

Parallel Algorithms to Set Up the Benes Permutation Network

DAVID NASSIMI AND SARTAJ SAHNI, MEMBER, IEEE

Abstract—A parallel algorithm to determine the switch settings for a Benes permutation network is developed. This algorithm can determine the switch settings for an N input/output Benes network in $O(\log^2 N)$ time when a fully interconnected parallel computer with N processing elements is used. The algorithm runs in $O(N^{1/2})$ time on an $N^{1/2} \times N^{1/2}$ mesh-connected computer and $O(\log^4 N)$ time on both a cube connected and a perfect shuffle computer with N processing elements. It runs in $O(k \log^3 N)$ time on cube connected and perfect shuffle computers with $N^{1+1/k}$ processing elements.

Index Terms—Benes permutation network, complexity, cube connected computer, fully connected SIMD computer, mesh-connected computer, parallel algorithm, perfect shuffle computer, set-up algorithm.

I. INTRODUCTION

THE Benes permutation network $B(n)$ is a network with $N = 2^n$ inputs and outputs. The network is capable of delivering at its output end any permutation of its N inputs. This network has been proposed for use in telephone networks, self-repairing multiprocessors [10], as an interconnection network in parallel computers [12], etc. It forms the heart of a common generalized connection network [7].

Fig. 1 gives a schematic of $B(n)$ and Fig. 2 gives the two possible states of a switch. Observe that there are $N/2$ switches at the input stage and only $N/2 - 1$ switches at the output stage. So the network of Fig. 1 incorporates the switch saving scheme suggested by Waksman [9]. From Fig. 1 it follows that $B(n)$ has $2n - 1$ switch stages and $N \log N - N + 1$ switches (note that $B(1)$ is just a single switch). Let the switch stages be numbered 0 through $2n - 2$ left to right (see Fig. 1).

Waksman [9] has shown that the network $B(n)$ is capable of delivering at its output end any permutation of its N inputs. His proof of this fact is constructive and it directly leads to a switch setting algorithm. This algorithm runs in $O(N \log N)$ time on a single processor computer. Thus, the set-up time for the network is much larger than the network delay [which is $O(\log N)$]. One cannot set up the Benes network in less than $O(N \log N)$ time using a single processor as the network has $O(N \log N)$ switches. In order to obtain a set-up algorithm of complexity comparable to the delay time, it is therefore necessary to consider parallel algorithms. An alternative is to make the network self-setting, as has been done by Nassimi and Sahni [5]. Their self-routing scheme, however, does not

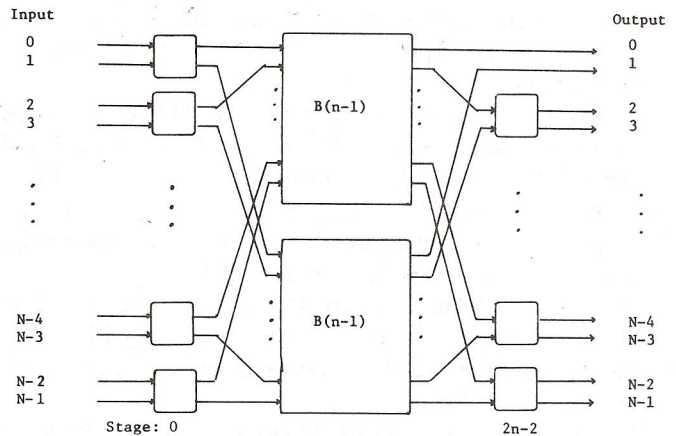


Fig. 1. The Benes permutation network $B(n)$, $N = 2^n$.



Fig. 2. States of a binary switch. (a) State 0. (b) State 1.

work for all permutations. Another alternative is to precompute and store the switch settings for some permutations. This requires $O(N \log N)$ bits of storage per permutation. Furthermore, the precomputation approach is not suitable for dynamic situations.

In this paper we develop a parallel set-up algorithm that is significantly faster than the single processor algorithm of Waksman. The complexity of our algorithm depends on the parallel computer model and the number of processing elements available. We consider four computer models. All four are SIMD (single instruction stream, multiple data stream) type computers (see Flynn [13]). An SIMD computer consists of some number M of processing elements (PE's), each having some local memory. The PE's are indexed 0 through $M - 1$. We shall refer to the i th PE as PE(i). The PE's are synchronized and operate under the control of a single instruction stream. An enable/disable mask may be used to select a subset of PE's that will perform the instruction to be executed at any given time. All enabled PE's perform the same instruction. The four SIMD models we shall consider differ in the way the PE's are interconnected. The PE interconnection pattern is important as PE's can communicate only through the interconnection network. The four PE interconnection networks defining our four SIMD models are as follows.

1) *Completely Interconnected Computer (CIC)*: In a CIC model every pair of PE's is directly connected. The time needed to transfer data from one PE to another is $O(1)$.

Manuscript received March 23, 1981; revised August 4, 1981. This work was supported in part by the National Science Foundation under Grant 78-15455.

D. Nassimi is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60201.

S. Sahni is with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.

