ICOT Technical Report: TR-119

TR-119

並列推論マシン PIM-R の ハードウェア・シュミレータ試作

杉江　衛, 米山　貢, 坂部俊文, 岩崎正明
吉住誠一（日立製作所）
麻生盛敏, 清水　肇, 尾内理紀夫(ICOT)

June, 1985

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

並列推論マシンPIM-Rのハードウェア・シミュレータ試作

杉江　衛，米山　貢，坂部俊文，岩崎正明，吉住誠一（日立）
麻生盛敏，清水　肇，尾内理紀夫（ＩＣＯＴ）

ABSTRACT－ A hardware simulator of PIM-R (Reduction－Based Parallel Inference Machine) has been developed. 8 MC68000 single board computers and shared storage operate as the inference modules and the network, respectively. Detailed structure of inference mechanism are implemented by software on MC68000. The control module and the inference modules are connected by means of GPIB. In order to realize high simulation rate, an event-driven method is introduced. "Queens" program and "Quick-sort" program were executed on the simulator. The results show that a PIM-R architecture can effectively utilize the parallelism in Prolog/ Concurrent Prolog programs.

１．　はじめに

第五世代コンピュータ・プロジェクトでは述語論理型言語をベースにした知識情報処理システムの研究開発が行なわれている。述語論理の基本操作は推論であるから、そのハードウェアは推論マシンと呼ばれる。推論は、並列処理が可能なので、このマシンは並列動作をその基盤に置いている。並列推論処理方式および述語論理型言語に関して、これまでいくつかが提案されてきた［1］、［2］、［3］、［4］。我々は、第五世代コンピュータ・プロジェクトが核言語第１版(KL1-84)のベース言語として位置づけているPrologとConcurrent Prolog[4]を対象言語に選択して、リダクションの概念に基づいた並列推論マシンＰＩＭ－Ｒ(Reduction-Based Parallel Inference Machine) のアーキテクチャを提案し［5］、［6］、［7］、研究開発を進めてきた。ＰＩＭ－Ｒのア-キテクチャ面での特徴は、①並行プロセス間チャネル用の分散化メモリの導入、②効率的なパケット分配のためのNetwork Nodeの導入、③ground instanceである長い構造体データ格納のための構造体メモリの導入等にある[8]。また、処理方式面では、①PrologのOR並列処理、② Concurrent PrologのAND並列処理、③ ストラクチャコピ-方式、④ reverse compaction方式、⑤ only-reducible-goal-copy方式、⑥ 独特のプロセス構成法[9]を特徴としている。上記④~⑥は、ストラクチャコピ-方式の採用に伴う処理量とネットワーク・トラヒックの増加を抑制する目的で導入している。

我々は、アーキテクチャの提案と共に、ＰＩＭ－Ｒアーキテクチャ検証の第一歩として、ＰＩＭ－Ｒの原理的動作確認のためにPrologで記述されたソフトウェア・シミュレータ(Prolog版シミュレータ)を開発し、各種データを収集した。その結果、ＰＩＭ－ＲがPrologおよびConcurrent Prologに内在する並列性を引き出すこと、Message Boardのための専用Controller導入効果、Network Nodeによる子プロセスの動的分配効果等を確認した[10]。しかしこの Prolog版シミュレータは原理的動作の確認を目的としたため、データ構造等ＰＩＭ－Ｒを詳細にシミュレートすることが難しく、また、シミュレーションに時間を要した。そこで、①ＰＩＭ－Ｒ詳細構造のシミュレーション、②シミュレーション

速度の向上、③使用メモリ空間の拡大、④並列環境シミュレーション を目標にハードウェア・シミュレータを並行開発することにした。特に並列環境シミュレーションを目標の1つにしたのは、並列環境ゆえに生起する各種問題（デッドロック等）に関するデータ収集、あるいは、並列環境のためのソフトウェア・ユーティリティ構築のための各種データが、逐次マシン上のソフトウェア・シミュレーションでは十分には収集できないからである。

開発にあたっては、早期開発と柔軟性を開発条件とした。なぜならば、ＰＩＭ－Ｒのデータ収集／評価／改良というサイクルに耐えるためには、ハードウェア・シミュレータは柔軟でなければならない。また、Prologで記述されたソフトウェア・シミュレータによるＰＩＭ－Ｒの原理的動作確認に引き続き、ハードウェア・シミュレータによるデータ収集を開始することが望ましく、早期開発を目指す必要があった。このように、早期開発と柔軟性実現のため、我々はMC68000搭載のシングル・ボード・マイクロコンピュータ（以下SBCと略す）を用いて、ＰＩＭ－ＲのInference Moduleを実現することにした。そしてＰＩＭ－Ｒの各種モジュールは、ソフトウェアで実現し、各モジュールの変更改良に容易に追従できるようにした。ハードウェア・シミュレータ上での各種応用プログラムの走行を可能にするため、各SBCには、十分なローカルメモリを付けることとし、10MBまで実装可能とした。また、SBC群と共有メモリのスーパバイザとしてVAX11/780を導入した。

このようにして開発した結果、ネットワークについては、2種類のメッシュ構造と線状構造についてシミュレーションを行なうことができ、また、Prolog版シミュレータによる原理的動作確認に引き続き、ハードウェア・シミュレータによるデータ収集を開始することができた。

本稿では、並列推論マシンＰＩＭ－Ｒハードウェア・シミュレータの試作詳細と、PrologとConcurrent Prologプログラムによるシミュレーション結果について述べる。

2．ハードウェア・シミュレータ

2.1 構成と動作概要

図1に試作したハードウェア・シミュレータの概念的構成を示す。ＰＩＭ－Ｒのアーキテクチャに従って、推論モジュール（IM: Inference Module）、ネットワークおよび構造体メモリ・モジュール（NSM:Network and Structure Memory Module）、制御モジュール（CM:Control Module）の3つのモジュールから構成されている。

これらのモジュールの主な機能は次のとおりである。CMは言語処理系から成り、プログラムのコンパイル、IMの起動と結果の出力を行なう。NSMはCM-IM間の交信データのバッファ（CMバッファ）、IM-IM間の交信データのバッファ（IMバッファ）を行ない、長い構造体データの格納も受け持つ。IMは5つの処理系と3つの格納領域からなる。以下にその機能を記す。

（ａ）Nucleus

①CMからの指令割り込みに対する処理。（CMからの指令については、2.2節に述べる。）

②PPU(Process Pool Unit)処理系の起動。

（ｂ）PPU処理系

①プロセスの生成、削減およびReady, Wait,Suspendの状態遷移の管理。

②Ready Processからのゴールの切り出しと該ゴールのUU（Unification Unit)処理系への送信。

③NET処理系へのパケットの送信。

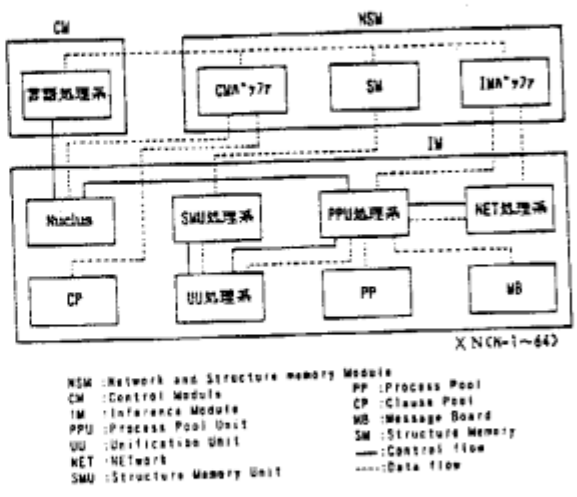


図1　ハードウェア・シミュレータの概念構成

（c）UU処理系

①ゴールと該当クローズのユニフィケーションによるリダクション。

②長い構造体データのユニフィケーションに関するSMU(Structure Memory Unit)処理系の呼び出し。

（d）NET処理系

①送信先のIMに対応したIMバッファへのパケットの書き込み。

②IMバッファ内のパケットの待ち行列管理。

（e）SMU処理系

①長い構造体データのユニフィケーション。

（f）Process Pool

①プロセスに関係したデータを格納する。これらのデータには、プロセスの状態、親プロセス、変数に関するデータが含まれる。

（g）Clause Pool

①ソースプログラムを格納する。

（h）Message Board

①プロセス間の同期用のチャネルに関するデータを格納する。

図2にシミュレータのハードウェア構成を示す。IMはSBCとそのバスに接続されたローカル・メモリから構成される。NSMは8台のSBCがアクセス可能な共有メモリからなり、CMはVAX11/780からなる。NSMに対応する共有メモリは、共有バスに接続される。この共有バスには、共有メモリの他に、GPIBインタフェイスとSBC8台分のIMボードインタフェイスが接続され、CM-IM間、IM-NSM間の交信を実現する。VAX11/780はGPIBインタフェイスを介して共有メモリにアクセスし、かつ、目的のSBCに指令を送る。各SBCはIMボードインタフェイスを介して共有メモリにアクセスする。バスアービタは共有バスの所有権を制御することによって、8台のSBCとVAX11/780の共有メモリへのアクセスを可能にする。IMボードインタフェイスは各SBCにローカル・メモリを所有させるために導入した。ローカル・メモリと共有メモリはアドレス空間を分割して実現し、共有メモリのアドレス空間に対して、SBCバスを共有バスに接続する。各SBCのアドレス・マップを図3に示す。ローカル・メモリは最大10MB、共有メモリは最大4MBの容量を持ち得る。試作したハードウェア・シミュレータでは、共有メモリに2MB、ローカル・メモリに30MB(2MB～6MB/SBC)を実装した。

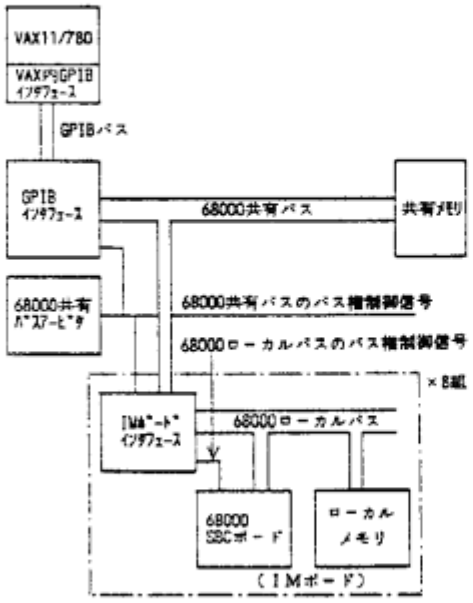


図2　シミュレータのハードウェア構成

GPIBはインタフェイス・ハードウェアが簡単で約50KB/secの転送速度が得られる。この点とVAX11/780用インタフェイスが市販されている点を考慮して採用を決定した。また、各種のネットワークのシミュレーションを可能にするために、NSMには共有メモリを用いることにし、特定のハードウェアはインプリメントしないことにした。

ハードウェア・シミュレータの動作の概要は次の通りである。CMは、まず、全IMをリセットする。ユーザ・ファイルより、プログラム、シミュレーション用パラメータおよび質問を読み込んで、前2者については、全IMのローカル・メモリにブロードキャストし、質問は特定の IM(#0)のバッファに書き込む。

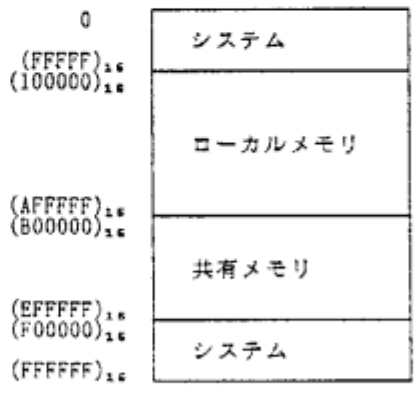


図3　SBC のアドレス・マップ

そして、これらの処理完了後、全IMに起動をかける。

IMでは、Nucleusがパワー・オン・リセット又はCMからのリセットによって起動され、Clause Poolの先頭アドレス、Process Poolの先頭アドレスおよびカレント・ポインタ、Message Boardの先頭アドレスおよびカレント・ポインタ等を初期設定した後、CMからの起動を待つ。CMから起動されると、Nucleusは制御をPPU処理系に移す。PPU処理系は言語指定パラメータを判定して、Prolog処理モジュール、Concurrent Prolog処理モジュールのうちの一方を呼び出す。Prolog処理モジュール、Concurrent Prolog処理モジュールは、他PPUからのパケット処理ルーチンを呼び出して、リダクションを行なう。他PPUからのパケット処理ルーチン、レディ・プロセス処理ルーチンは、リダクションすべきゴールが存在した場合にUU処理系を呼び出してユニフィケーションを実施する。また、UUからのパケット処理ルーチンでは、分配ストラテジに従ってゴールを分配し、NET処理系を呼び出して他のIMにパケットを送りつける。

CM-NSM間の交信は、VAX11/780によるメモリの書き込みと読み出しであり、GPIBインタフェイスを介してIMに指令を送ってDMAコントローラの初期設定を行なった後、GPIBインタフェイスを介してデータを送受する。CM-IM間の交信は、CMおよびIMから他方への指令と、データの転送に分けられる。CM-IM間のデータ転送は、NSM内のCMバッファを経由して行なわれる。即ち、CMからIMへの転送の場合は、VAV11/780がCMバッファへデータを書き込み、この書き込み終了後、SBCへ指令を送ってCMバッファ内のデータをSBCに読み込ませる。IMからCMへの転送の場合は、VAX11/780がSBCへ指令を送ってCMバッファにデータを書き込ませ、SBCからの書き込み終了指令を受け取った後に、CMバッファ内のデータを読み出す。各指令はGPIBインタフェイスを介して送る。IM-IM間の交信は、NSMを経由してデータ・パケットを転送することによって行なわれる。NSMには、各IMと1対1に対応したIMバッファがあり、転送先のIMに対応するIMバッファにデータ・パケットを書き込む。データ・パケットの読み出しは、転送先のIMが行なう。各IMはデータ・パケットの到着をサーチし、到着していればこれを読み出す。

2.2 ハードウェア

インタフェイス方式を検討し、表1に示す4項目を定めた。

(1) VAX11/780からSBCへの指令に関するインタフェイスの方式は、総てVAX11/780からSBCへの割込み方式とした。

(2) SBCからVAX11/780への処理要求に関するインタフェイスの方式は、総て VAX11/780による状態監視方式とした。

(3) VAX11/780とSBC間のデータ転送に関するインタフェイスの方式は、総て DMA方式とした。また、DMA転送は、DMA専用LSI(日立HM68450)を使用して行なうことにした。

(4) SBC間のデータ転送に関するインタフェイスの方式は、総てPIO方式とした。

VAX11/780-SBC間のインタフェイス動作はレジスタへのデータ書き込み/読み出しによって実現される。表2にレジスタ仕様を示す。CDTR1は、VAX11/780とSBCのデータ転送に使用する。CDTR2は、VAX11/780とSBC間のデータ転送をDMA方式で行なう場合の、DMA転送バイト数、 DMA転送するデータ格納番地をVAX11/780からSBCに知らせる目的に使用する。CSTR1はDMA転送準備完了、データ転送完了などをSBCからVAX11/780に知らせる目的に使用する。CSTR2は、データ転送要求を出しているSBCを識別する目的に使用する。 CSTR3は、VAX11/780が、共有メモリのリード/ライト動作以外の指令をSBCに対して発行した場合の、SBCの処理完了を確認する目的に使用する。CCTRは、VAX11/780が、指令を送るSBCを指定する目的に使用する。CCMRは、VAX11/780の

表1 インタフェイス方式

項目	インタフェース方式
VAX11/780からSBC68Kへの指令	VAX11/780からSBC68Kへの割り込み
SBC68KからVAX11/780への指令	VAX11/780による状態監視
VAX11/780とSBC68K間のデータ転送	DMA
SBC68K間のデータ転送	PIO

表2 レジスタ仕様

レジスタ名	略称	ビット幅(ビット)	仕様
データレジスタ1	CDTR1	8	データ転送に使用する。
データレジスタ2	CDTR2	8	DMA制御情報転送に使用する。
ステータスレジスタ1	CSTR1	8	SBC68Kのステータス確認に使用する。
ステータスレジスタ2	CSTR2	8	データ転送要求を出しているSBC68Kの識別に使用する。
ステータスレジスタ3	CSTR3	8	SBC68Kの処理完了確認に使用する。
コントロールレジスタ	CCTR	8	VAX11/780が指令を送るSBC68Kの指定に使用する。
コマンドレジスタ	CCMR	8	VAX11/780の指令を格納する。

指令を格納する目的に使用する。

VAX11/780とSBCのインタフェイス指令に関しては表3に示す8種類を設けた。共有メモリライトは、VAX11/780から共有メモリへのデータ転送動作の指令である。共有メモリリードは、共有メモリ VAX11/780へのデータ転送動作の指令である。ローカル・メモリ・セットは、共有メモリからSBCのローカル・メモリへのデータ転送動作の指令である。共有メモリ・セットは、SBCのローカル・メモリから共有メモリへのデータ転送動作の指令である。評価用データ要求は、アーキテクチャ解析用データをSBCのローカル・メモリから共有メモリに転送する動作の指令である。ローカル・メモリ・ダンプ要求は、SBCのローカル・メモリから共有メモリへのデータ転送の指示である。イニシャライズ要求は、SBCのテーブル類の初期化指令である。強制終了要求は、SBCのリダクション処理の終了指令である。

IMバッファへのアクセスに際しては、ロック機構を用いて一定期間アクセスを独占する。このロック機構とデータ転送は、SBC上のソフトウェアで実現した。

専用ハードウェアは、図2のGPIBインタフェイスおよびIMボードインタフェイスのみであり、これらを、標準基板(IC最大80個搭載可能)4種18枚で実現した。なお、これ以外の部分については、市販品を使用して組み立てた。表4に、設計、製作した専用ハードウェアの諸元を示す。

GPIBIF1基板は、CDTR1、CSTR1、CCTR、CCMRの4レジスタおよびその制御論理を搭載している。

GPIBIF2基板は、CSTR1、CSTR2の2レジスタおよびその制御論理、DMA制御論理およびバス権制御論理を搭載している。

CONECT基板は、共有メモリ実装のマザーボードとSBC実装のマザーボード間で送受信する信号のドライバのICを搭載している。

表3 VAX11/780→SBC インタフェイス指令

VAX11/780の指令	内 容
共有メモリ ライト	VAX11/780から共有メモリへのデータ転送
共有メモリ リード	共有メモリからVAX11/780へのデータ転送
ローカルメモリセット	共有メモリからSBC68Kのローカルメモリへのデータ転送
共有メモリセット	SBC68Kのローカルメモリから共有メモリへのデータ転送
評価用データ要求	SBC68Kの評価用データ格納エリアから共有メモリへのデータ転送
ローカルメモリダンプ要求	SBC68Kのローカルメモリから共有メモリへのデータ転送
イニシャライズ要求	SBC68Kのテーブル類の初期化
強制終了要求	SBC68Kのリダクション処理の強制終了

ISMBDIF基板は、SBCへの割込制御論理、共有メモリ・アクセス制御論理、SBCのバス権制御論理を搭載している。

2.3 ソフトウェア

IMの各機能は、SBC上のソフトウェアで実現した。

PPU処理系は、Prolog処理系と、Concurrent Prolog処理系で構成される。各Prolog処理系は、更に機能毎に以下のようなルーチンからなる。

(1) 他PPUからのパケット処理ルーチン(PPKT)
(2) UUからのパケット処理ルーチン(PUPKT)
(3) レディ・プロセス処理ルーチン(PRDPR)
(4) プロセス間同期処理ルーチン(Concurrent Prolog処理系のみ)

他PPUからのパケット処理ルーチンは、他のIMのPPUからネットワークを介して送られて来るパケットを処理する。

図4にPPU-PPU間パケットの形式を示す。図5には、パケットを形成する一語の基本形

表4 ハードウェア諸元

基板名	IC数	主な搭載論理
GPIBIF1基板	58	CDTR1 CSTR1 CCTR CCMR
GPIBIF2基板	63	CSTR2 CSTR3 DMAコントローラ
CONECT 基板	14	バスドライバ
ISMBDIF基板	56	割り込み制御

相対番地	タグ部	データ部
0	INT	パケット長
1～2		シミュレーション・クロック
3	INT	送出元IM番号
4	INT	パケットタイプ識別子
5～n	TAG	以下のいずれか ・ 子プロセス生成要求情報 ・ 子プロセス成功情報 ・ 子プロセス失敗情報 ・ メッセージボード・リード要求情報 ・ メッセージボード・ライト要求情報 ・ アクティベート要求情報 ・ リダクション要求情報

図4 PPU-PPU間 パケット形式

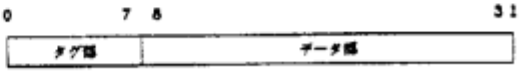


図5 基本データ形式

式を示す。一語は32ビットからなり、先頭の8ビットがタグ部で、残りの24ビットがデータ部である。Prolog処理系のパケットには、子プロセス生成要求、子プロセス成功報告、子プロセス失敗報告がある。Concurrent Prolog処理系のパケットとしては、リダクション要求、子プロセス成功報告、子プロセス失敗報告、Message Boardリード要求、Message Boardライト要求、activate要求がある。

UUからのパケット処理ルーチンは、同一IMのUU処理系から送られてくるユニフィケーション結果パケットを処理する。UU－PPU間パケットは、同一IM内で送受するのでNET処理系は介さない。Prolog処理系では、ユニフィケーション成功パケットは、UUからのパケット処理ルーチンによって子プロセス要求パケットとして他IMへ分配する。Concurrent Prolog処理系では、ユニフィケーション成功パケットは、総て自己のプロセスとしてレディ・プロセス・キューへ登録する。これによって、コミット処理のためにIM間の通信量が増加するのを防いでいる。

レディ・プロセス処理ルーチンは、レディ・プロセス・キューに登録されているプロセスからANDリテラルを取り出し、UU処理系をコールしてユニフィケーション処理を行なう。この際、Concurrent Prolog処理系では、並列AND関係の複数のリテラルを分配戦略に従って、他IM内のPPUへ分配する。

表5に、Nucleus、NET、PPU、UUの各処理系の開発規模(ソース・プログラム行数)を示す。Nucleusはアセンブリ言語とC言語で、その他の処理系は総てC言語で記述されている。

ベンチマーク・プログラム(4-Queens)の実行時における試作処理系の実行ステップ数(MC68000機械語対応)を測定した結果を表6に示す。今回の試作では、シミュレーション専用機ということで、処理効率を重視せずに設計したため、1推論あたり約1万ステップを要している。今後、データ構造、処理アルゴリズムの改良等で、1桁程度の改善が達成できる見込みである。

表5　作成プログラム・ステップ数

処理系名称	ステップ数　(KS)
ＰＰＵ処理系	5．5
ＵＵ処理系	9．3
ＮＥＴ処理系	0．5
Nucleus	0．8
言語処理系	0．9
合　計	17．0

3．　シミュレーション方式

本ハードウェア・シミュレータは本格的なベンチマークによるＰＩＭ－Ｒのアーキテクチャの評価を目指しており、シミュレーションの高速化は不可欠である。そこで、本ハードウェア・シミュレータではイベント駆動方式を採用し、シミュレーション実行時におけるアイドル時間の省略を図った。

タイマに関して、システム全体を統一的に計時するTOD (Time of the Day Clock)は有しておらず、各IM内にソフトウェア・タイマを設けている。タイマは、ある機能単位の処理を実行する毎に、予めパラメータとして指定された値を加えることによって更新される。他のIMに対してパケットを送信する際には、タイマの値に、パラメータ指定されたネットワーク遅延時間を加えた値を、到着時刻として送出パケットに付加しておく。送り先のIMでは、パケットの取り出しの際に、この到着時刻と自己のタイマの値を比較することによって、計時を制御する。IMバッファ内のパケットはキュー管理されており、到着時刻の早い順にパケット・キューが作られている。このキュー管理は、パケットを送出する際、送出元のIMが行なう。

図6に、計時の制御に関する部分のフローを示す。IM内タイマの値がパケット・キューの先頭パケットの到着時刻より小さい限り、レディ・プロセスの処理を続ける。即ち、未来に到着するはずのパケットは処理しない。レディ・プロセスが無く、かつ、先頭パケットの到着時刻がタイマ値より大きかった場合には、パケット到着時刻をタイマ値にセットし、パケットを取り込んで処理を続ける。

表6　Prolog 処理の動的ステップ数詳細

レベル1			レベル2			レベル3		
処理	ステップ数	重み	処理	ステップ数	重み	処理	ステップ数	重み
PPROLG	9743	1	PPRT1	3146	1	子プロセス生成	2526	0.46
						True処理	6248	0.25
						Fail処理	1386	0.29
			PUPR1	4461	1	IMパケット生成	858	1
						ユニフィケーション	3582	1
			PUPRT1	2136	1	自ＩＭへの子プロセス生成	1750	0.08
						他ＩＭへの子プロセス生成	2075	0.33
						True処理	2302	0.5
						Fail処理	1556	0.09

この際のタイマ値とパケット到着時刻の差はIMのアイドル時間として積算される。

本ハードウェア・シミュレータには、「実行の順序性が守られない。」という、イベント駆動方式に特徴的な問題点が存在する。これは次のように理解される。実際のパケット転送が実行される時刻はハードウェア・シミュレータの物理的な実行時間によって決定される。しかしながら、シミュレーション上のパケット到着時刻は物理的実行時間とは無関係に決められており、かつ、複数のIMが同一のIMにパケットを転送するために、シミュレーション上のパケット到着時間が早いパケットが物理的に後から到着する事態が生じる。パケット到着時刻の遅いパケットがパケット・キューに残っている場合にはキュー管理によって順序性は守られるが、先に処理されてしまった場合には、もはやシミュレーション上の時刻に従った処理順序は守ることができない。プロセスは、それが発生したIMで処理されることもある。このプロセスの処理と他のIMから送られてくるプロセスの処理の順序性も同様に守ることができない。

本ハードウェア・シミュレータによる評価にあたっては、上に述べた問題点に注意を払った。Prologで記述したソフトウェア・シミュレータ[9](クロック駆動方式使用)の結果と比較検討し、 Concurrent Prologのクイック・ソート・プログラム(10要素)をベンチマークとした台数効果シミュレーションにおいて、非常に良い一致を確認している。なお、処理順序性逆転の事態の発生状況を定量的に把握するために、今後、発生回数を計数する予定である。

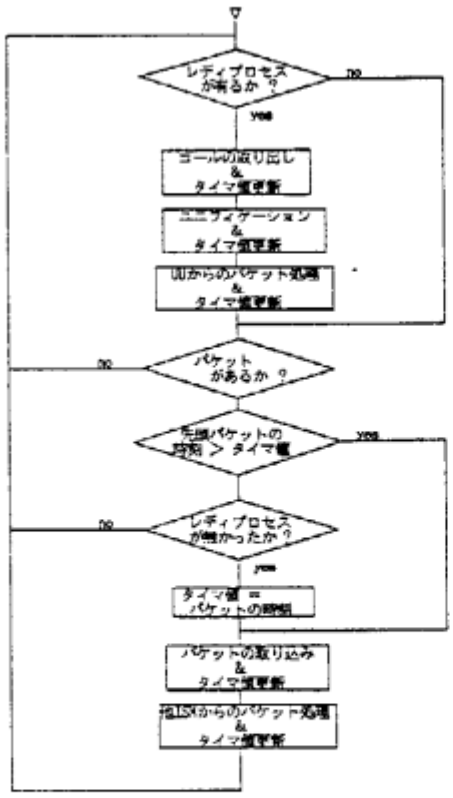


図6 タイマ処理フロー

4. 評価結果

試作したハードウェア・シミュレータ上でテスト・プログラムを走行させ、プロセッサ・エレメント台数効果等のデータを収集した。並列推論マシンでは、プロセスの分配が重要になる。シミュレーションにおいては、これに関連したネットワークとプロセス分配方式に焦点を合わせ、ネットワーク・スループットの台数効果への影響、プロセス分配方式のプロセッサ・エレメント稼働率分布への影響を調べた。また、プロセッサ・エレメント間の交信量の低減のための基礎データとして、ネットワークを通過するパケットの統計的データも収集した。

4.1 シミュレーション条件

(1) 転送パケットの衝突のないネットワークを仮定する。

(2) 各IMが有する入出力バッファは十分に大きいとし、ハードウェア・シミュレータの物理的バッファの容量オーバーによる待ち時間は計数しない。

(3) Prologではルールに対してユニフィケーションが成功したときに、Concurrent PrologではAND-fork時に、新しい子プロセスを各IMに分散させるが、ネットワーク方式によって、次の分配方式を採用した。

鎖状ネットワーク：自IM→右IM→左IM→自IM→……

メッシュ状ネットワーク [7]：自IM→東IM→南IM→西IM→北IM→自IM→……

4.2 シミュレーション結果

Prolog処理系に関して、Queensプログラム(4-Queens、6-Queens、7-Queens)を走行させ、データを収集した。図7に、6-Queensのソース・リストを示す。

図8、9、10、11、12に、6-Queensの場合のIM台数効果、IM稼働率、IM稼働率の時間変化を示す。図8からは、メッシュ方式ネットワークによって、IM台数≦8では、処理性能が、台数に応じて改善されていることがわかる。この結果は、ＰＩＭ－ＲがPrologプログラムに内在する並列性を引き出していることを示していると考えられる。また、図8は、鎖方式ネットワークでは、5～6台で処理性能向上に飽和が見られることも示している。その原因は、図9に見られるとおり、各IMの稼働率の低下にある。一方、メッシュ方式では、図10が示すように、全IMの平均稼働率も高く、稼働率分布も小さい。鎖方式ネットワークにおける稼働率の低下は次のように考えられる。図10が示すように、CMから与えられたゴールを処理するIM(IM#0)から離れたIMでは、初期の稼働率が低い。即ち、ゴールが分配されてくる迄に時間を要している。今回のシミュレーションでは、4.1節に述べたような分配方式を採用した。この方式では、たとえ、右方向に稼働率の低いIMが存在しても1/3は左方向のIMに分配する。稼働率の高いIMは、その分だけ多くのプロセスを生成し、その結果として左右のIMからプロセス生成パケットを送りつけられることになり、そのまた結果として、高い稼働率を維持することになる。このように今回採用した分配方式には、プロセスの伝搬速度が低いという問題があり、IM#0から右のIMでは右方向にしかプロセスを分配しないといったような改善が必要である。図11からわかるように、IM#0、#1、#2、#7の4個のIMで稼働率が高い。このことから、鎖方式ネットワークでは、～4個のIM鎖について、プロセス分配を均質化できると考えることができる。2×2および2×4のメッシュ構造で、処理性能向上が得られたのは、このためと考えられる。

6-Queensにおけるネットワーク通過パケットの種類別個数とそのパケット長を鎖方式ネットワークについて表7に示す。6-Queensの総リダクション回数は6,353であるから、約1.8reductionに1回の割合でパケットの転送が生じていることになる。図13にPP容量と総プロセス数の関係を示す。同図からは、ガーベジ・コレクションを行なわない場合でプロセス当り63ワード、ガーベジ・コレクションを実行した場合でプロセス当り6.3ワードのPPが必要となることがわかる。

Concurrent Prolog処理系に関しては、Quick-Sortプログラムをベンチマークとした。図14にQuick-Sortのソースリストを示す。

図15,図16に、Quick-SortのIM台数効果、IM稼働率を、表8にネットワーク通過パケットの種類別個数とそのパケット長を示す。

図14のソースプログラムからわかるとおり、Quick-SortプログラムにおいてAND-forkが起きるのは、'qsort'の第１クローズのボディ部だけであり、しかも、第２、第３のリテラルは第１リテラルの 'partition'によってactivateされるまで、第４リテラルは、第２および第３リテラルによってactivateされるまでsuspendしている。 そのため、Quick-Sortの並列性は低く、約4と考えられる。図16の稼働率分布は、8台構成の場合でも約4台にしかプロセスが分配されていないことを示している。図15のIM台数効果が図8の6-Queensのそれに比して低いのは、Quick-sortプログラムの並列性の低さによる。

表8からは、IM台数が増につれて、ネットワークを通過するパケット数が増していくことがわかる。これは、IM台数が増加によって、message board関連のパケットが増加することによる。図14のQuick-Sortプログラムの総リダクション回数は1387であるから、IMが8台の場合、約0.6リダクションに1回の割合でパケットの転送が生じることになり、しかも、転送パケットの約90%がmessage board関連のパケットである。このことから、Concurrent Prolog処理系では、message boardアクセスのためのネットワーク・スループットの確保が重要になることがわかる。

以上の考察は6-Queensプログラム,Quick-sortプログラムに限られている。したがって、実行するプログラムによっては異なる結論も得られよう。今後、ベンチマークを増やして、ＰＩＭ－Ｒの平均像を得る予定である。

[プログラム]

```
queens(X,Y) : - queens1(X,[ ]).
queens1([ ],Y).
queens1(X,Y) : - select(U,X,V),check(U,Y),
                 queens1(V,[U|Y]).
select(X,[X|Y],Y).
select(U,[X|Y],[X,V]) : - select(U,Y,V).
check(P,Q) : - check1(P,Q,1).
check1(_,[ ],_).
check1(P,[Q|R],N) :- nodiag(P,Q,N),M is N+1,
                     check1(P,R,M).
nodiag(P,Q,N) : - T1 is P+N,T2 is P-N,T1=\=Q,
                  T2=\=Q.
```

図7　6-Queens プログラム・ソース・リスト

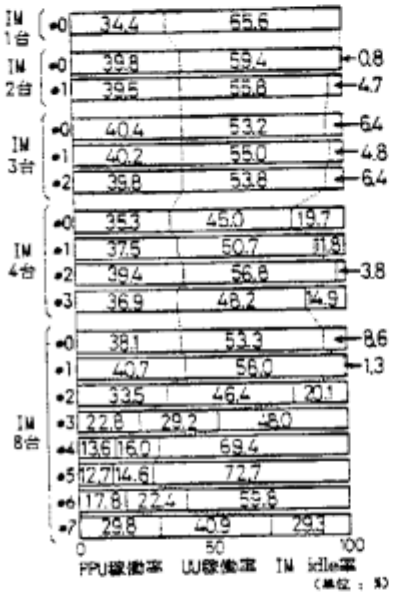
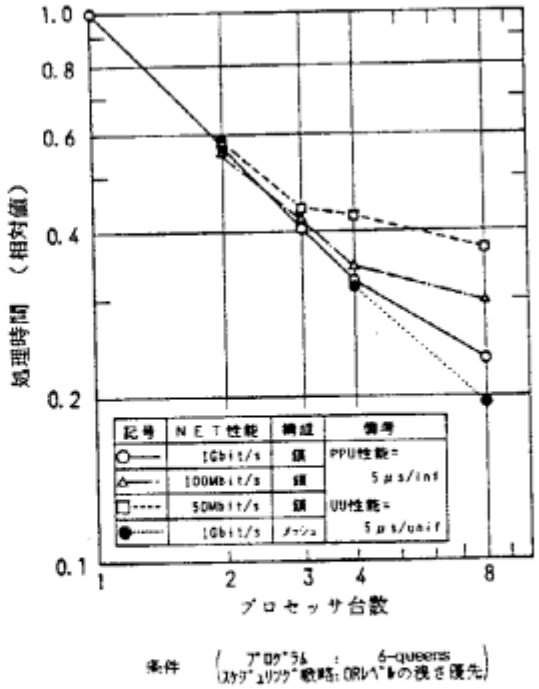


図9　IM 稼働率比較　（アプリ：6-queens　構成：8IM　スケジュリング戦略：ORレベルの浅さ優先　PPU/UU/NET：5μs/5μs/1Gb）



条件　（アプリ：6-queens　スケジュリング戦略：ORレベルの浅さ優先）

図8　台数効果

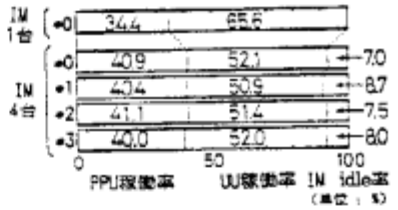


図10　IM 稼働率比較　（アプリ：6-queens　構成：4IM　スケジュリング戦略：ORレベルの浅さ優先　PPU/UU/NET：5μs/5μs/1Gb）

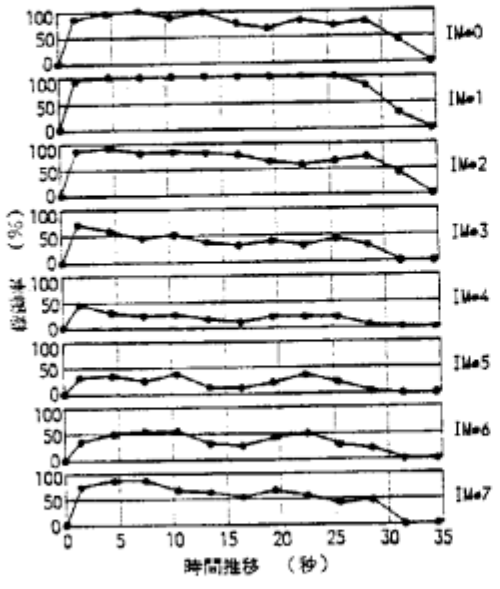


図11　稼働率推移（1）　（アプリ：6-queens　構成：8IM　スケジュリング戦略：ORレベルの浅さ優先　PPU/UU/NET：5μs/5μs/1Gb）

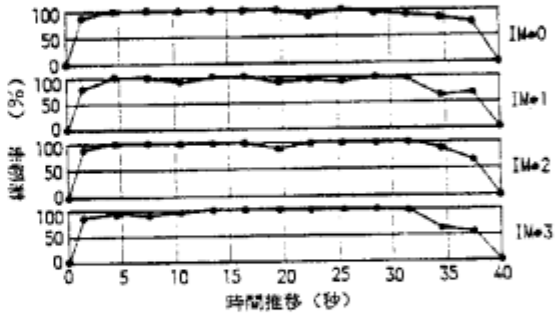


図12　稼働率推移（2）　（アプリ：6-queens　構成：4IM　スケジュリング戦略：ORレベルの浅さ優先　PPU/UU/NET：5μs/5μs/1Gb）

表7　送出パケット・データ

（プログラム：6-queens，転送性能 =1,000 Mbit/sec.，構成 = 順，スケジュリング戦略 = ORレベルの浅さ優先）

台数	ISM #	総パケット数	総パケット長 (W)	子プロセス生成パケット数	子プロセス生成パケット長 (W)	成功パケット数	成功パケット長 (W)	失敗パケット数	失敗パケット長 (W)
2	0	1,754	56,392	801	36,205	516	16,691	437	3,496
	1	1,758	56,946	821	36,856	514	16,706	423	3,384
3	0	1,169	37,277	513	23,289	357	11,596	299	2,392
	1	1,164	37,977	545	24,686	339	11,051	280	2,240
	2	1,169	37,596	564	25,379	302	9,793	303	2,424
4	0	805	25,207	354	16,140	230	7,299	221	1,768
	1	929	30,862	460	20,599	266	8,639	203	1,624
	2	935	30,933	460	20,694	262	8,535	213	1,704
	3	821	25,498	348	15,798	244	7,868	229	1,832
8	0	651	21,394	326	14,610	169	5,536	156	1,248
	1	687	22,378	330	14,851	189	6,181	168	1,344
	2	595	19,370	290	12,833	165	5,417	140	1,120
	3	378	12,040	160	7,235	125	4,061	93	744
	4	213	6,624	81	3,742	77	2,442	55	440
	5	190	5,913	74	3,453	64	2,044	52	416
	6	305	9,752	130	6,029	96	3,091	79	632
	7	495	15,890	230	10,302	140	4,588	125	1,000

(註) パケット長の1W=4バイト

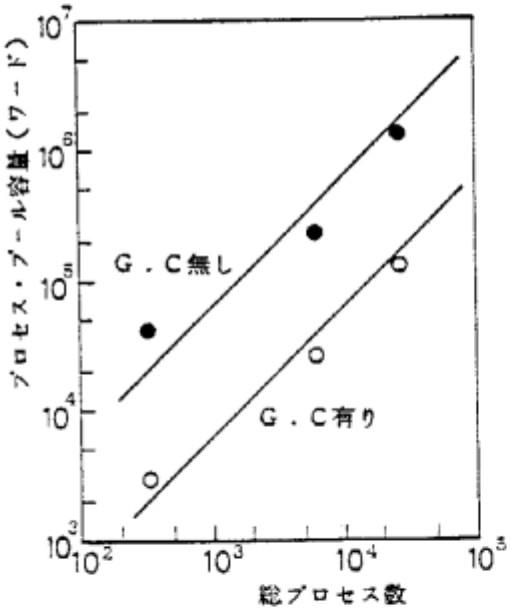


図13　プロセス・プール容量

〔プログラム〕

```
go :- quicksort([27,74,17,33,・・・,59,8])//
      screen(S?).
screen([ ]) :- display([ ]) | true.
screen([E|F] :- display(E) | screen(F?).
quicksort(U,S) :- true | qsort(U,S).
qsort([X|Un],S) :- true | partition(Un?,
           X,Smaller,Larger) // qsort
           (Smaller?,S1) //qsort(Larger?,S2)
           // lappend(S1?,[X|S2?],S).
qsort([ ],[ ]).
partition([X|Xs],A,S,[X|L]) :- A ＜ X |
                    partition(Xs?,A,S,L).
partition([X|Xs],A,[X|S],L) :- A ≧ X |
                    partition(Xs?,A,S,L).
partition([ ],_,[ ],[ ]).
lappend([W|X],Y,[W|Z]) :- lappend(X?,Y,Z).
lappend([ ],X,X).
```

図14　Quick-sort プログラム・ソース・リスト

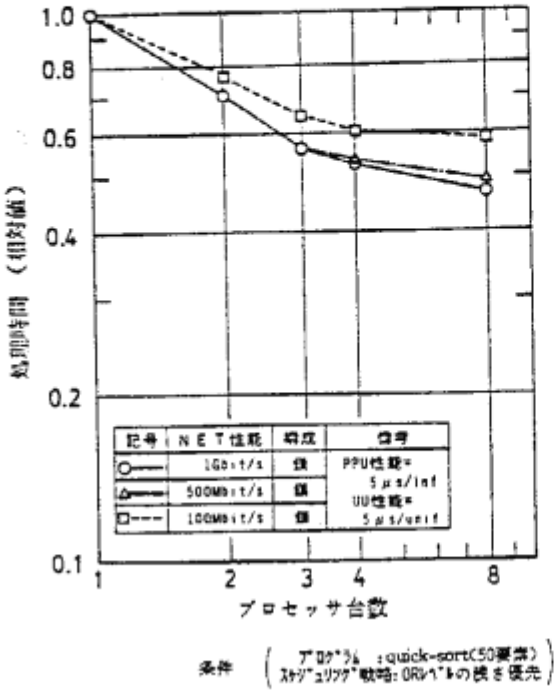


図15　台数効果

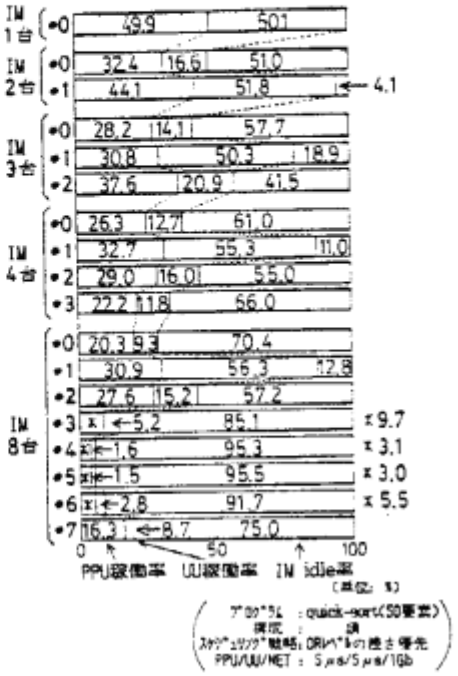


図16　IM 稼働率比較

表8　送出パケット・データ

（プログラム:Quick-sort，転送性能＝1 Gbit/sec，構成＝網，スケジュリング戦略＝ORレベルの浅さ優先）

台数	IM #	総パケット数	総パケット長	成功パケット数	成功パケット長	ゴールパケット数	ゴールパケット長	MBリードパケット数	MBリードパケット長	MBライトパケット数	MBライトパケット長	activateパケット数	activateパケット長
2	0	910	15,559	69	3,350	66	3,181	423	3,807	65	883	288	4,338
	1	905	16,863	65	3,254	69	3,490	288	2,592	60	799	423	6,728
3	0	645	10,833	42	2,088	49	2,417	310	2,790	38	518	206	3,010
	1	852	12,407	42	2,211	50	2,582	184	1,656	41	562	335	5,376
	2	749	12,211	49	2,514	34	1,622	330	2,970	53	733	283	4,572
4	0	592	9,712	36	1,799	41	2,035	295	2,655	31	421	189	2,802
	1	678	12,140	31	1,669	42	2,200	208	1,872	28	379	369	6,020
	2	590	9,307	35	1,679	21	1,002	268	2,412	37	492	229	3,722
	3	509	8,383	31	1,535	29	1,384	217	1,853	31	449	201	3,062
8	0	431	6,918	26	1,238	30	1,509	228	2,052	23	309	124	1,810
	1	601	10,841	24	1,342	37	1,962	175	1,575	22	314	343	5,648
	2	523	8,155	25	1,544	18	858	239	2,151	25	342	216	3,560
	3	188	3,140	10	538	13	620	77	693	9	129	79	1,160
	4	81	1,293	9	422	2	94	38	342	11	141	21	294
	5	77	1,269	6	272	5	238	32	288	6	83	28	388
	6	137	2,349	11	480	10	476	56	504	11	139	49	750
	7	400	6,221	20	1,001	16	764	179	1,611	21	311	164	2,534

（註）パケット長の1W＝4バイト

5. おわりに

リダクションの概念に基づく並列推論マシンＰＩＭ－Ｒのハードウェア・シミュレータを試作し、プロセッサ・エレメント台数効果等のデータを収集した。その結果、ＰＩＭ－Ｒはプログラムに内在する並列性を引き出せること、プロセッサ・エレメントの台数増加による処理性能向上には、プロセスの伝搬速度の高いプロセス分配方式が重要であることを確認した。

今後、ネットワーク方式および Network Nodeでの子プロセスの動的分配方式の考案によって、プロセスの伝搬速度を高め、100台規模のＰＩＭ－Ｒ用の方式に改善していく予定である。リダクション処理負荷の低減についても、データ形式の改良、共有メモリの導入等によって改善を図っていく予定である。そして、実際の並列環境下での実行という特徴を生かして、並列推論マシン・ソフトウェア・ユーティリティ構築のためのデータ収集も進める予定である。

最後に、本研究の推進にあたって日頃御指導いただいた日立中研 堀越 彌 前第8部長，千葉 常世 第8部長 および ＩＣＯＴ 村上 国男 前第１研究室長，内田 俊一 第４研究室長に深謝する。

参考文献

1) T.Motooka et al. ;"The Architecture of a Parallel Inference Engine -PIE- ", Proc.of Int. Conf. on Fifth Generation Computer Systems 1984,ICOT,pp479-488 (1984)

2) N.Ito et al.; "Parallel Inference Machine Based on the Data Flow Model",Proc. of the International Workshop on High Level Computer Architecture 84,pp4.31-4. 40(1984.5)

3) K.L.Clark et al. ; "PARLOG : Parallel Programming in Logic",Research Report DOC 84/4,Dept. of Computing, Imperial College London(1984)

4) E.Y.Shapiro ; "A Subset of Concurrent Prolog and Its Interpreter",ICOT Technical Report TR-003(1983)

5) 尾内 他 ;"並列推論マシンＰＩＭ－Ｒのアーキテクチャ",情報処理学会第30回全国大会6C-6(1985)

6) R.Onai et al. ; "Architecture of a Reduction-Based Parallel Inference Machine : PIM-R",New Generation Computing, vol.3, No.2(1985)

7) 尾内 他 ;"リダクション方式並列推論マシンＰＩＭ－Ｒのアーキテクチャ",Logic Programming Conference '85(1985) 2.1

8) 益田 他 ; "並列推論マシンＰＩＭ－Ｒの構造体メモリの一構成法",情報処理学会第30回全国大会6C-5(1985)

9) 清水 他 ; "並列推論マシンＰＩＭ－Ｒにおけるプロセス内部表現 ",情報処理学会第30回全国大会6C-7(1985)

10) 尾内 他 ;"並列推論マシンＰＩＭ－Ｒのソフトウェア・シミュレーション",情報処理学会第30回全国大会6C-9(1985)