計算機複合体 MICS-II のシステム評価†

大森健児†† 小池誠彦††
山崎竹視†† 大宮哲夫††

計算機複合体の技術発展のためには，その性能特性を表わす式を得ることと，実システムによる性能評価が重要である．

本論文では，資源獲得のためのプロセッサ間の競合等による性能低下が存在しないときの計算機複合体の性能と価格の関係式を求めた．この関係式によれば，プロセッサ台数がある数を越えない範囲において，計算機複合体が GROSCH の法則に従う現行の計算機より優位にあるといえる．

次に実システムによる性能低下を，ハードウェア構成とシステム制御に特徴のある計算機複合体 MICS-II を用いて求めた．

第1に，MICS-II 構成のために生じるメモリバス獲得による性能低下，獲得優先度の違いによる性能低下率の差，ページフォールト処理による性能低下にについて実測を行った．その結果，Ravindran のシミュレーション結果とよく一致すること，アービタトリーの性質に著しく影響されていること，ページ追出しアルゴリズムの効果の有無がメインモリの容量に影響されること等が各々判明した．

第2に，オセロゲームをするプログラムを作成し，実際のプログラムでの性能低下を実測した．このプログラムによれば性能低下はほとんど生ぜず，計算機複合体の本来の目的「プロセッサ台数を n 倍にすればシステムの性能は n 倍になる」が達成されていることが判明した．

1. 序　論

LSI 技術の急激な進歩は，計算機設計技術に大きな影響を与え，計算機複合体の出現を引き起した．計算機複合体の設計は 1970 年初頭に始まり，近年実験機が出現し始めている．計算機複合体は，(1)経済性，(2)信頼性，(3)拡張性，(4)量産性等において従来の計算機システムより優れているといわれているが，実用機として利用されるためには，応用分野の明確化とともに，定量的な裏付けが必要である．

計算機複合体の性質を最も顕著に表わす設問として，“計算機台数の増加に対して性能がどのように増加するか”というのがある．この設問に対して，従来とられた手法は，各プロセッサがランダムにメインメモリへアクセスした時に，性能がどのように変化するかということを計算またはシミュレーションで解くことであった．

しかし，現実のシステムにおいては，プログラムの性質，OS のオーバヘッドの影響が大であり，従来の方法は必ずしも計算機複合体の性質を明確に表現するには適切でない．

そこで，我々は，1977 年5月より稼動を開始した MICS-II を用いて，計算機複合体の性質を追求した．

2. 計算機複合体の性能特性

GROSCH の法則——価格を n 倍にすれば，n^2 倍の性能を有する計算機が入手できる——で性格づけられている従来の計算機を打破できるかという命題が，計算機複合体に対して課せられている．一般に計算機複合体においては，システムオーバヘッドやアクセス時の衝突等によって起る性能低下が無視できるものであれば，CPU 数を n 倍にしたときその性能は n 倍に増加する．

CPU 数の増加が価格に及ぼす影響を一般化することは相当に困難であるが，ここでは次のように仮定して，性能と価格の関係を求めた．その仮定は，1) 計算機複合体においては，CPU 数を n 倍にしたとき，イ) n 倍増加が必要な一次線形型と，ロ) n^α 倍増加が必要な非一次線形型の2通りの装置がある．2) CPU 数を n 倍にしたときの価格は，その時のシステムを構成する一次線形型と非一次線形型装置の価格の単純和である．

CPU 一台のときのシステム価格を c，その時システム価格にしめる一次線形型装置の価格の割合を q とする．これより，CPU 数を n 台にしたときのシステム価格を求めると次のようになる．

† System Evaluation for a Multi Processor System MICS-II by KENJI OHMORI, NOBUHIKO KOIKE, TAKEMI YAMAZAKI, and TETSUO OHMIYA (Central Research Laboratories, Nippon Electric Co., Ltd.).

†† 日本電気(株)中央研究所

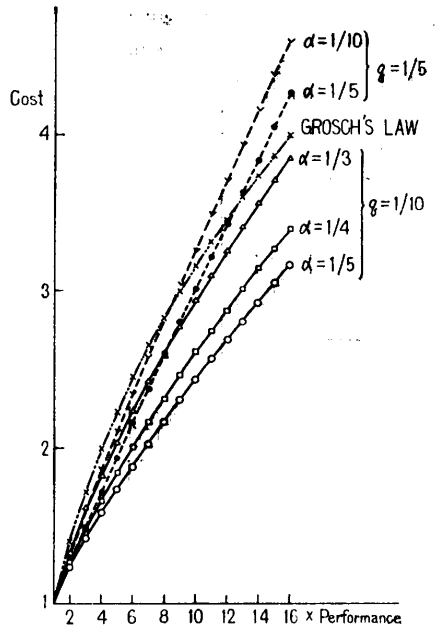

図 1 システム価格比
Fig. 1 Cost-performance relation.

$$c\{nq+n^{\alpha}(1-q)\} \qquad (1)$$

そこで，計算機複合体では CPU 数を n 倍にすればその性能は n 倍になると仮定して，式(1)をもとに性能と価格の関係を求めると**図 1** のようになる．

計算機複合体においては，プロセッサ，ローカルメモリ等が一次線形型に属し，メインメモリ，I/O 装置，筐体等の共有資源装置が非一次線形型に属す．そのため，一般に α は 1 より小，また，q は 1 より相当に小である．そこで，

$$2^{\alpha}<\frac{\sqrt{2}-2q}{1-q} \qquad (2)$$

を満足する (α, q) の場合，CPU 数がある台数より少ない部分では，計算機複合体の方が GROSCH の法則に従う将来の計算機よりも性能価格比において勝るといえる．**表 1** に計算機複合体が勝るときの最大の CPU 数を示す．

MICS-II では，素子及び装置技術の現状を考慮して，最大構成を 8 台とし，α を 1/3，q を 1/8 程度になるように目標をおいて設計を行った．

表 1 最大プロセッサ数
Table 1 Maximum processor number.

α \ q	1/5	1/8	1/10
1/3	2	8	20
1/4	5	23	41
1/5	8	29	52
1/10	12	41	69

さらに構成規模が大となったときでも，CPU の実行効率が低下しないように，設計段階でいくつかの考慮を払った．

その一つは，システムオーバヘッドを低く押さえるためプログラム実行部分とシステム管理部分とを分離したことであり，また他の一つは，アクセス時の衝突を減少させるために，ローカルメモリを設けたことである．そこで，この効果をみるために，MICS-II の概略をまず説明し，次に実際のプログラムによる測定結果を示す．

3. システム構成

MICS-II は，**図 2** に示すように現在 6 台のプロセッシングモジュール (PM) を中核として構成されている．これら PM は，コントロールバス，メモリバス，I/O バスの 3 種類の共通バスによって結合され，PM 相互間の通信，メインメモリ，入出力装置等の共有資源へのアクセス等によって，有機的なつながりをもつ．

各 PM は，ユーザプログラムが実装されるユーザモジュール (UM) と，コントロールプログラムが実装されるコントロールモジュール (CM) で構成される．

UM は，ユーザプログラムを実行するユーザプロセッサ (UP)，ユーザプログラム全体，またはその一部を記憶するローカルメモリ (LMM)，UP の論理アドレスをメインメモリ上の物理アドレスに変換するアドレストランスレータ (AT)，UP の論理デバイスアドレスを入出力装置に与えられている物理デバイスアドレスに変換する I/O ポート (IOP) で構成される．

CM は，コントロールプログラムを実行するマイクロプロセッサ (μP)，コントロールプログラムを記憶するコントロールメモリ (CMM)，UP の制御を行うコンソールモジュール (CON)，CM と UM 間でのデータ交換を行うインタプロセッサポート (IPP)，UM に与えられたメモリ空間にアクセスするバスウィンドモジュール (BWM)，PM 間での通信を行うコミュニケーションモジュール (COM) で構成される．

PM の台数別に MICS-II システムにおける一次線形型装置と非一次線形型装置の価格を，現在市販されているプロセッサ，I/O 装置等の価格をもとに推定すると，**図 3** のようになる．これより，α, q とも設計時に設定した値の 1/3, 1/8 になっていることがわかる．

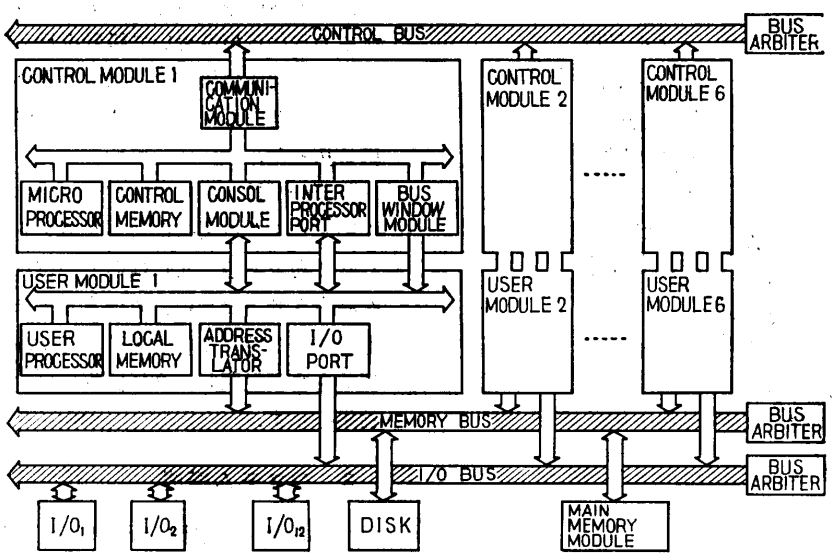

図 2 MICS-II ブロックダイアグラム
Fig. 2 MICS-II blockdiagram.

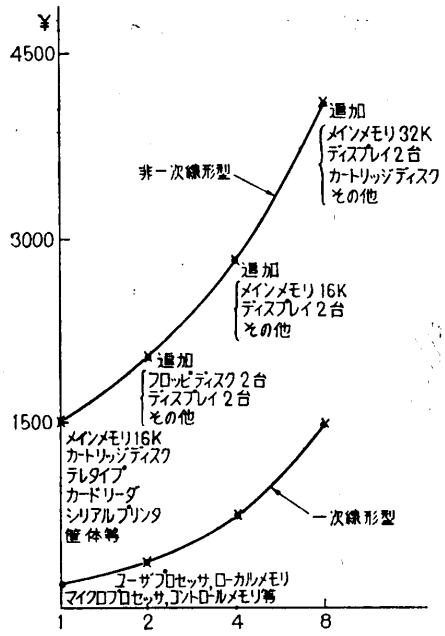

図 3 システム価格
Fig. 3 System cost.

4. 性能低下要因

MICS-II において，性能を低下させる要因は2つある．

その一つは，共通バス方式による性能低下である．各 PM は，メインメモリ，入出力装置を，各々メモリバス，I/O バスを介して共有している．そのため，バスの使用権をめぐって相互に競争しあう．その結果，競争に負けた PM は，次の機会まで共有資源へのアクセスを待たされ，その分だけ処理速度が低下する．

他の一つは，仮想メモリ方式による性能低下である．MICS-II では，メインメモリの動的割当てによる資源の有効利用，メモリ空間の共有による並列処理の実現，独立したメモリ空間を各 PM に提供することによる保護の確立等に重点を置いて仮想メモリ方式を実現した．そのために，UP が，メインメモリの割当てを受けていないページにアクセスした場合には，メモリ割当て，ページスワップ等のページフォールト処理が起こり，そのため，UP の処理速度が低下する．

4.1 共通バス方式による性能低下

MICS-II には，UP の性能低下に係る共通バスとして，メモリバス，I/O バスの2種類ある．しかし I/O バスでは，UP と入出力装置間でのデータ転送の速度，頻度とも低いので，性能低下はほとんど起こらない．そこで，ここではメモリバスに起因する性能低下について述べる．

これは，次のような仮定をすると，RAVINDRAN 等[1]の方法で図4のように求めることができる．その仮定は，1）各命令の実行に要する時間は同一で，各 UP は同時に命令の実行を開始し終了する．2）各 UP のメインメモリへのアクセス確率 B は，過去の状況によらず常に一定である．

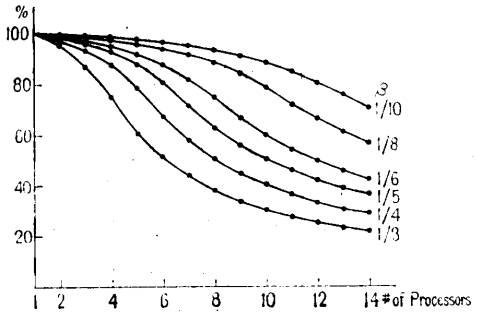

図 4 共通バスに起因する性能低下
Fig. 4 Performance degradation caused by common bus conflicts.

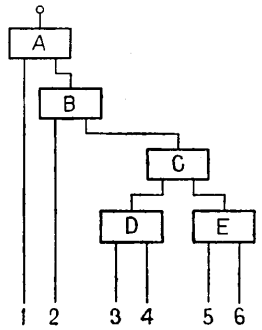

図 5 バス制御のためのアービタトリー
Fig. 5 Arbiter tree for bus control.

4.2 バスの制御構造

MICS-II のバス使用権は，図5のようにアービタを樹枝状に接続した回路によって制御されている．個個のアービタは，競争する2者の間を調停する．すなわち，2者の間で使用権をめぐって争いが生じたとき，アービタは早く要求を出した方に，その要求が続いている間承認を与える．そして後から要求を出した方には，前者の要求が取下げられたとき承認を与える．

アービタを樹枝状に接続した回路においては，例えば次のように制御される．今，5と6がほぼ同時に要求を出したとする．これは，アービタEで調停され，その結果，5の方が早く要求を出したように調停されたとする．さらに，1, 2, 3, 4 のいずれもが要求を出していないとすると，5の要求は，アービタEよりアービタCとBを経てアービタAに達し，ここで承認が与えられ，この承認が逆にアービタBとCを経てアービタEに至り最後に5に伝達される．そこで，4から新たな要求が発生したとしよう．この要求はアービタDよりアービタCに伝達されるが，5に対して承認が与えられているため，この時点ではこの要求に対して承認は与えられない．しばらくして，5が要求を取下げたとすると，アービタEでは，6が要求を出している場合でも，アービタCへの要求を一瞬取下げその後で6の要求をアービタCへ伝達する．そのため，5の要求が取下げられたとき，アービタCでは，4からの要求の方が6からのそれより早く出されたように認識され，次の承認は前述した経路をへて4に与えられる．

従って，アービタによるバス制御方式では，各 UP に対して使用権に優先席をもたせることができる．図5の接続では，各々が要求を頻繁に発した場合には，3, 4, 5, 6 が一回ずつ承認を得る間に，1は8回の承認を，2は4回の承認を得る．

4.3 メモリバスによる性能低下

実際のシステムでのバスの衝突による影響を調べるために，各 UP に同一のプログラムを実装し，それを同時に実行させることによって，UP の処理時間の変化を調べた．

同時に走行する UP の台数が自身を含めて n 台のときの UP_k の処理時間を $T_{n,k}(\beta)$，また他に同時に走行する UP が存在しないときの処理時間を $T_1(\beta)$ とし，同時に走行する UP の台数が n 台のときの平均性能低下率 D_n，最大の性能低下率 $D_{max,n}$，最小の性能低下率 $D_{min,n}$ を以下により求めこれを図6に表わした．

$$D_n=\frac{\sum_{k=1}^{n}(T_1(\beta)/T_{n,k}(\beta))}{n} \tag{3}$$

$$D_{max,n}=\mathrm{MAX}\left(\frac{T_1(\beta)}{T_{n,1}(\beta)},\ \frac{T_1(\beta)}{T_{n,2}(\beta)},\ \cdots,\ \frac{T_1(\beta)}{T_{n,n}(\beta)}\right) \tag{4}$$

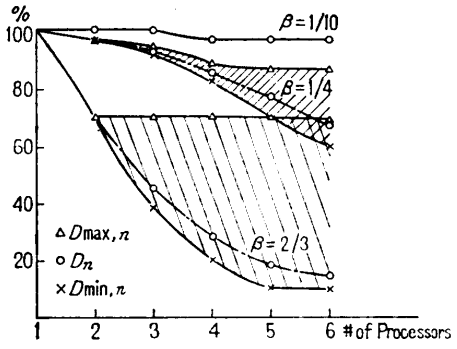

図 6 メモリバスに起因する性能低下
Fig. 6 Performance degradation caused by M-BUS conflicts.

$$D_{\min,n}=\mathrm{MIN}\left(\frac{T_1(\beta)}{T_{n,1}(\beta)},\ \frac{T_1(\beta)}{T_{n,2}(\beta)},\ \cdots,\ \frac{T_1(\beta)}{T_{n,n}(\beta)}\right) \quad (5)$$

上式においてβはメインメモリへのアクセス確率であるが，これは先の仮定から，βは一台の UP のみが動作している時それがメモリバスを占有する割合とみなすことができる．そこで，実験ではバスの占有率を計測しこれをβとした．図6には代表的な値のものを選んでのせたがプログラムの詳細は以下の通りである．

第1の実験は，命令がローカルメモリに，データがメインメモリに置かれていて，命令のほとんどがデータの参照を伴うプログラムを用いて行った．このプログラムのバス占有率は 1/4 であり，MICS-II のプログラムとしてはメインメモリ参照率が高い部類に属する．この場合，平均の性能低下率は，ほとんど RAVINDRAN 等の方法による結果と一致した．また，バスの制御構造による優先度のために，UP 間で性能低下率に差が生じ始めている．これを顕著に表わしたものが次の実験である．

第2の実験は，命令，データが共にメインメモリ上に置かれているプログラムを用いて行った．この場合，使用されている UP の総数が6台のとき，UP1 と UP6 の性能比が約7倍であり，理論上の最大値8に近づいているのが特徴である．

また，MICS-II のプログラムの多くは，バス占有率が 1/10 前後と考えられる．そこでバス占有率が 1/10 程度となるプログラムを選びだして実験を行ったが，この場合には性能低下はほとんど観察されなかった．

4.4 仮想メモリ方式による性能低下

MICS-II では．システムコントロールは，ユーザプログラムの実行を妨げることなく独立した μP で行われる．そのため，システムコントロールが UP の処理速度に影響を及ぼすことはほとんどないが，唯一の例外は仮想メモリ方式である．

仮想メモリ方式においては，UP が未割当てページへアクセスした時ページフォールトが生じる．この時メモリ割当て，ページスワップのために UP が停止させられる．そのため，UP の処理速度が低下することが懸念される．しかし，MICS-II の仮想メモリ方式の目的は前述した通りであり，そのため，一般にページフォールトの生じる確率はプログラムロード時に高く，プログラム実行時には極めて低いと当初予想された．そこで，ページフォールト処理が UP の処理速度を著しく低下させることはないとの推定に基づき，MICS-II の仮想メモリ方式は，経済性に重点を置いて低速な二次メモリの使用と，簡単な追出しアルゴリズムの導入によって実現された．

実現されたシステムにおいても所期の傾向を示し，ページフォールト発生に伴うシステムオーバヘッドはプログラムロードの中に隠れてしまうことが確認された．しかし，メインメモリが故障した場合にはシステムオーバヘッドが無視できないことも判明した．すなわち，故障時の縮退によって，使用可能なメインメモリの容量が小さくなるため，プログラム実行時のページフォールト処理が増し，プログラムの実行時間に影響を及ぼすことがわかった．

4.5 ページフォールト処理

ページフォールト処理は，追出しページの決定とページのスワッピングから成立っている．

MICS-II におけるページ追出しアルゴリズムはワーキングセットである．すなわち，最近τ時間の間にアクセスされなかったメインメモリのブロックに実装されているページ群を，追出しページの候補とし，その中の一つを次のように選ぶ．今，追出しページに決定されたページが，実際に追出されたか，またはアクセスされたかによって追出しページでなくなったとすると，次に追出しページに決定されるのは，ページの低い側で，今まで追出しページに決定されていたページにもっとも隣接している追出しページの候補である．

また，ページのスワッピングは図7のように処理される．今，PM_j 内の UP でページフォールトが生じたとすると，ページフォールト通知の通信がリソースを管理している PM_k に対して行われる．そして，追出しページが決定されると，このページを利用している PM に対して PM_k からページアンロードの通信が行われる．その後，PM_k でページのスワッピングが行われてから，PM_k から PM_j に対してページロー

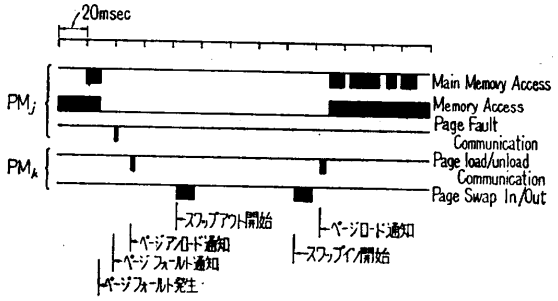

図7 ページフォールト処理
Fig. 7 Page fault process.

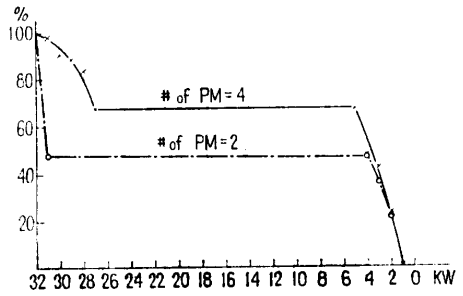

図 8 ページフォールトに起因する性能低下
Fig. 8 Performance degradation caused by page fault.

ド通知の通信が行われる．これにより，一連のページフォールト処理が終了する．この間に要する時間は，測定の結果，平均 160 msec であった．

4.6 故障時の性能低下

メインメモリの一部が故障したときに生じる性能低下を測定したものが**図 8** である．

第一の実験は，PM 4 台を用い，各々にメインメモリ 8 ページを順番にアクセスするプログラムを実装し，そして，メインメモリが故障したと想定して使用可能なメモリ容量を減少させて行った．この結果，次のことが判明した．使用可能なメモリ容量が 28 kW から 31 kW の場合は，メモリアクセスが次のページへ推移したとき，ある確率でページフォールトが起こる．その確率は，使用可能なメモリ容量が小さい程大きい．そのため，メモリ容量の減少に伴って性能は低下する．また，使用可能なメモリ容量が，5 kW から 30 kW の場合は，メモリアクセスが次のページへ推移したとき，必ずページフォールトが起こる．そのため，メモリ容量が減少しても，性能はそれ以上低下しない．さらに，使用可能なメモリ容量が 1 kW から 4 kW の場合は，メモリアクセスが次のページへ推移したとき，前の場合と同様に必ずページフォールトが起こるが，この場合，追出すべきページが存在しないため，使用中のページを追出す．そのため，性能低下が著しい．

もう一つの実験は，PM も故障したと想定し，PM の台数を減らして行った．すなわち，PM 2 台を用い，各々にメインメモリページを順番にアクセスするプログラムを実装し，メインメモリを前回と同様に減少させて行った．この場合は，ページフォールトが確率的に起こる部分がなく，すぐに性能低下率が一定の所に落着くのが特徴である．

5. 並列処理

以上，MICS-II の性能特性について個々に述べてきたが，ここでは，MICS-II の応用分野の中から典型的なプログラムを選び出し，全般的な性能特性について述べる．

既に述べてきたように，MICS-II はメインメモリを各 UP で共有する密結合の計算機複合体である．計算機複合体は，共通のデータベースの上で同時に複数の計算機が走行できるという特徴を有するが，密結合では，特に負荷分散，機能分割による並列処理が可能である．しかし，計算機複合体による並列処理は，ILLIAC-IV 等の専用機にみられるそれとは異なる．すなわち，専用機の場合は命令単位で行われるのに対して，計算機複合体の場合はプロセスまたはタスク単位で行われる．

このため，並列処理への計算機複合体の応用として人工知能，パターン認識等の分野を挙げることができる．

そこで，ここでは人工知能の一分野であるゲームの中からオセロゲームを選びだし，これを計算機に行わせることを考えた．

5.1 ミニマックス法

計算機でゲームを行う方法として，ミニマックス法が一般的である．ミニマックス法は，次に打てるすべての打手の中で最大の評価値を与えるものを，計算機の次の打手とするものである．次の打手の評価値は，その打手に続いて打てるすべての打手の評価値を求め，その中で最小の値（相手の打手を決定するときは最小，自分の打手を決定するときは最大）を次の打手の評価値とすることによって求められる．このため，評価値を得るために，評価関数に基づいて評価計算が行われる深度 n の打手まで，次から次へと打手が求められる．この様子を**図 9** に示す．図では 2 手目で評価関数を用いて評価計算を行っている．

このように，ミニマックス法は，回帰的呼出しを巧みに利用した方法であるが，一方で並列処理を不可能にしている．

5.2 プロセスの生成消滅法

ゲームをするプログラムを並列処理によって実行可能とするために，プロセスの生成，消滅による方法を考えた．ゲームをするプログラムは，前述の例題からもわかるように，次の 3 種類のプロセスの組合せからなりたっている．それは，1）次の打手を探す，2）

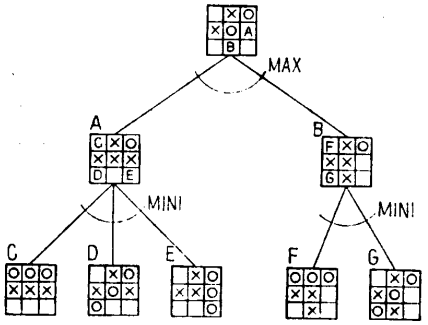

図 9 ミニマックス法
Fig. 9 MINI MAX METHOD.

評価値を求める，3）ミニマックス計算を行う，である．

ゲームをするプログラムは，まず与えられた局面に対して最適な打手を採すために，プロセス“次の打手を探す”で始まる．このプロセスは，与えられた局面に対して，まず次に打てるすべての打手を探す．そして，これら打手の中でもっとも良いものを選び出すためのプロセス“ミニマックス計算を行う”を待機中のプロセスとして生成する．次に，各打手に対して新たな局面を作り，それに対してプロセス“次の打手を探す”またはプロセス“評価値を求める”を実行可能なプロセスとして生成して自身を消滅する．

MICS-II の各 UP は，実行可能なプロセスを取り出してきては，そのプロセスを実行する．従って，最初のプロセスが実行されると，実行可能なプロセスが複数個生成されるため，各 UP がプロセスを分担することによって並列処理が実現される．

プロセス“次の打手を探す”を取り出した UP は，前述と同様な手順を行う．また，プロセス“評価値を求める”を取り出した UP は，与えられた評価関数によって評価値を求める．そして，このプロセスと同時に生成されたプロセス“ミニマックス計算を行う”を実行可能なプロセスにしそれに評価値を渡して，現在実行中のプロセスを消滅する．そして，UP は実行可能なプロセスを取り出し，それを実行する．

プロセス“ミニマックス計算を行う”を取り出した UP は，今まで渡された評価値の中で，最大または最小のものを探す．そして，まだ渡されるべき評価値があるときは，実行中のプロセスを待機中にし，また，すべての評価値の比較が済んだときは，このプロセスを生成したプロセスと同時に生成されたプロセス“ミニマックス計算を行う”を実行可能なプロセスにしそれに評価値を渡して，現在実行中のプロセスを消滅する．そして，UP は実行可能なプロセスを取り出しそれを実行する．なお，プロセス“ミニマックス計算を行う”が計算機が次に打つべき最適な打手を得たときは，この手を打って相手が打つのを待つ．

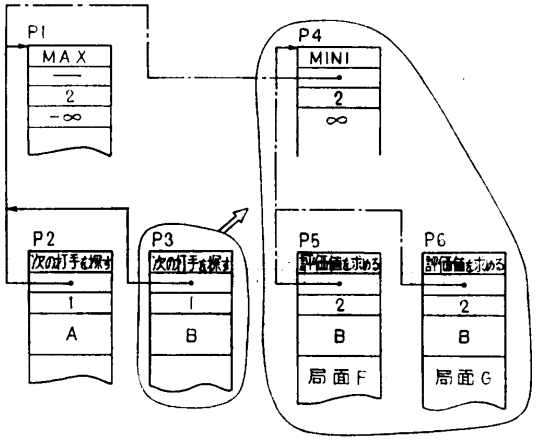

図 10 プロセスコントロールブロックの推移
Fig. 10 Transition of process control blocks.

図 10 に，このプログラムの制御ブロックの変化を示す．図 9 の例題で，局面 A と B を求めるところまで終了したと仮定し，局面 B に対してのプロセス“次の打手を探す”（P 3）が実行されたときの変化を示したものである．この例で，P5，P6 で得られた評価値が P4 に，P4 で得られた評価値が P1 に渡されるように，各々のプロセスは，ポインタによって関係づけられている．

5.3 処理時間

実際のプログラムでは，プロセスの制御ブロックをメインメモリ上に置き，各 UP がここから実行可能なプロセスを取り出してきては，それをローカルメモリ上で実行する方法によって，並列処理を実現した．

次に，筆者の一人が，PM 5 台を用いてオセロゲームのプログラムと闘ったときの打手と，プログラムが打手を探すために必要とした処理時間を**表 2** にあげる．また，同表に，PM 一台のときの同プログラムが必要とした処理時間をあげ，処理時間の比を示した．

この表から，このオセロゲームのプログラムでは，メモリバス上で起こった UP 間の衝突，また，10回前後生じたページフォールトの影響はほとんど表われず，計算機複合体の所期の目標の——CPU 台数が n 倍になれば，性能も同様に n 倍になる——が達成されていることがわかる．

この結果の多くは，プログラム実行部分とシステム管理部分がハードウェア的にもソフトウェア的にも分

表 2 性能評価
Table 2 Performance evaluation.

手数	Man x	Man y	MICS-II x	MICS-II y	(PM数)処理時間 (PM=1) T_1 秒	(PM=5) T_2 秒	T_1/T_2
1	5	6	6	4	6.9	1.6	2.3
2	4	3	3	4	17.4	3.9	4.7
3	3	3	3	2	34.1	7.2	4.7
4	6	3	6	2	30.5	6.4	4.8
5	7	3	6	5	60.0	12.3	4.9
6	7	4	6	6	75.3	15.2	5.0
7	7	5	8	3	103.3	20.8	〃
8	6	1	4	6	69.0	13.8	〃
9	3	5	3	6	97.6	19.6	〃
10	5	3	5	2	82.4	16.6	〃
11	4	7	4	8	190.9	38.2	〃
12	5	1	8	4	71.4	14.4	〃
13	8	6	7	6	73.0	14.7	〃
14	2	5	1	5	117.6	23.7	〃
15	2	3	1	3	124.8	24.9	〃
16	3	8	2	8	54.8	11.2	〃
17	1	4	1	6	77.2	15.5	〃
18	2	4	5	8	88.6	17.7	〃
19	4	1	5	7	63.3	12.8	4.9
20	2	6	3	7	76.3	15.5	〃
21	4	2	2	1	46.5	9.5	〃
22	6	8	7	8	55.7	11.3	〃
23	8	5	8	7	29.6	6.2	4.8
24	7	7	7	1	21.3	4.6	4.6
25	8	1	8	8	19.1	4.1	4.7
26	3	1	6	7	8.6	1.8	4.8
27	8	2	7	2	3.7	1.0	3.7
28	2	7	2	2	2.3	0.7	3.3
29	1	2	1	1	0.5	0.4	1.3
30	1	7	1	8	0.3	0.3	1.0
合計	24		40		1702.0	345.9	4.92 (98.4%)

離されているために，PM の増加がコントロールプロセッサの有効な処理の割合の増加をもたらすにもかかわらずユーザプログラムに対しては PM の増加が，それを実行するプロセッサの UP の実行効率をほとんど減少させることなくその増加をもたらしたことと，プロセス内での内部処理がほとんどローカルメモリを用いて実行されたことに起因している．

6. 結　論

以上，MICS-II をもとに計算機複合体の性質について述べてきた．

“CPU の台数を n 倍にしたときその性能を n 倍にする” という計算機複合体の目標を，一例ではあるがオセロゲームのプログラムにおいて達成することができた．これは，プログラム実行部分とシステム管理部分の分離及びローカルメモリ導入による所が大きい．他の負荷分散型の並列処理プログラムにおいても，オセロゲームのプログラムと同様な処理構造をもつため，CPU 数と性能の関係はこの目標に近い値を示すと推定されるが，今後はこの検証と共に，他の分野，例えばマルチユーザサービスシステムではどのようになるかを実験することが必要である．

また，計算機複合体が GROSCH の法則に従う従来の計算機に，ある部分で優位をしめることを示したが，さらにより大きな範囲で優位をしめるためには今後の LSI 技術の一層の発展に期待するところが多い．現在の MICS-II をみると，CPU の周囲に相当数の IC を必要としている．そのため，α 及び q の値を低下させるためには，CPU，メモリの価格を低下させるばかりでなく，これら周辺回路の価格低減にも重点を置く必要がある．そこで，少量多品種の LSI が低価格で生産できる技術の確立が強く望まれる．

謝辞 本システムの研究を行うにあたって，常に有意義な助言を与えて下さる日本電気(株)中央研究所木地部長，禰津部長に深謝します．また，ハードウェアの試作を担当して下さった三野輪氏，OS の作成をして下さった日本タイムシェア(株)出来氏に感謝します．工技院の方々にも合せて感謝します．

なお，本研究は通産省工業技術院大型プロジェクト「パターン情報処理システムの研究開発」の一環として行われているもので，MICS-II は対話型複合ターミナルプロセッサの小規模実験セットの通称である．

参考文献

1) Ravindran, V. K. and Thomas, T.: Characterization of Multiple Microprocessor Network, COMPCON, pp. 133-137 (1973).
2) Fuller, S. H.: Price/Performance Comparison of C. MMP and PDP-10, Computer Architecture, pp. 195-202 (1976).
3) Ohmori, K., Koike, N., Nezu, K., and Suzuki, S.: MICS-A Multi-Microprocessor System, IFIP pp. 98-102 (1974).
4) Ohmori, K., Koike, N., Yamazaki, T., Ohmiya, T., and Nezu, K.: MICS-II A Virtual Machine Complex Controlled by Dedicated Microprocessors, COMPCON Spring (1978).
5) 大森健児，小池誠彦，山崎竹視，大宮哲夫：マルチプロセッサシステム MICS-II による並列処理，信学会研資 EC 77-64 (1978).
6) Ohmori, K., Koike, N., Yamazaki, T., Ohmiya, T., and Nezu, K.: System Management of MICS-II-A Virtual Machine Complex, UJCC pp. 425-429 (1978).

(昭和 53 年 3 月 1 日受付)
(昭和 53 年 8 月 17 日採録)