，循環パイプラインアーキテクチャを基に設計しており，一つの PE で論理的に複雑の PE として動作することが可能である（物理的に一つの PE を物理 PE，その上で論理的に実現される PE を論理 PE と呼ぶ）．一般に画像処理においては，一つの論理 PE に一画素を割り当てて処理を行うことが考えられる．

画素のデータはあるメモリに一括して格納するのではなく，各 PE に分散して保持するので，必要なデータは PE 間通信により各論理 PE に転送する．同じ物理 PE 内に割り当てられる論理 PE 間の通信は，PE 内の循環パイプラインだけで行われるので極めて効率がよく，この機能により多重解像度や階層構造のデータ構造を用いた画像処理[13)]を容易にかつ効率

† Programming System and Performance Evaluation of AMP by NORIYASU YAMAMOTO, NAOYUKI TSURUTA, RIN-ICHIRO TANIGUCHI and MAKOTO AMAMIYA (Department of Information Systems, Graduate School of Engineering Sciences, Kyushu University).

†† 九州大学大学院総合理工学研究科

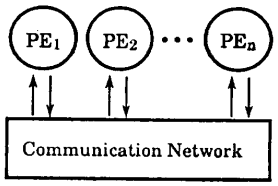


図 1 AMP の基本構成
Fig. 1 Basic structure of AMP.

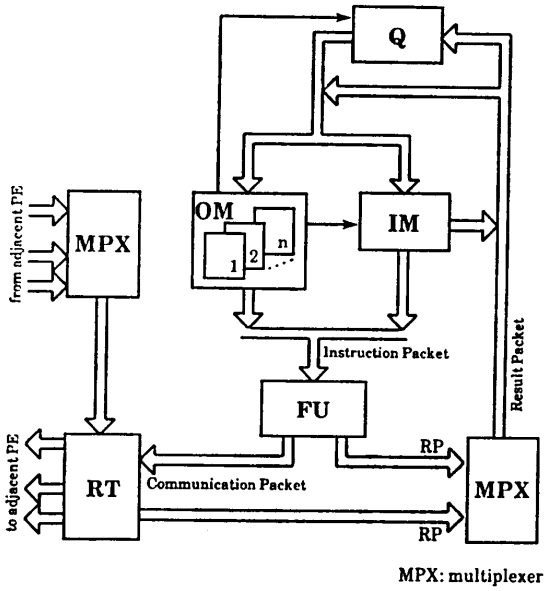


図 2 AMP 用 PE の構造
Fig. 2 Configuration of PE.

よく実現できる[10),11)].

AMP では，10^4 以上の PE が実装できるようなシステム構成を想定しているため，通信ネットワークはできる限り簡単なものが望ましく，ここでは格子状のものを考える．

2.2 物理 PE の構成

図 2 に示すように，AMP 用 PE の構造は循環パイプラインアーキテクチャを基に構成しており，実際に命令を実行する演算部 **FU** (Function Unit)，命令コードと次の命令へのリンク情報を持つ命令メモリ部 **IM**，オペランドデータを格納するオペランドメモリ部 **OM**，同期とデータストリームの順序保持を行うキューバッファ **Q** (Queue buffer)，他の PE への通信パスを管理するルーティング部 **RU** (Routing Unit) が循環パイプラインで結ばれている[10),11)].

この PE 構成の特徴は，**OM** が大容量のレジスタファイルであり，複数のブロックに分割されていることである．この各ブロックが一つの論理 PE に対応し，ブロックを切り換えることで複数の論理 PE のプログラムを実行することができる．また，**IM** は演算命令コードおよび FU での演算結果の行き先，つまり次のサイクルで実行される命令のアドレスを保持している．PE 間通信では，この FU での演算結果の行き先が他の論理 PE となるだけであり，また，他の物理 PE から送られた通信パケットと自 PE 内での通信パケットは基本的には区別されない．すなわち，バッファリングなど PE 間通信のための特別な処理は何ら行われないので，非常に効率が高いものとなっている[14)]．

3. AMP における画像処理プログラミング

3.1 プログラミング言語 Valid-A

AMP のプログラミング言語 Valid-A は，データフローマシン用の記号処理向け関数型言語 Valid[15)]を基として，AMP ハードウェアの特性を損なわず，分かりやすく柔軟なプログラミングが可能であるように設計したものである．Valid-A で新たに導入した機能は以下のとおりである．

（1） 論理 PE の物理 PE へのマッピング．
（2） 関数を各論理 PE に割り付けるための関数配列．
（3） 主にルックアップテーブルとしての小規模配列．
（4） PE 間通信（および PE 間通信を簡単に記述するためのリモート変数）．
（5） 画像のピラミッド構造を効率的に処理するための記述．

詳細については文献 12) に譲ることとし，本論文では次節で述べるプログラミング例でその概要を示すにとどめる．

3.2 Valid-A による画像処理プログラミング

画像処理においては，各画素に対して繰り返し処理を行うものが多い．一般に，繰り返し処理は，すべての画素での計算が終了しないと次の繰り返しへ移れない同期型処理と，そのような拘束のない非同期型処理に分けることができる．例えば，画像の細線化処理は，領域を均一に細線化するために，各画素の繰り返し処理を同期をとって行わなければならず，これは同期型処理に属する．これに対して，画像の畳み込み演算などは，各画素の繰り返し処理を非同期に行えるため，非同期型処理に属する．

前述のように，AMP は各画素に対するプログラム

を非同期に実行するので，非同期型処理は効率の良いプログラムが簡単に記述できる[12]が，同期型処理は，同期のとり方によっては効率の低下を招く恐れがある．今回は，この同期型処理のプログラミングについて述べる．他の画像処理のプログラミング例については，参考文献12)を参照されたい．

3.2.1 AMP上での同期型処理のプログラミング

AMP上で同期型処理を効率良く実行するためには，同期はできるだけ局所的にとることが必要である．また，処理が終った論理PEは，速やかに停止させることが望ましい．以下では，二値（白画素：背景，黒画素：オブジェクト）画像の細線化処理を例にAMP上での同期型処理のプログラミングについて示す．

1サイクルの細線化処理における各黒画素の削除は，以下の手順で，画素の周囲8近傍の画素の値を参照して局所的に処理できる．

（i） 画素の周囲8近傍に画素の値を送信し，また8近傍の画素の値を得る．

（ii） 8近傍の値をもとに，削除可能な画素[16]を削除する．

細線化は，この局所処理を黒画素の削除が行われなくなるまで繰り返せばよいのであるが，画素削除の偏りをなくすため，各画素に対する局所処理は同期をとって行わなければならない．また，画像全体で，細線化のための画素削除がなくなった場合は，各画素の細線化処理を停止させるという同期も必要である．

AMPは非同期マシンであるので，このような同期をとる必要がある処理では，同期のためのプログラムを各局所処理に追加することになる．一つの方法は，PE間通信を利用して，**図3**(a)の矢印のように画像の左上から右下まで，同期のための signal を伝搬させるプログラムが考えられる．具体的には，各画素の処理において1サイクルの細線化処理の後，黒画素の削除が行われたかどうかを伝搬してきた signal にのせてさらに伝搬させる．これにより，一回の細線化により削除された画素があるかどうかのデータが右下端に集計される．削除された画素があれば細線化続行とし，もう一度左上端より値を伝搬させる．削除された画素がなければ，処理終了の信号を伝搬させて各画素における細線化プログラムを終了させる．この手法を **Program A** とする．

この方法では，同期のための値伝搬を画像の横方向に逐次に行うため効率が悪く，$n\times n$ 画素の画像に対し $O(n)$ の計算時間が必要になる．しかし，細線化処理の場合は，以下の手法により局所的に同期をとることが可能である．

（1） 細線化処理の同期

前述のように，黒画素の削除は，隣接する8近傍の画素の値を参照して実行される．すなわち，図3 (b) の矢印のように各画素が一回の細線化においてそれぞれの隣接画素に値を伝搬するだけでよい．よって同期は自然にとれていることになり，あえて同期のための値伝搬を行う必要はない．

（2） 細線化処理の終了

画像全体での細線化の終了は，もともと0である画素や，細線化により削除され0となった画素，細線化処理によって削除される可能性のない画素[16]が隣接画素に停止信号を送って繰り返し処理を停止することで行うことができる．停止信号を受け取った画素では，以後その停止した画素とは通信を行わない．すべての画素の繰り返し処理が停止した時，画像の細線化処理が完了したことになる．

この手法を **Program B** とし，プログラムリストを**図4**に示す．

Program B では，細線化の同期と終了判定が局所的に行えるので，画像が大きくなっても物理PEに対する画素の割付数が同じであれば，処理時間の増加は起こらない．よって，**Program B** は **Program A** に比べて効率の良い方法であるといえる．

Program B の手法は，削除可能な画素の定義を変更することで，二値画像の縮退[17]にそのまま適用することが可能なだけでなく，反復型局所並列処理の距離変換[18),19)]や弛緩法[20)~22)]にも適用できる．

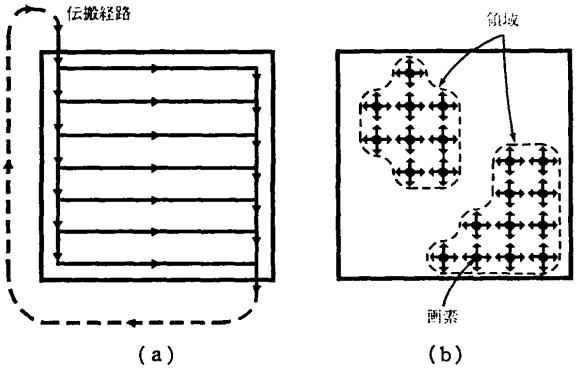


図3 同期をとるアルゴリズム
Fig. 3 Algorithm for synchronization.

```
define PE_size = 100:100;
array of function thinning[256][256](pix)
return(integer);
for each function[i][j]{
/* Macro definition */
macro Comp(x1,x2,x3,x4,x5,x6,x7,x8)
return(integer) =
 let xx1 = if (x1=0) then 0 else 1,
     xx2 = if (x2=0) then 0 else 1,
     xx3 = if (x3=0) then 0 else 1,
     xx4 = if (x4=0) then 0 else 1,
     xx5 = if (x5=0) then 0 else 1,
     xx6 = if (x6=0) then 0 else 1,
     xx7 = if (x7=0) then 0 else 1,
     xx8 = if (x8=0) then 0 else 1
 in  xx1<<7|xx2<<6|xx3<<5|xx4<<4
       |xx5<<3|xx6<<2|xx7<<1|xx8,
macro Neighbor(z1,z2,z3,z4,z5,z6,z7,z8,y)
 return(integer,integer,integer,integer,
        integer,integer,integer,integer) =
 let
  a1=if (z1=1) then thinning[i][j-1].y else z1,
  a2=if (z2=1) then thinning[i+1][j-1].y else z2,
  a3=if (z3=1) then thinning[i+1][j].y else z3,
  a4=if (z4=1) then thinning[i+1][j+1].y else z4,
  a5=if (z5=1) then thinning[i][j+1].y else z5,
  a6=if (z6=1) then thinning[i-1][j+1].y else z6,
  a7=if (z7=1) then thinning[i-1][j].y else z7,
  a8=if (z8=1) then thinning[i-1][j-1].y else z8
 in  (a1,a2,a3,a4,a5,a6,a7,a8),
/* Main body of thinning program */
if (i>0 & i<255 & j>0 & j<255) then
thinning[i][j](pix) =
 let Mask = array(256,0,1,0,2,.....),
     x01 = thinning[i][j-1].pix,
     x02 = thinning[i+1][j-1].pix,
     x03 = thinning[i+1][j].pix,
     x04 = thinning[i+1][j+1].pix,
     x05 = thinning[i][j+1].pix,
     x06 = thinning[i-1][j+1].pix,
     x07 = thinning[i-1][j].pix,
     x08 = thinning[i-1][j-1].pix
 in if (pix=1) then
  {for (xa1,xa2,xa3,xa4,xa5,xa6,xa7,xa8)
    init (x01,x02,x03,x04,x05,x06,x07,x08)
    do let y1 = Mask[Comp(xa1,xa2,xa3,xa4,
                          xa5,xa6,xa7,xa8)],
          (xb1,xb2,xb3,xb4,xb5,xb6,xb7,xb8)
             = Neighbor(xa1,xa2,xa3,xa4,
                 xa5,xa6,xa7,xa8,y1)
       in if (y1=1) then
            let
             y2 = Mask[Comp(xb1,xb2,xb3,xb4,
                    xb5,xb6,xb7,xb8)],
              (xc1,xc2,xc3,xc4,xc5,xc6,xc7,xc8)
                 = Neighbor(xb1,xb2,xb3,xb4,
                     xb5,xb6,xb7,xb8,y2)
            in if (y2=1) then
                recur(xc1,xc2,xc3,xc4,
                      xc5,xc6,xc7,xc8)
              else if (y2=0) then return 0
            else return 1
           else if (y1=0) then return 0
      else return 1                    }}
```

図 4 Program B [細線化]
Fig. 4 Program B [Thinning].

4. AMP システムの評価

本章では，前章で述べた Valid-A を用いて画像処理を記述した場合，AMP がどの程度の処理能力を発揮するかを示す．ここでは，Valid-A で記述した画像処理プログラムをハンドコンパイルし，そのオブジェクトコードをシミュレータ上で実行して AMP システムの性能評価を行う．シミュレータは，レジスタトランスファレベルで AMP システムのシミュレーションを行うことができるものである．

AMP システムの評価では，まず次節で述べるシステムパラメータを設定する必要がある．その後で，設定したパラメータ値における AMP システムの評価を行う．

4.1 システムパラメータの設定

まず，AMP システムの評価では，次の三つが重要なパラメータとなる．

(1) 一つの物理 PE にマッピングする論理 PE の数

一つの物理 PE にマッピングする論理 PE の数は，少なすぎると PE 間通信のオーバヘッドが大きくなり稼働率が低下する．一方，多すぎると稼働率は上昇するが処理の並列度が低くなるというトレードオフがある．

(2) PE 間の通信速度

物理 PE に割り当てる論理 PE の数が多ければ，すなわち，通信以外の物理 PE の処理量が多ければ，PE 間通信によるオーバヘッドの影響をある程度除くことができる．したがって，論理 PE のマッピング数との兼ね合いで，PE 間通信速度をどの程度にすれば良いか求める必要がある．

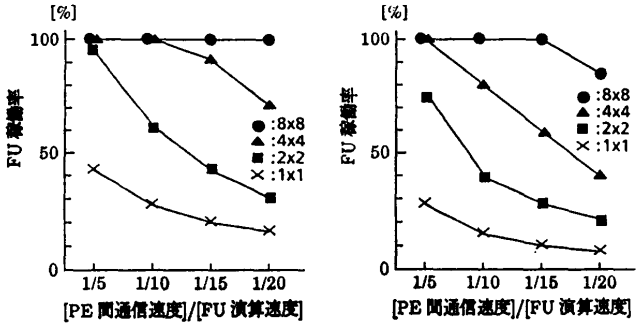
(3) メモリアクセス速度と FU 演算速度の比

AMP は基本的に二つのオペランドが到着しないと FU で一つの命令を実行できない．したがって，FU の稼働率を上げるため，メモリアクセス速度は FU 演算速度よりも高速でなければならない．したがって，メモリアクセス速度と FU 演算速度の比をどの程度にすれば良いかを決める必要がある．

ここでは，AMP の FU 演算速度を 100[ns] とし，PE 間ネットワークを 4 隣接格子網と仮定する．上記のパラメータを決定するために，3×3 convolution およびさらに物理 PE 間の通信が多い 5×5 convolution* についてシミュレーションを行った．その結果を図 5〜8 に示す．

図 5〜8 は，論理 PE の割り付け数をパラメータとして，[PE 間通信速度]/[FU 演算速度] に対する FU の稼働率（図 5）と処理速度（図 6），[メモリア

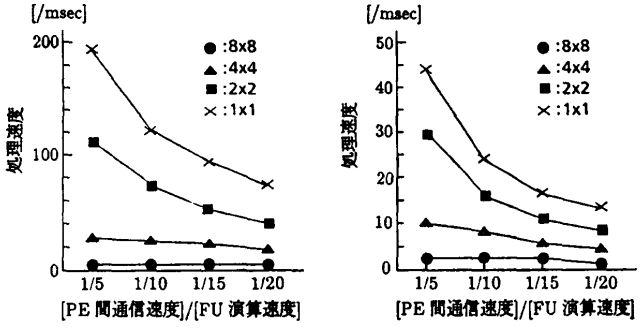
* 3×3 convolution では距離が 2 以内の周囲 8 近傍の画素との通信であるのに対して，5×5 convolution では 3×3 の場合に加えて距離が最大 4 の画素とも通信を行うので，ある物理 PE にマッピングされた論理 PE から他の物理 PE の論理 PE への通信が多くなる．



(a) 3×3 畳み込み (b) 5×5 畳み込み
(a) 3×3 convolution. (b) 5×5 convolution.

図 5 [PE 間通信速度]/[FU 演算速度] に対する FU 稼働率 (ただし, FU 演算速度=100 [ns], メモリアクセス速度=50 [ns].)

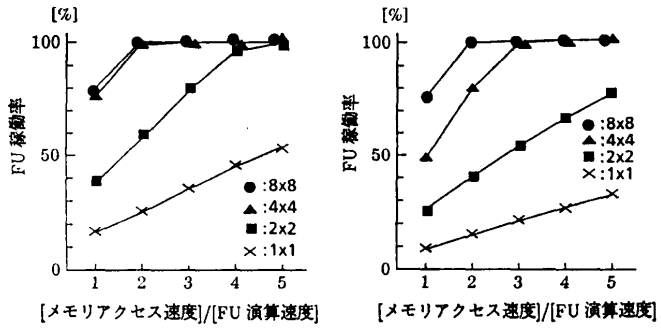
Fig. 5 The activity of FU plotted for [Communication speed]/[FU execution speed].



(a) 3×3 畳み込み (b) 5×5 畳み込み
(a) 3×3 convolution. (b) 5×5 convolution.

図 6 [PE 間通信速度]/[FU 演算速度] に対する処理速度 (ただし, FU 演算速度=100 [ns], メモリアクセス速度=50 [ns].)

Fig. 6 Speed plotted for [Communication speed]/[FU execution speed].



(a) 3×3 畳み込み (b) 5×5 畳み込み
(a) 3×3 convolution. (b) 5×5 convolution.

図 7 [メモリアクセス速度]/[FU 演算速度] に対する FU 稼働率 (ただし, FU 演算速度=100[ns], PE 間通信速度=1[μs].)

Fig. 7 The activity of FU plotted for [memory access speed]/[FU execution speed].

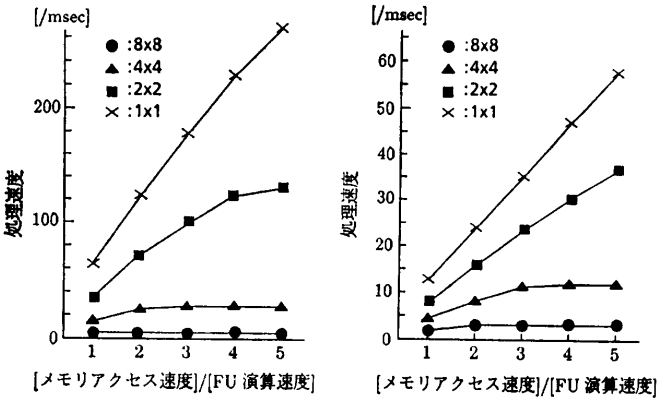
クセス速度]/[FU 演算速度] に対する FU の稼働率 (図7) と処理速度 (図8) を示したものである.

まず, 一つの物理 PE にマッピングする論理 PE の数について考える. 図5に示されているように, 論理 PE の割り付け数が少ない場合は, 物理 PE 間の通信が多いうえに, 物理 PE の処理量が少ないので通信のオーバヘッドの影響が顕著に現れるため稼働率が低い. これに対して論理 PE の割り付け数が多い場合は逆に, PE 間通信の多くが論理 PE 間で行われ, また各物理 PE の処理量が多いので通信のオーバヘッドの影響が出にくいため稼働率が高い. そこで, 論理 PE の割り付け数は, 5×5 convolution 程度の通信量でも FU の稼働率が 100% 近く効率の良い 8×8 を標準値とする.

次に, PE 間の通信速度を決める. 図5, 6より, PE 間通信速度は速いほど良いということになるが, あまり速くしても物理 PE の割り付け数によっては FU の稼働率が飽和してしまう. 論理 PE の割り付け数が 8×8 の場合, 図5のグラフでは [PE 間通信速度]/[FU 演算速度] を 1/15 より小さくしても, 稼働率は 100% 近くで飽和してしまい, 処理速度が変化しなくなる. 通信量が増加すると通信のオーバヘッドのため稼働率が低下するので, ここでは通信量が 5×5 convolution よりも多い場合を想定し, また, 実際に実現可能な数値として [PE 間通信速度]/[FU 演算速度] を 1/8〜1/10 程度とする.

最後に, メモリアクセス速度と FU 演算速度の比を決める. [メモリアクセス速度]/[FU 演算速度] が大きくなると, FU の1サイクルに対してメモリのオペランドマッチングの回数が増える. このため, 二つのオペランドが揃って実行可能となる命令が増えるので, 稼働率が上昇し (図7), 処理速度が速くなる (図8). しかし, 論理 PE の割り付け数が 8×8 の場合, 図7より [メモリアクセス速度]/[FU 演算速度] が2以上では FU 稼働率が飽和しており, 求める比は2が適当であると考えられる.

以上より, FU 演算速度を 100 [ns], メモ



(a) 3×3 畳み込み
(a) 3×3 convolution.

(b) 5×5 畳み込み
(b) 5×5 convolution.

図 8 [メモリアクセス速度]/[FU 演算速度] に対する処理速度 (ただし, FU 演算速度=100[ns], PE 間通信速度=1[μs].)
Fig. 8 Speed plotted for [memory access speed]/[FU execution speed].

リアクセス速度を 50[ns], PE 間通信速度を 0.8～1.0[μs], ネットワークを 4 隣接格子網, 物理 PE への論理 PE の割り付け数を 8×8 とするのを標準的なパラメータ設定値とする.

4.2 各種画像処理による性能評価

ここでは前述の標準パラメータを用いて, 各種画像処理アルゴリズムの計算時間の評価を行う. この評価も 4.1 節と同様のシミュレータを用いた.

表 1 は, 各種の同期を必要としない局所的な画像処理における所用時間を示している. このような局所処理では AMP は極めて高い性能を示す. 表中で, AMP システムは逐次マシン (68020 CPU, 16.6 MHz) の平均 1.45×10^5 倍の性能を示している. 逐次マシンが 1 CPU であるのに対して, AMP システムでは 4096 個の PE を使用している点を考慮すると, PE 当たりの性能比は約 35 倍であり, さらに, 逐次マシンが公称 1 MIPS であるのに対して, AMP システムの各 PE はピーク値で約 10 MIPS であることを考慮すると, 比は 3.5 倍になる. 逐次マシンのプログラムが FORTRAN で書かれており, 最適化が十分なされていないことを差し引いても, この値は AMP システムが高能率な並列処理を実現することを示すものである.

表 2 は, 同期を必要とする画像処理の例として, AMP マシンで 512×512 画素の画像に対する 3.2 節で述べた一回の細線化を行ったときに要する時間を示している. AMP では一つの物理 PE に 8×8 の画素を割り付けているので, 全体で 4096 個の物理 PE を使用したものとしている. この場合, 前述の **Program B** は, **Program A** の 3 倍程度の処理速度が得られている.

また, **図 9** に画像の大きさを変化させたときの **Program A** と **Program B** の処理時間の比較を示した. ここでは, 一つの物理 PE には 8×8 の画素を

表 1 画像処理アルゴリズム (局所処理) に対する AMP システムの処理時間
Table 1 Execution time of asynchronous image processing algorithms.

画像処理アルゴリズム	処理時間 [msec]	
	AMP システム	逐次マシン*
3×3 convolution	0.141	20.4×10^3
5×5 convolution	0.397	54.4×10^3
binary shrinking	0.333	6.5×10^3
Roberts's edge[23]	0.051	8.0×10^3
Prewitt's edge[24]	2.34	293×10^3
3×3 max filter	0.186	20.9×10^3
Edge-pres smoothing[25]	0.48	135.2×10^3
Dither[26]	0.035	6.6×10^3

ただし, 逐次マシンは 68020 CPU (16.6 MHz) を使用し, 512×512 画素の画像を処理した. また, AMP システムのパラメータは, FU 演算速度=100[nsec], メモリアクセス速度=50[nsec], PE 間通信速度=800[nsec], PE 数=4096 個を仮定している.
* 一部 SPIDER[27] を利用.

表 2 一回の細線化の処理時間 [msec]
Table 2 Execution time in one iteration of thinning process.

AMP システム		逐次マシン
Program A	Program B	68020 CPU (16.6 MHz)
1.2	0.38	6.5×10^3

ただし, AMP システムのパラメータは, FU 演算速度=100[nsec], メモリアクセス速度=50[nsec], PE 間通信速度=800[nsec], PE 数=4096 個を仮定している.

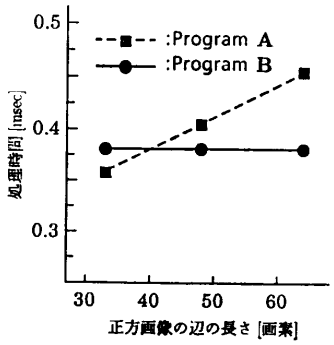


図 9 正方画像の辺の長さに対する細線化の処理時間
Fig. 9 Execution time plotted for the size of image.

割り付けるものとし，画像の大きさに必要な物理 PE を使用するものとした*．グラフより，**Program A** では，画像の辺の長さに対してほぼリニアに処理時間が増加しているが，**Program B** では画像の辺の長さが変化してもほぼ一定の処理時間となっている．これは，**Program B** では PE の数に関係なく同期のためのオーバヘッドが一定であることを示しており，**Program A** よりも効率の良い手法であることを実証している．

5. おわりに

本論文では，主に AMP システムにおける同期を必要とするような画像処理プログラムの記述方法および AMP の性能評価について述べた．

AMP のような非同期の超並列プロセッサでは，同期を必要としない局所的な画像処理は，極めて高い性能が期待できる．本論文ではその期待される具体的数値をシミュレーションにより示し，高い性能が得られることを示した．また，AMP 上で同期処理を行うには，同期は局所的にとり，処理が完了した論理 PE は速やかに停止させることが望ましいが，その場合にも高効率な処理が実現されることを細線化処理を例に示した．

今後は，PE 間通信が近距離のみならず遠距離の場合のシミュレーション評価と，他の同期を必要とする処理のプログラムの記述およびそのシミュレーション評価を行っていく予定である．

参考文献

1) Kung, H.T.: On the Implementation and Use of Systolic Array Processors, *Proc. Int. Conf. Computer Design: VLSI in Computer* (1983).
2) Kidode, M. et al.: High-speed Image Processor: TOSPIX-II, *Evaluation of Multicomputers for Image Processing* (Uhr, L. et al. eds.), Academic Press (1986).
3) 福島ほか：マルチマスクオペレーションを効率よく実行する画像処理用 LSI-IPS のアーキテクチャ，情報処理学会論文誌，Vol. 26, No. 2, pp. 239-246 (1985).
4) Duff, M.J.B.: Parallel Processors for Digital Image Processing, *Advances in Digital Image Processing* (Strucki, P. ed.), Plenum (1979).
5) Flanders, P.M. et al.: Efficient High Speed Computing with Distributed Array Processor, *High Speed Computer and Algorithm Organization* (Kuck, D.J. et al. eds.), Academic, Press (1977).
6) Hillis, W.D.: *The Connection Machine*, MIT Press (1985).
7) Iwashita, M. et al.: Data Flow Chip ImPP and Its System for Image Processing, *Proc. ICASSP '86* (1986).
8) Siegel, H.J.: PASM: a Reconfigurable Multimicro-computer System for Image Processing, *Languages and Architectures for Image Processing* (Duff, M.J.B. et al. eds.), Academic Press (1981).
9) Kushner, T. et al.: Image Processing on ZMOB, *IEEE Trans. Comp.*, Vol. C-31, No. 10 (1982).
10) 谷口，雨宮：画像処理と理解のための自律型非同期超並列プロセッサ AMP，「画像理解の高度化と高速化」シンポジウム講演論文集，pp. 53-58 (1989).
11) Taniguchi, R. and Amamiya, M.: AMP: an Autonomous Multi-Processor for Image Processing and Computer Vision, *Proc. 10th ICPR*, Vol. II, pp. 497-500, IEEE Computer Society Press (1990).
12) 山元，堀田，谷口，雨宮：画像処理用超並列プロセッサ AMP のプログラミング言語 Valid-A について，第 40 回情報処理学会全国大会論文集 (II)，pp. 547-548 (1990).
13) 谷口，河口：画像処理のためのデータ構造，別冊 O puls E 画像処理アルゴリズムの最新動向，pp. 73-83 (1986).
14) Takahashi, N. et al.: A Data Flow Processor Array System Design and Analysis, *Proc. 10th ISCA* (1983).
15) 長谷川，雨宮：データフローマシン用関数型高級言語 Valid，電子情報通信学会論文誌 D, Vol. J 91-D, No. 8, pp. 1532-1539 (1988).
16) 横井，鳥脇，福村：標本化された2値図形のトポロジカルな性質について，信学論，Vol. 56-D, No. 11, pp. 662-669 (1973. 11).
17) 横井，鳥脇，福村：2値図形収縮のための逐次型アルゴリズムについて，信学論，Vol. J 62-D, No. 8, pp. 537-542 (1979. 4).
18) Toriwaki, J. and Yokoi, S.: Distance Transformation and Skeleton of Digitized Picture with Applications, *Progress in Pattern Recognition* (Rosenfeld, A. and Kanal, L. eds.), Vol. 1, pp. 187-264, North-Holland, N.Y. (1981).
19) 横井：画像処理アルゴリズムの最新動向2 幾何学的特徴の処理 (1)，O plus E, No. 68, pp. 120-126 (1985. 7).
20) 坂上：画像処理における反復演算の応用，情報

* すなわち，一つの物理 PE の処理量が同一になるようにしている．

処理, Vol. 23, No. 7, pp. 641-650 (1982).
21) Zucker, S. W., Hummel, R. A. and Rosenfeld, A.: An Application of Relaxation Labeling to Line and Curve Enhancement, *IEEE Trans. Comput.*, Vol. C-26, No. 4, pp. 394-403 (1977. 4).
22) Peleg, S.: A New Probabilistic Relaxation Scheme, *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-2, No. 4, pp. 362-369 (1980. 7).
23) Roberts, L. G.: Machine Perception of Three-Dimensional Solids, *Optical Electo-optical Processing of Information*, pp. 159-197, MIT Press (1965).
24) Prewitt, J. M. S.: Object Enhancement and Extraction, *Picture Processing and Psycho-pictorics*, pp. 75-149, Academic Press (1970).
25) 浅野, 横矢, 大久保, 田中: 領域解析に適したフィルタの提案とその比較, 信学技報, PRL 77-13 (1977).
26) Jarvis, J. F., Judice, C. N. and Nenke, W. H.: A Survey of Techniques for the Display of Continuous-Tone Pictures on Bilevel Displays, *Comp. Graph. Imag. Proc.*, Vol. 5, pp. 13-40 (1976).
27) 田村ほか: SPIDER USER'S MANUAL, 電子技術総合研究所 (1980. 9).

(平成2年8月29日受付)
(平成3年2月12日採録)

山元 規靖 (学生会員)

昭和 38 年生. 平成元年長崎大学工学部電気工学科卒業. 平成3年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了. 現在, 同専攻博士後期課程在学中. 並列画像処理システムの研究に従事.

鶴田 直之 (正会員)

昭和 39 年生. 昭和 63 年九州大学工学部情報工学科卒業. 平成2年同大学院総合理工学研究科情報システム学専攻修士課程修了. 現在, 同専攻助手. ニューラルネットワークを用いた画像処理・理解およびその並列実行環境についての研究に従事.

谷口倫一郎 (正会員)

昭和 30 年生. 昭和 53 年九州大学工学部情報工学科卒業. 昭和 55 年同大学院工学研究科修士課程修了. 同年九州大学大学院総合理工学研究科情報システム学専攻助手. 平成元年より同助教授. 工学博士. 画像理解, 画像処理システム, 並列処理システムの研究に従事. 電子情報通信学会, 人工知能学会各会員.

雨宮 真人 (正会員)

昭和 17 年生. 昭和 42 年九州大学工学部電子工学科卒業. 昭和 44 年同大学院工学研究科修士課程修了. 同年日本電信電話公社武蔵野電気通信研究所入所. 以来, プログラミング言語・処理系, 自然言語理解, データフロー・アーキテクチャ, 並列処理, 関数型／論理型言語, 知能処理アーキテクチャ, 等の研究に従事. 現在九州大学大学院総合理工学研究科情報システム学専攻教授. 工学博士. 電子情報通信学会, ソフトウェア科学会, 人工知能学会, IEEE, AAAI 各会員.