超立方体結合と大域バス結合より成る一相互結合モデルについて†

渋沢 進††

本論文では，超立方体結合と大域結合より成る一結合モデルのレイアウトと，このモデル上でのいくつかの操作を考察している．超立方体結合に大域結合を付加しても2次元レイアウト面積のオーダは変化しない．ノード数が大きいとき，超立方体結合のノードを2方向反射2進順に配置したレイアウト面積は，昇順に配置したレイアウト面積の約4/9で実現できる．超立方体結合と大域結合より成る結合モデルでの1ノードから他ノードへのデータ系列の転送は，超立方体結合の各端子で同時に送受信できる場合でも，なおデータ数が少ない領域で大域結合の方が速い．データ転送の応用として，2次元大域結合をもつnノード結合モデル上で，$\sqrt{n}$ 個の行列積の最小コストは動的計画法を用いてオーダ $O(\sqrt{n})$ 時間で求められる．また，nノード超立方体結合上で，昇順と反射2進順のデータの並べ換えを $O(\log n)$ 時間で行う方法を示している．

1. はじめに

高度に並列的で高速な処理を行うことのできる計算システムのアーキテクチャ，アルゴリズム，言語モデル等が，工業や科学など多数の分野で強く求められている[1)~3)]．

相互に結合された多数の処理要素より成る結合モデルは，高速な処理を行う上で重要な構成である．これまで，隣接ノードとのみ通信する局所結合と，複数のノードに一度に情報を転送または放送することのできる大域結合に対して，多くの結合モデルが提案されてきた．

局所結合には超立方体結合[2),4)~7)]，格子結合，木結合，多段相互結合ネットワーク，CCC ネットワーク[8)]，シャフル交換ネットワーク，完全グラフ[9)]，円環等が含まれる．これらのうち，木結合，多段相互結合ネットワーク，CCC ネットワーク，シャフル交換ネットワークは超立方体の部分結合または変形したものである．完全グラフは他の構成に比較して多量のハードウェアを必要とし，むしろ並列システムの部分構成等に向いている．円環は処理要素数に比例した通信時間がかかり，分散的な環境向きである[10)~13)]．

相互に結合された処理要素より成る結合モデルはしばしばグラフで表され，グラフの頂点（ノード）と辺はそれぞれ結合モデルの処理要素と結合線で表す．結合モデル上で並列かつ高速な処理を実行するために，通信と経路の設定は重要な役割を果たす．なぜならば，もし結合モデル内の任意のノード間で同時に多数の経路を設定することができるならば，これは並列的にデータを転送し，処理することができるからである[14),15)]．

また，集積回路技術の進歩に伴って，多数の処理要素と結合線より成る並列計算向きの相互結合システムも集積化されていく．このとき，小さな2次元レイアウト面積をもつ結合モデルの実現方法を開発していくことは，効率的なコンピュータを実現する上で重要なステップである[1),16)~18)]．

通信の誤り訂正として発達した符号語のうち，引き続く符号語が1ビットだけ異なるような巡回的系列を Gray 符号とよぶ．このうち，符号語が2のべき乗ごとに反射的に1ビットずつ変化する Gray 符号が，反射2進 Gray 符号である[19),20)]．反射2進 Gray 符号語は，2進符号語と1対1対応し，かつこの2種類の符号間には簡単な関係が存在する．

本論文では，超立方体結合と大域結合より成る並列計算向きの結合モデルを導入し，その2次元レイアウトを評価し，この結合モデル上でのデータの並べ換え，ノード間のデータ転送，動的計画法の一解法を与えている．本論文では，超立方体のノードの番号付けが2方向昇順と2方向反射2進順の2種類の結合モデルを導入し，これらをそれぞれ単純 Cubemat，反射2進 Cubemat とよぶ．反射2進 Cubemat の各ノードは，東西南北の位置にあるノードと超立方体結合で巡回的に接続されているので，反射2進 Cubemat の超立方体結合は格子結合またはトーラスを陽に含んでいる．

† An Interconnection Model Consisting of Hypercube and Global Bus Connections by SUSUMU SHIBUSAWA (Department of Computer Science, Faculty of Technology, Gunma University).

†† 群馬大学工学部情報工学科

nノード Cubemat の2次元レイアウトの面積は，レイアウトに関する格子モデルを用いると $O(n^2)$ であり，これは超立方体結合に対する最適のオーダである[17]．またノード数が大きいとき，反射2進 Cubemat は単純 Cubemat の約 4/9 のレイアウト面積で実現することができる．

これまで，線形結合と格子結合に大域結合を付加した構成が提案されており，和，積，最大・最小値等を求める半群計算などの操作が局所結合だけの構成よりも高速に解けることが示されてきた[21)~24)]．超立方体結合と大域結合を用いた問題の解法に対しては，超立方体結合の方が大域結合よりも速く解ける場合，その逆の場合，両結合とも等しい時間かかる場合がある．

2ノード間で定数個のデータを転送するとき，単一の大域結合による転送は定数時間であるのに対して，nノード超立方体のノード間距離は高々 $\log_2 n$ であることより，つねに大域結合の方が速い．2ノード間でデータ系列を転送するとき，超立方体のノードの各端子を一度に1端子しか使用をしない場合には，大域結合の方が超立方体結合よりもつねに速い．2ノード間の s データ転送に対して，超立方体の各ノードの複数端子の同時使用を認めるとき，$s<O(\log n)$ では大域結合の方が速く，$s>O(\log n)$ では超立方体結合の方が速く，$s=O(\log n)$ では同一オーダの時間がかかるという結果を導くことができる．以上のことは大域結合が超立方体結合と比較しても高いデータ転送能力をもっていることを示している．Cubemat は超立方体結合と大域結合より成る結合モデルであるので，データ数に関するすべての領域で，超立方体結合と大域結合の速い方の時間でデータ転送できる．

問題の部分集合がデータ転送を含むような問題に動的計画法がある．動的計画法を用いた $\sqrt{n}$ 個の行列積の最小コストは，nノード超立方体結合を用いると $O(\sqrt{n}\log n)$ 時間で解くことができ，2次元大域結合を用いると $O(\sqrt{n})$ 時間で解くことができる．ノード間のデータ転送と動的計画法は，大域結合が超立方体結合よりも有効な二つの例である．

これとは逆に，超立方体結合の方が大域結合よりも速く解ける問題があり，バイトニックソートや高速フーリエ変換等はよく知られた超立方体結合向きの問題である．本論文では nノード Cubemat 上での昇順と反射2進順のデータの並べ換えを超立方体結合を用いて $O(\log n)$ 時間で行う方法を与えている．

なお，最近完全符号と準完全符号の集合で表される超立方体のノードに大域結合を付加することによって通信特性を向上させようという研究が発表されている[25)]．

次章では反射2進符号を導入する．第3章では単純 Cubemat と反射2進 Cubemat を構成し，これらの2次元レイアウトの面積を評価するとともに，他の結合モデルとの関係と大域結合の役割を簡単に述べている．第4章では2種類の Cubemat 上での昇順↔反射2進順のデータの並べ換えを与えている．第5章では Cubemat 上でのデータ転送を述べ，第6章では Cubemat 上での動的計画法の一解法を述べている．

2. 準　　備

反射2進 Gray 符号[19),20)]は，引き続く2符号語が巡回的に1ビットだけ異なり，2進符号と1対1対応する．ある非負整数に対する mビット2進符号語 e と反射2進 Gray 符号語 u を，次のように2進表現する．

$$e=[e_{m-1}\cdots e_1e_0],\quad u=[u_{m-1}\cdots u_1u_0] \qquad (1)$$
$$(e_i, u_i\in\{0,1\}, 0\leqq i\leqq m-1)$$

このとき，反射2進符号 u の第 i 桁 u_i は2進符号 e の第 i，$i+1$ 桁 e_i，e_{i+1} を用いて，次のように表される．

$$u_i=\begin{cases} e_i\oplus e_{i+1} & (0\leqq i\leqq m-2)\\ e_{m-1} & (i=m-1)\end{cases} \qquad (2)$$

ここに，記号⊕は排他的論理和を表す．また，2進符号 e の第 i 桁 e_i は反射2進符号 u を用いて次のように表される．

$$e_i=\bigoplus_{j=i}^{m-1} u_j \quad (0\leqq i\leqq m-1) \qquad (3)$$

3. Cubemat の構成

3.1 単純 Cubemat

正整数 m に対して，2次元平面 xy 上の配列 I, J $(0\leqq I, J\leqq 2^m-1)$ に $n=2^m\times 2^m$ 個のノードを置く．配列 I，J をそれぞれ座標 x，y 方向にとり，配列 I，J の2進符号をそれぞれ e^x, e^y とおく．

$$I=e^x=[e_{m-1}\cdots e_1e_0],\quad J=e^y=[e_{2m-1}\cdots e_{m+1}]$$
$$(e_i\in\{0,1\}, 0\leqq i\leqq 2m-1) \qquad (4)$$

配列 I，J にあるノードを $e(I,J)=\langle e^x, e^y\rangle$ で表せば，

$$e(I,J)=\langle e^x, e^y\rangle=[e_{2m-1}\cdots e_1e_0] \qquad (5)$$

式(5)で表される2ノード間に超立方体結合を導入する．2ノードの2進表現のハミング距離が1のとき，これら2ノード間に超立方体結合が引かれ，この

結合はx，yの両方向にまたがることはない．また，多くのノードに一度にデータを転送または放送できる大域結合を，ノード配列の各行，列方向に導入する．こうして構成されるノード結合構成を単純 Cubemat とよぶ．

図1は $n=2^6$ $(m=3)$ の単純 Cubemat の2次元レイアウト例である．図中□はノードを表し，実線は超立方体結合，点線は大域結合を表す．外周ノードの外向きの結合線は，Cubemat の外部の計算システムとのデータ入出力に使用する．単純 Cubemat の各ノードの端子構成の一例を**図2**に示す．図中各端子の数字はノード $e(I,J)$ の2進表現が，結合する相手ノードの2進表現と異なる桁を表す．桁 $0,1,\cdots,m-1$ は行方向の超立方体結合を表し，桁 $m,m+1,\cdots,2m-1$ は列方向の結合を表す．図には，行，列方向の大域結合端子も加えてある．各ノードの内部は，超立方体結合と大域結合の $(2m+2)$ 端子が相互に通信できるものであればどのような構成でもよい．

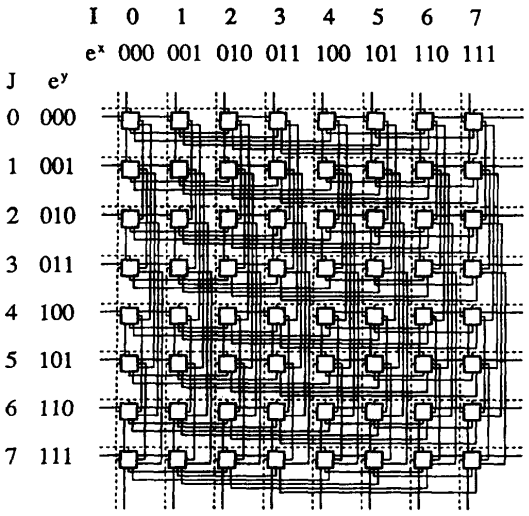


図1 単純 Cubemat の2次元レイアウト $(n=2^3\times 2^3)$
Fig. 1 A two-dimensional layout for a simple Cubemat $(n=2^3\times 2^3)$.

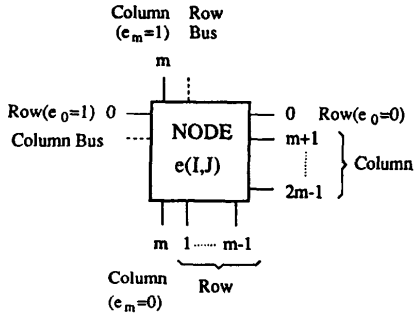


図2 単純 Cubemat のノードの端子構成
Fig. 2 A terminal layout of a node for the simple Cubemat.

配線幅を単位長さとするレイアウトの格子モデルに従って，単純 Cubemat のレイアウト面積を評価する．図2のよう $(2m+2)$ 端子をノード領域の縁に置くとき，ノードの一辺の長さは高々 $O(m^c)=O(\log^c n)$ (c：定数)，面積が $O(m^{2c})=O(\log^{2c} n)$ で実現できる[1),17)]．また，各行，列の超立方体結合による配線領域の幅が $(2+2^2+\cdots+2^{m-1})$ であることより，大域結合にある程度の余裕をもたせても，単純 Cubemat の一辺の長さは高々

$$2^m\times[(2+2^2+\cdots+2^{m-1})+O(m^c)]$$
$$=n+O(\sqrt{n}\log^c n) \qquad (6)$$

これより，単純 Cubemat のレイアウト面積は次のようになる．

［補題1］ ある正整数mに対して，$n=2^m\times 2^m$ ノード単純 Cubemat のレイアウト面積 A_{bin} は，次のように表される．

$$A_{bin}=n^2+O(n^{3/2}\log^c n) \qquad (7)$$

面積 A_{bin} は，超立方体の2次元レイアウトの最適オーダを満たしている[1),16)~18)]．

3.2 反射2進 Cubemat

2次元平面 xy 上の座標x，y方向にそれぞれ配列I，Jをとり，配列 (I,J) に $n=2^m\times 2^m$ 個のノードを置く．配列I，Jの反射2進符号をそれぞれ u^x，u^y とし，これらを次のように2進表現する．

$$u^x=[u_{m-1}\cdots u_1u_0],\quad u^y=[u_{2m-1}\cdots u_{m+1}u_m]$$
$$(u_i\in\{0,1\},\ 0\leqq i\leqq 2m-1) \qquad (8)$$

配列I，Jにあるノードを $u(I,J)=\langle u^x,u^y\rangle$ で表せば，

$$u(I,J)=\langle u^x,u^y\rangle=[u_{2m-1}\cdots u_1u_0] \qquad (9)$$

単純 Cubemat と同様に，ノード間，および行，列方向に超立方体結合と大域結合を導入する．こうして構成されるノード結合を反射2進 Cubemat とよぶ[26),27)]．**図3**は $n=2^6$ $(m=3)$ の反射2進 Cubemat の2次元レイアウト例である．図の表し方は図1と同様である．図3は，超立方体結合を上位桁から2ビットずつ対にしてレイアウトしたものである．反射2進 Cubemat の各ノード端子の構成を**図4**に示す．ここに，記号b，fはそれぞれノード $u(I,J)$ の2進表現がx方向の直前と直後のノードの2進表現と異なる桁を表し，記号 b'，f' はy方向に関するものである．その他の端子番号の表し方は図2と同様である．図4

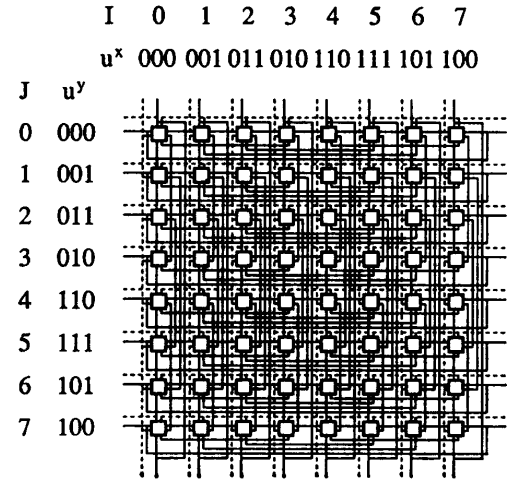


図 3 反射2進 Cubemat の2次元レイアウト ($n=2^3\times2^3$)
Fig. 3 A two-dimensional layout for a binary-reflected Cubemat ($n=2^3\times2^3$).

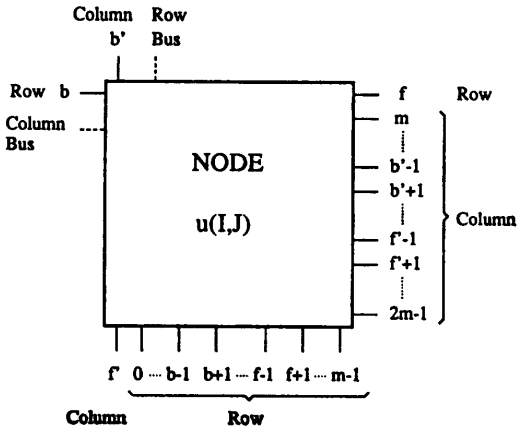


図 4 反射2進 Cubemat のノードの端子構成
Fig. 4 A terminal layout of a node for the binary-reflected Cubemat.

のノードの端子構成は，第2章で述べた2進符号と反射2進符号の関係を用いて求めることができる．

反射2進 Cubemat のレイアウト面積は次のように評価できる．

[定理1] ある正整数mに対して，$n=2^m\times2^m$ ノード反射2進 Cubemat のレイアウト面積 A_{ref} は，各方向とも下位または上位桁から2ビットずつ対にしてレイアウトするとき，次のように表される．

(1) 下位桁から対にするレイアウト

$$A_{ref}=\begin{cases}4n^2/9+O(n^{3/2}\log^c n) & (m:\text{偶数})\\ 25n^2/36+O(n^{3/2}\log^c n) & (m:\text{奇数})\end{cases} \tag{10}$$

ここに，cは定数を表す．

(2) 上位桁から対にするレイアウト

$$A_{ref}=4n^2/9+O(n^{3/2}\log^c n) \tag{11}$$

(証明)

(1) 図4のノードを用いるとき，ノードの一辺の長さが $O(m^c)=O(\log^c n)$ で各ノードの内部を構成できる．反射2進 Cubemat の各行または各列の超立方体結合による配線領域の幅[28]は，mが偶数のとき

$$1+(2^3+1)+(2^5+1)+\cdots+(2^{m-1}+1)=2\cdot2^m/3+m/2-8/3 \tag{12}$$

mが奇数のとき

$$1+[(2^3+1)+(2^5+1)+\cdots+(2^{m-2}+1)]+(2^{m-1}-1)=5\cdot2^m/6+m/2-25/6 \tag{13}$$

大域結合を含む各行，列の配線幅wは，大域結合にある程度の余裕をもたせても，次のように表せる．

$$w=\begin{cases}(2/3)\sqrt{n}+O(\log^c n) & (m:\text{偶数})\\ (5/6)\sqrt{n}+O(\log^c n) & (m:\text{奇数})\end{cases} \tag{14}$$

反射2進 Cubemat のレイアウトの各辺の長さは，wにノードの一辺の長さを加えたものを $2^m=\sqrt{n}$ 倍したものであり，それを平方すれば式(10)が求められる．

(2) 各行または各列の超立方体結合による配線領域の幅は，mが偶数のとき式(12)に等しい．mが奇数のとき

$$2^2+[(2^4+1)+(2^6+1)+\cdots+(2^{m-1}+1)]=2\cdot2^m/3+m/2-17/6 \tag{15}$$

式(12)，(15)を用いて式(11)が得られる． □

定理1(2)の反射2進 Cubemat のレイアウトは，mが偶数の場合と奇数の場合の間に拡張の互換性がないが，(1)のレイアウトより面積が効率的である．

補題1と定理1より，単純 Cubemat と反射2進 Cubemat のレイアウト面積の間に次の関係が成り立つ．

[定理2] $n=2^m\times2^m$ ノードの単純 Cubemat と反射2進 Cubemat の2次元レイアウト面積 A_{bin}, A_{ref} の間には，ノード数nが大きいとき次の関係がある．

(1) 反射2進 Cubemat のレイアウトを下位ビットから対にするとき

$$A_{ref}\simeq\begin{cases}4A_{bin}/9 & (m:\text{偶数})\\ 25A_{bin}/36 & (m:\text{奇数})\end{cases} \tag{16}$$

(2) 上位ビットから対にするとき

$$A_{ref}\simeq4A_{bin}/9 \tag{17}$$

□

定理2より，ノード数が大きいとき，反射2進

Cubemat のレイアウトは単純 Cubemat のレイアウトの半分未満の面積で実現することができる.

3.3 Cubemat と他の結合モデルとの関係

$n=2^m\times 2^m$ ノード超立方体において, $q=2^m$ ノードをもつ部分超立方体は長さ 2^m の線形巡回結合を形成する. これより, Cubemat の各行, 列方向の結合は巡回的な格子結合(トーラス)を包含している. 特に, 反射2進 Cubemat の各ノードは, その東西南北の位置にあるノードと巡回的に1ビットだけ異なるので超立方体結合で接続されており, 反射2進 Cubemat は配列順の格子結合を陽に包含している.

他方, CCC ネットワークは, $n=2^k$ ノード超立方体の各ノードを k ノードに分解し, 各 k ノードを巡回的に結合したものである[8]. Cubemat の超立方体結合のみを考えるとき, $n=2^k$ ノード Cubemat の各ノードを巡回的に k ノードに分解すれば, 2次元レイアウト型の CCC ネットワークが構成できる. この CCC ネットワークは, Preparata-Vuillemin の1次元レイアウト型の CCC ネットワークと同様に, $k\cdot 2^k=n\log_2 n$ 個のノードと結合線が面積 $O(n^2)$ の領域に埋め込まれ, 下界を実現している.

3.4 大域結合の役割

Cubemat の大域結合は入出力容易化, 制御容易化, 問題解法の多様化, 演算の高速化等に役立つ. 演算の高速化については本論文第5章と第6章で詳しく述べる.

Cubemat とその外部とでデータ入出力を行うとき, データの送り方には大域結合を用いる場合と超立方体結合を用いる場合が考えられる. n ノード Cubemat への n データの入力に対しては, 大域結合を用いても超立方体結合を用いても $O(\sqrt{n})$ ステップかかるが, 大域結合を用いると各行または列ごとにデータを1ステップで送ることができ, かつ超立方体結合を用いた場合のように他のノードを中継することなくデータを送れるという長所がある. この意味で大域結合は入出力の容易化を助けている.

送るデータが制御信号であるとき, 大域結合は制御容易化に役立つ.

Cubemat は超立方体結合と大域結合を含んでいるので, 問題の多様な解き方が可能になる. どちらの結合線を用いても計算時間が同じとき, 超立方体結合を用いる解き方, 大域結合を用いる解き方, 状況に応じて用いる結合線の種類を変える解き方が可能である.

4. Cubemat 上でのデータの並べ換え

Cubemat 上で演算を実行するとき, 用いる Cubemat の種類に応じて, 結果の並びが昇順(降順)または反射2進順で求められる. この結果を出力したり, さらに次の演算に利用したりする場合に, 昇順↔反射2進順の並べ換えが必要となる. 本章ではこの操作を超立方体結合を用いて実行する方法を与える.

4.1 反射2進→昇順変換

反射2進 Cubemat の配列 (I,J) のノード $u(I,J)$ は式(9)で表される. 配列 (I,J) にあるノードの昇順表現を $u'(I,J)$ とし, 次のように2進表現する.

$$u'(I,J)=\langle u'^{x},u'^{y}\rangle=[u_{2m-1}'\cdots u_1'u_0'] \quad (18)$$
$$(u_i'\in\{0,1\},\ 0\leqq i\leqq 2m-1)$$

このとき, 式(3)より u と u' の間には次の関係が成り立つ.

$$u_i'=\begin{cases}u_i\oplus u_{i+1}' & (i\neq m-1,2m-1)\\ u_i & (i=m-1,2m-1)\end{cases} \quad (19)$$

反射2進 Cubemat 上で, 反射2進順のデータを昇順に変換するには, ノード u にあるデータをノード u' に送ればよい. 反射2進→昇順変換アルゴリズムは次のようになる.

[アルゴリズム1] $n=2^m\times 2^m$ ノード反射2進 Cubemat 上のデータを, 各行, 列ごとに昇順に変換する操作 transform 1 を**図5**に示す. 図中記号 for…do …od は順序的な操作, for…pardo…odpar は並列的な操作を表す. 命令 exchange (i) は, ノードの2進表現の第 i 桁が異なる隣接ノード間でデータを交換する操作を表す. □

```
    procedure transform1;
    { initialization }
1.  for u ← 0 to 2^2m − 1 pardo
2.      u'_{2m−1} ← u_{2m−1};
3.      u'_{m−1} ← u_{m−1}
    odpar; {u}
    { bit-computation and data-exchange }
4.  for i ← 2m − 2 downto 0 do
5.      if i ≠ m − 1 then
6.          for u ← 0 to 2^2m − 1 pardo
7.              u'_i ← u_i ⊕ u'_{i+1};
8.              if u'_i ≠ u_i then exchange(i)
            odpar {u}
    od. {i}
```

図5 反射2進 Cubemat 上での反射2進→昇順変換
Fig. 5 Data transformation of binary-reflected order into the ascending order on the binary-reflected Cubemat.

列方向と行方向は独立であり，アルゴリズム1のiに関する操作はどちらから始めてもよい．ただし，各方向とも上位桁から実行する必要がある．各ノードでの基本計算時間と超立方体結合による単位通信時間の評価のオーダをそれぞれ$O_{cp}(1)$，$O_{cm}(1)$で表すとき，アルゴリズム1は次のように評価される．

［定理3］ nノード反射2進 Cubemat 上で，アルゴリズム1を用いて反射2進→昇順変換を行うのにかかる時間は$O_{cp}(\log n)+O_{cm}(\log n)$である． □

4.2 昇順→反射2進順変換

単純 Cubemat の配列(I,J)のノード$e(I,J)$は式(5)で与えられる．配列(I,J)にあるノードの反射2進表現を$e'(I,J)$とし，次のように2進表現する．

$$e'(I,J)=\langle e'^{x}, e'^{y}\rangle=[e_{2m-1}'\cdots e_1'e_0'] \quad (20)$$
$$(e_i'\in\{0,1\}, 0\leqq i\leqq 2m-1)$$

このとき，式(2)よりeとe'の間には次の関係が成り立つ．

$$e_i'=\begin{cases} e_i\oplus e_{i+1} & (i\neq m-1, 2m-1)\\ e_i & (i=m-1, 2m-1)\end{cases} \quad (21)$$

単純 Cubemat 上で，昇順のデータを反射2進順に変換するには，ノードeにあるデータをノードe'に送ればよい．

［アルゴリズム2］ 図5の手続き transform 1 において，第1～3，6，8行のu_i, u_i'をそれぞれe_i, e_i'に変更し，第7行を$e_i'\leftarrow e_i\oplus e_{i+1}$で交換した操作を手続き transform 2 とする．transform 2 は単純 Cubemat 上の昇順データを反射2進順に変換する． □

反射2進 Cubemat 上での変換と同様に，単純 Cubemat 上での変換においても列方向と行方向は独立であり，上位桁から行うならばどちらの方向から始めてもよい．

［系1］ nノード単純 Cubemat 上で，アルゴリズム2を用いて昇順→反射2進順変換を行うのにかかる時間は$O_{cp}(\log n)+O_{cm}(\log n)$である． □

5. ノード間のデータ転送

本章では，Cubemat 上のノード間におけるデータ系列の転送を考察する．データ系列の転送は各種処理における基本操作である[15]．

結合モデルの各ノードは固有メモリを所有しているとする．本章では，各ノードのすべての端子は，データを同時に送信または受信，またはある端子で送信しながら他の端子で受信することができるとする．局所結合と大域結合に関しても同時にデータを送受信することができるとする．

5.1 大域結合のない場合

多数のノードより成る結合モデル内の複数の経路について次の定義をする．

［定義1］ 結合モデル内の1ノードからいくつかのノードを経由して目的ノードに到達する複数の経路が途中の辺を共有しないとき，これらの経路は互いに独立であるとよぶ． □

超立方体内の経路について次の性質が成り立つ．

［補題2］ k次元超立方体には，任意の1ノードから高々kノードを経由して他の目的ノードに到達できる互いに独立なk本の経路が存在する．

(略証)

(1) 出発ノードと目的ノードの2進表現の桁がすべて異なる場合

出発ノードから等距離の互いに異なるkノードを選ぶことができ，かつ出発ノードから距離kで目的ノードに到達できるような経路を選ぶことができる．このことより，出発ノードから目的ノードへ$(k-1)$ノードを経由する独立なk本の経路が存在する．

(2) 2ノードの2進表現が1桁だけ等しい場合

出発ノードから等距離の互いに異なるkノードを選ぶことができ，かつ出発ノードから距離$(k-1)$で目的ノードに到達できるような$(k-1)$経路を選ぶことができる．残りの1経路については，距離$(k+1)$で目的ノードに到達するような経路を選ぶことができる．このことより出発ノードから目的ノードへ高々kノードを経由する独立なk本の経路が存在する．

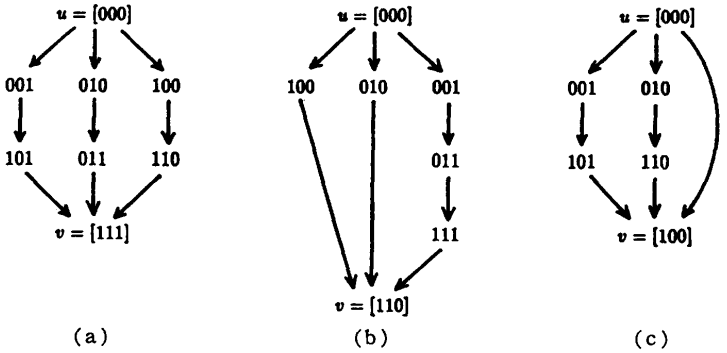
(3) 2ノードの2進表現が2桁以上等しい場合

同様にして，出発ノードから高々$(k-1)$ノードを経由して目的ノードに到達する独立なk経路を選択できる． □

［例1］ u, vをそれぞれ3次元超立方体の出発ノードと目的ノードとする．このとき，ノード$u=$[000]からノードvへ次のような経路を選択できる．まず，2ノードの2進表現がすべて異なるとき，**図6**(a)のような3本の独立な経路が得られる．2ノードが1桁だけ等しいとき，図6(b)のような独立な経路が得られる．また，2ノードが2桁等しいときは，図6(c)のような独立な経路が得られる． □

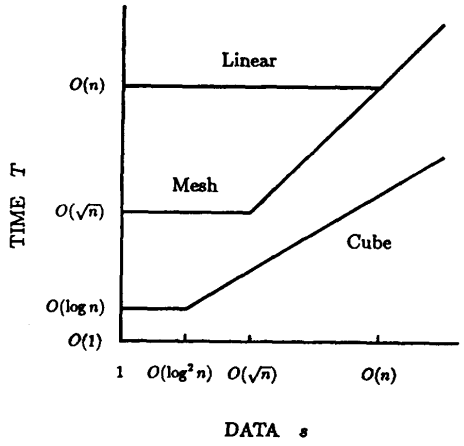
超立方体結合内の2ノード間のデータ転送に関して次のアルゴリズムが成り立つ．

［アルゴリズム3］ k次元超立方体結合内の1ノー

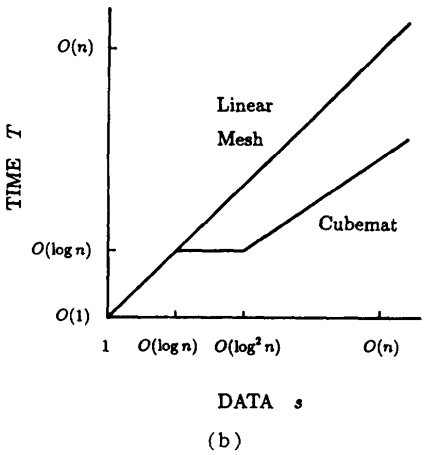


(a) (b) (c)

図 6 互いに独立な複数の経路の例 ($k=3$)
Fig. 6 Examples of independent paths for $k=3$.



(a)



(b)

図 7 2ノード間のデータ転送
(a) 大域結合をもたない場合
(b) 大域結合をもつ場合
Fig. 7 Outlines of time order for data transmission.
(a) The case without global connections.
(b) The case with global connections.

ドから他の目的ノードへ複数データを次のように送る.

(1) 補題2に従って,出発ノードから目的ノードへ独立な k 本の経路を設定する.この経路は一度設定したら以後変更する必要がない.

(2) 出発ノードの k 端子から一度に k データずつストリーム処理で転送する. □

[補題3] n ノード超立方体結合内の2ノード間で,アルゴリズム3に従って s データを転送するのにかかる時間 T は次のようになる.

$$T=O(\log n+s/\log n) \tag{22}$$

記号 O は通信に関する操作の上界を表す.

(証明) 一度に転送するデータは $k=\log_2 n$ 個であり, s データを送るのに必要な回数は $\lceil s/k \rceil$ である.補題2より1データが目的ノードに到達するステップ数は高々 $(k+1)$ であり,各経路ごとに $\lceil s/k \rceil$ データをストリーム処理で送れば,式(22)が得られる. □

n ノード線形結合内の2ノード間で, s データをストリーム処理で送るのにかかる時間は次のようになる.

$$T=O(n+s) \tag{23}$$

また, n ノード格子結合に対しては

$$T=O(\sqrt{n}+s) \tag{24}$$

式(22)～(24)の時間 T とデータ数 s の関係を図示すれば,**図7**(a)のようになる.

5.2 大域結合をもつ構成

大域結合と局所結合より成る結合モデル内の1ノードから目的ノードに s データを送る際,大域結合と局所結合にデータを分割して転送する.大域結合と局所結合で送るデータ数をそれぞれ s_g, s_l とするとき

$$s=s_g+s_l \tag{25}$$

大域結合と局所結合による転送時間をそれぞれ T_g, T_l で表せば,ある特定のデータ分割に対してかかる転送時間 T' は次のように表される.

$$T'=\max\{T_g, T_l\} \tag{26}$$

大域結合と局所結合を用いて s データを転送するのにかかる時間 T は,分割に関して T' の最小値をとって次のように求められる.

$$T=\min_{s_g} T' \tag{27}$$

2次元レイアウトした超立方体の各行,列方向に大

域結合を付加した構成は Cubemat にほかならない．2ノード間のデータ転送時間は，超立方体のレイアウトの仕方や大域結合の次元が1次元か2次元かには依存しない．大域結合をもつ超立方体上でのデータ転送について次のアルゴリズムが成り立つ．

［アルゴリズム4］ 大域結合と k 次元超立方体結合より成る結合モデル上の2ノード間で，式(27)を満たすような分割で s データを次のように転送する．

（1） 大域結合

（a） 出発ノードから目的ノードへ独立な経路を設定する．

（b） s_g データをストリーム処理で送る．

（2） 超立方体結合

（a） 出発ノードから目的ノードへ独立な k 本の経路を設定する．

（b） 出発ノードの k 端子から一度に k データずつ $s_l=s-s_g$ データをストリーム処理で送る． □

このアルゴリズムは次のように評価される．

［定理4］ アルゴリズム4を用いて，大域結合と超立方体より成る n ノード結合モデル上で，1ノードから目的ノードに s データを転送するのにかかる時間 T は次のように表される．

$$T=\begin{cases}O(s) & (s<O(\log n))\\ O(\log n+s/\log n) & (s\geqq O(\log n))\end{cases} \tag{28}$$

この T の値を実現するのは，$s<O(\log n)$ ではすべてのデータを大域結合で送る場合であり，$s=O(\log n)$ では任意の分割に対して成り立つ．また，$s>O(\log n)$ では主として超立方体結合で送り，そのデータ数が s のオーダである場合である．

（略証） s データを分割して，大域結合と超立方体結合でそれぞれ s_g, s_l データを転送するのにかかる時間 T_g, T_l は

$$T_g=O(s_g)$$

$$T_l=\begin{cases}O(\log n+s_l/\log n) & (s_l\neq 0)\\ 0 & (s_l=0)\end{cases} \tag{29}$$

これに式(26)，(27)を適用すれば式(28)が得られる． □

大域結合をもつ n ノード線形結合と格子結合内で，1ノードから目的ノードに s データを転送するのにかかる時間 T は，ともに s の全領域で

$$T=O(s) \tag{30}$$

線形結合と大域結合より成る構成において，式(30)の T の値を実現するのは，$s<O(n)$ ではすべてのデータを大域結合で送る場合であり，$s\geqq O(n)$ では任意の分割に対して成り立つ．他方格子結合と大域結合より成る構成において，式(30)の T の値を実現するのは，$s<O(\sqrt{n})$ ではすべてのデータを大域結合で送る場合であり，$s\geqq O(\sqrt{n})$ では任意の分割に対して成り立つ．式(28)，(30)の関係を図示すれば図7（b）のようになる．

2ノード間のデータ転送を結合モデル間で比較すれば次のようになる．

［定理5］ 大域結合をもつ場合ともたない場合の線形，格子，超立方体結合内の2ノード間で s データを転送する操作は，$s\leqq O(\log n)$ では大域結合をもつ3構成が最も速く，$s>O(\log n)$ では大域結合をもつ超立方体と大域結合をもたない超立方体が最も速い．任意のデータ数 s に対してつねに最も速くデータ転送できる構成は大域結合をもつ超立方体結合（Cubemat）である． □

大域結合と超立方体結合より成る結合モデルの1ノードから多ノードへのデータ系列の転送も同様に議論できる．

6. 動的計画法を用いた行列積の最小コストの計算

n データに対する動的計画法は順序的に $O(n^3)$ 時間かかるが，並列計算可能であり[29]，大域結合を用いれば $O(n)$ 時間で解けることがこれまで指摘されていた[30]．本章では，$q=\sqrt{n}$ 個の行列 $M_0, M_1, \cdots, M_{q-1}$ の積の最小コストを n ノード結合モデル上で動的計画法を用いて求める方法を与える．

行列 M_i は $r_i\times r_{i+1}$ 次行列であり，$r_i\times r_k$ 次行列と $r_k\times r_j$ 次行列の積を求めるには $r_ir_kr_j$ 回の演算が必要であるとする．動的計画法を用いて行列積 $M_I\times M_{I+1}\times\cdots\times M_J$ を計算するための最小コストを m_{IJ} とするとき，m_{IJ} は次のように表される[31]．

$$m_{IJ}=\begin{cases}0 & (I=J)\\ \min\limits_{I\leq k<J}(m_{Ik}+m_{k+1,J}+r_Ir_{k+1}r_{J+1}) & (I<J)\end{cases} \tag{31}$$

いま，

$$p=k-I+1,\quad l=J-I \tag{32}$$

とおけば，$I<J$ のとき，式(31)は次のように表される．

$$m_{IJ}=\min_{1\leq p\leq l} m_{IJ}^{(p)} \tag{33}$$

ここに

$$m_{IJ}^{(p)}=m_{I,I+p-1}+m_{I+p,J}+r_I r_{I+p} r_{J+1} \quad (34)$$

$n=q\times q$ 個のノード配列 (I,J) $(0\leqq I,J\leqq q-1)$ をもつ結合モデル上で，動的計画法を用いた行列積の最小コストを計算する．係数 m_{IJ} をノード (I,J) $(I\leqq J)$ で計算し，最終的な結果をノード $(0,q-1)$ の係数 $m_{0,q-1}$ で与える．ノード配列 I，J をそれぞれ水平右方向と垂直下方向にとれば，係数 m_{IJ} は2次元配列の対角線とその左下半分のノードを用いて求められる．本章では，各ノードは高々 q 個のデータを保持できるとする．行列積の最小コストを求めるアルゴリズムの大要は次のようになる．

［アルゴリズムの大要］

（1） ノード (I,J) $(0\leqq I,J\leqq q-1)$ で m_{IJ} を初期化する．

$$m_{IJ}\leftarrow\begin{cases}0 & (I=J)\\ L & (I<J)\end{cases} \quad (35)$$

ここに L は十分大きい数とする．

（2） 各ノード (I,J) $(J\geqq I+1)$ で積のコスト $r_I r_{I+p} r_{J+1}$ $(1\leqq p\leqq l)$ を順次計算し，これを $m_{IJ}^{(p)}$ とおく．

$$m_{IJ}^{(p)}\leftarrow r_I r_{I+p} r_{J+1} \quad (36)$$

（3） ノード $(I,I+i)$ $(i\geqq 0)$ は $m_{I,I+i}$ を垂直方向に送る．

（4） ノード (I,J) $(J\geqq I+i+1)$ は $m_{I,I+i}$ を取り込み，次の和を実行する．

$$m_{IJ}^{(i+1)}\leftarrow m_{IJ}^{(i+1)}+m_{I,I+i} \quad (37)$$

もし項 $m_{IJ}^{(i+1)}$ が完成したならば，m_{IJ} と比較し，小さい方を m_{IJ} とする．

$$m_{IJ}\leftarrow\min\{m_{IJ},m_{IJ}^{(i+1)}\} \quad (38)$$

（5） ノード $(J-i,J)$ $(i\geqq 0)$ は $m_{J-i,J}$ を水平方向に送る．

（6） ノード (I,J) $(J\geqq I+i+1)$ は $m_{J-i,J}$ を取り込み，次の和を実行する．

$$m_{IJ}^{(I-i)}\leftarrow m_{IJ}^{(I-i)}+m_{J-i,J} \quad (39)$$

もし項 $m_{IJ}^{(I-i)}$ が完成したならば，m_{IJ} と比較し，小さい方を m_{IJ} とする．

$$m_{IJ}\leftarrow\min\{m_{IJ},m_{IJ}^{(I-i)}\} \quad (40)$$

（7） 操作（3）～（6）を $i=0,1,\cdots,q-2$ に対して繰り返す． □

格子結合と大域結合より成る結合モデル上での操作は次のようになる．

［アルゴリズム5］ 格子結合と大域結合より成る $n=q\times q$ ノードの結合モデル上で，$q=\sqrt{n}$ 個の行列 $M_0,M_1,\cdots,M_{q-1}$ の積の最小コストを動的計画法を用いて求める．

入力：行列 M_i の次数 r_i と r_{i+1} $(0\leqq i\leqq q-1)$．

出力：$r_i\times r_k$ 次行列と $r_k\times r_j$ 次行列の積には $r_i r_k r_j$ 回の演算が必要であるとして，q 個の行列の積 $M_0\times M_1\times\cdots\times M_{q-1}$ の最小コスト．

方法：図8の手続き DPmatrix (q,r_i) による．図8において，第1～4行は係数 m_{IJ} の初期化であり，第5～7行は各行列積のコストである．第8～18行が動的計画法の本体であり，そのうち第9～13行は垂直方向の操作，第14～18行は水平方向の操作である．第10，15行での係数 m_{IJ} の垂直，水平方向への転送には大域結合を用いる．第12，17行の条件は項 $m_{IJ}^{(p)}$ が完成したかどうかを示す． □

アルゴリズム5は次のように評価される．

［定理6］ 格子結合と大域結合より成る $n=q\times q$ ノードの結合モデル上で，q 個の行列 $M_0,M_1,\cdots,$

```
    procedure DPmatrix(q, r_i);
    { initialization }
1.  for I ← 0 to q − 1 pardo
2.      for J ← I to q − 1 pardo
3.          if J = I then m_IJ ← 0;
4.                   else m_IJ ← L;
            { each cost }
5.          l ← J − I;
6.          for p ← 1 to l do
7.              m_IJ^(p) ← r_I r_{I+p} r_{J+1}
            od; {p}
            { body of DP }
8.          for i ← 0 to q − 2 do
                { vertical operations }
9.              s ← 2i + 1;
10.             node (I, I + i) transmits m_{I,I+i} through vertical bus;
11.             m_IJ^(i+1) ← m_IJ^(i+1) + m_{I,I+i} at node (I, J) (J ≥ I + i + 1);
12.             if (s ≤ q − 1 and I + i + 1 ≤ J ≤ I + 2i)
                        or (q ≤ s ≤ 2q − 2 and I + i + 1 ≤ J)
13.                 then m_IJ ← min{m_IJ, m_IJ^(i+1)} at node (I, J);
                { horizontal operations }
14.             s ← 2i + 2;
15.             node (J − i, J) transmits m_{J−i,J} through horizontal bus;
16.             m_IJ^(I−i) ← m_IJ^(I−i) + m_{J−i,J} at node (I, J) (J ≥ I + i + 1);
17.             if (s ≤ q − 1 and I + i + 1 ≤ J ≤ I + 2i + 1)
                        or (q ≤ s ≤ 2q − 2 and I + i + 1 ≤ J)
18.                 then m_IJ ← min{m_IJ, m_IJ^(I−i)} at node (I, J);
            od {i}
        odpar {J}
    odpar. {I}
```

図8 行列積の最小コストを計算する手続き

Fig. 8 A procedure to compute the minimum cost of matrix product.

M_{q-1} の積の最小コストを求めるアルゴリズム5は $O(q)=O(\sqrt{n})$ 時間で実行できる．

(証明) 第5～7行の各行列積の計算はノード (0, $q-1$) で最も時間がかかり，$O(q)$ 時間である．第8～18行は各ノードとも並列に $O(q)$ 時間で実行できる． □

大域結合をもたない $n=q\times q$ ノードの格子結合上で，q 個の行列 $M_0, M_1, \cdots, M_{q-1}$ の積の最小コストはアルゴリズム5と同様に求められる．ただし，手続き DPmatrix (q, r_i) で第10，15行の係数 m_{IJ} の垂直，水平方向への転送には格子結合を用いる．このとき，行列式の最小コストは $O(q^2)=O(n)$ 時間で求められる．

次に，2次元レイアウトした超立方体結合上で行列積の最小コストを求める．

[アルゴリズム6] 2方向反射2進順にレイアウトした $n=q\times q$ ノードの超立方体結合上で，q 個の行列 $M_0, M_1, \cdots, M_{q-1}$ の積の最小コストを動的計画法を用いて求める．入出力と方法はアルゴリズム5と同様である．ただし，手続き DPmatrix (q, r_i) で第10，15行の係数 m_{IJ} の垂直，水平方向への転送には超立方体結合を用いる． □

[補題4] 2方向反射2進順にレイアウトした $n=q\times q$ ノードの超立方体結合上で，q 個の行列 $M_0, M_1, \cdots, M_{q-1}$ の積の最小コストは，アルゴリズム6を用いて $O(q \log q)$ 時間で求められる． □

結合モデル Cubemat 上での行列積の最小コストは次のように求められる．

[アルゴリズム7] $n=q\times q$ ノード反射2進 Cubemat 上で，q 個の行列 $M_0, M_1, \cdots, M_{q-1}$ の積の最小コストを動的計画法を用いて求める．入出力と方法はアルゴリズム5と同様である． □

[定理7] $n=q\times q$ ノード反射2進 Cubemat 上で，q 個の行列 $M_0, M_1, \cdots, M_{q-1}$ の積の最小コストはアルゴリズム7を用いて $O(q)=O(\sqrt{n})$ 時間で求められる． □

定理6，定理7と補題4は，行列積の最小コストを動的計画法で求めるとき，2次元大域結合の方が超立方体結合よりも有効であることを示している．

7. おわりに

超立方体結合と大域結合より成る結合モデル Cubemat を導入し，その配線面積を節約する効率的な2次元レイアウト法を示した．ノード数が大きいとき，超立方体結合のノードを2方向反射2進順に配置したレイアウト面積は，昇順に配置したレイアウト面積の約4/9で実現できる．また，Cubemat 上での1ノードから他のノードへのデータ系列の転送は，超立方体結合の各端子で同時に送受信できる場合でも，なおデータ数の少ない領域で大域結合の方が速いことを導いた．さらに，行列積の最小コストを動的計画法で求めるとき，超立方体結合よりも大域結合を用いた方が速く解ける方法を示した．また，大域結合よりも超立方体結合を用いた方が有利な例として，本論文では昇順↔反射2進順のデータの並べ換えを与えた．

超立方体結合と大域結合を用いて問題を解くとき，超立方体結合の方が大域結合よりも速い場合，その逆の場合，計算時間が同じ場合がある．Cubemat は超立方体結合と大域結合より成る結合モデルであるので，その速い方に等しいかそれよりも速く問題を解くことができる．超立方体結合と大域結合は，それぞれ単一で用いても高い計算・通信能力をもっている結合モデルであるので，Cubemat はさらに強力な結合モデルであると予想できる．今後の課題としては，この結合モデルの計算能力の範囲をさらに詳しく調べることである．

謝辞 本研究に関してご討論頂いた京都大学矢島脩三教授，安浦寛人助教授に深謝いたします．また，有益なご指摘を頂いた査読者に感謝します．

参考文献

1) Ullman, J. D.: *Computational Aspects of VLSI*, Computer Science Press, Maryland (1984).
2) Hillis, W. D.: *The Connection Machine*, The MIT Press, Massachusetts (1985).
3) Fox, G. C., Johnson, M. A., Lyzenga, G. A., Otto, S. W., Salmon, J. K. and Walker, D. W.: *Solving Problems on Concurrent Processors*, Vol. I, Prentice-Hall, New Jersey (1988).
4) Seitz, C. L.: The Cosmic Cube, *Comm. ACM*, Vol. 23, No. 1, pp. 22-33 (1985).
5) Johnsson, S. L.: Communication Efficient Basic Linear Algebra Computations on Hypercube Architectures, *J. Parallel and Distrib. Comput.*, Vol. 4, pp. 133-172 (1987).
6) Ranade, A. G. and Johnsson, S. L.: The Communication Efficiency of Meshes, Boolean Cubes and Cube Connected Cycles for Wafer Scale Integration, *Proc. Int. Conf. Parallel Processing*, pp. 479-482 (1987).
7) Saad, Y. and Schultz, M. H.: Topological

Properties of Hypercubes, *IEEE Trans. Comput.*, Vol. C-37, No. 7, pp. 867-872 (1988).
8) Preparata, F. P. and Vuillemin, J.: The Cube-Connected Cycles: A Versatile Network for Parallel Computation, *Comm. ACM*, Vol. 24, No. 5, pp. 300-309 (1981).
9) Harary, F. (池田貞雄訳): グラフ理論, 共立出版, 東京 (1971).
10) Hwang, K. and Briggs, F. A.: *Computer Architecture and Parallel Processing*, McGraw-Hill, New York (1984).
11) 富田真治: 並列計算機構成論, 昭晃堂, 東京 (1986).
12) 小池誠彦: 超並列マシン, 情報処理, Vol. 28, No. 1, pp. 94-105 (1987).
13) Shibusawa, S.: A Routing Method for A Class of Cyclic Permutation Network, *Trans. IEICE*, Vol. E-72, No. 2, pp. 130-140 (1989).
14) Ho, C.-T. and Johnsson, S. J.: Distributed Routing Algorithms for Broadcasting and Personalized Communication in Hypercubes, *Proc. Int. Conf. Parallel Processing*, pp. 640-648 (1986).
15) Saad, Y. and Schultz, M. H.: Data Communication in Hypercubes, *J. Parallel and Distrib. Comput.*, Vol. 6, No. 1, pp. 115-135 (1989).
16) Thompson, C. D.: A Complexity Theory for VLSI, Ph. D. Dissertation, Dept. of Computer Science, Carnegie-Mellon Univ. (1980).
17) Leiserson, C. E.: *Area-Efficient VLSI Computation*, The MIT Press, Massachusetts (1983).
18) 都倉信樹, 萩原兼一, 和田幸一: VLSI モデルと面積複雑度, 情報処理, Vol. 26, No. 6, pp. 583-592 (1985).
19) Reingold, E. M., Nievergelt, J. and Deo, N.: *Combinatorial Algorithms: Theory and Practice*, Chapter 5, Prentice-Hall, New Jersey (1977).
20) 矢島脩三: 計算機の機能と構造, 岩波書店, 東京 (1982).
21) Stout, Q. F.: Mesh-Connected Computers with Broadcasting, *IEEE Trans. Comput.*, Vol. C-32, No. 9, pp. 826-830 (1983).
22) Aggrawal, A.: Optimal Bounds for Finding Maximum on Array of Processors, *IEEE Trans. Comput.*, Vol. C-35, No. 1, pp. 62-64 (1986).
23) Prasanna Kumar, V. K. and Raghavendra, C. S.: Array Processor with Multiple Broadcasting, *J. Parallel and Distrib. Comput.*, Vol. 4, pp. 173-190 (1987).
24) 梅尾博司, Worsch, T. and Vollmar, R.: グローバル・バスを利用した並列アルゴリズムについて, 情報処理学会研究会資料, 88-AL-4-7 (1988).
25) 石川 勉: 通信特性向上のためのバス結合付加型ハイパーキューブアーキテクチャ, 電子情報通信学会論文誌, Vol. J 73-D-I, No. 4, pp. 415-423 (1990).
26) 渋沢 進: VLSI 向き並列処理システム Cubemat, 電子情報通信学会技術報告, COMP88-39 (1988).
27) 渋沢 進: 並列計算モデル Cubemat 上でのいくつかのデータ転送操作, 並列処理シンポジウム JSPP '90 論文集, T1-3, pp. 17-24 (1990).
28) 渋沢 進: CCC ネットワークのレイアウトの一評価, 電子情報通信学会論文誌, Vol. J 71-D, No. 7, pp. 1349-1353 (1988).
29) 瀬口靖幸, 田中正夫, 中島利朗, 金田悠紀夫, 小畑正貴, 西野佐登史: 動的計画法の並列計算—並列計算性とアルゴリズム, 情報処理学会論文誌, Vol. 26, No. 5, pp. 824-830 (1985).
30) Li, G. and Wah, B. W.: Systolic Processing for Dynamic Programming Problems, *Proc. Int. Conf. Parallel Processing*, pp. 434-441 (1985).
31) Aho, A. V., Hopcroft, J. E. and Ullman, J. D. (野崎昭弘, 野下浩平共訳): アルゴリズムの設計と解析, サイエンス社, 東京 (1977).

(平成元年8月31日受付)
(平成3年1月11日採録)

渋沢 進 (正会員)

昭和25年生. 昭和49年新潟大学理学部物理学科卒業. 昭和51年同大学院修士課程修了. 同年群馬大学工学部情報工学科助手. 工学博士. ハードウェアシステム, 並列計算の研究に従事. 電子情報通信学会, 日本ソフトウェア科学会, IEEE 各会員.