相互診断機能をもつマルチ・マイクロプロセッサシステム AKOVST

A MULTIPLE MICROPROCESSOR SYSTEM WITH MUTUAL DIAGNOSING CAPABILITY

阿江 忠　高橋 浩一　伊藤 鶴美　千葉 治義
Tadashi AE　Koich TAKAHASHI　Turumi ITO　Haruyoshi CHIBA
広島大学・工学部
Faculty of Engineering Hiroshima University

あらまし

並列処理機能をもつマルチマイクロプロセッサシステム（処理用プロセッサ4台とスーパバイザ1台）を製作した。特長は，1.プロセッサ間交信を極力ハードウェア化することでOSを小さくしたこと，2.相互に診断しあう機能を設け信頼性の向上に努めたことにある。これらの2点を当面の目標である画像表示システムとして使用した場合について述べるが，現在計画中の本格的なマルチマイクロプロセッサシステム製作の指針となる。

1．まえがき

マイクロプロセッサがLSI技術の進歩に伴い安価に供給されるようになって，従来，どちらかといえば理論面の研究が先行していた並列処理計算機の研究が製作をともなって行なわれるようになってきている。

我々も同様の趣旨でマルチマイクロプロセッサシステムの製作にとりかかったが[1]，当初は小規模なシステムではあるが，(i)特に使用目的を定めず，より大規模なシステムのパイロットモデルとしての使用（モード1），(ii)専用目的として画像表示のマルチプロセッサ化（モード2）という欲張った使い方のシステムAKOVSTを昨年度ひとまず完成した[2]。当初の予定，プロセッサ数8個は費用等の点から4個とせざるを得なかったが，おおむね予定どおりに作られている。

特長としては，従来の諸研究から考察した結果[3]，プロセッサ数の上限を大きくするには極力，オーバーヘッドを小さくする必要があり，AKOVSTでは，オーバーヘッドの減少のため極力ハードウェア化する方針をとっていることである。具体的には，メモリのコンテンションに伴う時間遅れを少なくするためのハードウェア化をはかり，一応の効果が得られた。

その半面，若干データ転送における誤り率を考慮せざるを得なくなった。

本報告では，直接的にはこのような理由で，プロセッサ間相互に診断機能を設けたマルチプロセッサシステムの現状を述べるが，相互診断方式そのものは従来からのシステムにも適用できる性質のものであり，マルチプロセッサシステムにおける相互診断の一般的考察も同時に行っている。

2．プロセッサ間通信の方式

AKOVSTは共有メモリ（16KB）を介してプロセッサ間通信を行う方式（図1）をとっている。

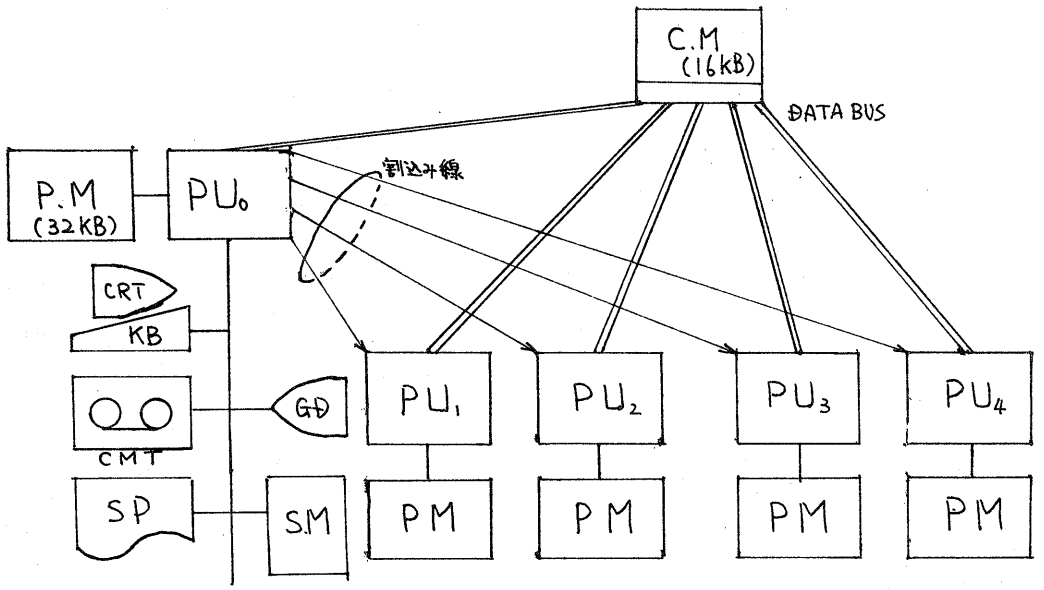
FIFO（First in First out）スケジューリングによりパイプライン並列処理を行おうとするとき，アービタを用いると都合がよい。

アービタを用いて共有メモリ（CM）をシェアする方式は通常は図2(a)のように，

Request-Acknowledge方式（R-A方式と略す）

1). 共有メモリをRequestする。
2). Acknowledgeが帰ってくるまで待機！帰ってきたら共有メモリをaccessする。
3). accessが終了したらreleaseする。

という方式がとられる。



PU : Processor Unit　　CRT : Cathod Ray Tube　　GD : Graphic Display
PM : Private Memory　　KB : Key Boad　　SM : Switch Matrics
CM : Common Memory　　CMT : Cassette Mag. Tape.　　S.P : Sirial Printer

図1　AKOVSTの概略図

R－A方式の場合，1つのプロセッサユニットPUiが資源CMを専有している間，ほかのプロセッサは要求を出し続けて，Acknowledgeの帰ってくるのを待つ。

一方，本方式はオーバーヘッド軽減のため，図2(b)に示す次の方式をとった。

Access-Wait方式　（A－W方式と略す）

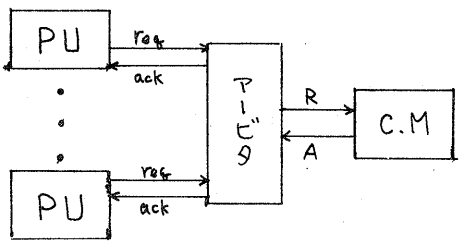
1). 共有メモリをAccessする.

2). もしAccess不可能な場合は，WAIT状態に入る。

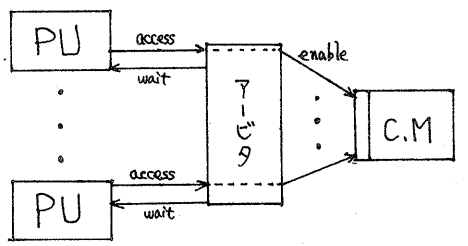
3). 1マシンサイクルが終わるとReleaseする。

具体的なハードウェア構成は図3のとおりである。　なお，アービタは付図1に示すレース形を用いている。

A－W方式とR－A方式との比較を表1に掲げる。　A－W方式とR－A方式との伝送効率をステート数で比較する。（PUiはすべて8080Aである。）



(a) R－A方式



(b) A－W方式

図2　共有メモリアクセスの方式

A－W方式			ステート数
TRANS：	LXI	H, Commonadr.	10
	LXI	D, Count	10
TRANS1：	STAX	B	7
	INX	H	5
	INX	B	5
	DCR	E	5
	JNZ	TRANS1	10

この例ではXByteの転送に要するステート数 T_{S1} は $T_{S1}=20+32X$ ……（1）である。

R－A方式			
Request：	MVI	A, 1	7
	OUT	Port	10
Ack：	IN	AckPort	10
	ANI	1	7
	JNZ	Ack	10
TRANS：	上と同じ		20+32X
Release：	MVI	A, 0	7
	OUT	Port	10

この例ではXByteの転送に要するステート数 T_{S2} は $T_{S2}=81+32X$ ……（2）である。

3．マルチプロセッサ化した画像表示方式

画像表示のための並列処理システムは一般に図4のように表わすことができる。入力はアレイ（配列）で示してあるが リスト構造等でも構わない。いずれにしても，画像表示システムへの入力は大型機，あるいはミニコンなどで処理されて得られたデータと考え，この並列処理を行う画像表示システムは，それらのデータを入力とするインテリジェントターミナルとして位置づけられる。

	R－A方式	A－W方式
ソフトウェア	短いメッセージに対しても Request－Release が必要	要求に対して不要
効率	短いメッセージに対して悪い	メッセージの長短に関らずR－A方式より良い
誤り率	普通	R－A方式より若干悪い

注．AKOSTにおけるA－W方式では，CMを使ったプロセス間の転送にはサムチェックを行い，誤まればRoll Back。システム変数には二度読みでサポートしている。

表1　R-A方式とA-W方式との比較

重要なことは，マルチプロセッサ化されたシステム（Multi Microprocessor System．MMSと略す）はパイプライン処理を行うことで並列化の効果をあげなければならない。汎用機の場合一般にオーバーヘッドのため，並列化に効果のあるプロセッサ数には意外に低い上限があり，悪ければ8，良くても16程度というデータが多い[3]。

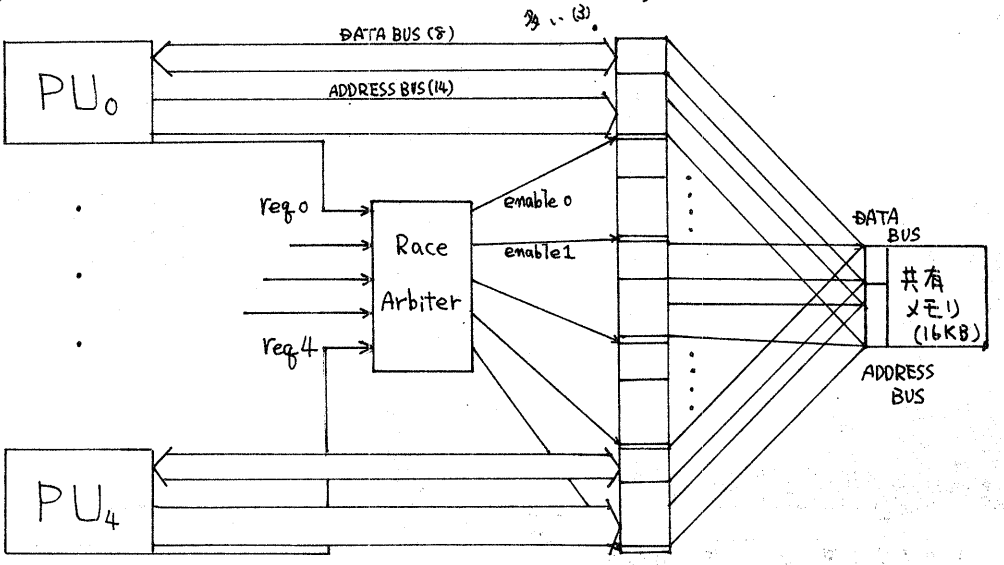


図3　A－W方式による共有メモリシェア

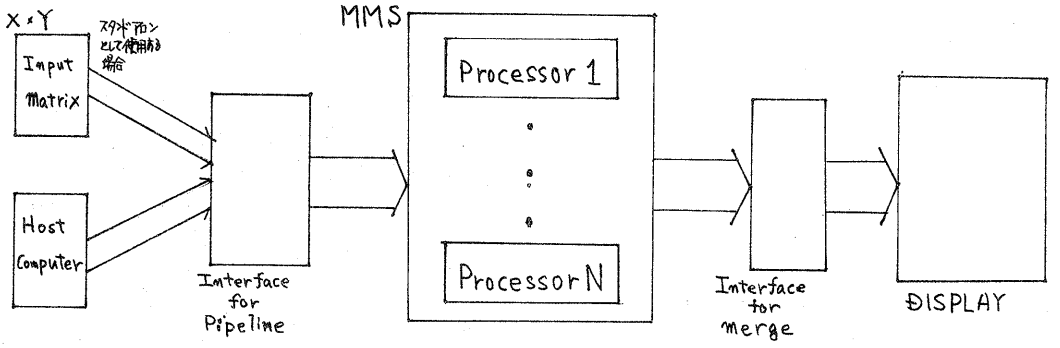


図4　画像表示用マルチマイクロプロセッサシステム

一方，専用機化するとその上限は大きくできシュミレーションによる上限は32程度もあり並列効果を上げる方向で研究が進められている㈱。いずれにしても　並列処理の促進にはオーバーヘッドの問題を解決しなければならない。

なお，AKOUSTではスタンドアロンの場合にも一応の動作が行なえるよう，入力マトリクスとして　16×16のビットパターンを入力する専用装置（写真1）も備えており，ビットマトリクス入力を　接続行列として実際の装置（写真2）へ与え，表示させると写真3のようになる†。

この場合，スーパバイザプロセッサはプロセッサへのデータの分配，集約及びビデオRAMへの転送を受持っている。

具体例を1つ述べると　スーパバイザは16×16のビットパターンを図5のように（1,1）（1,2）（1,3）…要素の順にサーチし，1の要素を見つければ空いているプロセッサPUiをi=1～4の順にサーチし，見つかったプロセッサへ直線を描くよう指示する。（ノードの位置は指定された位置に描かれる。）

なお，ここで述べた画像表示は primitive な一例を述べたにすぎない。　本来，画像表示において処理を伴うような問題（例えば　隠線消去）をインテリジェントターミナルが受持って良いと思われる。　そのような場合のソフトウェアについても　検討中である。

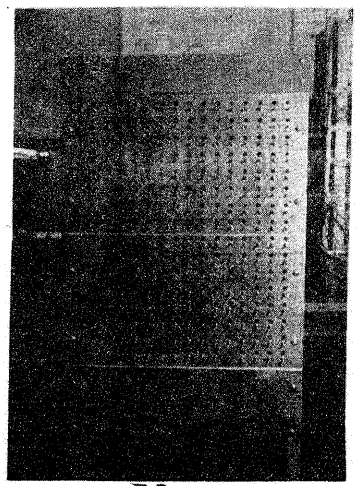


写真1

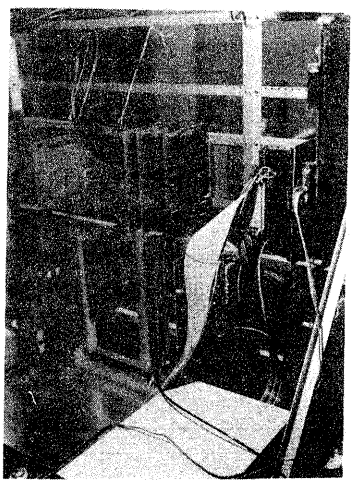


写真2（AKOUSTの一部）

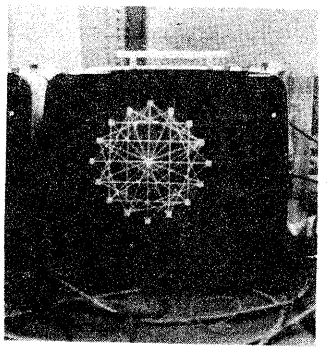


写真3

† この入力装置は漢字の1文字入力装置としても使用できる。

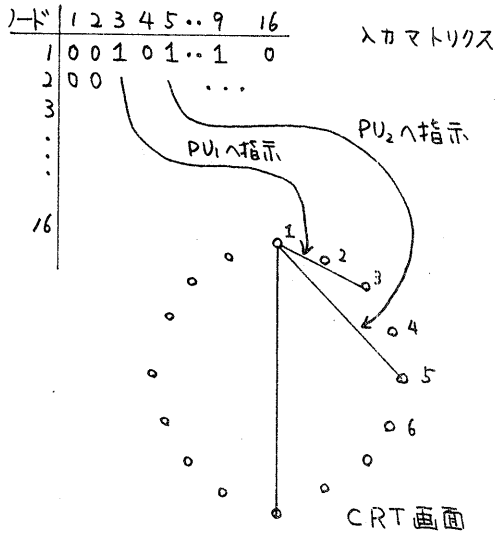


図5. 並列画像表示

4. 並列処理の効果について

従来，並列化に適した問題としては行列演算がしばしばとりあげられており，実際に並列処理システムが作られた例もある[4]。

一方，我々はオーバーヘッド軽減の為にはハードウェア化にあるとの見方から，ハードウェア化による並列処理の促進を進めている[6]~[8]。

メモリのコンテンションが並列処理プロセッサ数の上限になることから[5]，AKOVSTでは 時間インターリーブ形の共有メモリを用いている。(図3中で用いているアービタの回路図を付図1に示す。)

n台のプロセッサからコモンメモリへアクセスする部分だけをとり出して（最も競合が激しい場合）考えてみる。

マスタープロセッサからの割込みにより，スレーブプロセッサが一斉にコモンメモリをアクセスする場合，2節のプログラム例から，
R－A方式では図6のようになる。

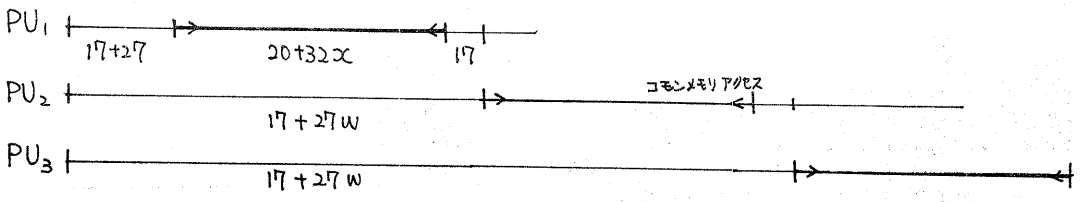


図6 R－A方式でのコモンメモリアクセス

このときの平均待ち時間は

$$T_w = \frac{n-1}{2} \cdot \frac{1}{\mu} \qquad (3)$$

ここで $1/\mu$ はスレーブプロセッサがメモリへ1バイトアクセスする時間（ステート数）で $1/\mu \simeq 32\alpha$。

(3)式から，プロセッサ数，アクセスワード数を大きくすると その積で待ち時間が大きくなる。

これに対して A－W方式では図7のようになり，平均待時間(3)式の $1/\mu$ は $3 \ll 32\alpha$ であり，R－A方式に比べ格段の効率化が望める。 2バイトめ以降のアクセスに関して 各プロセッサで独立に実行できる部分 ⓐ=29 とコモンメモリアクセス部分 ⓒ=3との比 β が大きくなれば（アクセス頻度の低化）その比の値近くまで並行アクセスが可能である。

R－A方式とA－W方式とで，n台のプロセッサが同時にコモンメモリへ1Byteのアクセスを試みた場合の平均転送時間をグラフ1とグラフ2に示す。

実際に，3節の入力マトリクスの表示では，親プロセッサがコマンドバッファへ線分1本につき3Byteの指示を置き，スレーブプロセッサに割込みをかけて実行させ，結果をコモンメモリ中の各自のバッファに書き込む。 それらを親プロセッサが収集して表示する。(図8)。

その実行速度を比較したものを図9に示す。

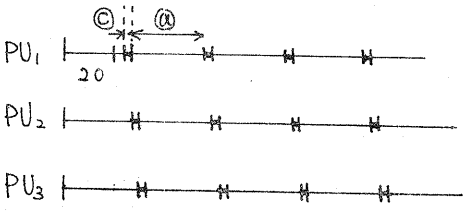
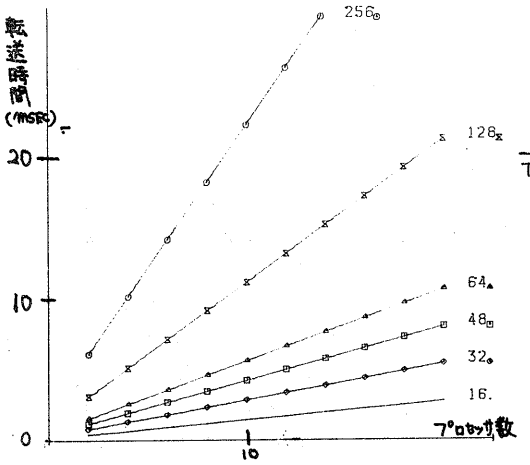
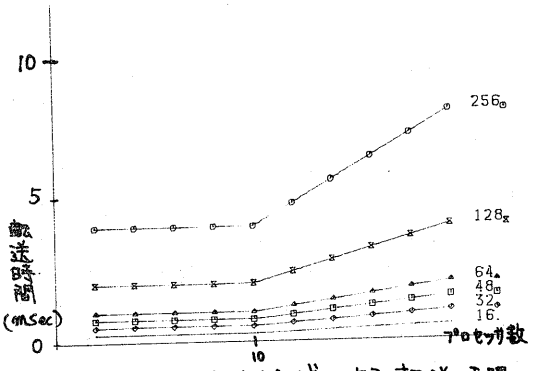


図7. A－W方式でのコモンメモリアクセス



グラフ1. R-A方式による転送時間



グラフ2 A-W方式による転送時間

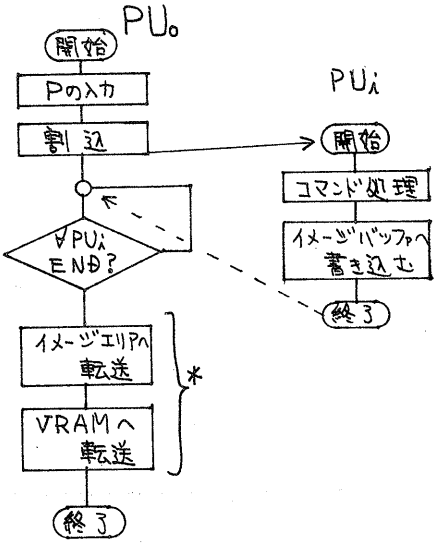


図8 並列画像表示アルゴリズム

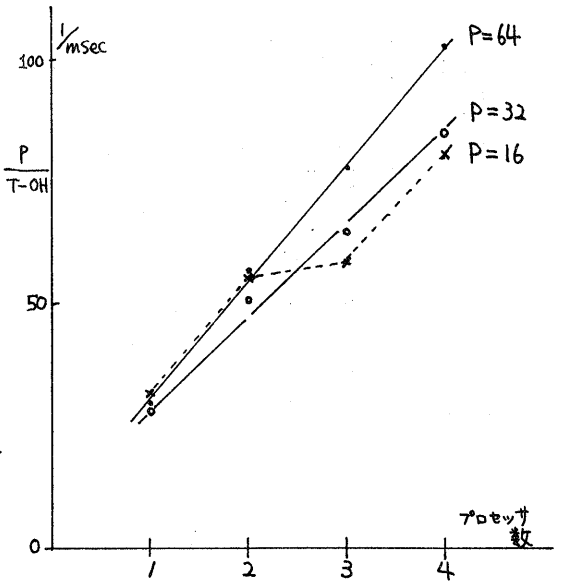


図9 並列化による処理速度の向上

この図9は，仕事量（P：線分の本数）をマスタープロセッサが転送している時間（図8の＊部）を差し引いた処理時間で割った値（処理速度）をプロットした。

この結果，並列実行により処理速度はほぼプロセッサ数に比例しているので メモリによる競合はあまりない。 またその比例定数は，スレーブプロセッサでの処理時間が短かいもの程1より小さくなっている。

5. 相互診断の方式

マルチプロセッサシステムのような分散型システムでは互いに診断しあうことによって，具合の悪いユニットを自動的に指摘するようなシステム（さらには それらのユニットを除外して全体としては正常に働くシステム）を構成する方式が考えられる(9)。

このような診断方式のモデル化は，古くから理論的には取扱われており(10)，マルチプロセッサシステムとしての扱いもすでに幾つか考察されている(11)(12)。 このような取扱いで用いられているモデルをPMCモデル（付録2参照）では図10のように，すべてのユニットの検査結果を総合して，真に故障の疑いの強いユニットを指摘するためのスーパバイザが存在し，そのスーパバイザはハードコア化されていなければならない。 このハードコアの存在は，AKO

ＶＳＴはもちろん，一般に安価なマイクロプロセッサおよびメモリチップを用いるシステムでは適当な仮定とはいえない。

そこで，ＡＫＯＶＳＴではスーパバイザの存在を仮定しない診断方式をとる。

さらに，個々の診断の方式はＰＭＣモデルより修正を加えたVancura & Kime のモデル（ＶＫモデル）[3] を用いる。　ＶＫモデルでは，複数個のユニットが協力して他のユニットを診断することも許される。

スーパバイザの存在を仮定しない診断方式をとるため，逆に次のような仮定を置く。

＜仮定Ｃ＞　正常なユニット群は強連結である。

この仮定は，ユニット自体のハードコア化を計る代わりに，ユニットの集合としてのハードコア化を考えるべく用いられた仮定で，現実のシステムを考える上で適切なものと思われる（図１１参照）。

次に，故障するユニット数については，通常ＰＭＣモデルで行なわれているよう高々ｔ個という仮定もおく。　従って一般には

$$|\bar{F}| \gg t$$

（ただし $\bar{F}$ は正常なユニットの集合で $|\ |$ は元の数をあらわす。）という仮定のもとで議論を進めていく。

故障ユニットが存在したら，そのユニットからの検査結果は信頼ができない。　したがって＜仮定Ｃ＞が成立するためには，

Ｐ１．強連結な診断グラフ　Ｇ＝（Ｖ，Ｅ）より，故障ユニットの集合Ｆに相当するノードを除去したとき†，Ｇはやはり強連結でなければならない。

Ｐ１に関しての理論的問題については別の機会にゆずる。

†　同時に，それらに付随する枝も除去する。

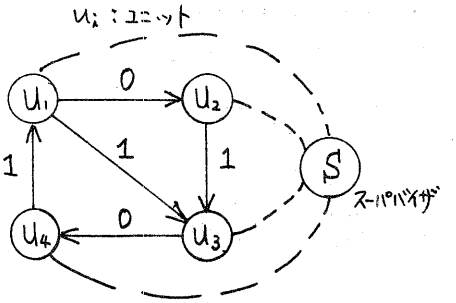


図１０　分散形計算機システムの
ＰＭＣモデルによる診断

（注　ユニットの結合の仕方は　例として挙げただけで，ＡＫＯＶＳＴとは関係ない。）

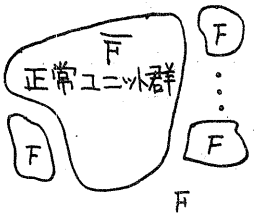


図１１＜仮定Ｃ＞の説明図

６．診断システムプログラム

５節で述べた診断方式をＡＫＯＶＳＴでは次のようにして実現している。

１）診断グラフは　図１２のようにホイール（wheel）である。

ここで，wheel とした理由は次のとおりである。

ＡＫＯＶＳＴは先に述べたように，全ユニット数が６｛処理プロセッサ４，スーパバイザ（ただし，ハードコアではない）１，通信用共有メモリ１｝であり，本来は　図１で示したシステムであるから，星（star）状の結合しかない（図１３）。　starが信頼性に欠けることは周知の通りである。

高々１個のユニットの故障に対しても，＜仮定Ｃ＞が成立する結合の例としてループ（loop）状の結合が挙げられるが，プロセッサ間通信の速度に問題がある。　ＡＫＯＶＳＴは前述のように，プロセッサ間通信のハードウェアに特長があり，図１３の結合は本来の並列処理の目的のために必要である。　そこで　ループの形

状を保ったままで信頼性をあげる形状に修正したのが図12のホイールである。

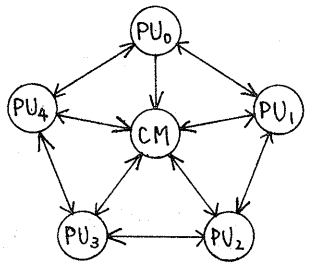


図12 AKOVSTにおける診断グラフ

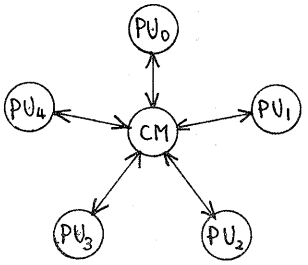


図13 AKOVST本来のプロセッサ間通信形態

P2. ホイールボネットワークでは高々2個のユニットが故障してもく仮定<>は成立つ。

このような性質をふまえて，AKOVSTでは 次のようなアルゴリズムに基づく自動診断を行うシステムプログラムを作成した。

2) 診断アルゴリズム

phase 0 自己診断
unit i ($i=0,1,\cdots,4$) は自分自身をテストする。異常があればユニット交換。

phase 1 故障ユニット数を高々1個と仮定。
[step 1]
各ユニットから出発して，時計方向に一巡する向きにテストを行う。
例えば，unit 0 からは
$PU_0 \to PU_1 \to \cdots\cdots \to PU_4 \to PU_0$
の順にテストメッセージを送る。
このようなテストをループテストと呼ぶ。
正常ならば step 2 (a)へ，
異常があれば step 2 (b)へ。

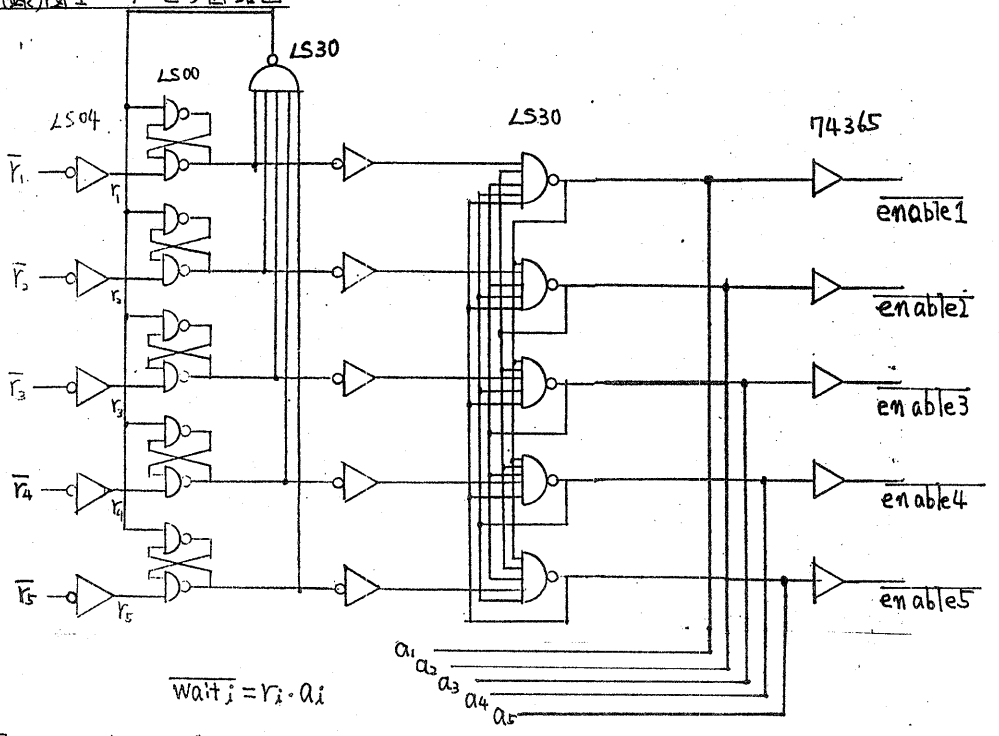
[step 2]
$PU_i \to CM \to PU_j \to CM \to PU_i$ の順にメッセージを送るテストを T_{ij} と記す。
(a) すべての T_{ij} が正常ならば
step 3 (a-0)へ。
異常があれば，step 3 (a-1)へ。
(b) 辞書式順序に T_{ij} を行い，異常が見つかった時点で，
b-1. T_{xj} (x は i 以外のすべて) を行いすべて異常ならば，step 3 (b-1)へ。
さもなくば次へ。
b-2. T_{xi} (x は i 以外のすべて) を行いすべて異常ならば，step 3 (b-2)へ。
さもなくば，phase 2へ。

[step 3]
(a-0) すべてのユニットが正常である旨のメッセージをループ上で1巡させ，同意されればすべて正常。さもなくば，phase 2へ。
(a-1) すべての PU_i が CM をテストして結果を一巡させる。すべてが，CM を異常と判定すれば CM は異常。さもなくば phase 2へ。
(b-1) PU_j が異常である旨を PU_j を除き，CM を通るループで一巡させ，同意されれば PU_j は異常。さもなくば phase 2へ。
(b-2) PU_i が異常である旨を PU_i を除き，CM を通るループで一巡させ，同意されれば PU_i は異常。さもなくば phase 2へ。

phase 2 故障ユニット数を2個として同様の手法を展開。(省略)

なお，現在のAKOVSTでは，図12の診断の方向のうち，ループは時計方向のみしかインプリメントされていないため，単一ユニット故障しか検知できない。しかし，スーパバイザを仮定しない一般的な分散形システムにおける相互診断法の一つを与えており，今後，さらに詳しい検討を行う予定である。

付(録)図1　アービタ回路図



付録2　PMCモデル

まず Preparata, Metze & Chien のモデル（PMCモデルと略す）を述べる。

分散形計算機システムにおける各ユニットを節点に，2つのユニット間の検査を有向枝に対応させて得られる有向グラフ（これをネットワークと呼ぶ）の上で，故障ユニットの指摘を行うことを『診断問題』と呼ぶ。

各ユニット U_i は

(1) 相異なる U_j を検査可能，または

(2) 相異なる U_j によって検査されることが可能

とする。ただし，(1)，(2)とも成立してもよい。U_i が U_j を「故障」と判定すれば1，「正常」と判定すれば0を，U_i から U_j へ向う枝の重みとして与える。

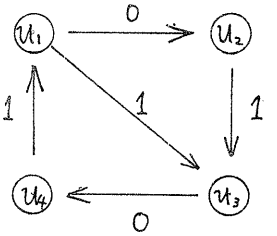
故障ユニットの集合 F には2つの条件

[条件1]　$U_i \in \overline{F} = V - F$，$U_j \in F$ なるすべての $(U_i, U_j) \in E$ に対して $e_{ij} = 1$

[条件2]　U_i，$U_j \in \overline{F}$ なるすべての $(U_i, U_j) \in E$ に対して $e_{ij} = 0$

が課せられ，この条件を満足する $F \subseteq V$ を「無矛盾故障集合」という。

付図αのようなツンドロームをもつネットワークで表わされる分散形計算機システムを考えてみよう。無矛盾故障集合 F をすべて列挙すると $\{U_1, U_2, U_3, U_4\}$，$\{U_1, U_3, U_4\}$，$\{U_1, U_2, U_3\}$，$\{U_1, U_2\}$，$\{U_1, U_3\}$，$\{U_3, U_4\}$ となる。通常，故障の発生はユニット数が多くなるにつれ，同時に起る頻度は減少するという経験的な事実から，真に故障の疑いの強いユニットの集合として，最少個数の元からなる無矛盾故障集合を その候補としてあげる。



付図α　ネットワークの例

文献

(1) 阿江，大崎，Vuong：“小規模マルチマイクロプロセッサシステムの一方式”，信学会電算機研資 EC78-35（1978）

(2) 阿江，高橋，川口，神川：“インターリーブ形共有メモリをもつマルチプロセッサシステム AKOVST”，信学会情報・システム部門全大 381（1979）

(3) たとえば，情報処理：特集「分散処理」，20，4（1979）

(4) たとえば，市川，坂村，諸陽，相磯：“連想プロセッサ ARES”，信学論(D)，J61-D，10，pp. 743-750（1978）

(5) 高橋，吉村：“並列プログラム処理実験装置”，情報処理，20，4，pp. 319-322（1979）

(6) 阿江，高橋，片山，伊藤：“イメージプロセシング向小規模マルチプロセッサシステム MIP-1”，電気4学会中支連大 62310（1979）

(7) 阿江，高橋，千葉：“ディスプレイ端末としての MIP-1 とその応用”，同上 62311（1979）

(8) 阿江，高橋，千葉，松本：“画像処理マルチプロセッサシステムのデータフロー化について”，信学会全大 1980.3 発表

(9) C. Bellon et G. Saucier：“On line test modelling in non redundent distributed systems”，Proc. FTCS-7，p. 94（1977）

(10) F. P. Preparata，G. Metze and R. T. Chien：“On the assignment problem of diagnosis systems”，IEEE Trans. Electronic Computer，EC-16，6，p. 848（Dec. 1967）

(11) 阿江，大崎：“自己診断システムにおけるネットワーク構造と計算複雑さ”，信学論(D)，J62-D，2，pp. 126-132（1979）

(12) 阿江，ブンバンクイ：“分散形計算機システムの診断理論 ―構造と検査パターンからみた診断複雑さ―”，信学会オートマトンと言語研資 AL79-33（1979）

(13) R. P. Vancura and C. R. Kime：“On numerical bounds in diagnosable systems”，Proc. FTCS-2，pp. 148-153（1972）

謝辞 システムの製作に当っては，計算機研の神川正博（現中国計算センター），川口秀樹（現日立造船情報システム），新谷哲生（現安立電気）および電子回路研の輝平益重各氏の援助を受けており，厚く御礼申上げる。また，本研究の一部は文部省科学研究費の補助によるものである。