並列処理用言語ＩＬＯＳの開発

渡辺慎哉　　宮本衛市　（北海道大学　工学部）

1. はじめに

近年、処理の高速化を目指して、複数のプロセッサを用いた並列マシンが数多く提案されており、そのうちの幾つかは実用段階にある[1),2)]。このような実際の並列マシン上に並列処理を行なう為の高級言語を実現する場合、そのターゲットマシンの持つ能力をどの程度引き出すことができるかが重要なポイントとなる。特に、言語仕様とハードウェアとの適合性は処理効率を大きく左右する要因の一つである。

現在、北海道大学汎用シミュレータ施設には連続系シミュレーションの高速計算を目的として開発された並列プロセッサシステムＰＰＡ（Parallel Processor Array）が導入されている。これは、３２台のＣＰＵが共有メモリを介して１次元循環アレイ状に並べられたものであり、ホストマシンであるVAX-11/780と接続され、連続系シミュレーション用の専用サブシステムとして使用されている[1)]。

我々は先に、ＰＰＡのより自由度の高い使用と、並列処理における問題点の考察を目的として、試作的な並列処理用言語ＩＬＯＳをＰＰＡ上に開発し、その性能評価を行ってきた[3)-7)]。その際、データ転送や同期によるオーバーヘッドの為に、並列化に見合うほどの計算速度の向上は得る事が出来なかった。その要因の一つとして、ＩＬＯＳの言語仕様とターゲットマシンとの適合性という問題が考えられた。そこで、今回我々は、旧ＩＬＯＳにおける言語仕様上の問題点およびターゲットマシンの機能を考慮し、数値計算、特に連続系シミュレーションの計算効率、記述性の向上を目的とした並列処理用言語を考案し、現在そのインプリメンテーションを行っている。

2. 非同期実行と同期実行

複数のプロセスが並列に処理を行なう場合、何らかの形で互いにデータを交換し合う必要がある。その場合、意味のあるデータを授受する為には同期操作を行なわなければならない。非同期実行は、各プロセスがそれぞれデータ通信が必要になった時点で、セマフォ等により同期をとりながら通信を行なう方法であり、オペレーティング

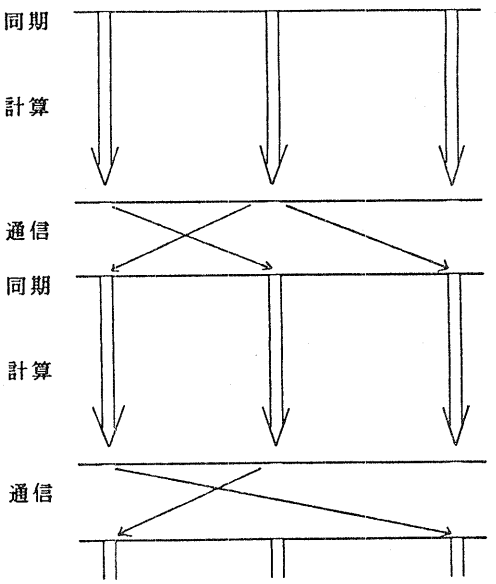

図１　同期実行方式

システムなどのような複雑な同期問題を記述することがきる。しかし、大量のデータを頻繁にやりとりする場合にはオーバーヘッドを少なくすることは難しい。それに対して同期実行は、図２に示すように全プロセスの実行を同期的に行ない、各実行の間にプロセス間でデータを交換し合う方法である。この方法は、各プロセスが比較的似かよった処理をする場合に有効であり、適用分野によっては通信のオーバーヘッドを減らすことが可能である。しかし、各プロセスの実行時間が大きく異なっている場合には無駄が生じやすく、また、複雑な問題の記述という点では、非同期方式の方が有利である。

ＩＬＯＳでは、数値計算の処理効率と記述性の向上という立場から、基本的にプロセスは同期的に実行されるが、より複雑な問題に対応する為に、その中に非同期的な通信機能を付け加えた柔軟な構造を有している（図２）。

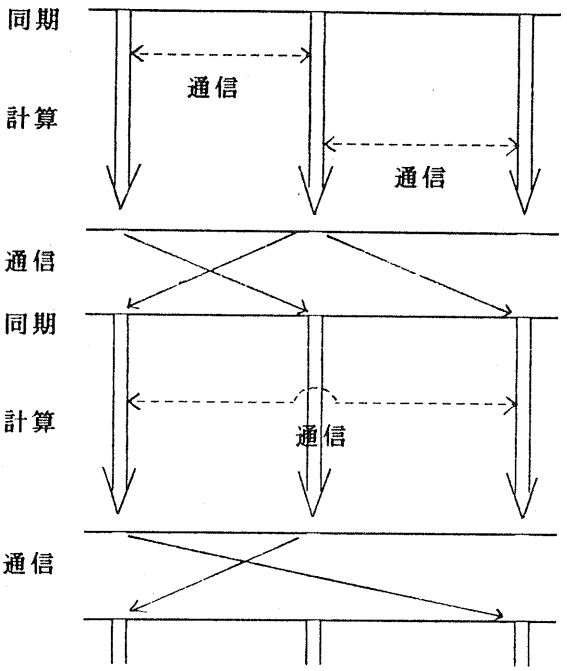

図２　ＩＬＯＳの実行方式

3.　ターゲットマシンの構造

ターゲットマシンであるＰＰＡの概略図を図３に示す。図のＭＰ（マスター・プロセッサ）、ＳＰ（スレーブ・プロセッサ）は共にＤＥＣの PDP-11 相当のプロセッサに拡張命令と浮動小数点演算装置を付加したものとなっている。特にＳＰには、ベクトル演算命令が付加され、高速な数値計算を可能としている。各ＳＰは、共有メモリを介して３２台が１次元循環アレイを構成し、２台のＭＰがそれらの実行管理やデータの受け渡しを行なっている。また、この２台のＭＰはＤＭＡを介してホストマシンであるVAX-11/780に接続されており、プログラムのロードや入出力操作などは、これを通じて行なうことになっている。

図からわかるように、隣接ＳＰ間の通信は互いの間にある共有メモリを介して行なうことができるが、遠隔ＳＰ同志はＭＰを中継して通信し合う必要がある。これは、一般的なプロセッサ・アレイ・システムに共通の性質であり、この中継プロセッサによる通信が頻繁に発生する場合にはオーバーヘッドが非常に大きくなるので、このようなシステムに対して効率のよいインプリメントを行なうためには、この性質を考慮に入れて言語を設計する必要がある。

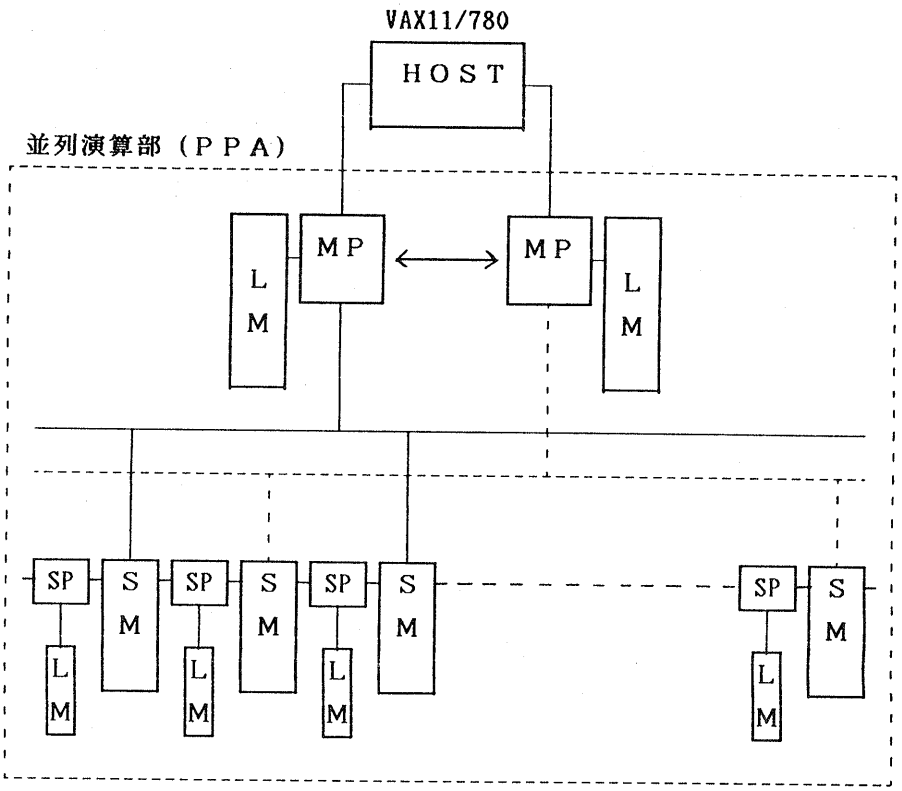

図３　並列演算部

4.　ＩＬＯＳの言語仕様と特長

ＩＬＯＳの構文は、Pascalに並列記述部を付加したものとなっている。図４にＩＬＯＳの構造の概略を示した。

4.1 プロセス及びプロセスの同期実行

ＩＬＯＳにおいては、プロセスが並列の単位となる。プロセスは、手続き宣言部のあとに、次のような形式で宣言する。

```
process プロセス名；
  宣言部
begin
  ・・・・
end;
```

このプロセスは、図４の主プログラム中で、

execute(プロセス名の並び)

を実行することにより、並列に起動をかけることができる。例えば、図５ａのプログラムでは、execute(P1, P2) を実行した時点でプロセスＰ１とＰ２は同時に起動され main は実行を停止する。その後、両プロセスの実行が終了した時点で main が再開されることになる。これを模式的に示したのが図５ｂである。また、プロセスは配列的に宣言することが可能であり、例えば次のように宣言する。

process プロセス名[1..5];

この場合、同じ処理内容を持つ５個のプロセスを一度に宣言したことになる。

また、この execute(...) をループ構造の中に入れることにより、同期的な反復実行が可能となる。反復実行の合間にプロセス間でデータの交換を行なうためには、次のようにtransmit文を用いる。

```
transmit
  P1.x -> P2.x;
  P2.x -> P1.x
end
```

この例では、プロセスＰ１の旧ｘの値がプロセスＰ２の新ｘへ、プロセスＰ２の旧ｘの値がプロセスＰ１の新ｘへ渡されたことになる。このように、execute と transmit を組み合わせることにより、２節で述べた同期実行を実現することができる。図６ａ、図６ｂはその例である。

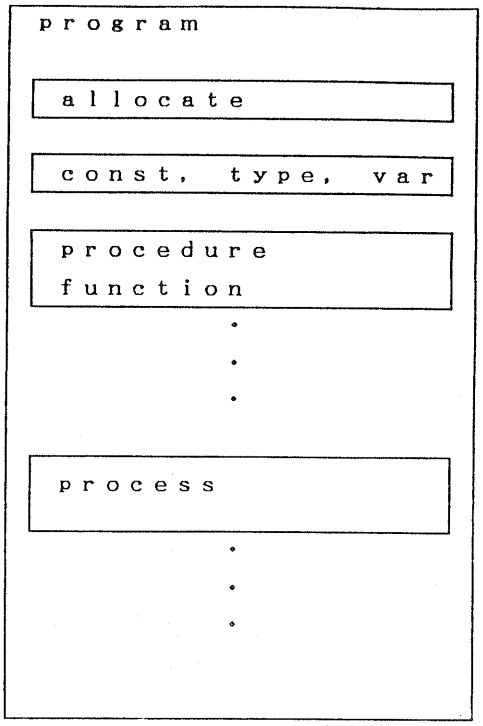

図４　ＩＬＯＳの構造

```
program SAMPLE1;

  process P1;
    ....

  process P2;
    ....

begin
  {section 1}
  execute( P1, P2 );
  {section 2}
end.
```

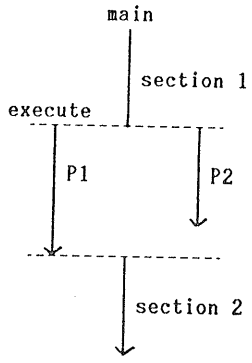

図５ａ プロセスの実行　図５ｂ

```
program SAMPLE( input, output );

  process P1;
    var x: shared real;
    ....

  process P2;
    var x: shared real;
    ....

begin
  while true do begin
    transmit
      P1.x -> P2.x;  P2.x -> P1.x
    end;
    execute( P1, P2 )
  end
end.
```

図６ａ　同期実行

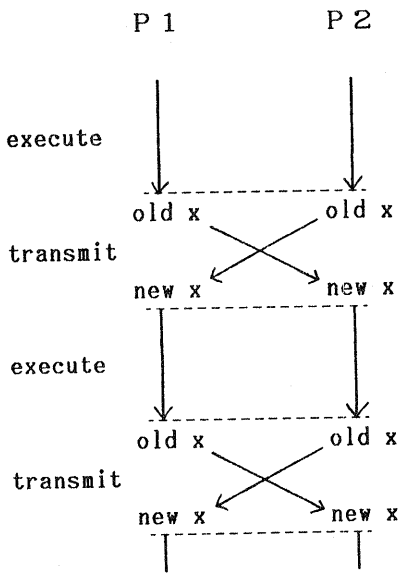

図６ｂ

4.2 メイルボックスによるプロセス間通信（非同期的通信）

4.1 で述べたようにプロセスは、execute により起動されるが、各プロセスが実行中に互いに通信し合う必要がある場合には、これから述べる非同期的な通信を用いることができる。ＩＬＯＳでは、この部分にメイルボックスによる通信方式を採用した。この方式を採用した理由としては、

（１）各プロセスは通信の為にメイルボックス名を指定すればよいので、多対多のプロセス間通信の記述が容易である。

（２）ＰＰＡのように、中継プロセッサを持つシステムでは、メイルボックスを中継プロセッサが管理できるため、実現が比較的容易である。

という２つが挙げられる。

ＩＬＯＳでは、プロセス間通信に用いられるメイルボックスは、図４の allocate 部で宣言される。例えば次の宣言

```
allocate BOX ( S1, S2 => R1, R2, R3 ): real;
```

は、BOX という名のメイルボックスには、送信プロセス S1, S2 からの書き込みが許され、R1, R2, R3 からの読み出しが許されるということを宣言している。このように宣言されたメイルボックスに対して各プロセスが読み書きをする場合には、各プロセス中において、図７ａ，図７ｂのようにして行なう。

```
process S1;
  var sdata: real;
  ....
begin
  ....
  send sdata to BOX;
  ....
end;
```

図７ａ　書き込み

```
process R1;
  var rdata: real;
  ....
begin
  ....
  receive rdata from BOX;
  ....
end;
```

図７ｂ　読み出し

この場合、送信プロセスは互いに排他的に書き込みが行なわれ、受信プロセスは並列に読み出しが行なわれる。

ＩＬＯＳでは、もう一つの読み出し方法として、select文を用意した。これは、一つの受信プロセスに複数のメイルボックスから受信データがやって来る場合に対応する為のものであり、次の規則に従った動作を行なう。

（１）select文中で指定したメイルボックスのうちどれかにデータが存在する場合には、そのメイルボックスに対応する文を実行する。
（２）複数のメイルボックスにデータが存在する場合には、そのうちの一つが非決定的に選択され、それに対応する文が実行される。
（３）どれにもデータが存在しない場合には、何もせずに select 文をぬける。

例えば、BOX1, BOX2 という２つのメイルボックスがある場合には、図８のように記述することができる。

```
select
  BOX1: begin
          {action 1}
        end;
  BOX2: begin
          {action 2}
        end
end;
```

図８　ＩＬＯＳの select文

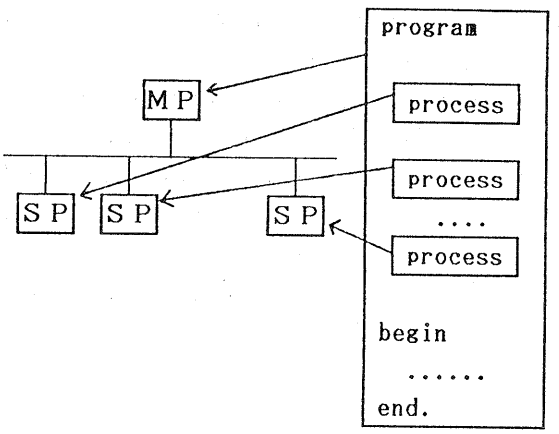

図９　プログラムの割り付け

5. 並列マシン上への実現

ＩＬＯＳは、２節で述べたように当大学汎用シミュレータ施設の並列演算部ＰＰＡ上に実現中であるが、それは次に述べるように３つの部分から構成されている。

（１）ＩＬＯＳコンパイラ

（２）並列部実行管理プログラム

（３）ホスト部実行管理プログラム

ＩＬＯＳコンパイラは、ＩＬＯＳソースプログラムをオブジェクトに変換するが、この時コンパイラは、各プロセス毎に別々のファイルを作成する。それと同時に、ホスト部用テーブルを作成するが、これはホスト部がＰＰＡにプログラムをダウンロードするために必要な情報や入出力の割り付け情報より成っている。また、コンパイラは並列部用テーブルも作成するが、これはＭＰがメイルボックスを管理するための情報と transmit によるプロセス間のデータの受け渡しに関する情報から成っている。

並列部実行管理プログラムは、ＭＰやＳＰのローカルメモリに置かれ、コンパイラの作成したテーブルによりプロセッサ間のデータの受け渡しを主に受け持つ部分であり、並列部のＯＳ的役割をする。

ホスト部実行管理プログラムは、ホストマシンであるVAX-11/780に置かれるが、その最も重要な仕事は、プログラムやテーブルを、コンパイラの作成したホスト用テーブルを参照して、指定したプロセッサにダウンロードすることである。その時のプログラムの割り付けを図９に示す。ホストはこの仕事が済むとＰＰＡのＭＰに起動をかけ、並列計算を開始させる。その後はＭＰからやってくる入出力の要求に対しての処理を行なう。

図１０にＩＬＯＳ処理系における処理の流れを示す。

6. おわりに

実際の並列マシン上に実現することを前提として、特に数値計算の処理効率を重視した並列処理用言語を開発した。これにより、専用サブシステムとして利用されることが多いプロセッサ・アレイ・システムのより広い応用が可能となる。しかし、処理の高速化という点で言えば、コンパイラによるオブジェクトコードの最適化も重要な問題であり、並列マシンに適したコード生成が今後の課題である。

< 参考文献 >

1) 牧野，三木；ＰＰＡ（パラレル・プロセッサ・アレイ）：汎用シミュレータ"ＨＯＳＳ"における並列演算サブシステム；電子通信学会研究会資料 EC82-72；1982.

2) Tutomu HOSHINO, et.al.; PACS: A Parallel Microprocessor Array for Scientific Calculations; ACM Trans. Comp. Syst., Vol.1, No.3, pp.195-221, 1983.

3) 渡辺，三谷，垣原，宮本；メモリ共有型マルチプロセッサシステムを対象とした並列処理用言語ＩＬＯＳの開発；昭和５９年度電機関係学会北海道支部連合大会；1984.

4) 三谷，渡辺，垣原，宮本；並列処理用言語ＩＬＯＳの処理系の実現；昭和５９年度電機関係学会北海道支部連合大会；1984.

5) 渡辺，三谷，垣原，宮本；並列処理用言語ＩＬＯＳ；電子通信学会昭和６０年度総合全国大会；1985.
6) 垣原，渡辺，三谷，宮本；ＩＬＯＳ処理系の実現；電子通信学会昭和６０年度総合全国大会；1985.
7) 三谷，渡辺，垣原，宮本；ＩＬＯＳの実行制御方式；電子通信学会昭和６０年度総合全国大会；1985.

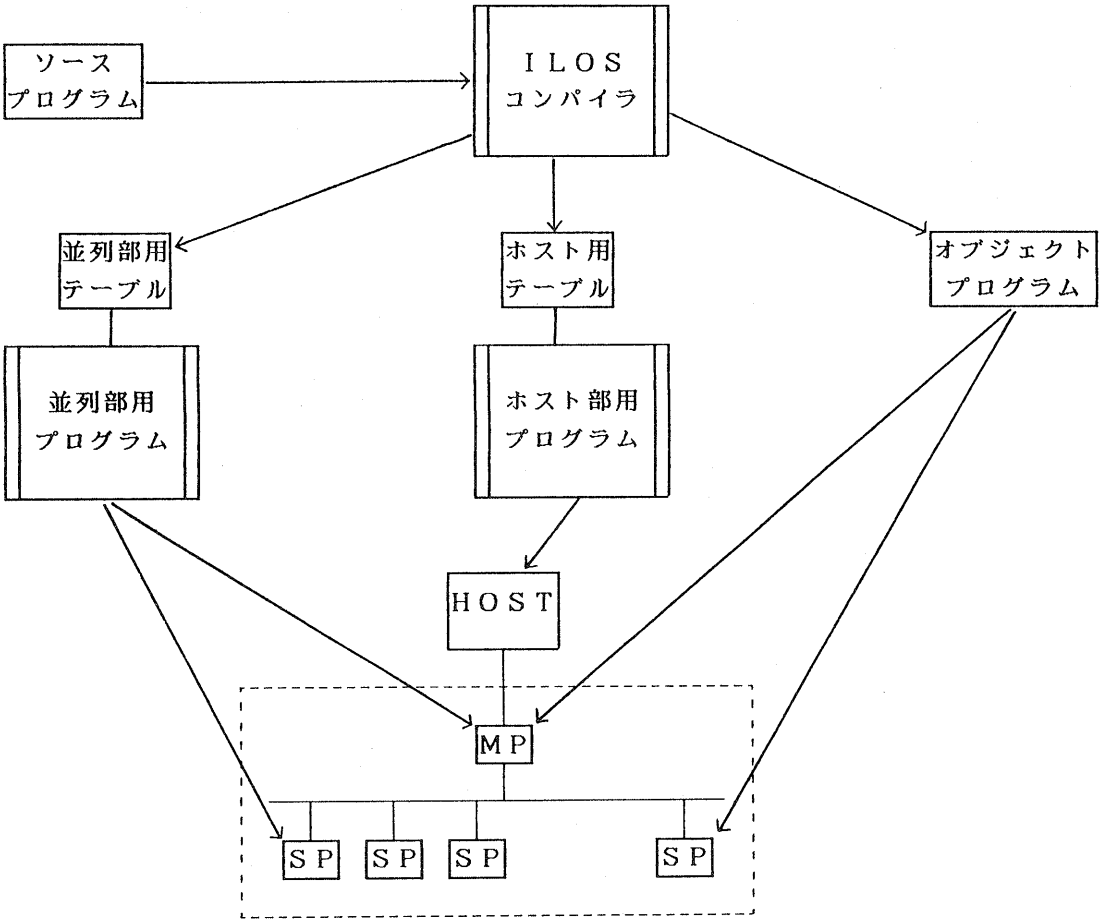

図１０　処理の流れ