解　説

動的可変バスをもつ並列計算機上の定数時間アルゴリズム

Constant-Time Algorithms on the Reconfigurable Mesh by Koji NAKANO (Department of Electrical and Computer Engineering, Nagoya Institute of Technology).

中　野　浩　嗣[1]

1 名古屋工業大学電気情報工学科

1. まえがき

これまでに多くの並列アルゴリズムの研究が，共有メモリ型並列計算機（PRAM, Parallel Random Access Machine）や超立方体結合，メッシュ結合などのネットワーク型並列計算機を対象として行われてきた．ネットワーク型並列計算機では，プロセッサが送信したデータは，通信用リンクを介して転送され，このリンクの接続先のプロセッサが受信する．よって，受信プロセッサが送信時に決まっているという意味で通信路が固定されている．また，PRAM☆の場合は，各プロセッサが，共有メモリへの書き込みの対象となるメモリセルのアドレスを書き込み時に指定するという意味で，通信路が固定されていると考えられる．この通信路の固定のために，これらの並列計算機では，計算能力に限界があった．たとえば，n ビットの入力を 1 ビットずつプロセッサに与えたときに，その 1 の数を求めるといったような簡単な計算ですら，これらの並列計算機では高速に行うことはできないことが証明されている[2]．

ところが，通信路を動的に変化させることができるという仮定をおくと，1 の数を求める問題も含めたさまざまな問題が高速に解くことができる．この通信路を動的に変化させるバスをもつ並列計算機を理論的にモデル化したのが再構成メッシュである．かなり昔から，このような動的可変バスについての研究はあったが，Miller らが 1988 年に発表した論文[7]をきっかけにして，多くの研究者が注目し，さまざまな並列アルゴリズムが考案されてきた．本稿では，再構成メッシュ上の代表的な定数時間並列アルゴリズムについて紹介する．また，計算能力に関して，PRAM との比較を行うことにより，再構成メッシュが定数時間で解ける問題に対する示唆を与える．

☆ 本稿では，PRAM に関して，CRCW モデル（同一メモリセルへの同時読み出し・同時書き込みを許す）を仮定する．

2. 再構成メッシュ

再構成メッシュは，大きさ $m \times n$ の 2 次元格子状に並べたプロセッサとそれらを接続する通信用リンクにより構成させる並列計算機である（図-1）．各プロセッサは 4 つの通信ポート（N, E, W, S と表される）をもち，これらを介して上下左右に隣接したプロセッサと接続する．2 次元格子上のプロセッサを $P_{i,j}(1 \le i \le m, 1 \le j \le n)$ と表す．プロセッサは完全に同期して動作し同一のプログラムを実行する SIMD 型であるが，入力データや 2 次元格子における座標に依存して，異なる動作をさせることもできる．通常のメッシュ結合計算機と異なり，各プロセッサはそのプログラムによる制御に従って内部的に 4 つのポートを通信用リンクで接続したり切り離したりすることができる．つまり，図-1 に示した 15 とおりの接続パターンから 1 つを選択することができる．その結果，プロセッサ間の通信用リンクとプロセッサ内部の接続による連結成分はサブバスを構成する．プロセッサはこのサブバスを介したデータの放送を行うことができる．再構成メッシュでは，プロセッサの能力やサブバスの通信能力に関するさまざまな仮定が行われる．ここでは，最もよく使われる次の仮定を採用する．

- ワードモデル：プロセッサは $O(\log n)$ ビットのワードデータに対する算術演算を含めた操作が 1 単位時間で行える．

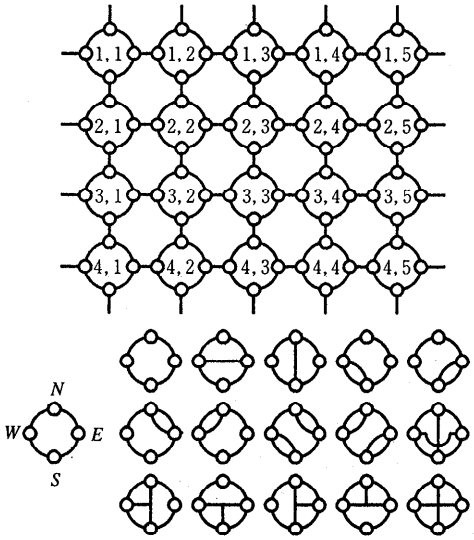


図-1 プロセッサ台数4×5の再構成メッシュとプロセッサ内部の接続パターン

• 定数時間遅延モデル：サブバスを介した$O(\log n)$ビットのワードデータの放送は1単位時間で完了する．

ポートや接続パターンを設定するための回路での遅延を考えると定数時間遅延モデルは現実的でない仮定である．しかし，将来実現されるであろう光バスと，光を反射し経路を遅延なく動的に変えられる鏡の役割をするようなデバイスがあれば，妥当な仮定と考えられる．

プロセッサ内部の接続パターンの制限に関してもさまざまな仮定が考えられている．たとえば，

• 通信の減衰を考慮し，サブバスが分岐するような接続は許さない．たとえば，$N \cdot E \cdot W$の3つのポートを1つに接続することは許さない．

• ハードウェアの複雑さを考慮し，$N \cdot S$と$E \cdot W$を独立に接続する交差接続は許さない．

などの制限が仮定されることがある．ここでは，接続パターンを制限せずに15とおりのすべての接続パターンが設定可能であると仮定する．

3. 再構成メッシュ上の並列アルゴリズム

3.1 算術演算アルゴリズム

nビットの入力が与えられたとき，その中に現れる1つの数を求める問題を考える．入力は，各列に1ビットずつ与えられるものとする．このとき，$(n+1)\times n$台のプロセッサからなる再構成メッシュで定数時間で1の数を求めることができる[10]．まず，入力ビットが0の列のすべてのプロセッサは，左右のポート$E \cdot W$を接続する(**図-2**参照)．入力ビットが1である列のすべてのプロセッサは，ポート$N \cdot E$とポート$S \cdot W$を独立に接続する．そして，プロセッサ$P_{1,1}$は，ポートWにデータ(たとえば1)を送信する．すると，入力ビットが1の列のところで，データが1つ下の行に移るので，右端の列のどのプロセッサにデータが送信されるかにより1の数を判定することができる．定数時間遅延モデルなので，以上の処理に要する時間は定数時間である．よって，

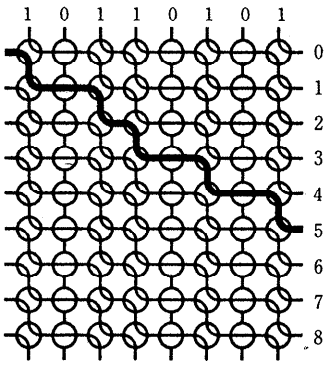


図-2 8ビットの入力の1の数を求める

定理1 nビットの入力が与えられたとき，その中の1の数は$(n+1)\times n$の再構成メッシュ上で定数時間で求められる．

計算能力がきわめて高いPRAMでさえ1の数を求めるのに$\Omega(\log n / \log\log n)$時間必要であることが知られている[2]．(ただし，プロセッサ台数，または，メモリセル数がnの多項式の場合)．よって，定理1より，再構成メッシュが従来の並列計算モデルにない計算能力をもっているといえる．

次に，n個のdビット整数の合計を求めるより一般的な場合を考える．これには，先読みキャリー生成器(lookahead carry generator)を再構成メッシュ上で模倣する手法を用いる[5](**図-3**)．合計値の各桁の値を求めるために，先読みキャリー生成器を各桁に1つずつ割り当てる．1つの先読みキャリー生成器は$2n \times 2n$で構成され，下位桁からのキャリーを右側から受け取り，上位桁へのキャリーを左側へ出力する．図-3の場合，行0，1，2，3に信号(値は何でもよく，たとえば1)が入力されているので，下位桁からのキャリー

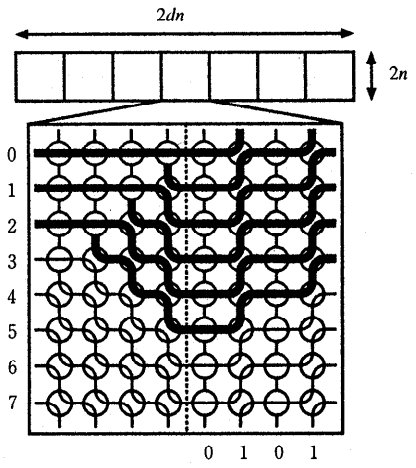


図-3 先読みキャリー生成器を用いた整数加算

は3であると考える．先読みキャリー生成器の右半分では，それが担当する桁の入力に現れる1の数を加算する．この加算は，定理1と同様に行える．図-3の場合，0，1，0，1を加算し，合計は5である．この5は奇数なので，合計値のこの桁の値は1である．もし，偶数ならば，0である．そして，左半分では，5を2で割った商の2を出力する．図-3の場合，行0，1，2に信号が出力される．この先読みキャリー生成器を図-3のようにd個並べることによりの合計値の各桁の値を求めることができる．右端が最下位桁であり，順に左が上位桁に対応する．このd個の先読みキャリー生成器に与えられた入力に対して，定理1と同様に，各プロセッサがサブバスを独立かつ同時に設定した後，下位桁からの1回の信号の放送で，全桁の合計値を求めることができる．したがって，次の定理が成り立つ．

定理2 n個のdビット整数の合計は，$2n \times 2dn$の再構成メッシュで定数時間で求められる．

3.2 ソーティングアルゴリズム

n個のデータ$a_1, a_2, \cdots, a_n$のソーティングは再構成メッシュ上で定数時間で行える．簡単のため，すべての入力データは相違なると仮定する．まず，大きさ$n \times n$の大きさの総当たり表Aを作る．つまり，Aの第(i, j)要素$A_{i,j}$は，$a_i > a_j$のとき1，$a_i < a_j$のとき0とする．すると，$A_{i,1}$, $A_{i,2}$,..., $A_{i,n}$に現れる1の数を求めれば，a_iが何番目に小さいデータか求めることができる．あとは，単純なルーティングでデータの並び換えをすればよい．定理1より，各a_iに対して，$n \times n$台のプロセッサがあれば，$A_{i,1}, A_{i,2}, \ldots, A_{i,n}$に現れる1の数を定数時間で求められる(1の数は高々n-1なので，$n \times n$台で十分であることに注意せよ)．したがって，プロセッサ台数$n^2 \times n$の再構成メッシュでソーティングが定数時間で行える．

さらに，Columnsortと呼ばれるソーティングの手法[6]を用いれば，プロセッサ台数を$n \times n$に減らしても定数時間でソーティングが行える[3]．Columnsortは，入力データを$\sqrt{n}$個ずつに$\sqrt{n}$のグループに分け，

Step 1 グループごとのソーティング

Step 2 あらかじめ決まっているデータの置換

を交互に定数回繰り返せばソーティングが完了することを保証している．ここで，あらかじめ決まっているデータの置換とは，たとえば，「各i番目($(1 \le 1 \le n)$)のデータを$((i+\sqrt{n}) \bmod n)+1$番目に移動する」というような置換で，データの中身や大小にまったく依存しないものである．上に述べたソーティングアルゴリズムにより，1グループあたり$n \times \sqrt{n}$個のプロセッサがあればグループ内のソーティングが定数時間で行える．グループ数は，$\sqrt{n}$なので，全体で$n \times n$個のプロセッサがあればStep 1は定数時間で行える．どのようなデータの置換もあらかじめ決まっていれば，$n \times n$の再構成メッシュで定数時間で行える．たとえば，前の例の場合，各$P_{1,i}$がもつデータを$P_{1,(i+\sqrt{n})+1}$へ送信する必要があるが，$P_{1,i} \to P_{i,i} \to P_{i,(i+\sqrt{n})+1} \to P_{1,(i+\sqrt{n})+1}$の順にデータを転送すればよい．この転送はすべての$i$について同時に行えるので，Step 2も定数時間で完了できる．したがって，次の定理が成り立つ．

定理3 n個のデータのソーティングは，$n \times n$の再構成メッシュで定数時間で行える．

n個のデータのソーティングは，最悪時にはすべてのデータを入れ換える必要がある．したがって，定数時間でソーティングを行うためには，再構成メッシュは$n \times n$より小さくできない．よって，定理3は最適なアルゴリズムといえる．

3.3 グラフの到達可能頂点問題

グラフ$G = (V, E)$(Vは頂点集合，Eは辺集合)と1つの頂点$v \in V$が与えられたときに，uから到達可能な頂点をすべて求める問題を考える．簡

単のために $V=\{1,2,...,n\}$ とする．辺集合が大きさ $n \times e$ の再構成メッシュの各列に1つずつ与えられているものとする．到達可能な頂点は，次の方法でみつけることができる．大きさ $n \times e$ の再構成メッシュの列 $i(1 \le i \le n)$ に辺 (u_i, v_i) が与えられているものとする．アルゴリズムでは，再構成メッシュの各行 $j(1 \le j \le n)$ を頂点 j に対応させて考える．辺 (u_i, v_i) に対して Pu_i,i と Pv_i,i は4つのポート $N \cdot E \cdot W \cdot S$ を1つに接続する．ほかのプロセッサは $N \cdot S$ と $E \cdot W$ を独立に接続する(図-4参照)．この接続により，再構成メッシュの行 u_i と行 v_i にある水平なサブバスが列 i の垂直なサブバスで接続されていることになる．そして，$P_{u,1}$ がポート W に信号を送信する．すると，頂点 $v \in V$ が頂点 u から到達可能であるときのみ，かつそのときのみ $P_{v,1}$ がポート W から信号を受信することができる．よって，頂点 u と同じ連結成分に属するすべての頂点をみつけることができる．したがって，次の定理が成り立つ．

定理4 頂点数 n，辺数 e のグラフの到達可能頂点問題は大きさ $n \times e$ の再構成メッシュで定数時間を求めることができる．

3.4 最小全域木問題

各辺に重みが与えられたグラフの最小全域木とは，重みの合計が最小になるようなすべての頂点を含む部分グラフである．従来の並列アルゴリズムでは，部分最小全域木の合併を繰り返す手法が用いられる．再構成メッシュ上では，定数時間を達成するためにまったく異なる手法を用い，Kruskalの逐次アルゴリズム[1]を再構成メッシュにインプリメントする．Kruskalのアルゴリズムは，辺集合から重みの小さい順に辺を取り出し，その辺を中間最小木に追加しても閉路ができないならば，その辺を追加する．最終的に得られる中間最小木が最小全域木となる．これを再構成メッシュ上では次のように行う．まず，e 個の辺からなる辺集合が与えられたとき，それを重みの小さい順にソーティングを行う．これは，定理3により，$e \times e$ の再構成メッシュで定数時間で行える．いま，辺を重みの小さい順に $(u_1, v_1),(u_2, v_2),...,(u_e, v_e)$ とする．このとき，Kruskalのアルゴリズムより，次のことが成り立つ．各 i について，辺 (u_i, v_i) が最小全域木の辺であるときかつそのときのみ $(u_1, v_1),(u_2, v_2),...,(u_{i-1}, v_{i-1})$ の辺からなるグラフ上で u_i と v_i が同じ連結成分に属している．よって，定理4より，辺 (u_i, v_i) が最小全域木の辺か否かは，$n \times e$ の再構成メッシュで定数時間で判定できる．これをすべての辺に対して同時に実行すればよいので，次の定理が成り立つ．

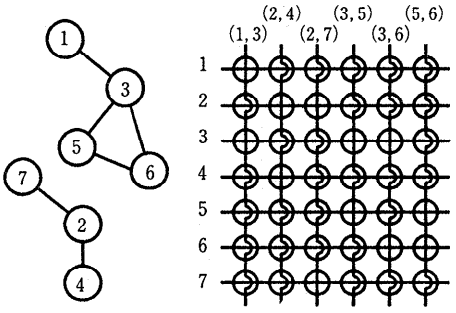


図-4 到達可能頂点問題のためのサブバス配置

定理5 頂点数 n，辺数 e のグラフの最小全域木は，$ne \times e$ の再構成メッシュで定数時間で求めることができる．

グラフ問題については，2連結成分，間接点，橋なども再構成メッシュ上で定数時間で求められることが知られている．

3.5 幾何学アルゴリズム

平面上の n 個の点の凸包は，$n \times n$ の再構成メッシュで求められることを示す．ここで，凸包とは，すべての点を含む最小の単純凸多角形のことである．まず，$n^2 \times n$ の再構成メッシュ上の定数時間アルゴリズムを示す．1つの点 a が凸包の点であるための必要十分条件は，ほかの n-1 個の点のうち任意の2つを通る直線(合計 $(n\text{-}1)(n\text{-}2)/2$ 本)を調べ，a がこれらすべての直線より上にあればよい(図-5)．よって，この判定は $n \times n$ 台のプロセッサがあれば定数時間で行える．したがって，全頂点に対する判定は，$n^2 \times n$ 台のプロセッサがあれば定数時間で行える．下凸包も同様に求められる．

小さな上凸包を求めて，それを合併するという手法を用いれば，プロセッサ台数を $n \times n$ にできる[9]．まず，n 個の頂点を x 座標でソーティングを行う．これは，定理3より，$n \times n$ の再構成メッシュがあれば定数時間で行える．そして，頂点

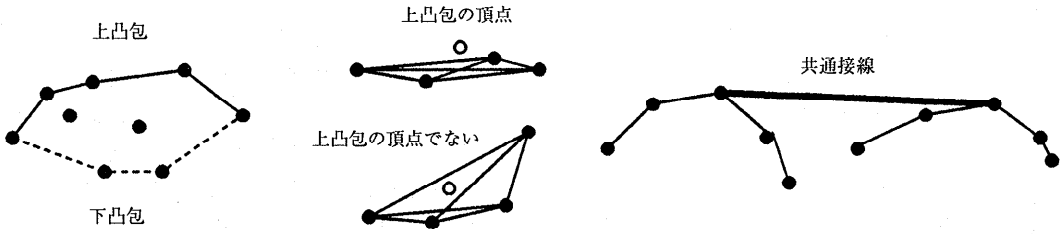


図-5 凸包を求めるアルゴリズム

を x 座標に関して隣接した $\sqrt{n}$ 個の点からなる $\sqrt{n}$ 個のグループに分割する．そしてグループごとに上凸包を求める．各グループに対して，$n \times \sqrt{n}$ 台のプロセッサを割り当てることができるので，これは上の凸包アルゴリズムで定数時間で行える．この結果，$\sqrt{n}$ 個の小さな上凸包が得られたことになる．これらの上凸包を合併するために，すべての2つの上凸包の組みに対して共通接線を求める(図-5)．各上凸包の点数は高々 $\sqrt{n}$ なので，$\sqrt{n} \times \sqrt{n}$ 台のプロセッサを用いれば，2つの上凸包の共通接線を定数時間で求めることができる．2つの上凸包の組合せは $\sqrt{n}(\sqrt{n}-1)/2$ なので，すべての共通接線は $n \times n$ の再構成メッシュがあれば定数時間で求められる．そして，1つの小さな上凸包に関する共通接線で，その上凸包の通接線のうち最も傾きの小さいものを比べることにより，その小さな上凸包の点のうちどの点が最終的に選ぶべき上凸包の点か判定できる．よって，次の定理が成り立つ．

定理6 平面上の n 個の点の凸包は，$n \times n$ の再構成メッシュで定数時間で求められる．

4. 再構成メッシュの計算能力

前章では，さまざまな問題を定数時間で解けるアルゴリズムを紹介した．本章では，並列アルゴリズムの標準的なモデルである PRAM と再構成メッシュの相互の模倣に要する時間により，再構成メッシュが定数時間で解くことのできる問題に対する示唆を与える．PRAM とは，複数のプロセッサが1つのメモリ空間を共有しており，各メモリセルに対して自由にアクセスできる並列計算機モデルである[4]．

まず，n 個のプロセッサ $P_1, P_2,\ldots,P_n$ と n 個の共有メモリセル $M_1, M_2,\ldots,M_n$ からなる PRAM の1単位時間の動作を再構成メッシュで模倣する方法について述べる．再構成メッシュは $n \times n$ 個

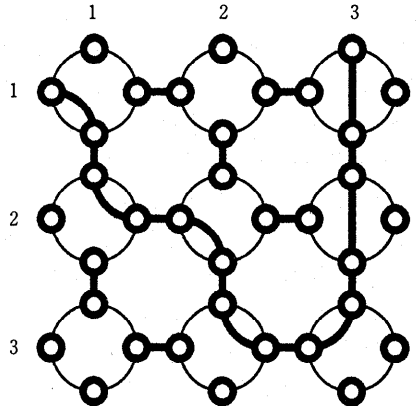


図-6 再構成メッシュのサブバス接続のグラフ化

のプロセッサを用いる．ここで問題となるのは，PRAM の各プロセッサが共有メモリに対して行う読み出しや書き込みの動作を，どのように模倣するかである．そのために，再構成メッシュ上の n 個のプロセッサ $P_{1,1}, P_{1,2},\ldots,P_{1,n}$ を1つずつ PRAM のプロセッサ $P_1, P_2,\ldots,P_n$ を模倣するためにに割り当てる．また，$P_{1,1}, P_{2,1},\ldots,P_{n,1}$ を1つずつ PRAM のメモリセル $M_1, M_2,\ldots,M_n$ の値を保持するために割り当てる．たとえば，P_i がメモリセル M_j への書き込みを模倣するには，$P_{1,i}$ と $P_{j,i}$ を接続する垂直なサブバスを設定し，$P_{j,i}$ に書き込むべきデータを送信する．そして，$P_{j,i}$ と $P_{j,1}$ を接続する水平なサブバスを設定し，$P_{j,1}$ に書き込みデータを転送する．$P_{j,1}$ は受信したデータを書き込みデータとして，M_j を更新する．読み出しも同様に模倣することができる．この模倣はすべてのプロセッサに関して同時に行うことができるので，次の定理が成り立つ．

定理7 n 台のプロセッサ，n 個の共有メモリセルの PRAM の1単位時間の動作は $n \times n$ の再構成メッシュにより定数時間で模倣できる．

逆に $n \times n$ の再構成メッシュの1単位時間の通信を PRAM で模倣することを考える．再構成

メッシュの各プロセッサがもつ4つのポートをグラフの頂点と考える．つまり，頂点集合は

$V = \{\langle i,j,p\rangle \mid 1 \le i,j \le n, p \in \{N,E,W,S\}\}$

である．そして，辺集合 E を

$E = \{(\langle i,j,p\rangle, \langle i',j',p'\rangle) \mid \langle i,j,p\rangle$ と $\langle i',j',p'\rangle$ に対応するポートが内部または外部リンクで直接接続されている$\}$ とする．たとえば，**図-6** の場合，$(\langle 1,1,W\rangle, \langle 1,1,S\rangle), (\langle 1,1,S\rangle, \langle 2,1,N\rangle)$ などが，E に含まれる．すると，1つのサブバスで接続されているポートはグラフ $G=(V,E)$ 上で同じ連結成分に属することになる．よって，このグラフ G に対して連結成分を求めることにより全ポートの接続関係を知ることができる．一般に，グラフ $G=(V,E)$ の連結成分は，PRAM上で $|V|+|E|$ プロセッサ，$O(\log|V|)$ 時間で求められることが知られている[4]．この模倣の場合，$|V|=4n^2, |E|=O(n^2)$ なので，$O(n^2)$ プロセッサ，$O(\log n)$ 時間でポートの接続関係を知ることができる．そして，各連結成分に共有メモリセルを割り当てれば，メモリセルへの読み出し・書き込みにより，サブバスを介した送受信を模倣することができる．

定理8 $n \times n$ の再構成メッシュの1単位時間の動作は，$O(n^2)$ 個のプロセッサ，$O(n^2)$ 個の共有メモリセルからなるPRAMで $O(\log n)$ 時間で模倣できる．

以上より，プロセッサ台数と共有メモリのセル数が問題の入力の大きさ n の多項式 n^k（k は任意の定数）であるという制限のもとで，PRAMで定数時間で解くことができる問題は，すべて再構成メッシュで定数時間で解くことができる．逆に，再構成メッシュで定数時間で解ける問題はPRAMで $O(\log n)$ 時間で解くことができる．よって，PRAMで $O(\log n)$ 以下の時間で解ける問題ならば，再構成メッシュで定数時間で解ける可能性があるといえる．PRAMで $O(\log n)$ 時間より多くの時間（たとえば，$O(\log^2 n)$ 時間）を必要とする問題に対しては，再構成メッシュで定数時間では解けない．もし，再構成メッシュで定数時間で解けたならば，PRAMでも $O(\log n)$ 時間で解けてしまうからである．

5. あとがき

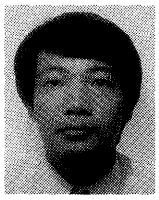
本稿では，再構成メッシュ上の定数時間アルゴリズムを紹介し，また，PRAMとの計算能力の比較を行った．再構成メッシュに関する1995年までの研究成果は文献リスト8）を参照されたい．また，1994年から，IPPS（International Parallel Processing Symposium）のワークショップとしてRAW（Reconfigurable Architecture Workshop）が毎年4月頃行われており，再構成メッシュに関する最新の研究動向を知ることができる．

謝辞 日頃からご議論いただく名古屋工業大学林達也教授に感謝いたします．

参考文献

1) Aho, A.V., Hopcroft, J.E. and Ullman, J.D. : The Design and Analysis of Computer Algorithms, Addison-Wesley (1974).
2) Beame, P. and Hastad, J. : Optimal bounds for Decision Problems on the CRCW PRAM, Journal of the ACM, 36(3), pp.643-670(July 1989).
3) Ben-Asher, Y., Peleg, D., Ramaswami, R. and Schuster, A. : The Power of Reconfiguration, Journal of Parallel and Distributed Computing, 13, pp.139-153(1991).
4) JáJá, J. : An Introduction to Parallel Algorithms, Addison-Wesley (1992).
5) Jang, J.-W. and Prasanna, V.K. : An Optimal Sorting Algorithms on Reconfigurable Mesh, In Proc. 6th International Paralle Processing Sympogiumu, pp.130-137, IEEE(1992).
6) Leighton, F.T. : Tight bounds on the Complexity of Parallel Sorting, In Proc. 16th Symposium on Theory of Computing, pp.71-80, ACM(May 1984).
7) Miller, R., Kumar, V.K.P., Reisis, D. and Stout, Q.F. : Meshes with Reconfigurable buses, Proc. 15th MIT Conference on Advanced Research in VLSI, pp.163-178(Mar. 1988).
8) Nakano, K. : A Bibliography of Published Papers on Dynamically Reconfigurable Architectures, Parallel Processing Letters, 5(1), pp.111-124(1995).
9) Nigam, M. and Sahni, S. : Computational Geometry on a Reconfigurable Mesh, In Proc. 8th International Parallel Processing Symposium, pp.86-93, IEEE(Apr. 1994).
10) Wang, B.-F., Chen, G.-H. and Lin, F.-C. : Constant Time Sorting on a Processor Array with a Reconfigurable Bus System, Information Processing Letters, 34(4), pp.187-192(Apr. 1990).

（平成9年3月17日受付）



中野　浩嗣（正会員）

平成4年大阪大学大学院基礎工学研究科博士後期課程修了．工学博士．同年日立製作所基礎研究所勤務．平成7年より名古屋工業大学講師．現在に至る．並列・分散アルゴリズムの研究に従事．IEEE，ACM，EATCS，電子情報通信学会各会員．