「情報処理」第26巻第6号別刷　昭和60年6月発行

# ソーティングのハードウェアアルゴリズム

## —VLSI モデル上での計算複雑度—

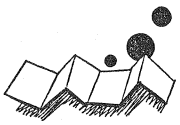
Clark D. Thompson
安浦寛人　高木直史 共訳

社団法人 情報処理学会

Vol. 26 No. 6　　情　報　処　理　　June 1985

解　説

4. 各種のハードウェアアルゴリズム



# 4.2 ソーティングのハードウェアアルゴリズム†

## —VLSI モデル上での計算複雑度—

Clark D. Thompson††
安浦寛人†††　高木直史†††共訳

## 1. まえがき

ソーティングは，過去20年間，計算機科学の中で魅力的なテーマであった．それは，ソーティングが実用的にきわめて重要であるとともに，理論的にも興味深い問題であるからである．全世界の計算機の1/4の計算時間がソーティングに費されているといわれたこともあった[19]．今はそのようなこともないであろうが，ソーティングや情報のシャフル（並べ換え）は，近年急速に発展しているデータベースの中でその重要性を増している．

ソーティングは入力された $N$ 個の値を小さい順に並べ換える問題である．本稿では，VLSI 上で実現することを前提にソーティングの複雑さを調べる．他の計算モデルの上では，ソーティングについて多くのことが知られており[19]，これらの知識は VLSI 上でのソーティングにも利用できる．しかし，VLSI は，回路の大きさがゲートの量だけでなく配線によって決まるという性質を持つ新しい計算機構であり，「VLSI モデル上での計算」としてそのアルゴリズムを再評価する必要がある．

VLSI モデルの上で，ソーティング回路に面積時間2乗積のトレードオフがあることが示せる．このトレードオフの最初の研究は，著者の博士論文[42]に含まれているが，そこには，2つのソーティング回路がとり上げられている．本稿では，さらに11種の回路（アルゴリズム）を新しい VLSI モデルの上で解析する．

ここで取り上げるソーティング回路はすべて新しいものでなく，逐次型アルゴリズムとしてよく知られているものに基づいたものばかりである．これらはハードウェアアルゴリズムとしてもすでに提案されているが，その多くのものは VLSI 上での面積や時間の複雑度を評価されるのは初めてである．10種のアルゴリズムは面積時間2乗積が $O(N^2 \lg^2 N)$* から $O(N^2 \lg^5 N)$ の間に入ることがわかった．どのような設計においても面積時間2乗積は $\Omega(N^2 \lg^2 N)$ になることが知られているので[50),51]，これらのアルゴリズムは面積および時間に関して対数ファクタの違いを許せば最適であるといえる．

実用的な見地からすれば，ソーティング回路は順序付け（rank）と並べ換え（permute）をする必要がある．もちろんこの一方だけで良い場合もあるが，多くの応用においては両方とも必要である．例えば，関係データベースの射影（projection）操作における同一レコードの除去を考えてみよう．同一レコードは，レコードが大きさの順に並べ換えられていれば簡単に検出し除くことができ，その後のデータベース演算の処理時間やスペースを節約できる．

歴史的にも，ソーティング回路は順序付けと並べ換えをするとした方がよい．「ソーティング」の本来の意味は，「クラスや種類によって分離または整理すること」[19]とあり，クラス分けと移動の双方を含む操作である．すなわち，実用的にも歴史的にも，（順序付け＋並べ換え）＝「ソーティング」と考えて良く，ここでは，これだけを考える．

---

† Hardware Algorithms for Sorting—The VLSI Complexity of Sorting— by Clark D. Thompson (University of California Berkeley) Translated by Hiroto YASUURA and Naofumi TAKAGI (Faculty of Engineering, Kyoto University).

†† カリフォルニア大学バークレー分校

††† 京都大学工学部



* $\lg \equiv \log_2$

## 2. VLSI のモデル

ここで用いる VLSI モデルは以下のようなものである.

仮定1〔埋め込み〕

a) VLSI の回路は2次元のユークリッド平面上の節点 (node) と配線 (wire) から構成される. グラフ理論の言葉で言えば, 配線は2個以上の節点を結ぶハイパ枝 (節点の集合) である. このハイパ枝は線分である配線要素 (wire segment) と適当に加えられたファンアウト点の木として埋め込まれる.

b) 配線要素は単位幅を持つ. 配線の長さは各配線要素の長さの総和である.

c) 平面の各点で重なり合うことのできる配線の数は高々2である.

d) 節点は $O(1)$ の面積を占める. 節点は $O(1)$ 本の入力および出力配線を持つ.

e) 節点と配線, 節点同士は重なり合わない.

f) 長さ $k$ で $n$ 個の節点の入力となる配線は, $c_w k + c_g n$ 個以上の節点の出力となっている.

仮定2〔問題の定義〕

a) チップが $c$ 個の問題のインスタンスを同時に解く時, $c$ の並列度を持つという.

b) $N$ 個の入力変数の各々は, 等確率で $M$ 個の値のうちの1つを取る.

c) $M=N^{1+\varepsilon}$ ($\varepsilon$ は正の定数). 冗長性の無い符号化を使うので, 入力および出力変数は $(1+\varepsilon)(\lg N)=\theta(\lg N)$ ビットで表わされる.

d) 問題のインスタンスの出力値は入力値を昇順に並べ換えたものである.

仮定3〔タイミング〕

a) 配線は単位バンド幅を持つ. すなわち単位時間に1ビットの信号を伝播する.

b) 各節点は $O(1)$ の遅延を持つ.

仮定4〔伝達関数〕

a) 各節点の伝達関数 (Transmission Function) は, 入力配線上の信号に対して, 節点の出力や内部状態がどう変化するかを規定する. 厳密にいえば, 節点の状態 (state) は, 入力信号のある関数によって各単位時刻ごとに更新されるビットベクタである. 同様に, 出力配線上の出力信号は, 現在の状態の関数として決まる.

b) 2つ以上の出力が同じ配線に接続されている時, これらの出力信号はいつも同じである. これを確実にするために, これらの出力を出す節点は同一の伝達関数を持ち, 同一の入力配線に接続されるものとする.

c) 節点には, 論理節点, 入出力ポート, 入出力記憶の3種がある. 入出力記憶には, さらに RAM 型とシフトレジスタ型とがある.

d) 入出力記憶は論理節点に直接接続することはできない.

e) 論理節点と入出力ポートは $O(1)$ ビットの状態を持てる.

f) $(k_1\times k_2)$ ビットの入出力記憶は, $k_1k_2+\lg k_1k_2$ ビットの状態を持つ. アドレスレジスタが $\lg k_1k_2$ ビットであり, 残りの $k_1k_2$ ビットはデータの状態である.

g) 各入力ビットの与えられる位置は固定であり, 入出力メモリのデータ部分に入る. 計算の開始時には, すべての他のビットは, 問題に依存しない固定値に初期化される.

h) 各問題の出力ビットは, 入出力メモリの固定された位置に入る. 計算の終了時には, 出力ビットに対応する入出力メモリの値は, 仮定2に定める値となっている.

i) RAM 型の $(k_1\times k_2)$ ビットのメモリに接続される入出力ポートは, $O(k_2+\lg k_1)$ 単位時間のメモリサイクルで動作する. 最初の $\lg k_1$ 単位時間で, ビット直列に語のアドレスを受け取り, 次に読み出しか書き込みを指示する信号が入る. さらに, 次の $k_2$ サイクルで, ビット直列に読み出しまたは書き込みを行う.

j) シフトレジスタ型の $(k_1\times k_2)$ ビットのメモリに接続される入出力ポートは $O(k_2)$ 単位時間のメモリサイクルで動作する. 1つのサイクル内では, 奇数番目の時刻 (単位時間) で, 現在のアドレスレジスタで示される番地の内容が1ビットずつ出力され, 偶数番目の時刻では, 書き込みが行われてアドレスレジスタが1増加する (ただし $\bmod k_1k_2$).

仮定5〔面積, 時間, 効率〕

a) チップ上の総処理面積 $A_{\text{processing}}$ は, 回路を含む最小の長方形内の単位正方形の数とする.

b) チップの面積効率 $A$ は, 総処理面積をその並列度 $c$ で割ったものである.

c) 総計算時間 $T_{\text{total}}$ は, $c$ 個の問題のインスタンスの計算の開始から終了までの平均時間 (単位時間の数) である.

d) チップの時間効率 (遅延時間) $T_d$ は, 各問題

のインスタンスについての最初のメモリアクセスサイクルから最後のアクセスサイクルの間の時間の平均である．

e）チップの周期 $T_p$ は $T_{\text{total}}$ を並列度 $c$ で割ったものである．ここで，$T_p > T_d$ となる場合は必ず無駄な時間があり，$T_p = T_d$ となるように設計しなおすことができる．また，$c=1$ のときは $T_p = T_d = T_{\text{total}}$ である．

f）ソーティングチップの入出力バンド幅は入出力メモリに書き込みまたは読み出される総ビット数を $T_{\text{total}}$ で割ったものである．

## 3. 回路構成

ここでは，種々のソーティングチップの回路構成を紹介する．最初に，各設計の中で使われる基本モジュールについて説明する．

直列比較-交換モジュール（「比較器」）は，$O(1)$ のゲート数で $A=O(1)$ で実現できる[27]．このモジュールは，入力 $A, B$ に対応する2つのビット直列入力と，出力 $\max(A,B)$, $\min(A,B)$ に対応する2つのビット直列出力を持つ．これらの入出力は最上位桁を先頭とした直列の2進符号で表わされる．

応用によっては，このモジュールに2本の制御線が付加される．これにより，1)無条件に入力をそのまま出力する．2)無条件に入力を入れ換えて出力する．3)大きい方の入力を出力1に出す．4)小さい方を出力1に出すの4つの状態を制御する．この複雑化したモジュールも，$O(1)$ の面積で実現でき，$O(1)$ 単位時間ごとに各出力ビットを出力する．

比較-交換モジュールは，図-1 のようにパイプライン化して作ることもできる．入力の対がモジュールの上部から入力され，単位時間ごとに1段ずつ下へ移動して行く．各段では，円の素子で1ビットの比較-交換が行われる．正方形の素子では，入力がそのまま通過する．比較-交換の方向に関する情報は，対角線方向に円の素子を次々に伝わっていく．

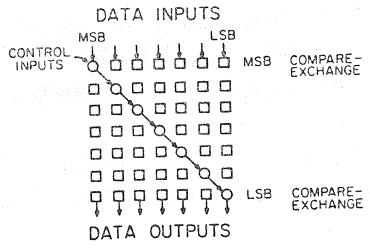



図-1 パイプライン比較-交換モジュール

このパイプライン化した比較-交換モジュールは，$O(\lg N)$ ビットの入力に対し，周期 $T_p=O(1)$ 単位時間，遅延 $T_d=O(\lg N)$ 単位時間で比較-交換を完了する．総面積は $O(\lg^2 N)$ で並列度は $\lg N$ であるから，面積効率 $A=O(\lg N)$ となる．たいていの応用では，入力と出力は時間差がついているとしてよいので，正方形の素子は省略できる．これにより，総面積 $A_{\text{processing}}=O(\lg N)$ となり，面積効率 $A=O(1)$ とすることができる．これは，パイプライン化しないものと同じである．

3番目の基本モジュールは，プログラム制御ユニット（PCU）である．PCU は多数の制御信号をごく小さい面積で生成するのに用いられる．後に述べる設計では，完全なソーティングアルゴリズムが $O(1)$ 個の PCU 命令に符号化される．各命令長は $O(\lg N)$ ビットで，実行時間は $T_d=O(\lg N)$ 単位時間である．命令セットは，分岐，算術演算（シフト，加算，否定等），テスト，レジスタ間転送等を含む．PCU は，$O(1)$ 個のレジスタを持ち，そのうち1つが比較演算モジュールの制御線に接続されている．残りのうち1つは，アドレスや近くの入出力ポートの制御信号を作るのに使われる．

PCU と $O(1)$ 個の入出力ポートおよび直列比較-交換モジュールを組み合わせたものを「ビット直列プロセッサ」と呼ぶ．このプロセッサは $O(1)\times O(\lg N)$ の長方形の中に入り，比較-交換操作を $T_p=T_d=O(\lg N)$ 単位時間で行う．

「語並列プロセッサ」は，PCU とパイプライン化比較-交換モジュールおよびシフトレジスタ型記憶に接続された $O(\lg N)$ 個の入出力ポートから構成される．語並列プロセッサは，比較-交換操作を $T_p=O(1)$ 単位時間で行うことができる．また，$O(1)\times O(\lg N)$ の長方形の中に入るので，直列プロセッサと定数倍しか面積は違わない．

VLSI 向きソーティング回路を調べる準備ができたので，並列度の低いものから順に設計を調べて行こう．

A．単一プロセッサ上のヒープソート

これは考えうる最小のソータである．すなわち，N 語のデータに対し，普通のヒープソートアルゴリズム[19]を行うビット直列プロセッサである．各比較-交換と入力データのランダムアクセスには，$O(\lg N)$ 時間がかかるので，ソーティング全体では $T_d=T_p=$

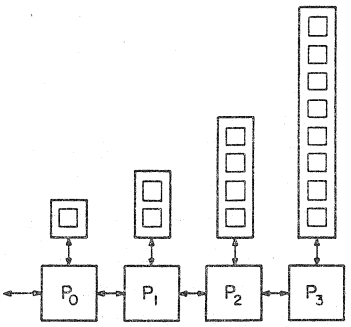



図-2 ($\lg N$) プロセッサ上のヒープソート ($N=16$)

$O(N\lg^2 N)$ となる．他にマージソートやクィックソートのアルゴリズムも考えられるが，$O(N\lg N)$ 回の入出力記憶へのランダムアクセスが必要なので，$AT_d^2$ をより小さくすることはできない．

B．($\lg N$) プロセッサ上のヒープソート

ヒープソートは $\lg N$ 個のビット直列プロセッサの1次元配列上で並列化できる．各プロセッサは，図-2のように，ヒープの各レベルに対応する[2),40)]．ヒープ操作はパイプライン化され，挿入（取出し）モードではデータが $O(\lg N)$ 単位時間ごとに1レベルずつ右方（左方）へ移動する．ヒープの最上レベルのプロセッサは，ヒープに入っている最小の要素1個だけを記憶する．$k$ 番目 ($0\leq k<\lg N$) のプロセッサは，($2^k\times\lg N$) ビット RAM に $2^k$ 個の要素を記憶する．全体のソーティング時間は，$T_d=T_p=O(N\lg N)$ で面積は $A=O(\lg^2 N)$ である．

C．($1+\lg N$) プロセッサ上のマージソート

マージソートアルゴリズムは，$\lg N$ 個程度のプロセッサによる実現に適している[45)]．各プロセッサは，2つの可変長 FIFO キューを持つ．すなわち，プロセッサ $P_k(0\leq k\leq\lg N)$ には2つの $2^k$ 語のキューがその出力線についている．

図-3に示すように，プロセッサ $P_k$ は，すでにソートされた長さ $2^{k-1}$ の2つのリストから，ソートされた長さ $2^k$ のリストを作る．これは，2つの入力キューの先頭の要素の小さい方を取り，出力リストの最後尾につけるだけでよい．1つの出力キューに $2^k$ 個の要素を出力すると，次はもう一つの出力キューに対し出力を行う．これを入力がなくなるまで繰り返す．プロセッサ $P_3$ は特別で，入力系列を交互に左右の出力キューへふり分ける操作を行う．

最大の効率を得るためには，パイプライン化した語並列のプロセッサを使いたい所だが，マージソートで

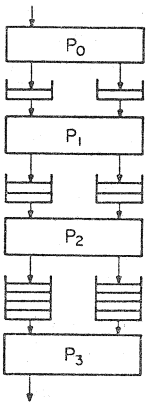



図-3 ($1+\lg N$) プロセッサを用いたマージソート ($N=8$)

は1つの比較が終了するまで次の比較を開始できないのでこの型のプロセッサを有効には使えない．そこで，ビット直列のプロセッサで考える．

FIFO キューは，RAM 型記憶によって容易に実現できる．$2^k$ 語の RAM のアクセス時間は $O(k)$ であるが，$k\leq\lg N$ なので，$O(\lg N)$ ビットの語に対するビットシリアルなプロセッサの処理速度に対して十分速い．

このマージソートの時間効率は各プロセッサの効率によって決まる．ソーティングにかかる遅延時間は，$T_d=O(N\lg N)$ となる．また面積は，各プロセッサが $O(\lg N)$ の長方形中に作れるので，$A=O(\lg^2 N)$ となる．

D．($\lg N$) プロセッサ上のバイトニックソート

この設計も，一見すると前の2つとよく似ている．ここでは，プロセッサ $P_k$ ($0\leq k<\lg N$) は，($N/2^{k+1}$) 語の補助 FIFO キューを持つ（図-4 参照）．この回路は，バイトニックソートアルゴリズムに従って動作する[19)]．バイトニックソートアルゴリズムは $\lg N$ 回の大きな繰り返しからなると考えられる．各繰り返しは，$N$ 個の入力データ上の距離 $N/2$ 操作，距離 $N/4$ 操作，…，距離2操作，距離1操作と呼ばれる，$\lg N$ 個の操作からなる．プロセッサ $P_k$ は，距離 ($N/2^{k+1}$) 操作を担当する．1回の大きな繰返しが図-4 の $P_0$ から $P_3$ までを含む大きなループを1周することに対応する．

各操作における“距離”とは，入力データ列の中での順番の差として普通に定義されるものである．まず，入力は，$P_0$ に $x_0x_1\cdots x_{N-1}$ の順に与えられる．$N/2$ 語の FIFO を用いて，$P_0$ は $x_{N/2+i}$ と $x_i$ を

比較し，必要に応じて交換する．その結果新しい系列 $(x'_0x'_1\cdots x'_{N-1})$ が $P_1$ へ送られる．同様に，プロセッサ $P_k$ は，$i$ 番目と $N/2^{k+1}+i$ 番目のデータに対し比較-交換を行う．

各プロセッサでの比較-交換の方向は，一定ではない．ある時は，$x_i>x_{N/2^{k+1}+i}$ のとき両者を交換するし，別の時は $x_i<x_{N/2^{k+1}+i}$ のとき交換を行うし，また無条件に交換をしない時もある．この3つの可能性は，各プロセッサにおけるデータの処理の3つのパターンに反映される．第1のパターンでは，$P_k$ は，$P_{k-1}$ から受け取ったデータをそのまま FIFO の最後に入れ，同時に FIFO の先頭の要素を $P_{k+1}$ へ送る．パターン2では，$P_k$ は $P_{k-1}$ から受け取ったデータと FIFO の先頭のデータの比較を行い，小さい方を $P_{k+1}$ へ送り，大きい方を FIFO の最後尾につける．3番目のパターンは，2と同様に比較を行い，大きい方を $P_{k+1}$ へ，小さい方を FIFO へ送る．バイトニックソートアルゴリズムの詳細は**図-5** に示す．

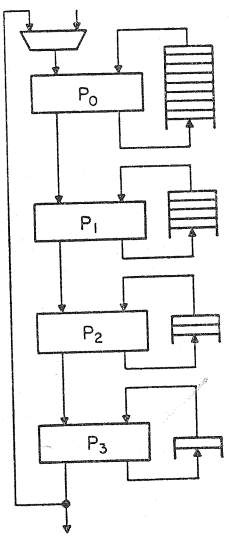



**図-4** $(\lg N)$ プロセッサ上のバイトニックソート $(N=16)$

図-5 のアルゴリズムで，プロセッサ $P_k$ は，最初に入ってくる $(\lg N-k-1)$ $N$ 個のデータに対してパターン1を実行する．これが図-5 の1番外側のループを $(\lg N-k-1)$ 回繰り返すことに相当する．その後，プロセッサ $P_k$ はアクティブになり，残りの $k+1$ 回分の繰り返しの中でデータの入れ換えを行う．この中で FIFO に新しいデータを入れる $(N/2^{k+1}$ 回のパターン1の実行) と FIFO の内容と入ってくるデータに対する比較-交換操作 $(N/2^{k+1}$ 回のパターン2またはパターン3の実行) が交互に行われる．

7行目と8行目の条件判定のためには，3つのカウンタ $(g,i,j$ に対応) を用意すればよい．DIV と MOD の演算は，これらのカウンタの1ビットを見るだけでよい．よって，このバイトニックソートアルゴリズムは，$\lg N$ 個のビット直列プロセッサ (面積 $O(\lg N)$) の上で，面積 $A=O(\lg^2 N)$ で実現できる．1つのデータが各プロセッサを通るのには $T_p=O(\lg N)$ 時間がかかるので，1回の大きな繰り返しには $O(N\lg N)$ 時間がかかる．よって，全体では $T_d=O(N\lg^2 N)$ となる．

語並列プロセッサを用いる時は，$O(\lg N)$ 本の通信線を各プロセッサ間に用意すれば，1回の大きな繰り返しには $O(N)$ 時間しかかからず，$T_d=O(N\lg N)$ となるが，面積は $A=O(\lg^2 N)$ と変わらない．ただし，この設計では，$O(\lg^2 N)$ 個の入出力ポートが必要となる．

E．$O(\lg^2 N)$ プロセッサ上のバイトニックソート

この設計はDの設計を展開したものと考えればよい．各プロセッサは1つの距離 $(N/2^{k+1})$ 操作だけを担当する．実際には，バイトニックソートアルゴリズムの中で約半分の操作は NO-OP (何もしない) なので，**図-6** のように約 1/2 $(\lg^2 N)$ 個のプロセッサがあればよい．各プロセッサは $O(1)\times O(\lg N)$ の長方形

```
1.  FOR g←0 TO lg N−1 DO
2.    FOR j←0 TO 2^k−1 DO
3.      FOR i←0 TO N/2^(k+1)−1 DO /* Fill up FIFO with new data */
4.        {execute pattern 1};
5.      OD;
6.      FOR i←0 TO N/2^(k+1)−1 DO /* Perform comparison-exchanges as necessary */
7.        IF g<lg N−k−1 THEN {execute pattern 1}
8.        ELSEIF ((j DIV(2^(g+k+1)/N))MOD 2)=0 THEN {execute pattern 2}
9.        ELSE {execute pattern 3} FI;
10.     OD;
11.   OD;
12. OD.
```

**図-5** バイトニックソートアルゴリズム (図-4 のプロセッサ $P_k$ の実行するアルゴリズム)

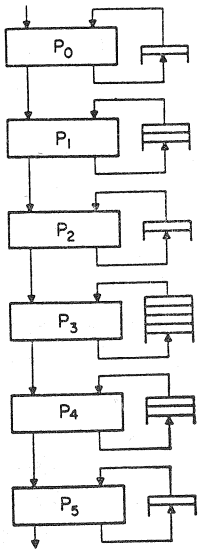



図-6 $(1/2)(\lg N)(1+\lg N)$ プロセッサを用いたバイトニックソート ($N=8$)

に入るので，全体の面積は $O(\lg^3 N)$ となる．

距離 $(N/2^{k+1})$ 操作は $N/2^{k+1}$ 語の FIFO で実現でき，距離 $(N/2^{k+1})$ 操作のうち NO-OP でないものは $k+1$ 個であるから，全体の記憶量は $\sum_k (k+1)(N/2^{k+1}) \approx 2N$ 語である．遅延時間はこのパイプラインの長さに比例するから，語並列プロセッサを用いた時には，$T_d=O(N)$ となる．このパイプラインは2つのソーティング問題を同時に入れることができるので，面積効率は総処理面積の半分となり，$A=O(\lg^3 N)$ である．

この設計の面積時間2乗積 $AT_d^2$ は $\lg N$ 倍だけDより改善されている．この事情をもう少し理解するために，Dの設計のソータを $\lg N$ 個ならべたものとこの設計を較べてみよう．両方とも総面積は同じであり，$\lg N$ 個の問題を解くのに双方とも $T_{\text{total}}=O(N \lg N)$ 時間かかる．しかし，この $O(\lg^2 N)$ プロセッサ上実現は不要な NO-OP を削除することによって，1つの問題を解く遅延時間を $O(1/\lg N)$ 倍に削減することに成功しているわけである．

F. $\sqrt{N \lg N}$ プロセッサ上のバイトニックソート

これは，Chung らが最近提案した設計で[8]，$\sqrt{N \lg N}$ 個のプロセッサの1次元配列上のバイトニックソートである．各プロセッサは，$\sqrt{N/\log N}$ 語のシフトレジスタ記憶を持つ．これらの語並列プロセッサは，その局所記憶に対してバブルソートを $T_d=O(N/\lg N)$ で行う．配列全体では $N$ 個の要素のソーティングを $T_d=O(N)$, $A=O(\sqrt{N \lg^3 N})$ で実行する．ただ，この設計では，入出力バンド幅が大きくなりすぎるので，シフトレジスタを回路内で実現することにすれば，$A=O(N \lg N)$ に増加する．

本稿のモデルでは，この設計は，Eと較べ速度も速くないし，面積も大幅に大きいので，$AT_d^2$ の意味で最適とはほど遠い．

G. $(N/2)$ 比較器上のバブルソートアルゴリズム

多くの論文で指摘されているように，バブルソートアルゴリズムは $N/2$ 個のビット直列比較-交換モジュールの1次元配列の上で完全に並列化できる[3),6),12),15),24),28),29)]．各モジュールは，次のような単純な動作をする．左右の隣接するモジュールから1つずつデータを受け取り，比較して小さい方を左に，大きい方を右に送る．最初に，配列全体は並列に0に初期化され，左端のモジュールから直列にデータが入れられる．そして，入れ換えが行われ，最左端のモジュールから大きな順に出力される．

比較器は $O(\lg N)$ の面積なので，この設計の面積は $O(N \lg N)$ となる．ビット直列のモジュールを使うと，$T_d=O(N \lg N)$ であり，語並列のモジュールを使うと $T_d=O(N)$ となる．$AT_d^2=\Omega(N^2 \lg N)$ という下界と較べると，最適な設計とはいえない．

この他にも，$O(N)$ 個のプロセッサを使って同様の面積時間積となるような設計が少なくとも3つ知られている．ヒープソートを $N$ 個のビット直列プロセッサからなる2分木上で行う方法がその1つである[26)]．この2分木構造は，最近発表された $N$ 個のプロセッサの上での並列計数ソート[49)]やラディックスソート[10),47)]の放送 (Broadcast) にも使え，これらの設計においても，ビット直列プロセッサを用いると $A=O(N \lg N)$, $T_d=O(N \lg N)$ となる．これらに語並列プロセッサを使うこともできるが，そうしてみても $AT_d^2$ は $N$ の3乗以下にはならない．

H. $N$ プロセッサの2次元格子上のバイトニックソート

バイトニックソートは，$N$ 個のビット直列プロセッサの2次元配列の上で，非常に効率良く実現できる[31),42)]．さらに，語並列プロセッサを使って以下のように高速化することも可能である．

このアルゴリズムはかなり複雑なので，ここでは説明しない．ここでは，バイトニックソートの中の $O(N \lg^2 N)$ 回の比較-交換が，$O(\lg^3 N)$ の時間でできることを知るだけで十分である．しかし，次の比

較-交換の準備をするためのデータの並べ換えになんと $O(\sqrt{n})$ の時間を要してしまう．幸いなことに，このような時間のかかる並べ換えが必要なのは，ほんとうにわずかなので，ソーティング全体の遅延時間 $T_d=O(\sqrt{n})$ となる．

この速度を実現するには，プロセッサ間のデータの転送が $O(1)$ の時間でできなければならない．これには少し工夫がいる．隣り合うプロセッサ間の結線の長さは $O(\lg N)$ であるので，モデルの仮定 (1 f) によると，時間 $O(1)$ の転送をするためには，$O(\lg\lg N)$ の時間を要して信号を増幅（複製）しなければならない．しかし，一度データを増幅しておきさえすれば，$O(\lg N)$ 個の同一で同じ動作をする節点で回路全体を $O(\lg N)$ 倍になるように作ることにより，その後の転送はすべて $O(1)$ の時間で行える．

結果として，総面積は $A=O(N\lg^2 N)$ となる．プロセッサだけの占める面積は $O(N\lg N)$ であり，残りの部分は語並列転送とドライバの工夫のための面積の増加分である．

I．$N$プロセッサのシャフル交換網上でのバイトニックソート

Stone は，バイトニックソートが $N$ 個のプロセッサのシャフル交換網上で効率良く実現できることを示した[39]．接続をビット直列で行えば，バイトニックソートの中の $O(N\lg^2 N)$ 回の比較-交換を全体で $T_d=O(\lg^3 N)$ で行うことができる．

この設計は，小さな領域に埋め込めるわけではない．シャフル交換グラフの埋め込みに必要な面積の下界は $\Omega(N^2/\lg^2 N)$ であることが知られている[42]．最近，この面積の埋め込みが与えられた[18),23)]．この埋め込みは，各々 $O(1)\times O(\lg N)$ の長方形に入るビット直列プロセッサ$N$個分の場所を作るために垂直方向に $O(\lg N)$ 分だけ引き延ばすことができる．このように変形しても，面積は $O(N^2/\lg^2 N)$ であり，その最も長い配線は，$O(N/\lg N)$ の長さである．

この設計の面積時間積は，最適に近い．さらに，プロセッサ間通信を並列化することによって改善できるかもしれない．例えば，シャフル交換グラフの各辺に $k$ 本の線をはったとすると，$k$ 倍早くなると考えられる．しかし，残念ながら面積も $k^2$ 倍になるので，$AT_d^2$ は変わらない．

J．$N$プロセッサの PCCC 上でのバイトニックソート

Preparata と Vuillemin は，彼らの提案した CCC (Cube-Connected Cycle) の上でバイトニックソートのアルゴリズムが I の場合と同様に $A=O(N^2/\lg^2 N)$, $T_d=O(\lg^3 N)$ で実現できることを示した[33]．シャフル交換網の最適レイアウトがあまり規則的でなかったのに対し，この結合網は単純な最適レイアウトを持つという利点がある．しかし，この CCC 上のバイトニックソートアルゴリズムは，シャフル交換網上のものにくらべ多少複雑になる．

最近，この CCC の $AT_d^2$ に関する最適性がさらに改善された．新しい構成は，「PCCC (Pleated CCC)」と呼ばれている[4]．基本的なアイデアは，バイトニックソートの短距離操作に対し，より短かい配線を用意してやろうということである．その代り，たまに発生する長距離操作は，少し長い配線となってしまう．PCCC 上では平均の配線長が大きくなるので面積は増大するが，ソーティングの時間は減少する．その結果面積が $A=O(N\lg N)$ から $A=O(N^2/\lg^4 N)$ の間で，時間 $T_d=O(\sqrt{N\lg N})$ から $T_d=O(\lg^3 N)$ の設計が種々得られる．いずれも $AT_d^2=O(N^2\lg^2 N)$ となり，この意味で最適である．

K．$(N\log^2 N)$ 比較器上のバイトニックソート

Batcher のバイトニックソート網[19] をそのまま VLSI チップにしたものがこれである．$(1/2)(\lg^2 N+\lg N)$ 個の比較-交換操作が，それぞれ１つの行の $N/2$ 個のビット直列比較-交換モジュールで並列に行われる．配線領域は比較器の面積よりも大きくなる．距離 $(N/2^{k+1})$ 操作を行う比較器の前には，$O(N/2^{k+1})\times O(N)$ の配線領域が必要となる．距離 $(N/2^{k+1})$ 操作は $k+1$ 個あるので，全体の面積は $\sum_k (k+1)(N^2/2^{k+1})=O(N^2)$ となる．

全体の遅延は $T_d=\theta(\lg^3 N)$ である．フィードバックループがないため，これは容易にパイプライン化でき，その並列度を $\theta(\lg^2 N)$ とできる．よって，面積効率 $A=O(N^2/\lg^2 N)$ となる．

さらに，この $AT_d^2$ 性能を改善する手法がある．この設計の中では，長い配線をドライブするために，多段の増幅回路を付けているが，回路全体をその出力配線の長さに応じて大きく (Scale up) すれば，この増幅回路による遅延を無くすことができる．すなわち，距離 $(N/2^{k+1})$ 操作をする比較器は $O(N/2^{k+1})$ の面積を持つようになる．それでも面積は全体で $O(N^2)$ となる．遅延は，次の行への転送が $O(1)$ の時間で行えるので，$T_d=O(\lg^2 N)$ に減少する．この時，並列度も $O(\lg N)$ になり，面積効率は，$O(N^2/\lg N)$ と

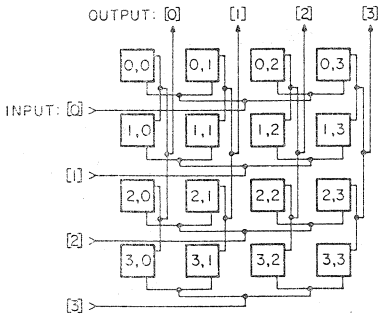



**図-7** 直交木網 ($N=4$)

なる.

最近, $O(N \lg N)$ 個の比較器を用い, $O(\lg N)$ 段のソーティング回路が発表された[1]. 完全に並列的に実現すると, これはそれぞれ $N/2$ 個の比較器からなる, $O(\lg N)$ 行の配列で実現できる. 行の間の結合は"拡大グラフ (Expander Graph)"となり, その平均配線長は $\theta(N)$ となる. 各プロセッサを $N$ 倍に拡大 (Scale up) すれば, 面積は $O(N^2 \lg N)$, ソーティングの時間は, $T_d=O(\lg N)$ とできる.

L. $N^2$ 比較器の上でのバブルソート

これは, バブルソート中の $N$ 個の要素間の $N^2$ 回の比較のそれぞれに比較-交換モジュールを用意したものである[19),52)]. ビット直列比較器を用いると面積が $O(N^2)$, 遅延が $O(N)$ となり, また同時に $N/\lg N$ 個の問題を処理することになる. よって面積効率は $A=O(N \lg N)$ である.

パイプライン化した比較-交換モジュールを用いると, 総面積は $N^2 \lg^2 N$, 並列度は約 $N \lg N$ となり $A=O(N \lg N)$ である. 一方, 遅延は $T_d=O(N \lg N)$ と悪くなる.

M. $N^2$ プロセッサ上のランクソート

次のような $N^2$ 個のプロセッサからなる正方行列を考えよう. 各行のプロセッサはそれらを葉とする完全2分木状に接続されている. この各行の木はその根と葉(プロセッサ)の間でビット直列通信を行える. 同様に, 列方向にも木状に接続される. すなわち, 各プロセッサは, 直交する2つの木の葉となっている[32),22),23)]. **図-7** に $N=4$ の例を示す.

この行列の上で, 力まかせのアルゴリズム[36)]が実現できる. 各入力データは各行の木の根に1つずつ置かれる. 入力データはその木の全部の葉へ送られる. 各行のプロセッサは同じデータを持つことになる. 次に列方向の木を使い, $j$ 番目の列の $j$ 番目の葉プロセッサから, データ (入力の $j$ 番目のデータ) をその列内のすべての葉プロセッサへ転送する. これで配列の $(i, j)$ 要素に当るプロセッサは, 入力の $i$ 番目と $j$ 番目のデータ (input $(i)$ と input $(j)$) を持つ.

各プロセッサは, 2つのデータを比較し, input $(i)$ > input $(j)$ (ただし, $i>j$ のときは input $(i)\geq$ input $(j)$) のときは, 1を各行の木へ送出する. 各行の木では, この結果を加え合せる. これで input $(i)$ の順位 (ランク) が求まる. この順位は, 各行の木の葉プロセッサへ伝えられる. $(i, j)$ 番目のプロセッサで, 順位が $j$ と等しいものが, そのデータ input $(i)$ を列方向の木の根へ送る. その結果, $j$ 列目の木の根に, $j$ 番目に小さなデータが出力される.

転送, 加算, 選択の各操作をビット直列で行えば, 遅延は $T_d=O(\lg N)$ で, 面積は $O(N^2 \lg^2 N)$ である[23)]. 図-7 のようなレイアウトでは, 根に近くなるほど配線が長くなる. そこで木のレベル $k$ の内部節点に, $O((N \lg N)/2^k)$ 個のドライバ節点を加えて $O((N \lg N)/2^k)$ の長さの配線を単位時間でドライブできるようにする. 各内部節点の計算は $O(1)$ の面積で行え, レベル $k$ の節点は $2^k$ 個あるので, 各レベルは $O(1)\times O(N \lg N)$ の長方形の中に入り, 面積が $O(N^2 \lg^2 N)$ となる.

## 4. 設計の比較

**表-1** に前節で紹介した13個の設計の面積および時間効率をまとめている. 設計名の後にSが付いているのは直列比較器, Pが付いているのはパイプライン比較器を用いたものである. 面積効率は処理面積を並行度で割ったものである. これは各ソーティングに消費される電力の尺度と見ることもできる. 一方, 時間効率 (遅延) は入力を入れ始めてから出力を終るまでにかかる時間である.

表-1から, ほとんどの設計が $AT_d^2$ に関して $O(\lg^k N)$ 倍の違いを除いて最適となっていることがわかる. 例外は, 2つのバブルソートと $\sqrt{N \lg N}$ プロセッサ上のバイトニックソートである.

**表-2** は, 別の尺度で比較したものである. 最初の並列度は, 同時に解く問題の数である. 2番目と3番目の列は, 処理面積と記憶の面積 (1ビットが1単位面積と考える) である. 最後の列はプロセッサと記憶の間の入出力バンド幅を単位時間あたりのビット数で表わしたものである. この数は, プロセッサチップの入出力ポートの数と考えることができる.

表-1 ソーティング問題に対する面積-時間複雑度

| | 設計 | 面積効率($A$) | 時間効率($T_d$) | $AT_d^2$ |
|---|---|---|---|---|
| | 下界 | —— | —— | $\Omega(N^2 \lg N)$ |
| A | 単一プロセッサ(S) | $\lg N$ | $N \lg^2 N$ | $N^2 \lg^5 N$ |
| B | ($\lg N$) プロセッサ上のピープソート(S) | $\lg^2 N$ | $N \lg N$ | $N^2 \lg^4 N$ |
| C | ($\lg N$) プロセッサ上のマージソート(S) | $\lg^2 N$ | $N \lg N$ | $N^2 \lg^4 N$ |
| D | ($\lg N$) プロセッサ上のバイトニックソート(P) | $\lg^2 N$ | $N \lg N$ | $N^2 \lg^4 N$ |
| E | ($\lg^2 N$) プロセッサ上のバイトニックソート(P) | $\lg^3 N$ | $N$ | $N^2 \lg^3 N$ |
| F | ($\sqrt{N \lg N}$) プロセッサ上のバイトニックソート(P) | $N \lg N$ | $N$ | $N^3 \lg N$ |
| G | ($N/2$) 比較器上のバブルソート(P) | $N \lg N$ | $N$ | $N^3 \lg N$ |
| H | $N$プロセッサの2次元格子上のバイトニックソート(S) | $N \lg^2 N$ | $\sqrt{N}$ | $N^2 \lg^2 N$ |
| I | $N$プロセッサのシャフル交換網上のバイトニックソート (S) | $N^2/\lg^2 N$ | $\lg^3 N$ | $N^2 \lg^4 N$ |
| J | $N$プロセッサの PCCC 上のバイトニックソート(S) | $N^2 \lg^2 N/T_d^2$ | $\lg^3 N \leq T_d \leq \sqrt{N \lg N}$ | $N^2 \lg^2 N$ |
| K | ($N \lg^2 N$) 比較器上のバイトニックソート(P) | $N^2/\lg N$ | $\lg^2 N$ | $N^2 \lg^3 N$ |
| L | $N^2$ 比較器上のバブルソート(S) | $N \lg N$ | $N$ | $N^3 \lg N$ |
| M | $N^2$ プロセッサ上のランクソート(S) | $N^2 \lg^2 N$ | $\lg N$ | $N^2 \lg^4 N$ |

これまでは記憶面積 $A_{\text{memory}}$ と入出力バンド幅は，あまり考慮してこなかった．また，プロセッサの面積についても $A=A_{\text{processing}}/c$ を中心に考えてきた．しかし実際のシステムの立場からは，表-2 の尺度の方が表-1 のものよりも重要である．実際に回路を実現するボードの大きさは，$A_{\text{processing}}+A_{\text{memory}}$ に比例するであろうし，プロセッサの部分は入出力バンド幅に見合うだけのピンを持たなければならないであろう．

もちろん，ソーティング回路の良否は，漸近的な最適性だけで決まるものではない．技術者にとっては，実際のスピードと規模が重要なのである．本論文の計算モデルはこのような精密な解析ができるほど十分なものではないが，相対的な速度や規模についていくつかのことは言える．

最小の回路は明らかに単一プロセッサの $O(\lg N)$ のものである．しかも，この設計は，$O(N \lg N)$ ステップの逐次アルゴリズムを使うなら，$AT_d^2$ について最適に近いのである．

さらに，高速化を図る場合には，($\lg N$) プロセッサ上のヒープソートが良い．これは単一プロセッサに較べ面積がほぼ $\lg N$ 倍である．また，入力をビット直列で逐次的に行う場合，最小の遅延（入力の最後のビットが入った後，直ちに出力が開始できる）で行える特徴を持つ．このような遅延に関する性質を持つのは，本論文で紹介したものの中では，他に ($N/2$) プロセッサ上のバブルソートだけであり，これ以外はすべて，最後の入力から最初の出力までの間に $O(\lg N)$ 以上の遅延が存在する．このヒープソートでは，各プロセッサに1つずつ記憶を持つので，入出力バンド幅は単位時間あたり $O(\lg N)$ ビットとなる．

($\lg N$) プロセッサ上のバイトニックソートも，ヒープソートと同じ面積および時間性能である．バイトニックソートの方が制御は簡単であり，メモリもシフトレジスタ型でよい．しかし，入出力バンド幅はヒープソートの方が小さい．

($\lg^2 N$) プロセッサ上のバイトニックソートは，($\lg N$) プロセッサ上のものに較べ，適度の$N$に対して面積が小さい．制御はきわめて簡単で，ほとんど比較-

表-2 他の効率の尺度

| | 設計 | 並列度 ($c$) | 処理面積 ($A_{processing}$) | 記憶面積 ($A_{memory}$) | 入出力バンド幅 |
|---|---|---|---|---|---|
| A | 単一プロセッサ | 1 | $\lg N$ | $N \lg N$ | 1 |
| B | ($\lg N$) プロセッサ上のピープソート | 1 | $\lg^2 N$ | $N \lg N$ | $\lg N$ |
| C | ($\lg N$) プロセッサ上のマージソート | 1 | $\lg^2 N$ | $N \lg N$ | $\lg N$ |
| D | ($\lg N$) プロセッサ上のバイトニックソート | 1 | $\lg^2 N$ | $N \lg N$ | $\lg^2 N$ |
| E | ($\lg^2 N$) プロセッサ上のバイトニックソート | 2 | $\lg^3 N$ | $N \lg N$ | $\lg^3 N$ |
| F | ($\sqrt{N \lg N}$) プロセッサ上のバイトニックソート | 1 | $N \lg N$ | $N \lg N$ | $\lg N$ |
| G | ($N/2$) 比較器上のバブルソート | 1 | $N \lg N$ | $N \lg N$ | $\lg N$ |
| H | $N$プロセッサの2次元格子上のバイトニックソート | 1 | $N \lg^2 N$ | $N \lg N$ | $\sqrt{N} \lg N$ |
| I | $N$プロセッサのシャフル交換網上のバイトニックソート | 1 | $N^2/\lg^2 N$ | $N \lg N$ | $N/\lg^2 N$ |
| J | $N$プロセッサの PCCC 上のバイトニックソート | 1 | $N \lg N$ | $N \lg N$ | $\sqrt{N \lg N}$ |
| | | 1 | $N^2/\lg^4 N$ | $N \lg N$ | $N/\lg^2 N$ |
| K | ($N \lg^2 N$) 比較器上のバイトニックソート | $\lg N$ | $N^2$ | $N \lg^2 N$ | $N$ |
| L | $N^2$ 比較器上のバブルソート | $N/\lg N$ | $N^2$ | $N^2$ | $N$ |
| M | $N^2$ プロセッサ上のランクソート | 1 | $N^2 \lg^2 N$ | $N \lg N$ | $N$ |

交換モジュールそのものでよい．ただ $(1/2)\times(\lg N)\times(\lg N+1)$ 個の語並列シフトレジスタ記憶を使うのが最大の欠点である．

($\sqrt{N \lg N}$) プロセッサ上のバイトニックソートは，入出力バンド幅は ($\lg^2 N$) プロセッサより小さくできるが，制御が複雑で，面積は大きくなる．

($N/2$) プロセッサ上のバブルソートは，表-1，表-2に現われない2つの大きな特長を持っている．1つは，制御が簡単で，$N$が小さな所では面積が小さくてすむ．2つ目は，オンラインで挿入や削除ができる“自己ソート記憶”として使えることである．(単一プロセッサや ($\lg N$) プロセッサ上のヒープソートもこのように使える．)

$N$プロセッサの2次元格子上のバイトニックソートは，線形時間以下で問題を解くために考えられたものであるが，大きな面積を必要とする．各プロセッサは複雑なアルゴリズムを実行しなければならず，入出力バンド幅も大きくなる．しかし，既存の格子型結合のマルチプロセッサ上での応用を考えると，これらは必ずしも重大な問題とはいえない．

表-1，2 の I，J，K の3つの設計は，バイトニックソートの完全並列化である．シャフル交換網は CCC や PCCC より制御が簡単な分だけ面積が小さくなる可能性がある．しかし，CCC や PCCC は規則的な結合であるので，配線がし易い．これらは，($N \lg^2 N$) 比較器上のものより少しだけ総面積の漸近的な値が小さい．しかし，($N \lg^2 N$) 比較器上のものは制御が簡単なので，$N<2^{20}$ くらいではこちらの方が小さくなると思われる．$N^2$ 比較器上のバブルソートは，漸近的な面積・時間性能には見劣りがするが，$N$の小さな所では有効なことがある．これは，語長が可変で長いような時にも使える．関係データベースの質問処理等で，$N=16$ 程度の所では有用であろう[52]．

最後の $N^2$ プロセッサ上のランクソートは，その遅延が最小である点が理論的に重要である．$T_d=O(\lg N)$ でかつ，これより小さな面積を持つような設計は知られていない．

**追記** 最近，任意長の語のソーティングについて

$AT_d^2$ の上界と下界のギャップを埋める仕事が発表されている[50),51),53)].

## 参 考 文 献


1) Ajtai, M., Komlós, J. and Szemerédi, E.: An $O(n \log n)$ Sorting Network, in *Proc.* 15th Annu. ACM Symp. Theory Comput., pp. 1-9 (Apr. 1983).
2) Armstrong, P. K.: U. S. Patent 4 131 947, (Dec. 26, 1978).
3) Armstrong, P. K. and Rem, M.: A Serial Sorting Machine, Comput. Elect. Eng., Pergamon, Vol. 9 (Mar. 1982).
4) Bilardi, G., Pracchi, M. and Preparata, F. P.: A Critique of Network Speed in VLSI Models of Computation, IEEE J. Solid-State Circuits, Vol. SC-17, pp. 696-702 (Aug. 1982).
5) Brent, R. and Kung, H. T.: The Area-time Complexity of Binary Multiplication, J. Ass. Comput. Mach., Vol. 28, pp. 521-534 (July 1981).
6) Chan, T. C., Eswaren, K. P., Lum, V. Y. and Tung, C.: Simplified Odd-Even Sort Using Multiple Shift-Register Loops, Int. J. Comput. Inform. Sci., Vol. 7, pp. 295-314 (Sept. 1978).
7) Chazelle, B. and Monier, L.: Towards More Realistic Models of Computation for VLSI, in Proc. 11th Annu. ACM Symp Theory Comput., pp. 209-213 (Apr. 1979).
8) Chung, K.-M., Luccio, F. and Wong, C. K.: On the Complexity of Sorting in Magnetic Bubble Memory Systems, IEEE Trans. Comput., Vol. C-29, pp. 553-562 (July 1980).
9) Despain, A.: Very Fast Fourier Transform Algorithms for Hardware Implementation, IEEE Trans. Comput., Vol. C-28, pp. 333-341 (May 1979).
10) Dohi, Y., Suzuki, A. and Matsui, N.: Hardware Sorter and Its Application to Data Base Machine, in Proc. 9th Annu. Symp. Comput. Arch. (ACM SIGARCH Newsletter), Vol. 10, pp. 218-225 (Apr. 1982).
11) Evans, S. A.: Scaling $I^2L$ for VLSI, IEEE J. Solid-State Circuits, Vol. SC-14, pp. 318-326 (Apr. 1979).
12) Guibas L. J. and Liang, F. M.: Systolic Stacks, Queues, and Counters, in Proc. 1982 Conf. Advanced Res. VLSI, Massachusetts Inst. Technol., Cambridge, pp. 155-164 (Jan. 1982).
13) Hong, J.-W. and Kung, H. T.: I/O Complexity: The Red-Blue Pebble Game, in Proc. 13th Annu. ACM Symp. Theory Comput., pp. 326-333 (May 1981).
14) Hong, J.-W.: On Similarity and Duality of Computation, Peking Munic. Computing Center, Peking, People's Repub. China, Unpublished (1981).
15) Kedem, G.: A First in, First Out and a Priority Queue, Dep. Comput. Sci., Univ. Rochester, Rochester, NY, Tech. Rep. 90 (Mar. 1981).
16) Kedem, Z. M. and Zorat, A.: Replication of Inputs May Save Computational Resources in VLSI, in Proc. 22nd Symp. Found. Comput. Sci., IEEE Comput. Soc. (Oct. 1981).
17) Ketchen, M. B.: AC Powered Josephson Miniature System, in Proc. 1980 Int. Conf. Circuits Comput., IEEE Comput. Soc. pp. 874-877 (Oct. 1980).
18) Kleitman, D., Leighton, F. T., Lepley, M. and Miller, G. L.: New Layouts for the Shuffle-Exchange Graph, in Proc. 13th Annu. ACM Symp. Theory Comput, pp. 334-341 (May 1981).
19) Knuth, D. E.: The Art of Computer Programming, Vol. 3: Sorting and Searching. Reading, MA: Addison-Wesley (1973).
20) ——,: Big Omicron and Big Omega and Big Theta, SIGACT News, Vol. 8, pp. 18-24 (Apr.-June 1976).
21) Kumar, M. and Hirschberg, D. S.: An Efficient Implementation of Batcher's Odd-Even Merge Algorithm and Its Application in Parallel Sorting Schemes, IEEE Trans, Comput., Vol. C-32, pp. 254-264 (Mar. 1983).
22) Leighton, F. T.: New Lower Bound Techniques for VLSI, in Proc. 22nd Symp. Found. Comput. Sci., IEEE Comput. Soc. (Oct. 1981).
23) ——,: Layouts for the Shuffle-exchange Graph and Lower Bound Techniques for VLSI, Ph. D. Dissertation MIT/LCS/TR-724, M. I. T. Lab. for Comput. Sci., Massachusetts Inst. Technol., Cambridge (June 1982).
24) Leiserson, C. E.: Area-efficient VLSI Computation, Ph. D. Dissertation CMU-CS-82-108, Comput. Sci. Dep., Carnegie-Mellon Univ., Pittsburgh, PA (Oct. 1981).
25) Lipton R. J. and Sedgewick, R.: Lower bounds for VLSI, in Proc. 13th Annu. ACM Symp. Theory Comput., pp. 300-307 (May 1981).
26) Mead, C. and Conway, L.: Introduction to VLSI Systems. Reading, MA: Addison-Wesley (1980).
27) Moravec, H. P.: Fully Interconnecting Multiple Computers with Pipelined Sorting Nets, IEEE Trans. Comput., Vol. C-28, pp. 795-798 (Oct. 1979).

28) Mukhopadhyay, A. and Ichikawa, T.: An $n$-step Parallel Sorting Machine, Dep. Comput. Sci., Univ. Iowa, Iowa City, Tech. Rep. 72-03 (1972).
29) Mukhopadhyay, A.: WEAVESORT—A New Sorting Algorithm for VLSI, Univ. Central Florida, Orlando, Tech. Rep. 53-81 (1981).
30) Muller, D. E. and Preparata, F. P.: Bounds to Complexities of Networks for Sorting and for Switching, J. Ass. Comput. Mach., Vol. 22, pp. 195-201 (Apr. 1975).
31) Nassimi, D. and Sahni, S.: Bitonic Sort on a Mesh-connected Parallel Computer, IEEE Trans. Comput., Vol. C-28, pp. 2-7 (Jan. 1979).
32) Nath, D., Maheshwari, S. N. and Bhatt, P. C. P.: Efficient VLSI Networks for Parallel Processing Based on Orthogonal Trees, Dep. Elec. Eng., Indian Inst. Technol., New Delhi, India, 1981, Unpublished.
33) Preparata, F. and Vuillemin, J.: The Cube-connected Cycles: A Versatile Network for Parallel Computation, in Proc. 20th Annu. Symp. Found. Comput. Sci., IEEE Comput. Soc., pp. 140-147 (Oct. 1979).
34) Reif, J. H. and Valiant, L. G.: A Logarithmic Time Sort for Linear Size Networks, in Proc. 15th Annu. ACM Symp. Theory Comput., pp. 10-17 (Apr. 1983).
35) Rosenberg, A. L.: Three-dimensional VLSI, I: A Case Study, in Proc. CMU Conf. VLSI, Comput. Sci. Press, pp. 69-79 (Oct. 1981).
36) Savage, J.: Planar Circuit Complexity and the Performance of VLSI Algorithms, in VLSI Systems and Computations, H. T. Kung, B. Sproull, and G. Steele, Eds. Woodland Hills, CA: Comput. Sci. Press, (Oct. 1981).
37) ——,: Area-time Tradeoffs for Matrix Multiplication and Related Problems in VLSI Models, J. Comput. Syst. Sci., Vol. 22, pp. 230-242, (Apr. 1981).
38) Seitz, C. L.: Self-timed VLSI Systems, in Proc. Caltech Conf VLSI, Dep. Comput. Sci., California Inst. Technol., Pasadena, pp. 345-356 (Jan. 1979).
39) Stone, H.: Parallel Processing with the Perfect Shuffle, IEEE Trans. Comput., Vol. C-20, pp. 153-161 (Feb. 1971).
40) Tanaka, Y., Nozaka, Y. and Masuyama, A.: Pipeline Searching and Sorting Modules as Components of Data Flow Database Computer, in Proc. Int. Fed. Inform. Processing, pp. 427-432 (Oct. 1980).
41) Thompson, C. D. and Kung, H. T.: Sorting on a Mesh-connected Parallel Computer, Commun. Ass. Comput. Mach. Vol. 20, pp. 263-271 (Apr. 1977).
42) Thompson, C. D.: A Complexity Theory for VLSI, Ph. D. Dissertation CMU-CS-80-140, Comput. Sci. Dep., Carnegie-Mellon Univ., Pittsburgh, PA (Aug. 1980).
43) ——,: Fourier Transforms in VLSI, IEEE Trans. Comput., Vol. C-32, pp. 1047-1057 (Nov. 1983).
44) Thompson, C. D. and Angluin, D.: On $AT^2$ Lower Bounds for Sorting, Unpublished Manuscript (May 1983).
45) Todd, S.: Algorithm and Hardware for a Merge Sort Using Multiple Processors, IBM J. Res. Develop., Vol. 22, pp. 509-517 (Sept. 1978).
46) Vuillemin, J.: A Combinational Limit to the Computing Power of VLSI Circuits, IEEE Trans. Comput., Vol. C-32, pp. 294-300 (Mar. 1983).
47) Winslow, L. E. and Chow, Y.-C.: Parallel Sorting Machines: Their Speed and Efficiency, in Proc. AFIPS 1981 Nat. Comput. Conf., Fall 1981 pp. 163-165.
48) Yao, A. C.: Some Complexity Questions Related to Distributive Computing, in Proc. 11th Annu. ACM Symp. Theory Comput. pp. 209-213 (May 1979).
49) Yasuura, H., Takagi, N. and Yajima, S.: The Parallel Enumeration Sorting Scheme for VLSI, IEEE Trans, Comput., Vol. C-31, No. 12, pp. 1192-1201 (Dec. 1982).
50) Leighton, T.: Tight Bounds on the Complexity of Parallel Sorting, in Proc. 16th Annu. ACM Symp. Theory Comput., pp. 71-80 (May 1984).
51) Siegel, A.: Tight Area Bounds and Provably Good $AT^2$ Bounds for Sorting Circuits, Report of Courant Institute, New York Univ., No. 122 (1984).
52) Kim, W., Gajski, D. and Kuck, D. J.: A Parallel Pipelined Relational Query Processor, ACM. Trans. on Database Systems, Vol. 9, No. 2, pp. 214-242 (June 1984).
53) Bilardi, Gianfranco: The Area-time Complexity of Sorting, ACT-52 (Ph. D. Dissertation), Coordinated Science Laboratory, University of Illinois at Urbana-Champaign (Dec. 1984).

(昭和60年1月9日受付)