Parallel Sorting of n-strings in Kn Time

G. A. De Biase
Dipartimento di Informatica
Università di Roma la Sapienza
Via Salaria 113, 00198 Roma, Italy

A. Massini
Dipartimento di Informatica
Università di Roma la Sapienza
Via Salaria 113, 00198 Roma, Italy

Abstract *In this work, using a suitable permutation network of size $N = 2^n$, followed by a hyperconcentrator network, M n-strings ($2 \leq M \leq N$) are sorted in Kn time. This is possible by using the binary string values as destination addresses on the permutation network. Since $M \leq N$, $N - M$ outputs do not receive strings. If a hyperconcentrator network follows, the $N - M$* holes *are removed and the sorted sequence of M values is obtained. Using a high efficiency permutation network on which a probabilistic routing algorithm runs, and a suitable hyperconcentrator network, the time of the overall operation is Kn or $K(\log_2 N)$ where K is a very moderate constant.*

Keywords: Parallel sorting, probabilistic routing algorithm, permutation networks, concentrator networks

1 Introduction

Sorting is a key task because it can often represent an intermediate step in many different applications. For this reason it is important to study algorithms or devices capable to perform the sorting of a set of elements in a time as short as possible. The classical Batcher sorting algorithm/network [1] is an example of the correspondence between the algorithm and its wired (network) realization. Batcher sorting algorithm is based on comparisons between pairs of elements chosen according to their reciprocal positions in the network. Batcher sorting network is realized connecting comparison cells according to the scheme defined by the algorithm. The Batcher parallel sorting is deterministic and has $O(\log_2^2 M \log_2 n)$ time complexity, where M is the number of elements to be sorted, n is the n-string length and $N = 2^n$ (for the determination of this time complexity see Ref. [6]).

In this work a probabilistic parallel sorting procedure with Kn time complexity which uses a permutation network followed by a hyperconcentrator is presented.

2 The sorting procedure outline

The sorting procedure is based on the use of a permutation network followed by a hyperconcentrator as shown in Fig. 1.

A non-blocking permutation network of size N, $\mathbf{P_N}$, always allows all one-to-one connections between N inputs channels and N output channels. In other words $\mathbf{P_N}$ allows a permutation of its outputs on its inputs in a time T_P

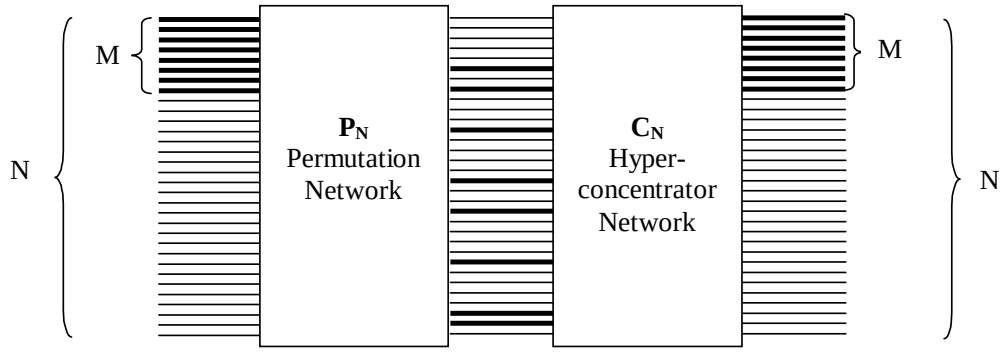

Figure 1: A permutation network $\mathbf{P_N}$ and a concentrator network $\mathbf{C_N}$ cascaded. $N = 2^n$ is the size of the two networks and M is the number of the n-strings to be sorted.

(depending on the depth of the network and on the routing algorithm time complexity).

A hyperconcentrator network of size N, $\mathbf{C_N}$, [11] has N inputs and N outputs and can establish disjoint connections from any subset of k inputs, for any $1 \leq k \leq N$, to the first k outputs. The hyperconcetrator eliminates *holes* in a time T_C.

As shown in Fig. 1, values to be sorted (n-strings) $v_i, i = 0, ..., M$ are presented on M ($0 \leq M \leq N$) of the N inputs of $\mathbf{P_N}$. n-strings can assume all values between 0 and $N = 2^n$ and, if these values are used as output addresses on $\mathbf{P_N}$, they are sorted and scattered on M outputs. Hence $N - M$ holes are present among the M sorted values. To eliminate holes, and generate the sequence of sorted values, the hyperconcentrator network acts. The overall time of the sorting procedure T_S is:

$$T_S = T_P + T_C$$

3 An implementation of the parallel sorting

Following the outline presented in the previous section (use of $\mathbf{P_N}$ and $\mathbf{C_N}$), the choice of the requested networks is critical for its influence on the overall time complexity T_S of the sorting procedure.

In the past an asymptotically nonblocking permutation network has been presented, see Ref. [4, 5]. It consists of three cascaded stacks of banyan networks on which a parallel probabilistic algorithm runs. In this permutation network the faults of the probabilistic algorithm can become negligible. The time complexity T_P of this device is $3 \log_2 N$, where N is the network size. In another work [2] Cormen and Leisersons present an hyperconcen-

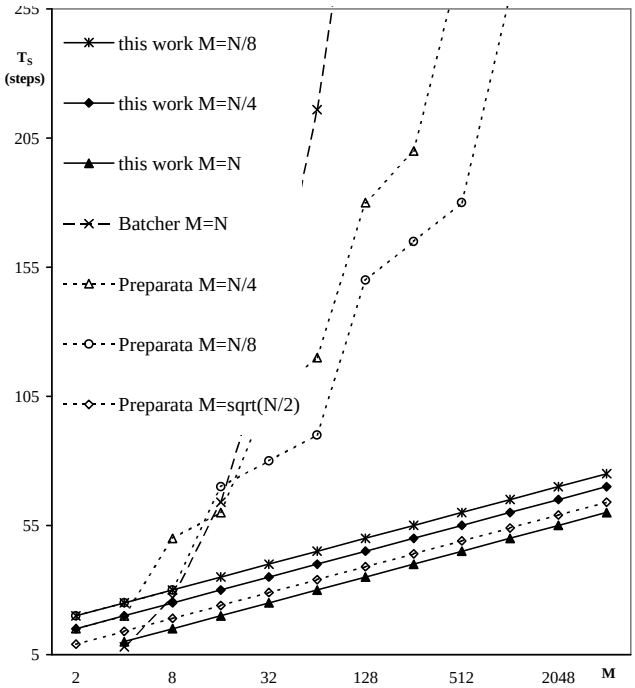

Figure 2: Behavior comparison for some M/N ratios of the time T_S (in steps) versus the number of elements to be sorted M in the cited sorting procedures.

trator which consists of many merge boxes of various sizes suitably connected. On this network a deterministic algorithm runs and the holes among strings are eliminated with time complexity $T_C = 2\log_2 N$.

Using an asymptotically nonblocking permutation network followed by the Cormen and Leiserson hyperconcentrator the total sorting time is:

$$T_S = T_P + T_C = 5\log_2 N$$

Because the length of n-strings is $n = \log_2 N$ the overall complexity T_S is

$$T_S = 5n$$

4 Discussion and conclusions

Preparata proposed a deterministic algorithm to sort M elements in $O(\frac{\log_2 M \log_2 N}{\log_2(M/N)})$ steps if $M \leq N/4$ on N-node hypercubic networks (see Ref. [9, 7, 8]). This algorithm, for small values of M ($M \leq \sqrt{N/2}$), considerably improves the time of the Batcher sorting network which requires $O(\log_2^2 M \log_2 n)$ steps.

In Fig. 2 a comparison among times of Preparata sorting algorithm, for $M = N/4, M = N/8, M = \sqrt{N/2}$ elements, times of Batcher sorting algorithm, and times of the sorting algorithm proposed in this work, for $M = N, M = N/4, M = N/8$ elements, is shown.

As one can see, times of the proposed parallel sorting procedure is better than all other ones except the case $M \leq \sqrt{N/2}$ of the Preparata procedure.

The fact that on the permutation network runs a probabilistic routing algorithm is irrelevant because its faults become negligible under suitable conditions [4] (fault occurrences can become smaller than the MTBF ratio, in this case the difference between the deterministic and probabilistic case disappears, see Ref. [5, 10]).

References

[1] K. E. Batcher, "Sorting networks and their application", *Proceeding of Spring Joint Computer Conference*, 307-314, 1968.

[2] T. H. Cormen and C.E. Leiserson, "A Hyperconcentrator Switch for Routing Bit-Serial Messages", *J. Par. Distr. Comput.*, 10, 193-204, 1990

[3] R. E. Cypher and C. G. Plaxton, "Deterministic Sorting in Nearly Logarithmic Time on the Hypercube and Related Computers", Proc. ACM Symposium on Theory of Computing, 193-203, 1990.

[4] G. A. De Biase, C. Ferrone and A. Massini, "An $O(\log_2 N)$ depth asymptotically nonblocking self-routing permutation network", *IEEE Trans. Comp.*, 44, 1047-1050, 1995

[5] G. A. De Biase and A. Massini, "A Virtually Nonblocking Self-Routing Permutation Network which Routes Packets in $O(\log_2 N)$ Time", *Telecomm. Sys.*, 10, 135-147, 1998

[6] D. M. Koppelman and A. Y. Oruç, "A Self-Routing Permutation Network", *J. Par. Distr. Comput.*, 10, 140-151, 1990.

[7] F. T. Leighton, Introduction to Parallel Algorithms and Architectures: Arrays, Trees and Hypercubes, Morgan Kaufmann Publ., San Mateo, CA, 1992.

[8] D. Nassimi and S. Sanhi, "Parallel Permutation and Sorting Algorithms and a New Generalized Connection Network", *J. of the ACM*, 29, 642-667, 1982.

[9] F. Preparata, "New Parallel Sorting Schemes", *IEEE Trans. Comp.*, C-27, 7, 669-673, 1978.

[10] T. H. Szymansky and C. Fang, "Randomized Routing of Virtual Connections in Essentially Nonblocking $\log N$ Depth Networks", *IEEE Trans. Comm.*, 43, 2521-2531, 1995

[11] L. G. Valiant, "Graph-theoretic properties in computational complexity", *J. Comput. System Sci.*, 13, 278-285, 1976