Reprint from Computing Reviews

Michael C. Loui
University of Illinois at Urbana-Champaign

As a service to our readers, *SIGACT News* has reached an agreement to reprint reviews that originally appeared in *Computing Reviews* of books and articles of interest to the theoretical computer science community. *Computing Reviews* is a monthly journal that publishes critical reviews on a broad range of computing subjects, including models of computation, formal languages, computational complexity theory, analysis of algorithms, and logics and semantics of programs. ACM members can receive a subscription to *Computing Reviews* for $32 per year by writing to ACM headquarters.

AJTAI, M.; KOMLOS, J.; AND SZEMEREDI, E.
Sorting in c log n parallel steps.
Combinatorica **3,** 1 (Jan. 1983), 1-19.

LEIGHTON, TOM.
Tight bounds on the complexity of parallel sorting.
IEEE Trans. Comput. **C-34,** 4 (April 1985), 344-354.

BILARDI, GIANFRANCO; AND PREPARATA, FRANCO P.
A minimum area VLSI network for O(log n) time sorting.
IEEE Trans. Comput. **C-34,** 4 (April 1985), 336-343.

COLE, RICHARD.
Parallel merge sort.
SIAM J. Comput. **17,** 4 (Aug. 1988), 770-785.

Early research into sorting focused on *in situ* comparison-based sorting algorithms. Such an algorithm is said to be *oblivious* (a term which can be traced, in a different context, to Paterson) if the sequence of cells accessed is dependent on the number of cells but not on their contents. An oblivious algorithm has a particularly elegant hardware implementation called a *sorting network,* which consists of a circuit constructed without fan-out from building blocks called *comparators*. Comparators have two inputs and two outputs, and swap the values on the inputs if they are out of order, passing them through unchanged otherwise. The *depth* (number of layers) of the circuit is equal to the number of phases of nonoverlapping comparisons, and the *size* (number of comparators) of the circuit is equal to the number of comparisons used in the oblivious sorting algorithm. Note that size is bounded above by $n/2$ times the depth.

The most widely known oblivious *in situ* sorting algorithm is bubblesort, which has depth and size $O(n^2)$. Bose and Nelson [1] produced a sorting network of size $O(n^{\log_2 3})$, and Floyd and Knuth produced one of size $O(n^{1+c/\log^{0.5} n})$ [2]. Batcher [3] discovered the *odd-even* and *bitonic* sorting networks, which have depth $O(\log^2 n)$ and remain practical even for small values of n (see Knuth [4]). A slight improvement to the constant multiple of the lower-order terms in the size can be made at a great cost in depth [5]. To date the best sorting networks for reasonable values of n are essentially constructed from Batcher's sorting networks by terminating the recursion early and substituting a hand-constructed network with a small constant number of inputs. The standard lower-bound argument for comparator-based sorting gives size and depth lower bounds of $\Omega(n \log n)$ and $\Omega(\log n)$, respectively. Batcher's sorting networks are larger and deeper by a factor of $O(\log n)$. It was unknown for 15 years whether Batcher's techniques were asymptotically optimal.

Ajai, Komlós, and Szemerédi

It was not until 1983 that Ajtai, Komlós, and Szemerédi laid this open problem to rest in the first paper under review by constructing a sorting network, subsequently dubbed the *AKS sorting network* in their honor, that asymptotically matches the naive lower bounds. Their paper contains a fairly detailed account of the construction, expressed as an oblivious sorting algorithm. The algorithm also appears (in abbreviated form) expressed as a sorting network in Gabber and Galil [6]. This version is attributed to Joel Spencer. Both papers are rather difficult to read. Paterson produced a more accessible proof in 1983 and presented it at the Complexity Theory meeting in Oberwolfach in 1983 and at the Columbia University Theory Day in 1984. From there Paterson's description was rapidly assimilated into the oral tradition, and many authors (including myself) wrote up drafts in the form of lecture notes. After much urging from colleagues, Paterson produced the canonical version of his proof [7], soon to be published in *Algorithmica* [8].

The essence of the construction by Ajtai, Komlós, and Szemerédi, as improved by Paterson, is as follows. Expander graphs of constant degree are used to construct constant-depth comparator networks called γ-halvers. These have the property that all but $\gamma n/2$ of the smallest $n/2$ inputs appear in the leftmost $n/2$ outputs (symmetrically for the larger half of the inputs). Constant-depth γ-halvers are used to construct constant-depth separators. These have the additional property of being ε-nearsorters, that is, for all $1 \leq k \leq n/2$, all but $\lceil \varepsilon k \rceil$ of the smallest k inputs appear in the leftmost k outputs (symmetrically for the largest k). For the sake of clarity the algorithm is described on binary tree of separators with n leaves.

Initially the values to be sorted are placed in the root. Each node in parallel separates its values, passes a constant fraction of its leftmost and rightmost elements to its parent, divides the remainder in half, and sends the leftmost half to its left child and the rightmost half to its right child. This process is repeated indefinitely. The aim is to end up with the values in sorted order in the leaf nodes. A *stranger* is an item that has strayed from the direct path from the root to its leaf node. Strangers are created because, for example, the γ-halving may cause up to a γ fraction of the smallest items in a node to be sent to its right child instead of its left child. The intuition behind the algorithm is that the separation puts most of the strangers at the ends, whence they get passed up and eventually rejoin the correct path. If the constants involved in the algorithm (including γ and ε) are chosen suitably, it terminates correctly in $O(\log n)$ steps. Since each step involves only a separation (which can be done in constant depth) and a reassignment of values in bags (which, since they are only abstractions, have no physical realization), it is easy to transform the algorithm into a sorting network of asymptotically optimal size and depth.

The AKS sorting network uses bounded-degree expander graphs. Uniform AKS sorting networks can be based on explicit constructions of expander graphs [9]. The degree of explicit constructions of expander graphs, though constant, is inferior to that of nonuniform expander graphs, which can be shown to exist by the probabilistic method (a random bipartite graph with sufficiently large but constant degree is with high probability an expander graph). Thus the constant multiple in the depth bound of uniform AKS sorting networks is greatly inferior to that of nonuniform AKS sorting networks, which at 6100 according to Paterson, is still too large to be practical for reasonable values of n.

Leighton

One advantage of Batcher's sorting networks is that, due to their highly regular recursive construction, they can readily be implemented on various standard constant-degree parallel computers of n processors without asymptotic time loss. This implies that constant-degree parallel computers with p processors can simulate p-processor abstract computers, such as the PRAM, with a slowdown factor of $O(\log^2 p)$ (see Parberry [10] for survey). This is yet another example of the robustness of the class NC. One way of reducing the slowdown factor without requiring extra processors would be to find a way of implementing the AKS sorting network on a constant-degree network of n processors.

In the second paper under review, Leighton establishes this result as the corollary of an important general principle; any n-input sorting network of depth d can be transformed into an n-processor, $O(d)$-time, degree-3 parallel sorting algorithm. Leighton uses an elegant algorithm called *columnsort,* which consists of routing steps and four invocations of $\log n$ copies of the sorting network with $n/\log n$ inputs. The number of invocations can be reduced to one invocation of $\log n$ copies with $n/\log n$ inputs, and one invocation of $n/\log^2 n$ copies with $\log^2 n$ inputs, plus merging [11].

Thompson demonstrated that the AT^2 complexity of a VLSI sorter of area A and time T for sorting n words of length $(1+\varepsilon)\log n$, for $\varepsilon > O$, is $\Omega(n^2\log^2 n)$ (using a model with input data that are read once from specified input ports and with unit wire-delay) under the assumption that all of the bits of the same word enter the circuit at the same point [12]. Leighton was able to remove this assumption. His result implies that, for AT^2 optimal sorting circuits, the only interesting range for T is from $\Omega(\log n)$ to $O(\sqrt{n \log n})$. Leighton matches his lower bound with an AT^2-optimal VLSI implementation of the AKS sorting network for all T within this range.

Bilardi and Preparata

In the third paper under review, Bilardi and Preparata describe an AT^2-optimal VLSI sorter for all $T \in \Omega(\log n) \cap O(\log^3 n)$. This, combined with their earlier result [13], duplicates Leighton's AT^2 optimal sorting circuits. Although their design uses more processors than Leighton's, it is based on a clever divide-and-conquer approach using a hybrid of classical VLSI designs rather than the AKS sorting network and thus does not suffer from the latter's legacy of large constant multiples.

Cole

The PRAM, which consists of a collection of sequential processors that communicate by reading and writing into a shared memory, is a popular abstract model of parallel computation. The acronyms CREW, EREW, and CRCW describe whether read and write conflicts are allowed (for example, CREW stands for Concurrent Read, Exclusive Write). In the CRCW model, write conflicts are typically handled by a scheme that blocks all but one of the processors contending to write in the same cell. An EREW PRAM can simulate the AKS sorting network directly, using only n processors and $O(\log n)$ time.

In the fourth paper under review, Cole gives an n-processor, $O(\log n)$-time EREW PRAM sorting algorithm with a much lower constant multiple. For the sake of clarity the algorithm is described on a binary tree with n leaves. Initially, the values to be sorted are placed in the leaves. Each node in parallel merges the elements in its children. The aim is to end up with the values in sorted order in the root. Extra information, in the form of a sorted sample, is also passed up the tree. This enables the $O(\log n)$ time merging process in each node to be pipelined. The algorithm is initially described on CREW PRAM. Conversion to an EREW PRAM increases the constant in the time bound slightly. The algorithm is also used to obtain a sublogarithmic-time, polynomial CRCW PRAM sorting algorithm.

–Ian Parberry, University Park, PA

REFERENCES

[1] Bose, R. C. and Nelson, R. J. A sorting problem. *J. ACM* **9** (1962), 282-296. See *CR* **3,** 5 (Sept.-Oct. 1962), Rev. 2,620.

[2] Floyd, R. W. and Knuth, D. E. Improved constructions for the Bose-Nelson sorting problem (preliminary report). *Not. AMS* **14** (1967), 283.

[3] Batcher, K. E. Sorting networks and their applications. In *Proceedings of the AFIPS Spring Joint Computer Conference,* Atlantic City, NJ, April 30–May 2, 1968 (A. S. Hoagland, Chair). Thompson Book Co., Washington, DC, 1968, 307-314.

[4] Knuth, D. E. *The art of computer programming, vol. 3–sorting and searching.* Addison-Wesley, Reading, MA, 1973. See *CR* **14,** 8 (Aug. 1973), Rev. 25,533.

[5] Van Voorhis, D. C. An economical construction for sorting networks. In *Proceedings of the AFIPS National Computer Conference,* Chicago, IL, May 6-10, 1974 (S. S. Yau, Chair). AFIPS Press, Montvale, NJ, 1974.

[6] Ajtai, M.; Komlós, J.; and Szemerédi, E. An $O(n \log n)$ sorting network. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing* (Boston, MA, April 25-27, 1983). ACM, New York, 1983, 1-9.

[7] Paterson, M. S. *Improved sorting networks with* $O(\log n)$ depth. Research Report 89, Dept. of Computer Science, Univ. of Warwick, Warwick, UK, Jan. 1987.

[8] Paterson, M. S. Improved sorting networks with $O(\log n)$ depth. *Algorithmica,* to appear.

[9] Gabber, O. and Galil, Z. Explicit constructions of linear size superconcentrators. In *Proceedings of the Twentieth Annual IEEE Symposium on Foundations of Computer Science* (Oct. 1979). IEEE, New York, 1979, 364-370.

[10] Parberry, I. Some practical simulations of impractical parallel computers. *Parallel Comput.* **4,** 1 (1987), 93-101.

[11] Parker, B. PRAM simulations on bounded degree networks. Ph.D. Thesis, Dept. of Computer Science, Pennsylvania State Univ., State College, PA, Dec. 1988.

[12] Thompson, C. D. The VLSI complexity of sorting, *IEEE Trans. Comput.* **C-32** (Dec. 1983), 1171-1184.

[13] Bilardi, G. and Preparata, F. An architecture for bitonic sorting with optimal VLSI performance. *IEEE Trans. Comput.* **C-33** (July 1984), 646-651.

Copyright 1989, Association for Computing Machinery, Inc. Reprinted by permission from *Computing Reviews,* vol. 30, no. 11, pp. 578-580, Nov. 1989, Rev. no. 8909-0816.