記 号 処 理　46－3
（１９８８．７．１）

Ｌｉｓｐ並列処理マシンのための共有メモリ構成とそのＥＶＬＩＳマシンでの実現

坂口寿和，廣西伸幸，齋藤年史，安井　裕
（大阪大学・工学部）

Ｌｉｓｐ並列処理マシン -ＥＶＬＩＳマシン- の動特性について述べ，その結果に基づき，メモリ競合の回避を目的とする，共有メモリシステムの特徴ある設計と，試作について報告する．

マルチプロセッサシステムの性能は，各プロセッサへのプロセスの配分，プロセス生成・消滅のためのオーバヘッド，及びプロセッサ間のデータ授受の容易性等によって大きく変化する．我々の研究室では，ＥＶＬＩＳマシンの並列用Ｌｉｓｐ処理系に，並列処理効率の向上を目的とした種々の並列処理制御機能[6)8)9)]を導入し，オーバヘッドと，負荷分散を制御できることを，動特性により確認している．しかし，メモリ競合の回避を，ソフトウェアによる並列処理制御機能のみで積極的に行うことは困難である．ここでは，これらの並列処理制御機能によるプログラム実行時の動特性の一部を示し，read時の競合を回避する共有メモリシステムの設計と，通常のメモリチップとランダムロジックにより試作したメモリシステム及びそのＥＶＬＩＳマシンへの実装について述べる．

*Design of the shared memory system for multiprocessor Lisp machines and its implementation on the **EVLIS** machine*

Toshikazu SAKAGUCHI, Nobuyuki HIRONISHI, Toshifumi SAITO and Hiroshi YASUI
Department of Applied Physics, Faculty of Engineering, Osaka University
2-1, Yamadaoka, Suita-shi, Osaka 565, JAPAN

*This paper describes design and the implementation of a shared memory system to reduce memory interference in an environment of the multiprocessor system composed of pools of **EVAL-II** processors and aiming for high performance through concurrent evaluation of a Lisp program.*

*The performance of multiprocessor system varies with many factors. We have introduced several useful methods to control granularity of processes, and have confirmed their good effect through dynamic measurements on the multiprocessor Lisp machine - the **EVLIS** machine - . Dynamic measurements also shows that the interference of accesses to shared memory is a difficult problem without the help of hardware. We present the design of the shared memory system which solves the memory interference on read accesses especially, and its implementation with the use of a conventional memory chips and TTLs.*

1，はじめに

我々の研究室では，Lisp処理系の高速化のため，1979年以来リスト処理を複数台のプロセッサで並列に実行するLisp並列処理マシン - EVLISマシン[3)4)5)] - の試作・研究を続けている．すでに，3台のEVAL-IIプロセッサ[3)]を備えた EVLISマシンが稼働しているが，そのマシン上に，Parallel-Lispのインタプリタ[6)]，コンパイラ[8)]（以下，並列インタプリタ，並列コンパイラ）が完成しており，その高速性が示されている．

本稿では， EVLISマシンの並列インタプリタと並列コンパイラの実行特性測定から，マルチプロセッサシステムにおける実行効率低下の要因を分析し，並列処理効率の向上について述べる．またメモリ競合の回避，調停等に着目し，ソフトウェアでの解決が困難な，共有メインメモリのハードウェア上のアクセス競合を減少させるための新しいメインメモリシステムを提案し，メモリシステムの側からの並列処理の高速化について述べる．

2，EVLISマシンとその並列処理方式

本節では我々の研究室で提案し実現したLispの並列処理方式について述べ，EVLIS マシンのハードウェアの構成について概説する．

2-1，EVLISマシンにおけるLispの並列処理方式

Lispは関数型の言語であるために，そのプログラムは次々と関数を適用させその値を求めるといった，逐次的な処理を行なっており，並列化が困難に見える．しかし我々は，関数の引数が複数個のとき，引数評価の処理をその引数ひとつひとつが並列に実行できることに着目し，Lispの並列処理マシン -EVLISマシン- をすでに提案した．

Lispにおいては，この引数評価はインタプリタ内で使用される関数 evlis[1)]で行われる．またこの関数はインタプリタの基本的な部分で使用されていることにより，リスト処理中に頻繁に呼び出される．それゆえ evlisを並列に処理すれば，並列処理を意識せずに記述されたプログラムにおいても，並列化の機会が多く存在することになる．[2)]

そこでEVLISマシンにおいてはリスト処理の並列化の対象を関数 evlisに設定している．EVLISマシンの名前の由来はここにある．上記の考え方を基に，EVLISマシンは製作され，1981年から稼働中である．

EVLIS マシンで並列に処理される，一つの処理の単位をプロセスと呼ぶ．第４節で述べる並列処理制御機能を用いないときは，プロセスは引数一つの評価を行う処理単位である．

2-2．EVLISマシンの構成

EVLISマシンは，高速リスト処理専用プロセッサ"EVAL-IIプロセッサ[4)]" の複数台でLispを並列処理するマシンである．以下に従来から研究を行なってきた，EVLIS マシンのハードウェアシステム構成を示す．（図１）

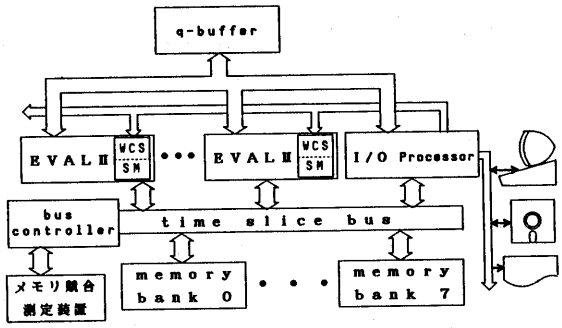

図１　ＥＶＬＩＳマシンの構成

EVAL-IIプロセッサ及び，入出力とシステムのコントロールを行う "I/O プロセッサ" が，時分割バスを介して各プロセッサ共有の "メインメモリ" に接続している．また並列処理の管理の高速化を図るための FIFOメモリ "q-buffer" も全プロセッサからアクセス可能である．

これらの構成要素は各々独立したクロックで非同期に動作している．このためハードウェアでのシステムの拡張や変更が容易となっている．

①EVAL-IIプロセッサ[4)]

Lispの特徴であるリスト処理を，高速に行うように設計し，ランダムロジックによって研究室で製作された，マイクロプログラム制御方式の専用プロセッサである．以下にその特徴を述べる．

- 基本命令サイクルは100nsecである．
- 書換可能な制御記憶(WCS)を8K 語(1語 48bits)実装している．WCSはI/Oプロセッサを通して書換可能である．
- 高速アクセス可能なローカルメモリで，ハードウェアでサポートされたスタックとしても利用できるスクラッチパッドメモリ(SM)を 4K語実装している．
- WCS や SM のアクセスはパイプライン処理を行い命令やデータをフェッチする時間が表面に現れない．
- オペランドとしてリストセルなどへのポインタが頻繁に用いられるために，バス上では，データとアドレスの区別をしていない．
- 分岐命令は他の演算と並行して分岐が可能である．
- データの特定のフィールドを抽出し，その値をもとに一命令で他方向への分岐が可能なディスパッチ機能がある
- メインメモリ上のリストを手繰ることを，ALU による他の演算と同時に並列処理可能とする CAR･CDR演算命令を持っている．
- ALU演算は3アドレス方式として，無駄なレジスタ転送をなくしている．
- EVAL-IIプロセッサの制御，デバッグ及び 動特性測定のためのデータ収集などに用いる，診断用インターフェースをプロセッサ内部にハードウェアで実装

している．このインタフェースを介して，プロセッサ内部バスの状態などの情報を I/Oプロセッサに取り入れることができる．また WCS上の 1bit（デバッグビット）を用い，任意の番地にある命令の実行頻度の情報も得ることができる．

②メインメモリと時分割バス

メインメモリは全てのプロセッサからアクセス可能な共有メモリであり，種々の共有データを格納するメモリである．1語40bits(CAR部，CDR部各20bits)で32K語実装されている．このメインメモリの特色を以下に示す．

・アトム，リスト，フレームなど共有データ，及びプロセッサ間通信のための，メイルボックスを格納している．フレームについては，後述するが，並列処理を管理するための情報を一つにまとめたものである．
・CAR部あるいは CDR部のみの書換が可能である．
・バスコントローラによって管理された時分割バスにより全てのプロセッサから対等にアクセスできる．
・時分割バス
　複数台のプロセッサが，同時にこのバスを利用しようとし競合したとき（"バス競合"が生じた時)のみ，優先順位に従って，時分割に利用される．バス競合を生じてないときは時分割でなく，プロセッサはこのバスをすぐ利用できる．
・アドレス空間のバンク分け
　メインメモリはそのアドレス空間を物理的に複数個のメモリバンク(現在８バンク実装)に分割している．そしてこのバンクを単位に動作を行っている．
　従って，一つのバンクがプロセッサからアクセスされている時でも，他のバンクは，別のプロセッサからのアクセスが可能である．つまりアクセス動作を各バンクで並行して行うことができる．
　この並列アクセスのため，単体のプロセッサの動作でもアクセスが高速化される．さらに複数のバンクにアクセスが分散することにより，一つのバンクへの相対的なアクセス頻度が減少するので，複数台のプロセッサで同時にアクセスすることにより生じる競合("バンク競合")を軽減することができる．
・バンクロック機能
　ソフトウェア上での論理的な排他制御を実現するため，プロセッサからメモリに出したLock信号によって，他のプロセッサからのアクセスを，バンク単位で受けつけなくすることが可能である．
・アクセス時間
　メインメモリのクロックサイクル時間をϕ_m時間とすると，競合の調停回路とバスからメモリバンクへのアクセス情報の転送処理に2ϕ_m時間，メモリバンク内のアクセス動作に5ϕ_m時間を要する．そのため，競合が生じていない場合アクセスの受付からのメモリアクセス時間は，Readアクセスに7ϕ_m時間必要とする．Writeアクセスは，バンクまでアクセスに必要な情報の転送がすめば，プロセッサ側の仕事は終了するため，2ϕ_m時間である．
　ただしメインメモリとプロセッサは非同期に動作しているため，プロセッサからみたアクセス時間には，平均して1/2ϕ_m時間のアクセス受付までの待ち時間が加わる．

以上は新しいメインメモリシステムを提案するまでのシステムであり，第５節では，このシステムのメモリ競合の解決への方針と，実現について述べる．

③ｑ－ｂｕｆｆｅｒ

各プロセッサからアクセス可能なFIFOメモリである．未処理のプロセスへのフレームへのポインタを登録し，各プロセッサのプロセスの獲得を高速化する．１語 21 bitsで 4k語実装している．サイクルタイムは 50 nsec である．

④Ｉ／Ｏプロセッサ

このプロセッサはサービスプロセッサとしてシステムのコントロールや，サービスを行うために，次のような機能を備えている．

・CRTディスプレイ，キーボード，フロッピーディスクドライブ等の入出力デバイスをコントロールする．
・EVAL-Ⅱプロセッサの WCSを読み書きする．
・ハードウェア割り込みとメイルボックスを用いた通信をEVAL-Ⅱプロセッサとの間でおこなう．
・LispのＳ式を読み，メインメモリ上のリストデータとして展開する．また，出力も行う．
・GC時における，各EVAL-Ⅱプロセッサの制御を行う．
・診断用インタフェースを使用して，EVAL-Ⅱプロセッサのコントロールを行う．またデバッグや動特性測定のための情報も診断用インタフェースを用いて収集する．
・メモリ競合測定装置を使用した情報収集を行う．

⑤メモリ競合測定装置[7)]

共有メモリであるメインメモリは複数台のプロセッサからのアクセスにより競合が生じることがある．本装置は，その競合を定量的に測定するものであり，メモリ競合やメモリ構成の評価に必要な情報を収集することができる．

３．並列処理効率の分析

高速化を目的として並列処理を行うとき，n台のプロセッサを使用したからといって，単純に n倍の処理速度が得られるわけではなく，プロセッサ台数に対する処理速度の向上の比率 "並列処理効率" を考えねばならない．

特に並列化の対象をLispによるリスト処理とする時，いくつもの要因によって並列処理効率は低下する．

本節では，実際の並列処理用Lisp処理系における並列処理の実現方法と，その並列用Lispの並列実行時の処理時間の分類を行い，並列処理効率を低下させる要因を分析する．

3-1. 並列処理時の処理時間の分類

並列処理の向上のためまず，実際の処理時の処理時間の内訳を行い，これを分析する．以下，処理時間の内訳としては，並列処理に加わる全てのプロセッサの延べの処理時間の内訳とする．

(a)Evaluation時間: リスト処理を行なっている時間 (teval)
(b)オーバヘッド処理時間: 主にプロセスの生成･消滅･管理に関する処理に必要な時間 (toh)
(c)プロセス待ち時間: 処理すべきプロセスがないときの待ち時間 (tidle)
(d)メモリ競合待ち時間: メモリアクセスの競合待ちの時間 (tmc)
(e)その他の処理時間 (tsync)

(a) Evaluation時間

本来のリストデータの処理のため必要とされる処理時間．並列処理による実行時間Tが，T=teval /n となれば，理想状態である．

tevalとしては，1台のプロセッサが逐次処理用Lisp処理系で，プログラムを処理したときに必要な時間を，本来のリスト処理のために必要な時間であるとしてこれを用いる．

(b) オーバヘッド処理の時間

並列処理を行なう時は，逐次処理と比べて，余分な処理(以下オーバヘッドと呼ぶ処理)が必要となる．

これは2-1.で述べたプロセスの生成と消滅のためおよびフレーム管理のための処理である．

また未使用のリストセルを管理するフリーリストの取扱いを，複数のプロセッサで利用できる形式で行わねばならないため生じる処理の増加もオーバヘッドに含まれる．

1台のプロセッサのみを用いて，他のプロセッサはメモリ競合等を生じないように停止させた状態で，並列用Lisp処理系を並列実行のための処理をも行いながら使用した時の処理時間をt_1とすると，toh=t_1-tevalにより得られる．

またWCS上の Lisp処理系のマイクロコードのプログラムにおいて，オーバヘッドに当たる処理を行っている各ルーチンに，デバッグビットを立てそのルーチンの命令実行ステップ数を測定することにより，たとえば，フレーム生成のための処理のみといったオーバヘッドの一部分の処理の時間を得ることも可能である．

(c) プロセス待ち時間

プロセッサが一つのプロセスの処理を終了した後は，q-bufferに登録されている未処理の新しい子フレームへのポインタを得て，また次の処理を続けてゆくことができる．

こうして全てのプロセッサに次々ととぎれることなく，処理すべきプロセスを供給できる時は，いずれのプロセッサも，その処理能力を有効に利用することができる．

しかし並列に処理できるプロセスが生成されていない時は，ただプロセスを待つだけとなってしまい，並列処理効率を低下させる．

n台のプロセッサで並列処理を行うときは，n台の全てがプロセスの処理を行っており，プロセス待ちのプロセッサが0台の場合から，並列に処理できるプロセスがなく1台のプロセッサのみで処理が行われている，つまり (n-1)台のプロセッサが，プロセス待ち時間にある場合まで，プロセス待ち時間の増減は，全体の並列処理効率に大きく影響を与える．

tidle の測定は，WCS上の Lisp処理系の中で，プロセス待ちのための，アイドルループのルーチンの部分にデバッグビットを立てて行うことができる．実際に複数台のプロセッサを用いて並列処理を行い，各プロセッサがアイドルループを何回繰り返したかを測定することにより，それぞれのプロセッサがプロセス待ちであった時間を推定することができる．

(d) メモリ競合待ち時間

メインメモリへのアクセスが他のプロセッサからのアクセスと重なったとき，どちらがアクセスを待つ必要が生じる．これがメモリ競合待ち時間である．

このメモリ競合はさらに次のふたつの種類に分けられる．

① 論理的な排他処理のための待ち時間

フリーリストへのベースポインタや，未使用なフレームを管理しているフリーフレームへのベースポインタなど，常に論理的にも排他的に一つのプロセスからしかアクセスを許されないソフトウェア資源がある．

これらをアクセスする時には意図的な排他処理を行うため，ハードウェア的にはアクセス可能であってもそのソフトウェア資源へのアクセスはその間，他のプロセッサからは禁止される．このような待ち時間を，論理的な排他処理による待ち時間と呼ぶ．

複数の自分の子プロセスからアクセスされる，親フレームへのアクセスにも，この論理的な排他処理が必要となり，ここでも論理的な排他処理による待ち時間が生じる．

② 物理的な競合の待ち時間

時分割バスや，メモリバンクなどハードウェアが一度に一つのプロセッサからのアクセスにしか使用できないために生じる競合の待ち時間．

たとえば異なるプロセスがそれぞれメモリ上の異なるアドレスへ同時にアクセスを行おうとする時など，プログラム上では論理的になんら排他処理を必要としない状況においても，ハードウェアの制約のため一つのプロセッサのみのアクセスしか可能でないなら，アクセスが終了してハードウェアが利用可能となるまで他のプロセッサは，アクセスを待たねばならない． これを物理的な競合の待ち時間と呼ぶ．

メモリ競合の待ち時間は，メモリ競合測定装置を用い

ることにより，複数台のプロセッサによる実際の並列処理の実行中に測定することが可能である．

競合測定装置では，物理的な競合であるところの，時分割バスにおける競合(バス競合)と，物理的なものと論理的な競合の両方を含んだ，メモリバンクでの競合(バンク競合)の測定ができる．

また並列処理に要した全処理時間から，他の方法で測定できる，teval，tidle，toh　等を差し引くことにより，実際にプロセッサがメモリ競合のため待たされた時間を得ることも可能である．

(e)　その他の処理時間

その他に並列処理のため必要となった処理時間では次の二つが存在する．

① 割り込みを用いたプロセッサ間の通信時間

② q-bufferのアクセス競合待ち時間時間

いずれの処理時間もプロセッサの台数の増加に関連して，増加する傾向を持つが，しかし現在これらの処理時間は合計しても全処理時間に占める割合は約1%と極く小さい．

以上を基に，次節では並列処理効率を低下させる要因として，オーバヘッド処理の時間，プロセス待ち時間，メモリ競合による待ち時間の調停･回避について論じる．

4．並列処理の制御と効率向上

本節ではソフトウェアの立場から，並列処理効率の向上を目的に，並列用Lisp処理系に導入されている並列処理制御機能について述べる．

EVLISマシン 並列インタプリタ及び並列コンパイラでは，オーバヘッド処理時間とプロセス待ち時間の減少を目的とする並列処理制御を行う．並列評価関数呼び出しの指示[6)]や，評価コントロールリスト[8)]の記述，レベルによる制御[6)9)]等の並列処理制御機能をすでに提案，実現している．これらを用いて，プロセスの処理の大きさ(粒度)等を制御することにより，オーバヘッドとプロセス待ち時間を減少させ，並列処理効率の向上に成功している．

4-1. 引数並列評価を選択できる機能

EVLIS マシンの名前にあるとおり，プログラムの全ての関数の実行において，それらの引数の全てを並列に評価する場合を考える．この場合プロセスの数は階層的に非常に増大し，その処理内容は低いレベルでは，例えばCARやCDRのような非常に小さなものとなる．

一つのプロセスの粒度が小さくなるとそのプロセスのEvaluation時間に比べ，オーバヘッドの時間が大きくなり，並列処理効率が低下する．

この問題の解決のため，EVLISマシンでは，プログラム上の関数呼び出しにおいて，引数の逐次処理か並列処理の選択をできる機能を導入している．

これは関数呼び出し時の，関数名の前に "*" をエディタ等で付加することにより，引数の並列評価を指示する．この制御を用いることにより，並列化したときのプロセスの粒度が大きな，引数評価を持つ関数呼び出しのみにおいて並列処理を行うように指示できる．

以下にその制御の使用例を示す(図２)．

```
        :
(* fn1 x (fn2 y)(fn3 z))
        :
```

図２　引数の並列評価を指示した例

4-2. レベルによる制御[6)9)]

"*" を用いて引数の並列評価の指示を行うだけでは，並列処理効率が向上できない場合がある．

これは並列処理を指示した関数呼び出しが，再帰的に呼び出されたり，入れ子になっている場合である．この時もプロセスの粒度が小さくなり，オーバヘッドの処理が増加する．

この問題の解決のため，EVLISマシンでは，設定したレベル以上では，プロセスを生成せず，逐次的に評価を行うことが出来る制御をコマンド及び関数形式で指示することができる．これを "レベルによる制御" と呼び，設定する制限値を "レベル制限値" と呼ぶ．図３にこの使用例を示す．

例 1)　/SETLIMIT 4　･･･コマンドによる指示

例 2)　(SETLIMIT 4) ･･･関数による指示

図３　レベル制限値を４とする例

4-3. 評価コントロールリストによる制御[8)]

引数を並列に評価するとき，各引数評価のプロセスの粒度がアンバランスな場合がある．

例えば図２の例で３個の引数の中で１番目の引数はアトムの評価を行うだけであり，他の２個より小さい．この小さな処理のためにもプロセスを生成するとオーバヘッドが増加する．

またLispでの副作用関数を実行するとき，その結果を正しく伝播させるため並列処理においても，副作用伝播の順位(優先順位と呼ぶ)を保つ必要がある．

これらの問題の効率的な制御のために，並列コンパイラでは，引数の並列評価において，処理の優先順位やプロセスの生成をエディタ等で簡単に指示できる機能を実現している．これは引数の並列評価を指示する "*" の次に，<評価コントロールリスト>を記述することによりこの関数の引数の評価の詳細について制御する．図４にこの指示の例を示す．

```
(* ((1 2) 3) fn1 x (fn2 y) (fn3 z))
```

下線部が＜評価コントロールリスト＞

図４　＜評価コントロールリスト＞の使用例

4-4. 並列コンパイラと並列インタプリタの動特性

4-1.～4-3.で述べた並列処理制御の指示や組み合わせを変化させたときの，並列処理用コンパイルオブジェクトおよび並列インタプリタ実行時の動特性を測定した．

はじめに，並列コンパイラを用い，制御機能の組み合わせと制限値を変化させたときの処理時間の変化を図５に示す．例題としては "List-Tarai" を用い，EVAL-Ⅱプロセッサは3台動作，クロックは200nsec，φm時間は100 nsec で実行している． 実行時間は，逐次処理実行時を100%として正規化している．

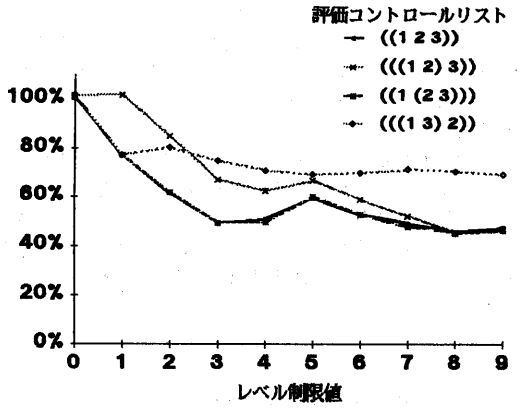

図５　並列処理制御機能の組合せによる
実行時間の変化
List-Tarai-4 コンパイルオブジェクト
（実行時間は逐次実行時間を100%とする．）

List-TaraiはLispコンテスト[10]の課題である．図６にこのプログラムを示す．

```
(de List-Tarai (lambda (X Y Z)
  (cond ((lessp (car X)(car Y))
    (* <評価コントロールリスト> List-Tarai
          (List-Tarai (copy (cdr X)) Y Z)
          (List-Tarai (copy (cdr Y)) Z X)
          (List-Tarai (copy (cdr Z)) X Y) ))
    (T Y) ) ))
```

図６　関数 List-Tarai に並列指示および
評価コントロールリストを用いた制御を行なう場合

また，並列インタプリタ，並列コンパイラにおいて，<評価コントロールリスト>により関数の引数の評価を制御した場合について，実行特性の例を図７に示す．

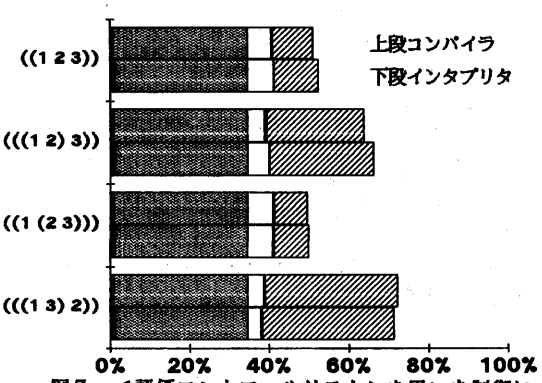

図７　<評価コントロールリスト>を用いた制御に
対する実行特性
List-Tarai-4:レベル制限値 4
（実行時間は逐次実行時間を100%とする．）

次に，並列インタプリタ，並列コンパイラにおいて，レベルの制限値を変化させたときの，実行特性をそれぞれ図８，図９に示す．

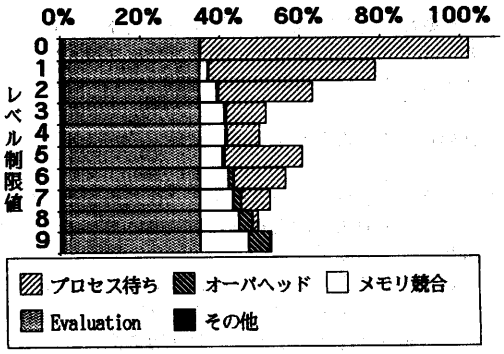

図８　レベル制御に対する並列インタプリタの
実行特性
List-Tarai-4:評価コントロールリスト((1 (2 3)))
（実行時間は逐次実行時間を100%とする．）

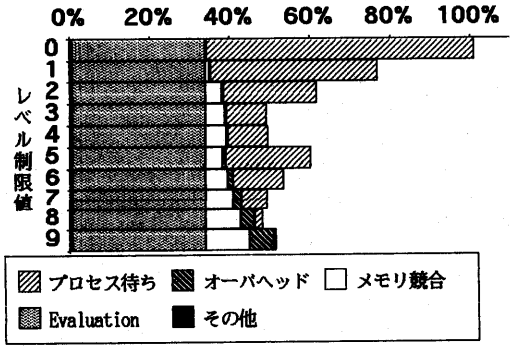

図９　レベル制御に対する並列コンパイルオブジェクトの実行特性
List-Tarai-4:評価コントロールリスト((1 (2 3)))
（実行時間は逐次実行時間を100%とする．）

そしてこの上さらに並列処理の効率向上のためには，
・前述の種々の並列処理制御機能では，メモリ競合を積極的に減少させることは困難である．
・コンパイラを用いれば，Lisp のプログラムが，メインメモリ上のＳ式から各プロセッサ上のコントロールメモリ上のマイクロコードとなるためにメインメモリのアクセス総回数が減少し，メモリ競合の総回数も減少する．しかし全体の処理時間も短くなるために，全処理時間に占めるメモリ競合の比率は，あまり減少しない．
・プロセッサ台数の増加にともない，メモリ競合も増加する．

以上のような特性が有り，ソフトウェアのみでは本質的な解決は困難と考えられる．

したがって現在以上の処理効率の向上を望むときには，新しいメモリシステムとしてのハードウェアからのサポートが必要となる．そのため本研究では，ハードウェアによるメモリ競合の解消に着目した．

5．新しいメインメモリシステム[11]の提案

本節ではハードウェアの立場から，並列処理効率をより高めることを目的とした，新しいメモリシステムの設計方針について示す．

5-1．メインメモリアクセスの改良

実際に各プロセッサが，メインメモリをアクセスするときに必要とする時間は，次のようなものがある．

① アクセス受付け待ち時間：EVAL-Ⅱプロセッサとメインメモリが，各々非同期に独立したクロックで動作しているため，アクセスを受け付けるまで必要な時間

② 競合の調停回路等に要する時間

③ 論理的な排他処理による待ち時間：副作用の伝播や，フリーセル獲得のため等の論理的な排他処理を他のプロセッサが行なったための待ち時間

④ 物理的な競合による待ち時間：バスあるいは，メモリバンク等のハードウェア資源の競合のために生じる待ち時間．

⑤ 使用するメモリＩＣのアクセス時間

以上の時間のうちで，①は，EVLIS マシンを構成している各要素の改良や拡張のフレキシビリティを保つための非同期なアクセスに伴うもので残さざるをえない．また③は，Lispの処理系や，応用プログラムのアルゴリズムに関係し，これを小さくすることは困難である．これらのため新しいメモリシステムの高速化は，②，④を中心に考える．

5-2．共有メモリ方式の継承

新しいメモリシステムにおいても，共有のメインメモリにリストデータを置いて，これを並列処理する方式を継承する．

またEVLISマシンの場合，全てのEVAL-Ⅱプロセッサがハードウェア的にも，またLisp処理系の上での並列処理においても対等な動作を行なうので，各プロセッサからの共有メモリへのアクセスも対等に取り扱う．

5-3．Readアクセスの高速化

Lisp言語では，インタプリタが多用されるが，この時は，メインメモリ上のプログラムへ繰り返しReadアクセスが行なわれる．

また，Lisp処理系では，一つの要素をアクセスするためにリストをたぐるなど，ポインタを利用した間接的なアクセスが多用される．このためにもReadアクセスが繰り返される．

これらの結果，メモリアクセスに対するReadアクセスはWriteアクセスより非常に(応用プログラムにもよるが10倍程度[7])多い．

これを受けメモリシステムの設計にはReadアクセスの高速化に重点を置く．また繰り返してのReadアクセスに対しても，アクセス速度の低下を生じないように設計する．

5-4．新しいメインメモリシステムの構成

並列処理効率をより高めるために提案する新しい共有メインメモリシステムについて述べる．5-1.～5-3.での設計方針に基づき，さらに一般に入手できるメモリチップ等を用いることによって構成できる新しいメモリシステムについて図１０に示す．

各プロセッサは，メモリシステム上の各々のインタフェース部とRead-Write切り替えスイッチを介して，メモリをアクセスする．実際のメモリ部分は，プロセッサ台数をＮとしたとき，メモリ空間の１記憶単位に対して，システムのメモリ空間の１バンクは２×Ｎ個のバンクで２×Ｎ個の記憶単位を必要とする．従って新しいメモリ構成される．そしてこれらは Readアクセス用とWriteアクセス用との二つのバンク群に区別されるが，二つのバンク群は，φm時間で高速に切り換えられ相互にそのアクセス用途を交換する．

Write 時には，各バンクの内容の一致を保つため２×Ｎ個の全バンクに，同じ内容を書き込まなければならない．このためまずWriteアクセス用のバンクすべて(Ｎ個)に,同時に同じ内容を書き込む．次のサイクルで，Readアクセス用と Writeアクセス用のバンクの役割が交代し，書き込みをまだ行なっていない もとのReadアクセス用のバンク(Ｎ個)に同じ内容を書き込む．

Write アクセスは，同時に複数のプロセッサからのアクセスは許さず，ただ１つのプロセッサのみがアクセスできる．そのため他のプロセッサとのアクセス調停回路が必要となる．従って Writeアクセスには，最低 ３φm（調停のためと２回の書き込み）が必要となる．さらに，他のプロセッサからの Writeと競合したときは,Writeアクセス完了までの時間は延長される．しかし，プロセッサ側からみると各インタフェース部内に存在する書き込みバッファにデータを渡すことができれば，プロセッサの仕事は終わるので，インタフェース部のバッファが空いているときは，プロセッサからみたアクセス時間は，Read と同じ時間で可能である．

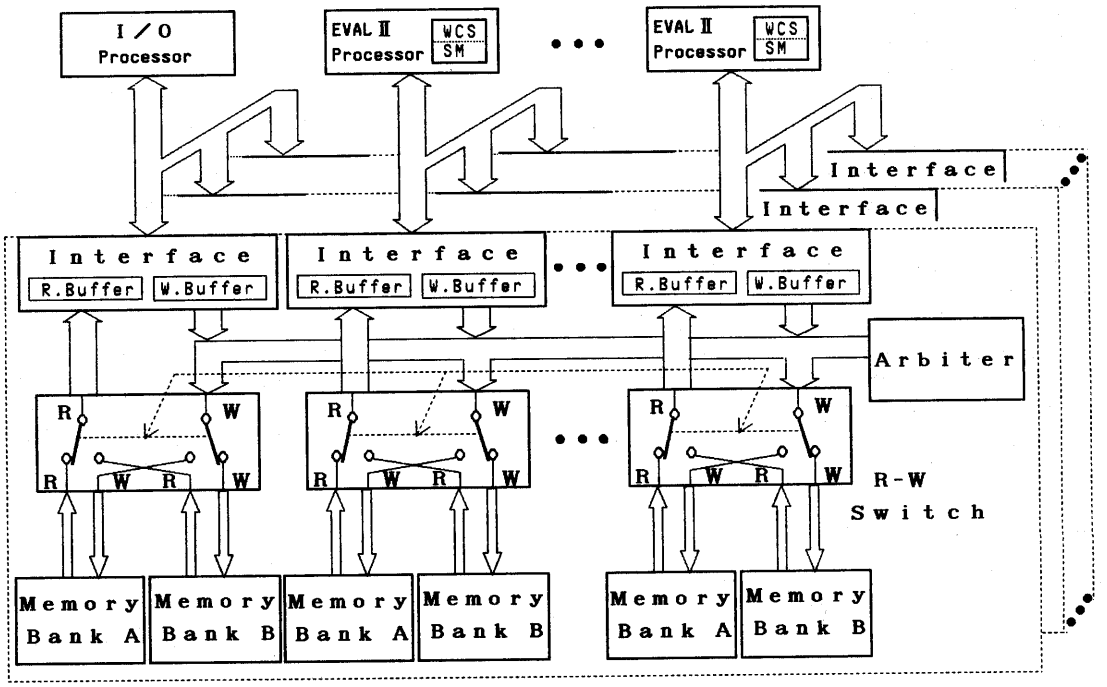

図１０　新しいメモリシステムの構成

Read アクセスは，Ｎ個の Readアクセス用のバンクを，Ｎ台のプロセッサに対応させ，各プロセッサから物理的な競合なしに１φm時間内でReadアクセスできる．

プロセッサが，書き込みで物理的な競合により待つのは，そのプロセッサ自身の行った，前回の Writeサイクルが完了しておらず，バッファがまだ使用されているときであり，短い時間にいくつものWriteアクセスが集中したときのみである．

このため，メモリアクセス全体に対して，少ない割合を占める Writeアクセスがさらに集中したためにおこる競合の発生はごく少ないものと考えられる．

６．おわりに

本研究において現在製作実装中の新しいメインメモリシステムでは，メモリ単体としての動作を確認しており，まず現システムのアドレス空間の一部分を新しい方式で置き換え，実際の効果を測定すべく準備中である．

さらに並列処理効率を高くするための，今後のシステムのアーキテクチャ上の研究として，プロセッサ間，プロセス間の通信と管理を行うハードウェアによるサポートの拡充が求められる．

また，ユーザにより並列制御指示がなくても，自動的に並列制御の指示を行なうプリコンパイラの開発がある．

［参考文献］

(1) McCarthy, J., et al.: LISP 1.5 Programmer's Manual, The M.I.T. Press (1962).
(2) 安井，齋藤，三石，宮崎:evlisの並列処理，情処第20回全大，3K-8(1979).
(3) 安井 裕 他:LISPでの並列処理における動特性と，EVLISﾏｼﾝの構成,情処,記号処理資料10-4(1979).
(4) 前川 博俊 他:高速LISPﾏｼﾝとﾘｽﾄ処理ﾌﾟﾛｾｯｻEVAL Ⅱ，情処論文誌,Vol.24,No.5,pp683-688(1983).
(5) Pleszukun, A.R., Thazhuthaveetil, M.J."The Architecture of Lisp Machines",IEEE COMPUTER Vol.20,No.3,pp35-44(1987).
(6) 西川 岳 他:EVLISﾏｼﾝの並列処理ｱﾙｺﾞﾘｽﾞﾑとそのｲﾝﾌﾟﾘﾒﾝﾃｰｼｮﾝ,情処第23回全大,4H-7(1981).
(7) 西開地 秀和 他:LISP並列処理ﾏｼﾝ -EVLISﾏｼﾝ- の動特性測定と評価,情処,記号処理資料31-9(1985).
(8) 安田 弘幸 他:EVLISﾏｼﾝにおける並列LISPｺﾝﾊﾟｲﾗの実現,情処第33回全大,2E-5(1986).
(9)三野 雅仁 他:EVLISﾏｼﾝの並列処理Lispﾌﾟﾛｸﾞﾗﾑ作成支援環境と並列実行特性,情処第35回全大,6Q-8(1987).
(10)奥野 博:第3回LISPｺﾝﾃｽﾄ及び第1回Prologｺﾝﾃｽﾄ報告,情処,記号処理資料33-4(1985).
(11)坂口 寿和 他:Lisp並列処理ﾏｼﾝ-EVLISﾏｼﾝ-のﾒﾓﾘ構成情処第36回全大,4H-7(1988).