ブロードキャスト・メモリ結合形並列計算機の試作

PARALLEL COMPUTER SYSTEM WITH BROADCAST MEMORY

小畑正貴　角木裕成　西野佐登史　田中敏幸　金田悠紀夫　前川禎男
Masaki KOHATA, Hironari KAKUKI, Satoshi NISHINO, Toshiyuki TANAKA, Yukio KANEDA, Sadao MAEKAWA
神戸大学・工学部
Faculty of Engineering, KOBE University

〔1〕まえがき

コンピュータの能力向上にともない、数値計算の大規模高速化が実現されてきているが、有限要素法・ディジタルフィルタ・各種シミュレーション技法などの発展普及のため、ますます大規模化・高速化・低コスト化の要求が高まってきている。大規模数値計算を高速に行うことを目的とした計算機としては、大規模なものではCRAY－1などのスーパーコンピュータが、また、小規模ではミニコンピュータにアレイプロセッサを付加したものが使われている。しかしながら、これらのコンピュータはコストパーフォーマンスなどの点で、一般ユーザに手軽に利用できるというものではない。

一方、LSI技術の発達により、16ビット並列処理のマイクロプロセッサと高速数値計算を目的とした数値データプロセッサが出現し、マイクロコンピュータレベルでの数値計算の高速化は急速に向上してきている。さらに、32ビットマイクロプロセッサと数値データプロセッサが実現すれば、数値計算におけるコストパーフォーマンスは飛躍的に向上し、より手軽に大規模数値計算を行うことができるものと期待できる。

本システムは、ブロードキャスト・メモリと呼ぶメモリシステムによって結合されたMIMD形多重マイクロプロセッサシステムである。近年、各所で種々の多重マイクロプロセッサシステムが開発されているが[1]、本システムは大規模数値計算の高速処理を主な目的として試作された。

CPUには16ビットマイクロプロセッサと、高速数値演算用データプロセッサとを用い、1ユニット当りの計算能力も高い。われわれは、高速性・コストパーフォーマンス向上・手軽さを目標としてシステム設計を行った。以下に本システムの特徴を述べる。

(1) 共通バス構成
各プロセッサは共通バスにより結合されており、プロセッサの数を容易に増すことができる。

(2) 1ボード構成
ハードウェアの簡略化をはかり、1台のプロセッサユニットを1ボードに組み込むことを目標とした。これにより、量産によるコストの低下が期待できる。

(3) 分散形共有メモリ
共有メモリは各ユニットに分散された形で実装される。これは、並列処理されるデータの多くはローカリティを持っており、これらは共通バスを使わずにアクセスできる方がオーバーヘッドが少なくなるからである。もちろん、共通バスを使って全プロセッサからアクセスすることも可能である。

(4) ブロードキャストメモリ
全プロセッサに必要なデータを効率よく、同時に転送できるブロードキャスト転送機能を持つ。

(5) 並列動作制御機能
行列計算だけでなく種々の並列処理に対して柔軟な対応ができるよう、セマフォ操作

・同期機能・プロセッサ間相互割込などの並列動作制御機能を持つ。

以下、ハードウェアの詳細、並列プロセス制御機構、動作のシミュレーション結果について述べる。

〔2〕ハードウェア

2.1 プロセッサ及びメモリ

システム全体の構成を図1に示す。並列計算を行うべきプロセッサP1～Pnは共通バスによって結合されている。CPUにはインテル社の16ビットマイクロプロセッサ8086に、数値データプロセッサ8087を付加したものを用いる。[2] 8086は1Mバイトのアドレス空間を持つ高性能マイクロプロセッサである。また、8087は8086に付加して用いる数値データプロセッサであり、浮動小数点演算などの数値演算を高速（例えば、単精度浮動小数点乗算約20μsec）に行う。8086と8087は並列動作が可能であり、8087が数値演算をしている間に、並行して8086は他の処理を行うことができる。

1Mバイトのアドレス空間を、64Kバイトずつのセグメントに分け、1つのセグメントに1台のプロセッサを割り当てる。メモリのアドレス配分を図2に示す。各セグメント内のメモリはそのアクセス法によってローカルメモリ・データメモリ・パケットメモリの3種類に分けられる。

ローカルメモリは各プロセッサのローカルバスに接続されており、各プロセッサは他からの

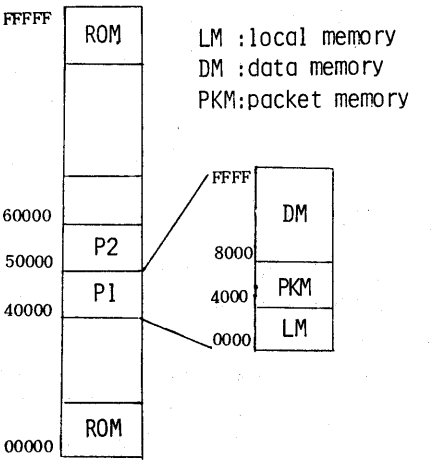


図2 アドレス配分

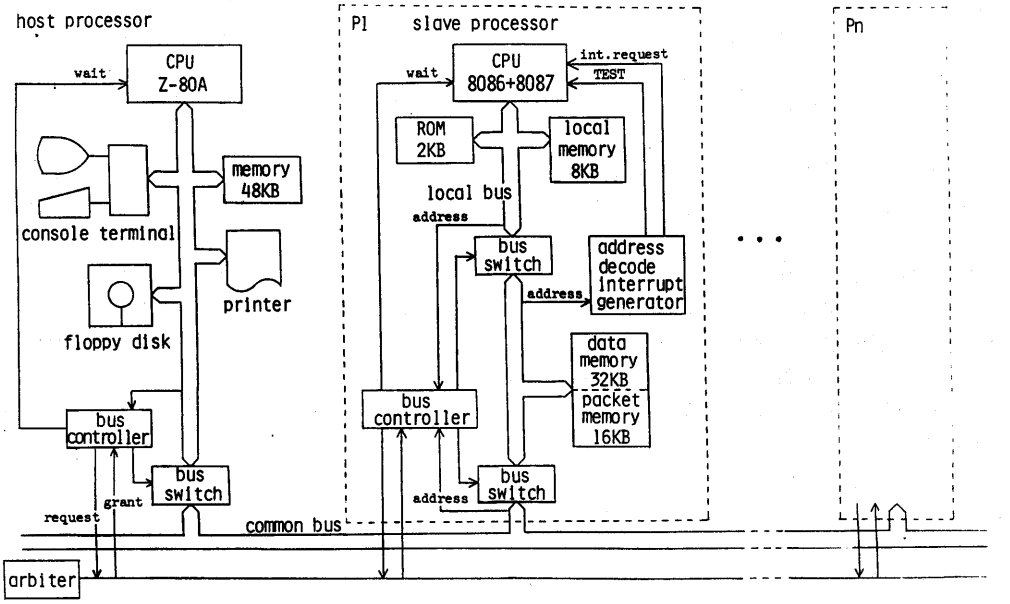


図1 システム構成

影響をうけずにこれをアクセスできる。ローカルメモリは主としてプログラムの蓄積とスタックに用いる。ローカルメモリを用いることにより、共通バスおよび共有メモリへのアクセスの競合にともなう時間的損失を軽減できる。

データメモリはローカルメモリと共有メモリの両方の性質を持つ。すなわち、各プロセッサは自分のデータメモリに対しては共通バスを使わずに独立してアクセスでき、共通バスを使って他のデータメモリをアクセスすることもできる。通常データメモリにはローカルなデータが格納される。

ブロードキャストメモリはパケットメモリの集合として構成される。パケットメモリは、読み出し時には共通バスを使わずに独立して読み出すことができる。一方書き込み時には、データは共通バスを通してすべてのパケットメモリの同じ位置に自動的にブロードキャスト転送される。

このようなブロードキャストメモリを用いることにより、ベクトル・行列計算などにおける並列処理を効率よく行うことができる。[3] ブロードキャストメモリには並列計算を行う時の共有データを入れるほか、並列プロセスの同期や相互排除を制御するためのセマフォの格納にも用いる。

以上3種のメモリは、図2の様にアドレスによって機能の判別がなされる。

なお、本システムでは、共有メモリはデータメモリとパケットメモリに分散された形で実現されており、独立した共有メモリを持っていない。

2.2 バス構成とメモリアクセスの機構

バス構成とメモリの位置を図3に示す。ローカルバスと共通バスは2つのバススイッチ（SWLとSWC）を通して結ばれる。ローカルメモリはローカルバス上にあり、バススイッチの状態に関係なくCPUからアクセスできる。

データメモリとパケットメモリは2つのスイッチの間に位置する。これらのメモリがどのようにアクセスされるかは、その時のバススイッチの状態による。バススイッチの状態とメモリアクセスとの関係を表1に示す。

SWCは双方向性であり、アクセスの方向によってON/OFFと同時に方向も制御する必要がある。

プロセッサ1がデータをブロードキャスト転送する場合のバススイッチの状態が図3である。プロセッサ1は両方のスイッチを閉じ、その他のすべてのプロセッサはSWCのみを閉じる。この時、プロセッサ2～nが自分のデータメモリあるいはパケットメモリをアクセスしようとすると、そのプロセッサはブロードキャスト転送が終るまで待たねばならないが、命令フェッチ等のローカルアクセスは待たずに行える。

バススイッチの制御機構を図4に示す。リクエスト発生部ではローカルバスおよび共通バスのアドレス・Read・Write・Lock信号から表2に示す各リクエスト信号を発生する。コントロール部はこのリクエスト信号を受けとって、バススイッチとCPUに制御信号を出す。Lockはセマフォオペレーションを行う時に出される。この信号によりバスはロックされ、セマフォの読み書きは一連のサイクルで行われる。

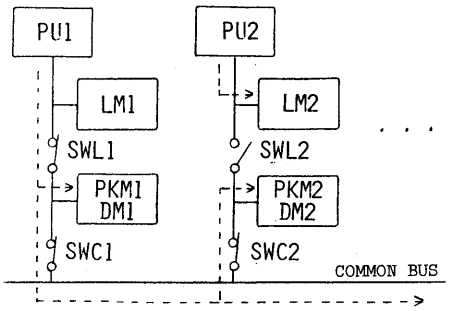


図3 バス構成

SWL	SWC	メモリ アクセス
ON	ON (OUT)	他データメモリ，パケットメモリ（書込み）
ON	OFF	自データメモリ，パケットメモリ（読出し）
OFF	ON (IN)	他プロセッサからのアクセス
OFF	OFF	ローカルメモリ

表1 バススイッチの状態

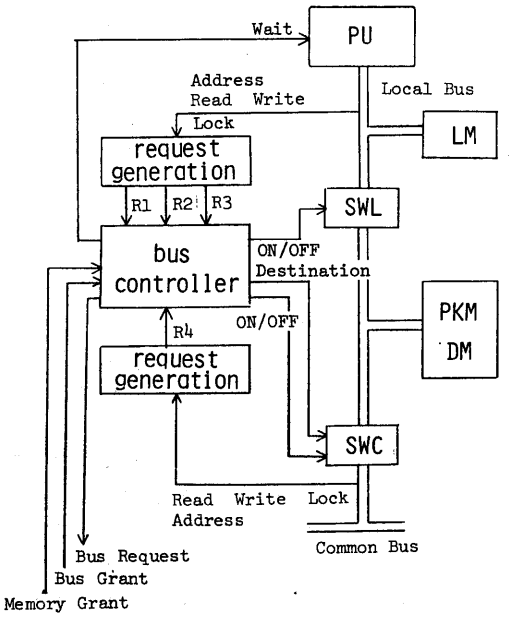


図4　バススイッチの制御

A19-A16	self		other
A15 A14	PKM	DM	DM
read	R1	R1	R3
write	R2	R1	R3
lock	R2	R1	R3

Request from Local Bus

A19-A16	self		other
A15 A14	PKM	DM	DM
read	X	R4	X
write	R4	R4	X
lock	R4	R4	X

Request from Common Bus

R 1 ：自データメモリへの要求
R 2 ：ブロードキャスト転送要求
R 3 ：他データメモリへの要求
R 4 ：他プロセッサからの要求

表2　メモリへのリクエスト信号

R1とR4は自メモリに対する要求信号であり、共通バスに対する要求は出力されない。R1とR4に対するコントローラの出力、メモリアクセスのタイミングを図5に示す。R4はR1よりも高いプライオリティを持っており、同時にリクエストが出された時にはR1に対してWaitがかけられる。これは、共通バスが使用される時間をへらすためである。

また、Memory Grant信号はメモリアクセスを要求してきたプロセッサに対するWait信号と考えることができる。

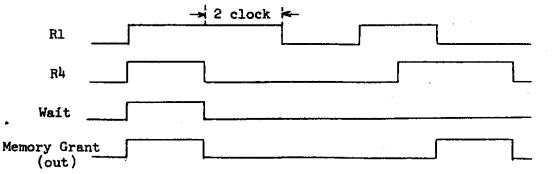


図5　データメモリのアクセスタイミング

R2とR3は他プロセッサのメモリに対する要求信号である。他メモリへのアクセスは次の手順で行われる。

(1) 共通バス使用リクエストを出す。
(2) アービタより許可がおりるまでWait。
(3) SWL，SWCを閉じる。
(4) 相手メモリが使われている時は、それが終るまでWait。
(5) 相手メモリをアクセスする。

ここで、(4)における判断は、R3に対しては指定セグメントのメモリからの許可信号を、R2に対しては全セグメントからの許可信号を用いる。この両信号は各ユニットからオープンコレクタ出力されており、ワイヤードロジックにより作成される。

ブロードキャスト転送の時のタイミングを図6に示す。1回目はすぐに許可がおりた場合であり、2回目は共通バスの獲得と相手メモリからの許可を得る時の2回に待たされた場合である。

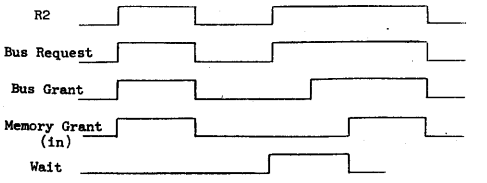


図6　ブロードキャスト・アクセスタイミング

Lock 信号によりバスロックを行えば、読み出しと書き込みを一連のメモリアクセスサイクルに行うことができ、セマフォ操作を行うことが可能となる。セマフォに対する test and set のタイミングを図7に示す。

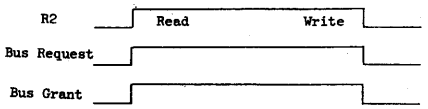


図7　バスロック・アクセスタイミング

共通バス使用の要求に対する調停については、試作システムではハードウェアの簡易な固定プライオリティのデイジーチェイン方式をとっている。これは

(1) プロセッサ数がそれほど多くなく
(2) 多くの並列処理において、共通バスに対する要求が多くない

場合においては、プライオリティの差によるアンバランスが無視できると考えられるからである。ただし、上の理由によるアンバランスが無視できない場合には、リングアービタ[4]等の調停方式に変更することも可能である。

2.3　プロセッサ間割込機構

本システムは任意のプロセッサ間で割込をかけ合うことのできるプロセッサ間割込機能を持っており、これによって柔軟な並列処理を行うことができる。

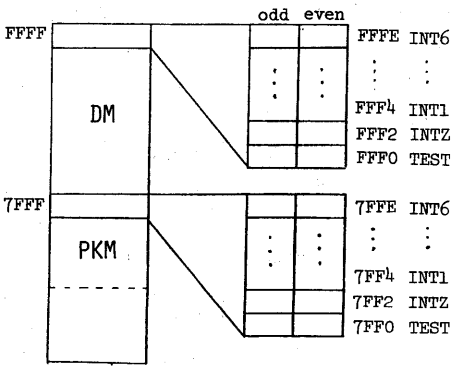


図8　割込みアドレス配置

割込の発生は、各ユニットのデータメモリあるいはパケットメモリの特定アドレス（図8）にデータを書き込むことによってなされる。この時書き込まれるデータは、どのプロセッサが割込をかけてきたかを知るための識別子となる。

割込を発生するアドレスは図8に示したアドレス空間のうちの偶数番地のみとなっており、奇数番地は、複数のプロセッサから割込をかける時に処理を正常に行うためのセマフォとして用いる。

たとえば、プロセッサAがプロセッサBに割込1をかける時は、AはBのデータメモリのFFF4番地に自分のユニット番号を書けばよい。一方、パケットメモリの7FF4番地に書き込むと、データはブロードキャスト転送機能によって全パケットメモリに書かれるので、結果としてすべてのプロセッサに同時に割込がかかることになる。

アドレスのFFF0と7FF0は割込信号でなく、8086にTEST 信号を発生する。8086にはWAITと呼ばれる命令があり、これを実行すると8086はハードウェアのTEST信号が入るまで命令の実行を止める。この機能によって並列動作の同期をとることが容易となる。

また、INTZ信号はホストプロセッサへの割込である。

2.4　ホストプロセッサ

ホストプロセッサは1つのバススイッチを介して共通バスに接続される（図1）。ホストプロセッサはスレーブの全データメモリ・パケットメモリに対してアクセス可能であり、プログラムやデータのロード等を行う。また、スレーブプロセッサのスタート・ストップ等の制御も行う。

CPUとしてZ-80を用いている。これは、われわれの研究室では8080系マイクロコンピュータのオペレーティングシステムとして広く使われているCP/M[5]を以前から使用しており、本システムでもこれを使うことにしたためである。ホストプロセッサは64Kバイトのメモリ空間のうち48KバイトをOSに使用し、残り16Kバイトの空間は空けている。ホストからのスレーブのメモリに対するアクセスは、この16Kバイトのウインドを通して行う（図9）。

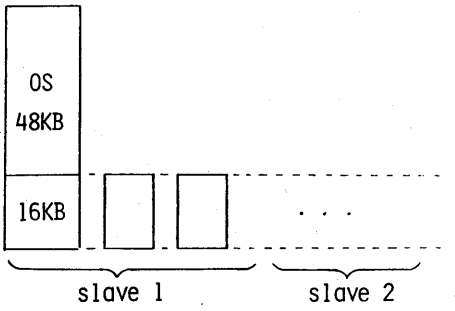


図9　ホストプロセッサ・アドレス配分

回路の簡略化のため、スレーブプロセッサのローカルメモリは、自分以外はアクセスできないようなバス制御を行っているが、これではローカルメモリにプログラムをロードする時、途中にデータメモリあるいはパケットメモリを介さなければならない。この点のトレードオフに関しては、使用を通して考えてゆく。

また、スレーブプロセッサは入出力装置をまったく持っておらず、データの入出力やフロッピーディスクのアクセス等はすべてホストプロセッサを利用する。スレーブからの入出力動作は、データメモリに必要パラメータを書き、ホストプロセッサに割込をかけることによって行う。

[3] ソフトウェア

以下、並行プロセスを制御するための機構を中心にソフトウェアに関して述べる。

3.1 セマフォ操作

本システムでは、相互排除問題をセマフォによって制御する。P, Vオペレーションは図10のプログラムによって実現される。

```
P operation                    V operation

WAIT:   MOV     AL,sem         MOV     sem,1
        AND     AL,AL
        JZ      WAIT
        SUB     AL,AL
        LOCK
        XCHG    sem,AL
        AND     AL,AL
        JZ      WAIT
```

図10　セマフォ操作

図7で、Pオペレーションの前半では共通バスを使わずにセマフォを調べており、共通バスの負荷を軽くしている。LOCK命令によって次のXCHG命令の間バスがロックされ、セマフォのtest and setが行われる。

3.2 同期

図11に示すような並列プロセスの同期をとる場合のメカニズムについて考える。

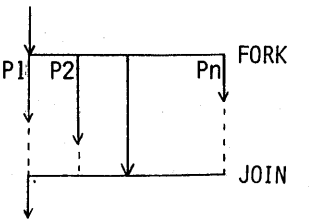


図11　並行プロセスの同期

FORKでは、プロセッサ1はカウンタにプロセッサ数nを書き込んでから全プロセッサに並行プロセス実行の起動をかける。各プロセッサは自分の実行が終るとカウンタをデクリメントし、WAIT命令を実行して停止する。この時、カウンタ操作に対してはセマフォを用いる必要がある。最後のプロセッサがデクリメントした時にカウンタが0になるので、このプロセッサがプロセッサ1に起動をかけ、JOIN動作が完了する。

3.3 ホストプロセッサの機能

8086のプログラム開発は、CP/M上のクロスアセンブラを用いて行う。開発したプログラムのロードや実行は、ホストプロセッサ上に作成した簡易モニタプログラムによって行う。簡易モニタの機能を示す。

(1) メモリの表示・書き替え・転送
(2) プログラムのロード
(3) スレーブプロセッサのスタート・ストップ
(4) レジスタの表示
(5) スレーブプロセッサのI/Oとしての動作

スレーブプロセッサを起動した後は、ホストプロセッサは受動的な立場となり、インテリジェントI/O的な働きをする。

3.4 応用プログラム

本システムの応用分野の1つである大次元連立一次方程式の並列計算については、文献(3)等で報告しているので、ここでは詳細を述べず、ガウス消去法の前進消去を例にとった簡単な説明にとどめておく。

$Ax = b$ をガウス消去で解く場合、行列Aおよびベクトルbはn行おきに各プロセッサのデータメモリに格納される(図12)。ピボット行はそれを持つプロセッサから全プロセッサにブロードキャスト転送され、各プロセッサはこれを用いて並列に消去を行ってゆく。

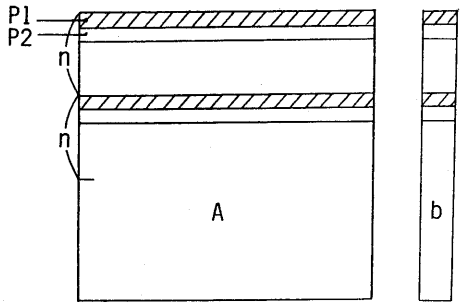


図12 ガウス消去法でのデータ配分

〔4〕シミュレーション

本システムの試作機は現在プロセッサ数4であるが、さらにプロセッサ数を増すことも考えている。しかし多くの場合、プロセッサ数の増加に対して計算速度の向上はリニアには増加せず、徐々に鈍化する。[6] そこでわれわれはプロセッサ数を増加した時のシステムの性能の評価を行うために、FORTRANで作成したシミュレータを用いてシミュレーションを行った。ここでは、連立一次方程式をガウス・ザイデル法を用いて並列計算する場合を想定し、シミュレーションを行いその結果を求めた。

$Ax = b$ で表わされる方程式をガウス・ザイデル法を用いて並列計算する場合、$x^{(i)}$ はブロードキャストメモリに、Aとbは行ごとに分割して各データメモリに格納して計算を進めてゆく。

ガウス・ザイデル法による連立一次方程式の並列計算のシミュレーションは次の条件によって行った。

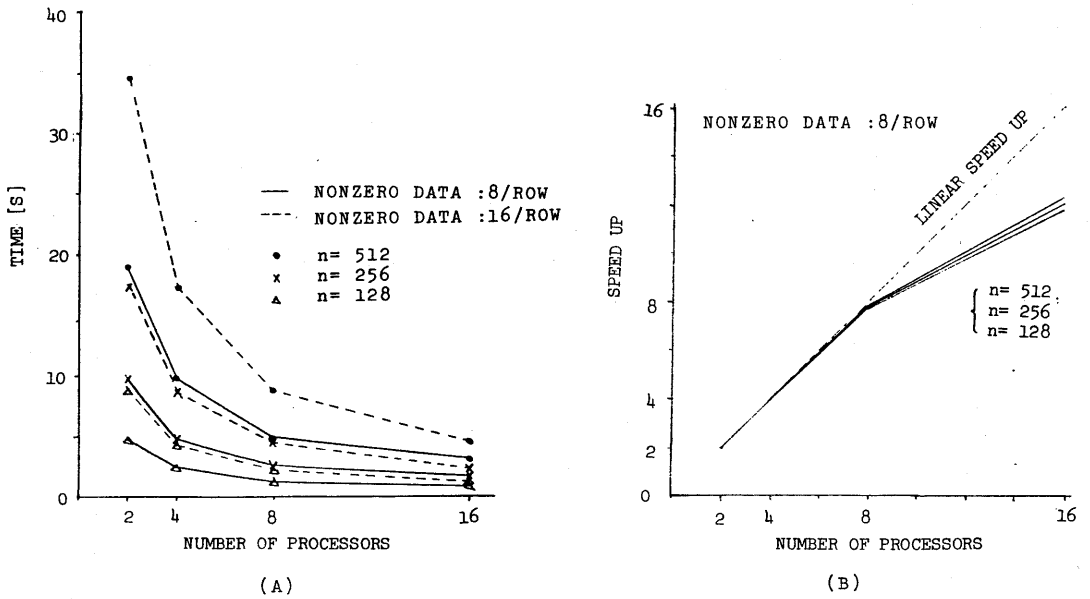
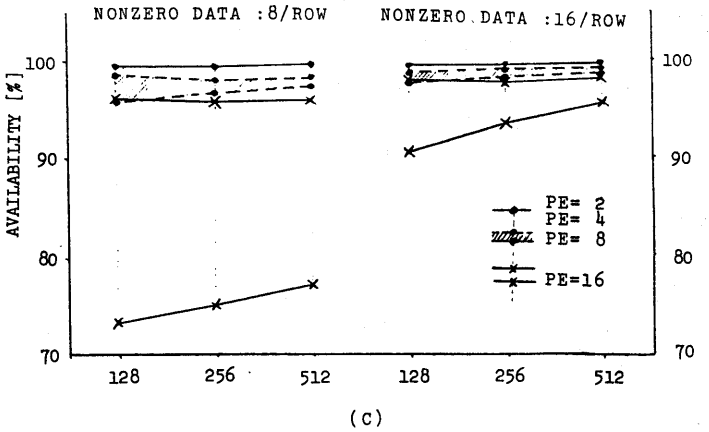


図13 ガウス・ザイデル法のシミュレーション結果



(C)

プロセッサ台数	2,4,8,16
変数の数	128,256,512
一行あたりの非ゼロデータ数	8,10,16
反復回数	100
使用プロセッサ	8086+8087
プライオリティ	固定

図13（A）～（C）に結果を示す。

（B）はプロセッサ数の増加に対する計算速度の向上を表わす。8台まではほとんどリニアに増加するが、16台になると速度の向上は鈍化する。この例では、リニアな関係に対して25%程度低下している。

（C）に示しているアベイラビリティは次式で表わされる量である。

$$\frac{\text{計算時間} - \text{Wait時間}}{\text{計算時間}} \times 100 \quad (\%)$$

Wait 時間はバス競合のために実行が停止する時間である。

アベイラビリティは各プロセッサに対して個別に計算しており、（C）には、このうち最大値と最小値が示されている。また、図の左と右は全データアクセスに対するブロードキャストアクセス頻度の違いを表わしており、左がブロードキャスト・アクセス頻度が多い場合、右が少ない場合である。

プロセッサ数が多くなるほど、またブロードキャストアクセスが多くなるほどアベイラビリティは低下しており、最悪なものでは25%も待たされている。並列計算では、全体の計算時間は最も遅いプロセッサに左右され、（B）の25%の能率低下は（C）の最悪のものと対応している。

（C）において、プロセッサ数16の時、アベイラビリティに大きな差がでたのはアービタのプライオリティを固定としたためであり、ブロードキャスト・アクセスが多くなるとプライオリティを平等に与える必要がある。しかし、8台までは大きな差はなく、固定プライオリティでもそれほど悪くならない。

以上はわれわれが行ったシミュレーションの一例であるが、これによっても本システムの行列計算への有効性が示せたと考えている。

〔5〕むすび

大規模数値計算の並列処理による高速化を主な目的として試作したブロードキャストメモリ結合形並列計算機システムについて述べた。

本システムは高機能16ビットマイクロプロセッサと数値データプロセッサの特長を生かした設計が行われており、次のような点で実用性を重視した最初の設計目標が達成できたと考えられる。

(1) 数値データプロセッサと並列計算による高速化

(2) ハードウェア量を少なくできたこと。試作システムでは1台当りの部品数は、メモリを除いてLSI3個，TTL－IC約50個で構成されており、価格や実装の手軽さなどの点ですぐれている。

また、本システムは並列動作制御機能やプロセッサ間割込機能などの機能を持っており、数値計算だけでなく、各種並列動作の研究に幅広く応用できると考える。

現在、4台のスレーブプロセッサによる試作システムが完成しており、連立方程式の並列処理プログラムの開発とシステムの評価を行っている。今後さらに多方面の分野に対する応用をおし進め、本システムの実用性を実証したいと考えている。また、並列処理機能をもったOSや高級言語の実現も興味のあるところであり、研究を進めてゆきたい。

参考文献

1) 村岡，坂間：並列処理技術，信学誌，Vol. 63, No. 10, pp. 1064～1071
2) The 8086 Family user's Manual, Intel Co.
3) 金田：ブロードキャスト・メモリを持つ並列計算機システムによる大次元連立一次方程式の並列計算，信学技報，EC 80-42, 1980
4) 増山，吉田：非同期式アービタの合成法，信学論，Vol. J64-D, No. 4, pp. 332～339, 1981
5) CP/M User's Guide, Digital Research
6) Jones, A.K., Schwarz, P.: Experience Using Multiprocessor Systems, Computing Surveys, Vol. 12, No. 2, June 1980