MIMDにおけるInterconnection Networkとその制御

AN INTERCONNECTION NETWORK FOR MIMD SYSTEMS AND ITS CONTROLS

岸則政　石井繁　横山清　川上英　小原啓義
NORIMASA KISHI　SHIGERU ISHII　KIYOSHI YOKOYAMA　SUGURU KAWAKAMI　HIROYOSHI OHARA
早稲田大学理工学部
WASEDA UNIVERSITY

1. はじめに

Flynnの分類[1]によるMIMD（Multiple Instruction Stream Multiple Data Stream）型の大規模マルチプロセッサシステムは，ジョブ（タスク）の並列性の大きいプログラム（例：作戦の問題，人工知能，整数問題）に対応できるマルチプロセッサシステムとして最近脚光を浴びつつある。

本稿では，まずMIMDシステムの中で，特に大規模で実現可能であると考えられるプロセッサ群接続法の提案を行なっている。

プロセッサ群接続法に関する研究は，システム効率を高める意味からも極めて重要であり，最近，大規模マルチプロセッサの研究の一環として，かなり活発に研究が成されている。なかでも，L.D.Wittieの提案である2-portマイクロプロセッサを10^6個結合するMega-Micro接続法[2]やArnold Page等の提案であるプロセッサ群の木構造を可変にできる接続法[3]等が興味あるものである。その他にも，シーメンス社SMS 201，カーネギー・メロン大学の$C.mmp$，スタンフォード大学のMINERVA，ニューヨーク州立大学のサイド・ドア・インターフェースをもった接続法等があげられる[4]。しかしながら，これらの接続法は，Multi-Busに基づくものであり，コスト的にメリットがある半面，さらに大規模化を目指すには，バスのコンフリクトや，そのためのコントローラの制御等の多くの未解決な問題が残されている。

一方，プロセッサ間を直接接続したものとして，コロンビア大学のCHOPPシステムのn-cube接続法[5]がある。これは10^3～10^6個程度のプロセッサをn次元立方体の各頂点に配置し直接接続するものである。しかしながら，このシステムの具体的な実現法については詳しくは提案されていない。

本稿では，このn-cube直接接続の利点を重視し，各プロセッサのポート数を極力少なくし，制御も極めて容易にした大規模システムとしてより実現可能な時分割n-cube接続法を提案する。

この接続法の特長は，プロセッサ1個内部接続網をスイッチング素子のみで構成し，その制御をMIMDシステム内の他の構成要素とは独立に制御可能（dedicated type）にしたところにある。

次に，大規模MIMDシステムで基本的問題である並列処理に対する柔軟性，及び，故障に対する信頼性を考慮して，我々が提案した接続法を，プロセッサ間通信の立場で評価，検討を行なっている。

そこでは，まず，並列ジョブを他のプロセッサに割り当てる際にできるかぎりI.N.の構造に依存しないこと，及び，信頼性等を考慮した効率の良い転送方法を提案している。この転送方式の特長は，n-cube接続法で行なう転送方式に比較し，各プロセッサでの転送のためのハード・ソフトの負荷が小さく，1つのポートの使用率を向上させている点にある。

2. 本MIMDシステムの基本的特徴とその構成

2.1 MIMDシステムの基本方針

ここでは，我々が目指すMIMDシステムの基本的な骨子について簡単に述べる。

MIMDシステムの基本は，各プロセッサが独立稼動し，並列ジョブ（タスク）を他のプロセッサに自由に割り当て可能であることであろう。このことを考慮に入れた上で，大規模計算機複合体を可能にするシステムを目指すことが我々の目的である。

大規模化可能なシステムは，その構成を機能別に分離すること，そして，その分離された各構成要素はできる限り独自の制御（dedicated control）を行ない，その制御も容易で，多少の故障に対しても耐え得ること等が，基本的な骨子であると考えている。

2.2 システム構成

このシステム構成は，図2.1に示すものである。このシステムは，プロセッサ（P），ローカルメモリ（M），そして通信用プロセッサ（CP）で構成されるプロセッサモジュール（PM），補助メモリ（AM），I/O チャネル，そして，プロセッサ間及び補助メモリ間の内部接続網（P-P. I.N.，I/O. I.N.）から成る。この構成をとる理由については文献〔6〕に譲る。

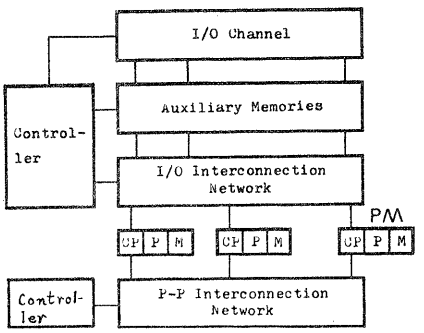


図2.1 システム構成

3. プロセッサ間内部接続網（P-P. I.N.）

ここでは，大規模MIMDシステムの並列処理の効率を決定する大きな要素であるP-P. I.N. の提案を行なう。その接続法としては前述したように n-cube 接続法をとる。この n-cube 接続の実現に対しては，Ω網を極めて簡単な時分割制御を行ない仮想的に n-cube 接続をとる間接接続法の提案を行なう。

3.1 時分割 n-cube 接続回路とその制御

まず，n-cube 上のプロセッサ（$N=2^n$）の位置はよく知られている様に，n bit の情報で表現される。

$$P(\lambda)=(p_{n-1}, p_{n-2}, \cdots, p_1, p_0)_2$$

そして，各ビットに対応する（$a_{n-1}, a_{n-2}, \cdots, a_1, a_0$）軸に対して，その軸に対応するハミング距離1のプロセッサ相互の接続が存在する。（図3.1（a））

次に，P-P. I. N. は，図3.1（b）に示される2つのΩ網によって構成される。1ポートをもつ各プロセッサ（$N=2^n$）は，そのプロセッサ番号の奇パリティ，偶パリティとによって，それぞれA，Bに2分割されている。また，Ω_0 網，Ω_1 網は，それぞれ，AからB，BからAへのブロック間の通信を行なうために用いられる。

このとき，Ω_λ（$\lambda=0,1$）網の各段を受信側から $S_0^\lambda, S_1^\lambda, \cdots, S_{n-2}^\lambda$ とし，S_j^λ 段の素子群 E_j^λ を上から $E_{j0}^\lambda, E_{j1}^\lambda, \cdots, E_{j2^{n-2}}^\lambda$ とし，ライン番号 l を上から，$l_{j0}^\lambda, l_{j1}^\lambda, \cdots, l_{j2^{n-1}}^\lambda$ とする。

このΩ網の素子Eは，その素子状態としては，"通過"（E=0）と"交叉"（E=1）の2状態をもついわゆるβ素子である。（図3.2）

図3.2 素子状態

次に，この接続網の素子制御法を示す。

〔時分割 n-cube 接続法〕

ある時刻 t_k において，基本制御コントロールコードを $B_k=(b_{n-1}, b_{n-2}, \cdots, b_0)_2$，コントロールコードを $C_k=(c_{n-2}, c_{n-3}, \cdots, c_0)_2$ とする。又，左へ1巡回シフトする操作を f とする。

このとき，

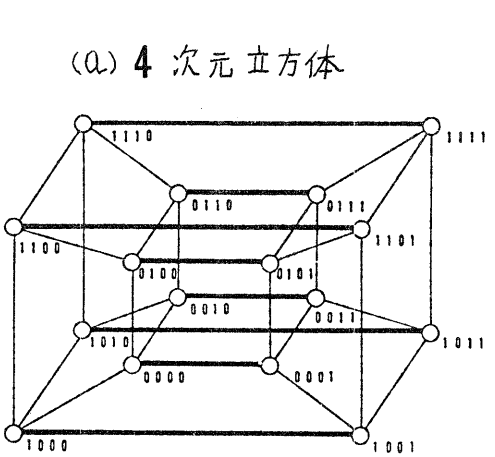
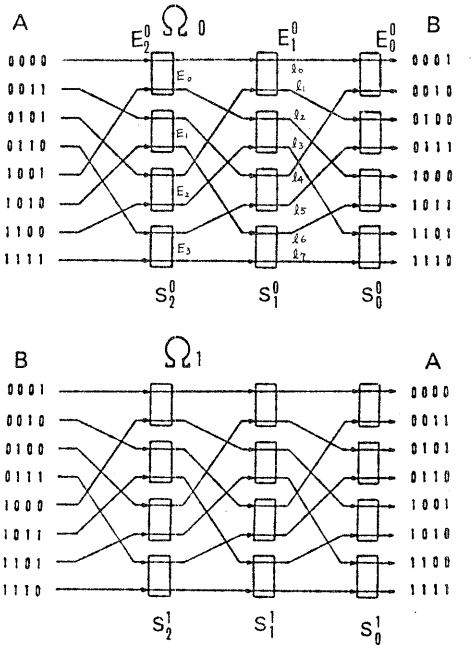
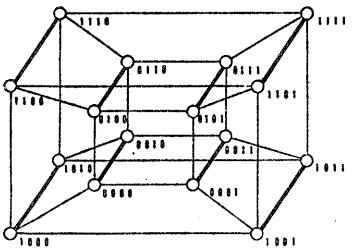


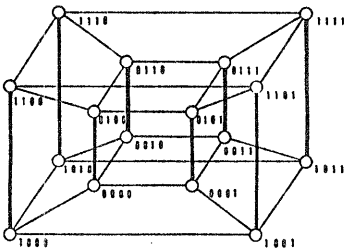
図 3.1 4-cube と時分割 4-cube 接続

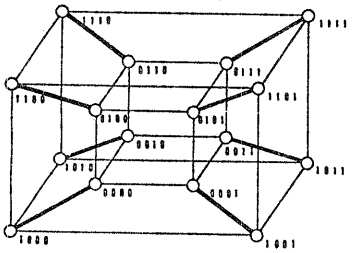


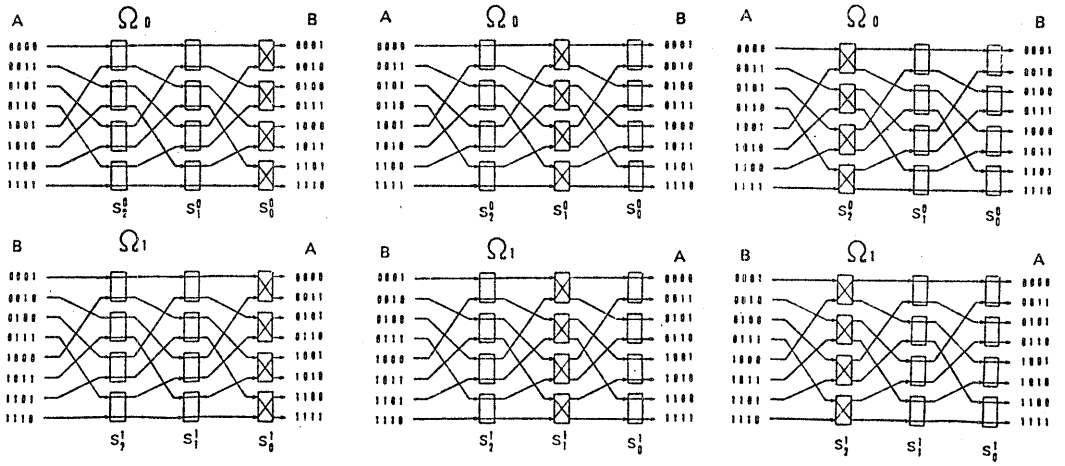
(b) 時分割 4-cube

図 3.3 時分割転送図









$$\left.\begin{aligned} \mathbb{B}_0 &\triangleq (1,0,0,\cdots,0)_2 \\ \mathbb{B}_{k+1} &\triangleq f\cdot \mathbb{B}_k \\ \mathbb{C}_k &\triangleq \mathbb{B}_k \pmod{2^{n-1}} \end{aligned}\right\} ①$$

①式で与えられる $\mathbb{C}_k$ に対して，その要素 C_ℓ に対応する段 S_ℓ 全体のコントロール $C_\ell^0(k)$, $C_\ell^1(k)$ は，

$$C_\ell^0(k) = C_\ell^1(k) = C_\ell \quad (\in \mathbb{C}_k)$$

でコントロールするものである。ここで，

$C_\ell^i(k)=0$ のとき $\forall E \in \mathbb{E}_\ell^i$, $E=$"0"

$C_\ell^i(k)=1$ のとき $\forall E \in \mathbb{E}_\ell^i$, $E=$"1"

とする。

以上の接続法により，t_0 では a_0 ビット目，t_1 では a_1 ビット目，…，t_λ では a_λ ビット目，そして t_{n-1} では a_{n-1} ビット目の転送が行なえることになり，仮想的に最悪 n サイクル単位で n-cube と同値な転送を，この回路網上で行なえることが示される。（図3.3）

3.2. kポート時分割 n-cube 接続回路

次に，n-cube 接続と時分割 n-cube 接続との中間的性質をもつネットワークとして，kポート時分割 n-cube を考える。これは，一般に同時転送が k 個できる時の場合であると考えられる。

この kポート時分割 n-cube では，$k=\frac{n}{m}$ とする。今，プロセッサ番号を mビット単位で k 個のブロックに分割する。そこで，プロセッサ番号の $(\lambda-1)$ブロック目以外のコードが同じ 2^m 個のプロセッサ群は，λ番目のポートで2つの大きさ 2^{m-1} のΩ網によって接続される構成になっている。（図3.4）このΩ網全体の数は $(n-k)\cdot 2^n$ で与えられる。以上のように構成された kポート時分割 n-cube は，$k=n$ の時直接 n-cube 接続と同値になる。

3.3 時分割 n-cube 接続網での故障の性質

ここでは，時分割 n-cube 接続回路であるΩ網での故障の性質について述べる。

故障に対する異常検出のために，便宜上，素子の故障を出力のラインで示すことにする。素子故障のパターンは，図3.5に示すように，"交叉"における故障①と，"通過"における故障②との2つを考える。

1) ①のパターンの故障

この故障は時刻 t_{j+1} に対してのみ起こる故障である。

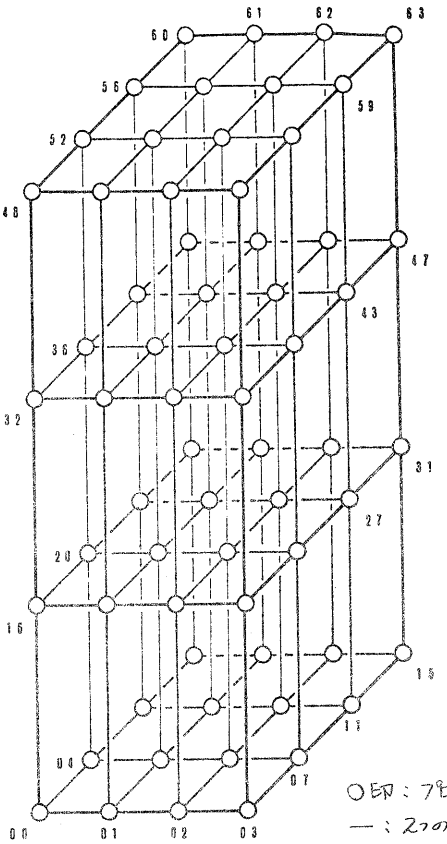


図3.4　3ポート時分割6cube 接続法

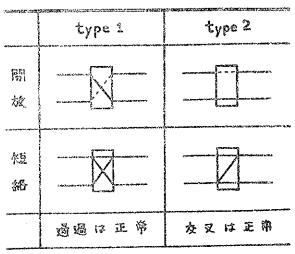


図3.5　素子の故障パターン

Ω_i 上で j段目のあるライン ℓ_j^i が故障した時，異常が生じるプロセッサ番号 P, P' は，

$P=(p^i\cdot \ell_j^i)\cdot 2+g$

（$g\in\{0,1\}$ で P の parity を1にするもの）

$P'=P\oplus 2^{j+1}$ (但し，$\oplus$ は EXOR である)

で与えられる。

すなわち，時刻 t_{j+1} のみにおいて，プロセッサ番号 P なるプロセッサは受信不能になり，逆に，プロセッサ番号 P' なるものは送信不能となる。

ii) ②のパターンの故障

この故障は時刻 t_{j+1} 以外のすべての時刻に対して起こる故障である。

Ω_i 上で j 段目にあるライン l_j^i が故障した時，時刻 t_k ($0 \leq k \leq j$) に対して異常が生じるプロセッサ番号 P_1, P_1' は，

$P_1 = (f^i \cdot l_j^i) \cdot 2 + \delta$

($\delta \in \{0,1\}$ で P_1 の parity を0にするもの)

$P_1' = P_1 \oplus 2^k$

で与えられる。

すなわち，時刻 t_k において，プロセッサ番号 P_1 なるプロセッサは送信不能となり，逆に，プロセッサ番号 P_1' なるものは受信不能になる。

加えて，時刻 t_0 と t_k ($j+2 \leq k \leq n-1$) に対しては，異常が生じるプロセッサ番号 P_2, P_2' は，

$P_2 = (f^i \cdot l_j^i) \cdot 2 + \delta$

($\delta \in \{0,1\}$ で P_2 の parity を1とするもの)

$P_2' = P_2 \oplus 2^k$

で与えられる。

すなわち，時刻 t_k において，番号 P_2 なるプロセッサは受信不能で，逆に，P_2' なるプロセッサは送信不能となる。

以上が故障パターンに対する異常状態の関係である。

3.4 時分割 n-cube の接続回路としての評価

MIMDシステムでの並列処理の効率を決定する1つとして，内部接続網の構造が上げられる。リソースの問題を除くなら，プロセッサ間の考えられる接続に対して，直接プロセッサ間接続する完全グラフ接続が，要求の衝突や転送の待ちが無いという点で理想的であると言える。しかし，プロセッサ数Nの増加に伴って，結線やポートの数が $O(N^2)$ で増加するという欠点を持っている。

そのためMIMDシステムも含めて一般の計算機複合体では完全グラフ接続でない内部接続網を用いて，仮想的に完全グラフ接続を実現している。この実現方法には，時分割によるもの（バス，XB網，Beneš Net，Batcher Net等）と，空間分割によるもの（ループ・ネット，2次元格子網，n次元トロイダル網等），そして，それらの組み合わせによるものの3通りが考えられる。時分割型の場合には，接続を決定するまでの計算時間，要求の重なった場合（コンフリクト）による待ちの発生が問題となる。空間分割型の場合には，ルーティングの決定，ルーティングが重なった場合（コンフリクト）による待ちの発生が問題となる。

プロセッサの数が極端に多い場合には，各プロセッサからの要求を一括して，I.N. を管理するのは困難である。集中制御型の時分割内部接続網はこの点好ましくない。空間分割型接続網では，経路決定を各プロセッサで分散し，この問題を解決できる。

従って，大規模なシステムにおける I.N. は次の条件を満たすことが望まれる。

(i) 結線ポート素子数が少ない。

(ii) 空間分割的な手法で転送を行なう。

(iii) 実現の容易さ。

メレンのMDMPやCHOPPシステムが採用している Boolean n-cube 接続は(ii)を満たしている I.N. になる。しかし，プロセッサ数Nに対して，ポートの数が $N \cdot \log_2 N$，結線が $(N \cdot \log_2 N)/2$ 組必要になるため(i)の条件を完全に満たしていない。また立場をかえて，この様なマルチポートシステムにおいては，各ポートを最大限に使用した場合，メモリのアクセスタイムは，$1/\{(\text{ポートの速度}) \times (\text{ポートの数})\}$ 以下であること，又，各モジュールでの待ちのトランザクトの処理をしなければならない。従って，このn-cube接続法は，各ポートのメモリを分割し，各々独立に制御することが要求され，実現上極めて困難なものになる。加えて，各ポートは大半が同時に動作することは少なく，ポートの使用効率を高めることもできなくなる。

我々は，これらの欠点を補なうために，時分割 n-cube 型接続網を用いることを提案する。この接続網は時分割型の特長である素子数，ポート数の少なさ，各プロセッサの待なり処理の軽減と，空間分割型の特長であるシンプルな接続とを兼ね備えた接続網であると言えよう。この時分割接続網においては，Ω網には，M.C.

Pease が提案[7]している素子数最小の接続法が当たっている。又，その制御に関しては，今までに提案されている様な複雑な制御は行なわず，単に各段を"通過"や"交叉"にするー定のくり返し制御を行なうだけであるから，Ω網を制御するためのプロセッサは，もはや必要がない。(iii)の実装の問題に関して，Boolean n-cube では，最適な時 $\lceil (n+1)/4 \rceil$ 層で実装可能なグラフであるのに対して，時分割 n-cube ではΩ網を用いているため，2層で実装可能となっている。又，Nポート時分割型 n-cube は，Boolean n-cube と時分割型 n-cube との中間の性質を持っていることは前述の通りである。

最後に，各接続法の比較を表3.1に示す。

表3.1 各接続法の比較

	直接・接続			間接・接続			
	完全グラフ	n-cube	n-トロイダル	マトリックス	Benes	T.S.n-cube	Nポート T.S.n-cube
1モジュール当りポート数	N-1	n	$2n$	1	1	1	k
選択制御の複雑度	$N \times \log_2(N-1)$	$N \times \log_2(n-1)$	$N \times \log_2(2n-1)$	$n \cdot N$	$N \times \log_2 N$	0	$N \times \log_2 k$
制御方式	分散	分散	分散	集中	集中	集中	集中・分散
素子数				N^2	$N \log_2 N$	$N \log_2 \frac{N}{2}$	$N(\log_2 N - k)$
平1近傍数	N-1	n	$2n$	N-1	N-1	n	n

4. プロセッサ間通信からのI.N.の評価

ここでは，大規模MIMDシステムに適合したプロセッサ間通信のあり方を考察し，前述した時分割 n-cube 接続網上で効率の良い転送方式を明らかにしていく。

4.1 MIMDシステムに適合したプロセッサ間通信

プロセッサ間通信の中でP-P.I.N.で行なわれる主な通信は次の2つに大別される。

(1) Broadcasting 通信

(2) メッセージ通信

まず，(1)の"Broadcasting 通信"とは，あるプロセッサにおいて，複数個の新しい並列プログラム，ユニットが発生した際に，他のプロセッサにその並列プログラム，ユニットを割り当てるために，任意の空きのプロセッサに対して，プロセッサを獲得する事を知らせる通信である。

次に(2)の"メッセージ通信"とは，プログラム，データ，制御用(プログラムの開始，終了等)情報等の交換，転送のために行なわれる通信である。

(方針1) P-P.I.N.では，以上のように異なる使用目的のために2種類の通信が存在する。システムの実現の容易さを考慮するとき，この2つの通信は，同じ接続網を多重使用することによって実行されることが望ましい。

MIMDシステムにおけるもう1つの基本的問題として，並列処理に対する柔軟性の問題がある。この解決法としては，並列ジョブを他のプロセッサに割り当てる際に，できる限りI.N.の構造を意識せず，又，依存しない事が望ましいことがあげられる。

(方針2) あるプロセッサが他のプロセッサを必要とする際，ブロードキャスティングを用いて配置する場合，距離の近い空きのプロセッサを獲得する。

この理由としては，距離の近いプロセッサに割り当てない場合，転送時間が長くなり，I.N.のトラフィックも増加し，さらに全体のメッセージの転送時間にも影響をおよぼすことがあげられる。

そして，最後の方針としては，このシステムは大規模を指向していることから，ある程度の故障に対し，耐え得るものでなくてはならない。

(方針3) プロセッサ間通信で重要な情報は決してその通信時の間消滅しないということがあげられる。

4.2 プロセッサ間通信方式

以上3つの方針に基づくプロセッサ通信を，時分割 n-cube 上で行なう効率の良い転送方式を次に述べていく。

4.2.1 プロセッサ間通信の経路選択

ここでは，P-P 間の接続が，この n-cube 上でどのように行なわれるかを明らかにする。

任意の2つのプロセッサ間通信は，次に定義

される経路 R によって決定され，かつ転送される。

$R = P_S \oplus P_D$

P_S ： 送信側プロセッサ番号

P_D ： 受信側プロセッサ番号

各プロセッサは，Rによって表わされるコードのうち"1"のある任意の軸へ転送を行ない，受信側のプロセッサが転送をチェックした後にそのビットを"0"に変える。このことにより $P_S \rightarrow P_D$ の転送が可能になる。

以上の経路選択は，n-cube，及び，kポート時分割 n-cube 接続に対して成立する。従って，これらの接続法が大規模システムにおいてのアドレス指定，及び，経路選択に関し，極めて有利な性質をもっていることを示している。

実際の転送に当たっては，図4.1に示すシステムにおいて行なわれる。そこでは，時刻 t_i における基本コントロールビット $B \triangleq (b_{n-1}, b_{n-2}, \cdots, b_0)_2$ がΩ網上の素子制御の役目と各CPにおいて a_i 軸の転送を行なわせるためのGate信号の役目とを果している。

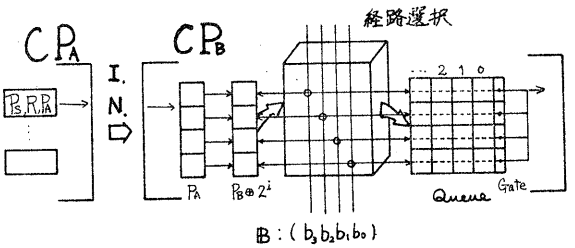


図4.1 情報転送概略図

[転送方法]

今，プロセッサ番号 P_A から P_B へ送信しようとする際，次のような転送を行なう。

P_A 側では，3つ組情報（P_S, R, P_A）をもって，Bにより時刻 t_i で $P_A \rightarrow P_B$ への転送を行なう。このとき，もし a_i 軸の回線が切れている時，それが他の回線で転送可能の時はそのまま Queue に入れておく。もし，全ての可能な回線が切れている時は，P_S と P_D を入れ替え，発信局に送り返すこととする。

P_B 側では，

(i) Bにより転送時刻 t_i を確認

(ii) IF $P_B \oplus 2^\lambda = P_A$ GO TO (iii) else 転送ミス

(iii) IF $R \cdot 2^\lambda = 1$ GO TO (iv) else 転送ミス

(iv) Rの λ ビット目を"1"から"0"にする。

(v) P_B の Queue に入れる。

4.2.2 ブロードキャスト転送アルゴリズム

ブロードキャスト（リクエストとキャプチャー）は，重み W をもって送られる。この重み W とは，発信又は中継する CP が W 回連続してブロードキャストを行なうためのコードである。全てのプロセッサにブロードキャストが行き渡るために，発信局の CP の重みは，n であれば良い。

この転送のアルゴリズムは，図4.2のフローチャートに示すもので，図4.3のように行なわれる。

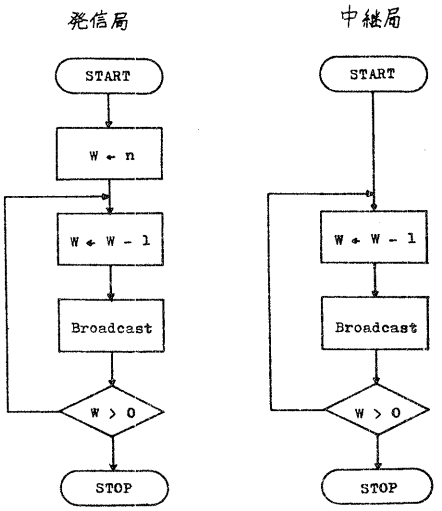


図4.2 B.C. 転送アルゴリズム

4.2.3 プロセッサ間通信方式

プロセッサ通信の方式はパケット交換を用いどのパケットも，図4.4に示すパケット構成をとる。又，時分割制御であるため，パケットは固定長とする。又，4.2.1の経路選択法では，メッセージ転送の経路は可変であることから，数パケットに分割してメッセージを送る場合，メッセージ領域内にパケットの順番を示す番号を必要とする。

ブロードキャスト用領域は，B.C.R.，B.C.C.，B.C.A.，のコードより成る。

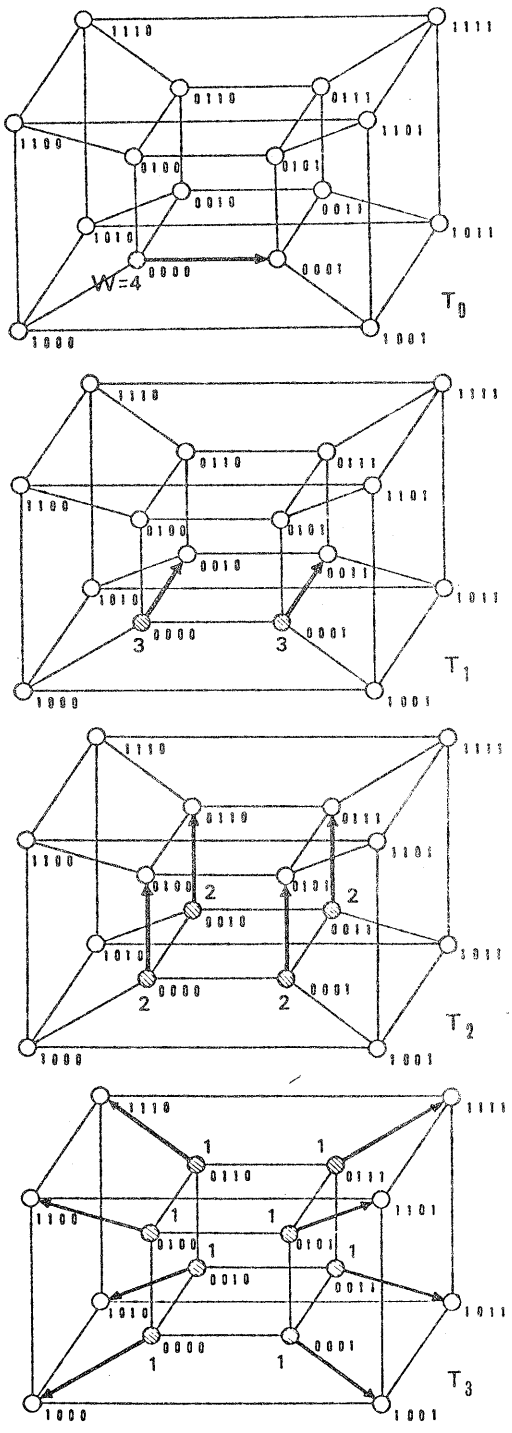


図4.3　ブロードキャスト転送図
（4-cube）

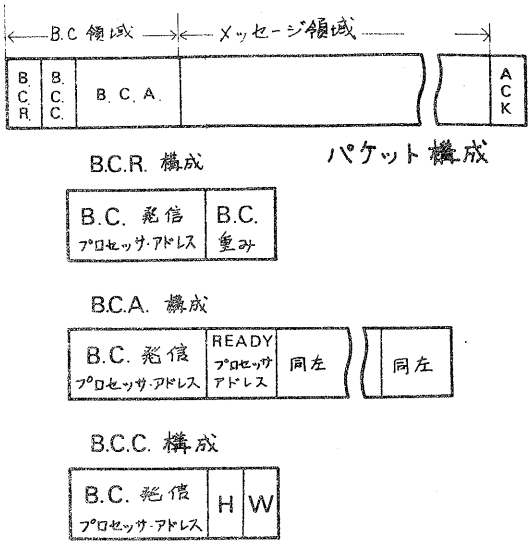


図4.4　パケット構成図

・B.C.R.（Broadcasting Request）

並列プログラム・ユニットを割り当てるプロセッサを獲得する要求を示す信号であり，転送方法は 4.2.2 に示す通りで，その情報は，図4.4に示す通りである。

方針2を実現するために，各プロセッサは，L.B.R.Reg（Latest Broadcasting Request Register）を設け，これに送られて来た B.C.R. を入れる。新たに，B.C.R. が到着した時，それが L.B.R.Reg 内の B.C.R. よりハミング距離が自分に近い時，L.B.R.Reg を書き変える。又，B.C.R. を発信，及び中継を行なっているプロセッサに，他の B.C.R. が到着した場合，両者の発信局アドレスを比較しハミング距離で自分に近い B.C.R. のみをブロードキャストする。（但し，等距離の場合，先着順とする。）これにより，B.C.R. の衝突は解消される。

B.C.R. を受けたプロセッサは，B.C.A. を返すべき最初の時刻 T_f（図4.5参照）までに，プログラム・ユニットを処理できる状態であるとき，B.C.A. によって "Ready" を直ちに知らせ，自身は "Wait" 状態に入り，以後他の B.C.R. を受けつけない。但し，"Wait" 状態は，B.C.R. が到着してから，$2n \cdot$ unit times の間だけ保持され，その間に B.C.C. が転送されてこない場合 "Ready" 状態に戻り，

L.B.R.Regの内容は初期化される。又，B.C.A.を返すべき最初の時刻までに，新たなハミング距離のより近いB.C.R.が到着した場合，L.B.R.Regの内容をより近いB.C.R.に書き変える。

◦ B.C.A.（Broadcasting Answer）

これは，B.C.R.に対して"Ready"であることを伝えるための情報であり，図4.4に示すものである。

B.C.R.を受けたプロセッサは，返答できる最も早い時刻T_fまでに"Ready"になった場合，"Ready"であることを示すB.C.A.領域に自分のアドレスを入れ転送する。B.C.Aの中継局に，B.C.A.領域以上のB.C.A.が到着した場合，B.C.R.を出したプロセッサに近いプロセッサだけをB.C.A.領域に入れ，他を切り捨てる。（図4.5）

◦ B.C.C.（Broadcasting Capture）

これは，B.C.R.を出したプロセッサが，B.C.A.を返したプロセッサを獲得することを示す情報である。

B.C.A.を受けると，その中から最も近いプロセッサから順に選択し，要求する個数まで取る。この情報は，この獲得するプロセッサの内最も遠いプロセッサの距離（H）を情報として，B.C.C.をB.C.R.を出した$2n$ unit times後に送る。B.C.Aを出し，B.C.C.に書かれた距離H内にあるプロセッサは獲得され，プログラム，ユニットの割り当てを待つ。H外にあるプロセッサは解放され"Wait"状態"から"Ready"状態となる。獲得するプロセッサの情報をハミング距離Hで送るため，一般に，端数があり，獲得されたにもかかわらず，プログラム，ユニットを割り当てられないプロセッサが生じる。これに対する解放は，個々にメッセージ・モードで行なう。

なお，1回のB.C.R.によって獲得できたプロセッサが，要求する数より少ない場合，新たなB.C.R.を行なうが，1つのプロセッサが多くのプロセッサを独占しないために，新たなB.C.R.は以前に行なったB.C.R.の$3n$ unit times以後とする。

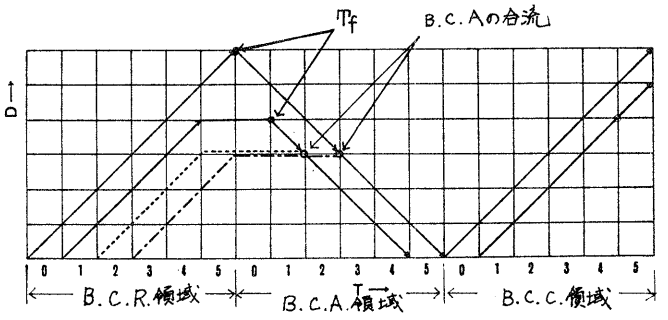


図4.5　B.C.転送のタイムチャート（6-cube）

以上の方式を用いることによって，方針2を実現し，近くのプロセッサだけを獲得し，転送時間の短縮と，I.N.全体のトラフィックを下げることができる。

4.2.4　ポート時分割n-cube転送方式

ポート時分割n-cubeにおける転送においても，時分割n-cube網で論じた転送方式が，ほとんど同様にできることが示される。

まず，プロセッサ番号をmビットごとに分割したl個のブロックに対して，

$$D_i \triangleq \bigcap_{j=0}^{m-1} R_{ij}$$

（R_{ij}：iブロックでの$j$bit目である）

とし，Broadcastにおけるweightを次のように定義しておく。

W：ポートの選択

$w(i)$（$i=0\sim n-1$）：各ポートのタイミング

(1) 情報転送に関しては

(i) $D_i=1$である最小のiを求め，その番号の出力ポートのQueueへ送る。

(ii) 各ポートは，時分割n-cubeの転送方式に従う。

(iii) D_iが全て0なく，転送終る。

(2) Broadcastingに関しては。

（Broadcasting発信）

i番のポートに対して，$W \leftarrow i$なる重みをつけて，時分割n-cubeのBroadcastに従って，Broadcastを行なう。

〈Broadcast 中継〉

Wなる重みのBroadcastを受け取ったく，0～W-1番までのポートに対しては，時分割n-cubeのBroadcastに従って，Broadcastを新たに行なう。その他の軸に対しては，時分割 n-cube の Broadcast を続ける。

4.3 情報の保持とその制御

本システムでは，大規模なシステムを指向しているので，P-P.I.N.の素子数は非常に多くなり（ex. 10 cube で 2^{10}個のプロセッサに対し，40~6個のβ素子），故障する可能性を考慮しなければならない。

そこで，前述の経路選択アルゴリズム，n-cube 時分割接続における故障問題を，信頼性の立場から情報の保持について，又，I.N.保守の立場から故障検出について論じる。

4.3.1 回線切れのための"情報保持型"

方針3に基づき，故障の場合でも転送情報が抹消されないように情報の保持をしなければならない。

このためには，各プロセッサは，時刻 t_i に正常に情報を受信した時ACK信号を，異常であった時 $\overline{\text{ACK}}$ 信号を返信することにする。

そして，各プロセッサは，又，送信すべき情報を別途に保持して転送することにする。そして相手方のACK信号の未獲得の場合や，$\overline{\text{ACK}}$信号の受信の場合，自分の情報を保持し，又，相手方の情報も抹消してしまう。

このことより，必ず転送異常があった時情報を送らない状態で情報を保持できる。（例，表4.1，図4.6）

4.3.2 故障素子の検出

3.3 で行なった議論に基づき，次に示す表4.2の様に検出が可能である。

なお，素子の誤動作の場合，ACK信号の上手な転送により，正しく転送された情報のACK信号を，発信局に戻すことが可能であるが，この詳しい議論は，別の機会に譲る。

表4.1 接続網の断線に対する処理

パターン	通信用プロセッサ	情報パケット ℓ→m 入力端	出力端	処理	情報パケット m→ℓ 入力端	出力端	処理
1	CPℓ	$\text{ACK}_{m\to\ell}$	Send	delete	Recieve	$\text{ACK}_{\ell\to m}$	
	CPm	Recieve	$\text{ACK}_{m\to\ell}$		$\text{ACK}_{\ell\to m}$	Send	delete
2	CPℓ	$\overline{\text{ACK}}_{m\to\ell}$	Send	hold	Recieve	$\text{ACK}_{\ell\to m}$	delete
	CPm	——	$\overline{\text{ACK}}_{m\to\ell}$		——	Send	hold
3	CPℓ	——	Send	hold	——	$\overline{\text{ACK}}_{\ell\to m}$	
	CPm	——	$\overline{\text{ACK}}_{m\to\ell}$		——	Send	hold

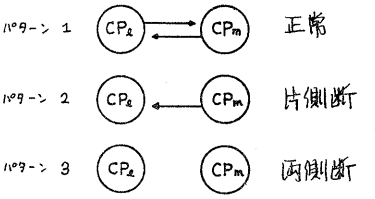


図4.6 回線切れのパターン

表4.2 故障検出

故障パターン / 個数	"交叉"における故障	"通過"における故障	両者の複合
1個	検出可能	検出可能	
複数個 (m)	検出可能	少なくとも2個検出可能	少なくとも1個検出可能

4.4 転送能力に関する評価

4.4.1 メッセージ転送能力に関する評価

ここでは，時分割n-cube接続の転送能力をn-cube接続のものと比較，検討する。

本システムでは，ハミング距離の近いプロセッサを優先して獲得する方式をとっているため，パケット分配率は距離の近いプロセッサほど高くなる。ここでは，分配率が均等である場合（ネットワーク転送能力のワーストケース）について，次の仮定のもとにシミュレーションを行なう。

［仮定］ 時分割n-cube接続とn-cube接続とを対等な立場で比較を行なうため，時分割n-cube接続の各プロセッサでのバス，ドライバーの転送速度をn-cube接続のものの n 倍とする。

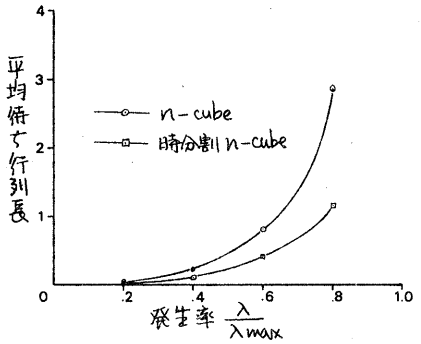
次に1パケット転送に要する時間を1 unit time とし，パケット分配率を 1/(N-1)，各プロセッサのパケット発生率を λ(パケット/T) とする。

以上の仮定より，1 unit time に，あるプロセッサが出力できるパケット発生率の限界 λmax は次式で与えられる。

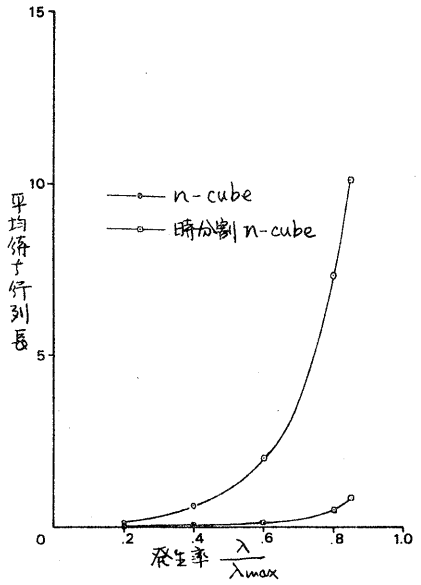
$$\lambda_{max} = \frac{(2^n - 1)}{2^{n-1}} \quad (\text{パケット/T})$$

〔結果〕

このシミュレーション結果は，図4.7に示す。



(a) n=4



(b) n=8

図4.7 発生率と平均待ち行列長の関係

このシミュレーション結果では，時分割 n-cube 接続は，n-cube 接続に比べ，転送効率の面において良く，特にこの転送効率は，高次元モデル程良くなっている。

このことは，n-cube 接続で並列メモリを使用し，又，同じ転送速度のバスドライバーを使用し，n-cube 接続では n 個，時分割 n-cube 接続では1個とした仮定に対しても，n-cube 接続の時分割 n-cube 接続に対する転送能力比は n 倍であるにもかかわらず，転送効率比は n 倍をかなり下まわっている。

これらのことから，時分割 n-cube 接続の転送効率は n-cube 接続に比べ，良いことが示される。

〔検討〕

この原因は，次のように考えられる。時分割 n-cube 接続は，各プロセッサでバッファが1個である。しかし，n-cube 接続では，バッファを n 個持つため，各バッファ内の待ちの長さが一時的に不均衡を生じ，そのことが転送効率の劣下をもたらす原因であると考えられる。

従って，n-cube 接続を，時分割 n-cube 接続の転送効率と同値にするには，n-cube 接続で転送速度 1/n のバスドライバーを n 個用い，かつ，パケットをバッファ内に入れる際に，n 個のバッファの待ち行列長を比較し，転送できる回線軸の内で最も待ちの短いバッファにパケットを入れるという難かしい制御を行なうことが必要であると考えられる。

4.4.2 ブロードキャスト転送に関する評価

時分割 n-cube 接続では，ブロードキャストは重み W だけの情報を用いて，非常に簡単なアルゴリズムで転送でき，かつ，n-cube 接続と同じ n unit times で終了する。従って，n-cube 接続と時分割 n-cube 接続とにて，各プロセッサの転送能力を等しくした時，時分割 n-cube 接続は，n-cube 接続の 1/n の時間で転送を終了できることになる。

又，時分割 n-cube 接続は，固定長のパケットを使用しているが，複数個のプロセッサの Ready 信号を，1つの B,C,A, で返すことにしている。この時，B,C,C, の情報は，ハミング距離 H で表わされていることから，1回の

B.C.R. で多くのプロセッサを獲得可能になっている。

以上の時分割 n-cube 接続網に，この B.C. の転送方法を用いることにより，並列度の大きく変化する問題に対して適したものであり，大規模MIMDシステム向きであると考えられる。

参考のために，実際の時分割 n-cube の近傍数は，みかけ上次のように求められる。

kポート時分割 n-cube 接続において，あるプロセッサから距離 r 離れたプロセッサ群の個数を N(r) とする。

$$N(r)=\sum_{\text{for all partition}}\left\{U(d_0,d_1,\cdots d_{k-1})\cdot\prod_{\substack{j=0\\ d_j\neq 0}}^{k}2^{d_j-1}\right\}$$

$$\begin{cases}\sum_{i=0}^{n} d_i = r\\ 0\leq d_i\leq D_i\\ d_i\leq d_j\ (i<j)\end{cases}$$

ここで $D_k=\log_2 q$

q は a_i 軸に接続する入力数

$U(d_0,d_1,\cdots,d_{k-1})$：

$(d_0,d_1,\cdots,d_{k-1})$ の並びにおける重複順列

$k=1$ のとき $N(r)=2^{r-1}$

$k=n$ のとき $N(r)=nC_r$ となる。

5. おわりに

本稿では大規模MIMDシステムの基本方針を明らかにし，それに適したプロセッサ間内部接続法として時分割 n-cube 接続法を提案した。この接続法は，他のシステム要素とは独立に稼動する極めて制御の簡単なスイッチング回路によって実現されることを示した。

又，MIMDシステムでのプロセッサ間通信をこの時分割 n-cube 接続網上で，効率よく転送するための転送方法について論じ，この時分割 n-cube 接続網は，その直接実現した n-cube 接続網に比べ，転送の面で種々の利点があることを示した。

又，時分割 n-cube 接続網の故障対策は，大規模システムにおいて重要な問題の1つであることを示した。

又，故障に対してかなりな程度対処できる可能性をもたせる重要な課題であるが，kポート時分割 n-cube 接続法（$k\geq 2$）を用いることが有力な解決法であると考え，種々の故障対策の問題とあわせて，現在研究を行なっている。

<参考文献>

[1] M.J.Flynn,"Some Computer Organizations and Their Effectiveness," IEEE Trans. on Comput.,Vol.C-21,No.9,pp948-960,Sep.,1972.

[2] L.D.Wittie,"Efficient Message Routing in Mega-Micro-Computer Networks," IEEE Annu. Symp.,Vol.4,No.4,ppl36-140,Jan.,1976.

[3] H.G.Arnold and E.W.Page,"A Hierarchical, Restructurable Multi-Microprocessor Architecture," IEEE Annu. Symp.,Vol.4,No.4,pp40-45, Jan.,1976.

[4] 飯塚肇，"マイクロ・プロセッサ複合体の技術" 通信学会誌 別冊 Vol.60, No.2 1977

[5] H.Sullivan and T.R.Bashkow,"A Large Scale, Homogeneous, Fully Distributed Parallel Machine, I," IEEE Annu. Symp.,Vol.5,No.7, ppl05-ll7,Mar.,1977.
H.Sullivan and T.R.Bashkow,"A Large Scale, Homogeneous, Fully Distributed Parallel Machine,II," IEEE Annu. Symp.,Vol.5,No.7, pp90-94,Mar.,1977.

[6] 川上，岸 他，"MIMDシステムにおける一提案"，信学会全国大会 S53，発表予定

[7] M.C.Pease,"The Indirect Binary n-Cube Microprocessor Array," IEEE Trans. on Comput.,Vol.C-26,No.5,pp458-473,May,1977.