記号処理 31－9
（1985. 3. 12）

LISP並列処理マシン—EVLISマシン—の動特性測定と評価

西関地秀和，高橋俊樹，安田弘幸，斎藤年史，安井裕
（大阪大学工学部）

1.はじめに

LISPの高速処理を目的としたLISP並列処理マシン—EVLISマシン—を製作し、研究を進めている。複数台のリスト処理専用プロセッサEVALⅡと、各プロセッサに共有されるメインメモリ（MM）を中心にした、マルチプロセッサシステムである。既にLISP並列処理インタプリタが製作され、2台のEVALⅡ上で稼動しており、ベンチマークプログラムにおいて並列処理により速度が向上した例を示すことができた。[1]

共有メモリ型マルチプロセッサでは、メモリでの競合によりプロセッサの待ち時間が増加するため、並列処理効率を低下させるおそれがある。とくにリスト処理ではメモリ操作が主体となるため、その影響を調べる必要がある。今回、この点に注目し、実現したマシン上でのメモリ競合による待ち時間を直接測定し、その動特性を明らかにした。それをもとに、EVLISの共有メモリ—MM—に関するハードウェア構成の評価を行い、リスト並列処理での問題点について検討を行った。

まず、2章でLISP並列処理方式におけるメモリ競合の問題にふれ、3章でEVLISマシンのハードウェアについて説明する。4章でメモリ競合の測定方法について述べ、5章で測定結果を示し、6章で考察する。

2. LISP並列処理とEVLISマシン

2－1. EVLISマシンにおけるLISP並列処理

EVLISマシンでは、LISP言語での関数レベルの並列処理を行う。ユーザは並列処理に伴う実行制御については、考慮する必要はない。インタプリタ内の関数evlisの第1引数の各要素の評価を、複数台のEVALⅡで自動的に並列に実行する（ここで分割された各処理をプロセスと呼ぶ）。

プロセス実行の各プロセッサに対する割り付け等は、各プロセッサを対等の立場として動的に行い、資源の有効利用をはかる。MMにはリストデータを格納し、各プロセスごとの実行環境についてはスタックフレーム（単にフレームと呼ぶ）を生成し、格納・利用する。また、プロセス間の優先関係を1本のリストで表現している（このリストをg-listと呼ぶ）。これらの処理を、LISP並列処理インタプリタは含んでいる。

ソフトウェアシミュレーションにより、この方式での並列処理効果が確められており、並列化をとくに意識していないプログラムに対しては処理時間の短縮に有効なプロセッサ台数が4台程度であることを示した。[2]しかし、このときの仮定には、メモリ競合等の問題を含んでいなかった。現在は、更に処理効率の良いシステムを実現する上での問題を検討するため、製作したマシンの上で、研究を進めている。

2－2．LISP並列処理実現上の問題点

並列処理マシンの実現において、並列処理効率を減ずる要因として、次の2点があげられる。ひとつは並列処理に伴うオーバーヘッドの増加で、もうひとつは、共有メモリでの競合による待ち時間の増加である。

前者については、並列処理方式の細部の検討によるオーバーヘッド量の削減と、処理高速化のためのハードウェア化が解決方法として考えられる。そして、既に、EVLISマシンでは、選択的な並列化と、プロセス数を総数あるいは並列化の深さで制限するインタプリタを実現し、処理速度向上の成果を上げている。(1)

後者のメモリ競合の問題は、共有メモリ型マルチプロセッサでは避けられない。とくにリスト並列処理を行う場合、性能低下の要因となると考えられる。今回この問題について、EVLISマシン上で、実測による解析を試みた。

2－3．メモリ競合

リスト処理では、MMへのアクセスが主体と考えられるため、メモリ競合の影響を考慮したハードウェア構成の設計が重要である。

メモリ競合の解析には、一般に待ち行列解法や、ソフトウェアシミュレーションが用いられる。しかしモデルによっては実際のマシンでのふるまいと隔たるおそれがある。とくに、我々の並列処理方式では、リスト，フレームという異なるデータ構造をMM上で扱い、また並列化のスケジューリングを動的に行うので、メモリ競合に影響を与えると考えられる要因が多く、それぞれのふるまいも複雑となり、モデルによる解析が困難である。したがって現実のマシンでのデータが要望される。

現実のマシンでのデータであってもタスクレベルでの実行時間等の性能を測定したのでは、マクロ的であり、実際の並列処理のオーバーヘッドによる処理効率の低下を含んでしまい、また複雑なメモリアクセスの振舞いを知ることができない。そのため今回メモリ競合の影響をアクセス単位でハードウェア的に直接測定を行った。測定方法については、4章で述べる。

3．EVLISマシンの構成

3－1．ハードウェア構成

主な構成要素について説明する。

- プロセッサEVALⅡ --- マイクロプログラム制御のリスト処理専用プロセッサでLISP並列処理インタプリタが稼動する。パイプライン処理や、リスト処理向きのアーキテクチャを実現している。内部にマイクロプログラムを格納するWCS（Writable Control Storage），スタック等に用いるScratchpad Memoryをもち、また診断用インタフェースを組み込んでいる。現在2台実装している。詳細については文献(3)を参照されたい。
- MM --- リストデータ，フレーム等を格納する共有メモリで、1語40bit，4k語のバンク8台と，共有バスおよびコントローラからなる。MMの詳細は次節で述べる。
- g－buffer --- 未処理プロセスを、フレームへのポインタを用いて登録し、各EVALⅡのプロセス獲得を高速化する。共有メモリの一つで、1語21bitの高速FIFOバッファである。

・I/O プロセッサ ---LISP の入出力の実行や、並列処理におけるガーベジコレクションのマスタプロセッサとして働く。各ハードウェアの管理機能をもち，また，ファイルの管理を行う。

3－2．MM

MMは、EVALⅡとは独立したクロックを持ち、共有バスはこのクロックで時分割することにより、多重化している。

バンク間のバス競合は、バンク間の優先順位（1クロックサイクルごとに変化する）がより高いバンクへのアクセスを許可する方法をとり、一つのバンクに対するプロセッサ間の競合は、バンクごとに固定された優先順位により制御する。これらの競合解決は、1クロック内で行われる。

排他制御機構として、lock操作が可能である。プロセッサからの read-with-lock 命令で、バンク単位の lock を行い、次にそのプロセッサからそのバンクを lock 以外の命令でアクセスすることにより、lock を解除する。

3－3．LISP並列処理

LISPインタプリタの言語仕様は、LISP1.5に準拠しており、並列処理機能を加えている。変数束縛には、並列処理における多重環境を実現するために、a-list を用いている。

記憶管理について、フレーム用領域と、リストセル領域に分けて述べる。

フレーム領域は、16語ごとに基本フレームとして分割されている。未使用時には、フリーフレームリストとして1本にリンクし、使用中のフレームは、g-list によりリンクしている。フレームの回収は、プロセス消滅時に逐次的に行う。

リストセル領域の未使用セルは、1本のフリーリストで管理する。ガーベジコレクションは、一括型で、その際コレクティングを、リストセル領域のバンク間で、インタリーブした順に行い、cons によるセル生成を各バンクに分散させる。

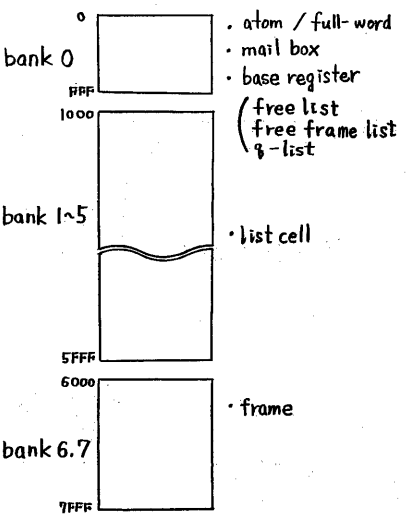

図 1．メインメモリアロケーション

フリーリスト，フリーフレームの root は、MM内に base register を置き、共有し、獲得には排他制御を行う。また、プロセス獲得，消滅時のフレーム，g-list の操作にも排他制御を必要とする。このため、デッドロックを避けて、メモリアロケーションを行う（図1）。

4．メモリ競合の測定方法

4－1．メモリ競合測定装置

メモリ競合の表現方法は、いろいろ考えられるが、我々はメモリアクセスにおけるプロセッサの待ち時間を計測し、その増加分でメモリ競合を表わす。

そして、測定に伴うオーバーヘッドをなくすために、ハードウェアで測定装置を実装している（図２）。

装置の基本設計は、1981年に並列処理が稼動したと同時に行われた[4]。今回、競合の振舞いを明らかにするための改良を加え実現した。基本設計では、1つのプロセッサにおける競合量を一括して測定していたのに対し、今回の改良では ①バンクへのアクセスのかたよりを知るため、各バンクごとに分離した測定を行う。②バス競合とバンク競合を分離するために、バンク競合の影響のみを測定できるモードを加えた。 実際には、同時測定せずにプロセッサ、バンクおよび測定モードの各組合せについて、それぞれ測定を行う。

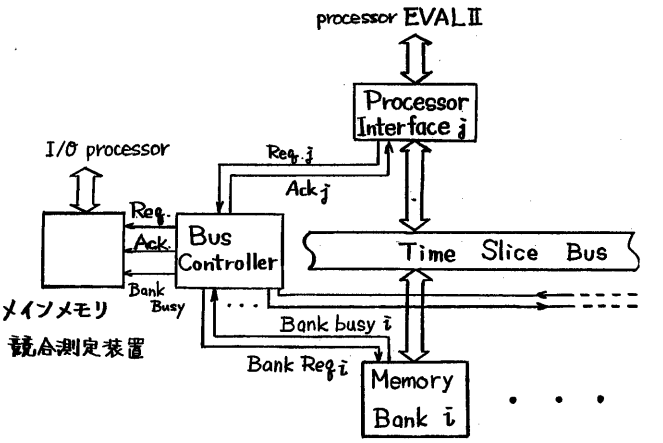

図２．MMと競合測定装置

４－２．測定プログラムについて

測定は、LISP並列処理インタプリタで、LISPのプログラムを実行して行った。プログラムについては、メモリ競合の影響が顕在化するような、十分な並列度とメモリアクセス頻度の高さを考慮した。ここでは、BITA－7，TARAI－4を用いた（第３回LISPコンテストより）。

５．測定結果

５－１，メモリ競合の影響

図３に各プログラム実行時間に対する各EVALⅡとMMの稼動率および、競合の実行時間への影響を示す。EVALⅡ Idleは、処理すべきプロセスを持たない時間である。EVALⅡ Busyは、命令実行頻度を測定し、算出した処理時間（MMアクセス待ちを除く）で、MM Busyは、各EVALⅡが必要とするMMアクセス動作時間である。そして、競合の影響として、測定により求めた待ち時間を用いている。

MMの競合の影響は、図３で、実行時間の３～５％となっている。

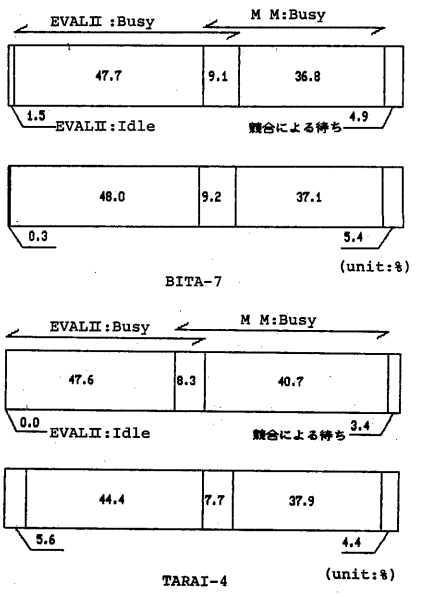

図３．EVAL，MMの稼動率と競合による待ち（上：1号機，下：2号機）

５－２，バス競合の影響

MMの競合のうち、バス競合による待ち時間の割合を図４に示す。バス競合はMMの競合の６～９％を占め、残りはバンク競合による。

5－3、バンク競合

図5にそれぞれのプログラムでの、各バンクごとのアクセス回数（2台の合計）と、1アクセスあたりのバンク競合による待ち時間（MMのクロックサイクルで表わす）を示す。

リスト領域については、図5ではインタリーブしているのに対し、図6にインタリーブしていない場合のアクセス回数とバンク競合を示す。

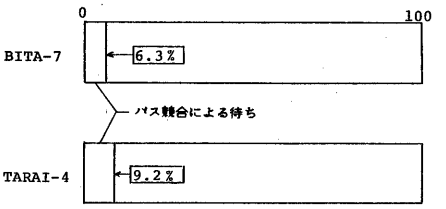

図4．全競合に対するバス競合の割合

5-4、consにおけるフリーセル獲得

バンク0へのアクセスのうち、フリーリストのbase registerへのアクセスがBITA－7の場合、45%を占める。その際排他制御を行うので、バンクの占有時間は、MMのクロックで約26サイクルとなる（read時の占有時間は通常5サイクル）。その影響について、図7に、バンク0からフリーリストのbase registerを分離して測定した結果を示す。

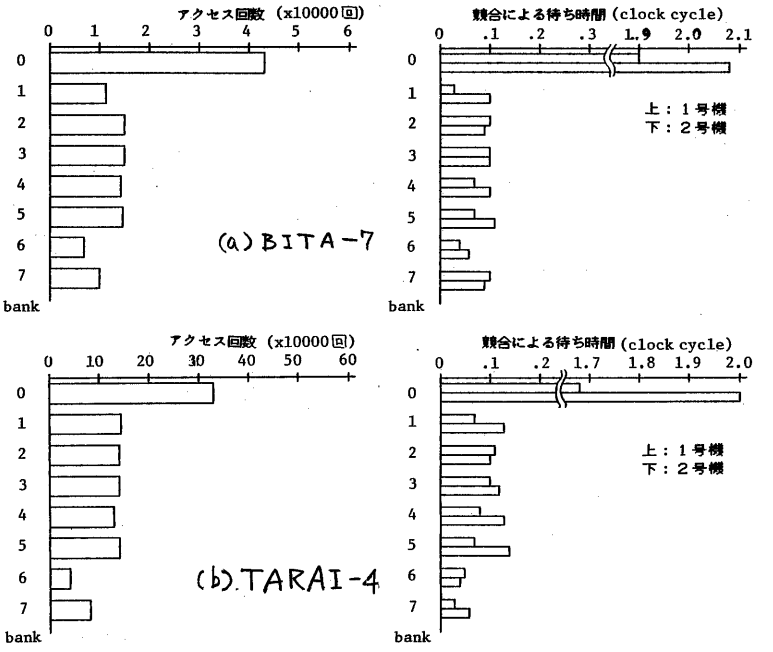

図5．バンクごとのアクセス回数と、バンク競合

5-5、g-bufferの効果

開発当初の設計では、EVAL II は未処理プロセスを獲得する際に、g-listをたぐり、フレームのSTATUSを順に、しかも排他的に調べていかなければならない（g-list searchと呼ぶ）。g-bufferを製作したことによって、

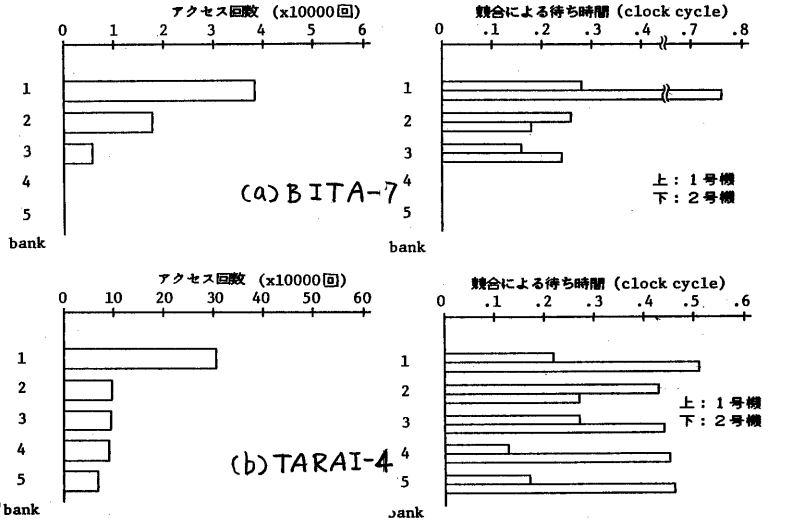

図6．リスト領域をインタリーブしていない場合のアクセス回数とバンク競合

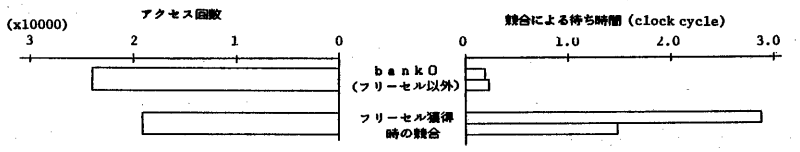

図7. consにおけるフリーセル獲得の競合(BITA-7)

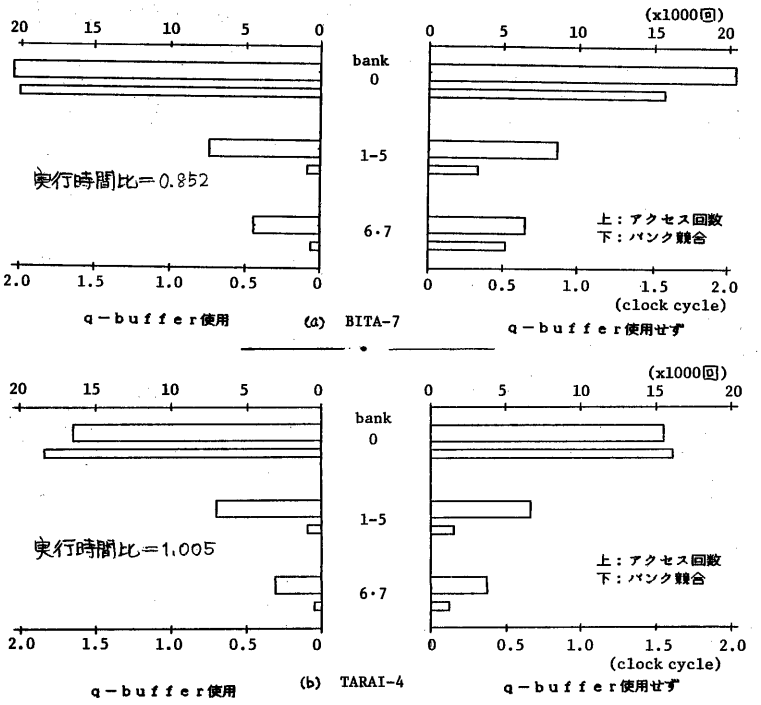

図8. q-bufferの効果

未処理プロセスはq-bufferに登録され、EVALⅡはq-listの検索が不要となった。q-bufferはプロセス割付けのオーバーヘッドを減らすハードウェアであるのと同時に、MMでの競合を減らす効果も期待できる。

図8にq-bufferを使用した場合、使用しなかった場合について各領域での1バンクあたりのアクセス回数と、1アクセスあたりのバンク競合による平均待ち時間、および実行時間の比を示す。

6. 考察

6-1. EVALⅡ2台でのメモリ競合

図3においてEVALⅡIdle以外の状態を、EVALⅡがプロセス実行中であるとしたとき、BITA, TARAIは共に十分高い並列度をもったプログラムとなっていた。これらのexprプログラムをインタプリタで実行する場合、メモリアクセス頻度はプログラムに依存していないと思われる。したがって、今回用いたプログラムは、MMの競合の影響を顕著に示す例といえ、EVALⅡ2台の場合の競合の影響は、実行時間の5%をこえない。

5章で示した結果をもとに、メモリ競合とMMの構成との関係を検討する。

(1) バス競合

図3, 図4より、バス競合による影響は、全実行時間に対し、0.5%以下であり、時分割して1アクセス中におけるバス占有時間を小さくすることにより、共有バス方式であっても、効果がでると考えられる。

(2) バンク競合とbase registerに対するアクセス競合

図5より、バンク0へのアクセスが集中し、またそのときの競合による平均待ち時間が非常に大きいことがわかる。

これは、base registerにおける共有資源の獲得のためのアクセス競合が原因と考えられる。獲得時の排他制御により、バンクの占有時間が大きくなり、競合時の待ち時間が増大するためである。

特にconsでのフリーセル獲得は、リスト処理の基本操作の一つであるためアクセス回数が多く、影響が大きいと考えられる。バンク0から、このフリーリストのbase registerを分離して、測定した結果が図7である。このときバンク0における バンク競合量は、1/17に減少している。減少した時間は、バンク競合による待ち時間の約86%に当たる。

base registerに対するアクセス競合は、本質的には避けられないものである。図7では、フリーリストのbase registerそのものでのアクセス競合が他のバンクでの競合より大きいことを示している。

この問題は、プロセッサ台数の増加にしたがって、ますます大きく影響する。したがって、LISP並列処理におけるメモリ競合に関する最大の問題点は、consにおけるフリーセル獲得時の競合ということができる。

フリーリストで管理する場合は、その対策として、base registerのMMからの分離が考えられる。またMM自身にフリーセル管理機能を持たせることが考えられる。前者に比して特に後者は、かなりのハードウェア量を必要とする。

(3) リスト領域の構成

リスト領域については、3-3で述べたフリーリストの作成方式をとり、各バンクへのアクセスの分散をはかっている。ここで用いられたプログラムに対して、図5によりその効果が確められた。また実際の競合量については、インタリーブを用いていない図6の結果と比較して、総競合量で約30%におさえられている。

(4) q-bufferの効果

図10によると、BITAとTARAIで、異なった結果が出ている。BITAの場合、実行時間で約15%，メモリ競合による待ち時間で、約22%の向上が見られたが、TARAIの場合、逆に実行時間で5.3%，待ち時間で8.5%低下していた。

これは、q-list searchのアルゴリズムと、LISPプログラムによっては、未処理プロセス獲得のオーバーヘッドが十分小さくなる場合が存在することを示している。オーバーヘッドに関する動特性の測定等を行った上で評価する必要があり、今後の課題のひとつである。

しかしq-bufferを使用しない場合、並列度が低いプログラム等ではidle状態が続き、そのidleプロセッサのq-list searchに基づくメモリ競合がbusyなプロセッサへ影響を与えるが、これがq-bufferへのアクセスにかわることにより影響をなくすことになる。プロセッサ台数を増やした場合このq-bufferの効果がより期待できる。

6-2. プロセッサ台数とメモリ構成

MMのハードウェア構成に対する評価をするには、EVALIIを3台，4台と増やした場合にメモリ競合により処理効率がどの程度低下するかを知ることが必要である。そこで、今回、明らかにしたメモリ競合のふるまいを用いて、できるだけ簡単なモデルを考えて、処理効率の推定を試みた。

今回の測定によると、バンクに対するアクセス回数には、極端な差が出た。一方、プロセッサによるMMアクセス回数等には差がなく、どのプロセッサのふるまいも同じと仮定する。またアクセス回数と、バンク占有時間が大きな要因であると推定できる。バンク競合については、各バンクごとに異なり、バス競合はバンク競合と独立なものと仮定する。

まず、EVALⅡ1台のみ稼動させた場合の処理時間をT_1，n台同時に稼動させた場合をT_nとする。T_1はメモリ競合の影響がない場合である。ここで各バンクごとの処理を考える。1台のEVALⅡのバンクiへのアクセス回数を$N(i)$とすると、バンクiへの平均アクセス間隔$Th(i)$は、

$$Th(i) = T_1 / N(i)$$

と書ける。そのときの、1アクセスに対する平均バンク占有時間を$\alpha(i)$とすると、n台のEVALⅡのバンクiへのアクセスで生じる平均バンク競合待ち時間$t_w(n,i)$は、

$$t_w(n,i) = \sum_{j=2}^{n} \left\{ \binom{n}{j} \cdot \left(\frac{\alpha(i)}{Th(i)}\right)^{j-1} \cdot \left(1 - \frac{\alpha(i)}{Th(i)}\right)^{n-j} \cdot \frac{(j-1)^2}{2} \cdot \alpha(i) \right\}$$

となる。次にバス競合については、平均メモリアクセス間隔をTh_bとすると、

$$Th_b = T_1 / \sum_{i=0}^{7} N(i)$$

で、平均バス占有時間をα_bとする。α_bは一定である。バス競合による平均待ち時間$t_b(n)$は、

$$t_b(n) = \sum_{j=2}^{n} \left\{ \binom{n}{j} \cdot \left(\frac{\alpha_b}{Th_b}\right)^{j-1} \cdot \left(1 - \frac{\alpha_b}{Th_b}\right)^{n-j} \cdot \frac{(j-1)^2}{2} \cdot \alpha_b \right\}$$

となる。したがって、n台のEVALⅡを同時に稼動させた場合の処理能力として、

$$\frac{nT_1}{T_n} = n / \left\{ 1 + \sum_{i=0}^{7} \frac{t_w(n,i)}{Th(i)} + t_b(n) \right\}$$

を用いて、以下、EVLISマシンのハードウェア構成を評価する。

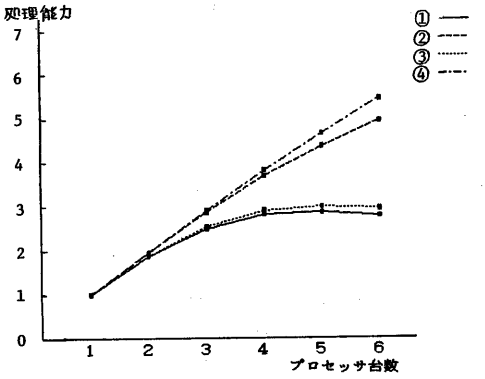

図9、プロセッサ台数と処理性能

このモデルに対し、BITA-7のメモリアクセス回数，実行時間を代入し、$t_w(2,i)$を計算したところ実測値より10～20%小さい値となった。これは、アクセス間隔等を平均して扱っているが実際には、時間的にもかたよりがあるためと思われる。

前出のシミュレーションではEVALⅡ4台まで効率向上を示していたが、そのとき除外していたメモリ競合等の影響を取り出したのが、今回示した式であり、総合的な性能を得るためには、この計算値でも4台までの効率向上が期待される。

図9に計算結果を示す。①は、現構成を用いた場合であり、②はフリーリストのbase registerの影響を取り除いた場合で、③，④はそれぞれ①，②からバス競合の影響を無視した場合である。

図9より、現在の構成では、4台で処理効率の向上が押えられる。しかし、フリーセルのbase registerの問題が解決できれば、②で示されるように、潜在的な能力は十分あると評価できる。またバス競合については、時分割共有バスが、EVALⅡ5台程度までなら、バス競合による性能低下の少ない結合方式であると評価できる。

7、おわりに

今回、EVLISマシンにおけるメモリ競合の問題をとり上げ、LISP並列処理インタプリタにおけるメモリ競合を直接測定した。これによりconsでのフリーセル獲得時の競合が、処理効率向上のボトルネックであることが明らかになり、またEVALⅡを増設した場合の時分割共有バスの有効性が確認できた。

【参考文献】
1)西開地 他，EVLISマシンの並列処理における動特性，29回情処全大6B-1(1984)
2)斎藤 他，EVLISマシンのための並列処理特性の測定と評価，21回情処全大11-9(1970)
3)前川 他，高速LISPマシンとリスト処理プロセッサEVALⅡ，情処学会論文誌，Vol.24,No.5(1983)
4)前川 他，試作EVLISマシンのEVALⅡと開発支援機能，記号処理17-1(1982)