SPEEDING UP RANDOM ACCESS MACHINES BY FEW PROCESSORS

(Preliminary Version)

Friedhelm Meyer auf der Heide

FB 20-Informatik, Johann Wolfgang Goethe Universität Frankfurt
6000 Frankfurt a.M.
Fed. Rep. of Germany 1)

Abstract: Sequential and parallel random access machines (RAMs, PRAMs) with arithmetic operations + and - are considered. PRAMs may also multiply with constants. These machines work on integer inputs. It is shown that, in contrast to bit orientated models as Turing machines or log-cost RAMs, one can in many cases speed up RAMs by PRAMs with few processors. More specifically, a RAM without indirect addressing can be uniformly sped up by a PRAM with q processors by a factor $\frac{(\log \log q)^2}{\log q}$. A similar result holds for nonuniform speed ups of RAMs with indirect addressing. Furthermore, certain networks of RAMs (such as k-dimensional grids) with q processors can be sped up significantly with only $q^{1+\epsilon}$ processors. Nonuniformly, the above speed up can even be achieved for arbitrary bounded degree networks (including powerful networks such as permutation networks or Cube-Connected Cycles), if only few input variables are allowed. It is previously shown by the author, that the speed ups for RAMs are almost best possible.

1) This research was done at the IBM Research Laboratory, San Jose, CA, USA.
Introduction

Parallel random access machines (PRAMs) are a widely accepted model of parallel computation. Many algorithms are known in this model which show that sometimes surprisingly strong speed ups of certain sequential algorithms are possible. On the other hand many problems look inherently sequential, i.e. there seems to be no significant speed up possible, at least when only few processors are allowed. It is known [PR] that with many, namely $2^t$ processors, one can speed up $t$ steps of a Turing machine by a factor $\frac{\log \log \log t}{\log t}$, but no speed ups are known with $\text{poly}(t)$ processors.

In this paper we show that such speed ups are possible for RAMs with operations $+$ and $-$, uniform cost measure, and inputs given integer by integer, not bit by bit. We show that such RAMs without indirect addressing (storage addresses are functions in the number, but not in the values of the inputs) can be sped up by PRAMs with $q$ processors by a factor $\frac{\log \log q}{\log q}$, if the PRAMs can multiply with constants. Here uniform means that, if the RAM computes $f: N^* \to N^*$ in time $T(n)$ ($n = \#(\text{input variables})$), then the PRAM needs time $T(n) \frac{\log \log q}{\log q}$. We also can speed up RAMs with indirect addressing in a similar way, but only to the expense of nonuniformity, i.e. we need a new PRAM for each new number $n$ of input variables. This is one of the examples where fast algorithms can be designed to the expense of nonuniformity. Other, more surprising examples are the fast nonuniform algorithms for the knapsack or traveling salesman problem [M1], [M2], or the fast nonuniform simulations of probabilistic by deterministic computations [BG], [A], [M3].

We furthermore show that even certain networks of RAMs can be sped up. If $q$ RAMs are for example connected to a $k$-dimensional grid, a PRAM with $q^k$ processors can uniformly speed it up by a factor $\frac{\log \log q}{\log q}$, where $d = \frac{1}{k+1}$. Nonuni-