1

並列推論エンジンPIE64

小池汎平・田中英彦

1. はじめに

本稿では，現在我々が開発を進めている並列推論エンジン PIE 64 について紹介する．PIE 64 は，知識処理をはじめとする各種記号処理を並列処理により高速実行することを目的としており，推論ユニットと呼ばれる処理要素64台が2系統の負荷分散機構付き多段結合網で結合されたアーキテクチャを持ち，Committed-Choice 型の論理型言語であるFLENGおよびその上位言語であるFLENG＋＋を実行する．

我々は並列処理技術の基本はプロセッサ間を結合するネットワークにあると考え，大規模並列計算機（プロセッサ台数100台以上）のための高性能なネットワークの現実的な構成法をまず第1に確立し，次にこのネットワークを用いた通信プロトコルを明確化し，これらをもとに，個々のプロセッサ上での効率的な処理方式を明確化していくという方針をとって，設計を進めてきた．そこで本稿では，PIE 64 について，特に並列処理機能（ネットワーク，ネットワーク・インタフェース・プロセッサ，分散メモリのガーベジコレクション）に焦点を当てて紹介することにする．

2. PIE 64 の言語

PIE 64 の目標は，知識処理をはじめとする各種記号処理を，並列処理により高速実行することである．PIE 64 では基本言語として，Committed-Choice 型の論理型言語である FLENG を用いている．そして，FLENG をベースとして，より高水準化/低水準化した二種類の言語を用意することにより，統一性を保ちつつ，高い問題記述能力と，高い処理効率を両立させたシステムを構築することを目指している．

PIE 64 の基本言語 FLENG は，Committed-Choice 型の並列記号処理言語である．Committed-Choice 型言語は，論理学に基づく明快な宣言的セマンティクスを持ち，同時に，単純で強力な並列記号処理記述能力を持つことを特徴としており，GHC, Concurrent Prolog などが知られている．FLENG では，セマンティクスの明確化，より高い柔軟性の獲得および実装の容易性のために，従来の Committed-Choice 型言語からさらに可能な限り機能が削られており，その結果として，定義節の非決定的選択，ゴールリダクションに基づくプロセスとデータの生成，および論理変数を用いたプロセス間同期のみが言語機能として残されている．GHC, Concurrent Prolog など，他の Committed-Choice 型言語との大きな違いは，

（1） ガード部を持たない

（2） ゴール同士に論理的な AND 関係がない

の2点である．つまり，FLENG はもはやゴール間の暗黙の AND 関係を用いてプログラムを記述するいわゆる論理型言語ではない．その代わりに，FLENG では，条件分岐のための論理条件もデータとして明示的に扱い，ゴール間で真理値を受け渡すことになる．このように，FLENG は論理型言語を元にしながらも，よりデータ駆動記号処理言語としての色彩を強めた言語といえる．

FLENG は

（1） 高い並列記号処理記述能力

（2） 明確なセマンティクス

（3） 柔軟性

という，基本言語として望ましい特徴を持つが，実用的な知識情報処理システムの言語として，そのまま用いるには，

（1） 複雑なプログラムを直接書くためには記述対象に向けた抽象度が不足している

（2） 効率的な実行処理を考えたとき，グローバルな最適化情報などに基づいた，きめの細かい低レベル最適化操作が記述できない

という点で問題であると考え，実用的なシステムに用いる言語として，上記二つの問題点にそれぞれ対応するために，より高度なプログラムの記述を目指した上位言語と，より低レベルでの効率的処理を目的とした下位言語に拡張することにした．上位言語を **FLENG＋＋**，下位語言を **FLENG－－**と呼ぶ．ただし，いずれの言語とも共通言語 FLENG をベースとして設計することで，言語システムとしての一貫性を維持することを考慮している．

上位言語 FLENG＋＋は，並列オブジェクト指向言語，並列解探索言語，知識ベース記述言語など，さまざまな知識処理パラダイムが基本言語のもとに自然に統合された並列言語を目指して現在実装が進められている．

下位言語 FLENG－－は，基本言語 FLENG に対して，宣言的解釈の可能性をなくさない範囲で，処理効率向上のためのさまざまなアノテーションを導入した言語である．効率的なメモリ管理を行なうためのアノテーション，負荷分散戦略を指定するアノテーションなどが導入されている．

3. PIE 64 の全体構成

PIE 64 は，64台の推論ユニット（IU）が，プロセス分配網（PAN），データ配置網（DAN）と呼ばれる2系統の自動負荷分散機構付き相互結合網で結合されている．2系統の網には異なった負荷情報が流される．負荷分散以外のネットワーク・オペレーションは2系統のネットワークをインタリーブして実行される．PIE 64 の全体構成を図1に示す．

4. 推論ユニットの内部構成

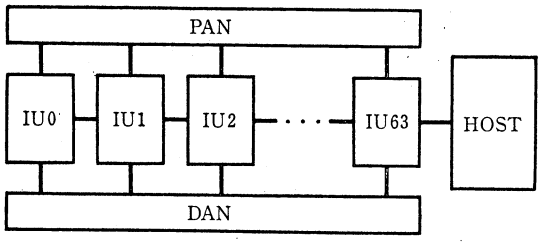
PIE 64 の個々の処理要素を**推論ユニット**と呼ぶ．それぞれの推論ユニットは，図2に示すように，

（1） 推論プロセッサ UNIRED

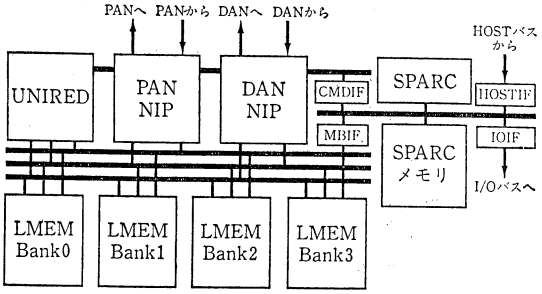
（2） ネットワーク・インタフェース・プロセッサNIP（2系統）

（3） 管理用プロセッサ SPARC

（4） ローカルメモリ（4バンク，4MB）



IU：Inference Unit
PAN：Process Allocation Network
DAN：Data Allocation Network
図1　PIE 64 の全体構成



UNIRED：Unifier/Reducer
NIP：Network Interface Processor
LMEM：Local Memory
図2　推論ユニットの内部構成

（5） SPARC 用ローカルメモリ

（6） SPARC 用コマンドバス・インタフェース

（7） SPARC 用メモリバス・インタフェース

（8） ホスト・インタフェース

（9） I/O インタフェース

から構成されている．

UNIRED，NIP は SPARC のコプロセッサとなっている．FLENG の実行処理が独立性の高い処理に分割され，これらのプロセッサにマッピングされる．各プロセッサはコマンド/リプライをやり取りしながら協調動作する．FLENG の実行処理のうち，ゴール書換え処理は UNIRED が，リモートデータアクセス，プロセス間同期処理などネットワークを介する処理は NIP が，ゴール管理，負荷分散，システムメンテナンスなどは SPARC が実行する．UNIRED，NIP，SPARC は32ビット幅の高速コマンドバスで結合されており，このバスを通して，コマンド/リプライのやり取りがなされる．また，UNIRED，NIP，SPARC はバスアービトレーションとメモリアクセスがパイプライン化された3本のメモリバスを通して，4バンクのローカルメモリを共有する．

ホストインタフェースを介して64台の推論ユニットの

全資源はホストワークステーションのメモリ空間にマップされる．また，I/O インタフェースを介して8台の推論ユニットごとに1本の I/O バスを共有し，PIE 64 全体で8本の独立した I/O バスを持つ．これらの I/O バスには，データベースマシン，グラフィックプロセッサ，信号処理プロセッサなどが接続されることを想定している．

5. PIE 64 のネットワーク

PIE 64 のネットワークは，**図3**に示すように，4×4 のクロスバースイッチを3段組み合わせた64入力64出力の多段結合網である．1ポート当りのデータ幅は32ビットで，各ポートは10M ワード/秒のデータ転送能力を持つ．また，このネットワークは自動負荷分散機能を備え，接続要求時点で接続可能な最小負荷のプロセッサへの経路を自動的に接続できるという特徴を持つ．以下で，ネットワークの基本構成要素であるスイッチングユニット・チップとその特徴，およびこれを用いたネットワーク・ハードウェアの構成方法について説明する．

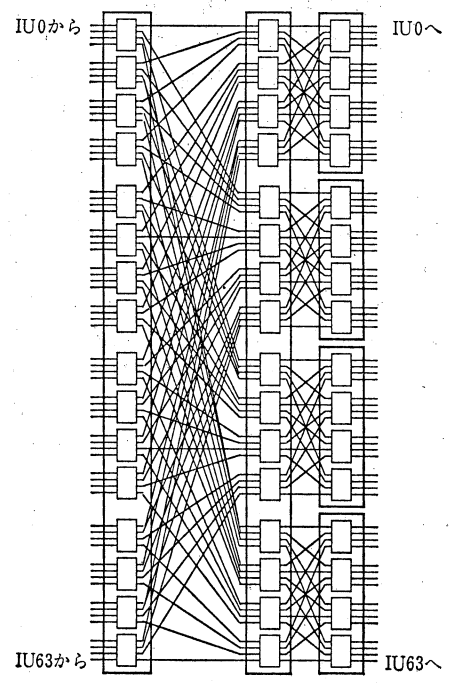


図3　PIE 64 のネットワーク

6. スイッチングユニット・チップ

スイッチングユニット（SU）・チップ†は，PIE 64 のネットワークの基本構成要素となる LSI である．SU チップは8700ゲート/179ピンのゲートアレイで実装され，8ビット幅4入力4出力のクロスバスイッチ，ルーティングコントローラなどから構成されている．SU チップの主な特徴は次のとおりである．

（1）相手を指定する通常の接続のほか，最小負荷の相手への経路を自動的に接続する自動負荷分散機能を持つ．これは，未使用のネットワーク経路を逆方向に流れる負荷情報を比較して接続先を決定することによって実現される．

（2）ビットスライス構成に対応するために，SU チップはマスタ/スレーブ二つのモードを持つ．マスタモードのチップがルーティングを行ない，接続情報を出力する．残りのスレーブモードのチップは接続情報を受け取ってクロスバスイッチの接続のみを行なう．PIE 64 では一つのマスタチップに三つのスレーブチップを接続し，データ幅32ビットとして用いている．

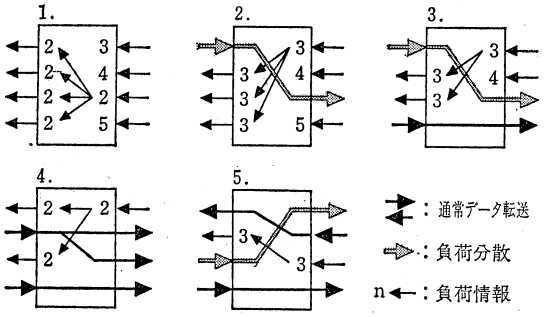


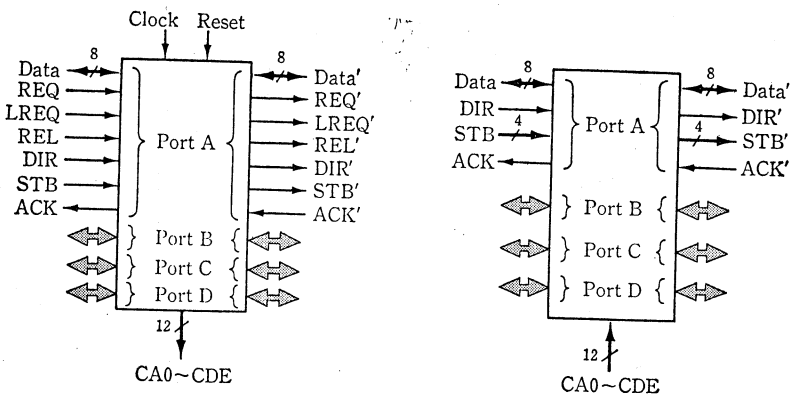
図4　SU チップを用いた通信の例

（3）回線交換，双方向のデータ転送をサポートする．

図4に SU チップを用いた通信の例を示す．1. はどのポートも未接続のときに負荷情報が逆方向に流れる様子を，2. は自動負荷分散接続要求に対して経路が設定された様子を，3. はさらに相手を指定した接続が起こった様子を，4. はマルチキャスト通信を行なっている様子を，5. は逆方向のデータ転送が行なわれている様子を表わしている．

図5に，マスタ/スレーブ両モードのチップを示す．

† 本 SU チップの基本概念は坂井修一氏（現 電総研）により考案され，山内 宗氏（現 NEC）によってブレッドボードが設計された．



（a）マスタモード　　（b）スレーブモード

図 5　マスタ/スレーブ SU チップ

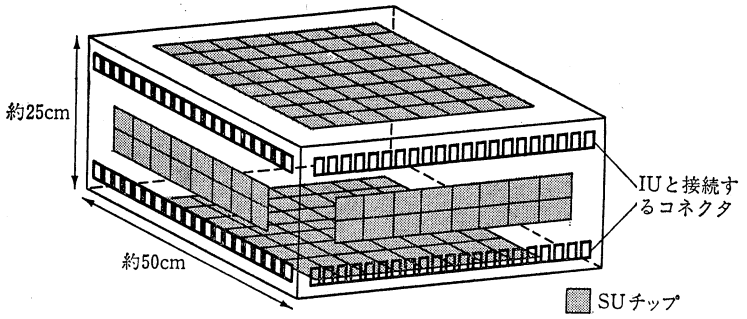


図 6　ネットワーク・ハードウェアの実装

7. ネットワーク・ハードウェアの構成

PIE 64 のネットワークは 64 入力64出力，データ幅32ビットであり，上で述べた SU チップ192個から成る．これらの SU チップ間を結線してネットワークを構成するに当たり，

（1）　配線遅延を最小に抑える

（2）　信号を安定に伝送する

などの理由から，配線長をなるべく短くし，ネットワーク・ハードウェアをコンパクトに作ることに目標をおいた．

ネットワーク・ハードウェアは，図6に示すように，1段目の16個の SU チップが載った基板4枚と，2段目と3段目の SU チップが64個ずつ載った基板2枚，合わせて二種類6枚の基板から構成される．これらの基板を箱状に組み，各基板間は箱の内側でフラットケーブルを用いて3次元的に配線する．

PIE 64 は，図7に示すように，中心にこのようなネットワーク・ハードウェア2系統が縦に詰まれ，4方向から16枚ずつの推論ユニット基板が挿入される構造となる．

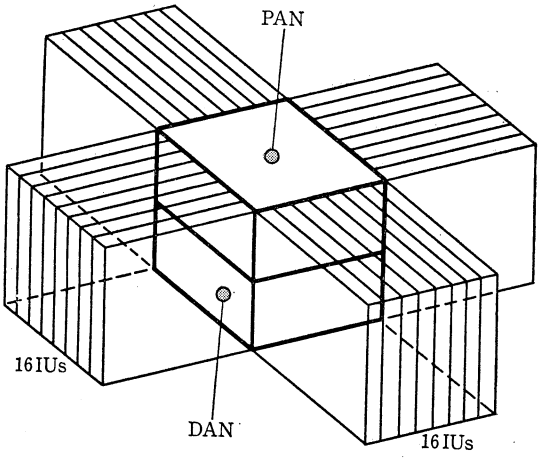


図 7　推論ユニットと2系統のネットワークの実装

8. ネットワーク・インタフェース・プロセッサ

PIE 64 のネットワーク・インタフェース・プロセッサ（NIP）は，各推論ユニット内部と相互結合網とのインタフェースを行ない，ネットワークを介して，データ

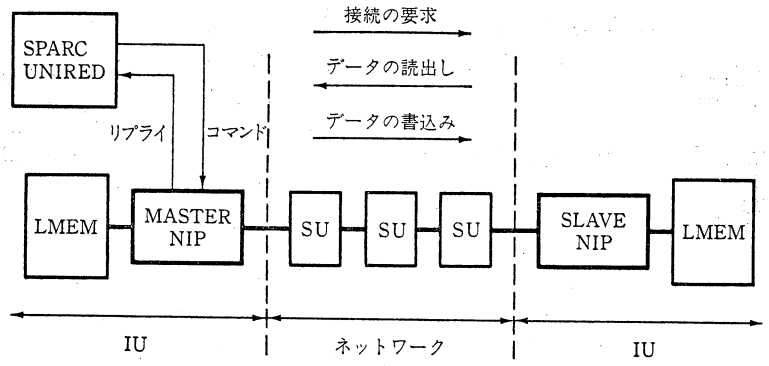


図 8　NIP の働き

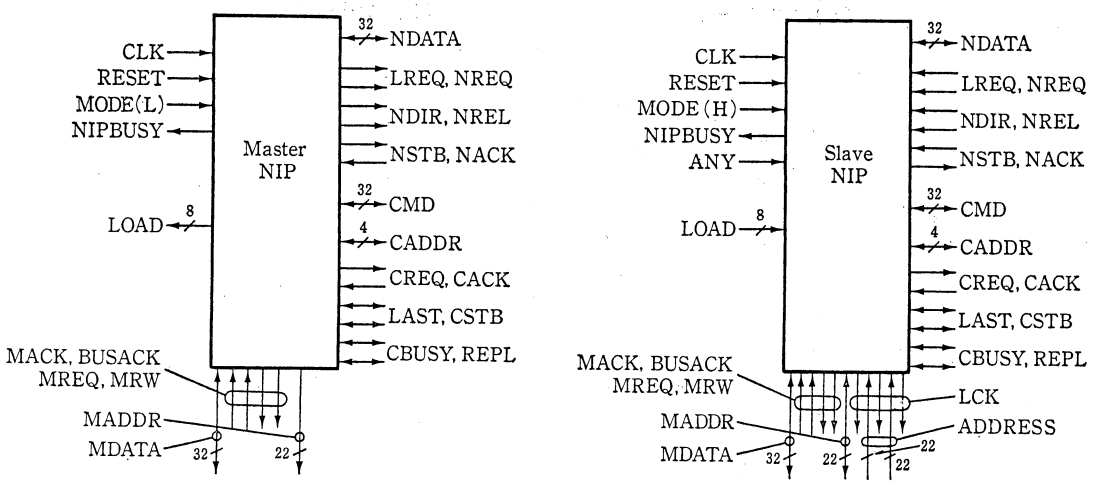


図 9　マスタ/スレーブ NIP チップ

転送，プロセス間同期などのためのコマンドを2台の推論ユニットの間で実行することにより，PIE 64 の並列処理機能を支援する．NIP により，推論ユニット内からネットワークを介して見た外部の世界は高度に抽象化される．このことは，個々の推論ユニット内で行なうべき処理とその効率化の方針を明確化するのに大いに役立つ．

NIP の主な特徴を以下にあげる．

（1）　並列処理で必要なネットワークを介したさまざまな操作を，ネットワークの形態（トポロジ，交換方式など）と独立に，コマンドバス上のコマンドとリプライという形に抽象化してローカルプロセッサに提供する

（2）　FLENG をはじめとする並列記号処理言語で用いられるデータ型を考慮したデータ転送コマンド体系を持つ

（3）　ネットワークの自動負荷分散機構を利用するデータ転送コマンドを持つ

（4）　FLENG の論理変数によるプロセス間同期処理を直接支援するコマンドを持つ

（5）　分散メモリのガーベジコレクションを直接支援するコマンドを持つ

このように，単純なメッセージ通信機能などと比べ，より目的指向で機能レベルの高い並列処理支援を行なうことによって，並列処理コプロセッサとしての独立性を高め，メインプロセッサの並列処理のオーバヘッドを抑えることを目指している．

NIP は，図8に示されるように，相互結合網と推論ユニットの内部バスとに接続され，推論ユニットのローカルメモリを直接アクセスし，ネットワークとデータをやりとりする．NIPはコマンドバスを通じて UNIRED/SPARC/他の NIP からコマンドを受け取り，必要に応じてリプライを返す．NIP は，マスタ部とスレーブ部とに分かれている．マスタ NIP は，UNIRED および SPARC からコマンドを受け取り，相手推論ユニットまでのネットワークの経路を接続したのちに，接続先推論ユニット内のスレーブ NIP と協調してコマンドを実行する．PIE 64 は2系統のネットワークを持つので，各

推論ユニットにはそれぞれのネットワークのために2系統の NIP が用意されている．NIP は 20000 ゲート/256 ピンの CMOS ゲートアレイで実装する予定である．マスタ NIP，スレーブ NIP の信号を図9に示す．

9. ネットワーク・インタフェース・プロセッサのコマンド

NIP のコマンドは基本的に，

（1） データ転送用

（2） プロセス間同期用

（3） 分散ガーベジコレクション用

の三種類に分けられる．いずれも，FLENG の実行処理に適合することが考慮されている．コマンドの実行は，コマンドバスを介して，1ワード以上のコマンドに対し0ワード以上のリプライが返される形でなされる．以下でこれらのコマンドを，

[reply=] command(parameter, …)

の形で示しながら説明していく．

データ転送用コマンドとしては，

```
val=read1(src)
dst=read2(src, dst)
dst=readn(src, dst)
dst=readx(src, dst, len)
dst=write1(dst, val)
dst=write2(dst, src)
dst=writen(dst, src)
dst=writex(dst, src, len)
dst=writem1(dstPE, val)
dst=writem2(dstPE, src)
dst=writemn(dstPE, src)
dst=writemx(dstPE, src, len)
dst=writel1(val)
dst=writel2(src)
dst=writeln(src)
dst=writelx(src, len)
```

がある．ここで，src は転送元(グローバル)アドレス，dst は転送先（グローバル）アドレス，val は読み出された値または書き込まれた値を表わす．read，write，writem，writel は，それぞれ，

（1） 相手推論ユニット内データの読出し

（2） 相手推論ユニットメモリへのデータの書込み

（3） 指定推論ユニットへのメッセージデータの書込み

（4） 負荷最小推論ユニットへの負荷データの書込み

を表わす．また，末尾の 1，2，n，x は，転送するデータのサイズを表わし，1ワード，2ワード，データの先頭からサイズを読み出す，コマンドのパラメータで転送長を直接指定する，をそれぞれ表わす．前三者は，それぞれ FLENG の，変数型，リスト型，ベクタ型に直接対応する．

データ転送は，基本的には，コピー元，コピー先のメモリアドレスを指定した，2台の推論ユニットのローカルメモリ間でのメモリ内容のブロック転送である．ただし，1ワードのメモリアクセスの場合，データはコマンドバスを通して直接やり取りされる．リプライが返ってきたときにはメモリへの書込みが終了していることが保証され，これによって処理の同期をとることができる．メッセージデータ，負荷データの書込みはスレーブ NIP がページ単位で管理するヒープ領域に対して行なわれる．また，FLENG の実装において，構造データ（リスト，ベクタ）の内部に論理変数のための領域を埋め込み，メモリの節約と変数のデレファレンス回数の減少を図るために，リスト，ベクタを転送する際に，埋め込まれた未定義変数値を元の場所を指す変数型ポインタに書き換え，変数領域の一意性を保証する機能を持つ．

プロセス間同期用のコマンドとして，

```
suspend(remotevariable, suspendID)
oldval=bind(remotevariable, val)
activate(remotesuspendID, val)
```

が用意され，FLENG の持つプロセス間同期機構をハードウェア・レベルで直接サポートする．

suspend は，ゴール書換え処理が，別の推論ユニットのメモリ上にある論理変数が未束縛であるためにサスペンドしたとき，その論理変数が保持するサスペンションレコードにゴールの識別子を登録するためのコマンドである．サスペンションレコードにはリスト構造が用いられ，スレーブ NIP がフリーリストを管理する．suspend コマンドの動作の例を図10に示す．

bind は，ある推論ユニット内で active unification を行なうときに，別の推論ユニットのメモリ上の論理変数に値を束縛するために用いるコマンドである．bind コマンドを受け取ったスレーブ NIP は，変数に値を束縛する前に変数の値を読み出し，未束縛かどうかをチェックするとともに，束縛の成否を知らせるために，読み出した値をマスタ NIP に送り返す．変数が未束縛だった場合，スレーブ NIP は変数に値を書き込むとともに，その変数が保持するサスペンションレコードに登録されているすべてのゴールに対して activate コマンドを発行

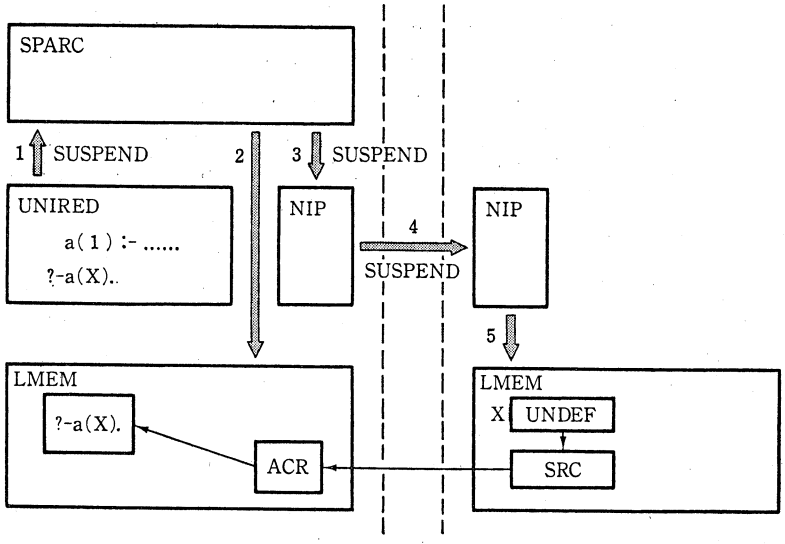


図 10 suspend コマンドの動作

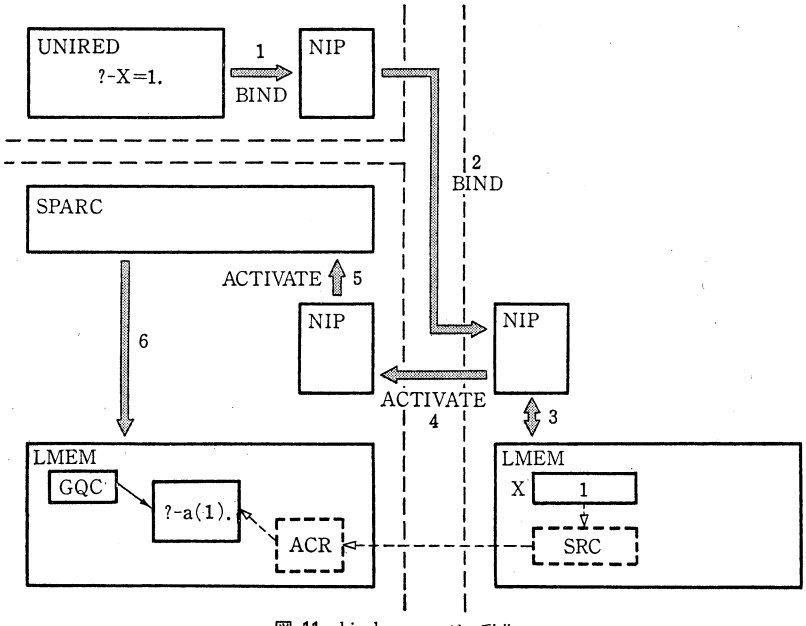


図 11 bind コマンドの動作

する．変数を読み書きするメモリ操作は不可分に行なわれる．図11に bind コマンドの動作を示す．

activate は，変数のサスペンションリストに登録されている個々のゴールが置かれた推論ユニットに対して，ゴールの識別子と変数に束縛された値を送りつけ，ゴールの管理を行なう相手の SPARC に対してゴールのアクティベートを要求するコマンドである．

分散ガーベジコレクション用のコマンドには，

mark1(remotepointer, location)
mark2(remotepointer, location)
markn(remotepointer, location)
restore(remotepointer, location)

がある．ここで remotepointer はガーベジコレクションの際に出現したリモートポインタの内容を，location はリモートポインタの置かれているローカルメモリのアドレスを表わす．これらのコマンドの動作は，後に分散メモリのガーベジコレクションについて述べる際に説明する．

NIP による1回のコマンドの実行は，

1. コマンドの受取り
2. マスタ NIP による前処理
3. ネットワークの接続

4. ネットワークを介したデータのやり取り（双方向）
5. ネットワークの解除
6. マスタ NIP およびスレーブ NIP による後処理
7. リプライの返送

の順で行なわれる．ネットワークの接続は衝突がなければ3クロック，データの転送は1クロック/ワード（10 MW/sec），ネットワークの解除は1クロックで行なわれる．この間，マスタ NIP とスレーブ NIP は1クロック遅延のあるハンドシェークを行なって同期をとる．また，ネットワークが回線交換方式なので1クロックの手間でデータ転送方向を反転させ，双方向のデータのやりとりを行なうことができる．このため，実行頻度の高いリモートデータの読出しも1回のネットワーク接続で行なうことができる．

10. PIE 64 のメモリ管理

ガーベジコレクションをはじめとするヒープメモリ管理は，記号処理において必要不可欠な基本技術である．高い頻度で行なわれるデータ領域のアロケートと再利用のための回収を最小限のオーバヘッドで実現する必要がある．

特に，PIE 64 では，

（1） ヒープが64台の推論ユニットのローカルメモリ上に分散されて形成される
（2） 1台の推論ユニット内にもコプロセッサごとに複数のヒープ成長点が存在する
（3） 可変長データを取り扱う

などの点を考慮しつつ統一的な方式を確立する必要がある．そこで PIE 64 では，

（1） スライディングコンパクションを伴う分散メモリの一括ガーベジコレクションを各種ハードウェアサポートのもとに高速に行なう
（2） ヒープメモリを多数のページに分割し，複数のヒープ成長点それぞれに領域をページ単位で動的に割り当てる．また，一部のデータは特別のページに割り当て，ページごとに設けた参照カウンタを用いて領域をページ単位で即時回収する
（3） プログラムの静的解析によって可能な限りメモリ消費量を抑える

という方法を併用している．ここでは，これらのうち分散メモリの一括ガーベジコレクションについて詳しく説明する．

11. 分散メモリのガーベジコレクション

記号処理において，ベクタなどの可変長データをサポートすることは，実用的なシステムを作る上で当然必要なことであるが，メモリ管理の立場から見ると，ヒープ内に生じるさまざまなサイズの不用メモリ領域を回収するためにデータのコンパクションを行ない，データの移動に伴うアドレスの変更をすべてのポインタに反映させるという難しい問題を解決しなければならない．特に，PIE 64 のような分散メモリ構成の並列計算機の場合，分散メモリ間にわたってデータを参照するすべてのリモートポインタに対して，ポインタ書換えを行なうのは困難とされ，これを避けるために，間接参照テーブルを用意し，データの移動をプロセッサの外部に見せないようにする場合が多かった．しかしこの方法は，

（1） 通常処理時のデータアクセスの手間が増大する
（2） 構造データをプロセッサ間で移動させるたびにその中に含まれているローカルポインタとリモートポインタとの変換を行なう手間が必要である
（3） 間接参照テーブルエントリの回収はいずれにしても必要である

などさまざまな問題を持っており，処理の複雑化，データ転送，オーバヘッドの増大などを招いて，柔軟な並列処理を阻害する．

PIE 64 では，このような間接参照テーブルを通常処理時は用いず，一括ガーベジコレクションを行なう際だけネットワーク・インタフェース・プロセッサのサポートで一時的に作成することにより，コンパクションによるアドレスの変更をすべてのリモートポインタに直接反映させるようにしている．この結果，上記のような問題はなくなり，処理の単純化，データ転送オーバヘッドの大幅な減少などがもたらされている．

一括ガーベジコレクションは，いずれかの推論ユニットがヒープメモリを使いきると起動され，64台の推論ユニットがいっせいに処理を中断して，それぞれのローカルメモリ内の生きているデータのマーキングとコンパクションの各フェーズを順に同期をとりながら行なう．コンパクションには Morris のアルゴリズムを使用している．リモートポインタは，NIP の mark コマンド，restore コマンドを用いて処理する．この様子を図12に示す．

マーキングの際にリモートポインタが出現すると（a），NIP に mark コマンドを発行し，リモートデー

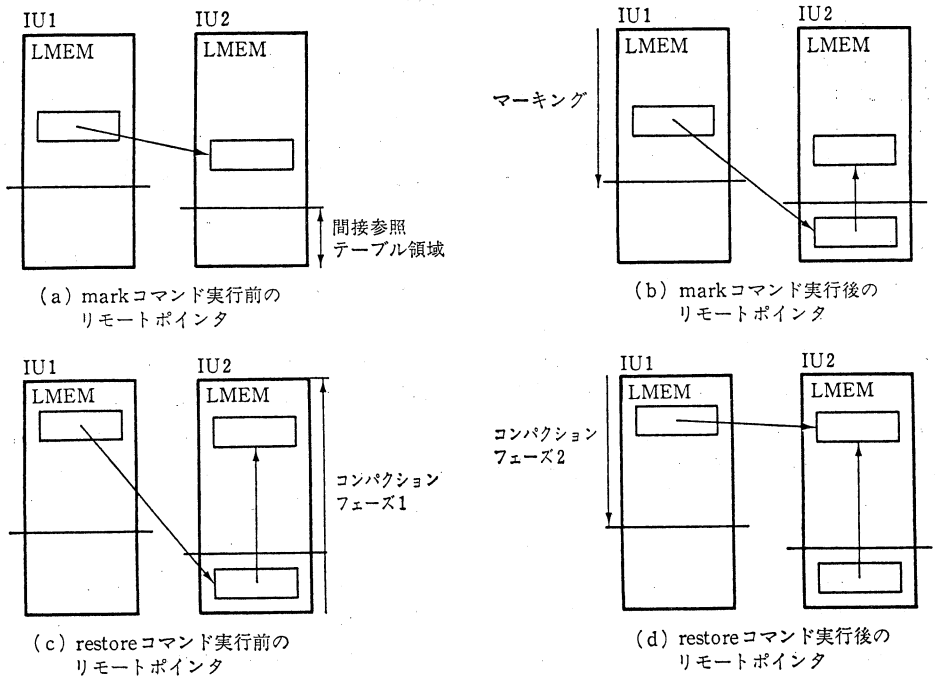


図 12　ガーベジコレクションにおけるリモートポインタの書換え

タのある推論ユニット内に間接参照テーブルのエントリを作り，リモートポインタをここを指すように書き換える(b)とともに，ルートの管理を行なっている相手のSPARC に対しマーキングの継続を依頼する．

コンパクションの第1フェーズで間接参照テーブルの各エントリは新しい移動先に書き換えられる(c)．

コンパクションの第2フェーズで間接参照テーブルのエントリを指すリモートポインタが出現すると NIP に restore コマンドを発行して，コンパクションで移動したデータの新しいリモートアドレスをリモートポインタに書き戻す(d)．

以上のリモートポインタの処理は，2系統のネットワークの NIP がコマンドを受け取ってローカルポインタのマーキング/コンパクションと並列に実行する．シミュレーションによると，64台の推論ユニットが並列に64台分のメモリ（256MB）のガーベジコレクションを行なう時間は，1台で1台分のメモリ（4MB）をガーベジコレクションする場合と比べ1.2倍程度の時間に抑えられており，NIP の機能レベルを高く設定することにより，並列処理オーバヘッドを抑えるという基本方針の有効性が示されている．

一括ガーベジコレクションはマーキングフェーズ，コンパクション第1フェーズ，コンパクション第2フェーズの各フェーズを正確に同期して移行する必要がある．64台の推論ユニットの間の同期を高速にとるために，ワイアドオア信号によって状態遷移条件を高速にやり取りするためのハードウェアが用意されている．また，もし推論ユニット間にローカルメモリの使用量や消費速度の偏りがあると，その推論ユニットが原因となって，一括ガーベジコレクションを頻発させてしまう恐れがあるので，これを防ぐために，一度ガーベジコレクションを要求した推論ユニットは，そのあと何回かメモリを使い切ってもガーベジコレクションを要求することを控えるなど，アルゴリズムに工夫を加えている．また，そもそもデータに偏りが起きないようなソフトウェア上の工夫も重要と考える．

12. 推論コプロセッサ UNIRED

推論コプロセッサ UNIRED は，FLENG の実行処理のうち，ゴール書換え処理を行なう．ゴール書換えの基本処理である，

- Passive Unification
- Active Unification
- Goal Reduction
- Overwrite Goal Reduction

を行なう命令列がパイプラインを用いて高速実行される．Passive Unification はゴールと定義節ヘッド部とのユニフィケーション，Active Unification は定義節がコミットされてから実行されるゴール側変数への値の束

縛を許すユニフィケーション，Goal Reduction は定義節ボディ部を用いたゴールとデータのメモリ上への生成である．特にプログラムの静的な解析によりゴミになることが判断できるゴール/データについては，Overwrite Goal Reduction によってメモリ領域の再利用が行なわれ，メモリ消費速度の低減と不要なデータ転送の除去による最適化が図られる．UNIRED の内部では，複数のコンテクストが並行処理され，ゴール書換え処理で，リモートデータのアクセスが生じるたびに，NIPからのリプライを待ち合わせている間，別のコンテクストに切り替えて処理を進め，処理のスループットを維持する．

UNIRED は，現在機能設計を進めている段階であり，詳細は別の機会に譲りたいと思う．

13. おわりに

現在，我々が開発を進めている並列推論マシンPIE 64の概要について解説した．現在，各部の詳細設計とLSIの開発が進められている．

（1） シミュレーションによる性能予測
（2） プログラミング環境の整備
（3） 応用プログラムの開発

などが今後の課題である．

謝　辞

本研究を手伝ってくださっている多くの方々，特に田中研究室の，Martin Nilsson 氏（FLENG），高橋栄一氏（SU チップ），中村宏明氏（FLENG＋＋），許魯氏（ガーベジコレクタ），吉田　実氏(データベースマシン・インタフェース)，島田健太郎氏（FLENG 処理系），清水　剛氏（NIPチップ）に深く感謝したいと思う．なお，本研究は文部省特別推進研究の一環として行なわれている．

参考文献

1) 小池汎平，田中英彦：並列推論マシン PIE 64 の概要，他関連発表，情報処理学会第37回全国大会予稿集，5N-4～9，1988年9月．
2) Nilsson, M. and Tanaka, H.: The Art of Building a Parallel Logic Programming System, ロジックプログラミングコンファレンス87, ICOT, 1987年6月．
3) 坂井修一，小池汎平，田中英彦，元岡　達：動的負荷分散を行なう相互結合網の構成，情報処理学会論文誌，Vol. 27, No. 5, 1986.
4) Koike, H., Takahashi, E., Yamauchi, T. and Tanaka, H.: The High Performance Interconnection Network of Parallel Inference Machine PIE64, コンピュータアーキテクチャシンポジウム予稿集，情報処理学会，1988年5月．
5) Koike, H. and Tanaka, H.: Multi-Context Processing and Data-Balancing Mechanism of the Parallel Inference Machine PIE 64, *Proc. of the International Conference on Fifth Generation Computer Systems*, ICOT, Dec. 1988.
6) 清水　剛，小池汎平，島田健太郎，田中英彦：並列推論マシン PIE64 のネットワークインタフェースプロセッサ，並列処理シンポジウム予稿集，情報処理学会，1989年2月．