記号処理 32－3
（1985. 6. 18）

連想メモリを用いたPrologマシンの構成と処理アルゴリズム

長沼 次郎　　小倉 武　　山田 慎一郎
NTT 厚木電気通信研究所

1．はじめに

推論等の知識情報処理に適した言語 Prolog の処理系の高速化が望まれている。汎用計算機上にソフトウェアで実現されている処理系では，1)メモリアクセス，データタイプの判定等の負荷が大きい，2)処理アルゴリズム自体が逐次的な汎用ハードウェアを前提としており並列化が困難である，等の問題を有しており，その高速化には限界がある。このため，専用マシン（Prologマシン）の研究が精力的に行われている。Prologマシンの研究は大きく分類して，1個のプロセッサによる逐次型のもの(1)(2)(3)と，複数個の要素プロセッサによる並列型のもの(4)(5)(6)がある。ここでは逐次型Prologマシンについて議論する。並列型の高速化においても，個々の要素プロセッサの高速化は必要不可欠であるため，逐次型Prologマシンの処理能力を追求することは極めて重要である。

現在までに提案されている逐次型Prologマシンは，基本的にDEC-10 Prolog の仮想マシン(7)をファームウェア化あるいは専用ハードウェア化したものである。このため，上記1)の問題は解決しているが，2)の問題については逐次的なハードウェアを前提とした処理アルゴリズムを踏襲しており，その高速化には汎用計算機上のものと同様な限界がある。また，コンパイラの使用を前提として高速化を図ったPrologマシンも実現されつつあるが(3)，その場合，Prolog言語の持つ会話性は失われる。一方，Prologの処理における連想処理をハッシュ手法で実現することにより，DEC-10 Prologの仮想マシンからの脱却を図ったProlog処理系も構築されている(8)。また，ユニフィケーションを高速に実行するハードウェアアルゴリズムの検討も行なわれている(9)。

これに対して，我々は，Prologの処理に含まれる連想処理とPrologの処理自体の簡単な計算機構に着目し，連想メモリと専用ハードウェアの適用によるパイプライン処理により，インタプリタのままで高速に走る新たな構成に基づくPrologマシンの検討を進めてきた(10)。本Prologマシンには，連想メモリと専用ハードウェアを活用した新たな処理アルゴリズムを採用し，制御の簡単化と処理の高速化を追及している。

ここでは，連想メモリを用いたPrologマシンの設計思想，アーキテクチャとその特徴，処理アルゴリズムおよび性能予測について述べる。本Prologマシンの予測性能は，同一マシンサイクルを仮定した場合のコンパイラ型のPrologマシンの性能に匹敵しており，連想メモリと専用ハードウェアの適用によるパイプライン処理の効果が現れている。各種最適化により，一層の改善が可能である。

2．設計思想

Prologの処理は，非決定的な節の起動と引数のユニフィケーションの処理から成り立っている。このため，それらを2つの専用の処理部に分割することにより，節の起動と引数のユニフィケーションが並列にパイプライン処理できる(10)(11)。さらに，各処理部の内部処理は，スタック機構，連想機構，データ判定機構等の単一処理レベルでは比較的簡単な処理が複合的に組み合わされて実現される。このため，これらの内部処理機構を専用化して機能ブロックに分割し，これに適した処理アルゴリズムを導入することにより，各内部処理単位毎のパイプライン処理が可能となる。このようにPrologの処理を2つの処理部と各処理部内の2階層にパイプライン処理するマシン構成，処理アルゴリズムを導入することにより，より一層の高速化が図れる可能性がある。

しかし，このようなパイプライン化したPrologマシンは，汎用計算機と類似したマシン構成では実現困難である。汎用的なマシン構成では，複数のスタックを1次元の主記憶上に割り付けるため，Prologの内部処理の大部分を占める複数のスタック操作をパイプライン化できない。また，複数のスタックを専用化したとしても，従来のように物理的なスタックポインタで複数のスタックを制御する処理アルゴリズムでは，各スタックのスタックポインタを集中

的に管理せざるを得ない。パイプライン化を図るためには，複数のスタックの分散制御の実現が必要である。このため，パイプライン化したPrologマシンを実現するためには，パイプライン処理に適した新たなマシン構成，処理アルゴリズムが必要不可欠である。

一方，連想メモリを用いることにより，連想機構とともに，パイプライン化したPrologマシンに必要不可欠な分散制御可能なスタックを容易に実現できる(10)。

我々は，これらの点に着目し，知識情報処理への応用を意図し開発を進めてきた連想メモリ(12)の適用と，汎用計算機の構成にとらわれない自由な構成に基づく専用ハードウェアの適用を前提として，制御の簡単化と処理の高速化を追求したPrologマシンの検討を行った。本Prologマシンの設計思想を以下に要約する。

1) Prologの処理の分割と専用ハードウェア上でのパイプライン処理による高速化の追求
2) 連想メモリの適用による制御の簡単化と処理の高速化の追求
3) 専用ハードウェアと連想メモリの機能を活用した新たな処理アルゴリズムの採用

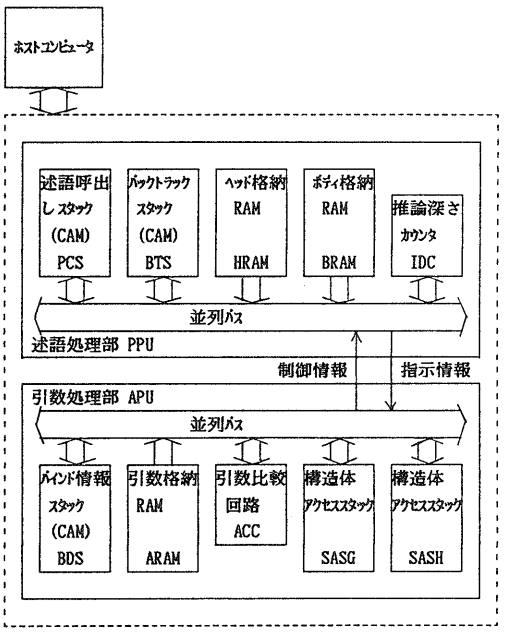


図1．Prologマシンの機能ブロック構成

3．アーキテクチャとその特徴

3.1 機能ブロック構成

本Prologマシンの機能ブロック構成を図1に示す。本Prologマシンは，先で述べたように，節の起動を行う述語処理部と引数のユニフィケーションを行う引数処理部の2つの処理部から成る。本構成により，Prologの処理における節の起動と引数のユニフィケーションを並列にパイプライン処理する。また，各処理部内の機能ブロック間においてもパイプライン処理が可能な構成を追求しており，より一層の高速化が図れる。両処理部におけるPrologに不可欠なスタック機構と連想機構を連想メモリを用いて実現しており，制御の分散化，簡単化を図っている。本Prologマシンの全体の処理は，述語処理部と引数処理部の間で簡単な情報の授受を行いながら，各処理部の動作を繰り返すことにより実現する。2つの処理部間のインタフェースは，4つのポインタ類からなる指示情報と1ビットの情報からなる制御情報のみであり，通信負荷は小さい。各処理部のスタックを連想メモリを用いて構成するため，推論深さというプログラムの実行過程における意味論的な抽象度の高い情報によって各処理部間のスタックの分散制御が可能となる。

3.2 特徴的なアーキテクチャ

本Prologマシンのアーキテクチャの特徴は，処理の分割によるパイプライン化とスタック機構，連想機構への連想メモリの適用である。これらは，いずれも本Prologマシンの高速化に大きく寄与する。本Prologマシンのアーキテクチャの特徴を表1にまとめる。

表1．　アーキテクチャの特徴

アーキテクチャの特徴	
処理の分割によるパイプライン化	節の起動とユニフィケーションの分離
	節の起動を2つのスタックで実現
	バインド情報の生成と作用のフェーズの分離
連想メモリの適用	分散的なスタックの同期の実現
	トレイルスタックの不要化
	スタックのアドレッシングの不要化

3.2.1 処理の分割によるパイプライン化

Prologの処理を分割し，その分割された各処理を専用ハードウェア上でパイプライン処理することにより，Prologの処理の高速化を図った。

(1) 節の起動とユニフィケーションの分離

Prologの処理を節の起動と引数のユニフィケーションの処理に分離し，それらを専用の処理部に受け持たせることにより，２つの処理部間のパイプライン化を図る。

(2) 節の実行制御を２つの専用スタックで実現

述語処理部においては，節の実行制御の情報を，述語呼出しのための情報とバックトラックのための情報に分離し，それぞれ専用の機能ブロック（述語呼出しスタック，バックトラックスタック）に格納した。これにより，次のゴールと選択枝の情報をそれぞれのスタックトップから取り出すことができ，簡単なスタック操作で節の実行制御が可能となる。

(3) バインド情報の生成，作用フェーズの分離

引数処理部においては，一連のユニフィケーションを，バインド情報の生成と作用の２つの処理に分割し，それぞれ専用の機能ブロックで実現した。バインド情報の生成では，ユニフィケーションの対象となる２つの引数の種類（タイプ），内容の組合せによって多方向に制御を切換えるための専用ハードウェアを設け高速化を図った。バインド情報の作用では，引数の評価（dereference:バインドリストをたどる）をハードウェア機構でサポートし，高速化を図った。

このように，Prologの処理を大きく２つの処理部に，各処理部を専用の機能ブロックに分割することにより，処理部間，処理部内の機能ブロック間の２階層のパイプライン処理を可能とし，高速化を図った。

3.2.2 連想メモリの適用

連想メモリを用いて，連想機構とスタック機構を実現する。連想機構はバインド情報の格納部で用い，スタック機構は実行制御に用いる。連想メモリの機能として，一般的な検索，読出し，書込みの機能の他に，我々が提案，実現した不要ワードを一括消去する不要ワード管理機能等を活用している(13)。

(1)分散的なスタックの同期の実現

Prologの処理は複数のスタック間の同期を取って処理を進めて行く必要がある。従来のように，RAMを用いて構成すると，物理的なスタックポインタによる集中的な管理が必要であり，分散制御が困難である。また各スタックのスタックトップアドレス等の本質的でないポインタが必要である。

このようなスタックは，格納したい情報にスタックの深さを示す情報を付与して連想メモリに格納することにより実現できる。スタックの深さを示す情報を対象として検索，読出し，消去を行うことにより，極めて簡単な制御で各スタック間で分散的な同期が実現できる。また，本質的でないポインタ類が軽減でき，スタックアクセスが高速化される。

(2)トレイルスタックの不要化

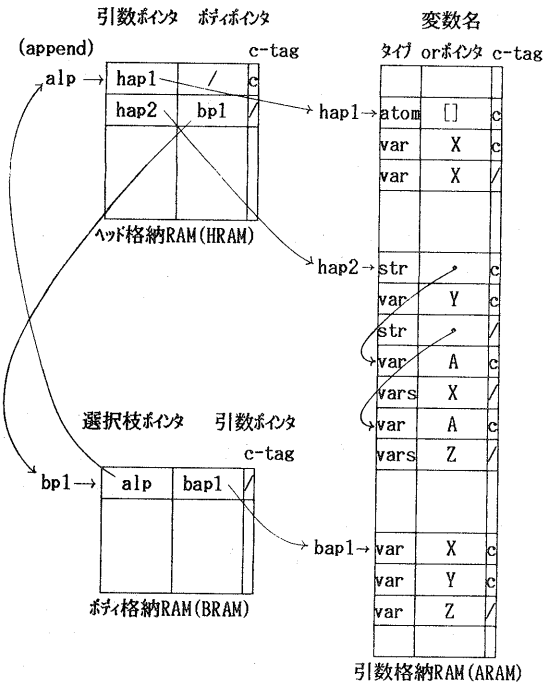
バインド情報（変数セル）を連想メモリに格納することにより，変数セルの格納場所とスタックアドレスを対応させる必要がなくなる。このため，節の起動時ではなく変数の束縛時に変数セルをスタック上に取ることが可能となる。変数の束縛時に変数セルを取ることができるため，バインド情報に変数束縛時の推論深さを付与して連想メモリに格納すれば，従来のようにトレイルスタックを用いた逐次的なundo操作を行う必要がない。バックトラックに際しては，付与した推論深さを対象として検索を行い，該当する複数のバインド情報を一括して消去するだけでよく，制御の簡単化，処理の高速化が図れる。

(3)スタックのアドレッシングの不要化

バインド情報を連想メモリに格納することにより，その時点の推論深さと変数名を連想アクセスのキーとすることにより，バインド情報を容易にアクセスすることが可能である。このため，従来のように変数セル（バインド情報）をスタック上に取る際に，スタックアドレスを求めるためのアドレッシングが不要となる。

```
append([],X,X).
append([A|X],Y,[A|Z]):- append(X,Y,Z).
```

(a) Prologのソースプログラム



(b)節の格納形式

var:変数　vars:|表記の変数　atom:ｱﾄﾑ　str:ﾘｽﾄ

図2．節の内部表現

このように，Prologの処理に不可欠なスタック機構，連想機構の部分に連想メモリを適用することにより，分散的なスタックの同期，分散処理によるスタックアクセスの高速化，新たな処理アルゴリズムの採用が可能となる。

4．処理アルゴリズム

連想メモリの適用と専用ハードウェアによるパイプライン処理を可能とする本Prologマシンの処理アルゴリズムについて述べる。節の内部表現を示した後，各処理部，各機能ブロックの処理アルゴリズムを示す。

4.1 節の格納形式（内部表現）

図2に appendプログラムの内部表現を示す。本PrologマシンではPrologのソースプログラムは，ヘッド部（選択枝）の情報，ボディ部の情報，引数の情報に分けて，それぞれヘッド格納RAM，ボディ格納RAM，引数格納RAM に格納する。これらの構造体データの格納には，次にデータが続くかどうかを示すタグ(c-tag) を用いて，メモリの連続するワードに格納するコンパクトな格納形式を採用している。

4.2 述語処理部

述語処理部の動作には，初期化，述語呼出し，バックトラックの３つの動作モードがある。それらのモードは，初期化を除いて，引数処理部からのユニフィケーションに成功したか否かを示す制御情報により，選択されて実行される。述語処理部は，引数処理部からの制御情報を受け，その内容により動作を切り換え，次のユニフィケーションのための指示情報（ユニフィケーションの対象となる２つの引数のポインタとそれらの引数の推論深さ）を生成し，引数処理部に渡す。

図１に示す述語処理部において，述語呼出しスタック，バックトラックスタックは，連想メモリを用いて構成し，各スタックに格納される情報には，推論深さが付与される。

図３に述語呼出しの場合の指示情報の生成過程の概念図を示した。述語呼出しの場合は述語呼出しスタックのスタックトップをアクセスする(1)。得られた情報を用いてボディ格納RAM アクセスすることにより呼出し元の節の情報を取り出し(2)，その情報を用いてヘッド格納RAM をアクセスすることにより呼出された側の節の情報を取り出す(3)。また，そのスタックおよび推論深さカウンタをアクセスすることにより，それぞれの節の推論深さの情報を取り出す(4)(5)。これらの操作により，指示情報を生成して引数処理部に渡す。

バックトラックの場合は，バックトラックスタックのスタックトップをアクセスすることにより実現でき，述語呼出しの場合とほぼ同様な処理となる。バックトラックの処理が，述語呼出し同様に簡単なスタック操作で実現できるのは，述語呼出しとバックトラックのための情報を分離したためである。これより，次のゴールと選択枝の情報をそれぞれのスタックトップから取り出すことが可能となる。

4.3 引数処理部

引数処理部の動作には，終了，解の生成，および通常のユニフィケションの３つの動作モードがある。それらの動作モードは，それぞれ述語処理部からの指示情報により，選択され実行される。引数処理部は，指示情報に従ってユニフィケーションを行い，そのユニフィケーションが成功か，失敗かを示す制御情報を生成し，述語処理部に渡す。このユニフィケーションに伴い，バインド情報が生成される。

図１に示す引数処理部において，バインド情報スタックは，連想メモリを用いて構成し，バインド情報に推論深さを付与した情報を格納する。バインド情報スタックは，図４に示すように，バインドリストをたどる(rr)，書込む(w)，およびある推論深さまで消去する(c)の３つの動作モードがある。rrモードは引数の評価に用いる。cモードは，バックトラックによって不要になったバインド情報の一括消去に用いる。wモードはバインド情報の格納に用いる。引数比較回路は引数の種類，内容を比較する専用回路である。

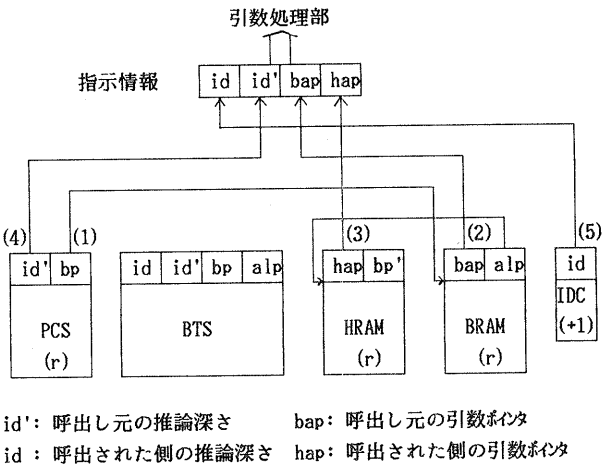


図３．指示情報の生成過程(述語呼出しの場合)

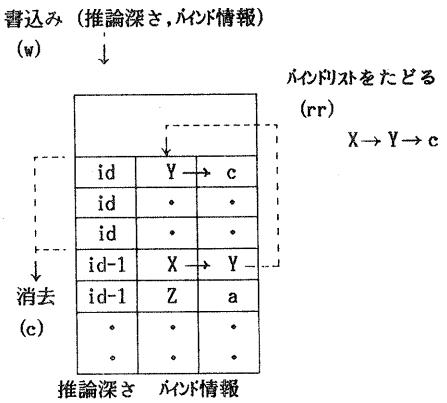
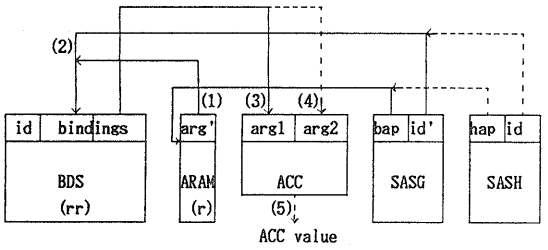


図４．バインド情報スタックの概念図

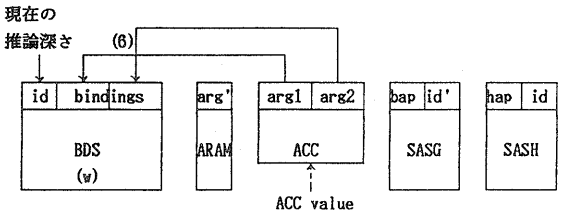
図５に引数処理部の核となる引数のユニフィケーションの概念図を示している。本Prologマシンでは，引数のユニフィケーションをバインド情報の生成と作用という２つのフェーズに分割して処理を行っている。

バインド情報の作用においては，図５(a)示すように，ユニフィケーションの対象となる呼出し元の引数ポインタにより引数格納RAM を読出し(1)，その引数の推論深さと合わせて(2)，引数の評価を行い，引数比較回路に送る(3)。引数の評価には，バインド情報スタックのrrモードを用いる。呼出された側の引数についても同様な引数の評価を行い，引数比較回路に送り(4)，次の制御の分岐の情報（ACC value）を得る(5)。

バインド情報の生成においては，図５(b)に示すように，引数比較回路からの情報により制御を切り換え，その情報に対応したバインド情報を生成し，バインド情報スタックに格納する(6)。



(a)作用のフェーズ



(b)生成のフェーズ

id': 呼出し元の推論深さ　　bap: 呼出し元の引数ポインタ
id : 呼出された側の推論深さ　　hap: 呼出された側の引数ポインタ
arg' : 評価前の引数　　arg1,arg2 : 評価後の引数
bindings : バインド情報

図５．引数のユニフィケーション概念図

５．性能予測

図６に示すリスト要素を結合するappendプログラムを用いて，本Prologマシンの概算性能（LIPS値：Logical Inference Per Second）を予測した。図１に示した本Prologマシンの機能ブロックの動作タイミングの最適化について述べ，各機能ブロックの処理速度の高速化についても簡単に触れる。

図１に示した本Prologマシンにおいて，appendプログラムの１LI(Logial Inference)を実行した時の各機能ブロックの動作回数を表２に示す。表２より明らかなように，約40ステップ（述語処理部：７ステップ，引数処理部：31ステップ）でappendプログラムの１LIが実行できる。

```
append([A|X],Y,[A|Z]):-append(X,Y,Z).
append([],X,X).

?-append([a,b,・・・,z],[],X).
```

図６．性能予測に用いたappendプログラム

表２．１LI当りの各機能ブロックの動作回数
(appendﾌﾟﾛｸﾞﾗﾑ ： 基本的な再帰の部分)

処理部名	機能ﾌﾞﾛｯｸ名	動作回数	計
述語処理部	述語呼出しｽﾀｯｸ	2	7
	ﾊﾞｯｸﾄﾗｯｸｽﾀｯｸ	1	
	ﾍｯﾄﾞ格納RAM	1	
	ﾎﾞﾃﾞｨ格納RAM	2	
	推論深さｶｳﾝﾀ	1	
引数処理部	ﾊﾞｲﾝﾄﾞ情報ｽﾀｯｸ	10	31
	引数比較回路	5	
	引数格納RAM	10	
	構造体ｱｸｾｽｽﾀｯｸ(H)	3	
	構造体ｱｸｾｽｽﾀｯｸ(G)	3	
計			38

本Prologマシンでは，述語処理部と引数処理部のパイプライン動作が可能であるため，述語処理部では引数処理部からの制御情報が渡されるより前に，先行的にその制御情報（成功または失敗）に対応した２通りの指示情報を生成することができる。両処理部の動作ステップの割合を考慮すると，述語処理部の動作を引数処理部の動作の中に埋め込むことができる。

引数処理部においては，ユニフィケーションの対象となる呼出し元と呼出された側の２つの引数に対する，２回の引数の評価（バインドリストをたどる処理）をパイプライン化する。appendプログラムを実行した場合の引数処理部の動作タイミングの例を図７に示す。図より明らかなように，各機能ブロック間のパイプライン化により，引数処理部の動作を約20ステップで実現できる。

	引数格納RAM	構造体ｱｸｾｽｽﾀｯｸ	ﾊﾞｲﾝﾄﾞ情報ｽﾀｯｸ	引数比較回路	
step	ARAM	SASG,SASH	BDS	ACC	
1	ARAM(r)				
2	ARAM(r)		BDS(rr)		
3				ACC(ss)	
4	ARAM(r)	SASG(push)			
5	ARAM(r)	SASH(push)	BDS(rr)		
6				ACC(av)	
7	ARAM(r)		BDS(w)		
8	ARAM(r)		BDS(rr)		
9				ACC(svs)	
10		SASG(pop)	BDS(w)		
11	ARAM(r)	SASH(pop)			
12	ARAM(r)		BDS(rr)		
13			BDS(rr)		
14				ACC(sv)	
15	ARAM(r)		BDS(w)		
16	ARAM(r)	SASG(emp)	BDS(rr)		
17		SASH(emp)		ACC(vs)	
18			BDS(w)		述語処理部
19					
20					

ｶｯｺ内は機能ﾌﾞﾛｯｸの動作ﾓｰﾄﾞ
ss=str-str　av=atom-var　svs=str-vars
sv=str-var　vs=var-str

図７．引数処理部の動作タイミング(appendプログラム)

このように処理部間と各機能ブロック間の２階層のパイプライン化を行うことにより，本Prologマシン全体として，約20ステップでappendプログラムの１LIを実行することができる。

ここで，図１の各機能ブロックの１ステップ当りの処理速度をデータ転送込みで500ns （この値は，現状の連想メモリLSI の動作サイクル時間：140ns [12] から考えて，現状で十分実現可能である）とすると，１LI当たり，

20(ステップ) × 500 (ns) ＝ 10 (μs)

必要であり，LIPS値に換算すると100kLIPSになる。この予測性能値は，同一マシンサイクルを仮定した場合のコンパイラ型のPrologマシンの性能に匹敵する値である[3] 。これは，本Prologマシンにおける連想メモリの適用と専用ハードウェアによるパイプライン処理の効果によるものである。

なお，バックトラックのあるプログラムにおいては，先に述べたように，逐次的なundo操作を不要化しているため，LIPS値で評価できない速度性能を発揮する。

近い将来，連想メモリ，RAM といった単体LSI レベルで２～３倍程度の速度向上（動作サイクル時間で50ns程度）は十分可能であろう。また，各機能ブロックを実現するハードウェア構成等により，各機能ブロックの処理速度を数倍程度に見積もることは可能である。これら各機能ブロックに関する最適化により，本Prologマシンの性能は，コンパイラを用いなくても，Tick等の言うコンパイラの使用を前提としたマイクロプログラム制御の逐次型Prologマシンの限界性能予測[14] に近づく。Prologのような会話型言語において，インタプリタのままで高速に走ることは，本質的な利点である。

6．おわりに

本稿では，連想メモリを用いたPrologマシンの設計思想，アーキテクチャとその特徴，処理アルゴリズムおよび性能予測について述べた。本Prologマシンは，連想メモリと専用ハードウェアの適用および新たな処理アルゴリズムの採用により，処理部間と処理部内の２階層のパイプライン化を図り，処理の高速化と制御の簡単化を追求した。

本Prologマシンはインタプリタ型であり，その予測性能は100kLIPS以上である。この値は，同一マシンサイクルを仮定した場合のコンパイラ型のPrologマシンの性能に匹敵する。さらに，各種最適化により，一層の改善が可能である。

現在，連想メモリ，RAM ，TTL等の汎用LSI を用いた本Prologマシンのプロトタイプとなる実験装置の設計を進めている。

［謝辞］

本研究を進めるに当たり，終始御指導頂いた浅岡主席担当調査役，家田記憶回路研究室長，並びに中島集積応用研究室長に感謝いたします。また，種々の有益な御討論を頂いた木村調査役，二階堂室長補佐をはじめとする研究室の方々に感謝いたします。

［参考文献］

(1)西川他：逐次型パーソナル推論マシンφの設計思想とそのアーキテクチャ ， Proc. of the Logic Programming Conference'83, 7-2, ICOT, 1983.

(2)田村他：シーケンシャルPROLOGマシンPEKのアーキテクチャとソフトウェアシステム，情報処理学会，記号処理研究会資料25-2，1983.

(3)梅村：SIMの拡張機構：高速プロセッサモジュール，第５世代コンピュータに関するシンポジウム，ICOT, May , 1985.

(4)後藤他：推論向き高並列計算機システムの基本アーキテクチャ，信学技法，EC82-42，1982.

(5)伊藤他：データフロー方式の並列Prologマシン，Proc. of the Logic Programming Conference'83, 9-2, ICOT, 1983.

(6)尾内他：リダクション機構に基づくPrologマシンの一構成法， 情報処理学会， 第26回全国大会，4N-3, Mar. 1983.

(7)Warren,D.H.D.：IMPLEMENTING PROLOG compiling predicate logic programs,Research Report 39&40, Dept. of Artificial Intelligence, Univ. of Edinburgh, 1977.

(8)中村： Associative Evaluation of PROLOG Programs，情報処理学会，記号処理研究会資料，21-4，1982.

(9)安浦他：論理型言語の単一化操作のためのハードウェアアルゴリズム，信学技法，EC84-67，Mar. 1985.

(10)長沼他：連想メモリを用いたPrologマシンの構成法，情報処理学会，第28回全国大会， 5F-10，Mar. 1984.

同上：連想メモリを用いたPrologマシンの詳細構成と性能予測,情報処理学会,第30回全国大会，7C- 2，Mar. 1985.

(11)Dilger,W. and Schneider,H.-A. ： ASSIP-T. A THEOREM PROVING MACHINE， Proc. of the International Conference on FGCS'84, Nov.1984.

(12)小倉他：4Kb CMOS 連想メモリLSI ，信学技法，SSD83-78，1983.

(13)小倉他：大容量連想メモリLSIの構成とその応用手法，信学技法，CAS84-192，Feb. 1985.

(14)Tick,E. and Warren,D.H.D. ： TOWARDS A PIPELINED PROLOG PROCESSOR, Proc. of the International Symposium on Logic Programming, pp.29~40, Feb. 1984 .